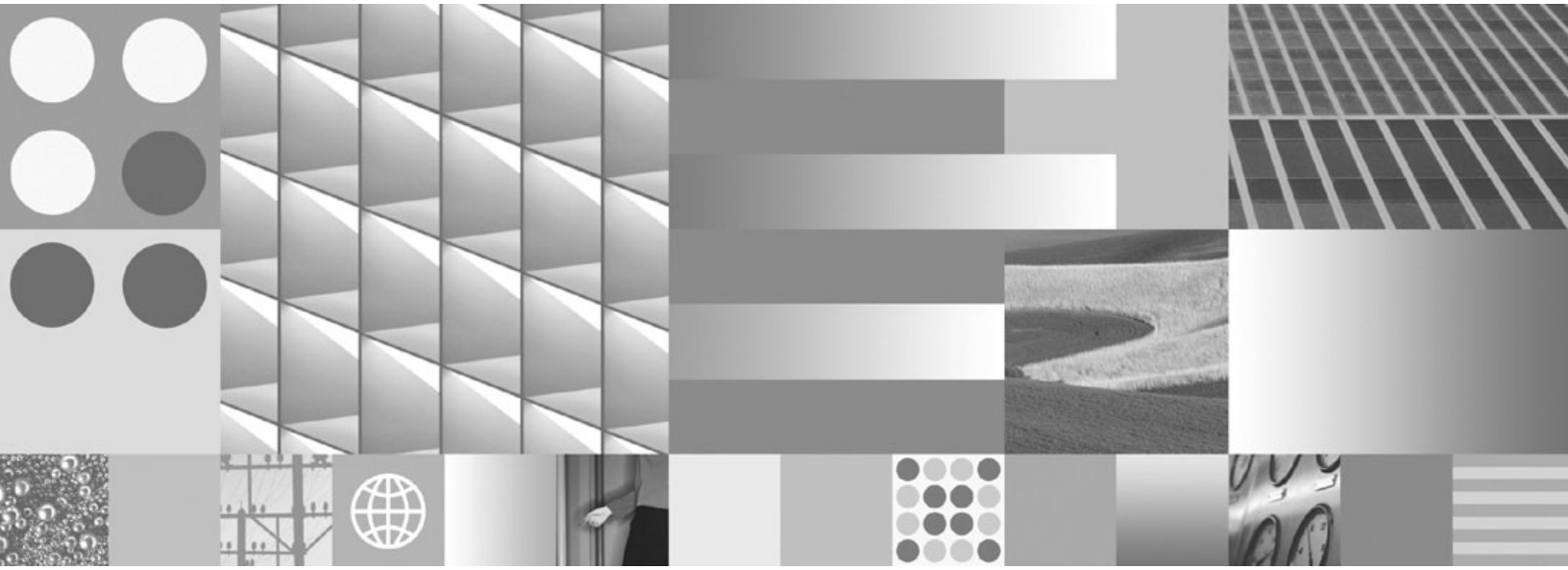
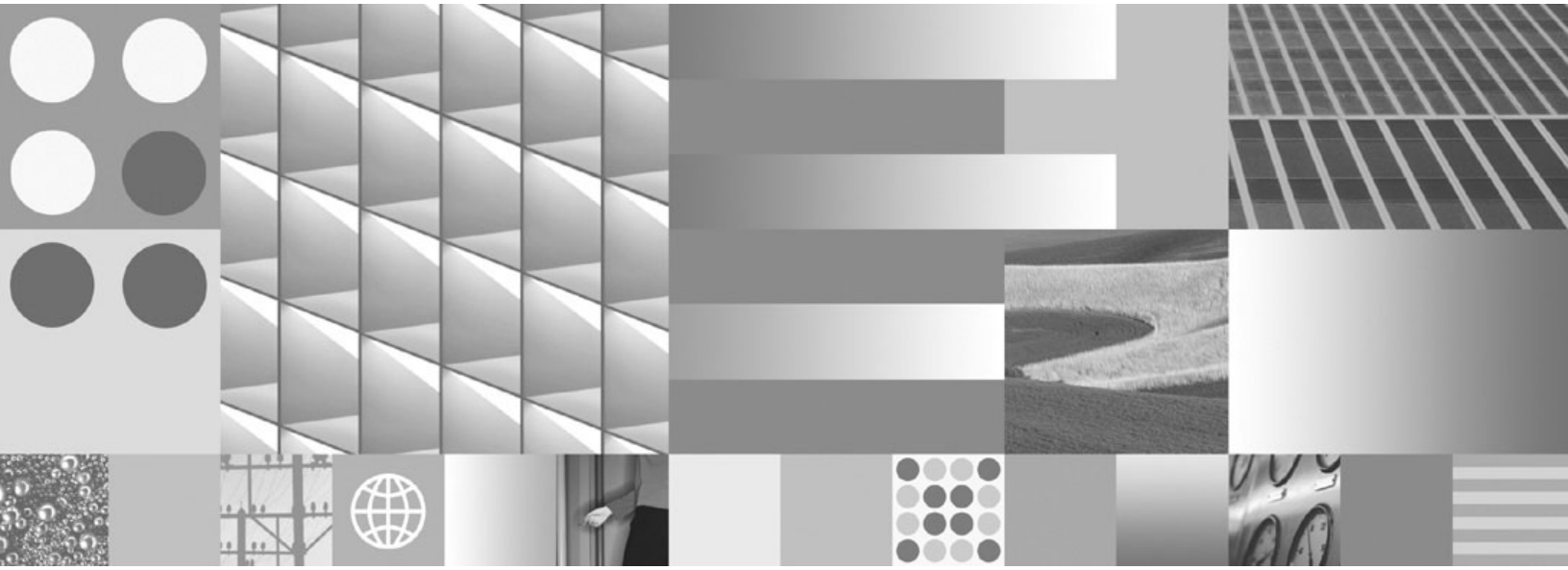
Linux, UNIX 및 Windows용
IBM DB2 9.7



SQL 참조서, 볼륨 1

SA30-3956-00

Linux, UNIX 및 Windows용
IBM DB2 9.7



SQL 참조서, 볼륨 1

SA30-3956-00

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 1191 페이지의 부록 O『주의사항』의 정보를 읽으십시오.

개정판 주의사항

이 문서에는 IBM에서 소유하고 있는 정보가 있습니다. 이는 라이센스 계약에 따라 제공한 것이며 저작권의 보호를 받습니다. 이 책의 정보에는 제품 보증이 포함되지 않으며, 이 매뉴얼에서 제공된 어떠한 문장도 이와 같이 해석할 수 없습니다.

온라인으로 IBM 서적을 주문하거나 로컬 IBM 담당자를 통해 서적을 주문할 수 있습니다.

- 온라인으로 서적을 주문하려면 IBM Publications Center(www.ibm.com/shop/publications/order)로 이동하십시오.
- 로컬 IBM 담당자를 찾으려면 IBM Directory of Worldwide Contacts(www.ibm.com/planetwide)로 이동하십시오.

미국 또는 캐나다의 DB2 Marketing and Sales에서 DB2 서적을 주문하려면 1-800-IBM-4YOU (426-4968)로 전화하십시오.

IBM은 귀하가 IBM으로 보낸 정보를 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 사용하거나 배포할 수 있습니다.

목차

이 책에 대한 정보

두 볼륨으로 구성된 SQL 참조서는 Linux®, UNIX® 및 Windows®용 DB2® 데이터베이스에서 사용되는 SQL 언어를 정의합니다. 다음 내용으로 구성됩니다.

- 관계형 데이터베이스 개념, 언어 요소, 함수 및 쿼리 양식에 관한 정보(볼륨 1)
- SQL문의 구문 및 시맨틱에 관한 정보(볼륨 2)

이 책의 사용자

이 책은 구조적 쿼리 언어(SQL)를 사용하여 데이터베이스에 액세스하는 사람들을 위한 것입니다. 기본적으로는 프로그래머와 데이터베이스 관리자용이지만 명령행 처리기(CLP)를 통해 데이터베이스에 액세스하는 사용자가 이용할 수 있습니다.

이 책은 자습서가 아닌 참조서입니다. 응용프로그램을 작성할 것이라고 가정하므로 데이터베이스 관리 프로그램의 전체 기능을 설명합니다.

이 책의 구성

SQL 참조서의 첫 번째 볼륨에는 관계형 데이터베이스 개념, 언어 요소, 기능 및 쿼리 양식에 관한 정보가 포함됩니다. 이 볼륨의 특정 장 및 부록을 간단하게 설명합니다.

- 『개념』은 관계형 데이터베이스와 SQL의 기본 개념을 설명합니다.
- 『언어 요소』는 다수의 SQL문에 공통적인 SQL 및 언어 요소의 기본 구문을 설명합니다.
- 『함수』는 SQL 집계와 스칼라 함수의 사용 예, 구문 다이어그램, 시맨틱 설명, 규칙을 포함합니다.
- 『프로시저』는 프로시저의 사용 예, 구문 다이어그램, 시맨틱 설명, 규칙을 포함합니다.
- 『SQL 쿼리』는 다양한 쿼리 양식을 설명합니다.
- 『SQL 및 XML 한계』는 SQL 한계를 나열합니다.
- 『SQLCA(SQL 통신 영역)』는 SQLCA 구조를 설명합니다.
- 『SQLDA(SQL 디스크립터 영역)』는 SQLDA 구조를 설명합니다.
- 『시스템 카탈로그 뷰』는 시스템 카탈로그 뷰를 설명합니다.
- 『페더레이티드 시스템』은 페더레이티드 시스템에 대한 옵션 및 유형 맵핑을 설명합니다.
- 『샘플 데이터베이스』는 여러 예에서 사용된 테이블을 포함한 샘플 데이터베이스를 소개합니다.

- 『예약 스키마 이름 및 예약어』는 IBM® SQL 및 ISO/ANSI SQL2003 표준용 예약 스키마 이름 및 예약어를 포함합니다.
- 『트리거 및 참조 제한조건 사이의 상호작용 예』는 트리거와 참조 제한조건의 상호작용을 설명합니다.
- 『Explain 테이블』은 Explain 테이블을 설명합니다.
- 『Explain 레지스터 값』은 CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값의 상호작용과 이 값과 PREP 및 BIND 명령과의 상호작용을 설명합니다.
- 『예외 테이블』은 SET INTEGRITY 명령문과 사용된 사용자가 작성한 테이블에 관한 정보를 포함합니다.
- 『루틴에서 허용된 SQL문』은 다른 SQL 데이터 액세스 컨텍스트와 함께 루틴에서 실행되도록 허용된 SQL문을 나열합니다.
- 『컴파일된 명령문에서 호출된 CALL』은 컴파일된 명령문에서 호출될 수 있는 CALL 문을 설명합니다.

구문 다이어그램을 읽는 방법

구문은 다음과 같은 구조를 사용하여 설명합니다.

라인의 경로를 따라서 왼쪽에서 오른쪽으로 및 위에서 아래로 구문 다이어그램을 읽으십시오.

▶▶── 기호는 구문 다이어그램의 시작을 표시합니다.

──▶ 기호는 구문이 다음 라인에서 계속됨을 표시합니다.

▶── 기호는 구문이 이전 라인에서 계속됨을 표시합니다.

──▶◀ 기호는 구문 다이어그램의 끝을 표시합니다.

구문 조각은 ├── 기호로 시작하고 ──┤ 기호로 끝납니다.

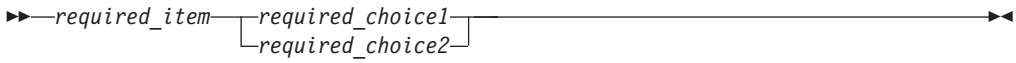
필수 항목은 수평선(주 경로)에 나타납니다.

선택 항목은 기본 경로 아래에 나타납니다.

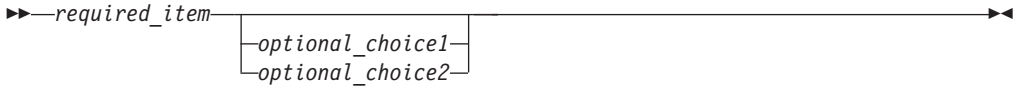
선택적 항목이 주 경로 위에 나타나는 경우 해당 항목은 실행 시 적용되지 않으며 판독성을 위해서만 사용됩니다.

둘 이상의 항목에서 선택할 수 있는 경우 해당 항목은 스택으로 나타납니다.

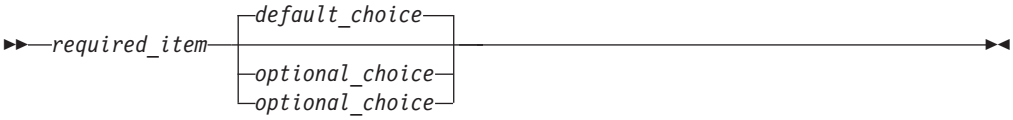
항목 중 하나를 *반드시* 선택해야 하는 경우 스택 중 하나의 항목이 주 경로에 나타납니다.



항목 중 하나의 선택이 선택적인 경우 전체 스택이 주 경로 아래에 나타납니다.



항목 중 하나가 디폴트인 경우 해당 항목은 주 경로 위에 나타나고 나머지 선택사항은 아래에 표시됩니다.



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

반복 화살표가 쉼표를 포함하는 경우 반복되는 항목을 쉼표로 구분해야 합니다.

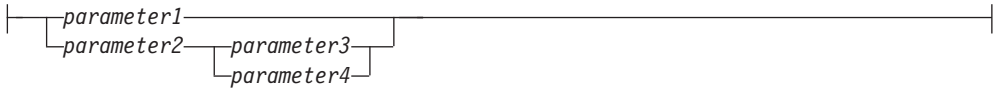
스택 위에 있는 반복 화살표는 스택된 항목에서 둘 이상을 선택하거나 단일 선택사항을 반복할 수 있음을 표시합니다.

키워드는 대문자로 나타납니다(예: FROM). 키워드는 표시된 대로 정확하게 입력해야 합니다. 변수는 소문자로 나타납니다(예: column-name). 구문에서 사용자가 제공하는 이름이나 값을 나타냅니다.

구두점, 괄호, 산술 연산자 또는 기호 같은 기타가 표시되는 경우 이들을 구문의 일부로 입력해야 합니다.

가끔 단일 변수가 구문의 더 큰 조각을 나타냅니다. 예를 들어 다음 다이어그램에서 parameter-block 변수는 **parameter-block**으로 레이블되는 전체 구문 조각을 나타냅니다.

parameter-block:



『큰 글머리표』(●) 사이에 발생하는 인접한 세그먼트는 임의의 순서로 지정할 수 있습니다.

위의 다이어그램은 item2 및 item3이 어느 순서로든 지정할 수 있음을 표시합니다. 다음의 두 가지가 모두 유효합니다.

```
required_item item1 item2 item3 item4
required_item item1 item3 item2 item4
```

이 매뉴얼에서 사용된 규칙

오류 조건

괄호 안의 오류와 연관된 SQLSTATE를 나열하여 매뉴얼 텍스트에 오류 조건을 표시합니다. 예를 들어, 다음과 같습니다.

시그니처가 중복되면 SQL 오류가 리턴됩니다(SQLSTATE 42723).

강조표시 규칙

이 책에서 사용되는 규칙은 다음과 같습니다.

굵게	명령, 키워드 및 이름이 시스템에서 사전 정의된 기타 항목을 표시합니다.
기울임꼴	다음 중 하나를 표시합니다. • 사용자가 제공해야 되는 이름이나 값(변수) • 일반 강조 • 새 용어 소개 • 다른 정보 소스에 대한 참조

유니코드 데이터를 설명하는 규칙

특정 유니코드 코드 포인트를 참조하는 경우 U+*n*으로 표시합니다. 여기서, *n*은 숫자 0-9 및 대문자 A-F를 사용하는 4자리-6자리의 16진수입니다. 코드 포인트가 4자리 미만의 16진수가 아니면 앞자리에 오는 0은 생략합니다. 예를 들어, 공백 문자는 U+0020으로 표시합니다. 대부분의 경우, *n* 값은 UTF-16BE 인코딩과 동일합니다.

관련 문서

다음은 응용프로그램을 준비할 때 유용한 책입니다.

- *Getting Started with Database Application Development*
  - 플랫폼 전제조건, 지원되는 개발 소프트웨어, 지원되는 프로그래밍 API의 장점과 제한사항을 포함한 DB2 응용프로그램 개발에 대한 개요를 제공합니다.
- *i5/OS용 DB2 SQL 참조서*
  - 이 책은 System i®에서 DB2 쿼리 관리 프로그램 및 SQL 개발 킷에서 지원하는 SQL을 정의합니다. 시스템 관리, 데이터베이스 관리, 응용프로그램 프로그래

밍 및 조작 태스크에 대한 참조 정보를 포함합니다. 이 매뉴얼에는 DB2를 실행하는 i5/OS® 시스템에서 사용되는 구문, 사용법, 키워드 및 각 SQL문의 예가 포함됩니다.

- *z/OS용 DB2 SQL 참조서*
  - 이 책은 z/OS®용 DB2에서 사용되는 SQL을 정의합니다. 이 책에서는 쿼리 양식, SQL문, DB2를 실행하는 z/OS 시스템용 SQL 프로시저 명령문, DB2 한계, SQLCA, SQLDA, 카탈로그 테이블 및 SQL 예약어를 제공합니다.
- *DB2 Spatial Extender 사용자 안내 및 참조서*
  - 이 책은 응용프로그램을 작성하여 GIS(Geographic Information System)를 생성하고 사용하는 방법에 대해 설명합니다. GIS 작성과 사용은 위치, 거리와 영역 내의 분산과 같은 정보를 얻기 위한 데이터 쿼리 및 자원을 포함한 데이터베이스 제공과 관련됩니다.
- *IBM SQL 참조서*
  - 이 책은 IBM의 데이터베이스 제품에 포함된 일반 SQL 요소 모두를 포함합니다. IBM 데이터베이스를 사용하여 포터블 프로그램 준비를 보조하는 한계와 규칙을 제공합니다. 이 매뉴얼은 SQL92E, XPG4-SQL, IBM-SQL 표준 및 IBM 관계형 데이터베이스 제품 간의 SQL 확장자 및 비호환 목록을 제공합니다.
- *미국 표준 X3.135-1992, 데이터베이스 언어 SQL*
  - SQL의 ANSI 표준 정의를 포함합니다.
- *ISO/IEC 9075:1992, 데이터베이스 언어 SQL*
  - 1992 SQL의 ISO 표준 정의를 포함합니다.
- *ISO/IEC 9075-2:2003, 정보 기술 -- 데이터베이스 언어 -- SQL -- 파트 2: 기초 (SQL/기초)*
  - 2003 SQL의 ISO 표준 정의 대부분을 포함합니다.
- *ISO/IEC 9075-4:2003, 정보 기술 -- 데이터베이스 언어 -- SQL -- 파트 4: Persistent Stored Modules (SQL/PSM)*
  - SQL 프로시저 제어 명령문의 2003 ISO 표준 정의를 포함합니다.

제 1 장 개념

데이터베이스

DB2 데이터베이스는 *관계형 데이터베이스*입니다. *데이터베이스*에는 서로 관련되어 있는 테이블의 모든 데이터가 저장됩니다. 데이터가 공유되고 중복이 최소화될 수 있도록 테이블 간에 관계가 설정됩니다.

*관계형 데이터베이스*는 테이블 세트로 취급되고 관계형 데이터 모델에 따라 조작되는 데이터베이스입니다. 관계형 데이터베이스에는 데이터 저장, 관리 및 액세스에 사용되는 오브젝트 세트가 들어 있습니다. 이러한 오브젝트에는 테이블, 뷰, 인덱스, 함수, 트리거 및 패키지가 있습니다. 오브젝트는 시스템을 통해 정의되거나(시스템 정의 오브젝트) 사용자를 통해 정의될 수 있습니다(사용자 정의 오브젝트).

*분산 관계형 데이터베이스*는 서로 연결된 여러 컴퓨터 시스템에 분산되어 있는 테이블 세트와 기타 오브젝트로 구성됩니다. 각 컴퓨터 시스템에는 해당 환경의 테이블을 관리하기 위한 관계형 데이터베이스 관리 프로그램이 있습니다. 데이터베이스 관리 프로그램은 지정된 데이터베이스 관리 프로그램이 다른 컴퓨터 시스템에서 SQL문을 실행할 수 있도록 서로 통신하고 협력합니다.

*파티션된 관계형 데이터베이스*는 다중 데이터베이스 파티션에서 데이터가 관리되는 관계형 데이터베이스입니다. 대부분의 SQL문은 이와 같이 데이터베이스 파티션에서 데이터가 분리되는 것을 명확히 알 수 있습니다. 그러나 일부 DDL(Data Definition Language) 명령문은 데이터베이스 파티션 정보를 고려합니다(예: CREATE DATABASE PARTITION GROUP). DDL은 데이터베이스에서 데이터 관계를 설명하는 데 사용되는 SQL문의 서브세트입니다.

*페더레이티드 데이터베이스*는 다중 데이터 소스(예: 개별 관계형 데이터베이스)에 데이터가 저장되어 있는 관계형 데이터베이스입니다. 데이터는 모두 단일 대형 데이터베이스에 있는 것처럼 표시되며 일반적인 SQL 쿼리를 통해 데이터에 액세스할 수 있습니다. 데이터의 변경사항은 명시적으로 해당 데이터 소스로 경로가 지정됩니다.

SQL

SQL은 관계형 데이터베이스에서 데이터를 정의하고 조작하기 위한 표준화된 언어입니다. 데이터 관계 모델에 따라, 데이터베이스는 테이블 세트로 처리되고, 관계는 테이블의 값으로 표시되며, 데이터는 하나 이상의 기본 테이블에서 파생될 수 있는 결과 테이블을 지정하여 검색합니다.

SQL문은 데이터베이스 관리 프로그램에서 실행됩니다. 데이터베이스 관리 프로그램의 기능 중 하나는 결과 테이블 스펙을 데이터 검색을 최적화하는 내부 조작 시퀀스로 변환하는 것입니다. 변환은 준비 및 바인딩의 두 단계로 발생합니다.

모든 실행 가능한 SQL문은 실행되기 전에 준비해야 합니다. 준비 결과는 실행 가능 또는 작동 가능 양식의 명령문입니다. SQL문을 준비하는 방법과 작동 가능 양식의 지속성으로 정적 SQL과 동적 SQL이 구별됩니다.

쿼리 및 테이블 표현식

*쿼리*는 특정 SQL문의 구성요소로, (임시) 결과 테이블을 지정합니다.

*테이블 표현식*은 단순 쿼리에서 임시 결과 테이블을 작성합니다. 절은 결과 테이블을 더욱 세분화합니다. 예를 들어, 쿼리로 테이블 표현식을 사용하여 몇몇 부서에서 모든 관리자를 선택하고, 그러한 관리자가 15년 이상의 작업 경력을 가지고 있고 뉴욕 지방 사무소에 위치되도록 지정할 수 있습니다.

*공통 테이블 표현식*은 복합 쿼리 내의 임시 뷰와 유사합니다. 쿼리 내의 다른 위치에서 참조하고 뷰 대신 사용할 수 있습니다. 복합 쿼리 내에서 특정의 공통 테이블 표현식을 사용할 때마다 동일한 임시 뷰를 공유합니다.

쿼리 내에서 공통 테이블 표현식의 반복적 사용을 통해 비행기 예약 시스템, BOM(bill of materials) 생성 프로그램 및 네트워크 계획과 같은 응용프로그램을 지원할 수 있습니다.

DB2 콜 레벨 인터페이스(CLI) 및 ODBC 개요

DB2 콜 레벨 인터페이스(CLI)(DB2 CLI)는 데이터베이스 서버의 DB2 제품군에 대해 IBM이 지원하는 호출 가능한 SQL 인터페이스입니다. 이 인터페이스는 함수 호출을 사용하여 동적 SQL문을 함수 인수로 전달하는 관계형 데이터베이스 액세스에 사용되는 'C' 및 'C++' 응용프로그램 프로그래밍 인터페이스입니다.

DB2 CLI 인터페이스를 사용하여 다음 IBM Data Server 데이터베이스에 액세스할 수 있습니다.

- DB2 버전 9 Linux, UNIX 및 Windows용
- DB2® Universal Database™(DB2 UDB) 버전 8 이상Linux, UNIX 및 Windows용
- OS/390® 및 z/OS용 DB2 Universal Database 버전 8 이상
- IBM i 5.4용 DB2 이상

DB2 CLI는 Embedded 동적 SQL을 대체하지만 Embedded SQL과는 달리 호스트 변수 또는 프리컴파일러가 필요하지 않습니다. 응용프로그램은 데이터베이스 각각에 대해

컴파일하지 않고 다양한 데이터베이스에 대해 실행할 수 있습니다. 응용프로그램은 런타임 시 프로시저 호출을 사용하여 데이터베이스에 연결하고 SQL문을 발행하며 데이터 및 상태 정보를 검색합니다.

DB2 CLI 인터페이스는 Embedded SQL에서 사용할 수 없는 많은 기능을 제공합니다. 예를 들어, 다음과 같습니다.

- CLI는 DB2 계열 사이에 일관성 있는 데이터베이스 카탈로그를 쿼리하는 방법을 지원하는 함수 호출을 제공합니다. 이로서 특정 데이터베이스 서버에 맞춰 조정해야 하는 카탈로그 쿼리를 작성해야 하는 필요성이 감소합니다.
- CLI는 커서를 통해 스크롤할 수 있는 기능을 제공합니다.
  - 하나 이상의 행만큼 앞으로 이동합니다.
  - 하나 이상의 행만큼 뒤로 이동합니다.
  - 첫 번째 행에서 하나 이상의 행만큼 앞으로 이동합니다.
  - 마지막 행에서 하나 이상의 행만큼 뒤로 이동합니다.
  - 커서에서 이전에 저장된 위치로부터
- CLI를 사용하여 작성된 응용프로그램에서 호출된 스토어드 프로시저는 해당 프로그램에 결과 세트를 리턴할 수 있습니다.

DB2 CLI는 Microsoft® ODBC(Open Database Connectivity) 스펙 및 SQL/CLI의 국제 표준을 기반으로 합니다. 산업 규격을 준수하고, 이미 해당 데이터베이스 인터페이스에 익숙한 응용프로그램 프로그래머가 더 짧은 기간에 학습을 마칠 수 있도록 DB2 콜 레벨 인터페이스(CLI)에 대한 기초로 이들 스펙을 선택했습니다. 또한 응용프로그램 프로그래머가 특히 DB2 기능을 사용하는 데 유용하도록 DB2에 따라 다른 확장자를 일부 추가했습니다.

DB2 CLI 드라이버는 ODBC 드라이버 관리자를 통해 로드되는 경우 ODBC 드라이버 기능도 수행합니다. ODBC 3.51을 준수합니다.

DB2 CLI 백그라운드 정보

DB2 CLI 또는 호출 가능한 SQL 인터페이스를 이해하기 위해서는 이들의 기본을 이해하고, 기존 인터페이스와 비교하는 것이 도움이 됩니다.

X/Open사와 SQL Access Group은 함께 *X/Open 콜 레벨 인터페이스(CLI)*라고 하는 호출 가능한 SQL 인터페이스의 스펙을 개발했습니다. 이 인터페이스는 응용프로그램이 임의의 데이터베이스 공급자의 프로그래밍 인터페이스에 대해 독립되도록 하여 응용프로그램의 이식성을 증가시키기 위해 개발되었습니다. 대부분의 X/Open 콜 레벨 인터페이스(CLI) 스펙은 ISO 콜 레벨 인터페이스(CLI) 국제 표준(ISO/IEC 9075-3:1995 SQL/CLI)의 일부로 채택되었습니다.

Microsoft에서는 X/Open CLI의 예비 초안을 기본으로 하여 Microsoft 운영 체제용 ODBC(Open Database Connectivity)라고 하는 호출 가능한 SQL 인터페이스를 개발했습니다.

ODBC 스펙에는 연결 요청 시 제공되는 데이터 소스(데이터베이스 이름)를 기준으로, 드라이버 관리자가 데이터베이스별 ODBC 드라이버를 런타임 시 동적으로 로드하는 운영 환경도 포함됩니다. 응용프로그램은 각 DBMS의 라이브러리가 아니라 단일 드라이버 관리자 라이브러리에 직접 링크됩니다. 드라이버 관리자는 런타임 시 응용프로그램의 함수 호출을 중개하고 함수 호출이 DBMS에 맞는 ODBC 드라이버에 방향지정되는지 확인합니다. ODBC 드라이버 관리자에는 ODBC 특정 함수에 대한 정보만 있으므로 ODBC 환경에서는 DBMS 특정 함수에 액세스할 수 없습니다. 이스케이프 절이라고 하는 메커니즘을 통해 DBMS에 맞는 동적 SQL문이 지원됩니다.

Microsoft 운영 체제에서만 ODBC가 구현되는 것은 아니며 여러 플랫폼에서 다양하게 구현이 가능합니다.

ODBC 드라이버 관리자는 DB2 CLI 로드 라이브러리를 ODBC 드라이버로 로드할 수 있습니다. ODBC 응용프로그램을 개발하려면 ODBC 소프트웨어 개발 킷을 확보해야 합니다. Windows 플랫폼에서는 MDAC(Microsoft Data Access Components) SDK의 일부로서 ODBC SDK가 사용 가능하며, http://www.microsoft.com/data/에서 다운로드할 수 있습니다. Windows가 아닌 플랫폼에서는 기타 공급자가 ODBC SDK를 제공합니다. DB2 서버에 연결할 ODBC 응용프로그램 개발 시, Call Level Interface Guide and Reference, Volume 1 및 Call Level Interface Guide and Reference, Volume 2(DB2에 맞는 확장자 및 진단 정보에 대한 정보 참조용)와 함께 Microsoft에 있는 ODBC Programmer's Reference 및 SDK 안내서를 사용하십시오.

DB2 CLI에서 직접 작성된 응용프로그램은 DB2 CLI 로드 라이브러리에 바로 링크됩니다. DB2 CLI에는 DB2에 맞는 함수 외에도 여러 ODBC 및 ISO SQL/CLI에 대한 지원이 포함됩니다.

다음 DB2 기능은 ODBC와 DB2 CLI 응용프로그램 모두에서 사용 가능합니다.

- 2바이트(그래픽) 데이터 유형
- 스토어드 프로시저
- 분산 작업 단위(DUOW), 2단계 커미트
- Compound SQL
- 사용자 정의 유형(UDT)
- 사용자 정의 함수(UDF)

IBM Data Server용 Java 응용프로그램 개발

DB2 및 IBM Informix® Dynamic Server(IDS) 데이터베이스 시스템은 Java™로 작성된 애플릿과 클라이언트 응용프로그램에 대한 드라이버를 지원합니다.

JDBC, SQL 또는 pureQuery를 사용하여 DB2 및 IDS 데이터베이스 시스템에 있는 데이터에 액세스할 수 있습니다.

JDBC

JDBC는 Java 응용프로그램이 관계형 데이터베이스에 액세스하는 데 사용하는 API입니다. JDBC용 IBM Data Server 지원을 통해 로컬 DB2, IDS 데이터 또는 리모트 관계형 데이터에 액세스하는 Java 응용프로그램을 DRDA®를 지원하는 서버에 작성할 수 있습니다.

SQLJ

SQLJ는 Java 응용프로그램에서 Embedded 정적 SQL 관련 지원을 제공합니다. 처음에는 IBM, Oracle 및 Tandem에서 동적 SQL JDBC 모델을 정적 SQL 모델로 보완하기 위해 SQLJ를 개발했습니다.

일반적으로 DB2에 연결하기 위해 Java 응용프로그램에서는 동적 SQL용 JDBC 및 정적 SQL용 SQLJ를 사용합니다.

IDS에 연결하기 위해 JDBC 또는 SQLJ 응용프로그램에서 SQL문을 동적으로 실행합니다.

SQLJ를 JDBC와 상호운영할 수 있으므로 응용프로그램에서 JDBC와 SQLJ를 동일한 작업 단위(UOW)에서 사용할 수 있습니다.

pureQuery

pureQuery는 높은 성능의 데이터 액세스 플랫폼으로서 간편하게 데이터 액세스 개발, 최적화, 보안 및 관리를 수행할 수 있습니다. 다음으로 이루어져 있습니다.

- 사용을 간편하게 하고 우수 사례 사용을 간소화하기 위해 빌드된 API(Application Programming Interface)
- Java 및 SQL 개발용으로 IBM Optim Development Studio에서 사용 가능한 개발 도구
- 데이터베이스 액세스 최적화 및 보안과 관리 태스크 간소화를 위해 IBM Optim pureQuery Runtime에서 사용 가능한 런타임

pureQuery를 사용하여, 해당 데이터가 데이터베이스에 있는지 또는 JDBC DataSource 오브젝트에 있는지 여부에 관계 없이 관계형 데이터를 오브젝트로 처리하는 Java 응용프로그램을 작성할 수 있습니다. 사용자의 응용프로그램에서 메모리에 있는 Java 콜렉

션에 저장된 오브젝트를 관계형 데이터처럼 처리할 수도 있습니다. 관계형 데이터 또는 Java 오브젝트를 쿼리하거나 갱신하려면 SQL을 사용하십시오.

pureQuery에 대한 자세한 정보는 통합 데이터 관리 정보 센터를 참조하십시오.

스키마

*스키마*는 이름 지정된 오브젝트의 콜렉션입니다. 스키마를 통해 해당 오브젝트를 논리적으로 그룹화할 수 있습니다. 스키마는 이름 규정자이기도 합니다. 스키마를 사용함으로써 여러 오브젝트에 동일한 자연 이름을 사용할 수 있고 해당 오브젝트를 명확하게 참조할 수 있습니다.

예를 들어, 'INTERNAL' 및 'EXTERNAL'이라는 스키마 이름을 사용하면 두 개의 서로 다른 SALES 테이블(INTERNAL.SALES, EXTERNAL.SALES)을 쉽게 구별할 수 있습니다.

또한 스키마를 사용하면 이름 스페이스가 충돌하는 일 없이 다중 응용프로그램이 단일 데이터베이스에 데이터를 저장할 수 있습니다.

스키마는 *XML 스키마*와 구별되며, 이와 혼동하면 안 됩니다. XML 스키마는 XML 문서 컨텐츠의 유효성을 확인하고 구조에 대해 설명하는 표준입니다.

스키마에는 테이블, 뷰, 별칭, 트리거, 함수, 패키지 및 기타 오브젝트가 포함될 수 있습니다. 스키마 자체가 데이터베이스 오브젝트입니다. CREATE SCHEMA문을 사용하여 명시적으로 스키마를 작성하며 현재 사용자 또는 지정된 권한 부여 ID가 스키마 소유자로 기록됩니다. 사용자에게 IMPLICIT_SCHEMA 권한이 있는 경우에는 기타 오브젝트가 작성될 때 내재적으로 작성될 수도 있습니다.

*스키마 이름*은 두 부분으로 된 오브젝트 이름에서 상위 순서 파트로 사용됩니다. 오브젝트 작성 시 스키마 이름을 사용하여 구체적으로 오브젝트를 규정하는 경우 오브젝트가 해당 스키마에 지정됩니다. 오브젝트 작성 시 스키마 이름을 지정하지 않는 경우에는 CURRENT SCHEMA 특수 레지스터에 지정된 디폴트 스키마 이름이 사용됩니다.

예를 들어, DBADM 권한을 가진 사용자가 다음과 같이 사용자 A가 사용할 C라는 스키마를 작성합니다.

```
CREATE SCHEMA C AUTHORIZATION A
```

그러면 사용자 A가 다음 명령문을 실행하여 스키마 C에 X라는 테이블을 작성할 수 있습니다(사용자 A에게 CREATETAB 데이터베이스 권한이 있는 경우).

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

일부 스키마 이름은 예약되어 있습니다. 예를 들면, SYSIBM 스키마에 속하는 내장 함수와 SYSFUN 스키마에 속하는 사전 설치된 사용자 정의 함수(UDF)가 있습니다.

데이터베이스 작성 시 RESTRICTIVE 옵션을 사용하여 작성하지 않을 경우 모든 사용자가 IMPLICIT_SCHEMA 권한을 갖습니다. 이 권한이 있으면 사용자가 아직 존재하지 않는 스키마 이름을 사용하여 오브젝트를 작성할 때마다 내재적으로 스키마를 작성합니다. 스키마가 내재적으로 작성되면 CREATEIN 특권이 부여되어 모든 사용자가 이 스키마에서 기타 오브젝트를 작성할 수 있습니다. 내재적으로 작성된 스키마에서도 별명, 구별 유형, 함수 및 트리거와 같은 오브젝트 작성 기능을 사용할 수 있습니다. 내재적으로 작성된 스키마에 대한 디폴트 특권을 통해 이전 버전과 호환이 가능합니다.

PUBLIC에서 IMPLICIT_SCHEMA 권한이 취소되는 경우 CREATE SCHEMA문을 사용하여 명시적으로 스키마를 작성하거나 IMPLICIT_SCHEMA 권한이 부여된 사용자(예: DBADM 권한을 가진 사용자)가 내재적으로 스키마를 작성할 수 있습니다. PUBLIC에서 IMPLICIT_SCHEMA 권한이 취소되면 스키마 이름 사용에 대한 제어가 증가하지만 기존 응용프로그램에서 오브젝트를 작성하려 할 경우 권한 부여 오류가 발생할 수 있습니다.

스키마에도 특권이 있어 스키마 소유자가 스키마에서 오브젝트를 작성, 변경, 복사 및 삭제(drop)할 특권을 갖는 사용자를 제어할 수 있습니다. 이는 데이터베이스의 오브젝트 서브세트 처리를 제어할 방법을 제공합니다. 처음에 스키마 소유자에게는 스키마에 대한 모든 해당 특권이 부여되고, 다른 사용자에게 특권을 부여하는 기능도 주어집니다. 내재적으로 작성된 스키마는 시스템이 소유하며 모든 사용자에게는 처음에 해당 스키마에서 오브젝트를 작성할 특권이 부여됩니다. ACCESSCTRL 또는 SECADM 권한을 가진 사용자는 모든 스키마에 대해 사용자가 보유하고 있는 특권을 변경할 수 있습니다. 따라서 내재적으로 작성된 스키마를 포함하여 모든 스키마에서 오브젝트를 작성, 변경, 복사 및 삭제(drop)하기 위해 액세스하는 것을 제어할 수 있습니다.

테이블

테이블은 데이터베이스 관리 프로그램에서 유지보수하는 논리적 구조입니다. 테이블은 컬럼과 행으로 구성됩니다.

모든 컬럼과 행의 교차에 *값*이라고 부르는 특정 데이터 항목이 있습니다. *컬럼*은 동일한 유형 또는 부속 유형 중 하나의 값 세트입니다. *행*은 *n*번째 값이 테이블의 *n*번째 컬럼의 값이 되도록 정렬되는 값의 시퀀스입니다.

응용프로그램은 행이 테이블에 채워지는 순서를 판별할 수 있지만, 행의 실제 순서는 데이터베이스 관리 프로그램에 의해 판별되며 일반적으로 제어할 수 없습니다. 다차원 클러스터링(MDC)이 어느 정도 클러스터링을 제공하지만 행 사이의 실제 순서 지정을 제공하지는 않습니다.

제한조건

모든 비즈니스에서 데이터는 특정 제한사항 또는 규칙을 준수해야 합니다. 예를 들어, 직원 번호는 고유해야 합니다. 데이터베이스 관리 프로그램은 이러한 규칙을 적용하기 위한 방법으로 *제한조건*을 제공합니다.

다음과 같은 제한조건 유형이 있습니다.

- NOT NULL 제한조건
- 고유(또는 고유 키) 제한조건
- 기본 키 제한조건
- 외부 키(또는 참조 무결성) 제한조건
- 테이블 점검 제한조건
- 정보용 제한조건

제한조건은 테이블하고만 연관되며, CREATE TABLE문을 사용하여 테이블 작성 프로세스의 파트로 정의되거나 ALTER TABLE문을 사용하여 테이블 작성 후 테이블의 정의에 추가됩니다. ALTER TABLE문을 사용하여 제한조건을 수정할 수 있습니다. 대부분의 경우 언제든지 기존 제한조건을 삭제할 수 있습니다. 삭제 조치는 테이블의 구조나 테이블에 저장되어 있는 데이터에 영향을 주지 않습니다.

주: 고유 제한조건 및 1차 제한조건은 테이블 오브젝트하고만 연관되며, 종종 하나 이상의 고유 또는 기본 키 인덱스 사용을 통해 적용됩니다.

인덱스

*인덱스*는 하나 이상의 키 값에 의해 논리적으로 정렬되는 포인터 세트입니다. 포인터는 테이블의 행, MDC 테이블의 블록, XML 스토리지 오브젝트의 XML 데이터 등을 의미할 수 있습니다.

인덱스를 사용하여 다음을 수행합니다.

- 성능 향상. 대부분의 경우 인덱스를 사용하면 데이터 액세스가 더 빠릅니다. 뷰에 대한 인덱스를 작성할 수 없지만, 뷰가 기초로 하는 테이블에 대해 작성된 인덱스는 가끔 해당 뷰에 대한 조작 성능을 향상시킬 수 있습니다.
- 고유성 보장. 고유 인덱스를 갖는 테이블은 동일한 키를 갖는 행을 가질 수 없습니다.

데이터는 테이블에 추가될 때, 테이블이나 추가될 데이터에 대해 다른 조치가 수행되지 않았으면 맨 아래에 추가됩니다. 데이터에 대한 본질적인 순서는 없습니다. 특정 데이터의 행을 검색할 때 테이블의 첫 번째 행부터 마지막 행까지 점검해야 합니다. 인덱스가 사용 가능할 경우 테이블에서 순서대로 데이터에 액세스하기 위한 수단으로 사용됩니다.

일반적으로 테이블에서 데이터를 검색할 때 특정 값을 갖는 컬럼이 있는 행을 찾습니다. 데이터 행에 있는 컬럼 값을 사용하여 전체 행을 식별할 수 있습니다. 예를 들어 직원 번호는 대개 특정 개별 직원을 고유하게 정의할 수 있습니다. 또는 행을 식별하기 위해 둘 이상의 컬럼이 필요할 수 있습니다. 예를 들어 고객 이름과 전화번호의 조합이 필요할 수 있습니다. 데이터 행을 식별하는 데 사용되는 인덱스의 컬럼을 *키*라고 합니다. 한 컬럼이 둘 이상의 키에서 사용될 수 있습니다.

인덱스는 키의 값에 따라 순서가 정해집니다. 키는 고유하거나 고유하지 않을 수 있습니다. 각 테이블에는 최소 하나의 고유 키가 있어야 하지만 다른 고유하지 않은 키도 있을 수 있습니다. 각 인덱스는 정확히 한 개의 키를 가집니다. 예를 들어, 사원 ID 번호(고유)를 한 인덱스의 키로 사용하고 부서 번호(고유하지 않음)를 다른 인덱스의 키로 사용할 수 있습니다.

모든 인덱스가 테이블의 행을 가리키지는 않습니다. MDC 블록 인덱스는 데이터의 Extent(또는 블록)를 가리킵니다. XML 데이터에 대한 XML 인덱스는 특정 XML 패턴 표현식을 사용하여 단일 컬럼 안에 저장된 XML 문서의 경로 및 값을 인덱싱합니다. 이 컬럼의 데이터 유형은 XML이어야 합니다. MDC 블록 인덱스와 XML 인덱스 둘 다 시스템 생성 인덱스입니다.

예제

10 페이지의 그림 1의 테이블 A는 테이블에 있는 직원 번호를 기초로 하는 인덱스를 갖습니다. 이 키 값은 테이블 A의 행에 대한 포인터입니다. 예를 들어, 사원 번호 19가 사원 KMP를 지시합니다. 인덱스는 포인터를 통해 데이터에 대한 경로를 작성하므로 테이블에 있는 행에 효율적으로 액세스할 수 있습니다.

인덱스 키가 고유하도록 고유 인덱스를 작성할 수 있습니다. *인덱스 키*는 인덱스가 정의된 여러 개의 정렬된 컬럼 콜렉션 또는 한 개의 컬럼을 의미합니다. 고유 인덱스를 사용하면 인덱스가 정의된 컬럼에 있는 각 인덱스 키 값을 고유하게 만들 수 있습니다.

10 페이지의 그림 1은 인덱스와 테이블의 관계를 보여줍니다.

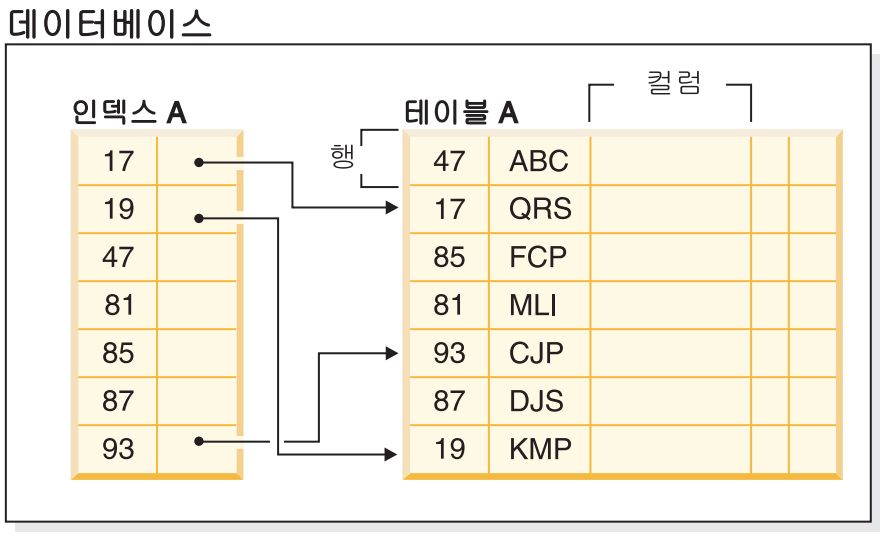


그림 1. 인덱스와 테이블의 관계

그림 2는 일부 데이터베이스 오브젝트 간의 관계를 보여줍니다. 또한 테이블, 인덱스 및 LONG 데이터가 테이블 스페이스에 저장된다는 사실도 나타냅니다.

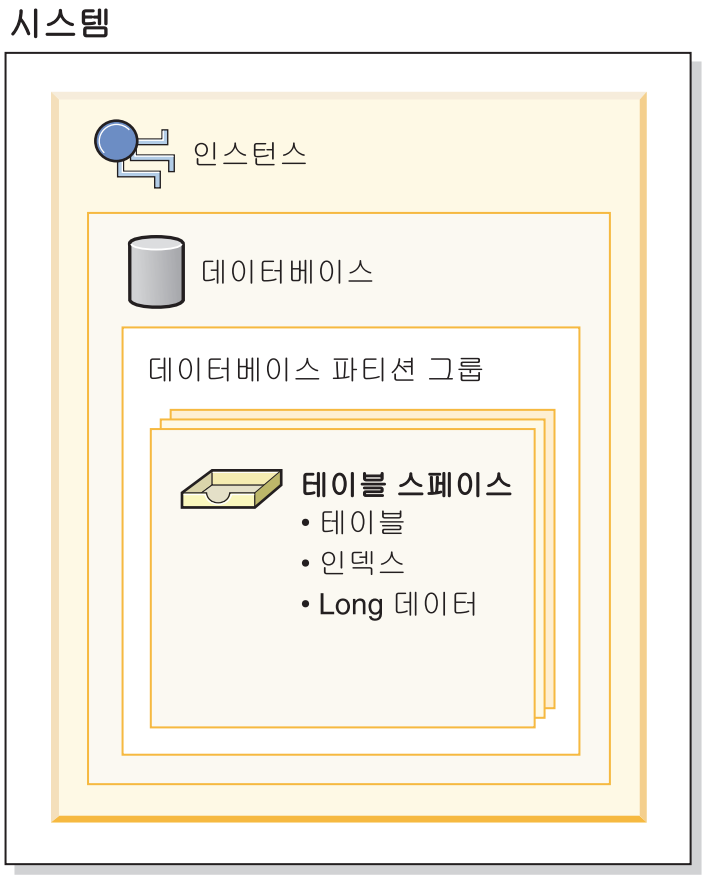


그림 2. 선택된 데이터베이스 오브젝트 간의 관계

트리거

*트리거*는 지정된 테이블에서 삽입, 갱신 또는 삭제 조작에 대한 응답으로 수행되는 조치 세트를 정의합니다. 이러한 SQL 조작이 실행되면 트리거가 *활성화*되었다고 합니다. 트리거는 선택적이며 CREATE TRIGGER문을 사용하여 정의됩니다.

참조 제한조건 및 점검 제한조건과 함께 트리거를 사용하여 데이터 무결성 규칙을 적용할 수 있습니다. 또한 트리거를 사용하여 기타 테이블이 갱신되도록 하거나, 삽입되거나 갱신된 행의 값을 자동으로 생성 또는 변환하거나, 함수를 호출하여 경보 발행과 같은 태스크를 수행할 수 있습니다.

트리거는 *트랜잭션* 비즈니스 규칙을 정의하고 적용하는 데 유용한 메커니즘으로, 트랜잭션 비즈니스 규칙은 여러 데이터 상태를 포함하는 규칙입니다(예: 급여를 10% 넘게 인상할 수 없음).

트리거를 사용하면 비즈니스 규칙을 적용하는 논리가 데이터베이스 내에 한정됩니다. 이는 응용프로그램에서는 이러한 규칙을 적용할 책임이 없다는 의미입니다. 모든 테이블에 적용되는 중심 논리를 사용하면 논리가 변경될 경우 응용프로그램을 변경할 필요가 없기 때문에 유지보수가 간편합니다.

트리거 작성 시 다음 내용이 지정됩니다.

- *주제 테이블*은 트리거가 정의된 테이블을 지정합니다.
- *트리거 이벤트*는 주제 테이블을 수정하는 특정 SQL 조작을 정의합니다. 이벤트는 삽입, 갱신 또는 삭제 조작 중 하나입니다.
- *트리거 활성화 시간*은 트리거 이벤트 발생 이전 또는 이후에 트리거를 활성화할지 여부를 지정합니다.

트리거를 활성화시키는 명령문에는 *영향받은 행 세트*가 포함됩니다. 이들 행은 삽입, 갱신 또는 삭제되는 중인 주제 테이블의 행입니다. *트리거 수준*은 트리거의 조치를 명령문마다 한 번 수행할지 또는 각각의 영향받은 행마다 한 번 수행할지 여부를 지정합니다.

*트리거 조치*는 선택적 검색 조건과 트리거가 활성화될 때마다 실행되는 명령문 세트로 구성됩니다. 명령문은 검색 조건이 참으로 평가하는 경우에만 실행됩니다. 트리거 활성화 시간이 트리거 이벤트 이전인 경우 트리거 조치에는 선택하거나, 전이 변수를 설정하거나, SQL 상태를 신호하는 명령문이 포함됩니다. 트리거 활성화 시간이 트리거 이벤트 이후인 경우에는 트리거 조치에 선택, 삽입, 갱신, 삭제 또는 SQL 상태를 신호하는 명령문이 포함됩니다.

트리거 조치에서는 *전이 변수*를 사용하여 영향받은 행 세트의 값을 참조할 수 있습니다. 전이 변수에서는 주제 테이블의 컬럼 이름을 사용하며 이 이름은 참조가 갱신 이전

의 이전 값에 대한 것인지 또는 갱신 이후의 새 값에 대한 것이지 여부를 식별하는 지정된 이름으로 규정됩니다. 사전, 삽입 또는 갱신 트리거의 SET 변수 명령문을 사용하여 새 값을 변경할 수도 있습니다.

영향받은 행 세트의 값을 참조하는 또다른 방법은 *전이 테이블*을 사용하는 것입니다. 전이 테이블에서도 주제 테이블의 컬럼 이름을 사용하지만 영향받은 행의 전체 세트를 테이블로 취급할 수 있도록 이름을 지정합니다. AFTER 트리거에서만 전이 테이블을 사용할 수 있으며(즉, BEFORE 및 INSTEAD OF 트리거에서는 사용할 수 없음) 이전 및 새 값에 개별 전이 테이블을 정의할 수 있습니다.

테이블, 이벤트(삽입, 갱신, 삭제, INSTEAD OF) 또는 활성화 시간(BEFORE, AFTER)의 조합에 다중 트리거를 지정할 수 있습니다. 특정 테이블, 이벤트 및 활성화 시간에 둘 이상의 트리거가 존재하는 경우 트리거가 활성화되는 순서는 트리거 작성 순서와 동일합니다. 따라서 최근에 작성된 트리거가 마지막으로 활성화되는 트리거입니다.

트리거를 활성화시키면 *트리거 연쇄*가 발생할 수 있습니다. 트리거 연쇄는 다른 트리거 또는 동일한 트리거를 다시 활성화시키는 명령문을 실행하는 하나의 트리거를 활성화시킴으로써 발생합니다. 트리거된 조치는 또한 삭제에 적용되는 참조 무결성 규칙의 응용프로그램으로 인해 발생하는 갱신의 원인이 될 수 있습니다. 그런 다음 추가적인 트리거 활성화가 발생합니다. 트리거 연쇄가 발생하면 트리거 및 참조 무결성 삭제 규칙의 체인이 활성화되어 단일 INSERT, UPDATE 또는 DELETE문 실행으로 데이터베이스가 상당히 변경될 수 있습니다.

다중 트리거에 동일한 오브젝트에 대한 삽입, 갱신 또는 삭제 조치가 있는 경우, 임시 테이블과 마찬가지로 액세스 충돌을 해결하기 위해 충돌 해결 메커니즘이 사용되며 이는 성능에(특히 파티션된 데이터베이스 환경에서) 상당한 영향을 미칠 수 있습니다.

뷰

*뷰*는 데이터를 유지하지 않고도 효율적으로 데이터를 나타낼 수 있는 방법입니다. 뷰는 실제 테이블이 아니며 영구 스토리지가 필요하지 않습니다. 『가상 테이블』이 작성되어 사용됩니다.

*뷰*는 하나 이상의 테이블에서 데이터를 다르게 검토할 수 있는 방법을 제공합니다. 뷰는 결과 테이블의 이름 지정된 스펙입니다. 스펙은 SQL문에서 뷰가 참조될 때마다 실행되는 SELECT문입니다. 뷰에는 테이블과 마찬가지로 컬럼과 행이 있습니다. 모든 뷰를 테이블처럼 데이터 검색에 사용할 수 있습니다. 삽입, 갱신 또는 삭제에 뷰를 사용할 수 있는지 여부는 뷰의 정의에 따라 다릅니다.

뷰의 기본이 되는 테이블에 포함된 모든 또는 일부 컬럼이나 행을 뷰에 포함시킬 수 있습니다. 예를 들어, 특정 부서의 모든 직원을 나열할 수 있도록 하나의 뷰에 부서 테이블과 직원 테이블을 결합시킬 수 있습니다.

그림 3에는 테이블과 뷰 간의 관계가 표시되어 있습니다.

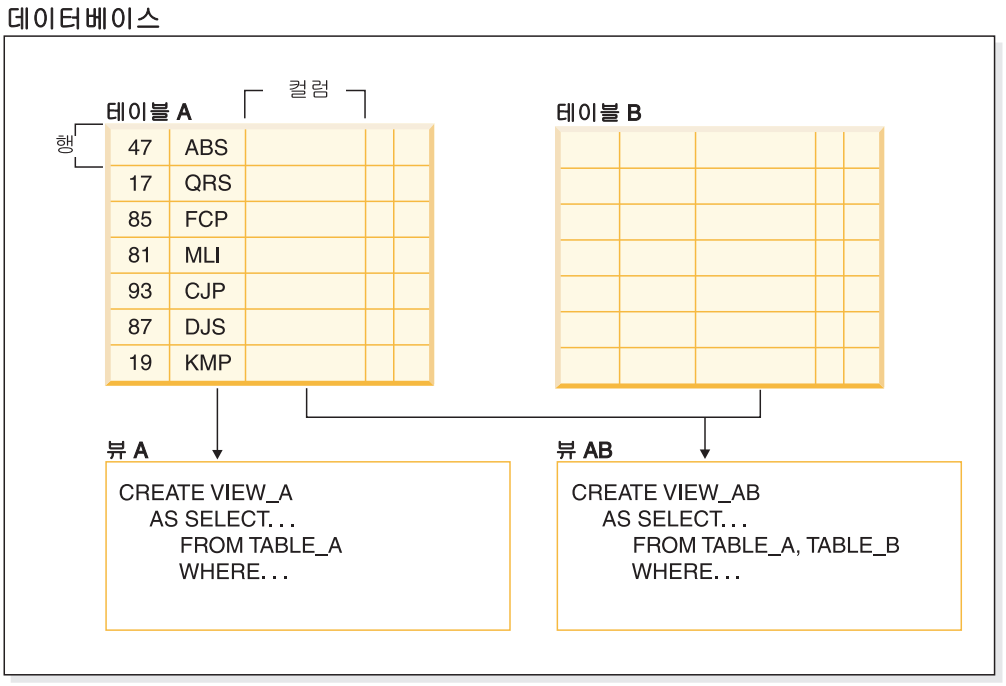


그림 3. 테이블과 뷰의 관계

뷰를 사용하면 여러 사용자가 동일한 데이터를 서로 다른 표시 형식으로 볼 수 있기 때문에 뷰를 사용하여 중요한 데이터에 액세스하는 것을 제어할 수 있습니다. 예를 들어, 몇몇 사용자가 직원에 대한 데이터 테이블에 액세스 중인 경우가 있습니다. 관리자는 자신의 직원에 대한 데이터를 확인할 수 있지만 다른 부서 직원의 데이터는 볼 수 없습니다. 채용 담당자는 모든 직원의 채용 날짜를 확인할 수 있지만 직원의 급여는 볼 수 없으며 재무 담당자는 급여를 확인할 수 있지만 채용 날짜는 볼 수 없습니다. 이들 각각의 사용자는 테이블에서 파생된 뷰를 사용하여 작업합니다. 각 뷰는 테이블로 표시되고 고유 이름을 갖습니다.

뷰의 컬럼이 기본 테이블 컬럼에서 직접 파생된 경우 해당 뷰 컬럼은 테이블 컬럼에 적용되는 모든 제한조건을 상속합니다. 예를 들어, 뷰에 해당 테이블의 외부 키가 포함된 경우 이 뷰를 사용하는 삽입 및 갱신 조작에는 테이블과 동일한 참조 제한조건이 적용됩니다. 또한 뷰의 테이블이 상위 테이블인 경우 해당 뷰를 사용하는 삭제 및 갱신 조작에는 테이블의 삭제 및 갱신 조작에 적용되는 것과 동일한 규칙이 적용됩니다.

뷰는 결과 테이블에서 각 컬럼의 데이터 유형을 추출하거나 사용자 정의 구조화된 유형의 속성을 기본으로 유형을 지정할 수 있습니다. 이를 *유형이 지정된 뷰*라고 합니다. 유형이 지정된 테이블과 마찬가지로 유형이 지정된 뷰는 뷰 계층 구조의 파트가 될 수 있습니다. *서브뷰*는 *수퍼 뷰*에서 컬럼을 상속합니다. *서브뷰*라는 용어는 하나의 유형이 지정된 뷰 및 뷰 계층 구조에서 해당 뷰 아래 있는 모든 유형이 지정된 뷰에 적용됩니다. V 뷰의 *올바른 서브뷰*는 유형이 지정된 뷰 계층 구조에서 V 아래 있는 뷰입니다.

예를 들어, 테이블이 삭제된 경우 뷰가 작동 불능 상태가 될 수 있습니다. 이런 경우 해당 뷰를 더 이상 SQL 조작에 사용할 수 없습니다.

별명

*별명*은 모듈, 테이블 또는 다른 별명과 같은 오브젝트의 대체 이름입니다. 해당 오브젝트를 바로 참조할 수 있는 위치 어디서나 오브젝트를 참조하는 데 별명을 사용할 수 있습니다.

일부 컨텍스트에서는 별명을 사용할 수 없습니다. 예를 들어, 점검 제한조건의 점검 조건에서는 별명을 사용할 수 없습니다. 별명은 선언된 임시 테이블을 참조할 수 없지만 작성된 임시 테이블을 참조할 수는 있습니다.

기타 오브젝트와 마찬가지로 별명을 작성하거나 삭제할 수 있으며 별명과 연관된 주석이 있을 수 있습니다. 순환 참조가 없는 한 별명은 *연결*이라는 프로세스에 있는 기타 별명을 참조할 수 있습니다. 별명을 사용하기 위해 특수한 권한 또는 특권이 필요하지 않습니다. 그러나 별명을 통해 참조하는 오브젝트에 액세스하려면 해당 오브젝트와 연관된 권한이 필요합니다.

별명이 *공용* 별명으로 정의된 경우 현재 디폴트 스키마 이름의 영향을 받지 않고 규정되지 않은 이름으로 별명을 참조할 수 있습니다. SYSPUBLIC 규정자를 사용하여 참조할 수도 있습니다.

*동의어*는 별명의 대체 이름입니다.

자세한 정보는 *SQL 참조서, 볼륨 1*의 "ID의 별명"을 참조하십시오.

패키지

*패키지*는 데이터베이스에 저장된 오브젝트로서 응용프로그램의 소스 파일과 연관된 SQL 문을 처리하는 데 필요한 정보를 포함합니다.

패키지는 다음 방법 중 하나를 통해 생성됩니다.

- PREP 명령을 사용하여 소스 파일 프리컴파일
- BIND 명령을 사용하여 프리컴파일러를 통해 생성된 바인드 파일 바인딩

권한 부여, 특권 및 오브젝트 소유권

사용자(권한 부여 ID로 식별됨)에게 지정된 기능을 수행할 수 있는 권한이 있는 경우에만 사용자가 조작을 실행할 수 있습니다. 테이블을 작성하려면 사용자는 테이블 작성 권한을 부여받아야 하고, 테이블을 변경하려면 사용자는 테이블 변경 권한을 부여받아야 합니다.

데이터베이스 관리 프로그램에서 각 사용자는 특정 태스크를 수행하는 데 필요한 각 데이터베이스 함수를 사용할 수 있도록 특정 권한을 부여 받아야 합니다. 사용자는 자신의 사용자 ID에 대한 권한 부여를 통해 또는 권한을 보유하고 있는 역할 또는 그룹의 멤버십을 통해 필요한 권한을 얻을 수 있습니다.

권한 부여의 종류에는 *관리자 권한*, *특권* 및 *LBAC 증명서*라는 세 가지가 있습니다. 또한 오브젝트의 소유권은 작성된 오브젝트에 대한 권한 부여 등급과 함께 제공됩니다. 이러한 권한 부여의 종류는 아래에서 설명합니다.

관리자 권한

관리자 권한을 갖는 사람은 데이터베이스 관리 프로그램을 제어하는 태스크의 의무가 부여되고 데이터의 안정성 및 무결성을 책임집니다.

시스템 레벨 권한

시스템 레벨 권한은 인스턴스 레벨 함수에 대해 다양한 수준의 제어를 제공합니다.

- SYSADM(시스템 관리자) 권한

  SYSADM(시스템 관리자) 권한은 데이터베이스 관리 프로그램에 의해 작성되고 유지보수되는 모든 자원에 대한 제어를 제공합니다. 시스템 관리자는 SYSCTRL, SYSMAINT 및 SYSMON의 모든 권한을 보유합니다. SYSADM 권한을 가진 사용자는 데이터베이스 관리 프로그램을 제어하고 데이터의 무결성 및 보안을 보장할 수 있습니다.

- SYSCTRL 권한

  SYSCTRL 권한은 시스템 자원에 영향을 주는 조작에 대한 제어를 제공합니다. 예를 들어, SYSCTRL 권한이 있는 사용자는 데이터베이스를 작성, 갱신, 시작, 중지 또는 삭제할 수 있습니다. 또한 이 사용자는 인스턴스를 시작 및 중지할 수도 있지만 테이블 데이터에 액세스할 수는 없습니다. SYSCTRL 권한이 있는 사용자는 SYSMON 권한을 보유할 수도 있습니다.

- SYSMAINT 권한

  SYSMAINT 권한은 인스턴스와 연관된 모든 데이터베이스에서 유지보수 조작을 수행하기 위해 필요한 권한을 제공합니다. SYSMAINT 권한이 있는 사용자는 데이터베이스 구성을 갱신하고, 데이터베이스 또는 테이블 스페이스를 백업하며, 기존 데이터베이스를 복원하고, 데이터베이스를 모니터할 수 있습니다. SYSCTRL과 마찬가지로 SYSMAINT는 테이블 데이터에 대한 액세스를 제공하지 않습니다. SYSMAINT 권한이 있는 사용자는 SYSMON 권한을 보유할 수도 있습니다.

- SYSMON(시스템 모니터) 권한

  SYSMON(시스템 모니터) 권한은 데이터베이스 시스템 모니터를 사용하는 데 필요한 권한을 제공합니다.

데이터베이스 레벨 권한

데이터베이스 레벨 권한은 데이터베이스 내에서의 제어를 제공합니다.

- DBADM(데이터베이스 관리자)

  DBADM 권한 레벨은 단일 데이터베이스에 대한 관리 권한을 제공합니다. 이 데이터베이스 관리자는 오브젝트 작성 및 데이터베이스 명령 발행에 필요한 특권을 보유하고 있습니다.

  SECADM 권한을 보유한 사용자만 DBADM 권한을 부여할 수 있습니다. DBADM 권한은 PUBLIC에 부여할 수 없습니다.

- SECADM(보안 관리자)

  SECADM 권한 레벨은 단일 데이터베이스에 대한 보안 관리 권한을 제공합니다. 보안 관리자 권한을 보유한 경우, 데이터베이스 보안 오브젝트(데이터베이스 역할, 감사 규정, 트러스트된 컨텍스트, 보안 레이블 구성요소 및 보안 레이블)를 관리하고 모든 데이터베이스 특권과 권한을 부여 및 취소할 수 있습니다. SECADM 권한을 보유한 사용자는 자신이 소유하지 않은 오브젝트의 소유권을 양도할 수 있으며, AUDIT 문을 사용하여 서버에 있는 특정 데이터베이스 또는 데이터베이스 오브젝트와 감사 규정을 연관시킬 수도 있습니다.

  SECADM 권한에는 테이블에 저장된 데이터에 액세스할 수 있는 고유 특권이 없습니다. 이 권한은 SECADM 권한을 가진 사용자만 부여할 수 있습니다. SECADM 권한은 PUBLIC에 부여할 수 없습니다.

- SQLADM(SQL 관리자)

  SQLADM 권한 레벨은 단일 데이터베이스 내에 있는 SQL문을 모니터하고 조정할 수 있는 관리 권한을 제공하며, ACCESSCTRL 또는 SECADM 권한을 가진 사용자가 부여할 수 있습니다.

- WLMADM(워크로드 관리자)

  WLMADM 권한은 워크로드 관리 오브젝트(예: 서비스 클래스, 작업 조치 세트, 작업 클래스 세트 및 워크로드)를 관리할 수 있는 관리 권한을 제공하며, ACCESSCTRL 또는 SECADM 권한을 가진 사용자가 부여할 수 있습니다.

- EXPLAIN(설명 권한)

  EXPLAIN 권한 레벨은 데이터 액세스 권한을 부여하지 않고 쿼리 계획을 설명할 수 있는 관리 권한을 제공하며, ACCESSCTRL 또는 SECADM 권한을 가진 사용자만 부여할 수 있습니다.

- ACCESSCTRL(액세스 제어 권한)

ACCESSCTRL 권한 레벨은 다음 GRANT 및 REVOKE문을 발행할 수 있는 관리 권한을 제공하며, SECADM 권한을 보유한 사용자만 ACCESSCTRL 권한을 부여할 수 있습니다. ACCESSCTRL 권한은 PUBLIC에 부여할 수 없습니다.

- GRANT(데이터베이스 권한)

  ACCESSCTRL 권한을 보유한 사용자는 ACCESSCTRL, DATAACCESS, DBADM 또는 SECADM 권한을 부여할 수 없습니다. SECADM 권한을 보유한 사용자만 이러한 권한을 부여할 수 있습니다.
- GRANT(전역 변수 특권)
- GRANT(인덱스 특권)
- GRANT(모듈 특권)
- GRANT(패키지 특권)
- GRANT(루틴 특권)
- GRANT(스키마 특권)
- GRANT(시퀀스 특권)
- GRANT(서버 특권)
- GRANT(테이블, 뷰 또는 별칭 특권)
- GRANT(테이블 스페이스 권한)
- GRANT(워크로드 특권)
- GRANT(XSR 오브젝트 특권)

• DATAACCESS(데이터 액세스 권한)

  DATAACCESS 권한 레벨은 다음과 같은 특권 및 권한을 제공하며, SECADM 권한을 보유한 사용자만 부여할 수 있습니다. DATAACCESS 권한은 PUBLIC에 부여할 수 없습니다.
  - LOAD 권한
  - 테이블, 뷰, 별칭 및 구체화된 쿼리 테이블에 대한 SELECT, INSERT, UPDATE, DELETE 특권
  - 패키지에 대한 EXECUTE 특권
  - 모듈에 대한 EXECUTE 특권
  - 루틴에 대한 EXECUTE 특권

    예외가 적용되는 감사 루틴: AUDIT_ARCHIVE, AUDIT_LIST_LOGS, AUDIT_DELIM_EXTRACT

• 데이터베이스 권한(비관리자)

  테이블 또는 루틴의 작성과 같은 활동을 수행하거나 데이터를 테이블로 로드하려면 특정 데이터베이스 권한이 필요합니다. 예를 들어, load 유틸리티를 사용하여 데이터

를 테이블로 로드하려면 LOAD 데이터베이스 권한이 필요합니다. 이 경우 사용자는 테이블에 대한 INSERT 권한도 보유하고 있어야 합니다.

특권

특권이란 특정 조치 또는 태스크를 수행하는 데 필요한 권한을 말합니다. 권한 부여된 사용자는 오브젝트를 작성하고, 자신이 소유하는 오브젝트에 액세스하며, GRANT문을 사용하여 자신의 오브젝트에 대한 특권을 다른 사용자에게 전달할 수 있습니다.

특권은 특정 사용자, 그룹 또는 PUBLIC에 부여될 수 있습니다. PUBLIC은 향후 사용자를 포함하는 모든 사용자로 구성된 특수 그룹입니다. 그룹의 구성원인 사용자는 해당 그룹에 권한 부여된 특권을 간접적으로 이용합니다.

CONTROL 특권: 오브젝트에 대한 CONTROL 특권을 보유하면 해당 데이터베이스 오브젝트에 액세스하고 해당 오브젝트에 대한 특권을 다른 사용자에게 권한 부여하거나 권한 취소할 수 있습니다.

주: CONTROL 특권은 테이블, 뷰, 별칭, 인덱스 및 패키지에만 적용됩니다.

다른 사용자에게 해당 오브젝트에 대한 CONTROL 특권이 필요한 경우, SECADM 또는 ACCESSCTRL 권한을 보유한 사용자가 해당 오브젝트에 대한 CONTROL 권한을 부여할 수 있습니다. CONTROL 특권을 오브젝트 소유자로부터 권한 취소할 수 없지만, TRANSFER OWNERSHIP문을 사용하여 오브젝트 소유자를 변경할 수 있습니다.

개별적 특권: 사용자가 특정 오브젝트에 대해 특정 태스크를 수행할 수 있도록 개별적 특권을 부여할 수 있습니다. 관리 권한(SYSADM 또는 SECADM) 또는 CONTROL 특권을 보유한 사용자는 사용자에게 권한을 부여하거나 사용자로부터 권한을 취소할 수 있습니다.

개별적 특권 및 데이터베이스 권한은 특정 함수를 사용할 수 있지만 기타 사용자에게 동일한 특권이나 권한을 부여하는 권한은 가지고 있지 않습니다. 다른 사용자에게 테이블, 뷰, 스키마, 패키지, 루틴 및 시퀀스 특권을 부여하는 권한은 GRANT 명령문에서 WITH GRANT OPTION을 통해 다른 사용자에게 확장될 수 있습니다. 그러나 WITH GRANT OPTION으로 일단 부여된 권한을 취소할 수 있는 특권을 사용자가 부여할 수는 없습니다. 권한을 취소하려면 SECADM 권한, ACCESSCTRL 권한 또는 CONTROL 권한이 있어야 합니다.

패키지 또는 루틴의 오브젝트에 대한 특권: 사용자에게 패키지 또는 루틴을 실행하는 특권이 있는 경우, 패키지 또는 루틴에서 사용하는 오브젝트의 특정 특권을 필요로 하지 않습니다. 패키지 또는 루틴에 정적 SQL 또는 XQuery문이 포함된 경우 패키지 소유자의 특권이 이들 명령문에 사용됩니다. 패키지 또는 루틴에 동적 SQL 또는 XQuery 명령문이 포함된 경우, 동적 쿼리 명령문을 발행하는 패키지의 **DYNAMICRULES**

BIND 옵션의 설정 및 패키지가 루틴 컨텍스트에서 사용되고 있을 때 해당 명령문이 발행되었는지 여부에 따라 특권 검사에 사용되는 권한 부여 ID가 달라집니다.

사용자 또는 그룹은 개별적 특권 또는 권한의 임의 조합에 대해 권한 부여될 수 있습니다. 특권이 오브젝트와 연관되는 경우에는 해당 오브젝트가 존재해야 합니다. 예를 들어 사용자는 이전에 테이블이 작성된 경우에만 테이블에 대한 SELECT 특권을 부여할 수 있습니다.

주: 사용자 또는 그룹을 나타내는 권한 부여 이름에 권한 및 특권을 부여했는데 해당 이름으로 작성된 사용자 또는 그룹이 없는 경우 주의해야 합니다. 일정 시간이 경과하면 사용자 또는 그룹이 이 이름으로 작성되어 이 권한 부여 이름과 연관된 모든 권한 및 특권을 자동으로 받을 수 있습니다.

REVOKE 명령문은 이전에 권한 부여된 특권을 취소하기 위해 사용됩니다. 권한 부여 이름의 특권 취소는 모든 권한 부여 이름으로 부여된 특권을 취소합니다.

권한 부여 이름의 특권을 취소해도 이 권한 부여 이름으로 특권을 부여한 임의의 기타 권한 부여 이름의 동일한 특권은 취소되지 않습니다. 예를 들어, CLAIRE가 SELECT WITH GRANT OPTION을 RICK에게 권한 부여한 후에 RICK이 SELECT를 BOBBY 및 CHRIS에게 권한 부여한다고 가정하십시오. CLAIRE가 RICK의 SELECT 특권을 취소해도 BOBBY과 CHRIS는 여전히 SELECT 특권을 갖습니다.

LBAC 증명서

레이블 기반 액세스 제어(LBAC)를 사용하면 보안 관리자가 개별 행 및 개별 컬럼에 대해 쓰기 액세스 권한 및 읽기 액세스 권한을 갖고 있는 사용자를 정확히 판별할 수 있습니다. 보안 관리자는 보안 규정을 작성하여 LBAC 시스템을 구성합니다. 보안 규정에는 어떤 사용자가 어떤 데이터에 액세스할 수 있는지를 결정하는 데 사용되는 기준이 기술되어 있습니다. 임의의 한 테이블을 보호하는데는 하나의 보안 테이블만 사용할 수 있으나 다른 테이블은 다른 보안 규정을 사용하여 보호할 수 있습니다.

보안 규정을 작성한 후 보안 관리자는 해당 규정에 포함될 보안 레이블 및 면제라는 데이터베이스 오브젝트를 작성합니다. 보안 레이블에는 특정 보안 기준 세트가 기술되어 있습니다. 면제를 보유한 사용자가 해당 보안 규정으로 보호된 데이터에 액세스하는 경우, 면제를 통해 보안 레이블을 비교하는 규칙이 해당 사용자에게는 적용되지 않도록 할 수 있습니다.

작성되면, 보안 레벨을 개별 테이블의 행 및 컬럼에 연관시켜 여기에 데이터가 보류되는 것을 방지할 수 있습니다. 보안 레이블에 의해 보호되는 데이터를 보호 데이터라고 합니다. 관리 관리자는 사용자에게 보안 레이블을 부여함으로써 사용자가 보호 데이터에 액세스하게 할 수 있습니다. 사용자가 보호 데이터에 액세스를 시도하면 사용자의 보안 레이블을 데이터를 보호하는 보안 레이블과 비교합니다. 보호 레이블은 일부 보안 레이블은 차단하고 일부 레이블은 차단하지 않습니다.

오브젝트 소유권

오브젝트가 작성되면, 하나의 권한 부여 ID가 오브젝트의 *소유권*에 지정됩니다. 소유권은 사용자가 적용 가능한 SQL 또는 XQuery문의 오브젝트를 참조할 수 있는 권한이 있음을 의미합니다.

오브젝트가 스키마 내에서 작성될 때, 명령문의 권한 부여 ID가 내재적 또는 명시적으로 지정된 스키마에 오브젝트를 작성하기 위한 필수 특권을 가져야 합니다. 즉, 권한 부여 이름은 스키마의 소유자이거나 스키마에 관한 CREATEIN 특권을 소유해야 합니다.

주: 이 요구사항은 테이블 스페이스, 버퍼 풀 또는 데이터베이스 파티션 그룹을 작성할 때 적용되지 않습니다. 이들 오브젝트는 스키마에서 작성되지 않습니다.

오브젝트가 작성될 때 명령문의 권한 부여 ID는 해당 오브젝트의 정의자이며, 오브젝트가 작성된 후에는 디폴트로 오브젝트 소유자가 됩니다.

주: 한 가지 예외가 존재합니다. AUTHORIZATION 옵션을 CREATE SCHEMA문에 대해 지정할 경우, CREATE SCHEMA 조작의 파트로 작성되는 모든 기타 오브젝트는 AUTHORIZATION 옵션에 의해 지정되는 권한 부여 ID에 의해 소유됩니다. 그러나 초기 CREATE SCHEMA 조작 후 스키마에 작성된 오브젝트는 특정 CREATE 문과 연관된 권한 부여 ID에서 소유합니다.

예를 들어 명령문 `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C1 INT)`는 SCOTT이 소유하는 `SCOTTSTUFF` 스키마와 `SCOTTSTUFF.T1` 테이블을 작성합니다. BOBBY 사용자에게 `SCOTTSTUFF` 스키마에 관한 CREATEIN 특권이 부여되었고 `SCOTTSTUFF.T1` 테이블에 관한 인덱스를 작성한다고 가정하십시오. 인덱스는 스키마 이후에 작성되므로, BOBBY는 `SCOTTSTUFF.T1`에 관한 인덱스를 소유합니다.

특권은 다음과 같이 작성 중인 오브젝트 유형을 기반으로 오브젝트 소유자에게 지정됩니다.

- CONTROL 특권은 새로 작성된 테이블, 인덱스 및 패키지에 내재적으로 부여됩니다. 이 특권을 사용하여 오브젝트 작성자는 데이터베이스 오브젝트를 액세스하고, 해당 오브젝트에 관하여 다른 사용자에게/로부터 특권을 부여하고 취소할 수 있습니다. 다른 사용자에게 해당 오브젝트에 대한 CONTROL 특권이 필요한 경우, ACCESSCTRL 또는 SECADM 권한을 보유한 사용자가 해당 오브젝트에 대한 CONTROL 권한을 부여해야 합니다. CONTROL 특권을 해당 오브젝트 소유자가 취소할 수 없습니다.
- 오브젝트 소유자에게 뷰 정의에서 참조하는 모든 테이블, 뷰 및 별칭에 관한 CONTROL 특권이 있을 경우 CONTROL 특권이 새로 작성된 뷰에 내재적으로 부여됩니다.

- 트리거, 루틴, 시퀀스, 테이블 스페이스 및 버퍼 풀과 같은 기타 오브젝트에는 연관된 CONTROL 특권이 없습니다. 그러나 오브젝트 소유자는 오브젝트와 연관된 각각의 특권을 자동으로 수신하며, 이러한 특권에는 WITH GRANT OPTION(지원되는 경우)이 포함됩니다. 따라서 오브젝트 소유자는 GRANT문을 사용하여 이러한 특권을 다른 사용자에게 제공할 수 있습니다. 예를 들어, USER1이 테이블 스페이스를 작성한 경우, USER1은 자동으로 이 테이블 스페이스에 대해 GRANT OPTION을 포함하는 USEAUTH 특권을 갖게 되며 USEAUTH 특권을 다른 사용자에게 부여할 수 있습니다. 추가로 오브젝트 소유자는 오브젝트를 변경하거나, 오브젝트에 관한 주석을 추가하거나 또는 삭제할 수 있습니다. 이러한 권한 부여는 오브젝트 소유자에 대하여 내재적이며 취소할 수 없습니다.

테이블 변경과 같은 오브젝트에 대한 특정 특권은 소유자가 부여할 수 있으며, ACCESSCTRL 또는 SECADM 권한을 보유한 사용자가 소유자로부터 이러한 특권을 취소할 수 있습니다. 테이블에 주석을 처리하는 것과 같은 오브젝트에서의 특정 특권은 소유자가 권한 부여 및 권한 취소할 수 없습니다. TRANSFER OWNERSHIP문을 사용하여 이러한 특권을 다른 사용자에게 이동시키십시오. 오브젝트가 작성될 때 명령문의 권한 부여 ID는 해당 오브젝트의 정의자이며, 오브젝트가 작성된 후에는 디폴트로 오브젝트 소유자가 됩니다. 그러나 BIND 명령을 사용하여 패키지를 작성하고 **OWNER** *authorization id* 옵션을 지정한 경우 패키지에 있는 정적 SQL문에 의해 작성된 오브젝트의 소유자는 *authorization id* 값입니다. 또한 AUTHORIZATION절이 CREATE SCHEMA문에서 지정된 경우 AUTHORIZATION 키 이후에 지정된 권한 이름이 스키마의 소유자입니다.

보안 관리자 또는 오브젝트 소유자는 TRANSFER OWNERSHIP문을 사용하여 데이터베이스 오브젝트의 소유권을 변경할 수 있습니다. 따라서 관리자는 권한 부여 ID를 규정자로서 사용하여 오브젝트를 작성한 후 TRANSFER OWNERSHIP문을 사용하여 관리자가 오브젝트에 대해 갖는 소유권을 권한 부여 ID로 전송함으로써 권한 부여 ID 대신 오브젝트를 작성할 수 있습니다.

시스템 카탈로그 뷰

데이터베이스 관리 프로그램은 제어하는 데이터에 대한 정보가 포함되어 있는 뷰 및 테이블 세트를 유지보수합니다. 이러한 테이블 및 뷰는 통틀어 *시스템 카탈로그*라고 알려져 있습니다.

시스템 카탈로그에는 테이블, 뷰, 인덱스, 패키지 및 함수와 같은 데이터베이스 오브젝트의 논리적 구조 및 물리적 구조에 대한 정보가 포함되어 있습니다. 여기에는 통계 정보도 들어 있습니다. 데이터베이스 관리 프로그램은 시스템 카탈로그에 있는 설명이 항상 정확하도록 관리합니다.

시스템 카탈로그 뷰는 기타 모든 데이터베이스 뷰와 같습니다. SQL문을 사용하여 시스템 카탈로그 뷰의 데이터를 쿼리할 수 있습니다. 갱신 가능한 시스템 카탈로그 뷰 세트를 사용하여 시스템 카탈로그의 특정 값을 수정할 수 있습니다.

응용프로그램 프로세스, 동시성 및 복구

모든 SQL 프로그램은 *응용프로그램 프로세스* 또는 에이전트의 일부분으로 실행됩니다. 응용프로그램 프로세스는 하나 이상의 프로그램을 실행해야 하며 데이터베이스 관리 프로그램이 자원 및 잠금을 할당하는 단위입니다. 응용프로그램 프로세스가 다르면 다른 프로그램을 실행하거나 동일한 프로그램을 다르게 실행할 수 있습니다.

여러 응용프로그램 프로세스가 동일한 데이터에 동시에 액세스를 요청할 수 있습니다. *잠금*은 예를 들어, 두 개의 응용프로그램 프로세스가 동일한 데이터 행을 동시에 갱신하지 못하도록 예방하여 데이터 무결성을 유지하기 위해 사용하는 메커니즘입니다.

데이터베이스 관리 프로그램은 한 응용프로그램 프로세스의 커미트되지 않은 변경사항을 다른 프로세스에서 우연히 발견하는 경우를 예방하기 위해 잠금을 획득합니다. 데이터베이스 관리 프로그램은 응용프로그램 프로세스가 종료되면 해당 프로세스가 획득하고 보유한 모든 잠금을 해제합니다. 그러나 응용프로그램 프로세스는 잠금을 해제하도록 직접 요청할 수 있습니다. 이 경우 *커미트* 조작을 사용하며, 이 조작은 하나의 작업 단위(UOW) 중 획득한 잠금을 해제하고 이 작업 단위 중 데이터베이스 변경사항도 커미트합니다.

*작업 단위(UOW)*는 응용프로그램 프로세스에서 복구 가능한 조작 시퀀스입니다. 응용프로그램 프로세스가 시작되거나 응용프로그램 프로세스 종료 이외의 이유로 이전 UOW가 종료된 경우 작업 단위가 시작됩니다. 커미트 조작, 롤백 조작 또는 응용프로그램 프로세스 종료 시 작업 단위가 종료됩니다. 커미트 또는 롤백 조작은 종료 중인 UOW에서 발생한 데이터베이스 변경사항에만 영향을 줍니다.

데이터베이스 관리 프로그램은 응용프로그램 프로세스가 수행한 커미트되지 않은 변경사항을 백아웃하는 방법을 제공합니다. 응용프로그램 프로세스의 일부분에 장애가 발생했거나 교착 상태 또는 잠금 시간종료가 발생한 경우 이러한 방법을 사용해야 할 수도 있습니다. 응용프로그램 프로세스는 데이터베이스 변경을 취소하도록 명시적으로 요청할 수 있습니다. 이 경우 *롤백* 조작을 사용합니다.

이러한 변경사항이 계속 커미트되지 않으면 다른 응용프로그램 프로세스가 해당 변경사항을 볼 수 없으며 변경사항이 롤백될 수 있습니다. 그러나 일반 분리 레벨이 커미트되지 않은 읽기(UR)인 경우에는 위의 내용이 적용되지 않습니다. 이러한 데이터베이스 변경사항이 커미트되었으면 다른 응용프로그램 프로세스가 이러한 변경사항에 액세스할 수 있으며 변경사항은 더 이상 롤백되지 않습니다.

DB2 콜 레벨 인터페이스(CLI) 및 Embedded SQL은 각각 독립 트랜잭션인 여러 연결을 지원하는 *동시 트랜잭션*이라는 연결 모드를 허용합니다. 응용프로그램은 동일한 데이터베이스에 대한 여러 개의 동시 연결을 사용할 수 있습니다.

응용프로그램 프로세스 대신 데이터베이스 관리 프로그램이 획득한 잠금은 분리 레벨이 커서 안정성(CS, 커서가 한 행에서 다른 행으로 이동하면 잠금이 해제됨)이거나 커미트되지 않은 읽기(UR)인 경우를 제외하고는 UOW가 종료될 때까지 보유합니다.

응용프로그램 프로세스는 자체 잠금으로 인해 조작 수행이 금지되지 않습니다. 그러나 응용프로그램이 동시 트랜잭션을 사용하는 경우 한 트랜잭션의 잠금이 동시 트랜잭션의 조작에 영향을 줄 수 있습니다.

UOW의 시작 및 종료는 응용프로그램 프로세스에서 *일관성 지점*을 정의합니다. 예를 들어, 한 계좌에서 다른 계좌로 예금을 이체하는 은행 거래가 있습니다. 이러한 거래에서는 해당 예금을 첫 번째 계좌에서 뺀 후 두 번째 계좌에 더해야 합니다. 빼기 단계 이후 데이터가 불일치합니다. 예금을 두 번째 계좌에 더한 후에만 다시 일치하게 됩니다. 두 단계가 모두 완료되었으면 커미트 조작을 사용하여 UOW를 종료하여 다른 응용프로그램 프로세스에서 변경사항을 사용할 수 있도록 합니다. UOW가 종료되기 전에 장애가 발생한 경우 데이터베이스 관리 프로그램은 커미트되지 않은 변경사항을 롤백하여 데이터 일관성을 복원합니다.

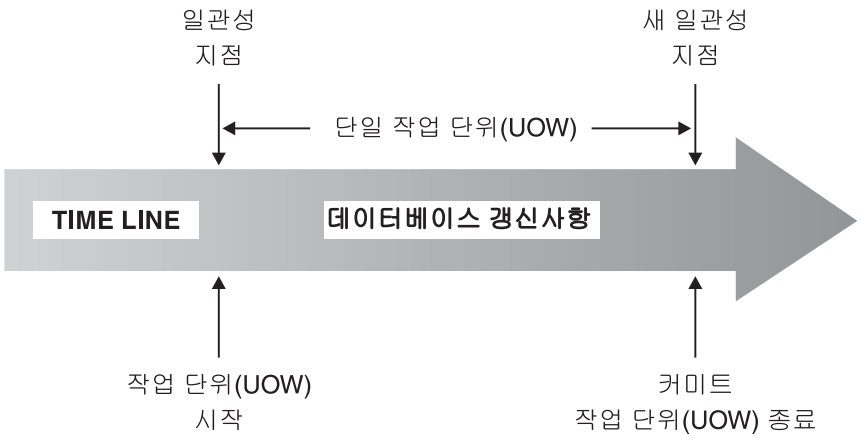


그림 4. COMMIT 문을 사용하는 작업 단위(UOW)

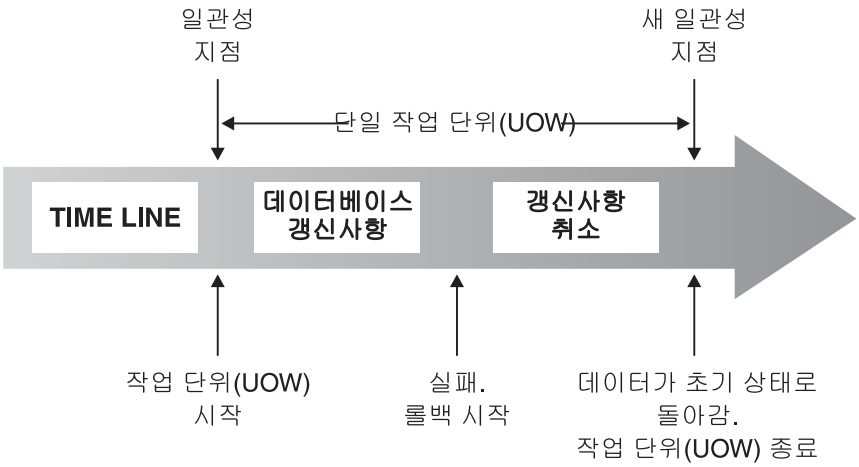


그림 5. ROLLBACK 문을 사용하는 작업 단위(UOW)

분리 레벨

응용프로그램 프로세스와 연관된 *분리 레벨*은 프로세스에서 액세스 중인 데이터를 잠그거나 동시에 실행 중인 기타 프로세스에서 분리하는 정도를 결정합니다. 분리 레벨은 작업 단위 중에만 적용됩니다.

따라서 응용프로그램 프로세스의 분리 레벨은 다음을 지정합니다.

- 응용프로그램이 읽거나 갱신하는 행을 동시에 실행 중인 다른 응용프로그램 프로세스에서 사용할 수 있는 정도
- 동시에 실행 중인 다른 응용프로그램 프로세스의 갱신 활동이 응용프로그램에 영향을 줄 수 있는 정도

정적 SQL문의 분리 레벨은 패키지의 속성으로 지정되며 해당 패키지를 사용하는 응용프로그램 프로세스에 적용됩니다. 분리 레벨은 ISOLATION 바인드 또는 프리컴파일 옵션을 설정하여 프로그램 준비 프로세스 중 지정합니다. 동적 SQL문의 경우 디폴트 분리 레벨은 명령문을 준비하는 패키지에 대해 지정된 분리 레벨입니다. 세션에서 발행된 동적 SQL문에 대해 다른 분리 레벨을 지정하려면 SET CURRENT ISOLATION문을 사용하십시오. 자세한 정보는 『CURRENT ISOLATION 특수 레지스터』를 참조하십시오. 정적 SQL문 및 동적 SQL문에서 *select-statement*의 *isolation-clause*는 특수 레지스터(설정된 경우) 및 바인드 옵션 값을 모두 겹쳐씁니다. 자세한 정보는 『Select-statement』를 참조하십시오.

분리 레벨은 잠금으로 실행되며 사용되는 잠금 유형에 따라 동시 응용프로그램 프로세스가 데이터에 대한 액세스 제한 및 방지 수준이 다릅니다. 선언된 임시 테이블 및 해당 행은 이를 선언한 응용프로그램만 액세스할 수 있으므로 잠글 수 없습니다.

데이터베이스 관리 프로그램은 세 가지 일반 잠금 범주를 지원합니다.

공유(S)

S 잠금에서 동시 응용프로그램 프로세스는 데이터에 대한 읽기 전용 조작으로 제한됩니다.

갱신(U)

U 잠금에서 동시 응용프로그램 프로세스는 행을 갱신할 수 있음을 선언하지 않은 경우 데이터에 대한 읽기 전용 조작으로 제한됩니다. 데이터베이스 관리 프로그램은 현재 행을 보는 프로세스가 행을 갱신할 수도 있는 것으로 가정합니다.

배타적(X)

X 잠금에서 동시 응용프로그램 프로세스는 어떠한 방식으로든 데이터에 액세스할 수 없습니다. 이 사항은 데이터를 읽을 수 있지만 수정할 수 없는 커미트되지 않은 읽기(UR) 분리 레벨인 응용프로그램 프로세스의 경우에는 적용되지 않습니다.

분리 레벨에 관계없이 데이터베이스 관리 프로그램은 삽입, 갱신 또는 삭제된 모든 행에 배타적 잠금을 설정합니다. 따라서 모든 분리 레벨은 작업 단위 중 응용프로그램 프로세스가 변경한 행을 작업 단위가 완료되기 전까지 다른 응용프로그램 프로세스에서 변경하지 못하도록 보장합니다.

데이터베이스 관리 프로그램은 네 가지 분리 레벨을 지원합니다.

- 『반복 읽기(RR)』
- 26 페이지의 『읽기 안정성(RS)』
- 27 페이지의 『커서 안정성(CS)』
- 27 페이지의 『커미트되지 않은 읽기(UR)』

주: 일부 호스트 데이터베이스 서버는 *커미트 없음(NC)* 분리 레벨을 지원합니다. 기타 데이터베이스 서버에서 이 분리 레벨은 커미트되지 않은 읽기 분리 레벨과 비슷한 작업을 수행합니다.

각 분리 레벨에 대한 자세한 설명은 성능 영향의 내림차순과 데이터 액세스 또는 갱신 시 필요한 주의성의 오름차순으로 표시됩니다.

반복 읽기(RR)

반복 읽기(RR) 분리 레벨은 응용프로그램이 작업 단위(UOW) 중 참조하는 모든 행을 잠급니다. 응용프로그램이 동일한 작업 단위에서 SELECT문을 발행하면 매번 동일한 결과가 리턴됩니다. RR에서 갱신사항 유실, 커미트되지 않은 데이터에 대한 액세스, 비반복 읽기 및 팬텀 읽기는 가능하지 않습니다.

RR에서 응용프로그램은 UOW가 완료될 때까지 필요하면 여러 번 행을 검색하고 해당 행에 대해 작업을 수행할 수 있습니다. 그러나 UOW가 완료되기 전까지 다른 응용프

로그램이 결과 세트에 영향을 주는 행을 갱신, 삭제 또는 삽입할 수 없습니다. RR 분리 레벨에서 실행 중인 응용프로그램은 다른 응용프로그램의 커미트되지 않은 변경사항을 볼 수 없습니다. 이 분리 레벨은 응용프로그램이 데이터를 보기 전까지(임시 테이블 또는 행 블로킹 사용 시도 포함) 리턴된 모든 데이터를 그대로 변경하지 않습니다.

검색된 행만이 아닌 참조된 모든 행을 잠급니다. 예를 들어, 10,000개의 행을 스캔하고 이러한 행에 술어를 적용하는 경우 10개의 행만 규정된 경우에도 10,000개의 모든 행에서 잠금을 유지합니다. 쿼리를 다시 실행하는 경우 해당 쿼리가 참조한 행 목록에 추가되는 행을 다른 응용프로그램에서 삽입하거나 갱신할 수 없습니다. 따라서 팬텀 읽기가 방지됩니다.

RR이 상당한 수의 잠금을 획득할 수 있으므로 이 수는 **locklist** 및 **maxlocks** 데이터베이스 구성 매개변수에서 지정한 한계를 초과할 수 있습니다. 잠금 에스컬레이션을 방지하기 위해 옵티마이저는 잠금 에스컬레이션이 유망한 경우 인덱스 스캔을 위해 단일 테이블 레벨 잠금을 획득하도록 선택할 수 있습니다. 테이블 레벨 잠금을 원하지 않는 경우 읽기 안정성 분리 레벨을 사용하십시오.

참조 제한조건을 평가하는 동안 DB2 서버는 사용자가 이전에 설정한 분리 레벨에 관계없이 하위 테이블의 스캔에서 사용되는 분리 레벨을 RR로 업그레이드하는 경우가 있습니다. 이 경우 커미트되기 전까지 추가 잠금을 유지하므로 교착 상태 또는 잠금 시간 종료가 발생할 가능성이 증가합니다. 이러한 문제를 방지하려면 외부 키 컬럼만 들어 있으며 참조 무결성 스캔에서 대신 사용할 수 있는 인덱스를 작성하십시오.

읽기 안정성(RS)

읽기 안정성 분리 레벨은 작업 단위 중 응용프로그램이 검색하는 행만 잠급니다. RS를 사용하면 UOW가 완료될 때까지 UOW 중 규정 행 읽기를 다른 응용프로그램 프로세스에서 변경할 수 없으며 해당 프로세스에서 변경사항을 커미트할 때까지 다른 응용프로그램 프로세스에서 변경한 행을 읽을 수 없습니다. RS에서는 커미트되지 않은 데이터 및 비반복 읽기에 액세스할 수 없습니다. 그러나 팬텀 읽기는 가능합니다.

이 분리 레벨은 응용프로그램이 데이터를 보기 전까지(임시 테이블 또는 행 블로킹 사용 시도 포함) 리턴된 모든 데이터를 그대로 변경하지 않습니다.

RS 분리 레벨은 높은 수준의 동시성과 안정적인 데이터 뷰를 모두 제공합니다. 이로 인해 옵티마이저를 사용하면 잠금 에스컬레이션이 발생할 때까지 테이블 레벨 잠금을 확보할 수 없습니다.

RS 분리 레벨은 다음과 같은 응용프로그램에 적합합니다.

- 동시 환경에서 작동되는 응용프로그램
- 작업 단위 중에 계속 안정적인 상태의 규정 행이 필요한 응용프로그램

- 작업 단위 중 동일한 쿼리를 여러 번 발행하지 않거나 작업 단위 중 쿼리가 여러 번 발행된 경우 동일한 결과 세트가 필요하지 않은 응용프로그램

커서 안정성(CS)

커서 안정성 분리 레벨은 해당 행에 커서가 있는 동안 트랜잭션 중 액세스하는 행을 잠급니다. 다음 행을 페치하거나 트랜잭션이 종료되기 전까지 이 잠금은 계속 유효합니다. 그러나 행의 데이터가 변경된 경우 변경사항이 커미트되기 전까지 잠금을 유지합니다.

이 분리 레벨에서는 갱신 가능한 커서가 해당 행에 있는 동안 다른 응용프로그램이 행을 갱신하거나 삭제할 수 없습니다. CS에서는 다른 응용프로그램의 커미트되지 않은 데이터에 액세스할 수 없습니다. 그러나 비반복 읽기 및 팬텀 읽기는 가능합니다.

CS가 디폴트 분리 레벨입니다. 동시성을 최대한 확대하고 커미트된 데이터만 확인해야 하는 경우에 적합합니다.

주: 버전 9.7에서 도입된 *현재 커미트됨* 시맨틱에서는 이전의 경우와 마찬가지로 커미트된 데이터만 리턴되지만 이제 데이터를 읽을 사용자는 갱신자가 행 잠금을 해제할 때까지 대기할 필요가 없습니다. 대신, 데이터를 읽을 사용자는 현재 커미트된 버전을 따르는 데이터(즉, 쓰기 조작이 시작되기 전의 데이터)를 리턴합니다.

커미트되지 않은 읽기(UR)

커미트되지 않은 읽기 분리 레벨을 사용하면 응용프로그램은 다른 트랜잭션의 커미트되지 않은 변경사항에 액세스할 수 있습니다. 또한 UR은 응용프로그램이 테이블을 변경하거나 삭제하려고 하지 않는 한, 읽고 있는 행에 다른 응용프로그램이 액세스할 수 없도록 예방하지 않습니다.

UR에서는 커미트되지 않은 데이터에 대한 액세스, 비반복 읽기 및 팬텀 읽기가 가능합니다. 이 분리 레벨은 읽기 전용 테이블에 대해 쿼리를 실행하거나 SELECT문만 발행하는 경우 및 다른 응용프로그램에서 커미트하지 않은 데이터를 보는 것이 문제가 되지 않을때 적합합니다.

UR은 읽기 전용 및 갱신 가능한 커서의 경우에 다르게 작동합니다.

- 읽기 전용 커서는 다른 트랜잭션의 커미트되지 않은 대부분의 변경사항에 액세스할 수 있습니다.
- 다른 트랜잭션에서 작성 또는 삭제 중인 테이블, 뷰 및 인덱스는 트랜잭션이 처리되는 동안 사용할 수 없습니다. 다른 트랜잭션의 기타 변경사항은 커미트 또는 롤백되기 전에 읽을 수 있습니다. UR에서 작동되는 갱신 가능한 커서는 분리 레벨이 CS일 때와 마찬가지로 작동합니다.

커미트되지 않은 읽기 응용프로그램이 앰비규어스 커서를 사용하는 경우 실행 시 CS 분리 레벨을 사용할 수 있습니다. PREP 또는 BIND 명령에서 BLOCKING 옵션의 값이 UNAMBIG(디폴트)인 경우 앰비규어스 커서는 CS로 에스컬레이션될 수 있습니다. 이 에스컬레이션을 방지하려면 다음을 수행하십시오.

- 응용프로그램의 커서를 명확한 것으로 수정하십시오. FOR READ ONLY절을 포함하여 SELECT문을 변경하십시오.
- 응용프로그램의 커서를 앰비규어스로 유지하지만 BLOCKING ALL 및 STATICREADONLY YES 옵션을 사용하여 프로그램을 프리컴파일하거나 바인드하여 프로그램 실행 시 앰비규어스 커서를 읽기 전용으로 처리하도록 하십시오.

분리 레벨 비교

표 1에서는 지원되는 분리 레벨을 요약합니다.

표 1. 분리 레벨 비교

	UR	**CS**	**RS**	**RR**
응용프로그램이 다른 응용프로그램 프로세스에서 수행한 커미트되지 않은 변경사항을 볼 수 있습니까?	예	아니오	아니오	아니오
응용프로그램이 다른 응용프로그램 프로세스에서 수행한 커미트되지 않은 변경사항을 갱신할 수 있습니까?	아니오	아니오	아니오	아니오
명령문의 재실행이 다른 응용프로그램 프로세스의 영향을 받을 수 있습니까? [1]	예	예	예	아니오[2]
갱신된 행을 다른 응용프로그램 프로세스에서 갱신할 수 있습니까? [3]	아니오	아니오	아니오	아니오
갱신된 행을 UR이 아닌 분리 레벨에서 실행 중인 다른 응용프로그램 프로세스에서 읽을 수 있습니까?	아니오	아니오	아니오	아니오
갱신된 행을 UR 분리 레벨에서 실행 중인 다른 응용프로그램 프로세스에서 읽을 수 있습니까?	예	예	예	예
액세스한 행을 다른 응용프로그램 프로세스에서 갱신할 수 있습니까? [4]	예	예	아니오	아니오
액세스한 행을 다른 응용프로그램 프로세스에서 읽을 수 있습니까?	예	예	예	예
갱신할 현재 행을 다른 응용프로그램 프로세스에서 갱신하거나 삭제할 수 있습니까? [5]	예/아니오[6]	예/아니오[6]	아니오	아니오

표 1. 분리 레벨 비교 (계속)

	UR	CS	RS	RR

주:

1. *팬텀 읽기 현상*의 예는 다음과 같습니다. 작업 단위 UW1은 일부 검색 조건을 충족시키는 *n*개의 행 세트를 읽습니다. 작업 단위 UW2는 동일한 검색 조건을 충족시키는 하나 이상의 행을 삽입한 후 커미트합니다. UW1이 동일한 검색 조건으로 읽기를 반복하는 경우 다른 결과 세트가 표시됩니다. 원래 읽은 행에 UW2에서 삽입한 행이 함께 표시됩니다.
2. 두 번의 읽기 사이에 레이블 기반 액세스 제어(LBAC) 증명서가 변경된 경우 다른 행에 액세스할 수 있으므로 두 번째 읽기의 결과가 다를 수 있습니다.
3. 응용프로그램이 테이블에서 읽고 테이블에 쓰는 경우 분리 레벨은 응용프로그램에 대한 보호 기능을 제공하지 않습니다. 예를 들어, 응용프로그램은 테이블에서 커서를 열고 동일한 테이블에서 삽입, 갱신 또는 삭제 조작을 수행합니다. 열린 커서에서 추가 행이 페치되는 경우 응용프로그램이 일치하지 않는 데이터를 볼 수도 있습니다.
4. *비반복 읽기 현상*의 예는 다음과 같습니다. 작업 단위 UW1이 한 행을 읽습니다. 작업 단위 UW2가 이 행을 수정하고 커미트합니다. UW1이 이후에 이 행을 다시 읽는 경우 다른 값을 볼 수도 있습니다.
5. *DR(Dirty Read)* 현상의 예는 다음과 같습니다. 작업 단위 UW1이 한 행을 수정합니다. 작업 단위 UW2가 UW1이 커미트하기 전에 해당 행을 읽습니다. UW1이 이후에 변경사항을 롤백하는 경우 UW2가 존재하지 않는 데이터를 읽었습니다.
6. UR 또는 CS에서 커서를 갱신할 수 없는 경우 다른 응용프로그램 프로세스에서 현재 행을 갱신하거나 삭제할 수도 있습니다. 예를 들어, 버퍼링으로 클라이언트의 현재 행이 서버의 현재 행과 다를 수 있습니다. 또한 CS에서 현재 커미트된 시맨틱을 사용하는 경우 읽고 있는 행에 보류 중인 커미트되지 않은 갱신사항이 포함될 수도 있습니다. 이 경우 현재 커미트된 행 버전은 항상 응용프로그램으로 리턴됩니다.

분리 레벨 요약

표 2에서는 여러 가지 다른 분리 레벨과 연관된 동시성 문제를 보여 줍니다.

표 2. 분리 레벨 요약

분리 레벨	커미트되지 않은 데이터에 대한 액세스	비반복 읽기	팬텀 읽기
반복 읽기(RR)	불가능	불가능	불가능
읽기 안정성(RS)	불가능	불가능	가능
커서 안정성(CS)	불가능	가능	가능
커미트되지 않은 읽기(UR)	가능	가능	가능

잠금을 확보하고 해제하는 데 필요한 처리 및 메모리 자원은 분리 레벨에 따라 다르므로 분리 레벨은 여러 응용프로그램들 간 분리 정도와 개별 응용프로그램의 성능에 영향을 줍니다. 교착 상태 가능성도 분리 레벨에 따라 다릅니다. 30 페이지의 표 3에서는 응용프로그램의 초기 분리 레벨을 선택하는 데 유용한 단순 발견법을 제공합니다.

표 3. 분리 레벨 선택 지침

응용프로그램 유형	높은 데이터 안정성 필요	높은 데이터 안정성 불필요
읽기 쓰기 트랜잭션	RS	CS
읽기 전용 트랜잭션	RR 또는 RS	UR

테이블 스페이스

*테이블 스페이스*는 테이블, 인덱스, 대형 오브젝트(LOB) 및 Long 데이터가 포함된 스토리지 구조입니다. 데이터베이스의 데이터를 데이터가 시스템에서 저장되는 위치와 관련된 논리적 스토리지 그룹으로 구성하는 데 사용됩니다. 테이블 스페이스는 데이터베이스 파티션 그룹에 저장됩니다.

테이블 스페이스를 사용하여 스토리지를 구성하면 다음과 같은 많은 이점이 있습니다.

복구 가능성

함께 백업 또는 리스토어되어야 하는 오브젝트를 동일한 테이블 스페이스에 넣으면 하나의 명령으로 테이블 스페이스의 모든 오브젝트를 백업 또는 리스토어할 수 있으므로 백업 및 리스토어 조작이 더욱 편리해집니다. 테이블 스페이스에 분산되는 파티션된 테이블 및 인덱스가 있는 경우 주어진 테이블 스페이스에 상주하는 데이터 및 인덱스 파티션만 백업 또는 리스토어할 수 있습니다.

더 많은 테이블

하나의 테이블 스페이스에 저장할 수 있는 테이블 수에 한계가 있습니다. 테이블 스페이스에 포함될 수 있는 것보다 많은 추가 테이블이 필요한 경우 해당 테이블을 위한 추가 테이블 스페이스만 작성하면 됩니다.

스토리지 유연성

DMS 및 SMS 테이블 스페이스의 경우 데이터를 저장하는 데 사용되는 스토리지 디바이스를 지정할 수 있습니다. 예를 들어 현재 조작 데이터는 더 빠른 디바이스에 상주하는 테이블 스페이스에 저장하고 실행기록 데이터는 더 느린 (그리고 덜 비싼) 디바이스에 상주하는 테이블 스페이스에 저장할 수 있습니다.

성능 또는 메모리 활용도 향상을 위해 버퍼 풀에 있는 데이터를 분리하는 기능

자주 쿼리되는 오브젝트(예: 테이블, 인덱스) 세트가 있는 경우 하나의 CREATE 또는 ALTER TABLESPACE문을 사용하여 버퍼 풀이 상주하는 테이블 스페이스를 지정할 수 있습니다. 임시 테이블 스페이스를 자체 버퍼 풀에 지정하여 정렬이나 조인 같은 활동을 성능을 향상시킬 수 있습니다. 일부 경우에는 자주 액세스되지 않는 데이터나 매우 큰 테이블에 대한 무작위 액세스가 필요한 응용프로그램에 대해 더 작은 버퍼 풀을 정의하는 것이 좋을 수 있습니다. 이 경우 단일 쿼리보다 오래 동안 데이터를 버퍼 풀에 보존할 필요가 없습니다.

테이블 스페이스는 하나 이상의 *컨테이너*로 구성됩니다. 컨테이너는 디렉토리 이름, 디바이스 이름 또는 파일 이름일 수 있습니다. 하나의 테이블 스페이스가 여러 개의 컨테

이너를 가질 수 있습니다. 다중 컨테이너(하나 이상의 테이블 스페이스의)가 동일한 실제 스토리지 디바이스에 작성될 수 있습니다(작성하는 각 컨테이너가 서로 다른 스토리지 디바이스를 사용하는 경우 최상의 성능을 얻을 수 있습니다). 자동 스토리지 테이블 스페이스를 사용 중인 경우 컨테이너 작성 및 관리는 데이터베이스 관리 프로그램에 의해 자동으로 처리됩니다. 자동 스토리지 테이블 스페이스를 사용하고 있지 않으면 사용자가 직접 컨테이너를 정의하고 관리해야 합니다.

그림 6은 데이터베이스 내의 테이블과 테이블 스페이스의 관계, 그리고 데이터베이스와 연관된 컨테이너를 나타냅니다.

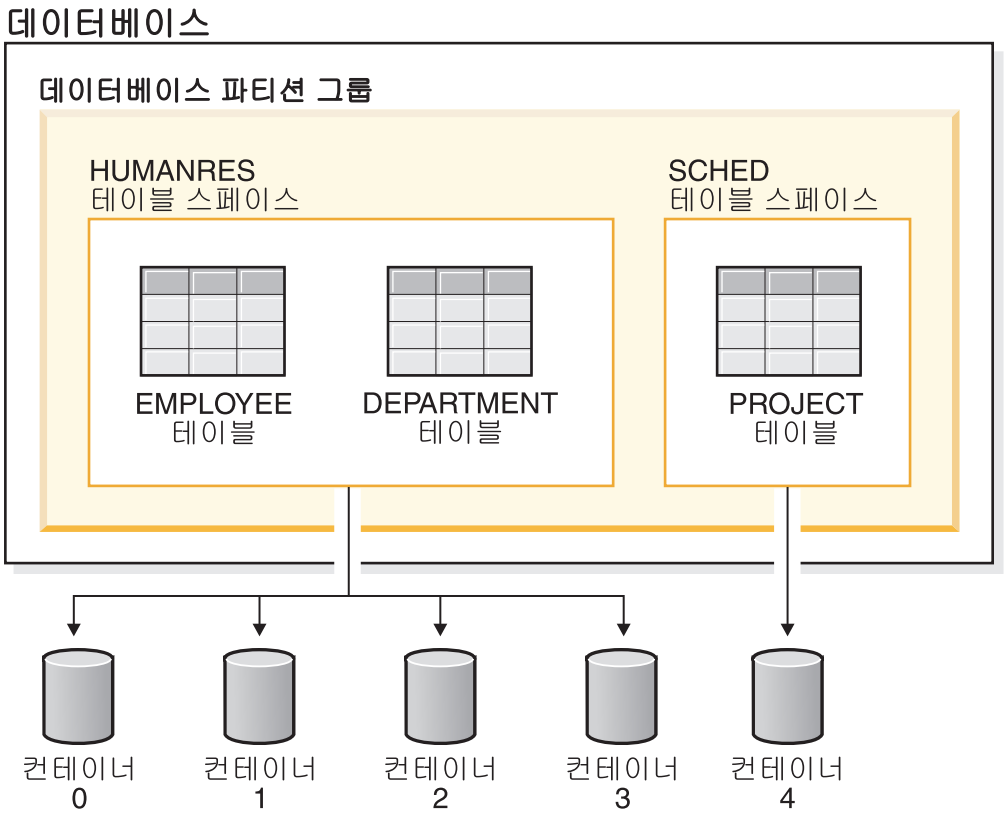


그림 6. 데이터베이스 내에 있는 테이블 스페이스 및 테이블

EMPLOYEE 및 DEPARTMENT 테이블은 컨테이너 0, 1, 2, 3 전체에 있는 HUMANRES 테이블 스페이스에 있습니다. PROJECT 테이블은 컨테이너 4의 SCHED 테이블 스페이스에 있습니다. 이 예는 별도의 디스크에 존재하는 각 컨테이너를 보여줍니다.

데이터베이스 관리 프로그램은 가능한 한 컨테이너 간에 균형 있게 데이터를 로드하려고 합니다. 결과적으로 모든 컨테이너가 데이터를 저장하는데 사용됩니다. 다른 컨테이너를 사용하기 전에 데이터베이스 관리 프로그램이 해당 컨테이너에 기록하는 페이지 수를 *Extent 크기*라고 합니다. 데이터베이스 관리 프로그램은 항상 첫 번째 컨테이너에 테이블 데이터를 저장하는 것은 아닙니다.

그림 7에서는 Extent 크기가 두 개의 4KB페이지이고, 4개의 컨테이너(각각에는 적은 수의 할당된 Extent가 있음)가 있는 HUMANRES 테이블 스페이스를 보여 줍니다. DEPARTMENT 및 EMPLOYEE 테이블은 둘 다 7 페이지를 가지고 있고 4개의 컨테이너 전체에 있습니다.

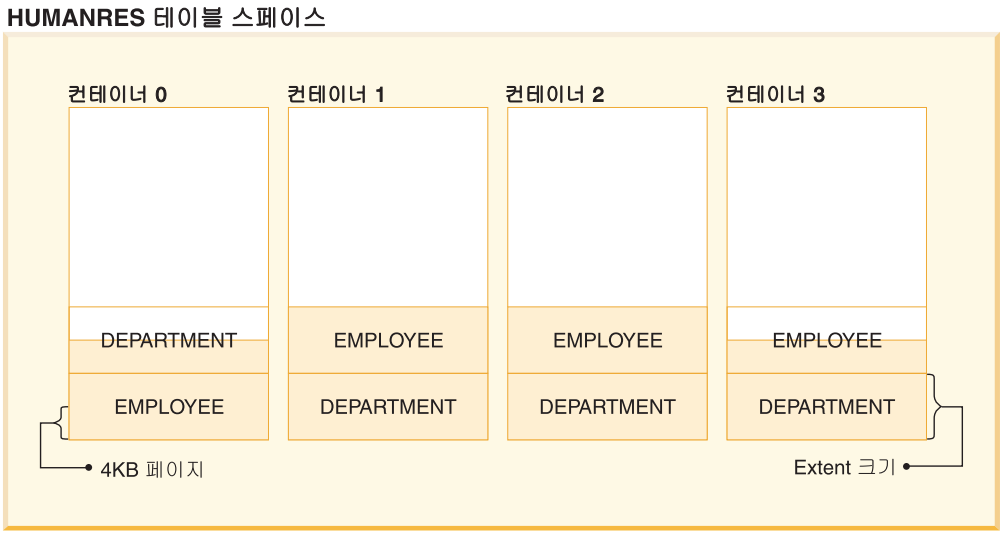


그림 7. 테이블 스페이스 내에 있는 컨테이너 및 Extent

문자 변환

*문자열*은 문자를 나타내는 바이트 시퀀스입니다. 문자열 내의 모든 문자는 공통적인 코딩 표현을 갖습니다. 어떤 경우에는 이 문자를 다른 코딩 표현으로 변환해야 할 수도 있습니다. 이 프로세스를 *문자 변환*이라고 합니다.

문자 변환이 필요한 경우 변환이 자동으로 수행됩니다. DB2 데이터베이스 서버 및 클라이언트가 자동으로 필요한 모든 문자 변환을 수행하므로 응용프로그램에서 명시적으로 문자 변환을 호출할 필요가 없습니다.

문자 변환은 SQL문이 리모트로 실행될 때 발생할 수 있습니다. 예를 들어, 코딩 표현이 송신 및 수신 시스템에서 다를 수 있는 다음 시나리오를 고려해 보십시오.

- 호스트 변수 값을 응용프로그램 리퀘스터(AR)에서 응용프로그램 서버(AS)로 보냅니다.
- 결과 컬럼의 값을 응용프로그램 서버(AS)에서 응용프로그램 리퀘스터(AR)로 보냅니다.

다음은 문자 변환 설명 시 사용되는 용어 목록입니다.

문자 세트

정의된 문자 세트. 예를 들어, 다음 문자 세트는 몇 개의 코드 페이지에 표시됩니다.

- 26개의 액센트 없는 문자 A - Z
- 26개의 액센트 없는 문자 a - z
- 숫자 0 - 9
- . , : ; ? ( ) ' " / - _ & + % * = < >

코드 페이지

코드 포인트에 지정된 문자 세트. 예를 들어 코드 페이지 850에 대한 ASCII 인코딩 스킴에서, "A"는 지정된 코드 포인트 X'41'이고 "B"는 지정된 코드 포인트 X'42'입니다. 코드 페이지 내에서, 각각의 코드 포인트는 단 하나의 특정 의미를 갖습니다. 코드 페이지는 데이터베이스의 속성입니다. 응용프로그램이 데이터베이스에 연결되면 데이터베이스 관리 프로그램이 응용프로그램의 코드 페이지를 결정합니다.

코드 포인트

문자를 나타내는 고유한 비트 패턴

인코딩 스킴

문자 데이터를 표시하기 위해 사용되는 규칙 세트. 예를 들면 다음과 같습니다.

- 1바이트 ASCII
- 1바이트 EBCDIC
- 2바이트 ASCII
- 혼합 1바이트 및 2바이트 ASCII

다음 그림은 일반 문자 세트가 두 가지의 다른 코드 페이지에서 다른 코드 포인트에 맵핑되는 방법을 보여줍니다. 인코딩 스킴이 동일해도, 다른 많은 코드 페이지가 있으며 동일한 코드 포인트가 다른 코드 페이지의 다른 문자를 표시할 수 있습니다. 또한, 문자열의 1바이트가 반드시 1바이트 문자 세트(SBCS)의 문자를 표시하는 것은 아닙니다. 문자열은 혼합 및 비트 데이터에도 사용됩니다. *혼합 데이터*는 1바이트, 2바이트 또는 멀티바이트 문자가 혼합된 데이터입니다. *비트 데이터*(FOR BIT DATA, BLOB로 정의된 컬럼이나 2진 문자열)는 문자 세트와 연관되지 않습니다.

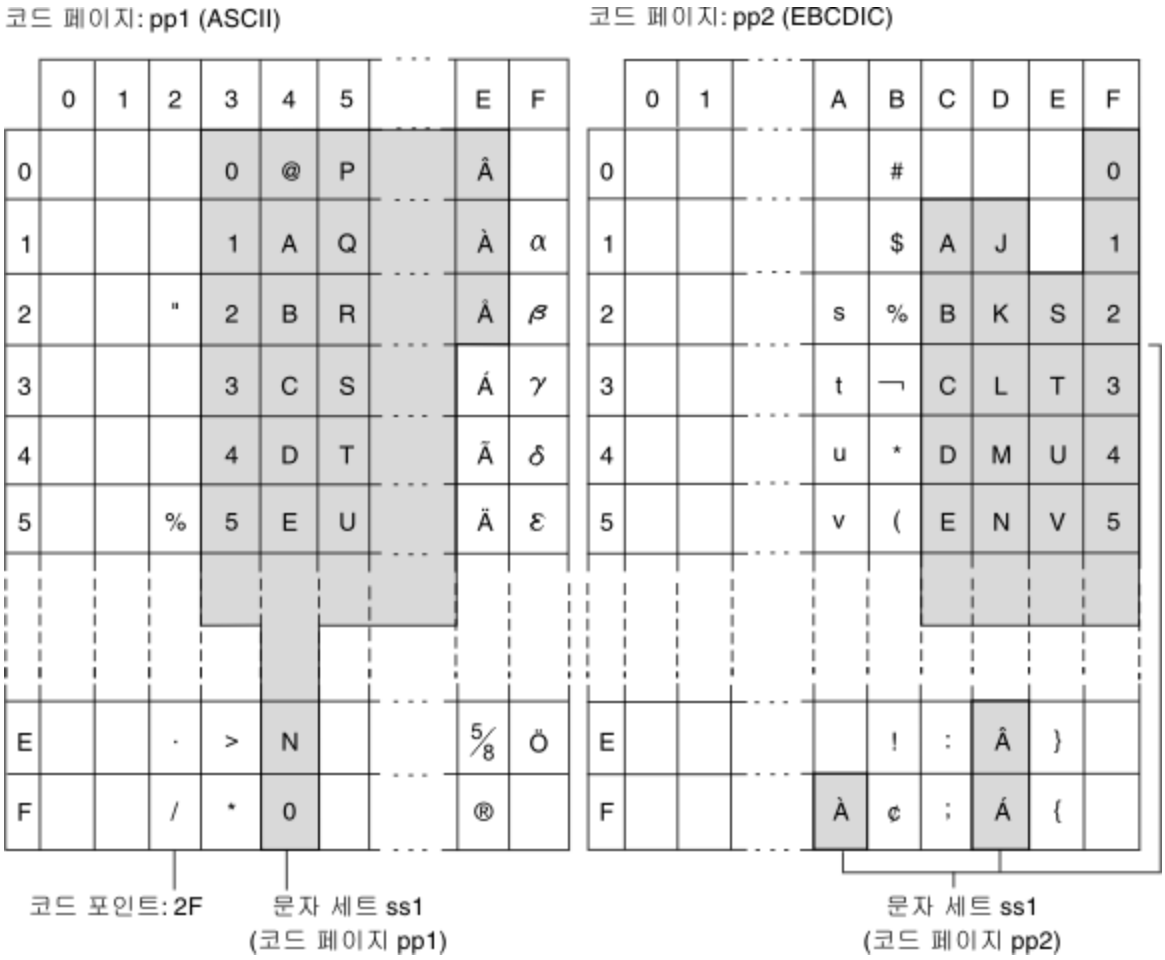


그림 8. 다른 코드 페이지에서 문자 세트 맵핑

데이터베이스 관리 프로그램은 응용프로그램이 데이터베이스에 바인드될 때 모든 문자열의 코드 페이지 속성을 판별합니다. 가능한 코드 페이지 속성은 다음과 같습니다.

데이터베이스 코드 페이지

데이터베이스 코드 페이지는 데이터베이스 구성 파일에 저장됩니다. 값은 데이터베이스가 작성될 때 지정되므로 변경할 수 없습니다.

응용프로그램 코드 페이지

응용프로그램이 실행되는 코드 페이지. 응용프로그램이 바인드된 동일한 코드 페이지일 필요는 없습니다.

섹션 코드 페이지

SQL문이 실행되는 코드 페이지. 일반적으로 섹션 코드 페이지는 데이터베이스 코드 페이지입니다. 그러나 유니코드 코드 페이지(UTF-8)는 다음의 경우 섹션 코드 페이지로 사용됩니다.

- 명령문이 유니코드가 아닌 데이터베이스에서 유니코드 인코딩 스킴으로 작성되는 테이블을 참조하는 경우

- 명령문이 유니코드가 아닌 데이터베이스에서 PARAMETER CCSID UNICODE로 정의되는 테이블 함수를 참조하는 경우

코드 페이지 0

FOR BIT DATA 값이나 BLOB 값을 포함하는 표현식에서 파생되는 문자열을 나타냅니다.

문자열 코드 페이지의 속성은 다음과 같습니다.

- 컬럼은 데이터베이스 코드 페이지, 유니코드 코드 페이지(UTF-8) 또는 코드 페이지 0(FOR BIT DATA 또는 BLOB로 정의된 경우)에 있을 수 있습니다.
- 상수 및 특수 레지스터(예: USER, CURRENT SERVER)는 섹션 코드 페이지에 있습니다. 상수는 필요하면 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환되며, 그 이후에 SQL문이 데이터베이스에 바인드될 때 섹션 코드 페이지로 변환됩니다.
- 입력 호스트 변수는 응용프로그램 코드 페이지에 있습니다. 버전 8에서와 같이, 입력 호스트 변수의 문자열 데이터는 필요한 경우 사용 전에 응용프로그램 코드 페이지에서 섹션 코드 페이지로 변환됩니다. 비트 데이터로 해석될 컨텍스트에서 호스트 변수가 사용되는 경우 예외가 발생합니다(예: 호스트 변수가 FOR BIT DATA로 정의된 컬럼에 지정될 경우).

스칼라 조작, 세트 조작 또는 병합과 같이 문자열 오브젝트를 조합하는 조작의 코드 페이지 속성을 판별하기 위해 규칙 세트가 사용됩니다. 코드 페이지 속성은 런타임 시 문자열의 코드 페이지 변환을 위한 요구사항을 판별하는 데 사용됩니다.

다문화 지원 및 SQL문

SQL문 코딩은 언어 영향을 받지 않습니다. SQL 키워드는 대문자, 소문자 또는 혼합 문자로 입력할 수 있어도, 표시된 대로 입력해야 합니다. SQL문에서 발생하는 데이터베이스 오브젝트, 호스트 변수 및 프로그램 레이블의 이름은 응용프로그램 코드 페이지에서 지원되는 문자여야 합니다.

서버는 파일 이름을 변환하지 않습니다. 파일 이름을 코딩하려면 ASCII 불변값 세트를 사용하거나, 파일 시스템에 실제로 저장된 16진수 값으로 경로를 제공하십시오.

멀티바이트 환경에는 불변 문자 세트에 속하지 않는 특수한 문자로 여겨지는 네 개의 문자가 있습니다. 이 문자는 다음과 같습니다.

- LIKE 처리에서 사용되는 2바이트 백분율 및 2바이트 밑줄 문자.
- 그래픽 문자열 및 기타 위치에서 공백을 채우는 데 사용되는 2바이트 공백 문자.
- 소스 코드 페이지와 목표 코드 페이지 사이의 맵핑이 존재하지 않는 경우 코드 페이지 변환 중에 교체로 사용되는 2바이트 대체 문자.

이러한 문자 각각에 대한 코드 포인트는 코드 페이지별로 다음과 같습니다.

표 4. 특수한 2바이트 문자에 대한 코드 포인트

코드 페이지	2바이트 백분율	2바이트 밑줄	2바이트 공백	2바이트 대체 문자
932	X'8193'	X'8151'	X'8140'	X'FCFC'
938	X'8193'	X'8151'	X'8140'	X'FCFC'
942	X'8193'	X'8151'	X'8140'	X'FCFC'
943	X'8193'	X'8151'	X'8140'	X'FCFC'
948	X'8193'	X'8151'	X'8140'	X'FCFC'
949	X'A3A5'	X'A3DF'	X'A1A1'	X'AFFE'
950	X'A248'	X'A1C4'	X'A140'	X'C8FE'
954	X'A1F3'	X'A1B2'	X'A1A1'	X'F4FE'
964	X'A2E8'	X'A2A5'	X'A1A1'	X'FDFE'
970	X'A3A5'	X'A3DF'	X'A1A1'	X'AFFE'
1381	X'A3A5'	X'A3DF'	X'A1A1'	X'FEFE'
1383	X'A3A5'	X'A3DF'	X'A1A1'	X'A1A1'
13488	X'FF05'	X'FF3F'	X'3000'	X'FFFD'
1363	X'A3A5'	X'A3DF'	X'A1A1'	X'A1E0'
1386	X'A3A5'	X'A3DF'	X'A1A1'	X'FEFE'
5039	X'8193'	X'8151'	X'8140'	X'FCFC'

유니코드 데이터베이스의 경우, GRAPHIC 공백은 X'0020'으로 eucJP(확장 UNIX 코드 - 일본) 및 eucTW (확장 UNIX 코드 - 대만) 데이터베이스에 사용되는 X'3000'의 GRAPHIC 공백과 다릅니다.

X'0020' 및 X'3000' 둘 다 유니코드 표준의 공백 문자입니다. GRAPHIC 공백 코드 포인트에서의 차이는 EUC 데이터베이스의 데이터와 유니코드 데이터베이스의 데이터를 비교할 때 고려해야 합니다.

분산 관계형 데이터베이스에 연결

분산 관계형 데이터베이스는 정규 요청자-서버 프로토콜 및 함수를 기반으로 빌드됩니다.

*응용프로그램 리퀘스터(AR)*는 연결의 응용프로그램 측면을 지원합니다. 응용프로그램 리퀘스터는 응용프로그램의 데이터베이스 요청을 분산 데이터베이스 네트워크에서 사용하기에 적합한 통신 프로토콜로 변환합니다. 이러한 요청은 연결의 다른 쪽 끝에 있는 *데이터베이스 서버*를 통해 수신되고 처리됩니다. 응용프로그램이 로컬 데이터베이스에 액세스 중인 것처럼 작동할 수 있도록 응용프로그램 리퀘스터(AR)와 데이터베이스 서버가 함께 작동하여 통신 및 위치 고려사항을 처리합니다.

테이블 또는 뷰를 참조하는 SQL문을 실행하려면 먼저 응용프로그램 프로세스를 데이터베이스 관리 프로그램의 응용프로그램 서버(AS)에 연결해야 합니다. CONNECT문은 응용프로그램 프로세스와 해당 서버 사이를 연결합니다.

CONNECT문에는 두 가지 유형이 있습니다.

- CONNECT(유형 1)는 작업 단위(UOW)(리모트 작업 단위(RUOW)) 시맨틱당 단일 데이터베이스를 지원합니다.
- CONNECT(유형 2)는 작업 단위(UOW)(응용프로그램 지향 분산 작업 단위(DUOW)) 시맨틱당 다중 데이터베이스를 지원합니다.

DB2 콜 레벨 인터페이스(CLI) 및 Embedded SQL은 *동시 트랜잭션*이라는 연결 모드를 지원하여 다중 연결이 가능하도록 하며 각각의 트랜잭션은 독립적인 트랜잭션입니다. 응용프로그램은 동일한 데이터베이스에 대한 여러 개의 동시 연결을 사용할 수 있습니다.

응용프로그램 서버(AS)는 프로세스가 시작된 환경에서 로컬이거나 리모트입니다. 해당 환경에서 분산 관계형 데이터베이스를 사용하고 있지 않더라도 응용프로그램 서버(AS)가 존재합니다. 이러한 환경에는 CONNECT문에서 식별할 수 있는 응용프로그램 서버(AS)에 대해 설명하는 로컬 디렉토리가 포함됩니다.

응용프로그램 서버(AS)는 테이블 또는 뷰를 참조하는 바운드 양식의 정적 SQL문을 실행합니다. 데이터베이스 관리 프로그램에서 바인드 조작을 통해 이전에 작성한 패키지에서 바운드 명령문을 가져옵니다.

대부분의 경우 응용프로그램 서버(AS)에 연결된 응용프로그램에서는 응용프로그램 서버(AS)의 데이터베이스 관리 프로그램이 지원하는 명령문 및 절을 사용할 수 있습니다. 해당 명령문 및 절의 일부를 지원하지 않는 데이터베이스 관리 프로그램의 응용프로그램 리퀘스터(AR)를 통해 응용프로그램을 실행 중인 경우에도 이들 명령문 및 절을 사용할 수 있습니다.

테이블, 파일 및 파이프에 쓰는 이벤트 모니터

일부 이벤트 모니터는 테이블, 파이프 또는 파일에 데이터베이스 이벤트에 관한 정보를 쓰도록 구성할 수 있습니다.

이벤트 모니터는 지정된 이벤트가 발생할 때 데이터베이스 및 모든 연결된 응용프로그램에 관한 정보를 수집하는 데 사용됩니다. 이벤트는 연결, 교착 상태, 명령문 또는 트랜잭션과 같은 데이터베이스 활동의 변화를 나타냅니다. 이벤트 모니터가 모니터하기 원하는 이벤트의 유형으로 이벤트 모니터를 정의할 수 있습니다. 예를 들어 교착 상태 이벤트 모니터는 교착 상태가 발생하기를 기다립니다. 교착 상태가 발생하면 관련된 응용프로그램 및 경합 상태의 잠금에 관한 정보를 수집합니다.

이벤트 모니터를 작성하려면 CREATE EVENT MONITOR SQL문을 사용하십시오. 이벤트 모니터는 사용 중일 때만 이벤트 데이터를 수집합니다. 이벤트 모니터를 활성화 또는 비활성화하려면 SET EVENT MONITOR STATE SQL문을 사용하십시오. 이벤트 모니터의 상태(활성 또는 비활성인지 여부)는 SQL 함수 EVENT_MON_STATE로 판별할 수 있습니다.

CREATE EVENT MONITOR SQL문이 실행될 때 작성하는 이벤트 모니터의 정의는 다음 데이터베이스 시스템 카탈로그 테이블에 저장됩니다.

- SYSCAT.EVENTMONITORS: 데이터베이스에 대해 정의된 이벤트 모니터
- SYSCAT.EVENTS: 데이터베이스에 대해 모니터되는 이벤트
- SYSCAT.EVENTTABLES: 테이블 이벤트 모니터에 대한 목표 테이블

각 이벤트 모니터는 모니터 요소에서 인스턴스 데이터의 고유한 개인용 논리적 뷰를 갖고 있습니다. 특정 이벤트 모니터가 비활성화된 후 다시 활성화되면 이들 카운터의 뷰가 재설정됩니다. 새로 활성화된 이벤트 모니터만 영향을 받습니다. 다른 모든 이벤트 모니터는 계속해서 카운터 값(더하기 모든 새 추가)의 뷰를 사용합니다.

이벤트 모니터 출력을 파티션되지 않은 SQL 테이블, 파일 또는 Named Pipe로 보낼 수 있습니다.

주: 사용되지 않는 상세한 교착 상태 이벤트 모니터인 DB2DETAILDEADLOCK이 각 데이터베이스에 대해 디폴트로 작성되고 해당 데이터베이스가 활성화될 때 시작합니다. 이 이벤트 모니터를 삭제하여 이 이벤트 모니터가 초래하는 오버헤드를 피하십시오. DB2DETAILDEADLOCK 모니터 요소의 사용은 더 이상 권장하지 않습니다. 이 사용되지 않는 이벤트 모니터는 추후 릴리스에서 제거될 수 있습니다. CREATE EVENT MONITOR FOR LOCKING문을 사용하여 잠금 시간종료, 잠금 대기 및 교착 상태 같은 잠금 관련 이벤트를 모니터하십시오.

다중 데이터베이스 파티션 전반에 데이터베이스 파티셔닝

데이터베이스 관리 프로그램은 파티션된 데이터베이스의 다중 데이터베이스 파티션(노드) 전반에 데이터 전개 시 강한 유연성을 허용합니다. 사용자는 분산 키를 선언하여 데이터를 분배하는 방법을 선택할 수 있고 데이터가 저장되는 데이터베이스 파티션 그룹 및 테이블 스페이스를 선택하여 테이블 데이터를 전개할 수 있는 데이터베이스 파티션과 해당 파티션 수를 판별할 수 있습니다.

또한 분산 맵(갱신 가능)은 분산 키 값 대 데이터베이스 파티션의 맵핑을 지정합니다. 결과적으로, 대형 테이블의 파티션된 데이터베이스에서 유연한 워크로드 병렬화가 가능해지고 응용프로그램 설계자가 소형 테이블이 하나 또는 소수의 데이터베이스 파티션에

저장되도록 선택하는 경우 이와 같이 처리되게 합니다. 각 로컬 데이터베이스 파티션에는 고성능 로컬 데이터 액세스를 제공하기 위해 저장하는 데이터에 대한 로컬 인덱스가 있을 수 있습니다.

파티션된 데이터베이스에서 분산 키는 데이터베이스 파티션 세트에 테이블 데이터를 분배하는 데 사용됩니다. 또한 인덱스 데이터는 대응하는 테이블로 파티션되고 각 데이터베이스 파티션에 로컬로 저장됩니다.

데이터베이스 파티션이 데이터를 저장하는 데 사용되기 위해서는 먼저 데이터베이스 관리 프로그램에 정의되어 있어야 합니다. 데이터베이스 파티션은 `db2nodes.cfg`라는 파일에 정의됩니다.

파티션된 데이터베이스 파티션 그룹의 테이블 스페이스에 있는 테이블의 분산 키는 CREATE TABLE문 또는 ALTER TABLE문에 지정되어 있습니다. 지정되지 않은 경우, 1차 키의 첫 번째 컬럼에서 테이블 분산 키가 디폴트로 작성됩니다. 1차 키가 정의되지 않은 경우, 디폴트 분산 키는 long 또는 LOB 데이터 유형이 아닌 다른 데이터 유형을 가지고 있는 테이블에 정의된 첫 번째 컬럼입니다. 파티션된 데이터베이스의 테이블에는 long이나 LOB 데이터 유형이 아닌 최소 하나 이상의 컬럼이 있어야 합니다. 단일 파티션 데이터베이스 파티션 그룹에 있는 테이블 스페이스의 테이블에 분산 키가 명시적으로 지정되어 있는 경우에만 분산 키가 있습니다.

행은 다음과 같이 데이터베이스 파티션에 배치됩니다.

1. 해싱 알고리즘(데이터베이스 파티셔닝 기능)이 모든 분산 키 컬럼에 적용되고, 그 결과 분산 맵 인덱스 값이 생성됩니다.
2. 분산 맵에서 해당 인덱스 값의 데이터베이스 파티션 번호는 행이 저장되는 데이터베이스 파티션을 식별합니다.

데이터베이스 관리 프로그램은 *부분 클러스터링 해제*를 지원하는데, 이 기능은 테이블이 시스템의 데이터베이스 파티션 서브세트(즉, 데이터베이스 파티션 그룹)에 분배될 수 있습니다. 테이블을 시스템의 모든 데이터베이스 파티션 전반에 분배해야 할 필요가 없습니다.

데이터베이스 관리 프로그램은 조인 또는 서브쿼리를 위해 액세스되는 데이터가 동일한 데이터베이스 파티션 그룹의 동일한 데이터베이스에 위치하는 경우 인식할 수 있습니다. 이 기능은 *테이블 공동 배치*라고 합니다. 공동 배치된 테이블에서 동일한 분산 키 값을 갖는 행은 동일한 데이터베이스 파티션에 배치됩니다. 데이터베이스 관리 프로그램은 데이터가 저장되어 있는 데이터베이스 파티션에서 조인 또는 서브쿼리 처리를 수행하도록 선택할 수 있습니다. 이 기능으로 성능이 현저하게 향상될 수 있습니다.

공동 배치된 테이블은 다음과 같아야 합니다.

- 재분배 중이 아닌 동일한 데이터베이스 파티션 그룹에 배치됩니다. (재분배 중에 데이터베이스 파티션 그룹의 테이블은 다른 분산 맵을 사용 중일 수 있고, 해당 테이블은 공동 배치되지 않습니다.)
- 컬럼과 같은 수의 분산 키를 가지고 있습니다.
- 데이터베이스 파티션과 호환 가능한 분산 키의 대응하는 컬럼을 가지고 있습니다.
- 동일한 데이터베이스 파티션에 정의된 단일 파티션 데이터베이스 파티션 그룹에 있습니다.

파티션된 테이블의 대형 오브젝트(LOB) 동작

파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 범위 또는 호출된 데이터 파티션이라고 하는 복수의 스토리지 오브젝트에 나누어져 있는 데이터 조직 스킴을 사용합니다. 제공된 테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 제공된 스펙에 따라 복수의 스토리지 오브젝트로 파티션됩니다. 이러한 스토리지 오브젝트는 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘 조합 모두에 있을 수 있습니다.

파티션된 테이블에 대한 대형 오브젝트는 디폴트로 이에 대응하는 데이터 오브젝트와 동일한 테이블 스페이스에 저장됩니다. 이는 단 하나의 테이블 스페이스를 사용하거나 다중 테이블 스페이스를 사용하는 파티션된 테이블에 적용됩니다. 파티션된 테이블의 데이터가 다중 테이블 스페이스에 저장되어 있는 경우, 대형 오브젝트(LOB) 데이터도 또한 다중 테이블 스페이스에 저장됩니다.

CREATE TABLE문의 LONG IN절을 사용하여 이 디폴트 동작을 겹쳐쓰십시오. Long 데이터가 저장되는 테이블의 테이블 스페이스 목록을 지정할 수 있습니다. 디폴트 동작을 겹쳐쓰도록 선택하는 경우, LONG IN절에 지정된 테이블 스페이스가 대형 테이블 스페이스여야 합니다. 하나 이상의 데이터 파티션의 독립된 테이블 스페이스에 Long 데이터가 저장되도록 지정하는 경우 테이블의 모든 데이터 파티션에 대해 해당 조치를 수행해야 합니다. 즉, 어떤 데이터 파티션에는 리모트로 다른 데이터 파티션에는 로컬로 Long 데이터가 저장되게 할 수 없습니다. 디폴트 동작을 겹쳐쓰기 위해 디폴트 동작을 사용하든 LONG IN절을 사용하든 각 데이터 파티션에 대응하는 Long 오브젝트가 작성됩니다. SMS 테이블 스페이스의 Long 데이터는 이것이 속해 있는 데이터 오브젝트와 같은 테이블 스페이스에 상주해야 합니다. 각 데이터 파티션에 대응하는 Long 데이터 오브젝트를 저장하는 데 사용된 모든 테이블 스페이스는 동일한 페이지 *크기*, Extent *크기*, 스토리지 메커니즘(DMS 또는 SMS) 및 유형(일반 또는 대형)을 가지고 있어야 합니다. 리모트 대형 테이블 스페이스는 LARGE 유형이어야 하고 SMS일 수 없습니다.

예를 들어, 다음 CREATE TABLE문은 데이터와 같은 테이블 스페이스에 각 데이터 파티션의 CLOB 데이터에 대한 오브젝트를 작성합니다.

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

LONG IN을 사용하여 데이터가 있는 테이블 스페이스와는 다른 하나 이상의 대형 테이블 스페이스에 CLOB 데이터를 배치할 수 있습니다.

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

주: 테이블 레벨에서는 각 데이터 파티션에 대해 LONG IN절이 하나만 허용됩니다.

DB2 페더레이티드 시스템

페더레이티드 시스템

*페더레이티드 시스템*은 특수한 유형의 분산 데이터베이스 관리 시스템(DBMS)입니다. 페더레이티드 시스템은 페더레이티드 서버로 작동하는 DB2 인스턴스, 페더레이티드 데이터베이스 역할을 하는 데이터베이스, 하나 이상의 데이터 소스 및 데이터베이스와 데이터 소스에 액세스하는 클라이언트(사용자 및 응용프로그램)로 구성됩니다.

페더레이티드 시스템을 통해 분산 요청(DR)을 단일 SQL문으로 다중 데이터 소스에 보낼 수 있습니다. 예를 들어, 단일 SQL문으로 DB2 테이블, Oracle 테이블 및 XML 태그 파일에 위치한 데이터를 조인시킬 수 있습니다. 다음 그림에는 액세스할 수 있는 페더레이티드 시스템 구성요소 및 데이터 소스 샘플이 표시되어 있습니다.

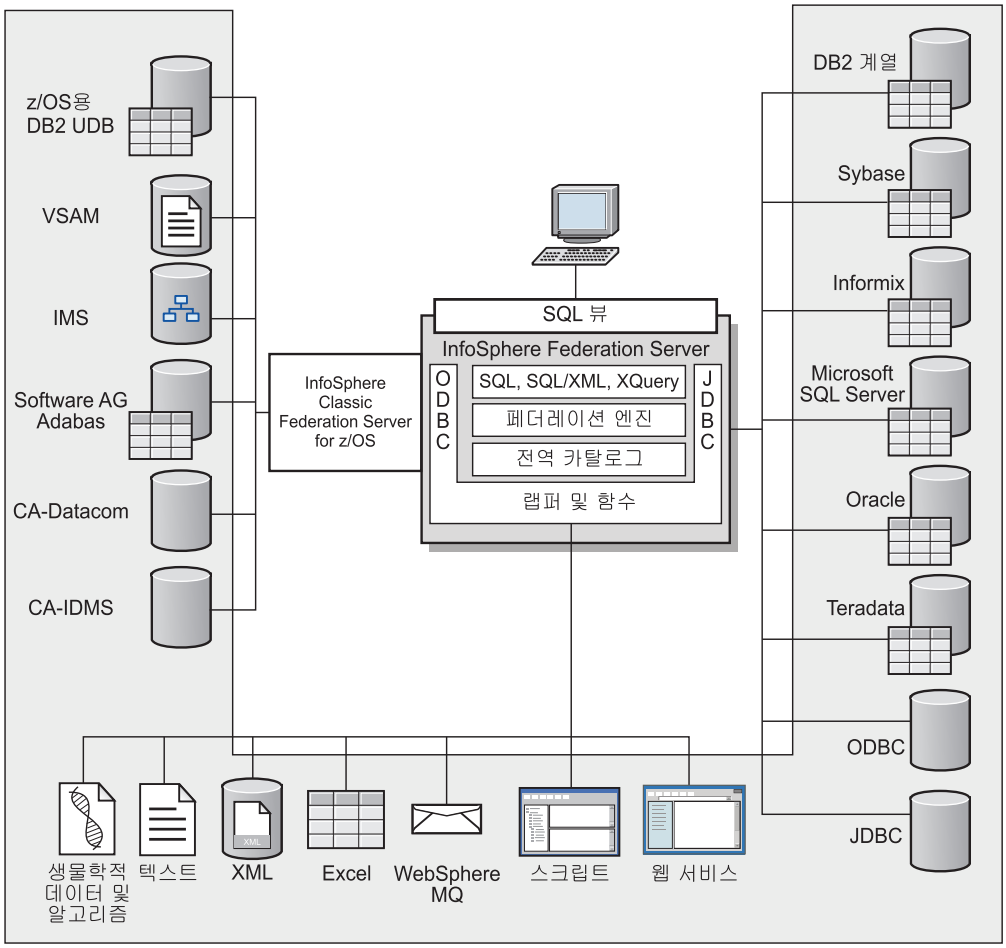


그림 9. 페더레이티드 시스템 구성요소

페더레이티드 시스템에는 다음과 같은 기능이 있어 매우 유용합니다.

- 모든 데이터가 페더레이티드 데이터베이스에 로컬로 저장되어 있는 것처럼 로컬 테이블 및 리모트 데이터 소스의 데이터를 상관시킴
- 데이터가 페더레이티드 데이터베이스에 저장되어 있는 것처럼 관계형 데이터 소스의 데이터 갱신
- 관계형 데이터 소스 간에 데이터 이동
- 데이터 소스를 처리하도록 데이터 소스에 요청을 보냄으로써 데이터 소스 처리 능력 사용
- 페더레이티드 서버에서 분산 요청(DR)의 일부를 처리하여 데이터 소스의 SQL 제한사항 보완

데이터 소스의 개념

페더레이티드 시스템에서 *데이터 소스*는 관계형 데이터베이스(예: Oracle이나 Sybase) 또는 관계형이 아닌 데이터 소스(예: XML 태그 파일) 중 하나입니다.

몇몇 데이터 소스를 통해 다른 데이터 소스에 액세스할 수 있습니다. 예를 들어, ODBC 랩퍼를 사용하여 z/OS용 IBM InfoSphere™ Classic Federation Server 데이터 소스(예: z/OS용 DB2, IMS™, CA-IDMS, CA-Datacom, Software AG Adabas 및 VSAM)에 액세스할 수 있습니다.

데이터 소스에 액세스하는 데 사용되는 방법이나 프로토콜은 데이터 소스 유형에 따라 다릅니다. 예를 들어, DRDA는 DB2 for z/OS 데이터 소스에 액세스하는 데 사용됩니다.

데이터 소스는 독립적입니다. 예를 들면, Oracle 응용프로그램이 데이터 소스에 액세스하는 것과 동시에 페더레이티드 서버가 Oracle 데이터 소스에 쿼리를 보낼 수 있습니다. 페더레이티드 시스템은 무결성 및 잠금 제한조건을 넘어 기타 데이터 소스에 액세스하는 것을 독점하거나 제한하지 않습니다.

페더레이티드 데이터베이스

일반 사용자 및 클라이언트 응용프로그램에는 데이터 소스가 DB2 데이터베이스 시스템의 단일 집합 데이터베이스로 표시됩니다. 사용자 및 응용프로그램은 페더레이티드 서버가 관리하는 *페더레이티드 데이터베이스*와 연결됩니다.

페더레이티드 데이터베이스에는 데이터에 대한 정보를 저장하는 시스템 카탈로그가 포함되어 있습니다. 페더레이티드 데이터베이스 시스템 카탈로그에는 데이터 소스 및 해당 특성을 식별하는 엔트리가 들어 있습니다. 페더레이티드 서버는 페더레이티드 데이터베이스 시스템 카탈로그 및 데이터 소스 랩퍼에 저장된 정보를 참조하여 SQL문을 처리하는 데 가장 좋은 플랜을 판별합니다.

페더레이티드 시스템은 데이터 소스의 데이터가 일반 관계형 테이블 또는 페더레이티드 데이터베이스의 뷰에 있었던 것처럼 SQL문을 처리합니다. 그 결과는 다음과 같습니다.

- 페더레이티드 시스템이 관계형 데이터와 관계형이 아닌 형식의 데이터를 상관시킬 수 있습니다. 데이터 소스에서 여러 SQL 통용어가 사용되거나 SQL을 전혀 지원하지 않는 경우에도 상관이 가능합니다.
- 페더레이티드 데이터베이스의 특성과 데이터 소스의 특성 사이에 다른점이 있는 경우 페더레이티드 데이터베이스 특성이 우선시됩니다. DB2 이외의 데이터 소스의 데이터를 사용하여 쿼리 결과를 계산하는 경우에도 쿼리 결과는 DB2 시맨틱을 준수합니다.

  예를 들어, 다음과 같습니다.

  – 페더레이티드 서버에서 사용되는 코드 페이지는 데이터 소스에서 사용되는 코드 페이지와 다릅니다. 이 경우, 데이터 소스의 문자 데이터는 해당 데이터가 페더레이티드 사용자에게 리턴되는 경우 페더레이티드 데이터베이스에서 사용되는 코드 페이지에 따라 변환됩니다.

  - 페더레이티드 서버에서 사용되는 조합 시퀀스는 데이터 소스에서 사용되는 조합 시퀀스와 다릅니다. 이 경우, 문자 데이터에 대한 정렬 조작은 데이터 소스가 아니라 페더레이티드 서버에서 수행됩니다.

SQL 컴파일러

DB2 SQL 컴파일러는 쿼리 처리에 유용한 정보를 수집합니다.

데이터 소스에서 데이터를 가져오기 위해 사용자 및 응용프로그램이 SQL에서 페더레이티드 데이터베이스에 쿼리를 제출합니다. 쿼리가 제출되면 DB2 SQL 컴파일러가 전역 카탈로그 및 데이터 소스 랩퍼의 정보를 참조하여 쿼리 처리를 돕습니다. 해당 정보에는 데이터 소스에 연결하는 작업, 서버 정보, 맵핑, 인덱스 정보 및 통계 처리에 대한 정보가 포함됩니다.

랩퍼 및 랩퍼 모듈

*랩퍼*는 페더레이티드 데이터베이스가 데이터 소스와 상호작용하는 메커니즘입니다. 페더레이티드 데이터베이스는 *랩퍼 모듈*이라는 라이브러리에 저장된 루틴을 사용하여 랩퍼를 구현합니다.

이러한 루틴을 통해 페더레이티드 데이터베이스가 대화식으로 데이터 소스에 연결하고 데이터 소스에서 데이터를 검색하는 등의 조작을 수행할 수 있습니다. 일반적으로 페더레이티드 인스턴스 소유자는 CREATE WRAPPER문을 사용하여 페더레이티드 데이터베이스에서 랩퍼를 등록합니다. DB2_FENCED 랩퍼 옵션을 사용하여 랩퍼를 분리(fenced) 또는 트러스트된 랩퍼로 등록할 수 있습니다.

액세스하려는 데이터 소스의 유형마다 하나의 랩퍼를 작성합니다. 예를 들어, 3개의 z/OS용 DB2 데이터베이스 테이블, 하나의 System i용 DB2 테이블, 2개의 Informix 테이블 및 하나의 Informix 뷰에 액세스하려고 합니다. 이 경우, DB2 데이터 소스 오브젝트에서 사용할 랩퍼를 한 개 작성하고 Informix 데이터 소스 오브젝트에서 사용할 랩퍼를 한 개 작성해야 합니다. 이들 랩퍼를 페더레이티드 데이터베이스에서 등록한 후 등록한 랩퍼를 사용하여 해당 데이터 소스의 기타 오브젝트에 액세스할 수 있습니다. 예를 들면, DRDA 랩퍼를 모든 DB2 제품군 데이터 소스 오브젝트와 함께 사용할 수 있습니다(예: Linux, UNIX 및 Windows용 DB2 데이터베이스, z/OS용 DB2, System i용 DB2 및 VM과 VSE용 DB2 서버).

서버 정의 및 별칭을 사용하여 각 데이터 소스 오브젝트의 세부사항(이름, 위치 등)을 식별합니다.

랩퍼는 여러 태스크를 수행합니다. 그 중 일부 태스크는 다음과 같습니다.

- 데이터 소스에 연결합니다. 랩퍼는 데이터 소스의 표준 연결 API를 사용합니다.
- 데이터 소스에 쿼리를 제출합니다.
  - 데이터 소스가 SQL을 지원하는 경우 SQL로 쿼리를 제출합니다.

  - 데이터 소스가 SQL을 지원하지 않는 경우 쿼리를 소스의 원시(native) 쿼리 언어 또는 일련의 소스 API 호출로 변환합니다.
- 데이터 소스에서 결과 세트를 수신합니다. 랩퍼는 결과 세트를 수신하기 위해 데이터 소스 표준 API를 사용합니다.
- 데이터 소스의 디폴트 데이터 유형 맵핑에 대한 페더레이티드 데이터베이스 쿼리에 응답합니다. 랩퍼에는 데이터 소스 오브젝트에 대한 별칭 작성 시 사용되는 디폴트 유형 맵핑이 포함되어 있습니다. 관계형 랩퍼의 경우, 사용자가 작성하는 데이터 유형 맵핑이 디폴트 데이터 유형 맵핑을 겹쳐씁니다. 사용자 정의 데이터 유형 맵핑은 전역 카탈로그에 저장됩니다.
- 데이터 소스의 디폴트 함수 맵핑에 대한 페더레이티드 데이터베이스 쿼리에 응답합니다. 페더레이티드 데이터베이스에는 쿼리 계획에 사용할 데이터 유형 맵핑 정보가 필요합니다. 랩퍼에는 페더레이티드 데이터베이스가 DB2 함수가 데이터 소스의 함수에 맵핑되는지 여부 및 함수를 맵핑하는 방법을 판별하는 데 필요한 정보가 포함되어 있습니다. SQL 컴파일러가 이 정보를 사용하여 데이터 소스가 쿼리 조작을 수행할 수 있는지 여부를 판별합니다. 관계형 랩퍼의 경우, 사용자가 작성하는 함수 맵핑이 디폴트 함수 유형 맵핑을 겹쳐씁니다. 사용자 정의 함수(UDF) 맵핑은 전역 카탈로그에 저장됩니다.

*랩퍼 옵션*은 랩퍼를 구성하거나 IBM InfoSphere Federation Server에서 랩퍼를 사용하는 방법을 정의하는 데 사용됩니다.

서버 정의 및 서버 옵션

데이터 소스에서 사용하도록 랩퍼를 작성한 후 페더레이티드 인스턴스 소유자가 페더레이티드 데이터베이스에 데이터 소스를 정의합니다.

인스턴스 소유자는 데이터 소스 및 데이터 소스와 관련된 기타 정보를 식별하기 위한 이름을 입력합니다. 이 정보에 포함되는 내용은 다음과 같습니다.

- 데이터 소스의 유형 및 버전
- 데이터 소스의 데이터베이스 이름(RDBMS 전용)
- 데이터 소스에 대한 메타데이터

예를 들어, DB2 제품군 데이터 소스는 여러 데이터베이스를 가질 수 있습니다. 정의는 페더레이티드 서버를 연결할 수 있는 데이터베이스를 지정해야 합니다. 이에 비하여 Oracle 데이터 소스에는 데이터베이스가 하나 있으므로 데이터베이스 이름을 몰라도 페더레이티드 서버를 데이터베이스에 연결할 수 있습니다. 데이터베이스 이름은 Oracle 데이터 소스의 페더레이티드 서버 정의에 포함되지 않습니다.

인스턴스 소유자가 페더레이티드 서버에 입력하는 이름 및 기타 정보를 함께 *서버 정의*라고 합니다. 데이터 소스는 데이터 요청에 응답하며 본질적으로 서버입니다.

서버 정의를 작성하고 수정하는 데 CREATE SERVER 및 ALTER SERVER문을 사용합니다.

서버 정의에 포함된 일부 정보는 *서버 옵션*으로 저장됩니다. 서버 정의 작성 시 서버에 대해 지정할 수 있는 옵션을 이해하는 것이 중요합니다.

데이터 소스에 연속적으로 연결할 때 서버 옵션이 유지되도록 설정하거나 단일 연결의 지속기간에 적합하게 설정할 수 있습니다.

사용자 맵핑

*사용자 맵핑*은 페더레이티드 서버의 권한 부여 ID와 리모트 데이터 소스에 연결하는 데 필요한 정보 간의 연관입니다.

사용자 맵핑을 작성하려면 CREATE USER MAPPING문을 사용하십시오. 이 명령문에서 로컬 권한 부여 ID, 서버 정의에서 지정한 리모트 데이터 소스 서버의 로컬 이름과 리모트 ID 및 암호를 지정합니다.

예를 들어, 리모트 서버에 대한 서버 정의를 작성했으며 'argon'을 리모트 서버의 로컬 이름으로 지정했다고 가정합니다. Mary에게 리모트 서버에 대한 액세스 권한을 부여하려면 다음 사용자 맵핑을 작성하십시오.

```
CREATE USER MAPPING FOR Mary
SERVER argon
OPTIONS (REMOTE_AUTHID 'remote_ID', REMOTE_PASSWORD 'remote_pw')
```

Mary가 SQL문을 발행하여 리모트 서버에 연결하면 페더레이티드 서버가 다음 단계를 수행합니다.

1. Mary의 사용자 맵핑 검색
2. 리모트 서버와 연관된 리모트 암호 'remote_pw' 암호 해독
3. 리모트 서버에 연결할 랩퍼 호출
4. 리모트 ID 'remote_ID' 및 암호 해독된 리모트 암호를 랩퍼에 전달
5. Mary가 사용할 수 있도록 리모트 서버에 연결

디폴트로, 페더레이티드 서버는 사용자 맵핑을 전역 카탈로그의 SYSCAT.USEROPTIONS 뷰에 저장하고 리모트 암호를 암호화합니다. 다른 방법으로, 파일이나 LDAP 서버와 같은 외부 저장소를 사용하여 사용자 맵핑을 저장할 수도 있습니다. 페더레이티드 서버와 외부 저장소 간의 인터페이스를 제공하려면 사용자 맵핑 플러그인을 작성하십시오.

사용자 맵핑 저장 방법에 상관 없이 사용자 맵핑에 액세스하는 것을 주의해서 제한하십시오. 사용자 맵핑이 손상되는 경우 리모트 데이터베이스에 있는 데이터가 권한이 부여되지 않은 활동에 취약해집니다.

별칭 및 데이터 소스 오브젝트

*별칭*은 액세스하려는 데이터 소스 오브젝트를 식별하는 데 사용되는 ID입니다. 별칭으로 식별되는 오브젝트를 *데이터 소스 오브젝트*라고 합니다.

별명이 대체 이름인 것과 달리 별칭은 데이터 소스 오브젝트의 대체 이름이 아닙니다. 별칭은 페더레이티드 서버가 오브젝트를 지칭하는 포인터입니다. 일반적으로 특정 별칭 컬럼 옵션 및 별칭 옵션과 함께 CREATE NICKNAME문을 사용하여 별칭을 정의합니다.

클라이언트 응용프로그램 또는 사용자가 페더레이티드 서버에 분산 요청(DR)을 제출하는 경우 요청에서 데이터 소스를 지정할 필요가 없습니다. 대신에 요청에서 데이터 소스 오브젝트를 별칭으로 칭합니다. 별칭은 데이터 소스의 특정 오브젝트에 맵핑됩니다. 이러한 맵핑을 수행하면 데이터 소스 이름으로 별칭을 규정할 필요가 없습니다. 클라이언트 응용프로그램 또는 사용자는 데이터 소스 오브젝트의 위치를 명확히 알 수 있습니다.

*DEPT*라는 별칭을 정의하여 *NFX1.PERSON*이라는 Informix 데이터베이스 테이블을 나타낸다고 가정합니다. SELECT * FROM *DEPT*문을 페더레이티드 서버에서 사용할 수 있습니다. 그러나 *NFX1.PERSON*이라는 페더레이티드 서버에 로컬 테이블이 없으면 SELECT * FROM*NFX1.PERSON*문을 페더레이티드 서버에서 사용할 수 없습니다 (pass-through 세션에서는 예외).

데이터 소스 오브젝트의 별칭 작성 시 오브젝트에 대한 메타데이터가 전역 카탈로그에 추가됩니다. 쿼리 옵티마이저는 랩퍼에 있는 정보 외에도 이 메타데이터를 사용하여 데이터 소스 오브젝트에 액세스할 수 있도록 합니다. 예를 들어, 별칭이 인덱스가 있는 테이블의 별칭인 경우 전역 카탈로그에는 인덱스에 대한 정보가 포함되고 랩퍼에는 DB2 데이터 유형과 데이터 소스 데이터 유형 간의 맵핑이 포함됩니다.

레이블 기반 액세스 제어(LBAC)를 사용하는 오브젝트의 별칭은 캐시되지 않습니다. 따라서 오브젝트의 데이터가 안전하게 유지됩니다. 예를 들어, Oracle(Net8) 랩퍼를 사용하여 Oracle Label Security를 사용하는 테이블의 별칭을 작성하는 경우 자동으로 테이블이 안전한 것으로 식별됩니다. 작성되는 별칭 데이터는 캐시할 수 없습니다. 따라서 해당 테이블에서 구체화된 쿼리 테이블을 작성할 수 없습니다. LBAC을 사용하면 적절한 보안 특권을 가진 사용자만 정보를 볼 수 있습니다. LBAC 지원 이전에 작성된 별칭의 경우 캐싱을 승인하지 않으려면 ALTER NICKNAME문을 사용해야 합니다. LBAC은 DRDA(Linux, UNIX 및 Windows용 DB2 버전 9.1 이상을 사용하는 데이터 소스의 경우) 및 Net8 랩퍼 모두에서 지원됩니다.

별칭 컬럼 옵션

전역 카탈로그에 별칭이 지정된 오브젝트에 대한 추가 메타데이터 정보를 입력할 수 있습니다. 이 메타데이터는 데이터 소스 오브젝트의 특정 컬럼에 있는 값에 대해 설명합니다. 이 메타데이터를 *별칭 컬럼 옵션*이라는 매개변수에 지정하십시오.

별칭 컬럼 옵션은 컬럼의 데이터를 일반적으로 처리하는 방법과 다르게 처리하도록 랩퍼에 알립니다. SQL 컬파일러와 쿼리 옵티마이저는 메타데이터를 사용하여 데이터에 액세스하는 개선된 플랜을 개발합니다.

별칭 컬럼 옵션 또한 랩퍼에 기타 정보를 제공하는 데 사용됩니다. 예를 들어 XML 데이터 소스의 경우, 랩퍼가 XML 문서의 컬럼을 구문 분석할 때 사용할 XPath 표현식을 랩퍼에 알리기 위해 별칭 컬럼 옵션을 사용합니다.

페더레이션을 적용하면 DB2 서버는 별칭이 가리키는 데이터 소스 오브젝트를 마치 로컬 DB2 테이블인 것처럼 처리합니다. 따라서 별칭을 작성한 모든 데이터 소스 오브젝트에 별칭 컬럼 옵션을 설정할 수 있습니다. 일부 별칭 컬럼 옵션은 특정 유형의 데이터 소스에서 사용되도록 설계되어 해당 데이터 소스에만 적용할 수 있습니다.

데이터 소스에 페더레이티드 데이터베이스 조합 시퀀스와 다른 조합 시퀀스가 있다고 가정합니다. 페더레이티드 서버는 일반적으로 데이터 소스에서 문자 데이터가 포함된 컬럼을 정렬하지 않습니다. 페더레이티드 데이터베이스에 데이터를 리턴하여 로컬로 정렬을 수행합니다. 그러나, 컬럼이 문자 데이터 유형(CHAR 또는 VARCHAR)이고 컬럼에 숫자('0','1',...,'9')만 포함되어 있다고 가정합니다. 'Y' 값을 NUMERIC_STRING 별칭 컬럼 옵션에 지정하여 이를 표시할 수 있습니다. 이와 같이 지정하면 DB2 쿼리 옵티마이저가 데이터 소스에서 정렬을 수행할지 선택할 수 있습니다. 정렬이 리모트로 수행되는 경우 페더레이티드 서버로 데이터를 이동하고 로컬로 정렬을 수행하는 경우의 오버헤드가 발생하지 않습니다.

ALTER NICKNAME문을 사용하여 관계형 별칭에 별칭 컬럼 옵션을 정의할 수 있습니다. CREATE NICKNAME 및 ALTER NICKNAME문을 사용하여 관계형이 아닌 별칭에 별칭 컬럼 옵션을 정의할 수 있습니다.

데이터 유형 맵핑

페더레이티드 서버가 데이터 서버에서 데이터를 검색할 수 있도록 데이터 소스의 데이터 유형을 상응하는 DB2 데이터 유형에 맵핑해야 합니다.

다음과 같은 디폴트 데이터 유형 맵핑 예가 있습니다.

- Oracle 유형 FLOAT는 DB2 유형 DOUBLE로 맵핑됩니다.
- Oracle 유형 DATE는 DB2 유형 TIMESTAMP로 맵핑됩니다.
- z/OS™용 DB2 유형 DATE는 DB2 유형 DATE로 맵핑됩니다.

대부분의 데이터 소스의 경우 디폴트 유형 맵핑은 랩퍼에 있습니다. DB2 데이터 소스의 디폴트 유형 맵핑은 DRDA 랩퍼에 있습니다. Informix의 디폴트 유형 맵핑은 INFORMIX 랩퍼 등에 있습니다.

일부 관계형이 아닌 데이터 소스의 경우, CREATE NICKNAME문에서 데이터 유형 정보를 지정해야 합니다. 별칭 작성 시 각 데이터 소스 오브젝트의 컬럼에 상응하는 DB2 데이터 유형을 지정해야 합니다. 각 컬럼은 특정 필드 또는 데이터 소스 오브젝트의 컬럼에 맵핑되어야 합니다.

관계형 데이터 소스의 경우 디폴트 데이터 유형 맵핑을 겹쳐쓸 수 있습니다. 예를 들면, 디폴트로 Informix INTEGER 데이터 유형을 DB2 INTEGER 데이터 유형으로 맵핑합니다. 디폴트 맵핑을 겹쳐쓰고 Informix의 INTEGER 데이터 유형을 DB2 DECIMAL(10,0) 데이터 유형으로 맵핑할 수 있습니다.

페더레이티드 서버

페더레이티드 시스템에 있는 DB2 서버를 페더레이티드 서버라고 합니다. 개수에 관계없이 DB2 인스턴스를 페더레이티드 서버 기능을 수행하도록 구성할 수 있습니다. 기존 DB2 인스턴스를 페더레이티드 서버로 사용하거나 페더레이티드 시스템으로 사용할 인스턴스를 새로 작성할 수 있습니다.

페더레이티드 시스템을 관리하는 DB2 인스턴스는 일반 사용자 및 클라이언트 응용프로그램의 요청에 응답하기 때문에 서버라고 합니다. 페더레이티드 서버는 종종 수신한 요청 일부를 처리하도록 데이터 소스에 보냅니다. 푸시다운 조작은 리모트로 처리되는 조작입니다. 페더레이티드 시스템을 관리하는 DB2 인스턴스가 요청을 데이터 소스에 푸시다운할 때 클라이언트 역할을 하는 경우에도 이 인스턴스를 페더레이티드 서버라고 합니다.

기타 응용프로그램 서버(AS)와 마찬가지로 페더레이티드 서버는 데이터베이스 관리 프로그램 인스턴스입니다. 응용프로그램은 연결을 처리하고 페더레이티드 서버에 있는 데이터베이스에 요청을 제출합니다. 그러나 다른 응용프로그램 서버(AS)와 구별되는 2개의 기본 기능이 있습니다.

- 페더레이티드 서버는 부분적으로 또는 전체적으로 데이터 소스에 보내기 위해 작성된 요청을 수신하도록 구성됩니다. 페더레이티드 서버는 이러한 요청을 데이터 소스에 분배합니다.
- 기타 응용프로그램 서버(AS)와 마찬가지로 페더레이티드 서버는 DRDA 통신 프로토콜(TCP/IP 사용)을 사용하여 DB2 제품군 인스턴스와 통신합니다. 그러나 다른 응용프로그램 서버(AS)와는 달리 페더레이티드 서버는 데이터 소스의 원시(native) 클라이언트를 사용하여 데이터 소스에 액세스합니다. 예를 들어, 페더레이티드 서버는 Sybase Open Client를 사용하여 Sybase 데이터 소스에 액세스하고 Microsoft SQL Server ODBC 드라이버를 사용하여 Microsoft SQL Server 데이터 소스에 액세스합니다.

지원되는 데이터 소스

페더레이티드 시스템을 사용하여 여러 데이터 소스에 액세스할 수 있습니다.

IBM InfoSphere Federation Server에서는 다음 표에 표시된 데이터 소스를 지원합니다. 첫 번째 표에는 데이터 클라이언트 소프트웨어 관련 요구사항이 나열되어 있습니다. 달리 지정되지 않은 한 클라이언트 소프트웨어는 별도로 구해야 합니다.

액세스하려는 데이터 소스에서 사용할 클라이언트 소프트웨어를 설치해야 합니다. 클라이언트 소프트웨어는 IBM InfoSphere Federation Server와 동일한 시스템에 설치되어야 합니다. DB2 제어 센터와 같은 몇몇 도구를 사용하고, 스토어드 프로시저 및 사용자 정의 함수(UDF)를 포함한 Java 응용프로그램을 작성하고 실행하려면 적합한 Java SDK도 필요합니다.

대부분의 최신 정보는 웹에 있는 데이터 소스 요구사항 페이지를 참조하십시오.

표 5. 지원되는 데이터 소스, 클라이언트 소프트웨어 요구사항 및 32비트 운영 체제의 지원

			32비트 하드웨어 아키텍처 및 운영 체제	
			X86-32	**X86-32**
데이터 소스	**지원되는 버전**	**클라이언트 소프트웨어**	**Linux, RedHat Enterprise Linux(RHEL), SUSE**	**Windows**
BioRS	5.2, 5.3	없음	Y	Y
Linux, UNIX 및 Windows용 DB2	8.1.x, 8.2.x, 9.1, 9.5, 9.7	없음	Y	Y
z/OS용 DB2	7.x, 8.x, 9.x	DB2® Connect™ V9.7	Y	Y
System i용 DB2	5.2, 5.3, 5.4, 6.1	DB2 Connect V9.7	Y	Y
VSE 및 VM용 DB2 서버	7.2, 7.4	DB2 Connect V9.7	Y	Y
플랫 파일		없음	Y	Y
Informix	Informix XPS 8.50, 8.51 및 Informix IDS IDS 7.31, IDS 9.40, IDS 10.0, 11.5, 11.10	Informix Client SDK 버전 2.81.TC2 이상 (Power의 SLES 10에는 3.0이 필요함)	Y	Y
JDBC	3.0 이상	JDBC 3.0 이상을 준수하는 JDBC 드라이버	Y	Y
Microsoft Excel	2000, 2002, 2003, 2007	없음		Y

표 5. 지원되는 데이터 소스, 클라이언트 소프트웨어 요구사항 및 32비트 운영 체제의 지원 (계속)

			32비트 하드웨어 아키텍처 및 운영 체제	
			X86-32	**X86-32**
데이터 소스	**지원되는 버전**	**클라이언트 소프트웨어**	**Linux, RedHat Enterprise Linux(RHEL), SUSE**	**Windows**
Microsoft SQL Server	Microsoft SQL Server 2000/SP4, 2005, 2008	Windows의 경우, Microsoft SQL Server Client ODBC 3.0 이상의 드라이버. Unix의 경우, DataDirect ODBC 5.3	Y	Y
MQ	MQ7	MQ7	Y	Y
ODBC	3.0	ODBC 3.0 이상**을 준수하는 ODBC 드라이버	Y	Y
OLE DB	2.7, 2.8	OLE DB 2.0 이상	Y	Y
Oracle	10g, 10gR2, 11g, 11gR1	Oracle NET 클라이언트 10.0 - 10.1, 10.2.0.1(3807408 패치 포함), 10.1.0.3(3807408 패치 포함), 11.1.0.6.0	Y	Y
Sybase	Sybase ASE 12.5, 15.0	Sybase Open Client 12.5 - 15.0	Y	Y

표 5. 지원되는 데이터 소스, 클라이언트 소프트웨어 요구사항 및 32비트 운영 체제의 지원 (계속)

			32비트 하드웨어 아키텍처 및 운영 체제	
			X86-32	**X86-32**
데이터 소스	**지원되는 버전**	**클라이언트 소프트웨어**	**Linux, RedHat Enterprise Linux(RHEL), SUSE**	**Windows**
Teradata	2.5, 2.6, 12	Shared Common Components for Internationalization for Teradata(tdicu) 버전 1.01 이상, Teradata Generic Security Services(TeraGSS) 버전 6.01 이상 및 다음 운영 체제의 소프트웨어 Windows의 경우 Teradata Client TTU 8.0 이상 및 Teradata API Library CLIv2 4.8.0 이상 UNIX 및 Linux의 경우 Teradata Call-Level Interface 버전 2 CLIv2 릴리스 4.8.0 이상	Y	Y
웹 서비스	WSDL 1.0, 1.1 SOAP 1.0, 1.1	없음	Y	Y
XML	XML1.0, XML1.1	없음	Y	Y

** 기타 데이터 소스 중에서 RedBrick 6.20cu5 및 InfoSphere Classic Federation V8.2 이상에 액세스하는 데 ODBC를 사용할 수 있습니다.

표 6. 64비트 운영 체제의 지원

64비트 하드웨어 아키텍처	**X86-64**	**X86-64**	**Power**	**Itanium®**	**Power**	**Sparc**	**zSeries®**
운영 체제	**Linux RHEL SUSE**	**Windows**	**AIX®**	**HP-UX**	**Linux RHEL SUSE**	**Solaris**	**Linux RHEL SUSE**
데이터 소스							
BioRS	Y	Y	Y	Y	Y	Y	Y

표 6. 64비트 운영 체제의 지원 (계속)

64비트 하드웨어 아키텍처	**X86-64**	**X86-64**	**Power**	**Itanium®**	**Power**	**Sparc**	**zSeries®**
운영 체제	**Linux RHEL SUSE**	**Windows**	**AIX®**	**HP-UX**	**Linux RHEL SUSE**	**Solaris**	**Linux RHEL SUSE**
데이터 소스							
Linux, UNIX 및 Windows용 DB2	Y	Y	Y	Y	Y	Y	Y
z/OS용 DB2	Y	Y	Y	Y	Y	Y	Y
System i용 DB2	Y	Y	Y	Y	Y	Y	Y
VSE 및 VM용 DB2 서버	Y	Y	Y	Y	Y	Y	Y
Informix	Y		Y	Y	Y	Y	Y
JDBC	Y	Y	Y	Y	Y	Y	Y
Microsoft Excel							
Microsoft SQL Server	Y	Y	Y	Y		Y	
MQ	Y	N	Y	Y	Y	Y	Y
ODBC	Y	Y	Y***	Y		Y***	Y
OLE DB		Y		Y			
Oracle	Y	Y	Y	Y	Y	Y	Y
스크립트	Y	Y	Y	Y	Y	Y	Y
Sybase	Y		Y	Y	Y	Y	
Teradata	Y		Y	Y		Y	
웹 서비스	Y	Y	Y	Y	Y	Y	Y
XML	Y	Y	Y	Y	Y	Y	Y

*** 32비트 및 64비트 클라이언트를 사용하여 RedBrick 6.20cu5 및 z/OS용 IBM InfoSphere Classic Federation Server에 액세스하는 데 ODBC를 사용할 수 있습니다.

페더레이티드 데이터베이스 시스템 카탈로그

페더레이티드 데이터베이스 시스템 카탈로그에는 페더레이티드 데이터베이스의 오브젝트에 대한 정보 및 데이터 소스의 오브젝트에 대한 정보가 포함되어 있습니다.

페더레이티드 데이터베이스의 카탈로그에는 전체 페더레이티드 시스템에 대한 정보가 들어 있기 때문에 전역 카탈로그라고 합니다. DB2 쿼리 옵티마이저는 전역 카탈로그 및 데이터 소스 랩퍼에 있는 정보를 사용하여 SQL문을 처리하는 데 가장 좋은 방법을 계획합니다. 전역 카탈로그에 저장된 정보에는 컬럼 이름, 컬럼 데이터 유형, 컬럼 디폴트 값, 인덱스 정보 및 통계 정보와 같은 리모트 및 로컬 정보가 포함됩니다.

리모트 카탈로그 정보는 데이터 소스에서 사용되는 정보 또는 이름입니다. 로컬 카탈로그 정보는 페더레이티드 데이터베이스에서 사용되는 정보 또는 이름입니다. 예를 들어, 리모트 테이블에 이름이 *EMPNO*인 컬럼이 포함되어 있다고 가정합니다. 전역 카탈로그는 리모트 컬럼 이름을 *EMPNO*로 저장합니다. 사용자가 다른 이름을 지정하지 않는 한 로컬 컬럼 이름은 *EMPNO*로 저장됩니다. 로컬 컬럼 이름을 *Employee_Number*로 변경할 수 있습니다. 이 컬럼이 포함된 쿼리를 제출하는 사용자는 해당 쿼리에서 *EMPNO*가 아니라 *Employee_Number*를 사용합니다. ALTER NICKNAME문을 사용하여 데이터 소스 컬럼의 로컬 이름을 변경합니다.

관계형 및 관계형이 아닌 데이터 소스의 경우, 전역 카탈로그에 저장된 정보에는 리모트 및 로컬 정보가 둘 다 포함됩니다.

전역 카탈로그에 저장되어 있는 데이터 소스 테이블 정보를 확인하려면 페더레이티드 데이터베이스에서 SYSCAT.TABLES, SYSCAT.NICKNAMES, SYSCAT.TABOPTIONS, SYSCAT.INDEXES, SYSCAT.INDEXOPTIONS, SYSCAT.COLUMNS 및 SYSCAT.COLOPTIONS 카탈로그 뷰를 쿼리하십시오.

전역 카탈로그에는 데이터 소스에 대한 기타 정보도 포함되어 있습니다. 예를 들어, 전역 카탈로그에는 페더레이티드 서버가 데이터 소스에 연결하고 페더레이티드 사용자 권한을 데이터 소스 사용자 권한에 맵핑하기 위해 사용하는 정보도 포함되어 있습니다. 전역 카탈로그에는 서버 옵션과 같이 사용자가 명시적으로 설정한 데이터 소스에 대한 속성도 들어 있습니다.

쿼리 옵티마이저

SQL 컴파일러 프로세스의 파트로서 쿼리 옵티마이저가 쿼리를 분석합니다. 컴파일러는 쿼리를 처리하기 위해 액세스 플랜이라는 대체 전략을 개발합니다.

액세스 플랜은 쿼리를 다음과 같이 처리하도록 요청합니다.

- 데이터 소스를 통해 처리
- 페더레이티드 서버를 통해 처리
- 일부는 데이터 소스를 통해, 일부는 페더레이티드 서버를 통해 처리

쿼리 옵티마이저는 데이터 소스 성능 및 데이터에 대한 정보를 기초로 액세스 플랜을 미리 평가합니다. 랩퍼와 전역 카탈로그에 이 정보가 들어 있습니다. 쿼리 옵티마이저는 쿼리를 쿼리 조각이라는 세그먼트로 분할합니다. 일반적으로 데이터 소스가 조각을 처리할 수 있는 경우에는 쿼리 조각을 데이터 소스에 푸시다운하는 것이 더 효율적입니다. 그러나 쿼리 옵티마이저는 다음과 같은 기타 요소를 고려합니다.

- 처리해야 하는 데이터의 양
- 데이터 소스의 처리 속도
- 조각이 리턴하는 데이터 양

- 통신 대역폭
- 페더레이티드 서버에 동일한 쿼리 결과를 나타내는 사용 가능한 구체화된 쿼리 테이블이 있는지 여부

쿼리 옵티마이저는 쿼리 조각을 처리하는 데 사용할 대체 액세스 플랜을 생성합니다. 대체 플랜은 페더레이티드 서버 및 리모트 데이터 소스에서 로컬로 다양한 양의 작업을 수행합니다. 쿼리 옵티마이저는 비용을 기본으로 하기 때문에 자원 사용 비용을 대체 액세스 플랜에 지정합니다. 그런 다음 쿼리 옵티마이저는 가장 적은 자원 사용 비용으로 쿼리를 처리할 플랜을 선택합니다.

데이터 소스를 통해 처리할 조각이 있는 경우 페더레이티드 데이터베이스는 해당 조각을 데이터 소스에 제출합니다. 데이터 소스가 조각을 처리한 후 그 결과를 검색하여 페더레이티드 데이터베이스에 리턴합니다. 페더레이티드 데이터베이스가 일부 처리를 수행한 경우 해당 결과를 데이터 소스에서 검색한 결과와 조합합니다. 그런 다음 페더레이티드 데이터베이스가 모든 결과를 클라이언트에 리턴합니다.

조합 시퀀스

데이터베이스에서 문자 데이터를 정렬하는 순서는 데이터베이스에 정의된 데이터 구조 및 조합 시퀀스에 따라 다릅니다.

데이터베이스의 데이터가 모두 대문자로 되어 있고 숫자 또는 특수 문자가 포함되어 있지 않다고 가정합니다. 데이터를 데이터 소스에서 정렬했는지 또는 페더레이티드 데이터베이스에서 정렬했는지 여부에 관계 없이 데이터 정렬 결과는 동일합니다. 각 데이터베이스에서 사용되는 조합 시퀀스는 정렬 결과에 영향을 주지 않습니다. 마찬가지로, 데이터베이스의 데이터가 모두 소문자 또는 숫자로 되어 있는 경우 데이터를 정렬하면 실제로 정렬이 수행되는 위치에 관계 없이 동일한 결과가 생성됩니다.

데이터가 다음과 같은 구조 중 하나로 구성된 경우:

- 문자 및 숫자의 조합
- 대문자 및 소문자 모두 사용
- 특수 문자(예: @, #, €)

이 데이터를 정렬하면 페더레이티드 데이터베이스와 데이터 소스에서 서로 다른 조합 시퀀스를 사용하는 경우 결과가 다르게 나타납니다.

일반적으로 설명하자면, 조합 시퀀스는 문자 데이터에 정의된 순서로서 특정 문자를 다른 문자보다 상위로, 하위로 또는 동일하게 정렬할지 여부를 판별합니다.

조합 시퀀스로 정렬 순서를 판별하는 방법

조합 시퀀스는 코드화된 문자 세트에서 문자의 정렬 순서를 판별합니다.

문자 세트는 컴퓨터 시스템 또는 프로그래밍 언어에서 사용되는 문자의 집합입니다. 코드화된 문자 세트에서 각 문자는 0 - 255 또는 이에 상응하는 16진수 범위에 있는 서로 다른 숫자에 지정됩니다. 이 숫자를 코드 포인트라고 하며 세트의 문자에 숫자를 지정한 것을 집합적으로 코드 페이지라고 합니다.

문자에 지정하는 것 외에도 코드 포인트를 정렬 순서의 문자 위치에 맵핑할 수 있습니다. 기술적인 용어로 설명하자면 조합 시퀀스는 세트의 문자 정렬 순서 위치에 문자 세트의 코드 포인트를 집합적으로 맵핑한 것입니다. 문자의 위치는 숫자로 표시되며 이 숫자를 문자의 가중치라고 합니다. ID 시퀀스라고 하는 가장 간단한 조합 시퀀스에서 가중치는 코드 포인트와 동일합니다.

예: ALPHA 데이터베이스는 EBCDIC 코드 페이지의 디폴트 조합 시퀀스를 사용합니다. BETA 데이터베이스는 ASCII 코드 페이지의 디폴트 조합 시퀀스를 사용합니다. 이들 두 데이터베이스에서 문자열의 정렬 순서는 서로 다릅니다.

```
SELECT.....
  ORDER BY COL2

EBCDIC-Based Sort           ASCII-Based Sort
```

```
COL2                    COL2
----                    ----
V1G                     7AB
Y2W                     V1G
7AB                     Y2W
```

예: 위 예와 비슷하게 데이터베이스의 문자 비교는 해당 데이터베이스에 정의된 조합 시퀀스에 따라 다릅니다. ALPHA 데이터베이스는 EBCDIC 코드 페이지의 디폴트 조합 시퀀스를 사용합니다. BETA 데이터베이스는 ASCII 코드 페이지의 디폴트 조합 시퀀스를 사용합니다. 이들 두 데이터베이스에서의 문자 비교 결과는 다르게 나타납니다.

```
SELECT.....
  WHERE COL2 > 'TT3'

EBCDIC-Based Results     ASCII-Based Results

COL2                     COL2
----                     ----
TW4                      TW4
X82                      X82
39G
```

쿼리 최적화를 위한 로컬 조합 시퀀스 설정

관리자는 데이터 소스 조합 시퀀스와 일치하는 특정 조합 시퀀스로 페더레이티드 데이터베이스를 작성할 수 있습니다.

그러면 각 데이터 소스 서버 정의에서 COLLATING_SEQUENCE 서버 옵션이 'Y'로 설정됩니다. 이와 같이 설정하면 페더레이티드 데이터베이스에서 페더레이티드 데이터베이스와 데이터 소스의 조합 시퀀스가 일치하는 것을 확인할 수 있습니다.

CREATE DATABASE 명령의 파트로 페더레이티드 데이터베이스 조합 시퀀스를 설정합니다. 이 명령을 통해 다음 시퀀스 중 하나를 지정할 수 있습니다.

- ID 시퀀스
- 시스템 시퀀스(데이터베이스를 지원하는 운영 체제에서 사용되는 시퀀스)
- 사용자 정의 시퀀스(DB2 데이터베이스 시스템이 제공하거나 사용자가 직접 정의한 사전 정의된 시퀀스)

데이터 소스가 z/OS용 DB2라고 가정합니다. ORDER BY 절에서 정의된 정렬이 EBCDIC 코드 페이지를 기본으로 하는 조합 시퀀스를 통해 구현됩니다. ORDER BY 절에 따라 정렬된 z/OS용 DB2 데이터를 검색하려면 페더레이티드 데이터베이스가 적합한 EBCDIC 코드 페이지를 기본으로 하는 사전 정의된 조합 시퀀스를 사용할 수 있도록 페더레이티드 데이터베이스를 구성하십시오.

제 2 장 언어 요소

문자

SQL 언어 키워드 및 연산자의 기본 기호는 모든 IBM 문자 세트의 일부인 1바이트 문자입니다. 언어의 문자는 문자, 숫자 또는 특수 문자로 분류됩니다.

*문자*가 26개의 대문자(A - Z) 또는 26개의 소문자(a - z) 중 하나입니다.문자에는 자국어의 영문자 Extender로 예약된 세 개의 코드 포인트(미국의 경우 #, @ 및 $)도 포함됩니다. 그러나 이 세 개의 코드 포인트는 CCSID에 따라 다른 문자를 나타내므로 특히 휴대용 응용프로그램의 경우 사용하지 마십시오. 또한 문자는 확장 문자 세트의 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.

*숫자*는 0 - 9까지의 문자입니다.

*특수 문자*는 다음과 같습니다.

문자	설명	문자	설명
	스페이스 또는 공백	-	빼기 부호
″	작은따옴표 또는 큰따옴표	.	마침표
%	퍼센트	/	슬래시
&	앰퍼샌드	:	콜론
’	어포스트로피 또는 작은따옴표	;	세미콜론
(	왼쪽 괄호	<	보다 작음
)	오른쪽 괄호	=	같음
*	별표	>	보다 큼
+	더하기 부호	?	물음표
,	쉼표	_	밑줄
|	수직바[1]	^	캐럿
!	느낌표	[	왼쪽 대괄호
{	왼쪽 중괄호	]	오른쪽 대괄호
}	오른쪽 중괄호	₩	역방향 사선 또는 백슬래시[2]

[1] 수직 바(|) 문자를 사용하면 IBM 관련 제품 간의 코드 이식성을 방지할 수 있습니다. || 연산자 대신 CONCAT 연산자를 사용하십시오.

[2] 일부 코드 페이지에는 역방향 사선(₩) 문자의 코드 포인트가 없습니다. 유니코드 문자열 상수를 입력한 경우, UESCAPE 절은 역방향 사선이 아닌 유니코드 Escape 문자를 지정하는 데 사용될 수 있습니다.

모든 복수 바이트 문자는 특수 문자인 2바이트 공백을 제외하고는 문자로 취급됩니다.

토큰

토큰은 SQL의 기본적인 구문 단위입니다. 토큰은 하나 이상의 문자 시퀀스입니다. 토큰은 공백을 포함할 수 있는 문자열 상수나 분리 ID가 아닌 경우 공백 문자를 포함할 수 없습니다.

토큰은 일반 또는 분리문자로 분류됩니다.

- *일반* 토큰은 숫자 상수, 일반 ID, 호스트 ID 또는 키워드입니다.

  예

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

- *분리문자* 토큰은 문자열 상수, 분리 ID, 연산자 기호 또는 구문 다이어그램에 표시된 특수 문자 중 하나입니다. 물음표도 매개변수 표시문자 역할을 할 때 분리문자 토큰입니다.

  예

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

스페이스: 스페이스는 하나 이상의 공백 문자입니다. 문자열 상수 및 분리 ID가 아닌 다른 토큰은 스페이스를 포함할 수 없습니다. 토큰 다음에 공백이 올 수 있습니다. 구문에서 허용되는 경우 모든 일반 토큰 다음에는 스페이스나 분리문자 토큰이 있어야 합니다.

주석: SQL 주석은 일괄로 묶이거나(/*로 시작하고 */로 끝남) 단순(두 개의 연속 하이픈으로 시작하여 행 끝에서 끝남) 주석입니다. 정적 SQL문은 호스트 언어 주석이나 SQL 주석을 포함할 수 있습니다. 주석은 분리문자 토큰 내 또는 EXEC 및 SQL 사이를 제외하고, 스페이스가 지정될 수 있는 어디에서나 지정할 수 있습니다.

대소문자 구분: 모든 토큰은 소문자를 포함할 수 있지만, 일반 토큰의 소문자는 대문자로 변환됩니다. 대소문자 구분 ID가 있는 C 언어로 된 호스트 변수는 제외입니다. 분리문자 토큰은 대문자로 변환되지 않습니다. 다음 명령문은

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

대문자 변환 후 다음과 같습니다.

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

멀티바이트 영문자는 대문자로 변환되지 않습니다. 1바이트 문자(a - z)는 대문자로 변환됩니다.

유니코드에 있는 문자의 경우:

- 적용 가능한 경우, UTF-8의 대문자가 UTF-8의 소문자와 길이가 같으면 문자가 대문자로 변환됩니다. 예를 들어, 점이 없는 터키어 소문자 'i'는 대문자로 변환되지 않습니다. UTF-8에서, 이 문자의 값은 X'C4B1'이지만 점 없는 대문자 'I'의 값은 X'49'이기 때문입니다.
- 대문자 변환은 로케일에 민감하지 않은 방식으로 수행됩니다. 예를 들어, 점 있는 터키어 소문자 'i'는 영어 대문자(점 없는) 'I'로 변환됩니다.
- 반자 및 전자 영문자 모두 대문자로 변환됩니다. 예를 들어, 전자 소문자 'a'(U+FF41)는 전자 대문자 'A'(U+FF21)로 변환됩니다.

ID

*ID*는 이름을 구성하는 데 사용되는 토큰입니다. SQL문의 ID는 SQL ID 또는 호스트 ID입니다.

- SQL ID

  *SQL ID*에는 일반 ID 및 분리 ID의 두 가지 유형이 있습니다.

  – *일반 ID*는 다음에 0개 이상의 문자가 오는 대문자이며 각각 대문자, 숫자 또는 밑줄 문자가 올 수 있습니다. 일반 ID가 대문자로 변환되었다는 것에 유의하십시오. 일반 ID는 예약어이어서는 안됩니다.

  예

  ```
  WKLYSAL     WKLY_SAL
  ```

  – *분리 ID*는 큰따옴표로 묶인 하나 이상의 문자 시퀀스입니다. 두 개의 연속적인 큰따옴표는 분리 ID 내에서 하나의 따옴표를 나타내는 데 사용됩니다. 이러한 방식으로 ID에 소문자가 포함될 수 있습니다.

  예

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

  2바이트 코드 페이지에서 작성되었지만 복수 바이트 코드 페이지의 응용프로그램 또는 데이터베이스에서 사용되는 ID의 문자 변환에는 특별히 고려해야 할 사항이 있습니다. 변환 후 이러한 ID는 ID 길이 제한을 초과할 수 있습니다.

- 호스트 ID

  호스트 *ID*는 호스트 프로그램에서 선언된 이름입니다. 호스트 ID를 구성하는 규칙은 호스트 언어의 규칙입니다. 호스트 ID의 길이는 255바이트를 초과할 수 없으며 SQL 또는 DB2(대문자 또는 소문자)로 시작할 수 없습니다.

이름 지정 규칙 및 내재된 오브젝트 이름 규정

데이터베이스 오브젝트 이름을 이름별 지정된 오브젝트의 유형에 따라 양식화하기 위한 규칙입니다. 이름은 단일 SQL ID로 구성되거나 오브젝트를 더욱 명확히 식별하는 하나 이상의 ID로 규정될 수 있습니다. 기간은 각 ID로 분리해야 합니다.

다음 오브젝트 이름은 SQL 프로시저의 컨텍스트에서 사용될 때 이름이 구분되는 경우에도 일반 ID에 허용되는 문자만 사용할 수 있습니다.

- condition-name
- label
- parameter-name
- procedure-name

- SQL-variable-name
- statement-name

구문 도표는 다른 유형의 이름에 대해 다른 용어를 사용합니다. 다음 목록에는 이러한 용어가 정의되어 있습니다.

alias-name
별명을 지정하는 스키마 규정 이름

attribute-name
구조화된 데이터 유형의 속성을 지정하는 ID

authorization-name
사용자, 그룹 또는 역할을 지정하는 ID. 사용자 또는 그룹:

- 유효한 문자는 다음과 같습니다. 'A'부터 'Z'까지, 'a'부터 'z'까지, '0'부터 '9'까지, '#', '@', '$', '_', '!'; ' '('; ')'; '{'; '}'; '-'; '.' 및 '^'입니다.
- 명령행 처리기를 통해 입력할 때 인용 부호와 분리되어야 하는 문자는 다음과 같습니다. '!'; ' '('; ')'; '{'; '}'; '-'; '.' 및 '^'입니다.
- 이름은 문자 'SYS', 'IBM' 또는 'SQL'로 시작하면 안됩니다.
- 다음과 같은 이름은 사용할 수 없습니다. 'ADMINS', 'GUESTS', 'LOCAL', 'PUBLIC' 또는 'USERS'입니다.
- 분리된 권한 부여 ID에는 소문자가 포함될 수 없습니다.

bufferpool-name
버퍼 풀을 지정하는 ID.

column-name
테이블이나 뷰의 컬럼을 지정하는 규정된 이름이거나 규정되지 않은 이름. 규정자는 테이블 이름, 뷰 이름, 별칭 또는 상관 이름일 수 있습니다.

component-name
보안 레이블 구성요소를 지정하는 ID입니다.

condition-name
조건을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 조건 이름은 컨텍스트에 따라 내재적으로 규정됩니다. 조건이 모듈에서 정의되고 동일한 모듈 외부에서 사용되는 경우 모듈 이름으로 규정되어야 합니다.

constraint-name
참조 제한조건, 기본 키 제한조건, 고유 제한조건 또는 테이블 점검 제한조건을 지정하는 ID

correlation-name
결과 테이블을 지정하는 ID

cursor-name

SQL 커서를 지정하는 ID. 호스트 호환성을 위해 이름에 하이픈을 사용할 수도 있습니다.

cursor-type-name

사용자 정의 커서 유형을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 커서 유형 이름은 컨텍스트에 따라 내재적으로 규정됩니다.

cursor-variable-name

커서 유형의 전역 변수, 로컬 변수 또는 SQL 매개변수를 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 커서 변수 이름은 컨텍스트에 따라 내재적으로 규정됩니다.

data-source-name

데이터 소스를 지정하는 ID. 이 ID는 세 부분으로 구성된 리모트 오브젝트 이름 중 첫 번째 부분입니다.

db-partition-group-name

데이터베이스 파티션 그룹을 지정하는 ID

descriptor-name

콜론 다음에 SQL 디스크립터 영역(SQLDA)을 지정하는 호스트 ID가 옵니다. 호스트 ID에 대한 설명은 82 페이지의 『호스트 변수에 대한 참조』의 내용을 참조하십시오. 디스크립터 이름에는 표시기 변수가 포함될 수 없다는 점에 유의하십시오.

distinct-type-name

구별 유형을 지정하는 규정된 이름이거나 규정되지 않은 이름. 구별 유형 이름의 규정되지 않은 형식은 SQL ID입니다. SQL문의 규정되지 않은 구별 유형 이름은 내재적으로 규정됩니다. 내재된 규정자는 스키마 이름 또는 모듈 이름이며 이는 구별 유형 이름이 표시되는 컨텍스트로 판별됩니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름이거나 마침표와 SQL ID가 뒤에 오는 모듈 이름(스키마 이름으로도 규정되는 이름)입니다. 구별 유형이 모듈에서 정의되고 같은 모듈 외부에서 사용되는 경우 모듈 이름으로 규정되어야 합니다.

event-monitor-name

이벤트 모니터를 지정하는 ID

function-mapping-name

함수 맵핑을 지정하는 ID

function-name

함수를 지정하는 규정된 이름이거나 규정되지 않은 이름. 함수 이름의 규정되지 않은 형식은 SQL ID입니다. SQL문의 규정되지 않은 함수 이름은 내재적으로 규정됩니다. 내재된 규정자는 스키마 이름이며 이는 함수가 표시하는 컨텍스트

로 판별됩니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름이거나, 마침표와 SQL ID가 뒤에 오는 모듈 이름입니다. 함수는 모듈에서 발행되고 같은 모듈 외부에서 사용되는 경우 모듈 이름으로 규정되어야 합니다.

global-variable-name
전역 변수를 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 전역 변수 이름은 컨텍스트에 따라 내재적으로 규정됩니다. 전역 변수가 모듈에서 정의되고 같은 모듈 외부에서 사용되는 경우 모듈 이름으로 규정되어야 합니다.

group-name
구조화된 유형에 대해 정의된 변환 그룹을 지정하는 규정되지 않은 ID

host-variable
호스트 변수를 지정하는 토큰 시퀀스. 82 페이지의『호스트 변수에 대한 참조』에 설명된 대로 호스트 변수에는 최소한 하나의 호스트 ID가 포함됩니다.

index-name
인덱스 또는 인덱스 스펙을 지정하는 스키마 규정 이름

label SQL 프로시저의 레이블을 지정하는 ID

method-name
메소드를 지정하는 ID. 메소드에 대한 스키마 컨텍스트는 메소드의 주제 유형(또는 주제 유형의 슈퍼 유형)의 스키마에 따라 결정됩니다.

module-name
모듈을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 모듈 이름은 내재적으로 규정됩니다. 내재된 규정자는 스키마 이름이며, 이는 모듈 이름이 나타나는 컨텍스트로 판별됩니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름입니다.

별칭 테이블이나 뷰에 대한 페더레이티드 서버 참조를 지정하는 스키마 규정 이름

package-name
패키지를 지정하는 스키마 규정 이름. 패키지에 포함된 버전 ID가 빈 문자열이 아닌 경우, 패키지 이름 끝에 `schema-id.package-id.version-id`와 같은 형식의 버전 ID가 포함되기도 합니다.

parameter-name
프로시저, 사용자 정의 함수, 메소드 또는 인덱스 확장에서 참조될 수 있는 매개변수를 지정하는 ID

partition-name
파티션된 테이블의 데이터 파티션을 지정하는 ID입니다.

procedure-name
프로시저를 지정하는 규정된 이름이거나 규정되지 않은 이름. 프로시저 이름의

규정되지 않은 형식은 SQL ID입니다. SQL문에서 규정되지 않은 프로시저 이름은 내재적으로 규정됩니다. 내재된 규정자는 스키마 이름이며, 이는 프로시저가 나타나는 컨텍스트로 판별됩니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름이거나, 마침표와 SQL ID가 뒤에 오는 모듈 이름입니다. 프로시저가 모듈에서 정의되고 같은 모듈 외부에서 사용되는 경우 모듈 이름으로 규정되어야 합니다.

remote-authorization-name

데이터 소스 사용자를 지정하는 ID. 권한 부여 이름에 대한 규칙은 데이터 소스에 따라 다릅니다.

remote-function-name

데이터 소스 데이터베이스에 등록된 함수를 지정하는 이름

remote-object-name

데이터 소스 테이블이나 뷰를 지정하고 테이블이나 뷰가 있는 데이터 소스를 식별하는 세 부분으로 구성된 이름. 이 이름은 data-source-name, remote-schema-name 및 remote-table-name의 세 부분으로 구성됩니다.

remote-schema-name

데이터 소스 테이블이나 뷰가 속해 있는 스키마를 지정하는 이름. 이 이름은 세 부분으로 구성된 리모트 오브젝트 이름 중 두 번째 부분입니다.

remote-table-name

데이터 소스에서 테이블이나 뷰를 지정하는 이름. 이 이름은 세 부분으로 구성된 리모트 오브젝트 이름 중 세 번째 부분입니다.

remote-type-name

데이터 소스 데이터베이스에서 지원하는 데이터 유형. 내장 유형에는 long 형식을 사용하지 마십시오. 예를 들어, CHARACTER 대신에 CHAR을 사용하십시오.

role-name

역할을 지정하는 ID.

row-type-name

행 유형을 지정하는 규정된 이름 또는 규정되지 않은 이름. row-type-name의 규정되지 않은 형식은 SQL ID입니다. SQL문의 규정되지 않은 row-type-name은 내재적으로 규정됩니다. 내재된 규정자는 스키마 이름으로, 스키마 이름은 “규정되지 않은 사용자 정의 유형, 함수, 프로시저 및 특정 이름”의 규칙에서 설명한 대로 row-type-name이 나타나는 컨텍스트에서 판별됩니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름입니다.

savepoint-name

세이브포인트를 지정하는 ID

schema-name

SQL 오브젝트에 대한 논리 그룹화를 제공하는 ID. 오브젝트 이름의 규정자로 사용되는 스키마 이름은 내재적으로 다음과 같이 판별될 수 있습니다.

- CURRENT SCHEMA 특수 레지스터의 값으로부터
- QUALIFIER 프리컴파일/바인드 옵션 값으로부터
- CURRNT PATH 특수 레지스터를 사용하는 분석 알고리즘에 따라
- 동일한 SQL문에 있는 다른 오브젝트의 스키마 이름에 따라

복잡하지 않게 하려면 스키마 이름 SESSION을 *반드시* 사용해야 하는 선언된 전역 임시 테이블의 스키마로 사용하는 것을 제외하고는 스키마로 SESSION을 사용하지 않는 것이 바람직합니다.

security-label-name

보안 레이블을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 보안 레이블 이름은 적용 가능한 보안 규정 이름에 의해 내재적으로 규정됩니다. 내재적으로 적용 가능한 보안 규정 이름이 없는 경우 이름이 규정되어야 합니다.

security-policy-name

보안 규정을 지정하는 ID입니다.

sequence-name

시퀀스를 지정하는 ID.

server-name

응용프로그램 서버를 지정하는 ID. 페더레이티드 시스템에서는 서버 이름이 데이터 소스의 로컬 이름을 지정하기도 합니다.

specific-name

특정 이름을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 특정 이름은 컨텍스트에 따라 내재적으로 규정됩니다.

SQL-variable-name

SQL 프로시저 명령문의 로컬 변수 이름. SQL 변수 이름은 호스트 변수 이름이 허용되는 다른 SQL문에서 사용할 수 있습니다. 이름은 SQL 변수를 선언한 복합 명령문의 레이블로 규정될 수 있습니다.

statement-name

Prepared SQL문을 지정하는 ID

supertype-name

슈퍼 유형을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 상위 유형 이름은 컨텍스트에 따라 내재적으로 규정됩니다.

table-name

테이블을 지정하는 스키마 규정 이름

table-reference

테이블을 지정하는 규정된 이름이거나 규정되지 않은 이름. 공통 테이블 표현식에서 규정되지 않은 테이블 참조는 디폴트 스키마에 의해 내재적으로 규정됩니다.

tablespace-name

테이블 스페이스를 지정하는 ID

trigger-name

트리거를 지정하는 스키마 규정 이름

type-mapping-name

데이터 유형 맵핑을 지정하는 ID

type-name

유형을 지정하는 규정된 이름이거나 규정되지 않은 이름. SQL문의 규정되지 않은 유형 이름은 컨텍스트에 따라 내재적으로 규정됩니다.

typed-table-name

유형이 지정된 테이블을 지정하는 스키마 규정 이름

typed-view-name

유형이 지정된 뷰를 지정하는 스키마 규정 이름

user-defined-type-name

사용자 정의 데이터 유형을 지정하는 규정된 이름이거나 규정되지 않은 이름. 사용자 정의 유형 이름의 규정되지 않은 형식은 SQL ID입니다. SQL문의 규정되지 않은 사용자 정의 유형 이름은 내재적으로 규정됩니다. 내재된 규정자는 스키마 이름 또는 모듈 이름이며, 이는 사용자 정의 유형 이름이 표시되는 컨텍스트로 판별됩니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름이거나 마침표와 SQL ID가 뒤에 오는 모듈 이름(스키마 이름으로도 규정되는 이름)입니다. 사용자 정의 데이터 유형이 모듈에서 정의되고 같은 모듈 외부에서 사용되는 경우 모듈 이름으로 규정되어야 합니다.

view-name

뷰를 지정하는 스키마 규정 이름

wrapper-name

랩퍼를 지정하는 ID

XML-스키마-이름

XML 스키마를 지정하는 규정된 이름이거나 규정되지 않은 이름입니다.

xsrobject-name

XML 스키마 저장소의 오브젝트를 지정하는 규정된 이름이거나 규정되지 않은 이름입니다.

데이터베이스 오브젝트의 별명

별명은 SQL 오브젝트에 대한 대체 이름으로 간주될 수 있습니다. 따라서 SQL 오브젝트는 이름이나 테이블 별명으로 SQL문에서 참조할 수 있습니다.

공용 별명은 스키마 이름으로 이름을 규정하지 않고 항상 참조할 수 있는 별명입니다. 공용 별명의 내재된 규정자는 SYSPUBLIC이며, 명시적으로 규정될 수도 있습니다.

별명은 동의어라고도 합니다.

기본이 되는 오브젝트가 사용될 수 있는 곳마다 별명이 사용될 수 있습니다. 오브젝트가 없는 경우에도(오브젝트를 참조하는 명령문이 컴파일될 때 존재해야 할 경우에도) 작성될 수 있습니다. 별명 체인을 따라 순환적으로 또는 반복적으로 참조하지 않는 경우, 다른 별명을 참조할 수 있습니다. 별명은 동일한 데이터베이스 내에서만 모듈, 별칭, 시퀀스, 테이블, 뷰 또는 다른 별명을 참조할 수 있습니다. CREATE TABLE 또는 CREATE VIEW 문과 같이 새로운 오브젝트 이름을 예측하는 곳에서는 별명 이름을 사용할 수 없습니다. 예를 들어, 별명 이름 PERSONNEL이 작성된 경우, CREATE TABLE PERSONNEL...과 같은 연속문은 오류를 리턴합니다.

별명으로 오브젝트를 참조하는 옵션은 구문 다이어그램에 명시적으로 표시되지 않거나 SQL문의 설명에서 언급되지 않습니다.

주어진 오브젝트 유형의 규정되지 않은 새 별명은, 시퀀스의 경우, 해당 오브젝트 유형의 기존 오브젝트와 동일한 완전한 이름을 가질 수 없습니다. 예를 들어, KANDIL 스키마의 ORDERID라는 시퀀스 별명 이름은 시퀀스 이름 KANDIL.ORDERID로 정의될 수 없습니다.

SQL문에서 별명을 사용하는 효과는 텍스트 대체 효과와 유사합니다. SQL문을 컴파일할 때 정의해야 하는 별명은 규정된 오브젝트 이름으로 명령문 컴파일 시 대체됩니다. 예를 들어, PBIRD.SALES가 DSPN014.DIST4_SALES_148의 별명인 경우, 컴파일 시 다음과 같이 대체됩니다.

```
SELECT * FROM PBIRD.SALES
SELECT * FROM DSPN014.DIST4_SALES_148
```

권한 부여 ID 및 권한 부여 이름

*권한 부여 ID*는 데이터베이스 관리 프로그램과 응용프로그램 프로세스 또는 프로그램 준비 프로세스 간의 연결을 설정할 때 데이터베이스 관리 프로그램에서 확보하는 문자열로서, 특권 세트를 지정합니다. 또한 사용자나 사용자 그룹을 지정할 수도 있지만, 이 등록 정보는 데이터베이스 관리 프로그램에서 제어하지는 않습니다.

데이터베이스 관리 프로그램은 권한 부여 ID를 사용하여 다음을 제공합니다.

- SQL문의 권한 부여 점검

- QUALIFIER 프리컴파일/바인드 옵션 및 CURRENT SCHEMA 특수 레지스터의 디폴트값. 권한 부여 ID는 디폴트 CURRENT PATH 특수 레지스터 및 FUNCPATH 프리컴파일/바인드 옵션에 포함되기도 합니다.

권한 부여 ID는 모든 SQL문에 적용됩니다. 정적 SQL문에 적용되는 권한 부여 ID는 프로그램의 바인딩 중에 사용되는 권한 부여 ID입니다. 동적 SQL문에 적용되는 권한 부여 ID는 바인드 시 제공되는 DYNAMICRULES 옵션과, 동적 SQL문을 발행하는 패키지의 현재 런타임 환경에 따라 다릅니다.

- 바인드 동작이 있는 패키지에서 사용되는 권한 부여 ID는 패키지 소유자의 권한 부여 ID입니다.
- 정의 동작이 있는 패키지에서 사용되는 권한 부여 ID는 해당 루틴 정의자의 권한 부여 ID입니다.
- 실행 동작이 있는 패키지에서 사용되는 권한 부여 ID는 패키지를 실행하는 사용자의 현재 권한 부여 ID입니다.
- 호출 동작이 있는 패키지에서 사용되는 권한 부여 ID는 루틴이 호출될 때 현재 유효한 권한 부여 ID입니다. 이를 런타임 권한 부여 ID라고 합니다.

자세한 정보는 72 페이지의 『런타임시 동적 SQL 등록 정보』의 내용을 참조하십시오.

SQL문에 지정된 *권한 부여 이름*을 명령문의 권한 부여 ID와 혼동해서는 안됩니다. 권한 부여 이름은 다양한 SQL문 내에서 사용되는 ID입니다. 또한 권한 부여 이름은 스키마의 소유자를 지정하기 위해 CREATE SCHEMA문에 사용되며, 권한 부여 또는 권한 취소 조작의 목표를 지정하기 위해 GRANT 및 REVOKE문에 사용됩니다. *X*에 특권을 부여한다는 것은 *X*(또는 그룹 또는 역할 *X*의 구성원)가 결과적으로 이러한 특권이 필요한 명령문의 권한 부여 ID가 된다는 것을 의미합니다.

예를 들면, 다음과 같습니다.

- SMITH가 응용프로그램 프로세스에서 연결을 설정할 때 데이터베이스 관리 프로그램에서 확보한 사용자 ID 및 권한 부여 ID라고 가정합니다. 다음 명령문은 대화식으로 실행됩니다.

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

  SMITH는 명령문의 권한 부여 ID입니다. 따라서 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터의 디폴트값이 SMITH이며, 정적 SQL에서는 QUALIFIER 프리컴파일/바인드 옵션의 디폴트값이 SMITH입니다. 명령문 실행 권한은 SMITH로 점검되며, SMITH는 63 페이지의 『이름 지정 규칙 및 내재된 오브젝트 이름 규정』에 설명된 자격 규칙을 기초로 한 내재된 *table-name* 규정자입니다.

  KEENE은 명령문에 지정된 권한 부여 이름입니다. KEENE에게 SMITH.TDEPT에 대한 SELECT 특권이 부여됩니다.

- SMITH에게 관리자 권한이 있으며 SMITH가 세션 동안 SET SCHEMA문이 발행되지 않은 다음 동적 SQL문의 권한 부여 ID라고 가정합시다.

  ```
  DROP TABLE TDEPT
  ```

  SMITH.TDEPT 테이블을 제거합니다.

  ```
  DROP TABLE SMITH.TDEPT
  ```

  SMITH.TDEPT 테이블을 제거합니다.

  ```
  DROP TABLE KEENE.TDEPT
  ```

  KEENE.TDEPT 테이블을 제거합니다. KEENE.TDEPT 및 SMITH.TDEPT가 서로 다른 테이블이라는 점에 유의하십시오.

  ```
  CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
  ```

  KEENE은 PAYROLL이라는 스키마를 작성하는 명령문에 지정된 권한 부여 이름입니다. KEENE은 스키마 PAYROLL의 소유자이므로, KEENE에게 CREATEIN, ALTERIN 및 DROPIN 특권이 부여되며 이러한 권한을 다른 사용자에게 부여할 수 있습니다.

런타임시 동적 SQL 등록 정보

바인드 옵션 DYNAMICRULES는 동적 SQL문 처리시 권한 점검에 사용되는 권한 부여 ID를 판별합니다. 또한 옵션은 규정되지 않은 오브젝트 참조에 사용되는 내재된 규정자 및 등록 정보 SQL문이 동적으로 호출될 수 있는지 여부와 같은 기타 동적 SQL 속성을 제어하기도 합니다.

권한 부여 ID 및 기타 동적 SQL 속성에 대한 값 세트를 동적 SQL문 동작이라고 합니다. 네 가지 가능한 동작은 실행, 바인드, 정의 및 호출입니다. 다음 표에서와 같이, DYNAMICRULES 바인드 옵션 값의 조합 및 런타임 환경에 따라 사용되는 동작이 결정됩니다. 실행 동작을 의미하는 DYNAMICRULES RUN이 디폴트값입니다.

표 7. DYNAMICRULES 및 런타임 환경에 따라 동적 SQL문 동작이 결정되는 방식

	동적 SQL문의 동작	
DYNAMICRULES 값	**독립형 프로그램 환경**	**루틴 환경**
BIND	바인드 동작	바인드 동작
RUN	실행 동작	실행 동작
DEFINEBIND	바인드 동작	정의 동작
DEFINERUN	실행 동작	정의 동작
INVOKEBIND	바인드 동작	호출 동작
INVOKERUN	실행 동작	호출 동작

실행 동작

DB2에서는 동적 SQL문의 권한 점검에 사용될 값 및 동적 SQL문에 있는 규

정되지 않은 오브젝트 참조의 내재된 규정에 사용될 초기값으로 패키지를 실행하는 사용자의 권한 부여 ID(DB2에 초기 연결된 ID)를 사용합니다.

바인드 동작

런타임시 DB2에서는 권한 및 규정에 필요한 정적 SQL에 적용되는 모든 규칙을 사용합니다. 동적 SQL문의 권한 부여 점검에 사용할 값으로 패키지 소유자의 권한 부여 ID를 사용하며 동적 SQL문 내에 있는 규정되지 않은 오브젝트 참조의 내재된 규정에 대해 패키지 디폴트 규정자를 사용합니다.

정의 동작

동적 SQL문이 루틴 컨텍스트 내에서 실행되는 패키지에 있고 패키지가 DYNAMICRULES DEFINEBIND 또는 DYNAMICRULES DEFINERUN을 사용하여 바인드된 경우에만 적용됩니다. DB2에서는 동적 SQL문의 권한 점검에 사용될 값 및 해당 루틴에 있는 동적 SQL문의 규정되지 않은 오브젝트 참조의 내재된 규정에 사용될 값으로 루틴 정의자(루틴의 패키지 바인더가 아님)의 권한 부여 ID를 사용합니다.

호출 동작

동적 SQL문이 루틴 컨텍스트 내에서 실행되는 패키지에 있고 패키지가 DYNAMICRULES INVOKEBIND 또는 DYNAMICRULES INVOKERUN을 사용하여 바인드된 경우에만 적용됩니다. DB2에서는 동적 SQL의 권한 점검에 사용될 값 및 해당 루틴에 있는 동적 SQL문의 규정되지 않은 오브젝트 참조의 내재된 규정에 사용될 값으로 루틴 호출시 유효한 명령문 권한 부여 ID를 사용합니다. 이러한 내용이 다음 표에 요약되어 있습니다.

호출 환경	사용되는 ID
정적 SQL	루틴을 호출하는 SQL이 있는 패키지 소유자의 내재된 또는 명시된 값
뷰 또는 트리거 정의에 사용	뷰 또는 트리거의 정의자
바인드 동작 패키지의 동적 SQL	루틴을 호출하는 SQL이 있는 패키지 소유자의 내재된 또는 명시된 값
실행 동작 패키지의 동적 SQL	DB2에 처음 연결하는 데 사용되는 ID
정의 동작 패키지의 동적 SQL	루틴을 호출하는 SQL이 있는 패키지를 사용하는 루틴의 정의자
호출 동작 패키지의 동적 SQL	루틴을 호출하는 현재 권한 부여 ID

실행 동작이 적용되지 않을 경우의 제한된 명령문

바인드, 정의 또는 호출 동작이 적용되면 GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT, RENAME, SET INTEGRITY, SET EVENT MONITOR STATE와 같은 동적 SQL문이나 별칭을 참조하는 쿼리를 사용할 수 없습니다.

DYNAMICRULES 옵션에 관한 고려사항

CURRENT SCHEMA 특수 레지스터는 바인드, 정의 또는 호출 동작 패키지에서 실행되는 동적 SQL문 내의 규정되지 않은 오브젝트 참조를 규정하는 데 사용될 수 없습니다. SET CURRENT SCHEMA문을 발행하여 CURRENT SCHEMA 특수 레지스터를 변경한 후에도 이 사항이 해당되며, 레지스터값은 변경되나 사용되지는 않습니다.

단일 연결시 복수 패키지가 참조될 경우 해당 패키지에서 준비한 모든 동적 SQL문은 사용되는 특정 패키지 및 사용 환경에 대해 DYNAMICRULES 옵션에서 지정한 동작을 수행합니다.

패키지가 바인드 동작을 나타낼 때 동적 명령문은 패키지 소유자의 권한 부여 ID를 사용하게 되므로 패키지 바인더에게 패키지 사용자가 받아서는 안 되는 권한을 부여하면 안 된다는 점을 명심하십시오. 마찬가지로, 패키지가 정의 동작을 나타낼 때 루틴의 정의자에게 패키지 사용자가 받아서는 안 되는 권한을 부여해서는 안됩니다.

권한 부여 ID 및 명령문 준비

바인드 시 VALIDATE BIND 옵션을 지정할 경우, 테이블 및 뷰를 조작하는 데 필요한 특권도 바인드 시 존재해야 합니다. 이러한 특권이나 참조된 오브젝트가 존재하지 않고 SQLERROR NOPACKAGE 옵션이 적용되는 경우, 바인드 조작에 실패합니다. SQLERROR CONTINUE 옵션을 지정할 경우, 바인드 조작은 성공하지만 오류가 있는 모든 명령문에는 플래그가 표시됩니다. 이러한 명령문을 실행하려고 하면 오류가 발생합니다.

패키지가 VALIDATE RUN 옵션을 사용하여 바인드되는 경우, 정상적인 바인드 프로세싱은 완료되지만 응용프로그램에서 참조된 테이블 및 뷰를 사용하는 데 필요한 특권이 아직 존재하지 않아도 됩니다. 바인드 시 필수 특권이 존재하지 않는 경우, 명령문이 응용프로그램에서 처음 실행될 때마다 증분 바인드 조작이 수행되며 명령문에 필요한 모든 특권이 존재해야 합니다. 필수 특권이 존재하지 않는 경우, 명령문 실행에 실패합니다.

런타임시 권한 부여 점검은 패키지 소유자의 권한 부여 ID를 사용하여 수행됩니다.

컬럼 이름

*컬럼 이름*의 의미는 상황에 따라 다릅니다. 컬럼 이름은 다음을 수행하는 데 사용될 수 있습니다.

- CREATE TABLE문에서와 같이 컬럼 이름 선언
- CREATE INDEX문에서와 같이 컬럼 식별
- 다음 컨텍스트에서와 같이 컬럼값 지정
  - 집계 함수에서 컬럼 이름은 함수가 적용되는 그룹이나 중간 결과 테이블에 있는 컬럼의 모든 값을 지정합니다. 예를 들어, MAX(SALARY)는 MAX 함수를 그룹에 있는 SALARY 컬럼의 모든 값에 적용합니다.

  - GROUP BY나 ORDER BY절에서 컬럼 이름은 절이 적용되는 중간 결과 테이블에 있는 모든 값을 지정합니다. 예를 들어, ORDER BY DEPT는 DEPT 컬럼의 값에서 결과 테이블의 순서를 지정합니다.
  - 표현식, 검색 조건 또는 스칼라 함수에서 컬럼 이름은 구문이 적용되는 각 행이나 그룹에 대한 값을 지정합니다. 예를 들어, 검색 조건 CODE = 20이 일부 행에 적용되면 컬럼 이름 CODE로 지정된 값은 그 행에 있는 CODE 컬럼의 값입니다.
- FROM절의 *table-reference*에 있는 *correlation-clause*에서와 같이 일시적으로 컬럼의 이름 바꾸기

규정된 컬럼 이름

컬럼 이름에 대한 규정자는 테이블, 뷰, 별칭, 별명 또는 상관 이름이 될 수 있습니다.

컬럼 이름이 규정될 수 있는지의 여부는 상황에 따라 다릅니다.

- COMMENT ON문의 양식에 따라 단일 컬럼 이름을 규정해야 합니다. 복수 컬럼 이름은 규정될 수 없습니다.
- 컬럼 이름이 컬럼 값을 지정하는 경우 사용자 옵션에서 규정됩니다.
- UPDATE문의 assignment-clause에서 컬럼 이름은 사용자 옵션에서 규정됩니다.
- 다른 모든 컨텍스트에서는 컬럼 이름을 규정할 수 없습니다.

규정자가 선택적인 경우 컬럼 이름은 두 가지 목적을 제공합니다. 이 점에 대해서는 77 페이지의 『앰비규어티를 방지하는 컬럼 이름 규정자』 및 80 페이지의 『상관 참조의 컬럼 이름 규정자』에서 설명합니다.

상관 이름

*상관 이름*은 쿼리의 FROM절과 UPDATE나 DELETE문의 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 이 SELECT문에 있는 X.MYTABLE 인스턴스의 컬럼에 대한 참조를 규정하는 데 Z만 사용할 수 있습니다.

상관 이름은 정의된 컨텍스트 내에서만 테이블, 뷰, 별칭, 별명, 중첩 테이블 표현식, 테이블 함수, 또는 데이터 변경 테이블 참조와만 연관됩니다. 그러므로 같은 상관 이름은 다른 명령문에서 또는 같은 명령문의 다른 절에서 다른 목적으로 정의될 수 있습니다.

규정자로서 상관 이름은 앰비규어티를 방지하거나 상관 참조를 설정하는 데 사용될 수 있으며, 또한 테이블 참조의 단축 이름으로만 사용될 수 있습니다. 이 예에서, Z는 X.MYTABLE을 두 번 이상 입력하지 않기 위해 사용되었습니다.

테이블, 뷰, 별칭 또는 별명 이름에 대해 상관 이름을 지정한 경우, 테이블, 뷰, 별칭 또는 별명의 해당 인스턴스 컬럼에 대한 규정된 참조에는 테이블, 뷰, 별칭 또는 별명 이름이 아닌 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 *표시 이름*이거나 *비표시 이름*입니다. 상관 이름을 지정하지 않은 경우, 테이블, 뷰, 별칭 또는 별명 이름이 FROM절에 표시되어 있다고 합니다. 상관 이름은 항상 표시 이름입니다. 예를 들어, 다음 FROM절에서 상관 이름은 DEPARTMENT가 아닌 EMPLOYEE에 대해 지정되므로, DEPARTMENT는 표시 이름이고 EMPLOYEE는 표시 이름이 아닙니다.

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM절에 표시된 테이블, 뷰, 별칭 또는 별명 이름은 그 FROM절에 표시된 다른 테이블 이름, 뷰 이름 또는 별칭과 같거나, FROM절에 있는 상관 이름과 같을 수 있습니다. 이로 인해 앰비규어스 컬럼 이름 참조가 발생하여 오류(SQLSTATE 42702)가 리턴될 수 있습니다.

각 FROM절에는 표시된 EMPLOYEE에 대한 참조가 한 개만 포함되므로 아래에 표시된 처음 두 FROM절은 올바릅니다.

1. FROM절에서

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   EMPLOYEE.PROJECT와 같은 규정된 참조는 FROM절에서 EMPLOYEE의 첫 번째 인스턴스의 컬럼을 표시합니다. EMPLOYEE의 첫 번째 인스턴스에 대한 규정된 참조는 상관 이름 “E1”(E1.PROJECT)을 사용해야 합니다.

2. FROM절에서

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

   EMPLOYEE.PROJECT와 같은 규정된 참조는 FROM절에서 EMPLOYEE의 첫 번째 인스턴스의 컬럼을 표시합니다. EMPLOYEE의 두 번째 인스턴스에 대한 규정 참조는 상관 이름 “E2”(E2.PROJECT)를 사용해야 합니다.

3. FROM절에서

```
FROM EMPLOYEE, EMPLOYEE
```

   이 절에 포함된 두 개의 표시 테이블 이름(EMPLOYEE와 EMPLOYEE)이 동일합니다. 동일한 이름은 허용되지만 특정 컬럼 이름에 대한 참조가 앰비규어스할 수 있습니다(SQLSTATE 42702).

4. 다음 명령문에서

```
SELECT *
  FROM EMPLOYEE E1, EMPLOYEE E2             * incorrect *
  WHERE EMPLOYEE.PROJECT = 'ABC'
```

   FROM절에 있는 EMPLOYEE의 인스턴스에 모두 상관 이름이 있으므로 규정된 참조 EMPLOYEE.PROJECT는 올바르지 않습니다. 대신, PROJECT에 대한 참조가 상관 이름(E1.PROJECT 또는 E2.PROJECT)으로 규정되어야 합니다.

5. FROM절에서

```
FROM EMPLOYEE, X.EMPLOYEE
```

   EMPLOYEE의 두 번째 인스턴스에 있는 컬럼에 대한 참조는 X.EMPLOYEE (X.EMPLOYEE.PROJECT)를 사용해야 합니다. 동적 SQL 또는 정적 SQL의 QUALIFIER 프리컴파일/바인드 옵션의 CURRENT SCHEMA 특수 레지스터값이 X일 경우, 이러한 참조는 앰비규어스하기 때문에 컬럼을 참조할 수 없습니다.

FROM절에서 상관 이름을 사용하면 결과 테이블의 컬럼과 연관될 컬럼 이름 목록을 지정하는 옵션을 사용할 수 있습니다. 상관 이름에서와 마찬가지로, 나열되는 이러한 이름들이 쿼리 전반에 걸쳐 컬럼에 대한 참조로 사용되어야 하는 컬럼의 *표시* 이름이 됩니다. 컬럼 이름 목록을 지정할 경우, 하위 테이블의 컬럼 이름은 *비표시* 이름이 됩니다.

FROM절에서

```
FROM DEPARTMENT D (NUM,NAME,MGR,ANUM,LOC)
```

D.NUM과 같은 규정된 참조는 테이블에서 DEPTNO로 정의된 DEPARTMENT 테이블의 첫 번째 컬럼을 표시합니다. 이러한 FROM절을 사용한 D.DEPTNO에 대한 참조는 컬럼 이름 DEPTNO가 비표시 컬럼 이름이므로 올바르지 않습니다.

앰비규어티를 방지하는 컬럼 이름 규정자

함수, GROUP BY절, ORDER BY절, 표현식 또는 검색 조건 컨텍스트에서 컬럼 이름은 일부 테이블, 뷰, 별칭, 중첩 테이블 표현식 또는 테이블 함수의 컬럼 값을 참조합니다. 컬럼을 포함할 수 있는 테이블, 뷰, 별칭, 중첩 테이블 표현식 및 테이블을 컨텍스트의 *오브젝트 테이블*이라고 합니다. 둘 이상의 오브젝트 테이블에는 같은 이름의 컬럼이 있을 수 있습니다. 컬럼 이름을 규정하는 한 가지 이유는 컬럼이 파생된 테이블을

지정하기 위해서입니다. 컬럼 이름의 규정자는 또한 SQL문에 사용되는 SQL 변수 이름을 컬럼 이름과 구별하기 위한 SQL 프로시저에서 유용합니다.

중첩 테이블 표현식이나 테이블 함수는 FROM절에서 선행하는 *테이블 참조*를 오브젝트 테이블로 간주합니다. 뒤에 오는 *테이블 참조*는 오브젝트 테이블로 간주되지 않습니다.

테이블 지정자

특정 오브젝트 테이블을 지정하는 규정자를 *테이블 지정자*라고 합니다. 오브젝트 테이블을 식별하는 절에서는 이들에 대한 테이블 지정자도 설정합니다. 예를 들어, SELECT 절에 있는 표현식의 오브젝트 테이블은 FROM절에서 다음과 같이 이름이 지정됩니다.

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
  FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

FROM절에서 테이블 지정자는 다음과 같이 설정됩니다.

- 테이블, 뷰, 별칭, 별명, 중첩 테이블 표현식 또는 테이블 함수 뒤에 나오는 이름은 상관 이름과 테이블 지정자입니다. 그러므로 CORZ는 테이블 지정자입니다. CORZ는 선택 목록의 첫 번째 컬럼 이름을 규정하는 데 사용됩니다.
- 표시 테이블, 뷰 이름, 별칭 또는 별명은 테이블 지정자입니다. 그러므로 OWNY.MYTABLE은 테이블 지정자입니다. OWNY.MYTABLE은 선택 목록의 두 번째 컬럼 이름을 규정하는 데 사용됩니다.

테이블 지정자의 표시 테이블 이름 형식으로 컬럼을 규정할 때 규정되거나 규정되지 않은 형식의 표시 테이블 이름을 사용할 수 있습니다. 규정된 형식이 사용되면, 규정자는 표시 테이블 이름의 기본 규정자와 같아야 합니다.

예를 들어, 현재 스키마가 CORPDATA라고 간주합니다.

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT FROM EMPLOYEE
```

는 FROM절에서 참조된 EMPLOYEE 테이블이 CORPDATA.EMPLOYEE를 완전히 규정하므로 유효하며, WORKDEPT 컬럼의 규정자와 일치합니다.

```
SELECT EMPLOYEE.WORKDEPT, REGEMP.WORKDEPT
  FROM CORPDATA.EMPLOYEE, REGION.EMPLOYEE REGEMP
```

는 첫 번째 선택 목록 컬럼이 FROM절에 있는 규정되지 않은 표시 테이블 지정자 CORPDATA.EMPLOYEE를 참조하고, 두 번째 선택 목록 컬럼은 FROM절에 있는 테이블 오브젝트 REGION.EMPLOYEE의 상관 이름 REGEMP를 참조하므로 유효합니다.

현재 스키마가 REGION이라고 간주합니다.

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT FROM EMPLOYEE
```

는 FROM절에서 참조된 EMPLOYEE 테이블이 REGION.EMPLOYEE를 완전히 규정하고 WORKDEPT 컬럼의 규정자가 CORPDATA.EMPLOYEE 테이블을 나타내므로 유효하지 않습니다.

컬럼에 대한 앰비규어스 참조의 가능성을 방지하려면 각 테이블 지정자는 특정 FROM 절 내에서 고유해야 합니다.

미정의 또는 앰비규어스 참조 방지

컬럼 이름이 컬럼 값을 참조할 때 정확히 하나의 오브젝트 테이블이 해당 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. 이 경우에도 참조가 정의되지 않았습니다.
- 이름이 규정되지 않았고, 둘 이상의 오브젝트 테이블에 해당 이름의 컬럼이 있습니다. 참조가 앰비규어스합니다.
- 컬럼 이름이 테이블 지정자에 의해 규정되었지만 지정된 테이블이 FROM절에서 고유하지 않고, 지정된 테이블의 어커런스가 컬럼에 있습니다. 참조가 앰비규어스합니다.
- 컬럼 이름이 TABLE 키워드가 선행하지 않는 중첩 테이블 표현식에 있거나, 오른쪽 외부 조인 또는 완전 외부 조인의 오른쪽 피연산자인 중첩 테이블 표현식 또는 테이블 함수에 있습니다. 이 컬럼 이름은 중첩 테이블 표현식의 fullselect 내의 *table-reference* 컬럼을 참조하지 않습니다. 참조가 정의되지 않았습니다.

고유하게 정의된 테이블 지정자로 컬럼 이름을 규정하여 앰비규어스 참조를 방지하십시오. 컬럼이 다른 이름의 여러 오브젝트 테이블에 포함된 경우, 테이블 이름을 지정자로 사용할 수 있습니다. 상관 이름 다음의 컬럼 이름 목록을 사용하여 한 오브젝트 테이블의 컬럼에 고유 이름을 제공하면 테이블 지정자를 사용하지 않아도 앰비규어스 참조를 방지할 수 있습니다.

테이블 지정자의 표시 테이블 이름 형식으로 컬럼을 규정할 때 규정되거나 규정되지 않은 형식의 표시 테이블 이름을 사용할 수 있습니다. 그러나 사용된 규정자와 사용된 테이블은 테이블 이름, 뷰 이름 또는 별칭과 테이블 지정자를 완전히 규정한 이후에도 같아야 합니다.

1. 명령문의 권한 부여 ID가 CORPDATA인 경우,

   ```
   SELECT CORPDATA.EMPLOYEE.WORKDEPT
     FROM EMPLOYEE
   ```

   위 명령문은 유효한 명령문입니다.

2. 명령문의 권한 부여 ID가 REGION인 경우, 다음은 유효하지 않은 명령문입니다.

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
  FROM EMPLOYEE                              * incorrect *
```

이는 EMPLOYEE가 테이블 REGION.EMPLOYEE를 나타내지만 WORKDEPT에 대한 규정자는 다른 테이블 CORPDATA.EMPLOYEE를 나타내기 때문입니다.

상관 참조의 컬럼 이름 규정자

*fullselect*는 다양한 SQL문의 구성요소로 사용될 수 있는 쿼리 양식입니다. 명령문의 검색 조건 내에서 사용된 fullselect를 *서브쿼리*라고 합니다. 단일 값을 검색하기 위해 명령문 내의 표현식으로 사용되는 fullselect를 *스칼라 fullselect* 또는 *스칼라 서브쿼리*라고 합니다. 쿼리의 FROM절에 사용된 fullselect를 *중첩 테이블 표현식*이라고 합니다. 검색 조건의 서브쿼리, 스칼라 서브쿼리 및 중첩 테이블 표현식은 이 주제 항목의 나머지 부분에서 서브쿼리로 지칭됩니다.

서브쿼리는 서브쿼리 자체를 포함할 수 있으며, 이 안에 다시 서브쿼리가 포함될 수 있습니다. 그러므로 SQL문은 서브쿼리의 계층 구조를 포함할 수 있습니다. 서브쿼리가 포함된 계층 구조의 요소는 이들이 포함된 서브쿼리보다 상위 레벨에 있다고 합니다.

계층 구조의 모든 요소에는 하나 이상의 테이블 지정자가 있습니다. 서브쿼리는 계층 구조의 해당 고유 레벨에서 식별된 테이블의 컬럼뿐만 아니라 이전에 식별된 계층 구조 내 테이블의 컬럼까지 계층 구조의 상위 레벨로 참조할 수 있습니다. 상위 레벨에서 식별된 테이블의 컬럼에 대한 참조를 *상관 참조*라고 합니다.

SQL 기존 표준과의 호환성을 위해 규정된 컬럼 이름과 규정되지 않은 컬럼 이름이 모두 상관 참조로 허용됩니다. 그러나 서브쿼리에 사용된 모든 컬럼 참조를 규정하는 것이 바람직하며, 그렇지 않을 경우에는 동일한 컬럼 이름으로 인해 의도하지 않은 결과가 초래될 수 있습니다. 예를 들어, 한 계층 구조에 있는 테이블이 상관 참조와 동일한 컬럼 이름을 포함하기 위해 변경되고 명령문이 다시 준비되는 경우, 참조는 변경된 테이블에 적용됩니다.

서브쿼리에 있는 컬럼 이름이 규정되면 규정된 컬럼 이름이 나타나는 동일한 서브쿼리에서 시작하여 규정자와 일치하는 테이블 지정자를 찾을 때까지 계층 구조의 각 레벨이 계층 구조의 상위 레벨까지 계속 검색됩니다. 일단 발견되면 테이블이 주어진 컬럼을 포함하는지 확인합니다. 테이블을 컬럼 이름이 들어 있는 레벨보다 상위에서 발견한 경우, 이것은 테이블 지정자가 발견된 레벨의 상관 참조입니다. 중첩 테이블 표현식의 fullselect 상위 계층 구조를 검색하려면 선택적 TABLE 키워드가 중첩 테이블 표현식보다 선행되어야 합니다.

서브쿼리에 있는 컬럼 이름이 규정되지 않으면 컬럼 이름이 나타나는 동일한 서브쿼리에서 시작하여 일치하는 컬럼 이름을 찾을 때까지 계층 구조의 각 레벨에서 참조된 테이블이 계층 구조의 상위 레벨까지 계속 검색됩니다. 컬럼을 컬럼 이름이 들어 있는 레벨보다 상위 레벨에 있는 테이블에서 발견한 경우, 이는 컬럼이 들어 있는 테이블이 발

견된 레벨의 상관 참조입니다. 컬럼 이름이 특정 레벨에 있는 둘 이상의 테이블에서 발견되는 경우, 참조가 앰비규어스하므로, 오류로 간주됩니다.

이 경우, 다음 예에서 사용된 T는 컬럼 C가 들어 있는 테이블 지정자를 참조합니다. 컬럼 이름 T.C(여기서, T는 내재적 또는 명시적 규정자를 나타냄)는 이러한 조건을 충족시킬 경우에만 상관 참조가 됩니다.

- T.C는 서브쿼리의 표현식에 사용됩니다.
- T는 서브쿼리의 FROM절에 사용된 테이블을 지정하지 않습니다.
- T는 서브쿼리가 들어 있는 계층 구조의 상위 레벨에서 사용된 테이블을 지정합니다.

같은 테이블, 뷰 또는 별칭을 여러 레벨에서 식별할 수 있으므로, 테이블 지정자로 고유한 상관 이름을 사용하는 것이 좋습니다. 둘 이상의 레벨에서 테이블을 지정하기 위해 T가 사용될 경우(T는 그 자체가 테이블 이름이거나 중복되는 상관 이름임), T.C는 대부분 T.C를 포함하는 서브쿼리를 직접 포함하는 T가 사용되는 레벨을 참조합니다. 상위 레벨에 대한 상관이 필요할 경우, 고유한 상관 이름을 사용해야 합니다.

상관 참조 T.C는 두 개의 검색 조건(서브쿼리의 경우 조건 1, 상위 레벨의 경우 조건 2)이 적용되는 행이나 그룹 T에 있는 값 C를 식별합니다. 조건 2가 WHERE절에 사용되는 경우, 서브쿼리는 조건 2가 적용되는 각 행에 대해 평가됩니다. 조건 2가 HAVING절에 사용되는 경우, 서브쿼리는 조건 2가 적용되는 각 그룹에 대해 평가됩니다.

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.) 이 명령문은 해당 부서의 평균 급여보다 급여가 적은 직원을 나열합니다.

```
SELECT EMPNO, LASTNAME, WORKDEPT
  FROM EMPLOYEE X
  WHERE SALARY < (SELECT AVG(SALARY)
                  FROM EMPLOYEE
                  WHERE WORKDEPT = X.WORKDEPT)
```

다음 예에서는 상관 이름으로 THIS를 사용합니다. 이 명령문은 직원이 없는 부서에 대한 행을 삭제합니다.

```
DELETE FROM DEPARTMENT THIS
   WHERE NOT EXISTS(SELECT *
                    FROM EMPLOYEE
                    WHERE WORKDEPT = THIS.DEPTNO)
```

변수에 대한 참조

SQL문의 *변수*는 SQL문이 실행될 때 변경할 수 있는 값을 지정합니다. 다음은 SQL문에서 사용되는 여러 유형의 변수입니다.

호스트 변수

호스트 변수는 호스트 언어의 명령문에서 정의됩니다. 호스트 변수 참조 방법에 관한 자세한 정보는 『호스트 변수에 대한 참조』의 내용을 참조하십시오.

전이 변수

전이 변수는 트리거로 정의되며 컬럼의 이전 또는 새로운 값을 참조합니다. 전이 변수 참조 방법에 관한 자세한 정보는 『CREATE TRIGGER 명령문』(*SQL 참조서, 볼륨 2*에서)을 참조하십시오.

SQL 변수

SQL 변수는 SQL 함수, SQL 메소드, SQL 프로시저, 트리거 또는 동적 SQL 문의 SQL 복합 명령문으로 정의됩니다. SQL 변수에 관한 자세한 정보는 『SQL 매개변수, SQL 변수 및 글로벌 변수에 대한 참조』(*SQL 참조서, 볼륨 2*에서)를 참조하십시오.

전역 변수

전역 변수는 CREATE VARIABLE문에서 정의됩니다. 전역 변수에 관한 자세한 정보는 『CREATE VARIABLE』 및 『SQL 매개변수, SQL 변수 및 전역 변수에 대한 참조』(*SQL 참조서, 볼륨 2*에서)를 참조하십시오.

모듈 변수

모듈 변수는 ADD VARIABLE 또는 PUBLISH VARIABLE 조작을 사용하는 ALTER MODULE문으로 정의됩니다. 모듈 변수에 대한 자세한 정보는 『ALTER MODULE』을 참조하십시오.

SQL 매개변수

SQL 매개변수는 CREATE FUNCTION, CREATE METHOD 또는 CREATE PROCEDURE문에서 정의됩니다. SQL 매개변수에 관한 자세한 정보는 『SQL 매개변수, SQL 변수 및 글로벌 변수에 대한 참조』(*SQL 참조서, 볼륨 2*에서)를 참조하십시오.

매개변수 표시문자

매개변수 표시문자는 명령문이 정적 SQL문인 경우 호스트 변수가 지정되는 동적 SQL문에서 지정됩니다. 동적 SQL문이 처리되는 동안, 바인딩되는 SQL 디스크립터 또는 매개변수는 값을 매개변수 표시문자와 연관시키는 데 사용됩니다. 매개변수 표시문자에 관한 자세한 정보는 『매개변수 표시문자』(*SQL 참조서, 볼륨 2*)를 참조하십시오.

호스트 변수에 대한 참조

*호스트 변수*는 다음 중 하나입니다.

- 호스트 언어 변수(예: C 변수, C++ 변수, COBOL 데이터 항목, FORTRAN 변수 또는 Java 변수)

또는

- SQL문에서 참조되는 SQL 확장자를 사용하여 선언된 변수에서 SQL 프리컴파일러에 의해 생성된 호스트 언어 구문

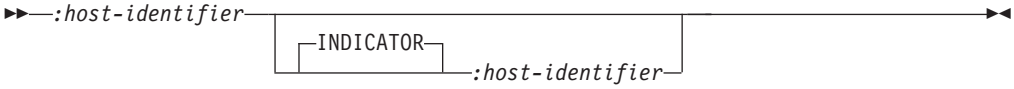
호스트 변수는 호스트 언어에 있는 명령문에 의해 직접 정의되거나 SQL 확장자를 사용하여 간접적으로 정의됩니다.

SQL문에 있는 호스트 변수는 호스트 변수 선언 규칙에 따라 프로그램에서 설명한 호스트 변수를 식별해야 합니다.

SQL문에 사용되는 모든 호스트 변수는 REXX™를 제외한 모든 호스트 언어로 SQL DECLARE 섹션에 선언되어야 합니다. SQL DECLARE 섹션에 선언된 변수와 동일한 이름으로 SQL DECLARE 섹션 외부에서 변수를 선언할 수 없습니다. SQL DECLARE 섹션은 BEGIN DECLARE SECTION으로 시작하여 END DECLARE SECTION으로 끝납니다.

구문 도표에 사용된 것처럼 메타 변수 *host-variable*은 호스트 변수에 대한 참조를 나타냅니다. SET 변수 명령문이나 FETCH, SELECT INTO 또는 VALUES INTO 명령문의 INTO 절에서 목표 변수로 존재하는 호스트 변수는 행의 컬럼 또는 표현식의 값이 지정될 호스트 변수를 식별합니다. 기타 모든 컨텍스트에서 호스트 변수는 응용프로그램에서 데이터베이스 관리 프로그램으로 전달된 값을 지정합니다.

구문 도표에 있는 메타 변수 *host-variable*은 일반적으로 다음과 같이 확장될 수 있습니다.



각 *host-identifier*는 소스 프로그램에 선언되어야 합니다. 두 번째 host-identifier로 지정된 변수의 데이터 유형은 작은 정수여야 합니다.

첫 번째 host-identifier는 *기본 변수*를 지정합니다. 조작에 따라 데이터베이스 관리 프로그램에 값을 제공하거나 데이터베이스 관리 프로그램에서 값을 제공받습니다. 입력 호스트 변수는 런타임 응용프로그램 코드 페이지에 값을 제공합니다. 출력 호스트 변수에는 필요한 경우 데이터가 출력 응용프로그램 변수로 복사될 때 런타임 응용프로그램 코드 페이지로 변환되는 값이 제공됩니다. 제공된 호스트 변수는 같은 프로그램에서 입력과 출력 변수의 역할을 모두 수행할 수 있습니다.

두 번째 host-identifier는 *표시기 변수*를 지정합니다. 표시기 변수의 목적은 다음 작업을 수행하는 것입니다:

- 널(NULL) 값을 지정합니다. 표시기 변수의 음수 값은 널(NULL) 값을 지정합니다. 값 -2는 결과를 파생할 때 발생한 산술 연산식 오류나 숫자 변환을 나타냅니다.

- 절단된 문자열의 원래 길이를 기록합니다(값의 소스가 대형 오브젝트(LOB) 유형이 아닌 경우).
- 시간이 호스트 변수에 지정될 때 절단된 경우 시간의 초 부분을 기록합니다.

예를 들어, :HV1:HV2를 사용하여 삽입 또는 갱신 값을 지정하는 경우, 그리고 HV2가 음수일 경우, 지정된 값은 널(NULL) 값입니다. HV2가 음수가 아닌 경우, 지정되는 값은 HV1의 값입니다.

마찬가지로 :HV1:HV2가 FETCH의 INTO 절, SELECT INTO 또는 VALUES INTO 문에 지정된 경우 및 리턴된 값이 널(NULL)인 경우 HV1은 변경되지 않고 HV2는 음수 값으로 설정됩니다. DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우(또는 정적 SQL문 바인드 시 구성된 경우) HV2는 -2일 수 있습니다. HV2가 -2인 경우, 숫자 유형 HV1으로 변환시 발생하는 오류나 HV1의 값을 판별하는 데 사용되는 산술 연산식 평가시 발생하는 오류로 인해 HV1 값이 리턴될 수 없습니다. DB2 Universal Database 버전 5 이전의 클라이언트 버전으로 데이터베이스에 액세스할 때 HV2는 산술 예외로 -1이 됩니다. 리턴된 값이 널(NULL)이 아닌 경우, 이 값은 HV1으로 지정되고 HV2는 0으로 설정됩니다. (이는 HV1으로 지정할 때 비LOB 문자열의 문자열 절단이 필요하지 않는 경우, 비LOB 문자열의 절단이 필요한 때에는 HV2가 문자열의 원래 길이로 설정됩니다.) 지정할 때 시간의 초 부분이 절단되어야 하는 경우, HV2는 초 수로 설정됩니다.

두 번째 호스트 ID를 생략할 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) 값을 포함할 수 있으면 INTO절에 사용해서는 안됩니다. 이 형식이 사용되고 컬럼에 널(NULL)이 포함된 경우, 런타임시 데이터베이스 관리 프로그램에서 오류가 발생합니다.

호스트 변수를 참조하는 SQL문은 이러한 호스트 변수 선언 범위 내에 있어야 합니다. 커서의 SELECT문에서 참조되는 호스트 변수의 경우, 이 규칙은 DECLARE CURSOR 문이 아닌 OPEN문에 적용됩니다.

예 :

PROJECT 테이블을 사용하여 호스트 변수 PNAME(VARCHAR(26))을 프로젝트 이름(PROJNAME)으로 설정하고 호스트 변수 STAFF(dec(5,2))를 중간 직원 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)이 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 고려사항: 호스트 변수 이름에 복수 바이트 문자가 사용되는지 여부는 호스트 언어에 따라 달라집니다.

동적 SQL의 호스트 변수

동적 SQL문에서 매개변수 표시문자는 호스트 변수 대신 사용됩니다. 매개변수 표시문자는 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 나타냅니다.

```
INSERT INTO DEPARTMENT
  VALUES (:hv_deptno, :hv_deptname, :hv_mgrno, :hv_admrdept)
```

이 예는 unnamed 매개변수 표시문자를 사용하여 동적 SQL문을 나타냅니다.

```
INSERT INTO DEPARTMENT VALUES (?, ?, ?, ?)
```

이 예는 매개변수 표시문자를 사용하는 동적 SQL문을 나타냅니다.

```
INSERT INTO DEPARTMENT
  VALUES(:deptno, :deptname, :mgrno, :admrdept)
```

이름 지정된 매개변수 표시문자는 동적문의 가독성을 개선하는 데 사용될 수 있습니다. 이름 지정된 매개변수 표시문자가 호스트 변수와 유사하지만 이름 지정된 매개변수 표시문자는 연관된 값이 없으므로 명령문이 실행될 때 매개변수 표시문자에 대한 값이 제공되어야 합니다. 이름지정된 매개변수 표시문자를 사용하는 INSERT문이 준비되고 준비된 명령문 DYNSTMT가 제공되면 다음 명령문을 사용하여 매개변수 표시문자 값을 제공할 수 있습니다.

```
EXECUTE DYNSTMT
  USING :hv_deptno, :hv_deptname :hv_mgrno, :hv_admrdept
```

이름지정되지 않은 매개변수 표시문자를 사용하는 INSERT문이 준비되고 준비된 명령문 이름 DYNSTMT가 제공되면 동일한 EXECUTE문을 사용할 수 있습니다.

BLOB, CLOB 및 DBCLOB 호스트 변수에 대한 참조

일반적인 BLOB, CLOB 및 DBCLOB 변수, LOB 로케이터 변수(86 페이지의 『로케이터 변수에 대한 참조』 참조) 및 LOB 파일 참조 변수(86 페이지의 『BLOB, CLOB 및 DBCLOB 파일 참조 변수에 대한 참조』 참조)는 모든 호스트 언어로 정의될 수 있습니다. LOB가 허용되는 경우, 구문 도표에서 *host-variable*이라는 용어는 일반 호스트 변수, 로케이터 변수 또는 파일 참조 변수를 의미할 수 있습니다. 이것은 원시(native) 데이터 유형이 아니므로, SQL 확장자가 사용되며 프리컴파일러는 각 변수를 나타내는 데 필요한 호스트 언어 구문을 생성합니다. REXX의 경우, LOB는 문자열에 맵핑됩니다.

때때로 대형 오브젝트(LOB) 값 전체를 보관하기에 충분히 큰 변수를 정의할 수도 있습니다. 이것이 가능하고 서버에서 데이터 전송을 지연시킴으로써 얻게 될 성능상의 이

익이 없는 경우, 로케이터는 필요하지 않습니다. 그러나 호스트 언어나 공간 제한사항이 임시 스토리지에 전체 대형 오브젝트를 한 번에 저장하는 것에 대해 설명하거나 성능상의 이익이 있으므로, 대형 오브젝트는 한 번에 대형 오브젝트 부분만을 포함하는 호스트 변수에서 갱신하거나 호스트 변수에서 선택될 수 있는 오브젝트 부분과 로케이터를 통해 대형 오브젝트를 참조할 수 있습니다.

로케이터 변수에 대한 참조

*로케이터 변수*는 응용프로그램 서버(AS)에서 LOB 값을 나타내는 로케이터를 포함하는 호스트 변수입니다.

SQL문에 있는 로케이터 변수는 로케이터 변수 선언 규칙에 따라 프로그램에서 설명한 로케이터 변수를 식별해야 합니다. 이것은 언제나 SQL문을 통해 간접적으로 수행됩니다.

구문 도표에서 사용된 것처럼 로케이터 변수라는 용어는 로케이터 변수에 대한 참조를 표시합니다. 메타 변수인 *locator-variable*은 *host-variable*과 동일한 *host-identifier*를 포함하도록 확장될 수 있습니다.

다른 모든 호스트 변수와 마찬가지로 대형 오브젝트 로케이터 변수에는 연관된 표시기 변수가 있을 수 있습니다. 대형 오브젝트 로케이터 호스트 변수에 대한 표시기 변수는 다른 데이터 유형에 대한 표시기 변수와 같은 방법으로 작동합니다. 널(NULL)이 데이터베이스에서 리턴되면 표시기 변수가 설정되고 로케이터 호스트 변수는 변경되지 않습니다. 이것은 로케이터가 결코 널(NULL) 값으로 지정되지 않음을 의미합니다.

값을 나타내지 않는 로케이터 변수가 참조되는 경우, 오류가 발생합니다(SQLSTATE 0F001).

트랜잭션 커미트 또는 트랜잭션 종료시 해당 트랜잭션에서 확보한 모든 로케이터는 해제됩니다.

BLOB, CLOB 및 DBCLOB 파일 참조 변수에 대한 참조

BLOB, CLOB 및 DBCLOB 파일 참조 변수는 LOB에 대한 직접 파일 입력 및 출력에 사용되고, 모든 호스트 언어로 정의될 수 있습니다. 이것은 원시(native) 데이터 유형이 아니므로, SQL 확장자가 사용되며 프리컴파일러는 각 변수를 나타내는 데 필요한 호스트 언어 구문을 생성합니다. REXX의 경우, LOB는 문자열에 맵핑됩니다.

파일 참조 변수는 LOB 로케이터가 LOB 바이트를 포함하지 않고 나타내는 것처럼 파일을 포함하지 않고 나타냅니다. 데이터베이스 쿼리, 갱신 및 삽입은 파일 참조 변수를 사용하여 하나의 컬럼 값을 저장하거나 검색할 수 있습니다.

파일 참조 변수에는 다음과 같은 등록 정보가 있습니다.

데이터 유형

BLOB, CLOB 또는 DBCLOB. 이 등록 정보는 변수가 선언될 때 지정됩니다.

방향 런타임시 (파일 옵션 값의 일부로) 응용프로그램에서 지정해야 합니다. 방향은 다음 중 하나입니다.

- 입력(EXECUTE문, OPEN문, UPDATE문, INSERT문 또는 DELETE문에서 데이터의 소스로 사용됨)
- 출력(FETCH문이나 SELECT INTO문에서 데이터의 목표로 사용됨)

파일 이름

런타임시 응용프로그램에서 지정해야 합니다. 파일 이름은 다음 중 하나입니다.

- 파일의 전체 경로 이름(권장사항)
- 상대 파일 이름. 상대 파일 이름이 제공되는 경우, 이것은 클라이언트 프로세스의 현재 경로에 추가됩니다.

응용프로그램 내에서는 한 개의 파일 참조 변수에서 한 개의 파일이 참조되어야 합니다.

파일 이름 길이

런타임시 응용프로그램에서 지정해야 합니다. 파일 이름의 길이입니다(바이트).

File Options

응용프로그램은 해당 변수를 사용하기 전에 파일 참조 변수에 여러 옵션 중 하나를 지정해야 합니다. 옵션은 파일 참조 변수 구조 내의 필드에서 INTEGER 값으로 설정됩니다. 각 파일 참조 변수에 대해 다음 값 중 하나를 지정해야 합니다.

- 입력(클라이언트에서 서버로)

  SQL_FILE_READ

  열어서 읽은 다음 닫을 수 있는 일반 파일입니다. (옵션은 COBOL의 경우 SQL-FILE-READ이며, FORTRAN의 경우 sql_file_read, REXX의 경우 READ입니다.)

- 출력(서버에서 클라이언트로)

  SQL_FILE_CREATE

  새 파일을 작성합니다. 파일이 이미 존재하는 경우, 오류가 리턴됩니다. (옵션은 COBOL의 경우 SQL-FILE-CREATE, FORTRAN의 경우 sql_file_create, REXX의 경우 CREATE입니다.)

  SQL_FILE_OVERWRITE(겹쳐쓰기)

  지정된 이름의 기존 파일이 존재하는 경우에는 이 파일이 대체되고, 존재하지 않는 경우에는 새 파일이 작성됩니다. (옵션은 COBOL의

경우 SQL-FILE-OVERWRITE, FORTRAN의 경우 sql_file_overwrite, REXX의 경우 OVERWRITE입니다.)

SQL_FILE_APPEND
지정된 이름의 기존 파일이 존재하는 경우에는 출력이 추가되며, 존재하지 않는 경우에는 새 파일이 작성됩니다. (옵션은 COBOL의 경우 SQL-FILE-APPEND, FORTRAN의 경우 sql_file_append, REXX의 경우 APPEND입니다.)

데이터 길이
출력 시에는 사용되지 않습니다. 출력 시 구현은 데이터 길이를 파일에 기록된 새 데이터의 길이로 설정합니다. 길이는 바이트입니다.

다른 모든 호스트 변수와 마찬가지로 파일 참조 변수에는 연관된 표시기 변수가 있을 수 있습니다.

출력 파일 참조 변수 예(C의 경우)

선언 섹션이 다음과 같이 코딩될 경우

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS CLOB_FILE  hv_text_file;
char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

사전 처리 결과는 다음과 같습니다.

```
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
```

그런 다음, 다음 코드를 사용하여 데이터베이스의 CLOB 컬럼에서 선택하여 hv_text_file에서 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(C의 경우)

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);
```

구조화된 유형 호스트 변수에 대한 참조

구조화된 유형 변수는 FORTRAN, REXX 및 Java를 제외한 모든 호스트 언어로 정의될 수 있습니다. 이것은 원시(native) 데이터 유형이 아니므로, SQL 확장자가 사용되며 프리컴파일러는 각 변수를 나타내는 데 필요한 호스트 언어 구문을 생성합니다.

다른 모든 호스트 변수와 마찬가지로 구조화된 유형 변수에는 연관된 표시기 변수가 있을 수 있습니다. 구조화된 유형 호스트 변수에 대한 표시기 변수는 다른 데이터 유형에 대한 표시기 변수와 같은 방법으로 작동합니다. 데이터베이스에서 널(NULL)이 리턴되면 표시기 변수가 설정되고 구조화된 유형 호스트 변수는 변경되지 않습니다.

구조화된 유형에 대한 실제 호스트 변수는 내장 데이터 유형으로 정의됩니다. 구조화된 유형과 연관되는 내장 데이터 유형은 다음과 같이 지정할 수 있어야 합니다.

- 프리컴파일 명령에 지정된 TRANSFORM GROUP 옵션에 의해 정의된 구조화된 유형에 대한 FROM SQL 변환 함수의 결과에서
- 프리컴파일 명령에 지정된 TRANSFORM GROUP 옵션에 의해 정의된 구조화된 유형에 대한 TO SQL 변환 함수의 매개변수로

호스트 변수 대신 매개변수 표시문자를 사용할 경우, 적절한 매개변수 유형 등록 정보를 SQLDA에 지정해야 합니다. 이 때 SQLDA에서 SQLVAR 구조의 ″doubled″ 세트와, 2차 SQLVAR의 SQLDATATYPE_NAME 필드는 구조화된 유형의 스키마와 유형 이름으로 채워져야 합니다. SQLDA 구조에서 스키마를 생략하면 오류가 발생합니다(SQLSTATE 07002).

예 :

C 프로그램에서 내장 유형 BLOB(1048576)을 사용하여 POLYGON 유형의 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;
```

SQL 경로

SQL 경로는 순서화된 스키마 이름 목록입니다. 데이터베이스 관리 프로그램은 SQL 경로를 사용하여 CREATE, DROP, COMMENT, GRANT 또는 REVOKE 문의 기본

오브젝트로서가 아니라 다른 컨텍스트에 나타나는 규정되지 않은 데이터 유형 이름(내장 유형 및 구별 유형 모두), 전역 변수 이름, 모듈 이름, 함수 이름 및 프로시저 이름에 대한 스키마 이름을 분석합니다. 세부사항은 『규정되지 않은 오브젝트 이름의 규정』을 참조하십시오.

예를 들어, SQL 경로가 SYSIBM인 경우입니다. SYSFUN, SYSPROC, SYSIBMADM, SMITH, XGRAPHICS2 및 규정되지 않은 구별 유형 이름 MYTYPE이 지정되면 데이터베이스 관리 프로그램은 스키마 SYSIBM에서 첫 번째로, 그 다음은 SYSFUN, SYSPROC, SYSIBMADM, SMITH, XGRAPHICS2 순으로 MYTYPE을 찾습니다.

사용되는 SQL 경로는 SQL문에 따라 다릅니다.

- 정적 SQL문(CALL 변수 명령문 제외)의 경우, 사용되는 SQL 경로는 포함하는 패키지, 프로시저, 함수, 트리거 또는 뷰가 작성될 때 지정한 SQL 경로입니다.
- 동적 SQL문의 경우(CALL 변수 명령문의 경우에도), SQL 경로는 CURRENT PATH 특수 레지스터의 값입니다. CURRENT PATH는 SET PATH 문으로 설정할 수 있습니다.

SQL 경로가 명시적으로 지정되지 않은 경우, SQL 경로는 뒤에 명령문의 권한 부여 ID가 있는 시스템 경로입니다.

규정되지 않은 오브젝트 이름의 규정

규정되지 않은 오브젝트 이름은 내재적으로 규정됩니다. 이름 규정 규칙은 이름이 식별하는 오브젝트의 유형에 따라 다릅니다.

규정되지 않은 별명, 인덱스, 패키지 시퀀스, 테이블, 트리거 및 뷰 이름

규정되지 않은 별명, 인덱스, 패키지, 시퀀스, 테이블, 트리거 및 뷰 이름은 디폴트 스키마에 의해 내재적으로 규정됩니다.

정적 SQL문의 경우, 디폴트 스키마는 포함하는 함수, 패키지, 프로시저 또는 트리거가 작성될 때 지정한 디폴트 스키마입니다.

동적 SQL문의 경우, 디폴트 스키마는 응용프로그램 프로세스에 지정된 디폴트 스키마입니다. 디폴트 스키마는 SET SCHEMA 문을 사용하여 응용프로그램 프로세스에 대해 지정할 수 있습니다. 디폴트 스키마를 명시적으로 지정하지 않은 경우, 디폴트 스키마는 명령문의 권한 부여 ID입니다.

규정되지 않은 사용자 정의 유형, 함수, 프로시저, 특정, 전역 변수 및 모듈 이름

데이터 유형(내장 유형 및 구별 유형 모두), 전역 변수, 모듈, 함수, 프로시저 및 특정 이름의 규정은 규정되지 않은 이름이 나타나는 SQL문에 따라 다릅니다.

- 규정되지 않은 이름이 CREATE, ALTER, COMMENT, DROP, GRANT 또는 REVOKE 문의 기본 오브젝트인 경우, 이름은 규정되지 않은 테이블 이름을 규정하는 것과 같은 규칙을 사용하여 내재적으로 규정됩니다("규정되지 않은 별명, 인덱스, 패키지, 시퀀스, 테이블, 트리거 및 뷰 이름" 참조).ALTER MODULE 명령문의 ADD, COMMENT, DROP 또는 PUBLISH 조작의 기본 오브젝트는 규정자 없이 지정되어야 합니다.
- 참조 컨텍스트가 모듈 안에 있는 경우, 데이터베이스 관리 프로그램은 일치를 찾기 위한 적절한 오브젝트 유형 분석을 적용하여 오브젝트의 모듈을 검색합니다. 일치가 발견되지 않으면 다음 글머리표에 지정된 대로 계속 검색합니다.
- 그렇지 않으면, 내재된 스키마 이름은 다음과 같이 판별됩니다.
  - 구별 유형 이름의 경우, 데이터베이스 관리 프로그램은 SQL 경로를 검색하고 SQL 경로에서 스키마에 데이터 유형이 존재하는 첫 번째 스키마를 선택합니다.
  - 전역 변수의 경우, 데이터베이스 관리 프로그램은 SQL 경로를 검색하고 SQL 경로에서 스키마에 전역 변수가 존재하는 첫 번째 스키마를 선택합니다.
  - 프로시저 이름의 경우, 데이터베이스 관리 프로그램은 프로시저 분석과 함께 SQL 경로를 사용합니다.
  - 함수 이름의 경우, 데이터베이스 관리 프로그램은 함수 결정과 함께 SQL 경로를 사용합니다.
  - 전래 함수에 지정된 특정 이름의 경우 "CREATE FUNCTION(전래)"을 참조하십시오.

규정되지 않은 오브젝트 이름 분석

모듈에 정의되고 모듈 외부에서 사용 가능한 오브젝트는 모듈 이름으로 규정해야 합니다. 모듈은 내재적으로 규정될 수도 있는 스키마 오브젝트이므로, 발행된 모듈 오브젝트는 규정되지 않은 모듈 이름이나 스키마 규정 모듈 이름이 될 수 있습니다. 규정되지 않은 모듈 이름을 사용하는 경우, 모듈 오브젝트에 대한 참조는 모듈의 일부가 아닌 스키마 규정 오브젝트와 동일하게 나타납니다. 복합 SQL문과 같은 특정 범위 내에서, 두 파트 ID는 다음이 될 수도 있습니다.

- 테이블 이름으로 규정된 컬럼 이름
- 변수 이름으로 규정된 행 필드 이름
- 레이블로 규정된 변수 이름
- 루틴 이름으로 규정된 루틴 매개변수 이름

이 오브젝트는 스키마 오브젝트나 모듈 오브젝트를 고려하기 전에 해당 범위 내에서 분석됩니다. 다음 프로세스는 스키마 오브젝트나 모듈 오브젝트가 될 수 있는 두 파트 ID로 오브젝트를 분석하는 데 사용됩니다.

- 참조 컨텍스트가 모듈에 있고 규정자가 모듈 이름과 일치하는 경우, 데이터베이스 관리 프로그램은 발행된 모듈 오브젝트와 발행되지 않은 모듈 오브젝트 사이에 일치를

찾기 위한 적절한 오브젝트 유형 분석을 적용하여 오브젝트의 모듈을 검색합니다. 일치가 발견되지 않으면 다음 글머리표에 지정된 대로 계속 검색합니다.

- 규정자가 스키마 이름인 것으로 가정하고, 스키마가 존재하면 스키마에서 오브젝트를 분석합니다.
- 규정자가 기존 스키마가 아니거나 스키마에서 규정자와 일치하는 오브젝트가 발견되지 않고 규정자가 컨텍스트 모듈 이름과 일치하지 않는 경우, SQL 경로에서 스키마의 규정자와 일치하는 첫 번째 모듈을 검색합니다. 일치하는 모듈에 대해 권한이 부여된 경우 발행된 모듈 오브젝트만 고려하여 해당 모듈에서 오브젝트를 분석합니다.
- SQL 경로에서 모듈로 규정자가 발견되지 않고 규정자가 컨텍스트 모듈 이름과 일치하지 않는 경우, 규정자와 일치하는 모듈 공용 동의어가 있는지 점검합니다. 발견되면, 발행된 모듈 오브젝트만 고려하여 모듈 공용 동의어로 식별된 모듈에서 오브젝트를 분석합니다.

데이터 유형

SQL에서 조작할 수 있는 가장 작은 단위의 데이터는 *값*입니다. 값은 해당 소스의 데이터 유형에 따라 해석됩니다. 소스에는 다음이 포함됩니다.

- 상수
- 컬럼
- 함수
- 표현식
- 특수 레지스터
- 변수(예: 호스트 변수, SQL 변수, 전역 변수, 매개변수 표시문자, 모듈 변수 및 루틴의 매개변수)
- 부울 값

DB2는 다수의 내장 데이터 유형을 지원합니다. 사용자 정의 데이터 유형도 지원합니다. 94 페이지의 그림 10은 지원되는 내장 데이터 유형을 나타냅니다.

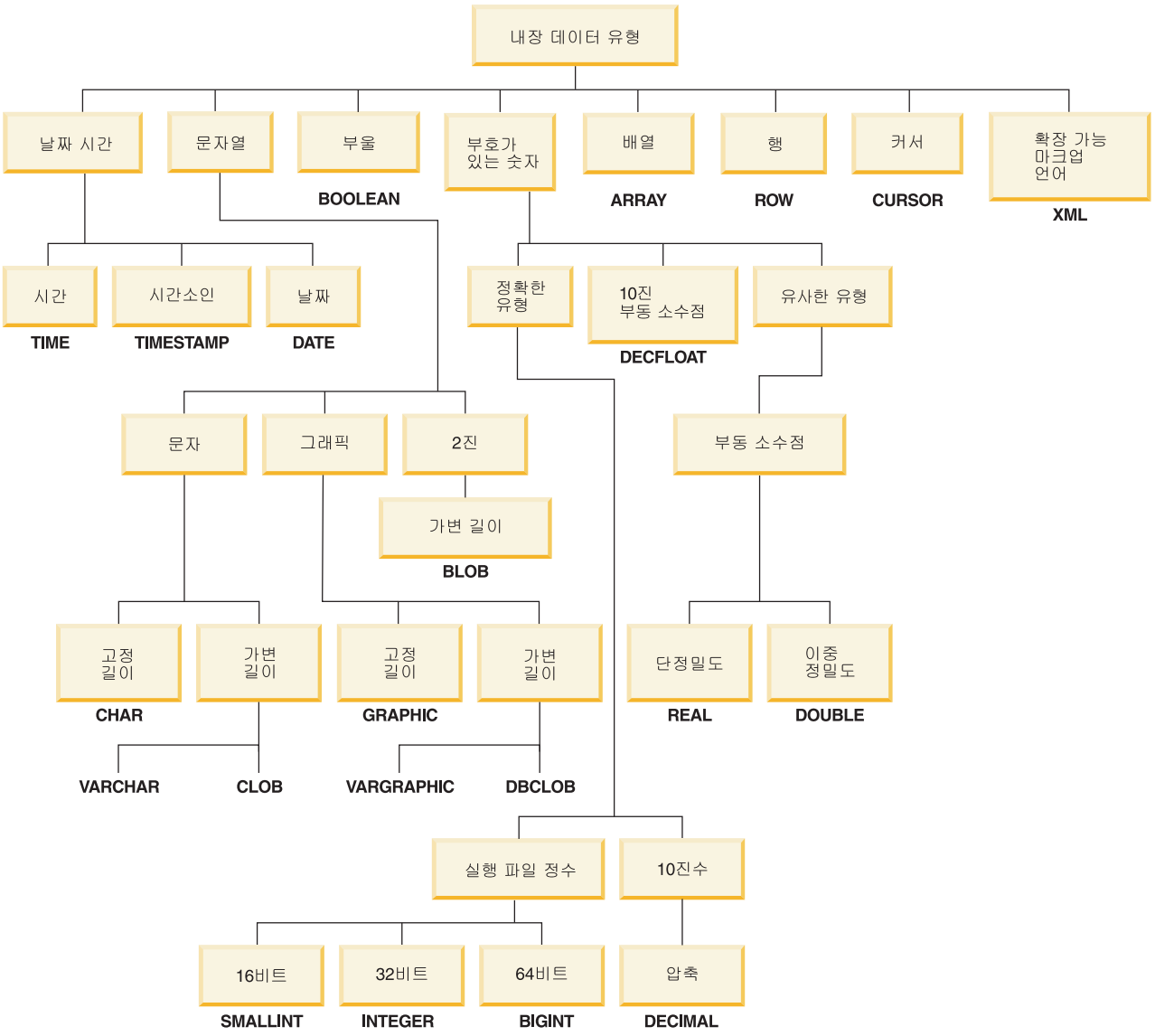


그림 10. DB2 내장 데이터 유형

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

데이터 유형 목록

숫자

숫자 데이터 유형은 정수, 10진수, 부동 소수점 및 10진 부동 소수점입니다.

숫자 데이터 유형은 다음과 같이 구분됩니다.

- 완전 숫자: 정수 및 10진수
- 10진 부동 소수점
- 근사 숫자: 부동 소수점

정수에는 작은 정수, 대형 정수 및 큰 정수가 있습니다. 정수(integer)의 정확한 표현은 정수(integer number)입니다. 10진수는 고정 정밀도와 스케일을 가진 숫자의 정확한 표현입니다. 정수와 10진수는 완전 숫자 유형으로 간주됩니다.

10진 부동 소수점 숫자의 정밀도는 16 또는 34입니다. 10진 부동 소수점은 실수의 정확한 표현과 실수의 근사값을 둘 다 지원하므로 정확한 숫자 유형 또는 근사 숫자 유형으로 간주되지 않습니다.

부동 소수점에는 단정밀도와 배정밀도가 있습니다. 부동 소수점 숫자는 실수의 근사값이며 근사 숫자 유형으로 간주됩니다.

모든 숫자에는 *기호*, *정밀도* 및 *스케일*이 있습니다. 10진 부동 소수점을 제외한 모든 숫자의 경우, 컬럼 값이 영(0)이면 기호가 양수입니다. 10진 부동 소수점 숫자에는 음수 영(0)과 양수 영(0)이 있습니다. 10진 부동 소수점에는 숫자 구별 값이 있으며 여러 지수를 가진 동일한 숫자가 있습니다(예: 0.0, 0.00, 0.0E5, 1.0, 1.00, 1.0000). 정밀도는 기호를 제외한 총 소수 자리수입니다. 스케일은 소수점 오른쪽의 총 소수 자리수입니다. 소수점이 없으면, 스케일은 영(0)입니다.

CREATE TABLE문 설명의 데이터 유형 절을 참조하십시오.

작은 정수(SMALLINT)

*SMALLINT*는 정밀도가 5자리인 2바이트 정수입니다. 작은 정수 범위는 -32 768에서 32 767까지입니다.

큰 정수(INTEGER)

*INTEGER*는 정밀도가 10자리인 4바이트 정수입니다. 대형 정수 범위는 -2 147 483 648에서 +2 147 483 647까지입니다.

큰 정수(BIGINT)

*BIGINT*는 정밀도가 19자리인 8바이트의 정수입니다. 큰 정수 범위는 -9 223 372 036 854 775 808에서 +9 223 372 036 854 775 807까지입니다.

10진수(DECIMAL 또는 NUMERIC)

10진수 값은 내재된 소수점이 있는 압축 10진수 값입니다. 소수점의 위치는 정밀도와 스케일에 따라 결정됩니다. 숫자의 소수점 이하 자릿수인 스케일은 음수이거나 정밀도보다 클 수 없습니다. 최대 정밀도는 31자리입니다.

10진수 컬럼에 있는 모든 값의 정밀도와 스케일은 동일합니다. 10진수 컬럼의 소수 변수나 숫자의 범위는 -*n*에서 +*n*입니다. 여기서, *n*의 절대값은 적용 가능한 정밀도 및 스케일로 표시될 수 있는 가장 큰 수입니다. 최대 범위는 $-10^{31}+1$에서 $10^{31}-1$까지입니다.

단정밀도 부동 소수점(REAL)

단정밀도 부동 소수점 숫자는 대략 32비트의 실수입니다. 이 숫자는 0이거나 -3.4028234663852886e+38 - -1.1754943508222875e-38 또는 1.1754943508222875e-38 - 3.4028234663852886e+38 사이의 범위입니다.

배정밀도 부동 소수점(DOUBLE 또는 FLOAT)

배정밀도 부동 소수점 숫자는 대략 64비트의 실수입니다. 숫자는 0이거나 -1.7976931348623158e+308 - -2.2250738585072014e-308 또는 2.2250738585072014e-308 - 1.7976931348623158e+308 사이의 범위입니다.

10진 부동 소수점(DECFLOAT)

10진 부동 소수점 값은 소수점이 있는 IEEE 754r 숫자입니다. 소수점 위치는 각 10진 부동 소수점 값에 저장됩니다. 최대 정밀도는 34자리입니다. 10진 부동 소수점 숫자의 범위는 16 또는 34자리 정밀도이며, 지수 범위는 각각 10^{-383}에서 10^{+384} 또는 10^{-6143}에서 10^{+6144}입니다. DECFLOAT 값의 최소 지수(E_{min})는 DECFLOAT(16)의 경우 -383이고 DECFLOAT(34)의 경우 -6143입니다. DECFLOAT 값의 최대 지수(E_{max})는 DECFLOAT(16)의 경우 384이고 DECFLOAT(34)의 경우 6144입니다.

유한수 외에 10진 부동 소수점 숫자는 이름 지정된 다음 10진 부동 소수점 특수 값 중 하나를 나타낼 수 있습니다.

- Infinity - *크기*가 무한히 큰 숫자를 나타내는 값
- Quiet NaN - 정의되지 않은 결과를 나타내고 유효하지 않은 숫자 경고를 초래하지 않는 값
- Signalling NaN - 숫자 조작에서 사용되는 경우에 정의되지 않은 결과를 나타내고 유효하지 않은 숫자 경고를 초래하지 않는 값

숫자가 이러한 특수 값 중 하나를 가지면, 계수 및 지수가 정의되지 않습니다. 양수 또는 음수 infinity를 가질 수 있기 때문에 infinity 값을 나타내는 기호는 중요합니다. NaN 값을 나타내는 기호는 산술 연산에 대한 의미가 없습니다.

기형 숫자 및 언더플로우

조정된 지수가 E_{min}보다 작은 0이 아닌 숫자를 기형 숫자라고 합니다. 이러한 기형 숫자는 모든 연산에 피연산자로 허용되며 모든 연산에서 결과를 가져올 수 있습니다.

기형 결과의 경우, 지수의 최소 값은 E_{tiny}라는 E_{min} - (precision-1)입니다. 여기서, precision은 작업 중인 정밀도입니다. 필요하면 지수가 E_{tiny}보다 작지 않도록 결과가 반올림됩니다. 반올림 중에 결과가 부정확하게 되면 언더플로우 경고가 리턴됩니다. 기형 결과가 항상 언더플로우 경고를 리턴하지는 않습니다.

계산 중에 숫자가 영(0)으로 언더플로우되면, 지수는 E_{tiny}입니다. 지수 최대값은 영향을 받지 않습니다.

기형 숫자의 경우 지수 최대값은 기형 숫자를 초래하지 않는 연산 중에 발생할 수 있는 지수 최소값과 동일합니다. 소수 자리수의 계수 길이가 정밀도와 동일하면 이런 상황이 발생합니다.

문자열

*문자열*은 일련의 바이트입니다. 문자열의 길이는 문자열의 바이트 수입니다. 길이가 0인 경우, 이 값을 *빈 문자열*이라고 합니다. 이 값을 널(NULL) 값과 혼동해서는 안됩니다.

고정 길이 문자열(CHAR)

고정 길이 문자열 컬럼에 있는 모든 값의 길이는 동일하며, 컬럼의 길이 속성에 따라 결정됩니다. 길이 속성의 범위는 1 - 254이어야 합니다.

가변 길이 문자열

가변 길이 문자열 유형은 다음 두 가지입니다.

- VARCHAR 값의 길이는 최대 32 672바이트일 수 있습니다.
- 문자 대형 오브젝트(CLOB) 값의 길이는 최대 2GB 빼기 1바이트(2 147 483 647 바이트)까지 가능합니다. CLOB은 대형 SBCS 또는 혼합(SBCS 및 MBCS) 문자 기반 데이터(예: 단일 문자 세트로 기록된 문서)를 저장하는 데 사용되므로, 이와 연관된 SBCS 또는 혼합 코드 페이지가 있습니다.

CLOB 데이터 유형을 초래하는 표현식과 구조화된 유형 컬럼에는 특별한 제한사항이 적용됩니다. 이러한 표현식 및 컬럼은 다음에서 사용할 수 없습니다.

- DISTINCT절이 앞에 오는 SELECT 목록
- GROUP BY절
- ORDER BY절
- UNION ALL을 제외한 집합 연산자의 subselect
- 규정된 기본 BETWEEN 또는 IN 술어
- 집계 함수
- VARGRAPHIC, TRANSLATE 및 datetime 스칼라 함수
- LIKE 술어에 있는 패턴 피연산자나 POSSTR 함수에 있는 검색 문자열 피연산자
- 날짜 시간 값의 문자열 표현

VARCHAR을 인수로 취하는 SYSFUN 스키마의 함수는 길이가 4 000바이트를 넘는 VARCHAR을 인수로 승인하지 않습니다. 그러나 이들 함수 중 대부분이 또한 CLOB(1M)을 받아들이는 대체 시그니처를 가지고 있습니다. 이러한 함수의 경우, 사용자는 4 000개를 넘는 VARCHAR 문자열을 명시적으로 CLOB로 캐스트한 후 결과를 원하는 길이의 VARCHAR로 다시 캐스트할 수 있습니다.

C에서의 NUL 종료 문자열은 프리컴파일 옵션의 표준 레벨에 따라 다르게 처리됩니다.

각 문자열은 다음 중 하나로 정의됩니다.

비트 데이터

코드 페이지와 연관되지 않은 데이터

1바이트 문자 세트(SBCS) 데이터

모든 문자가 1바이트로 표시되는 데이터

혼합 데이터

1바이트 문자 세트(SBCS)와 복수 바이트 문자 세트(MBCS)의 문자가 혼합될 수 있는 데이터

주: LONG VARCHAR 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다. 추후 릴리스에서는 제거될 수 있습니다.

내장 함수의 문자열 단위

특정 내장 함수에 대한 문자열 단위 지정 기능을 사용하여 "바이트 기반 방식"이 아닌 "문자 기반 방식"으로 문자열 데이터를 처리할 수 있습니다. *문자열 단위*는 조작이 발생하는 길이를 판별합니다. 조작에 대한 문자열 단위로 CODEUNITS16, CODEUNITS32 또는 OCTETS를 지정할 수 있습니다.

CODEUNITS16

Unicode UTF-16이 조작의 단위임을 지정합니다. CODEUNITS16은 너비가 2바이트인 코드 단위로 응용프로그램이 데이터를 처리할 때 유용합니다. *첨부 문자*로 알려진 일부 문자는 두 개의 UTF-16 코드 단위를 인코딩해야 합니다. 예를 들어, 악상 기호인 G 음자리표는 두 개의 UTF-16 코드 단위(UTF-16BE에서 X'D834' 및 X'DD1E')를 필요로 합니다.

CODEUNITS32

Unicode UTF-32가 조작의 단위임을 지정합니다. CODEUNITS32는 단순한 고정 길이 형식의 데이터를 처리하며 데이터의 저장 형식(ASCII, UTF-8 또는 UTF-16)에 관계없이 동일한 응답을 리턴해야 하는 응용프로그램에 유용합니다.

OCTETS

바이트가 조작의 단위임을 지정합니다. OCTETS는 종종 응용프로그램에서 버퍼 공간 할당이 중요하거나 조작에 단순 바이트 처리가 필요할 때 사용됩니다.

OCTETS(바이트)를 사용하여 계산된 문자열의 계산된 길이가 CODEUNITS16 또는 CODEUNITS32를 사용하여 계산된 것과 다를 수도 있습니다. OCTETS를 사용할 경우, 문자열의 길이는 단순히 문자열에서 바이트 수를 계산하여 판별합니다. CODEUNITS16 또는 CODEUNITS32를 사용할 경우, 문자열의 길이는 UTF-16 또는 UTF-32에서 각각 문자열을 표시하는 데 필요한 16비트 또는 32비트 코드 단위의 수를 계산하여 판별합니다. CODEUNITS16 및 CODEUNITS32를 사용하여 판별된 길이는 데이터가 첨부 문자를 포함하지 않을 경우 같습니다(100 페이지의 『CODEUNITS16 및 CODEUNITS32 간 차이』 참조).

예를 들어, Unicode UTF-8로 인코딩된 NAME, VARCHAR(128)에 'Jürgen'이라는 값이 들어 있다고 가정하십시오. CODEUNITS16 및 CODEUNITS32의 문자열 길이를 각각 계산하는 다음 두 쿼리는 동일한 값(6)을 리턴합니다.

```
SELECT CHARACTER_LENGTH(NAME,CODEUNITS16) FROM T1
  WHERE NAME = 'Jürgen'

SELECT CHARACTER_LENGTH(NAME,CODEUNITS32) FROM T1
  WHERE NAME = 'Jürgen'
```

OCTETS의 문자열 길이를 계산하는 다음 쿼리는 값 7을 리턴합니다.

```
SELECT CHARACTER_LENGTH(NAME,OCTETS) FROM T1
  WHERE NAME = 'Jürgen'
```

이들 값은 지정된 문자열 단위로 표시되는 문자열의 길이를 나타냅니다.

다음 테이블은 'Jürgen'이라는 이름의 UTF-8, UTF-16BE(빅 엔디안(big-endian)) 및 UTF-32BE 표시를 나타냅니다.

```
형식      'Jürgen' 이름의 표시
--------  ------------------------------------
UTF-8   X'4AC3BC7267656E'
UTF-16BE  X'004A00FC007200670065006E'
UTF-32BE  X'0000004A000000FC00000072000000670000006500000006E'
```

'ü' 문자의 표시가 다음 세 가지 문자열 단위마다 다릅니다.

- 'ü' 문자의 UTF-8 표시는 X'C3BC'입니다.
- 'ü' 문자의 UTF-16BE 표시는 X'00FC'입니다.
- 'ü' 문자의 UTF-32BE 표시는 X'000000FC'입니다.

내장 함수의 문자열 단위 지정은 데이터 유형 또는 함수 결과의 코드 페이지에 영향을 미치지 않습니다. 필요한 경우, DB2는 CODEUNITS16 또는 CODEUNITS32를 지정시 평가를 위해 데이터를 Unicode로 변환합니다.

LOCATE 또는 POSITION 함수에 OCTETS를 지정하고 문자열 인수의 코드 페이지가 서로 다를 경우, DB2는 데이터를 *source-string* 인수의 코드 페이지로 변환합니다. 이 경우, 함수의 결과가 *source-string* 인수의 코드 페이지에 속합니다. 단일 문자열 인수를 취하는 함수에 대해 OCTETS를 지정할 경우, 데이터가 문자열 인수의 코드 페이지에서 평가되며 함수의 결과가 문자열 인수의 코드 페이지에 속합니다.

CODEUNITS16 및 CODEUNITS32 간 차이

CODEUNITS16 또는 CODEUNITS32를 지정할 경우, 결과는 데이터가 Unicode 첨부 문자를 포함할 경우를 제외하고 동일합니다. 이는 첨부 문자가 두 개의 UTF-16 코드 단위 또는 하나의 UTF-32 코드 단위로 표시되기 때문입니다. UTF-8에서는 첨부 문자가 아닐 경우 1 - 3바이트로 표시되며, 첨부 문자는 4바이트로 표시됩니다. UTF-16에서는 첨부 문자가 아닐 경우 하나의 CODEUNITS16 코드 단위 또는 2바이트로 표

시되며, 첨부 문자는 두 개의 CODEUNITS16 코드 단위 또는 4바이트로 표시됩니다. UTF-32에서 문자는 하나의 CODEUNITS32 코드 단위 또는 4바이트로 표시됩니다.

예를 들어, 다음 표는 산술용 굵은 대문자 A와 라틴어 대문자 A를 나타냅니다. 산술용 굵은 대문자 A는 UTF-8, UTF-16 및 UTF-32에서 4바이트로 표시되는 첨부 문자입니다.

문자	**UTF-8** 표시	**UTF-16BE** 표시	**UTF-32BE** 표시
유니코드 값 X'1D400' - 'A'(산술용 굵은 대문자 A)	X'F09D9080'	X'D835DC00'	X'0001D400'
유니코드 값 X'0041' - 'A'(라틴어 대문자 A)	X'41'	X'0041'	X'00000041'

C1이 유니코드 UTF-8로 인코딩된 VARCHAR(128) 컬럼이고, T1 테이블이 산술용 굵은 대문자 A 값(X'F09D9080')을 갖는 하나의 행을 포함한다고 가정하십시오. 다음 쿼리는 서로 다른 결과를 리턴합니다.

```
쿼리                                                    리턴값
-----                                                   -------
SELECT CHARACTER_LENGTH(C1,CODEUNITS16) FROM T1         2

SELECT CHARACTER_LENGTH(C1,CODEUNITS32) FROM T1         1

SELECT CHARACTER_LENGTH(C1,OCTETS) FROM T1              4
```

그래픽 문자열

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

그래픽 문자열 값에 2진 문자 코드 포인트만 들어 있는지 점검되지 않습니다. 이 규칙에 대한 예외는 WCHARTYPE CONVERT 옵션으로 프리컴파일된 응용프로그램입니다. 이 경우에는 유효성이 확인됩니다. 오히려 데이터베이스 관리 프로그램에서는 2바이트 문자 데이터가 그래픽 데이터 필드에 포함되어 있는 것으로 간주하고 그래픽 문자열 값의 길이가 짝수인지 *점검합니다*.

C에서의 NUL 종료 그래픽 문자열은 프리컴파일 옵션의 표준 레벨에 따라 다르게 처리됩니다. 이 데이터 유형은 테이블에 작성될 수 없고, 데이터베이스에 데이터를 입력하거나 데이터베이스에서 데이터를 검색하는 데만 사용할 수 있습니다.

고정 길이 그래픽 문자열(GRAPHIC)

고정 길이 그래픽 문자열 컬럼에 있는 모든 값의 길이는 동일하며, 컬럼의 길이 속성에 따라 결정됩니다. 길이 속성의 범위는 1 - 127이어야 합니다.

가변 길이 그래픽 문자열

가변 길이 그래픽 문자열 유형은 다음 두 가지입니다.

- VARGRAPHIC 값의 길이는 최대 16 336 2바이트 문자일 수 있습니다.
- 2바이트 문자 대형 오브젝트(DBCLOB) 값의 길이는 최대 1,073,741,823 2바이트 문자일 수 있습니다. DBCLOB는 대형 DBCS 문자 기반 데이터(예: 단일 문자 세트로 기록된 문서)를 저장하는 데 사용되므로, 이와 연관된 DBCS 코드 페이지가 있습니다.

최대 길이가 127바이트를 초과하는 가변 길이 그래픽 문자열인 표현식에 특별한 제한사항이 적용됩니다. 이러한 제한사항은 98 페이지의 『가변 길이 문자열』에 지정된 것과 같습니다.

주: LONG VARGRAPHIC 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다. 추후 릴리스에서는 제거될 수 있습니다.

실행 파일 문자열

*실행 파일 문자열*은 일련의 바이트입니다. 보통 텍스트 데이터가 들어 있는 문자열과는 달리, 실행 파일 문자열은 그림, 음성 또는 혼합 미디어와 같이 일반적이지 않은 데이터를 보유하는 데 사용됩니다. FOR BIT DATA 부속 유형의 문자열을 유사한 용도로 사용할 수 있습니다. 실행 파일 문자열은 코드 페이지와 연관되지 않습니다. 실행 파일 문자열에는 문자열과 동일한 제한사항이 있습니다(세부사항은 98 페이지의 『가변 길이 문자열』 참조). FOR BIT DATA 부속 유형의 문자열만 실행 파일 문자열과 호환 가능합니다.

실행 파일 대형 오브젝트(BLOB)

*실행 파일 대형 오브젝트(BLOB)*는 길이가 최대 2GB 빼기 1바이트(2 147 483 647 바이트)까지 가능한 가변 길이 실행 파일 문자열입니다. BLOB은 사용자 정의 유형 및 사용자 정의 함수(UDF)에서 사용할 구조화된 데이터를 보유할 수 있습니다. FOR BIT DATA 문자열과 마찬가지로 BLOB 문자열은 코드 페이지와 연관되지 않습니다.

대형 오브젝트(LOB)

*대형 오브젝트(LOB)*라는 용어와 일반적인 두문자어 LOB은 BLOB, CLOB 또는 DBCLOB 데이터 유형을 지칭합니다. LOB 값은 98 페이지의 『가변 길이 문자열』에 설명된 대로 제한사항을 따릅니다. 이러한 제한사항은 LOB 문자열의 길이 속성이 254 바이트 이하인 경우에도 적용됩니다.

LOB 값은 매우 클 수 있으므로, 이러한 값을 데이터베이스 서버에서 클라이언트 응용프로그램 호스트 변수로 전송하는 작업에는 시간이 소요될 수 있습니다. 응용프로그램에서는 일반적으로 한 번에 전체가 아닌 하나의 LOB 값을 처리하므로, 대형 오브젝트(LOB) 로케이터를 사용하여 LOB 값을 참조할 수 있습니다.

대형 오브젝트(LOB) 로케이터 또는 LOB 로케이터는 값이 데이터베이스 서버에서 단일 LOB 값을 나타내는 호스트 변수입니다.

응용프로그램에서는 LOB 값을 LOB 로케이터로 선택할 수 있습니다. 그런 다음 LOB 로케이터를 사용하여 로케이터 값을 입력으로 제공하여 응용프로그램은 LOB값에 대해 데이터베이스 조작을 요청할 수 있습니다(예: SUBSTR, CONCAT, VALUE 또는 LENGTH 스칼라 함수 적용, 지정 수행, LIKE나 POSSTR로 LOB 검색 또는 LOB에 대해 사용자 정의 함수(UDF) 적용). 결과 출력(클라이언트 호스트 변수에 지정된 데이터)은 일반적으로 입력 LOB 값의 작은 서브세트입니다.

LOB 로케이터는 디폴트값 이상을 나타낼 수 있으며, LOB 표현식과 연관된 값을 나타내기도 합니다. 예를 들어, LOB 로케이터는 다음과 연관된 값을 나타낼 수도 있습니다.

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

널(NULL)이 일반적인 호스트 변수로 선택되면 표시기 변수는 -1로 설정되어 그 값이 널(NULL)임을 나타냅니다. 그러나 LOB 로케이터의 경우, 표시기 변수의 의미는 약간 다릅니다. 로케이터 호스트 변수 자체가 널(NULL)이 될 수는 없으므로, 음의 표시기 변수 값은 LOB 로케이터가 나타내는 LOB 값이 널(NULL)임을 나타냅니다. 널(NULL) 정보는 표시기 변수 값에 의해 클라이언트에 대해서 로컬 정보로 보존되며, 서버는 유효한 로케이터로 널(NULL) 값을 추적하지 않습니다.

LOB 로케이터가 데이터베이스에 있는 행이나 위치가 아닌 값을 나타낸다는 것을 이해하는 것이 중요합니다. 일단 값이 로케이터로 선택되면 로케이터가 참조하는 값에 영향을 주는 원래의 행이나 테이블에 아무런 조작을 수행할 수 없습니다. 로케이터와 연관된 값은 트랜잭션이 종료되거나 로케이터가 명확히 해제될 때까지(어느 것이 먼저 일어나도 상관없음) 유효합니다. 로케이터는 이 함수를 제공하기 위해 데이터를 여분으로 복사하지는 않습니다. 대신, 로케이터 메커니즘은 기본 LOB 값에 대한 설명을 저장합니다. LOB 값(또는 위의 표시된 것과 같은 표현식)을 구체화하는 것은 이 값이 실제로

일부 위치에 지정될 때까지 지연되는데, 호스트 변수의 양식으로 사용자 버퍼에 지정되거나, 데이터베이스에서 또 다른 레코드에 지정됩니다.

LOB 로케이터는 트랜잭션 동안 LOB 값을 참조하는 데 사용되는 유일한 메커니즘으로, 작성된 트랜잭션의 범위를 벗어나지 않습니다. 또한 LOB 로케이터는 데이터베이스 유형이 아니며, 데이터베이스에 저장될 수 없으므로, 결과적으로 뷰나 점검 제한조건에 참여할 수 없습니다. 그러나 LOB 로케이터는 LOB 유형의 클라이언트 표현이므로, FETCH, OPEN 및 EXECUTE문에 사용되는 SQLDA 구조 내에서 설명될 수 있도록 LOB 로케이터에 대한 SQLTYPE이 있습니다.

날짜 시간 값

날짜 시간 데이터 유형에는 DATE, TIME 및 TIMESTAMP가 있습니다. 날짜 시간 값이 특정 산술이나 문자열 조작에서 사용될 수 있고 특정 문자열과 호환 가능한 경우에도 이들은 문자열이나 숫자가 아닙니다.

날짜

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

날짜의 내부 표현은 4바이트의 문자열입니다. 각 바이트는 2자리의 압축 10진수로 구성됩니다. 처음 2바이트는 연도를, 그 다음 세 번째 바이트는 월을, 마지막 바이트는 일을 나타냅니다.

SQLDA에 설명된 DATE 컬럼의 길이는 10바이트이며, 이것은 값의 문자열 표현에 적합한 길이입니다.

시간

*시간*은 24시간을 하루로 나타내는 세 부분의 값(시, 분, 초)으로 구성됩니다. 시 부분의 범위는 0 - 24입니다. 다른 부분의 범위는 0 - 59입니다. 시가 24이면 분과 초 스펙은 0입니다.

시간의 내부 표현은 3바이트의 문자열입니다. 각 바이트는 2자리의 압축 10진수로 구성됩니다. 첫 번째 바이트는 시간을, 그 다음 바이트는 분을, 마지막 바이트는 초를 나타냅니다.

SQLDA에 설명된 TIME 컬럼의 길이는 8바이트이며, 이것은 값의 문자열 표현에 적합한 길이입니다.

시간소인

시간에 초의 소수를 지정하는 추가 파트도 포함되는 것을 제외하고는 *시간소인*은 위에 정의된 대로 날짜와 시간을 지정하는 6 또는 7파트 값(년, 월, 일, 시, 분, 초 및 선택적 소수 초)으로 구성됩니다. 소수 초의 숫자 수는 0 - 12(디폴트값은 6) 범위의 속성을 사용하여 지정됩니다.

시간소인의 내부 표현은 7 - 13바이트의 문자열입니다. 각 바이트는 2자리의 압축 10진수로 구성됩니다. 처음 4바이트는 날짜를, 그 다음 3바이트는 시간을, 마지막 0 - 6 바이트는 소수 초를 나타냅니다.

SQLDA에 설명된 TIMESTAMP 컬럼의 길이는 19 - 32바이트이며, 이는 값의 문자열 표현에 적합한 길이입니다.

변환된 날짜 시간 값 리턴

데이터 유형이 DATE, TIME 또는 TIMESTAMP인 값은 사용자에게 투명한 내부 양식으로 표시됩니다. 날짜, 시간 및 시간소인 값은 문자열로 표시될 수도 있습니다. 날짜 유형이 DATE, TIME 또는 TIMESTAMP인 상수나 변수는 없으므로, 이는 유용합니다. 날짜 시간 값을 검색하면 먼저 날짜 시간 값을 문자열 변수에 지정해야 합니다. CHAR 함수나 GRAPHIC 함수(유니코드 데이터베이스 전용)는 날짜 시간 값을 문자열 표현으로 변경하는 데 사용할 수 있습니다. 프로그램이 프리컴파일되고 데이터베이스에 바인드될 때 DATETIME 옵션의 스펙으로 대체되는 경우를 제외하고 문자열 표현은 정상적으로 응용프로그램의 지역 코드와 연관된 날짜 시간 값의 디폴트 형식이 됩니다.

길이에 관계 없이, 대형 오브젝트(LOB) 문자열은 날짜 시간 값을 표시하는 문자열로 사용될 수 없습니다(SQLSTATE 42884).

날짜 시간 값의 유효한 문자열 표현이 내부 날짜 시간 값과 함께 조작에 사용되면 문자열 표현은 조작이 수행되기 전에 날짜, 시간 또는 시간소인 값의 내부 형식으로 변환됩니다.

날짜, 시간 및 시간소인 문자열에는 문자와 숫자만 포함되어야 합니다.

날짜 문자열

날짜의 문자열 표현은 숫자로 시작하는 문자열이며 길이는 최소한 8자입니다. 후행 공백을 포함할 수 있으며, 월과 일 위치에서 선행 0은 생략할 수 있습니다.

날짜의 유효한 문자열 형식은 다음 표에 나열되어 있습니다. 각 형식은 이름 및 연관된 약어로 식별됩니다.

표 8. 날짜 문자열 표현 형식

형식 이름	약어	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 표준	JIS	yyyy-mm-dd	1991-10-27
사이트 정의	LOC	응용프로그램의 지역 코드에 따라 다름	—

시간 문자열

시간 문자열 표현은 숫자로 시작하는 문자열이며 길이는 최소한 4자입니다. 후미 공백을 포함할 수 있지만, 시간의 시 부분에서 선행 0은 생략할 수 있으며, 초 부분도 생략할 수 있습니다. 초를 생략할 경우 내재적으로 0초가 지정된 것으로 간주됩니다. 따라서 13:30은 13:30:00와 같습니다.

시간의 유효한 문자열 형식은 다음 표에 나열되어 있습니다. 각 형식은 이름 및 연관된 약어로 식별됩니다.

표 9. 시간 문자열 표현 형식

형식 이름	약어	시간 형식	예 :
국제 표준화 기구	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 표준	JIS	hh:mm:ss	13:30:05
사이트 정의	LOC	응용프로그램의 지역 코드에 따라 다름	—

주:

1. ISO, EUR 또는 JIS 형식에서 .ss(또는 :ss)는 선택적입니다.
2. 국제 표준화 기구는 최근 일본 산업 표준 형식과 동일하게 시간 형식을 변경했습니다. 따라서 응용프로그램에 현재 국제 표준화 기구 형식이 필요한 경우 JIS 형식을 사용하십시오.
3. USA 시간 문자열 형식에서 분 지정을 생략할 수 있는데, 이것은 00분이 내재적으로 지정되었음을 나타냅니다. 따라서 1 PM은 1:00 PM과 같습니다.
4. USA 시간 문자열 형식에서 시간은 12보다 클 수 없으며 00:00 AM과 같은 특수한 경우를 제외하고 0이 될 수 없습니다. 'AM' 또는 'PM' 앞에는 하나의 공백이 있습니다. 'AM' 및 'PM'은 소문자 또는 대문자로 표시할 수 있습니다.

   24시간의 JIS 형식을 사용하는 경우, USA 형식과 대응하는 24시간 형식은 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 숫자로 시작하는 문자열이며 길이는 최소한 16자입니다. 시간소인의 전체 문자열 표현은 *yyyy-mm-dd-hh.mm.ss* 또는 *yyyy-mm-dd-hh.mm.ss.nnnnnnnnnnnn* 양식입니다(여기서, 소수 초의 숫자 수는 0 - 12 범위일 수 있음). 후행 공백을 포함할 수 있습니다. 선행 영(0)은 시간소인의 월, 일, 시 부분에서 생략될 수 있습니다. 후행 영(0)은 절단되거나 소수 초에서 완전히 생략될 수 있습니다. 시간소인의 문자열 표현이 TIMESTAMP 데이터 유형의 값으로 내재적으로 캐스트되는 경우, 캐스트 결과의 시간소인 정밀도는 표현식의 TIMESTAMP 피연산자의 정밀도 또는 지정의 TIMESTAMP 목표의 정밀도에 의해 결정됩니다. 캐스트의 시간소인 정밀도를 초과하는 문자열의 숫자는 절단되거나 캐스트의 시간소인 정밀도에 도달하기 위해 누락된 모든 숫자는 0인 것으로 간주됩니다. 예를 들어, 1991-3-2-8.30.00은 1991-03-02-08.30.00.000000000000과 같습니다.

지정된 정밀도를 가진 시간소인에 명시적으로 값을 캐스트하여 시간소인의 문자열 표현에 다른 시간소인 정밀도를 제공할 수 있습니다. 문자열이 상수인 경우, 대안은 문자열 상수에 TIMESTAMP 키워드를 붙이는 것입니다. 예를 들어, TIMESTAMP ’2007-03-28 14:50:35.123’의 데이터 유형은 TIMESTAMP(3)입니다.

SQL문은 시간소인의 ODBC 문자열 표현을 지원하기도 하나 입력 값으로만 지원합니다. 시간소인의 ODBC 문자열 표현 양식은 *yyyy-mm-dd hh:mm:ss.nnnnnnnnnnnn*입니다(여기서, 소수 초의 숫자 수는 0 - 12 범위일 수 있음).

부울 값

부울 값은 TRUE 또는 FALSE의 참/거짓 값을 나타냅니다. 부울 표현식 또는 술어는 널(NULL) 값으로 표시되는 알 수 없음 값을 작성할 수 있습니다.

BOOLEAN 유형은 다음의 데이터 유형으로만 사용할 수 있는 내장 데이터 유형입니다.

- 복합 SQL(컴파일된)문의 로컬 변수
- SQL 루틴의 매개변수
- SQL 함수의 리턴 유형
- 전역 변수

BOOLEAN 유형으로 정의된 변수 또는 매개변수는 복합 SQL(컴파일된)문에서만 사용할 수 있습니다.

커서 값

커서 값은 밑줄 긋기 커서에 대한 참조를 표시하기 위해 사용됩니다.

CURSOR 유형은 데이터 유형으로만 사용할 수 있는 내장 데이터 유형입니다.

- 복합 SQL(컴파일된)문의 로컬 변수
- SQL 루틴의 매개변수
- SQL 함수의 리턴 유형
- 전역 변수

CURSOR 유형으로 정의된 변수 또는 매개변수는 복합 SQL(컴파일된)문에서만 사용할 수 있습니다.

커서 변수가 커서 유형의 SQL 변수, SQL 매개변수 또는 전역 변수입니다. 커서 변수는 SELECT문에 대해 작성되어 해당 변수에 지정된 커서에 해당하는 기반 커서를 포함합니다. 둘 이상의 커서 변수가 동일한 기반 변수를 공유할 수도 있습니다.

커서 변수는 일반적인 SQL 커서와 동일한 방식으로 사용하여 OPEN, FETCH 및 CLOSE문이 포함된 SELECT문의 결과 세트를 통해 반복됩니다.

XML 값

XML 값은 XML 문서, XML 컨텐츠 또는 XML 노드 시퀀스 형식의 잘 형식화된 XML을 나타냅니다. 테이블에 XML 데이터 유형으로 정의되는 컬럼의 값으로 저장되는 XML 값은 잘 형식화된 XML 문서여야 합니다. XML 값은 문자열 값과 비교할 수 없는 내부 표현에서 처리됩니다. XML 값은 XMLSERIALIZE 함수를 사용하여 XML 문서를 표시하는 구체화된 문자열 값으로 변환될 수 있습니다. 비슷하게 XML 문서를 표시하는 문자열 값은 XMLPARSE 함수를 사용하여 XML 값으로 변환될 수 있습니다. XML 값은 응용프로그램 문자열 및 2진 데이터 유형을 사용하여 교환될 때 내재적으로 구문 분석 또는 직렬화될 수 있습니다.

XML 데이터 유형 값을 결과로 생성하는 표현식에는 특별한 제한사항이 적용됩니다. 이러한 표현식 및 컬럼은 다음에서 사용할 수 없습니다(SQLSTATE 42818).

- DISTINCT절이 앞에 오는 SELECT 목록
- GROUP BY절
- ORDER BY절
- UNION ALL을 제외한 집합 연산자의 subselect
- 기본, 정량화된, BETWEEN, IN 또는 LIKE 술어
- DISTINCT를 사용하는 집계 함수

배열 값

*배열*은 각 요소가 콜렉션에서 해당 인덱스 값으로 참조될 수 있는 순서화된 데이터 요소 콜렉션이 포함된 구조입니다. 배열의 *카디널리티(cardinality)*는 배열에 있는 요소 수입니다. 배열에서 모든 요소의 데이터 유형은 동일합니다.

*일반 배열*에는 요소 수에 대한 상한이 있으며 이를 최대 카디널리티(cardinality)라고도 합니다. 배열의 각 요소는 해당 일반 위치를 인덱스 값으로 참조합니다. *N*이 일반 배열의 요소 수인 경우 각 요소와 연관된 일반 위치는 1 이상으로 *N* 이하인 정수값입니다. *연관된 배열*에는 요소 수에 대한 특정 상한이 없습니다. 각 요소는 해당하는 연관 인덱스 값으로 참조됩니다. 인덱스 값의 데이터 유형은 정수 또는 문자열일 수 있지만 전체 배열에 대해 동일한 데이터 유형입니다.

일반 배열의 최대 카디널리티(cardinality)는 C와 같은 프로그래밍 언어에서 배열의 최대 카디널리티와는 달리 해당하는 실제 표현과 관련되지는 않습니다. 대신, 최대 카디널리티는 서브스크립트가 경계내에 있도록 시스템이 런타임에서 사용합니다. 일반 배열 값을 표시하는 데 필요한 메모리 양은 해당 유형의 최대 카디널리티에 비례하지는 않습니다.

배열 값을 표시하는 데 필요한 메모리 양은 일반적으로 해당 카디널리티에 비례합니다. 배열이 참조될 때 배열의 모든 값은 주기억장치에 저장됩니다. 따라서 많은 양의 데이터를 포함하는 배열은 주기억장치를 많이 사용합니다.

앵커된 유형

앵커된 유형은 컬럼, 전역 변수, SQL 변수, SQL 매개변수 또는 테이블이나 뷰의 행과 같은 다른 SQL 오브젝트를 기초로 데이터 유형을 정의합니다.

앵커된 유형 정의를 사용하여 정의된 데이터 유형은 고정 대상 오브젝트에 대해 종속성을 유지합니다. 앵커 오브젝트 데이터 유형에서의 변경사항은 앵커된 데이터 유형에 영향을 줍니다. 테이블이나 뷰의 행에 앵커되는 경우 앵커된 데이터 유형은 앵커 테이블 또는 앵커 뷰의 컬럼으로 정의된 필드가 있는 ROW입니다.

사용자 정의 유형

사용자 정의 데이터 유형은 다음 여섯 가지입니다.

- 구별 유형
- 구조화된 유형
- 참조 유형
- 배열 유형
- 행 유형
- 커서 유형

이러한 유형 각각에 대해서는 다음 절에서 설명합니다.

구별 유형

*구별 유형*은 해당 내부 표현을 기존 유형(″소스″ 유형)과 공유하지만 대부분의 연산에서 별도의 호환되지 않는 유형으로 간주되는 사용자 정의 데이터 유형입니다. 예를 들어, 그림 유형, 텍스트 유형 및 오디오 유형을 정의하려고 할 경우, 이들 모두는 서로 다른 시맨틱을 갖지만 내부 표현으로 내장 데이터 유형인 BLOB을 사용합니다.

다음 예에서는 AUDIO라는 구별 유형의 작성에 대해 설명합니다.

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

AUDIO가 내장 데이터 유형인 BLOB과 동일하게 표현되더라도, 별도의 유형으로 간주됩니다. 이로써 AUDIO용으로 특별히 작성된 함수를 작성할 수 있으며 이 함수가 다른 데이터 유형(그림, 텍스트 등)의 값에는 적용되지 않도록 합니다.

구별 유형에는 구정된 ID가 있습니다. 스키마 이름이 CREATE TYPE(구별), DROP 또는 COMMENT문 이외의 명령문에서 사용될 때 구별 유형 이름을 규정하는 데 사용되지 않는 경우, SQL 경로에서 일치하는 구별 유형을 가진 첫 번째 스키마를 순서대로 검색합니다.

구별 유형은 구별 유형에 명시적으로 정의된 함수와 연산자만 이 인스턴스에 적용될 수 있도록 하여 명백한 유형 지정을 지원합니다. 이러한 이유로, 소스 유형의 함수와 연산자가 중요하지 않으므로 구별 유형은 이를 자동으로 획득하지 않습니다. 예를 들어, AUDIO의 LENGTH 함수는 오브젝트의 길이를 바이트가 아닌 초 단위로 리턴합니다.

LOB 유형에서 전래된 구별 유형에는 소스 유형과 동일한 제한사항이 적용됩니다.

그러나 소스 유형의 특정 함수와 연산자를 명시적으로 지정하여 구별 유형에 적용할 수 있습니다. 이는 구별 유형의 소스 유형에 정의된 함수에서 전래된 사용자 정의 함수(UDF)를 작성하여 완료될 수 있습니다. 비교 연산자는 BLOB, CLOB 또는 DBCLOB를 소스 유형으로 사용하는 구별 유형을 제외하고, 사용자 정의 구별 유형에 대해 자동으로

생성됩니다. 또한 소스 유형에서 구별 유형으로, 구별 유형에서 소스 유형으로의 캐스팅을 지원하는 함수가 작성됩니다.

구조화된 유형

*구조화된 유형*은 데이터베이스에 정의된 구조가 있는 사용자 정의 데이터 유형입니다. 여기에는 일련의 *속성*이 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, 이러한 테이블이나 뷰를 각각 *유형이 지정된 테이블* 또는 *유형이 지정된 뷰*라고 합니다. 유형이 지정된 테이블 및 유형이 지정된 뷰의 경우, 구조화된 유형의 데이터 이름과 속성은 이 유형이 지정된 테이블이나 유형이 지정된 뷰의 컬럼에 대한 이름과 데이터 유형이 됩니다. 유형이 지정된 테이블 또는 유형이 지정된 뷰의 행들은 구조화된 유형 인스턴스의 표현으로 볼 수 있습니다. 컬럼에 대한 데이터 유형으로 사용될 경우, 컬럼에는 구조화된 유형의 값이나, 아래에 설명된 해당 유형의 부속 유형 값이 포함됩니다. 메소드는 구조화된 컬럼 오브젝트 속성을 검색하거나 조작하는 데 사용됩니다.

용어: *슈퍼 유형*은 *부속 유형*이라는 다른 구조화된 유형이 정의된 구조화된 유형입니다. 부속 유형은 해당 슈퍼 유형의 모든 속성과 메소드를 상속하며 추가 속성과 메소드를 정의할 수 있습니다. 공통된 슈퍼 유형에 연관된 구조화된 유형 세트를 자료형 *계층*이라고 하며 슈퍼 유형을 전혀 포함하지 않는 유형은 자료형 계층의 *루트 유형*이라고 합니다.

부속 유형이라는 용어는 사용자 정의 구조화된 유형과, 유형 계층 구조에서 그 아래에 있는 모든 사용자 정의 구조화된 유형에 적용됩니다. 그러므로 구조화된 유형 **T**의 부속 유형은 계층 구조에서 **T**와 **T** 아래에 있는 모든 구조화된 유형입니다. 구조화된 유형 **T**의 *적절한 부속 유형*은 유형 계층에서 **T** 아래에 있는 구조화된 유형입니다.

유형 계층 구조에서 재귀 유형 정의를 포함하는 데는 제한사항이 있습니다. 이러한 이유로, 허용되는 재귀 정의의 특정 유형을 참조하는 간단한 방법을 개발해야 합니다. 사용되는 정의는 다음과 같습니다.

- *직접 사용*: 다음 중 하나가 참인 경우 유형 **A**가 다른 유형 **B**를 직접 사용한다고 합니다.
  1. 유형 **A**에 유형 **B**의 속성이 있습니다.
  2. 유형 **B**가 **A**의 부속 유형이거나 **A**의 슈퍼 유형입니다.
- *간접 사용*: 다음 중 하나가 참인 경우 유형 **A**가 유형 **B**를 간접적으로 사용한다고 합니다.
  1. 유형 **A**가 직접적으로 유형 **B**를 사용합니다.

2. 유형 **A**가 유형 **C**를 직접적으로 사용하고, 유형 **C**는 간접적으로 유형 **B**를 사용합니다.

유형은 해당 속성 유형 중 하나가 직접 또는 간접적으로 자신을 사용하도록 정의될 수 없습니다. 이러한 구성이 필요할 경우, 참조를 속성으로 사용하는 것을 고려해 보십시오. 예를 들어, 구조화된 유형 속성에서 ″manager″가 ″employee″ 유형일 경우, 속성이 ″manager″인 ″employee″ 인스턴스는 있을 수 없습니다. 그러나 유형이 REF(employee)인 ″manager″ 속성은 있을 수 있습니다.

특정 다른 오브젝트에서 유형을 직접 또는 간접적으로 사용할 경우 유형을 삭제할 수 없습니다. 예를 들어, 테이블이나 뷰 컬럼이 직접 또는 간접적으로 유형을 사용할 경우 유형을 삭제할 수 없습니다.

참조 유형

*참조 유형*은 구조화된 유형의 동종 유형입니다. 구별 유형과 마찬가지로 참조 유형은 내장 데이터 유형 중 하나와 공통 표현을 공유하는 스칼라 유형입니다. 이러한 동일한 표현은 유형 계층 구조의 모든 유형들이 공유합니다. 참조 유형 표현은 유형 계층 구조의 루트 유형을 작성할 때 정의됩니다. 참조 유형을 사용할 때 구조화된 유형은 유형의 매개변수로 지정됩니다. 이 매개변수를 참조의 *목표 유형*이라 합니다.

참조 목표는 항상 유형이 지정된 테이블이나 유형이 지정된 뷰에 있는 행입니다. 참조 유형을 사용할 때 *범위*를 정의할 수 있습니다. 범위는 참조 값의 목표 행이 들어 있는 테이블(*목표 테이블*) 또는 뷰(*목표 뷰*)를 식별합니다. 목표 테이블 또는 뷰에는 참조 유형의 목표 유형과 동일한 유형이 있어야 합니다. 범위가 지정된 참조 유형의 인스턴스는 *목표 행*이라는 유형이 지정된 뷰 또는 유형이 지정된 테이블의 행을 고유하게 식별합니다.

배열 유형

사용자 정의 *배열 유형*은 다른 데이터 유형의 요소가 포함된 배열로 정의된 데이터 유형입니다. 모든 일반 배열 유형에는 데이터 유형이 INTEGER인 인덱스가 포함되고 정의된 최대 카디널리티(cardinality)가 있습니다. 모든 연관된 배열에는 INTEGER 또는 VARCHARE 데이터 유형의 인덱스가 있으며 정의된 최대 카디널리티가 없습니다.

행 유형

*행 유형*은 행을 효과적으로 나타내며 각각 연관된 데이터 유형을 가진 이름 지정된 필드의 순서화된 시퀀스로 정의된 데이터 유형입니다. 행 유형을 SQL PL의 변수 및 매개변수에 대한 데이터 유형으로 사용하여 데이터 행을 간단히 조작할 수 있습니다.

커서 데이터 유형

사용자 정의 *커서 유형*은 키워드 CURSOR 및 선택적으로 연관된 행 유형으로 정의된 사용자 정의 데이터 유형입니다. 연관된 행 유형이 포함된 사용자 정의 커서 유형은 *명백히 유형이 지정된 커서 유형*이며 그 이외의 경우는 *약하게 유형이 지정된 커서 유형*입니다. 사용자 정의 커서 유형 값은 기본 커서에 대한 참조를 나타냅니다.

데이터 유형 승격

데이터 유형은 관련 데이터 유형 그룹으로 분류할 수 있습니다. 그룹 내에는 우선순위가 있으므로 하나의 데이터 유형은 다른 데이터 유형에 선행되는 것으로 간주됩니다. 이러한 우선순위에서는 한 데이터 유형을 뒤에 나오는 데이터 유형으로 *승격*시킬 수 있습니다. 예를 들어, 데이터 유형 CHAR는 VARCHAR로 승격될 수 있고, INTEGER는 DOUBLE-PRECISION으로 승격될 수 있지만, CLOB은 VARCHAR로 승격될 수 없습니다.

데이터 유형 승격은 다음 경우에 사용됩니다.

- 함수 결정 수행
- 사용자 정의 유형 캐스팅
- 사용자 정의 유형을 내장 데이터 유형에 지정

표 10은 각 데이터 유형에 대한 우선순위 목록(순서)을 보여주며, 제공된 데이터 유형이 승격될 수 있는 데이터 유형을 결정하는 데 사용됩니다. 이 표는 다른 데이터 유형으로 승격하는 대신 항상 같은 데이터 유형을 사용하는 것이 가장 좋은 선택임을 나타냅니다.

표 10. 데이터 유형 우선순위 표

데이터 유형	데이터 유형 우선순위 목록(가장 적합한 데이터 유형부터)
SMALLINT	SMALLINT, INTEGER, BIGINT, 10진수, real, double, DECFLOAT
INTEGER	INTEGER, BIGINT, 10진수, real, double, DECFLOAT
BIGINT	BIGINT, 10진수, real, double, DECFLOAT
10진수	10진수, real, double, DECFLOAT
real	real, double, DECFLOAT
double	double, DECFLOAT
DECFLOAT	DECFLOAT
CHAR	CHAR, VARCHAR, CLOB
VARCHAR	VARCHAR, CLOB
CLOB	CLOB
GRAPHIC	GRAPHIC, VARGRAPHIC, DBCLOB
VARGRAPHIC	VARGRAPHIC, DBCLOB
DBCLOB	DBCLOB
BLOB	BLOB
DATE	DATE, TIMESTAMP
TIME	TIME
TIMESTAMP	TIMESTAMP
BOOLEAN	BOOLEAN
CURSOR	CURSOR
ARRAY	ARRAY
udt	udt(같은 이름) 또는 udt의 슈퍼 유형

표 10. 데이터 유형 우선순위 표 (계속)

데이터 유형	데이터 유형 우선순위 목록(가장 적합한 데이터 유형부터)
REF(T)	REF(S)(S는 T의 슈퍼 유형임)
ROW	ROW

주:

1. 위의 소문자 유형은 다음과 같이 정의됩니다.
   - decimal = DECIMAL(p,s) 또는 NUMERIC(p,s)
   - real = REAL 또는 FLOAT(*n*). 여기서, *n*는 24보다 크지 않습니다.
   - double = DOUBLE, DOUBLE-PRECISION, FLOAT 또는 FLOAT(*n*). 여기서, *n*은 24보다 큽니다.
   - udt = 사용자 정의 유형

   나열된 데이터 유형의 짧거나 긴 양식의 동의어는 나열된 양식과 동일한 것으로 간주됩니다.
2. 유니코드 데이터베이스의 경우, 다음은 동일한 데이터 유형으로 간주됩니다.
   - CHAR 및 GRAPHIC
   - VARCHAR 및 VARGRAPHIC
   - CLOB 및 DBCLOB

   유니코드 데이터베이스에서 함수를 해석할 때, 사용자 정의 함수(UDF) 및 내장 함수가 둘 다 주어진 함수 호출에 적용할 수 있는 경우 일반적으로 내장 함수가 호출됩니다. UDF는 유니코드 데이터 유형 동등성과는 상관없이 CURRENT PATH 특수 레지스터에서 스키마가 SYSIBM 앞에 오는 경우 및 인수 데이터 유형이 모든 함수 호출 인수 데이터 유형과 일치하는 경우에만 호출됩니다.

데이터 유형 간 캐스팅

주어진 데이터 유형의 값을 다른 데이터 유형으로 또는 길이, 정밀도 또는 스케일이 다른 같은 데이터 유형으로 *캐스트*해야 하는 경우가 종종 있습니다. 데이터 유형 승격은 한 데이터 유형에서 다른 데이터 유형으로 승격하기 위해서는 값을 새 데이터 유형으로 캐스트해야 하는 한 예입니다. 다른 데이터 유형으로 캐스트할 수 있는 데이터 유형은 소스 데이터 유형에서 목표 데이터 유형으로 *캐스트 가능*합니다.

한 데이터 유형에서 다른 데이터 유형으로의 캐스팅은 내재적 또는 명시적으로 발생할 수 있습니다. 캐스트(cast) 함수, CAST 스펙 또는 XMLCAST 스펙을 사용하여 관련된 데이터 유형에 따라 데이터 유형을 명시적으로 변경할 수 있습니다. 또한 목표 사용자 정의 함수(UDF) 작성시에도 소스 함수에 대한 매개변수의 데이터 유형을 작성하려는 함수의 데이터 유형에 캐스트할 수 있어야 합니다.

내장 데이터 유형 간에 지원되는 캐스트는 123 페이지의 표 11에 나와 있습니다. 첫 번째 컬럼은 캐스트 피연산자의 데이터 유형(소스 데이터 유형)을 나타내며, 맨 위에 있는 데이터 유형은 캐스트 스펙의 목표 데이터 유형을 나타냅니다. 'Y'는 소스 및 목표 데이터 유형의 조합에 CAST 스펙을 사용할 수 있음을 나타냅니다. XMLCAST 스펙만 사용할 수 있는 경우를 명시합니다.

데이터 유형이 문자 또는 그래픽 데이터 유형으로 캐스트될 때 절단이 발생하는 경우, 비공백 문자가 절단되면 경고가 리턴됩니다. 이 절단 동작은 비공백 문자가 절단되는 경우 오류가 발생할 때 문자 또는 그래픽 데이터 유형을 지정하는 것과 다릅니다.

구별 유형과 관련하여 다음과 같은 캐스트를 지원합니다(다르게 명시되지 않은 경우 CAST 스펙 사용).

- 구별 유형 *DT*에서 소스 데이터 유형 *S*로 캐스트
- 구별 유형 *DT*의 소스 데이터 유형 *S*에서 구별 유형 *DT*로 캐스트
- 구별 유형 *DT*에서 같은 구별 유형 *DT*로 캐스트
- 데이터 유형 *A*에서 구별 유형 *DT*로 캐스트. 여기서, *A*는 구별 유형 *DT*의 소스 데이터 유형 *S*로 승격할 수 있습니다.
- INTEGER에서 소스 데이터 유형이 SMALLINT인 구별 유형 *DT*로 캐스트
- DOUBLE에서 소스 데이터 유형이 REAL인 구별 유형 *DT*로 캐스트
- DECFLOAT에서 소스 데이터 유형이 DECFLOAT인 구별 유형 DT로 캐스트
- VARCHAR에서 소스 데이터 유형이 CHAR인 구별 유형 *DT*로 캐스트
- VARGRAPHIC에서 소스 데이터 유형이 GRAPHIC인 구별 유형 *DT*로 캐스트
- 유니코드 데이터베이스의 경우, VARCHAR 또는 VARGRAPHIC에서 소스 데이터 유형이 CHAR 또는 GRAPHIC인 구별 유형 *DT*로 캐스트
- XMLCAST 스펙을 사용하여 소스 데이터 유형이 *S*인 구별 유형 *DT*에서 XML로 캐스트

- XML 값의 XML 스키마 데이터 유형에 따라 XMLCAST 스펙을 사용하여 XML에서 소스 데이터 유형이 내장 데이터 유형인 구별 유형 *DT*로 캐스트

FOR BIT DATA 문자 유형을 CLOB로 캐스트할 수 없습니다.

배열 유형을 목표로 포함하는 캐스트의 경우, 소스 배열 값 요소의 데이터 유형이 목표 배열 데이터 요소의 데이터 유형에 캐스트할 수 있어야 합니다(SQLSTATE 42846). 목표 배열 유형이 일반 배열인 경우, 소스 배열 값은 일반 배열이고(SQLSTATE 42821) 소스 배열 값의 카디널리티(cardinality)가 목표 배열 데이터 유형의 최대 카디널리티 이하여야 합니다(SQLSTATE 2202F). 목표 배열 유형이 연관된 배열인 경우, 소스 배열 값에 대한 인덱스의 데이터 유형은 목표 배열 유형에 대한 인덱스의 데이터 유형에 캐스트 가능해야 합니다. 사용자 정의 배열 유형 값은 동일한 이름을 갖는 다른 사용자 정의 배열 유형에만 캐스트될 수 있습니다(SQLSTATE 42846).

커서 유형은 커서 유형으로 매개변수 표시문자를 캐스트하는 경우를 제외하고 CAST 스펙의 목표 데이터 유형 또는 소스 데이터 유형일 수 없습니다.

행 유형을 목표로 포함하는 캐스트의 경우, 소스 행 값 표현식의 등급과 목표 행 유형의 등급이 일치해야 하며 소스 행 값 표현식의 각 필드를 해당 목표 필드에 캐스트할 수 있어야 합니다. 사용자 정의 행 유형 값은 동일한 이름을 갖는 다른 사용자 정의 행 유형에만 캐스트될 수 있습니다(SQLSTATE 42846).

구조화된 유형 값은 다른 어떤 것으로도 캐스트할 수 없습니다. 구조화된 유형 *ST*는 *ST*의 슈퍼 유형에 대한 모든 메소드가 *ST*에 적용 가능하므로 해당 슈퍼 유형 중 하나로 캐스트할 수 없습니다. 원하는 조작이 *ST*의 부속 유형에만 적용 가능할 경우, 부속 유형 처리 표현식을 사용하여 *ST*를 해당되는 부속 유형 중 하나로 처리하십시오.

캐스트와 관련된 사용자 정의 데이터 유형이 스키마 이름으로 규정되지 않은 경우, 사용자 정의 데이터 유형이 포함된 첫 번째 스키마를 이름으로 찾는 데 *SQL 경로*가 사용됩니다.

참조 유형과 관련하여 다음과 같은 캐스트가 지원됩니다.

- 참조 유형 *RT*에서 표현 데이터 유형 *S*로 캐스트
- 참조 유형 *RT*의 표현 데이터 유형 *S*에서 참조 유형 *RT*로 캐스트
- 목표 유형이 *T*인 참조 유형 *RT*에서 목표 유형이 *S*인 참조 유형 *RS*로 캐스트. 여기서, *S*는 *T*의 슈퍼 유형입니다.
- 데이터 유형 *A*에서 참조 유형 *RT*로 캐스트. 여기서, *A*는 참조 유형 *RT*의 표현 데이터 유형 *S*로 승격할 수 있습니다.

캐스트와 관련된 참조 데이터 유형의 목표 유형이 스키마 이름으로 규정되지 않은 경우, 사용자 정의 데이터 유형이 포함된 첫 번째 스키마를 이름으로 찾는 데 *SQL 경로*가 사용됩니다.

표 11. 내장 데이터 유형 간에 지원되는 캐스트

	목표 데이터 유형																				
소스 데이터 유형	SMALLINT	INTEGER	BIGINT	DECIMAL	REAL	DOUBLE	DECFLOAT	CHAR	CHAR FOR BIT DATA	VARCHAR	VARCHAR FOR BIT DATA	CLOB	GRAPHIC	VARGRAPHIC	DBCLOB	BLOB	DATE	TIME	TIMESTAMP	XML	BOOLEAN
SMALLINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	Y[3]	-
INTEGER	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	Y[3]	-
BIGINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	Y[3]	-
DECIMAL	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	Y[3]	-
REAL	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	Y[3]	-
DOUBLE	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	Y[3]	-
DECFLOAT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	-	-	-	-
CHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y	Y	Y	Y	Y[4]	-
CHAR FOR BIT DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	Y[3]	-
VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y	Y	Y	Y	Y[4]	-
VARCHAR FOR BIT DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	Y[3]	-
CLOB	-	-	-	-	-	-	-	Y	-	Y	-	Y	Y[1]	Y[1]	Y[1]	Y	-	-	-	Y[4]	-
GRAPHIC	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	-	Y[1]	-	Y[1]	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y[3]	-
VARGRAPHIC	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	-	Y[1]	-	Y[1]	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y[3]	-
DBCLOB	-	-	-	-	-	-	-	Y[1]	-	Y[1]	-	Y[1]	Y	Y	Y	Y	-	-	-	Y[3]	-
BLOB	-	-	-	-	-	-	-	-	Y	-	Y	-	-	-	-	Y	-	-	-	Y[4]	-
DATE	-	Y	Y	Y	-	-	-	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	Y	-	Y	Y[3]	-
TIME	-	Y	Y	Y	-	-	-	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	Y	-	Y[3]	-
TIMESTAMP	-	-	Y	Y	-	-	-	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	Y	Y	Y	Y[3]	-
XML	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y	-
BOOLEAN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

표 11. 내장 데이터 유형 간에 지원되는 캐스트 (계속)

	목표 데이터 유형																					
소스 데이터 유형	SMALLINT	INTEGER	BIGINT	DECIMAL	NUMERIC	REAL	DOUBLE	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	BLOB	DATE	TIME	TIMESTAMP	XML	DECFLOAT	BOOLEAN
SMALLINT																						
INTEGER																						
BIGINT																						
DECIMAL																						
NUMERIC																						
REAL																						
DOUBLE																						
CHAR																						
VARCHAR																						
LONG VARCHAR																						
CLOB																						
GRAPHIC																						
VARGRAPHIC																						
LONG VARGRAPHIC																						
DBCLOB																						
BLOB																						
DATE																						
TIME																						
TIMESTAMP																						
XML																						
DECFLOAT																						
BOOLEAN																						

참고:

- 사용자 정의 유형 및 참조 유형과 관련하여 지원되는 캐스트에 대해서는 표 앞에 있는 설명을 참조하십시오.
- 구조화된 유형 값은 다른 어떤 것으로도 캐스트할 수 없습니다.
- LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다. 추후 릴리스에서는 제거될 수 있습니다.

[1] 캐스트는 유니코드 데이터베이스에 대해서만 지원됩니다.

[2] FOR BIT DATA

[3] XMLCAST만을 사용하여 캐스트를 수행할 수 있습니다.

[4] XMLPARSE 함수를 내재적으로 처리하여 문자열을 XML 컬럼에 지정할 때(INSERT 또는 UPDATE) 문자열을 XML로 변환합니다. 지정을 성공적으로 수행하려면 문자열이 well-formed XML 문서여야 합니다.

[5] XMLCAST만을 사용하여 캐스트를 수행할 수 있으며 XML 값의 기본 XML 스키마 데이터 유형에 종속됩니다. 자세한 내용은 『XMLCAST』를 참조하십시오.

[6] 커서 유형은 커서 유형으로 매개변수 표시문자를 캐스트하는 경우를 제외하고 CAST 스펙의 목표 데이터 유형 또는 소스 데이터 유형일 수 없습니다.

테이블 표 12는 식별된 목표 데이터 유형으로 캐스트할 때 적용되는 규칙에 대한 정보가 있는 위치를 표시합니다.

표 12. 데이터 유형 캐스팅 규칙

목표 데이터 유형	규칙
SMALLINT	*SQL 참조서, 볼륨 1*의 『SMALLINT 스칼라 함수』
INTEGER	*SQL 참조서, 볼륨 1*의 『INTEGER 스칼라 함수』
BIGINT	*SQL 참조서, 볼륨 1*의 『BIGINT 스칼라 함수』
DECIMAL	*SQL 참조서, 볼륨 1*의 『DECIMAL 스칼라 함수』
NUMERIC	*SQL 참조서, 볼륨 1*의 『DECIMAL 스칼라 함수』
REAL	*SQL 참조서, 볼륨 1*의 『REAL 스칼라 함수』

표 12. 데이터 유형 캐스팅 규칙 (계속)

목표 데이터 유형	규칙
DOUBLE	*SQL 참조서, 볼륨 1*의 『DOUBLE 스칼라 함수』
DECFLOAT	*SQL 참조서, 볼륨 1*의 『DECFLOAT 스칼라 함수』
CHAR	*SQL 참조서, 볼륨 1*의 『CHAR 스칼라 함수』
VARCHAR	*SQL 참조서, 볼륨 1*의 『VARCHAR 스칼라 함수』
CLOB	*SQL 참조서, 볼륨 1*의 『CLOB 스칼라 함수』
GRAPHIC	*SQL 참조서, 볼륨 1*의 『GRAPHIC 스칼라 함수』
VARGRAPHIC	*SQL 참조서, 볼륨 1*의 『VARGRAPHIC 스칼라 함수』
DBCLOB	*SQL 참조서, 볼륨 1*의 『DBCLOB 스칼라 함수』
BLOB	*SQL 참조서, 볼륨 1*의 『BLOB 스칼라 함수』
DATE	*SQL 참조서, 볼륨 1*의 『DATE 스칼라 함수』
TIME	*SQL 참조서, 볼륨 1*의 『TIME 스칼라 함수』
TIMESTAMP	소스 유형이 문자열이면, *SQL 참조서, 볼륨 1*에서 하나의 피연산자가 지정된 『TIMESTAMP 스칼라 함수』를 참조하십시오. 소스 데이터 유형이 DATE 이면, 시간소인은 지정된 날짜와 00:00:00 시간으로 구성됩니다.

XML 값으로 비XML 값 캐스팅

표 13. 비XML 값에서 XML 값으로의 지원되는 캐스트

	목표 데이터 유형	
소스 데이터 유형	**XML**	결과 **XML** 스키마 유형
SMALLINT	Y	xs:short
INTEGER	Y	xs:int
BIGINT	Y	xs:long
DECIMAL 또는 NUMERIC	Y	xs:decimal
REAL	Y	xs:float
DOUBLE	Y	xs:double
DECFLOAT	N	-
CHAR	Y	xs:string
VARCHAR	Y	xs:string
CLOB	Y	xs:string
GRAPHIC	Y	xs:string
VARGRAPHIC	Y	xs:string
DBCLOB	Y	xs:string
DATE	Y	xs:date
TIME	Y	xs:time
TIMESTAMP	Y	xs:dateTime[1]

표 13. 비XML 값에서 XML 값으로의 지원되는 캐스트 (계속)

	목표 데이터 유형	
소스 데이터 유형	**XML**	**결과 XML 스키마 유형**
BLOB	Y	xs:base64Binary
FOR BIT DATA 문자 유형	Y	xs:base64Binary
구별 유형		구별 유형을 소스 유형으로 하여 해당 차트 사용

주

[1] TIMESTAMP 소스 데이터 유형은 0 - 12의 시간소인 정밀도를 지원합니다. xs:dateTime의 최대 소수초 정밀도는 6입니다. TIMESTAMP 소스 데이터 유형의 시간소인 정밀도가 6을 초과하면, xs:dateTime으로 캐스트될 때 값이 절단됩니다.

LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다. 추후 릴리스에서는 제거될 수 있습니다.

문자열 값을 XML 값으로 캐스트할 때, 결과로 생성되는 xs:string 원자값이 유효하지 않은 XML 문자를 포함할 수 없습니다(SQLSTATE 0N002). 입력 문자열이 유니코드가 아닐 경우, 입력 문자를 유니코드로 변환합니다.

SQL 2진 유형으로 캐스팅하면 유형이 xs:base64Binary인 XQuery 원자값이 생성됩니다.

비XML 값으로 XML 값 캐스팅

XML 값에서 비XML 값으로의 XMLCAST를 다음과 같은 두 가지 캐스트로 설명할 수 있습니다. 소스 XML 값을 SQL 목표 유형에 해당하는 XQuery 유형으로 변환하는 XQuery 캐스트 다음 해당 XQuery 유형에서 실제 SQL 유형으로의 캐스트.

목표 유형에 해당하는 지원되는 XQuery 목표 유형이 있을 경우 및 소스 값 유형에서 해당하는 XQuery 목표 유형으로의 지원되는 XQuery 캐스트가 있을 경우 XMLCAST가 지원됩니다. XQuery 캐스트에서 사용되는 목표 유형은 해당하는 XQuery 목표 유형을 기반으로 하며 몇 가지 추가 제한사항을 포함할 수도 있습니다.

다음 표는 해당 변환의 결과로 생성되는 XQuery 유형을 나열합니다.

표 14. XML 값에서 비XML 값으로의 지원되는 캐스트

	소스 데이터 유형	
목표 데이터 유형	**XML**	**해당하는 XQuery 목표 유형**
SMALLINT	Y	xs:short
INTEGER	Y	xs:int
BIGINT	Y	xs:long
DECIMAL 또는 NUMERIC	Y	xs:decimal
REAL	Y	xs:float
DOUBLE	Y	xs:double

표 14. XML 값에서 비XML 값으로의 지원되는 캐스트 (계속)

목표 데이터 유형	소스 데이터 유형	
	XML	**해당하는 XQuery 목표 유형**
DECFLOAT	Y	일치하는 유형 없음[1]
CHAR	Y	xs:string
VARCHAR	Y	xs:string
CLOB	Y	xs:string
GRAPHIC	Y	xs:string
VARGRAPHIC	Y	xs:string
DBCLOB	Y	xs:string
DATE	Y	xs:date
TIME(시간대 없음)	Y	xs:time
TIMESTAMP(시간대 없음)	Y	xs:dateTime[2]
BLOB	Y	xs:base64Binary
CHAR FOR BIT DATA	N	캐스트 불가능
VARCHAR FOR BIT DATA	Y	xs:base64Binary
구별 유형		구별 유형을 소스 유형으로 하여 해당 차트 사용
행, 참조, 구조화된 또는 추상 데이터 유형 (ADT), 기타	N	캐스트 불가능

주

[1] DB2는 DECFLOAT에 일치하는 XML 스키마 유형을 제공하지 않는 XML 스키마 1.0을 지원합니다. XMLCAST의 XQuery 캐스트 단계 처리는 다음과 같이 처리됩니다.

- 소스 값이 XML 스키마 숫자 값을 사용하여 유형이 지정된 경우, 해당 숫자 유형을 사용하십시오.
- 소스 값이 XML 스키마 유형 xs:boolean을 사용하여 유형이 지정된 경우, xs:double을 사용하십시오.
- 그렇지 않으면, xs:string을 사용하여 유효한 숫자 형식을 추가로 검사하십시오.

[2] xs:dateTime의 최대 소수 초 정밀도는 6입니다. TIMESTAMP 소스 데이터 유형은 0 - 12의 시간소인 정밀도를 지원합니다. TIMESTAMP 목표 데이터 유형의 시간소인 정밀도가 6보다 작으면, xs:dateTime에서 캐스트될 때 값이 절단됩니다. TIMESTAMP 목표 데이터 유형의 시간소인 정밀도가 6을 초과하면, xs:dateTime에서 캐스트될 때 값이 영(0)으로 채워집니다.

다음과 같은 제한사항 사례에서 유도된 제한사항 XML 스키마 데이터 유형은 XQuery 캐스트의 목표 데이터 유형으로 효과적으로 사용됩니다.

- 문자열로 변환하려는 XML 값이 문자 또는 바이트를 절단하지 않고도 해당 DB2 유형의 길이 한계 이내여야 합니다. 유도된 XML 스키마 유형에 사용되는 이름은 대문자 SQL 유형 이름 다음 밑줄 문자 및 문자열의 최대 길이여야 합니다(예: XMLCAST 목표 데이터 유형이 VARCHAR(20)일 경우 VARCHAR_20).
- DECIMAL 값으로 변환될 XML 값이 지정된 DECIMAL 값의 정밀도 이내여야 하며, 스케일보다 많은 소수점 뒤에 0이 아닌 숫자를 더 많이 포함하지 않아야 합니다. 유도된 XML 스키마 유형에 사용되는 이름은 DECIMAL_*precision_scale*이며,

여기서 *precision*은 목표 SQL 데이터 유형의 정밀도이고, *scale*은 목표 SQL 데이터 유형의 스케일입니다(예: XMLCAST 목표 데이터 유형이 DECIMAL(9,2)일 경우 DECIMAL_9_2).

- TIME 값으로 변환될 XML 값이 소수점 뒤에 0이 아닌 숫자가 있는 초 구성요소를 포함할 수 없습니다. 유도된 XML 스키마 유형에 사용되는 이름은 TIME입니다.

유도된 XML 스키마 유형 이름은 XML 값이 다음 제한사항 중 하나를 따르지 않을 경우에만 메시지에 표시됩니다. 이 유형 이름은 오류 메시지를 이해하는 데 도움이 되며, 정의된 XQuery 유형에 대응하지 않습니다. 입력 값이 기본 XML 스키마 유형의 기본 유형(대응하는 XML 목표 유형), 오류 메시지에서 대신 해당 유형을 표시할 수도 있습니다. 해당 유도된 XML 스키마 유형 이름 형식이 나중에 변경될 수도 있으므로, 프로그래밍 인터페이스로 사용할 수 없습니다.

XML 값을 XQuery 캐스트에서 처리하기 전에, 시퀀스의 문서 노드가 제거되며 제거된 문서 노드의 각 직접 하위가 시퀀스의 항목이 됩니다. 문서 노드에 복수의 직접 하위 노드가 있을 경우, 개정된 시퀀스는 원래 시퀀스보다 많은 항목을 갖습니다. 문서 노드가 없는 XML 값은 XQuery fn:data 함수를 사용하여 세분화되어 결과적으로 XQuery 캐스트에 사용되는 시퀀스 값을 세분화합니다. 세분화된 시퀀스 값이 공백 시퀀스이면 추가 처리 없이 캐스트로부터 널(NULL) 값이 리턴됩니다. 세분화된 시퀀스 값에 복수의 항목이 있으면 오류가 리턴됩니다(SQLSTATE 10507).

XMLCAST의 목표 유형이 SQL 데이터 유형 DATE, TIME 또는 TIMESTAMP일 경우, XQuery 캐스트로부터 결과로 생성되는 XML 값 또한 UTC로 조정되며 값의 시간대 구성요소는 제거됩니다.

해당하는 XQuery 목표 유형 값을 SQL 목표 유형으로 변환할 경우, xs:base64Binary 또는 xs:hexBinary 같은 2진 XML 데이터 유형이 문자 형식에서 실제 2진 데이터로 변환됩니다.

INF, -INF 또는 NaN의 xs:double 또는 xs:float 값을 SQL 데이터 유형 DOUBLE 또는 REAL 값으로 캐스트할 경우, 오류가 리턴됩니다(SQLSTATE 22003). -0의 xs:double 또는 xs:float 값은 +0으로 변환됩니다.

소스 피연산자가 사용자 정의 구별 유형이 아닐 경우 목표 유형이 사용자 정의 구별 유형일 수 있습니다. 이 때 XMLCAST 스펙을 사용하여 소스 값을 사용자 정의 구별 유형(즉, 목표 유형)의 소스 유형으로 캐스트한 후 CAST 스펙을 사용하여 이 값을 사용자 정의 구별 유형으로 캐스트합니다.

비유니코드 데이터베이스에서, XML 값에서 비XML 목표 유형으로의 캐스팅은 내부 UTF-8 형식에서 데이터베이스 코드 페이지로의 코드 페이지 변환을 포함합니다. 이 변환의 결과 XML 값의 코드 포인트가 데이터베이스 코드 페이지에 없으면 대체 문자가 도입됩니다.

지정 및 비교

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

두 조작에 대한 한 가지 기본 규칙은 포함된 피연산자의 데이터 유형이 호환되어야 한다는 것입니다. 호환성 규칙은 설정 조작에도 적용됩니다.

지정 조작에 대한 또 다른 기본 규칙은 널(NULL) 값을 포함할 수 없는 컬럼이나 연관된 표시기 변수가 없는 호스트 변수에 널(NULL) 값을 지정할 수 없다는 것입니다.

다음은 지정 및 비교 조작에 대한 시스템 정의 데이터 유형 호환성을 보여주는 호환성 매트릭스입니다.

표 15. 지정 및 비교에 대한 데이터 유형 호환성

피연산자	2진 정수	10진수	부동 소수점	10진 부동 소수점	문자열	그래픽 문자열	날짜	시간	시간소인	실행 파일 문자열	부울	UDT
2진 정수	예	예	예	예	YES	예 [5]	없음	없음	없음	없음	아니오	[2]
10진수	예	예	예	예	YES	예 [5]	없음	없음	없음	없음	아니오	[2]
부동 소수점	예	예	예	예	YES	예 [5]	없음	없음	없음	없음	아니오	[2]
10진 부동 소수점	예	예	예	예	YES	예 [5]	없음	없음	없음	없음	아니오	[2]
문자열	YES	YES	YES	YES	예	예 [5,6]	YES	YES	YES	아니오[3]	아니오	[2]
그래픽 문자열	예 [5]	예 [5]	예 [5]	예 [5]	예 [5,6]	예	예 [5]	예 [5]	예 [5]	없음	아니오	[2]
날짜	없음	없음	없음	없음	YES	예 [5]	예	없음	YES	없음	아니오	[2]
시간	없음	없음	없음	없음	YES	예 [5]	없음	예	[1]	없음	아니오	[2]
시간- 소인	없음	없음	없음	없음	YES	예 [5]	YES	[1]	예	없음	아니오	[2]
실행 파일 문자열	없음	없음	없음	없음	예[3]	없음	없음	없음	없음	예	아니오	[2]
부울	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	예 [7]	[2]
UDT	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	예

[1] TIMESTAMP 값을 TIME 값에 지정할 수 있습니다. 그러나 TIME 값을 TIMESTAMP 값에 지정할 수는 없으며 TIMESTAMP 값을 TIME 값과 비교할 수 없습니다.

[2] 사용자 정의 구별 유형 값은 동일한 사용자 정의 구별 유형으로 정의된 값하고만 비교할 수 있습니다. 일반적으로 지정은 구별 유형 값과 해당 소스 데이터 유형 간에 지원됩니다. 사용자 정의 구조화된 유형은 비교할 수 없으며 같은 구조화된 유형의 피연산자나 해당 슈퍼 유형 중 하나에만 지정할 수 있습니다. 일부 추가적인 지정 규칙이 행, 커서 및 배열 유형에 적용됩니다. 자세한 내용은 137 페이지의 『사용자 정의 유형 지정』의 내용을 참조하십시오.

[3] 지정(비교 아님)에 대해서만 지원되며 FOR BIT DATA로 정의된 문자열에 대해서만 지원합니다.

[4] 참조 유형의 지정 및 비교에 대한 정보는 139 페이지의 『참조 유형 지정』 및 145 페이지의 『참조 유형 비교』의 내용을 참조하십시오.

[5] 유니코드 데이터베이스에 대해서만 지원됩니다.

[6] 비트 데이터 및 그래픽 문자열은 호환되지 않습니다.

[7] 부울 데이터 유형의 변수를 직접 비교할 수는 없습니다. 리터럴 값(TRUE, FALSE, NULL)을 사용해야만 비교를 수행할 수 있습니다.

숫자 지정

숫자 지정의 경우, 오버플로우가 허용되지 않습니다.

- 완전 숫자 데이터 유형에 지정할 때 숫자의 정수 부분의 숫자가 제거되면 오버플로우가 발생합니다. 필요하면, 숫자의 소수 부분이 절단됩니다.
- 근사 숫자 데이터 유형 또는 10진 부동 소수점에 지정할 때 숫자의 정수 부분의 최상위 숫자가 제거되면 오버플로우가 발생합니다. 부동 소수점 및 10진 부동 소수점 숫자의 경우, 숫자의 정수 부분은 부동 소수점 또는 10진 부동 소수점 숫자가 무제한 정밀도를 가진 10진수로 변환된 경우 발생하는 숫자입니다. 필요하면 반올림으로 인해 숫자의 최하위 숫자가 제거될 수 있습니다.

  10진 부동 소수점의 경우, 숫자의 정수 부분 절단은 허용되지 않으므로 오류가 발생합니다.

  부동 소수점 숫자의 경우, 언더플로우도 허용되지 않습니다. 0이 아닌 최상위 숫자가 제거되는 경우 1과 -1 사이의 숫자에 대해 언더플로우가 발생합니다. 10진 부동 소수점의 경우, 언더플로우가 허용되며 반올림 모드에 따라 결과는 경고와 함께 나타날 수 있는 0 또는 가장 작은 양수나 가장 큰 음수가 됩니다.

  표시기 변수를 사용하여 호스트 변수에 지정 시 오버플로우 또는 언더플로우가 발생하는 경우 오류 대신 오버플로우 또는 언더플로우 경고가 리턴됩니다. 이 경우, 호스트 변수에 숫자가 지정되지 않으며 표시기 변수는 음수 2로 설정됩니다.

10진 부동 소수점의 경우, CURRENT DECFLOAT ROUNDING MODE 특수 레지스터는 적용되는 반올림 모드를 표시합니다.

정수에 지정

10진수, 부동 소수점 또는 10진 부동 소수점 숫자가 정수 컬럼이나 변수에 지정되면 숫자의 소수 부분이 제거됩니다. 따라서 1과 -1 사이의 숫자는 0으로 감소합니다.

10진수에 지정

정수가 10진수 컬럼이나 변수에 지정되면, 숫자는 먼저 임시 10진수로 변환된 후 필요하면 목표의 정밀도와 스케일로 변환됩니다. 임시 10진수의 정밀도와 스케일은 작은 정수의 경우에는 5,0이고 대형 정수의 경우에는 11,0이며 큰 정수의 경우에는 19,0입니다.

소수가 소수 컬럼이나 변수에 지정되면 숫자는 필요한 경우 목표의 정밀도와 스케일로 변환됩니다. 필요한 수만큼 선행 영(0)이 추가되고, 10진수의 소수 부분에서는 필요한 수만큼 후행 영(0)이 추가되거나 필요한 수만큼 후행 숫자가 제거됩니다.

부동 소수점 숫자가 10진수 컬럼이나 변수에 지정되면, 숫자는 먼저 정밀도 31의 임시 10진수로 변환된 후 필요하면 목표의 정밀도 및 스케일로 절단됩니다. 이 변환에서 숫

자는 소수점 이하 자릿수가 31인 정밀도로 반올림(부동 소수점 산술 사용)됩니다. 따라서 10진수 컬럼이나 변수에 나타낼 수 있는 가장 큰 음수보다 크거나 가장 작은 양수보가 작은 1과 -1 사이의 숫자는 0으로 감소합니다. 스케일에는 유의값을 유실하지 않고 숫자의 정수 부분을 나타낼 수 있는 가장 큰 가능한 값이 제공됩니다.

10진 부동 소수점 숫자가 10진수 컬럼이나 변수에 지정되면, 숫자는 10진수 컬럼이나 변수의 정밀도와 스케일로 반올림됩니다. 따라서 10진수 컬럼이나 변수에 나타낼 수 있는 가장 큰 음수보다 크거나 가장 작은 양수보가 작은 1과 -1 사이의 숫자는 0으로 감소하거나, 반올림 모드에 따라 10진수 컬럼이나 변수에 나타낼 수 있는 가장 작은 양수 또는 가장 큰 음수로 반올림됩니다.

부동 소수점에 지정

부동 소수점 숫자는 실수로 된 근사값입니다. 그러므로 정수, 10진수, 부동 소수점 또는 10진 부동 소수점 숫자가 부동 소수점 컬럼이나 변수에 지정되면, 결과는 원래 숫자와 동일하지 않을 수도 있습니다. 숫자는 부동 소수점 산술을 사용하여 부동 소수점 컬럼이나 변수의 정밀도로 반올림됩니다. 10진 부동 소수점 값은 먼저 문자열 표현으로 변환된 후 부동 소수점 숫자로 변환됩니다.

10진 부동 소수점에 지정

정수가 10진 부동 소수점 컬럼이나 변수에 지정되면, 숫자는 먼저 임시 10진수로 변환된 후 10진 부동 소수점 숫자로 변환됩니다. 임시 10진수의 정밀도와 스케일은 작은 정수의 경우에는 5,0이고 대형 정수의 경우에는 11,0이며 큰 정수의 경우에는 19,0입니다. DECFLOAT(16) 컬럼이나 변수에 BIGINT를 지정하면 반올림이 발생할 수 있습니다.

10진수가 10진 부동 소수점 컬럼이나 변수에 지정되면, 숫자는 목표의 정밀도(16이나 34)로 변환됩니다. 선행 영(0)은 제거됩니다. 10진수의 정밀도와 스케일 및 목표의 정밀도에 따라 값이 반올림될 수 있습니다.

부동 소수점 숫자가 10진 부동 소수점 컬럼이나 변수에 지정되면, 숫자는 먼저 부동 소수점 숫자의 임시 문자열 표현으로 변환됩니다. 그런 다음, 숫자의 문자열 표현이 10진 부동 소수점으로 변환됩니다.

DECFLOAT(16) 숫자가 DECFLOAT(34) 컬럼이나 변수에 지정되면, 결과 값은 DECFLOAT(16) 숫자와 동일합니다.

DECFLOAT(34) 숫자가 DECFLOAT(16) 컬럼이나 변수에 지정되면, 결과 형식에서 소스의 지수가 대응하는 지수로 변환됩니다. DECFLOAT(34) 숫자의 가수가 목표의 정밀도로 반올림됩니다.

문자열에서 숫자로 지정

문자열이 숫자 데이터 유형에 지정되면, CAST 스펙 규칙을 사용하여 목표 숫자 데이터 유형으로 변환됩니다. 자세한 정보는 *SQL 참조서, 볼륨 1*의 『CAST 스펙』을 참조하십시오.

문자열 지정

지정 유형에는 다음 두 가지가 있습니다.

- *스토리지 지정*에서 값을 지정하고 유효 데이터를 절단하지 않습니다. 컬럼에 값을 지정할 경우를 예로 들 수 있습니다.
- *검색 지정*에서 값을 지정하고 절단이 허용됩니다. 데이터베이스에서 데이터를 검색할 경우를 예로 들 수 있습니다.

문자열 지정 규칙은 지정 유형에 따라 다릅니다.

스토리지 지정

기본 규칙은 목표에 지정된 문자열의 길이가 목표의 길이 속성보다 길 수 없다는 것입니다. 문자열의 길이가 목표 속성의 길이보다 긴 경우 다음 조치가 발생할 수 있습니다.

- 목표의 길이 속성에 맞게 (LOB 문자열을 제외한 모든 문자열에서) 뒤 공백을 절단하여 문자열을 지정합니다.
- 다음과 같은 경우에 오류가 리턴됩니다(SQLSTATE 22001).
  - LOB 문자열 이외의 문자열에서 비공백 문자가 절단되는 경우
  - LOB 문자열에서 임의의 문자(또는 바이트)가 절단되는 경우

고정 길이 목표에 문자열을 지정할 때 문자열의 길이가 목표의 길이 속성보다 작으면, 문자열의 오른쪽에 필요한 수의 1바이트, 2바이트 또는 UCS-2 공백이 채워집니다. 채워진 문자는 FOR BIT DATA 속성으로 정의된 컬럼의 경우에도 항상 공백입니다. UCS-2는 여러 등록 정보로 여러 공백 문자를 정의합니다. 유니코드 데이터베이스의 경우, 데이터베이스 관리 프로그램에서는 항상 x'0020' 위치에서 UCS-2 공백으로 ASCII SPACE를 사용합니다. EUC 데이터베이스의 경우, x'3000' 위치의 IDEOGRAPHIC SPACE는 GRAPHIC 문자열을 채우는 데 사용됩니다.

검색 지정

목표에 지정된 문자열의 길이는 목표의 길이 속성보다 길 수 있습니다. 목표에 문자열을 지정한 경우에 문자열의 길이가 목표의 길이 속성보다 길면, 문자열의 오른쪽에서 필요한 문자 수(또는 바이트)만큼 절단합니다. 절단이 발생하면 경고가 리턴되고 (SQLSTATE 01004) 값 'W'가 SQLCA의 SQLWARN1 필드에 지정됩니다.

또한 표시기 변수가 제공되었는데 값의 소스가 LOB이 아닌 경우, 표시기 변수가 문자열의 원래 길이로 설정됩니다.

고정 길이 목표에 문자열을 지정할 때 문자열의 길이가 목표의 길이 속성보다 작으면, 문자열의 오른쪽에 필요한 수의 1바이트, 2바이트 또는 UCS-2 공백이 채워집니다. 채워진 문자는 FOR BIT DATA 속성으로 정의된 문자열의 경우에도 항상 공백입니다. UCS-2는 여러 등록 정보로 여러 공백 문자를 정의합니다. 유니코드 데이터베이스의 경우, 데이터베이스 관리 프로그램에서는 항상 x'0020' 위치에서 UCS-2 공백으로 ASCII SPACE를 사용합니다. EUC 데이터베이스의 경우, x'3000' 위치의 IDEOGRAPHIC SPACE는 GRAPHIC 문자열을 채우는 데 사용됩니다.

C NUL 종료 호스트 변수의 검색 지정은 PREP 또는 BIND 명령에 지정된 기본 옵션에 따라 처리됩니다.

문자열 지정을 위한 변환 규칙

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

- 코드 페이지가 서로 다릅니다.
- 문자열이 널(NULL)도 아니고 비어 있지도 않습니다.
- 문자열에 코드 페이지 값 0(FOR BIT DATA)이 있습니다.

유니코드 데이터베이스의 경우, 문자열은 그래픽 컬럼에 지정될 수 있으며 그래픽 문자열은 문자열 컬럼에 지정될 수 있습니다.

문자열 지정을 위한 MBCS 고려사항

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

- 공백 채우기는 항상 1바이트 공백 문자(X'20')를 사용하여 수행됩니다.
- 공백 절단은 항상 1바이트 공백 문자(X'20')를 기준으로 수행됩니다. 2바이트 공백 문자는 절단과 관련하여 다른 문자와 같이 처리됩니다.
- 호스트 변수에 문자열을 지정하면 목표 호스트 변수가 전체 소스 문자열을 포함하기에 크기가 충분하지 않은 경우에는 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 나타냅니다. 문자열이 FOR BIT DATA로 정의될 경우에도 동일한 MBCS 문자 단편 처리가 적용된다는 점에 유의하십시오.

그래픽 문자열 지정을 위한 DBCS 고려사항

그래픽 문자열 지정은 문자열에 대한 방법과 같은 방법으로 수행됩니다. 비유니코드 데이터베이스의 경우, 그래픽 문자열 데이터 유형은 다른 그래픽 문자열 데이터 유형과만 호환되며, 숫자, 문자열 또는 날짜 시간 데이터 유형과는 호환되지 않습니다. 유니코드 데이터베이스의 경우, 그래픽 문자열 데이터 유형은 문자열 데이터 유형과 호환됩니다. 그러나 그래픽 및 문자열 데이터 유형을 SELECT INTO 또는 VALUES INTO문에서 서로 바꿔서 사용할 수 없습니다.

그래픽 문자열 값이 그래픽 문자열 컬럼에 지정되는 경우, 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 길이보다 짧은 경우, 목표에는 소스 문자열의 사본이 포함되며, 이 문자열의 오른쪽은 목표 길이와 같은 길이의 값을 작성하는 데 필요한 2바이트 공백 문자 수로 채워집니다.

그래픽 문자열 값이 그래픽 문자열 호스트 변수에 지정되고 소스 문자열의 길이가 호스트 변수의 길이보다 큰 경우, 호스트 변수에는 소스 문자열의 사본이 포함되며, 이 문자열은 호스트 변수의 길이와 같은 길이의 값을 작성하는 데 필요한 2바이트 문자가 오른쪽에서 절단됩니다. (이 경우, 2바이트 문자의 중간에서 절단하는 것은 고려할 필요가 없습니다. 단, 이와 같은 절단이 발생하는 경우, 소스 값이나 목표 호스트 변수는 모두 잘못 정의된 그래픽 문자열 데이터 유형이 됨에 유의하십시오.) SQLCA에서 경고 플래그 SQLWARN1가 'W'로 설정됩니다. 표시기 변수(지정된 경우)에는 소스 문자열의 원래 길이(2바이트 문자)가 포함되어 있습니다. 그러나 DBCLOB의 경우, 표시기 변수에는 원래 길이가 없습니다.

C NULL 종료 호스트 변수(wchart_t를 이용하여 선언된)의 검색 지정은 PREP 또는 BIND 명령으로 지정된 옵션에 따라 처리됩니다.

숫자에서 문자열로 지정

숫자가 문자열 데이터 유형에 지정되면, CAST 스펙 규칙을 사용하여 목표 문자열 데이터 유형으로 변환됩니다. 자세한 정보는 *SQL 참조서, 볼륨 1*의 『CAST 스펙』을 참조하십시오.

숫자 값을 문자 또는 그래픽 데이터 유형으로 캐스트하는 동안 비공백 문자가 절단되면, 경고가 리턴됩니다. 이 절단 동작은 스토리지 지정 규칙을 따르는 문자 또는 그래픽 데이터 유형에 지정하는 것과 다릅니다. 지정 중에 비공백 문자가 절단되는 경우 오류가 리턴됩니다.

날짜 시간 지정

TIME 컬럼이나 문자열 변수 또는 문자열 컬럼에만 TIME 값을 지정할 수 있습니다.

DATE는 DATE, TIMESTAMP 또는 문자열 데이터 유형에 지정될 수 있습니다. DATE 값이 TIMESTAMP 데이터 유형에 지정된 경우, 누락 시간 정보는 모두 영(0)으로 가정됩니다.

TIMESTAMP 값은 DATE, TIME, TIMESTAMP 또는 문자열 데이터 유형으로 지정될 수 있습니다. TIMESTAMP 값이 DATE 데이터 유형에 지정되면, 날짜 부분이 추출되고 시간 부분이 절단됩니다. TIMESTAMP 값이 TIME 데이터 유형에 지정되면 날짜 부분이 무시되고 시간 부분이 추출되지만 소수 초가 절단됩니다. TIMESTAMP 값이 낮은 정밀도인 TIMESTAMP로 지정되면, 초과 소수 초가 절단됩니다. TIMESTAMP 값이 높은 정밀도인 TIMESTAMP로 지정되면 누락 숫자가 영(0)으로 가정됩니다.

CLOB, DBCLOB 또는 BLOB 변수나 컬럼에 지정하면 안됩니다.

날짜 시간 값이 문자열 변수나 문자열 컬럼에 지정되면 문자열 표현으로의 변환이 자동으로 수행됩니다. 선행 0은 날짜, 시간 또는 시간소인 부분에서 생략되지 않습니다. 목표의 필요한 길이는 문자열 표현의 형식에 따라 다릅니다. 목표 길이가 필요한 길이보다 크고 목표가 고정 길이 문자열인 경우, 오른쪽이 공백으로 채워집니다. 목표의 길이가 필요한 길이보다 적은 경우, 결과는 포함된 날짜 시간 값의 유형과 목표의 유형에 따라 다릅니다.

목표가 호스트 변수가 아니고 문자 데이터 유형을 가진 경우, 절단이 허용되지 않습니다. 컬럼의 길이 속성은 최소한 날짜의 경우 10, 시간의 경우 8, TIMESTAMP(0)의 경우 19, TIMESTAMP(*p*)의 경우 20+*p*여야 합니다.

목표가 문자열 호스트 변수이면 다음과 같은 규칙이 적용됩니다.

- **DATE의 경우:** 호스트 변수의 길이가 10자 미만이면 오류가 리턴됩니다.
- **TIME의 경우:** USA 형식을 사용하는 경우, 호스트 변수의 길이는 8자 이상이어야 합니다. 다른 형식에서는 길이가 5자 이상이어야 합니다.

  ISO나 JIS 형식을 사용하는 경우와 호스트 변수의 길이가 8자 미만인 경우, 시간의 두 번째 부분은 결과에서 생략되어 표시기 변수(제공되는 경우)에 지정됩니다. SQLCA의 SQLWARN1 필드는 생략을 나타내도록 설정됩니다.
- **TIMESTAMP의 경우:** 호스트 변수의 길이가 19자 미만이면 오류가 리턴됩니다. 길이가 32자 미만이지만 19자 이상인 경우, 값의 소수 초 부분의 뒤에 오는 숫자가 생략됩니다. SQLCA의 SQLWARN1 필드는 생략을 표시하도록 설정됩니다.

DATE가 TIMESTAMP로 지정되면 시간소인의 시간 및 소수 구성요소가 각각 자정과 0으로 설정됩니다. TIMESTAMP가 DATE로 지정되면 날짜 부분은 추출되고 시간 및 소수 구성요소는 절단됩니다.

TIMESTAMP가 TIME으로 지정되면 DATE 부분은 무시되고 소수 구성요소는 절단됩니다.

XML 지정

XML 지정의 일반 규칙은 XML 값만을 XML 컬럼 또는 XML 변수에 지정할 수 있다는 것입니다. 이 규칙에 다음과 같은 예외가 있습니다.

- ***입력 XML 호스트 변수 처리:*** 호스트 변수가 문자열 값을 기반으로 하기 때문에 XML 지정 값의 특수한 경우입니다. SQL 내에서 XML에 지정을 수행하기 위해, CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 설정을 사용하여 문자열 값을 XML 값으로 내재적으로 구문 분석합니다. 이는 호스트 변수가 불필요한 공백을 항상 스트리핑하는 XMLVALIDATE 함수의 인수가 아닌 경우, 공백을 보존할 것인지 스트리핑할 것인지 여부를 결정합니다.
- ***데이터 유형 XML의 입력 매개변수 표시기에 문자열 지정:*** 입력 매개변수 표시기에 XML의 내재적 또는 명시적 데이터 유형이 있을 경우, 해당 매개변수 표시기에 바인드되는(지정되는) 값은 문자열 변수, 그래픽 문자열 변수 또는 실행 파일 문자열 변수일 수 있습니다. 이 경우 CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 설정을 사용하여 매개변수 표시기가 불필요한 공백을 항상 스트리핑하는 XMLVALIDATE 함수의 인수가 아닌 경우, 공백을 보존할 것인지 또는 스트리핑할 것인지 여부를 결정하는 XML 값으로 문자열 값이 내재적으로 구문 분석됩니다.
- ***데이터 변경 명령문에서 XML 컬럼에 직접 문자열 지정:*** 데이터 변경 명령문에서 XML 유형의 컬럼에 직접 지정할 경우, 지정된 표현식은 문자열 또는 실행 파일 문자열일 수 있습니다. 이 경우, `XMLPARSE (DOCUMENT` *`expression`* `STRIP WHITESPACE)`의 결과가 목표 컬럼에 지정됩니다. 지원되는 문자열 데이터 유형은 XMLPARSE 함수에 대한 지원되는 인수에 의해 정의됩니다. 이러한 XML 지정 예외에 의해 문자 또는 실행 파일 문자열 값을 SQL 변수 또는 XML 데이터 유형의 SQL 매개변수에 지정하도록 허용되지 않음을 주의하십시오.
- ***검색 시 문자열에 XML 지정:*** Embedded SQL에서 FETCH INTO문 또는 EXECUTE INTO문을 사용하여 XML 값을 호스트 변수로 검색할 경우, 호스트 변수의 데이터 유형은 CLOB, DBCLOB 또는 BLOB일 수 있습니다. 다른 API(예: CLI, JDBC 또는 .NET)를 사용할 경우, API에서 지원하는 문자, 그래픽 또는 실행 파일 문자열 유형으로 XML 값을 검색할 수 있습니다. 이러한 모든 경우에서 XML 값은 UTF-8로 인코딩된 문자열로 내재적으로 변환되며, 문자 또는 그래픽 문자열의 경우 클라이언트 코드 페이지로 변환됩니다.

문자열 또는 실행 파일 문자열 값을 XML 호스트 변수로 검색할 수 없습니다. XML 호스트 변수의 값을 문자열 데이터 유형 또는 실행 파일 문자열 데이터 유형의 컬럼, SQL 변수 또는 SQL 매개변수에 지정할 수 없습니다.

인라인된 SQL 본문 UDF 및 SQL 프로시저에서 XML 매개변수 및 변수에 대한 지정은 참조별로 수행됩니다. XML 데이터 유형의 매개변수를 전달하여 인라인된 SQL UDF

또는 SQL 프로시저를 호출하는 것도 참조별로 수행됩니다. XML 값이 참조에 의해 전달되는 경우, 입력 노드 트리가 직접 사용됩니다. 이러한 직접 사용은 문서 순서, 원래 노드 ID 및 모든 상위 등록 정보를 포함하여 모든 등록 정보를 보존합니다.

사용자 정의 유형 지정

구별 유형 및 구조화된 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.

- 이 구별 유형의 소스 유형을 이 호스트 변수에 지정할 수 있는 경우 지정의 오른쪽에 있는 구별 유형 값은 왼쪽에 있는 호스트 변수에 지정될 수 있습니다.

지정의 목표가 구별 유형을 기초로 하는 컬럼인 경우, 소스 데이터 유형은 목표 데이터 유형으로 캐스트할 수 있어야 합니다.

구조화된 유형: 호스트 변수 간의 지정은 호스트 변수의 선언된 유형에 따라 다릅니다. 즉, 규칙을 준수합니다.

- 호스트 변수의 선언된 유형이 구조화된 유형이거나 구조화된 유형의 슈퍼 유형인 경우 지정의 오른쪽에 있는 구조화된 유형 값은 왼쪽에 있는 호스트 변수에 지정될 수 있습니다.

지정 목표가 구조화된 유형 컬럼인 경우, 소스 데이터 유형은 목표 데이터 유형이나 목표 데이터 유형의 슈퍼 유형이어야 합니다.

배열 유형의 경우, 다른 규칙이 SQL 변수 및 매개변수에 대한 지정에 적용됩니다. SQL 변수 또는 매개변수에 대한 지정의 타당성은 다음 규칙에 따라 판별됩니다.

- 지정 오른쪽이 SQL 변수 또는 매개변수, TRIM_ARRAY 함수의 호출, ARRAY_DELETE 함수의 호출 또는 CAST 표현식이면 해당 유형은 지정의 왼쪽에 있는 SQL 변수 또는 매개변수와 유형과 동일해야 합니다.
- 지정의 오른쪽이 배열 컨스트럭터 또는 ARRAY_AGG 함수의 호출이면, 이는 왼쪽에 있는 SQL 변수 또는 매개변수의 유형으로 내재적으로 캐스트됩니다.

예를 들어, 변수 *V*의 유형이 MYARRAY라고 가정하면 다음 명령문은

```
SET V = ARRAY[1,2,3];
```

다음과 같습니다.

```
SET V = CAST(ARRAY[1,2,3] AS MYARRAY);
```

다음 명령문은

```
SELECT ARRAY_AGG(C1) INTO V FROM T
```

다음과 같습니다.

```
SELECT CAST(ARRAY_AGG(C1) AS MYARRAY) INTO V FROM T
```

특정 사용자 정의 유형에 대한 추가 정보는 다음 절에 있습니다.

배열 유형 지정

배열 요소의 값은 배열 요소의 데이터 유형에 지정할 수 있어야 합니다. 해당 데이터 유형에 대한 지정 규칙이 값 지정에 적용됩니다. 배열의 인덱스에 대해 지정되는 값을 배열에 대한 인덱스의 데이터 유형에 지정할 수 있어야 합니다. 해당 데이터 유형에 대한 지정 규칙이 값 지정에 적용됩니다. 일반 배열의 경우 인덱스 데이터 유형이 INTEGER이고 연관된 배열의 경우 데이터 유형이 INTEGER 또는 VARCHAR(n)입니다. 여기서 n은 VARCHAR 데이터 유형에 대해 유효한 모든 길이 속성입니다. 일반 배열에 대한 지정의 인덱스 값이 배열의 현재 카디낼리티(cardinality)보다 큰 경우, 값이 INTEGER 데이터 유형에 대한 최대값을 초과하지 않으면 배열의 카디낼리티는 새 인덱스 값으로 늘어납니다. 연관된 배열에 하나의 새 요소를 지정하면 인덱스 값은 산재할 수 있으므로 카디낼리티(cardinality)가 정확하게 1만큼 늘어납니다.

다음은 배열 유형 값을 포함하는 유효한 지정입니다.

- 소스 변수와 동일한 배열 유형을 갖는 다른 배열 변수에 대한 배열 변수 지정.
- 배열 변수에 대한 배열 유형의 표현식 지정. 소스 표현식의 배열 요소 유형은 목표 배열 변수의 배열 요소 유형에 지정할 수 있습니다.

행 유형 지정

행 변수에서 필드에 대한 지정은 필드 자체가 필드와 동일한 데이터 유형의 변수인 것처럼 동일한 규칙을 따라야 합니다. 행 변수는 동일한 사용자 정의 행 유형이 있는 행 변수에만 지정될 수 있습니다. FETCH, SELECT 또는 VALUES INTO를 사용하여 행 변수에 값을 지정하는 경우, 소스 값 유형을 목표 행 필드에 지정할 수 있어야 합니다.지정의 소스 또는 목표 변수(또는 둘 다)가 테이블이나 뷰의 행에 앵커되는 경우, 필드 수는 동일해야 하며 소스 값의 필드 유형을 목표 값의 필드 유형에 지정할 수 있어야 합니다.

커서 유형 지정

커서에 지정은 커서 유형에 종속됩니다. 내장 유형 CURSOR의 변수 또는 매개변수에 지정할 수 있는 값은 다음과 같습니다.

- 커서 값 컨스트럭터
- 내장 유형 CURSOR 값
- 사용자 정의 커서 유형 값

약하게 유형이 지정된 사용자 정의 커서 유형의 변수 또는 매개변수에 지정할 수 있는 값은 다음과 같습니다.

- 커서 값 컨스트럭터
- 내장 유형 CURSOR 값
- 동일한 유형 이름이 있는 사용자 정의 커서 유형 값

명백히 유형이 지정된 사용자 정의 커서 유형의 변수 또는 매개변수에 지정할 수 있는 값은 다음과 같습니다.

- 커서 값 컨스트럭터
- 동일한 유형 이름이 있는 사용자 정의 커서 유형 값

부울 유형 지정

내장 유형 BOOLEAN의 변수, 매개변수 또는 리턴 유형에 지정할 수 있는 시스템 정의 값은 다음과 같습니다.

- TRUE
- FALSE
- NULL

검색 조건의 평가 결과도 지정할 수 있습니다. 검색 조건이 알 수 없음으로 평가되면 널(NULL)의 값이 지정됩니다.

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 슈퍼 유형입니다. 범위가 지정된 참조 컬럼이나 변수에 지정이 수행될 경우, 지정되는 실제 값이 범위에 의해 정의된 목표 테이블이나 뷰에 존재하는지 확인하는 점검은 수행되지 않습니다.

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

- 이 참조 유형의 표현 유형이 이 호스트 변수에 지정 가능한 경우 지정의 오른쪽에 있는 참조 유형 값은 왼쪽에 있는 호스트 변수에 지정될 수 있습니다.

지정 목표가 컬럼이고 지정의 오른쪽에 호스트 변수가 있는 경우, 호스트 변수는 목표 컬럼의 참조 유형으로 명시적으로 캐스트되어야 합니다.

숫자 비교

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

한 숫자가 정수이고 나머지는 소수인 경우, 소수로 변환된 정수의 임시 사본과 비교가 이뤄집니다.

스케일이 다른 소수를 비교할 때 분수 부분의 자릿수가 다른 숫자와 같은 자릿수가 되도록 후행 0을 추가한 숫자의 임시 사본과 비교가 이뤄집니다.

한 숫자가 부동 소수점이고 나머지는 정수이거나 소수인 경우, 배정밀도 소수점으로 변환된 다른 숫자의 임시 사본과 비교가 이뤄집니다.

두 개의 부동 소수점 숫자는 이들의 정규화된 형식의 비트 구성이 동일할 때에만 같습니다.

한 숫자는 10진 부동 소수점이고 다른 숫자는 정수, 10진수, 단정밀도 부동 소수점 또는 배정밀도 부동 소수점인 경우, 10진 부동 소수점으로 변환된 다른 숫자의 임시 사본과 비교가 이뤄집니다.

한 숫자는 DECFLOAT(16)이고 다른 숫자는 DECFLOAT(34)이면, 비교가 이뤄지기 전에 DECFLOAT(16) 값이 DECFLOAT(34)로 변환됩니다.

10진 부동 소수점 데이터 유형은 양수 및 음수 영(0)을 둘 다 지원합니다. 양수 및 음수 영(0)은 서로 다른 2진 표현을 갖지만 =(같음) 술어는 음수 및 양수 영(0) 비교에 대해 참을 리턴합니다.

COMPARE_DECFLOAT 및 TOTALORDER 스칼라 함수를 사용하여 예를 들어 2.0 <> 2.00 비교가 필요한 경우 2진 레벨에서 비교를 수행할 수 있습니다.

10진 부동 소수점 데이터 유형은 음수 및 양수 NaN(quiet 및 signalling), 음수 및 양수 infinity 스펙을 지원합니다. SQL 관점에서 INFINITY = INFINITY, NAN = NAN, SNAN = SNAN, -0 = 0입니다.

특수 값 비교 및 순서지정 규칙은 다음과 같습니다.

- (+/-) INFINITY 비교는 동일 기호의 (+/-) INFINITY인 경우에만 같습니다.
- (+/-) NAN 비교는 동일 기호의 (+/-) NAN인 경우에만 같습니다.
- (+/-) SNAN 비교는 동일 기호의 (+/-) SNAN인 경우에만 같습니다.

여러 특수 값 사이의 순서 지정은 다음과 같습니다.

- -NAN < -SNAN < -INFINITY < 0 < INFINITY < SNAN < NAN

문자열 및 숫자 데이터 유형을 비교하는 경우, CAST 스펙 규칙을 사용하여 문자열이 DECFLOAT(34)로 캐스트됩니다. 자세한 정보는 *SQL 참조서, 볼륨 1*의 『CAST 스펙』을 참조하십시오.문자열은 숫자의 유효한 문자열 표현을 포함해야 합니다.

문자열 비교

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

길이가 다른 문자열을 비교할 경우, 짧은 문자열의 논리적 사본을 사용하여 비교가 이루어지며 짧은 문자열에는 긴 문자열의 길이에 맞도록 길이를 늘리기 위해 오른쪽이 해당되는 길이만큼 단일 바이트 공백으로 채워집니다. 논리 확장은 FOR BIT DATA로 태그가 붙어 있는 문자열을 포함한 모든 문자열에 대해 수행됩니다.

문자열(FOR BIT DATA로 태그가 붙어 있는 문자열은 제외)은 데이터베이스 작성시 지정되는 조합 시퀀스에 따라 비교됩니다. 예를 들어, 데이터베이스 관리 프로그램에서 제공하는 디폴트 조합 시퀀스는 같은 문자, 같은 가중치의 소문자 및 대문자 버전을 제공할 수도 있습니다. 데이터베이스 관리 프로그램은 두 단계의 비교를 수행하여 하나의 동일 문자열만 같은 것으로 간주되도록 합니다. 첫 단계에서 문자열은 데이터베이스 조합 시퀀스에 따라 비교됩니다. 문자열에 있는 문자의 가중치가 같은 경우, 두 번째 ″타이-브레이커(tie-breaker)″ 단계가 수행되어 이들의 실제 코드 포인트 값을 기준으로 문자열을 비교합니다.

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

LOB 문자열은 기본 비교 연산자(=, <>, <, >, <= 및 >=)를 사용하는 비교 연산에서는 지원되지 않습니다. LIKE 술어와 POSSTR 함수를 사용하는 비교에서는 지원됩니다.

문자열 분할 영역은 SUBSTR 및 VARCHAR 스칼라 함수를 사용하여 비교할 수 있습니다. 예를 들어, 다음 컬럼이 제공되는 경우,

```
    MY_SHORT_CLOB   CLOB(300)
MY_LONG_VAR     VARCHAR(8000)
```

다음은 유효합니다.

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

예를 들면, 다음과 같습니다.

이들 예의 경우, ‘A’, ‘Á’, ‘a’ 및 ‘á’에는 각각 코드 포인트 값 X‘41’, X‘C1’, X‘61’ 및 X‘E1’이 있습니다.

문자 ‘A’, ‘Á’, ‘a’, ‘á’의 가중치가 136, 139, 135 및 138인 조합 시퀀스를 고려해 보십시오. 문자는 다음과 같이 가중치순으로 정렬됩니다.

```
'a' < 'A' < 'á' < 'Á'
```

이제 코드 포인트가 각각 0xC141, 0xC161, 0xE141 및 0xE161인 네 개의 DBCS 문자 D1, D2, D3 및 D4를 고려하십시오. 이들 DBCS 문자가 CHAR 컬럼에 있는 경우, 이러한 바이트의 조합 가중치에 따라 바이트의 시퀀스로 정렬됩니다. 첫 번째 바이트는 138과 139의 가중치를 가지며, 따라서 D3과 D4가 D2와 D1 앞에 옵니다. 두 번째 바이트는 135와 136의 가중치를 가집니다. 그러므로 순서는 다음과 같습니다.

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

그러나 비교되는 값의 속성이 FOR BIT DATA이거나 이들 DBCS 문자가 GRAPHIC 컬럼에 저장된 경우, 조합 가중치는 무시되고, 문자는 다음과 같이 코드 포인트에 따라 비교됩니다.

```
'A' < 'a' < 'Á' < 'á'
```

DBCS 문자는 다음과 같이 코드 포인트 순서대로 바이트 시퀀스로 정렬됩니다.

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

문자 'A', 'Á', 'a', 'á'의 가중치 74, 75, 74 및 75(고유하지 않음)인 조합 시퀀스를 고려하십시오. 조합 가중치만을 고려하면(첫 번째 단계) 'a'는 'A'와 같고, 'á'는 'Á'와 같습니다. 문자의 코드 포인트는 다음과 같이 타이 브레이크(tie break)(두 번째 단계)에 사용됩니다.

```
'A' < 'a' < 'Á' < 'á'
```

CHAR 컬럼에 있는 DBCS 문자는 가중치에 따라(첫 번째 단계), 그런 후 타이 브레이크를 위한(두 번째 단계) 코드 포인트에 따라 바이트의 시퀀스를 정렬합니다. 첫 번째 바이트는 동일한 가중치를 가지므로, 코드 포인트(0xC1 및 0xE1)에 의해 타이 브레이크가 이뤄집니다. 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 DATA이거나 DBCS 문자가 GRAPHIC 컬럼에 저장된 경우, 조합 가중치는 무시되고, 문자는 다음과 같이 코드 포인트에 따라 비교됩니다.

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

이 특정 예의 경우, 조합 가중치를 사용할 때와 결과가 동일해질 수 있습니다. 그렇지만 항상 그렇지는 않았습니다.

비교를 위한 변환 규칙

두 문자열을 비교할 때 필요에 따라 문자열 중 하나가 다른 문자열의 코드화 체계 및 코드 페이지로 먼저 변환됩니다.

결과 순서

정렬해야 하는 결과는 140 페이지의 『문자열 비교』에 설명된 문자열 비교 규칙에 따라 정렬됩니다. 비교는 데이터베이스 서버에서 수행됩니다. 클라이언트 응용프로그램 결과를 리턴할 때 코드 페이지 변환이 수행될 수 있습니다. 이러한 후속 코드 페이지 변환은 서버가 결정한 결과 세트의 순서에 영향을 미치지 않습니다.

문자열 비교를 위한 MBCS 고려사항

혼합 SBCS/MBCS 문자열은 데이터베이스 작성시 지정된 조합 시퀀스에 따라 비교됩니다. 디폴트(SYSTEM) 조합 시퀀스로 작성된 데이터베이스의 경우, 모든 1바이트의 ASCII 문자는 올바른 시퀀스로 정렬되나 2바이트 문자는 반드시 코드 포인트 순서로 정렬되지는 않습니다. IDENTITY 시퀀스로 작성된 데이터베이스의 경우, 모든 2바이트 문자 세트도 해당 코드 포인트 순서대로 제대로 정렬되지만 1바이트 ASCII 문자도 해당 코드 포인트 순서대로 정렬됩니다. COMPATIBILITY 시퀀스로 작성된 데이터베이스에서 대부분의 2바이트 문자를 제대로 정렬하는 데는 절충 순서가 사용됩니다. 이 순서는 ASCII에 대해서도 거의 정확합니다. 이는 DB2 버전 2의 디폴트 조합 테이블입니다.

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

예:

이 예의 경우, 'A', 'B', 'a' 및 'b' 2바이트 문자의 코드 포인트 값은 각각 X'8260', X'8261', X'8281' 및 X'8282'입니다.

X'8260', X'8261', X'8281' 및 X'8282' 코드 포인트의 가중치가 96, 65, 193 및 194인 조합 시퀀스를 고려하십시오. 다음과 같습니다.

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

및

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

그래픽 문자열 비교는 문자열에 대한 방법과 같은 방법으로 수행됩니다.

그래픽 문자열 비교는 DBCLOB를 제외한 모든 그래픽 문자열 데이터 유형 간에 유효합니다.

그래픽 문자열의 경우, 데이터베이스의 조합 시퀀스는 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'
```

및

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

길이가 다른 그래픽 문자열을 비교할 때 Long 문자열에 맞추기 위해 오른쪽에 2바이트 공백 문자로 채운 Short 문자열의 논리 사본을 사용하여 비교합니다.

두 그래픽 값이 모두 비어 있거나 모든 해당 그래픽이 모두 같으면 두 그래픽 값은 같습니다. 두 피연산자 중 하나가 널(NULL)인 경우, 결과는 알 수 없습니다. 두 값이 같지 않은 경우, 이들의 관계는 간단한 실행 파일 문자열 비교에 의해 결정됩니다.

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

날짜 시간 비교

DATE, TIME 또는 TIMESTAMP 값은 같은 데이터 유형의 다른 값이나 같은 데이터 유형의 문자열 표현과 비교됩니다. 또한 DATE 또는 날짜의 문자열 표현을 TIMESTAMP와 비교할 수 있습니다. 이 경우, 날짜 값에 대한 누락된 시간 정보는 모든 영(0)으로 가정됩니다. 모든 비교는 연대순으로 진행됩니다. 즉, 0001년 1월 1일에서 멀어질수록 시간의 값이 커집니다.

TIME 값과 시간 값의 문자열 표현이 포함된 비교에는 언제나 초가 포함됩니다. 문자열 표현에서 초가 누락된 경우, 0초를 의미합니다.

TIMESTAMP 값이 포함된 비교는 그 표현이 같은 것으로 간주될 수 있는지에 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 비교할 수 있습니다. 사용자 정의 구별 유형은 WITH COMPARISONS절을 사용하여 정의되어 있어야 합니다.

예:

다음과 같은 YOUTH 구별 유형과 CAMP_DB2_ROSTER 테이블이 제공될 경우,

```
CREATE 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 간에 캐스트는 함수 또는 CAST 스펙을 사용하여 ATTENDEE_NUMBER와 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)
```

사용자 정의 구조화된 유형의 값은 다른 값과 비교할 수 없습니다. NULL 술어와 TYPE 술어는 사용할 수 있습니다.

배열 유형 값의 비교는 지원되지 않습니다. 배열의 요소는 요소의 데이터 유형에 대한 비교 규칙에 따라서 비교할 수 있습니다.

행 유형 비교

행 유형 이름이 동일한 경우에도 행 변수를 다른 행 변수와 비교할 수 없습니다. 행 유형의 개별 필드를 다른 값과 비교할 수 있으며 필드의 데이터 유형에 대한 비교 규칙이 적용됩니다.

커서 유형 비교

커서 유형 이름이 동일하더라도 커서 변수는 다른 커서 변수와 비교될 수 없습니다.

부울 유형 비교

부울 값은 부울 리터럴 값으로 비교할 수 있습니다. 참 값이 거짓 값보다 큽니다.

참조 유형 비교

참조 유형 값은 목표 유형에 공통 슈퍼 유형이 있는 경우에만 비교가 가능합니다. 공통 슈퍼 유형의 스키마 이름이 SQL 경로에 포함되어 있는 경우에만 적절한 비교 함수를 찾을 수 있습니다. 비교는 참조 유형의 표현 유형을 사용하여 수행됩니다. 비교에서 참조 범위는 고려되지 않습니다.

비유니코드 데이터베이스에서 XML 비교

비유니코드 데이터베이스에서 비교가 수행되는 경우, XML 데이터와 문자 또는 그래픽 문자열 값 간의 비교에는 비교되는 두 세트의 데이터 중 하나의 코드 페이지를 변환해야 합니다. SQL 또는 XQuery문에서 쿼리 술어 또는 문자 또는 그래픽 문자열 데이터 유형을 가진 호스트 변수로 사용되는 문자 또는 그래픽 값은 비교 전에 데이터베이스 코드 페이지로 변환됩니다. 이 데이터에 포함된 문자가 데이터베이스 코드 페이지의 일부가 아닌 코드 포인트를 갖는 경우, 해당 위치에 대체 문자가 추가되어 예기치 않은 쿼리 결과를 초래할 가능성이 있습니다.

예를 들어, UTF-8 코드 페이지를 갖는 클라이언트는 그리스어 인코딩 ISO8859-7을 사용하여 작성된 데이터베이스 서버에 연결하는 데 사용됩니다. $\Sigma_G\Sigma_M$ 표현식은 XQuery 문의 술어로 송신됩니다(여기서, Σ_G는 유니코드로 된 그리스어 시그마 문자(U+03A3)를 나타내고 Σ_M은 유니코드로 된 수학 기호 시그마(U+2211)를 나타냄). ″Σ″ 문자가 둘 다 그리스어 데이터베이스 코드 페이지(0xD3)의 시그마에 대한 해당 코드 포인트로 변환되도록 이 표현식은 먼저 데이터베이스 코드 페이지로 변환됩니다. 이 코드 포인트를 Σ_A로 표시할 수 있습니다. 그런 다음, 새로 변환된 표현식 $\Sigma_A\Sigma_A$가 목표 XML 데이터와의 비교를 위해 다시 UTF-8로 변환됩니다. 이 두 코드 포인트 간 구별이 데이터베이스로 술어 표현식을 전달하는 데 필요한 코드 페이지 변환의 결과로 유실되었기 때문에 처음 두 개의 구별 값 Σ_G와 Σ_M이 $\Sigma_G\Sigma_G$ 표현식으로 XML 구문 분석기로 전달됩니다. XML 문서의 $\Sigma_G\Sigma_M$ 값과 비교 시 이 표현식은 일치에 실패합니다.

코드 페이지 변환 문제로 인해 발생할 수 있는 예기치 않은 쿼리 결과를 방지하는 한 가지 방법은 쿼리 표현식에서 사용되는 모든 문자가 데이터베이스 코드 페이지에 일치하는 코드 포인트를 갖게 하는 것입니다. 일치하는 코드 포인트가 없는 문자를 유니코드 문자 엔티티 참조를 사용하여 포함시킬 수 있습니다. 문자 엔티티 참조는 항상 코드 페이지 변환을 무시합니다. 예를 들어, Σ_M 문자를 대신하여 ࢣ 문자 엔티티 참조를 사용하면 데이터베이스 코드 페이지에 관계없이 올바른 유니코드 코드 포인트가 비교에 사용됩니다.

결과 데이터 유형 규칙

결과 데이터 유형은 연산의 피연산자에 적용되는 규칙에 따라 결정됩니다. 이 절에서는 다음 규칙에 대해 설명합니다.

이러한 규칙이 다음에 적용됩니다.

- 집합 연산(UNION, INTERSECT 및 EXCEPT)의 fullselect에 있는 해당 컬럼
- CASE 표현식 및 DECODE 스칼라 함수의 결과 표현식
- 스칼라 함수 COALESCE(또한 NVL 및 VALUE)의 인수
- 스칼라 함수 GREATEST, LEAST, MAX 및 MIN의 인수
- IN 술어 목록에 있는 표현식 값
- 복수 행 VALUES 절의 해당 표현식
- 배열 컨스트럭터에 있는 요소의 표현식 값
- BETWEEN 술어의 인수(모든 피연산자의 데이터 유형이 숫자인 경우 제외)
- OLAP 스펙에서 집계 그룹 범위에 대한 인수

이러한 규칙은 다양한 연산의 Long 문자열과 LOB 문자열에 대한 기타 제한사항에도 적용됩니다.

다양한 데이터 유형과 관련된 규칙은 다음과 같습니다. 경우에 따라, 가능한 결과 데이터 유형을 표시하는 데 테이블이 사용됩니다. LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다.

이러한 테이블은 적용 가능한 길이 또는 정밀도와 스케일을 포함하여 결과의 데이터 유형을 식별합니다. 결과 유형은 피연산자에 따라 결정됩니다. 두 쌍 이상의 피연산자가 있는 경우, 첫 번째 쌍부터 시작하십시오. 그러면 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 들어, 다음과 같은 상황을 고려하십시오.

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

첫 번째 쌍의 결과는 CHAR(4)입니다. 결과 값은 항상 4바이트입니다. 최종 결과 유형은 VARCHAR(4)입니다. 첫 번째 UNION 연산 결과 값의 길이는 항상 4가 됩니다.

문자열

문자열 값은 다른 문자열 값과 호환 가능합니다. 문자열에는 CHAR, VARCHAR 및 CLOB 데이터 유형이 포함됩니다.

하나의 피연산자	다른 피연산자	결과 데이터 유형
CHAR(x)	CHAR(y)	CHAR(z). 여기서, z = max(x,y)
CHAR(x)	VARCHAR(y)	VARCHAR(z). 여기서, z = max(x,y)
VARCHAR(x)	CHAR(y) 또는 VARCHAR(y)	VARCHAR(z). 여기서, z = max(x,y)
CLOB(x)	CHAR(y), VARCHAR(y) 또는 CLOB(y)	CLOB(z). 여기서, z = max(x,y)

결과 문자열의 코드 페이지는 문자열 변환 규칙에 따라 파생됩니다.

그래픽 문자열

그래픽 문자열 값은 다른 그래픽 문자열 값과 호환 가능합니다. 그래픽 문자열에는 GRAPHIC, VARGRAPHIC 및 DBCLOB 데이터 유형이 포함됩니다.

하나의 피연산자	다른 피연산자	결과 데이터 유형
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z). 여기서, z = max(x,y)
VARGRAPHIC(x)	GRAPHIC(y) 또는 VARGRAPHIC(y)	VARGRAPHIC(z). 여기서, z = max(x,y)
DBCLOB(x)	GRAPHIC(y), VARGRAPHIC(y) 또는 DBCLOB(y)	DBCLOB(z). 여기서, z = max (x,y)

결과 그래픽 문자열의 코드 페이지는 문자열 변환 규칙에 따라 파생됩니다.

유니코드 데이터베이스의 문자열 및 그래픽 문자열

유니코드 데이터베이스에서 문자열 값은 그래픽 문자열 값과 호환 가능합니다.

하나의 피연산자	다른 피연산자	결과 데이터 유형
GRAPHIC(x)	CHAR(y) 또는 GRAPHIC(y)	GRAPHIC(z). 여기서, z = max(x,y)
VARGRAPHIC(x)	CHAR(y) 또는 VARCHAR(y)	VARGRAPHIC(z). 여기서, z = max(x,y)
VARCHAR(x)	GRAPHIC(y) 또는 VARGRAPHIC	VARGRAPHIC(z). 여기서, z = max(x,y)
DBCLOB(x)	CHAR(y) 또는 VARCHAR(y) 또는 CLOB(y)	DBCLOB(z). 여기서, z = max (x,y)
CLOB(x)	GRAPHIC(y) 또는 VARGRAPHIC(y)	DBCLOB(z). 여기서, z = max (x,y)

실행 파일 대형 오브젝트(BLOB)

실행 파일 문자열(BLOB) 값은 다른 실행 파일 문자열(BLOB) 값과만 호환 가능합니다. BLOB 유형으로 취급해야 할 경우, BLOB 스칼라 함수를 사용하여 다른 유형에서 캐스트할 수 있습니다. 결과 BLOB의 길이는 모든 데이터 유형의 최대 길이입니다.

숫자

숫자 유형은 다른 숫자 데이터 유형, 문자열 데이터 유형(CLOB 제외)과 호환 가능하며, 유니코드 데이터베이스에서는 그래픽 문자열 데이터 유형(DBCLOB 제외)과 호환 가능합니다. 숫자 유형에는 SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE 및 DECFLOAT가 포함됩니다.

하나의 피연산자	다른 피연산자	결과 데이터 유형
SMALLINT	SMALLINT	SMALLINT
SMALLINT	문자열	DECFLOAT(34)
INTEGER	SMALLINT 또는 INTEGER	INTEGER
INTEGER	문자열	DECFLOAT(34)
BIGINT	SMALLINT, INTEGER 또는 BIGINT	BIGINT
BIGINT	문자열	DECFLOAT(34)
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)
decimal(w,x)	문자열	DECFLOAT(34)
REAL	REAL	REAL
REAL	SMALLINT, INTEGER, BIGINT 또는 DECIMAL	DOUBLE
REAL	문자열	DECFLOAT(34)
DOUBLE	SMALLINT, INTEGER, BIGINT, DECIMAL, REAL 또는 DOUBLE	DOUBLE
DOUBLE	문자열	DECFLOAT(34)
DECFLOAT(n)	SMALLINT, INTEGER, DECIMAL (<=16,s), REAL 또는 DOUBLE	DECFLOAT(n)
DECFLOAT(n)	BIGINT 또는 DECIMAL(>16,s)	DECFLOAT(34)

하나의 피연산자	다른 피연산자	결과 데이터 유형
DECFLOAT(n)	DECFLOAT(m)	DECFLOAT(MAX(n,m))
DECFLOAT(n)	문자열	DECFLOAT(34)

[1] 정밀도는 31을 초과할 수 없습니다.

날짜 시간

날짜 시간 데이터 유형은 동일 데이터 유형의 다른 피연산자 또는 동일 데이터 유형의 유효한 문자열 표현을 포함하는 CHAR 또는 VARCHAR 표현식과 호환 가능합니다. 또한 DATE는 TIMESTAMP와 호환 가능하며 TIMESTAMP의 다른 피연산자는 시간소인 또는 날짜의 문자열 표현일 수 있습니다. 유니코드 데이터베이스에서 문자열 및 그래픽 문자열은 다른 날짜 시간 피연산자와 호환 가능한 날짜 시간 값의 GRAPHIC 또는 VARGRAPHIC 문자열 표현과 호환 가능합니다.

표 16. 날짜 시간 피연산자가 있는 결과 데이터 유형

하나의 피연산자	다른 피연산자	결과 데이터 유형
DATE	DATE, CHAR(*y*) 또는 VARCHAR(*y*)	DATE
TIME	TIME, CHAR(*y*) 또는 VARCHAR(*y*)	TIME
TIMESTAMP(*x*)	TIMESTAMP(*y*)	TIMESTAMP(*max(x,y)*)
TIMESTAMP(*x*)	DATE, CHAR(*y*) 또는 VARCHAR(*y*)	TIMESTAMP(*x*)

XML

XML 값은 다른 XML 값과 호환 가능합니다. 결과의 데이터 유형은 XML입니다.

부울

부울 값은 다른 부울 값과 호환 가능합니다. 결과의 데이터 유형은 부울입니다.

구별 유형

사용자 정의 구별 유형 값은 동일한 사용자 정의 구별 유형의 다른 값과만 호환 가능합니다. 결과 데이터 유형은 사용자 정의 구별 유형입니다.

배열 데이터 유형

사용자 정의 배열 데이터 유형 값은 동일한 사용자 정의 배열 데이터 유형의 다른 값과만 호환 가능합니다. 결과 데이터 유형은 사용자 정의 배열 데이터 유형입니다.

커서 데이터 유형

CURSOR 값은 다른 CURSOR 값과 호환 가능합니다. 결과 데이터 유형은

CURSOR입니다. 사용자 정의 커서 데이터 유형 값은 동일한 사용자 정의 커서 데이터 유형의 다른 값과만 호환 가능합니다. 결과 데이터 유형은 사용자 정의 커서 데이터 유형입니다.

행 데이터 유형

사용자 정의 행 데이터 유형 값은 동일한 사용자 정의 행 데이터 유형의 다른 값과만 호환 가능합니다. 결과 데이터 유형은 사용자 정의 행 데이터 유형입니다.

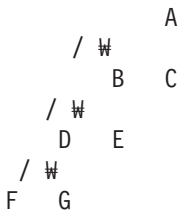
참조 유형

목표 유형이 공통 상위 유형을 갖는 경우 참조 유형 값은 동일한 참조 유형의 다른 값과 호환 가능합니다. 결과 데이터 유형은 목표 유형으로서 공통 슈퍼 유형을 가진 참조 유형입니다. 모든 피연산자에 동일한 범위 테이블이 있는 경우, 결과에는 해당 범위 테이블이 포함됩니다. 그렇지 않으면 결과의 범위가 지정되지 않습니다.

구조화된 유형

공통 상위 유형을 갖는 경우 구조화된 유형 값은 동일한 구조화된 유형의 다른 값과 호환 가능합니다. 결과로 생성되는 구조화된 유형 컬럼의 정적 데이터 유형은 어느 한 컬럼의 최소 공통 슈퍼 유형인 구조화된 유형입니다.

예를 들어, 다음과 같은 구조화된 유형 계층 구조를 살펴보십시오.



정적 유형 E 및 F의 구조화된 유형은 E 및 F의 최소 공통 슈퍼 유형인 B에 대해 결과로 생성되는 정적 유형과 호환됩니다.

결과의 널(NULL) 입력 가능 속성

INTERSECT와 EXCEPT를 제외하고, 두 피연산자가 널(NULL)을 허용하는 한 결과에 널(NULL)이 허용됩니다.

- INTERSECT의 경우, 피연산자 중 하나가 널(NULL)을 허용하지 않으면 결과에는 널(NULL)이 허용되지 않습니다. 교집합은 널(NULL)이 될 수 없습니다.
- EXCEPT의 경우, 첫 번째 피연산자가 널(NULL)을 허용하지 않으면 결과에도 널(NULL)이 허용되지 않습니다. 첫 번째 피연산자 값만 결과가 될 수 있습니다.

문자열 변환 규칙

연산을 수행하는 데 사용되는 코드 페이지는 해당 연산의 피연산자에 적용된 규칙에 따라 결정됩니다. 이 절에서는 다음 규칙에 대해 설명합니다.

이러한 규칙이 다음에 적용됩니다.

- 집합 연산(UNION, INTERSECT 및 EXCEPT)이 있는 fullselect의 해당 문자열 컬럼
- 병합의 피연산자
- 술어 피연산자(LIKE 제외)
- CASE 표현식 및 DECODE 스칼라 함수의 결과 표현식
- 스칼라 함수 COALESCE(또한 NVL 및 VALUE)의 인수
- 스칼라 함수 GREATEST, LEAST, MAX 및 MIN의 인수
- 스칼라 함수 OVERLAY(및 INSERT)의 *source-string* 및 *insert-string* 인수
- IN 술어 목록에 있는 표현식 값
- 복수 행 VALUES절의 해당 표현식

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

결과의 코드 페이지는 피연산자의 코드 페이지에서 결정됩니다. 처음 두 피연산자의 코드 페이지가 중간 결과 코드 페이지를 결정하고, 다음 피연산자의 코드 페이지와 이 코드 페이지가 새로운 중간 결과 코드 페이지를 결정합니다. 마지막 중간 결과 코드 페이지와 마지막 피연산자의 코드 페이지가 결과 문자열이나 컬럼의 코드 페이지를 결정합니다. 각 코드 페이지 쌍의 경우, 결과는 다음 규칙의 순차적 응용프로그램에서 결정됩니다.

- 코드 페이지가 같은 경우, 결과는 해당 코드 페이지입니다.
- 어느 한 코드 페이지가 BIT DATA(코드 페이지 0)일 경우, 결과 코드 페이지는 BIT DATA입니다.
- 유니코드 데이터베이스에서 하나의 코드 페이지가 다른 코드 페이지와 다른 코드화 체계의 데이터를 나타내는 경우, 결과는 UTF-8의 UCS-2입니다(즉, 문자 데이터 유형의 그래픽 데이터 유형). 비유니코드 데이터베이스에서는 다른 코드화 체계 간의 변환이 지원되지 않습니다.
- 호스트 변수인 피연산자의 경우(코드 페이지가 BIT DATA가 아님), 결과 코드 페이지는 데이터베이스 코드 페이지입니다. 이러한 호스트 변수에 있는 입력 데이터는 사용되기 전에 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환됩니다.

필요하면 다음에 대해 결과의 코드 페이지 변환이 수행됩니다.

- 병합 연산자의 피연산자
- COALESCE(또한 NVL 및 VALUE) 스칼라 함수의 선택된 인수
- 스칼라 함수 GREATEST, LEAST, MAX 및 MIN의 선택된 인수
- 스칼라 함수 OVERLAY(및 INSERT)의 *source-string* 및 *insert-string* 인수
- CASE 표현식 및 DECODE 스칼라 함수의 선택된 결과 표현식
- IN 술어 목록에 있는 표현식
- 복수 행 VALUES절의 해당 표현식
- 집합 연산에 포함된 해당 컬럼

문자 변환은 다음 사항이 해당되는 경우에만 필요합니다.

- 코드 페이지가 다릅니다.
- 문자열이 BIT DATA가 아닙니다.
- 문자열이 널(NULL)이 아니고 비어 있지도 않습니다.

예:

예 1: 코드 페이지 850으로 작성된 데이터베이스에 다음이 제공될 경우,

표현식	유형	코드 페이지
COL_1	컬럼	850
HV_2	호스트 변수	437

술어 평가는 다음과 같이 수행됩니다.

```
COL_1 CONCAT :HV_2
```

호스트 변수 데이터는 사용되기 전에 데이터베이스 코드 페이지로 변환되기 때문에 두 피연산자의 결과 코드 페이지는 850입니다.

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.

유니코드 데이터베이스에서 문자열 비교

패턴 일치는 기존 MBCS 데이터베이스 동작이 유니코드 데이터베이스 동작과 조금 다른 한 영역입니다.

Linux, UNIX 및 Windows용 DB2 데이터베이스의 MBCS 데이터베이스에 대해 현재 동작은 다음과 같습니다. match-expression에 MBCS 데이터가 포함된 경우 패턴은 SBCS 및 비SBCS 문자를 모두 포함할 수 있습니다. 패턴의 특수 문자는 다음과 같이 해석됩니다.

- SBCS 반자(halfwidth) 밑줄은 하나의 SBCS 문자를 지칭합니다.
- 비SBCS 전자(fullwidth) 밑줄은 하나의 비SBCS 문자를 지칭합니다.
- 퍼센트(SBCS 반자 또는 비SBCS 전자)는 0개 이상의 SBCS 또는 비SBCS 문자입니다.

유니코드 데이터베이스에서는 ″단일 바이트″와 ″비단일 바이트″ 문자 간에 차이가 없습니다. UTF-8 형식은 유니코드 문자의 ″혼합 바이트″ 인코딩이지만 UTF-8의 SBCS와 비SBCS 문자를 구별하지는 않습니다. UTF-8 형식의 바이트 수와 상관없이 모든 문자는 유니코드 문자입니다. 유니코드 그래픽 컬럼에서 반자(halfwidth) 밑줄(U+005F) 및 반자 퍼센트(U+0025)를 포함하여 모든 비보조(non-supplementary) 문자의 길이는 2바이트입니다. 유니코드 데이터베이스의 경우 패턴에 있는 특수 문자는 다음과 같이 해석됩니다.

- 문자열의 경우 반자 밑줄(X’5F’) 또는 전자 밑줄(X’EFBCBF’)은 한 개의 유니코드 문자입니다. 반자 퍼센트(X’25’) 또는 전자 퍼센트(X’EFBC85’)는 0개 이상의 유니코드 문자입니다.
- 그래픽 문자열의 경우 반자 밑줄(U+005F) 또는 전자 밑줄(U+FF3F)은 한 개의 유니코드 문자입니다. 반자 퍼센트(U+0025) 또는 전자 퍼센트(U+FF05)는 0개 이상의 유니코드 문자입니다.

주: 유니코드 보조 그래픽 문자와 일치시키기 위해 두 개의 밑줄이 필요하며 이는 문자가 GRAPHIC 컬럼에서 두 개의 UCS-2 문자로 표시되기 때문입니다. CHAR 컬럼에서 유니코드 보조 문자를 일치시키려면 한 개의 밑줄만 필요합니다.

밑줄 및 퍼센트 기호 문자의 특수한 의미를 수정하는 데 사용할 수 있는 문자를 지정한 선택적 ″이스케이프 표현식″의 경우 다음 중 하나로 표현식을 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 사항을 병합한 표현식

다음과 같은 제한사항이 있습니다.

- 표현식의 어떤 요소도 CLOB 또는 DBCLOB 유형이 될 수 없습니다. 또한 BLOB 파일 참조 변수가 될 수 없습니다.

- CHAR 컬럼의 경우, 표현식의 결과는 한 문자 또는 정확히 (1)바이트를 포함하는 FOR BIT DATA 문자열이어야 합니다(SQLSTATE 22019). GRAPHIC 컬럼의 경우 표현식 결과는 한 개의 문자여야 합니다(SQLSTATE 22019).

앵커된 유형에 대해 앵커 오브젝트 분석

ROW를 지정하지 않은 앵커된 유형의 앵커 오브젝트는 SQL 변수, SQL 매개변수, 전역 변수, 모듈 변수, 테이블 컬럼 또는 뷰의 컬럼을 나타낼 수 있는 이름으로 지정됩니다.

앵커 오브젝트 분석 방법은 ANCHOR절을 사용하는 명령문의 컨텍스트와 앵커 오브젝트 이름의 ID 수에 따라 다릅니다.

- 앵커 오브젝트 이름이 1개의 ID로 지정되면 이름은 SQL 변수, SQL 매개변수, 모듈 변수 또는 전역 변수로 나타낼 수 있습니다.
- 앵커 오브젝트 이름이 2개의 ID로 지정되면 이름은 레이블에서 규정된 SQL 변수, 루틴에서 규정된 SQL 매개변수, 스키마에서 규정된 전역 변수, 모듈 변수, 테이블 컬럼 또는 뷰의 컬럼을 나타낼 수 있습니다.
- 앵커 오브젝트 이름이 3개의 ID로 지정되면 이름은 스키마에서 규정된 테이블의 컬럼, 스키마에서 규정된 뷰의 컬럼, 현재 서버 이름과 스키마에서 규정된 전역 변수 또는 스키마에서 규정된 모듈 변수를 나타냅니다.

ANCHOR절이 복합 명령문 내의 SQL 변수 선언에서 사용되면 SQL 변수가 앵커 오브젝트 이름에 대해서만 후보가 됩니다. ANCHOR 절이 SQL 루틴 본문에서 사용된 복합 명령문 내에서 SQL 변수 선언으로 사용된 경우에만 SQL 매개변수가 앵커 오브젝트 이름에 대해서만 후보가 됩니다.

하나의 ID가 있는 앵커 오브젝트 이름 분석은 다음 단계를 사용하여 완료됩니다.

1. ANCHOR절은 복합 명령문의 SQL 변수 선언에 있다면 가장 내부에 중첩된 복합 명령문에서부터 시작하여 가장 외부의 복합 명령문까지 일치하는 SQL 변수 이름을 검색하십시오.
2. ANCHOR절이 루틴 본문 내의 복합 명령문의 SQL 변수 선언에 있다면 해당 루틴의 일치하는 SQL 매개변수 이름을 검색하십시오.
3. ANCHOR절이 모듈 오브젝트 정의에 사용된다면 모듈 내에서 일치되는 모듈 변수 이름을 검색하십시오.
4. 테이블이나 뷰 컬럼을 찾을 수 없는 경우, 첫 번째 ID를 스키마 이름으로 사용하고 두 번째 ID를 전역 변수 이름으로 사용하여 전역 변수를 검색하십시오.
5. 아직 찾지 못했다면 SQL 경로에서의 전역 변수 이름과 일치하는 전역 변수를 검색하십시오.

2개의 ID가 있는 다른 앵커 오브젝트 이름 분석은 다음 단계를 사용하여 완료됩니다.

1. ANCHOR절은 복합 명령문의 SQL 변수 선언에 있다면 가장 내부에 중첩된 복합 명령문에서 시작하여 가장 외부의 복합 명령문까지 일치하는 규정된 SQL 변수 이름을 검색하십시오.

2. ANCHOR절이 루틴 본문 내의 복합 명령문의 SQL 변수 선언에 있다면, 앵커 오브젝트 이름의 첫 번째 ID가 루틴 이름과 일치하는 경우 루틴과 일치하는 SQL 매개변수 이름을 검색하십시오.
3. ANCHOR절이 모듈 오브젝트 정의에 사용되고 첫 번째 ID가 해당 모듈의 모듈 이름과 일치한다면, 두 번째 ID와 일치하는 모듈 내에서의 모듈 변수 이름을 검색하십시오.
4. 모듈 변수 이름을 찾을 수 없는 경우, 첫 번째 ID를 테이블이나 뷰 이름으로 사용하고 두 번째 ID를 컬럼 이름으로 사용하여 현재 스키마에서 테이블이나 뷰 컬럼을 검색하십시오.
5. 모듈 변수 이름을 찾을 수 없는 경우, 첫 번째 ID를 스키마 이름으로 사용하고 두 번째 ID를 전역 변수 이름으로 사용하여 전역 변수를 검색하십시오.
6. 모듈 변수 이름을 찾지 못했고 3단계에서도 모듈이 검색되지 않는 경우, 첫 번째 ID와 일치하는 이름으로 SQL 경로에서 모듈을 검색하십시오. 찾은 경우 두 번째 ID를 사용하여 모듈에서 일치되는 발행된 모듈 변수 이름을 검색하십시오.
7. 6단계에서 SQL 경로를 사용하여 모듈을 찾을 수 없으면 첫 번째 ID의 이름과 일치하는 모듈 공용 별명을 점검하십시오. 찾은 경우 두 번째 ID를 사용하여 모듈 공용 별명에서 식별된 모듈에서 일치되는 발행된 모듈 변수 이름을 검색하십시오.

3개의 ID가 있는 앵커 오브젝트 이름 분석은 다음 단계를 사용하여 이루어집니다.

1. ANCHOR절이 모듈 오브젝트 정의에 사용되고 첫 번째 2개의 ID가 해당 모듈의 모듈 이름과 스키마 이름과 일치하면, 마지막 ID와 일치하는 이름의 모듈 변수를 검색하십시오.
2. 이전 단계에서 찾을 수 없거나 이 단계를 적용할 수 없는 경우, 첫 번째 ID를 스키마 이름으로 사용하고 두 번째 ID를 테이블이나 뷰 이름으로, 세 번째 ID를 컬럼 이름으로 각각 사용하여 테이블이나 뷰 컬럼을 검색하십시오.
3. 이전 단계에서 찾을 수 없거나 첫 번째 ID가 현재 서버 이름과 같은 경우, 스키마 이름으로 두 번째 ID를 사용하고 전역 변수 이름으로 세 번째 ID를 사용하여 전역 변수를 검색하십시오.
4. 아직 찾지 못했고 1단계에서 모듈이 검색되지 않는 경우 스키마 이름으로 첫 번째 ID를 사용하고 모듈 이름으로 두 번째 ID를 사용하여 발행된 모듈 변수를 검색하십시오. 이러한 모듈이 존재하는 경우 세 번째 ID를 사용하여 모듈에서 일치하는 발행된 모듈 변수 이름을 검색하십시오.

앵커된 행 유형에 대해 앵커 오브젝트 분석

ROW 키워드를 포함한 앵커된 유형의 앵커 오브젝트는 ANCHOR절의 컨텍스트와 이름의 ID 수와 컨텍스트에 따라 SQL 변수, SQL 매개변수, 전역 변수, 모듈 변수, 테이블 또는 뷰를 나타낼 수 있는 이름으로 지정됩니다.

앵커 오브젝트 분석 방법은 ANCHOR절을 사용하는 명령문의 컨텍스트와 앵커 오브젝트 이름의 ID 수에 따라 다릅니다.

- 앵커 오브젝트 이름이 한 개의 ID로 지정되면 이름은 SQL 변수, SQL 매개변수, 모듈 변수, 전역 변수, 테이블이나 뷰를 나타낼 수 있습니다.
- 앵커 오브젝트 이름이 두 개의 ID로 지정되면 이름은 레이블에서 규정된 SQL 변수, 루틴에서 규정된 SQL 매개변수, 스키마에서 규정된 전역 변수, 모듈 변수, 스키마에서 규정된 테이블이나 스키마에서 규정된 뷰를 나타낼 수 있습니다.
- 앵커 오브젝트 이름이 세 개의 ID로 지정되면 이름은 현재 서버 이름과 스키마에서 규정된 전역 변수, 현재 서버 이름과 스키마에서 규정된 테이블, 현재 서버 이름과 스키마에서 규정된 뷰 또는 스키마에서 규정된 모듈 변수를 나타냅니다.

ANCHOR절이 복합 명령문 내의 SQL 변수 선언에서 사용되면 SQL 변수가 앵커 오브젝트 이름에 대해서만 후보가 됩니다. ANCHOR절이 SQL 루틴 본문에서 사용된 복합 명령문 내에서 SQL 변수 선언으로 사용된 경우에만 SQL 매개변수가 앵커 오브젝트 이름에 대해서만 후보가 됩니다. 한 개의 ID가 있는 앵커 오브젝트 이름 분석은 다음 단계를 사용하여 완료됩니다.

1. ANCHOR절은 복합 명령문의 SQL 변수 선언에 있다면 가장 내부에 중첩된 복합 명령문에서부터 시작하여 가장 외부의 복합 명령문까지 일치하는 SQL 변수 이름을 검색하십시오.
2. ANCHOR절이 루틴 본문 내의 복합 명령문의 SQL 변수 선언에 있다면 해당 루틴의 이름과 일치하는 SQL 매개변수를 검색하십시오.
3. ANCHOR절이 모듈 오브젝트 정의에 사용된다면 모듈 내에서 일치되는 모듈 변수 이름을 검색하십시오.
4. 아직 찾지 못했다면 현재 스키마에서 일치되는 이름으로 테이블이나 뷰를 검색하십시오.
5. 아직 찾지 못했다면 SQL 경로에서 일치되는 전역 변수 이름으로 스키마 전역 변수를 검색하십시오.

두 개의 ID가 있는 다른 앵커 오브젝트 이름 분석은 다음 단계를 사용하여 완료됩니다.

1. ANCHOR절은 복합 명령문의 SQL 변수 선언에 있다면 가장 내부에 중첩된 복합 명령문에서 시작하여 가장 외부의 복합 명령문까지 일치하는 규정된 SQL 변수 이름을 검색하십시오.

2. ANCHOR절이 루틴 본문 내의 복합 명령문의 SQL 변수 선언에 있다면, 앵커 오브젝트 이름의 첫 번째 ID가 루틴 이름과 일치하는 경우 루틴과 일치하는 SQL 매개변수 이름을 검색하십시오.
3. ANCHOR절이 모듈 오브젝트 정의에 사용되고 첫 번째 ID가 해당 모듈의 모듈 이름과 일치한다면, 두 번째 ID와 일치하는 모듈 내에서의 모듈 변수 이름을 검색하십시오.
4. 아직 찾지 못했다면 스키마 이름으로 첫 번째 ID를 사용하고 테이블이나 뷰 이름으로 두 번째 ID를 사용하여 테이블이나 뷰를 검색하십시오.
5. 아직 찾지 못했다면 스키마 이름으로 첫 번째 ID를 사용하고 전역 변수 이름으로 두 번째 ID를 사용하여 전역 변수를 검색하십시오.
6. 아직 찾지 못했고 3단계에서 모듈이 검색되지 않았다면 첫 번째 ID와 일치하는 이름으로 SQL 경로에서 모듈을 검색하십시오. 찾은 경우 두 번째 ID를 사용하여 모듈에서 일치되는 발행된 모듈 변수 이름을 검색하십시오.
7. 6단계에서 SQL 경로를 사용하여 모듈을 찾을 수 없으면 첫 번째 ID의 이름과 일치하는 모듈 공용 별명을 점검하십시오. 찾은 경우 두 번째 ID를 사용하여 모듈 공용 별명에서 식별된 모듈에서 일치되는 발행된 모듈 변수 이름을 검색하십시오.

세 개의 ID가 있는 앵커 오브젝트 이름 분석은 다음 단계를 사용하여 완료됩니다.

1. ANCHOR절이 모듈 오브젝트 정의에 사용되고 첫 번째 두 개의 ID가 해당 모듈의 모듈 이름과 스키마 이름과 일치하면, 마지막 ID와 일치하는 이름의 모듈 변수를 검색하십시오.
2. 아직 찾지 못했고 첫 번째 ID가 현재 서버 이름과 같다면 스키마 이름으로 두 번째 ID를 사용하고 테이블이나 뷰 이름으로 세 번째 ID를 사용하여 테이블이나 뷰를 검색하십시오.
3. 아직 찾지 못했고 첫 번째 ID가 현재 서버 이름과 같다면, 스키마 이름으로 두 번째 ID를 사용하고 전역 변수 이름으로 세 번째 ID를 사용하여 전역 변수를 검색하십시오.
4. 아직 찾지 못했고 1단계에서 모듈을 검색하지 못했다면 스키마 이름으로 첫 번째 ID를 사용하고 모듈 이름으로 두 번째 ID를 사용하여 발행된 모듈 변수를 검색하며, 이러한 모듈이 존재하는 경우 세 번째 ID를 사용하여 모듈에서 일치되는 발행된 변수 이름을 검색하십시오.

데이터베이스 파티션 호환 가능 데이터 유형

*데이터베이스 파티션 호환성*은 파티션 키의 해당 컬럼에 대한 기본 데이터 유형 사이에서 정의됩니다. 데이터베이스 파티션 호환 데이터 유형은 한 유형에서 하나씩 동일한 값의 두 변수가 동일한 파티션 함수에 의해 동일한 분배 맵 인덱스에 맵핑되는 등록 정보를 갖습니다.

161 페이지의 표 17에서는 데이터베이스 파티션에서 데이터 유형의 호환성을 나타냅니다.

데이터베이스 파티션 호환성은 다음과 같은 등록 정보를 갖습니다.

- 내부 형식은 DATE, TIME 및 TIMESTAMP에 사용됩니다. 이들은 서로 호환되지 않으며, 이들 중 어느 것도 문자 또는 그래픽 데이터 유형과 호환되지 않습니다.
- 널(null) 값 허용은 파티션 호환성에 영향을 주지 않습니다.
- 조합은 파티션 호환성에 영향을 줍니다. 로케일 구분 UCA 기반 조합은 조합의 강도(S) 속성이 무시되는 경우를 제외하고 조합에서 완전 일치를 필요로 합니다. 기타 모든 조합은 파티션 호환성 판별 목적으로 같다고 간주됩니다.
- FOR BIT DATA를 사용하여 정의된 문자 컬럼은 로케일 구분 UCA 기반 조합 이외의 조합이 사용되는 경우 FOR BIT DATA가 없는 문자 컬럼과만 호환 가능합니다.
- 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 유형을 사용하여 데이터베이스 파티션 호환성을 분석합니다.
- 분배 키에서 동일한 값의 시간소인은 시간소인 정밀도가 다르더라도 동일하게 취급됩니다.
- 분배 키에서 동일한 값의 10진수는 스케일과 정밀도가 다르더라도 동일하게 취급됩니다.
- 문자열(CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC)에 있는 후행 공백은 시스템이 제공하는 해시 기능에 의해 무시됩니다.
- 로케일 구분 UCA 기반 조합이 사용되는 경우, CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC는 호환 가능한 데이터 유형입니다. 기타 조합이 사용되는 경우, CHAR 및 VARCHAR은 호환 가능한 유형이고 GRAPHIC 및 VARGRAPHIC는 호환 가능한 유형이지만 CHAR 및 VARCHAR은 GRAPHIC 및 VARGRAPHIC와 호환 가능한 유형이 아닙니다. 길이가 서로 다른 CHAR 또는 VARCHAR는 호환되는 데이터 유형입니다.
- 같은 DECFLOAT 값은 정밀도가 다르더라도 동일하게 취급됩니다. 숫자적으로 같은 DECFLOAT 값은 유효 숫자 수가 다르더라도 동일하게 취급됩니다.

- 분산 키의 일부로 지원되지 않는 데이터 유형은 데이터베이스 파티션 호환성에 적용할 수 없습니다. 여기에는 데이터 유형이 BLOB, CLOB, DBCLOB, XML, 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형인 컬럼이 포함됩니다.

표 17. 데이터베이스 파티션 호환성

피연산자	2진 정수	10진수	부동 소수점	10진 부동 소수점	문자열	그래픽 문자열	날짜	시간	시간소인	구별 유형
2진 정수	예	없음	없음	없음	없음	없음	없음	없음	없음	[1]
10진수	없음	예	없음	없음	없음	없음	없음	없음	없음	[1]
부동 소수점	없음	없음	예	없음	없음	없음	없음	없음	없음	[1]
10진 부동 소수점	없음	없음	없음	예	없음	없음	없음	없음	없음	[1]
문자열	없음	없음	없음	없음	예[2]	2, 3	없음	없음	없음	[1]
그래픽 문자열	없음	없음	없음	없음	2, 3	예[2]	없음	없음	없음	[1]
날짜	없음	없음	없음	없음	없음	없음	예	없음	없음	[1]
시간	없음	없음	없음	없음	없음	없음	없음	예	없음	[1]
시간소인	없음	없음	없음	없음	없음	없음	없음	없음	예	[1]
구별 유형	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]

주:

[1] 구별 유형 값은 구별 유형의 소스 데이터 유형 또는 소스 데이터 유형이 같은 다른 구별 유형과 호환할 수 있는 데이터베이스 파티션입니다. 구별 유형의 소스 데이터 유형은 분산 키의 일부로 지원되는 데이터 유형이어야 합니다. 사용자 정의 구별 유형(UDT) 값은 데이터베이스 파티션 호환 소스 유형의 다른 UDT 또는 UDT의 소스 유형과 호환되는 파티션입니다. 구별 유형은 BLOB, CLOB, DBCLOB 또는 XML을 기반으로 할 수 없습니다.

[2] 문자 및 그래픽 문자열 유형은 호환 가능한 조합을 가진 경우 호환 가능합니다.

[3] 문자 및 그래픽 문자열 유형은 로케일 구분 UCA 기반 조합이 적용되는 경우 호환 가능합니다. 그렇지 않으면, 호환 가능한 유형이 아닙니다.

상수

상수(*리터럴*이라고도 함)는 값을 지정합니다. 상수는 문자열이나 숫자로 구분됩니다. 숫자 상수는 정수, 부동 소수점 숫자 또는 소수로 다시 분류됩니다.

모든 상수에는 NOT NULL 속성이 있습니다.

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

사용자 정의 유형에는 명백한 유형 지정이 있습니다. 이것은 사용자 정의 유형이 자신의 유형과만 호환됨을 의미합니다. 그러나 상수에는 내장 유형이 있습니다. 따라서 사용자 정의 유형과 상수를 포함하는 연산은 사용자 정의 유형이 상수의 내장 유형으로 캐스트되었거나 상수가 사용자 정의 유형으로 캐스트된 경우에만 가능합니다. 예를 들어, 144 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
```

정수 상수

*정수 상수*는 최대 자릿수가 19개(소수점은 포함되지 않음)인 부호가 있거나 부호가 없는 숫자로서 정수를 지정합니다. 정수 상수의 데이터 유형 값이 정수 범위 내에 있으면 이는 정수입니다. 정수 상수의 데이터 유형 값이 정수 범위 밖에 있으나 대정수 범위 내에 있으면 이는 대정수입니다. 대정수 값의 범위를 벗어나 정의된 상수는 10진 상수로 간주됩니다.

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

예:

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

구문 도표에서 '정수'라는 용어는 부호를 포함해서는 안 되는 큰 정수 상수에 사용됩니다.

부동 소수점 상수

*부동 소수점 상수*는 부동 소수점 숫자를 E로 분리된 두 개의 숫자로 지정합니다. 첫 번째 숫자에는 부호와 소수점이 포함되고 두 번째 숫자에는 부호는 포함되나 소수점은 포함되지 않습니다. 부동 소수점 상수의 데이터 유형은 배정밀도입니다. 상수 값은 두 번째 숫자에 의해 지정된 10의 승수와 첫 번째 숫자로 산출한 것으로서, 부동 소수점 숫자 범위 내에 있어야 합니다. 상수의 바이트 수는 30을 초과해서는 안 됩니다.

예:

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

소수 상수

*소수 상수*는 부호가 있거나 부호가 없는 숫자로서 최대 31개의 숫자로 구성되며, 소수점이 들어 있거나 2진 정수 범위 내에 있지 않습니다. 소수 범위 내에 있어야 합니다. 정밀도는 숫자의 총 수(앞에 오는 0과 뒤에 오는 0 포함)이며, 스케일은 소수점 오른쪽에 있는 숫자 수(뒤에 오는 0 포함)입니다.

예:

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

10진수 부동 소수점 상수

10진수 부동 소수점 특정 값을 제외한 10진수 부동 소수점 상수는 없으며, DECFLOAT(34)로 해석됩니다.

이들 특정 값은 INFINITY, NAN 및 SNAN입니다. INFINITY는 숫자의 *크기*가 무한대로 큰 무한대를 나타냅니다. INFINITY는 선택적 기호가 앞에 나타날 수 있습니다. INFINITY 대신 INF를 지정할 수 있습니다. NAN은 NaN(Not a Number)을 나타내며 때로는 quiet NaN이라고도 합니다. 경고나 예외를 발생시키지 않는 정의되지 않은 결과를 나타내는 값입니다. SNAN은 sNaN(signaling NaN)을 나타냅니다. 숫자 연산으로 정의되는 연산에서 사용되는 경우 경고나 예외를 발생시키는 정의되지 않은 결과를 나타내는 값입니다. NAN 및 SNAN 모두 앞에 선택적 기호가 표시되지만 산술 연산의 경우 이 기호는 중요하지 않습니다. 예를 들어 INSERT의 VALUES 목록 또는 술어에서 비교된 상수로서, SNAN은 경고나 예외를 발생시키지 않고 숫자 이외의 연산에서 사용될 수 있습니다.

```
SNAN   -INFINITY
```

특수 값(INFINITY, NAN, 및 SNA) 중 하나가 이름으로 해석 가능한 컨텍스트에서 사용되는 경우 값을 10진수의 부동 소수점으로 명시적으로 캐스트하십시오. 예를 들어,

```
CAST ('SNAN' AS DECFLOAT)
```

특정 값이 아닌 값 모두는 위에서 지정한 규칙에 따라 정수, 부동 소수점 또는 10진수 상수로 해석됩니다. 숫자 10진수 부동 소수점 값을 얻으려면, 문자열 상수와 함께 DECFLOAT 캐스트 기능을 사용합니다. 부동 소수점이 정확하지 않으며 결과 10진수 부동 소수점 값은 인수를 구성하는 10진수 숫자 문자와 다를 수 있으므로, 부동 소수점 상수를 DECFLOAT 함수의 인수로 사용하는 것이 좋습니다. 대신에 문자 상수를 DECFLOAT 함수에 대한 인수로 사용합니다.

예를 들어 DECFLOAT('6.0221415E23', 34)는 10진수 부동 소수점 값 6.0221415E+23을 리턴하지만 DECFLOAT(6.0221415E23, 34)는 10진수 부동 소수점 값 6.0221415000000003E+23을 리턴합니다.

문자열 상수

*문자열 상수*는 가변 길이 문자열을 지정합니다. 문자열 상수의 세 가지 양식은 다음과 같습니다.

- 아포스트로피(')인 문자열 분리문자로 시작되고 종료되는 문자 시퀀스. 문자열 분리문자 사이의 바이트 수는 32 672보다 클 수 없습니다. 두 개의 연속적인 문자열 분리문자는 문자열 내에 있는 하나의 문자열 분리문자를 나타내는 데 사용됩니다. 문자열 내에 포함되지 않는 두 개의 연속 문자열 분리문자는 비어 있는 문자열을 나타냅니다.
- X 다음에 문자열 분리문자로 시작되고 종료되는 문자의 시퀀스가 옵니다. 이 양식의 문자열 상수를 *16진수 상수*라고도 합니다. 문자열 분리문자 사이에 있는 문자는 짝수의 16진수여야 합니다. 문자열 분리문자 사이의 공백은 무시됩니다. 16진수의 숫자는 32 672를 초과해서는 안됩니다. 16진수는 숫자이거나 A - F(대문자 또는 소문자) 사이의 문자입니다. 16진수 표기법의 규칙에서, 16진수 숫자의 각 쌍은 문자를 나타냅니다. 이 양식의 문자열 상수를 사용하여 키보드 표시가 없는 문자를 지정할 수 있습니다.
- U& 다음에 문자열 분리문자로 시작되고 종료되는 문자의 시퀀스, 그리고 선택적으로 UESCAPE절이 옵니다. 이 양식의 문자열 상수를 *유니코드 문자열 상수*라고도 합니다. 문자열 분리문자 사이의 바이트 수는 32 672보다 클 수 없습니다. 유니코드 문자열 상수는 명령문 컴파일 시 UTF-8에서 섹션 코드 페이지로 변환됩니다. 두 개의 연속적인 문자열 분리문자는 문자열 내에 있는 하나의 문자열 분리문자를 나타내는 데 사용됩니다. 두 개의 연속 유니코드 Escape 문자는 문자열 내에서 하나의 유니코드 Escape 문자를 나타내는 데 사용되지만, 이들 문자는 문자 상수의 길이를 계산할 때 하나의 문자로 계산됩니다. 문자열 내에 포함되지 않는 두 개의 연속 문자열 분리문자는 비어 있는 문자열을 나타냅니다. UTF-8의 문자 범위는 1 - 4바이트이므로, 유니코드 문자열 상수의 최대 길이는 실제로 32 672 문자 미만으로 나타납니다.

  문자는 타이포그래프 문자(*그리프*) 또는 유니코드 코드 포인트로 표시될 수 있습니다. 유니코드 문자의 코드 포인트 범위는 X'000000' - X'10FFFF'입니다. 코드 포

인트로 유니코드 문자를 표현하려면, 유니코드 Escape 문자, 4개의 16진수 숫자 또는 유니코드 Escape 문자, 플러스 부호(+)와 6개의 16진수 숫자를 사용합니다. 디폴트 유니코드 Escape 문자는 역방향 사선(₩)이지만 UESCAPE절에서는 다른 문자가 지정될 수 있습니다. UESCAPE절은 UESCAPE 키워드, 문자열 분리문자 사이의 단일 문자로 지정됩니다. 유니코드 Escape 문자는 플러스 부호(+), 큰따옴표 기호(“), 작은 따옴표 기호(’), 공백 또는 0 - 9나 A - F의 문자(대문자나 소문자)일 수 없습니다(SQLSTATE 42604). 라틴어 대문자 A를 유니코드 코드 포인트로 지정할 수 있는 두 가지 방법 예는 ₩0041과 ₩+000041입니다.

상수 값은 이것이 데이터베이스로 제한될 때 항상 데이터베이스 코드 페이지로 변환됩니다. 이것은 데이터베이스 코드 페이지에 있는 것으로 간주됩니다. 따라서 FOR BIT DATA 컬럼과 상수를 조인하여 그 결과가 FOR BIT DATA인 표현식에 사용되는 경우, 상수 값은 사용시 데이터베이스 코드 페이지 표현에서 변환되지 않습니다.

예를 들면, 다음과 같습니다.

```
'12/14/1985'    '32'    'DON''T CHANGE'    ''
X'FFFF'    X'46 72 61 6E 6B'
U&'₩0141ód₩017A is a city in Poland'   U&'c:₩₩temp'
U&'@+01D11E' UESCAPE '@'
```

예제의 두 번째 행에서의 가장 오른쪽의 문자열은 ASCII 문자열 ’Frank’의 VARCHAR 패턴을 나타냅니다. 마지막 행은 다음과 같습니다. ’Ł ód□는 폴란드의 도시’, ’c:₩temp’ 및 음악 기호 G 음자리표를 나타내는 단일 문자입니다.

그래픽 문자열 상수

*그래픽 문자열 상수*는 1바이트 어포스트로피(’)로 시작하고 끝나며 1바이트 G 또는 N이 앞에 오는 일련의 2바이트 문자로 구성되는 가변 길이 그래픽 문자열을 지정합니다. 어포스트로피 사이의 문자는 짝수 바이트를 표시해야 하며, 그래픽 문자열의 길이는 16 336바이트를 초과해서는 안됩니다.

예:

```
G'2바이트 문자열'
N'2바이트 문자열'
```

분리문자로 간주되는 MBCS 문자의 일부로 어포스트로피를 표시해서는 안됩니다.

유니코드 데이터베이스는 가변 길이 그래픽 문자열을 지정하는 16진 그래픽 문자열 상수도 지원합니다. 16진 그래픽 문자열 상수의 형식은 다음과 같습니다. 어포스트로피로 시작하고 끝나는 일련의 문자가 뒤에 오는 GX입니다. 어포스트로피 사이에 있는 문자는 네 자리 16진수의 짝수 배수여야 합니다. 16진수 자릿수는 16 336을 초과할 수 없으며 초과하면 오류가 리턴됩니다(SQLSTATE 54002). 16진 그래픽 문자열 상수가 부

적절하게 구성되면 오류가 리턴됩니다(SQLSTATE 42606). 각 네 자리 숫자 그룹은 단일 그래픽 문자를 나타냅니다. 유니코드 데이터베이스의 경우는 단일 UCS-2 그래픽 문자가 됩니다.

예:

```
GX'FFFF'
```

유니코드 데이터베이스에서 비트 패턴 '1111111111111111'을 나타냅니다.

```
GX'005200690063006B'
```

유니코드 데이터베이스에서 ASCII 문자열 'Rick'의 VARGRAPHIC 패턴을 나타냅니다.

날짜 및 시간 상수

*날짜 및 시간 상수*는 날짜, 시간 또는 시간소인을 지정합니다.

일반적으로 문자열 상수는 지정 및 비교에서 상수 날짜 시간 값을 나타내는 데 사용됩니다. 그러나 연관된 데이터 유형 이름은 문자열 상수의 특정 양식을 표시하는 데 사용되어 문자열 상수 대신 날짜 및 시간 상수로서 상수를 특별히 표시할 수 있습니다. 세 가지 날짜 및 시간 상수에 대한 형식은 다음과 같습니다.

DATE '*yyyy-mm-dd***'**
값의 데이터 유형은 DATE입니다.

TIME '*hh:mm:ss***'**
또는
TIME '*hh:mm***'**
값의 데이터 유형은 TIME입니다.

TIMESTAMP '*yyyy-mm-dd hh:mm:ss.nnnnnnnnnnnn***'**
또는
TIMESTAMP '*yyyy-mm-dd-hh.mm.ss.nnnnnnnnnnnn***'**
여기서 소수 초의 숫자 자릿수는 0 - 12이며 소수 초가 없는 경우 마침표는 생략될 수 있습니다. 값의 데이터 유형은 TIMESTAMP(*p*)이며, 여기서 *p*는 소수 초의 숫자 자릿수입니다.

선행 영(0)은 문자열 상수 분할 영역의 시간소인의 월, 일, 시 부분에서 생략될 수 있으며, 이는 각 날짜 및 시간 상수에서 적용 가능합니다. 선행 영(0) 문자는 TIME 또는 TIMESTMAP 상수의 분 및 초 요소에 포함되어야 합니다. 뒤 공백은 포함 및 무시될 수 있습니다.

UCS-2 그래픽 문자열 상수

유니코드 데이터베이스는 가변 길이 UCS-2 그래픽 문자열 상수를 지정하는 16진 UCS-2 그래픽 문자열을 지원합니다. 16진 UCS-2 그래픽 문자열 상수의 형식은 다음과 같습니다. 어포스트로피로 시작하고 끝나는 일련의 문자가 뒤에 오는 UX입니다. 어포스트로피 사이에 있는 문자는 네 자리 16진수의 짝수 배수여야 합니다. 16진수 자릿수는 16 336을 초과할 수 없으며 초과하면 오류가 리턴됩니다(SQLSTATE 54002). 16진 UCS-2 그래픽 문자열 상수가 부적절하게 구성되면 오류가 리턴됩니다(SQLSTATE 42606). 각 네 자리 숫자 그룹은 단일 UCS-2 그래픽 문자를 나타냅니다.

예:

```
UX'0042006F006200620079'
```

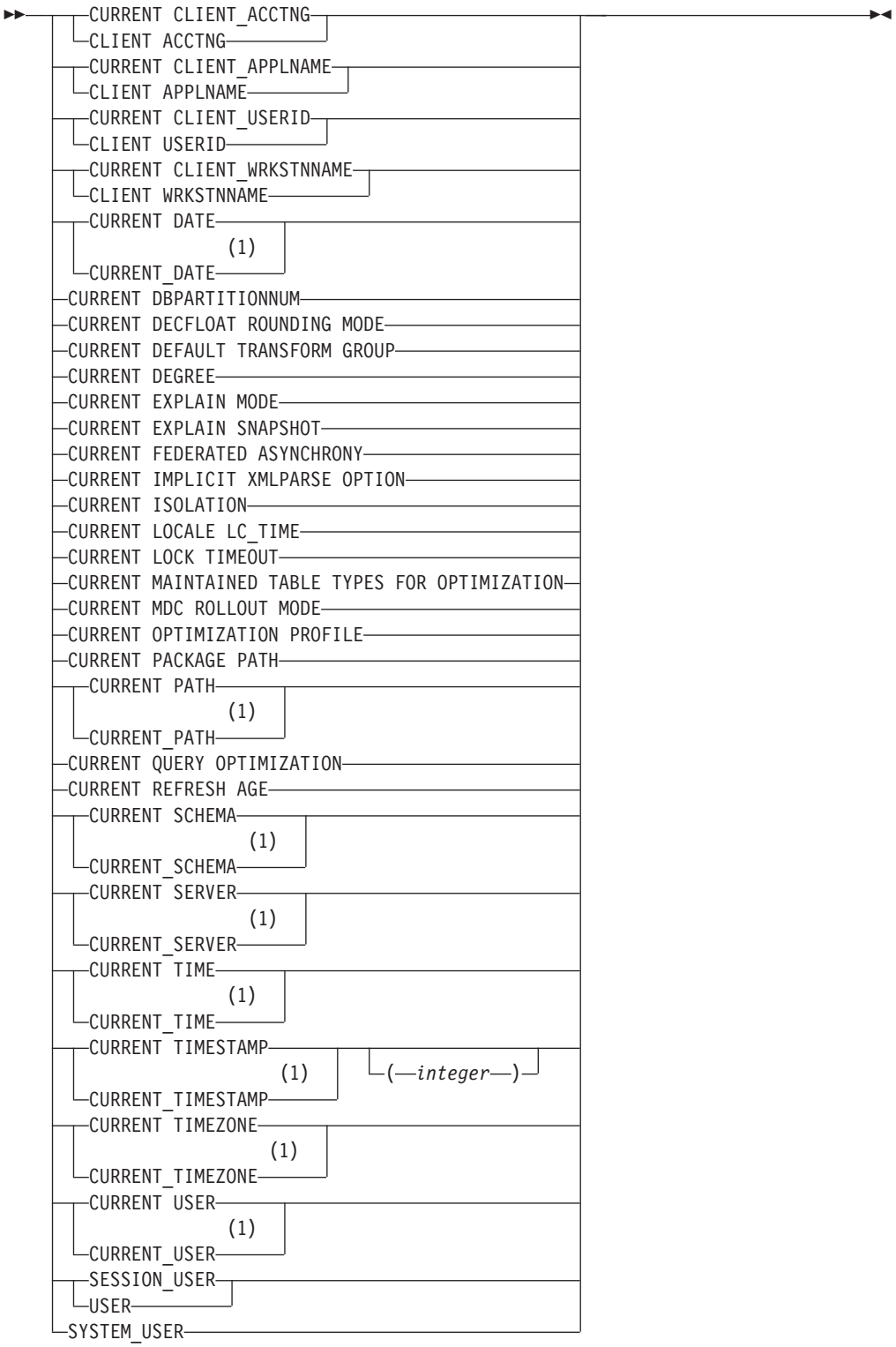
ASCII 문자열 'Bobby'의 VARGRAPHIC 패턴을 나타냅니다.

부울 상수

부울 상수는 키워드를 참 또는 거짓으로 지정하여 진리값을 각각 참 및 거짓으로 나타냅니다. 알 수 없는 진리값은 CAST(NULL AS BOOLEAN)를 사용하여 지정할 수 있습니다.

특수 레지스터

*특수 레지스터*는 데이터베이스 관리 프로그램이 응용프로그램 프로세스에 정의한 스토리지 영역으로, SQL문에서 참조할 수 있는 정보를 저장하는 데 사용됩니다. 특수 레지스터에 대한 참조는 현재 서버가 제공한 값에 대한 참조입니다. 값이 문자열인 경우, 해당 CCSID는 현재 서버의 디폴트 CCSID입니다. 특수 레지스터는 다음과 같이 참조될 수 있습니다.



주:

1 SQL2003 코어 표준은 밑줄이 있는 양식을 사용합니다.

일부 특수 레지스터는 SET문을 사용하여 갱신할 수 있습니다. 다음 테이블은 널(NULL) 값일 수 있는 특수 레지스터는 물론 갱신할 수 있는 특수 레지스터를 표시합니다.

표 18. 갱신 가능한 널(NULL) 입력 가능 특수 레지스터

특수 레지스터	갱신 가능 여부	널(NULL) 입력 가능
CURRENT CLIENT_ACCTNG	없음	없음
CURRENT CLIENT_APPLNAME	없음	없음
CURRENT CLIENT_USERID	없음	없음
CURRENT CLIENT_WRKSTNNAME	없음	없음
CURRENT DATE	없음	없음
CURRENT DBPARTITIONNUM	없음	없음
CURRENT DECFLOAT ROUNDING MODE	없음	없음
CURRENT DEFAULT TRANSFORM GROUP	예	없음
CURRENT DEGREE	예	없음
CURRENT EXPLAIN MODE	예	없음
CURRENT EXPLAIN SNAPSHOT	예	없음
CURRENT FEDERATED ASYNCHRONY	예	없음
CURRENT IMPLICIT XMLPARSE OPTION	예	없음
CURRENT ISOLATION	예	없음
CURRENT LOCALE LC_TIME	예	없음
CURRENT LOCK TIMEOUT	예	예
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	예	없음
CURRENT MDC ROLLOUT MODE	예	없음
CURRENT OPTIMIZATION PROFILE	예	예
CURRENT PACKAGE PATH	예	없음
CURRENT PATH	예	없음
CURRENT QUERY OPTIMIZATION	예	없음
CURRENT REFRESH AGE	예	없음
CURRENT SCHEMA	예	없음
CURRENT SERVER	없음	없음
CURRENT TIME	없음	없음
CURRENT TIMESTAMP	없음	없음
CURRENT TIMEZONE	없음	없음
CURRENT USER	없음	없음
SESSION_USER	예	없음
SYSTEM_USER	없음	없음
USER	예	없음

특수 레지스터가 루틴에서 참조될 경우 루틴의 특수 레지스터값은 특수 레지스터가 갱신 가능한지 여부에 따라 달라집니다. 갱신 불가능한 특수 레지스터의 경우, 값은 특수 레지스터의 디폴트값으로 설정됩니다. 갱신 가능한 특수 레지스터의 경우, 초기 값은 루틴의 호출자로부터 상속받으며 루틴 내에서 후속 SET문을 사용하여 변경할 수 있습니다.

CURRENT CLIENT_ACCTNG

CURRENT CLIENT_ACCTNG(또는 CLIENT ACCTNG) 특수 레지스터에는 이 연결에 대해 지정된 클라이언트 정보의 어카운팅 문자열 값이 들어있습니다. 레지스터의 데이터 유형은 VARCHAR(255)입니다. 이 레지스터의 디폴트값은 비어 있는 문자열입니다.

어카운팅 문자열의 값은 클라이언트 정보 설정(sqleseti) API를 사용하여 변경할 수 있습니다.

sqleseti API를 통해 제공된 값은 응용프로그램 코드 페이지에 있으며 특수 레지스터 값은 데이터베이스 코드 페이지에 저장된다는 점에 유의하십시오. 클라이언트 정보를 설정할 때 사용되는 데이터 값에 따라서 특수 레지스터에 저장되는 데이터 값의 절단이 코드 페이지 변환 중에 발생할 수 있습니다.

예: 이 연결에 대한 어카운팅 문자열의 현재 값을 가져오십시오.

```
VALUES (CURRENT CLIENT_ACCTNG)
  INTO :ACCT_STRING
```

CURRENT CLIENT_APPLNAME

CURRENT CLIENT_APPLNAME(또는 CLIENT APPLNAME) 특수 레지스터에는 이 연결에 대한 클라이언트 정보의 응용프로그램 이름 값이 들어있습니다. 레지스터의 데이터 유형은 VARCHAR(255)입니다. 이 레지스터의 디폴트값은 비어 있는 문자열입니다.

응용프로그램 이름의 값은 클라이언트 정보 설정(sqleseti) API를 사용하여 변경할 수 있습니다.

sqleseti API를 통해 제공된 값은 응용프로그램 코드 페이지에 있으며 특수 레지스터 값은 데이터베이스 코드 페이지에 저장된다는 점에 유의하십시오. 클라이언트 정보를 설정할 때 사용되는 데이터 값에 따라서 특수 레지스터에 저장되는 데이터 값의 절단이 코드 페이지 변환 중에 발생할 수 있습니다.

예: 이 연결에서 사용 중인 응용프로그램을 사용하도록 허용되는 부서를 선택하십시오.

```
SELECT DEPT
  FROM DEPT_APPL_MAP
  WHERE APPL_NAME = CURRENT CLIENT_APPLNAME
```

CURRENT CLIENT_USERID

CURRENT CLIENT_USERID(또는 CLIENT USERID) 특수 레지스터에는 이 연결에 대해 지정된 클라이언트 정보의 클라이언트 사용자 ID 값이 들어있습니다. 레지스터의 데이터 유형은 VARCHAR(255)입니다. 이 레지스터의 디폴트값은 비어 있는 문자열입니다.

클라이언트 사용자 ID 값은 클라이언트 정보 설정(sqleseti) API를 사용하여 변경할 수 있습니다.

sqleseti API를 통해 제공된 값은 응용프로그램 코드 페이지에 있으며 특수 레지스터 값은 데이터베이스 코드 페이지에 저장된다는 점에 유의하십시오. 클라이언트 정보를 설정할 때 사용되는 데이터 값에 따라서 특수 레지스터에 저장되는 데이터 값의 절단이 코드 페이지 변환 중에 발생할 수 있습니다.

예: 현재 클라이언트 사용자 ID가 작업하는 부서를 찾으십시오.

```
SELECT DEPT
  FROM DEPT_USERID_MAP
  WHERE USER_ID = CURRENT CLIENT_USERID
```

CURRENT CLIENT_WRKSTNNAME

CURRENT CLIENT_WRKSTNNAME(또는 CLIENT WRKSTNNAME) 특수 레지스터에는 이 연결에 대해 지정된 클라이언트 정보의 워크스테이션 이름 값이 들어있습니다. 레지스터의 데이터 유형은 VARCHAR(255)입니다. 이 레지스터의 디폴트값은 비어 있는 문자열입니다.

워크스테이션 이름의 값은 클라이언트 정보 설정(sqleseti) API를 사용하여 변경할 수 있습니다.

sqleseti API를 통해 제공된 값은 응용프로그램 코드 페이지에 있으며 특수 레지스터 값은 데이터베이스 코드 페이지에 저장된다는 점에 유의하십시오. 클라이언트 정보를 설정할 때 사용되는 데이터 값에 따라서 특수 레지스터에 저장되는 데이터 값의 절단이 코드 페이지 변환 중에 발생할 수 있습니다.

예: 이 연결에 사용되고 있는 워크스테이션 이름을 가져오십시오.

```
VALUES (CURRENT CLIENT_WRKSTNNAME)
  INTO :WS_NAME
```

CURRENT DATE

CURRENT DATE(또는 CURRENT_DATE) 특수 레지스터는 SQL문이 응용프로그램 서버에서 실행될 때의 시간 읽기를 기준으로 날짜를 지정합니다. 이 특수 레지스터가 단일 SQL문에서 두 번 이상 사용되거나 단일 명령문에서 CURRENT TIME 또는 CURRENT TIMESTAMP와 함께 사용된 경우, 모든 값은 단일 시계 읽기를 기준으로 합니다.

루틴 내의 SQL문에서 사용될 때 CURRENT DATE은 호출 명령문에서 상속받지 않습니다.

페더레이티드 시스템에서는 데이터 소스용 쿼리에 CURRENT DATE를 사용할 수 있습니다. 쿼리가 처리될 때 리턴되는 날짜는 데이터 소스가 아닌 페더레이티드 서버의 CURRENT DATE 레지스터로부터 확보됩니다.

예: DB2 CLP에서 다음 명령을 실행하여 현재 날짜를 알 수 있습니다.

```
db2 values CURRENT DATE
```

예: PROJECT 테이블을 사용하여 MA2111 프로젝트(PROJNO)의 프로젝트 종료 날짜를 현자 날짜로 설정하십시오.

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

CURRENT DBPARTITIONNUM

CURRENT DBPARTITIONNUM 특수 레지스터는 명령문에 대한 코디네이터 노드 번호를 식별하는 INTEGER 값을 지정합니다. 응용프로그램에서 발행되는 명령문의 경우 코디네이터는 응용프로그램이 연결하는 데이터베이스 파티션입니다. 루틴에서 발행되는 명령문의 경우 코디네이터는 루틴이 호출되는 데이터베이스 파티션입니다.

루틴 내에서 SQL문에서 사용될 때 CURRENT DBPARTITIONNUM은 호출 명령문에서 상속되지 않습니다.

CURRENT DBPARTITIONNUM은 데이터베이스 인스턴스가 데이터베이스 파티셔닝을 지원하기 위해 정의되지 않은 경우 0을 리턴합니다. (즉, db2nodes.cfg 파일이 없는 경우입니다. 파티션된 데이터베이스의 경우 db2nodes.cfg 파일이 존재하며 데이터베이스 파티션 정의가 들어있습니다.)

CURRENT DBPARTITIONNUM은 CONNECT문을 통해 변경할 수 있지만 특정 조건에서만 가능합니다.

버전 8 이전의 버전과의 호환성을 위해 DBPARTITIONNUM 대신 NODE 키워드를 사용할 수 있습니다.

예: 호스트 변수 APPL_NODE(정수)를 응용프로그램이 연결되는 데이터베이스 파티션의 번호로 설정하십시오.

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

CURRENT DECFLOAT ROUNDING MODE

CURRENT DECFLOAT ROUNDING MODE 특수 레지스터는 DECFLOAT 값에 사용되는 근사값 모드를 지정합니다.

데이터 유형은 VARCHAR(128)입니다. 다음 근사값 모드가 지원됩니다.

- ROUND_CEILING은 값을 양의 무한대 방향으로 라운드합니다. 버려지는 모든 숫자가 영(0)이거나 부호가 음수이면 결과는 변경되지 않습니다(버려지는 숫자 제거를 제외하고). 그렇지 않은 경우, 결과 계수는 1씩 증가합니다.
- ROUND_DOWN은 값을 0 방향으로 라운드합니다(절단). 버려지는 숫자는 무시됩니다.
- ROUND_FLOOR는 값을 음의 무한대 방향으로 라운드합니다. 버려지는 모든 숫자가 영(0)이거나 부호가 양수이면 결과는 변경되지 않습니다(버려지는 숫자 제거를 제외하고). 그렇지 않은 경우, 부호는 음수이고 결과 계수는 1씩 증가합니다.
- ROUND_HALF_EVEN은 값을 가장 가까운 값으로 라운드합니다. 값이 등거리이면 최종 숫자가 짝수가 되도록 값을 라운드합니다. 버려지는 숫자가 다음 왼쪽 위치에 있는 숫자 값의 반보다 큰 숫자를 나타내는 경우, 결과 계수는 1씩 증가합니다. 반보다 작은 숫자를 나타내면 결과 계수는 조정되지 않습니다(즉, 버려지는 숫자는 무시됨). 그렇지 않으면 결과 계수는 가장 오른쪽 숫자가 짝수인 경우 변경되지 않고 홀수이면 1씩 증가합니다(짝수 숫자를 만들기 위해).
- ROUND_HALF_UP은 값을 가장 가까운 값으로 라운드합니다. 값이 등거리이면 값을 위로 라운드합니다. 버려지는 숫자가 다음 왼쪽 위치에 있는 숫자 값의 반이거나 반보다 큰 숫자를 나타내는 경우, 결과 계수는 1씩 증가합니다. 그렇지 않으면 버려지는 숫자는 무시됩니다.

클라이언트에 대한 DECFLOAT 근사값 모드의 값은 SET CURRENT DECFLOAT ROUNDING MODE문을 호출하여 서버의 값과 일치하도록 확인할 수 있습니다. 그러나 이 명령문을 사용하여 서버의 근사값 모드를 변경할 수는 없습니다. CURRENT DECFLOAT ROUNDING MODE의 초기값은 **decflt_rounding** 데이터베이스 구성 매개변수에 따라 결정되며 이 데이터베이스 구성 매개변수 값을 변경하는 방법으로만 변경할 수 있습니다.

CURRENT DEFAULT TRANSFORM GROUP

CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터는 사용자 정의 구조화된 유형 값을 호스트 프로그램과 교환하기 위해 동적 SQL문에 사용되는 변환 그룹의 이름을 식별하는 VARCHAR(18) 값을 지정합니다. 이 특수 레지스터는 정적 SQL문에 사용되거나, 매개변수와 결과를 외부 함수 또는 메소드와 교환할 때 사용되는 변환 그룹을 지정하지 않습니다.

값은 SET CURRENT DEFAULT TRANSFORM GROUP문으로 설정할 수 있습니다. 어떤 값도 설정하지 않을 경우, 이 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 있지 않으면 이 특수 레지스터의 값과 같습니다. 레지스터에 비어 있는 문자열이 들어있는 경우(SET CURRENT DEFAULT TRANSFORM GROUP문을 사용하여 값을 설정하지 않았음), DB2_PROGRAM 변환 그룹이 변환에 사용됩니다. DB2_PROGRAM 변환 그룹이 구조화된 유형 주제에 대해 정의되지 않은 경우, 런타임 시 오류가 발생합니다(SQLSTATE 42741).

예를 들면, 다음과 같습니다.

디폴트 변환 그룹을 MYSTRUCT1로 설정하십시오. MYSTRUCT1 변환에 정의된 TO SQL 및 FROM SQL 함수는 사용자 정의 구조화된 유형 변수를 호스트 프로그램과 교환하는 데 사용됩니다.

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

이 특수 레지스터에 지정된 디폴트 변환 그룹의 이름을 검색하십시오.

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

CURRENT DEGREE 특수 레지스터는 동적 SQL문 실행을 위한 파티션 내 병렬 처리의 등급을 지정합니다. (정적 SQL의 경우 DEGREE 바인드 옵션이 동일한 제어를 제공합니다.) 레지스터의 데이터 유형은 CHAR(5)입니다. 유효한 값은 ANY 또는 1 - 32 767 범위의 정수의 문자열 표시입니다.

SQL문이 동적으로 준비될 때 정수로 표시되는 CURRENT DEGREE의 값이 1인 경우, 해당 명령문 실행 시 파티션 내 병렬 처리를 사용하지 않습니다.

SQL문이 동적으로 준비될 때 정수로 표시되는 CURRENT DEGREE의 값이 1보다 크고 32,767 이하인 경우, 해당 명령문 실행 시 지정된 등급을 사용하여 파티션 내 병렬 처리가 수행될 수 있습니다.

SQL문이 동적으로 준비될 때 CURRENT DEGREE의 값이 ANY인 경우, 해당 명령문 실행 시 데이터베이스 관리 프로그램에서 결정한 등급을 사용하여 파티션 내 병렬 처리가 관련될 수 있습니다.

병렬 처리의 실제 런타임 등급은 다음 중 낮은 것입니다.

- 최대 쿼리 등급(**max_querydegree**) 구성 매개변수의 값
- 응용프로그램 런타임 등급
- SQL문 컴파일 등급

intra_parallel 데이터베이스 관리 프로그램 구성 매개변수가 NO로 설정되는 경우, 최적화를 위해 CURRENT DEGREE 특수 레지스터의 값이 무시되고 명령문은 파티션 내 병렬 처리를 사용하지 않습니다.

값은 SET CURRENT DEGREE문을 호출하여 변경할 수 있습니다.

CURRENT DEGREE의 초기 값은 **dft_degree** 데이터베이스 구성 매개변수에 따라 결정됩니다.

CURRENT EXPLAIN MODE

CURRENT EXPLAIN MODE 특수 레지스터는 적합한 동적 SQL문에 대해 Explain 기능의 동작을 제어하는 VARCHAR(254) 값을 보유합니다. 이 기능은 Explain 정보를 생성하여 Explain 테이블에 삽입합니다. 이 정보는 Explain 스냅샷에 포함되지 않습니다. 가능한 값은 YES, EXPLAIN, NO, REOPT, RECOMMEND INDEXES 및 EVALUATE INDEXES입니다. (정적 SQL의 경우, EXPLAIN 바인드 옵션은 동일한 제어를 제공합니다. PREP 및 BIND 명령의 경우, EXPLAIN 옵션 값은 YES, NO 및 ALL입니다.)

YES Explain 기능을 작동시키고 동적 SQL문에 대한 Explain 정보가 명령문 컴파일 시 캡처되도록 합니다.

EXPLAIN
기능을 작동시키나 동적 명령문은 실행되지 않습니다.

NO Explain 기능을 작동 불가능하게 합니다.

REOPT
명령문이 입력 변수(호스트 변수, 특수 레지스터, 전역 변수 또는 매개변수 표시문자)에 실제 값을 사용하여 다시 최적화될 때에만 Explain 기능을 사용 가능하게 하고 동적(또는 증분식 바인드) SQL문에 대한 Explain 정보가 캡처되게 합니다.

RECOMMEND INDEXES
각 동적 쿼리에 대해 인덱스 세트를 권장합니다. ADVISE_INDEX 테이블을 인덱스 세트로 채웁니다.

EVALUATE INDEXES
권장 인덱스가 존재하는 것처럼 동적 쿼리를 Explain합니다. 인덱스는 ADVISE_INDEX 테이블에서 선택됩니다.

초기 값은 NO입니다. 하지만 SET CURRENT EXPLAIN MODE문을 호출하여 값을 변경할 수 있습니다.

CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터값은 Explain 기능이 호출될 때 상호작용합니다. CURRENT EXPLAIN MODE 특수 레지스터도 EXPLAIN 바인드 옵션과 상호작용합니다. 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))을 CURRENT EXPLAIN MODE 특수 레지스터의 현재 값으로 설정하십시오.

```
VALUES CURRENT EXPLAIN MODE
  INTO :EXPL_MODE
```

CURRENT EXPLAIN SNAPSHOT

CURRENT EXPLAIN SNAPSHOT 특수 레지스터는 Explain 스냅샷 기능의 동작을 제어하는 CHAR(8) 값을 보유합니다. 이 기능은 액세스 플랜 정보, 운영자 비용 및 바인드 시 통계를 포함한 압축된 정보를 생성합니다.

CALL, 복합 SQL(동적), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES 또는 VALUES INTO문만 이 레지스터 값을 고려합니다. 가능한 값은 YES, EXPLAIN, NO, 및 REOPT입니다. 정적 SQL의 경우, EXPLSNAP 바인드 옵션은 동일한 제어를 제공합니다. PREP 및 BIND 명령의 경우, EXPLSNAP 옵션 값은 YES, NO 및 ALL입니다.

YES Explain 스냅샷 기능을 작동시키고 명령문이 컴파일될 때 동적 SQL문의 내부 표현에 대한 스냅샷을 작성합니다.

EXPLAIN
Explain 스냅샷 기능을 작동시키나 동적 명령문은 실행되지 않습니다.

NO Explain 스냅샷 기능을 작동 불가능하게 합니다.

REOPT
명령문이 입력 변수(호스트 변수, 특수 레지스터, 전역 변수 또는 매개변수 표시문자)에 실제 값을 사용하여 다시 최적화될 때에만 Explain 기능을 사용 가능하게 하고 동적(또는 증분식 바인드) SQL문에 대한 Explain 정보가 캡처되게 합니다.

초기 값은 NO입니다. 하지만 SET CURRENT EXPLAIN SNAPSHOT문을 호출하여 값을 변경할 수 있습니다.

CURRENT EXPLAIN SNAPSHOT 및 CURRENT EXPLAIN MODE 특수 레지스터값은 Explain 기능이 호출될 때 상호작용합니다. CURRENT EXPLAIN SNAPSHOT 특수 레지스터도 EXPLSNAP 바인드 옵션과 상호작용합니다.

예: 호스트 변수 EXPL_SNAP (char(8))을 CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 현재 값으로 설정하십시오.

```
VALUES CURRENT EXPLAIN SNAPSHOT
  INTO :EXPL_SNAP
```

CURRENT FEDERATED ASYNCHRONY

CURRENT FEDERATED ASYNCHRONY 특수 레지스터는 동적 SQL문의 실행에 대한 비동기 등급을 지정합니다. (FEDERATED_ASYNCHRONY 바인드 옵션은 정적 SQL에 대해 동일한 제어를 제공합니다.) 레지스터의 데이터 유형은 INTEGER입니다. 올바른 값은 ANY(-1을 나타냄) 또는 0 - 32 767(0과 32,767 포함) 사이의 정수입니다. SQL문이 동적으로 준비될 때 CURRENT FEDERATED ASYNCHRONY의 값은 다음과 같습니다.

- 0. 해당 명령문의 실행은 비동시성을 사용하지 않습니다.
- 0보다 크고 32 767 이하. 해당 명령문의 실행은 지정된 등급을 사용하여 비동시성을 포함할 수 있습니다.
- ANY(-1을 나타냄). 해당 명령문의 시행은 데이터베이스 관리 프로그램이 결정한 등급을 사용하여 비동시성을 포함할 수 있습니다.

CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 값은 SET CURRENT FEDERATED ASYNCHRONY문을 호출하여 변경할 수 있습니다.

명령행 처리기(CLP)를 통해 동적 명령문을 발행한 경우 CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 초기값은 **federated_async** 데이터베이스 관리 프로그램 구성 매개변수에 의해 결정됩니다. 동적 명령문이 바인드 중인 응용프로그램에 포함된 경우 초기값은 FEDERATED_ASYNCHRONY 바인드 옵션에 의해 결정됩니다.

예: 호스트 변수 FEDASYNC (INTEGER)를 CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 값으로 설정하십시오.

```
VALUES CURRENT FEDERATED ASYNCHRONY INTO :FEDASYNC
```

CURRENT IMPLICIT XMLPARSE OPTION

CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터는 직렬화된 XML 데이터가 내재적으로 유효성 확인 없이 DB2 서버에 의해 구문 분석될 때 사용될 공백 조절 옵션을 지정합니다. 내재된 유효성 검증 없는 구문 분석 조작은 SQL문이 XML 호스트 변수 또는 XMLVALIDATE 함수의 인수가 아닌 내재적 또는 명시적으로 유형이 지정된 XML 매개변수 표시문자를 처리 중일 때 발생합니다. 레지스터의 데이터 유형은 VARCHAR(19)입니다.

CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 값은 SET CURRENT IMPLICIT XMLPARSE OPTION문을 호출하여 변경할 수 있습니다. 초기값은 'STRIP WHITESPACE'입니다.

예를 들면, 다음과 같습니다.

CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 값을 검색하여 이름이 CURXMLPARSEOPT인 호스트 변수에 넣으십시오.

```
EXEC SQL VALUES (CURRENT IMPLICIT XMLPARSE OPTION) INTO :CURXMLPARSEOPT;
```

CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터를 'PRESERVE WHITESPACE'로 설정하십시오.

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

그러면 공백이 다음 SQL문이 실행할 때 보존됩니다.

```
INSERT INTO T1 (XMLCOL1) VALUES (?)
```

CURRENT ISOLATION

CURRENT ISOLATION 특수 레지스터는 현재 세션에서 발행되는 모든 동적 SQL 문에 대한 분리 레벨(다른 동시 세션에 상대적인)을 식별하는 CHAR(2) 값을 보유합니다.

가능한 값은 다음과 같습니다.

(공백) 설정되지 않음. 패키지의 분리 속성을 사용하십시오.

UR 언커미트 읽기

CS 커서 안정성

RR 반복 읽기

RS 읽기 안정성

CURRENT ISOLATION 특수 레지스터의 값은 SET CURRENT ISOLATION문으로 변경할 수 있습니다.

SET CURRENT ISOLATION문이 세션 내에서 발행될 때까지 또는 SET CURRENT ISOLATION에 대해 RESET이 지정된 이후, CURRENT ISOLATION 특수 레지스터는 공백으로 설정되어 동적 SQL문에 적용되지 않습니다. 따라서 사용되는 분리 레벨은 동적 SQL문을 발행한 패키지의 분리 속성에서 가져옵니다. SET CURRENT ISOLATION문이 발행된 후, CURRENT ISOLATION 특수 레지스터는 명령문을 발행하는 패키지에 대한 설정과 상관없이 세션에서 컴파일되는 모든 후속 동적 SQL문에 대한 분리 레벨을 제공합니다. 이것은 세션이 종료하거나 SET CURRENT ISOLATION 문이 RESET 옵션과 함께 발행될 때까지 계속 적용됩니다.

예: 호스트 변수 ISOLATION_MODE(CHAR(2))를 CURRENT ISOLATION 특수 레지스터에 현재 저장된 값으로 설정하십시오.

```
VALUES CURRENT ISOLATION
  INTO :ISOLATION_MODE
```

CURRENT LOCALE LC_TIME

CURRENT LOCALE LC_TIME 특수 레지스터는 날짜 및 시간 관련 내장 함수 DAYNAME, MONTHNAME, NEXT_DAY, ROUND, ROUND_TIMESTAMP, TIMESTAMP_FORMAT, TRUNCATE, TRUNC_TIMESTAMP 및 VARCHAR_FORMAT를 포함하는 SQL 인수에 사용되는 로케일을 식별합니다. 이 함수는 *locale-name* 인수를 명시적으로 지정하지 않은 경우 CURRENT LOCALE LC_TIME 값을 사용합니다.

데이터 유형은 VARCHAR(128)입니다.

CURRENT LOCALE LC_TIME의 초기값은 영어(미국)의 『en_US』입니다. 값은 SET CURRENT LOCALE LC_TIME문을 호출하여 변경할 수 있습니다.

CURRENT LOCK TIMEOUT

CURRENT LOCK TIMEOUT 특수 레지스터는 잠금을 확보할 수 없음을 표시하는 오류를 리턴하기 전에 잠금을 대기할 시간(초)을 지정합니다. 이 특수 레지스터는 행, 테이블, 인덱스 키, XML 블록 및 XML 경로(XPath) 잠금에 영향을 줍니다. 레지스터의 데이터 유형은 INTEGER입니다.

CURRENT LOCK TIMEOUT 특수 레지스터의 유효한 값은 -1부터 32767까지의 정수입니다. 이 특수 레지스터는 널(NULL) 값으로 설정할 수도 있습니다. 값 -1은 시간 종료가 발생하지 않으며 잠금이 해제되거나 교착 상태가 발견될 때까지 응용프로그램이 대기하도록 지정합니다. 값 0은 응용프로그램이 잠금을 대기하지 않도록 지정합니다. 잠금을 확보할 수 없으면 오류가 즉시 리턴됩니다.

CURRENT LOCK TIMEOUT 특수 레지스터의 값은 SET CURRENT LOCK TIMEOUT문을 호출하여 변경할 수 있습니다. 초기값은 널(NULL)이며, 이 경우, 잠금을 대기할 때 **locktimeout** 데이터베이스 구성 매개변수의 현재 값이 사용되고 이 값은 특수 레지스터에 대해 리턴됩니다.

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터는 동적 SQL 처리를 최적화할 때 고려할 수 있는 테이블 유형을 식별하는 VARCHAR(254) 값을 지정합니다. 구체화된 쿼리 테이블은 정적 Embedded SQL 쿼리가 고려하지 않습니다.

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION의 초기 값은 SYSTEM입니다. SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION문으로 값을 변경할 수 있습니다.

CURRENT MDC ROLLOUT MODE

CURRENT MDC ROLLOUT MODE 특수 레지스터는 롤아웃 처리에 대해 규정하는 DELETE문의 다차원적으로 클러스터된(MDC) 테이블에 대한 동작을 지정합니다.

이 레지스트리의 디폴트값은 DB2_MDC_ROLLOUT 레지스트리 변수에 의해 판별됩니다. SET CURRENT MDC ROLLOUT MODE문을 호출하여 값을 변경할 수 있습니다. CURRENT MDC ROLLOUT MODE 특수 레지스터를 특정 값으로 설정하는 경우 롤아웃에 대해 규정하는 연속 DELETE문의 실행 동작에 영향이 미칠 수 있습니다. DELETE문은 변경 동작에 대해 다시 컴파일하지 않아도 됩니다.

CURRENT OPTIMIZATION PROFILE

CURRENT OPTIMIZATION PROFILE 특수 레지스터는 최적화를 위해 동적으로 준비되는 DML 명령문이 사용할 최적화 프로파일의 자격이 부여된 이름을 지정합니다.

초기값은 널(NULL) 값입니다. 값은 SET CURRENT OPTIMIZATION PROFILE 문을 호출하여 변경할 수 있습니다. 스키마 이름으로 규정되지 않는 최적화 프로파일은 CURRENT DEFAULT SCHEMA 특수 레지스터의 값으로 내재적으로 자격이 부여됩니다.

예 1: 최적화 프로파일을 'JON.SALES'로 설정하십시오.

```
SET CURRENT OPTIMIZATION PROFILE = JON.SALES
```

예 2: 이 연결에 대한 최적화 프로파일 이름의 현재 값을 가져오십시오.

```
VALUES (CURRENT OPTIMIZATION PROFILE) INTO :PROFILE
```

CURRENT PACKAGE PATH

CURRENT PACKAGE PATH 특수 레지스터는 SQL문 실행 시 필요한 패키지에 대한 참조를 분석할 때 사용할 경로를 식별하는 VARCHAR(4096) 값을 지정합니다.

값은 비어 있거나 공백인 문자열, 또는 큰따옴표로 구분되고 쉼표로 분리되는 하나 이상의 스키마 이름 목록일 수 있습니다. 문자열의 파트로서 나타나는 모든 큰따옴표는 분리 ID의 일반적인 관례대로 두 개의 큰따옴표로 표시되어야 합니다. 분리문자 및 쉼표도 특수 레지스터의 길이에 기여합니다.

이 특수 레지스터는 정적 및 동적문 모두에 적용됩니다.

사용자 정의 함수(UDF), 메소드 또는 프로시저에서 CURRENT PACKAGE PATH의 초기값은 호출 응용프로그램에서 상속받습니다. 기타 컨텍스트에서 CURRENT PACKAGE PATH의 초기값은 비어 있는 문자열입니다. 값은 응용프로그램 프로세스가 SET CURRENT PACKAGE PATH문을 사용하여 스키마 목록을 명시적으로 지정한 경우에만 스키마 목록입니다.

예를 들면, 다음과 같습니다.

응용프로그램은 다중 SQLJ 패키지(스키마 SQLJ1 및 SQLJ2에서) 및 JDBC 패키지(DB2JAVA 스키마에서)를 사용 중입니다. CURRENT PACKAGE PATH 특수 레지스터를 설정하여 SQLJ1, SQLJ2 및 DB2JAVA를 순서대로 점검하십시오.

```
SET CURRENT PACKAGE PATH = "SQLJ1", "SQLJ2", "DB2JAVA"
```

HVPKLIST 호스트 변수를 CURRENT PACKAGE PATH 특수 레지스터에 현재 저장된 값으로 설정하십시오.

```
VALUES CURRENT PACKAGE PATH INTO :HVPKLIST
```

CURRENT PATH

CURRENT PATH(또는 CURRENT_PATH) 특수 레지스터는 동적으로 준비된 SQL 문에서 규정되지 않은 함수 이름, 프로시저 이름, 데이터 유형 이름, 전역 변수 이름 및 모듈 오브젝트 이름을 분석할 때 사용될 SQL 경로를 식별하는 VARCHAR(2048) 값을 지정합니다. CURRENT FUNCTION PATH는 CURRENT PATH의 동의어입니다. 초기 값은 아래에 지정된 디폴트값입니다. 정적 SQL의 경우, FUNCPATH 바인드 옵션은 함수 및 데이터 유형 분석에 사용되는 SQL 경로를 제공합니다.

CURRENT PATH 특수 레지스터는 큰따옴표로 묶이고 쉼표로 구분된 하나 이상의 스키마 이름 목록을 포함합니다. 예를 들어, 데이터베이스 관리 프로그램이 FERMAT 스키마를 먼저 검사한 후 XGRAPHIC 스키마를 검사하고 마지막으로 SYSIBM 스키마를 검사하도록 지정하는 SQL 경로는 CURRENT PATH 특수 레지스터에서 다음과 같이 리턴됩니다.

```
"FERMAT","XGRAPHIC","SYSIBM"
```

디폴트값은 『SYSIBM』, 『SYSFUN』, 『SYSPROC』, 『SYSIBMADM』 및 *X*입니다. 여기서 *X*는 큰따옴표로 분리된 USER 특수 레지스터의 값입니다. SET CURRENT PATH 문을 호출하여 값을 변경할 수 있습니다. 스키마 SYSIBM은 지정될 필요가 없습니다. SQL 경로에 포함되어 있지 않을 경우, 이것은 내재적으로 첫 번째 스키마로 간주됩니다. 내재적으로 가정되면 SYSIBM이 2048바이트를 차지하지 않습니다.

스키마 이름으로 규정되지 않은 데이터 유형은 규정되지 않은 동일한 이름의 데이터 유형을 포함하는 SQL 경로에 있는 첫 번째 스키마로 내재적으로 규정됩니다. CREATE TYPE(구별), CREATE FUNCTION, COMMENT 및 DROP문에서 설명된 대로 이 규칙의 예외가 있습니다.

예: CURRENT PATH 특수 레지스터에는 스키마 이름이 포함되어 있기 때문에 SYSCAT.ROUTINES 카탈로그 뷰를 사용하여 루틴 이름을 규정하지 않고 호출할 수 있는 사용자 정의 루틴을 모두 찾으십시오.

```
SELECT ROUTINENAME, ROUTINESCHEMA FROM SYSCAT.ROUTINES
  WHERE POSITION (ROUTINESCHEMA, CURRENT PATH, CODEUNITS16) <> 0
```

CURRENT QUERY OPTIMIZATION

CURRENT QUERY OPTIMIZATION 특수 레지스터는 동적 SQL문을 바인드할 때 데이터베이스 관리 프로그램이 수행하는 쿼리 최적화의 클래스를 제어하는 INTEGER 값을 지정합니다. QUERYOPT 바인드 옵션은 정적 SQL문에 대한 쿼리 최적화 클래스를 제어합니다. 가능한 값의 범위는 0 - 9입니다. 예를 들어 쿼리 최적화 클래스가 0(최소 최적화)으로 설정되는 경우 특수 레지스터의 값은 0입니다. 디폴트값은 **dft_queryopt** 데이터베이스 구성 매개변수로 판별됩니다. SET CURRENT QUERY OPTIMIZATION문을 호출하여 값을 변경할 수 있습니다.

예: SYSCAT.PACKAGES 카탈로그 뷰를 사용하여 CURRENT QUERY OPTIMIZATION 특수 레지스터의 현재 값과 동일한 설정을 사용하여 바인드된 모든 플랜을 찾으십시오.

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
  WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

CURRENT REFRESH AGE

CURRENT REFRESH AGE 특수 레지스터는 데이터 유형 DECIMAL(20,6)을 갖는 시간소인 지속 기간 값을 지정합니다. 특정 시간소인이 지정된 이벤트가 캐시된 데이터 오브젝트에 발행한 이후(예를 들어 REFRESH TABLE문이 시스템이 유지보수하는 REFRESH DEFERRED 구체화된 쿼리 테이블에서 처리됨) 최대 지속기간이므로, 캐시된 데이터 오브젝트를 사용하여 쿼리의 처리를 최적화할 수 있습니다. CURRENT REFRESH AGE의 값이 99 999 999 999 999이고, 쿼리 최적화 클래스가 5 이상인 경우, 동적 SQL 쿼리 처리를 최적화할 때 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION에 지정되는 테이블의 유형이 고려됩니다.

CURRENT REFRESH AGE 값은 0 또는 99 999 999 999 999여야 합니다. 초기 값은 0입니다. SET CURRENT REFRESH AGE문을 호출하여 값을 변경할 수 있습니다.

CURRENT SCHEMA

CURRENT SCHEMA(또는 CURRENT_SCHEMA) 특수 레지스터는 동적으로 준비된 SQL문에서 데이터베이스 오브젝트 참조(해당되는 경우)를 규정하는 데 사용되는 스키마 이름을 식별하는 VARCHAR(128) 값을 지정합니다. z/OS용 DB2와의 호환성을 위해 CURRENT SCHEMA 대신 CURRENT SQLID(또는 CURRENT_SQLID)를 지정할 수 있습니다.

CURRENT SCHEMA의 초기 값은 현재 세션 사용자의 권한 부여 ID입니다. SET SCHEMA문을 호출하여 값을 변경할 수 있습니다.

QUALIFIER 바인드 옵션은 적용 가능한 경우 정적 SQL문에 대한 데이터베이스 오브젝트 참조를 규정하는 데 사용되는 스키마 이름을 제어합니다.

예: 오브젝트 자격에 대한 스키마를 'D123'으로 설정하십시오.

```
SET CURRENT SCHEMA =  'D123'
```

CURRENT SERVER

CURRENT SERVER(또는 CURRENT_SERVER) 특수 레지스터는 현재 응용프로그램 서버를 식별하는 VARCHAR(18) 값을 지정합니다. 레지스터에는 별명이 아니라 응용프로그램 서버의 실제 이름이 들어있습니다.

CURRENT SERVER는 CONNECT문을 통해 변경할 수 있지만 특정 조건에서만 가능합니다.

루틴 내의 SQL문에 사용될 때 CURRENT SERVER는 호출 명령문에서 상속받지 않습니다.

예: 호스트 변수 APPL_SERVE(VARCHAR(18))를 응용프로그램이 연결되는 응용프로그램 서버의 이름으로 설정하십시오.

```
VALUES CURRENT SERVER INTO :APPL_SERVE
```

CURRENT TIME

CURRENT TIME(또는 CURRENT_TIME) 특수 레지스터는 SQL문이 응용프로그램 서버(AS)에서 실행될 때의 시간 읽기를 기준으로 시간을 지정합니다. 이 특수 레지스터가 단일 SQL문에서 두 번 이상 사용되거나 단일 명령문에서 CURRENT DATE나 CURRENT TIMESTAMP와 함께 사용된 경우, 모든 값은 단일 시계 읽기를 기준으로 합니다.

루틴 내의 SQL문에서 사용될 때 CURRENT TIME은 호출 명령문에서 상속받지 않습니다.

페더레이티드 시스템에서는 데이터 소스용 쿼리에 CURRENT TIME을 사용할 수 있습니다. 쿼리가 처리될 때 리턴되는 시간은 데이터 소스가 아닌 페더레이티드 서버의 CURRENT TIME 레지스터로부터 확보됩니다.

예: DB2 CLP에서 다음 명령을 실행하여 현재 시간을 알 수 있습니다.

```
db2 values CURRENT TIME
```

예: CL_SCHED 테이블을 사용하여 오늘 늦게 시작하는(STARTING) 모든 수업(CLASS_CODE)을 선택하십시오. 오늘 수업의 DAY 컬럼에는 3의 값이 있습니다.

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME AND DAY = 3
```

CURRENT TIMESTAMP

CURRENT TIMESTAMP(또는 CURRENT_SERVER) 특수 레지스터는 SQL문이 응용프로그램 서버(AS)에서 실행될 때의 시간을 기준으로 시간소인을 지정합니다. 이 특수 레지스터가 단일 SQL문에서 두 번 이상 사용된 경우 또는 단일 명령문에서 CURRENT DATE나 CURRENT TIME과 함께 사용된 경우, 모든 값은 단일 시계 읽기를 기준으로 합니다. 별도의 CURRENT TIMESTAMP 특수 레지스터 요청이 동일한 값을 리턴할 수 있습니다. 고유 값이 필요하면 GENERATE_UNIQUE 함수, 시퀀스 또는 ID 컬럼 사용을 고려하십시오.

특정 정밀도를 가진 시간소인을 원하면, CURRENT TIMESTAMP(*정수*)로 특수 레지스터를 참조할 수 있습니다(여기서, *정수*는 0 - 12 범위일 수 있음). 디폴트 정밀도는 6입니다. 클럭 눈금의 정밀도는 플랫폼에 따라 다르며 검색된 클럭 눈금의 정밀도가 요청된 정밀도보다 작은 경우 결과 값이 0으로 채워집니다.

루틴 내의 SQL문에 사용될 때 CURRENT TIMESTAMP는 호출 명령문에서 상속받지 않습니다.

페더레이티드 시스템에서는 데이터 소스용 쿼리에 CURRENT TIMESTAMP를 사용할 수 있습니다. 쿼리가 처리될 때 리턴되는 시간소인은 데이터 소스가 아닌 페더레이티드 서버의 CURRENT TIMESTAMP 레지스터로부터 확보됩니다.

또한 CURRENT TIMESTAMP(0)의 동의어로 SYSDATE를 지정할 수 있습니다.

예: IN_TRAY 테이블에 한 행을 삽입하십시오. RECEIVED 컬럼의 값은 행이 삽입된 때를 나타내는 시간소인 값이어야 합니다. 다른 세 가지 컬럼에 대한 값은 호스트 변수 SRC(char(8)), SUB(char(64)) 및 TXT(VARCHAR(200))로 구한 값입니다.

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

CURRENT TIMEZONE(또는 CURRENT_TIMEZONE) 특수 레지스터는 UTC(세계 표준시, 일반적으로 GMT로 알려짐)와 응용프로그램 서버의 로컬 시간과의 차이를 지정합니다. 차이는 시간 지속 기간(10진수로서, 처음 두 자릿수는 시간, 다음 두 자릿수는 분, 마지막 두 자릿수는 초임)으로 표현됩니다. 시간은 -24부터 24까지(24는 제외)입니다. 로컬 시간에서 CURRENT TIMEZONE을 빼면 로컬 시간이 UTC로 변환됩니다. 시간은 SQL문이 실행되는 순간에 운영 체제 시간으로부터 계산됩니다. (CURRENT TIMEZONE 값은 C 런타임 함수로부터 판별됩니다.)

CURRENT TIMEZONE 특수 레지스터는 DECIMAL(6,0) 데이터 유형의 표현식이 사용될 때마다(예를 들어, 시간 및 시간소인 산술에서) 사용할 수 있습니다.

루틴 내의 SQL문에서 사용될 때 CURRENT TIMEZONE은 호출 명령문에서 상속받지 않습니다.

예: RECEIVED 컬럼에 대한 UTC 시간소인을 사용하여 레코드를 IN_TRAY 테이블에 삽입하십시오.

```
INSERT INTO IN_TRAY VALUES (
  CURRENT TIMESTAMP - CURRENT TIMEZONE,
  :source,
  :subject,
  :notetext )
```

CURRENT USER

CURRENT USER(또는 CURRENT_USER) 특수 레지스터는 명령문 권한 부여에 사용될 권한 부여 ID를 지정합니다. 정적 SQL문의 경우 값은 패키지가 바인드될 때 사용되는 권한 부여 ID를 나타냅니다. 동적 SQL문의 경우 값은 DYNAMICRULES(RUN) 바인드 옵션으로 바인드된 패키지에 대한 SESSION_USER 특수 레지스터의 값과 동일합니다. 레지스터의 데이터 유형은 VARCHAR(128)입니다.

예: 스키마가 CURRENT USER 특수 레지스터의 값과 일치하는 테이블 이름을 선택하십시오.

```
SELECT TABNAME FROM SYSCAT.TABLES
  WHERE TABSCHEMA = CURRENT USER AND TYPE = 'T'
```

이 명령문이 정적 SQL문으로 실행되는 경우 스키마 이름이 명령문을 포함하는 패키지의 바인더와 일치하는 테이블을 리턴합니다. 이 명령문이 동적 SQL문으로 실행되는 경우 스키마 이름이 SESSION_USER 특수 레지스터의 현재 값과 일치하는 테이블을 리턴합니다.

SESSION_USER

SESSION_USER 특수 레지스터는 현재 세션에 사용될 권한 부여 ID를 지정합니다. 이 레지스터의 값이 DYNAMICRULES 실행 동작이 패키지에 적용될 때 동적 SQL문의 권한 부여 검사에 사용됩니다. 레지스터의 데이터 유형은 VARCHAR(128)입니다.

새 연결에 사용되는 SESSION_USER의 초기값은 SYSTEM_USER 특수 레지스터의 값과 동일합니다. 값은 SET SESSION AUTHORIZATION문을 호출하여 변경할 수 있습니다.

SESSION_USER는 USER 특수 레지스터의 동의어입니다.

예: 동적 SQL을 사용하여 실행할 수 있는 루틴을 판별하십시오. 루틴을 호출하는 동적 SQL문을 발행할 패키지에 대해 DYNAMICRULES 실행 동작이 적용된다고 가정하십시오.

```
SELECT SCHEMA, SPECIFICNAME FROM SYSCAT.ROUTINEAUTH
  WHERE GRANTEE = SESSION_USER
  AND EXECUTEAUTH IN ('Y', 'G')
```

SYSTEM_USER

SYSTEM_USER 특수 레지스터는 데이터베이스에 연결된 사용자의 권한 부여 ID를 지정합니다. 이 레지스터의 값은 다른 권한 부여 ID를 갖는 사용자로 연결해야 변경할 수 있습니다. 레지스터의 데이터 유형은 VARCHAR(128)입니다.

SET SESSION AUTHORIZATION문의 설명에 있는 『예』를 참조하십시오.

USER

USER 특수 레지스터는 응용프로그램이 데이터베이스에서 시작할 때 데이터베이스 관리 프로그램으로 전달되는 런타임 권한 부여 ID를 지정합니다. 레지스터의 데이터 유형은 VARCHAR(128)입니다.

루틴 내의 SQL문에 사용될 때 USER는 호출 명령문에서 상속받지 않습니다.

예: IN_TRAY 테이블에서 사용자가 해당 테이블에 위치시킨 모든 참고를 선택하십시오.

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

전역 변수

전역 변수는 SQL문을 통해 액세스하고 수정할 수 있는 이름 지정된 메모리 변수입니다.

전역 변수를 사용하면 데이터 전송을 지원하기 위한 응용프로그램 논리 없이 SQL문 사이에 관계형 데이터를 공유할 수 있습니다. GRANT(전역 변수 특권) 및 REVOKE(전역 변수 특권) 문을 통해 전역 변수에 대한 액세스를 제어할 수 있습니다.

DB2는 작성된 세션 전역 변수를 지원합니다. *세션 전역 변수*는 특정 세션과 연관되며 해당 세션에 고유한 값을 포함합니다. 작성된 세션 글로벌 변수는 변수가 정의된 데이터베이스에 대해 실행 중인 활성 SQL문에 사용 가능합니다. 세션 전역 변수는 두 개 이상의 세션과 연관될 수 있지만 해당 값은 각 세션에 특정합니다. 작성된 세션 전역 변수와, 이 전역 변수와 연관되는 특권은 시스템 카탈로그에서 정의됩니다.

*사용자 정의 세션 전역 변수*는 SQL 데이터 정의 명령문을 사용하여 작성되고 카탈로그의 데이터베이스 관리 프로그램에 등록되는 전역 변수입니다. 전역 변수는 작성된 스키마 또는 추가 또는 발행된 모듈에 있습니다. *스키마 변수*는 CREATE VARIABLE 문을 사용하여 작성됩니다. 자세한 정보는 ″CREATE VARIABLE″를 참조하십시오. *모듈 변수*는 ALTER MODULE문의 ADD *module-variable-definition*절이나 PUBLISH *module-variable-definition*절을 사용하여 작성됩니다. 자세한 정보는 “ALTER MODULE”을 참조하십시오.

전역 변수 참조 분석은 전역 변수가 참조되는 컨텍스트 및 전역 변수 이름의 규정화 방법에 따라 달라집니다. 전역 변수로 사용되는 변수 참조는 참조 컨텍스트 및 해당 컨텍스트에서 참조가 규정화되는 방법에 따라 SQL 변수, SQL 매개변수 또는 컬럼 이름으로도 분석될 수 있습니다. 다음에서는 전역 변수 참조가 SQL 변수, SQL 매개변수 또는 컬럼 이름으로 분석되지 않는 것으로 간주됩니다.

- 전역 변수 이름이 규정화된 경우 데이터베이스 관리 프로그램이 다음 단계를 사용하여 분석을 수행합니다.
  1. 전역 변수 참조가 모듈내에서 시작되고 규정자가 전역 변수가 참조되는 모듈 이름과 일치하는 경우 모듈은 일치하는 모듈 변수에 대해 검색됩니다. 규정자가 단일 ID이면, 모듈 이름을 일치시킬 때 모듈의 스키마 이름이 무시됩니다. 규정자가 두 파트의 ID이면, 일치를 판별할 때 스키마 규정 모듈 이름과 비교합니다. 모듈 변수가 참조에서 규정되지 않은 전역 변수 이름과 일치하는 경우 분석이 완료됩니다. 규정자가 일치하지 않거나 일치하는 모듈 변수가 없는 경우 분석은 다음 단계로 계속됩니다.
  2. 규정자가 스키마 이름이고 해당 스키마가 일치하는 스키마 변수에 대해 검색됩니다. 스키마 변수가 참조에서 규정되지 않은 전역 변수 이름과 일치하는 경우 분석이 완료됩니다. 스키마가 없거나 스키마에 일치하는 스키마 변수가 없는 경우

또는 규정자가 첫 번째 단계의 모듈 이름과 일치하는 경우에는 오류가 리턴됩니다(SQLSTATE 42703). 그 이외의 경우에는 분석이 다음 단계로 진행됩니다.

3. 규정자가 모듈 이름입니다.
   - 모듈 이름이 스키마 이름으로 규정되고 해당 모듈이 일치하는 발행된 모듈 변수에 대해 검색됩니다.
   - 모듈 이름이 스키마 이름으로 규정되지 않으면, 모듈의 스키마는 모듈 이름과 일치하는 SQL 경로에 있는 첫 번째 스키마입니다. 찾은 경우 해당 모듈은 일치하는 발행된 모듈 변수에 대해 검색됩니다.
   - SQL 경로를 사용하여 모듈을 찾지 못한 경우 전역 변수 규정자의 이름에 일치하는 모듈 공유 별명의 존재가 고려됩니다. 찾은 경우 모듈 공용 별명에 연관된 모듈이 일치하는 발행된 모듈 변수에 대해 검색됩니다.

   발행된 모듈 변수가 전역 변수 참조에서 규정되지 않은 전역 변수 이름과 일치하는 경우 분석이 완료됩니다. 일치하는 모듈이 없거나 일치하는 모듈에 일치하는 모듈 변수가 없는 경우 오류가 리턴됩니다(SQLSTATE 42703).

- 전역 변수 이름이 규정되지 않은 경우 데이터베이스 관리 프로그램이 다음 단계를 사용하여 분석을 수행합니다.
  1. 규정되지 않은 전역 변수 참조가 모듈 오브젝트에서 시작된 경우 모듈은 일치하는 모듈 변수에 대해 검색됩니다. 모듈 변수가 참조에서 전역 변수 이름과 일치하는 경우 분석이 완료됩니다. 일치하는 모듈 변수가 없는 경우 분석은 다음 단계로 계속됩니다.
  2. SQL 경로의 스키마는 일치하는 스키마 변수에서, 왼쪽에서 오른쪽 순으로 검색됩니다. 스키마 변수가 참조에서 전역 변수 이름과 일치하는 경우 분석이 완료됩니다.

  2단계를 완료한 후에 일치하는 전역 변수가 없는 경우 오류가 리턴됩니다(SQLSTATE 42703).

전역 변수가 SQL문이나 트리거, 뷰 또는 루틴에서 참조되는 경우 완전한 전역 변수 이름의 종속성이 명령문이나 오브젝트에 대해 기록됩니다. 전역 변수에 필요한 권한 부여는 정의 위치 및 사용 방법에 따라 다릅니다.

- 스키마 변수를 참조하고 값을 검색하는 SQL문의 권한 부여 ID에는 전역 변수에 대한 READ 특권이 있어야 합니다.
- 스키마 변수를 참조하고 값을 지정하는 SQL문의 권한 부여 ID에는 전역 변수에 대한 WRITE 특권이 있어야 합니다.
- 모듈 변수를 참조하고 값을 검색하거나 값을 지정하는 SQL문의 권한 부여 ID에는 전역 변수의 모듈에 대한 EXECUTE 특권이 있어야 합니다.

전역 변수는 결정적이지 않아도 되는 표현식 내에서 참조할 수 있습니다. 결정적 표현식은 전역 변수를 사용할 수 없는 다음과 같은 상황에서 필요합니다.

- 점검 제한조건
- 생성된 컬럼의 정의
- 새로 고침 즉시 구체화된 쿼리 테이블(MQT)

전역 변수 값은 EXECUTE, FETCH, SET, SELECT INTO 또는 VALUES INTO 문을 사용하여 변경할 수 있습니다. 또한 CALL 문에서 OUT 또는 INOUT 매개변수의 인수인 경우에도 변경할 수 있습니다.

전역 변수의 데이터 유형이 커서 유형인 경우, 전역 커서 변수의 밑줄 긋기 커서는 *cursor-variable-name*이 지정될 수 있는 어느 곳에서나 참조될 수 있습니다.

전역 변수의 데이터 유형이 행 유형인 경우, 전역 행 변수의 필드는 필드와 동일한 유형을 갖는 전역 변수를 참조할 수 있는 어디에서나 참조할 수 있습니다. 필드 이름을 규정하는 전역 변수 이름은 다른 모든 전역 변수와 동일한 방식으로 해결됩니다.

함수

*함수*는 괄호 안에 들어 있는 하나 이상의 피연산자가 뒤에 붙는 함수 이름이 표시하는 조작입니다. 예를 들어, TIMESTAMP 함수는 DATE 및 TIME 유형의 입력 데이터 값을 전달할 수 있으며, 결과는 TIMESTAMP입니다. 함수는 내장 함수 또는 사용자 정의 함수가 될 수 있습니다.

- *내장 함수*는 데이터베이스 관리 프로그램에 제공되며, 하나의 결과 값을 리턴하고 SYSIBM 스키마의 일부로 식별됩니다. 이러한 함수에는 집계 함수(예: AVG), 연산자 함수(예: +), 캐스팅 함수(예: DECIMAL) 및 스칼라 함수(예: CEILING)가 포함됩니다.
- *사용자 정의 함수(UDF)*는 SQL 데이터 정의 명령문을 사용하여 작성되고 카탈로그의 데이터베이스 관리 프로그램에 등록되는 함수입니다. *사용자 정의 스키마 함수*는 CREATE FUNCTION 명령문을 사용하여 작성됩니다. 자세한 정보는 『CREATE FUNCTION』을 참조하십시오. *사용자 정의 스키마 함수* 중 한 세트는 SYSFUN 스키마에서 데이터베이스 관리 프로그램에 제공됩니다. *사용자 정의 모듈 함수*는 ALTER MODULE ADD FUNCTION 또는 ALTER MODULE PUBLISH FUNCTION 문을 사용하여 생성됩니다. 자세한 정보는 『ALTER MODULE』을 참조하십시오. *사용자 정의 모듈 함수*의 세트는 SYSIBMADM 스키마의 모듈 세트에서 데이터베이스 관리 프로그램과 함께 제공됩니다. *사용자 정의 함수(UDF)*는 작성된 스키마에 있거나 함수가 추가 또는 발행된 모듈에 있습니다.

  사용자 정의 함수는 데이터베이스 엔진 자체에 적용될 수 있는 함수 정의(사용자 또는 써드 파티 벤더가 제공)를 추가하여 데이터베이스 시스템의 기능을 확장합니다. 데이터베이스 함수를 확장하면 데이터베이스는 응용프로그램이 사용하는 엔진에서 동일한 함수를 이용할 수 있으며, 응용프로그램과 데이터베이스 사이에 더 많은 상승 효과를 제공합니다.

외부, SQL 및 전래 사용자 정의 함수

사용자 정의 함수는 외부 함수, SQL 함수 또는 전래 함수일 수 있습니다. *외부 함수*는 오브젝트 코드 라이브러리를 참조하는 데이터베이스와, 함수 호출시 실행될 해당 라이브러리 내의 함수에 정의됩니다. 외부 함수는 집계 함수가 될 수 없습니다. SQL 함수는 최소 하나의 RETURN문을 포함하는 SQL문만 사용한 데이터베이스에 정의됩니다. 이 함수는 스칼라 값, 행 또는 테이블을 리턴할 수 있습니다. SQL 함수는 집계 함수가 될 수 없습니다. *전래 함수*는 데이터베이스에 이미 알려진 다른 내장 함수나 사용자 정의 함수를 참조하는 데이터베이스에 정의됩니다. 전래 함수는 스칼라 함수 또는 집계 함수가 될 수 있습니다. 사용자 정의 유형으로 기존 함수를 지원하는 데 유용합니다.

스칼라, 집계, 행 및 테이블 사용자 정의 함수

각 사용자 정의 함수(UDF)는 스칼라, 집계, 행 또는 테이블 함수로 분류되기도 합니다. *스칼라 함수*는 호출될 때마다 단일 값으로 응답을 리턴하는 함수입니다. 예를 들어, 내장 함수 SUBSTR()은 스칼라 함수입니다. 스칼라 UDF는 외부 또는 전래 함수일 수 있습니다.

*집계 함수*는 개념적으로 유사 값 세트(한 컬럼)에 전달되고 단일 값으로 응답을 리턴하는 함수입니다. 집계 함수의 예로는 내장 함수 AVG()가 있습니다. 외부 컬럼 UDF는 DB2에 정의할 수 없으나 내장 집계 함수 중 하나를 기반으로 하는 컬럼 UDF는 정의할 수 있습니다. 이는 구별 유형에 유용합니다. 예를 들어, 기본 유형 INTEGER로 구별 유형 SHOESIZE가 정의되어 있는 경우, 내장 함수 AVG(INTEGER)에 근거하는 UDF AVG(SHOESIZE)를 정의할 수 있으며 이는 집계 함수가 됩니다.

*행 함수*는 하나의 행 값을 리턴하는 함수입니다. 행 함수는 행 표현식이 지원되는 컨텍스트에 사용될 수 있습니다. 또한 이 함수는 구조화된 유형의 속성 값을 하나의 행의 값에 맵핑하여 변환 함수로 사용할 수도 있습니다. 행 함수는 SQL 함수로 정의되어야 합니다.

*테이블 함수*는 테이블을 참조하는 SQL문에 테이블을 리턴하는 함수입니다. 테이블 함수는 SELECT문의 FROM절에서만 참조할 수 있습니다. 이러한 함수는 SQL 언어 처리 능력을 DB2 데이터가 아닌 데이터에 적용하거나 해당 데이터를 DB2 테이블로 변환하는 데 사용할 수 있습니다. 테이블 함수는 파일을 읽고 웹에서 데이터를 가져오거나, 또는 Lotus Notes® 데이터베이스에 액세스하여 결과 테이블을 리턴할 수 있습니다. 이 정보는 데이터베이스의 다른 테이블과 조인할 수 있습니다. 테이블 함수는 외부 함수 또는 SQL 함수로 정의될 수 있습니다. 테이블 함수는 전래 함수가 될 수 없습니다.

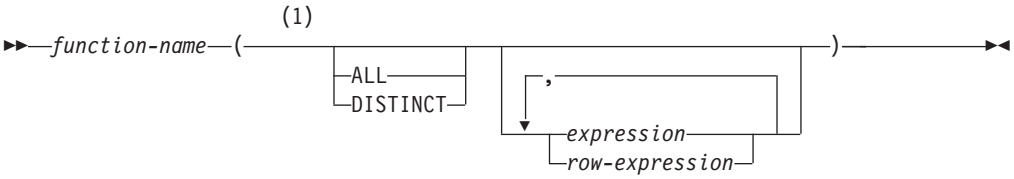
함수 시그니처

스키마 함수는 스키마 이름, 함수 이름, 매개변수 수 및 해당 매개변수의 데이터 유형으로 식별됩니다. 모듈 함수는 스키마 이름, 모듈 이름, 함수 이름, 매개변수 수 및 해당 매개변수의 데이터 유형으로 식별됩니다. 이 스키마 함수 또는 모듈 함수의 ID를 *함수 시그니처*라고 하며, 데이터베이스 내에서 고유해야 합니다(예: TEST.RISK (INTEGER)). 매개변수 수나 매개변수의 데이터 유형이 다른 경우, 스키마 또는 모듈에 동일한 이름의 함수가 둘 이상 있을 수 있습니다. 매개변수 수가 동일한 여러 함수 인스턴스가 있는 함수 이름을 *오버로드 함수*라고 합니다. 함수 이름은 스키마 내에서 오버로드될 수 있는데, 이 경우 스키마에 동일한 수의 매개변수가 있는 해당 이름의 함수가 둘 이상 있게 됩니다. 마찬가지로, 함수 이름은 모듈 내에서 오버로드될 수 있는데, 이 경우 모듈에 동일한 수의 매개변수가 있는 해당 이름의 함수가 둘 이상 있게 됩니다. 이러한 함수에서는 매개변수 데이터 유형이 서로 달라야 합니다. 또한 함수는 SQL 경로의 스키마 간에도 오버로드될 수 있는데, 이 경우 SQL 경로의 다른 스키마에 동

일한 수의 매개변수가 있는 해당 이름의 함수가 둘 이상 있게 됩니다. 이러한 함수에서는 매개변수 데이터 유형이 서로 다르지 않아도 됩니다.

함수 호출

함수에 대한 각 참조는 다음 구문을 준수합니다.



주:

1 ALL 또는 DISTINCT 키워드는 집계 함수 또는 집계 함수의 소스인 사용자 정의 함수(UDF)에만 지정될 수 있습니다.

위 구문에서 *expression* 및 *row-expression*에는 집계 함수를 포함시킬 수 없습니다. *expression*의 기타 규칙은 『표현식』을 참조하십시오.

함수는 괄호로 묶인 인수 목록이 뒤에 오는 규정된 함수 이름 또는 규정되지 않은 함수 이름을 허용 가능한 컨텍스트에서 참조하여 호출됩니다. 함수 이름의 가능한 규정자는 다음과 같습니다.

- 스키마 이름
- 규정되지 않은 모듈 이름
- 스키마 규정 모듈 이름

함수 호출 시 사용되는 규정자가 일치 기능 검색에 사용되는 범위를 판별합니다.

- 스키마가 규정된 모듈 이름을 규정자로 사용하면 범위는 지정된 모듈이 됩니다.
- 규정자로 한 개의 ID가 사용된 경우 범위에는 다음이 포함됩니다.
  - 규정자에 일치하는 스키마
  - 다음 모듈 중 하나:
    - 호출 모듈 - 호출 모듈 이름이 규정자에 일치하는 경우
    - 규정자에 일치하는 SQL 경로에서 스키마의 첫 번째 모듈
- 규정자가 사용되지 않으면 범위에 SQL 경로의 스키마가 포함되며 모듈 오브젝트 내에서 함수가 호출되는 경우에는 함수가 호출된 동일한 모듈이 포함됩니다.

정적 SQL문의 경우, SQL 경로는 FUNCPATH 바인드 옵션을 사용하여 지정됩니다. 동적 SQL문의 경우, SQL 경로는 CURRENT PATH 특수 레지스터의 값입니다.

함수가 호출되면, 데이터베이스 관리 프로그램은 실행할 함수를 판별해야 합니다. 이 프로세스를 *함수 결정*이라고 하며 내장 및 사용자 정의 함수 모두에 적용됩니다. 사용자

정의 함수를 호출하는 함수 호출을 완전히 규정하는 것이 좋습니다. 이렇게 하면 함수 결정의 성능이 향상되어 새 함수가 추가되거나 특권이 부여될 때 예기치 않은 함수 결정으로 인한 결과를 방지합니다.

*인수*는 호출시 함수에 전달되는 값입니다. 함수가 SQL에서 호출되면 0개 이상의 인수 목록이 함수에 전달됩니다. 인수의 시맨틱은 인수 목록의 위치에서 결정됩니다. *매개변수*는 함수 입력에 대한 공식적인 정의입니다. 함수를 내부적으로(내장 함수) 또는 사용자(사용자 정의 함수)가 데이터베이스로 정의하면 해당 매개변수(0 이상)가 지정되며 해당 정의의 순서에 따라 위치와 시맨틱이 정의됩니다. 따라서 모든 매개변수는 함수에 대한 특정 위치 입력입니다. 호출시 인수는 인수 목록에 있는 위치에 따라 특정 매개변수에 대응됩니다. 함수 호출 인수가 선택한 함수의 매개변수 데이터 유형과 정확히 일치하지 않는 경우, 인수는 컬럼에 대한 지정과 동일한 규칙을 사용하여 실행 시 매개변수의 데이터 유형으로 변환됩니다. 여기에는 인수와 매개변수 간의 정밀도, 스케일 또는 길이가 다른 경우가 포함됩니다.

스키마 함수에 대한 액세스는 스키마 함수에서 EXECUTE 특권을 통해 제어됩니다. 함수 호출 명령문의 권한 부여 ID에 EXECUTE 특권이 부여되지 않으면, 스키마 함수는 일치하는 내용이 많은 경우에도 함수 결정 알고리즘에서 고려되지 않습니다. 내장 함수(SYSIBM 함수) 및 SYSFUN 스키마의 함수에서는 EXECUTE 특권이 PUBLIC에 내재적으로 부여됩니다.

모듈 함수에 대한 액세스는 모듈 내의 모든 함수에 대한 모듈에서의 EXECUTE 특권을 통해 제어됩니다. 함수 호출 명령문의 권한 부여 ID에는 모듈에 대한 EXECUTE 특권이 없을 수 있습니다. 이러한 경우 스키마 함수와 달리 해당 모듈 내의 모듈 함수는 실행할 수 없지만 함수 결정 알고리즘에서 계속 고려됩니다.

사용자 정의 함수(UDF)가 호출되면, 각 인수 값은 스토리지 지정을 사용하여 해당 함수 매개변수에 지정됩니다. 호출하는 호스트 언어 규칙에 따라 제어권이 외부 함수에 전달됩니다. 사용자 정의 스칼라 함수 또는 사용자 정의 집계 함수의 실행이 완료되면 스토리지 지정을 사용하여 함수 결과가 결과 데이터 유형에 지정됩니다. 지정 규칙에 대한 자세한 내용은 『지정 및 비교』를 참조하십시오.

테이블 함수는 subselect의 FROM절에서만 참조할 수 있습니다. 테이블 함수 참조에 대한 자세한 내용은 『table-reference』를 참조하십시오.

함수 결정

함수를 호출한 후 데이터베이스 관리 프로그램은 실행할 함수를 판별해야 합니다. 이 프로세스는 피호출 함수 결정이며 내장 및 사용자 정의 함수 모두에 적용됩니다.

함수 결정에는 두 단계가 있습니다.

1. 첫 번째 단계에서 데이터베이스 관리 프로그램이 호출된 함수 이름의 자격에 기반한 *후보 함수* 세트, 함수를 호출하는 컨텍스트, 호출된 함수의 규정되지 않은 이름 및 지정된 인수의 수를 판별합니다.
2. 두 번째 단계에서 후보 함수 세트의 함수 매개변수 데이터 유형과 비교하여 데이터베이스 관리 프로그램이 호출된 함수의 인수 데이터 유형에 기반한 후보 함수 세트에서 최적의 함수를 판별합니다.

후보 함수의 세트 판별

후보 함수 세트에 선택된 함수는 다음 검색 스페이스 중 하나 이상의 검색 스페이스에서 선택됩니다.

1. 함수를 호출한 모듈 오브젝트가 포함된 모듈인 *컨텍스트 모듈*
2. 하나 이상의 *스키마* 세트
3. 컨텍스트 모듈 이외의 모듈

고려되는 특정 검색 스페이스는 호출된 함수의 이름 자격에 따라 달라집니다.

- *규정된 함수 결정:* 함수를 함수 이름 및 규정자로 호출할 경우 데이터베이스 관리 프로그램은 *규정자*를 사용하며 경우에 따라 후보 함수 세트를 판별하기 위해 호출된 함수의 컨텍스트를 사용합니다.
  1. 규정자가 있는 함수 이름을 사용하여 모듈 오브젝트 내에서 함수를 호출하는 경우, 데이터베이스 관리 프로그램은 규정자가 컨텍스트 모듈 이름과 일치하는지 고려합니다. 규정자가 단일 ID인 경우, 일치를 판별할 때 모듈의 스키마 이름이 무시됩니다. 규정자가 두 파트의 ID인 경우, 일치를 판별할 때 스키마 규정 모듈 이름과 비교합니다. 규정자가 컨텍스트 모듈 이름과 일치하면, 데이터베이스 관리 프로그램은 다음 기준에 따라 후보 함수의 컨텍스트 모듈을 검색합니다.
     – 함수 인스턴스의 이름은 함수 호출의 이름과 일치해야 합니다.
     – 함수 인스턴스의 입력 매개변수 수는 함수 호출의 인수 수와 일치해야 합니다.

     하나 이상의 후보 함수가 컨텍스트 모듈에서 발견되면 다른 검색 스페이스의 가능한 후보 함수를 고려하지 않고 이 후보 함수 세트가 최적을 위해 처리됩니다. (『최적 판별』 참조) 그렇지 않으면, 다음 검색 스페이스를 계속하십시오.
  2. 규정자가 단일 ID인 경우 데이터베이스 관리 프로그램은 규정자를 스키마 이름으로 간주하고 다음 기준에 따라 후보 함수의 해당 스키마를 검색합니다.
     – 함수 인스턴스의 이름은 함수 호출의 이름과 일치해야 합니다.
     – 함수 인스턴스의 입력 매개변수 수는 함수 호출의 인수 수와 일치해야 합니다.
     – 명령문의 권한 부여 ID는 함수 인스턴스에 EXECUTE 특권을 보유하고 있어야 합니다.

하나 이상의 후보 함수가 스키마에서 발견되면 다른 검색 스페이스의 가능한 후보 함수를 고려하지 않고 이 후보 함수 세트가 최적을 위해 처리됩니다.(『최적 판별』 참조) 그렇지 않으면, 다음 검색 스페이스를 계속하십시오(적용 가능한 경우).

3. 함수가 모듈 외부에서 호출되거나 모듈 오브젝트 내에서 호출될 때 규정자가 컨텍스트 모듈 이름과 일치하지 않는 경우, 데이터베이스 관리 프로그램은 규정자를 모듈 이름으로 간주합니다. 모듈에서의 EXECUTE 특권을 고려하지 않으면 데이터베이스 관리 프로그램은 다음 기준에 따라 일치하는 첫 번째 모듈을 선택합니다.
   - 모듈 이름이 스키마 이름으로 규정된 경우, 해당 스키마 이름 및 모듈 이름의 모듈을 선택하십시오.
   - 모듈 이름이 스키마 이름으로 규정되지 않는 경우, SQL 경로에서 가장 빠른 스키마에 있는 해당 모듈 이름의 모듈을 선택하십시오.
   - SQL 경로를 사용하여 모듈을 찾을 수 없으면, 해당 모듈 이름의 모듈 공용 별명을 선택하십시오.

   일치하는 모듈을 찾을 수 없으면, 후보 함수가 없는 경우입니다. 일치하는 모듈이 발견되면, 데이터베이스 관리 프로그램은 다음 기준에 따라 후보 함수의 선택된 모듈을 검색합니다.
   - 함수 인스턴스의 이름은 함수 호출의 이름과 일치해야 합니다.
   - 함수 인스턴스의 입력 매개변수 수는 함수 호출의 인수 수와 일치해야 합니다.
   - 함수는 발행된 모듈 함수여야 합니다.

   하나 이상의 후보 함수가 선택된 모듈에서 발견되면 이 후보 함수 세트가 최적을 위해 처리됩니다.(『최적 판별』 참조)

데이터베이스 관리 프로그램에서 후보를 찾지 못한 경우, 오류가 리턴됩니다 (SQLSTATE 42884).

- *규정되지 않은 함수 결정:* 함수가 규정자 없이 호출되면 데이터베이스 관리 프로그램은 후보 함수 세트를 판별하기 위해 호출된 함수의 컨텍스트를 고려합니다.

  1. 모듈 오브젝트 내에서 규정되지 않은 함수 이름을 사용하여 함수를 호출하는 경우, 데이터베이스 관리 프로그램은 다음 기준에 따라는 후보 함수의 컨텍스트 모듈을 검색합니다.
     - 함수 인스턴스의 이름은 함수 호출의 이름과 일치해야 합니다.
     - 함수 인스턴스의 입력 매개변수 수는 함수 호출의 인수 수와 일치해야 합니다.

     하나 이상의 후보 함수가 컨텍스트 모듈에서 발견되면 SQL 경로의 스키마에 있는 후보 함수와 함께 이 후보 함수가 포함됩니다.(다음 항목 참조)

2. 모듈 오브젝트 또는 모듈 외부에서 규정되지 않은 함수 이름을 사용하여 함수를 호출하는 경우, 데이터베이스 관리 프로그램은 SQL 경로의 스키마 목록을 검색하여 실행할 함수를 결정합니다. SQL 경로의 각 스키마의 경우(『SQL 경로』 참조), 데이터베이스 관리 프로그램은 다음 기준에 따라 후보 함수의 스키마를 검색합니다.
   - 함수 인스턴스의 이름은 함수 호출의 이름과 일치해야 합니다.
   - 함수 인스턴스의 입력 매개변수 수는 함수 호출의 함수 인수 수와 일치해야 합니다.
   - 명령문의 권한 부여 ID는 함수 인스턴스에 대한 EXECUTE 특권을 보유하고 있어야 합니다.

   하나 이상의 후보 함수가 SQL 경로의 스키마에서 발견되면 컨텍스트 모듈의 후보 함수와 함께 이 후보 함수가 포함됩니다.(이전 항목 참조) 이 후보 함수 세트가 최적을 위해 처리됩니다.(『최적 판별』 참조)

데이터베이스 관리 프로그램에서 후보 함수를 찾을 수 없는 경우, 오류가 리턴됩니다 (SQLSTATE 42884).

최적 판별

후보 함수 세트에는 함수가 한 개 포함되거나 같은 이름의 함수가 여러 개 포함될 수 있습니다. 두 경우 모두 후보 함수 세트에 포함된 각 함수의 매개변수 데이터 유형을 사용하여 함수가 최적의 요구사항에 맞는지 여부를 판별합니다.

데이터베이스 관리 프로그램은 인수와 매개변수 데이터 유형을 비교하여 호출하는 데 가장 적합한 요구사항을 충족하는 함수 또는 함수 세트를 판별합니다. 함수 결과의 데이터 유형 또는 함수 유형(집계, 스칼라 또는 테이블)은 이 결정에 영향을 주지 않습니다.

매개변수의 데이터 유형이 인수와 동일한지 판별할 경우 다음과 같습니다.

- 데이터 유형의 동의어가 일치합니다. 예를 들어, FLOAT 및 DOUBLE은 동일한 것으로 간주됩니다.
- 길이, 정밀도, 스케일 및 코드 페이지와 같은 데이터 유형의 속성이 무시됩니다. 그러므로 CHAR(8) 및 CHAR(35)이 같은 유형으로 간주되고, DECIMAL(11,2) 및 DECIMAL(4,3)도 마찬가지입니다.

함수 호출의 각 입력 인수의 데이터 유형이 함수 인스턴스의 해당 매개변수의 데이터 유형과 일치하거나 이 유형으로 승격 가능한 함수만 고려하여 후보 함수의 서브세트를 확보합니다. 『데이터 유형 승격』에서의 데이터 유형 승격에 대한 순서 목록은 각 데이터 유형에 대해 최상에서 최하위 순서로(승격 고려) 데이터 유형을 나타냅니다. 이 서브세트가 비어 있지 않는 경우, 후보 함수의 이 서브세트에 있는 승격 가능한 프로세스를 사용하여 최적이 판별됩니다. 이 서브세트가 비어 있으면, 후보 함수의 원래 세트에 있는 캐스트 가능한 프로세스를 사용하여 최적이 판별됩니다.

승격 가능한 프로세스

이 프로세스는 함수 호출의 입력 인수가 일치하거나 함수 정의의 해당 매개변수의 데이터 유형으로 승격될 수 있는지 여부를 고려할 때만 최적을 판별합니다. 후보 함수의 서브세트의 경우, 다음 프로세스를 사용하여 왼쪽에서 오른쪽으로 매개변수 목록을 비교합니다.

1. 함수 호출의 인수 데이터 유형을 각 후보 함수 정의의 해당 매개변수의 데이터 유형과 비교합니다.(이전에 언급한 것과 같이 데이터 유형의 동의어는 일치되며 데이터 유형 속성은 무시됩니다.)
2. 이 인수의 경우, 하나의 후보 함수에 다른 후보 함수의 데이터 유형보다 함수 호출에 더 적합한(승격만을 고려) 데이터 유형이 있으면, 해당 함수가 가장 적합합니다. 『데이터 유형 승격』에서의 데이터 유형 승격에 대한 순서 목록은 각 데이터 유형에 대해 최상에서 최하위 순서로(승격 고려) 데이터 유형을 나타냅니다.
3. 둘 이상의 후보 함수에 대한 매개변수의 데이터 유형이 함수 호출에 똑같이 최적인 경우(승격만을 고려), 함수 호출에 똑같이 최적이지 않는 후보 함수는 지우십시오. 함수 호출의 다음 인수에 대해서 이 프로세스를 반복하십시오.

모든 인수를 비교한 후 하나의 후보 함수가 남으면, 그 함수가 최적입니다. 둘 이상의 후보 함수가 남으면, 남은 모든 후보 함수는 동일하게 최적입니다. 이 경우 데이터베이스 관리 프로그램은 함수가 여전히 후보 함수인 경우 컨텍스트 모듈에서 함수를 선택하고 그렇지 않은 경우 SQL 경로에서 가장 먼저인 스키마의 함수를 선택합니다.

함수가 선택되면 성공적인 함수 사용은 리턴된 결과가 허용되는 컨텍스트에서 호출되는지에 따라 다릅니다. 예를 들어, 함수가 테이블이 허용되지 않은 경우에 테이블을 리턴하면, 오류가 리턴됩니다.

캐스트 가능한 프로세스

이 프로세스는 함수 호출의 입력 인수가 일치하거나 함수 정의의 해당 매개변수의 데이터 유형으로 승격될 수 있는지 여부, 및 입력 인수가 해당 매개변수의 데이터 유형에 대한 함수 결정을 위해 내재적으로 캐스트될 수 있는지를 고려할 때 최적을 판별합니다. 후보 함수 세트의 경우, 다음 프로세스를 사용하여 왼쪽에서 오른쪽으로 매개변수 목록을 비교합니다.

1. 함수 호출의 인수 데이터 유형을 각 후보 함수 정의의 해당 매개변수의 데이터 유형과 비교합니다.(이전에 언급한 것과 같이 데이터 유형의 동의어는 일치되며 데이터 유형 속성은 무시됩니다.)
2. 이 인수의 경우, 하나의 후보 함수에 다른 후보 함수의 데이터 유형보다 함수 호출에 더 적합한(승격만을 고려) 데이터 유형이 있으면, 해당 함수가 가장 적합합니다. 『데이터 유형 승격』에서의 데이터 유형 승격에 대한 순서 목록은 각 데이터 유형에 대해 최상에서 최하위 순서로(승격 고려) 데이터

유형을 나타냅니다. 이 경우 하나의 후보 함수가 남아 있는 곳에서, 남아 있는 인수가 평가되어 각 인수가 해당 매개변수로 승격 가능한지 또는 캐스트 가능한지를 확인합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42884).

3. 둘 이상의 후보 함수에 대한 매개변수의 데이터 유형이 함수 호출에 똑같이 최적인 경우(승격만을 고려), 함수 호출에 똑같이 최적이지 않는 후보 함수는 지우십시오. 매개변수의 데이터 유형이 후보 함수의 함수 호출에 적합하지 않으면(승격만을 고려) 지워지는 후보 함수는 없습니다. 함수 호출의 다음 인수에 대해서 이 프로세스를 반복하십시오.

인수 처리 후에 둘 이상의 후보 함수가 남아 있으면 맞지 않는 데이터 유형을 가진 각 매개변수를(승격만을 고려) 왼쪽에서 오른쪽으로 다음 프로세스를 사용하여 비교합니다.

1. 남아 있는 모든 후보 함수에 대해 해당 매개변수의 데이터 유형을 점검하십시오. 『데이터 유형의 승격』에서 지정한대로 모든 데이터 유형이 같은 데이터 유형 우선순위 목록에 해당되지 않는 경우, 오류가 리턴됩니다 (SQLSTATE 428F5).
2. 함수의 내재적 캐스팅에서 지정한대로 인수의 데이터 유형이 해당 매개변수의 데이터 유형에 내재적으로 캐스트될 수 없으면, 오류가 리턴됩니다 (SQLSTATE 42884).
3. 함수 호출의 인수 데이터 유형을 각 후보 함수 정의의 해당 매개변수의 데이터 유형과 비교합니다. 하나의 후보 함수에 다른 후보 함수의 데이터 유형보다 함수 호출에 더 적합한(내재적 캐스트 고려) 데이터 유형이 있으면, 해당 함수가 가장 적합합니다. "함수 결정을 위한 내재적 캐스트"의 데이터 유형 목록은 보다 적합한 데이터 유형(내재적 캐스트 고려)을 나타냅니다. 함수 호출에 동등하게 맞지 않는 후보 함수를 제거합니다(내재적 캐스트 고려). 맞지 않는 데이터 유형이 있는 함수 호출의 다음 인수에 대해서 이 프로세스를 반복하십시오(승격만을 고려).

모든 인수를 비교한 후 하나의 후보 함수가 남으면, 그 함수가 최적입니다. 둘 이상의 후보 함수가 남으면, 남은 모든 후보 함수는 동일하게 최적입니다. 이 경우 데이터베이스 관리 프로그램은 함수가 여전히 후보 함수인 경우 컨텍스트 모듈에서 함수를 선택하고 그렇지 않은 경우 SQL 경로에서 가장 먼저인 스키마의 함수를 선택합니다.

함수가 선택되면 오류가 리턴될 가능성이 있습니다.

- 모듈 함수가 선택되어 함수가 모듈 외부에서 호출되거나 모듈 오브젝트 내에서 호출되어 규정자가 컨텍스트 모듈 이름과 일치하는 않는 경우, 함수를 호출한 명령문의 권한 부여 ID에 선택된 함수(SQLSTATE 42501)가 포함된 모듈에 대한 EXECUTE 특권이 있어야 합니다.

- 함수가 선택되면 성공적인 함수 사용은 리턴된 결과가 허용되는 컨텍스트에서 호출되는지에 따라 다릅니다. 예를 들어, 함수가 테이블이 허용되지 않은 경우에 테이블을 리턴하면, 오류가 리턴됩니다(SQLSTATE 42887).
- 내장 또는 사용자 정의된 캐스트 함수 및 인수를 매개변수의 데이터 유형에 승격이 아니라 내재적으로 캐스트해야 하는 경우 오류가 리턴됩니다 (SQLSTATE 42884).
- 함수 호출에 unnamed row 자료형 인수가 포함되면, 다음 조건 중 하나가 발생하는 경우 오류가 리턴됩니다(SQLSTATE 42884).
  - 인수의 unnamed row 자료형 필드 수가 매개변수의 unnamed row 자료형 필드 수와 일치하지 않는 경우
  - 인수 필드의 데이터 유형을 해당 매개변수 필드의 데이터 유형에 지정할 수 없는 경우

함수 결정을 위한 내재된 캐스팅

함수 결정을 위한 내재된 캐스팅은 사용자 정의 유형, 참조 유형 또는 XML 데이터 유형의 인수에 대해서는 지원하지 않습니다. 내장 또는 사용자 정의된 함수도 지원하지 않습니다. 다음 경우를 지원합니다.

- 『데이터 유형의 승격』에서 지정한대로, 하나의 데이터 유형 값은 같은 데이터 유형 순서 목록에 있는 다른 데이터 유형으로 캐스트할 수 있습니다.
- 숫자 또는 날짜 및 시간 데이터 유형은 LOB를 제외하고 문자나 그래픽 문자열 데이터 유형으로 캐스트할 수 있습니다.
- LOB를 제외한 문자나 그래픽 문자열 유형은 숫자나 날짜 및 시간 데이터 유형으로 캐스트할 수 있습니다.
- 문자 FOR BIT DATA는 BLOB로 캐스트할 수 있으며 BLOB는 문자 FOR BIT DATA로 캐스트할 수 있습니다.
- TIMESTAMP 데이터 유형은 TIME 데이터 유형으로 캐스트할 수 있습니다.
- 유형이 지정되지 않은 인수를 문자열, 숫자 또는 날짜 및 시간 데이터 유형으로 캐스트할 수 있습니다.

승격을 위한 데이터 유형 순서 목록과 마찬가지로, 내재된 캐스팅에 대해서는 관련 데이터 유형 그룹에 있는 데이터 유형에 대해 순서가 있습니다. 내재된 캐스팅을 고려하는 함수 결정을 수행할 때 이 순서가 사용됩니다. 216 페이지의 표 19는 함수 결정의 내재된 캐스팅에 대한 데이터 유형 순서를 나타냅니다. 이 데이터 유형은 최상에서 최하의 순서로 열거됩니다(승격에 대한 데이터 유형 순서 목록의 순서와 다르다는 것에 유의). 함수 결정이 내장 함수를 선택하고 내재된 캐스팅이 일부 인수에 대해 필요할 때, 내장 함수가 매개변수에 대한 그래픽 입력 및 문자 입력 모두를 지원하는 경우 인수는 내재적으로 문자로 캐스트됩니다.

표 19. 함수 결정을 위한 내재된 캐스팅의 데이터 유형 순서

데이터 유형 그룹	함수 결정을 위한 내재된 캐스팅의 데이터 유형 순서(최상에서 최하 순서)
숫자 데이터 유형	DECFLOAT, double, real, decimal, BIGINT, INTEGER, SMALLINT
문자 및 그래픽 문자열 데이터 유형	VARCHAR 또는 VARGRAPHIC, CHAR 또는 GRAPHIC, CLOB 또는 DBCLOB
날짜 및 시간 데이터 유형	TIMESTAMP, DATE

참고:

1. 위의 소문자 유형은 다음과 같이 정의됩니다.
   - decimal = DECIMAL (*p*,*s*) 또는 NUMERIC(*p*,*s*)
   - real = REAL 또는 FLOAT(*n*) 여기서 *n*은 24보다 크지 않습니다.
   - double = DOUBLE, DOUBLE-PRECISION, FLOAT 또는 FLOAT(*n*). 여기서, *n*은 24보다 큽니다.

   나열된 데이터 유형의 짧거나 긴 양식의 동의어는 나열된 양식과 동일한 것으로 간주됩니다.
2. 유니코드 데이터베이스의 경우, 다음은 동일한 데이터 유형으로 간주됩니다.
   - CHAR 또는 GRAPHIC
   - VARCHAR 및 VARGRAPHIC
   - CLOB 및 DBCLOB

표 20. 내재된 캐스팅이 필요한 경우 내장 스칼라 함수를 호출할 때 인수의 파생 길이.

	목표 유형 및 길이								
소스 데이터 유형	**Char**	**Graphic**	**Varchar**	**Vargraphic**	**Clob**	**DBclob**	**Blob**	시간소인	**Decfloat**
UNTYPED	127	127	254	254	32767	32767	32767	12	34
SMALLINT	6	6	6	6	-	-	-	-	-
INTEGER	11	11	11	11	-	-	-	-	-
BIGINT	20	20	20	20	-	-	-	-	-
DECIMAL(p,s)	2+p	2+p	2+p	2+p	-	-	-	-	-
REAL	24	24	24	24	-	-	-	-	-
DOUBLE	24	24	24	24	-	-	-	-	-
DECFLOAT	42	42	42	42	-	-	-	-	-
CHAR(n)	-	-	-	-	-	-	min(n,254)	12	34
VARCHAR(n)	min(n,254)	min(n,127)	-	-	-	-	min(n,32672)	12	34
CLOB(n)	min(n,254)	min(n,127)	min(n,32672)	min(n,16336)	-	-	-	-	-
GRAPHIC(n)	-	-	-	-	-	-	-	12	34
VARGRAPHIC(n)	min(n,254)	min(n,127)	-	-	-	-	-	12	34
DBCLOB(n)	min(n,254)	min(n,127)	min(n,32672)	min(n,16336)	-	-	-	-	-
BLOB(n)	min(n,254)	-	min(n,32672)	-	-	-	-	-	-
TIME	8	8	8	8	-	-	-	-	-
DATE	10	10	10	10	-	-	-	-	-
TIMESTAMP (p)	if p=0 then 19 else p+20	if p=0 then 19 else p+20	if p=0 then 19 else p+20	if p=0 then 19 else p+20	-	-	-	-	-

내장 함수에 대한 SQL 경로 고려사항

내장 함수는 SYSIBM이라는 특수 스키마에 있습니다. 추가 함수는 SYSIBMADM 스키마 내의 모듈과 SYSFUN, SYSPROC 및 SYSIBMADM 스키마에서 사용할 수 있으나, 사용자 정의 함수로 개발되었고 특별한 처리 고려사항이 없으므로 내장 함수로 간주되지 않습니다. 사용자는 SYSIBM, SYSFUN, SYSPROC 또는 SYSIBMADM 스키마(또는 'SYS' 문자로 시작하는 이름의 다른 스키마(SYSTOOLS 제외))에서 추가 함수를 정의할 수 없습니다.

이미 언급한 바와 같이, 내장 함수는 이들이 사용자 정의 함수에서 수행한 것과 같이 함수 결정 프로세스에도 참여합니다. 함수 결정 측면에서 내장 및 사용자 정의 함수 사이의 한 가지 차이점은 내장 함수는 함수 결정 시 항상 고려되어야 한다는 점입니다. 따라서 경로에서 SYSIBM을 생략하면(함수와 데이터 유형 결정의 경우) SYSIBM이 경로에서 첫 번째 스키마로 간주됩니다.

예를 들어, 사용자의 SQL 경로가 다음과 같이 정의되고,

```
"SHAREFUN","SYSIBM","SYSFUN"
```

SYSIBM.LENGTH와 인수의 수 및 유형이 같은 LENGTH 함수가 스키마 SHAREFUN에 정의된 경우, 이 사용자의 SQL문에 있는 LENGTH에 대해 규정되지 않은 참조로 인해 SHAREFUN.LENGTH가 선택됩니다. 그러나 사용자의 SQL 경로가 다음과 같이 정의되고,

```
"SHAREFUN","SYSFUN"
```

동일한 SHAREFUN.LENGTH 함수가 존재하는 경우, 이 사용자의 SQL문에 있는 LENGTH에 대해 규정되지 않은 참조로 인해 SYSIBM.LENGTH가 선택됩니다. 왜냐하면 SYSIBM은 내재적으로 경로에서 가장 먼저 나타나기 때문입니다.

이 영역에서 잠재적인 문제점을 최소화하려면 다음을 수행하십시오.

- 사용자 정의 함수에 내장 함수 이름을 절대로 사용하지 마십시오.
- 몇 가지 이유로 인해 내장 함수와 이름이 동일한 사용자 정의 함수를 작성해야 할 경우, 이에 대한 모든 참조를 규정해야 합니다.

주: 내장 함수의 일부 호출은 SYSIBM을 명시적 규정자로 지원하지 않으며 SQL 경로를 고려하지 않은 경우 내장 함수를 직접 결정하지 않습니다. 특수한 경우에 대해서는 내장 함수 설명에서 다룹니다.

함수 결정 예

다음은 함수 결정에 대한 예입니다. (필수 키워드를 모두 표시한 것이 아니라는 점에 유의하십시오.)

- 다음은 함수 결정에서 SQL 경로 고려사항을 설명하는 예입니다. 이 예의 경우, 세 개의 서로 다른 스키마에서 8개의 ACT 함수가 다음과 같이 등록되어 있습니다.

```
CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE) SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT (INT, DOUBLE, DOUBLE) SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_7 ...
CREATE FUNCTION NERO.ACT (INT, INT, DEC(7,2)) SPECIFIC ACT_8 ...
```

함수 참조는 다음과 같습니다. (여기서, I1과 I2는 INTEGER 컬럼이고, D는 10진수 컬럼입니다.)

```
SELECT ... ACT(I1, I2, D) ...
```

이 참조를 구성하는 응용프로그램에 다음과 같이 설정될 수 있는 SQL 경로가 있다고 가정해 보십시오.

```
"JULIUS","AUGUSTUS","CAESAR"
```

위 알고리즘에 따르면...

- 특정 이름이 ACT_8인 함수는 스키마 NERO가 SQL 경로에 포함되지 않으므로 후보에서 제외됩니다.
- 특정 이름이 ACT_3인 함수는 매개변수 수가 틀리므로 후보에서 제외됩니다. ACT_1 및 ACT_6은 모두 첫 번째 인수가 첫 번째 매개변수의 데이터 유형으로 승격될 수 없으므로 제외됩니다.
- 둘 이상의 후보가 남아 있으므로, 인수는 순서대로 고려되어야 합니다.
- 첫 번째 인수의 경우, 나머지 함수 ACT_2, ACT_4, ACT_5 및 ACT_7은 인수 유형과 정확히 일치합니다. 어떤 함수도 고려할 필요가 없으므로, 다음 인수를 검사해야 합니다.
- 두 번째 인수의 경우, ACT_2, ACT_5 및 ACT_7은 정확히 일치하지만 ACT_4는 일치하지 않으므로 고려할 필요가 없습니다. ACT_2, ACT_5 및 ACT_7의 차이점을 판별하기 위해 다음 인수를 검사합니다.
- 세 번째와 마지막 인수의 경우, ACT_2, ACT_5 및 ACT_7은 인수 유형이 정확히 일치하지 않습니다. DOUBLE이 DECFLOAT보다 DECIMAL에 가까우므로, ACT_2와 ACT_5가 동등하게 양호하지만 ACT_7은 다른 두 개와 같이 양호하지 않습니다. ACT_7이 제거됩니다..
- 나머지 두 함수 ACT_2와 ACT_5에는 동일한 매개변수 시그니처가 있습니다. 마지막 결정 기준으로, SQL 경로에서 가장 먼저 나타나는 함수 스키마를 찾습니다. 이 기준에 따라 ACT_5 함수가 선택됩니다.

• 둘 이상의 후보 함수가 호출에 동등하게 최적이지만 인수 중 하나에 대한 해당 매개변수가 같은 유형 순서 목록에 속하지 않기 때문에 함수 결정에서 오류가 발생하는 (SQLSTATE 428F5) 상황의 예입니다.

  이 예의 경우, 다음과 같이 정의되는 단일 스키마에 세 가지 함수만 있습니다.

```
CREATE FUNCTION CAESAR.ACT (INT, VARCHAR(5), VARCHAR(5))SPECIFIC ACT_1 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DATE)     SPECIFIC ACT_2 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE)   SPECIFIC ACT_3 ...
```

함수 참조는 다음과 같습니다(여기서 I1과 I2는 INTEGER 컬럼이고 VC는 VARCHAR 컬럼임).

```
SELECT ... ACT(I1, I2, VC) ...
```

이 참조를 구성하는 응용프로그램에 다음과 같이 설정될 수 있는 SQL 경로가 있다고 가정해 보십시오.

```
"CAESAR"
```

위 알고리즘에 따르면...

– 함수 호출의 각 입력 인수의 데이터 유형이 함수 인스턴스의 해당 매개변수의 데이터 유형과 일치하거나 승격 가능한지 판별하기 위해 각각의 후보 함수를 평가합니다.
  - 첫 번째 인수의 경우, 모든 후보 함수는 매개변수 유형과 정확히 일치합니다.
  - 두 번째 인수의 경우, INTEGER가 VARCHAR로 승격될 수 없으므로 ACT_1이 제거됩니다.
  - 세 번째 인수의 VARCHAR이 DATE 또는 DOUBLE로 승격될 수 없으므로 ACT_2와 ACT_3이 제거되어 남아 있는 후보 함수가 없습니다.

– 위의 후보 함수의 서브세트가 비어 있으므로, 캐스팅 가능한 프로세스를 사용하여 후보 함수가 고려됩니다.
  - 첫 번째 인수의 경우, 모든 후보 함수는 매개변수 유형과 정확히 일치합니다.
  - 두 번째 인수의 경우, INTEGER가 VARCHAR로 승격될 수 없으므로 ACT_1이 제거됩니다. ACT_2와 ACT_3이 더 나은 후보입니다.
  - 세 번째 인수의 경우, ACT_2와 ACT_3의 해당 매개변수의 데이터 유형은 같은 데이터 유형 순서 목록에 해당되지 않으므로, 오류가 리턴됩니다(SQLSTATE 428F5).

• 이 예는 캐스팅 가능한 프로세스를 사용하여 함수 결정이 이루어지는 상황을 설명합니다. 이 예의 경우, 다음과 같이 정의되는 단일 스키마에 세 가지 함수만 있습니다.

```
CREATE FUNCTION CAESAR.ACT (INT, VARCHAR(5), VARCHAR(5))SPECIFIC ACT_1 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DECFLOAT)     SPECIFIC ACT_2 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE)   SPECIFIC ACT_3 ...
```

함수 참조는 다음과 같습니다(여기서 I1과 I2는 INTEGER 컬럼이고 VC는 VARCHAR 컬럼임).

```
SELECT ... ACT(I1, I2, VC) ...
```

이 참조를 구성하는 응용프로그램에 다음과 같이 설정될 수 있는 SQL 경로가 있다고 가정해 보십시오.

```
"CAESAR"
```

위 알고리즘에 따르면...

– 함수 호출의 각 입력 인수의 데이터 유형이 함수 인스턴스의 해당 매개변수의 데이터 유형과 일치하거나 승격 가능한지 판별하기 위해 각각의 후보 함수를 평가합니다.

- 첫 번째 인수의 경우, 모든 후보 함수는 매개변수 유형과 정확히 일치합니다.
- 두 번째 인수의 경우, INTEGER가 VARCHAR로 승격될 수 없으므로 ACT_1이 제거됩니다.
- 세 번째 인수의 경우, VARCHAR이 DECFLOAT 또는 DOUBLE로 승격될 수 없으므로 ACT_2와 ACT_3이 제거되어 남아 있는 후보 함수가 없습니다.

– 위의 후보 함수의 서브세트가 비어 있으므로, 캐스팅 가능한 프로세스를 사용하여 후보 함수가 고려됩니다.

- 첫 번째 인수의 경우, 모든 후보 함수는 매개변수 유형과 정확히 일치합니다.
- 두 번째 인수의 경우, INTEGER가 VARCHAR로 승격될 수 없으므로 ACT_1이 제거됩니다. ACT_2와 ACT_3이 더 나은 후보입니다.
- 세 번째 인수의 경우, DECFLOAT 및 DOUBLE은 동일한 데이터 유형 순서 목록에 있으며 VARCHAR은 두 DECFLOAT 및 DOUBLE로 내재적으로 캐스트될 수 있습니다. DECFLOAT가 내재된 캐스팅의 용도로 더 최적이기 때문에 ACT_2가 최적입니다.

• 이 예는 나중 인수의 승격이 내재된 캐스팅에 우선하는 캐스팅 가능한 프로세스를 사용하는 함수 결정 과정을 설명합니다. 이 예의 경우, 다음과 같이 정의되는 단일 스키마에 세 가지 함수만 있습니다.

```
CREATE FUNCTION CAESAR.ACT (INT, INT, VARCHAR(5))SPECIFIC ACT_1 ...
   CREATE FUNCTION CAESAR.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_2 ...
   CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE)  SPECIFIC ACT_3 ...
```

함수 참조는 다음과 같습니다(여기서 I1은 INTEGER 컬럼이고 VC1은 VARCHAR 컬럼이며 C1은 CHAR 컬럼임).

```
SELECT ... ACT(I1, VC1, C1) ...
```

이 참조를 구성하는 응용프로그램에 다음과 같이 설정될 수 있는 SQL 경로가 있다고 가정해 보십시오.

```
"CAESAR"
```

위 알고리즘에 따르면...

– 함수 호출의 각 입력 인수의 데이터 유형이 함수 인스턴스의 해당 매개변수의 데이터 유형과 일치하거나 승격 가능한지 판별하기 위해 각각의 후보 함수를 평가합니다.

  - 첫 번째 인수의 경우, 모든 후보 함수는 매개변수 유형과 정확히 일치합니다.
  - 세 번째 인수의 경우, VARCHAR이 INTEGER로 승격될 수 없으므로 모든 후보 함수가 제거되어 남아 있는 후보 함수가 없습니다.

– 위의 후보 함수의 서브세트가 비어 있으므로, 캐스팅 가능한 프로세스를 사용하여 후보 함수가 고려됩니다.
  - 첫 번째 인수의 경우, 모든 후보 함수는 매개변수 유형과 정확히 일치합니다.
  - 세 번째 인수의 경우, 후보 함수에 해당 인수에서 승격될 수 있는 매개변수가 없으므로 제거된 후보 함수가 없습니다.
  - 세 번째 인수는 ACT_1의 매개변수로 승격될 수 있지만, ACT_2 또는 ACT_3의 매개변수로 승격될 수 없으므로, ACT_1이 최적입니다.

메소드

구조화된 유형의 데이터베이스 메소드는 입력 데이터 값 세트와 결과 값 세트 사이의 관계입니다. 여기서, 첫 번째 입력 값(또는 *주제 인수*)은 같은 유형이거나 메소드의 주제 유형(*주제 매개변수*라고도 함)에 대한 부속 유형입니다. 예를 들어, 유형이 ADDRESS인 CITY라는 메소드는 유형이 VARCHAR인 입력 데이터 값에 전달될 수 있고 결과는 ADDRESS(또는 ADDRESS의 부속 유형)가 됩니다.

메소드는 사용자 정의 구조화된 유형 정의의 일부로 내재적으로 또는 명시적으로 정의됩니다.

내재적으로 정의된 메소드는 구조화된 유형마다 작성됩니다. *관찰자 메소드*는 구조화된 유형의 각 속성에 정의됩니다. 관찰자 메소드는 응용프로그램이 해당 유형의 인스턴스에 대한 속성 값을 가져오도록 허용합니다. *Mutator 메소드*도 각 속성에 정의되어, 응용프로그램이 유형 인스턴스의 속성 값을 변경하여 유형 인스턴스를 변형할 수 있게 합니다. 위에 설명된 CITY 메소드는 ADDRESS 유형에 대한 mutator 메소드의 예입니다.

명시적으로 정의된 메소드나 *사용자 정의 메소드*는 CREATE TYPE(또는 ALTER TYPE ADD METHOD) 및 CREATE METHOD문의 조합을 사용하여 SYSCAT.ROUTINES에서 데이터베이스에 등록되는 메소드입니다. 구조화된 유형에 대해 정의된 모든 메소드는 해당 유형과 같은 스키마에서 정의됩니다.

구조화된 유형의 사용자 정의 메소드는 데이터베이스 엔진의 구조화된 유형 인스턴스에 적용될 수 있는 메소드 정의(사용자나 써드 파티 벤더가 제공한)를 추가하여 데이터베이스 시스템의 기능을 확장합니다. 데이터베이스 메소드를 정의하면 데이터베이스는 응용프로그램이 사용하는 엔진에서 동일한 메소드를 사용할 수 있으며 응용프로그램과 데이터베이스 사이에 더 많은 상승 효과를 제공합니다.

외부 및 SQL 사용자 정의 메소드

사용자 정의 메소드는 외부 메소드이거나 SQL 표현식에 기반한 메소드일 수 있습니다. 외부 메소드는 오브젝트 코드 라이브러리를 참조하는 데이터베이스와, 메소드가 호출될 때 실행될 해당 라이브러리 내의 함수에 정의됩니다. SQL 표현식에 기반한 메소드는 메소드가 호출되면 SQL 표현식의 결과를 리턴합니다. 이러한 메소드는 완전하게 SQL로 작성되므로 오브젝트 코드 라이브러리가 필요하지 않습니다.

사용자 정의 메소드는 호출될 때마다 단일 값의 응답을 리턴할 수 있습니다. 이 값은 구조화된 유형일 수 있습니다. 메소드는 주제 인수의 동적 유형이 메소드의 리턴된 유형으로 리턴될 수 있도록 *유형 보존*(SELF AS RESULT 사용)으로 정의될 수 있습니다. 내재적으로 정의된 모든 mutator 메소드의 유형은 보존됩니다.

메소드 시그니처

메소드는 주제 유형, 메소드 이름, 매개변수 수 및 해당 매개변수의 데이터 유형으로 식별됩니다. 이를 *메소드 시그니처*라고 하며, 이는 데이터베이스 내에서 고유해야 합니다.

다음과 같은 경우 구조화된 유형에 대해 이름이 동일한 둘 이상의 메소드가 있을 수 있습니다.

- 매개변수 수나 매개변수의 데이터 유형이 다른 경우
- 메소드가 동일한 메소드 계층 구조의 일부인 경우(즉, 메소드가 대체 관계에 있거나 동일한 원래 메소드를 대체하는 경우)
- 동일한 함수 시그니처(주제 유형이나 첫 번째 매개변수로 부속 유형 또는 슈퍼 유형 사용)가 존재하지 않는 경우

다중 메소드 인스턴스가 있는 메소드 이름을 *오버로드된 메소드*라고 합니다. 메소드 이름은 한 유형 내에서 오버로드될 수 있으며, 이런 경우 유형에 대해 해당 이름의 메소드가 둘 이상 있게 됩니다(매개변수 유형은 모두 다름). 메소드 이름은 주제 유형 계층 구조에서 오버로드될 수 있으며, 이런 경우 유형 계층 구조에는 해당 이름의 메소드가 둘 이상 있게 됩니다. 이러한 메소드에서는 매개변수 유형이 서로 달라야 합니다.

메소드는 구조화된 유형 인스턴스(주제 인수)와 이중점 연산자가 앞에 오는 메소드 이름을 참조하여(허용 가능한 컨텍스트에서) 호출될 수 있습니다. 괄호로 묶인 인수 목록이 다음에 와야 합니다. 실제로 호출되는 메소드는 다음 절에 설명된 메소드 결정 프로세스를 사용하여 주제 유형의 정적 유형에 따라 달라집니다. WITH FUNCTION ACCESS로 정의된 메소드는 함수 호출을 사용하여 호출될 수도 있습니다. 이 경우 함수 결정에 대한 일반 규칙이 적용됩니다.

함수 결정으로 메소드가 WITH FUNCTION ACCESS로 정의되는 경우, 메소드 호출의 모든 후속 단계가 처리됩니다.

메소드에 대한 액세스는 EXECUTE 특권을 통해 제어됩니다. GRANT 및 REVOKE 문은 특정 메소드나 메소드 세트를 실행할 수 있거나 실행할 수 없는 사용자를 지정하는 데 사용됩니다. EXECUTE 특권(또는 DATAACCESS 권한)은 메소드를 호출하는 데 필요합니다. 메소드 정의자는 자동으로 EXECUTE 특권을 부여받습니다. 모든 하위 오브젝트에 대한 WITH GRANT 옵션을 갖는 외부 메소드 또는 SQL 메소드의 정의자는 메소드에 대한 EXECUTE 특권이 있는 WITH GRANT 옵션도 부여받습니다. 그러면 정의자(또는 ACCESSCTRL 또는 SECADM 권한을 포함한 권한 부여 ID)는 SQL문에서 메소드를 호출하거나, DDL문(예: CREATE VIEW, CREATE TRIGGER 또는 제한조건 정의 시)에서 메소드를 참조하려는 사용자에게 부여해야 합니다. EXECUTE 특권이 사용자에게 부여되지 않으면, 메소드는 보다 더 일치하는 경우에도 메소드 결정 알고리즘에서 고려되지 않습니다.

메소드 결정

메소드가 호출되면 데이터베이스 관리 프로그램에서는 동일한 이름의 가능한 메소드 중 어느 것이 『최적』인지 결정해야 합니다. 함수(내장 또는 사용자 정의)는 메소드 결정 시 고려되지 않습니다.

*인수*는 호출시 메소드에 전달되는 값입니다. 메소드가 SQL에서 호출되면 주제 인수(구조화된 유형)와 0개 이상의 인수 목록이 메소드에 전달됩니다. 인수의 시맨틱은 인수 목록의 위치에서 결정됩니다. *매개변수*는 메소드 입력에 대한 공식적인 정의입니다. 메소드를 내재적으로(유형에 대해 시스템 생성) 또는 사용자(사용자 정의 메소드)가 데이터베이스에 정의하면 해당 매개변수가 지정되며(첫 번째 매개변수로서 주제 매개변수로) 해당 정의 순서에 따라 위치와 시맨틱이 정의됩니다. 따라서 모든 매개변수는 메소드에 대한 특정 위치 입력입니다. 호출시 인수는 인수 목록에 있는 위치에 따라 특정 매개변수에 대응됩니다.

데이터베이스 관리 프로그램에서는 호출시 제공되는 메소드 이름, 메소드에 대한 EXECUTE 특권, 인수의 수 및 데이터 유형, 주제 인수의 정적 유형(및 해당 슈퍼 유형)에 대한 동일한 이름의 모든 메소드, 해당 매개변수의 데이터 유형을 메소드 선택 여부를 결정하기 위한 기준으로 사용합니다. 다음은 결정 프로세스의 가능한 결과입니다.

- 특정 메소드가 최적으로 판단됩니다. 예를 들어, 시그니처가 다음과 같이 정의된 SITE 유형에 대한 RISK 메소드와,

  ```
  PROXIMITY(INTEGER) FOR SITE
  PROXIMITY(DOUBLE) FOR SITE
  ```

  다음 메소드 호출(여기서, ST는 SITE 컬럼이고 DB는 DOUBLE 컬럼임)이 제공될 경우

  ```
  SELECT ST..PROXIMITY(DB) ...
  ```

  두 번째 PROXIMITY가 선택됩니다.

  다음 메소드 호출(여기서, SI는 SMALLINT 컬럼임)에서

  ```
  SELECT ST..PROXIMITY(SI) ...
  ```

  SMALLINT가 INTEGER로 승격될 수 있고 우선순위 목록 아래에 있는 DOUBLE보다 더 적합하므로 첫 번째 PROXIMITY를 선택합니다.

  구조화된 유형의 인수를 고려할 때 우선순위 목록에 인수의 정적 유형에 대한 슈퍼 유형이 포함됩니다. 최적의 함수는 구조화된 유형 계층 구조에서 함수 인수의 정적 유형에 가장 근접한 슈퍼 유형 매개변수로 정의된 함수입니다.
- 적합한 것으로 판단되는 메소드가 없습니다. 예를 들어, 이전 예에서와 같은 두 함수와 다음의 함수 참조가 있을 경우(여기서, C는 CHAR(5) 컬럼임),

  ```
  SELECT ST..PROXIMITY(C) ...
  ```

인수는 어느 PROXIMITY 함수의 매개변수와도 일치하지 않습니다.

- 특정 메소드는 유형 계층 구조 내의 메소드와, 호출시 전달된 인수의 수 및 데이터 유형을 기준으로 선택됩니다. 예를 들어, 시그니처가 다음과 같이 정의된 SITE 및 DRILLSITE(SITE의 부속 유형) 유형에 대한 RISK 메소드와,

  ```
  RISK(INTEGER) FOR DRILLSITE
  RISK(DOUBLE) FOR SITE
  ```

  다음 메소드 호출(여기서, DRST는 DRILLSITE 컬럼이고 DB는 DOUBLE 컬럼)이 제공될 경우

  ```
  SELECT DRST..RISK(DB) ...
  ```

  DRILLSITE가 SITE로 승격될 수 있으므로, 두 번째 RISK가 선택됩니다.

  다음 메소드 참조(여기서, SI는 SMALLINT 컬럼임)는

  ```
  SELECT DRST..RISK(SI) ...
  ```

  SMALLINT가 DOUBLE보다 우선순위 목록에 더 근접한 INTEGER로 승격될 수 있고 DRILLSITE가 슈퍼 유형인 SITE보다 더 적합하므로, 첫 번째 RISK를 선택합니다.

  동일한 유형 계층 구조 내의 메소드는 주제 매개변수 이외의 매개변수를 고려하여 동일한 시그니처를 수반할 수 없습니다.

최적 판별

고려 중인 메소드 매개변수의 정의된 데이터 유형과 인수의 데이터 유형 비교는 유사한 메소드 그룹에서 어느 함수가 『최적』인지 결정하는 기준입니다. 고려 중인 메소드 결과의 데이터 유형은 이 결정에 영향을 주지 않는다는 점에 유의하십시오.

메소드 결정의 경우, 입력 인수의 데이터 유형이 해당 매개변수의 데이터 유형으로 프롬프트될 수 있는지 여부는 최적을 판별할 때 고려됩니다. 함수 결정과는 다르게, 입력 인수가 해당 매개변수의 데이터 유형으로 내재적으로 캐스트될 수 있는지 여부는 최적을 판별할 때 고려되지 않습니다. 메소드가 모듈에 정의될 수 없으므로 모듈은 메소드 결정 시 고려되지 않습니다.

메소드 결정은 다음 단계를 사용하여 수행됩니다.

1. 먼저, 카탈로그(SYSCAT.FUNCTIONS)에서 다음 사항을 모두 만족하는 모든 메소드를 찾습니다.
   - 메소드 이름이 호출 이름과 일치하고, 주제 매개변수가 같은 유형이거나 주제 인수의 정적 유형에 대한 슈퍼 유형입니다.
   - 호출자에게 메소드에 대한 EXECUTE 특권이 있습니다.
   - 정의된 매개변수 수가 호출과 일치합니다.

- 각 호출 인수는 데이터 유형에서 메소드의 해당 정의 매개변수와 일치하거나 그 매개변수로 『승격』될 수 있습니다.

2. 다음으로, 왼쪽에서 오른쪽으로 메소드 호출의 각 인수를 고려합니다. 가장 왼쪽에 있는 인수(첫 번째 인수)는 내재된 SELF 매개변수입니다. 예를 들어, ADDRESS_T 유형에 대해 정의된 메소드에는 ADDRESS_T 유형의 내재된 첫 번째 인수가 있습니다. 각각의 인수에 대해 최적 일치가 아닌 모든 함수를 제거합니다. 주어진 인수에 대한 최적 일치는 해당 데이터 유형의 매개변수를 갖는 함수가 존재하는 인수 데이터 유형에 해당하는 우선순위 목록에 처음 나타나는 데이터 유형입니다. 길이, 정밀도, 스케일 및 FOR BIT DATA 속성은 이 비교에서 고려되지 않습니다. 예를 들어, DECIMAL(9,1) 인수는 DECIMAL(6,5) 매개변수와 정확히 일치하는 것으로 간주되고, DECFLOAT(34) 인수는 VARCHAR(19) 매개변수와 정확히 일치하는 것으로 간주되며, VARCHAR(19) 인수는 VARCHAR(6) 매개변수와 정확히 일치하는 것으로 간주됩니다.

   사용자 정의 구조화된 유형 인수에 대해 최적 일치는 그 자체입니다. 다음 최적 일치는 해당되는 중간 슈퍼 유형이며, 인수의 각 슈퍼 유형에 대해 마찬가지로 진행됩니다. 구조화된 유형 인수의 정적 유형(선언된 유형)만 고려되고, 동적 유형(대부분의 특정 유형)은 고려되지 않습니다.
3. 2단계 후에는 많아야 하나의 후보 메소드가 남습니다. 이것이 선택되는 메소드입니다.
4. 2단계 이후에 남아 있는 후보 메소드가 없는 경우, 오류가 리턴됩니다(SQLSTATE 42884).

메소드 결정 예

다음은 성공적인 메소드 결정의 예입니다.

다음 시그니처로 등록된 HEADOFSTATE의 부속 유형으로, EMPEROR의 부속 유형으로 GOVERNOR 계층 구조에 정의된 세 개의 구조화된 유형에 대해 7개의 FOO 메소드가 있습니다.

```
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 DECIMAL 컬럼, E는 EMPEROR 컬럼입니다.

```
SELECT E..FOO(I1, I2, D) ...
```

위 알고리즘에 따르면...

- FOO_7은 GOVERNOR 유형이 EMPEROR의 부속 유형(슈퍼 유형이 아님)이므로 후보에서 제외됩니다.
- FOO_3은 매개변수 수가 틀리므로 후보에서 제외됩니다.
- FOO_1과 FOO_6은 두 경우 모두 첫 번째 인수(주제 인수가 아니라)가 첫 번째 매개변수의 데이터 유형으로 승격될 수 없으므로 제외됩니다. 둘 이상의 후보가 남아 있으므로, 인수는 순서대로 고려되어야 합니다.
- 주제 인수에 대해 FOO_2는 슈퍼 유형이고, FOO_4 및 FOO_5는 주제 인수와 일치합니다.
- 첫 번째 인수에 대해 나머지 메소드 FOO_4 및 FOO_5는 인수 유형과 정확히 일치합니다. 어떤 메소드도 고려할 필요가 없으므로, 다음 인수를 검사해야 합니다.
- 두 번째 인수에 대해 FOO_5는 정확히 일치하지만 FOO_4는 일치하지 않으므로, 고려할 필요가 없습니다. 이로써 FOO_5 메소드가 선택됩니다.

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 컨텍스트와 호환되지 않을 경우, 오류가 발생합니다. 예를 들어, STEP이라는 다음 메소드들이 정의되어 있을 경우 각각은 서로 다른 데이터 유형을 결과로 리턴합니다.

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

메소드 참조가 다음과 같은 경우(여기서, S는 SMALLINT 컬럼이고 TA는 TYPEA 컬럼임),

```
SELECT 3 + TA..STEP(S) ...
```

인수 유형과 정확히 일치하는 것이 있으므로, 첫 번째 STEP이 선택됩니다. 결과 유형이 추가 연산자의 인수에 필요한 숫자 유형이 아닌 CHAR(5)이므로, 명령문에서 오류가 발생합니다.

선택한 메소드에서 시작되는 『메소드의 동적 디스패치』에 설명된 알고리즘이 컴파일 시 디스패치 가능 메소드 세트를 빌드하는 데 사용됩니다. 정확히 어떤 메소드가 호출될지는 『메소드의 동적 디스패치』에 설명되어 있습니다.

선택된 메소드가 유형 보존 메소드일 경우 다음 사항에 유의하십시오.

- 함수 결정 이후의 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 슈퍼 유형이 될 수 있습니다.

메소드 호출의 인수가 선택된 메소드의 매개변수 데이터 유형과 정확히 일치하지 않는 경우, 인수는 컬럼에 대한 지정과 같은 규칙을 사용하여 실행시 매개변수의 데이터 유형으로 변환됩니다. 인수와 매개변수 간에 정밀도, 스케일 또는 길이가 다른 경우가 여기에 해당하며, 인수의 동적 유형이 매개변수의 정적 유형에 대한 부속 유형인 경우는 제외됩니다.

메소드의 동적 디스패치

메소드는 기능을 제공하며 유형의 데이터를 캡슐화합니다. 메소드는 유형에 대해 정의되므로 해당 유형과 항상 연관될 수 있습니다. 메소드의 매개변수 중 하나는 내재된 SELF 매개변수입니다. SELF 매개변수의 유형은 메소드가 선언된 유형입니다. 메소드가 DML문에서 호출될 때 SELF 인수로 전달된 인수를 *주제*라고 합니다.

메소드가 메소드 결정(224 페이지의 『메소드 결정』 참조)을 사용하여 선택되거나 메소드가 DDL문에 지정되면 이 메소드를 『가장 구체적인 적용 가능한 권한 부여 메소드』라고 합니다. 주제가 구조화된 유형인 경우, 해당 메소드에는 하나 이상의 대체 메소드가 있을 수 있습니다. DB2는 런타임시 주제의 동적 유형(가장 구체적인 유형)에 기반하여 호출할 메소드를 결정해야 합니다. 이러한 결정을 『가장 구체적인 디스패치 가능 메소드 결정』이라고 합니다. 해당 프로세스가 여기에 설명되어 있습니다.

1. 가장 구체적인 적용 가능한 권한 부여 메소드가 속해 있는 메소드 계층 구조에서 원래 메소드를 찾으십시오. 이를 *루트 메소드*라고 합니다.
2. 다음을 포함하는 디스패치 가능 메소드 세트를 작성하십시오.
   - 가장 구체적인 적용 가능 권한 부여 메소드.
   - 가장 구체적인 적용 가능 권한 부여 메소드를 대체하며 이 호출 주제의 부속 유형인 유형에 대해 정의되어 있는 메소드.
3. 다음과 같이 가장 구체적인 디스패치 가능 메소드를 결정하십시오.
   a. 디스패치 가능 메소드 세트의 요소이자 주제의 동적 유형 또는 해당 슈퍼 유형 중 하나의 메소드인 임의 메소드로 시작하십시오. 이는 초기의 가장 구체적인 디스패치 가능 메소드입니다.
   b. 디스패치 가능 메소드 세트의 요소로 반복하십시오. 각 메소드의 경우: 메소드가 가장 구체적인 디스패치 가능 메소드가 정의된 유형의 적절한 부속 유형 중 하나에 대해 정의된 경우와 주제에 대한 가장 구체적인 유형의 슈퍼 유형 중 하나에 대해 정의된 경우, 가장 구체적인 디스패치 가능 메소드로서 이 메소드를 사용하여 2단계를 반복하십시오. 그 외에는 반복을 계속하십시오.
4. 가장 구체적인 디스패치 가능 메소드를 호출하십시오.

예:

″Person″, ″Employee″ 및 ″Manager″의 세 가지 유형이 제공됩니다. 개인의 수입을 계산하는 ″Person″에 대해 정의된 원래 메소드 ″income″이 있습니다. 개인은 디폴트로 수입이 없습니다(어린이, 퇴직자 등). 따라서 ″Person″ 유형의 ″income″은 항상 0을 리턴합니다. ″Employee″ 유형 및 ″Manager″ 유형의 경우 수입을 계산하는 데 다른 알고리즘이 적용되어야 합니다. 따라서 ″Person″ 유형의 ″income″ 메소드는 ″Employee″ 및 ″Manager″로 대체됩니다.

다음과 같이 테이블을 작성한 후 채우십시오.

```
CREATE TABLE aTable (id integer, personColumn Person);
  INSERT INTO aTable VALUES (0, Person()), (1, Employee()), (2, Manager());
```

수입이 $40000 이상인 모든 개인을 나열하십시오.

```
SELECT id, person, name
  FROM aTable
  WHERE person..income() >= 40000;
```

가장 구체으로 적용 가능한 권한 부여 메소드가 되도록 ″Person″ 유형의 ″income″ 메소드가 메소드 결정을 사용하여 선택됩니다.

1. 루트 메소드는 ″Person″ 자체의 ″income″입니다.
2. 위의 알고리즘의 두 번째 단계가 디스패치 가능 메소드 세트를 작성하기 위해 실행됩니다.
   - ″Person″ 유형의 ″income″ 메소드는 가장 구체적으로 적용 가능한 권한 부여 메소드이므로 포함됩니다.
   - ″Employee″ 유형의 ″income″ 메소드와 ″Manager″ 유형의 ″income″ 메소드는 모두 루트 메소드를 대체하고 ″Employee″ 및 ″Manager″는 모두 ″Person″의 부속 유형이므로, 이러한 메소드가 포함됩니다.

   따라서 디스패치 가능 메소드 세트는 ″Person″의 경우 ″income″, ″Employee″의 경우 ″income″, ″Manager″의 경우 ″income″입니다.
3. 가장 구체적인 디스패치 가능 메소드를 결정하십시오.
   - 가장 구체적인 유형이 ″Person″인 주제의 경우
     a. 초기의 가장 구체적인 디스패치 가능 메소드가 ″Person″ 유형의 경우 ″income″이 됩니다.
     b. ″Person″의 적절한 부속 유형 및 주제에 대한 가장 구체적인 유형의 슈퍼 유형에 대해 정의된 디스패치 가능 메소드 세트에 다른 메소드가 없으므로, ″Person″ 유형의 ″income″은 가장 구체적인 디스패치 가능 메소드입니다.
   - 가장 구체적인 유형이 ″Employee″인 주제의 경우
     a. 초기의 가장 구체적인 디스패치 가능 메소드가 ″Person″ 유형의 경우 ″income″이 됩니다.

b. 디스패치 가능 메소드 세트로 반복하십시오. ″Employee″ 유형의 ″income″ 메소드가 ″Person″의 적절한 부속 유형 및 주제에 대한 가장 구체적인 유형의 슈퍼 유형에 대해 정의되어 있으므로(참고: 한 유형은 고유한 슈퍼 및 부속 유형임), ″Employee″ 유형의 ″income″ 메소드는 가장 구체적인 디스패치 가능 메소드에 더 적합합니다. 가장 구체적인 디스패치 가능 메소드로서 ″Employee″ 유형의 ″income″ 메소드를 사용하여 이 단계를 반복하십시오.

c. ″Employee″의 적절한 부속 유형 및 주제에 대한 가장 구체적인 유형의 슈퍼 유형에 대해 정의된 디스패치 가능 메소드 세트에 다른 메소드가 없으므로, ″Employee″ 유형의 ″income″ 메소드는 가장 구체적인 디스패치 가능 메소드입니다.

- 가장 구체적인 유형이 ″Manager″인 주제의 경우

  a. 초기의 가장 구체적인 디스패치 가능 메소드가 ″Person″ 유형의 경우 ″income″이 됩니다.

  b. 디스패치 가능 메소드 세트로 반복하십시오. ″Manager″ 유형의 ″income″ 메소드가 ″Person″의 적절한 부속 유형 및 주제에 대한 가장 구체적인 유형의 슈퍼 유형에 대해 정의되어 있으므로(참고: 한 유형은 고유한 슈퍼 및 부속 유형임), ″Manager″ 유형의 ″income″ 메소드는 가장 구체적인 디스패치 가능 메소드에 더 적합합니다. 가장 구체적인 디스패치 가능 메소드로서 ″Manager″ 유형의 ″income″ 메소드를 사용하여 단계를 반복하십시오.

  c. ″Manager″의 적절한 부속 유형 및 주제에 대한 가장 구체적인 유형의 슈퍼 유형에 대해 정의된 디스패치 가능 메소드 세트에 다른 메소드가 없으므로, ″Manager″ 유형의 ″income″ 메소드는 가장 구체적인 디스패치 가능 메소드입니다.

4. 가장 구체적인 디스패치 가능 메소드를 호출하십시오.

제한적 바인딩 시맨틱

SQL 오브젝트를 정의하거나 패키지 바인드 조작을 처리할 때 오브젝트 분석이 수행됩니다.

데이터베이스 관리 프로그램은 DDL문에서 참조되거나 응용프로그램에서 코딩되는 SQL 오브젝트에 사용되도록 정의된 특정 SQL 오브젝트를 선택합니다.

이후에 데이터베이스 관리 프로그램은 원래 SQL 오브젝트가 변경되지 않은 경우에도 다른 SQL 오브젝트로 해석할 수도 있습니다. 오브젝트 분석 알고리즘이 원래 선택한 SQL 오브젝트 이전에 분석되도록 정의하는 다른 SQL 오브젝트를 정의하면(또는 기존 함수에 특권을 추가하면) 다른 SQL 오브젝트에 대해 이 분석이 수행됩니다. 다른 SQL 오브젝트에 대한 이 분석이 적용되는 SQL 오브젝트 및 상황의 예에는 다음이 포함됩니다.

- 루틴 - 이전의 SQL 경로에 있던 루틴보다 더 향상 또는 동등한 품질의 새 루틴을 정의하거나 이전의 SQL 경로에 있는 루틴보다 더 향상 또는 동등한 품질의 기존 루틴에 특권을 부여할 수 있습니다.
- 사용자 정의 데이터 유형 - 새 사용자 정의 데이터 유형은 동일한 이름으로 이전 SQL 경로에 있던 스키마에 정의할 수 있습니다.
- 전역 변수 - 새 전역 변수는 동일한 이름으로 이전 SQL 경로에 있는 스키마에 정의할 수 있습니다.
- 공용 별명으로 분석하는 테이블 또는 뷰 - 실제 테이블, 뷰 또는 개인용 별명은 동일한 이름으로 현재 스키마에 정의할 수 있습니다.
- 공용 시퀀스 별명으로 분석하는 시퀀스 - 실제 시퀀스 또는 개인용 시퀀스는 동일한 이름으로 현재 스키마에 정의할 수 있습니다.
- 공용 모듈 별명으로 분석하는 모듈 - 실제 모듈 또는 개인용 모듈 별명은 동일한 이름으로 SQL 경로에 있는 스키마에 정의할 수 있습니다.

데이터베이스 관리 프로그램이 명령문 처리 시에 원래 분석한 대로 SQL 오브젝트 분석을 반복할 수 있는 인스턴스가 있습니다. 다음 정적 오브젝트가 사용되는 경우에 적용됩니다.

- 패키지의 정적 DML문
- 뷰
- 트리거
- 점검 제한조건
- SQL 루틴
- 사용자 정의 유형 또는 디폴트 표현식이 포함된 전역 변수
- 사용자 정의 매개변수 유형 또는 디폴트 표현식이 포함된 루틴

패키지의 정적 DML문의 경우 SQL 오브젝트 참조는 바인드 조작 중에 분석됩니다. 뷰, 트리거, SQL 루틴 및 점검 제한조건에서의 SQL 오브젝트 참조는 SQL 오브젝트 정의 또는 유효성 다시 확인 중에 분석됩니다. 기존의 정적 오브젝트가 사용되는 경우 데이터베이스 스키마에서의 변경으로 인해 오브젝트가 유효하지 않거나 작동 불능으로 표시된 경우가 아니면 *제한적 바인딩 시맨틱*이 적용됩니다.

제한적 바인딩 시맨틱을 사용하면 SQL 오브젝트 참조는 이전에 분석될 때 사용된 동일한 SQL 경로, 디폴트 스키마 및 루틴 세트를 사용하여 분석됩니다. 제한적 바인딩 분석 중에 고려된 SQL 오브젝트 정의의 시간소인은 명령문이 *비제한적 바인딩* 시맨틱을 사용하여 마지막으로 바인드되거나 유효성이 확인된 시간소인 이후가 아니여야 합니다. 비제한적 바인딩 시맨틱은 패키지나 명령문의 원래 생성과 동일한 SQL 경로 및 디폴트 스키마를 사용하지만 SQL 오브젝트 정의의 시간소인 및 이전에 분석된 루틴 세트도 고려하지 않습니다.

오브젝트 삭제, 오브젝트 변경 또는 특권 부여 취소와 같은 데이터베이스 스키마의 일부 변경으로 인해 데이터베이스 관리 프로그램이 제한적 바인딩 시맨틱을 사용하는 기존 SQL 오브젝트의 모든 종속된 SQL 오브젝트를 더 이상 분석할 수 없게 되기 때문에 SQL 오브젝트에도 영향을 줄 수 있습니다.

- 이 경우 SQL 패키지의 정적 명령문에서 패키지는 작동 불능으로 표시됩니다. 이 패키지에서 다음에 명령문을 사용하면 해당 패키지가 작동 불능이 되도록 하는 데이터베이스 스키마의 최신 변경사항을 고려하여 SQL 오브젝트를 분석할 수 있도록 비제한적 바인딩 시맨틱을 사용하여 패키지가 내재적으로 리바인드됩니다.
- 이 경우 뷰, 트리거, 점검 제한조건 또는 SQL 루틴에서는 SQL 오브젝트가 유효하지 않음으로 표시됩니다. 오브젝트를 다음에 사용할 때 비제한적 바인딩 시맨틱을 사용하여 SQL 오브젝트의 유효성이 내재적으로 다시 확인됩니다.

시그니처 SCHEMA1.BAR(INTEGER) 및 SCHEMA2.BAR(DOUBLE)이 있는 두 가지 함수가 포함된 데이터베이스를 고려해 보십시오. SQL 경로에 SCHEMA1 및 SCHEMA2가 모두 포함되어 있다고 가정해 보십시오(SQL 경로 내의 순서가 상관없는 경우에도). USER1에는 함수 SCHEMA2.BAR(DOUBLE)에 대한 EXECUTE 특권이 부여되었습니다. USER1이 BAR(INT_VAL)을 호출하는 뷰를 작성한다고 가정해 보십시오. 여기서, INT_VAL은 INTEGER 데이터 유형으로 정의한 컬럼이나 전역 변수입니다. USER1은 SCHEMA1.BAR(INTEGER)에 대한 EXECUTE 특권이 없기 때문에 뷰에서 이 함수의 참조는 함수 SCHEMA2.BAR(DOUBLE)로 분석됩니다. 뷰가 작성된 후에 USER1에 SCHEMA1.BAR(INTEGER)에 대한 EXECUTE 특권이 부여된 경우 데이터베이스 스키마 변경으로 뷰가 유효하지 않음으로 표시된 경우를 제외하고는 뷰는 계속 SCHEMA2.BAR(DOUBLE)을 사용합니다. 뷰는 필요한 특권이 취소되었거나 특권이 종속된 오브젝트가 삭제 또는 변경된 경우에 유효하지 않음으로 표시됩니다.

패키지의 정적 DML의 경우 패키지는 내재적으로 또는 명시적으로 REBIND 명령(또는 해당 API)을 실행하거나 BIND 명령(또는 해당 API)을 실행하여 리바인드할 수 있습니다. 패키지가 유효하지 않음으로 표시되었지만 패키지가 작동 불능으로 표시되어 비제한적 바인딩 시맨틱을 사용하는 경우 내재된 리바인드는 제한적 바인딩 시맨틱을 사용하여 수행됩니다. 패키지는 종속된 인덱스가 삭제 또는 변경된 경우에만 유효하지 않음으로 표시됩니다. REBIND 명령은 제한적 바인딩 시맨틱(RESOLVE CONSERVATIVE)으로 분석 또는 새 루틴, 데이터 유형 또는 전역 변수(디폴트 옵션인 RESOLVE ANY)로 분석하는 옵션을 제공합니다. RESOLVE CONSERVATIVE 옵션은 데이터베이스 관리 프로그램이 패키지를 작동 불능으로 표시하지 않은 경우에만 사용할 수 있습니다(SQLSTATE 51028).

이 주제에서 제한적 바인딩 시맨틱의 설명은 데이터베이스 구성 매개변수인 **auto_reval**의 설정이 DISABLED가 아닌 것으로 간주합니다. 새 데이터베이스의 디폴트는 DEFERRED이며 버전 9.7로 업그레이드된 데이터베이스의 디폴트는 DISABLED입니다. **auto_reval**이 DISABLED로 설정된 경우 제한적 바인딩 시맨틱, 무효화 및 유효성 다시 확인 동작은 버전 9.7 이전의 릴리스의 동작과 동일합니다. 이 설정에서 제한적 바인딩 시맨틱은 함수, 메소드, 사용자 정의 유형 및 전역 변수에 대한 SQL 오브젝트 정의의 시간소인만 고려합니다. 무효화 및 유효성 다시 확인 동작의 경우 DROP, REVOKE 및 ALTER문에서 이는 시맨틱이 더 제한적이거나 종속 오브젝트에 대한 영향이 연쇄되어 오브젝트가 삭제되는 것을 나타냅니다. 패키지의 경우 대부분의 데이터베이스 스키마 변경사항으로 인해 패키지는 유효하지 않음으로 표시되고 내재된 리바인드 중에 제한적 바인딩 시맨틱을 사용합니다. 그렇지만 종속 함수를 삭제하여 스키마가 변경되고 **auto_reval**이 DISABLED로 설정되면 함수에 종속된 패키지는 작동 불능으로 표시되고 작동 불능 패키지에 대한 내재된 리바인드는 없습니다.

표현식

표현식은 값을 지정합니다. 표현식은 단 하나의 상수나 컬럼 이름으로 구성되는 간단한 값이 되거나, 더 복잡할 수 있습니다. 반복적으로 유사한 복합 표현식을 사용할 경우, 일반 표현식을 캡슐화하는 SQL 함수를 고려해 볼 수 있습니다.

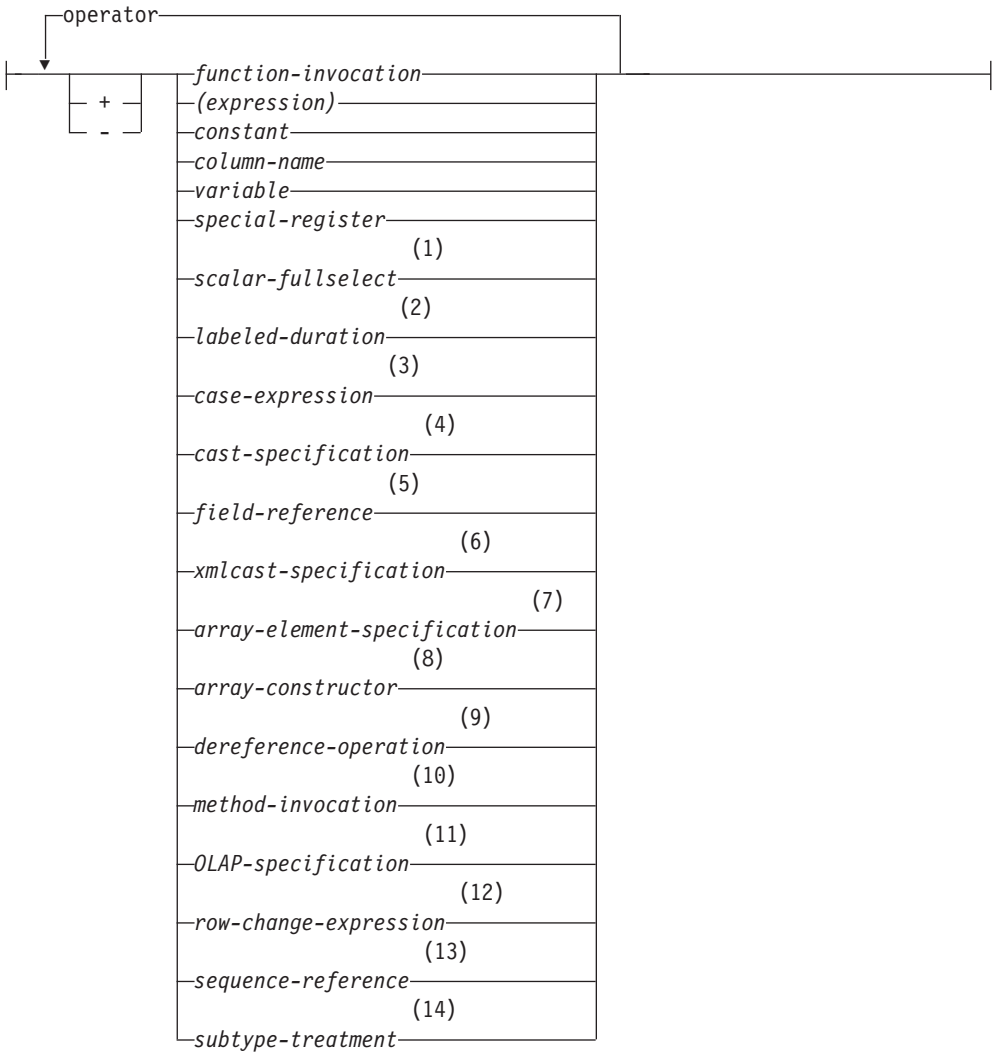
권한 부여

scalar-subselect, sequence-reference 또는 function-invocation과 같은 일부 표현식을 사용하려면, 적절한 권한이 필요할 수 있습니다. 이와 같은 표현식의 경우, 명령문의 권한 부여 ID가 보유하는 특권에 다음 권한이 포함되어야 합니다.

- scalar-subselect. 권한 부여 고려사항에 대한 정보는 ″쿼리″를 참조하십시오.
- sequence-reference. 시퀀스를 참조하기 위한 권한. 권한 부여 고려사항에 대한 정보는 ″시퀀스 권한″을 참조하십시오.
- function-invocation. 사용자 정의 함수(UDF)를 실행하기 위한 권한. 권한 부여 고려사항에 대한 정보는 ″함수 호출″을 참조하십시오.
- 변수. 변수가 전역 변수인 경우, 전역 변수를 참조하기 위한 권한이 필요합니다. 자세한 정보는 전역 변수를 참조하십시오.

유니코드 데이터베이스에서 문자 또는 그래픽 문자열을 허용하는 표현식은 변환이 지원되는 모든 문자열 유형을 허용합니다.

expression:



operator:

주:

1 자세한 정보는 245 페이지의 『스칼라 fullselect』의 내용을 참조하십시오.

2 자세한 정보는 246 페이지의 『지속 기간』의 내용을 참조하십시오.

3 자세한 정보는 252 페이지의 『CASE 표현식』의 내용을 참조하십시오.

4 자세한 정보는 255 페이지의 『CAST 스펙』의 내용을 참조하십시오.

5 자세한 정보는 261 페이지의 『필드 참조』의 내용을 참조하십시오.

6 자세한 정보는 262 페이지의 『XMLCAST 스펙』의 내용을 참조하십시오.

7 자세한 정보는 264 페이지의 『ARRAY 요소 스펙』의 내용을 참조하십시오.

8 자세한 정보는 265 페이지의 『배열 컨스트럭터』의 내용을 참조하십시오.

9 자세한 정보는 267 페이지의 『비참조 조작』의 내용을 참조하십시오.

10 자세한 정보는 269 페이지의 『메소드 호출』의 내용을 참조하십시오.

11 자세한 정보는 271 페이지의 『OLAP 스펙』의 내용을 참조하십시오.

12 자세한 정보는 281 페이지의 『ROW CHANGE 표현식』의 내용을 참조하십시오.

13 자세한 정보는 283 페이지의 『시퀀스 참조』의 내용을 참조하십시오.

14 자세한 정보는 288 페이지의 『부속 유형 처리』의 내용을 참조하십시오.

15 CONCAT에 대한 동의어로 ||를 사용할 수 있습니다.

연산자 없는 표현식

연산자가 사용되지 않을 경우, 표현식의 결과는 지정된 값이 됩니다.

예:

```
SALARY:SALARY'SALARY'MAX(SALARY)
```

병합 연산자를 사용하는 표현식

병합 연산자(CONCAT)는 두 개의 피연산자를 조합하여 *문자열 표현식*을 구성합니다.

첫 번째 피연산자는 문자열 데이터 유형, 숫자 데이터 유형 또는 날짜 및 시간 데이터 유형의 값을 리턴하는 표현식입니다. 두 번째 피연산자도 문자열 데이터 유형, 숫자 데이터 유형 또는 날짜 및 시간 데이터 유형의 값을 리턴하는 표현식입니다. 그러나 일부 데이터 유형은 아래에 설명된대로 첫 번째 피연산자의 데이터 유형이 있는 조합에서 지원되지 않습니다.

피연산자는 문자열(실행 파일 문자열 제외), 숫자 및 날짜 시간 값의 조합이 될 수 있습니다. 피연산자가 문자열 값이 아닌 경우 내재적으로 VARCHAR로 캐스트됩니다. 실행 파일 문자열은 다른 실행 파일 문자열과만 병합될 수 있습니다. 그러나 함수 결정의 캐스트 가능한 프로세스를 통해 첫 번째 피연산자가 실행 파일 문자열일 때 실행 파일 문자열은 FOR BIT DATA로 정의된 문자열과 병합될 수 있습니다.

문자열 피연산자와 그래픽 문자열 피연산자 모두를 포함하는 병합은 유니코드 데이터베이스에서만 지원됩니다. 병합 전에 문자 피연산자가 먼저 그래픽 데이터 유형으로 변환됩니다. FOR BIT DATA로 정의된 문자열은 그래픽 데이터 유형으로 캐스트될 수 없습니다.

피연산자 중 하나가 널(NULL)이 될 수 있는 경우, 결과는 널(NULL)이 될 수 있으며, 피연산자 중 하나가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다. 그렇지 않은

경우, 결과는 첫 번째 피연산자 문자열과 그 뒤에 오는 두 번째 피연산자 문자열로 구성됩니다. 병합이 수행될 때 부적절하게 구성된 혼합 데이터에 대해 어떠한 점검도 수행되지 않음에 유의하십시오.

결과의 길이는 피연산자 길이의 합입니다.

결과의 데이터 유형과 길이 속성은 다음 표에 표시된 것처럼 피연산자의 데이터 유형과 길이 속성으로부터 결정됩니다.

표 21. 병합된 피연산자의 데이터 유형과 길이

피연산자	조인된 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
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 호환성을 위해, LONG VARCHAR 또는 LONG VARGRAPHIC 데이터 유형을 포함한 결과가 LOB 데이터 유형으로 자동으로 에스컬레이션되지 않습니

다. 예를 들어, CHAR(200) 값과 완전한 전체 LONG VARCHAR 값을 병합할 경우 CLOB 데이터 유형으로의 승격되는 것이 안라, 오류가 발생합니다.

결과 코드 페이지는 파생된 코드 페이지로 간주되며, 해당 피연산자의 코드 페이지에 따라 결정됩니다.

하나의 피연산자는 매개변수 표시문자가 될 수 있습니다. 매개변수 표시문자가 사용되는 경우, 해당 피연산자의 데이터 유형과 길이 속성은 비매개변수 표시문자 피연산자의 것과 같은 것으로 간주됩니다. 중첩된 병합의 경우 이러한 속성을 결정하는 데 연산의 순서를 고려해야 합니다.

예 1: FIRSTNME이 Pierre이고 LASTNAME이 Fermat일 경우,

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

값 Pierre Fermat가 리턴됩니다.

예 2: 다음과 같이 주어질 경우,

- 값이 'AA'인 VARCHAR(5)로 정의된 COLA
- 길이가 5이고 값이 'BB '인 문자 호스트 변수로 정의된 :host_var
- 값이 'CC'인 CHAR(5)로 정의된 COLC
- 값이 'DDDDD'인 CHAR(5)로 정의된 COLD

COLA **CONCAT :** host_var **CONCAT** COLC **CONCAT** COLD의 값은 'AABB CC DDDDD' 입니다.

데이터 유형은 VARCHAR이고, 길이 속성은 17이며, 결과 코드 페이지는 section 코드 페이지입니다.섹션 코드 페이지에 대한 자세한 정보는 "코드 페이지 값의 파생"을 참조하십시오.

예 3: 다음과 같이 주어질 경우,

- CHAR(10)으로 정의된 COLA
- VARCHAR(5)로 정의된 COLB

다음 표현식의 매개변수 표시문자는

```
COLA CONCAT COLB CONCAT ?
```

VARCHAR(15)로 간주됩니다. 왜냐하면 COLA CONCAT COLB 먼저 평가됨으로써 그 결과로 두 번째 CONCAT 연산의 첫 번째 피연산자가 되기인 결과를 때문입니다.

사용자 정의 유형

소스 데이터 유형이 문자열 유형인 구별 유형일 경우에도, 사용자 정의 유형은 병합 연산자와 함께 사용할 수 없습니다. 병합하려면 CONCAT 연산자를 갖는 함수를 소스로

작성하십시오. 예를 들어, 구별 유형 TITLE과 TITLE_DESCRIPTION이 있는데 둘 다 VARCHAR(25) 데이터 유형인 경우, 다음 사용자 정의 함수 ATTACH는 이들을 병합하는 데 사용할 수 있습니다.

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

또 다른 방법으로, 새 데이터 유형을 추가하는 사용자 정의 함수를 사용하여 병합 연산자를 오버로드할 수 있습니다.

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

산술 연산자를 사용하는 표현식

산술 연산자를 사용할 경우, 표현식의 결과는 피연산자의 값에 연산자를 적용하여 파생된 값이 됩니다.

피연산자가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우, 결과는 널(NULL)이 될 수 있습니다.

피연산자에 널(NULL)이 있는 경우, 표현식의 결과는 널(NULL)이 됩니다.

산술 연산자는 부호가 있는 숫자 유형과 날짜 시간 유형에 적용될 수 있습니다(247 페이지의 『SQL에서 날짜 시간 산술』 참조). 예를 들어, USER+2는 유효하지 않습니다. 전래 함수는 소스 유형이 부호가 있는 숫자 유형인 구별 유형에서 산술 연산에 대해 정의될 수 있습니다.

접두부 연산자 +(단일 더하기)는 해당 피연산자를 변경하지 않습니다. 접두부 연산자 -(단항 빼기)는 0이 아닌 10진수 부동 소수점 피연산자의 부호를 역으로 합니다. 접두부 연산자 -(단항 빼기)는 0과 특정값 즉, 기호가 있거나 부호없는 NaN 및 더하기와 빼기 무한대를 포함한 모든 10진수 부동 소수점 피연산자의 부호를 역으로 합니다. A의 데이터 유형이 작은 정수인 경우, -A의 데이터 유형은 큰 정수입니다. 접두부 연산자 뒤에 오는 토큰의 첫 번째 문자는 더하기 또는 빼기 부호가 될 수 없습니다.

인픽스 연산자 +, -, *, /는 더하기, 빼기, 곱하기, 나누기를 각각 지정합니다. 나누기에서 두 번째 피연산자의 값은 0이 될 수 없습니다(10진수 부동 소수점 산술을 사용하여 계산한 경우는 제외). 이러한 연산자는 함수로 처리될 수도 있습니다. 따라서 표현식 *"+"(a,b)*는 표현식 *a+b*. 『연산자』 함수와 동일합니다.

LOB를 제외하고 문자 또는 그래픽 문자열 데이터 유형의 피연산자는 산술 연산을 수행하기 전에 CAST 스펙의 규칙을 사용하여 DECFLOAT(34)로 변환됩니다. 자세한 정보는 『데이터 유형 사이의 캐스팅』을 참조하십시오. 그래픽 문자열 피연산자를 포함하는 산술은 유니코드 데이터베이스에서만 지원됩니다.

문자열 데이터 유형의 피연산자는 산술 연산을 수행하기 전에 CAST 스펙의 규칙을 사용하여 DECFLOAT(34)로 변환됩니다. 자세한 정보는 『데이터 유형 사이의 캐스팅』을 참조하십시오. 문자열은 유효한 숫자 표현을 포함해야 합니다.

산술 오류

10진수가 아닌 부동 소수점 표현식 처리 중에 0으로 나누기 또는 숫자 오버플로우와 같은 산술 오류가 발행할 경우, 오류가 리턴됩니다(SQLSTATE 22003 또는 22012). 10진수 부동 소수점 표현식의 경우, 산술 조건의 특성에 따라 경고가 리턴됩니다 (SQLSTATEs 0168C, 0168D, 0168E 또는 0168F).

산술 오류가 10진수가 아닌 부동 소수점 표현식에 대해 널(NULL) 값을 리턴하고, 쿼리가 경고(SQLSTATE 01519 또는 01564)를 리턴하고 SQL문의 처리를 계속하도록 데이터베이스를 구성(DFT_SQLMATHWARN를 예로 설정하여)할 수 있습니다.

10진수 부동 소수점 표현식의 경우, DFT_SQLMATHWARN은 영향이 없으며, 산술 조건은 적절한 값(가능한 10진수 부동 소수점 특정값)을 리턴하며, 쿼리는 경고를 리턴하며(SQLSTATE 0168C, 0168D, 0168E 또는 0168F), SQL문의 처리를 계속합니다. 리턴된 특정 값에는 더하기 및 빼기 무한대가 포함되며 번호가 포함되지 않습니다. 표현식에 대한 하나 이상의 인수가 널이 아닌 경우 하나 이상의 부동 소수점 숫자를 포함하는 산술 표현식은 널 값으로 평가되지 않습니다.

산술 오류가 널(NULL)로 취급될 경우, SQL문의 결과에 대해 의미하는 바가 있습니다. 다음은 이에 대한 예입니다.

- 집계 함수의 인수인 표현식에서 발생한 산술 오류로 인해 집계 함수의 결과를 결정할 때 행이 무시됩니다. 산술 오류가 오버플로우일 경우, 이는 결과 값에 중대한 영향을 줄 수 있습니다.
- WHERE절에 있는 술어의 표현식에서 발생한 산술 오류로 인해 결과에 행이 포함되지 않을 수 있습니다.
- 점검 제한조건에 있는 술어의 표현식에서 발생한 산술 오류로 인해 제한조건이 거짓이 아닐 수 있으므로 갱신이나 삽입 작업이 수행됩니다.

이러한 유형의 영향을 받아들일 수 없는 경우, 산술 오류를 처리하기 위한 추가 조치를 취하여 허용 가능한 결과를 산출해야 합니다. 예를 들면 다음과 같습니다.

- 0으로 나누기가 있는지 점검하는 CASE 표현식을 추가하고 그러한 경우에 대비하여 원하는 값을 설정합니다.
- 널(NULL)을 처리하는 추가 술어를 추가합니다. 널(NULL) 입력 불가능 컬럼에 대한 점검 제한조건과 같이, 다음과 같이 되어 오버플로우에 대하여 제한사항이 위반되게 할 수 있습니다.

```
check (c1*c2 is not null and c1*c2>5000)
```

두 개의 정수 피연산자

산술 연산자의 두 피연산자가 모두 정수인 경우, 2진으로 조작이 수행되며 결과는 *큰 정수*입니다. 두 피연산자(또는 둘 중의 하나)가 대정수이면 결과는 대정수입니다. 나누기의 나머지는 버립니다. 정수 산술 연산(단일 빼기 포함)의 결과는 결과 유형 범위 내에 있어야 합니다.

정수 및 소수 피연산자

한 피연산자가 정수이고 다른 피연산자가 소수인 경우, 정밀도가 *p*이고 스케일이 0인 소수로 변환된 정수의 임시 사본을 사용하여 연산이 소수로 수행됩니다. *p*는 대 정수의 경우 19, 큰 정수의 경우 11, 작은 정수의 경우 5입니다.

두 개의 10진수 피연산자

피연산자가 모두 소수인 경우, 연산은 소수로 수행됩니다. 소수 산술 연산의 결과는 연산과 피연산자의 정밀도 및 스케일에 따라 달라지는 정밀도와 스케일을 갖는 소수입니다. 연산이 더하기나 빼기이고 피연산자의 스케일이 같지 않은 경우, 연산은 피연산자 중 하나의 임시 사본에서 수행됩니다. 짧은 피연산자의 사본 뒤에 0이 첨부되어 확장되어 분수 부분이 긴 피연산자와 같은 자릿수를 갖도록 합니다.

소수 연산 결과의 정밀도는 31보다 클 수 없습니다. 소수의 더하기, 빼기, 곱하기의 결과는 정밀도가 31보다 클 수 있는 임시 결과에서 산출됩니다. 임시 결과의 정밀도가 31보다 크지 않은 경우, 마지막 결과는 임시 결과와 같습니다.

SQL에서 소수 연산

다음 공식은 SQL에서 소수 연산 결과의 정밀도와 스케일을 정의합니다. 부호 *p*와 *s*는 첫 번째 피연산자의 정밀도와 스케일을 나타내며, 부호 *p'*와 *s'*는 두 번째 피연산자의 정밀도와 스케일을 나타냅니다.

더하기 및 빼기

정밀도는 min(31,max(*p - s,p' - s'*) + max(*s,s'*) + 1)입니다. 더하기 및 빼기의 결과 스케일은 max(*s,s'*)입니다.

곱하기

곱하기 결과의 정밀도는 min(31,*p+p'*)이고, 스케일은 min(31,*s+s'*)입니다.

나누기

나누기 결과의 정밀도는 31입니다. 스케일은 *31-p+s-s'*입니다. 스케일은 음수가 될 수 없습니다.

주: min_dec_div_3 데이터베이스 구성 매개변수는 나누기를 포함하는 10진수 산술 연산에 대한 스케일을 변경합니다. 매개변수 값이 NO로 설정되는 경우, 스케일은 *31-p+s-s*'로 계산됩니다. 매개변수가 YES로 설정되면, 스케일은 MAX(3, *31-p+ s-s*')로 계산됩니다. 이것은 소수 나누기의 결과가 항상 최소한 3의 스케일을 갖도록 합니다. 정밀도는 항상 31입니다.

부동 소수점 피연산자

산술 연산자의 피연산자가 10진수 부동 소수점이 아닌 부동 소수점일 경우, 연산은 부동 소수점으로 수행됩니다. 필요한 경우 피연산자는 먼저 배정밀도 부동 소수점 숫자로 변환됩니다. 따라서 표현식의 어느 한 요소라도 부동 소수점 숫자일 경우, 표현식의 결과는 배정밀도 부동 소수점 숫자입니다.

부동 소수점 숫자와 정수를 포함하는 연산은 배정밀도 부동 소수점 숫자로 변환된 정수의 임시 사본에서 수행됩니다. 부동 소수점 숫자와 소수를 포함하는 연산은 배정밀도 부동 소수점 숫자로 변환된 소수의 임시 사본에서 수행됩니다. 부동 소수점 연산의 결과는 부동 소수점 숫자의 범위 내에 있어야 합니다.

부동 소수점 피연산자가 실수로 대략 표현되므로 부동 소수점 피연산자(또는 함수에 대한 인수)의 처리 순서가 결과에 다소 영향을 줄 수 있습니다. 피연산자의 처리 순서가 옵티마이저에 의해 내재적으로 수정될 수 있으므로(예를 들어, 옵티마이저는 사용될 병렬 처리 수준 및 사용될 액세스 플랜을 결정할 수 있음), 부동 소수점 피연산자를 사용하는 응용프로그램은 SQL문이 실행될 때마다 정확하게 동일한 값이 되는 결과에 의존하지 않아야 합니다.

10진수 부동 소수점 피연산자

산술 연산자의 피연산자가 10진수 부동 소수점인 경우, 연산은 10진수 부동 소수점에서 수행됩니다.

정수 및 10진수 부동 소수점 피연산자

하나의 피연산자가 작은 정수나 큰 정수이고 다른 피연산자가 DECFLOAT(*n*) 숫자인 경우, DECFLOAT(*n*) 숫자로 변환된 정수의 임시 복사본을 사용하여 DECFLOAT(*n*)에서 연산이 수행됩니다. 하나의 피연산자가 큰 정수이고 다른 피연산자가 10진수 부동 소수점 숫자인 경우, 큰 정수의 임시 복사본은 DECFLOAT(34) 숫자로 변환됩니다. 두 개의 10진수 부동 소수점 피연산자에 대한 규칙이 적용됩니다.

10진수 및 10진수 부동 소수점 피연산자

한 피연산자가 10진수이고 다른 피연산자가 10진수 부동 소수점 숫자인 경우, 10진수 숫자의 정밀도를 기반으로 10진수 부동 소수점 숫자로 변환된 10진수의 임시 복사본을 사용하여 연산이 10진수 부동 소수점에서 수행됩니다. 10진수 숫자 정밀도가 17보다 작으면, 10진수 숫자는 DECFLOAT(16) 숫자로 변

환되며, 그렇지 않은 경우 10진수 숫자는 DECFLOAT(34) 숫자로 변환됩니다. 두 개의 10진수 부동 소수점 피연산자에 대한 규칙이 적용됩니다.

부동 소수점 및 10진수 부동 소수점 피연산자

한 피연산자가 부동 소수점 숫자(REAL 또는 DOUBLE)이고 다른 피연산자가 DECFLOAT(*n*) 숫자인 경우, DECFLOAT(*n*) 숫자로 변환된 부동 소수점 숫자의 임시 복사본을 사용하여 10진수 부동 소수점으로 연산이 수행됩니다.

두 개의 10진수 부동 소수점 피연산자

두 개의 피연산자가 모두 DECFLOAT(*n*)인 경우, 연산은 DECFLOAT(*n*)로 수행됩니다. 한 피연산자가 DECFLOAT(16)이고 다른 피연산자가 DECFLOAT(34)인 경우, 연산은 DECFLOAT(34)에서 수행됩니다.

10진수 부동 소수점에 대한 일반 산술 연산 규칙

다음 일반 규칙이 10진수 부동 소수점 데이터 유형의 모든 산술 연산에 대해 적용됩니다.

- 한정된 숫자에서의 모든 연산은 가능한 한 상관 계수의 정수 산술을 사용하여 정확한 수학적 결과가 계산되는 것과 같이 이루어집니다.

  이론적인 정확한 결과의 상관계수가 더 이상 정밀도(16이나 34)를 반영하는 10진수 값보다 크지 않으면 변경 없이 결과에 사용됩니다(언더플로우나 오버플로우 조건이 아닌 경우). 상관계수가 정밀도를 반영하는 10진수 값보다 크면, 정밀도(16이나 34)를 반영하는 10진수의 숫자로 정확히 반올림하며, 지수는 제거된 10진수의 숫자로 늘어납니다.

  CURRENT DECFLOAT ROUNDING MODE 특수 레지스터는 반올림 모드를 판별합니다.

  조정된 결과의 지수 값이 E_{min}보다 작으면, 계산된 상관계수와 지수가 결과가 되며, 지수 값이 E_{tiny}보다 작지 않은 경우, 지수가 E_{tiny}로 설정된 경우에는 상관계수가 반올림되어(가능한 0으로) 조정된 지수와 일치하며 부호는 변경되지 않습니다. 이 근사값이 정확하지 않은 결과를 제공하면, 언더플로우 예외 조건이 리턴됩니다.

  결과의 조절된 지수 값이 E_{max}보다 크면, 오버플로우 예외 조건이 리턴됩니다. 이 경우, 결과는 오버플로우 예외 조건으로 정의되며 무한대가 될 수 있습니다. 이론적인 결과와 같은 부호가 됩니다.
- 특수한 값 무한대를 사용하는 산술은 일반 규칙을 따르며, 음수 무한대는 모든 유한대 숫자보다 작고 양수 무한대는 모든 유한대 숫자보다 큽니다. 이 규칙에서 무한대 결과는 항상 정확합니다. 무한대의 특정 사용은 유효하지 않은 연산 조건을 리턴합니다. 다음 목록은 유효하지 않은 연산 조건을 유발할 수 있는 연산을 나타냅니다. 피연산자의 하나가 무한대이고 다른 피연산자가 NaN이나 sNaN이 아닌 경우 이러한 연산의 결과는 NaN입니다.

- 더하기 또는 빼기 연산 시 +무한대를 -무한대에 추가
- 0을 +무한대나 -무한대에 곱하기
- +무한대나 -무한대 중 하나로 +무한대나 -무한대를 나누기
- QUANTIZE 함수의 인수 중 하나가 +무한대나 -무한대
- POWER 함수의 두 번째 인수가 +무한대나 -무한대
- 기호가 있는 NaN을 산술 연산의 피연산자로 사용

다음 규칙은 산술 연산 및 NaN 값에 적용됩니다.

- NaN(quiet 또는 signalling) 피연산자가 있는 산술 연산의 결과는 NaN입니다. 결과 기호는 기호가 있는 NaN인 첫 번째 피연산자에서 복사됩니다. 피연산자가 기호가 없으면 기호는 NaN인 첫 번째 피연산자에서 복사됩니다. 결과가 NaN일 때마다, 결과의 기호는 복사된 피연산자에 따라 다릅니다.
- 피연산자가 다른 기호이고 NaN이 아닌 경우에만 곱셈 또는 나눗셈 작업의 결과 기호는 음수입니다.
- 결과가 음수 0인 다음 경우를 제외하고는 결과가 0보다 작고 피연산자가 NaN이 아닌 경우에만, 더하기나 빼기 작업의 결과 기호는 음수입니다.
  - 결과는 0으로 반올림되며 반올림 전의 값이 음수 기호
  - -0이 0에 추가됨
  - 0을 -0에서 빼기
  - 반대 기호가 있는 피연산자가 추가되거나 같은 기호의 피연산자를 뺍니다. 결과는 0의 상관계수가 포함되며 근사값 모드는 ROUND_FLOOR임
  - 피연산자가 곱해지거나 나누어지며 결과는 0의 상관계수이고, 피연산자의 부호가 다름
  - POWER 함수의 첫 번째 인수는 -0이며, 두 번째 인수는 양의 홀수임
  - CEIL, FLOOR 또는 SQRT 함수의 인수가 -0임
  - ROUND 또는 TRUNCATE 함수의 첫 번째 인수가 -0임

다음 예제는 피연산자로 특정 10진수 부동 소수점 값을 나타냅니다.

```
INFINITY + 1          = INFINITY
INFINITY + INFINITY   = INFINITY
INFINITY + -INFINITY  = NAN         -- warning
NAN + 1               = NAN
NAN + INFINITY        = NAN
1 - INFINITY          = -INFINITY
INFINITY - INFINITY   = NAN         -- warning
-INFINITY - -INFINITY = NAN         -- warning
-0.0 - 0.0E1          = -0.0
-1.0 * 0.0E1          = -0.0
1.0E1 / 0             = INFINITY    -- warning
-1.0E5 / 0.0          = -INFINITY   -- warning
1.0E5 / -0            = -INFINITY   -- warning
```

```
INFINITY / -INFINITY  = NAN         -- warning
INFINITY / 0          = INFINITY
-INFINITY / 0         = -INFINITY
-INFINITY / -0        = INFINITY
```

피연산자로서 사용자 정의 유형

사용자 정의 유형은 해당 소스 데이터 유형이 숫자인 경우에도 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 DECIMAL(8,2) 데이터 유형인 경우, 다음에 오는 사용자 정의 함수 REVENUE는 다른 것에서 하나를 빼는 데 사용될 수 있습니다.

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

또 다른 방법으로, 사용자 정의 함수를 사용하여 -(빼기) 연산자를 오버로드하여 새 데이터 유형을 뺄 수 있습니다.

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

연산 우선순위

괄호 안의 표현식과 비참조 연산자가 먼저 왼쪽에서 오른쪽으로 평가됩니다. (괄호는 fullselects, 검색 조건 및 함수에서도 사용됩니다. 그러나 SQL문 내에서 그룹 섹션에 임의적으로 사용해서는 안됩니다. 평가 순서가 괄호로 지정되지 않은 경우, 곱하기와 나누기보다 먼저 접두부 연산자가 적용되고 더하기와 빼기보다 먼저 곱하기와 나누기가 적용됩니다. 같은 우선순위 레벨에 있는 연산자의 경우는 왼쪽에서 오른쪽으로 적용됩니다.

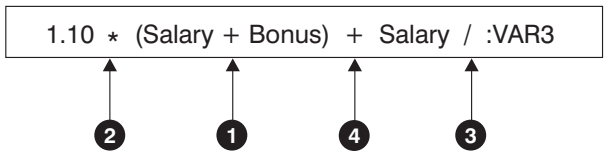


그림 11. 연산 우선순위

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)이 됩니다. 선택 목록 요소가 컬럼 이름만 있거나 비참조 연산자만 있는 표현식인 경우, 결과 컬럼 이름은 해당 컬럼 이름에 기초합니다. 스칼라 fullselect에 필요한 권한 부여는 SQL 쿼리에 필요한 것과 동일합니다.

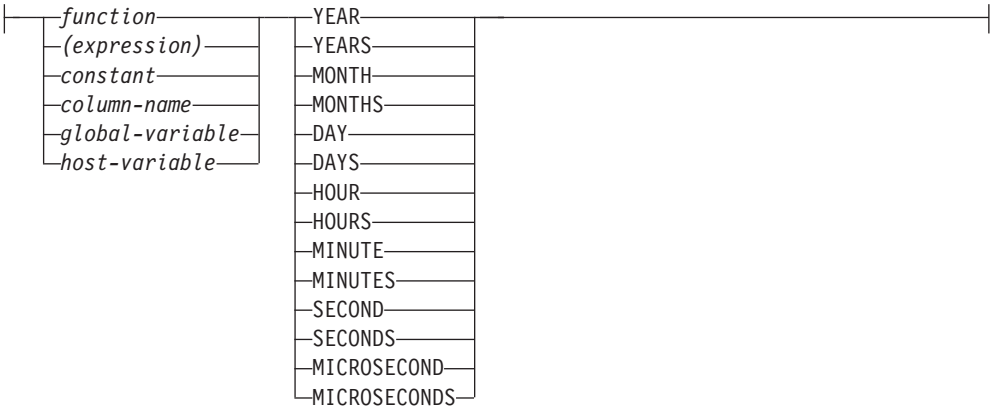
날짜 시간 연산 및 지속기간

날짜 시간 값은 빼거나 증가 또는 감소할 수 있습니다. 이러한 연산에는 지속 기간이라는 10진수가 포함될 수 있습니다. 다음 섹션에서는 지속 기간 유형을 설명하고 날짜 시간 산술에 대한 규칙을 자세히 설명합니다.

지속 기간

*지속 기간*은 시간의 간격을 표시하는 숫자입니다. 지속 기간 유형에는 네 가지가 있습니다.

labeled-duration:



*레이블된 지속 기간*은 7개의 지속 기간 키워드인 YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS 또는 MICROSECONDS 중 하나가 뒤에 오는 숫자(표현식의 결과가 될 수 있음)로 표시되는 특정 시간 단위를 나타냅니다. YEAR, MONTH, DAY, HOUR, MINUTE, SECOND 및 MICROSECOND와 같은 이 키워드의 하나의 양식도 사용할 수 있습니다. 지정된 숫자는 DECIMAL(15,0) 숫자에 지정된 것처럼 변환됩니다. 0 - 12 자리의 소수 초 수가 포함될 수 있도록 DECIMAL(27,12)을 사용하는 SECONDS의 경우는 예외입니다. 레이블된 지속 기간은 산술 연산자의 피연산자로 사용될 수 있으며, 다른 피연산자는 데이터 유형 DATE, TIME 또는 TIMESTAMP의 값이 됩니다. 그러므로 표현식 HIREDATE + 2 MONTHS + 14 DAYS는 유효한 반면 표현식 HIREDATE +(2 MONTHS + 14 DAYS)는 유효하지 않습니다. 이러한 두 표현식의 경우, 레이블된 지속 기간은 2 MONTHS와 14 DAYS입니다.

*날짜 기간*은 DECIMAL(8,0) 숫자로 표시되며 연도, 월, 일의 숫자를 나타냅니다. 제대로 해석하려면 숫자의 형식은 *yyyymmdd*이어야 합니다. 여기서, *yyyy*는 연도, *mm*은 월, *dd*는 일의 숫자를 나타냅니다. 이 형식에서 기간은 DECIMAL 데이터 유형을 나타냅니다. HIREDATE - BRTHDATE 표현식에서와 같이 다른 날짜에서 한 날짜를 뺀 결과가 날짜 기간입니다.

*시간 지속 기간*은 DECIMAL(6,0) 숫자로 표시되는 시, 분, 초를 나타냅니다. 제대로 해석하려면 숫자의 형식은 *hhmmss*이어야 합니다. 여기서, *hh*는 시, *mm*은 분, *ss*는 초를 나타냅니다. 이 형식에서 기간은 DECIMAL 데이터 유형을 나타냅니다. 다른 시간 값에서 한 시간 값을 뺀 결과가 시간 지속 기간입니다.

*시간소인 지속 기간*은 DECIMAL(14+*s*,*s*) 숫자로 표시되는 연도, 월, 일, 시, 분, 초 및 소수 초를 나타냅니다. 여기서 *s*는 0 - 12 범위의 소수 초 자릿수입니다. 적절하게 해석하려면 숫자의 형식이 *yyyymmddhhmmss.nnnnnnnnnnnn*이어야 하며, 여기서 *yyyy*, *mm*, *dd*, *hh*, *mm*, *ss* 및 *nnnnnnnnnnnn*은 각각, 연도, 월, 일, 시, 분, 초 및 소수 초를 나타냅니다. 다른 시간소인에서 한 시간소인 값을 뺀 결과가 시간소인 지속 기간이 됩니다. 스케일은 시간소인 피연산자의 최대 시간소인 정밀도와 일치합니다.

SQL에서 날짜 시간 산술

날짜 시간 값에서 수행할 수 있는 유일한 산술 연산은 더하기와 빼기입니다. 날짜 시간 값이 더하기의 피연산자인 경우, 다른 피연산자는 지속 기간이어야 합니다. 날짜 시간 값이 있는 더하기 연산자의 사용에 대해 특별히 적용되는 규칙은 다음과 같습니다.

- 한 피연산자가 날짜인 경우, 다른 피연산자는 날짜 기간 또는 레이블된 지속 기간인 YEARS, MONTHS 또는 DAYS여야 합니다.
- 한 피연산자가 시간인 경우, 나머지 피연산자는 시간 지속 기간 또는 레이블된 지속 기간인 HOURS, MINUTES 또는 SECONDS여야 합니다.
- 한 피연산자가 시간 소인인 경우, 나머지 피연산자는 지속 기간이어야 합니다. 어떤 유형의 지속 기간이든지 유효합니다.
- 더하기 연산자의 어떤 피연산자도 매개변수 표시문자가 될 수 없습니다.

날짜 시간 값에 대한 빼기 연산자의 사용 규칙은 더하기의 사용 규칙과 같지 않습니다. 왜냐하면 날짜 시간 값은 지속 기간에서 뺄 수 없고, 두 날짜 시간 값의 빼기 연산은 날짜 시간 값에서 지속 기간을 빼는 연산과 같지 않기 때문입니다. 날짜 시간 값이 있는 빼기 연산자의 사용에 대해 특별히 적용되는 규칙은 다음과 같습니다.

- 첫 번째 피연산자는 timestamp인 경우 두 번째 피연산자는 date, timestamp, date의 문자열 표시, timestamp의 문자열 표시 또는 duration이어야 합니다.
- 두 번째 피연산자가 timestamp인 경우 첫 번째 피연산자는 date, timestamp, date의 문자열 표시 또는 timestamp의 문자열 표시여야 합니다.
- 첫 번째 피연산자가 날짜인 경우, 두 번째 피연산자는 날짜, 날짜 기간, 날짜의 문자열 표현 또는 레이블된 지속 기간인 YEARS, MONTHS 또는 DAYS여야 합니다.
- 두 번째 피연산자가 날짜인 경우, 첫 번째 피연산자는 날짜 또는 날짜의 문자열 표현이어야 합니다.
- 첫 번째 피연산자가 시간인 경우, 두 번째 피연산자는 시간, 시간 지속 기간, 시간의 문자열 표현 또는 레이블된 지속 기간인 HOURS, MINUTES 또는 SECONDS여야 합니다.

- 두 번째 피연산자가 시간인 경우, 첫 번째 피연산자는 시간 또는 시간의 문자열 표현이어야 합니다.
- 빼기 연산자의 어느 피연산자도 매개변수 표시문자가 될 수 없습니다.

날짜 산술

날짜는 빼거나, 증가 또는 감소할 수 있습니다.

- 한 날짜(DATE1)에서 다른 날짜(DATE2)를 뺀 결과는 두 날짜 간의 연도, 월, 일 수를 지정하는 날짜 기간입니다. 결과의 데이터 유형은 DECIMAL(8,0)입니다. DATE1이 DATE2와 같거나 큰 경우, DATE1에서 DATE2를 뺍니다. 그러나 DATE1이 DATE2보다 작은 경우, DATE2에서 DATE1을 빼고 결과의 부호는 음수가 됩니다. 다음 프로시저는 연산 결과 = DATE1 - DATE2와 관련된 단계를 나타냅니다.

```
   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. 즉, 0년, 2개월, 15일의 지속 기간입니다.

- 지속 기간을 날짜에 더하거나, 날짜에서 지속 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 0001년 1월 1일과 9999년 12월 31일 사이에 있어야 합니다.

  연도의 지속 기간을 더하거나 빼는 경우, 날짜의 연도 부분에만 영향을 줍니다. 결과가 윤년이 아닌 해의 2월 29일 경우를 제외하면, 월은 변경되지 않습니다. 이런 경우, 일은 28로 변경되고, SQLCA의 경고 표시기가 조정을 나타내도록 표시됩니다.

  마찬가지로, 월의 지속 기간을 더하거나 빼는 경우, 월과 연도(필요한 경우)에만 영향을 줍니다. 날짜의 일 부분은 결과가 유효하지 않으면 (예: 9월 31일) 변경되지 않습니다. 이런 경우, 일은 월의 마지막 일로 설정되고, SQLCA의 경고 표시기가 조정을 나타내도록 설정됩니다.

물론 지속 기간을 더하거나 빼면 날짜의 일 부분에 영향을 주고, 잠재적으로 월과 연도에도 영향을 줍니다.

음수나 양수에 관계없이 날짜 기간을 날짜에 더하거나 뺄 수 있습니다. 레이블된 지속 기간과 같이 결과는 유효한 날짜이고, 경고 표시기는 월 종료의 조정이 필요할 때마다 SQLCA에 설정됩니다.

양수의 날짜 기간을 날짜에 더하거나 음수의 날짜 기간을 날짜에서 빼면, 날짜는 연도, 월, 일의 지정된 숫자만큼 순서대로 증가합니다. 그러므로 DATE1 + X(여기서, X는 양의 DECIMAL(8,0) 숫자임)는 다음 표현식과 같은 값입니다.

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.
```

양수의 날짜 기간을 날짜에서 빼거나, 음수의 날짜 기간을 날짜에 더하면 날짜는 연도, 월, 일의 지정된 숫자만큼 순서대로 감소합니다. 그러므로 DATE1 - X(여기서, X는 양의 DECIMAL(8,0) 숫자임)는 다음 표현식과 같은 값입니다.

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.
```

날짜에 기간을 추가할 때, 주어진 날짜에 한 달을 더하면 그 날짜가 다음 달에 없는 경우를 제외하고 1개월 이후의 같은 날짜로 됩니다. 해당 날짜가 다음 달에 없는 경우, 날짜는 다음 달의 말일이 됩니다. 예를 들어, 1월 28일에 1개월을 더하면 2월 28일이 되고, 1월 29, 30, 31일에 1개월을 더하면 모두 2월 28일이 되고, 윤년의 경우는 2월 29일이 됩니다.

주: 주어진 날짜에 한 달 이상의 월을 더한 다음 월과 같은 수를 결과에서 빼는 경우, 마지막 날짜는 원래의 날짜와 같지 않습니다.

시간 산술

시간은 빼거나, 증가 또는 감소할 수 있습니다.

- 한 시간(TIME1)에서 다른 시간(TIME2)을 뺀 결과는 두 시간 간의 시, 분, 초를 지정하는 시간 지속 기간입니다. 결과의 데이터 유형은 DECIMAL(6,0)입니다.

  TIME1이 TIME2보다 크거나 같은 경우, TIME1에서 TIME2를 뺍니다.

  그러나 TIME1이 TIME2보다 작은 경우, TIME2에서 TIME1을 빼고 결과의 부호는 음수가 됩니다. 다음 프로시저는 연산 결과 = TIME1 - TIME2와 관련된 단계를 나타냅니다.

```
   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. 즉, 10시간, 29분, 30초의 지속 기간입니다.

- 지속 기간을 시간에 더하거나 시간 기간에서 뺀 결과는 그 자체가 시간입니다. 시간의 오버플로우와 언더플로우는 무시되므로, 결과는 항상 시간입니다. 시간의 지속 기간을 더하거나 빼는 경우, 시간의 시 부분에만 영향을 줍니다. 분과 초는 변경되지 않습니다.

  마찬가지로, 분의 지속 기간을 더하거나 빼는 경우, 분과 시(필요한 경우)에만 영향을 줍니다. 시간의 초 부분은 변경되지 않습니다.

  물론 초의 지속 기간을 더하거나 빼면 시간의 초 부분에 영향을 주고, 잠재적으로 분과 시에도 영향을 줍니다.

  음수나 양수에 관계없이 시간 지속 기간을 시간에 더하고 시간에서 뺄 수 있습니다. 결과는 지정된 시, 분, 초 수만큼씩 이 순서대로 증가하거나 감소한 시간입니다. TIME1 + X(여기서 "X"는 DECIMAL(6,0) 숫자)는 다음 표현식에 해당합니다.

  ```
  TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
  ```

  초의 분수를 포함하는 값을 갖는 SECOND 또는 SECONDS의 레이블된 지속 기간을 뺄 때, 빼기는 시간 값이 최고 12개의 분수 초 숫자를 갖는 것처럼 수행되지만 결과는 분수 초가 절단되어 리턴됩니다.

  주: 시간 '24:00:00'이 유효한 시간으로 승인되어도, 이 시간은 지속 기간 피연산자가 0(예: 시간('24:00:00')±0 초 = '00:00:00')일 경우라도 시간 더하기 및 빼기 결과로 리턴되지 않습니다.

시간소인 산술

시간소인은 빼거나, 증가 또는 감소할 수 있습니다.

- 한 시간소인(TS1)에서 다른 시간소인(TS2)을 뺀 결과는 두 시간소인 간의 연도, 월, 일, 시, 분, 초 및 소수 초를 나타내는 시간소인 지속 기간입니다. 결과의 데이터 유형은 DECIMAL(14+*s*,*s*)입니다. 여기서 *s*는 TS1 및 TS2의 최대 시간소인 정밀도입니다.

  TS1이 TS2보다 크거나 같은 경우, TS1에서 TS2를 뺍니다. 그러나 TS1이 TS2보다 작은 경우, TS2에서 TS1을 빼고, 결과의 부호는 음수가 됩니다. 다음 프로시저는 연산 결과 = TS1 - TS2와 관련된 단계를 나타냅니다.

```
If SECOND(TS2,s) <= SECOND(TS1,s)
then SECOND(RESULT,s) = SECOND(TS1,s) -
SECOND(TS2,s).

If SECOND(TS2,s) > SECOND(TS1,s)
then SECOND(RESULT,s) = 60 +
SECOND(TS1,s) – SECOND(TS2,s).
MINUTE(TS2) is then incremented by 1.
```

시간소인의 분 부분을 시간 빼기 규칙에 지정된 대로 뺍니다.

```
   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.
```

시간소인의 날짜 부분은 날짜 빼기 규칙에 지정된 대로 뺍니다.

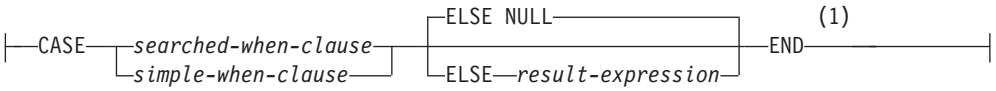
- 시간소인(TS1)에서 날짜(D1)를 뺀 결과는 TS1에서 TIMESTAMP(D1)를 뺀 것과 같습니다. 마찬가지로, 날짜(D2)에서 한 시간소인(TS1)을 뺀 결과는 TIMESTAMP(D2)에서 TS1을 뺀 것과 같습니다.
- 지속 기간을 시간소인에 더하거나 시간소인에서 지속 기간을 뺀 결과는 그 자체가 시간소인입니다.결과 시간소인의 정밀도는 시간소인 피연산자의 정밀도와 일치합니다. 날짜 산술 분할 영역은 이전에 정의된 것처럼 수행됩니다. 단, 시간의 오버플로우나 언더플로우는 결과의 날짜 부분에 영향을 주는데 이는 유효한 날짜 범위 내에 있어야 합니다. 시간 산술 분할 영역은 시간 산술과 유사하지만 지속기간에 포함된 분수 초도 고려합니다. 따라서, 지속기간 TIMESTAMP1 - *X* 빼기(여기서 *X*는 DECIMAL(14+s,s) 숫자)는 다음 표현식과 동등합니다.

```
TIMESTAMP1 - YEAR(X)  YEARS - MONTH(X)  MONTHS - DAY(X)  DAYS
           - HOUR(X)  HOURS - MINUTE(X)  MINUTES - SECOND(X, s)  SECONDS
```

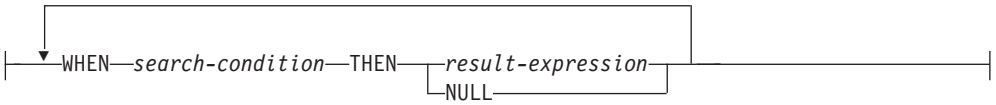
0이 아닌 스케일을 갖는 지속기간 또는 초의 분수를 포함하는 값을 갖는 SECOND 또는 SECONDS의 레이블된 지속 기간을 뺄 때, 빼기는 시간소인 값이 최고 12개의 분수 시간 숫자를 갖는 것처럼 수행됩니다. 결과 값은 분수 초 숫자의 절단이 발생할 수 있는 시간소인 피연산자의 시간소인 정밀도를 갖는 시간소인 값에 지정됩니다.

CASE 표현식

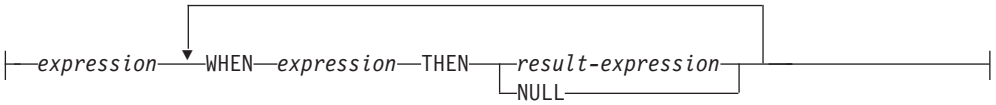
case-expression:



searched-when-clause:



simple-when-clause:



주:

1 *result-expression*의 결과 유형이 행 유형이면, 구문은 *row-case-expression*을 표시하며 *row-expression*이 허용되는 경우에만 사용할 수 있습니다.

CASE 표현식에서는 하나 이상의 조건을 평가하여 그것을 기준으로 표현식을 선택할 수 있습니다. 일반적으로, *case-expression*의 값은 참으로 평가된 첫 번째(가장 왼쪽) 경우의 뒤에 오는 *result-expression*의 값입니다. 참인 경우가 없고 ELSE 키워드가 있는 경우, 결과는 *result-expression*의 값이나 널(NULL)이 됩니다. 참인 경우가 없고 ELSE 키워드도 없는 경우, 결과는 널(NULL)이 됩니다. 널(NULL)로 인해 CASE가 알 수 없음으로 평가되면 그 경우는 참이 아니므로 거짓으로 평가되는 경우와 동일하게 처리됩니다.

CASE 표현식이 VALUES절, IN 술어, GROUP BY절 또는 ORDER BY절에 있는 경우, *searched-when-clause*에 있는 *search-condition*은 정량화 술어, fullselect를 사용하는 IN 술어 또는 EXISTS 술어가 될 수 없습니다(SQLSTATE 42625).

*simple-when-clause*를 사용할 때 첫 번째 *WHEN* 키워드 앞에 오는 *expression* 값은 WHEN 키워드 뒤에 오는 *expression* 값과 같은지 테스트됩니다. 첫 번째 *WHEN* 키워드 앞에 오는 *expression*의 데이터 유형은 WHEN 키워드가 뒤에 오는 각각의 *expression* 데이터 유형과 비교될 수 있어야 합니다. *simple-when-clause*에서 첫 번째 *WHEN* 키워드 앞에 오는 *expression*은 결정적이 아니거나 외부 조치가 있는 함수를 포함할 수 없습니다(SQLSTATE 42845).

*result-expression*은 THEN 또는 ELSE 키워드 뒤에 오는 *expression*입니다. CASE 표현식에는 적어도 하나의 *result-expression*이 있어야 합니다. 모든 경우에 NULL이 지정될 수 없습니다(SQLSTATE 42625). 모든 결과 표현식에는 호환 가능한 데이터 유형이 있어야 합니다(SQLSTATE 42804).

예:

- 부서 번호의 첫 번째 문자가 조직의 부서일 경우, CASE 표현식을 사용하여 각 직원이 속하는 부서의 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 연도의 수는 EMPLOYEE 테이블에서 교육 레벨을 결정하는 데 사용됩니다. CASE 표현식을 사용하여 이들을 그룹화하고, 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 수당으로 수입의 25퍼센트 이상을 벌지만 수당을 완전히 지불받지 않은 직원을 찾습니다.

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
  ELSE COMM/SALARY
  END) > 0.25;
```

- 다음 CASE 표현식은 같습니다.

```
SELECT LASTNAME,
  CASE
  WHEN LASTNAME = 'Haas' THEN 'President'
  ...

SELECT LASTNAME,
  CASE LASTNAME
  WHEN 'Haas' THEN 'President'
  ...
```

CASE에서 제공하는 기능의 서브세트를 처리하도록 지정된 두 개의 스칼라 함수 NULLIF와 COALESCE가 있습니다. 표 22은 CASE 또는 이러한 함수를 사용한 동일한 표현식을 나타냅니다.

표 22. 동일한 CASE 표현식

표현식	동일한 표현식

`CASE  WHEN e1=e2 THEN NULL`
`  ELSE e1`
`  END`
	NULLIF(e1,e2)

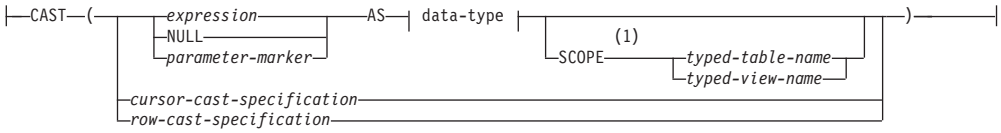
`CASE  WHEN e1 IS NOT NULL THEN e1`
`  ELSE e2`
`  END`
	COALESCE(e1,e2)

`CASE  WHEN e1 IS NOT NULL THEN e1`
`  ELSE COALESCE(e2,...,eN)`
`  END`
	COALESCE(e1,e2,...,eN)

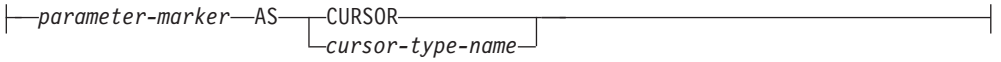
`CASE  WHEN c1=var1 OR (c1 IS NULL AND var1 IS NULL)`
`  THEN 'a'`
`  WHEN c1=var2 OR (c1 IS NULL AND var2 IS NULL)`
`  THEN 'b'`
`  ELSE NULL  END`
	DECODE(c1,var1, 'a', var2, 'b')

CAST 스펙

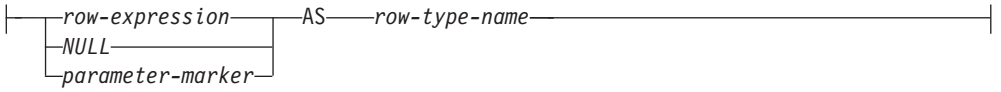
cast-specification:



cursor-cast-specification:

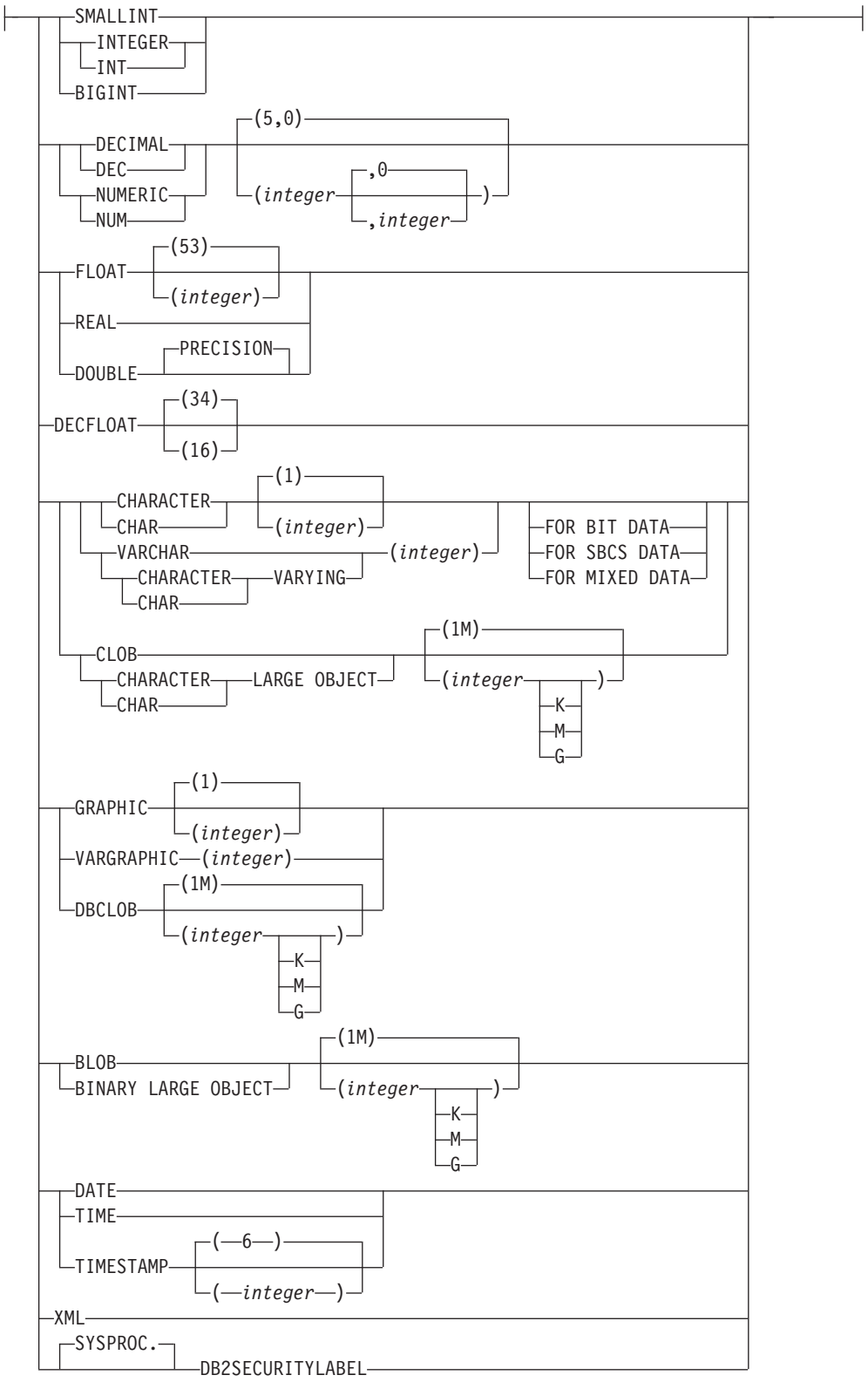


row-cast-specification:



data-type:

built-in-type:



주:

1 SCOPE절은 REF 데이터 유형에만 적용됩니다.

CAST 스펙은 캐스트 피연산자(첫 번째 피연산자) CAST를 *data-type*에 지정된 유형으로 리턴합니다. 캐스트가 지원되지 않는 경우, 오류가 리턴됩니다(SQLSTATE 42846).

expression

캐스트 피연산자가 표현식(매개변수 표시문자나 널(NULL)이 아님)인 경우, 결과는 지정된 목표 *data-type*으로 변환된 인수 값입니다.

문자열(CLOB은 제외)을 길이가 다른 문자열로 캐스팅할 때 후행 공백 이외의 절단이 발생하는 경우, 경고(SQLSTATE 01004)가 리턴됩니다. 그래픽 문자열(DBCLOB 제외)을 길이가 다른 그래픽 문자열로 캐스팅할 때 후행 공백 이외의 절단이 발생하는 경우, 경고(SQLSTATE 01004)가 리턴됩니다. BLOB, CLOB 및 DBCLOB 캐스트 피연산자의 경우, 문자가 절단되면 경고가 표시됩니다.

배열을 캐스팅하는 경우, 목표 데이터 유형은 사용자 정의 배열 데이터 유형이어야 합니다(SQLSTATE 42821). 배열 요소의 데이터 유형은 대상 배열 데이터 유형의 요소 데이터 유형과 같아야 합니다(SQLSTATE 42846). 배열의 카디널리티(cardinality)는 대상 배열 데이터 유형의 최대 카디널리티 이하여야 합니다(SQLSTATE 2202F).

NULL

캐스트 피연산자가 NULL 키워드이면 결과는 지정된 *data-type*을 갖는 널(NULL) 값입니다.

parameter-marker

매개변수 표시문자는 일반적으로 표현식으로 간주되지만 이 경우에는 특별한 의미를 가지므로 별도로 설명됩니다. 캐스트 피연산자가 *parameter-marker*일 경우, 지정된 *data-type*은 대체를 지정된 데이터 유형으로 지정할 수 있음(문자열에 대한 저장 지정을 통해)을 나타내는 보증으로 간주됩니다. 이러한 매개변수 표시문자는 *유형이 지정된 parameter marker*로 간주됩니다. 유형이 지정된 매개변수 표시문자는 함수 결정을 위해 유형이 지정된 다른 값, 선택 목록의 DESCRIBE 또는 컬럼 지정에 대한 것처럼 처리됩니다.

cursor-cast-specification

매개변수 표시문자가 커서 유형이 될 것으로 예상됨을 표시하기 위해 사용되는 캐스트 스펙. 커서 유형을 허용하는 컨텍스트에서 표현식이 지원될 때마다 사용할 수 있습니다.

parameter-marker

캐스트 피연산자는 매개변수 표시문자이며 지정된 커서 유형에 교체가 지정 가능하다는 보증으로 간주됩니다.

CURSOR

내장 데이터 유형 CURSOR를 지정합니다.

cursor-type-name

사용자 정의 커서 유형의 이름을 지정합니다.

row-cast-specification

입력이 행 값이고 결과가 사용자 정의 행 유형인 캐스트 스펙. *row-cast-specification*은 *row-expression*이 허용되는 경우에만 유효합니다.

row-expression

*row-expression*의 데이터 유형은 테이블 또는 보기의 정의에 고정된 행 유형의 변수여야 합니다. *row-expression*의 데이터 유형은 사용자 정의 행 유형이 될 수 없습니다(SQLSTATE 42846).

NULL

캐스트 피연산자가 널(NULL) 값임을 지정합니다. 결과는 지정된 데이터 유형의 모든 필드에 널(NULL) 값이 있는 행입니다.

parameter-marker

캐스트 피연산자는 매개변수 표시문자이며 지정된 *row-type-name*에 교체가 지정 가능하다는 보증으로 간주됩니다.

row-type-name

사용자 정의 행 유형의 이름을 지정합니다. *row-expression*은 *row-type-name* (SQLSTATE 42846)에 캐스트 가능해야 합니다.

data-type

기존 데이터 유형의 이름입니다. 유형 이름이 규정되지 않은 경우, 데이터 유형 분석을 수행하는 데 SQL 경로가 사용됩니다. 연관된 속성(예: 길이 또는 정밀도 및 스케일)을 가지고 있는 데이터 유형은 *data-type*을 지정할 때 이 속성을 포함해야 합니다. (지정되지 않은 경우 CHAR은 디폴트로 길이가 1로 설정되고 DECIMAL은 디폴트로 정밀도 5 및 스케일 0으로 설정되며, DECFLOAT는 디폴트로 정밀도 34가 설정됩니다.) FOR SBCS DATA 절 또는 FOR MIXED DATA 절(데이터베이스가 그래픽 데이터 유형을 지정하는지 여부에 따라 하나만 지원됨)을 사용하여 FOR BIT DATA 문자열을 데이터베이스 코드 페이지에 캐스트할 수 있습니다. 지원되는 데이터 유형에 대한 제한사항은 지정된 캐스트 피연산자에 따라 달라집니다.

- *expression*인 캐스트 피연산자의 경우, 지원되는 목표 데이터 유형은 캐스트 피연산자의 데이터 유형(소스 데이터 유형)에 따라 다릅니다.
- 캐스트 피연산자가 널(NULL) 키워드인 경우, 기존의 데이터 유형을 사용할 수 있습니다.
- 매개변수 표시문자인 캐스트 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 사용자 정의 구조화된 유형일 경우, 매개변수 표시문자를 사용하는 응용프로그램은 사용자 정의 구조화된 유형에 대한 TO SQL 변환 함수의 입력 매개변수 유형을 사용합니다.

SCOPE

데이터 유형이 참조 유형이면 참조의 목표 뷰 또는 목표 테이블을 식별하는 범위를 지정할 수 있습니다.

typed-table-name

유형이 지정된 테이블의 이름입니다. 이 테이블은 이미 존재해야 합니다 (SQLSTATE 42704). 캐스트는 *data-type* REF(*S*)로 되어야 합니다. 여기서, *S*는 *typed-table-name*의 유형입니다(SQLSTATE 428DM).

typed-view-name

유형이 지정된 뷰의 이름입니다. 뷰는 존재하거나, 작성하려는 뷰와 동일한 이름을 가져야 합니다. 이 뷰에는 뷰 정의 일부로서 캐스트가 포함되어 있습니다 (SQLSTATE 42704). 캐스트는 *data-type* REF(*S*)로 되어야 합니다. 여기서, *S*는 *typed-view-name*의 유형입니다(SQLSTATE 428DM).

숫자 데이터가 문자 데이터로 캐스트될 때 결과 데이터 유형은 고정 길이 문자열입니다. 문자 데이터가 숫자 데이터로 캐스트될 때 결과 데이터 유형은 지정된 숫자 유형에 따라 다릅니다. 예를 들어 정수로 캐스트되는 경우 큰 정수가 됩니다.

예:

- 응용프로그램이 EMPLOYEE 테이블에서 SALARY(decimal(9,2)로 정의됨)의 정수 부분에만 적용될 경우, 직원 번호와 정수 값 SALARY를 포함한 다음과 같은 쿼리가 준비됩니다.

```
SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

- SMALLINT로 정의되고 PERSONNEL 테이블에서 컬럼 AFE를 작성하는 데 사용되는 T_AGE 구별 유형이 존재한다고 가정하고 또한 INTEGER로 정의되고 PERSONNEL 테이블에서 컬럼 RETIRE_YEAR를 작성하는 데 사용되는 R_YEAR 구별 유형이 존재한다고 가정해 보십시오. 다음과 같은 갱신 명령문이 준비됩니다.

```
UPDATE PERSONNEL SET RETIRE_YEAR =?
  WHERE AGE = CAST( ? AS T_AGE)
```

첫 번째 매개변수는 응용프로그램에서 이 매개변수 표시문자에 정수를 사용할지라도 R_YEAR의 데이터 유형을 갖는 유형이 지정되지 않은 매개변수 표시문자입니다. 이것은 지정이므로 명시적인 CAST 스펙이 필요하지 않습니다.

두 번째 매개변수 표시문자는 구별 유형 T_AGE로 캐스트되고 유형이 지정된 매개변수 표시문자입니다. 이 매개변수 표시문자는 호환되는 데이터 유형에 대해 비교가 수행되어야 하는 요구사항을 충족시킵니다. 응용프로그램에서는 소스 데이터 유형(SMALLINT)을 사용하여 이 매개변수 표시문자를 처리하게 됩니다.

이 명령문을 성공적으로 처리하려면 SQL 경로에 두 개의 구별 유형이 정의된 스키마(들)의 스키마 이름이 포함되어야 합니다.

- 응용프로그램은 일련의 비트 값(예: 오디오 스트림)을 제공하므로 SQL문에 사용되기 전에 코드 페이지 변환이 실행되면 안됩니다. 응용프로그램은 다음 CAST를 사용할 수 있습니다.

  ```
  CAST( ? AS VARCHAR(10000) FOR BIT DATA)
  ```

- 배열 유형 및 테이블은 다음과 같이 작성되었다고 가정하십시오.

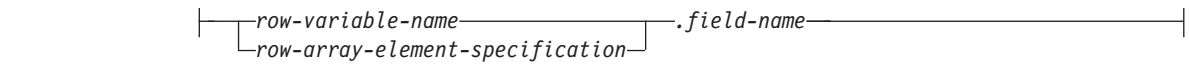
  ```
  CREATE TYPE PHONELIST AS DECIMAL(10, 0) ARRAY[5]

  CREATE TABLE EMP_PHONES
         (ID          INTEGER,
     PHONENUMBER DECIMAL(10,0) )
  ```

  다음 프로시저는 ID가 1775인 사원의 전화번호가 있는 배열을 리턴합니다. 이 사원에 대해 전화번호가 6개 이상인 경우 오류가 리턴됩니다(SQLSTATE 2202F).

  ```
  CREATE PROCEDURE GET_PHONES(OUT EPHONES PHONELIST)
  BEGIN
    SELECT CAST(ARRAY_AGG(PHONENUMBER) AS PHONELIST)
    INTO EPHONES
    FROM EMP_PHONES
    WHERE ID = 1775;
  END
  ```

필드 참조

field-reference:



행 유형의 필드는 다음과 같이 규정된 필드 이름을 사용하여 참조됩니다.

- 해당 필드 이름과 함께 필드를 포함하는 행 유형을 리턴하는 변수
- 해당 필드 이름과 함께 필드를 포함하는 행 유형을 리턴하는 배열 요소 스펙

row-variable-name

행 유형인 데이터 유형을 가지고 있는 변수의 이름입니다.

row-array-element-specification

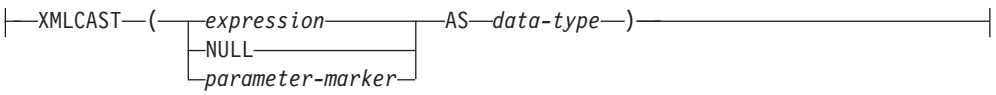
*array-element-specification*은 배열 요소의 데이터 유형이 행 유형입니다.

field-name

행 유형 내에서 필드의 이름입니다.

XMLCAST 스펙

xmlcast-specification:



XMLCAST 스펙은 캐스트 피연산자(첫 번째 피연산자) 캐스트를 데이터 유형에 지정된 유형으로 리턴합니다. XMLCAST는 비XML 데이터 유형과 XML 데이터 유형 간의 변환을 포함하여 XML 값을 포함하는 캐스트를 지원합니다. 캐스트가 지원되지 않는 경우 오류가 리턴됩니다(SQLSTATE 22003).

expression
캐스트 피연산자가 표현식(매개변수 표시문자나 널(NULL)이 아님)인 경우 결과는 지정된 목표 데이터 유형으로 변환된 인수 값입니다. 이 표현식 또는 목표 데이터 유형은 XML 데이터 유형이어야 합니다(SQLSTATE 42846).

NULL
캐스트 피연산자가 NULL 키워드인 경우 목표 데이터 유형은 XML 데이터 유형이어야 합니다(SQLSTATE 42846). 결과는 널(NULL) XML 값입니다.

parameter-marker
캐스트 피연산자가 매개변수 표시문자인 경우 목표 데이터 유형은 XML이어야 합니다(SQLSTATE 42846). 매개변수 표시문자는 일반적으로 표현식으로 간주되지만 이 경우에는 특별한 의미를 가지므로 별도로 설명됩니다. 캐스트 피연산자가 매개변수 표시문자인 경우 지정된 데이터 유형은 지정된 (XML) 데이터 유형으로 대체를 지정(저장 지정을 사용하여)할 수 있다는 것으로 간주됩니다. 이러한 매개변수 표시문자는 유형이 지정된 매개변수 표시문자로 간주되는데, 이는 함수 결정, 선택 목록의 설명 조작 또는 컬럼 지정을 위해 다른 유형이 지정된 값과 같이 처리됩니다.

data-type
기존 SQL 데이터 유형의 이름입니다. 이 이름이 규정되어 있지 않으면 SQL 경로를 사용하여 데이터 유형 분석을 수행합니다. 데이터 유형이 연관된 속성(예: 길이 또는 정밀도 및 스케일)을 가지고 있는 경우, *data-type*에 대한 값을 지정할 때 이들 속성을 포함해야 합니다. 지정되지 않은 경우 CHAR은 디폴트로 길이가 1로 설정되고 DECIMAL은 디폴트로 정밀도 5와 스케일 0으로 설정됩니다. 지원되는 데이터 유형에 대한 제한사항은 지정된 캐스트 피연산자에 따라 달라집니다.

- 캐스트 피연산자가 표현식인 경우, 지원되는 목표 데이터 유형은 캐스트 피연산자의 데이터 유형(소스 데이터 유형)에 따라 지정됩니다.
- 캐스트 피연산자가 NULL 키워드인 경우 목표 데이터 유형은 XML이어야 합니다.

- 캐스트 피연산자가 매개변수 표시문자인 경우 목표 데이터 유형은 XML이어야 합니다.

주: 유니코드가 아닌 데이터베이스에서의 지원: XMLCAST가 사용되어 XML 값이 SQL 데이터 유형으로 변환되는 경우 코드 페이지 변환이 수행됩니다. 캐스트 표현식의 인코딩은 UTF-8에서 데이터베이스 코드 페이지로 변환됩니다. 데이터베이스 코드 페이지에 존재하지 않는 원래 표현식의 문자는 이 변환의 결과로 대체 문자에 의해 교체됩니다.

예:

- 널(NULL) XML 값을 작성합니다.

```
XMLCAST(NULL AS XML)
```

- XMLQUERY 표현식에서 추출된 값을 INTEGER로 변환합니다.

```
XMLCAST(XMLQUERY('$m/PRODUCT/QUANTITY'
  PASSING BY REF xmlcol AS "m" RETURNING SEQUENCE) AS INTEGER)
```

- XMLQUERY 표현식에서 추출된 값을 가변 길이 문자열로 변환합니다.

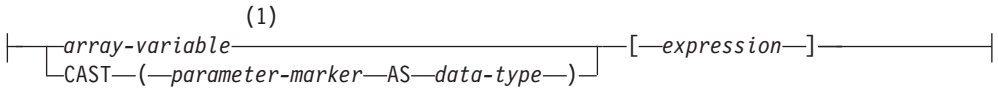
```
XMLCAST(XMLQUERY('$m/PRODUCT/ADD-TIMESTAMP'
  PASSING BY REF xmlcol AS "m" RETURNING SEQUENCE) AS VARCHAR(30))
```

- SQL 스칼라 서브쿼리에서 추출된 값을 XML 값으로 변환합니다.

```
XMLCAST((SELECT quantity FROM product AS p
  WHERE p.id = 1077) AS XML)
```

ARRAY 요소 스펙

array-element-specification:



주:

1 배열에서 요소의 데이터 유형이 행 유형인 경우, 구문은 행 데이터 유형과 함께 배열 요소 스펙을 표시하며 *row-expression*이 허용되는 경우에만 사용할 수 있습니다.

ARRAY 요소 스펙은 *expression*에 지정된 배열의 요소를 리턴합니다. *expression*의 임의 요소가 널(NULL)이면 널(NULL) 값이 리턴됩니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수입니다.

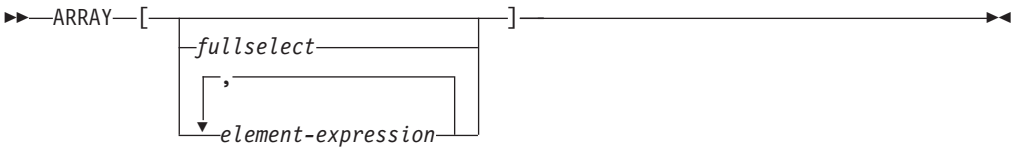
CAST (*parameter-marker* **AS** *data-type***)**
매개변수 표시문자는 일반적으로 표현식으로 간주되지만 이 경우에는 사용자 정의 배열 데이터 유형에 명시적으로 캐스트되어야 합니다.

[*expression***]**
배열에서 추출될 요소의 배열 인덱스를 지정합니다. 일반 배열의 배열 인덱스는 INTEGER에 지정 가능해야 합니다(SQLSTATE 428H1). 값은 1과 배열의 카디널리티(cardinality) 사이여야 합니다(SQLSTATE 2202E). 연관된 배열의 배열 인덱스는 인덱스 데이터 유형에 지정 가능해야 합니다(SQLSTATE 428H1).

배열 컨스트럭터

배열 컨스트럭터는 유효한 컨텍스트에 배열 데이터 유형 값을 정의 및 구성하는 데 사용할 수 있는 언어 요소입니다.

구문



권한 부여

SQL문 내에서 배열 컨스트럭터를 참조하기 위해 특정 권한이 필요한 것은 아니지만, 명령문 실행에 성공하려면 명령문에 대한 기타 모든 권한 요구사항을 충족해야 합니다.

설명

ARRAY[]

비어 있는 배열을 지정합니다.

ARRAY[*fullselect*]

요소가 한 개의 컬럼을 리턴하는 *fullselect*의 결과 행인 배열을 지정합니다.

*fullselect*에 *order-by-clause*절이 포함된 경우에는 순서는 행 값이 배열 요소에 지정되는 순서를 판별합니다. *order-by-clause*절이 지정하지 않으면 행 값이 배열 요소에 지정되는 순서를 판별할 수 없습니다.

ARRAY[*element-expression*,...]

각 요소에 대한 표현식 값으로 배열을 지정합니다. 첫 번째 *element-expression*은 배열 인덱스 1로 배열 요소에 지정됩니다. 두 번째 *element-expression*은 배열 인덱스 2로 배열 요소에 지정되는 방식입니다. 모든 *element-expression*에는 다른 모든 *element-expression*과 호환되는 데이터 유형이 있어야 하며 element-expression에서는 기본 배열 유형이 ″결과 데이터 유형에 대한 규칙″을 기반으로 판별됩니다.

규칙

- *element-expression* 또는 *fullselect*에서 파생된 *array-constructor*의 기본 유형은 목표 배열의 기본 유형에 지정할 수 있어야 합니다(SQLSTATE 42821).
- *array-constructor*의 요소 수는 목표 배열 변수의 최대 카디널리티(cardinality)를 초과하면 안됩니다(SQLSTATE 2202F).

주

- 배열 컨스트럭터를 사용하여 행 유형이 아닌 요소가 포함된 일반 배열만 정의할 수 있습니다. 배열 컨스트럭터를 사용하여 연관된 배열 또는 행 유형인 요소가 포함된 일반 배열을 정의할 수는 없습니다. 이런 배열은 각 요소를 개별적으로 지정해서만 구문으로 작성할 수 있습니다.

예

예 1: 배열 유형이 PHONENUMBERS인 배열 변수 RECENT_CALLS를 고정된 번호의 배열로 설정하십시오.

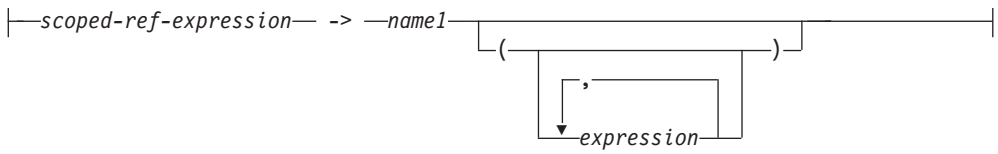
```
SET RECENT_CALLS = ARRAY[9055553907, 4165554213, 4085553678]
```

*예 2:*배열 유형이 PHONENUMBERS인 배열 변수 DEPT_PHONES를 DEPARTMENT_INFO 테이블에서 검색된 전화번호 배열로 설정하십시오.

```
SET DEPT_PHONES = ARRAY[SELECT DECIMAL(AREA_CODE CONCAT '555' CONCAT EXTENSION,16)
                                  FROM DEPARTMENT_INFO
                                  WHERE DEPTID = 624]
```

비참조 조작

dereference-operation:



범위 지정된 참조 표현식의 범위는 목표 테이블 또는 뷰라고 하는 테이블 또는 뷰입니다. 범위 지정된 참조 표현식은 목표 *행*을 식별합니다. 목표 *행*은 오브젝트 ID(OID) 컬럼 값이 참조 표현식과 일치하는 목표 테이블 또는 뷰의(또는 그의 서브테이블이나 서브뷰의) 행입니다. 비참조 조작을 사용하면 목표 행의 컬럼에 액세스하거나 목표 행을 메소드의 주제로 사용하여 메소드를 호출할 수 있습니다. 비참조 조작의 결과는 항상 널(Null)일 수 있습니다. 비참조 조작은 다른 모든 연산자에 대해 우선권을 갖습니다.

scoped-ref-expression

범위를 갖는 참조 유형인 표현식입니다(SQLSTATE 428DT). 표현식이 호스트 변수, 매개변수 표시문자 또는 기타 범위가 지정되지 않은 참조 유형 값인 경우, 참조에 범위를 부여하기 위해 SCOPE절을 갖는 CAST 스펙이 필요합니다.

name1

규정되지 않은 ID를 지정합니다.

name1 다음에 괄호가 없고 *name1*이 목표 유형의 속성 이름과 일치할 경우, 비참조 연산자의 값은 목표 행에서 이름이 지정된 컬럼의 값입니다. 이 경우, 널(NULL) 입력 가능 컬럼의 데이터 유형에 따라 비참조 연산자의 결과 유형이 결정됩니다. 오브젝트 ID가 참조 표현식과 일치하는 목표 행이 존재하지 않는 경우 비참조 조작의 결과는 널(Null)입니다. 비참조 조작이 선택 목록에서 사용되고 표현식의 파트로 포함되지 않는 경우 *name1*이 결과 컬럼 이름이 됩니다.

name1 뒤에 괄호가 있거나 *name1*이 목표 유형의 속성 이름과 일치하지 않는 경우 비참조 조작은 메소드 호출로 처리됩니다. 호출된 메소드의 이름은 *name1*입니다. 메소드의 주제는 목표 행이며 구조화된 유형의 인스턴스로 간주됩니다. 오브젝트 ID가 참조 표현식과 일치하는 목표 행이 없는 경우 메소드의 주제는 목표 유형의 널(NULL) 값입니다. 괄호 안에 있는 표현식이(있는 경우) 메소드 호출의 나머지 매개변수를 제공합니다. 메소드 호출의 분석을 위해 일반 프로세스가 사용됩니다. 널(NULL) 입력 가능 선택된 메소드의 결과 유형에 따라 비참조 조작의 결과 유형이 결정됩니다.

비참조 조작을 사용하는 명령문의 권한 부여 ID에는 *scoped-ref-expression*의 목표 테이블에 대한 SELECT 특권이 있어야 합니다(SQLSTATE 42501).

비참조 조작은 데이터베이스의 값을 수정할 수 없습니다. 비참조 조작이 mutator 메소드를 호출하는 데 사용되는 경우 mutator 메소드가 목표 행의 사본을 수정하고 사본을 리턴하며 데이터베이스는 변경되지 않은 채로 있습니다.

예:

- DEPTNAME 속성을 포함하는 유형을 기초로 유형이 지정된 테이블로 범위가 지정되는 참조 유형인 DEPTREF 컬럼을 포함하는 EMPLOYEE 테이블이 있다고 가정하십시오. EMPLOYEE 테이블의 DEPTREF 값은 DEPTREF 컬럼의 목표 테이블에 있는 OID 컬럼 값에 해당해야 합니다.

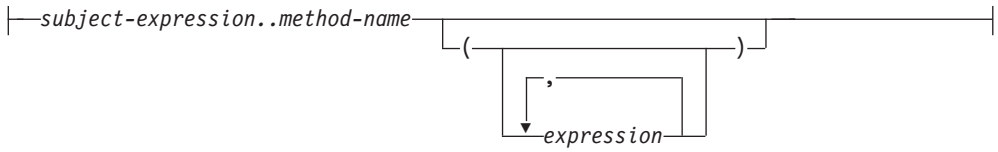
```
SELECT EMPNO, DEPTREF->DEPTNAME
  FROM EMPLOYEE
```

- 이전 예에서와 동일한 테이블을 사용하고, 비참조 조작을 사용하여 목표 행을 주제 매개변수로 사용하고 '1997'을 추가 매개변수로 사용하여 BUDGET 메소드를 호출하십시오.

```
SELECT EMPNO, DEPTREF->BUDGET('1997') AS DEPTBUDGET97
  FROM EMPLOYEE
```

메소드 호출

method-invocation:



사용자 정의 메소드 뿐 아니라 시스템 생성 관찰자 및 mutator 메소드 모두 이중 도트 연산자를 사용하여 호출됩니다.

subject-expression

사용자 정의 구조화된 유형인 정적 결과 유형을 갖는 표현식.

method-name

메소드의 규정되지 않은 이름입니다. *subject-expression* 또는 슈퍼 유형 중 하나의 정적 유형이 지정된 이름을 갖는 메소드를 포함해야 합니다.

(*expression***,...)**

*method-name*의 인수가 괄호 안에 지정됩니다. 비어 있는 괄호를 사용하여 인수가 없음을 표시할 수 있습니다. *method-name* 및 지정된 인수 표현식의 데이터 유형이 *subject-expression*의 정적 유형을 기초로 특정 메소드로 분석되는 데 사용됩니다.

메소드 호출에 사용하는 이중 도트 연산자는 높은 우선순위의 왼쪽에서 오른쪽으로 인픽스 연산자입니다. 예를 들어, 다음 두 표현식은 같습니다.

```
a..b..c + x..y..z
```

및

```
((a..b)..c) + ((x..y)..z)
```

메소드에 주제 이외의 매개변수가 없는 경우 괄호를 사용하거나 사용하지 않고 호출할 수 있습니다. 예를 들어, 다음 두 표현식은 같습니다.

```
point1..x
point1..x()
```

메소드 호출에서의 널(NULL) 주제는 다음과 같이 처리됩니다.

- 시스템 생성 mutator 메소드가 널(NULL) 주제와 함께 호출되는 경우 오류가 발생합니다(SQLSTATE 2202D).
- 시스템 생성 mutator 외의 메소드가 널(NULL) 주제와 함께 호출되는 경우 메소드는 실행되지 않고 결과는 널(Null)입니다. 이 규칙은 SELF AS RESULT를 갖는 사용자 정의 메소드를 포함합니다.

데이터베이스 오브젝트(예: 패키지, 뷰 또는 트리거)가 작성될 때 각 메소드 호출에 대해 존재하는 가장 잘 맞는 메소드가 발견됩니다.

주: WITH FUNCTION ACCESS로 정의된 유형의 메소드는 일반 함수 표기를 사용하여 호출할 수도 있습니다. 함수 결정은 함수 액세스를 갖는 메소드 뿐 아니라 모든 함수를 후보 함수로 고려합니다. 그러나 함수는 메소드 호출을 사용하여 호출할 수 없습니다. 메소드 분석은 모든 메소드를 고려하고 함수를 후보 메소드로 고려하지 않습니다. 적합한 함수 또는 메소드로 분석하지 못하면 오류가 발생합니다(SQLSTATE 42884).

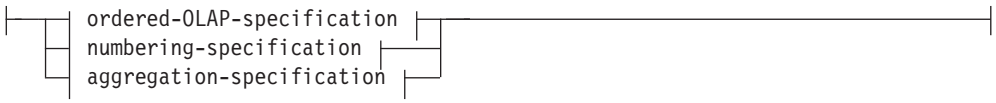
예 :

- 이중 도트 연산자를 사용하여 메소드 AREA를 호출하십시오. 구조화된 유형 CIRCLE의 CIRCLE_COL 컬럼을 갖는 RINGS 테이블이 존재한다고 가정하십시오. 또한 AREA 메소드가 이전에 CIRCLE 유형에 대해 AREA() RETURNS DOUBLE로 정의되었다고 가정하십시오.

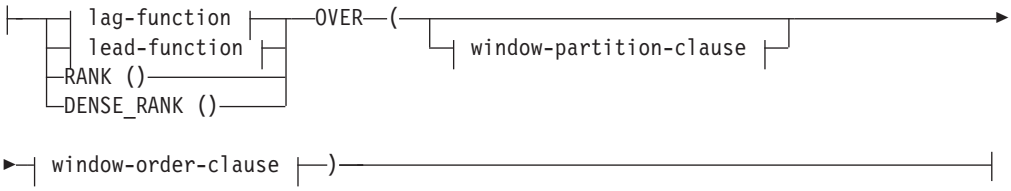
```
SELECT CIRCLE_COL..AREA() FROM RINGS
```

OLAP 스펙

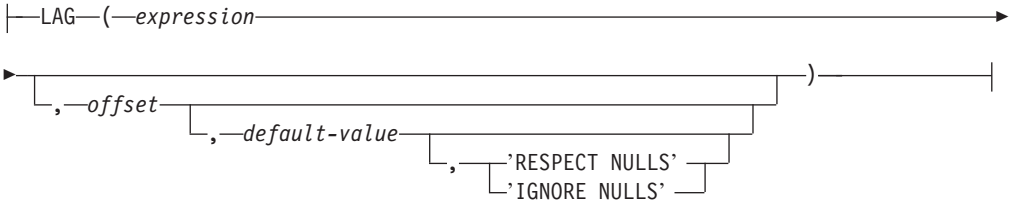
OLAP-specification:



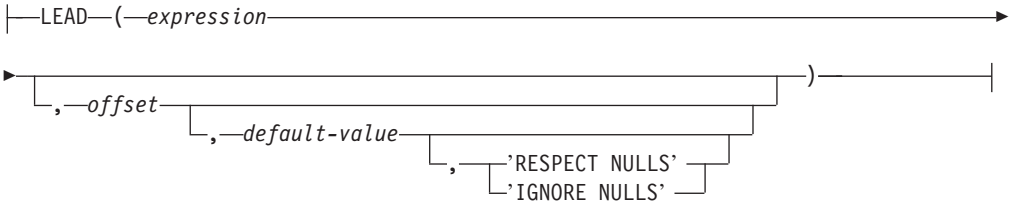
ordered-OLAP-specification:



lag-function:

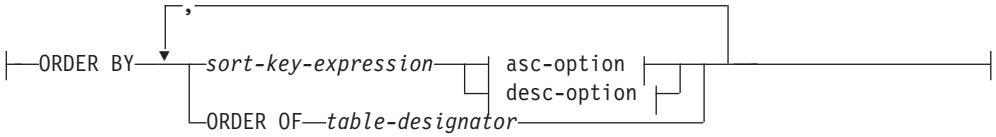


lead-function:



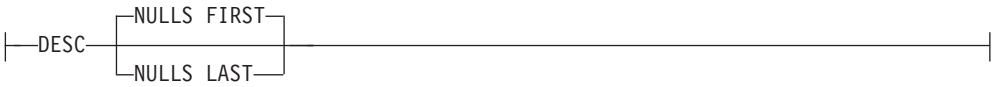
window-partition-clause:

window-order-clause:

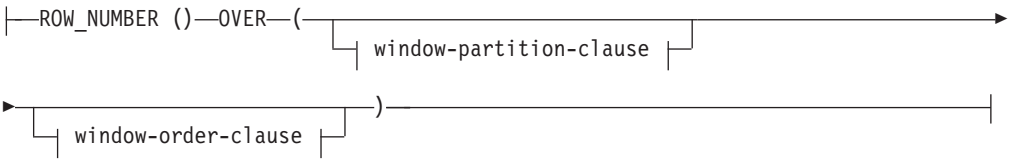


asc-option:

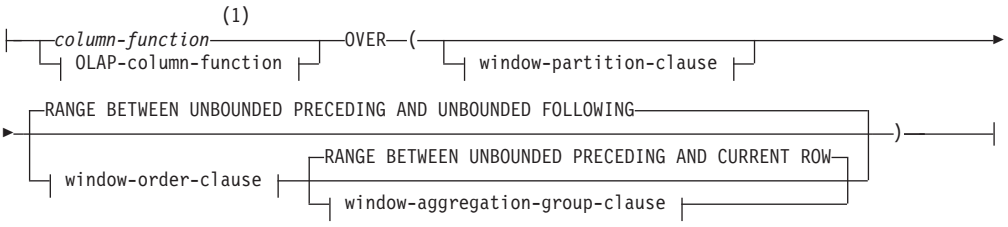
desc-option:



numbering-specification:



aggregation-specification:



OLAP-column-function:

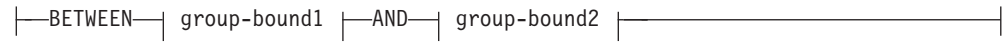
first-value-function:

last-value-function:

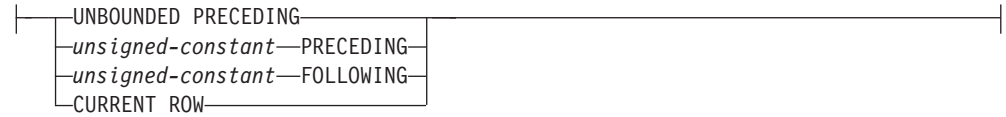
window-aggregation-group-clause:

group-start:

group-between:

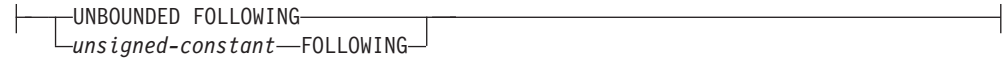


group-bound1:



group-bound2:

group-end:



주:

1 ARRAY_AGG는 *aggregation-specification*에서 집계 함수로 지원됩니다 (SQLSTATE 42887).

OLAP(On-Line Analytical Processing) 함수는 쿼리 결과로 순위 지정, 행 번호매김 및 기존의 집계 함수 정보를 스칼라 값으로 리턴할 수 있는 기능을 제공합니다. OLAP 함수는 선택 목록이나 Select문의 ORDER BY절의 표현식에 포함될 수 있습니다 (SQLSTATE 42903). OLAP 함수는 XMLQUERY 또는 XMLEXISTS 표현식에 대한 인수에서 사용될 수 없습니다(SQLSTATE 42903). OLAP 함수는 집계 함수의 인수로 사용할 수 없습니다(SQLSTATE 42607). OLAP 함수가 적용되는 쿼리 결과는 OLAP 함수를 포함하는 가장 안쪽의 subselect에 대한 결과 테이블입니다.

OLAP 함수를 지정할 때 함수가 적용되는 행과 그 순서를 정의하는 창이 지정됩니다. 집계 함수와 함께 사용할 경우, 적용 가능한 행은 현재 행 앞과 뒤에 있는 행의 범위나 행 수로서 현재 행에 상대적으로 더 세분화될 수 있습니다. 예를 들어, 월별 파티션 내에서 이전 3개월 동안의 평균을 계산할 수 있습니다.

순위 지정 함수는 창 내에서 행의 일반 순위를 계산합니다. 창 내에서의 순서 지정에 대해 구별되지 않는 행에는 같은 순위가 지정됩니다. 순위 지정의 결과는 중복 값을 초래하는 숫자에 간격을 두거나 두지 않고 정의할 수 있습니다.

RANK를 지정할 경우, 행의 순위는 행 앞에 있는 행 수에 1을 더해 정의됩니다. 그러므로 순서 지정에 대해 두 개 이상의 행이 구별되지 않는 경우, 시퀀스 순위 지정에서 하나 이상의 간격이 생기게 됩니다.

DENSE_RANK(또는 DENSERANK)를 지정할 경우, 행의 순위는 순서를 지정하는 행 앞에 있는 행 수에 1을 더해 정의됩니다. 그러므로 시퀀스 순위 지정에 간격이 없습니다.

ROW_NUMBER(또는 ROWNUMBER) 함수는 첫 번째 행을 1로 시작하여 순서 지정에서 정의되는 창 내의 행의 시퀀스 행 번호를 계산합니다. ORDER BY절이 창에 지정되지 않은 경우, 행 번호는 subselect에 의해 리턴되는 임의의 순서로 행에 지정됩니다(SELECT문의 ORDER BY절에 따르지 않음).

FETCH FIRST *n* ROWS ONLY절을 ROW_NUMBER 함수와 함께 사용할 경우 행 번호가 순서대로 표시되지 않을 수 있습니다. FETCH FIRST절은 결과 세트(ROW_NUMBER 지정문을 포함하는)가 생성된 후에 적용됩니다. 따라서 행 번호 순서가 결과 세트의 순서와 동일하지 않으면 일부 지정된 숫자가 시퀀스에서 누락될 수 있습니다.

RANK, DENSE_RANK 또는 ROW_NUMBER의 결과에 대한 데이터 유형은 BIGINT입니다. 결과는 널(NULL)이 될 수 없습니다.

LAG 함수는 현재 행 이전에 *offset* 행에서 행에 대한 표현식 값을 리턴합니다. *offset*은 양의 정수여야 합니다(SQLSTATE 42815). *offset* 값 0은 현재 행을 의미합니다. window-partition-clause를 지정한 경우 *offset*은 현재 행 이전, 현재 파티션 내의 *offset* 행을 의미합니다. *offset*이 지정되지 않으면 값 1이 사용됩니다. *default-value*(표현식일 수 있음)를 지정한 경우, 오프셋이 현재 파티션 범위를 넘어가는 경우 리턴됩니다. 그렇지 않으면, 널(NULL) 값이 리턴됩니다. 'IGNORE NULLS'를 지정한 경우 행의 표현식 값이 널(NULL) 값인 모든 행이 계산에서 고려되지 않습니다. 'IGNORE NULLS'를 지정하고 모든 행이 널(NULL)인 경우 *default-value*(또는 *default-value*를 지정하지 않은 경우 널(NULL) 값)가 리턴됩니다.

LEAD 함수는 현재 행 이후에 *offset* 행에서 행에 대한 표현식 값을 리턴합니다. *offset*은 양의 정수여야 합니다(SQLSTATE 42815). *offset* 값 0은 현재 행을 의미합니다.

window-partition-clause를 지정한 경우 *offset*은 현재 행 이후, 현재 파티션 내의 *offset* 행을 의미합니다. *offset*이 지정되지 않으면 값 1이 사용됩니다. *default-value*(표현식일 수 있음)를 지정한 경우, 오프셋이 현재 파티션 범위를 넘어가는 경우 리턴됩니다. 그렇지 않으면, 널(NULL) 값이 리턴됩니다. 'IGNORE NULLS'를 지정한 경우 행의 표현식 값이 널(NULL) 값인 모든 행이 계산에서 고려되지 않습니다. 'IGNORE NULLS'를 지정하고 모든 행이 널(NULL)인 경우 *default-value*(또는 *default-value*를 지정하지 않은 경우 널(NULL) 값)가 리턴됩니다.

FIRST_VALUE 함수는 OLAP 창에서 첫 번째 행에 대한 표현식 값을 리턴합니다. 'IGNORE NULLS'를 지정한 경우 행의 표현식 값이 널(NULL) 값인 모든 행이 계산에서 고려되지 않습니다. 'IGNORE NULLS'를 지정하고 OLAP 창의 모든 값이 널(NULL)인 경우 FIRST_VALUE는 널(NULL) 값을 리턴합니다.

LAST_VALUE 함수는 OLAP 창에서 마지막 행에 대한 표현식 값을 리턴합니다. 'IGNORE NULLS'를 지정한 경우 행의 표현식 값이 널(NULL) 값인 모든 행이 계산에서 고려되지 않습니다. 'IGNORE NULLS'를 지정하고 OLAP 창의 모든 값이 널(NULL)인 경우 LAST_VALUE는 널(NULL) 값을 리턴합니다.

FIRST_VALUE, LAG, LAST_VALUE 및 LEAD 결과의 데이터 유형은 expression의 데이터 유형입니다. 결과는 널(NULL)이 될 수 있습니다.

PARTITION BY (*partitioning-expression***,...)**

함수가 적용되는 파티션을 정의합니다. *partitioning-expression*은 결과 세트의 파티션을 정의하는 데 사용되는 표현식입니다. *partitioning-expression*에서 참조되는 각 *column-name*은 OLAP 스펙을 포함하는 subselect의 결과 테이블 컬럼을 확실하게 참조합니다. *partitioning-expression*은 스칼라 fullselect나 XMLQUERY나 XMLEXISTS 표현식(SQLSTATE 42822) 또는 결정적이지 않거나 외부 조치가 있는 함수 또는 쿼리(SQLSTATE 42845)를 포함할 수 없습니다.

window-order-clause

ORDER BY (*sort-key-expression***,...)**

OLAP 함수의 값이나 window-aggregation-group-clause에서 ROW 값의 의미를 판별하는 행 순서를 파티션 내에서 정의합니다. 쿼리 결과 세트의 순서는 정의하지 않습니다.

sort-key-expression

창 파티션 내에서 행 순서를 정의하는 데 사용되는 표현식입니다. *sort-key-expression*에 참조된 각 컬럼 이름은 OLAP 함수를 포함하여 subselect의 결과 세트 컬럼을 확실하게 참조해야 합니다(SQLSTATE 42702 또는 42703). *sort-key-expression*은 스칼라 fullselect 또는 XMLQUERY나 XMLEXISTS 표현식(SQLSTATE 42822) 또는 결정적이지 않거나 외부 조치가 있는 함수 또는 쿼리(SQLSTATE 42845)를 포함할 수 없습니다. 이 절은 RANK 및 DENSE_RANK 함수에 필요합니다(SQLSTATE 42601).

ASC

sort-key-expression의 값을 오름차순으로 사용합니다.

DESC

sort-key-expression의 값을 내림차순으로 사용합니다.

NULLS FIRST

창 순서는 정렬 순서에서 모든 널(NULL)이 아닌 값보다 *먼저* 널(NULL) 값을 고려합니다.

NULLS LAST

창 순서는 정렬 순서에서 널(NULL)이 아닌 모든 값 *이후에* 널(NULL) 값을 고려합니다.

ORDER OF *table-designator*

*table-designator*에 사용된 동일한 순서가 subselect의 결과 테이블에 적용되도록 지정합니다. 이 절을 지정하는 *table-designator*와 일치하는 테이블 참조가 subselect의 FROM절에 있어야 합니다(SQLSTATE 42703). 지정된 *table-designator*에 해당하는 subselect(또는 fullselect)에는 데이터에 따라 달라지는 ORDER BY절이 포함되어야 합니다(SQLSTATE 428FI). 적용되는 순서는 중첩된 subselect(또는 fullselect)의 ORDER BY절의 컬럼이 외부 subselect(또는 fullselect)에 포함되고 이러한 컬럼이 ORDER OF절 대신에 지정된 경우와 같습니다.

window-aggregation-group-clause

행 R의 집계 그룹은 (행 R의 파티션의 순서 지정에서) R에 관해 정의된 행 세트입니다. 이 절은 집계 그룹을 지정합니다. 이 절이 지정되지 않고 window-order-clause도 지정하지 않으면 집계 그룹은 창 파티션의 모든 행으로 구성됩니다. 이 디폴트는 RANGE(표시된 대로) 또는 ROWS를 사용하여 명시적으로 지정할 수 있습니다.

window-order-clause가 지정된 경우 window-aggregation-group-clause가 지정되어 있지 않을 때 디폴트 작동은 차이가 있습니다. 창 집계 그룹은 R의 앞에 있는 R 파티션의 모든 행 또는 widows-order-clause에서 정의된 창 파티션의 창 순서화에서 R의 피어로 구성됩니다.

ROWS

집계 그룹이 계산 행에서 정의됨을 나타냅니다.

RANGE

집계 그룹이 정렬 키의 오프셋에 의해 정의됨을 나타냅니다.

group-start

집계 그룹에 대한 시작점을 지정합니다. 집계 그룹의 끝은 현재 행입니다. group-start절의 스펙은 ″BETWEEN group-start AND CURRENT ROW″ 형식의 group-between절과 같습니다.

group-between

ROWS 또는 RANGE를 기반으로 집계 그룹의 시작 및 끝을 지정합니다.

group-end

집계 그룹에 대한 종료점을 지정합니다. 집계 그룹의 시작은 현재 행입니다. group-end절의 스펙은 ″BETWEEN CURRENT ROW AND group-end″ 형식의 group-between절과 같습니다.

UNBOUNDED PRECEDING

현재 행 앞에 있는 전체 파티션을 포함합니다. 이것은 ROWS 또는 RANGE를 사용하여 지정할 수 있습니다. 또한 이것은 window-order-clause에서 다중 sort-key-expressions를 사용하여 지정할 수 있습니다.

UNBOUNDED FOLLOWING

현재 행 다음에 있는 전체 파티션을 포함합니다. 이것은 ROWS 또는 RANGE를 사용하여 지정할 수 있습니다. 또한 이것은 window-order-clause에서 다중 sort-key-expressions를 사용하여 지정할 수 있습니다.

CURRENT ROW

현재 행을 기반으로 집계 그룹의 시작과 끝을 지정합니다. ROWS를 지정하는 경우, 현재 행은 집계 그룹 바운더리입니다. RANGE를 지정하는 경우, 집계 그룹 바운더리는 현재 행과 동일한 *sort-key-expressions* 값의 행 세트를 포함합니다. 이 절은 *group-bound1*이 *value* FOLLOWING을 지정하는 경우에는 *group-bound2*에 지정될 수 없습니다.

value **PRECEDING**

현재 행 앞에 있는 행 범위나 행 수를 지정합니다. ROWS를 지정하는 경우, *value*는 행 수를 나타내는 양의 정수입니다. RANGE를 지정하는 경우, *value*의 데이터 유형은 window-order-clause의 sort-key-expression 유형과 비교할 수 있어야 합니다. 하나의 sort-key-expression이 있을 수 있으며 정렬 키 표현식의 데이터 유형은 빼기를 허용해야 합니다. 이 절은 *group-bound1*이 CURRENT ROW이거나 *value* FOLLOWING일 경우에는 *group-bound2*에 지정될 수 없습니다.

value **FOLLOWING**

현재 행 다음에 있는 행 범위나 행 수를 지정합니다. ROWS를 지정하는 경우, *value*는 행 수를 나타내는 양의 정수입니다. RANGE를 지정하는 경우, *value*의 데이터 유형은 window-order-clause의 sort-key-expression 유형과 비교할 수 있어야 합니다. 하나의 sort-key-expression이 있을 수 있으며 sort-key-expression의 데이터 유형은 더하기를 허용해야 합니다.

예:

- 총 급여가 $30,000을 초과하는 직원들의 순위를 총 급여에 따라(급여에 보너스를 더한 값을 기준으로) 성 순서로 표시하십시오.

```
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
```

순위에 따라 결과 순서가 지정될 경우, ORDER BY LASTNAME을 다음 두 가지 중 하나로 대체하십시오.

```
ORDER BY RANK_SALARY
```

또는

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- 평균 총 급여에 따라 부서 순위를 지정하십시오.

```
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 직원들의 순위를 지정하십시오. 부서에서 여러 직원들의 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
```

- 쿼리 결과에 행 번호를 제공하십시오.

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,
          LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- 급여 순위 상위 5명의 직원을 나열하십시오.

```
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
```

WHERE절에 순위가 사용되기 전에 중첩 테이블 표현식이 결과(순위 포함)를 처음 계산하는 데 사용되었다는 점에 유의하십시오. 공통 테이블 표현식도 사용될 수 있었습니다.

- 부서마다 사원 급료를 나열하고 각각의 개인을 해당 부서에서 다음으로 높은 급료를 가지고 있는 사원과 비교하여 어느 정도 낮은지 표시하십시오.

```
SELECT EMPNO, WORKDEPT, LASTNAME, FIRSTNME, JOB, SALARY,
    LEAD(SALARY, 1) OVER (PARTITION BY WORKDEPT
      ORDER BY SALARY) - SALARY AS DELTA_SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, SALARY
```

- 동일 유형의 작업에 대해 처음 고용된 사원의 급료에 상대적으로 사원 급료를 계산하십시오.

```
SELECT JOB, HIREDATE, EMPNO, LASTNAME, FIRSTNME, SALARY,
    FIRST_VALUE(SALARY) OVER (PARTITION BY JOB
      ORDER BY HIREDATE) AS FIRST_SALARY,
    SALARY - FIRST_VALUE(SALARY) OVER (PARTITION BY JOB
      ORDER BY HIREDATE) AS DELTA_SALARY
  FROM EMPLOYEE
  ORDER BY JOB, HIREDATE
```

- 2006년 1월 동안 재고 XYZ에 대한 평균 마감 가격을 계산하십시오. 재고가 지정된 날에 거래되지 않으면 DAILYSTOCKDATA 테이블에서 해당 마감 가격은 널(NULL) 값이 됩니다. 재고가 거래되지 않은 날에 대해 널(NULL) 값을 리턴하는 대신, COALESCE 함수와 LAG 함수를 사용하여 재고가 거래된 최근 날의 마감 가격을 리턴하십시오. 이전의 널이 아닌 마감 값 검색을 2006년 1월 1일 이전 한달로 제한하십시오.

```
WITH V1(SYMBOL, TRADINGDATE, CLOSEPRICE) AS
(
SELECT SYMBOL, TRADINGDATE,
  COALESCE(CLOSEPRICE,
    LAG(CLOSEPRICE,
             1,
      CAST(NULL AS DECIMAL(8,2)),
      'IGNORE NULLS')
    OVER (PARTITION BY SYMBOL
    ORDER BY TRADINGDATE)
)
FROM DAILYSTOCKDATA
WHERE SYMBOL = 'XYZ' AND
  TRADINGDATE BETWEEN '2005-12-01' AND '2006-01-31'
)
SELECT SYMBOL, AVG(CLOSEPRICE) AS AVG
  FROM V1
  WHERE TRADINGDATE BETWEEN '2006-01-01' AND '2006-01-31'
  GROUP BY SYMBOL
```

- 2005년 동안 재고 ABC 및 XYZ의 30일 이동 평균을 계산하십시오.

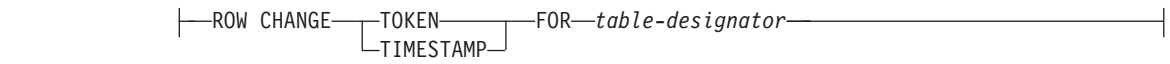
```
WITH V1(SYMBOL, TRADINGDATE, MOVINGAVG30DAY) AS
(
SELECT SYMBOL, TRADINGDATE,
  AVG(CLOSEPRICE) OVER (PARTITION BY SYMBOL
    ORDER BY TRADINGDATE
  ROWS BETWEEN 29 PRECEDING AND CURRENT ROW)
FROM DAILYSTOCKDATA
WHERE SYMBOL IN ('ABC', 'XYZ')
  AND TRADINGDATE BETWEEN DATE('2005-01-01') - 2 MONTHS
    AND '2005-12-31'
)
SELECT SYMBOL, TRADINGDATE, MOVINGAVG30DAY
  FROM V1
  WHERE TRADINGDATE BETWEEN '2005-01-01' AND '2005-12-31'
  ORDER BY SYMBOL, TRADINGDATE
```

- 표현식을 사용하여 커서 위치를 정의하고 해당 위치 이전 50개 행의 슬라이딩 창을 쿼리하십시오.

```
SELECT DATE, FIRST_VALUE(CLOSEPRICE + 100) OVER
    (PARTITION BY SYMBOL
    ORDER BY DATE
    ROWS BETWEEN 50 PRECEDING AND 1 PRECEDING) AS FV
  FROM DAILYSTOCKDATA
  ORDER BY DATE
```

ROW CHANGE 표현식

row-change-expression:



ROW CHANGE 표현식은 행에 대한 마지막 변경을 나타내는 시간소인이나 토큰을 리턴합니다.

TOKEN

행의 수정 시퀀스에서 상대적 위치를 나타내는 BIGINT 값이 리턴됨을 지정합니다. 행이 변경되지 않은 경우 결과는 초기값이 삽입된 시기를 나타내는 토큰입니다. 결과는 널(NULL)이 될 수 있습니다. ROW CHANGE TOKEN은 결정적이지 않습니다.

TIMESTAMP

행이 변경된 마지막 시간을 나타내는 TIMESTAMP 값이 리턴됨을 지정합니다. 행이 변경되지 않은 경우 결과는 초기값이 삽입된 시간입니다. 결과는 널(NULL)이 될 수 있습니다. ROW CHANGE TIMESTAMP는 결정적이지 않습니다.

FOR *table-designator*

표현식이 참조된 테이블을 식별합니다. *table-designator*는 기본 테이블, 뷰 또는 중첩 테이블 표현식을 고유하게 식별해야 합니다(SQLSTATE 42867). *table-designator*가 뷰 또는 중첩 테이블 표현식을 식별하는 경우 ROW CHANGE 표현식은 뷰 또는 중첩 테이블 표현식의 기본 테이블 TIMESTAMP 또는 TOKEN을 리턴합니다. 뷰 또는 중첩 테이블 표현식은 외부 subselect에 단 하나의 기본 테이블만 포함해야 합니다(SQLSTATE 42867). *table-designator*가 뷰 또는 중첩된 테이블 표현식인 경우 삭제할 수 있어야 합니다(SQLSTATE 42703). 삭제 가능한 뷰에 대한 정보는 『CREATE VIEW』의 『주의사항』 섹션을 참조하십시오. ROW CHANGE TIMESTAMP 표현식의 테이블 지정자는 행 변경 시간소인 컬럼을 포함하는 기본 테이블로 분석되어야 합니다(SQLSTATE 55068).

주

- ROW CHANGE TOKEN 표현식이 리턴한 값은 낙관적 잠금을 사용하는 응용프로그램이 RID_BIT 스칼라 함수와 함께 사용할 수 있습니다.

예:

- 부서 20의 사원에 대한 EMPLOYEE 테이블에서 각 행에 대한 최신 변경에 해당되는 시간소인 값을 리턴하십시오. ROW CHANGE TIMESTAMP 절에서 정의된 컬럼을 포함하도록 EMPLOYEE 테이블이 변경되었다고 가정하십시오.

```
SELECT ROW CHANGE TIMESTAMP FOR EMPLOYEE
  FROM EMPLOYEE WHERE DEPTNO = 20
```

- 사원 번호 3500에 해당되는 행의 수정 시퀀스에서 상대적 위치를 나타내는 BIGINT 값을 리턴하십시오. 또한 낙관적 잠금 DELETE 시나리오에서 사용할 RID_BIT 스칼라 함수 값도 리턴하십시오. WITH UR 옵션을 지정하여 최신 ROW CHANGE TOKEN 값을 가져오십시오.

```
SELECT ROW CHANGE TOKEN FOR EMPLOYEE, RID_BIT (EMPLOYEE)
  FROM EMPLOYEE WHERE EMPNO = '3500' WITH UR
```

위의 명령문은 EMPLOYEE 테이블에 행 변경 시간소인 컬럼이 있는지 여부에 관계없이 성공합니다. 다음의 검색된 DELETE문은 위의 SELECT문에서 ROW CHANGE TOKEN 및 RID_BIT 값에 지정된 행을 삭제합니다. 이때 두 개의 매개변수 표시문자 값이 위의 명령문에서 얻은 값으로 설정된다고 가정합니다.

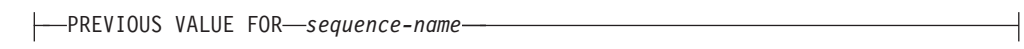
```
DELETE FROM EMPLOYEE E
  WHERE RID_BIT (E) = ? AND ROW CHANGE TOKEN FOR E = ?
```

시퀀스 참조

sequence-reference:

nextval-expression:

prevval-expression:



NEXT VALUE FOR *sequence-name*

NEXT VALUE 표현식은 *sequence-name*으로 지정되는 시퀀스에 대한 다음 값을 생성하고 리턴합니다.

PREVIOUS VALUE FOR *sequence-name*

PREVIOUS VALUE 표현식은 현재 응용프로그램 프로세스의 이전 명령문에 대해 지정된 시퀀스에 대한 가장 최근에 생성된 값을 리턴합니다. 이 값은 시퀀스 이름을 지정하는 PREVIOUS VALUE 표현식을 사용하여 반복적으로 참조할 수 있습니다. 단일 명령문 안에서 동일한 시퀀스 이름을 지정하는 PREVIOUS VALUE 표현식의 다중 인스턴스가 있을 수 있는데, 모두 동일한 값을 리턴합니다. 파티션된 데이터베이스 환경에서 PREVIOUS VALUE 표현식은 가장 최근에 생성된 값을 리턴할 수 없습니다.

PREVIOUS VALUE 표현식은 동일한 시퀀스 이름을 지정하는 NEXT VALUE 표현식이 현재 또는 이전 트랜잭션 중 하나에서 이미 현재 응용프로그램 프로세스에서 참조된 경우에만 사용할 수 있습니다(SQLSTATE 51035).

주

- NEXT VALUE 표현식이 시퀀스의 이름을 지정할 때 해당 시퀀스에 대해 새 값이 생성됩니다. 그러나 쿼리 안에서 동일한 시퀀스 이름을 지정하는 NEXT VALUE 표현식의 다중 인스턴스가 있는 경우, 시퀀스 카운터는 결과의 각 행에 대해 한 번만 증분되고 NEXT VALUE의 모든 인스턴스가 결과 행에 대해 동일한 값을 리턴합니다.
- 아래 표시된 것처럼, 첫 번째 행에 대해 NEXT VALUE 표현식으로 시퀀스 번호를 지정하고(시퀀스 값을 생성함) 기타 행에 대해 PREVIOUS VALUE 표현식으로 시퀀스 번호를 참조하여(PREVIOUS VALUE의 인스턴스는 현재 세션에서 기장 최근에 생성된 시퀀스 값을 참조함) 두 개의 개별 테이블에서 동일한 시퀀스 번호를 고유 키 값으로 사용할 수 있습니다.

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- NEXT VALUE 및 PREVIOUS VALUE 표현식은 다음 위치에 지정할 수 있습니다.
  - Select문 또는 SELECT INTO문(select절 내에서 명령문이 DISTINCT 키워드, GROUPBY절, ORDER BY절, UNION 키워드, INTERSECT 키워드 또는 EXCEPT 키워드를 포함하지 않는 경우)
  - INSERT문(VALUES절 내)
  - INSERT문(fullselect의 select-clause 내)
  - UPDATE문(SET절 내(검색 또는 위치 지정된 UPDATE문), NEXT VALUE는 SET절의 표현식의 fullselect의 select-clause에 지정할 수 없음)
  - SET 변수 명령문(표현식 fullselect의 select-clause 내부는 제외. NEXT VALUE 표현식을 트리거에 지정할 수 있지만 PREVIOUS VALUE 표현식은 지정할 수 없음)
  - VALUES INTO문(표현식 fullselect의 select-clause 내)
  - CREATE PROCEDURE문(SQL 프로시저의 routine-body 내)
  - 트리거된 조치의 CREATE TRIGGER문(NEXT VALUE 표현식은 지정할 수 있지만 PREVIOUS VALUE 표현식은 지정할 수 없음)
- 다음 위치에는 NEXT VALUE 및 PREVIOUS VALUE 표현식을 지정할 수 없습니다(SQLSTATE 428F9).
  - 완전 외부 조인의 조인 조건
  - CREATE 또는 ALTER TABLE문의 컬럼에 대한 DEFAULT 값
  - CREATE 또는 ALTER TABLE문의 생성된 컬럼 정의
  - CREATE TABLE 또는 ALTER TABLE문의 요약 테이블 정의
  - CHECK 제한조건의 조건
  - CREATE TRIGGER문(NEXT VALUE 표현식은 지정할 수 있지만 PREVIOUS VALUE 표현식은 지정할 수 없음)
  - CREATE VIEW문
  - CREATE METHOD문
  - CREATE FUNCTION문
  - XMLQUERY, XMLEXISTS 또는 XMLTABLE 표현식의 인수
- 또한 다음 위치에 NEXT VALUE 표현식을 지정할 수 없습니다(SQLSTATE 428F9).
  - CASE 표현식
  - 집계 함수의 매개변수 목록

  - 위에서 명시적으로 허용되는 것을 제외한 컨텍스트의 서브쿼리
  - 외부 SELECT가 DISTINCT 연산자를 포함하는 SELECT문
  - 조인의 조인 조건
  - 외부 SELECT가 GROUP BY절을 포함하는 SELECT문
  - 외부 SELECT가 UNION, INTERSECT 또는 EXCEPT 집합 연산자를 사용하여 다른 SELECT문과 결합되는 SELECT문
  - 중첩 테이블 표현식
  - 테이블 함수의 매개변수 목록
  - 가장 외부의 SELECT문, DELETE 또는 UPDATE문의 WHERE절
  - 가장 외부의 SELECT문의 ORDER BY절
  - UPDATE문의 SET절에서 표현식의 fullselect의 select-clause
  - SQL 루틴의 IF, WHILE, DO ... UNTIL 또는 CASE문
- 값이 시퀀스에 대해 생성되고, 해당 값이 이용되고, 다음에 값이 요청될 때 새 값이 생성됩니다. 이는 NEXT VALUE 표현식을 포함하는 명령문이 실패하거나 롤백될 때도 적용됩니다.

  INSERT문이 컬럼에 대한 VALUES 목록에 NEXT VALUE 표현식을 포함하는 경우 및 INSERT 실행 중에 어느 지점에서 오류가 발생하는 경우(다음 시퀀스 값 생성 문제 또는 또 다른 컬럼에 대한 값의 문제일 수 있음), 삽입 실패가 발생하며 (SQLSTATE 23505), 시퀀스에 대해 생성된 값은 이용된 것으로 간주됩니다. 일부 경우에는 동일한 INSERT문을 다시 발행하면 성공할 수도 있습니다.

  예를 들어 NEXT VALUE가 사용되었고 생성된 시퀀스 값이 이미 인덱스에 존재하는 컬럼에 대한 고유 인덱스의 존재 결과인 오류를 고려하십시오. 시퀀스에 대해 생성되는 다음 값이 인덱스에 존재하지 않으므로 후속 INSERT가 성공할 수 있습니다.
- ***PREVIOUS VALUE 범위:*** PREVIOUS VALUE는 현재 세션에서 시퀀스에 대한 다음 값이 생성되거나 시퀀스가 삭제 또는 변경 또는 응용프로그램 세션이 종료될 때까지는 유지됩니다. 값은 COMMIT 또는 ROLLBACK문에 영향을 받지 않습니다. PREVIOUS VALUE 값은 직접 설정할 수 없으며 시퀀스에 대해 NEXT VALUE 표현식을 실행하면 작성됩니다.

  흔히 사용되는 기술, 특히 성능은 연결 설정을 관리하고 임의 연결에 대한 트랜잭션을 라우트할 응용프로그램 또는 제품을 위한 것입니다. 이런 상황에서 PREVIOUS VALUE의 가용성은 트랜잭션 종료시까지만 신뢰할 수 있어야 합니다. 이런 상황이 발생할 수 있는 예에는 XA 프로토콜 사용, 연결 풀 사용, 연결 집중기(connection concentrator) 사용 및 장애 복구를 위해 HADR을 사용하는 응용프로그램이 포함됩니다.

- 시퀀스에 대한 값을 생성할 때 시퀀스의 최대값(또는 내림차순 시퀀스의 경우 최소값)이 초과되었고 순환이 허용되지 않는 경우 오류가 발생합니다(SQLSTATE 23522). 이 경우에 사용자는 시퀀스를 ALTER(변경)하여 승인할 수 있는 값의 범위를 확장하거나 시퀀스의 순환을 사용 가능하게 하거나, DROP(삭제)하고 더 큰 값의 범위를 갖는 다른 데이터 유형을 갖는 새 시퀀스를 CREATE(작성)할 수 있습니다.

  예를 들어, 데이터 유형이 SMALLINT인 시퀀스를 정의했을 수 있으며 결과적으로 해당 시퀀스는 지정 가능한 값을 모두 소비합니다. 시퀀스를 삭제(Drop)하고 새 정의를 갖는 시퀀스를 다시 작성하여 시퀀스를 INTEGER로 재정의하십시오.
- 커서의 선택 명령문에서 NEXT VALUE 표현식에 대한 참조는 결과 테이블의 행에 대해 생성되는 값을 참조합니다. 데이터베이스에서 페치되는 각 행에 대한 NEXT VALUE 표현식에 대해 시퀀스 값 하나가 생성됩니다. 클라이언트에서 블로킹이 수행되는 경우 값은 FETCH문의 처리 전에 서버에 생성되었을 수 있습니다. 결과 테이블의 행 블로킹이 있을 때 발생할 수 있습니다. 클라이언트 응용프로그램이 데이터베이스가 구체화한 모든 행을 명시적으로 FETCH(페치)하지 않는 경우 응용프로그램은(리턴되지 않은 구체화된 행에 대한) 모든 생성된 시퀀스 값의 결과를 확인하지 않습니다.
- 커서의 선택 명령문에서 PREVIOUS VALUE 표현식에 대한 참조는 커서를 열기 전에 지정된 시퀀스에 대해 생성된 값을 참조합니다. 그러나 커서를 닫으면 후속 명령문에서 지정된 시퀀스 또는 커서가 다시 열리는 이벤트에서 동일한 명령문에 대한 PREVIOUS VALUE로 리턴되는 값에 영향을 미칠 수 있습니다. 커서의 선택 명령문이 동일한 시퀀스 이름에 대한 NEXT VALUE 참조를 포함한 경우입니다.
- *호환성*
  - 이전 버전 DB2와의 호환성을 위해,
    - NEXTVAL 및 PREVVAL을 NEXT VALUE 및 PREVIOUS VALUE 대신 지정할 수 있습니다.
  - IBM IDS와의 호환성을 위해,
    - *sequence-name*.NEXTVAL을 NEXT VALUE FOR *sequence-name* 대신 지정할 수 있습니다.
    - *sequence-name*.CURRVAL을 PREVIOUS VALUE FOR *sequence-name* 대신 지정할 수 있습니다.

예:

다음과 같이 ″order″라는 테이블이 있고, ″order_seq″라는 시퀀스가 작성된다고 가정하십시오.

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE     CACHE 24
```

다음은 NEXT VALUE 표현식으로 ″order_seq″ 시퀀스 번호를 생성하는 방법의 몇 가지 예입니다.

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

또는

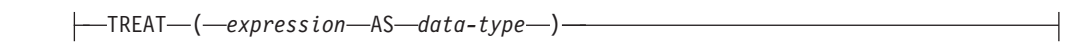
```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

또는

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

부속 유형 처리

subtype-treatment:



*subtype-treatment*는 구조화된 유형 표현식을 그의 부속 유형 중 하나로 캐스트하는 데 사용됩니다. *expression*의 정적 유형은 사용자 정의 구조화된 유형이어야 하며, 해당 유형은 *data-type*의 슈퍼 유형이거나 그와 동일한 유형이어야 합니다. *data-type*의 유형 이름이 규정되지 않은 경우 SQL 경로가 유형 참조를 해석하는 데 사용됩니다. subtype-treatment의 결과에 대한 정적 유형은 *data-type*이고, subtype-treatment의 값은 표현식의 값입니다. 런타임 시 표현식의 동적 유형이 *data-type*이 아니거나 *data-type*의 부속 유형이 아닌 경우, 오류가 리턴됩니다(SQLSTATE 0D000).

예 :

- 응용프로그램이 CIRCLE_COL 컬럼의 모든 컬럼 오브젝트 인스턴스에 동적 유형 COLOREDCIRCLE이 있음을 아는 경우 다음 쿼리를 사용하여 해당 오브젝트에서 RGB 메소드를 호출하십시오. 구조화된 유형 CIRCLE의 CIRCLE_COL 컬럼을 갖는 RINGS 테이블이 존재한다고 가정하십시오. 또한 COLOREDCIRCLE이 CIRCLE의 부속 유형이고 RGB 메소드가 이전에 COLOREDCIRCLE에 대해 RGB() RETURNS DOUBLE로 정의되었다고 가정하십시오.

  ```
  SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
    FROM RINGS
  ```

  런타임 시, 동적 유형 CIRCLE의 인스턴스가 있는 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
  ```

유형이 지정되지 않은 표현식의 데이터 유형 판별

유형이 지정되지 않은 표현식은 연관된 목표 데이터 유형 없이 지정된 널(NULL) 값 또는 매개변수 표시문자의 사용을 가리킵니다.

다음 조건 중 하나가 충족되면 SQL문에서 유형이 지정되지 않은 표현식을 사용할 수 있습니다.

- PREPARE문은 SQL문을 컴파일하기 위해 실행됩니다. 클라이언트 인터페이스는 지연된 준비를 사용하고 레지스트리 변수 DB2_DEFERRED_PREPARE_SEMANTICS는 YES로 설정됩니다. 이 경우, 유형이 지정되지 않은 매개변수 표시문자는 연속 OPEN 또는 EXECUTE문과 연관된 입력 디스크립터를 기초로 데이터 유형을 파생시킵니다. 길이 속성은 『함수』의 216 페이지의 표 20에서 설명하는 대로 UNTYPED 행에 따른 최대 길이로, 아래 테이블에서 정의된 길이로 설정됩니다. 『함수』의 216 페이지의 표 20에 목표 유형으로 나열되지 않는 데이터 유형의 경우, 연속 OPEN 또는 EXECUTE문과 연관된 입력 디스크립터의 길이가 사용됩니다. 데이터 유형 및 길이는 SQL문에서 유형이 지정되지 않은 매개변수 표시문자 사용에 따라서 수정될 수 있습니다.
- 데이터 유형은 SQL문의 컨텍스트를 기초로 판별할 수 있습니다. 위치 및 결과 데이터 유형은 다음 테이블에 표시됩니다. 위치는 표현식, 술어, 내장 함수 및 사용자 정의 루틴으로 그룹화되어 유형이 지정되지 않은 표현식의 적용 가능성을 판별하는 데 도움이 됩니다. 데이터 유형을 컨텍스트를 기초로 판별할 수 없는 경우, 오류가 발행됩니다.

나열되지 않은 어떤 경우에는, 선택 목록의 유형이 지정되지 않은 표현식이 SQL문에서의 사용을 기초로 판별된 데이터 유형으로 해석됩니다.

유형이 지정되지 않은 표현식의 코드 페이지는 컨텍스트로 판별됩니다. 컨텍스트가 없는 경우, 코드 페이지는 유형이 지정되지 않은 표현식이 VARCHAR 데이터 유형으로 캐스트된 경우와 동일합니다.

표 23. 표현식(선택 목록, CASE 및 VALUES 포함)에서 유형이 지정되지 않은 표현식 사용

유형이 지정되지 않은 표현식 위치	데이터 유형
선택 목록의 단독 표현식.	유형이 지정되지 않은 표현식이 이름 지정되지 않거나 이름이 지정되었지만 그 뒤로 SQL문에서 참조되지 않은 경우 오류가 발생합니다. 유형이 지정되지 않은 표현식이 이름 지정되고 그 뒤로 SQL문에서 참조된 경우 데이터 유형은 연속 사용을 통해 판별될 수 있습니다. 자세한 정보는 이 테이블 다음에 있는 "사용을 통해 데이터 유형 판별" 메모를 참조하십시오.

표 23. 표현식(선택 목록, CASE 및 VALUES 포함)에서 유형이 지정되지 않은 표현식 사용 (계속)

유형이 지정되지 않은 표현식 위치	데이터 유형
연산자 우선순위 및 조작 규칙 순서를 고려한 후 단일 산술 연산자의 모든 피연산자 다음과 같은 경우가 포함됩니다. `(? + ?) + 10`	DECFLOAT(34)
산술식(날짜/시간 표현식이 아닌)에서 단일 연산자의 한 피연산자 다음과 같은 경우가 포함됩니다. `? + (? * 10)`	다른 피연산자의 데이터 유형
날짜 및 시간 표현식 내의 레이블된 지속 기간(단위 유형을 나타내는 레이블된 지속 기간의 일부는 매개변수 표시문자가 될 수 없음)	DECIMAL(15,0)
날짜 시간 표현식의 다른 피연산자(예: 'timecol + ?' 또는 '? - datecol')	오류
CONCAT 연산자의 두 피연산자	VARCHAR(254)
다른 피연산자가 비CLOB 문자 데이터 유형인 경우 CONCAT 연산자의 피연산자	한 피연산자가 CHAR(*n*) 또는 VARCHAR(*n*)인 경우(여기서 *n*은 128 미만임) 다른 피연산자는 VARCHAR(254 - *n*)이고, 그 외의 모든 경우 데이터 유형은 VARCHAR(254)입니다.
다른 피연산자가 비DBCLOB 그래픽 데이터 유형인 경우 CONCAT 연산자의 피연산자	한 피연산자가 GRAPHIC(*n*) 또는 VARGRAPHIC(*n*)인 경우(여기서 *n*은 64 미만임), 다른 피연산자는 VARCHAR(127 - *n*)이고, 그 외의 모든 경우 데이터 유형은 VARCHAR(127)입니다.
다른 피연산자가 대형 오브젝트 문자열인 경우 CONCAT 연산자의 피연산자	다른 피연산자의 데이터 유형과 동일
단순 CASE 표현식에서 CASE 키워드 다음에 오는 표현식	유형이 지정되지 않은 표현식이 아닌, WHEN 키워드 다음에 오는 표현식에 『결과 데이터 유형 규칙』을 적용한 결과
CASE 표현식(단순 및 검색)에 있는 하나 이상의 결과 표현식(나머지 결과 표현식은 유형이 지정되지 않은 표현식임)	오류
단순 CASE 표현식에서 WHEN 키워드 다음에 오는 임의 또는 모든 표현식	CASE 다음에 오는 표현식과 유형이 지정되지 않은 표현식이 아닌 WHEN 키워드 다음에 오는 표현식에 『결과 데이터 유형 규칙』을 적용한 결과
최소 하나의 결과 표현식이 유형이 지정되지 않은 표현식이 아닌 경우 CASE 표현식(단순 및 검색)의 결과 표현식	유형이 지정되지 않은 표현식이 아닌 모든 결과 표현식에 『결과 데이터 유형 규칙』을 적용한 결과

표 23. 표현식(선택 목록, CASE 및 VALUES 포함)에서 유형이 지정되지 않은 표현식 사용 (계속)

유형이 지정되지 않은 표현식 위치	데이터 유형
INSERT문 내에 없고 MERGE문의 삽입 조작에서 VALUES절 내에 없는 단일 행 VALUES절의 계산 결과 컬럼으로서의 단독 표현식	유형이 지정되지 않은 표현식이 이름 지정되지 않거나 이름이 지정되었지만 그 뒤로 SQL문에서 참조되지 않은 경우 오류가 발생합니다. 유형이 지정되지 않은 표현식이 이름 지정되고 그 뒤로 SQL문에서 참조된 경우 데이터 유형은 연속 사용을 통해 판별될 수 있습니다. 자세한 정보는 이 테이블 다음에 있는 ″사용을 통해 데이터 유형 판별″ 메모를 참조하십시오.
INSERT문 내에 없고, 다른 모든 행 표현식의 동일한 위치에 있는 계산 결과 컬럼이 유형이 지정되지 않은 표현식인 복수 행 VALUES절의 계산 결과 컬럼으로서의 단독 표현식	유형이 지정되지 않은 표현식이 이름 지정되지 않거나 이름이 지정되었지만 그 뒤로 SQL문에서 참조되지 않은 경우 오류가 발생합니다. 유형이 지정되지 않은 표현식이 이름 지정되고 그 뒤로 SQL문에서 참조된 경우 데이터 유형은 연속 사용을 통해 판별될 수 있습니다. 자세한 정보는 이 테이블 다음에 있는 ″사용을 통해 데이터 유형 판별″ 메모를 참조하십시오.
INSERT문 내에 있지 않고, 최소 하나의 다른 행 표현식의 동일한 위치에 있는 표현식이 유형이 지정되지 않은 표현식인 복수 행 VALUES절의 계산 결과 컬럼으로서의 단독 표현식	유형이 지정되지 않은 표현식이 아닌 모든 피연산자에 『결과 데이터 유형 규칙』을 적용한 결과
INSERT문 내의 단일 행 VALUES절에 있는 단일 계산 결과 컬럼	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우 사용자 정의 구별 유형의 소스 데이터 유형이 됩니다. 컬럼이 사용자 정의된 구조화 유형으로 정의된 경우, 이 컬럼은 구조화된 유형이며 변환 함수의 리턴 유형을 나타냅니다.
INSERT문 내의 복수 행 VALUES절에 있는 단일 컬럼 표현식	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우 사용자 정의 구별 유형의 소스 데이터 유형이 됩니다. 컬럼이 사용자 정의된 구조화 유형으로 정의된 경우, 이 컬럼은 구조화된 유형이며 변환 함수의 리턴 유형을 나타냅니다.
MERGE문에 대한 소스 테이블 VALUES절의 계산 결과 컬럼으로서의 단독 표현식	유형이 지정되지 않은 표현식이 이름 지정되지 않거나 이름이 지정되었지만 그 뒤로 SQL문에서 참조되지 않은 경우 오류가 발생합니다. 유형이 지정되지 않은 표현식이 이름 지정되고 그 뒤로 SQL문에서 참조된 경우 데이터 유형은 연속 사용을 통해 판별될 수 있습니다. 자세한 정보는 이 테이블 다음에 있는 ″사용을 통해 데이터 유형 판별″ 메모를 참조하십시오.
MERGE문의 삽입 조작 VALUES절의 계산 결과 컬럼으로서의 단독 표현식	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우 사용자 정의 구별 유형의 소스 데이터 유형이 됩니다. 컬럼이 사용자 정의된 구조화 유형으로 정의된 경우, 이 컬럼은 구조화된 유형이며 변환 함수의 리턴 유형을 나타내기도 합니다.

표 23. 표현식(선택 목록, CASE 및 VALUES 포함)에서 유형이 지정되지 않은 표현식 사용 (계속)

유형이 지정되지 않은 표현식 위치	데이터 유형
MERGE문의 갱신 조작에 대한 assignment-clause의 오른쪽에 있는 계산 결과 컬럼으로서의 단독 표현식	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우 사용자 정의 구별 유형의 소스 데이터 유형이 됩니다. 컬럼이 사용자 정의된 구조화 유형으로 정의된 경우, 이 컬럼은 구조화된 유형이며 변환 함수의 리턴 유형을 나타내기도 합니다.
UPDATE문에 있는 SET절의 오른쪽에 있는 계산 결과 컬럼으로서의 단독 표현식	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우 사용자 정의 구별 유형의 소스 데이터 유형이 됩니다. 컬럼이 사용자 정의된 구조화 유형으로 정의된 경우, 이 컬럼은 구조화된 유형이며 변환 함수의 리턴 유형을 나타내기도 합니다.
SET 특수 레지스터 명령문의 오른쪽에 있는 값	특수 레지스터의 데이터 유형
테이블 참조 tablesample-clause의 TABLESAMPLE절 인수	DOUBLE
테이블 참조 tablesample-clause의 REPEATABLE 하위절 인수	INTEGER
FREE LOCATOR문에서 값으로	로케이터
SET ENCRYPTION PASSWORD문에서 암호에 대한 값으로	VARCHAR(128)

주:

사용을 통한 데이터 유형 판별

다음은 연속 사용을 통해 유형이 지정되지 않은 표현식의 데이터 유형을 판별할 수 있는 방법에 대한 예입니다.

이름 지정된 유형이 지정되지 않은 표현식이 그 뒤로 비교 연산자에 참조되는 경우, 다른 피연산자의 데이터 유형을 갖게 됩니다. SQL문에 이름 지정된 유형이 지정되지 않은 표현식 참조가 여러 개 있는 경우, 해당 참조 각각에 대해 독립적으로 판별되는 데이터 유형, 길이, 정밀도, 스케일 및 코드 페이지가 동일해야 합니다. 그렇지 않으면 오류가 리턴됩니다.

표 24. 술어에서 유형이 지정되지 않은 표현식 사용

유형이 지정되지 않은 표현식 위치	데이터 유형
비교 연산자의 두 피연산자	VARCHAR(254)
다른 피연산자가 유형이 지정되지 않은 표현식이 아닌 경우 비교 연산자의 한 피연산자	다른 피연산자의 데이터 유형
BETWEEN 술어의 모든 피연산자	VARCHAR(254)
BETWEEEN 술어의 두 피연산자	유형이 지정된 표현식과 같음
BETWEEN 술어의 단 하나의 피연산자	유형이 지정되지 않은 표현식이 아닌 모든 피연산자에 『결과 데이터 유형 규칙』을 적용한 결과
IN 술어의 모든 피연산자(예: ? IN (?,?,?)	VARCHAR(254)

표 24. 술어에서 유형이 지정되지 않은 표현식 사용 (계속)

유형이 지정되지 않은 표현식 위치	데이터 유형
오른쪽이 fullselect인 경우 IN 술어의 첫 번째 피연산자(예: IN (fullselect))	선택된 컬럼의 데이터 유형
오른쪽이 subselect가 아닌 경우 IN 술어의 첫 번째 피연산자(예: ? IN (?,A,B), 또는 ? IN (A,?,B,?)	유형이 지정되지 않은 표현식이 아닌 IN 목록의 모든 피연산자에 대해 ″결과 데이터 유형 규칙″을 적용한 결과
IN 술어의 IN 목록에 있는 임의 또는 모든 피연산자(예: A IN (?,B, ?))	유형이 지정되지 않은 표현식이 아닌 IN 술어의 모든 피연산자(IN 키워드의 왼쪽 및 오른쪽에 있는 피연산자)에 대해 ″결과 데이터 유형 규칙″을 적용한 결과
IN 술어의 row-value-expression의 피연산자와, fullselect의 해당되는 결과 컬럼 둘 다(예: (c1, ?) IN (SELECT c1, ? FROM ...))	VARCHAR(254)
IN 술어의 row-value-expression에 있는 모든 피연산자(예:(c1,?)), IN fullselect	fullselect의 해당 결과 컬럼 데이터 유형
row-value-expression이 IN 술어(예:(c1,c2) IN (SELECT?, c1, FROM ...))에 지정된 경우 서브쿼리에 있는 모든 선택 목록 항목	row-value-expression에 있는 해당 피연산자의 데이터 유형
LIKE 술어의 세 피연산자 모두	일치 표현식(피연산자 1) 및 패턴 표현식(피연산자 2)은 VARCHAR(32672)이고 이탈 표현식(피연산자 3)은 VARCHAR(2)임
패턴 표현식 또는 이스케이프 표현식이 유형이 지정되지 않은 표현식이 아닌 경우 LIKE 술어의 일치 표현식	유형이 지정되지 않은 표현식이 아닌 첫 번째 피연산자의 데이터 유형에 따라 VARCHAR(32672) 또는 VARGRAPHIC(16336)
일치 표현식 또는 이스케이프 표현식이 유형이 지정되지 않은 표현식이 아닌 경우 LIKE 술어의 패턴 표현식	유형이 지정되지 않은 표현식이 아닌 첫 번째 피연산자의 데이터 유형에 따라 VARCHAR(32672) 또는 VARGRAPHIC(16336)이고, 일치 표현식의 데이터 유형이 BLOB인 경우 패턴 표현식의 데이터 유형은 BLOB(32672)인 것으로 가정됨
일치 표현식 또는 패턴 표현식이 유형이 지정되지 않은 표현식이 아닌 경우 LIKE 술어의 이스케이프 표현식	유형이 지정되지 않은 표현식이 아닌 첫 번째 피연산자의 데이터 유형에 따라 VARCHAR(2) 또는 VARGRAPHIC(1)이고, 일치 표현식 또는 패턴 표현식의 데이터 유형이 BLOB인 경우 이스케이프 표현식의 데이터 유형은 BLOB(1)인 것으로 가정됨
널(NULL) 술어의 피연산자	VARCHAR(254)

표 25. 내장 함수에서 유형이 지정되지 않은 표현식 사용

미입력 매개변수 표시문자 위치	데이터 유형
COALESCE, MIN, MAX, NULLIF 또는 VALUE의 모든 인수	오류
하나 이상의 인수가 유형이 지정되지 않은 매개변수 표시문자인 경우 COALESCE, MIN, MAX, NULLIF 또는 VALUE의 임의 인수	유형이 지정되지 않은 매개변수 표시문자가 아닌 모든 인수에 대해 『결과 데이터 유형 규칙』을 적용한 결과
DIGITS의 인수	DECIMAL(31,6)
POSSTR(두 인수 모두)	두 인수 모두 VARCHAR(32672)임

표 25. 내장 함수에서 유형이 지정되지 않은 표현식 사용 (계속)

미입력 매개변수 표시문자 위치	데이터 유형
POSSTR(다른 인수가 문자 데이터 유형인 경우 하나의 인수)	VARCHAR(32672)
POSSTR(다른 인수가 그래픽 데이터 유형인 경우 하나의 인수)	VARGRAPHIC(16336)
POSSTR(다른 인수가 BLOB인 경우 검색 문자열 인수)	BLOB(32672)
SUBSTR의 첫 번째 인수	VARCHAR(32672)
SUBSTR의 두 번째 및 세 번째 인수	INTEGER
TRANSLATE의 두 번째 및 세 번째 인수	첫 번째 인수가 문자 유형인 경우 VARCHAR(32672), 첫 번째 인수가 그래픽 유형인 경우 VARGRAPHIC(16336)
TRANSLATE의 네 번째 인수	첫 번째 인수가 문자 유형일 경우 VARCHAR(1), 첫 번째 인수가 그래픽 유형일 경우 VARGRAPHIC(1)
TIMESTAMP의 두 번째 인수	TIME
VARCHAR_FORMAT의 첫 번째 인수	TIMESTAMP(12)
TIMESTAMP_FORMAT의 첫 번째 인수	VARCHAR(254)
XMLVALIDATE의 첫 번째 인수	XML
XMLCOMMENT의 첫 번째 인수	VARCHAR(32672)
XMLTEXT의 첫 번째 인수	VARCHAR(32672)
XMLPI의 두 번째 인수	VARCHAR(32672)
XMLSERIALIZE의 첫 번째 인수	XML
XMLDOCUMENT의 첫 번째 인수	XML
XMLXSROBJECTID의 첫 번째 인수	XML
XMLCONCAT의 모든 인수	XML
TRIM_ARRAY의 두 번째 인수	BIGINT
ARRAY의 배열 인덱스	BIGINT
단항 빼기	DECFLOAT(34)
단항 더하기	DECFLOAT(34)
다른 모든 스칼라 함수의 다른 모든 인수	함수 결정에 의해 판별된 함수 정의의 매개변수 데이터 유형. 인수 길이는 함수 결정 섹션의 216 페이지의 표 20를 기초로 파생됩니다.
집계 함수의 인수	오류

표 26. 사용자 정의 루틴에서 유형이 지정되지 않은 표현식 사용

미입력 매개변수 표시문자 위치	데이터 유형
함수의 인수	함수를 작성할 때 정의된 매개변수의 데이터 유형 및 길이.
메소드의 인수	오류
프로시저의 인수	프로시저를 작성할 때 정의된 매개변수의 데이터 유형

행 표현식

행 표현식은 특정한 사용자 정의 행 유형 또는 내장 데이터 유형 ROW가 포함되도록 데이터 행을 지정합니다.

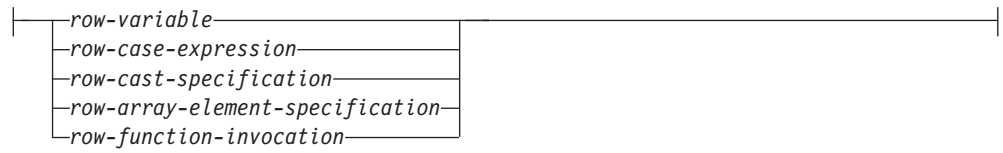
권한 부여

일부 행 표현식을 사용하려면 적절한 권한이 필요할 수도 있습니다. 이 행 표현식에 대해 명령문의 권한 부여 ID가 보유한 특권에는 다음과 같은 권한이 포함되어야 합니다.

- *row-variable*. *row-variable*이 전역 변수인 경우 권한 부여 고려사항에 대한 정보는 『전역 변수』를 참조하십시오.
- *row-function-invocation*. 함수를 실행하는 권한. 권한 부여 고려사항에 대한 정보는 『함수 호출』을 참조하십시오.
- *expression*. *row-expression*에서 참조되는 특정 표현식을 사용하려면 권한이 필요할 수도 있습니다. 권한 부여 고려사항에 대한 정보는 『표현식』을 참조하십시오.

구문

row-expression:



설명

row-variable

행 유형으로 정의되는 변수입니다.

row-case-expression

행 유형을 리턴하는 CASE 표현식입니다.

row-cast-specification

행 유형을 리턴하는 CAST입니다.

row-array-element-specification

행 유형 요소가 포함된 배열의 array-element-specification입니다.

row-function-invocation

행 유형인 리턴 유형을 포함하는 사용자 정의 함수(UDF)의 *function-invocation*입니다. 함수는 사용자 정의 행 유형 또는 정의된 필드 이름 및 필드 유형이 포함된 데이터 유형 ROW를 리턴할 수 있습니다.

주

- 행 표현식을 사용하여 SQL PL 컨텍스트에 행을 생성할 수 있습니다.

예 :

술어

*술어*는 주어진 값, 행 또는 그룹에 대해 참, 거짓 또는 알 수 없음 조건을 지정합니다.

다음 규칙은 술어의 모든 유형에 적용됩니다.

- 술어에 지정된 모든 값은 호환되어야 합니다.
- 규정된 기본 IN 또는 BETWEEN 술어에 사용되는 표현식은 길이 속성이 4000보다 큰 문자열, 길이 속성이 2000보다 큰 그래픽 문자열 또는 임의 *크기*의 LOB 문자열이 되어서는 안됩니다.
- 호스트 변수의 값은 널(NULL)이 될 수 있습니다(즉, 변수는 음수 표시기 값을 가질 수 있음).
- LIKE를 제외한 둘 이상의 피연산자를 포함하는 술어의 피연산자 코드 페이지 변환은 문자열 변환 규칙에 따라 수행됩니다.
- 구조화된 유형 값을 사용할 경우 NULL 술어와 TYPE 술어로 제한됩니다.
- 유니코드 데이터베이스에서 문자열 또는 그래픽 문자열을 허용하는 모든 술어는 변환이 지원되는 모든 문자열 유형을 허용합니다.

fullselect는 SELECT문 양식으로, 술어에서 사용될 때 *서브쿼리*라고도 합니다.

쿼리에 대한 술어 처리

술어는 비교 연산을 표현하거나 포함하는 검색 조건의 요소입니다. 술어는 평가 프로세스에서 술어가 사용되는 방법과 시기로 판별되는 네 범주로 그룹화할 수 있습니다. 범주는 가장 자주 사용되는 것으로 시작하여 성능 관점으로 순서화되어 아래에 나열되어 있습니다.

- 범위 구분 술어는 인덱스 스캔을 일괄로 다루기 위해 사용되는 술어로, 인덱스 검색에 대한 시작 또는 중지 키 값을 제공합니다. 이 술어는 인덱스 관리 프로그램에 의해 평가됩니다.
- 인덱스 Sargable 술어는 검색을 일괄로 다루기 위해 사용되지는 않지만 선택된 경우 인덱스를 통해 평가됩니다. 술어에 포함된 컬럼이 인덱스 키의 일부이기 때문입니다. 이 술어도 인덱스 관리 프로그램에 의해 평가됩니다.
- 데이터 Sargable 술어는 인덱스 관리 프로그램이 평가할 수 없는 술어이지만 DMS(Data Management Services)에 의해 평가될 수 있습니다. 일반적으로 이 술어를 사용하려면 기본 테이블에서 개별 행에 액세스할 수 있어야 합니다. 필요하면, DMS는 술어 평가에 필요한 컬럼과, 인덱스에서 얻을 수 없는 SELECT 목록의 컬럼을 충족시키기 위한 컬럼을 검색합니다.
- 레지듀얼(Residual) 술어는 기본 테이블의 단순 액세스를 벗어나는 입출력이 필요한 술어입니다. 레지듀얼(Residual) 술어에는 규정된 서브쿼리(ANY, ALL, SOME 또는 IN이 있는 서브쿼리)를 사용하거나, 테이블로부터 개별적으로 저장되는 대형 오브젝트(LOB) 데이터를 읽는 술어가 포함됩니다. 이 술어는 RDS(Relational Data Services)에 의해 평가되며 술어의 네 범주 중 가장 비용이 많이 듭니다.

다음 테이블은 다양한 술어의 예를 제공하고 그 예가 사용되는 컨텍스트를 기초로 해당 유형을 식별합니다.

주: 이 예에서는 다중 컬럼 오름차순 인덱스가 (c1, c2, c3)에 존재하고 적절한 경우 술어 평가에서 사용됩니다. 인덱스의 컬럼이 내림차순이면, 범위 구분 술어의 경우 시작 및 중지 키가 전환될 수도 있습니다.

표 27. 다양한 쿼리에 대한 술어 처리

술어	컬럼 c1	컬럼 c2	컬럼 c3	설명
c1 = 1 and c2 = 2 and c3 = 3	범위 구분(시작-중지)	범위 구분(시작-중지)	범위 구분(시작-중지)	인덱스의 모든 컬럼에 대한 등호 술어가 시작-중지 키로 적용될 수 있습니다.
c1 = 1 and c2 = 2 and c3 ≥ 3	범위 구분(시작-중지)	범위 구분(시작-중지)	범위 구분(시작)	컬럼 c1 및 c2는 등호 술어에 의해 바운드되고, c3의 술어만 시작 키로 적용됩니다.

표 27. 다양한 쿼리에 대한 술어 처리 (계속)

술어	컬럼 c1	컬럼 c2	컬럼 c3	설명
c1 ≥1 and c2 = 2	범위 구분(시작)	범위 구분(시작-중지)	해당되지 않음	선행 컬럼 c1에는 ≥ 술어가 있으며 시작 키로 사용될 수 있습니다. 다음 컬럼 c2에는 등호 술어가 있으므로, 시작-중지 키로 적용될 수도 있습니다.
c1 = 1 and c3 = 3	범위 구분(시작-중지)	해당되지 않음	인덱스 sargable	c3의 술어는 시작-중지 키로 사용할 수 없습니다. c2에 술어가 없기 때문입니다. 그러나 인덱스 Sargable 술어로 적용될 수 있습니다.
c1 = 1 and c2 > 2 and c3 = 3	범위 구분(시작-중지)	범위 구분(시작)	인덱스 sargable	c3의 술어는 시작-중지 술어로 적용될 수 없습니다. 이전 컬럼에 > 술어가 있기 때문입니다. 대신 ≥이었으면 이를 시작-중지 키로 사용할 수 있습니다.
c1 = 1 and c2 ≤ 2 and c4 = 4	범위 구분(시작-중지)	범위 구분(중지)	데이터 sargable	여기에서 c2의 술어는 ≤ 술어입니다. 이 술어는 중지 키로 사용될 수 있습니다. c4의 술어는 인덱스에서 적용될 수 없고 FETCH 중에 데이터 Sargable 술어로 적용됩니다.
c2 = 2 and UDF_with_ external_ action (c4)	해당되지 않음	인덱스 sargable	레지듀얼 (Residual)	선행 컬럼 c1에는 술어가 없으므로, c2의 술어가, 전체 인덱스가 스캔되는 인덱스 Sargable 술어로 적용될 수 있습니다. 외부 조치와 함께 사용자 정의 함수를 포함하는 술어는 레지듀얼(Residual) 술어로 적용됩니다.
c1 = 1 or c2 = 2	인덱스 sargable	인덱스 sargable	해당되지 않음	OR이 있다고 해서 해당되는 다중 컬럼 인덱스를 시작-중지 키로 사용할 수 있는 것은 아닙니다. 두 개의 인덱스(c1의 선행 컬럼이 있는 인덱스와 c2의 선행 컬럼이 있는 다른 인덱스)가 있었고 DB2 옵티마이저가 "인덱스-OR 처리" 플랜을 선택하였을 수 있습니다. 그러나 이 경우 두 술어는 인덱스 Sargable 술어로 처리됩니다.
c1 < 5 and (c2 = 2 or c3 = 3)	범위 구분(중지)	인덱스 sargable	인덱스 sargable	여기에서 선행 컬럼 c1은 중지 키가 있는 술어를 사용하여 인덱스 스캔을 중지하기 위해 사용됩니다. c2 및 c3의 OR 술어는 인덱스 Sargable 술어로 적용됩니다.

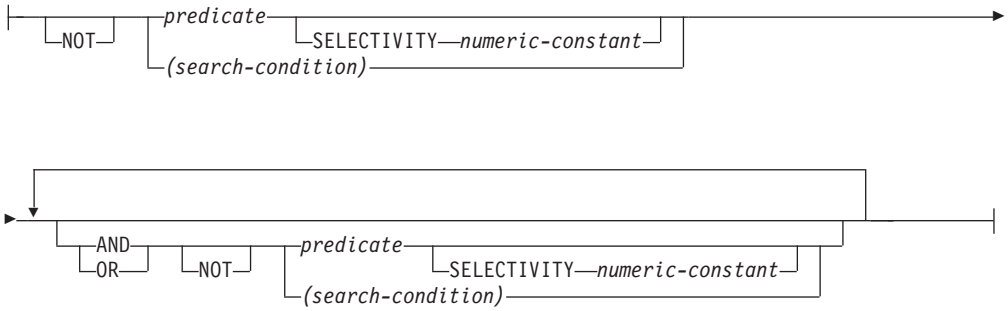
DB2 옵티마이저는 다음 표에 표시된 것처럼, 쿼리 다시 쓰기 메커니즘을 사용하여 많은 복잡한 사용자 작성 술어를 더 나은 수행 쿼리로 변환합니다.

표 28. 쿼리 다시 쓰기 술어

원래 술어 또는 쿼리	최적화된 술어	주석
c1 between 5 and 10	c1 ≥ 5 and c1 ≤ 10	BETWEEN 술어는 사용자가 지정하는 범위 구분 술어를 통해 내부적으로 사용될 수 있도록 해당되는 범위 구분 술어로 다시 작성됩니다.
c1 not between 5 and 10	c1 < 5 or c1 > 10	OR 술어가 있어도 DB2 옵티마이저가 인덱스-OR 처리 플랜을 선택하지 않으면 시작-중지 키를 사용할 수 없습니다.
SELECT * **FROM** t1 **WHERE EXISTS** (**SELECT** c1 **FROM** t2 **WHERE** t1.c1 = t2.c1)	**SELECT** t1.* **FROM** t1 EOJOIN t2 **WHERE** t1.c1= t2.c1	쿼리는 조인으로 변환될 수 있습니다.
SELECT * **FROM** t1 **WHERE** t1.c1 **IN** (**SELECT** c1 **FROM** t2)	**SELECT** t1* **FROM** t1 EOJOIN t2 **WHERE** t1.c1= t2.c1	이는 위의 EXISTS 술어 예에 대한 변환과 유사합니다.
c1 like 'abc%'	c1 ≥ 'abc X X X ' and c1 ≤ 'abc Y Y Y'	인덱스의 선행 컬럼으로 c1을 가지고 있는 경우, DB2는 범위 구분 시작-중지 술어로 적용될 수 있도록 이 술어를 생성합니다. 여기에서 문자 X 및 Y는 최하위 및 최상위 조합 문자의 기호입니다.
c1 like 'abc%def'	c1 ≥ 'abc X X X ' and c1 ≤ 'abc Y Y Y' and c1 like 'abc%def'	이는 원래 술어도 인덱스 Sargable 술어로 적용해야 하는 것을 제외하고 이전 경우와 유사합니다. 이는 문자 정의가 올바르게 일치하도록 합니다.

검색 조건

search-condition:



*검색 조건*은 지정한 값, 행 또는 그룹에 대해 ″true″, ″false″ 또는 ″unknown″인 조건을 지정합니다.

검색 조건의 결과는 지정된 *논리 연산자*(AND, OR, NOT)를 지정된 각 술어의 결과에 적용하여 파생됩니다. 논리 연산자를 지정하지 않은 경우, 검색 조건의 결과는 지정된 술어의 결과입니다.

AND 및 OR은 표 29에 정의되며, P 및 Q는 술어입니다.

표 29. AND 및 OR에 대한 진리표

P	Q	P AND Q	P OR Q
True	True	True	True
True	False	False	True
True	Unknown	Unknown	True
False	True	False	True
False	False	False	False
False	Unknown	False	Unknown
Unknown	True	Unknown	True
Unknown	False	False	Unknown
Unknown	Unknown	Unknown	Unknown

NOT(true)가 false, NOT(false)가 true, NOT(unknown)가 알 수 없음입니다.

괄호 내의 검색 조건이 먼저 평가됩니다. 평가 순서가 괄호 안에 지정되지 않은 경우 AND 이전에 NOT가 적용되고 AND는 OR 이전에 적용됩니다. 같은 우선순위 레벨에 있는 연산자가 평가되는 순서는 검색 조건의 최적화를 위해 정의되지 않습니다.

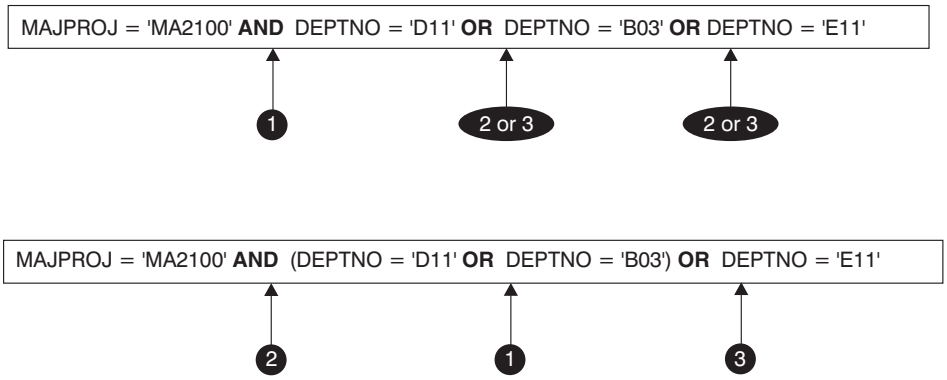


그림 12. 검색 조건 평가 순서

SELECTIVITY *value*

SELECTIVITY절은 술어에 대해 예측된 선택 빈도 백분율을 DB2에 표시하기 위해 사용됩니다. SELECTIVITY는 술어가 사용자 정의 술어인 경우에만 지정할 수 있습니다.

사용자 정의 술어는 CREATE FUNCTION의 PREDICATES절에 있는 술어 스펙과 일치하는 술어 스펙의 컨텍스트에서, 사용자 정의 함수 호출로 구성되는 술어입니다. 예를 들어, 함수 foo가 PREDICATES WHEN=1...을 사용하여 정의되는 경우, 다음의 SELECTIVITY 사용은 유효합니다.

```
SELECT *
  FROM STORES
  WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

선택 빈도 값은 0 - 1 범위(0과 1 포함)의 숫자 리터럴 값이어야 합니다(SQLSTATE 42615). SELECTIVITY를 지정하지 않은 경우, 디폴트값은 0.01입니다. 즉, 사용자 정의 술어는 모두 필터링할 것으로 예상되지만 테이블의 모든 행에서 1%만 필터링됩니다. SELECTIVITY 디폴트는 SYSSTAT.ROUTINES 뷰에서 SELECTIVITY 컬럼을 갱신하여 제공된 함수에 맞게 변경할 수 있습니다. 사용자 정의 술어가 아닌 술어에 SELECTIVITY절을 지정한 경우 오류가 리턴됩니다 (SQLSTATE 428E5).

사용자 정의 함수(UDF)는 사용자 정의 술어로 적용될 수 있으므로, 다음의 경우에 잠재적으로 인덱스 이용에 적용 가능합니다.

- CREATE FUNCTION 문에 술어 스펙이 있습니다.
- 술어 스펙에 지정된 것과 같은 방법으로 비교되는(구문적으로) WHERE절에서 UDF가 호출됩니다.
- 부정(NOT 연산자)이 없습니다.

예:

다음 쿼리에서, WHERE절 내의 within UDF 스펙은 세 가지의 조건을 모두 충족하므로 사용자 정의 술어로 고려됩니다.

```
SELECT *
  FROM customers
  WHERE within(location, :sanJose) = 1 SELECTIVITY  0.2
```

그러나 다음 쿼리에 within이 있어도 부정으로 인해 인덱스를 이용할 수 없고 사용자 정의 술어로 고려되지 않습니다.

```
SELECT *
  FROM customers
  WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY  0.3
```

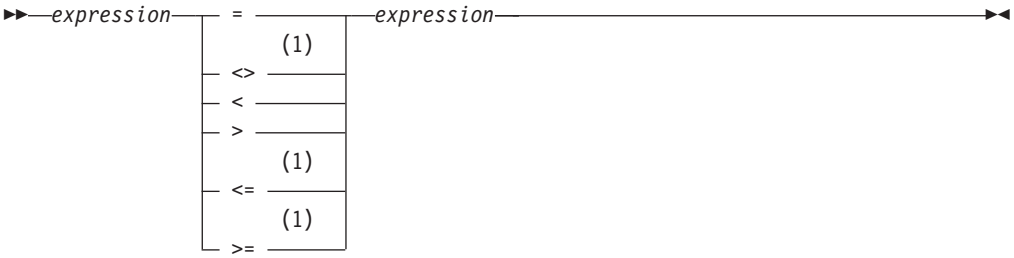
다음 예에서는 서로 특정 거리 안에 있는 고객 및 저장소를 식별하십시오. 저장소 사이의 거리는 고객이 사는 도시의 반경으로 계산됩니다.

```
SELECT *
  FROM customers, stores
  WHERE distance(customers.loc, stores.loc) <
    CityRadius(stores.loc) SELECTIVITY 0.02
```

위의 쿼리에서, WHERE절의 술어는 사용자 정의 술어로 고려됩니다. CityRadius에서 생성되는 결과는 탐색 범위 생성 함수에 대한 검색 인수로 사용됩니다.

그러나 CityRadius에서 생성되는 결과는 탐색 범위 생성 함수로 사용되므로, 위의 사용자 정의 술어는 stores.loc 컬럼에 정의된 인덱스 확장을 사용할 수 없습니다. 따라서 UDF는 customers.loc 컬럼에 정의된 인덱스만 사용합니다.

BASIC 술어



주:

1 ^=, ^<, ^>, !=, !< 및 !> 양식의 비교 연산자도 BASIC 및 QUANTIFIED 술어에서 지원됩니다. 코드 페이지 437, 819 및 850에서, 양식 ¬=, ¬< 및 ¬>이 지원됩니다. 이 비교 연산자의 제품 특정 양식 모두는 이 연산자를 사용하는 기존 SQL문만을 지원하기 위한 것이지만 새 SQL문을 작성할 때는 사용하지 않는 것이 좋습니다.

*BASIC 술어*는 두 개의 값을 비교합니다.

어느 하나의 피연산자 값이 널(NULL)인 경우, 술어의 결과는 알 수 없습니다. 그렇지 않으면 결과는 true 또는 false입니다.

값 *x* 및 *y*의 경우:

술어 다음의 경우에만 True

x **=** *y* *x*가 *y*와 같음

x **<>** *y*
*x*가 *y*와 같지 않음

x **<** *y* *x*가 *y*보다 작음

x **>** *y* *x*가 *y*보다 큼

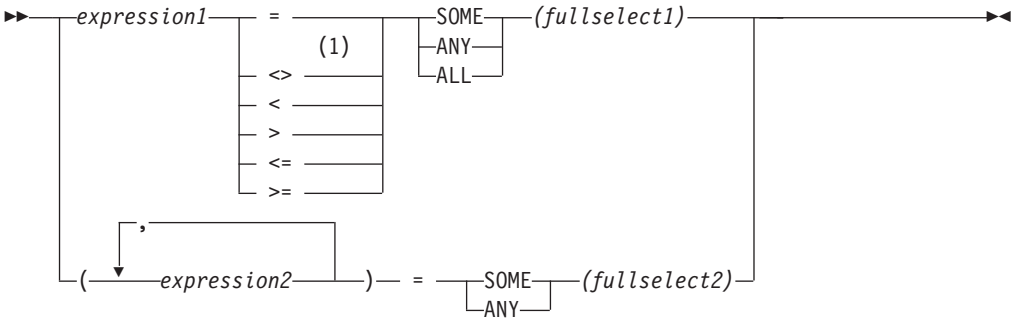
x **>=** *y*
*x*가 *y*보다 크거나 같음

x **<=** *y*
*x*가 *y*보다 작거나 같음

예:

```
EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

QUANTIFIED 술어



주:

1 ^=, ^<, ^>, !=, !< 및 !> 양식의 비교 연산자도 BASIC 및 QUANTIFIED 술어에서 지원됩니다. 코드 페이지 437, 819 및 850에서, 양식 ¬=, ¬< 및 ¬>이 지원됩니다. 이 비교 연산자의 제품 특정 양식 모두는 이 연산자를 사용하는 기존 SQL문만을 지원하기 위한 것이지만 새 SQL문을 작성할 때는 사용하지 않는 것이 좋습니다.

QUANTIFIED 술어는 값을 값 콜렉션과 비교합니다.

fullselect는 술어 연산자 왼쪽에 지정된 표현식 수와 같은 컬럼 수를 식별해야 합니다 (SQLSTATE 428C4). fullselect는 임의 행 개수를 리턴할 수 있습니다.

ALL이 지정된 경우:

- 술어 결과는 fullselect가 값을 리턴하지 않거나 지정된 관계가 fullselect에서 리턴되는 모든 값에 대해 참인 경우에 참이 됩니다.
- 지정된 관계가 fullselect에서 리턴되는 하나 이상의 값에 대해 거짓인 경우 결과는 거짓입니다.
- 지정된 관계가 fullselect에서 리턴되는 값에 대해 거짓이 아니고 널(NULL) 값으로 인해 하나 이상의 비교를 알 수 없는 경우 결과는 알 수 없음입니다.

SOME 또는 ANY가 지정된 경우:

- 술어 결과는 지정된 관계가 fullselect에서 리턴되는 하나 이상의 행 값마다 참인 경우에 참이 됩니다.
- fullselect가 행을 리턴하지 않거나 지정된 관계가 fullselect에서 리턴되는 모든 행의 하나 이상의 값에 대해 거짓인 경우 결과는 거짓입니다.
- 지정된 관계가 행 중 하나에 대해 참이 아니고 널(NULL) 값으로 인해 하나 이상의 비교를 알 수 없는 경우 결과는 알 수 없음입니다.

예: 다음 예를 참조할 때 다음 표를 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 13. 정량화 술어에 대한 테이블 예

예 1

```
SELECT COLA FROM TBLAB
   WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

결과는 2,3입니다. subselect는 (2,3)을 리턴합니다. 행 2 및 3의 COLA는 이 값 중 하나 이상과 같습니다.

예 2:

```
SELECT COLA FROM TBLAB
   WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

결과는 3,4입니다. subselect는 (2,3)을 리턴합니다. 행 3 및 4의 COLA는 이 값 중 하나 이상보다 큽니다.

예 3:

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

결과는 4입니다. subselect는 (2,3)을 리턴합니다. 행 4의 COLA는 이 두 개의 값 모두보다 큰 유일한 것입니다.

예 4

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY
                    WHERE COLX<0)
```

결과는 1, 2, 3, 4, null입니다. subselect는 어떤 값도 리턴하지 않습니다. 따라서 술어는 TBLAB의 모든 행에 대해 참입니다.

예 5

```
SELECT * FROM TBLAB
  WHERE (COLA,COLB+10) = SOME (SELECT COLX, COLY FROM TBLXY)
```

subselect는 TBLXY의 모든 항목을 리턴합니다. 술어는 subselect에 대해 참이므로 결과는 다음과 같습니다.

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

예 6

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB) = ANY (SELECT COLX,COLY-10 FROM TBLXY)
```

subselect는 TBLXY로부터 COLX와 COLY-10을 리턴합니다. 술어는 subselect에 대해 참이므로 결과는 다음과 같습니다.

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

ARRAY_EXISTS

ARRAY_EXISTS 술어는 배열에 배열 인덱스가 있는지 테스트합니다.

array-variable

배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

array-index

*array-index*의 데이터 유형은 배열의 배열 인덱스 데이터 유형에 지정할 수 있어야 합니다. *array-variable*이 일반 배열이면 *array-index*는 INTEGER에 지정할 수 있어야 합니다(SQLSTATE 428H1).

결과는 *array-variable*에 *array-variable* 배열 인덱스의 데이터 유형에 대한 *array-index* 캐스트와 동등한 배열 인덱스가 포함된 경우에는 true이고 포함되지 않은 경우에는 false입니다.

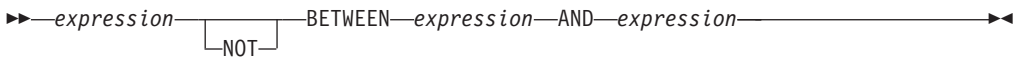
결과가 알 수 없음이 되는 것은 불가능합니다. 인수가 널(NULL)이거나 결과가 false입니다.

예 :

- 배열 변수 RECENT_CALLS가 PHONENUMBERS 배열 유형의 일반 배열로 정의되었다고 가정해 봅시다. 다음 IF문은 최근 전화 목록이 저장된 40번째 호출에 도달하지는 않았는지 테스트합니다. 도달한 경우에는 로컬 부울 변수 EIGHTY_PERCENT가 true로 설정됩니다.

```
IF (ARRAY_EXISTS(RECENT_CALLS, 40)) THEN
  SET EIGHTY_PERCENT = TRUE;
END IF
```

BETWEEN 술어



BETWEEN 술어는 특정 범위에 있는 값을 비교합니다. 피연산자의 데이터 유형이 동일하지 않으면, 모든 피연산자의 데이터 유형이 변환될 값이 없는 숫자인 경우를 제외하고는 모든 값은 『결과 데이터 유형에 대한 규칙』을 적용한 값이 될 수 있는 데이터 유형으로 변환됩니다.

BETWEEN 술어:

```
value1 BETWEEN value2 AND value3
```

이는 다음 검색 조건과 같습니다:

```
value1 >= value2 AND value1 <= value3
```

BETWEEN 술어:

```
value1 NOT BETWEEN value2 AND value3
```

이는 다음 검색 조건과 같습니다:

```
NOT(value1 BETWEEN value2 AND value3); that is,
   value1 < value2 OR value1 > value3.
```

첫 번째 피연산자(표현식)은 결정적이 아니거나 외부 조치를 포함하는 함수를 포함할 수 없습니다(SQLSTATE 42845).

예:

예 1

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000
```

$20,000.00과 $40,000.00 사이의 모든 급여가 결과로 나옵니다.

예 2

```
SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

:HV1이 5000이라고 가정하면, 모든 급여는 $25,000.00 아래와 $40,000.00 위에 있게 됩니다.

커서 술어

커서 술어는 현재 범위 내에 정의된 커서의 상태를 판별하기 위해 사용할 수 있는 SQL 키워드입니다. 이 키워드는 커서가 열려 있거나, 닫혀 있거나, 또는 커서와 연관된 행이 있는 경우 쉽게 참조할 수 있는 수단을 제공합니다.

cursor-variable-name
커서 유형의 SQL 변수 또는 SQL 매개변수의 이름입니다.

IS 커서 술어 등록 정보를 테스트할 것을 지정합니다.

NOT
커서 술어 등록 정보 테스트의 반대 값이 리턴됨을 지정합니다.

FOUND
FETCH문 실행 후 커서가 행을 포함하는지 여부를 확인할 것을 지정합니다. 실행된 마지막 FETCH문이 성공하고 IS FOUND 술어 구문이 사용되는 경우, 리턴값은 TRUE입니다. 실행된 마지막 FETCH문이 결국 행을 찾을 수 없음 상태가 된 경우 결과는 false입니다. 다음의 경우 결과를 알 수 없습니다.

- cursor-variable-name 값이 널(NULL)인 경우
- cursor-variable-name의 밑줄 긋기 커서가 열려 있지 않은 경우
- 밑줄 긋기 커서에 대해 첫 번째 FETCH 조치가 수행되기 전에 술어가 평가되는 경우
- 마지막 FETCH 조치가 오류를 리턴한 경우

IS FOUND 술어는 반복마다 페치를 루핑하고 수행하는 SQL PL 논리의 일부에서 유용할 수 있습니다. 술어를 사용하여 행이 페치되도록 남아 있는지 판별할 수 있습니다. 페치할 추가 행이 남아 있지 않은 경우 발생하는 오류 상태에 대해 검사하는 조건 핸들러를 사용하는 것보다 효율적인 대안을 제공합니다.

구문이 IS NOT FOUND가 되도록 NOT 키워드가 지정된 경우 결과 값은 반대입니다.

OPEN
커서가 열린 상태인지 여부를 검사할 것을 지정합니다. 커서가 열려 있고 IS OPEN 술어 구문이 사용되는 경우, 리턴값은 TRUE입니다. 이는 커서가 함수 및 프로시저에 매개변수로 전달되는 경우에 유용한 술어가 될 수 있습니다. 커서를 열려고 시도하기 전에 이 술어를 사용하여 커서가 이미 열려 있는지 판별할 수 있습니다.

구문이 IS NOT OPEN이 되도록 NOT 키워드가 지정된 경우 결과 값은 반대입니다.

주

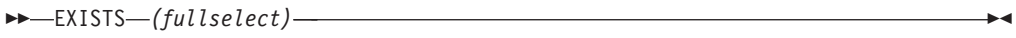
- 커서 술어는 복합 SQL(컴파일된)문 내의 명령문에서만 사용할 수 있습니다 (SQLSTATE 42818).

예 :

다음 스크립트는 이러한 술어와, 프로시저를 성공적으로 컴파일 및 호출하기 위해 필요한 전제 오브젝트에 대한 참조를 포함하는 SQL 프로시저를 정의합니다.

```
CREATE TABLE T1 (c1 INT, c2 INT, c3 INT)@

insert into t1 values (1,1,1),(2,2,2),(3,3,3) @

CREATE TYPE myRowType AS ROW(c1 INT, c2 INT, c3 INT)@

CREATE TYPE myCursorType AS myRowType CURSOR@


CREATE PROCEDURE p(OUT count INT)
     LANGUAGE SQL
   BEGIN
  DECLARE C1 cursor;
  DECLARE lvarInt INT;

  SET count = -1;
  SET c1 = CURSOR FOR SELECT c1 FROM t1;

  IF (c1 IS NOT OPEN) THEN
     OPEN c1;
           ELSE
     set count = -2;
           END IF;

  set count = 0;
  IF (c1 IS OPEN) THEN

    FETCH c1 into lvarInt;

    WHILE (c1 IS FOUND) DO
      SET count = count + 1;
      FETCH c1 INTO lvarInt;
    END WHILE;
           ELSE
     SET count = 0;
           END IF;

END@

CALL p()@
```

EXISTS 술어



EXISTS 술어는 특정 행의 존재 여부를 테스트합니다.

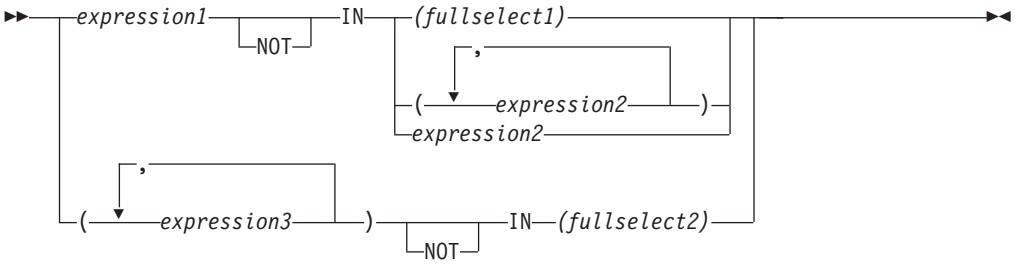
fullselect는 임의 개수의 컬럼을 지정할 수 있으며,

- 결과는 fullselect에 지정된 행 수가 0이 아닌 경우에만 true입니다.
- 결과는 지정된 행 수가 0인 경우 false입니다.
- 결과는 알 수 없음이 될 수 없습니다.

예 :

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

IN 술어



IN 술어는 값을 값 콜렉션과 비교합니다.

fullselect는 IN 키워드 왼쪽에 지정된 표현식 수와 같은 컬럼 수를 식별해야 합니다 (SQLSTATE 428C4). fullselect는 임의 행 개수를 리턴할 수 있습니다.

- 다음 IN 술어 양식은

  ```
  expression IN expression
  ```

  다음 기본 술어 양식과 같습니다.

  ```
  expression = expression
  ```

- 다음 IN 술어 양식은

  ```
  expression IN (fullselect)
  ```

  다음 QUANTIFIED 술어 양식과 같습니다.

  ```
  expression = ANY (fullselect)
  ```

- 다음 IN 술어 양식은

  ```
  expression NOT IN (fullselect)
  ```

  다음 QUANTIFIED 술어 양식과 같습니다.

  ```
  expression <> ALL (fullselect)
  ```

- 다음 IN 술어 양식은

  ```
  expression IN (expressiona, expressionb, ..., expressionk)
  ```

  다음과 같습니다.

  ```
  expression = ANY (fullselect)
  ```

  여기서, 값 정의 fullselect 양식은 다음과 같습니다.

  ```
  VALUES (expressiona), (expressionb), ..., (expressionk)
  ```

- 다음 IN 술어 양식은

  ```
  (expressiona, expressionb,..., expressionk) IN (fullselect)
  ```

  다음 QUANTIFIED 술어 양식과 같습니다.

```
(expressiona, expressionb,..., expressionk) = ANY (fullselect)
```

다음 술어 양식의 왼편에 있는 피연산자는 *row-value-expression*으로 참조됨을 참고하십시오.

expression1 및 *expression2*의 값이나 IN 술어의 *fullselect1* 컬럼은 호환 가능해야 합니다. 각각의 *expression3* 값과 해당되는 IN 술어의 *fullselect2* 컬럼은 호환 가능해야 합니다. 결과 데이터 유형에 대한 규칙을 사용하여 비교에 사용되는 결과 속성을 판별할 수 있습니다.

IN 술어의 표현식 값(fullselect의 해당 컬럼을 포함하여)은 다른 코드 페이지를 가질 수 있습니다. 변환이 필요한 경우, 두 번째 연산자로 IN 목록에 대해 파생된 코드 페이지를 사용하여 먼저 IN 목록에 문자열 변환 규칙을 적용한 후 술어에 적용하여 코드 페이지를 판별합니다.

예:

예 1: 다음은 DEPTNO 컬럼에서 평가하는 행의 값에 D01, B01 또는 C01이 포함되는 경우 참으로 평가됩니다.

```
DEPTNO IN ('D01', 'B01', 'C01')
```

예 2: 다음은 왼쪽에 있는 EMPNO(직원 번호)가 부서 E11의 직원 EMPNO와 일치하는 경우에만 참으로 평가됩니다.

```
EMPNO IN (SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

예 3: 다음 정보가 제공되는 경우, 이 예는 COL_1 컬럼의 행에 있는 특정 값이 목록에 있는 값과 일치하는 경우에 참으로 평가됩니다.

표 30. IN 술어 예

표현식	유형	코드 페이지
COL_1	컬럼	850
HV_2	호스트 변수	437
HV_3	호스트 변수	437
CON_1	constant	850

술어 평가는 다음과 같이 수행됩니다.

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

두 개의 호스트 변수는 문자열 변환 규칙을 기초로 코드 페이지 850으로 변환됩니다.

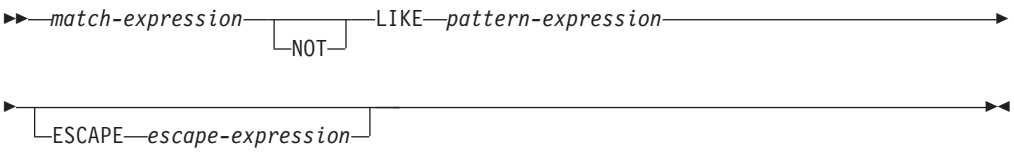
예 4: 다음은 EMENDATE(프로젝트에 대한 직원 활동이 종료된 날짜)에 지정된 연도가 목록에 지정된 값(현재 연도 또는 이전의 두 개 연도)과 일치하는 경우 참으로 평가됩니다.

```
YEAR(EMENDATE) IN (YEAR(CURRENT DATE),
                   YEAR(CURRENT DATE - 1 YEAR),
                   YEAR(CURRENT DATE - 2 YEARS))
```

예 5: 다음은 왼쪽에 있는 ID 및 DEPT 모두 ORG 테이블의 임의 행에 대한 MANAGER 및 DEPTNUMB와 각각 일치하는 경우 참으로 평가됩니다.

```
(ID, DEPT) IN (SELECT MANAGER, DEPTNUMB FROM ORG)
```

LIKE 술어



LIKE 술어는 특정 패턴을 가지고 있는 문자열을 검색합니다. 패턴은 밑줄 및 퍼센트 부호가 특수한 의미를 가질 수 있는 문자열에 지정됩니다. 패턴의 후행 공백도 패턴의 일부입니다.

인수 값이 널(NULL)일 경우, LIKE 술어의 결과는 알 수 없습니다.

match-expression, *pattern-expression* 및 *escape-expression*의 값은 호환 가능한 문자열 표현식입니다. 각 인수에 대해 지원되는 문자열 표현식 유형에는 약간의 차이가 있습니다. 유효한 표현식 유형은 각 인수에 대한 설명 아래에 나열되어 있습니다.

어떤 표현식도 구별 유형을 생성할 수 없습니다. 그러나 표현식은 구별 유형에서 소스 유형으로 캐스트하는 함수가 될 수는 있습니다.

match-expression

특정 문자 패턴을 따르는지 검사하는 문자열을 지정하는 표현식입니다.

표현식은 다음에서 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수(로케이터 변수 또는 파일 참조 변수 포함)
- 스칼라 함수
- 대형 오브젝트(LOB) 로케이터
- 컬럼 이름
- 위의 사항을 병합한 표현식

pattern-expression

일치시킬 문자열을 지정하는 표현식입니다.

표현식은 다음에서 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수

- 위의 사항을 병합한 표현식
- SQL 프로시저 매개변수

다음과 같은 제한사항이 있습니다.

- 표현식의 어떤 요소도 CLOB 또는 DBCLOB 유형이 될 수 없습니다. 또한 BLOB 파일 참조 변수가 될 수도 없습니다.
- *pattern-expression*의 실제 길이는 32 672바이트 이상일 수 없습니다.

다음은 올바르지 않은 문자열 표현식 또는 문자열의 예입니다.

- SQL 사용자 정의 함수 매개변수
- 트리거 전이 변수
- 동적 복합문의 로컬 변수

LINK 술어 사용에 대해 **간단히 설명**하자면, 패턴은 *match-expression*에 있는 값에 대한 준수 기준을 지정하는 데 사용됩니다. 여기서,

- 밑줄 문자(_)는 단일 문자를 나타냅니다.
- 퍼센트 부호(%)는 0개 이상의 문자로 된 문자열을 나타냅니다.
- 그 외의 문자는 문자 자체를 나타냅니다.

*pattern-expression*에 밑줄이나 퍼센트 문자를 포함해야 할 경우, *escape-expression*을 사용하여 패턴에서 밑줄이나 퍼센트 문자 앞에 오는 문자를 지정할 수 있습니다.

LIKE 술어 사용에 대한 **자세한 설명**은 다음과 같습니다. 이 설명에서는 *escape-expression* 사용에 대해서는 설명하지 않습니다. 이 표현식의 사용에 대해서는 이후에 설명하기로 합니다.

- *m*은 *match-expression*의 값을, *p*는 *pattern-expression*의 값을 나타낸다고 가정합니다. *p* 문자열은 부속 문자열 지정자의 최소 수로 해석되어 *p*의 각 문자는 정확히 한 부속 문자열 지정자의 일부와 일치합니다. 부속 문자열 지정자는 밑줄, 퍼센트 부호 또는 그 이외의 공백이 아닌 문자열 시퀀스입니다.

  술어의 결과는 *m*이나 *p*가 널(NULL) 값인 경우에는 알 수 없습니다. 그렇지 않은 경우, 결과는 참이나 거짓이 됩니다. 결과는 *m*과 *p*가 빈 문자열이고 다음과 같이 *m*에서 부속 문자열로의 파티션이 존재할 경우에 참이 됩니다.
  - *m*의 부속 문자열은 0개 이상의 연속적인 문자열로서, *m*의 각 문자는 정확히 한 부속 문자열의 일부와 일치합니다.
  - *n*번째 부속 문자열 지정자가 밑줄인 경우, *m*의 *n*번째 부속 문자열은 하나의 문자입니다.
  - *n*번째 부속 문자열 지정자가 퍼센트 부호인 경우, *m*의 *n*번째 부속 문자열은 0개 이상의 문자 시퀀스입니다.

- *n*번째 부속 문자열 지정자가 밑줄도 퍼센트 부호도 아닌 경우, *m*의 *n*번째 부속 문자열은 부속 문자열 지정자와 같으며, 부속 문자열 지정자의 길이와 같습니다.
- *m*의 부속 문자열 수는 부속 문자열 지정자의 수와 같습니다.

따라서 *p*가 빈 문자열이고 *m*이 빈 문자열이 아닌 경우, 결과는 거짓이 됩니다. 마찬가지로, *m*이 빈 문자열이고 *p*가 빈 문자열이 아닌 경우(퍼센트 부호만 있는 문자열은 예외), 결과는 거짓이 됩니다.

술어 *m* NOT LIKE *p*는 검색 조건 NOT(*m* LIKE *p*)와 같습니다.

*escape-expression*을 지정할 경우 바로 다음에 Escape 문자, 밑줄 또는 퍼센트 부호가 오는 경우를 제외하고 *pattern-expression*에 *escape-expression*으로 식별되는 Escape 문자가 포함되서는 안됩니다(SQLSTATE 22025).

*match-expression*이 MBCS 데이터베이스의 문자열인 경우, 혼합 데이터가 포함될 수 있습니다. 이 경우, 패턴에는 SBCS 및 비SBCS 문자가 모두 포함될 수 있습니다. 비유니코드 데이터베이스의 경우 패턴에 있는 특수 문자는 다음과 같이 해석됩니다.

- SBCS 반자(halfwidth) 밑줄은 하나의 SBCS 문자를 지칭합니다.
- 비SBCS 전자(fullwidth) 밑줄은 하나의 비SBCS 문자를 지칭합니다.
- SBCS 반자 또는 비SBCS 전자 퍼센트 부호는 0개 이상의 SBCS 또는 비SBCS 문자를 지칭합니다.

유니코드 데이터베이스에서는 ″단일 바이트″와 ″비단일 바이트″ 문자 간에 차이가 없습니다. UTF-8 형식은 유니코드 문자의 ″혼합 바이트″ 인코딩이지만 UTF-8의 SBCS와 비SBCS 문자를 구별하지는 않습니다. UTF-8 형식의 바이트 수와 상관없이 모든 문자는 유니코드 문자입니다.

유니코드 그래픽 컬럼에서 반자(halfwidth) 밑줄 문자(U+005U&'₩005F') 및 반자 퍼센트 부호 문자(U+002U&'₩0025')를 포함하여 모든 비보조(non-supplementary) 문자의 길이는 2바이트입니다. 유니코드 데이터베이스의 경우 패턴에 있는 특수 문자는 다음과 같이 해석됩니다.

- 문자열의 경우 반자 밑줄 문자(X'5F') 또는 전자 밑줄 문자((X'EFBCBF')는 하나의 유니코드 문자를 지칭하며 반자 퍼센트 부호 문자(X'25') 또는 전자 퍼센트 부호 문자((X'EFBC85')는 0개 이상의 유니코드 문자를 지칭합니다.
- 그래픽 문자열의 경우 반자 밑줄 문자(X'5FU&'₩005F') 또는 전자 밑줄 문자(U&'₩FF3F')는 하나의 유니코드 문자를 지칭하며 반자 퍼센트 부호 문자(U&'₩0025') 또는 전자 퍼센트 부호 문자(U&'₩FF05')는 0개 이상의 유니코드 문자를 지칭합니다.
- 로케일 구분 UCA-기반의 조합이 영향을 줄 경우 특수 문자로 인식하려면 밑줄 문자 및 퍼센트 기호 문자 다음에 공백이 없는 조합 마크(구분 발음 부호)가 와

서는 안됩니다. 예를 들어, 패턴 U&'%₩0300'(퍼센트 기호 문자 다음에 공백이 없는 조합 그레이브 액센트)은 %̀에 대한 검색 대상으로 해석되며 그레이브 액센트가 있는 문자 앞의 0 이상의 유니코드 문자에 대한 검색 대상으로 해석되지는 않습니다.

유니코드 보완 문자는 유니코드 그래픽 컬럼에서 두 개의 그래픽 코드 포인트로 저장됩니다. 유니코드 그래픽 컬럼에서 유니코드 보완 문자와 일치시키려면 데이터베이스가 로케일 구분 UCA 기반의 조합을 사용하고 있는 경우 하나의 밑줄을 사용하고 그렇지 않은 경우 두 개의 밑줄을 사용합니다. 유니코드 문자 컬럼에서 유니코드 보완 문자와 일치시키려면, 모든 조합에 대해 하나의 밑줄을 사용합니다. 기본 문자를 하나 이상의 공백이 없는 조합 문자와 일치시키려면, 데이터베이스가 로케일 구분 UCA 기반의 조합을 사용하고 있는 경우 하나의 밑줄을 사용합니다. 그렇지 않으면 공백이 포함되지 않은 조합 문자의 수에 기본 문자의 수를 더한 수만큼의 밑줄 문자를 사용합니다.

escape-expression

이 선택적 인수는 *pattern-expression*에서 밑줄(_)과 퍼센트(%)의 특별한 의미를 수정하는 데 사용될 문자를 지정하는 표현식입니다. 따라서 LIKE 술어가 실제 퍼센트 문자와 밑줄 문자가 들어 있는 값과 일치시키는 데 사용될 수 있습니다.

표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 사항을 병합한 표현식

다음과 같은 제한사항이 있습니다.

- 표현식의 어떤 요소도 CLOB 또는 DBCLOB 유형이 될 수 없습니다. 또한 BLOB 파일 참조 변수가 될 수 없습니다.
- 문자 컬럼의 경우, 표현식의 결과가 한 문자 또는 정확히 1바이트를 포함하는 실행 파일 문자열이어야 합니다(SQLSTATE 22019).
- 그래픽 컬럼의 경우, 표현식의 결과가 한 문자여야 합니다(SQLSTATE 22019).
- 표현식 결과가 공백을 포함하지 않은 조합 문자 시퀀스여서는 안됩니다(예를 들어, U&'₩0301', Combining Acute Accent).

Escape 문자가 패턴 문자열에 있으면 밑줄, 퍼센트 부호 또는 Escape 문자는 자신의 리터럴 어커런스를 나타낼 수 있습니다. 알 수 없는 문자 앞에 홀수의 Escape 문자가 오는 경우, 이것은 참이 됩니다. 그 밖의 경우에는 거짓입니다.

패턴에서 연속적인 Escape 문자 시퀀스는 다음과 같이 처리됩니다.

- S를 Escape 문자 시퀀스로 하고, S는 연속적인 Escape 문자의 대형 시퀀스 일부가 아니라고 가정합니다. 또한 S에 총 n자가 들어 있다고 가정합니다. 그러면 S에 적용되는 규칙은 n의 값에 따라 달라집니다.
  - n이 홀수인 경우, S 다음에는 밑줄이나 퍼센트 부호가 와야 합니다 (SQLSTATE 22025). S와 그 뒤에 오는 문자는 밑줄이나 퍼센트 부호의 리터럴 어커런스가 뒤에 오는 Escape 문자의 (n-1)/2 리터럴 어커런스를 나타냅니다.
  - n이 짝수인 경우, S는 Escape 문자의 n/2 리터럴 어커런스를 나타냅니다. n이 홀수인 경우와는 달리, S는 패턴을 종료할 수 있습니다. S가 패턴을 종료시키지 않는 경우, S 다음에는 모든 문자가 올 수 있습니다. 물론, S가 연속적인 Escape 문자 시퀀스의 일부가 아니라는 가정에 위배된 Escape 문자는 제외됩니다. S 다음에 밑줄이나 퍼센트 부호가 오는 경우, 그 문자에는 특별한 의미가 있습니다.

  다음은 Escape 문자(여기서는 백슬래시(₩))의 연속적인 어커런스가 어떤 영향을 주는지 나타냅니다.

  패턴 문자열
  실제 패턴

  ₩% 퍼센트 부호

  ₩₩% 백슬래시 다음에 0개 이상의 임의의 문자가 옵니다.

  ₩₩₩% 백슬래시 다음에 퍼센트 부호가 옵니다.

비교에 사용되는 코드 페이지는 *match-expression* 값의 코드 페이지를 기준으로 합니다.

- *match-expression* 값은 변환되지 않습니다.
- *pattern-expression*의 코드 페이지가 *match-expression*의 코드 페이지와 다른 경우, *pattern-expression*의 값은 피연산자가 FOR BIT DATA로 정의(이런 경우, 변환되지 않음)되지 않으면 *match-expression*의 코드 페이지로 변환됩니다.
- *escape-expression*의 코드 페이지가 *match-expression*의 코드 페이지와 다른 경우, *escape-expression*의 값은 피연산자가 FOR BIT DATA로 정의(이런 경우 변환되지 않음)되지 않으면 *match-expression*의 코드 페이지로 변환됩니다.

주

- 후행 공백 수는 *match-expression*과 *pattern-expression*에서 모두 의미가 있습니다. 따라서 문자열의 길이가 같지 않은 경우, 더 짧은 문자열이 공백으로 채워지지 않습니다. 예를 들어, 표현식 `'PADDED ' LIKE 'PADDED'`는 일치하지 않게 됩니다.

- LIKE 술어에 지정된 패턴이 매개변수 표시문자이고 고정 길이 문자 호스트 변수가 매개변수 표시문자를 대체하는 데 사용되는 경우, 호스트 변수에 지정된 값은 올바른 길이를 가져야 합니다. 올바른 길이를 지정하지 않은 경우, 선택 조작이 원하는 결과를 리턴하지 않습니다.

  예를 들어, 호스트 변수가 CHAR(10)으로 정의되고 WYSE% 값이 해당 호스트 변수에 지정되는 경우, 지정시 호스트 변수가 공백으로 채워집니다. 사용되는 패턴은 다음과 같습니다.

  ```
  'WYSE%     '
  ```

  데이터베이스 관리 프로그램에서는 WYSE로 시작하고 5개의 공백으로 끝나는 모든 값을 검색합니다. 'WYSE'로 시작하는 값만 검색하려는 경우, 호스트 변수에 'WSYE%%%%%%' 값을 지정하십시오.
- 피연산자가 FOR BIT DATA로 정의되지 않으면, 2진 비교를 사용하여 패턴이 일치되는 경우, 데이터베이스 조합을 사용하면 패턴이 일치됩니다.

예:

- PROJECT 테이블에 있는 PROJNAME 컬럼에 나타나는 문자열'SYSTEMS'를 검색하십시오.

  ```
  SELECT PROJNAME FROM PROJECT
    WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
  ```

- EMPLOYEE 테이블의 FIRSTNME 컬럼에서 문자의 길이가 정확히 2이고'J'로 시작하는 문자열을 검색하십시오.

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
  ```

- EMPLOYEE 테이블의 FIRSTNME 컬럼에서 첫 번째 문자가 'J'인 임의의 길이를 가진 문자열을 검색하십시오.

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
  ```

- CORP_SERVERS 테이블에서 CURRENT SERVER 특수 레지스터 내의 값과 일치하는 LA_SERVERS 컬럼의 문자열을 검색하십시오.

  ```
  SELECT LA_SERVERS FROM CORP_SERVERS
    WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
  ```

- 테이블 T의 컬럼 A에서 문자 시퀀스 '₩'로 시작하는 모든 문자열을 검색하십시오.

  ```
  SELECT A FROM T
    WHERE T.A LIKE '₩_₩₩%' ESCAPE '₩'
  ```

- BLOB 스칼라 함수를 사용하여 일치 및 패턴 데이터 유형(모두 BLOB)과 호환 가능한 1바이트 Escape 문자를 검색하십시오.

  ```
  SELECT COLBLOB FROM TABLET
    WHERE COLBLOB LIKE :pattern_var ESCAPE BLOB(X'0E')
  ```

- 대소문자가 구별되지 않는 조합 UCA500R1_LEN_S1으로 정의된 유니코드 데이터 베이스에서, 'Bill'로 시작되는 모든 이름을 찾습니다.

  ```
  SELECT NAME FROM CUSTDATA WHERE NAME LIKE 'Bill%'
  ```

  이름 'Bill Smith', 'billy simon' 및 'BILL JONES'를 리턴합니다.

NULL 술어

NULL 술어는 널(NULL) 값에 대해 테스트합니다.

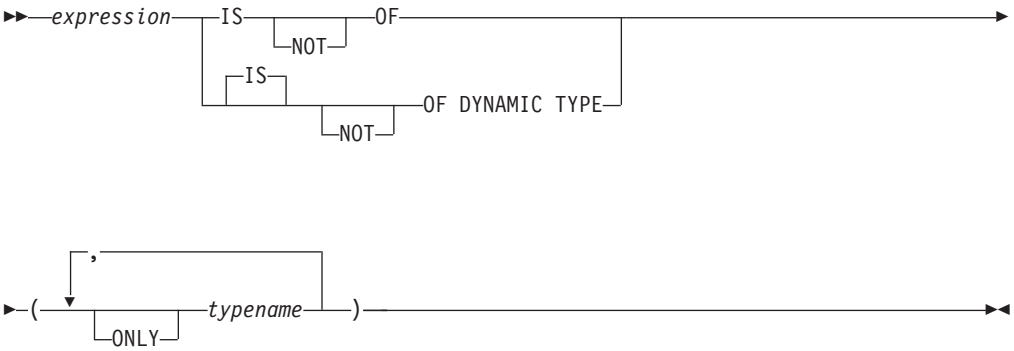
NULL 술어의 결과는 알 수 없습니다. 표현식 값이 널(NULL)일 경우, 결과는 참이 됩니다. 값이 널(NULL)이 아닐 경우, 결과는 거짓이 됩니다. NOT이 지정된 경우, 결과는 반대가 됩니다.

행 유형 값은 필드 이름의 규정자인 경우를 제외하고는 NULL 술어에서 피연산자로 사용될 수 없습니다. *expression*이 행 유형인 경우, 오류가 리턴됩니다(SQLSTATE 428H2).

예:

```
PHONENO IS NULL

SALARY IS NOT NULL
```

TYPE 술어



TYPE 술어는 표현식의 유형을 하나 이상의 사용자 정의 구조화 유형과 비교합니다.

참조 유형의 비참조를 포함하는 표현식의 동적 유형은 대상 유형이 지정된 테이블 또는 뷰에서 참조된 행의 실제 유형입니다. 이는 표현식의 정적 유형이라고 하는 참조를 포함하는 표현식의 목표 유형과 다를 수 있습니다.

expression 값이 널(NULL)인 경우, 술어의 결과는 알 수 없습니다. *expression*의 동적 유형이 *typename*으로 지정된 구조화된 유형 중 하나의 하위 유형인 경우에 술어의 결과는 참이 되며, 그렇지 않은 경우에는 거짓이 됩니다. ONLY가 *typename* 앞에 있는 경우 해당 유형의 적절한 하위 유형은 고려되지 않습니다.

*typename*을 규정하지 않는 경우 SQL 경로를 사용하여 분석됩니다. 각 *typename*은 *expression*의 정적 유형에 대한 유형 계층에 있는 사용자 정의 유형을 식별해야 합니다(SQLSTATE 428DU).

TYPE 술어에 참조 유형 값을 포함하는 표현식이 있을 때마다 DEREF 함수를 사용해야 합니다. 이 양식의 *expression*에 대한 정적 유형은 참조의 대상 유형입니다.

구문 IS OF 및 OF DYNAMIC TYPE은 TYPE 술어에 대해 동등한 대체사항입니다. 마찬가지로 IS NOT OF 및 NOT OF DYNAMIC TYPE도 동등한 대체사항입니다.

예:

EMP 유형의 루트 테이블 EMPLOYEE와 MGR 유형의 서브테이블 MANAGER가 있는 테이블 계층이 존재합니다. 다른 테이블 ACTIVITIES에는 REF(EMP) SCOPE EMPLOYEE로 정의된 WHO_RESPONSIBLE 컬럼이 포함되어 있습니다. 다음은 WHO_RESPONSIBLE에 해당되는 행이 관리자인 경우 true로 평가되는 TYPE 술어입니다.

```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

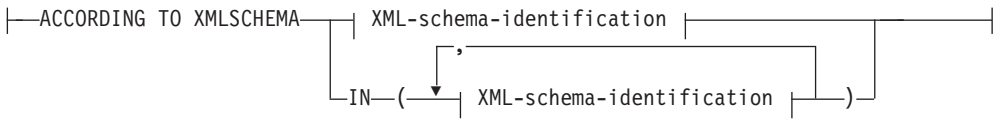
테이블에 EMP 유형의 컬럼 EMPLOYEE가 포함되는 경우, EMPLOYEE는 유형 EMP의 값과 MGR과 같은 하위 유형의 값을 포함할 수 있습니다. 다음 술어는

```
EMPL IS OF (MGR)
```

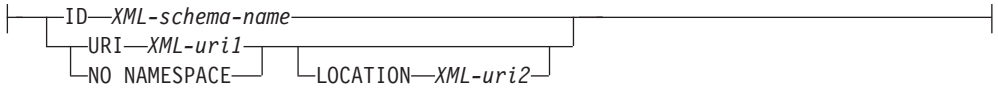
EMPL이 널(NULL)이 아니고 실제로 관리자인 경우 true를 리턴합니다.

VALIDATED 술어

according-to-clause:



XML-schema-identification:



VALIDATED 술어는 *XML-expression*에 지정된 값이 XMLVALIDATE 함수를 사용하여 유효성이 확인되었는지 여부를 테스트합니다. 지정된 값이 널(NULL)일 경우 유효성 확인 제한조건의 결과는 알 수 없습니다. 널(NULL)이 아닐 경우 유효성 확인 제한조건의 결과는 참 또는 거짓입니다. 지정하는 값은 XML 유형이어야 합니다.

ACCORDING TO XMLSCHEMA 절을 지정하지 않은 경우, 유효성 확인에 사용한 XML 스키마는 유효성 확인 제한조건의 결과에 영향을 주지 않습니다.

설명

XML-expression

테스트한 XML 값을 지정합니다. 여기서 *XML-expression*은 XML 문서, XML 컨텐츠, XML 노드 시퀀스, XML *column-name* 또는 XML *correlation-name*으로 구성될 수 있습니다.

XML *column-name*이 지정된 경우 술어는 지정된 컬럼 이름과 연관된 XML 문서의 유효성이 확인되었는지 여부를 평가합니다.

트리거의 일부로 XML 유형의 상관 이름을 지정하는 것에 대한 정보는 ″CREATE TRIGGER″를 참조하십시오.

IS VALIDATED 또는 IS NOT VALIDATED

XML-expression 피연산자에 대한 필수 유효성 확인 상태를 지정합니다.

참으로 평가되도록 IS VALIDATED를 지정하는 제한조건의 경우 피연산자의 유효성이 확인되어 있어야 합니다. 선택적 ACCORDING TO XMLSCHEMA 절에 하나 또는 몇 개의 XML 스키마가 포함되는 경우, 피연산자는 식별된 XML 스키마 중 하나를 사용하여 유효성이 확인되어 있어야 합니다.

거짓으로 평가되도록 IS NOT VALIDATED를 지정하는 제한조건의 경우 피연산자는 유효성이 확인된 상태여야 합니다. 선택적 ACCORDING TO XMLSCHEMA 절에 하나 또는 몇 개의 XML 스키마가 포함되는 경우, 피연산자는 식별된 XML 스키마 중 하나를 사용하여 유효성이 확인되어 있어야 합니다.

according-to-clause

피연산자 유효성이 확인되거나 확인되지 않아야 하는 하나 또는 몇 개의 XML 스키마를 지정합니다. XML 스키마 저장소에서 이전에 등록한 XML 스키마만 지정할 수 있습니다.

ACCORDING TO XMLSCHEMA

ID *XML-schema-name*

XML 스키마의 SQL ID를 지정합니다. 내재적 또는 명시적 SQL 스키마 규정자를 포함하는 이름은 현재 서버에서 XML 스키마 저장소에 있는 기존 XML 스키마를 고유하게 식별합니다. 내재적 또는 명시적으로 지정된 SQL 스키마에 이 이름에 의한 XML 스키마가 존재하지 않으면 오류가 리턴됩니다(SQLSTATE 42704).

URI *XML-uri1*

XML 스키마의 목표 이름 공간 URI를 지정합니다. *XML-uri1*의 값은 URI를 공백이 아닌 문자열 상수로 지정합니다. URI는 등록된 XML 스키마의 목표 이름 공간이어야 하며(SQLSTATE 4274A), LOCATION절이 지정되어 있지 않은 경우 등록된 XML 스키마를 고유하게 식별해야 합니다(SQLSTATE 4274B).

NO NAMESPACE

XML 스키마에 목표 이름 공간이 없음을 지정합니다. 목표 이름 공간 URI는 명시 목표 이름 공간 URI로 지정될 수 없는 공백 문자열과 동일합니다.

LOCATION *XML-uri2*

XML 스키마의 XML 스키마 위치 URI를 지정합니다. *XML-uri2*의 값은 URI를 공백이 아닌 문자열 상수로 지정합니다. 목표 이름 공간 URI와 조인된 XML 스키마 위치 URI는 등록된 XML 스키마를 식별해야 하며(SQLSTATE 4274A) 이러한 XML 스키마는 한 개만 등록되어야 합니다(SQLSTATE 4274B).

예:

예 1: 테이블 T1에 컬럼 XMLCOL이 정의되어 있다고 가정하십시오. XML 스키마에 의해 유효성이 확인된 XML 값만 검색하십시오.

```
SELECT XMLCOL FROM T1
  WHERE XMLCOL IS VALIDATED
```

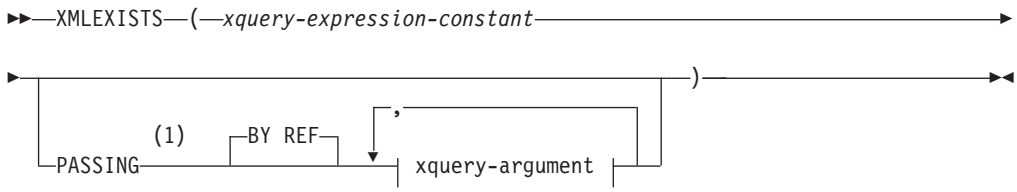
예 2: 테이블 T1에 컬럼 XMLCOL이 정의되어 있다고 가정하십시오. 유효성이 확인되지 않았으면 값을 삽입하거나 갱신할 수 없도록 규칙을 강제 수행하십시오.

```
ALTER TABLE T1 ADD CONSTRAINT CK_VALIDATED
  CHECK (XMLCOL IS VALIDATED)
```

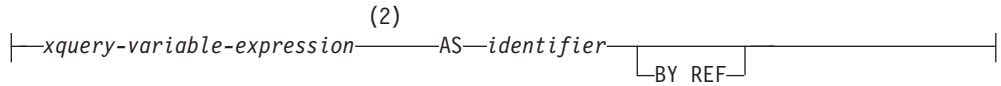
예 3: XML 스키마 URI http://www.posample.org에 대해 유효성이 확인된 XML 컬럼 XMLCOL이 있는 테이블 T1의 행만 선택한다고 가정합니다.

```
SELECT XMLCOL FROM T1
  WHERE XMLCOL IS VALIDATED
   ACCORDING TO XMLSCHEMA URI
   'http://www.posample.org'
```

XMLEXISTS 술어



xquery-argument:



주:

1 데이터 유형은 DECFLOAT가 될 수 없습니다.

2 표현식의 데이터 유형은 DECFLOAT가 될 수 없습니다.

XMLEXISTS 술어는 XQuery 표현식이 하나 이상 항목의 시퀀스를 리턴하는지 여부를 테스트합니다.

xquery-expression-constant

XQuery 표현식으로 해석되는 SQL 문자열 상수를 지정합니다. 이 상수 문자열은 데이터베이스 또는 섹션 코드 페이지로 변환되지 않고 직접 UTF-8로 변환됩니다. XQuery 표현식은 입력 XML 값의 선택적 설정을 사용하여 실행된 후 XMLEXISTS 술어의 결과를 판별하도록 테스트되는 출력 시퀀스를 리턴합니다. *xquery-expression-constant*의 값은 빈 문자열 또는 공백 문자의 문자열이 아니어야 합니다(SQLSTATE 10505).

PASSING

입력 값 및 이들 입력 값이 *xquery-expression-constant*에서 지정된 XQuery 표현식으로 패스되는 방법을 지정합니다. 디폴트로, 함수가 호출되는 범위 내에 있는 모든 고유 컬럼 이름은 변수 이름으로 컬럼의 이름을 사용하여 XQuery 표현식으로 내재적으로 전달됩니다. 지정된 xquery-argument의 *identifier*가 범위 내 컬럼 이름과 일치하는 경우 명시적 xquery-argument가 XQuery 표현식에 전달되어 해당되는 내재된 컬럼을 대체합니다.

BY REF

디폴트 전달 메커니즘이 XML 데이터 유형의 *xquery-variable-expression*에 대한 참조에 의한 메커니즘을 지정합니다. 참조에 의해 XML 값이 패스되면 XQuery 평가는 지정된 입력 표현식으로부터 직접 입력 노드 트리(있는 경우)를 사용하여 원본 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보유합니다. 두 개의 인수가 동일한 XML 값을 패스할 경우, 두 입력 인수 사이에 들

어 있는 일부 노드가 관련된 노드 ID 비교 및 문서 순서화 비교는 동일한 XML 노드 트리에 있는 노드를 참조할 수 있습니다.

이 절은 비XML 값이 패스되는 방법에 아무 영향을 주지 않습니다. 비XML 값은 XML로 캐스트되는 동안 값의 사본을 새로 작성합니다.

xquery-argument

*xquery-expression-constant*에서 지정된 XQuery 표현식으로 패스되는 인수를 지정합니다. 인수는 값과 값을 전달할 방법을 지정합니다. 인수에는 평가되는 SQL 표현식이 포함됩니다.

- 결과 값이 XML 유형인 경우 이는 *input-xml-value*로 됩니다. 널(NULL) XML 값은 빈 XML 시퀀스로 변환됩니다.
- 결과 값이 XML 유형이 아닌 경우 이는 XML 데이터 유형으로 캐스트될 수 있어야 합니다. 널(NULL) 값은 빈 XML 시퀀스로 변환됩니다. 변환된 값은 *input-xml-value*로 됩니다.

*xquery-expression-constant*를 평가할 때 XQuery 변수는 *input-xml-value*와 같은 값 및 AS절에서 지정된 이름으로 표시됩니다.

xquery-variable-expression

실행 중 *xquery-expression-constant*에서 지정된 XQuery 표현식에 사용할 수 있는 값을 갖는 SQL 표현식을 지정합니다. 이 표현식은 시퀀스 참조(SQLSTATE 428F9) 또는 OLAP 함수(SQLSTATE 42903)를 포함할 수 없습니다.표현식의 데이터 유형은 DECFLOAT가 될 수 없습니다.

AS *identifier*

*xquery-variable-expression*에서 생성된 값이 XQuery 변수인 *xquery-expression-constant*로 패스되는 것을 지정합니다. 변수 이름은 *identifier*입니다. *identifier*에는 XQuery 언어에서 변수 이름 앞에 오는 달러 부호($)가 포함되지 않습니다. 이 ID는 유효 XQuery 변수 이름이어야 하며 XML NCName으로 제한됩니다. ID의 길이는 128바이트를 초과해서는 안됩니다. 동일한 PASSING절에서 두 개의 인수가 동일한 ID를 사용할 수 없습니다(SQLSTATE 42711).

BY REF

XML 입력 값이 참조에 의해 패스되는 것을 표시합니다. 참조에 의해 XML 값이 패스되면 XQuery 평가는 지정된 입력 표현식으로부터 직접 입력 노드 트리(있는 경우)를 사용하여 원본 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보유합니다. 두 개의 인수가 동일한 XML 값을 패스할 경우, 두 입력 인수 사이에 들어 있는 일부 노드가 관련된 노드 ID 비교 및 문서 순서화 비교는 동일한 XML 노드 트리에 있는 노드를 참조할 수 있습니다. *xquery-variable-expression* 다음에 BY REF를 지정하지 않으면 XML 인수는 PASSING 키워드 다음에 오는 구문을 통해 제공된 디폴트 전달 메

커니즘 방식으로 패스됩니다. 비XML 값에는 이 옵션을 지정할 수 없습니다. 비XML 값이 패스될 때 이 값은 XML로 변환되며 이 프로세스는 사본을 작성합니다.

주

다음의 경우 XMLEXISTS 술어를 사용할 수 없습니다.

- JOIN 연산자 또는 MERGE문과 연관된 ON절에 포함된 경우(SQLSTATE 42972)
- CREATE INDEX EXTENSION문의 GENERATE KEY USING 또는 RANGE THROUGH절에 포함된 경우(SQLSTATE 428E3)
- CREATE FUNCTION(외부 스칼라)문의 FILTER USING절 또는 CREATE INDEX EXTENSION문의 FILTER USING절에 포함된 경우(SQLSTATE 428E4)
- 점검 제한조건 또는 컬럼 생성 표현식에 포함된 경우(SQLSTATE 42621)
- group-by-clause에 포함된 경우(SQLSTATE 42822)
- column-function에 대한 인수에 포함된 경우(SQLSTATE 42607)

서브쿼리와 관련된 XMLEXISTS 술어는 서브쿼리를 제한하는 명령문에 의해 제한될 수 있습니다.

예 :

```
SELECT c.cid FROM customer c
  WHERE XMLEXISTS('$d/*:customerinfo/*:addr[ *:city = "Aurora" ]'
    PASSING info AS "d")
```

제 3 장 함수

함수 개요

*함수*는 괄호 쌍으로 묶인 인수 스펙(인수가 없을 수도 있음)이 다음에 오는 함수 이름으로 표시되는 연산입니다.

*내장 함수*는 데이터베이스 관리 프로그램과 함께 제공되며, 하나의 결과 값을 리턴하고 SYSIBM 스키마의 일부로 식별됩니다. 내장 함수에는 컬럼 함수(예: AVG), 연산자 함수(예: 『+』), 캐스트 함수(예: DECIMAL) 및 기타 함수(예: SUBSTR)가 포함됩니다.

*사용자 정의 함수(UDF)*는 CREATE FUNCTION 문을 사용하여 SYSCAT.ROUTINES에서 데이터베이스에 등록됩니다. 사용자 정의 함수는 SYSIBM 스키마의 일부가 되지 않습니다. 이와 같은 한 세트의 함수가 SYSFUN 스키마에서, 다른 세트가 SYSPROC 스키마에서 데이터베이스 관리 프로그램과 함께 제공됩니다.

함수는 집계(컬럼) 함수, 스칼라 함수, 행 함수 또는 테이블 함수로 분류됩니다.

- *집계 함수*의 인수는 유사한 값의 콜렉션입니다. 집계 함수는 단일 값(널(NULL)도 가능)을 리턴하며 표현식을 사용할 수 있는 모든 SQL문에 지정될 수 있습니다.
- *스칼라 함수*의 인수는 다른 유형이 될 수 있고 다른 의미를 가질 수 있는 개별적인 스칼라 값입니다. 스칼라 함수는 단일 값(널(NULL)이 가능)을 리턴하며 표현식을 사용할 수 있는 모든 SQL문에 지정될 수 있습니다.
- *행 함수*의 인수는 구조화된 유형입니다. 행 함수는 내장 데이터 유형의 행을 리턴하며 구조화된 유형의 변환 함수로 지정될 수 있습니다.
- *테이블 함수*의 인수는 다른 유형이 될 수 있고 다른 의미를 가질 수 있는 개별적인 스칼라 값입니다. 테이블 함수는 테이블을 SQL문으로 리턴하며 SELECT문의 FROM절 내에서만 지정할 수 있습니다.

스키마와 조인된 함수 이름은 함수의 전체 이름을 제공합니다. 스키마, 함수 이름 및 입력 매개변수의 조합이 *함수 시그니처*를 구성합니다.

어떤 경우에는 입력 매개변수 유형은 특정의 내장 데이터 유형으로 지정되고, 다른 경우에는 *any-numeric-type*과 같은 일반 변수를 통해 지정됩니다. 특정의 데이터 유형을 지정하는 경우, 지정된 데이터 유형에 대해서만 정확한 일치가 발생합니다. 일반 변수를 사용하는 경우, 해당 변수와 연관되는 데이터 유형 각각에 정확한 일치가 발생합니다.

사용자 정의 함수를 다른 스키마에서 작성할 수 있으므로, 소스로 함수 시그니처 중 하나를 사용하여 추가 함수를 사용할 수 있습니다. 또한 사용자 응용프로그램에서 외부 함수를 작성할 수도 있습니다.

지원되는 함수 및 관리 SQL 루틴 및 뷰

이 주제에는 유형별로 분류된 지원되는 내장 함수가 나열되어 있습니다.

- 집계 함수(표 31)
- 배열 함수(335 페이지의 표 32)
- 캐스트 스칼라 함수(336 페이지의 표 33)
- 날짜 시간 스칼라 함수(337 페이지의 표 34)
- 기타 스칼라 함수(338 페이지의 표 35)
- 숫자 스칼라 함수(339 페이지의 표 36)
- 파티션 스칼라 함수(340 페이지의 표 37)
- 보안 스칼라 함수(340 페이지의 표 38)
- 문자열 스칼라 함수(341 페이지의 표 39)
- 테이블 함수(342 페이지의 표 40)
- XML 함수(343 페이지의 표 41)

기능별로 분류된 지원되는 관리 SQL 루틴 및 뷰 목록은 *관리 루틴 및 뷰*의 『지원되는 관리 SQL 루틴 및 뷰』를 참조하십시오. 해당 루틴 및 뷰는 다음과 같이 그룹화됩니다.

- 활동 모니터 관리 SQL 루틴
- ADMIN_CMD 스토어드 프로시저 및 연관된 관리 SQL 루틴
- 구성 관리 SQL 루틴 및 뷰
- 환경 관리 뷰
- Health 스냅샷 관리 SQL 루틴
- MQSeries® 관리 SQL 루틴
- 보안 관리 SQL 루틴 및 뷰
- 스냅샷 관리 SQL 루틴 및 뷰
- SQL 프로시저 관리 SQL 루틴
- Stepwise 재분배 관리 SQL 루틴
- 스토리지 관리 도구 관리 SQL 루틴
- 기타 관리 SQL 루틴 및 뷰

표 31. 집계 함수

함수	설명
345 페이지의 『ARRAY_AGG』	요소 세트를 배열로 집계합니다.
347 페이지의 『AVG』	숫자 세트의 평균을 리턴합니다.
349 페이지의 『CORRELATION』	숫자 쌍 세트의 상관 계수를 리턴합니다.
350 페이지의 『COUNT』	행이나 값 세트에 행이나 값의 수를 리턴합니다.

표 31. 집계 함수 (계속)

함수	설명
352 페이지의『COUNT_BIG』	행이나 값 세트에 행이나 값의 수를 리턴합니다. 결과는 INTEGER의 최대값보다 클 수 있습니다.
354 페이지의『COVARIANCE』	숫자 쌍 세트의 편차를 리턴합니다.
355 페이지의『GROUPING』	그룹화 세트가 생성한 부분합계(subtotal) 행을 가리키는 데 그룹화 세트와 슈퍼 그룹이 사용됩니다. 리턴되는 값은 0 또는 1입니다. 값이 1이면 리턴된 행에서 인수의 값이 널(NULL) 값이고 행이 그룹화 세트에 대해 생성되었음을 의미합니다. 생성된 행은 그룹화 세트에 대한 부분합계(subtotal)를 제공합니다.
357 페이지의『MAX』	값 세트에서 최대값을 리턴합니다.
359 페이지의『MIN』	값 세트에서 최소값을 리턴합니다.
360 페이지의『Regression 함수』	REGR_AVGX 집계 함수는 진단 통계를 계산하는 데 사용되는 수량을 리턴합니다.
360 페이지의『Regression 함수』	REGR_AVGY 집계 함수는 진단 통계를 계산하는 데 사용되는 수량을 리턴합니다.
360 페이지의『Regression 함수』	REGR_COUNT 집계 함수는 회귀 행을 맞추는 데 사용된 널(NULL)이 아닌 숫자 쌍의 수를 리턴합니다.
360 페이지의『Regression 함수』	REGR_INTERCEPT 또는 REGR_ICPT 집계 함수는 회귀 행의 y 절편을 리턴합니다.
360 페이지의『Regression 함수』	REGR_R2 집계 함수는 회귀에 대한 결정 계수를 리턴합니다.
360 페이지의『Regression 함수』	REGR_SLOPE 집계 함수는 행의 기울기를 리턴합니다.
360 페이지의『Regression 함수』	REGR_SXX 집계 함수는 진단 통계를 계산하는 데 사용되는 수량을 리턴합니다.
360 페이지의『Regression 함수』	REGR_SXY 집계 함수는 진단 통계를 계산하는 데 사용되는 수량을 리턴합니다.
360 페이지의『Regression 함수』	REGR_SYY 집계 함수는 진단 통계를 계산하는 데 사용되는 수량을 리턴합니다.
363 페이지의『STDDEV』	숫자 세트의 표준 편차를 리턴합니다.
364 페이지의『SUM』	숫자 세트의 합을 리턴합니다.
365 페이지의『VARIANCE』	숫자 세트의 분산을 리턴합니다.
366 페이지의『XMLAGG』	XML 값 세트에 있는 각 널(NULL)이 아닌 값에 대한 항목을 포함하는 XML 시퀀스를 리턴합니다.
368 페이지의『XMLGROUP』	하나의 최상위 레벨 요소 노드를 포함하는 단일 XQuery 문서 노드가 있는 XML 값을 리턴합니다.

표 32. 배열 함수

함수	설명
345 페이지의『ARRAY_AGG』	요소 세트를 배열로 집계합니다.
376 페이지의『ARRAY_DELETE』	연관된 배열에서 요소 또는 요소의 범위를 삭제합니다.
378 페이지의『ARRAY_FIRST』	배열의 최소 배열 인덱스 값을 리턴합니다.
379 페이지의『ARRAY_LAST』	배열의 최대 배열 인덱스 값을 리턴합니다.
380 페이지의『ARRAY_NEXT』	지정된 배열 인덱스 인수에 상대적인 배열에 대해 다음으로 큰 배열 인덱스 값을 리턴합니다.

표 32. 배열 함수 (계속)

함수	설명
381 페이지의 『ARRAY_PRIOR』	지정된 배열 인덱스 인수에 상대적인 배열에 대해 다음으로 작은 배열 인덱스 값을 리턴합니다.
393 페이지의 『CARDINALITY』	배열의 요소 수를 나타내는 BIGINT 유형 값을 리턴합니다.
524 페이지의 『MAX_CARDINALITY』	배열이 포함할 수 있는 최대 요소 수를 나타내는 BIGINT 유형 값을 리턴합니다.
647 페이지의 『TRIM_ARRAY』	*array-variable*과 동일한 배열 유형의 값을 리턴하지만 카디널리티(cardinality)는 *numeric-expression* 값만큼 감소됩니다.
736 페이지의 『UNNEST』	지정된 배열의 각 요소에 대한 행을 포함한 결과 테이블을 리턴합니다.

표 33. 캐스트 스칼라 함수

함수	설명
387 페이지의 『BIGINT』	정수 상수의 양식으로 값의 64비트 정수 표현을 리턴합니다.
392 페이지의 『BLOB』	모든 문자열 유형의 BLOB 표현을 리턴합니다.
395 페이지의 『CHAR』	값의 CHARACTER 표현을 리턴합니다.
405 페이지의 『CLOB』	값의 CLOB 표현을 리턴합니다.
418 페이지의 『DATE』	값에서 DATE를 리턴합니다.
427 페이지의 『DBCLOB』	문자열의 DBCLOB 표현을 리턴합니다.
430 페이지의 『DECFLOAT』	값의 10진 부동 소수점 표현을 리턴합니다.
435 페이지의 『DECIMAL 또는 DEC』	값의 DECIMAL 표현을 리턴합니다.
449 페이지의 『DOUBLE_PRECISION 또는 DOUBLE』	값의 부동 소수점 표현을 리턴합니다.
EMPTY_BLOB, EMPTY_CLOB 및 EMPTY_DBCLOB 스칼라 함수	연관된 데이터 유형의 0 길이 값을 리턴합니다.
460 페이지의 『FLOAT』	값의 FLOAT 표현을 리턴합니다.
465 페이지의 『GRAPHIC』	문자열의 GRAPHIC 표현을 리턴합니다.
489 페이지의 『INTEGER 또는 INT』	값의 INTEGER 표현을 리턴합니다.
563 페이지의 『REAL』	값의 단정밀도 부동 소수점 표현을 리턴합니다.
602 페이지의 『SMALLINT』	값의 SMALLINT 표현을 리턴합니다.
622 페이지의 『TIME』	값에서 TIME을 리턴합니다.
623 페이지의 『TIMESTAMP』	하나의 값 또는 값 쌍에서 TIMESTAMP를 리턴합니다.
636 페이지의 『TO_CLOB』	문자열 유형의 CLOB 표현을 리턴합니다.
663 페이지의 『VARCHAR』	값의 VARCHAR 표현을 리턴합니다.
679 페이지의 『VARGRAPHIC』	값의 VARGRAPHIC 표현을 리턴합니다.

표 34. 날짜/시간 스칼라 함수

함수	설명
374 페이지의 『ADD_MONTHS』	*expression* + 지정된 월 수를 나타내는 날짜 시간 값을 리턴합니다.
420 페이지의 『DAY』	값의 일 부분을 리턴합니다.
421 페이지의 『DAYNAME』	locale-name 또는 CURRENT LOCALE LC_TIME 특수 레지스터 값을 기반으로 표현식의 일 부분에 대한 일 이름(예: 금요일)이 포함된 문자열을 리턴합니다.
423 페이지의 『DAYOFWEEK』	값에서 요일을 리턴합니다. 여기서, 1은 일요일이며 7은 토요일입니다.
424 페이지의 『DAYOFWEEK_ISO』	값에서 요일을 리턴합니다. 여기서, 1은 월요일이며 7은 일요일입니다.
425 페이지의 『DAYOFYEAR』	값에서 일을 리턴합니다.
426 페이지의 『DAYS』	날짜의 정수 표현을 리턴합니다.
457 페이지의 『EXTRACT』	인수를 기반으로 날짜 또는 시간소인 부분을 리턴합니다.
476 페이지의 『HOUR』	값의 시간 부분을 리턴합니다.
492 페이지의 『JULIAN_DAY』	B.C. 4712년 1월 1일에서 인수에 지정된 날짜까지의 일 수를 나타내는 정수 값을 리턴합니다.
493 페이지의 『LAST_DAY』	인수의 월의 마지막 날을 나타내는 날짜 시간 값을 리턴합니다.
525 페이지의 『MICROSECOND』	값의 마이크로초 부분을 리턴합니다.
526 페이지의 『MIDNIGHT_SECONDS』	자정과 지정된 시간 값 사이의 시간(초)를 나타내는 정수 값을 리턴합니다.
529 페이지의 『MINUTE』	값의 분 부분을 리턴합니다.
531 페이지의 『MONTH』	값의 월 부분을 리턴합니다.
532 페이지의 『MONTHNAME』	locale-name 또는 CURRENT LOCALE LC_TIME 특수 레지스터 값을 기반으로 표현식의 월 부분에 대한 월 이름(예: 1월)이 포함된 문자열을 리턴합니다.
534 페이지의 『MONTHS_BETWEEN』	*expression1* 및 *expression2* 사이의 월 수 예상값을 리턴합니다.
538 페이지의 『NEXT_DAY』	*string-expression*이라는, *expression*의 날짜보다 이후인 첫 번째 요일을 나타내는 날짜 시간 값을 리턴합니다.
558 페이지의 『QUARTER』	날짜가 포함된 분기를 표시하는 정수를 리턴합니다.
580 페이지의 『ROUND』	*format-string*에서 지정한 단위로 반올림된 날짜 시간 값을 리턴합니다.
587 페이지의 『ROUND_TIMESTAMP』	*format-string*으로 지정된 단위로 반올림된 *expression*인 시간소인을 리턴합니다.
597 페이지의 『SECOND』	값의 초 부분을 리턴합니다.
625 페이지의 『TIMESTAMP_FORMAT』	형식 템플리트(*argument2*)를 사용하여 해석된 문자열(*argument1*)에서 시간소인을 리턴합니다.
632 페이지의 『TIMESTAMP_ISO』	날짜, 시간 또는 시간소인 인수에 기초화 시간소인 값을 리턴합니다. 인수가 날짜이면 모든 시간 요소에 대해 0을 삽입합니다. 인수가 시간이면 날짜 요소에 대해 CURRENT DATE 값을 삽입하고 분할 시간 요소에 대해 0을 삽입합니다.

표 34. 날짜/시간 스칼라 함수 (계속)

함수	설명
633 페이지의 『TIMESTAMPDIFF』	두 시간소인 사이의 차이에 근거하여 *argument1* 유형의 계산된 간격 수를 리턴합니다. 두 번째 인수는 두 개의 시간소인 유형을 뺀 결과를 CHAR로 변환한 결과입니다.
635 페이지의 『TO_CHAR』	시간소인의 CHARACTER 표현을 리턴합니다.
637 페이지의 『TO_DATE』	문자열에서 시간소인을 리턴합니다.
639 페이지의 『TO_TIMESTAMP』	지정된 형식을 사용하여 입력 문자열 해석을 기반으로 하는 시간소인을 리턴합니다.
650 페이지의 『TRUNCATE 또는 TRUNC』	*format-string*에서 지정한 단위로 절단된 날짜 시간 값을 리턴합니다.
648 페이지의 『TRUNC_TIMESTAMP』	*format-string*으로 지정된 단위로 절단된 *expression*인 시간소인을 리턴합니다.
670 페이지의 『VARCHAR_FORMAT』	템플리트(*argument2*)에 따라 형식화된 시간소인(*argument1*)의 CHARACTER 표현을 리턴합니다.
685 페이지의 『WEEK』	값에서 주(일요일부터 시작)를 리턴합니다.
686 페이지의 『WEEK_ISO』	값에서 주(월요일부터 시작)를 리턴합니다.
733 페이지의 『YEAR』	값의 연도 부분을 리턴합니다.

표 35. 기타 스칼라 함수

함수	설명
389 페이지의 『BITAND, BITANDNOT, BITOR, BITXOR 및 BITNOT』	이러한 비트 방식 함수는 입력 인수 정수값의 ″2의 보수″ 표현으로 작동하며 결과를 입력 인수의 데이터 유형을 기반으로 데이터 유형에 있는 대응하는 기본 10 정수값으로 리턴합니다.
406 페이지의 『COALESCE』	널(NULL)이 아닌 첫 번째 인수를 리턴합니다.
416 페이지의 『CURSOR_ROWCOUNT』	커서가 열린 이후로 지정된 커서에 의해 페치되는 모든 누적 행 수를 리턴합니다.
440 페이지의 『DECODE』	지정된 각 *expression2*를 *expression1*과 비교합니다. *expression1*이 *expression2*와 같거나 *expression1*과 *expression2*가 둘 다 널(NULL)이면, 다음 *result-expresssion* 값이 리턴됩니다. *expression2*가 *expression1*과 일치하지 않으면, *else-expression* 값이 리턴됩니다. 그렇지 않으면, 널(NULL) 값이 리턴됩니다.
446 페이지의 『DEREF』	참조 유형 인수의 목표 유형 인스턴스를 리턴합니다.
455 페이지의 『EVENT_MON_STATE』	특정 이벤트 모니터의 작동 상태를 리턴합니다.
471 페이지의 『GREATEST』	값 세트에서 최대값을 리턴합니다.
474 페이지의 『HEX』	값의 16진수 표현을 리턴합니다.
477 페이지의 『IDENTITY_VAL_LOCAL』	식별 컬럼에 대해 가장 최근에 지정된 값을 리턴합니다.
496 페이지의 『LEAST』	값 세트에서 최소값을 리턴합니다.
501 페이지의 『LENGTH』	값의 길이를 리턴합니다.
523 페이지의 『MAX』	값 세트에서 최대값을 리턴합니다.
528 페이지의 『MIN』	값 세트에서 최소값을 리턴합니다.
541 페이지의 『NULLIF』	인수가 같을 경우에는 널(NULL) 값을, 같지 않을 경우에는 첫 번째 인수의 값을 리턴합니다.

표 35. 기타 스칼라 함수 (계속)

함수	설명
542 페이지의『NVL』	널(NULL)이 아닌 첫 번째 인수를 리턴합니다.
560 페이지의『RAISE_ERROR』	SQLCA에서 오류를 발생시킵니다. 리턴된 sqlstate는 *argument1*에 의해 표시됩니다. 두 번째 인수에 리턴되는 모든 텍스트가 포함됩니다.
565 페이지의『REC2XML』	XML 태그로 형식화되고 컬럼 이름과 컬럼 데이터가 들어 있는 문자열을 리턴합니다.
574 페이지의『RID_BIT 및 RID』	RID_BIT 스칼라 함수는 행의 행 ID(RID)를 문자열 형식으로 리턴합니다. RID 스칼라 함수는 행의 RID를 대형 정수 형식으로 리턴합니다. RID 함수는 파티션된 데이터베이스 환경에서는 지원되지 않습니다. RID_BIT 함수는 RID 함수보다 선호됩니다.
615 페이지의『TABLE_NAME』	*argument1*에서 지정된 오브젝트 이름 및 *argument2*에서 지정된 선택적 스키마 이름을 근거로 뷰 또는 테이블의 규정되지 않은 이름을 리턴합니다. 리턴된 값은 별명 분석에 사용됩니다.
617 페이지의『TABLE_SCHEMA』	두 파트 테이블의 스키마 이름 부분 또는 뷰 이름 부분(*argument1*의 오브젝트 이름 및 *argument2*의 선택적 스키마 이름이 제공)을 리턴합니다. 리턴된 값은 별명 분석에 사용됩니다.
654 페이지의『TYPE_ID』	인수에서 동적 데이터 유형의 내부 데이터 유형 ID를 리턴합니다. 이 함수의 결과는 데이터베이스 간에 호환이 되지 않습니다.
655 페이지의『TYPE_NAME』	인수의 동적 데이터 유형의 규정화되지 않은 이름을 리턴합니다.
656 페이지의『TYPE_SCHEMA』	인수의 동적 데이터 유형의 스키마 이름을 리턴합니다.
662 페이지의『VALUE』	널(NULL)이 아닌 첫 번째 인수를 리턴합니다.

표 36. 숫자 스칼라 함수

함수	설명
372 페이지의『ABS 또는 ABSVAL』	숫자의 절대값을 리턴합니다.
373 페이지의『ACOS』	숫자의 arc cosine을 라디안으로 리턴합니다.
383 페이지의『ASIN』	숫자의 arc sine을 라디안으로 리턴합니다.
384 페이지의『ATAN』	숫자의 arc tangent를 라디안으로 리턴합니다.
386 페이지의『ATANH』	숫자의 hyperbolic arc tangent를 라디안으로 리턴합니다.
385 페이지의『ATAN2』	xy 좌표의 arc tangent를 라디안으로 표시되는 각도로 리턴합니다.
394 페이지의『CEILING 또는 CEIL』	숫자보다 크거나 같은 최소의 정수 값을 리턴합니다.
409 페이지의『COMPARE_DECFLOAT』	두 인수가 같은지 무순인지 여부, 또는 한 인수가 다른 인수보다 큰지 여부를 표시하는 SMALLINT 값을 리턴합니다.
413 페이지의『COS』	숫자의 cosine을 리턴합니다.
414 페이지의『COSH』	숫자의 hyperbolic cosine을 리턴합니다.
415 페이지의『COT』	인수의 cotangent를 리턴합니다. 여기서, 인수는 라디안으로 표시되는 각도입니다.
432 페이지의『DECFLOAT_FORMAT』	문자열에서 DECFLOAT(34)를 리턴합니다.
445 페이지의『DEGREES』	각도의 숫자를 리턴합니다.

표 36. 숫자 스칼라 함수 (계속)

함수	설명
448 페이지의 『DIGITS』	숫자 절대값의 문자열 표현을 리턴합니다.
456 페이지의 『EXP』	인수가 지정한 제곱으로 올릴 자연 대수의 기본 값을 리턴합니다.
461 페이지의 『FLOOR』	숫자보다 크거나 작은 최대의 정수 값을 리턴합니다.
504 페이지의 『LN』	숫자의 자연 대수를 리턴합니다.
513 페이지의 『LOG10』	숫자의 일반 대수(기본 10)를 리턴합니다.
530 페이지의 『MOD』	첫 번째 인수를 두 번째 인수로 나눈 나머지를 리턴합니다.
536 페이지의 『MULTIPLY_ALT』	두 인수의 곱을 10진수 값으로 리턴합니다. 이 함수는 인수 정밀도의 합이 31보다 클 경우에 유용합니다.
540 페이지의 『NORMALIZE_DECFLOAT』	가장 단순한 양식으로 설정된 인수의 결과인 10진 부동 소수점 값을 리턴합니다.
555 페이지의 『POWER』	첫 번째 인수를 두 번째 인수의 제곱으로 올린 결과를 리턴합니다.
556 페이지의 『QUANTIZE』	값이 같고 첫 번째 인수에 대한 기호이며 지수가 두 번째 인수의 지수와 같은 10진 부동 소수점 숫자를 리턴합니다.
559 페이지의 『RADIANS』	각도로 표시된 인수에 대해 라디안 수를 리턴합니다.
562 페이지의 『RAND』	난수를 리턴합니다.
580 페이지의 『ROUND』	10진수 위치의 지정된 숫자로 반올림된 숫자 값을 리턴합니다.
599 페이지의 『SIGN』	숫자의 부호를 리턴합니다.
600 페이지의 『SIN』	숫자의 sine을 리턴합니다.
601 페이지의 『SINH』	숫자의 hyperbolic sine을 리턴합니다.
606 페이지의 『SQRT』	숫자의 제곱근을 리턴합니다.
620 페이지의 『TAN』	숫자의 tangent를 리턴합니다.
621 페이지의 『TANH』	숫자의 hyperbolic tangent를 리턴합니다.
638 페이지의 『TO_NUMBER』	문자열에서 DECFLOAT(34)를 리턴합니다.
640 페이지의 『TOTALORDER』	두 인수의 비교 순서를 표시하는 SMALLINT 값(-1, 0 또는 1)을 리턴합니다.
650 페이지의 『TRUNCATE 또는 TRUNC』	10진수 위치의 지정된 숫자에서 절단된 숫자 값을 리턴합니다.

표 37. 파티션 스칼라 함수

함수	설명
417 페이지의 『DATAPARTITIONNUM』	행이 있는 데이터 파티션의 시퀀스 번호 (SYSDATAPARTITIONS.SEQNO)를 리턴합니다. 인수는 테이블 내의 컬럼 이름입니다.
428 페이지의 『DBPARTITIONNUM』	행의 데이터베이스 파티션 번호를 리턴합니다. 인수는 테이블 내의 컬럼 이름입니다.
472 페이지의 『HASHEDVALUE』	행의 분산 맵 인덱스(0 - 32767)를 리턴합니다. 인수는 테이블 내의 컬럼 이름입니다.

표 38. 보안 스칼라 함수

함수	설명
593 페이지의 『SECLABEL』	이름 지정되지 않은 보안 레이블을 리턴합니다.

표 38. 보안 스칼라 함수 (계속)

함수	설명
594 페이지의 『SECLABEL_BY_NAME』	특정 보안 레이블을 리턴합니다.
595 페이지의 『SECLABEL_TO_CHAR』	보안 레이블을 승인하고 보안 레이블의 모든 요소를 포함하는 문자열을 리턴합니다.

표 39. 문자열 스칼라 함수

함수	설명
382 페이지의 『ASCII』	인수에서 가장 왼쪽 문자의 ASCII 코드 값을 정수로 리턴합니다.
402 페이지의 『CHARACTER_LENGTH』	지정된 *string-unit*로 된 표현식의 길이를 리턴합니다.
404 페이지의 『CHR』	인수에서 지정한 ASCII 코드 값을 갖고 있는 문자를 리턴합니다.
407 페이지의 『COLLATION_KEY_BIT』	지정된 *collation-name*에 지정된 *string-expression*의 조합 키를 나타내는 VARCHAR FOR BIT DATA 문자열을 리턴합니다.
411 페이지의 『CONCAT』	두 문자열의 병합인 문자열을 리턴합니다.
442 페이지의 『DECRYPT_BIN 및 DECRYPT_CHAR』	암호 문자열을 사용하여 암호화된 데이터 해독의 결과 값을 리턴합니다.
447 페이지의 『DIFFERENCE』	SOUNDEX 함수로 판별된 대로 두 개의 인수 문자열에 있는 단어의 음조 간 차이를 리턴합니다. 값 4는 문자열 음조가 같음을 의미합니다.
452 페이지의 『ENCRYPT』	데이터 문자열 표현식을 암호화한 결과 값을 리턴합니다.
462 페이지의 『GENERATE_UNIQUE』	동일한 함수의 다른 모든 실행과 비교하여 고유한 비트 데이터 문자열을 리턴합니다.
464 페이지의 『GETHINT』	암호 힌트가 있을 경우 암호 힌트를 리턴합니다.
482 페이지의 『INITCAP』	각 단어의 첫 번째 문자가 대문자로 변환되고 나머지는 소문자로 변환된 문자열을 리턴합니다.
484 페이지의 『INSERT』	문자열을 리턴합니다. 여기서, *argument3*바이트는 *argument1*(*argument2*에서 시작)에서 삭제되었으며 *argument4*는 *argument1*(*argument2*에서 시작)으로 삽입되었습니다.
488 페이지의 『INSTR』	다른 문자열 내에서 문자열의 시작 위치를 리턴합니다.
494 페이지의 『LCASE』	모든 SBCS 문자가 소문자로 변환된 문자열을 리턴합니다.
495 페이지의 『LCASE(로케일 구분)』	지정된 로케일과 연관된 유니코드 표준의 규칙을 사용하여 모든 문자가 소문자로 변환된 문자열을 리턴합니다.
517 페이지의 『LOWER(로케일 구분)』	지정된 로케일과 연관된 유니코드 표준의 규칙을 사용하여 모든 문자가 소문자로 변환된 문자열을 리턴합니다.
497 페이지의 『LEFT』	문자열에서 가장 왼쪽의 문자를 리턴합니다.
505 페이지의 『LOCATE』	다른 문자열 내에서 한 문자열의 시작 위치를 리턴합니다.
509 페이지의 『LOCATE_IN_STRING』	다른 문자열 내에서 한 문자열이 처음 나타나는 시작 위치를 리턴합니다.
516 페이지의 『LOWER』	모든 문자가 소문자로 변환된 문자열을 리턴합니다.
519 페이지의 『LPAD』	지정된 문자 또는 공백으로 왼쪽이 채워진 문자열을 리턴합니다.
522 페이지의 『LTRIM』	문자열 표현식의 시작에서 공백을 제거합니다.
543 페이지의 『OCTET_LENGTH』	표현식의 길이를 8진수(바이트)로 리턴합니다.

표 39. 문자열 스칼라 함수 (계속)

함수	설명
544 페이지의 『OVERLAY』	지정된 *source-string*의 *start*에서 시작, 지정된 코드 단위의 *length*가 삭제되고 *insert-string*이 삽입된 문자열을 리턴합니다.
549 페이지의 『POSITION』	*argument1* 내에서 *argument2*의 시작 위치를 리턴합니다.
552 페이지의 『POSSTR』	다른 문자열 내에서 한 문자열의 시작 위치를 리턴합니다.
570 페이지의 『REPEAT』	*argument2*의 배수만큼 반복되는 *argument1*로 구성된 문자열을 리턴합니다.
571 페이지의 『REPLACE』	*argument1*에서 *argument2*의 모든 어커런스를 *argument3*으로 대체합니다.
576 페이지의 『RIGHT』	문자열에서 가장 오른쪽의 문자를 리턴합니다.
589 페이지의 『RPAD』	지정된 문자, 문자열 또는 공백으로 오른쪽이 채워진 문자열을 리턴합니다.
592 페이지의 『RTRIM』	문자열 표현식의 끝에서 공백을 제거합니다.
604 페이지의 『SOUNDEX』	인수에서 단어들의 음조를 나타내는 4문자 코드를 리턴합니다. 이 결과는 다른 문자열의 음조와 비교될 수 있습니다.
605 페이지의 『SPACE』	지정된 수의 공백으로 구성된 문자열을 리턴합니다.
607 페이지의 『STRIP』	문자열 표현식에서 선두 또는 후미 공백이나 기타 지정된 선두 또는 후미 문자를 제거합니다.
609 페이지의 『SUBSTR』	문자열의 하위 문자열을 리턴합니다.
612 페이지의 『SUBSTRING』	문자열의 하위 문자열을 리턴합니다.
642 페이지의 『TRANSLATE』	문자열의 하나 이상의 문자가 다른 문자로 변환된 문자열을 리턴합니다.
645 페이지의 『TRIM』	문자열 표현식에서 선두 또는 후미 공백이나 기타 지정된 선두 또는 후미 문자를 제거합니다.
657 페이지의 『UCASE』	UCASE 함수는 첫 번째 인수(*char-string-exp*)만 지정된다는 점을 제외하고 TRANSLATE 함수와 동일합니다.
658 페이지의 『UCASE(로케일 구분)』	지정된 로케일과 연관된 유니코드 표준의 규칙을 사용하여 모든 문자가 대문자로 변환된 문자열을 리턴합니다.
659 페이지의 『UPPER』	모든 문자가 대문자로 변환된 문자열을 리턴합니다.
660 페이지의 『UPPER(로케일 구분)』	지정된 로케일과 연관된 유니코드 표준의 규칙을 사용하여 모든 문자가 대문자로 변환된 문자열을 리턴합니다.

표 40. 테이블 함수

함수	설명
734 페이지의 『BASE_TABLE』	별명 체인이 분석된 이후에 발견된 오브젝트의 오브젝트 이름 및 스키마 이름을 리턴합니다.
736 페이지의 『UNNEST』	지정된 배열의 각 요소에 대한 행을 포함한 결과 테이블을 리턴합니다.
739 페이지의 『XMLTABLE』	지정된 입력 인수를 XQuery 변수로 사용하여 XQuery 표현식 평가로부터 테이블을 리턴합니다. 행 XQuery 표현식의 결과 시퀀스에서 각 시퀀스 항목은 결과 테이블의 행을 나타냅니다.

표 41. XML 함수

함수	설명
548 페이지의『PARAMETER』	db2-fn:sqlquery 함수 호출의 일부로 XQuery에 의해 동적으로 값이 제공되는 SQL문의 위치를 나타냅니다.
366 페이지의『XMLAGG』	XML 값 세트에 있는 각 널(NULL)이 아닌 값에 대한 항목을 포함하는 XML 시퀀스를 리턴합니다.
687 페이지의『XMLATTRIBUTES』	인수의 XML 속성을 작성합니다.
689 페이지의『XMLCOMMENT』	입력 인수를 내용으로 하는 단일 XQuery 주석 노드와 함께 XML 값을 리턴합니다.
690 페이지의『XMLCONCAT』	XML 입력 인수의 변수 번호 조합을 포함하는 시퀀스를 리턴합니다.
692 페이지의『XMLDOCUMENT』	하위 노드가 0개 이상인 단일 XQuery 문서 노드와 함께 XML 값을 리턴합니다.
694 페이지의『XMLELEMENT』	XML 요소 노드인 XML 값을 리턴합니다.
700 페이지의『XMLFOREST』	XML 요소 노드 시퀀스인 XML 값을 리턴합니다.
368 페이지의『XMLGROUP』	하나의 최상위 레벨 요소 노드를 포함하는 단일 XQuery 문서 노드가 있는 XML 값을 리턴합니다.
704 페이지의『XMLNAMESPACES』	인수로부터 이름 스페이스 선언을 작성합니다.
707 페이지의『XMLPARSE』	인수를 XML 문서로 구문 분석한 후 XML 값을 리턴합니다.
710 페이지의『XMLPI』	단일 XQuery 처리 명령어 노드와 함께 XML 값을 리턴합니다.
712 페이지의『XMLQUERY』	지정된 입력 인수를 XQuery 변수로 사용하여 XQuery 표현식 평가로부터 XML 값을 리턴합니다.
716 페이지의『XMLROW』	하나의 최상위 레벨 요소 노드를 포함하는 단일 XQuery 문서 노드가 있는 XML 값을 리턴합니다.
719 페이지의『XMLSERIALIZE』	인수로부터 생성되는 지정된 데이터 유형의 직렬화된 XML 값을 리턴합니다.
739 페이지의『XMLTABLE』	지정된 입력 인수를 XQuery 변수로 사용하여 XQuery 표현식 평가로부터 테이블을 리턴합니다. 행 XQuery 표현식의 결과 시퀀스에서 각 시퀀스 항목은 결과 테이블의 행을 나타냅니다.
721 페이지의『XMLTEXT』	입력 인수를 내용으로 하는 단일 XQuery 텍스트 노드와 함께 XML 값을 리턴합니다.
723 페이지의『XMLVALIDATE』	디폴트값을 포함하여 XML 스키마 유효성 확인에서 확보한 정보에 의해 증가된 입력 XML 값의 사본을 리턴합니다.
728 페이지의『XMLXSROBJECTID』	인수에 지정된 XML 문서의 유효성을 확인하는 데 사용되는 XML 스키마의 XSR 오브젝트 ID를 리턴합니다.
729 페이지의『XSLTRANSFORM』	한 XML 스키마를 준수하는 XML 문서를 다른 스키마를 준수하는 문서로 변환하는 것을 포함하여 XML 데이터를 기타 형식으로 변환합니다.

집계 함수

집계 함수 인수는 표현식에서 파생된 값 세트입니다. 표현식에 컬럼은 포함되지만 *scalar-fullselect*, 다른 집계 함수나, XMLQUERY 또는 XMLEXISTS 표현식(SQLSTATE 42607)은 포함되지 않습니다. 세트의 범위는 그룹 또는 중간 결과 테이블입니다.

GROUP BY 절이 쿼리에 지정되고 FROM, WHERE, GROUP BY 및 HAVING절의 중간 결과가 빈 세트인 경우, 집계 함수가 적용되지 않고 쿼리 결과는 빈 세트이며 SQLCODE는 +100으로 설정되고 SQLSTATE는 '02000'으로 설정됩니다.

GROUP BY 절이 쿼리에 지정되지 않고 FROM, WHERE 및 HAVING절의 중간 결과가 빈 세트이면 집계 함수는 빈 세트에 적용됩니다.

예를 들어, 다음 SELECT문의 결과는 부서 D01에 있는 직원들에 대한 JOBCODE 값의 수입니다.

```
SELECT COUNT(DISTINCT JOBCODE)
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

DISTINCT 키워드는 함수의 인수가 되는 것으로 간주되지 않고 함수가 적용되기 전에 수행되는 조작의 스펙이 되는 것으로 간주됩니다. DISTINCT가 지정되면 중복되는 값은 제거됩니다. 수치상으로 동등한 10진수 부동 소수점 값에 대해 DISTINCT 절을 해석하는 경우 값의 유효 숫자 수가 고려되지 않습니다. 예를 들어 10진수 부동 소수점 숫자 123.00은 10진수 부동 소수점 숫자 123과 구별되지 않습니다. 쿼리에서 리턴된 수 표시는 발생한 임의 표현이 됩니다(예: 123.00 또는 123).

ALL이 내재적이나 명시적으로 지정되면 중복되는 값은 제거되지 않습니다.

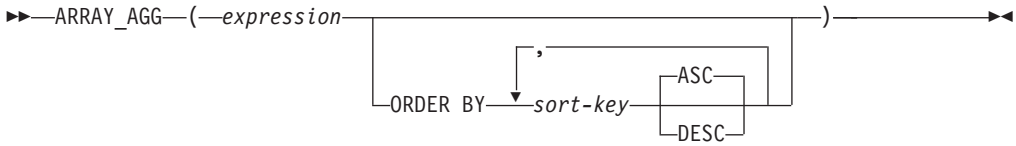
다른 SQL 구현과의 호환성을 위해 집계 함수에서 UNIQUE를 DISTINCT의 동의어로 지정할 수 있습니다.

집계 함수에서 표현식을 사용할 수 있습니다. 예를 들면, 다음과 같습니다.

```
SELECT MAX(BONUS + 1000)
  INTO :TOP_SALESREP_BONUS
  FROM EMPLOYEE
  WHERE COMM > 5000
```

집계 함수는 스키마 이름으로 규정될 수 있습니다(예: SYSIBM.COUNT(*)).

ARRAY_AGG



스키마는 SYSIBM입니다.

ARRAY_AGG 함수는 요소 세트를 배열로 집계합니다. 표현식의 데이터 유형은 CREATE TYPE(배열)문에 지정될 수 있는 데이터 유형이어야 합니다(SQLSTATE 429C2).

*sort-key*를 지정한 경우 배열에서 집계 요소의 순서를 결정합니다. *sort-key*를 지정하지 않은 경우 배열 내에서의 요소 순서 지정은 결정적이지 않습니다. *sort-key*를 지정하지 않았지만 ARRAY_AGG가 동일한 SELECT 절에서 두 번 이상 지정된 경우, 배열 내에서의 동일한 순서 지정이 ARRAY_AGG의 각 결과에 사용됩니다.

SELECT 절에 *sort-key*를 지정하는 XMLAGG 또는 ARRAY_AGG의 여러 어커런스가 있는 경우, 모든 정렬 키는 동일해야 합니다(SQLSTATE 428GZ).

ARRAY_AGG 함수는 다음 특정 컨텍스트에서 SQL 프로시저 내에서만 지정할 수 있습니다(SQLSTATE 42887).

- SELECT INTO문의 선택 목록
- 스크롤할 수 없는 커서 정의에서 fullselect의 선택 목록
- SET문의 오른쪽에 있는 스칼라 서브쿼리의 선택 목록

ARRAY_AGG는 OLAP 함수의 일부로 사용될 수 없습니다(SQLSTATE 42887). ARRAY_AGG를 사용하는 SELECT문은 ORDER BY절이나 DISTINCT절을 포함할 수 없으며, SELECT 또는 HAVING절은 서브쿼리를 포함하거나 SQL 함수를 호출할 수 없습니다(SQLSTATE 42887).

ARRAY_AGG는 연관된 배열 또는 행 요소 데이터 유형을 사용한 배열을 생성하는 데 사용될 수 없습니다(SQLSTATE 42846).

예:

- 다음 DDL이 제공된 경우:

```
CREATE TYPE PHONELIST AS DECIMAL(10, 0)ARRAY[10]

CREATE TABLE EMPLOYEE (
  ID          INTEGER NOT NULL,
  PRIORITY    INTEGER NOT NULL,
  PHONENUMBER DECIMAL(10, 0),
  PRIMARY KEY(ID, PRIORITY))
```

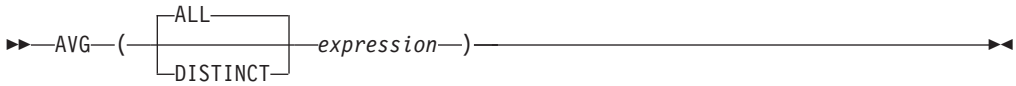
SELECT INTO문을 사용하여 사원이 도달할 수 있는 우선순위화된 문의처 번호를 리턴하는 프로시저를 작성하십시오.

```
CREATE PROCEDURE GETPHONENUMBERS
  (IN  EMPID   INTEGER,
   OUT NUMBERS PHONELIST)
BEGIN
  SELECT ARRAY_AGG(PHONENUMBER ORDER BY PRIORITY)
    INTO NUMBERS
  FROM EMPLOYEE
  WHERE ID = EMPID;
END
```

SET문을 사용하여 사원의 문의처 번호 목록을 임의 순서로 리턴하는 프로시저를 작성하십시오.

```
CREATE PROCEDURE GETPHONENUMBERS
   (IN  EMPID   INTEGER,
    OUT NUMBERS PHONELIST)
BEGIN
  SET NUMBERS =
    (SELECT ARRAY_AGG(PHONENUMBER)
    FROM EMPLOYEE
    WHERE ID = EMPID);
END
```

AVG



스키마는 SYSIBM입니다.

AVG 함수는 숫자 세트의 평균을 리턴합니다.

인수 값은 숫자이어야 하고(내장 유형 전용) 10진수 결과 데이터 유형을 제외한 합은 결과의 데이터 유형 범위 내에 있어야 합니다. 10진수 결과의 경우 그 합이 정밀도가 31이고 인수 값의 스케일과 동일한 스케일을 갖는 10진수 데이터에 의해 지원되는 범위 내에 있어야 합니다. 결과는 널(NULL)이 될 수 있습니다.

결과의 데이터 유형은 다음 경우를 제외하고 인수 값의 데이터 유형과 같습니다.

- 인수 값이 작은 정수(small integer)일 경우 결과는 큰 정수(large integer)입니다.
- 인수 값이 단정밀도의 부동 소수점일 경우 결과는 배정밀도의 부동 소수점입니다.
- 인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(34)입니다.

인수 값의 데이터 유형이 정밀도가 *p*이고 스케일이 *s*인 10진수인 경우 결과의 정밀도는 31이고 스케일은 31-*p*+*s*입니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. DISTINCT가 지정되는 경우 여분의 중복 값은 제거됩니다. 수적으로 동등한 10진수 부동 소수점 값에 대한 DISTINCT절을 해석할 때 값의 유효 숫자 수는 고려되지 않습니다. 예를 들어, 10진수 부동 소수점 숫자 123.00은 10진수 부동 소수점 숫자 123과 구별되지 않습니다. 쿼리에서 리턴된 숫자 표현은 직면한 표현 중 하나가 됩니다(예를 들어, 123.00 또는 123).

함수가 공집합에 적용되는 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 그 결과는 세트의 평균 값입니다.

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

결과의 유형이 정수인 경우 평균의 소수 부분은 유실됩니다.

예를 들면, 다음과 같습니다.

- PROJECT 테이블을 사용하여 호스트 변수 AVERAGE(decimal(5,2))를 부서(DEPTNO) 'D11'의 프로젝트에 대한 평균 직원 레벨(PRSTAFF)로 설정하십시오.

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
   FROM PROJECT
  WHERE DEPTNO = 'D11'
```

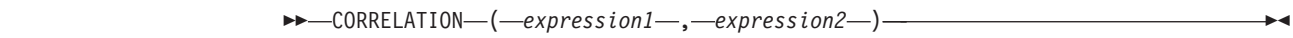
샘플 테이블을 사용할 때 AVERAGE 결과는 4.25(즉 17/4)로 설정됩니다.

- PROJECT 테이블을 사용하여 호스트 변수 ANY_CALC(decimal(5,2))를 부서(DEPTNO) 'D11'의 프로젝트에 대한 각 고유 직원 레벨 값(PRSTAFF) 평균으로 설정하십시오.

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
   FROM PROJECT
  WHERE DEPTNO = 'D11'
```

샘플 테이블을 사용할 때 ANY_CALC 결과는 4.66(즉, 14/3)으로 설정됩니다.

CORRELATION



스키마는 SYSIBM입니다.

CORRELATION 함수는 숫자 쌍 세트의 상관 계수를 리턴합니다.

인수 값은 숫자이어야 합니다.

인수가 10진수 부동 소수점이면 결과는 DECFLOAT(34)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과는 널(NULL)이 될 수 있습니다. 널(NULL)이 아니면 결과는 -1과 1 사이입니다.

함수는 *expression1* 또는 e*xpression2*가 널(NULL)인 모든 쌍을 제거하여 인수 값에서 파생된 (*expression1*, *expression2*) 쌍의 세트에 적용됩니다.

함수가 빈 세트에 적용되거나 STDDEV(*expression1*) 또는 STDDEV(*expression2*)가 0인 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 결과는 세트에 있는 값 쌍에 대한 상관 계수입니다. 결과는 다음 표현식에 해당합니다.

```
COVARIANCE(expression1,expression2)/
(STDDEV(expression1)*
STDDEV(expression2))
```

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

CORRELATION 대신 CORR을 지정할 수 있습니다.

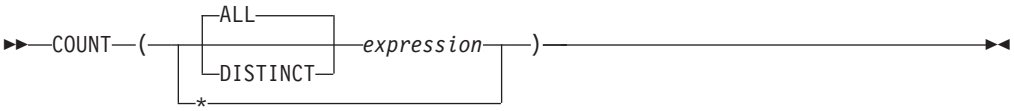
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 CORRLN(배정밀도의 부동 소수점)를 부서(WORKDEPT) ‘A00’에 있는 직원의 급여와 보너스 사이의 상관으로 설정하십시오.

  ```
  SELECT CORRELATION(SALARY, BONUS)
    INTO :CORRLN
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  샘플 테이블을 사용할 때 CORRLN는 대략 9.99853953399538E-001로 설정됩니다.

COUNT



스키마는 SYSIBM입니다.

COUNT 함수는 행 또는 값 세트에서 행 수 또는 값 수를 리턴합니다.

DISTINCT가 지정되면 *expression*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, XML, 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 될 수 없습니다(SQLSTATE 42907). 그렇지 않으면 *expression*의 결과 데이터 유형은 어떤 데이터 유형도 될 수 있습니다.

함수의 결과는 정수(integer)입니다. 결과는 널(NULL)이 될 수 없습니다.

COUNT(*)의 인수는 행 세트입니다. 결과는 세트 내의 행 수입니다. NULL 값만 포함하는 행이 계수에 포함됩니다.

COUNT(DISTINCT *expression*)의 인수는 값 세트입니다. 함수는 널(NULL) 및 중복 값이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. 결과는 세트 내의 서로 다른 널(NULL)이 아닌 값 수입니다.

COUNT(*expression*) 또는 COUNT(ALL *expression*)의 인수는 값 세트입니다. 함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. 결과는 중복을 포함하여 세트에 있는 널(NULL)이 아닌 값 수입니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블을 사용하여 호스트 변수 FEMALE(int)을 SEX 컬럼 값이 'F'인 행 수로 설정하십시오.

  ```
  SELECT COUNT(*)
    INTO :FEMALE
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

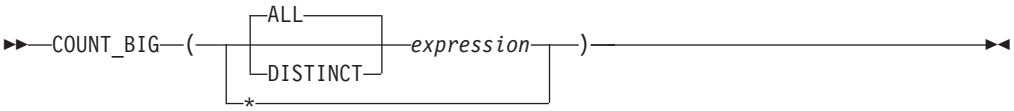
  샘플 테이블을 사용할 때 FEMALE의 결과는 13으로 설정됩니다.

- EMPLOYEE 테이블을 사용하여 호스트 변수 FEMALE_IN_DEPT(int)를 구성원으로 한 명 이상의 여성이 있는 부서(WORKDEPT) 수로 설정하십시오.

  ```
  SELECT COUNT(DISTINCT WORKDEPT)
    INTO :FEMALE_IN_DEPT
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

샘플 테이블을 사용할 때 FEMALE_IN_DEPT의 결과는 5로 설정됩니다. (부서 A00, C01, D11, D21, E11에 한 명 이상의 여성이 있습니다.)

COUNT_BIG



스키마는 SYSIBM입니다.

COUNT_BIG 함수는 행 또는 값 세트에서 행 수 또는 값 수를 리턴합니다. 결과가 정수의 최대값보다 클 수 있다는 점을 제외하고 COUNT와 유사합니다.

DISTINCT가 지정되면 *expression*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, XML, 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 될 수 없습니다(SQLSTATE 42907). 그렇지 않으면 *expression*의 결과 데이터 유형은 어떤 데이터 유형도 될 수 있습니다.

함수의 결과는 정밀도가 31이고 스케일이 0인 10진수입니다. 결과는 널(NULL)이 될 수 없습니다.

COUNT_BIG(*)의 인수는 행 세트입니다. 결과는 세트 내의 행 수입니다. NULL 값만 포함하는 행이 계수에 포함됩니다.

COUNT_BIG(DISTINCT *expression*)의 인수는 값 세트입니다. 함수는 널(NULL) 및 중복 값이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. 결과는 세트 내의 서로 다른 널(NULL)이 아닌 값 수입니다.

COUNT_BIG(*expression*) 또는 COUNT_BIG(ALL *expression*)의 인수는 값 세트입니다. 함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. 결과는 중복을 포함하여 세트에 있는 널(NULL)이 아닌 값 수입니다.

예를 들면, 다음과 같습니다.

- COUNT 예를 참조하고 COUNT 발생을 COUNT_BIG로 대체하십시오. 결과는 결과의 데이터 유형을 제외하고 동일합니다.
- 일부 응용프로그램에서는 COUNT를 사용할 수도 있지만 가장 큰 정수보다 큰 값을 지원해야 합니다. 이는 전래 사용자 정의 함수(UDF)를 사용하고 SQL 경로를 설정하여 달성할 수 있습니다. 다음 일련의 명령문은 COUNT_BIG을 기초로 COUNT(*)를 지원하고 정밀도 15의 10진수 값을 리턴하는 전래 함수를 작성하는 방법을 나타냅니다. SQL 경로는 COUNT_BIG을 기초로 한 전래 함수가 표시된 쿼리와 같은 연속 명령문에서 사용되도록 설정됩니다.

```
CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)
  SOURCE SYSIBM.COUNT_BIG();
SET CURRENT PATH RICK, SYSTEM PATH;
SELECT COUNT(*) FROM EMPLOYEE;
```

COUNT(*)를 지원하기 위한 매개변수 없이 전래 함수를 정의하는 방법에 유의하십시오. 이는 사용자가 함수에 COUNT 이름을 지정하고 사용 시 스키마 이름으로 함수를 규정하지 않는 경우에만 작동합니다. COUNT가 아닌 다른 이름으로 COUNT(*)와 같은 효과를 가져오려면, 매개변수 없이 함수를 호출하십시오. 따라서, RICK.COUNT가 대신 RICK.MYCOUNT로 정의된 경우 쿼리는 다음과 같이 작성해야 합니다.

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

특정 컬럼에서 계수를 가져오는 경우, 전래 함수는 컬럼 유형을 지정해야 합니다. 다음 명령문은 인수로 CHAR 컬럼을 가져오고 카운팅을 수행하기 위해 COUNT_BIG을 사용할 전래 함수를 작성했습니다.

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE SYSIBM.COUNT_BIG(CHAR());
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE

스키마는 SYSIBM입니다.

COVARIANCE 함수는 숫자 쌍 세트의 (데이터 채우기) 공분산을 리턴합니다.

인수 값은 숫자이어야 합니다.

인수가 10진수 부동 소수점이면 결과는 DECFLOAT(34)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 *expression1* 또는 e*xpression2*가 널(NULL)인 모든 쌍을 제거하여 인수 값에서 파생된 (*expression1*, *expression2*) 쌍의 세트에 적용됩니다.

함수가 공집합에 적용되는 경우 결과는 널(NULL)이 됩니다. 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.

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

COVARIANCE 대신 COVAR을 지정할 수 있습니다.

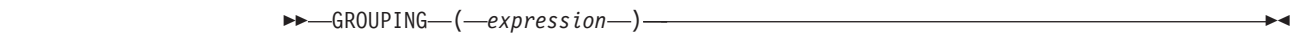
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 COVARNCE(배정밀도의 부동 소수점)를 부서(WORKDEPT) 'A00'에 있는 직원의 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 세트 및 수퍼 그룹과 함께 사용되는 GROUPING 함수는 GROUP BY 응답 세트로 리턴된 행이 *expression*에 표시된 컬럼을 제외하는 그룹화 세트에 의해 생성된 행인지 여부를 표시하는 값을 리턴합니다.

인수는 어떤 유형도 될 수 있지만 GROUP BY 절의 항목이어야 합니다.

함수의 결과는 작은 정수입니다. 다음 값 중 하나로 설정됩니다.

1 리턴된 행에 있는 *expression* 값이 널(NULL) 값이고 행이 수퍼 그룹에 의해 생성되었습니다. 이 생성된 행을 사용하여 GROUP BY 표현식에 대한 부분합 값을 제공할 수 있습니다.

0 위의 값 이외의 값입니다.

예:

다음 쿼리에서

```
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
```

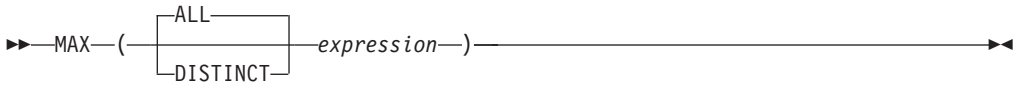
결과는 다음과 같습니다.

```
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 1이라는 사실에 의해 SALES_DATE 부분합 행을 인식할 수 있습니다. SALES_PERSON 부분합 행은 DATE_GROUP의 값이 1이고 SALES_GROUP의 값이 0이라는 사실에 의해 인식될 수 있습니다. 총계 행은 DATE_GROUP 및 SALES_GROUP 둘 다에 대해 값 1에 의해 인식될 수 있습니다.

MAX



스키마는 SYSIBM입니다.

MAX 함수는 값 세트에서 최대값을 리턴합니다.

인수값은 LOB 문자열을 제외한 내장 유형이 될 수 있습니다.

*expression*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB와 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 될 수 없습니다(SQLSTATE 42907).

결과의 데이터 유형, 길이 및 코드 페이지는 인수 값의 데이터 유형, 길이 및 코드 페이지와 같습니다. 결과는 파생된 값으로 간주되고 널(NULL)이 될 수 있습니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다.

함수가 공집합에 적용되는 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 결과는 세트 내에서 최대값이 됩니다.

DISTINCT를 지정해도 결과에는 아무런 영향도 주지 않으므로 사용하지 않는 것이 좋습니다. 이것은 다른 관계형 시스템과의 호환성을 위해 포함됩니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블을 사용하여 호스트 변수 MAX_SALARY(decimal(7,2))를 최대 월 급여(SALARY/12) 값으로 설정하십시오.

  ```
  SELECT MAX(SALARY) / 12
    INTO :MAX_SALARY
    FROM EMPLOYEE
  ```

  샘플 테이블을 사용할 때 MAX_SALARY가 4395.83으로 설정됩니다.

- PROJECT 테이블을 사용하여 호스트 변수 LAST_PROJ(char(24))를 조합 시퀀스에서 마지막으로 제공되는 프로젝트 이름(PROJNAME)으로 설정하십시오.

  ```
  SELECT MAX(PROJNAME)
    INTO :LAST_PROJ
     FROM PROJECT
  ```

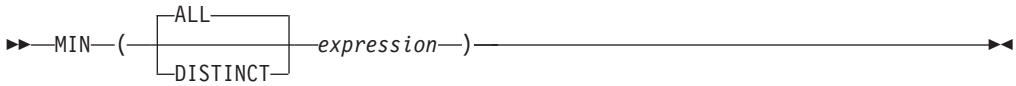
  샘플 테이블을 사용할 때 LAST_PROJ는 'WELD LINE PLANNING'으로 설정됩니다.

- 이전 예와 마찬가지로 호스트 변수 LAST_PROJ(char(40))를 프로젝트 이름이 호스트 변수 PROJSUPP와 병합될 때 조합 시퀀스에서 마지막으로 제공되는 프로젝트 이름(PROJNAME)으로 설정하십시오. PROJSUPP는 '_Support'이고 데이터 유형은 char(8)입니다.

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)
  INTO :LAST_PROJ
   FROM PROJECT
```

  샘플 테이블을 사용할 때 LAST_PROJ는 'WELD LINE PLANNING_SUPPORT'로 설정됩니다.

MIN



MIN 함수는 값 세트에서 최소값을 리턴합니다.

인수값은 LOB 문자열을 제외한 내장 유형이 될 수 있습니다.

*expression*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, 이들 유형의 구별 유형 또는 구조화된 유형(SQLSTATE 42907)일 수 없습니다.

결과의 데이터 유형, 길이 및 코드 페이지는 인수 값의 데이터 유형, 길이 및 코드 페이지와 같습니다. 결과는 파생된 값으로 간주되고 널(NULL)이 될 수 있습니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다.

이 함수가 빈 세트에 적용되는 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 결과는 세트에서 최소값이 됩니다.

DISTINCT를 지정해도 결과에는 아무런 영향도 주지 않으므로 사용하지 않는 것이 좋습니다. 이것은 다른 관계형 시스템과의 호환성을 위해 포함됩니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블을 사용하여 호스트 변수 COMM_SPREAD(decimal(7,2))를 부서 (WORKDEPT) 'D11' 구성원에 대한 최대 및 최소 수당(COMM) 사이의 차이로 설정하십시오.

  ```
  SELECT MAX(COMM) - MIN(COMM)
    INTO :COMM_SPREAD
    FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
  ```

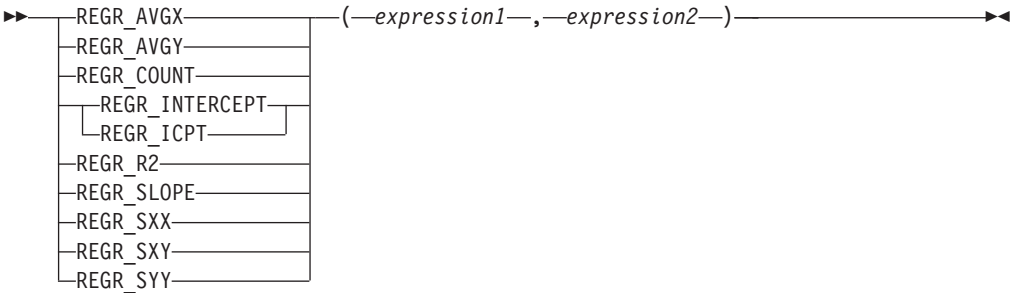
  샘플 테이블을 사용할 때 COMM_SPREAD가 1118(즉, 2580 - 1462)로 설정됩니다.

- PROJECT 테이블을 사용하여 호스트 변수 FIRST_FINISHED(char(10))가 처음으로 완료되도록 스케줄된 프로젝트의 예상 종료 날짜(PRENDATE)로 설정하십시오.

  ```
  SELECT MIN(PRENDATE)
    INTO :FIRST_FINISHED
     FROM PROJECT
  ```

  샘플 테이블을 사용할 때 FIRST_FINISHED는 '1982-09-15'로 설정됩니다.

Regression 함수



스키마는 SYSIBM입니다.

REGRESSION 함수는 y = a * x + b 양식의 ordinary-least-squares 회귀 행을 일련의 숫자 쌍으로 맞추는 것을 지원합니다. 각 쌍의 첫 번째 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.

REGR_COUNT 함수는 회귀 행을 맞추는 데 사용된 널(NULL)이 아닌 숫자 쌍의 수를 리턴합니다(아래 참조).

REGR_INTERCEPT(또는 REGR_ICPT) 함수는 회귀 행의 y 절편을 리턴합니다(위의 등식에서 ″b″).

REGR_R2 함수는 회귀에 대한 결정 계수(″R-squared″ 또는 ″goodness-of-fit″)를 리턴합니다.

REGR_SLOPE 함수는 행의 기울기(위의 등식에서 ″a″)를 리턴합니다.

REGR_AVGX, REGR_AVGY, REGR_SXX, REGR_SXY 및 REGR_SYY는 회귀 모델의 등록 정보 및 통계 유효성의 평가에 필요한 여러 가지 진단 통계를 계산하는 데 사용할 수 있는 수량을 리턴합니다(아래 참조).

인수 값은 숫자이어야 합니다.

REGR_COUNT 결과의 데이터 유형은 정수입니다. 남아 있는 함수의 경우 인수가 DECFLOAT(*n*)이면 결과의 데이터 유형은 DECFLOAT(34)입니다. 기타의 경우 결과의 데이터 유형은 배정밀도 부동 소수점입니다. 인수가 특수 10진수 부동 소수점 값인 경우 10진수 부동 소수점에 대해 일반 산술 연산의 규칙이 적용됩니다. 243 페이지의 『10진수 부동 소수점에 대한 일반 산술 연산 규칙』에서 『10진수 부동 소수점에 대한 일반 산술 연산 규칙』을 참조하십시오.

결과는 널(NULL)이 될 수 있습니다. 널(NULL)이 아니면 REGR_R2의 결과는 0과 1사이이며 REGR_SXX와 REGR_SYY 둘 모두의 결과는 음이 아닌 정수입니다.

각 함수는 *expression1* 또는 e*xpression2*가 널(NULL)인 모든 쌍이 제거된 인수 값에서 파생된 (*expression1*, *expression2*) 쌍의 세트에 적용됩니다.

세트가 비워 있지 않고 VARIANCE(*expression2*)가 양수인 경우 REGR_COUNT가 세트에서 널(NULL)이 아닌 숫자 쌍을 리턴하며 나머지 함수는 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*)

세트가 비어 있지 않고 VARIANCE(*expression2*)가 0인 경우 회귀 행이 무한 기울기를 갖거나 정의되지 않습니다. 이 경우 함수 REGR_SLOPE, REGR_INTERCEPT 및 REGR_R2는 각각 널(NULL) 값을 리턴하며 나머지 함수는 위에서 정의된 값을 리턴합니다. 세트가 비어 있는 경우 REGR_COUNT가 0을 리턴하며 나머지 함수들은 널(NULL) 값을 리턴합니다.

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

REGRESSION 함수는 데이터를 한 개 통과하는 동안 모두 동시에 계산됩니다. 일반적으로 AVERAGE, VARIANCE, COVARIANCE 등과 같은 일반 컬럼 함수를 사용하여 해당 계산을 수행하는 것보다 회귀 분석에 필요한 통계를 계산하기 위해 REGRESSION 함수를 사용하는 것이 더 효율적입니다.

위의 함수로 선형 회귀 분석을 하는 일반적인 진단 통계를 계산할 수 있습니다. 예를 들면, 다음과 같습니다.

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))

제곱의 총 합계
REGR_SYY

제곱의 회귀 합계
POWER(REGR_SXY,2) / REGR_SXX

제곱의 나머지 합계
(제곱의 총 합계) - (제곱의 회귀 합계)

기울기에 대한 t STATISTIC
REGR_SLOPE * SQRT(REGR_SXX) / (표준 오류)

y 절편에 대한 t STATISTIC
REGR_INTERCEPT / ((Standard error) * SQRT((1/REGR_COUNT) + (POWER(REGR_AVGX,2) / REGR_SXX))

예:

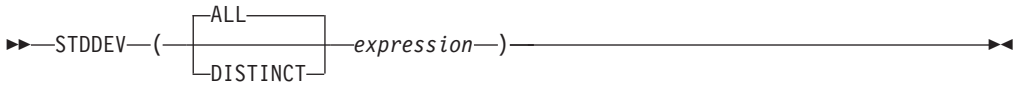
- EMPLOYEE 테이블을 사용하여 부서 (WORKDEPT) 'A00'에 있는 직원의 보너스를 직원 급여의 선형 함수로서 표현하는 ordinary-least-squares 회귀 행을 계산하십시오. 호스트 변수 SLOPE, ICPT, RSQR(배정밀도 부동 소수점)을 각각 회귀 행 결정의 기울기, 절편 및 계수로 설정하십시오. 또한 호스트 변수 AVGSAL 및 AVGBONUS를 각각 부서 'A00'에 있는 직원의 평균 급여와 평균 보너스로 설정하고, 호스트 변수 CNT(정수)를 급여와 보너스 데이터가 모두 사용 가능한 부서 'A00'에 있는 직원 번호로 설정하십시오. 나머지 회귀 통계를 호스트 변수 SXX, SYY 및 SXY에 저장하십시오.

```
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 값으로 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 편차(/*n*)를 리턴합니다. STDDEV를 계산하기 위해 사용되는 공식은 다음과 같습니다.

```
-   STDDEV = SQRT(VARIANCE)
```

SQRT(VARIANCE)는 분산의 제곱근입니다.

인수 값은 숫자이어야 합니다.

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. DISTINCT가 지정되는 경우 여분의 중복 값은 제거됩니다. 수적으로 동등한 10진수 부동 소수점 값에 대한 DISTINCT절을 해석할 때 값의 유효 숫자 수는 고려되지 않습니다. 예를 들어, 10진수 부동 소수점 숫자 123.00은 10진수 부동 소수점 숫자 123과 구별되지 않습니다. 쿼리에서 리턴된 숫자 표현은 직면한 표현 중 하나가 됩니다(예를 들어, 123.00 또는 123).

함수가 공집합에 적용되는 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 결과는 세트 내에 있는 값의 표준 편차입니다.

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

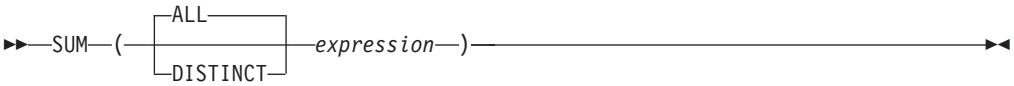
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 DEV(배정밀도의 부동 소수점)를 부서(WORKDEPT) 'A00'에 있는 직원의 급여 표준 편차로 설정하십시오.

```
SELECT STDDEV(SALARY)
  INTO :DEV
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  DEV는 9938.00의 근사값을 가진 수로 설정됩니다.

SUM



스키마는 SYSIBM입니다.

SUM 함수는 숫자 세트의 합을 리턴합니다.

인수 값은 숫자이어야 하고(내장 유형 전용) 그 합은 결과의 데이터 유형 범위 내에 있어야 합니다.

결과의 데이터 유형은 다음 경우를 제외하고 인수 값의 데이터 유형과 같습니다.

- 인수 값이 작은 정수(small integer)일 경우 결과는 큰 정수(large integer)입니다.
- 인수 값이 단정밀도의 부동 소수점일 경우 결과는 배정밀도의 부동 소수점입니다.
- 인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(34)입니다.

인수 값의 데이터 유형이 10진수인 경우 결과의 정밀도가 31이며 스케일은 인수 값의 스케일과 동일합니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. DISTINCT가 지정되면, 여분의 중복 값도 제거됩니다. 수적으로 동등한 10진수 부동 소수점 값에 대한 DISTINCT절을 해석할 때 값의 유효 숫자 수는 고려되지 않습니다. 예를 들어, 10진수 부동 소수점 숫자 123.00은 10진수 부동 소수점 숫자 123과 구별되지 않습니다. 쿼리에서 리턴된 숫자 표현은 직면한 표현 중 하나가 됩니다(예를 들어, 123.00 또는 123).

함수가 공집합에 적용되는 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 결과는 세트 내에 있는 값의 합입니다.

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

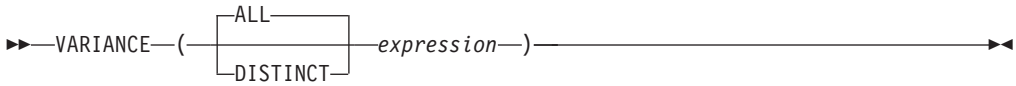
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 JOB_BONUS(decimal(9,2))를 직원(JOB='CLERK')에게 지불된 총 보너스(BONUS)로 설정하십시오.

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
  FROM EMPLOYEE
  WHERE JOB = 'CLERK'
```

  샘플 테이블을 사용할 때 JOB_BONUS 결과는 2800으로 설정됩니다.

VARIANCE



스키마는 SYSIBM입니다.

VARIANCE 함수는 숫자 세트의 분산을 리턴합니다.

인수 값은 숫자이어야 합니다.

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다. DISTINCT가 지정되는 경우 여분의 중복 값은 제거됩니다. 수적으로 동등한 10진수 부동 소수점 값에 대한 DISTINCT절을 해석할 때 값의 유효 숫자 수는 고려되지 않습니다. 예를 들어, 10진수 부동 소수점 숫자 123.00은 10진수 부동 소수점 숫자 123과 구별되지 않습니다. 쿼리에서 리턴된 숫자 표현은 직면한 표현 중 하나가 됩니다(예를 들어, 123.00 또는 123).

함수가 공집합에 적용되는 경우 결과는 널(NULL)이 됩니다. 그렇지 않으면 결과는 세트 내에 있는 값의 분산입니다.

값을 추가하는 순서는 정의되어 있지 않지만 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

VARIANCE 대신 VAR을 지정할 수 있습니다.

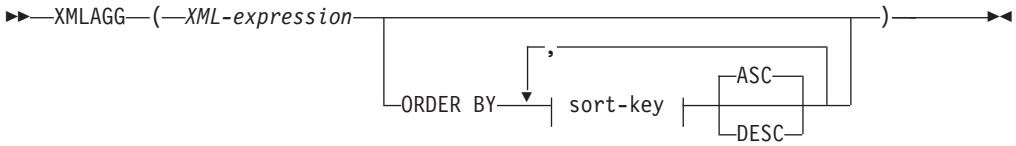
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 VARNCE(배정밀도 부동 소수점)를 부서(WORKDEPT) ‘A00’에 있는 직원의 급여 분산으로 설정하십시오.

```
SELECT VARIANCE(SALARY)
  INTO :VARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  샘플 테이블을 사용할 때 VARNCE 결과는 약 98763888.88으로 설정됩니다.

XMLAGG



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLAGG 함수는 XML 값 세트에 있는 각 널(NULL)이 아닌 값에 대한 항목을 포함하는 XML 시퀀스를 리턴합니다.

XML-expression
XML 데이터 유형의 표현식을 지정합니다.

ORDER BY
집계에서 처리되는 동일한 그룹 세트의 행 순서를 지정합니다. ORDER BY절이 생략되거나 ORDER BY절이 컬럼 데이터의 순서를 구분할 수 없는 경우, 동일한 그룹화 세트의 행은 임의적으로 순서가 지정됩니다.

sort-key
정렬 키는 컬럼 이름 또는 *sort-key-expression*일 수 있습니다. 정렬 키가 상수인 경우, 출력 컬럼의 위치를 참조하지 않으나(일반적인 ORDER BY절에서와 마찬가지로) 단순히 정렬 키가 없음을 의미하는 상수라는 점에 유의하십시오.

결과의 데이터 유형은 XML입니다.

함수는 널(NULL)이 제거된 인수 값에서 파생되는 값 세트에 적용됩니다.

XML-expression 인수가 널(NULL)일 경우, 결과는 널(NULL)이 될 수 있습니다. 값 세트가 비어 있을 경우, 결과는 널(NULL) 값이 됩니다. 그렇지 않으면 결과는 세트의 각 값에 대한 항목을 포함하는 XML 시퀀스입니다.

주:

1. **OLAP 표현식에서의 지원**: XMLAGG는 OLAP 집계 함수의 컬럼 함수로 사용할 수 없습니다(SQLSTATE 42601).

예:

주: XMLAGG은 출력에 공백 또는 줄 바꾸기 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- 성에 의해 정렬된 직원 목록을 포함하는 각각의 부서에 대한 부서 요소를 작성합니다.

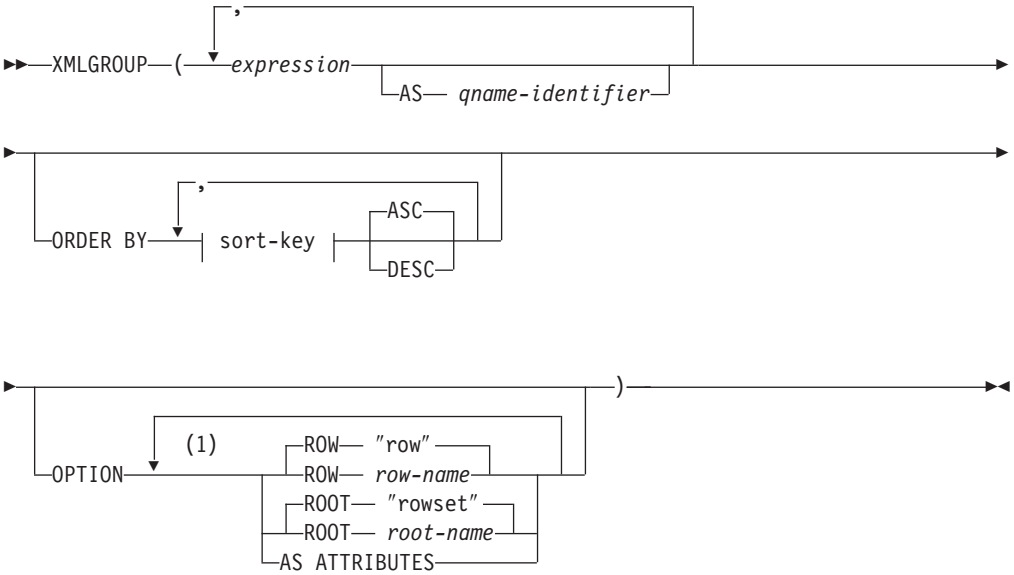
```
SELECT XMLSERIALIZE(
  CONTENT XMLELEMENT(
    NAME "Department", XMLATTRIBUTES(
      E.WORKDEPT AS "name"
    ),
    XMLAGG(
      XMLELEMENT(
        NAME "emp", E.LASTNAME
      )
      ORDER BY E.LASTNAME
    )
  )
  AS CLOB(110)
)
AS "dept_list"
FROM EMPLOYEE E
WHERE E.WORKDEPT IN ('C01','E21')
GROUP BY WORKDEPT
```

이 쿼리는 다음 결과를 생성합니다.

```
dept_list
----------------------...
<Department name="C01">
    <emp>KWAN</emp>
    <emp>NICHOLLS</emp>
    <emp>QUINTANA</emp>
</Department>
<Department name="E21">
    <emp>GOUNOT</emp>
    <emp>LEE</emp>
    <emp>MEHTA</emp>
    <emp>SPENSER</emp>
</Department>
```

XMLGROUP

XMLGROUP 함수는 하나의 최상위 레벨 요소 노드를 포함하는 단일 XQuery 문서 노드와 함께 XML 값을 리턴합니다. 이는 각각의 행이 행 하위 요소에 맵핑되는 행 그룹에서 단일 루트 XML 문서를 리턴하는 집계 표현식입니다.



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

expression

생성된 각 XML 요소 노드의 컨텐츠(또는 생성된 각 속성의 값)는 표현식에 의해 지정됩니다. *expression*의 데이터 유형은 구조화 유형이 될 수 없습니다(SQLSTATE 42884). 표현식은 임의의 SQL 표현식일 수 있습니다. 표현식이 단순 컬럼 참조가 아닌 경우, *qname-identifier*를 지정해야 합니다.

AS *qname-identifier*

XML 요소 이름 또는 속성 이름을 SQL ID로 지정합니다. *qname-identifier*는 XML 규정 이름 또는 QName 양식이어야 합니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 이름을 지정할 경우, 이름 스페이스 접두부를 범위 내에 선언해야 합니다(SQLSTATE 42635). *qname-identifier*를 지정하지 않은 경우 *expression*은 컬럼 이름이어야 합니다(SQLSTATE 42703). 요소 이름 또는 속성 이름은 컬럼 이름에서 QName으로의 완전 이스케이프 맵핑을 사용하여 컬럼 이름으로부터 작성됩니다.

OPTION

XML 값을 구성하기 위한 추가 옵션을 지정합니다. OPTION절을 지정하지 않을 경우, 디폴트 동작이 적용됩니다.

ROW *row-name*

각 행이 맵핑될 요소의 이름을 지정합니다. 이 옵션을 지정하지 않는 경우 디폴트 요소 이름은 ″row″입니다.

ROOT *root-name*

루트 요소 노드의 이름을 지정합니다. 이 옵션을 지정하지 않는 경우 디폴트 루트 요소 이름은 ″rowset″입니다.

AS ATTRIBUTES

컬럼 이름 또는 *qname-identifier*가 속성 이름으로 제공되는 속성 값에 각 표현식이 맵핑됨을 지정합니다.

ORDER BY

집계에서 처리되는 동일한 그룹 세트의 행 순서를 지정합니다. ORDER BY절이 생략되거나 ORDER BY절이 컬럼 데이터의 순서를 구분할 수 없는 경우, 동일한 그룹화 세트의 행은 임의적으로 순서가 지정됩니다.

sort-key

정렬 키는 컬럼 이름 또는 *sort-key-expression*일 수 있습니다. 정렬 키가 상수인 경우, 출력 컬럼의 위치를 참조하지 않으나(일반적인 ORDER BY절에서와 마찬가지로) 단순히 정렬 키가 없음을 의미하는 상수라는 점에 유의하십시오.

주

디폴트 동작은 결과 세트와 XML 값 사이의 단순 맵핑을 정의합니다. 함수 동작에 대해 다음과 같은 일부 추가 주의사항이 적용됩니다.

- 디폴트로, 각 행은 이름이 ″row″인 XML 요소로 변환되고 각 컬럼은 컬럼 이름이 요소 이름으로 제공되는 중첩 요소로 변환됩니다.
- 널(NULL) 조절 동작은 NULL ON NULL입니다. 컬럼의 NULL 값은 하위 요소 없음으로 맵핑됩니다. 모든 컬럼 값이 NULL이면 어떤 행 요소도 생성되지 않습니다.
- BLOB 및 FOR BIT DATA 데이터 유형에 대한 2진 인코딩 스킴은 base64Binary 인코딩입니다.
- 디폴트로, 그룹의 행에 해당되는 요소는 이름이 ″rowset″인 루트 요소의 하위 요소입니다.
- 루트 요소에서 행 하위 요소의 순서는 행이 쿼리 결과 세트에서 리턴되는 순서와 같습니다.
- 문서 노드는 내재적으로 루트 요소에 추가되어 XML 결과를 제대로 형성된 단일 루트 XML 문서로 작성합니다.

예:

관계형 형식으로 저장된 숫자 데이터를 포함하는 정수 컬럼 C1 및 C2가 있는 다음 테이블 T1을 가정합니다.

```
C1          C2
----------- -----------
          1           2
          -           2
          1           -
          -           -

  4 record(s) selected.
```

- 다음 예는 테이블을 표시하기 위해 단일 최상위 레벨 요소를 사용하고 디폴트로 동작하는 XMLGroup 쿼리 및 출력 조각을 나타냅니다. :

```
SELECT XMLGROUP(C1, C2)FROM T1
```

```
<rowset>
   <row>
      <C1>1</C1>
      <C2>2</C2>
      </row>
   <row>
      <C2>2</C2>
      </row>
   <row>
      <C1>1</C1>
      </row>
</rowset>

  1 record(s) selected.
```

- 다음 예는 속성 중심의 맵핑이 있는 XMLGroup 쿼리 및 출력 조각을 나타냅니다. 이전 예에서와 같이 중첩 요소로 표시되는 대신, 관계형 데이터가 요소 속성에 맵핑됩니다.

```
SELECT XMLGROUP(C1, C2 OPTION AS ATTRIBUTES) FROM T1
```

```
<rowset>
   <row C1="1" C2="2"/>
   <row C2="2"/>
   <row C1="1"/>
</rowset>

  1 record(s) selected.
```

- 다음 예는 디폴트 <rowset> 루트 요소가 <document>로 교체되고 디폴트 <row> 요소가 <entry>로 교체된 XMLGroup 쿼리 및 출력 조각을 나타냅니다. 컬럼 C1 및 C2는 <column1> 및 <column2> 요소로 리턴되고 리턴 세트는 컬럼 C1을 기준으로 정렬됩니다.

```
SELECT XMLGROUP(
   C1 AS "column1", C2 AS "column2"
   ORDER BY C1 OPTION ROW "entry" ROOT "document")
FROM T1
```

```
<document>
   <entry>
      <column1>1</column1>
      <column2>2</column2>
   </entry>
   <entry>
      <column1>1</column1>
   </entry>
   <entry>
      <column2>2</column2>
    </entry>
</document>
```

스칼라 함수

스칼라 함수는 표현식이 사용될 수 있는 곳이면 어디든지 사용할 수 있습니다. 그러나 표현식과 집계 함수의 사용에 적용되는 제한사항은 표현식이나 집계 함수가 스칼라 함수 내에 사용될 때도 적용됩니다. 예를 들어, 스칼라 함수의 인수는 스칼라 함수가 사용되는 컨텍스트에서 집계 함수를 허용할 경우에만 집계 함수가 될 수 있습니다.

스칼라 함수는 값 세트가 아닌 단일 값에 적용되므로 집계 함수 사용에 대한 제한사항은 스칼라 함수에 적용되지 않습니다.

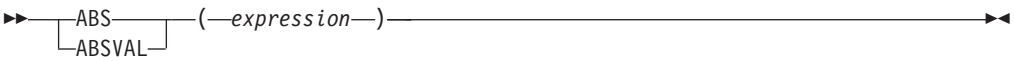
다음 SELECT문의 결과에는 부서 D01에 있는 직원 수만큼의 행이 있습니다.

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

스칼라 함수는 스키마 이름(예: SYSIBM.CHAR(123))으로 규정될 수 있습니다.

유니코드 데이터베이스에서 문자 또는 그래픽 문자열을 허용하는 모든 스칼라 함수는 변환이 지원되는 모든 문자열 유형을 허용합니다.

ABS 또는 ABSVAL



스키마는 SYSIBM입니다.

ABS(또는 ABSVAL) 함수의 SYSFUN 버전은 계속 사용 가능합니다.

인수의 절대값을 리턴합니다. 인수는 내장 숫자 데이터 유형이면 어느 것이나 가능합니다.

결과에는 인수와 동일한 데이터 유형 및 길이 속성이 있습니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다. 인수가 SMALLINT, INTEGER 또는 BIGINT에 대해 최대 음수 값이면 결과는 오버플로우 오류입니다.

주

DECFLOAT 특수값을 포함하는 결과: 10진수 부동 소수점의 경우, 특수값은 다음과 같이 처리됩니다.

- ABS(NaN) 및 ABS(-NaN)는 NaN을 리턴합니다.
- ABS(Infinity) 및 ABS(-Infinity)는 무한대를 리턴합니다.
- ABS(sNaN) 및 ABS(-sNaN)는 sNaN을 리턴합니다.

예 :

```
ABS(-51234)
```

숫자의 INTEGER 표현을 리턴합니다.

ACOS

스키마는 SYSIBM입니다. (ACOS 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 arccosine을 라디안으로 표시되는 각도로 리턴합니다.

인수는 어떤 내장 숫자 데이터 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

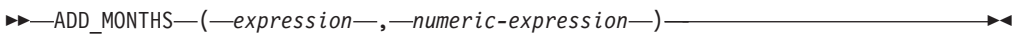
예:

호스트 변수 ACOSINE이 값이 0.070737202인 DECIMAL(10,9) 호스트 변수라고 가정해 보십시오.

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

이 명령문은 대략 1.49 값을 리턴합니다.

ADD_MONTHS



스키마는 SYSIBM입니다.

ADD_MONTHS 함수는 *expression*이 나타내는 날짜 및 시간 값 + 지정된 월 수를 리턴합니다.

expression
시작 날짜를 지정하는 표현식입니다. 표현식은 내장 데이터 유형은 날짜 또는 시간 소인 중 하나의 값을 리턴해야 합니다.

numeric-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. *numeric-expression*은 *expression*에 지정된 시작 날짜에 추가할 월 수를 지정합니다. 음수 값이 허용됩니다.

함수의 결과는 결과 데이터 유형이 DATE인 경우 *expression*이 문자열이 아니면 *expression*과 같은 데이터 유형을 갖습니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

*expression*이 해당 월의 마지막 날이거나 결과 월이 *expression*의 일(day) 구성요소보다 일 수가 적은 경우 결과는 결과 월의 마지막 날입니다. 그렇지 않으면 결과는 *expression*과 동일한 일(day) 구성요소를 갖습니다. *expression*에 포함된 시, 분, 초 또는 소수 초 정보는 함수에서 변경되지 않습니다.

예:

- 오늘이 2007년 1월 31일이라고 가정합니다. 호스트 변수 ADD_MONTH를 1월의 마지막 날 + 1달로 설정하십시오.

```
SET :ADD_MONTH = ADD_MONTHS(LAST_DAY(CURRENT_DATE), 1);
```

  호스트 변수 ADD_MONTH는 2월의 마지막 날을 나타내는 값(2007-02-28)으로 설정됩니다.

- DATE가 값이 1965년 7월 27일인 호스트 변수라고 가정하십시오. 호스트 변수 ADD_MONTH를 해당 일의 값 + 3달로 설정하십시오.

```
SET :ADD_MONTH = ADD_MONTHS(:DATE,3);
```

  호스트 변수 ADD_MONTH는 세 달이 추가된 날을 나타내는 값(1965-10-27)으로 설정됩니다.

- ADD_MONTHS 함수와 날짜 산술을 사용하여 유사한 결과를 성취할 수 있습니다. 다음 예는 유사성과 대비를 나타냅니다.

```
SET :DATEHV = DATE('2008-2-28') + 4 MONTHS;
SET :DATEHV = ADD_MONTHS('2008-2-28', 4);
```

두 경우 모두에서, 호스트 변수 DATEHV는 값 '2008–06–28'로 설정됩니다.

이제 동일한 예이지만 인수로 날짜 '2008–2–29'를 사용하는 예를 고려해 보십시오.

```
SET :DATEHV = DATE('2008-2-29') + 4 MONTHS;
```

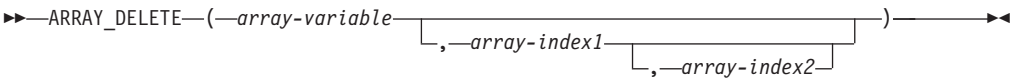
호스트 변수 DATEHV는 값 '2008–06–29'로 설정됩니다.

```
SET :DATEHV = ADD_MONTHS('2008-2-29', 4);
```

호스트 변수 DATEHV는 값 '2008–06–30'으로 설정됩니다.

이 경우, ADD_MONTHS 함수는 월의 마지막 날(2008년 6월 29일 대신 2008년 6월 30일)을 리턴합니다. 이유는 2월 29일이 월의 마지막 날이기 때문입니다. 따라서 ADD_MONTHS 함수는 6월의 마지막 날을 리턴합니다.

ARRAY_DELETE



스키마는 SYSIBM입니다.

ARRAY_DELETE 함수는 배열에서 요소를 삭제합니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

array-index1
배열 인덱스의 데이터 유형에 지정할 수 있는 값의 결과인 표현식. *array-variable*이 일반 배열이면, *array-index1*은 널(NULL) 값이어야 합니다.

array-index2
배열 인덱스의 데이터 유형에 지정할 수 있는 값의 결과인 표현식. *array-variable*이 일반 배열이면, *array-index2*는 널(NULL) 값이어야 합니다. *array-index2*가 지정되어 값이 널(NULL)이 아닌 경우 *array-index1*은 *array-index2*의 값보다 적은 널(NULL)이 아닌 값이어야 합니다(SQLSTATE42815).

함수의 결과에는 *array-variable*과 동일한 데이터 유형이 있습니다. 선택적 인수가 지정되지 않았거나 값이 널(NULL)인 경우 *array-variable*의 모든 요소가 삭제되고 결과 배열 값의 카디널리티(cardinality)는 0이 됩니다. *array-index1*만 널(NULL)이 아닌 값으로 지정된 경우 인덱스 값 *array-index1*의 배열 요소는 삭제됩니다. *array-index2*가 널(NULL)이 아닌 값으로 지정된 경우 인덱스 값 *array-index1*에서 *array-index2*까지 범위의 요소는 삭제됩니다.

결과가 널(NULL)이 될 수 있습니다. 즉, *array-variable*이 널(NULL)이면 결과는 널(NULL) 값입니다.

주

- ARRAY_DELETE 함수는 배열이 지원되는 컨텍스트에서 지정 명령문의 오른쪽에만 사용할 수 있습니다.

예

- PHONENUMBERS 배열 유형의 일반 배열 변수 RECENT_CALLS에서 모든 요소를 삭제하십시오. .

```
SETRECENT_CALLS = ARRAY_DELETE(RECENT_CALLS)
```

- 공급자가 일부 제품의 공급을 중지하였습니다. 배열 유형 PRODUCTS의 연관된 배열 변수 FLOOR_TILES에서 인덱스 값 'PK5100'에서 인덱스 값 'PS2500'의 요소를 삭제하십시오.

```
SETFLOOR_TILES = ARRAY_DELETE(FLOOR_TILES,'PK5100','PS2500')
```

ARRAY_FIRST

스키마는 SYSIBM입니다.

ARRAY_FIRST 함수는 배열의 최소 배열 인덱스 값을 리턴합니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

결과의 데이터 유형은 일반 배열에 대해서는 INTEGER인 배열 인덱스의 데이터 유형입니다. *array-variable*이 널(null)이 아니고 배열의 카디널리티(cardinality)가 0보다 큰 경우 결과 값은 최소 배열 인덱스 값으로 일반 배열의 경우 1입니다.

결과는 널(NULL)이 될 수 있으며 *array-variable*이 널(NULL)이거나 배열의 카디널리티가 0이면 결과는 널(NULL) 값입니다.

예

- 일반 배열 변수 SPECIALNUMBERS의 첫 번째 인덱스 값을 SQL 변수 E_CONSTIDX로 리턴합니다.

  ```
  SET E_CONSTIDX = ARRAY_FIRST(SPECIALNUMBERS)
  ```

  결과는 1입니다.

- 인덱스 값 및 전화번호가 'Home'은 '4163053745', 'Work'는 '4163053746'이고 'Mom'은 '416-4789683'인 연관된 배열 변수 PHONELIST에서 배열의 최소 인덱스 값을 문자열 변수 X에 지정합니다.

  ```
  SET X = ARRAY_FIRST(PHONELIST)
  ```

  'Home' 값이 X에 지정됩니다. 인덱스 값 'Home'에 연관된 요소 값에 액세스하여 이를 SQL 변수 NUMBER_TO_CALL에 지정합니다.

  ```
  SET NUMBER_TO_CALL = PHONELIST[X]
  ```

ARRAY_LAST

스키마는 SYSIBM입니다.

ARRAY_LAST 함수는 배열의 최대 배열 인덱스 값을 리턴합니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

결과의 데이터 유형은 일반 배열에 대해서는 INTEGER인 배열 인덱스의 데이터 유형입니다. *array-variable*가 널(null)이 아니고 배열의 카디널리티(cardinality)가 0보다 큰 경우 결과 값은 최대 배열 인덱스 값으로 일반 배열에 대한 배열의 카디널리티입니다.

결과가 널(NULL)이 될 수 있습니다. 배열 변수가 널(NULL)이거나 배열의 카디널리티가 영(0)이면 결과는 널(NULL) 값입니다.

예

- 일반 배열 변수 SPECIALNUMBERS의 마지막 인덱스 값을 SQL 변수 PI_CONSTIDX에 리턴합니다.

  ```
  SET PI_CONSTIDX = ARRAY_LAST(SPECIALNUMBERS)
  ```

  결과는 10입니다.

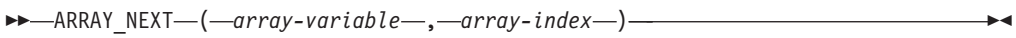
- 인덱스 값 및 전화번호가 'Home'은 '4163053745', 'Work'는 '4163053746'이고 'Mom'은 '4164789683'인 연관된 배열 변수 PHONELIST에서 배열의 최대 인덱스 값을 문자열 변수 X에 지정합니다.

  ```
  SET X = ARRAY_LAST(PHONELIST)
  ```

  'Work' 값이 X에 지정됩니다. 인덱스 값 'Work'에 연관된 요소 값에 액세스하여 이를 SQL 변수 NUMBER_TO_CALL에 지정합니다.

  ```
  SET NUMBER_TO_CALL = PHONELIST[X]
  ```

ARRAY_NEXT



스키마는 SYSIBM입니다.

ARRAY_NEXT 함수는 지정된 배열 인덱스 인수에 상대적인 배열에 대해 다음으로 큰 배열 인덱스 값을 리턴합니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

array-index
배열의 인덱스 데이터 유형에 지정할 수 있는 값을 지정합니다. 유효한 값은 데이터 유형으로 유효한 값을 포함합니다.

결과는 지정한 *array-index* 값에 연관된 배열에 정의된 다음으로 큰 배열 인덱스 값입니다. *array-index*가 배열의 최소 인덱스 배열 값보다 작은 경우 결과는 배열에 정의된 첫 번째 배열 인덱스 값입니다.

함수 결과의 데이터 유형은 배열 인덱스의 데이터 유형입니다. 결과는 널(NULL)이 될 수 있습니다. 인수가 널(NULL), 첫 번째 인수의 카디널리티(cardinality)가 0 또는 *array-index* 값이 배열의 마지막 인덱스 값 이상인 경우 결과는 널(NULL) 값입니다.

예

- 일반 배열 변수 SPECIALNUMBERS에서 9번째 인덱스 위치 뒤의 다음 인덱스 값을 SQL 변수 NEXT_CONSTIDX에 리턴합니다.

  ```
  SET NEXT_CONSTIDX = ARRAY_NEXT(SPECIALNUMBERS,9)
  ```

  결과는 10입니다.

- 인덱스 값 및 전화번호가 'Home'은 4163053745', 'Work'는 '4163053746'이며 'Mom'은 '416-4789683'인 연관된 배열 변수 PHONELIST에서 배열 값에 없는 인덱스 값 'Dad' 다음의 인덱스인 배열의 인덱스 값을 문자열 변수 X로 지정합니다.

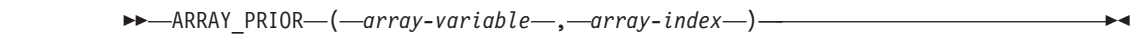
  ```
  SET X = ARRAY_NEXT(PHONELIST, 'Dad')
  ```

  'Dad'가 배열 변수에 대한 모든 인덱스 값보다 작은 값이기 때문에 'Home'이 X에 지정됩니다. 인덱스 값 'Work' 다음의 인덱스인 배열의 인덱스 값을 지정합니다.

  ```
  SET X = ARRAY_NEXT(PHONELIST, 'Work')
  ```

  널(NULL) 값이 X에 지정됩니다.

ARRAY_PRIOR



스키마는 SYSIBM입니다.

ARRAY_PRIOR 함수는 지정된 배열 인덱스 인수에 상대적인 배열에 대해 다음으로 작은 배열 인덱스 값을 리턴합니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

array-index
배열의 인덱스 데이터 유형에 지정할 수 있는 값을 지정합니다. 유효한 값은 데이터 유형으로 유효한 값을 포함합니다.

결과는 지정한 *array-index* 값에 연관된 배열에 정의된 다음으로 작은 배열 인덱스 값입니다. *array-index*가 배열의 최대 인덱스 배열 값보다 큰 경우 결과는 배열에 정의된 마지막 배열 인덱스 값입니다.

함수 결과의 데이터 유형은 배열 인덱스의 데이터 유형입니다. 결과는 널(NULL)이 될 수 있습니다. 인수가 널(NULL), 첫 번째 인수의 카디널리티(cardinality)가 0 또는 *array-index* 값이 배열의 첫 번째 인덱스 값 이상인 경우 결과는 널(NULL) 값입니다.

예

- 일반 배열 변수 SPECIALNUMBERS에서 두 번째 인덱스 위치 앞의 이전 인덱스 값을 SQL 변수 PREV_CONSTIDX에 리턴합니다.

  ```
  SET PREV_CONSTIDX = ARRAY_PRIOR(SPECIALNUMBERS,2)
  ```

  결과는 1입니다.

- 인덱스 값 및 전화번호가 'Home'은 4163053745', 'Work'는 '4163053746'이며 'Mom'이 '416-4789683'인 연관된 배열 변수 PHONELIST에서 인덱스 값 'Work' 앞의 이전 인덱스인 배열의 인덱스 값을 문자열 변수인 X로 지정합니다.

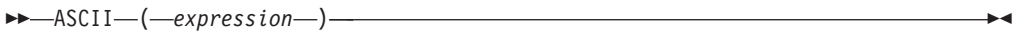
  ```
  SET X = ARRAY_PRIOR(PHONELIST, 'Work')
  ```

  'Mom' 값이 X로 지정됩니다. 인덱스 값 'Home' 앞의 이전 인덱스인 배열의 인덱스 값을 지정합니다.

  ```
  SET X = ARRAY_PRIOR(PHONELIST, 'Home')
  ```

  널(NULL) 값이 X에 지정됩니다.

ASCII



스키마는 SYSFUN입니다.

인수에서 가장 왼쪽 문자의 ASCII 코드 값을 정수로 리턴합니다.

인수는 모든 유형의 내장 문자열이 될 수 있습니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB의 최대 길이는 1 048 576바이트입니다. LONG VARCHAR은 함수가 처리할 수 있도록 CLOB로 변환됩니다.

함수의 결과는 항상 INTEGER입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

ASIN

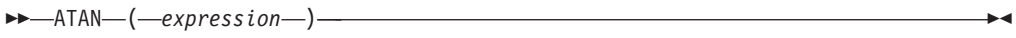
스키마는 SYSIBM입니다. (ASIN 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 arcsine을 라디안으로 표시되는 각도로 리턴합니다.

인수는 어떤 내장 숫자 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

ATAN



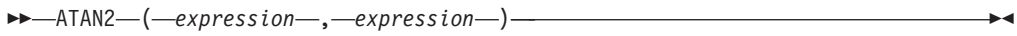
스키마는 SYSIBM입니다. (ATAN 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 arctangent를 라디안으로 표시되는 각도로 리턴합니다.

인수는 어떤 내장 숫자 데이터 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

ATAN2



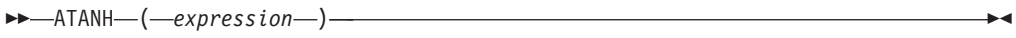
스키마는 SYSIBM입니다. (ATAN2 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

x 및 y 좌표의 arctangent를 라디안으로 표시되는 각도로 리턴합니다. x 및 y 좌표는 각각 첫 번째 및 두 번째 인수로 지정됩니다.

첫 번째 및 두 번째 인수는 어떤 내장 숫자 데이터 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 두 인수 모두 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

ATANH



스키마는 SYSIBM입니다.

인수의 hyperbolic arctangent를 리턴합니다. 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 모든 내장 숫자 데이터 유형(DECFLOAT 제외)일 수 있습니다. 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

BIGINT

숫자에서 큰 정수로

문자열에서 큰 정수로

날짜 및 시간에서 큰 정수로

스키마는 SYSIBM입니다.

BIGINT 함수는 다음의 64비트 정수 표현을 리턴합니다.

- 숫자
- 숫자의 문자열 표현
- 날짜 시간 값

숫자에서 큰 정수로

numeric-expression

임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

인수가 큰 정수 컬럼이나 변수에 지정된 경우, 결과는 같은 숫자입니다. 인수의 분수 부분이 절단됩니다. 인수의 전체 부분이 큰 정수 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

문자열에서 큰 정수로

string-expression

문자 상수의 최대 길이보다 길지 않은 길이의 숫자의 문자열 또는 유니코드 그래픽 문자열 표현인 값을 리턴하는 표현식입니다.

결과는 CAST(*string-expresssion* AS BIGINT)에서 발생하는 수와 동일합니다. 앞뒤 공백은 제거되고 결과 문자열은 정수, 부동 소수점이나 10진수 부동 소수점 상수에 대한 규칙에 따라야 합니다(SQLSTATE 22018). 인수의 전체 부분이 큰 정수 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).*string-expresssion*의 데이터 유형은 CLOB, LONG VARCHAR, DBCLOB 또는 LONG VARGRAPHIC이면 안됩니다(SQLSTATE 42884).

날짜 및 시간에서 큰 정수로

datetime-expression

다음 데이터 유형 중 하나인 표현식입니다.

- DATE. 결과는 날짜를 *yyyymmdd*로 표시하는 BIGINT 값입니다.
- TIME. 결과는 시간을 *hhmmss*로 표시하는 BIGINT 값입니다.
- TIMESTAMP. 결과는 시간소인을 *yyyymmddhhmmss*로 표시하는 BIGINT 값입니다. 시간소인 값의 소수 초 부분은 결과에 포함되지 않습니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주: CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예:

- ORDERS_HISTORY 테이블에서 순서 번호를 계산한 후 결과를 bigint 값으로 리턴하십시오.

```
SELECT BIGINT (COUNT_BIG(*))
  FROM ORDERS_HISTORY
```

- EMPLOYEE 테이블을 사용하여 응용프로그램을 추가로 처리하려면 큰 정수 형식으로 EMPNO 컬럼을 선택하십시오.

```
SELECT BIGINT (EMPNO) FROM EMPLOYEE
```

- RECEIVED 컬럼(시간소인)에 '1988-12-22-14.07.21.136421'에 해당하는 내부 값이 있다고 가정하십시오.

```
BIGINT(RECEIVED)
```

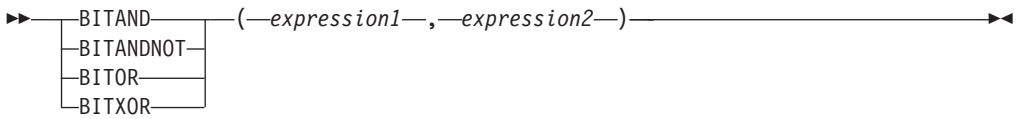
  결과 값은 19 881 222 140 721입니다.

- STARTTIME 컬럼(시간)에 '12:03:04'에 해당하는 내부 값이 있다고 가정하십시오.

```
BIGINT(STARTTIME)
```

  결과 값은 120 304입니다.

BITAND, BITANDNOT, BITOR, BITXOR 및 BITNOT



스키마는 SYSIBM입니다.

이 비트 방식 함수는 입력 인수의 정수 값에 대한 ″2의 보수″ 표현으로 작동하고 입력 인수의 데이터 유형을 기초로 데이터 유형의 해당되는 기수 10 정수 값으로 결과를 리턴합니다.

표 42. 비트 처리 함수

함수	설명	결과의 2의 보수 표현으로 된 비트는 다음과 같음
BITAND	비트 방식 AND 연산을 수행합니다.	두 인수의 해당 비트가 모두 1인 경우에만 1입니다.
BITANDNOT	두 번째 인수에 있는 첫 번째 인수에서 비트를 지웁니다.	두 번째 인수에서 해당되는 비트가 1인 경우 영(0)입니다. 그렇지 않으면 결과는 첫 번째 인수에 있는 해당 비트에서 복사됩니다.
BITOR	비트 방식 OR 연산을 수행합니다.	두 인수의 해당 비트가 모두 영(0)이 아니면 1입니다.
BITXOR	비트 방식 배타적 OR 연산을 수행합니다.	두 인수의 해당 비트가 같지 않으면 1입니다.
BITNOT	비트 방식 NOT 연산을 수행합니다.	인수에서 해당되는 비트의 반대입니다.

인수는 데이터 유형 SMALLINT, INTEGER, BIGINT 또는 DECFLOAT가 표시하는 정수 값이어야 합니다. 유형 DECIMAL, REAL 또는 DOUBLE의 인수는 DECFLOAT에 캐스트됩니다. 값은 정수로 절단됩니다.

비트 처리 함수는 SMALLINT의 경우 16개의 비트, INTEGER의 경우 32개의 비트, BIGINT의 경우 64개의 비트, DECFLOAT의 경우 113개의 비트까지 작동할 수 있습니다. 지원되는 DECFLOAT 값의 범위에는 -2^{112}에서 2^{112} -1까지의 정수가 포함되고, NaN 또는 INFINITY와 같은 특수 값은 지원되지 않습니다(SQLSTATE 42815). 두 인수가 다른 데이터 유형을 갖는 경우 더 적은 비트를 지원하는 인수가 더 많은 비트를 지원하는 인수의 데이터 유형을 갖는 값에 캐스트됩니다. 이 캐스트는 음수 값에 대해 설정되는 비트에 영향을 줍니다. 예를 들어, SMALLINT 값으로서의 -1은 16개의 비트가 1로 설정됩니다. INTEGER 값에 캐스트되는 경우 32개의 비트가 1로 설정됩니다.

두 개의 인수가 있는 함수의 결과는 프로모션에 대한 데이터 유형 순위 목록에서 가장 높은 인수의 데이터 유형을 갖습니다. 어느 하나의 인수가 DECFLOAT이면, 결과의 데이터 유형은 DECFLOAT(34)입니다. 인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우 결과는 널(NULL)이 됩니다.

BITNOT 함수의 결과는 DECIMAL, REAL, DOUBLE 또는 DECFLOAT(16)가 DECFLOAT(34)를 리턴하는 것을 제외하고 입력 인수와 같은 데이터 유형을 갖습니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

데이터 유형 사이, 그리고 서로 다른 하드웨어 플랫폼에서의 내부 표현 차이로 인해, 함수(예: HEX) 또는 호스트 언어 구문을 사용하여 BIT 함수 결과 및 인수의 내부 표현을 보거나 비교하는 것은 데이터 유형에 따라 다르므로 이식 가능하지 않습니다. BIT 함수 결과 및 인수를 보거나 비교하기 위한 데이터 유형 및 플랫폼 독립 방식은 실제 정수 값을 사용하는 것입니다.

BITXOR 함수를 사용하여 값에서 비트를 전환할 것을 권장합니다. BITANDNOT 함수를 사용하여 비트를 지우십시오. BITANDNOT(val, pattern)는 BITAND(val, BITNOT(pattern))보다 한층 효율적으로 작동합니다.

예를 들면, 다음과 같습니다.

다음 예는 PROPERTIES 컬럼 유형이 INTEGER인 ITEM 테이블을 기초로 합니다.

- 세 번째 등록 정보 비트가 설정된 모든 항목을 리턴합니다.

```
SELECT ITEMID FROM ITEM
  WHERE BITAND(PROPERTIES, 4) = 4
```

- 네 번째 또는 6번째 등록 정보 비트가 설정된 모든 항목을 리턴합니다.

```
SELECT ITEMID FROM ITEM
  WHERE BITAND(PROPERTIES, 40) <> 0
```

- ID가 3412인 항목의 12번째 등록 정보를 지웁니다.

```
UPDATE ITEM
    SET PROPERTIES = BITANDNOT(PROPERTIES, 2048)
  WHERE ITEMID = 3412
```

- ID가 3412인 항목의 5번째 등록 정보를 설정합니다.

```
UPDATE ITEM
    SET PROPERTIES = BITOR(PROPERTIES, 16)
  WHERE ITEMID = 3412
```

- ID가 3412인 항목의 11번째 등록 정보를 전환합니다.

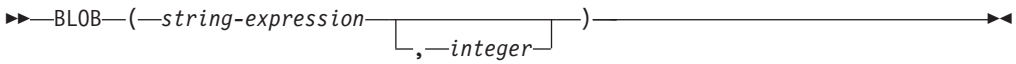
```
UPDATE ITEM
    SET PROPERTIES = BITXOR(PROPERTIES, 1024)
  WHERE ITEMID = 3412
```

- 두 번째 비트만 설정된 16비트 값에서 모든 비트를 전환합니다.

```
VALUES BITNOT(CAST(2 AS SMALLINT))
```

-3(데이터 유형 SMALLINT)을 리턴합니다.

BLOB



스키마는 SYSIBM입니다.

BLOB 함수는 어떤 유형의 문자열 BLOB 표현도 리턴합니다.

string-expression
값이 문자열, 그래픽 문자열 또는 실행 파일 문자열이 될 수 있는 *string-expression*.

integer
결과로 발생하는 BLOB 데이터 유형의 길이 속성을 지정하는 정수 값입니다. *integer*를 지정하지 않으면 결과의 길이 속성은 입력이 그래픽인 경우를 제외하고는 입력 길이와 동일합니다. 이 경우 결과의 길이 속성은 입력 길이의 두 배입니다.

함수의 결과는 BLOB입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예:

- 이름이 TOPOGRAPHIC_MAP인 BLOB 컬럼과 이름이 MAP_NAME인 VARCHAR 컬럼이 있는 테이블이 있는 경우, 문자열 'Pellow Island'를 포함하는 맵을 찾고 실제 맵 앞에 맵 이름이 병합된 단일 실행 파일 문자열을 리턴하십시오.

```
SELECT BLOB(MAP_NAME  CONCAT ':  ') CONCAT TOPOGRAPHIC_MAP
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE BLOB('%Pellow Island%')
```

CARDINALITY

스키마는 SYSIBM입니다.

CARDINALITY 함수는 배열의 요소 수를 표시하는 유형 BIGINT의 값을 리턴합니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

일반 배열의 경우, CARDINALITY 함수가 리턴하는 값은 배열이 지정된 요소를 가지고 있는 최상위 배열 인덱스입니다. 여기에는 널(NULL) 값이 지정된 요소가 포함됩니다. 연관된 배열의 경우, CARDINALITY 함수가 리턴하는 값은 *array-variable*에 정의된 고유 배열 인덱스 값의 실제 수입니다.

배열이 비어 있는 경우 함수는 0을 리턴합니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예

- 현재까지의 최신 호출에 저장된 호출 수를 리턴합니다.

```
SET HOWMANYCALLS = CARDINALITY(RECENT_CALLS)
```

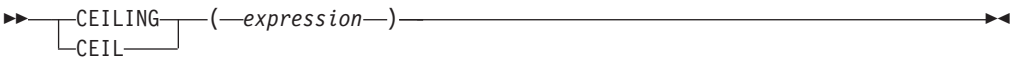
  SQL 변수 HOWMANYCALLS에는 값 3이 포함됩니다.

- 배열 유형 CAPITALSARRAY의 연관된 배열 변수 CAPITALS에는 캐나다의 10개 주 및 3개 영역에 대한 모든 수도 뿐만 아니라 국가의 수도인 오타와도 포함됩니다. 배열 변수의 카디널리티(cardinality)를 리턴합니다.

```
SET NUMCAPITALS = CARDINALITY(CAPITALS)
```

  SQL 변수 NUMCAPITALS에는 값 14가 포함됩니다.

CEILING 또는 CEIL



스키마는 SYSIBM입니다. (CEILING 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수보다 크거나 같은 최소의 정수 값을 리턴합니다.

인수는 모든 유형의 내장 숫자가 될 수 있습니다. 함수의 결과에는 인수가 DECIMAL인 경우 스케일이 0이라는 점을 제외하고는 인수와 동일한 데이터 유형 및 길이 속성이 있습니다. 예를 들어, 데이터 유형이 DECIMAL(5,5)인 인수는 DECIMAL(5,0)을 리턴합니다.

인수가 널(NULL)이 될 수 있거나 인수가 10진수 부동 소수점이 아니고 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

주

DECFLOAT 특수값을 포함하는 결과: 10진수 부동 소수점의 경우, 특수값은 다음과 같이 처리됩니다.

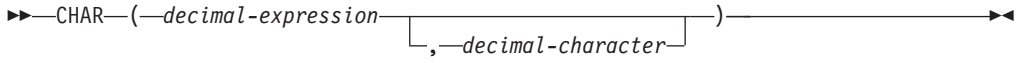
- CEILING(NaN)는 NaN을 리턴합니다.
- CEILING(-NaN)는 -NaN을 리턴합니다.
- CEILING(infinity)는 Infinity를 리턴합니다.
- CEILING(-Infinity)는 -Infinity를 리턴합니다.
- CEILING(sNaN)는 NaN과 경고를 리턴합니다.
- CEILING(-sNaN)는 -NaN과 경고를 리턴합니다.

구문 대체: CEILING 대신 CEIL을 지정할 수 있습니다.

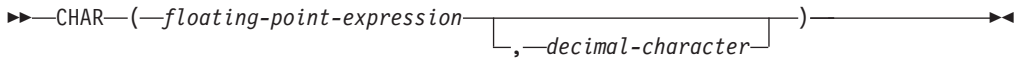
CHAR

정수에서 문자로

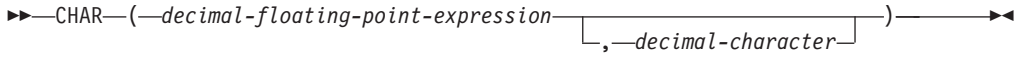
10진수를 문자 값으로 변환



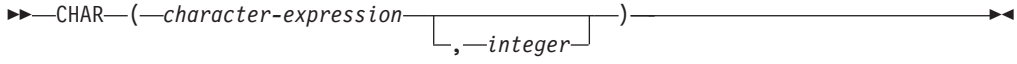
부동 소수점을 문자로



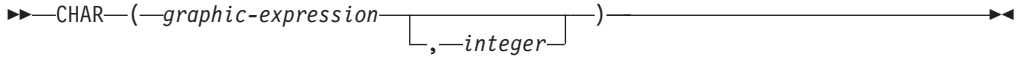
10진수 부동 소수점을 문자로



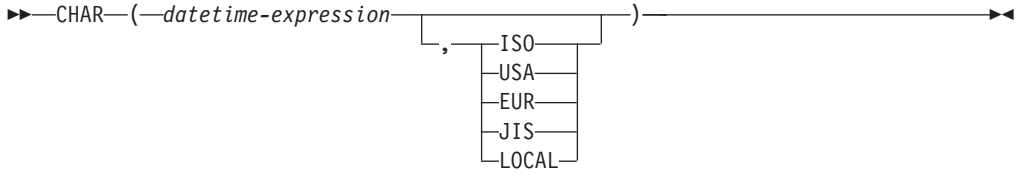
문자를 문자로



그래픽을 문자로



날짜 시간을 문자로



스키마는 SYSIBM입니다. 함수 시그니처에서 키워드를 사용할 때 함수 이름을 규정된 이름으로 지정할 수 없습니다. SYSFUN.CHAR(*floating-point-expression*) 시그니처는 계속 사용 가능합니다. 이 경우 10진 문자는 로케일을 구분하므로 데이터베이스 서버의 로케일에 따라 마침표나 쉼표를 리턴합니다.

CHAR 함수는 다음의 고정 길이 문자열 표현을 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우, 정수
- 첫 번째 인수가 10진수인 경우, 10진수
- 첫 번째 인수가 DOUBLE 또는 REAL인 경우, 배정밀도 부동 소수점 숫자
- 첫 번째 인수가 DECFLOAT인 경우, 10진수 부동 소수점 숫자
- 첫 번째 인수가 임의의 문자열 유형인 경우 문자열
- 첫 번째 인수가 임의의 그래픽 문자열(Unicode 데이터베이스 전용) 유형인 경우 그래픽 문자열
- 첫 번째 인수가 날짜, 시간 또는 시간소인인 경우, 날짜 시간 값

유니코드 데이터베이스에서는 다중 바이트 문자를 통해 출력 문자열이 파트별로 절단됩니다.

- 입력이 문자열인 경우 부분 문자가 하나 이상의 공백으로 대체됨
- 입력이 그래픽 문자열인 경우 부분 문자가 비어 있는 문자열로 대체됨

추후 릴리스에서 변경될 수 있으므로 이러한 동작의 영향을 받지 않는 것이 좋습니다.

함수의 결과는 고정 길이 문자열입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

정수에서 문자로

integer-expression
데이터 유형이 정수(SMALLINT, INTEGER 또는 BIGINT)인 값을 리턴하는 표현식입니다.

결과는 SQL 정수 상수의 양식으로 된 *integer-expression*의 고정 길이의 문자열 표현입니다. 결과는 인수의 유효 자릿수를 표시하는 *n*문자로 구성되며, 인수가 음수일 경우에는 빼기 부호가 앞에 나타납니다. 결과는 왼쪽으로 정렬됩니다.

- 첫 번째 인수가 '작은 정수(small integer)'인 경우 결과 길이는 6입니다.
- 첫 번째 인수가 '큰 정수(large integer)'인 경우 결과 길이는 11입니다.
- 첫 번째 인수가 '대정수(big integer)'인 경우 결과 길이는 20입니다.

결과의 바이트 수가 정의된 결과 길이보다 작으면 결과의 오른쪽에 1바이트 공백이 채워집니다.

결과의 코드 페이지는 섹션의 코드 페이지입니다.

10진수를 문자 값으로 변환

decimal-expression

10진수 데이터 유형인 값을 리턴하는 표현식입니다. 다른 정밀도 및 스케일이 필수일 경우 먼저 DECIMAL 스칼라 함수를 사용하여 변경할 수 있습니다.

decimal-character

결과 문자열에서 10진 숫자를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 상수의 양식으로 된 *decimal-expression*의 고정 길이의 문자열 표현입니다. 결과 길이는 2 + *p*입니다. 여기서, *p*는 *decimal-expression*의 정밀도입니다. 앞에 0이 포함되지 않습니다. 트레일링 0이 포함됩니다. *decimal-expression*이 음수이면 결과의 첫 문자는 빼기 기호이고, 그렇지 않으면 첫 문자가 숫자이거나 10진수 문자입니다. *decimal-expression*의 스케일이 0이면, 10진수 문자가 리턴되지 않습니다. 결과의 바이트 수가 정의된 결과 길이보다 작으면 결과의 오른쪽에 1바이트 공백이 채워집니다.

결과의 코드 페이지는 섹션의 코드 페이지입니다.

부동 소수점을 문자로

floating-point-expression

부동 소수점 데이터 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식입니다.

decimal-character

결과 문자열에서 10진 숫자를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *floating-point-expression*의 고정 길이의 문자열 표현입니다. 결과의 길이는 24입니다. 가수(mantissa)가 마침표와 일련의 숫자가 뒤에 나오는 0 이외의 단일 숫자로 구성될 수 있도록 *floating-point-expression*의 값을 표시할 수 있는 최소 문자 수가 결과가 됩니다. *floating-point-expression*이 음수이면 결과의 첫 문자는 빼기 기호이고, 그렇지 않으면 첫 문자가 숫자입니다. *floating-point-expression*이 0이면, 결과는 0E0입니다. 결과의 바이트 수가 24보다 작으면, 결과의 오른쪽에 1바이트 공백이 채워집니다.

결과의 코드 페이지는 섹션의 코드 페이지입니다.

10진수 부동 소수점을 문자로

decimal-floating-point-expression

부동 소수점 데이터 유형(DOUBLE)인 값을 리턴하는 표현식입니다.

decimal-character

결과 문자열에서 10진 숫자를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *decimal-floating-point-expression*의 고정 길이 문자열 표현입니다. 결과의 길이 속성은 42입니다. 결과는 *decimal-floating-point-expression*의 값을 표시할 수 있는 최소 문자 수입니다. *decimal-floating-point-expression*이 음수이면, 결과의 첫 문자는 빼기 기호이고, 그렇지 않으면 첫 문자가 숫자입니다. *decimal-floating-point-expression*이 0이면 결과는 0입니다.

decimal-floating-point-expression 값이 특수 값인 Infinity, sNaN 또는 NaN인 경우에는 각각 'INFINITY', 'SNAN' 및 'NAN' 문자열이 리턴됩니다. 특수 값이 음수이면 결과의 첫 문자는 빼기 기호입니다. 문자열로 변환될 때 10진수 부동 소수점 특수 값 sNaN은 경고를 발생시키지 않습니다. 결과의 문자 수가 42보다 작으면, 결과의 오른쪽에 1바이트 공백이 채워집니다.

결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

문자를 문자로

character-expression

값을 리턴하는 표현식은 내장 문자열 데이터 유형(CHAR, VARCHAR 또는 CLOB)입니다.

integer

결과로 발생하는 고정 길이 문자열의 길이 속성입니다. 값의 범위는 0 - 254이어야 합니다.

두 번째 인수가 지정되지 않은 경우:

- *character-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다. 첫 번째 인수의 실제 길이(뒤 공백 제외)가 254보다 크면, 오류가 리턴됩니다(SQLSTATE 22001).

결과의 실제 길이는 결과의 길이 속성과 같습니다. *character-expression*의 길이가 결과의 길이보다 작으면, 결과에는 결과의 길이까지 공백이 채워집니다. *character-expression*의 길이가 결과의 길이 속성보다 크면, 결과가 절단됩니다. 절단된 문자가 모두 공백이고 *character-expression*이 CLOB가 아닌 경우를 제외하고는 경고가 리턴됩니다(SQLSTATE 01004).

문자 표현식의 길이가 결과의 길이 속성보다 작으면 결과에는 결과의 길이까지 공백이 채워집니다. 문자 표현식의 길이가 결과의 길이 속성보다 크면 결과가

절단됩니다. 절단된 문자가 모두 공백이고 문자 표현식이 CLOB가 아닌 경우를 제외하고는 경고가 리턴됩니다(SQLSTATE 01004).

그래픽을 문자로

graphic-expression
내장 그래픽 문자열 데이터 유형(GRAPHIC, VARGRAPHIC 또는 DBCLOB)인 값을 리턴하는 표현식입니다.

integer
결과로 발생하는 고정 길이 문자열의 길이 속성입니다. 값의 범위는 0 - 254이어야 합니다.

두 번째 인수가 지정되지 않은 경우:

- *graphic-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 MIN과 같습니다(254, 3 * 첫 번째 인수의 길이 속성). 첫 번째 인수의 실제 길이(뒤 공백 포함)가 254보다 크면, 오류가 리턴됩니다(SQLSTATE 22001).

결과의 실제 길이는 결과의 길이 속성과 같습니다. *graphic-expression*의 길이가 결과의 길이보다 작으면, 결과에는 결과의 길이까지 공백이 채워집니다. *graphic-expression*의 길이가 결과의 길이 속성보다 크면, 경고가 리턴되지 않고 절단이 수행됩니다.

날짜 시간을 문자로

datetime-expression
다음 데이터 유형 중 하나인 표현식입니다.

DATE
결과는 두 번째 인수가 지정하는 형식으로 된 날짜의 문자열 표현입니다. 결과 길이는 10입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않은 경우 오류가 발생합니다(SQLSTATE 42703).

TIME 결과는 두 번째 인수가 지정하는 형식으로 된 시간의 문자열 표현입니다. 결과의 길이는 8입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않으면 오류가 발생합니다(SQLSTATE 42703).

TIMESTAMP
결과는 시간소인의 문자열 표현입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(0)이면, 결과의 길이는 19입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(*n*)이고 *n*이 1과 12 사이인 경우, 결과의 길이는 20+*n*입니다. 그렇지 않으면 결과의 길이는 26입니다.두 번째 인수는 적용될 수 없으므로 지정하지 마십시오(SQLSTATE 42815).

결과의 코드 페이지는 섹션의 코드 페이지입니다.

참고:

- 첫 번째 인수가 숫자이거나 첫 번째 인수가 문자열이고 길이 인수가 지정되어 있을 때 CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.
- 실행 파일 문자열이 함수에 대해 첫 번째 인수로 허용되며, 결과로 발생되는 고정 길이 문자열은 필요한 경우 공백으로 채워진 FOR BIT DATA 문자열입니다.
- **문자에 대한 10진수 및 선행 영(0)**: 9.7 이전 버전의 경우 이 함수에 대한 10진수 입력의 결과는 선행 영(0) 및 트레일링 10진수 문자입니다. 데이터베이스 구성 매개변수 *dec_to_char_fmt*는 『V95』로 설정되어 10진수 입력에 대하여 이 함수가 버전 9.5 결과를 리턴하도록 할 수 있습니다. 새 데이터베이스에 대한 *dec_to_char_fmt*의 디폴트값은 『NEW』이고 이 값은 SQL 표준 캐스팅 규칙과 일치하는 이 함수의 리턴 결과를 포함하며 VARCHAR 함수의 결과와 일치합니다.

예를 들면, 다음과 같습니다.

- PRSTDATE 컬럼에 1988-12-25에 해당하는 내부 값이 있다고 가정합니다. 다음 함수는 '12/25/1988' 값을 리턴합니다.

  ```
  CHAR(PRSTDATE, USA)
  ```

- STARTING 컬럼에 17:12:30에 해당되는 내부 값이 있다고 가정하고 호스트 변수 HOUR_DUR(10진수(6,0))이 050000(즉, 5시간)의 값을 가지는 시간 지속 기간이라고 가정합니다. 다음 함수는 '5:12 PM' 값을 리턴합니다.

  ```
  CHAR(STARTING, USA)
  ```

  다음 함수는 '10:12 PM' 값을 리턴합니다.

  ```
  CHAR(STARTING + :HOUR_DUR, USA)
  ```

- RECEIVED 컬럼(TIMESTAMP)에 PRSTDATE 및 STARTING 컬럼의 조합에 해당되는 내부 값이 있다고 가정합니다. 다음 함수는 '1988-12-25-17.12.30.000000' 값을 리턴합니다.

  ```
  CHAR(RECEIVED)
  ```

- LASTNAME 컬럼은 VARCHAR(15)로 정의되어 있습니다. 다음 함수는 이 컬럼의 값을 길이가 10자인 고정 길이 문자열로 리턴합니다. 길이가 10자(뒤 공백 제외)를 넘으면 LASTNAME 값은 절단되고 경고가 리턴됩니다.

  ```
  SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
  ```

- EDLEVEL 컬럼은 SMALLINT로 정의되어 있습니다. 다음 함수는 고정 길이 문자열로 이 컬럼의 값을 리턴합니다. EDLEVEL 값 18은 뒤에 4개의 공백이 있는 CHAR(6) 값 '18'로 리턴됩니다.

  ```
  SELECT CHAR(EDLEVEL) FROM EMPLOYEE
  ```

- SALARY 컬럼은 스케일 2 및 정밀도 9인 DECIMAL로 정의되어 있습니다. 현재 값(18357.50)은 쉼표를 사용하여 10진 문자(18357,50)로 표시됩니다. 다음 함수는 뒤에 3개의 공백이 있는 '18357,50' 값을 리턴합니다.

  ```
  CHAR(SALARY, ',')
  ```

- SALARY 컬럼의 값을 20000.25에서 빼야 하며 디폴트 10진수 문자로 표시해야 합니다. 다음 함수는 뒤에 3개의 공백이 있는 '-0001642.75' 값을 리턴합니다.

  ```
  CHAR(20000.25 - SALARY)
  ```

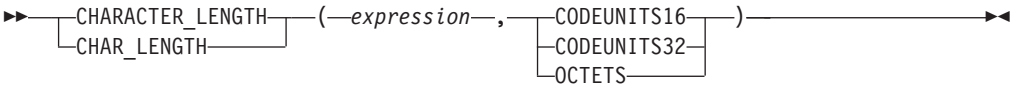
- 호스트 변수 DOUBLE_NUM이 SEASONS로 정의되어 있고 -987.654321E-35 값을 가진다고 가정합니다. 다음 함수는 '10000.00' 값을 리턴합니다.

  ```
  CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
  ```

- 호스트 변수 DOUBLE_NUM이 DOUBLE로 정의되어 있고 -987.654321E-35 값을 가진다고 가정합니다. 다음 함수는 결과 데이터 유형이 CHAR(24)이기 때문에 뒤에 9개의 후미 공백이 있는 '-9.87654321E-33' 값을 리턴합니다.

  ```
  CHAR(:DOUBLE_NUM)
  ```

CHARACTER_LENGTH



스키마는 SYSIBM입니다.

CHARACTER_LENGTH 함수는 *expression*의 길이를 지정된 문자열 길이 단위로 리턴합니다.

expression
내장 문자 또는 그래픽 문자열의 값을 리턴하는 표현식입니다.

CODEUNITS16, CODEUNITS32, 또는 OCTETS
결과의 문자열 단위를 지정합니다. CODEUNITS16은 결과가 16비트 UTF-16 코드 단위로 표시될 것을 지정합니다. CODEUNITS32는 결과가 32비트 UTF-32 코드 단위로 표시될 것을 지정합니다. OCTETS는 결과가 바이트 단위로 표시될 것을 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *expression*이 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *expression*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815). CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

문자 또는 그래픽 문자열의 길이는 뒤 공백을 포함합니다. 가변 길이 문자열의 실이는 실제 길이이며 최대 길이가 아닙니다.

예를 들면, 다음과 같습니다.

- NAME이 VARCHAR(128) 컬럼이고 유니코드 UTF-8로 인코드되었으며 'Jürgen' 값이 들어 있다고 가정하십시오. 다음 두 쿼리는 값 6을 리턴합니다.

```
SELECT CHARACTER_LENGTH(NAME, CODEUNITS32)
FROM T1 WHERE NAME = 'Jürgen'

SELECT CHARACTER_LENGTH(NAME, CODEUNITS16)
FROM T1 WHERE NAME = 'Jürgen'
```

  다음 두 쿼리는 값 7을 리턴합니다.

```
SELECT CHARACTER_LENGTH(NAME, OCTETS)
FROM T1 WHERE NAME = 'Jürgen'

SELECT LENGTH(NAME)
FROM T1 WHERE NAME = 'Jürgen'
```

- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'
UTF-32BE	X'0001D11E'	X'0000004E'	X'00000303'	X'00000041'	X'00000042'

변수 UTF8_VAR에 해당 문자열의 UTF-8 표시가 들어 있다고 가정하십시오.

```
SELECT CHARACTER_LENGTH(UTF8_VAR, CODEUNITS16),
  CHARACTER_LENGTH(UTF8_VAR, CODEUNITS32),
  CHARACTER_LENGTH(UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

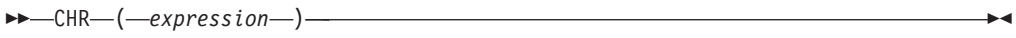
리턴되는 값은 각각 6, 5, 9입니다.

변수 UTF16_VAR에 해당 문자열의 UTF-16BE 표시가 들어 있다고 가정하십시오.

```
SELECT CHARACTER_LENGTH(UTF16_VAR, CODEUNITS16),
   CHARACTER_LENGTH(UTF16_VAR, CODEUNITS32),
   CHARACTER_LENGTH(UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

리턴되는 값은 각각 6, 5, 12입니다.

CHR



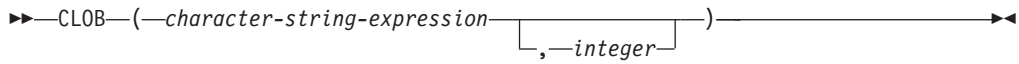
스키마는 SYSFUN입니다.

인수에서 지정한 ASCII 코드 값을 갖고 있는 문자를 리턴합니다. *expression*이 0인 경우 결과는 공백 문자(X'20')입니다.

인수는 INTEGER 또는 SMALLINT가 될 수 있습니다. 인수 값은 0 - 255 사이여야 합니다. 그렇지 않으면 리턴값은 255에 해당되는 ASCII 코드 값을 갖는 문자입니다.

함수의 결과는 CHAR(1)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

CLOB



스키마는 SYSIBM입니다.

CLOB 함수는 문자열 유형의 CLOB 표현을 리턴합니다. 유니코드 데이터베이스의 경우, 제공된 인수가 그래픽 문자열이면 함수를 실행하기 전에 먼저 문자열 데이터 유형으로 변환해야 합니다.

character-string-expression

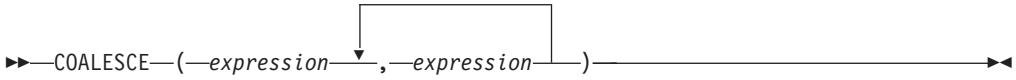
문자열 값을 리턴하는 표현식. 표현식은 FOR BIT DATA로 정의된 문자열이 될 수 없습니다(SQLSTATE 42846).

integer

결과로 발생하는 CLOB 데이터 유형의 길이 속성을 지정하는 정수 값. 값은 0 및 2 147 483 647 사이여야 합니다. *integer* 값을 지정하지 않으면 결과 길이는 첫 번째 인수의 길이와 같습니다.

함수의 결과는 CLOB입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

COALESCE



스키마는 SYSIBM입니다.

COALESCE는 널(NULL)이 아닌 첫 번째 인수를 리턴합니다.

인수는 지정된 순서대로 평가되고 함수의 결과는 널(NULL)이 아닌 첫 번째 인수입니다. 결과는 모든 인수가 널(NULL)이 될 수 있는 경우에만 널(NULL)이 될 수 있으며, 모든 인수가 널(NULL)인 경우에만 결과도 널(NULL)이 됩니다. 필요한 경우 선택된 인수는 결과의 속성으로 변환됩니다.

인수는 호환 가능해야 합니다. 인수는 내장 또는 사용자 정의 데이터 유형일 수 있습니다. 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 또한 이 함수는 인수로 호환 가능한 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하는 추가 시그니처를 작성할 필요가 없습니다.

예를 들면, 다음과 같습니다.

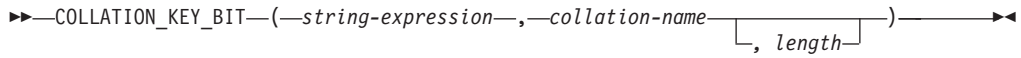
- DEPARTMENT 테이블의 모든 행에서 모든 값을 선택할 때, 부서 관리자(MGRNO)가 누락되면(즉, 널(NULL)) 'ABSENT' 값을 리턴합니다.

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- EMPLOYEE 테이블의 모든 행에서 사원 번호(EMPNO) 및 봉급(SALARY)을 선택할 때, 봉급이 누락되면(즉, 널(NULL)) 0 값을 리턴합니다.

```
SELECT EMPNO, COALESCE(SALARY, 0)
  FROM EMPLOYEE
```

COLLATION_KEY_BIT



스키마는 SYSIBM입니다.

COLLATION_KEY_BIT 함수는 지정된 *collation-name*에서 *string-expression*의 조합 키를 나타내는 VARCHAR FOR BIT DATA 문자열을 리턴합니다.

두 문자열에 대한 COLLATION_KEY_BIT의 결과는 지정된 *collation-name*에서 해당 순서를 판별하기 위해 비교되는 2진수가 될 수 있습니다. 의미있는 비교를 위해, 사용되는 결과는 동일한 *collation-name*의 결과여야 합니다.

string-expression
조합 키를 판별해야 하는 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 문자열을 리턴하는 표현식입니다. *string-expression*이 CHAR 또는 VARCHAR이면, 표현식은 FOR BIT DATA가 될 수 없습니다(SQLSTATE 429BM).

*string-expression*이 UTF-16으로 되어 있지 않으면, 이 함수는 *string-expression*에서 UTF-16으로의 코드 페이지 변환을 수행합니다. 코드 페이지 변환의 결과에 최소 하나의 대체 문자가 포함되어 있으면 이 함수는 대체 문자와 함께 UTF-16 문자열의 조합 키를 리턴하고 SQLCA의 경고 플래그 SQLWARN8이 'W'로 설정됩니다.

collation-name
조합 키를 판별할 때 사용할 조합을 지정하는 문자 상수입니다. *collation-name*의 값은 대소문자가 구분되지 않으며 *자국어 안내서* 의 『UCA(Unicode Collation Algorithm) 기반 조합』이나 *자국어 안내서* 의 『유니코드 데이터에 대한 언어 인식 조합』 중 하나여야 합니다(SQLSTATE 42616).

length
결과의 길이 속성을 바이트 단위로 지정하는 표현식입니다. 지정된 경우 *length*는 1 - 32 672 사이의 정수여야 합니다(SQLSTATE 42815).

*length*의 값을 지정하지 않으면 결과 길이는 다음과 같이 판별됩니다.

표 43. 결과 길이 판별

문자열 인수 데이터 유형	결과 데이터 유형 길이
CHAR(*n*) 또는 VARCHAR(*n*)	12*n*바이트 및 32,672바이트 중 최소
GRAPHIC(*n*) 또는 VARGRAPHIC(*n*)	12*n*바이트 및 32,672바이트 중 최소

length 지정 여부에 관계없이 조합 키의 길이가 결과 데이터 유형의 길이보다 길면, 오류가 리턴됩니다(SQLSTATE 42815). 조합 키의 실제 결과 길이는 UTF-16으로 변환된 후 대략 *string-expression* 길이의 6배입니다.

*string-expression*이 빈 문자열인 경우 결과는 길이가 0이 아닐 수 있는 유효한 조합 키입니다.

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

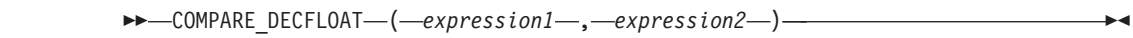
다음 쿼리는 코드 페이지 923에서 독일어의 언어 인식 조합을 사용하여 성별로 사원의 순서를 지정합니다.

```
SELECT FIRSTNME, LASTNAME
  FROM EMPLOYEE
  ORDER BY COLLATION_KEY_BIT (LASTNAME, 'SYSTEM_923_DE')
```

다음 쿼리는 문화적으로 올바른 비교를 사용하여 Québec 지역에 있는 사원의 부서를 찾습니다.

```
SELECT E.WORKDEPT
  FROM EMPLOYEE AS E INNER JOIN SALES AS S
  ON COLLATION_KEY_BIT(E.LASTNAME, 'UCA400R1_LFR') =
    COLLATION_KEY_BIT(S.SALES_PERSON, 'UCA400R1_LFR')
  WHERE S.REGION = 'Quebec'
```

COMPARE_DECFLOAT



스키마는 SYSIBM입니다.

COMPARE_DECFLOAT 함수는 두 인수가 동일하거나 순서화되지 않았는지 여부, 또는 하나의 인수가 다른 인수보다 큰 지 여부를 표시하는 SMALLINT 값을 리턴합니다.

expression1
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 DECFLOAT(34)가 아니면 처리를 위해 논리적으로 DECFLOAT(34)로 변환됩니다.

expression2
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 DECFLOAT(34)가 아니면 처리를 위해 논리적으로 DECFLOAT(34)로 변환됩니다.

*expression1*의 값은 *expression2* 값과 비교되며 결과는 다음 규칙에 따라 리턴됩니다.

- 두 인수 모두 유한인 경우 비교는 대수이며 10진 부동 소수점 빼기에 대한 프로시저를 따릅니다. 차이가 정확히 0(두 부호 중 하나의)인 경우 인수는 같습니다. 0이 아닌 차이가 양수이면 첫 번째 인수가 두 번째 인수보다 큽니다. 0이 아닌 차이가 음수이면 첫 번째 인수가 두 번째 인수보다 작습니다.
- 양수 영(0) 및 음수 영(0)은 같은 것으로 비교됩니다.
- 양수 무한은 양수 무한과 같은 것으로 비교됩니다.
- 양수 무한은 임의 유한 숫자보다 큰 것으로 비교됩니다.
- 음수 무한은 음수 무한과 같은 것으로 비교됩니다.
- 음수 무한은 임의 유한 숫자보다 작은 것으로 비교됩니다.
- 숫자 비교는 정확합니다. 결과는 범위 및 정밀도가 무제한인 것처럼 유한 피연산자에 대해 판별됩니다. 오버플로우 또는 언더플로우 조건은 발생할 수 없습니다.
- 인수 중 하나가 NaN 또는 sNaN(양수 또는 음수)이면 결과는 순서화되지 않습니다.

결과 값은 다음과 같습니다.

- 인수가 정확이 같은 경우 0
- *expression1*이 *expression2*보다 작은 경우 1
- *expression1*이 *expression2*보다 큰 경우 2
- 인수가 순서화되지 않는 경우 3

함수의 결과는 SMALLINT 값입니다. 인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

- 다음 예는 다양한 입력 10진 부동 소수점 값이 제공되는 경우 COMPARE_DECFLOAT 함수가 리턴하는 값을 나타냅니다.

```
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(2.17)) = 0
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(2.170)) = 2
COMPARE_DECFLOAT(DECFLOAT(2.170), DECFLOAT(2.17)) = 1
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(0.0)) = 2
COMPARE_DECFLOAT(INFINITY,INFINITY) = 0
COMPARE_DECFLOAT(INFINITY,-INFINITY) = 2
COMPARE_DECFLOAT(DECFLOAT(-2),INFINITY) = 1
COMPARE_DECFLOAT(NAN,NAN) = 3
COMPARE_DECFLOAT(DECFLOAT(-0.1),SNAN) = 3
```

CONCAT

스키마는 SYSIBM입니다.

CONCAT 함수는 두 개의 인수를 조합하여 문자열 표현식을 형성합니다.

expression1
: 문자열 데이터 유형, 숫자 데이터 유형 또는 날짜 및 시간 데이터 유형의 값을 리턴하는 표현식입니다.

expression2
: 문자열 데이터 유형, 숫자 데이터 유형 또는 날짜 및 시간 데이터 유형의 값을 리턴하는 표현식입니다. 그러나 일부 데이터 유형은 아래에 설명된대로 *expression1*의 데이터 유형이 있는 조합에서 지원되지 않습니다.

인수는 문자열(실행 파일 문자열 제외), 숫자 및 날짜 시간 값의 조합이 될 수 있습니다. 인수가 문자열 값이 아닌 경우, 명시적으로 VARCHAR로 캐스트됩니다.실행 파일 문자열은 다른 실행 파일 문자열과만 병합될 수 있습니다. 그러나 함수 결정의 캐스트 가능한 프로세스를 통해, 첫 번째 인수가 실행 파일 문자열일 때 실행 파일 문자열은 FOR BIT DATA로 정의된 문자열과 병합될 수 있습니다.

문자열 인수 및 그래픽 문자열 인수를 포함하는 병합은 유니코드 데이터베이스에서만 지원됩니다. 병합 전에 문자 인수가 먼저 그래픽 데이터 유형으로 변환됩니다. FOR BIT DATA로 정의된 문자열은 그래픽 데이터 유형으로 캐스트될 수 없습니다.

함수의 결과는 첫 번째 인수와 두 번째 인수로 구성되는 문자열입니다. 결과의 데이터 유형 및 길이는 적용 가능한 캐스팅이 완료된 후 인수의 데이터 유형과 길이로 판별됩니다. 자세한 정보는, 『표현식』 주제에서 『병합된 피연산자의 데이터 유형과 길이』 테이블을 참조하십시오.

인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우 결과는 널(NULL)이 됩니다.

주

- 병합이 수행될 때 부적절하게 구성된 혼합 데이터에 대해 점검이 이루어지지 않았음에 유의하십시오.
- CONCAT 함수는 CONCAT 연산자와 동일합니다. 자세한 정보는, 『표현식』을 참조하십시오.

예 :

- 컬럼 FIRSTNME와 컬럼 LASTNAME를 병합합니다.

```
SELECT CONCAT(FIRSTNME, LASTNAME)
FROM EMPLOYEE
WHERE EMPNO = '000010'
```

값 『CHRISTINEHAAS』를 리턴합니다.

COS

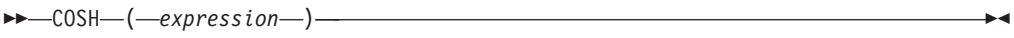
스키마는 SYSIBM입니다. (COS 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 cosine을 리턴합니다. 여기서, 인수는 라디안으로 표시되는 각도입니다.

인수는 어떤 내장 숫자 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

COSH



스키마는 SYSIBM입니다.

인수의 hyperbolic cosine을 리턴합니다. 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 모든 내장 숫자 데이터 유형(DECFLOAT 제외)일 수 있습니다. 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

COT

스키마는 SYSIBM입니다. (COT 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 cotangent를 리턴합니다. 여기서, 인수는 라디안으로 표시되는 각도입니다.

인수는 어떤 내장 숫자 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

CURSOR_ROWCOUNT

스키마는 SYSIBM입니다.

CURSOR_ROWCOUNT 함수는 커서가 열린 이후로 지정된 커서에 의해 페치되는 모든 누적 행 수를 리턴합니다.

cursor-variable-name

커서 유형의 SQL 변수 또는 SQL 매개변수의 이름입니다. *cursor-variable-name*의 밑줄 긋기 커서가 열려야 합니다(SQLSTATE 24501).

cursor-variable-name의 밑줄 긋기 커서에 대한 FETCH 조치가 함수 평가 이전에 수행되지 않은 경우 결과는 0입니다.

이 함수는 복합 SQL (컴파일된)문에서만 사용할 수 있습니다.

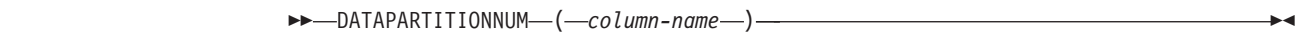
결과의 데이터 유형은 BIGINT입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예:

다음 예는 커서 *curEmp*와 연관된 행 수를 검색하고 *rows_fetched* 변수에 지정하기 위한 함수의 사용 방법을 나타냅니다.

```
SET rows_fetched = CURSOR_ROWCOUNT(curEmp);
```

DATAPARTITIONNUM



스키마는 SYSIBM입니다.

DATAPARTITIONNUM 함수는 행이 있는 데이터 파티션의 시퀀스 번호(SYSDATAPARTITIONS.SEQNO)를 리턴합니다. 데이터 파티션은 범위별로 정렬되며 시퀀스 번호는 0에서 시작됩니다. 예를 들어, DATAPARTITIONNUM 함수는 최하위 범위의 데이터 파티션에 있는 행에 대해 0을 리턴합니다.

인수는 테이블의 컬럼에 대해 규정된 이름 또는 규정되지 않은 이름일 수 있습니다. 행 레벨 정보가 리턴되므로 지정된 컬럼에 관계 없이 결과가 동일합니다. 컬럼의 데이터 유형에는 제한이 없습니다.

*column-name*에서 뷰의 컬럼을 참조할 경우, 뷰의 컬럼에 대한 표현식은 기본 테이블의 컬럼을 참조해야 하며 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다.

결과의 데이터 유형은 INTEGER이며 절대 널(NULL)이 될 수 없습니다.

이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 함수가 모든 데이터 유형을 인수로 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

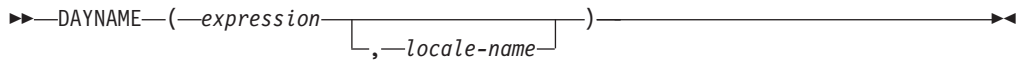
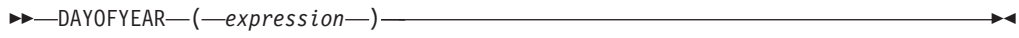
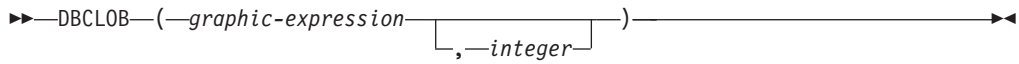
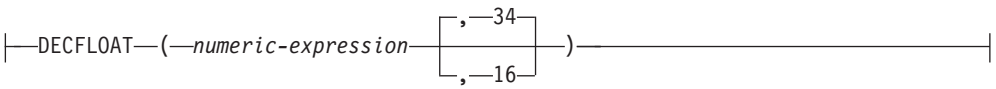
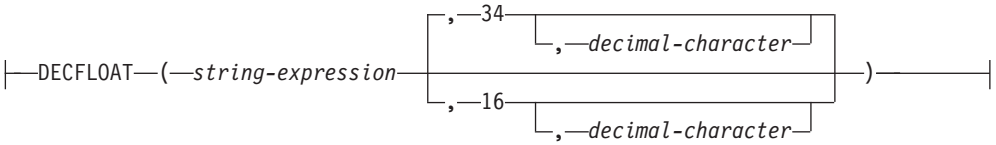
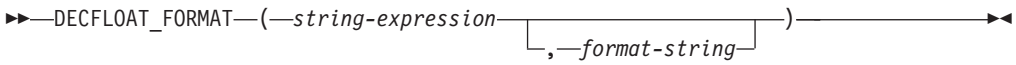
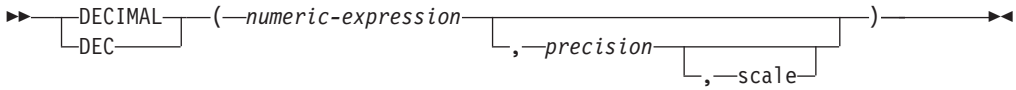
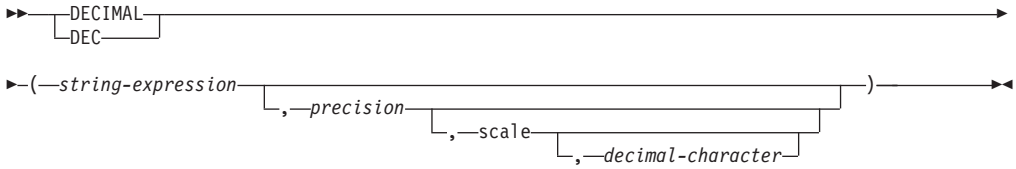
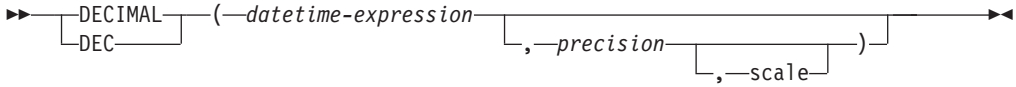
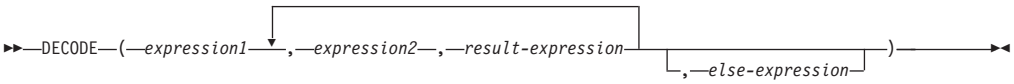
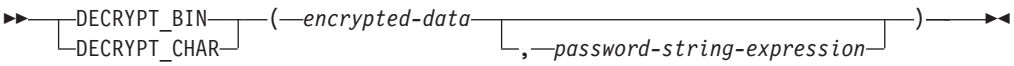
DATAPARTITIONNUM 함수는 점검 제한조건 또는 생성된 컬럼에 정의에 사용할 수 없습니다(SQLSTATE 42881). DATAPARTITIONNUM 함수는 구체화된 쿼리 테이블(MQT) 정의에 사용할 수 없습니다(SQLSTATE 428EC).

예:

- ```
  SELECT DATAPARTITIONNUM (EMPNO)
  FROM EMPLOYEE
  ```

  DATAPARTITIONNUM이 리턴한 시퀀스 번호(예: 0)를 다른 SQL문(예: ALTER TABLE...DETACH PARTITION)에서 사용할 수 있는 데이터 파티션 이름으로 변환하기 위해 SYSCAT.DATAPARTITIONS 카탈로그 뷰를 쿼리할 수 있습니다. 다음 예에 표시된 대로 WHERE절의 DATAPARTITIONNUM에서 얻은 SEQNO를 포함시키십시오.

  ```
  SELECT DATAPARTITIONNAME
  FROM SYSCAT.DATAPARTITIONS
  WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
  ```

  결과 값은 'PART0'입니다.
  ```

DATE

스키마는 SYSIBM입니다.

DATE 함수는 값에서 날짜를 리턴합니다.

인수는 날짜, 시간소인, 3 652 059 이하의 양수, 날짜나 시간소인의 유효한 문자열 표현, 또는 CLOB, LONG VARCHAR, DBCLOB 또는 LONG VARGRAPHIC이 아닌 길이가 7인 문자열이어야 합니다.

유니코드 데이터베이스만이 날짜 또는 시간소인의 그래픽 문자열 표현인 인수를 지원합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

인수가 길이 7의 문자열인 경우 *yyyynnn* 양식으로 유효한 날짜를 표시해야 합니다. 여기서 *yyyy*는 연도를 나타내는 숫자이고 *nnn*은 001 및 366 사이의, 해당 연도 일을 나타내는 숫자입니다.

함수의 결과는 날짜입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 날짜 파트입니다.
- 인수가 숫자인 경우:
  - 결과는 0001년 1월 1일 이후의 (*n* - 1)일인 날짜입니다. 여기서 *n*은 숫자의 정수 부분입니다.
- 인수가 7 길이의 문자열인 경우:
  - 결과는 문자열로 표시되는 날짜입니다.

예를 들면, 다음과 같습니다.

RECEIVED 컬럼(시간소인)에 '1988-12-25-17.12.30.000000'에 해당하는 내부 값이 있다고 가정하십시오.

- 다음 예의 결과는 '1988-12-25'의 내부 표현입니다.

```
DATE(RECEIVED)
```

- 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')
```

- 이 예는 '0001-02-04' 내부 표현을 작성합니다.

```
DATE(35)
```

DAY

스키마는 SYSIBM입니다.

DAY 함수는 값의 일 부분을 리턴합니다.

인수는 날짜, 시간소인, 날짜 기간, 시간소인 지속 기간 또는 CLOB가 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 일 부분으로 1과 31 사이의 정수입니다.
- 인수가 날짜 기간이거나 시간소인 지속 기간인 경우:
  - 결과는 값의 일부분으로 -99와 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예를 들면, 다음과 같습니다.

- PROJECT 테이블을 사용하여 호스트 변수 END_DAY(smallint)를 WELD LINE PLANNING 프로젝트(PROJNAME)가 중지하도록 스케줄된 날짜(PRENDATE)로 설정하십시오.

  ```
  SELECT DAY(PRENDATE)
    INTO :END_DAY
     FROM PROJECT
    WHERE PROJNAME = 'WELD LINE PLANNING'
  ```

  샘플 테이블을 사용할 때 END_DAY의 결과는 15로 설정됩니다.

- DATE1 컬럼(날짜)에 2000-03-15에 해당되는 내부 값이 있고 DATE2 컬럼(날짜)에는 1999-12-31에 해당되는 내부 값이 있다고 가정하십시오.

  ```
  DAY(DATE1 - DATE2)
  ```

  결과 값은 15입니다.

DAYNAME



스키마는 SYSIBM입니다. DAYNAME 함수의 SYSFUN 버전은 계속 사용 가능합니다.

DAYNAME 함수는 *locale-name* 또는 CURRENT LOCALE LC_TIME 특수 레지스터 값을 기반으로, *expression*의 일 부분에 대한 일 이름(예: 금요일)을 포함하는 문자열을 리턴합니다.

expression
내장 데이터 유형인 날짜 또는 시간소인 중 하나의 값을 리턴하는 *expression*입니다.

locale-name
결과의 언어를 판별하는 데 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자를 구분하지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 이름에 대해서는 ″SQL 및 XQuery에 대한 로케일 이름″을 참조하십시오. *locale-name*이 지정되지 않으면, 특수 레지스터의 값 CURRENT LOCALE LC_TIME이 사용됩니다.

결과는 가변 길이 문자열입니다. 길이 속성은 100입니다. 결과 문자열이 결과의 길이 속성을 초과하면, 결과가 절단됩니다. *expression* 인수가 널(NULL)일 수 있는 경우 결과는 널(NULL)일 수 있습니다. *expression* 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다. 결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

주

- **Julian 및 Gregorian 달력**: 1582년 10월 15일 이 함수에 의해 Julian 달력에서 Gregorian 달력으로 전이가 고려되었습니다. 그러나 DAYNAME 함수의 SYSFUN 버전은 모든 계산에 대해 Gregorian 달력을 사용한다고 가정합니다.
- **결정론**: DAYNAME은 결정 함수입니다. 그러나 *locale-name*이 명시적으로 지정되지 않으면, 함수의 호출은 특수 레지스터 CURRENT LOCALE LC_TIME의 값에 따라 다릅니다. 특수 레지스터의 값에 따라 다른 이 호출은 특수 레지스터를 사용할 수 없을 때에는 사용할 수 없습니다(SQLSTATE 42621 또는 428EC).

예 :

- 변수 TMSTAMP가 TIMESTAMP로 정의되고 다음 값을 가지고 있다고 가정합니다. 2007–03–09–14.07.38.123456. 다음 예는 함수의 여러 호출 및 결과로 발생되는 문자열 값을 나타냅니다. 각 경우의 결과 유형은 VARCHAR(100)입니다.

```
Function invocation                          Result
---------------------------                  ----------
DAYNAME (TMSTAMP, 'CLDR 1.5:en_US')          Friday
DAYNAME (TSMTAMP, 'CLDR 1.5:de_DE')          Freitag
DAYNAME (TMSTAMP, 'CLDR 1.5:fr_FR')          vendredi
```

DAYOFWEEK

스키마는 SYSFUN입니다.

DAYOFWEEK 함수는 인수에 있는 요일을 1 - 7 범위 내의 정수 값으로 리턴하며, 1은 일요일을 나타냅니다.

인수는 날짜, 시간소인, 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

DAYOFWEEK_ISO

스키마는 SYSFUN입니다.

인수에 있는 요일을 1 - 7 범위 내의 정수 값으로 리턴합니다. 여기서, 1은 월요일을 나타냅니다.

인수는 날짜, 시간소인 또는 CLOB가 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

DAYOFYEAR



스키마는 SYSFUN입니다.

인수에 있는 연도의 날짜를 1 - 366 범위 내의 정수 값으로 리턴합니다.

인수는 날짜, 시간소인 또는 CLOB가 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

DAYS

스키마는 SYSIBM입니다.

DAYS 함수는 날짜의 정수 표현을 리턴합니다.

인수는 날짜, 시간소인 또는 CLOB가 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

결과는 0001년 1월 1일에서 *D*까지의 일 수보다 1 많은 값입니다. 여기서 *D*는 DATE 함수가 인수에 적용된 경우 발생하는 날짜입니다.

예를 들면, 다음과 같습니다.

- PROJECT 테이블을 사용하여 호스트 변수 EDUCATION_DAYS(int)를 프로젝트(PROJNO) 'IF2000'에 대해 예측된 경과 일 수(PRENDATE - PRSTDATE)로 설정하십시오.

  ```
  SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
    INTO :EDUCATION_DAYS
     FROM PROJECT
    WHERE PROJNO = 'IF2000'
  ```

  EDUCATION_DAYS의 결과는 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 설정하십시오.

  ```
  SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
    INTO :TOTAL_DAYS
     FROM PROJECT
    WHERE DEPTNO = 'E21'
  ```

  샘플 테이블을 사용할 때 TOTAL_DAYS 결과는 1584로 설정됩니다.

DBCLOB



스키마는 SYSIBM입니다.

DBCLOB 함수는 그래픽 문자열 유형의 DBCLOB 표현을 리턴합니다.

유니코드 데이터베이스의 경우 제공된 인수가 문자열이면 함수를 실행하기 전에 먼저 그래픽 문자열로 변환됩니다. 출력 문자열이 절단되면 마지막 문자는 대용 가능성이 높으며 다음 중 하나로 대용됩니다.

- 제공된 인수가 문자열인 경우 그대로 남겨둡니다.
- 제공된 인수가 그래픽 문자열인 경우 공백 문자(X'0020')로 변환됩니다.

추후 릴리스에서 변경될 수 있으므로 이 동작에 의존하지 마십시오.

함수의 결과는 DBCLOB입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

graphic-expression
그래픽 문자열인 값을 리턴하는 표현식입니다.

integer
결과로 발생하는 DBCLOB 데이터 유형의 길이 속성을 지정하는 정수 값. 값은 0 및 1 073 741 823 사이여야 합니다. *integer*를 지정하지 않으면 결과 길이는 첫 번째 인수의 길이와 같습니다.

DBPARTITIONNUM

스키마는 SYSIBM입니다.

DBPARTITIONNUM 함수는 행의 데이터베이스 파티션 번호를 리턴합니다. 예를 들어, SELECT절에 사용하는 경우 결과 세트에 있는 각 행의 데이터베이스 파티션 번호를 리턴합니다.

인수는 테이블의 컬럼에 대해 규정된 이름 또는 규정되지 않은 이름일 수 있습니다. 행 레벨 정보가 리턴되므로 지정된 컬럼에 관계 없이 결과가 동일합니다. 컬럼의 데이터 유형에는 제한이 없습니다.

*column-name*에서 뷰의 컬럼을 참조할 경우, 뷰의 컬럼에 대한 표현식은 기본 테이블의 컬럼을 참조해야 하며 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다.

데이터베이스 파티션 번호가 DBPARTITIONNUM 함수에 의해 리턴되는 특정 행(및 테이블)은 함수를 사용하는 SQL문의 컨텍스트에서 결정됩니다.

전이 변수 및 테이블에서 리턴되는 데이터베이스 파티션 번호는 분산 키 컬럼의 현재 전이 값에서 파생됩니다. 예를 들어, 사전 삽입 트리거에서 새 전이 변수의 현재 값을 제공하면 이 함수는 프로젝트된 데이터베이스 파티션 번호를 리턴합니다. 그러나 분산 키 컬럼 값은 이후의 사전 삽입 트리거에서 수정될 수 있습니다. 따라서 데이터베이스에 삽입될 때 행의 최종 데이터베이스 파티션 번호는 프로젝트된 값과 다를 수 있습니다.

결과의 데이터 유형은 INTEGER이며 절대 널(NULL)이 될 수 없습니다. `db2nodes.cfg` 파일이 없는 경우 결과는 0입니다.

이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 함수가 모든 데이터 유형을 인수로 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

DBPARTITIONNUM 함수는 점검 제한조건 내의 복제된 테이블이나 생성된 컬럼의 정의에서 사용할 수 없습니다(SQLSTATE 42881).

이전 버전의 DB2 제품과의 호환을 위해 DBPARTITIONNUM 대신 NODENUMBER를 지정할 수 있습니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블에 있는 제공된 직원의 행이 DEPARTMENT 테이블에 있는 직원 부서의 설명과 다른 데이터베이스 파티션에 있는 인스턴스 수를 계산하십시오.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- 두 테이블의 행이 동일한 데이터베이스 파티션에 있도록 EMPLOYEE 및 DEPARTMENT 테이블을 조인하십시오.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- EMPLOYEE 테이블의 BEFORE 트리거를 사용하여 이름이 EMPINSERTLOG1인 테이블에서 EMPLOYEE 테이블의 새 행에 대한 프로젝트된 데이터베이스 파티션 번호를 로그하십시오.

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

DECFLOAT

숫자를 10진수의 부동 소수점으로:



문자를 10진수의 부동 소수점으로:



스키마는 SYSIBM입니다.

DECFLOAT 함수는 숫자의 10진 부동 소수점 표시나 숫자의 문자열 표시를 리턴합니다.

numeric-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

string-expression
문자 상수의 최대 길이보다 길지 않은 길이의 숫자의 문자열 또는 유니코드 그래픽 문자열 표현인 값을 리턴하는 표현식입니다. string-expression의 데이터 유형은 CLOB, 또는 DBCLOB일 수 없습니다(SQLSTATE 42884). 앞뒤 공백은 문자열에서 제거됩니다. 결과 서브스트링은 대문자로 접히며, 정수, 10진수, 부동 소수점 또는 10진 부동 소수점 상수(SQLSTATE 22018) 형성 규칙을 준수하고 42바이트보다 크지 않아야 합니다(SQLSTATE 42820).

34 또는 16
결과 정밀도의 자릿수를 지정합니다. 디폴트는 34입니다.

decimal-character
수의 전체 부분에서 *character-expression*의 10진 숫자를 구분하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자는 숫자, 플러스(+), 마이너스(-) 또는 공백일 수 없으며 *character-expression*에 최대 한 번 표시될 수 있습니다.

결과는 CAST(*string-expression* AS DECFLOAT(*n*)) 또는 CAST(*numeric-expression* AS DECFLOAT(*n*))의 결과와 같은 숫자입니다. 앞뒤 공백은 문자열에서 제거됩니다.

함수의 결과는 정밀도 자릿수가 내재적으로나 명시적으로 지정되는 10진 부동 소수점 숫자입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

필요한 경우 소스는 대상의 정밀도로 라운드됩니다. CURRENT DECFLOAT ROUNDING MODE 특수 레지스터는 반올림 모드를 판별합니다.

주

- CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.
- 모든 숫자 값은 정수, 10진수 또는 부동 소수점 상수로 해석된 후 10진 부동 소수점으로 캐스트됩니다. 부동 소수점 상수를 사용하면 라운드 오프 오류가 발생하므로 사용하지 않도록 하십시오. 대신 DECFLOAT 함수의 문자열 대 10진 부동 소수점 버전을 사용하십시오.

예

- EMPLOYEE 테이블에서 EDLEVEL 컬럼(데이터 유형 = SMALLINT)에 대한 선택 목록에서 DECFLOAT 데이터 유형이 리턴되도록 하려면 DECFLOAT 함수를 사용하십시오. EMPNO 컬럼도 선택 목록에 나타나야 합니다.

```
SELECT EMPNO, DECFLOAT(EDLEVEL,16)
FROM EMPLOYEE
```

DECFLOAT_FORMAT



스키마는 SYSIBM입니다.

DECFLOAT_FORMAT 함수는 지정된 형식을 사용하여 입력 문자열 해석을 기반으로 하는 DECFLOAT(34) 값을 리턴합니다.

string-expression

CHAR 및 VARCHAR 데이터 유형인 값을 리턴하는 표현식입니다. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다. 앞뒤 공백은 문자열에서 제거됩니다. *format-string*을 지정하지 않는 경우, 결과 서브스트링은 SQL 정수, 10진수, 부동 소수점 또는 10진 부동 소수점 상수(SQLSTATE 22018) 형성 규칙을 준수하고 42바이트보다 크면 안됩니다(SQLSTATE 42820). 그렇지 않으면 결과 서브스트링은 *format-string*에 지정된 형식에 해당되는 숫자 구성요소를 포함해야 합니다(SQLSTATE 22018).

format-string

내장 문자열 데이터 유형(CLOB 제외)인 값을 리턴하는 표현식입니다. 유니코드 데이터베이스의 경우, 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면, 함수를 평가하기 전에 먼저 문자열로 변환됩니다. 실제 길이는 254보다 크지 않아야 합니다(SQLSTATE 22018). 값은 *string-expression*이 DECFLOAT 값으로의 변환에 대해 해석되는 방식에 대한 템플리트입니다. 접두부로 지정된 형식 요소는 템플리트 맨 앞에서만 사용할 수 있습니다. 접미부로 지정된 형식 요소는 템플리트 끝에서만 사용할 수 있습니다. 형식 요소는 대소문자를 구분합니다. 템플리트는 MI, S 또는 PR 형식 요소 중 두 개 이상을 포함할 수 없습니다(SQLSTATE 22018).

표 44. DECFLOAT_FORMAT 함수의 형식 요소

형식 요소	설명
0 또는 9	0 또는 9 각각은 숫자를 나타냅니다.
MI	접미부: *string-expression*이 음수를 나타내기 위한 것이면 뒤에 마이너스 부호(–)가 예상됩니다. *string-expression*이 양수를 나타내기 위한 것이면 뒤 공백을 지정할 수 있습니다.
S	접두부: *string-expression*이 음수를 나타내기 위한 것이면 앞에 마이너스 부호(–)가 예상됩니다. *string-expression*이 양수를 나타내기 위한 것이면 앞 플러스 부호(+) 또는 앞 공백을 지정할 수 있습니다.
PR	접미부: *string-expression*이 음수를 나타내기 위한 것이면 앞에 미만(<), 뒤에 초과(>) 부호가 예상됩니다. *string-expression*이 양수를 나타내기 위한 것이면 앞 공백과 뒤 공백을 지정할 수 있습니다.
$	접두부: 앞에 달러 부호($)로 지정되어야 합니다.
,	쉼표의 예상 위치를 지정합니다. 이 쉼표는 그룹 구분자로 사용됩니다.

표 44. DECFLOAT_FORMAT 함수의 형식 요소 (계속)

형식 요소	설명
.	마침표의 예상 위치를 지정합니다. 이 마침표는 소수점으로 사용됩니다.

*format-string*을 지정하지 않는 경우, *string-expression*은 SQL 정수, 10진수, 부동 소수점 또는 10진 부동 소수점 상수(SQLSTATE 22018) 형성 규칙을 준수하고 길이는 42바이트보다 길면 안됩니다(SQLSTATE 42820).

결과는 DECFLOAT(34)입니다. 첫 번째 또는 두 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 첫 번째 또는 두 번째 인수가 널(NULL)일 경우, 결과는 널(NULL) 값입니다.

주

- 구문 대체: TO_NUMBER는 DECFLOAT_FORMAT의 동의어입니다.

예

- `DECFLOAT_FORMAT( '123.45' )`

  123.45를 리턴합니다.
- `DECFLOAT_FORMAT( '-123456.78' )`

  -123456.78을 리턴합니다.
- `DECFLOAT_FORMAT( '+123456.78' )`

  123456.78을 리턴합니다.
- `DECFLOAT_FORMAT( '1.23E4' )`

  12300을 리턴합니다.
- `DECFLOAT_FORMAT( '123.4', '9999.99' )`

  123.40을 리턴합니다.
- `DECFLOAT_FORMAT( '001,234', '000,000' )`

  1234를 리턴합니다.
- `DECFLOAT_FORMAT( '1234 ', '9999MI' )`

  1234를 리턴합니다.
- `DECFLOAT_FORMAT( '1234-', '9999MI' )`

  -1234를 리턴합니다.
- `DECFLOAT_FORMAT( '+1234', 'S9999' )`

  1234를 리턴합니다.
- `DECFLOAT_FORMAT( '-1234', 'S9999' )`

-1234를 리턴합니다.

- **DECFLOAT_FORMAT**(' 1234 ', '9999PR')

  1234를 리턴합니다.

- **DECFLOAT_FORMAT**('<1234>', '9999PR')

  -1234를 리턴합니다.

- **DECFLOAT_FORMAT**('$123,456.78', '$999,999.99')

  123456.78을 리턴합니다.

DECIMAL 또는 DEC

숫자에서 10진수로:



문자열에서 10진수로:



날짜 시간에서 10진수로:



스키마는 SYSIBM입니다.

DECIMAL 함수는 다음의 10진수 표현을 리턴합니다.

- 숫자
- 숫자의 문자열 표현
- 날짜 시간 값

숫자에서 10진수로

numeric-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

precision
값의 범위가 1 - 31인 정수 상수입니다.

*precision*의 디폴트값은 *numeric-expression*의 데이터 유형에 따라 다릅니다.

- 10진수 부동 소수점의 경우에는 31
- 부동 소수점 및 10진수의 경우에는 15
- 대정수(bigint)의 경우에는 19
- 정수(integer)의 경우에는 11
- 작은 정수(smallint)의 경우에는 5

scale

범위가 0 - *precision* 값인 정수 상수입니다. 디폴트값은 0입니다.

결과는 정밀도가 *p*이고 스케일이 *s*인 10진수 컬럼이나 변수에 첫 번째 인수를 지정한 경우 발생하는 수와 동일합니다. 여기서, *p*와 *s*는 각각 선택적인 두 번째와 세 번째 인수입니다. 수의 정수 부분을 나타내는 데 필요한 유효 10진수 숫자가 *p-s*보다 크면 오류가 발생합니다.

문자열에서 10진수로

string-expression

문자 상수의 최대 길이보다 길지 않은 길이의 숫자의 문자열이나 유니코드 그래픽 문자열 표현인 값을 리턴하는 *expression*입니다. *string-expression*의 데이터 유형은 CLOB 또는 DBCLOB일 수 없습니다(SQLSTATE 42884). 앞뒤 공백은 문자열에서 제거되고 결과 문자열은 정수, 10진수, 부동 소수점이나 10진수 부동 소수점 상수에 대한 규칙에 따라야 합니다(SQLSTATE 22018).

*string-expression*은 필요에 따라 상수 *decimal-character*의 코드 페이지와 일치하도록 섹션 코드 페이지로 변환됩니다.

precision

결과의 정밀도를 지정하는 1 - 31 범위의 값을 갖는 정수 상수입니다. 지정하지 않으면 디폴트값은 15입니다.

scale

결과의 스케일을 지정하는 0 - *precision* 범위의 값을 갖는 정수 상수입니다. 지정하지 않으면 디폴트값은 0입니다.

decimal-character

수의 전체 부분에서 *string-expression*의 10진 숫자를 구분하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자는 숫자, 더하기(+), 빼기(-) 또는 공백일 수 없으며 *string-expression*에 최대 한 번 나타날 수 있습니다(SQLSTATE 42815).

결과는 CAST(*string-expression* AS DECIMAL(*p*,*s*))의 결과의 수와 동일하며, 여기서 *p*와 *s*는 각각 선택적 두 번째와 세 번째 인수입니다. 10진 문자 오른쪽의 숫자가 스케일보다 크면 10진수 끝에서 숫자가 절단됩니다. *string-expression*에서 10진 문자 왼쪽의 유효 숫자(숫자의 전체 부분)가 *p-s*보다 크면 오류가 발생합니다(SQLSTATE 22003). 디폴트 10진 문자는 *decimal-character* 인수에 대해 다른 값이 지정된 경우에 부속 문자열에서 유효하지 않습니다(SQLSTATE 22018).

Datetime에서 Decimal로

datetime-expression

DATE, TIME 또는 TIMESTAMP 유형 값을 리턴하는 표현식입니다.

precision

결과의 정밀도를 지정하는 1 - 31 범위의 값을 갖는 정수 상수입니다. 지정되어 있지 않으면 정밀도와 스케일의 디폴트는 다음과 같이 *datetime-expression*의 데이터 유형에 따라 다릅니다.

- DATE의 경우 정밀도는 8이고 스케일은 0입니다. 결과는 날짜를 *yyyymmdd*로 표시하는 DECIMALK(8,0) 값입니다.
- TIME의 경우 정밀도는 6이고 스케일은 0입니다. 결과는 시간을 *hhmmss*로 표시하는 DECIMAL(6,0) 값입니다.
- TIMESTAMP(*tp*)의 경우 정밀도는 14+*tp*이고 스케일은 *tp*입니다. 결과는 시간소인을 *yyyymmddhhmmss.nnnnnnnnnnnn*으로 표시하는 DECIMAL(14+*tp*,*tp*) 값입니다.

scale

결과의 스케일을 지정하는 0 - *precision* 범위의 값을 갖는 정수 상수입니다. 스케일은 지정되어 있지 않고 *precision*이 지정되어 있으면 디폴트값은 0입니다.

결과는 정밀도가 *p*이고 스케일이 *s*인 10진수일 수 있습니다. 여기서, *p*와 *s*는 각각 선택적 두 번째와 세 번째 인수입니다. 10진 문자 오른쪽의 숫자가 스케일보다 크면 끝에서 숫자가 절단됩니다. *datetime-expression*에서 10진 문자 왼쪽의 유효 숫자(숫자의 정수 부분)가 *p*-*s*보다 크면 오류가 발생합니다 (SQLSTATE 22003).

첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주: CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예:

- EMPLOYEE 테이블에서 EDLEVEL 컬럼(데이터 유형 = SMALLINT)에 대한 선택 목록에서 DECIMAL 데이터 유형(정밀도가 5이고 스케일이 2임)을 리턴하려면 DECIMAL 함수를 사용하십시오. EMPNO 컬럼도 선택 목록에 나타나야 합니다.

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- 호스트 변수 PERIOD의 유형을 INTEGER로 가정하십시오. 그런 다음 그 값을 날짜 기간으로 사용하기 위해 decimal(8,0)로 "캐스트"해야 합니다.

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
   FROM PROJECT
```

- SALARY 컬럼에 대한 갱신사항이 10진 문자로 쉼표를 사용하는 문자열로 창에서 입력된다고 가정하십시오(예를 들어, 사용자는 21400,50을 입력함). 응용프로그램에서 유효성을 확인하면 CHAR(10)로 정의되는 호스트 변수 *newsalary*에 지정됩니다.

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid;
```

  newsalary 값은 21400.50이 됩니다.

- 디폴트 10진 문자(.)를 값에 추가하십시오.

```
DECIMAL('21400,50', 9, 2, '.')
```

  이것은 마침표(.)가 10진 문자로 지정되었으나 쉼표(,)가 분리문자로 첫 번째 인수에 나타나므로 실패합니다.

- STARTING 컬럼(시간)에 '12:10:00'에 해당하는 내부 값이 있다고 가정하십시오.

```
DECIMAL(STARTING)
```

  결과 값은 121 000입니다.

- RECEIVED 컬럼(시간소인)에 '1988-12-22-14.07.21.136421'에 해당하는 내부 값이 있다고 가정하십시오.

```
DECIMAL(RECEIVED)
```

  결과 값은 19 881 222 140 721.136421입니다.

- 이 예는 다양한 날짜 시간 입력 값에 대해 10진수 결과와 이에 따른 정밀도 및 스케일을 나타냅니다. 연관된 값이 있는 다음 컬럼이 있다고 가정합니다.

컬럼 이름	데이터 유형	값
ENDDT	DATE	2000-03-21
ENDTM	TIME	12:02:21
ENDTS	TIMESTAMP	2000-03-21-12.02.21.123456
ENDTS0	TIMESTAMP(0)	2000-03-21-12.02.21
ENDTS9	TIMESTAMP(9)	2000-03-21-12.02.21.123456789

  다음 표는 다양한 날짜 시간 입력 값에 대해 10진수 결과와 이에 따른 정밀도 및 스케일을 나타냅니다.

DECIMAL(인수)	정밀도 및 스케일	결과
DECIMAL(ENDDT)	(8,0)	20000321.
DECIMAL(ENDDT, 10)	(10,0)	20000321.
DECIMAL(ENDDT, 12, 2)	(12,2)	20000321.00
DECIMAL(ENDTM)	(6,0)	120221.
DECIMAL(ENDTM, 10)	(10,0)	120221.
DECIMAL(ENDTM, 10, 2)	(10,2)	120221.00
DECIMAL(ENDTS)	(20,6)	20000321120221.123456

DECIMAL(인수)	정밀도 및 스케일	결과
DECIMAL(ENDTS, 23)	(23, 0)	20000321120221.
DECIMAL(ENDTS, 23, 4)	(23,4)	20000321120221.1234
DECIMAL(ENDTS0)	(14,0)	20000321120221.
DECIMAL(ENDTS9)	(23,9)	20000321120221.123456789

DECODE



스키마는 SYSIBM입니다.

DECODE 함수는 각 *expression2*를 *expression1*과 비교합니다. *expression1*이 *expression2*와 같거나, *expression1* 및 *expression2* 둘 다가 널(NULL)이면 다음 *result-expresssion*의 값이 리턴됩니다. *expression2*가 *expression1*과 전혀 일치하지 않으면 *else-expression* 값이 리턴됩니다. 그렇지 않으면 널(NULL) 값이 리턴됩니다.

DECODE 함수는 널(NULL) 값 처리를 제외하고 CASE 표현식과 유사합니다.

- *expression1*의 널(NULL) 값은 해당되는 *expression2* 널(NULL) 값과 일치합니다.
- NULL 키워드가 DECODE 함수에서 인수로 사용되는 경우 해당되는 데이터 유형에 캐스트되어야 합니다.

DECODE 표현식의 결과 유형 판별에 대한 규칙은 해당되는 CASE 표현식을 기초로 합니다.

예를 들면, 다음과 같습니다.

다음 DECODE 표현식은

```
DECODE (c1, 7, 'a', 6, 'b', 'c')
```

다음 CASE 표현식과 동일한 결과를 얻습니다.

```
CASE c1
  WHEN 7 THEN 'a'
  WHEN 6 THEN 'b'
  ELSE 'c'
END
```

마찬가지로, 다음 DECODE 표현식은

```
DECODE (c1, var1, 'a', var2, 'b')
```

(여기서, c1, var1 및 var2의 값이 널(NULL) 값이 될 수 있음) 다음 CASE 표현식과 같은 결과를 얻습니다.

```
CASE
  WHEN c1 = var1 OR (c1 IS NULL AND var1 IS NULL) THEN 'a'
  WHEN c1 = var2 OR (c1 IS NULL AND var2 IS NULL) THEN 'b'
  ELSE NULL
END
```

다음 쿼리도 고려하십시오.

```
SELECT ID, DECODE(STATUS, 'A', 'Accepted',
                          'D', 'Denied',
                          CAST(NULL AS VARCHAR(1)), 'Unknown',
                          'Other')
FROM CONTRACTS
```

다음은 CASE 표현식을 사용하는 동일한 명령문입니다.

```
SELECT ID,
  CASE
    WHEN STATUS = 'A' THEN 'Accepted'
    WHEN STATUS = 'D' THEN 'Denied'
    WHEN STATUS IS NULL THEN 'Unknown'
    ELSE 'Other'
  END
FROM CONTRACTS
```

DECRYPT_BIN 및 DECRYPT_CHAR



스키마는 SYSIBM입니다.

DECRYPT_BIN 및 DECRYPT_CHAR 함수는 둘 다 *encrypted-data* 해독의 결과 값을 리턴합니다. 암호 해독에 사용되는 암호는 *password-string-expression* 값이나 SET ENCRYPTION PASSWORD문이 지정하는 암호화 암호값 중 하나입니다. 시스템에서 최상의 보안 레벨을 유지하기 위해 쿼리에서 DECRYPT_BIN 및 DECRYPT_CHAR 함수를 사용하여 명시적으로 암호화 암호를 전달하지 않는 것이 좋습니다. 대신, SET ENCRYPTION PASSWORD문을 사용하여 암호를 설정하고 SET ENCRYPTION PASSWORD문 사용 시 리터럴 문자열이 아닌 호스트 변수 또는 동적 매개변수 표시문자를 사용하십시오.

DECRYPT_BIN 및 DECRYPT_CHAR 함수는 ENCRYPT 함수를 사용하여 암호화되는 값만을 암호 해독할 수 있습니다(SQLSTATE 428FE).

encrypted-data

CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA 값을 완전히 암호화된 데이터 문자열로 리턴하는 표현식입니다. 데이터 문자열은 ENCRYPT 함수를 사용하여 암호화되어야 합니다.

password-string-expression

최소 6바이트에서 127바이트를 넘지 않는 CHAR 또는 VARCHAR 값을 리턴하는 표현식(SQLSTATE 428FC)입니다. 이 표현식은 데이터를 암호화하는 데 사용된 것과 동일한 암호여야 합니다(SQLSTATE 428FD). 암호 인수의 값이 널(NULL)이거나 제공되지 않으면 데이터는 SET ENCRYPTION PASSWORD문이 세션에 대해 설정한 암호화 암호값을 사용하여 암호 해독됩니다(SQLSTATE 51039).

ECRYPT_BIN 함수의 결과는 VARCHAR FOR BIT DATA입니다. DECRYPT_CHAR 함수의 결과는 VARCHAR입니다. *encrypted-data*에 힌트가 포함될 경우 힌트는 함수에서 리턴되지 않습니다. 결과의 길이 속성은 *encrypted-data*의 데이터 유형 길이에서 8바이트를 뺀 것과 같습니다. 함수에서 리턴되는 값의 실제 길이는 암호화된 원래 문자열의 길이와 일치합니다. *encrypted-data*에 암호화 문자열 이후에 바이트가 포함되는 경우 이러한 바이트는 함수에서 리턴되지 않습니다.

첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

데이터가 암호화된 코드 페이지와는 다른 코드 페이지를 사용하는 다른 시스템에서 암호 해독되는 경우, 암호 해독된 값을 데이터베이스 코드 페이지로 변환할 때 확장이 발생할 수 있습니다. 이 경우 *encrypted-data* 값은 더 큰 바이트 수를 갖는 VARCHAR 문자열로 캐스트되어야 합니다.

예

다음 예에서는 Embedded SQL 응용프로그램의 코드 단편을 표시하여 DECRYPT_CHAR 함수 사용에 대해 설명합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
      char hostVarCreateTableStmt[100];
      char hostVarSetEncPassStmt[200];
      char hostVarPassword[128];
      char hostVarInsertStmt1[200];
      char hostVarInsertStmt2[200];
      char hostVarSelectStmt1[200];
      char hostVarSelectStmt2[200];
EXEC SQL END DECLARE SECTION;
/* prepare the statement */
strcpy(hostVarCreateTableStmt, "CREATE TABLE EMP (SSN VARCHAR(24) FOR BIT DATA)");
EXEC SQL PREPARE hostVarCreateTableStmt FROM :hostVarCreateTableStmt;

/* execute the statement */
EXEC SQL EXECUTE hostVarCreateTableStmt;
```

SET ENCRYPTION PASSWORD문을 사용하여 세션에 대한 암호화된 암호를 설정하십시오.

```
/* prepare the statement with a parameter marker */
strcpy(hostVarSetEncPassStmt, "SET ENCRYPTION PASSWORD = ?");
EXEC SQL PREPARE hostVarSetEncPassStmt FROM :hostVarSetEncPassStmt;

/* execute the statement for hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarSetEncPassStmt USING :hostVarPassword;

/* prepare the statement */
strcpy(hostVarInsertStmt1, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832')");
EXEC SQL PREPARE hostVarInsertStmt1 FROM :hostVarInsertStmt1;

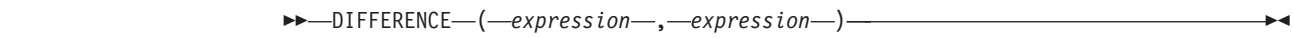
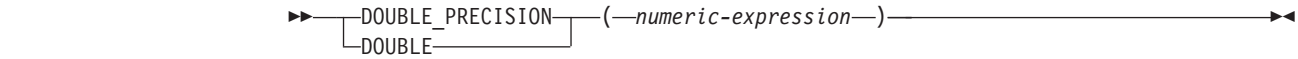
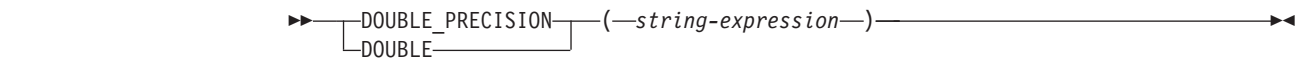
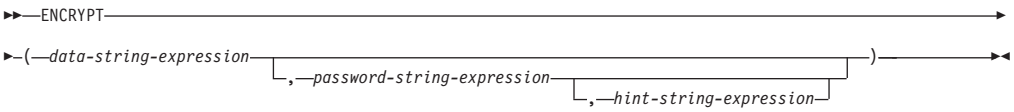
/* execute the statement */
EXEC SQL EXECUTE hostVarInsertStmt1;

/* prepare the statement */
strcpy(hostVarSelectStmt1, "SELECT DECRYPT_CHAR(SSN) FROM EMP");
EXEC SQL PREPARE hostVarSelectStmt1 FROM :hostVarSelectStmt1;

/* execute the statement */
EXEC SQL EXECUTE hostVarSelectStmt1;
```

이 쿼리의 리턴값은 '289-46-8832'입니다.

명시적으로 암호화된 암호를 전달하십시오.

```
/* prepare the statement */
strcpy(hostVarInsertStmt2, "INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832',?)");
EXEC SQL PREPARE hostVarInsertStmt2 FROM :hostVarInsertStmt2;

/* execute the statement for hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
```

```
EXEC SQL EXECUTE hostVarInsertStmt2 USING :hostVarPassword;

/* prepare the statement */
strcpy(hostVarSelectStmt2, "SELECT DECRYPT_CHAR(SSN,?) FROM EMP");
EXEC SQL PREPARE hostVarSelectStmt2 FROM :hostVarSelectStmt2;

/* execute the statement for hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarSelectStmt2 USING :hostVarPassword;
```

이 쿼리의 리턴값은 '289-46-8832'입니다.

DEGREES

스키마는 SYSIBM입니다. (DEGREES 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

DEGREES 함수는 인수의 등급 수를 리턴하며, 이는 라디안으로 표시되는 각도입니다.

인수는 내장 숫자 데이터 유형이면 어느 것이나 가능합니다. 인수가 10진수 부동 소수점이면 연산은 10진수 부동 소수점으로 수행됩니다. 기타의 경우 인수는 배정밀도 부동 소수점 숫자로 변환되어 함수가 처리됩니다.

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예:

- RAD가 값이 3.142인 DECIMAL(4,3) 호스트 변수라고 가정해 보십시오.

  ```
  VALUES DEGREES(:RAD)
  ```

  근사 값 180.0을 리턴합니다.

DEREF

DEREF 함수는 인수의 목표 유형 인스턴스를 리턴합니다.

인수는 정의된 범위를 가지고 있는 참조 데이터 유형의 값이 될 수 있습니다(SQLSTATE 428DT).

결과의 정적 데이터 유형은 인수의 목표 유형입니다. 결과의 동적 데이터 유형은 인수에 대한 목표 유형의 하위 유형입니다. 결과는 널(NULL)이 될 수 있습니다. 결과는 *expression*이 널(NULL) 값인 경우, 또는 *expression*이 목표 테이블에 일치하는 OID가 없는 참조인 경우 널(NULL) 값입니다.

결과는 참조에 대한 목표 유형의 하위 유형 인스턴스입니다. 결과는 참조 값과 일치하는 오브젝트 ID를 가지고 있는 참조의 목표 뷰 또는 목표 테이블 행을 찾아서 판별됩니다. 이 행의 유형은 결과의 동적 유형을 판별합니다. 결과 유형이 목표 테이블이나 목표 뷰의 서브테이블 또는 서브뷰의 행에 기초한 것일 수 있으므로, 명령문의 권한 부여 ID에는 목표 테이블 및 모든 서브테이블 또는 목표 뷰 및 모든 서브뷰에 대해 SELECT 특권이 있어야 합니다(SQLSTATE 42501).

예를 들면, 다음과 같습니다.

EMPLOYEE가 EMP 유형의 테이블이고, 해당 ID 컬럼 이름이 EMPID인 것으로 가정합니다. 이 때 다음 쿼리는 EMPLOYEE 테이블(및 해당되는 서브테이블)의 행마다 EMP 유형(또는 해당되는 하위 유형 중 하나)의 오브젝트를 리턴합니다. 이 쿼리를 수행하려면 EMPLOYEE 및 해당되는 모든 서브테이블에 대한 SELECT 특권이 필요합니다.

```
SELECT DEREF(EMPID) FROM EMPLOYEE
```

DIFFERENCE



스키마는 SYSFUN입니다.

문자열에 SOUNDEX 함수를 적용한 것을 기초로 두 문자열 음조 사이의 차이를 표시하는 0 - 4 범위의 값을 리턴합니다. 4 값은 가능한 최상의 음조 일치입니다.

인수는 4000바이트를 초과하지 않는 CHAR 또는 VARCHAR 문자열이 될 수 있습니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다. 함수는 UTF-8로 인코드된 경우에도 ASCII 문자인 것처럼 전달된 데이터를 해석합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

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
```

첫 번째 행에서, 단어가 SOUNDEX와 동일한 결과를 수반하는 반면 두 번째 행에서는 단어가 일부만 유사합니다.

DIGITS

스키마는 SYSIBM입니다.

DIGITS 함수는 숫자의 문자열 표현을 리턴합니다.

표현식은 SMALLINT, INTEGER, BIGINT, DECIMAL, CHAR 또는 VARCHAR 데이터 유형인 값을 리턴해야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다. CHAR 또는 VARCHAR 값은 함수가 평가되기 전에 내재적으로 DECIMAL(31,6)로 캐스트됩니다.

인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

함수의 결과는 스케일과 상관없이 인수의 절대값을 표시하는 고정 길이 문자열입니다. 결과는 기호나 10진수 문자를 포함하지 않습니다. 대신 필요한 경우 문자열을 채우기 위한 선행 영(0)을 포함한 숫자로만 구성됩니다. 문자열의 길이는 다음과 같습니다.

- 인수가 작은 정수인 경우 5
- 인수가 큰 정수인 경우 10
- 인수가 대정수인 경우 19
- 인수가 *p* 정밀도인 10진수인 경우 *p*.

예를 들면, 다음과 같습니다.

- TABLEX라는 테이블은 10자리 숫자를 포함한 INTCOL이라는 INTEGER 컬럼을 포함한다고 가정합니다. 컬럼 INTCOL에 포함된 처음 네 자리의 고유한 4자리 조합을 모두 나열합니다.

  ```
  SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
    FROM TABLEX
  ```

- COLUMNX에 DECIMAL(6,2) 데이터 유형이 있으며 값 중 하나가 -6.28이라고 가정합니다. 이 값인 경우,

  ```
  DIGITS(COLUMNX)
  ```

  값 '000628'을 리턴합니다.

  결과는 길이 6까지 선행 영(0)으로 채운 길이 6의 문자열(컬럼의 정밀도)입니다. 결과에는 기호나 소수점이 나타나지 않습니다.

DOUBLE_PRECISION 또는 DOUBLE

숫자를 double로:



문자열을 double로:



스키마는 SYSIBM입니다.

DOUBLE_PRECISION 및 DOUBLE 함수는 둘 중 하나의 배정밀도 부동 소수점 표현을 리턴합니다.

- 숫자
- 숫자의 문자열 표현

숫자를 double로

numeric-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

결과는 인수가 배정밀도 부동 소수점 컬럼이나 변수에 지정된 경우에 발생하는 수와 동일합니다. 인수의 숫자 값이 배정밀도 부동 소수점 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

문자열을 double로

string-expression
숫자의 문자열 또는 유니코드 그래픽 문자열 표현인 값을 리턴하는 표현식입니다. *string-expression*의 데이터 유형은 CLOB, LONG VARCHAR, DBCLOB 또는 LONG VARGRAPHIC일 수 없습니다(SQLSTATE 42884).

결과는 CAST(*string-expression* AS DOUBLE PRECISION)에서 발생하는 수와 동일합니다. 앞뒤 공백은 제거되고 결과 문자열은 유효한 숫자 상수를 작성하는 규칙에 따라야 합니다(SQLSTATE 22018). 인수의 숫자 값이 배정밀도 부동 소수점 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

참고:

- CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.
- FLOAT는 DOUBLE_PRECISION 및 DOUBLE과 동의어입니다.
- DOUBLE(*string_expression*)의 SYSFUN 버전은 계속 사용 가능합니다.

예:

EMPLOYEE 테이블을 사용하여 수당이 0이 아닌 직원들의 수당에 대한 급여 비율을 찾습니다. 관련된 컬럼(SALARY 및 COMM)은 DECIMAL 데이터 유형을 가집니다. 범위 밖의 결과가 발생할 수 있는 가능성을 없애기 위해 부동 소수점에서 나누기가 수행되도록 DOUBLE이 SALARY에 적용됩니다.

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

EMPTY_BLOB, EMPTY_CLOB 및 EMPTY_DBCLOB

스키마는 SYSIBM입니다.

값이 비어 있는 함수는 연관된 데이터 유형의 0 길이 값을 리턴합니다. 이 함수에 대한 인수가 없습니다(비어 있는 괄호를 지정해야 함).

- EMPTY_BLOB 함수는 BLOB(1) 데이터 유형의 0 길이 값을 리턴합니다.
- EMPTY_CLOB 함수는 데이터 유형이 CLOB(1)인 0 길이 값을 리턴합니다.
- EMPTY_DBCLOB 함수는 데이터 유형이 DBCLOB(1)인 0 길이 값을 리턴합니다.

이 함수의 결과는 필요한 경우에 0 길이 값을 제공하기 위해 지정에서 사용할 수 있습니다.

ENCRYPT



스키마는 SYSIBM입니다.

ENCRYPT 함수는 *data-string-expression* 암호화의 결과 값을 리턴합니다. 암호화에 사용되는 암호는 *password-string-expression* 값이나 SET ENCRYPTION PASSWORD 문이 지정하는 암호화 암호값 중 하나입니다. 시스템에서 최상의 보안 레벨을 유지하기 위해 쿼리에서 ENCRYPT 함수를 사용하여 명시적으로 암호화 암호를 전달하지 않는 것이 좋습니다. 대신, SET ENCRYPTION PASSWORD문을 사용하여 암호를 설정하고 SET ENCRYPTION PASSWORD문 사용 시 리터럴 문자열이 아닌 호스트 변수 또는 동적 매개변수 표시문자를 사용하십시오.

유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

data-string-expression

암호화될 CHAR 또는 VARCHAR 값을 리턴하는 표현식입니다. *data-string-expression*의 데이터 유형에 대한 길이 속성은 *hint-string-expression* 인수 없이 32663으로, *hint-string-expression* 인수가 지정되면 32631로 제한됩니다(SQLSTATE 42815).

password-string-expression

최소 6바이트에서 127바이트를 넘지 않는 CHAR 또는 VARCHAR 값을 리턴하는 표현식(SQLSTATE 428FC)입니다. 이 값은 *data-string-expression*을 암호화하는 데 사용되는 암호를 나타냅니다. 암호 인수 값이 널(NULL)이거나 제공되지 않으면 데이터는 SET ENCRYPTION PASSWORD 문이 세션에 대해 지정한 암호화 암호값을 사용하여 암호화됩니다(SQLSTATE 51039).

hint-string-expression

데이터 소유자가 암호를 기억하는 데 도움이 되는 최대 32바이트의 CHAR 또는 VARCHAR 값을 리턴하는 표현식(예: 'Pacific'을 기억하기 위한 힌트로 'Ocean') 입니다. 힌트 값이 제공되면 힌트는 결과에 임베드되며 GETHINT 함수를 사용하여 검색할 수 있습니다. 이 인수가 널(NULL)이거나 제공되지 않으면 결과에 임베드될 힌트는 없습니다.

이 함수의 결과 데이터 유형은 VARCHAR FOR BIT DATA입니다.

- 선택적 힌트 매개변수가 지정될 경우, 결과의 길이 속성은 암호화되지 않는 데이터의 길이 속성 + 8바이트 + 다음 8바이트 바운더리까지의 바이트 수 + 힌트 길이 32바이트입니다.

- 선택적 힌트 매개변수가 지정되지 않을 경우, 결과의 길이 속성은 암호화되지 않는 데이터의 길이 속성 + 8바이트 + 다음 8바이트 바운더리까지의 바이트 수입니다.

첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

암호화된 결과가 *data-string-expression* 값보다 더 길다는 점에 유의하십시오. 따라서 암호화된 값을 지정할 때, 목표가 전체 암호화된 값을 포함할 수 있는 충분한 크기로 선언되도록 하십시오.

주

- ***암호화 알고리즘:*** 내부 암호화 알고리즘은 채우기가 있는 RC2 블록 암호이며, 128 비트 비밀 키가 MD5 메시지 요약을 사용하여 암호에서 파생됩니다.
- ***암호화 암호 및 데이터:*** 암호 관리는 사용자의 책임입니다. 일단 데이터가 암호화되면 데이터를 암호화하는 데 사용된 암호를 사용해야만 데이터의 암호를 해독할 수 있습니다(SQLSTATE 428FD).

  암호화된 결과는 널(NULL) 종료자와 기타 인쇄 불가능한 문자를 포함할 수 있습니다. 제안된 데이터 길이보다 더 짧은 길이에 대한 모든 지정 또는 캐스트는 나중에 암호 해독 실패와 데이터 유실을 가져올 수 있습니다. 공백은 너무 짧은 컬럼에 저장될 때 절단될 수 있는 유효한 암호화 데이터 값입니다.
- ***암호화된 데이터의 관리:*** 암호화된 데이터는 ENCRYPT 함수에 해당하는 암호 해독 함수를 지원하는 서버에서만 암호 해독할 수 있습니다. 그러므로 암호화된 데이터를 갖는 컬럼의 복제는 DECRYPT_BIN 또는 DECRYPT_CHAR 함수를 지원하는 서버에서만 수행되어야 합니다.

예

다음 예에서는 Embedded SQL 응용프로그램의 코드 단편을 표시하여 ENCRYPT 함수 사용에 대해 설명합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
      char hostVarCreateTableStmt[100];
      char hostVarSetEncPassStmt[200];
      char hostVarPassword[128];
      char hostVarInsertStmt1[200];
      char hostVarInsertStmt2[200];
      char hostVarInsertStmt3[200];
EXEC SQL END DECLARE SECTION;
/* prepare the statement */
strcpy(hostVarCreateTableStmt, "CREATE TABLE EMP (SSN VARCHAR(24) FOR BIT DATA)");
EXEC SQL PREPARE hostVarCreateTableStmt FROM :hostVarCreateTableStmt;

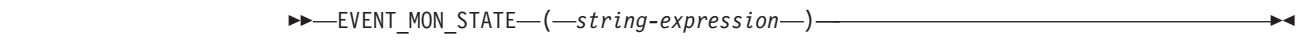
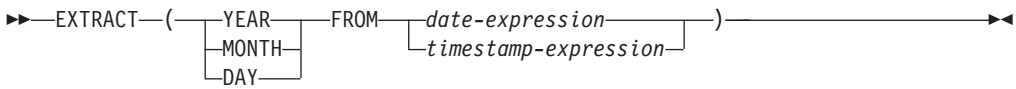
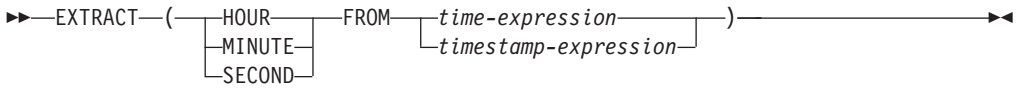
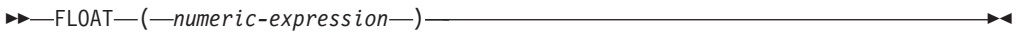
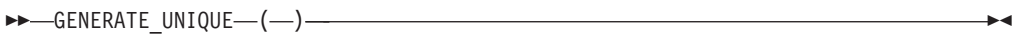
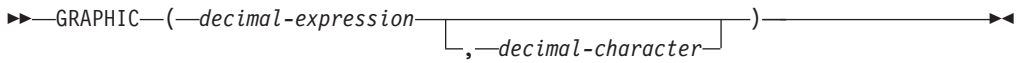
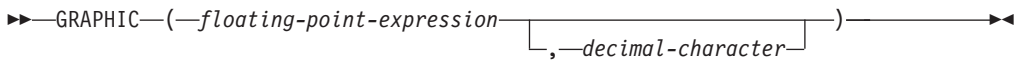
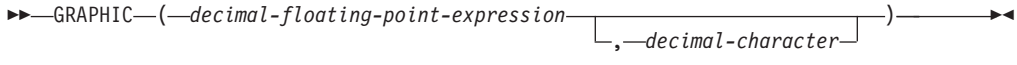
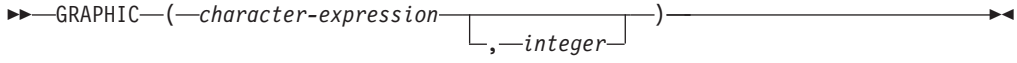
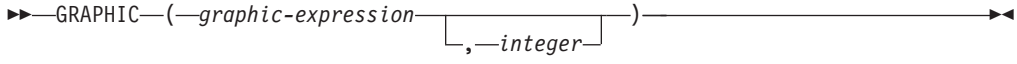
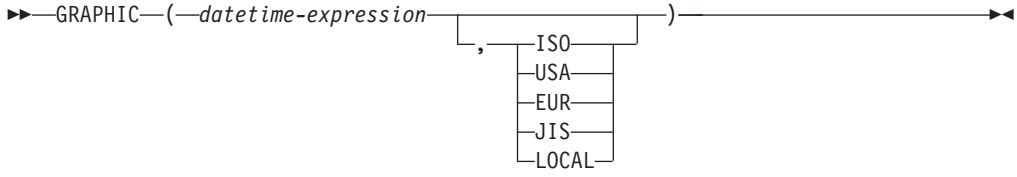
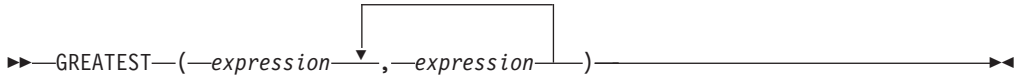
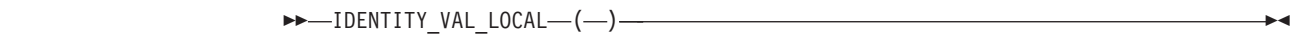
/* execute the statement */
EXEC SQL EXECUTE hostVarCreateTableStmt;
```

SET ENCRYPTION PASSWORD문을 사용하여 세션에 대한 암호화된 암호를 설정하십시오.

```
/* prepare the statement with a parameter marker */
strcpy(hostVarSetEncPassStmt, "SET ENCRYPTION PASSWORD = ?");
EXEC SQL PREPARE hostVarSetEncPassStmt FROM :hostVarSetEncPassStmt;

/* execute the statement for hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarSetEncPassStmt USING :hostVarPassword;

/* prepare the statement */
strcpy(hostVarInsertStmt1, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832')");
EXEC SQL PREPARE hostVarInsertStmt1 FROM :hostVarInsertStmt1;

/* execute the statement */
EXEC SQL EXECUTE hostVarInsertStmt1;
```

명시적으로 암호화된 암호를 전달하십시오.

```
/* prepare the statement */
strcpy(hostVarInsertStmt2, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832',?)");
EXEC SQL PREPARE hostVarInsertStmt2 FROM :hostVarInsertStmt2;

/* execute the statement for hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarInsertStmt2 USING :hostVarPassword;
```

암호 힌트를 정의하십시오.

```
/* prepare the statement */
strcpy(hostVarInsertStmt3, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832',?,'Ocean')");
EXEC SQL PREPARE hostVarInsertStmt3 FROM :hostVarInsertStmt3;

/* execute the statement for hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarInsertStmt3 USING :hostVarPassword;
```

EVENT_MON_STATE



스키마는 SYSIBM입니다.

EVENT_MON_STATE 함수는 이벤트 모니터의 현재 상태를 리턴합니다.

인수는 결과 유형이 CHAR 또는 VARCHAR이고 값이 이벤트 모니터의 이름인 문자열 표현식입니다. 이름 지정된 이벤트 모니터가 SYSCAT.EVENTMONITORS 카탈로그 테이블에 존재하지 않으면 SQLSTATE 42704가 리턴됩니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

결과는 다음 값 중 하나의 정수입니다.

0 이벤트 모니터가 비활성 상태입니다.

1 이벤트 모니터가 활성 상태입니다.

인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예:

다음 예는 정의된 모든 이벤트 모니터를 선택하고 각각의 활성 또는 비활성 상태 여부를 표시합니다.

```
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 SYSIBM입니다. (EXP 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

EXP 함수는 인수가 지정한 값으로 거듭제곱된 자연 대수(e)의 기본 값을 리턴합니다. EXP 및 LN 함수는 역연산입니다.

인수는 내장 숫자 데이터 유형 값을 리턴하는 표현식이어야 합니다. 인수가 10진수 부동 소수점이면 연산은 10진수 부동 소수점으로 수행됩니다. 기타의 경우 인수는 배정밀도 부동 소수점 숫자로 변환되어 함수가 처리됩니다.

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주

DECFLOAT 특수값을 포함하는 결과: 10진수 부동 소수점의 경우, 특수값은 다음과 같이 처리됩니다.

- EXP(NaN)는 NaN을 리턴합니다.
- EXP(-NaN)는 -NaN을 리턴합니다.
- EXP(infinity)는 Infinity를 리턴합니다.
- EXP(-infinity)는 0을 리턴합니다.
- EXP(sNaN)는 NaN과 경고를 리턴합니다.
- EXP(-sNaN)는 -NaN과 경고를 리턴합니다.

예 :

- E가 값 3.453789832인 DECIMAL(10,9) 호스트 변수라고 가정해 보십시오.

```
VALUES EXP(:E)
```

  DOUBLE 값 +3.16200000069145E+001을 리턴합니다.

EXTRACT

날짜 값 추출



시간 값 추출



스키마는 SYSIBM입니다.

EXTRACT 함수는 해당 인수를 기초로 날짜 또는 시간소인 부분을 리턴합니다.

인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

날짜 값 추출

YEAR

date-expression 또는 *timestamp-expression*의 연도 부분이 리턴됨을 지정합니다. 결과는 YEAR 스칼라 함수와 같습니다.

MONTH

date-expression 또는 *timestamp-expression*의 월 부분이 리턴됨을 지정합니다. 결과는 MONTH 스칼라 함수와 같습니다.

DAY

date-expression 또는 *timestamp-expression*의 일 부분이 리턴됨을 지정합니다. 결과는 DAY 스칼라 함수와 같습니다.

date-expression

내장 날짜 또는 내장 문자열 데이터 유형의 값을 리턴하는 표현식입니다.

*date-expression*이 문자 또는 그래픽 문자열이면 CLOB가 아닌 date의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스에서 *date-expression*이 그래픽 문자열인 경우 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

timestamp-expression

내장 시간소인 또는 내장 문자열 데이터 유형의 값을 리턴하는 표현식입니다.

*timestamp-expression*이 문자 또는 그래픽 문자열인 경우 CLOB가 아닌 timestamp의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스에서 *timestamp-expression*이 그래픽 문자열인 경우 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

시간 값 추출

HOUR

time-expression 또는 *timestamp-expression*의 시 부분이 리턴됨을 지정합니다. 결과는 HOUR 스칼라 함수와 같습니다.

MINUTE

time-expression 또는 *timestamp-expression*의 분 부분이 리턴됨을 지정합니다. 결과는 MINUTE 스칼라 함수와 같습니다.

SECOND

time-expression 또는 *timestamp-expression*의 초 부분이 리턴됨을 지정합니다. 결과는 다음과 같습니다.

- SECOND(*expression*, 6) *expression*의 데이터 유형이 TIME 값 또는 TIME 또는 TIMESTAMP의 문자열 표현인 경우
- SECOND(*expression*, *s*), *expression*의 데이터 유형이 TIMESTAMP(*s*) 값인 경우

time-expression

내장 시간 또는 내장 문자열 데이터 유형의 값을 리턴하는 표현식입니다.

*time-expression*이 문자 또는 그래픽 문자열인 경우 CLOB가 아닌 time의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스에서 *time-expression*이 그래픽 문자열인 경우 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

timestamp-expression

내장 날짜, 시간소인 또는 문자열 데이터 유형의 값을 리턴하는 표현식입니다.

*timestamp-expression*이 날짜이면 TIMESTAMP(0) 값으로 변환됩니다(정확하게 자정 시간(00.00.00)을 가정함).

*timestamp-expression*이 문자 또는 그래픽 문자열인 경우 CLOB가 아닌 timestamp 또는 date의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스에서 *timestamp-expression*이 그래픽 문자열인 경우 함수를 실행하기 전에 먼저 문자열로 변환됩니다. 문자열은 TIMESTAMP(6) 값으로 변환됩니다.

함수 결과의 데이터 유형은 지정된 날짜 시간 값 파트에 따라 다릅니다.

- YEAR, MONTH, DAY, HOUR 또는 MINUTE가 지정된 경우 결과의 데이터 유형은 INTEGER입니다.

- TIMESTAMP(*p*) 값을 사용하여 SECOND가 지정된 경우, 결과의 데이터 유형은 DECIMAL(2+*p*, *p*)입니다. 여기서 *p*는 소수 초 정밀도입니다.
- TIME 값 또는 TIME이나 TIMESTAMP의 문자열 표현으로 SECOND가 지정된 경우, 결과의 데이터 유형은 DECIMAL(8,6)입니다.

예 :

- PRSTDATE 컬럼에 1988-12-25에 해당하는 내부 값이 있다고 가정하십시오.

```
SELECT EXTRACT(MONTH FROM PRSTDATE)
   FROM PROJECT;
```

FLOAT



스키마는 SYSIBM입니다.

FLOAT 함수는 숫자의 부동 소수점 표현을 리턴합니다. FLOAT는 DOUBLE의 동의어입니다.

FLOOR

스키마는 SYSIBM입니다. (FLOOR 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수보다 작거나 같은 최대 정수 값을 리턴합니다.

함수의 결과에는 인수가 DECIMAL인 경우 스케일이 0이라는 점을 제외하고는 인수와 동일한 데이터 유형 및 길이 속성이 있습니다. 예를 들어, 데이터 유형이 DECIMAL(5,5)인 인수는 DECIMAL(5,0)을 리턴합니다.

인수가 널(NULL)이 될 수 있거나 인수가 10진수 부동 소수점이 아니고 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

주

DECFLOAT 특수값을 포함하는 결과: 10진수 부동 소수점의 경우, 특수값은 다음과 같이 처리됩니다.

- FLOOR(NaN)은 NaN을 리턴합니다.
- FLOOR(-NaN)은 -NaN을 리턴합니다.
- FLOOR(Infinity)은 Infinity를 리턴합니다.
- FLOOR(-Infinity)은 -Infinity를 리턴합니다.
- FLOOR(sNaN)은 NaN과 경고를 리턴합니다.
- FLOOR(-sNaN)은 -NaN과 경고를 리턴합니다.

예:

- FLOOR 함수를 사용하여 숫자를 소수점 오른쪽 두 자리로 절단하십시오.

```
SELECT FLOOR(SALARY)
  FROM EMPLOYEE
```

- 양수와 음수 모두에 대해 FLOOR 함수를 사용하십시오.

```
VALUES FLOOR(3.5), FLOOR(3.1),
  FLOOR(-3.1), FLOOR(-3.5)
```

  이 예는 각각 3., 3., -4. 및 -4.를 리턴합니다.

GENERATE_UNIQUE



스키마는 SYSIBM입니다.

GENERATE_UNIQUE 함수는 동일한 함수의 다른 실행과 비교하여 고유한 13바이트 길이의 비트 데이터 문자열을 리턴합니다(CHAR(13) FOR BIT DATA). (시스템 시계는 함수에서 실행되는 데이터베이스 파티션 번호와 함께 내부 세계 표준시(UTC) 시간소인을 생성하는 데 사용됩니다. 실제 시스템 시계를 뒤로 가도록 조정하면 값이 중복될 수 있습니다.) 함수는 비결정 함수로 정의됩니다.

이 함수에 대한 인수가 없습니다. (공백 괄호를 지정해야 합니다.)

함수의 결과는 세계 표준시(UTC) 및 함수가 처리된 데이터베이스 파티션 번호의 내부 양식을 포함하는 고유 값입니다. 결과는 널(NULL)이 될 수 없습니다.

이 함수의 결과를 사용하여 테이블에 고유한 값을 제공할 수 있습니다. 연속되는 각 값은 이전 값보다 크므로 테이블 내에서 사용할 수 있는 시퀀스를 제공합니다. 값에는 여러 데이터베이스 파티션에 걸쳐 파티션된 테이블도 일부 시퀀스에서 고유한 값을 갖도록 함수에서 실행된 데이터베이스 파티션 번호가 포함됩니다. 함수에서 실행된 시간에 따라 시퀀스가 달라집니다.

이 함수는 복수 행 INSERT문이나 fullselect를 갖는 INSERT문에 대해 고유한 값이 생성된다는 점에서 특수 레지스터를 사용하는 것과는 다릅니다.

이 함수 결과의 일부 시간소인 값은 GENERATE_UNIQUE의 결과를 인수로 한 상태에서 TIMESTAMP 스칼라 함수를 사용하여 결정할 수 있습니다.

예를 들면, 다음과 같습니다.

- 각 행에 고유한 컬럼을 포함하는 테이블을 작성하십시오. GENERATE_UNIQUE 함수를 사용하여 이 컬럼에 데이터를 채우십시오. UNIQUE_ID 컬럼에는 컬럼을 비트 데이터 문자열로 식별하기 위해 ″FOR BIT DATA″가 지정되어 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 ID를 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
  SNEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

이 트리거가 정의되면 다음과 같이 첫 번째 컬럼없이 이전의 INSERT문을 발행할 수 있습니다.

```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)
  VALUES ('000020', 'Update entry 1...'),
  ('000050', 'Update entry 2...')
```

다음을 사용하여 EMP_UPDATE에 행이 추가될 때의 시간소인(UTC에서)을 리턴할 수 있습니다.

```
SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT
  FROM EMP_UPDATE
```

그러므로 행이 삽입된 시간을 기록하기 위해 테이블에 시간소인 컬럼을 가질 필요가 없습니다.

GETHINT

스키마는 SYSIBM입니다.

GETHINT 함수는 *encrypted-data*에서 하나가 발견되는 경우 암호 힌트를 리턴합니다. 암호 힌트는 데이터 소유자가 암호를 기억하도록 도와주는 구문입니다. 예를 들어 'Pacific'을 기억하기 위한 힌트로 'Ocean'을 사용합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

encrypted-data
전체 암호화된 데이터 문자열인 CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA 값을 리턴하는 표현식입니다. 데이터 문자열은 ENCRYPT 함수를 사용하여 암호화되어야 합니다(SQLSTATE 428FE).

함수의 결과는 VARCHAR(32)입니다. 결과는 널(NULL)이 될 수 있습니다. 즉, ENCRYPT 함수에서 힌트 매개변수가 *encrypted-data*에 추가되지 않았거나 첫 번째 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

예:

이 예에서 힌트 'Ocean'이 저장되어 사용자가 암호화 암호 'Pacific'을 기억하는 데 도움이 됩니다.

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Pacific','Ocean');
SELECT GETHINT(SSN)  FROM EMP;
```

리턴되는 값은 'Ocean'입니다.

GRAPHIC

정수를 그래픽으로:

10진수를 그래픽으로:



부동 소수점을 그래픽으로:



10진수 부동 소수점을 그래픽으로:



문자를 그래픽으로:



그래픽을 그래픽으로:



날짜 시간을 그래픽으로:



스키마는 SYSIBM입니다. 함수 시그니처에서 키워드를 사용할 때 함수 이름을 규정된 이름으로 지정할 수 없습니다.

GRAPHIC 함수는 다음과 같은 고정 길이 그래픽 문자열 표현을 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우, 정수(유니코드 데이터베이스 전용)

- 첫 번째 인수가 10진수인 경우, 10진수(유니코드 데이터베이스 전용)
- 첫 번째 인수가 DOUBLE 또는 REAL인 경우, 배정밀도 부동 소수점 숫자(유니코드 데이터베이스 전용)
- 인수가 10진수 부동 소수점 숫자(DECFLOAT)인 경우, 10진수 부동 소수점 숫자(유니코드 데이터베이스 전용)
- 첫 번째 인수가 임의의 문자열 유형인 경우 1바이트 문자를 2바이트 문자로 변환한 문자열
- 첫 번째 인수가 임의의 그래픽 문자열 유형인 경우 그래픽 문자열
- 첫 번째 인수가 날짜, 시간 또는 시간소인인 경우 날짜 시간 값(유니코드 데이터베이스 전용)

유니코드 데이터베이스의 경우 제공된 인수가 문자열이면 함수를 실행하기 전에 먼저 그래픽 문자열로 변환됩니다. 출력 문자열이 절단되면 마지막 문자는 바뀔 가능성이 높으며 이는 공백 문자(X'0020')로 변환됩니다. 추후 릴리스에서 변경될 수 있으므로 이 동작에 의존하지 마십시오.

함수의 결과는 고정 길이 그래픽 문자열입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

정수를 그래픽으로

integer-expression
데이터 유형이 정수(SMALLINT, INTEGER 또는 BIGINT)인 값을 리턴하는 표현식입니다.

결과는 SQL 정수 상수의 양식으로 된 *integer-expression*의 고정 길이의 그래픽 문자열 표현입니다. 결과는 인수의 유효 숫자를 표시하는 *n* 2바이트 문자로 구성되며, 인수가 음수일 경우에는 빼기 부호가 앞에 나타납니다. 결과는 왼쪽으로 정렬됩니다.

- 첫 번째 인수가 '작은 정수(small integer)'인 경우 결과 길이는 6입니다.
- 첫 번째 인수가 '큰 정수(large integer)'인 경우 결과 길이는 11입니다.
- 첫 번째 인수가 '대정수(big integer)'인 경우 결과 길이는 20입니다.

결과의 2바이트 문자 수가 정의된 결과 길이보다 작으면, 결과는 결과의 오른쪽에 공백이 채워집니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

10진수를 그래픽으로

decimal-expression
10진수 데이터 유형인 값을 리턴하는 표현식입니다. DECIMAL 스칼라 함수는 정밀도와 스케일을 변경하기 위해 사용될 수 있습니다.

decimal-character

결과 그래픽 문자열에서 10진 숫자를 분리하는 데 사용되는 2바이트 문자 상수를 지정합니다. 2바이트 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 상수의 양식으로 된 *decimal-expression*의 고정 길이의 그래픽 문자열 표현입니다. 결과 길이는 2 + *p*입니다. 여기서, *p*는 *decimal-expression*의 정밀도입니다. 앞에 0이 포함되지 않습니다. 뒤의 영(0)이 포함됩니다. *decimal-expression*이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호이고, 그렇지 않으면 첫 번째 2바이트 문자가 숫자이거나 10진수 문자입니다. *decimal-expression*의 스케일이 0이면, 10진수 문자가 리턴되지 않습니다. 결과의 2바이트 문자 수가 정의된 결과 길이보다 작으면, 결과는 결과의 오른쪽에 공백이 채워집니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

부동 소수점을 그래픽으로

floating-point-expression

부동 소수점 데이터 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식입니다.

decimal-character

결과 그래픽 문자열에서 10진수 숫자를 분리하는 데 사용되는 2바이트 문자 상수를 지정합니다. 2바이트 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *floating-point-expression*의 고정 길이의 그래픽 문자열 표현입니다. 결과의 길이는 24입니다. 가수(mantissa)가 마침표와 일련의 숫자가 뒤에 나오는 0 이외의 단일 숫자로 구성될 수 있도록 *floating-point-expression*의 값을 표시할 수 있는 2바이트 문자의 최소 문자 수가 결과가 됩니다. *floating-point-expression*이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호이고, 그렇지 않으면 첫 번째 2바이트 문자가 숫자입니다. *floating-point-expression*이 0이면, 결과는 0E0입니다. 결과의 문자 수가 24자보다 작으면 결과의 오른쪽에 공백이 채워집니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

10진수 부동 소수점을 그래픽으로

decimal-floating-point-expression

부동 소수점 데이터 유형(DOUBLE)인 값을 리턴하는 표현식입니다.

decimal-character

결과 그래픽 문자열에서 10진수 숫자를 분리하는 데 사용되는 2바이트 문자 상수를 지정합니다. 2바이트 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *decimal-floating-point-expression*의 고정 길이 그래픽 문자열 표현입니다. 결과의 길이 속성은 42입니다. 결과는 *decimal-floating-point-expression*의 값을 표시할 수 있는 2바이트 문자의 최소 문자 수입니다. *decimal-floating-point-expression*이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호이고, 그렇지 않으면 첫 번째 2바이트 문자가 숫자입니다. *decimal-floating-point-expression*이 0이면 결과는 0입니다.

*decimal-floating-point-expression*의 값이 특수 값 무한대, sNaN 또는 NaN이면, 문자열 G'INFINITY', G'SNAN'및 G'NAN'이 각각 리턴됩니다. 특수 값이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호입니다. 문자열로 변환될 때 10진수 부동 소수점 특수 값 sNaN은 경고를 발생시키지 않습니다. 결과의 문자 수가 42자보다 작으면 결과의 오른쪽에 공백이 채워집니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

문자를 그래픽으로

character-expression

내장 문자열 데이터 유형(CHAR, VARCHAR 또는 CLOB)인 값을 리턴하는 표현식입니다.

integer

결과로 발생하는 고정 길이 그래픽 문자열의 길이 속성입니다. 값의 범위는 0 - 127이어야 합니다. 두 번째 인수가 지정되지 않은 경우:

- *character-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다. 첫 번째 인수의 실제 길이(뒤 공백 포함)가 127보다 크면, 오류가 리턴됩니다(SQLSTATE 22001).

결과의 실제 길이는 결과의 길이 속성과 같습니다. *character-expression*의 길이가 결과의 길이보다 작으면, 결과에는 결과의 길이까지 공백이 채워집니다. *character-expression*의 길이가 결과의 길이 속성보다 크면, 경고가 리턴되지 않고 결과가 절단됩니다.

그래픽을 그래픽으로

graphic-expression

그래픽 문자열 데이터 유형(GRAPHIC, VARGRAPHIC 또는 DBCLOB)인 내장 값을 리턴하는 표현식입니다.

integer

결과로 발생하는 고정 길이 그래픽 문자열의 길이 속성입니다. 값의 범위는 0 - 127이어야 합니다.

두 번째 인수가 지정되지 않은 경우:

- graphic-expression이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다. 첫 번째 인수의 실제 길이(뒤 공백 제외)가 127보다 크면, 오류가 리턴됩니다(SQLSTATE 22001).

결과의 실제 길이는 결과의 길이 속성과 같습니다. *graphic-expression*의 길이가 결과의 길이보다 작으면, 결과에는 결과의 길이까지 공백이 채워집니다. *graphic-expression*의 길이가 결과의 길이 속성보다 크면, 결과가 절단됩니다. 절단된 문자가 모두 공백이고 *graphic-expression*이 DBCLOB가 아닌 경우를 제외하고는 경고가 리턴됩니다(SQLSTATE 01004).

날짜 시간을 그래픽으로

datetime-expression

다음 데이터 유형 중 하나인 표현식입니다.

DATE

결과는 두 번째 인수가 지정하는 형식으로 된 날짜의 그래픽 문자열 표현입니다. 결과 길이는 10입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않은 경우 오류가 발생합니다(SQLSTATE 42703).

TIME 결과는 두 번째 인수가 지정하는 형식으로 된 시간의 그래픽 문자열 표현입니다. 결과의 길이는 8입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않으면 오류가 발생합니다(SQLSTATE 42703).

TIMESTAMP

결과는 시간소인의 그래픽 문자열 표현입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(0)이면 결과의 길이는 19입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(*n*)이고 *n*이 1과 12 사이인 경우, 결과의 길이는 20+*n*입니다. 그렇지 않으면 결과의 길이는 26입니다.

문자열의 코드 페이지는 섹션의 코드 페이지입니다.

주: 첫 번째 인수가 숫자이거나 첫 번째 인수가 문자열이고 길이 인수가 지정되어 있을 때 CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예:

- EDLEVEL 컬럼은 SMALLINT로 정의되어 있습니다. 다음은 값을 고정 길이 그래픽 문자열로 리턴합니다.

```
SELECT GRAPHIC(EDLEVEL)
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  결과 값은 G'18입니다.

- SALARY 컬럼 및 COMN 컬럼은 스케일 2 및 정밀도 9인 DECIMAL로 정의되어 있습니다. 쉼표 10진수 문자를 사용하여 직원 Haas의 총 수입을 리턴합니다.

```
SELECT GRAPHIC(SALARY + COMM, ',')
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  결과 값은 G'56970,00 '입니다.

GREATEST



스키마는 SYSIBM입니다.

GREATEST 함수는 값 세트에서 최대값을 리턴합니다.

인수는 호환 가능해야 하며 각 인수는 ARRAY, LOB, LONG VARCHAR, LONG VARGRAPHIC, XML, 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 아닌 다른 데이터 유형의 값을 리턴하는 표현식이어야 합니다(SQLSTATE 42815). 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 호환 가능한 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 서명을 작성할 필요가 없습니다.

필요한 경우 선택된 인수는 결과의 속성으로 변환됩니다. 결과의 속성은 결과 데이터 유형의 규칙을 기초로 모든 피연산자에 의해 판별됩니다.

함수 결과는 가장 큰 인수 값입니다. 하나 이상의 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 어느 한 인수가 널(NULL)일 경우 결과는 널(NULL) 값이 됩니다.

GREATEST 스칼라 함수는 MAX 스칼라 함수의 동의어입니다.

예를 들면, 다음과 같습니다.

테이블 T1에 값이 각각 1, 7, 4인 세 개의 컬럼 C1, C2, C3이 있다고 가정합니다. 다음 쿼리는

```
SELECT GREATEST (C1, C2, C3) FROM T1
```

7을 리턴합니다.

컬럼 C3에 4 대신 NULL 값이 있는 경우 동일한 쿼리는 NULL을 리턴합니다.

HASHEDVALUE

스키마는 SYSIBM입니다.

HASHEDVALUE 함수는 행의 분산 키 값에 파티션 함수를 적용하여 얻어진 행의 분산 맵 인덱스를 리턴합니다. 예를 들어, SELECT절에 사용하는 경우 SELECT문의 결과를 작성하는 데 사용된 테이블의 각 행에 대한 분산 맵 인덱스를 리턴합니다.

전이 변수 및 테이블에서 리턴되는 분산 맵 인덱스는 분산 키 컬럼의 현재 전이 값에서 파생됩니다. 예를 들어, 사전 삽입 트리거에서 새 전이 변수의 현재 값을 제공하면 이 함수는 프로젝트된 분산 맵 인덱스를 리턴합니다. 그러나 분산 키 컬럼 값은 이후의 사전 삽입 트리거로 수정될 수 있습니다. 따라서 데이터베이스에 삽입될 때 행의 최종 분산 맵 인덱스는 프로젝트된 값과 다를 수 있습니다.

인수는 테이블의 컬럼에 대해 규정된 이름이거나 규정되지 않은 이름이어야 합니다. 컬럼의 데이터 유형에는 제한이 없습니다. 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 또한 이 함수는 인수로 모든 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다. *column-name*이 뷰의 컬럼을 참조하는 경우 해당 컬럼에 대한 뷰의 표현식은 하위 기본 테이블의 컬럼을 참조해야 하고 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다.

분산 맵 인덱스가 HASHEDVALUE 함수에서 리턴되는 특정 행(및 테이블)은 함수를 사용하는 SQL문의 컨텍스트로 결정됩니다.

결과의 데이터 유형은 0에서 32767 까지 범위의 INTEGER입니다. 분산 키가 없는 테이블의 경우 결과는 항상 0입니다. 널(NULL) 값은 리턴되지 않습니다. 행 레벨 정보가 리턴되므로 테이블에 지정되는 컬럼에 관계없이 결과는 동일합니다.

HASHEDVALUE 함수는 점검 제한조건 내의 복제된 테이블이나 생성된 컬럼의 정의에서 사용할 수 없습니다(SQLSTATE 42881).

버전 8 이전 제품과의 호환성을 위해 함수 이름 PARTITION은 HASHEDVALUE를 대체할 수 있습니다.

예:

- 분산 맵 인덱스가 100인 모든 행에 대해 EMPLOYEE 테이블의 직원 수(EMPNO)를 나열하십시오.

```
SELECT EMPNO FROM EMPLOYEE
  WHERE HASHEDVALUE(PHONENO) = 100
```

- EMPLOYEE 테이블에서 BEFORE 트리거를 작성하여 직원을 삽입할 경우 EMPINSERTLOG2 테이블의 새 행에 프로젝트된 분산 맵 인덱스 및 직원 번호를 기록하십시오.

```
CREATE TRIGGER EMPINSLOGTRIG2
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AW NEWTABLE
  FOR EACH ROW
  INSERT INTO EMPINSERTLOG2
    VALUES(NEWTABLE.EMPNO, HASHEDVALUE(NEWTABLE.EMPNO))
```

HEX

스키마는 SYSIBM입니다.

HEX 함수는 문자열 값의 16진 표현을 리턴합니다.

인수는 최대 길이가 16 336바이트인 내장 데이터 유형 값의 표현식이 될 수 있습니다.

함수의 결과는 문자열입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

코드 페이지는 섹션 코드 페이지입니다.

결과는 16진수의 문자열입니다. 처음 두 숫자는 인수의 첫 바이트를 나타내고 다음 두 숫자는 인수의 두 번째 바이트를 나타냅니다. 인수가 날짜 시간 값이거나 숫자 값인 경우 결과는 인수의 내부 양식에 대한 16진 표현입니다. 리턴되는 16진 표현은 함수가 실행되는 응용프로그램 서버(AS)에 따라 다를 수 있습니다. 차이점이 명백한 경우는 다음과 같습니다.

- HEX 함수는 EBCDIC 서버가 있는 ASCII 클라이언트나 ASCII 서버가 있는 EBCDIC 클라이언트에서 수행될 때의 문자열 인수입니다.
- HEX 함수가 수행될 때의 숫자 인수(일부 경우에서)입니다. 여기서, 클라이언트 및 서버 시스템에는 숫자 값에 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 길이 문자열입니다.
  - 127보다 긴 고정 길이
    - 결과는 인수의 정의된 길이의 두 배인 가변 길이 문자열입니다.
  - 가변 길이
    - 결과는 인수의 정의된 최대 길이의 두 배인 가변 길이의 문자열입니다.
- 그래픽 문자열
  - 63보다 길지 않은 고정 길이
    - 결과는 인수의 정의된 길이의 네 배인 고정 길이 문자열입니다.
- 63보다 긴 고정 길이
  - 결과는 인수의 정의된 길이의 네 배인 가변 길이 문자열입니다.
- 가변 길이

– 결과는 인수의 정의된 최대 길이의 네 배인 가변 길이의 문자열입니다.

예를 들면, 다음과 같습니다.

다음 예에서 AIX 응용프로그램 서버에 대해 DB2를 사용한다고 가정하십시오.

- DEPARTMENT 테이블을 사용하여 호스트 변수 HEX_MGRNO(char(12))를 'PLANNING' 부서(DEPTNAME)에 대한 관리자 번호(MGRNO)의 16진수 표시로 설정하십시오.

  ```
  SELECT HEX(MGRNO)
    INTO :HEX_MGRNO
    FROM DEPARTMENT
    WHERE DEPTNAME = 'PLANNING'
  ```

  샘플 테이블을 사용할 때 HEX_MGRNO가 '303030303230'으로 설정됩니다(문자 값은 '000020').
- COL_1을 데이터 유형이 char(1)이고 값이 'B'인 컬럼으로 가정해 보십시오. 문자 'B'의 16진 표현은 X'42'입니다. HEX(COL_1) 함수에서는 2바이트의 Long 문자열 '42'가 리턴됩니다.
- COL_3을 데이터 유형이 decimal(6,2)이고 값이 40.1인 컬럼으로 가정해 보십시오. 8바이트의 Long 문자열 '0004010C'는 HEX 함수를 10진수 값(40.1)의 내부 표현에 적용한 결과입니다.

HOUR

스키마는 SYSIBM입니다.

HOUR 함수는 값의 시간 부분을 리턴합니다.

내장 데이터 유형인 날짜, 시간, 시간소인, 문자열 또는 정확한 숫자 데이터 유형 중 하나의 값을 리턴하는 표현식입니다.

- *expression*이 문자열인 경우, CLOB이어서는 안되며 값은 날짜 시간 값의 유효한 문자열 표현이어야 합니다. 날짜 시간 값의 문자열 표현의 유효한 포맷에 대해서는, 『날짜 시간 값』의 『날짜 시간 값의 문자열 표현』을 참조하십시오.
- *expression*이 정확한 숫자 값인 경우, 이는 시간 지속 기간 또는 시간소인 지속 기간이어야 합니다. 유효한 시간 지속 기간 및 시간소인 지속 기간에 대한 정보는 『날짜 및 시간 피연산자 및 지속기간』을 참조하십시오.
- 유니코드 데이터베이스만 DBCLOB가 아닌 날짜 시간 값의 유효한 그래픽 문자열 표현인 표현식을 지원합니다. 그래픽 문자열은 함수를 실행하기 전에 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 시간, 시간소인 또는 시간이나 시간소인의 유효한 문자열 표현인 경우
  - 결과는 값의 시간 부분으로 0과 24 사이의 정수입니다.
- 인수가 날짜 또는 날짜의 유효한 문자열 표현인 경우
  - 결과는 0입니다.
- 인수가 시간 지속 기간이거나 시간소인 지속 기간인 경우
  - 결과는 값의 시간 부분으로 1과 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
```

IDENTITY_VAL_LOCAL



스키마는 SYSIBM입니다.

IDENTITY_VAL_LOCAL 함수는 VALUES절을 사용하는 단일 INSERT문의 결과로 지정된 ID 컬럼에 대해 가장 최근에 지정된 값을 리턴하는 비결정 함수입니다. 함수에 입력 매개변수가 없습니다.

결과는 해당하는 식별 컬럼의 실제 데이터 유형과는 상관없이 DECIMAL(31,0)입니다.

함수가 리턴한 값은 가장 최근의 단일 행 삽입 조작에서 식별된 테이블 컬럼을 식별하기 위해 지정된 값입니다. INSERT문에는 식별 컬럼이 들어 있는 테이블에 대해 VALUES절이 포함되어야 합니다. 또한 INSERT문은 동일한 레벨에서 발행되어야 합니다. 즉, 값은 다음 지정된 값으로 대체될 때까지 지정된 레벨에서 로컬로 사용 가능해야 합니다. 새 레벨은 트리거나 루틴이 호출될 때마다 시작됩니다.

지정된 값은 사용자가 제공하는 값(식별 컬럼이 GENERATED BY DEFAULT로 정의되는 경우)이나 데이터베이스 관리 프로그램에서 생성되는 식별 값이어야 합니다.

ID 컬럼에 대해 지정된 값을 가져올 때 SELECT FROM data-change-table-reference문을 사용하는 것이 좋습니다. 자세한 정보는 ″subselect″의 ″table-reference″를 참조하십시오.

VALUES 절을 갖는 단일 행 INSERT문이 식별 컬럼이 들어 있는 테이블에 대해 현재 처리 레벨에서 발행되지 않을 경우 함수는 널(NULL) 값을 리턴합니다.

함수의 결과는 다음의 영향을 받지 않습니다.

- 식별 컬럼이 없는 테이블에 대해 VALUES절을 갖는 단일 행 INSERT문
- VALUES절을 갖는 복수 행 INSERT문
- fullselect를 갖는 INSERT문
- ROLLBACK TO SAVEPOINT문

주

- INSERT문의 VALUES절에 있는 표현식은 삽입 조작의 목표 컬럼 지정 이전에 평가됩니다. 따라서 INSERT문의 VALUES절 내에서 IDENTITY_VAL_LOCAL 함수 호출은 이전 삽입 조작의 ID 컬럼에 가장 최근에 지정된 값을 사용합니다. 식별 컬럼이 들어 있는 테이블에 대해 VALUES 절을 갖는 이전 단일 행 INSERT문이 IDENTITY_VAL_LOCAL 함수와 동일한 레벨 내에서 실행된 적이 없는 경우 함수는 널(NULL) 값을 리턴합니다.

- 트리거가 정의된 테이블의 식별 컬럼 값은 식별 컬럼에 대한 트리거 전이 변수를 참조하여 트리거 내에서 판별할 수 있습니다.
- 삽입 트리거의 트리거 조건 내에서 IDENTITY_VAL_LOCAL 함수를 호출하면 그 결과는 널(NULL)이 됩니다.
- 한 테이블에 대해 복수의 BEFORE 또는 AFTER INSERT 트리거가 존재할 수 있습니다. 이 경우 각 트리거는 개별적으로 처리되며 하나의 트리거 조치에서 지정된 식별 값은 IDENTITY_VAL_LOCAL 함수를 사용하는 다른 트리거 조치에 사용할 수 없습니다. 이것은 복수 트리거 조치가 개념적으로 동일한 레벨에서 정의되는 경우에도 해당됩니다.
- 일반적으로 사전 삽입 트리거의 본문에 IDENTITY_VAL_LOCAL 함수를 사용하는 것은 권장되지 않습니다. 사전 삽입 트리거의 트리거 조치 내에서 IDENTITY_VAL_LOCAL 함수를 호출하면 그 결과는 널(NULL)이 됩니다. 트리거가 정의된 테이블의 식별 컬럼 값은 사전 삽입 트리거의 트리거 조치 내에서 IDENTITY_VAL_LOCAL 함수를 호출하여 확보할 수 없습니다. 그러나 식별 컬럼 값은 식별 컬럼에 대한 트리거 전이 변수를 참조하여 트리거 조치에서 확보할 수 있습니다.
- AFTER INSERT 트리거의 트리거 조치 내에서 IDENTITY_VAL_LOCAL 함수 호출 결과는 ID 컬럼을 포함하는 테이블에 대해 VALUES절을 갖는 동일한 트리거 조치에서 호출된 가장 최근의 단일 행 삽입 조작에서 식별된 테이블의 ID 컬럼에 지정된 값입니다. 이는 FOR EACH ROW 및 FOR EACH STATEMENT 사후 삽입 트리거에 모두 적용됩니다. 식별 컬럼이 들어 있는 테이블에 대해 VALUES절을 갖는 단일 행 INSERT문이 IDENTITY_VAL_LOCAL 함수 호출 전에 동일한 트리거 조치 내에서 실행되지 않은 경우 함수는 널(NULL) 값을 리턴합니다.
- IDENTITY_VAL_LOCAL은 비결정적 함수이므로 커서의 SELECT문 내에서 이 함수의 호출 결과는 각 FETCH문에 대해 다를 수 있습니다.
- 지정된 값은 식별 컬럼에 실제로 지정된 값(즉, 후속 SELECT문에서 리턴되는 값)입니다. 이 값이 반드시 INSERT문의 VALUES절에 제공되는 값이나 데이터베이스 관리 프로그램이 생성하는 값은 아닙니다. 지정된 값은 식별 컬럼과 연관된 트리거 전이 변수에 대해 사전 삽입 트리거의 본문 내에서 SET 전이 변수 명령문에 지정된 값일 수 있습니다.
- ***IDENTITY_VAL_LOCAL 범위***: IDENTITY_VAL_LOCAL 값은 현재 세션에서 정의된 식별 컬럼을 포함하는 테이블로의 다음 삽입 또는 응용프로그램 세션 종료 시까지 유지됩니다. 값은 COMMIT 또는 ROLLBACK문에 영향을 받지 않습니다. IDENTITY_VAL_LOCAL 값은 직접 설정할 수 없으며 테이블에 행을 삽입한 결과로 작성됩니다.

  흔히 사용되는 기술, 특히 성능은 연결 설정을 관리하고 임의 연결에 대한 트랜잭션을 라우트할 응용프로그램 또는 제품을 위한 것입니다. 이런 상황에서 IDENTITY_VAL_LOCAL 값의 가용성은 트랜잭션 종료시까지만 신뢰할 수 있어

야 합니다. 이런 상황이 발생할 수 있는 예에는 XA 프로토콜 사용, 연결 풀 사용, 연결 집중기(connection concentrator) 사용 및 장애 복구를 위해 HADR을 사용하는 응용프로그램이 포함됩니다.

- 식별 컬럼을 갖는 테이블로 VALUES절을 갖는 실패한 단일 행 INSERT문 다음에 오는 함수가 리턴하는 값은 예측 불가능합니다. 실패한 삽입 조작 이전에 호출된 함수로부터 리턴된 값이거나 삽입 조작이 성공한 경우 할당된 값일 수 있습니다. 리턴되는 실제 값은 실패 지점에 따라 다르므로 예측이 불가능합니다.

예를 들면, 다음과 같습니다.

예 1: C1이라는 식별 컬럼을 가진 T1 및 T2의 2개 테이블을 각각 작성하십시오. C1. 테이블 2의 식별 시퀀스를 10에서 시작하십시오. C2에 대한 일부 값을 T1에 삽입하십시오.

```
CREATE TABLE T1
       (C1 INTEGER GENERATED ALWAYS AS IDENTITY,
        C2 INTEGER)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY (START WITH 10),
        C2 INTEGER)

INSERT INTO T1 (C2) VALUES (5)

INSERT INTO T1 (C2) VALUES (6)

SELECT * FROM T1
```

이 쿼리는 다음을 리턴합니다.

```
C1          C2
----------- -----------
                  1           5
                  2           6
```

컬럼 C2가 IDENTITY_VAL_LOCAL 함수에서 값을 얻는 테이블 T2에 단일 행을 삽입하십시오.

```
INSERT INTO T2 (C2) VALUES (IDENTITY_VAL_LOCAL())

SELECT * FROM T2
```

이 쿼리는 다음을 리턴합니다.

```
C1                  C2
------------------- -----------
             10. 2
```

예 2: 트리거를 포함하는 중첩된 환경에서 IDENTITY_VAL_LOCAL 함수를 사용하여 더 낮은 레벨에서 지정된 식별 값이 있었던 경우에도 특정 레벨에서 지정된 식별 값을 검색하십시오. EMPLOYEE, EMP_ACT 및 ACCT_LOG의 세 테이블이 있다고 가

정합시다. EMP_ACT 및 ACCT_LOG 테이블에 추가 삽입을 발생시키는 EMPLOYEE에 정의된 AFTER INSERT 트리거가 있습니다.

```
CREATE TABLE EMPLOYEE
       (EMPNO SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1000),
         NAME CHAR(30),
         SALARY DECIMAL(5,2),
         DEPTNO SMALLINT)

CREATE TABLE EMP_ACT
       (ACNT_NUM SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1),
         EMPNO SMALLINT)

CREATE TABLE ACCT_LOG
       (ID SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 100),
         ACNT_NUM SMALLINT,
         EMPNO SMALLINT)

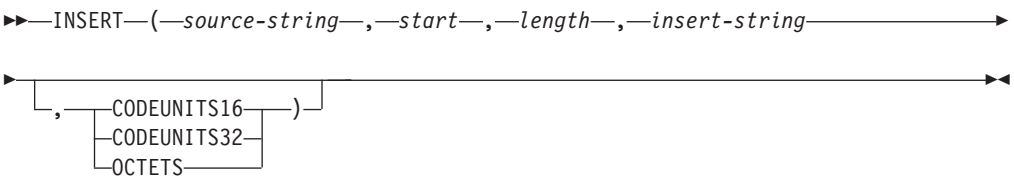
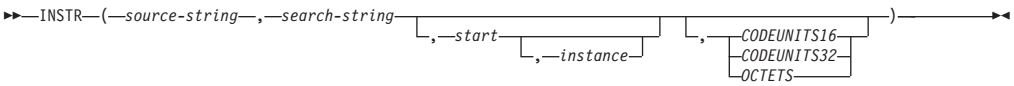
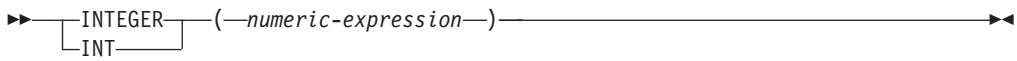
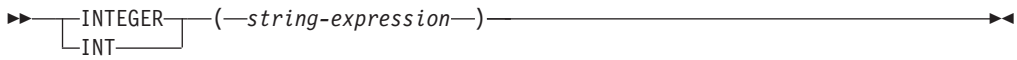
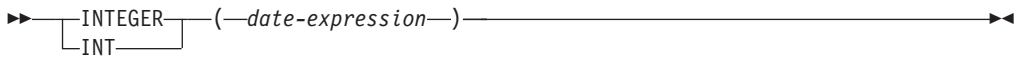
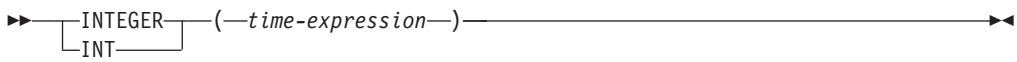
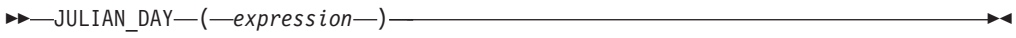
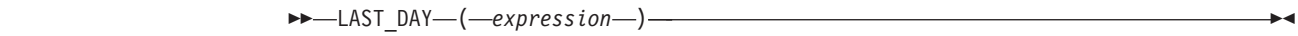
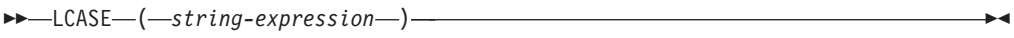
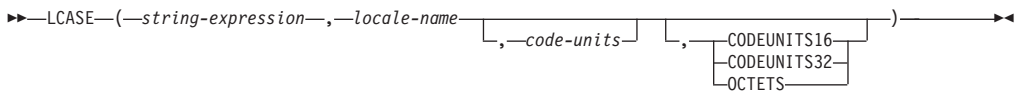
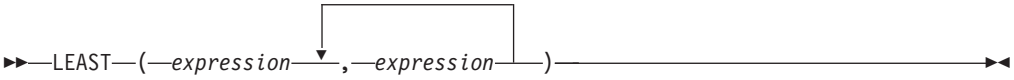
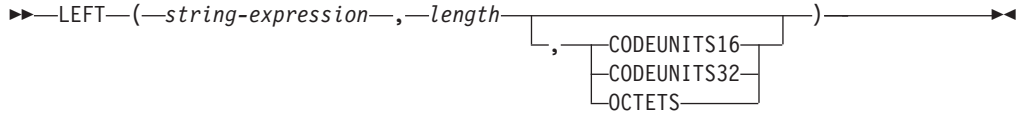
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  REFERENCING NEW AS NEW_EMP
  FOR EACH ROW
  BEGIN ATOMIC
    INSERT INTO EMP_ACT (EMPNO) VALUES (NEW_EMP.EMPNO);
    INSERT INTO ACCT_LOG (ACNT_NUM, EMPNO)
      VALUES (IDENTITY_VAL_LOCAL(), NEW_EMP.EMPNO);
  END
```

첫 번째 트리거 삽입 조작은 EMP_ACT 테이블에 행을 삽입합니다. 이 명령문은 EMPLOYEE 테이블의 EMPNO 컬럼에 트리거 전이 변수를 사용하여 EMPLOYEE 테이블의 EMPNO 컬럼에 대한 ID 값이 EMP_ACT 테이블의 EMPNO 컬럼으로 복사됨을 나타냅니다. INSERT문이 이 중첩 레벨에서 발행되지 않으므로 IDENTITY_VAL_LOCAL 함수를 사용하여 EMPLOYEE 테이블의 EMPNO 컬럼에 지정된 값을 확보하지 못할 수 있습니다. IDENTITY_VAL_LOCAL 함수가 EMP_ACT 테이블에 대한 INSERT문의 VALUES 절에서 호출된 경우 널(NULL) 값을 리턴합니다. EMP_ACT 테이블에 대한 삽입 조작도 ACNT_NUM 컬럼의 새 ID 값 생성을 초래합니다.

두 번째 트리거 삽입 조작은 ACCT_LOG 테이블에 행을 삽입합니다. 이 명령문은 트리거 조치의 이전 삽입 조작에서 EMP_ACT 테이블의 ACNT_NUM 컬럼에 지정된 ID 값이 ACCT_LOG 테이블의 ACNT_NUM 컬럼으로 복사됨을 나타내기 위해 IDENTITY_VAL_LOCAL 함수를 호출합니다. EMPNO 컬럼에 EMPLOYEE 테이블의 EMPNO 컬럼과 동일한 값이 지정됩니다.

다음의 INSERT문 및 모든 트리거 조치가 진행된 후에

```
INSERT INTO EMPLOYEE (NAME, SALARY, DEPTNO)
  VALUES ('Rupert', 989.99, 50)
```

3개 테이블의 내용이 다음과 같이 나타납니다.

```
   SELECT EMPNO, SUBSTR(NAME,1,10) AS NAME, SALARY, DEPTNO
     FROM EMPLOYEE

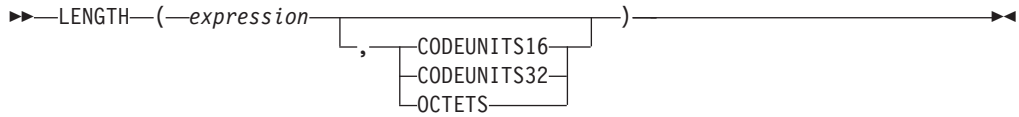
EMPNO  NAME       SALARY  DEPTNO
------ ---------- ------- ------
    1000 Rupert      989.99     50

   SELECT ACNT_NUM, EMPNO
     FROM EMP_ACT

ACNT_NUM EMPNO
-------- ------
               1   1000

   SELECT * FROM ACCT_LOG

ID     ACNT_NUM EMPNO
------ -------- ------
      100        1   1000
```

IDENTITY_VAL_LOCAL 함수의 결과는 동일한 중첩 레벨에서 식별 컬럼에 대해 가장 최근에 지정된 값입니다. 원래의 INSERT문과 모든 트리거 조치를 처리한 후 IDENTITY_VAL_LOCAL 함수가 값 1000을 리턴하는데, 이는 이 값이 EMPLOYEE 테이블의 EMPNO 컬럼에 지정된 값이기 때문입니다.

INITCAP

스키마는 SYSIBM입니다.

INITCAP 함수는 각 *단어*의 첫 번째 문자는 UPPER 함수 시맨틱을 사용하여 대문자로 변환되고 다른 문자는 LOWER 함수 시맨틱을 사용하여 소문자로 변환된 문자열을 리턴합니다. *단어*는 다음 문자 중 하나로 구분됩니다.

표 45. 단어 분리문자

문자 또는 문자 범위	유니코드 코드 포인트 또는 유니코드 코드 포인트 범위
(공백)	U+0020
! ″ # $ % & ’ ( ) * + , − . /	U+0021 - U+002F
: ; < = > ? @	U+003A - U+0040
[₩] ^ _ `	U+005B - U+0060
{ | } ~	U+007B - U+007E
다음 SQL 제어 문자를 포함한 제어 문자: • 탭 • 줄 바꾸기 • 용지 넘김 • 캐리지 리턴 • 라인 피드	U+0009, U+000A, U+000B, U+000C, U+000D, U+0085

주: 위의 테이블에 나열된 문자는 특정 데이터베이스 코드 페이지에서 할당된 코드 포인트를 수반하지 못할 수도 있습니다.

string-expression

CHAR 또는 VARCHAR 데이터 유형인 값을 리턴하는 표현식입니다. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다.

결과의 데이터 유형은 다음 테이블에 설명된 것처럼 *string-expression*의 데이터 유형에 따라 결정됩니다.

표 46. 결과 데이터 유형에 비교되는 string-expression의 데이터 유형

*string-expression*의 데이터 유형	결과의 데이터 유형
CHAR 또는 VARCHAR	VARCHAR
GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC

결과의 길이 속성은 *string-expression*의 길이 속성과 동일합니다.

인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

- 문자열『a prospective book title』을 입력하여『A Prospective Book Title』을 리턴하십시오.

```
VALUES INITCAP ('a prospective book title')
1
------------------------
A Prospective Book Title
```

- 문자열『YOUR NAME』을 입력하여 문자열『Your Name』을 리턴하십시오.

```
VALUES INITCAP ('YOUR NAME')
1
---------
Your Name
```

- 문자열『my_résumé』를 입력하여 문자열『My_Résumé』를 리턴하십시오.

```
VALUES INITCAP ('my_résumé')
1
---------
My_Résumé
```

- 문자열『élégant』를 입력하여 문자열『Élégant』를 리턴하십시오.

```
VALUES INITCAP ('FORMAT:élégant')
1
--------------
Format:Élégant
```

INSERT



스키마는 SYSIBM입니다. INSERT 함수의 SYSFUN 버전은 계속 사용 가능합니다.

INSERT 함수는 *source-string*에서 시작되는, *length* 바이트가 삭제되고 *insert-string*이 삽입된 문자열을 리턴합니다.

INSERT 함수는 길이 인수가 필수인 것을 제외하고는 OVERLAY 함수와 동일합니다.

source-string

소스 문자열을 지정하는 표현식입니다. 표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아니면, 함수를 평가하기 전에 명시적으로 VARCHAR로 캐스트됩니다.

start

정수 값을 리턴하는 표현식입니다. 정수 값은 바이트 삭제 및 다른 문자열의 삽입이 시작되는 소스 문자열 내의 시작 포인트를 지정합니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 명시적으로 INTEGER로 캐스트됩니다.정수 값은 1과 *source-string* 더하기 1의 길이 사이여야 합니다(SQLSTATE 42815). OCTETS가 지정되어 있고 결과가 그래픽 데이터이면, 값은 1과 *source-string* 더하기 1의 두 배 길이 속성 사이의 홀수여야 합니다(SQLSTATE 428GC).

length

소스 문자열에서 삭제될 코드 단위(지정된 문자열 단위에서)의 수를 지정하는 표현식으로, *start*에서 식별되는 위치에서 시작됩니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 명시적으로 INTEGER로 캐스트됩니다.정수 값은 0 - *source-string*의 길이 범위에 있어야 하며, 내재적으로나 명시적으로 지정된 단위로 표현됩니다(SQLSTATE 22011). OCTETS가 지정되어 있고 결과가 그래픽 데이터이면, 값은 0과 *source-string*의 속성 길이의 두 배 사이여야 합니다(SQLSTATE 428GC).

insert-string

*source-string*에서 삭제될 문자열을 지정하는 표현식으로, *start*에서 식별되는 위치

에서 시작됩니다. 표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아니면, 함수를 평가하기 전에 명시적으로 VARCHAR로 캐스트됩니다.

CODEUNITS16, CODEUNITS32 또는 OCTETS

start 및 *length*의 문자열 단위를 지정합니다.

CODEUNITS16은 *start* 및 *length*가 16비트 UTF-16 코드 단위로 표시되도록 지정합니다. CODEUNITS32는 *start* 및 *length*가 32비트 UTF-32 코드 단위로 표시되도록 지정합니다. OCTETS는 *start* 및 *length*가 바이트 단위로 표시되도록 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 결과가 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *insert-string* 및 *source-string*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815). 문자열 단위가 OCTETS로 지정되면 해당 조작은 *source-string*의 코드 페이지에서 수행됩니다. 문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 결과가 그래픽 데이터이면 *start* 및 *length*는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다. CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는, 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

결과의 데이터 유형은 지원된 유형 조합의 다음 표에서 나타낸 것과 같이 *source-string* 및 *insert-string*의 데이터 유형에 따라 다릅니다.

표 47. source-string 및 insert-string의 데이터 유형의 함수로서의 결과의 데이터 유형

source-string	*insert-string*	결과
CHAR 또는 VARCHAR	CHAR 또는 VARCHAR	VARCHAR
GRAPHIC 또는 VARGRAPHIC	GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC
CLOB	CHAR, VARCHAR 또는 CLOB	CLOB
DBCLOB	GRAPHIC, VARGRAPHIC 또는 DBCLOB	DBCLOB
CHAR 또는 VARCHAR	CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA	CHAR, VARCHAR, CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
BLOB	BLOB	BLOB
유니코드 데이터베이스 전용:		
CHAR 또는 VARCHAR	GRAPHIC 또는 VARGRAPHIC	VARCHAR
GRAPHIC 또는 VARGRAPHIC	CHAR 또는 VARCHAR	VARGRAPHIC

표 47. source-string 및 insert-string의 데이터 유형의 함수로서의 결과의 데이터 유형 (계속)

source-string	*insert-string*	결과
CLOB	GRAPHIC, VARGRAPHIC 또는 DBCLOB	CLOB
DBCLOB	CHAR, VARCHAR 또는 CLOB	DBCLOB

*source-string*의 길이는 0 입니다. 이 경우, *start*는 1이어야 하며 *length*는 0이어야 하며(위에서 설명한 것처럼 *start* 및 *length*의 바운드로 내재된), 함수의 결과는 *insert-string*의 사본입니다.

*insert-string*의 길이는 0일 수 있습니다. 위치 *start*에서 *source-string*로부터의 *start* + *length* - 1까지의 코드 단위가 삭제됩니다.

결과의 길이 속성은 *source-string*과 *insert-string*의 길이 속성을 더한 것입니다. 결과의 실제 길이는 A1 - MIN((A1 - V2 + 1), V3) + A4이며, 여기서:

- A1은 *source-string*의 실제 길이입니다.
- V2는 *start*의 값입니다.
- V3은 *length*의 값입니다.
- A4는 *insert-string*의 실제 길이입니다.

결과 문자열의 실제 길이가 리턴 데이터 유형에 대한 최대값을 초과할 경우, 오류가 리턴됩니다(SQLSTATE 54006).

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- 기존 텍스트 중간에 텍스트를 삽입하여 문자열 'INSERTING'에서 문자열 'INSISTING', 'INSISERTING' 및 'INSTING'를 작성합니다.

```
SELECT INSERT('INSERTING',4,2,'IS'),
  INSERT('INSERTING',4,0,'IS'),
  INSERT('INSERTING',4,2,'')
  FROM SYSIBM.SYSDUMMY1
```

- 시작 포인트로 1을 사용하여 기존 텍스트 앞에 텍스트를 삽입하여, 문자열 'INSERTING'에서 문자열 'XXINSERTING', 'XXNSERTING', 'XXSERTING' 및 'XXERTING'를 작성합니다.

```
SELECT INSERT('INSERTING',1,0,'XX'),
  INSERT('INSERTING',1,1,'XX'),
  INSERT('INSERTING',1,2,'XX'),
  INSERT('INSERTING',1,3,'XX')
  FROM SYSIBM.SYSDUMMY1
```

- 기존 텍스트 다음에 텍스트를 삽입하여 문자열 'ABCABC'에서 문자열 'ABCABCXX'를 작성합니다. 소스 문자열의 길이가 6문자이기 때문에 시작 위치를 7로 설정합니다(소스 문자열의 길이에 1을 더함).

```
SELECT INSERT('ABCABC',7,0,'XX')
  FROM SYSIBM.SYSDUMMY1
```

- 문자열 'Hegelstraße'를 'Hegelstrasse'로 변경합니다.

```
SELECT INSERT('Hegelstraße',10,1,'ss',CODEUNITS16)
  FROM SYSIBM.SYSDUMMY1
```

- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

변수 UTF8_VAR 및 UTF16_VAR에 각각 해당 문자열의 UTF-8 및 UTF-16BE 표시가 들어 있다고 가정하십시오. INSERT 함수를 사용하여 'C'를 유니코드 문자열 '&N~AB'에 삽입합니다.

```
SELECT INSERT(UTF8_VAR, 1, 4, 'C', CODEUNITS16),
  INSERT(UTF8_VAR, 1, 4, 'C', CODEUNITS32),
  INSERT(UTF8_VAR, 1, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 'CAB', 'CB' 및 'CN~AB'를 각각 리턴합니다.

```
SELECT INSERT(UTF8_VAR, 5, 1, 'C', CODEUNITS16),
  INSERT(UTF8_VAR, 5, 1, 'C', CODEUNITS32),
  INSERT(UTF8_VAR, 5, 1, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~CB', '&N~AC' 및 '&C~AB'를 각각 리턴합니다.

```
SELECT INSERT(UTF16_VAR, 1, 4, 'C', CODEUNITS16),
  INSERT(UTF16_VAR, 1, 4, 'C', CODEUNITS32),
  INSERT(UTF16_VAR, 1, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 'CAB', 'CB' 및 'CN~AB'를 각각 리턴합니다.

```
SELECT INSERT(UTF16_VAR, 5, 2, 'C', CODEUNITS16),
  INSERT(UTF16_VAR, 5, 1, 'C', CODEUNITS32),
  INSERT(UTF16_VAR, 5, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~C', '&N~AC' 및 '&CAB'를 각각 리턴합니다.

INSTR



스키마는 SYSIBM입니다.

INSTR 함수는 한 문자열(*source-string*) 내에서 다른 문자열(*search-string*)의 시작 위치를 리턴합니다.

INSTR 스칼라 함수는 LOCATE_IN_STRING 스칼라 함수의 동의어입니다.

INTEGER 또는 INT

숫자를 정수로:



문자열을 정수로:



날짜를 정수로:



시간을 정수로:



스키마는 SYSIBM입니다.

INTEGER 함수는 다음의 정수 표현을 리턴합니다.

- 숫자
- 숫자의 문자열 표현
- 날짜 값
- 시간 값

숫자를 정수로:

numeric-expression

임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

인수가 큰 정수 컬럼이나 변수에 지정된 경우, 결과는 발생한 같은 숫자입니다. 인수의 소수 부분이 절단됩니다. 인수의 전체 부분이 정수 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

문자열을 정수로:

string-expression

문자 상수의 최대 길이보다 길지 않은 길이의 숫자의 문자열 또는 유니코드 그래픽 문자열 표현인 값을 리턴하는 표현식입니다.

결과는 CAST(*string-expresssion* AS INTEGER)에서 발생하는 수와 동일합니다. 앞뒤 공백은 제거되고 결과 문자열은 정수, 10진수, 부동 소수점이나 10진수 부동 소수점 상수에 대한 규칙에 따라야 합니다(SQLSTATE 22018). 인수의 전체 부분이 정수 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003). *string-expression*의 데이터 유형은 CLOB 또는 DBCLOB일 수 없습니다(SQLSTATE 42884).

날짜를 정수로:

date-expression

DATE 데이터 유형 값을 리턴하는 표현식입니다. 결과는 날짜를 *yyyymmdd*로 표시하는 INTEGER 값입니다.

시간을 정수로:

time-expression

TIME 데이터 유형 값을 리턴하는 표현식입니다. 결과는 시간을 *hhmmss*로 표시하는 INTEGER 값입니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주: CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예:

- EMPLOYEE 테이블을 사용하여 교육 수준(EDLEVEL)별로 나눠진 급여(SALARY)를 포함하는 목록을 선택하십시오. 계산 시 소수 부분은 절단됩니다. 목록에는 계산과 직원 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 응용프로그램에서 추가 처리를 위해 정수 양식의 EMPNO 컬럼을 선택하십시오.

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```

- BIRTHDATE 컬럼(날짜)에 '1964-07-20'에 해당하는 내부 값이 있다고 가정하십시오.

```
INTEGER(BIRTHDATE)
```

  결과 값은 19 640 720입니다.

- STARTTIME 컬럼(시간)에 '12:03:04'에 해당하는 내부 값이 있다고 가정하십시오.

```
INTEGER(STARTTIME)
```

결과 값은 120 304입니다.

JULIAN_DAY



스키마는 SYSFUN입니다.

B.C. 4713년 1월 1일(율리우스력의 시작)에서 인수에 지정된 날짜 값까지의 일 수를 나타내는 정수 값을 리턴합니다.

인수는 날짜, 시간소인, 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

LAST_DAY



스키마는 SYSIBM입니다.

LAST_DAY 스칼라 함수는 인수 월의 마지막 날을 표시하는 날짜 및 시간 값을 리턴합니다.

expression
시작 날짜를 지정하는 표현식입니다. 표현식은 내장 데이터 유형인 날짜 또는 시간소인 중 하나의 값을 리턴해야 합니다.

함수의 결과는 결과 데이터 유형이 DATE인 경우 *expression*이 문자열이 아니면 *expression*과 같은 데이터 유형을 갖습니다. 결과는 널(NULL)이 될 수 있고, *date-expression*의 값이 널(NULL)이면 결과는 널(NULL) 값입니다.

*expression*에 포함된 시, 분, 초 또는 소수 초 정보는 함수에서 변경되지 않습니다.

예:

- 호스트 변수 *END_OF_MONTH*를 현재 월의 마지막 날로 설정하십시오.

```
SET :END_OF_MONTH = LAST_DAY(CURRENT_DATE);
```

  호스트 변수 *END_OF_MONTH*는 현재 월의 끝을 표시하는 값으로 설정됩니다. 현재 일이 2000-02-10이면, *END_OF_MONTH*는 2000-02-29로 설정됩니다.

- 호스트 변수 *END_OF_MONTH*를 지정된 날짜에 대한 EUR 형식으로 당월의 마지막 날로 설정하십시오.

```
SET :END_OF_MONTH = CHAR(LAST_DAY('1965-07-07'), EUR);
```

  호스트 변수 *END_OF_MONTH*는 값 '31.07.1965'로 설정됩니다.

LCASE



스키마는 SYSIBM입니다.

LCASE 함수는 모든 SBCS 문자가 소문자로 변환된 문자열을 리턴합니다.

LCASE는 LOWER의 동의어입니다.

LCASE(로케일 구분)



스키마는 SYSIBM입니다.

LCASE 함수는 지정된 로케일과 연관된 규칙을 사용하여 모든 문자가 소문자로 변환된 문자열을 리턴합니다.

LCASE는 LOWER의 동의어입니다.

LEAST



스키마는 SYSIBM입니다.

LEAST 함수는 값 세트에서 최소값을 리턴합니다.

인수는 호환 가능해야 하며 각 인수는 ARRAY, LOB, LONG VARCHAR, LONG VARGRAPHIC, XML, 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 아닌 다른 데이터 유형의 값을 리턴하는 표현식이어야 합니다(SQLSTATE 42815). 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 호환 가능한 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 서명을 작성할 필요가 없습니다.

필요한 경우 선택된 인수는 결과의 속성으로 변환됩니다. 결과의 속성은 결과 데이터 유형의 규칙을 기초로 모든 피연산자에 의해 판별됩니다.

함수 결과는 가장 작은 인수 값입니다. 하나 이상의 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 어느 한 인수가 널(NULL)일 경우 결과는 널(NULL) 값이 됩니다.

LEAST 스칼라 함수는 MIN 스칼라 함수의 동의어입니다.

예를 들면, 다음과 같습니다.

테이블 T1에 값이 각각 1, 7, 4인 세 개의 컬럼 C1, C2, C3이 있다고 가정합니다. 다음 쿼리는

```
SELECT LEAST (C1, C2, C3) FROM T1
```

1을 리턴합니다.

컬럼 C3에 4 대신 NULL 값이 있는 경우 동일한 쿼리는 NULL을 리턴합니다.

LEFT



스키마는 SYSIBM입니다. LEFT 함수의 SYSFUN 버전은 계속 사용 가능합니다.

LEFT 함수는 지정된 문자열 단위에서 표현된 길이 *length*의 *string-expression*의 맨 왼쪽 문자열을 리턴합니다. *string-expression*이 문자열이면 결과가 문자열입니다. *string-expression*이 그래픽 문자열이면 결과는 그래픽 문자열입니다.

string-expression

결과가 파생될 문자열을 지정하는 표현식입니다. 이 표현식은 내장 문자열, 숫자 또는 날짜 및 시간 데이터 유형의 값을 리턴합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *string-expression*의 하위 문자열은 *string-expression*의 0 이상의 인접 코드 포인트입니다.

length

결과 길이를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 명시적으로 INTEGER로 캐스트됩니다.값은 내재적으로나 명시적으로 지정된 단위에서 표현된 0과 *string-expression*의 길이 사이여야 합니다(SQLSTATE 22011). OCTETS가 지정되어 있고 결과가 그래픽 데이터이면, 값은 0과 *string-expression*의 속성 길이의 두 배 사이여야 합니다(SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 또는 OCTETS

*length*의 문자열 단위를 지정합니다.

CODEUNITS16은 *length*를 16비트 UTF-16 코드 단위로 표시할 것을 지정합니다. CODEUNITS32는 *length*가 32비트 UTF-32 코드 단위로 표시할 것을 지정합니다. OCTETS는 *length*를 바이트로 표시할 것을 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *string-expression*이 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *string-expression*이 그래픽 문자열이면, *length*는 짝수여야 합니다. 그 외의 경우에는 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 결과가 그래픽 데이터이면 *length*는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다. CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는, 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

*string-expression*은 *string-expression*의 지정된 부속 문자열이 항상 존재하도록 필요한 수의 공백이 오른쪽에 채워집니다. 채우는 데 사용된 문자는 채울 컨텍스트의 문자열을 채우는 데 사용된 문자와 같습니다. 채우기에 관한 자세한 정보는 『지정 및 비교』에서 『문자열 지정』을 참조하십시오.

함수의 결과는 *string-expression*의 길이 속성과 같은 길이 속성인 다양한 길이 문자열이며 *string-expression*의 데이터 유형에 따라 다른 데이터 유형입니다.

- *string-expression*이 CHAR 또는 VARCHAR인 경우 VARCHAR
- *string-expression*이 CLOB인 경우 CLOB
- *string-expression*이 GRAPHIC 또는 VARGRAPHIC인 경우 VARGRAPHIC
- *string-expression*이 DBCLOB인 경우 DBCLOB
- *string-expression*이 BLOB인 경우 BLOB

결과의 실제 길이는(문자열 단위에서) *length*입니다.

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- 변수 ALPHA에 값 'ABCDEF'가 있다고 가정하십시오. 다음 명령문:

  ```
  SELECT LEFT(ALPHA,3)
    FROM SYSIBM.SYSDUMMY1
  ```

  ALPHA에서 맨 왼쪽의 3개의 문자인 'ABC'를 리턴합니다.

- VARCHAR(50)로 정의된 변수 NAME에 값 'KATIE AUSTIN'이 있으며 정수 변수 FIRSTNAME_LEN에 값 5가 있다고 가정하십시오. 다음 명령문은

  ```
  SELECT LEFT(NAME,FIRSTNAME_LEN)
    FROM SYSIBM.SYSDUMMY1
  ```

  값 'KATIE'를 리턴합니다.

- 다음 명령문은 길이가 0인 문자열을 리턴합니다.

  ```
  SELECT LEFT('ABCABC',0)
    FROM SYSIBM.SYSDUMMY1
  ```

- EMPLOYEE 테이블의 FIRSTNME 컬럼은 VARCHAR(12)로 정의됩니다. 성이 'BROWN'인 직원의 이름을 찾은 다음 10바이트 문자열의 이름을 리턴합니다.

  ```
  SELECT LEFT(FIRSTNME, 10)
    FROM EMPLOYEE
    WHERE LASTNAME = 'BROWN'
  ```

  값 'DAVID' 다음에 5개의 공백 문자가 있는 VARCHAR(12) 문자열을 리턴합니다.

- 유니코드 데이터베이스에서 FIRSTNAME은 VARCHAR(12) 컬럼입니다. 해당 값 중 하나는 6자 문자열 'Jürgen'입니다. FIRSTNAME이 이 값을 가질 때 리턴되는 값은 다음과 같습니다.

```
함수...                          다음을 리턴...

LEFT(FIRSTNAME,2,CODEUNITS32)   'Jü' -- x'4AC3BC'
LEFT(FIRSTNAME,2,CODEUNITS16)   'Jü' -- x'4AC3BC'
LEFT(FIRSTNAME,2,OCTETS)        'J'  -- x'4A20', a truncated string
```

- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

20바이트의 길이 속성이 있는 변수 UTF8_VAR에 문자열의 UTF-8 표현을 포함한다고 가정하십시오.

```
SELECT LEFT(UTF8_VAR, 2, CODEUNITS16),
  LEFT(UTF8_VAR, 2, CODEUNITS32),
  LEFT(UTF8_VAR, 2, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&', '&N' 및 'bb'를 각각 리턴하고, 여기서 'b'는 공백 문자를 나타냅니다.

```
SELECT LEFT(UTF8_VAR, 5, CODEUNITS16),
  LEFT(UTF8_VAR, 5, CODEUNITS32),
  LEFT(UTF8_VAR, 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~A', '&N~AB' 및 '&N'을 각각 리턴합니다.

```
SELECT LEFT(UTF8_VAR, 10, CODEUNITS16),
  LEFT(UTF8_VAR, 10, CODEUNITS32),
  LEFT(UTF8_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~ABbbbb', '&N~ABbbbbb' 및 '&N~ABb'를 각각 리턴하고, 여기서 'b'는 공백 문자를 나타냅니다.

20 코드 단위의 길이 속성이 있는 변수 UTF16_VAR에 문자열의 UTF-16BE 표현을 포함한다고 가정하십시오.

```
SELECT LEFT(UTF16_VAR, 2, CODEUNITS16),
  LEFT(UTF16_VAR, 2, CODEUNITS32),
  HEX (LEFT(UTF16_VAR, 2, OCTETS))
FROM SYSIBM.SYSDUMMY1
```

값 '&', '&N' 및 X'D834'를 각각 리턴하고, 여기서 X'D834'는 일치되지 않은 문자를 나타냅니다.

```
SELECT LEFT(UTF16_VAR, 5, CODEUNITS16),
  LEFT(UTF16_VAR, 5, CODEUNITS32),
  LEFT(UTF16_VAR, 6, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~A', '&N~AB' 및 '&N'을 각각 리턴합니다.

```
SELECT LEFT(UTF16_VAR, 10, CODEUNITS16),
  LEFT(UTF16_VAR, 10, CODEUNITS32),
  LEFT(UTF16_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~ABbbbb', '&N~ABbbbbb' 및 '&N~A'를 각각 리턴하고, 여기서 'b'는 공백 문자를 나타냅니다.

LENGTH



스키마는 SYSIBM입니다.

LENGTH 함수는 내재적 또는 명시적 문자열 단위의 *expression* 길이를 리턴합니다.

expression
내장 데이터 유형인 값을 리턴하는 표현식입니다. *expression*이 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, *expression*이 널(NULL)이면 결과도 널(NULL)이 됩니다.

CODEUNITS16, CODEUNITS32 또는 OCTETS
결과의 문자열 단위를 지정합니다. CODEUNITS16은 결과가 16비트 UTF-16 코드 단위로 표시될 것을 지정합니다. CODEUNITS32는 결과가 32비트 UTF-32 코드 단위로 표시될 것을 지정합니다. OCTETS는 결과가 바이트 단위로 표시될 것을 지정합니다.

문자열 단위가 명시적으로 지정되고 *expression*이 문자열 데이터가 아니면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *expression*이 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *expression*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815). CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 만일 결과가 그래픽 데이터이면 리턴된 값은 2바이트 단위의 길이를 지정합니다. 결과가 그래픽 데이터가 아니면 리턴된 값은 바이트 단위의 길이를 지정합니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

문자 또는 그래픽 문자열의 길이는 뒤 공백을 포함합니다. 실행 파일 문자열의 길이는 2진 0을 포함합니다. 가변 길이 문자열의 실이는 실제 길이이며 최대 길이가 아닙니다. 다른 모든 값의 길이는 값을 표시하는 데 사용되는 바이트 수입니다.

- 작은 정수: 2
- 큰 정수: 4
- 정밀도가 *p*인 10진수: (*p*/2)+1
- DECFLOAT(16)의 경우 8

- DECFLOAT(34)의 경우 16
- 실행 파일 문자열: 문자열의 길이
- 문자열: 문자열의 길이
- 단정밀도 부동 소수점: 4
- 배정밀도 부동 소수점: 8
- 날짜: 4
- 시간: 3
- TIMESTAMP(*p*)의 경우 7 + (*p* + 1) / 2

예를 들면, 다음과 같습니다.

- ADDRESS 호스트 변수를 값이 '895 Don Mills Road'인 가변 길이 문자열이라고 가정하십시오.

  ```
  LENGTH(:ADDRESS)
  ```

  값 18을 리턴합니다.
- START_DATE를 유형이 DATE인 컬럼으로 가정하십시오.

  ```
  LENGTH(START_DATE)
  ```

  값 4를 리턴합니다.
- ```
  LENGTH(CHAR(START_DATE, EUR))
  ```

  값 10을 리턴합니다.
- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'
UTF-32BE	X'0001D11E'	X'0000004E'	X'00000303'	X'00000041'	X'00000042'

  변수 UTF8_VAR에 해당 문자열의 UTF-8 표시가 들어 있다고 가정하십시오.

  ```
  SELECT LENGTH(UTF8_VAR, CODEUNITS16),
    LENGTH(UTF8_VAR, CODEUNITS32),
    LENGTH(UTF8_VAR, OCTETS)
  FROM SYSIBM.SYSDUMMY1
  ```

  리턴되는 값은 각각 6, 5, 9입니다.

  변수 UTF16_VAR에 해당 문자열의 UTF-16BE 표시가 들어 있다고 가정하십시오.
  ```

```
SELECT LENGTH(UTF16_VAR, CODEUNITS16),
   LENGTH(UTF16_VAR, CODEUNITS32),
   LENGTH(UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

리턴되는 값은 각각 6, 5, 12입니다.

LN

스키마는 SYSIBM입니다. (LN 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

LN 함수는 숫자의 대수를 리턴합니다. EXP 및 LN 함수는 역연산입니다.

인수는 내장 숫자 데이터 유형 값을 리턴하는 표현식이어야 합니다. 인수가 10진수 부동 소수점인 경우 연산은 10진수 부동 소수점으로 수행됩니다. 기타의 경우 인수는 배정밀도 부동 소수점 숫자로 변환되어 함수가 처리됩니다.인수의 값은 0보다 커야 합니다(SQLSTATE 22003).

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주

- ***DECFLOAT 특수값을 포함하는 결과:*** 10진수 부동 소수점의 경우, 특수값은 다음과 같이 처리됩니다.
  - LN(NaN)은 NaN을 리턴합니다.
  - LN(-NaN)은 -NaN을 리턴합니다.
  - LN(Infinity)은 Infinity를 리턴합니다.
  - LN(-Infinity)은 NaN 및 경고를 리턴합니다.
  - LN(sNaN)은 NaN 및 경고를 리턴합니다.
  - LN(-sNaN)은 -NaN 및 경고를 리턴합니다.
  - LN(DECFLOAT('0')은 -Infinity를 리턴합니다.
- ***구문 대체:*** LN 대신 LOG를 지정할 수 있습니다. 이전 버전의 DB2 제품과의 호환성을 위해서만 지원됩니다. 일부 데이터베이스 관리 프로그램과 응용프로그램이 LOG를 숫자의 자연 대수 대신 숫자의 일반 대수로 구현하기 때문에 LN이 LOG 대신 사용되어야 합니다.

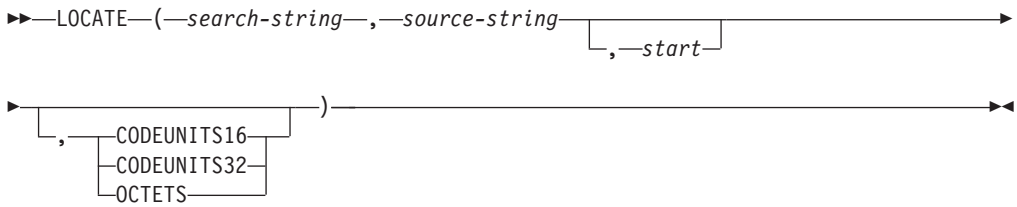
예:

- NATLOG가 값이 31.62인 DECIMAL(4,2) 호스트 변수라고 가정해 보십시오.

```
VALUES LN(:NATLOG)
```

  근사 값 3.45를 리턴합니다.

LOCATE



스키마는 SYSIBM입니다. LOCATE 함수의 SYSFUN 버전은 계속해서 사용할 수 있지만, 데이터베이스 조합에 민감하지 않습니다.

LOCATE 함수는 한 문자열(*source-string*) 내에서 다른 문자열(*search-string*)의 첫 번째 어커런스의 시작 위치를 리턴합니다. *search-string*이 발견되지 않거나 인수가 널(NULL)이 아니면 결과는 0이 됩니다. *search-string*이 발견되면 결과는 1부터 *source-string*의 실제 길이까지가 됩니다. 실행 파일 비교를 사용하여 검색이 수행되는 경우, *search-string* 또는 *source-string*이 FOR BIT DATA로 정의되지 않으면, 데이터베이스 조합을 사용하여 검색이 수행됩니다.

선택적 *start*가 지정되는 경우 검색이 시작되는 *source-string*의 문자 위치를 표시합니다. *start* 및 함수 결과의 표현 단위를 표시하기 위해 선택적 문자열 단위를 지정할 수 있습니다.

*search-string*의 길이가 0이면 함수에서 리턴되는 결과는 1이며, *source-string*의 길이가 0이면 함수에서 리턴되는 결과는 0입니다. 그 외의 경우는 다음과 같습니다.

- *search-string* 값이 *source-string* 값 이내의 근접 위치 부속 문자열의 동일 길이와 같은 경우, 함수가 리턴하는 결과는 *source-string* 값 이내의 첫 번째 해당 부속 문자열의 시작 위치입니다.
- 그렇지 않을 경우, 함수에서 리턴되는 결과는 0입니다.

search-string

검색의 오브젝트인 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BLOB, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC 또는 BLOB 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식은 BLOB 파일 참조 변수가 될 수 없습니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수

- 위의 항목을 병합(CONCAT 또는 ||를 사용하여)하는 표현식
- SQL 프로시저 매개변수

이들 규칙은 LIKE 술어에 대한 *pattern-expression*에 대해 설명된 규칙과 유시합니다.

source-string

검색이 수행될 문자열을 지정하는 표현식입니다. 표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아니면, 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수(로케이터 변수 또는 파일 참조 변수 포함)
- 스칼라 함수
- 대형 오브젝트(LOB) 로케이터
- 컬럼 이름
- 위의 항목을 병합(CONCAT 또는 ||를 사용하여)하는 표현식

start

검색이 시작될 *source-string* 내의 위치를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 명시적으로 INTEGER로 캐스트됩니다.정수의 값은 0 이상이어야 합니다. *start*가 지정되면 LOCATE 함수는 다음과 유사하게 표현됩니다.

```
POSITION(search-string,
  SUBSTRING(source-string, start, string-unit),
  string-unit) + start - 1
```

여기서 *string-unit*은 CODEUNITS16, CODEUNITS32 또는 OCTETS입니다.

*start*가 지정되지 않으면 검색은 소스 문자열의 첫 번째 위치에서 시작되고 LOCATE 함수는 다음과 유사하게 표현됩니다.

```
POSITION(search-string, source-string, string-unit)
```

CODEUNITS16, CODEUNITS32 또는 OCTETS

start 및 결과의 문자열 단위를 지정합니다. CODEUNITS16는 *start* 및 결과를 16비트 UTF-16 코드 단위로 표현하도록 지정합니다. CODEUNITS32는 *start* 및 결과를 32비트 UTF-32 코드 단위로 표현하도록 지정합니다. OCTETS *start* 및 결과를 바이트 단위로 표현하도록 지정합니다

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *search-string* 또는 *source-string*이 실행 파일 문자열 또는 비트 데이터이면, 오류가 리턴됩니다 (SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *search-string* 및 *source-string*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815).

문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 결과가 그래픽 데이터이면 *start* 및 리턴 위치는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다.

로케일 구분 UCA-기반의 조합은 이 함수에 대해 사용되며 CODEUNITS16 옵션은 최상의 성능 특성을 제공합니다.

CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는, 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

첫 번째와 두 번째 인수는 호환 가능한 문자열 유형을 가져야 합니다. 호환성에 대한 자세한 정보는 『문자열 변환에 대한 규칙』을 참조하십시오. 유니코드 데이터베이스에서 하나의 문자열 인수가 문자이고(FOR BIT DATA가 아닌) 다른 문자열 인수가 그래픽인 경우, *search-string*이 *source-string*의 데이터 유형으로 변환되어 처리됩니다. 하나의 인수가 문자 FOR BIT DATA인 경우, 다른 인수는 그래픽이어서는 안됩니다 (SQLSTATE 42846).

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- 문자열 'DINING'의 문자 'N'의 첫 번째 어커런스 위치를 찾으십시오.

```
SELECT LOCATE('N', 'DINING')
   FROM SYSIBM.SYSDUMMY1
```

  결과는 값 3입니다.

- 테이블 IN_TRAY의 모든 행에 대해 NOTE_TEXT 컬럼 내의 문자열 'GOOD'의 시작 위치, RECEIVED 컬럼 및 SUBJECT 컬럼을 선택하십시오.

```
SELECT RECEIVED, SUBJECT, LOCATE('GOOD', NOTE_TEXT)
   FROM IN_TRAY
   WHERE LOCATE('GOOD', NOTE_TEXT) <> 0
```

- 문자열 'Jürgen lives on Hegelstraße'에서 문자 'ß'를 찾은 후에, 문자열 내에 CODEUNITS32 단위로 계산한 대로 위치가 있는 호스트 변수 LOCATION을 설정하십시오.

```
SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, CODEUNITS32)
```

  호스트 변수 LOCATION의 값은 26으로 설정됩니다.

- 문자열 'Jürgen lives on Hegelstraße'에서 문자 'ß'를 찾은 후에, 문자열 내에 CODEUNITS16 단위로 계산한 대로 위치가 있는 호스트 변수 LOCATION을 설정하십시오.

```
SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, CODEUNITS16)
```

  호스트 변수 LOCATION의 값은 26으로 설정됩니다.

- 문자열 'Jürgen lives on Hegelstraße'에서 문자 'ß'를 찾은 후에, 문자열 내에 OCTETS로 계산한 대로 위치가 있는 호스트 변수 LOCATION을 설정하십시오.

```
SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, OCTETS)
```

  호스트 변수 LOCATION 값은 27로 설정됩니다.

- 다음의 예는 '&'가 음악 기호 G 음자리표이고 '~'가 공백을 포함하지 않은 틸드 문자 조합인 유니코드 문자열 '&N~AB'를 사용합니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  변수 UTF8_VAR에 해당 문자열의 UTF-8 표시가 들어 있다고 가정하십시오.

```
SELECT LOCATE('~', UTF8_VAR, CODEUNITS16),
  LOCATE('~', UTF8_VAR, CODEUNITS32),
  LOCATE('~', UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  값 4, 3, 및 6을 각각 리턴합니다.

  변수 UTF16_VAR에 해당 문자열의 UTF-16BE 표시가 들어 있다고 가정하십시오.

```
SELECT LOCATE('~', UTF16_VAR, CODEUNITS16),
   LOCATE('~', UTF16_VAR, CODEUNITS32),
   LOCATE('~', UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

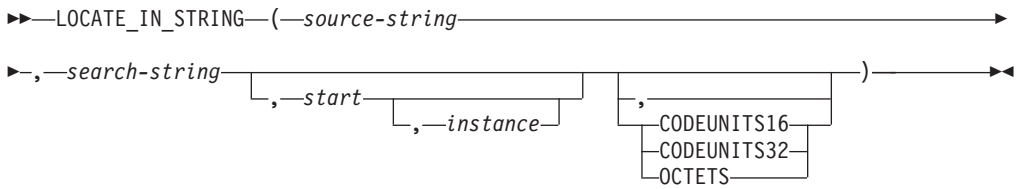
  값 4, 3, 및 7을 각각 리턴합니다.

- 대소문자가 구별되지 않는 조합 UCA500R1_LEN_S1으로 작성된 유니코드 데이터베이스에서, 'The quick brown fox' 문구에서 단어 'Brown'의 위치를 찾습니다.

```
SET :LOCATION = LOCATE('Brown', 'The quick brown fox', CODEUNITS16)
```

  호스트 변수 LOCATION 값은 11로 설정됩니다.

LOCATE_IN_STRING



스키마는 SYSIBM입니다.

LOCATE_IN_STRING 함수는 한 문자열(*source-string*) 내에서 다른 문자열(*search-string*)의 시작 위치를 리턴합니다. *search-string*이 발견되지 않거나 인수가 널(NULL)이 아니면 결과는 0이 됩니다. *search-string*이 발견되면 결과는 1부터 *source-string*의 실제 길이까지가 됩니다. *search-string* 또는 *source-string*이 FOR BIT DATA로 정의되지 않으면(이 경우 검색은 2진 비교를 사용하여 수행됨), 검색은 데이터베이스 조합을 사용하여 수행됩니다.

선택적 *start*가 지정되는 경우 검색이 시작되는 *source-string*의 문자 위치를 표시합니다. *start*를 지정한 경우 인스턴스 번호도 지정할 수 있습니다. *instance* 인수를 사용하여 *source-string* 내에서 *search-string*의 특정 어커런스 위치를 판별하는 데 사용됩니다. *start* 및 함수 결과의 표현 단위를 표시하기 위해 선택적 문자열 단위를 지정할 수 있습니다.

*search-string*의 길이가 0인 경우 함수에서 리턴되는 결과는 1이고 *source-string*의 길이가 0인 경우 함수에서 리턴되는 결과는 0입니다. 어느 조건도 존재하지 않고 *search-string* 값이 *source-string* 값 이내의 근접 위치 부속 문자열 길이와 같은 경우, 함수가 리턴하는 결과는 *source-string* 값 내에서 해당 부속 문자열의 시작 위치입니다. 그렇지 않으면 함수는 결과로 0을 리턴합니다.

source-string

검색이 수행될 문자열을 지정하는 표현식입니다. 이 표현식은 내장 문자열, 숫자 또는 날짜 및 시간 데이터 유형의 값을 리턴합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수(LOB 로케이터 변수 또는 파일 참조 변수)
- 스칼라 함수
- 대형 오브젝트(LOB) 로케이터
- 컬럼 이름

- 이전 항목 중 하나를 병합하는(CONCAT 또는 ||을 사용하여) 표현식

search-string

검색의 오브젝트인 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BLOB, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC 또는 BLOB 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 실제 길이는 VARCHAR의 최대 길이를 초과해서는 안됩니다. *search-string*은 BLOB 파일 참조 변수가 될 수 없습니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수
- 인수가 이전 항목 중 하나인 스칼라 함수
- 이전 항목 중 하나를 병합하는(CONCAT 또는 ||을 사용하여) 표현식

이 규칙은 LIKE 술어에 대한 *pattern-expression*에 대해 설명된 규칙과 유사합니다.

start

일치 검색이 시작될 *source-string* 내의 위치를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다.

정수 값이 0보다 큰 경우 검색은 *start*에서 시작하여 문자열 끝까지 위치마다 계속됩니다. 정수 값이 0 미만인 경우 검색은 LENGTH(*source-string*) + *start* + 1에서 시작하여 문자열 시작 부분까지 위치마다 계속됩니다.

*start*를 지정하지 않은 경우 디폴트는 1입니다. 정수 값이 0이면 오류가 리턴됩니다(SQLSTATE 42815).

인스턴스

source-string 내에서 검색할 *search-string* 인스턴스를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. *instance*를 지정하지 않은 경우 디폴트는 1입니다. 정수 값은 1 이상이어야 합니다(SQLSTATE 42815).

CODEUNITS16, CODEUNITS32 또는 OCTETS

start 및 결과의 문자열 단위를 지정합니다. CODEUNITS16는 *start* 및 결과를 16비트 UTF-16 코드 단위로 표현하도록 지정합니다. CODEUNITS32는 *start* 및 결

과를 32비트 UTF-32 코드 단위로 표현하도록 지정합니다. OCTETS *start* 및 결과를 바이트 단위로 표현하도록 지정합니다

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *search-string* 또는 *source-string*이 실행 파일 문자열 또는 비트 데이터이면, 오류가 리턴됩니다 (SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *search-string* 및 *source-string*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815).

문자열 단위가 명시적으로 지정되지 않으면 *source-string*의 데이터 유형이 사용되는 문자열 단위를 결정합니다. *source-string*이 그래픽 데이터이면 *start* 및 리턴 위치는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다.

로케일이 구분되는 UCA 기반 조합이 이 함수에 사용되는 경우, CODEUNITS16 옵션은 최상의 성능 특성을 제공합니다.

CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 ″문자열″에서 ″내장 함수의 문자열 단위″를 참조하십시오.

첫 번째와 두 번째 인수는 호환 가능한 문자열 유형을 가져야 합니다. 호환성에 대한 자세한 정보는 ″문자열 변환에 대한 규칙″을 참조하십시오. 유니코드 데이터베이스에서, 하나의 문자열 인수가 문자이고(FOR BIT DATA가 아니라) 다른 문자열 인수가 그래픽인 경우 *search-string*은 처리를 위해 *source-string* 데이터 유형으로 변환됩니다. 하나의 인수가 문자 FOR BIT DATA인 경우, 다른 인수는 그래픽이 될 수 없습니다 (SQLSTATE 42846).

검색 위치마다, 해당 위치에 있는 서브스트링과 *source-string*에서 검색 위치 오른쪽에 있는 LENGTH(*search-string*) - 1 값이 *search-string*과 같을 경우 일치가 발견됩니다.

함수의 결과는 정수(integer)입니다. 결과는 *source-string* 내에서 *search-string* 인스턴스의 시작 위치입니다. 값은 문자열 시작 부분과 상대적입니다(*start* 스펙에 관계없이). 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

LOCATE_IN_STRING의 동의어로 INSTR을 사용할 수 있습니다.

예:

- 문자열의 끝으로부터 검색하여 문자열 'Jürgen lives on Hegelstraße'에서 문자 'ß'를 찾은 후에, 문자열 내에 CODEUNITS32 단위로 계산한 대로 위치가 있는 호스트 변수 LOCATION을 설정하십시오.

```
SET :POSITION = LOCATE_IN_STRING('Jürgen lives on Hegelstraße',
                                 'ß',-1,CODEUNITS32);
```

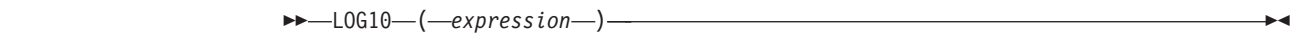
  호스트 변수 POSITION의 값은 26으로 설정됩니다.

- 문자열의 시작으로부터 검색하여 'WINNING' 문자열에서 'N' 문자의 세 번째 어커런스 위치를 찾은 후 이 문자 위치를 사용하여 호스트 변수 POSITION을 문자열에서 바이트 단위로 설정하십시오.

  ```
  SET :POSITION =
  LOCATE_IN_STRING('WINNING','N',1,3,OCTETS);
  ```

  호스트 변수 POSITION의 값은 6으로 설정됩니다.

LOG10



스키마는 SYSIBM입니다. (LOG1O 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

LOG10 함수는 숫자의 일반 대수(기본 10)를 리턴합니다.

인수는 내장 숫자 데이터 유형 값을 리턴하는 표현식이어야 합니다. 인수가 10진수 부동 소수점인 경우 연산은 10진수 부동 소수점으로 수행됩니다. 기타의 경우 인수는 배정밀도 부동 소수점 숫자로 변환되어 함수가 처리됩니다. 인수의 값은 0보다 커야 합니다(SQLSTATE 22003).

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주

- ***DECFLOAT 특수값을 포함하는 결과:*** 10진수 부동 소수점의 경우, 특수값은 다음과 같이 처리됩니다.
  - LOG10(NaN)은 NaN을 리턴합니다.
  - LOG10(-NaN)은 -NaN을 리턴합니다.
  - LOG10(Infinity)은 Infinity를 리턴합니다.
  - LOG10(-Infinity)은 NaN 및 경고를 리턴합니다.
  - LOG10(sNaN)은 NaN 및 경고를 리턴합니다.
  - LOG10(-sNaN)은 -NaN 및 경고를 리턴합니다.
  - LOG10(DECFLOAT('0')은 -Infinity를 리턴합니다.

예 :

- L이 값이 31.62인 DECIMAL(4,2) 호스트 변수라고 가정해 보십시오.

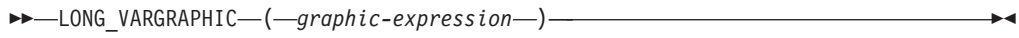
```
VALUES LOG10(:L)
```

  DOUBLE 값 +1.49996186559619E+000을 리턴합니다.

LONG_VARCHAR

LONG_VARCHAR 함수는 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 함수는 이전 DB2 버전과 호환됩니다.

LONG_VARGRAPHIC



LONG_VARGRAPHIC 함수는 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 함수는 이전 DB2 버전과 호환됩니다.

LOWER

스키마는 SYSIBM입니다. (이 함수의 SYSFUN 버전은 LONG VARCHAR 및 CLOB 인수를 지원하여 계속 사용 가능합니다.)

LOWER 함수는 모든 SBCS 문자가 소문자로 변환된 문자열을 리턴합니다. 즉 A - Z 문자는 a - z 문자로 변환되고, 그밖의 문자가 존재하는 경우는 해당하는 소문자로 변환됩니다. 예를 들어 코드 페이지 850에서, É는 é에 맵핑됩니다. 결과 문자의 코드 포인트 길이가 소스 문자의 코드 포인트 길이와 같지 않으면 소스 문자는 변환되지 않습니다. 모든 문자가 변환되지는 않기 때문에, LOWER(UPPER(*string-expression*))이 반드시 LOWER(*string-expression*)과 동일한 결과를 리턴하지는 않습니다.

인수는 값이 CHAR 또는 VARCHAR 데이터 유형인 표현식이어야 합니다.

함수의 결과는 인수와 동일한 데이터 유형 및 길이 속성을 가집니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

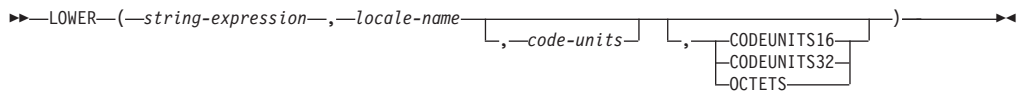
예:

EMPLOYEE 테이블에서 컬럼 JOB 값의 문자가 소문자로 리턴되도록 하십시오.

```
SELECT LOWER(JOB)
  FROM EMPLOYEE
  WHERE EMPNO = '000020';
```

결과는 값 'manager'입니다.

LOWER(로케일 구분)



스키마는 SYSIBM입니다.

LOWER 함수는 지정된 로케일과 연관된 규칙을 사용하여 모든 문자가 소문자로 변환된 문자열을 리턴합니다.

string-expression

CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 문자열을 리턴하는 표현식입니다. *string-expression*이 CHAR 또는 VARCHAR이면, 표현식은 FOR BIT DATA가 될 수 없습니다(SQLSTATE 42815).

locale-name

소문자로의 변환에 대한 규칙을 정의하는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자가 구분되지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 해당되는 이름 지정 방법에 대한 정보는 『SQL 및 XQuery의 로케일 이름』을 참조하십시오.

code-units

결과의 코드 단위 수를 지정하는 정수 상수입니다. 지정된 경우 *code-units*는 결과가 문자 데이터인 경우 1 - 32 672 사이의 정수여야 하고 결과가 그래픽 데이터인 경우 1 - 16 336 사이의 정수여야 합니다(SQLSTATE 42815). *code-units*를 명시적으로 지정하지 않은 경우 내재적으로 *string-expression*의 길이 속성이 됩니다. OCTETS가 지정되고 결과가 그래픽 데이터일 경우, *code-units*의 값은 짝수여야 합니다(SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 또는 OCTETS

*code-units*의 문자열 단위를 지정합니다.

CODEUNITS16은 *code-units*가 16비트 UTF-16 코드 단위로 표시될 것을 지정합니다. CODEUNITS32는 *code-units*가 32비트 UTF-32 코드 단위로 표시될 것을 지정합니다. OCTETS는 *code-units*가 바이트 단위로 표시됨을 지정합니다.

문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 결과가 그래픽 데이터이면 *code-units*는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다. CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 『문자열』에서 『내장 함수의 문자열 단위』를 참조하십시오.

함수의 결과는 *string-expression*이 CHAR 또는 VARCHAR인 경우 VARCHAR이고, *string-expression*이 GRAPHIC 또는 VARGRAPHIC인 경우 VARGRAPHIC입니다.

결과의 길이 속성은 다음 테이블에 표시된 것처럼 *code-units*의 내재된 또는 명시적 값, 내재된 또는 명시적 문자열 단위, 그리고 결과 데이터 유형으로 결정됩니다.

표 48. 문자열 단위 및 결과 유형 함수로서의 LOWER 결과의 길이 속성

문자열 단위	문자 결과 유형	그래픽 결과 유형
CODEUNITS16	MIN(*code-units* * 3, 32672)	*code-units*
CODEUNITS32	MIN(*code-units* * 4, 32672)	MIN(*code-units* * 2, 16336)
OCTETS	*code-units*	MIN(*code-units* / 2, 16336)

결과의 실제 길이는 *string-expression*의 길이보다 길 수 있습니다. 결과의 실제 길이가 결과의 길이 속성보다 길면 오류가 리턴됩니다(SQLSTATE 42815). 결과의 코드 단위 수가 *code-units*의 값을 초과하면 오류가 리턴됩니다(SQLSTATE 42815).

*string-expression*이 UTF-16으로 되어 있지 않으면, 이 함수는 *string-expression*에서 UTF-16으로, 그리고 UTF-16에서 *string-expression*의 코드 페이지로의 변환을 수행합니다. 코드 페이지 변환의 결과에 최소 하나의 대체 문자가 포함되어 있으면 결과에는 대체 문자가 포함되고, 경고가 리턴되며(SQLSTATE 01517) SQLCA의 경고 플래그 SQLWARN8이 'W'로 설정됩니다.

첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블에서 컬럼 JOB 값의 문자가 소문자로 리턴되도록 하십시오.

```
SELECT LOWER(JOB, 'en_US')
  FROM EMPLOYEE
  WHERE EMPNO = '000020'
```

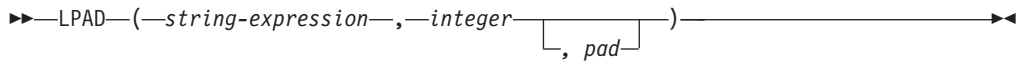
  결과는 값 'manager'입니다.

- 터키어 문자열에서 모든 'I' 문자에 대한 소문자를 찾으십시오.

```
VALUES LOWER('I□ıi', 'tr_TR', CODEUNITS16)
```

  결과는 문자열 'ıiii'입니다.

LPAD



스키마는 SYSIBM입니다.

LPAD 함수는 왼쪽이 *패*드 또는 공백으로 채워지는 *string-expression* 구성 문자열을 리턴합니다. LPAD 함수는 *string-expression*의 앞 또는 뒤 공백을 의미있는 것으로 처리합니다. *string-expression*의 실제 길이가 *integer*보다 작고 *pad*가 비어 있는 문자열이 아닌 경우에만 채우기가 발생합니다.

string-expression
소스 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

integer
결과 길이를 지정하는 정수 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. 값은 *n* 이하의 0 또는 양의 정수여야 합니다. 여기서 *n*은 *string-expression*이 문자열인 경우 32 672이고 *string-expression*이 그래픽 문자열인 경우 16 336입니다.

pad
채울 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

*pad*를 지정하지 않는 경우 패드 문자는 다음과 같이 판별됩니다.

- *string-expression*이 문자열인 경우 SBCS 공백 문자
- *string-expression*이 그래픽 문자열인 경우 표의 문자에서의 공백 문자. EUC 데이터베이스의 그래픽 문자열의 경우 X'3000'이 사용됩니다. 유니코드 데이터베이스의 그래픽 문자열의 경우 X'0020'이 사용됩니다.

함수의 결과는 코드 페이지가 *string-expression*과 같은 가변 길이 문자열입니다. *string-expression*의 값과 *pad*의 값은 호환 가능한 데이터 유형을 가지고 있어야 합니다. *string-expression* 및 *pad*가 다른 코드 페이지를 가지고 있는 경우 *pad*는

*string-expression*의 코드 페이지로 변환됩니다. *string-expression* 또는 *pad* 중 하나가 FOR BIT DATA인 경우 어떤 문자 변환도 발생하지 않습니다.

결과의 길이 속성은 *integer*의 값이 함수 호출을 포함하는 SQL문이 컴파일될 때 사용 가능한지(예를 들어, 상수 또는 상수 표현식으로 지정된 경우) 또는 함수가 실행될 때만 사용 가능한지(예를 들어, 함수 호출 결과로 지정된 경우) 여부에 따라 다릅니다. 값이 함수 호출을 포함하는 SQL문이 컴파일될 때 사용 가능하면, *integer*가 0보다 클 경우 결과의 길이 속성은 *integer*입니다. *integer*가 0일 경우 결과의 길이 속성은 1입니다. 값이 함수가 실행될 때만 사용 가능하면, 결과의 길이 속성은 다음 테이블에 따라 판별됩니다.

표 49. 함수가 실행될 때만 정수가 사용 가능한 경우의 결과 길이 판별

*string-expression*의 데이터 유형	결과 데이터 유형 길이
CHAR(*n*) 또는 VARCHAR(*n*)	*n*+100 및 32 672 중 최소
GRAPHIC(*n*) 또는 VARGRAPHIC(*n*)	*n*+100 및 16 336 중 최소

결과의 실제 길이는 *integer*를 통해 판별됩니다. *integer*가 0이고 실제 길이가 0이면 결과는 비어 있는 결과 문자열입니다. *integer*가 *string-expression*의 실제 길이보다 작으면 실제 길이는 *integer*이므로 결과는 절단됩니다.

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 다음 쿼리는 값의 왼쪽을 마침표로 완전히 채웁니다.

  ```
  SELECT LPAD(NAME,15,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  ---------------
  .........Chris
  ............Meg
  ...........Jeff
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 다음 쿼리는 각 값을 5 길이가 되도록 채웁니다.

  ```
  SELECT LPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  -----
  Chris
  ..Meg
  .Jeff
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. NAME이 고정 길이 문자 필드이고 이미 공백으로 채워져 있으므로, LPAD 함수는 채우지 않습니다. 그러나 결과의 길이가 5이므로 컬럼은 절단됩니다.

  ```
  SELECT LPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  -----
  Chris
  Meg
  Jeff
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 어떤 경우에는 패드 스펙의 부분 인스턴스가 리턴됩니다.

  ```
  SELECT LPAD(NAME,15,'123' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  ---------------
  1231231231Chris
  123123123123Meg
  12312312312Jeff
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 『Chris』는 절단되고 『Meg』는 채워지며 『Jeff』는 변경되지 않습니다.

  ```
  SELECT LPAD(NAME,4,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  ----
  Chri
  .Meg
  Jeff
  ```

LTRIM

스키마는 SYSIBM입니다. (이 함수의 SYSFUN 버전은 LONG VARCHAR 및 CLOB 인수를 지원하여 계속 사용 가능합니다.)

LTRIM 함수는 *string-expression*의 시작 부분에서 공백을 제거합니다.

표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

- 인수가 DBCS 또는 EUC 데이터베이스에 있는 그래픽 문자열인 경우 앞에 있는 2 바이트 공백이 제거됩니다.
- 인수가 유니코드 데이터베이스에 있는 그래픽 문자열인 경우 앞에 있는 UCS-2 공백이 제거됩니다.
- 그렇지 않으면 앞에 있는 1바이트 공백이 제거됩니다.

함수의 결과 데이터 유형은 다음과 같습니다.

- *string-expression*의 데이터 유형이 VARCHAR 또는 CHAR이면 VARCHAR입니다.
- *string-expression*의 데이터 유형이 VARGRAPHIC 또는 GRAPHIC이면 VARGRAPHIC입니다.

리턴된 유형의 길이 매개변수는 인수 데이터 유형의 길이 매개변수와 동일합니다.

문자열에 대한 결과의 실제 길이는 *string-expression*의 길이에서 공백 문자로 제거된 바이트 수를 뺀 것입니다. 그래픽 문자열에 대한 결과의 실제 길이는 *string-expression*의 길이(2바이트 문자의 수로 표시)에서 제거된 2바이트 공백 문자의 수를 뺀 것입니다. 모든 문자가 제거되는 경우 결과는 공백의 가변 길이 문자열(길이는 0)입니다.

인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

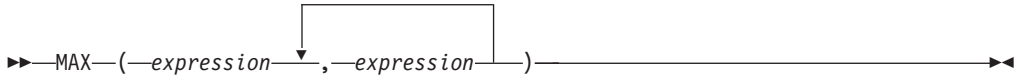
예:

호스트 변수 HELLO가 CHAR(9)로 정의되며 'Hello' 값을 갖는다고 가정하십시오.

```
VALUES LTRIM(:HELLO)
```

결과는 'Hello'입니다.

MAX



스키마는 SYSIBM입니다.

MAX 함수는 값 세트에서 최대값을 리턴합니다.

인수는 호환 가능해야 하며 각 인수는 ARRAY, LOB, LONG VARCHAR, LONG VARGRAPHIC, XML, 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 아닌 다른 데이터 유형의 값을 리턴하는 표현식이어야 합니다(SQLSTATE 42815). 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 호환 가능한 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 서명을 작성할 필요가 없습니다.

필요한 경우 선택된 인수는 결과의 속성으로 변환됩니다. 결과의 속성은 결과 데이터 유형의 규칙을 기초로 모든 피연산자에 의해 판별됩니다.

함수 결과는 가장 큰 인수 값입니다. 하나 이상의 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 어느 한 인수가 널(NULL)일 경우 결과는 널(NULL) 값이 됩니다.

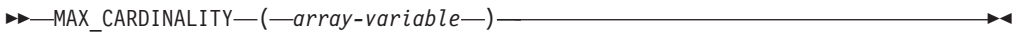
MAX 스칼라 함수는 GREATEST 스칼라 함수의 동의어입니다.

예:

사원에 대한 보너스(500과 사원 급료의 5% 중 많은 것)를 리턴합니다.

```
SELECT EMPNO, MAX(SALARY * 0.05, 500)
  FROM EMPLOYEE
```

MAX_CARDINALITY



스키마는 SYSIBM입니다.

MAX_CARDINALITY 함수는 배열이 포함할 수 있는 최대 요소 수를 표시하는 유형 BIGINT의 값을 리턴합니다. 이 함수는 일반 배열 유형에 대한 CREATE TYPE문에 지정된 카디널리티(cardinality)입니다.

array-variable
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 배열 유형의 매개변수 표시문자에 대한 CAST 스펙

결과는 널(NULL)이 될 수 있습니다. 인수가 널(NULL)이거나 연관된 배열인 경우 결과는 널(NULL) 값입니다.

예 :

- 배열 유형 PHONENUMBERS의 배열 변수 RECENT_CALLS에 대한 최대 카디널리티(cardinality)가 리턴됩니다.

```
SET LIST_SIZE = MAX_CARDINALITY(RECENT_CALLS)
```

  SQL 변수 LIST_SIZE가 50으로 설정되며 이는 배열 유형 PHONENUMBERS가 정의된 최대 카디널리티(cardinality)입니다.

MICROSECOND

스키마는 SYSIBM입니다.

MICROSECOND 함수는 값의 마이크로초 부분을 리턴합니다.

인수는 날짜, 시간소인, 시간소인 지속 기간이거나, CHAR 또는 VARCHAR 데이터 유형인 날짜 또는 시간소인의 유효한 문자열 표현이어야 합니다. 제공되는 인수가 날짜인 경우, 먼저 TIMESTAMP(0) 값으로 변환되며, 정확한 자정의 시간(00.00.00)을 가정합니다. 유니코드 데이터베이스의 경우 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

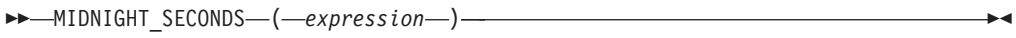
- 인수가 날짜, 시간소인, 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 정수 범위는 0 - 999 999입니다.
  - 시간소인의 정밀도가 6을 초과하면, 값이 절단됩니다.
- 인수가 지속 기간인 경우
  - 결과는 -999 999와 999 999 사이의 정수 값의 마이크로초 파트를 반영합니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- TABLEA 테이블에 유형이 TIMESTAMP인 TS1과 TS2의 두 컬럼이 있다고 가정하십시오. TS1의 마이크로초 부분이 0이 아니고 TS1과 TS2의 두 번째 부분이 동일한 모든 행을 선택하십시오.

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0
    AND
  SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS



스키마는 SYSFUN입니다.

자정(밤 12시)과 인수에서 지정된 시간 값 사이의 시간(초)을 나타내는 0 - 86 400 범위의 정수 값을 리턴합니다.

날짜, 시간, 시간소인이거나, CLOB 및 DBCLOB가 아닌 날짜, 시간, 시간소인의 유효한 문자열 표현이어야 하는 값을 리턴하는 표현식입니다.

*expression*이 날짜 또는 날짜의 유효한 문자열 표현인 경우, 정확한 자정의 시간(00.00.00)을 가정하여 TIMESTAMP(0)로 먼저 변환됩니다.

유니코드 데이터베이스만이 날짜, 시간 또는 시간소인의 그래픽 문자열 표현인 인수를 지원합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

*expression*이 문자열인 경우, 값이 날짜 시간 값으로 변환되기 전에 앞에 공백이 포함되며 뒤 공백이 제거됩니다.날짜 시간 값의 문자열 표현의 유효한 포맷에 대해서는, 『날짜 시간 값』의 『날짜 시간 값의 문자열 표현』을 참조하십시오.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

- 자정(밤 12시)에서 00:10:10과 밤 12시에서 13:10:00 사이의 초 수를 찾으십시오.

  ```
  VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
  ```

  이 예에서는 다음을 리턴합니다.

  ```
  1           2
  ----------- -----------
                  610       47410
  ```

  1분이 60초이므로 자정과 지정된 시간 사이는 610초가 됩니다. 두 번째 예에 대해서도 동일합니다. 1시간이 3600초이고 1분이 60초이므로 지정된 시간과 자정 사이는 47 410초가 됩니다.

- 자정에서 24:00:00, 자정에서 00:00:00 사이의 초 수를 찾으십시오.

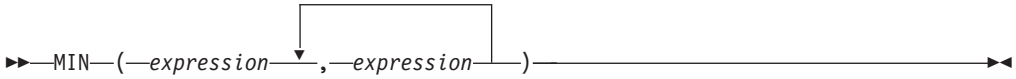
  ```
  VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
  ```

  이 예에서는 다음을 리턴합니다.

```
1           2
----------- -----------
            86400           0
```

두 값이 동일한 지점을 나타내지만 다른 MIDNIGHT_SECONDS 값을 리턴한다는 사실에 유의하십시오.

MIN



스키마는 SYSIBM입니다.

MIN 함수는 값 세트에서 최소값을 리턴합니다.

인수는 호환 가능해야 하며 각 인수는 ARRAY, LOB, LONG VARCHAR, LONG VARGRAPHIC, XML, 이 유형 중 하나에 대한 구별 유형 또는 구조화된 유형이 아닌 다른 데이터 유형의 값을 리턴하는 표현식이어야 합니다(SQLSTATE 42815). 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 호환 가능한 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 서명을 작성할 필요가 없습니다.

필요한 경우 선택된 인수는 결과의 속성으로 변환됩니다. 결과의 속성은 결과 데이터 유형의 규칙을 기초로 모든 피연산자에 의해 판별됩니다.

함수 결과는 가장 작은 인수 값입니다. 하나 이상의 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 어느 한 인수가 널(NULL)일 경우 결과는 널(NULL) 값이 됩니다.

MIN 스칼라 함수는 LEAST 스칼라 함수의 동의어입니다.

예:

사원에 대한 보너스(5000과 사원 급료의 5% 중 작은 것)를 리턴합니다.

```
SELECT EMPNO, MIN(SALARY * 0.05, 5000)
  FROM EMPLOYEE
```

MINUTE

스키마는 SYSIBM입니다.

MINUTE 함수는 값의 분 부분을 리턴합니다.

인수는 날짜, 시간, 시간소인, 시간 지속 기간, 시간소인 지속 기간 또는 CLOB가 아닌 날짜, 시간 또는 시간소인의 유효한 문자열 표현이어야 합니다. 제공되는 인수가 날짜인 경우, 먼저 TIMESTAMP(0) 값으로 변환되며, 정확한 자정의 시간(00.00.00)을 가정합니다. 유니코드 데이터베이스의 경우 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간, 시간소인이나, 날짜, 시간, 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 분 부분으로 0과 59 사이의 정수입니다.
- 인수가 시간 지속 기간이거나 시간소인 지속 기간인 경우
  - 결과는 값의 분 부분으로 -99와 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- CL_SCHED 샘플 테이블을 사용하여 지속 기간이 50분 미만인 모든 수업을 선택하십시오.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0
    AND
  MINUTE(ENDING - STARTING) < 50
```

MOD

스키마는 SYSFUN입니다.

첫 번째 인수를 두 번째 인수로 나눈 나머지를 리턴합니다. 첫 번째 인수가 음수일 경우에만 결과가 음수가 됩니다.

함수의 결과는 다음과 같습니다.

- 두 인수가 모두 SMALLINT이면 SMALLINT입니다.
- 하나의 인수가 INTEGER이고 다른 인수가 INTEGER 또는 SMALLINT이면 INTEGER입니다.
- 한 인수는 BIGINT이고 다른 인수가 BIGINT, INTEGER 또는 SMALLINT이면 BIGINT입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

MONTH

스키마는 SYSIBM입니다.

MONTH 함수는 값의 월 부분을 리턴합니다.

인수는 날짜, 시간소인, 날짜 기간, 시간소인 지속 기간 또는 CLOB가 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

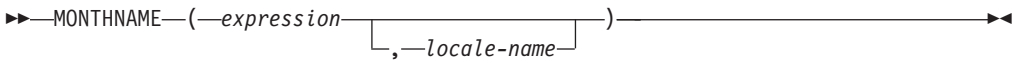
- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 월 부분으로 1과 12 사이의 정수입니다.
- 인수가 날짜 기간이거나 시간소인 지속 기간인 경우:
  - 결과는 값의 월 부분으로 1과 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- DECEMBER에 태어난(BIRTHDATE) 개인의 EMPLOYEE 테이블에서 모든 행을 선택하십시오.

```
SELECT * FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME



스키마는 SYSIBM입니다. **MONTHNAME** 함수의 SYSFUN 버전은 계속 사용 가능합니다.

MONTHNAME 함수는 *expression*의 월 부분에 대해, *locale-name* 또는 특수 레지스터 CURRENT LOCALE LC_TIME 값을 기반으로, 월의 이름(예: January)을 포함한 문자열을 리턴합니다.

expression
내장 데이터 유형인 날짜 또는 시간소인 중 하나의 값을 리턴하는 표현식입니다.

locale-name
결과의 언어를 판별하는 데 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자를 구분하지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 이름에 대해서는 ″SQL 및 XQuery에 대한 로케일 이름″을 참조하십시오. *locale-name*이 지정되지 않으면, 특수 레지스터의 값 CURRENT LOCALE LC_TIME이 사용됩니다.

결과는 가변 길이 문자열입니다. 길이 속성은 100입니다. 결과 문자열이 결과의 길이 속성을 초과하면, 결과가 절단됩니다. *expression* 인수가 널(NULL)일 수 있는 경우 결과는 널(NULL)일 수 있습니다. *expression* 인수가 널(NULL)인 경우 결과는 널(NULL) 값이 됩니다. 결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

주

- **Julian 및 Gregorian 달력**: 1582년 10월 15일에 이 함수에 의해 Julian 달력에서 Gregorian 달력으로 전이가 고려되었습니다. 그러나 MONTHNAME 함수의 SYSFUN 버전은 모든 계산에 대해 Gregorian 달력을 가정합니다.
- **결정론**: MONTHNAME은 결정 함수입니다. 그러나 *locale-name*이 명시적으로 지정되지 않으면, 함수의 호출은 특수 레지스터 CURRENT LOCALE LC_TIME의 값에 따라 다릅니다. 특수 레지스터의 값에 따라 다른 이 호출은 특수 레지스터를 사용할 수 없을 때에는 사용할 수 없습니다(SQLSTATE 42621 또는 428EC).

예 :

- 변수 TMSTAMP가 TIMESTAMP로 정의되고 다음 값을 가지고 있다고 가정합니다. 2007–03–09–14.07.38.123456. 다음 예는 함수의 여러 호출 및 결과로 발생되는 문자열 값을 나타냅니다. 각 경우의 결과 유형은 VARCHAR(100)입니다.

```
Function invocation                              Result
-------------------------                        ----------
MONTHNAME (TMSTAMP, 'CLDR 1.5:en_US')              March
MONTHNAME (TSMTAMP, 'CLDR 1.5:de_DE')              Marz
MONTHNAME (TMSTAMP, 'CLDR 1.5:fr_FR')              mars
```

MONTHS_BETWEEN

스키마는 SYSIBM입니다.

MONTHS_BETWEEN 함수는 *expression1* 및 *expression2* 사이의 월 수 추정값을 리턴합니다.

expression1 또는 *expression2*
DATE 또는 TIMESTAMP 데이터 유형 중 하나의 값을 리턴하는 표현식입니다.

*expression1*이 *expression2* 이후의 날짜를 나타내는 경우 결과는 양수입니다. *expression1*이 expression2 이전의 날짜를 나타내는 경우 결과는 음수입니다.

- *expression1* 및 *expression2*가 월의 동일한 날 날짜 또는 시간소인을 나타내거나 두 인수 모두 각 월의 마지막 날을 나타내는 경우, 결과는 연도와 월을 기초로 한 정수 차이입니다. 시간소인 인수의 시간 부분은 무시됩니다.
- 그렇지 않으면 결과의 정수 파트는 연도와 월 값을 기초로 한 차이입니다. 결과의 소수 파트는 모든 월이 31일이라는 가정을 기초로 나머지로부터 계산됩니다. 어느 한 인수가 시간소인을 나타내는 경우, 인수는 최대 정밀도를 갖는 시간소인으로 처리되며 이들 값의 시간 부분은 결과를 판별할 때 고려됩니다.

함수의 결과는 DECIMAL(31,15)입니다. 하나의 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예:

- 프로젝트 AD3100이 취할 월의 이름을 계산하십시오. 시작 날짜는 1982-01-01이고 종료 날짜는 1983-02-01인 것으로 가정하십시오.

```
SELECT MONTHS_BETWEEN(PRENDATE, PRSDATE)
    FROM PROJECT
   WHERE PROJNO='AD3100'
```

  결과는 13.000000000000000입니다.

다음은 고려할 추가 예입니다.

표 50. MONTHS_BETWEEN을 사용하는 추가적인 예

인수 *e1*의 값	인수 *e2*의 값	다음에 의해 리턴된 값 **MONTHS_BETWEEN** (*e1*,*e2*)	다음에 의해 리턴된 값 **ROUND** (**MONTHS_BETWEEN** (*e1*,*e2*)***31,2**)	설명
2005-02-02	2005-01-01	1.032258064516129	32.00	

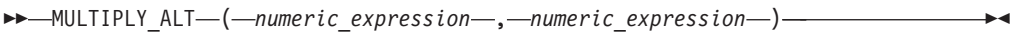
표 50. MONTHS_BETWEEN을 사용하는 추가적인 예 (계속)

인수 *e1*의 값	인수 *e2*의 값	다음에 의해 리턴된 값 **MONTHS_BETWEEN** (*e1*,*e2*)	다음에 의해 리턴된 값 **ROUND** (**MONTHS_BETWEEN** (*e1*,*e2*)***31,2**)	설명
2007-11-01-09.00.00.00000	2007-12-07-14.30.12.12345	-1.200945386592741	-37.23	
2007-12-13-09.40.30.00000	2007-11-13-08.40.30.00000	1.000000000000000	31.00	메모 1 참조
2007–03–15	2007–02–20	0.838709677419354	26.00	메모 2 참조
2008-02-29	2008-02-28-12.00.00	0.016129032258064	0.50	
2008-03-29	2008-02-29	1.000000000000000	31.00	
2008-03-30	2008-02-29	1.032258064516129	32.00	
2008-03-31	2008-02-29	1.000000000000000	31.00	메모 3 참조

주:

1. 시간 차이는 무시됩니다. 당월의 날이 두 값 모두에 대해 동일하기 때문입니다.
2. 2월이 28일이라 하더라도 모든 달에는 31일이 있다고 가정하므로 결과는 23이 아닙니다.
3. 두 날짜가 모두 해당 월의 마지막 일이므로 결과는 33이 아니며 따라서 결과는 연도 및 월 분할에만 기반합니다.

MULTIPLY_ALT



스키마는 SYSIBM입니다.

MULTIPLY_ALT 스칼라 함수는 두 인수의 곱을 리턴합니다. 특히 인수의 10진수 정밀도가 31을 초과하는 경우 곱셈 연산자에 대한 대안으로 제공됩니다.

인수는 내장 숫자 데이터 유형일 수 있습니다.

두 인수가 정확히 숫자 데이터 유형(DECIMAL, BIGINT, INTEGER, 또는 SMALLINT)인 경우 함수 결과는 DECIMAL입니다. 그렇지 않으면 연산은 10진수 부동 소수점 산술을 사용하여 수행되며 함수 결과는 정밀도가 10진수 부동 소수점 산술에 대해 결정되는 것과 같은 동일한 방식으로 인수의 데이터 유형에 의해 결정된 정밀도를 포함한 10진수 부동 소수점입니다. 함수를 평가하기 전에 부동 소수점이나 문자열 인수는 DECFLOAT(34)로 캐스트됩니다.

함수의 결과가 DECIMAL인 경우, 결과의 정밀도와 스케일은 다음과 같이 기호 *p* 및 *s*를 사용하여 결정되며 첫 번째 인수의 정밀도와 스케일을 표시하며 기호 *p*' 및 *s*'를 사용하여 두 번째 인수의 정밀도와 스케일을 표시합니다.

- 정밀도는 MIN(31, *p* + *p*')입니다.
- 스케일은 다음과 같습니다.
  - 두 인수의 스케일이 0인 경우 0입니다.
  - *p* + *p*'가 31 이하인 경우 MIN(31, *s* + *s*')입니다.
  - *p* + *p*'가 31보다 큰 경우 MAX(MIN(3, *s* + *s*'), 31 - (*p* - *s* + *p*' - *s*'))입니다.

적어도 하나의 인수가 널(NULL)이거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 인수 중 하나가 널(NULL)인 경우 결과는 널(NULL) 값이 될 수 있습니다.

적어도 3의 스케일이 필요하고 정밀도 합이 31을 초과하는 곳에서 10진수 산술을 수행하는 경우 곱셈 연산자에 대해 MULTIPLY_ALT 함수를 선택하는 것이 좋습니다. 이 경우 오버플로우를 피하도록 내부 계산이 수행됩니다. 최종 결과는 스케일과 일치해야 하는 경우 절단을 사용하여 결과 데이터 유형에 지정됩니다. 스케일이 3인 경우 최종 결과의 오버플로우가 여전히 가능하다는 것을 참고하십시오.

다음은 MULTIPLY_ALT 및 곱셈 연산자를 사용하여 결과 유형을 비교하는 샘플입니다.

인수 유형 1	인수 유형 2	MULTIPLY_ALT를 사용한 결과	곱셈 연산자를 사용한 결과
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

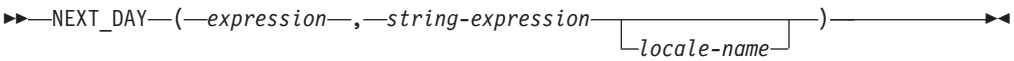
예:

첫 번째 인수의 데이터 유형이 DECIMAL(26,3)이고 두 번째 인수의 데이터 유형이 DECIMAL(9,8)인 경우 두 값을 곱합니다. 결과의 데이터 유형은 DECIMAL(31,7)입니다.

```
values multiply_alt(98765432109876543210987.654,5.43210987)
1
--------------------------------
536504678578875294857887.5277415
```

이들 두 숫자의 전체 결과는 536504678578875294857887.52774154498이지만, 마지막 4자리는 결과 데이터 유형의 스케일과 일치하도록 절단됩니다. 같은 값의 곱셈 연산자를 사용하면 산술 오버플로우가 발생되며, 결과 데이터 유형이 DECIMAL(31,11)이고 결과 값이 소수점 왼쪽으로 24자리이지만, 결과 데이터 유형은 20자리만을 지원합니다.

NEXT_DAY



스키마는 SYSIBM입니다.

NEXT_DAY 스칼라 함수는 *string-expression*이라고 하는, *expression*의 날짜보다 이후인 첫 번째 요일을 나타내는 날짜 시간 값을 리턴합니다.

expression
내장 데이터 유형인 날짜 또는 시간소인 중 하나의 값을 리턴하는 표현식입니다.

string-expression
내장 문자 데이터 유형을 리턴하는 표현식입니다. 값은 *locale-name*에 유효한 요일이어야 합니다. 값은 요일의 전체 이름이나 연관된 약어로 지정할 수 있습니다. 예를 들어, 로케일이 'en_US'인 경우 다음 값이 유효합니다.

표 51. 'en_US' 로케일의 유효한 요일 및 약어

요일	약어
MONDAY	MON
TUESDAY	TUE
WEDNESDAY	WED
THURSDAY	THU
FRIDAY	FRI
SATURDAY	SAT
SUNDAY	SUN

입력 값의 최소 길이는 약어의 길이입니다. 문자는 소문자나 대문자로 지정할 수 있으며 유효한 약어 바로 다음에 있는 문자는 무시됩니다.

locale-name
string-expression 값의 언어를 판별하기 위해 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자가 구분되지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 해당되는 이름 지정 방법에 대한 정보는 『SQL 및 XQuery의 로케일 이름』을 참조하십시오. *locale-name*을 지정하지 않은 경우 특수 레지스터 CURRENT LOCALE LC_TIME의 값이 사용됩니다.

함수의 결과는 결과 데이터 유형이 TIMESTAMP(6)인 경우 *expression*이 문자열이 아니면 *expression*과 같은 데이터 유형을 갖습니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

*expression*에 포함된 시, 분, 초 또는 소수 초 정보는 함수에서 변경되지 않습니다. *expression*이 날짜를 나타내는 문자열인 경우 결과 TIMESTAMP 값의 시간 정보는 모두 영(0)으로 설정됩니다.

주

- **결정론:** NEXT_DAY는 결정 함수입니다. 그러나 locale-name이 명시적으로 지정되지 않을 때 함수 호출은 특수 레지스터 CURRENT LOCALE LC_TIME의 값에 따라 다릅니다. 특수 레지스터의 값에 종속되는 함수의 호출은 특수 레지스터를 사용할 수 없는 경우 사용할 수 없습니다.

예:

- 변수 NEXTDAY를 2007년 4월 24일 다음의 화요일 날짜로 설정하십시오.

```
NEXTDAY = NEXT_DAY(DATE '2007-04-24', 'TUESDAY');
```

  변수 NEXTDAY는 '2007–05–01–00.00.00.000000' 값으로 설정됩니다. 2007년 4월 24일 자체가 화요일이기 때문입니다.

- 변수 vNEXTDAY를 2007년 5월 첫 번째 월요일 날짜로 설정하십시오. 변수 vDAYOFWEEK = 'MON'을 가정하십시오.

```
vNEXTDAY = NEXT_DAY(LAST_DAY(CURRENT_TIMESTAMP),vDAYOFWEEK);
```

  변수 vNEXTDAY는 '2007-05-07-12.01.01.123456' 값으로 설정됩니다. 이때 CURRENT_TIMESTAMP 특수 레지스터의 값이 '2007-04-24-12.01.01.123456'인 것으로 가정합니다.

NORMALIZE_ DECFLOAT

스키마는 SYSIBM입니다.

NORMALIZE_DECFLOAT 함수는 가장 간단한 양식으로 설정된 입력 인수와 동일한 10진 부동 소수점 값을 리턴합니다. 즉, 계수에서 뒤에 0이 있는 0이 아닌 숫자에서 뒤에 있는 0이 제거됩니다. 이때 계수를 적절한 10제곱으로 나누고 해당되는 지수를 조정한 표준 양식으로 숫자를 표시해야 합니다. 영(0) 값의 지수는 0으로 설정됩니다.

expression

임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 유형 SMALLINT, INTEGER, REAL, DOUBLE 또는 DECIMAL(*p*,*s*)의 인수(여기서 $p <= 16$)는 처리를 위해 DECFLOAT(16)로 변환됩니다. 유형 BIGINT 또는 DECIMAL(*p*,*s*)의 인수(여기서 $p > 16$)는 처리를 위해 DECFLOAT(34)로 변환됩니다.

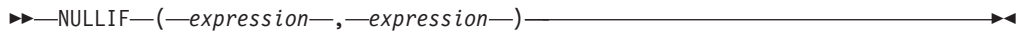
함수의 결과는 10진 부동 소수점으로 변환 후 표현식의 데이터 유형이 DECFLOAT(16)인 경우 DECFLOAT(16) 값입니다. 그렇지 않으면 함수 결과는 DECFLOAT(34) 값입니다. 인수가 특수한 10진 부동 소수점 값인 경우 결과는 동일한 특수 10진 부동 소수점 값입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

- 다음 예는 다양한 10진 부동 소수점 값이 제공되는 경우 NORMALIZE_DECFLOAT 함수가 리턴하는 값을 나타냅니다.

```
NORMALIZE_DECFLOAT(DECFLOAT(2.1)) = 2.1
NORMALIZE_DECFLOAT(DECFLOAT(-2.0)) = -2
NORMALIZE_DECFLOAT(DECFLOAT(1.200)) = 1.2
NORMALIZE_DECFLOAT(DECFLOAT(-120)) = -1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(120.00)) = 1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(0.00)) = 0
NORMALIZE_DECFLOAT(-NAN) = -NaN
NORMALIZE_DECFLOAT(-INFINITY) = -Infinity
```

NULLIF



스키마는 SYSIBM입니다.

NULLIF 함수는 인수가 같을 경우에는 널(NULL) 값을 리턴하고, 그 외에는 첫 번째 인수 값을 리턴합니다.

인수는 비교 가능해야 합니다. 인수는 내장(LOB 문자열 제외한) 또는 구별 데이터 유형(LOB 문자열을 기초한 것을 제외한)일 수 있습니다. 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 또한 이 함수는 인수로 호환 가능한 데이터 유형을 허용하므로 사용자 정의 구별 유형을 지원하는 추가 시그니처를 작성할 필요가 없습니다. 결과의 속성은 첫 번째 인수의 속성입니다.

NULLIF(e1,e2)를 사용한 결과는 다음 표현식을 사용한 결과와 동일합니다.

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

e1=e2가 알 수 없음으로 평가되면(하나 또는 두 인수가 널(NULL)이므로) CASE 표현식은 이것을 참이 아닌 것으로 간주합니다. 그러므로 이 상황에서 NULLIF는 첫 번째 인수의 값을 리턴합니다.

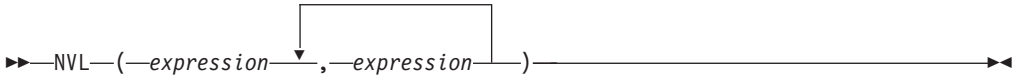
예:

- 호스트 변수 PROFIT, CASH 및 LOSSES의 데이터 유형이 DECIMAL이고 값이 각각 4500.00, 500.00, 5000.00이라고 가정하십시오.

  ```
  NULLIF (:PROFIT + :CASH , :LOSSES )
  ```

  널(NULL) 값을 리턴합니다.

NVL

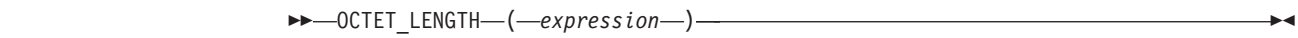


스키마는 SYSIBM입니다.

NVL 함수는 널(NULL)이 아닌 첫 번째 인수를 리턴합니다.

NVL은 COALESCE의 동의어입니다.

OCTET_LENGTH



스키마는 SYSIBM입니다.

OCTET_LENGTH 함수는 *expression*의 길이를 OCTETS(바이트) 단위로 리턴합니다.

expression
내장 string 데이터 유형의 값을 리턴하는 표현식입니다.

함수 결과는 INTEGER입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

문자열 또는 그래픽 문자열의 길이는 뒤 공백을 포함합니다. 실행 파일 문자열의 길이는 2진수 값 0을 포함합니다. 가변 길이 문자열의 실이는 실제 길이이며 최대 길이가 아닙니다.

더 큰 이식성을 위해 데이터 유형 DECIMAL(31)의 결과를 승인할 수 있도록 응용프로그램을 코딩하십시오.

예를 들면, 다음과 같습니다.

- 테이블 T1에 C1이라는 GRAPHIC(10) 컬럼이 있다고 가정하십시오.

  ```
  SELECT OCTET_LENGTH(C1) FROM T1
  ```

  은 값 20을 리턴합니다.
- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

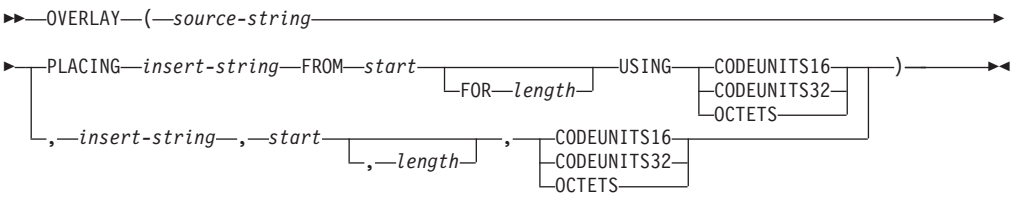
	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  변수 UTF8_VAR 및 UTF16_VAR에 각각 해당 문자열의 UTF-8 및 UTF-16BE 표시가 들어 있다고 가정하십시오.

  ```
  SELECT OCTET_LENGTH(UTF8_VAR),
    OCTET_LENGTH(UTF16_VAR)
  FROM SYSIBM.SYSDUMMY1
  ```

  리턴되는 값은 각각 9와 12입니다.

OVERLAY



스키마는 SYSIBM입니다.

OVERLAY 함수는 *source-string*의 *start*에서 시작하고 지정된 코드 단위의 *length*가 삭제되었으며 *insert-string*이 삽입된 문자열을 리턴합니다.

source-string

소스 문자열을 지정하는 표현식입니다. 표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

insert-string

*start*에서 식별된 위치에서 시작하여 *source-string*에 삽입될 문자열을 지정하는 표현식입니다. 이 표현식은 내장 문자열, 숫자 또는 날짜 및 시간 데이터 유형의 값을 리턴합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *insert-string* 코드 페이지가 *source-string*의 코드 페이지와 다른 경우 *insert-string*은 *source-string*의 코드 페이지로 변환됩니다.

start

정수값을 리턴하는 표현식입니다. 정수값은 소스 문자열 내에서 바이트 삭제 및 또 다른 문자열 삽입이 시작될 시작 위치를 지정합니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. 정수값은 1 및 *source-string* + 1의 길이 사이여야 합니다(SQLSTATE 42815). OCTETS가 지정되고 결과가 그래픽 데이터인 경우 값은 1과 *source-string*의 길이 속성 + 1의 두 배 사이의 홀수여야 합니다(SQLSTATE 428GC).

length

*start*에서 식별된 위치에서 시작하여 소스 문자열에서 삭제될 코드 단위(지정된 문자열 단위의) 수를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. 정수값은 0 및 *source-string* 길이 사이여야 하며, 내재적 또는 명시적으로 지정된 단위로 표시됩니다(SQLSTATE 22011). OCTETS

가 지정되고 결과가 그래픽 데이터인 경우 값은 0과 *source-string* 길이 속성의 두 배 사이의 짝수여야 합니다(SQLSTATE 428GC).

*length*를 지정하지 않는 것은 OCTETS를 지정하고 결과가 그래픽 데이터인 경우를 제외하고 1 값을 지정하는 것과 같습니다. 이 경우 *length*를 지정하지 않는 것은 2 값을 지정하는 것과 같습니다.

CODEUNITS16, CODEUNITS32 또는 OCTETS

start 및 *length*의 문자열 단위를 지정합니다.

CODEUNITS16은 *start* 및 *length*가 16비트 UTF-16 코드 단위로 표시됨을 지정합니다. CODEUNITS32는 *start* 및 *length*가 32비트 UTF-32 코드 단위로 표시됨을 지정합니다. OCTETS는 *start* 및 *length*가 바이트 단위로 표시됨을 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 결과가 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *insert-string* 및 *source-string*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815). 문자열 단위가 OCTETS로 지정된 경우 해당 조작은 *source-string*의 코드 페이지에서 수행됩니다. CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 『문자열』에서 『내장 함수의 문자열 단위』를 참조하십시오.

결과의 데이터 유형은 지원 유형 조합에 대한 다음 표에 표시된 대로 *source-string* 및 *insert-string*의 데이터 유형에 따라 결정됩니다.

표 52. source-string 및 insert-string 데이터 유형의 기능으로서의 결과 데이터 유형

source-string	*insert-string*	결과
CHAR 또는 VARCHAR	CHAR 또는 VARCHAR	VARCHAR
GRAPHIC 또는 VARGRAPHIC	GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC
CLOB	CHAR, VARCHAR 또는 CLOB	CLOB
DBCLOB	GRAPHIC, VARGRAPHIC 또는 DBCLOB	DBCLOB
CHAR 또는 VARCHAR	CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA	CHAR, VARCHAR, CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
BLOB	BLOB	BLOB
유니코드 데이터베이스의 경우:		
CHAR 또는 VARCHAR	GRAPHIC 또는 VARGRAPHIC	VARCHAR
GRAPHIC 또는 VARGRAPHIC	CHAR 또는 VARCHAR	VARGRAPHIC

표 52. source-string 및 insert-string 데이터 유형의 기능으로서의 결과 데이터 유형 (계속)

source-string	*insert-string*	결과
CLOB	GRAPHIC, VARGRAPHIC 또는 DBCLOB	CLOB
DBCLOB	CHAR, VARCHAR 또는 CLOB	DBCLOB

*source-string*의 길이는 0이 될 수 있습니다. 이 경우 *start*는 1이어야 하고 *length*는 0이어야 하며(위에서 설명한 *start* 및 *length*의 바운드로 암시된 것처럼) 함수 결과는 *insert-string*의 사본입니다. 이 경우에 길이가 명시적으로 지정되지 않은 경우 가정된 길이가 0이 아니므로 오류가 리턴됩니다(SQLSTATE 22011).

*insert-string*의 길이도 0이 될 수 있습니다. 이는 *source-string*에서 위치 *start*로부터 *start* + *length* - 1까지 코드 단위를 삭제하는 효과를 갖습니다.

결과의 길이 속성은 *source-string*의 길이 속성 + *insert-string*의 길이 속성입니다. 결과의 실제 길이는 A1 - MIN((A1 - V2 + 1), V3) + A4입니다. 여기서,

- A1은 *source-string*의 실제 길이입니다.
- V2는 *start*의 값입니다.
- V3는 *length*의 값입니다.
- A4는 *insert-string*의 실제 길이입니다.

결과 문자열의 실제 길이가 리턴 데이터 유형에 대한 최대값을 초과할 경우 오류가 리턴됩니다(SQLSTATE 54006).

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- 기존 텍스트 중간에 텍스트를 삽입하여 문자열 'INSERTING'으로부터 'INSISTING', 'INSISERTING' 및 'INSTING' 문자열을 작성하십시오.

```
SELECT OVERLAY('INSERTING','IS',4,2,OCTETS),
  OVERLAY('INSERTING','IS',4,0,OCTETS),
  OVERLAY('INSERTING','',4,2,OCTETS)
  FROM SYSIBM.SYSDUMMY1
```

- 시작점으로 1을 사용하여 기존 텍스트 앞에 텍스트를 삽입하여 문자열 'INSERTING'으로부터 'XXINSERTING', 'XXNSERTING', 'XXSERTING' 및 'XXERTING' 문자열을 작성하십시오.

```
SELECT OVERLAY('INSERTING','XX',1,0,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,1,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,2,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,3,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- 기존 텍스트 다음에 텍스트를 삽입하여 'ABCABC' 문자열에서 'ABCABCXX' 문자열을 작성하십시오. 소스 문자열은 길이가 6자이므로, 시작 위치를 7(1 + 소스 문자열의 길이)로 설정하십시오.

```
SELECT OVERLAY('ABCABC','XX',7,0,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- 문자열 'Hegelstraße'를 'Hegelstrasse'로 변경하십시오.

```
SELECT OVERLAY('Hegelstraße','ss',10,1,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

변수 UTF8_VAR 및 UTF16_VAR에 각각 해당 문자열의 UTF-8 및 UTF-16BE 표시가 들어 있다고 가정하십시오. OVERLAY 함수를 사용하여 'C'를 유니코드 문자열 '&N~AB'에 삽입하십시오.

```
SELECT OVERLAY(UTF8_VAR, 'C', 1, CODEUNITS16),
  OVERLAY(UTF8_VAR, 'C', 1, CODEUNITS32),
  OVERLAY(UTF8_VAR, 'C', 1, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

각각 값 'C?N~AB', 'CN~AB' 및 'CbbbN~AB'를 리턴합니다. 여기서 '?'는 중간 UTF-16 양식의 X'DD1E'에 해당되는 X'EDB49E'를 표시하고 'bbb'는 UTF-8 불완전 문자 X'9D849E'를 교체합니다.

```
SELECT OVERLAY(UTF8_VAR, 'C', 5, CODEUNITS16),
  OVERLAY(UTF8_VAR, 'C', 5, CODEUNITS32),
  OVERLAY(UTF8_VAR, 'C', 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

각각 값 '&N~CB', '&N~AC' 및 '&N~AB'를 리턴합니다.

```
SELECT OVERLAY(UTF16_VAR, 'C', 1, CODEUNITS16),
  OVERLAY(UTF16_VAR, 'C', 1, CODEUNITS32)
FROM SYSIBM.SYSDUMMY1
```

각각 값 'C?N~AB' 및 'CN~AB'를 리턴합니다. 여기서 '?'는 일치되지 않는 낮은 대리 U+DD1E를 표시합니다.

```
SELECT OVERLAY(UTF16_VAR, 'C', 5, CODEUNITS16),
  OVERLAY(UTF16_VAR, 'C', 5, CODEUNITS32)
FROM SYSIBM.SYSDUMMY1
```

각각 값 '&N~CB', '&N~AC'를 리턴합니다.

PARAMETER

PARAMETER 함수는 값이 XQuery에 의해 db2-fn:sqlquery 함수 일부로 동적으로 제공되는 SQL문 내의 위치를 표시합니다.

스키마는 SYSIBM입니다.

*integer-constant*는 db2-fn:sqlquery의 인수에서 값에 대한 위치 인덱스입니다. 값은 1과 db2-fn:sqlquery SQL문에 지정된 총 인수 개수 사이여야 합니다(SQLSTATE 42815).

PARAMETER 함수는 값이 XQuery에 의해 db2-fn:sqlquery 함수 일부로 동적으로 제공되는 SQL문 내의 위치를 표시합니다. PARAMETER 함수의 인수는 db2-fn:sqlquery 함수가 실행될 때 PARAMETER 함수에 대해 대체되는 값을 판별합니다. PARAMETER 함수에서 제공된 값은 동일한 SQL문에서 여러 번 참조될 수 있습니다.

이 함수는 XQuery 표현식에서 db2-fn:sqlquery 함수의 문자열 리터럴 인수에 포함되는 fullselect에만 사용할 수 있습니다(SQLSTATE 42887).

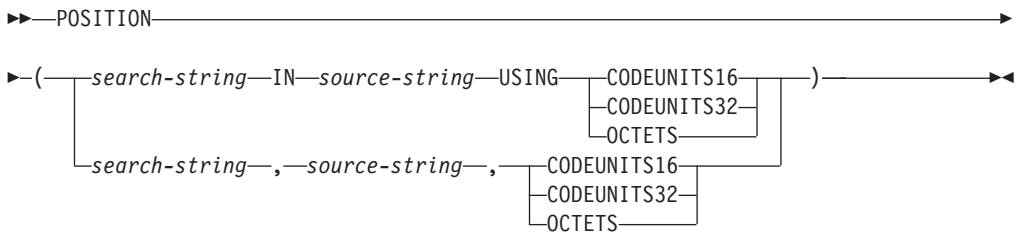
예:

다음 예에서, db2-fn:sqlquery 함수 호출은 하나의 PARAMETER 함수 호출과 XQuery 표현식 `$po/@OrderDate`(주문 날짜 속성)를 사용합니다. PARAMETER 함수는 주문 날짜 속성의 값을 리턴합니다.

```
xquery
declare default element namespace "http://posample.org";
for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')/PurchaseOrder,
  $item in $po/item/partid
for $p in db2-fn:sqlquery(
  "select description from product where promostart < PARAMETER(1)",
    $po/@OrderDate )
where $p//@pid = $item
return
<RESULT>
  <PoNum>{data($po/@PoNum)}</PoNum>
  <PartID>{data($item)} </PartID>
  <PoDate>{data($po/@OrderDate)}</PoDate>
</RESULT>
```

예는 판촉 시작 날짜 이후에 팔린 모든 파트의 구입 ID, 파트 ID 및 구입 날짜를 리턴합니다.

POSITION



스키마는 SYSIBM입니다.

POSITION 함수는 한 문자열(*source-string*) 내에서 다른 문자열(*search-string*)의 첫 번째 시작 위치를 리턴합니다. *search-string*이 발견되지 않고 인수가 널(NULL)이 아니면 결과는 0이 됩니다. *search-string*을 찾은 경우 결과는 1에서 *source-string*의 실제 길이까지의 숫자이며 명시적으로 지정된 문자열 단위로 표시됩니다. *search-string* 또는 *source-string*이 FOR BIT DATA로 정의되지 않으면(이 경우 검색은 2진 비교를 사용하여 수행됨), 검색은 데이터베이스 조합을 사용하여 수행됩니다.

*source-string*의 실제 길이가 0이면 함수 결과는 0입니다. *search-string*의 실제 길이가 0이고 *source-string*이 널(NULL)이 아닐 경우 함수의 결과는 1입니다.

search-string

검색의 오브젝트인 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BLOB, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC 또는 BLOB 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식은 BLOB 파일 참조 변수가 될 수 없습니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 항목을 병합(CONCAT 또는 ||를 사용하여)하는 표현식
- SQL 프로시저 매개변수

이들 규칙은 LIKE 술어에 대한 *pattern-expression*에 대해 설명된 규칙과 유시합니다.

source-string

이 표현식은 내장 문자열, 숫자 또는 날짜 및 시간 데이터 유형의 값을 리턴합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수(로케이터 변수 또는 파일 참조 변수 포함)
- 스칼라 함수
- 대형 오브젝트(LOB) 로케이터
- 컬럼 이름
- 위의 항목을 병합(CONCAT 또는 ||를 사용하여)하는 표현식

CODEUNITS16, CODEUNITS32 또는 OCTETS

결과의 문자열 단위를 지정합니다. CODEUNITS16은 결과가 16비트 UTF-16 코드 단위로 표시될 것을 지정합니다. CODEUNITS32는 결과가 32비트 UTF-32 코드 단위로 표시될 것을 지정합니다. OCTETS는 결과가 바이트 단위로 표시될 것을 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *search-string* 또는 *source-string*이 실행 파일 문자열 또는 비트 데이터이면, 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *search-string* 및 *source-string*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815).

로케일이 구분되는 UCA 기반 조합이 이 함수에 사용되는 경우, CODEUNITS16 옵션은 최상의 성능 특성을 제공합니다.

CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는, 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

첫 번째와 두 번째 인수는 호환 가능한 문자열 유형을 가져야 합니다. 호환성에 대한 자세한 정보는 『문자열 변환에 대한 규칙』을 참조하십시오. 유니코드 데이터베이스에서, 하나의 문자열 인수가 문자이고(FOR BIT DATA가 아니라) 다른 문자열 인수가 그래픽인 경우 *search-string*은 처리를 위해 *source-string* 데이터 유형으로 변환됩니다. 하나의 인수가 문자 FOR BIT DATA인 경우, 다른 인수는 그래픽이 될 수 없습니다(SQLSTATE 42846).

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- IN_TRAY 테이블에서 ″GOOD BEER’ 문자열이 들어 있는 모든 행에 대한 RECEIVED 컬럼, SUBJECT 컬럼 및 NOTE_TEXT 컬럼에서 ’GOOD BEER’ 문자열의 시작 위치를 선택하십시오.

```
SELECT RECEIVED, SUBJECT, POSITION('GOOD BEER', NOTE_TEXT, OCTETS)
FROM IN_TRAY
WHERE POSITION('GOOD BEER', NOTE_TEXT, OCTETS) <> 0
```

- 'Jürgen lives on Hegelstraße' 문자열에서 'ß' 문자의 위치를 찾은 후 이 위치를 사용하여 호스트 변수 LOCATION을 문자열에서 CODEUNITS32 단위로 설정하십시오.

```
SET :LOCATION = POSITION(
  'ß', 'Jürgen lives on Hegelstraße', CODEUNITS32
)
```

  호스트 변수 LOCATION의 값은 26으로 설정됩니다.

- 'Jürgen lives on Hegelstraße' 문자열에서 'ß' 문자의 위치를 찾은 후 이 위치를 사용하여 호스트 변수 LOCATION을 문자열에서 OCTETS 단위로 설정하십시오.

```
SET :LOCATION = POSITION(
  'ß', 'Jürgen lives on Hegelstraße', OCTETS
)
```

  호스트 변수 LOCATION 값은 27로 설정됩니다.

- 다음 예는 '&'가 음악 기호 높은음자리표이고 '~'가 공백 없는 조합 틸드 문자인 유니코드 문자열 '&N~AB'에 대해 작동합니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  변수 UTF8_VAR에 해당 문자열의 UTF-8 표시가 들어 있다고 가정하십시오.

```
SELECT POSITION('N', UTF8_VAR, CODEUNITS16),
  POSITION('N', UTF8_VAR, CODEUNITS32),
  POSITION('N', UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  리턴되는 값은 각각 3, 2, 5입니다.

  변수 UTF16_VAR에 해당 문자열의 UTF-16BE 표시가 들어 있다고 가정하십시오.

```
SELECT POSITION('B', UTF16_VAR, CODEUNITS16),
   POSITION('B', UTF16_VAR, CODEUNITS32),
   POSITION('B', UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  리턴되는 값은 각각 6, 5, 11입니다.

- 대소문자가 구분되지 않는 조합 UCA500R1_LEN_S1으로 작성된 유니코드 데이터베이스의 'The quick brown fox' 구문에서 'Brown' 단어의 위치를 찾으십시오.

```
SET :LOCATION = POSITION('Brown', 'The quick brown fox', CODEUNITS16)
```

  호스트 변수 LOCATION의 값은 11로 설정됩니다.

POSSTR

스키마는 SYSIBM입니다.

POSSTR 함수는 한 문자열(*source-string*) 내에서 다른 문자열(*search-string*)의 첫 번째 시작 위치를 리턴합니다. *search-string* 위치에 대한 숫자는 1(0이 아님)에서 시작합니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

source-string

검색이 수행될 소스 문자열을 지정하는 표현식입니다.

표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수(로케이터 변수 또는 파일 참조 변수 포함)
- 스칼라 함수
- 대형 오브젝트(LOB) 로케이터
- 컬럼 이름
- 위의 항목을 병합(CONCAT 또는 ||를 사용하여)하는 표현식

search-string

검색될 문자열을 지정하는 표현식입니다.

표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BLOB, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC 또는 BLOB 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 실제 길이는 VARCHAR의 최대 길이를 초과해서는 안됩니다. 표현식은 BLOB 파일 참조 변수가 될 수 없습니다. 표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 전역 변수
- 호스트 변수

- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 항목을 병합(CONCAT 또는 ||를 사용하여)하는 표현식
- SQL 프로시저 매개변수

다음은 올바르지 않은 문자열 표현식 또는 문자열의 예입니다.

- SQL 사용자 정의 함수 매개변수
- 트리거 전이 변수
- 동적 복합문의 로컬 변수

유니코드 데이터베이스에서 하나의 인수가 문자이고(FOR BIT DATA가 아닌) 다른 인수가 그래픽인 경우, *search-string*이 *source-string*의 데이터 유형으로 변환되어 처리됩니다. 하나의 인수가 문자 FOR BIT DATA인 경우, 다른 인수는 그래픽이어서는 안됩니다(SQLSTATE 42846).

*search-string*과 *source-string*은 모두 0 이상의 연속적 위치를 갖습니다. 문자열이 문자이거나 실행 파일 문자열인 경우 위치는 1바이트입니다. 문자열이 그래픽 문자열인 경우 위치는 그래픽(DBCS) 문자입니다.

POSSTR 함수는 혼합 데이터 문자열을 허용합니다. 그러나 POSSTR은 1바이트와 복수 바이트 문자 간의 변경사항과, 데이터베이스 조합에 관계없이 엄격한 바이트 수를 기준으로 작동됩니다.

다음 규칙이 적용됩니다.

- *source-string*과 *search-string*의 데이터 유형은 호환성이 있어야 하며 그렇지 않은 경우에는 오류가 발생합니다(SQLSTATE 42884).
  - *source-string*이 문자열인 경우, *search-string*은 CLOB 또는 LONG VARCHAR이 아닌 문자열이어야 하며 실제 길이는 32 672바이트 이하이어야 합니다.
  - *source-string*이 그래픽 문자열인 경우, *search-string*은 DBCLOB 또는 LONG VARGRAPHIC가 아닌 그래픽 문자열이어야 하며 실제 길이는 16 336바이트 이하이어야 합니다.
  - *source-string*이 실행 파일 문자열인 경우, *search-string*은 실제 길이가 32 672 바이트 이하인 실행 파일 문자열이어야 합니다.
- *search-string*의 길이가 0인 경우, 함수에서 리턴되는 결과는 1입니다.
- 그렇지 않을 경우 다음과 같습니다.
  - *source-string*의 길이가 0이면 함수에서 리턴되는 길이는 0입니다.
  - 그렇지 않을 경우 다음과 같습니다.
    - *search-string*의 값이 *source-string*의 값에서 연속적 위치의 동일한 길이의 부속 문자열과 같은 경우, 함수에서 리턴되는 결과는 *source-string* 값 내에서 이러한 문자열의 첫 번째 시작 위치입니다.

- 그렇지 않을 경우, 함수에서 리턴되는 결과는 0입니다.

예 :

- ‘GOOD BEER’ 단어를 포함하는 IN_TRAY 테이블에 있는 모든 항목에 대해 NOTE_TEXT 컬럼 내에서 그 단어들의 시작 위치뿐만 아니라 RECEIVED 및 SUBJECT 컬럼을 선택하십시오.

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER

스키마는 SYSIBM입니다. (POWER 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

POWER 함수는 첫 번째 인수를 두 번째 인수로 거듭제곱한 결과를 리턴합니다.

인수는 내장 데이터 유형이면 어느 것이나 될 수 있습니다. DECIMAL 및 REAL 인수는 배정밀도 부동 소수점 숫자로 변환됩니다. 인수가 10진수 부동 소수점이면 인수는 DECFLOAT(34)로 변환되어 함수에서 처리됩니다.

함수의 결과는 다음과 같습니다.

- 인수가 INTEGER 또는 SMALLINT이면 INTEGER입니다.
- 한 인수는 BIGINT이고 다른 인수가 BIGINT, INTEGER 또는 SMALLINT일 경우, BIGINT입니다.
- 인수 중 하나가 10진수 부동 소수점인 경우 DECFLOAT(34). 인수가 DECFLOAT이고 다음 명령문 중 하나가 참이면, 결과는 NAN 및 유효하지 않은 연산 조건입니다.
  - 두 인수 모두 0인 경우
  - 두 번째 인수는 0이 아닌 소수 파트입니다.
  - 두 번째 인수는 9개 이상의 숫자여야 합니다.
  - 두 번째 인수는 INFINITY입니다.
- 그 외의 경우 DOUBLE

인수가 특수 10진수 부동 소수점 값인 경우 10진수 부동 소수점에 대해 일반 산술 연산의 규칙이 적용됩니다. 243 페이지의 『10진수 부동 소수점에 대한 일반 산술 연산 규칙』에서 『10진수 부동 소수점에 대한 일반 산술 연산 규칙』을 참조하십시오.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

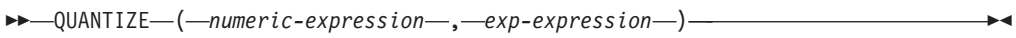
예:

- 호스트 변수 HPOWER이 값이 3인 정수임을 가정하십시오.

```
VALUES POWER(2,:HPOWER)
```

  값 8을 리턴합니다.

QUANTIZE



스키마는 SYSIBM입니다.

QUANTIZE 함수는 값(근사값의 경우 제외)과 부호에서 *numeric-expression*과 동일하고 지수가 *exp-expression*의 지수와 같은 10진 부동 소수점 값을 리턴합니다. 자릿수(16 또는 34)는 *numeric-expression*의 자릿수와 같습니다.

numeric-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 10진 부동 소수점 값이 아니면 처리를 위해 DECFLOAT(34)로 변환됩니다.

exp-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 10진 부동 소수점 값이 아니면 처리를 위해 DECFLOAT(34)로 변환됩니다. *exp-expression*은 *numeric-expression*을 다시 스케일링하기 위한 예 패턴으로 사용됩니다. *exp-expression*의 부호와 계수는 무시됩니다.

결과의 계수는 *numeric-expression*의 계수에서 파생됩니다. 필요하면(지수가 증가하는 경우) 라운드하거나, 10제곱으로 곱하거나(지수가 감소하는 경우) 변경하지 않고 유지합니다(지수가 이미 *exp-expression*의 지수와 같은 경우).

CURRENT DECFLOAT ROUNDING MODE 특수 레지스터는 반올림 모드를 판별합니다.

10진 부동 소수점 데이터 유형에 대한 다른 산술 연산과 달리, 양자화(quantize) 조작 후 계수의 길이가 *exp-expression*에 지정된 정밀도보다 긴 경우 결과는 NaN이고 경고가 리턴됩니다(SQLSTATE 0168D). 이렇게 하면 경고 조건이 없는 경우 QUANTIZE 결과의 지수가 항상 *exp-expression*의 지수와 같게 됩니다.

- 어느 하나의 인수가 NaN이면 NaN이 리턴됩니다.
- 어느 하나의 인수가 sNaN이면, NaN이 리턴되고 경고가 리턴됩니다(SQLSTATE 01565)
- 두 인수 모두 무한대이면(양수 또는 음수) 첫 번째 인수와 부호가 같은 무한대가 리턴됩니다.
- 하나의 인수가 무한이고(양수 또는 음수) 다른 인수가 무한이 아니면, NaN이 리턴되고 경고가 리턴됩니다(SQLSTATE 0168D).

함수의 결과는 두 인수 모두 DECFLOAT(16)인 경우 DECFLOAT(16) 값입니다. 그렇지 않으면 함수 결과는 DECFLOAT(34) 값입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

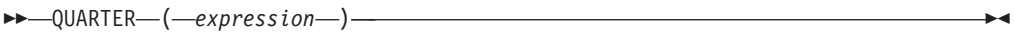
- 다음 예는 다양한 입력 10진 부동 소수점 값이 제공되는 경우 QUANTIZE 함수가 리턴하는 값을 나타냅니다. 이때 ROUND_HALF_UP 근사값 모드로 가정합니다.

```
QUANTIZE(2.17, DECFLOAT(0.001)) = 2.170
QUANTIZE(2.17, DECFLOAT(0.01)) = 2.17
QUANTIZE(2.17, DECFLOAT(0.1)) = 2.2
QUANTIZE(2.17, DECFLOAT('1E+0')) = 2
QUANTIZE(2.17, DECFLOAT('1E+1')) = 0E+1
QUANTIZE(2, DECFLOAT(INFINITY)) = NaN    -- warning
QUANTIZE(0, DECFLOAT('1E+5')) = 0E+5
QUANTIZE(217, DECFLOAT('1E-1')) = 217.0
QUANTIZE(217, DECFLOAT('1E+0')) = 217
QUANTIZE(217, DECFLOAT('1E+1')) = 2.2E+2
QUANTIZE(217, DECFLOAT('1E+2')) = 2E+2
```

- 다음 예에서, 값 -0은 QUANTIZE 함수에 대해 리턴됩니다. CHAR 함수는 클라이언트 프로그램의 마이너스 부호 제거 가능성을 방지하기 위해 사용됩니다.

```
CHAR(QUANTIZE(-0.1, DECFLOAT(1))) = -0
```

QUARTER



스키마는 SYSFUN입니다.

인수에 지정된 날짜에 대한 분기를 나타내는 1 - 4 범위의 정수 값을 리턴합니다.

인수는 날짜, 시간소인 또는 CLOB가 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열(DBCLOB 제외)이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

RADIANS

스키마는 SYSIBM입니다. (RADIANS 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

RADIANS 함수는 각도로 표시된 인수에 대해 라디안 수를 리턴합니다.

인수는 내장 숫자 데이터 유형이면 어느 것이나 가능합니다. 인수가 10진수 부동 소수점인 경우 연산은 10진수 부동 소수점으로 수행됩니다. 기타의 경우 인수는 배정밀도 부동 소수점 숫자로 변환되어 함수가 처리됩니다.

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

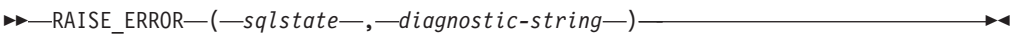
예:

- 호스트 변수 HDEG가 값이 180인 INTEGER임을 가정하십시오. 다음 명령문은

  ```
  VALUES RADIANS(:HDEG)
  ```

  값 +3.14159265358979E+000을 리턴합니다.

RAISE_ERROR



스키마는 SYSIBM입니다.

RAISE_ERROR 함수로 인해 함수를 포함하는 명령문이 지정된 SQLSTATE, SQLCODE -438 및 *diagnostic-string*과 함께 오류를 리턴합니다. RAISE_ERROR 함수는 항상 정의되지 않은 데이터 유형을 가진 NULL을 리턴합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

sqlstate

정확히 5바이트를 포함하는 문자열. 이것은 길이 5로 정의된 CHAR 유형이거나 길이 5 이상으로 정의된 VARCHAR 유형이어야 합니다. *sqlstate* 값은 다음과 같이 응용프로그램 정의 SQLSTATE의 규칙을 따라야 합니다.

- 각 문자는 숫자 세트('0' - '9')이거나 강조 표시가 없는 대문자('A' - 'Z')이어야 합니다.
- SQLSTATE 클래스(처음 두 문자)는 오류 클래스가 아니므로 '00', '01' 또는 '02'가 될 수 없습니다.
- SQLSTATE 클래스(처음 두 문자)가 문자 '0' - '6'이나 'A' - 'H'로 시작할 경우 서브클래스(마지막 세 문자)는 'I' - 'Z' 범위 내의 문자로 시작해야 합니다.
- SQLSTATE 클래스(처음 두 문자)가 문자 '7', '8', '9' 또는 'I' - 'Z'로 시작할 경우 서브클래스(마지막 세 문자)는 '0' - '9' 또는 'A' - 'Z'가 될 수 있습니다.

SQLSTATE가 이 규칙을 따르지 않는 경우 오류가 발생합니다(SQLSTATE 428B3).

diagnostic-string

오류 상태를 설명하는 최대 70바이트 문자열을 리턴하는 CHAR 또는 VARCHAR 유형의 표현식입니다. 문자열이 70바이트보다 크면 해당 문자열은 절단됩니다.

데이터 유형을 판별할 수 없는 컨텍스트에서 이 함수를 사용하려면, 캐스트 스펙을 사용하여 널(NULL)로 리턴된 값에 데이터 유형을 제공해야 합니다. CASE 표현식에서는 RAISE_ERROR 함수가 가장 유용한 표현식입니다.

예:

직원 번호와 교육 수준을 Post Graduate, Graduate 및 Diploma로 나열하십시오. 교육 수준이 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.

RAND 함수는 0과 1 사이의 부동 소수점 값을 리턴합니다.

표현식이 지정된 경우 시드(seed) 값으로 사용됩니다. 표현식은 0 및 2 147 483 647 사이의 값을 갖는 내장 SMALLINT 또는 INTEGER 데이터 유형이어야 합니다.

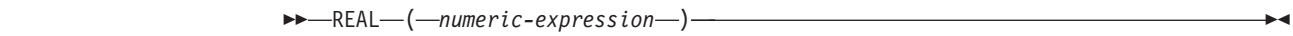
결과의 데이터 유형은 배정밀도 부동 소수점입니다. 인수가 널(NULL)인 경우 결과는 널(NULL) 값입니다.

특정 시드(seed) 값은 쿼리가 실행될 때마다 쿼리에서 특정 RAND 함수 인스턴스에 대해 동일한 무작위 숫자 시퀀스를 생성합니다. 시드(seed) 값이 지정되지 않은 경우 동일한 세션에서 쿼리가 실행될 때마다 서로 다른 무작위 숫자 시퀀스가 생성됩니다. 세션 사이에 다른 무작위 숫자 세트를 생성하려면 무작위 시드(seed)를 지정하십시오(예: 현재 시간을 기반으로 하는 무작위 시드).

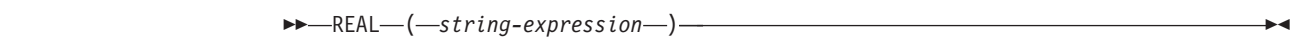
RAND는 비결정론적 함수입니다.

REAL

숫자를 real로



문자열을 real로



스키마는 SYSIBM입니다.

REAL 함수는 숫자의 단정밀도 부동 소수점 표현을 리턴합니다.

- 숫자
- 숫자의 문자열 표현

숫자를 real로

numeric-expression
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

결과는 인수가 단정밀도 부동 소수점 컬럼이나 변수에 지정된 경우에 발생하는 수와 동일합니다. 인수의 숫자 값이 단정밀도 부동 소수점 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

문자열을 real로

string-expression
숫자의 문자열 또는 유니코드 그래픽 문자열 표현인 값을 리턴하는 표현식입니다. string-expression의 데이터 유형은 CLOB 또는 DBCLOB일 수 없습니다(SQLSTATE 42884).

결과는 CAST(string-expression AS REAL)에서 발생하는 수와 동일합니다. 앞뒤 공백은 제거되고 결과 문자열은 유효한 숫자 상수를 작성하는 규칙에 따라야 합니다(SQLSTATE 22018). 인수의 숫자 값이 단정밀도 부동 소수점 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

함수의 결과는 단정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

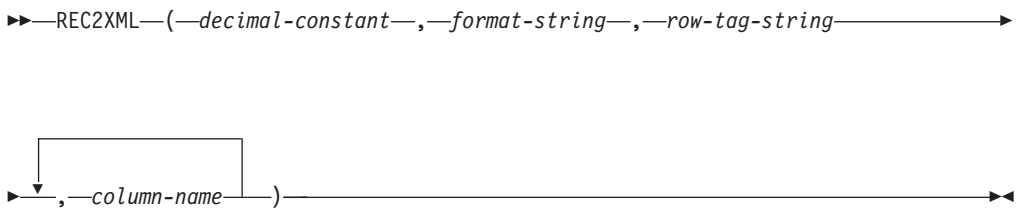
주: CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예:

EMPLOYEE 테이블을 사용하여 수당이 0이 아닌 직원들의 수당에 대한 급여 비율을 찾습니다. 관련된 컬럼(SALARY 및 COMM)은 DECIMAL 데이터 유형을 가집니다. 결과는 단정밀도 부동 소수점이어야 합니다. 따라서 부동 소수점(실질적으로 배정밀도)에서 나눗셈이 수행되도록 REAL이 SALARY에 적용된 다음 단정밀도 부동 소수점의 결과를 리턴하도록 REAL이 전체 표현식에 적용됩니다.

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
  FROM EMPLOYEE
  WHERE COMM > 0
```

REC2XML



스키마는 SYSIBM입니다.

REC2XML 함수는 컬럼 이름 및 컬럼 데이터를 포함하는 문자열(XML 태그 형식)을 리턴합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

decimal-constant

컬럼 데이터 문자를 대체하기 위한 확장 인수입니다. 소수 값은 0.0보다 크고 6.0 이하여야 합니다(SQLSTATE 42820).

decimal-constant 값은 함수의 결과 길이를 계산하는 데 사용됩니다. 문자 데이터 유형을 갖는 모든 컬럼의 경우 결과 길이에 추가되기 전에 컬럼의 길이 속성에 이 확장 인수가 곱해집니다.

확장을 지정하지 않으려면 값 1.0을 사용하십시오. 1.0보다 작은 값을 지정하면 계산된 결과 길이가 줄어듭니다. 결과 문자열의 실제 길이가 함수의 계산된 결과 길이보다 크면 오류가 발생합니다(SQLSTATE 22001).

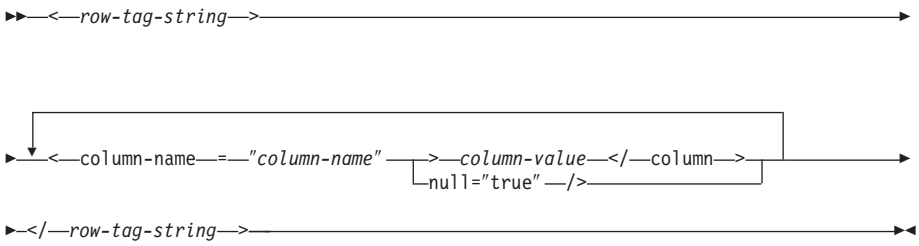
format-string

함수가 실행 중에 사용할 형식을 지정하는 문자열 상수입니다.

*format-string*은 대소문자를 구분하므로 다음 값이 인식되려면 대문자로 지정해야 합니다.

COLATTVAL 또는 COLATTVAL_XML

이들 형식은 컬럼을 속성 값으로 갖는 문자열을 리턴합니다.



컬럼 이름은 유효한 XML 속성 값이거나 아닐 수 있습니다. 유효한 XML 속성 값이 아닌 컬럼 이름의 경우 결과 문자열에 포함되기 전에 컬럼 이름에서 문자 교체가 수행됩니다.

컬럼 값은 유효한 XML 요소 이름이거나 아닐 수 있습니다. *format-string* COLATTVAL이 지정되는 경우, 유효한 XML 요소 값이 아닌 컬럼 이름의 경우 결과 문자열에 포함되기 전에 컬럼 값에서 문자 교체가 수행됩니다. *format-string* COLATTVAL_XML이 지정되는 경우, 컬럼 값에서 문자 교체가 수행되지 않습니다(컬럼 이름에서는 여전히 문자 교체가 수행됨).

row-tag-string

각 행에 사용되는 태그를 지정하는 문자열 상수입니다. 비어 있는 문자열이 지정되면 값 'row'가 가정됩니다.

하나 이상의 공백 문자의 문자열이 지정되는 경우 시작 *row-tag-string* 또는 종료 *row-tag-string*(각진 괄호 분리문자 포함)이 결과 문자열에 나타나지 않습니다.

column-name

테이블 컬럼 이름에 대해 규정된 이름이거나 규정되지 않은 이름입니다. 컬럼은 다음 데이터 유형 중 하나여야 합니다(SQLSTATE 42815).

- 숫자(SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE)
- 문자열(CHAR, VARCHAR. BIT DATA의 부속 유형을 갖는 문자열은 허용되지 않음)
- 날짜 시간(DATE, TIME, TIMESTAMP)
- 위의 유형 중 하나를 기초로 하는 사용자 정의 유형

같은 컬럼 이름을 두 번 이상 지정할 수 없습니다(SQLSTATE 42734).

함수의 결과는 VARCHAR입니다. 최대 길이는 32 672바이트입니다(SQLSTATE 54006).

다음 호출을 고려하십시오.

```
REC2XML (dc, fs, rt, c1, c2, ..., cn)
```

″fs″의 값이 ″COLATTVAL″ 또는 ″COLATTVAL_XML″인 경우, 결과는 다음 표현식과 같습니다.

```
'<' CONCAT rt CONCAT '>' CONCAT y1 CONCAT y2
CONCAT ... CONCAT yn CONCAT '</' CONCAT rt CONCAT '>'
```

여기서 y_n은 다음과 동등합니다.

```
'<column name="' CONCAT xvcn CONCAT vn
```

또한 vn은 다음과 동등합니다.

```
'">' CONCAT rn CONCAT '</column>'
```

컬럼이 널(null)이 아닌 경우 및

```
'" null="true"/>'
```

컬럼 값이 널(null)인 경우입니다.

xvc_n은 c_n의 컬럼 이름의 문자열 표현이며, 표 54에 나타나는 모든 문자가 대응하는 표현으로 바뀝니다. 이는 결과 문자열이 유효한 XML 속성 또는 요소 값 토큰임을 보장합니다.

r_n은 표 53에서 표시되는 문자열 표현과 동일합니다.

표 53. 컬럼 값 문자열 결과

c_n의 데이터 유형	**r_n**
CHAR, VARCHAR	값은 문자열입니다. *format-string*이 문자 "_XML"로 끝나지 않는 경우, c_n의 각 문자가 표시된 것처럼 표 54의 대응하는 교체 표현으로 바뀝니다. 길이 속성은 dc * c_n의 길이 속성입니다.
SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE	값은 LTRIM(RTRIM(CHAR(c_n)))입니다. 길이 속성은 CHAR(c_n)의 결과 길이입니다. 소수점 문자는 항상 마침표('.') 문자입니다.
DATE	값은 CHAR(c_n,ISO)입니다. 길이 속성은 CHAR(c_n,ISO)의 결과 길이입니다.
TIME	값은 CHAR(c_n,JIS)입니다. 길이 속성은 CHAR(c_n,JIS)의 결과 길이입니다.
TIMESTAMP	값은 CHAR(c_n)입니다. 길이 속성은 CHAR(c_n)의 결과 길이입니다.

문자 교체:

*format-string*에 대해 지정되는 값에 따라서, 컬럼 이름이 유효한 XML 속성 값을 형성하고 컬럼 값이 유효한 XML 요소 값을 형성하도록 컬럼 이름 및 컬럼 값의 특정 문자가 교체됩니다.

표 54. XML 속성 값 및 요소 값에 대한 문자 교체

문자	교체
<	<
>	>
"	"
&	&
'	'

예를 들면, 다음과 같습니다.

주: REC2XML은 출력에 공백 스페이스나 개행 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- 샘플 데이터베이스에서 DEPARTMENT 테이블을 사용하여 부서 'D01'에 대해 DEPTNAME 및 LOCATION 컬럼을 제외한 부서 테이블 행을 XML 문자열로 형식화하십시오. 데이터가 교체가 필요한 어떤 문자도 포함하지 않으므로 확장 인수는 1.0(확장 없음)입니다. 또한 이 행의 경우 MGRNO 값이 널(null)입니다.

  ```
  SELECT REC2XML (1.0, 'COLATTVAL', '', DEPTNO, MGRNO, ADMRDEPT)
    FROM DEPARTMENT
    WHERE DEPTNO = 'D01'
  ```

  이 요소는 다음 VARCHAR(117) 문자열을 리턴합니다.

  ```
  <row>
  <column name="DEPTNO">D01</column>
  <column name="MGRNO" null="true"/>
  <column name="ADMRDEPT">A00</column>
     </row>
  ```

- 5일 대학 스케줄은 CLASS_CODE 컬럼에 대해 새 형식을 사용하여 '&43<FIE'라는 클래스를 CL_SCHED라는 테이블에 도입합니다. REC2XML 함수를 사용하여 이 예는 클래스 종료 시간을 제외하고 이 새 클래스 데이터로 XML 문자열을 형식화합니다.

  확장 인수 1.0을 갖는 REC2XML 호출(아래 참조)에 대한 길이 속성은 128입니다('<row>' 및 '</row>' 오버헤드에 대해 11, 컬럼 이름에 대해 21, '<column name=', '>', '</column>' 및 큰따옴표에 대해 75, CLASS_CODE 데이터에 대해 7, DAY 데이터에 대해 6 및 STARTING 데이터에 대해 8). '&' 및 '<' 문자가 교체될 것이므로 확장 인수 1.0은 충분하지 않습니다. 함수의 길이 속성은 새 형식 CLASS_CODE 데이터에 대해 7에서 14바이트로의 증가를 지원해야 합니다.

  그러나 DAY 값은 길이가 1바이트를 넘지 않을 것임이 알려졌으므로 길이의 사용되지 않는 추가 5 단위가 총계에 추가됩니다. 그러므로 확장은 2의 증가만 처리해야 합니다. CLASS_CODE가 인수 목록에 있는 유일한 문자열 컬럼이므로 이것이 확장 인수가 적용되는 유일한 컬럼 데이터입니다. 길이에 대해 2의 증가를 얻으려면 9/7(대략 1.2857)의 확장 인수가 필요합니다. 1.3의 확장 인수를 사용하게 됩니다.

  ```
  SELECT REC2XML (1.3, 'COLATTVAL', 'record', CLASS_CODE, DAY, STARTING)
       FROM CL_SCHED
       WHERE CLASS_CODE = '&43<FIE'
  ```

  이 예는 다음 VARCHAR(167) 문자열을 리턴합니다.

  ```
  <record>
  <column name="CLASS_CODE">&amp;43&lt;FIE</column>
  <column name="DAY">5</column>
  <column name="STARTING">06:45:00</column>
  </record>
  ```

- 새 행이 샘플 데이터베이스의 EMP_RESUME 테이블에 추가되었다고 가정하십시오. 새 행은 다시 시작을 유효한 XML의 문자열로서 저장합니다. COLATTVAL_XML *format-string*이 사용되므로 문자 교체는 수행되지 않습니다. 다시 시작은 길이가 3500

바이트를 넘지 않습니다. 다음 쿼리가 EMP_RESUME 테이블에서 다시 시작의 XML 버전을 선택하고 XML 문서 조각으로 형식화하는 데 사용됩니다.

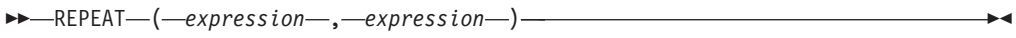
```
SELECT REC2XML (1.0, 'COLATTVAL_XML', 'row', EMPNO, RESUME_XML)
  FROM (SELECT EMPNO, CAST(RESUME AS VARCHAR(3500)) AS RESUME_XML
    FROM EMP_RESUME
    WHERE RESUME_FORMAT = 'XML')
  AS EMP_RESUME_XML
```

이 예는 XML 형식으로 다시 시작을 갖는 각 직원에 대해 행 하나를 리턴합니다. 리턴되는 각 행은 다음 형식을 갖는 문자열입니다.

```
<row>
<column name="EMPNO">{employee number}</column>
<column name="RESUME_XML">{resume in XML}</column>
</row>
```

여기서 ″{employee number}″는 컬럼에 대한 실제 EMPNO 값이고 ″{resume in XML}″은 다시 시작인 실제 XML 조각 문자열 값입니다.

REPEAT



스키마는 SYSFUN입니다.

두 번째 인수에 지정된 횟수만큼 반복되는 첫 번째 인수로 구성된 문자열을 리턴합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

첫 번째 인수는 문자열 또는 실행 파일 문자열 유형입니다. VARCHAR의 최대 길이는 4000바이트이고 CLOB 또는 실행 파일 문자열의 최대 길이는 1 048 576바이트입니다. 두 번째 인수는 SMALLINT 또는 INTEGER가 될 수 있습니다.

함수의 결과는 다음과 같습니다.

- 첫 번째 인수가 VARCHAR(4000바이트 미만) 또는 CHAR이면 VARCHAR(4000)입니다.
- 첫 번째 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.
- 첫 번째 인수가 BLOB이면 BLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

예:

- 'REPEAT THIS' 구문을 5번 나열하십시오.

```
VALUES CHAR(REPEAT('REPEAT THIS', 5), 60)
```

이 예에서는 다음을 리턴합니다.

```
1
------------------------------------------------------------
REPEAT THISREPEAT THISREPEAT THISREPEAT THISREPEAT THIS
```

언급한 대로, REPEAT 함수의 결과는 VARCHAR(4000)입니다. 이 예의 경우, REPEAT 출력을 60바이트로 제한하기 위해 CHAR 함수가 사용되었습니다.

REPLACE

스키마는 SYSIBM입니다. REPLACE 함수의 SYSFUN 버전은 계속해서 사용할 수 있지만, 데이터베이스 조합에 민감하지 않습니다.

*source-string*에서 *search-string*의 모든 어커런스를 *replace-string*으로 대체합니다. *search-string*이 *source-string*에 없으면, *search-string*이 변경되지 않고 리턴됩니다. 실행 파일 비교를 사용하여 검색이 수행되는 경우, *source-string*, *search-string* 또는 *replace-string*이 FOR BIT DATA로 정의되지 않으면, 데이터베이스 조합을 사용하여 검색이 수행됩니다.

source-string

소스 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

search-string

소스 문자열에서 제거될 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

replace-string

교체 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 표현식이 비어 있는 문자열인 경우 소스 문자열에서 제거되는 문자열을 교체하는 것이 없습니다.

각 문자열의 실제 길이는 문자열의 경우 32 672바이트 미만이며, 그래픽 문자열의 경우 16 336 미만입니다. 세 인수 모두는 호환 가능한 데이터 유형이어야 합니다.

source-string, *search-string* 또는 *replace-string*이 FOR BIT DATA로 정의되어 있으면, 결과는 VARCHAR FOR BIT DATA입니다. *source-string*이 문자열인 경우 결과는 VARCHAR입니다. *source-string*이 그래픽 문자열인 경우 결과는 VARGRAPHIC입니다. 하나의 인수가 문자 FOR BIT DATA인 경우, 다른 인수는 그래픽이어서는 안됩니다(SQLSTATE 42846).

결과의 길이 속성은 인수에 따라 다릅니다.

- *replace-string*의 길이 속성이 *search-string*의 길이 속성보다 작거나 같으면, 결과의 길이 속성은 *source-string*의 길이 속성입니다.
- *replace-string*의 길이 속성이 *search-string*의 길이 속성보다 작거나 크면, 결과의 길이 속성은 다음과 같이 결과의 데이터 유형에 따라 판별됩니다.
  - VARCHAR의 경우:
    - L1 < = 4000인 경우, 결과의 길이 속성은 `MIN(4000, (L3*(L1/L2)) + MOD(L1,L2))`입니다.
    - 그 외의 경우 결과의 길이 속성은 `MIN(32672, (L3*(L1/L2)) + MOD(L1,L2))` 입니다.
  - VARGRAPHIC의 경우:
    - L1 < = 2000인 경우, 결과의 길이 속성은 `MIN(2000, (L3*(L1/L2)) + MOD(L1,L2))`입니다.
    - 그 외의 경우 결과의 길이 속성은 `MIN(16336, (L3*(L1/L2)) + MOD(L1,L2))` 입니다.

  여기서:
  - L1은 *source-string*의 길이 속성입니다.
  - 검색 문자열이 문자열 상수인 경우 L2는 *search-string*의 길이 속성입니다. 그 외의 경우 L2는 1입니다.
  - L3은 *replace-string*의 길이 속성입니다.

결과가 문자열인 경우 결과의 길이 속성은 32 672를 초과해서는 안됩니다. 결과가 그래픽 문자열인 경우 결과의 길이 속성은 16 336을 초과해서는 안됩니다.

결과의 실제 길이는 *source-string*의 실제 길이에 *replace-string*의 실제 길이로 곱한 *source-string*에 있는 *search-string*의 어커런스 수를 더하고 *search-string*의 실제 길이를 뺀 값입니다.

*replace-string*의 실제 길이가 리턴 데이터 유형에 대한 최대값을 초과할 경우, 오류가 리턴됩니다. 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- 단어 ‘DINING’에서 문자 ‘N’의 모든 어커런스를 ‘VID’로 대체하십시오.

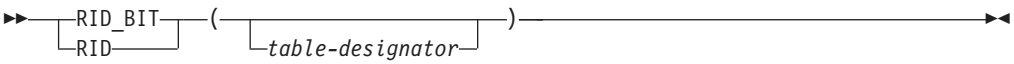
  ```
  VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
  ```

  결과는 문자열 ’DIVIDIVIDG’입니다.
- 대소문자를 구분하지 않는 조합 UCA500R1_LEN_S1이 있는 유니코드 데이터베이스에서, 단어 ’QUICK’를 단어 ’LARGE’로 교체합니다.

```
VALUES REPLACE ('The quick brown fox', 'QUICK', 'LARGE')
```

결과는 문자열 'The LARGE brown fox'입니다.

RID_BIT 및 RID



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

RID_BIT 및 RID 함수는 행의 행 ID(RID)를 다른 양식으로 리턴합니다. RID는 행을 고유하게 식별하기 위해 사용됩니다. 각 함수는 하나의 행에 대해 여러 번 호출될 경우 다른 값을 리턴할 수 있습니다. 예를 들어, 테이블에 대해 reorg 유틸리티가 실행되면 RID_BIT 및 RID 함수는 유틸리티를 실행하기 전에 리턴된 것과 다른 값을 행에 대해 리턴할 수 있습니다. RID_BIT 및 RID 함수는 비결정적 함수입니다. RID 함수와 달리 RID_BIT 함수 결과에는 실수로 다른 테이블과 함께 사용하는 일이 없도록 보호하기 위한 테이블 정보가 포함됩니다. RID 함수는 파티션된 데이터베이스 환경에서 지원되지 않습니다.

table-designator

기본 테이블, 뷰 또는 중첩 테이블 표현식을 고유하게 식별합니다(SQLSTATE 42703). *table-designator*가 뷰 또는 중첩 테이블 표현식을 지정하는 경우 RID_BIT 및 RID 함수는 뷰 또는 중첩 테이블 표현식의 기본 테이블 RID를 리턴합니다. 지정된 뷰 또는 중첩 테이블 표현식은 외부 subselect에 단 하나의 기본 테이블만 포함해야 합니다(SQLSTATE 42703). *table-designator*는 삭제 가능해야 합니다(SQLSTATE 42703). 삭제 가능한 뷰에 대한 정보는 『CREATE VIEW』의 『주의사항』 섹션을 참조하십시오.

*table-designator*를 지정하지 않은 경우 FROM 절은 테이블 지정자가 되도록 파생될 수 있는 하나의 요소만 포함해야 합니다(SQL STATE 42703).

RID_BIT 함수의 결과는 VARCHAR (16) FOR BIT DATA입니다. 결과는 널(NULL)이 될 수 있습니다. RID 함수의 결과는 BIGINT입니다. 결과는 널(NULL)이 될 수 있습니다.

참고:

- 응용프로그램에서 낙관적 잠금을 구현하려면, ROW CHANGE TOKEN 표현식이 RID_BIT 스칼라 함수에 인수로 리턴한 값을 사용하십시오.
- **호환성:** 다음 구문은 지원되지만 표준이 아니므로 사용하면 안됩니다.
  - 의사(pseudo) 컬럼 『ROWID』를 사용하여 RID를 참조할 수 있습니다. 규정되지 않은 ROWID 참조는 RID_BIT()와 같고 규정된 ROWID(예: EMPLOYEE.ROWID)는 RID_BIT(EMPLOYEE)와 같습니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블에서 부서 20에 있는 사원의 RID 및 성을 리턴하십시오.

```
SELECT RID_BIT (EMPLOYEE), ROW CHANGE TOKEN FOR EMPLOYEE, LASTNAME
  FROM EMPLOYEE
  WHERE DEPTNO = '20'
```

- 다음과 같이 정의된 테이블 EMP1이 제공됩니다.

```
CREATE TABLE EMP1 (
  EMPNO   CHAR(6),
        NAME CHAR(30),
        SALARY       DECIMAL(9,2),
  PICTURE BLOB(250K),
  RESUME  CLOB(32K)
)
```

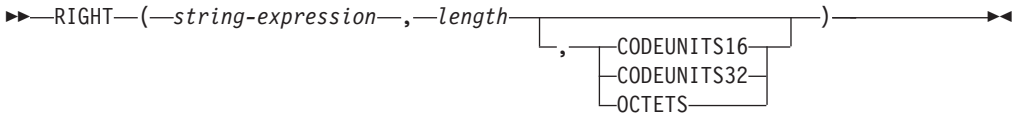
호스트 변수 HV_EMP_RID를 RID_BIT 내장 스칼라 함수의 값으로 설정하고 HV_EMP_RCT를 사원 번호 3500에 해당되는 행에 대한 ROW CHANGE TOKEN 표현식 값으로 설정하십시오.

```
SELECT RID_BIT(EMP1), ROW CHANGE TOKEN FOR EMP1
  INTO :HV_EMP_RID, :HV_EMP_RCT FROM EMP1
  WHERE EMPNO = '3500'
```

해당 RID 값을 사용하여 사원을 식별하고 사용자 정의 함수 UPDATE_RESUME이 사원의 급료를 $1000씩 늘리고 사원의 이력을 갱신합니다.

```
UPDATE EMP1 SET
  SALARY = SALARY + 1000,
  RESUME = UPDATE_RESUME(:HV_RESUME)
WHERE RID_BIT(EMP1) = :HV_EMP_RID
AND ROW CHANGE TOKEN FOR EMP1 = :HV_EMP_RCT
```

RIGHT



스키마는 SYSIBM입니다. RIGHT 함수의 SYSFUN 버전은 계속 사용 가능합니다.

RIGHT 함수는 지정된 문자열 단위에서 표현된 길이 *length*의 *string-expression*의 맨 오른쪽 문자열을 리턴합니다. *string-expression*이 문자열이면 결과가 문자열입니다. *string-expression*이 그래픽 문자열이면 결과는 그래픽 문자열입니다.

string-expression
결과가 파생될 문자열을 지정하는 표현식입니다. 이 표현식은 내장 문자열, 숫자 또는 날짜 및 시간 데이터 유형의 값을 리턴합니다. 값이 문자열 데이터 유형이 아니면, 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *string-expression*의 하위 문자열은 *string-expression*의 0 이상의 인접 코드 포인트입니다.

length
결과 길이를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 명시적으로 INTEGER로 캐스트됩니다.값은 내재적으로나 명시적으로 지정된 단위에서 표현된 0과 *string-expression*의 길이 사이여야 합니다(SQLSTATE 22011). OCTETS가 지정되어 있고 결과가 그래픽 데이터이면, 값은 0과 *string-expression*의 속성 길이의 두 배 사이여야 합니다(SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 또는 OCTETS
*length*의 문자열 단위를 지정합니다.

CODEUNITS16은 *length*를 16비트 UTF-16 코드 단위로 표시할 것을 지정합니다. CODEUNITS32는 *length*가 32비트 UTF-32 코드 단위로 표시할 것을 지정합니다. OCTETS는 *length*를 바이트로 표시할 것을 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *string-expression*이 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *string-expression*이 그래픽 문자열이면, *length*는 짝수여야 합니다. 그 외의 경우에는 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 결과가 그래픽 데이터이면 *length*는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다. CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는, 『문자열』의 『내장 함수의 문자열 단위』를 참조하십시오.

*string-expression*은 *string-expression*의 지정된 부속 문자열이 항상 존재하도록 필요한 수의 공백이 오른쪽에 채워집니다. 채우는 데 사용된 문자는 채울 컨텍스트의 문자열을 채우는 데 사용된 문자와 같습니다. 채우기에 관한 자세한 정보는 『지정 및 비교』에서 『문자열 지정』을 참조하십시오.

함수의 결과는 *string-expression*의 길이 속성과 같은 길이 속성인 다양한 길이 문자열이며 *string-expression*의 데이터 유형에 따라 다른 데이터 유형입니다.

- *string-expression*이 CHAR 또는 VARCHAR인 경우 VARCHAR
- *string-expression*이 CLOB인 경우 CLOB
- *string-expression*이 GRAPHIC 또는 VARGRAPHIC인 경우 VARGRAPHIC
- *string-expression*이 DBCLOB인 경우 DBCLOB
- *string-expression*이 BLOB인 경우 BLOB

결과의 실제 길이는(문자열 단위에서) *length*입니다.

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- 변수 ALPHA에 값 'ABCDEF'가 있다고 가정하십시오. 다음 명령문은

  ```
  SELECT RIGHT(ALPHA,3)
    FROM SYSIBM.SYSDUMMY1
  ```

  ALPHA에서 맨 오른쪽에서 3개의 문자인 'DEF'를 리턴합니다.

- VARCHAR(50)로 정의된 변수 NAME에 값 'KATIE AUSTIN'이 있으며 정수 변수 LASTNAME_LEN에 값 6이 있다고 가정하십시오. 다음 명령문은

  ```
  SELECT RIGHT(NAME,LASTNAME_LEN)
    FROM SYSIBM.SYSDUMMY1
  ```

  'AUSTIN' 값을 리턴합니다.

- 다음 명령문은 길이가 0인 문자열을 리턴합니다.

  ```
  SELECT RIGHT('ABCABC',0)
    FROM SYSIBM.SYSDUMMY1
  ```

- EMPLOYEE 테이블의 FIRSTNME 컬럼은 VARCHAR(12)로 정의됩니다. 성이 'BROWN'인 직원의 이름을 찾은 다음 10바이트 문자열의 이름을 리턴합니다.

  ```
  SELECT RIGHT(FIRSTNME, 10)
    FROM EMPLOYEE
    WHERE LASTNAME = 'BROWN'
  ```

  값 'DAVID' 다음에 5개의 공백 문자가 있는 VARCHAR(12) 문자열을 리턴합니다.

- 유니코드 데이터베이스에서 FIRSTNAME은 VARCHAR(12) 컬럼입니다. 해당 값 중 하나는 6자 문자열 'Jürgen'입니다. FIRSTNAME이 이 값을 가질 때 리턴되는 값은 다음과 같습니다.

```
함수...                          다음을 리턴...

RIGHT(FIRSTNAME,5,CODEUNITS32)   'ürgen' -- x'C3BC7267656E'
RIGHT(FIRSTNAME,5,CODEUNITS16)   'ürgen' -- x'C3BC7267656E'
RIGHT(FIRSTNAME,5,OCTETS)        'rgen'  -- x'207267656E', a truncated string
```

- 다음 예는 유니코드 문자열 '&N~AB'에 대한 작업업의 예로, 여기서 '&'는 음악 기호 높은 음자리표이고 '~'는 조인 틸드 문자입니다. 해당 문자열은 아래의 다른 유니코드 인코딩 양식에 표시됩니다.

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

20바이트의 길이 속성이 있는 변수 UTF8_VAR에 문자열의 UTF-8 표현을 포함한다고 가정하십시오.

```
SELECT RIGHT(UTF8_VAR, 2, CODEUNITS16),
  RIGHT(UTF8_VAR, 2, CODEUNITS32),
  RIGHT(UTF8_VAR, 2, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 'AB', 'AB' 및 'AB'를 각각 리턴합니다.

```
SELECT RIGHT(UTF8_VAR, 5, CODEUNITS16),
  RIGHT(UTF8_VAR, 5, CODEUNITS32),
  RIGHT(UTF8_VAR, 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '?N~AB', '&N~AB' 및 'N~AB'를 각각 리턴합니다. 여기서 '?' 는 X'EDB49E' 입니다.

```
SELECT RIGHT(UTF8_VAR, 10, CODEUNITS16),
  RIGHT(UTF8_VAR, 10, CODEUNITS32),
  RIGHT(UTF8_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

값 '&N~ABbbbb', '&N~ABbbbbb' 및 '&N~ABb'를 각각 리턴하고, 여기서 'b' 는 공백 문자를 나타냅니다.

20 코드 단위의 길이 속성이 있는 변수 UTF16_VAR에 문자열의 UTF-16BE 표현을 포함한다고 가정하십시오.

```
SELECT RIGHT(UTF16_VAR, 2, CODEUNITS16),
  RIGHT(UTF16_VAR, 2, CODEUNITS32),
  RIGHT(UTF16_VAR, 2, OCTETS))
FROM SYSIBM.SYSDUMMY1
```

값 'AB', 'AB' 및 'B'를 각각 리턴합니다.

```
SELECT RIGHT(UTF16_VAR, 5, CODEUNITS16),
  RIGHT(UTF16_VAR, 5, CODEUNITS32),
  RIGHT(UTF16_VAR, 6, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

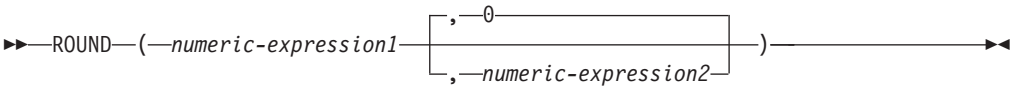
값 '?N~AB', '&N~AB' 및 '~AB'를 각각 리턴하며, 여기서 '?' 는 독립형 대체 X'DD1E'입니다.

```
SELECT RIGHT(UTF16_VAR, 10, CODEUNITS16),
  RIGHT(UTF16_VAR, 10, CODEUNITS32),
  RIGHT(UTF16_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

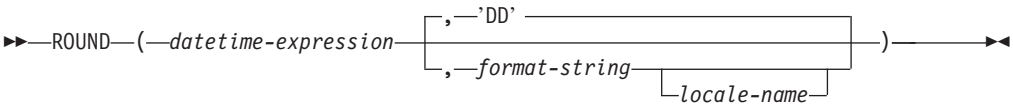
값 '&N~ABbbbb', '&N~ABbbbbb' 및 '?N~AB'를 각각 리턴하고, 여기서 'b'는 공백 문자를 나타내며 '?'는 X'DD1E'입니다.

ROUND

ROUND 숫자:



ROUND 날짜 시간:



스키마는 SYSIBM입니다. ROUND 숫자 함수의 SYSFUN 버전은 계속 사용 가능합니다.

ROUND 함수는 다음의 반올림 값을 리턴합니다.

- 첫 번째 인수의 결과가 숫자 값이면, 소수점 오른쪽이나 왼쪽으로 지정된 자릿수로 반올림된 숫자
- 첫 번째 인수가 날짜, 시간 또는 시간소인인 경우, *format-string*에서 지정한 단위로 반올림된 날짜 시간 값

ROUND 숫자

*numeric-expression1*이 양수일 경우, 5 이상의 숫자 값은 다음으로 높은 양수의 반올림을 표시합니다. 예를 들어, ROUND(3.5,0) = 4입니다. *numeric-expression1*이 음수일 경우 5 이상의 숫자 값은 다음으로 낮은 음수의 반올림을 표시합니다. 예를 들어, ROUND(-3.5,0) = -4입니다.

numeric-expression1

내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC 또는 숫자 데이터 유형인 값을 리턴해야 하는 표현식. 값이 숫자 데이터 유형이 아니면, 함수를 평가하기 전에 내재적으로 DECFLOAT(34)로 캐스트됩니다.

표현식이 10진수 부동 소수점 데이터 유형이면 DECFLOAT 반올림 모드가 사용되지 않습니다. ROUND의 반올림 동작은 ROUND_HALF_UP의 값에 해당됩니다. 다른 반올림 동작이 필요하면 QUANTIZE 함수를 사용합니다.

numeric-expression2

내장 숫자 데이터 유형의 값을 리턴하는 표현식입니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다.

*numeric-expression2*가 음수가 아닌 경우 *numeric-expression1*은 소수점 오른쪽에 있는 *numeric-expression2* 숫자의 절대값으로 반올림됩니다.

*numeric-expression2*가 음수인 경우 *numeric-expression1*은 소수점의 왼쪽에 있는 *numeric-expression2* + 1 숫자의 절대값으로 반올림됩니다.

음수 *numeric-expression2*의 절대값이 소수점 왼쪽의 숫자보다 크면 결과는 0이 됩니다. 예를 들어, ROUND(748.58,-4) = 0입니다. *numeric-expression1*이 양수일 경우, 5의 숫자 값은 다음으로 높은 양수로 반올림됩니다. *numeric-expression1*이 음수일 경우, 5의 숫자 값은 다음으로 낮은 음수로 반올림됩니다.

결과의 데이터 유형과 길이 속성은 첫 번째 인수의 데이터 유형 및 길이 속성과 동일하며 *numeric-expression1*이 DECIMAL이고 정밀도가 31보다 작은 경우에 정밀도가 1만큼 증가하는 것은 예외입니다. 예를 들어, 데이터 유형이 DECIMAL(5,2)인 인수의 결과는 DECIMAL(6,2)입니다. 데이터 유형이 DECIMAL(31,2)인 인수의 결과는 DECIMAL(31,2)입니다. 스케일은 첫 번째 인수의 스케일과 동일합니다.

인수가 널(NULL)이 될 수 있거나 인수가 10진수 부동 소수점 숫자가 아니고 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성하면, 결과는 널(NULL)이 될 수 있습니다. 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

이 함수는 10진수 부동 소수점 인수이더라도 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터의 설정에 의한 영향을 받지 않습니다. ROUND의 반올림 동작은 ROUND_HALF_UP의 값에 해당됩니다. CURRENT DECFLOAT ROUNDING MODE 특수 레지스터로 지정된 반올림 모드를 준수하는 10진수 부동 소수점 값에 대한 동작을 수행하려면 QUANTIZE 함수를 대신 사용하십시오.

ROUND 날짜 시간

*datetime-expression*에 날짜 시간 데이터 유형이 있으면, ROUND 함수는 *format-string*에서 지정한 단위로 반올림된 *datetime-expression*을 리턴합니다. *format-string*이 지정되어 있지 않으면, 'DD'가 *format-string*에 대해 지정된 것처럼 *datetime-expression*이 가장 가까운 날로 반올림됩니다.

datetime-expression

날짜, 시간 또는 시간소인의 값을 리턴해야 하는 표현식입니다. 이들 데이터 유형의 문자열 표현은 지원되지 않으며 이 함수를 사용하기 위해 날짜, 시간 또는 시간소인에 명시적으로 캐스트되어야 합니다. 또는 날짜나 시간소인의 문자열 표현에 대해 ROUND_TIMESTAMP 함수를 사용할 수 있습니다.

format-string

254바이트보다 크지 않은 실제 길이의 내장 문자열 데이터 유형을 리턴하는 표현식입니다. *format-string*에서의 형식 요소는 *datetime-expression*이

반올림되어야 하는 방식을 지정합니다. 예를 들어 *format-string*이 'DD'인 경우, *datetime-expression*으로 표현된 시간소인은 가장 가까운 날로 반올림됩니다. 앞뒤 공백은 문자열에서 제거되며 결과 부속 문자열은 *datetime-expression*의 유형에 대해 유효한 형식 요소여야 합니다(SQLSTATE 22007). 디폴트는 'DD'이며 *datetime-expression*의 데이터 유형이 TIME이면 사용할 수 없습니다.

*format-string*에 대해 허용 가능한 값은 아래 나열된 형식 요소의 테이블에 나열됩니다.

locale-name

형식 요소 값 DAY, DY 또는 D를 사용할 때 해당 주의 첫 번째 날을 판별하는 데 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자를 구분하지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 이름에 대해서는 『SQL 및 XQuery에 대한 로케일 이름』을 참조하십시오. *locale-name*이 지정되지 않으면, 특수 레지스터의 값 CURRENT LOCALE LC_TIME이 사용됩니다.

함수의 결과는 *datetime-expression*와 같은 데이터 유형을 갖습니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

다음 형식 요소는 ROUND, ROUND_TIMESTAMP, TRUNCATE 및 TRUNC_TIMESTAMP 함수의 날짜 시간 값의 반올림 또는 절단 단위를 식별하는 데 사용됩니다.

표 55. ROUND, ROUND_TIMESTAMP, TRUNCATE 및 TRUNC_TIMESTAMP에 대한 형식 요소

형식 요소	단위 반올림 또는 절단	ROUND 예	TRUNCATE 예
CC SCC	Century 해당 세기의 50번째 해 이후에 다음 세기의 시작으로 반올림합니다(예: 1951–01–01–00.00.00). TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 1897-12-04-12.22.22.000000 결과: 1901-01-01-00.00.00.000000	입력 값: 1897-12-04-12.22.22.000000 결과: 1801-01-01-00.00.00.000000
SYYYY YYYY YEAR SYEAR YYY YY Y	Year 7월 1일을 다음 해의 1월 1일로 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 1897-12-04-12.22.22.000000 결과: 1898-01-01-00.00.00.000000	입력 값: 1897-12-04-12.22.22.000000 결과: 1897-01-01-00.00.00.000000

표 55. ROUND, ROUND_TIMESTAMP, TRUNCATE 및 TRUNC_TIMESTAMP에 대한 형식 요소 (계속)

형식 요소	단위 반올림 또는 절단	ROUND 예	TRUNCATE 예
IYYY IYY IY I	ISO Year 7월 1일을 다음 ISO 해의 첫 번째 날로 반올림합니다. ISO 해의 첫 번째 날은 첫 번째 ISO 주의 월요일로 정의됩니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 1897-12-04-12.22.22.000000 결과: 1898-01-03-00.00.00.000000	입력 값: 1897-12-04-12.22.22.000000 결과: 1897-01-04-00.00.00.000000
Q	Quarter 분기의 두 번째 월의 16번째 날에서 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 1999-06-04-12.12.30.000000 결과: 1999-07-01-00.00.00.000000	입력 값: 1999-06-04-12.12.30.000000 결과: 1999-04-01-00.00.00.000000
MONTH MON MM RM	Month 월의 16번째 날에서 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 1999-06-18-12.12.30.000000 결과: 1999-07-01-00.00.00.000000	입력 값: 1999-06-18-12.12.30.000000 결과: 1999-06-01-00.00.00.000000
WW	해당 연도의 첫 번째 날과 같은 요일. 해당 연도의 첫 번째 날에 대해 4번째 요일의 12시간을 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 2000-05-05-12.12.30.000000 결과: 2000-05-06-00.00.00.000000	입력 값: 2000-05-05-12.12.30.000000 결과: 2000-04-29-00.00.00.000000
IW	ISO 해의 첫 번째 날과 같은 요일. 자세한 내용은 『WEEK_ISO 스칼라 함수』를 참조하십시오. ISO 해의 첫 번째 날에 대해 4번째 요일의 12시간을 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 2000-05-05-12.12.30.000000 결과: 2000-05-08-00.00.00.000000	입력 값: 2000-05-05-12.12.30.000000 결과: 2000-05-01-00.00.00.000000

표 55. ROUND, ROUND_TIMESTAMP, TRUNCATE 및 TRUNC_TIMESTAMP에 대한 형식 요소 (계속)

형식 요소	단위 반올림 또는 절단	ROUND 예	TRUNCATE 예
W	해당 월의 첫 번째 날과 같은 요일. 해당 월의 첫 번째 날에 대해 4번째 요일의 12시간을 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 2000-06-21-12.12.30.000000 결과: 2000-06-22-00.00.00.000000	입력 값: 2000-06-21-12.12.30.000000 결과: 2000-06-15-00.00.00.000000
DDD DD J	Day 해당 일의 12시간을 반올림합니다. TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 2000-05-17-12.59.59.000000 결과: 2000-05-18-00.00.00.000000	입력 값: 2000-05-17-12.59.59.000000 결과: 2000-05-17-00.00.00.000000
DAY DY D	요일 시작. 해당 주의 4번째 날의 12시간을 반올림합니다. 첫 번째 요일은 로케일을 기반으로 합니다 (*locale-name* 참조). TIME 인수에 대해서는 유효하지 않습니다.	입력 값: 2000-05-17-12.59.59.000000 결과: 2000-05-21-00.00.00.000000	입력 값: 2000-05-17-12.59.59.000000 결과: 2000-05-14-00.00.00.000000
HH HH12 HH24	Hour 30분을 반올림합니다.	입력 값: 2000-05-17-23.59.59.000000 결과: 2000-05-18-00.00.00.000000	입력 값: 2000-05-17-23.59.59.000000 결과: 2000-05-17-23.00.00.000000
MI	Minute 30초를 반올림합니다.	입력 값: 2000-05-17-23.58.45.000000 결과: 2000-05-17-23.59.00.000000	입력 값: 2000-05-17-23.58.45.000000 결과: 2000-05-17-23.58.00.000000
SS	Second 1/2초를 반올림합니다.	입력 값: 2000-05-17-23.58.45.500000 결과: 2000-05-17-23.58.46.000000	입력 값: 2000-05-17-23.58.45.500000 결과: 2000-05-17-23.58.45.000000

주: 582 페이지의 표 55의 형식 요소는 대문자로 지정해야 합니다.

값의 시간 부분에 적용되는 형식 요소가 날짜 인수에 지정되면, 날짜 인수는 변경되지 않고 리턴됩니다. 시간 인수에 대해 유효하지 않은 형식 요소가 시간 인수에 대해 지정되면, 오류가 리턴됩니다(SQLSTATE 22007).

주

- **결정론:** ROUND는 결정 함수입니다. 그러나 다음 함수 호출은 CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
  - *locale-name*이 명시적으로 지정되지 않고 다음 중 하나가 참인 경우 날짜 시간 값의 반올림:
    - *format-string*이 상수가 아님
    - *format-string*이 상수이고 로케일 구분 형식 요소를 포함

    특수 레지스터를 사용할 수 없는 경우에는 특수 레지스터 값에 종속된 함수의 호출을 사용할 수 없습니다.

예:

- 각각 2, 1, 0, -1, -2, -3 및 -4 소수 자리로 반올림된 873.726의 값을 계산하십시오.

```
VALUES (
  ROUND(873.726, 2),
  ROUND(873.726, 1),
  ROUND(873.726, 0),
  ROUND(873.726,-1),
  ROUND(873.726,-2),
  ROUND(873.726,-3),
  ROUND(873.726,-4) )
```

  이 예에서는 다음을 리턴합니다.

```
1         2         3         4         5         6         7
--------- --------- --------- --------- --------- --------- ---------
   873.730   873.700   874.000   870.000   900.000  1000.000     0.000
```

- 양수와 음수를 모두 사용하여 계산하십시오.

```
VALUES (
  ROUND(3.5, 0),
  ROUND(3.1, 0),
  ROUND(-3.1, 0),
  ROUND(-3.5,0) )
```

  이 예에서는 다음을 리턴합니다.

```
1    2    3    4
---- ---- ---- ----
  4.0  3.0 -3.0 -4.0
```

- 세 자릿수로 반올림되는 10진수 부동 소수점 숫자 3.12350을 계산합니다.

```
VALUES (
  ROUND(DECFLOAT('3.12350'), 3))
```

이 예에서는 다음을 리턴합니다.

```
1
-----
3.12400
```

- 호스트 변수 RND_DT를 가장 가까운 월 값으로 반올림되는 입력 날짜로 설정합니다.

  ```
  SET :RND_DATE = ROUND(DATE('2000-08-16'), 'MONTH');
  ```

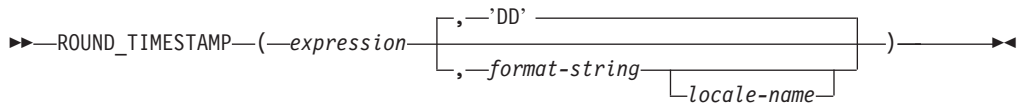
  값 설정은 2000-09-01입니다.

- 호스트 변수 RND_TMSTMP를 가장 가까운 연도 값으로 반올림되는 입력 시간소인으로 설정합니다.

  ```
  SET :RND_TMSTMP = ROUND(TIMESTAMP('2000-08-14-17.30.00'),
                          'YEAR');
  ```

  값 설정은 2001-01-01-00.00.00.000000입니다.

ROUND_TIMESTAMP



스키마는 SYSIBM입니다.

ROUND_TIMESTAMP 스칼라 함수는 *format-string*에 지정된 단위로 라운드된 *expression*인 시간소인을 리턴합니다. *format-string*을 지정하지 않은 경우 *expression*은 *format-string*에 대해 'DD'가 지정된 것처럼 가장 가까운 날로 라운드됩니다.

expression

내장 데이터 유형인 날짜 또는 시간소인 중 하나의 값을 리턴하는 표현식입니다.

format-string

실제 길이가 254바이트를 넘지 않는 내장 문자열 데이터 유형을 리턴하는 표현식입니다. *format-string*의 형식 요소는 *expression*을 라운드하는 방법을 지정합니다. 예를 들어, format-string이 'DD'이면, *expression*으로 표시되는 시간소인은 가장 가까운 날로 라운드됩니다. 앞뒤 공백은 문자열에서 제거되며 결과 서브스트링은 시간소인에 유효한 형식 요소여야 합니다(SQLSTATE 22007). 디폴트는 'DD'입니다.

*format-string*에 대해 허용 가능한 값은 ROUND 함수 설명에 있는 형식 요소 테이블에 나열되어 있습니다.

locale-name

형식 모델 값 DAY, DY 또는 D를 사용할 때 첫 번째 요일을 판별하기 위해 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자가 구분되지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 해당되는 이름 지정 방법에 대한 정보는 ″SQL 및 XQuery의 로케일 이름″을 참조하십시오. *locale-name*을 지정하지 않은 경우 특수 레지스터 CURRENT LOCALE LC_TIME의 값이 사용됩니다.

함수의 결과는 다음 시간소인 정밀도가 있는 TIMESTAMP입니다.

- *expression*의 데이터 유형이 TIMSTAMP(*p*)이면 *p*이고
- *expression*의 데이터 유형이 DATE이면 0이며
- 그렇지 않으면 6입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

주

- **결정론:** ROUND_TIMESTAMP는 결정 함수입니다. 그러나 다음 함수 호출은 CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
  - *locale-name*이 명시적으로 지정되지 않고 다음 중 하나가 참인 경우 날짜 또는 시간소인 값의 반올림:
    - *format-string*이 상수가 아님
    - *format-string*이 상수이고 로케일 구분 형식 요소를 포함

    특수 레지스터의 값에 종속되는 함수의 호출은 특수 레지스터를 사용할 수 없는 경우 사용할 수 없습니다.

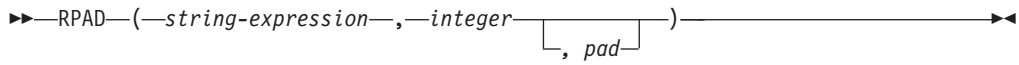
예 :

- 호스트 변수 *RND_TMSTMP*를 가장 가까운 연도 값으로 라운드된 입력 시간소인으로 설정하십시오.

```
SET :RND_TMSTMP = ROUND_TIMESTAMP('2000-08-14-17.30.00', 'YEAR');
```

  값 세트는 2001-01-01-00.00.00.000000입니다.

RPAD



스키마는 SYSIBM입니다.

RPAD 함수는 오른쪽이 *패*드 또는 공백으로 채워지는 *string-expression* 구성 문자열을 리턴합니다. RPAD 함수는 *string-expression*의 앞 또는 뒤 공백을 의미있는 것으로 처리합니다. *string-expression*의 실제 길이가 *integer*보다 작고 *pad*가 비어 있는 문자열이 아닌 경우에만 채우기가 발생합니다.

string-expression
소스 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

integer
결과 길이를 지정하는 정수 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. 값은 *n* 이하의 0 또는 양의 정수여야 합니다. 여기서 *n*은 *string-expression*이 문자열인 경우 32 672이고 *string-expression*이 그래픽 문자열인 경우 16 336입니다.

pad
채울 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

*pad*를 지정하지 않는 경우 패드 문자는 다음과 같이 판별됩니다.

- *string-expression*이 문자열인 경우 SBCS 공백 문자
- *string-expression*이 그래픽 문자열인 경우 표의 문자에서의 공백 문자. EUC 데이터베이스의 그래픽 문자열의 경우 X'3000'이 사용됩니다. 유니코드 데이터베이스의 그래픽 문자열의 경우 X'0020'이 사용됩니다.

함수의 결과는 코드 페이지가 *string-expression*과 같은 가변 길이 문자열입니다. *string-expression*의 값과 *pad*의 값은 호환 가능한 데이터 유형을 가지고 있어야 합니다. *string-expression* 및 *pad*가 다른 코드 페이지를 가지고 있는 경우 *pad*는

*string-expression*의 코드 페이지로 변환됩니다. *string-expression* 또는 *pad* 중 하나가 FOR BIT DATA인 경우 어떤 문자 변환도 발생하지 않습니다.

결과의 길이 속성은 *integer*의 값이 함수 호출을 포함하는 SQL문이 컴파일될 때 사용 가능한지(예를 들어, 상수 또는 상수 표현식으로 지정된 경우) 또는 함수가 실행될 때만 사용 가능한지(예를 들어, 함수 호출 결과로 지정된 경우) 여부에 따라 다릅니다. 값이 함수 호출을 포함하는 SQL문이 컴파일될 때 사용 가능하면, *integer*가 0보다 클 경우 결과의 길이 속성은 *integer*입니다. *integer*가 0일 경우 결과의 길이 속성은 1입니다. 값이 함수가 실행될 때만 사용 가능하면, 결과의 길이 속성은 다음 테이블에 따라 판별됩니다.

표 56. 함수가 실행될 때만 정수가 사용 가능한 경우의 결과 길이 판별

*string-expression*의 데이터 유형	결과 데이터 유형 길이
CHAR(*n*) 또는 VARCHAR(*n*)	*n*+100 및 32 672 중 최소
GRAPHIC(*n*) 또는 VARGRAPHIC(*n*)	*n*+100 및 16 336 중 최소

결과의 실제 길이는 *integer*를 통해 판별됩니다. *integer*가 0이고 실제 길이가 0이면 결과는 비어 있는 결과 문자열입니다. *integer*가 *string-expression*의 실제 길이보다 작으면 실제 길이는 *integer*이므로 결과는 절단됩니다.

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예를 들면, 다음과 같습니다.

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 다음 쿼리는 값의 오른쪽을 마침표로 완전히 채웁니다.

  ```
  SELECT RPAD(NAME,15,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  ---------------
  Chris..........
  Meg............
  Jeff...........
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 다음 쿼리는 값의 오른쪽을 *패드*로 완전히 채웁니다(어떤 경우에는 채우기 스펙의 부분 인스턴스가 있음).

  ```
  SELECT RPAD(NAME,15,'123' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

```
NAME
---------------
Chris1231231231
Meg123123123123
Jeff12312312312
```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 다음 쿼리는 각 값을 5 길이가 되도록 채웁니다.

  ```
  SELECT RPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  -----
  Chris
  Meg..
  Jeff.
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. RTRIM의 결과는 공백이 제거되는 가변 길이 문자열입니다.

  ```
  SELECT RPAD(RTRIM(NAME),15,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  ---------------
  Chris..........
  Meg............
  Jeff...........
  ```

- NAME이 『Chris』, 『Meg』 및 『Jeff』 값을 포함하는 VARCHAR(15) 컬럼이라고 가정하십시오. 『Chris』는 절단되고 『Meg』는 채워지며 『Jeff』는 변경되지 않습니다.

  ```
  SELECT RPAD(NAME,4,'.' ) AS NAME FROM T1;
  ```

  다음을 리턴합니다.

  ```
  NAME
  ----
  Chri
  Meg.
  Jeff
  ```

RTRIM

스키마는 SYSIBM입니다. (이 함수의 SYSFUN 버전은 LONG VARCHAR 및 CLOB 인수를 지원하여 계속 사용 가능합니다.)

RTRIM 함수는 *string-expression*의 끝에서 공백을 제거합니다.

표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

- 인수가 DBCS 또는 EUC 데이터베이스에 있는 그래픽 문자열인 경우 뒤에 있는 2 바이트 공백이 제거됩니다.
- 인수가 유니코드 데이터베이스에 있는 그래픽 문자열인 경우 뒤에 있는 UCS-2 공백이 제거됩니다.
- 그렇지 않으면 뒤에 있는 1바이트 공백이 제거됩니다.

함수의 결과 데이터 유형은 다음과 같습니다.

- *string-expression*의 데이터 유형이 VARCHAR 또는 CHAR이면 VARCHAR입니다.
- *string-expression*의 데이터 유형이 VARGRAPHIC 또는 GRAPHIC이면 VARGRAPHIC입니다.

리턴된 유형의 길이 매개변수는 인수 데이터 유형의 길이 매개변수와 동일합니다.

문자열에 대한 결과의 실제 길이는 *string-expression*의 길이에서 공백 문자로 제거된 바이트 수를 뺀 것입니다. 그래픽 문자열에 대한 결과의 실제 길이는 *string-expression*의 길이(2바이트 문자의 수로 표시)에서 제거된 2바이트 공백 문자의 수를 뺀 것입니다. 모든 문자가 제거되는 경우 결과는 공백의 가변 길이 문자열(길이는 0)입니다.

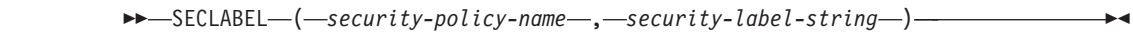
인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예: 호스트 변수 HELLO가 CHAR(9)로 정의되며 'Hello'의 값을 가진다고 가정하십시오.

```
VALUES RTRIM(:HELLO)
```

결과는 'Hello'입니다.

SECLABEL



스키마는 SYSIBM입니다.

SECLABEL 함수는 데이터 유형 DB2SECURITYLABEL을 갖는 이름 지정되지 않음 보안 레이블을 리턴합니다. 이름 지정된 보안 레이블을 작성할 필요없이 주어진 구성요소 값을 갖는 보안 레이블을 삽입하려면 SECLABEL 함수를 사용하십시오.

security-policy-name

현재 서버에 존재하는 보안 규정을 지정하는 문자열입니다(SQLSTATE 42704). 문자열은 문자, 그래픽 문자열 상수 또는 호스트 변수여야 합니다.

security-label-string

*security-policy-name*으로 이름 지정된 보안 규정에 대한 보안 레이블의 유효한 표시를 리턴하는 표현식입니다(SQLSTATE 4274I). 표현식은 내장 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다.

예를 들면, 다음과 같습니다.

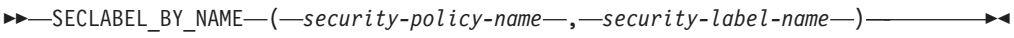
- 다음 명령문은 보안 규정 CONTRIBUTIONS로 보호되는 REGIONS 테이블에 행을 삽입합니다. 삽입될 행의 보안 레이블은 SECLABEL 함수에 의해 제공됩니다. CONTRIBUTIONS 보안 규정에는 두 개의 구성요소가 있습니다. 주어진 보안 레이블은 첫 번째 구성요소에 대한 LIFE MEMBER 요소, 두 번째 구성요소에 대한 BLUE 및 YELLOW 요소를 갖습니다.

```
INSERT INTO REGIONS
  VALUES (SECLABEL('CONTRIBUTIONS', 'LIFE MEMBER:(BLUE,YELLOW)'),
    1, 'Northeast')
```

- 다음 명령문은 세 개의 구성요소를 갖는 TS_SECPOLICY 보안 규정으로 보호되는 CASE_IDS 테이블에 행을 삽입합니다. 보안 레이블은 SECLABEL 함수에 의해 제공됩니다. 삽입되는 보안 레이블에는 첫 번째 구성요소에 대한 HIGH PROFILE 요소, 두 번째 구성요소에 대한 비어 있는 값, 세 번째 구성요소에 대한 G19 요소가 있습니다.

```
INSERT INTO CASE_IDS
  VALUES (SECLABEL('TS_SECPOLICY', 'HIGH PROFILE:():G19') , 3, 'KLB')
```

SECLABEL_BY_NAME



스키마는 SYSIBM입니다.

SECLABEL_BY_NAME 함수는 지정된 보안 레이블을 리턴합니다. 리턴된 보안 레이블은 DB2SECURITYLABEL 데이터 유형을 갖습니다. 이름 지정된 보안 레이블을 삽입하려면 이 함수를 사용하십시오.

security-policy-name

현재 서버에 존재하는 보안 규정을 지정하는 문자열입니다(SQLSTATE 42704). 문자열은 문자, 그래픽 문자열 상수 또는 호스트 변수여야 합니다.

security-label-name

*security-policy-name*으로 이름 지정되는 보안 규정에 대한 현재 서버에 존재하는 보안 레이블의 이름을 리턴하는 표현식입니다(SQLSTATE 4274I). 표현식은 내장 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다.

예를 들면, 다음과 같습니다.

- 사용자 Tina가 CONTRIBUTIONS 보안 규정으로 보호되는 REGIONS 테이블에 행을 삽입하려고 시도합니다. Tina는 행이 EMPLOYEESECLABEL 보안 레이블로 보호되기 원합니다. 이 명령문은 CONTRIBUTIONS.EMPLOYEESECLABEL이 알 수 없는 ID이기 때문에 실패합니다.

  ```
  INSERT INTO REGIONS
  VALUES (CONTRIBUTIONS.EMPLOYEESECLABEL, 1, 'Southwest')   -- incorrect
  ```

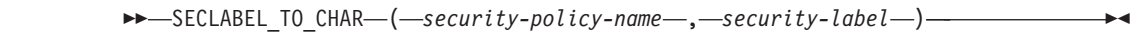
  이 명령문은 첫 번째 값이 문자열이고 데이터 유형 DB2SECURITYLABEL을 갖지 않기 때문에 실패합니다.

  ```
  INSERT INTO REGIONS
  VALUES ('CONTRIBUTIONS.EMPLOYEESECLABEL', 1, 'Southwest') -- incorrect
  ```

  이 명령문은 SECLABEL_BY_NAME 함수가 데이터 유형이 DB2SECURITYLABEL인 보안 레이블을 리턴하므로 성공합니다.

  ```
  INSERT INTO REGIONS
  VALUES (SECLABEL_BY_NAME('CONTRIBUTIONS', 'EMPLOYEESECLABEL'),
    1, 'Southwest')                                        -- correct
  ```

SECLABEL_TO_CHAR



스키마는 SYSIBM입니다.

SECLABEL_TO_CHAR 함수는 보안 레이블을 승인하고 보안 레이블의 모든 요소를 포함하는 문자열을 리턴합니다. 문자열은 보안 레이블 문자열 형식입니다.

security-policy-name

현재 서버에 존재하는 보안 규정을 지정하는 문자열입니다(SQLSTATE 42704). 문자열은 문자, 그래픽 문자열 상수 또는 호스트 변수여야 합니다.

security-label

*security-policy-name*으로 이름 지정된 보안 규정에 대해 유효한 보안 레이블 값을 리턴하는 표현식입니다(SQLSTATE 4274I). 표현식은 내장 SYSPROC.DB2SECURITYLABEL 구별 유형인 값을 리턴해야 합니다.

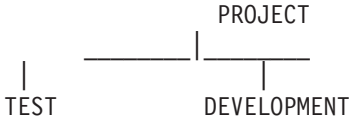
주

- 명령문의 권한 부여 ID가 DB2SECURITYLABEL 데이터 유형의 컬럼에서 읽는 보안 레이블에 대해 이 함수를 실행할 경우, 함수 출력이 해당 권한 부여 ID의 LBAC 증명서에 의해 영향을 받을 수도 있습니다. 이 경우에 권한 부여 ID가 요소에 대한 읽기 액세스를 갖지 않는 경우 해당 요소는 출력에 포함되지 않습니다. 권한 부여 ID는 LBAC 증명서가 해당 요소만 포함하고 다른 것은 포함하지 않는 보안 레이블로 보호된 데이터를 읽도록 허용하는 경우 요소에 대한 읽기 액세스를 갖습니다.

  규칙 세트 DB2LBACRULES의 경우 TREE 유형의 구성요소만 사용자에게 읽기 액세스가 없는 요소를 포함할 수 있습니다. 기타 유형의 구성요소의 경우 요소 중 하나가 읽기 액세스를 차단하는 경우 행을 전혀 읽을 수 없습니다. 따라서 트리 유형의 구성요소만 이 방식으로 제외된 요소를 갖습니다.

예:

- EMP 테이블에 두 컬럼 RECORDNUM 및 LABEL이 있고, RECORDNUM은 데이터 유형 INTEGER를 갖고, LABEL은 DB2SECURITYLABEL 유형을 갖습니다. 테이블 EMP는 보안 규정 DATA_ACCESSPOLICY로 보호되며, 이는 DB2LBACRULES 규칙 세트를 사용하고 구성요소를 하나만(TREE 유형의 GROUPS) 갖습니다. GROUPS에는 PROJECT, TEST, DEVELOPMENT, CURRENT 및 FIELD라는 5개의 요소가 있습니다. 다음 다이어그램은 해당 요소의 서로에 대한 관계를 나타냅니다.



EMP 테이블에 다음 데이터가 들어있습니다.

```
RECORDNUM  LABEL
---------  ----------------
        1  PROJECT
        2  (TEST, FIELD)
        3  (CURRENT, FIELD)
```

사용자 ID가 Djavan인 사용자는 DEVELOPMENT 요소만 포함하는 읽기에 대한 보안 레이블을 보유합니다. 이것은 Djavan이 DEVELOPMENT, CURRENT 및 FIELD 요소에 대한 읽기 액세스를 가짐을 의미합니다.

```
SELECT RECORDNUM, SECLABEL_TO_CHAR('DATA_ACCESSPOLICY', LABEL) FROM EMP
```

다음을 리턴합니다.

```
RECORDNUM  LABEL
---------  ----------------
        2  FIELD
        3  (CURRENT, FIELD)
```

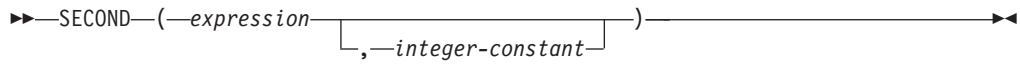
Djavan의 LBAC 증명서가 해당 행을 읽도록 허용하지 않기 때문에 RECORDNUM 값 1을 갖는 행은 출력에 포함되지 않습니다. RECORDNUM 값 2를 갖는 행에서, Djavan이 해당 요소에 대한 읽기 액세스가 없기 때문에 TEST 요소가 출력에 포함되지 않습니다. Djavan은 TEST가 보안 레이블의 유일한 요소인 경우 행을 전혀 액세스할 수 없었습니다. Djavan이 CURRENT 및 FIELD 요소에 대한 읽기 액세스를 갖기 때문에 두 요소가 모두 출력에 나타납니다.

이제 Djavan에게 DB2LBACREADTREE 규칙에 대한 면제가 권한 부여됩니다. 이는 TREE 유형 구성요소의 요소가 읽기 액세스를 차단하지 않음을 의미합니다. 동일한 쿼리가 다음을 리턴합니다.

```
RECORDNUM  LABEL
---------  ----------------
        1  PROJECT
        2  (TEST, FIELD)
        3  (CURRENT, FIELD)
```

면제가 Djavan에게 모든 요소에 대한 읽기 액세스를 부여하기 때문에 현재 출력은 모든 행과 모든 요소를 포함합니다.

SECOND



스키마는 SYSIBM입니다.

SECOND 함수는 값의 초 부분을 선택적 소수 초로 리턴합니다.

expression

날짜, 시간, 시간소인, 시간 지속 기간, 시간소인 지속 기간, CLOB 또는 DBCLOB가 아닌 날짜, 시간 또는 시간소인의 유효한 문자열 표현인 값을 리턴하는 표현식입니다.

*expression*이 날짜 또는 날짜의 유효한 문자열 표현인 경우, 정확한 자정의 시간(00.00.00)을 가정하여 TIMESTAMP(0)로 먼저 변환됩니다.

유니코드 데이터베이스만이 날짜, 시간 또는 시간소인의 그래픽 문자열 표현인 인수를 지원합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

*expression*이 문자열인 경우, 값이 날짜 시간 값으로 변환되기 전에 앞에 공백이 포함되며 뒤 공백이 제거됩니다.날짜 시간 값의 문자열 표현의 유효한 포맷에 대해서는, 『날짜 시간 값』의 『날짜 시간 값의 문자열 표현』을 참조하십시오.

integer-constant

소수 초에 대해 스케일을 표현하는 정수 상수. 값의 범위는 0 - 12 내에 있어야 합니다.

단일 인수인 함수의 결과는 큰 정수(integer)입니다. 두 개의 인수가 있는 함수의 결과는 DECIMAL(2+*s*,*s*)이며 여기서 *s*는 *integer-constant*의 값입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

첫 번째 인수의 데이터 유형과 인수에 따라 다른 규칙이 적용됩니다.

- 첫 번째 인수가 날짜, 시간, 시간소인이거나, 날짜, 시간 또는 시간소인의 유효한 문자열 표현인 경우:
  - 하나의 인수만 지정되면 결과는 값의 초 파트가 됩니다(0과 59).
  - 두 인수가 지정되면, 결과는 값의 초 파트(0 - 59)와 적용 가능한 값의 소수 초 파트의 *integer-constant* 숫자가 됩니다. 값의 소수 초가 없으면, 영(0)이 리턴됩니다.
- 첫 번째 인수가 시간 지속 기간이거나 시간소인 지속 기간인 경우:
  - 하나의 인수만 지정되면, 결과는 값의 초 파트입니다(-99 - 99).

– 두 인수가 지정되면, 결과는 값의 초 파트(-99 - 99)와 적용 가능한 값의 소수 초 파트의 *integer-constant* 숫자가 됩니다. 값의 소수 초가 없으면, 영(0)이 리턴됩니다.0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- 호스트 변수 TIME_DUR(decimal(6,0))의 값을 153045로 가정하십시오.

```
SELECT SECOND(:TIME_DUR)
  FROM SYSIBM.SYSDUMMY1
```

  값 45를 리턴합니다.

- RECEIVED 컬럼(시간소인)에 '1988-12-25-17.12.30.000000'에 해당되는 내부 값을 가진다고 가정하십시오.

```
SELECT SECOND(RECEIVED)
  FROM IN_TRAY
```

  값 30을 리턴합니다.

- 밀리초의 현재 시간소인에서 소수 초로 시간(초)을 가져옵니다.

```
SELECT SECOND (CURRENTTIMESTAMP(3), 3)
  FROM SYSIBM.SYSDUMMY1
```

  54.321과 같은 현재 시간소인을 기반으로 DECIMAL(5,3) 값을 리턴합니다.

SIGN

스키마는 SYSIBM입니다. (SIGN 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 부호 표시자를 리턴합니다. 인수가 0보다 작으면, -1이 리턴되고, 인수가 -0의 부동 소수점 값이면, -0의 10진수 부동 소수점 값이 리턴됩니다. 0과 같으면 0이 리턴되며, 0보다 크면 1이 리턴됩니다.

인수는 모든 유형의 내장 숫자 데이터가 될 수 있습니다. 인수는 함수가 처리할 수 있도록 DECIMAL 및 REAL 값을 배정밀도 부동 소수점 숫자로 변환합니다.

함수의 결과는 다음과 같습니다.

- 인수가 SMALLINT이면 SMALLINT입니다.
- 인수가 INTEGER이면 INTEGER입니다.
- 인수가 BIGINT이면 BIGINT입니다.
- 인수가 DECFLOAT(*n*)인 경우 DECFLOAT(*n*)
- 그렇지 않으면 DOUBLE입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

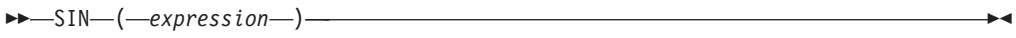
예:

- 호스트 변수 PROFIT이 값이 50000인 큰 정수임을 가정하십시오.

```
VALUES SIGN(:PROFIT)
```

  값 1을 리턴합니다.

SIN



스키마는 SYSIBM입니다. (SIN 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 sine을 리턴합니다. 여기서, 인수는 라디안으로 표시되는 각도입니다.

인수는 어떤 내장 숫자 데이터 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

SINH

스키마는 SYSIBM입니다.

인수의 hyperbolic sine을 리턴합니다. 여기서 인수는 라디안으로 표시되는 각도입니다.

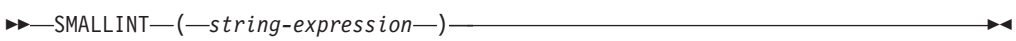
인수는 모든 내장 숫자 데이터 유형(DECFLOAT 제외)일 수 있습니다. 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

SMALLINT

숫자를 Smallint로:

문자열을 Smallint로:



스키마는 SYSIBM입니다.

SMALLINT 함수는 다음의 작은 정수 표현을 리턴합니다.

- 숫자
- 숫자의 문자열 표현

숫자를 Smallint로:

numeric-expression

임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다.

인수가 작은 정수 컬럼이나 변수에 지정된 경우, 결과는 같은 숫자입니다. 인수의 소수 파트가 절단됩니다. 인수의 전체 부분이 작은 정수 범위 내에 없으면 오류가 리턴됩니다(SQLSTATE 22003).

문자열을 Smallint로:

string-expression

문자 상수의 최대 길이보다 길지 않은 길이의 숫자의 문자열 또는 유니코드 그래픽 문자열 표현인 값을 리턴하는 표현식입니다.

결과는 CAST(*string-expresssion* AS SMALLINT)에서 발생하는 수와 동일합니다. 앞뒤 공백은 제거되고 결과 문자열은 정수, 10진수, 부동 소수점이나 10진수 부동 소수점 상수에 대한 규칙에 따라야 합니다(SQLSTATE 22018). 인수의 전체 부분이 작은 정수 범위 내에 없으면, 오류가 리턴됩니다(SQLSTATE 22003). *string-expression*의 데이터 유형은 CLOB 또는 DBCLOB일 수 없습니다(SQLSTATE 42884).

함수의 결과는 작은 정수입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주: CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예 :

EMPLOYEE 테이블을 사용하여 교육 수준(EDLEVEL)별로 나눠진 급여(SALARY)를 포함하는 목록을 선택하십시오. 계산 시 소수 부분은 절단됩니다. 목록에는 계산과 직원 번호(EMPNO)에 사용되는 값도 포함되어야 합니다.

```
SELECT SMALLINT(SALARY / EDLEVEL), SALARY, ESDLEVEL, EMPNO
  FROM EMPLOYEE
```

SOUNDEX

스키마는 SYSFUN입니다.

인수에서 단어들의 음조를 나타내는 4문자 코드를 리턴합니다. 결과는 다른 문자열의 음조와 비교하는 데 사용할 수 있습니다.

인수는 4000바이트를 초과하지 않는 CHAR 또는 VARCHAR 문자열이 될 수 있습니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다. 함수는 UTF-8로 인코드된 경우에도 ASCII 문자인 것처럼 전달된 데이터를 해석합니다.

함수의 결과는 CHAR(4)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

SOUNDEX 함수는 음조가 알려져 있지만 정확한 철자가 아닌 문자열을 찾는 데 유용합니다. 유사한 음조의 단어를 검색하는 데 도움이 될 수 있는 문자 및 문자 조합 음조 방법에 대해 가정합니다. 직접 비교하거나 인수로 문자열을 DIFFERENCE 함수에 전달하여 비교할 수 있습니다.

예:

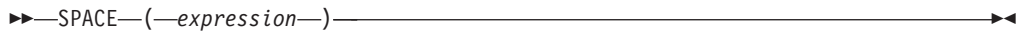
EMPLOYEE 테이블을 사용하여 'Loucesy' 음조의 성을 가지고 있는 직원의 EMPNO 및 LASTNAME을 찾으십시오.

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
  WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

이 예에서는 다음을 리턴합니다.

```
EMPNO  LASTNAME
------ ---------------
000110 LUCCHESSI
```

SPACE



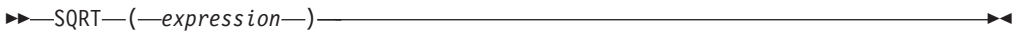
스키마는 SYSFUN입니다.

인수에 길이가 지정된 공백으로 구성되는 문자열을 리턴합니다.

인수는 SMALLINT 또는 INTEGER입니다.

함수의 결과는 VARCHAR(4000)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

SQRT



스키마는 SYSIBM입니다. (SQRT 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

SQRT 함수는 숫자의 제곱근을 리턴합니다.

인수는 내장 숫자 데이터 유형 값을 리턴하는 표현식이어야 합니다. 인수가 10진수 부동 소수점인 경우 연산은 10진수 부동 소수점으로 수행됩니다. 기타의 경우 인수는 배정밀도 부동 소수점 숫자로 변환되어 함수가 처리됩니다.

인수가 DECFLOAT(*n*)인 경우 결과는 DECFLOAT(*n*)입니다. 기타의 경우 결과는 배정밀도 부동 소수점입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

주

- ***DECFLOAT 특수 값에 연관된 결과:*** 인수가 특수한 10진수의 부동 소수점 값인 경우 10진수 부동 소수점에 대해 일반 산술 연산 규칙이 적용됩니다. 243 페이지의 『10진수 부동 소수점에 대한 일반 산술 연산 규칙』에서 『10진수 부동 소수점에 대한 일반 산술 연산 규칙』을 참조하십시오.

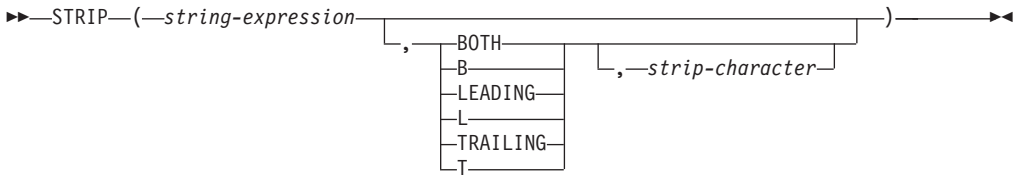
예

- SQUARE가 값 9.0인 DECIMAL(2,1) 호스트 변수라고 가정해 보십시오.

```
VALUES SQRT(:SQUARE)
```

  근사 값 3.00을 리턴합니다.

STRIP



스키마는 SYSIBM입니다. 함수 시그니처에서 키워드를 사용할 때 함수 이름을 규정된 이름으로 지정할 수 없습니다.

STRIP 함수는 문자열 표현식의 끝이나 시작에서 다른 지정 문자 또는 공백을 제거합니다.

STRIP 함수는 TRIM 스칼라 함수와 동일합니다.

string-expression
결과가 파생될 문자열을 지정하는 표현식입니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

BOTH, LEADING, 또는 TRAILING
문자열 표현식의 시작, 끝 또는 양쪽 끝에서 문자열을 제거하는지 여부를 지정합니다. 이 인수를 지정하지 않으면 문자열의 시작과 끝에서 문자를 제거합니다.

strip-character
제거할 문자를 지정하는 단일 문자 상수입니다. *strip-character*는 UTF-32 인코딩이 단일 문자 또는 단일 숫자 값인 어떤 문자도 가능합니다. 문자의 2진 표시가 일치됩니다.

*strip-character*를 지정하지 않은 경우,

- *string-expression*이 DBCS 그래픽 문자열이면 디폴트 *strip-character*는 DBCS 공백이고 해당 코드 포인트는 데이터베이스 코드 페이지에 따라 결정됩니다.
- *string-expression*이 UCS-2 그래픽 문자열이면 디폴트 *strip-character*는 UCS-2 공백(X'0020')입니다.
- 그렇지 않으면 디폴트 *strip-character*는 SBCS 공백(X'20')입니다.

결과는 *string-expression*의 길이 속성과 동일한 최대 길이를 갖는 가변 길이 문자열이 됩니다. 결과의 실제 길이는 *string-expression* 길이에서 제거할 바이트 수를 뺀 것입니다. 모든 문자를 제거하면 결과는 빈 가변 길이 문자열이 됩니다. 결과의 코드 페이지는 *string-expression*의 코드 페이지와 같습니다.

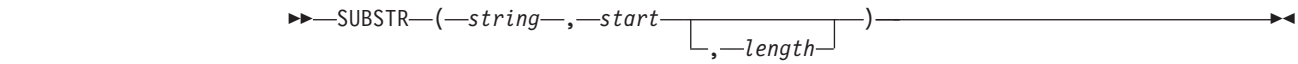
예:

- 유형이 CHAR(9)인 호스트 변수 BALANCE의 값이 '000345.50'이라고 가정하십시오.

```
SELECT STRIP(:BALANCE, LEADING, '0'),
FROM SYSIBM.SYSDUMMY1
```

리턴되는 값은 '345.50'입니다.

SUBSTR



SUBSTR 함수는 문자열의 부속 문자열을 리턴합니다.

*string*이 문자열이면 함수의 결과는 첫 번째 인수의 코드 페이지에서 표시되는 문자열이고, 실행 파일 문자열이면 함수의 결과는 실행 파일 문자열이며, 그래픽 문자열이면 함수의 결과는 첫 번째 인수의 코드 페이지에서 표시되는 그래픽 문자열입니다. 첫 번째 인수가 호스트 변수일 경우 결과 코드 페이지는 데이터베이스 코드 페이지입니다. 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)이 됩니다.

string

결과가 파생될 문자열을 지정하는 표현식입니다.

표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아니면, 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *string*이 문자열이거나 실행 파일 문자열이면, *string*의 부속 문자열은 *string*의 0 이상의 연속 바이트이어야 합니다. *string*이 그래픽 문자열인 경우, *string*의 부속 문자열은 *string*의 0 이상의 연속된 2바이트 문자입니다.

start

문자열이나 실행 파일 문자열에 대한 결과의 첫 번째 바이트 위치 또는 그래픽 문자열에 대한 결과의 첫 번째 문자 위치를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다.정수 값이 1과 *string*의 길이나 최대 길이 사이에 있어야 하며, *string*이 고정 길이 또는 가변 길이인지에 따라 다릅니다(범위 밖인 경우 SQLSTATE 22011). 이것은 응용프로그램 코드 페이지가 아니라 데이터베이스 코드 페이지의 구문에서 바이트 수로 지정해야 합니다.

length

결과 길이를 지정하는 표현식입니다. 지정된 경우, 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다.정수의 값이 0 - *n* 범위에 있어야 하며, 여기서 *n*은 (*string*의 길이 속성) - *start* + 1과 같습니다(범위 밖인 경우 SQLSTATE 22011).

*length*를 명시적으로 지정할 경우 *string*은 *string*의 지정된 부속 문자열이 항상 존재하도록 효율적으로 필요한 만큼의 공백(문자열의 경우 1바이트, 그래픽 문자열의 경우 2바이트) 또는 16진수 0 문자(BLOB 문자열을 위한)로 오른쪽에 채워집니다. *length*의 디폴트값은 문자열이나 실행 파일 문자열의 경우에 *start*에 지정된 바이

트에서 *string*의 마지막 바이트까지의 바이트 수이고, 그래픽 문자열의 경우엔 *start*에 지정된 문자에서 *string*의 마지막 문자까지의 2바이트 문자 수입니다. 그러나 *string*이 *start* 이하의 길이를 갖는 가변 길이 문자열인 경우, 디폴트값은 0이고 결과는 빈 문자열입니다. (이것은 응용프로그램 코드 페이지가 아니라, 데이터베이스 코드 페이지의 컨텍스트에서 바이트 수로 지정해야 합니다. 예를 들어, 데이터 유형이 VARCHAR(18)이고 값이 'MCKNIGHT'인 NAME 컬럼이 `SUBSTR(NAME,10)`에서 빈 문자열을 생성합니다.)

표 57에서는 입력 유형과 속성에 따라 SUBSTR 함수의 결과 유형과 길이를 나타냅니다.

표 57. 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 *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)
CLOB(A)	상수(*l*)	CLOB(*l*)
CLOB(A)	상수가 아니거나 지정되지 않음	CLOB(A)
GRAPHIC(A)	상수(*l*<128)	GRAPHIC(*l*)
GRAPHIC(A)	지정되지 않았으나 *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)
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)

참고: LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다.

*string*이 고정 길이 문자열일 경우 *length*를 생략하면 내재적으로 LENGTH(*string*) - *start* + 1이 지정됩니다. *string*이 가변 길이 문자열이면 *length*를 생략할 경우 0이나 LENGTH(*string*) - *start* + 1 중에서 큰 값이 내재적으로 지정됩니다.

예를 들면, 다음과 같습니다.

- 호스트 변수 NAME(VARCHAR(50))이 값 'BLUE JAY'를 갖고 호스트 변수 SURNAME_POS(int)가 값 6을 갖는다고 가정하십시오.

```
SUBSTR(:NAME, :SURNAME_POS)
```

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.

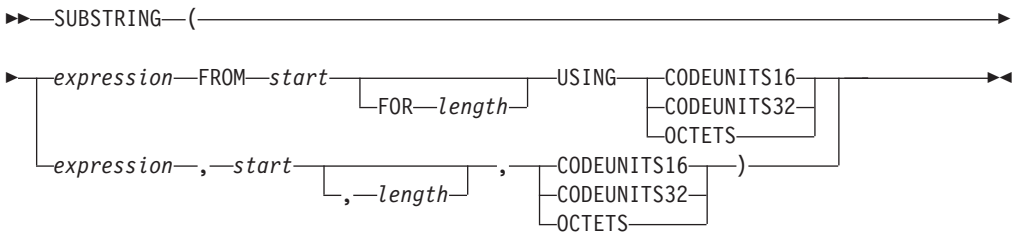
```
SELECT * FROM PROJECT
  WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

상수의 끝에 있는 공백은 'OPERATIONS'와 같이 초기 단어를 사용할 때 필요합니다.

주:

1. 동적 SQL에서 *string*, *start* 및 *length*는 매개변수 표시문자로 나타낼 수 있습니다. 매개변수 표시문자가 *string*에 사용된 경우 피연산자의 데이터 유형은 VARCHAR이고 해당 피연산자에 널(NULL)을 입력할 수 있습니다.
2. 위의 결과 정의에서 명시적으로 언급하지 않아도 *string*이 1바이트 및 복수 바이트의 혼합일 경우, 결과에는 *start* 및 *length* 값에 따라 복수 바이트 문자의 조각이 들어 있을 수 있다는 의미입니다. 즉, 결과는 2바이트 문자의 두 번째 바이트로 시작하거나, 2바이트 문자의 첫 번째 바이트로 끝날 수 있습니다. SUBSTR 함수는 이러한 조각을 발견하지 않고, 수행해야 하는 어떤 특수 처리도 제공하지 않습니다.

SUBSTRING



스키마는 SYSIBM입니다.

SUBSTRING 함수는 문자열의 하위 문자열을 리턴합니다.

expression

결과가 파생될 문자열을 지정하는 표현식입니다. 표현식은 내장 문자열, 숫자 또는 날짜 시간 데이터 유형인 값을 리턴해야 합니다. 값이 문자열 데이터 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *expression*이 문자열이면 결과도 문자열입니다. *expression*이 그래픽 문자열이면 결과도 그래픽 문자열입니다. *expression*이 실행 파일 문자열이면 결과도 실행 파일 문자열입니다.

*expression*의 하위 문자열은 *expression*의 0 이상의 인접 문자열 단위입니다.

시작

*expression*에서 결과의 첫 번째 문자열 단위가 될 위치를 지정하는 표현식입니다. 표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. 정수 값은 양수, 음수 또는 0이 될 수 있으며, 1 값은 결과의 첫 번째 문자열 단위가 *expression*의 첫 번째 문자열 단위임을 나타냅니다. OCTETS가 지정되고 *expression*이 그래픽 데이터일 경우 정수 값은 홀수가 되어야 하며 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 428GC).

length

표현식은 내장 숫자, CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형인 값을 리턴해야 합니다. 값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다.

*expression*이 고정 길이 문자열일 경우 *length*를 생략하면 내재적으로 CHARACTER_LENGTH(*expression* USING *string-unit*) - *start* + 1이 지정되고, 이는 *start*에서 *expression*의 마지막 위치까지의 *string units*(CODEUNITS16, CODEUNITS32 또는 OCTETS) 수입니다. *expression*이 가변 길이 문자열이고 *length*를 생략하면 내재적으로 0 또는 CHARACTER_LENGTH(*expression* USING *string-unit*) - *start* + 1 중에서 큰 값이 지정됩니다. 원하는 길이가 0일 경우 결과는 빈 문자열입니다.

값이 INTEGER 유형이 아닌 경우 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다. 값은 0보다 크거나 같아야 합니다. *n*보다 큰 값을 지정하면(여기서 *n*은 (*expression*의 length 속성) - *start* + 1), *n*이 결과 서브스트링의 길이로 사용됩니다. 값은 명시적으로 지정되는 단위로 표현됩니다. OCTETS가 지정되고 *expression*이 그래픽 데이터일 경우, 값은 짝수여야 합니다(SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 또는 OCTETS

start 및 *length*의 문자열 단위를 지정합니다. CODEUNITS16은 *start* 및 *length*가 16비트 UTF-16 코드 단위로 표시될 것을 지정합니다. CODEUNITS32는 *start* 및 *length*가 32비트 UTF-32 코드 단위로 표시될 것을 지정합니다. OCTETS는 *start* 및 *length*가 바이트 단위로 표시될 것을 지정합니다.

문자열 단위가 CODEUNITS16 또는 CODEUNITS32로 지정되고 *expression*이 실행 파일 문자열 또는 비트 데이터이면 오류가 리턴됩니다(SQLSTATE 428GC). 문자열 단위가 OCTETS로 지정되고 *expression*이 실행 파일 문자열이면 오류가 리턴됩니다(SQLSTATE 42815).

CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 『문자열』에서 『내장 함수의 문자열 단위』를 참조하십시오.

OCTETS를 사용하여 SUBSTRING 함수를 호출하고 *source-string*을 코드 포인트(혼합 또는 MBCS) 당 2바이트 이상이 필요한 코드 페이지로 인코드한 경우, SUBSTRING 조작은 다중 바이트 코드 포인트를 분할할 수 있으며 결과 하위 문자열은 부분 코드 포인트로 시작되거나 종료될 수 있습니다. 이러한 경우, 함수는 결과의 바이트 길이를 변경하지 않는 방법으로 앞 또는 뒤에 오는 부분 코드 포인트의 바이트를 공백으로 바꿉니다. (아래의 관련된 예를 참조하십시오).

결과의 길이 속성은 *expression*의 길이 속성과 동일합니다. 함수의 인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값이 됩니다. 결과는 문자로 채워지지 않습니다. *expression*의 실제 길이가 0이면 결과위 실제 길이도 0이 됩니다.

참고:

- 결과의 길이 속성은 입력 문자열 표현식의 길이 속성과 동일합니다. 이 동작은 SUBSTR 함수의 동작과 다릅니다. 여기서 길이 속성은 함수의 *start* 및 *length* 인수에서 파생됩니다.

예를 들면, 다음과 같습니다.

- FIRSTNAME은 테이블 T1의 VARCHAR(12) 컬럼입니다. 해당 값 중 하나는 6자 문자열 'Jürgen'입니다. FIRSTNAME이 이 값을 가질 때 리턴되는 값은 다음과 같습니다.

```
함수                                리턴 값
----------------------------------  ----------------------------------
SUBSTRING(FIRSTNAME,1,2,CODEUNITS32)  'Jü' -- x'4AC3BC'
SUBSTRING(FIRSTNAME,1,2,CODEUNITS16)  'Jü' -- x'4AC3BC'
```

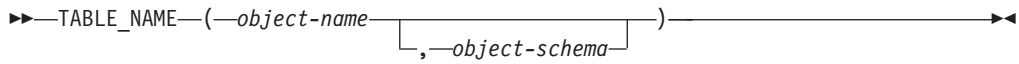
```
SUBSTRING(FIRSTNAME,1,2,OCTETS)       'J ' -- x'4A20'(절단 문자열)
SUBSTRING(FIRSTNAME,8,CODEUNITS16)    길이가 0인 문자열
SUBSTRING(FIRSTNAME,8,4,OCTETS)       길이가 0인 문자열
```

- 다음 예는 문자열 길이 단위가 OCTETS일 때 SUBSTRING이 앞 또는 뒤에 오는 부분 다중 바이트 코드 포인트를 공백으로 바꾸는 방법을 설명합니다. UTF8_VAR에 유니코드 문자열 '&N~AB'의 UTF-8 표시가 들어 있으며 여기서 '&'는 음악 기호 높은 음자리표이고 ~'는 조인 틸드 문자입니다.

  ```
  SUBSTRING(UTF8_VAR, 2, 5, OCTETS)
  ```

  세 개의 공백 바이트 다음에 'N'이 있고 그 뒤에 한 개의 공백 바이트가 있습니다.

TABLE_NAME



스키마는 SYSIBM입니다.

TABLE_NAME 함수는 별명 체인이 분석된 이후에 발견된 오브젝트의 규정되지 않은 이름을 리턴합니다. 지정된 *object-name*(및 *object-schema*)은 분석의 시작점으로 사용됩니다. 시작점이 별명을 참조하지 않은 경우, 규정되지 않은 시작점 이름이 리턴됩니다. 그 결과 이름은 테이블, 뷰 또는 정의되지 않은 오브젝트가 될 수 있습니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

object-name
분석될 규정되지 않은 이름(보통 기존 별명의 이름)을 나타내는 문자 표현식입니다. *object-name*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0바이트보다 크고 129바이트보다 작아야 합니다.

object-schema
분석에 앞서 제공된 *object-name* 값을 규정하는 데 사용된 스키마를 나타내는 문자 표현식입니다. *object-schema*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0바이트보다 크고 129바이트보다 작아야 합니다.

*object-schema*가 제공되지 않으면 규정자에 디폴트 스키마가 사용됩니다.

함수 결과의 데이터 유형은 VARCHAR(128)입니다. *object-name*이 널(NULL)이 될 수 있으면 결과도 널(NULL) 값이 될 수 있습니다. 즉, *object-name*이 널(NULL)이면 결과도 널(NULL) 값이 됩니다. *object-schema*가 널(NULL) 값이면 디폴트 스키마 이름이 사용됩니다. 결과는 규정되지 않은 이름을 나타내는 문자열입니다. 결과 이름은 다음 중 하나로 표시할 수 있습니다.

테이블 *object-name*의 값이 테이블 이름(입력 값이 리턴됨)이거나 해당 이름이 리턴되는 테이블에 대해 분석된 별명 이름입니다.

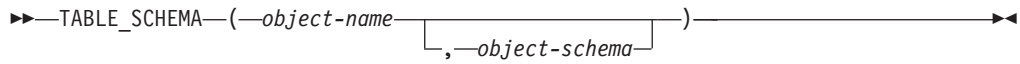
뷰 *object-name*의 값이 뷰 이름(입력 값이 리턴됨)이거나 해당 이름이 리턴되는 뷰에 대해 분석된 별명 이름입니다.

정의되지 않은 오브젝트
*object-name*의 값은 이름이 리턴된 정의되지 않은 오브젝트에 대해 분석된 별명 이름이거나 입력 값이 리턴된 정의되지 않은 오브젝트 이름 중 하나입니다.

그러므로 이 함수에 널(NULL)이 아닌 값이 제공되면 결과 이름을 갖고 있는 오브젝트가 없더라도 값이 항상 리턴됩니다.

주: TABLE_SCHEMA 및 TABLE_NAME 스칼라 함수 사용 시 코디네이터 파티션과 카탈로그 파티션 사이에 발생되는 불필요한 통신을 방지하여 파티션된 데이터베이스 구성의 성능을 개선하기 위해 BASE_TABLE 테이블 함수를 대신 사용할 수 있습니다.

TABLE_SCHEMA



스키마는 SYSIBM입니다.

TABLE_SCHEMA 함수는 별명 체인이 분석된 이후에 발견된 오브젝트의 스키마 이름을 리턴합니다. 지정된 *object-name*(및 *object-schema*)은 분석의 시작점으로 사용됩니다. 시작점이 별명을 참조하지 않은 경우 시작점의 스키마 이름이 리턴됩니다. 그 결과 스키마 이름은 테이블, 뷰 또는 정의되지 않은 오브젝트가 될 수 있습니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

object-name
분석될 규정되지 않은 이름(보통 기존 별명의 이름)을 나타내는 문자 표현식입니다. *object-name*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0바이트보다 크고 129바이트보다 작아야 합니다.

object-schema
분석에 앞서 제공된 *object-name* 값을 규정하는 데 사용된 스키마를 나타내는 문자 표현식입니다. *object-schema*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0바이트보다 크고 129바이트보다 작아야 합니다.

*object-schema*가 제공되지 않으면 규정자에 디폴트 스키마가 사용됩니다.

함수 결과의 데이터 유형은 VARCHAR(128)입니다. *object-name*이 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, *object-name*이 널(NULL)이면 결과도 널(NULL)이 됩니다. *object-schema*가 널(NULL) 값이면 디폴트 스키마 이름이 사용됩니다. 결과는 스키마 이름을 나타내는 문자열입니다. 결과 스키마는 다음 중 하나에 대한 스키마 이름을 표시할 수 있습니다.

테이블 *object-name* 값은 테이블 이름(*object-schema*의 입력 또는 디폴트값이 리턴) 또는 스키마 이름이 리턴되는 테이블로 해결되는 별명 이름입니다.

뷰 *object-name* 값은 뷰 이름(*object-schema*의 입력 또는 디폴트값이 리턴) 또는 스키마 이름이 리턴되는 뷰로 해결되는 별명 이름입니다.

정의되지 않은 오브젝트
object-name 값은 정의되지 않은 오브젝트(*object-schema*의 입력 또는 디폴트 값이 리턴) 또는 스키마 이름이 리턴되는 정의되지 않은 오브젝트로 해결되는 별명 이름입니다.

따라서 이 함수에 널(NULL)이 아닌 *object-name* 값이 제공되면 결과 스키마 이름을 포함한 오브젝트 이름이 없더라도 항상 값이 리턴됩니다. 예를 들어, TABLE_SCHEMA('DEPT', 'PEOPLE')는 카탈로그 항목이 발견되지 않을 경우에 'PEOPLE'을 리턴합니다.

주: TABLE_SCHEMA 및 TABLE_NAME 스칼라 함수 사용 시 코디네이터 파티션과 카탈로그 파티션 사이에 발생되는 불필요한 통신을 방지하여 파티션된 데이터베이스 구성의 성능을 개선하기 위해 BASE_TABLE 테이블 함수를 대신 사용할 수 있습니다.

예를 들면, 다음과 같습니다.

- PBIRD는 HEDGES.T1 테이블에 정의된 PBIRD.A1 별명을 사용하여 SYSCAT.TABLES로부터 제공된 테이블에 대한 통계를 선택합니다.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('A1')
  AND TABSCHEMA = TABLE_SCHEMA ('A1')
```

  HEDGES.T1에 대해 요청된 통계가 카탈로그에서 검색됩니다.

- HEDGES.X1을 사용하여 SYSCAT.TABLES로부터 HEDGES.X1이라는 오브젝트에 대한 통계를 선택하십시오. HEDGES.X1이 별명인지 아니면 테이블인지 알 수 없으므로, TABLE_NAME 및 TABLE_SCHEMA를 사용하십시오.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
  AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
```

  HEDGES.X1이 테이블이고 HEDGES.X1에 대해 요청된 통계가 카탈로그에서 검색된다고 가정하십시오.

- HEDGES.T2가 존재하지 않는 HEDGES.T2에 정의된 PBIRD.A2 별명을 사용하여 SYSCAT.TABLES로부터 제공된 테이블에 대한 통계를 선택하십시오.

```
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 리턴합니다.

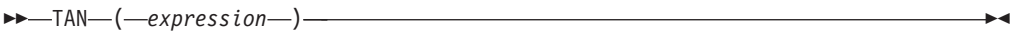
- 임의의 별명 항목에 대해 최종 참조 이름과 함께 SYSCAT.TABLES에서 각 항목의 규정된 이름을 선택하십시오.

```
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 오브젝트의 규정된 이름 및 각 별명 항목의 최종 참조 이름(별명이 분석된 이후)을 리턴합니다. 별명이 아닌 모든 항목의 경우

BASE_TABNAME과 BASE_TABSCHEMA는 널(NULL)이므로 REAL_SCHEMA 및 REAL_NAME 컬럼에 널(NULL)이 포함됩니다.

TAN



스키마는 SYSIBM입니다. (TAN 함수의 SYSFUN 버전은 계속 사용 가능합니다.)

인수의 tangent를 리턴합니다. 여기서, 인수는 라디안으로 표시되는 각도입니다.

인수는 어떤 내장 숫자 데이터 유형도 될 수 있습니다(DECFLOAT의 경우 제외). 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

TANH

스키마는 SYSIBM입니다.

인수의 hyperbolic tangent를 리턴합니다. 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 모든 내장 숫자 데이터 유형(DECFLOAT 제외)일 수 있습니다. 인수는 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 숫자입니다. 인수가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

TIME

스키마는 SYSIBM입니다.

TIME 함수는 값에서 시간(time)을 리턴합니다.

인수는 날짜, 시간, 시간소인이거나, CLOB, LONG VARCHAR, DBCLOB 또는 LONG VARGRAPHIC이 아닌 날짜, 시간, 시간소인의 유효한 문자열 표현이어야 합니다.

유니코드 데이터베이스에서만 시간 또는 시간소인의 그래픽 문자열 표현인 인수를 지원합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 시간입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

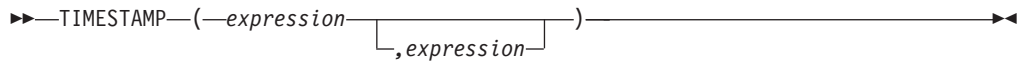
- 인수가 날짜 또는 날짜 문자열 표현인 경우:
  - 결과는 자정입니다.
- 인수가 시간인 경우
  - 결과는 해당 시간입니다.
- 인수가 시간소인인 경우
  - 결과는 시간소인의 시간 부분입니다.
- 인수가 시간 또는 시간소인 문자열 표현인 경우:
  - 결과는 문자열로 표시되는 시간입니다.

예:

- 당일(임의의 날) 현재 시간보다 1시간 이상 늦게 수신된 모든 정보를 IN_TRAY 샘플 테이블에서 선택하십시오.

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



스키마는 SYSIBM입니다.

TIMESTAMP 함수는 하나의 값이나 값 쌍에서 시간소인을 리턴합니다.

유니코드 데이터베이스만이 날짜, 시간 또는 시간소인의 그래픽 문자열 표현인 인수를 지원합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

인수에 대한 규칙은 두 번째 인수의 지정 여부 및 두 번째 인수의 데이터 유형에 따라 다릅니다.

- 하나의 인수만 지정된 경우, 이는 내장 데이터 유형인 날짜, 시간소인 또는 CLOB가 아닌 문자열 중 하나의 값을 리턴하는 표현식이어야 합니다. 인수가 문자열인 경우, 이는 다음 중 하나여야 합니다.
  - 날짜 또는 시간소인의 유효한 문자열 표현. 날짜 시간소인 값의 문자열 표현의 유효한 포맷에 대해서는, 『날짜 시간 값』의 『날짜 시간 값의 문자열 표현』을 참조하십시오.
  - GENERATE_UNIQUE 함수의 결과로 가정되는 실제 길이가 13인 문자열.
  - 길이가 14인 문자열은 *yyyyxxddhhmmss* 형식으로 유효한 날짜 및 시간을 나타내는 숫자 문자열입니다. 여기서, *yyyy*는 연도, *xx*는 월, *dd*는 일, *hh*는 시, *mm*은 분, *ss*는 초입니다.
- 두 인수를 모두 지정할 경우
  - 두 번째 인수의 데이터 유형이 정수가 아닌 경우:
    - 첫 번째 인수는 날짜 또는 날짜의 유효한 문자열 표현이고 두 번째 인수는 시간 또는 시간의 유효한 문자열 표현이어야 합니다.
  - 두 번째 인수의 데이터 유형이 정수인 경우:
    - 첫 번째 인수가 날짜, 시간소인이거나, 시간소인 또는 날짜의 유효한 문자열 표현이어야 합니다. 두 번째 인수는 시간소인 정밀도를 나타내는 범위 0 - 12인 정수 상수여야 합니다.

함수의 결과는 시간소인입니다.

시간소인 정밀도 및 기타 규칙은 두 번째 인수의 지정 여부에 따라 다릅니다.

- 모든 인수가 지정되어 있고 두 번째 인수가 정수가 아닌 경우:
  - 결과는 첫 번째 인수에서 지정된 날짜와 두 번째 인수에서 지정된 시간을 포함한 timestamp(6)입니다. 시간소인의 소수 초 파트가 0입니다.

- 모든 인수가 지정되어 있고 두 번째 인수가 정수인 경우:
  - 결과는 두 번째 인수에서 지정된 정밀도가 있는 시간소인입니다.
- 하나의 인수만이 지정되고 인수가 시간소인(*p*)인 경우:
  - 결과는 시간소인(*p*)입니다.
- 하나의 인수만 지정되고 인수가 날짜인 경우:
  - 결과는 timestamp(0)로 캐스트된 자정의 시간으로 가정된 날짜입니다.
- 하나의 인수만 지정되고 인수가 문자열인 경우
  - 결과는 누락된 시간 정보를 포함한 이전에 설명한 것과 같이 확장된 문자열로 나타낸 timestamp(6) 값입니다. 인수가 길이 14의 문자열인 경우, 시간소인의 소수초 파트는 0입니다.

인수에 날짜 정보만을 포함한 경우 결과 값의 시간 정보는 모두 0입니다. 인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우 결과는 널(NULL)이 됩니다.

예:

- 컬럼 START_DATE(날짜)의 값에 1988-12-25에 해당되는 값이 있고, 컬럼 START_TIME(시간)의 값에 17.12.30에 해당되는 값이 있다고 가정하십시오.

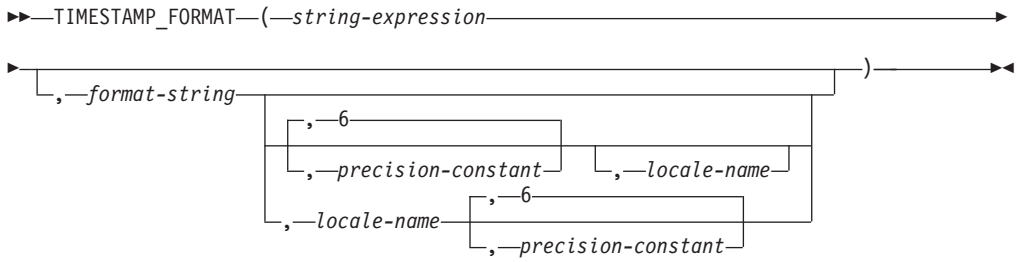
  ```
  TIMESTAMP(START_DATE, START_TIME)
  ```

  값 '1988-12-25-17.12.30.000000'을 리턴합니다.
- 7자리의 소수 초인 시간소인 문자열을 TIMESTAMP(9) 값으로 변환합니다.

  ```
  TIMESTAMP('2007-09-24-15.53.37.2162474',9)
  ```

  값 '2007-09-24-15.53.37.216247400'을 리턴합니다.

TIMESTAMP_FORMAT



스키마는 SYSIBM입니다.

TIMESTAMP_FORMAT 함수는 지정된 형식을 사용하여 입력 문자열 해석을 기반으로 하는 시간소인을 리턴합니다.

string-expression

표현식은 내장 CHAR 또는 VARCHAR 데이터 유형인 값을 리턴해야 합니다. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다. *string-expression*은 *format-string*이 지정한 형식에 대응하는 시간소인 구성요소를 포함해야 합니다.

format-string

표현식은 내장 CHAR 또는 VARCHAR 데이터 유형인 값을 리턴해야 합니다. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다. 실제 길이는 254바이트를 넘지 않아야 합니다(SQLSTATE 22007). 값은 *string-expression*이 해석된 후 시간소인 값으로 변환되는 방법에 대한 템플리트입니다.

유효한 *format-string*은 최소 하나의 형식 요소를 포함하고 시간소인 구성요소에 대한 다중 스펙을 포함하지 않아야 하며, 626 페이지의 표 58에 별도로 명시되지 않으면 형식 요소 조합을 포함할 수 있습니다(SQLSTATE 22007). 예를 들어, *format-string*은 YY와 YYYY를 둘 다 포함할 수는 없습니다. 둘 다 *string-expression*의 연도 구성요소를 해석하는 데 사용되기 때문입니다. 함께 지정할 수 없는 형식 요소를 판별하려면 테이블을 참조하십시오. 하나 이상의 다음 구분자 문자로 두 형식 요소를 선택적으로 구분할 수 있습니다.

- 마이너스 부호(-)
- 마침표(.)
- 슬래시(/)
- 쉼표(,)
- 어포스트로피 (')
- 세미콜론(;)

- 콜론(:)
- 공백( )

또한 *format-string*의 시작 또는 끝에 구분자 문자를 지정할 수 있습니다. 출력 문자열에서 이러한 구분자 문자를 조합하여 사용할 수 있습니다(예: 'YYYY/MM-DD HH:MM.SS').

표 58. TIMESTAMP_FORMAT 함수의 형식 요소

형식 요소	시간소인의 관련 구성요소	설명
AM 또는 PM	시	마침표가 없는 정오 표시기(아침 또는 저녁). 이 형식 요소는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
A.M. 또는 P.M.	시	마침표가 있는 정오 표시기(아침 또는 저녁). 이 형식 요소는 완전 문자열(『A.M.』 또는 『P.M.』)을 사용하며 적용되는 로케일 이름과 독립적입니다.
DAY, Day 또는 day	없음	대문자, 첫 글자만 대문자 또는 소문자 형식의 일 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
DY, Dy 또는 dy	없음	대문자, 첫 글자만 대문자 또는 소문자 형식의 약어화된 일 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
D	없음	요일(1 - 7). 첫 번째 요일은 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
DD	일	월 중 일(01 - 31).
DDD	월, 일	연 중 일(001 - 366).

표 58. TIMESTAMP_FORMAT 함수의 형식 요소 (계속)

형식 요소	시간소인의 관련 구성요소	설명
FF 또는 FF*n*	소수 초	소수 초(0 - 999999999999). 숫자 *n*은 *string-expression*에서 예측된 숫자 수를 지정하는 데 사용됩니다. *n*에 유효한 값은 선행 영(0)이 없는 1 - 12입니다. FF 지정은 FF6 지정과 동일합니다. FF가 마지막 형식 요소이고 소수 초의 숫자 수가 형식이 지정한 것보다 작으면, 지정된 숫자의 오른쪽에 영(0) 숫자가 채워집니다.
HH	시	HH는 HH12와 동일하게 동작합니다.
HH12	시	12시간 형식의 하루 중 시각(01 - 12). AM은 디폴트 정오 표시기입니다.
HH24	시	24시간 형식의 하루 중 시각(00 - 24).
J	년, 월, 일	율리우스력 일(BC 4713년 1월 1일 이후의 경과일 수).
MI	분	분(00 - 59).
MM	월	월(01 - 12).
MONTH, Month 또는 month	월	대문자, 첫 글자만 대문자 또는 소문자 형식의 월 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
MON, Mon 또는 mon	월	대문자, 첫 글자만 대문자 또는 소문자 형식의 약어화된 월 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
NNNNNN	마이크로초	마이크로초(000000 - 999999). FF6과 같습니다.
RR	년	조정된 연도의 마지막 두 숫자(00 - 99).
RRRR	년	조정된 4자리 연도(0000 - 9999).
SS	초	초(00 - 59).
SSSSS	시, 분, 초	지난 자정 이후의 초 수(00000 - 86400).

표 58. TIMESTAMP_FORMAT 함수의 형식 요소 (계속)

형식 요소	시간소인의 관련 구성요소	설명
Y	년	연도의 마지막 숫자(0 - 9). 현재 연도의 처음 세 숫자는 전체 4자리 연도를 판별하는 데 사용됩니다.
YY	년	연도의 마지막 두 숫자(00 - 99). 현재 연도의 처음 두 숫자는 전체 4자리 연도를 판별하는 데 사용됩니다.
YYY	년	연도의 마지막 세 숫자(000 - 999). 현재 연도의 첫 번째 숫자는 전체 4자리 연도를 판별하는 데 사용됩니다.
YYYY	년	4자리 연도(0000 - 9999).

주: 626 페이지의 표 58의 형식 요소는 다음 경우를 제외하고 대소문자를 구분하지 않습니다.

- AM, PM
- A.M., P.M.
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

DAY, Day, day, DY, Dy, dy 및 D 형식 요소는 결과 시간소인의 구성요소에 제공되지 않습니다. 그러나 이러한 형식 요소에 대해 지정된 값은 결과 시간소인의 년, 월, 일 구성요소 조합에 올바른 것이어야 합니다(SQLSTATE 22007). 예를 들어, 'en_US' 값이 *locale-name*에 사용된다고 가정하면 *string-expression* 값 'Monday 2008-10-06'은 'Day YYYY-MM-DD' 값에 유효합니다. 그러나 *string-expression* 값으로 'Tuesday 2008-10-06'을 사용하면 동일한 *format-string*에 대해 오류가 발생합니다.

RR 및 RRRR 형식 요소를 사용하여 다음 표에 따라 현재 연도의 맨 왼쪽 2자리 숫자에 종속된 2자리 값 또는 4자리 값을 생성하도록 값을 조정하여 연도 스펙을 해석하는 방법을 변경할 수 있습니다.

현재 연도의 마지막 두 숫자	*string-expression*의 2자리 연도	시간소인의 연도 구성요소의 처음 두 숫자
00-50	00-49	현재 연도의 처음 두 숫자
51-99	00-49	현재 연도의 처음 두 숫자 + 1
00-50	50-99	현재 연도의 처음 두 숫자 - 1

현재 연도의 마지막 두 숫자	*string-expression*의 2자리 연도	시간소인의 연도 구성요소의 처음 두 숫자
51-99	50-99	현재 연도의 처음 두 숫자

예를 들어, 현재 연도가 2007이면 'RR' 형식의 '86'은 1986을 의미하지만 현재 연도가 2052이면 2086을 의미합니다.

*format-string*에 시간소인의 다음 구성요소 중 하나에 대한 형식이 포함되지 않으면 다음 디폴트값이 사용됩니다.

시간소인 구성요소	디폴트값
년	4자리로 된 현재 연도
월	2자리로 된 현재 월
일	01(그 달의 첫 번째 날)
시	00
분	00
초	00
소수 초	결과의 시간소인 정밀도와 일치하는 다수의 영(0)

*format-string*의 대응하는 형식 요소에 대해 최대 유효 숫자 수를 갖지 않는 시간소인 값의 구성요소(즉, 월, 일, 시, 분, 초)에 대해 선행 영(0)을 지정할 수 있습니다.

시간소인의 구성요소(예: 년, 월, 일, 시, 분, 초)를 나타내는 *format-string*의 서브스트링은 대응하는 형식 요소가 표시한 시간소인의 해당 구성요소에 대해 최대 숫자 수 미만을 포함할 수 있습니다. 누락된 숫자는 디폴트로 영(0)으로 설정됩니다. 예를 들어, *format-string*이 'YYYY-MM-DD HH24:MI:SS'이면 '999-3-9 5:7:2'의 입력 값은 '0999-03-09 05:07:02'와 동일한 결과를 생성합니다.

*format-string*이 지정되지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값을 기반으로 디폴트 형식을 사용하여 *string-expression*을 해석합니다.

precision-constant

결과의 시간소인 정밀도를 지정하는 정수 상수. 값은 0 - 12 범위여야 합니다. 지정하지 않으면, 시간소인 정밀도 디폴트값은 6입니다.

locale-name

다음 형식 요소에 사용되는 로케일을 지정하는 문자 상수.

- AM, PM
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

locale-name 값은 대소문자를 구분하지 않으며 유효한 로케일이어야 합니다 (SQLSTATE 42815). 유효한 로케일 및 로케일 이름 지정에 대한 정보는 *자국어 안내서* 의 『SQL 및 XQuery의 로케일 이름』을 참조하십시오. *locale-name*이 지정되지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값이 사용됩니다.

함수의 결과는 *precision-constant*를 기반으로 하는 정밀도를 가진 시간소인입니다. 처음 두 인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 처음 두 인수 중 하나가 널(NULL)인 경우, 결과는 널(NULL) 값입니다.

주

- ***율리우스력 및 그레고리오력***: 이 함수에서는 1582년 10월 15일에 율리우스력에서 그레고리오력으로의 전이를 고려합니다.
- ***결정론:*** TIMESTAMP_FORMAT은 결정 함수입니다. 그러나 다음 함수 호출은 CURRENT LOCALE LC_TIME 또는 CURRENT TIMESTAMP 특수 레지스터 값에 종속됩니다.
  - *format-string*이 명시적으로 지정되지 않거나 *locale-name*이 명시적으로 지정되지 않고 다음 중 하나가 참인 경우:
    - *format-string*이 상수가 아님
    - *format-string*이 상수이고 로케일 구분 형식 요소를 포함
    - *format-string*이 상수이고 연도를 완벽히 정의하는 형식 요소(즉, J 또는 YYYY)를 포함하지 않으므로 현재 연도 값을 사용
    - *format-string*이 상수이고 월을 완벽히 정의하는 형식 요소(예: J, MM, MONTH 또는 MON)를 포함하지 않으므로 현재 월 값을 사용

  특수 레지스터를 사용할 수 없는 경우에는 특수 레지스터 값에 종속된 이러한 호출을 사용할 수 없습니다(SQLSTATE 42621 또는 428EC).
- ***구문 대체:*** TO_DATE 및 TO_TIMESTAMP는 TIMESTAMP_FORMAT의 동의어입니다.

예:

- IN_TRAY 테이블에 2000년 시작 전 1초(1999년 12월 31일 23:59:59)와 동일한 수신 시간소인을 가진 행을 삽입하십시오.

```
INSERT INTO IN_TRAY (RECEIVED)
  VALUES (TIMESTAMP_FORMAT('1999-12-31 23:59:59',
           'YYYY-MM-DD HH24:MI:SS'))
```

- 응용프로그램은 INDATEVAR 변수에 날짜 정보 문자열을 수신합니다. 이 값은 엄밀히 형식화되지 않으며 연도의 경우에는 둘 또는 네 숫자를, 월 및 일의 경우에는 하나 또는 두 숫자를 포함할 수 있습니다. 날짜 구성요소는 마이너스 부호(-) 또는 슬래시(/) 문자를 사용하여 구분할 수 있으며 일, 월 및 년 순으로 예측됩니다. 시간

정보는 시(24시간 형식)와 분으로 구성되며, 일반적으로 콜론으로 구분됩니다. 샘플 값으로 '15/12/98 13:48' 및 '9-3-2004 8:02'를 들 수 있습니다. IN_TRAY 테이블에 이러한 값을 삽입하십시오.

```
INSERT INTO IN_TRAY (RECEIVED)
  VALUES (TIMESTAMP_FORMAT(:INDATEVAR,
    'DD/MM/RRRR HH24:MI'))
```

RRRR 형식 사용은 2자리 및 4자리 연도 값을 허용하며 현재 연도를 기반으로 누락된 처음 두 숫자를 지정합니다. YYYY가 사용된 경우, 2자리 연도를 가진 입력 값에는 선행 영(0)이 있습니다. 슬래시 구분자도 마이너스 부호 문자를 허용합니다. 현재 연도가 2007년이라고 가정하면 샘플 값의 결과 시간소인은 다음과 같습니다.

```
'15/12/98 13:48' --> 1998-12-15-13.48.00.000000
'9-3-2004 8:02'  --> 2004-03-09-08.02.00.000000
```

TIMESTAMP_ISO

스키마는 SYSFUN입니다.

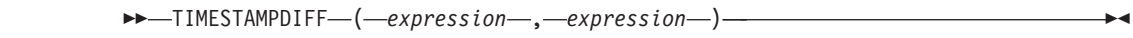
날짜, 시간 또는 시간소인 인수에 기초화 시간소인 값을 리턴합니다. 인수가 날짜이면 모든 시간 요소에 대해 0을 삽입합니다. 인수가 시간이면 날짜 요소에 대해 CURRENT DATE 특수 레지스터 값을 삽입하고 분할 시간 요소에 대해 0을 삽입합니다.

표현식은 내장 CHAR, VARCHAR, DATE, TIME 또는 TIMESTAMP 데이터 유형인 값을 리턴해야 합니다. 유니코드 데이터베이스의 경우, 제공된 인수에 GRAPHIC 또는 VARGRAPHIC 데이터 유형이 있으면, 함수를 평가하기 전에 먼저 문자열로 변환됩니다. 문자열 표현식은 날짜 또는 시간소인의 유효한 문자열 표현을 리턴해야 합니다.

TIMESTAMP_ISO 함수는 일반적으로 결정적 함수로 정의됩니다. 첫 번째 인수에 TIME 데이터 유형이 있으면, 함수는 결정적이지 않습니다. CURRENT DATE가 시간소인 값의 날짜 분할 영역에 사용되기 때문입니다.

함수의 결과는 TIMESTAMP입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

TIMESTAMPDIFF



스키마는 SYSFUN입니다.

두 시간소인의 차이에 따라 첫 번째 인수가 정의한 유형의 추정된 간격 수를 리턴합니다.

첫 번째 인수는 INTEGER 또는 SMALLINT가 될 수 있습니다. 간격(첫 번째 인수)의 유효한 값은 다음과 같습니다.

1 분할초

2 초

4 분

8 시간

16 일

32 주

64 월

128 분기

256 연도

두 번째 인수는 두 개의 시간소인 유형을 뺀 결과를 CHAR(22)로 변환한 결과입니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

차이를 계산할 때 다음 가정이 사용될 수 있습니다.

- 1년은 365일입니다.
- 한 달은 30일입니다.
- 하루는 24시간입니다.
- 한 시간은 60분입니다.
- 1분은 60초입니다.

이러한 가정은 시간소인 지속 기간인 두 번째 인수의 정보를 첫 번째 인수에 지정된 간격 유형으로 변환할 때 사용됩니다. 리턴되는 추정은 일 수에 따라 다를 수 있습니다. 예를 들어, '1997-03-01-00.00.00' 및 '1997-02-01-00.00.00' 사이의 차이에 대해 일

수(간격 16)가 요청되는 경우 결과는 30입니다. 이는 시간소인 사이의 차이가 한 달이고 한 달은 30일인 것으로 가정하기 때문입니다.

예:

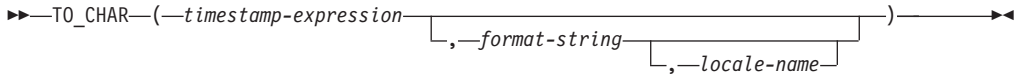
다음 예는 4277(두 시간소인 사이의 분 수)입니다.

```
TIMESTAMPDIFF(4,CHAR(TIMESTAMP('2001-09-29-11.25.42.483219') -
  TIMESTAMP('2001-09-26-12.07.58.065497')))
```

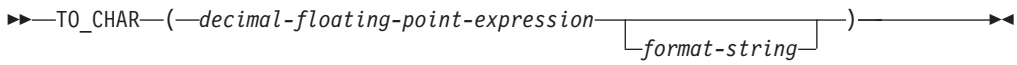
TO_CHAR

문자를 varchar로

varchar 시간소인:



varchar 10진 부동 소수점:

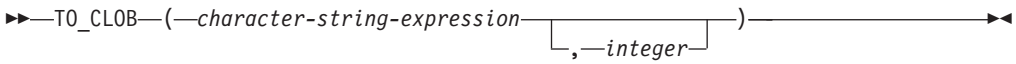


스키마는 SYSIBM입니다.

TO_CHAR 함수는 문자 템플리트를 사용하여 형식화된 입력 표현식의 문자 표현을 리턴합니다.

TO_CHAR 스칼라 함수는 VARCHAR_FORMAT 스칼라 함수의 동의어입니다.

TO_CLOB

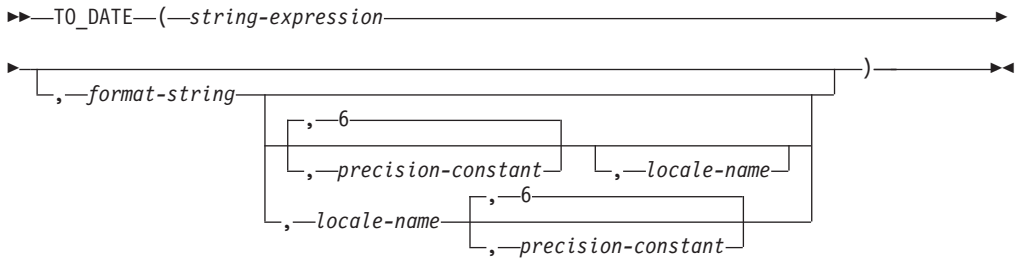


스키마는 SYSIBM입니다.

TO_CLOB 함수는 문자열 유형의 CLOB 표시를 리턴합니다.

TO_CLOB 스칼라 함수는 CLOB 스칼라 함수의 동의어입니다.

TO_DATE

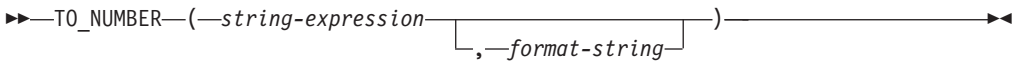


스키마는 SYSIBM입니다.

TO_DATE 함수는 지정된 형식을 사용하여 입력 문자열 표현을 기반으로 하는 시간소인을 리턴합니다.

TO_DATE 스칼라 함수는 TIMESTAMP_FORMAT 스칼라 함수의 동의어입니다.

TO_NUMBER

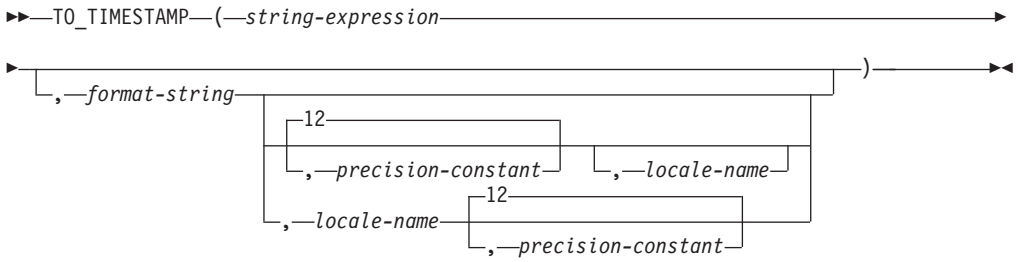


스키마는 SYSIBM입니다.

TO_NUMBER 함수는 지정된 형식을 사용하여 입력 문자열 해석을 기반으로 하는 DECFLOAT(34) 값을 리턴합니다.

TO_NUMBER 스칼라 함수는 DECFLOAT_FORMAT 스칼라 함수의 동의어입니다.

TO_TIMESTAMP

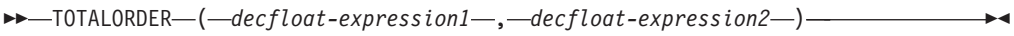


스키마는 SYSIBM입니다.

TO_TIMESTAMP 함수는 지정된 형식을 사용하여 입력 문자열 해석을 기반으로 하는 시간소인을 리턴합니다.

TO_TIMESTAMP 스칼라 함수는 *precision-constant*의 디폴트값이 12인 것은 제외하고 TIMESTAMP_FORMAT 스칼라 함수의 동의어입니다.

TOTALORDER



스키마는 SYSIBM입니다.

TOTALORDER 함수는 두 인수의 비교 순서를 표시하는 -1, 0 또는 1의 SMALLINT 값을 리턴합니다.

decfloat-expression1
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 DECFLOAT(34)가 아니면 처리를 위해 논리적으로 DECFLOAT(34)로 변환됩니다.

decfloat-expression2
임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 10진 부동 소수점 값이 아니면 처리를 위해 DECFLOAT(34)로 변환됩니다.

숫자 비교는 정확하며 결과는 범위 및 정밀도가 무제한인 것처럼 제한된 피연산자에 대해 판별됩니다. 오버플로우 또는 언더플로우 조건은 발생할 수 없습니다.

하나의 값이 DECFLOAT(16)이고 다른 값이 DECFLOAT(34)이면 DECFLOAT(16) 값은 비교 이전에 DECFLOAT(34)로 변환됩니다.

TOTALORDER 함수의 시맨틱은 IEEE 754R의 전체 순서 술어 규칙을 기초로 합니다. TOTALORDER는 다음 값을 리턴합니다.

- *decfloat-expression1*이 *decfloat-expression2*에 비해 순서에서 더 낮은 경우 -1
- *decfloat-expression1* 및 *decfloat-expression2* 둘 다 동일한 순서를 가지고 있는 경우 0
- *decfloat-expression1*이 *decfloat-expression2*에 비해 순서에서 더 높은 경우 1

특수 값 및 제한된 숫자의 순서 지정은 다음과 같습니다.

```
-NAN<-SNAN<-INFINITY<-0.10<-0.100<-0<0<0.100<0.10<INFINITY<SNAN<NAN
```

함수의 결과는 SMALLINT 값입니다. 인수 중 하나가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

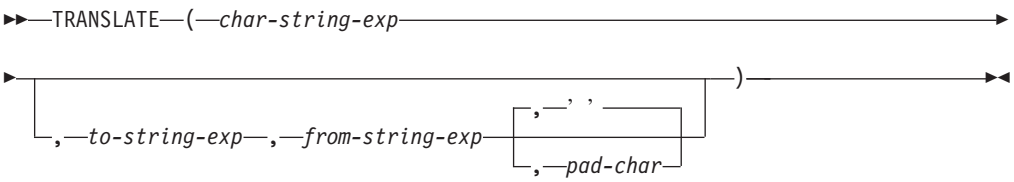
- 다음 예는 10진 부동 소수점 값을 비교하기 위해 TOTALORDER 함수를 사용하는 경우를 나타냅니다.

```
TOTALORDER(-INFINITY, -INFINITY) = 0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.0)) = 0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.00)) = -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-0.5)) = -1
```

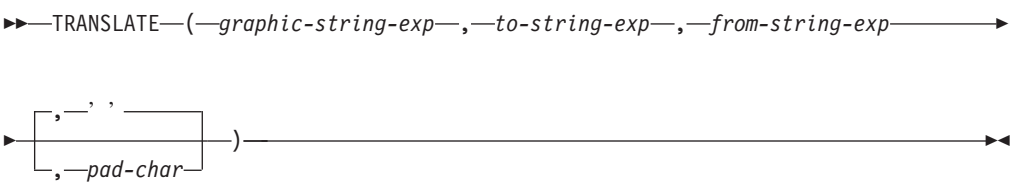
```
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(0.5)) = -1
TOTALORDER(DECFLOAT(-1.0), INFINITY) = -1
TOTALORDER(DECFLOAT(-1.0), SNAN) = -1
TOTALORDER(DECFLOAT(-1.0), NAN) = -1
TOTALORDER(NAN, DECFLOAT(-1.0)) = 1
TOTALORDER(-NAN, -NAN) = 0
TOTALORDER(-SNAN, -SNAN) = 0
TOTALORDER(NAN, NAN) = 0
TOTALORDER(SNAN, SNAN) = 0
TOTALORDER(-1.0, -1.0) = 0
TOTALORDER(-1.0, -1.00) = -1
TOTALORDER(-1.0, -0.5) = -1
TOTALORDER(-1.0, 0.5) = -1
TOTALORDER(-1.0, INFINITY) = -1
TOTALORDER(-1.0, SNAN) = -1
TOTALORDER(-1.0, NAN) = -1
```

TRANSLATE

문자열 표현식:



그래픽 문자열 표현식:



스키마는 SYSIBM입니다.

TRANSLATE 함수는 문자열 표현식에 있는 하나 이상의 문자가 다른 문자로 변환될 수 있는 값을 리턴합니다.

TRANSLATE 함수는 *char-string-exp*의 모든 문자 또는 *from-string-exp*에서도 발생하는 *graphic-string-exp*의 모든 문자를 *to-string-exp*의 해당 문자로 변환하거나, 해당 문자가 없을 경우 *pad-char-exp*가 지정한 채움 문자로 변환합니다.

char-string-exp **또는** *graphic-string-exp*

변환할 문자열을 지정합니다.표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

to-string-exp

*char-string-exp*의 특정 문자가 변환될 문자열을 지정합니다.

표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *to-string-exp* 값이 지정되지 않고 데이터 유형이 그래픽이 아닌 경우 *char-string-exp*의 모든 문자는 단일 문자형이 됩니다. 즉 a - z 문자는 A - Z 문자로 변환되며, 그밖의 다른 문자가 존재할 경우 해당 대문자로 변환됩니다. 예를 들어, 코드 페이지 850에서 é는 É에 맵핑되

나 ÿ는 코드 페이지 850이 Ÿ를 포함하지 않으므로 맵핑되지 않습니다. 결과 문자의 코드 포인트 길이가 소스 문자의 코드 포인트 길이와 같지 않으면 소스 문자는 변환되지 않습니다.

from-string-exp

문자열이 *char-string-exp*에서 발견될 경우 *to-string-exp*의 해당 문자로 변환될 문자열을 지정합니다.

표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. *from-string-exp*에 중복 문자가 포함되어 있으면 발견되는 첫 번째 문자가 사용되고 중복되는 것은 무시됩니다. *to-string-exp*이 *from-string-exp*보다 길면 여분의 문자는 무시됩니다. *to-string-exp*이 지정되면 *from-string-exp*도 지정되어야 합니다.

pad-char-exp

*to-string-exp*이 *from-string-exp*보다 짧은 경우 *to-string-exp*을 채우는 데 사용할 단일 문자를 지정합니다. 표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 값은 하나의 길이 속성을 가져야 합니다. 값을 지정하지 않은 경우 1바이트 공백 문자로 처리됩니다.

*graphic-string-exp*에서 *pad-char-exp*만 선택적이며 (값이 지정되지 않으면 2바이트 공백 문자로 처리됨) 채움 문자를 포함하여 각 인수는 그래픽 데이터 유형이어야 합니다.

결과의 데이터 유형과 코드 페이지는 첫 번째 인수의 데이터 유형과 코드 페이지와 같습니다. 첫 번째 인수가 호스트 변수일 경우 결과 코드 페이지는 데이터베이스 코드 페이지입니다. 각 인수 또는 첫 번째 인수가 FOR BIT DATA로 정의되어 있지 않은 경우, 첫 번째 인수가 아닌 각 인수는 결과 코드 페이지로 변환됩니다.

문자 및 그래픽이 데이터 유형과 같다고 간주되는 유니코드 데이터베이스에는 다음 예외가 있습니다.

- 첫 번째 인수를 제외한 모든 인수가 FOR BIT DATA인 경우 결과의 코드 페이지는 1208입니다.
- FOR BIT DATA 인수가 없는 경우, 결과의 코드 페이지는 인수 세트에서 자주 나타나는 코드 페이지입니다.
- FOR BIT DATA 인수가 없는 경우 두 개의 다른 코드 페이지가 인수 세트에 똑같이 자주 나타나면 결과의 코드 페이지는 1200입니다.

결과의 길이 속성은 첫 번째 인수의 속성과 같습니다. 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

인수의 데이터 유형이 CHAR 또는 VARCHAR이면 *to-string-exp* 및 *from-string-exp*의 해당 문자는 동일한 바이트 수를 가져야 합니다. 예를 들면, 1바이트 문자에서 복수 바이트 문자로 또는 복수 바이트 문자에서 1바이트 문자로의 변환은 유효하지 않습니다. *pad-char-exp* 인수는 유효한 복수 바이트 문자의 첫 바이트가 될 수 없습니다 (SQLSTATE 42815).

실행 파일을 비교하여 문자를 일치시킵니다. 데이터베이스 조합이 사용되지 않습니다.

*char-string-exp*만 지정되면 1바이트 문자는 단일 문자형이 되고 복수 바이트 문자는 변경되지 않습니다.

예를 들면, 다음과 같습니다.

- 호스트 변수 SITE(VARCHAR(30))가 'Hanauma Bay'의 값을 갖는다고 가정하십시오.

  ```
  TRANSLATE(:SITE)
  ```

  값 'HANAUMA BAY'를 리턴합니다.

  ```
  TRANSLATE(:SITE, 'j', 'B')
  ```

  값 'Hanauma jay'를 리턴합니다.

  ```
  TRANSLATE(:SITE,'ei','aa')
  ```

  값 'Heneume Bey'를 리턴합니다.

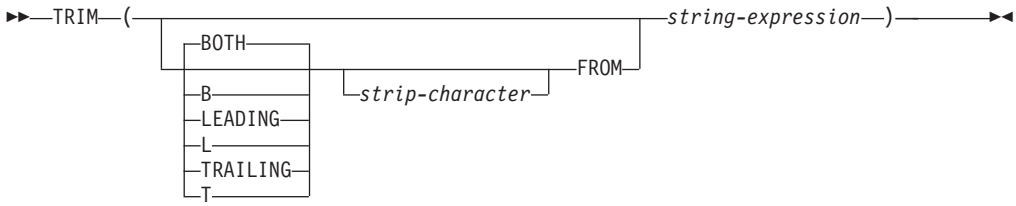
  ```
  TRANSLATE(:SITE, 'bA', 'Bay', '%')
  ```

  값 'HAnAumA bA%'를 리턴합니다.

  ```
  TRANSLATE(:SITE,'r','Bu')
  ```

  값 'Hana ma ray'를 리턴합니다.

TRIM



스키마는 SYSIBM입니다. 함수 시그니처에서 키워드를 사용할 때 함수 이름을 규정된 이름으로 지정할 수 없습니다.

TRIM 함수는 문자열 표현식의 끝이나 시작에서 다른 지정 문자 또는 공백을 제거합니다.

BOTH, LEADING, 또는 TRAILING
문자열 표현식의 시작, 끝 또는 양쪽 끝에서 문자열을 제거하는지 여부를 지정합니다. 이 인수를 지정하지 않으면 문자열의 시작과 끝에서 문자를 제거합니다.

strip-character
제거할 문자를 지정하는 단일 문자 상수입니다. *strip-character*는 UTF-32 인코딩이 단일 문자 또는 단일 숫자 값인 어떤 문자도 가능합니다. 문자의 2진 표시가 일치됩니다.

*strip-character*를 지정하지 않은 경우,

- *string-expression*이 DBCS 그래픽 문자열이면 디폴트 *strip-character*는 DBCS 공백이고 해당 코드 포인트는 데이터베이스 코드 페이지에 따라 결정됩니다.
- *string-expression*이 UCS-2 그래픽 문자열이면 디폴트 *strip-character*는 UCS-2 공백(X'0020')입니다.
- 그렇지 않으면 디폴트 *strip-character*는 SBCS 공백(X'20')입니다.

FROM *string-expression*
표현식은 내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 아닌 경우, 이는 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다.

결과는 *string-expression*의 길이 속성과 동일한 최대 길이를 갖는 가변 길이 문자열이 됩니다. 결과의 실제 길이는 *string-expression* 길이에서 제거할 바이트 수를 뺀 것입니다. 모든 문자를 제거하면 결과는 빈 가변 길이 문자열이 됩니다. 결과의 코드 페이지는 *string-expression*의 코드 페이지와 같습니다.

예를 들면, 다음과 같습니다.

- 유형이 CHAR(9)인 호스트 변수 HELLO의 값이 ' Hello'라고 가정하십시오.

```
SELECT TRIM(:HELLO),
  TRIM(TRAILING FROM :HELLO)
FROM SYSIBM.SYSDUMMY1
```

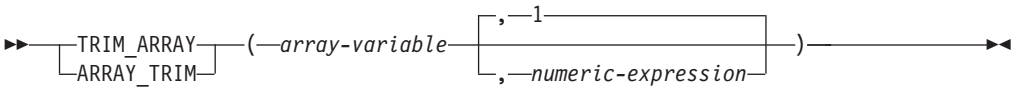
리턴되는 값은 각각 'Hello'와 ' Hello'입니다.

- 유형이 CHAR(9)인 호스트 변수 BALANCE의 값이 '000345.50'이라고 가정하십시오.

```
SELECT TRIM(L '0' FROM :BALANCE),
FROM SYSIBM.SYSDUMMY1
```

리턴되는 값은 '345.50'입니다.

TRIM_ARRAY



스키마는 SYSIBM입니다.

TRIM_ARRAY 함수는 배열 끝에서 요소를 삭제합니다.

array-variable

일반 배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 일반 배열 유형의 매개변수 표시문자에 대한 CAST 스펙. 연관된 배열 데이터 유형은 지정할 수 없습니다(SQLSTATE 42884).

numeric-expression

배열 끝에서 트리밍되는 요소 수를 지정합니다. *numeric-expression*은 INTEGER로 캐스트 가능한 값이 포함된 모든 숫자 데이터 유형이 가능합니다. *numeric-expression* 값은 0과 array-variable 카디널리티 사이여야 합니다(SQLSTATE 2202E). 디폴트값은 1입니다.

함수는 *array-variable*와 동일한 배열 유형이지만 INTEGER(*numeric-expression*)의 값만큼 감소된 카디널리티(cardinality)가 포함된 값을 리턴합니다. 결과는 널(NULL)일 수 있으며 인수가 널(NULL)인 경우 결과는 널(NULL) 값입니다.

규칙

연관된 배열에는 TRIM_ARRAY 함수가 지원되지 않습니다(SQLSTATE 42884).

TRIM_ARRAY 함수는 배열이 지원되는 컨텍스트의 지정 명령문에서 오른쪽에만 사용할 수 있습니다(SQLSTATE 42884).

예

- 배열 변수 RECENT_CALLS에서 마지막 요소를 제거합니다.

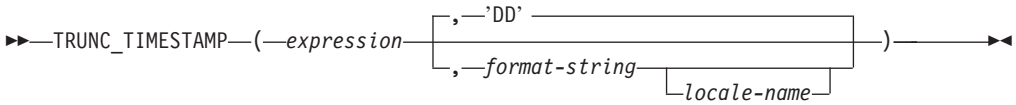
```
SET RECENT_CALLS = TRIM_ARRAY(RECENT_CALLS, 1)
```

- 배열 변수 SPECIALNUMBERS에서 처음 두 개의 요소만 SQL 배열 변수 EULER_CONST에 지정합니다.

```
SET EULER_CONST = TRIM_ARRAY(SPECIALNUMBERS, 8)
```

결과는 EULER_CONST가 두 개의 요소가 포함된 배열을 지정하고 첫 번째 요소 값은 2.71828183이며 두 번째 요소 값은 널(NULL) 값입니다.

TRUNC_TIMESTAMP



스키마는 SYSIBM입니다.

TRUNC_TIMESTAMP 스칼라 함수는 *format-string*에 지정된 단위로 전달된 *expression*인 시간소인을 리턴합니다. *format-string*을 지정하지 않은 경우 *expression*은 *format-string*에 대해 'DD'가 지정된 것처럼 가장 가까운 날로 절단됩니다.

expression
내장 데이터 유형인 날짜 또는 시간소인 중 하나의 값을 리턴하는 표현식입니다.

format-string
실제 길이가 254바이트를 넘지 않는 내장 문자열 데이터 유형을 리턴하는 표현식입니다. *format-string*의 형식 요소는 *expression*을 절단하는 방법을 지정합니다. 예를 들어, *format-string*이 'DD'이면, *expression*으로 표시되는 시간소인은 가장 가까운 날로 절단됩니다. 앞뒤 공백은 문자열에서 제거되며 결과 서브스트링은 시간소인에 유효한 형식 요소여야 합니다(SQLSTATE 22007). 디폴트는 'DD'입니다.

*format-string*에 대해 허용 가능한 값은 ROUND 함수 설명에 있는 형식 요소 테이블에 나열되어 있습니다.

locale-name
형식 모델 값 DAY, DY 또는 D를 사용할 때 첫 번째 요일을 판별하기 위해 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자가 구분되지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 해당되는 이름 지정 방법에 대한 정보는 ″SQL 및 XQuery의 로케일 이름″을 참조하십시오. *locale-name*을 지정하지 않은 경우 특수 레지스터 CURRENT LOCALE LC_TIME의 값이 사용됩니다.

함수의 결과는 시간소인 정밀도가 *expression*과 같은 TIMESTAMP입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

함수의 결과는 다음 시간소인 정밀도가 있는 TIMESTAMP입니다.

- 표현식의 데이터 유형이 TIMESTAMP(*p*)이면 *p*이고
- 표현식의 데이터 유형이 DATE이면 0이며
- 그렇지 않으면 6입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과는 널(NULL)이 됩니다.

주

- **결정론:** TRUNC_TIMESTAMP는 결정 함수입니다. 그러나 다음 함수 호출은 CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
  - *locale-name*이 명시적으로 지정되지 않고 다음 중 하나가 참인 경우 날짜 또는 시간소인 값의 절단:
    - *format-string*이 상수가 아님
    - *format-string*이 상수이고 로케일 구분 형식 요소를 포함

    특수 레지스터의 값에 종속되는 함수의 호출은 특수 레지스터를 사용할 수 없는 경우 사용할 수 없습니다.

예:

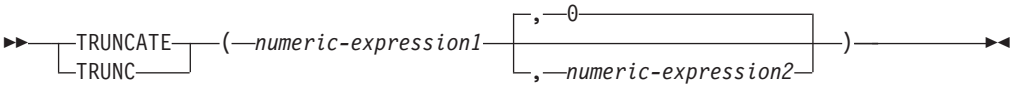
- 호스트 변수 TRNK_TMSTMP를 가장 가까운 연도 값으로 라운드된 현재 연도로 설정하십시오.

```
SET :TRNK_TMSTMP = TRUNC_TIMESTAMP('2000-03-14-17.30.00', 'YEAR');
```

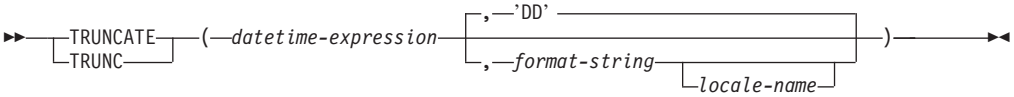
  호스트 변수 TRNK_TMSTMP는 값 2000-01-01-00.00.00.000000으로 설정됩니다.

TRUNCATE 또는 TRUNC

TRUNCATE 숫자:



TRUNCATE 날짜 시간:



스키마는 SYSIBM입니다. TRUNCATE 숫자 함수의 SYSFUN 버전은 계속 사용 가능합니다.

TRUNCATE 함수는 다음의 절단 값을 리턴합니다.

- 첫 번째 인수의 결과가 숫자 값이면, 소수점 오른쪽이나 왼쪽으로 지정된 자릿수로 절단된 숫자
- 첫 번째 인수가 날짜, 시간 또는 시간소인인 경우, *format-string*에서 지정한 단위로 절단된 날짜 시간 값

TRUNCATE 숫자

*numeric-expression1*에 숫자 데이터 유형이 있는 경우, TRUNCATE 함수는 *numeric-expression2*가 양수이면 소수점 오른쪽으로, *numeric-expression2*가 영(0)이거나 음수이면 소수점 왼쪽으로 *numeric-expression2* 자릿수만큼 절단된 *numeric-expression1*을 리턴합니다. *numeric-expression2*가 지정되지 않으면, *numeric-expression1*이 소수점 왼쪽의 영(0) 자리로 절단됩니다.

numeric-expression1

내장 CHAR, VARCHAR, GRAPHIC, VARGRAPHIC 또는 숫자 데이터 유형인 값을 리턴해야 하는 표현식. 값이 숫자 데이터 유형이 아니면, 함수를 평가하기 전에 내재적으로 DECFLOAT(34)로 캐스트됩니다.

표현식이 10진수 부동 소수점 데이터 유형이면, DECFLOAT 반올림 모드가 사용되지 않습니다. TRUNCATE의 반올림 동작은 ROUND_DOWN의 값에 해당됩니다. 다른 반올림 동작이 필요하면 QUANTIZE 함수를 사용합니다.

numeric-expression2

내장 숫자 데이터 유형인 값을 리턴하는 표현식입니다. 값이 INTEGER 유형이 아니면, 함수를 평가하기 전에 내재적으로 INTEGER로 캐스트됩니다.

*numeric-expression2*가 음수가 아닌 경우 *numeric-expression1*은 소수점 오른쪽 *numeric-expression2* 자릿수의 절대값으로 절단됩니다.

*numeric-expression2*가 음수인 경우 *numeric-expression1*은 소수점의 왼쪽에 있는 *numeric-expression2*+1 숫자의 절대값으로 절단됩니다. 음수 *numeric-expression2*의 절대값이 소수점 왼쪽의 자릿수보다 큰 경우, 결과는 0입니다. 예는 다음과 같습니다.

```
TRUNCATE(748.58,-4) = 0
```

결과의 데이터 유형, 길이 및 스케일 속성은 첫 번째 인수의 데이터 유형, 길이 및 스케일 속성과 같습니다.

인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

TRUNCATE 날짜 시간

*datetime-expression*에 날짜 시간 데이터 유형이 있으면, TRUNCATE 함수는 *format-string*에서 지정한 단위로 반올림된 *datetime-expression*을 리턴합니다. *format-string*이 지정되어 있지 않으면, 'DD'가 *format-string*에 대해 지정된 것처럼 *datetime-expression*이 가장 가까운 날로 절단됩니다.

datetime-expression

날짜, 시간 또는 시간소인의 값을 리턴해야 하는 표현식입니다. 이들 데이터 유형의 문자열 표현은 지원되지 않으며 이 함수를 사용하기 위해 날짜, 시간 또는 시간소인에 명시적으로 캐스트되어야 합니다. 또는 날짜나 시간 소인의 문자열 표현에 대해 TRUNC_TIMESTAMP 함수를 사용할 수 있습니다.

format-string

254바이트보다 크지 않은 실제 길이의 내장 문자열 데이터 유형을 리턴하는 표현식입니다. *format-string*에서의 형식 요소는 *datetime-expression*이 절단되어야 하는 방식을 지정합니다. 예를 들어 *format-string*이 'DD'인 경우, *datetime-expression*으로 표현된 시간소인은 가장 가까운 날로 절단됩니다. 앞뒤 공백은 문자열에서 제거되며 결과 부속 문자열은 *datetime-expression*의 유형에 대해 유효한 형식 요소여야 합니다(SQLSTATE 22007). 디폴트는 'DD'이며 *datetime-expression*의 데이터 유형이 TIME이면 사용할 수 없습니다.

*format-string*에 대해 허용 가능한 값은 ROUND 함수의 설명에 있습니다.

locale-name

형식 요소 값 DAY, DY 또는 D를 사용할 때 해당 주의 첫 번째 날을 판별하는 데 사용되는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자를 구분하지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 이름에 대해서는 『SQL 및 XQuery에 대한 로케일 이름』을 참조하십시오. *locale-name*이 지정되지 않으면, 특수 레지스터의 값 CURRENT LOCALE LC_TIME이 사용됩니다.

함수의 결과에 *datetime-expression*와 같은 날짜 유형이 있습니다. 결과가 널(NULL)이 될 수 있습니다. 인수가 널(NULL)일 경우, 결과는 널(NULL) 값이 됩니다.

결과의 데이터 유형과 길이 속성은 첫 번째 인수의 데이터 유형 및 길이 속성과 동일합니다.

인수가 널(NULL)이 될 수 있거나 인수가 10진수 부동 소수점이 아니고 DFT_SQLMATHWARN을 YES로 설정하여 데이터베이스를 구성한 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL) 값이 됩니다.

주

- **결정론:** TRUNCATE는 결정 함수입니다. 그러나 다음 함수 호출은 CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
  - *locale-name*이 명시적으로 지정되지 않고 다음 중 하나가 참인 경우 날짜 시간 값의 절단:
    - *format-string*이 상수가 아님
    - *format-string*이 상수이고 로케일 구분 형식 요소를 포함

    특수 레지스터를 사용할 수 없는 경우에는 특수 레지스터 값에 종속된 함수의 호출을 사용할 수 없습니다.

예:

- EMPLOYEE 테이블을 사용하여 최고 급여의 직원에 대한 평균 월별 급여를 계산하십시오. 결과를 소수점 오른쪽 두 자리로 절단하십시오.

```
SELECT TRUNCATE(MAX(SALARY)/12,2)
  FROM EMPLOYEE;
```

  급여가 가장 높은 직원이 연봉 $52750.00을 받으므로 예에서는 4395.83을 리턴합니다.

- 각각 2, 1, 0, -1 및 -2 소수 자릿수로 절단된 숫자 873.726을 표시하십시오.

```
VALUES (
  TRUNCATE(873.726,2),
  TRUNCATE(873.726,1),
  TRUNCATE(873.726,0),
  TRUNCATE(873.726,-1),
  TRUNCATE(873.726,-2),
  TRUNCATE(873.726,-3) );
```

  이 예에서는 873.720, 873.700, 873.000, 870.000, 800.000 및 0.000을 리턴합니다.

- 0 소수점 자리를 절단한 부동 소수점 숫자 873.726을 표시합니다.

```
VALUES(TRUNCATE(DECFLOAT(873.726),0))
```

값 873을 리턴합니다.

- 변수 vTRNK_DT를 가장 가까운 월 값으로 반올림되는 입력 날짜로 설정합니다.

```
SET vTRNK_DT = TRUNC(DATE('2000-08-16'), 'MONTH');
```

값 설정은 2000-08-01입니다.

- 호스트 변수 TRNK_TMSTMP를 가장 가까운 연도 값으로 반올림되는 현재 연도로 설정합니다.

```
SET :TRNK_TMSTMP = TRUNCATE('2000-03-14-17.30.00'), 'YEAR');
```

값 설정은 2000-01-01-00.00.00.000000입니다.

TYPE_ID

스키마는 SYSIBM입니다.

TYPE_ID 함수는 *expression*의 동적 데이터 유형에 대한 내부 유형 ID를 리턴합니다.

인수는 사용자 정의된 구조화 유형이어야 합니다. 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 구조화된 데이터 유형을 승인하므로, 다른 사용자 정의 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

함수 결과의 데이터 유형은 INTEGER입니다. *expression*이 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, *expression*이 널(NULL)이면 결과도 널(NULL)이 됩니다.

TYPE_ID 함수에서 리턴되는 값은 데이터베이스 사이에 이식 가능하지 않습니다. 동적 데이터 유형의 유형 스키마 및 유형 이름이 같아도, 값이 다를 수 있습니다. 이식성에 대해 코딩할 때, TYPE_SCHEMA 및 TYPE_NAME 함수를 사용하여 유형 스키마 및 유형 이름을 판별하십시오.

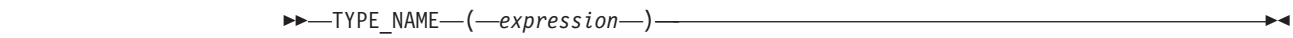
예를 들면, 다음과 같습니다.

- EMP 유형의 루트 테이블 EMPLOYEE와 MGR 유형의 서브테이블 MANAGER가 있는 테이블 계층이 존재합니다. 다른 테이블 ACTIVITIES에는 REF(EMP) SCOPE EMPLOYEE로 정의된 WHO_RESPONSIBLE 컬럼이 포함되어 있습니다. ACTIVITIES에 있는 참조마다, 참조에 해당되는 행의 내부 유형 ID를 표시하십시오.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_ID(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  DEREF 함수는 행에 해당되는 오브젝트를 리턴하기 위해 사용됩니다.

TYPE_NAME



스키마는 SYSIBM입니다.

TYPE_NAME 함수는 *expression*의 동적 데이터 유형의 규정되지 않은 이름을 리턴합니다.

인수는 사용자 정의된 구조화 유형이어야 합니다. 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 구조화된 데이터 유형을 승인하므로, 다른 사용자 정의 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

함수 결과의 데이터 유형은 VARCHAR(18)입니다. *expression*이 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, *expression*이 널(NULL)이면 결과도 널(NULL)이 됩니다. TYPE_NAME에서 리턴되는 유형 이름의 스키마 이름을 판별하려면 TYPE_SCHEMA 함수를 사용하십시오.

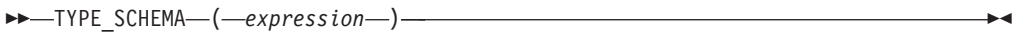
예를 들면, 다음과 같습니다.

- EMP 유형의 루트 테이블 EMPLOYEE와 MGR 유형의 서브테이블 MANAGER가 있는 테이블 계층이 존재합니다. 다른 테이블 ACTIVITIES에는 REF(EMP) SCOPE EMPLOYEE로 정의된 WHO_RESPONSIBLE 컬럼이 포함되어 있습니다. ACTIVITIES에 있는 참조마다, 참조에 해당되는 행 유형을 표시하십시오.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_NAME(DEREF(WHO_RESPONSIBLE)),
      TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  DEREF 함수는 행에 해당되는 오브젝트를 리턴하기 위해 사용됩니다.

TYPE_SCHEMA



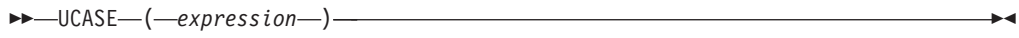
스키마는 SYSIBM입니다.

TYPE_SCHEMA 함수는 *expression*의 동적 데이터 유형 스키마 이름을 리턴합니다.

인수는 사용자 정의된 구조화 유형이어야 합니다. 이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 구조화된 데이터 유형을 승인하므로, 다른 사용자 정의 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

함수 결과의 데이터 유형은 VARCHAR(128)입니다. *expression*이 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, *expression*이 널(NULL)이면 결과도 널(NULL)이 됩니다. TYPE_SCHEMA에서 리턴되는 스키마 이름과 연관되는 유형 이름을 판별하려면 TYPE_NAME 함수를 사용하십시오.

UCASE

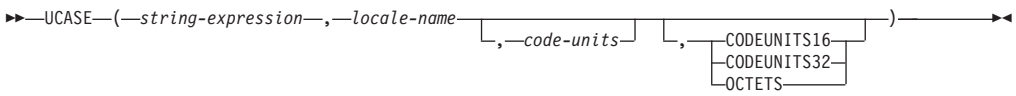


스키마는 SYSIBM입니다.

UCASE 함수는 첫 번째 인수(*char-string-exp*)만 지정된다는 점을 제외하고 TRANSLATE 함수와 동일합니다.

UCASE는 UPPER의 동의어입니다.

UCASE(로케일 구분)

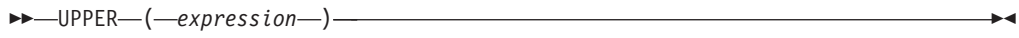


스키마는 SYSIBM입니다.

UCASE 함수는 지정된 로케일과 연관된 규칙을 사용하여 모든 문자가 대문자로 변환된 문자열을 리턴합니다.

UCASE는 UPPER의 동의어입니다.

UPPER

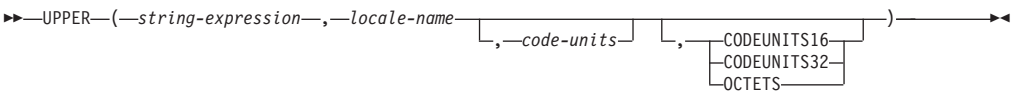


스키마는 SYSIBM입니다. (이 함수의 SYSFUN 버전은 이후의 호환성을 위해 계속 사용할 수 있습니다.)

UPPER 함수는 첫 번째 인수(*char-string-exp*)만 지정된다는 점을 제외하고는 TRANSLATE 함수와 동일합니다.

유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

UPPER(로케일 구분)



스키마는 SYSIBM입니다.

UPPER 함수는 지정된 로케일과 연관된 규칙을 사용하여 모든 문자가 대문자로 변환된 문자열을 리턴합니다.

string-expression

CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 문자열을 리턴하는 표현식입니다. *string-expression*이 CHAR 또는 VARCHAR이면, 표현식은 FOR BIT DATA가 될 수 없습니다(SQLSTATE 42815).

locale-name

대문자로의 변환에 대한 규칙을 정의하는 로케일을 지정하는 문자 상수입니다. *locale-name*의 값은 대소문자가 구분되지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 해당되는 이름 지정 방법에 대한 정보는 『SQL 및 XQuery의 로케일 이름』을 참조하십시오.

code-units

결과의 코드 단위 수를 지정하는 정수 상수입니다. 지정된 경우 *code-units*는 결과가 문자 데이터인 경우 1 - 32 672 사이의 정수여야 하고 결과가 그래픽 데이터인 경우 1 - 16 336 사이의 정수여야 합니다(SQLSTATE 42815). *code-units*를 명시적으로 지정하지 않은 경우 내재적으로 *string-expression*의 길이 속성이 됩니다. OCTETS가 지정되고 결과가 그래픽 데이터일 경우, *code-units*의 값은 짝수여야 합니다(SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 또는 OCTETS

*code-units*의 문자열 단위를 지정합니다.

CODEUNITS16은 *code-units*가 16비트 UTF-16 코드 단위로 표시될 것을 지정합니다. CODEUNITS32는 *code-units*가 32비트 UTF-32 코드 단위로 표시될 것을 지정합니다. OCTETS는 *code-units*가 바이트 단위로 표시됨을 지정합니다.

문자열 단위가 명시적으로 지정되지 않으면 결과의 데이터 유형은 사용 단위를 결정합니다. 결과가 그래픽 데이터이면 *code-units*는 2바이트 단위로 표현되고, 그 외의 경우는 바이트로 표현됩니다. CODEUNITS16, CODEUNITS32 및 OCTETS에 대한 자세한 정보는 『문자열』에서 『내장 함수의 문자열 단위』를 참조하십시오.

함수의 결과는 *string-expression*이 CHAR 또는 VARCHAR인 경우 VARCHAR이고, *string-expression*이 GRAPHIC 또는 VARGRAPHIC인 경우 VARGRAPHIC입니다.

결과의 길이 속성은 다음 테이블에 표시된 것처럼 *code-units*의 내재된 또는 명시적 값, 내재된 또는 명시적 문자열 단위, 그리고 결과 데이터 유형으로 결정됩니다.

표 59. 문자열 단위 및 결과 유형 함수로서의 UPPER 결과의 길이 속성

문자열 단위	문자 결과 유형	그래픽 결과 유형
CODEUNITS16	MIN(*code-units* * 3, 32768)	*code-units*
CODEUNITS32	MIN(*code-units* * 4, 32768)	MIN(*code-units* * 2, 16336)
OCTETS	*code-units*	MIN(*code-units* / 2, 16336)

결과의 실제 길이는 *string-expression*의 길이보다 길 수 있습니다. 결과의 실제 길이가 결과의 길이 속성보다 길면 오류가 리턴됩니다(SQLSTATE 42815). 결과의 코드 단위 수가 *code-units*의 값을 초과하면 오류가 리턴됩니다(SQLSTATE 42815).

*string-expression*이 UTF-16으로 되어 있지 않으면, 이 함수는 *string-expression*에서 UTF-16으로, 그리고 UTF-16에서 *string-expression*의 코드 페이지로의 변환을 수행합니다. 코드 페이지 변환의 결과에 최소 하나의 대체 문자가 포함되어 있으면 결과에는 대체 문자가 포함되고, 경고가 리턴되며(SQLSTATE 01517) SQLCA의 경고 플래그 SQLWARN8이 'W'로 설정됩니다.

첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블에서 컬럼 JOB 값의 문자가 대문자로 리턴되도록 하십시오.

```
SELECT UPPER(JOB, 'en_US')
  FROM EMPLOYEE
  WHERE EMPNO = '000020'
```

  결과는 값 'MANAGER'입니다.

- 터키어 문자열에서 모든 'I' 문자에 대한 대문자를 찾으십시오.

```
VALUES UPPER('I□ıi', 'tr_TR', CODEUNITS16)
```

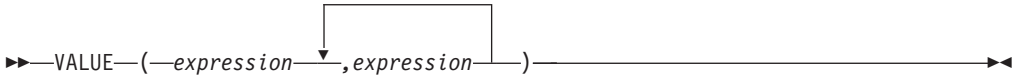
  결과는 문자열 'I□I□'입니다.

- 독일어 'ß'([@185e] S)의 대문자 양식을 찾으십시오.

```
VALUES UPPER('ß', 'de', 2, CODEUNITS16)
```

  결과는 문자열 'SS'입니다. 이 예에서 *code-units*를 지정해야 합니다. 결과에 *string-expression*에서 보다 많은 코드 단위가 있기 때문입니다.

VALUE



스키마는 SYSIBM입니다.

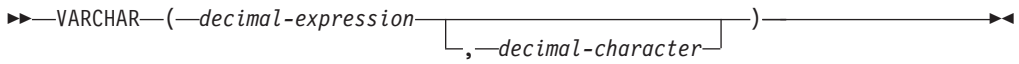
VALUE 함수는 널(NULL)이 아닌 첫 번째 인수를 리턴합니다.

VALUE는 COALESCE의 동의어입니다.

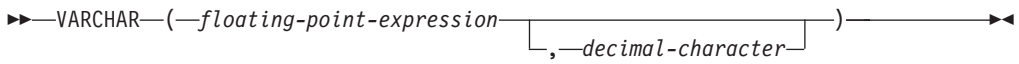
VARCHAR

정수를 varchar로

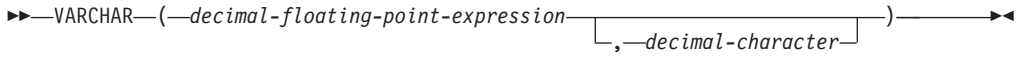
10진수를 varchar로



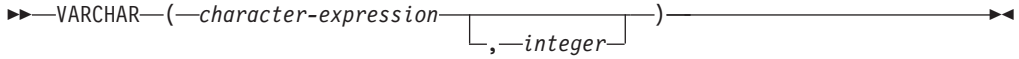
부동 소수점을 varchar로



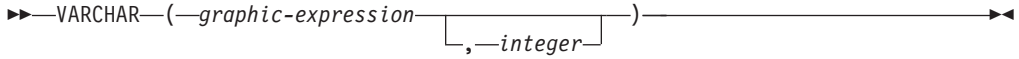
10진수 부동 소수점을 varchar로



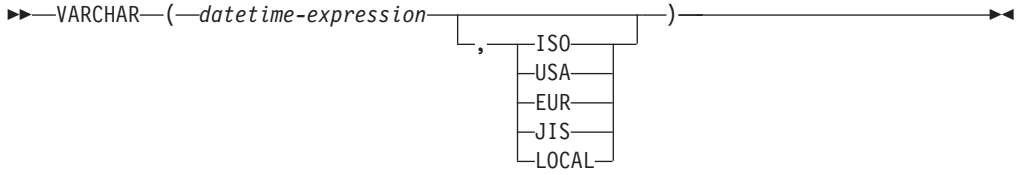
문자를 varchar로



그래픽을 varchar로



날짜 시간을 varchar로



스키마는 SYSIBM입니다. 함수 시그니처에서 키워드를 사용할 때 함수 이름을 규정된 이름으로 지정할 수 없습니다.

VARCHAR 함수는 다음의 가변 길이 문자 표현을 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우, 정수
- 첫 번째 인수가 10진수인 경우, 10진수

- 첫 번째 인수가 DOUBLE 또는 REAL인 경우, 배정밀도 부동 소수점 숫자
- 첫 번째 인수가 (DECFLOAT)인 경우, 10진수 부동 소수점 숫자
- 첫 번째 인수가 임의의 문자열 유형인 경우 문자열
- 첫 번째 인수가 임의의 그래픽 문자열(Unicode 데이터베이스 전용) 유형인 경우 그래픽 문자열
- 첫 번째 인수가 날짜, 시간 또는 시간소인인 경우, 날짜 시간 값

유니코드 데이터베이스에서는 다중 바이트 문자를 통해 출력 문자열이 파트별로 절단됩니다.

- 입력이 문자열인 경우 부분 문자가 하나 이상의 공백으로 대체됨
- 입력이 그래픽 문자열인 경우 부분 문자가 비어 있는 문자열로 대체됨

추후 릴리스에서 변경될 수 있으므로 이러한 동작의 영향을 받지 않는 것이 좋습니다.

함수의 결과는 가변 길이 문자열입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

정수를 varchar로

integer-expression
데이터 유형이 정수(SMALLINT, INTEGER 또는 BIGINT)인 값을 리턴하는 표현식입니다.

결과는 SQL 정수 상수의 양식으로 된 *integer-expression*의 가변 길이의 문자열 표현입니다. 결과의 길이 속성은 다음과 같이 *integer-expression*이 작은, 대형 또는 큰 정수인가에 따라 다릅니다.

- 첫 번째 인수가 작은 정수(small integer)인 경우 결과의 최대 길이는 6입니다.
- 첫 번째 인수가 큰 정수(large integer)인 경우 결과의 최대 길이는 11입니다.
- 첫 번째 인수가 큰 정수(big integer)인 경우 결과의 최대 길이는 20입니다.

결과의 실제 길이는 인수의 값을 표현하는 데 사용할 수 있는 최소 문자 수입니다. 앞에 영(0)이 포함되지 않습니다. 인수가 음수이면 결과의 첫 번째 문자는 빼기 기호입니다. 그 외의 경우, 첫 번째 문자는 숫자입니다.

결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

10진수를 varchar로

decimal-expression
10진수 데이터 유형인 값을 리턴하는 표현식입니다. DECIMAL 스칼라 함수는 정밀도와 스케일을 변경하기 위해 사용될 수 있습니다.

decimal-character

결과 문자열에서 10진 숫자를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 상수의 양식으로 된 *decimal-expression*의 가변 길이의 문자열 표현입니다. 결과의 길이 속성은 2 + *p*입니다. 여기서, *p*는 *decimal-expression*의 정밀도입니다. 결과의 실제 길이는 뒤에 영(0)이 포함된다는 점을 제외하고는 결과를 표현하는 데 사용할 수 있는 최소 문자 수입니다. 앞에 영(0)이 포함되지 않습니다. decimal-expression이 음수이면 결과의 첫 문자는 빼기 기호이고, 그렇지 않으면 첫 문자가 숫자이거나 10진수 문자입니다. *decimal-expression*의 스케일이 0이면, 10진수 문자가 리턴되지 않습니다.

결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

부동 소수점을 varchar로

floating-point-expression

부동 소수점 데이터 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식입니다.

decimal-character

결과 문자열에서 10진 숫자를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *floating-point-expression*의 가변 길이의 문자열 표현입니다.

결과의 최대 길이는 24입니다. 가수(mantissa)가 영(0)이 아닌 단일 숫자, *decimal-character*와 일련의 숫자로 구성될 수 있도록 floating-point-expression의 값을 나타낼 수 있는 최소 문자 수가 결과의 실제 길이가 됩니다. *floating-point-expression*이 음수이면 결과의 첫 문자는 빼기 기호이고, 그렇지 않으면 첫 문자가 숫자입니다. *floating-point-expression*이 0이면, 결과는 0E0입니다.

결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

10진수 부동 소수점을 varchar로

decimal-floating-point-expression

부동 소수점 데이터 유형(DOUBLE)인 값을 리턴하는 표현식입니다.

decimal-character

결과 문자열에서 10진 숫자를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *decimal-floating-point-expression*의 가변 길이 문자열 표현입니다. 결과의 최대 길이는 42입니다. 결과는 *decimal-floating-point-expression*의 값을 나타낼 수 있는 최소 문자 수가 결과의 실제 길이가 됩니다. *decimal-floating-point-expression*이 음수이면, 결과의 첫 문자는 빼기 기호이고, 그렇지 않으면 첫 문자가 숫자입니다. *decimal-floating-point-expression*이 0이면 결과는 0입니다.

*decimal-floating-point-expression*의 값이 특수 값 Infinity, sNaN 또는 NaN인 경우, 문자열 『INFINITY』, 『SNAN』 및 『NAN』이 각각 리턴됩니다. 특수 값이 음수이면 결과의 첫 문자는 빼기 기호입니다. 문자열로 변환될 때 10진수 부동 소수점 특수 값 sNaN은 경고를 발생시키지 않습니다.

결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

문자를 varchar로

character-expression
값을 리턴하는 표현식은 내장 문자열 데이터 유형(CHAR, VARCHAR 또는 CLOB)입니다.

integer
결과로 발생하는 가변 길이 문자열의 길이 속성입니다. 값은 0 - 32 672 범위여야 합니다.

두 번째 인수가 지정되지 않은 경우:

- *character-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다.

결과의 실제 길이는 결과의 최소 길이 속성이며 *character-expression*의 실제 길이입니다. *character-expression*의 길이가 결과의 길이 속성보다 크면, 결과가 절단됩니다. 절단된 문자가 모두 공백이고 *character-expression*이 CLOB가 아닌 경우를 제외하고는 경고가 리턴됩니다(SQLSTATE 01004).

그래픽을 varchar로

graphic-expression
내장 그래픽 문자열 데이터 유형(GRAPHIC, VARGRAPHIC 또는 DBCLOB)인 값을 리턴하는 표현식입니다.

integer
결과로 발생하는 가변 길이 문자열의 길이 속성입니다. 값은 0 - 32 672 범위여야 합니다.

두 번째 인수가 지정되지 않은 경우:

- *graphic-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 그 외의 경우, 결과의 길이 속성은 첫 번째 인수의 3배 길이 속성과 같습니다.

결과의 실제 길이는 결과의 최소 길이 속성이며 *graphic-expression*의 실제 길이입니다. *graphic-expression*의 길이가 결과의 길이 속성보다 크면, 경고를 리턴하지 않고 결과가 절단됩니다.

날짜 시간을 varchar로

datetime-expression
다음 데이터 유형 중 하나인 표현식입니다.

DATE
결과는 두 번째 인수가 지정하는 형식으로 된 날짜의 문자열 표현입니다. 결과 길이는 10입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않은 경우 오류가 발생합니다(SQLSTATE 42703).

TIME 결과는 두 번째 인수가 지정하는 형식으로 된 시간의 문자열 표현입니다. 결과의 길이는 8입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않으면 오류가 발생합니다(SQLSTATE 42703).

TIMESTAMP
결과는 시간소인의 문자열 표현입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(0)이면 결과의 길이는 19입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(*n*)이고 *n*이 1과 12 사이인 경우, 결과의 길이는 20+*n*입니다. 그렇지 않으면 결과의 길이는 26입니다. 두 번째 인수는 지정하지 마십시오(SQLSTATE 42815).

결과의 코드 페이지는 해당 섹션의 코드 페이지입니다.

참고:

- 첫 번째 인수가 숫자이거나 첫 번째 인수가 문자열이고 길이 인수가 지정되어 있을 때 CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.
- 실행 파일 문자열이 함수에 대해 첫 번째 인수로 허용되며 결과로 발생되는 가변 길이 문자열은 FOR BIT DATA 문자열입니다.

예를 들면, 다음과 같습니다.

- *예 1:* 길이가 10인 EMPNO 가변 길이 작성.

```
SELECT VARCHAR(EMPNO,10)
  INTO :VARHV
  FROM EMPLOYEE
```

- *예 2:* VARCHAR(8)로 정의된 호스트 변수 JOB_DESC를 Dolores Quintana 사원에 대해 CHAR(8)로 정의된 VARCHAR과 동등한 작업 설명(JOB 컬럼의 값)으로 설정하십시오.

```
SELECT VARCHAR(JOB)
  INTO :JOB_DESC
  FROM EMPLOYEE
  WHERE LASTNAME = 'QUINTANA'
```

- *예 3:* EDLEVEL 컬럼은 SMALLINT로 정의되어 있습니다. 다음은 가변 길이 문자열로 값을 리턴합니다.

```
SELECT VARCHAR(EDLEVEL)
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

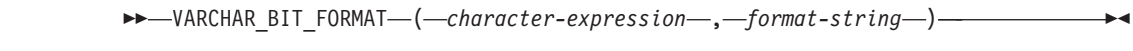
  결과 값은 '18'입니다.

- *예 4:* SALARY 컬럼 및 COMN 컬럼은 스케일 2 및 정밀도 9인 DECIMAL로 정의되어 있습니다. 쉼표 10진수 문자를 사용하여 직원 Haas의 총 수입을 리턴합니다.

```
SELECT VARCHAR(SALARY + COMM, ',')
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  결과 값은 '56970,00 '입니다.

VARCHAR_BIT_FORMAT



스키마는 SYSIBM입니다.

VARCHAR_BIT_FORMAT 함수는 문자 템플리트를 사용하여 형식화된 문자열의 비트 문자열 표현을 리턴합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

character-expression

CLOB가 아닌 내장 문자열인 값을 리턴하는 표현식입니다(SQLSTATE 42815). 필수 길이는 지정된 형식 문자열과 값 해석 방법으로 판별됩니다.

format-string

>*character-expression*의 바이트가 해석되는 방법에 대한 템플리트를 포함하는 문자 상수입니다.

유효한 형식 문자열에는 `'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'` 및 `'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'`이 포함됩니다(SQLSTATE 42815). 여기서 각 'x' 또는 'X'는 결과에서 하나의 16진수에 해당됩니다.

함수의 결과는 길이 속성 및 실제 길이가 형식 문자열을 기초로 하는 가변 길이 문자열 FOR BIT DATA입니다. 위에 나열된 유효한 두 형식 문자열의 경우, 결과의 길이 속성은 36이고 실제 값은 16입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)인 경우, 결과는 널(NULL)이 됩니다.

예:

- 범용 고유 ID를 해당되는 2진 형식으로 표시하십시오.

```
VARCHAR_BIT_FORMAT('d83d6360-1818-11db-9804-b622a1ef5492',
                   'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  리턴되는 결과는 다음과 같습니다.

```
x'D83D6360181811DB9804B622A1EF5492'
```

- 범용 고유 ID를 해당되는 2진 형식으로 표시하십시오.

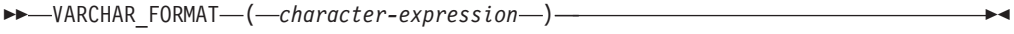
```
VARCHAR_BIT_FORMAT('D83D6360-1818-11DB-9804-B622A1EF5492',
                   'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

  리턴되는 결과는 다음과 같습니다.

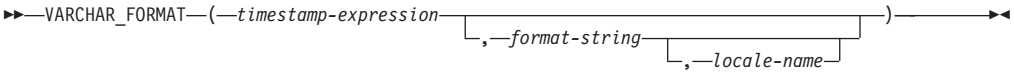
```
x'D83D6360181811DB9804B622A1EF5492'
```

VARCHAR_FORMAT

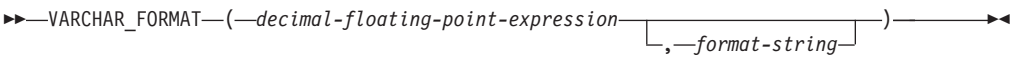
문자를 varchar로



varchar 시간소인:



varchar 10진 부동 소수점:



스키마는 SYSIBM입니다.

VARCHAR_FORMAT 함수는 지정된 출력 문자열 인수를(제공된 경우) 첫 번째 인수 값에 적용하여 문자열을 리턴합니다. VARCHAR_FORMAT 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.

지정된 문자 템플리트에 따라 표현식을 형식화해야 합니다.

문자를 varchar로

character-expression
내장 CHAR 또는 VARCHAR 데이터 유형이어야 하는 값을 리턴하는 표현식. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다.

결과가 인수의 길이 속성과 일치하는 길이 속성이 있는 VARCHAR입니다. 결과 값은 *character-expression*의 값과 동일합니다.

varchar 시간소인

timestamp-expression
날짜 또는 시간소인이어야 하거나 CLOB 또는 DBCLOB가 아닌 날짜 또는 시간소인의 유효한 문자열 표현인 값을 리턴하는 표현식. 인수가 문자열이면 *format-string* 인수를 지정해야 합니다. 유니코드 데이터베이스의 경우, 제공된 인수가 데이터, 시간 또는 시간소인의 그래픽 문자열 표현이면, 함수를 평가하기 전에 먼저 문자열로 변환됩니다.

*timestamp-expression*이 날짜 또는 날짜의 유효한 문자열 표현이면, 자정 정각(00.00.00)을 가정하여 먼저 TIMESTAMP(0) 값으로 변환됩니다.

날짜 시간 값의 유효한 문자열 표현 형식은 『날짜 시간 값』의 『날짜 시간 값 문자열 표현』을 참조하십시오.

format-string

표현식은 내장 CHAR, VARCHAR, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR 또는 VARCHAR 데이터 유형이 아닌 경우, 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다. 실제 길이는 254바이트를 넘지 않아야 합니다(SQLSTATE 22007). 값은 *timestamp-expression* 형식화 방법에 대한 템플리트입니다.

유효한 *format-string*은 아래 나열된 형식 요소 조합을 포함해야 합니다(SQLSTATE 22007). 하나 이상의 다음 구분자 문자로 두 형식 요소를 선택적으로 구분할 수 있습니다.

- 마이너스 부호(-)
- 마침표(.)
- 슬래시(/)
- 쉼표(,)
- 어포스트로피 (')
- 세미콜론(;)
- 콜론(:)
- 공백( )

또한 *format-string*의 시작 또는 끝에 구분자 문자를 지정할 수 있습니다.

표 60. VARCHAR_FORMAT 함수의 형식 요소

형식 요소	설명
AM 또는 PM	마침표가 없는 정오 표시기(아침 또는 저녁). 이 형식 요소는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
A.M. 또는 P.M.	마침표가 있는 정오 표시기(아침 또는 저녁). 이 형식 요소는 완전 문자열 'A.M.' 또는 'P.M.'을 사용합니다. 적용되는 로케일 이름과 독립적입니다.
CC	세기(01 - 99). 4자리 연도의 마지막 두 숫자가 0이면, 결과는 연도의 처음 두 숫자입니다. 그렇지 않으면, 결과는 연도의 처음 두 숫자 + 1입니다.

표 60. VARCHAR_FORMAT 함수의 형식 요소 (계속)

형식 요소	설명
DAY, Day 또는 day	대문자, 첫 글자만 대문자 또는 소문자 형식의 일 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
DY, Dy 또는 dy	대문자, 첫 글자만 대문자 또는 소문자 형식의 약어화된 일 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
D	요일(1 - 7). 첫 번째 요일은 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
DD	월 중 일(01 - 31).
DDD	연 중 일(001 - 366).
FF 또는 FF*n*	소수 초(0 - 999999999999). 숫자 *n*은 *string-expression*에서 예측된 숫자 수를 지정하는 데 사용됩니다. *n*에 유효한 값은 선행 영(0)이 없는 1 - 12입니다. FF 지정은 FF6 지정과 동일합니다. *timestamp-expression*의 시간소인 정밀도의 형식으로 지정된 것보다 작으면 지정된 숫자의 오른쪽에 영(0) 숫자가 채워집니다.
HH	HH는 HH12와 동일하게 동작합니다.
HH12	12시간 형식의 하루 중 시각(01 - 12).
HH24	24시간 형식의 하루 중 시각(00 - 24).
IW	ISO 연 중 주(01 - 53). 주는 월요일에 시작하고 7일을 포함합니다. 주 1은 목요일을 포함하는 연 중 첫 번째 주이며, 1월 4일을 포함하는 연 중 첫 번째 주와 같습니다.
I	ISO 연도(0 - 9). 리턴되는 ISO 주를 기반으로 하는 연도의 마지막 숫자.
IY	ISO 연도(00 - 99). 리턴되는 ISO 주를 기반으로 하는 연도의 마지막 두 숫자.
IYY	ISO 연도(000 - 999). 리턴되는 ISO 주를 기반으로 하는 연도의 마지막 세 숫자.
IYYY	ISO 연도(0000 - 9999). 리턴되는 ISO 주를 기반으로 하는 4자리 연도.
J	율리우스력 일(BC 4713년 1월 1일 이후의 경과일 수).
MI	분(00 - 59).
MM	월(01 - 12).
NNNNNN	마이크로초(000000 - 999999). FF6과 같습니다.

표 60. VARCHAR_FORMAT 함수의 형식 요소 (계속)

형식 요소	설명
MONTH, Month 또는 month	대문자, 첫 글자만 대문자 또는 소문자 형식의 월 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
MON, Mon 또는 mon	대문자, 첫 글자만 대문자 또는 소문자 형식의 약어화된 월 이름. 사용되는 언어는 *locale-name*(지정된 경우)에 종속됩니다. 그렇지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
Q	분기(1 - 4). 1월에서 3월까지는 1을 리턴합니다.
RR	RR은 YY와 동일하게 동작합니다.
RRRR	RRRR은 YYYY와 동일하게 동작합니다.
SS	초(00 - 59).
SSSSS	지난 자정 이후의 초 수(00000 - 86400).
W	월 중 주(1 - 5). 주 1은 그 달의 첫 번째 날에 시작하고 7번째 날에 끝납니다.
WW	연 중 주(01 - 53). 주 1은 1월 1일에 시작하고 1월 7일에 끝납니다.
Y	연도의 마지막 숫자(0 - 9).
YY	연도의 마지막 두 숫자(00 - 99).
YYY	연도의 마지막 세 숫자(000 - 999).
YYYY	4자리 연도(0000 - 9999).

주: 671 페이지의 표 60의 형식 요소는 다음 경우를 제외하고 대소문자를 구분하지 않습니다.

- AM, PM
- A.M., P.M.
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

형식 요소가 앰비규어스한 경우, 대소문자 비구분 형식 요소를 먼저 고려합니다. 예를 들어, DDYYYY는 D 다음에 DY가 오고 그 다음에 YYY가 오는 것으로 해석되지 않고 DD 다음에 YYYY가 오는 것으로 해석됩니다.

*format-string*이 지정되지 않으면, 사용되는 형식은 CURRENT LOCALE LC_TIME 특수 레지스터 값을 기반합니다.

locale-name

다음 형식 요소에 사용되는 로케일을 지정하는 문자 상수.

- AM, PM
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

locale-name 값은 대소문자를 구분하지 않으며 유효한 로케일이어야 합니다(SQLSTATE 42815). 유효한 로케일 및 로케일 이름 지정에 대한 정보는 *자국어 안내서* 의 『SQL 및 XQuery의 로케일 이름』을 참조하십시오. *locale-name*이 지정되지 않으면, CURRENT LOCALE LC_TIME 특수 레지스터 값이 사용됩니다.

결과는 *format-string*으로 지정된 형식의 *timestamp-expression* 표현입니다. *format-string*은 하나 이상의 구분자 문자로 선택적으로 구분할 수 있는 일련의 형식 요소로 해석됩니다. *format-string*의 문자열은 671 페이지의 표 60의 가장 긴 일치하는 형식 요소로 해석됩니다. 동일한 문자를 포함하는 두 형식 요소가 구분자 문자로 구분되지 않은 경우, 스펙은 왼쪽부터 시작하여 테이블에서 가장 긴 일치하는 형식 요소 스펙으로 해석되고 나머지 출력 문자열에 대해 일치를 찾을 때까지 계속됩니다. 예를 들어, 'YYYYYYDD'는 'YYYY', 'YY' 및 'DD' 형식 요소로 해석됩니다.

결과는 가변 길이 문자열입니다. 길이 속성은 254입니다. *format-string*은 결과의 실제 길이를 판별합니다. 결과 문자열이 결과의 길이 속성을 초과하면, 결과가 절단됩니다.

varchar 부동 소수점

decimal-floating-point-expression

임의의 내장 숫자 데이터 유형 값을 리턴하는 표현식입니다. 인수가 10진 부동 소수점 값이 아니면, 처리를 위해 DECFLOAT(34)로 변환됩니다.

format-string

표현식은 내장 CHAR, VARCHAR, 숫자 또는 날짜 및 시간 데이터 유형인 값을 리턴해야 합니다. 값이 CHAR 또는 VARCHAR 데이터 유형이 아닌 경우, 함수를 평가하기 전에 내재적으로 VARCHAR로 캐스트됩니다. 유니코드 데이터베이스의 경우, 제공된 인수가 GRAPHIC 또는 VARGRAPHIC 데이터 유형이면, 함수를 평가하기 전에 먼저 VARCHAR로 변환됩니다. 실제 길이는 254바이트를 넘지 않아야 합니다(SQLSTATE 22018). 값은 *decimal-floating-point-expression* 형식화 방법에 대한 템플리트입니다. 템플리트 처음에서만 접두부로 지정된 형식 요소를 사용할 수 있습니다. 템플리트 끝에서만 접미부로 지정된 형식 요소를 사용할 수 있

습니다. 형식 요소는 대소문자를 구분합니다. 템플리트는 MI, S 또는 PR 형식 요소를 두 개 이상 포함하지 않아야 합니다(SQLSTATE 22018).

표 61. VARCHAR_FORMAT 함수의 형식 요소

형식 요소	설명
0	각 0은 유효 숫자를 나타냅니다. 숫자에서 선행 영(0)은 영(0)으로 표시됩니다.
9	각 9는 유효 숫자를 나타냅니다. 숫자에서 선행 영(0)은 공백으로 표시됩니다.
MI	접미부: *decimal-floating-point-expression*이 음수이면, 후행 마이너스 부호(–)가 결과에 포함됩니다. *decimal-floating-point-expression*이 양수이면, 뒤 공백이 결과에 포함됩니다.
S	접두부: *decimal-floating-point-expression*이 음수이면, 선행 마이너스 부호(–)가 결과에 포함됩니다. 접두부: *decimal-floating-point-expression*이 양수이면, 선행 플러스 부호(+)가 결과에 포함됩니다.
PR	접미부: *decimal-floating-point-expression*이 음수이면, 선행 보다 작음 문자(<) 및 후행 보다 큼 문자(>)가 결과에 포함됩니다. *decimal-floating-point-expression*이 양수이면, 앞 공백과 뒤 공백이 결과에 포함됩니다.
$	접두부: 선행 달러 기호($)가 결과에 포함됩니다.
,	결과에서 해당 위치에 쉼표가 포함되도록 지정합니다. 이 쉼표는 그룹 구분자로 사용됩니다.
.	결과에서 해당 위치에 마침표가 포함되도록 지정합니다. 이 마침표는 소수점으로 사용됩니다.

format-string이 지정되지 않으면, *decimal-floating-point-expression*은 SQL 10진 부동 소수점 상수 양식으로 형식화됩니다. *decimal-floating-point-expression*이 음수이면, 결과의 첫 번째 문자는 마이너스 부호이고, 그렇지 않으면 첫 번째 문자는 숫자입니다. *decimal-floating-point-expression*이 영(0)이면, 결과가 0입니다.

결과는 *decimal-floating-point-expression*의 가변 길이 문자열 표현입니다. 길이 속성은 254입니다. 결과의 실제 길이는 *format-string*(지정된 경우)이 판별합니다. 그렇지 않으면, 결과의 실제 길이는 *decimal-floating-point-expression* 값을 나타낼 수 있는 가장 작은 문자 수입니다. 결과 문자열이 결과의 길이 속성을 초과하면, 결과가 절단됩니다.

decimal-floating-point-expression 값이 특수 값인 Infinity, sNaN 또는 NaN이면, 각각 ″INFINITY″, ″SNAN″ 및 ″NAN″ 문자열이 리턴됩니다. 특수 값이 음수이면, 결과의 첫 문자는 마이너스 부호입니다. 10진 부동 소수점 특수 값 sNaN은 문자열로 변환될 때 예외가 발생하지 않습니다.

*format-string*이 형식 요소 MI, S 또는 PR을 포함하지 않고 *decimal-floating-point-expression* 값이 음수이면, 마이너스 부호(–)가 결과에 포함됩니다. 그렇지 않으면, 결과에 공백이 포함됩니다.

*decimal-floating-point-expression*의 소수점 왼쪽의 숫자 수가 *format-string*의 소수점 왼쪽의 숫자 수보다 크면, 결과는 번호 기호(#) 문자입니다. *decimal-floating-point-*

*expression*의 소수점 오른쪽의 숫자 수가 *format-string*의 소수점 오른쪽의 숫자 수보다 크면, 결과는 *decimal-floating-point-expression*이 *format-string*의 소수점 오른쪽의 숫자 수로 반올림됩니다. DECFLOAT 반올림 모드가 사용되지 않습니다. VARCHAR_FORMAT의 반올림 동작은 ROUND_HALF_UP 값에 대응합니다.

결과의 코드 페이지는 섹션의 코드 페이지입니다.

참고:

- *율리우스력 및 그레고리오력*: varchar 시간소인의 경우, 이 함수에서는 1582년 10월 15일에 율리우스력에서 그레고리오력으로의 전이를 고려합니다.
- *결정론:* VARCHAR_FORMAT은 결정 함수입니다. 그러나 다음 함수 호출은 CURRENT LOCALE LC_TIME 특수 레지스터 값에 종속됩니다.
  - varchar 시간소인(format-string이 명시적으로 지정되지 않거나 locale-name이 명시적으로 지정되지 않고 다음 중 하나가 참인 경우)
    - *format-string*이 상수가 아님
    - *format-string*이 상수이고 로케일 구분 형식 요소를 포함

    특수 레지스터를 사용할 수 없는 경우에는 특수 레지스터 값에 종속된 이러한 호출을 사용할 수 없습니다(SQLSTATE 42621 또는 428EC).
- *구문 대체:* TO_CHAR은 VARCHAR_FORMAT의 동의어입니다.

예를 들면, 다음과 같습니다.

- 이름이 'SYSU'로 시작하는 모든 시스템 테이블에 대한 테이블 이름 및 작성시간소인을 표시하십시오.

```
SELECT VARCHAR(TABNAME, 20) AS TABLE_NAME,
  VARCHAR_FORMAT(CREATE_TIME, 'YYYY-MM-DD HH24:MI:SS')
    AS CREATION_TIME
  FROM SYSCAT.TABLES
  WHERE TABNAME LIKE 'SYSU%'
```

  이 예에서는 다음을 리턴합니다.

```
    TABLE_NAME          CREATION_TIME
-------------------- -------------------
    SYSUSERAUTH         2000-05-19 08:18:56
    SYSUSEROPTIONS      2000-05-19 08:18:56
```

- TMSTAMP 변수가 TIMESTAMP로 정의되고 2007-03-09-14.07.38.123456 값을 갖는다고 가정하십시오. 다음 예는 함수의 여러 호출 및 결과로 발생되는 문자열 값을 나타냅니다. 각 경우의 결과 데이터 유형은 VARCHAR(254)입니다.

```
Function invocation                              Result
-------------------                              ------
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMISSFF3')  20070309020738123
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHH24MISS')   20070309140738
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMI')       200703090207
VARCHAR_FORMAT(TMSTAMP,'DD/MM/YY')           09/03/07
```

```
VARCHAR_FORMAT(TMSTAMP,'MM-DD-YYYY')        03-09-2007
VARCHAR_FORMAT(TMSTAMP,'J')                 2454169
VARCHAR_FORMAT(TMSTAMP,'Q')                 1
VARCHAR_FORMAT(TMSTAMP,'W')                 2
VARCHAR_FORMAT(TMSTAMP,'IW')                10
VARCHAR_FORMAT(TMSTAMP,'WW')                10
VARCHAR_FORMAT(TMSTAMP,'Month','en_US')     March
VARCHAR_FORMAT(TMSTAMP,'MONTH','en_US')     MARCH
VARCHAR_FORMAT(TMSTAMP,'MON','en_US')       MAR
VARCHAR_FORMAT(TMSTAMP,'Day','en_US')       Friday
VARCHAR_FORMAT(TMSTAMP,'DAY','en_US')       FRIDAY
VARCHAR_FORMAT(TMSTAMP,'Dy','en_US')        Fri
VARCHAR_FORMAT(TMSTAMP,'Month','de_DE')     März
VARCHAR_FORMAT(TMSTAMP,'MONTH','de_DE')     MÄRZ
VARCHAR_FORMAT(TMSTAMP,'MON','de_DE')       MRZ
VARCHAR_FORMAT(TMSTAMP,'Day','de_DE')       Freitag
VARCHAR_FORMAT(TMSTAMP,'DAY','de_DE')       FREITAG
VARCHAR_FORMAT(TMSTAMP,'Dy','de_DE')        Fr
```

- DTE 변수가 DATE로 정의되고 2007-03-09 값을 갖는다고 가정하십시오. 다음 예는 몇 개의 함수 호출과 결과 문자열 값을 표시합니다. 각 경우의 결과 데이터 유형은 VARCHAR(254)입니다.

```
Function invocation                         Result
------------------                          ------
VARCHAR_FORMAT(DTE,'YYYYMMDD')              20070309
VARCHAR_FORMAT(DTE,'YYYYMMDDHH24MISS')      20070309000000
```

- POSNUM 및 NEGNUM 변수가 DECFLOAT(34)로 정의되고 각각 1234.56과 -1234.56 값을 갖는다고 가정하십시오. 다음 예는 몇 개의 함수 호출과 결과 문자열 값을 표시합니다. 각 경우의 결과 데이터 유형은 VARCHAR(254)입니다.

```
Function invocation                         Result
------------------                          ------
VARCHAR_FORMAT(POSNUM)                      '1234.56'
VARCHAR_FORMAT(NEGNUM)                      '-1234.56'
VARCHAR_FORMAT(POSNUM,'9999.99')            '1234.56'
VARCHAR_FORMAT(NEGNUM,'9999.99')            '1234.56'
VARCHAR_FORMAT(POSNUM,'99999.99')           ' 1234.56'
VARCHAR_FORMAT(NEGNUM,'99999.99')           ' 1234.56'
VARCHAR_FORMAT(POSNUM,'00000.00')           '01234.56'
VARCHAR_FORMAT(NEGNUM,'00000.00')           '01234.56'
VARCHAR_FORMAT(POSNUM,'9999.99MI')          '1234.56 '
VARCHAR_FORMAT(NEGNUM,'9999.99MI')          '1234.56-'
VARCHAR_FORMAT(POSNUM,'S9999.99')           '+1234.56'
VARCHAR_FORMAT(NEGNUM,'S9999.99')           '-1234.56'
VARCHAR_FORMAT(POSNUM,'9999.99PR')          ' 1234.56 '
VARCHAR_FORMAT(NEGNUM,'9999.99PR')          '<1234.56>'
VARCHAR_FORMAT(POSNUM,'S$9,999.99')         '+$1,234.56'
VARCHAR_FORMAT(NEGNUM,'S$9,999.99')         '-$1,234.56'
```

VARCHAR_FORMAT_BIT

스키마는 SYSIBM입니다.

VARCHAR_FORMAT_BIT 함수는 문자 템플리트를 사용하여 형식화된 비트 문자열의 문자 표현을 리턴합니다.

bit-data-expression
내장 문자열 FOR BIT DATA 데이터 유형인 값을 리턴하는 표현식입니다 (SQLSTATE 42815). 필수 길이는 지정된 형식 문자열과 값 해석 방법으로 판별됩니다.

format-string
결과가 형식화되는 방법에 대한 템플리트가 들어 있는 문자 상수입니다.

유효한 출력 문자열에는 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' 및 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'이 포함됩니다(SQLSTATE 42815). 여기서 각 'x' 또는 'X'는 *bit-data-expression*의 하나의 16진수 숫자에 해당합니다.

함수의 결과는 출력 문자열을 기초로 하는 길이 속성 및 실제 길이를 갖는 가변 길이 문자열입니다. 위에 나열되는 두 가지 유효한 출력 문자열의 경우 길이 속성은 36이고 실제 길이는 36바이트입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)인 경우, 결과는 널(NULL)이 됩니다.

예:

- 범용 고유 ID를 해당되는 형식화된 양식으로 표시하십시오.

```
VARCHAR_FORMAT_BIT(cast(x'd83d6360181811db9804b622a1ef5492'
                     as varchar(16) for bit data),
                  'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  리턴되는 결과는 다음과 같습니다.

```
'd83d6360-1818-11db-9804-b622a1ef5492'
```

- 범용 고유 ID를 해당되는 형식화된 양식으로 표시하십시오.

```
VARCHAR_FORMAT_BIT(cast(x'd83d6360181811db9804b622a1ef5492'
                     as char(16) for bit data),
                  'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

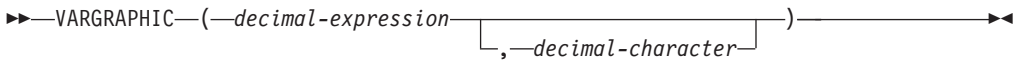
  리턴되는 결과는 다음과 같습니다.

```
'D83D6360-1818-11DB-9804-B622A1EF5492'
```

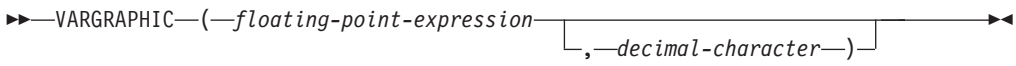
VARGRAPHIC

정수를 vargraphic으로:

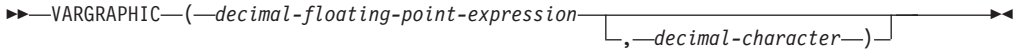
10진수를 vargraphic으로:



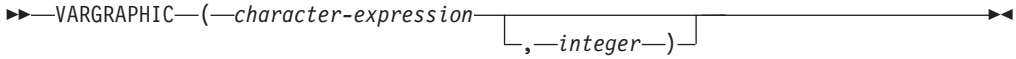
부동 소수점을 vargraphic으로:



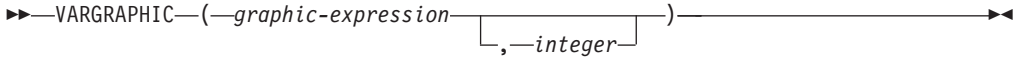
10진수 부동 소수점을 vargraphic으로:



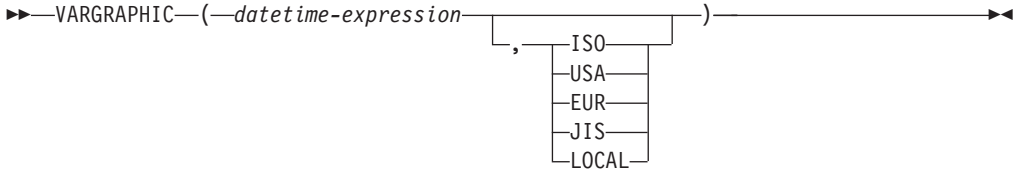
문자를 vargraphic으로:



그래픽을 vargraphic으로:



날짜 시간을 vargraphic으로:



스키마는 SYSIBM입니다. 함수 시그니처에서 키워드를 사용할 때 함수 이름을 규정된 이름으로 지정할 수 없습니다.

VARGRAPHIC 함수는 다음과 같은 가변 길이 그래픽 문자열 표현을 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우, 정수(유니코드 데이터베이스 전용)

- 첫 번째 인수가 10진수인 경우, 10진수(유니코드 데이터베이스 전용)
- 첫 번째 인수가 DOUBLE 또는 REAL인 경우, 배정밀도 부동 소수점 숫자(유니코드 데이터베이스 전용)
- 첫 번째 인수가 10진수 부동 소수점 숫자(DECFLOAT)인 경우, 10진수 부동 소수점 숫자(유니코드 데이터베이스 전용)
- 첫 번째 인수가 임의의 문자열 유형인 경우 1바이트 문자를 2바이트 문자로 변환한 문자열
- 첫 번째 인수가 임의의 그래픽 문자열 유형인 경우 그래픽 문자열
- 첫 번째 인수가 날짜, 시간 또는 시간소인인 경우 날짜 시간 값(유니코드 데이터베이스 전용)

유니코드 데이터베이스의 경우 제공된 인수가 문자열이면 함수를 실행하기 전에 먼저 그래픽 문자열로 변환됩니다. 출력 문자열이 절단되면 마지막 문자는 바뀔 가능성이 높으며 이는 공백 문자(X'0020')로 변환됩니다. 추후 릴리스에서 변경될 수 있으므로 이러한 동작의 영향을 받지 않는 것이 좋습니다.

함수의 결과는 가변 길이 그래픽 문자열(VARGRAPHIC 데이터 유형)입니다. 첫 번째 인수가 널(NULL)이 될 수 있으면 결과는 널(NULL)이 될 수 있습니다. 즉, 첫 번째 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

정수를 vargraphic으로

integer-expression
데이터 유형이 정수(SMALLINT, INTEGER 또는 BIGINT)인 값을 리턴하는 표현식입니다.

결과는 SQL 정수 상수의 양식으로 된 *integer-expression*의 가변 길이 그래픽 문자열 표현입니다. 결과의 길이 속성은 다음과 같이 *integer-expression*이 작은, 대형 또는 큰 정수인가에 따라 다릅니다.

- 첫 번째 인수가 작은 정수(small integer)인 경우 결과의 최대 길이는 6입니다.
- 첫 번째 인수가 큰 정수(large integer)인 경우 결과의 최대 길이는 11입니다.
- 첫 번째 인수가 큰 정수(big integer)인 경우 결과의 최대 길이는 20입니다.

결과의 실제 길이는 인수의 값을 표현하는 데 사용할 수 있는 최소 2바이트 문자 수입니다. 앞에 영(0)이 포함되지 않습니다. 인수가 음수이면 결과의 첫 2바이트 문자는 빼기 기호이고, 그 외의 경우는 첫 2바이트 문자는 숫자입니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

10진수를 vargraphic으로

decimal-expression

10진수 데이터 유형인 값을 리턴하는 표현식입니다. DECIMAL 스칼라 함수는 정밀도와 스케일을 변경하기 위해 사용될 수 있습니다.

decimal-character

결과 그래픽 문자열에서 10진 숫자를 분리하는 데 사용되는 2바이트 문자 상수를 지정합니다. 2바이트 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 상수의 양식으로 된 *decimal-expression*의 가변 길이의 그래픽 문자열 표현입니다. 결과의 길이 속성은 2 + *p*입니다. 여기서, *p*는 *decimal-expression*의 정밀도입니다. 결과의 실제 길이는 뒤의 영(0)이 포함된다는 점을 제외하고는 결과를 표현하는 데 사용할 수 있는 최소 2바이트 문자 수입니다. 앞에 영(0)이 포함되지 않습니다. *decimal-expression*이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호이고, 그렇지 않으면 첫 번째 2바이트 문자가 숫자이거나 10진수 문자입니다. *decimal-expression*의 스케일이 0이면, 10진수 문자가 리턴되지 않습니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

부동 소수점을 vargraphic으로

floating-point-expression

부동 소수점 데이터 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식입니다.

decimal-character

결과 그래픽 문자열에서 10진수 숫자를 분리하는 데 사용되는 2바이트 문자 상수를 지정합니다. 2바이트 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *floating-point-expression*의 가변 길이의 그래픽 문자열 표현입니다.

결과의 최대 길이는 24입니다. 가수(mantissa)가 영(0)이 아닌 단일 숫자, *decimal-character*와 일련의 숫자로 구성될 수 있도록 *floating-point-expression*의 값을 나타낼 수 있는 최소 2바이트 문자 수가 결과의 실제 길이가 됩니다. *floating-point-expression*이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호이고, 그렇지 않으면 첫 번째 2바이트 문자가 숫자입니다. *floating-point-expression*이 0이면, 결과는 0E0입니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

10진수 부동 소수점을 vargraphic으로

decimal-floating-point-expression
부동 소수점 데이터 유형(DOUBLE)인 값을 리턴하는 표현식입니다.

decimal-character
결과 그래픽 문자열에서 10진수 숫자를 분리하는 데 사용되는 2바이트 문자 상수를 지정합니다. 2바이트 문자 상수는 숫자, 더하기 부호(+), 빼기 부호(-) 또는 공백이 될 수 없습니다(SQLSTATE 42815). 디폴트값은 마침표(.) 문자입니다.

결과는 SQL 10진수 부동 소수점 상수의 양식으로 된 *decimal-floating-point-expression*의 가변 길이 그래픽 문자열 표현입니다. 결과의 최대 길이는 42입니다. 결과는 *decimal-floating-point-expression*의 값을 나타낼 수 있는 최소 2바이트 문자 수가 결과의 실제 길이가 됩니다. *decimal-floating-point-expression*이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호이고, 그렇지 않으면 첫 번째 2바이트 문자가 숫자입니다. *decimal-floating-point-expression*이 0이면 결과는 0입니다.

*decimal-floating-point-expression*의 값이 특수 값 무한대, sNaN 또는 NaN인 경우, 문자열 G'INFINITY', G'SNAN'및 G'NAN'이 각각 리턴됩니다. 특수 값이 음수이면 결과의 첫 번째 2바이트 문자는 빼기 기호입니다. 문자열로 변환될 때 10진수 부동 소수점 특수 값 sNaN은 경고를 발생시키지 않습니다.

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

문자를 vargraphic으로

character-expression
내장 문자열 데이터 유형(CHAR, VARCHAR 또는 CLOB)인 값을 리턴하는 표현식입니다.

integer
결과로 발생하는 가변 길이 그래픽 문자열의 길이 속성입니다. 값은 0 - 16 336 범위여야 합니다. 두 번째 인수가 지정되지 않은 경우:

- *character-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다.

결과의 실제 길이는 결과의 최소 길이 속성이며 *character-expression*의 실제 길이입니다. *character-expression*의 길이가 결과의 길이 속성보다 크면, 경고가 리턴되지 않고 결과가 절단됩니다.

*character-expression*의 1바이트 문자 각각은 이에 해당하는 2바이트 표현 또는 결과의 2바이트 대체 문자로 변환됩니다. *character-expression*의 각 2바이트 문자는 추가 변환없이 맵핑됩니다. 2바이트 문자의 첫 바이트는 *character-*

*expression*의 마지막 바이트로 나타나며 2바이트 대체 문자로 변환됩니다. *character-expression*의 문자 순서는 보존됩니다.

유니코드 데이터베이스의 경우 이 함수는 문자열을 인수의 코드 페이지에서 UCS-2로 변환합니다. 2바이트 문자를 포함하여 인수의 모든 문자가 변환됩니다. 두 번째 인수 값이 제공되면 결과로 발생하는 문자열의 필수 길이를 지정합니다(UCS-2 문자로).

VARGRAPHIC 함수에 의한 2바이트 코드 포인트로의 변환은 인수의 코드 페이지에 따라 달라집니다.

인수의 2바이트 문자는 변환되지 않습니다. 기타 모든 문자는 이와 일치하는 2바이트에 해당하는 문자로 변환됩니다. 이와 일치하는 2바이트에 해당하는 문자가 없으면 코드 페이지의 2바이트 대체 문자가 사용됩니다.

하나 이상의 2바이트 대체 문자가 결과로 리턴되면 경고 또는 오류 코드가 생성되지 않습니다.

그래픽을 vargraphic으로

graphic-expression
내장 그래픽 문자열 데이터 유형(GRAPHIC, VARGRAPHIC 또는 DBCLOB)인 값을 리턴하는 표현식입니다.

integer
결과로 발생하는 가변 길이 그래픽 문자열의 길이 속성입니다. 값은 0 - 16 336 범위여야 합니다. 두 번째 인수가 지정되지 않은 경우:

- *graphic-expression*이 비어 있는 문자열 상수인 경우, 결과의 길이 속성은 0입니다.
- 비어 있지 않으면, 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다.

결과의 실제 길이는 결과의 최소 길이 속성이며 *graphic-expression*의 실제 길이입니다. *graphic-expression*의 길이가 결과의 길이 속성보다 크면, 결과가 절단됩니다. 절단된 문자가 모두 공백이고 *graphic-expression*이 DBCLOB가 아닌 경우를 제외하고는 경고가 리턴됩니다(SQLSTATE 01004).

그래픽 표현식의 길이가 결과의 길이 속성보다 큰 경우 결과가 절단됩니다. 절단된 문자가 모두 공백이고 그래픽 표현식이 DBCLOB가 아닌 경우를 제외하고는 경고가 리턴됩니다(SQLSTATE 01004).

날짜 시간을 vargraphic으로

datetime-expression
다음 데이터 유형 중 하나인 표현식입니다.

DATE
결과는 두 번째 인수가 지정하는 형식으로 된 날짜의 그래픽 문자열 표현입니다. 결과 길이는 10입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않은 경우 오류가 발생합니다(SQLSTATE 42703).

TIME 결과는 두 번째 인수가 지정하는 형식으로 된 시간의 그래픽 문자열 표현입니다. 결과의 길이는 8입니다. 두 번째 인수가 지정되었으나 그 값이 올바르지 않으면 오류가 발생합니다(SQLSTATE 42703).

TIMESTAMP
결과는 시간소인의 그래픽 문자열 표현입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(0)이면 결과의 길이는 19입니다. *datetime-expression*의 데이터 유형이 TIMESTAMP(*n*)이고 *n*이 1과 12 사이인 경우, 결과의 길이는 20+*n*입니다. 그렇지 않으면 결과의 길이는 26입니다. 두 번째 인수는 지정하지 마십시오(SQLSTATE 42815).

결과의 코드 페이지는 해당 섹션의 DBCS 코드 페이지입니다.

주: 첫 번째 인수가 숫자이거나 첫 번째 인수가 문자열이고 길이 인수가 지정되어 있을 때 CAST 스펙은 응용프로그램의 이식성을 늘리는 데 사용되어야 합니다. 자세한 정보는 『CAST 스펙』을 참조하십시오.

예:

- EDLEVEL 컬럼은 SMALLINT로 정의되어 있습니다. 다음은 값을 가변 길이 그래픽 문자열로 리턴합니다.

```
SELECT VARGRAPHIC(EDLEVEL)
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  결과 값은 G'18입니다.

- SALARY 컬럼 및 COMN 컬럼은 스케일 2 및 정밀도 9인 DECIMAL로 정의되어 있습니다. 쉼표 10진수 문자를 사용하여 직원 Haas의 총 수입을 리턴합니다.

```
SELECT VARGRAPHIC(SALARY + COMM, ',')
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  결과 값은 G'56970,00 '입니다.

WEEK

스키마는 SYSFUN입니다.

WEEK 스칼라 함수는 인수에 있는 연도의 주를 1-54 범위 내의 정수 값으로 리턴합니다. 주는 일요일부터 시작합니다.

인수는 날짜, 시간소인, 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

WEEK_ISO

스키마는 SYSFUN입니다.

WEEK_ISO 함수는 인수에 있는 연도의 주를 1-53 범위 내의 정수 값으로 리턴합니다. 주는 월요일부터 시작하고 항상 7일을 포함합니다. 주 1은 목요일을 포함하는 연도의 첫 번째 주입니다. 이것은 1월 4일을 포함하는 첫 번째 주와 같습니다. 따라서 이전 연도의 마지막 주에 연도의 시작에 있는 최고 3일이 나타나도록 하는 것이 가능합니다. 역으로, 연도의 끝에 있는 최고 3일은 다음 연도의 첫 번째 주에 나타날 수 있습니다.

인수는 날짜, 시간소인, 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

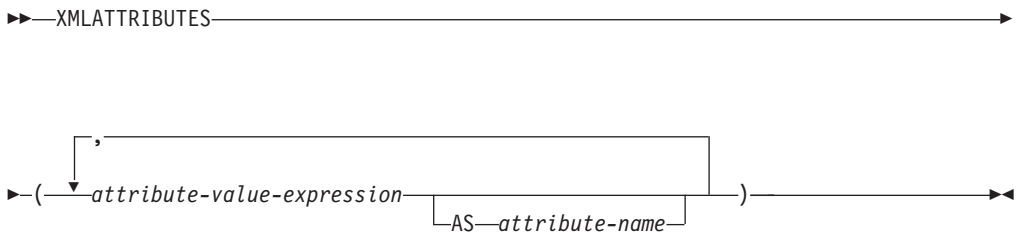
함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

예:

다음 목록은 WEEK_ISO 및 DAYOFWEEK_ISO 결과의 예를 나타냅니다.

```
DATE       WEEK_ISO    DAYOFWEEK_ISO
---------- ----------- -------------
1997-12-28          52             7
1997-12-31           1             3
1998-01-01           1             4
1999-01-01          53             5
1999-01-04           1             1
1999-12-31          52             5
2000-01-01          52             6
2000-01-03           1             1
```

XMLATTRIBUTES



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLATTRIBUTES 함수는 인수로부터 XML 속성을 작성합니다. 이 함수는 XMLELEMENT 함수의 인수로만 사용할 수 있습니다. 결과는 각각의 널(NULL)이 아닌 입력 값에 대한 XQuery 속성 노드를 포함하는 XML 시퀀스입니다.

attribute-value-expression

표현식을 결과 값으로 하는 표현식. *attribute-value-expression*의 데이터 유형은 구조화된 유형일 수 없습니다(SQLSTATE 42884). 표현식은 임의의 SQL 표현식일 수 있습니다. 표현식이 단순 컬럼 참조가 아닌 경우, 속성 이름을 지정해야 합니다.

attribute-name

속성 이름을 지정합니다. 이름은 XML 규정 이름 또는 QName 형식의 SQL ID입니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 속성 이름이 `xmlns`이거나 `xmlns:` 접두부가 있어서는 안됩니다. 이름 스페이스는 XMLNAMESPACES 함수를 사용하여 선언됩니다. 내재된 또는 명시적 중복된 속성 이름은 허용되지 않습니다(SQLSTATE 42713).

*attribute-name*을 지정하지 않을 경우, *attribute-value-expression*은 컬럼 이름이어야 합니다(SQLSTATE 42703). 속성 이름은 컬럼 이름 대 XML 속성 이름 간의 완전 이스케이프 맵핑을 사용하여 컬럼 이름으로부터 작성됩니다.

결과 데이터 유형은 XML입니다. *attribute-value-expression*의 결과가 널(NULL)일 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, 모든 *attribute-value-expression*의 결과가 널(NULL)이면 결과도 널(NULL) 값이 됩니다.

예를 들면, 다음과 같습니다.

주: XMLATTRIBUTES은 출력에 공백 또는 줄 바꾸기 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- 속성이 있는 요소를 생성합니다.

```
SELECT E.EMPNO, XMLELEMENT(
  NAME "Emp",
  XMLATTRIBUTES(
           E.EMPNO, E.FIRSTNME ||' '|| E.LASTNAME AS "name"
```

```
  )
)
AS "Result"
FROM EMPLOYEE E WHERE E.EDLEVEL = 12
```

이 쿼리는 다음 결과를 생성합니다.

```
EMPNO  Result
000290 <Emp EMPNO="000290" name="JOHN PARKER"></Emp>
000310 <Emp EMPNO="000310" name="MAUDE SETRIGHT"></Emp>
200310 <Emp EMPNO="200310" name="MICHELLE SPRINGER"></Emp>
```

- QName에서 사용되지 않는 이름 스페이스 선언을 가진 요소를 생성합니다. 접두부가 속성 값에서 사용됩니다.

```
VALUES XMLELEMENT(
  NAME "size",
  XMLNAMESPACES(
           'http://www.w3.org/2001/XMLSchema-instance' AS "xsi",
           'http://www.w3.org/2001/XMLSchema' AS "xsd"
  ),
  XMLATTRIBUTES(
           'xsd:string' AS "xsi:type"
       ), '1'
)
```

이 쿼리는 다음 결과를 생성합니다.

```
<size xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">1</size>
```

XMLCOMMENT

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLCOMMENT 함수는 입력 인수를 내용으로 하는 단일 XQuery 주석 노드와 함께 XML 값을 리턴합니다.

string-expression

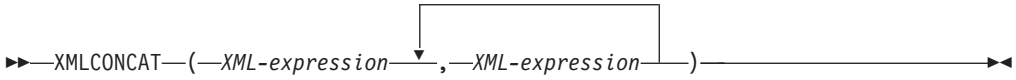
값이 문자열 유형을 갖는 표현식: CHAR, VARCHAR 또는 CLOB. XML 1.0 규칙에 정의된 대로 XML 주석에 대한 요구사항에 일치하는지 점검하기 위해 *string-expression*을 구문 분석합니다. *string-expression*의 결과는 다음 정규식을 따라야 합니다.

```
((Char - '-') | ('-' (Char - '-')))*
```

여기서 Char는 대리 블록 X'FFFE' 및 X'FFFF'를 제외한 유니코드 문자로 정의됩니다. 기본적으로 XML 주석은 두 개의 인접한 하이픈을 포함할 수 없으며, 하이픈으로 끝날 수 없습니다(SQLSTATE 2200S).

결과 데이터 유형은 XML입니다. *string-expression*의 결과가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 입력 값이 널(NULL)일 경우 그 결과도 널(NULL) 값입니다.

XMLCONCAT



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLCONCAT 함수는 XML 입력 인수의 변수 숫자를 병합한 시퀀스를 리턴합니다.

XML-expression
XML 데이터 유형의 표현식을 지정합니다.

결과 데이터 유형은 XML입니다. 결과는 널(NULL)이 아닌 입력 값의 조합을 포함하는 XML 시퀀스입니다. 입력의 널(NULL) 값은 무시됩니다. *XML-expression*의 결과가 널(NULL)이 될 수 있으면 결과는 널(NULL)일 수 있습니다. 즉, 모든 입력 값의 결과가 널(NULL)이면 결과는 널(NULL) 값입니다.

예:

주: XMLCONCAT은 출력에 공백 또는 줄 바꾸기 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- 성을 기준으로 정렬된 직원 목록을 포함하는 A00 및 A01 부서에 대한 부서 요소를 작성합니다. 부서 요소 바로 앞에 소개하는 주석을 포함시키십시오.

```
SELECT XMLCONCAT(
  XMLCOMMENT(
          'Confirm these employees are on track for their product schedule'
  ),
  XMLELEMENT(
    NAME "Department",
    XMLATTRIBUTES(
              E.WORKDEPT AS "name"
    ),
    XMLAGG(
      XMLELEMENT(
        NAME "emp", E.FIRSTNME
      )
    ORDER BY E.FIRSTNME
    )
  )
)
FROM EMPLOYEE E
WHERE E.WORKDEPT IN ('A00', 'B01')
GROUP BY E.WORKDEPT
```

이 쿼리는 다음 결과를 생성합니다.

```
<!--Confirm these employees are on track for their product schedule-->
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>DIAN</emp>
<emp>GREG</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
```

```
<!--Confirm these employees are on track for their product schedule-->
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

XMLDOCUMENT

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLDOCUMENT 함수는 하위 노드가 0개 이상인 단일 XQuery 문서 노드와 함께 XML 값을 리턴합니다.

XML-expression
XML 값을 리턴하는 표현식. XML 값의 시퀀스 항목은 속성 노드가 아니어야 합니다(SQLSTATE 10507).

결과 데이터 유형은 XML입니다. *XML-expression*의 결과가 널(NULL)일 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, 입력 값이 널(NULL)이면 결과도 널(NULL) 값이 됩니다.

결과로 생성되는 문서 노드의 하위 노드는 다음 단계에 설명된 대로 구성됩니다. 입력 표현식은 이들 단계에서 컨텐츠 시퀀스로 참조되는 노드 또는 요소 값의 시퀀스입니다.

1. 컨텐츠 시퀀스가 문서 노드를 포함할 경우, 문서 노드가 문서 노드의 하위에 의해 컨텐츠 시퀀스에서 대체됩니다.
2. 컨텐츠 시퀀스에 있는 하나 이상의 요소 값의 각각의 인접한 시퀀스가 각각의 요소 값을 인접한 값 사이에 단일 공백 문자가 삽입된 문자열로 캐스팅한 결과를 포함하는 텍스트 노드로 대체됩니다.
3. 컨텐츠 시퀀스의 각 노드에 대해, 노드의 새로운 딥(deep) 사본이 구성됩니다. 노드의 딥(deep) 사본은 노드 자신 및 하위 구성원을 포함하여 해당 노드에서 루트로 지정된 전체 서브트리의 사본입니다. 각각의 복사된 노드에는 새로운 노드 ID가 있습니다.
4. 컨텐츠 시퀀스의 노드가 새 하위 문서 노드의 하위 노드가 됩니다.

XMLDOCUMENT 함수는 XQuery 계산 문서 컨스트럭터를 효과적으로 실행합니다. 다음 함수의 결과는

```
XMLQUERY('document {$E}' PASSING BY REF XML-expression AS "E")
```

아래와 동일합니다.

```
XMLDOCUMENT( XML-expression )
```

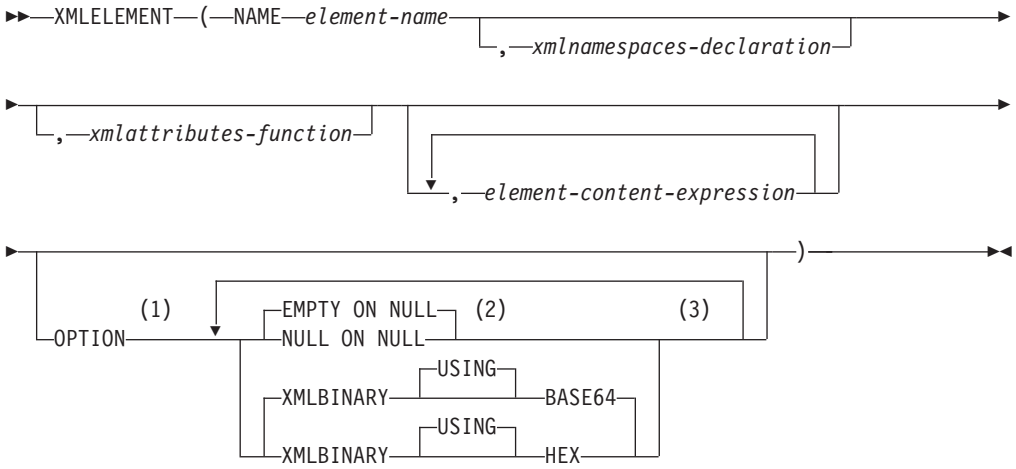
단, *XML-expression*이 널(NULL)일 경우 XMLQUERY는 널(NULL) 값을 리턴하는 XMLDOCUMENT와 달리 빈 시퀀스를 리턴합니다.

예:

- XML 컬럼에 구성된 문서를 삽입합니다.

```
INSERT INTO T1 VALUES(
       123, (
    SELECT XMLDOCUMENT(
      XMLELEMENT(
        NAME "Emp", E.FIRSTNME || ' ' || E.LASTNAME, XMLCOMMENT(
                       'This is just a simple example'
        )
      )
    )
    FROM EMPLOYEE E
    WHERE E.EMPNO = '000120'
  )
)
```

XMLELEMENT



주:

1 적어도 하나의 *xmlattributes-function* 또는 *element-content-expression*을 지정할 경우에만 OPTION절을 지정할 수 있습니다.

2 적어도 하나의 *element-content-expression*을 지정할 경우에만 NULL ON NULL 또는 EMPTY ON NULL을 지정할 수 있습니다.

3 같은 절은 두 번 이상 지정될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLELEMENT 함수는 XQuery 요소 노드인 XML 값을 리턴합니다.

NAME *element-name*

XML 요소의 이름을 지정합니다. 이름은 XML 규정 이름 또는 QName 형식의 SQL ID입니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 이름이 규정된 경우, 이름 스페이스 접두부를 범위 내에 선언해야 합니다(SQLSTATE 42635).

xmlnamespaces-declaration

XMLNAMESPACES declaration의 결과인 XML 이름 스페이스 선언을 지정합니다. 선언되는 이름 스페이스는 XMLELEMENT 함수의 범위 안에 있어야 합니다. 이름 스페이스는 다른 subselect 내부에 표시되는지 여부에 관계없이 XMLELEMEN 함수 내의 모든 중첩된 XML 함수에 적용됩니다.

*xmlnamespaces-declaration*을 지정하지 않을 경우, 이름 스페이스 선언이 구성된 요소와 연관되지 않습니다.

xmlattributes-function

요소에 대한 XML 속성을 지정합니다. 속성은 XMLATTRIBUTES 함수의 결과입니다.

element-content-expression

생성된 XML 요소 노드의 컨텐츠는 표현식 또는 표현식 목록에 의해 지정됩니다. *element-content-expression*의 데이터 유형은 구조화된 유형일 수 없습니다 (SQLSTATE 42884). 표현식은 임의의 SQL 표현식일 수 있습니다.

*element-content-expression*을 지정하지 않을 경우, 빈 문자열이 요소의 컨텐츠로 사용되며 OPTION NULL ON NULL 또는 EMPTY ON NULL을 지정해야 합니다.

OPTION

XML 요소를 구성하기 위한 추가 옵션을 지정합니다. OPTION절을 지정하지 않을 경우, 디폴트값은 EMPTY ON NULL XMLBINARY USING BASE64입니다. 이 절은 *element-content-expression*에 지정된 중첩된 XMLELEMENT 호출에 영향을 미치지 않습니다.

EMPTY ON NULL 또는 NULL ON NULL

각각의 *element-content-expression* 값이 널(NULL) 값일 경우 널(NULL) 값 또는 공백 요소를 리턴할 것인지 여부를 지정합니다. 이 옵션은 속성 값이 아닌 요소 컨텐츠에 대한 널(NULL) 처리에만 영향을 미칩니다. 디폴트값은 EMPTY ON NULL입니다.

EMPTY ON NULL

각각의 *element-content-expression* 값이 널(NULL)일 경우, 공백 요소가 리턴됩니다.

NULL ON NULL

각각의 *element-content-expression* 값이 널(NULL)일 경우, 널(NULL) 값이 리턴됩니다.

XMLBINARY USING BASE64 또는 XMLBINARY USING HEX

2진 입력 데이터, FOR BIT DATA 속성을 갖는 문자열 데이터 또는 해당 유형 중 하나를 기반으로 하는 구별 유형 중 가정한 인코딩을 지정합니다. 인코딩은 요소 컨텐츠 또는 속성 값에 적용됩니다. 디폴트값은 XMLBINARY USING BASE64입니다.

XMLBINARY USING BASE64

가정된 인코딩이 XML 스키마 유형 xs:base64Binary 인코딩에 정의된 대로 Base-64 문자임을 지정합니다. Base-64 인코딩은 65자의 US-ASCII 서브세트(10개의 숫자, 26개의 소문자, 26개의 대문자, '+' 및 '/')를 사용하여 서브세트에서 하나의 인쇄 가능한 문자로 각각의 6비트 2진수 또는 비트 데이터를 나타냅니다. 국제적인 표시를 위해 이들 문자를 선택합니다. 이 방법을 사용할 경우, 인코드된 데이터의 크기는 원래 2진수 또는 비트 데이터보다 33퍼센트 큽니다.

XMLBINARY USING HEX
가정된 인코딩이 XML 스키마 유형 xs:hex64Binary 인코딩에 정의된 대로 16진 문자임을 지정합니다. 16진수 인코딩은 두 개의 16진수 문자로 각각의 바이트(8비트)를 나타냅니다. 이 방법을 사용할 경우, 인코드된 데이터의 크기는 원래 2진수 또는 비트 데이터의 2배입니다.

이 함수는 요소 이름, 이름 스페이스 선언의 선택적 콜렉션, 속성의 선택적 콜렉션 및 XML 요소의 컨텐츠를 구성하는 0개 이상의 인수를 취합니다. 결과는 XML 요소 노드 또는 널(NULL) 값을 포함하는 XML 시퀀스입니다.

결과 데이터 유형은 XML입니다. *element-content-expression* 인수 중 하나가 널(NULL)이 될 수 있으면, 결과는 널(NULL)이 될 수 있습니다. 모든 *element-content-expression* 인수 값이 널(NULL)이고 NULL ON NULL 옵션이 유효할 경우, 결과도 널(NULL) 값이 됩니다.

주:

1. 디폴트 이름 공간을 정의하는 다른 요소의 내용으로서 복사할 요소를 구성할 경우, 새 상위 요소에서 디폴트 이름 공간을 상속할 때 발생할 수 있는 오류를 방지하려면 복사된 요소에서 디폴트 이름 공간을 명시적으로 선언 해제해야 합니다. 또한 사전 정의된 이름 스페이스 접두부('xs', 'xsi', 'xml' 및 'sqlxml')를 사용시 명시적으로 선언해야 합니다.
2. **요소 노드 구성**: 결과로 생성되는 요소 노드는 다음과 같이 구성됩니다.
   a. *xmlnamespaces-declaration*은 구성된 요소에 대한 일련의 범위 내 이름 스페이스를 추가합니다. 각각의 범위 내 이름 스페이스는 이름 스페이스 접두부(또는 디폴트 이름 스페이스)를 이름 스페이스 URI와 연관시킵니다. 범위 내 이름 스페이스는 요소의 범위 내에서 QNames를 해석하는 데 사용 가능한 일련의 이름 스페이스 접두부를 정의합니다.
   b. xmlattributes-function을 지정할 경우, 평가되며 결과는 속성 노드 시퀀스입니다.
   c. 각각의 *element-content-expression*이 평가되며 결과는 다음과 같은 노드 시퀀스로 변환됩니다.
      - 결과 유형이 XML이 아닐 경우, XML 텍스트 노드로 변환되며 컨텐츠는 SQL 데이터 값 대 XML 데이터 값 맵핑(『데이터 유형 간 캐스팅』의 비XML 값에서 XML 값으로의 지원되는 캐스트를 설명하는 테이블 참조) 규칙에 따라 XML에 맵핑된 *element-content-expression*입니다.
      - 결과 유형이 XML일 경우, 일반적으로 결과는 항목의 시퀀스입니다. 해당 시퀀스에서 일부 항목은 문서 노드일 수도 있습니다. 시퀀스의 각 문서 노드는 해당 최상위 레벨 하위의 시퀀스로 대체됩니다. 그런 다음 결과로 생성된 시퀀스의 각 노드에 대해 하위 및 속성을 포함한 노드의 새로운 딥(deep) 사본이 구성됩니다. 각각의 복사된 노드에는 새로운 노드 ID가 있습니다. 복사된

요소 및 속성 노드는 해당 유형 주석을 보존합니다. 시퀀스에서 리턴된 하나 이상의 요소 값의 각각의 인접한 시퀀스에 대해 인접한 값 사이에 단일 공백 문자가 삽입된 문자열로 각각의 요소 값을 캐스팅한 결과를 포함하는 새 텍스트 노드가 구성됩니다. 삽입된 공백 없이 해당 컨텐츠를 연결하여 컨텐츠 시퀀스의 인접한 텍스트 노드를 단일 텍스트 노드에 병합합니다. 연결 이후 해당 컨텐츠의 길이가 0인 문자열인 모든 텍스트 노드는 컨텐츠 시퀀스에서 삭제됩니다.

d. XML 속성의 결과 시퀀스 및 모든 *element-content-expression* 스펙의 결과로 생성된 시퀀스는 컨텐츠 시퀀스라고 하는 하나의 시퀀스로 연결됩니다. 컨텐츠 시퀀스의 인접한 텍스트 노드 시퀀스가 단일 텍스트 노드로 병합됩니다. 모든 *element-content-expression* 인수가 공백 문자열이거나 *element-content-expression* 인수를 지정하지 않을 경우, 공백 요소를 리턴합니다.

e. 컨텐츠 시퀀스는 속성 노드가 아닌 노드 다음에 속성 노드를 포함할 수 없습니다(SQLSTATE 10507). 컨텐츠 시퀀스에서 발생하는 속성 노드는 새 요소 노드의 속성이 됩니다. 해당 속성 노드의 두 개 이상이 동일한 이름을 가질 수 없습니다(SQLSTATE 10503). 이름 스페이스 URI가 구성된 요소의 범위 내 이름 스페이스에 없을 경우 속성 노드의 이름에 사용된 모든 이름 스페이스에 대응하는 이름 스페이스 선언이 작성됩니다.

f. 컨텐츠 시퀀스의 요소, 텍스트, 주석 및 처리 지시사항 노드가 구성된 요소 노드의 하위가 됩니다.

g. 구성된 요소 노드에는 `xs:anyType`의 유형 주석이 제공되며, 각각의 해당 속성에는 `xdt:untypedAtomic`의 유형 주석이 제공됩니다. 구성된 요소 노드의 노드 이름은 NAME 키워드 이후 지정되는 element-name입니다.

3. **XMLELEMENT 내에서 이름 스페이스 사용 규칙**: 이름 스페이스의 범위를 지정할 때 다음과 같은 규칙을 고려하십시오.

   - XMLNAMESPACES declaration에 선언된 이름 스페이스는 XMLELEMENT 함수에 의해 구성된 요소 노드의 범위 내 이름 스페이스입니다. 요소 노드가 직렬화된 경우, 각각의 해당 범위 내 이름 스페이스는 상위 요소 노드의 범위 내 이름 스페이스이고 상위 요소 또한 직렬화되지 않은 경우 각각의 해당 범위 내 이름 스페이스가 이름 스페이스 속성으로 직렬화됩니다.
   - XMLQUERY 또는 XMLEXISTS가 *element-content-expression*내에 있을 경우, 이름 스페이스가 XMLQUERY 또는 XMLEXISTS의 XQuery 표현식의 정적으로 알려진 이름 스페이스가 됩니다. 정적으로 알려진 이름 스페이스는 XQuery 표현식에서 QNames를 분석하는 데 사용됩니다. XQuery 프롤로그가 XQuery 표현식 범위 내에서 동일한 접두부를 사용하여 이름 스페이스를 선언할 경우, 프롤로그에 선언된 이름 스페이스가 XMLNAMESPACES declaration에 선언된 이름 스페이스를 대체합니다.

- 구성된 요소의 속성이 *element-content-expression*으로부터 제공된 경우, 해당 이름 스페이스가 구성된 요소의 범위 내 이름 스페이스로 아직 선언되지 않았을 수도 있습니다. 이 경우 새 이름 스페이스가 작성됩니다. 이로 인하여 충돌이 발생할 경우(속성 이름의 접두부가 한 범위 내 이름 스페이스에 의해 서로 다른 URI로 바인드될 경우), DB2는 그러한 충돌을 야기하지 않는 접두부를 생성하고 속성 이름에 사용된 접두부가 새 접두부로 변경되며 해당 새 접두부에 대해 이름 스페이스가 작성됩니다. 생성된 새 접두부의 패턴은 다음과 같습니다. ″db2ns-xx″, 여기서 ″x″는 [A-Z,a-z,0-9] 세트에서 선택된 문자입니다. 예를 들면, 다음과 같습니다.

```
VALUES XMLELEMENT(
  NAME "c", XMLQUERY(
           'declare namespace ipo="www.ipo.com"; $m/ipo:a/@ipo:b' PASSING XMLPARSE(
      DOCUMENT '<tst:a xmlns:tst="www.ipo.com" tst:b="2"/>'
             ) AS "m"
  )
)
```

다음을 리턴합니다.

```
<c xmlns:tst="www.ipo.com" tst:b="2"/>
```

두 번째 예:

```
VALUES XMLELEMENT(
  NAME "tst:c", XMLNAMESPACES(
           'www.tst.com' AS "tst"
  ),
  XMLQUERY(
           'declare namespace ipo="www.ipo.com"; $m/ipo:a/@ipo:b' PASSING XMLPARSE(
      DOCUMENT '<tst:a xmlns:tst="www.ipo.com" tst:b="2"/>'
             ) AS "m"
  )
)
```

다음을 리턴합니다.

```
<tst:c xmlns:tst="www.tst.com" xmlns:db2ns-a1="www.ipo.com" db2ns-a1:b="2"/>
```

예를 들면, 다음과 같습니다.

주: XMLELEMENT는 출력에 공백 또는 줄 바꾸기 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- NULL ON NULL 옵션을 사용하여 요소를 작성합니다.

```
SELECT E.FIRSTNME, E.LASTNAME, XMLELEMENT(
  NAME "Emp", XMLELEMENT(
    NAME "firstname", E.FIRSTNME
  ),
  XMLELEMENT(
    NAME "lastname", E.LASTNAME
  )
  OPTION NULL ON NULL
```

```
)
AS "Result"
FROM EMPLOYEE E
WHERE E.EDLEVEL = 12
```

이 쿼리는 다음 결과를 생성합니다.

```
FIRSTNME     LASTNAME        Emp
JOHN         PARKER          <Emp><firstname>JOHN</firstname>
                                                  <lastname>PARKER</lastname></Emp>
MAUDE        SETRIGHT        <Emp><firstname>MAUDE</firstname>
                                                  <lastname>SETRIGHT</lastname></Emp>
MICHELLE     SPRINGER        <Emp><firstname>MICHELLE</firstname>
                               <lastname>SPRINGER</lastname></Emp>
```

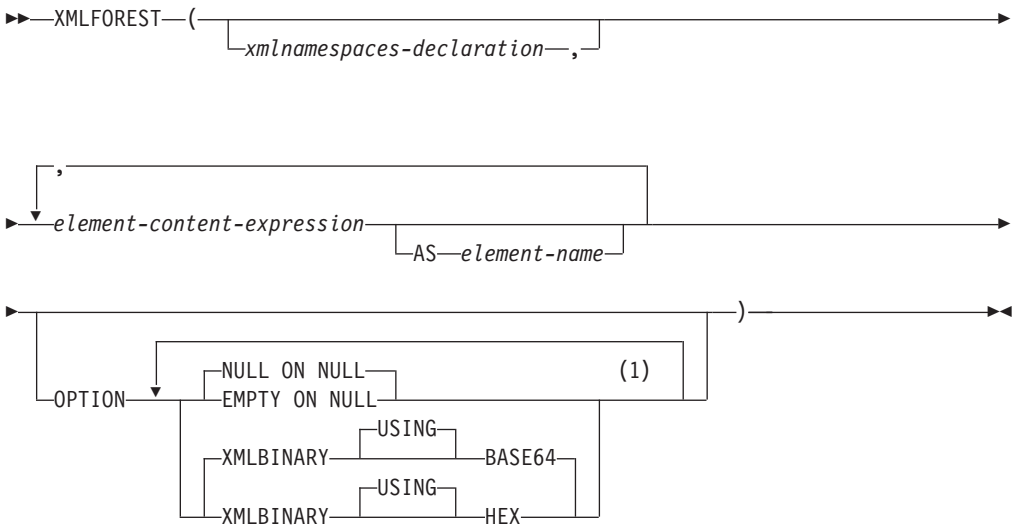
- 하위 요소로 중첩된 요소 목록을 갖는 요소를 생성합니다.

```
SELECT XMLELEMENT(
  NAME "Department", XMLATTRIBUTES(
             E.WORKDEPT AS "name"
  ),
  XMLAGG(
    XMLELEMENT(
      NAME "emp", E.FIRSTNME
    )
    ORDER BY E.FIRSTNME
  )
)
AS "dept_list"
FROM EMPLOYEE E
WHERE E.WORKDEPT IN ('A00', 'B01')
GROUP BY WORKDEPT
```

이 쿼리는 다음 결과를 생성합니다.

```
dept_list
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

XMLFOREST



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLFOREST 함수는 XQuery 요소 노드의 시퀀스인 XML 값을 리턴합니다.

xmlnamespaces-declaration

XMLNAMESPACES declaration의 결과인 XML 이름 스페이스 선언을 지정합니다. 선언되는 이름 스페이스는 XMLFOREST 함수의 범위 안에 있어야 합니다. 이름 스페이스는 다른 subselect 내부에 표시되는지 여부에 관계없이, XMLFOREST 함수 내의 모든 중첩된 XML 함수에 적용됩니다.

*xmlnamespaces-declaration*을 지정하지 않을 경우, 이름 스페이스 선언이 구성된 요소와 연관되지 않습니다.

element-content-expression

생성된 XML 요소 노드의 컨텐츠는 표현식에 의해 지정됩니다. *element-content-expression*의 데이터 유형은 구조화된 유형일 수 없습니다(SQLSTATE 42884). 표현식은 임의의 SQL 표현식일 수 있습니다. 표현식이 단순 컬럼 참조가 아닌 경우, 요소 이름을 지정해야 합니다.

AS *element-name*

XML 요소 이름을 SQL ID로 지정합니다. 요소 이름은 XML 규정 이름 또는 QName의 현식이어야 합니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 이름을 지정할 경우, 이름 스페이스 접두부를 범위 내에 선언해야 합니다(SQLSTATE 42635).

*element-name*이 지정되지 않은 경우, *element-content-expression*은 컬럼 이름이어야 합니다(SQLSTATE 42703).). 요소 이름은 컬럼 이름 대 QName 간의 완전 이스케이프 맵핑을 사용하여 컬럼 이름으로부터 작성됩니다.

OPTION

XML 요소를 구성하기 위한 추가 옵션을 지정합니다. OPTION절을 지정하지 않을 경우, 디폴트값은 NULL ON NULL XMLBINARY USING BASE64입니다. 이 절은 *element-content-expression*에 지정된 중첩된 XMLELEMENT 호출에 영향을 미치지 않습니다.

EMPTY ON NULL 또는 NULL ON NULL

각각의 *element-content-expression* 값이 널(NULL) 값일 경우 널(NULL) 값 또는 공백 요소를 리턴할 것인지 여부를 지정합니다. 이 옵션은 속성 값이 아닌 요소 컨텐츠에 대한 널(NULL) 처리에만 영향을 미칩니다. 디폴트값은 NULL ON NULL입니다.

EMPTY ON NULL

각각의 *element-content-expression* 값이 널(NULL)일 경우, 공백 요소가 리턴됩니다.

NULL ON NULL

각각의 *element-content-expression* 값이 널(NULL)일 경우, 널(NULL) 값이 리턴됩니다.

XMLBINARY USING BASE64 또는 XMLBINARY USING HEX

2진 입력 데이터, FOR BIT DATA 속성을 갖는 문자열 데이터 또는 해당 유형 중 하나를 기반으로 하는 구별 유형 중 가정한 인코딩을 지정합니다. 인코딩은 요소 컨텐츠 또는 속성 값에 적용됩니다. 디폴트값은 XMLBINARY USING BASE64입니다.

XMLBINARY USING BASE64

가정된 인코딩이 XML 스키마 유형 xs:base64Binary 인코딩에 정의된 대로 Base-64 문자임을 지정합니다. Base-64 인코딩은 65자의 US-ASCII 서브세트(10개의 숫자, 26개의 소문자, 26개의 대문자, '+' 및 '/')를 사용하여 서브세트에서 하나의 인쇄 가능한 문자로 각각의 6비트 2진수 또는 비트 데이터를 나타냅니다. 국제적인 표시를 위해 이들 문자를 선택합니다. 이 방법을 사용할 경우, 인코드된 데이터의 크기는 원래 2진수 또는 비트 데이터보다 33퍼센트 큽니다.

XMLBINARY USING HEX

가정된 인코딩이 XML 스키마 유형 xs:hex64Binary 인코딩에 정의된 대로 16진 문자임을 지정합니다. 16진수 인코딩은 두 개의 16진수 문자로 각각의 바이트(8비트)를 나타냅니다. 이 방법을 사용할 경우, 인코드된 데이터의 크기는 원래 2진수 또는 비트 데이터의 2배입니다.

이 함수는 선택적인 일련의 이름 스페이스 선언 및 하나 이상의 요소 노드에 대한 이름 및 요소 내용을 구성하는 하나 이상의 인수를 취합니다. 결과는 XQuery 요소 노드 또는 널(NULL) 값 시퀀스를 포함하는 XML 시퀀스입니다.

결과 데이터 유형은 XML입니다. *element-content-expression* 인수 중 하나가 널(NULL)이 될 수 있으면, 결과는 널(NULL)이 될 수 있습니다. 모든 *element-content-expression* 인수 값이 널(NULL)이고 NULL ON NULL 옵션이 유효할 경우, 결과도 널(NULL) 값이 됩니다.

XMLFOREST 함수는 XMLCONCAT 및 XMLELEMENT를 사용하여 표현할 수 있습니다. 예를 들어, 다음 두 표현식은 시맨틱적으로 같습니다.

```
XMLFOREST(xmlnamespaces-declaration, arg1 AS name1, arg2 AS name2 ...)
```

```
XMLCONCAT(
    XMLELEMENT(
        NAME name1, xmlnamespaces-declaration, arg1
  ),
    XMLELEMENT(
        NAME name2, xmlnamespaces-declaration, arg2
  )
  ...
)
```

주:

1. 디폴트 이름 공간을 정의하는 다른 요소의 내용으로서 복사할 요소를 구성할 경우, 새 상위 요소에서 디폴트 이름 공간을 상속할 때 발생할 수 있는 오류를 방지하려면 복사된 요소에서 디폴트 이름 공간을 명시적으로 선언 해제해야 합니다. 또한 사전 정의된 이름 스페이스 접두부('xs', 'xsi', 'xml' 및 'sqlxml')를 사용시 명시적으로 선언해야 합니다.

예:

주: XMLFOREST는 출력에 공백 또는 줄 바꾸기 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- 디폴트 이름 스페이스를 사용하여 요소의 포리스트(forest)를 작성합니다.

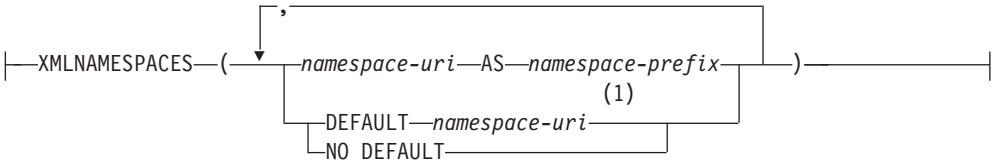
```
SELECT EMPNO,
  XMLFOREST(
    XMLNAMESPACES(
      DEFAULT 'http://hr.org', 'http://fed.gov' AS "d"
    ),
          LASTNAME, JOB AS "d:job"
  )
AS "Result"
FROM EMPLOYEE
WHERE EDLEVEL = 12
```

이 쿼리는 다음 결과를 생성합니다.

```
EMPNO  Result
000290 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">PARKER
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>

000310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SETRIGHT
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>

200310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SPRINGER
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
```

XMLNAMESPACES

xmlnamespaces-declaration:



주:

1 XMLNAMESPACES의 인수에서 DEFAULT 또는 NO DEFAULT는 한 번만 지정할 수 있습니다.

스키마는 SYSIBM입니다. 선언 이름은 규정된 이름으로 지정할 수 없습니다.

XMLNAMESPACES 선언은 인수로부터 이름 스페이스 선언을 작성합니다. 이 선언은 XMLELEMENT, XMLFOREST 및 XMLTABLE과 같은 세부 함수에 대한 인수로만 사용할 수 있습니다. 결과는 각각의 널(NULL)이 아닌 입력 값에 대한 범위 내 이름 스페이스를 포함하는 하나 이상의 XML 이름 스페이스 선언입니다.

namespace-uri
이름 스페이스 URI(Universal Resource Identifier)를 SQL 문자열 상수로 지정합니다. *namespace-prefix*와 함께 사용될 경우 이 문자열 상수가 공백이 아니어야 합니다(SQLSTATE 42815).

namespace-prefix
이름 스페이스 접두부를 지정합니다. 접두부는 XML NCName 형식의 SQL ID입니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 접두부는 `xml` 또는 `xmlns`일 수 없으며, 이름 스페이스 선언 목록 내에서 고유해야 합니다(SQLSTATE 42635).

DEFAULT *namespace-uri*
해당 이름 스페이스 선언 범위 내에서 사용할 디폴트 이름 스페이스를 지정합니다. 다른 *namespace-uri*는 다른 DEFAULT 선언 또는 NO DEFAULT 선언에 의해 중첩된 범위에서 대체되지 않을 경우 *namespace-uri*는 범위의 규정되지 않은 이름에 적용됩니다.

NO DEFAULT
해당 이름 스페이스 선언 범위 내에서 사용할 디폴트 스페이스가 없음을 지정합니다. DEFAULT 선언에 의해 중첩된 범위에서 대체되지 않을 경우 범위에는 디폴트 이름 스페이스가 없습니다.

결과의 데이터 유형은 XML입니다. 결과는 각각의 지정된 이름 스페이스에 대한 XML 이름 스페이스 선언입니다. 결과는 널(NULL)이 될 수 없습니다.

예를 들면, 다음과 같습니다.

주: XMLNAMESPACES는 출력에 공백 또는 줄 바꾸기 문자를 삽입하지 않습니다. 모든 출력 예는 읽기 쉽게 형식화되어 있습니다.

- 접두부가 adm인 이름 스페이스와 연관된 XML 요소 adm:employee와 XML 속성 adm:department를 생성합니다.

```
SELECT EMPNO, XMLELEMENT(
  NAME "adm:employee", XMLNAMESPACES(
           'http://www.adm.com' AS "adm"
  ),
  XMLATTRIBUTES(
           WORKDEPT AS "adm:department"
  ),
   LASTNAME
)
FROM EMPLOYEE
WHERE JOB = 'ANALYST'
```

  이 쿼리는 다음 결과를 생성합니다.

```
000130 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
    QUINTANA</adm:employee>
000140 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
    NICHOLLS</adm:employee>
200140 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  NATZ</adm:employee>
```

- 디폴트 이름 스페이스와 연관된 XML 요소 'employee' 및 디폴트 이름 스페이스를 사용하지 않지만 해당 하위 요소 'department'가 디폴트 이름 스페이스를 사용하는 하위 요소 'job'.

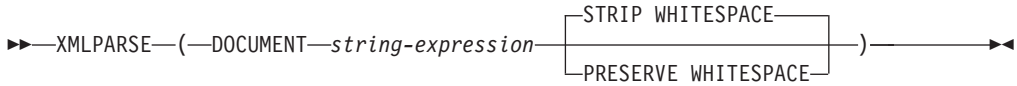
```
SELECT EMP.EMPNO, XMLELEMENT(
  NAME "employee", XMLNAMESPACES(
    DEFAULT 'http://hr.org'
  ),
       EMP.LASTNAME, XMLELEMENT(
    NAME "job", XMLNAMESPACES(
      NO DEFAULT
    ),
          EMP.JOB, XMLELEMENT(
      NAME "department", XMLNAMESPACES(
        DEFAULT 'http://adm.org'
      ),
               EMP.WORKDEPT
    )
  )
)
FROM EMPLOYEE EMP
WHERE EMP.EDLEVEL = 12
```

  이 쿼리는 다음 결과를 생성합니다.

```
000290 <employee xmlns="http://hr.org">PARKER<job xmlns="">OPERATOR
    <department xmlns="http://adm.org">E11</department></job></employee>
000310 <employee xmlns="http://hr.org">SETRIGHT<job xmlns="">OPERATOR
```

```
    <department xmlns="http://adm.org">E11</department></job></employee>
200310 <employee xmlns="http://hr.org">SPRINGER<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
```

XMLPARSE



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLPARSE 함수는 인수를 XML 문서로 구문 분석한 후 XML 값을 리턴합니다.

DOCUMENT
XML 이름 스페이스 권장사항에 의해 수정된 대로, 구문 분석할 문자열 표현식이 XML 1.0을 따르는 well-formed XML 문서로 평가되어야 함을 지정합니다 (SQLSTATE 2200M).

string-expression
문자열 또는 BLOB 값을 리턴하는 표현식을 지정합니다. 매개변수 표시문자를 사용할 경우, 지원되는 데이터 유형 중 하나로 명시적으로 캐스트해야 합니다.

STRIP WHITESPACE 또는 PRESERVE WHITESPACE
입력 인수의 공백을 보존해야 하는지 여부를 지정합니다. 아무 것도 지정하지 않을 경우, 디폴트값은 STRIP WHITESPACE입니다.

STRIP WHITESPACE
인접한 포함 요소 속성이 xml:space='preserve'가 아닐 경우 길이가 최대 1000 바이트인 공백 문자만을 포함하는 텍스트 노드를 스트립하도록 지정합니다. 텍스트 노드가 1000바이트를 넘는 공백으로 시작할 경우, 오류가 리턴됩니다 (SQLSTATE 54059).

CDATA 섹션이 공백 문자 또한 이 옵션의 영향을 받습니다. DTD에 요소에 대한 DOCTYPE 선언이 있을 수도 있지만, 공백이 스트립되는지 여부를 판별할 때 요소의 내용 모델을 사용하지 않습니다.

PRESERVE WHITESPACE
가장 인접한 포함 요소의 속성이 xml:space='default'일 경우에도 모든 공백을 보존하도록 지정합니다.

결과 데이터 유형은 XML입니다. *string-expression*의 결과가 널(NULL)일 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, *string-expression*의 결과가 널(NULL)이면 결과도 널(NULL) 값이 됩니다.

주:

1. **입력 문자열 인코딩**: 입력 문자열은 XML 문서의 문자 인코딩을 식별하는 XML 선언을 포함할 수도 있습니다. 문자열이 XMLPARSE 함수에 문자 문자열로 전달될 경우, 데이터베이스 서버에서 코드 페이지로 변환됩니다. 이 코드 페이지는 원래 코드 페이지 및 XML 선언에서 식별된 인코딩과 다를 수도 있습니다.

따라서 응용프로그램은 문자열 입력이 있는 XMLPARSE를 직접 사용하지 않아야 하며 호스트 변수를 사용하여 XML 문서를 포함하는 문서를 직접 전송하여 외부 코드 페이지 및 XML 선언의 인코딩이 일치하도록 유지해야 합니다. 이러한 상황에서 XMLPARSE를 사용해야 할 경우, 코드 페이지 변환을 방지하려면 BLOB 유형을 인수로 지정합니다.

2. **DTD 처리**: 외부 DTD(Document Type Definition) 및 엔티티는 데이터베이스에 등록되어 있어야 합니다. 내부 및 외부 DTD 모두의 구문이 유효한지 점검합니다. 구문 분석 프로세스 중에 다음과 같은 조치도 수행됩니다.
   - 내부 및 외부 DTD에 의해 정의된 디폴트값이 적용됩니다.
   - 엔티티 참조 및 매개변수 엔티티를 확장된 형식으로 바꿉니다.
   - 내부 DTD 및 외부 DTD가 동일한 요소를 정의할 경우 오류가 리턴됩니다 (SQLSTATE 2200M).
   - 내부 DTD 및 외부 DTD가 동일한 엔티티 또는 속성을 정의할 경우, 내부 정의가 선택됩니다.

   구문 분석 후, 외부 DTD 및 엔티티에 대한 참조뿐 아니라 내부 DTD 및 엔티티 값은 저장된 표현식으로 보존되지 않습니다.
3. **비UTF-8 데이터베이스에서의 문자 변환**: 코드 페이지 변환은 문서가 문자 데이터 유형의 매개변수 표시문자 또는 호스트 변수나 문자열 리터럴에서 전달되는 경우 XML 문서가 유니코드가 아닌 데이터베이스 서버로 구문 분석될 때 발생합니다. 유형 XML, BLOB 또는 FOR BIT DATA(CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA)의 매개변수 표시문자나 호스트 변수를 사용하여 XML 문서를 구문 분석하면 코드 페이지 변환이 이루어지지 않습니다. 문자 데이터 유형이 사용되는 경우 XML 문서의 모든 문자가 대상 데이터베이스 코드 페이지에 일치하는 코드 포인트를 가지고 있는지 주의하여 확인해야 합니다. 가지고 있지 않으면 대체 문자가 도입됩니다. 구성 매개변수 **enable_xmlchar**을 사용하면 유니코드가 아닌 데이터베이스에 저장된 XML 데이터의 무결성을 확인하는 데 도움이 될 수 있습니다. 이 매개변수를 ″NO″로 설정하면 문자 데이터 유형으로부터의 XML 문서 삽입이 차단됩니다. BLOB 및 FOR BIT DATA 데이터 유형은 계속 허용됩니다. 이 데이터 유형을 사용하여 데이터베이스에 전달된 문서는 코드 페이지 변환을 피하기 때문입니다.

예 :

PRESERVE WHITESPACE 옵션을 사용하면 설명 요소의 공백 문자를 포함하여 테이블에 삽입되는 XML 문서의 공백 문서를 보존합니다.

```
INSERT INTO PRODUCT VALUES ('100-103-99','Tool bag',14.95,NULL,NULL,NULL,
XMLPARSE( DOCUMENT
  '<produce xmlns="http://posample.org" pid="100-103-99">
    <description>
     <name>Tool bag</name>
     <details>
```

```
      Super Deluxe tool bag:
      - 26 inches long, 12 inches wide
      - Curved padded handle
      - Locking latch
      - Reinforced exterior pockets
     </details>
     <price>14.95</price>
     <weight>3 kg</weight>
    </description>
  </product>' PRESERVE WHITESPACE ));
```

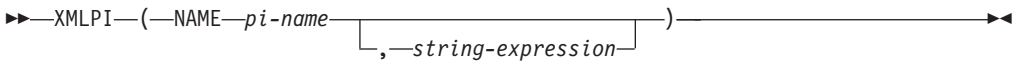
선택 명령문

```
SELECT XMLQUERY ('$d/*:product/*:description/*:details' passing DESCRIPTION as "d" )
FROM PRODUCT WHERE PID = '100-103-99' ;
```

을 실행하면 공백 문자를 갖는 세부사항 요소를 리턴합니다.

```
<details xmlns="http://posample.org">
  Super Deluxe tool bag:
  - 26 inches long, 12 inches wide
  - Curved padded handle
  - Locking latch
  - Reinforced exterior pockets
</details>
```

XMLPI



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLPI 함수는 단일 XQuery 처리 명령어 노드와 함께 XML 값을 리턴합니다.

NAME *pi-name*
처리 명령어의 이름을 지정합니다. 이름은 XML NCName 형식의 SQL ID입니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 이름은 대소문자 조합에 관계없이 'xml'이라는 단어가 될 수 없습니다(SQLSTATE 42634).

string-expression
문자열 값을 리턴하는 표현식. 결과로 생성되는 문자열은 UTF-8로 변환되며 XML 1.0 규칙에 지정된 대로 다음과 같은 XML 처리 명령어의 컨텐츠를 따라야 합니다(SQLSTATE 2200T).

- 문자열이 '?>' 하위 문자열을 포함할 수 없으며, 이는 해당 하위 문자열이 처리 명령어를 종료하기 때문입니다.
- 각 문자열의 문자는 대리 블록 X'FFFE' 및 X'FFFF'를 제외한 임의의 유니코드 문자일 수 있습니다.

결과 문자열이 작성된 처리 명령어 노드의 컨텐츠가 됩니다.

결과 데이터 유형은 XML입니다. *string-expression*의 결과가 널(NULL)일 경우, 결과는 널(NULL)이 될 수 있습니다. 즉, *string-expression*의 결과가 널(NULL)이면 결과도 널(NULL) 값이 됩니다. *string-expression*이 공백 문자열이거나 지정되지 않은 경우, 공백 처리 명령어 노드가 리턴됩니다.

예를 들면, 다음과 같습니다.

- XML 처리 명령어 노드를 생성합니다.

```
SELECT XMLPI(
  NAME "Instruction", 'Push the red button'
)
FROM SYSIBM.SYSDUMMY1
```

  이 쿼리는 다음 결과를 생성합니다.

```
<?Instruction Push the red button?>
```

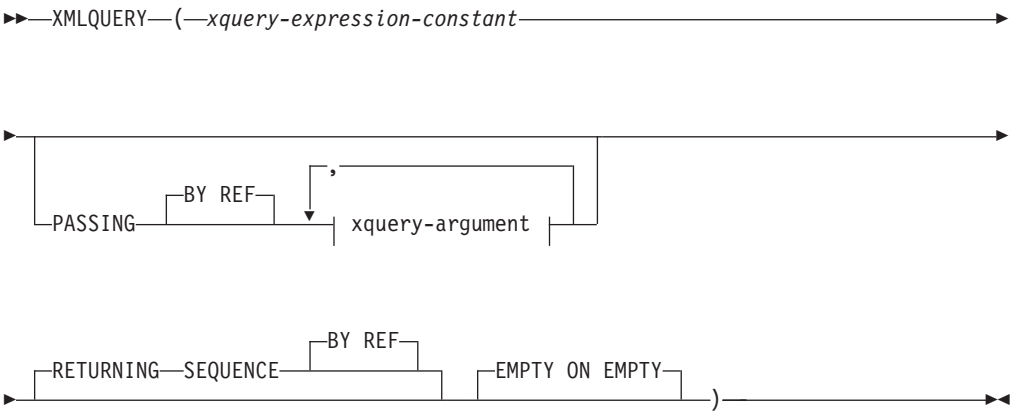
- 공백 XML 처리 명령어 노드를 생성합니다.

```
SELECT XMLPI(
  NAME "Warning"
)
FROM SYSIBM.SYSDUMMY1
```

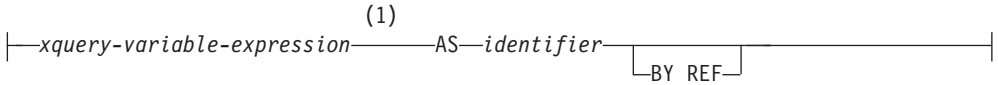
이 쿼리는 다음 결과를 생성합니다.

```
<?Warning ?>
```

XMLQUERY



xquery-argument:



주:

1 표현식의 데이터 유형은 DECFLOAT가 될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLQUERY 함수는 지정된 입력 인수를 XQuery 변수로 사용하여 XQuery 표현식 평가로부터 XML 값을 리턴합니다.

xquery-expression-constant

지원되는 XQuery 언어 구문을 사용하여 XQuery 표현식으로 해석되는 SQL 문자열 상수를 지정합니다. 상수 문자열은 XQuery문으로 구문 분석되기 전에 UTF-8로 변환됩니다. XQuery 표현식은 선택적인 일련의 입력 XML 값을 사용하여 실행되며, XMLQUERY 표현식의 값으로 또한 리턴되는 출력 시퀀스를 리턴합니다. *xquery-expression-constant*의 값은 빈 문자열 또는 공백 문자의 문자열이 아니어야 합니다(SQLSTATE 10505).

PASSING

입력 값 및 이들 입력 값이 *xquery-expression-constant*에서 지정된 XQuery 표현식으로 패스되는 방법을 지정합니다. 디폴트로, 함수가 호출되는 범위 내에 있는 모든 고유 컬럼 이름은 변수 이름으로 컬럼의 이름을 사용하여 XQuery 표현식으로 내재적으로 전달됩니다. 지정된 xquery-argument의 *identifier*가 범위 내 컬럼 이름과 일치하는 경우 명시적 xquery-argument가 XQuery 표현식에 전달되어 해당되는 내재된 컬럼을 대체합니다.

BY REF

디폴트 전달 메커니즘이 데이터 유형 XML의 *xquery-variable-expression* 및 리턴되는 값에 대한 참조에 의한 메커니즘임을 지정합니다. 참조에 의해 XML 값이 패스되면 XQuery 평가는 지정된 입력 표현식으로부터 직접 입력 노드 트리(있는 경우)를 사용하여 원본 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보유합니다. 두 개의 인수가 동일한 XML 값을 패스할 경우, 두 입력 인수 사이에 들어 있는 일부 노드와 관련된 노드 ID 비교 및 문서 순서화 비교는 동일한 XML 노드 트리에 있는 노드를 나타낼 수 있습니다.

이 절은 비XML 값이 패스되는 방법에 아무 영향을 주지 않습니다. 비XML 값은 XML로 캐스트되는 동안 값의 사본을 새로 작성합니다.

xquery-argument

*xquery-expression-constant*에서 지정된 XQuery 표현식으로 패스되는 인수를 지정합니다. 인수는 값과 값을 전달할 방법을 지정합니다. 인수에는 평가되는 SQL 표현식이 포함됩니다.

- 결과 값이 XML 유형인 경우 이는 *input-xml-value*로 됩니다. 널(NULL) XML 값은 빈 XML 시퀀스로 변환됩니다.
- 결과 값이 XML 유형이 아닌 경우 이는 XML 데이터 유형으로 캐스트될 수 있어야 합니다. 널(NULL) 값은 빈 XML 시퀀스로 변환됩니다. 변환된 값은 *input-xml-value*로 됩니다.

*xquery-expression-constant*를 평가할 때 XQuery 변수는 *input-xml-value*와 같은 값 및 AS절에서 지정된 이름으로 표시됩니다.

xquery-variable-expression

실행 중 *xquery-expression-constant*에서 지정된 XQuery 표현식에 사용할 수 있는 값을 갖는 SQL 표현식을 지정합니다. 이 표현식은 시퀀스 참조(SQLSTATE 428F9) 또는 OLAP 함수(SQLSTATE 42903)를 포함할 수 없습니다.표현식의 데이터 유형은 DECFLOAT가 될 수 없습니다.

AS *identifier*

*xquery-variable-expression*에서 생성된 값이 XQuery 변수인 *xquery-expression-constant*로 패스되는 것을 지정합니다. 변수 이름은 *identifier*입니다. *identifier*에는 XQuery 언어에서 변수 이름 앞에 오는 달러 부호($)가 포함되지 않습니다. identifier는 유효한 XQuery 변수 이름이어야 하며 XML NCName으로 제한됩니다(SQLSTATE 42634). identifier의 길이는 128바이트보다 커야 합니다. 동일한 PASSING절에서 두 개의 인수가 동일한 ID를 사용할 수 없습니다(SQLSTATE 42711).

BY REF

XML 입력 값이 참조에 의해 패스되는 것을 표시합니다. 참조에 의해 XML 값이 패스되면 XQuery 평가는 지정된 입력 표현식으로부터 직접 입력 노

드 트리(있는 경우)를 사용하여 원본 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보유합니다. 두 개의 인수가 동일한 XML 값을 패스할 경우, 두 입력 인수 사이에 들어 있는 일부 노드가 관련된 노드 ID 비교 및 문서 순서화 비교는 동일한 XML 노드 트리에 있는 노드를 참조할 수 있습니다. *xquery-variable-expression* 다음에 BY REF를 지정하지 않으면 XML 인수는 PASSING 키워드 다음에 오는 구문을 통해 제공된 디폴트 전달 메커니즘 방식으로 패스됩니다. 비XML 값에는 이 옵션을 지정할 수 없습니다. 비XML 값이 패스될 때 이 값은 XML로 변환되며 이 프로세스는 사본을 작성합니다.

RETURNING SEQUENCE

XMLQUERY 표현식이 시퀀스를 리턴함을 표시합니다.

BY REF

XQuery 표현식의 결과가 참조에 의해 리턴됨을 표시합니다. 이 값이 노드를 포함할 경우, XQuery 표현식의 리턴값을 사용하는 모든 표현식은 노드 참조를 직접 수신하며 원래 노드 ID 및 문서 순서를 제외한 모든 노드 등록 정보를 보존합니다. 참조된 노드는 해당 노드 트리 내에서 연결된 상태를 유지합니다. BY REF절을 지정하지 않고 PASSING을 지정할 경우, 디폴트 전달 메커니즘이 사용됩니다. BY REF를 지정하지 않고 PASSING을 지정하지 않을 경우, 디폴트 리턴 메커니즘은 BY REF입니다.

EMPTY ON EMPTY

XQuery 표현식 처리로부터 공백 시퀀스 결과가 공백 시퀀스로 리턴됨을 지정합니다.

결과의 데이터 유형은 XML이며 널(NULL)이 될 수 없습니다.

XQuery 표현식의 평가 결과 오류가 발생할 경우, XMLQUERY 함수는 XQuery 오류를 리턴합니다(SQLSTATE 클래스 '10').

주:

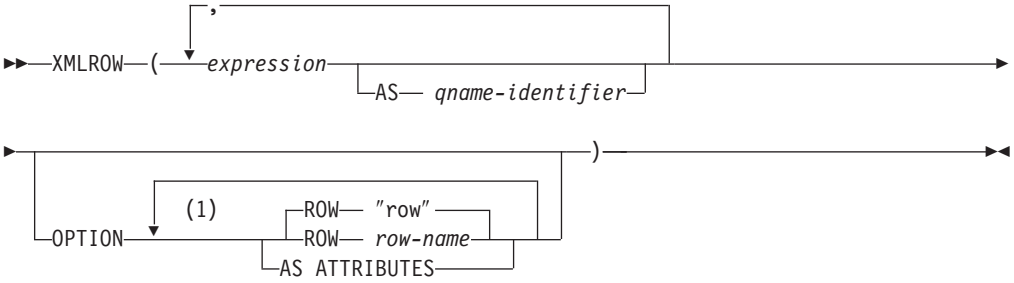
1. **XMLQUERY 사용 제한사항**: XMLQUERY 함수가 다음과 같을 수 없습니다.
   - JOIN 연산자 또는 MERGE문과 연관된 ON절에 포함된 경우(SQLSTATE 42972)
   - CREATE INDEX EXTENSION문의 GENERATE KEY USING 또는 RANGE THROUGH절에 포함된 경우(SQLSTATE 428E3)
   - CREATE FUNCTION(외부 스칼라)문의 FILTER USING절 또는 CREATE INDEX EXTENSION문의 FILTER USING절에 포함된 경우(SQLSTATE 428E4)
   - 점검 제한조건 또는 컬럼 생성 표현식에 포함된 경우(SQLSTATE 42621)
   - group-by-clause에 포함된 경우(SQLSTATE 42822)

- column-function에 대한 인수에 포함된 경우(SQLSTATE 42607)

2. **서브쿼리로 XMLQUERY**: 서브쿼리를 제한하는 명령문으로 서브쿼리로 작동하는 XMLQUERY 표현식을 제한할 수 있습니다.

XMLROW

XMLROW 함수는 하나의 최상위 레벨 요소 노드를 포함하는 단일 XQuery 문서 노드와 함께 XML 값을 리턴합니다.



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

expression

생성된 XML 요소 노드의 컨텐츠는 표현식에 의해 지정됩니다. 표현식의 데이터 유형은 구조화된 유형이 될 수 없습니다(SQLSTATE 42884). 표현식은 임의의 SQL 표현식일 수 있습니다. 표현식이 단순 컬럼 참조가 아닌 경우, 요소 이름을 지정해야 합니다.

AS *qname-identifier*

XML 요소 이름 또는 속성 이름을 SQL ID로 지정합니다. *qname-identifier*는 XML 규정 이름 또는 QName 양식이어야 합니다(SQLSTATE 42634). 유효한 이름에 대한 자세한 정보는 W3C XML 이름 스페이스 권장 스펙을 참조하십시오. 이름을 지정할 경우, 이름 스페이스 접두부를 범위 내에 선언해야 합니다(SQLSTATE 42635). *qname-identifier*를 지정하지 않은 경우 *expression*은 컬럼 이름이어야 합니다(SQLSTATE 42703). 요소 이름 또는 속성 이름은 컬럼 이름에서 QName으로의 완전 이스케이프 맵핑을 사용하여 컬럼 이름으로부터 작성됩니다.

OPTION

XML 값을 구성하기 위한 추가 옵션을 지정합니다. OPTION절을 지정하지 않을 경우, 디폴트 동작이 적용됩니다.

AS ATTRIBUTES

컬럼 이름 또는 *qname-identifier*가 속성 이름으로 제공되는 속성 값에 각 표현식이 맵핑됨을 지정합니다.

ROW *row-name*

각 행이 맵핑될 요소의 이름을 지정합니다. 이 옵션을 지정하지 않는 경우 디폴트 요소 이름은 ″row″입니다.

주

디폴트로, 결과 세트의 각 행은 다음과 같이 XML 값에 맵핑됩니다.

- 각 행은 이름이 ″row″인 XML 요소로 변환되고 각 컬럼은 컬럼 이름이 요소 이름인 중첩 요소로 변환됩니다.
- 널(NULL) 조절 동작은 NULL ON NULL입니다. 컬럼의 NULL 값은 하위 요소 없음으로 맵핑됩니다. 모든 컬럼 값이 NULL이면 함수에 의해 NULL 값이 리턴됩니다.
- BLOB 및 FOR BIT DATA 데이터 유형에 대한 2진 인코딩 스킴은 base64Binary 인코딩입니다.
- 문서 노드는 내재적으로 행 요소에 추가되어 XML 결과를 제대로 형성된 단일 루트 XML 문서로 작성합니다.

예:

관계형 형식으로 저장된 숫자 데이터를 포함하는 컬럼 C1 및 C2가 있는 다음 테이블 T1을 가정하십시오.

```
C1          C2
----------- -----------
          1           2
          -           2
          1           -
          -           -

  4 record(s) selected.
```

- 다음 예는 테이블을 나타내기 위해 행 요소 시퀀스를 사용하고 디폴트로 동작하는 XMLRow 쿼리 및 출력 조각을 나타냅니다. :

```
SELECT XMLROW(C1, C2) FROM T1

<row><C1>1</C1><C2>2</C2></row>
<row><C2>2</C2></row>
<row><C1>1</C1></row>

  4 record(s) selected.
```

- 다음 예는 속성 중심의 맵핑이 있는 XMLRow 쿼리 및 출력 조각을 나타냅니다. 이전 예에서와 같이 중첩 요소로 표시되는 대신, 관계형 데이터가 요소 속성에 맵핑됩니다.

```
SELECT XMLROW(C1, C2 OPTION AS ATTRIBUTES) FROM T1
```

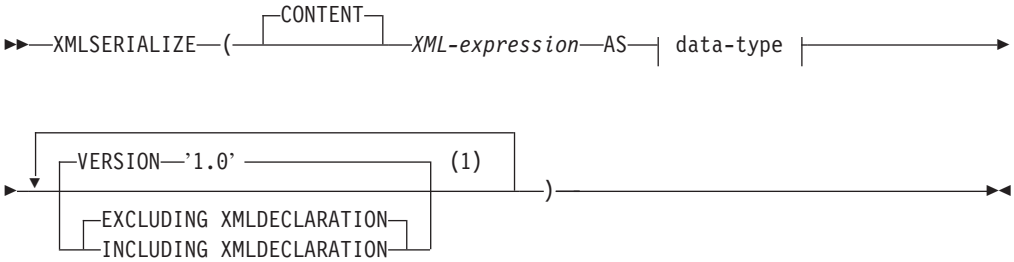
```
<row C1="1" C2="2"/>
<row C2="2"/>
<row C1="1"/>

  4 record(s) selected.
```

- 다음 예는 디폴트 <row> 요소가 <entry>로 교체된 XMLRow 쿼리 및 출력 조각을 나타냅니다. 컬럼 C1 및 C2는 <column1> 및 <column2> 요소로 리턴되고 C1 및 C2의 총계가 <total> 요소 내에서 리턴됩니다.

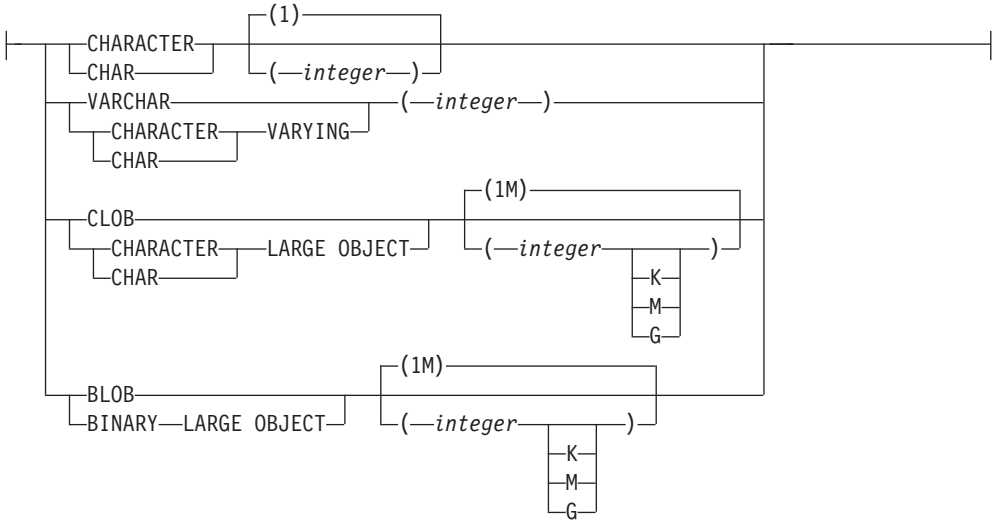
```
SELECT XMLROW(
   C1 AS "column1", C2 AS "column2",
   C1+C2 AS "total" OPTION ROW "entry")
FROM T1
```

```
<entry><column1>1</column1><column2>2</column2><total>3</total></entry>
<entry><column2>2</column2></entry>
<entry><column1>1</column1></entry>

  4 record(s) selected.
```

XMLSERIALIZE



data-type:



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLSERIALIZE 함수는 *XML-expression* 인수로부터 생성되는 지정된 데이터 유형의 직렬화된 XML 값을 리턴합니다.

CONTENT

XML 값을 지정할 수 있으며 직렬화의 결과가 해당 입력 값을 기반으로 함을 지정합니다.

XML-expression

데이터 유형 XML의 값을 리턴하는 표현식을 지정합니다. XML 시퀀스 값은 속성 노드인 항목을 포함하지 않아야 합니다(SQLSTATE 2200W). 이는 직렬화 프로세스의 입력입니다.

AS *data-type*
결과 유형을 지정합니다. 지정된 결과 데이터 유형의 내재된 또는 명시적 길이 속성이 직렬화된 출력을 포함할 만큼 충분해야 합니다(SQLSTATE 22001).

VERSION '1.0'
직렬화된 값의 XML 버전을 지정합니다. 지정되는 유일한 버전은 '1.0'이며 문자열 상수로 지정해야 합니다(SQLSTATE 42815).

EXCLUDING XMLDECLARATION 또는 INCLUDING XMLDECLARATION
XML 선언이 결과에 포함되는지 여부를 지정합니다. 디폴트값은 EXCLUDING XMLDECLARATION입니다.

EXCLUDING XMLDECLARATION
XML 선언이 결과에 포함되지 않음을 지정합니다.

INCLUDING XMLDECLARATION
XML 선언이 결과에 포함됨을 지정합니다. XML 선언은 '<?xml version="1.0" encoding="UTF-8"?>' 문자열입니다.

결과에는 사용자가 지정한 데이터 유형이 포함됩니다. 결과로 생성된 XML 노드를 직렬화하기 전에 XMLDOCUMENT를 *XML-expression*에 적용하여 단일 문서 노드를 갖도록 XML 시퀀스를 효과적으로 변환합니다. *XML-expression*의 결과가 널(NULL)이 될 수 있습니다. *XML-expression*의 결과가 널(NULL)일 경우, 결과도 널(NULL) 값이 됩니다.

주:

1. **직렬화된 결과에서 인코딩**: 직렬화된 결과는 UTF-8을 사용하여 인코드됩니다. XMLSERIALIZE가 문자 데이터 유형에서 사용되고 INCLUDING XMLDECLARATION절을 지정할 경우, 직렬화된 XML을 포함하는 결과로 생성된 문자열이 문자열의 코드 페이지와 일치하지 않는 XML 인코딩 선언을 가질 수도 있습니다. UTF-8 인코딩을 사용하는 직렬화 이후, 서버에서 클라이언트로 리턴되는 문자열이 클라이언트의 코드 페이지로 변환되며 해당 코드 페이지가 UTF-8과 다를 수도 있습니다.

   따라서 응용프로그램은 문자열 유형을 리턴하는 XMLSERIALIZE INCLUDING XMLDECLARATION을 직접 사용하지 않아야 하며 호스트 변수로 직접 XML 값을 검색하여 외부 코드 페이지 및 XML 선언의 인코딩이 일치하도록 유지해야 합니다. 이러한 상황에서 XMLSERIALIZE를 사용해야 할 경우, 코드 페이지 변환을 방지하려면 BLOB 유형을 지정합니다.
2. **구문 대안**: XMLCLOB(*XML-expression*)를 XMLSERIALIZE(*XML-expression* AS CLOB(2G)) 대신 지정할 수 있습니다. 이전 DB2 릴리스와의 호환성을 위해서만 지원됩니다.

XMLTEXT

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLTEXT 함수는 입력 인수를 내용으로 갖는 단일 XQuery 텍스트 노드가 있는 XML 값을 리턴합니다.

string-expression
값이 문자열 유형 CHAR, VARCHAR 또는 CLOB인 표현식.

결과 데이터 유형은 XML입니다. *string-expression*의 결과가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 입력 값이 널(NULL)일 경우 그 결과도 널(NULL) 값입니다. *string-expression*의 결과가 빈 문자열이면 그 결과 값은 빈 텍스트 노드입니다.

예를 들면, 다음과 같습니다.

- 단순 XMLTEXT 쿼리를 작성합니다.

```
VALUES (
  XMLTEXT(
           'The stock symbol for Johnson&Johnson is JNJ.'
  )
)
```

  이 쿼리는 다음과 같이 연속된 결과를 생성합니다.

```
1
--------------------------------------------------
The stock symbol for Johnson&amp;Johnson is JNJ.
```

  텍스트 노드가 연속될 때 '&' 부호가 '&'로 맵핑된 것을 참고하십시오.

- 혼합 내용을 구성하려면 XMLAGG와 함께 XMLTEXT를 사용하십시오. 테이블 T의 내용을 다음과 같이 간주하십시오.

```
seqno  plaintext                                                  emphtext
------ ---------------------------------------------------------- -------------
1      This query shows how to construct                          mixed content
2       using XMLAGG and XMLTEXT. Without                         XMLTEXT
3      XMLAGG will not have text nodes to group with other nodes, mixed content
              therefore, cannot generate
```

```
SELECT XMLELEMENT(
  NAME "para", XMLAGG(
    XMLCONCAT(
      XMLTEXT(
                    PLAINTEXT
      ),
      XMLELEMENT(
        NAME "emphasis", EMPHTEXT
      )
    )
```

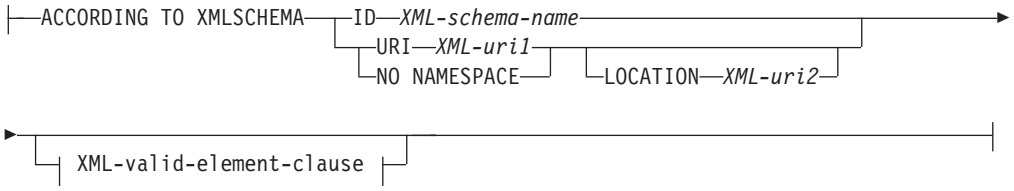
```
      ORDER BY SEQNO
    ), '.'
     ) AS "result"
  FROM T
```

이 쿼리는 다음 결과를 생성합니다.

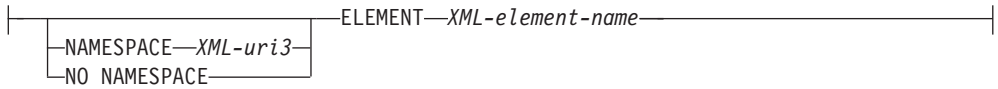
```
result
--------------------------------------------------------------------------------
<para>This query shows how to construct <emphasis>mixed content</emphasis>
using XMLAGG and XMLTEXT. Without <emphasis>XMLTEXT</emphasis> , XMLAGG
will not have text nodes to group with other nodes, therefore, cannot generate
<emphasis>mixed content</emphasis>.</para>
```

XMLVALIDATE

XML-validate-according-to-clause:



XML-valid-element-clause:



스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLVALIDATE 함수는 디폴트값을 포함하여 XML 스키마 유효성 확인에서 확보한 정보에 의해 증가된 입력 XML 값의 사본을 리턴합니다.

DOCUMENT

*XML-expression*에서 생성된 XML 값이 XML 버전 1.0에 일치하는 올바른 형식의 XML 문서이어야 함을 지정합니다(SQLSTATE 2200M).

XML-expression

XML 데이터 유형의 값을 리턴하는 표현식입니다. *XML-expression*이 XML 호스트 변수이거나 내재적 또는 명시적으로 유형이 지정된 매개변수 표시문자인 경우, 이 함수는 무시할 수 있는 공백을 스트립하는 유효성 확인 구문 분석을 수행하므로 CURRENT IMPLICIT XMLPARSE OPTION 설정을 고려하지 않습니다.

XML-validate-according-to-clause

입력 XML 값을 유효성 확인할 때 사용될 정보를 지정합니다.

ACCORDING TO XMLSCHEMA

유효성 확인에 대한 XML 스키마 정보가 명시적으로 지정되었는지 표시합니다. 이 절을 포함하지 않은 경우, *XML-expression* 값의 내용에 XML 스키마 정보가 제공되어야 합니다.

ID *XML-schema-name*

유효성 확인에 사용될 XML 스키마에 대한 SQL ID를 지정합니다. 내재적 또는 명시적 SQL 스키마 규정자를 포함하는 이름은 현재 서버에서 XML 스키마 저장소에 있는 기존 XML 스키마를 고유하게 식별합니다. 내

재적 또는 명시적으로 지정된 SQL 스키마에 이 이름에 의한 XML 스키마가 존재하지 않으면 오류가 리턴됩니다(SQLSTATE 42704).

URI *XML-uri1*

유효성 확인에 사용될 XML 스키마의 목표 이름 공간 URI를 지정합니다. *XML-uri1*의 값은 URI를 공백이 아닌 문자열 상수로 지정합니다. URI는 등록된 XML 스키마의 목표 이름 공간이어야 하며(SQLSTATE 4274A), LOCATION절이 지정되어 있지 않은 경우 등록된 XML 스키마를 고유하게 식별해야 합니다(SQLSTATE 4274B).

NO NAMESPACE

유효성 확인에 대한 XML 스키마에 목표 이름 공간이 없음을 지정합니다. 목표 이름 공간 URI는 명시 목표 이름 공간 URI로 지정될 수 없는 공백 문자열과 동일합니다.

LOCATION *XML-uri2*

유효성 확인에 사용될 XML 스키마의 XML 스키마 위치 URI를 지정합니다. *XML-uri2*의 값은 URI를 공백이 아닌 문자열 상수로 지정합니다. 목표 이름 공간 URI와 조인된 XML 스키마 위치 URI는 등록된 XML 스키마를 식별해야 하며(SQLSTATE 4274A) 이러한 XML 스키마는 한 개만 등록되어야 합니다(SQLSTATE 4274B).

XML-valid-element-clause

*XML-expression*의 XML 값에는 XML 문서의 루트 요소로서 지정된 요소 이름이 있어야 함을 지정합니다.

NAMESPACE *XML-uri3* **또는 NO NAMESPACE**

유효성 확인될 요소에 대한 목표 이름 공간을 지정합니다. 절이 지정되지 않은 경우, 지정된 요소는 유효성 확인에 사용될 등록된 XML 스키마의 목표 이름 공간과 동일한 이름 공간에 있는 것으로 간주됩니다.

NAMESPACE *XML-uri3*

유효성 확인될 요소에 대한 이름 공간 URI를 지정합니다. *XML-uri3*의 값은 URI를 공백이 아닌 문자열 상수로 지정합니다. 이는 유효성 확인에 사용될 등록된 XML 스키마에 두 개 이상의 이름 공간이 있을 때 사용될 수 있습니다.

NO NAMESPACE

유효성 확인에 대한 요소에 목표 이름 공간이 없음을 지정합니다. 목표 이름 공간 URI는 명시 목표 이름 공간 URI로 지정될 수 없는 공백 문자열과 동일합니다.

ELEMENT *xml-element-name*

유효성 확인에 사용될 XML 스키마의 글로벌 요소 이름을 지정합니다. 내

재적 또는 명시적 이름 공간이 있는 지정된 요소는 *XML-expression*의 값의 루트 요소와 일치해야 합니다(SQLSTATE 22535 또는 22536).

결과 데이터 유형은 XML입니다. *XML-expression*의 값이 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, *XML-expression*의 값이 널(NULL)이면 그 결과도 널(NULL) 값입니다.

XML 유효성 검증 프로세스는 연속된 XML 값에서 수행됩니다. XMLVALIDATE는 XML 유형의 인수를 사용하여 호출되므로, 이 값은 다음 두 가지 예외로 유효성 확인을 처리하기 전에 자동으로 연속화됩니다.

- XMLVALIDATE에 대한 인수가 XML 호스트 변수이거나 내재적 또는 명시적으로 유형이 지정된 매개변수 표시문자인 경우, 이 입력 값에 대해 유효성 확인 구문 분석 조작이 수행됩니다(내재적 비유효성 확인 구문 분석이 수행되지 않으므로 CURRENT IMPLICIT XMLPARSE OPTION 설정을 고려하지 않습니다).
- XMLVALIDATE에 대한 인수가 PRESERVE WHITESPACE 옵션을 사용하는 XMLPARSE 호출인 경우, 이 문서의 XML 구문 분석 및 XML 유효성 확인은 단일 유효성 확인 구문 분석 조작으로 조인될 수 있습니다.

XML 값이 이전에 유효성 확인된 경우, 순번 매김 프로세스는 이전 유효성 확인에서 주석이 표시된 유형 정보를 제거합니다. 그러나 이전 유효성 확인에서 모든 디폴트값 및 엔티티 확장은 변경되지 않고 그대로 남아 있습니다. 유효성 확인이 완료되면 무시할 수 있는 모든 공백 문자가 결과에서 제거됩니다.

루트 요소에 이름 공간이 들어 있지 않은 문서를 유효성 확인하려면 루트 요소에 xsi:noNamespaceSchemaLocation 속성이 있어야 합니다.

주:

1. **XML 스키마 판별**: XML 스키마는 XMLVALIDATE 호출에 포함되도록 명시적으로 지정되거나 입력 XML 값의 XML 스키마 정보로부터 판별될 수 있습니다. 호출 중 XML 스키마 정보가 지정되지 않은 경우 유효성 확인에 대해 등록된 스키마를 식별하려면 목표 이름 공간 및 입력 XML 값의 스키마 위치를 사용합니다. 명시적 XML 스키마가 지정되지 않은 경우 입력 XML 값에 XML 스키마 정보 힌트가 포함되어야 합니다(SQLSTATE 2200M). 명시적 또는 내재적 XML 스키마 정보는 등록된 XML 스키마를 식별해야 하며(SQLSTATE 42704, 4274A 또는 22532), 이러한 XML 스키마는 한 개만 등록되어야 합니다(SQLSTATE 4274B 또는 22533).
2. **XML 스키마 권한 부여**: 유효성 확인에 사용되는 XML 스키마는 사용되기 전에 먼저 XML 스키마 저장소에 등록되어야 합니다. 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.
   - 유효성 확인 중 사용될 XML 스키마에 대한 USAGE 특권
   - DBADM 권한

예를 들면, 다음과 같습니다.

- XML 인스턴스 문서에서 XML 스키마 힌트에 의해 식별되는 XML 스키마를 사용하여 유효성 확인

```
INSERT INTO T1(XMLCOL)
  VALUES (XMLVALIDATE(?))
```

입력 매개변수 표시문자가 XML 스키마 정보를 포함하는 XML 값으로 바운드되는 것으로 가정합니다.

```
<po:order
                  xmlns:po='http://my.world.com'
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://my.world.com/world.xsd" >
...
</po:order>
```

또한 목표 이름 공간 ″http://my.world.com″ 및 schemaLocation 힌트 ″http://my.world.com/world.xsd″와 연관된 XML 스키마가 XML 스키마 저장소에 있다고 가정하십시오.

이러한 가정에 기초하여 해당 XML 스키마에 따라 입력 XML 값이 유효성 확인됩니다.

- SQL 이름 PODOCS.WORLDPO에 의해 식별되는 XML 스키마를 사용하여 유효성 확인

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA ID PODOCS.WORLDPO
    )
  )
```

SQL 이름 FOO.WORLDPO와 연관된 XML 스키마가 XML 저장소에 있다고 가정할 경우 해당 XML 스키마에 따라 입력 XML 값이 유효성 확인됩니다.

- XML 값의 지정된 요소를 유효성 확인합니다.

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA ID FOO.WORLDPO
      NAMESPACE 'http://my.world.com/Mary'
      ELEMENT "po"
    )
  )
```

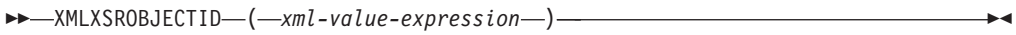
SQL 이름 FOO.WORLDPO와 연관된 XML 스키마가 XML 저장소에 있다고 가정할 경우 XML 스키마는 이름 공간이 'http://my.world.com/Mary'인 요소 ″po″에 대해 유효성 확인됩니다.

- XML 스키마는 목표 이름 공간 및 스키마 위치에 의해 식별됩니다.

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA URI 'http://my.world.com'
      LOCATION 'http://my.world.com/world.xsd'
    )
  )
```

또한 목표 이름 공간 ″http://my.world.com″ 및 schemaLocation 힌트 ″http://my.world.com/world.xsd″와 연관된 XML 스키마가 XML 스키마 저장소에 있다고 가정할 경우 해당 스키마에 따라 입력 XML 값이 유효성 확인됩니다.

XMLXSROBJECTID



스키마는 SYSIBM입니다.

XMLXSROBJECTID 함수는 인수에 지정된 XML 문서를 유효성 확인하는 데 사용되는 XML 스키마의 XSR 오브젝트 ID를 리턴합니다. XSR 오브젝트 ID는 BIGINT 값으로 리턴되며 SYSCAT.XSROBJECTS에 있는 단일 행에 키를 제공합니다.

xml-value-expression

결과가 XML 데이터 유형의 값이 되는 표현식을 지정합니다. 결과되는 XML 값은 널(NULL) 값 또는 XML 문서인 단일 항목이 있는 XML 시퀀스여야 합니다 (SQLSTATE 42815). 인수가 널(NULL)인 경우 함수에서 널(NULL)이 리턴됩니다. *xml-value-expression*이 유효성이 확인된 XML 문서를 지정할 경우 함수에서 0이 리턴됩니다.

주:

1. XML 스키마를 사용하여 유효성이 확인된 XML 값에 영향을 미치지 않고 XML을 삭제할 수 있으므로 함수에서 리턴되는 XSR 오브젝트 ID에 해당하는 XML 스키마는 더 이상 존재하지 않습니다. 따라서 카탈로그 뷰에서 추가 XML 스키마 정보를 페치하기 위해 XSR 오브젝트 ID를 사용하는 쿼리는 공백 결과 세트를 리턴할 수 있습니다.
2. 응용프로그램은 XSR 오브젝트 ID를 사용하여 XML 스키마에 대한 추가 정보를 검색할 수 있습니다. 예를 들어, XSR 오브젝트 ID는 SYSCAT.XSROBJECTHIERARCHIES로부터 XML 스키마의 XML 스키마 문서 계층 구조 및 SYSCAT.SYSXSROBJECTCOMPONENTS로부터 등록된 XML 스키마를 작성하는 개별 XML 스키마 문서를 리턴하는 데 사용될 수 있습니다.

예를 들면, 다음과 같습니다.

- 다음 예는 MYTABLE 테이블에 저장된 XML 문서 XMLDOC의 XML 스키마 ID를 검색합니다.

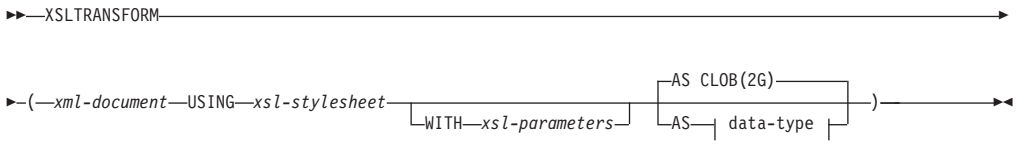
```
SELECT XMLXSROBJECTID(XMLDOC) FROM MYTABLE
```

- 다음 예는 XML 스키마를 구성하는 XML 스키마 문서의 계층 구조를 포함하여 MYTABLE 테이블에서 특정 ID를 갖는 XML 문서(이 경우 DOCKEY = 1인 문서)와 연관된 XML 스키마 문서를 검색합니다.

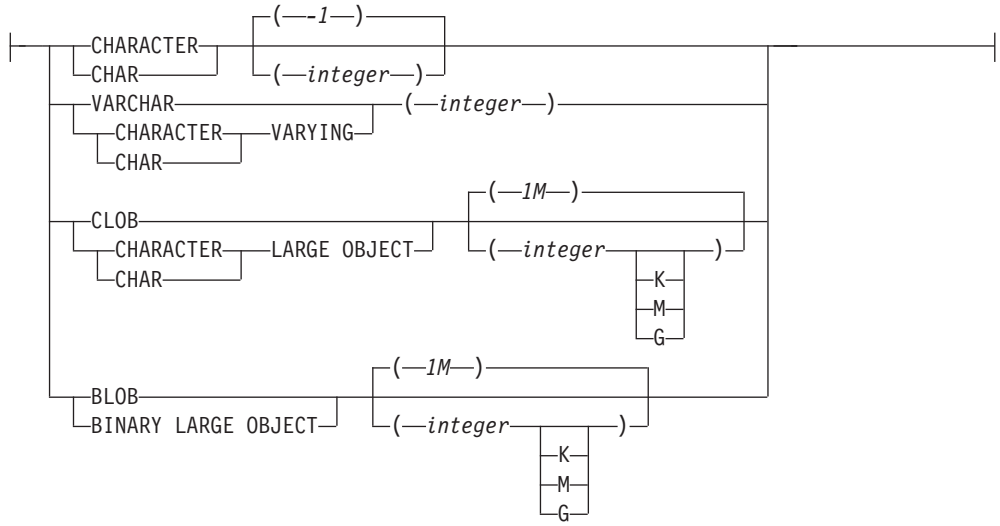
```
SELECT H.HTYPE, C.TARGETNAMESPACE, C.COMPONENT
FROM SYSCAT.XSROBJECTCOMPONENTS C, SYSCAT.XSROBJECTHIERARCHIES H
WHERE C.OBJECTID =
       (SELECT XMLXSROBJECTID(XMLDOC) FROM MYTABLE
   WHERE DOCKEY = 1)
AND C.OBJECTID = H.XSROBJECTID
```

XSLTRANSFORM

하나의 XML 스키마를 준수하는 XML 문서를 다른 스키마를 준수하는 문서로 변환하는 것을 비롯하여, XML 데이터를 다른 형식으로 변환하려면 XSLTRANSFORM을 사용하십시오.



data-type:



스키마는 SYSIBM입니다. 이 함수는 규정된 이름으로 지정할 수 없습니다.

XSLTRANSFORM 함수는 XML 문서를 다른 데이터 형식으로 변환합니다. 데이터는 XML, HTML 또는 일반 텍스트를 포함하여(단, 이에 한하지 않음) XSLT 프로세서에 가능한 양식으로 변환될 수 있습니다.

XSLTRANSFORM에서 사용되는 모든 경로는 데이터베이스 시스템에 내부적입니다. 이 명령은 현재 외부 파일 시스템에 있는 스타일 시트 또는 파일에 대해 직접 사용할 수 없습니다.

xml-document

데이터 유형이 XML, CHAR, VARCHAR, CLOB 또는 BLOB인 제대로 형성된 XML 문서를 리턴하는 표현식입니다. 이는 *xsl-stylesheet*에 지정된 XSL 스타일 시트를 사용하여 변환되는 문서입니다.

주:

XML 문서는 최소한 단일 루트 및 제대로 형성된 문서여야 합니다.

xsl-stylesheet

데이터 유형이 XML, CHAR, VARCHAR, CLOB 또는 BLOB인 제대로 형성된 XML 문서를 리턴하는 표현식입니다. 문서는 W3C XSLT 버전 1.0 권장사항을 준수하는 XSL 스타일 시트입니다. XQUERY문이나 `xsl:include` 선언을 통합하는 스타일 시트는 지원되지 않습니다. 이 스타일 시트는 *xml-document*에 지정된 값을 변환하는 데 적용됩니다.

xsl-parameters

데이터 유형이 XML, CHAR, VARCHAR, CLOB 또는 BLOB인 제대로 형성된 XML 문서나 널(NULL)을 리턴하는 표현식입니다. 이는 *xsl-stylesheet*에 지정된 XSL 스타일 시트에 매개변수 값을 제공하는 문서입니다. 매개변수의 값은 속성이나 텍스트 노드로 지정할 수 있습니다.

매개변수 문서의 구문은 다음과 같습니다.

```
<params xmlns="http://www.ibm.com/XSLTransformParameters">
<param name="..." value="..."/>
<param name="...">enter value here</param>
  ...
</params>
```

주:

스타일시트 문서에는 매개변수 문서에 지정된 것과 일치하는 이름 속성 값을 가지고 있는 `xsl:param` 요소가 있어야 합니다.

AS *data-type*

결과 데이터 유형을 지정합니다. 지정된 결과 데이터 유형의 내재된 또는 명시적 길이 속성이 변환된 출력을 포함할 만큼 충분해야 합니다(SQLSTATE 22001). 디폴트 결과 데이터 유형은 CLOB(2G)입니다.

주:

xml-document 인수나 *xsl-stylesheet* 인수가 널(NULL)인 경우 결과는 널(NULL)이 됩니다.

위의 문서 중 하나를 CHAR, VARCHAR 또는 CLOB 컬럼에 저장할 때 코드 페이지 변환이 발생할 수 있습니다. 이로 인해 문자가 손실될 수 있습니다.

예 :

이 예는 형식화 엔진으로 XSLT를 사용하는 방법을 나타냅니다. 설정하려면 먼저 아래에 있는 두 개의 예 문서를 데이터베이스에 삽입하십시오.

```
INSERT INTO XML_TAB VALUES
(1,
         '<?xml version="1.0"?>
<students xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "/home/steffen/xsd/xslt.xsd">
<student studentID="1" firstName="Steffen" lastName="Siegmund"
```

```
    age=â€23â€ university=â€Rostockâ€/>
</students>',
    '<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="headline"/>
<xsl:param name="showUniversity"/>
<xsl:template match="students">
                 <html>
                    <head/>
                       <body>
                              <h1><xsl:value-of select="$headline"/></h1>
                              <table border="1">
                                      <th>
                                          <tr>
                                <td width="80">StudentID</td>
                                <td width="200">First Name</td>
                                <td width="200">Last Name</td>
                                <td width="50">Age</td>
                                <xsl:choose>
      <xsl:when test="$showUniversity ='true'">
                                                  <td width="200">University</td>
                                     </xsl:when>
      </xsl:choose>
                             </tr>
                                <th>
                                <xsl:apply-templates/>
                               </table>
                        </body>
                </html>
               </xsl:template>
                    <xsl:template match="student">
                    <tr>
                  <td><xsl:value-of select="@studentID"/></td>
                  <td><xsl:value-of select="@firstName"/></td>
                  <td><xsl:value-of select="@lastName"/></td>
                  <td><xsl:value-of select="@age"/></td>
                  <xsl:choose>
                      <xsl:when test="$showUniversity = 'true' ">
                        <td><xsl:value-of select="@university"/></td>
                      </xsl:when>
                  </xsl:choose>
                         </tr>
                    </xsl:template>
</xsl:stylesheet>'
);
```

다음으로, XSLTRANSFORM 함수를 호출하여 XML 데이터를 HTML로 변환하여 표시하십시오.

```
SELECT XSLTRANSFORM (XML_DOC USING XSL_DOC AS CLOB(1M)) FROM XML_TAB;
```

결과는 다음 문서입니다.

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1></h1>
<table border="1">
<th>
<tr>
<td width="80">StudentID</td>
<td width="200">First Name</td>
```

```
<td width="200">Last Name</td>
<td width="50">Age</td>
</tr>
</th>
 <tr>
<td>1</td>
<td>Steffen</td><td>Siegmund</td>
<td>23</td>
</tr>
              </table>
</body>
</html>
```

이 예에서, 출력은 HTML이고 매개변수는 생성되는 HTML과 이 HTML로 가져오는 데이터에 영향을 미칩니다. 이 예는 일반 사용자 출력에 대한 형식화 엔진으로서의 XSLT 사용을 나타냅니다.

사용 메모:

이 함수는 높은 성능의 응용프로그램을 위한 것이 **아니므로** 응용프로그램 서버(AS)에서 유사한 기능을 교체할 수 **없습니다**.

YEAR

스키마는 SYSIBM입니다.

YEAR 함수는 값의 연도 부분을 리턴합니다.

인수는 날짜, 시간소인, 날짜 기간, 시간소인 지속 기간, 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시간소인의 유효한 문자열 표현이어야 합니다. 유니코드 데이터베이스의 경우 제공된 인수가 그래픽 문자열이면, 함수를 실행하기 전에 먼저 문자열로 변환됩니다.

함수의 결과는 정수(integer)입니다. 인수가 널(NULL)이 될 수 있는 경우 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과는 널(NULL)이 됩니다.

지정된 인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 연도 부분으로 1과 9999 사이의 정수입니다.
- 인수가 날짜 기간이거나 시간소인 지속 기간인 경우:
  - 결과는 값의 연도 부분으로 -9999와 9999 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예를 들면, 다음과 같습니다.

- PROJECT 테이블에서, 동일한 달력 연도에서 시작(PRSTDATE) 및 종료(PRENDATE)하도록 스케줄된 모든 프로젝트를 선택하십시오.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

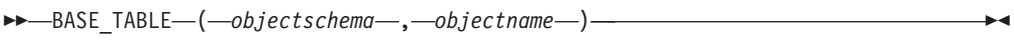
- PROJECT 테이블에서, 1년 이내에 완료되도록 스케줄된 모든 프로젝트를 선택하십시오.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

테이블 함수

테이블 함수는 명령문의 FROM절에서만 사용할 수 있습니다. 테이블 함수는 단순한 CREATE TABLE 문을 통해 작성된 테이블을 닮은 테이블의 컬럼을 리턴합니다. 테이블 함수는 스키마 이름으로 규정될 수 있습니다.

BASE_TABLE



스키마는 SYSPROC입니다.

BASE_TABLE 함수는 별명 체인이 분석된 이후에 발견된 오브젝트의 오브젝트 이름 및 스키마 이름을 리턴합니다. 지정된 objectname(및 objectschema)은 분석의 시작점으로 사용됩니다. 시작점이 별명을 참조하지 않는 경우, 스키마 이름과 시작점의 규정되지 않은 이름이 리턴됩니다. 함수는 다음 컬럼으로 구성되는 단일 행 테이블을 리턴합니다.

표 62. BASE_TABLE 함수에서 리턴한 정보

컬럼 이름	데이터 유형	설명
BASESCHEMA	VARCHAR(128)	별명 체인이 분석된 이후에 발견된 오브젝트의 스키마 이름입니다. 일치하는 별명이 없으면 objectschema를 일치시킵니다.
BASENAME	VARCHAR(128)	별명 체인이 분석된 이후에 발견된 오브젝트의 규정되지 않은 이름입니다. 일치하는 별명이 없으면 objectname을 일치시킵니다. 이름은 테이블, 뷰 또는 정의되지 않은 오브젝트를 식별할 수 있습니다.

objectschema
분석에 앞서 제공된 *objectname* 값을 규정하는 데 사용된 스키마를 나타내는 문자 표현식입니다. *objectschema*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0바이트보다 크고 129바이트보다 작아야 합니다.

objectname
분석될 규정되지 않은 이름을 나타내는 문자 표현식입니다. *objectname*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0바이트보다 크고 129바이트보다 작아야 합니다.

주: BASE_TABLE 테이블 함수는 TABLE_SCHEMA 및 TABLE_NAME 스칼라 함수를 사용할 때 코디네이터 파티션 및 카탈로그 파티션 사이에 발생하는 불필요한 통신을 방지하여 파티션된 데이터베이스 구성에서 성능을 개선합니다.

예 :

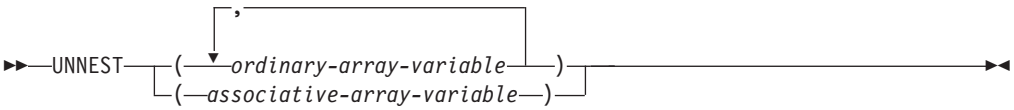
TABLE_SCHEMA 및 TABLE_NAME 함수를 사용하는 다음 명령문이 작성됩니다.

```
SELECT COLCOUNT INTO :H00030
FROM SYSIBM.SYSTABLES
WHERE CREATOR = TABLE_SCHEMA(:H00031 ,:H00032 )
AND NAME = TABLE_NAME(:H00031 ,:H00032 )
```

BASE_TABLE 함수를 사용하는 동일한 명령문을 작성할 수 있습니다.

```
SELECT COLCOUNT INTO :H00030
FROM SYSIBM.SYSTABLES A, TABLE(SYSPROC.BASE_TABLE(:H00032, :H00031)) AS B
WHERE A.CREATOR = B.BASESCHEMA
AND A.NAME = B.BASENAME
```

UNNEST



스키마는 SYSIBM입니다.

UNNEST 함수는 지정된 배열의 각 요소에 대한 행을 포함한 결과 테이블을 리턴합니다. 여러 개의 일반 배열 인수가 지정된 경우 행 수는 가장 큰 카디낼리티(cardinality)의 배열과 일치합니다.

ordinary-array-variable
일반 배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 일반 배열 유형의 매개변수 표시문자에 대한 CAST 스펙.

associative-array-variable
연관된 배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수 또는 연관된 배열 유형의 매개변수 표시문자에 대한 CAST 스펙.

UNNEST 함수에서 생성된 결과 컬럼 이름은 *collection-derived-table*절의 필수 *correlation-clause* 파트로 제공될 수 있습니다.

UNNEST 함수는 배열이 지원되는 컨텍스트의 *collection-derived-table* 절에서만 사용할 수 있습니다(SQLSTATE 42887).

결과 테이블은 입력 인수에 따라 달라집니다.

- 한 개의 일반 배열 인수만 지정한 경우,
  - 배열 요소가 행 데이터 유형이 아닌 경우, 배열 요소 데이터 유형에 일치하는 컬럼 데이터 유형이 포함된 한 개의 컬럼 테이블이 결과입니다.
  - 배열 요소가 행 데이터 유형인 경우, 요소 데이터 유형의 각 행 필드에 대한 한 개 컬럼이 포함된 테이블이 결과입니다. 결과 테이블 컬럼 데이터 유형은 해당 배열 요소 행 필드 데이터 유형과 일치합니다.
- 두 개 이상의 배열 인수가 지정되었지만 배열 요소 모두가 행 데이터 유형을 포함하지 않는 경우 첫 번째 배열은 결과 테이블의 첫 번째 컬럼을 제공하고 두 번째 배열은 두 번째 컬럼을 제공하는 방식입니다. 각 컬럼의 데이터 유형은 해당 배열 인수의 배열 요소 데이터 유형에 일치합니다. 배열의 카디낼리티(cardinalities)가 동일하지 않으면, 결과 테이블의 카디낼리티는 가장 큰 카디낼리티의 배열과 동일합니다. 테이블의 컬럼 값은 배열 인덱스 값이 해당 배열의 카디낼리티(cardinality)보다 큰 모든 행에 대해 널(NULL) 값으로 설정됩니다. 즉, 각 배열이 두 개의 컬럼이 있는 테

이블로 표시되면(하나는 배열 인덱스용이고 다른 하나는 데이터용) UNNEST는 Join 술어로 배열 인덱스에서 등식을 사용하여 배열 사이에서 OUTER JOIN을 수행합니다.

- 한 개의 연관된 배열 인수가 지정된 경우
  - 배열 요소가 행 데이터 유형이 아닌 경우, 2개의 컬럼이 포함된 테이블이 결과이며 여기서 첫 번째 컬렘 데이터 유형은 배열 인덱스 데이터 유형과 일치하고 두 번째 컬럼 데이터 유형은 배열 요소 데이터 유형과 일치합니다.
  - 배열 요소가 행 데이터 유형인 경우 결과는 행 데이터 유형의 필드 수보다 한 개가 더 많은 테이블로 여기서 첫 번째 컬럼 데이터 유형은 배열 인덱스 데이터 유형과 일치하고 나머지 컬럼 데이터 유형은 배열 요소 행 필드 데이터 유형과 일치합니다.
- 오류가 리턴됩니다(SQLSTATE 42884).
  - 둘 이상의 연관된 배열 인수가 지정된 경우
  - 둘 이상의 배열 인수가 지정되고 하나 이상의 배열에 행 유형인 요소 데이터 유형이 포함됩니다.
  - 일반 배열 인수 및 연관된 배열 인수 모두가 지정된 경우

UNNEST 함수로 작성된 결과 컬럼 이름은 *collection-derived-table*의 필수 *correlation-clause* 의 파트로 제공될 수 있습니다.

이 특수 테이블 함수는 FROM절에서 *table-reference*의 *collection-derived-table*에만 사용됩니다.

둘 이상의 배열이 제공되고 하나 이상의 인수가 연관된 배열인 경우 오류가 리턴됩니다(SQLSTATE 42884).

연관된 배열을 중첩되지 않게 할 때 WITH ORDINALITY절을 사용하면 오류가 리턴됩니다(SQLSTATE 428HT).

예

- 배열 유형 PHONENUMBERS의 일반 배열 변수 RECENT_CALLS에는 세 개의 요소 값인 9055553907, 4165554213 및 4085553678만 포함됩니다. 다음 쿼리

```
SELECT T.ID, T.NUM
FROM UNNEST(RECENT_CALLS) WITH ORDINALITY AS T(NUM, ID)
```

  는 다음과 같이 형식화된 테이블을 리턴합니다.

```
ID    NUM
----------
1     9055553907
2     4165554213
3     4085553678
```

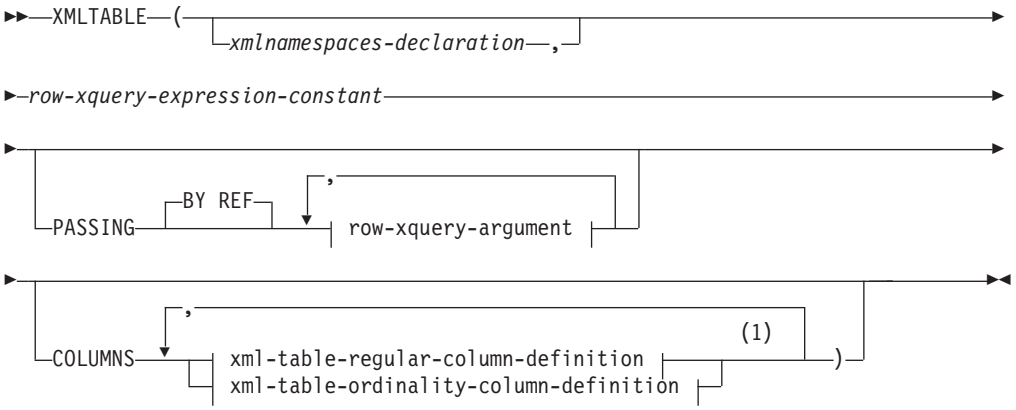
- 배열 유형 PERSONAL_PHONENUMBERS의 배열 변수 PHONELIST에서 개인 전화번호 목록과 인덱스 문자열 값을 리턴합니다. 다음 쿼리

  ```
  SELECT T.ID, T.PHONE
  FROM UNNEST(PHONELIST) AS T(ID, PHONE)
  ```

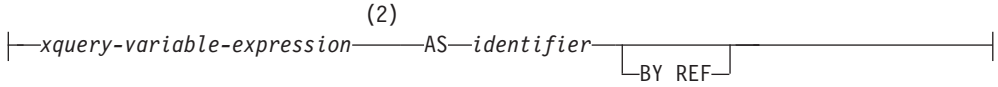
  는 다음과 같이 형식화된 테이블을 리턴합니다.

  ```
  ID    PHONE
  ----------------
  Home  4163053745
  Work  4163053746
  Mom   4164789683
  ```

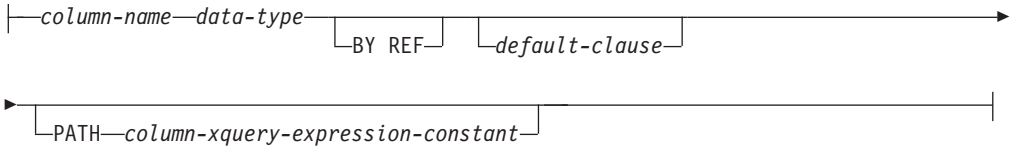
XMLTABLE



row-xquery-argument:



xml-table-regular-column-definition:



xml-table-ordinality-column-definition:

주:

1 xml-table-ordinality-column-definition 절은 두 번 이상 지정할 수 없습니다 (SQLSTATE 42614).

2 표현식의 데이터 유형은 DECFLOAT가 될 수 없습니다.

스키마는 SYSIBM입니다. 함수 이름은 규정된 이름으로 지정할 수 없습니다.

XMLTABLE 함수는 지정된 입력 인수를 XQuery 변수로 사용하여 XQuery 표현식을 평가한 결과 테이블을 리턴합니다. 행 XQuery 표현식의 결과 시퀀스에서 각 시퀀스 항목은 결과 테이블의 행을 나타냅니다.

xmlnamespaces-declaration

row-xquery-expression-constant 및 *column-xquery-expression-constant*의 정적 컨텍스트의 일부가 될 하나 이상의 XML 이름 스페이스 선언을 지정합니다.

XMLTABLE의 인수인 XQuery 표현식의 정적으로 알려진 이름 스페이스 세트는 사전 설정된 정적으로 알려진 이름 스페이스 세트 및 해당 절에 지정된 이름 스페이스 선언 조합입니다. XQuery 표현식 내의 XQuery 프롤로그가 이들 이름 스페이스를 대체할 수도 있습니다.

*xmlnamespaces-declaration*을 지정하지 않을 경우, 사전 설정된 정적으로 알려진 이름 스페이스 세트만이 XQuery 표현식에 적용됩니다.

row-xquery-expression-constant

지원되는 XQuery 언어 구문을 사용하여 XQuery 표현식으로 해석되는 SQL 문자열 상수를 지정합니다. 이 상수 문자열은 데이터베이스 또는 섹션 코드 페이지로 변환되지 않고 직접 UTF-8로 변환됩니다. XQuery 표현식은 선택적인 일련의 입력 XML 값을 사용하여 실행되며, 시퀀스의 각 항목에 대한 행을 생성하는 출력 XQuery 시퀀스를 리턴합니다. *row-xquery-expression-constant*의 값은 빈 문자열 또는 공백 문자의 문자열이 아니어야 합니다(SQLSTATE 10505).

PASSING

입력 값 및 값을 *row-xquery-expression-constant*에서 지정한 XQuery 표현식으로 전달하는 방법을 지정합니다. 디폴트로, 함수가 호출되는 범위 내에 있는 모든 고유 컬럼 이름은 변수 이름으로 컬럼의 이름을 사용하여 XQuery 표현식으로 내재적으로 전달됩니다. 지정된 row-xquery-argument의 *identifier*가 범위 내 컬럼 이름과 일치하는 경우 명시적 row-xquery-argument가 XQuery 표현식에 전달되어 해당되는 내재된 컬럼을 대체합니다.

BY REF

XML 입력 인수가 디폴트로 참조에 의해 전달됨을 지정합니다. 참조에 의해 XML 값이 패스되면 XQuery 평가는 지정된 입력 표현식으로부터 직접 입력 노드 트리(있는 경우)를 사용하여 원본 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보유합니다. 두 개의 인수가 동일한 XML 값을 패스할 경우, 두 입력 인수 사이에 들어 있는 일부 노드가 관련된 노드 ID 비교 및 문서 순서화 비교는 동일한 XML 노드 트리에 있는 노드를 참조할 수 있습니다.

이 절은 비XML 값이 패스되는 방법에 아무 영향을 주지 않습니다. 비XML 값은 XML로 캐스트되는 동안 값의 사본을 새로 작성합니다.

row-xquery-argument

*row-xquery-expression-constant*에서 지정한 XQuery 표현식에 전달할 인수를 지정합니다. 인수는 값과 값을 전달할 방법을 지정합니다. 인수에는 결과를 XQuery 표현식에 전달하기 전에 평가되는 SQL 표현식이 포함됩니다.

- 결과 값이 XML 유형인 경우 이는 *input-xml-value*로 됩니다. 널(NULL) XML 값은 빈 XML 시퀀스로 변환됩니다.

- 결과 값이 XML 유형이 아닌 경우 이는 XML 데이터 유형으로 캐스트될 수 있어야 합니다. 널(NULL) 값은 빈 XML 시퀀스로 변환됩니다. 변환된 값은 *input-xml-value*로 됩니다.

*row-xquery-expression-constant*를 평가할 때, XQuery 변수는 *input-xml-value*와 동일한 값 및 AS절에서 지정한 이름으로 표시됩니다.

xquery-variable-expression

값이 실행 도중 *row-xquery-expression-constant*에서 지정한 XQuery 표현식에 사용 가능한 SQL 표현식을 지정합니다. 표현식은 NEXT VALUE 표현식, PREVIOUS VALUE 표현식(SQLSTATE 428F9) 또는 OLAP 함수를 포함할 수 없습니다(SQLSTATE 42903).표현식의 데이터 유형은 DECFLOAT가 될 수 없습니다.

AS *identifier*

*xquery-variable-expression*에 의해 생성된 값이 *row-xquery-expression-constant*에 XQuery 변수로 전달됨을 지정합니다. 변수 이름은 *identifier*입니다. *identifier*에는 XQuery 언어에서 변수 이름 앞에 오는 달러 부호($)가 포함되지 않습니다. 이 ID는 유효 XQuery 변수 이름이어야 하며 XML NCName으로 제한됩니다. ID의 길이는 128바이트를 초과해서는 안됩니다. 동일한 PASSING절에서 두 개의 인수가 동일한 ID를 사용할 수 없습니다(SQLSTATE 42711).

BY REF

XML 입력 값이 참조에 의해 패스되는 것을 표시합니다. 참조에 의해 XML 값이 패스되면 XQuery 평가는 지정된 입력 표현식으로부터 직접 입력 노드 트리(있는 경우)를 사용하여 원본 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보유합니다. 두 개의 인수가 동일한 XML 값을 패스할 경우, 두 입력 인수 사이에 들어 있는 일부 노드가 관련된 노드 ID 비교 및 문서 순서화 비교는 동일한 XML 노드 트리에 있는 노드를 참조할 수 있습니다. BY REF를 *xquery-expression-variable* 다음에 지정하지 않을 경우, XML 인수는 PASSING 키워드 뒤에 나오는 구문을 통해 제공되는 디폴트 전달 메커니즘의 방식으로 전달됩니다. XML이 아닌 값에는 이 옵션을 지정할 수 없습니다(SQLSTATE 42636). 비XML 값이 패스될 때 이 값은 XML로 변환되며 이 프로세스는 사본을 작성합니다.

COLUMNS

결과 테이블의 출력 컬럼을 지정합니다.. 이 절을 지정하지 않는 경우 참조에 의해 데이터 유형 XML의 이름 지정되지 않은 단일 컬럼이 리턴됩니다. 값은 *row-xquery-expression-constant*(PATH '.'를 지정하는 것과 같음)에서 XQuery 표현식을 평가한 시퀀스 항목을 기반으로 합니다. 결과 컬럼을 참조하려면 함수 다음에 *correlation-clause*에서 *column-name*을 지정해야 합니다.

xml-table-regular-column-definition

행의 시퀀스 항목에서 값을 추출하기 위해 컬럼 이름, 데이터 유형, XML 전달 메커니즘 및 XQuery 표현식을 포함한 결과 테이블의 출력 컬럼을 지정합니다.

column-name

결과 테이블의 컬럼 이름을 지정합니다. 이름을 규정할 수 없으며 테이블의 둘 이상의 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

data-type

컬럼의 데이터 유형을 지정합니다. 구문 및 사용 가능 유형에 대한 설명은 CREATE TABLE을 참조하십시오. XML 데이터 유형에서 지정된 *data-type*으로 지원되는 XMLCAST가 있을 경우 *data-type*을 XMLTable에서 사용할 수도 있습니다.

BY REF

XML 값이 XML 데이터 유형의 컬럼에 대해 참조에 의해 리턴됨을 지정합니다. 디폴트로 XML 값은 참조에 의래 리턴됩니다. XML 값이 참조에 의해 리턴될 경우, XML 값은 존재하는 경우 결과 값으로부터 직접 입력 노드 트리를 사용하며 원래 노드 ID 및 문서 순서를 포함한 모든 등록 정보를 보존합니다. XML이 아닌 컬럼에는 이 옵션을 지정할 수 없습니다(SQLSTATE 42636). XML이 아닌 컬럼을 처리하면 값이 XML로부터 변환됩니다. 이 프로세스는 사본을 작성합니다.

default-clause

컬럼의 디폴트값을 지정합니다. *default-clause*에 대한 구문 및 설명은 CREATE TABLE을 참조하십시오. XMLTABLE 결과 컬럼의 경우 디폴트값은 *column-xquery-expression-constant*에 포함된 XQuery 표현식이 공백 시퀀스를 리턴할 때 적용됩니다.

PATH *column-xquery-expression-constant*

지원되는 XQuery 언어 구문을 사용하여 XQuery 표현식으로 해석되는 SQL 문자열 상수를 지정합니다. 이 상수 문자열은 데이터베이스 또는 섹션 코드 페이지로 변환되지 않고 직접 UTF-8로 변환됩니다. *column-xquery-expression-constant*는 *row-xquery-expression-constant*에서 XQuery 표현식 평가의 결과 항목에 대한 컬럼 값을 판별하는 XQuery 표현식을 지정합니다. 외부에서 제공된 컨텍스트 항목으로 *row-xquery-expression-constant*를 처리한 결과로부터 항목이 제공되면, *column-xquery-expression-constant*가 평가되며 출력 시퀀스를 리턴합니다. 컬럼 값은 다음과 같이 해당 출력 시퀀스에 따라 결정됩니다.

- 출력 시퀀스가 제로(0) 항목을 포함할 경우, *default-clause*에서 컬럼 값을 제공합니다.

- 공백 시퀀스가 리턴되고 *default-clause*를 지정하지 않은 경우, 널(NULL) 값이 컬럼에 지정됩니다.
- 공백이 아닌 시퀀스가 리턴될 경우, 값은 컬럼에 지정된 *data-type*에 대한 XMLCAST입니다. 해당 XMLCAST 처리에서 오류가 발생할 수 있습니다.

*column-xquery-expression-constant*의 값은 빈 문자열 또는 공백 문자의 문자열이 아니어야 합니다(SQLSTATE 10505). 이 절을 지정하지 않으면, 디폴트 XQuery 표현식은 *column-name*입니다.

xml-table-ordinality-column-definition

결과 테이블의 순서 컬럼을 지정합니다.

column-name

결과 테이블의 컬럼 이름을 지정합니다. 이름을 규정할 수 없으며 테이블의 둘 이상의 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

FOR ORDINALITY

*column-name*이 결과 테이블의 순서 컬럼임을 지정합니다. 이 컬럼의 데이터 유형은 BIGINT입니다. 결과 테이블에서 이 컬럼의 값은 *row-xquery-expression-constant*의 XQuery 표현식 평가 결과로 생성되는 시퀀스에서 해당 행에 대한 항목의 순차 번호입니다.

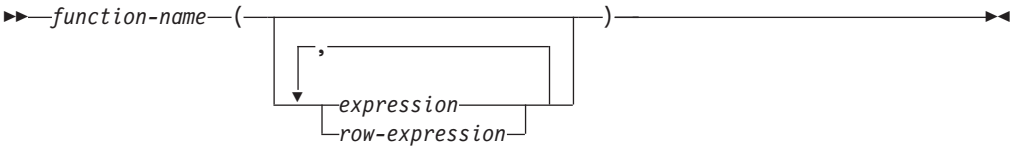
XQuery 표현식의 평가 결과 오류가 발생할 경우, XMLTABLE 함수는 XQuery 오류를 리턴합니다(SQLSTATE 클래스 '10').

예를 들면, 다음과 같습니다.

- 상태가 'NEW'인 주문에 대한 구매 주문 항목을 테이블 결과로 나열합니다.

```
   SELECT U."PO ID", U."Part #", U."Product Name",
       U."Quantity", U."Price", U."Order Date"
FROM PURCHASEORDER P,
       XMLTABLE(XMLNAMESPACES('http://podemo.org' AS "pod"),
                              '$po/PurchaseOrder/itemlist/item' PASSING P.PORDER AS "po"
                COLUMNS "PO ID"         INTEGER       PATH '../../@POid',
                                        "Part #"        CHAR(6)       PATH 'product/@pid',
                                        "Product Name"  CHAR(50)      PATH 'product/pod:name',
                                        "Quantity"      INTEGER       PATH 'quantity',
                                        "Price"         DECIMAL(9,2)  PATH 'product/pod:price',
                                        "Order Date"    TIMESTAMP     PATH '../../dateTime'
                              ) AS U
     WHERE P.STATUS = 'NEW'
```

사용자 정의 함수(UDF)



*사용자 정의 함수(UDF)*는 SQL 언어의 기존 내장 함수에 대한 추가사항 또는 확장입니다. 사용자 정의 함수(UDF)는 호출될 때마다 단일 값을 리턴하는 스칼라 함수, 유사한 값 세트로 전달되어 해당 세트의 단일 값을 리턴하는 집계 함수, 하나의 행을 리턴하는 행 함수 또는 테이블을 리턴하는 테이블 함수일 수 있습니다.

많은 사용자 정의 함수(UDF)가 SYSFUN 및 SYSPROC 스키마에서 제공됩니다.

UDF가 기존 집계 함수에 대한 소스일 경우에만 집계 함수일 수 있습니다. UDF는 뒤에 함수 인수(있을 경우)를 묶는 괄호가 있는 규정되거나 규정되지 않은 함수 이름에 의해 참조됩니다. 사용자 정의 컬럼 또는 데이터베이스로 등록된 스칼라 함수는 모든 내장 함수가 나타날 수 있는 같은 컨텍스트에서 참조될 수 있습니다. 사용자 정의 행 함수는 사용자 정의 유형에 대한 변형 함수로 등록될 때에만 참조될 수 있습니다. 데이터베이스로 등록된 사용자 정의 테이블 함수는 SELECT문의 FROM절에서만 참조될 수 있습니다.

함수 인수는 데이터베이스로 등록될 때 사용자 정의 함수에 대해 지정된 매개변수의 위치 및 수와 일치해야 합니다. 또한 인수는 해당 정의된 매개변수의 데이터 유형으로 승격 가능한 데이터 유형이어야 합니다.

함수의 결과는 RETURNS절에 지정됩니다. UDF가 등록될 때 정의된 RETURNS절은 함수가 테이블 함수인지를 판별합니다. 함수가 등록될 때 RETURNS NULL ON NULL INPUT절이 지정된 경우(또는 디폴트로 지정된 경우) 인수가 널(NULL)이면 결과도 널(NULL)입니다. 테이블 함수의 경우 이는 행이 없는 리턴 테이블(즉, 비어 있는 테이블)을 의미하는 것으로 해석됩니다.

규칙 및 행 데이터 유형에 대한 자세한 정보는 ″행 표현식″을 참조하십시오.

다음은 사용자 정의 함수(UDF)의 예입니다.

- ADDRESS라는 스칼라 UDF는 스크립트 형식으로 저장된 이력서에서 집 주소를 추출합니다. ADDRESS 함수는 CLOB을 예상하고 VARCHAR(4000) 값을 리턴합니다.

  ```
  SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME
    WHERE RESUME_FORMAT = 'SCRIPT'
  ```

- 테이블 T2에는 숫자 컬럼 A가 있습니다. 이전 예에서 ADDRESS라는 스칼라 UDF를 호출합니다.

```
SELECT ADDRESS(A) FROM T2
```

인수에서 승격 가능한 매개변수 및 일치하는 이름의 함수가 존재하지 않으므로 오류가 발생합니다(SQLSTATE 42884).

- WHO라는 테이블 UDF는 명령문이 실행될 때 사용 중이던 서버 머신의 세션에 대한 정보를 리턴합니다. WHO 함수는 키워드 TABLE 및 필수 상관 변수를 포함하는 FROM절 내부에서 호출됩니다. WHO() 테이블의 컬럼 이름은 CREATE FUNCTION문에서 정의되어 있습니다.

```
SELECT ID, START_DATE, ORIG_MACHINE
  FROM TABLE( WHO() ) AS QQ
  WHERE START_DATE LIKE 'MAY%'
```

제 4 장 프로시저

프로시저 개요

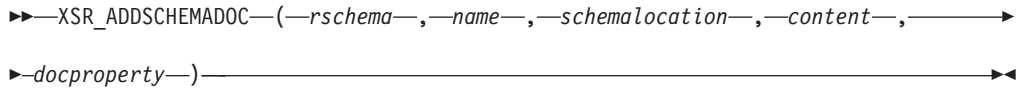
프로시저는 SQL CALL 문을 통해 시작할 수 있는 응용프로그램입니다. 프로시저는 괄호 안에 묶여 있는 인수가 뒤에 올 수 있는 프로시저 이름으로 지정됩니다.

프로시저 인수는 다른 유형이 될 수 있고 다른 의미를 가질 수 있는 개별적인 스칼라 값입니다. 값을 프로시저로 전달하거나 프로시저에서 리턴 값을 수신하거나, 둘 다를 위해 인수를 사용할 수 있습니다.

사용자 정의 프로시저는 CREATE PROCEDURE 문을 사용하여 SYSCAT.ROUTINES에서 데이터베이스에 등록된 프로시저입니다. 이와 같은 한 세트의 함수가 SYSFUN 스키마에서, 다른 세트가 SYSPROC 스키마에서 데이터베이스 관리 프로그램과 함께 제공됩니다.

프로시저는 스키마 이름으로 규정될 수 있습니다.

XSR_ADDSCHEMADOC



스키마는 SYSPROC입니다.

XML 스키마 저장소(XSR)의 각 XML 스키마는 하나 이상의 XML 스키마 문서로 구성될 수 있습니다. XML 스키마가 여러 개의 문서로 구성되는 경우, XSR_ADDSCHEMADOC 스토어드 프로시저를 사용하여 1차 XML 스키마 문서가 아닌 다른 모든 XML 스키마를 추가합니다.

권한 부여

프로시저 호출자의 권한 부여 ID는 카탈로그 뷰 SYSCAT.XSROBJECTS에 기록된 대로 XSR의 소유자여야 합니다.

rschema

XML 스키마의 SQL 스키마를 지정하는 VARCHAR(128) 유형의 입력 인수. SQL 스키마는 XSR에서 XML 스키마를 식별하는 데 사용되는 SQL ID의 한 파트입니다. 이 스키마는 완료 상태로 이동됩니다. (SQL ID의 기타 파트는 이름 인수에 의해 제공됩니다.) 이 인수는 NULL 값을 가질 수 있습니다. 이는 CURRENT

SCHEMA 특수 레지스터에 정의된 디폴트 SQL 스키마를 사용함을 표시합니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다. XSR 오브젝트는 XML 스키마 저장소 외부의 오브젝트가 아닌 다른 이름 공간 내에서 발생하기 때문에 XSR 외부에 존재하는 데이터베이스 오브젝트와는 이름 충돌이 발생하지 않습니다.

name

XML 스키마의 이름을 지정하는 VARCHAR(128) 유형의 입력 인수. XML 스키마의 완전한 SQL ID는 *rschema.name*입니다. XML 스키마 이름은 XSR_REGISTER 스토어드 프로시저 호출 결과로 이미 존재하고 있어야 하며 XML 스키마 등록은 아직 완료될 수 없습니다. 이 인수는 널(NULL) 값을 가질 수 없습니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

schemalocation

널(NULL) 값을 가질 수 있는 VARCHAR(1000) 유형의 입력 인수로서 XML 스키마가 추가되는 1차 XML 스키마 문서의 스키마 위치를 표시합니다. 이 인수는 XML 스키마의 외부 이름으로서, xsi:schemaLocation 속성을 사용하여 XML 인스턴스에서 1차 문서를 식별할 수 있습니다.

content

추가되는 XML 스키마 문서의 내용이 포함되어 있는 BLOB(30M) 유형의 입력 매개변수. 이 인수는 널(NULL) 값을 가질 수 없습니다. XML 스키마 문서를 제공해야 합니다.

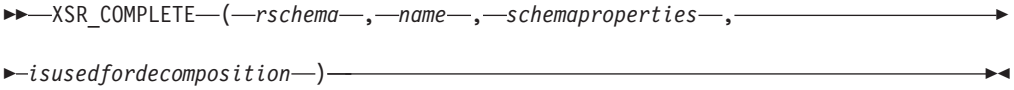
docproperty

추가되는 XML 스키마 문서의 등록 정보를 표시하는 BLOB(30M) 유형의 입력 매개변수. 이 매개변수는 널(NULL) 값을 가질 수 있습니다. 널(NULL) 값이 아니면 값은 XML 문서입니다.

예:

```
CALL SYSPROC.XSR_ADDSCHEMADOC(
       'user1',
       'POschema',
  'http://myPOschema/address.xsd',
       :content_host_var,
  0)
```

XSR_COMPLETE



스키마는 SYSPROC입니다.

XSR_COMPLETE 프로시저는 XML 스키마 등록 프로세스의 일부로서 호출될 최종 스토어드 프로시저이며, XML 스키마를 XML 스키마 저장소(XSR)에 등록합니다. XML 스키마는 스토어드 프로시저에 대한 호출을 통해 스키마 등록이 완료될 때까지 유효성 확인에 사용할 수 없습니다.

권한 부여:

프로시저 호출자의 권한 부여 ID는 카탈로그 뷰 SYSCAT.XSROBJECTS에 기록된 대로 XSR의 소유자여야 합니다.

rschema

XML 스키마의 SQL 스키마를 지정하는 VARCHAR(128) 유형의 입력 인수. SQL 스키마는 XSR에서 XML 스키마를 식별하는 데 사용되는 SQL ID의 한 파트입니다. 이 스키마는 완료 상태로 이동됩니다. (SQL ID의 기타 파트는 이름 인수에 의해 제공됩니다.) 이 인수는 NULL 값을 가질 수 있습니다. 이는 CURRENT SCHEMA 특수 레지스터에 정의된 디폴트 SQL 스키마를 사용함을 표시합니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다. XSR 오브젝트는 XML 스키마 저장소 외부의 오브젝트가 아닌 다른 이름 공간 내에서 발생하기 때문에 XSR 외부에 존재하는 데이터베이스 오브젝트와는 이름 충돌이 발생하지 않습니다.

name

XML 스키마의 이름을 지정하는 VARCHAR(128) 유형의 입력 인수. XML 스키마의 완전한 SQL ID(완료 점검이 수행되는)는 *rschema.name*입니다. XML 스키마 이름은 XSR_REGISTER 스토어드 프로시저 호출 결과로 이미 존재하고 있어야 하며 XML 스키마 등록은 아직 완료될 수 없습니다. 이 인수는 널(NULL) 값을 가질 수 없습니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

schemaproperties

XML 스키마와 연관되는 등록 정보(있는 경우)를 지정하는 BLOB(5M) 유형의 입력 인수. 이 인수의 값은 연관된 등록 정보가 없는 경우 NULL이거나, XML 스키마의 등록 정보를 나타내는 XML 문서입니다.

isusedfordecomposition

XML 스키마가 분석에 사용될지 여부를 표시하는 정수 유형의 입력 매개변수입니다. 분석에 XML 스키마가 사용되는 경우 이 값은 1로 설정해야 합니다. 그렇지 않으면 0으로 설정해야 합니다.

예:

```
CALL SYSPROC.XSR_COMPLETE(
       'user1',
       'POschema',
  :schemaproperty_host_var,
  0)
```

XSR_DTD

스키마는 SYSPROC입니다.

XSR_DTD 프로시저는 XML 스키마 저장소(XSR)에 문서 유형 선언(DTD)을 등록합니다.

권한 부여

프로시저 호출자의 권한 부여 ID는 다음 중 하나 이상이 있어야 합니다.

- DBADM 권한
- SQL 스키마가 존재하지 않을 경우 IMPLICIT_SCHEMA 데이터베이스 권한
- SQL 스키마가 존재할 경우 CREATEIN 특권

rschema

외부 엔티티의 SQL 스키마를 지정하는 유형 VARCHAR(128)의 입력및 출력 DTD의 SQL 스키마를 지정하는 VARCHAR(128) 유형의 인수. SQL 스키마는 XSR에서 이 DTD를 식별하는 데 사용되는 SQL ID의 한 파트입니다. (SQL ID의 다른 파트는 *name* 인수에 의해 제공됩니다.) 이 인수는 NULL 값을 가질 수 있습니다. 이는 CURRENT SCHEMA 특수 레지스터에 정의된 디폴트 SQL 스키마를 사용함을 표시합니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다. 이 값에 문자열 'SYS'로 시작하는 관계 스키마를 사용해서는 안 됩니다. XSR 오브젝트는 XML 스키마 저장소 외부의 오브젝트가 아닌 다른 이름 공간 내에서 발생하기 때문에 XSR 외부에 존재하는 데이터베이스 오브젝트와는 이름 충돌이 발생하지 않습니다.

name

DTD의 이름을 지정하는 VARCHAR(128) 유형의 입력 및 출력 인수입니다. DTD의 완전한 SQL ID는 *rschema.name*이고 XSR의 모든 오브젝트에 걸쳐 고유해야 합니다. 이 인수는 널(NULL) 값을 허용합니다. 이 인수에 널(NULL) 값을 제공하면 고유한 값이 생성되어 XSR 내에 저장됩니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

systemid

DTD의 시스템 ID를 지정하는 VARCHAR(1000) 유형의 입력 매개변수입니다. DTD의 시스템 ID는 XML 인스턴스 문서의 DOCTYPE 선언 또는 ENTITY 선언(사용되는 경우 SYSTEM 키워드가 접두부로 붙는)에 있는 DTD의 URI(uniform resource identifier)와 일치해야 합니다. 이 인수는 널(NULL) 값을 가질 수 없습니다. 시스템 ID는 공용 ID와 함께 지정될 수 있습니다.

publicid

DTD의 공용 ID를 지정하는 유형 VARCHAR (1000)의 입력 매개변수입니다. DTD의 공용 ID는 XML 인스턴스 문서의 DOCTYPE 선언 또는 ENTITY 선언(사용되는 경우 PUBLIC 키워드가 접두부로 붙는)에 있는 DTD의 URI(uniform resource identifier)와 일치해야 합니다. 이 인수는 널(NULL) 값을 승인하며 XML 인스턴스 문서의 DOCTYPE 선언 또는 ENTITY 선언에도 지정되는 경우에만 사용해야 합니다.

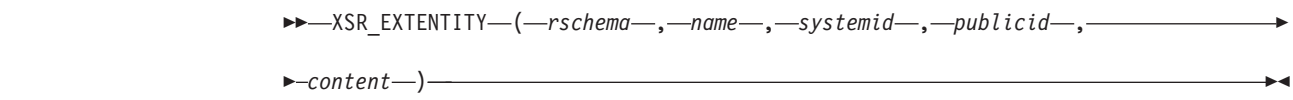
content

DTD의 내용이 포함되어 있는 BLOB(30M) 유형의 입력 매개변수입니다. 이 인수는 널(NULL) 값을 가질 수 없습니다.

예: 시스템 ID *http://www.test.com/person.dtd* 및 공용 ID *http://www.test.com/person*로 식별되는 DTD를 등록하십시오.

```
CALL SYSPROC.XSR_DTD ( 'MYDEPT' ,
      'PERSONDTD' ,
      'http://www.test.com/person.dtd' ,
      'http://www.test.com/person',
      :content_host_variable
   )
```

XSR_EXTENTITY



스키마는 SYSPROC입니다.

XSR_EXTENTITY 프로시저는 외부 엔티티를 XML 스키마 저장소(XSR)에 등록합니다.

권한 부여

프로시저 호출자의 권한 부여 ID는 다음 중 최소한 하나를 갖고 있어야 합니다.

- DBADM 권한
- SQL 스키마가 존재하지 않을 경우 IMPLICIT_SCHEMA 데이터베이스 권한
- SQL 스키마가 존재할 경우 CREATEIN 특권

rschema

외부 엔티티의 SQL 스키마를 지정하는 유형 VARCHAR(128)의 입력및 출력 인수입니다. SQL 스키마는 XSR에서 이 외부 엔티티를 식별하는 데 사용되는 SQL ID의 한 파트입니다. (SQL ID의 다른 파트는 *name* 인수에 의해 제공됩니다.) 이

인수는 NULL 값을 가질 수 있습니다. 이는 CURRENT SCHEMA 특수 레지스터에 정의된 디폴트 SQL 스키마를 사용함을 표시합니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다. 이 값에 문자열 'SYS'로 시작하는 관계 스키마를 사용해서는 안 됩니다. XSR 오브젝트는 XML 스키마 저장소 외부의 오브젝트가 아닌 다른 이름 공간 내에서 발생하기 때문에 XSR 외부에 존재하는 데이터베이스 오브젝트와는 이름 충돌이 발생하지 않습니다.

name

외부 엔티티의 이름을 지정하는 VARCHAR(128) 유형의 입력 및 출력 인수입니다. 외부 엔티티의 완전한 SQL ID는 *rschema.name*이고 XSR의 모든 오브젝트에 걸쳐 고유해야 합니다. 이 인수는 널(NULL) 값을 허용합니다. 이 인수에 널(NULL) 값을 제공하면 고유한 값이 생성되어 XSR 내에 저장됩니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

systemid

외부 엔티티의 시스템 ID를 지정하는 VARCHAR(1000) 유형의 입력 매개변수입니다. 외부 엔티티의 시스템 ID는 ENTITY 선언(사용되는 경우 SYSTEM 키워드가 접두부로 붙는)에 있는 외부 엔티티의 URI(uniform resource identifier)와 일치해야 합니다. 이 인수는 널(NULL) 값을 가질 수 없습니다. 시스템 ID는 공용 ID와 함께 지정될 수 있습니다.

publicid

외부 엔티티의 공용 ID를 지정하는 VARCHAR(1000) 유형의 입력 매개변수입니다. 외부 엔티티의 공용 ID는 ENTITY 선언(사용되는 경우 PUBLIC 키워드가 접두부로 붙는)에 있는 외부 엔티티의 URI(uniform resource identifier)와 일치해야 합니다. 이 인수는 널(NULL) 값을 승인하며 XML 인스턴스 문서의 DOCTYPE 선언 또는 ENTITY 선언에도 지정되는 경우에만 사용해야 합니다.

content

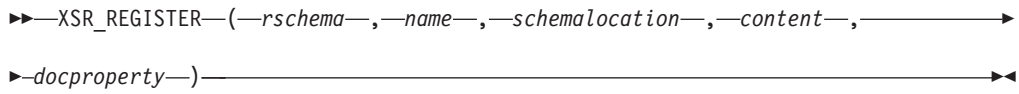
외부 엔티티 문서의 내용이 포함되어 있는 BLOB(30M) 유형의 입력 매개변수입니다. 이 인수는 널(NULL) 값을 가질 수 없습니다.

예: 시스템 ID *http://www.test.com/food/chocolate.txt* 및 *http://www.test.com/food/cookie.txt*로 식별되는 외부 엔티티를 등록하십시오.

```
CALL SYSPROC.XSR_EXTENTITY ( 'FOOD' ,
      'CHOCLATE' ,
      'http://www.test.com/food/chocolate.txt' ,
      NULL ,
      :content_of_chocolate.txt_as_a_host_variable
   )

CALL SYSPROC.XSR_EXTENTITY ( 'FOOD' ,
      'COOKIE' ,
```

```
  'http://www.test.com/food/cookie.txt' ,
  NULL ,
  :content_of_cookie.txt_as_a_host_variable
)
```

XSR_REGISTER



스키마는 SYSPROC입니다.

XSR_REGISTER 프로시저는 XML 스키마 등록 프로세스의 일부로서 호출될 첫 번째 스토어드 프로시저이며, XML 스키마를 XML 스키마 저장소(XSR)에 등록합니다.

권한 부여

프로시저 호출자의 권한 부여 ID는 다음 중 하나 이상이 있어야 합니다.

- DBADM 권한
- SQL 스키마가 존재하지 않을 경우 IMPLICIT_SCHEMA 데이터베이스 권한
- SQL 스키마가 존재할 경우 CREATEIN 특권

rschema

외부 엔티티의 SQL 스키마를 지정하는 유형 VARCHAR(128)의 입력및 출력 XML 스키마의 SQL 스키마를 지정하는 VARCHAR(128) 유형의 인수. SQL 스키마는 XSR에서 XML 스키마를 식별하는 데 사용되는 SQL ID의 한 파트입니다. (SQL ID의 기타 파트는 이름 인수에 의해 제공됩니다.) 이 인수는 NULL 값을 가질 수 있습니다. 이는 CURRENT SCHEMA 특수 레지스터에 정의된 디폴트 SQL 스키마를 사용함을 표시합니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다. 이 값에 문자열 'SYS'로 시작하는 관계 스키마를 사용해서는 안 됩니다. XSR 오브젝트는 XML 스키마 저장소 외부의 오브젝트가 아닌 다른 이름 공간 내에서 발생하기 때문에 XSR 외부에 존재하는 데이터베이스 오브젝트와는 이름 충돌이 발생하지 않습니다.

name

XML 스키마의 이름을 지정하는 VARCHAR(128) 유형의 입력 및 출력 인수. XML 스키마의 완전한 SQL ID는 *rschema.name*이고 XSR의 모든 오브젝트에 걸쳐 고유해야 합니다. 이 인수는 널(NULL) 값을 허용합니다. 이 인수에 널(NULL) 값을 제공하면 고유한 값이 생성되어 XSR 내에 저장됩니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

schemalocation

널(NULL) 값을 가질 수 있는 VARCHAR(1000) 유형의 입력 인수로서 1차 XML

스키마 문서의 스키마 위치를 표시합니다. 이 인수는 XML 스키마의 외부 이름으로서, xsi:schemaLocation 속성을 사용하여 XML 인스턴스에서 1차 문서를 식별할 수 있습니다.

content

1차 XML 스키마 문서의 내용이 포함되어 있는 BLOB(30M) 유형의 입력 매개변수. 이 인수는 널(NULL) 값을 가질 수 없습니다. XML 스키마 문서를 제공해야 합니다.

docproperty

1차 XML 스키마 문서의 등록 정보를 표시하는 BLOB(30M) 유형의 입력 매개변수. 이 매개변수는 널(NULL) 값을 가질 수 있습니다. 널(NULL) 값이 아니면 값은 XML 문서입니다.

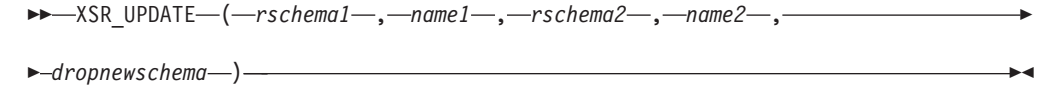
예: 다음 예는 명령행에서 XSR_REGISTER 프로시저를 호출하는 방법을 나타냅니다.

```
CALL SYSPROC.XSR_REGISTER(
    'user1',
    'POschema',
    'http://myPOschema/PO.xsd',
    :content_host_var,
    :docproperty_host_var)
```

예: 다음 예는 Java 응용프로그램에서 XSR_REGISTER 프로시저를 호출하는 방법을 나타냅니다.

```
stmt = con.prepareCall("CALL SYSPROC.XSR_REGISTER (?, ?, ?, ?, ?)");
String xsrObjectName = "myschema1";
String xmlSchemaLocation = "po.xsd";
stmt.setNull(1, java.sql.Types.VARCHAR);
stmt.setString(2, xsrObjectName);
stmt.setString(3, xmlSchemaLocation);
stmt.setBinaryStream(4, buffer, (int)length);
stmt.setNull(5, java.sql.Types.BLOB);
stmt.registerOutParameter(1, java.sql.Types.VARCHAR);
stmt.registerOutParameter(2, java.sql.Types.VARCHAR);
stmt.execute();
```

XSR_UPDATE



스키마는 SYSPROC입니다.

XSR_UPDATE 스토어드 프로시저는 XML 스키마 저장소 (XSR)의 기존 XML 스키마를 전개하는 데 사용됩니다. 기존 XML 스키마를 수정하거나 확장하여 기존 문서 및 새로 삽입된 XML 문서 모두의 유효성을 확인하는 데 사용할 수 있습니다.

XSR_UPDATE에 인수로 지정된 새 XML 스키마 및 원래 XML 스키마 모두가 프로시저가 호출되기 전에 XSR에 등록되고 완료됩니다. 이들 XML 스키마는 호환 가능해야 합니다. 호환성 요구사항에 관한 세부사항은 *XML 스키마 전개를 위한 호환성 요구사항*을 참조하십시오.

권한 부여

프로시저 호출자의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- DBADM 권한
- 카탈로그 뷰 SYSCAT.XSROBJECTS 및 SYSCAT.XSROBJECTCOMPONENTS의 SELECT 특권 및 다음 특권 세트 중 하나.
  - SQL 스키마 *rschema1* 및 오브젝트 이름 *name1*에서 지정한 XML 스키마의 소유자
  - *rschema1* 인수에서 지정한 SQL 스키마의 ALTERIN 특권 및 *dropnewschema* 인수가 0이 아닌 경우, *rschema2* 인수에서 지정한 SQL 스키마의 DROPIN 특권.

rschema1

갱신될 원래 XML 스키마의 SQL 스키마를 지정하는 VARCHAR(128) 유형의 입력 인수입니다. SQL 스키마는 XSR에서 XML 스키마를 식별하는 데 사용되는 SQL ID의 한 파트입니다. (SQL ID의 다른 파트는 *name1* 인수에 의해 제공됩니다.) 이 인수는 널(NULL) 값을 가질 수 없습니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

name1

갱신될 원래 XML 스키마의 이름을 지정하는 VARCHAR(128) 유형의 입력 인수입니다. XML 스키마의 완전한 SQL ID는 *rschema1.name1*입니다. 이 XML 스키마는 이미 XSR에 등록되고 완료되어 있어야 합니다. 이 인수는 널(NULL) 값을 가질 수 없습니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

rschema2

원래 XML 스키마 갱신에 사용될 새로운 XML 스키마의 SQL 스키마를 지정하는 VARCHAR(128) 유형의 입력 인수입니다. SQL 스키마는 XSR에서 XML 스키마를 식별하는 데 사용되는 SQL ID의 한 파트입니다. (SQL ID의 다른 파트는 *name2* 인수에 의해 제공됩니다.) 이 인수는 널(NULL) 값을 가질 수 없습니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

name2

원래 XML 스키마 갱신에 사용될 새로운 XML 스키마의 이름을 지정하는 VARCHAR(128) 유형의 입력 인수입니다. XML 스키마의 완전한 SQL ID는

*rschema2.name2*입니다. 이 XML 스키마는 이미 XSR에 등록되고 완료되어 있어야 합니다. 이 인수는 널(NULL) 값을 가질 수 없습니다. 모든 SQL ID에 적용되는 유효한 문자 및 분리문자에 대한 규칙이 이 인수에도 적용됩니다.

dropnewschema

원래 XML 스키마 갱신에 사용된 후에 새로운 XML 스키마를 삭제할지 여부를 지정하는 정수 유형의 입력 매개변수입니다. 이 매개변수를 0이 아닌 값으로 설정하면 새로운 XML 스키마가 삭제됩니다. 이 인수는 널(NULL) 값을 가질 수 없습니다.

예:

```
CALL SYSPROC.XSR_UPDATE(
  'STORE',
  'PROD',
  'STORE',
  'NEWPROD',
  1)
```

XML 스키마 STORE.PROD의 컨텐츠는 STORE.NEWPROD의 컨텐츠로 갱신되며 XML 스키마 STORE.NEWPROD가 삭제됩니다.

제 5 장 SQL 쿼리

SQL 쿼리

*쿼리*는 결과 테이블을 지정합니다. 쿼리는 특정 SQL문의 구성요소입니다. 쿼리 양식에는 세 가지가 있습니다.

- subselect
- fullselect
- select-statement.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 권한 중 하나를 가지고 있어야 합니다.

- 쿼리에서 식별되는 각 테이블 또는 뷰의 경우, 다음 중 하나:
  - 테이블 또는 뷰에 대한 SELECT 특권
  - 테이블 또는 뷰에 대한 CONTROL 특권
- DATAACCESS 권한

쿼리의 표현식으로 사용되는 각 전역 변수의 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 READ 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

쿼리에 SQL 데이터 변경 명령문이 있는 경우, 해당 명령문의 권한 요구사항이 쿼리에도 적용됩니다.

그룹 특권(PUBLIC은 예외)은 정적 SQL문 또는 DDL문에 포함되는 쿼리의 경우 점검되지 않습니다.

별칭의 경우 별칭으로 참조된 오브젝트에 대한 데이터 소스의 권한 부여 요구사항은 쿼리가 처리될 때 적용됩니다. 명령문의 권한 부여 ID는 데이터 소스에 있는 다른 권한 부여 ID로 맵핑될 수 있습니다.

쿼리 및 테이블 표현식

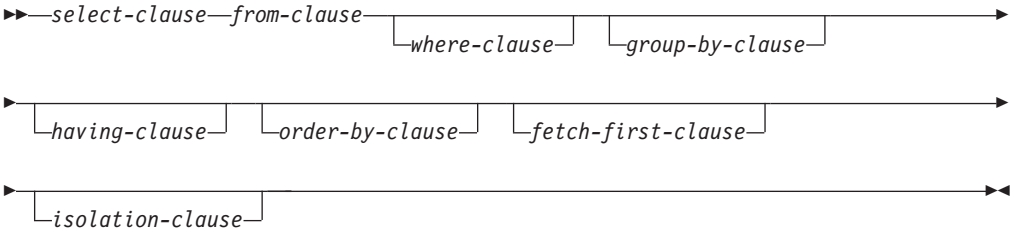
*쿼리*는 특정 SQL문의 구성요소로, (임시) 결과 테이블을 지정합니다.

*테이블 표현식*은 단순 쿼리에서 임시 결과 테이블을 작성합니다. 절은 결과 테이블을 더욱 세분화합니다. 예를 들어, 쿼리로 테이블 표현식을 사용하여 몇몇 부서에서 모든 관리자를 선택하고, 그러한 관리자가 15년 이상의 작업 경력을 가지고 있고 뉴욕 지방 사무소에 위치되도록 지정할 수 있습니다.

*공통 테이블 표현식*은 복합 쿼리 내의 임시 뷰와 유사합니다. 쿼리 내의 다른 위치에서 참조하고 뷰 대신 사용할 수 있습니다. 복합 쿼리 내에서 특정의 공통 테이블 표현식을 사용할 때마다 동일한 임시 뷰를 공유합니다.

쿼리 내에서 공통 테이블 표현식의 반복적 사용을 통해 비행기 예약 시스템, BOM(bill of materials) 생성 프로그램 및 네트워크 계획과 같은 응용프로그램을 지원할 수 있습니다.

subselect



*subselect*는 fullselect의 구성요소입니다.

subselect는 FROM절에서 식별된 테이블, 뷰 또는 별칭에서 파생되는 결과 테이블을 지정합니다. 이러한 파생은 각 조작의 결과가 다음 조작의 입력으로 사용되는 조작 시퀀스로 설명할 수 있습니다. (이것은 subselect를 설명하는 유일한 방법입니다. 파생을 수행하기 위해 사용된 방법은 이 설명과 다를 수 있습니다. subselect 부분이 확보될 올바른 결과에 대해 실제로 실행될 필요가 없는 경우 subselect 부분은 실행될 수도, 실행되지 않을 수도 있습니다.)

*subselect*에 대한 권한 부여는 ″SQL 쿼리″의 권한 부여 섹션에 설명되어 있습니다.

subselect절은 다음 시퀀스로 처리됩니다.

1. FROM절
2. WHERE절
3. GROUP BY절
4. HAVING절
5. SELECT절
6. ORDER BY절
7. FETCH FIRST절

다음 경우에 ORDER BY 또는 FETCH FIRST절이 포함되는 subselect를 지정할 수 없습니다.

- 뷰의 가장 외부 fullselect에서
- 구체화된 쿼리 테이블에서
- subselect를 괄호로 묶지 않은 경우

예를 들어, 다음은 유효하지 않습니다(SQLSTATE 428FJ).

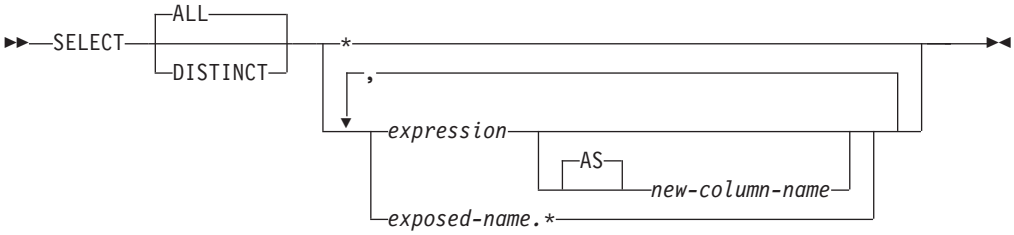
```
SELECT * FROM T1
      ORDER BY C1
   UNION
SELECT * FROM T2
      ORDER BY C1
```

다음 예는 유효합니다.

```
(SELECT * FROM T1
      ORDER BY C1)
   UNION
(SELECT * FROM T2
      ORDER BY C1)
```

주: subselect의 ORDER BY절은 쿼리에서 리턴하는 행 순서에 영향을 미치지 않습니다. ORDER BY절은 가장 외부 fullselect에 지정된 경우에만 리턴되는 행의 순서에 영향을 미칩니다.

select-clause



SELECT절은 마지막 결과 테이블, R의 컬럼을 지정합니다. 컬럼 값은 *선택 목록*을 R에 적용함으로써 생성됩니다. 선택 목록은 SELECT절에 지정된 이름이나 표현식이고 R은 subselect의 이전 조작에 대한 결과입니다. 예를 들어, SELECT, FROM 및 WHERE 절만 지정하면 R은 해당 WHERE절의 결과가 됩니다.

ALL

최종 결과 테이블의 모든 행을 보유하고 중복 항목을 제거하지 않습니다. 이는 디폴트값입니다.

DISTINCT

최종 결과 테이블에서 각 중복 행 세트 중 하나를 제외한 나머지를 모두 제거합니다. DISTINCT를 사용하는 경우 결과 테이블의 문자열 컬럼은 LOB 유형, 이러한 유형에 기초하는 구별 유형 또는 구조화된 유형이 될 수 없습니다. DISTINCT는 subselect에서 한 번 이상 사용할 수 있습니다. 여기에는 SELECT DISTINCT, 선택 목록의 집계 함수나 HAVING절에서 DISTINCT 사용 및 subselect의 서브쿼리가 포함됩니다.

첫 번째 행의 각 값이 두 번째 행의 값과 같은 경우에만 두 행이 서로 중복되는 행입니다. 중복 행을 판별하기 위해 두 개의 널(NULL) 값은 동일한 것으로 간주되며, 같은 수의 10진수 부동 소수점 표현은 동일한 것으로 간주됩니다. 예를 들어, -0은 +0과 같으며 2.0은 2.00과 같습니다. 각 10진수 부동 소수점 특수 값은 다음과 동일한 것으로 간주됩니다. -NAN은 -NAN과 같으며, -SNAN은 -SNAN과 같으며, -INFINITY는 -INFINITY와 같고, INFINITY는 INFINITY와 같으며, SNAN은 SNAN과 같으며, NAN은 NAN과 같습니다.

컬럼의 데이터 유형이 10진수 부동 소수점이고 같은 숫자의 다중 표현이 컬럼에 있으면, SELECT DISTINCT에 대해 리턴되는 특수 값이 컬럼의 표현 중 하나일 수 있습니다. 자세한 정보는 139 페이지의 『숫자 비교』의 내용을 참조하십시오.

다른 SQL 구현과의 호환성을 위해, UNIQUE를 DISTINCT의 동의어로 지정할 수 있습니다.

선택 목록 표기

* IMPLICITLY HIDDEN으로 정의되는 컬럼을 제외한, 테이블 R의 컬럼을 식별하는 이름 목록을 나타냅니다. 목록의 첫 번째 이름은 R의 첫 번째 컬럼을 나타내며, 두 번째 이름은 R의 두 번째 컬럼을 나타내며, 나머지 이름도 이와 같은 방식으로 나타냅니다.

이름 목록은 SELECT절을 포함하는 프로그램이 바인드될 때 작성됩니다. 따라서 *(별표)는 테이블 참조를 포함하는 명령문이 바인드된 후 테이블에 추가된 컬럼을 나타내지 않습니다.

expression

결과 컬럼의 값을 지정합니다. 유효한 SQL 언어 요소이지만, 일반적으로 컬럼 이름을 포함하는 표현식이 될 수 있습니다. 선택 목록에 사용된 각 컬럼 이름은 분명하게 R의 컬럼을 나타내야 합니다. 표현식의 결과 유형은 행 유형일 수 없습니다 (SQLSTATE 428H2).

new-column-name **또는 AS** *new-column-name*

결과 컬럼의 이름을 지정하거나 바꿉니다. 이름을 규정해서는 안되며 고유하지 않아도 됩니다. 컬럼 이름을 다음 번에 사용할 때 다음과 같이 제한됩니다.

- AS절에 지정된 *new-column-name*은 제공된 이름이 고유할 경우 order-by-clause에서 사용할 수 있습니다.
- 선택 목록의 AS절에 지정된 *new-column-name*은 subselect 내의 다른 절 (where-clause, group-by-clause 또는 having-clause)에서 사용할 수 없습니다.
- AS절에 지정된 *new-column-name*은 update-clause에서 사용할 수 없습니다.
- AS절에 지정된 *new-column-name*은 중첩 테이블 표현식의 fullselect 외부, 공통 테이블 표현식 및 CREATE VIEW에 알려집니다.

*name.**

IMPLICITLY HIDDEN으로 정의되는 컬럼을 제외한, *exposed-name*으로 식별되는 결과 테이블의 컬럼을 식별하는 이름 목록을 나타냅니다. *exposed-name*은 테이블 이름, 뷰 이름, 별칭 또는 상관 이름이 될 수 있으므로 FROM절에 이름이 지정된 테이블, 뷰 또는 별칭을 지정해야 합니다. 목록의 첫 번째 이름은 테이블, 뷰 또는 별칭의 첫 번째 컬럼을 나타내고, 목록의 두 번째 이름은 테이블, 뷰 또는 별칭의 두 번째 컬럼을 나타내며, 나머지 이름도 이와 같은 방식으로 나타냅니다.

이름 목록은 SELECT절을 포함하는 명령문이 바인드될 때 작성됩니다. 따라서 *(별표)는 명령문이 바인드된 후 테이블에 추가된 컬럼을 나타내지 않습니다.

SELECT 결과의 컬럼 수는 선택 목록(즉, 명령문 준비 시 작성된 목록)의 조작 양식에 있는 표현식 수와 같으며, 4K 페이지 크기를 나타내는 500이나 8K, 16K 또는 32K 페이지 크기를 나타내는 1012를 초과할 수 없습니다.

문자열 컬럼의 제한사항

선택 목록의 제한사항은 『가변 길이 문자열 사용 제한사항』을 참조하십시오.

선택 목록 적용

선택 목록을 R에 적용하면 일부 결과는 GROUP BY 또는 HAVING의 사용 여부에 따라 다릅니다. 결과는 두 개의 다른 목록으로 설명됩니다.

GROUP BY 또는 HAVING이 사용된 경우

- 선택 목록에서 사용된 표현식 *X*(집계 함수가 아님)에는 다음과 같은 GROUP BY 절이 포함되어야 합니다.
  - 각 표현식 또는 컬럼 이름이 R의 컬럼을 분명하게 식별하는 *grouping-expression*(778 페이지의 『group-by-clause』 참조) 또는
  - *X*에서 별도의 *grouping-expression*으로 참조되는 각 R 컬럼
- 선택 목록이 각 R 그룹에 적용되므로 결과에는 R에 있는 그룹 수만큼의 행이 포함됩니다. 선택 목록을 R 그룹에 적용하면 해당 그룹이 선택 목록에 있는 집계 함수 인수의 소스가 됩니다.

GROUP BY 또는 HAVING이 사용되지 않은 경우

- 선택 목록에 집계 함수가 포함되면 안되거나, 선택 목록의 각 *column-name*이 집계 함수 내에 지정되거나 상관 컬럼 참조가 되어야 합니다.
- 선택에 집계 함수가 포함되지 않으면 선택 목록이 각 R 행에 적용되고 R에 있는 행 수만큼의 행이 결과에 포함됩니다.
- 선택 목록이 집계 함수 목록이면 R이 함수 인수의 소스가 되고 선택 목록의 적용 결과는 한 행이 됩니다.

두 경우 모두 결과의 *n*번째 컬럼에는 선택 목록의 조작 양식으로 *n*번째 표현식을 적용하여 지정된 값이 포함됩니다.

결과 컬럼의 널(NULL) 속성

결과 컬럼이 다음에서 파생된 경우에는 널(NULL) 값을 사용할 수 없습니다.

- 널(NULL) 값을 허용하지 않는 컬럼
- 상수

- COUNT 또는 COUNT_BIG 함수
- 표시기 변수가 없는 호스트 변수
- 널(NULL)을 허용하는 피연산자를 포함하지 않는 스칼라 함수 또는 표현식

결과 컬럼이 다음에서 파생된 경우에는 널(NULL) 값을 사용할 수 있습니다.

- COUNT 또는 COUNT_BIG를 제외한 모든 집계 함수
- 널(NULL) 값 허용 컬럼
- 널(NULL)을 허용하는 피연산자를 포함하는 스칼라 함수 또는 표현식
- 같은 값을 포함하는 인수가 있는 NULLIF 함수
- 표시기 변수, SQL 매개변수, SQL 변수 또는 전역 변수가 있는 호스트 변수
- 선택 목록에서 해당 항목 중 적어도 하나가 널(NULL) 입력이 기능한 경우 설정 조작의 결과
- 산술 연산식 또는 산술 연산식에서 파생된 뷰 컬럼 및 DFT_SQLMATHWARN이 Yes로 설정되어 구성된 데이터베이스
- 스칼라 subselect
- 비참조 조작
- GROUPING SETS *grouping-expression*

결과 컬럼의 이름

- AS절을 지정하는 경우 결과 컬럼의 이름은 AS절에 지정된 이름입니다.
- AS절을 지정하지 않고 컬럼 목록이 참조 절에 지정된 경우, 결과 컬럼 이름은 상관 컬럼 목록에서의 해당 이름입니다.
- 상관 절에서 AS절이나 컬럼 목록이 지정되지 않고 결과 컬럼이 단일 컬럼에서 파생된 경우(함수나 연산자가 아닌), 결과 컬럼 이름은 해당 컬럼의 규정되지 않은 이름입니다.
- 상관 절에서 AS절이나 컬럼 목록이 지정되지 않고 결과 컬럼이 단일 SQL 변수나 SQL 매개변수에서 파생된 경우(함수나 연산자가 아닌), 결과 컬럼 이름은 해당 SQL 변수나 SQL 매개변수의 규정되지 않은 이름입니다.
- 상관 절에서 AS절이나 컬럼 목록이 지정되지 않고 결과 컬럼이 비참조 조작을 통해 파생된 경우 결과 컬럼 이름은 비참조 조작의 목표 컬럼에 대한 규정되지 않은 이름입니다.
- 다른 모든 결과 결럼의 이름은 지정되지 않습니다. 시스템은 이러한 컬럼에 임시 번호(문자열)를 지정합니다.

결과 컬럼의 데이터 유형

SELECT 결과의 각 컬럼은 컬럼이 파생된 표현식으로부터 데이터 유형을 획득합니다.

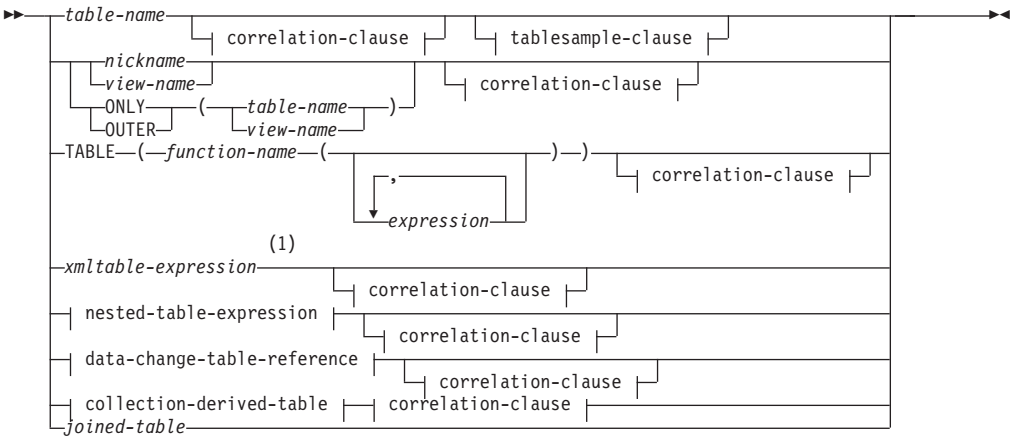
표현식 유형	결과 컬럼의 데이터 유형
숫자 컬럼의 이름	컬럼의 데이터 유형과 동일하며 DECIMAL 컬럼에 대한 정밀도와 스케일이 동일하거나, DECFLOAT 컬럼과 정밀도가 동일합니다.
상수	상수의 데이터 유형과 동일합니다.
숫자 변수의 이름	변수의 데이터 유형과 동일하며 DECIMAL 변수에 대한 정밀도와 스케일이 동일하거나, DECFLOAT 변수와 정밀도가 동일합니다.
문자열 컬럼의 이름	컬럼의 데이터 유형과 동일하며 길이 속성이 동일합니다.
문자열 변수의 이름	변수의 데이터 유형과 동일하며 길이 속성이 동일합니다. 변수의 데이터 유형이 SQL 데이터 유형과 같지 않으면(예: C에서 NUL로 종료되는 문자열) 결과 컬럼이 가변 길이의 문자열이 됩니다.
날짜 시간 컬럼의 이름	컬럼의 데이터 유형과 동일
사용자 정의된 유형 컬럼의 이름	컬럼의 데이터 유형과 동일
참조 유형 컬럼의 이름	컬럼의 데이터 유형과 동일

from-clause

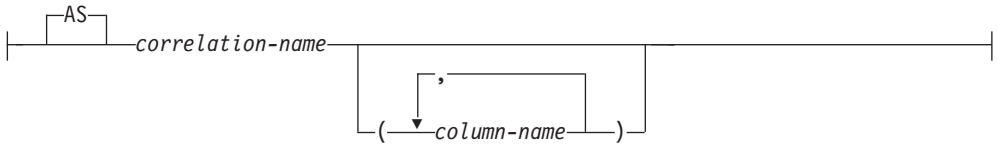
FROM절은 중간 결과 테이블을 지정합니다.

하나의 *table-reference*를 지정하는 경우, 중간 결과 테이블은 해당 *table-reference*의 결과가 됩니다. 둘 이상의 *table-reference*를 지정하면, 중간 결과 테이블이 지정한 *table-reference*에 있는 행의 가능한 모든 조합으로 구성됩니다(Cartesian 제품). 각 결과 행은 두 번째 *table-reference* 행과 병합된 첫 번째 *table-reference*이고, 두 번째 참조 행은 세 번째 테이블 참조 행과 병합되며, 나머지 행도 이와 같은 방식으로 병합됩니다. 결과의 행 수는 모든 개별 테이블 참조에 있는 행 수의 곱입니다. *table-reference*의 설명은 『table-reference』의 내용을 참조하십시오.

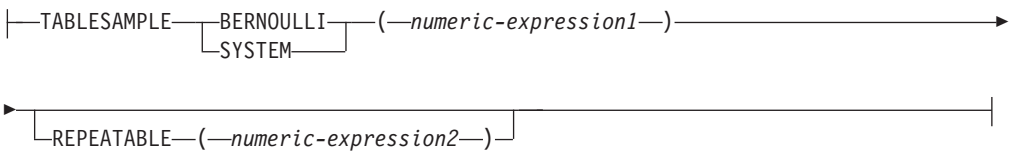
table-reference



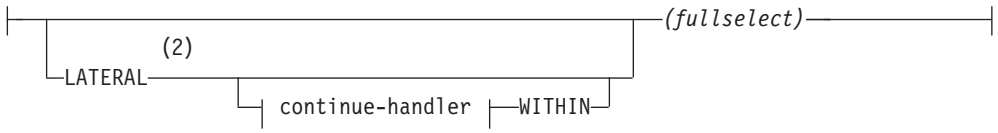
correlation-clause:



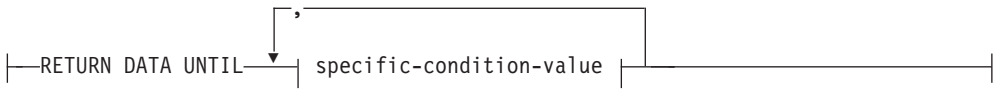
tablesample-clause:



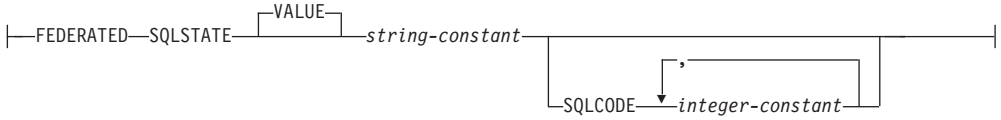
nested-table-expression:



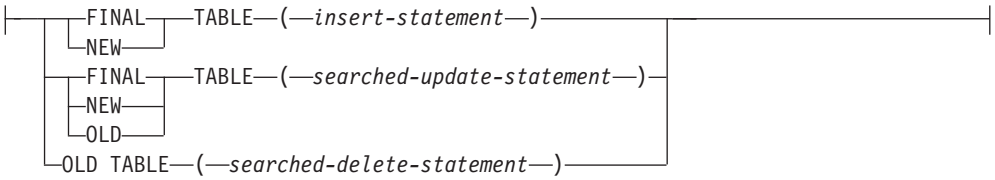
continue-handler:



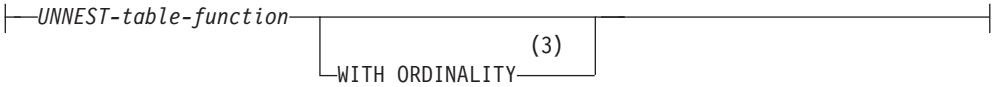
specific-condition-value:



data-change-table-reference:



콜렉션 파생 테이블:



주:

1 XMLTABLE 표현식은 테이블 참조의 일부가 될 수 있습니다. 이 경우 XMLTABLE 표현식 내의 하위 표현식은 FROM절에 있는 이전의 범위 변수의 유효 범위 안에 있습니다. 자세한 정보는『XMLTABLE』페이지의 설명을 참조하십시오.

2 LATERAL 대신 TABLE을 지정할 수 있습니다.

3 WITH ORDINALITY는 UNNEST 테이블 함수의 인수가 하나 이상의 일반 배열 변수인 경우에만 지정할 수 있으며 연관 배열 변수에는 지정할 수 없습니다 (SQLSTATE 428HT).

테이블 참조로 지정한 각 *table-name*, *view-name* 또는 *nickname*은 응용프로그램 서버에 있는 기존 테이블, 뷰 또는 별칭을 나타내거나 테이블 참조를 포함하는 fullselect 앞에 정의된 공통 테이블 표현식의 *table-name*을 나타내야 합니다. *table-name*이 유형이 지정된 테이블을 참조하는 경우 이름은 모든 서브테이블과 *table-name* 컬럼만 있는 테이블의 UNION ALL을 나타냅니다. 마찬가지로 *view-name*이 유형이 지정된 뷰를 참조하는 경우 이름은 모든 서브뷰와 *view-name* 컬럼만 있는 뷰의 UNION ALL을 나타냅니다.

ONLY(*table-name*) 또는 ONLY(*view-name*)의 사용은 올바른 서브테이블이나 서브뷰의 행이 포함되지 않음을 의미합니다. ONLY와 함께 사용된 *table-name*에 서브테이블이 없으면 ONLY(*table-name*)는 *table-name*을 지정하는 것과 같습니다. ONLY와 함께 사용된 *view-name*에 서브뷰가 없으면 ONLY(*view-name*)는 *view-name*을 지정하는 것과 같습니다.

OUTER(*table-name*) 또는 OUTER(*view-name*)의 사용은 가상 테이블을 나타냅니다. OUTER와 함께 사용된 *table-name* 또는 *view-name*에 서브테이블 또는 서브뷰가 없으면 OUTER를 지정해도 의미가 없습니다. OUTER(*table-name*)는 *table-name*에서 다음과 같이 추출됩니다.

- 컬럼에 각 서브테이블(있는 경우)에서 소개한 추가 컬럼이 뒤에 나오는 *table-name* 컬럼이 포함됩니다. 추가 컬럼은 오른쪽에서 추가되고 서브테이블 계층 구조를 좌우 이동합니다. 서브테이블에 공통 상위 테이블이 있으면 유형 작성 순서대로 이동합니다.
- 행에 *table-name*의 모든 행과 서브테이블의 모든 행이 포함됩니다. 행의 서브테이블에 없는 컬럼에는 널(NULL)이 리턴됩니다.

table-name 대신 *view-name*을 사용하고 서브테이블 대신 서브뷰를 사용하면 앞에서 언급한 사항은 OUTER(*view-name*)에도 적용됩니다.

ONLY 또는 OUTER를 사용하려면 *table-name*의 모든 서브테이블 또는 *view-name*의 모든 서브뷰에 대한 SELECT 권한이 필요합니다.

각 *함수 이름*은 테이블 참조로 지정된 인수 유형과 함께 응용프로그램 서버의 기존 테이블 함수로 분석되어야 합니다.

괄호 안의 fullselect를 *중첩 테이블 표현식*이라고 합니다.

*조인 테이블*은 하나 이상의 조인 조작 결과인 중간 결과 세트를 지정합니다. 자세한 정보는 775 페이지의 『joined-table』의 내용을 참조하십시오.

모든 테이블 참조에 대한 표시 이름은 고유해야 합니다. 표시 이름은 다음과 같습니다.

- *correlation-name*
- 뒤에 *correlation-name*이 없는 *table-name*
- 뒤에 *correlation-name*이 없는 *view-name*
- 뒤에 *correlation-name*이 없는 *nickname*
- 뒤에 *correlation-name*이 없는 *alias-name*

correlation-clause 뒤에 *function-name* 참조, *xmltable-expression*, 중첩 테이블 표현식 또는 *data-change-table-reference*가 없으면, 해당 테이블 참조에 대한 표시 이름이 없습니다.

각 *correlation-name*은 바로 앞에 있는 *table-name*, *view-name*, *nickname*, *function-name* 참조, *xmltable-expression*, 중첩된 테이블 표현식, 또는 *data-change-table-reference*의 지정자로 정의됩니다. 컬럼에 대해 규정된 참조는 표시 이름을 사용해야 합니다. 같은 테이블 이름, 뷰 또는 별칭 이름이 두 번 지정되는 경우 적어도 한 스펙 뒤에는 *correlation-name*이 있어야 합니다. *correlation-name*은 테이블 컬럼, 뷰 또는 별칭에 대한 참조를 규정하는 데 사용됩니다. *correlation-name*이 지정되면 *column-name*도 지정하여 테이블 참조의 컬럼에 이름을 지정할 수 있습니다. *correlation-clause*에 *column-name*가 포함되어 있지 않으면, 표시 이름은 다음과 같이 판별됩니다.

- *table-reference*가 *table-name*, *view-name*, *nickname*, 또는 *alias-name*인 경우의 참조 테이블의 컬럼 이름
- *table-reference*가 *function-name* 참조인 경우 CREATE FUNCTION의 RETURNS 절에서 지정된 컬럼 이름
- *table-reference*가 *xmltable-expression*인 경우 *xmltable-expression*의 COLUMNS 에서 지정된 컬럼 이름
- *table-reference*가 *nested-table-expression*인 경우 fullselect로 표시된 컬럼 이름
- *table-reference*가 *data-change-table-reference*인 경우 정의된 INCLUDE 컬럼과 함께 데이터 변경 명령문에서의 목표 테이블 컬럼 이름

일반적으로 콜렉션 파생 테이블, 테이블 함수 및 중첩 테이블 표현식은 from절에서 지정할 수 있습니다. 테이블 함수 및 중첩 테이블 표현식 또는 콜렉션 파생 테이블의 컬럼은 상관 이름을 사용하여 선택 목록 및 나머지 subselect에서 참조할 수 있습니다. 이 상관 이름의 범위는 FROM절의 다른 테이블, 뷰 또는 별칭에 대한 상관 이름과 같습니다. 중첩 테이블 표현식은 다음과 같이 사용할 수 있습니다.

- 뷰 작성을 피하기 위해 뷰 대신(일반적인 뷰 사용이 필수가 아님)
- 원하는 결과 테이블이 호스트 변수에 기반할 경우

*콜렉션 파생 테이블*은 배열 요소를 다른 행에 컬럼 값으로 변환하는 데 사용할 수 있습니다. WITH ORDINALITY가 지정된 경우 데이터 유형 INTEGER의 추가 컬럼이 추가됩니다. 이 컬럼에는 배열에서 요소의 위치가 포함됩니다. 컬럼은 correlation-clause의 컬럼에 지정된 이름을 사용하여 선택 목록 및 subselect의 나머지에서 참조될 수 있습니다. *collection-derived-table*절은 배열이 지원되는 컨텍스트에서만 사용할 수 있습니다(SQLSTATE 42887). 자세한 내용은 ″UNNEST 테이블 함수″를 참조하십시오.

fullselect에 있는 데이터 변경 명령문에서 참조되거나 변경문의 대상이 되는 중첩 테이블 표현식의 선택 목록에 사용되는 표현식은 다음과 같은 함수가 없는 경우에만 유효합니다.

- SQL 데이터를 읽거나 수정하는 함수
- 결정적이지 않은(non-deterministic) 함수
- 외부 조치를 갖는 함수
- OLAP 함수

FROM절의 데이터 변경 명령문에 있는 중첩 테이블 표현식에서 참조되거나 표현식의 대상이 되는 뷰의 경우, 좌우 대칭 뷰이거나(WITH CHECK OPTION이 지정됨) WITH CHECK OPTION 뷰의 제한사항을 충족시키는 뷰입니다.

FROM절에 있는 데이터 변경 명령문의 대상이 중첩 테이블 표현식인 경우 수정된 행이 필요하지 않고 WHERE절 술어를 재평가하지 않으며 ORDER BY 또는 FETCH FIRST 조작을 다시 실행하지 않습니다.

선택적 tablesample-clause를 사용하여 해당 *table-name*의 전체 내용이 아닌 지정된 *table-name*으로부터 이 쿼리에 대한 무작위 서브세트(샘플) 행을 얻을 수 있습니다. 이 샘플링은 *where-clause*에 지정된 술어 이외에 추가되는 것입니다. 선택적 REPEATABLE 절을 지정하지 않으면 대개 쿼리를 실행할 때마다 다른 샘플이 리턴됩니다. 그러나 샘플이 같은 행을 리턴해야 하는 샘플 *크기*에 비해 테이블이 너무 작은 경우에는 제외됩니다. 샘플 *크기*는 괄호 안의 *numeric-expression1*으로 제어되며 리턴할 테이블의 대략적인 백분율(P)을 나타냅니다. 샘플을 얻는 메소드는 TABLESAMPLE 키워드 다음에 지정되고 BERNOULLI 또는 SYSTEM이 될 수 있습니다. 두 메소드에서 술어가 행 수를 더 줄이기 전의 정확한 샘플 행 수는 쿼리를 실행할 때마다 다를 수 있지만 평균 대략적으로 테이블의 P 퍼센트가 되어야 합니다.

*테이블 이름*은 저장된 테이블이어야 합니다. 테이블은 구체화된 쿼리 테이블(MQT) 이름이 될 수 있지만, 데이터베이스 관리 프로그램이 subselect에 대한 MQT로 라우트할 것이라고 보장할 수 없기 때문에 MQT가 정의된 테이블 표현식이나 subselect가 될 수는 없습니다.

의미상, 테이블 샘플링은 술어 적용 또는 조인 수행과 같은 다른 쿼리 처리 전에 수행됩니다. 쿼리의 단일 실행(예: 중첩 루프 조인 또는 상관 서브쿼리)에서 샘플링된 테이블을 반복 액세스하면 같은 샘플이 리턴됩니다. 한 쿼리에서 둘 이상의 테이블이 샘플링될 수도 있습니다.

BERNOULLI 샘플링은 각 행을 개별적으로 간주합니다. 여기에는 샘플에서 확률이 P/100(P는 *numeric-expression1*의 값)인 행이 포함되고 확률이 1 - P/100인 행은 제외됩니다. 다른 행과는 서로 독립적입니다. 따라서 *numeric-expression1*이 값 10으로 평가된 경우 10퍼센트의 샘플을 나타내며, 확률이 0.1인 행은 포함되고 확률이 0.9인 행은 제외됩니다.

SYSTEM 샘플링을 사용하면 데이터베이스 관리 프로그램이 샘플링을 수행하는 가장 효율적인 방식을 결정할 수 있습니다. 대부분의 경우 *table-name*에 적용된 SYSTEM 샘플링은 *table-name*에서 확률이 P/100인 페이지는 샘플에 포함되고 확률이 1 - P/100인 페이지는 제외됨을 의미합니다. 포함된 각 페이지의 모든 행은 샘플에 적합합니다. 일반적으로 *table-name*의 SYSTEM 샘플링은 검색해야 하는 데이터 페이지 수가 더 적으므로 BERNOULLI 샘플링보다 더 빠르게 실행됩니다. 그러나 종종 집계 함수(예: SUM(SALES))에 대한 추정치의 정확도가 떨어집니다. 특히 *table-name*의 행이 해당 쿼리의 참조된 컬럼에서 클러스터된 경우에 그렇습니다. 옵티마이저는 특정 상황에서 SYSTEM 샘플링을 BERNOULLI 샘플링인 것처럼 수행하는 것이 더 효율적이라고 판단할 수 있습니다. 예를 들어, *table-name*의 술어가 인덱스에 의해 적용되어 샘플링 비율 P보다 훨씬 더 선택의 폭이 넓은 경우에 그렇습니다.

*numeric-expression1*은 *table-name*에서 가져올 샘플의 *크기*를 지정하며 백분율로 표현됩니다. 이 표현식은 컬럼을 포함할 수 없는 상수 산술식이어야 합니다. 표현식은 100 이하의 양수로 평가되어야 하지만 1과 0 사이의 숫자가 될 수 있습니다. 예를 들어, 값

0.01은 1/100퍼센트를 나타내고, 이는 평균적으로 10,000행 중 한 행이 샘플링됨을 의미합니다. 100으로 평가되는 *numeric-expression1*은 tablesample절을 지정하지 않은 것처럼 처리됩니다. *numeric-expression1*이 널(NULL) 값으로 평가되거나 100보다 크거나 0보다 작은 값으로 평가되면 오류가 리턴됩니다(SQLSTATE 2202H).

쿼리의 한 실행에서 다음 실행까지 샘플링을 반복하는 것이 바람직합니다. 예를 들어, 회귀 테스트를 하거나 쿼리를 ″디버깅″하는 경우에 그렇습니다. 이는 REPEATABLE 절을 지정하여 수행할 수 있습니다. REPEATABLE절에는 괄호 안의 *numeric-expression2* 스펙이 필요합니다. 이는 난수 생성 프로그램의 시드(seed)와 같은 역할을 합니다. *table-name*의 tablesample절에 REPEATABLE절을 추가하면 쿼리의 반복 실행시(*numeric-expression2*에 대해 같은 값 사용) 같은 샘플이 리턴됩니다. 물론, 데이터 자체가 갱신 또는 재구성되거나 다시 파티션되지 않았다고 가정하는 경우에 그렇습니다. 여러 쿼리에서 *table-name*의 같은 샘플이 사용되도록 전역 임시 테이블을 사용하는 것이 좋습니다. 다른 방법으로 WITH절을 통해 정의된 샘플에 대한 여러 참조를 사용하여 여러 쿼리를 하나의 쿼리로 조인할 수 있습니다.

예는 다음과 같습니다.

예 1: 감사 목적으로 Sales 테이블의 10퍼센트의 Bernoulli 샘플을 요청하십시오.

```
SELECT * FROM Sales
  TABLESAMPLE BERNOULLI(10)
```

예 2: Sales 테이블의 무작위 1퍼센트 SYSTEM 샘플을 사용하여 각 제품 범주에 대한 북동 지역의 총 판매 수익을 계산하십시오. SUM의 시맨틱은 샘플 자체에 대한 것이므로, 전체 Sales 테이블의 판매액을 추론하려면 쿼리에서 해당 SUM을 샘플링 비율(0.01)로 나누어야 합니다.

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

예 3: REPEATABLE절을 사용하여, 쿼리를 실행할 때마다 같은(무작위) 결과를 얻을 수 있도록 이전 쿼리를 수정하십시오. (괄호 안의 상수값은 임의의 값입니다.)

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1) REPEATABLE(3578231)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

테이블 함수 참조

일반적으로 테이블 함수는 인수 값과 함께 테이블 또는 뷰와 똑같은 방식으로 SELECT의 FROM절에서 참조될 수 있습니다. 그러나 몇 가지 특수한 고려사항이 적용됩니다.

- 테이블 함수 컬럼 이름

correlation-name 다음에 대체 컬럼 이름을 제공하지 않으면 테이블 함수의 컬럼 이름은 CREATE FUNCTION문의 RETURNS절에 지정된 이름입니다. 이것은 CREATE TABLE문에 정의된 테이블의 컬럼 이름과 유사합니다.

- 테이블 함수 결정

  테이블 함수 참조에 지정된 인수는 함수 이름과 함께 *함수 결정* 알고리즘에서, 사용할 정확한 함수를 결정하는 데 사용됩니다. 이것은 명령문에 사용된 다른 함수(예를 들어, 스칼라 함수)에 발생하는 것과 다른 점이 없습니다.

- 테이블 함수 인수

  스칼라 함수 인수와 마찬가지로 테이블 함수 인수는 일반적으로 유효한 SQL 표현식이 될 수 있습니다. 다음 예는 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
```

- SQL 데이터를 수정하는 테이블 함수

  MODIFIES SQL DATA 옵션으로 지정된 테이블 함수는 *select-statement*, *common-table-expression* 또는 RETURN문(SET문의 *row-fullselect*, SELECT INTO 또는 subselect임)에서 최신 테이블 참조로만 사용할 수 있습니다. FROM절에는 하나의 테이블 함수만을 사용할 수 있으며 테이블 함수 인수를 subselect에 있는 다른 모든 테이블 참조와 상관시켜야 합니다(SQLSTATE 429BL). 다음 예는 MODIFIES SQL DATA 등록 정보를 갖는 테이블 함수에 대한 유효한 구문입니다.

```
예 1:  SELECT c1
          FROM TABLE( tfmod('Jones') ) AS z

예 2:  SELECT c1
          FROM t1, t2, TABLE( tfmod(t1.c1, t2.c1) ) AS z

예 3:  SET var =
          (SELECT c1
          FROM TABLE( tfmod('Jones') ) AS z

예 4:  RETURN SELECT c1
          FROM TABLE( tfmod('Jones') ) AS z

예 5:  WITH v1(c1) AS
                        (SELECT c1
          FROM TABLE( tfmod(:hostvar1) ) AS z)
                        SELECT c1
          FROM v1, t1 WHERE v1.c1 = t1.c1
```

오류가 허용되는 nested-table-expression

nested-table-expression 내에서 발생하는 특정 오류는 허용될 수 있고, 오류를 리턴하는 대신 쿼리는 계속되어 결과를 리턴할 수 있습니다.

RETURN DATA UNTIL절을 지정하면 fullselect에서 표시된 조건을 찾아 결과 세트를 작성하기 전에 fullselect에서 리턴된 행을 생성합니다. fullselect의 부분적 결과 세트(빈 결과 세트일 수도 있음)는 *nested-table-expression*의 결과로 승인됨을 의미합니다.

FEDERATED 키워드는 조건을 제한하여 리모트 데이터 소스에서 발생하는 오류만을 처리합니다.

string-constant 길이 5를 가진 SQLSTATE 값으로 조건을 지정할 수 있습니다. 각 지정된 SQLSTATE 값에 대해 SQLCODE 값을 선택적으로 지정할 수 있습니다. SQLCODE 값은 일반적으로 여러 플랫폼에서 호환될 수 없고 SQL 표준의 일부가 될 수 없으므로 호환 가능한 응용프로그램의 경우 가능한 많은 SQLSTATE 값을 지정하십시오.

특정 조건만이 허용될 수 있습니다. 나머지 쿼리의 실행을 불허하는 오류는 허용될 수 없고 전체 쿼리에 대해 오류가 리턴됩니다. *specific-condition-value*는 특정 SQLSTATE 또는 SQLCODE 값이 지정된다 해도 데이터베이스 관리 프로그램이 실제로 허용할 수 없는 조건을 지정할 수 있고, 이 경우 오류가 리턴됩니다.

지정된 경우 다음의 SQLSTATE 값 및 SQLCODE 값에는 데이터베이스 관리 프로그램이 허용하는 미리 정의됨이 있습니다.

- SQLSTATE 08001; SQLCODEs -1336, -30080, -30081, -30082
- SQLSTATE 08004
- SQLSTATE 42501
- SQLSTATE 42704; SQLCODE -204
- SQLSTATE 42720
- SQLSTATE 28000

오류가 허용되는 *nested-table-expression*을 포함하는 쿼리 또는 뷰는 읽기 전용입니다.

오류가 허용되는 *nested-table-expression*의 fullselect는 구체화된 쿼리 테이블을 사용하여 최적화되지 않습니다.

테이블 참조의 상관 참조

상관 참조는 중첩 테이블 표현식에서 사용하거나 테이블 함수의 인수로 사용할 수 있습니다. 이러한 두 경우에 상관 참조는 기본적으로 서브쿼리 계층 구조에서 더 높은 레벨의 *테이블 참조*에 속해야 합니다. 이 계층 구조에는 이미 FROM절의 왼쪽에서 오른

쪽으로 처리할 때 결정된 테이블 참조가 포함됩니다. 중첩 테이블 표현식에서 LATERAL 키워드는 fullselect 앞에 있어야 합니다. 다음 예는 유효한 구문입니다.

```
예 1:  SELECT t.c1, z.c5
          FROM t, TABLE( tf3(t.c2) ) AS z      -- t precedes tf3
          WHERE t.c3 = z.c4;                   -- in FROM, so t.c2
                                                      -- is known

예 2:  SELECT t.c1, z.c5
          FROM t, TABLE( tf4(2 * t.c2) ) AS z  -- t precedes tf4
          WHERE t.c3 = z.c4;                   -- in FROM, so t.c2
                                                      -- is known

예 3:  SELECT d.deptno, d.deptname,
                                  empinfo.avgsal, empinfo.empcount
          FROM department d,
               LATERAL (SELECT AVG(e.salary) AS avgsal,
                               COUNT(*) AS empcount
                        FROM employee e           -- department precedes
                        WHERE e.workdept=d.deptno -- and TABLE is
                       ) AS empinfo;              -- specified, so
                                                      -- d.deptno is known
```

다음 예는 유효하지 않는 구문입니다.

```
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
                        WHERE e.workdept=d.deptno -- but TABLE is not
                       ) AS empinfo;              -- specified, so
                                                      -- d.deptno is unknown
```

데이터 변경 테이블 참조

*data-change-table-reference*절은 중간 결과 테이블을 지정합니다. 이 테이블은 절에 포함되어 있는 검색된 UPDATE, 검색된 DELETE 또는 INSERT문이 직접 변경한 행을 기초로 합니다. *data-change-table-reference*는 *select-statement*, SELECT INTO 문 또는 공통 테이블 표현식에서 사용되는 외부 fullselect의 FROM절에 필요한 *table-reference*로서만 지정될 수 있습니다. *data-change-table-reference*는 SET 변수 명령문(SQLSTATE 428FL)의 fullselect에서만 유일한 테이블 참조로 지정될 수 있습니다. 데이터 변경 명령문의 목표 테이블 또는 뷰는 쿼리에서 참조되는 테이블 또는 뷰입니다. 따라서 쿼리의 권한 부여 ID에는 해당 목표 테이블 또는 뷰에 대한 SELECT 특권이 있어야 합니다. *data-change-table-reference*절은 뷰 정의, 구체화된 쿼리 테이블 또는 FOR문에 지정될 수 없습니다(SQLSTATE 428FL).

UPDATE, DELETE 또는 INSERT문의 목표는 공통 테이블 표현식에 정의되는 임시 뷰가 될 수 없습니다(SQLSTATE 42807).

FINAL TABLE

SQL 데이터 변경 명령문에서 변경한 행 세트가 변경문 완료 시 나타날 경우, 이러한 행 세트가 중간 결과 테이블에 표시되도록 지정합니다. SQL 데이터 변경 명령문의 목표 테이블에 대한 추후 조작이 요구되는 AFTER 트리거 또는 참조 제한조건이 있는 경우 오류가 리턴됩니다(SQLSTATE 560C6). SQL 데이터 변경 명령문의 대상이 데이터 변경 유형에 해당되는 INSTEAD OF 트리거를 사용하여 정의된 뷰인 경우 오류가 리턴됩니다(SQLSTATE 428G3).

NEW TABLE

중간 결과 테이블의 행이 참조 제한조건 및 AFTER 트리거의 응용프로그램에 앞서 SQL 데이터 변경 명령문에서 변경한 행 세트를 표시하도록 지정합니다. 참조 제한조건 및 AFTER 트리거에 필요한 추가 처리로 인해 명령문 완료 시 목표 테이블에 있는 데이터는 중간 결과 테이블에 있는 데이터와 일치하지 않을 수도 있습니다.

OLD TABLE

SQL 데이터 변경문에서 변경한 행 세트가 데이터 변경문의 응용프로그램 이전에 존재했을 경우, 이러한 행 세트가 중간 결과 테이블에 표시되도록 지정합니다.

(searched-update-statement)

검색된 UPDATE문을 지정합니다. UPDATE문에 있는 WHERE절 또는 SET절에는 UPDATE문 외부의 컬럼과 상관된 참조가 있을 수 없습니다.

(searched-delete-statement)

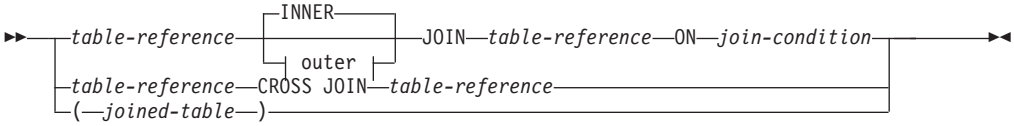
검색된 DELETE문을 지정합니다. DELETE문에 있는 WHERE절에는 DELETE문 외부의 컬럼과 상관된 참조가 있을 수 없습니다.

(insert-statement)

INSERT문을 지정합니다. INSERT문의 fullselect에는 INSERT문 fullselect 외부의 컬럼과 상관된 참조가 있을 수 없습니다.

*data-change-table-reference*를 사용하는 중간 결과 테이블의 내용은 커서 열기 후 결정됩니다. 중간 결과 테이블에는 지정한 목표 테이블이나 뷰에 있는 모든 컬럼을 포함하여 조작된 모든 행이 있습니다. 목표 테이블이나 뷰의 컬럼 이름을 사용하여 SQL 데이터 변경 명령문에 필요한 목표 테이블이나 뷰의 모든 컬럼을 액세스할 수 있습니다. 데이터 변경 명령문에 INCLUDE절을 지정한 경우 해당 컬럼이 중간 결과 테이블에 추가됩니다.

joined-table



outer:

*joined table*은 내부 조인 또는 외부 조인의 결과인 중간 결과 테이블을 지정합니다. 테이블은 조인 연산자 CROSS, INNER, LEFT OUTER, RIGHT OUTER 또는 FULL OUTER 중 하나를 피연산자에 적용함으로써 파생됩니다.

교차 조인은 테이블의 교차 곱을 나타내며, 왼쪽 테이블의 각 행을 오른쪽의 모든 행과 조합합니다. 내부 조인은 교차 곱으로 간주할 수 있으며, 조인 조건이 참인 행만 보관합니다. 결과 테이블의 경우 조인 테이블 중 하나 또는 둘 다에서 행이 누락될 수도 있습니다. 외부 조인에는 내부 조인이 포함되고 이러한 누락된 행이 보존됩니다. 세 가지 유형의 외부 조인이 있습니다.

- ***왼쪽 외부 조인***에는 내부 조인에서 누락된 왼쪽 테이블의 행이 포함됩니다.
- ***오른쪽 외부 조인***에는 내부 조인에서 누락된 오른쪽 테이블의 행이 포함됩니다.
- ***완전 외부 조인***에는 내부 조인에서 누락된 왼쪽 및 오른쪽 테이블의 행이 포함됩니다.

조인 연산자를 지정하지 않으면 INNER가 적용됩니다. 다중 조인의 수행 순서가 결과에 영향을 미칠 수 있습니다. 조인은 다른 조인 내에 중첩될 수 있습니다. 일반적으로 조인 처리 순서는 왼쪽에서 오른쪽이지만 필수 조인 조건의 위치에 기초합니다. 중첩 조인의 순서를 더 쉽게 알아볼 수 있도록 괄호를 사용하는 것이 좋습니다. 예를 들어, 다음과 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
```

위 예는 다음 예와 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
```

조인 테이블은 SELECT문의 형식이 사용되는 컨텍스트에서 사용할 수 있습니다. SELECT문에 조인 테이블이 포함되면 뷰 또는 커서가 읽기 전용입니다.

*join-condition*은 다음의 경우를 제외하고 *search-condition*입니다.

- 서브쿼리, 스칼라 등을 포함할 수 없는 경우
- 참조 값이 오브젝트 ID 컬럼 이외의 값일 때, 비참조 조작 및 DEREF 함수를 포함할 수 없는 경우
- SQL 함수를 포함할 수 없는 경우

- *join-condition* 표현식에서 참조된 컬럼이 동일한 조인된 테이블 절의 범위에 있는 연관된 조인의 피연산자 테이블 중 하나의 컬럼이어야 하는 경우
- 완전 외부 조인의 *join-condition* 표현식에서 참조된 함수가 결정적이어야 하고 외부 조치가 없어야 하는 경우
- XMLQUERY 또는 XMLEXISTS 표현식을 포함할 수 없는 경우

조인 조건이 이러한 규칙을 따르지 않으면 오류가 발생합니다(SQLSTATE 42972).

컬럼 참조는 컬럼 이름 규정자의 결정 규칙을 사용하여 결정됩니다. 술어에 적용되는 동일한 규칙이 조인 조건에 적용됩니다.

조인 조작

*join-condition*은 T1과 T2를 쌍으로 지정합니다. 여기서, T1과 T2는 *join-condition*의 JOIN 연산자에서 사용되는 왼쪽 및 오른쪽 피연산자 테이블입니다. T1과 T2의 가능한 모든 행 조합에서 *join-condition*이 적용되면 T1의 한 행이 T2의 한 행과 쌍을 이룹니다. T1의 행이 T2의 행과 조인되면 결과 행은 T2의 해당 행 값과 병합된 T1의 해당 행 값으로 구성됩니다. 실행에는 널(NULL) 행의 생성이 포함될 수 있습니다. 테이블의 널(NULL) 행은 컬럼의 널(NULL)값 허용 여부와 상관없이 각 테이블 컬럼의 널(NULL)값으로 구성됩니다.

다음은 조인 조작의 결과를 요약한 것입니다.

- T1 CROSS JOIN T2의 결과는 행에서 가능한 모든 쌍으로 구성됩니다.
- T1 INNER JOIN T2의 결과는 조인 조건이 적용되고 쌍을 이루는 행으로 구성됩니다.
- T1 LEFT OUTER JOIN T2의 결과는 조인 조건이 적용되고 쌍을 이루는 행 및 쌍을 이루지 않는 T1의 각 행에 대해 T2의 널(NULL) 행과 해당 행의 병합으로 구성됩니다. T2에서 파생된 모든 컬럼은 널(NULL) 값을 허용합니다.
- T1 RIGHT OUTER JOIN T2의 결과는 조인 조건이 적용되고 쌍을 이루는 행 및 쌍을 이루지 않는 T2의 각 행에 대해 T1의 널(NULL) 행과 해당 행의 병합으로 구성됩니다. T1에서 파생된 모든 컬럼은 널(NULL) 값을 허용합니다.
- T1 FULL OUTER JOIN T2의 결과는 쌍을 이루는 행과 쌍을 이루지 않는 T2의 각 행에 대해 T1의 널(NULL) 행과 해당 행의 병합 및 쌍을 이루지 않는 T1의 각 행에 대해 T2의 널(NULL)행과 해당 행의 병합으로 구성됩니다. T1 및 T2에서 파생된 모든 컬럼은 널(NULL) 값을 허용합니다.

where-clause

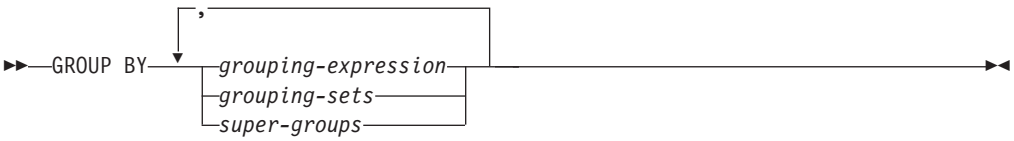
WHERE절은 *search-condition*이 적용되는 R의 행으로 구성되는 중간 결과 테이블을 지정합니다. R은 subselect의 FROM절에 대한 결과입니다.

*search-condition*은 다음 규칙을 따라야 합니다.

- 각 *column-name*이 R의 컬럼을 분명하게 나타내거나 상관 참조이어야 합니다. *column-name*이 외부 subselect에서 *테이블* 참조의 컬럼을 나타내면 상관 참조입니다.
- HAVING절의 서브쿼리에서 WHERE절을 지정하지 않고 함수 인수가 그룹에 대한 상관 참조가 아니면 집계 함수를 지정해서는 안됩니다.

*search-condition*의 서브쿼리는 R의 각 행에 대해 효과적으로 실행되고 결과는 주어진 R의 행에 *search-condition*을 적용할 때 사용됩니다. 실제로 서브쿼리는 상관 참조가 포함되는 경우에만 R의 각 행에 대해 실행됩니다. 그리고 상관된 참조가 없는 서브쿼리는 한 번만 실행되는 반면, 상관된 참조가 있는 서브쿼리는 각 행에 한 번씩 실행되어야 할 수도 있습니다.

group-by-clause



GROUP BY절은 R의 행 그룹화로 구성되는 중간 결과 테이블을 지정합니다. R은 subselect의 이전 절에 대한 결과입니다.

6GROUP BY절의 가장 단순한 형식에는 *grouping expression*이 포함됩니다. 그룹화 표현식은 R의 그룹화를 정의할 때 사용되는 *expression*입니다. 그룹화 표현식에 포함되는 각 표현식이나 *column name*은 R의 컬럼을 명확하게 식별해야 합니다(SQLSTATE 42702 또는 42703). 그룹화 표현식에는 스칼라 fullselect, XMLQUERY 또는 XMLEXISTS 표현식(SQLSTATE 42822) 또는 가변적이거나 외부 조치가 있는 표현식이나 함수가 포함될 수 없습니다(SQLSTATE 42845).

주: 명시적 컬럼 참조를 포함하지 않는 다음 표현식을 *grouping-expression*에서 사용하여 R의 컬럼을 식별할 수 있습니다.

- *table-designator*에 대한 행 변경 시간소인
- *table-designator*에 대한 행 변경 토큰
- RID_BIT 또는 RID 스칼라 함수

더 복잡한 형식의 GROUP BY절에는 *grouping-sets* 및 *super-groups*가 포함됩니다. 이러한 형식에 대한 설명은 각각 779 페이지의 『grouping-sets』 및 780 페이지의 『super-groups』의 내용을 참조하십시오.

GROUP BY의 결과는 행 그룹 세트입니다. 이 결과의 각 행은 *그룹화 표현식*이 동일한 행 세트를 나타냅니다. 그룹화의 경우, *grouping-expression*의 모든 널(NULL) 값은 동일한 것으로 간주됩니다.

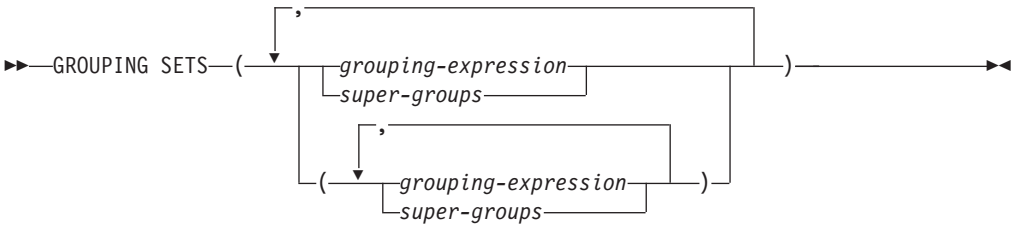
*grouping-expression*에 10진수 부동 소수점 컬럼이 포함되고 동일 수의 다중 표현이 이들 컬럼에 존재하는 경우, 리턴되는 수가 숫자의 표현일 수 있습니다.

*grouping-expression*은 HAVING절의 검색 조건, SELECT절의 표현식 또는 ORDER BY절의 *sort-key-expression*에서 사용할 수 있습니다(자세한 내용은 786 페이지의 『order-by-clause』 참조). 모든 경우 참조는 각 그룹에 대해 하나의 값만 지정합니다. 예를 들어, *grouping-expression*이 *col1+col2*인 경우, 선택 목록에서 허용되는 표현식은 *col1+col2+3*입니다. 해당 표현식이 같은 순서로 평가되도록 하는 괄호를 사용하지 않으면, 표현식의 연관성 규칙이 유사한 표현식인 *3+col1+col2*를 허용하지 않습니다. 따라서 선택 목록에서 *3+(col1+col2)*도 허용됩니다. 병합 연산자를 사용하는 경우, *grouping-expression*은 선택 목록에 지정된 것과 똑같은 방식으로 사용해야 합니다.

*grouping-expression*에 뒤 공백이 있는 가변 길이 문자열이 포함되는 경우, 그룹에 있는 값의 뒤 공백 수가 다를 수 있고 길이가 다를 수도 있습니다. 이 경우, *grouping-expression*에 대한 참조는 계속 각 그룹에 대해 하나의 값만 지정하지만, 그룹 값은 사용 가능한 값 세트에서 임의적으로 선택됩니다. 그러므로 결과 값의 실제 길이는 예측할 수 없습니다.

설명한 대로 GROUP BY절이 SELECT절에 지정된 컬럼을 표현식(스칼라 fullselect, 가변 또는 외부 조치 함수)으로 직접 참조할 수 없는 경우가 있습니다. 이러한 표현식을 사용하여 그룹화하려면, 중첩 테이블 표현식 또는 공통 테이블 표현식을 사용하여 먼저 표현식을 결과 컬럼으로 결과 테이블에 제공하십시오. 중첩 테이블 표현식을 사용한 예는 예 A9를 참조하십시오.

grouping-sets



*grouping-sets*를 지정하면 단일 명령문에서 여러 개의 그룹화 절을 사용할 수 있습니다. 이는 둘 이상의 행 그룹을 단일 결과 세트로 통합하는 것으로 간주할 수 있습니다. 이것은 논리적으로 여러 개의 subselect를 하나의 그룹화 세트에 해당하는 각 subselect의 group by절과 통합하는 것과 같습니다. 그룹화 세트는 단일 요소 또는 괄호로 분리되

는 요소 목록으로 구성될 수 있습니다. 이러한 요소는 그룹화 표현식 또는 슈퍼 그룹입니다. *grouping-sets*를 사용하면 기본 테이블에 대해 한 과정으로 그룹을 계산할 수 있습니다.

*grouping-sets*를 지정하면 단순 *grouping-expression*을 사용하거나 더 복잡한 형식의 *super-groups*를 사용할 수 있습니다. *super-groups*의 설명은 『super-groups』에서 참조하십시오.

그룹화 세트는 GROUP BY 조작의 기반이 되는 빌딩 블록입니다. 단일 컬럼이 있는 단순 GROUP BY절은 하나의 요소가 있는 그룹화 세트로 간주할 수 있습니다. 예를 들면, 다음과 같습니다.

```
GROUP BY a
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a))
```

및

```
GROUP BY a,b,c
```

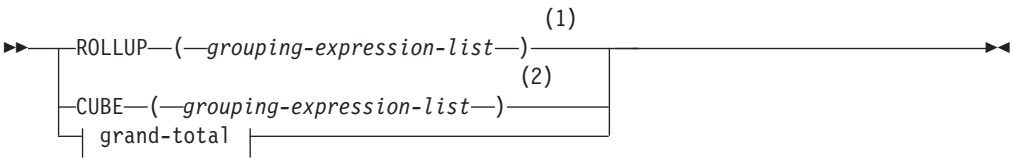
아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b,c))
```

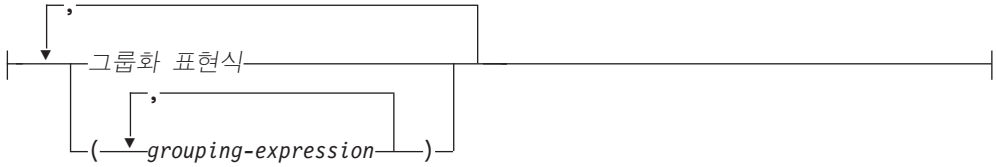
그룹화 세트에서 제외된 subselect의 선택 목록의 비집계 컬럼은 해당 그룹 세트에 대해 생성되는 각 행의 이러한 컬럼에 대해 널(NULL)을 리턴합니다. 이는 이러한 컬럼에 대한 값을 고려하지 않고 집계가 수행되었음을 나타냅니다.

예 C2에서 예 C7까지 그룹화 세트의 사용을 보여 줍니다.

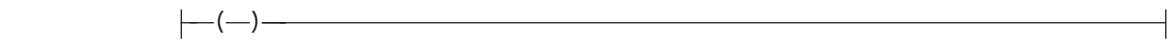
super-groups



grouping-expression-list:



grand-total:



주:

1 group-by절에서 단독으로 사용되는 경우 대체 스펙은 그룹화 표현식 목록 WITH ROLLUP입니다.

2 group-by-clause에서 단독으로 사용되는 경우 대체 스펙은 grouping-expression-list WITH CUBE입니다.

ROLLUP (*grouping-expression-list*)

*ROLLUP grouping*은 『일반』 그룹화 행 이외에 *부분합계(subtotal)* 행을 포함하는 결과 세트를 생성하는 GROUP BY절의 확장입니다. *sub-total* 행은 그룹화된 행을 얻기 위해 사용한 것과 동일한 집계 함수를 적용함으로써 값이 파생된 추가 집계를 포함하는 『과집계』 행입니다. 이러한 행은 가장 일반적으로 사용되므로 부분합(subtotal) 행이라고 합니다. 그러나 집계를 위해 집계 함수를 사용할 수 있습니다. 예를 들어, MAX 및 AVG가 예 C8에서 사용됩니다. 슈퍼 그룹으로 행이 생성되었는지 여부를 나타내는 데 GROUPING 집계 함수를 사용할 수 있습니다.

ROLLUP 그룹화는 일련의 *grouping-sets*입니다. *n*개 요소를 사용한 ROLLUP의 일반적인 스펙의 경우

```
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*개 요소가 *n*+1 그룹화 세트로 변환됨에 주의하십시오. 또한 *grouping-expressions*의 지정 순서가 ROLLUP에 중요하다는 점을 염두에 두십시오. 예를 들면, 다음과 같습니다.

```
GROUP BY ROLLUP(a,b)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b)
                                        (a)
                       () )
```

반면 다음은

```
GROUP BY ROLLUP(b,a)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((b,a)
                                        (b)
                       () )
```

ORDER BY절은 결과 세트 행의 순서를 보장하는 유일한 방법입니다. 예 C3은 ROLLUP의 사용을 보여 줍니다.

CUBE (*grouping-expression-list*)

*CUBE grouping*은 ROLLUP 집계의 모든 행 및 ″cross-tabulation″ 행을 포함하는, 결과 세트를 생성하는 GROUP BY절의 확장입니다. *Cross-tabulation* 행은 부분합계(subtotal)가 있는 집계의 일부가 아닌 추가 ″슈퍼 집계″ 행입니다. 슈퍼 그룹으로 행이 생성되었는지 여부를 나타내는 데 GROUPING 집계 함수를 사용할 수 있습니다.

ROLLUP와 마찬가지로 CUBE 그룹화도 일련의 *grouping-sets*로 간주할 수 있습니다. CUBE의 경우 큐브 *grouping-expression-list*의 모든 변경은 총 합계와 함께 계산됩니다. 그러므로, CUBE의 *n*개 요소는 2**n(2의 *n* 제곱) *그룹화 세트*로 변환됩니다. 예를 들어 다음의 스펙을 참조하십시오.

```
GROUP BY CUBE(a,b,c)
```

다음과 같습니다.

```
GROUP BY GROUPING SETS((a,b,c)
                                        (a,b)
                                        (a,c)
                                        (b,c)
                                        (a)
                                        (b)
                                        (c)
                       () )
```

CUBE의 3개 요소가 8개의 그룹화 세트로 변환됨에 유의하십시오.

CUBE에서 요소의 지정 순서는 중요하지 않습니다.‘CUBE(DayOfYear, Sales_Person)’ 및 ‘CUBE(Sales_Person, DayOfYear)’는 같은 결과 세트를 생성합니다. ‘같다’라는 의미는 순서가 아닌 결과 세트의 내용에 적용됩니다. ORDER BY절은 결과 세트 행의 순서를 보장하는 유일한 방법입니다. 예 C4는 CUBE의 사용을 나타냅니다.

grouping-expression-list

*grouping-expression-list*는 CUBE 또는 ROLLUP절 내에서 사용되어 CUBE 또는 ROLLUP 조작의 요소 수를 정의합니다. 이 목록은 여러 *grouping-expression*에서 요소를 분리하는 괄호를 사용하여 제어됩니다.

*grouping-expression*에 대한 규칙은 778 페이지의 『group-by-clause』에서 설명됩니다. 예를 들어, 쿼리가 County가 아닌 Province 내에서 City의 ROLLUP에 대한 총 비용을 리턴한다고 가정해 보십니다. 그러나 다음 절의 경우:

```
GROUP BY ROLLUP(Province, County, City)
```

County에 대해 원하지 않는 부분합(subtotal) 행을 리턴합니다. 다음 절에서:

```
GROUP BY ROLLUP(Province, (County, City))
```

(County, City) 복합 유형은 ROLLUP에서 하나의 요소를 형성하므로 이 절을 사용하는 쿼리는 원하는 결과를 생성하게 됩니다. 즉, 두 요소의 ROLLUP 경우:

```
GROUP BY ROLLUP(Province, (County, City))
```

다음을 생성:

```
GROUP BY GROUPING SETS((Province, County, City)
                                                    (Province)
                                 () )
```

및 3개의 요소 롤업(ROLLUP) 생성:

```
GROUP BY GROUPING SETS((Province, County, City)
                                                (Province, County)
                                                (Province)
                             () )
```

예 C2에서도 복합 컬럼 값을 사용합니다.

grand-total

CUBE 및 ROLLUP은 총(총 합계) 집계 행을 리턴합니다. 이 행은 GROUPING SET절 내에서 빈 괄호를 사용하여 별도로 지정할 수 있습니다. 또한 쿼리 결과에 영향을 미치지는 않지만 GROUP BY절에서 직접 지정할 수도 있습니다. 예 C4는 총 합계 구문을 사용합니다.

그룹화 세트 조인

GROUP BY절의 유형을 조인할 때 사용할 수 있습니다. 단순 *grouping-expression* 필드가 다른 그룹과 조인되면, 필드가 결과 *grouping sets*의 시작 부분에 ″추가″됩니다. ROLLUP 또는 CUBE 표현식을 조인하면, 나머지 표현식에서 ″멀티티어″처럼 수행되어 ROLLUP 또는 CUBE의 정의에 따라 추가 그룹화 세트 항목을 형성합니다.

예를 들어, *grouping-expression* 요소를 조인하면 다음은,

```
GROUP BY a, ROLLUP(b,c)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b,c)
                                          (a,b)
                          (a) )
```

또는 다음은,

```
GROUP BY a, b, ROLLUP(c,d)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b) )
```

ROLLUP 요소를 조인하면 다음은,

```
GROUP BY ROLLUP(a), ROLLUP(b,c)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a)
                       (b,c)
                       (b)
                       () )
```

또는 다음은,

```
GROUP BY ROLLUP(a), CUBE(b,c)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (a)
                       (b,c)
                       (b)
                       (c)
                       () )
```

*CUBE*와 *ROLLUP* 요소를 조인하면 다음은,

```
GROUP BY CUBE(a,b), ROLLUP(c,d)
```

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)
                       () )
```

단순 *grouping-expression*과 마찬가지로 그룹화 세트를 조인해도 각 그룹화 세트 내의 중복 항목이 제거됩니다. 예를 들어 다음은,

```
GROUP BY a, ROLLUP(a,b)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((a,b)
                       (a) )
```

그룹화 세트를 조인하는 더 완전한 예는 전체 CUBE 집계에 대해 리턴될 특정 행을 제거하는 결과 세트를 구성하는 것입니다.

예를 들어, 다음 GROUP BY절에 주의를 기울여 보십시오.

```
GROUP BY Region,
         ROLLUP(Sales_Person, WEEK(Sales_Date)),
         CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

GROUP BY절의 바로 오른쪽에 나열된 컬럼은 단지 그룹화만 되고, ROLLUP 뒤의 괄호 안에 있는 컬럼은 롤업되며 CUBE 뒤의 괄호 안에 있는 컬럼은 큐브됩니다. 따라서 위의 절로 인해 YEAR 내에서 MONTH의 큐브가 생성된 후 Region 집계의 Sales_Person에 있는 WEEK 내에서 롤업됩니다. 이로 인해 Region, Sales_Person 또는 WEEK(Sales_Date)에 대한 총 합계 행이나 cross-tabulation 행은 생성되지 않으므로 다음 절보다 더 적은 수의 행이 생성됩니다.

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                                       YEAR(Sales_Date), MONTH(Sales_Date) )
```

having-clause

HAVING절은 *search-condition*이 적용되는 R의 그룹으로 구성되는 중간 결과 테이블을 지정합니다. R은 subselect의 이전 절에 대한 결과입니다. 이 절이 GROUP BY가 아니면 R이 그룹화 컬럼이 없는 단일 그룹인 것으로 간주됩니다.

검색 조건에서 각 *column-name*은 다음 중 하나를 수행해야 합니다.

- R의 그룹화 컬럼을 명확하게 나타내야 합니다.
- 집계 함수 내에 지정해야 합니다.
- 상관 참조이어야 합니다. *column-name*이 외부 subselect에서 *테이블 참조*의 컬럼을 나타내면 상관 참조입니다.

검색 조건이 적용되는 R 그룹은 인수가 상관 참조인 함수를 제외하고 검색 조건의 각 집계 함수에 대해 인수를 제공합니다.

검색 조건에 서브쿼리가 포함되는 경우, 서브쿼리는 검색 조건이 R 그룹에 적용될 때마다 실행되고 그 결과는 검색 조건 적용 시 사용된다고 볼 수 있습니다. 실제로 서브쿼리는 상관 참조가 포함되는 경우에만 각 그룹에 대해 실행됩니다. 이 차이점에 대한 설명은 예 A6 및 예 A7을 참조하십시오.

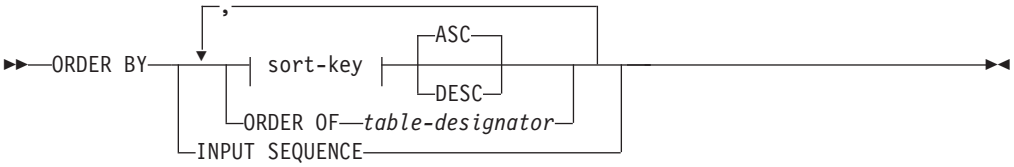
R 그룹의 상관 참조는 그룹화 컬럼을 나타내거나 집계 함수 내에 포함되어야 합니다.

HAVING을 GROUP BY없이 사용하면, 집계 함수, 상관 컬럼 참조, 전역 변수, 호스트 변수, 리터럴, 특수 레지스터, SQL 변수 또는 SQL 매개변수에 대한 인수인 경우, 선택 목록은 컬럼 이름만을 포함할 수 있습니다.

주: 집계 함수 내에 포함되어 있으면 다음 표현식은 HAVING 절에 지정할 수 있습니다(SQLSTATE 42803).

- *table-designator*에 대한 행 변경 시간소인
- *table-designator*에 대한 행 변경 토큰
- RID_BIT 또는 RID 스칼라 함수

order-by-clause



sort-key:

ORDER BY절은 결과 테이블 행의 순서를 지정합니다. 단일 정렬 스펙(지시문이 연관된 하나의 *sort-key*)을 지정하면 정렬 스펙의 값에 따라 행의 순서가 지정됩니다. 둘 이상의 정렬 스펙을 지정하는 경우 첫 번째 지정된 정렬 스펙의 값에 따라 행의 순서가 지정된 후, 두 번째 지정된 정렬 스펙의 값에 따라 다시 정렬됩니다. 각 *sort-key*에는 CLOB, DBCLOB, BLOB, 이러한 유형에 기초하는 구별 유형 또는 구조화된 유형이 포함될 수 없습니다(SQLSTATE 42907).

선택 목록에서 이름이 지정된 컬럼은 *simple-integer* 또는 *simple-column-name*인 *sort-key*로 식별할 수 있습니다. 선택 목록에서 이름이 지정되지 않은 컬럼은 *simple-integer*로 식별하거나 일부 경우에는 선택 목록의 표현식과 일치하는 *sort-key-expression*으로 식별해야 합니다(*sort-key-expression*의 세부사항 참조). AS절을 지정하지 않고 컬럼이 상수, 연산자가 있는 표현식 또는 함수에서 파생된 경우에는 컬럼 이름이 지정되지 않습니다.

순서는 비교 규칙에 따라 정해집니다. ORDER BY 절에 10진수 부동 소수점 컬럼이 포함되어 있고 동일 수의 다중 표현이 이들 컬럼에 존재하는 경우, 같은 수의 다중 표현의 순서가 규정되어 있지 않습니다. 널(NULL) 값은 다른 모든 값보다 높습니다. ORDER BY절이 행의 순서를 완전히 지정하지 않으면 식별된 모든 컬럼에 대한 중복 값이 있는 행이 임의의 순서대로 표시됩니다.

simple-column-name

대부분 결과 테이블의 컬럼을 나타냅니다. 이 때 *simple-column-name*은 선택 목록에서 이름이 지정된 컬럼의 컬럼 이름이어야 합니다.

*simple-column-name*은 쿼리가 subselect인 경우 FROM절에 지정된 테이블, 뷰 또는 중첩 테이블의 컬럼 이름도 나타낼 수 있습니다. 내재적으로 숨김으로 정의된 컬럼을 포함합니다. subselect가 다음을 수행하는 경우에 오류가 발생합니다.

- SELECT절에서 DISTINCT를 지정합니다(SQLSTATE 42822).
- 그룹화된 결과를 생성하지만 *simple-column-name*이 *grouping-expression*이 아닙니다(SQLSTATE 42803).

결과 순서에 사용되는 컬럼 판별은 아래 『정렬 키의 컬럼 이름』에서 설명됩니다.

simple-integer

0보다 크고 결과 테이블에 있는 컬럼 수보다 작아야 합니다(SQLSTATE 42805). 정수 *n*은 결과 테이블에서 *n*번째 컬럼을 나타냅니다.

sort-key-expression

단지 컬럼 이름 또는 부호가 없는 정수 상수가 아닌 표현식입니다. 정렬이 적용되는 쿼리는 이러한 형식의 정렬 키를 사용할 *subselect*이어야 합니다. *sort-key-expression*에는 상관 스칼라 fullselect(SQLSTATE 42703), XMLQUERY 또는 XMLEXISTS 표현식(SQLSTATE 42822) 또는 외부 조치가 있는 함수가 포함될 수 없습니다(SQLSTATE 42845).

sort-key-expression 내의 *column-name*은 『정렬 키의 컬럼 이름』 아래에서 설명된 규칙을 따라야 합니다.

다음과 같은 특별한 경우에 지정할 수 있는 표현식은 더 제한됩니다.

- DISTINCT가 subselect의 SELECT절에 지정되었습니다(SQLSTATE 42822).

  정렬 키 표현식은 subselect의 선택 목록에 있는 표현식과 정확히 일치해야 합니다(스칼라-fullselect는 일치되지 않음).

- subselect가 그룹화되었습니다(SQLSTATE 42803).

  정렬 키 표현식은 다음과 같을 수 있습니다.

  – subselect의 선택 목록에 있는 표현식이 될 수 있습니다.

  – subselect의 GROUP BY절에 있는 *grouping-expression*을 포함할 수 있습니다.

  – 집계 함수, 상수 또는 호스트 변수를 포함할 수 있습니다.

ASC

컬럼의 값을 오름차순으로 사용합니다. 이는 디폴트값입니다.

DESC

컬럼의 값을 내림차순으로 사용합니다.

ORDER OF *table-designator*

*table-designator*에 사용된 동일한 순서가 subselect의 결과 테이블에 적용되도록 지정합니다. 이 절을 지정하는 *table-designator*와 일치하는 테이블 참조가 subselect의 FROM절에 있어야 합니다(SQLSTATE 42703). 지정된 *table-designator*에 해당하는 subselect(또는 fullselect)에는 데이터에 따라 달라지는 ORDER BY절이 포함되어야 합니다(SQLSTATE 428FI). 적용되는 순서는 중첩된 subselect(또는 fullselect)의 ORDER BY절의 컬럼이 외부 subselect(또는 fullselect)에 포함되고 이러한 컬럼이 ORDER OF절 대신에 지정된 경우와 같습니다.

이 형식은 fullselect(변질적 형식의 fullselect 이외에)에서 허용되지 않음에 유의하십시오. 예를 들어, 다음은 유효하지 않습니다.

```
(SELECT C1 FROM T1
      ORDER BY C1)
   UNION
SELECT C1 FROM T2
      ORDER BY ORDER OF T1
```

다음 예는 유효합니다.

```
SELECT C1 FROM
   (SELECT C1 FROM T1
         UNION
    SELECT C1 FROM T2
        ORDER BY C1 ) AS UTABLE
ORDER BY ORDER OF UTABLE
```

INPUT SEQUENCE

INSERT문의 경우 결과 테이블이 요청된 데이터 행의 입력 순서를 반영하도록 지정합니다. INSERT문을 FROM절에 사용하는 경우에만 INPUT SEQUENCE 정렬을 지정할 수 있습니다(SQLSTATE 428G4). 765 페이지의 『table-reference』의 내용을 참조하십시오. INPUT SEQUENCE를 지정하고 입력 데이터를 정렬하지 않은 경우 INPUT SEQUENCE절은 무시됩니다.

주

- **정렬 키의 컬럼 이름**:
  - 컬럼 이름이 규정됩니다.

    쿼리는 *subselect*이어야 합니다(SQLSTATE 42877). 컬럼 이름은 subselect의 FROM절에서 일부 테이블, 뷰 또는 중첩 테이블의 컬럼을 명확하게 나타내야 합니다(SQLSTATE 42702). 컬럼 값은 정렬 스펙의 값을 계산하는 데 사용됩니다.
  - 컬럼 이름이 규정되지 않았습니다.
    - 쿼리는 subselect입니다.

      컬럼 이름이 결과 테이블에 있는 둘 이상의 컬럼 이름과 같은 경우, 컬럼 이름은 순서 지정 subselect의 FROM절에서 일부 테이블, 뷰 또는 중첩 테이블의 컬럼을 명확하게 나타내야 합니다(SQLSTATE 42702). 컬럼 이름이 하나의 컬

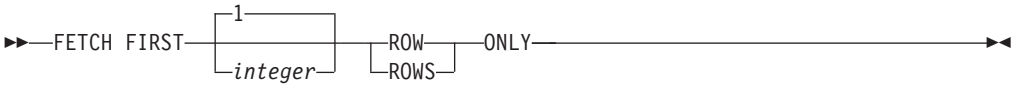
럼과 같으면 해당 컬럼이 정렬 스펙의 값을 계산하는 데 사용됩니다. 컬럼 이름이 결과 테이블의 컬럼과 다르면 select문에서 fullselect의 FROM절에 있는 일부 테이블, 뷰 또는 중첩 테이블의 컬럼을 명확하게 나타내야 합니다 (SQLSTATE 42702).

- 쿼리가 subselect가 아닙니다. (쿼리에 union, except 또는 intersect와 같은 설정 조작이 포함됩니다.)

  컬럼 이름이 결과 테이블에 있는 둘 이상의 컬럼 이름과 같아서는 안됩니다 (SQLSTATE 42702). 컬럼 이름은 결과 테이블의 한 컬럼과만 같아야 하며 (SQLSTATE 42707) 이 컬럼은 정렬 스펙의 값을 계산하는 데 사용됩니다.

- **제한사항**: 선택 목록에 컬럼이 없는 *sort-key-expression* 또는 *simple-column-name*을 사용하면 정렬하기 위해 사용할 임시 테이블에 컬럼이나 표현식이 추가될 수 있습니다. 이로 인해, 테이블의 컬럼 수 한계 또는 테이블의 행 크기 한계에 도달할 수도 있습니다. 이러한 한계를 초과하면 임시 테이블이 정렬 조작을 수행해야 하는 경우에 오류가 발생합니다.

fetch-first-clause



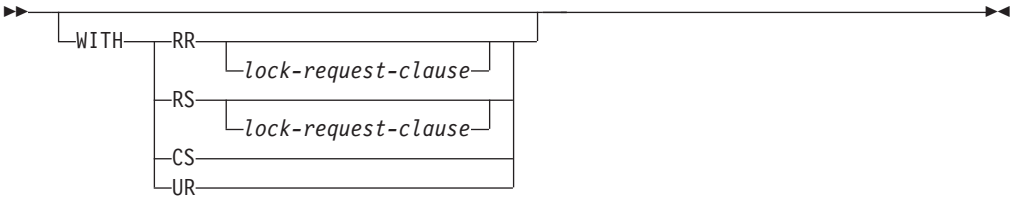
*fetch-first-clause*는 검색할 수 있는 최대 행 수를 설정합니다. 이 절을 사용하면 데이터베이스 관리 프로그램이 이 절을 지정하지 않는 경우에 결과 테이블에 있을 수 있는 행 수에 상관없이 응용프로그램이 *integer*개 이상의 행을 검색하지 않는다는 것을 알 수 있습니다. *integer*개 이상의 행을 페치하려는 시도는 데이터의 일반 목적과 같은 방식으로 처리됩니다(SQLSTATE 02000). *integer* 값은 (0이 아닌) 양의 정수여야 합니다.

*fetch-first-clause*를 사용하면 많아야 *integer* 행이 검색된다는 사실을 기초로 하는 subselect 또는 fullselect의 쿼리 최적화에 영향을 줍니다. 가장 외부 fullselect에 fetch-first-clause이 지정되고 optimize-for-clause가 select 명령문에 지정되면, 데이터베이스 관리 프로그램은 optimize-for-clause의 *integer*를 사용하여 가장 외부 fullselect의 쿼리 최적화에 영향을 줍니다.

결과 테이블을 처음 *integer*개의 행으로 제한하면 성능이 향상됩니다. 데이터베이스 관리 프로그램이 처음 *integer*개의 행을 판별하면 쿼리의 처리를 중지합니다. *fetch-first-clause* 및 *optimize-for-clause*이 모두 지정되면 이 절에서 보다 낮은 *integer* 값은 통신 버퍼 크기를 좌우하는 데 사용됩니다.

fullselect의 FROM절에 SQL 데이터 변경 명령문이 포함되어 있으면 페치할 행 수에 대한 한계와 무관하게 모든 행이 수정됩니다.

isolation-clause



선택적 *isolation-clause*는 subselect 또는 fullselect가 실행되는 분리 레벨 및 특정 잠금 유형의 획득 여부를 지정합니다.

- RR = 반복 읽기
- RS = 읽기 안정성
- CS = 커서 안정성
- UR = 언커미트 읽기

lock-request-clause

*lock-request-clause*는 삽입, 갱신 또는 삭제 조작 내의 위치 지정 읽기 조작과 쿼리에만 적용됩니다. 삽입, 갱신 및 삭제 조작 자체는 데이터베이스 관리 프로그램에서 판별되는 잠금을 사용하여 실행합니다.

선택적 *lock-request-clause*는 데이터베이스 관리 프로그램이 획득 및 보유하는 잠금 유형을 지정합니다.

SHARE
동시 프로세스는 데이터에 대해 SHARE 또는 UPDATE 잠금을 획득할 수 있습니다.

UPDATE
동시 프로세스는 데이터에 대해 SHARE 잠금을 획득할 수는 있지만 UPDATE 또는 EXCLUSIVE 잠금은 획득할 수 없습니다.

EXCLUSIVE
동시 프로세스는 데이터에 대해 잠금을 획득할 수 없습니다.

isolation-clause **제한사항**:

- *isolation-clause*는 CREATE TABLE, CREATE VIEW 또는 ALTER TABLE 명령문에 대해 지원되지 않습니다(SQLSTATE 42601).

- *isolation-clause*는 트리거 호출, 참조 무결성 스캔 또는 MQT 유지보수를 야기하는 subselect 또는 fullselect에 대해 지정될 수 없습니다.
- 해당 subselect 또는 fullselect가 옵션 INHERIT ISOLATION LEVEL WITH LOCK REQUEST와 함께 선언되지 않은 SQL 함수를 참조하는 경우, subselect 또는 fullselect는 *lock-request-clause*를 포함할 수 없습니다(SQLSTATE 42601).
- *lock-request-clause*를 포함하는 subselect 또는 fullselect는 MQT 경로지정에 적합하지 않습니다.
- *isolation-clause*가 SQL 함수 본문, SQL 메소드 또는 트리거 내의 *subselect* 또는 *fullselect*에 지정된 경우, 절이 무시되고 경고가 리턴됩니다.
- *isolation-clause*가 스크롤 가능한 커서를 사용하여 *subselect* 또는 *fullselect*에 대해 지정된 경우 절이 무시되고 경고가 리턴됩니다.
- *isolation-clause*와 *lock-request-clause*는 공통 테이블 표현식에서 분리 또는 잠금 의도 충돌이 발생될 컨텍스트에 지정할 수 없습니다(SQLSTATE 42601). 이 제한사항은 별명 또는 기본 테이블에 적용되지 않습니다. 다음 예는 *a*에서 분리 충돌을 작성하여 오류를 리턴합니다.
  - 뷰:

    ```
    create view a as (...);
    (select * from a with RR USE AND KEEP SHARE LOCKS)
       UNION ALL
    (select * from a with UR);
    ```

  - 공통 테이블 표현식:

    ```
    WITH a as (...)
    (select * from a with RR USE AND KEEP SHARE LOCKS)
       UNION ALL
    (select * from a with UR);
    ```

- *isolation-clause*는 XML 컨텍스트에 지정될 수 없습니다(SQLSTATE 2200M).

subselect의 예

예 A1: EMPLOYEE 테이블에서 모든 컬럼 및 행을 선택하십시오.

```
SELECT * FROM EMPLOYEE
```

예 A2: EMP_ACT과 EMPLOYEE 테이블을 조인하고 EMP_ACT 테이블에서 모든 컬럼을 선택한 다음, EMPLOYEE 테이블에서 직원의 성(LASTNAME)을 결과의 각 행에 추가하십시오.

```
SELECT EMP_ACT.*, LASTNAME
  FROM EMP_ACT, EMPLOYEE
  WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

예 A3: EMPLOYEE와 DEPARTMENT 테이블을 조인하고 1930년 이전에 출생한 (BIRTHDATE) 모든 직원의 직원 번호(EMPNO), 직원 성(LASTNAME), 부서 번호 (EMPLOYEE 테이블의 WORKDEPT 및 DEPARTMENT 테이블의 DEPTNO) 및 부서 이름(DEPTNAME)을 선택하십시오.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
   FROM EMPLOYEE, DEPARTMENT
   WHERE WORKDEPT = DEPTNO
   AND YEAR(BIRTHDATE) < 1930
```

예 A4: 직업(JOB)을 선택하고 EMPLOYEE 테이블에서 직업 코드가 같은 각 행 그룹의 최소 및 최대 급여(SALARY)를 선택하십시오. 단, 둘 이상의 행이 있고 최대 급여가 27000 이상인 그룹에 대해서만 선택하십시오.

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1
  AND MAX(SALARY) >= 27000
```

예 A5: 부서(WORKDEPT) 'E11'의 직원(EMPNO)에 대해 EMP_ACT 테이블의 모든 행을 선택하십시오. (직원 부서 번호는 EMPLOYEE 테이블에 표시되어 있습니다.)

```
SELECT *
  FROM EMP_ACT
  WHERE EMPNO IN
           (SELECT EMPNO
               FROM EMPLOYEE
               WHERE WORKDEPT = 'E11')
```

예 A6: EMPLOYEE 테이블에서 부서 번호(WORKDEPT)를 선택하고 최대 급여가 모든 직원의 평균 급여보다 적은 모든 부서의 최대 부서 급여(SALARY)를 선택하십시오.

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                           FROM EMPLOYEE)
```

이 예에서 HAVING절의 서브쿼리는 한 번만 실행됩니다.

예 A7: EMPLOYEE 테이블을 사용하여 부서 번호(WORKDEPT)를 선택하고 최대 급여가 다른 모든 부서의 평균 급여보다 적은 모든 부서의 최대 부서 급여(SALARY)를 선택하십시오.

```
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 A6과는 대조적으로 HAVING절의 서브쿼리를 각 그룹에 대해 실행해야 합니다.

예 A8: 해당 부서의 평균 급여 및 인원 수와 함께 영업 담당자의 사번 및 급여를 판별하십시오.

이 쿼리는 먼저 AVGSALARY 컬럼과 EMPCOUNT 컬럼 및 WHERE절에서 사용되는 DEPTNO 컬럼을 얻기 위해 중첩 테이블 표현식(DINFO)을 작성해야 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 일반 뷰로서 DINFO 뷰 작성에 드는 오버헤드가 절약됩니다. 명령문 준비 중에는 나머지 쿼리의 컨텍스트로 인해 뷰의 카탈로그에 액세스하지 말고 영업 담당자 부서의 행만 뷰에서 고려해야 합니다.

예 A9: 5개의 무작위 직원 그룹에 대한 평균 교육 수준과 급여를 표시하십시오.

이 쿼리는 중첩 테이블 표현식을 사용하여 각 직원에 대한 무작위 값을 설정함으로써 이후에 GROUP BY절에서 이 값을 사용할 수 있도록 해야 합니다.

```
SELECT RANDID , AVG(EDLEVEL), AVG(SALARY)
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
           FROM EMPLOYEE
       ) AS EMPRAND
  GROUP BY RANDID
```

예 A10: EMP_ACT 테이블을 쿼리하여 급여가 모든 직원의 상위 10위에 있는 직원의 프로젝트 번호를 리턴하십시오.

```
SELECT EMP_ACT.EMPNO,PROJNO
   FROM EMP_ACT
   WHERE EMP_ACT.EMPNO IN
       (SELECT EMPLOYEE.EMPNO
        FROM EMPLOYEE
        ORDER BY SALARY DESC
        FETCH FIRST 10 ROWS ONLY)
```

예 A11: PHONES 및 IDS가 같은 카디널리티의 배열 값을 가진 두 개의 SQL 변수라고 가정하면 이들 배열을 3개의 컬럼(하나는 각 배열에 대해, 하나는 위치에 대해)과, 각 배열 요소당 하나의 행이 있는 테이블로 변환합니다.

```
  SELECT T.PHONE, T.ID, T.INDEX FROM UNNEST(PHONES, IDS)
    WITH ORDINALITY AS T(PHONE, ID, INDEX)
    ORDER BY T.INDEX
```

조인 예

예 B1: 이 예는 테이블 J1 및 J2를 사용하여 수행한 여러 조인의 결과를 보여 줍니다. 이 테이블에는 다음과 같이 행이 포함됩니다.

```
SELECT * FROM J1

   W   X
--- ------
   A       11
   B       12
   C       13

SELECT * FROM J2

   Y   Z
--- ------
   A       21
   C       22
   D       23
```

다음 쿼리는 두 테이블의 첫 번째 컬럼과 일치하는 J1 및 J2 테이블의 내부 조인을 수행합니다.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y

   W   X      Y   Z
--- ------ --- ------
   A       11 A       21
   C       13 C       22
```

이 내부 조인 예에서 J1 테이블에서 컬럼이 W='C'인 행과 J2 테이블에서 컬럼이 Y='D' 인 행은 다른 테이블에 일치하는 항목이 없으므로 결과에 포함되지 않습니다. 다음과 같은 다른 양식의 내부 조인도 같은 결과를 생성함에 유의하십시오.

```
SELECT * FROM J1, J2 WHERE W=Y
```

다음의 왼쪽 외부 조인은 J1에서 J2의 컬럼에 대해 널(NULL)인 누락된 행을 다시 얻습니다. J1의 모든 행이 포함됩니다.

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y

   W   X      Y   Z
--- ------ --- ------
   A       11 A       21
   B       12 -        -
   C       13 C       22
```

다음의 오른쪽 외부 조인은 J2에서 J1의 컬럼에 대해 널(NULL)인 누락된 행을 다시 얻습니다. J2의 모든 행이 포함됩니다.

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y

   W   X      Y   Z
--- ------ --- ------
   A       11 A       21
   C       13 C       22      -        - D       23
```

다음의 완전 외부 조인은 적절할 경우 J1 및 J2에서 널(NULL)을 갖는 누락된 행을 다시 얻습니다. J1 및 J2의 모든 행이 포함됩니다.

```
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: 앞의 예에 나온 J1 및 J2 테이블을 사용하여 검색 조건에 추가 술어를 추가할 경우 어떻게 되는지 조사하십시오.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13

  W   X      Y   Z
--- ------ --- ------
  C       13 C       22
```

예 B1의 내부 조인과 비교해 볼 때 추가 조건으로 인해 내부 조인이 한 개의 행만 선택했습니다.

이 조건이 완전 외부 조인에 어떠한 영향을 미치는지 주의하여 살펴보십시오.

```
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 내부 조인에 한 개의 행만 있으므로 결과에 5개 행이 포함되며(추가 술어가 없는 경우 4개와 비교) 두 테이블의 모든 행이 리턴되어야 합니다.

다음 쿼리는 WHERE절에 같은 추가 술어를 삽입할 경우 완전히 다른 결과가 리턴됨을 보여 줍니다.

```
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 다음에 적용됩니다. 이 중간 결과는 예 B1의 완전 외부 조인 쿼리에 대한 결과와 같습니다. WHERE절이 중간 결과에 적용되어 X=13인 행을 제외한 모든 행을 제거합니다. 외부 조인을 수행할 때 술어의 위치를 선택하면 결과에 큰 영향을 미칠 수 있습니다. 술어가 X=13 대신 X=12일 때 발생하는 상황을 고려하십시오. 다음 내부 조인은 행을 리턴하지 않습니다.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=12
```

다음의 오른쪽 외부 조인은 J2에서 J1의 컬럼에 대해 널(NULL)인 누락된 행을 다시 얻습니다.

```
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 -       -
```

추가 술어가 대신 WHERE절에 있으면 1개 행이 리턴됩니다.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=12

   W   X      Y   Z
--- ------ --- ------
   B       12 -       -
```

예 B3: 매니저가 없는 부서를 포함하여 직원 번호 및 매니저의 성과 함께 모든 부서를 나열하십시오.

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
  FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
       ON MGRNO = EMPNO
```

예 B4: 매니저가 없는 직원을 포함하여 직원 번호 및 매니저의 성과 함께 모든 직원 번호와 성을 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 테이블에서 식별된 매니저의 성을 판별하고 왼쪽 외부 조인은 해당 부서가 DEPARTMENT에 없는 경우에도 각 직원이 나열되도록 합니다.

그룹화 세트, 큐브 및 롤업 예

예 C1부터 예 C4까지의 쿼리는 'WEEK(SALES_DATE) = 13' 술어에 기초하여 SALES 테이블 행의 서브세트를 사용합니다.

```
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
```

결과는 다음과 같습니다.

```
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 컬럼이 3개 이상인 기본 GROUP BY절이 있는 쿼리입니다.

```
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
```

결과는 다음과 같습니다.

```
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 행의 서로 다른 두 개의 그룹화 세트에 기초하여 결과를 생성하십시오.

```
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
```

결과는 다음과 같습니다.

```
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 13이 있는 행은 첫 번째 그룹화 세트에 속하고 나머지 행은 두 번째 그룹화 세트에 속합니다.

예 C3: 예 C2의 그룹화 세트에 포함되는 3개의 다른 컬럼을 사용하고 롤업(ROLLUP)을 수행하는 경우 (WEEK,DAY_WEEK,SALES_PERSON), (WEEK, DAY_WEEK), (WEEK) 및 총 합계에 대한 그룹화 세트를 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
```

결과는 다음과 같습니다.

```
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: ROLLUP을 CUBE로 바꾸어 예 C3과 같은 쿼리를 실행하면 결과에서 (WEEK,SALES_PERSON), (DAY_WEEK,SALES_PERSON), (DAY_WEEK), (SALES_PERSON)에 대한 추가 그룹화 세트를 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
```

결과는 다음과 같습니다.

```
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 테이블에서 선택한 행의 총 합계와 SALES_PERSON 및 MONTH에 의해 집계된 행 그룹을 포함하는 결과 세트를 생성하십시오.

```
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
```

결과는 다음과 같습니다.

```
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 두 개의 단순 ROLLUP 쿼리 다음에 두 개의 ROLLUP을 단일 결과 세트의 그룹화 세트로 처리하는 쿼리를 보여 주며 그룹화 세트에 포함된 각 컬럼에 대해 행 순서를 지정합니다.

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
```

결과는 다음과 같습니다.

```
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
```

결과는 다음과 같습니다.

```
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
```

결과는 다음과 같습니다.

```
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
```

두 개의 ROLLUP을 그룹화 세트로 사용하면 결과에 중복 행이 포함됩니다. 즉, 총 합계 행이 두 개가 있습니다.

ORDER BY의 사용이 어떠한 방식으로 결과에 영향을 미치는지 주의하여 살펴보십시오.

- 첫 번째 그룹화 세트에서 주 53이 끝으로 재배치되었습니다.
- 두 번째 그룹화 세트에서 12월이 끝에 배치되었고 영역이 영문 순서대로 표시됩니다.
- 널(Null) 값은 높게 정렬됩니다.

예 C7: 여러 ROLLUP을 한 과정에서 수행하는 쿼리(예 C6-3과 같은)에서 각 행을 생성한 그룹화 세트를 나타내기를 원할 수 있습니다. 다음 단계는 결과 세트에 있는 각 행의 원점을 나타내는 컬럼(GROUP)을 제공하는 방법을 보여 줍니다. 원점은 두 그룹화 세트 중 결과 세트의 행을 생성한 세트를 의미합니다.

1단계: VALUES절(fullselect의 다른 양식)에서 선택한 쿼리를 사용하여 새 데이터 값을 ″생성″하는 방법을 소개합니다. 이 쿼리는 2개의 컬럼 ″R1″과 ″R2″이 있고 1개의 데이터 행이 있는 ″X″ 테이블을 파생하는 방법을 보여 줍니다.

```
SELECT R1,R2
FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

결과는 다음과 같습니다.

```
R1      R2
------- -------
  GROUP 1 GROUP 2
```

2단계: ″X″ 테이블과 SALES 테이블의 교차 곱을 생성하십시오. 이에 따라, 컬럼 ″R1″ 및 ″R2″가 모든 행에 추가됩니다.

```
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)
```

이에 따라, 컬럼 ″R1″ 및 ″R2″가 모든 행에 추가됩니다.

3단계: 이제 롤업(rollup) 분석에 그룹화 세트가 있는 이러한 컬럼이 포함되도록 컬럼을 조인할 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
```

결과는 다음과 같습니다.

```
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
```

4단계: R1과 R2는 서로 다른 그룹화 세트에서 사용되므로 결과에서 R1이 널(NULL)이 아닐 때만 R2가 널(NULL)이 되고, 결과에서 R2가 널(NULL)이 아닐 때만 R1이 널(NULL)이 됩니다. 이는 COALESC 함수를 사용하여 이러한 컬럼을 단일 컬럼으로 통합할 수 있음을 의미합니다. 또한 ORDER BY절에서 이 컬럼을 사용하여 두 그룹화 세트의 결과를 함께 보관할 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;
```

결과는 다음과 같습니다.

```
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
```

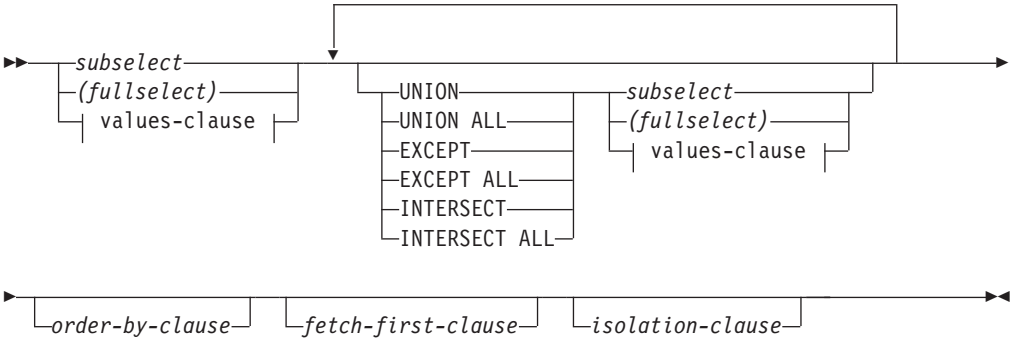
예 C8: 다음 예는 CUBE를 수행할 때 여러 집계 함수의 사용을 보여 줍니다. 또한 캐스트(CAST) 함수 및 반올림을 사용하여 적절한 정밀도 및 스케일로 10진수 결과를 생성합니다.

```
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
```

결과는 다음과 같습니다.

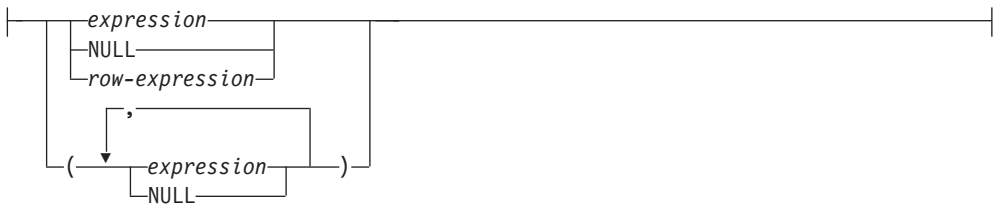
```
MONTH       REGION          UNITS_SOLD  BEST_SALE   AVG_UNITS_SOLD
----------- --------------- ----------- ----------- --------------
                  3 Manitoba                22           7           3.14
                  3 Ontario-North            8           3           2.67
                  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



values-clause:

values-row:



*fullselect*는 select-statement, INSERT문 및 CREATE VIEW문의 구성요소이며 명령문의 구성요소인 특정 술어의 구성요소이기도 합니다. 술어의 구성요소인 fullselect를 *subquery*라 하며 괄호로 닫힌 fullselect도 subquery라고 합니다.

집합 연산자 UNION, EXCEPT 및 INTERSECT는 관계 연산자 union, difference 및 intersection에 해당합니다.

A는 결과 테이블을 지정합니다. 집합 연산자를 사용하지 않을 경우 fullselect의 결과는 지정된 subselect 또는 values-clause의 결과입니다.

*fullselect*에 대한 권한 부여는 ″SQL 쿼리″의 권한 부여 섹션에 설명되어 있습니다.

values-clause

결과 테이블에 있는 행의 각 컬럼에 대해 표현식 또는 행 표현식을 사용하여 실제 값을 지정하여 결과 테이블을 유도합니다. 여러 행을 지정할 수 있습니다. *values-clause*에 있는 표현식의 결과 유형은 행 유형이 될 수 없습니다(SQLSTATE 428H2).

널(NULL)은 단일 컬럼 결과 테이블의 컬럼 값 또는 *row-expression* 내에서 *values-row*의 다중 스펙과 함께 사용되어야만 하며, 동일한 컬럼에서 최소한 한 개 행은 널(NULL)이 되어서는 안됩니다(SQLSTATE 42608).

*values-row*는 다음 중 하나에 의해 지정됩니다.

- 단일 컬럼 결과 테이블의 단일 표현식
- 괄호로 묶여 있고 쉼표로 구분되는 *n* 표현식(또는 널(NULL)). 여기서, *n*은 결과 테이블의 컬럼 수, 또는 다중 컬럼 결과 테이블의 행 표현식입니다.

다중 행 VALUES 절의 각 *values-row*에는 동일한 수의 컬럼이 있어야 합니다(SQLSTATE 42826).

다음은 *values-clause* 및 의미의 예입니다.

```
VALUES (1),(2),(3)          - 1 개의 컬럼에서 3 개 행
VALUES 1, 2, 3              - 1 개의 컬럼에서 3 개 행
VALUES (1, 2, 3)            - 3 개의 컬럼에서 1 개 행
VALUES (1,21),(2,22),(3,23) - 2 개의 컬럼에서 3 개 행
```

*values-row*의 *n* 스펙으로 구성된 *values-clause*(RE_1부터 RE_n, 여기서, *n*은 1보다 큼)는 다음과 동일합니다.

RE_1 UNION ALL RE_2 ... UNION ALL RE_n

이는 각 *values-row*의 해당 컬럼이 비교 가능해야 함을 의미합니다(SQLSTATE 42825).

UNION 또는 UNION ALL

두 개의 다른 결과 테이블(R1 및 R2)을 조인하여 결과 테이블을 파생합니다. UNION ALL이 지정된 경우 결과는 R1 및 R2의 모든 행으로 구성됩니다. ALL 옵션없이 UNION이 지정된 경우 결과는 R1 또는 R2의 모든 행 세트이며 중복 행은 제거됩니다. 그러나 어느 경우에서도 UNION 테이블의 각 행은 R1 또는 R2의 행이 됩니다.

EXCEPT 또는 EXCEPT ALL

두 개의 다른 결과 테이블(R1 및 R2)을 조인하여 결과 테이블을 파생합니다. EXCEPT ALL이 지정된 경우 결과는 R2에 해당 행이 없는 모든 행으로 구성되며 중복 행도 적용됩니다. ALL 옵션없이 EXCEPT가 지정된 경우 결과는 R1에만 있는 모든 행으로 구성되며 이 조작의 결과에 있는 중복 행은 제거됩니다.

다른 SQL 구현과의 호환성을 위해, MINUS를 EXCEPT의 동의어로 지정할 수 있습니다.

INTERSECT 또는 INTERSECT ALL

두 개의 다른 결과 테이블(R1 및 R2)을 조인하여 결과 테이블을 파생합니다. INTERSECT ALL이 지정된 경우 결과는 R1 및 R2 모두에 있는 모든 행으로 구성됩니다. ALL 옵션없이 INTERSECT가 지정된 경우 결과는 R1 및 R2 모두에 있는 모든 행으로 구성되며 중복 행은 제거됩니다.

order-by-clause

*order-by-clause*의 세부사항은 『subselect』를 참조하십시오. ORDER BY절을 포함한 fullselect는 다음에 지정될 수 없습니다(SQLSTATE 428FJ).

- 구체화된 쿼리 테이블(MQT)
- 뷰의 가장 외부 fullselect

주: fullselect의 ORDER BY절은 쿼리가 리턴하는 행의 순서에는 영향을 미치지 않습니다. ORDER BY절은 가장 외부 fullselect에 지정된 경우에만 리턴되는 행의 순서에 영향을 미칩니다.

fetch-first-clause

*fetch-first-clause*의 세부사항은 『subselect』를 참조하십시오. FETCH FIRST절을 포함한 fullselect는 다음에 지정될 수 없습니다(SQLSTATE 428FJ).

- 구체화된 쿼리 테이블(MQT)
- 뷰의 가장 외부 fullselect

주: fullselect의 FETCH FIRST절은 쿼리가 리턴하는 행의 수에 영향을 미치지 않습니다. FETCH FIRST절은 가장 외부 fullselect에 지정된 경우에만 리턴되는 행의 수에 영향을 미칩니다.

isolation-clause

*isolation-clause*의 세부사항은 『subselect』를 참조하십시오. *isolation-clause*이 fullselect에 지정되고 fullselect의 subselect에 동일하게 적용 가능한 경우, *isolation-clause*는 fullselect에 적용됩니다. 예를 들어, 다음 쿼리를 고려하십시오.

```
SELECT NAME FROM PRODUCT
UNION
SELECT NAME FROM CATALOG
WITH UR
```

분리 절 WITH UR을 subselect SELECT NAME FROM CATALOG에만 적용할 수 있더라도, 이는 전체 fullselect에 적용됩니다.

결과 테이블 R1 및 R2의 컬럼 수는 같아야 합니다(SQLSTATE 42826). ALL 키워드가 지정되지 않은 경우, R1 및 R2에는 CLOB, DBCLOB, BLOB의 데이터 유형, 이러한 유형에 대한 구별 유형 또는 구조화된 유형이 있는 컬럼은 포함되어서는 안됩니다(SQLSTATE 42907).

결과의 컬럼은 다음과 같이 이름이 지정됩니다.

- R1의 *n* 번째 컬럼과 R2의 *n* 번째 컬럼에 동일한 결과 컬럼 이름이 있을 경우 R의 *n* 번째 컬럼이 결과 컬럼 이름을 가집니다.
- R1의 *n* 번째 컬럼과 R2의 *n* 번째 컬럼이 다른 결과 컬럼 이름을 가질 경우 이름이 생성됩니다. 이 이름은 ORDER BY 또는 UPDATE절에서 컬럼 이름으로 사용될 수 없습니다.

SQL문의 DESCRIBE를 수행하고 SQLNAME 필드를 참조하여 생성된 이름을 판별할 수 있습니다.

중복 행: 각 첫 번째 행의 값이 두 번째의 해당 값과 같은 경우에 두 행이 중복됩니다. 중복 행을 판별하기 위해 두 개의 널(NULL) 값은 동일한 것으로 간주되며, 같은 수의 10진수 부동 소수점 표현은 동일한 것으로 간주됩니다. 예를 들어, 2.00 및 2.0은 동일한 값을 가지지만(2.00과 2.0은 같은 값으로 비교) 다른 지수를 가지며, 2.00 및 2.0 모두로 나타낼 수 있습니다. 예를 들어, UNION 조작의 결과 테이블에 10진수 부동 소수점 컬럼이 포함되고 동일 수의 다중 표현이 존재하면, 리턴되는 값을(예를 들어 2.00 또는 2.0) 예상할 수 없습니다. 자세한 정보는 139 페이지의 『숫자 비교』의 내용을 참조하십시오.

표현식에 여러 조작이 조인된 경우, 괄호 안의 조작이 먼저 수행됩니다. 괄호가 없을 경우, 조작은 왼쪽에서 오른쪽으로 수행되며 예외적으로 UNION 또는 EXCEPT 조작에 앞서 모든 INTERSECT 조작이 수행됩니다.

다음 예에서 테이블 R1 및 R2의 값은 왼쪽에 표시됩니다. 나열된 기타 표제는 R1 및 R2에 대한 다양한 세트 조작의 결과로서 값을 표시합니다.

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					

Fullselect의 예

예 1: EMPLOYEE 테이블에서 모든 컬럼과 행을 선택하십시오.

```
SELECT * FROM EMPLOYEE
```

예 2: 해당 부서 번호(WORKDEPT)가 'E'로 시작하거나 프로젝트 번호(PROJNO)가 'MA2100', 'MA2110' 또는 'MA2112'와 같은 EMP_ACT 테이블의 프로젝트에 지정된 EMPLOYEE 테이블에 있는 모든 사원의 사원 번호(EMPNO)를 나열하십시오.

```
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 쿼리를 작성함과 동시에 EMPLOYEE 테이블의 행을 'emp'로, EMP_ACT 테이블의 행을 'emp_act'로 "태그"하십시오. 예 2의 결과와는 달리, 이 쿼리는 연관된 "태그"에서 행을 검색한 테이블을 식별함으로써 동일 EMPNO를 두 번 이상 리턴할 수 있습니다.

```
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: 중복 행이 제거되지 않도록 UNION ALL만 사용하여 예 2에서와 같은 쿼리를 작성하십시오.

```
SELECT EMPNO
    FROM EMPLOYEE
    WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

예 5: 예 3에서와 같은 쿼리를 작성하고 현재 테이블에 없는 두 사원만 추가로 포함시킨 다음, 이들 행을 ″new″로 태그하십시오.

```
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: EXCEPT를 사용하여 T2가 아닌 T1에 있는 모든 행을 산출하십시오.

```
   (SELECT * FROM T1)
EXCEPT ALL
   (SELECT * FROM T2)
```

널(NULL)이 포함되지 않은 경우 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: INTERSECT를 사용하여 중복을 제거한 다음, T1 및 T2 테이블에 있는 모든 행을 산출하십시오.

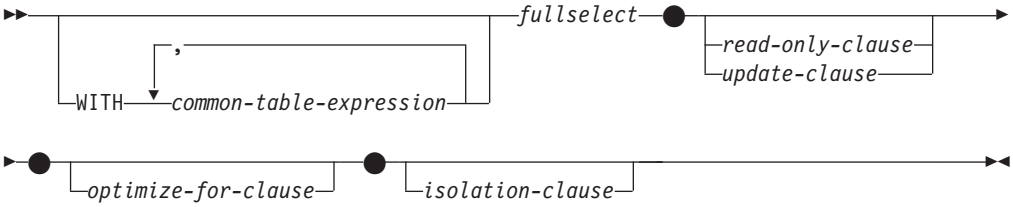
```
   (SELECT * FROM T1)
INTERSECT
   (SELECT * FROM T2)
```

널(NULL)이 포함되지 않은 경우 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.

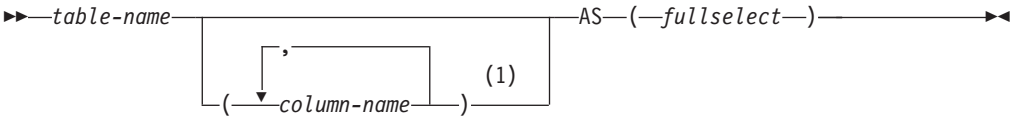
SELECT



*select-statement*는 DECLARE CURSOR문에 직접 지정될 수 있거나 준비된 후 DECLARE CURSOR문에서 참조될 수 있는 쿼리 양식입니다. 명령행 처리기(또는 유사 도구)를 사용하여 동적 SQL문의 사용을 통해 발행될 수 있으며 결과 테이블이 사용자 화면에 표시되도록 합니다. 이런 경우에 *select-statement*가 지정한 테이블은 fullselect의 결과입니다.

*select-statement*에 대한 권한 부여는 ″SQL 쿼리″의 권한 부여 섹션에 설명되어 있습니다.

common-table-expression



주:

1 공통 테이블 표현식이 반복적이거나 fullselect가 중복된 컬럼 이름으로 나타나는 경우 컬럼 이름이 지정되어야 합니다.

*common table expression*은 fullselect의 FROM절에서 테이블 이름으로 지정될 수 있는 *table-name*으로 결과 테이블을 정의할 수 있습니다. 다중 공통 테이블 표현식은 단일 WITH 키워드 다음에 지정될 수 있습니다. 지정된 각 공통 테이블 표현식은 후속 공통 테이블 표현식의 FROM절에서 이름으로 참조될 수도 있습니다.

컬럼의 목록이 지정되면 이 목록에 포함된 이름 수는 fullselect의 결과 테이블에 있는 컬럼 수와 동일해야 합니다. 각 *column-name*은 고유해야 하고 규정화되어서는 안됩니다. 이러한 컬럼 이름이 지정되지 않으면 그 이름들은 공통 테이블 표현식을 정의하는 데 사용된 fullselect의 선택 목록으로부터 유출됩니다.

공통 테이블 표현식의 *table-name*은 동일한 명령문의 또 다른 공통 테이블 표현식 *table-name*과 달라야 합니다(SQLSTATE 42726). 공통 테이블 표현식이 INSERT문에 지정될 경우 *table-name*은 삽입의 오브젝트인 테이블 또는 뷰 이름과 같을 수 없습니다(SQLSTATE 42726). 공통 테이블 표현식 *table-name*은 fullselect 전체의 FROM

절에서 테이블 이름으로 지정될 수 있습니다. 공통 테이블 표현식의 *table-name*은 기존의 테이블, 뷰 또는 별명(카탈로그에서)을 같은 규정된 이름으로 대체합니다.

둘 이상의 공통 테이블 표현식이 같은 명령문에서 정의된 경우 공통 테이블 표현식 간의 주기적 참조는 허용되지 않습니다(SQLSTATE 42835). 두 개의 공통 테이블 표현식 *dt1* 및 *dt2*가 작성될 때 *cyclic reference*가 발생하여 *dt1*은 *dt2*를 참조하고 *dt2*는 *dt1*을 참조합니다.

공통 테이블 표현식의 fullselect가 FROM절에서 *data-change-table-reference*를 포함하고 있으면 공통 테이블 표현식은 데이터를 수정합니다. 데이터를 수정하는 공통 테이블 표현식은 공통 테이블 표현식이 명령문의 다른 곳에 사용되는지 여부와 상관없이 명령문이 프로세스될 때마다 평가됩니다. 데이터를 읽거나 수정하는 공통 테이블 표현식이 하나 이상 있는 경우 모든 공통 테이블 표현식은 발생 순서대로 처리되며 모든 제한조건 및 트리거를 포함하여 데이터를 읽거나 수정하는 각 공통 테이블 표현식은 뒤이은 공통 테이블 표현식이 실행되기 전에 완전히 실행됩니다.

또한 CREATE VIEW 및 INSERT문에서 fullselect 이전의 공통 테이블 표현식은 선택사항입니다.

다음에 공통 테이블 표현식이 사용될 수 있습니다.

- 뷰를 작성하지 않는 뷰를 대신할 경우(뷰의 일반 사용이 필요하지 않아 위치 지정된 갱신 또는 삭제가 사용되지 않을 때)
- 스칼라 subselect로부터 유출된 컬럼 또는 결정적이 아니거나 외부 조치가 있는 함수로 그룹화를 가능하게 할 경우
- 원하는 결과 테이블이 호스트 변수에 기반할 경우
- 동일한 결과 테이블이 fullselect에서 공유되어야 할 경우
- 재귀를 사용하여 결과를 얻어야 할 경우
- 쿼리에서 다중 SQL 데이터 변경 명령문을 프로세스할 필요가 있을 경우

공통 테이블 표현식의 fullselect가 FROM절에서 그 자체에 대한 참조를 포함하고 있으면 공통 테이블 표현식은 *recursive common table expression*입니다. 재귀를 사용하는 쿼리는 BOM(bill of materials), 예약 시스템 및 네트워크 플래닝과 같은 지원 응용프로그램에서 유용합니다.

다음은 재귀 공통 테이블 표현식에 대한 설명입니다.

- 재귀 주기의 일부인 각 fullselect는 SELECT 또는 SELECT ALL로 시작해야 합니다. SELECT DISTINCT 사용은 허용되지 않습니다(SQLSTATE 42925). 또한 통합은 UNION ALL(SQLSTATE 42925)을 사용해야 합니다.
- 컬럼 이름은 공통 테이블 표현식(SQLSTATE 42908)의 지정된 다음 *table-name*이어야 합니다.

- 처음 통합의 처음 fullselect(초기화 fullselect)는 FROM절에서(SQLSTATE 42836) 공통 테이블 표현식의 모든 컬럼에 대한 참조를 포함해서는 안됩니다.
- 공통 테이블 표현식의 컬럼 이름이 반복 fullselect에서 참조되는 경우 그 컬럼에 대한 데이터 유형, 길이 및 코드 페이지는 초기화 fullselect에 기초하여 판별됩니다. 반복 fullselect의 해당 컬럼은 초기화 fullselect에 기초하여 판별된 데이터 유형 및 길이와 같은 데이터 유형 및 길이를 가져야 하고 코드 페이지가 일치해야 합니다 (SQLSTATE 42825). 그러나 문자열 유형의 경우 두 개의 데이터 유형 길이가 다를 수도 있습니다. 이 경우 반복 fullselect의 컬럼은 초기화 fullselect로부터 판별된 길이에 항상 지정 가능한 길이를 가져야 합니다.
- 재귀 주기의 일부인 각 fullselect는 모든 집계 함수, group-by-clauses 또는 having-clauses를 포함해서는 안됩니다(SQLSTATE 42836).

  이러한 fullselect의 FROM절은 재귀 주기의 일부인 공통 테이블 표현식에 대해 1개 이하의 참조를 포함할 수 있습니다(SQLSTATE 42836).
- 반복 fullselect 및 전체 반복 fullselect는 order-by-clause를 포함해서는 안됩니다 (SQLSTATE 42836).
- 서브쿼리(스칼라 또는 양이 정해진)는 재귀 주기의 일부이어서는 안됩니다(SQLSTATE 42836).

재귀 공통 테이블 표현식을 개발할 때 무한 재귀 주기(루프)가 작성될 수 있다는 점을 기억하십시오. 재귀 주기가 종료되는지 확인하십시오. 관련된 데이터가 주기적일 경우 이것은 특히 중요합니다. 재귀 공통 테이블 표현식은 무한 루프를 방해하는 술어를 포함해야 합니다. 재귀 공통 테이블 표현식은 다음을 포함해야 합니다.

- 반복 fullselect에서 상수만큼 증가된 정수 컬럼
- ″counter_col < constant″ 또는 ″counter _col < :hostvar″ 양식에서 대화식 fullselect의 Where절의 술어

이 구문을 재귀 공통 테이블 표현식(SQLSTATE 01605)에서 찾을 수 없는 경우 경고가 발행됩니다.

재귀 예: 부품표(BOM)

부품표(BOM) 응용프로그램은 많은 비즈니스 환경에서의 공통 요구사항입니다. BOM 응용프로그램에 대한 재귀 공통 테이블 표현식의 성능을 표시하려면 부품에 필요한 부속 부품 수량과 관련 부속 부품에 대해 설명한 부품 테이블을 살펴보십시오. 이 예의 경우 다음과 같이 테이블을 작성하십시오.

```
CREATE TABLE PARTLIST
                                (PART VARCHAR(8),
                                 SUBPART VARCHAR(8),
                                 QUANTITY INTEGER);
```

이 예에 대한 쿼리 결과를 제공하기 위해 PARTLIST 테이블이 다음 값으로 채워진다고 가정해 보십시오.

```
PART     SUBPART  QUANTITY
-------- -------- -----------
  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

첫 번째 예는 단일 레벨 전개입니다. 이는 『'01'로 식별되는 파트를 빌드하는 데 필요한 파트는 무엇입니까?』라는 질문에 대한 응답입니다. 이 목록에는 직접적인 부속 부품, 부속 부품의 부속 부품 등이 포함됩니다. 그러나 한 부품이 여러 번 사용되면 해당 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;
```

위의 쿼리에는 공통 테이블 표현식이 포함되는데, 이 표현식은 이 쿼리의 재귀 부분을 표시하는 *RPL*이란 이름으로 식별됩니다. 이는 재귀 공통 테이블 표현식의 기본 요소를 예시합니다.

*initialization fullselect*로 지칭되는 UNION의 첫 번째 피연산자(fullselect)는 부품 '01'에 대한 직접 하위(children)를 가져옵니다. 이 fullselect의 FROM절은 소스 테이블을 참조하며 그 자체(이 경우 *RPL*)를 참조하지 않습니다. 이러한 첫 번째 fullselect의 결과는 공통 테이블 표현식 *RPL*(재귀 PARTLIST)로 갑니다. 이 예에서와 같이 UNION은 항상 UNION ALL이어야 합니다.

UNION의 두 번째 피연산자(fullselect)는 *RPL*을 사용하여 부속 부품의 부속 부품을 계산합니다. 이 때 FROM절은 공통 테이블 표현식 *RPL*을 참조하고, 소스 테이블(하위)의 부품을 현재 결과의 부속 부품과 조인한 소스 테이블은 *RPL*(상위)에 들어 있습니다. 그 결과는 다시 *RPL*로 갑니다. 그런 다음 UNION의 두 번째 피연산자가 하위 부품이 더 이상 존재하지 않을 때까지 반복적으로 사용됩니다.

이 쿼리의 주요 fullselect에 있는 SELECT DISTINCT는 같은 부품/부속 부품이 두 번 이상 나열되지 않도록 합니다.

쿼리의 결과는 다음과 같습니다.

```
PART     SUBPART  QUANTITY
-------- -------- -----------
  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 간 후, 다시 '06'으로 가는 결과를 관찰하십시오. 또한 부품 '01'을 통해 직접적으로 한 번, '02'를 통해서 또 한 번, 합쳐서 부품 '06'에 두 번 도달함에 유의하십시오. 그러나 출력 시에 그 부속 구성요소들은 필요에 따라 한 번만 나열됩니다. (이는 SELECT DISTINCT를 사용한 결과입니다.)

재귀 공통 테이블 표현식을 가지고 무한 루프를 도입하는 것이 가능하다는 사실을 기억하는 것이 중요합니다. 이 예에서 상위 테이블과 하위 테이블을 조인시키는 두 번째 피연산자의 검색 조건이 다음과 같이 코딩된 경우 무한 루프가 작성될 수 있습니다.

```
PARENT.SUBPART = CHILD.SUBPART
```

무한 루프를 야기시키는 이 예는 의도한 대로 코딩하지 못한 경우임이 분명합니다. 그러나 재귀 주기가 명백히 종료되도록 하려면 코딩할 것이 무엇인지를 결정하는 데에도 주의해야 합니다.

이 예에서 생성된 결과는 재귀 공통 테이블 표현식을 사용하지 않고 응용프로그램에서 생성될 수 있습니다. 그러나 이러한 접근 방법은 모든 레벨의 재귀에 대해 새 쿼리를 시작해야 합니다. 또한 응용프로그램은 결과를 정렬하기 위해 데이터베이스에 모든 결과들을 다시 두어야 합니다. 이러한 접근 방법은 응용프로그램 논리를 복잡하게 하고 제대로 수행되지 않습니다. 응용프로그램 논리는 요약 전개 쿼리 및 들여쓰기된 전개 쿼리와 같은 다른 BOM 쿼리의 경우 훨씬 어렵고 비효율적입니다.

예 2: 요약 전개

두 번째 예는 요약 전개입니다. 여기에서 제기되는 문제는 부품 '01' 구축에 필요한 각 부품의 전체 수량은 얼마인가 하는 것입니다. 단일 레벨 전개와의 가장 큰 차이점은 수량을 집계해야 한다는 것입니다. 첫 번째 예에서는 필요할 때마다 부품에 필요한 부속 부품의 수량을 표시합니다. 여기에서는 '01' 부품을 구축하는 데 얼마나 많은 부속 부품이 필요한지를 표시하지 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 쿼리는 *RPL* 이름으로 식별되는 재귀 공통 테이블 표현식에서 UNION의 두 번째 피연산자 선택 목록은 수량 집계를 나타냅니다. 얼마나 많은 부속 부품이 사용되는지를 알아 보기 위해 상위 부품 수량을 하위 부품의 상위 부품당 수량으로 곱합니다. 다른 곳에서 한 부품이 여러 번 사용된 경우 또 다른 최종 집계가 필요합니다. 이는 공통 테이블 표현식 *RPL*을 그룹화하고 주요 fullselect의 선택 목록에 있는 SUM 집계 함수를 사용하여 수행됩니다.

쿼리의 결과는 다음과 같습니다.

```
  PART     SUBPART  Total Qty Used
  -------- -------- --------------
    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 직접 파생된 것입니다.

예 3: 용량 제어

테이블에 원하는 부품 레벨보다 높은 레벨이 있을 경우 어떻게 되는지 생각해 봅시다. 즉, 『'01'로 식별되는 부품을 만드는 데 필요한 처음 두 레벨의 부품은 무엇인가』라는 질문의 쿼리를 작성하는 방법을 생각해 봅시다. 예에서는 명확히 설명하기 위해 레벨이 결과에 포함됩니다.

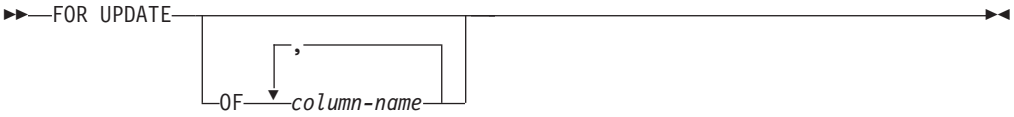
```
WITH RPL (LEVEL, PART, SUBPART, QUANTITY) AS
      (
         SELECT 1,              ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
          FROM PARTLIST ROOT
          WHERE ROOT.PART = '01'
       UNION ALL
                SELECT PARENT.LEVEL+1, CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE  PARENT.SUBPART = CHILD.PART
            AND PARENT.LEVEL < 2
      )
 SELECT PART, LEVEL, SUBPART, QUANTITY
   FROM RPL;
```

이 쿼리는 예 1과 비슷합니다. 원래의 부품으로부터 레벨을 계산하기 위해 컬럼 *LEVEL*이 도입됩니다. 초기화 fullselect에서 *LEVEL* 컬럼 값은 1로 초기화됩니다. 후속 fullselect에서 상위 제품의 레벨은 1씩 증가합니다. 그런 다음 결과의 레벨 수를 제어하기 위해 두 번째 fullselect에는 상위 레벨은 2 미만이어야 한다는 조건이 포함됩니다. 이는 두 번째 fullselect는 하위 부품만이 두 번째 레벨이 되도록 합니다.

쿼리의 결과는 다음과 같습니다.

```
PART     LEVEL       SUBPART  QUANTITY
-------- ----------- -------- -----------
  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
```

update-clause



FOR UPDATE절은 후속 UPDATE문에서 갱신될 수 있는 컬럼을 식별합니다. 각 *column-name*은 규정화되지 않아야 하며 fullselect의 처음 FROM절에서 식별된 테이블 또는 뷰의 컬럼을 식별해야 합니다. FOR UPDATE절이 컬럼 이름없이 지정된 경우, fullselect의 처음 FROM절에서 식별된 테이블 또는 뷰의 모든 갱신 가능한 컬럼이 포함됩니다.

다음 중 하나에 해당하는 경우 FOR UPDATE절이 사용될 수 없습니다.

- select문과 연결된 커서는 삭제할 수 없습니다.
- 선택한 컬럼 중의 하나는 카탈로그 테이블에서 갱신이 불가능한 컬럼이며 FOR UPDATE절은 그 컬럼을 제외하도록 사용되지 않았습니다.

read-only-clause

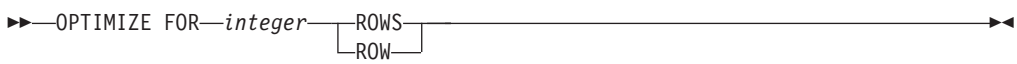
FOR READ ONLY절은 결과 테이블이 읽기 전용이라는 것을 나타내므로 커서는 Positioned UPDATE 및 DELETE문에서 참조될 수 없습니다. FOR FETCH ONLY도 같은 의미입니다.

일부 결과 테이블은 원래 읽기 전용입니다 (예: 읽기 전용 뷰에 기반한 테이블). FOR READ ONLY를 이러한 테이블에 지정할 수는 있으나 스펙에는 영향이 없습니다.

갱신 및 삭제가 허용되는 결과 테이블의 경우 FOR READ ONLY(또는 FOR FETCH ONLY)를 지정하면 데이터베이스 관리 프로그램이 블로킹을 수행하도록 허용하여 FETCH 조작의 성능을 향상시킬 수 있습니다. 예를 들어, FOR READ ONLY 또는 ORDER BY절이 없는 동적 SQL문을 포함하는 프로그램에서 데이터베이스 관리 프로그램은 FOR UPDATE절이 지정된 것처럼 커서를 열 수 있습니다. 따라서 쿼리가 지정된 UPDATE 또는 DELETE문에서 사용되는 경우를 제외하고는 성능을 향상시키기 위해 FOR READ ONLY절을 사용하는 것을 권장합니다.

읽기 전용 테이블은 원래부터 읽기 전용이거나 FOR READ ONLY(FOR FETCH ONLY)로서 지정되었는지에 관계없이 Positioned UPDATE 또는 DELETE문에서 참조되어서는 안됩니다.

optimize-for-clause

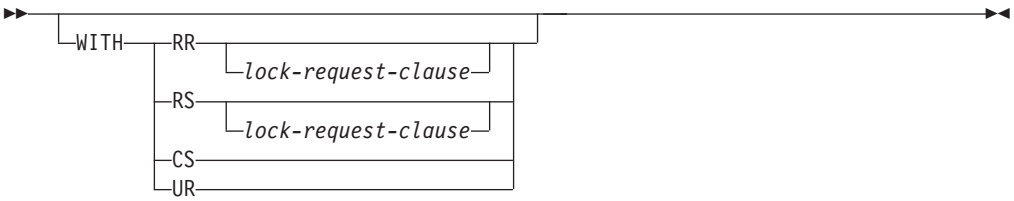


OPTIMIZE FOR절은 *select*문의 특수 처리를 요구합니다. 이 절이 생략되면 모든 결과 테이블의 행이 검색되는 것으로 간주됩니다. 이 절이 지정되면 검색된 행 수가 *n*을 초과하지 않을 것입니다. 여기서, *n*은 *integer* 값입니다. *n* 값은 양수이어야 합니다. OPTIMIZE FOR절 사용은 *n* 행이 검색된다는 가정 하에 쿼리 최적화에 영향을 줍니다. 또한 블록된 커서의 경우 이 절은 각 블록에서 리턴될 행 수에 영향을 줍니다. (즉, *n* 행이 각 블록에서 리턴됩니다.) *fetch-first-clause* 및 *optimize-for-clause*가 모두 지정되면 이러한 절의 하위 정수 값은 통신 버퍼 크기에 영향을 주는 데 사용됩니다. 이 값은 최적화 목록을 위해 개별적으로 고려됩니다.

이 절은 페치될 수 있는 행 수를 제한하지 않습니다. 그러나 성능 이외의 방식으로 결과에 영향을 줍니다. OPTIMIZE FOR *n* ROWS를 사용하면 *n* 행이 검색될 경우 성능을 향상시킬 수 있으나 *n* 이상의 행이 검색될 경우 성능을 저하시킬 수 있습니다.

행의 *크기*를 곱한 *n* 값이 통신 버퍼의 *크기*를 초과하는 경우 OPTIMIZE FOR절은 데이터 버퍼에 영향을 주지 않습니다. 통신 버퍼의 *크기*는 **rqrioblk** 또는 **aslheapsz** 구성 매개변수에 의해 정의됩니다.

isolation-clause



선택적 *isolation-clause*는 명령문이 실행되는 분리 레벨 및 특정 잠금 유형의 획득 여부를 지정합니다.

- RR = 반복 읽기
- RS = 읽기 안정성
- CS = 커서 안정성
- UR = 언커미트 읽기

명령문의 디폴트 분리 레벨은 명령문이 바운드되는 패키지의 분리 레벨입니다. 별칭이 *select-statement*에서 사용되어 DB2 제품군 및 Microsoft SQL Server 데이터 소스의 데이터에 액세스하는 경우, *isolation-clause*는 해당 명령문에 포함되어 명령문 분리 레벨을 지정할 수 있습니다. *isolation-clause*가 다른 데이터 소스를 액세스하는 명령문에 포함되면 지정된 분리 레벨이 무시됩니다. 페더레이티드 서버의 현재 분리 레벨은 데이터 소스로의 각 연결의 데이터 소스에 있는 해당 분리 레벨에 맵핑됩니다. 데이터 소스로의 연결이 완료된 후, 분리 레벨은 해당 연결의 지속기간 동안 변경될 수 없습니다.

lock-request-clause

선택적 *lock-request-clause*는 데이터베이스 관리 프로그램이 획득 및 보유하는 잠금 유형을 지정합니다.

SHARE

동시 프로세스는 데이터에 대해 SHARE 또는 UPDATE 잠금을 획득할 수 있습니다.

UPDATE

동시 프로세스는 데이터에 대해 SHARE 잠금을 획득할 수는 있지만 UPDATE 또는 EXCLUSIVE 잠금은 획득할 수 없습니다.

EXCLUSIVE

동시 프로세스는 데이터에 대해 잠금을 획득할 수 없습니다.

*lock-request-clause*는 하위 쿼리, SQL 함수 및 SQL 메소드 내의 항목을 포함하여 쿼리에서 필요로 하는 모든 기본 테이블 및 인덱스 스캔에 적용됩니다. 프로시저, 외부 함수 또는 외부 메소드가 수행하는 잠금에는 영향을 주지 않습니다. 명령문이 호출(직접 또는 간접적으로)한 모든 SQL 함수 또는 SQL 메소드는 INHERIT ISOLATION LEVEL WITH LOCK REQUEST로 작성되어야 합니다(SQLSTATE 42601). *lock-request-clause*는 트리거를 호출하거나 참조 무결성 점검을 요청하는 쿼리 수정에 사용할 수 없습니다(SQLSTATE 42601).

SELECT문의 예

예 1: EMPLOYEE 테이블에서 모든 컬럼과 행을 선택하십시오.

```
SELECT * FROM EMPLOYEE
```

예 2: PROJECT 테이블에서 프로젝트 이름(PROJNAME), 시작 날짜(PRSTDATE) 및 종료 날짜(PRENDATE)를 선택하십시오. 그런 다음, 가장 최신의 날짜가 먼저 나타나도록 종료 날짜별로 결과 테이블을 나열하십시오.

```
SELECT PROJNAME, PRSTDATE, PRENDATE
 FROM PROJECT
 ORDER BY PRENDATE DESC
```

예 3: EMPLOYEE 테이블에서 모든 부서에 대해 부서 번호(WORKDEPT) 및 평균 부서 월급(SALARY)을 선택하십시오. 그런 다음, 평균 부서 월급별로 오름차순에 따라서 결과 테이블을 배열하십시오.

```
SELECT WORKDEPT, AVG(SALARY)
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   ORDER BY 2
```

예 4: PROJET 테이블에서 시작 날짜(PRSTDATE) 및 종료 날짜(PRENDATE) 컬럼을 갱신하기 위해 C 프로그램에서 사용할 UP_CUR이라는 커서를 선언하십시오. 프로그램은 각 행에 대해서 이러한 값 모두를 프로젝트 번호(PROJNO)와 함께 수신해야 합니다.

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
          SELECT PROJNO, PRSTDATE, PRENDATE
             FROM PROJECT
            FOR UPDATE OF PRSTDATE, PRENDATE;
```

예 5: SAL+BONUS+COMM 표현식을 TOTAL_PAY로 이름을 지정합니다.

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY
  FROM EMPLOYEE
  ORDER BY TOTAL_PAY
```

예 6: 사원 번호와 영업대표부의 급여를 부서의 평균 급여와 함께 인원 수를 결정하십시오. 또한 가장 높은 평균 급여부터 부서의 평균 급여를 나열합니다.

이 경우에 공통 테이블 표현식을 사용하면 일반 뷰와 같이 DINFO 뷰를 작성하여 오버헤드를 줄입니다. 명령문 준비 중에는 뷰 카탈로그에 대한 액세스가 거부되고, 나머지 fullselect의 컨텍스트로 인하여 영업부에 대한 행만 뷰에서 고려되어야 합니다.

```
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
```

예 7: EMPLOYEE 및 PROJ 두 개 테이블에서 피고용인 SALLY를 새로 입사한 피고용인 GEORGE로 바꾸고 SALLY가 담당하던 모든 프로젝트를 GEORGE에게 할당한 다음, 갱신한 프로젝트 이름을 리턴하십시오.

```
WITH
     NEWEMP AS (SELECT EMPNO FROM NEW TABLE
                       (INSERT INTO EMPLOYEE(EMPNO, FIRSTNME)
                              VALUES(NEXT VALUE FOR EMPNO_SEQ, 'GEORGE'))),
     OLDEMP AS (SELECT EMPNO FROM EMPLOYEE WHERE FIRSTNME = 'SALLY'),
     UPPROJ AS (SELECT PROJNAME FROM NEW TABLE
                       (UPDATE PROJECT
                  SET RESPEMP = (SELECT EMPNO FROM NEWEMP)
                              WHERE RESPEMP = (SELECT EMPNO FROM OLDEMP))),
     DELEMP AS (SELECT EMPNO FROM OLD TABLE
                       (DELETE FROM EMPLOYEE
                 WHERE EMPNO = (SELECT EMPNO FROM OLDEMP)))
  SELECT PROJNAME FROM UPPROJ;
```

예 8: DEPT 테이블에서 데이터를 검색하십시오. 해당 데이터는 검색된 갱신으로 나중에 갱신되며 쿼리가 실행될 때 잠겨 있어야 합니다.

```
SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPT
  WHERE ADMRDEPT ='A00'
  FOR READ ONLY WITH RS USE AND KEEP EXCLUSIVE LOCKS
```

부록 A. SQL 및 XML 한계

다음 테이블은 특정 SQL 및 XML 한계를 설명합니다. 가장 제한적인 사례를 충실히 지키면 쉽게 이식할 수 있는 응용프로그램을 설계할 수 있습니다.

표 63는 목록을 바이트로 나열합니다. ID 작성 시 응용프로그램 코드로부터 데이터베이스 코드 페이지로 변환 후 이들 제한이 강제 실행됩니다. 데이터베이스에서 ID 검색 시 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 변환 후 제한이 강제 실행됩니다. 이들 프로세스 중 하나의 프로세스가 처리되는 동안 ID 길이 제한이 초과되면, 절단이 발생하거나 오류가 리턴됩니다.

데이터베이스의 코드 페이지 및 응용프로그램의 코드 페이지에 따라 문자 제한이 다를 수 있습니다. 예를 들어 UTF-8 문자의 너비가 1 - 4바이트 사이의 범위이기 때문에, 사용된 문자에 따라 128바이트가 제한인 유니코드 테이블의 ID에 대한 문자 제한이 32 - 128문자입니다. 데이터베이스 코드 페이지로 변환 후 이 테이블에 대한 제한보다 긴 이름의 ID를 작성하도록 시도하면, 오류가 리턴됩니다.

코드 페이지 변환이 발생한 다음 ID 이름을 저장하는 응용프로그램은 잠재적으로 늘어난 *크기*의 ID를 처리할 수 있어야 합니다. ID가 카탈로그에서 검색되면 응용프로그램 코드 페이지로 변환됩니다. 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 변환되면 ID가 테이블에 대한 바이트 제한 보다 더 길게 됩니다. 코드 페이지 변환 후 응용프로그램에서 선언한 호스트 변수가 전체 ID를 저장할 수 없으면 절단됩니다. 승인할 수 없으면 전체 ID 이름을 승인할 수 있도록 호스트 변수의 *크기*를 늘릴 수 있습니다.

동일한 규칙이 사용자 지정 코드 페이지로 변환되며 데이터를 검색하는 DB2 유틸리티에 적용됩니다. 익스포트와 같은 DB2 유틸리티는 데이터를 검색하며 사용자 지정 코드 페이지로 강제 변환되고(CODEPAGE 수정자 또는 **DB2CODEPAGE** 레지스트리 변수 사용), 코드 페이지 변환때문에 이 테이블에서 문서화된 제한 이상으로 ID가 확장되면, 오류가 리턴되거나 ID가 절단될 수 있습니다.

표 63. ID 길이 한계

설명	바이트 단위 최대값
별명 이름	128
속성 이름	128
감사 규정 이름	128
권한 부여 이름(1바이트 문자만 가능)	128
버퍼 풀 이름	18
컬럼 이름[2]	128
제한조건 이름	128

표 63. ID 길이 한계 (계속)

설명	바이트 단위 최대값
상관 이름	128
커서 이름	128
데이터 파티션 이름	128
데이터 소스 컬럼 이름	255
데이터 소스 인덱스 이름	128
데이터 소스 이름	128
데이터 소스 테이블 이름(*remote-table-name*)	128
데이터베이스 파티션 그룹 이름	128
데이터베이스 파티션 이름	128
이벤트 모니터 이름	128
외부 프로그램 이름	128
함수 맵핑 이름	128
그룹 이름	128
호스트 ID[1]	255
데이터 소스 사용자 ID(*remote-authorization-name*)	128
SQL 프로시저의 ID(조건 이름, 루프 ID의 경우, 레이블, 결과 세트 로케이터, 명령문 이름, 변수 이름)	128
인덱스 이름	128
인덱스 확장 이름	18
인덱스 스펙 이름	128
레이블 이름	128
이름 스페이스 단일 자원 ID(URI)	1000
별칭	128
패키지 이름	128
패키지 버전 ID	64
매개변수 이름	128
데이터 소스를 액세스하기 위한 암호	32
프로시저 이름	128
역할 이름	128
세이브포인트 이름	128
스키마 이름[2]	128
보안 레이블 구성요소 이름	128
보안 레이블 이름	128
보안 규정 이름	128
시퀀스 이름	128
서버(데이터베이스 별명) 이름	8
특정 이름	128
SQL 조건 이름	128
SQL 변수 이름	128
명령문 이름	128

표 63. ID 길이 한계 (계속)

설명	바이트 단위 최대값
테이블 이름	128
테이블 스페이스 이름	18
변환 그룹 이름	18
트리거 이름	128
트러스트된 컨텍스트 이름	128
유형 맵핑 이름	18
사용자 정의 함수(UDF) 이름	128
사용자 정의 메소드 이름	128
사용자 정의 유형 이름[2]	128
뷰 이름	128
랩퍼 이름	128
XML 요소 이름, 속성 이름 또는 접두어 이름	1000
XML 스키마 위치 단일 자원 ID(URI)	1000

주:

1. 개별 호스트 언어 컴파일러에는 변수 이름에 대해 더 제한적인 한계가 있을 수 있습니다.
2. SQLDA 구조는 사용자 정의 유형에 대해 30바이트 컬럼 이름, 18바이트 사용자 정의 유형 이름 및 8바이트 스키마 이름을 저장하는 데 제한이 있습니다. SQLDA가 DESCRIBE문에서 사용되기 때문에 컬럼이나 사용자 정의 유형 이름을 검색하는 DESCRIBE문을 사용하는 Embedded SQL 응용프로그램은 이들 제한을 준수해야 합니다.

표 64. 숫자 한계

설명	한계
최소 SMALLINT 값	-32 768
최대 SMALLINT 값	+32 767
최소 INTEGER 값	-2 147 483 648
최대 INTEGER 값	+2 147 483 647
최소 BIGINT 값	-9 223 372 036 854 775 808
최대 BIGINT 값	+9 223 372 036 854 775 807
최대 Decimal 정밀도	31
REAL 값에 대한 최대 지수 (E_{max})	38
최소 REAL 값	-3.402E+38
최대 REAL 값	+3.402E+38
REAL 값에 대한 최소 지수(E_{min})	-37
최소 양수 REAL 값	+1.175E-37
최대 음수 REAL 값	-1.175E-37
DOUBLE 값에 대한 최대 지수 (E_{max})	308
최소 DOUBLE 값	-1.79769E+308
최대 DOUBLE 값	+1.79769E+308

표 64. 숫자 한계 (계속)

설명	한계
DOUBLE 값에 대한 최소 지수 (E_{min})	-307
최소 양수 DOUBLE 값	+2.225E-307
최대 음수 DOUBLE 값	-2.225E-307
DECFLOAT(16) 값에 대한 최대 지수(E_{max})	384
가장 작은 DECFLOAT(16) 값[1]	-9.999999999999999E+384
가장 큰 DECFLOAT(16) 값	9.999999999999999E+384
DECFLOAT(16) 값에 대한 최소 지수(E_{min})	-383
가장 작은 양수 DECFLOAT(16) 값	1.000000000000000E-383
가장 큰 음수 DECFLOAT(16) 값	-1.000000000000000E-383
DECFLOAT(34) 값에 대한 최대 지수(E_{max})	6144
가장 작은 DECFLOAT(34) 값[1]	-9.999999999999999999999999999999999E+6144
가장 큰 DECFLOAT(34) 값	9.999999999999999999999999999999999E+6144
DECFLOAT(34) 값에 대한 최소 지수(E_{min})	-6143
가장 작은 양수 DECFLOAT(34) 값	1.000000000000000000000000000000000E-6143
가장 큰 음수 DECFLOAT(34) 값	-1.000000000000000000000000000000000E-6143

주:

1. 일반 10진수 부동 소수점의 제한이 있습니다. 유효한 10진수 부동 소수점 값은 특수 값 NAN, -NAN, SNAN, -SNAN, INFINITY 및 -INFINITY를 포함합니다. 또한 유효한 값은 subnormal 수를 포함합니다.

   subnormal 숫자는 조정된 지수가 E_{min} 미만인 0이 아닌 숫자입니다. subnormal 숫자의 경우 지수의 최소 값은 E_{tiny}라는 E_{min} - (*precision*-1)이며, 여기서 *precision*은 사용 중인 정밀도(16 또는 34)입니다. 즉 subnormal 숫자는 0에 가까운 숫자 범위를 DECFLOAT(16) 또는 DECFLOAT(34)에 대한 *크기*의 15나 33 순서로 각각 확장합니다. Subnormal 숫자는 subnormal 숫자의 최대 자릿수가 사용할 수 있는 정밀도(16 또는 34)보다 작으므로 일반 숫자와 다릅니다. 10진수 부동 소수점은 일반 숫자를 표시할 수 있는 같은 정밀도 subnormal 숫자를 표시할 수 없습니다. DECFLOAT(34)에 대한 가장 작은 양수의 subnormal 숫자는 $1x10^{-6176}$이며, 하나의 숫자만을 포함하는 데 반해, DECFLOAT(34)의 가장 작은 양수의 일반 숫자는 $1.000000000000000000000000000000000x10^{-6143}$이며, 34개의 숫자를 포함합니다. DECFLOAT(16)에 대한 가장 작은 양수의 subnormal 숫자는 $1x10^{-398}$입니다.

표 65. 문자열 한계

설명	한계
CHAR 최대 길이(바이트)	254
VARCHAR 최대 길이(바이트)	32 672
LONG VARCHAR 최대 길이(바이트)[1]	32 700
CLOB 최대 길이(바이트)	2 147 483 647
연속 XML의 최대 길이(바이트)	2 147 483 647
GRAPHIC의 길이(2바이트 문자)	127
VARGRAPHIC의 최대 길이(2바이트 문자)	16 336
LONG VARGRAPHIC의 최대 길이(2바이트 문자)[1]	16 350
DBCLOB의 최대 길이(2바이트 문자)	1 073 741 823
BLOB 최대 길이(바이트)	2 147 483 647
문자 상수 최대 길이	32 672
그래픽 상수 최대 길이	16 336
병합 문자열 최대 길이	2 147 483 647
병합 그래픽 문자열 최대 길이	1 073 741 823
병합 실행 파일 문자열 최대 길이	2 147 483 647
16진 상수 최대 자릿수	32 672
런타임시(GB에서) 구조화된 유형 컬럼 오브젝트의 최대 인스턴스	1
카탈로그 주석 최대 크기(바이트)	254
주: 1. LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다.	

표 66. XML 한계

설명	한계
XML 문서의 최대 용량(레벨)	125
XML 스키마 문서의 최대 크기(바이트)	31 457 280

표 67. 날짜 시간 한계

설명	한계
최소 DATE 값	0001-01-01
최대 DATE 값	9999-12-31
최소 TIME 값	00:00:00
최대 TIME 값	24:00:00
최소 TIMESTAMP 값	0001-01-01-00.00.00.000000000000
최대 TIMESTAMP 값	9999-12-31-24.00.00.000000000000
최소 시간소인 정밀도	0
최대 시간소인 정밀도	12

표 68. 데이터베이스 관리 프로그램 한계

설명	한계
응용프로그램	
프리컴파일된 프로그램의 최대 호스트 변수 선언 수[3]	스토리지
호스트 변수 값의 최대 길이(바이트)	2 147 483 647
프로그램의 선언된 커서의 최대수	스토리지
작업 단위(UOW)에서 변경된 최대 행 수	스토리지
한 번에 열리는 최대 커서 수	스토리지
DB2 클라이언트 내의 프로세스당 최대 연결 수	512
트랜잭션에서 동시에 열리는 최대 LOB 로케이터 수	4 194 304
SQLDA의 최대 *크기*(바이트)	스토리지
준비된 명령문의 최대 수	스토리지
버퍼 풀	
32비트 릴리스용 버퍼 풀의 최대 NPAGES	1 048 576
64비트 릴리스용 버퍼 풀의 최대 NPAGES	2 147 483 647
모든 버퍼 풀 슬롯의 최대 전체 *크기*(4K)	2 147 483 646
동시성	
서버의 최대 동시 사용자 수[4]	64 000
인스턴스당 최대 동시 사용자 수	64 000
데이터베이스당 최대 동시 응용프로그램 수	60 000
동시 사용되는 인스턴스당 최대 데이터베이스 수	256
제한조건	
테이블의 최대 제한조건 수	스토리지
UNIQUE 인덱스를 통해 지원되는 UNIQUE 제한조건의 최대 컬럼 수	64
UNIQUE 인덱스를 통해 지원되는 UNIQUE 제한조건에 있는 컬럼의 최대 조인 길이(바이트)[9]	8192
외부 키의 최대 참조 컬럼 수	64
외부 키에 있는 참조 컬럼의 최대 조인 길이(바이트)[9]	8192
점검 제한조건 스펙의 최대 길이(바이트)	65 535
데이터베이스	
최대 데이터베이스 파티션 번호	999
인덱스	
최대 테이블 인덱스 수	32 767 또는 스토리지
최대 인덱스 키 컬럼 수	64
오버헤드를 모두 포함한 인덱스 키의 최대 길이[7] [9]	*indexpagesize*/4
변수 인덱스 키 파트의 최대 길이(바이트)[8]	1022 또는 스토리지
SMS 테이블 스페이스의 데이터베이스 파티션당 인덱스의 최대 *크기*(GB)[7]	16 384
일반 DMS 테이블 스페이스의 데이터베이스 파티션당 인덱스의 최대 *크기*(GB)[7]	512
대형 DMS 테이블 스페이스의 데이터베이스 파티션당 인덱스의 최대 *크기*(GB)[7]	16 384

표 68. 데이터베이스 관리 프로그램 한계 (계속)

설명	한계
데이터베이스 파티션당 XML 데이터에 있는 인덱스의 최대 크기 (TB)	2
XML 데이터에 있는 인덱스용 변수 인덱스 키 파트의 최대 길이(바이트)[7]	*pagesize*/4 - 207
로그 레코드	
최대 로그 시퀀스 번호	17 984 000 000 000 000 000
모니터링	
동시 사용 중인 이벤트 모니터의 최대 수	128
DB2 데이터베이스 파티션 기능(DPF)을 사용할 경우의 동시 활성 GLOBAL 이벤트 모니터의 최대 수	32
루틴	
프로시저의 최대 매개변수 수	32 767
커서 값 컨스트럭터의 최대 매개변수 수	32,767
사용자 정의 함수의 최대 매개변수 수	90
루틴의 최대 중첩 레벨 수	64
SQL 경로의 최대 스키마 수	64
SQL 경로의 최대 길이(바이트)	2048
보안	
유형 세트 또는 트리의 보안 레이블 구성요소의 최대 요소 수	64
유형 배열의 보안 레이블 구성요소의 최대 요소 수	65 535
보안 규정의 최대 보안 레이블 구성요소 수	16
SQL	
SQL문의 최대 총 길이(바이트)	2 097 152
SQL문 또는 뷰에서 참조되는 최대 테이블 수	스토리지
SQL문의 최대 호스트 변수 참조 수	32 767
명령문의 최대 상수 수	스토리지
선택 목록의 최대 요소 수[7]	1012
WHERE 또는 HAVING절의 최대 술어 수	스토리지
GROUP BY절의 최대 컬럼 수[7]	1012
GROUP BY절의 최대 총 컬럼 길이(바이트)[7]	32 677
ORDER BY절의 최대 컬럼 수[7]	1012
ORDER BY절의 최대 총 컬럼 길이(바이트)[7]	32 677
서브쿼리 중첩의 최대 레벨	스토리지
단일 명령문의 최대 서브쿼리 수	스토리지
삽입 조작에 있는 최대 값의 수[7]	1012
단일 갱신 조작에 있는 최대 SET절의 수[7]	1012
테이블 및 뷰	
테이블의 최대 컬럼 수[7]	1012
뷰의 최대 컬럼 수[1]	5000
별칭으로 참조되는 데이터 소스 테이블이나 뷰에서의 최대 컬럼 수	5000

표 68. 데이터베이스 관리 프로그램 한계 (계속)

설명	한계
분산 키의 최대 컬럼 수[5]	500
오버헤드를 모두 포함한 행의 최대 길이[2] [7]	32 677
데이터베이스 파티션당 파티션되지 않은 테이블의 최대 행 수	128×10^{10}
데이터베이스 파티션당 데이터 파티션의 최대 행 수	128×10^{10}
일반 테이블 스페이스의 데이터베이스 파티션당 테이블의 최대 크기 (GB)[3] [7]	512
대형 DMS 테이블 스페이스의 데이터베이스 파티션당 테이블의 최대 크기(GB)[7]	16 384
단일 테이블에 대한 최대 데이터 파티션 수	32 767
최대 테이블 파티션 컬럼 수	16
사용자 정의 행 유형의 최대 필드 수	1012
테이블 스페이스	
LOB 오브젝트의 최대 크기(TB)	4
LF 오브젝트의 최대 크기(TB)	2
데이터베이스 내의 최대 테이블 스페이스 수	32 768
SMS 테이블 스페이스의 최대 테이블 수	65 534
일반 DMS 테이블 스페이스의 최대 크기(GB)[3] [7]	512
대형 DMS 테이블 스페이스의 최대 크기(TB)[3] [7]	64
임시 DMS 테이블 스페이스의 최대 크기(TB) [3] [7]	64
DMS 테이블 스페이스의 최대 테이블 오브젝트 수[6]	51 000
자동 스토리지 데이터베이스의 최대 스토리지 경로 수	128
자동 스토리지 데이터베이스와 연결된 스토리지 경로 최대 길이(바이트)	175
트리거	
연쇄 트리거의 최대 런타임 용량	16
사용자 정의 유형	
구조화된 유형의 최대 속성 수	4082

표 68. 데이터베이스 관리 프로그램 한계 (계속)

설명	한계
주:	
1. 이 최대값은 CREATE VIEW문에서 조인을 사용하여 얻을 수 있습니다. 이러한 뷰에서 선택하면 선택 목록의 대부분 요소가 제한됩니다.	
2. BLOB, CLOB, LONG VARCHAR, DBCLOB 및 LONG VARGRAPHIC 컬럼의 실제 데이터는 이 계수에 포함되지 않습니다. 그러나 해당 데이터의 위치에 대한 정보는 행에서 일부 스페이스를 차지합니다.	
3. 표시된 수는 구조적인 한계 및 근사값입니다. 실제 한계는 이것보다 작을 수 있습니다.	
4. 실제 값은 **max_connections** 및 **max_coordagents** 데이터베이스 관리 프로그램 구성 매개변수에서 제어됩니다.	
5. 이 한계는 구조적인 한계입니다. 인덱스 키의 최대 컬럼에 대한 제한이 실질적인 한계로 사용되어야 합니다.	
6. 테이블 오브젝트는 데이터, 인덱스, LONG VARCHAR 또는 VARGRAPHIC 컬럼 및 LOB 컬럼을 포함합니다. 테이블 데이터와 동일한 테이블 스페이스에 있는 테이블 오브젝트의 한계에는 여분이 포함되어 있지 않습니다. 단, 테이블 데이터와 다른 테이블 스페이스에 있는 각 테이블 오브젝트는 테이블 오브젝트가 있는 테이블 스페이스의 테이블마다 있는 각 테이블 오브젝트 유형의 한계에 1을 더합니다.	
7. 페이지 크기 특정 값에 대해서는 표 69 페이지를 참조하십시오.	
8. 이는 모든 오버헤드를 포함하여 인덱스 키의 최대 길이(바이트)로만 제한됩니다. 인덱스 키 파트의 수가 증가함에 따라 각 키 파트의 최대 길이는 감소됩니다.	
9. 인덱스 옵션에 따라 최대 길이가 더 짧아질 수 있습니다.	

표 69. 데이터베이스 관리 프로그램 페이지 크기 특정 한계

설명	4K 페이지 크기 한계	8K 페이지 크기 한계	16K 페이지 크기 한계	32K 페이지 크기 한계
테이블의 최대 컬럼 수	500	1012	1012	1012
오버헤드를 모두 포함한 행의 최대 길이	4005	8101	16 293	32 677
일반 테이블 스페이스의 데이터베이스 파티션당 테이블의 최대 크기(GB)	64	128	256	512
대형 DMS 테이블 스페이스의 데이터베이스 파티션당 테이블의 최대 크기(GB)	2048	4096	8192	16 384
오버헤드를 모두 포함한 인덱스 키의 최대 길이(바이트)	1024	2048	4096	8192
SMS 테이블 스페이스의 데이터베이스 파티션당 인덱스의 최대 크기(GB)	2048	4096	8192	16 384
일반 DMS 테이블 스페이스의 데이터베이스 파티션당 인덱스의 최대 크기(GB)	64	128	256	512

표 69. 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
대형 DMS 테이블 스페이스의 데이터베이스 파티션당 인덱스의 최대 크기(GB)	2048	4096	8192	16 384
데이터베이스 파티션당 XML 데이터에 있는 인덱스의 최대 크기 (TB)	2	2	2	2
정규 DMS 테이블 스페이스의 최대 크기(GB)	64	128	256	512
대형 DMS 테이블 스페이스의 최대 크기(TB)	8	14	32	64
임시 DMS 테이블 스페이스의 최대 크기(TB)	8	14	32	64
선택 목록의 최대 요소 수	500	1012	1012	1012
GROUP BY절의 최대 컬럼수	500	1012	1012	1012
GROUP BY절의 최대 컬럼 합계 길이(바이트)	4005	8101	16 293	32 677
ORDER BY절의 최대 컬럼 수	500	1012	1012	1012
ORDER BY절의 최대 컬럼 합계 길이(바이트)	4005	8101	16 293	32 677
삽입 조작에 있는 최대 값의 수	500	1012	1012	1012
단일 갱신 조작에 있는 최대 SET절의 수	500	1012	1012	1012

부록 B. SQLCA(SQL 통신 영역)

SQLCA는 모든 SQL문의 실행 끝에 갱신되는 변수의 콜렉션입니다. 실행 가능한 SQL 문을 포함하며 옵션 LANGLEVEL SAA1(디폴트) 또는 MIA로 프리컴파일된 프로그램은 다중 스레드 응용프로그램에서 스레드당 하나의 SQLCA를 가짐으로써 둘 이상의 SQLCA가 가능한 경우에도 정확히 하나의 SQLCA를 제공해야 합니다.

프로그램이 LANGLEVEL SQL92E 옵션으로 프리컴파일되면 SQLCODE 또는 SQLSTATE 변수를 SQL 선언 섹션에 선언할 수 있거나 SQLCODE 변수를 프로그램 어딘가에서 선언할 수 있습니다.

LANGLEVEL SQL92E를 사용할 때에는 SQLCA를 제공하면 안됩니다. REXX를 제외한 모든 언어에서 SQL INCLUDE문을 SQL의 선언을 제공하는 데 사용할 수 있습니다. SQLCA는 REXX에서 자동으로 제공됩니다.

명령행 처리기를 통해 각 명령이 실행된 후 SQLCA를 표시하려면 db2 -a 명령을 발행하십시오. 그러면 SQLCA가 서브쿼리 명령에 대한 출력의 일부로 제공됩니다. SQLCA는 db2diag 로그 파일에서 덤프되기도 합니다.

SQLCA 필드 설명

표 70. SQLCA 필드. 표시된 필드 이름은 INCLUDE문을 통해 확보된 SQLCA에 있는 필드 이름임.

이름	데이터 유형	필드 값
sqlcaid	CHAR(8)	'SQLCA'를 포함하는 스토리지 덤프용 ″eye catcher″. SQL 프로시저 본문 구문 분석 시 행 번호 정보가 리턴되는 경우 6번째 바이트는 'L'입니다.
sqlcabc	INTEGER	SQLCA의 길이(136)를 포함합니다.
sqlcode	INTEGER	SQL 리턴 코드를 포함합니다. **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(0과 70 사이의 범위). 0은 *sqlerrmc의 값이* 사용되지 않았음을 의미합니다.

표 70. SQLCA 필드 (계속). 표시된 필드 이름은 INCLUDE문을 통해 확보된 SQLCA에 있는 필드 이름임.

이름	데이터 유형	필드 값
sqlerrmc	VARCHAR(70)	오류 상태의 설명에서 변수가 대체되고 X'FF'로 구분되는 하나 이상의 토큰을 포함합니다. 이 필드는 성공적인 연결이 완료될 때도 사용됩니다. NOT ATOMIC 복합 SQL문이 발행될 때 최대 7개의 오류에 대한 정보를 포함하게 됩니다. 마지막 토큰 다음에 X'FF'가 옵니다. *sqlerrml* 값의 마지막에 X'FF'가 포함됩니다.
sqlerrp	CHAR(8)	제품의 수정 레벨과 버전, 릴리스를 나타내는 5개의 영숫자가 따라오며 제품을 나타내는 세 개의 문자 ID로 시작합니다. A에서 Z까지의 문자는 9 이상의 수정 레벨을 표시합니다. A는 수정 레벨 10을 B는 수정 레벨 11을 표시하는 식으로 진행됩니다. 예를 들어 SQL0907C는 DB2 버전 9, 릴리스 7, 수정 레벨 12를 표시합니다. SQLCODE가 오류 상태를 나타내는 경우 이 필드는 오류를 리턴한 모듈을 식별합니다. 이 필드는 성공적인 연결이 완료될 때도 사용됩니다.
sqlerrd	ARRAY	진단 정보를 제공하는 6개의 INTEGER 변수. 파티션된 데이터베이스에서 sqlerrd(6)에 대한 경우를 제외하면 오류가 없을 경우에 이 값은 일반적으로 비어 있습니다.
sqlerrd(1)	INTEGER	연결 호출에 성공한 경우 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환되면 혼합 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. SQL 프로시저로부터 리턴 시 SQL 프로시저의 리턴 상태 값을 포함합니다.
sqlerrd(2)	INTEGER	연결 호출에 성공한 경우 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 변환되면 혼합 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 오류가 발생한 NOT ATOMIC 복합 SQL문의 결과인 경우, 값은 실패한 명령문 수로 설정됩니다.

표 70. SQLCA 필드 (계속). 표시된 필드 이름은 INCLUDE문을 통해 확보된 SQLCA에 있는 필드 이름임.

이름	데이터 유형	필드 값
sqlerrd(3)	INTEGER	PREPARE가 호출되어 성공할 경우 리턴될 행 수의 추정치를 포함합니다. INSERT, UPDATE, DELETE 또는 MERGE 다음에 조작에 규정된 실제 행 수를 포함합니다. TRUNCATE문의 경우 값은 -1이 됩니다. 복합 SQL을 호출할 경우 모든 부속 명령문 행의 누적을 포함합니다. CONNECT를 호출할 경우 데이터베이스가 갱신되면 1을 포함하고 데이터베이스가 읽기 전용이면 2를 포함합니다. OPEN문을 호출한 경우 커서에 SQL 데이터 변경 명령문이 있으면 이 필드는 임베디드 삽입, 갱신, 삭제 또는 병합 조작이 가능한 행 수의 합을 포함합니다. SQL 프로시저에 대한 CREATE PROCEDURE가 호출되고 SQL 프로시저 본문 구문 분석 시 오류가 발견되는 경우 오류가 발견된 행 번호를 포함합니다. sqlcaid의 6번째 바이트는 유효한 행 번호가 될 'L'이어야 합니다.
sqlerrd(4)	INTEGER	PREPARE가 호출되어 성공할 경우 명령문 처리에 필요한 자원의 비교 상대적 추정치를 포함합니다. 복합 SQL을 호출할 경우 성공적인 부속 명령문의 수 계산을 포함합니다. CONNECT를 호출한 경우, 다운 레벨 클라이언트에서 1단계 커미트에 대해 0을, 1단계 커미트에 대해 1을, 1단계와 읽기 전용 커미트에 대해 2를, 2단계 커미트에 대해 3을 포함합니다.
sqlerrd(5)	INTEGER	다음 두 경우의 결과로서 삭제, 삽입, 갱신된 행의 총 수를 포함합니다. • 성공적인 삭제 조작 후 제한조건의 강제 시행 • 활성화된 트리거에서 트리거 SQL문의 처리 복합 SQL을 호출할 경우, 모든 부속 명령문 행 수의 누적을 포함합니다. 일부 경우에 필드는 내부 오류 포인터인 음수 값을 포함합니다. CONNECT가 호출된 경우, 다음의 인증 유형 값을 포함합니다. 0: 서버 인증, 1: 클라이언트 인증, 2: DB2 Connect를 사용한 인증, 4: SERVER_ENCRYPT 인증, 5: 암호화된 DB2 Connect를 사용하는 인증, 7: KERBEROS 인증, 9: GSSPLUGIN 인증, 11: DATA_ENCRYPT 인증, 255: 지정되지 않은 인증.
sqlerrd(6)	INTEGER	파티션된 데이터베이스에 대해 오류 또는 경고를 발견한 데이터베이스 파티션의 파티션 번호를 포함합니다. 오류 또는 경고를 발견하지 않은 경우 해당 필드는 코디네이터 파티션의 파티션 번호를 포함합니다. 이 필드에 있는 번호는 db2nodes.cfg 파일의 데이터베이스 파티션에 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 공백인 경우 공백. 최소한 하나의 다른 표시기가 공백이 아니면 'W'를 포함합니다.

표 70. SQLCA 필드 (계속). 표시된 필드 이름은 INCLUDE문을 통해 확보된 SQLCA에 있는 필드 이름임.

이름	데이터 유형	필드 값
sqlwarn1	CHAR(1)	호스트 변수에 지정될 때 문자열 컬럼의 값이 절단되면 'W'를 포함합니다. 널(NULL) 종료자가 절단되면 'N'을 포함합니다. CONNECT 또는 ATTACH가 성공하고 연결에 대한 권한 부여 이름이 8바이트보다 길면 'A'를 포함합니다. sqlerrd(4)에 저장된 PREPARE문 관련 비용 평가가 INTEGER에 저장될 수 있는 값을 초과하였거나 1 미만인 경우 및 CURRENT EXPLAIN MODE 또는 CURRENT EXPLAIN SNAPSHOT 특수 레지스터가 NO가 아닌 값으로 설정된 경우 'P'를 포함합니다.
sqlwarn2	CHAR(1)	널(NULL) 값이 집계 함수의 인수에서 제거되면 'W'를 포함합니다. [a] CONNECT가 호출되어 성공할 경우 데이터베이스가 Quiesce 상태이면 'D'를, 인스턴스가 Quiesce 상태이면 'I'를 포함합니다.
sqlwarn3	CHAR(1)	컬럼 수가 호스트 변수의 수와 일치하지 않으면 'W'를 포함합니다. ASSOCIATE LOCATORS문에 지정된 결과 세트 로케이터의 수가 프로시저가 리턴한 결과 세트 수보다 작은 경우 'Z'를 포함합니다.
sqlwarn4	CHAR(1)	준비된 UPDATE 또는 DELETE문이 WHERE절을 포함하지 않으면 'W'를 포함합니다.
sqlwarn5	CHAR(1)	SQL문 실행 동안 오류가 허용되면 'E'를 포함합니다.
sqlwarn6	CHAR(1)	날짜 계산의 결과가 불가능한 날짜를 피하도록 조정되면 'W'를 포함합니다.
sqlwarn7	CHAR(1)	나중에 사용하기 위해 예약됩니다. CONNECT가 호출되어 성공할 경우, **dyn_query_mgmt** 데이터베이스 구성 매개변수가 사용 가능하게 되면 'E'를 포함합니다.
sqlwarn8	CHAR(1)	변환될 수 없는 문자가 대체 문자로 대체되면 'W'를 포함합니다. 트러스트된 연결 설정 시도가 실패하면 'Y'를 포함합니다.
sqlwarn9	CHAR(1)	오류가 있는 산술 연산식이 집계 함수 처리 중에 무시되면 'W'를 포함합니다.
sqlwarn10	CHAR(1)	SQLCA의 필드 중 하나에서 문자 데이터 값을 변환할 때 변환 오류가 있으면 'W'를 포함합니다.
sqlstate	CHAR(5)	가장 최근에 실행된 SQL문의 결과를 나타내는 리턴 코드

[a] 결과가 널(NULL) 값 제거에 종속되지 않으므로 널(NULL) 값이 제거된 경우에도 일부 함수는 SQLWARN2를 W로 설정할 수 없습니다.

오류 보고

오류 보고의 순서는 다음과 같습니다.

1. 심각한 오류 상태는 항상 보고됩니다. 심각한 오류가 보고될 때 SQLCA에 추가되지 않습니다.
2. 심각한 오류가 발생하는 경우 교착 상태 오류가 다른 오류에 우선합니다.
3. 다른 모든 오류의 경우 첫 번째 음수 SQL 코드에 대한 SQLCA가 리턴됩니다.

4. 음수 SQL 코드가 삭제되지 않은 경우 첫 번째 경고(즉, 양수 SQL 코드)에 대한 SQLCA가 리턴됩니다.

   파티션된 데이터베이스 시스템에서 이러한 규칙에 대한 예외는 데이터 조작 연산이 하나의 데이터베이스 파티션에는 비어 있지만 다른 데이터베이스 파티션에는 데이터가 있는 테이블에 대해 호출될 경우에 발생합니다. 모든 데이터베이스 파티션에 테이블이 비어 있거나 UPDATE문의 WHERE 절을 충족시키는 행이 없으므로 SQLCODE +100은 모든 데이터베이스 파티션의 에이전트가 SQL0100W를 리턴할 경우 응용프로그램으로만 리턴됩니다.

파티션된 데이터베이스 시스템에서 SQLCA 사용

파티션된 데이터베이스 시스템에서 하나의 SQL문이 다른 데이터베이스 파티션의 수많은 에이전트에 의해 실행될 수 있으며 각 에이전트는 다른 오류 또는 경고에 대해 다른 SQLCA를 리턴할 수 있습니다. 코디네이터 에이전트는 또한 고유의 SQLCA를 갖습니다.

응용프로그램에 일관성 있는 뷰를 제공하기 위해 모든 SQLCA 값은 하나의 구조로 병합되며 SQLCA 필드는 전역 계수를 나타냅니다.

- 모든 오류와 경고에 대해 *sqlwarn* 필드는 모든 에이전트에서 수신한 경고 플래그를 포함합니다.
- 행 계수를 나타내는 *sqlerrd* 필드에 있는 값은 모든 에이전트로부터의 누적입니다.

트리거 SQL문 처리시 오류가 발생할 때마다 SQLSTATE 09000이 리턴될 수 없다는 점에 유의하십시오.

부록 C. SQLDA(SQL 디스크립터 영역)

SQLDA는 SQL DESCRIBE문 실행에 필요한 변수의 콜렉션입니다. SQLDA 변수는 PREPARE, OPEN, FETCH 및 EXECUTE문에서 사용될 수 있는 옵션입니다. SQLDA는 동적 SQL과 통신하여, DESCRIBE문에 사용되고, 호스트 변수 주소로 수정된 후 FETCH 또는 EXECUTE문에서 재사용될 수 있습니다.

SQLDA는 모든 언어를 지원하지만 사전 정의된 선언은 C, REXX, FORTRAN 및 COBOL에 대해서만 제공됩니다.

SQLDA에 있는 정보 의미는 사용 방식에 따라 다릅니다. PREPARE 및 DESCRIBE에서 SQLDA는 준비된 명령문에 대한 정보를 응용프로그램에 제공합니다. OPEN, EXECUTE 및 FETCH에서 SQLDA는 호스트 변수를 설명합니다.

DESCRIBE 및 PREPARE에서 설명된 컬럼 중 하나가 LOB 유형(LOB 로케이터 및 파일 참조 변수에는 두 배의 SQLDA가 필요하지는 않음), 참조 유형 또는 사용자 정의 유형인 경우, 전체 SQLDA의 SQLVAR 항목 수는 두 배가 됩니다. 예를 들면, 다음과 같습니다.

- 3 VARCHAR 컬럼과 1 INTEGER 컬럼으로 테이블을 설명할 때 4 SQLVAR 항목이 있게 됩니다.
- 2 VARCHAR 컬럼, 1 CLOB 컬럼 및 1 정수 컬럼으로 테이블을 설명할 때 8 SQLVAR 항목이 있게 됩니다.

EXECUTE, FETCH 및 OPEN에서 설명된 변수 중 하나가 LOB 유형(LOB 로케이터 및 파일 참조 변수에는 두 배의 SQLDA가 필요하지는 않음) 또는 구조화된 유형인 경우, 전체 SQLDA의 SQLVAR 항목 수는 두 배가 됩니다. (이 경우, 데이터베이스에는 두 배 항목의 추가 정보가 필요하지 않으므로 구별 유형 및 참조 유형은 관련되지 않습니다. 배열, 커서 및 행 유형은 EXECUTE, FETCH 및 OPEN문의 SQLDA 변수로 지원되지 않습니다.)

SQLDA 필드 설명

SQLDA는 네 변수와 SQLVAR로 이름 지정되는 일련의 변수 어커런스 수로 구성되어 있습니다. OPEN, FETCH 및 EXECUTE에서 SQLVAR의 각 어커런스는 호스트 변수를 설명합니다. DESCRIBE 및 PREPARE에서 SQLVAR의 각 어커런스는 결과 테이블 또는 매개변수 표시문자의 컬럼을 설명합니다. SQLVAR 항목에는 두 종류가 있습니다.

- **기본 SQLVAR:** 세 가지 항목이 항상 존재합니다. 여기에는 데이터 유형 코드, 길이 속성, 컬럼 이름, 호스트 변수 주소 및 표시기 변수 주소와 같은 호스트 변수, 매개변수 표시문자 또는 컬럼에 대한 기본 정보가 포함됩니다.

- **2차 SQLVAR:** 이들 항목은 위에 설명된 규칙에 따라 SQLVAR 항목의 수가 두 배로 되는 경우에만 제시됩니다. 사용자 정의 유형(참조 유형 제외)의 경우에는 사용자 정의 유형 이름이 포함됩니다. 참조 유형의 경우에는 참조의 목표 유형이 포함됩니다. LOB의 경우에는 호스트 변수의 길이 속성 및 실제 길이를 포함하는 버퍼에 대한 포인터가 포함됩니다. (구별 유형과 LOB 정보는 겹치지 않으므로 구별 유형은 DESCRIBE의 SQLVAR 항목 수를 세 배로 만들지 않고 LOB을 따를 수 있습니다.) 로케이터 또는 파일 참조 변수가 LOB을 나타내는 데 사용되는 경우 이들 항목이 필요하지는 않습니다.

두 종류의 항목이 들어 있는 SQLDA에서 기본 SQLVAR은 2차 SQLVAR 블록 앞에 있는 블록에 있습니다. 각각의 경우 다수의 2차 SQLVAR 항목이 사용되지 않는 경우에도 항목 수는 SQLD의 값과 같습니다.

SQLVAR 항목이 DESCRIBE에 의해 설정되는 환경은 842 페이지의 『SQLDA에서 DESCRIBE의 영향』의 내용을 참조하십시오.

SQLDA 헤더에서 필드

표 71. SQLDA 헤더 내의 필드

C 이름	SQL 데이터 유형	DESCRIBE 및 PREPARE에서의 사용 (SQLN을 제외한 데이터베이스 관리 프로그램에서 설정)	FETCH, OPEN 및 EXECUTE에서의 사용 (명령문을 실행하기 전에 응용프로그램에서 설정)
sqldaid	CHAR(8)	이 필드의 7번째 바이트는 SQLDOUBLED라는 플래그 바이트입니다. 데이터베이스 관리 프로그램은 컬럼마다 두 개의 SQLAR 항목이 작성된 경우에 SQLDOUBLED를 문자 '2'로 설정합니다. 그 외에는 공백으로 설정됩니다 (ASCII로는 X'20', EBDIC으로는 X'40'). SQLDOUBLED의 설정 시기에 대해서는 842 페이지의 『SQLDA에서 DESCRIBE의 영향』의 내용을 참조하십시오.	이 필드의 7번째 바이트는 SQLVAR의 수가 두 배가 될 때 사용됩니다. SQLDOUBLED가 그 이름입니다. 설명되는 호스트 변수가 구조화된 유형, BLOB, CLOB 또는 DBCLOB인 경우, 7번째 바이트를 문자 '2'로 설정해야 합니다. 그렇지 않을 경우에는 임의 문자로 설정할 수 있으나 공백 사용을 권장합니다.
sqldabc	INTEGER	32비트의 경우 SQLDA 길이는 SQLN*44+16입니다. 64비트의 경우 SQLDA 길이는 SQLN*56+16입니다.	32비트의 경우 SQLDA 길이는 >= SQLN*44+16입니다. 64비트의 경우 SQLDA 길이는 >= SQLN*56+16입니다.
sqln	SMALLINT	데이터베이스 관리 프로그램에서 변경하지 않습니다. DESCRIBE문이 실행되기 전에 0 이상의 값으로 설정되어야 합니다. SQLVAR의 총 어커런스 수를 표시합니다.	SQLDA에 제공된 SQLDA의 총 어커런스 수. SQLN은 0 이상의 값으로 설정되어야 합니다.
sqld	SMALLINT	데이터베이스 관리 프로그램에서 결과 테이블의 컬럼 수 또는 매개변수 표시문자 수로 설정합니다.	SQLVAR 어커런스로 기술되는 호스트 변수의 수.

기본 SQLVAR의 어커런스 내의 필드

표 72. 기본 SQLVAR 내의 필드

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN 및 EXECUTE에서의 사용
sqltype	SMALLINT	컬럼 데이터 유형 또는 매개변수 표시문자 및 널(NULL) 포함 여부를 표시합니다. (매개변수 표시문자는 항상 널(NULL) 입력 가능한 것으로 간주됩니다.) 844 페이지의 표 74에는 허용되는 값과 그 의미가 나열되어 있습니다. 구별, 배열, 커서, 행 또는 참조 유형의 경우 기본 유형의 데이터 유형이 이 필드에 위치함에 유의하십시오. 구조화된 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 동일함 날짜 시간 값에 대한 호스트 변수는 문자열 변수여야 합니다. FETCH의 경우 날짜 시간 유형 코드는 고정 길이의 문자열을 의미합니다. sqltype이 짝수 값인 경우 sqlind 필드는 무시됩니다.
sqllen	SMALLINT	컬럼 또는 매개변수 표시문자의 길이 속성. 날짜 시간 컬럼 및 매개변수 표시문자의 경우 값의 문자열 표현 길이. 844 페이지의 표 74의 내용을 참조하십시오. 대형 오브젝트(LOB) 문자열에 대한 값은 길이 속성이 2바이트 정수 정도로 작은 경우에도 0으로 설정됨에 유의하십시오.	호스트 변수의 길이 속성. 844 페이지의 표 74의 내용을 참조하십시오. CLOB, DBCLOB 및 BLOB 컬럼 값은 데이터베이스 관리 프로그램에 의해 무시됨에 유의하십시오. 2차 SQLVAR의 len.sqllonglen 필드가 대신 사용됩니다.
sqldata	포인터	문자열 SQLVARS의 경우 sqldata에는 코드 페이지가 들어 있습니다. 컬럼이 FOR BIT DATA 속성으로 정의된 문자열 SQLVAR의 경우, sqldata에는 0이 포함됩니다. 그렇지 않은 문자열 SQLVARS의 경우, sqldata에는 SBCS 데이터에 대해 SBCS 코드 페이지 또는 MBCS 데이터에 대해 복합 MBCS 코드 페이지와 연결된 SBCS 코드 페이지가 들어 있습니다. 일본어 EUC, 대만어 EUC 및 유니코드 UTF-8 문자열 SQLVARS의 경우, sqldata에는 각각 954, 964 및 1208이 들어 있습니다. 다른 모든 컬럼 유형의 경우 sqldata는 정의되어 있지 않습니다.	호스트 변수의 주소를 포함합니다. (이 때 페치된 데이터는 저장됩니다.)
sqlind	포인터	문자열 SQLVARS의 경우 sqlind에 복합 MBCS 코드 페이지와 연관된 DBCS 코드 페이지가 들어 있으면 sqlind에는 MBCS 데이터를 제외한 0이 포함됩니다. 다른 모든 유형의 경우 sqlind는 정의되지 않습니다.	연관 표시기 변수 주소가 있는 경우 그 주소가 포함됩니다. 그렇지 않으면 사용되지 않습니다. sqltype이 짝수 값인 경우 sqlind 필드는 무시됩니다.

표 72. 기본 SQLVAR 내의 필드 (계속)

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN 및 EXECUTE에서의 사용
sqlname	VARCHAR (30)	컬럼 또는 매개변수 표시문자의 규정되지 않은 이름이 들어 있습니다. 시스템 생성 이름이 있는 컬럼 및 매개변수 표시문자의 경우 13번째 바이트가 X'FF'로 설정됩니다. AS절로 지정된 컬럼 이름의 경우 이 바이트는 X'00'입니다.	호스트 데이터베이스로 연결할 때 sqlname은 다음과 같이 FOR BIT DATA 문자열을 표시하도록 설정될 수 있습니다. • SQLDA 헤더에 있는 SQLDAID의 여섯 번째 바이트는 '+'로 설정됩니다. • sqlname의 길이는 8입니다. • sqlname의 처음 2바이트는 X'0000'입니다. • sqlname의 3번째 및 4번째 바이트는 X'0000'입니다. • sqlname의 나머지 4바이트는 예약되며 X'00000000'로 설정되어야 합니다. XML 데이터로 작업할 경우 sqlname은 다음과 같이 XML 하위 유형을 표시하도록 설정될 수 있습니다. • sqlname의 길이는 8입니다. • sqlname의 처음 2바이트는 X'0000'입니다. • sqlname의 3번째 및 4번째 바이트는 X'0000'입니다. • sqlname의 5번째 바이트는 X'01'입니다. • sqlname의 나머지 3바이트는 예약되며 X'000000'로 설정되어야 합니다.

2차 SQLVAR의 어커런스 내의 필드

표 73. 2차 SQLVAR 내의 필드

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN 및 EXECUTE에서의 사용
len.sqllonglen	INTEGER	BLOB, CLOB 또는 DBCLOB 컬럼이나 매개변수 표시문자의 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 CLOB의 바이트 수와 DBCLOB의 2바이트 문자 수를 저장합니다.
reserve2	32비트의 경우는 CHAR(3), 64비트의 경우는 CHAR(11).	사용되지 않음	사용되지 않음

표 73. 2차 SQLVAR 내의 필드 (계속)

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN 및 EXECUTE에서의 사용
sqlflag4	CHAR(1)	값은 SQLVAR이 sqldatatype_name에 이름 지정된 목표 유형으로 참조 유형을 나타내는 경우 X'01'입니다. 값은 SQLVAR이 sqldatatype_name의 사용자 정의 유형 이름으로 구조화된 유형을 나타내는 경우 X'12'입니다. 그렇지 않으면 값은 X'00'입니다.	SQLVAR이 sqldatatype_name에 이름 지정된 목표 유형으로 참조 유형을 나타내는 경우에는 X'01'로 설정합니다. SQLVAR이 sqldatatype_name의 사용자 정의 유형 이름으로 구조화된 유형을 나타내는 경우에는 X'12'로 설정합니다. 그렇지 않으면 값은 X'00'입니다.
sqldatalen	포인터	사용되지 않음	BLOB, CLOB 및 DBCLOB 호스트 변수에만 사용됩니다. 이 필드가 NULL인 경우 실제 길이(2바이트 문자 단위)는 데이터의 시작 바로 전 4바이트에 저장되어야 하며, SQLDATA는 필드 길이의 첫 바이트를 지정해야 합니다. 이 필드가 NULL이 아닌 경우 여기에는 기본 SQLVAR과 대응하는 SQLDATA 필드로부터 지정된 버퍼에서 데이터의 실제 길이가 *바이트*로 포함되어 있는 4바이트 길이 버퍼(DBCLOB 경우에도 해당)에 대한 포인터가 포함됩니다. 이 필드의 사용 여부에 관계없이 len.sqllonglen 필드를 설정해야 함에 유의하십시오.
sqldatatype_name	VARCHAR(27)	사용자 정의 유형의 경우, 데이터베이스 관리 프로그램에서는 이를 완전한 사용자 정의 유형 이름으로 설정합니다.[1] 참조 유형의 경우 데이터베이스 관리 프로그램에서는 이를 참조 목표 유형의 완전한 유형 이름으로 설정합니다.	구조화된 유형의 경우 테이블 참고에 나와 있는 형식의 완전한 사용자 정의 유형 이름으로 설정합니다.[1]
예약	CHAR(3)	사용되지 않음	사용되지 않음

[1] 처음 8바이트에는 유형의 스키마 이름이 포함됩니다(필요에 따라 오른쪽 스페이스로 확장). 9번째 바이트에는 점(.)이 포함됩니다. 10 - 27번째 바이트에는 오른쪽 스페이스로 확장되지 않는 유형 이름의 아랫부분이 포함됩니다.

해당 필드의 주 목적이 사용자 정의 유형 이름에 대한 것이지만 해당 필드는 IBM 사전 정의 데이터 유형에 대해서도 설정되어 있음에 유의하십시오. 이 경우 스키마 이름은 SYSIBM이고, 이름의 아랫부분은 DATATYPES 카탈로그 뷰의 TYPENAME 컬럼에 저장된 이름입니다. 예를 들면, 다음과 같습니다.

```
type name         length   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 OUTPUT 또는 PREPARE OUTPUT INTO문의 경우 데이터베이스 관리 프로그램에서는 항상 SQLD를 결과 세트에 있는 컬럼 수나 매개변수 표시문자의 출력 수로 설정합니다. DESCRIBE INPUT 또는 PREPARE INPUT INTO문의 경우, 데이터베이스 관리 프로그램에서는 항상 SQLD를 명령문 내의 입력 매개변수 표시문자의 수로 설정합니다. CALL문의 INOUT 매개변수에 해당하는 매개변수 표시문자가 입력 및 출력 디스크립터에 모두 설명되어 있습니다.

SQLDA 내의 SQLVAR은 다음 경우에 설정됩니다.

- SQLN >= SQLD 및 LOB, 사용자 정의 유형 또는 참조 유형인 항목이 없을 경우

  첫 번째 SQLD SQLVAR 항목이 설정되고 SQLDOUBLED가 공백으로 설정될 경우
- SQLN >= 2*SQLD 및 최소한 하나의 항목이 LOB, 사용자 정의 유형 또는 참조 유형일 경우

  두 번째 SQLD SQLVAR 항목이 설정되고 SQLDOUBLED가 '2'로 설정될 경우
- SQLD <= SQLN < 2*SQLD 및 최소한 하나의 항목이 구별, 배열, 커서, 행 또는 참조 유형이지만 LOB 항목 또는 구조화된 유형 항목이 없을 경우

  첫 번째 SQLD SQLVAR 항목이 설정되고 SQLDOUBLED가 공백으로 설정될 경우 SQLWARN 바인드 옵션이 YES인 경우, 경고 SQLCODE +237(SQLSTATE 01594)이 발행됩니다.

SQLDA 내의 SQLVAR은 다음의 경우 NOT으로 설정됩니다(추가 공간과 또 다른 DESCRIBE의 할당이 필요).

- SQLN < SQLD 및 LOB, 사용자 정의 유형 또는 참조 유형인 항목이 없을 경우

  SQLVAR 항목이 설정되지 않고, SQLDOUBLED가 공백으로 설정됩니다. SQLWARN 바인드 옵션이 YES인 경우, 경고 SQLCODE +236(SQLSTATE 01005)이 발행됩니다.

  성공적인 DESCRIBE를 위해 SQLD SQLVAR을 할당하십시오.
- SQLN < SQLD 및 최소한 하나의 항목이 구별, 배열, 커서, 행 또는 참조 유형이지만 LOB 항목 또는 구조화된 유형 항목이 없을 경우

  SQLVAR 항목이 설정되지 않고, SQLDOUBLED가 공백으로 설정됩니다. SQLWARN 바인드 옵션이 YES인 경우, 경고 SQLCODE +239(SQLSTATE 01005)가 발행됩니다.

구별, 배열, 커서 및 행 유형의 이름과 참조 유형의 목표 유형을 포함하여 성공적인 DESCRIBE를 위해 2*SQLD SQLVAR을 할당하십시오.

- SQLN < 2*SQLD 및 최소한 하나의 항목이 LOB 또는 구조화된 유형일 경우

  SQLVAR 항목이 설정되지 않고, SQLDOUBLED가 공백으로 설정됩니다. 경고 SQLCODE +238(SQLSTATE 01005)이 발행됩니다(SQLWARN 바인드 옵션의 설정과는 상관없음).

  성공적인 DESCRIBE를 위해 2*SQLD SQLVAR을 할당하십시오.

LOB 항목에 대한 위 목록에서 참조사항에는 소스 유형이 LOB 유형인 구별 유형 항목이 포함됩니다.

BIND 또는 PREP 명령의 SQLWARN 옵션은 DESCRIBE(또는 PREPARE INTO)가 경고 SQLCODE +236, +237, +239 등의 리턴 여부를 제어하기 위해 사용됩니다. 사용자의 응용프로그램 코드는 항상 이러한 SQLCODE가 리턴될 수 있음을 고려하는 것이 좋습니다. 선택 목록에 LOB 또는 구조화된 유형 항목이 있고 SQLDA에 충분한 SQLVAR이 없는 경우에는 항상 경고 SQLCODE +238이 리턴됩니다. 이는 응용프로그램이 결과 세트의 LOB 또는 구조화된 유형 항목으로 인해 SQLVAR의 수를 두 배로 만들어야 한다는 것을 알 수 있는 유일한 방법입니다.

구조화된 유형 항목이 설명되고 있으나 FROM SQL 변환이 정의되지 않은 경우(CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터를 사용하여 TRANSFORM GROUP을 지정하지 않았거나(SQLSTATE 42741), 이름 그룹에 정의된 FROM SQL 변환 함수가 없기 때문에(SQLSTATE 42744), DESCRIBE에서 오류를 리턴합니다. 이 오류는 구조화된 유형 항목이 있는 테이블의 DESCRIBE에 대해 리턴된 것과 동일한 오류입니다.

데이터베이스 관리 프로그램이 SQLDA에서 저장될 수 있는 ID보다 긴 ID를 리턴하면, ID는 절단되며 경고가 리턴됩니다(SQLSTATE 01665). 그러나 구조화된 유형의 이름이 절단되면 오류가 리턴됩니다(SQLSTATE 42622). ID 길이 제한은 『SQL 및 XQuery 제한』을 참조하십시오.

SQLTYPE 및 SQLLEN

844 페이지의 표 74에서는 SQLDA의 SQLTYPE 및 SQLLEN 필드에 나타날 수 있는 값을 나타냅니다. DESCRIBE 및 PREPARE INTO에서 SQLTYPE의 짝수 값은 컬럼에 널(NULL)이 허용되지 않는다는 것을 의미하고, 홀수 값은 컬럼에 널(NULL)이 허용된다는 것을 의미합니다. FETCH, OPEN 및 EXECUTE에서 SQLTYPE의 짝수 값은 표시기 변수가 제공되지 않는다는 것을 의미하고 홀수 값은 SQLIND에 표시기 변수의 주소가 포함된다는 것을 의미합니다.

표 74. DESCRIBE, FETCH, OPEN 및 EXECUTE에 대한 SQLTYPE 및 SQLLEN 값

	DESCRIBE 및 PREPARE INTO의 경우		FETCH, OPEN 및 EXECUTE의 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	시간	8	시간의 고정 길이 문자열 표현	호스트 변수의 길이 속성
392/393	시간소인	TIMESTAMP(0)의 경우 19, TIMESTAMP(*p*)의 경우 20+*p*	시간소인의 고정 길이 문자열 표현	호스트 변수의 길이 속성
400/401	N/A	N/A	NULL-종료된 그래픽 문자열	호스트 변수의 길이 속성
404/405	BLOB	0 *	BLOB	사용되지 않음 *
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	해당되지 않음	해당되지 않음	NULL-종료된 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	압축 10진수	바이트 1은 정밀도, 바이트 2는 스케일	압축 10진수	바이트 1은 정밀도, 바이트 2는 스케일
492/493	Bigint	8	Bigint	8
496/497	큰 정수(large integer)	4	큰 정수(large integer)	4
500/501	작은 정수(small integer)	2	작은 정수(small integer)	2
916/917	해당되지 않음	해당되지 않음	BLOB 파일 참조 변수	267
920/921	해당되지 않음	해당되지 않음	CLOB 파일 참조 변수	267
924/925	해당되지 않음	해당되지 않음	DBCLOB 파일 참조 변수	267
960/961	해당되지 않음	해당되지 않음	BLOB 로케이터	4
964/965	해당되지 않음	해당되지 않음	CLOB 로케이터	4
968/969	해당되지 않음	해당되지 않음	DBCLOB 로케이터	4
988/989	XML	0	적용되지 않음. 대신에 XML AS <문자열 또는 2진 LOB 유형> 호스트 변수를 사용하십시오.	사용되지 않음
996	10진수 부동 소수점	DECFLOAT(16)의 경우 8, DECFLOAT(34)의 경우 16	10진수 부동 소수점	DECFLOAT(16)의 경우 8, DECFLOAT(34)의 경우 16

표 74. DESCRIBE, FETCH, OPEN 및 EXECUTE에 대한 SQLTYPE 및 SQLLEN 값 (계속)

	DESCRIBE 및 PREPARE INTO의 경우		FETCH, OPEN 및 EXECUTE의 경우	
SQLTYPE	**컬럼 데이터 유형**	**SQLLEN**	**호스트 변수 데이터 유형**	**SQLLEN**
2440/2441	행	해당되지 않음	행	사용되지 않음
2440/2441	커서	해당되지 않음	행	사용되지 않음

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 지원됩니다.

인식되지 않아 지원되지 않는 SQLTYPE

SQLDA의 SQLTYPE 필드에 나타나는 값은 데이터의 수신자뿐 아니라 송신자에서 사용 가능한 데이터 유형 지원 레벨에 따라 다릅니다. 이는 특히 새로운 데이터 유형이 제품에 추가될 때 중요합니다.

새 데이터 유형은 데이터의 수신자 또는 송신자에서 지원 또는 지원되지 않을 수도 있으며, 데이터의 송신자 또는 수신자에 의해 인식되거나 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 수신자가 모두 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 응용프로그램 또는 데이터베이스 관리 프로그램으로부터 제공될 수 있습니다.

데이터 유형	호환되는 데이터 유형
BIGINT	DECIMAL(19, 0)
ROWID[1]	VARCHAR(40) FOR BIT DATA

[1] ROWID는 z/OS용 DB2 Universal Database 버전 8에서 지원됩니다.

데이터 유형이 대체되는 SQLDA에 아무런 표시도 되지 않음을 유념하십시오.

압축 10진수

압축 10진수는 2진 코드화 10진수(BCD) 표의 변형으로 저장됩니다. BCD에서 각 니블(nybble 4비트)은 한 자릿수의 10진수를 의미합니다. 예를 들어, 0001 0111 1001은 179를 나타냅니다. 따라서 압축 10진수 값을 니블(nybble)별로 읽으십시오. 이 값을 바이트로 저장한 후 16진수로 표기된 바이트 수를 읽어 10진수로 돌리십시오. 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 들어, 맨 오른쪽 5 자릿수는 10진수 소수점 오른쪽에 놓이게 됩니다.

부호는 숫자를 표시하는 니블 오른쪽에 니블로 표시할 수 있습니다. 양수 또는 음수 부호가 아래와 같이 표시됩니다.

표 75. 압축 10진수에 대한 부호 표시기

	표기		
표시	**2**진수	**10**진수	**16**진수
양수 (+)	1100	12	C
음수 (-)	1101	13	D

요약:

- 값을 저장하려면 *p*/2+1바이트를 할당하십시오. 여기서, *p*는 정밀도를 나타냅니다.
- 왼쪽에서 오른쪽으로 값을 표시하기 위한 니블을 할당합니다. 숫자에 균등한 정밀도가 있는 경우, 선행 0 니블이 추가됩니다. 이 지정에는 앞(유효하지 않음)과 뒤(유효)에 0자릿수가 포함됩니다.
- 부호 니블은 마지막 바이트의 두 번째 니블이 됩니다.

예를 들어, 다음과 같습니다.

컬럼	값	바이트별로 그룹화된 **16**진수의 니블
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

10진수를 위한 SQLLEN 필드

SQLLEN 필드에는 10진수 컬럼의 정밀도(첫 번째 바이트)와 스케일(두 번째 바이트)이 포함됩니다. 이식 가능한 응용프로그램을 작성하는 경우 정밀도와 스케일 바이트를 각각 설정해야 하거나, short 정수로 이 둘을 함께 설정해야 합니다. 이렇게 하면 정수의 바이트 리버설을 피할 수 있습니다.

예를 들어, C에서는 다음과 같이 됩니다.

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

부록 D. 시스템 카탈로그 뷰

데이터베이스 관리 프로그램은 기본 시스템 카탈로그 테이블 위에서 정의되는 두 세트의 시스템 카탈로그 뷰를 작성하고 유지합니다.

- SYSCAT 뷰는 SYSCAT 스키마에 있는 읽기 전용 카탈로그 뷰입니다. CREATE DATABASE문의 RESTRICT 옵션은 SELECT 특권을 부여하는 방법을 판별합니다. RESTRICT 옵션을 지정하지 않으면 SELECT 특권을 PUBLIC에 부여합니다.
- SYSSTAT 뷰는 SYSSTAT 스키마에 있는 갱신 가능 카탈로그 뷰입니다. 갱신 가능 뷰에는 옵티마이저가 사용하는 통계 정보가 들어있습니다. 성능을 테스트하기 위해 이들 뷰의 일부 컬럼의 값을 변경할 수 있습니다. (통계를 변경하기 전에 모든 통계가 현재 상태를 반영하도록 RUNSTATS 명령을 호출할 것을 권장합니다.)

응용프로그램은 기본 카탈로그 테이블이 아닌 SYSCAT 및 SYSSTAT 뷰에 기록되어야 합니다.

모든 시스템 카탈로그 뷰는 데이터베이스 작성 시에 작성됩니다. 카탈로그 뷰를 명시적으로 작성하거나 삭제할 수 없습니다. 유니코드 데이터베이스에서 카탈로그 뷰는 IDENTITY 조합과 함께 작성됩니다. 유니코드가 아닌 데이터베이스에서, 카탈로그 뷰는 데이터베이스 조합으로 작성됩니다. 뷰는 SQL 데이터 정의 명령문, 환경 루틴 및 특정 유틸리티에 대한 응답으로 일반 조작 중에 갱신됩니다. 시스템 카탈로그 뷰의 데이터는 일반 SQL 쿼리 기능을 통해 사용할 수 있습니다. 시스템 카탈로그 뷰(일부 갱신 가능 카탈로그 뷰는 예외)는 일반 SQL 데이터 처리 명령문을 사용하여 수정할 수 없습니다.

오브젝트 테이블, 컬럼 또는 인덱스 오브젝트는 사용자가 오브젝트에 대한 CONTROL 특권을 보유하거나 명시적 DATAACCESS 권한을 보유하는 경우에만 사용자의 갱신 가능 SYSSTAT 카탈로그 뷰에 나타납니다. 해당 사용자가 루틴을 소유하거나 SQLADM 권한을 보유한 경우 루틴 오브젝트가 사용자의 갱신 가능 SYSSTAT.ROUTINES 카탈로그 뷰에 표시됩니다.

뷰에서 컬럼의 순서는 릴리스에 따라서 변할 수 있습니다. 이것이 프로그래밍 논리에 영향을 미치지 못하게 하려면 명시적으로 선택 목록에서 컬럼을 지정하고 SELECT * 사용을 피하십시오. 컬럼은 컬럼이 설명하는 오브젝트의 유형을 기초로 일관성 있는 이름을 갖습니다.

표 76. 일관된 컬럼 이름이 설명하는 오브젝트의 일관된 컬럼 이름 샘플

설명되는 오브젝트	컬럼 이름
테이블	TABSCHEMA, TABNAME
인덱스	INDSCHEMA, INDNAME

표 76. 일관된 컬럼 이름이 설명하는 오브젝트의 일관된 컬럼 이름 샘플 (계속)

설명되는 오브젝트	컬럼 이름
인덱스 확장	IESCHEMA, IENAME
뷰	VIEWSCHEMA, VIEWNAME
제한조건	CONSTSCHEMA, CONSTNAME
트리거	TRIGSCHEMA, TRIGNAME
패키지	PKGSCHEMA, PKGNAME
유형	TYPESCHEMA, TYPENAME, TYPEID
함수	ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, ROUTINEID
메소드	ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, ROUTINEID
프로시저	ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, ROUTINEID
컬럼	COLNAME
스키마	SCHEMANAME
테이블 스페이스	TBSPACE
데이터베이스 파티션 그룹	DBPGNAME
감사 규정	AUDITPOLICYNAME, AUDITPOLICYID
버퍼 풀	BPNAME
이벤트 모니터	EVMONNAME
조건	CONDSCHEMA, CONDMODULENAME, CONDNAME, CONDMODULEID
데이터 소스	SERVERNAME, SERVERTYPE, SERVERVERSION
전역 변수	VARSCHEMA, VARMODULENAME, VARNAME, VARMODULEID
막대 그래프 템플리트	TEMPLATENAME, TEMPLATEID
모듈	MODULESCHEMA, MODULENAME, MODULEID
역할	ROLENAME, ROLEID
보안 레이블	SECLABELNAME, SECLABELID
보안 규정	SECPOLICYNAME, SECPOLICYID
순서	SEQSCHEMA, SEQNAME
임계값	THRESHOLDNAME, THRESHOLDID
트러스트된 컨텍스트	CONTEXTNAME, CONTEXTID
작업 조치	ACTIONNAME, ACTIONID
작업 조치 세트	ACTIONSETNAME, ACTIONSETID
작업 클래스	WORKCLASSNAME, WORKCLASSID
작업 클래스 세트	WORKCLASSSETNAME, WORKCLASSSETID
워크로드	WORKLOADID, WORKLOADNAME
랩퍼	WRAPNAME
변경 시간소인	ALTER_TIME
작성 시간소인	CREATE_TIME

카탈로그 뷰에 대한 로드 맵

표 77. 읽기 전용 카탈로그 뷰에 대한 로드 맵

설명	카탈로그 뷰
구조화된 데이터 유형의 속성	854 페이지의『SYSCAT.ATTRIBUTES』
감사 규정	856 페이지의『SYSCAT.AUDITPOLICIES』
	858 페이지의『SYSCAT.AUDITUSE』
데이터베이스에 대한 권한	889 페이지의『SYSCAT.DBAUTH』
데이터베이스 파티션 그룹의 버퍼 풀 구성	860 페이지의『SYSCAT.BUFFERPOOLS』
데이터베이스 파티션의 버퍼 풀 크기	859 페이지의『SYSCAT.BUFFERPOOLDBPARTITIONS』
캐스트(cast) 함수	861 페이지의『SYSCAT.CASTFUNCTIONS』
점검 제한조건	862 페이지의『SYSCAT.CHECKS』
컬럼 특권	863 페이지의『SYSCAT.COLAUTH』
컬럼	872 페이지의『SYSCAT.COLUMNS』
점검 제한조건이 참조하는 컬럼	864 페이지의『SYSCAT.COLCHECKS』
차원에서 사용되는 컬럼	877 페이지의『SYSCAT.COLUSE』
키에서 사용되는 컬럼	927 페이지의『SYSCAT.KEYCOLUSE』
조건	878 페이지의『SYSCAT.CONDITIONS』
제한조건 종속성	879 페이지의『SYSCAT.CONSTDEP』
데이터베이스 파티션 그룹 데이터베이스 파티션	891 페이지의『SYSCAT.DBPARTITIONGROUPDEF』
데이터베이스 파티션 그룹 정의	892 페이지의『SYSCAT.DBPARTITIONGROUPS』
데이터 파티션	882 페이지의『SYSCAT.DATAPARTITIONEXPRESSION』
	883 페이지의『SYSCAT.DATAPARTITIONS』
데이터 유형 종속성	885 페이지의『SYSCAT.DATATYPEDEP』
데이터 유형	886 페이지의『SYSCAT.DATATYPES』
상세한 컬럼 그룹 통계	866 페이지의『SYSCAT.COLGROUPCOLS』
	867 페이지의『SYSCAT.COLGROUPDIST』
	868 페이지의『SYSCAT.COLGROUPDISTCOUNTS』
	869 페이지의『SYSCAT.COLGROUPS』
상세한 컬럼 옵션	871 페이지의『SYSCAT.COLOPTIONS』
상세한 컬럼 통계	865 페이지의『SYSCAT.COLDIST』
분산 맵	943 페이지의『SYSCAT.PARTITIONMAPS』
이벤트 모니터 정의	893 페이지의『SYSCAT.EVENTMONITORS』
현재 모니터하는 이벤트	895 페이지의『SYSCAT.EVENTS』
	896 페이지의『SYSCAT.EVENTTABLES』
행 데이터 유형의 필드	968 페이지의『SYSCAT.ROWFIELDS』
함수 종속성[1]	951 페이지의『SYSCAT.ROUTINEDEP』
함수 맵핑	900 페이지의『SYSCAT.FUNCMAPPINGS』
함수 맵핑 옵션	898 페이지의『SYSCAT.FUNCMAPOPTIONS』
함수 매개변수 맵핑 옵션	899 페이지의『SYSCAT.FUNCMAPPARMOPTIONS』
함수 매개변수[1]	955 페이지의『SYSCAT.ROUTINEPARMS』
함수[1]	958 페이지의『SYSCAT.ROUTINES』

표 77. 읽기 전용 카탈로그 뷰에 대한 로드 맵 (계속)

설명	카탈로그 뷰
전역 변수	1015 페이지의 『SYSCAT.VARIABLEAUTH』
	1016 페이지의 『SYSCAT.VARIABLEDEP』
	1018 페이지의 『SYSCAT.VARIABLES』
계층 구조(유형, 테이블, 뷰)	901 페이지의 『SYSCAT.HIERARCHIES』
	897 페이지의 『SYSCAT.FULLHIERARCHIES』
식별 컬럼	870 페이지의 『SYSCAT.COLIDENTATTRIBUTES』
인덱스 컬럼	906 페이지의 『SYSCAT.INDEXCOLUSE』
인덱스 데이터 파티션	922 페이지의 『SYSCAT.INDEXPARTITIONS』
인덱스 종속성	907 페이지의 『SYSCAT.INDEXDEP』
인덱스 이용	915 페이지의 『SYSCAT.INDEXEXPLOITRULES』
인덱스 확장 종속성	916 페이지의 『SYSCAT.INDEXEXTENSIONDEP』
인덱스 확장 매개변수	919 페이지의 『SYSCAT.INDEXEXTENSIONPARMS』
인덱스 확장 검색 방법	918 페이지의 『SYSCAT.INDEXEXTENSIONMETHODS』
인덱스 확장	920 페이지의 『SYSCAT.INDEXEXTENSIONS』
인덱스 옵션	921 페이지의 『SYSCAT.INDEXOPTIONS』
인덱스 특권	905 페이지의 『SYSCAT.INDEXAUTH』
인덱스	909 페이지의 『SYSCAT.INDEXES』
유효하지 않은 오브젝트	926 페이지의 『SYSCAT.INVALIDOBJECTS』
메소드 종속성[1]	951 페이지의 『SYSCAT.ROUTINEDEP』
메소드 매개변수[1]	958 페이지의 『SYSCAT.ROUTINES』
메소드[1]	958 페이지의 『SYSCAT.ROUTINES』
모듈 오브젝트	929 페이지의 『SYSCAT.MODULEOBJECTS』
모듈 특권	928 페이지의 『SYSCAT.MODULEAUTH』
모듈	930 페이지의 『SYSCAT.MODULES』
별칭	932 페이지의 『SYSCAT.NICKNAMES』
오브젝트 맵핑	931 페이지의 『SYSCAT.NAMEMAPPINGS』
패키지 종속성	936 페이지의 『SYSCAT.PACKAGEDEP』
패키지 특권	935 페이지의 『SYSCAT.PACKAGEAUTH』
패키지	938 페이지의 『SYSCAT.PACKAGES』
파티션된 테이블	992 페이지의 『SYSCAT.TABDETACHEDDEP』
pass-through 특권	944 페이지의 『SYSCAT.PASSTHRUAUTH』
술어 스펙	945 페이지의 『SYSCAT.PREDICATESPECS』
프로시저 옵션	953 페이지의 『SYSCAT.ROUTINEOPTIONS』
프로시저 매개변수 옵션	954 페이지의 『SYSCAT.ROUTINEPARMOPTIONS』
프로시저 매개변수[1]	955 페이지의 『SYSCAT.ROUTINEPARMS』
프로시저[1]	958 페이지의 『SYSCAT.ROUTINES』

표 77. 읽기 전용 카탈로그 뷰에 대한 로드 맵 (계속)

설명	카탈로그 뷰
보호 테이블	971 페이지의『SYSCAT.SECURITYLABELACCESS』
	972 페이지의『SYSCAT.SECURITYLABELCOMPONENT ELEMENTS』
	973 페이지의『SYSCAT.SECURITYLABELCOMPONENTS』
	974 페이지의『SYSCAT.SECURITYLABELS』
	975 페이지의『SYSCAT.SECURITYPOLICIES』
	976 페이지의『SYSCAT.SECURITYPOLICYCOMPONENT RULES』
	977 페이지의『SYSCAT.SECURITYPOLICYEXEMPTIONS』
	986 페이지의『SYSCAT.SURROGATEAUTHIDS』
z/OS용 DB2 호환성을 제공합니다.	1044 페이지의『SYSIBM.SYSDUMMY1』
참조 제한조건	946 페이지의『SYSCAT.REFERENCES』
리모트 테이블 옵션	1001 페이지의『SYSCAT.TABOPTIONS』
역할	947 페이지의『SYSCAT.ROLEAUTH』
	948 페이지의『SYSCAT.ROLES』
루틴 종속성	951 페이지의『SYSCAT.ROUTINEDEP』
루틴 매개변수[1]	955 페이지의『SYSCAT.ROUTINEPARMS』
루틴 특권	949 페이지의『SYSCAT.ROUTINEAUTH』
루틴[1]	958 페이지의『SYSCAT.ROUTINES』
	966 페이지의『SYSCAT.ROUTINESFEDERATED』
스키마 특권	969 페이지의『SYSCAT.SCHEMAAUTH』
스키마	970 페이지의『SYSCAT.SCHEMATA』
시퀀스 특권	978 페이지의『SYSCAT.SEQUENCEAUTH』
시퀀스	979 페이지의『SYSCAT.SEQUENCES』
서버 옵션	981 페이지의『SYSCAT.SERVEROPTIONS』
서버 특정 사용자 옵션	1014 페이지의『SYSCAT.USEROPTIONS』
패키지의 명령문	985 페이지의『SYSCAT.STATEMENTS』
스토어드 프로시저	958 페이지의『SYSCAT.ROUTINES』
시스템 서버	982 페이지의『SYSCAT.SERVERS』
테이블 제한조건	989 페이지의『SYSCAT.TABCONST』
테이블 종속성	990 페이지의『SYSCAT.TABDEP』
테이블 특권	987 페이지의『SYSCAT.TABAUTH』
테이블 스페이스 특권	1002 페이지의『SYSCAT.TBSPACEAUTH』
테이블 스페이스	999 페이지의『SYSCAT.TABLESPACES』
테이블	993 페이지의『SYSCAT.TABLES』
변환	1006 페이지의『SYSCAT.TRANSFORMS』
트리거 종속성	1007 페이지의『SYSCAT.TRIGDEP』
트리거	1009 페이지의『SYSCAT.TRIGGERS』
트러스트된 컨텍스트	880 페이지의『SYSCAT.CONTEXTATTRIBUTES』
	881 페이지의『SYSCAT.CONTEXTS』

표 77. 읽기 전용 카탈로그 뷰에 대한 로드 맵 (계속)

설명	카탈로그 뷰
유형 맵핑	1011 페이지의 『SYSCAT.TYPEMAPPINGS』
사용자 정의 함수	958 페이지의 『SYSCAT.ROUTINES』
뷰 종속성	990 페이지의 『SYSCAT.TABDEP』
뷰	993 페이지의 『SYSCAT.TABLES』
	1020 페이지의 『SYSCAT.VIEWS』
워크로드 관리	902 페이지의 『SYSCAT.HISTOGRAMTEMPLATEBINS』
	903 페이지의 『SYSCAT.HISTOGRAMTEMPLATES』
	904 페이지의 『SYSCAT.HISTOGRAMTEMPLATEUSE』
	983 페이지의 『SYSCAT.SERVICECLASSES』
	1003 페이지의 『SYSCAT.THRESHOLDS』
	1021 페이지의 『SYSCAT.WORKACTIONS』
	1024 페이지의 『SYSCAT.WORKACTIONSETS』
	1025 페이지의 『SYSCAT.WORKCLASSES』
	1026 페이지의 『SYSCAT.WORKCLASSSETS』
	1027 페이지의 『SYSCAT.WORKLOADAUTH』
	1028 페이지의 『SYSCAT.WORKLOADCONNATTR』
	1029 페이지의 『SYSCAT.WORKLOADS』
랩퍼 옵션	1032 페이지의 『SYSCAT.WRAPOPTIONS』
랩퍼	1033 페이지의 『SYSCAT.WRAPPERS』
XML 문자열	1036 페이지의 『SYSCAT.XMLSTRINGS』
XML 값 인덱스	925 페이지의 『SYSCAT.INDEXXMLPATTERNS』
XSR 오브젝트	1034 페이지의 『SYSCAT.XDBMAPGRAPHS』
	1035 페이지의 『SYSCAT.XDBMAPSHREDTREES』
	1037 페이지의 『SYSCAT.XSROBJECTAUTH』
	1038 페이지의 『SYSCAT.XSROBJECTCOMPONENTS』
	1039 페이지의 『SYSCAT.XSROBJECTDEP』
	1041 페이지의 『SYSCAT.XSROBJECTDETAILS』
	1042 페이지의 『SYSCAT.XSROBJECTHIERARCHIES』
	1043 페이지의 『SYSCAT.XSROBJECTS』

[1] DB2 버전 7.1 및 이전에서 정의되는 함수, 메소드 및 프로시저에 대한 다음 카탈로그 뷰는 아직 사용할 수 있습니다.

```
함수:  SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
메소드:    SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
프로시저: SYSCAT.PROCEDURES, SYSCAT.PROCPARMS
```

그러나 이들 뷰는 DB2 버전 7.1 이후 갱신되지 않았습니다. 대신 SYSCAT.ROUTINES, SYSCAT.ROUTINEDEP 또는 SYSCAT.ROUTINEPARMS 카탈로그 뷰를 사용하십시오.

표 78. 갱신 가능한 카탈로그 뷰에 대한 로드 맵

설명	카탈로그 뷰
컬럼	1049 페이지의 『SYSSTAT.COLUMNS』

표 78. 갱신 가능한 카탈로그 뷰에 대한 로드 맵 (계속)

설명	카탈로그 뷰
상세한 컬럼 그룹 통계	1046 페이지의 『SYSSTAT.COLGROUPDIST』
	1047 페이지의 『SYSSTAT.COLGROUPDISTCOUNTS』
	1048 페이지의 『SYSSTAT.COLGROUPS』
상세한 컬럼 통계	1045 페이지의 『SYSSTAT.COLDIST』
인덱스	1051 페이지의 『SYSSTAT.INDEXES』
루틴[1]	1054 페이지의 『SYSSTAT.ROUTINES』
테이블	1055 페이지의 『SYSSTAT.TABLES』

[1] SYSSTAT.FUNCTIONS 카탈로그 뷰가 함수 및 메소드에 대한 통계 갱신을 위해 아직 존재합니다. 그러나 이 뷰는 DB2 버전 7.1 이후의 어떤 변경도 반영하지 않습니다.

SYSCAT.ATTRIBUTES

각 행은 사용자 정의 구조화 데이터 유형에 대해 정의되는 속성을 나타냅니다. 부속 유형의 상속된 속성이 포함됩니다.

표 79. SYSCAT.ATTRIBUTES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPESCHEMA	VARCHAR(128)		속성을 포함하는 구조화된 데이터 유형의 스키마 이름.
TYPEMODULENAME	VARCHAR(128)	Y	구조화된 데이터 유형이 속한 모듈의 규정되지 않은 이름. 모듈 구조화된 데이터 유형이 아닌 경우 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)		속성을 포함하는 구조화된 데이터 유형의 규정되지 않은 이름.
ATTR_NAME	VARCHAR(128)		속성 이름.
ATTR_TYPESCHEMA	VARCHAR(128)		속성 데이터 유형의 스키마 이름.
ATTR_TYPEMODULENAME	VARCHAR(128)	Y	속성의 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 모듈 속성이 아닌 경우에는 널(NULL) 값입니다.
ATTR_TYPENAME	VARCHAR(128)		속성 데이터 유형의 규정되지 않은 이름.
TARGET_TYPESCHEMA	VARCHAR(128)	Y	목표 행 유형의 스키마 이름입니다. 참조 유형에만 적용되며, 그 외에는 널(NULL) 값입니다.
TARGET_TYPEMODULENAME	VARCHAR(128)	Y	목표 행 유형이 속한 모듈의 규정되지 않은 이름. 모듈 행 유형이 아닌 경우 널(NULL) 값입니다. 참조 유형에만 적용되며, 그 외에는 널(NULL) 값입니다.
TARGET_TYPENAME	VARCHAR(128)	Y	목표 행 유형의 규정되지 않은 이름입니다. 참조 유형에만 적용되며, 그 외에는 널(NULL) 값입니다.
SOURCE_TYPESCHEMA	VARCHAR(128)		상속된 속성의 경우 속성이 처음 정의되는 데이터 유형의 스키마 이름입니다. 상속되지 않은 속성의 경우 이 컬럼은 TYPESCHEMA와 동일합니다.
SOURCE_TYPEMODULENAME	VARCHAR(128)	Y	상속된 속성의 경우, 속성이 처음 정의된 데이터 유형이 속한 모듈의 규정되지 않은 이름. 상속되지 않은 속성의 경우 이 컬럼은 TYPEMODULEID와 동일합니다. 모듈 데이터 유형이 아닌 경우 널(NULL) 값입니다.
SOURCE_TYPENAME	VARCHAR(128)		상속된 속성의 경우 속성이 처음 정의되는 데이터 유형의 규정되지 않은 이름입니다. 상속되지 않은 속성의 경우 이 컬럼은 TYPENAME과 동일합니다.
ORDINAL	SMALLINT		구조화된 데이터 유형의 정의에서 속성의 위치이며, 0에서 시작합니다.
LENGTH	INTEGER		속성 데이터 유형의 길이. 속성이 사용자 정의 유형인 경우 0입니다.

표 79. SYSCAT.ATTRIBUTES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
SCALE	SMALLINT		속성 데이터 유형이 DECIMAL을 기반으로 하는 구별 유형 또는 DECIMAL인 경우 스케일이고, 속성 데이터 유형이 TIMESTAMP를 기반으로 하는 구별 유형 또는 TIMESTAMP인 경우 분수 초의 자릿수입니다. 그렇지 않으면, 0입니다.
CODEPAGE	SMALLINT		문자열 유형의 경우 코드 페이지를 표시합니다. 0은 FOR BIT DATA를 표시합니다. 비문자열 유형의 경우 0입니다.
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 속성 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 속성 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
LOGGED	CHAR(1)		LOB 유형에만 적용되며, 그 외에는 공백입니다. • N = 변경이 로그되지 않음 • Y = 변경이 로그됨
COMPACT	CHAR(1)		LOB 유형에만 적용되며, 그 외에는 공백입니다. • N = 압축되지 않은 형식으로 저장됨 • Y = 압축 형식으로 저장됨
DL_FEATURES	CHAR(10)		이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다.
JAVA_FIELDNAME	VARCHAR(256)	Y	나중에 사용하기 위해 예약됩니다.

SYSCAT.AUDITPOLICIES

각 행은 감사 규정을 나타냅니다.

표 80. SYSCAT.AUDITPOLICIES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
AUDITPOLICYNAME	VARCHAR(128)		감사 규정 이름.
AUDITPOLICYID	INTEGER		감사 규정 ID.
CREATE_TIME	TIMESTAMP		감사 규정이 작성된 시간.
ALTER_TIME	TIMESTAMP		감사 규정이 마지막으로 변경된 시간.
AUDITSTATUS	CHAR(1)		AUDIT 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
CONTEXTSTATUS	CHAR(1)		CONTEXT 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
VALIDATESTATUS	CHAR(1)		VALIDATE 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
CHECKINGSTATUS	CHAR(1)		CHECKING 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
SECMAINTSTATUS	CHAR(1)		SECMAINT 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
OBJMAINTSTATUS	CHAR(1)		OBJMAINT 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공

표 80. SYSCAT.AUDITPOLICIES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
SYSADMINSTATUS	CHAR(1)		SYSADMIN 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
EXECUTESTATUS	CHAR(1)		EXECUTE 범주의 상태. • B = 둘 다 • F = 실패 • N = 없음 • S = 성공
EXECUTEWITHDATA	CHAR(1)		EXECUTE 범주로 로그되는 호스트 변수 및 매개변수 표시문자. • N = 아니오 • Y = 예
ERRORTYPE	CHAR(1)		감사 오류 유형. • A = 감사 • N = 일반
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.AUDITUSE

각 행은 USER, GROUP 또는 권한(SYSADM, SYSCTRL, SYSMAINT) 같은 데이터베이스 이외의 오브젝트와 연관된 감사 규정을 나타냅니다.

표 81. SYSCAT.AUDITUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
AUDITPOLICYNAME	VARCHAR(128)		감사 규정 이름.
AUDITPOLICYID	INTEGER		감사 규정 ID.
OBJECTTYPE	CHAR(1)		이 감사 규정과 연관되는 오브젝트의 유형. • S = MQT • T = 테이블 • g = 권한 • i = 권한 부여 ID • x = 트러스트된 컨텍스트 • 공백 = 데이터베이스
SUBOBJECTTYPE	CHAR(1)		OBJECTTYPE이 'i'인 경우 권한 부여 ID가 나타내는 유형입니다. • G = 그룹 • R = 역할 • U = 사용자 • 공백 = 적용할 수 없음
OBJECTSCHEMA	VARCHAR(128)		감사 규정이 사용 중인 오브젝트의 스키마 이름입니다. OBJECTTYPE이 스키마가 적용되지 않는 오브젝트를 식별하는 경우 OBJECTSCHEMA는 널(Null)입니다.
OBJECTNAME	VARCHAR(128)		이 감사 규정이 사용 중인 오브젝트의 규정되지 않은 이름.

SYSCAT.BUFFERPOOLDBPARTITIONS

각 행은 버퍼 풀 및 데이터베이스 파티션의 조합을 나타내며, 해당 파티션의 버퍼 풀 *크기*는 동일한 데이터베이스 파티션 그룹(SYSCAT.BUFFERPOOLS에서 표시되는)의 다른 파티션의 디폴트 *크기*와 다릅니다.

표 82. SYSCAT.BUFFERPOOLDBPARTITIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
BUFFERPOOLID	INTEGER		내부 버퍼 풀 ID.
DBPARTITIONNUM	SMALLINT		데이터베이스 파티션 번호.
NPAGES	INTEGER		이 데이터베이스 파티션의 이 버퍼 풀의 페이지 수.

SYSCAT.BUFFERPOOLS

각 행은 데이터베이스의 하나의 데이터베이스 파티션 그룹 또는 데이터베이스의 모든 데이터베이스 파티션에 있는 버퍼 풀의 구성을 표시합니다.

표 83. SYSCAT.BUFFERPOOLS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
BPNAME	VARCHAR(128)		버퍼 풀 이름.
BUFFERPOOLID	INTEGER		버퍼 풀 ID.
DBPGNAME	VARCHAR(128)	Y	데이터베이스 파티션 그룹의 이름(버퍼가 데이터베이스의 모든 데이터베이스 파티션에 존재하는 경우 널(NULL) 값).
NPAGES	INTEGER		이 데이터베이스 파티션 그룹에 있는 데이터베이스 파티션에서 이 버퍼 풀에 있는 디폴트 페이지 수.
PAGESIZE	INTEGER		이 데이터베이스 파티션 그룹에 있는 데이터베이스 파티션에서 이 버퍼 풀에 대한 페이지 크기.
ESTORE	INTEGER		항상 'N'입니다. 확장 스토리지는 더 이상 적용되지 않습니다.
NUMBLOCKPAGES	INTEGER		블록 기반 영역에 있어야 하는 버퍼 풀의 페이지 수입니다. 버퍼 풀의 블록 기반 영역은 시퀀스 프리페치를 수행하는 프리페처에 의해서만 사용됩니다.
BLOCKSIZE	INTEGER		블록의 페이지 수
NGNAME[1]	VARCHAR(128)	Y	데이터베이스 파티션 그룹의 이름(버퍼가 데이터베이스의 모든 데이터베이스 파티션에 존재하는 경우 널(NULL) 값).

주:

1. 이전 버전과의 호환성을 위해 NGNAME 컬럼이 포함됩니다. DBPGNAME을 참조하십시오.

SYSCAT.CASTFUNCTIONS

각 행은 내장 캐스트(cast) 함수를 포함하지 않은 캐스트 함수를 나타냅니다.

표 84. SYSCAT.CASTFUNCTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
FROM_TYPESCHEMA	VARCHAR(128)		매개변수 데이터 유형의 스키마 이름.
FROM_TYPEMODULENAME	VARCHAR(128)		매개변수의 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 모듈 데이터 유형이 아닌 경우 널(NULL) 값입니다.
FROM_TYPENAME	VARCHAR(128)		매개변수의 데이터 유형 이름.
FROM_TYPEMODULEID	INTEGER	Y	매개변수의 데이터 유형이 속하는 모듈의 ID. 모듈 데이터 유형이 아닌 경우 널(NULL) 값입니다.
TO_TYPESCHEMA	VARCHAR(128)		캐스팅 후 결과 데이터 유형의 스키마 이름.
TO_TYPEMODULENAME	VARCHAR(128)		캐스팅 후 결과의 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 모듈 데이터 유형이 아닌 경우 널(NULL) 값입니다.
TO_TYPENAME	VARCHAR(128)		캐스팅 후 결과의 데이터 유형 이름.
TO_TYPEMODULEID	INTEGER	Y	캐스팅 후 결과의 데이터 유형이 속하는 모듈의 ID. 모듈 데이터 유형이 아닌 경우 널(NULL) 값입니다.
FUNCSCHEMA	VARCHAR(128)		함수의 스키마 이름.
FUNCMODULENAME	VARCHAR(128)		함수가 속하는 모듈의 규정되지 않은 이름. 모듈 함수가 아니면 널(NULL) 값입니다.
FUNCNAME	VARCHAR(128)		함수의 규정되지 않은 이름.
SPECIFICNAME	VARCHAR(128)		루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
FUNCMODULEID	INTEGER	Y	함수가 속하는 모듈의 ID. 모듈 함수가 아니면 널(NULL) 값입니다.
ASSIGN_FUNCTION	CHAR(1)		• N = 지정 함수가 아님 • Y = 내재된 지정 함수

SYSCAT.CHECKS

각 행은 구체화된 쿼리 테이블에 있는 점검 제한조건 또는 파생 컬럼을 나타냅니다. 테이블 계층의 경우 각 점검 제한조건은 제한조건이 작성된 계층 구조의 레벨에서만 기록됩니다.

표 85. SYSCAT.CHECKS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONSTNAME	VARCHAR(128)		점검 제한조건의 이름.
OWNER	VARCHAR(128)		제한조건 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
TABSCHEMA	VARCHAR(128)		이 제한조건이 적용되는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		이 제한조건이 적용되는 테이블 이름.
CREATE_TIME	TIMESTAMP		제한조건이 정의된 시간입니다. 이 제한조건의 파트인 함수를 해결하는 데 사용됩니다. 제한조건이 정의된 후에 작성된 함수는 선택되지 않습니다.
QUALIFIER	VARCHAR(128)		오브젝트 정의 시 디폴트 스키마의 값입니다. 규정되지 않은 참조를 완료하는 데 사용됩니다.
TYPE	CHAR(1)		점검 제한조건 유형: • C = 점검 제한조건 • F = 함수 종속성 • O = 제한조건이 오브젝트 등록 정보임 • S = GENERATED ALWAYS 컬럼에 대한 시스템 생성 점검 제한조건
FUNC_PATH	CLOB(2K)		제한조건이 정의된 시간의 SQL 경로
TEXT	CLOB(2M)		파생 컬럼의 점검 조건 또는 정의의 텍스트입니다.[1]
PERCENTVALID	SMALLINT		정보용 제한조건이 유효한 행 수로서, 총계의 백분율로 표현됩니다.
COLLATIONSCHEMA	VARCHAR(128)		제한조건에 대한 조합의 스키마 이름.
COLLATIONNAME	VARCHAR(128)		제한조건에 대한 조합의 규정되지 않은 이름.
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		제한조건의 ORDER BY절에 대한 조합의 스키마 이름.
COLLATIONNAME_ORDERBY	VARCHAR(128)		제한조건의 ORDER BY절에 대한 조합의 규정되지 않은 이름.
DEFINER[2]	VARCHAR(128)		제한조건 소유자의 권한 부여 ID

주:

1. 카탈로그 뷰에서 점검 조건의 텍스트가 항상 데이터베이스 코드 페이지에 표시되며 대체 문자를 포함할 수 있습니다. 점검 제한조건은 항상 목표 테이블의 코드 페이지에서 적용되며, 적용될 때 대체 문자를 포함하지 않습니다. (점검 제한조건은 목표 테이블의 코드 페이지에 있는 원래 텍스트를 기초로 적용되며, 대체 문자를 포함하지 않을 수 있습니다.)
2. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.COLAUTH

각 행은 컬럼에서 하나 이상의 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 86. SYSCAT.COLAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권의 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
TABSCHEMA	VARCHAR(128)		특권이 보유된 테이블 또는 뷰의 스키마 이름.
TABNAME	VARCHAR(128)		특권이 보유된 테이블 또는 뷰의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		이 특권이 적용되는 컬럼 이름.
COLNO	SMALLINT		테이블에서 이 컬럼의 컬럼 번호(0에서 시작).
PRIVTYPE	CHAR(1)		• R = 참조 특권 • U = 갱신 특권
GRANTABLE	CHAR(1)		• G = 특권이 권한 부여 가능함 • N = 특권이 권한 부여 불가능함

주:

1. 특권은 컬럼별로 권한 부여될 수 있지만, 테이블 전체에서만 권한 취소할 수 있습니다.

SYSCAT.COLCHECKS

각 행은 점검 제한조건 또는 구체화된 쿼리 테이블의 정의에 의해 참조되는 컬럼을 나타냅니다. 테이블 계층의 경우 각 점검 제한조건은 제한조건이 작성된 계층 구조의 레벨에서만 기록됩니다.

표 87. SYSCAT.COLCHECKS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONSTNAME	VARCHAR(128)		점검 제한조건의 이름.
TABSCHEMA	VARCHAR(128)		참조된 컬럼을 포함하는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		참조된 컬럼을 포함하는 테이블의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		컬럼 이름
USAGE	CHAR(1)		• D = 컬럼이 함수 종속성에서 하위임 • P = 컬럼이 함수 종속성에서 상위임 • R = 컬럼이 점검 제한조건에서 참조됨 • S = 컬럼이 구체화된 쿼리 테이블을 지원하는 시스템 생성 컬럼 점검 제한조건에서 소스임 • T = 컬럼이 구체화된 쿼리 테이블을 지원하는 시스템 생성 컬럼 점검 제한조건에서 목표임

SYSCAT.COLDIST

각 행은 일부 컬럼의 *n*번째로 자주 사용되는 값이나 컬럼의 *n*번째 Quantile(누적 분산) 값을 나타냅니다. 실제 테이블의 컬럼(뷰가 아님)에만 적용됩니다. 유형이 지정된 테이블의 상속된 컬럼에 대해서는 통계가 기록되지 않습니다.

표 88. SYSCAT.COLDIST 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		통계가 적용되는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		통계가 적용되는 테이블의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		통계가 적용되는 컬럼 이름.
TYPE	CHAR(1)		• F = 빈도 값 • Q = Quantile 값
SEQNO	SMALLINT		TYPE = 'F'인 경우, 이 컬럼의 *n*이 *n*번째로 자주 사용되는 값을 식별합니다. TYPE = 'Q'인 경우, 이 컬럼의 *n*은 *n*번째 Quantile 값을 식별합니다.
COLVALUE[1]	VARCHAR(254)	Y	문자 리터럴 또는 널(NULL) 값으로서의 데이터 값.
VALCOUNT	BIGINT		TYPE = 'F'인 경우, VALCOUNT는 컬럼에서 COLVALUE의 어커런스 수입니다. TYPE = 'Q'인 경우, VALCOUNT는 값이 COLVALUE보다 작거나 같은 행 수입니다.
DISTCOUNT[2]	BIGINT	Y	TYPE = 'Q'인 경우, 이 컬럼은 COLVALUE보다 작거나 같은 구별 값 수를 기록합니다(사용 불가능한 경우 널(NULL) 값).

주:

1. 카탈로그 뷰에서 COLVALUE의 값이 항상 데이터베이스 코드 페이지에 표시되며 대체 문자를 포함할 수 있습니다. 그러나 통계는 컬럼 테이블의 코드 페이지에서 내부적으로 수집되며, 쿼리 최적화 중에 적용될 때 실제 컬럼 값을 사용합니다.
2. DISTCOUNT는 인덱스에서 첫 번째 키 컬럼인 컬럼의 경우에만 수집됩니다.

SYSCAT.COLGROUPCOLS

각 행은 컬럼 그룹을 구성하는 컬럼을 나타냅니다.

표 89. SYSCAT.COLGROUPCOLS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
COLGROUPID	INTEGER		컬럼 그룹 ID.
COLNAME	VARCHAR(128)		컬럼 그룹에 있는 컬럼의 이름.
TABSCHEMA	VARCHAR(128)		컬럼 그룹에 있는 컬럼에 대한 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		컬럼 그룹에 있는 컬럼에 대한 테이블의 규정되지 않은 이름.
ORDINAL	SMALLINT		컬럼 그룹에 있는 컬럼의 서수.

SYSCAT.COLGROUPDIST

각 행은 컬럼 그룹의 *n*번째로 자주 사용되는 값이나 컬럼 그룹의 *n*번째 Quantile 값으로 구성되는 컬럼 그룹의 컬럼 값을 나타냅니다.

표 90. SYSCAT.COLGROUPDIST 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
COLGROUPID	INTEGER		컬럼 그룹 ID.
TYPE	CHAR(1)		• F = 빈도 값 • Q = Quantile 값
ORDINAL	SMALLINT		컬럼 그룹에 있는 컬럼의 서수.
SEQNO	SMALLINT		TYPE = 'F'인 경우, 이 컬럼의 *n*이 *n*번째로 자주 사용되는 값을 식별합니다. TYPE = 'Q'인 경우, 이 컬럼의 *n*은 *n*번째 Quantile 값을 식별합니다.
COLVALUE[1]	VARCHAR(254)		문자 리터럴 또는 널(NULL) 값으로서의 데이터 값.

주:

1. 카탈로그 뷰에서 COLVALUE의 값이 항상 데이터베이스 코드 페이지에 표시되며 대체 문자를 포함할 수 있습니다. 그러나 통계는 컬럼 테이블의 코드 페이지에서 내부적으로 수집되며, 쿼리 최적화 중에 적용될 때 실제 컬럼 값을 사용합니다.

SYSCAT.COLGROUPDISTCOUNTS

각 행은 컬럼 그룹의 *n*번째로 자주 사용되는 값이나 컬럼 그룹의 *n*번째 Quantile에 적용되는 분산 통계를 나타냅니다.

표 91. SYSCAT.COLGROUPDISTCOUNTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
COLGROUPID	INTEGER		컬럼 그룹 ID.
TYPE	CHAR(1)		• F = 빈도 값 • Q = Quantile 값
SEQNO	SMALLINT		*n*번째 TYPE 값을 나타내는 시퀀스 번호 *n*.
VALCOUNT	BIGINT		TYPE = 'F'인 경우, VALCOUNT는 이 SEQNO를 갖는 컬럼 그룹에 대한 COLVALUE의 어커런스 수입니다. TYPE = 'Q'인 경우, VALCOUNT는 값이 이 SEQNO를 갖는 컬럼 그룹에 대한 COLVALUE보다 작거나 같은 행 수입니다.
DISTCOUNT	BIGINT		TYPE = 'Q'인 경우, 이 컬럼은 이 SEQNO를 갖는 컬럼 그룹에 대한 COLVALUE보다 작거나 같은 구별 값 수를 기록합니다(사용 불가능한 경우 널(NULL) 값).

SYSCAT.COLGROUPS

각 행은 전체 컬럼 그룹에 적용되는 컬럼 그룹 및 통계를 나타냅니다.

표 92. SYSCAT.COLGROUPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
COLGROUPSCHEMA	VARCHAR(128)		컬럼 그룹의 스키마 이름.
COLGROUPNAME	VARCHAR(128)		컬럼 그룹의 규정되지 않은 이름.
COLGROUPID	INTEGER		컬럼 그룹 ID.
COLGROUPCARD	BIGINT		컬럼 그룹의 카디널리티(cardinality).
NUMFREQ_VALUES	SMALLINT		컬럼 그룹에 대해 수집된 자주 사용되는 값 수.
NUMQUANTILES	SMALLINT		컬럼 그룹에 대해 수집된 Quantile 수.

SYSCAT.COLIDENTATTRIBUTES

각 행은 테이블에 대해 정의되는 ID 컬럼을 나타냅니다.

표 93. SYSCAT.COLIDENTATTRIBUTES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		컬럼을 포함하는 테이블 또는 뷰의 스키마 이름.
TABNAME	VARCHAR(128)		컬럼을 포함하는 테이블 또는 뷰의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		컬럼 이름
START	DECIMAL(31,0)	Y	시퀀스의 시작값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
INCREMENT	DECIMAL(31,0)	Y	증분 값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
MINVALUE	DECIMAL(31,0)	Y	시퀀스의 최소값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
MAXVALUE	DECIMAL(31,0)	Y	시퀀스의 최대값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
CYCLE	CHAR(1)		시퀀스가 최대 또는 최소값에 도달한 후 계속 값을 생성할 수 있는지 여부를 표시합니다. • N = 시퀀스가 순환할 수 없음 • Y = 시퀀스가 순환할 수 있음 • 공백 = 시퀀스가 별명임
CACHE	INTEGER		보다 빠른 액세스를 위해 메모리에 사전 할당할 시퀀스 값의 수입니다. 0은 시퀀스 값이 사전 할당되지 않음을 표시합니다. 파티션된 데이터베이스에서는 이 값이 각 데이터베이스 파티션에 적용됩니다. 시퀀스가 별명인 경우 -1입니다.
ORDER	CHAR(1)		시퀀스 번호를 요청 순서대로 생성해야 하는지 여부를 표시합니다. • N = 시퀀스 번호가 요청 순서대로 생성될 필요가 없음 • Y = 시퀀스 번호를 요청 순서대로 생성해야 함 • 공백 = 시퀀스가 별명임
NEXTCACHEFIRSTVALUE	DECIMAL(31,0)	Y	다음 캐시 블록에서 지정될 수 있는 첫 번째 값입니다. 캐싱이 없는 경우 지정할 수 있는 다음 값입니다.
SEQID	INTEGER		시퀀스 또는 별명의 ID.

SYSCAT.COLOPTIONS

각 행은 컬럼 특정 옵션 값을 포함합니다.

표 94. SYSCAT.COLOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		별칭의 스키마 이름.
TABNAME	VARCHAR(128)		옵션이 설정되는 컬럼의 별칭.
COLNAME	VARCHAR(128)		로컬 컬럼 이름.
OPTION	VARCHAR(128)		컬럼 옵션의 이름.
SETTING	CLOB(32K)		값.

SYSCAT.COLUMNS

각 행은 테이블, 뷰 또는 별칭에 정의된 컬럼을 나타냅니다.

표 95. SYSCAT.COLUMNS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		컬럼을 포함하는 테이블, 뷰 또는 별칭의 스키마 이름.
TABNAME	VARCHAR(128)		컬럼을 포함하는 테이블, 뷰 또는 별칭의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		컬럼 이름
COLNO	SMALLINT		테이블에서 이 컬럼의 번호(0에서 시작).
TYPESCHEMA	VARCHAR(128)		컬럼에 대한 데이터 유형의 스키마 이름.
TYPENAME	VARCHAR(128)		컬럼에 대한 데이터 유형의 규정되지 않은 이름.
LENGTH	INTEGER		데이터의 최대 길이. 구별 유형의 경우 0입니다. LENGTH 컬럼은 DECIMAL 필드의 경우 정밀도를 표시하고, 10진수 부동 소수점 컬럼에 필요한 스토리지의 바이트 수(DECFLOAT(16) 및 DECFLOAT(34)의 경우 각각 8과 16)를 표시합니다.
SCALE	SMALLINT		컬럼 유형이 DECIMAL인 경우 스케일, 컬럼 유형이 TIMESTAMP인 경우 분수 초의 자릿수, 그렇지 않으면 0입니다.
DEFAULT[1]	VARCHAR(254)	Y	테이블의 컬럼에 대한 디폴트값은 컬럼의 데이터 유형에 적합한 상수, 특수 레지스터 또는 캐스트 함수로 표현됩니다. 키워드 NULL일 수도 있습니다. 값은 디폴트값으로 지정된 값에서 변환될 수 있습니다. 예를 들어 날짜 및 시간 상수는 ISO 형식으로 표시되고, 캐스트 함수 이름은 스키마 이름으로 규정되며, ID는 구분됩니다. DEFAULT절이 지정되지 않았거나 컬럼이 뷰 컬럼인 경우 널(NULL) 값입니다.
NULLS[2]	CHAR(1)		컬럼의 널(null) 값 허용 속성. • N = 컬럼에 널(NULL)을 입력할 수 없음 • Y = 컬럼에 널(NULL) 입력 가능 표현식 또는 함수로부터 파생되는 뷰 컬럼의 경우 값이 'N'일 수 있습니다. 그러나 그러한 컬럼은 뷰를 사용하는 명령문이 산술 오류에 대한 경고와 함께 처리될 때 널(NULL) 값을 허용합니다.
CODEPAGE	SMALLINT		이 컬럼의 데이터에 사용되는 코드 페이지이며, 컬럼이 FOR BIT DATA로 정의되거나 문자열 유형이 아닌 경우 0입니다.
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 컬럼에 대한 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.

표 95. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 컬럼에 대한 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
LOGGED	CHAR(1)		유형이 LOB이거나 LOB에 기반한 구별인 컬럼에만 적용됩니다. 그렇지 않으면 공백입니다. • N = 컬럼이 로그되지 않음 • Y = 컬럼이 로그됨
COMPACT	CHAR(1)		유형이 LOB이거나 LOB에 기반한 구별인 컬럼에만 적용됩니다. 그렇지 않으면 공백입니다. • N = 컬럼이 간략화되지 않음 • Y = 컬럼이 스토리지에서 간략화됨
COLCARD	BIGINT		컬럼에 있는 구별 값의 수입니다. 통계가 수집되지 않는 경우 -1이며, 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 -2입니다.
HIGH2KEY[3]	VARCHAR(254)	Y	두 번째로 높은 데이터 값. 문자 리터럴로 변경된 숫자 데이터의 표시입니다. 통계가 수집되지 않는 경우 비어 있습니다. 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 비어 있습니다.
LOW2KEY[3]	VARCHAR(254)	Y	두 번째로 낮은 데이터 값. 문자 리터럴로 변경된 숫자 데이터의 표시입니다. 통계가 수집되지 않는 경우 비어 있습니다. 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 비어 있습니다.
AVGCOLLEN	INTEGER		컬럼이 데이터베이스 메모리 또는 임시 테이블에 저장될 때 바이트 단위의 평균 스페이스입니다. 인라인되지 않는 LOB 데이터 유형, LONG 데이터 유형 및 XML 문서의 경우 평균 컬럼 길이를 계산하는 데 사용되는 값은 데이터 디스크립터의 길이입니다. 컬럼이 널(NULL) 입력 가능한 경우 추가 바이트가 필요합니다. 통계가 수집되지 않은 경우 -1이며, 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 -2입니다. 주: 디스크에 컬럼을 저장하기 위해 필요한 평균 스페이스는 이 통계로 표시되는 값과 다를 수 있습니다.
KEYSEQ	SMALLINT	Y	테이블의 기본 키에 있는 컬럼의 숫자 위치입니다. 서브테이블 및 계층 구조 테이블의 컬럼의 경우 널(NULL) 값입니다.
PARTKEYSEQ	SMALLINT	Y	테이블의 분산 키에 있는 컬럼의 숫자 위치입니다. 컬럼이 분산 키에 있지 않은 경우 0 또는 널(NULL) 값입니다. 서브테이블 및 계층 구조 테이블의 컬럼의 경우 널(NULL) 값입니다.
NQUANTILES	SMALLINT		이 컬럼에 대한 SYSCAT.COLDIST에 기록되는 Quantile 값의 수입니다. 통계가 수집되지 않는 경우 -1이고, 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 -2입니다.

표 95. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
NMOSTFREQ	SMALLINT		이 컬럼에 대한 SYSCAT.COLDIST에 기록되는 빈도가 가장 높은 값의 수입니다. 통계가 수집되지 않는 경우 -1이고, 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 -2입니다.
NUMNULLS	BIGINT		컬럼에 있는 널(NULL) 값의 수입니다. 통계가 수집되지 않는 경우 -1입니다.
TARGET_TYPESCHEMA	VARCHAR(128)	Y	이 컬럼의 유형이 REFERENCE인 경우 목표 행 유형의 스키마 이름이고, 그렇지 않으면 널(NULL) 값입니다.
TARGET_TYPENAME	VARCHAR(128)	Y	이 컬럼의 유형이 REFERENCE인 경우 목표 행 유형의 규정되지 않은 이름이고, 그렇지 않으면 널(NULL) 값입니다.
SCOPE_TABSCHEMA	VARCHAR(128)	Y	이 컬럼의 유형이 REFERENCE인 경우 범위(목표 테이블)의 스키마 이름이고, 그렇지 않으면 널(NULL) 값입니다.
SCOPE_TABNAME	VARCHAR(128)	Y	이 컬럼의 유형이 REFERENCE인 경우 범위(목표 테이블)의 규정되지 않은 이름이고, 그렇지 않으면 널(NULL) 값입니다.
SOURCE_TABSCHEMA	VARCHAR(128)	Y	유형이 지정된 테이블 또는 뷰의 컬럼의 경우, 컬럼이 처음 도입된 테이블 또는 뷰의 스키마 이름입니다. 상속되지 않은 컬럼의 경우 이것은 TABSCHEMA와 동일합니다. 유형이 지정되지 않은 테이블 및 뷰의 컬럼의 경우 널(NULL) 값입니다.
SOURCE_TABNAME	VARCHAR(128)	Y	유형이 지정된 테이블 또는 뷰의 컬럼의 경우, 컬럼이 처음 도입된 테이블 또는 뷰의 규정되지 않은 이름입니다. 상속되지 않은 컬럼의 경우 이것은 TABNAME과 동일합니다. 유형이 지정되지 않은 테이블 및 뷰의 컬럼의 경우 널(NULL) 값입니다.
DL_FEATURES	CHAR(10)	Y	이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다.
SPECIAL_PROPS	CHAR(8)	Y	REFERENCE 유형 컬럼에만 적용되며, 그렇지 않으면 공백입니다. 각 바이트 위치는 다음과 같이 정의됩니다. • 1 = 오브젝트 ID(OID) 컬럼('Y' = 예, 'N' = 아니오) • 2 = 사용자 생성 또는 시스템 생성('U' = 사용자, 'S' = 시스템) 바이트 3 - 8은 나중에 사용하기 위해 예약되어 있습니다.
HIDDEN	CHAR(1)		숨겨진 컬럼의 유형 • I = 컬럼이 IMPLICITLY HIDDEN으로 정의됨 • S = 시스템이 관리하는 숨겨진 컬럼 • 공백 = 컬럼이 숨겨져 있지 않음

표 95. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
INLINE_LENGTH	INTEGER		기본 테이블에 저장할 수 있는 XML 문서, 구조화된 유형 또는 LOB 데이터 유형의 인스턴스 내부 표현의 최대 *크기*(바이트)이며, 적용할 수 없는 경우 0입니다.
PCTINLINED	SMALLINT		인라인된 XML 문서 또는 LOB 데이터의 백분율입니다. 통계가 수집되지 않은 경우 -1입니다.
IDENTITY	CHAR(1)		• N = ID 컬럼이 아님 • T = 행 변경 시간소인 컬럼 • Y = ID 컬럼
ROWCHANGETIMESTAMP	CHAR(1)		• N = 행 변경 시간소인 컬럼이 아님 • Y = 행 변경 시간소인 컬럼
GENERATED	CHAR(1)		생성된 컬럼의 유형. • A = 컬럼 값이 항상 생성됨 • D = 컬럼 값이 디폴트로 생성됨 • 공백 = 컬럼이 생성되지 않음
TEXT	CLOB(2M)	Y	표현식으로 생성되는 것으로 정의되는 컬럼의 경우 이 필드에 키워드 AS로 시작하는 생성된 컬럼 표현식의 텍스트가 있습니다.
COMPRESS	CHAR(1)		• O = 압축 해제 • S =시스템 디폴트값 압축
AVGDISTINCTPERPAGE	DOUBLE	Y	나중에 사용함.
PAGEVARIANCERATIO	DOUBLE	Y	나중에 사용함.
SUB_COUNT	SMALLINT		컬럼의 평균 하위 요소 수입니다. 문자열 컬럼에만 적용 가능합니다.
SUB_DELIM_LENGTH	SMALLINT		컬럼의 각 하위 요소를 분리하는 분리문자의 평균 길이입니다. 문자열 컬럼에만 적용 가능합니다.
AVGCOLLENCHAR	INTEGER		컬럼에 필요한 (컬럼에 영향을 미치는 조합을 기반으로) 평균 문자 수는 컬럼의 데이터 유형이 길거나, LOB, XML 또는 통계가 수집되지 않은 경우 -1이며 상속된 컬럼 및 계층 테이블의 컬럼의 경우 -2입니다.
IMPLICITVALUE[4]	VARCHAR(254)	Y	테이블이 작성된 후에 테이블에 추가된 컬럼의 경우 컬럼이 추가된 시간의 디폴트값을 저장합니다. 테이블이 작성될 때 정의된 컬럼의 경우 널(NULL) 값을 저장합니다.
SECLABELNAME	VARCHAR(128)	Y	보호 컬럼의 경우 컬럼과 연관되는 보안 레이블의 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

표 95. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명

주:

1. 버전 2.1.0의 경우, 캐스트 함수 이름이 구분되지 않았으며 DEFAULT 컬럼에서 이 방식으로 나타날 수 있습니다. 또한 일부 뷰 컬럼에는 DEFAULT 컬럼에 여전히 나타나는 디폴트값이 포함되었습니다.
2. 버전 2부터 값 D(디폴트에서 널 없음 표시)가 더 이상 사용되지 않습니다. 대신 WITH DEFAULT의 사용이 DEFAULT 컬럼에서 널(NULL)이 아닌 값으로 표시됩니다.
3. 카탈로그 뷰에서 HIGH2KEY 및 LOW2KEY의 값이 항상 데이터베이스 코드 페이지에 표시되며 대체 문자를 포함할 수 있습니다. 그러나 통계는 컬럼 테이블의 코드 페이지에서 내부적으로 수집되며, 쿼리 최적화 중에 적용될 때 실제 컬럼 값을 사용합니다.
4. 특정 컬럼에 대한 IMPLICITVALUE가 소스 컬럼 및 목표 컬럼 모두에 대해 널(NULL)이 아닌 값이고 값이 일치하지 않는 경우 데이터 파티션 첨부가 허용됩니다. 이 경우 소스 테이블을 삭제(drop)한 후 다시 작성해야 합니다. 다음 조건 중 하나가 만족되는 경우 컬럼은 IMPLICITVALUE 필드에 널(NULL)이 아닌 값을 가질 수 있습니다.
   - 컬럼이 ALTER TABLE...ADD COLUMN문의 결과로 작성됩니다.
   - IMPLICITVALUE 필드가 첨부 중에 소스 테이블로부터 전파됩니다.
   - IMPLICITVALUE 필드가 접속 해제 중에 소스 테이블로부터 상속됩니다.
   - IMPLICITVALUE 필드가 버전 8에서 버전 9로의 데이터베이스 업그레이드 중에 설정됩니다. 여기에서 추가된 컬럼인 것으로 판별되거나 추가된 컬럼일 수 있습니다. 데이터베이스가 컬럼이 추가되는지 여부를 확실히 알 수 없는 경우 추가된 것으로 처리됩니다. 추가된 컬럼은 ALTER TABLE...ADD COLUMN문의 결과로 작성된 컬럼입니다.

   이주가 아닌 시나리오 중에 이러한 불일치를 피하기 위해 항상 이미 정의된 모든 컬럼을 사용하여 첨부하려는 테이블을 작성하는 것이 바람직합니다. 즉 ALTER TABLE문을 사용하여 첨부하기 전에 테이블에 컬럼을 추가하지 마십시오.

SYSCAT.COLUSE

각 행은 CREATE TABLE문의 DIMENSIONS절에서 참조한 컬럼을 나타냅니다.

표 96. SYSCAT.COLUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		컬럼을 포함하는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		컬럼을 포함하는 테이블의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		컬럼 이름
DIMENSION	SMALLINT		DIMENSIONS절에 지정된 차원의 순서를 기초로 하는 차원 번호입니다(초기 위치는 0임). 복합 차원의 경우 이 값은 차원의 각 구성요소에 대해 동일합니다.
COLSEQ	SMALLINT		컬럼이 속하는 차원에서 컬럼의 숫자 위치입니다(초기 위치는 0임). 비복합 차원의 단일 컬럼의 경우 값은 0입니다.
TYPE	CHAR(1)		차원 유형. • C = 클러스터링 또는 다차원 클러스터링 • P = 파티셔닝

SYSCAT.CONDITIONS

각 행은 모듈에 정의된 조건을 나타냅니다.

표 97. SYSCAT.CONDITIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONDSCHEMA	VARCHAR(128)		조건의 스키마 이름.
CONDMODULENAME	VARCHAR(128)	Y	조건이 속하는 모듈의 규정되지 않은 이름
CONDNAME	VARCHAR(128)		조건의 규정되지 않은 이름.
CONDID	INTEGER		조건 ID.
CONDMODULEID	INTEGER	Y	조건이 속하는 모듈의 ID.
SQLSTATE	CHAR(5)	Y	조건과 연관된 SQLSTATE 값.
OWNER	VARCHAR(128)		조건 소유자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
CREATE_TIME	TIMESTAMP		조건이 작성된 시간.
REMARKS	VARCHAR(254)		사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.CONSTDEP

각 행은 일부 다른 오브젝트에 대한 제한조건의 종속성을 나타냅니다. 제한조건은 이름 BNAME의 유형 BTYPE의 오브젝트에 따라 달라지므로, 오브젝트에 대한 변경사항이 제한조건에 영향을 줍니다.

표 98. SYSCAT.CONSTDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONSTNAME	VARCHAR(128)		제한조건의 규정되지 않은 이름.
TABSCHEMA	VARCHAR(128)		제한조건이 적용되는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		제한조건이 적용되는 테이블의 규정되지 않은 이름.
BTYPE	CHAR(1)		제한조건이 종속되는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • F = 루틴 • I = 인덱스 • R = 사용자 정의 구조화된 유형 • u = 모듈 별명
BSCHEMA	VARCHAR(128)		제한조건이 종속되는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.
BNAME	VARCHAR(128)		제한조건이 종속되는 오브젝트의 규정되지 않은 이름.
BMODULEID	INTEGER	Y	제한조건이 종속되는 오브젝트의 모듈에 대한 ID.

SYSCAT.CONTEXTATTRIBUTES

각 행은 트러스트된 컨텍스트 속성을 나타냅니다.

표 99. SYSCAT.CONTEXTATTRIBUTES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONTEXTNAME	VARCHAR(128)		트러스트된 컨텍스트 이름.
ATTR_NAME	VARCHAR(128)		속성의 이름입니다. 다음 중 하나 • ADDRESS • ENCRYPTION
ATTR_VALUE	VARCHAR(128)		속성 값.
ATTR_OPTIONS	VARCHAR(128)	Y	ATTR_NAME이 'ADDRESS'인 경우, 이 특정 주소에 대해 필요한 암호화 수준을 지정합니다. 널(NULL) 값은 전역 ENCRYPTION 속성이 적용됨을 표시합니다.

SYSCAT.CONTEXTS

각 행은 트러스트된 컨텍스트를 나타냅니다.

표 100. SYSCAT.CONTEXTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONTEXTNAME	VARCHAR(128)		트러스트된 컨텍스트 이름.
CONTEXTID	INTEGER		트러스트된 컨텍스트 ID.
SYSTEMAUTHID	VARCHAR(128)		트러스트된 컨텍스트와 연관된 시스템 권한 부여 ID.
DEFAULTCONTEXTROLE	VARCHAR(128)	Y	컨텍스트의 디폴트 역할.
CREATE_TIME	TIMESTAMP		트러스트된 컨텍스트가 작성된 시간.
ALTER_TIME	TIMESTAMP		트러스트된 컨텍스트가 마지막으로 변경된 시간.
ENABLED	CHAR(1)		트러스트된 컨텍스트 상태. • N = 사용 불가능 • Y = 사용 가능
AUDITPOLICYID	INTEGER	Y	감사 규정 ID.
AUDITPOLICYNAME	VARCHAR(128)	Y	감사 규정 이름.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.DATAPARTITIONEXPRESSION

각 행은 테이블 파티셔닝 키의 해당 파트에 대한 표현식을 나타냅니다.

표 101. SYSCAT.DATAPARTITIONEXPRESSION 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		파티션된 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		파티션된 테이블의 규정되지 않은 이름.
DATAPARTITIONKEYSEQ	INTEGER		1에서 시작하는 표현식 키 파트 시퀀스.
DATAPARTITIONEXPRESSION	CLOB(32K)		SQL 구문의 시퀀스에서 이 항목에 대한 표현식.
NULLSFIRST	CHAR(1)		• N = 이 표현식의 널(NULL) 값이 비교적 높음 • Y = 이 표현식의 널(NULL) 값이 비교적 낮음

SYSCAT.DATAPARTITIONS

각 행은 데이터 파티션을 나타냅니다. 메모:

- 테이블이 여러 개의 데이터베이스 파티션에서 작성된 경우 데이터 파티션 통계는 한 개의 데이터베이스 파티션을 나타냅니다.

표 102. SYSCAT.DATAPARTITIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
DATAPARTITIONNAME	VARCHAR(128)		데이터 파티션 이름.
TABSCHEMA	VARCHAR(128)		이 데이터 파티션이 속하는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		이 데이터 파티션이 속하는 테이블의 규정되지 않은 이름.
DATAPARTITIONID	INTEGER		데이터 파티션 ID.
TBSPACEID	INTEGER	Y	이 데이터 파티션이 저장되는 테이블 스페이스의 ID입니다. STATUS가 'I'일 때는 널(NULL) 값입니다.
PARTITIONOBJECTID	INTEGER	Y	테이블 스페이스 내의 데이터 파티션 ID.
LONG_TBSPACEID	INTEGER	Y	Long 데이터가 저장된 테이블 스페이스 ID. STATUS가 'I'일 때는 널(NULL) 값입니다.
ACCESS_MODE	CHAR(1)		데이터 파티션의 액세스 제한 상태입니다. 이러한 상태는 무결성 설정 보류 상태에 있는 오브젝트나 SET INTEGRITY문으로 처리된 오브젝트에만 적용됩니다. 가능한 값은 다음과 같습니다. • D = 데이터 이동 없음 • F = 전체 액세스 • N = 권한 없음 • R = 읽기 전용 액세스
STATUS	VARCHAR(32)		• A = 데이터 파티션이 새로 접속됨 • D = 데이터 파티션이 접속 해제됨 • I = 카탈로그의 항목이 비동기 인덱스 정리 중에만 유지보수되는 접속 해제된 데이터 파티션. STATUS 값이 'I'인 행은 접속 해제된 파티션을 참조하는 모든 인덱스 레코드가 삭제되었을 때 제거됨 • 비어 있는 문자열 = 데이터 파티션을 볼 수 있음(정상 상태) 바이트 2 - 32는 나중에 사용하기 위해 예약됩니다.
SEQNO	INTEGER		데이터 파티션 시퀀스 번호(0에서 시작).
LOWINCLUSIVE	CHAR(1)		• N = 낮은 키 값이 포함되지 않음 • Y = 낮은 키 값이 포함됨
LOWVALUE	VARCHAR(512)		이 데이터 파티션에 대한 낮은 키 값(SQL 값의 문자열 표시).

표 102. SYSCAT.DATAPARTITIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
HIGHINCLUSIVE	CHAR(1)		• N = 높은 키 값이 포함되지 않음 • Y = 높은 키 값이 포함됨
HIGHVALUE	VARCHAR(512)		이 데이터 파티션에 대한 높은 키 값(SQL 값의 문자열 표시).
CARD	BIGINT		데이터 파티션에 있는 총 행 수이며, 통계가 수집되지 않는 경우 -1입니다.
OVERFLOW	BIGINT		데이터 파티션에 있는 오버플로우 레코드의 총수이며, 통계가 수집되지 않는 경우 -1입니다.
NPAGES	BIGINT		데이터 파티션의 행이 존재하는 페이지의 총수입니다. 통계가 수집되지 않는 경우 -1입니다.
FPAGES	BIGINT		데이터 파티션에 있는 페이지의 총수이며, 통계가 수집되지 않는 경우 -1입니다.
ACTIVE_BLOCKS	BIGINT		데이터 파티션에 있는 활성 블록의 총수 또는 -1입니다. 다차원적으로 클러스터된(MDC) 테이블에만 적용됩니다.
INDEX_TBSPACEID	INTEGER		이 데이터 파티션에 대한 모든 파티션된 인덱스를 보유하는 테이블 스페이스의 ID.
AVGROWSIZE	SMALLINT		이 데이터 파티션에 있는 압축 및 압축되지 않은 행 모두의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
PCTROWSCOMPRESSED	REAL		데이터 파티션에 있는 전체 행 수의 백분율로 표현되는 압축된 행입니다. 통계가 수집되지 않는 경우 -1입니다.
PCTPAGESAVED	SMALLINT		행 압축의 결과로 데이터 파티션에 저장되는 페이지의 대략적인 백분율입니다. 이 값에는 데이터 파티션의 각 사용자 데이터에 대한 오버헤드 바이트가 포함되지만 사전 오버헤드가 이용하는 스페이스는 포함되지 않습니다. 통계가 수집되지 않는 경우 -1입니다.
AVGCOMPRESSEDROWSIZE	SMALLINT		이 데이터 파티션에 있는 압축된 행의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
AVGROWCOMPRESSIONRATIO	REAL		데이터 파티션의 압축된 행의 경우 이는 행별 평균 압축비입니다. 즉, 평균 압축되지 않은 행 길이를 평균 압축 행 길이로 나눈 값입니다. 통계가 수집되지 않는 경우 -1입니다.
STATS_TIME	TIMESTAMP	Y	이 오브젝트에 대해 기록된 통계가 마지막으로 변경된 시간입니다. 통계가 수집되지 않은 경우 널(NULL)입니다.
LASTUSED	DATE		나중에 사용하기 위해 예약됨

SYSCAT.DATATYPEDEP

각 행은 일부 다른 오브젝트에 대한 사용자 정의 데이터 유형의 종속성을 나타냅니다.

표 103. SYSCAT.DATATYPEDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPESCHEMA	VARCHAR(128)		데이터 유형의 스키마 이름.
TYPEMODULENAME	VARCHAR(128)	Y	데이터 유형의 모듈 이름.
TYPENAME	VARCHAR(128)		데이터 유형의 규정되지 않은 이름.
TYPEMODULEID	INTEGER	Y	데이터 유형의 모듈 ID.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • N = 별칭 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트의 모듈 이름.
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE = 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 데이터 유형에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.DATATYPES

각 행은 내장 또는 사용자 정의 데이터 유형을 나타냅니다.

표 104. SYSCAT.DATATYPES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPESCHEMA	VARCHAR(128)		TYPEMODULEID가 널(Null)인 경우 데이터 유형의 스키마 이름이고, 그렇지 않으면 데이터 유형이 속하는 모듈의 스키마 이름입니다.
TYPEMODULENAME	VARCHAR(128)	Y	사용자 정의 유형이 속하는 모듈의 규정되지 않은 이름. 모듈 사용자 정의 유형이 아니면 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)		데이터 유형의 규정되지 않은 이름.
OWNER	VARCHAR(128)		유형 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
SOURCESCHEMA	VARCHAR(128)	Y	구별 유형 또는 배열 유형의 경우, 소스 데이터 유형의 스키마 이름입니다. 사용자 정의 구조화된 유형의 경우, 참조 표현 유형의 내장 유형의 스키마 이름입니다. 기타 데이터 유형의 경우 널(NULL)입니다.
SOURCEMODULENAME	VARCHAR(128)	Y	소스 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 모듈 소스 데이터 유형이 아니면 널(NULL) 값입니다.
SOURCENAME	VARCHAR(128)	Y	구별 유형 또는 배열 유형의 경우, 소스 데이터 유형의 규정되지 않은 이름입니다. 사용자 정의 구조화된 유형의 경우, 참조 표현 유형의 규정되지 않은 내장 유형 이름입니다. 기타 데이터 유형의 경우 널(NULL)입니다.
METATYPE	CHAR(1)		• A = 사용자 정의 배열 유형 • C = 사용자 정의 커서 유형 • F = 사용자 정의 행 유형 • L = 사용자 정의 연관된 배열 유형 • R = 사용자 정의 구조화된 유형 • S = 시스템 사전 정의 유형 • T = 사용자 정의 구별 유형
TYPEID	SMALLINT		데이터 유형 ID.
TYPEMODULEID	INTEGER	Y	사용자 정의 유형이 속하는 모듈의 ID입니다. 모듈 사용자 정의 유형이 아니면 널(NULL) 값입니다.
SOURCETYPEID	SMALLINT	Y	소스 유형에 대한 ID입니다(내장 유형의 경우 널(NULL) 값). 사용자 정의 구조화된 유형의 경우 참조 표시 유형의 ID입니다.
SOURCEMODULEID	INTEGER	Y	소스 데이터 유형이 속하는 모듈의 ID입니다. 모듈 소스 데이터 유형이 아니면 널(NULL) 값입니다.

표 104. SYSCAT.DATATYPES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
PUBLISHED	CHAR(1)		모듈 사용자 정의 유형을 모듈 밖에서 참조할 수 있는지 여부를 표시합니다. • N = 모듈 사용자 정의 유형이 발행되지 않음 • Y = 모듈 사용자 정의 유형이 발행됨 • 공백 = 적용할 수 없음
LENGTH	INTEGER		유형의 최대 길이. 내장 매개변수화된 유형(예: DECIMAL 및 VARCHAR)의 경우 0입니다. 사용자 정의 구조화된 유형의 경우 참조 표시 유형의 길이입니다.
SCALE	SMALLINT		내장 DECIMAL 유형을 기반으로 하는 구별 유형 또는 참조 표시 유형의 스케일은 내장 TIMESTAP 유형을 기반으로 하는 구별 유형은 소수 초의 숫자 수이며 내장 TIMESTAMP 유형은 6이며 DECIMAL 자체를 포함하여 다른 모든 유형은 0입니다.
CODEPAGE	SMALLINT		문자열 유형, 문자열 유형 기반 구별 유형 또는 참조 표시 유형의 경우 데이터베이스 코드 페이지이고, 그렇지 않으면 0입니다.
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 데이터 유형에 대한 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 데이터 유형에 대한 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
ARRAY_LENGTH	INTEGER	Y	배열의 최대 카디널리티(cardinality)입니다. METATYPE이 'A'가 아닌 경우 널(NULL) 값입니다.
ARRAYINDEXTYPESCHEMA	VARCHAR(128)	Y	배열 인덱스의 데이터 유형의 스키마입니다. METATYPE이 'L'이 아닌 경우 널(NULL) 값입니다.
ARRAYINDEXTYPENAME	VARCHAR(128)	Y	배열 인덱스의 데이터 유형 이름입니다. METATYPE이 'L'이 아닌 경우 널(NULL) 값입니다.
ARRAYINDEXTYPEID	SMALLINT	Y	배열 인덱스 유형의 경우 ID입니다. METATYPE이 'L'이 아닌 경우 널(NULL) 값입니다.
ARRAYINDEXTYPELENGTH	INTEGER	Y	배열 인덱스 데이터 유형의 최대 길이입니다. METATYPE이 'L'이 아닌 경우 널(NULL) 값입니다.
CREATE_TIME	TIMESTAMP		데이터 유형의 작성 시간.
VALID	CHAR(1)		• N = 데이터 유형이 유효하지 않음 • Y = 데이터 유형이 유효함
ATTRCOUNT	SMALLINT		데이터 유형에 있는 속성의 수.

표 104. SYSCAT.DATATYPES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
INSTANTIABLE	CHAR(1)		• N = 유형이 인스턴스화될 수 없음 • Y = 유형이 인스턴스화될 수 있음
WITH_FUNC_ACCESS	CHAR(1)		• N = 이 유형에 대한 메소드는 함수 표기를 사용하여 호출할 수 없습니다. • Y = 이 유형에 대한 모든 메소드를 함수 표기를 사용하여 호출할 수 있습니다.
FINAL	CHAR(1)		• N = 사용자 정의 유형이 부속 유형을 가질 수 있습니다. • Y = 사용자 정의 유형이 부속 유형을 가질 수 없습니다.
INLINE_LENGTH	INTEGER		기본 테이블 행으로 보존할 수 있는 구조화된 유형의 최대 길이이며, 그렇지 않으면 0입니다.
NATURAL_ INLINE_LENGTH	INTEGER	Y	구조화된 유형 인스턴스의 시스템 생성 기본 인라인 길이입니다. 이 유형이 구조화된 유형이 아닌 경우 널(NULL) 값입니다.
JARSCHEMA	VARCHAR(128)	Y	SQL 유형을 구현하는 Java 클래스가 포함된 Jar 파일을 식별하는 JAR_ID의 스키마 이름입니다. EXTERNAL NAME절이 지정되지 않은 경우 널(NULL) 값입니다.
JAR_ID	VARCHAR(128)	Y	SQL 유형을 구현하는 Java 클래스가 포함된 Jar 파일의 ID입니다. EXTERNAL NAME절이 지정되지 않은 경우 널(NULL) 값입니다.
CLASS	VARCHAR (384)	Y	SQL 유형을 구현하는 Java 클래스입니다. EXTERNAL NAME절이 지정되지 않은 경우 널(NULL) 값입니다.
SQLJ_REPRESENTATION	CHAR(1)	Y	SQL 유형을 구현하는 Java 클래스의 SQLJ ″representation_spec″입니다. EXTERNAL NAME ... LANGUAGE JAVA REPRESENTATION SPEC절이 지정되지 않는 경우 널(NULL) 값입니다. • D = SQL 데이터 • S = 직렬화 가능
ALTER_TIME	TIMESTAMP		데이터 유형이 마지막으로 변경된 시간.
DEFINER[1]	VARCHAR(128)		유형 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.DBAUTH

각 행은 하나 이상의 데이터 기본 레벨 권한이 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 105. SYSCAT.DBAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		권한의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		권한의 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
BINDADDAUTH	CHAR(1)		패키지를 작성할 권한. • N = 보유되지 않음 • Y = 보유됨
CONNECTAUTH	CHAR(1)		데이터베이스에 연결할 권한. • N = 보유되지 않음 • Y = 보유됨
CREATETABAUTH	CHAR(1)		테이블을 작성할 권한. • N = 보유되지 않음 • Y = 보유됨
DBADMAUTH	CHAR(1)		DBADM 권한 • N = 보유되지 않음 • Y = 보유됨
EXTERNALROUTINEAUTH	CHAR(1)		외부 루틴을 작성할 권한. • N = 보유되지 않음 • Y = 보유됨
IMPLSCHEMAAUTH	CHAR(1)		존재하지 않는 스키마에 오브젝트를 작성하여 내재적으로 스키마를 작성할 권한. • N = 보유되지 않음 • Y = 보유됨
LOADAUTH	CHAR(1)		DB2 로드 유틸리티를 사용할 권한 • N = 보유되지 않음 • Y = 보유됨
NOFENCEAUTH	CHAR(1)		비분리 사용자 정의 함수를 작성할 권한. • N = 보유되지 않음 • Y = 보유됨

표 105. SYSCAT.DBAUTH 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
QUIESCECONNECTAUTH	CHAR(1)		Quiesce 상태에서 데이터베이스에 액세스할 권한. • N = 보유되지 않음 • Y = 보유됨
LIBRARYADMAUTH	CHAR(1)		나중에 사용하기 위해 예약됨
SECURITYADMAUTH	CHAR(1)		데이터베이스 보안을 관리할 권한. • N = 보유되지 않음 • Y = 보유됨
SQLADMAUTH	CHAR(1)		SQL문을 모니터하고 조정하는 권한. • N = 보유되지 않음 • Y = 보유됨
WLMADMAUTH	CHAR(1)		WLM 오브젝트를 관리할 권한. • N = 보유되지 않음 • Y = 보유됨
EXPLAINAUTH	CHAR(1)		명령문에서 오브젝트에 대한 실제 특권을 사용하지 않고 SQL문을 Explain하는 권한. • N = 보유되지 않음 • Y = 보유됨
DATAACCESSAUTH	CHAR(1)		데이터에 액세스할 권한. • N = 보유되지 않음 • Y = 보유됨
ACCESSCTRLAUTH	CHAR(1)		데이터베이스 오브젝트 특권을 권한 부여 및 권한 취소할 권한. • N = 보유되지 않음 • Y = 보유됨

SYSCAT.DBPARTITIONGROUPDEF

각 행은 데이터베이스 파티션 그룹에 포함되는 데이터베이스 파티션을 나타냅니다.

표 106. SYSCAT.DBPARTITIONGROUPDEF 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
DBPGNAME	VARCHAR(128)		데이터베이스 파티션을 포함하는 데이터베이스 파티션 그룹의 이름.
DBPARTITIONNUM	SMALLINT		데이터베이스 파티션 그룹에 포함되는 데이터베이스 파티션의 파티션 번호입니다. 유효한 파티션 번호는 0 - 999입니다(경계 포함).
IN_USE	CHAR(1)		데이터베이스 파티션의 상태. • A = 새로 추가된 데이터베이스 파티션이 분산 맵에 없지만, 데이터베이스 파티션 그룹의 테이블 스페이스에 대한 컨테이너가 작성됩니다. 데이터베이스 파티션 그룹 재분배 조작이 완료되었을 때 데이터베이스 파티션이 분산 맵에 추가됩니다. • D = 데이터베이스 파티션 그룹 재분배 조작이 완료될 때 데이터베이스 파티션이 삭제됩니다. • T = 새로 추가된 데이터베이스 파티션이 분산 맵에 없으므로 WITHOUT TABLESPACES절을 사용하여 추가되었습니다. 컨테이너는 데이터베이스 파티션 그룹의 테이블 스페이스에 추가되어야 합니다. • Y = 데이터베이스 파티션이 분산 맵에 있습니다.

SYSCAT.DBPARTITIONGROUPS

각 행은 데이터베이스 파티션 그룹을 나타냅니다.

표 107. SYSCAT.DBPARTITIONGROUPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
DBPGNAME	VARCHAR(128)		데이터베이스 파티션 그룹 이름
OWNER	VARCHAR(128)		데이터베이스 파티션 그룹 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS 카탈로그 뷰의 분산 맵에 대한 ID.
REDISTRIBUTE_PMAP_ID	SMALLINT		현재 재분배에 사용되고 있는 분산 맵에 대한 ID입니다. 재분배가 현재 진행 중이 아닌 경우 -1입니다.
CREATE_TIME	TIMESTAMP		데이터베이스 파티션 그룹의 작성 시간.
DEFINER[1]	VARCHAR(128)		데이터베이스 파티션 그룹 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.EVENTMONITORS

각 행은 이벤트 모니터를 나타냅니다.

표 108. SYSCAT.EVENTMONITORS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
EVMONNAME	VARCHAR(128)		이벤트 모니터의 이름입니다.
OWNER	VARCHAR(128)		이벤트 모니터 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
TARGET_TYPE	CHAR(1)		이벤트 데이터가 기록되는 목표의 유형. • F = 파일 • P = 파이프 • T = 테이블 • U = 형식화되지 않은 이벤트 테이블
TARGET	VARCHAR (762)		파일 또는 파이프 이벤트 모니터 데이터가 기록되는 목표의 이름입니다. 파일의 경우 절대 경로 이름 또는 상대 경로 이름(데이터베이스에 대한 데이터베이스 경로에 상대적이며 LIST ACTIVE DATABASES 명령을 사용하여 볼 수 있음) 중 하나일 수 있습니다. 파이프의 경우 절대 경로 이름일 수 있습니다.
MAXFILES	INTEGER	Y	이 이벤트 모니터가 이벤트 경로에서 허용하는 이벤트 파일의 최대 수입니다. 최대값이 없는 경우 또는 TARGET_TYPE이 'F'(파일)가 아닌 경우 널(NULL) 값입니다.
MAXFILESIZE	INTEGER	Y	이벤트 모니터가 새 파일을 작성하기 전에 각 이벤트 파일이 도달할 수 있는 최대 크기(4K 페이지 단위)입니다. 최대값이 없는 경우 또는 TARGET_TYPE이 'F'(파일)가 아닌 경우 널(NULL) 값입니다.
BUFFERSIZE	INTEGER	Y	파일 목표를 가진 이벤트 모니터가 사용하는 버퍼의 크기(4K 페이지 단위)이며, 그렇지 않으면 널(NULL) 값입니다.
IO_MODE	CHAR(1)	Y	파일 입출력 모드(I/O). • B = 차단됨 • N = 차단되지 않음 • 널(NULL) 값 = TARGET_TYPE이 'F'(파일) 또는 'T'(테이블)가 아님

표 108. SYSCAT.EVENTMONITORS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
WRITE_MODE	CHAR(1)	Y	모니터가 활성화될 때 이 이벤트 모니터가 기존 이벤트 데이터를 처리하는 방법을 표시합니다. • A = 추가 • R = 바꾸기 • 널(NULL) 값 = TARGET_TYPE이 'F'(파일)가 아님
AUTOSTART	CHAR(1)		데이터베이스가 시작할 때 이 이벤트 모니터가 자동으로 활성화될지 여부를 표시합니다. • N = 아니오 • Y = 예
DBPARTITIONNUM	SMALLINT		이벤트 모니터가 실행되어 이벤트를 로그하는 데이터베이스 파티션 번호.
MONSCOPE	CHAR(1)		모니터링 범위. • G = 전역 • L = 로컬 • T = 테이블 스페이스가 존재하는 각 데이터베이스 파티션 • 공백 = WRITE TO TABLE 이벤트 모니터
EVMON_ACTIVATES	INTEGER		이벤트 모니터가 활성화된 횟수.
NODENUM[1]	SMALLINT		이벤트 모니터가 실행되어 이벤트를 로그하는 데이터베이스 파티션 번호.
DEFINER[2]	VARCHAR(128)		이벤트 모니터 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	나중에 사용하기 위해 예약됨

주:

1. 이전 버전과의 호환성을 위해 NODENUM 컬럼이 포함됩니다. DBPARTITIONNUM을 참조하십시오.
2. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.EVENTS

각 행은 모니터되고 있는 이벤트를 표시합니다. 일반적으로 한 이벤트 모니터가 다중 이벤트를 모니터합니다.

표 109. SYSCAT.EVENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
EVMONNAME	VARCHAR(128)		이 이벤트를 모니터링 중인 이벤트 모니터의 이름.
TYPE	VARCHAR(128)		모니터된 이벤트 유형입니다. 가능한 값은 다음과 같습니다. • ACTIVITIES • CONNECTIONS • DATABASE • DEADLOCKS • DETAILDEADLOCKS • LOCKING • PACKAGECACHESTMT • STATEMENTS • TABLES • TABLESPACES • THRESHOLD_VIOLATIONS • TRANSACTIONS • STATISTICS • UOW
FILTER	CLOB(64K)	Y	이 이벤트에 적용되는 WHERE절의 전체 텍스트.

SYSCAT.EVENTTABLES

각 행은 SQL 테이블에 쓰는 이벤트 모니터의 목표 테이블을 나타냅니다.

표 110. SYSCAT.EVENTTABLES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
EVMONNAME	VARCHAR(128)		이벤트 모니터의 이름입니다.
LOGICAL_GROUP	VARCHAR(128)		논리 데이터 그룹의 이름입니다. 가능한 값은 다음과 같습니다. • ACTIVITYHISTORY • BUFFERPOOL • CONN • CONNHEADER • CONTROL • DATAVAL • DB • DEADLOCK • DLCONN • DLLOCK • LOCKING • PACKAGECACHESTMT • SCSTATS • STMT • STMTHIST • STMTVALS • SUBSECTION • TABLE • TABLESPACE • THRESHOLDVIOLATIONS • UOW • WCSTATS • WLSTATS • XACT
TABSCHEMA	VARCHAR(128)		목표 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		목표 테이블의 규정되지 않은 이름.
PCTDEACTIVATE	SMALLINT		DMS 테이블 스페이스가 얼마나 가득 찼는지를 지정하는 퍼센트 값은 이벤트 모니터가 자동으로 비활성화되기 이전이어야 합니다. SMS 테이블 스페이스의 경우 100으로 설정하십시오.

SYSCAT.FULLHIERARCHIES

각 행은 서브테이블과 수퍼 테이블, 부속 유형과 수퍼 유형 또는 서브뷰와 수퍼 뷰 사이의 관계를 나타냅니다. 인접한 관계를 포함한 모든 계층 관계가 이 뷰에 포함됩니다.

표 111. SYSCAT.FULLHIERARCHIES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
METATYPE	CHAR(1)		관계 유형. • R = 구조화된 유형 사이 • U = 유형이 지정된 테이블 사이 • W = 유형이 지정된 뷰 사이
SUB_SCHEMA	VARCHAR(128)		부속 유형, 서브테이블 또는 서브뷰의 스키마 이름.
SUB_NAME	VARCHAR(128)		부속 유형, 서브테이블 또는 서브뷰의 규정되지 않은 이름.
SUPER_SCHEMA	VARCHAR(128)	Y	수퍼 유형, 수퍼 테이블 또는 수퍼 뷰의 스키마 이름.
SUPER_NAME	VARCHAR(128)	Y	수퍼 유형, 수퍼 테이블 또는 수퍼 뷰의 규정되지 않은 이름.
ROOT_SCHEMA	VARCHAR(128)		계층 구조의 루트에 있는 테이블, 뷰 또는 유형의 스키마 이름.
ROOT_NAME	VARCHAR(128)		계층 구조의 루트에 있는 테이블, 뷰 또는 유형의 규정되지 않은 이름.

SYSCAT.FUNCMAPOPTIONS

각 행은 함수 맵핑 옵션 값을 나타냅니다.

표 112. SYSCAT.FUNCMAPOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
FUNCTION_MAPPING	VARCHAR(128)		함수 맵핑 이름.
OPTION	VARCHAR(128)		함수 맵핑 옵션의 이름.
SETTING	VARCHAR(2048)		함수 맵핑 옵션의 값.

SYSCAT.FUNCMAPPARMOPTIONS

각 행은 함수 맵핑 매개변수 옵션 값을 나타냅니다.

표 113. SYSCAT.FUNCMAPPARMOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
FUNCTION_MAPPING	VARCHAR(128)		함수 맵핑 이름.
ORDINAL	SMALLINT		매개변수 위치.
LOCATION	CHAR(1)		매개변수 위치. • L = 로컬 매개변수 • R = 리모트 매개변수
OPTION	VARCHAR(128)		함수 맵핑 매개변수 옵션의 이름.
SETTING	VARCHAR(2048)		함수 맵핑 매개변수 옵션의 값.

SYSCAT.FUNCMAPPINGS

각 행은 함수 맵핑을 나타냅니다.

표 114. SYSCAT.FUNCMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
FUNCTION_MAPPING	VARCHAR(128)		함수 맵핑의 이름(시스템이 생성했을 수 있음).
FUNCSCHEMA	VARCHAR(128)	Y	함수의 스키마 이름입니다. 널(NULL) 값인 경우 함수는 내장 함수인 것으로 가정됩니다.
FUNCNAME	VARCHAR(1024)	Y	사용자 정의 또는 내장 함수의 규정되지 않은 이름.
FUNCID	INTEGER	Y	함수 ID.
SPECIFICNAME	VARCHAR(128)	Y	루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
OWNER	VARCHAR(128)		맵핑 소유자의 권한 부여 ID. 'SYSIBM'은 이 함수가 내장 함수임을 나타냅니다.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
WRAPNAME	VARCHAR(128)	Y	이 맵핑이 적용되는 랩퍼.
SERVERNAME	VARCHAR(128)	Y	데이터 소스의 이름.
SERVERTYPE	VARCHAR (30)	Y	이 맵핑이 적용되는 데이터 소스 유형.
SERVERVERSION	VARCHAR(18)	Y	이 맵핑이 적용되는 서버 유형의 버전.
CREATE_TIME	TIMESTAMP		맵핑이 작성된 시간.
DEFINER[1]	VARCHAR(128)		맵핑 소유자의 권한 부여 ID. 'SYSIBM'은 이 함수가 내장 함수임을 나타냅니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.HIERARCHIES

각 행은 서브테이블과 인접한 수퍼 테이블, 부속 유형과 인접한 수퍼 유형 또는 서브뷰와 인접한 수퍼 뷰 사이의 관계를 나타냅니다. 이 뷰에는 인접 계층 관계만 포함됩니다.

표 115. SYSCAT.HIERARCHIES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
METATYPE	CHAR(1)		관계 유형. • R = 구조화된 유형 사이 • U = 유형이 지정된 테이블 사이 • W = 유형이 지정된 뷰 사이
SUB_SCHEMA	VARCHAR(128)		부속 유형, 서브테이블 또는 서브뷰의 스키마 이름.
SUB_NAME	VARCHAR(128)		부속 유형, 서브테이블 또는 서브뷰의 규정되지 않은 이름.
SUPER_SCHEMA	VARCHAR(128)		수퍼 유형, 수퍼 테이블 또는 수퍼 뷰의 스키마 이름.
SUPER_NAME	VARCHAR(128)		수퍼 유형, 수퍼 테이블 또는 수퍼 뷰의 규정되지 않은 이름.
ROOT_SCHEMA	VARCHAR(128)		계층 구조의 루트에 있는 테이블, 뷰 또는 유형의 스키마 이름.
ROOT_NAME	VARCHAR(128)		계층 구조의 루트에 있는 테이블, 뷰 또는 유형의 규정되지 않은 이름.

SYSCAT.HISTOGRAMTEMPLATEBINS

각 행은 막대 그래프 템플리트 저장소를 나타냅니다.

표 116. SYSCAT.HISTOGRAMTEMPLATEBINS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TEMPLATENAME	VARCHAR(128)	Y	막대 그래프 템플리트 이름.
TEMPLATEID	INTEGER		막대 그래프 템플리트 ID.
BINID	INTEGER		막대 그래프 템플리트 저장소 ID.
BINUPPERVALUE	BIGINT		막대 그래프 템플리트에 있는 단일 저장소의 상한 값.

SYSCAT.HISTOGRAMTEMPLATES

각 행은 막대 그래프 템플리트를 나타냅니다.

표 117. SYSCAT.HISTOGRAMTEMPLATES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TEMPLATEID	INTEGER		막대 그래프 템플리트 ID.
TEMPLATENAME	VARCHAR(128)		막대 그래프 템플리트 이름.
CREATE_TIME	TIMESTAMP		막대 그래프 템플리트가 작성된 시간.
ALTER_TIME	TIMESTAMP		막대 그래프 템플리트가 마지막으로 변경된 시간.
NUMBINS	INTEGER		바운드되지 않은 최고 값을 갖는 마지막 저장소를 포함하여 막대 그래프 템플리트에 있는 저장소 수.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.HISTOGRAMTEMPLATEUSE

각 행은 막대 그래프 템플리트를 사용할 수 있는 워크로드 관리 오브젝트와 막대 그래프 템플리트 사이의 관계를 나타냅니다.

표 118. SYSCAT.HISTOGRAMTEMPLATEUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TEMPLATENAME	VARCHAR(128)	Y	막대 그래프 템플리트 이름.
TEMPLATEID	INTEGER		막대 그래프 템플리트 ID.
HISTOGRAMTYPE	CHAR(1)		이 템플리트를 기초로 막대 그래프가 수집하는 정보의 유형. • C = 활동 계산된 비용 막대 그래프 • E = 활동 실행 시간 막대 그래프 • I = 활동 도착 간격 시간 막대 그래프 • L = 활동 수명 막대 그래프 • Q = 활동 큐 시간 막대 그래프 • R = 요청 실행 시간 막대 그래프
OBJECTTYPE	CHAR(1)		WLM 오브젝트의 유형. • b = 서비스 클래스 • k = 작업 조치 • w = 워크로드
OBJECTID	INTEGER		WLM 오브젝트 ID.
SERVICECLASSNAME	VARCHAR(128)	Y	서비스 클래스 이름.
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	막대 그래프 템플리트를 사용하는 서비스 서브클래스의 상위 서비스 클래스 이름.
WORKACTIONNAME	VARCHAR(128)	Y	막대 그래프 템플리트를 사용하는 작업 조치 이름.
WORKACTIONSETNAME	VARCHAR(128)	Y	막대 그래프 템플리트를 사용하는 작업 조치를 포함하는 작업 조치 세트 이름.
WORKLOADNAME	VARCHAR(128)	Y	막대 그래프 템플리트를 사용하는 워크로드 이름.

SYSCAT.INDEXAUTH

각 행은 인덱스에 대해 CONTROL 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 119. SYSCAT.INDEXAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
INDSCHEMA	VARCHAR(128)		인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		인덱스의 규정되지 않은 이름.
CONTROLAUTH	CHAR(1)		CONTROL 특권. • N = 보유되지 않음 • Y = 보유됨

SYSCAT.INDEXCOLUSE

각 행은 인덱스에 참여하는 컬럼을 나타냅니다.

표 120. SYSCAT.INDEXCOLUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
INDSCHEMA	VARCHAR(128)		인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		인덱스의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		컬럼 이름
COLSEQ	SMALLINT		인덱스에서 컬럼의 숫자 위치(초기 위치는 1임).
COLORDER	CHAR(1)		이 인덱스 컬럼에서 값의 순서입니다. 가능한 값은 다음과 같습니다. • A = 오름차순 • D = 내림차순 • I = INCLUDE 컬럼(순서 무시)
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 컬럼에 대한 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 컬럼에 대한 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.INDEXDEP

각 행은 일부 다른 오브젝트에 대한 인덱스의 종속성을 나타냅니다. 인덱스는 이름 BNAME 및 유형 BTYPE의 오브젝트에 종속되므로, 오브젝트에 대한 변경사항이 인덱스에 영향을 줍니다.

표 121. SYSCAT.INDEXDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
INDSCHEMA	VARCHAR(128)		인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		인덱스의 규정되지 않은 이름.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • B = 트리거 • F = 루틴 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • K = 패키지 • L = 접속 해제된 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • Q = 시퀀스 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • X = 인덱스 확장 • Z = XSR 오브젝트 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.

표 121. SYSCAT.INDEXDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름입니다. 루틴(BTYPE = 'F')의 경우, 이것은 특정 이름입니다.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 인덱스에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.INDEXES

각 행은 인덱스를 나타냅니다. 유형이 지정된 테이블의 인덱스는 두 행으로 표시되는데, 하나는 유형이 지정된 테이블의 ″논리적 인덱스″에 대한 것이고 다른 하나는 계층 구조 테이블의 ″H 인덱스″에 대한 것입니다.

표 122. SYSCAT.INDEXES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
INDSCHEMA	VARCHAR(128)		인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		인덱스의 규정되지 않은 이름.
OWNER	VARCHAR(128)		인덱스 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
TABSCHEMA	VARCHAR(128)		인덱스가 정의되는 테이블 또는 별칭의 스키마 이름.
TABNAME	VARCHAR(128)		인덱스가 정의되는 테이블 또는 별칭의 규정되지 않은 이름.
COLNAMES	VARCHAR(640)		이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다. 이 정보에 대해서는 SYSCAT.INDEXCOLUSE를 사용하십시오.
UNIQUERULE	CHAR(1)		고유한 규칙. • D = 중복 허가 • U = 고유 • P = 기본 키 구현
MADE_UNIQUE	CHAR(1)		• N = 인덱스가 작성된 그대로 남음 • Y = 이 인덱스는 원래 고유하지 않았지만 고유 또는 기본 키 제한조건을 지원하기 위해 고유 인덱스로 변환되었습니다. 제한조건이 삭제되는 경우 인덱스는 고유하지 않은 것으로 되돌아갑니다.
COLCOUNT	SMALLINT		키 컬럼 수 + Include 컬럼 수(있을 경우)
UNIQUE_COLCOUNT	SMALLINT		고유 키에 필요한 컬럼 수입니다. 항상 <= COLCOUNT이며, Include 컬럼인 경우에만 < COLCOUNT입니다. 인덱스에 고유 키가 없는 경우(즉 중복을 허용함) -1입니다.

표 122. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
INDEXTYPE[5]	CHAR(4)		인덱스 유형. • BLOK = 블록 인덱스 • CLUS = 클러스터링 인덱스(새로 삽입된 행의 실제 배치를 제어함) • DIM = 차원 블록 인덱스 • REG = 일반 인덱스 • XPTH = XML 경로 인덱스 • XRGN = XML 영역 인덱스 • XVIL = XML 컬럼에 대한 인덱스(논리적) • XVIP = XML 컬럼에 대한 인덱스(실제)
ENTRYTYPE	CHAR(1)		• H = 이 행이 계층 구조 테이블의 인덱스를 나타냄 • L = 이 행이 유형이 지정된 테이블의 논리적 인덱스를 나타냄 • 공백 = 이 행은 유형이 지정되지 않은 테이블의 인덱스를 나타냄
PCTFREE	SMALLINT		인덱스의 초기 빌드 중에 예약될 각 인덱스 페이지의 백분율입니다. 이 스페이스는 인덱스가 빌드된 후에 데이터 삽입에 사용 가능합니다.
IID	SMALLINT		인덱스 ID.
NLEAF	BIGINT		리프 페이지 수. 통계가 수집되지 않는 경우 -1입니다.
NLEVELS	SMALLINT		인덱스 레벨 수. 통계가 수집되지 않는 경우 -1입니다.
FIRSTKEYCARD	BIGINT		구별 최초 키 값 수. 통계가 수집되지 않는 경우 -1입니다.
FIRST2KEYCARD	BIGINT		인덱스의 처음 두 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FIRST3KEYCARD	BIGINT		인덱스의 처음 세 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FIRST4KEYCARD	BIGINT		인덱스의 처음 네 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FULLKEYCARD	BIGINT		구별 전체 키 값 수. 통계가 수집되지 않는 경우 -1입니다.
CLUSTERRATIO[3]	SMALLINT		인덱스를 사용한 데이터 클러스터링 등급입니다. 통계가 수집되지 않거나 상세한 인덱스 통계가 수집되는 경우(이 경우 CLUSTERFACTOR가 대신 사용됨) -1입니다.

표 122. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
CLUSTERFACTOR[3]	DOUBLE		클러스터링 등급의 상세 측정입니다. 통계가 수집되지 않는 경우 또는 인덱스가 별칭에 대해 정의되는 경우 -1입니다.
SEQUENTIAL_PAGES	BIGINT		리프 페이지 사이의 큰 갭이 없거나 소수인 인덱스 키 순서에서 디스크에 위치한 리프 페이지 수입니다. 통계가 수집되지 않는 경우 -1입니다.
DENSITY	INTEGER		인덱스로 채워지는 페이지 범위에 있는 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로, 퍼센트(0 - 100 범위의 정수)로 표현됩니다. 통계가 수집되지 않는 경우 -1입니다.
USER_DEFINED	SMALLINT		이 인덱스가 사용자에 의해 정의되고 삭제되지 않은 경우 1이고, 그 외에는 0입니다.
SYSTEM_REQUIRED	SMALLINT		• 다음 조건 중 하나가 만족되는 경우 1입니다. – 이 인덱스가 기본 또는 고유 키 제한조건에 필요하거나, 이 인덱스가 다차원적으로 클러스터된(MDC) 테이블에 대한 차원 블록 인덱스 또는 복합 블록 인덱스입니다. - 이것이 유형이 지정된 테이블의 오브젝트 ID(OID) 컬럼에 대한 인덱스입니다. • 다음 조건이 모두 만족되는 경우 2입니다. – 이 인덱스가 기본 또는 고유 키 제한조건에 필요하거나, 이 인덱스가 MDC 테이블에 대한 차원 블록 인덱스 또는 복합 블록 인덱스입니다. - 이것이 유형이 지정된 테이블의 OID 컬럼에 대한 인덱스입니다. • 그렇지 않으면 0입니다.
CREATE_TIME	TIMESTAMP		인덱스가 작성된 시간.
STATS_TIME	TIMESTAMP	Y	이 인덱스에 대해 기록된 통계가 변경된 마지막 시간입니다. 통계를 사용할 수 없는 경우 널(NULL) 값입니다.
PAGE_FETCH_PAIRS[3]	VARCHAR(520)		문자 양식으로 표시되는 정수 쌍의 목록입니다. 각 쌍은 가상 버퍼의 페이지 수 및 해당 가상 버퍼를 사용하여 이 인덱스를 갖는 테이블을 스캔하기 위해 필요한 페이지 페치 수입니다. 사용 가능한 데이터가 없는 경우 길이가 0인 문자열입니다.
MINPCTUSED	SMALLINT		0이 아닌 정수값은 인덱스가 온라인 조각 모음에 사용 가능함을 표시하고 페이지 병합을 시도할 수 있기 전에 페이지의 사용한 공간의 최소 백분율을 나타냅니다. 값 영(0)은 페이지 병합을 시도하지 않음을 표시합니다.
REVERSE_SCANS	CHAR(1)		• N = 인덱스가 역방향 스캔을 지원하지 않음 • Y = 인덱스가 역방향 스캔을 지원함

표 122. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
INTERNAL_FORMAT	SMALLINT		가능한 값은 다음과 같습니다. • 1 = 인덱스에 역방향 포인터가 없음 • 2 이상 = 인덱스에 역방향 포인터가 있음 • 6 = 인덱스가 복합 블록 인덱스임
COMPRESSION	CHAR(1)		인덱스 압축이 활성화되었는지 여부를 지정합니다. • N = 활성화되지 않음 • Y = 활성화됨
IESCHEMA	VARCHAR(128)	Y	인덱스 확장의 스키마 이름입니다. 일반 인덱스의 경우 널(NULL) 값입니다.
IENAME	VARCHAR(128)	Y	인덱스 확장의 규정되지 않은 이름입니다. 일반 인덱스의 경우 널(NULL) 값입니다.
IEARGUMENTS	CLOB(64K)	Y	인덱스가 작성될 때 지정되는 매개변수의 외부 정보입니다. 일반 인덱스의 경우 널(NULL) 값입니다.
INDEX_OBJECTID	INTEGER		인덱스 오브젝트 ID.
NUMRIDS	BIGINT		인덱스의 행 ID(RID) 또는 블록 ID의 총수이며, 알 수 없는 경우 -1입니다.
NUMRIDS_DELETED	BIGINT		모든 ID가 삭제된 것으로 표시되는 리프 페이지의 ID를 제외하고, 삭제됨으로 표시되는 인덱스의 행 ID(또는 블록 ID)의 총수.
NUM_EMPTY_LEAFS	BIGINT		모든 행 ID(또는 블록 ID)가 삭제됨으로 표시되는 인덱스 리프 페이지의 총수.
AVERAGE_RANDOM_FETCH_PAGES[1,2]	DOUBLE		인덱스를 사용하여 페치할 때 순차 페이지 액세스 사이의 무작위 테이블 페이지의 평균 수이며, 알 수 없는 경우 -1입니다.
AVERAGE_RANDOM_PAGES[2]	DOUBLE		순차 페이지 액세스 사이의 무작위 테이블 페이지의 평균 수이며, 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_GAP[2]	DOUBLE		인덱스 페이지 시퀀스 사이의 갭입니다. 인덱스 리프 페이지의 스캔을 통해 발견되는 각 갭은 인덱스 페이지의 시퀀스 사이에 무작위로 페치되어야 하는 평균 인덱스 페이지 수를 나타냅니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_FETCH_GAP[1,2]	DOUBLE		인덱스를 사용하여 페치할 때 테이블 페이지 시퀀스 사이의 갭입니다. 인덱스 리프 페이지의 스캔을 통해 발견되는 각 갭은 테이블 페이지의 시퀀스 사이에 무작위로 페치되어야 하는 평균 테이블 페이지 수를 나타냅니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_PAGES[2]	DOUBLE		순차적으로 액세스할 수 있는 인덱스 페이지의 평균 수(즉, 프리페처가 시퀀스에 있는 것으로 발견하는 인덱스 페이지 수)입니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_FETCH_PAGES[1,2]	DOUBLE		인덱스를 사용하여 페치할 때 순차적으로 액세스할 수 있는 테이블 페이지의 평균 수(즉, 프리페처가 시퀀스에 있는 것으로 발견하는 테이블 페이지 수)입니다. 알 수 없는 경우 -1입니다.

표 122. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
TBSPACEID	INTEGER		인덱스 테이블 스페이스 ID.
LEVEL2PCTFREE	SMALLINT		인덱스의 초기 빌드 중에 예약될 각 인덱스 레벨 2 페이지의 백분율입니다. 이 스페이스는 인덱스가 빌드된 후에 추후 삽입에 사용 가능합니다.
PAGESPLIT	CHAR(1)		인덱스 페이지 분할 동작. • H = 높음 • L = 낮음 • S = 대칭
AVGPARTITION_ CLUSTERRATIO[3]	SMALLINT		단일 데이터 파티션 내의 데이터 클러스터링 등급입니다. 테이블이 파티션되지 않는 경우, 통계가 수집되지 않는 경우 또는 상세한 통계가 수집되는 경우(이 경우에는 AVGPARTITION_ CLUSTERFACTOR가 대신 사용됨) -1입니다.
AVGPARTITION_ CLUSTERFACTOR[3]	DOUBLE		단일 데이터 파티션에서 클러스터링 등급의 상세한 측정입니다. 테이블이 파티션되지 않는 경우, 통계가 수집되지 않는 경우 또는 인덱스가 별칭에 대해 정의되는 경우 -1입니다.
AVGPARTITION_PAGE_FETCH_ PAIRS[3]	VARCHAR(520)		문자 양식에서 쌍으로 된 정수 목록입니다. 각 쌍은 잠재적 버퍼 풀 크기 및 테이블에서 단일 데이터 파티션에 액세스하기 위해 필요한 대응하는 페이지 페치를 나타냅니다. 사용 가능한 데이터가 없는 경우 또는 테이블이 파티션되지 않는 경우 길이가 0인 문자열입니다.
PCTPAGESSAVED	SMALLINT		인덱스 압축의 결과로 인덱스에 저장되는 페이지의 대략적인 백분율입니다. 통계가 수집되지 않은 경우 -1입니다.
DATAPARTITION_ CLUSTERFACTOR	DOUBLE		데이터 파티션에 관하여 인덱스 키의 "클러스터링"을 측정하는 통계입니다. 0과 1 사이의 숫자이며, 1은 완벽한 클러스터링을 나타내고 0은 클러스터링이 없음을 나타냅니다.
INDCARD	BIGINT		인덱스의 카디널리티(cardinality)입니다. 테이블 행과 인덱스 항목 사이에 일대일 관계가 없는 인덱스의 경우 테이블의 카디널리티(cardinality)와 다를 수 있습니다.
AVGLEAFKEYSIZE	INTEGER		인덱스에 있는 리프 페이지의 키에 대한 평균 인덱스 키 크기.
AVGNLEAFKEYSIZE	INTEGER		인덱스에 있는 비 리프 페이지의 키에 대한 평균 인덱스 키 크기.
OS_PTR_SIZE	INTEGER		인덱스가 작성된 플랫폼 Word 크기. • 32 = 32비트 • 64 = 64비트

표 122. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
COLLECTSTATISTICS	CHAR(1)		인덱스 작성 시에 통계가 수집된 방법을 지정합니다. • D = 상세한 인덱스 통계 수집 • S = 샘플링된 상세한 인덱스 통계 수집 • Y = 기본 인덱스 통계 수집 • 공백 = 인덱스 통계를 수집하지 않음
DEFINER[4]	VARCHAR(128)		인덱스 소유자의 권한 부여 ID
LASTUSED	DATE		나중에 사용하기 위해 예약됨
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. DMS 테이블 스페이스를 사용할 때 이 통계를 연산할 수 없습니다.
2. LOAD...STATISTICS YES 또는 CREATE INDEX...COLLECT STATISTICS 조작 중 또는 데이터베이스 구성 매개변수 *seqdetect*가 꺼졌을 때 프리페치 통계가 수집되지 않습니다.
3. AVGPARTITION_CLUSTERRATIO, AVGPARTITION_CLUSTERFACTOR 및 AVGPARTITION_PAGE_FETCH_PAIRS가 단일 데이터 파티션에서 클러스터링 등급을 측정합니다(로컬 클러스터링). CLUSTERRATIO, CLUSTERFACTOR 및 PAGE_FETCH_PAIRS는 전체 테이블에서 클러스터링 등급을 측정합니다(전역 클러스터링). 테이블 파티셔닝 키가 인덱스 키의 접두부가 아닌 경우 또는 테이블 파티셔닝 키 및 인덱스 키가 서로 논리적으로 독립일 때 전역 클러스터링 및 로컬 클러스터링 값이 크게 달라질 수 있습니다.
4. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.
5. XPTH, XRGN 및 XVIP 인덱스가 인덱스 메타데이터를 리턴하는 API에 의해 인식되지 않습니다.

SYSCAT.INDEXEXPLOITRULES

각 행은 인덱스 개발 규칙을 나타냅니다.

표 123. SYSCAT.INDEXEXPLOITRULES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
FUNCID	INTEGER		함수 ID.
SPECID	SMALLINT		술어 스펙 번호.
IESCHEMA	VARCHAR(128)		인덱스 확장의 스키마 이름.
IENAME	VARCHAR(128)		인덱스 확장의 규정되지 않은 이름.
RULEID	SMALLINT		개발 규칙 ID.
SEARCHMETHODID	SMALLINT		특정 인덱스 확장에 있는 검색 메소드의 ID.
SEARCHKEY	VARCHAR (640)		인덱스 개발에 사용되는 키.
SEARCHARGUMENT	VARCHAR(2700)		인덱스 개발에 사용되는 검색 인수.
EXACT	CHAR(1)		• N = 인덱스 찾아보기가 술어 평가 측면에서 정확하지 않음 • Y = 인덱스 찾아보기가 술어 평가 측면에서 정확함

SYSCAT.INDEXEXTENSIONDEP

각 행은 일부 다른 오브젝트에 대한 인덱스 확장의 종속성을 나타냅니다. 인덱스 확장은 이름 BNAME의 유형 BTYPE의 오브젝트에 종속되므로, 오브젝트에 대한 변경사항이 인덱스 확장에 영향을 줍니다.

표 124. SYSCAT.INDEXEXTENSIONDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
IESCHEMA	VARCHAR(128)		인덱스 확장의 스키마 이름.
IENAME	VARCHAR(128)		인덱스 확장의 규정되지 않은 이름.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • B = 트리거 • F = 루틴 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • K = 패키지 • L = 접속 해제된 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • Q = 시퀀스 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • X = 인덱스 확장 • Z = XSR 오브젝트 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.

표 124. SYSCAT.INDEXEXTENSIONDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름입니다. 루틴(BTYPE = 'F')의 경우, 이것은 특정 이름입니다.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 인덱스 확장에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.INDEXEXTENSIONMETHODS

각 행은 검색 메소드를 나타냅니다. 인덱스 확장은 둘 이상의 검색 메소드를 포함할 수 있습니다.

표 125. SYSCAT.INDEXEXTENSIONMETHODS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
METHODNAME	VARCHAR(128)		검색 메소드 이름.
METHODID	SMALLINT		인덱스 확장의 메소드 수.
IESCHEMA	VARCHAR(128)		이 메소드가 정의되는 인덱스 확장의 스키마 이름.
IENAME	VARCHAR(128)		이 메소드가 정의되는 인덱스 확장의 규정되지 않은 이름.
RANGEFUNCSCHEMA	VARCHAR(128)		range-through 함수의 스키마 이름.
RANGEFUNCNAME	VARCHAR(128)		range-through 함수의 규정되지 않은 이름.
RANGESPECIFICNAME	VARCHAR(128)		range-through 함수의 함수 특정 이름.
FILTERFUNCSCHEMA	VARCHAR(128)	Y	필터 함수의 스키마 이름.
FILTERFUNCNAME	VARCHAR(128)	Y	필터 함수의 규정되지 않은 이름.
FILTERSPECIFICNAME	VARCHAR(128)	Y	필터 함수의 함수 특정 이름.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.INDEXEXTENSIONPARMS

각 행은 인덱스 확장 인스턴스 매개변수 또는 소스 키 컬럼을 나타냅니다.

표 126. SYSCAT.INDEXEXTENSIONPARMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
IESCHEMA	VARCHAR(128)		인덱스 확장의 스키마 이름.
IENAME	VARCHAR(128)		인덱스 확장의 규정되지 않은 이름.
ORDINAL	SMALLINT		매개변수 또는 키 컬럼의 시퀀스 번호.
PARMNAME	VARCHAR(128)		매개변수 또는 키 컬럼의 이름.
TYPESCHEMA	VARCHAR(128)		매개변수 또는 키 컬럼의 데이터 유형의 스키마 이름.
TYPENAME	VARCHAR(128)		매개변수 또는 키 컬럼의 데이터 유형의 규정되지 않은 이름.
LENGTH	INTEGER		매개변수 또는 키 컬럼의 데이터 유형 길이.
SCALE	SMALLINT		매개변수 또는 키 컬럼의 데이터 유형 스케일이며, 적용할 수 없는 경우 0입니다.
PARMTYPE	CHAR(1)		• K = 소스 키 컬럼 • P = 인덱스 확장 인스턴스 매개변수
CODEPAGE	SMALLINT		인덱스 확장 인스턴스 매개변수의 코드 페이지이며, 문자열 유형이 아닌 경우 0입니다.
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 매개변수 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 매개변수에 대한 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.INDEXEXTENSIONS

각 행은 인덱스 확장을 나타냅니다.

표 127. SYSCAT.INDEXEXTENSIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
IESCHEMA	VARCHAR(128)		인덱스 확장의 스키마 이름.
IENAME	VARCHAR(128)		인덱스 확장의 규정되지 않은 이름.
OWNER	VARCHAR(128)		인덱스 확장 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
CREATE_TIME	TIMESTAMP		인덱스 확장이 정의된 시간.
KEYGENFUNCSCHEMA	VARCHAR(128)		키 생성 함수의 스키마 이름.
KEYGENFUNCNAME	VARCHAR(128)		키 생성 함수의 규정되지 않은 이름.
KEYGENSPECIFICNAME	VARCHAR(128)		키 생성 함수 인스턴스의 이름(시스템이 생성했을 수 있음).
TEXT	CLOB(2M)		CREATE INDEX EXTENSION문의 전체 텍스트.
DEFINER[1]	VARCHAR(128)		인덱스 확장 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.INDEXOPTIONS

각 행은 인덱스 특정 옵션 값을 나타냅니다.

표 128. SYSCAT.INDEXOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
INDSCHEMA	VARCHAR(128)		인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		인덱스의 규정되지 않은 이름.
OPTION	VARCHAR(128)		인덱스 옵션의 이름.
SETTING	VARCHAR(2048)		인덱스 옵션 값.

SYSCAT.INDEXPARTITIONS

각 행은 하나의 데이터 파티션에 위치하는 파티션된 인덱스 조각을 나타냅니다. 메모:

- 테이블이 여러 개의 데이터베이스 파티션에서 작성된 경우 인덱스 파티션 통계는 한 개의 데이터베이스 파티션을 나타냅니다.

표 129. SYSCAT.INDEXPARTITIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
INDSCHEMA	VARCHAR(128)		인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		인덱스의 규정되지 않은 이름.
TABSCHEMA	VARCHAR(128)		인덱스가 정의되는 테이블 또는 별칭의 스키마 이름.
TABNAME	VARCHAR(128)		인덱스가 정의되는 테이블 또는 별칭의 규정되지 않은 이름.
IID	SMALLINT		인덱스 ID.
INDPARTITIONTBSPACEID	INTEGER		인덱스 파티션 테이블 스페이스 ID.
INDPARTITIONOBJECTID	INTEGER		인덱스 파티션 오브젝트 ID.
DATAPARTITIONID	INTEGER		SYSCAT.DATAPARTITIONS 뷰에 있는 DATAPARTITIONID에 대응합니다.
INDCARD	BIGINT		인덱스 파티션의 카디낼리티(cardinality)입니다. 데이터 파티션 행과 인덱스 항목 사이에 일대일 관계를 갖지 않는 파티션된 인덱스에 대한 대응하는 데이터 파티션의 카디낼리티(cardinality)와 다를 수 있습니다.
NLEAF	BIGINT		인덱스 파티션에 있는 리프 페이지 수이며, 통계가 수집되지 않는 경우 -1입니다.
NUM_EMPTY_LEAFS	BIGINT		모든 행 ID(RID) 또는 블록 ID(BID)가 삭제됨으로 표시되는 인덱스 파티션에 있는 인덱스 리프 페이스의 총수.
NUMRIDS	BIGINT		인덱스 파티션에 있는 행 ID(RID) 또는 블록 ID(BID)의 총수이며, 알 수 없는 경우 -1입니다.
NUMRIDS_DELETED	BIGINT		모든 ID가 삭제된 것으로 표시되는 리프 페이지의 ID를 제외하고, 삭제됨으로 표시되는 인덱스 파티션의 행 ID(RID) 또는 블록 ID(BID)의 총수.
FULLKEYCARD	BIGINT		인덱스 파티션에 있는 구별 전체 키 값의 수이며, 통계가 수집되지 않는 경우 -1입니다.
NLEVELS	SMALLINT		인덱스 파티션에 있는 인덱스 레벨 수이며, 통계가 수집되지 않는 경우 -1입니다.
CLUSTERRATIO	SMALLINT		인덱스 파티션과의 데이터 클러스터링 등급입니다. 다음 상황에서는 -1입니다. • 통계는 수집되지 않습니다. • 상세한 인덱스 통계가 수집됩니다. 이 상황에서는 CLUSTERFACTOR가 대신 사용됩니다.
CLUSTERFACTOR	DOUBLE		클러스터링 등급의 상세 측정입니다. 통계가 수집되지 않는 경우 -1입니다.

표 129. SYSCAT.INDEXPARTITIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
FIRSTKEYCARD	BIGINT		구별 최초 키 값 수. 통계가 수집되지 않는 경우 -1입니다.
FIRST2KEYCARD	BIGINT		인덱스 키의 처음 두 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FIRST3KEYCARD	BIGINT		인덱스 키의 처음 세 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FIRST4KEYCARD	BIGINT		인덱스 키의 처음 네 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
AVGLEAFKEYSIZE	INTEGER		인덱스 파티션에 있는 리프 페이지의 키에 대한 평균 인덱스 키 *크기*이며, 통계가 수집되지 않는 경우 -1입니다.
AVGNLEAFKEYSIZE	INTEGER		인덱스 파티션에 있는 리프가 아닌 페이지의 키에 대한 평균 인덱스 키 *크기*이며, 통계가 수집되지 않는 경우 -1입니다.
PCTFREE	SMALLINT		인덱스 파티션의 초기 빌드 중에 예약될 각 인덱스 페이지의 백분율입니다. 이 스페이스는 인덱스 파티션이 빌드된 후에 데이터 삽입에 사용 가능합니다.
PAGE_FETCH_PAIRS	VARCHAR(520)		문자 양식으로 표시되는 정수 쌍의 목록입니다. 각 쌍은 가상 버퍼의 페이지 수 및 해당 가상 버퍼를 사용하여 이 인덱스를 갖는 데이터 파티션을 스캔하기 위해 필요한 페이지 페치 수를 나타냅니다. 사용 가능한 데이터가 없는 경우 길이가 0인 문자열입니다.
SEQUENTIAL_PAGES	BIGINT		리프 페이지 사이의 큰 갭이 없거나 소수인 인덱스 키 순서에서 디스크에 위치한 리프 페이지 수입니다. 통계가 수집되지 않는 경우 -1입니다.
DENSITY	INTEGER		인덱스 파티션으로 채워지는 페이지 범위에 있는 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로, 퍼센트(0 - 100 범위의 정수)로 표현됩니다. 통계가 수집되지 않는 경우 -1입니다.
AVERAGE_SEQUENCE_GAP	DOUBLE		인덱스 파티션 내에서 인덱스 페이지 시퀀스 사이의 갭입니다. 인덱스 리프 페이지의 스캔을 통해 발견되는 각 갭은 인덱스 페이지의 시퀀스 사이에 무작위로 페치되어야 하는 평균 인덱스 페이지 수를 나타냅니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_ FETCH_GAP	DOUBLE		인덱스 파티션을 사용하여 페치할 때 테이블 페이지 시퀀스 사이의 갭입니다. 인덱스 리프 페이지의 스캔을 통해 발견되는 각 갭은 데이터 파티션 페이지의 시퀀스 사이에 무작위로 페치되어야 하는 평균 데이터 파티션 페이지 수를 나타냅니다. 알 수 없는 경우 -1입니다.

표 129. SYSCAT.INDEXPARTITIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
AVERAGE_SEQUENCE_ PAGES	DOUBLE		순차적으로 액세스할 수 있는 인덱스 페이지의 평균 수(즉, 프리페처가 시퀀스에 있는 것으로 발견하는 인덱스 페이지 수)입니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_ FETCH_PAGES	DOUBLE		인덱스를 사용하여 페치할 때 순차적으로 액세스할 수 있는 데이터 파티션 페이지의 평균 수(즉, 프리페처가 시퀀스에 있는 것으로 발견하는 데이터 파티션 페이지 수)입니다. 알 수 없는 경우 -1입니다.
AVERAGE_RANDOM_PAGES	DOUBLE		순차 페이지 액세스 사이의 무작위 데이터 파티션 페이지의 평균 수이며, 알 수 없는 경우 -1입니다.
AVERAGE_RANDOM_FETCH_ PAGES	DOUBLE		인덱스 파티션을 사용하여 페치할 때 순차 페이지 액세스 사이의 무작위 데이터 파티션 페이지의 평균 수이며, 알 수 없는 경우 -1입니다.
STATS_TIME	TIMESTAMP	Y	이 인덱스 파티션에 대해 기록된 통계가 변경된 마지막 시간입니다. 통계를 사용할 수 없는 경우 널(NULL) 값입니다.
COMPRESSION	CHAR(1)		인덱스 압축이 활성화되었는지 여부를 지정합니다. • N = 활성화되지 않음 • Y = 활성화됨
PCTPAGESSAVED	SMALLINT		인덱스 압축의 결과로 인덱스에 저장되는 페이지의 대략적인 백분율입니다. 통계가 수집되지 않은 경우 -1입니다.

SYSCAT.INDEXXMLPATTERNS

각 행은 XML 컬럼에 대한 인덱스의 패턴 절을 나타냅니다.

표 130. SYSCAT.INDEXXMLPATTERNS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
INDSCHEMA	VARCHAR(128)		논리적 인덱스의 스키마 이름.
INDNAME	VARCHAR(128)		논리적 인덱스의 규정되지 않은 이름.
PINDNAME	VARCHAR(128)		실제 인덱스의 규정되지 않은 이름.
PINDID	SMALLINT		실제 인덱스 ID.
TYPEMODEL	CHAR(1)		• Q = SQL DATA TYPE(유효하지 않은 값 무시) • R = SQL DATA TYPE(유효하지 않은 값 거부)
DATATYPE	VARCHAR(128)		데이터 유형 이름.
HASHED	CHAR(1)		값이 해시되는지의 여부를 나타냅니다. • N = 해시되지 않음 • Y = 해시됨
LENGTH	SMALLINT		VARCHAR(*n*) 길이. 그 외에는 0입니다.
PATTERNID	SMALLINT		패턴 ID.
PATTERN	CLOB(2M)	Y	패턴 정의.

주:

1. XML 컬럼에 대한 인덱스가 작성될 때, XML 패턴 정보를 이용하는 논리적 인덱스가 작성되어 논리적 인덱스를 지원하기 위한 DB2 생성 키 컬럼을 갖는 실제 B-트리 인덱스가 작성됩니다. 실제 인덱스는 CREATE INDEX문의 xmltype-clause에서 지정되는 데이터 유형을 지원하기 위해 작성됩니다.

SYSCAT.INVALIDOBJECTS

각 행은 유효하지 않은 오브젝트를 나타냅니다.

표 131. SYSCAT.INVALIDOBJECTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTSCHEMA	VARCHAR(128)		작성 또는 유효성이 다시 확인될 오브젝트의 스키마 이름.
OBJECTMODULENAME	VARCHAR(128)	Y	작성 또는 유효성 다시 확인 중인 오브젝트가 속하는 모듈의 규정되지 않은 이름. 오브젝트가 모듈에 속하지 않는 경우, 널(NULL) 값입니다.
OBJECTNAME	VARCHAR(128)		작성 또는 유효성이 다시 확인될 오브젝트의 규정되지 않은 이름입니다. 루틴(OBJECTTYPE = 'F')의 경우, 이것은 특정 이름입니다.
ROUTINENAME	VARCHAR(128)	Y	루틴의 규정되지 않은 이름.
OBJECTTYPE	CHAR(1)		작성 또는 유효성이 다시 확인될 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • B = 트리거 • F = 루틴 • R = 사용자 정의 데이터 유형 • V = 뷰 • v = 전역 변수
SQLCODE	INTEGER	Y	오류 또는 유효성 다시 확인과 함께 CREATE에서 리턴되는 SQLCODE입니다. 오브젝트가 유효성이 다시 확인된 적이 없는 경우 널(NULL) 값입니다.
SQLSTATE	CHAR(5)	Y	오류 또는 유효성 다시 확인과 함께 CREATE에서 리턴되는 SQLSTATE입니다. 오브젝트가 유효성이 다시 확인된 적이 없는 경우 널(NULL) 값입니다.
ERRORMESSAGE	VARCHAR(70)	Y	SQLCODE와 연관된 메시지의 간단한 텍스트. 오브젝트가 유효성이 다시 확인된 적이 없는 경우 널(NULL) 값입니다.
LINENUMBER	INTEGER	Y	컴파일된 오브젝트에서 오류가 발생하는 행 번호입니다. 오브젝트가 컴파일된 오브젝트가 아닌 경우 널(NULL) 값입니다.
INVALIDATE_TIME	TIMESTAMP		오브젝트가 마지막으로 무효화된 시간.
LAST_REGEN_TIME	TIMESTAMP	Y	오브젝트가 마지막으로 유효성이 다시 확인된 시간입니다. 오브젝트가 유효성이 다시 확인된 적이 없는 경우 널(NULL) 값입니다.

SYSCAT.KEYCOLUSE

각 행은 고유, 기본 키 또는 외부 키 제한조건에 의해 정의되는 키에 참여하는 컬럼을 나타냅니다.

표 132. SYSCAT.KEYCOLUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONSTNAME	VARCHAR(128)		제한조건 이름.
TABSCHEMA	VARCHAR(128)		컬럼을 포함하는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		컬럼을 포함하는 테이블의 규정되지 않은 이름.
COLNAME	VARCHAR(128)		컬럼 이름
COLSEQ	SMALLINT		제한조건에 대한 키에서 컬럼의 숫자 위치(초기 위치가 1). 제한조건에서 기존 인덱스를 사용하는 경우 이 값은 인덱스에서 컬럼에 대한 숫자 위치입니다.

SYSCAT.MODULEAUTH

각 행은 모듈에 대한 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 133. SYSCAT.MODULEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권의 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
MODULEID	INTEGER		이 특권이 적용되는 모듈 ID.
MODULESCHEMA	VARCHAR(128)		이 특권이 적용되는 모듈의 스키마 이름.
MODULENAME	VARCHAR(128)		이 특권이 적용되는 모듈의 규정되지 않은 이름.
EXECUTEAUTH	CHAR(1)		식별된 모듈에 있는 오브젝트를 실행할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.MODULEOBJECTS

각 행은 모듈에 속하는 함수, 프로시저, 전역 변수, 조건 또는 사용자 정의 유형을 나타냅니다.

표 134. SYSCAT.MODULEOBJECTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTSCHEMA	VARCHAR(128)	N	모듈의 스키마 이름.
OBJECTMODULENAME	VARCHAR(128)	N	오브젝트가 속하는 모듈의 규정되지 않은 이름
OBJECTNAME	VARCHAR(128)	N	오브젝트의 규정되지 않은 이름.
OBJECTTYPE	VARCHAR(9)	N	• CONDITION = 오브젝트가 조건임 • FUNCTION = 오브젝트가 함수임 • PROCEDURE = 오브젝트가 프로시저임 • TYPE = 오브젝트가 데이터 유형임 • VARIABLE = 오브젝트가 변수임
PUBLISHED	CHAR(1)	N	오브젝트를 해당 모듈 밖에서 참조할 수 있는지 여부를 표시합니다. • N = 오브젝트가 발행되지 않음 • Y = 오브젝트가 발행됨
SPECIFICNAME	VARCHAR(128)	N	OBJECTTYPE이 'FUNCTION', 'METHOD' 또는 'PROCEDURE'인 경우, 루틴 특화명입니다. 그렇지 않으면, 널(NULL) 값입니다.
USERDEFINED	CHAR(1)	N	오브젝트가 시스템에 의해 생성되는지 또는 사용자가 정의하는지 여부를 표시합니다. • N = 오브젝트가 시스템 생성됨 • Y = 오브젝트가 사용자에 의해 정의됨

SYSCAT.MODULES

각 행은 모듈을 나타냅니다.

표 135. SYSCAT.MODULES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
MODULESCHEMA	VARCHAR(128)		모듈의 스키마 이름.
MODULENAME	VARCHAR(128)		모듈의 규정되지 않은 이름.
MODULEID	INTEGER		모듈 ID.
DIALECT	VARCHAR(10)		SQL 모듈의 소스 표현 형식입니다. 가능한 값은 다음과 같습니다. • DB2 SQL PL • PL/SQL • 공백 = 별명에 적용할 수 없음
OWNER	VARCHAR(128)		모듈 소유자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
MODULETYPE	CHAR(1)		모듈 유형. • A = 별명 • M = 모듈 • P = PL/SQL 패키지
BASE_MODULESCHEMA	VARCHAR(128)	Y	MODULETYPE이 'A'이면 모듈의 스키마 이름 또는 이 별명이 참조하는 별명이 들어있고, 그 외에는 널(NULL) 값입니다.
BASE_MODULENAME	VARCHAR(128)	Y	MODULETYPE이 'A'이면 모듈의 규정되지 않은 이름 또는 이 별명이 참조하는 별명이 들어있고, 그 외에는 널(NULL) 값입니다.
CREATE_TIME	TIMESTAMP		모듈이 작성된 시간.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.NAMEMAPPINGS

각 행은 ″논리적″ 오브젝트(상속된 컬럼을 포함하여 유형이 지정된 테이블 또는 뷰 및 해당 컬럼과 인덱스)와 논리적 오브젝트를 구현하는 대응하는 ″구현″ 오브젝트(계층 구조 테이블 또는 계층 구조 뷰와 그의 컬럼 및 인덱스) 사이의 맵핑을 나타냅니다.

표 136. SYSCAT.NAMEMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPE	CHAR(1)		• C = 컬럼 • I = 인덱스 • U = 유형이 지정된 테이블
LOGICAL_SCHEMA	VARCHAR(128)		논리적 오브젝트의 스키마 이름.
LOGICAL_NAME	VARCHAR(128)		논리적 오브젝트의 규정되지 않은 이름.
LOGICAL_COLNAME	VARCHAR(128)	Y	TYPE = 'C'인 경우 논리적 컬럼의 이름이며, 그 외에는 널(NULL) 값입니다.
IMPL_SCHEMA	VARCHAR(128)		논리적 오브젝트를 구현하는 구현 오브젝트의 스키마 이름.
IMPL_NAME	VARCHAR(128)		논리적 오브젝트를 구현하는 구현 오브젝트의 규정되지 않은 이름.
IMPL_COLNAME	VARCHAR(128)	Y	TYPE = 'C'인 경우 구현 컬럼의 이름이며, 그 외에는 널(NULL) 값입니다.

SYSCAT.NICKNAMES

각 행은 별칭을 나타냅니다.

표 137. SYSCAT.NICKNAMES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		별칭의 스키마 이름.
TABNAME	VARCHAR(128)		별칭의 규정되지 않은 이름.
OWNER	VARCHAR(128)		테이블, 뷰, 별명 또는 별칭 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
STATUS	CHAR(1)		오브젝트 상태. • C = 무결성 설정 보류 • N = 보통 • X = 작동 불능
CREATE_TIME	TIMESTAMP		오브젝트가 작성된 시간.
STATS_TIME	TIMESTAMP	Y	이 오브젝트에 대해 기록된 통계가 마지막으로 변경된 시간입니다. 통계가 수집되지 않은 경우 널(NULL) 값입니다.
COLCOUNT	SMALLINT		상속된 컬럼(있는 경우)을 포함한 컬럼 수.
TABLEID	SMALLINT		내부 논리적 오브젝트 ID.
TBSPACEID	SMALLINT		이 오브젝트에 대한 기본 테이블 스페이스의 내부 논리적 ID.
CARD	BIGINT		테이블에 있는 총 행 수이며, 통계가 수집되지 않는 경우 -1입니다.
NPAGES	BIGINT		별칭의 행이 존재하는 페이지의 총수입니다. 통계가 수집되지 않는 경우 -1입니다.
FPAGES	BIGINT		페이지 총수. 통계가 수집되지 않는 경우 -1입니다.
OVERFLOW	BIGINT		오버플로우 레코드 총수. 통계가 수집되지 않는 경우 -1입니다.
PARENTS	SMALLINT	Y	이 오브젝트에 대한 상위 테이블 수입니다. 즉, 이 오브젝트가 종속되는 참조 제한조건의 수입니다.
CHILDREN	SMALLINT	Y	이 오브젝트에 대한 종속 테이블 수입니다. 즉, 이 오브젝트가 상위인 참조 제한조건의 수입니다.
SELFREFS	SMALLINT	Y	이 오브젝트에 대해 자체 참조하는 참조 제한조건의 수입니다. 즉, 이 오브젝트가 상위이면서 종속인 참조 제한조건의 수입니다.
KEYCOLUMNS	SMALLINT	Y	기본 키의 컬럼 수.
KEYINDEXID	SMALLINT	Y	기본 키 인덱스의 인덱스 ID입니다. 기본 키가 없는 경우 0 또는 널(NULL) 값입니다.
KEYUNIQUE	SMALLINT		이 오브젝트에 정의되는 고유 키 제한조건(기본 키 제한조건 제외)의 수.
CHECKCOUNT	SMALLINT		이 오브젝트에 정의된 점검 제한조건의 수.

표 137. SYSCAT.NICKNAMES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
DATACAPTURE	CHAR(1)		• L = 별칭이 LONG VARCHAR 및 LONG VARGRAPHIC 컬럼의 복제를 포함한 데이터 복제에 참여함 • N = 별칭이 데이터 복제에 참여하지 않음 • Y = 별칭이 데이터 복제에 참여함
CONST_CHECKED	CHAR(32)		• 바이트 1은 외부 키 제한조건을 나타냅니다. • 바이트 2는 점검 제한조건을 나타냅니다. • 바이트 5는 구체화된 쿼리 테이블을 나타냅니다. • 바이트 6은 생성된 컬럼을 나타냅니다. • 바이트 7은 스테이징 테이블을 나타냅니다. • 바이트 8은 데이터 파티셔닝 제한조건을 나타냅니다. • 나머지 바이트는 나중에 사용하기 위해 예약되어 있습니다. 가능한 값은 다음과 같습니다. • F = 바이트 5에서 구체화된 쿼리 테이블은 증분하여 새로 고칠 수 없습니다. 바이트 7에서 스테이징 테이블의 컨텐츠는 불완전하며 연관된 구체화된 쿼리 테이블의 증분 새로 고침에 사용할 수 없습니다. • N = 점검되지 않음 • U = 사용자가 점검함 • W = 테이블이 무결성 설정 보류 상태이면 'U' 상태 • Y = 시스템 점검
PARTITION_MODE	CHAR(1)		나중에 사용하기 위해 예약됨
STATISTICS_PROFILE	CLOB(10M)	Y	오브젝트에 대한 통계 프로파일을 등록하는 데 사용되는 RUNSTATS 명령.
ACCESS_MODE	CHAR(1)		오브젝트의 액세스 제한 상태입니다. 이러한 상태는 무결성 설정 보류 상태에 있는 오브젝트나 SET INTEGRITY문으로 처리된 오브젝트에만 적용됩니다. 가능한 값은 다음과 같습니다. • D = 데이터 이동 없음 • F = 전체 액세스 • N = 권한 없음 • R = 읽기 전용 액세스
CODEPAGE	SMALLINT		오브젝트의 코드 페이지입니다. 모든 문자 컬럼, 트리거, 점검 제한조건 및 표현식 생성 컬럼에 사용되는 디폴트 코드 페이지입니다.

표 137. SYSCAT.NICKNAMES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
REMOTE_TABLE	VARCHAR(128)	Y	별칭을 작성한 특정 데이터 소스 오브젝트(테이블 또는 뷰)의 규정되지 않은 이름.
REMOTE_SCHEMA	VARCHAR(128)	Y	별칭을 작성한 특정 데이터 소스 오브젝트(테이블 또는 뷰)의 스키마 이름.
SERVERNAME	VARCHAR(128)	Y	별칭을 작성한 테이블이나 뷰를 포함하는 데이터 소스의 이름.
REMOTE_TYPE	CHAR(1)	Y	데이터 소스에 있는 오브젝트의 유형. • A = 별명 • N = 별칭 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • V = 뷰(유형이 지정되지 않음)
CACHINGALLOWED	VARCHAR(1)		• N = 캐싱이 허용되지 않음 • Y = 캐싱이 허용됨
DEFINER[1]	VARCHAR(128)		테이블, 뷰, 별명 또는 별칭 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.PACKAGEAUTH

각 행은 패키지에서 하나 이상의 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 138. SYSCAT.PACKAGEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
PKGSCHEMA	VARCHAR(128)		패키지의 스키마 이름.
PKGNAME	VARCHAR(128)		패키지의 규정되지 않은 이름.
CONTROLAUTH	CHAR(1)		CONTROL 특권. • N = 보유되지 않음 • Y = 보유됨
BINDAUTH	CHAR(1)		패키지를 바인드할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
EXECUTEAUTH	CHAR(1)		패키지를 실행할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.PACKAGEDEP

각 행은 일부 다른 오브젝트에 대한 패키지의 종속성을 나타냅니다. 패키지는 이름 BNAME의 유형 BTYPE의 오브젝트에 종속되므로, 오브젝트에 대한 변경사항이 패키지에 영향을 줍니다.

표 139. SYSCAT.PACKAGEDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
PKGSCHEMA	VARCHAR(128)		패키지의 스키마 이름.
PKGNAME	VARCHAR(128)		패키지의 규정되지 않은 이름.
BINDER	VARCHAR(128)		패키지의 바인더.
BINDERTYPE	CHAR(1)		• U = 바인더가 개별 사용자임
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • B = 트리거 • D = 서버 정의 • F = 루틴 • I = 인덱스 • M = 기능 맵핑 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • P = 페이지 크기 • Q = 시퀀스 오브젝트 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • Z = XSR 오브젝트 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수
BSCHEMA	VARCHAR(128)		패키지가 종속되는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.
BNAME	VARCHAR(128)		패키지가 종속되는 오브젝트의 규정되지 않은 이름.

표 139. SYSCAT.PACKAGEDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE이 'O', 'S', 'T', 'U', 'V' 또는 'W'인 경우, 이 패키지에서 필요한 특권(SELECT, INSERT, UPDATE 또는 DELETE)을 인코드합니다. 비트 벡터는 SQL.H에서 정의됩니다.
VARAUTH	SMALLINT	Y	BTYPE이 'v'인 경우 이 패키지에서 필요한 특권(READ 또는 WRITE)을 인코드합니다. 비트 벡터는 SQL.H에서 정의됩니다.
UNIQUE_ID	CHAR(8) FOR BIT DATA		동일한 이름을 갖는 다중 패키지가 존재할 때 특정 패키지에 대한 ID.
PKGVERSION	VARCHAR(64)	Y	패키지의 버전 ID.

주:

1. 종속성을 갖는 기능 인스턴스가 삭제되는 경우 패키지는 "작동 불능" 상태에 들어가며 명시적으로 리바인드되어야 합니다. 종속성을 갖는 다른 오브젝트가 삭제되는 경우 패키지는 "유효하지 않음" 상태에 들어가고 시스템이 처음으로 패키지를 참조할 때 자동으로 패키지를 리바인드하려고 시도합니다.

SYSCAT.PACKAGES

각 행은 응용프로그램을 바인드하여 작성된 패키지를 나타냅니다.

표 140. SYSCAT.PACKAGES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
PKGSCHEMA	VARCHAR(128)		패키지의 스키마 이름.
PKGNAME	VARCHAR(128)		패키지의 규정되지 않은 이름.
BOUNDBY	VARCHAR(128)		패키지 바인더 및 소유자의 권한 부여 ID
BOUNDBYTYPE	CHAR(1)		• U = 바인더 및 소유자가 개별 사용자임
OWNER	VARCHAR(128)		패키지 바인더 및 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• U = 바인더 및 소유자가 개별 사용자임
DEFAULT_SCHEMA	VARCHAR(128)		정적 SQL문에서 규정되지 않은 이름에 대해 사용되는 디폴트 스키마 이름.
VALID[1]	CHAR(1)		• N = 리바인드가 필요함 • V = 런타임 시 유효성 확인 • X = 종속되는 일부 기능 인스턴스가 삭제되었기 때문에 패키지가 작동 불능임. 명시적 리바인드가 필요함 • Y = 유효함
UNIQUE_ID	CHAR(8) FOR BIT DATA		동일한 이름을 갖는 다중 패키지가 존재할 때 특정 패키지에 대한 ID.
TOTAL_SECT	SMALLINT		패키지의 섹션 수.
FORMAT	CHAR(1)		패키지와 연관된 날짜 및 시간 형식. • 0 = 클라이언트의 지역 코드와 연관된 형식 • 1 = USA • 2 = EUR • 3 = ISO • 4 = JIS • 5 = LOCAL
ISOLATION	CHAR(2)	Y	분리 레벨. • CS = 커서 안정성 • RR = 반복 읽기 • RS = 읽기 안정성 • UR = 언커미트 읽기
CONCURRENTACCESSRESOLUTION	CHAR(1)	Y	CONCURRENTACCESSRESOLUTION 바인드 옵션의 값: • U = USE CURRENTLY COMMITTED • W = WAIT FOR OUTCOME • 공백 = 지정되지 않음
BLOCKING	CHAR(1)	Y	커서 블로킹 옵션. • B = 전체 커서 블록화 • N = 블로킹 없음 • U = 명확한 커서 블록화
INSERT_BUF	CHAR(1)		INSERT 바인드 옵션의 설정(파티션된 데이터베이스 시스템에 적용됨). • N = 삽입이 버퍼 지정되지 않음 • Y = 데이터베이스 파티션 간의 트래픽을 최소화하기 위해 삽입이 코디네이터 데이터베이스 파티션에서 버퍼 지정됨

표 140. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
LANG_LEVEL	CHAR(1)	Y	LANGLEVEL 바인드 옵션 설정. • 0 = SAA1 • 1 = MIA • 2 = SQL92E
FUNC_PATH	CLOB(2K)		패키지가 바인드된 시간의 SQL 경로
QUERYOPT	INTEGER		패키지의 바인드에 따른 최적화 클래스입니다. 리바인드 조작에 사용됩니다.
EXPLAIN_LEVEL	CHAR(1)		Explain이 EXPLAIN 또는 EXPLSNAP 바인드 옵션을 사용하여 요청되었는지 여부를 표시합니다. • P = 패키지 선택 레벨 • 공백 = Explain이 요청되지 않음
EXPLAIN_MODE	CHAR(1)		EXPLAIN 바인드 옵션 값. • A = 모두 • N = 아니오 • R = REOPT • Y = 예
EXPLAIN_SNAPSHOT	CHAR(1)		EXPLSNAP 바인드 옵션 값. • A = 모두 • N = 아니오 • R = REOPT • Y = 예
SQLWARN	CHAR(1)		동적 SQL문의 결과인 양수 SQLCODE가 응용프로그램으로 리턴되는지 여부를 표시합니다. • N = 아니오. 억제됨 • Y = 예
SQLMATHWARN	CHAR(1)		바인드 시간에 *dft_sqlmathwarn* 데이터베이스 구성 매개변수의 값입니다. 산술 및 검색 변환 오류가 경고 및 널(NULL) 값(표시기 -2)을 리턴하여 가능한 경우 쿼리 처리가 계속될 수 있도록 허용하는지 여부를 표시합니다. • N = 아니오. 오류가 리턴됨 • Y = 예. 경고가 리턴됨
CREATE_TIME	TIMESTAMP		패키지가 처음 바인드된 시간.
EXPLICIT_BIND_TIME	TIMESTAMP		이 패키지가 다음에 의해 마지막으로 변경된 시간: • BIND • REBIND(명시적)
LAST_BIND_TIME	TIMESTAMP		패키지가 다음에 의해 마지막으로 변경된 시간: • BIND • REBIND(명시적) • REBIND(내재적)
ALTER_TIME	TIMESTAMP		이 패키지가 다음에 의해 마지막으로 변경된 시간: • BIND • REBIND(명시적) • REBIND(내재적) • ALTER PACKAGE
CODEPAGE	SMALLINT		바인드 시의 응용프로그램 코드 페이지. 알 수 없는 경우 -1입니다.
COLLATIONSCHEMA	VARCHAR(128)		패키지 조합의 스키마 이름.

표 140. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
COLLATIONNAME	VARCHAR(128)		패키지 조합의 규정되지 않은 이름.
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		패키지에서 ORDER BY절에 대한 조합의 스키마 이름.
COLLATIONNAME_ORDERBY	VARCHAR(128)		패키지의 ORDER BY절에 대한 조합의 규정되지 않은 이름.
DEGREE	CHAR(5)		패키지가 바운드될 때 지정된 파티션 내 병렬 처리의 등급. • 1 = 병렬 처리 없음 • 2 - 32767 = 사용자 지정 한계 • ANY = 시스템이 판별하는 등급(한계가 지정되지 않음)
MULTINODE_PLANS	CHAR(1)		• N = 패키지가 파티션된 데이터베이스 환경에서 바운드되지 않았음 • Y = 패키지가 파티션된 데이터베이스 환경에서 바운드되었음
INTRA_PARALLEL	CHAR(1)		패키지에서 정적 SQL문에 의한 파티션 내 병렬 처리의 사용. • F = 이 패키지의 하나 이상의 정적 SQL문이 파티션 내 병렬 처리를 사용할 수 있습니다. 이 병렬 처리는 파티션 내 병렬 처리에 대해 구성되지 않은 시스템에서의 사용에 대해 사용 불가능합니다. • N = 정적 SQL문이 파티션 내 병렬 처리를 사용하지 않음 • Y = 패키지의 하나 이상의 정적 SQL문이 파티션 내 병렬 처리를 사용함
VALIDATE	CHAR(1)		유효성 검사가 런타임까지 지연될 수 있는지 여부를 표시합니다. • B = 모든 검사가 바인드 시간에 수행되어야 함 • R = 바인드 시간에 존재하지 않는 테이블, 뷰 및 특권의 유효성 확인이 런타임 시에 수행됨
DYNAMICRULES	CHAR(1)		• B = BIND. 동적 SQL문이 DYNAMICRULES BIND 동작과 함께 실행됨 • D = DEFINERBIND. 패키지가 루틴 컨텍스트 내에서 실행될 때 패키지의 동적 SQL문은 DEFINE 동작으로 실행되며, 패키지가 루틴 컨텍스트 내에서 실행되지 않을 때 패키지의 동적 SQL문은 BIND 동작으로 실행됨 • E = DEFINERRUN. 패키지가 루틴 컨텍스트 내에서 실행될 때 패키지의 동적 SQL문은 DEFINE 동작으로 실행되며, 패키지가 루틴 컨텍스트 내에서 실행되지 않을 때 패키지의 동적 SQL문은 RUN 동작으로 실행됨 • H = INVOKEBIND. 패키지가 루틴 컨텍스트 내에서 실행될 때 패키지의 동적 SQL문은 INVOKE 동작으로 실행되며, 패키지가 루틴 컨텍스트 내에서 실행되지 않을 때 패키지의 동적 SQL문은 BIND 동작으로 실행됨 • I = INVOKERUN. 패키지가 루틴 컨텍스트 내에서 실행될 때 패키지의 동적 SQL문은 INVOKE 동작으로 실행되며, 패키지가 루틴 컨텍스트 내에서 실행되지 않을 때 패키지의 동적 SQL문은 RUN 동작으로 실행됨 • R = RUN. 동적 SQL문이 RUN 동작과 함께 실행됨. 이것이 디폴트임

표 140. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
SQLERROR	CHAR(1)		패키지를 바운드 또는 리바인드한 최신 서브명령의 SQLERROR 옵션. • C = CONTINUE. SQL문을 바인드하는 동안 오류가 발생하는 경우에도 패키지를 작성함 • N = NOPACKAGE. 오류가 발생한 경우 패키지 또는 바인드 파일을 작성하지 않음
REFRESHAGE	DECIMAL (20,6)		구체화된 쿼리 테이블(MQT)에 대한 REFRESH TABLE문의 실행과 기본 테이블 대신 MQT가 사용되는 시간 사이의 최대 시간 길이를 표시하는 시간소인 지속 기간.
FEDERATED	CHAR(1)		• N = FEDERATED 바인드 또는 prep 옵션이 꺼짐 • U = FEDERATED 바인드 또는 prep 옵션이 지정되지 않았음 • Y = FEDERATED 바인드 또는 prep 옵션이 켜짐
TRANSFORMGROUP	VARCHAR(1024)	Y	TRANSFORM GROUP 바인드 옵션의 값입니다. 변환 그룹이 지정되지 않은 경우 널(NULL) 값입니다.
REOPTVAR	CHAR(1)		액세스 경로가 실행 시간에 입력 변수값을 사용하여 다시 판별되는지 여부를 표시합니다. • A = 액세스 경로가 모든 OPEN 또는 EXECUTE 요청에 대해 다시 최적화됨 • N = 액세스 경로가 바인드 시간에 판별됨 • O = 액세스 경로가 첫 번째 OPEN 또는 EXECUTE 요청 시에만 다시 최적화되고, 그 뒤에는 캐시됨
OS_PTR_SIZE	INTEGER		패키지가 작성된 플랫폼에 대한 Word 크기. • 32 = 패키지가 32비트 패키지임 • 64 = 패키지가 64비트 패키지임
PKGVERSION	VARCHAR(64)		패키지의 버전 ID.
STATICREADONLY	CHAR(1)		정적 커서가 READ ONLY로 처리될 것인지의 여부를 나타냅니다. 가능한 값은 다음과 같습니다. • N = 정적 커서는 제공된 명령문 텍스트와 LANGLEVEL 프리컴파일 옵션 설정에 대해 정상적으로 생성된 속성에서 수행됨 • Y = FOR UPDATE 또는 FOR READ ONLY절을 포함하고 있지 않은 모든 정적 커서는 READ ONLY로 간주됨
FEDERATED_ASYNCHRONY	INTEGER		패키지가 바운드되었을 때 바인드 옵션으로 비동시성에 대한 한계(플랜의 ATQ 수)를 표시합니다. • 0 = 비동시성 없음 • *n* = 사용자 지정 한계(최대 32,767) • -1 = 시스템에서 판별하는 비동시성 등급 • -2 = 비동시성 등급이 지정되지 않음 비페더레이티드 시스템의 경우 값은 0입니다.
ANONBLOCK	CHAR(1)		• N = 패키지가 익명 블록과 연관되지 않음 • Y = 패키지가 익명 블록과 연관됨
OPTPROFILESCHEMA	VARCHAR(128)	Y	OPTPROFILE 바인드 옵션의 파트로서 지정되는 최적화 프로파일 스키마의 값.
OPTPROFILENAME	VARCHAR(128)	Y	OPTPROFILE 바인드 옵션의 파트로서 지정되는 최적화 프로파일 이름 값.
PKGID	BIGINT		패키지 ID.
DBPARTITIONNUM	SMALLINT		패키지가 바운드된 데이터베이스 파티션 번호.
DEFINER[2]	VARCHAR(128)		패키지 바인더 및 소유자의 권한 부여 ID
PKG_CREATE_TIME[3]	TIMESTAMP		패키지가 처음 바인드된 시간.

표 140. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
APREUSE	CHAR(1)		• N = 쿼리 컴파일러가 액세스 플랜을 재사용하려고 시도하지 않음 • Y = 이 패키지의 액세스 플랜이 재사용되어야 하며, 리바인드 시간에 쿼리 컴파일러가 현재 패키지에 있는 것과 유사한 플랜을 선택하려고 시도함을 의미함
EXTENDEDINDICATOR	CHAR(1)		• N = 확장 표시기 변수 값이 인식되지 않음 • Y = 확장 표시기 변수 값이 인식됨
LASTUSED	DATE		나중에 사용하기 위해 예약됨
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 종속성을 갖는 기능 인스턴스가 삭제되는 경우 패키지는 "작동 불능" 상태에 들어가며 명시적으로 리바인드되어야 합니다. 종속성을 갖는 다른 오브젝트가 삭제되는 경우 패키지는 "유효하지 않음" 상태에 들어가고 시스템이 처음으로 패키지를 참조할 때 자동으로 패키지를 리바인드하려고 시도합니다.
2. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.
3. 이전 버전과의 호환성을 위해 PKG_CREATE_TIME 컬럼이 포함됩니다. CREATE_TIME을 참조하십시오.

SYSCAT.PARTITIONMAPS

각 행은 테이블의 분산 해싱을 기초로 데이터베이스 파티션 그룹의 데이터베이스 파티션 사이에 테이블의 행을 분배하는 데 사용되는 분산 맵을 나타냅니다.

표 141. SYSCAT.PARTITIONMAPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
PMAP_ID	SMALLINT		분산 맵의 ID.
PARTITIONMAP	BLOB(65536)		다중 파티션 데이터베이스 파티션 그룹에 대한 32768개의 2바이트 정수의 벡터인 분산 맵입니다. 단일 파티션 데이터베이스 파티션 그룹의 경우 단일 파티션의 파티션 번호가 지정된 하나의 항목이 있습니다.

SYSCAT.PASSTHRUAUTH

각 행은 데이터 소스를 쿼리하는 pass-through 권한이 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 142. SYSCAT.PASSTHRUAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
SERVERNAME	VARCHAR(128)		권한이 부여되고 있는 데이터 소스의 이름.

SYSCAT.PREDICATESPECS

각 행은 술어 스펙을 나타냅니다.

표 143. SYSCAT.PREDICATESPECS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
FUNCSCHEMA	VARCHAR(128)		함수의 스키마 이름.
FUNCNAME	VARCHAR(128)		함수의 규정되지 않은 이름.
SPECIFICNAME	VARCHAR(128)		함수 인스턴스 이름.
FUNCID	INTEGER		함수 ID.
SPECID	SMALLINT		이 술어 스펙 번호.
CONTEXTOP	CHAR(8)		비교 연산자. 내장 관계 연산자(=, <, >, >= 등) 중 하나.
CONTEXTEXP	CLOB(2M)		상수 또는 SQL 표현식.
FILTERTEXT	CLOB(32K)	Y	데이터 필터 표현식의 텍스트.

SYSCAT.REFERENCES

각 행은 참조 무결성(외부 키) 제한을 나타냅니다.

표 144. SYSCAT.REFERENCES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONSTNAME	VARCHAR(128)		제한조건 이름.
TABSCHEMA	VARCHAR(128)		종속 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		종속 테이블의 규정되지 않은 이름.
OWNER	VARCHAR(128)		제한조건 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
REFKEYNAME	VARCHAR(128)		상위 키의 이름.
REFTABSCHEMA	VARCHAR(128)		상위 테이블의 스키마 이름.
REFTABNAME	VARCHAR(128)		상위 테이블의 규정되지 않은 이름.
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		제한조건이 정의된 시간.
FK_COLNAMES	VARCHAR(640)		이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다. 이 정보에 대해서는 SYSCAT.KEYCOLUSE를 사용하십시오.
PK_COLNAMES	VARCHAR(640)		이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다. 이 정보에 대해서는 SYSCAT.KEYCOLUSE를 사용하십시오.
DEFINER[1]	VARCHAR(128)		제한조건 소유자의 권한 부여 ID

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.ROLEAUTH

각 행은 사용자, 그룹, 역할 또는 PUBLIC에 권한 부여된 역할을 나타냅니다.

표 145. SYSCAT.ROLEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		역할을 권한 부여한 권한 부여 ID.
GRANTORTYPE	CHAR(1)		• U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		역할이 권한 부여된 권한 부여 ID.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
ROLENAME	VARCHAR(128)		역할 이름.
ROLEID	INTEGER		역할 ID.
ADMIN	CHAR(1)		역할을 다른 사용자에게 권한 부여 또는 권한 취소하거나 역할에 주석을 작성하는 특권. • N = 보유되지 않음 • Y = 보유됨

SYSCAT.ROLES

각 행은 역할을 나타냅니다.

표 146. SYSCAT.ROLES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROLENAME	VARCHAR(128)		역할 이름.
ROLEID	INTEGER		역할 ID.
CREATE_TIME	TIMESTAMP		역할이 작성된 시간.
AUDITPOLICYID	INTEGER	Y	감사 규정 ID.
AUDITPOLICYNAME	VARCHAR(128)	Y	감사 규정 이름.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.ROUTINEAUTH

각 행은 EXECUTE 특권이 부여된 사용자, 그룹 또는 역할을 나타냅니다.

- 모듈에 정의되지 않은 데이터베이스의 특정 루틴(함수, 메소드 또는 프로시저)
- 모듈에 정의되지 않은 데이터베이스의 특정 스키마에 있는 모든 루틴

표 147. SYSCAT.ROUTINEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자입니다. 시스템이 특권을 권한 부여한 경우 'SYSIBM'입니다.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
SCHEMA	VARCHAR(128)		루틴의 스키마 이름.
SPECIFICNAME	VARCHAR(128)	Y	루틴의 특정 이름입니다. SPECIFICNAME이 널(NULL) 값이고 ROUTINETYPE이 'M'이 아닌 경우, 특권이 SCHEMA에 지정된 스키마의 ROUTINETYPE에서 지정된 유형의 모든 루틴에 적용됩니다. SPECIFICNAME이 널(NULL) 값이고 ROUTINETYPE이 'M'인 경우, 특권이 TYPESCHEMA에 의해 지정되는 스키마에서 TYPENAME으로 지정되는 주제 유형에 대한 모든 메소드에 적용됩니다. SPECIFICNAME이 널(NULL) 값이고 ROUTINETYPE이 'M'이며 TYPENAME과 TYPESCHEMA가 모두 널(NULL) 값인 경우 특권은 스키마의 모든 유형에 대한 모든 메소드에 적용됩니다.
TYPESCHEMA	VARCHAR(128)	Y	메소드에 대한 유형의 스키마 이름입니다. ROUTINETYPE이 'M'이 아니면 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)	Y	메소드에 대한 유형의 규정되지 않은 이름입니다. ROUTINETYPE이 'M'이 아니면 널(NULL) 값입니다. TYPENAME이 널(NULL) 값이고 ROUTINETYPE이 'M'인 경우, 특권은 SCHEMA로 지정되는 스키마에 있는 모든 주제 유형의 모든 메소드에 적용됩니다.
ROUTINETYPE	CHAR(1)		루틴 유형. • F = 함수 • M = 메소드 • P = 프로시저

표 147. SYSCAT.ROUTINEAUTH 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
EXECUTEAUTH	CHAR(1)		루틴을 실행할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
GRANT_TIME	TIMESTAMP		특권이 부여된 시간.

SYSCAT.ROUTINEDEP

각 행은 일부 다른 오브젝트에 대한 루틴의 종속성을 나타냅니다. 루틴은 이름 BNAME의 유형 BTYPE의 오브젝트에 종속되므로, 오브젝트에 대한 변경사항이 루틴에 영향을 줍니다.

표 148. SYSCAT.ROUTINEDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROUTINESCHEMA	VARCHAR(128)		다른 오브젝트에 종속되는 루틴의 스키마 이름.
ROUTINEMODULENAME	VARCHAR(128)	Y	모듈의 규정되지 않은 이름.
SPECIFICNAME	VARCHAR(128)		다른 오브젝트에 종속되는 루틴의 특정 이름.
ROUTINEMODULEID	INTEGER	Y	다른 오브젝트에 종속되는 오브젝트 모듈의 ID.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • B = 트리거 • F = 루틴 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • K = 패키지 • L = 접속 해제된 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • Q = 시퀀스 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • X = 인덱스 확장 • Z = XSR 오브젝트 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.

표 148. SYSCAT.ROUTINEDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름입니다. 루틴(BTYPE = 'F')의 경우, 이것은 특정 이름입니다.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 루틴에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.
ROUTINENAME	VARCHAR(128)		이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다. SPECIFICNAME을 참조하십시오.

SYSCAT.ROUTINEOPTIONS

각 행은 루틴 특정 옵션 값을 나타냅니다.

표 149. SYSCAT.ROUTINEOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROUTINESCHEMA	VARCHAR(128)		ROUTINEMODULEID가 널(Null)인 경우 루틴의 스키마 이름이고, 그렇지 않으면 루틴이 속하는 모듈의 스키마 이름입니다.
ROUTINENAME	VARCHAR(128)		루틴의 규정되지 않은 이름.
SPECIFICNAME	VARCHAR(128)		루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
OPTION	VARCHAR(128)		페더레이티드 루틴 옵션 이름.
SETTING	VARCHAR(2048)		페더레이티드 루틴 옵션 값.

SYSCAT.ROUTINEPARMOPTIONS

각 행은 루틴 매개변수 특정 옵션 값을 나타냅니다.

표 150. SYSCAT.ROUTINEPARMOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROUTINESCHEMA	VARCHAR(128)		ROUTINEMODULEID가 널(Null)인 경우 루틴의 스키마 이름이고, 그렇지 않으면 루틴이 속하는 모듈의 스키마 이름입니다.
ROUTINENAME	VARCHAR(128)		루틴의 규정되지 않은 이름.
SPECIFICNAME	VARCHAR(128)		루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
ORDINAL	SMALLINT		루틴 서명에서 매개변수의 위치.
OPTION	VARCHAR(128)		페더레이티드 루틴 옵션 이름.
SETTING	VARCHAR(2048)		페더레이티드 루틴 옵션 값.

SYSCAT.ROUTINEPARMS

각 행은 SYSCAT.ROUTINES에서 정의되는 루틴의 매개변수 또는 결과를 나타냅니다.

표 151. SYSCAT.ROUTINEPARMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROUTINESCHEMA	VARCHAR(128)	Y	ROUTINEMODULEID가 널(Null)인 경우 루틴의 스키마 이름이고, 그렇지 않으면 루틴이 속하는 모듈의 스키마 이름입니다.
ROUTINEMODULENAME	VARCHAR(128)	Y	루틴이 속하는 모듈의 규정되지 않은 이름. 모듈 루틴이 아니면 널(NULL) 값입니다.
ROUTINENAME	VARCHAR(128)	Y	루틴의 규정되지 않은 이름.
ROUTINEMODULEID	INTEGER	Y	루틴이 속하는 모듈의 ID입니다. 모듈 루틴이 아니면 널(NULL) 값입니다.
SPECIFICNAME	VARCHAR(128)	Y	루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
PARMNAME	VARCHAR(128)	Y	매개변수 또는 결과 컬럼의 이름 또는 이름이 존재하지 않는 경우 널(NULL) 값입니다.
ROWTYPE	CHAR(1)		• B = 입력 및 출력 매개변수 둘 다 • C = 캐스팅 후 결과 • O = 출력 매개변수 • P = 입력 매개변수 • R = 캐스팅 전 결과
ORDINAL	SMALLINT		ROWTYPE = 'B', 'O' 또는 'P'인 경우, 1에서 시작하여 루틴 서명에서 매개변수의 숫자 위치입니다. ROWTYPE = 'R'이고 루틴이 테이블을 리턴하는 경우, 결과 테이블에서 이름 지정된 컬럼의 숫자 위치로서 1에서 시작합니다. 그렇지 않으면 0입니다.
TYPESCHEMA	VARCHAR(128)	Y	TYPEMODULEID가 널(Null)인 경우 데이터 유형의 스키마 이름이고, 그렇지 않으면 데이터 유형이 속하는 모듈의 스키마 이름입니다.
TYPEMODULENAME	VARCHAR(128)	Y	매개변수 또는 결과의 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 모듈 데이터 유형이 아닌 경우 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)	Y	데이터 유형의 규정되지 않은 이름.
LOCATOR	CHAR(1)		• N = 매개변수 또는 결과가 로케이터 양식으로 전달되지 않음 • Y = 매개변수 또는 결과가 로케이터 양식으로 전달됨
LENGTH[1]	INTEGER		매개변수 또는 결과의 길이입니다. 매개변수 또는 결과가 사용자 정의 데이터 유형인 경우 0입니다.

표 151. SYSCAT.ROUTINEPARMS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
SCALE[1]	SMALLINT		매개변수 또는 결과 데이터 유형이 DECIMAL인 경우 스케일이고, 매개변수 또는 결과 데이터 유형이 TIMESTAMP인 경우 분수 초의 자릿수입니다. 그렇지 않으면, 0입니다.
CODEPAGE	SMALLINT		이 매개변수 또는 결과의 코드 페이지입니다. 0은 적용할 수 없음 또는 FOR BIT DATA 속성과 함께 선언된 문자 데이터에 대한 매개변수 또는 결과를 나타냅니다.
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 매개변수 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 매개변수에 대한 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
CAST_FUNCSCHEMA	VARCHAR(128)	Y	인수 또는 결과를 캐스트하는 데 사용된 함수의 스키마 이름입니다. 소스 및 외부 함수에 적용됩니다. 그렇지 않으면, 널(NULL) 값입니다.
CAST_FUNCSPECIFIC	VARCHAR(128)	Y	인수 또는 결과를 캐스트하는 데 사용된 함수의 규정되지 않은 이름입니다. 소스 및 외부 함수에 적용됩니다. 그렇지 않으면, 널(NULL) 값입니다.
TARGET_TYPESCHEMA	VARCHAR(128)	Y	매개변수 또는 결과 유형이 REFERENCE인 경우 목표 유형의 스키마 이름. 매개변수 또는 결과의 유형이 REFERENCE가 아닌 경우 널(NULL) 값입니다.
TARGET_TYPEMODULENAME	VARCHAR(128)	Y	매개변수 또는 결과의 유형이 REFERENCE인 경우, 목표 유형이 속하는 모듈의 규정되지 않은 이름. 매개변수 또는 결과의 유형이 REFERENCE가 아니거나 목표 유형이 모듈 데이터 유형이 아닌 경우 널(NULL) 값.
TARGET_TYPENAME	VARCHAR(128)	Y	매개변수 또는 결과의 유형이 REFERENCE인 경우, 목표 유형이 속하는 모듈의 규정되지 않은 이름. 매개변수 또는 결과의 유형이 REFERENCE가 아니거나 목표 유형이 모듈 데이터 유형이 아닌 경우 널(NULL) 값.
SCOPE_TABSCHEMA	VARCHAR(128)	Y	매개변수 유형이 REFERENCE인 경우 범위(목표 테이블)의 스키마 이름이고 그 이외의 경우에는 널(NULL) 값.
SCOPE_TABNAME	VARCHAR(128)	Y	매개변수 유형이 REFERENCE인 경우 범위(목표 테이블)의 규정되지 않은 이름이고, 그 이외의 경우에는 널(NULL) 값.
TRANSFORMGRPNAME	VARCHAR(128)	Y	구조화된 유형 매개변수 또는 결과의 변환 그룹 이름
DEFAULT	CLOB(64K)	Y	매개변수의 디폴트값을 계산하는 데 사용되는 표현식입니다. 매개변수에 대해 DEFAULT절이 지정되지 않은 경우 널(NULL) 값입니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

표 151. SYSCAT.ROUTINEPARMS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명

주:

1. 전래 함수(다른 함수에 대한 참조를 사용하여 정의되는 함수)의 경우 LENGTH 및 SCALE은 0으로 설정됩니다. 이들 함수는 소스로부터 매개변수의 길이와 스케일을 상속하기 때문입니다.

SYSCAT.ROUTINES

각 행은 사용자 정의 루틴(스칼라 함수, 테이블 함수, 전래 함수, 메소드 또는 프로시저)을 나타냅니다. 내장 함수를 포함하지 않습니다.

표 152. SYSCAT.ROUTINES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROUTINESCHEMA	VARCHAR(128)		ROUTINEMODULEID가 널(Null)인 경우 루틴의 스키마 이름이고, 그렇지 않으면 루틴이 속하는 모듈의 스키마 이름입니다.
ROUTINEMODULENAME	VARCHAR(128)	Y	루틴이 속하는 모듈의 규정되지 않은 이름. 모듈 루틴이 아니면 널(NULL) 값입니다.
ROUTINENAME	VARCHAR(128)		루틴의 규정되지 않은 이름.
ROUTINETYPE	CHAR(1)		루틴의 유형. • F = 함수 • M = 메소드 • P = 프로시저
OWNER	VARCHAR(128)		루틴 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
SPECIFICNAME	VARCHAR(128)		루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
ROUTINEID	INTEGER		루틴의 ID.
ROUTINEMODULEID	INTEGER	Y	루틴이 속하는 모듈의 ID입니다. 모듈 루틴이 아니면 널(NULL) 값입니다.
RETURN_TYPESCHEMA	VARCHAR(128)	Y	스칼라 함수 또는 메소드에 대한 리턴 유형의 스키마 이름입니다.
RETURN_TYPEMODULE	VARCHAR(128)	Y	리턴 유형의 모듈 이름입니다. 리턴 유형이 어떤 모듈에도 속하지 않는 경우 널(NULL) 값입니다.
RETURN_TYPENAME	VARCHAR(128)	Y	스칼라 함수 또는 메소드에 대한 리턴 유형의 규정되지 않은 이름입니다.
ORIGIN	CHAR(1)		• B = 내장 • E = 사용자 정의, 외부 • M = 템플리트 함수 • F = 페더레이티드 프로시저 • Q = SQL 본문[1] • R = 시스템 생성 SQL 본문 루틴 • S = 시스템 생성 • T = 시스템 생성 변환 함수(직접 호출할 수 없음) • U = 사용자 정의, 소스 기반
FUNCTIONTYPE	CHAR(1)		• C = 컬럼 또는 집계 • R = 행 • S = 스칼라 • T = 테이블 • 공백 = 프로시저

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
PARM_COUNT	SMALLINT		루틴 매개변수의 수.
LANGUAGE	CHAR(8)		루틴 본문(또는 이 함수가 다른 함수의 전래 함수인 경우, 소스 함수 본문)에 대한 구현 언어. 가능한 값은 'C', 'COBOL', 'JAVA', 'OLE', 'OLEDB' 또는 'SQL'입니다. ORIGIN이 'E', 'Q' 또는 'R'이 아닌 경우 공백입니다.
DIALECT	VARCHAR(10)		SQL 루틴 본문의 소스 표현 방식. • DB2 SQL PL • PL/SQL • 공백 = SQL 루틴이 아님
SOURCESCHEMA	VARCHAR(128)	Y	ORIGIN = 'U'이고 소스 함수가 사용자 정의 함수인 경우, 소스 함수의 특정 이름에 대한 스키마 이름을 포함합니다. ORIGIN = 'U'이고 소스 함수가 내장 함수인 경우 값 'SYSIBM'을 포함합니다. ORIGIN이 'U'가 아니면 널(NULL) 값입니다.
SOURCESPECIFIC	VARCHAR(128)	Y	ORIGIN = 'U'이고 소스 함수가 사용자 정의 함수인 경우, 소스 함수의 규정되지 않은 특정 이름을 포함합니다. ORIGIN = 'U'이고 소스 함수가 내장 함수인 경우 '내장의 경우 N/A' 값을 포함합니다. ORIGIN이 'U'가 아니면 널(NULL) 값입니다.
PUBLISHED	CHAR(1)		모듈 루틴을 모듈 외부에서 호출할 수 있는지 여부를 표시합니다. • N = 모듈 루틴이 발행되지 않음 • Y = 모듈 루틴이 발행됨 • 공백 = 적용할 수 없음
DETERMINISTIC	CHAR(1)		• N = 결과가 결정적이 아님(같은 매개변수가 다른 루틴 호출에서 다른 결과가 발생할 수 있음) • Y = 결과가 결정적임 • 공백 = ORIGIN 이 'E', 'F', 'Q' 또는 'R'이 아님
EXTERNAL_ACTION	CHAR(1)		• E = 함수에 외부 부가작용이 있음(그러므로 호출 수가 중요함) • N = 부가작용 없음 • 공백 = ORIGIN 이 'E', 'F', 'Q' 또는 'R'이 아님
NULLCALL	CHAR(1)		• N = RETURNS NULL ON NULL INPUT(하나 이상의 피연산자가 널(NULL)인 경우 함수 결과가 내재적으로 널(NULL) 값임) • Y = CALLED ON NULL INPUT • 공백 = ORIGIN이 'E', 'Q' 또는 'R'이 아님
CAST_FUNCTION	CHAR(1)		• N = 캐스트(cast) 함수가 아님 • Y = 캐스트(cast) 함수 • 공백 = ROUTINETYPE이 'F'가 아님

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
ASSIGN_FUNCTION	CHAR(1)		• N = 지정 함수가 아님 • Y = 내재된 지정 함수 • 공백 = ROUTINETYPE이 'F'가 아님
SCRATCHPAD	CHAR(1)		• N = 루틴에 스크래치 패드가 없음 • Y = 루틴에 스크래치 패드가 있음 • 공백 = ORIGIN이 'E'가 아니거나 ROUTINETYPE이 'P'임
SCRATCHPAD_LENGTH	SMALLINT		루틴의 스크래치 패드 크기(바이트). • -1 = LANGUAGE가 'OLEDB'이고 SCRATCHPAD가 'Y'임 • 0 = SCRATCHPAD가 'Y'가 아님
FINALCALL	CHAR(1)		• N = 최종 호출이 작성되지 않음 • Y = 런타임 명령문의 끝에서 이 루틴에 최종 호출이 작성됨 • 공백 = ORIGIN이 'E'가 아니거나 ROUTINETYPE이 'P'임
PARALLEL	CHAR(1)		• N = 루틴이 병렬로 실행될 수 없음 • Y = 루틴이 병렬로 실행될 수 있음 • 공백 = ORIGIN이 'E'가 아님
PARAMETER_STYLE	CHAR(8)		루틴이 작성될 때 선언된 매개변수 스타일. 가능한 값은 다음과 같습니다. • DB2DARI • DB2GENRL • DB2SQL • GENERAL • GNRLNULL • JAVA • SQL • ORIGIN이 'E'가 아닌 경우 공백
FENCED	CHAR(1)		• N = 분리(fenced)되지 않음 • Y = 분리(fenced)됨 • 공백 = ORIGIN이 'E'가 아님
SQL_DATA_ACCESS	CHAR(1)		데이터베이스 관리 프로그램이 루틴에 포함되어 있다고 가정하는 SQL문의 유형(있는 경우)을 표시합니다. • C = SQL 포함(부속 쿼리가 없는 단순 표현식만) • M = 데이터를 수정하는 SQL문 포함 • N = SQL문 포함하지 않음 • R = 읽기 전용 SQL문 포함 • 공백 = ORIGIN 이 'E', 'F', 'Q' 또는 'R'이 아님

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
DBINFO	CHAR(1)		DBINFO 매개변수가 외부 루틴으로 전달되는지 여부를 표시합니다. • N = DBINFO가 전달되지 않음 • Y = DBINFO가 전달됨 • 공백 = ORIGIN이 'E'가 아님
PROGRAMTYPE	CHAR(1)		외부 루틴이 호출되는 방법을 나타냅니다. • M = 주 • S = 서브루틴 • 공백 = ORIGIN이 'F'임
COMMIT_ON_RETURN	CHAR(1)		트랜잭션이 이 프로시저에서 성공적으로 리턴할 때 커미트되는지 여부를 표시합니다. • N = 작업 단위(UOW)가 커미트되지 않음 • Y = 작업 단위(UOW)가 커미트됨 • 공백 = ROUTINETYPE이 'P'가 아님
AUTONOMOUS	CHAR(1)		루틴이 자동으로 실행되는지의 여부를 표시합니다. • N = 루틴이 호출 트랜잭션에서 자율적으로 실행되지 않음 • Y = 루틴이 호출 트랜잭션에서 자율적으로 실행됨 • 공백 = ROUTINETYPE이 'P'가 아님
RESULT_SETS	SMALLINT		추정되는 최대 결과 세트 수.
SPEC_REG	CHAR(1)		루틴을 호출할 때 사용되는 특수 레지스터 값을 표시합니다. • I = 상속된 특수 레지스터 • 공백 = PARAMETER_STYLE이 'DB2DARI'이거나 ORIGIN이 'E', 'Q' 또는 'R'이 아님
FEDERATED	CHAR(1)		페더레이티드 오브젝트를 루틴에서 액세스할 수 있는지 여부를 표시합니다. • Y = 페더레이티드 오브젝트에 액세스할 수 있음 • 공백 = ORIGIN 이 'F'가 아님
THREADSAFE	CHAR(1)		루틴이 다른 루틴과 동일한 프로세스에서 실행할 수 있는지 여부를 표시합니다. • N = 루틴이 스레드 안전하지 않음 • Y = 루틴이 스레드 안전함 • 공백 = ORIGIN이 'E'가 아님
VALID	CHAR(1)		LANGUAGE = 'SQL' 및 디폴트인 매개변수를 갖는 루틴에 적용되고, 그렇지 않으면 공백입니다. • N = 루틴이 리바인드가 필요함 • X = 루틴이 작동 불능하며 다시 작성해야 함 • Y = 루틴이 유효함
MODULEROUTINEIMPLEMENTED	CHAR(1)		• N = 모듈 루틴 본문이 구현되지 않음 • Y = 모듈 루틴 본문이 구현됨 • 공백 = ROUTINEMODULENAME이 널(null) 값임

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
METHODIMPLEMENTED	CHAR(1)		• N = 메소드 본문이 구현되지 않음 • Y = 메소드 본문이 구현됨 • 공백 = ROUTINETYPE이 'M'이 아니거나 ROUTINEMODULENAME이 널(NULL) 값이 아님
METHODEFFECT	CHAR(2)		• CN = 컨스트럭터 메소드 • MU = Mutator 메소드 • OB = 관찰자 메소드 • 공백 = 시스템 메소드가 아님
TYPE_PRESERVING	CHAR(1)		• N = 리턴 유형이 메소드의 선언된 리턴 유형임 • Y = 리턴 유형이 ″type-preserving″ 매개변수에 의해 조정됨. 모든 시스템 생성 mutator 메소드는 type-preserving임 • 공백 = ROUTINETYPE이 'M'이 아님
WITH_FUNC_ACCESS	CHAR(1)		• N = 이 메소드는 함수 표기를 사용하여 호출할 수 없음 • Y = 이 메소드는 함수 표기를 사용하여 호출할 수 있음. 즉 ″WITH FUNCTION ACCESS″ 속성이 지정됨 • 공백 = ROUTINETYPE이 'M'이 아님
OVERRIDDEN_METHODID	INTEGER	Y	사용자 정의 메소드에 대해 OVERRIDING 옵션이 지정될 때 겹쳐써진 메소드의 ID입니다. ROUTINETYPE이 'M'이 아니면 널(NULL) 값입니다.
SUBJECT_TYPESCHEMA	VARCHAR(128)	Y	사용자 정의 메소드에 대한 주제 유형의 스키마 이름입니다. ROUTINETYPE이 'M'이 아니면 널(NULL) 값입니다.
SUBJECT_TYPENAME	VARCHAR(128)	Y	사용자 정의 메소드에 대한 주제 유형의 규정되지 않은 이름입니다. ROUTINETYPE이 'M'이 아니면 널(NULL) 값입니다.
CLASS	VARCHAR(384)	Y	LANGUAGE JAVA, CLR 또는 OLE의 경우, 이 루틴을 구현하는 클래스입니다. 그렇지 않으면 널(NULL) 값입니다.
JAR_ID	VARCHAR(128)	Y	LANGUAGE JAVA의 경우, JAR 파일이 루틴 작성 시에 지정되면 이 루틴을 구현하는 설치된 JAR 파일의 JAR_ID입니다. 그렇지 않으면 널(NULL) 값입니다. LANGUAGE CLR의 경우, 이 루틴을 구현하는 어셈블리 파일입니다.
JARSCHEMA	VARCHAR(128)	Y	LANGUAGE JAVA의 경우 JAR_ID가 존재할 때 이 루틴을 구현하는 JAR 파일의 스키마 이름입니다. 그렇지 않으면 널(NULL) 값입니다.
JAR_SIGNATURE	VARCHAR(2048)	Y	LANGUAGE JAVA의 경우 이는 이 루틴에서 지정한 Java 메소드의 메소드 서명입니다. LANGUAGE CLR의 경우, 이 CLR 루틴에 대한 참조 필드입니다. 그 외에는 널(NULL) 값입니다.
CREATE_TIME	TIMESTAMP		루틴이 작성된 시간.
ALTER_TIME	TIMESTAMP		루틴이 마지막으로 변경된 시간.

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
FUNC_PATH	CLOB(2K)	Y	루틴이 정의된 시간의 SQL 경로. LANGUAGE가 'SQL'이 아닌 경우 널(NULL) 값입니다.
QUALIFIER	VARCHAR(128)		오브젝트 정의 시 디폴트 스키마의 값입니다. 규정되지 않은 참조를 완료하는 데 사용됩니다.
IOS_PER_INVOC	DOUBLE		개략적인 호출당 입출력(I/O) 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
INSTS_PER_INVOC	DOUBLE		개략적인 호출당 명령어 수입니다. 디폴트는 450이고, 알 수 없는 경우 -1입니다.
IOS_PER_ARGBYTE	DOUBLE		개략적인 입력 인수 바이트당 I/O 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
INSTS_PER_ARGBYTE	DOUBLE		개략적인 입력 인수 바이트당 명령어 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
PERCENT_ARGBYTES	SMALLINT		루틴이 실제로 읽을 입력 인수 바이트의 개략적인 평균 퍼센트입니다. 디폴트는 100이며, 알 수 없는 경우 -1입니다.
INITIAL_IOS	DOUBLE		처음 루틴이 호출될 때 수행되는 I/O의 개략적인 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
INITIAL_INSTS	DOUBLE		처음 루틴이 호출될 때 실행되는 명령어의 개략적인 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
CARDINALITY	BIGINT		테이블 함수의 예상 카디널리티(cardinality)입니다. 알 수 없는 경우 또는 루틴이 테이블 함수가 아닌 경우 -1입니다.
SELECTIVITY[2]	DOUBLE		사용자 정의 술어의 경우입니다. 사용자 정의 술어가 없는 경우 -1입니다.
RESULT_COLS	SMALLINT		테이블 함수(ROUTINETYPE = 'F' 및 FUNCTIONTYPE = 'T')의 경우, 결과 테이블의 컬럼 수가 들어있습니다. 프로시저(ROUTINETYPE = 'P')의 경우, 0을 포함합니다. 그렇지 않으면 1을 포함합니다.
IMPLEMENTATION	VARCHAR(762)	Y	ORIGIN = 'E'인 경우, 이 함수를 구현하는 경로/모듈/함수를 식별합니다. ORIGIN = 'U'이고 소스 함수가 내장 함수인 경우, 이 컬럼에는 소스 함수의 이름 및 서명이 들어있습니다. 그 외에는 널(NULL) 값입니다.
LIB_ID	INTEGER	Y	나중에 사용하기 위해 예약됨
TEXT_BODY_OFFSET	INTEGER		LANGUAGE = 'SQL'인 경우, CREATE문의 전체 텍스트에서 SQL 프로시저 본문의 시작에 대한 오프셋입니다. LANGUAGE가 'SQL'이 아닌 경우 -1입니다.
TEXT	CLOB(2M)	Y	LANGUAGE = 'SQL'인 경우, CREATE FUNCTION, CREATE METHOD 또는 CREATE PROCEDURE문의 전체 텍스트입니다. 이 외에는 널(NULL) 값입니다.
NEWSAVEPOINTLEVEL	CHAR(1)		루틴이 호출될 때 루틴이 새 세이브포인트 레벨을 시작하는지 여부를 표시합니다. • N = 이 루틴이 호출될 때 새 세이브포인트 레벨이 시작되지 않습니다. 루틴은 기존 세이브포인트 레벨을 사용함 • Y = 루틴이 호출될 때 새 세이브포인트 레벨이 시작됨 • 공백 = 적용할 수 없음

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
DEBUG_MODE[3]	VARCHAR(8)		DB2 디버거를 사용하는 루틴의 디버그 가능 여부를 나타냅니다. • DISALLOW = 루틴이 디버깅 불가능함 • ALLOW = 루틴을 디버그할 수 있으며 DB2 디버거로 클라이언트 디버그 세션에 참여할 수 있음 • DISABLE = 루틴이 디버그 가능하지 않고, 이 설정은 루틴을 삭제 및 재작성하지 않고 변경할 수 없음 • Blank = DB2 디버거가 현재 루틴 유형을 지원하지 않음
TRACE_LEVEL	VARCHAR (1)	Y	나중에 사용하기 위해 예약됨
DIAGNOSTIC_LEVEL	VARCHAR (1)	Y	나중에 사용하기 위해 예약됨
CHECKOUT_USERID	VARCHAR(128)	Y	오브젝트 체크아웃을 수행한 사용자의 ID이며, 오브젝트가 체크아웃되지 않은 경우 널(NULL) 값입니다.
PRECOMPILE_OPTIONS	VARCHAR(1024)	Y	루틴에 대해 지정된 프리컴파일 옵션.
COMPILE_OPTIONS	VARCHAR(1024)	Y	루틴에 대해 지정된 컴파일 옵션.
EXECUTION_CONTROL	CHAR(1)		공통 언어 런타임(CLR) 루틴의 실행 제어 모드입니다. 가능한 값은 다음과 같습니다. • N = 네트워크 • R = 파일읽기 • S = 안전 • U = 불안전 • W = 파일쓰기 • 공백 = LANGUAGE가 'CLR'이 아님
CODEPAGE	SMALLINT		루틴 본문에서 모든 문자 매개변수 유형, 결과 유형 및 로컬 변수에 사용되는 디폴트 코드를 지정하는 루틴 코드 페이지.
COLLATIONSCHEMA	VARCHAR(128)		루틴 조합의 스키마 이름.
COLLATIONNAME	VARCHAR(128)		루틴 조합의 규정되지 않은 이름.
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		루틴의 ORDER BY절에 대한 조합의 스키마 이름.
COLLATIONNAME_ORDERBY	VARCHAR(128)		루틴의 ORDER BY절에 대한 조합의 규정되지 않은 이름.
ENCODING_SCHEME	CHAR(1)		PARAMETER CCSID절에서 지정되는 루틴의 인코딩 스킴입니다. 가능한 값은 다음과 같습니다. • A = ASCII • U = UNICODE • 공백 = PARAMETER CCSID절이 지정되지 않았음
LAST_REGEN_TIME	TIMESTAMP		SQL 루틴 팩형 디스크립터가 마지막으로 다시 만들어진 시간.

표 152. SYSCAT.ROUTINES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
INHERITLOCKREQUEST	CHAR(1)		• N = 지정된 isolation-clause에 lock-request-clause가 포함된 SQL문의 컨텍스트에서 이 함수 또는 메소드를 호출할 수 없음 • Y = 이 함수 또는 메소드가 호출 명령문의 분리 레벨을 상속합니다. 또한 지정된 잠금 요청 절을 상속합니다. • 공백 = LANGUAGE가 'SQL'이 아니거나 ROUTINETYPE이 'P'임
DEFINER[4]	VARCHAR(128)		루틴 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 버전 8.2 이전에 작성되고 버전 9로 업그레이드된 SQL 프로시저의 경우 'E'('Q' 대신)입니다.
2. 데이터베이스 업그레이드 중에 SELECTIVITY 컬럼은 모든 사용자 정의 루틴에 대한 팩형 디스크립터 및 시스템 카탈로그에서 -1로 설정됩니다. 사용자 정의 술어의 경우 시스템 카탈로그에서의 선택 빈도가 -1입니다. 이 경우 옵티마이저가 사용하는 선택 빈도 값은 0.01입니다.
3. Java 루틴의 경우 DEBUG_MODE 설정이 Java 루틴이 실제로 디버그 모드에서 컴파일되었는지 또는 디버그 Jar가 서버에 설치되었는지를 표시하지는 않습니다.
4. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.ROUTINESFEDERATED

각 행은 페더레이티드 프로시저를 나타냅니다.

표 153. SYSCAT.ROUTINESFEDERATED 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ROUTINESCHEMA	VARCHAR(128)		ROUTINEMODULEID가 널(Null)인 경우 루틴의 스키마 이름이고, 그렇지 않으면 루틴이 속하는 모듈의 스키마 이름입니다.
ROUTINENAME	VARCHAR(128)		루틴의 규정되지 않은 이름.
ROUTINETYPE	CHAR(1)		루틴의 유형. • P = 프로시저
OWNER	VARCHAR(128)		루틴 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
SPECIFICNAME	VARCHAR(128)		루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
ROUTINEID	INTEGER		루틴의 ID.
PARM_COUNT	SMALLINT		루틴 매개변수의 수.
DETERMINISTIC	CHAR(1)		• N = 결과가 결정적이 아님(같은 매개변수는 다른 루틴 호출에서 다른 결과를 제공할 수 있음) • Y = 결과가 결정적임
EXTERNAL_ACTION	CHAR(1)		• E = 루틴에 외부 부가작용이 있음(그러므로 호출 수가 중요함) • N = 부가작용 없음
SQL_DATA_ACCESS	CHAR(1)		데이터베이스 관리 프로그램이 루틴에 포함되어 있다고 가정하는 SQL문의 유형(있는 경우)을 표시합니다. • C = SQL 포함(부속 쿼리가 없는 단순 표현식만) • M = 데이터를 수정하는 SQL문 포함 • N = SQL문 포함하지 않음 • R = 읽기 전용 SQL문 포함
COMMIT_ON_RETURN	CHAR(1)		트랜잭션이 이 프로시저에서 성공적으로 리턴할 때 커미트되는지 여부를 표시합니다. • N = 작업 단위(UOW)가 커미트되지 않음 • Y = 작업 단위(UOW)가 커미트됨 • 공백 = ROUTINETYPE이 'P'가 아님
RESULT_SETS	SMALLINT		추정되는 최대 결과 세트 수.
CREATE_TIME	TIMESTAMP		루틴이 작성된 시간.
ALTER_TIME	TIMESTAMP		루틴이 마지막으로 변경된 시간.
QUALIFIER	VARCHAR(128)		오브젝트 정의 시 디폴트 스키마의 값입니다. 규정되지 않은 참조를 완료하는 데 사용됩니다.

표 153. SYSCAT.ROUTINESFEDERATED 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
RESULT_COLS	SMALLINT		루틴의 경우(ROUTINETYPE = 'P') 0을 포함하고, 그 외에는 1을 포함합니다.
CODEPAGE	SMALLINT		루틴 본문에서 모든 문자 매개변수 유형, 결과 유형 및 로컬 변수에 사용되는 디폴트 코드를 지정하는 루틴 코드 페이지.
LAST_REGEN_TIME	TIMESTAMP		SQL 루틴 패형 디스크립터가 마지막으로 다시 만들어진 시간.
REMOTE_PROCEDURE	VARCHAR(128)	Y	페더레이티드 루틴이 작성된 소싱된 프로시저의 규정되지 않은 이름.
REMOTE_SCHEMA	VARCHAR(128)	Y	페더레이티드 루틴이 작성된 소싱된 프로시저의 스키마 이름.
SERVERNAME	VARCHAR(128)	Y	페더레이티드 루틴이 작성된 소싱된 프로시저를 포함하는 데이터 소스의 이름.
REMOTE_PACKAGE	VARCHAR(128)	Y	소싱된 프로시저가 속하는 패키지의 이름(Oracle 데이터 소스의 경우 랩퍼에만 적용됨).
REMOTE_PROCEDURE_ ALTER_TIME	VARCHAR(128)	Y	나중에 사용하기 위해 예약됨
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.ROWFIELDS

각 행은 사용자 정의 행 데이터 유형에 대해 정의되는 필드를 나타냅니다.

표 154. SYSCAT.ROWFIELDS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPESCHEMA	VARCHAR(128)		필드를 포함하는 행 데이터 유형의 스키마 이름.
TYPEMODULENAME	VARCHAR(128)	Y	행 데이터 유형이 속한 모듈의 규정되지 않은 이름. 모듈 행 데이터 유형이 아닌 경우에는 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)		필드를 포함하는 행 데이터 유형의 규정되지 않은 이름.
FIELDNAME	VARCHAR(128)		필드 이름.
FIELDTYPESCHEMA	VARCHAR(128)		필드의 데이터 유형의 스키마 이름.
FIELDTYPEMODULENAME	VARCHAR(128)	Y	필드의 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 필드 데이터 유형이 모듈 사용자 정의 데이터 유형이 아닌 경우 널(NULL) 값입니다.
FIELDTYPENAME	VARCHAR(128)		필드의 데이터 유형의 규정되지 않은 이름.
ORDINAL	SMALLINT		행 데이터 유형의 정의에서 필드의 위치이며, 0에서 시작합니다.
LENGTH	INTEGER		필드 데이터 유형의 길이입니다. 10진수 유형의 경우 정밀도를 포함합니다.
SCALE	SMALLINT		10진수 유형의 경우 필드 데이터 유형의 스케일을 포함하고, 시간소인 유형의 경우 필드 데이터 유형의 시간소인 정밀도를 포함하며, 그 외에는 0입니다.
CODEPAGE	SMALLINT		문자열 유형의 경우 코드 페이지를 표시합니다. 0은 FOR BIT DATA를 표시합니다. 비문자열 유형의 경우 0입니다.
COLLATIONSCHEMA	VARCHAR(128)	Y	문자열 유형의 경우 필드 조합의 스키마 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.
COLLATIONNAME	VARCHAR(128)	Y	문자열 유형의 경우 필드 조합의 규정되지 않은 이름입니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.SCHEMAAUTH

각 행은 스키마에 대해 하나 이상의 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 155. SYSCAT.SCHEMAAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권의 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
SCHEMANAME	VARCHAR(128)		이 특권이 적용되는 스키마 이름.
ALTERINAUTH	CHAR(1)		이름 지정된 스키마에 있는 오브젝트에 주석을 달거나 변경할 수 있는 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
CREATEINAUTH	CHAR(1)		이름 지정된 스키마에 오브젝트를 작성할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
DROPINAUTH	CHAR(1)		이름 지정된 스키마에서 오브젝트를 삭제할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.SCHEMATA

각 행은 스키마를 나타냅니다.

표 156. SYSCAT.SCHEMATA 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
SCHEMANAME	VARCHAR(128)		스키마 이름.
OWNER	VARCHAR(128)		스키마 소유자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
DEFINER	VARCHAR(128)		스키마의 소유권이 전송된 경우, 스키마 소유자의 권한 부여 ID 또는 스키마 정의자의 권한 부여 ID
DEFINERTYPE	CHAR(1)		• S = 정의자가 시스템임 • U = 정의자가 개별 사용자임
CREATE_TIME	TIMESTAMP		스키마가 작성된 시간.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.SECURITYLABELACCESS

각 행은 데이터베이스 권한 부여 ID에 권한 부여되었던 보안 레이블을 나타냅니다.

표 157. SYSCAT.SECURITYLABELACCESS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		보안 레이블의 권한 준 사용자.
GRANTEE	VARCHAR(128)		보안 레이블 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
SECLABELID	INTEGER		보안 레이블 ID입니다. 보안 레이블 이름의 경우 SYSCAT.SECURITYLABELS 카탈로그 뷰의 대응하는 SECLABELID 값에 대한 SECLABELNAME 컬럼을 선택하십시오.
SECPOLICYID	INTEGER		보안 레이블과 연관되는 보안 규정의 ID입니다. 보안 규정 이름의 경우, SYSCAT.SECURITYPOLICIES 카탈로그 뷰에서 대응하는 SECPOLICYID 값에 대한 SECPOLICYNAME 컬럼을 선택하십시오.
ACCESSTYPE	CHAR(1)		• B = 읽기 및 쓰기 액세스 둘 다 • R = 읽기 액세스 • W = 쓰기 액세스
GRANT_TIME	TIMESTAMP		보안 레이블이 부여된 시간.

SYSCAT.SECURITYLABELCOMPONENTELEMENTS

각 행은 보안 레이블 구성요소의 요소 값을 나타냅니다.

표 158. SYSCAT.SECURITYLABELCOMPONENTELEMENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
COMPID	INTEGER		보안 레이블 구성요소 ID.
ELEMENTVALUE	VARCHAR(32)		보안 레이블 구성요소의 요소 값.
ELEMENTVALUEENCODING	CHAR(8) FOR BIT DATA		요소 값의 코드화 양식.
PARENTELEMENTVALUE	VARCHAR(32)	Y	트리 구성요소의 경우 요소 상위의 이름이고, 세트 및 배열 구성요소 및 트리 구성요소의 ROOT 노드의 경우 널(NULL) 값.

SYSCAT.SECURITYLABELCOMPONENTS

각 행은 보안 레이블 구성요소를 나타냅니다.

표 159. SYSCAT.SECURITYLABELCOMPONENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
COMPNAME	VARCHAR(128)		보안 레이블 구성요소 이름.
COMPID	INTEGER		보안 레이블 구성요소 ID.
COMPTYPE	CHAR(1)		보안 레이블 구성요소 유형. • A = 배열 • S = 세트 • T = 트리
NUMELEMENTS	INTEGER		보안 레이블 구성요소의 요소 수.
CREATE_TIME	TIMESTAMP		보안 레이블 구성요소가 작성된 시간.
REMARKS	VARCHAR(254)		사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.SECURITYLABELS

각 행은 보안 레이블을 나타냅니다.

표 160. SYSCAT.SECURITYLABELS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
SECLABELNAME	VARCHAR(128)		보안 레이블 이름.
SECLABELID	INTEGER		보안 레이블 ID.
SECPOLICYID	INTEGER		보안 레이블이 속하는 보안 규정의 ID.
SECLABEL	SYSPROC. DB2SECURITYLABEL		보안 레이블의 내부 표현.
CREATE_TIME	TIMESTAMP		보안 레이블이 작성된 시간.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.SECURITYPOLICIES

각 행은 보안 규정을 나타냅니다.

표 161. SYSCAT.SECURITYPOLICIES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
SECPOLICYNAME	VARCHAR(128)		보안 규정 이름.
SECPOLICYID	INTEGER		보안 규정 ID.
NUMSECLABELCOMP	INTEGER		보안 규정에 있는 보안 레이블 구성요소 수.
RWSECLABELREL	CHAR(1)		동일한 권한 부여 ID에 부여된 읽기 및 쓰기 액세스에 대한 보안 레이블 사이의 관계. • S = 사용자에게 부여된 쓰기 액세스에 대한 보안 레이블은 동일한 사용자에게 부여된 읽기 액세스에 대한 보안 레이블의 서브세트임
NOTAUTHWRITESECLABEL	CHAR(1)		사용자에게 INSERT 또는 UPDATE문에 지정된 보안 레이블을 쓸 권한이 없을 때 취할 조치. • O = 겹쳐쓰기 • R = 제한
CREATE_TIME	TIMESTAMP		보안 규정이 작성된 시간.
GROUPAUTHS	CHAR(1)		그룹을 나타내는 권한 부여 ID에 권한 부여된 보안 레이블 및 면제의 권한 부여가 사용 또는 무시될지 여부를 표시합니다. • I = 무시 • U = 사용됨
ROLEAUTHS	CHAR(1)		역할을 나타내는 권한 부여 ID에 권한 부여된 보안 레이블 및 면제의 권한 부여가 사용 또는 무시될지 여부를 표시합니다. • I = 무시 • U = 사용됨
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.SECURITYPOLICYCOMPONENTRULES

각 행은 보안 규정의 보안 레이블 구성요소에 대한 읽기 및 쓰기 액세스를 나타냅니다.

표 162. SYSCAT.SECURITYPOLICYCOMPONENTRULES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
SECPOLICYID	INTEGER		보안 규정 ID.
COMPID	INTEGER		보안 규정의 보안 레이블 구성요소 ID.
ORDINAL	INTEGER		보안 규정에서 나타나는 대로 보안 레이블 구성요소의 위치이며, 1에서 시작합니다.
READACCESSRULENAME	VARCHAR(128)		보안 레이블 구성요소와 연관된 읽기 액세스 규칙 이름.
READACCESSRULETEXT	VARCHAR(512)		보안 레이블 구성요소와 연관된 읽기 액세스 규칙의 텍스트.
WRITEACCESSRULENAME	VARCHAR(128)		보안 레이블 구성요소와 연관된 쓰기 액세스 규칙 이름.
WRITEACCESSRULETEXT	VARCHAR(512)		보안 레이블 구성요소와 연관된 쓰기 액세스 규칙의 텍스트.

SYSCAT.SECURITYPOLICYEXEMPTIONS

각 행은 데이터베이스 권한 부여 ID에 권한 부여되었던 보안 규정 면제를 나타냅니다.

표 163. SYSCAT.SECURITYPOLICYEXEMPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		면제의 권한 준 사용자.
GRANTEE	VARCHAR(128)		면제 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
SECPOLICYID	INTEGER		면제가 권한 부여된 보안 규정에 대한 ID입니다. 보안 규정 이름의 경우, SYSCAT.SECURITYPOLICIES 카탈로그 뷰에서 대응하는 SECPOLICYID 값에 대한 SECPOLICYNAME 컬럼을 선택하십시오.
ACCESSRULENAME	VARCHAR(128)		면제가 권한 부여된 액세스 규칙에 대한 ID입니다.
ACCESSTYPE	CHAR(1)		규칙이 적용되는 액세스 유형. • R = 읽기 액세스 • W = 쓰기 액세스
ORDINAL	INTEGER		규칙이 적용되는 보안 규정에서 보안 레이블 구성요소의 위치.
ACTIONALLOWED	CHAR(1)		규칙이 DB2LBACWRITEARRAY이면 다음과 같습니다. • D = 아래 쓰기 • U = 위 쓰기 그렇지 않으면 공백.
GRANT_TIME	TIMESTAMP		면제가 권한 부여된 시간.

SYSCAT.SEQUENCEAUTH

각 행은 시퀀스에서 하나 이상의 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 164. SYSCAT.SEQUENCEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권의 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
SEQSCHEMA	VARCHAR(128)		시퀀스의 스키마 이름.
SEQNAME	VARCHAR(128)		시퀀스의 규정되지 않은 이름.
ALTERAUTH	CHAR(1)		시퀀스를 할당할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
USAGEAUTH	CHAR(1)		시퀀스를 참조할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.SEQUENCES

각 행은 시퀀스 또는 별명을 나타냅니다.

표 165. SYSCAT.SEQUENCES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
SEQSCHEMA	VARCHAR(128)		시퀀스의 스키마 이름.
SEQNAME	VARCHAR(128)		시퀀스의 규정되지 않은 이름.
DEFINER[1]	VARCHAR(128)		시퀀스 소유자의 권한 부여 ID.
DEFINERTYPE	CHAR(1)		• S = 정의자가 시스템임 • U = 정의자가 개별 사용자임
OWNER	VARCHAR(128)		시퀀스 소유자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
SEQID	INTEGER		시퀀스 또는 별명의 ID.
SEQTYPE	CHAR(1)		시퀀스 유형. • A = 별명 • I = ID 시퀀스 • S = 시퀀스
BASE_SEQSCHEMA	VARCHAR(128)	Y	SEQTYPE이 'A'인 경우, 이 별명이 참조하는 시퀀스 또는 별명의 스키마 이름이 들어 있고, 그 외에는 널(NULL) 값입니다.
BASE_SEQNAME	VARCHAR(128)	Y	SEQTYPE이 'A'인 경우, 이 별명이 참조하는 시퀀스 또는 별명의 규정되지 않은 이름이 들어 있고, 그 외에는 널(NULL) 값입니다.
INCREMENT	DECIMAL(31,0)	Y	증분 값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
START	DECIMAL(31,0)	Y	시퀀스의 시작값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
MAXVALUE	DECIMAL(31,0)	Y	시퀀스의 최대값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
MINVALUE	DECIMAL(31,0)	Y	시퀀스의 최소값입니다. 시퀀스가 별명인 경우 널(NULL) 값입니다.
NEXTCACHEFIRSTVALUE	DECIMAL(31,0)	Y	다음 캐시 블록에서 지정될 수 있는 첫 번째 값입니다. 캐싱이 없는 경우 지정할 수 있는 다음 값입니다.
CYCLE	CHAR(1)		시퀀스가 최대 또는 최소값에 도달한 후 계속 값을 생성할 수 있는지 여부를 표시합니다. • N = 시퀀스가 순환할 수 없음 • Y = 시퀀스가 순환할 수 있음 • 공백 = 시퀀스가 별명임

표 165. SYSCAT.SEQUENCES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
CACHE	INTEGER		보다 빠른 액세스를 위해 메모리에 사전 할당할 시퀀스 값의 수입니다. 0은 시퀀스 값이 사전 할당되지 않음을 표시합니다. 파티션된 데이터베이스에서는 이 값이 각 데이터베이스 파티션에 적용됩니다. 시퀀스가 별명인 경우 -1입니다.
ORDER	CHAR(1)		시퀀스 번호를 요청 순서대로 생성해야 하는지 여부를 표시합니다. • N = 시퀀스 번호가 요청 순서대로 생성될 필요가 없음 • Y = 시퀀스 번호를 요청 순서대로 생성해야 함 • 공백 = 시퀀스가 별명임
DATATYPEID	INTEGER		내장 유형의 경우 내장 유형의 내부 ID입니다. 구별 유형의 경우 구별 유형의 내부 ID입니다. 시퀀스가 별명인 경우 0입니다.
SOURCETYPEID	INTEGER		내장 유형의 경우 또는 시퀀스가 별명인 경우 값이 0입니다. 구별 유형의 경우 구별 유형에 대한 소스 유형인 내장 유형의 내부 ID입니다.
CREATE_TIME	TIMESTAMP		시퀀스가 작성된 시간.
ALTER_TIME	TIMESTAMP		시퀀스가 마지막으로 변경된 시간.
PRECISION	SMALLINT		시퀀스의 데이터 유형 정밀도입니다. 가능한 값은 다음과 같습니다. • 5 = SMALLINT • 10 = INTEGER • 19 = BIGINT DECIMAL의 경우, 이는 지정된 DECIMAL 데이터 유형의 정밀도입니다. 시퀀스가 별명인 경우 0입니다.
ORIGIN	CHAR(1)		시퀀스의 원점. • S = 시스템 생성 시퀀스 • U = 사용자 생성 시퀀스
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.SERVEROPTIONS

각 행은 서버 특정 옵션 값을 나타냅니다.

표 166. SYSCAT.SERVEROPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WRAPNAME	VARCHAR(128)	Y	랩퍼 이름.
SERVERNAME	VARCHAR(128)	Y	서버의 대문자 이름.
SERVERTYPE	VARCHAR (30)	Y	서버 유형.
SERVERVERSION	VARCHAR(18)	Y	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 식별합니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.SERVERS

각 행은 데이터 소스를 나타냅니다.

표 167. SYSCAT.SERVERS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WRAPNAME	VARCHAR(128)		랩퍼 이름.
SERVERNAME	VARCHAR(128)		서버의 대문자 이름.
SERVERTYPE	VARCHAR (30)	Y	서버 유형.
SERVERVERSION	VARCHAR(18)	Y	서버 버전.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.SERVICECLASSES

각 행은 서비스 클래스를 나타냅니다.

표 168. SYSCAT.SERVICECLASSES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
SERVICECLASSNAME	VARCHAR(128)		서비스 클래스 이름.
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	상위 서비스 수퍼 클래스의 서비스 클래스 이름.
SERVICECLASSID	SMALLINT		서비스 클래스 ID.
PARENTID	SMALLINT		이 서비스 클래스에 대한 상위 서비스 클래스의 ID입니다. 이 서비스 클래스가 수퍼 서비스 클래스인 경우 0입니다.
CREATE_TIME	TIMESTAMP		서비스 클래스가 작성된 시간.
ALTER_TIME	TIMESTAMP		서비스 클래스가 마지막으로 변경된 시간.
ENABLED	CHAR(1)		서비스 클래스 상태. • N = 사용 불가능 • Y = 사용 가능
AGENTPRIORITY	SMALLINT		DB2 스레드의 보통 우선순위에 대응하는 서비스 클래스에 있는 에이전트의 스레드 우선순위. • -20 - 20(Linux 및 UNIX) • -6 - 6 (Windows) • -32768 = 설정되지 않음
PREFETCHPRIORITY	CHAR(1)		서비스 클래스에서 에이전트의 프리페치 우선순위. • H = 높음 • L = 낮음 • M = 중간 • 공백 = 설정되지 않음
BUFFERPOOLPRIORITY	CHAR(1)		서비스 클래스에서 에이전트의 버퍼 풀 우선순위 • H = 높음 • L = 낮음 • M = 중간 • 공백 = 설정되지 않음
INBOUNDCORRELATOR	VARCHAR(128)	Y	나중에 사용함.
OUTBOUNDCORRELATOR	VARCHAR(128)	Y	서비스 클래스를 운영 체제 워크로드 관리 프로그램 서비스 클래스와 연관시키는 데 사용되는 문자열.
COLLECTAGGACTDATA	CHAR(1)		적용 가능한 이벤트 모니터가 서비스 클래스에 대해 캡처할 집계 활동 데이터를 지정합니다. • B = 기본 집계 활동 데이터 수집 • E = 확장 집계 활동 데이터 수집 • N = 없음

표 168. SYSCAT.SERVICECLASSES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
COLLECTAGGREQDATA	CHAR(1)		적용 가능한 이벤트 모니터가 서비스 클래스에 대해 캡처할 집계 활동 데이터를 지정합니다. • B = 기본 집계 요청 데이터 수집 • N = 없음
COLLECTACTDATA	CHAR(1)		적용 가능한 이벤트 모니터가 수집할 활동 데이터를 지정합니다. • D = 세부사항을 갖는 활동 데이터 • N = 없음 • S = 세부사항 및 섹션 환경이 있는 활동 데이터 • V = 세부사항 및 값이 있는 활동 데이터 • W = 세부사항이 없는 활동 데이터 • X = 세부사항, 섹션 환경 및 값이 있는 활동 데이터
COLLECTACTPARTITION	CHAR(1)		활동 데이터가 수집되는 위치를 지정합니다. • C = 활동 코디네이터의 데이터베이스 파티션 • D = 모든 데이터베이스 파티션
COLLECTREQMETRICS	CHAR(1)		서비스 수퍼 클래스와 연관된 연결이 제출하는 요청에 대한 모니터링 레벨을 지정합니다. • B = 기본 요청 메트릭 수집 • E = 확장 요청 메트릭 수집 • N = 없음
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.STATEMENTS

각 행은 패키지의 SQL문을 나타냅니다.

표 169. SYSCAT.STATEMENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
PKGSCHEMA	VARCHAR(128)		패키지의 스키마 이름.
PKGNAME	VARCHAR(128)		패키지의 규정되지 않은 이름.
STMTNO	INTEGER		응용프로그램의 소스 모듈에 있는 SQL문의 행 번호.
SECTNO	SMALLINT		SQL문을 포함하는 패키지 섹션 번호.
SEQNO	INTEGER		항상 1입니다.
TEXT	CLOB(2M)		SQL문의 텍스트.
UNIQUE_ID	CHAR(8) FOR BIT DATA		동일한 이름을 갖는 다중 패키지가 존재할 때 특정 패키지에 대한 ID.
VERSION	VARCHAR(64)	Y	패키지의 버전 ID.

SYSCAT.SURROGATEAUTHIDS

각 행은 사용자 또는 PUBLIC에 대한 SETSESSIONUSER 특권이 부여된 사용자 또는 그룹을 나타냅니다.

표 170. SYSCAT.SURROGATEAUTHIDS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		TRUSTEDID에 대리인으로 작용할 능력을 부여한 권한 부여 ID입니다. TRUSTEDID가 트러스트된 컨텍스트 오브젝트를 표시할 때 이 필드는 트러스트된 컨텍스트 오브젝트를 작성 또는 변경한 권한 부여 ID를 나타냅니다.
TRUSTEDID	VARCHAR(128)		대리인으로 작용하도록 트러스트되는 엔티티에 대한 ID.
TRUSTEDIDTYPE	CHAR(1)		• C = 트러스트된 컨텍스트 • G = 그룹 • U = 사용자
SURROGATEAUTHID	VARCHAR(128)		TRUSTEDID가 가정할 수 있는 대리인 권한 부여 ID입니다. 'PUBLIC'은 TRUSTEDID가 모든 권한 부여 ID를 가정할 수 있음을 표시합니다.
SURROGATEAUTHIDTYPE	CHAR(1)		• G = 그룹 • U = 사용자
AUTHENTICATE	CHAR(1)		• N = 인증이 필요 없음 • Y = 권한 부여 ID를 가정할 수 있기 전에 사용자를 인증하기 위해 권한 부여 ID와 함께 인증 토큰이 필요함 • 공백 = TRUSTEDIDTYPE이 'C'가 아님
CONTEXTROLE	VARCHAR(128)	Y	가정된 권한 부여 ID에 지정될 특정 역할이며, 트러스트된 컨텍스트에 대해 정의되는 디폴트 역할(있는 경우)을 대체합니다. TRUSTEDIDTYPE이 'C'가 아닌 경우 널(NULL) 값입니다.
GRANT_TIME	TIMESTAMP		권한 부여된 시간.

SYSCAT.TABAUTH

각 행은 테이블 또는 뷰에서 하나 이상의 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 171. SYSCAT.TABAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
TABSCHEMA	VARCHAR(128)		테이블 또는 뷰의 스키마 이름.
TABNAME	VARCHAR(128)		테이블 또는 뷰의 규정되지 않은 이름.
CONTROLAUTH	CHAR(1)		CONTROL 특권. • N = 보유되지 않음 • Y = 보유되지만 권한 부여 불가능함
ALTERAUTH	CHAR(1)		테이블을 변경할 특권. 이 테이블에 대한 상위 테이블이 기본 키 또는 고유 제한조건을 삭제할 수 있거나, 테이블이 구체화된 쿼리에서 이 테이블 또는 뷰를 참조하는 구체화된 쿼리 테이블이 될 수 있거나, 구체화된 쿼리에서 이 테이블 또는 뷰를 참조하는 테이블이 더 이상 구체화된 쿼리 테이블이 될 수 없습니다. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
DELETEAUTH	CHAR(1)		테이블 또는 갱신 가능 뷰에서 행을 삭제하는 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
INDEXAUTH	CHAR(1)		테이블에서 인덱스를 작성할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

표 171. SYSCAT.TABAUTH 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
INSERTAUTH	CHAR(1)		테이블 또는 갱신 가능한 뷰에 행을 삽입하거나 테이블 또는 뷰에 대해 중요한 유틸리티를 실행할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
REFAUTH	CHAR(1)		테이블을 상위로 참조하는 외부 키를 작성 및 삭제(drop)하는 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
SELECTAUTH	CHAR(1)		테이블 또는 뷰에서 행을 검색하거나, 테이블에 뷰를 작성하거나, 테이블 또는 뷰에 대해 익스포트 유틸리티를 실행할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
UPDATEAUTH	CHAR(1)		테이블 또는 갱신 가능 뷰에 대해 UPDATE문을 실행할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.TABCONST

각 행은 CHECK, UNIQUE, PRIMARY KEY 또는 FOREIGN KEY 유형의 테이블 제한조건을 나타냅니다. 테이블 계층의 경우 각 제한조건은 제한조건이 작성된 계층 구조의 레벨에서만 기록됩니다.

표 172. SYSCAT.TABCONST 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
CONSTNAME	VARCHAR(128)		제한조건 이름.
TABSCHEMA	VARCHAR(128)		이 제한조건이 적용되는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		이 제한조건이 적용되는 테이블의 규정되지 않은 이름.
OWNER	VARCHAR(128)		제한조건 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
TYPE	CHAR(1)		제한조건 유형을 나타냅니다. • F = 외부 키 • I = 함수 종속성 • K = 점검 • P = 기본 키 • U = 고유
ENFORCED	CHAR(1)		• N = 제한조건을 강제 실행하지 않음 • Y = 제한조건 강제 실행
CHECKEXISTINGDATA	CHAR(1)		• D = 모든 기존 데이터 점검 지연 • I = 기존 데이터 즉시 점검 • N = 기존 데이터를 점검하지 않음
ENABLEQUERYOPT	CHAR(1)		• N = 쿼리 최적화 사용 안함 • Y = 쿼리 최적화 사용
DEFINER[1]	VARCHAR(128)		제한조건 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.TABDEP

각 행은 일부 다른 오브젝트에 대한 뷰 또는 구체화된 쿼리 테이블의 종속성을 나타냅니다. 뷰 또는 구체화된 쿼리 테이블은 이름이 BNAME인 BTYPE 유형의 오브젝트에 따라 달라지므로, 오브젝트에 대한 변경사항은 뷰 또는 구체화된 쿼리 테이블에 영향을 줍니다. 또한 뷰에 대한 특권이 기초 테이블 및 뷰에 대한 특권에 종속되는 방법을 인코드합니다.

표 173. SYSCAT.TABDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블의 규정되지 않은 이름.
DTYPE	CHAR(1)		종속되는 오브젝트 유형. • S = 구체화된 쿼리 테이블 • T = 테이블(스테이징 전용) • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰
OWNER	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블 작성자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• U = 소유자가 개별 사용자임
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • F = 루틴 • I = 기본 테이블에 대한 종속성을 기록하는 경우 인덱스 • G = 전역 임시 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • R = 사용자 정의 구조화된 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • Z = XSR 오브젝트 • u = 모듈 별명 • v = 전역 변수

표 173. SYSCAT.TABDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BSCHEMA	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블이 종속되는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.
BNAME	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블이 종속되는 오브젝트의 규정되지 않은 이름.
BMODULEID	INTEGER	Y	뷰 또는 구체화된 쿼리 테이블이 종속되는 오브젝트 모듈의 ID.
TABAUTH	SMALLINT	Y	BTYPE이 'N', 'O', 'S', 'T', 'U', 'V' 또는 'W'인 경우, 이 뷰 또는 구체화된 쿼리 테이블이 종속되는 기초 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.
VARAUTH	SMALLINT	Y	BTYPE이 'v'인 경우, 이 뷰 또는 구체화된 쿼리 테이블이 종속되는 기초 전역 변수에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.
DEFINER[1]	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블 작성자의 권한 부여 ID.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.TABDETACHEDDEP

각 행은 접속 해제된 종속 테이블과 접속 해제된 테이블 사이의 접속 해제된 종속성을 나타냅니다.

표 174. SYSCAT.TABDETACHEDDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		접속 해제된 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		접속 해제된 테이블의 규정되지 않은 이름.
DEPTABSCHEMA	VARCHAR(128)		접속 해제된 종속 테이블의 스키마 이름.
DEPTABNAME	VARCHAR(128)		접속 해제된 종속 테이블의 규정되지 않은 이름.

SYSCAT.TABLES

각 행은 테이블, 뷰, 별명 또는 별칭을 나타냅니다. 각 테이블 또는 뷰 계층 구조에는 계층 구조를 구현하는 계층 구조 테이블 또는 계층 구조 뷰를 나타내는 추가 행이 하나 더 있습니다. 카탈로그 테이블 및 뷰가 포함됩니다.

표 175. SYSCAT.TABLES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		오브젝트의 스키마 이름.
TABNAME	VARCHAR(128)		오브젝트의 규정되지 않은 이름.
OWNER	VARCHAR(128)		테이블, 뷰, 별명 또는 별칭 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
TYPE	CHAR(1)		오브젝트의 유형. • A = 별명 • G = 작성된 임시 테이블 • H = 계층 구조 테이블 • L = 접속 해제된 테이블 • N = 별칭 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰
STATUS	CHAR(1)		오브젝트 상태. • C = 무결성 설정 보류 • N = 보통 • X = 작동 불능
BASE_TABSCHEMA	VARCHAR(128)	Y	TYPE = 'A'인 경우 이 별명이 참조하는 테이블, 뷰, 별명 또는 별칭의 스키마 이름을 포함합니다. 그 외에는 널(NULL) 값입니다.
BASE_TABNAME	VARCHAR(128)	Y	TYPE = 'A'인 경우 이 별명이 참조하는 테이블, 뷰, 별명 또는 별칭의 규정되지 않은 이름을 포함합니다. 그 외에는 널(NULL) 값입니다.
ROWTYPESCHEMA	VARCHAR(128)	Y	적용할 수 있는 경우 이 테이블의 행 유형의 스키마 이름이고, 그 외에는 널(NULL) 값입니다.
ROWTYPENAME	VARCHAR(128)	Y	적용할 수 있는 경우 이 테이블의 행 유형의 규정되지 않은 이름이고, 그 외에는 널(NULL) 값입니다.
CREATE_TIME	TIMESTAMP		오브젝트가 작성된 시간.
ALTER_TIME	TIMESTAMP		오브젝트가 마지막으로 변경된 시간.
INVALIDATE_TIME	TIMESTAMP		오브젝트가 마지막으로 무효화된 시간.
STATS_TIME	TIMESTAMP	Y	이 오브젝트에 대해 기록된 통계가 마지막으로 변경된 시간입니다. 통계가 수집되지 않은 경우 널(NULL) 값입니다.
COLCOUNT	SMALLINT		상속된 컬럼(있는 경우)을 포함한 컬럼 수.

표 175. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
TABLEID	SMALLINT		내부 논리적 오브젝트 ID.
TBSPACEID	SMALLINT		이 오브젝트에 대한 기본 테이블 스페이스의 내부 논리적 ID.
CARD	BIGINT		테이블에 있는 총 행 수이며, 통계가 수집되지 않는 경우 -1입니다.
NPAGES	BIGINT		테이블의 행이 존재하는 페이지의 총수입니다. 뷰 또는 별명의 경우 또는 통계가 수집되지 않는 경우 -1이고, 서브테이블 또는 계층 구조 테이블의 경우 -2입니다.
FPAGES	BIGINT		페이지의 총수입니다. 뷰 또는 별명의 경우 또는 통계가 수집되지 않는 경우 -1이고, 서브테이블 또는 계층 구조 테이블의 경우 -2입니다.
OVERFLOW	BIGINT		테이블에 있는 오버플로우 레코드의 총수입니다. 뷰 또는 별명의 경우 또는 통계가 수집되지 않는 경우 -1이고, 서브테이블 또는 계층 구조 테이블의 경우 -2입니다.
TBSPACE	VARCHAR(128)	Y	테이블에 대한 기본 테이블 스페이스 이름입니다. 다른 테이블 스페이스를 지정하지 않으면 테이블의 모든 부분이 이 테이블 스페이스에 저장됩니다. 별명, 뷰 및 파티션된 테이블의 경우 널(NULL) 값입니다.
INDEX_TBSPACE	VARCHAR(128)	Y	이 테이블에 대해 작성된 모든 인덱스를 보유하는 테이블 스페이스의 이름입니다. 별명, 뷰 및 파티션된 테이블의 경우 또는 INDEX IN절이 생략되었거나 CREATE TABLE문의 IN절에서와 같은 값으로 지정된 경우 널(NULL) 값입니다.
LONG_TBSPACE	VARCHAR(128)	Y	이 테이블에 대한 모든 Long 데이터(LONG 또는 LOB 컬럼 유형)를 보유하는 테이블 스페이스 이름입니다. 별명, 뷰 및 파티션된 테이블의 경우 또는 LONG IN절이 생략되었거나 CREATE TABLE문의 IN절에서와 같은 값으로 지정된 경우 널(NULL) 값입니다.
PARENTS	SMALLINT	Y	이 오브젝트에 대한 상위 테이블 수입니다. 즉, 이 오브젝트가 종속되는 참조 제한조건의 수입니다.
CHILDREN	SMALLINT	Y	이 오브젝트에 대한 종속 테이블 수입니다. 즉, 이 오브젝트가 상위인 참조 제한조건의 수입니다.
SELFREFS	SMALLINT	Y	이 오브젝트에 대해 자체 참조하는 참조 제한조건의 수입니다. 즉, 이 오브젝트가 상위이면서 종속인 참조 제한조건의 수입니다.
KEYCOLUMNS	SMALLINT	Y	기본 키의 컬럼 수.
KEYINDEXID	SMALLINT	Y	기본 키 인덱스의 인덱스 ID입니다. 기본 키가 없는 경우 0 또는 널(NULL) 값입니다.
KEYUNIQUE	SMALLINT		이 오브젝트에 정의되는 고유 키 제한조건(기본 키 제한조건 제외)의 수.
CHECKCOUNT	SMALLINT		이 오브젝트에 정의된 점검 제한조건의 수.
DATACAPTURE	CHAR(1)		• L = 테이블이 LONG VARCHAR 및 LONG VARGRAPHIC 컬럼의 복제를 포함한 데이터 복제에 참여함 • N = 테이블이 데이터 복제에 참여하지 않음 • Y = 테이블이 LONG VARCHAR 및 LONG VARGRAPHIC 컬럼의 복제를 제외한 데이터 복제에 참여함

표 175. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
CONST_CHECKED	CHAR(32)		• 바이트 1은 외부 키 제한조건을 나타냅니다. • 바이트 2는 점검 제한조건을 나타냅니다. • 바이트 5는 구체화된 쿼리 테이블을 나타냅니다. • 바이트 6은 생성된 컬럼을 나타냅니다. • 바이트 7은 스테이징 테이블을 나타냅니다. • 바이트 8은 데이터 파티셔닝 제한조건을 나타냅니다. • 나머지 바이트는 나중에 사용하기 위해 예약되어 있습니다. 가능한 값은 다음과 같습니다. • F = 바이트 5에서 구체화된 쿼리 테이블은 증분하여 새로 고칠 수 없습니다. 바이트 7에서 스테이징 테이블의 컨텐츠는 불완전하며 연관된 구체화된 쿼리 테이블의 증분 새로 고침에 사용할 수 없습니다. • N = 점검되지 않음 • U = 사용자가 점검함 • W = 테이블이 무결성 설정 보류 상태이면 'U' 상태 • Y = 시스템 점검
PMAP_ID	SMALLINT	Y	현재 이 테이블이 사용 중인 분산 맵의 ID(별명 또는 뷰의 경우 널(NULL) 값).
PARTITION_MODE	CHAR(1)		데이터가 파티션된 데이터베이스 시스템의 데이터베이스 파티션 사이에서 분배되는 방법을 나타냅니다. • H = 해싱 • R = 데이터베이스 파티션 사이에 복제됨 • 공백 = 데이터베이스 파티셔닝 없음
LOG_ATTRIBUTE	CHAR(1)		• 항상 0입니다. 이 컬럼은 더 이상 사용되지 않습니다.
PCTFREE	SMALLINT		나중에 삽입하기 위해 예약되는 각 페이지의 백분율.
APPEND_MODE	CHAR(1)		행이 페이지에 삽입되는 방법을 제어합니다. • N = 사용 가능한 경우 새 행이 기존 스페이스에 삽입됨 • Y = 새 행이 데이터 끝에 추가됨
REFRESH	CHAR(1)		새로 고침 모드. • D = 지연됨 • I = 즉시 • O = 한 번만 • 공백 = 구체화된 쿼리 테이블이 아님
REFRESH_TIME	TIMESTAMP	Y	REFRESH = 'D' 또는 'O'의 경우 데이터가 마지막으로 새로 고쳐진(REFRESH TABLE문) 시간이고, 그 외에는 널(NULL) 값입니다.

표 175. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
LOCKSIZE	CHAR(1)		DML(Data Manipulation Language) 명령문이 액세스하는 테이블에 대한 선호되는 잠금 단위를 표시합니다. 테이블에만 적용됩니다. 가능한 값은 다음과 같습니다. • I = 블록 삽입 • R = 행 • T = 테이블 • 공백 = 적용할 수 없음
VOLATILE	CHAR(1)		• C = 테이블의 카디널리티(cardinality)가 일시적임 • 공백 = 적용할 수 없음
ROW_FORMAT	CHAR(1)		사용되지 않음
PROPERTY	VARCHAR(32)		테이블의 등록 정보입니다. 단일 공백은 테이블에 등록 정보가 없음을 표시합니다. 다음은 문자열 안에서의 위치, 값 및 의미입니다. • 1, Y = 사용자가 유지하는 구체화된 쿼리 테이블 • 2, Y = 스테이징 테이블 • 3, Y = 즉시 전파 • 11, Y = 캐시되지 않는 별칭
STATISTICS_PROFILE	CLOB(10M)	Y	오브젝트에 대한 통계 프로파일을 등록하는 데 사용되는 RUNSTATS 명령.
COMPRESSION	CHAR(1)		• B = 값과 행 압축이 모두 활성화됨 • N = 압축이 활성화되지 않음. 압축을 지원하지 않는 행 형식이 사용됨 • R = 사용 허가된 경우 행 압축이 활성화됨. 압축을 지원하는 행 형식을 사용할 수 있음 • V = 값 압축이 활성화됨. 압축을 지원하는 행 형식이 사용됨 • 공백 = 적용할 수 없음
ACCESS_MODE	CHAR(1)		오브젝트의 액세스 제한 상태입니다. 이러한 상태는 무결성 설정 보류 상태에 있는 오브젝트나 SET INTEGRITY문으로 처리된 오브젝트에만 적용됩니다. 가능한 값은 다음과 같습니다. • D = 데이터 이동 없음 • F = 전체 액세스 • N = 권한 없음 • R = 읽기 전용 액세스
CLUSTERED	CHAR(1)	Y	• Y = 테이블이 다차원으로 클러스터됨(한 차원만으로도 다차원으로 클러스터됨) • 널(NULL) 값 = 테이블이 다차원으로 클러스터되지 않음
ACTIVE_BLOCKS	BIGINT		테이블의 활성 블록의 총수 또는 -1입니다. 다차원적으로 클러스터된(MDC) 테이블에만 적용됩니다.

표 175. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
DROPRULE	CHAR(1)		• N = 규칙 없음 • R = 제한 규칙이 삭제(drop)에 적용됨
MAXFREESPACESEARCH	SMALLINT		나중에 사용하기 위해 예약됨
AVGCOMPRESSEDROWSIZE	SMALLINT		이 테이블에 있는 압축된 행의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
AVGROWCOMPRESSIONRATIO	REAL		테이블의 압축된 행의 경우 행별로 평균 압축비입니다. 즉, 평균 압축되지 않은 행 길이를 평균 압축 행 길이로 나눈 값입니다. 통계가 수집되지 않는 경우 -1입니다.
AVGROWSIZE	SMALLINT		이 테이블에 있는 압축 및 압축되지 않은 행 모두의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
PCTROWSCOMPRESSED	REAL		테이블에 있는 전체 행 수의 백분율로 표현되는 압축된 행입니다. 통계가 수집되지 않는 경우 -1입니다.
LOGINDEXBUILD	VARCHAR(3)	Y	테이블에 대한 인덱스 재구성, 재작성 또는 작성 조작 중에 수행되는 로깅 레벨. • OFF = 테이블에 대한 인덱스 빌드 조작이 최소한으로 로그됨 • ON = 테이블에 대한 인덱스 빌드 조작 전체가 로그됨 • 널(NULL) 값 = *logindexbuild* 데이터베이스 구성 매개변수의 값은 인덱스 빌드 조작 전체를 로그할지 여부를 판별하는 데 사용됩니다.
CODEPAGE	SMALLINT		오브젝트의 코드 페이지입니다. 모든 문자 컬럼, 트리거, 점검 제한조건 및 표현식 생성 컬럼에 사용되는 디폴트 코드 페이지입니다.
COLLATIONSCHEMA	VARCHAR(128)		테이블 조합의 스키마 이름.
COLLATIONNAME	VARCHAR(128)		테이블 조합의 규정되지 않은 이름.
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		테이블에서 ORDER BY절에 대한 조합의 스키마 이름.
COLLATIONNAME_ORDERBY	VARCHAR(128)		테이블의 ORDER BY절에 대한 조합의 규정되지 않은 이름.
ENCODING_SCHEME	CHAR(1)		• A = CCSID ASCII가 지정됨 • U = CCSID UNICODE가 지정됨 • 공백 = CCSID절이 지정되지 않음
PCTPAGESSAVED	SMALLINT		행 압축의 결과로 테이블에 저장되는 페이지의 대략적인 백분율입니다. 이 값에는 테이블의 각 사용자 데이터에 대한 오버헤드 바이트가 포함되지만 사전 오버헤드가 이용하는 스페이스는 포함되지 않습니다. 통계가 수집되지 않는 경우 -1입니다.
LAST_REGEN_TIME	TIMESTAMP	Y	테이블에 대한 모든 뷰 또는 점검 제한조건이 마지막으로 다시 만들어진 시간.
SECPOLICYID	INTEGER		테이블을 보호하는 보안 규정의 ID입니다. 비보호 테이블의 경우 0입니다.
PROTECTIONGRANULARITY	CHAR(1)		• B = 컬럼 및 행 레벨 세분화도 둘 다 • C = 컬럼 레벨 세분화도 • R = 행 레벨 세분화도 • 공백 = 비보호 테이블
AUDITPOLICYID	INTEGER	Y	감사 규정 ID.

표 175. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
AUDITPOLICYNAME	VARCHAR(128)	Y	감사 규정 이름.
DEFINER[1]	VARCHAR(128)		테이블, 뷰, 별명 또는 별칭 소유자의 권한 부여 ID
ONCOMMIT	CHAR(1)		COMMIT 조작 수행 시 작성된 임시 테이블에 대해 취하는 조치를 지정합니다. • D = 행 삭제 • P = 행 보존 • 공백 = 테이블이 작성된 임시 테이블이 아님
LOGGED	CHAR(1)		작성된 임시 테이블이 로그되는지 여부를 지정합니다. • N = 로그되지 않음 • Y = 로그됨 • 공백 = 테이블이 작성된 임시 테이블이 아님
ONROLLBACK	CHAR(1)		ROLLBACK 조작 수행 시 작성된 임시 테이블에 대해 취하는 조치를 지정합니다. • D = 행 삭제 • P = 행 보존 • 공백 = 테이블이 작성된 임시 테이블이 아님
LASTUSED	DATE		나중에 사용하기 위해 예약됨
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.TABLESPACES

각 행은 테이블 스페이스를 나타냅니다.

표 176. SYSCAT.TABLESPACES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TBSPACE	VARCHAR(128)		테이블 스페이스의 이름.
OWNER	VARCHAR(128)		테이블 스페이스 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
CREATE_TIME	TIMESTAMP		테이블 스페이스가 작성된 시간.
TBSPACEID	INTEGER		테이블 스페이스 ID.
TBSPACETYPE	CHAR(1)		테이블 스페이스의 유형. • D = 데이터베이스 관리 스페이스 • S = 시스템 관리 스페이스
DATATYPE	CHAR(1)		이 테이블 스페이스에 저장할 수 있는 데이터 유형. • A = 모든 유형의 영구 데이터. 일반 테이블 스페이스 • L = 모든 유형의 영구 데이터. 큰 테이블 스페이스 • T = 시스템 임시 테이블만 • U = 작성된 임시 테이블 또는 선언된 임시 테이블만
EXTENTSIZE	INTEGER		PAGESIZE 크기의 페이지 수 단위로 각 Extent의 크기입니다. 이 많은 페이지가 다음 컨테이너로 전환하기 전에 테이블 스페이스의 한 컨테이너에 기록됩니다.
PREFETCHSIZE	INTEGER		프리페치가 수행될 때 읽을 PAGESIZE 크기의 페이지 수입니다. AUTOMATIC일 때 -1입니다.
OVERHEAD	DOUBLE		밀리초 단위로 제어기 오버헤드 및 디스크 탐색 및 대기 시간(이 테이블 스페이스의 컨테이너에 대한 평균).
TRANSFERRATE	DOUBLE		PAGESIZE 크기의 한 페이지를 버퍼로 읽는 시간(이 테이블 스페이스의 컨테이너의 경우 평균).
WRITEOVERHEAD	DOUBLE	Y	나중에 사용하기 위해 예약됨
WRITETRANSFERRATE	DOUBLE	Y	PAGESIZE 크기의 한 페이지를 버퍼에서 테이블 스페이스로 쓰는 시간(이 테이블 스페이스의 컨테이너의 경우 평균). 널(NULL) 값은 TRANSFERRATE와 동일한 값이 사용될 것임을 의미합니다.
PAGESIZE	INTEGER		이 테이블 스페이스에 있는 페이지의 크기(바이트).
DBPGNAME	VARCHAR(128)		이 테이블 스페이스와 연관되는 데이터베이스 파티션 그룹의 이름.

표 176. SYSCAT.TABLESPACES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BUFFERPOOLID	INTEGER		이 테이블 스페이스가 사용하는 버퍼 풀의 ID(1은 디폴트 버퍼 풀을 표시함).
DROP_RECOVERY	CHAR(1)		테이블 삭제 조작 후 이 테이블 스페이스의 테이블을 복구할 수 있는지 여부를 표시합니다. • N = 테이블 복구 불가능 • Y = 테이블 복구 가능
NGNAME[1]	VARCHAR(128)		이 테이블 스페이스와 연관되는 데이터베이스 파티션 그룹의 이름.
DEFINER[2]	VARCHAR(128)		테이블 스페이스 소유자의 권한 부여 ID
DATAPRIORITY	CHAR(1)		나중에 사용하기 위해 예약됩니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 NGNAME 컬럼이 포함됩니다. DBPGNAME을 참조하십시오.
2. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.TABOPTIONS

각 행은 리모트 테이블과 연관되는 옵션을 나타냅니다.

표 177. SYSCAT.TABOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		테이블, 뷰, 별명 또는 별칭의 스키마 이름.
TABNAME	VARCHAR(128)		테이블, 뷰, 별명 또는 별칭의 규정되지 않은 이름.
OPTION	VARCHAR(128)		테이블 옵션의 이름.
SETTING	CLOB(32K)		테이블 옵션 값.

SYSCAT.TBSPACEAUTH

각 행은 데이터베이스의 특정 테이블 스페이스에 대한 USE 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 178. SYSCAT.TBSPACEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
TBSPACE	VARCHAR(128)		테이블 스페이스의 이름.
USEAUTH	CHAR(1)		테이블 스페이스 내에서 테이블을 작성할 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.THRESHOLDS

각 행은 임계값을 나타냅니다.

표 179. SYSCAT.THRESHOLDS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
THRESHOLDNAME	VARCHAR(128)		임계값 이름.
THRESHOLDID	INTEGER		임계값 ID.
ORIGIN	CHAR(1)		임계값의 원점. • U = 임계값이 사용자에 의해 작성되었음 • W = 임계값이 작업 조치 세트를 통해 작성되었음
THRESHOLDCLASS	CHAR(1)		임계값의 분류. • A = 집계 임계값 • C = 활동 임계값
THRESHOLDPREDICATE	VARCHAR(15)		임계값의 유형. 가능한 값은 다음과 같습니다. • AGGTEMPSPACE • CONCDBC • CONCWCN • CONCWOC • CONNIDLETIME • CPUTIME • CPUTIMEINSC • DBCONN • ESTSQLCOST • ROWSREAD • ROWSREADINSC • ROWSRET • SCCONN • TEMPSPACE • TOTALTIME
THRESHOLDPREDICATEID	SMALLINT		임계값 술어 ID.
DOMAIN	CHAR(2)		임계값의 도메인. • DB = 데이터베이스 • SB = 서비스 서브클래스 • SP = 서비스 수퍼 클래스 • WA = 작업 조치 세트 • WD = 워크로드 정의

표 179. SYSCAT.THRESHOLDS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
DOMAINID	INTEGER		임계값이 연관되는 오브젝트의 ID입니다. 서비스 클래스, 작업 조치 또는 워크로드 고유 ID일 수 있습니다. 데이터베이스 임계값인 경우 이 값은 0입니다.
ENFORCEMENT	CHAR(1)		임계값에 대한 시행 범위. • D = 데이터베이스 • P = 데이터베이스 파티션 • W = 워크로드 어커런스
QUEUING	CHAR(1)		• N = 임계값이 큐에 들어가지 않음 • Y = 임계값이 큐에 들어감
MAXVALUE	BIGINT		임계값으로 지정되는 상한.
QUEUESIZE	INTEGER		QUEUEING이 'Y'인 경우 큐의 크기입니다. 그 외에는 -1입니다.
COLLECTACTDATA	CHAR(1)		적용 가능한 이벤트 모니터가 수집할 활동 데이터를 지정합니다. • D = 세부사항을 갖는 활동 데이터 • N = 없음 • S = 세부사항 및 섹션 환경이 있는 활동 데이터 • V = 세부사항 및 값이 있는 활동 데이터 • W = 세부사항이 없는 활동 데이터 • X = 세부사항, 섹션 환경 및 값이 있는 활동 데이터
COLLECTACTPARTITION	CHAR(1)		활동 데이터가 수집되는 위치를 지정합니다. • C = 활동 코디네이터의 데이터베이스 파티션 • D = 모든 데이터베이스 파티션
EXECUTION	CHAR(1)		임계값이 초과된 후 실행이 계속되는지, 중지되는지 또는 다른 서비스 서브클래스로 다시 맵핑되는지 여부를 표시합니다. • C = 실행 계속 • R = 실행 재맵핑 • S = 실행 중지
REMAPSCID	SMALLINT		REMAP ACTIVITY 조치의 목표 서비스 서브클래스 ID.
VIOLATIONRECORDLOGGED	CHAR(1)		임계값 위반 시 레코드가 이벤트 모니터에 기록되는지 여부를 표시합니다. • N = 아니오 • Y = 예

표 179. SYSCAT.THRESHOLDS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
CHECKINTERVAL	INTEGER		THRESHOLDPREDICATE가 다음과 같은 경우 임계값 조건을 점검하는 간격(초)입니다. • 'CPUTIME' • 'CPUTIMEINSC' • 'ROWSREAD' • 'ROWSREADINSC' 그 외에는 -1입니다.
ENABLED	CHAR(1)		• N = 이 임계값을 사용 안함 • Y = 이 임계값이 사용 가능함
CREATE_TIME	TIMESTAMP		임계값이 작성된 시간.
ALTER_TIME	TIMESTAMP		임계값이 마지막으로 변경된 시간.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.TRANSFORMS

각 행은 사용자 정의 유형과 기본 SQL 유형 사이의 변환 또는 역방향 변환을 처리하는 함수를 나타냅니다.

표 180. SYSCAT.TRANSFORMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPEID	SMALLINT		데이터 유형 ID.
TYPESCHEMA	VARCHAR(128)		TYPEMODULEID가 널(Null)인 경우 데이터 유형의 스키마 이름이고, 그렇지 않으면 데이터 유형이 속하는 모듈의 스키마 이름입니다.
TYPENAME	VARCHAR(128)		데이터 유형의 규정되지 않은 이름.
GROUPNAME	VARCHAR(128)		변환 그룹 이름
FUNCID	INTEGER		루틴의 ID.
FUNCSCHEMA	VARCHAR(128)		ROUTINEMODULEID가 널(Null)인 경우 루틴의 스키마 이름이고, 그렇지 않으면 루틴이 속하는 모듈의 스키마 이름입니다.
FUNCNAME	VARCHAR(128)		루틴의 규정되지 않은 이름.
SPECIFICNAME	VARCHAR(128)		루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
TRANSFORMTYPE	VARCHAR(8)		• 'FROM SQL' = 변환 함수가 SQL로부터 구조화된 유형을 변환함 • 'TO SQL' = 변환 함수가 구조화된 유형을 SQL로 변환함
FORMAT	CHAR(1)		FROM SQL 변환에 의해 생성되는 형식. • S = 구조화된 데이터 유형 • U = 사용자 정의
MAXLENGTH	INTEGER	Y	FROM SQL 변환 결과의 최대 길이(바이트)이며, TO SQL 변환의 경우 널(NULL) 값입니다.
ORIGIN	CHAR(1)		이 변환 그룹의 소스. • O = 원래 변환 그룹(내장 또는 시스템 정의) • R = 재정의한 변환 그룹(내장 그룹만 재정의할 수 있음)
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.TRIGDEP

각 행은 일부 다른 오브젝트에 대한 트리거의 종속성을 나타냅니다. 트리거는 이름 BNAME의 유형 BTYPE의 오브젝트에 따라 달라지므로, 오브젝트에 대한 변경사항이 트리거에 영향을 줍니다.

표 181. SYSCAT.TRIGDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TRIGSCHEMA	VARCHAR(128)		트리거 스키마 이름.
TRIGNAME	VARCHAR(128)		트리거의 규정되지 않은 이름.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • B = 트리거 • F = 루틴 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • K = 패키지 • L = 접속 해제된 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • Q = 시퀀스 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • X = 인덱스 확장 • Z = XSR 오브젝트 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.

표 181. SYSCAT.TRIGDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름입니다. 루틴(BTYPE = 'F')의 경우, 이것은 특정 이름입니다.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 트리거에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면 널(NULL) 값입니다.

SYSCAT.TRIGGERS

각 행은 트리거를 나타냅니다. 테이블 계층의 경우 각 트리거는 트리거가 작성된 계층 구조의 레벨에서만 기록됩니다.

표 182. SYSCAT.TRIGGERS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TRIGSCHEMA	VARCHAR(128)		트리거 스키마 이름.
TRIGNAME	VARCHAR(128)		트리거의 규정되지 않은 이름.
OWNER	VARCHAR(128)		트리거 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
TABSCHEMA	VARCHAR(128)		이 트리거가 적용되는 테이블 또는 뷰의 스키마 이름.
TABNAME	VARCHAR(128)		이 트리거가 적용되는 테이블 또는 뷰의 규정되지 않은 이름.
TRIGTIME	CHAR(1)		트리거를 시작한 이벤트에 상대적으로 트리거된 동작이 기본 테이블에 적용되는 시간. • A = 트리거가 이벤트 후에 적용됨 • B = 트리거가 이벤트 전에 적용됨 • I = 트리거가 이벤트 대신 적용됨
TRIGEVENT	CHAR(1)		트리거를 시작하는 이벤트. • D = 삭제 조작 • I = 삽입 조작 • U = 갱신 조작
GRANULARITY	CHAR(1)		트리거는 다음에 대해 한 번만 실행됩니다. • R = 행 • S = 명령문
VALID	CHAR(1)		• N = 트리거가 유효하지 않음 • X = 트리거가 작동 불능이며 다시 작성해야 함 • Y = 트리거가 유효함
CREATE_TIME	TIMESTAMP		트리거가 정의된 시간입니다. 함수 및 유형 해결에 사용됩니다.
QUALIFIER	VARCHAR(128)		오브젝트 정의 시 디폴트 스키마의 값입니다. 규정되지 않은 참조를 완료하는 데 사용됩니다.
FUNC_PATH	CLOB(2K)		트리거가 정의된 시간의 SQL 경로.
TEXT	CLOB(2M)		입력된 그대로 CREATE TRIGGER문의 전체 텍스트.
LAST_REGEN_TIME	TIMESTAMP		트리거의 팩형 디스크립터가 마지막으로 다시 만들어진 시간.
COLLATIONSCHEMA	VARCHAR(128)		트리거 조합의 스키마 이름.
COLLATIONNAME	VARCHAR(128)		트리거 조합의 규정되지 않은 이름.
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		트리거의 ORDER BY절에 대한 조합의 스키마 이름.
COLLATIONNAME_ORDERBY	VARCHAR(128)		트리거의 ORDER BY절에 대한 조합의 규정되지 않은 이름.
DEFINER[1]	VARCHAR(128)		트리거 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

표 182. SYSCAT.TRIGGERS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.TYPEMAPPINGS

각 행은 로컬로 정의된 데이터 유형과 데이터 소스 데이터 유형 사이의 데이터 유형 맵핑을 나타냅니다. 두 가지 맵핑 유형(맵핑 방향)이 있습니다.

- 포워드 유형 맵핑은 데이터 소스 데이터 유형을 로컬로 정의된 데이터 유형에 맵핑합니다.
- 역방향 유형 맵핑은 로컬로 정의된 데이터 유형을 데이터 소스 데이터 유형에 맵핑합니다.

표 183. SYSCAT.TYPEMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TYPE_MAPPING	VARCHAR(18)		유형 맵핑의 이름(시스템이 생성했을 수 있음).
MAPPINGDIRECTION	CHAR(1)		이 유형 맵핑이 포워드 또는 역방향 유형 맵핑인지를 표시합니다. • F = 포워드 유형 맵핑 • R = 역방향 유형 맵핑
TYPESCHEMA	VARCHAR(128)	Y	데이터 유형 맵핑의 로컬 유형의 스키마 이름입니다. 내장 유형의 경우 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)		데이터 유형 맵핑의 로컬 유형의 규정되지 않은 이름.
TYPEID	SMALLINT		데이터 유형 ID.
SOURCETYPEID	SMALLINT		소스 유형 ID.
OWNER	VARCHAR(128)		유형 맵핑 소유자의 권한 부여 ID. 'SYSIBM'은 내장 유형 맵핑을 표시합니다.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
LENGTH	INTEGER	Y	이 맵핑의 로컬 데이터 유형의 최대 길이 또는 정밀도입니다. 널(NULL) 값인 경우 시스템이 최대 길이 또는 정밀도를 판별합니다. 문자 유형의 경우 최대 바이트 수를 나타냅니다.
SCALE	SMALLINT	Y	이 맵핑의 로컬 10진수 값에서 분수 부분의 최대 자릿수 또는 이 맵핑의 로컬 TIMESTAMP 값에서 분수 초의 최대 자릿수. 널(NULL) 값인 경우 시스템이 최대 수를 판별합니다.
LOWER_LEN	INTEGER	Y	이 맵핑의 로컬 데이터 유형의 최소 길이 또는 정밀도입니다. 널(NULL) 값인 경우 시스템이 최소 길이 또는 정밀도를 판별합니다. 문자 유형의 경우 최소 바이트 수를 나타냅니다.
UPPER_LEN	INTEGER	Y	이 맵핑의 로컬 데이터 유형의 최대 길이 또는 정밀도입니다. 널(NULL) 값인 경우 시스템이 최대 길이 또는 정밀도를 판별합니다. 문자 유형의 경우 최대 바이트 수를 나타냅니다.

표 183. SYSCAT.TYPEMAPPINGS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
LOWER_SCALE	SMALLINT	Y	이 맵핑의 로컬 10진수 값에서 분수 부분의 최소 자릿수 또는 이 맵핑의 로컬 TIMESTAMP 값에서 분수 초의 최소 자릿수. 널(NULL) 값인 경우 시스템이 최소 수를 판별합니다.
UPPER_SCALE	SMALLINT	Y	이 맵핑의 로컬 10진수 값에서 분수 부분의 최대 자릿수 또는 이 맵핑의 로컬 TIMESTAMP 값에서 분수 초의 최대 자릿수. 널(NULL) 값인 경우 시스템이 최대 수를 판별합니다.
S_OPR_P	CHAR(2)	Y	이 맵핑의 로컬 10진수 값에서 스케일 및 정밀도 사이의 관계입니다. 기본 비교 연산자(=, <, >, <=, >=, <>)를 사용할 수 있습니다. 널(NULL) 값은 특정 관계가 필요하지 않음을 나타냅니다.
BIT_DATA	CHAR(1)	Y	이 문자 유형이 비트 데이터용인지 여부를 표시합니다. 가능한 값은 다음과 같습니다. • N = 이 유형이 비트 데이터용이 아님 • Y = 이 유형이 비트 데이터용임 • 널(NULL) 값 = 문자 데이터 유형이 아니거나 시스템이 비트 데이터 속성을 판별함
WRAPNAME	VARCHAR(128)	Y	이 맵핑이 적용되는 데이터 액세스 프로토콜(랩퍼).
SERVERNAME	VARCHAR(128)	Y	서버의 대문자 이름.
SERVERTYPE	VARCHAR (30)	Y	서버 유형.
SERVERVERSION	VARCHAR(18)	Y	서버 버전.
REMOTE_TYPESCHEMA	VARCHAR(128)	Y	데이터 소스 데이터 유형의 스키마 이름.
REMOTE_TYPENAME	VARCHAR(128)		데이터 소스 데이터 유형의 규정되지 않은 이름.
REMOTE_META_TYPE	CHAR(1)	Y	이 리모트 유형이 시스템 내장 유형인지 구별 유형인지를 표시합니다. • S = 시스템 내장 유형 • T = 구별 유형
REMOTE_LOWER_LEN	INTEGER	Y	이 맵핑에서 리모트 데이터 유형의 최소 길이 또는 정밀도, 또는 널(NULL) 값입니다. 문자 유형의 경우 최소 문자 수(바이트 수가 아님)를 나타냅니다. 2진 유형의 경우 최소 바이트 수를 나타냅니다. 값 -1은 디폴트 길이 또는 정밀도가 사용되거나, 리모트 유형이 길이 또는 정밀도를 갖지 않음을 표시합니다.
REMOTE_UPPER_LEN	INTEGER	Y	이 맵핑에서 리모트 데이터 유형의 최대 길이 또는 정밀도, 또는 널(NULL) 값입니다. 문자 유형의 경우 최대 문자 수(바이트 수가 아님)를 나타냅니다. 2진 유형의 경우 최대 바이트 수를 나타냅니다. 값 -1은 디폴트 길이 또는 정밀도가 사용되거나, 리모트 유형이 길이 또는 정밀도를 갖지 않음을 표시합니다.

표 183. SYSCAT.TYPEMAPPINGS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
REMOTE_LOWER_SCALE	SMALLINT	Y	이 맵핑의 리모트 10진수 값에서 분수 부분의 최소 자릿수 또는 이 맵핑의 리모트 TIMESTAMP 값에서 분수 초의 최소 자릿수
REMOTE_UPPER_SCALE	SMALLINT	Y	이 맵핑의 리모트 10진수 값에서 분수 부분의 최대 자릿수 또는 이 맵핑의 리모트 TIMESTAMP 값에서 분수 초의 최대 자릿수
REMOTE_S_OPR_P	CHAR(2)	Y	이 맵핑의 리모트 10진수 값에서 스케일 및 정밀도 사이의 관계입니다. 기본 비교 연산자(=, <, >, <=, >=, <>)를 사용할 수 있습니다. 널(NULL) 값은 특정 관계가 필요하지 않음을 나타냅니다.
REMOTE_BIT_DATA	CHAR(1)	Y	이 리모트 문자 유형이 비트 데이터용인지 여부를 표시합니다. 가능한 값은 다음과 같습니다. • N = 이 유형이 비트 데이터용이 아님 • Y = 이 유형이 비트 데이터용임 • 널(NULL) 값 = 문자 데이터 유형이 아니거나 시스템이 비트 데이터 속성을 판별함
USER_DEFINED	CHAR(1)		맵핑이 사용자 정의되었는지의 여부를 나타냅니다. 값은 항상 'Y'입니다. 즉 맵핑이 항상 사용자 정의됩니다.
CREATE_TIME	TIMESTAMP		맵핑이 작성된 시간.
DEFINER[1]	VARCHAR(128)		유형 맵핑 소유자의 권한 부여 ID. 'SYSIBM'은 내장 유형 맵핑을 표시합니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.USEROPTIONS

각 행은 서버 특정 사용자 옵션 값을 나타냅니다.

표 184. SYSCAT.USEROPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
AUTHID	VARCHAR(128)		대문자로 된 로컬 권한 부여 ID.
AUTHIDTYPE	CHAR(1)		• U = 권한 받은 사용자가 개별 사용자임
SERVERNAME	VARCHAR(128)		사용자가 정의되는 서버의 이름.
OPTION	VARCHAR(128)		사용자 옵션의 이름.
SETTING	VARCHAR(2048)		사용자 옵션 값.

SYSCAT.VARIABLEAUTH

각 행은 모듈에 정의되지 않은 데이터베이스의 전역 변수에 대해 특정 권한 부여자가 하나 이상의 특권을 권한 부여한 사용자, 그룹 또는 역할을 나타냅니다.

표 185. SYSCAT.VARIABLEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
VARSCHEMA	VARCHAR(128)		VARMODULEID가 널(Null)인 경우 스키마 이름입니다. 그렇지 않으면 전역 변수가 속하는 모듈의 스키마 이름입니다.
VARNAME	VARCHAR(128)		전역 변수의 규정되지 않은 이름.
VARID	INTEGER		전역 변수 ID.
READAUTH	CHAR(1)		전역 변수를 읽을 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨
WRITEAUTH	CHAR(1)		전역 변수를 쓸 특권. • G = 보유 및 권한 부여 가능 • N = 보유되지 않음 • Y = 보유됨

SYSCAT.VARIABLEDEP

각 행은 일부 다른 오브젝트에 대한 전역 변수의 종속성을 나타냅니다. 전역 변수는 이름 BNAME의 유형 BTYPE의 오브젝트에 종속되므로, 오브젝트에 대한 변경사항이 전역 변수에 영향을 줍니다.

표 186. SYSCAT.VARIABLEDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
VARSCHEMA	VARCHAR(128)		다른 오브젝트에 종속되는 전역 변수의 스키마 이름.
VARMODULENAME	VARCHAR(128)	Y	전역 변수가 속하는 모듈의 규정되지 않은 이름. 모듈 변수가 아닌 경우에는 널(NULL) 값입니다.
VARNAME	VARCHAR(128)		다른 오브젝트에 종속되는 전역 변수의 규정되지 않은 이름.
VARMODULEID	INTEGER	Y	다른 오브젝트에 종속되는 오브젝트 모듈의 ID.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • F = 루틴 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름입니다. 루틴(BTYPE = 'F')의 경우, 이것은 특정 이름입니다.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.

표 186. SYSCAT.VARIABLEDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
TABAUTH	SMALLINT	Y	BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 전역 변수에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.VARIABLES

각 행은 전역 변수를 나타냅니다.

표 187. SYSCAT.VARIABLES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
VARSCHEMA	VARCHAR(128)		VARMODULEID가 널(Null)인 경우 스키마 이름입니다. 그렇지 않으면 전역 변수가 속하는 모듈의 스키마 이름입니다.
VARMODULENAME	VARCHAR(128)	Y	전역 변수가 속하는 모듈의 규정되지 않은 이름. 모듈 변수가 아닌 경우에는 널(NULL) 값입니다.
VARNAME	VARCHAR(128)		전역 변수의 규정되지 않은 이름.
VARMODULEID	INTEGER	Y	전역 변수가 속하는 모듈의 ID입니다. 모듈 변수가 아닌 경우에는 널(NULL) 값입니다.
VARID	INTEGER		전역 변수 ID.
OWNER	VARCHAR(128)		전역 변수 소유자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• U = 소유자가 개별 사용자임
CREATE_TIME	TIMESTAMP		전역 변수가 작성된 시간.
LAST_REGEN_TIME	TIMESTAMP		디폴트 표현식이 마지막으로 다시 만들어진 시간.
VALID	CHAR(1)		• N = 전역 변수가 유효하지 않음 • Y = 전역 변수가 유효함
PUBLISHED	CHAR(1)		모듈 변수를 모듈 밖에서 참조할 수 있는지 여부를 표시합니다. • N = 모듈 변수가 발행되지 않음 • Y = 모듈 변수가 발행됨 • 공백 = 적용할 수 없음
TYPESCHEMA	VARCHAR(128)		TYPEMODULEID가 널(Null)인 경우 데이터 유형의 스키마 이름이고, 그렇지 않으면 데이터 유형이 속하는 모듈의 스키마 이름입니다.
TYPEMODULENAME	VARCHAR(128)		변수 데이터 유형이 속하는 모듈의 규정되지 않은 이름. 변수 데이터 유형이 모듈에 속하지 않는 경우, 널(NULL) 값입니다.
TYPENAME	VARCHAR(128)		데이터 유형의 규정되지 않은 이름.
TYPEMODULEID	INTEGER	Y	변수 데이터 유형이 속하는 모듈의 ID. 변수 데이터 유형이 모듈에 속하지 않는 경우, 널(NULL) 값입니다.
LENGTH	INTEGER		전역 변수의 최대 길이.
SCALE	SMALLINT		전역 변수 데이터 유형이 DECIMAL을 기반으로 하는 구별 유형 또는 DECIMAL인 경우 스케일이고, 전역 변수 데이터 유형이 TIMESTAMP를 기반으로 하는 구별 유형 또는 TIMESTAMP인 경우 분수 초의 자릿수입니다. 그렇지 않으면, 0입니다.
CODEPAGE	SMALLINT		전역 변수의 코드 페이지.
COLLATIONSCHEMA	VARCHAR(128)		변수 조합의 스키마 이름.
COLLATIONNAME	VARCHAR(128)		변수 조합의 규정되지 않은 이름.
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		변수에서 ORDER BY절에 대한 조합의 스키마 이름.
COLLATIONNAME_ORDERBY	VARCHAR(128)		변수의 ORDER BY절에 대한 조합의 규정되지 않은 이름.

표 187. SYSCAT.VARIABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
SCOPE	CHAR(1)		전역 변수 범위. • S = 세션
DEFAULT	CLOB(64K)	Y	처음 참조될 때 전역 변수의 초기값을 계산하는 데 사용되는 표현식.
QUALIFIER	VARCHAR(128)	Y	변수가 정의된 시간에 디폴트 스키마의 값.
FUNC_PATH	CLOB(2K)	Y	변수가 정의된 시간의 SQL 경로
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.
READONLY	CHAR(1)		• C = 전역 변수가 CONSTANT절과 함께 정의되기 때문에 읽기 전용 • N = 읽기 전용이 아님

SYSCAT.VIEWS

각 행은 뷰 또는 구체화된 쿼리 테이블을 나타냅니다.

표 188. SYSCAT.VIEWS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
VIEWSCHEMA	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블의 스키마 이름.
VIEWNAME	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블의 규정되지 않은 이름.
OWNER	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
SEQNO	SMALLINT		항상 1입니다.
VIEWCHECK	CHAR(1)		뷰 점검 유형. • C = 연쇄 점검 옵션 • L = 로컬 점검 옵션 • N = 점검 옵션 없음 또는 구체화된 쿼리 테이블임
READONLY	CHAR(1)		• N = 적합한 권한이 부여된 사용자만 뷰를 갱신할 수 있거나 구체화된 쿼리 테이블임 • Y = 뷰가 정의 때문에 읽기 전용임
VALID	CHAR(1)		• N = 뷰 또는 구체화된 쿼리 테이블 정의가 유효하지 않음 • X = 뷰 또는 구체화된 쿼리 테이블 정의가 작동 불능이며 다시 작성해야 함 • Y = 뷰 또는 구체화된 쿼리 테이블 정의가 유효함
QUALIFIER	VARCHAR(128)		오브젝트 정의 시 디폴트 스키마의 값입니다. 규정되지 않은 참조를 완료하는 데 사용됩니다.
FUNC_PATH	CLOB(2K)		뷰 또는 구체화된 쿼리 테이블이 정의된 시간의 SQL 경로
TEXT	CLOB(2M)		입력된 그대로 뷰 또는 구체화된 쿼리 테이블 CREATE문의 전체 텍스트
DEFINER[1]	VARCHAR(128)		뷰 또는 구체화된 쿼리 테이블 소유자의 권한 부여 ID

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.WORKACTIONS

각 행은 작업 조치 세트에 대해 정의되는 작업 조치를 나타냅니다.

표 189. SYSCAT.WORKACTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ACTIONNAME	VARCHAR(128)		작업 조치 이름.
ACTIONID	INTEGER		작업 조치 ID.
ACTIONSETNAME	VARCHAR(128)	Y	작업 조치 세트 이름.
ACTIONSETID	INTEGER		이 작업 조치가 속하는 작업 조치 세트의 ID입니다. 이 컬럼은 SYSCAT.WORKACTIONSETS 뷰의 ACTIONSETID 컬럼을 참조합니다.
WORKCLASSNAME	VARCHAR(128)	Y	작업 클래스 이름.
WORKCLASSID	INTEGER		작업 클래스 ID입니다. 이 컬럼은 SYSCAT.WORKCLASSES 뷰의 WORKCLASSID 컬럼을 참조합니다.
CREATE_TIME	TIMESTAMP		작업 조치가 작성된 시간.
ALTER_TIME	TIMESTAMP		작업 조치가 마지막으로 변경된 시간.
ENABLED	CHAR(1)		• N = 이 작업 조치를 사용 안합니다. • Y = 이 작업 조치가 사용 가능합니다.

표 189. SYSCAT.WORKACTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
ACTIONTYPE	CHAR(1)		범위내의 작업 클래스 속성에 일치하는 각 DB2 활동에서 수행된 조치 유형입니다. • B = 기본 집계 활동 데이터를 수집하고, 서비스 클래스에 적용되는 작업 조치 세트에만 지정할 수 있습니다. • C = 관련 작업 클래스에서 모든 DB2 활동을 실행하고 작업 클래스 카운터를 증가시킬 수 있습니다. • D = 활동 코디네이터의 모든 데이터베이스 파티션에서 활동 데이터를 세부사항과 함께 수집합니다. • E = 확장 집계 활동 데이터를 수집하고, 서비스 클래스에 적용되는 작업 조치 세트에만 지정할 수 있습니다. • F = 활동 코디네이터의 모든 데이터베이스 파티션에서 활동 데이터를 세부사항, 섹션 및 값과 함께 수집합니다. • G = 활동 데이터의 데이터베이스 파티션에서 활동 세부사항 및 섹션을 수집하고 모든 데이터베이스 파티션에서 활동 데이터를 수집합니다. • H = 활동 데이터의 데이터베이스 파티션에서 활동 세부사항, 섹션 및 값을 수집하고 모든 데이터베이스 파티션에서 활동 데이터를 수집합니다. • M = 서비스 서브클래스에 맵핑되고, 서비스 클래스에 적용되는 작업 조치 세트에만 지정할 수 있습니다. • P = 이 작업 조치가 연관된 작업 클래스에서 모든 DB2 활동 실행을 방지합니다. • S = 활동 코디네이터의 데이터베이스 파티션에서 활동 데이터를 세부사항 및 값과 함께 수집합니다. • T = 이 조치는 임계값을 나타내며, 데이터베이스와 연관되는 작업 조치 세트에만 지정할 수 있습니다. • U = 중첩 레벨이 0인 모든 활동 및 이들 활동 아래에 중첩되는 모든 활동을 서비스 서브클래스에 맵핑합니다. 서비스 클래스에 적용되는 작업 조치 세트에 대해서만 지정할 수 있습니다. • V = 코디네이터 파티션에서 세부사항 및 값과 함께 활동 데이터를 수집합니다. • W = 코디네이터 파티션에서 세부사항 없이 활동 데이터를 수집합니다. • X = 코디네이터 파티션에서 세부사항과 함께 활동 데이터를 수집하고 모든 데이터베이스 파티션에서 활동 데이터를 수집합니다. • Y = 코디네이터 파티션에서 세부사항 및 값과 함께 활동 데이터를 수집하고 모든 데이터베이스 파티션에서 활동 데이터를 수집합니다. • Z = 모든 데이터베이스 파티션에서 세부사항 없이 활동 데이터를 수집합니다.

표 189. SYSCAT.WORKACTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
REFOBJECTID	INTEGER	Y	ACTIONTYPE이 'M'(맵) 또는 'N'(중첩된 맵)인 경우 이 값은 DB2 활동이 맵핑되는 서비스 서브클래스의 ID로 설정됩니다. ACTIONTYPE이 'T'(임계값)인 경우 이 값은 사용될 임계값의 ID로 설정됩니다. 다른 모든 조치의 경우 이 값은 널(Null)입니다.
REFOBJECTTYPE	VARCHAR (30)		ACTIONTYPE이 'M' 또는 'N'이면 이 값은 'SERVICE CLASS'로 설정되고, ACTIONTYPE이 'T'이면 이 값은 'THRESHOLD'입니다. 그렇지 않으면, 널(NULL) 값입니다.

SYSCAT.WORKACTIONSETS

각 행은 작업 조치 세트를 나타냅니다.

표 190. SYSCAT.WORKACTIONSETS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
ACTIONSETNAME	VARCHAR(128)		작업 조치 세트 이름.
ACTIONSETID	INTEGER		작업 조치 세트 ID.
WORKCLASSSETNAME	VARCHAR(128)	Y	작업 클래스 세트 이름.
WORKCLASSSETID	INTEGER		OBJECTID로 지정되는 오브젝트에 맵핑될 작업 클래스 세트의 ID입니다. 이 컬럼은 SYSCAT.WORKCLASSSETS 뷰의 WORKCLASSSETID를 참조합니다.
CREATE_TIME	TIMESTAMP		작업 조치 세트가 작성된 시간.
ALTER_TIME	TIMESTAMP		작업 조치 세트가 마지막으로 변경된 시간.
ENABLED	CHAR(1)		• N = 이 작업 조치 세트를 사용 안함 • Y = 이 작업 조치 세트가 사용 가능함
OBJECTTYPE	CHAR(1)		• b = 서비스 수퍼 클래스 • 공백 = 데이터베이스
OBJECTNAME	VARCHAR(128)	Y	서비스 클래스 이름.
OBJECTID	INTEGER		작업 클래스 세트(WORKCLASSSETID로 지정되는)가 맵핑되는 오브젝트의 ID입니다. OBJECTTYPE이 공백인 경우 OBJECTID는 -1입니다. OBJECTTYPE이 'b'인 경우 OBJECTID는 서비스 수퍼 클래스 ID입니다.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.WORKCLASSES

각 행은 작업 클래스 세트에 대해 정의되는 작업 클래스를 나타냅니다.

표 191. SYSCAT.WORKCLASSES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WORKCLASSNAME	VARCHAR(128)		작업 클래스 이름.
WORKCLASSSETNAME	VARCHAR(128)	Y	작업 클래스 세트 이름.
WORKCLASSID	INTEGER		작업 클래스 ID.
WORKCLASSSETID	INTEGER		이 작업 클래스가 속하는 작업 클래스 세트의 ID 입니다. 이 컬럼은 SYSCAT.WORKCLASSSETS 뷰의 WORKCLASSSETID 컬럼을 참조합니다.
CREATE_TIME	TIMESTAMP		작업 클래스가 작성된 시간.
ALTER_TIME	TIMESTAMP		작업 클래스가 마지막으로 변경된 시간.
WORKTYPE	SMALLINT		DB2 활동 유형. • 1 = ALL • 2 = READ • 3 = WRITE • 4 = CALL • 5 = DML • 6 = DDL • 7 = LOAD
RANGEUNITS	CHAR(1)		맨 아래 및 맨 위 범위에 사용할 단위. • C = 카디널리티(cardinality) • T = Timeron • 공백 = 적용할 수 없음
FROMVALUE	DOUBLE	Y	RANGEUNITS로 지정되는 단위로 표시되는 범위의 낮은 값입니다. RANGEUNITS가 공백이면 널(NULL) 값입니다.
TOVALUE	DOUBLE	Y	RANGEUNITS로 지정되는 단위로 표시되는 범위의 높은 값입니다. RANGEUNITS가 공백이면 널(NULL) 값입니다. -1 값은 상한 없음을 표시하는 데 사용됩니다.
ROUTINESCHEMA	VARCHAR(128)	Y	CALL문에서 호출되는 프로시저의 스키마 이름입니다. WORKTYPE이 4(CALL) 또는 1(ALL)이 아닐 때는 널(NULL) 값입니다.
INITIALSQLDATAPRIORITY	CHAR(1)		나중에 사용하기 위해 예약됩니다.
EVALUATIONORDER	SMALLINT		작업 클래스 세트에서 작업 클래스 선택에 사용되는 평가 순서를 고유하게 식별합니다.

SYSCAT.WORKCLASSSETS

각 행은 작업 클래스 세트를 나타냅니다.

표 192. SYSCAT.WORKCLASSSETS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WORKCLASSSETNAME	VARCHAR(128)		작업 클래스 세트 이름.
WORKCLASSSETID	INTEGER		작업 클래스 세트 ID.
CREATE_TIME	TIMESTAMP		작업 클래스 세트가 작성된 시간.
ALTER_TIME	TIMESTAMP		작업 클래스 세트가 마지막으로 변경된 시간.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.WORKLOADAUTH

각 행은 워크로드에 대해 USAGE 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 193. SYSCAT.WORKLOADAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WORKLOADID	INTEGER		워크로드 ID.
WORKLOADNAME	VARCHAR(128)		워크로드 이름.
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• U = 권한 받은 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
USAGEAUTH	CHAR(1)		권한 받은 사용자가 워크로드에 대한 USAGE 특권을 보유하는지를 표시합니다. • N = 보유되지 않음 • Y = 보유됨

SYSCAT.WORKLOADCONNATTR

각 행은 워크로드의 정의에 있는 연결 속성을 나타냅니다.

표 194. SYSCAT.WORKLOADCONNATTR 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WORKLOADID	INTEGER		워크로드 ID.
WORKLOADNAME	VARCHAR(128)		워크로드 이름.
CONNATTRTYPE	VARCHAR (30)		연결 속성의 유형. • 1 = APPLNAME • 2 = SYSTEM_USER • 3 = SESSION_USER • 4 = SESSION_USER GROUP • 5 = SESSION_USER ROLE • 6 = CURRENT CLIENT_USERID • 7 = CURRENT CLIENT_APPLNAME • 8 = CURRENT CLIENT_WRKSTNNAME • 9 = CURRENT CLIENT_ACCTNG • 10 = ADDRESS
CONNATTRVALUE	VARCHAR(1000)		연결 속성의 값.

SYSCAT.WORKLOADS

각 행은 워크로드를 나타냅니다.

표 195. SYSCAT.WORKLOADS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WORKLOADID	INTEGER		워크로드 ID.
WORKLOADNAME	VARCHAR(128)		워크로드 이름.
EVALUATIONORDER	SMALLINT		워크로드 선택에 사용되는 평가 순서.
CREATE_TIME	TIMESTAMP		워크로드가 작성된 시간.
ALTER_TIME	TIMESTAMP		워크로드가 마지막으로 변경된 시간.
ENABLED	CHAR(1)		• N = 이 워크로드를 사용 안합니다. • Y = 이 워크로드가 사용 가능합니다.
ALLOWACCESS	CHAR(1)		• N = 이 워크로드와 연관된 UOW는 거부됩니다. • Y = 이 워크로드와 연관된 작업 단위(UOW)가 데이터베이스에 액세스할 수 있습니다.
SERVICECLASSNAME	VARCHAR(128)		작업 단위(UOW)(이 워크로드와 연관된)가 지정되는 서비스 서브클래스 이름.
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	작업 단위(UOW)(이 워크로드와 연관된)가 지정되는 서비스 수퍼 클래스 이름.
COLLECTAGGACTDATA	CHAR(1)		적용 가능한 이벤트 모니터가 워크로드에 대해 캡처할 집계 활동 데이터를 지정합니다. • B = 기본 집계 활동 데이터 수집 • E = 확장 집계 활동 데이터 수집 • N = 없음
COLLECTACTDATA	CHAR(1)		적용 가능한 이벤트 모니터가 수집할 활동 데이터를 지정합니다. • D = 세부사항을 갖는 활동 데이터 • N = 없음 • S = 세부사항 및 섹션 환경이 있는 활동 데이터 • V = 세부사항 및 값이 있는 활동 데이터. COLLECT 컬럼이 'C'로 설정될 때 적용됩니다. • W = 세부사항이 없는 활동 데이터 • X = 세부사항, 섹션 환경 및 값이 있는 활동 데이터
COLLECTACTPARTITION	CHAR(1)		활동 데이터가 수집되는 위치를 지정합니다. • C = 활동 코디네이터의 데이터베이스 파티션 • D = 모든 데이터베이스 파티션

표 195. SYSCAT.WORKLOADS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
COLLECTDEADLOCK	CHAR(1)		적용 가능한 이벤트 모니터가 교착 상태 이벤트를 수집해야 함을 지정합니다. • H = 지나간 활동만을 갖는 교착 상태 데이터 수집 • N = 교착 상태 데이터를 수집하지 않음 • V = 지나간 활동 및 값과 함께 교착 상태 데이터 수집 • W = 지나간 활동 및 값이 없이 교착 상태 데이터 수집
COLLECTLOCKTIMEOUT	CHAR(1)		적용 가능한 이벤트 모니터가 잠금 시간종료 이벤트를 수집해야 함을 지정합니다. • H = 지나간 활동만을 갖는 잠금 시간종료 데이터 수집 • N = 잠금 시간종료 데이터를 수집하지 않음 • V = 지나간 활동 및 값과 함께 잠금 시간종료 데이터 수집 • W = 지나간 활동 및 값 없이 잠금 시간종료 데이터 수집
COLLECTLOCKWAIT	CHAR(1)		적용 가능한 이벤트 모니터가 잠금 대기 이벤트를 수집해야 함을 지정합니다. • H = 지나간 활동만을 갖는 잠금 대기 데이터 수집 • N = 잠금 대기 데이터를 수집하지 않음 • V = 지나간 활동 및 값과 함께 잠금 대기 데이터 수집 • W = 지나간 활동 및 값 없이 잠금 대기 데이터 수집
LOCKWAITVALUE	INTEGER		적용 가능한 이벤트 모니터가 잠금 이벤트를 수집하기 전에 잠금이 대기할 시간을 밀리초 단위로 지정합니다. COLLECTLOCKWAIT = 'N'이면 0입니다.
COLLECTACTMETRICS	CHAR(1)		워크로드 어커런스가 제출하는 활동에 대한 모니터링 레벨을 지정합니다. • B = 기본 활동 메트릭 수집 • E = 확장 활동 메트릭 수집 • N = 없음
COLLECTUOWDATA	CHAR(1)		적용 가능한 이벤트 모니터가 수집할 작업 단위(UOW) 데이터를 지정합니다. • B = 기본 작업 단위(UOW) 데이터 수집 • N = 없음

표 195. SYSCAT.WORKLOADS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
EXTERNALNAME	VARCHAR(128)	Y	나중에 사용하기 위해 예약됨
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.WRAPOPTIONS

각 행은 랩퍼 특정 옵션을 나타냅니다.

표 196. SYSCAT.WRAPOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WRAPNAME	VARCHAR(128)		랩퍼 이름.
OPTION	VARCHAR(128)		랩퍼 옵션 이름.
SETTING	VARCHAR(2048)		랩퍼 옵션 값.

SYSCAT.WRAPPERS

각 행은 등록된 랩퍼를 나타냅니다.

표 197. SYSCAT.WRAPPERS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
WRAPNAME	VARCHAR(128)		랩퍼 이름.
WRAPTYPE	CHAR(1)		랩퍼 유형. • N = 비관계형 • R = 관계형
WRAPVERSION	INTEGER		랩퍼 버전.
LIBRARY	VARCHAR(255)		이 랩퍼와 연관된 데이터 소스와 통신하는 데 사용된 코드를 포함하는 파일 이름.
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSCAT.XDBMAPGRAPHS

각 행은 XDB 맵(XSR 오브젝트)의 스키마 그래프를 나타냅니다.

표 198. SYSCAT.XDBMAPGRAPHS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XSR 오브젝트에 대한 고유 생성 ID.
OBJECTSCHEMA	VARCHAR(128)		XSR 오브젝트의 스키마 이름.
OBJECTNAME	VARCHAR(128)		XSR 오브젝트의 규정되지 않은 이름.
SCHEMAGRAPHID	INTEGER		XDB 맵 ID 내에서 고유한 스키마 그래프 ID.
NAMESPACE	VARCHAR(1001)	Y	루트 요소의 이름 스페이스 URI에 대한 문자열 ID.
ROOTELEMENT	VARCHAR(1001)	Y	루트 요소의 요소 이름에 대한 문자열 ID.

SYSCAT.XDBMAPSHREDTREES

각 행은 주어진 스키마 그래프 ID에 대해 하나의 조각 트리를 나타냅니다.

표 199. SYSCAT.XDBMAPSHREDTREES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XSR 오브젝트에 대한 고유 생성 ID.
OBJECTSCHEMA	VARCHAR(128)		XSR 오브젝트의 스키마 이름.
OBJECTNAME	VARCHAR(128)		XSR 오브젝트의 규정되지 않은 이름.
SCHEMAGRAPHID	INTEGER		XDB 맵 ID 내에서 고유한 스키마 그래프 ID.
SHREDTREEID	INTEGER		XDB 맵 ID 내에서 고유한 조각 트리 ID.
MAPPINGDESCRIPTION	CLOB(1M)	Y	진단 맵핑 정보.

SYSCAT.XMLSTRINGS

각 행은 구조적 XML 데이터를 압축하는 데 사용되는 단일 문자열 및 고유 문자열 ID를 나타냅니다. 문자열은 UTF-8 인코딩 및 데이터베이스 코드 페이지 인코딩 모두로 제공됩니다.

표 200. SYSCAT.XMLSTRINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
STRINGID	INTEGER		고유 문자열 ID.
STRING	VARCHAR(1001)		데이터베이스 코드 페이지로 표시되는 문자열.
STRING_UTF8	VARCHAR(1001)		UTF-8 인코딩의 문자열(카탈로그 테이블에 저장되는 대로).

SYSCAT.XSROBJECTAUTH

각 행은 특정 XSR 오브젝트에 대한 USAGE 특권이 권한 부여된 사용자, 그룹 또는 역할을 나타냅니다.

표 201. SYSCAT.XSROBJECTAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
GRANTOR	VARCHAR(128)		특권의 권한 준 사용자.
GRANTORTYPE	CHAR(1)		• S = 권한 준 사용자가 시스템임 • U = 권한 준 사용자가 개별 사용자임
GRANTEE	VARCHAR(128)		특권 보유자.
GRANTEETYPE	CHAR(1)		• G = 권한 받은 사용자가 그룹임 • R = 권한 받은 사용자가 역할임 • U = 권한 받은 사용자가 개별 사용자임
OBJECTID	BIGINT		XSR 오브젝트 ID.
USAGEAUTH	CHAR(1)		XSR 오브젝트 및 해당 구성요소를 사용할 특권. • N = 보유되지 않음 • Y = 보유됨

SYSCAT.XSROBJECTCOMPONENTS

각 행은 XSR 오브젝트 구성요소를 나타냅니다.

표 202. SYSCAT.XSROBJECTCOMPONENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XSR 오브젝트에 대한 고유 생성 ID.
OBJECTSCHEMA	VARCHAR(128)		XSR 오브젝트의 스키마 이름.
OBJECTNAME	VARCHAR(128)		XSR 오브젝트의 규정되지 않은 이름.
COMPONENTID	BIGINT		XSR 오브젝트 구성요소에 대한 고유 생성 ID.
TARGETNAMESPACE	VARCHAR(1001)	Y	목표 이름 스페이스의 문자열 ID.
SCHEMALOCATION	VARCHAR(1001)	Y	스키마 위치에 대한 문자열 ID.
COMPONENT	BLOB(30M)		구성요소의 외부 표현.
CREATE_TIME	TIMESTAMP		XSR 오브젝트 구성요소가 등록된 시간.
STATUS	CHAR(1)		등록 상태. • C = 완료 • I = 미완료

SYSCAT.XSROBJECTDEP

각 행은 일부 다른 오브젝트에 대한 XSR 오브젝트의 종속성을 나타냅니다. XSR 오브젝트는 이름 BNAME의 유형 BTYPE의 오브젝트에 종속되므로, 오브젝트에 대한 변경사항이 XSR 오브젝트에 영향을 줍니다.

표 203. SYSCAT.XSROBJECTDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XSR 오브젝트에 대한 고유 생성 ID.
OBJECTSCHEMA	VARCHAR(128)		XSR 오브젝트의 스키마 이름.
OBJECTNAME	VARCHAR(128)		XSR 오브젝트의 규정되지 않은 이름.
BTYPE	CHAR(1)		종속성이 있는 오브젝트의 유형입니다. 가능한 값은 다음과 같습니다. • A = 테이블 별명 • B = 트리거 • F = 루틴 • G = 전역 임시 테이블 • H = 계층 구조 테이블 • K = 패키지 • L = 접속 해제된 테이블 • N = 별칭 • O = 테이블 또는 뷰 계층 구조의 모든 부속 테이블 또는 하위 뷰에 대한 특권 종속성 • Q = 시퀀스 • R = 사용자 정의 데이터 유형 • S = 구체화된 쿼리 테이블 • T = 테이블(유형이 지정되지 않음) • U = 유형이 지정된 테이블 • V = 뷰(유형이 지정되지 않음) • W = 유형이 지정된 뷰 • X = 인덱스 확장 • Z = XSR 오브젝트 • q = 시퀀스 별명 • u = 모듈 별명 • v = 전역 변수 • * = 기본 테이블의 행에 고정됨
BSCHEMA	VARCHAR(128)		종속성이 있는 오브젝트의 스키마 이름.
BMODULENAME	VARCHAR(128)	Y	종속성이 있는 오브젝트가 속하는 모듈의 규정되지 않은 이름. 모듈 오브젝트가 아니면 널(NULL) 값입니다.

표 203. SYSCAT.XSROBJECTDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
BNAME	VARCHAR(128)		종속성이 있는 오브젝트의 규정되지 않은 이름입니다. 루틴(BTYPE = 'F')의 경우, 이것은 특정 이름입니다.
BMODULEID	INTEGER	Y	종속성이 있는 오브젝트의 모듈에 대한 ID.
TABAUTH	SMALLINT	Y	BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' 또는 'v'인 경우, 종속 트리거에서 필요한 테이블 또는 뷰에 대한 특권을 인코드합니다. 그렇지 않으면 널(NULL) 값입니다.

SYSCAT.XSROBJECTDETAILS

각 행은 XML 스키마 저장소 오브젝트를 나타냅니다.

표 204. SYSCAT.XSROBJECTDETAILS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XML 스키마 오브젝트에 대한 고유 생성 ID
OBJECTSCHEMA	VARCHAR(128)		XML 스키마 오브젝트의 스키마 이름
OBJECTNAME	VARCHAR(128)		XML 스키마 오브젝트의 규정되지 않은 이름
GRAMMAR	BLOB(127M)	Y	XML 스키마 오브젝트에 대한 문법의 2진 표현
PROPERTIES	BLOB(4190000)	Y	XML 스키마 오브젝트에 대한 등록 정보 문서

SYSCAT.XSROBJECTHIERARCHIES

각 행은 XSR 오브젝트와 해당 구성요소 사이의 계층 관계를 나타냅니다.

표 205. SYSCAT.XSROBJECTHIERARCHIES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XSR 오브젝트 ID.
COMPONENTID	BIGINT		XSR 구성요소 ID.
HTYPE	CHAR(1)		계층 구조 유형. • D = 문서 • N = 최상위 레벨 이름 스페이스 • P = 기본 문서
TARGETNAMESPACE	VARCHAR(1001)	Y	구성요소의 목표 이름 스페이스에 대한 문자열 ID.
SCHEMALOCATION	VARCHAR(1001)	Y	구성요소의 스키마 위치에 대한 문자열 ID.

SYSCAT.XSROBJECTS

각 행은 XML 스키마 저장소 오브젝트를 나타냅니다.

표 206. SYSCAT.XSROBJECTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
OBJECTID	BIGINT		XSR 오브젝트에 대한 고유 생성 ID.
OBJECTSCHEMA	VARCHAR(128)		XSR 오브젝트의 스키마 이름.
OBJECTNAME	VARCHAR(128)		XSR 오브젝트의 규정되지 않은 이름.
TARGETNAMESPACE	VARCHAR(1001)	Y	목표 이름 스페이스의 문자열 ID 또는 공용 ID.
SCHEMALOCATION	VARCHAR(1001)	Y	스키마 위치에 대한 문자열 ID 또는 시스템 ID.
OBJECTINFO	XML	Y	메타데이터 문서.
OBJECTTYPE	CHAR(1)		XSR 오브젝트 유형. • D = DTD • E = 외부 엔티티 • S = XML 스키마
OWNER	VARCHAR(128)		XSR 오브젝트 소유자의 권한 부여 ID.
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
CREATE_TIME	TIMESTAMP		오브젝트가 등록된 시간.
ALTER_TIME	TIMESTAMP		오브젝트가 최종 갱신(교체)된 시간.
STATUS	CHAR(1)		등록 상태. • C = 완료 • I = 미완료 • R = 바꾸기 • T =임시
DECOMPOSITION	CHAR(1)		이 XSR 오브젝트에서 분석(조각내기)이 사용 가능한지를 표시합니다. • N = 사용 가능하지 않음 • X = 작동 불능 • Y = 사용 가능
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

SYSIBM.SYSDUMMY1

한 행을 포함합니다. 이 뷰는 z/OS용 DB2와의 호환성에 필요한 응용프로그램에 사용할 수 있습니다.

표 207. SYSIBM.SYSDUMMY1 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
IBMREQD	CHAR(1)		'Y'

SYSSTAT.COLDIST

각 행은 일부 컬럼의 *n*번째로 자주 사용되는 값이나 컬럼의 *n*번째 Quantile(누적 분산) 값을 나타냅니다. 실제 테이블의 컬럼(뷰가 아님)에만 적용됩니다. 유형이 지정된 테이블의 상속된 컬럼에 대해서는 통계가 기록되지 않습니다.

표 208. SYSSTAT.COLDIST 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
TABSCHEMA	VARCHAR(128)			통계가 적용되는 테이블의 스키마 이름.
TABNAME	VARCHAR(128)			통계가 적용되는 테이블의 규정되지 않은 이름.
COLNAME	VARCHAR(128)			통계가 적용되는 컬럼 이름.
TYPE	CHAR(1)			• F = 빈도 값 • Q = Quantile 값
SEQNO	SMALLINT			TYPE = 'F'인 경우, 이 컬럼의 *n*이 *n*번째로 자주 사용되는 값을 식별합니다. TYPE = 'Q'인 경우, 이 컬럼의 *n*은 *n*번째 Quantile 값을 식별합니다.
COLVALUE[1]	VARCHAR(254)	Y	Y	문자 리터럴 또는 널(NULL) 값으로서의 데이터 값.
VALCOUNT	BIGINT		Y	TYPE = 'F'인 경우, VALCOUNT는 컬럼에서 COLVALUE의 어커런스 수입니다. TYPE = 'Q'인 경우, VALCOUNT는 값이 COLVALUE보다 작거나 같은 행 수입니다.
DISTCOUNT[2]	BIGINT	Y	Y	TYPE = 'Q'인 경우, 이 컬럼은 COLVALUE보다 작거나 같은 구별 값 수를 기록합니다(사용 불가능한 경우 널(NULL) 값).

주:

1. 카탈로그 뷰에서 COLVALUE의 값이 항상 데이터베이스 코드 페이지에 표시되며 대체 문자를 포함할 수 있습니다. 그러나 통계는 컬럼 테이블의 코드 페이지에서 내부적으로 수집되며, 쿼리 최적화 중에 적용될 때 실제 컬럼 값을 사용합니다.
2. DISTCOUNT는 인덱스에서 첫 번째 키 컬럼인 컬럼의 경우에만 수집됩니다.

SYSSTAT.COLGROUPDIST

각 행은 컬럼 그룹의 *n*번째로 자주 사용되는 값이나 컬럼 그룹의 *n*번째 Quantile 값으로 구성되는 컬럼 그룹의 컬럼 값을 나타냅니다.

표 209. SYSSTAT.COLGROUPDIST 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
COLGROUPID	INTEGER			컬럼 그룹 ID.
TYPE	CHAR(1)			• F = 빈도 값 • Q = Quantile 값
ORDINAL	SMALLINT			컬럼 그룹에 있는 컬럼의 서수.
SEQNO	SMALLINT			TYPE = 'F'인 경우, 이 컬럼의 *n*이 *n*번째로 자주 사용되는 값을 식별합니다. TYPE = 'Q'인 경우, 이 컬럼의 *n*은 *n*번째 Quantile 값을 식별합니다.
COLVALUE	VARCHAR(254)	Y		문자 리터럴 또는 널(NULL) 값으로서의 데이터 값.

SYSSTAT.COLGROUPDISTCOUNTS

각 행은 컬럼 그룹의 *n*번째로 자주 사용되는 값이나 컬럼 그룹의 *n*번째 Quantile에 적용되는 분산 통계를 나타냅니다.

표 210. SYSSTAT.COLGROUPDISTCOUNTS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
COLGROUPID	INTEGER			컬럼 그룹 ID.
TYPE	CHAR(1)			• F = 빈도 값 • Q = Quantile 값
SEQNO	SMALLINT			*n*번째 TYPE 값을 나타내는 시퀀스 번호 *n*.
VALCOUNT	BIGINT		Y	TYPE = 'F'인 경우, VALCOUNT는 이 SEQNO를 갖는 컬럼 그룹에 대한 COLVALUE의 어커런스 수입니다. TYPE = 'Q'인 경우, VALCOUNT는 값이 이 SEQNO를 갖는 컬럼 그룹에 대한 COLVALUE보다 작거나 같은 행 수입니다.
DISTCOUNT	BIGINT		Y	TYPE = 'Q'인 경우, 이 컬럼은 이 SEQNO를 갖는 컬럼 그룹에 대한 COLVALUE보다 작거나 같은 구별 값 수를 기록합니다(사용 불가능한 경우 널(NULL) 값).

SYSSTAT.COLGROUPS

각 행은 전체 컬럼 그룹에 적용되는 컬럼 그룹 및 통계를 나타냅니다.

표 211. SYSSTAT.COLGROUPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
COLGROUPSCHEMA	VARCHAR(128)			컬럼 그룹의 스키마 이름.
COLGROUPNAME	VARCHAR(128)			컬럼 그룹의 규정되지 않은 이름.
COLGROUPID	INTEGER			컬럼 그룹 ID.
COLGROUPCARD	BIGINT		Y	컬럼 그룹의 카디널리티(cardinality).
NUMFREQ_VALUES	SMALLINT			컬럼 그룹에 대해 수집된 자주 사용되는 값 수.
NUMQUANTILES	SMALLINT			컬럼 그룹에 대해 수집된 Quantile 수.

SYSSTAT.COLUMNS

각 행은 테이블, 뷰 또는 별칭에 정의된 컬럼을 나타냅니다.

표 212. SYSSTAT.COLUMNS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
TABSCHEMA	VARCHAR(128)			컬럼을 포함하는 테이블, 뷰 또는 별칭의 스키마 이름.
TABNAME	VARCHAR(128)			컬럼을 포함하는 테이블, 뷰 또는 별칭의 규정되지 않은 이름.
COLNAME	VARCHAR(128)			컬럼 이름
COLCARD	BIGINT		Y	컬럼에 있는 구별 값의 수입니다. 통계가 수집되지 않는 경우 -1이며, 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 -2입니다.
HIGH2KEY[1]	VARCHAR(254)	Y	Y	두 번째로 높은 데이터 값입니다. 문자 리터럴로 변경된 숫자 데이터의 표시입니다. 통계가 수집되지 않는 경우 비어 있습니다. 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 비어 있습니다.
LOW2KEY[1]	VARCHAR(254)	Y	Y	두 번째로 낮은 데이터 값입니다. 문자 리터럴로 변경된 숫자 데이터의 표시입니다. 통계가 수집되지 않는 경우 비어 있습니다. 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 비어 있습니다.
AVGCOLLEN	INTEGER		Y	컬럼이 데이터베이스 메모리 또는 임시 테이블에 저장될 때 바이트 단위의 평균 스페이스입니다. 인라인되지 않는 LOB 데이터 유형, LONG 데이터 유형 및 XML 문서의 경우 평균 컬럼 길이를 계산하는 데 사용되는 값은 데이터 디스크립터의 길이입니다. 컬럼이 널(NULL) 입력 가능한 경우 추가 바이트가 필요합니다. 통계가 수집되지 않은 경우 -1이며, 상속된 컬럼 및 계층 구조 테이블의 컬럼의 경우 -2입니다. 주: 디스크에 컬럼을 저장하기 위해 필요한 평균 스페이스는 이 통계로 표시되는 값과 다를 수 있습니다.
NUMNULLS	BIGINT		Y	컬럼에 있는 널(NULL) 값의 수입니다. 통계가 수집되지 않는 경우 -1입니다.
PCTINLINED	SMALLINT			인라인된 XML 문서 또는 LOB 데이터의 백분율입니다. 통계가 수집되지 않은 경우 -1입니다.
SUB_COUNT	SMALLINT		Y	컬럼의 평균 하위 요소 수입니다. 문자열 컬럼에만 적용 가능합니다.
SUB_DELIM_LENGTH	SMALLINT		Y	컬럼의 각 하위 요소를 분리하는 분리문자의 평균 길이입니다. 문자열 컬럼에만 적용 가능합니다.

표 212. SYSSTAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
AVGCOLLENCHAR	INTEGER		Y	컬럼에 필요한 (컬럼에 영향을 미치는 조합을 기반으로) 평균 문자 수는 컬럼의 데이터 유형이 길거나, LOB, XML 또는 통계가 수집되지 않은 경우 -1이며 상속된 컬럼 및 계층 테이블의 컬럼의 경우 -2입니다.

주:

1. 카탈로그 뷰에서 HIGH2KEY 및 LOW2KEY의 값이 항상 데이터베이스 코드 페이지에 표시되며 대체 문자를 포함할 수 있습니다. 그러나 통계는 컬럼 테이블의 코드 페이지에서 내부적으로 수집되며, 쿼리 최적화 중에 적용될 때 실제 컬럼 값을 사용합니다.

SYSSTAT.INDEXES

각 행은 인덱스를 나타냅니다. 유형이 지정된 테이블의 인덱스는 두 행으로 표시되는데, 하나는 유형이 지정된 테이블의 "논리적 인덱스"에 대한 것이고 다른 하나는 계층 구조 테이블의 "H 인덱스"에 대한 것입니다.

표 213. SYSSTAT.INDEXES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
INDSCHEMA	VARCHAR(128)			인덱스의 스키마 이름.
INDNAME	VARCHAR(128)			인덱스의 규정되지 않은 이름.
TABSCHEMA	VARCHAR(128)			인덱스가 정의되는 테이블 또는 별칭의 스키마 이름.
TABNAME	VARCHAR(128)			인덱스가 정의되는 테이블 또는 별칭의 규정되지 않은 이름.
COLNAMES	VARCHAR(640)			이 컬럼은 더 이상 사용되지 않으며 추후 릴리스에서 제거됩니다.
NLEAF	BIGINT		Y	리프 페이지 수. 통계가 수집되지 않는 경우 -1입니다.
NLEVELS	SMALLINT		Y	인덱스 레벨 수. 통계가 수집되지 않는 경우 -1입니다.
FIRSTKEYCARD	BIGINT		Y	구별 최초 키 값 수. 통계가 수집되지 않는 경우 -1입니다.
FIRST2KEYCARD	BIGINT		Y	인덱스의 처음 두 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FIRST3KEYCARD	BIGINT		Y	인덱스의 처음 세 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FIRST4KEYCARD	BIGINT		Y	인덱스의 처음 네 컬럼을 사용하는 구별 키 수입니다. 통계가 수집되지 않거나 적용할 수 없는 경우 -1입니다.
FULLKEYCARD	BIGINT		Y	구별 전체 키 값 수. 통계가 수집되지 않는 경우 -1입니다.
CLUSTERRATIO[4]	SMALLINT		Y	인덱스를 사용한 데이터 클러스터링 등급입니다. 통계가 수집되지 않거나 상세한 인덱스 통계가 수집되는 경우(이 경우 CLUSTERFACTOR가 대신 사용됨) -1입니다.
CLUSTERFACTOR[4]	DOUBLE		Y	클러스터링 등급의 상세 측정입니다. 통계가 수집되지 않는 경우 또는 인덱스가 별칭에 대해 정의되는 경우 -1입니다.
SEQUENTIAL_PAGES	BIGINT		Y	리프 페이지 사이의 큰 갭이 없거나 소수인 인덱스 키 순서에서 디스크에 위치한 리프 페이지 수입니다. 통계가 수집되지 않는 경우 -1입니다.

표 213. SYSSTAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
DENSITY	INTEGER		Y	인덱스로 채워지는 페이지 범위에 있는 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로, 퍼센트(0 - 100 범위의 정수)로 표현됩니다. 통계가 수집되지 않는 경우 -1입니다.
PAGE_FETCH_PAIRS[4]	VARCHAR(520)		Y	문자 양식으로 표시되는 정수 쌍의 목록입니다. 각 쌍은 가상 버퍼의 페이지 수 및 해당 가상 버퍼를 사용하여 이 인덱스를 갖는 테이블을 스캔하기 위해 필요한 페이지 페치 수입니다. 사용 가능한 데이터가 없는 경우 길이가 0인 문자열입니다.
NUMRIDS[4]	BIGINT		Y	인덱스의 행 ID(RID) 또는 블록 ID의 총수이며, 알 수 없는 경우 -1입니다.
NUMRIDS_DELETED[4]	BIGINT		Y	모든 ID가 삭제된 것으로 표시되는 리프 페이지의 ID를 제외하고, 삭제됨으로 표시되는 인덱스의 행 ID(또는 블록 ID)의 총수.
NUM_EMPTY_LEAFS	BIGINT		Y	모든 행 ID(또는 블록 ID)가 삭제됨으로 표시되는 인덱스 리프 페이지의 총수.
AVERAGE_RANDOM_ FETCH_PAGES[1,2,4]	DOUBLE		Y	인덱스를 사용하여 페치할 때 순차 페이지 액세스 사이의 무작위 테이블 페이지의 평균 수이며, 알 수 없는 경우 -1입니다.
AVERAGE_RANDOM_ PAGES[2]	DOUBLE		Y	순차 페이지 액세스 사이의 무작위 테이블 페이지의 평균 수이며, 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_ GAP[2]	DOUBLE		Y	인덱스 페이지 시퀀스 사이의 갭입니다. 인덱스 리프 페이지의 스캔을 통해 발견되는 각 갭은 인덱스 페이지의 시퀀스 사이에 무작위로 페치되어야 하는 평균 인덱스 페이지 수를 나타냅니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_ FETCH_GAP[1,2,4]	DOUBLE		Y	인덱스를 사용하여 페치할 때 테이블 페이지 시퀀스 사이의 갭입니다. 인덱스 리프 페이지의 스캔을 통해 발견되는 각 갭은 테이블 페이지의 시퀀스 사이에 무작위로 페치되어야 하는 평균 테이블 페이지 수를 나타냅니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_ PAGES[2]	DOUBLE		Y	순차적으로 액세스할 수 있는 인덱스 페이지의 평균 수(즉, 프리페처가 시퀀스에 있는 것으로 발견하는 인덱스 페이지 수)입니다. 알 수 없는 경우 -1입니다.
AVERAGE_SEQUENCE_ FETCH_PAGES[1,2,4]	DOUBLE		Y	인덱스를 사용하여 페치할 때 순차적으로 액세스할 수 있는 테이블 페이지의 평균 수(즉, 프리페처가 시퀀스에 있는 것으로 발견하는 테이블 페이지 수)입니다. 알 수 없는 경우 -1입니다.

표 213. SYSSTAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
AVGPARTITION_ CLUSTERRATIO[3,4]	SMALLINT		Y	단일 데이터 파티션 내의 데이터 클러스터링 등급입니다. 테이블이 파티션되지 않는 경우, 통계가 수집되지 않는 경우 또는 상세한 통계가 수집되는 경우(이 경우에는 AVGPARTITION_ CLUSTERFACTOR가 대신 사용됨) -1입니다.
AVGPARTITION_ CLUSTERFACTOR[3,4]	DOUBLE		Y	단일 데이터 파티션에서 클러스터링 등급의 상세한 측정입니다. 테이블이 파티션되지 않는 경우, 통계가 수집되지 않는 경우 또는 인덱스가 별칭에 대해 정의되는 경우 -1입니다.
AVGPARTITION_PAGE_ FETCH_PAIRS[3,4]	VARCHAR(520)		Y	문자 양식에서 쌍으로 된 정수 목록입니다. 각 쌍은 잠재적 버퍼 풀 크기 및 테이블에서 단일 데이터 파티션에 액세스하기 위해 필요한 대응하는 페이지 페치를 나타냅니다. 사용 가능한 데이터가 없는 경우 또는 테이블이 파티션되지 않는 경우 길이가 0인 문자열입니다.
DATAPARTITION_ CLUSTERFACTOR	DOUBLE		Y	데이터 파티션에 관하여 인덱스 키의 ″클러스터링″을 측정하는 통계입니다. 0과 1 사이의 숫자이며, 1은 완벽한 클러스터링을 나타내고 0은 클러스터링이 없음을 나타냅니다.
INDCARD	BIGINT		Y	인덱스의 카디널리티(cardinality)입니다. 테이블 행과 인덱스 항목 사이에 일대일 관계가 없는 인덱스의 경우 테이블의 카디널리티(cardinality)와 다를 수 있습니다.
PCTPAGESSAVED	SMALLINT			인덱스 압축의 결과로 인덱스에 저장되는 페이지의 대략적인 백분율입니다. 통계가 수집되지 않은 경우 -1입니다.
AVGLEAFKEYSIZE	INTEGER		Y	인덱스에 있는 리프 페이지의 키에 대한 평균 인덱스 키 크기.
AVGNLEAFKEYSIZE	INTEGER		Y	인덱스에 있는 리프가 아닌 페이지의 키에 대한 평균 인덱스 키 크기.

주:

1. DMS 테이블 스페이스를 사용할 때 이 통계를 계산할 수 없습니다.
2. LOAD...STATISTICS YES 또는 CREATE INDEX...COLLECT STATISTICS 조작 중 또는 데이터베이스 구성 매개변수 *seqdetect*가 꺼졌을 때 프리페치 통계가 수집되지 않습니다.
3. AVGPARTITION_CLUSTERRATIO, AVGPARTITION_CLUSTERFACTOR 및 AVGPARTITION_PAGE_FETCH_PAIRS가 단일 데이터 파티션에서 클러스터링 등급을 측정합니다(로컬 클러스터링). CLUSTERRATIO, CLUSTERFACTOR 및 PAGE_FETCH_PAIRS는 전체 테이블에서 클러스터링 등급을 측정합니다(전역 클러스터링). 테이블 파티셔닝 키가 인덱스 키의 접두부가 아닌 경우 또는 테이블 파티셔닝 키 및 인덱스 키가 서로 논리적으로 독립일 때 전역 클러스터링 및 로컬 클러스터링이 크게 달라질 수 있습니다.
4. 인덱스 유형이 'XPTH'(XML 경로 인덱스)인 경우 이 통계를 갱신할 수 없습니다.
5. XML 컬럼에 대한 논리적 인덱스에는 통계가 없기 때문에 SYSSTAT.INDEXES 카탈로그 뷰는 인덱스 유형이 'XVIL'인 행을 제외합니다.

SYSSTAT.ROUTINES

각 행은 사용자 정의 루틴(스칼라 함수, 테이블 함수, 전래 함수, 메소드 또는 프로시저)을 나타냅니다. 내장 함수를 포함하지 않습니다.

표 214. SYSSTAT.ROUTINES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
ROUTINESCHEMA	VARCHAR(128)			ROUTINEMODULENAME이 널(NULL)인 경우 루틴의 스키마 이름이고 그 이외의 경우에는 루틴이 속한 모듈의 스키마 이름입니다.
ROUTINEMODULENAME	VARCHAR(128)			루틴이 속하는 모듈의 규정되지 않은 이름. 모듈 루틴이 아니면 널(NULL) 값입니다.
ROUTINENAME	VARCHAR(128)			루틴의 규정되지 않은 이름.
ROUTINETYPE	CHAR(1)			루틴의 유형. • F = 함수 • M = 메소드 • P = 프로시저
SPECIFICNAME	VARCHAR(128)			루틴 인스턴스의 이름(시스템이 생성했을 수 있음).
IOS_PER_INVOC	DOUBLE		Y	개략적인 호출당 입출력(I/O) 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
INSTS_PER_INVOC	DOUBLE		Y	개략적인 호출당 명령어 수입니다. 디폴트는 450이고, 알 수 없는 경우 -1입니다.
IOS_PER_ARGBYTE	DOUBLE		Y	개략적인 입력 인수 바이트당 I/O 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
INSTS_PER_ARGBYTE	DOUBLE		Y	개략적인 입력 인수 바이트당 명령어 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
PERCENT_ARGBYTES	SMALLINT		Y	루틴이 실제로 읽을 입력 인수 바이트의 개략적인 평균 퍼센트입니다. 디폴트는 100이며, 알 수 없는 경우 -1입니다.
INITIAL_IOS	DOUBLE		Y	처음 루틴이 호출될 때 수행되는 I/O의 개략적인 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
INITIAL_INSTS	DOUBLE		Y	처음 루틴이 호출될 때 실행되는 명령어의 개략적인 수입니다. 디폴트는 0이고, 알 수 없는 경우 -1입니다.
CARDINALITY	BIGINT		Y	테이블 함수의 예상 카디널리티(cardinality)입니다. 알 수 없는 경우 또는 루틴이 테이블 함수가 아닌 경우 -1입니다.
SELECTIVITY	DOUBLE		Y	사용자 정의 술어의 경우입니다. 사용자 정의 술어가 없는 경우 -1입니다.

SYSSTAT.TABLES

각 행은 테이블, 뷰, 별명 또는 별칭을 나타냅니다. 각 테이블 또는 뷰 계층 구조에는 계층 구조를 구현하는 계층 구조 테이블 또는 계층 구조 뷰를 나타내는 추가 행이 하나 더 있습니다. 카탈로그 테이블 및 뷰가 포함됩니다.

표 215. SYSSTAT.TABLES 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	갱신 가능	설명
TABSCHEMA	VARCHAR(128)			오브젝트의 스키마 이름.
TABNAME	VARCHAR(128)			오브젝트의 규정되지 않은 이름.
CARD	BIGINT		Y	테이블에 있는 총 행 수이며, 통계가 수집되지 않는 경우 -1입니다.
NPAGES	BIGINT		Y	테이블의 행이 존재하는 페이지의 총수입니다. 뷰 또는 별명의 경우 또는 통계가 수집되지 않는 경우 -1이고, 서브테이블 또는 계층 구조 테이블의 경우 -2입니다.
FPAGES	BIGINT		Y	페이지의 총수입니다. 뷰 또는 별명의 경우 또는 통계가 수집되지 않는 경우 -1이고, 서브테이블 또는 계층 구조 테이블의 경우 -2입니다.
OVERFLOW	BIGINT		Y	테이블에 있는 오버플로우 레코드의 총수입니다. 뷰 또는 별명의 경우 또는 통계가 수집되지 않는 경우 -1이고, 서브테이블 또는 계층 구조 테이블의 경우 -2입니다.
CLUSTERED	CHAR(1)	Y		• Y = 테이블이 다차원으로 클러스터됨(한 차원만으로도 다차원으로 클러스터됨) • 널(NULL) 값 = 테이블이 다차원으로 클러스터되지 않음
ACTIVE_BLOCKS	BIGINT		Y	테이블의 활성 블록의 총수 또는 -1입니다. 다차원적으로 클러스터된(MDC) 테이블에만 적용됩니다.
AVGCOMPRESSEDROWSIZE	SMALLINT		Y	이 테이블에 있는 압축된 행의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
AVGROWCOMPRESSIONRATIO	REAL		Y	테이블의 압축된 행의 경우 행별로 평균 압축비입니다. 즉, 평균 압축되지 않은 행 길이를 평균 압축 행 길이로 나눈 값입니다. 통계가 수집되지 않는 경우 -1입니다.
AVGROWSIZE	SMALLINT			이 테이블에 있는 압축 및 압축되지 않은 행 모두의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
PCTROWSCOMPRESSED	REAL		Y	테이블에 있는 전체 행 수의 백분율로 표현되는 압축된 행입니다. 통계가 수집되지 않는 경우 -1입니다.
PCTPAGESSAVED	SMALLINT		Y	행 압축의 결과로 테이블에 저장되는 페이지의 대략적인 백분율입니다. 이 값에는 테이블의 각 사용자 데이터에 대한 오버헤드 바이트가 포함되지만 사전 오버헤드가 이용하는 스페이스는 포함되지 않습니다. 통계가 수집되지 않는 경우 -1입니다.

부록 E. 페더레이티드 시스템

SQL문의 유효한 서버 유형

서버 유형은 서버 정의가 나타내는 데이터 소스의 종류를 표시합니다.

서버 유형은 벤더, 용도 및 운영 체제에 따라 다릅니다. 지원되는 값은 데이터 소스에 따라 다릅니다.

대부분의 데이터 소스의 경우, CREATE SERVER문에서 유효한 서버 유형을 지정해야 합니다.

표 216. 데이터 소스 및 서버 유형

데이터 소스	서버 유형
BioRS	CREATE SERVER문에 서버 유형이 필요하지 않습니다.
Excel	CREATE SERVER문에 서버 유형이 필요하지 않습니다.
Linux, UNIX 및 Windows용 IBM DB2 Universal Database	DB2/UDB
System i 및 AS/400®용 IBM DB2 Universal Database	DB2/ISERIES
z/OS용 IBM DB2 Universal Database	DB2/ZOS
VM용 IBM DB2	DB2/VM
Informix	INFORMIX
JDBC	JDBC(JDBC 드라이버 3.0 이상에서 지원하는 JDBC 데이터 소스에 필수)
Microsoft SQL Server	MSSQLSERVER(DataDirect Connect ODBC 4.2 이상의 드라이버 또는 Microsoft SQL Server ODBC 3.0 이상의 드라이버에서 지원하는 데이터 소스에 필수)
ODBC	ODBC(ODBC 3.x 드라이버에서 지원하는 ODBC 데이터 소스에 필수)
OLE DB	CREATE SERVER문에 서버 유형이 필요하지 않습니다.
Oracle	ORACLE(Oracle NET8 클라이언트 소프트웨어에서 지원하는 Oracle 데이터 소스에 필수)
Sybase(CTLIB)	SYBASE
테이블 구조 파일	CREATE SERVER문에 서버 유형이 필요하지 않습니다.
Teradata	TERADATA
웹 서비스	CREATE SERVER문에 서버 유형이 필요하지 않습니다.
XML	CREATE SERVER문에 서버 유형이 필요하지 않습니다.

페더레이티드 시스템의 함수 맵핑 옵션

페더레이티드 서버는 DB2 함수와 데이터 소스 함수 간의 디폴트 맵핑을 제공합니다. 대부분의 데이터 소스의 경우 디폴트 함수형 맵핑은 랩퍼에 있습니다. 페더레이티드 서버가 인식하지 않는 데이터 소스 함수를 사용하거나 디폴트 맵핑을 변경하려면 함수 맵핑을 작성하십시오.

함수 맵핑 작성 시 데이터 소스 함수의 이름을 지정하고 맵핑된 함수를 사용 가능하게 설정해야 합니다. 그런 다음 맵핑된 함수를 사용할 때 쿼리 옵티마이저가 데이터 소스에서 함수를 실행하는 경우의 비용과 페더레이티드 서버에서 함수를 실행하는 경우의 비용을 비교합니다.

표 217. 기능 맵핑의 옵션

이름	설명
DISABLE	디폴트 함수 맵핑 사용 가능 또는 사용 불가능으로 설정합니다. 유효한 값은 Y 및 N입니다. 디폴트값은 N입니다.
REMOTE_NAME	데이터 소스 함수의 이름입니다. 디폴트값은 로컬 이름입니다.

디폴트 포워드 데이터 유형 맵핑

데이터 소스 데이터 유형과 페더레이티드 데이터베이스 데이터 유형 사이에는 포워드 유형 맵핑 및 역방향 유형 맵핑이라는 두 종류의 맵핑이 있습니다. 포워드 유형 맵핑의 경우 리모트 유형에서 유형에서 비교 가능한 로컬 유형으로 맵핑이 이루어집니다.

디폴트 유형 맵핑을 겹쳐쓰거나 CREATE TYPE MAPPING문을 사용하여 새 유형 맵핑을 작성할 수 있습니다.

이러한 맵핑은 달리 언급되지 않는 한 지원되는 모든 버전에서 유효합니다.

데이터 소스에서 페더레이티드 데이터베이스로 수행되는 모든 디폴트 포워드 데이터 유형 맵핑에서 페더레이티드 스키마는 SYSIBM입니다.

다음 표에는 페더레이티드 데이터베이스 데이터 유형과 데이터 소스 데이터 유형 간의 디폴트 포워드 맵핑이 표시되어 있습니다.

Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 218. Linux, UNIX 및 Windows용 DB2 데이터베이스 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DECFLOAT[2]	-	-	-	-	-	-	DECFLOAT	-	0	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP(*p*)	-	*p*	*p*	-	-	TIMESTAMP(*p*)	-	*p*	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

주:

1. date_compat 구성 매개변수가 ON으로 설정된 경우 페더레이티드 유형은 TIMESTAMP(0)입니다.
2. 디폴트로 SAME_DECFLT_ROUNDING 서버 옵션은 N으로 설정되며 SAME_DECFLT_ROUNDING을 Y로 설정하지 않는 한 리모트 데이터 소스에 조작을 푸시다운하지 않습니다. SAME_DECFLT_ROUNDING 서버 옵션에 대한 자세한 정보는 DB2 데이터베이스 옵션 참조를 참조하십시오.

System i용 DB2 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 iSeries®용 DB2 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 219. System i용 DB2 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

VM 및 VSE용 DB2 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 VM 및 VSE용 DB2 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 220. VM 및 VSE용 DB2 Server 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

z/OS용 DB2 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 z/OS용 DB2 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 221. z/OS용 DB2 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Informix 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 Informix 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 222. Informix 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	0	-
DATETIME[2]	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP(6)	10	6	-
DATETIME	6	10	11	15	-	-	TIMESTAMP(6)	10	6	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
DECIMAL	1	32	255	255	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

주:

1. date_compat 구성 매개변수가 ON으로 설정된 경우 페더레이티드 유형은 TIMESTAMP(0)입니다.
2. Informix DATETIME 데이터 유형과 관련하여 DB2 UNIX 및 Windows 페더레이티드 서버는 Informix 상위 레벨 규정자를 REMOTE_LENGTH로 사용하고 Informix 하위 레벨 규정자를 REMOTE_SCALE로 사용합니다.

Informix 규정자는 Informix Client SDK `datatime.h` 파일에 정의된 "TU_" 상수입니다. 상수는 다음과 같습니다.

Informix 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

표 222. Informix 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음) (계속)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
0 = YEAR		8 = MINUTE					13 = FRACTION(3)			
2 = MONTH		10 = SECOND					14 = FRACTION(4)			
4 = DAY		11 = FRACTION(1)					15 = FRACTION(5)			
6 = HOUR		12 = FRACTION(2)								

Microsoft SQL Server 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 Microsoft SQL Server 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 223. Microsoft SQL Server 디폴트 포워드 데이터 유형 맵핑

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
bigint [1]	-	-	-	-	-	-	BIGINT	-	-	-
2진	1	254	-	-	-	-	CHARACTER	-	-	Y
2진	255	8000	-	-	-	-	VARCHAR	-	-	Y
비트	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
날짜 시간	-	-	-	-	-	-	TIMESTAMP(6)	10	6	-
10진수	1	31	0	31	-	-	DECIMAL	-	-	-
10진수	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
이미지	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
통화	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
수치	1	31	0	31	-	-	DECIMAL	-	-	-
수치	32	38	0	38	-	-	DOUBLE	8	-	-
ntext	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP(6)	10	6	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID	-	-	-	-	-	-	VARCHAR	-	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N

표 223. Microsoft SQL Server 디폴트 포워드 데이터 유형 맵핑 (계속)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	6	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	254	-	-	-	-	CHARACTER	-	-	N
SQL_WCHAR	255	8800	-	-	-	-	VARCHAR	-	-	N
SQL_ WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
텍스트	-	-	-	-	-	-	CLOB	-	-	N
시간소인	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

주:

1. 이러한 유형 맵핑은 Microsoft SQL Server 버전 2000에서만 유효합니다.

ODBC 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 ODBC 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 224. ODBC 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	-	-	-
SQL_DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	8	-	-	-	-	FLOAT	8	-	-
SQL_FLOAT	-	4	-	-	-	-	FLOAT	4	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	2147483647	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIMESTAMP(*p*)	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

주:

1. date_compat 구성 매개변수가 ON으로 설정된 경우 페더레이티드 유형은 TIMESTAMP(0)입니다.

Oracle NET8 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 Oracle NET8 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 225. Oracle NET8 디폴트 포워드f 데이터 유형 맵핑

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BLOB	0	0	0	0	-	₩0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	₩0	CHAR	0	0	N
CHAR	255	2000	0	0	-	₩0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	₩0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	₩0	TIMESTAMP(6)	0	0	N
FLOAT	1	126	0	0	-	₩0	DOUBLE	0	0	N
LONG	0	0	0	0	-	₩0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	₩0	BLOB	2147483647	0	Y
NUMBER	10	18	0	0	-	₩0	BIGINT	0	0	N
NUMBER	1	38	-84	127	-	₩0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	₩0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	₩0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	₩0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	₩0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	₩0	CHAR	18	0	N
TIMESTAMP(*p*) [1]	-	-	-	-	-	₩0	TIMESTAMP(6)	10	6	N
VARCHAR2	1	4000	0	0	-	₩0	VARCHAR	0	0	N

주:

1.

- TIMESTAMP*(p)*는 시간소인을 나타내며 변수 스케일은 0-9입니다. Oracle 시간소인의 스케일은 디폴트로 TIMESTAMP(6)에 맵핑됩니다. 이 디폴트 유형 맵핑을 변경할 수 있으며 사용자 정의 유형 맵핑을 사용하여 Oracle TIMESTAMP를 동일한 스케일의 페더레이티드 TIMESTAMP에 맵핑할 수 있습니다.
- 이러한 유형 맵핑은 Oracle 9i 이상 클라이언트 및 서버 구성에서만 유효합니다.

Sybase 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 Sybase 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 226. Sybase CTLIB 디폴트 포워드 데이터 유형 맵핑

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
2진	1	254	-	-	-	-	CHAR	-	-	Y
2진	255	32672	-	-	-	-	VARCHAR	-	-	Y
비트	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	32672	-	-	-	-	VARCHAR	-	-	N
char null(varchar 참조)										
날짜 시간	-	-	-	-	-	-	DATE	-	-	-
날짜 시간	-	-	-	-	-	-	TIMESTAMP[1]	-	-	-
날짜 시간	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
10진수	1	31	0	31	-	-	DECIMAL	-	-	-
10진수	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
이미지	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
통화	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	32672	-	-	-	-	VARCHAR	-	-	N
n c h a r null(nvarchar 참조)										
수치	1	31	0	31	-	-	DECIMAL	-	-	-
수치	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	32672	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	-	-	-	-	-	-	VARCHAR	30	-	N

표 226. Sybase CTLIB 디폴트 포워드 데이터 유형 맵핑 (계속)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
텍스트	-	-	-	-	-	-	CLOB	-	-	-
시간	-	-	-	-	-	-	TIME	-	-	-
시간소인	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[2]	1	254	-	-	-	-	CHAR	-	-	N
unichar[2]	255	32672	-	-	-	-	VARCHAR	-	-	N
unichar null(univarchar 참조)										
univarchar[2]	1	32672	-	-	-	-	VARCHAR	-	-	N
varbinary	1	32672	-	-	-	-	VARCHAR	-	-	Y
varchar	1	32672	-	-	-	-	VARCHAR	-	-	N

주:

1. date_compat 구성 매개변수가 ON으로 설정된 경우 페더레이티드 유형은 TIMESTAMP(0)입니다.
2. 유니코드가 아닌 페더레이티드 데이터베이스에서 유효합니다.

Teradata 데이터 소스의 디폴트 포워드 데이터 유형 맵핑

다음 표에는 Teradata 데이터 소스의 디폴트 포워드 데이터 유형 맵핑이 나열되어 있습니다.

표 227. Teradata 디폴트 포워드 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

리모트 유형 이름	리모트 하위 길이	리모트 상위 길이	리모트 하위 스케일	리모트 상위 스케일	리모트 비트 데이터	리모트 데이터 연산자	페더레이티드 유형 이름	페더레이티드 길이	페더레이티드 스케일	페더레이티드 비트 데이터
BLOB	1	2097088000	-	-	-	-	BLOB	-	-	-
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
CLOB	1	2097088000 (라틴어)	-	-	-	-	CLOB	-	-	-
CLOB	1	1048544000 (유니코드)	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	0	21	0	21	-	-	TIME	-	-	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(6)	10	6	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

주:

1. date_compat 구성 매개변수가 ON으로 설정된 경우 페더레이티드 유형은 TIMESTAMP(0)입니다.

디폴트 역방향 데이터 유형 맵핑

대부분의 데이터 소스의 경우 디폴트 유형 맵핑은 랩퍼에 있습니다.

데이터 소스 데이터 유형과 페더레이티드 데이터베이스 데이터 유형 사이에는 포워드 유형 맵핑 및 역방향 유형 맵핑이라는 두 종류의 맵핑이 있습니다. 포워드 유형 맵핑의 경우 리모트 유형에서 유형에서 비교 가능한 로컬 유형으로 맵핑이 이루어집니다. 다른 유형의 맵핑은 역방향 유형 맵핑으로 투명한 DDL과 함께 리모트 테이블을 작성하거나 수정하는 데 사용됩니다.

DB2 제품군 데이터 소스의 디폴트 유형 맵핑은 DRDA 랩퍼에 있습니다. Informix의 디폴트 유형 맵핑은 INFORMIX 랩퍼 등에 있습니다.

리모트 테이블 또는 뷰를 페더레이티드 데이터베이스에 정의하면 정의에 역방향 유형 맵핑이 포함됩니다. 맵핑은 각 컬럼의 로컬 페더레이티드 데이터베이스 데이터 유형 및 해당 리모트 데이터 유형에서 수행됩니다. 예를 들면, 로컬 유형 REAL 포인트에서 Informix 유형 SMALLFLOAT로 이루어지는 디폴트 역방향 유형 맵핑이 있습니다.

페더레이티드 데이터베이스는 LONG VARCHAR, LONG VARGRAPHIC 및 사용자 정의 유형에 대해 맵핑을 지원하지 않습니다.

CREATE TABLE문을 사용하여 리모트 테이블을 작성하는 경우 리모트 테이블에 포함시킬 로컬 데이터 유형을 지정하십시오. 이러한 디폴트 역방향 유형 맵핑은 상응하는 리모트 유형을 해당 컬럼에 지정합니다. 예를 들어, CREATE TABLE문을 사용하여 C2 컬럼이 있는 Informix 테이블을 정의한다고 가정합니다. 명령문에서 BIGINT를 C2의 데이터 유형으로 지정합니다. BIGINT의 디폴트 역방향 유형 맵핑은 테이블을 작성 중인 Informix의 버전에 따라 다릅니다. Informix 테이블의 C2는 Informix 버전 8의 DECIMAL 및 Informix 버전 9의 INT8로 맵핑됩니다.

디폴트 역방향 유형 맵핑을 겹쳐쓰거나 CREATE TYPE MAPPING문을 사용하여 새 역방향 유형 맵핑을 작성할 수 있습니다.

다음 표에는 페더레이티드 데이터베이스 로컬 데이터 유형과 리모트 데이터 소스 데이터 유형 간의 디폴트 역방향 맵핑이 표시되어 있습니다.

이러한 맵핑은 달리 언급되지 않는 한 지원되는 모든 버전에서 유효합니다.

Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 228. Linux, UNIX 및 Windows용 DB2 데이터베이스 디폴트 역방향 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	페더레이티드 비트 데이터
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DECFLOAT [1]	-	8	-	-	-	-	DECFLOAT	-	0	-
DECFLOAT [1]	-	16	-	-	-	-	DECFLOAT	-	0	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(*p*)	-	*p*	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

주:

1. 디폴트로 SAME_DECFLT_ROUNDING 서버 옵션은 N으로 설정되며 SAME_DECFLT_ROUNDING을 Y로 설정하지 않는 한 리모트 데이터 소스에 조작을 푸시다운하지 않습니다. SAME_DECFLT_ROUNDING 서버 옵션에 대한 자세한 정보는 DB2 데이터베이스 옵션 참조를 참조하십시오.

System i용 DB2 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 System i용 DB2 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 229. System i용 DB2 디폴트 역방향 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

VM 및 VSE용 DB2 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 VM 및 VSE용 DB2 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 230. VM 및 VSE용 DB2 디폴트 역방향 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

z/OS용 DB2 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 z/OS용 DB2 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 231. z/OS용 DB2 디폴트 역방향 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Informix 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 Informix 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 232. Informix 디폴트 역방향 데이터 유형 맵핑

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR [1]	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

주:

1. 이러한 유형 맵핑은 Informix 서버 버전 8 이하에서만 유효합니다.
2. 이러한 유형 맵핑은 Informix 서버 버전 9 이상에서만 유효합니다.

Informix DATETIME 데이터 유형과 관련하여 페더레이티드 서버는 Informix 상위 레벨 규정자를 REMOTE_LENGTH로 사용하고 Informix 하위 레벨 규정자를 REMOTE_SCALE로 사용합니다.

Informix 규정자는 Informix Client SDK `datatime.h` 파일에 정의된 "TU_" 상수입니다. 상수는 다음과 같습니다.

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 Microsoft SQL Server 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 233. Microsoft SQL Server 디폴트 역방향 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BIGINT [1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	이미지	-	-	-
CHARACTER	-	-	-	-	Y	-	2진	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	텍스트	-	-	-
DATE	-	4	-	-	-	-	날짜 시간	-	-	-
DECIMAL	-	-	-	-	-	-	10진수	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	날짜 시간	-	-	-
TIMESTAMP	-	10	-	-	-	-	날짜 시간	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	텍스트	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	이미지	-	-	-

주:

1. 이러한 유형 맵핑은 Microsoft SQL Server 버전 2000에서만 유효합니다.

Oracle NET8 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 Oracle NET8 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 234. Oracle NET8 디폴트 역방향 데이터 유형 맵핑

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BIGINT	0	8	0	0	N	₩0	NUMBER	19	0	N
BLOB	0	2147483647	0	0	Y	₩0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	₩0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	₩0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	₩0	CLOB	0	0	N
DATE[1]	0	4	0	0	N	₩0	DATE	0	0	N
DECIMAL	0	0	0	0	N	₩0	NUMBER	0	0	N
DECFLOAT	0	8	0	0	N	₩0	NUMBER	0	0	N
DECFLOAT	0	16	0	0	N	₩0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	₩0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	₩0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	₩0	NUMBER	10	0	N
REAL	0	4	0	0	N	₩0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	₩0	NUMBER	5	0	N
TIME	0	3	0	0	N	₩0	DATE	0	0	N
TIMESTAMP [2]	0	10	0	0	N	₩0	DATE	0	0	N
TIMESTAMP*(p)* [3]	-	-	-	-	N	₩0	TIMESTAMP*(p)*	-	-	N
VARCHAR	1	4000	0	0	N	₩0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	₩0	RAW	0	0	Y

주:

1. date_compat 매개변수가 OFF로 설정되면 페더레이티드 DATE가 Oracle 날짜에 맵핑됩니다. date_compat 매개변수가 ON으로 설정되면 페더레이티드 DATE(TIMESTAMP(0)에 해당)가 Oracle TIMESTAMP(0)에 맵핑됩니다.
2. 이러한 유형 맵핑은 Oracle 버전 8에서만 유효합니다.
3.
   - TIMESTAMP*(p)*는 시간소인을 나타내며 변수 스케일이 Oracle의 경우 0-9이고 페더레이션의 경우 0-12입니다. 스케일이 0-9인 경우 리모트 Oracle TIMESTAMP의 스케일은 페더레이티드 TIMESTAMP와 동일합니다. 페더레이티드 TIMESTAMP의 스케일이 9보다 큰 경우 이에 해당하는 Oracle TIMESTAMP 스케일은 가장 큰 Oracle 스케일인 9입니다.
   - 이러한 유형 맵핑은 Oracle 버전 9, 10 및 11에서만 유효합니다.

Sybase 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 Sybase 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 235. Sybase CTLIB 디폴트 역방향 데이터 유형 맵핑

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BIGINT	-	-	-	-	-	-	10진수	19	0	-
BLOB	-	-	-	-	-	-	이미지	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	2진	-	-	-
CLOB	-	-	-	-	-	-	텍스트	-	-	-
DATE	-	-	-	-	-	-	날짜 시간	-	-	-
DECIMAL	-	-	-	-	-	-	10진수	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	날짜 시간	-	-	-
TIMESTAMP	-	-	-	-	-	-	날짜 시간	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	텍스트	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	텍스트	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	이미지	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	이미지	-	-	-

주:

1. 이러한 유형 맵핑은 Sybase 서버 버전 12.0 이전을 사용하는 CTLIB에서만 유효합니다.
2. 이러한 유형 맵핑은 Sybase 서버 버전 12.5 이상을 사용하는 CTLIB에서만 유효합니다.

Teradata 데이터 소스의 디폴트 역방향 데이터 유형 맵핑

다음 표에는 Teradata 데이터 소스의 디폴트 역방향 데이터 유형 맵핑이 나열되어 있습니다.

표 236. Teradata 디폴트 역방향 데이터 유형 맵핑(일부 컬럼은 표시되지 않음)

페더레이티드 유형 이름	페더레이티드 하위 길이	페더레이티드 상위 길이	페더레이티드 하위 스케일	페더레이티드 상위 스케일	페더레이티드 비트 데이터	페더레이티드 데이터 연산자	리모트 유형 이름	리모트 길이	리모트 스케일	리모트 비트 데이터
BLOB	1	2097088000	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2097088000	-	-		-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DBCLOB [1]	1	64000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	8	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	8	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	15	-	-
TIMESTAMP	10	10	6	6	-	-	TIMESTAMP	26	6	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

주:

1. Teradata VARGRAPHIC 데이터 유형에는 지정된 길이(1 - 32000)의 DBCLOB 데이터 유형만 포함될 수 있습니다.

부록 F. SAMPLE 데이터베이스

응용프로그램 테스트, DB2의 여러 기능 사용 등과 같이 여러 용도로 샘플 데이터베이스를 사용할 수 있습니다. DB2PATH/sqllib/samples에 있는 대부분의 샘플 응용프로그램에서는 여러 DB2 기능을 시연하는 데 샘플 데이터베이스를 사용하여 쉽게 기술을 이해할 수 있도록 합니다.

샘플 데이터베이스가 작성되면 다음 사항이 작성된 것을 확인할 수 있습니다.

- XML이 아닌 데이터의 조직 스키마
- XML 데이터의 구입 주문 스키마

이들 스키마의 데이터 및 데이터베이스 오브젝트는 실시간 환경에서 작은 스케일로 작성됩니다.

다음은 SAMPLE 데이터베이스의 각 테이블에 대한 설명입니다. 각 테이블의 초기 데이터 값이 주어져 있습니다. 대시(-)는 NULL 값을 표시합니다.

ACT 테이블

이름:	ACTNO	ACTKWD	ACTDESC
다음을 입력하십시오.			
	SMALLINT	CHAR(6)	VARCHAR(20)
값:	10	MANAGE	MANAGE/ADVISE
	20	ECOST	ESTIMATE COST
	30	DEFINE	DEFINE SPECS
	40	LEADPR	LEAD PROGRAM/DESIGN
	50	SPECS	WRITE SPECS
	60	LOGIC	DESCRIBE LOGIC
	70	CODE	CODE PROGRAMS
	80	TEST	TEST PROGRAMS
	90	ADMQS	ADM QUERY SYSTEM
	100	TEACH	TEACH CLASSES
	110	COURSE	DEVELOP COURSES
	120	STAFF	PERS AND STAFFING
	130	OPERAT	OPER COMPUTER SYS
	140	MAINT	MAINT SOFTWARE SYS
	150	ADMSYS	ADM OPERATING SYS
	160	ADMDB	ADM DATA BASES
	170	ADMDC	ADM DATA COMM
	180	DOC	DOCUMENT

ADEFUSER 테이블

이름:	WORKDEPT	NO_OF_EMPLOYEES
다음을 입력하십시오.	CHAR(3)	INTEGER
값:	A00	5
	B01	1
	C01	4
	D11	11
	D21	7
	E01	1
	E11	7
	E21	6

CL_SCHED 테이블

이름:	CLASS_CODE	DAY	STARTING	ENDING
다음을 입력하십시오.	CHAR(7)	SMALLINT	TIME	TIME
설명:	클래스 코드(영역:선생님)	4일 중 #번 째 날의 스케줄	클래스 시작 시간	클래스 종료 시간
값:	042:BF	4	12:10:00	14:00:00
	553:MJA	1	10:30:00	11:00:00
	543:CWM	3	09:10:00	10:30:00
	778:RES	2	12:10:00	14:00:00
	044:HD	3	17:12:30	18:00:00

DEPT 테이블

이름:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
다음을 입력하십시오.	CHAR(3)	VARCHAR(36)	CHAR(6)	CHAR(3)	CHAR(16)
값:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	
	B01	PLANNING	000020	A00	
	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	
	F22	BRANCH OFFICE F2		E01	
	G22	BRANCH OFFICE G2		E01	
	H22	BRANCH OFFICE H2		E01	

이름:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
	I22	BRANCH OFFICE I2		E01	
	J22	BRANCH OFFICE J2		E01	

DEPARTMENT 테이블

이름:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
다음을 입력하십시오.	CHAR(3) Not Null	VARCHAR(29) Not Null	CHAR(6)	CHAR(3) Not Null	CHAR(16)
설명:	부서 번호	부서의 일반 활동에 대해 설명하는 이름	부서 관리자의 직원 번호 (EMPNO)	해당 부서의 보고 대상 부서 (DEPTNO)	리모트 위치 이름
값:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	
	B01	PLANNING	000020	A00	
	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	
	F22	BRANCH OFFICE F2		E01	
	G22	BRANCH OFFICE G2		E01	
	H22	BRANCH OFFICE H2		E01	
	I22	BRANCH OFFICE I2		E01	
	J22	BRANCH OFFICE J2		E01	

EMPLOYEE 및 EMP 테이블

이 두 테이블의 컨텐츠는 동일합니다.

이름:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
다음을 입력하십시오.	CHAR(6) Not Null	VARCHAR(12) Not Null	CHAR(1) Not Null	VARCHAR(15) Not Null	CHAR(3)	CHAR(4)	DATE
설명:	직원 번호	이름	중간 머릿글자	성	해당 직원이 근무하는 부서 (DEPTNO)	전화번호	고용 날짜

+

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
CHAR(8)	SMALLINT Not Null	CHAR(1)	DATE	DECIMAL(9,2)	DECIMAL(9,2)	DECIMAL(9,2)
작업	공식 교육받은 연 수	성별(M 남성, F 여성)	생년월일	연봉	연간 상여금	연간 커미션

다음 테이블에는 EMPLOYEE 테이블의 값이 들어 있습니다.

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
CHAR(6) Not Null	VARCHAR(12) Not Null	CHAR(1) Not Null	VARCHAR(15) Not Null	CHAR(3)	CHAR(4)	DATE	CHAR(8)	SMALLINT Not Null	CHAR(1)	DATE	DECIMAL (9,2)	DECIMAL (9,2)	DECIMAL (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
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
다음을 입력하십시오.	CHAR(6) Not Null	CHAR(6) Not Null	SMALLINT Not Null	DEC(5,2)	DATE	DATE
설명:	직원 번호	프로젝트 번호	활동 번호	프로젝트에 소요되는 직원의 시간 비율	활동 시작 날짜	활동 종료 날짜
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

이름:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	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
	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
	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
다음을 입력하십시오.	CHAR(6) Not Null	VARCHAR(10) Not Null	BLOB(100K)
설명:	직원 번호	사진 형식	직원 사진
값:	000130	비트맵	db200130.bmp
	000130	gif	db200130.gif
	000140	비트맵	db200140.bmp
	000140	gif	db200140.gif
	000150	비트맵	db200150.bmp
	000150	gif	db200150.gif

이름:	EMPNO	PHOTO_FORMAT	PICTURE
	000190	비트맵	db200190.bmp
	000190	gif	db200190.gif

EMPPROJACT 테이블

이름:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
다음을 입력하십시오.	CHAR(6)	CHAR(6)	SMALLINT	DEC(5,2)	DATE	DATE
값:	000070	AD3110	10	1.00	01/01/1982	02/01/1983
	000230	AD3111	60	1.00	01/01/1982	03/15/1982
	000230	AD3111	60	0.50	03/15/1982	04/15/1982
	000230	AD3111	70	0.50	03/15/1982	10/15/1982
	000230	AD3111	80	0.50	04/15/1982	10/15/1982
	000230	AD3111	180	0.50	10/15/1982	01/01/1983
	000240	AD3111	70	1.00	02/15/1982	09/15/1982
	000240	AD3111	80	1.00	09/15/1982	01/01/1983
	000250	AD3112	60	1.00	01/01/1982	02/01/1982
	000250	AD3112	60	0.50	02/01/1982	03/15/1982
	000250	AD3112	60	1.00	01/01/1983	02/01/1983
	000250	AD3112	70	0.50	02/01/1982	03/15/1982
	000250	AD3112	70	1.00	03/15/1982	08/15/1982
	000250	AD3112	70	0.25	08/15/1982	10/15/1982
	000250	AD3112	80	0.25	08/15/1982	10/15/1982
	000250	AD3112	80	0.50	10/15/1982	12/01/1982
	000250	AD3112	180	0.50	08/15/1982	01/01/1983
	000260	AD3113	70	0.50	06/15/1982	07/01/1982
	000260	AD3113	70	1.00	07/01/1982	02/01/1983
	000260	AD3113	80	1.00	01/01/1982	03/01/1982
	000260	AD3113	80	0.50	03/01/1982	04/15/1982
	000260	AD3113	180	0.50	03/01/1982	04/15/1982
	000260	AD3113	180	1.00	04/15/1982	06/01/1982
	000260	AD3113	180	1.00	06/01/1982	07/01/1982
	000270	AD3113	60	0.50	03/01/1982	04/01/1982
	000270	AD3113	60	1.00	04/01/1982	09/01/1982
	000270	AD3113	60	0.25	09/01/1982	10/15/1982
	000270	AD3113	70	0.75	09/01/1982	10/15/1982
	000270	AD3113	70	1.00	10/15/1982	02/01/1983
	000270	AD3113	80	1.00	01/01/1982	03/01/1982
	000270	AD3113	80	0.50	03/01/1982	04/01/1982
	000030	IF1000	10	0.50	06/01/1982	01/01/1983
	000130	IF1000	90	1.00	10/01/1982	01/01/1983

이름:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000130	IF1000	100	0.50	10/01/1982	01/01/1983
	000140	IF1000	90	0.50	10/01/1982	01/01/1983
	000030	IF2000	10	0.50	01/01/1982	01/01/1983
	000140	IF2000	100	1.00	01/01/1982	03/01/1982
	000140	IF2000	100	0.50	03/01/1982	07/01/1982
	000140	IF2000	110	0.50	03/01/1982	07/01/1982
	000140	IF2000	110	0.50	10/01/1982	01/01/1983
	000010	MA2100	10	0.50	01/01/1982	11/01/1982
	000110	MA2100	20	1.00	01/01/1982	03/01/1983
	000010	MA2110	10	1.00	01/01/1982	02/01/1983
	000200	MA2111	50	1.00	01/01/1982	06/15/1982
	000200	MA2111	60	1.00	06/15/1982	02/01/1983
	000220	MA2111	40	1.00	01/01/1982	02/01/1983
	000150	MA2112	60	1.00	01/01/1982	07/15/1982
	000150	MA2112	180	1.00	07/15/1982	02/01/1983
	000170	MA2112	60	1.00	01/01/1982	06/01/1983
	000170	MA2112	70	1.00	06/01/1982	02/01/1983
	000190	MA2112	70	1.00	01/01/1982	10/01/1982
	000190	MA2112	80	1.00	10/01/1982	10/01/1982
	000160	MA2113	60	1.00	07/15/1982	02/01/1983
	000170	MA2113	80	1.00	01/01/1982	02/01/1983
	000180	MA2113	70	1.00	04/01/1982	06/15/1982
	000210	MA2113	80	0.50	10/01/1982	02/01/1983
	000210	MA2113	180	0.50	10/01/1982	02/01/1983
	000050	OP1000	10	0.25	01/01/1982	02/01/1983
	000090	OP1010	10	1.00	01/01/1982	02/01/1983
	000280	OP1010	130	1.00	01/01/1982	02/01/1983
	000290	OP1010	130	1.00	01/01/1982	02/01/1983
	000300	OP1010	130	1.00	01/01/1982	02/01/1983
	000310	OP1010	130	1.00	01/01/1982	02/01/1983
	000050	OP1010	10	0.75	01/01/1982	02/01/1983
	000100	OP1010	10	1.00	01/01/1982	02/01/1983
	000320	OP2011	140	0.75	01/01/1982	02/01/1983
	000320	OP2011	150	0.25	01/01/1982	02/01/1983
	000330	OP2012	140	0.25	01/01/1982	02/01/1983
	000330	OP2012	160	0.75	01/01/1982	02/01/1983
	000340	OP2013	140	0.50	01/01/1982	02/01/1983
	000340	OP2013	170	0.50	01/01/1982	02/01/1983
	000020	PL2100	30	1.00	01/01/1982	09/15/1982

EMP_RESUME 테이블

이름:	EMPNO	RESUME_FORMAT	RESUME
다음을 입력하십시오.	CHAR(6) Not Null	VARCHAR(10) Not Null	CLOB(5K)
설명:	직원 번호	이력서 형식	직원 이력서
값:	000130	ascii	db200130.asc
	000130	html	db200130.htm
	000140	ascii	db200140.asc
	000140	html	db200140.htm
	000150	ascii	db200150.asc
	000150	html	db200150.htm
	000190	ascii	db200190.asc
	000190	html	db200190.htm

IN_TRAY 테이블

이름:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
다음을 입력하십시오.	TIMESTAMP	CHAR(8)	CHAR(64)	VARCHAR(3000)
설명:	수신 날짜 및 시간	메모를 보내는 사람의 사용자 ID	간단한 설명	메모
	1988-12-25-17.12.30.000000	BADAMSON	전달: 근사한 한 해입니다! 4분기 상여금.	수신: JWALKER 참조: QUINTANA, NICHOLLS Jim, 열심히 일한 성과가 있는 것 같네요. 냉장고에 좋은 맥주가 있으니 여기 와서 축하하는 것 어때요? Delores와 Heather도 관심 있나요? Bruce <ISTERN에서 전달> 주제: 전달: 근사한 한 해입니다! 4분기 상여금. 수신: Dept_D11 좋은 성과를 이룬 것을 축하합니다. 올 해의 상여금을 즐기시길 바랍니다. Irv <CHAAS에서 전달> 주제: 근사한 한 해입니다! 4분기 상여금. 수신: 모든 관리자 4분기 결과가 나왔습니다. 우리 모두 한 팀으로서 협력하여 계획을 초과 달성했습니다! 올 해 18%의 상여금을 발표하게 되어서 기쁩니다. 즐거운 휴일 보내시길 바랍니다. Christine Haas

이름:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
	1988-12-23-08.53.58.000000	ISTERN	전달: 근사한 한 해입니다! 4분기 상여금.	수신: Dept_D11 좋은 성과를 이룬 것을 축하합니다. 올 해의 상여금을 즐기시길 바랍니다. Irv <CHAAS에서 전달> 주제: 근사한 한 해입니다! 4분기 상여금. 수신: 모든 관리자 4분기 결과가 나왔습니다. 우리 모두 한 팀으로서 협력하여 계획을 초과 달성했습니다! 올 해 18%의 상여금을 발표하게 되어서 기쁩니다. 즐거운 휴일 보내시길 바랍니다. Christine Haas
	1988-12-22-14.07.21.136421	CHAAS	근사한 한 해입니다! 4분기 상여금.	수신: 모든 관리자 4분기 결과가 나왔습니다. 우리 모두 한 팀으로서 협력하여 계획을 초과 달성했습니다! 올 해 18%의 상여금을 발표하게 되어서 기쁩니다. 즐거운 휴일 보내시길 바랍니다. Christine Haas

ORG 테이블

이름:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
다음을 입력하십시오.	SMALLINT Not Null	VARCHAR(14)	SMALLINT	VARCHAR(10)	VARCHAR(13)
설명:	부서 번호	부서 이름	관리자 번호	기업의 지사	도시
값:	10	본사	160	기업	뉴욕
	15	뉴잉글랜드	50	동부	보스턴
	20	대서양 연안 중부	10	동부	Washington
	38	대서양 연안 남부	30	동부	Atlanta
	42	5대호	100	중서부	Chicago
	51	평원지대	140	중서부	Dallas
	66	태평양 연안	270	서부	San Francisco
	84	산악 지대	290	서부	Denver

PROJ 테이블

이름:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
다음을 입력하십시오.	CHAR(6)	VARCHAR(36)	CHAR(3)	CHAR(6)	DEC(5,2)	DATE	DATE	CHAR(6)

이름:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
값:	AD3100	ADMIN SERVICES	D01	000010	6.50	01/01/1982	02/01/1983	
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6.00	01/01/1982	02/01/1983	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2.00	01/01/1982	02/01/1983	AD3100
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1.00	01/01/1982	02/01/1983	AD3100
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2.00	01/01/1982	02/01/1983	AD3100
	IF1000	QUERY SERVICES	C01	000030	2.00	01/01/1982	02/01/1983	
	IF2000	USER EDUCATION	C01	000030	1.00	01/01/1982	02/01/1983	
	MA2100	WELD LINE AUTOMATION	D01	000010	12.00	01/01/1982	02/01/1983	
	MA2110	W L PROGRAMMING	D11	000060	9.00	01/01/1982	02/01/1983	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2.00	01/01/1982	12/01/1982	MA2100
	MA2112	W L ROBOT DESIGN	D11	000150	3.00	01/01/1982	12/01/1982	MA2100
	MA2113	W L PROD CONT PROGS	D11	000160	3.00	02/15/1982	12/01/1982	MA2100
	OP1000	OPERATION SUPPORT	E01	000050	6.00	01/01/1982	02/01/1983	
	OP1010	OPERATION	E11	000090	5.00	01/01/1982	02/01/1983	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5.00	01/01/1982	02/01/1983	
	OP2010	SYSTEMS SUPPORT	E21	000100	4.00	01/01/1982	02/01/1983	OP2010
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1.00	01/01/1982	02/01/1983	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1.00	01/01/1982	02/01/1983	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1.00	01/01/1982	02/01/1983	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1.00	01/01/1982	09/15/1982	MA2100

PROJACT 테이블

이름:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
다음을 입력하십시오.	CHAR(6)	SMALLINT	DEC(5,2)	DATE	DATE
값:	AD3100	10		01/01/1982	
	AD3110	10		01/01/1982	
	AD3111	60		01/01/1982	
	AD3111	60		03/15/1982	
	AD3111	70		03/15/1982	
	AD3111	80		04/15/1982	
	AD3111	180		10/15/1982	
	AD3111	70		02/15/1982	

이름:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
	AD3111	80		09/15/1982	
	AD3112	60		01/01/1982	
	AD3112	60		02/01/1982	
	AD3112	60		01/01/1983	
	AD3112	70		02/01/1982	
	AD3112	70		03/15/1982	
	AD3112	70		08/15/1982	
	AD3112	80		08/15/1982	
	AD3112	80		10/15/1982	
	AD3112	180		08/15/1982	
	AD3113	70		06/15/1982	
	AD3113	70		07/01/1982	
	AD3113	80		01/01/1982	
	AD3113	80		03/01/1982	
	AD3113	180		03/01/1982	
	AD3113	180		04/15/1982	
	AD3113	180		06/01/1982	
	AD3113	60		03/01/1982	
	AD3113	60		04/01/1982	
	AD3113	60		09/01/1982	
	AD3113	70		09/01/1982	
	AD3113	70		10/15/1982	
	IF1000	10		06/01/1982	
	IF1000	90		10/01/1982	
	IF1000	100		10/01/1982	
	IF2000	10		01/01/1982	
	IF2000	100		01/01/1982	
	IF2000	100		03/01/1982	
	IF2000	110		03/01/1982	
	IF2000	110		10/01/1982	
	MA2100	10		01/01/1982	
	MA2100	20		01/01/1982	
	MA2110	10		01/01/1982	
	MA2111	50		01/01/1982	
	MA2111	60		06/15/1982	
	MA2111	40		01/01/1982	
	MA2112	60		01/01/1982	
	MA2112	180		07/15/1982	
	MA2112	70		06/01/1982	
	MA2112	70		01/01/1982	
	MA2112	80		10/01/1982	

이름:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
	MA2113	60		07/15/1982	
	MA2113	80		01/01/1982	
	MA2113	70		04/01/1982	
	MA2113	80		10/01/1982	
	MA2113	180		10/01/1982	
	OP1000	10		01/01/1982	
	OP1010	10		01/01/1982	
	OP1010	130		01/01/1982	
	OP2010	10		01/01/1982	
	OP2011	140		01/01/1982	
	OP2011	150		01/01/1982	
	OP2012	140		01/01/1982	
	OP2012	160		01/01/1982	
	OP2013	140		01/01/1982	
	OP2013	170		01/01/1982	
	PL2100	30		01/01/1982	

PROJECT 테이블

이름:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
다음을 입력하십시오.	CHAR(6) Not Null	VARCHAR(24) Not Null	CHAR(3) Not Null	CHAR(6) Not Null	DEC(5,2)	DATE	DATE	CHAR(6)
설명:	프로젝트 번호	프로젝트 이름	담당 부서	담당자	추정 평균 직원 수	추정 시작 날짜	추정 종료 날짜	하위 프로젝트의 주 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
	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
다음을 입력하십시오.	DATE	VARCHAR(15)	VARCHAR(15)	INTEGER
설명:	판매 날짜	직원의 성	판매 지역	판매 수
값:	12/31/2005	LUCCHESSI	Ontario-South	1
	12/31/2005	LEE	Ontario-South	3
	12/31/2005	LEE	Quebec	1
	12/31/2005	LEE	Manitoba	2
	12/31/2005	GOUNOT	Quebec	1
	03/29/2006	LUCCHESSI	Ontario-South	3
	03/29/2006	LUCCHESSI	Quebec	1
	03/29/2006	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/2006	LEE	Quebec	3
	03/29/2006	LEE	Manitoba	5
	03/29/2006	GOUNOT	Ontario-South	3
	03/29/2006	GOUNOT	Quebec	1
	03/29/2006	GOUNOT	Manitoba	7
	03/30/2006	LUCCHESSI	Ontario-South	1
	03/30/2006	LUCCHESSI	Quebec	2
	03/30/2006	LUCCHESSI	Manitoba	1
	03/30/2006	LEE	Ontario-South	7
	03/30/2006	LEE	Ontario-North	3
	03/30/2006	LEE	Quebec	7
	03/30/2006	LEE	Manitoba	4
	03/30/2006	GOUNOT	Ontario-South	2
	03/30/2006	GOUNOT	Quebec	18
	03/30/2006	GOUNOT	Manitoba	1
	03/31/2006	LUCCHESSI	Manitoba	1
	03/31/2006	LEE	Ontario-South	14
	03/31/2006	LEE	Ontario-North	3
	03/31/2006	LEE	Quebec	7
	03/31/2006	LEE	Manitoba	3
	03/31/2006	GOUNOT	Ontario-South	2
	03/31/2006	GOUNOT	Quebec	1
	04/01/2006	LUCCHESSI	Ontario-South	3
	04/01/2006	LUCCHESSI	Manitoba	1
	04/01/2006	LEE	Ontario-South	8
	04/01/2006	LEE	Ontario-North	-
	04/01/2006	LEE	Quebec	8
	04/01/2006	LEE	Manitoba	9

이름:	SALES_DATE	SALES_PERSON	REGION	SALES
	04/01/2006	GOUNOT	Ontario-South	3
	04/01/2006	GOUNOT	Ontario-North	1
	04/01/2006	GOUNOT	Quebec	3
	04/01/2006	GOUNOT	Manitoba	7

STAFF 테이블

이름:	ID	이름	DEPT	JOB	YEARS	SALARY	COMM
다음을 입력하십시오.	SMALLINT Not Null	VARCHAR(9)	SMALLINT	CHAR(5)	SMALLINT	DECIMAL(7,2)	DECIMAL(7,2)
설명:	직원 번호	직원 이름	부서 번호	작업 유형	서비스 연도	현재 급여	커미션
값:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

PRODUCT 테이블

이름:	PID	이름	PRICE	PROMOPRICE	PROMOSTART	PROMOEND	설명
다음을 입력하십시오.	VARCHAR(10) Not Null	VARCHAR(128)	DECIMAL(30,2)	DECIMAL(30,2)	DATE	DATE	XML
값:	100-100-01	눈 삽, 기본 22인치	9.99	7.25	11/19/2004	12/19/2004	p1.xml
	100-101-01	눈 삽, 디럭스 24인치	19.99	15.99	12/18/2005	02/28/2006	p2.xml
	100-103-01	눈 삽, 수퍼 디럭스 26인치	49.99	39.99	12/22/2005	02/22/2006	p3.xml
	100-201-01	서리 제거기, 앞유리 4인치	3.99	--	--	--	p4.xml

다음은 위 테이블의 XML 컬럼에 대한 XML 스키마 정의 파일입니다. `product.xsd`

PURCHASEORDER 테이블

이름:	POID	STATUS	CUSTID	ORDERDATE	PORDER	COMMENTS
다음을 입력하십시오.	BIGINT Not Null	VARCHAR(10) Not Null	BIGINT	DATE	XML	VARCHAR(1000)

이름:	POID	STATUS	CUSTID	ORDERDATE	PORDER	COMMENTS
값:	5000	선적되지 않음	1002	02/18/2006	po1.xml	THIS IS A NEW PURCHASE ORDER
	5001	선적됨	1003	02/03/2005	po2.xml	THIS IS A NEW PURCHASE ORDER
	5002	선적됨	1001	02/29/2004	po3.xml	THIS IS A NEW PURCHASE ORDER
	5003	선적됨	1002	02/28/2005	po4.xml	THIS IS A NEW PURCHASE ORDER
	5004	선적됨	1005	11/18/2005	po5.xml	THIS IS A NEW PURCHASE ORDER
	5006	선적됨	1002	03/01/2006	po6.xml	THIS IS A NEW PURCHASE ORDER

다음은 위 테이블의 XML 컬럼에 대한 XML 스키마 정의 파일입니다. porder.xsd

CUSTOMER 테이블

이름:	CID	INFO
다음을 입력하십시오.	BIGINT Not Null	XML
값:	1000	c1.xml
	1001	c2.xml
	1002	c3.xml
	1003	c4.xml
	1004	c5.xml
	1005	c6.xml

다음은 위 테이블의 XML 컬럼에 대한 XML 스키마 정의 파일입니다. customer.xsd

CATALOG 테이블

이름:	이름	CATALOG
다음을 입력하십시오.	VARCHAR(128) Not Null	XML
값:	봄 카탈로그	cat1.xml
	여름 카탈로그	cat2.xml
	가을 카탈로그	cat3.xml
	겨울 카탈로그	cat4.xml

다음은 위 테이블의 XML 컬럼에 대한 XML 스키마 정의 파일입니다. catalog.xsd

INVENTORY 테이블

이름:	PID	QUANTITY	LOCATION
다음을 입력하십시오.	VARCHAR(10) Not Null	INTEGER	VARCHAR(128)
값:	100-100-01	5	--
	100-101-01	25	상점
	100-103-01	55	상점
	100-201-01	99	웨어하우스

PRODUCTSUPPLIER 테이블

이름:	PID	SID
다음을 입력하십시오.	VARCHAR(10) Not Null	VARCHAR(10) Not Null
값:	100-100-01	123-456-78
	100-101-01	123-456-78
	100-103-01	555-789-00
	100-201-01	989-897-23

SUPPLIERS 테이블

이름:	SID	ADDR
다음을 입력하십시오.	VARCHAR(10) Not Null	XML
값:	123-456-78	s1.xml
	555-789-00	s2.xml
	989-897-23	s3.xml
	111-898-45	s4.xml

다음은 위 테이블의 XML 컬럼에 대한 XML 스키마 정의 파일입니다. supplier.xsd

BLOB 및 CLOB 데이터 유형이 포함된 샘플 파일

이 섹션에는 EMP_PHOTO 파일(직원 사진) 및 EMP_RESUME 파일(직원의 이력서)에서 찾은 데이터가 표시됩니다.

Quintana 사진

그림 14. Dolores M. Quintana

Quintana 이력서

db200130.asc 파일에서 다음 텍스트를 찾았습니다.

이력서: Dolores M. Quintana

개인 정보

주소: 1150 Eglinton Ave Mellonville, Idaho 83725

전화번호:

(208) 555-9933

생년월일:

1925년 9월 15일

성별: 여성

결혼 유무:

기혼

신장: 5’2″

체중: 120lbs.

부서 정보

직원 번호:

000130

부서 번호:

C01

관리자:

Sally Kwan

직위: 분석가

전화번호:

(208) 555-4578

고용 날짜:

1971-07-28

교육

1965 수학 및 영어, B.A. Adelphi University

1960 치기공, Florida Institute of Technology

경력

10/91 - 현재

자문 시스템 분석가, 부서 운영에 필요한 문서화 도구 생성.

12/85 - 9/91

기술 작가, 작가, 텍스트 프로그래머 및 계획자.

1/79 - 11/85

COBOL 급여 프로그래머, 디젤 연료 회사를 위한 급여 프로그램 작성.

취미

- 요리
- 독서
- 재봉
- 리모델링

다음은 db200130.htm 파일의 컨텐츠입니다.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  Delores M. Quintana
<!-- Begin Header Records  ========================================== -->
<!-- DB200130 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:23                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:33:57">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:33:57">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:33:57"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  Delores M. Quintana<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  Delores M. Quintana<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  Delores M. Quintana<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>1150 Eglinton Ave
<BR>
Mellonville, Idaho 83725
<DT>Phone:
<DD>(208) 875-9933
<DT>Birthdate:
<DD>September 15, 1925
<DT>Sex:
<DD>Female
<DT>Marital Status:
<DD>Married
<DT>Height:
<DD>5'2"
<DT>Weight:
<DD>120 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000130
<DT>Dept Number:
<DD>C01
<DT>Manager:
<DD>Sally Kwan
<DT>Position:
<DD>Analyst
<DT>Phone:
<DD>(208) 385-4578
```

```
<DT>Hire Date:
<DD>1971-07-28<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1965
<DD>Math and English, B.A.
<BR>
Adelphi University
<P><DT>1960
<DD>Dental Technician
<BR>
Florida Institute of Technology<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>10/91 - present
<DD>Advisory Systems Analyst
<BR>
Producing documentation tools for engineering department.
<P><DT>12/85 - 9/91
<DD>Technical Writer
<BR>
Writer, text programmer, and planner.
<P><DT>1/79 - 11/85
<DD>COBOL Payroll Programmer
<BR>
Writing payroll programs for a diesel fuel company.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Cooking
<LI>Reading
<LI>Sewing
<LI>Remodeling<A NAME="Bot_Of_Page">
```

Nicholls 사진

그림 15. Heather A. Nicholls

Nicholls 이력서

db200140.asc 파일에서 다음 텍스트를 찾았습니다.

이력서: Heather A. Nicholls

개인 정보

주소: 844 Don Mills Ave Mellonville, Idaho 83734

전화번호:
(208) 555-2310

생년월일:
January 19, 1946

성별: 여성

결혼 유무:
미혼

신장: 5'8″

체중: 130lbs.

부서 정보

직원 번호:
000140

부서 번호:
C01

관리자:
Sally Kwan

직위: 분석가

전화번호:
(208) 555-1793

고용 날짜:
1976-12-15

교육

1972 컴퓨터 엔지니어링, Ph.D. University of Washington

1969 음악 및 물리학, M.A. Vassar College

경력

2/83 - 현재
건축가, OCR 제품의 아키텍처를 설계하여 OCR 개발.

12/76 - 1/83
텍스트 프로그래머, PL/I에서 광문자 인식(OCR) 프로그래밍.

9/72 - 11/76
천공 카드 품질 분석가, 천공 카드가 품질 스펙을 충족하는지 검사.

취미

- 모형 기차
- 실내 장식
- 수예
- 뜨게질

다음은 db200140.htm 파일의 컨텐츠입니다.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  Heather A. Nicholls
<!-- Begin Header Records  ========================================== -->
<!-- DB200140 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:21                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:17">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:17">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:17"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  Heather A. Nicholls<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  Heather A. Nicholls<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  Heather A. Nicholls<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>844 Don Mills Ave
<BR>
Mellonville, Idaho 83734
<DT>Phone:
<DD>(208) 610-2310
<DT>Birthdate:
<DD>January 19, 1946
<DT>Sex:
<DD>Female
<DT>Marital Status:
<DD>Single
<DT>Height:
<DD>5'8"
<DT>Weight:
<DD>130 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000140
<DT>Dept Number:
<DD>C01
<DT>Manager:
<DD>Sally Kwan
<DT>Position:
<DD>Analyst
<DT>Phone:
<DD>(208) 385-1793
```

```
<DT>Hire Date:
<DD>1976-12-15<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1972
<DD>Computer Engineering, Ph.D.
<BR>
University of Washington
<P><DT>1969
<DD>Music and Physics, B.A.
<BR>
Vassar College<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>2/83 - present
<DD>Architect, OCR Development
<BR>
Designing the architecture of OCR products.
<P><DT>12/76 - 1/83
<DD>Text Programmer
<BR>
Optical CHARacter recognition (OCR) programming in PL/I.
<P><DT>9/72 - 11/76
<DD>Punch Card Quality Analyst
<BR>
Checking punch cards met quality specifications.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Model railroading
<LI>Interior decorating
<LI>Embroidery
<LI>Knitting<A NAME="Bot_Of_Page">
```

Adamson 사진

그림 16. Bruce Adamson

Adamson 이력서

db200150.asc 파일에서 다음 텍스트를 찾았습니다.

이력서: Bruce Adamson

개인 정보

주소: 3600 Steeles Ave Mellonville, Idaho 83757

전화번호:

(208) 555-4489

생년월일:

May 17, 1947

성별: 남성

결혼 유무:

기혼

신장: 6’0″

체중: 175lbs.

부서 정보

직원 번호:

000150

부서 번호:

D11

관리자:

Irving Stern

직위: 디자이너

전화번호:

(208) 555-4510

고용 날짜:

1972-02-12

교육

1971 환경 공학, M.Sc. Johns Hopkins University

1968 미국 역사, B.A. Northwestern University

경력

8/79 - 현재

신경 회로망 설계, 인공 지능 제품에 사용할 신경 회로망 개발.

2/72 - 7/79

로봇 비전 개발, 시각을 에뮬레이트하기 위한 룰 기반 시스템 개발.

9/71 - 1/72

수치 적분 전문가, 은행 시스템 간 통신 지원.

취미

- 모터사이클 경주
- 확성기 조립
- 개인용 컴퓨터 조립
- 스케치

다음은 db200150.htm 파일의 컨텐츠입니다.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  Bruce Adamson
<!-- Begin Header Records  ========================================== -->
<!-- DB200150 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:21                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:39">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:39">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:39"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  Bruce Adamson<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  Bruce Adamson<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  Bruce Adamson<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>3600 Steeles Ave
<BR>
Mellonville, Idaho 83757
<DT>Phone:
<DD>(208) 725-4489
<DT>Birthdate:
<DD>May 17, 1947
<DT>Sex:
<DD>Male
<DT>Marital Status:
<DD>Married
<DT>Height:
<DD>6'0"
<DT>Weight:
<DD>175 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000150
<DT>Dept Number:
<DD>D11
<DT>Manager:
<DD>Irving Stern
<DT>Position:
<DD>Designer
<DT>Phone:
<DD>(208) 385-4510
```

```
<DT>Hire Date:
<DD>1972-02-12<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1971
<DD>Environmental Engineering, M.Sc.
<BR>
Johns Hopkins University
<P><DT>1968
<DD>American History, B.A.
<BR>
Northwestern University<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>8/79 - present
<DD>Neural Network Design
<BR>
Developing neural networks for machine intelligence products.
<P><DT>2/72 - 7/79
<DD>Robot Vision Development
<BR>
Developing rule-based systems to emulate sight.
<P><DT>9/71 - 1/72
<DD>Numerical Integration Specialist
<BR>
Helping bank systems communicate with each other.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Racing motorcycles
<LI>Building loudspeakers
<LI>Assembling personal computers
<LI>Sketching<A NAME="Bot_Of_Page">
```

Walker 사진

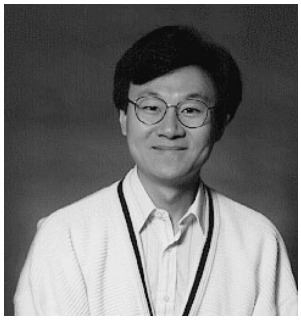


그림 17. James H. Walker

Walker 이력서

db200190.asc 파일에서 다음 텍스트를 찾았습니다.

이력서: James H. Walker

개인 정보

주소: 3500 Steeles Ave Mellonville, Idaho 83757

전화번호:
(208) 555-7325

생년월일:
June 25, 1952

성별: 남성

결혼 유무:
미혼

신장: 5'11"

체중: 166lbs.

부서 정보

직원 번호:
000190

부서 번호:
D11

관리자:
Irving Stern

직위: 디자이너

전화번호:
(208) 555-2986

고용 날짜:
1974-07-26

교육

1974 컴퓨터 과학, B.Sc. University of Massachusetts

1972 언어인류학, B.A. University of Toronto

경력

6/87 - 현재
마이크로코드 설계, 수학적 함수의 알고리즘 최적화.

4/77 - 5/87
프린터 기술 지원부, 레이저 프린터 설치 및 지원.

9/74 - 3/77
유지보수 프로그래밍, 메인프레임용 어셈블리 언어 컴파일러 패치.

취미

- 와인 시음
- 스키
- 수영
- 댄스

다음은 db200190.htm 파일의 컨텐츠입니다.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  James H. Walker
<!-- Begin Header Records  ========================================== -->
<!-- DB200190 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:20                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:59">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:59">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:59"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  James H. Walker<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  James H. Walker<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  James H. Walker<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>3500 Steeles Ave
<BR>
Mellonville, Idaho 83757
<DT>Phone:
<DD>(208) 725-7325
<DT>Birthdate:
<DD>June 25, 1952
<DT>Sex:
<DD>Male
<DT>Marital Status:
<DD>Single
<DT>Height:
<DD>5'11"
<DT>Weight:
<DD>166 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000190
<DT>Dept Number:
<DD>D11
<DT>Manager:
<DD>Irving Stern
<DT>Position:
<DD>Designer
<DT>Phone:
<DD>(208) 385-2986
```

```
<DT>Hire Date:
<DD>1974-07-26<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1974
<DD>Computer Studies, B.Sc.
<BR>
University of Massachusetts
<P><DT>1972
<DD>Linguistic Anthropology, B.A.
<BR>
University of Toronto<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>6/87 - present
<DD>Microcode Design
<BR>
Optimizing algorithms for mathematical functions.
<P><DT>4/77 - 5/87
<DD>Printer Technical Support
<BR>
Installing and supporting laser printers.
<P><DT>9/74 - 3/77
<DD>Maintenance Programming
<BR>
Patching assembly language compiler for mainframes.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Wine tasting
<LI>Skiing
<LI>Swimming
<LI>Dancing<A NAME="Bot_Of_Page">
```

부록 G. 예약된 스키마 이름 및 예약어

데이터베이스 관리 프로그램에 필요한 특정 이름 사용에는 제한사항이 있습니다. 경우에 따라 일부 이름은 예약되어 있어 응용프로그램에서 사용할 수 없거나 일부 이름은 데이터베이스 관리 프로그램에서 사용할 수 있어도, 응용프로그램에서 사용하도록 권장하지는 않습니다.

예약된 스키마 이름은 다음과 같습니다.

- SYSCAT
- SYSFUN
- SYSIBM
- SYSIBMADM
- SYSPROC
- SYSPUBLIC
- SYSSTAT

'SYS'는 규칙상 시스템에서 예약된 영역을 표시하는 데 사용되므로, 스키마 이름은 'SYS' 접두어로 시작하지 않을 것을 적극 권장합니다. 별명, 전역 변수, 트리거, 사용자 정의 함수 또는 사용자 정의 유형은 'SYS'로 시작하는 스키마 이름에 위치할 수 없습니다(SQLSTATE 42939).

DB2 도구에서는 DB2QP 스키마 및 SYSTOOLS 스키마를 사용할 수 없습니다. 이러한 스키마가 데이터베이스 관리 프로그램에서 사용될 수 있더라도 사용자가 오브젝트를 정의하지 않는 것이 권장됩니다.

또한 SESSION도 스키마 이름으로 사용하지 않을 것을 권장합니다. 선언된 임시 테이블이 SESSION에서 규정되어야 하므로 응용프로그램에서 영구 테이블의 이름과 동일한 이름의 임시 테이블을 선언할 수 있으나 이로 인해 응용프로그램 논리가 복잡해 집니다. 이러한 가능성을 피하기 위해서는 선언된 임시 테이블을 처리할 때를 제외하고는 SESSION 스키마를 사용하지 마십시오.

엄밀히 말하면 DB2 버전 9에는 예약어가 없습니다. 키워드는 SQL 키워드로도 해석될 수 있는 컨텍스트를 제외하고는 일반 ID로 사용될 수 있습니다. 그러한 경우, 단어는 분리 ID로 지정되어야 합니다. 예를 들어, COUNT는 분리되는 경우를 제외하고는 SELECT문에서 컬럼 이름으로 사용될 수 없습니다.

ISO/ANSI SQL2003 및 다른 DB2 데이터베이스 제품에는 Linux용 DB2 Database, UNIX 및 Windows에서 강제 실행되지 않는 예약어가 포함되어 있습니다. 그러나 이러한 예약어는 이식성을 저하시키므로 일반 ID로 사용하지 않는 것이 좋습니다.

DB2 데이터베이스 제품에서의 이식성을 위해 다음을 예약어로 고려해야 합니다.

ACTIVATE
ADD
AFTER
ALIAS
ALL
ALLOCATE
ALLOW
ALTER
AND
ANY
AS
ASENSITIVE
ASSOCIATE
ASUTIME
AT
ATTRIBUTES
AUDIT
AUTHORIZATION
AUX
AUXILIARY
BEFORE
BEGIN
BETWEEN
BINARY
BUFFERPOOL
BY
CACHE
CALL
CALLED
CAPTURE
CARDINALITY
CASCADED
CASE
CAST
CCSID
CHAR
CHARACTER
CHECK
CLONE
CLOSE
CLUSTER
COLLECTION
COLLID
COLUMN
COMMENT
COMMIT
CONCAT
CONDITION
CONNECT
CONNECTION
CONSTRAINT
CONTAINS
CONTINUE
COUNT
COUNT_BIG
CREATE
CROSS
DOCUMENT
DOUBLE
DROP
DSSIZE
DYNAMIC
EACH
EDITPROC
ELSE
ELSEIF
ENABLE
ENCODING
ENCRYPTION
END
END-EXEC
ENDING
ERASE
ESCAPE
EVERY
EXCEPT
EXCEPTION
EXCLUDING
EXCLUSIVE
EXECUTE
EXISTS
EXIT
EXPLAIN
EXTERNAL
EXTRACT
FENCED
FETCH
FIELDPROC
FILE
FINAL
FOR
FOREIGN
FREE
FROM
FULL
FUNCTION
GENERAL
GENERATED
GET
GLOBAL
GO
GOTO
GRANT
GRAPHIC
GROUP
HANDLER
HASH
HASHED_VALUE
HAVING
HINT
HOLD
HOUR
HOURS
IDENTITY
LOCK
LOCKMAX
LOCKSIZE
LONG
LOOP
MAINTAINED
MATERIALIZED
MAXVALUE
MICROSECOND
MICROSECONDS
MINUTE
MINUTES
MINVALUE
MODE
MODIFIES
MONTH
MONTHS
NAN
NEW
NEW_TABLE
NEXTVAL
NO
NOCACHE
NOCYCLE
NODENAME
NODENUMBER
NOMAXVALUE
NOMINVALUE
NONE
NOORDER
NORMALIZED
NOT
NULL
NULLS
NUMPARTS
OBID
OF
OLD
OLD_TABLE
ON
OPEN
OPTIMIZATION
OPTIMIZE
OPTION
OR
ORDER
OUT
OUTER
OVER
OVERRIDING
PACKAGE
PADDED
PAGESIZE
PARAMETER
PART
PARTITION
PARTITIONED
ROUND_CEILING
ROUND_DOWN
ROUND_FLOOR
ROUND_HALF_DOWN
ROUND_HALF_EVEN
ROUND_HALF_UP
ROUND_UP
ROUTINE
ROW
ROW_NUMBER
ROWNUMBER
ROWS
ROWSET
RRN
RUN
SAVEPOINT
SCHEMA
SCRATCHPAD
SCROLL
SEARCH
SECOND
SECONDS
SECQTY
SECURITY
SELECT
SENSITIVE
SEQUENCE
SESSION
SESSION_USER
SET
SIGNAL
SIMPLE
SNAN
SOME
SOURCE
SPECIFIC
SQL
SQLID
STACKED
STANDARD
START
STARTING
STATEMENT
STATIC
STATMENT
STAY
STOGROUP
STORES
STYLE
SUBSTRING
SUMMARY
SYNONYM
SYSFUN
SYSIBM
SYSPROC
SYSTEM
SYSTEM_USER

CURRENT	IF	PARTITIONING	TABLE
CURRENT_DATE	IMMEDIATE	PARTITIONS	TABLESPACE
CURRENT_LC_CTYPE	IN	PASSWORD	THEN
CURRENT_PATH	INCLUDING	PATH	TIME
CURRENT_SCHEMA	INCLUSIVE	PIECESIZE	TIMESTAMP
CURRENT_SERVER	INCREMENT	PLAN	TO
CURRENT_TIME	INDEX	POSITION	TRANSACTION
CURRENT_TIMESTAMP	INDICATOR	PRECISION	TRIGGER
CURRENT_TIMEZONE	INF	PREPARE	TRIM
CURRENT_USER	INFINITY	PREVVAL	TRUNCATE
CURSOR	INHERIT	PRIMARY	TYPE
CYCLE	INNER	PRIQTY	UNDO
DATA	INOUT	PRIVILEGES	UNION
DATABASE	INSENSITIVE	PROCEDURE	UNIQUE
DATAPARTITIONNAME	INSERT	PROGRAM	UNTIL
DATAPARTITIONNUM	INTEGRITY	PSID	UPDATE
DATE	INTERSECT	PUBLIC	USAGE
DAY	INTO	QUERY	USER
DAYS	IS	QUERYNO	USING
DB2GENERAL	ISOBID	RANGE	VALIDPROC
DB2GENRL	ISOLATION	RANK	VALUE
DB2SQL	ITERATE	READ	VALUES
DBINFO	JAR	READS	VARIABLE
DBPARTITIONNAME	JAVA	RECOVERY	VARIANT
DBPARTITIONNUM	JOIN	REFERENCES	VCAT
DEALLOCATE	KEEP	REFERENCING	VERSION
DECLARE	KEY	REFRESH	VIEW
DEFAULT	LABEL	RELEASE	VOLATILE
DEFAULTS	LANGUAGE	RENAME	VOLUMES
DEFINITION	LATERAL	REPEAT	WHEN
DELETE	LC_CTYPE	RESET	WHENEVER
DENSE_RANK	LEAVE	RESIGNAL	WHERE
DENSERANK	LEFT	RESTART	WHILE
DESCRIBE	LIKE	RESTRICT	WITH
DESCRIPTOR	LINKTYPE	RESULT	WITHOUT
DETERMINISTIC	LOCAL	RESULT_SET_LOCATOR	WLM
DIAGNOSTICS	LOCALDATE	RETURN	WRITE
DISABLE	LOCALE	RETURNS	XMLELEMENT
DISALLOW	LOCALTIME	REVOKE	XMLEXISTS
DISCONNECT	LOCALTIMESTAMP	RIGHT	XMLNAMESPACES
DISTINCT	LOCATOR	ROLE	YEAR
DO	LOCATORS	ROLLBACK	YEARS

다음 목록에는 이전 목록에 없는 ISO/ANSI SQL2003 예약어가 들어 있습니다.

ABS	GROUPING	REGR_INTERCEPT
ARE	INT	REGR_R2
ARRAY	INTEGER	REGR_SLOPE
ASYMMETRIC	INTERSECTION	REGR_SXX
ATOMIC	INTERVAL	REGR_SXY
AVG	LARGE	REGR_SYY
BIGINT	LEADING	ROLLUP
BLOB	LN	SCOPE
BOOLEAN	LOWER	SIMILAR
BOTH	MATCH	SMALLINT
CEIL	MAX	SPECIFICTYPE
CEILING	MEMBER	SQLEXCEPTION
CHAR_LENGTH	MERGE	SQLSTATE

CHARACTER_LENGTH
CLOB
COALESCE
COLLATE
COLLECT
CONVERT
CORR
CORRESPONDING
COVAR_POP
COVAR_SAMP
CUBE
CUME_DIST
CURRENT_DEFAULT_TRANSFORM_GROUP
CURRENT_ROLE
CURRENT_TRANSFORM_GROUP_FOR_TYPE
DEC
DECIMAL
DEREF
ELEMENT
EXEC
EXP
FALSE
FILTER
FLOAT
FLOOR
FUSION
METHOD
MIN
MOD
MODULE
MULTISET
NATIONAL
NATURAL
NCHAR
NCLOB
NORMALIZE
NULLIF
NUMERIC
OCTET_LENGTH
ONLY
OVERLAPS
OVERLAY
PERCENT_RANK
PERCENTILE_CONT
PERCENTILE_DISC
POWER
REAL
RECURSIVE
REF
REGR_AVGX
REGR_AVGY
REGR_COUNT
SQLWARNING
SQRT
STDDEV_POP
STDDEV_SAMP
SUBMULTISET
SUM
SYMMETRIC
TABLESAMPLE
TIMEZONE_HOUR
TIMEZONE_MINUTE
TRAILING
TRANSLATE
TRANSLATION
TREAT
TRUE
UESCAPE
UNKNOWN
UNNEST
UPPER
VAR_POP
VAR_SAMP
VARCHAR
VARYING
WIDTH_BUCKET
WINDOW
WITHIN

부록 H. 트리거 및 참조 제한조건 사이의 상호 작용 예

갱신 조작으로 인해 참조 제한조건 및 점검 제한조건과 트리거의 상호 작용이 발생할 수 있습니다.

그림 18 및 관련 설명은 데이터베이스에서 데이터를 갱신하는 명령문에 대해 수행되는 처리를 나타냅니다.

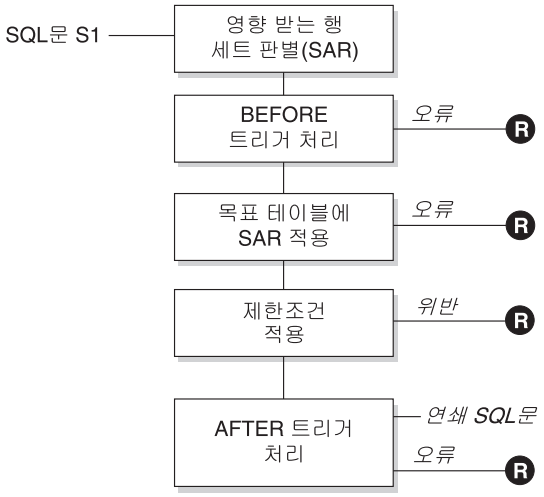


그림 18. 연관된 트리거 및 제한조건을 사용하여 명령문 처리

그림 18에서는 테이블을 갱신하는 명령문 처리의 일반적인 순서에 대해 설명합니다. 여기서는 테이블에 연쇄적으로 적용되는 BEFORE 트리거, 참조 제한조건, 점검 제한조건 및 AFTER 트리거가 포함되어 있는 상황을 가정합니다. 다음은 그림 18에 있는 상자 및 기타 항목에 대한 설명입니다.

- 명령문 S_1

  프로세스를 시작하는 DELETE, INSERT 또는 UPDATE 문입니다. 명령문 S_1은 이 설명에서 *주제 테이블*이라고 하는 테이블(또는 일부 테이블에 대한 갱신 가능 뷰)을 식별합니다.

- 영향을 받는 행 세트 판별

  이 단계에서는 AFTER 트리거에서 연쇄 명령문과 CASCADE 및 SET NULL에 대한 참조 제한조건 삭제 규칙에 대해 반복되는 프로세스가 시작됩니다.

  이 단계의 목적은 명령문의 *영향을 받는 행 세트*를 판별하는 것입니다. 포함된 행 세트는 다음 명령문에 따라 달라집니다.

  - DELETE의 경우: 명령문의 검색 조건을 충족하는 모든 행(또는 위치가 지정된 DELETE의 현재 행)
  - INSERT의 경우: VALUES절 또는 fullselect에 의해 식별되는 행
  - UPDATE의 경우: 검색 조건을 충족하는 모든 행(또는 위치가 지정된 UPDATE의 현재 행)

  영향을 받는 행이 비어 있으면 BEFORE 트리거, 주제 테이블에 적용되는 변경사항 또는 명령문 프로세스에 대한 제한조건이 없습니다.

- BEFORE 트리거 처리

  모든 BEFORE 트리거는 작성 순서에 따라 오름차순으로 처리됩니다. 각 BEFORE 트리거는 영향 받은 행 세트의 각 행에 대해 한 번씩 트리거 조치를 처리합니다.

  지금까지 원래 명령문인 S_1의 결과로 변경된 모든 사항이 롤백되는 경우 트리거 조치 처리 중 오류가 발생할 수 있습니다.

  BEFORE 트리거가 없거나 영향을 받는 행 세트가 비어 있는 경우 이 단계를 건너뜁니다.

- 주제 테이블에 영향을 받는 행 세트 적용

  영향을 받는 행 세트를 사용하여 실제 삭제, 삽입 또는 갱신이 데이터베이스의 주제 테이블에 적용됩니다.

  지금까지 원래 명령문인 S_1의 결과로 변경된 모든 사항이 롤백되는 경우 영향을 받는 행 세트를 적용할 때(예: 고유한 인덱스가 존재하는 중복 키와 함께 행을 삽입하려는 경우) 트리거 조치 처리 중 오류가 발생할 수 있습니다.

- 제한조건 적용

  영향을 받는 행 세트가 비어 있는 경우 주제 테이블과 연관된 제한 조건이 적용됩니다. 이러한 제한조건에는 고유 제한조건, 고유 인덱스, 참조 제한조건, 점검 제한조건 및 뷰에서 WITH CHECK OPTION 관련 점검이 있습니다. CASCADE 또는 SET NULL 삭제 규칙이 포함된 참조 제한조건으로 인해 추가 트리거가 활성화될 수 있습니다.

  모든 제한조건 또는 WITH CHECK OPTION 위반으로 인해 오류가 발생하고 지금까지 원래 명령문인 S_1의 결과로 변경된 모든 사항이 롤백됩니다.

- AFTER 트리거 처리

  S_1로 활성화된 모든 AFTER 트리거는 작성 순서에 따라 오름차순으로 처리됩니다.

  영향을 받는 행 세트가 비어 있더라도 FOR EACH STATEMENT 트리거는 트리거 조치를 정확히 한 번만 처리합니다. FOR EACH ROW 트리거는 영향을 받은 행 세트의 각 행에 대해 한 번씩 트리거 조치를 처리합니다.

지금까지 원래 S_1의 결과로 변경된 모든 사항이 롤백되는 경우 트리거 조치 처리 중 오류가 발생할 수 있습니다.

트리거의 트리거 조치에는 DELETE, INSERT 또는 UPDATE문인 트리거 명령문이 포함될 수 있습니다. 이렇게 설명하는 이유는 이러한 각 명령문을 *연쇄 명령문*으로 간주하기 위함입니다.

연쇄 명령문은 AFTER 트리거의 트리거 조치의 일부로 처리되는 DELETE, INSERT 또는 UPDATE문입니다. 이 명령문은 트리거 처리의 연쇄 레벨을 시작합니다. 연쇄 레벨은 트리거 명령문을 새 S_1으로 지정하고 새 여기서 설명한 모든 단계를 반복적으로 수행할 수 있습니다.

각 S_1에서 활성화한 모든 AFTER 트리거의 모든 트리거 명령문이 완료되도록 처리되면 원래 S_1의 처리가 완료됩니다.

- R = 이전 S_1에 대한 변경사항 롤백

  처리 중 발생한 모든 오류(제한조건 위반 포함)로 인해 원래 명령문 S_1의 결과로 직접 또는 간접적으로 변경된 모든 사항이 롤백됩니다. 원래 명령문 S_1이 실행되기 직전에 데이터베이스가 동일한 상태로 롤백됩니다.

부록 I. Explain 테이블

Explain 테이블은 Explain 기능이 활성화될 때 액세스 플랜을 캡처합니다. Explain을 호출하려면 Explain 테이블을 먼저 작성해야 합니다. 다음 메소드 중 하나를 사용하여 작성할 수 있습니다.

- SYSPROC.SYSINSTALLOBJECTS 프로시저 호출

  ```
  db2 CONNECT TO database-name
  db2 CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'C',
          CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128))
  ```

  이러한 호출은 SYSTOOLS 스키마 아래에 Explain 테이블을 작성합니다. 테이블을 다른 스키마 아래에 작성하려면, 호출 중 마지막 매개변수로 스키마 이름을 지정하십시오.

- EXPLAIN.DDL DB2 명령 파일 실행

  ```
  db2 CONNECT TO database-name
  db2 -tf EXPLAIN.DDL
  ```

  이 명령 파일은 현재 스키마 아래에 Explain 테이블을 작성합니다. 이는 Windows 운영 체제에서 DB2PATH₩misc 디렉토리에, Linux 및 UNIX 운영 체제에서 INSTHOME/sqllib/misc 디렉토리에 있습니다. DB2PATH는 사용자의 DB2 사본을 설치한 위치이며 INSTHOME은 인스턴스 홈 디렉토리입니다.

Explain 기능은 데이터를 입력 중인 Explain 테이블을 규정할 때 다음과 같은 ID를 사용합니다.

- 동적 SQL의 경우 세션 권한 부여 ID
- 정적 SQL의 경우 명령문 권한 부여 ID

서로 다른 스키마 아래 있는 Explain 테이블 세트 또는 Explain 테이블 세트를 가리키는 별명과 스키마를 연관시킬 수 있습니다. 만일 해당 스키마에 Explain 테이블이 없으면, Explain 기능은 SYSTOOLS 스키마의 Explain 테이블을 확인하고 그 테이블을 사용하려고 시도합니다.

Explain 기능으로 Explain 테이블을 채워도 트리거 또는 참조 또는 점검 제한조건을 활성화하지 않습니다. 예를 들어 삽입 트리거가 EXPLAIN_INSTANCE 테이블에서 정의되었고 적격한 명령문이 설명된 경우 트리거는 활성화되지 않습니다.

파티션된 데이터베이스 시스템에서 Explain 기능의 성능을 개선하려면 단일 파티션 데이터베이스 파티션 그룹, 가능하면 쿼리를 컴파일할 때 연결될 동일한 데이터베이스 파티션에서 Explain 테이블을 작성하는 것이 바람직합니다.

ADVISE_INDEX 테이블

ADVISE_INDEX 테이블은 권장되는 인덱스를 나타냅니다.

표 237. ADVISE_INDEX 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	없음	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	없음	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	없음	동적 명령문이 explain될 때 수행되는 패키지 이름 또는 정적 SQL이 explain될 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	없음	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	없음	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	없음	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	없음	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	없음	없음	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호
QUERYNO	INTEGER	없음	없음	설명된 SQL문에 대한 숫자 ID입니다. CLP 또는 CLI를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 순차적으로 증분되는 값입니다. 그렇지 않으면 디폴트값은 정적 SQL문의 경우 STMTNO의 값, 동적 SQL문의 경우 1입니다.
QUERYTAG	CHAR(20)	없음	없음	설명된 각 SQL문에 대한 ID 태그입니다. CLP를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 'CLP'입니다. CLI를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 'CLI'입니다. 그렇지 않으면, 사용된 디폴트값은 공백입니다.
NAME	VARCHAR(128)	없음	없음	인덱스 이름.
CREATOR	VARCHAR(128)	없음	없음	인덱스 이름의 규정자.
TBNAME	VARCHAR(128)	없음	없음	인덱스가 정의되는 테이블 또는 별칭의 이름.
TBCREATOR	VARCHAR(128)	없음	없음	테이블 이름의 규정자.
COLNAMES	CLOB(2M)	없음	없음	컬럼 이름 목록.
UNIQUERULE	CHAR(1)	없음	없음	고유한 규칙: • D = 중복 허용 • P = 1차 인덱스 • U = 고유 항목만 허용
COLCOUNT	SMALLINT	없음	없음	키의 컬럼 수 + Include 컬럼 수(있을 경우).
IID	SMALLINT	없음	없음	내부 인덱스 ID.
NLEAF	BIGINT	없음	없음	리프 페이지 수. 통계가 수집되지 않는 경우 -1입니다.
NLEVELS	SMALLINT	없음	없음	인덱스 레벨 수. 통계가 수집되지 않는 경우 -1입니다.
FIRSTKEYCARD	BIGINT	없음	없음	구별 최초 키 값의 수. 통계가 수집되지 않는 경우 -1입니다.

표 237. ADVISE_INDEX 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
FULLKEYCARD	BIGINT	없음	없음	구별 전체 키 값의 수. 통계가 수집되지 않는 경우 -1입니다.
CLUSTERRATIO	SMALLINT	없음	없음	인덱스를 사용한 데이터 클러스터링 등급입니다. 통계가 수집되지 않거나 상세한 인덱스 통계가 수집되는 경우(이 경우 CLUSTERFACTOR가 대신 사용됨) -1입니다.
AVGPARTITION_ CLUSTERRATIO	SMALLINT	없음	없음	단일 데이터 파티션 내의 데이터 클러스터링 등급입니다. 테이블이 테이블 파티션되지 않는 경우, 통계가 수집되지 않는 경우 또는 상세한 통계가 수집되는 경우(이 경우에는 AVGPARTITION_ CLUSTERFACTOR가 대신 사용됨) -1입니다.
AVGPARTITION_ CLUSTERFACTOR	DOUBLE	없음	없음	단일 데이터 파티션에서 클러스터링 등급의 상세한 측정입니다. 테이블이 테이블 파티션되지 않는 경우, 통계가 수집되지 않는 경우 또는 인덱스가 별칭에 대해 정의되는 경우 -1입니다.
AVGPARTITION_PAGE_ FETCH_PAIRS	VARCHAR(520)	없음	없음	문자 양식에서 쌍으로 된 정수 목록입니다. 각 쌍은 잠재적 버퍼 풀 크기 및 테이블에서 단일 데이터 파티션에 액세스하기 위해 필요한 대응하는 페이지 페치를 나타냅니다. 사용 가능한 데이터가 없는 경우 또는 테이블이 테이블 파티션되지 않는 경우 길이가 0인 문자열입니다.
DATAPARTITION_ CLUSTERFACTOR	DOUBLE	없음	없음	데이터 파티션에 관하여 인덱스 키의 "클러스터링"을 측정하는 통계입니다. 이 필드에는 0과 1 사이의 숫자가 들어있으며 1은 완벽한 클러스터링을 나타내고 0은 클러스터링이 없음을 나타냅니다.
CLUSTERFACTOR	DOUBLE	없음	없음	클러스터링 등급의 상세 측정입니다. 상세한 인덱스 통계가 수집되지 않은 경우 또는 인덱스가 별칭에 대해 정의되는 경우 -1입니다.
USERDEFINED	SMALLINT	없음	없음	사용자가 정의함.
SYSTEM_REQUIRED	SMALLINT	없음	없음	• 다음 조건 중 하나가 만족되는 경우 1입니다. – 이 인덱스가 기본 또는 고유 키 제한조건에 필요하거나, 이 인덱스가 다차원적으로 클러스터된(MDC) 테이블에 대한 차원 블록 인덱스 또는 복합 블록 인덱스입니다. – 유형이 지정된 테이블의 (OID) 컬럼에 대한 인덱스입니다. • 다음 조건이 모두 만족되는 경우 2입니다. – 이 인덱스가 기본 또는 고유 키 제한조건에 필요하거나, 이 인덱스가 MDC 테이블에 대한 차원 블록 인덱스 또는 복합 블록 인덱스입니다. – 유형이 지정된 테이블의 (OID) 컬럼에 대한 인덱스입니다. • 그렇지 않으면 0입니다.
CREATE_TIME	TIMESTAMP	없음	없음	인덱스가 작성된 시간.

표 237. ADVISE_INDEX 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
STATS_TIME	TIMESTAMP	예	없음	이 인덱스에 대해 기록된 통계가 변경된 마지막 시간입니다. 통계가 사용 가능하지 않으면 널(NULL)입니다.
PAGE_FETCH_PAIRS	VARCHAR(520)	없음	없음	문자 양식으로 표시되는 정수 쌍의 목록입니다. 각 쌍은 가상 버퍼의 페이지 수 및 해당 가상 버퍼를 사용하여 이 인덱스를 갖는 테이블을 스캔하기 위해 필요한 페이지 페치 수입니다. (사용 가능한 데이터가 없을 경우에는 길이가 0인 문자열).
REMARKS	VARCHAR(254)	예	없음	사용자가 제공하는 주석 또는 널(NULL).
DEFINER	VARCHAR(128)	없음	없음	인덱스를 작성한 사용자.
CONVERTED	CHAR(1)	없음	없음	나중에 사용하기 위해 예약됨
SEQUENTIAL_PAGES	BIGINT	없음	없음	리프 페이지 사이의 큰 갭이 없거나 소수인 인덱스 키 순서에서 디스크에 위치한 리프 페이지 수입니다. (사용 가능한 통계가 없을 경우에는 -1).
DENSITY	INTEGER	없음	없음	인덱스로 채워지는 페이지 범위에 있는 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로, 퍼센트(0 - 100 범위의 정수)로 표현됩니다. 통계를 사용할 수 없는 경우 -1입니다.
FIRST2KEYCARD	BIGINT	없음	없음	인덱스의 처음 두 컬럼을 사용하는 구별 키의 수입니다. 통계가 없거나 적용할 수 없는 경우 -1입니다.
FIRST3KEYCARD	BIGINT	없음	없음	인덱스의 처음 세 컬럼을 사용하는 구별 키의 수입니다. 통계가 없거나 적용할 수 없는 경우 -1입니다.
FIRST4KEYCARD	BIGINT	없음	없음	인덱스의 처음 네 컬럼을 사용하는 구별 키의 수입니다. 통계가 없거나 적용할 수 없는 경우 -1입니다.
PCTFREE	SMALLINT	없음	없음	인덱스의 초기 빌드 중에 예약될 각 인덱스 리프 페이지의 백분율입니다. 이 스페이스는 인덱스가 빌드된 후에 추후 삽입에 사용 가능합니다.
UNIQUE_COLCOUNT	SMALLINT	없음	없음	고유 키에 필요한 컬럼 수입니다. 항상 <=COLCOUNT 입니다. Include 컬럼이 있는 경우에만 < COLCOUNT 입니다. 인덱스에 고유 키가 없는 경우(중복 허용) -1입니다.
MINPCTUSED	SMALLINT	없음	없음	0이 아닌 경우 온라인 인덱스 조각 모음이 사용 가능하고 값은 페이지를 병합하기 전에 최소 사용 스페이스의 임계값입니다.
REVERSE_SCANS	CHAR(1)	없음	없음	• Y = 인덱스가 역방향 스캔을 지원함 • N = 인덱스가 역방향 스캔을 지원하지 않음
USE_INDEX	CHAR(1)	예	없음	• Y = 인덱스 권장 또는 평가 • N = 인덱스가 권장되지 않음 • R = 기존 클러스터링 RID 인덱스가 클러스터 해제되도록(디자인 어드바이저에 의해) 권장되었음. 이것은 테이블에 대해 새 클러스터링 RID 인덱스가 권장되는 경우입니다.
CREATION_TEXT	CLOB(2M)	없음	없음	인덱스를 작성하는 데 사용된 SQL문.

표 237. ADVISE_INDEX 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
PACKED_DESC	BLOB(1M)	예	없음	테이블의 내부 설명.
RUN_ID	TIMESTAMP	예	FK	ADVISE_INSTANCE 테이블의 한 행의 START_TIME에 대응하는 값으로, 동일한 디자인 어드바이저 실행에 링크합니다.
INDEXTYPE	VARCHAR(4)	없음	없음	인덱스 유형. • CLUS = 클러스터링 • REG = 일반 • DIM = 차원 블록 인덱스 • BLOK = 블록 인덱스
EXISTS	CHAR(1)	없음	없음	인덱스가 데이터베이스 카탈로그에 존재하는 경우 'Y'로 설정하십시오.
RIDTOBLOCK	CHAR(1)	없음	없음	RID 인덱스가 디자인 어드바이저에서 블록 인덱스를 작성하는 데 사용된 경우 'Y'로 설정하십시오.

ADVISE_INSTANCE 테이블

ADVISE_INSTANCE 테이블에는 시작 시간을 포함하여 db2advis 실행에 관한 정보가 들어있습니다. db2advis의 각 실행에 대해 하나의 행을 포함합니다. 기타 ADVISE 테이블에는 동일한 디자인 어드바이저 실행 중에 작성되는 행에 대한 ADVISE_INSTANCE 테이블의 START_TIME 컬럼에 링크하는 외부 키(RUN_ID)가 있습니다.

표 238. ADVISE_INSTANCE 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
START_TIME	TIMESTAMP	없음	PK	db2advis 실행이 시작되는 시간.
END_TIME	TIMESTAMP	없음	없음	db2advis 실행이 종료되는 시간
MODE	VARCHAR(4)	없음	없음	디자인 어드바이저에서 -m 옵션과 함께 지정된 값(예: MQT 및 MDC를 지정하려면 'MC').
WKLD_COMPRESSION	CHAR(4)	없음	없음	디자인 어드바이저가 실행된 워크로드 압축.
STATUS	CHAR(9)	없음	없음	디자인 어드바이저 실행 상태입니다. 상태는 'STARTED', 'COMPLETED'(성공한 경우) 또는 내부 오류의 경우 'EI' 또는 외부 오류의 경우 'EX'가 접두어로 붙는 오류 번호일 수 있습니다. 오류 번호는 SQLCODE를 나타냅니다.

ADVISE_MQT 테이블

ADVISE_MQT 테이블에는 디자인 어드바이저가 권장하는 구체화된 쿼리 테이블(MQT)에 관한 정보가 들어있습니다.

표 239. ADVISE_MQT 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	없음	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	없음	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	없음	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	없음	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	없음	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	없음	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	없음	이 Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	INTEGER	없음	없음	이 Explain 정보와 관련된 패키지 내의 명령문 번호
NAME	VARCHAR(128)	없음	없음	MQT 이름.
CREATOR	VARCHAR(128)	없음	없음	MQT 작성자 이름.
IID	SMALLINT	없음	없음	내부 ID.
CREATE_TIME	TIMESTAMP	없음	없음	MQT가 작성된 시간.
STATS_TIME	TIMESTAMP	예	없음	통계가 작성된 시간.
NUMROWS	DOUBLE	없음	없음	MQT에서 계산된 행 수.
NUMCOLS	SMALLINT	없음	없음	MQT에 정의된 컬럼 수.
ROWSIZE	DOUBLE	없음	없음	MQT에 있는 행의 평균 길이(바이트).
BENEFIT	FLOAT	없음	없음	나중에 사용하기 위해 예약됨
USE_MQT	CHAR(1)	예	없음	MQT가 권장될 경우 'Y'로 설정하십시오.
MQT_SOURCE	CHAR(1)	예	없음	MQT 후보가 생성되는 장소를 표시합니다. MQT 후보가 즉시 새로 고침 MQT인 경우 'I', 전체 새로 고침 지연 MQT로만 작성할 수 있는 경우 'D'로 설정하십시오.
QUERY_TEXT	CLOB(2M)	없음	없음	MQT를 정의하는 쿼리를 포함합니다.
CREATION_TEXT	CLOB(2M)	없음	없음	MQT에 대한 CREATE TABLE DDL이 포함됩니다.
SAMPLE_TEXT	CLOB(2M)	없음	없음	MQT에 대한 상세한 통계를 얻는 데 사용되는 샘플링 쿼리가 들어있습니다. 디자인 어드바이저를 위해 상세한 통계가 필요할 때만 사용됩니다. 결과 샘플 통계가 이 테이블에 표시됩니다. 널(NULL)인 경우 이 MQT에 대한 샘플링 쿼리가 작성되지 않았습니다.
COLSTATS	CLOB(2M)	없음	없음	MQT에 대한 컬럼 통계가 들어있습니다(널(null)이 아닌 경우). 이들 통계는 XML 형식으로 되어 있고 컬럼 이름, 컬럼 카디널리티(cardinality) 및 선택적으로 HIGH2KEY 및 LOW2KEY 값을 포함합니다.
EXTRA_INFO	BLOB(2M)	없음	없음	기타 출력을 위해 예약됨.
TBSPACE	VARCHAR(128)	없음	없음	MQT에 대해 권장되는 테이블 스페이스.

표 239. ADVISE_MQT 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
RUN_ID	TIMESTAMP	예	FK	ADVISE_INSTANCE 테이블의 한 행의 START_TIME에 대응하는 값으로, 동일한 디자인 어드바이저 실행에 링크합니다.
REFRESH_TYPE	CHAR(1)	없음	없음	'I'(즉시) 또는 'D'(지연됨)로 설정하십시오.
EXISTS	CHAR(1)	없음	없음	MQT가 데이터베이스 카탈로그에 존재하는 경우 'Y'로 설정하십시오.

ADVISE_PARTITION 테이블

ADVISE_PARTITION 테이블에는 디자인 어드바이저가 권장하는 데이터베이스 파티션에 관한 정보가 들어있으며 파티션된 데이터베이스 환경에서만 채워질 수 있습니다.

표 240. ADVISE_PARTITION 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	없음	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	없음	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	없음	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	없음	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	없음	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	없음	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	없음	이 Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	INTEGER	없음	없음	이 Explain 정보와 관련된 패키지 내의 명령문 번호
QUERYNO	INTEGER	없음	없음	설명된 SQL문에 대한 숫자 ID입니다. CLP 또는 CLI를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 순차적으로 증분되는 값입니다. 그렇지 않으면 디폴트값은 정적 SQL문의 경우 STMTNO의 값, 동적 SQL문의 경우 1입니다.
QUERYTAG	CHAR(20)	없음	없음	설명된 각 SQL문에 대한 ID 태그입니다. CLP를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 'CLP'입니다. CLI를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 'CLI'입니다. 그렇지 않으면, 사용된 디폴트값은 공백입니다.
TBNAME	VARCHAR(128)	예	없음	테이블의 이름
TBCREATOR	VARCHAR(128)	예	없음	테이블 작성자 이름을 지정합니다.
PMID	SMALLINT	예	없음	분산 맵 ID를 지정합니다.
TBSPACE	VARCHAR(128)	예	없음	테이블이 있는 테이블 스페이스를 지정합니다.
COLNAMES	CLOB(2M)	예	없음	쉼표로 구분된 데이터베이스 파티션 지정 컬럼 이름을 지정합니다.
COLCOUNT	SMALLINT	예	없음	데이터베이스 파티션 컬럼 수를 지정합니다.
REPLICATE	CHAR(1)	예	없음	데이터베이스 파티션 복제 여부를 지정합니다.
COST	DOUBLE	예	없음	데이터베이스 파티션 사용 비용을 지정합니다.
USEIT	CHAR(1)	예	없음	데이터베이스 파티션이 EVALUATE PARTITION 모드에서 사용되는지 여부를 지정합니다. 데이터베이스 파티션은 USEIT이 'Y' 또는 'y'로 설정되는 경우 사용됩니다.
RUN_ID	TIMESTAMP	예	FK	ADVISE_INSTANCE 테이블의 한 행의 START_TIME에 대응하는 값으로, 동일한 디자인 어드바이저 실행에 링크합니다.

ADVISE_TABLE 테이블

ADVISE_TABLE 테이블은 구체화된 쿼리 테이블(MQT), 다차원 클러스터된 테이블(MDC) 및 데이터베이스 파티션에 대한 최종 디자인 어드바이저 권장사항을 사용하여 테이블 작성에 대한 DDL(Data Definition Language)을 저장합니다.

표 241. ADVISE_TABLE 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
RUN_ID	TIMESTAMP	예	FK	ADVISE_INSTANCE 테이블의 한 행의 START_TIME에 대응하는 값으로, 동일한 디자인 어드바이저 실행에 링크합니다.
TABLE_NAME	VARCHAR(128)	없음	없음	테이블 이름.
TABLE_SCHEMA	VARCHAR(128)	없음	없음	테이블 스페이스의 이름.
TABLESPACE	VARCHAR(128)	없음	없음	테이블이 작성되는 테이블 스페이스.
SELECTION_FLAG	VARCHAR(4)	없음	없음	권장사항 유형을 표시합니다. 유효한 값은 'M'(MQT), 'P'(데이터베이스 파티셔닝) 및 'C'(MDC)입니다. 이 필드는 이들 값의 모든 서브세트를 포함할 수 있습니다. 예를 들어, 'MC'는 테이블이 MQT 및 MDC 테이블로서 권장됨을 표시합니다.
TABLE_EXISTS	CHAR(1)	없음	없음	테이블이 데이터베이스 카탈로그에 존재하는 경우 'Y'로 설정하십시오.
USE_TABLE	CHAR(1)	없음	없음	테이블에 디자인 어드바이저의 권장사항이 있는 경우 'Y'로 설정하십시오.
GEN_COLUMNS	CLOB(2M)	없음	없음	이 행이 테이블 작성 DDL에서 생성된 컬럼이 필요한 MDC 권장사항을 포함하는 경우 생성된 컬럼 문자열을 포함합니다.
ORGANIZE_BY	CLOB(2M)	없음	없음	MDC 권장사항의 경우 테이블 작성 DDL의 ORGANIZE BY절을 포함합니다.
CREATION_TEXT	CLOB(2M)	없음	없음	테이블 작성 DDL을 포함합니다.
ALTER_COMMAND	CLOB(2M)	없음	없음	테이블에 대한 ALTER TABLE문을 포함합니다.

ADVISE_WORKLOAD 테이블

ADVISE_WORKLOAD 테이블은 워크로드를 구성하는 명령문을 나타냅니다.

표 242. ADVISE_WORKLOAD 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
WORKLOAD_NAME	CHAR(128)	없음	없음	이 명령문이 속하는 SQL문(워크로드) 콜렉션의 이름.
STATEMENT_NO	INTEGER	없음	없음	이 Explain 정보와 관련된 워크로드 내의 명령문 번호.
STATEMENT_TEXT	CLOB(1M)	없음	없음	SQL문의 내용.
STATEMENT_TAG	VARCHAR(256)	없음	없음	설명된 각 SQL문에 대한 ID 태그.
FREQUENCY	INTEGER	없음	없음	이 명령문이 워크로드 내에 나타나는 횟수.
IMPORTANCE	DOUBLE	없음	없음	명령문의 중요도.
WEIGHT	DOUBLE	없음	없음	명령문의 우선순위.
COST_BEFORE	DOUBLE	예	없음	권장사항이 작성되지 않는 경우 쿼리 비용(timeron 단위).
COST_AFTER	DOUBLE	예	없음	권장사항이 작성되는 경우 쿼리 비용(timeron 단위)입니다. COST_AFTER는 클러스터된 인덱스 및 다차원 클러스터링(MDC)과 관련된 것을 제외한 모든 권장사항을 반영합니다.
COMPILABLE	CHAR(17)	예	없음	명령문을 준비하려는 중에 발생한 모든 쿼리 컴파일 오류를 표시합니다. 이 컬럼이 NULL이거나 SQLCA로 시작하지 않는 경우, SQL 쿼리는 db2advis에 의해 컴파일될 수 있습니다. db2advis 또는 디자인 어드바이저가 컴파일 오류를 발견하는 경우, COMPILABLE 컬럼 값은 8바이트 길이의 SQLCA.sqlcaid 필드, 콜론(:) 및 8바이트 길이의 SQLCA.sqlstate 필드(SQL문의 리턴 코드)로 구성됩니다.

EXPLAIN_ARGUMENT 테이블

모든 경우에 EXPLAIN_ARGUMENT 테이블은 각 개별 연산자에 대해 고유한 등록 정보를 나타냅니다.

표 243. EXPLAIN_ARGUMENT 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	FK	동적 명령문이 explain될 때 수행되는 패키지 이름 또는 정적 SQL이 explain될 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	FK	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호
OPERATOR_ID	INTEGER	없음	없음	이 쿼리 내의 연산자에 대한 고유 ID
ARGUMENT_TYPE	CHAR(8)	없음	없음	이 연산자에 대한 인수의 유형
ARGUMENT_VALUE	VARCHAR(1024)	예	없음	이 연산자에 대한 인수의 값. 값이 LONG_ARGUMENT_VALUE에 있는 경우에는 널(NULL)입니다.
LONG_ARGUMENT_ VALUE	CLOB(2M)	예	없음	텍스트가 ARGUMENT_VALUE에 적합하지 않을 경우 이 연산자에 대한 인수의 값. 값이 ARGUMENT_VALUE에 있는 경우에는 널(NULL)입니다.

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값

ARGUMENT_TYPE 값	가능한 **ARGUMENT_VALUE** 값	설명
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	부분 집계 표시기
BITFLTR	INTEGER FALSE	해시 조인에 의해 사용된 비트 필터의 크기
BLD_LEVEL	DB2 빌드 ID	소스 코드의 내부 ID 문자열
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	BLOCK 레벨 잠금 모드

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
CONCACCR	이 유형의 각 행에 포함된 내용 • 이 명령문에 대한 설정 레벨: **BIND** 응용프로그램 바인드, CONCURRENT ACCESS RESOLUTION 옵션 사용 **PREP** 명령문 준비, CONCURRENT ACCESS RESOLUTION 속성 사용 • 효과적인 동시 액세스 분석: **USE CURRENTLY COMMITTED** 응용프로그램 바인드 또는 명령문 준비에 사용할 동시 액세스 분석, USE CURRENTLY COMMITTED **WAIT FOR OUTCOME** 응용프로그램 바인드 또는 명령문 준비에 사용할 동시 액세스 분석, WAIT FOR OUTCOME	이 명령문에 대한 액세스 플랜을 생성하는 데 사용된 동시 액세스 분석을 표시합니다.
CSERQY	TRUE FALSE	리모트 쿼리는 공통 부속 표현식입니다.
CSETEMP	TRUE FALSE	일반 부속 표현식 플래그에 대한 임시 테이블
CUR_COMM	TRUE	데이터베이스 구성 매개변수 **cur_commit**의 값이 DISABLE되지 않을 때 현재 커미트된 행에 액세스합니다. 이 액세스 플랜은 다음 중 하나를 사용하여 해당 명령문에 사용 가능합니다. • 바인드되거나 준비 시 USE CURRENTLY COMMITTED 옵션으로 CONCURRENT ACCESS RESOLUTION • ON의 값을 포함한 데이터베이스 구성 매개변수 **cur_commit**
DIRECT	TRUE	직접 페치 표시기
DPESTFLG	TRUE FALSE	DPNUMPRT 값이 예상 숫자를 기본으로 하는지 여부를 표시합니다. 가능한 값은 'TRUE'(DPNUMPRT는 액세스된 데이터 파티션의 예상 숫자를 표시함) 또는 'FALSE'(DPNUMPRT는 액세스된 데이터 파티션의 실제 숫자를 표시함)입니다.

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
DPLSTPRT	NONE CHARACTER	액세스된 데이터 파티션을 표시합니다. 이는 액세스된 데이터 파티션의 쉼표 분리 목록(예: 1,3,5) 또는 하이픈으로 연결된 목록(예: 1-5)입니다. 'NONE' 값은 지정된 술어를 적용한 후 데이터 파티션이 남지 않음을 의미합니다.
DPNUMPRT	INTEGER	액세스한 데이터 파티션의 실제 또는 예상 숫자를 표시합니다.
DSTSEVER	서버 이름	목적지(발송원) 서버
DUPLWARN	TRUE FALSE	경고 플래그 중복
EARLYOUT	LEFT RIGHT GROUPBY NONE	Early out 표시기 LEFT는 바깥쪽 테이블의 각 행이 안쪽 테이블의 많아야 하나의 행과 연결되어야 함을 나타냅니다. RIGHT는 안쪽 테이블의 각 행이 바깥쪽 테이블의 많아야 하나의 행과 연결되어야 함을 나타냅니다. NONE은 처리된 early out이 없음을 표시합니다. GROUPBY는 조작 그룹 때문에 early out 처리가 허용됨을 표시합니다.
ENVVAR	이 유형의 각 행에 포함된 내용 • 환경 변수 이름 • 환경 변수 값	옵티마이저에 영향을 주는 환경 변수
ERRTOL	이 유형의 각 행은 SQLSTATE 및 SQLCODE 쌍을 포함합니다.	허용되는 오류 목록
EVALUNCO	TRUE	잠금 지연을 사용한 커미트되지 않은 데이터를 평가합니다. **DB2_EVALUNCOMMITTED** 레지스트리 변수로 사용 가능합니다.
FETCHMAX	IGNORE INTEGER	FETCH 연산자에서 MAXPAGES 인수의 겹쳐쓰기 값
GREEDY	TRUE	액세스를 계획하는 데 필요한 greedy 알고리즘을 사용하는 옵티마이저
GLOBLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE NO LOCK OBTAINED SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	파티션된 테이블 오브젝트의 전역 내재된 잠금(lock intent) 정보를 표시합니다.
GROUPBYC	TRUE FALSE	Group By 컬럼이 제공되는지 여부
GROUPBYN	정수	비교 컬럼의 수

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
GROUPBYR	이 유형의 각 행에 포함된 내용 • 절별 그룹에 있는 컬럼의 순서 값(뒤에 콜론과 스페이스가 위치) • 컬럼 이름	Group By 요구
HASHCODE	24 32	해시 조인에 사용된 해시 코드의 크기(비트 단위)
INNERCOL	이 유형의 각 행에 포함된 내용 • 순서대로 된 컬럼 서수 값(뒤에 콜론과 스페이스가 위치) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	내부 순서 컬럼
INPUTXID	컨텍스트 노드 ID	INPUTXID은 XSCAN 연산자가 사용하는 입력 컨텍스트를 식별합니다.
ISCANMAX	IGNORE INTEGER	ISCAN 연산자의 MAXPAGES 인수에 대한 겹쳐쓰기 값
JN INPUT	INNER OUTER	연산자가 내부 조인 및 외부 조인을 산출할지를 설정
LCKAVOID	TRUE	잠금 방지: 행 액세스는 커미트된 데이터를 잠그지 않습니다.
LISTENER	TRUE FALSE	리스너 테이블 큐 표시기
MAXPAGES	ALL NONE INTEGER	프리페치에 대해 예상되는 최대 페이지
MAXRIDS	NONE INTEGER	각 목록 프리페치 요청에 포함될 최대 행 ID
NUMROWS	INTEGER	저장되려는 행의 수
ONEFETCH	TRUE FALSE	한 개의 페치 표시기
OUTERCOL	이 유형의 각 행에 포함된 내용 • 순서대로 된 컬럼 서수 값(뒤에 콜론과 스페이스가 위치) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	외부 순서 컬럼

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 **ARGUMENT_VALUE** 값	설명
OUTERJN	LEFT RIGHT FULL LEFT (ANTI) RIGHT (ANTI)	외부 조인 표시기
PARTCOLS	컬럼 이름	피연산자에 대한 파티션 컬럼
PREFETCH	LIST NONE SEQUENTIAL	적당한 프리 페치 유형
REOPT	ALWAYS ONCE	명령문은 매개변수 표시문자, 호스트 변수 및 특수 레지스터에 대한 bind-in 값을 사용하여 최적화됩니다.
RMTQTEXT	쿼리 텍스트	리모트 쿼리 텍스트
RNG_PROD	함수 이름	인덱스 확장 액세스용 범위 생성 함수
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	행 잠금 의도
ROWWIDTH	INTEGER	저장될 행의 너비
RSUFFIX	쿼리 텍스트	리모트 SQL 접미부
SCANDIR	FORWARD REVERSE	스캔 방향
SCANGRAN	INTEGER	SCANUNITS에 표현된, 파티션 내 병렬 처리 또는 파티션 내 병행 스캔의 세분화도
SCANSPEED	SLOW FAST	'SLOW'는 테이블에서 천천히 처리될 스캔을 표시합니다. (예를 들어, 스캔이 중첩된 루프 조인의 바깥쪽에 있는 경우). 'FAST'는 고속으로 처리될 스캔을 표시합니다. 이 정보를 사용하여 효율적인 버퍼풀 레코드 공유와 함께 스캔을 그룹화합니다.
SCANTYPE	LOCAL PARALLEL	파티션 내 병렬 처리, 인덱스 또는 테이블 스캔
SCANUNIT	ROW PAGE	파티션 내 병렬 처리, 스캔 세분화도
SHARED	TRUE	파티션 내 병렬 처리, 공유 TEMP 표시기
SKIP_INS	TRUE	건너뛰기가 삽입됩니다. 행 액세스는 커미트되지 않고 삽입된 행을 건너뜁니다. 이 동작은 **DB2_SKIPINSERTED** 레지스트리 변수 또는 현재 커미트된 시맨틱이 효과가 있을 때 가능합니다.
SKIPDKEY	TRUE	삭제된 키를 건너뜁니다. 행 액세스는 커미트되지 않고 삭제된 키를 건너뜁니다. 이 동작은 **DB2_SKIPDELETED** 레지스트리 변수로 가능합니다.

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
SKIPDROW	TRUE	삭제된 행을 건너뜁니다. 행 액세스는 커미트되지 않고 삭제된 행을 건너뜁니다. 이 동작은 **DB2_SKIPDELETED** 레지스트리 변수로 가능합니다.
SLOWMAT	TRUE FALSE	느린 구체화 플래그
SNGLPROD	TRUE FALSE	단일 에이전트에서 작성된 파티션 내 병렬 처리 정렬 또는 TEMP
SORTKEY	이 유형의 각 행에 포함된 내용 • 키 내의 컬럼 서수 값(뒤에 콜론과 스페이스가 위치) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	정렬 키 컬럼
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	파티션 내 병렬 처리, 정렬 유형
SRCSEVER	서버 이름	소스(발송처) 서버
SPILLED	INTEGER	SORT spill의 평가된 페이지 수
SQLCA	경고 정보	Explain 조작 도중 발행된 경고 및 이유 코드
STMTHEAP	INTEGER	명령문 컴파일 시작 시의 명령문 힙(heap) 크기
STREAM	TRUE FALSE	리모트 소스의 스트리밍 여부
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	테이블 잠금 의도
TEMPSIZE	INTEGER	임시 테이블 페이지 크기
THROTTLE	TRUE FALSE	조절은 조절하지 않으면 늦어지는 기타 스캔의 성능을 개선하며 같은 페이지를 다시 읽도록 강제 실행합니다. '스캔이 조절될 수 있는 경우 'TRUE'입니다. 스캔이 조절될 수 없으면 'FALSE'입니다.

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
TMPCMPRS	YES ELIGIBLE	값 YES는 압축이 적용됨을 표시합니다. 값 ELIGIBLE은 테이블이 커져도 압축이 적용될 수 있음을 표시합니다. TMPCMPRS 없음은 임시 테이블이 압축되지 않음을 표시합니다.
TQDEGREE	INTEGER	파티션 내 병렬 처리, 테이블 큐를 액세스하는 서브에이전트 수
TQMERGE	TRUE FALSE	병합(저장) 테이블 큐 표시기
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	등록 정보를 읽는 테이블 큐
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	테이블 큐 송신 등록 정보
TQ TYPE	LOCAL	파티션 내 병렬 처리, 테이블 큐
TQ_ORIGIN	ASYNCHRONY XTQ	Table Queue가 액세스 플랜에서 사용되는 이유
TRUNCTQ	INPUTOUTPUTINPUT AND OUTPUT	절단된 테이블 큐 표시기. INPUT은 테이블 큐에 대한 입력에서 절단이 발생함을 표시합니다. OUTPUT은 테이블 큐의 출력에서 절단이 발생함을 표시합니다. INPUT 및 OUTPUT은 테이블 큐에 대한 입력과 테이블 큐의 출력 모두에서 절단이 발생함을 표시합니다.
TRUNCSRT	TRUE	정렬 절단(작성되는 행의 수를 제한)
UNIQUE	TRUE FALSE	고유 표시기
UNIQKEY	이 유형의 각 행에 포함된 내용 • 키 내의 컬럼 서수 값(뒤에 콜론과 스페이스가 위치) • 컬럼 이름	고유 키 컬럼
UR_EXTRA	TRUE	정확한 분리를 위해 추가로 처리할 커미트되지 않은 읽기 분리. 이 액세스는 커서 안정성과 동일한 테이블 레벨 잠금인, 추가 테이블 레벨 잠금을 포함합니다. 또한 명령문이 실행될 때, 온라인 로드가 동시에 실행된 경우와 같이 분리 레벨은 커서 안정성을 업그레이드할 수 있습니다. 명령문 실행의 다른 부분은 상위 분리 레벨에 있는 FETCH 연산자와 같이 분리 레벨이 정확한지 확인하는 것입니다.
VISIBLE	TRUE FALSE	공유 스캔이 다른 공유 스캔을 볼 수 있는지 여부. 볼 수 있는 공유 스캔은 다른 스캔의 동작에 영향을 줄 수 있습니다. 영향을 받는 동작의 예에는 시작 위치와 조절이 포함됩니다.

표 244. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 **ARGUMENT_VALUE** 값	설명
VOLATILE	TRUE	Volatile 테이블
WRAPPING	TRUE FALSE	공유 스캔이 테이블의 레코드에서 시작하며 마지막 레코드에 도달하면 한 번 래핑할 수 있는지 여부. 랩핑은 버퍼풀 레코드가 다른 진행 중인 스캔을 공유하도록 허용합니다.
XDFOUT	DECIMAL	XDFOUT는 각 컨텍스트 노드의 XISCAN 연산자가 리턴하는 문서의 수를 예상하여 표시합니다.
XLOGID	SQL 스키마 이름 및 XML 데이터에 있는 인덱스 이름으로 구성되는 ID	XLOGID는 XISCAN 연산자에 대한 옵티마이저가 선택한 XML 데이터에 있는 인덱스를 식별합니다.
XPATH	내부 형식의 XPATH 표현식 및 결과 세트	해당 인수는 XSCAN 연산자에 의한 XPATH 표현식 평가를 표시합니다.
XPHYID	SQL 스키마 이름 및 실제 XML 데이터에 있는 인덱스 이름으로 구성되는 ID	XPHYID는 XISCAN 연산자가 사용하는 XML 데이터에 있는 인덱스와 연결되는 실제 인덱스를식별합니다.

EXPLAIN_DIAGNOSTIC 테이블

EXPLAIN_DIAGNOSTIC 테이블에는 EXPLAIN_STATEMENT 테이블의 설명된 명령문의 특정 인스턴스에 대해 생성되는 각 진단 메시지에 대한 항목이 들어있습니다.

EXPLAIN_GET_MSGS 테이블 함수는 EXPLAIN_DIAGNOSTIC 및 EXPLAIN_DIAGNOSTIC_DATA Explain 테이블을 쿼리하고 형식화된 메시지를 리턴합니다.

표 245. EXPLAIN_DIAGNOSTIC 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	PK, FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	PK, FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	PK, FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	PK, FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	PK, FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	PK, FK	이 행이 관련된 Explain 정보 레벨 가능한 값은 다음과 같습니다. **O** 원래 텍스트(사용자가 입력한 대로) **P** PLAN SELECTION
STMTNO	INTEGER	없음	PK, FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호 동적 Explain SQL문의 경우 1로 설정하십시오. 정적 SQL문의 경우 이 값은 SYSCAT.STATEMENTS 시스템 카탈로그 뷰에 사용되는 값과 동일합니다.
SECTNO	INTEGER	없음	PK, FK	이 SQL문을 포함하는 패키지의 섹션 번호입니다. 동적 Explain SQL문의 경우 이것은 런타임 시 이 명령문에 대한 섹션을 보유하는 데 사용되는 섹션 번호입니다. 정적 SQL문의 경우 이 값은 SYSCAT.STATEMENTS 시스템 카탈로그 뷰에 사용되는 값과 동일합니다.
DIAGNOSTIC_ID	INTEGER	없음	PK	EXPLAIN_STATEMENT 테이블에 있는 명령문의 특정 인스턴스에 대한 진단의 ID.
CODE	INTEGER	없음	없음	각 진단 메시지에 지정된 고유 번호입니다. 이 숫자는 진단 메시지의 전체 텍스트를 검색하기 위해 메시지 API에서 사용될 수 있습니다.

EXPLAIN_DIAGNOSTIC_DATA 테이블

EXPLAIN_DIAGNOSTIC_DATA 테이블에는 EXPLAIN_DIAGNOSTIC 테이블에 기록되는 특정 진단 메시지에 대한 메시지 토큰이 들어있습니다. 메시지 토큰은 해당 메시지를 생성한 SQL문의 실행에 특정한 추가 정보를 제공합니다.

EXPLAIN_GET_MSGS 테이블 함수는 EXPLAIN_DIAGNOSTIC 및 EXPLAIN_DIAGNOSTIC_DATA Explain 테이블을 쿼리하고 형식화된 메시지를 리턴합니다.

표 246. EXPLAIN_DIAGNOSTIC_DATA 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	FK	이 행이 관련된 Explain 정보 레벨 가능한 값은 다음과 같습니다. **O** 원래 텍스트(사용자가 입력한 대로) **P** PLAN SELECTION
STMTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호 동적 Explain SQL문의 경우 1로 설정하십시오. 정적 SQL문의 경우 이 값은 SYSCAT.STATEMENTS 시스템 카탈로그 뷰에 사용되는 값과 동일합니다.
SECTNO	INTEGER	없음	FK	이 SQL문을 포함하는 패키지의 섹션 번호입니다. 동적 Explain SQL문의 경우 이것은 런타임 시 이 명령문에 대한 섹션을 보유하는 데 사용되는 섹션 번호입니다. 정적 SQL문의 경우 이 값은 SYSCAT.STATEMENTS 시스템 카탈로그 뷰에 사용되는 값과 동일합니다.
DIAGNOSTIC_ID	INTEGER	없음	PK	EXPLAIN_STATEMENT 테이블에 있는 명령문의 특정 인스턴스에 대한 진단의 ID.
ORDINAL	INTEGER	없음	없음	전체 메시지 텍스트에서 토큰의 위치.
TOKEN	VARCHAR(1000)	예	없음	전체 메시지 텍스트에 삽입될 메시지 토큰이며, 절단될 수 있습니다.
TOKEN_LONG	BLOB(3M)	예	없음	사용 가능할 경우 자세한 정보.

EXPLAIN_INSTANCE 테이블

EXPLAIN_INSTANCE 테이블은 모든 Explain 정보에 대한 주 제어 테이블입니다. Explain 테이블에 있는 데이터의 각 행은 이 테이블에 있는 하나의 고유 행에 명시적으로 링크됩니다. EXPLAIN_INSTANCE 테이블은 설명되는 SQL문에 관한 기본 정보뿐 아니라 설명이 발생하는 환경에 관한 정보를 제공합니다.

표 247. EXPLAIN_INSTANCE 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	PK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	PK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	PK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	PK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	PK	Explain 요청의 소스 버전
EXPLAIN_OPTION	CHAR(1)	없음	없음	이 요청에 대해 요청된 Explain 정보를 표시합니다. 가능한 값은 다음과 같습니다. **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	없음	없음	EXPLAIN 스냅샷이 이 요청에 대해 작성되었는지 여부를 표시합니다. 가능한 값은 다음과 같습니다. **Y** 예. EXPLAIN 스냅샷이 작성되어 EXPLAIN_STATEMENT 테이블에 저장되었습니다. 또한 일반 Explain 정보가 캡처됩니다. **N** 아니오. Explain 스냅샷이 작성되지 않았습니다. 일반 Explain 정보가 캡처되었습니다. **O** Explain 스냅샷만 작성됩니다. 일반 Explain 정보는 캡처되지 않았습니다.
DB2_VERSION	CHAR(7)	없음	없음	이 EXPLAIN 요청을 처리한 DB2 제품의 릴리스 번호입니다. 형식은 *vv.rr.m*입니다. 여기서, **vv** 버전 번호 **rr** 릴리스 번호 **m** 유지보수 릴리스 번호
SQL_TYPE	CHAR(1)	없음	없음	Explain 인스턴스가 정적 또는 동적 SQL에 대한 것인지 여부를 표시합니다. 가능한 값은 다음과 같습니다. **S** 정적 SQL **D** 동적 SQL

표 247. EXPLAIN_INSTANCE 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
QUERYOPT	INTEGER	없음	없음	Explain 호출 시에 SQL 컴파일러가 사용한 쿼리 최적화 클래스를 표시합니다. 값은 설명되는 SQL문에 대한 SQL 컴파일러가 수행한 쿼리 최적화 레벨을 표시합니다.
BLOCK	CHAR(1)	없음	없음	SQL문을 컴파일할 때 사용된 커서 블로킹의 유형을 표시합니다. 자세한 내용은 SYSCAT.PACKAGES의 BLOCK 컬럼을 참조하십시오. 가능한 값은 다음과 같습니다. **N** 블로킹 없음 **U** 명확한 커서 블록화 **B** 전체 커서 블록화
ISOLATION	CHAR(2)	없음	없음	SQL문을 컴파일할 때 사용된 분리 유형을 표시합니다. 자세한 내용은 SYSCAT.PACKAGES의 ISOLATION 컬럼을 참조하십시오. 가능한 값은 다음과 같습니다. **RR** 반복 읽기 **RS** 읽기 안정성 **CS** 커서 안정성 **UR** 언커미트 읽기
BUFFPAGE	INTEGER	없음	없음	Explain 호출 시에 BUFFPAGE 데이터베이스 구성 설정의 값을 포함합니다.
AVG_APPLS	INTEGER	없음	없음	Explain 호출 시에 **avg_appls** 구성 매개변수의 값을 포함합니다.
SORTHEAP	INTEGER	없음	없음	Explain 호출 시에 **sortheap** 데이터베이스 구성 매개변수의 값을 포함합니다.
LOCKLIST	INTEGER	없음	없음	Explain 호출 시에 **locklist** 데이터베이스 구성 매개변수의 값을 포함합니다.
MAXLOCKS	SMALLINT	없음	없음	Explain 호출 시에 **maxlocks** 데이터베이스 구성 매개변수의 값을 포함합니다.
LOCKS_AVAIL	INTEGER	없음	없음	각 사용자에 대해 옵티마이저가 사용할 수 있는 것으로 가정되는 잠금 수를 포함합니다. (**locklist** 및 **maxlocks**에서 파생됩니다.)
CPU_SPEED	DOUBLE	없음	없음	Explain 호출 시에 **cpuspeed** 데이터베이스 관리 프로그램 구성 매개변수의 값을 포함합니다.
REMARKS	VARCHAR(254)	예	없음	사용자 제공 주석.
DBHEAP	INTEGER	없음	없음	Explain 호출 시에 **dbheap** 데이터베이스 구성 매개변수의 값을 포함합니다.
COMM_SPEED	DOUBLE	없음	없음	Explain 호출 시에 **comm_bandwidth** 데이터베이스 구성 매개변수의 값을 포함합니다.

표 247. EXPLAIN_INSTANCE 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
PARALLELISM	CHAR(2)	없음	없음	가능한 값은 다음과 같습니다. • N = 병렬 처리 없음 • P = 파티션 내 병렬 처리 • IP = 파티션 간 병렬 처리 • BP = 파티션 내 병렬 처리 및 파티션 간 병렬 처리
DATAJOINER	CHAR(1)	없음	없음	가능한 값은 다음과 같습니다. • N = 페더레이트되지 않은 시스템 플랜 • Y = 페더레이티드 시스템 플랜

EXPLAIN_OBJECT 테이블

EXPLAIN_OBJECT 테이블은 SQL문을 만족하기 위해 생성된 액세스 플랜에서 필요한 데이터 오브젝트를 식별합니다.

표 248. EXPLAIN_OBJECT 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	FK	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호
OBJECT_SCHEMA	VARCHAR(128)	없음	없음	이 오브젝트가 속하는 스키마.
OBJECT_NAME	VARCHAR(128)	없음	없음	오브젝트 이름.
OBJECT_TYPE	CHAR(2)	없음	없음	오브젝트의 유형에 대한 설명 레이블.
CREATE_TIME	TIMESTAMP	예	없음	오브젝트 작성 시간. 테이블 함수의 경우 널(NULL).
STATISTICS_TIME	TIMESTAMP	예	없음	이 오브젝트에 대한 통계를 마지막으로 갱신한 시간. 이 오브젝트에 대한 통계가 없는 경우 널(NULL).
COLUMN_COUNT	SMALLINT	없음	없음	오브젝트의 컬럼 수.
ROW_COUNT	INTEGER	없음	없음	이 오브젝트의 추정된 행의 수.
WIDTH	INTEGER	없음	없음	바이트 단위의 오브젝트 평균 길이. 인덱스의 경우 -1로 설정합니다.
PAGES	BIGINT	없음	없음	오브젝트가 버퍼 풀에서 차지하는 개략적인 페이지 수입니다. 테이블 함수의 경우 -1로 설정합니다.
DISTINCT	CHAR(1)	없음	없음	오브젝트의 행이 구별인지 여부(즉, 중복이 있는지 여부)를 표시합니다. 가능한 값은 다음과 같습니다. **Y** 예 **N** 없음
TABLESPACE_NAME	VARCHAR(128)	예	없음	이 오브젝트가 저장되는 테이블 스페이스의 이름이며, 테이블 스페이스가 관련없는 경우 널(NULL)로 설정됩니다.
OVERHEAD	DOUBLE	없음	없음	지정된 테이블 스페이스로의 단일 무작위 입출력에 대한 전체 예상 오버헤드(밀리초)입니다. 제어기 오버헤드, 디스크 탐색과 대기 시간을 포함합니다. 테이블 스페이스가 포함되지 않을 경우 -1로 설정합니다.
TRANSFER_RATE	DOUBLE	없음	없음	지정된 테이블 스페이스에서 데이터 페이지를 읽는 예상 시간(밀리초)입니다. 테이블 스페이스가 포함되지 않을 경우 -1로 설정합니다.

표 248. EXPLAIN_OBJECT 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
PREFETCHSIZE	INTEGER	없음	없음	프리페치가 수행될 때 읽을 데이터 페이지 수입니다. 테이블 함수의 경우 -1로 설정합니다.
EXTENTSIZE	INTEGER	없음	없음	데이터 페이지에 있는 Extent 크기입니다. 이 많은 페이지가 다음 컨테이너로 전환하기 전에 테이블 스페이스의 한 컨테이너에 기록됩니다. 테이블 함수의 경우 -1로 설정합니다.
CLUSTER	DOUBLE	없음	없음	인덱스를 사용한 데이터 클러스터링 등급입니다. >= 1인 경우 CLUSTERRATIO입니다. >= 0 및 < 1인 경우 CLUSTERFACTOR입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
NLEAF	BIGINT	없음	없음	이 인덱스 오브젝트의 값이 차지하는 리프 페이지 수입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
NLEVELS	INTEGER	없음	없음	이 인덱스 오브젝트의 트리에 있는 인덱스 레벨 수입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
FULLKEYCARD	BIGINT	없음	없음	이 인덱스 오브젝트에 포함된 구별 전체 키 값의 수입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
OVERFLOW	BIGINT	없음	없음	테이블에서 오버플로우 레코드의 총수입니다. 인덱스, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
FIRSTKEYCARD	BIGINT	없음	없음	첫 번째 구별 키 값의 수입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
FIRST2KEYCARD	BIGINT	없음	없음	인덱스의 첫 번째 {2,3,4} 컬럼을 사용하는 첫 번째 구별 키 값의 수입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
FIRST3KEYCARD	BIGINT	없음	없음	
FIRST4KEYCARD	BIGINT	없음	없음	
SEQUENTIAL_PAGES	BIGINT	없음	없음	리프 페이지 사이의 큰 갭이 없거나 소수인 인덱스 키 순서에서 디스크에 위치한 리프 페이지 수입니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
DENSITY	INTEGER	없음	없음	인덱스로 채워지는 페이지 범위에 있는 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로, 퍼센트(0 - 100 범위의 정수)로 표현됩니다. 테이블, 테이블 함수, 또는 이 통계를 사용할 수 없는 경우 -1로 설정됩니다.
STATS_SRC	CHAR(1)	없음	없음	통계의 소스를 표시합니다. 단일 노드에서 오는 경우 1로 설정합니다.
AVERAGE_SEQUENCE_GAP	DOUBLE	없음	없음	시퀀스 사이의 갭.
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	없음	없음	인덱스를 사용하여 페치할 때 시퀀스 사이의 갭.
AVERAGE_SEQUENCE_PAGES	DOUBLE	없음	없음	시퀀스에서 액세스 가능한 인덱스 페이지의 평균 수.

표 248. EXPLAIN_OBJECT 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	없음	없음	인덱스를 사용하여 페치할 때 순차적으로 액세스할 수 있는 테이블 페이지의 평균 수.
AVERAGE_RANDOM_PAGES	DOUBLE	없음	없음	순차 페이지 액세스 사이의 평균 무작위 인덱스 페이지 수.
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	없음	없음	인덱스를 사용하여 페치할 때 순차 페이지 액세스 사이의 평균 무작위 테이블 페이지 수.
NUMRIDS	BIGINT	없음	없음	인덱스에 있는 행 ID의 총수.
NUMRIDS_DELETED	BIGINT	없음	없음	인덱스에 있는 가상 삭제 행 ID의 총수.
NUM_EMPTY_LEAFS	BIGINT	없음	없음	인덱스에 있는 비어 있는 리프 페이지의 총수.
ACTIVE_BLOCKS	BIGINT	없음	없음	테이블에 있는 활성 다차원 클러스터링(MDC) 블록의 총수.
NUM_DATA_PART	INTEGER	없음	없음	파티션된 테이블에 대한 데이터 파티션 수. 테이블이 파티션되지 않은 경우 1로 설정됩니다.

표 249. 가능한 OBJECT_TYPE 값

값	설명
IX	인덱스
NK	별칭
RX	RCT 인덱스
DP_TABLE	데이터 파티션 테이블
TA	테이블
TF	테이블 함수
+A	컴파일러 참조 별명
+C	컴파일러 참조 제한조건
+F	컴파일러 참조 함수
+G	컴파일러 참조 트리거
+N	컴파일러 참조 별칭
+T	컴파일러 참조 테이블
+V	컴파일러 참조 뷰

EXPLAIN_OPERATOR 테이블

EXPLAIN_OPERATOR 테이블에는 쿼리 컴파일러가 쿼리 명령문을 충족시키는 데 필요한 모든 연산자가 들어 있습니다.

표 250. EXPLAIN_OPERATOR 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	FK	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호
OPERATOR_ID	INTEGER	없음	없음	이 쿼리 내의 연산자에 대한 고유 ID
OPERATOR_TYPE	CHAR(6)	없음	없음	연산자의 유형에 대한 설명 레이블
TOTAL_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜 실행의 평가된 누적 총 비용(timeron 단위)
IO_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜 실행의 평가된 누적 I/O 비용(데이터 페이지 입출력 단위)
CPU_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜 실행의 평가된 누적 CPU 비용(지시 단위)
FIRST_ROW_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜에 대한 첫 번째 행 페치의 평가 누적된 비용(timerons 단위). 이 값에는 필수 초기 오버헤드가 포함되어 있습니다.
RE_TOTAL_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜에 대한 마지막 행 페치의 평가된 누적 비용(timerons 단위)
RE_IO_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜에 대한 마지막 행 페치의 평가된 누적 I/O 비용(데이터 페이지 입출력 단위)
RE_CPU_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜의 다음 행 페치에 대해 계산된 누적 CPU 비용(명령어 단위)
COMM_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜 실행의 평가 누적 통신 비용(TCP/IP 프레임 단위)
FIRST_COMM_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜에 대한 첫 번째 행 페치의 평가된 누적 통신 비용(TCP/IP 프레임 단위). 이 값에는 필수 초기 오버헤드가 포함되어 있습니다.
BUFFERS	DOUBLE	없음	없음	이 연산자 및 입력을 위한 예상 버퍼 요구사항
REMOTE_TOTAL_COST	DOUBLE	없음	없음	리모트 데이터베이스에서 조작 수행의 예상 누적 총 비용(timeron 단위)

표 250. EXPLAIN_OPERATOR 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
REMOTE_COMM_COST	DOUBLE	없음	없음	이 연산자를 포함할 때까지 선택된 액세스 플랜 실행의 예상 누적 통신 비용

표 251. OPERATOR_TYPE 값

값	설명
DELETE	삭제
EISCAN	확장된 인덱스 스캔
FETCH	페치
FILTER	필터 행
GENROW	생성 행
GRPBY	그룹 기준
HSJOIN	해시 조인
INSERT	삽입
IXAND	동적 비트 맵핑 인덱스 ANDing
IXSCAN	관계형 인덱스 스캔
MSJOIN	병합 스캔 조인
NLJOIN	중첩 루프 조인
RETURN	결과
RIDSCN	행 ID(RID) 스캔
RPD	리모트 푸시다운
SHIP	리모트 시스템으로 쿼리 발송
SORT	정렬
TBSCAN	테이블 스캔
TEMP	임시 테이블 구조
TQ	테이블 큐
UNION	단위
UNIQUE	중복 제거
UPDATE	갱신
XISCAN	XML 데이터에 대한 인덱스 스캔
XSCAN	XML 문서 탐색 스캔
XANDOR	XML 데이터에 대한 인덱스 ANDing 및 ORing

EXPLAIN_PREDICATE 테이블

EXPLAIN_PREDICATE 테이블은 특정 연산자가 적용하는 술어를 식별합니다.

표 252. EXPLAIN_PREDICATE 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	FK	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호
OPERATOR_ID	INTEGER	없음	없음	이 쿼리 내의 연산자에 대한 고유 ID
PREDICATE_ID	INTEGER	없음	없음	지정된 연산자에 대한 이 술어의 고유 ID입니다. 값 ″-1″은 옵티마이저 오브젝트가 아니고 옵티마이저 플랜에 존재하지 않는 Explain 도구에 의해 구성되는 연산자 술어에 대해 표시됩니다.
HOW_APPLIED	CHAR(10)	없음	없음	지정된 연산자가 술어를 사용 중인 방법.
WHEN_EVALUATED	CHAR(3)	없음	없음	이 술어에서 사용되는 서브쿼리가 평가되는 시기를 표시합니다. 가능한 값은 다음과 같습니다. **공백** 이 술어는 서브쿼리를 포함하지 않습니다. **EAA** 이 술어에서 사용되는 서브쿼리가 응용프로그램에서 평가(EAA)됩니다. 즉, 술어가 적용될 때 지정된 연산자에 의해 처리되는 모든 행에 대해 재평가됩니다. **EAO** 이 술어에서 사용되는 서브쿼리가 열릴 때 평가(EAO)됩니다. 즉, 지정된 연산자에 대해 한 번만 재평가되며 결과가 각 행에 대한 술어의 응용프로그램에서 재사용됩니다. **MUL** 이 술어에 서브쿼리의 둘 이상의 유형이 있습니다.
RELOP_TYPE	CHAR(2)	없음	없음	술어에 사용된 관계 연산자 유형.

표 252. EXPLAIN_PREDICATE 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
SUBQUERY	CHAR(1)	없음	없음	서브쿼리의 데이터 스트림이 이 술어에 필요한지 여부입니다. 필요한 서브쿼리 스트림이 여러 개일 수 있습니다. 가능한 값은 다음과 같습니다. **N** 서브쿼리 스트림이 필요하지 않음 **Y** 하나 이상의 서브쿼리 스트림이 필요함
FILTER_FACTOR	DOUBLE	없음	없음	이 술어가 자격을 규정할 행의 예상 소수입니다. 값 ″-1″은 FILTER_FACTOR를 적용할 수 없을 때 표시됩니다. FILTER_FACTOR는 옵티마이저 오브젝트가 아니고 옵티마이저 플랜에 존재하지 않는 Explain 도구에 의해 구성되는 연산자 술어에 적용할 수 없습니다.
PREDICATE_TEXT	CLOB(2M)	예	없음	SQL 또는 XQuery 명령문의 내부 표현으로부터 재작성되는 술어의 텍스트입니다. 호스트 변수, 특수 레지스터 또는 매개변수 표시문자의 값이 명령문 컴파일 중에 사용되는 경우, 이 값이 주석으로 묶은 술어 텍스트의 끝에 나타납니다. 명령문이 DBADM 권한이 있는 사용자에 의해 실행되는 경우 또는 DB2 레지스트리 변수 DB2_VIEW_REOPT_VALUES가 YES로 설정되는 경우에만 값이 EXPLAIN_PREDICATE 테이블에 저장됩니다. 그 외에는 빈 주석이 술어 텍스트의 끝에 나타납니다. 사용 가능하지 않을 경우 널(NULL)입니다.
RANGE_NUM	INTEGER	예	없음	데이터 파티션 제거 술어의 범위로서, 범위별로 데이터 파티션 제거에 사용되는 술어의 그룹화를 가능하게 합니다. 다른 모든 술어 유형의 경우 널(NULL) 값입니다.

표 253. 가능한 HOW_APPLIED 값

값	설명
BSARG	모든 블록에 대해 한 번만 Sargable 술어로 평가됨
DPSTART	데이터 파티션 제거에서 사용되는 시작 키 술어
DPSTOP	데이터 파티션 제거에서 사용되는 중지 키 술어
JOIN	테이블을 조인하는 데 사용됨
RESID	레지듀얼(Residual) 술어로 평가됨
SARG	인덱스 또는 데이터 페이지에 대한 Sargable 술어로 평가됨
START	시작 조건으로 사용됨
STOP	중지 조건으로 사용됨

표 254. 가능한 RELOP_TYPE 값

값	설명
공백	적용할 수 없음
EQ	등호
GE	크거나 같음
GT	보다 큼
IN	목록에 있음
LE	작거나 같음
LK	비슷함
LT	보다 작음
NE	같지 않음
NL	널(NULL)임
NN	널(Null)이 아님

EXPLAIN_STATEMENT 테이블

EXPLAIN_STATEMENT 테이블은 Explain 정보의 다양한 레벨에 존재하므로 이는 SQL문의 텍스트를 포함합니다. 사용자가 입력한 원래 SQL문은 SQL문을 만족할 액세스 플랜을 선택하는 데 사용된 버전(옵티마이저가 사용한)과 함께 이 테이블에 저장됩니다. 근래 버전은 SQL 컴파일러로 판별된 추가 술어로 재작성 및/또는 확장될 수 있으므로 원본에 대하여 미묘한 유사점을 감당할 수 있습니다. 또한, 명령문 집중기가 사용 가능하고 명령문이 명령문 집중기의 결과로 변경된 경우, 유효한 SQL문도 이 테이블에 저장됩니다. 이 명령문은 리터럴 값이 시스템 생성 이름 지정된 매개변수 표시 문자로 교체되는 것을 제외하고 원래 명령문과 유사합니다. 이러한 경우 플랜 정보는 유효한 명령문을 기반으로 합니다.

표 255. EXPLAIN_STATEMENT 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	PK, FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	PK, FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	PK, FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	PK, FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	PK	이 행과 관련된 Explain 정보의 레벨 가능한 값은 다음과 같습니다. **O** 원래 텍스트(사용자가 입력한) **P** PLAN SELECTION **E** 유효한 SQL 텍스트
STMTNO	INTEGER	없음	PK	이 Explain 정보와 관련된 패키지 내의 명령문 번호. 동적 Explain SQL문의 경우 1로 설정하십시오. 정적 SQL문의 경우, 이 값은 SYSCAT.STATEMENTS 카탈로그 뷰에 사용되는 값과 동일합니다.
SECTNO	INTEGER	없음	PK	이 SQL문을 포함하는 패키지의 섹션 번호입니다. 동적 Explain SQL문의 경우, 이는 런타임 시 이 명령문에 대한 섹션을 보유하는 데 사용되는 섹션 번호입니다. 정적 SQL문의 경우, 이 값은 SYSCAT.STATEMENTS 카탈로그 뷰에 사용되는 값과 동일합니다.
QUERYNO	INTEGER	없음	없음	설명된 SQL문에 대한 숫자 ID입니다. CLP 또는 CLI를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 순차적으로 증분되는 값입니다. 그렇지 않으면 디폴트값은 정적 SQL문의 경우 STMTNO의 값, 동적 SQL문의 경우 1입니다.

표 255. EXPLAIN_STATEMENT 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
QUERYTAG	CHAR(20)	없음	없음	설명된 각 SQL문에 대한 ID 태그입니다. CLP를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 'CLP'입니다. CLI를 통해 발행되는 동적 SQL문(EXPLAIN SQL문은 제외)의 경우 디폴트값은 'CLI'입니다. 그렇지 않으면, 사용된 디폴트값은 공백입니다.
STATEMENT_TYPE	CHAR(2)	없음	없음	설명 중인 쿼리의 유형에 대한 설명 레이블입니다. 가능한 값은 다음과 같습니다. **CL** 호출 **CP** 복합 SQL(동적) **D** 삭제 **DC** 현재 커서 삭제 **I** 삽입 **M** 병합 **S** 선택 **SI** 무결성 또는 새로 고침 테이블 설정 **U** 갱신 **UC** 현재 커서 갱신
UPDATABLE	CHAR(1)	없음	없음	이 명령문이 갱신 가능한 것으로 간주되는지 여부를 표시합니다. 이는 특히 잠재적으로 갱신 가능하도록 판별될 수 있는 SELECT문과 관련이 있습니다. 가능한 값은 다음과 같습니다. **' '** 해당되지 않음(공백) **N** 없음 **Y** Yes
DELETABLE	CHAR(1)	없음	없음	이 명령문이 삭제 가능한 것으로 간주되는지 여부를 표시합니다. 이는 특히 잠재적으로 삭제 가능하도록 판별될 수 있는 SELECT문과 관련이 있습니다. 가능한 값은 다음과 같습니다. **' '** 해당되지 않음(공백) **N** 없음 **Y** Yes
TOTAL_COST	DOUBLE	없음	없음	이 명령문에 대하여 선택된 액세스 플랜 실행의 추정된 총 비용(timeron 단위)으로, 현재 선택된 액세스 플랜이 없으므로 EXPLAIN_LEVEL이 *O*(원래 텍스트)인 경우 0(영)으로 설정합니다.

표 255. EXPLAIN_STATEMENT 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
STATEMENT_TEXT	CLOB(2M)	없음	없음	텍스트 또는 설명 중인 SQL문 텍스트의 분할 영역 Explain의 플랜 선택 레벨에 표시된 텍스트가 내부 표현에서 재구성되었으며 이는 특성 면에서 SQL과 유사합니다. 즉, 재구성된 명령문은 올바른 SQL 구문을 따르는 것을 보장할 수 없습니다.
SNAPSHOT	BLOB(10M)	Yes	없음	표시된 Explain_Level에서 이 SQL문에 대한 내부 표현의 스냅샷입니다. 이 컬럼은 DB2 Visual Explain와 함께 사용하도록 되어 있습니다. EXPLAIN_LEVEL이 *0*(원래 명령문)인 경우 컬럼은 널(NULL)로 설정됩니다. 명령문의 이러한 특정 버전이 캡쳐될 때 액세스 플랜이 선택되지 않았기 때문입니다.
QUERY_DEGREE	INTEGER	없음	없음	Explain 호출 시 파티션 내 병렬 처리의 등급을 표시합니다. 원래 명령문의 경우, 이는 파티션 내 병렬 처리의 방향지정 등급을 포함합니다. PLAN SELECTION의 경우, 이는 사용할 플랜에서 생성된 파티션 내 병렬 처리의 등급을 포함합니다.

EXPLAIN_STREAM 테이블

EXPLAIN_STREAM 테이블은 개별 연산자와 데이터 오브젝트 사이의 입력 및 출력 데이터 스트림을 나타냅니다. 데이터 오브젝트 자체는 EXPLAIN_OBJECT 테이블에 나타납니다. 데이터 스트림에 관련되는 연산자는 EXPLAIN_OPERATOR 테이블에 있습니다.

표 256. EXPLAIN_STREAM 테이블. PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	없음	FK	이 Explain 요청 개시자의 권한 부여 ID
EXPLAIN_TIME	TIMESTAMP	없음	FK	Explain 요청에 대한 시작 시간
SOURCE_NAME	VARCHAR(128)	없음	FK	동적 명령문이 Explain된 실행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	없음	FK	Explain 요청 소스의 스키마 또는 규정자
SOURCE_VERSION	VARCHAR(64)	없음	FK	Explain 요청의 소스 버전
EXPLAIN_LEVEL	CHAR(1)	없음	FK	이 행과 관련된 Explain 정보의 레벨
STMTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	없음	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호
STREAM_ID	INTEGER	없음	없음	지정된 연산자에서 이 데이터 스트림의 고유 ID입니다.
SOURCE_TYPE	CHAR(1)	없음	없음	데이터 스트림의 소스. **O** 연산자 **D** 데이터 오브젝트
SOURCE_ID	SMALLINT	없음	없음	이 데이터 스트림의 소스인 이 쿼리에서 연산자의 고유 ID입니다. SOURCE_TYPE이'D'인 경우 -1로 설정됩니다.
TARGET_TYPE	CHAR(1)	없음	없음	데이터 스트림의 목표를 표시합니다. **O** 연산자 **D** 데이터 오브젝트
TARGET_ID	SMALLINT	없음	없음	이 데이터 스트림의 목표인 이 쿼리에서 연산자의 고유 ID입니다. TARGET_TYPE이'D'인 경우 -1로 설정됩니다.
OBJECT_SCHEMA	VARCHAR(128)	예	없음	영향을 받은 데이터 오브젝트가 속하는 스키마입니다. SOURCE_TYPE 및 TARGET_TYPE 둘 다 'O'인 경우 널(NULL)로 설정됩니다.
OBJECT_NAME	VARCHAR(128)	예	없음	데이터 스트림의 주제인 오브젝트의 이름입니다. SOURCE_TYPE 및 TARGET_TYPE 둘 다 'O'인 경우 널(NULL)로 설정됩니다.
STREAM_COUNT	DOUBLE	없음	없음	데이터 스트림의 평가된 카디널리티(cardinality).
COLUMN_COUNT	SMALLINT	없음	없음	데이터 스트림 내의 컬럼 수.
PREDICATE_ID	INTEGER	없음	없음	이 스트림이 술어에 대한 서브쿼리의 파트인 경우 술어 ID가 여기에 반영되며, 그렇지 않으면 컬럼이 -1로 설정됩니다.

표 256. EXPLAIN_STREAM 테이블 (계속). PK는 컬럼이 기본 키의 일부임을 나타내며, FK는 컬럼이 외부 키의 일부임을 나타냅니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	키	설명
COLUMN_NAMES	CLOB(2M)	예	없음	이 컬럼은 이 스트림에 포함되는 컬럼의 이름 및 순서 지정 정보를 포함합니다. 이름은 다음 형식입니다. `NAME1(A)+NAME2(D)+NAME3+NAME4` 여기서 *(A)*는 오름차순의 컬럼을 표시하고, *(D)*는 내림차순의 컬럼을 표시하며, 순서 지정 없음 정보는 컬럼이 정렬되지 않거나 순서 지정이 관련없음을 표시합니다.
PMID	SMALLINT	없음	없음	분산 맵 ID.
SINGLE_NODE	CHAR(5)	예	없음	이 데이터 스트림이 단일 또는 다중 데이터베이스 파티션에 있는지 여부를 표시합니다. **MULT** 다중 데이터베이스 파티션에서 **COOR** 코디네이터 노드에서 **HASH** 해싱을 사용하여 방향지정 **RID** 행 ID를 사용하여 방향지정 **FUNC** 함수(HASHEDVALUE() 또는 DBPARTITIONNUM())를 사용한 방향지정 **CORR** 상관 값을 사용한 방향지정 **숫자** 사전 판별되는 단일 노드로 방향지정
PARTITION_COLUMNS	CLOB(2M)	예	없음	이 데이터 스트림이 분산되는 컬럼의 목록.
SEQUENCE_SIZES	CLOB(2M)	예	없음	XML 컬럼에 대한 예상 시퀀스 크기를 나열하거나, 데이터 스트림의 모든 비XML 컬럼의 경우 "NA"(적용할 수 없음)를 표시합니다. 데이터 스트림에 최소 하나의 XML 컬럼이 있는 경우 널(NULL)로 설정됩니다.

부록 J. Explain 레지스터 값

다음은 CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOP 특수 레지스터 값의 상호작용과 이 값과 PREP 및 BIND 명령과의 상호작용에 대한 설명입니다.

동적 SQL을 사용할 때 CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값은 다음과 같이 상호 작용합니다.

표 257. Explain 특수 레지스터 값의 상호 작용(동적 SQL)

	EXPLAIN MODE 값					
EXPLAIN SNAPSHOT 값	**NO**	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
NO	• 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 채워짐. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 인덱스 권장.	• Explain 테이블이 채워짐. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 인덱스 평가.
YES	• Explain 스냅샷 작성. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 인덱스 권장.	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 인덱스 평가.

표 257. Explain 특수 레지스터 값의 상호 작용(동적 SQL) (계속)

EXPLAIN SNAPSHOT 값	EXPLAIN MODE 값					
	NO	YES	EXPLAIN	REOPT	RECOMMEND INDEXES	EVALUATE INDEXES
EXPLAIN	• Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴되지 않음(동적 또는 증분식 바인드문이 실행되지 않음).	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 인덱스 권장.	• Explain 테이블이 채워짐. • Explain 스냅샷 작성. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 인덱스 평가.
REOPT	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴되지 않음(동적 또는 증분식 바인드문이 실행되지 않음).	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• Explain 테이블이 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴되지 않음(동적 또는 증분식 바인드문이 실행되지 않음). • 인덱스 권장.	• Explain 테이블이 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴되지 않음(동적 또는 증분식 바인드문이 실행되지 않음). • 인덱스 평가.

CURRENT EXPLAIN MODE 특수 레지스터는 동적 SQL에 대해 다음 방법으로 EXPLAIN 바인드 옵션과 상호 작용합니다.

표 258. EXPLAIN 바인드 옵션 및 CURRENT EXPLAIN MODE의 상호 작용

	EXPLAIN 바인드 옵션 값			
EXPLAIN MODE 값	**NO**	**YES**	**REOPT**	**ALL**
NO	• 쿼리 결과가 리턴됨.	• Explain 테이블이 정적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴됨.
YES	• 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴됨.	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴됨.
EXPLAIN	• 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).
REOPT	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴됨.

표 258. EXPLAIN 바인드 옵션 및 CURRENT EXPLAIN MODE의 상호 작용 (계속)

	EXPLAIN 바인드 옵션 값			
EXPLAIN MODE 값	**NO**	**YES**	**REOPT**	**ALL**
RECOMMEND INDEXES	• 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 권장 인덱스	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 권장 인덱스	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 권장 인덱스	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 권장 인덱스
EVALUATE INDEXES	• 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 평가 인덱스	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 평가 인덱스	• 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 정적 SQL에 대해 채워짐. • 명령문이 실행 시간에 재최적화를 규정할 때 Explain 테이블이 동적 SQL에 대해 채워짐. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 평가 인덱스	• Explain 테이블이 정적 SQL에 대해 채워짐. • 동적 SQL에 대한 데이터가 채워지는 Explain 테이블 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음). • 평가 인덱스

CURRENT EXPLAIN SNAPSHOT 특수 레지스터는 동적 SQL에 대해 다음 방법으로 EXPLSNAP 바인드 옵션과 상호 작용합니다.

표 259. EXPLSNAP 바인드 옵션 및 CURRENT EXPLAIN SNAPSHOT의 상호 작용

EXPLAIN SNAPSHOT 값	EXPLSNAP 바인드 옵션 값			
	NO	**YES**	**REOPT**	**ALL**
NO	• 쿼리 결과가 리턴됨.	• 정적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 정적 SQL에 대해 Explain 스냅샷이 작성됨. • 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• 정적 SQL에 대해 수행되는 Explain 스냅샷 • 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴됨.
YES	• 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴됨.	• 정적 SQL에 대해 수행되는 Explain 스냅샷 • 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 정적 SQL에 대해 Explain 스냅샷이 작성됨. • 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• 정적 SQL에 대해 수행되는 Explain 스냅샷 • 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴됨.
EXPLAIN	• 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 정적 SQL에 대해 수행되는 Explain 스냅샷 • 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 명령문이 실행 시간에 재최적화를 규정할 때 정적 SQL에 대해 Explain 스냅샷이 작성됨. • 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).	• 정적 SQL에 대해 수행되는 Explain 스냅샷 • 동적 SQL에 대해 수행되는 Explain 스냅샷 • 쿼리 결과가 리턴되지 않음(동적문이 실행되지 않음).

표 259. EXPLSNAP 바인드 옵션 및 CURRENT EXPLAIN SNAPSHOT의 상호 작용 (계속)

EXPLAIN SNAPSHOT 값	EXPLSNAP 바인드 옵션 값			
	NO	**YES**	**REOPT**	**ALL**
REOPT	• 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 정적 SQL에 대해 Explain 스냅샷이 작성됨. • 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 정적 SQL에 대해 Explain 스냅샷이 작성됨. • 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.	• 명령문이 실행 시간에 재최적화를 규정할 때 정적 SQL에 대해 Explain 스냅샷이 작성됨. • 명령문이 실행 시간에 재최적화를 규정할 때 동적 SQL에 대해 Explain 스냅샷이 작성됨. • 쿼리 결과가 리턴됨.

부록 K. 예외 테이블

예외 테이블은 IMMEDIATE CHECKED 옵션이 있는 SET INTEGRITY문을 사용하여 점검되도록 지정된 테이블의 정의를 모방하는 사용자 작성 테이블입니다. 이는 점검 중인 테이블의 제한조건을 위반하는 행의 사본을 저장하는 데 사용됩니다.

로드 유틸리티가 사용하는 예외 테이블은 여기에서 설명되는 예외 테이블과 동일하므로 SET INTEGRITY문으로 점검하는 동안 재사용할 수 있습니다.

문자열 변환 규칙

예외 테이블 작성 규칙은 다음과 같습니다.

- 보안 규정이 테이블을 보호하는 경우 동일한 보안 규정이 예외 테이블을 보호해야 합니다.
- 예외 테이블의 처음 『n』개 컬럼은 점검 중인 테이블의 컬럼과 같습니다. 이름, 데이터 유형 및 길이를 포함하는 모든 컬럼 속성은 동일해야 합니다. 보호된 컬럼의 경우 두 테이블 모두는 동일한 컬럼 보호 보안 레이블을 가져야 합니다.
- 예외 테이블의 모든 컬럼에는 제한조건과 트리거가 없어야 합니다. 제한조건에는 삽입시 오류를 발생시킬 수 있는 고유 인덱스 제한조건뿐만 아니라 참조 무결성 및 점검 제한조건이 포함됩니다.
- 예외 테이블의 『(n+1)』 컬럼은 선택적인 TIMESTAMP 컬럼입니다. 이는 데이터 점검을 위해 SET INTEGRITY문을 발행하기 전에 예외 테이블 내의 행이 삭제되지 않은 경우에 같은 테이블에 대해 SET INTEGRITY문으로 점검의 후속 호출을 식별하는 데 사용됩니다.시간소인 정밀도는 0에서 12까지의 값일 수 있으며 지정된 값은 CURRENT TIMESTAMP 특수 레지스터의 결과가 됩니다.
- 『(n+2)』 컬럼은 CLOB(32K) 유형 이상이어야 합니다. 이 컬럼은 선택적이지만 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 행만 복사됩니다.
- 예외 테이블은 『(n+1)』과 『(n+2)』 컬럼 모두로 작성되어야 합니다.
- 위의 추가 컬럼에 대한 특정 이름을 강제하는 것은 없습니다. 그러나 유형 스펙은 정확히 따라야 합니다.
- 추가 컬럼은 허용되지 않습니다.
- 원래 테이블에 생성된 컬럼(IDENTITY 등록 정보 포함)이 있는 경우, 예외 테이블의 해당 컬럼은 생성된 등록 정보를 지정하지 않아야 합니다.
- 데이터를 점검하기 위해 SET INTEGRITY문을 호출한 사용자는 예외 테이블에 INSERT 특권이 있어야 합니다.

- 예외 테이블은 데이터 파티션된 테이블, 범위 클러스터된 테이블 또는 접속 해제된 테이블일 수 없습니다.
- 예외 테이블은 구체화된 쿼리 테이블 또는 스테이징 테이블일 수 없습니다.
- 예외 테이블에는 종속되는 즉시 새로 고침 구체화된 쿼리 테이블 또는 종속되는 즉시 전파 스테이징 테이블이 있을 수 없습니다.

『메시지』 컬럼에 있는 정보에는 다음의 구조가 있습니다.

표 260. 예외 테이블 메시지 컬럼 구조

필드 번호	목차	크기	주석
1	제한조건 위반 수	5바이트	'0'으로 채워 오른쪽 정렬
2	첫 번째 제한조건 위반 유형	1바이트	'K' - 점검 제한조건 위반 'F' - 외부 키 위반 'G' - 생성된 컬럼 위반 'I' - 고유 인덱스 위반[a] 'D' - 연쇄 삭제 위반 'P' - 데이터 파티션 위반 'S' - 올바르지 않은 행 보안 레이블 'L' - DB2 LBAC 쓰기 규칙 위반 'X' - XML 값 인덱스 위반[d]
3	제한조건/컬럼[b] /인덱스 ID[c]의 길이	5바이트	'0'으로 채워 오른쪽 정렬
4	제한조건 이름/컬럼 이름[b]/인덱스 ID[c]	이전 필드의 길이	
5	분리 문자	3바이트	<스페이스><콜론><스페이스>
6	다음 제한조건 위반의 유형	1바이트	'K' - 점검 제한조건 위반 'F' - 외부 키 위반 'G' - 생성된 컬럼 위반 'I' - 고유 인덱스 위반 'D' - 연쇄 삭제 위반 'P' - 데이터 파티션 위반 'S' - 올바르지 않은 행 보안 레이블 'L' - DB2 LBAC 쓰기 규칙 위반 'X' - XML 값 인덱스 위반[d]
7	제한조건/컬럼/인덱스 ID의 길이	5바이트	'0'으로 채워 오른쪽 정렬
8	제한조건 이름/컬럼 이름/인덱스 ID	이전 필드의 길이	
.....	.....	.....	각 위반에 대해 필드 5 - 8 반복

- [a] 접속 조작 다음이 아니라면 SET INTEGRITY문을 사용하여 점검하는 동안 고유 인덱스 위반이 발생하지 않습니다. 그러나 FOR EXCEPTION 옵션을 선택하면 LOAD 실행시 보고됩니다. 한편 LOAD는 예외 테이블의 점검 제한조건, 생성된 컬럼, 외부 키, 삭제 연쇄 또는 데이터 파티션 위반을 보고하지 않습니다.
- [b] 카탈로그 뷰에서 생성된 컬럼의 표현식을 검색하려면 SELECT문을 사용하십시오. 예를 들어, 필드 4가 MYSCHEMA.MYTABLE.GEN_1이면 SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'; 에서 표현식의 처음 50바이트가 "AS(<expression>)" 양식으로 리턴됩니다.
- [c] 카탈로그 뷰에서 인덱스 ID를 검색하려면 SELECT문을 사용하십시오. 예를 들어, 필드 4가 1234인 경우 SELECT INDSCHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234입니다.
- [d] XML 값 인덱스 위반의 경우, 제한 조건 이름, 컬럼 이름 또는 인덱스 ID 필드는 인덱스 중 하나에 무결성 위반을 포함한 XML 컬럼을 식별합니다. 무결성 위반이 있었던 인덱스는 식별하지 않습니다. 이는 인덱스 위반이 발생하는 XML 컬럼의 이름만 식별합니다. 예를 들어, 메시지 컬럼의 값 'X00006XTCOL2'는 XTCOL2 컬럼의 인덱스 중 하나에서 인덱스 위반이 발생했음을 표시합니다.

예외 테이블의 행 조절

예외 테이블의 정보는 다양한 방법으로 처리될 수 있습니다. 데이터를 정정할 수 있고 원본 테이블에 행을 다시 삽입할 수 있습니다.

원래 테이블에 INSERT 트리거가 없는 경우, 예외 테이블에 있는 서브쿼리로 INSERT 문을 발행하여 정정된 행을 전송하십시오.

INSERT 트리거가 있고 트리거를 시작하지 않은 상태에서 예외 테이블의 정정된 행으로 로드 조작을 완료하려는 경우, 다음과 같은 방법이 있습니다.

- 목적에 맞도록 명시적으로 정의된 컬럼의 값에 따라 시작될 INSERT 트리거를 설계하십시오.
- 예외 테이블의 데이터를 언로드하고 로드 유틸리티를 사용하여 데이터를 추가하십시오. 이 경우 데이터를 다시 점검하려면 제한조건 점검이 추가된 행에 국한되지 않도록 유의하십시오.
- 관련 시스템 카탈로그 뷰의 트리거 정의 텍스트를 저장하십시오. 그런 다음, INSERT 트리거를 삭제하고 INSERT를 사용하여 예외 테이블의 정정된 행을 전송합니다. 마지막으로 저장된 트리거 정의를 사용하여 트리거를 재작성하십시오.

예외 테이블의 행을 삽입할 때 트리거를 시작할 수 없도록 금지하는 명시적 규정은 없습니다.

고유 인덱스 위반에 대해 행당 하나의 위반만 보고됩니다.

LONG VARCHAR, LONG VARGRAPHIC 또는 LOB 데이터 유형을 가진 값이 테이블에 있는 경우, 고유 인덱스 위반의 경우 값은 예외 테이블에 삽입되지 않습니다.

예외 테이블 쿼리

예외 테이블의 메시지 컬럼 구조는 이전에 설명한 바와 같이 제한조건 이름, 길이 및 분리문자의 병합된 목록입니다. 이 정보는 쿼리 가능합니다.

예를 들어 제한조건 이름만 있는 각 행을 반복하여 모든 위반 목록을 검색하려면 원본 테이블에 C1 및 C2의 2개 컬럼이 있었다고 가정하십시오. 해당 예외 테이블 E1에도 메시지 컬럼 MSGCOL과 T1의 컬럼에 해당하는 C1 및 C2 컬럼이 있다고 가정하십시오. 다음 쿼리는 재귀를 사용하여 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 나열하려면 이전의 쿼리를 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';
```

다음 쿼리는 모든 점검 제한조건 위반을 확보하는 데 쓰일 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';
```

부록 L. 루틴에 허용되는 SQL문

다음 표는 지정된 SQL 데이터 액세스 표시가 있는 루틴에서 SQL문(첫 번째 컬럼에 지정된)을 실행할 수 있는지 여부를 나타냅니다. 실행 가능한 SQL문이 NO SQL로 정의된 루틴에서 발견되는 경우, SQLSTATE 38001이 리턴됩니다. 다른 실행 컨텍스트의 경우, 컨텍스트에서 지원되지 않는 SQL문은 SQLSTATE 38003을 리턴합니다. CONTAINS SQL 컨텍스트에서 허용되지 않는 다른 SQL문의 경우, SQLSTATE 38004가 리턴됩니다. READS SQL DATA 컨텍스트에서는 SQLSTATE 38002가 리턴됩니다. SQL 루틴 작성시 SQL 데이터 액세스 표시와 일치하지 않는 명령문으로 인해 SQLSTATE 42985가 리턴됩니다.

명령문에서 루틴을 호출할 경우, 명령문에 대한 유효 SQL 데이터 액세스 표시는 다음보다 많아야 합니다.

- 다음 표에서 명령문의 SQL 데이터 액세스 표시
- 루틴 작성시 지정된 루틴의 SQL 데이터 액세스 표시

예를 들어, CALL문에는 CONTAINS SQL의 SQL 데이터 액세스 표시가 있습니다. 그러나 READS SQL DATA로 정의된 스토어드 프로시저가 호출되는 경우, CALL문의 유효 SQL 데이터 액세스 표시는 READS SQL DATA입니다.

루틴에서 SQL문을 호출하면 명령문의 유효 SQL 데이터 액세스 표시는 루틴에 대해 선언된 SQL 데이터 액세스 표시를 초과해서는 안됩니다. 예를 들어, READS SQL DATA로 정의된 함수는 MODIFIES SQL DATA로 정의된 스토어드 프로시저를 호출할 수 없습니다.

표 261. SQL문 및 SQL 데이터 액세스 표시

SQL문	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	Y
AUDIT	N	N	N	Y
BEGIN DECLARE SECTION	Y(1)	Y	Y	Y
CALL	N	Y	Y	Y
CLOSE	N	N	Y	Y
COMMENT ON	N	N	N	Y
COMMIT	N	N(4)	N(4)	N(4)
COMPOUND SQL	N	Y	Y	Y
CONNECT(2)	N	N	N	N
CREATE...	N	N	N	Y
DECLARE CURSOR	Y(1)	Y	Y	Y

표 261. SQL문 및 SQL 데이터 액세스 표시 (계속)

SQL문	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	Y
DELETE	N	N	N	Y
DESCRIBE	N	Y	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
MERGE	N	N	N	Y
OPEN	N	N	Y(5)	Y
PREPARE	N	Y	Y	Y
REFRESH TABLE	N	N	N	Y
RELEASE CONNECTION(2)	N	N	N	N
RELEASE SAVEPOINT	N	N	N	Y
RENAME TABLE	N	N	N	Y
REVOKE ...	N	N	N	Y
ROLLBACK	N	N(4)	N(4)	N(4)
ROLLBACK TO SAVEPOINT	N	N	N	Y
SAVEPOINT	N	N	N	Y
SELECT INTO	N	N	Y(5)	Y
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	Y
SET 특수 레지스터	N	Y	Y	Y
SET 변수	N	Y(6)	Y(5)	Y
TRANSFER OWNERSHIP	N	N	N	Y
TRUNCATE	N	N	N	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 없음이 내포되어 있지만 실행할 수 없는 명령문은 제한되지 않습니다.
2. 연결 관리 명령문은 루틴 실행 컨텍스트에서 허용되지 않습니다.
3. 이는 실행되는 명령문에 따라 다릅니다. EXECUTE문에 지정된 명령문은 적용되는 특정 SQL 액세스 레벨의 컨텍스트에서 허용되는 명령문이어야 합니다. 예를 들어, 적용되는 SQL 액세스 레벨이 READS SQL DATA일 경우, 명령문으로 INSERT, UPDATE 또는 DELETE를 수행할 수 없습니다.
4. TO SAVEPOINT절이 없는 COMMIT문 및 ROLLBACK문은 스토어드 프로시저에 사용할 수 있지만 스토어드 프로시저가 응용프로그램에서 직접 호출되거나 응용프로그램에서 중첩된 스토어드 프로시저 호출을 통해 간접적으로 호출되는 경우에만 사용할 수 있습니다. (트리거, 함수, 메소드 또는 ATOMIC 복합 명령문이 스토어드 프로시저에 대한 호출 체인에 있는 경우, 작업 단위(UOW)의 COMMIT 또는 ROLLBACK이 허용되지 않습니다.)
5. SQL 액세스 등급인 READS SQL DATA가 적용되는 경우, SELECT INTO문, OPEN문이 참조하는 커서 또는 SET 변수 명령문의 오른쪽 표현식에 SQL 데이터 변경 명령문을 임베드할 수 없습니다.
6. SQL 액세스 등급인 CONTAINS SQL이 적용되는 경우, SET 변수 명령문의 오른쪽 표현식에 스칼라 fullselect를 임베드할 수 없습니다.

부록 M. 컴파일된 명령문에서 호출되는 CALL

데이터베이스 위치에서 저장된 프로시저를 호출합니다. 예를 들어 프로시저는 데이터베이스 위치에서 실행하고 데이터를 클라이언트 응용프로그램으로 리턴합니다.

SQL CALL문을 사용하는 프로그램은 클라이언트에서 하나와 서버에서 다른 하나의 두 파트로 실행하도록 설계되었습니다. 데이터베이스의 서버 프로시저는 클라이언트 응용프로그램과 동일한 트랜잭션에서 실행됩니다. 클라이언트 응용프로그램 및 프로시저가 동일한 데이터베이스 파티션에 있는 경우 스토어드 프로시저는 로컬로 실행됩니다.

주: CALL문의 이 양식은 사용되지 않으며, 이전 버전의 DB2와의 호환성을 위해서만 제공되고 있습니다.

호출

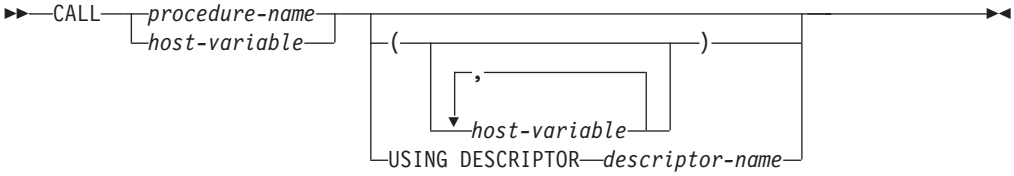
이 CALL문의 양식은 CALL_RESOLUTION DEFERRED 옵션으로 프리컴파일된 응용프로그램에만 임베드될 수 있습니다. 페더레이티드 프로시저를 호출할 수 없습니다. 트리거, SQL 프로시저 또는 다른 모든 응용프로그램이 아닌 컨텍스트에서 사용할 수 없습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 그러나 호스트 변수를 통해 프로시저 이름을 지정할 수 있으며, USING DESCRIPTOR절의 사용과 결합된 이 방식으로 런타임 시에 프로시저 이름과 매개변수 목록을 둘 다 제공할 수 있으므로 동적으로 준비된 명령문과 비슷한 효과를 달성합니다.

권한 부여

런타임 시 명령문의 권한 부여 ID에서 보유하는 특권은 다음 중 하나를 포함해야 합니다.

- 프로시저와 연관된 패키지에 관한 EXECUTE 특권. 프로시저에 관한 EXECUTE 특권은 점검되지 않음
- 프로시저와 연관된 패키지에 대한 CONTROL 특권
- SYSADM 또는 DBADM 권한

구문



설명

procedure-name 또는 *host-variable*

호출할 프로시저를 식별합니다. 프로시저 이름은 직접 또는 호스트 변수에서 지정할 수 있습니다. 식별된 프로시저는 현재 서버에 존재해야 합니다(SQLSTATE 42724).

*procedure-name*이 지정되면 254바이트보다 크지 않은 일반 ID여야 합니다. 일반 ID만 가능하기 때문에 공백이나 특수 문자를 포함할 수 없습니다. 값은 대문자로 변환됩니다. 소문자, 공백 또는 특수 문자를 사용해야 하는 경우 이름은 *host-variable*을 통해 지정해야 합니다.

*host-variable*이 지정되면, 길이가 254자 이하인 CHAR 또는 VARCHAR 변수이어야 하며 표시기 변수를 포함해서는 안됩니다. 값은 대문자로 변환되지 *않습니다*. 문자열은 왼쪽으로 정렬되어야 합니다.

프로시저 이름은 다음의 여러 가지 양식 중 하나일 수 있습니다.

procedure-name

실행할 프로시저의 이름(확장자 없음). 호출되는 프로시저는 다음과 같이 판별됩니다.

1. *procedure-name*을 사용하여 정의된 프로시저(SYSCAT.ROUTINES의)에서 일치하는 프로시저를 검색합니다. 일치하는 프로시저는 뒤에 오는 단계를 사용하여 판별됩니다.
   a. 카탈로그(SYSCAT.ROUTINES)에서 프로시저(ROUTINETYPE이 'P')를 찾으십시오. ROUTINENAME은 지정된 *procedure-name*과 일치하고, ROUTINESCHEMA는 SQL 경로(CURRENT PATH 특수 레지스터)의 스키마 이름입니다. 스키마 이름이 명시적으로 지정되면 SQL 경로가 무시되며, 지정된 스키마 이름을 갖는 프로시저만 고려됩니다.
   b. 그런 다음, CALL문에 지정된 인수의 수와 매개변수 수가 같지 않은 프로시저를 제거하십시오.
   c. SQL 경로에서 가장 빠른 남아 있는 프로시저를 선택하십시오.

   프로시저가 선택되면 DB2가 외부 이름으로 정의되는 프로시저를 호출합니다.
2. 일치하는 프로시저가 없는 경우 *procedure-name*이 프로시저 라이브러리의 이름 및 해당 라이브러리에 있는 함수 이름 모두로 사용됩니다. 예를 들어 *procedure-name*이 `proclib`인 경우 DB2 서버는 `proclib` 프로시저 라이브러리를 로드하고 해당 라이브러리의 함수 루틴 `proclib()`를 실행합니다.

   UNIX 시스템에서는 프로시저 라이브러리에 대한 디폴트 디렉토리는 `sqllib/function`입니다. 분리되지 않은 프로시저의 디폴트 디렉토리는 `sqllib/function/unfenced`입니다.

Windows 기반 시스템에서 프로시저 라이브러리의 디폴트 디렉토리는 sqllib₩function입니다. 분리되지 않은 프로시저의 디폴트 디렉토리는 sqllib₩function₩unfenced입니다.

라이브러리나 함수를 찾을 수 없는 경우 오류가 리턴됩니다(SQLSTATE 42884).

procedure-library!function-name

느낌표(!)는 라이브러리 이름과 프로시저의 함수 이름 사이의 분리문자로 사용됩니다. 예를 들어 proclib!func가 지정되면 proclib가 메모리에 로드되고 해당 라이브러리의 func 함수가 실행됩니다. 따라서 다중 함수가 동일한 프로시저 라이브러리에 위치할 수 있습니다.

프로시저 라이브러리는 해당 디렉토리 또는 *procedure-name*에서 설명하는 LIBPATH 변수에서 지정됩니다.

absolute-path!function-name

*absolute-path*는 스토어드 프로시저 라이브러리의 전체 경로를 지정합니다.

UNIX 시스템에서는 예를 들어 /u/terry/proclib!func가 지정되는 경우 프로시저 라이브러리 proclib가 /u/terry 디렉토리에서 오고 해당 라이브러리의 func 함수가 실행됩니다.

모든 경우에 내재된 또는 명시적 전체 경로를 포함한 프로시저 이름의 전체 길이는 254바이트 이하여야 합니다.

(host-variable,...)

*host-variable*의 각 스펙은 CALL문의 매개변수입니다. CALL의 *n*번째 매개변수는 서버 프로시저의 *n*번째 매개변수에 해당합니다.

각 *host-variable*이 클라이언트와 서버 사이의 양방향 모두에서 데이터 교환에 사용된다고 가정합니다. 클라이언트와 서버 사이의 불필요한 데이터 전송을 피하려면 클라이언트 응용프로그램이 각 매개변수와 함께 표시기 변수를 제공하고 매개변수가 프로시저로 데이터를 전송하는 데 사용되지 않는 경우 표시기를 -1로 설정해야 합니다. 프로시저는 클라이언트 응용프로그램으로 데이터를 리턴하는 데 사용되지 않는 모든 매개변수에 대해 표시기 변수를 -128로 설정해야 합니다.

서버가 DB2 9.1 데이터베이스 서버인 경우 매개변수는 클라이언트 및 서버 프로그램 모두에서 데이터 유형이 일치해야 합니다.

USING DESCRIPTOR *descriptor-name*

호스트 변수의 유효한 설명이 포함되어 있는 SQLDA를 식별합니다. *n*번째 매개변수는 서버 프로시저의 *n*번째 매개변수에 해당합니다.

CALL문이 처리되기 전에 응용프로그램이 다음 필드를 SQLDA에 설정해야 합니다.

- SQLDA에 제공되는 SQLVAR 어커런스 수를 나타내기 위한 SQLN
- SQLDA에 대해 할당되는 스토리지의 바이트 수를 나타내기 위한 SQLDABC

- 명령문 처리시 SQLDA에서 사용되는 변수의 수를 나타내기 위한 SQLDA
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수. 전달되는 각 기본 SQLVAR 요소의 다음 필드가 초기화되어야 합니다.
  - SQLTYPE
  - SQLLEN
  - SQLDATA
  - SQLIND

  전달되는 각 보조 SQLVAR 요소의 다음 필드가 초기화되어야 합니다.
  - LEN.SQLLONGLEN
  - SQLDATALEN
  - SQLDATATYPE_NAME

SQLDA가 클라이언트와 서버 사이의 양방향에서 데이터 교환에 사용된다고 가정합니다. 클라이언트와 서버 사이의 불필요한 데이터 전송을 피하려면 매개변수가 프로시저로 데이터를 전송하는 데 사용되지 않는 경우 클라이언트 응용프로그램이 SQLIND 필드를 -1로 설정해야 합니다. 프로시저는 클라이언트 응용프로그램으로 데이터를 리턴하는 데 사용되지 않는 모든 매개변수에 대해 SQLIND 필드를 -128로 설정해야 합니다.

주

- *대형 오브젝트(LOB) 데이터 유형 사용*:

  클라이언트 및 서버 응용프로그램이 SQLDA의 LOB 데이터를 지정해야 하는 경우 SQLVAR 항목의 배수를 할당하십시오.

  LOB 데이터 유형은 DB2 버전 2 이후 프로시저에서 지원되었습니다. LOB 데이터 유형은 모든 아래 레벨 클라이언트 또는 서버에서 지원되지 않습니다.

- *SQL 프로시저에서 DB2_RETURN_STATUS 검색*:

  SQL 프로시저가 상태 값과 함께 RETURN문을 성공적으로 발행할 경우, 이 값은 SQLCA의 첫 번째 SQLERRD 필드에서 리턴됩니다. CALL문이 SQL 프로시저에서 실행되면, GET DIAGNOSTICS문을 사용하여 DB2_RETURN_STATUS 값을 검색하십시오. SQLSTATE가 오류를 나타낼 경우, 값은 -1입니다.

- *프로시저에서 결과 세트 리턴*:

  클라이언트 응용프로그램이 CLI를 사용하여 작성된 경우, 결과 세트는 직접 클라이언트 응용프로그램으로 리턴될 수 있습니다. 프로시저는 해당 결과 세트에서 커서를 선언하고 결과 세트에서 커서를 연 다음 프로시저를 종료할 때 커서를 열린 상태로 유지함으로써 결과 세트가 리턴될 것임을 나타냅니다.

프로시저 끝에서:

- 열린 상태인 모든 커서의 경우 결과 세트가 응용프로그램으로 리턴됩니다.
- 둘 이상의 커서가 열려 있을 경우 결과 세트는 커서가 열린 순서대로 처리됩니다.
- 읽어들이지 않은 행만 다시 전달됩니다. 예를 들어, 커서의 결과 세트에 500개의 행이 있는데 이 중에서 150개의 행이 프로시저가 종료될 때 프로시저에 의해 읽힌 경우, 151부터 500까지의 행이 프로시저로 리턴됩니다.

- *특수 레지스터 조절*:

  호출자에 대한 특수 레지스터의 설정은 호출 시 프로시저에서 상속되며, 호출자로의 리턴 시에 리스토어됩니다. 특수 레지스터는 프로시저 내에서 변경될 수 있지만 이러한 변경이 호출자에게 영향을 주지 않습니다. Legacy 프로시저(매개변수 스타일 DB2DARI로 정의되거나 디폴트 라이브러리에서 발견되는 프로시저)의 경우에는 해당되지 않으며, 이 경우에는 프로시저에서 특수 레지스터에 작성된 변경사항이 호출자에 대한 설정이 됩니다.

- *호환성*

  응용프로그램에서 임베드될 수 있거나(CALL_RESOLUTION IMMEDIATE 옵션으로 응용프로그램을 프리컴파일하여), 동적으로 준비될 수 있는 보다 새롭고, 선호하는 CALL문의 양식이 있습니다.

예:

예 1:

C에서, ACHIEVE 라이브러리의 TEAMWINS 프로시저를 호출하고 호스트 변수 HV_ARGUMENT에 저장된 매개변수를 전달하십시오.

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

예 2:

C에서, INOUT_SQLDA라는 SQLDA를 사용하여 :SALARY_PROC 프로시저를 호출하십시오.

```
struct sqlda *INOUT_SQLDA;
/* Setup code for SQLDA variables goes here */
CALL :SALARY_PROC
USING DESCRIPTOR :*INOUT_SQLDA;
```

예 3:

Java 프로시저가 다음 명령문을 사용하여 데이터베이스에 정의됩니다.

```
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 조각을 사용하여 이 프로시저를 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) ;
...
```

CALL문에 지정된 프로시저 이름이 데이터베이스에 있고 외부 이름 'parts!onhand'를 사용하므로, 이 응용프로그램 코드 조각은 *parts* 클래스에서 Java 메소드 *onhand*를 호출합니다.

부록 N. DB2 기술 정보 개요

DB2 기술 정보는 다음 도구 및 메소드를 통해 사용할 수 있습니다.

- DB2 정보 센터
  - 주제 항목(태스크, 개념 및 참조 항목)
  - DB2 도구에 대한 도움말
  - 샘플 프로그램
  - 자습서
- DB2 서적
  - PDF 파일(다운로드)
  - PDF 파일(DB2 PDF DVD)
  - 인쇄된 서적
- 명령행 도움말
  - 명령 도움말
  - 메시지 도움말

주: DB2 정보 센터의 주제는 PDF 또는 하드카피 서적보다 더 자주 갱신됩니다. 최신 정보를 보려면 사용 가능한 문서 갱신사항을 설치하거나 ibm.com에서 DB2 정보 센터를 참조하십시오.

ibm.com에서 추가 DB2 기술 정보(예: 기술 노트, 백서 및 IBM Redbooks® 서적)를 온라인으로 액세스할 수 있습니다. 다음은 DB2 정보 관리 라이브러리 소프트웨어 사이트의 주소입니다. http://www.ibm.com/software/data/sw-library/

문서 피드백

DB2 문서에 대한 피드백을 환영합니다. DB2 문서를 향상시키는 방법에 대해서 제안사항이 있는 경우 db2docs@ca.ibm.com으로 전자 우편을 보내십시오. DB2 문서 팀에서는 고객의 모든 피드백을 읽지만 직접 응답할 수는 없습니다. 고객의 문제를 더 잘 이해할 수 있도록 가능한 한 구체적인 예를 제공해주십시오. 특정 주제 또는 도움말 파일에 대한 피드백을 보내실 경우, 제목 및 URL을 알려주십시오.

DB2 고객 지원에 문의할 때는 이 전자 우편 주소를 사용하지 마십시오. 문서에서 해결할 수 없는 DB2 기술 문제점이 있는 경우, 해당 지역의 IBM 서비스 센터에 도움을 요청하십시오.

DB2 기술 라이브러리(하드카피 또는 PDF 형식)

다음 표는 IBM Publications Center(www.ibm.com/shop/publications/order)에서 사용할 수 있는 DB2 라이브러리에 대한 설명입니다. PDF 형식의 영문 DB2 버전 9.7 매뉴얼 및 번역된 버전은 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947에서 다운로드할 수 있습니다.

표에 인쇄할 수 있는 책으로 설명된 경우라도, 사용 국가 또는 지역에 따라 해당 책을 사용할 수 없을 수도 있습니다.

매뉴얼이 갱신될 때마다 문서 번호가 증가합니다. 다음 사항을 참조하여 읽고 있는 매뉴얼이 최신 버전인지 확인하십시오.

주: *DB2 정보 센터*는 PDF 또는 하드카피 서적보다 자주 갱신됩니다.

표 262. DB2 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
관리 API 참조서	SA30-3958-00	예	2009년 8월
관리 루틴 및 뷰	SA30-3955-00	아니오	2009년 8월
Call Level Interface Guide and Reference, Volume 1	SC27-2437-00	예	2009년 8월
Call Level Interface Guide and Reference, Volume 2	SC27-2438-00	예	2009년 8월
명령어 참조서	SA30-3959-00	예	2009년 8월
데이터 이동 유틸리티 안내서 및 참조서	SA30-3969-00	예	2009년 8월
데이터 복구 및 고가용성 안내서 및 참조서	SA30-3970-00	예	2009년 8월
데이터베이스 관리 개념 및 구성 참조서	SA30-3951-00	예	2009년 8월
데이터베이스 모니터링 안내서 및 참조서	SA30-3953-00	예	2009년 8월
데이터베이스 보안 안내서	SA30-3971-00	예	2009년 8월
DB2 Text Search Guide	SC27-2459-00	예	2009년 8월
Developing ADO.NET and OLE DB Applications	SC27-2444-00	예	2009년 8월
Developing Embedded SQL Applications	SC27-2445-00	예	2009년 8월
Developing Java Applications	SC27-2446-00	예	2009년 8월
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-2447-00	아니오	2009년 8월

표 262. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Developing User-defined Routines(SQL and External)	SC27-2448-00	예	2009년 8월
Getting Started with Database Application Development	GI11-9410-00	예	2009년 8월
Linux 및 Windows에서 DB2 설치 및 관리 시작하기	GA30-3960-00	예	2009년 8월
자국어 안내서	SA30-3972-00	예	2009년 8월
DB2 Server 설치	GA30-3962-00	예	2009년 8월
IBM Data Server Client 설치	GA30-3963-00	아니오	2009년 8월
Message Reference Volume 1	SC27-2450-00	아니오	2009년 8월
Message Reference Volume 2	SC27-2451-00	아니오	2009년 8월
Net Search Extender Administration and User's Guide	SC27-2469-00	아니오	2009년 8월
파티셔닝 및 클러스터링 안내서	SA30-3973-00	예	2009년 8월
pureXML Guide	SC27-2465-00	예	2009년 8월
Query Patroller 관리 및 사용자 안내서	SA30-3974-00	아니오	2009년 8월
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC27-2468-00	아니오	2009년 8월
SQL Procedural Languages: Application Enablement and Support	SC27-2470-00	예	2009년 8월
SQL 참조서, 볼륨 1	SA30-3956-00	예	2009년 8월
SQL 참조서, 볼륨 2	SA30-3957-00	예	2009년 8월
문제점 해결 및 데이터베이스 성능 조정	SA30-3952-00	예	2009년 8월
DB2 버전 9.7로 업그레이드	SA30-3961-00	예	2009년 8월
Visual Explain 자습서	SA30-3968-00	아니오	2009년 8월
DB2 버전 9.7의 새로운 내용	SA30-3967-00	예	2009년 8월
Workload Manager Guide and Reference	SC27-2464-00	예	2009년 8월

표 262. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
XQuery Reference	SC27-2466-00	아니오	2009년 8월

표 263. DB2 Connect 특정 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
DB2 Connect Personal Edition 설치 및 구성	SA30-3965-00	예	2009년 8월
DB2 Connect Server 설치 및 구성	SA30-3966-00	예	2009년 8월
DB2 Connect 사용자 안내서	SA30-3964-00	예	2009년 8월

표 264. Information Integration 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	예	2009년 8월
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	예	2009년 8월
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	아니오	2009년 8월
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	예	2009년 8월
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	예	2009년 8월

인쇄된 DB2 서적 주문

인쇄된 DB2 서적이 필요한 경우, 대부분 온라인으로 구매할 수 있으나 모든 국가 또는 지역에서 가능한 것은 아닙니다. 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. *DB2 PDF 문서* DVD의 일부 소프트카피 서적은 인쇄할 수 없다는 점에 유의하십시오. 예를 들어, *DB2 메시지 참조서*의 볼륨은 인쇄된 서적으로 사용할 수 없습니다.

DB2 PDF 문서 DVD에서 사용할 수 있는 다수의 DB2 서적의 인쇄된 버전은 IBM에서 유료로 주문할 수 있습니다. 주문하는 위치에 따라 IBM Publications Center에서 온라인으로 서적을 주문할 수도 있습니다. 해당 국가 또는 지역에서 온라인 주문이

불가능하면, 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. DB2 PDF 문서 DVD의 모든 서적을 인쇄할 수는 없다는 점에 유의하십시오.

주: 가장 최신의 완전한 DB2 문서는 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7의 DB2 정보 센터에서 유지보수됩니다.

인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.

- 해당 국가 또는 지역에서 인쇄된 DB2 서적을 온라인으로 주문할 수 있는지 여부를 확인하려면 http://www.ibm.com/shop/publications/order의 IBM Publications Center를 확인하십시오. 서적 주문 정보를 액세스하려면 국가/지역/언어를 선택한 다음 해당 위치에서 주문 지시사항을 따르십시오.
- 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.
  1. 다음 웹 사이트 중 하나에서 해당 지역 담당자에 대한 문의처 정보를 찾으십시오.
     - www.ibm.com/planetwide에 있는 IBM 전세계 문의처 디렉토리
     - http://www.ibm.com/shop/publications/order의 IBM Publications 웹 사이트. 사용 지역의 해당 서적 홈 페이지에 액세스하려면 해당 국가, 지역 또는 언어를 선택해야 합니다. 이 페이지에서 "이 제품의 정보" 링크를 수행하십시오.
  2. 전화로 주문할 경우, 주문할 DB2 서적을 지정하십시오.
  3. 담당자에게 주문하려는 서적의 제목 및 문서 번호를 제공하십시오. 서적의 제목 및 문서 번호는 1180 페이지의 『DB2 기술 라이브러리(하드카피 또는 PDF 형식)』를 참조하십시오.

명령행 처리기에서 SQL 상태 도움말 표시

DB2 제품은 SQL문의 결과로 나타나는 상태에 대한 SQLSTATE 값을 리턴합니다. SQLSTATE 도움말은 SQL 상태 및 SQL 상태 클래스 코드의 의미를 설명합니다.

SQL 상태 도움말을 시작하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? sqlstate or ? class code
```

여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자를 나타냅니다.
예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

DB2 정보 센터의 다른 버전에 액세스

DB2 버전 9.7 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/입니다.

DB2 버전 9.5 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/입니다.

DB2 버전 9 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9/입니다.

DB2 버전 8 주제에 대한 버전 8 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v8/입니다.

DB2 정보 센터에서 원하는 언어로 항목 표시

DB2 정보 센터는 브라우저 환경 설정에 지정된 언어로 주제 항목을 표시합니다. 주제가 원하는 언어로 변환되지 않은 경우, DB2 정보 센터는 해당 주제 항목을 영어로 표시합니다.

- Internet Explorer 브라우저에서 원하는 언어로 항목을 표시하려면 다음을 수행하십시오.
  1. Internet Explorer에서 **도구 —> 인터넷 옵션 —> 언어** 단추를 누르십시오. 언어 환경 설정 창이 열립니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 누르십시오.

       주: 언어를 추가하더라도 원하는 언어로 항목을 표시하는 데 필요한 글꼴이 컴퓨터에 설치되지는 않습니다.
     - 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 해당 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
  3. 브라우저 캐시를 지운 후 페이지를 새로 고치면 원하는 언어로 DB2 정보 센터가 표시됩니다.
- Firefox 또는 Mozilla 브라우저에서 원하는 언어로 주제 항목을 표시하려면 다음을 수행하십시오.
  1. **도구 —> 설정 —> 내용** 대화 상자의 **언어** 섹션에서 단추를 선택하십시오. 환경 설정 창에 언어 패널이 표시됩니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 언어 선택 창에서 원하는 언어를 선택한 다음 **추가...** 단추를 누르십시오.

- 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 해당 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.

3. 브라우저 캐시를 지운 후 페이지를 새로 고치면 원하는 언어로 DB2 정보 센터가 표시됩니다.

일부 브라우저 및 운영 체제 조합에서는 운영 체제의 국가별 설정을 선택한 로케일 및 언어로 변경해야 합니다.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신

로컬로 설치된 DB2 정보 센터는 주기적으로 갱신해야 합니다.

시작하기 전에

DB2 버전 9.7 정보 센터는 미리 설치된 상태여야 합니다. 자세한 내용은 *DB2 Server 설치*의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치』 주제를 참조하십시오. 정보 센터 설치에 적용되는 모든 전제조건 및 제한사항은 정보 센터 갱신에도 적용됩니다.

이 태스크에 대한 정보

기존의 DB2 정보 센터는 자동 또는 수동으로 갱신할 수 있습니다.

- 자동 갱신 - 기존 정보 센터 기능 및 언어를 갱신합니다. 자동 갱신의 또 다른 이점으로는 갱신 동안 정보 센터를 사용할 수 없는 시간이 매우 짧다는 점입니다. 또한 자동 갱신은 주기적으로 실행되는 기타 일괄처리 작업의 일부로 실행되도록 설정할 수도 있습니다.
- 수동 갱신 - 갱신 프로세스 중에 기능이나 언어를 추가하려는 경우 사용하십시오. 예를 들어, 로컬 정보 센터는 기본적으로 영어와 프랑스어로 설치되어 있으며, 수동 갱신을 통해 기존 정보 센터의 기능 및 언어 갱신뿐만 아니라 독일어도 설치할 수 있습니다. 단, 수동 갱신을 수행하려면 정보 센터를 중지한 다음 갱신하고 재시작해야 합니다. 정보 센터는 갱신 프로세스 동안에는 사용할 수 없습니다.

프로시저

이 주제는 자동 갱신 프로세스에 대한 설명입니다. 수동 갱신에 대한 지시사항은 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신』 주제를 참조하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 자동으로 갱신하려면 다음을 수행하십시오.

1. Linux 운영 체제의 경우
   a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.

b. 설치 디렉토리에서 doc/bin 디렉토리로 이동하십시오.

c. 다음과 같이 ic-update 스크립트를 실행하십시오.

```
ic-update
```

2. Windows 운영 체제의 경우

a. 명령 창을 여십시오.

b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.

c. 설치 디렉토리에서 doc₩bin 디렉토리로 이동하십시오.

d. 다음과 같이 ic-update.bat 파일을 실행하십시오.

```
ic-update.bat
```

결과

DB2 정보 센터가 자동으로 재시작됩니다. 갱신사항이 사용 가능한 경우, 정보 센터에는 새로 갱신된 주제가 표시됩니다. 정보 센터 갱신을 사용할 수 없는 경우, 메시지가 로그에 추가됩니다. 로그 파일은 doc₩eclipse₩configuration 디렉토리에 있습니다. 이 로그 파일 이름은 임의로 생성된 번호입니다. 예: 1239053440785.log

---

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신

DB2 정보 센터를 로컬로 설치한 경우, IBM으로부터 문서 갱신사항을 받아 설치할 수 있습니다.

로컬로 설치된 DB2 정보 센터를 수동으로 갱신하려면 다음을 수행하십시오.

1. 컴퓨터에서 DB2 정보 센터를 중지한 후 독립형 모드에서 다시 시작하십시오. 독립형 모드에서 정보 센터를 실행하면 사용자의 네트워크와 연결된 다른 사용자는 정보 센터에 액세스할 수 없으므로 갱신사항을 적용할 수 있습니다. DB2 정보 센터의 워크스테이션 버전은 항상 독립형 모드에서 실행됩니다.
2. 사용 가능한 갱신사항을 확인하려면 갱신 기능을 사용하십시오. 설치해야 할 갱신사항이 있는 경우, 갱신 기능을 사용하여 이를 가져온 후 설치할 수 있습니다.

   주: 인터넷에 연결되지 않은 머신에 DB2 정보 센터 갱신사항을 설치해야 할 경우, 인터넷에 연결되고 DB2 정보 센터가 설치된 머신을 사용하여 갱신 사이트를 로컬 파일 시스템으로 미러하십시오. 네트워크 상에 문서 갱신사항을 설치하려는 사용자가 많을 경우에는 갱신 사이트를 로컬로 미러링하거나 갱신 사이트의 프록시를 작성하여 갱신을 수행하면 각 개인에게 필요한 시간을 줄일 수 있습니다.
   갱신 패키지가 사용 가능하면 갱신 기능을 사용하여 패키지를 가져오십시오. 그러나 갱신 기능은 독립형 모드에서만 사용할 수 있습니다.

3. 독립형 정보 센터를 중지한 후 컴퓨터에서 DB2 정보 센터를 재시작하십시오.

주: Windows 2008, Windows Vista 이상의 경우 이 절 다음에 나오는 명령은 관리자로 실행해야 합니다. 전체 관리자 권한으로 명령 프롬프트 또는 그래픽 도구를 열려면 단축 아이콘을 마우스 오른쪽 단추로 누른 후 **관리자로 실행**을 선택하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 갱신하려면 다음을 수행하십시오.

1. DB2 정보 센터를 중지하십시오.
   - Windows의 경우, **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 Information Center** 서비스를 마우스 오른쪽 단추로 누른 후 **중지**를 선택하십시오.
   - Linux의 경우, 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 독립형 모드에서 정보 센터를 시작하십시오.
   - Windows의 경우:
     a. 명령 창을 여십시오.
     b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.
     c. 설치 디렉토리에서 doc₩bin 디렉토리로 이동하십시오.
     d. 다음과 같이 help_start.bat 파일을 실행하십시오.

        ```
        help_start.bat
        ```
   - Linux의 경우:
     a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 /opt/ibm/db2ic/V9.7 디렉토리에 디폴트로 설치됩니다.
     b. 설치 디렉토리에서 doc/bin 디렉토리로 이동하십시오.
     c. 다음과 같이 help_start 스크립트를 실행하십시오.

        ```
        help_start
        ```

   시스템의 기본 웹 브라우저가 열리고 독립형 정보 센터가 표시됩니다.
3. **갱신** 단추( )를 누르십시오. (JavaScript™가 브라우저에서 사용 가능해야 합니다.) 정보 센터의 오른쪽 패널에서 **갱신사항 찾기**를 누르십시오. 기존 문서의 갱신사항 목록이 표시됩니다.
4. 설치 프로세스를 시작하려면 설치할 선택란을 체크한 후 **갱신사항 설치**를 누르십시오.
5. 설치 프로세스가 완료되면 **완료**를 누르십시오.

6. 독립형 정보 센터를 중지하십시오.
   - Windows의 경우, 설치 디렉토리의 doc¥bin 디렉토리로 이동한 후 다음과 같이 help_end.bat 파일을 실행하십시오.

     ```
     help_end.bat
     ```

     주: help_end 일괄처리 파일에는 help_start 일괄처리 파일로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start.bat를 중지할 때 Ctrl+C 또는 다른 메소드를 사용하지 마십시오.
   - Linux의 경우, 설치 디렉토리의 doc/bin 디렉토리로 이동한 후 다음과 같이 help_end 스크립트를 실행하십시오.

     ```
     help_end
     ```

     주: help_end 스크립트에는 help_start 스크립트로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start 스크립트를 중지할 때 다른 메소드를 사용하지 마십시오.
7. DB2 정보 센터를 재시작하십시오.
   - Windows의 경우, **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 Information Center** 서비스를 마우스 오른쪽 단추로 누른 후 **시작**을 선택하십시오.
   - Linux의 경우, 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 start
     ```

갱신된 DB2 정보 센터에는 새로 갱신된 주제가 표시됩니다.

DB2 자습서

DB2 자습서는 DB2 제품의 여러가지 측면을 학습하는 데 유용합니다. 각 레슨은 단계별 지시사항을 제공합니다.

시작하기 전에

정보 센터(http://publib.boulder.ibm.com/infocenter/db2help/)에서 XHTML 버전의 자습서를 볼 수 있습니다.

일부 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크에 필요한 전제조건 설명은 자습서를 참조하십시오.

DB2 자습서

자습서를 보려면 제목을 누르십시오.

『**pureXML®**』(*pureXML Guide*)

DB2 데이터베이스를 설정하여 XML 데이터를 저장하고 원시 XML 데이터 스토어로 기본 조작을 수행할 수 있습니다.

*Visual Explain 자습서*의 『**Visual Explain**』

더 나은 성능을 위해 Visual Explain을 사용하여 SQL문을 분석, 최적화 및 조정할 수 있습니다.

DB2 문제점 해결 정보

DB2 데이터베이스 제품 사용 시 발생하는 광범위한 문제점을 판별하고 해결하는 데 도움이 되는 정보를 사용할 수 있습니다.

DB2 문서

문제점 해결 정보는 *DB2 문제점 해결 안내서* 또는 *DB2 정보 센터*의 데이터베이스 기본 절을 참조하십시오. DB2 진단 도구 및 유틸리티를 사용하여 문제점을 찾아내고 식별하는 방법, 가장 일반적인 문제점에 대한 솔루션 및 DB2 데이터베이스 제품에서 발생할 수 있는 문제점을 해결하는 방법 등에 관한 정보가 있습니다.

DB2 기술 지원 웹 사이트

문제점이 발생한 경우 해당 원인 및 솔루션을 찾으려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 서적, 기술 노트, APAR(Authorized Program Analysis Report 또는 버그 수정), FixPack 및 기타 자원에 대한 링크가 있습니다. 이러한 기술 자료를 검색하여 문제에 대해 사용 가능한 솔루션을 찾을 수 있습니다.

다음은 DB2 기술 지원 웹 사이트의 주소입니다. http://www.ibm.com/software/data/db2/support/db2_9/

이용약관

다음 조건에 따라 이 책을 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 이 책을 개인적, 비상업적 용도로 복제할 수 있습니다. IBM의 명시적인 동의 없이는 이 책 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 이 책을 귀하 기업 집단 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 이 책의 2차적 저작물을 만들거나 이 책 또는 그 일부를 복제, 배포 또는 전시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 이 책이나 이 책에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대한 어떠한 허가나 라이센스 또는 권한도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 이 책의 사용이 IBM의 이익을 해친다고 판단되거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 이 사이트에서 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 경우에만 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 이 책의 내용에 대해 어떠한 보증도 제공하지 않습니다. 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 현 상태대로 제공합니다.

부록 O. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. 비IBM 제품에 대한 정보는 이 책을 처음 발행할 때의 정보에 기초하고 있으며 변경될 수 있습니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트 문자 세트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠. 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등) 하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 다른 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 되는 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 샘플 프로그램은 어떠한 보증없이 ″있는 그대로″ 제공됩니다. IBM은 샘플 프로그램의 사용으로 인해 발생하는 모든 손해에 대해 책임을 지지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. *_enter* 연도_. All rights reserved.

상표

IBM, IBM 로고 및 ibm.com®은 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 기타 회사의 상표입니다. 현재 IBM 상표 목록은 웹 “저작권 및 상표 정보”(www.ibm.com/legal/kr/copytrade.shtml)에 있습니다.

다음 용어는 기타 회사의 상표 또는 등록상표입니다.

- Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.
- Java 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.
- UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.
- Intel®, Intel 로고, Intel Inside®, Intel Inside 로고, Intel® Centrino®, Intel Centrino 로고, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium 및 Pentium®은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표 또는 등록상표입니다.
- Microsoft, Windows, Windows NT® 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

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

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

SA30-3956-00

Spine information:

Linux, UNIX 및 Windows용 IBM DB2 9.7

SQL 참조서, 볼륨 1