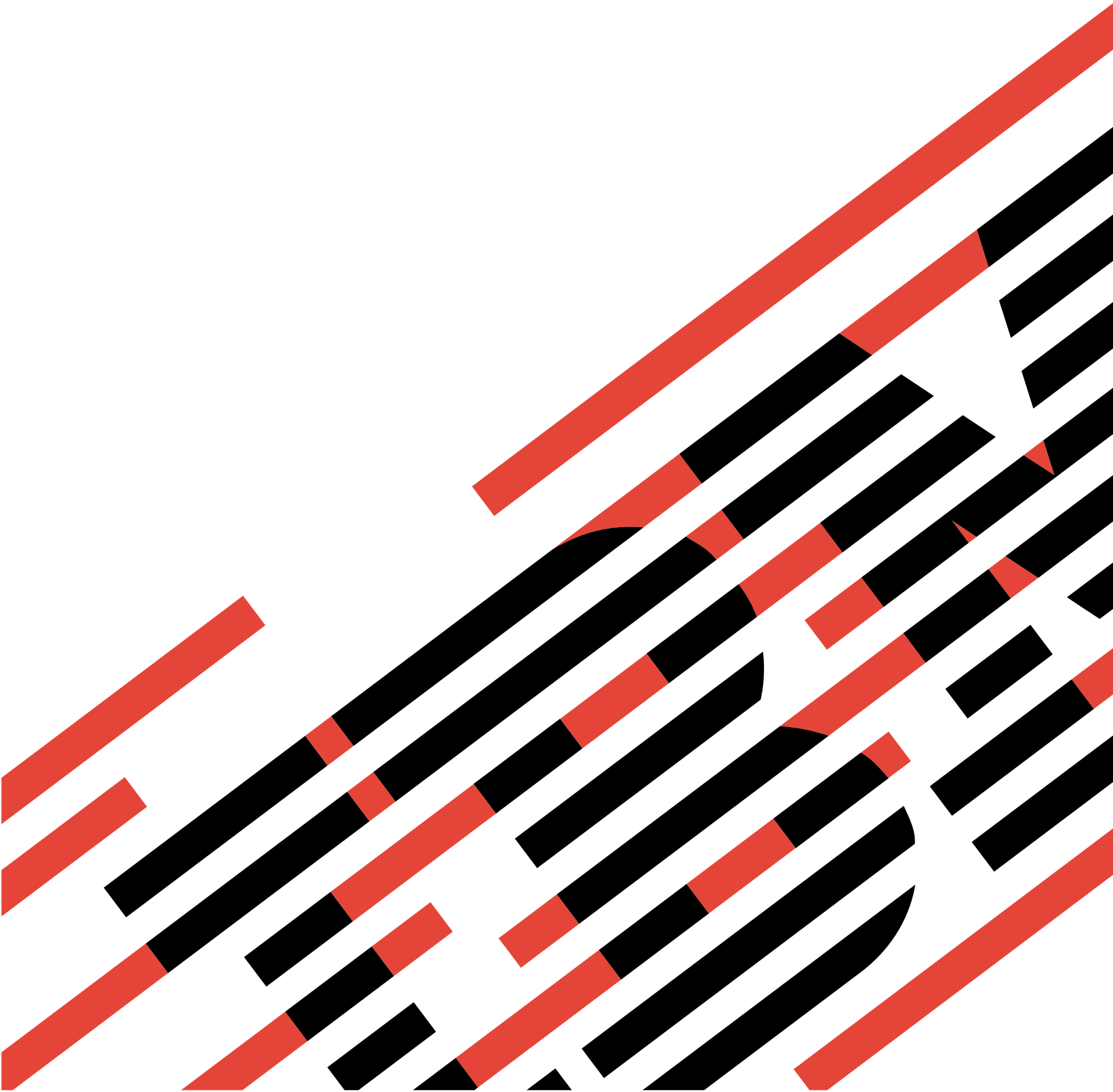
IBM

eserver

iSeries

iSeries용 DB2 Universal Database SQL 참조서

버전 5



IBM

@server

iSeries

iSeries용 DB2 Universal Database SQL 참조서

버전 5

목차

iSeries용 DB2 UDB SQL 참조서 관련 정보

이 책은 DB2 조회 관리자 및 SQL 개발 킷이 지원하는 SQL(Structured Query Language)을 정의합니다. 이 책에는 시스템 관리, 데이터베이스 관리, 어플리케이션 프로그래밍 및 연산에 대한 타스크를 위한 참조 정보가 수록되어 있습니다. 또한 시스템에서 사용되는 각 SQL문에 대한 구문, 사용 주, 키워드 및 예가 수록되어 있습니다.

안내서에 대한 자세한 정보는 다음 섹션을 참조하십시오.

- 『표준 준수』
- xvi 페이지의 『SQL 참조서를 읽어야 할 사람』
- xvii 페이지의 『SQL문 예에 관련된 전제』
- xviii 페이지의 『구문 도표 읽는 방법』
- xix 페이지의 『혼합 자료 값 설명 규칙』
- xix 페이지의 『SQL 액세스 가능성』
- xx 페이지의 『SQL 참조서에서 V5R2의 새로운 사항』

표준 준수

iSeries용 DB2 UDB 버전 5 릴리스 1은 다음과 같은 IBM 및 업계 SQL 표준을 준수합니다.

- ISO(국제 표준화 기구) 9075: 1992, 데이터베이스 언어 SQL - 항목 레벨
- ISO(국제 표준화 기구) 9075-4: 1996, 데이터베이스 언어 SQL - 제 4 부: 지속적 저장 모듈(SQL/PSM)
- ISO(국제 표준화 기구) 9075: 1999 데이터베이스 언어 SQL - 핵심
- ANSI(미국 표준 협회) X3.135-1992, 데이터베이스 언어 SQL - 항목 레벨
- 미국 표준 협회(ANSI) X3.135-4: 1996, 데이터베이스 언어 SQL - 제 4 부: 지속적 저장 모듈(SQL/PSM)
- ANSI(미국 표준 협회) 3.1355-1999 데이터베이스 언어 SQL - 핵심
- *IBM SQL 참조 버전 2*, SC26-8416.

표준을 정확히 준수하려면 표준 옵션을 사용하십시오. 자세한 내용은 772 페이지의 『SET OPTION』 및 SQL 사전컴파일러 명령의 SQLCURRULE를 참조하십시오.

SQL 참조서를 읽어야 할 사람

이 책은 iSeries 데이터베이스에 액세스하기 위해 SQL을 사용하는 어플리케이션을 작성하고자 하는 프로그래머를 대상으로 합니다.

이 책에서는 사용자가 SQL 프로그래밍 개념에서 제공하는 시스템 관리, 데이터베이스 관리 및 iSeries 서버의 어플리케이션 프로그래밍에 대하여 이해하고 있으며, 다음 항목에 대하여 어느 정도 지식을 갖추고 있다고 가정합니다.

- COBOL for iSeries
- ILE C 컴파일러
- ILE C++ 컴파일러
- ILE COBOL 컴파일러
- Toolbox for Java 또는 Developer Kit for Java
- ILE RPG 컴파일러
- iSeries PL/I
- REXX
- RPG III(RPG for iSeries의 일부)
- SQL(Structured Query Language)

이 책에서 언급하는 RPG 및 COBOL은 일반적인 RPG나 COBOL 언어를 의미합니다. COBOL for iSeries, ILE COBOL for iSeries, RPG for iSeries 또는 RPG III(RPG for iSeries의 일부)로 명시한 것은 동일하지 않은 각 제품의 특정 요소를 의미합니다.

이 책은 자습서라기보다 참조서입니다. 사용자가 이미 SQL 프로그래밍에 익숙하다는 가정을 전제로 합니다. 또한 이 책에서는 iSeries 서버 전용 어플리케이션을 작성하는 것으로 가정합니다.

SQL문, 명령문 구문 및 매개변수에 대한 자세한 내용은 SQL 프로그래밍 개념 책을 참조하십시오.

만일 다른 IBM 환경에 이식가능한 어플리케이션을 작성할 계획이면, 이러한 책(*IBM SQL 참조 버전 2* SC26-8416과 같은) 외에도 원하는 환경에 대한 책들을 반드시 참조하십시오.

자세한 내용은 다음 섹션을 참조하십시오.

- xvii 페이지의『SQL문 예에 관련된 전제』
- xviii 페이지의『구문 도표 읽는 방법』

SQL문 예에 관련된 전제

이 책에 표시된 SQL문의 예들은 SQL 프로그래밍 개념의 부록 A에 있는 샘플 표를 바탕으로 하며 다음을 가정합니다.

- 대화식 SQL 환경에서 표시되거나 COBOL로 작성되어 있습니다. EXEC SQL 및 END-EXEC은 COBOL 프로그램에서 SQL문을 구분하는 데 사용됩니다. COBOL 프로그램에서 SQL문 사용 방법의 설명은 SQL Programming with Host Languages 책에서 제공됩니다.
- 각 SQL 예는 몇 행에 걸쳐 보여지며 각 명령문 절이 분리된 행 하나마다 각각 나타납니다.
- SQL 키워드는 강조표시가 되어 있습니다.
- 예에서 사용된 표 이름은 SQL 프로그래밍 개념 책의 부록 A에서 제공된 샘플 표이며 CORPDATA 스키마를 사용합니다. 이 부록에 나와 있지 않은 표 이름은 사용자가 작성한 스키마를 사용해야 합니다. 다음 SQL문을 사용하여 사용자가 작성한 스키마에서 샘플 표를 작성할 수 있습니다.

  ```
  CALL QSYS.CREATE_SQL_SAMPLE ('your-schema-name')
  ```

- 연산된 열은 괄호()로 묶여 있습니다.
- SQL 명명 규칙이 사용됩니다.
- APOST 및 APOSTSQL 사전컴파일러 옵션이 전제됩니다(비록 COBOL 내의 디폴트는 아니지만). SQL 및 호스트 언어 명령문 내의 문자 스트링 상수는 어포스트로피(')로 구분됩니다.
- *HEX의 정렬 순서가 사용됩니다.

이 전제로부터 예가 발생할 때마다 명시됩니다.

또한 『코드 면책 사항』을 참조하십시오.

코드 면책 사항

이 문서에는 프로그래밍 예제가 들어 있습니다.

IBM은 귀하에게 유사한 기능을 귀하의 특정 요구에 맞게 조정하여 생성할 수 있도록 모든 프로그래밍 코드 예제를 사용할 수 있는 비독점적인 저작권 사용권을 부여합니다.

모든 샘플 예제는 IBM에 의해 예시 목적으로만 제공됩니다. 이러한 예제는 모든 조건 하에서 철저히 테스트된 것은 아닙니다. 따라서 IBM은 이들 프로그램의 신뢰성, 실용성 또는 기능에 대해 보증할 수 없습니다.

여기에 포함된 모든 프로그램은 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 어떠한 종류의 보증 없이 "현상태대로" 제공됩니다.

구문 도표 읽는 방법

이 책에서는 다음과 같이 정의된 구조를 사용하여 구문을 설명합니다.

- 구문 도표는 왼쪽에서 오른쪽, 위에서 아래, 행의 경로를 따라 읽습니다.

  ▶▶── 기호는 명령문의 시작을 나타냅니다.

  ──▶ 기호는 명령문 구문이 다음 행으로 이어짐을 나타냅니다.

  ▶── 기호는 명령문이 앞 행에서 이어짐을 나타냅니다.

  ──▶◀ 기호는 명령문의 끝을 나타냅니다.

  구문 프래그먼트는 |── 기호로 시작하여 ──| 기호로 끝납니다.

- 필수 항목은 수평 행(기본 경로)에 나옵니다.

```
▶▶─required_item─────────────────────────────────────────────────────────▶◀
```

- 선택적 항목은 기본 경로 밑에 나옵니다.

```
▶▶─required_item─┬──────────────┬──────────────────────────────────────────▶◀
                 └─optional_item─┘
```

  만일 선택적 항목이 기본 경로 위에 나오면 그 항목은 명령문 실행시 영향이 없는 것으로 읽기 전용으로만 사용됩니다.

```
                 ┌─optional_item─┐
▶▶─required_item─┴──────────────┴──────────────────────────────────────────▶◀
```

- 둘 이상의 항목을 선택하면 스택 안에 수직으로 각 행이 나옵니다.

  항목 중 하나를 *반드시* 선택해야 할 경우 스택의 한 항목이 기본 경로 위에 나옵니다.

```
▶▶─required_item─┬─required_choice1─┬──────────────────────────────────────▶◀
                 └─required_choice2─┘
```

  만일 항목 중 하나가 선택적이면 전체 스택이 기본 경로 밑에 나옵니다.

```
▶▶─required_item─┬──────────────────┬──────────────────────────────────────▶◀
                 ├─optional_choice1─┤
                 └─optional_choice2─┘
```

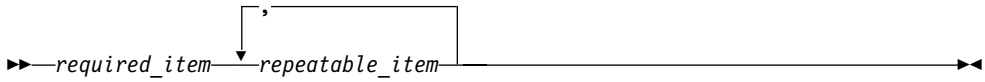
  만일 항목 중 하나가 디폴트이면 그 항목이 기본 경로 위에 나오며 나머지 선택사항들이 아래에 나옵니다.

```
                 ┌─default_choice──┐
▶▶─required_item─┼─────────────────┼──────────────────────────────────────▶◀
                 ├─optional_choice─┤
                 └─optional_choice─┘
```

- 기본 경로 위에 있는 좌측 화살표는 반복시킬 수 있는 항목을 나타냅니다.

```
                 ┌──────────────┐
                 ▼              │
▶▶─required_item─┴─repeatable_item─┴──────────────────────────────────────▶◀
```

  반복 화살표에 쉼표가 있으면 반복 항목을 쉼표로 분리시켜야 합니다.



스택 위의 반복 화살표는 스택에서 항목을 반복시킬 수 있음을 나타냅니다.

- 키워드는 대문자로 나옵니다(예를 들어, FROM). 보이는 그대로 철자를 입력해야 합니다. 변수는 모두 소문자로 나옵니다(예를 들어, *column-name*). 변수들은 사용자가 제공하는 이름이나 값을 나타냅니다.
- 마침표 표시, 괄호, 연산자 또는 기타 기호들이 있으면 구문의 일부로 기호들을 입력하십시오.
- 구문 다이어그램에는 우선 키워드 또는 표준 키워드만 들어 있습니다. 표준 키워드 외에 표준이 아닌 동의하거나 지원되는 경우, 구문 다이어그램이 아닌 **주** 섹션에 기술됩니다. 이식성을 높이기 위해, 우선 키워드 및 표준 키워드만 사용할 수 있습니다.

혼합 자료 값 설명 규칙

예에 혼합 자료 값이 나올 때, 다음 규칙이 적용됩니다.

규약	의미
S_O	EBCDIC SO 제어 문자를 나타냅니다(X' 0E').
S_I	EBCDIC SI 제어 문자를 나타냅니다(X' 0F').
sbcs-스트링	0개 이상의 1바이트 문자 스트링을 나타냅니다.
dbcs-스트링	0개 이상의 2바이트 문자 스트링을 나타냅니다.
'	DBCS 어포스트로피를 나타냅니다(EBCDIC X' 427D').
G	DBCS G를 나타냅니다(EBCDIC X' 42C7').

SQL 액세스 가능성

IBM은 장애인 집단에서도 쉽게 액세스할 수 있는 인터페이스 및 문서를 제공할 것을 약속드립니다. IBM의 액세스 가능성 지원에 대한 일반 정보는 http://www.ibm.com/able의 Accessibility Center를 참조하십시오.

SQL 액세스 가능성 지원은 다음 두 가지 기본 범주로 나누어집니다.

- iSeries Navigator는 iSeries 및 DB2 UDB로의 그래픽 사용자 인터페이스입니다. Windows 그래픽 사용자 인터페이스에서 지원되는 액세스 가능성 기능에 대한 정보는 Windows Help 색인의 '내게 필요한 옵션'을 참조하십시오.

- IBM Home Page Reader와 같은 Windows Reader 프로그램에서 온라인 문서, 온라인 도움말, 프롬트되는 SQL 인터페이스에 액세스할 수 있습니다. IBM Home Page Reader 및 기타 도구에 대한 정보는 Accessibility Center  를 참조하십시오.

IBM Home Page Reader를 사용하면 이 책의 모든 설명문, SQL Information Center의 모든 기사, 모든 SQL 메시지에 액세스할 수 있습니다. 그러나, SQL 구문 다이어그램은 복잡하기 때문에 독자는 구문 다이어그램을 건너뛸 것입니다. 쉽게 이용할 수 있도록 다음과 같은 두 가지 대안이 제공됩니다.

- 대화식 SQL 및 조회 관리자

  대화식 SQL 및 Query Manager는 SQL문에 대한 프롬트를 제공하는 종래의 파일 인터페이스입니다. 이것은 DB2 UDB 조회 관리자 및 SQL 개발 킷 부분입니다. 대화식 SQL 및 Query Manager에 대한 자세한 내용은 SQL Programming Concepts 및 Query Manager Use  책을 참조하십시오.

- SQL Assist

  SQL Assist는 SQL문으로 프롬트되는 인터페이스를 제공하는 그래픽 사용자 인터페이스입니다. 이것은 iSeries Navigator 부분입니다. 자세한 정보는 iSeries Navigator 온라인 도움말과 Information Center를 참조하십시오.

SQL 참조서에서 V5R2의 새로운 사항

이 책에서 다루는 새로운 주요 피처는 다음과 같습니다.

- ROWID 자료 유형과 ROWID 스칼라 함수
- IDENTITY 열 속성
- CREATE TABLE AS(부속 선택)
- DECLARE GLOBAL TEMPORARY 표
- 사용자 정의 표 함수
- COMMIT ON RETURN 프로시듀어
- 뷰에서 UNION
- 스칼라 부속 선택 확장 기능
- SET TRANSACTION에서 READ ONLY 및 READ WRITE
- SQL 프로시듀어, SQL 함수, SQL 트리거의 ITERATE 및 중첩 복합문
- 도출된 표 및 공통 표 표현식의 Fullselect
- 레이블된 기간에서 매개변수 마커
- 저장점
- SET SCHEMA 및 SET SQLID
- HOLD LOCATOR

- 선택 리스트에 필요하지 않은 ORDER BY 표현식
- 도출된 표 및 공통 표 표현식의 ORDER BY 및 FETCH FIRST n ROWS ONLY
- SQL문 길이는 64로 증가됩니다.
- 구분된 열 이름 ID의 길이는 증가됩니다.
- SUBSTRING 확장 기능
- VARCHAR 연결 증진
- SQL 프로시듀어, SQL 함수, SQL 트리거의 기존 소스 원본 디버그
- iSeries의 다중 관계형 데이터베이스
- 표준과 ODBC 및 JDBC 카탈로그 뷰
- C 도출 변수

제 1 장 개념

DB2 UDB for iSeries SQL 참조서에서는 다음과 같은 개념이 설명됩니다.

- 『관계형 데이터베이스(RDB)』
- 3 페이지의 『SQL』
- 6 페이지의 『스키마』
- 6 페이지의 『표』
- 7 페이지의 『키』
- 7 페이지의 『1차 키 및 고유 키』
- 8 페이지의 『참조 무결성』
- 10 페이지의 『검사 제한조건』
- 11 페이지의 『트리거』
- 13 페이지의 『색인』
- 14 페이지의 『뷰』
- 15 페이지의 『별명』
- 15 페이지의 『패키지 및 액세스 계획』
- 16 페이지의 『프로시듀어』
- 17 페이지의 『카탈로그』
- 18 페이지의 『어플리케이션 프로세스, 동시성 및 회복』
- 23 페이지의 『스레드』
- 24 페이지의 『분리 레벨』
- 28 페이지의 『분산 관계형 데이터베이스』
- 34 페이지의 『문자 변환』
- 38 페이지의 『정렬 순서』
- 39 페이지의 『권한부여 및 권한』
- 40 페이지의 『기억장치 구조』

관계형 데이터베이스(RDB)

관계형 데이터베이스(RDB)는 표 세트로 인식될 수 있는 데이터베이스로 관계형 자료 모델에 따라 조작될 수 있습니다. 관계형 데이터베이스(RDB)에는 자료를 저장, 액세스 및 관리하는 데 사용되는 오브젝트 세트가 들어 있습니다. 오브젝트 세트는 표, 뷰, 색인, 별명, 고유한 유형, 함수, 프로시듀어 및 패키지를 포함합니다.

사용자가 iSeries 시스템에서 액세스할 수 있는 관계형 데이터베이스에는 세 가지 유형이 있습니다.

시스템 관계형 데이터베이스

iSeries 시스템에 하나의 디폴트 관계형 데이터베이스(RDB)만 있습니다. 시스템 관계형 데이터베이스는 항상 해당 iSeries 시스템의 로컬에 존재합니다. 이 데이터베이스는 독립형 보조 기억장치 풀에 저장되지 않은 iSeries 시스템에 접속된 디스크에 존재하는 모든 데이터베이스 오브젝트로 구성됩니다. 독립형 보조 기억장치 풀에 대한 자세한 내용은 iSeries Information Center의 시스템 관리 범주를 참조하십시오.

시스템 관계형 데이터베이스의 디폴트 이름은 iSeries 시스템 이름과 같습니다. 다른 이름은 ADDRDBDIRE(RDB 디렉토리 입력 추가) 명령 또는 iSeries Navigator를 사용하여 지정할 수 있습니다.

사용자 관계형 데이터베이스

사용자는 시스템에 독립형 보고 기억장치 풀을 구성하여 iSeries 시스템에 추가 관계형 데이터베이스를 작성할 수 있습니다. 각 1차 독립 보조 기억장치 풀은 관계형 데이터베이스입니다. 독립 보조 기억장치 풀에 존재하는 모든 데이터베이스 오브젝트로 구성됩니다. 또한, 독립형 보조 기억장치 풀이 연결되어 있는 iSeries 시스템에 있는 시스템 관계형 데이터베이스의 모든 데이터베이스 오브젝트는 사용자 관계형 데이터베이스에 논리적으로 포함되어 있습니다. 따라서, 사용자 관계형 데이터베이스에서 작성된 스키마의 이름은 해당 사용자 관계형 데이터베이스 또는 관련 시스템 관계형 데이터베이스에 이미 존재하지 않는 것이어야 합니다.

시스템 관계형 데이터베이스의 오브젝트는 사용자 관계형 데이터베이스에 논리적으로 포함되지만, 시스템 관계형 데이터베이스와 사용자 관계형 데이터베이스의 특정 종속 관계는 허용되지 않습니다.

- 뷰는 참조된 표, 뷰 또는 함수와 동일한 관계형 데이터베이스에 존재하는 스키마로 작성되어야 합니다.
- 색인은 참조된 표와 동일한 관계형 데이터베이스에 존재하는 스키마로 작성되어야 합니다.
- 트리거나 제한조건은 기본 표와 동일한 관계형 데이터베이스에 존재하는 스키마로 작성되어야 합니다.
- 참조 제한조건의 상위 표 및 종속 표는 둘다 동일한 관계형 데이터베이스에 존재해야 합니다.
- 표는 참조된 고유한 유형와 동일한 관계형 데이터베이스에 존재하는 스키마로 작성되어야 합니다.
- 참조 제한조건의 상위 표 및 종속 표는 둘다 동일한 관계형 데이터베이스에 존재해야 합니다.

시스템 관계형 데이터베이스의 오브젝트와 사용자 관계형 데이터베이스의 오브젝트간의 다른 종속 관계는 허용되지 않습니다. 예를 들어, 사용자 관계형 데이터베이스의 스키마로 된 프로시듀어는 시스템 관계형 데이터베이스의 오브젝트를 참조할 수 있습니다. 그러나, 다른 관계형 데이터베이스가 사용 가능하지 않다면 해당 오브젝트에 대한 조작은 실패할 수 있습니다. 예를 들어, 사용자 관계형 데이터베이스가 단절변환되었다가 다른 시스템으로 연결변환된 경우입니다.

사용자 관계형 데이터베이스는 iSeries 시스템에 로컬로 존재하고 독립형 보조 기억장치 풀은 연결변환됩니다. 한 iSeries 시스템에서 독립형 보조 기억장치 풀를 단절변환한 다음 다른 iSeries 시스템에 연결변환할 수 있습니다. 따라서, 사용자 관계형 데이터베이스는 어떤 시점에서는 해당 iSeries 시스템에 로컬로 존재하다가 다른 시점에서는 리모트로 존재할 수 있습니다. 독립형 보조 기억장치 풀에 대한 자세한 내용은 iSeries Information Center의 시스템 관리 범주를 참조하십시오.

사용자 관계형 데이터베이스의 디폴트 이름은 독립형 보조 기억장치 풀의 이름과 같습니다. 다른 이름은 ADDRDBDIRE(RDB 디렉토리 입력 추가) 명령 또는 iSeries Navigator를 사용하여 지정할 수 있습니다.

리모트 관계형 데이터베이스

다른 iSeries 및 비iSeries 시스템에 있는 관계형 데이터베이스는 리모트로 액세스 될 수 있습니다. 이런 관계형 데이터베이스들은 ADDRDBDIRE(RDB 디렉토리 입력 추가) 명령 또는 iSeries Navigator를 사용하여 등록되어야 합니다.

데이터베이스 관리자는 iSeries 사용권 내부 코드 및 관계형 데이터베이스(RDB)를 관리하는 코드의 iSeries용 DB2 UDB 부분을 식별하기 위해 일반적으로 사용되는 이름입니다.

SQL

SQL은 관계형 데이터베이스(RDB)에서 자료를 정의하고 조작하기 위한 표준화된 언어입니다. 관계형 자료 모델에 따라, 데이터베이스는 표 세트로 인식되며, 표의 값에 의해 관계가 표시되고, 자료는 하나 이상의 기준 표에서 파생될 수 있는 결과표를 명시함으로써 검색됩니다.

SQL문은 데이터베이스 관리자에 의해 실행됩니다. 데이터베이스 관리자의 함수 중 하나는 결과표 스펙을 자료 검색을 최적화하는 일련의 내부 조작으로 변환하는 것입니다. 이런 변환은 SQL문이 *준비*될 때 발생합니다. 이런 변환을 *바인딩*이라고 합니다.

실행가능한 모든 SQL문은 실행되기 전에 반드시 준비되어야 합니다. 준비 결과는 실행가능하거나 조작 가능한 명령문 양식입니다. SQL문을 준비하는 메소드와 조작 양식의 영속성이 정적 SQL과 동적 SQL을 구분합니다.

정적 SQL

정적 SQL문의 소스 양식은 COBOL과 같은 호스트 언어로 작성된 어플리케이션 프로그램 안에 삽입됩니다. SQL문은 프로그램이 실행되기 전에 준비되며, SQL문 조작 양식은 프로그램 실행과 상관없이 지속됩니다.

정적 SQL문이 들어 있는 소스 프로그램은 컴파일 되기 전에 SQL 사전컴파일러에 의해 처리되어야 합니다. 사전컴파일러는 SQL문의 구문을 검사하여 호스트 언어 주석으로 바꾸고 호스트 언어 명령문을 생성하여 데이터베이스 관리자를 호출합니다.

SQL 어플리케이션 프로그램 준비에는 사전컴파일, 정적 SQL문 준비, 변경된 소스 프로그램 컴파일 등이 포함됩니다.

동적 SQL

동적 SQL문은 SQL 어플리케이션 실행 동안에 준비됩니다. 명령문의 조작 양식은 최종 SQL 프로그램이 호출 스택을 떠날 때까지 지속됩니다. SQL문 조작 양식은 정적 SQL문 PREPARE 또는 EXECUTE IMMEDIATE를 사용하는 프로그램에 의해 데이터베이스 관리자에 전달되는 문자 스트링입니다.

REXX 어플리케이션에 삽입 SQL문은 동적 SQL문입니다. 대화식 SQL 함수에 제공된 SQL문도 동적 SQL문입니다.

확장(Extanded) 동적 SQL

확장 동적 SQL문은 완전히 정적도 아니고 완전히 동적도 아닙니다. QSQPRCED API는 사용자에게 확장 동적 SQL 함수를 제공합니다. 동적 SQL처럼, 이 API를 사용하여 명령문을 준비, 설명, 실행할 수 있습니다. 동적 SQL과 달리, 이 API에 의해 패키지로 준비된 SQL문은 패키지 또는 명령문이 명시적으로 제거될 때까지 지속됩니다. 자세한 내용은 iSeries Information Center의 **프로그래밍** 범주에서 OS/400 API를 참조하십시오.

대화식 SQL

대화식 SQL 함수는 모든 데이터베이스 관리자와 연관됩니다. 본질적으로, 모든 대화식 SQL 함수는 터미널에서 명령문을 읽어 준비하여 동적으로 처리하고 그 결과를 사용자에게 표시하는 SQL 어플리케이션 프로그램입니다. 그런 SQL문은 *대화식으로* 발행됩니다. iSeries용 DB2 UDB에 대한 대화식 함수는 STRSQL 명령, STRQM 명령 또

는 iSeries Navigator에 대한 SQL 스크립트 지원으로 호출됩니다. SQL의 대화식 기능에 대한 자세한 정보는 SQL 프로그래밍 개념 및 조회 관리자 사용법  책을 참조하십시오.

SQL 호출 레벨 인터페이스(CLI) 및 ODBC(Open Database Connectivity)

DB2 호출 레벨 인터페이스는 동적 SQL문을 처리할 수 있도록 어플리케이션 프로그램에 함수가 제공되는 프로그래밍 인터페이스입니다. DB2 CLI를 사용하여 ILE 언어를 사용하는 사용자는 iSeries용 DB2 UDB가 제공하는 서비스 프로그램에 대한 프로시듀어 호출을 통해 직접 SQL 함수에 액세스할 수 있습니다. 또한 CLI 프로그램은 ODBC 자료 소스로 액세스할 수 있도록 해주는 ODBC(Open Database Connectivity) 소프트웨어 개발자 킷을 사용하여 컴파일할 수 있는데, ODBC 소프트웨어 개발자 킷은 Microsoft나 다른 업체를 통해 구할 수 있습니다. 삽입 SQL을 사용하는 경우와는 달리 사전컴파일이 필요없습니다. 호출 레벨 인터페이스를 사용하여 개발된 어플리케이션은 다양한 데이터베이스 각각에 대하여 컴파일하지 않고도 모두 실행될 수 있습니다. 호출 레벨 인터페이스를 통해 어플리케이션은 실행시 프로시듀어 호출을 사용하여 데이터베이스에 연결하고, SQL문을 발행하며, 리턴된 자료 및 상태 정보를 받을 수 있습니다.

DB2 CLI 인터페이스는 삽입 SQL에서 사용할 수 없는 많은 피처를 제공합니다. 예를 들면, 다음과 같습니다.

- CLI에서는 DB2 제품군의 데이터베이스 관리 시스템 전체에 걸친 데이터베이스 시스템 카탈로그 정보를 조회하고 검색하기 위해 일관성 있는 방법을 지원하는 함수 호출을 제공합니다. 따라서 각각의 데이터베이스 서버를 위한 카탈로그 조회를 작성할 필요가 없습니다.
- CLI를 사용하여 작성된 어플리케이션 프로그램에서 호출된 저장된 프로시듀어는 결과 집합을 이 프로그램들로 리턴할 수 있습니다.

사용할 수 있는 모든 함수 및 구문에 대한 설명은 SQL 호출 레벨 인터페이스 (ODBC) 책을 참조하십시오.

JDBC(Java Database Connectivity) 및 삽입 SQLJ(Java용 SQL) 프로그램

iSeries용 DB2 UDB에서는 JDBC(Java Database Connectivity) 및 삽입 SQLJ(Java용 SQL)의 두 가지 표준에 근거한 Java 프로그래밍 API를 구현합니다. 두 가지 모두 Java 어플리케이션 및 DB2에 액세스하는 애플릿을 작성하는 데 사용할 수 있습니다.

JDBC 호출은 Java 원시 메소드를 통해 DB2 CLI에 대한 호출로 변환됩니다. 두 가지 JDBC 드라이버인 IBM Developer Kit for Java 드라이버나 IBM Toolbox for

Java JDBC 드라이버를 통해 iSeries 데이터베이스에 액세스할 수 있습니다. IBM Toolbox for Java JDBC 드라이버에 대한 자세한 내용은 IBM Toolbox for Java를 참조하십시오.

JDBC는 정적 SQL을 사용할 수 없습니다. SQLJ 어플리케이션은 데이터베이스 연결 및 SQL 오류 처리 등의 타스크를 위한 기반으로 JDBC를 사용하지만 SQLJ 소스 파일에 삽입된 정적 SQL문을 포함할 수 있습니다. SQLJ 소스 파일은 최종 Java 소스 코드가 컴파일되기 전에 SQLJ 변환 프로그램을 사용하여 변환되어야 합니다.

JDBC와 SQLJ 어플리케이션에 대한 자세한 정보는 Developer Kit for Java 책을 참조하십시오.

스키마

스키마는 명명된 오브젝트의 집합입니다. 스키마는 관계형 데이터베이스에서 오브젝트를 논리적으로 분류하는 기능을 합니다. 스키마에 포함될 수 있는 오브젝트에는 표, 뷰, 별명, 함수, 프로시듀어, 유형 및 패키지가 있습니다. 스키마는 콜렉션 또는 라이브러리라고도 합니다.

또한 스키마는 관계형 데이터베이스의 한 오브젝트이기도 합니다. 스키마는 CREATE SCHEMA문을 사용하여 명시적으로 작성됩니다.[1]

A *스키마명*은 두 부분으로 이루어진 오브젝트명의 상위 부분으로 사용됩니다. 스키마에 포함된 오브젝트는 해당 오브젝트가 작성될 때 그 스키마에 할당됩니다. 오브젝트가 할당된 스키마는 스키마명으로 특별히 규정되었다면 그 오브젝트명으로 판별되고 규정되지 않은 경우 디폴트 스키마명으로 판별됩니다.

예를 들어 C라는 스키마를 작성합니다.

```
CREATE SCHEMA C
```

스키마 C에 X라는 표를 작성하려면 다음 명령문을 발생할 수 있습니다.

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

표

표는 사용자 자료를 저장하는 오브젝트입니다. 표는 데이터베이스 관리자에 의해 유지보수되는 논리 구조입니다. 표는 열 및 행으로 구성됩니다. 하나의 표 내에 행 순서는 계승되지 않습니다. 모든 열과 행이 교차하는 곳에 *값*이라는 특정 자료 항목이 있습니다. *열*은 동일한 유형 값 세트입니다. *행*은 *n*번째 값이 표의 *n*번째 열의 값이 되는 일련의 값입니다.

1. 스키마는 CRTLIB CL 명령을 사용하여 작성할 수 있지만, CREATE SCHEMA문을 사용하여 작성된 카탈로그 뷰와 저널은 CRTLIB를 사용하여 작성할 수 없습니다.

*기준 표*는 CREATE TABLE문으로 작성되며 지속적인 사용자 자료를 보유하는 데 사용됩니다. *결과표*는 데이터베이스 관리자가 하나 이상의 기준 표에서 선택하거나 생성하는 행 세트입니다.

기준 표는 하나의 이름을 가지며 하나의 다른 시스템명을 가질 수 있습니다. 시스템명은 OS/400에 의해 사용되는 이름입니다. *표 이름*이 SQL문의 어디에 지정되든 두 이름 중 하나를 사용할 수 있습니다. 자세한 내용은 541 페이지의 『CREATE TABLE』을 참조하십시오.

기준 표의 열은 하나의 이름을 가지며 하나의 다른 시스템 열 이름을 가질 수 있습니다. 시스템 열 이름은 OS/400에 의해 사용되는 이름입니다. *열 이름*이 SQL문의 어디에 지정되든 두 이름 중 하나를 사용할 수 있습니다. 자세한 내용은 541 페이지의 『CREATE TABLE』을 참조하십시오.

*분산 표*는 자료가 노드 그룹간에 분할되는 표입니다. 노드 그룹은 둘 이상의 시스템을 논리 그룹으로 묶는 오브젝트입니다. *분할 키*는 행이 속한 시스템을 판별하는 데 사용되는 분산 표에 있는 하나 이상의 열 세트입니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

*선언된 임시 표*는 DECLARE GLOBAL TEMPORARY TABLE문으로 작성되며 단일 어플리케이션을 대신하여 임시 자료를 보유하는 데 사용됩니다. 어플리케이션이 데이터베이스로부터 단절되면 이 표는 내부적으로 삭제됩니다.

키

*키*는 색인, 고유한 제한조건 또는 참조 제한조건의 설명에서와 같이 자체로 식별되는 하나 이상의 열입니다. 동일한 열이 하나 이상의 키의 부분이 될 수 있습니다. 하나 이상의 열로 이루어진 키를 복합 키라고 합니다.

*복합 키*는 동일한 표의 순서화된 열 세트입니다. 열 순서는 표 내의 순서에 의해 제한받지 않습니다. 복합 키와 관련하여 사용될 때 용어 값은 복합 값을 가리킵니다. 따라서, 『외부 키 값은 1차 키 값과 동일해야 한다』는 것과 같은 규칙은 외부 키 값의 각 구성요소가 1차 키 값의 해당 구성요소와 같아야 함을 의미합니다.

1차 키 및 고유 키

*고유 제한조건*은 키 값이 고유할 때만 유효한 규칙입니다. 고유한 값을 갖도록 제한된 키를 *고유 키*라고 하며 CREATE UNIQUE INDEX문을 사용하여 정의할 수 있습니다. 결과 고유 색인은 INSERT 및 UPDATE문이 실행되는 동안에 키에 고유함을 강제하기 위해 데이터베이스 관리자에 의해 사용됩니다. 대신에 고유 키를 정의할 수도 있습니다.

- 1차 키로서 CREATE TABLE 또는 ALTER TABLE문 사용. 표에 1차 키가 하나 이상 포함될 수 없습니다. 1차 키를 구성하는 열에 널값이 허용되지 않는다는 규칙을 강제하기 위해 CHECK 제한조건이 내재적으로 추가될 것입니다. 1차 키의 고유 색인을 1차 색인이라고 합니다.
- CREATE TABLE 또는 ALTER TABLE문의 UNIQUE 구 사용. 표는 UNIQUE 키를 임의의 수만큼 가질 수 있습니다.

참조 제한조건의 외부 키에 의해 참조되는 고유 키를 상위 키라고 합니다. 상위 키는 1차 키 또는 UNIQUE 키입니다. 참조 제한조건에서 표를 상위로 정의할 때 디폴트 상위 키는 1차 키입니다.

참조 무결성

*참조 무결성*은 모든 외부 키 값이 모두 유효한 데이터베이스 상태입니다. *외부 키*는 참조 제한조건 정의의 일부인 키입니다. *참조 제한조건*은 외부 키의 값이 다음의 경우에만 유효하다는 규칙입니다.

- 상위 키 값으로 나오는 경우
- 외부 키의 일부 구성요소가 널(null)인 경우

상위 키가 들어 있는 표를 참조 제한조건의 *상위 표*라고 하며, 외부 키가 들어 있는 표를 해당 표에서 *종속되어* 있다고 합니다.

참조 제한조건은 선택적이며, CREATE TABLE 명령문 및 ALTER TABLE 명령문에서 정의할 수 있습니다. 참조 제한조건은 INSERT, UPDATE 및 DELETE문 실행 동안 데이터베이스 관리자에 의해 강제됩니다. 행이 처리될 때 강제되는 RESTRICT의 삭제 및 갱신 규칙을 제외하면 강제는 명령문 완료시에 수행됩니다.

RESTRICT의 삭제 또는 갱신 규칙에 대한 참조 제한조건은 항상 다른 참조 제한조건에 앞서 강제됩니다. 기타 참조 제한조건은 순서에 상관없이 강제됩니다. 즉, 순서는 조작 결과에 영향을 미치지 않습니다. SQL문내에서 다음과 같습니다.

- 행은 CASCADE의 삭제 규칙이 있는 임의 수의 참조 제한조건에 의해 삭제 표시될 수 있습니다.
- 행은 SET NULL 또는 SET DEFAULT의 삭제 규칙이 있는 하나의 참조 제한조건에 의해서만 갱신될 수 있습니다.
- 참조 제한조건에 의해 갱신되었던 행은 CASCADE의 삭제 규칙이 있는 다른 참조 제한조건에 의해 삭제 표시될 수 없습니다.

참조 무결성 규칙은 다음과 같은 개념 및 전문용어와 관련있습니다.

상위 키 참조 제한조건의 1차 키 또는 고유 키

상위 행 최소 하나의 종속 행을 갖는 행

상위 표 최소 하나의 참조 제한조건에서 상위인 표. 표는 임의 수의 참조 제한조건에서 상위로 정의될 수 있습니다.

종속 표 최소 하나의 참조 제한조건에서 종속인 표. 표는 임의 수의 참조 제한조건에서 종속으로 정의될 수 있습니다. 종속 표는 상위 표가 될 수도 있습니다.

하위 표 표는 표 T에 종속되어 있거나 표 T 종속의 하위일 경우 표 T의 하위 입니다.

종속 행 최소 하나의 상위 행을 갖는 행

하위 행 행은 행 p에 종속되어 있거나 행 p 종속의 하위일 경우 행 p의 하위입니다.

참조 주기 세트의 각 표가 자체의 하위인 참조 제약 조건의 세트.

자체 참조 행 자체의 상위인 행

자체 참조 표 동일한 참조 제한조건에서 상위이고 종속인 표. 이런 제한조건을 *자동 참조 제한조건*이라고 합니다.

참조 제한조건의 삽입 규칙은 외부 키의 널(null)이 아닌 삽입 값이 상위 표의 상위 키 값과 일치해야 한다는 제한조건입니다. 값 구성요소가 널이면 복합 외부 키 값은 널입니다.

참조 제한조건의 갱신 규칙은 참조 제한조건이 정의될 때 명시됩니다. NO ACTION과 RESTRICT를 선택할 수 있습니다. 상위 또는 하위 표의 열이 갱신될 때 갱신 규칙이 적용됩니다. 갱신 규칙은 외부 키의 널(null)이 아닌 갱신 값이 상위 표의 상위 키 값과 일치해야 한다는 것입니다. 값 구성요소가 널이면 복합 외부 키 값은 널입니다.

참조 제한조건의 삭제 규칙은 참조 제한조건이 정의될 때 명시됩니다. NO ACTION, RESTRICT, CASCADE, SET NULL 또는 SET DEFAULT를 선택할 수 있습니다. SET NULL은 외부 키의 일부 열이 널값을 허용하는 경우에만 명시될 수 있습니다.

참조 제한조건의 삭제 규칙은 상위 열이 삭제될 때 적용됩니다. 보다 엄밀히, 이 규칙은 상위 표의 열이 삭제 또는 보급 삭제 조작(아래에 정의되어 있음)의 오브젝트이고 행이 참조 제한조건의 종속 표에 종속을 갖고 있을 때 적용됩니다. P가 상위 표를 가리키고, D는 종속 표를 가리키며 p는 삭제 또는 보급 삭제 조작의 오브젝트인 상위 행을 가리킵니다. 삭제 규칙이 다음과 같은 경우.

- RESTRICT 또는 NO ACTION, 오류가 발생하고 행은 삭제되지 않습니다.
- CASCADE, 삭제 조작이 D에서 p 종속에 보급됩니다.
- SET NULL, D에서 p의 각 종속 외부 키 각 널(null) 가능 열이 널로 설정됩니다.
- SET DEFAL, D에서 p의 각 종속 외부 키 각 열이 디폴트 값으로 설정됩니다.

표가 상위인 각 참조 제한조건은 고유의 삭제 규칙을 갖고 있으며, 모든 적용가능한 삭제 규칙이 삭제 조작 결과를 판별하는 데 사용됩니다. 따라서, RESTRICT 또는 NO ACTION의 삭제 규칙을 갖고 있는 참조 제한조건에서 종속을 갖고 있거나 RESTRICT 또는 NO ACTION의 삭제 규칙을 갖고 있는 참조 제한조건에서 종속인 임의의 종속에 대한 삭제 연속을 갖는 경우 행은 삭제될 수 없습니다.

상위 표 P로부터 행 삭제는 다른 표와 연관되며 이들 표의 행에 영향을 미칠 수도 있습니다.

- 표 D가 P 종속이고 삭제 규칙이 RESTRICT 또는 NO ACTION이면 D는 조작에 연관되기는 하지만 조작에 의해 영향을 받지는 않습니다.
- 표 D가 P 종속이고 삭제 규칙이 SET NULL이면 D는 조작에 연관되고 D의 행은 조작 동안 갱신될 수 있습니다.
- 표 D가 P 종속이고 삭제 규칙이 SET DEFAULT이면 D는 조작에 연관되고 D의 행은 조작 동안 갱신될 수 있습니다.
- 표 D가 P 종속이고 삭제 규칙이 CASCADE이면 D는 조작에 연관되고 D의 행은 조작 동안 삭제될 수 있습니다.

D의 행이 삭제되면 P에서 삭제 조작은 D로 보급됩니다. D도 상위 표인 경우 이 리스트에서 설명한 활동이 차례로 D 종속에 적용됩니다.

P에서 삭제 조작에 연관된 표는 P에 대하여 *삭제 연결*되어 있다고 합니다. 따라서 표가 P에 종속되거나 P 직렬로부터 삭제 조작이 이루어지는 표에 종속되는 경우 표는 삭제 연결된 것입니다.

검사 제한조건

*검사 제한조건*은 표의 모든 행 중 하나 이상의 열에서 허용되는 값을 명시하는 규칙입니다. 검사 제한조건은 선택적이며, SQL문 CREATE TABLE 및 ALTER TABLE을 사용하여 정의할 수 있습니다. 검사 제한조건에 대한 정의는 제한된 탐색 조건 양식입니다. 제한조건 중 하나는 표 T에서 검사 제한조건의 열 이름이 T 열을 식별해야 한다는 것입니다.

표는 임의 수의 검사 제한조건을 가질 수 있습니다. 다음과 같은 경우 데이터베이스 관리 프로그램에 의해 강제됩니다.

- 행이 표에 삽입될 때
- 표의 행이 갱신될 때

검사 제한조건은 각 행이 삽입되거나 갱신되는 각 검색 조건을 적용함으로써 강제됩니다. 탐색 조건 결과가 임의의 행에 대해 FALSE이면 오류가 발생합니다.

트리거

*트리거*는 지정된 표에서 삭제, 삽입 또는 갱신 조작이 발생할 때마다 자동으로 실행될 일련의 조치를 정의합니다. 이 SQL 연산이 실행되면 트리거가 활성화되었다고 합니다.[2]

시스템에서 사용할 수 있는 거의 모든 조작이 트리거에서 사용되는 조치에 포함될 수 있습니다. 그러나 다음과 같은 일부 조작은 허용되지 않습니다.

- 확약(commit) 또는 롤백(트리거 활동 및 트리거 이벤트에 동일한 확약 정의가 사용되는 경우)
- CONNECT, SET CONNECTION, DISCONNECT 및 RELEASE문

제한조건의 전체 리스트는 데이터베이스 프로그래밍 책을 참조하십시오.

트리거는 참조 제한사항 및 검사 제한사항을 사용하여 자료 무결성 규칙을 실행합니다. 트리거는 다른 표를 갱신하거나 삽입되었거나 갱신된 행의 값을 자동으로 생성 또는 변형하거나 DB2 내부 및 외부에서 조작을 수행하는 함수를 호출하는 데 사용될 수 있기 때문에 제한사항보다 더 강력합니다. 예를 들어 신규 값이 일정 양을 초과한 경우 열 갱신을 금지하는 대신 트리거가 유효한 값으로 대체하고 관리자에게 유효하지 않은 갱신에 대한 통지를 송신할 수 있습니다.

트리거는 서로 다른 자료 상태를 다루어야 하는 비즈니스 규칙을 정의하고 적용해야 하는 경우(예를 들어 10 퍼센트 이상 인상될 수 없는 급여의 경우) 매우 유용한 메카니즘입니다. 이러한 제한을 두기 위해서는 증가 전후로 급여 값을 비교해야 합니다. 둘 이상의 자료 상태를 처리할 필요가 없는 규칙의 경우 참조 제한사항 및 검사 제한사항을 사용하는 것을 고려해보십시오.

또한 트리거는 비즈니스 규칙을 실행하는 데 필요한 어플리케이션 논리를 데이터베이스로 이동시킴으로써 어플리케이션을 더 빨리 개발하고 더 쉽게 유지보수할 수 있도록 해줍니다. 예를 들어 데이터베이스에서 앞에서 언급한 표의 급여 열에 설정된 증가 제한 논리를 사용하여 DB2는 어플리케이션이 급여 열을 변경할 때 유효한지 검사합니다. 또한 논리가 변경되어도 어플리케이션 프로그램은 변경할 필요가 없습니다.

트리거는 선택적이며 CREATE TRIGGER문이나 ADDPFTRG(실제 파일 트리거 추가) CL 명령을 사용하여 정의합니다. 또한 DROP TRIGGER문이나 RMVPFTRG(실제 파일 트리거 제거) CL 명령을 사용하여 제거합니다. 트리거 작성에 대한 자세한 정보는 CREATE TRIGGER문을 참조하십시오. 트리거의 일반적인 내용에 대한 자세한 정보는 575 페이지의 『CREATE TRIGGER』문이나 SQL 프로그래밍개념 및 데이터베이스 프로그래밍 책을 참조하십시오.

2. 또한 ADDPFTRG CL 명령은 모든 읽기 조작에 대하여 활성화된 트리거를 정의합니다.

트리거가 활성화 되어야하는 시점을 판별하는 데 사용되는 트리거를 작성할 때 정의되는 몇 가지 기준이 있습니다.

- *주제 표*는 트리거가 정의되는 표를 정의합니다.
- *트리거 이벤트*는 주제 표를 수정하는 특정 SQL 연산을 정의합니다. 이 연산은 삭제, 삽입 또는 갱신이 될 수 있습니다.
- *트리거 활성화 시간*은 트리거가 주제 테이블에서 트리거 이벤트가 수행된 전에 활성화될 것인지 후에 활성화될 것인지를 정의합니다.

트리거를 활성화시키는 명령문에는 *영향을 받는 행 집합*이 포함됩니다. 즉 삭제되거나 삽입 또는 갱신되는 주제 표의 행입니다. *트리거 입도*는 트리거 조치가 명령문에 대하여 한 번 수행될 것인지 또는 영향을 받는 행 집합의 행 각각에 대하여 한 번 수행될 것인지를 정의합니다.

*트리거 조치*는 선택적 탐색 조건과 트리거가 활성화될 때마다 실행되는 SQL문 집합입니다. 탐색 조건이 참인 경우에만 이 SQL문이 실행됩니다.

트리거된 조치는 영향을 받는 행 집합의 값을 참조할 수 있습니다. 이것은 *전이 변수*를 사용하여 지원됩니다. 전이 변수는 참조가 이전 값(갱신 이전)에 대한 것인지 신규 값(갱신 이후)에 대한 것인지를 식별하는 지정된 이름으로 규정되는 주제 표의 열 이름을 사용합니다. 신규 값도 갱신 또는 삽입 트리거 전에 SET 전이 변수 명령문을 사용하여 변경할 수 있습니다. 영향을 받는 행 집합의 값을 참조하는 또 다른 방법은 *전이 표*를 사용하는 것입니다. 전이 표 역시 주제 표의 열 이름을 사용하는 데 영향을 받는 행 전체를 하나의 표로 처리하도록 지정된 이름을 사용합니다. 전이 표는 트리거 이후에만 사용할 수 있습니다. 별도의 전이 표를 이전 값 및 신규 값에 대하여 정의할 수 있습니다.

표, 이벤트 또는 활성화 시간을 조합하여 여러 트리거를 지정할 수 있습니다. 트리거가 활성화되는 순서는 작성된 순서와 동일합니다. 따라서 가장 최근에 작성된 트리거가 가장 마지막에 활성화됩니다.

트리거 활성화는 *트리거 직렬*을 유발할 수 있습니다. 이것은 다른 트리거 또는 동일한 트리거를 다시 활성화시키는 SQL문을 실행하는 트리거를 활성화시킨 결과입니다. 트리거된 조치는 원래 수정의 결과로 갱신을 유발하고, 이것이 추가적인 트리거 활성화를 유발할 수 있습니다. 트리거 직렬을 사용하여 한 번의 삭제, 삽입 또는 갱신 명령문으로 데이터베이스에 대한 중요한 변경을 일으키는 트리거를 연속적으로 활성화할 수 있습니다.

트리거에서 수행된 활동은 트리거 실행을 야기했던 조작의 일부로 간주됩니다. 따라서, 분리 레벨이 NC(확약 없음)가 아닌 경우 트리거 활동은 트리거 이벤트와 동일한 확약 정의를 사용하여 수행됩니다.

- 데이터베이스 관리자는 조작의 결과로 실행된 조작 및 트리거가 모두 완료되거나 중단되도록 합니다. 트리거 조작에 앞서 발생한 조작은 영향을 받지 않습니다.
- 데이터베이스 관리자는 조작 및 관련 트리거가 실행된 후에 모든 제한을 효율적으로 검사합니다(RESTRICT 삭제 규칙이 있는 제한의 경우 예외).

트리거는 트리거 실행을 야기시켰던 SQL문 내에 이미 삽입되거나 갱신된 행에 대한 삭제 또는 갱신 허용 여부를 지정하는 속성을 갖고 있습니다.

- 트리거가 정의될 때 ALWREPCHG(*YES)를 지정하면 SQL문 내에서 다음과 같이 됩니다.
  - 트리거는 동일한 SQL문에 의해 이미 삽입되거나 갱신된 임의의 행을 갱신 또는 삭제하도록 허용받습니다. 동일한 SQL문에 의해 야기된 트리거 또는 참조 제한 조건에 의해 삽입 또는 갱신된 행을 포함하기도 합니다.
- 트리거가 정의될 때 ALWREPCHG(*NO)를 지정하면 SQL문 내에서 다음과 같이 됩니다.
  - 동일한 SQL문에 의해 행이 삽입되거나 갱신되지 않은 경우에만 트리거에 의해 행이 삭제될수 있습니다. 따라서, 분리 레벨이 NC(확약 없음)가 아닌 경우 트리거 활동은 트리거 이벤트와 동일한 확약 정의를 사용하여 수행됩니다. 동일한 SQL문에 의해 야기된 트리거나 참조 제한조건에 의한 삽입 또는 갱신을 포함할 수도 있습니다.
  - 동일한 SQL문에 의해 행이 삽입되거나 갱신되지 않은 경우에만 트리거에 의해 행이 갱신될 수 있습니다. 따라서, 분리 레벨이 NC(확약 없음)가 아닌 경우 트리거 활동은 트리거 이벤트와 동일한 확약 정의를 사용하여 수행됩니다. 동일한 SQL문에 의해 야기된 트리거나 참조 제한조건에 의한 삽입 또는 갱신을 포함할 수도 있습니다.

CREATE TRIGGER문을 사용하여 작성된 모든 트리거는 묵시적으로 ALWREPCHG(*YES) 속성을 갖습니다.

색인

*색인*은 기준 표의 행에 대한 포인터 세트입니다. 각 색인은 하나 이상의 표 열에 있는 자료 값을 기준으로 합니다. *색인*은 표에 있는 자료로부터 분리된 오브젝트입니다. 색인을 요구할 때 데이터베이스 관리자가 이 구조를 빌드하고 자동으로 유지보수합니다.

색인은 하나의 이름을 가지며 하나의 다른 시스템명을 가질 수 있습니다. 시스템명은 OS/400에 의해 사용되는 이름입니다. 색인명이 SQL문에 명시되는 어느 곳에서든 둘 중 한 이름을 사용할 수 있습니다. 자세한 내용은 506 페이지의 『CREATE INDEX』를 참조하십시오.

데이터베이스 관리 프로그램은 두 가지 유형의 색인을 사용합니다.

- 2진 기수 트리 색인

2진 기수 트리 색인은 표의 행에 대해 특정 순서를 제공합니다. 데이터베이스 관리자는 이 색인을 다음과 같은 용도로 사용합니다.

  - 성능 향상을 위해. 대부분의 경우 색인이 없을 때보다 자료 액세스가 빠릅니다.
  - 고유성 예약을 위해. 고유 색인이 있는 표는 동일한 키를 갖는 행을 가질 수 없습니다.

- 코드화 벡터 색인

  코드화 벡터 색인은 표의 행에 특정 순서를 제공하지 않습니다. 데이터베이스 관리 프로그램은 이 색인을 성능 향상에만 사용합니다.

코드화 벡터 액세스 경로는 코드화 벡터 색인 도움말에 대한 작업을 하며, 고유한 벡터 색인에 코드를 지정하고 이 값을 배열에 표시하여 데이터베이스 파일에 액세스를 제공합니다. 배열 요소는 표시되어야 하는 고유 값 수에 따라 1, 2 또는 4바이트 길이가 될 수 있습니다. 압축 크기 및 상관 단순성 때문에 코드화 벡터 액세스 경로는 병렬로 보다 수월하게 처리 될 수 있는 빠른 스캔을 제공합니다.

SQL CREATE INDEX문을 사용하여 코드화 벡터 액세스 경로를 작성하십시오. 코드화 벡터 색인으로 조회 가속화  에 대한 자세한 정보는 iSeries용 DB2 UDB 웹페이지를 참조하십시오.

뷰

뷰는 하나 이상의 표에서 자료를 보는 다른 방법을 제공합니다.

뷰는 결과표에 대해 명명된 스펙입니다. 스펙은 SQL문에서 뷰가 참조될 때마다 실행되는 SELECT문입니다. 따라서, 뷰는 기준 표처럼 열과 행을 갖고 있다고 생각할 수 있습니다. 검색을 위해 모든 뷰는 기준 표처럼 사용할 수 있습니다. 뷰를 삽입, 갱신 또는 삭제 조작에 사용할 수 있는지 여부는 CREATE VIEW 설명시 설명한 정의에 따라 다릅니다(자세한 내용은 590 페이지의 『CREATE VIEW』를 참조하십시오).

색인은 뷰에 대해 작성될 수 없습니다. 그러나, 뷰가 기준으로 한 표에 대해 작성된 색인은 뷰에서 조작 성능을 향상시킬 수 있습니다.

뷰 열이 직접 기준 표 열에서 파생된 경우 해당 열은 기준 표의 열에 적용되는 제한조건을 계승합니다. 예를 들어, 뷰가 기준 표의 외부 키를 포함하고 있는 경우 해당 뷰를 사용하는 INSERT 및 UPDATE 조작은 기준 표에서와 동일한 참조 제한조건을 받습니다. 마찬가지로, 뷰의 기준 표가 상위 표인 경우 해당 뷰를 사용하는 DELETE 조작은 기준 표에서와 동일한 DELETE 조작 규칙을 적용받게 됩니다. 또한 뷰는 기준 표에 적용되는 트리거를 계승합니다. 예를 들어, 표의 기준 표가 갱신 트리거를 갖는 경우 트리거는 뷰에서 갱신이 수행될 때 시작됩니다.

뷰는 하나의 이름을 가지며, 하나의 다른 시스템명을 가질 수 있습니다. 시스템명은 OS/400에 의해 사용되는 이름입니다. 뷰 이름이 SQL문에 지정되는 어느 곳에서든 둘 중 한 이름을 사용할 수 있습니다. 자세한 내용은 590 페이지의 『CREATE VIEW』를 참조하십시오.

뷰 열은 하나의 이름을 가지며 하나의 다른 시스템명을 가질 수 있습니다. 시스템 열 이름은 OS/400에 의해 사용되는 이름입니다. 열 이름이 SQL문에 지정되는 어느 곳에서든 둘 중 한 이름을 사용할 수 있습니다. 자세한 내용은 590 페이지의 『CREATE VIEW』를 참조하십시오.

별명

*별명*은 표 또는 뷰에 대한 다른 이름입니다. 별명을 사용하여 기존 표나 뷰를 참조할 수 있는 경우에 표나 뷰를 참조할 수 있습니다.[3] 표와 뷰처럼 별명은 작성, 제거될 수 있으며, 관련된 주석 또는 레이블을 갖습니다. 별명을 사용하기 위해 권한은 필요없습니다. 그러나, 별명으로 참조되는 표와 뷰에 액세스하려면 현재 명령문에 대한 적절한 권한부여가 필요합니다.

별명은 하나의 이름을 가질 수 있으며, 하나의 다른 시스템명을 가질 수 있습니다. 시스템명은 OS/400에 의해 사용되는 이름입니다. 별명 이름이 SQL문에 지정되는 어느 곳에서든 둘 중 한 이름을 사용할 수 있습니다. 자세한 내용은 432 페이지의 『CREATE ALIAS』를 참조하십시오.

패키지 및 액세스 계획

분산 SQL 프로그램의 경우 *패키지*는 SQL문을 실행하는 데 사용된 제어 구조가 들어 있는 오브젝트입니다. 패키지는 프로그램 준비 동안 작성됩니다. 제어 구조는 SQL문의 바인드 또는 조작 양식으로 생각할 수 있습니다. 패키지에 있는 모든 제어 구조는 단일 소스 프로그램에 삽입 SQL문에서 파생됩니다.

패키지는 QSQPRCED API에 의해 작성될 수도 있습니다. QSQPRCED API로 작성된 패키지는 QSQPRCED API로만 사용할 수 있습니다. DRDA 프로토콜을 통해 서버에서는 사용할 수 없습니다. 자세한 내용은 iSeries Information Center의 **프로그래밍** 범주에서 OS/400 API를 참조하십시오.

QSQPRCED API는 ODBC, JDBC 및 SQLJ 인터페이스를 통해 실행된 SQL문을 캐쉬하기 위한 패키지를 작성하기 위해 Windows용 iSeries Access에 의해 사용됩니다.

비분산 SQL 프로그램의 경우 SQL문을 실행하는 데 사용된 제어 구조가 비분산 SQL 프로그램의 관련 영역에 저장됩니다.

3. 모든 문맥에서 별명을 사용할 수는 없습니다. 예를 들어, 데이터베이스 파일의 개별 멤버를 참조하는 별명은 DDL(data definition language)문에서 사용할 수 없습니다.

*액세스 계획*이라는 용어는 일반적으로 SQL문을 실행하는 데 사용된 SQL 프로그램이나 SQL 패키지 관련 영역에 있는 제어 구조를 설명하는 데 사용됩니다.

프로시듀어

프로시듀어(흔히 저장된 프로시듀어라고 함)는 일련의 조작을 수행하기 위해 호출될 수 있는 프로그래밍 구조입니다. 조작에는 호스트 언어 명령문 및 SQL문이 들어 있습니다.

프로시듀어는 일반적으로 SQL 프로시듀어나 외부 프로시듀어로 분류됩니다. SQL 프로시듀어는 SQL문만 포함합니다. 외부 프로시듀어는 SQL문이 있을 수도 있고 없을 수도 있는 호스트 언어 프로그램(또는 REXX의 경우 소스 파일 멤버)을 참조합니다. iSeries용 DB2 UDB에서는 외부 프로시듀어와 SQL 프로시듀어 모두 지원됩니다.

SQL 프로시듀어는 호스트 언어 프로시듀어와 동일한 장점을 제공합니다. 즉, 코드의 공통 부분은 작성만 하여 한번 유지보수만 하면 되고 몇 개 프로그램에서 호출할 수 있습니다. 호스트 언어와 SQL 모두 로컬 시스템에 있는 프로시듀어를 호출할 수 있습니다. 그러나, SQL은 리모트 시스템에 있는 프로시듀어도 호출할 수 있습니다. 사실, SQL 프로시듀어의 주요 장점은 분산 어플리케이션의 성능 특징을 향상시키는 데 사용할 수 있다는 것입니다.

몇 개의 SQL문이 리모트 시스템에서 실행되어야 한다고 가정합니다. 첫 번째 SQL문이 실행되면 어플리케이션 리퀘스터는 조작을 수행하기 위해 서버로 요구를 송신합니다. 그런 다음 명령문이 성공적으로 실행되는지 여부를 표시하고 선택적으로 결과를 리턴하는 응답을 기다립니다. 두 번째 및 각각의 후속 SQL문이 실행될 때 어플리케이션 리퀘스터는 다른 요구를 송신하고 다른 응답을 기다립니다. 서버의 프로시듀어에 동일한 SQL문이 저장된 경우 원격 프로시듀어를 참조하는 CALL문이 실행될 수 있습니다. CALL문이 실행될 때 어플리케이션 리퀘스터는 프로시듀어 호출하기 위해 현재 서버로 단일 요구를 송신합니다. 그런 다음 프로시듀어가 성공적으로 실행되는지 여부를 표시하고 선택적으로 결과를 리턴하는 단일 응답을 기다립니다.

다음 두 그림은 분산 어플리케이션에서 저장된 프로시듀어를 사용하여 리모트 요구를 제거할 수 있는 방법을 설명합니다.

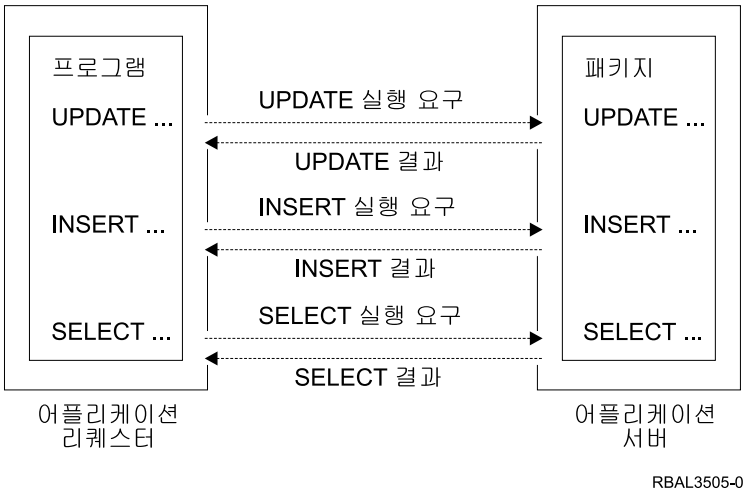


그림 1. 리모트 프로시듀어가 없는 어플리케이션

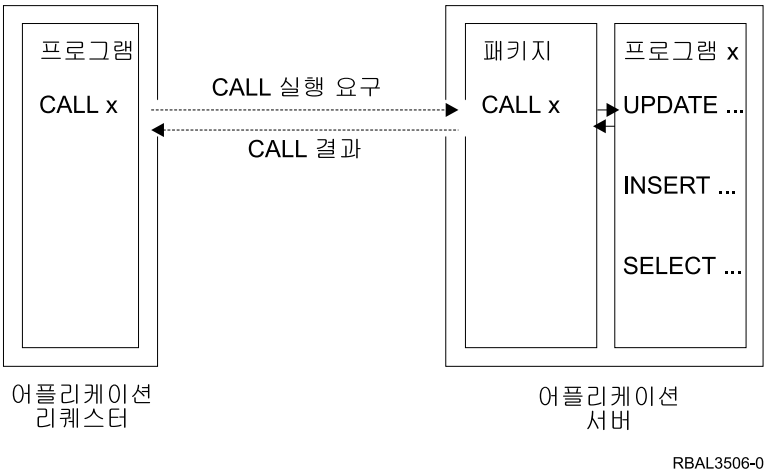


그림 2. 리모트 프로시듀어가 있는 어플리케이션

카탈로그

데이터베이스 관리 프로그램은 데이터베이스에서 자료에 대한 정보가 들어 있는 표 세트를 유지보수합니다. 이 표들을 집합적으로 *카탈로그*라고 합니다. *카탈로그* 표에는 시스템 상의 표, 매개변수, 프로시듀어, 패키지, 뷰, 색인 및 제한사항에 대한 정보가 들어 있습니다.

데이터베이스 관리자는 카탈로그 표에 대한 뷰를 제공합니다. 뷰는 다른 IBM SQL 제품에 대한 카탈로그와의 일관성을 제공하며, ANSI 및 ISO 표준(표준에서 정보 스키마(schema)라고 함)에 대한 카탈로그 뷰와의 일관성을 제공합니다. QSYS2의 카탈로그

뷰에는 시스템 상의 모든 표, 패키지, 뷰, 색인 및 제한사항에 대한 정보가 포함됩니다. 또한 SQL 스키마에는 해당 스키마의 표, 패키지, 뷰, 색인 및 제한사항에 대한 정보만을 담고 있는 뷰가 포함됩니다.

카탈로그 표 및 뷰는 다른 데이터베이스 표 및 뷰와 같습니다. 권한이 있다면 SQL문을 사용하여 시스템의 다른 표에서 자료를 검색하는 것과 같은 방법으로 자료를 볼 수 있습니다. 데이터베이스 관리 프로그램은 카탈로그에 언제든지 데이터베이스 오브젝트에 대한 정확한 설명이 들어 있도록 합니다.

카탈로그 표 및 뷰에 대한 자세한 내용은 927 페이지의 부록 G『iSeries용 DB2 UDB 카탈로그 뷰』를 참조하십시오.

어플리케이션 프로세스, 동시성 및 회복

모든 SQL 프로그램은 어플리케이션 프로세스의 일부로서 실행됩니다. OS/400에서 어플리케이션 프로세스를 작업이라고 합니다. ODBC, JDBC, DRDA의 경우, 연결이 종료되면 어플리케이션 프로세스가 종료되는데 이 경우에도 사용중인 작업은 종료되지 않고 재사용될 수 있습니다. 어플리케이션 프로세스는 하나 이상의 활성 그룹으로 이루어집니다. 각 활성 그룹은 하나 이상의 프로그램 실행을 수반합니다. 프로그램은 디폴트가 아닌 활성 그룹이나 디폴트 활성 그룹에서 실행됩니다. ILE 컴파일러에 의해 작성된 프로그램을 제외한 모든 프로그램이 디폴트 활성 그룹에서 실행됩니다.

활성화 그룹에 대한 자세한 정보는ILE 개념 책을 참조하십시오.

확약 정의를 사용하는 어플리케이션 프로세스는 하나 이상의 확약 정의로 실행될 수 있습니다. 확약 정의는 활성 그룹 레벨 또는 작업 레벨에서 확약 제어 범위를 지정하는 방법을 제공합니다. 확약 제어를 사용하는 활성 그룹은 확약 정의 중 하나와만 연관됩니다.

확약 정의는 STRCMTCTL(확약 제어 시작) 명령을 통해서 명시적으로 시작될 수 있습니다. 아직 시작되지 않은 경우 확약 정의는 첫 번째 SQL문이 COMMIT(*NONE)가 아닌 다른 분리 레벨에서 실행될 때 내재적으로 시작됩니다. 하나 이상의 활성 그룹이 작업 확약 정의를 공유할 수 있습니다.

19 페이지의 그림 3은 어플리케이션 프로세스, 해당 어플리케이션 프로세스에서 활성 그룹 및 확약 정의간의 관계를 표시합니다. 활성 그룹 A와 B는 활성 그룹으로 범위지정된 확약 제어로 실행됩니다. 이들 활성 그룹은 고유한 확약 정의를 갖습니다. 활성 그룹 C는 확약 제어로 실행되지 않으며, 확약 정의를 갖지 않습니다.

작업 레벨 확약 정의가 없는 어플리케이션 프로세스

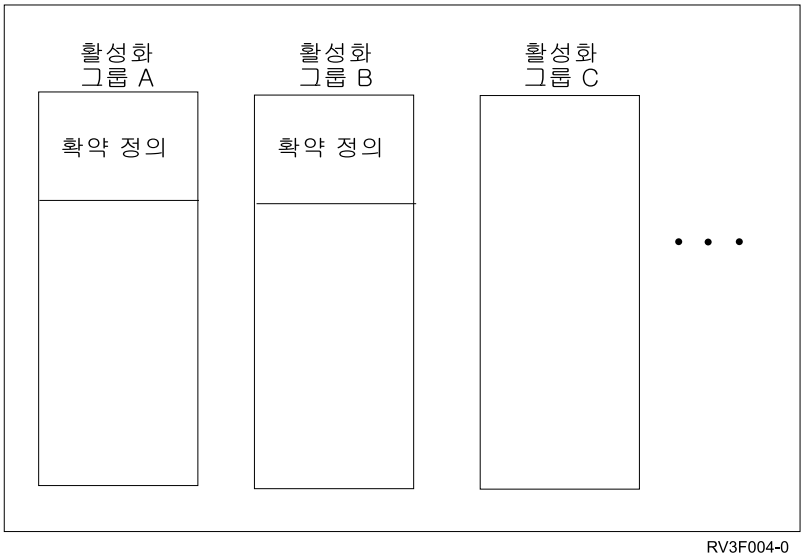


그림 3. 작업 확약 정의가 없는 활성 그룹

그림 4는 어플리케이션 프로세스, 해당 어플리케이션 프로세스에서 활성 그룹 및 확약 정의를 표시합니다. 일부 활성 그룹은 작업 확약 정의로 실행되고 있습니다. 활성 그룹 A와 B는 작업 확약 정의에서 실행되고 있습니다. 활성 그룹 A또는 B에서 확약 또는 롤백 조작은 확약 제어가 동일한 확약 정의로 범위지정되기 때문에 모두 영향을 미칩니다. 이 예에서 활성 그룹 C는 분리 확약 정의를 갖습니다. 이 활성 그룹에서 수행된 확약(commit) 및 롤백 조작은 C 안에서 이루어지는 조작에만 영향을 미칩니다.

작업 레벨 확약 제어가 없는 어플리케이션 프로세스

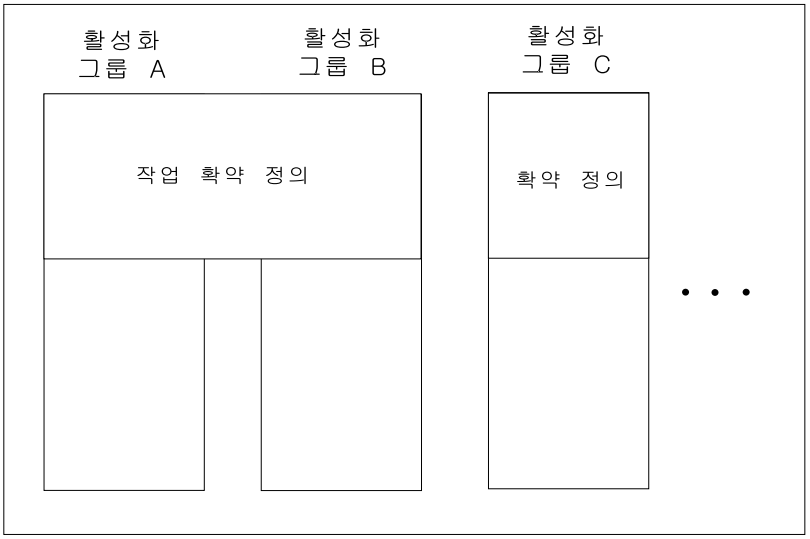


그림 4. 작업 확약 정의가 있는 활성 그룹

확약 정의에 대한 자세한 정보는 확약 제어 주제를 참조하십시오.

잠금, 확약, 롤백

다른 확약 정의를 사용하는 어플리케이션 프로세스 및 활성 그룹은 동시에 동일한 자료에 액세스를 요구할 수 있습니다. 잠금은 그런 상황하에서 자료 무결성(data integrity)을 유지보수하는 데 사용됩니다. 잠금은 두 개의 어플리케이션 프로세스가 동시에 동일한 자료 열을 갱신하지 못하도록 합니다.

데이터베이스 관리자는 다른 확약 정의를 사용하는 활성 그룹에 의해 감지되지 않은 한 활성 그룹의 확약되지 않은 변경을 보유하기 위해 잠금을 예약합니다. 오브젝트 잠금 및 기타 자원은 활성 그룹에 할당됩니다. 행 잠금이 확약 정의에 할당됩니다.

디폴트 활성 그룹이 아닌 다른 활성 그룹이 *정상적*으로 종료될 때 데이터베이스 관리자는 활성 그룹이 획득한 모든 잠금을 해제합니다. 사용자는 대부분의 잠금이 곧 해제되도록 명시적으로 요구할 수도 있습니다. 이 조작을 *확약*이라고 합니다. 확약 후에도 열린 커서와 관련된 오브젝트 잠금은 해제되지 않습니다.

데이터베이스 관리자의 회복 함수는 확약 정의에서 확약되지 않은 변경을 중단하는 방법을 제공합니다. 데이터베이스 관리자는 다음과 같은 상황에서 확약되지 않은 변경을 내재적으로 중단할 수 있습니다.

- 어플리케이션 프로세스가 종료될 때 디폴트 활성 그룹과 연관된 확약 정의 하에서 수행된 모든 변경이 중단됩니다. 디폴트 활성 그룹이 아닌 다른 활성 그룹이 *비정상적*으로 종료될 때 활성 그룹과 관련된 확약 정의하에서 수행된 모든 변경은 중단됩니다.
- 분산 작업 단위를 사용하고 있고 리모트 시스템에서 변경을 확약하려고 시도하는 동안 실패가 발생하면 리모트 연결과 관련된 확약 정의하에서 수행된 모든 변경은 중단됩니다.
- 분산 작업 단위를 사용하고 있고 해당 사이트에서 실패로 인해 리모트 시스템으로부터 중단 요구가 수신되면 리모트 연결과 관련된 확약 정의하에서 수행된 모든 변경은 중단됩니다.

사용자는 데이터베이스 변경이 중단되도록 명시적으로 요구할 수도 있습니다. 이 조작을 *롤백*이라고 합니다.

활성 그룹을 대신하여 데이터베이스 관리자에 의해 예약된 잠금은 작업 단위가 종료될 때까지 보류됩니다. LOCK TABLE문에 의해 명시적으로 예약된 잠금은 COMMIT HOLD 또는 ROLLBACK HOLD를 사용하여 작업 단위가 종료되면 작업 단위 종료 후에 보류될 수 있습니다.

커서는 커서가 위치한 행을 내재적으로 잠글 수 있습니다. 이 잠금은 다음과 같은 함수를 합니다.

- 다른 확약 정의와 관련된 다른 커서가 동일한 행을 잠그지 못하도록 합니다.

- 다른 확약 정의와 관련된 DELETE 또는 UPDATE문이 동일한 행을 잠그지 못하도록 합니다.

작업 단위

작업 단위(또한 *논리적 작업 단위* 또는 *회복 단위*라고도 함)는 회복가능한 일련의 조작입니다. 각 확약 정의는 하나 이상의 작업 단위 실행을 수반합니다. 확약 정의는 단일 작업 단위를 갖습니다.

확약 정의가 시작될 때 또는 이전 작업 단위가 확약이나 롤백 조작으로 종료될 때 작업 단위가 시작됩니다. 작업 단위는 확약 조작, 롤백 조작 또는 활성 그룹 종료로 종료됩니다. 확약(commit) 또는 롤백 조작은 확약 또는 롤백이 종료되는 작업 단위 내의 데이터베이스 변경에만 영향을 미칩니다. 변경이 확약되지 않은 동안 분리 레벨 COMMIT(*CS), COMMIT(*RS) 및 COMMIT(*RR)에서 실행되는 다른 확약 정의를 사용하는 다른 활성 그룹은 변경을 감지할 수 없습니다. 변경은 확약될 때까지 보류됩니다. 일단 변경이 확약되면 다른 확약 정의에서 실행되는 다른 활성 그룹은 변경에 액세스할 수 있으며, 변경은 더 이상 중단되지 않습니다.

작업 단위 시작 및 끝은 활성 그룹 내에서 일관성을 정의합니다. 예를 들어, 뱅킹 트랜잭션은 한 계좌에서 다른 계좌로 자금을 전송합니다. 그런 트랜잭션은 자금을 첫 번째 계좌에서 인출하여 두 번째 계좌에 넣을 것을 요구합니다. 인출 단계 후 자료는 일치하지 않습니다. 자금이 두 번째 계좌로 추가된 후에만 다시 일관성이 확립됩니다. 두 단계를 모두 완료하면 확약 조작을 사용하여 작업 단위를 종료할 수 있습니다. 확약 조작 후 다른 확약 정의를 사용하는 활성 그룹도 변경이 가능하게 됩니다.

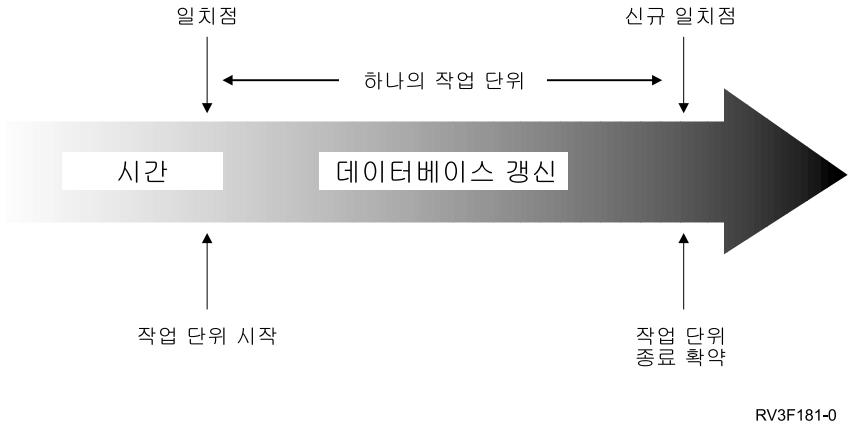


그림 5. 확약 명령문이 있는 작업 단위

작업 롤백

데이터베이스 관리자는 작업 단위의 모든 변경사항 또는 선택된 변경사항을 역처리할 수 있습니다. 모든 변경사항을 역처리해야만 일관성이 보장됩니다.

모든 변경사항 롤백

SQL ROLLBACK문을 TO SAVEPOINT절 없이 사용하면 전체 롤백 조작이 수행됩니다. 만약 롤백 연산이 성공적으로 수행되었다면 데이터베이스 관리자는 확약되지 않은 변경을 중단하여 작업 단위가 최기화 되었을 때 있었던 자료 일관성을 복원합니다. 즉, 아래의 다이어그램에서와 같이, 데이터베이스 관리자는 작업을 취소합니다.

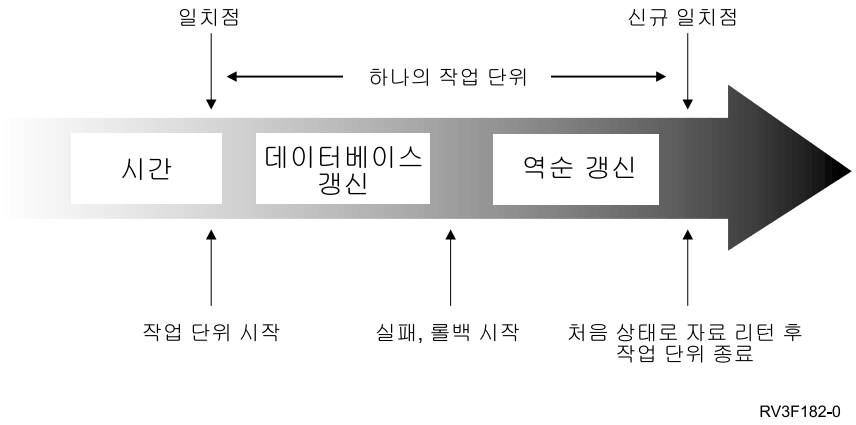


그림 6. 롤백 명령문이 있는 작업 단위

저장점을 사용하여 선택된 변경사항 롤백

*저장점*은 작업 단위 중 특정 시점에서의 자료 상태를 나타냅니다. 어플리케이션 프로세스는 작업 단위 내에 저장점을 설정한 다음, 로직이 지시하는 대로 저장점이 설정된 이후의 변경사항만을 롤백할 수 있습니다. 예를 들어, 예약 트랜잭션의 일부는 항공 노선 및 호텔 객실을 기록하는 작업이 될 수 있습니다. 항공 노선은 예약할 수 있지만 호텔 객실은 예약할 수 없는 경우, 어플리케이션 프로세스는 항공 노선을 예약하기 전에 트랜잭션에 가한 모든 데이터베이스 변경사항은 취소하지 않고 항공 노선 예약만을 취소하고자 할 것입니다. SQL 프로그램은 SQL SAVEPOINT문을 사용하여 저장점을 설정하고, SQL ROLLBACK문을 TO SAVEPOINT절과 함께 사용하여 설정된 특정 저장점 또는 최종 저장점의 변경사항을 취소하고, RELEASE SAVEPOINT문을 사용하여 저장점을 삭제합니다.

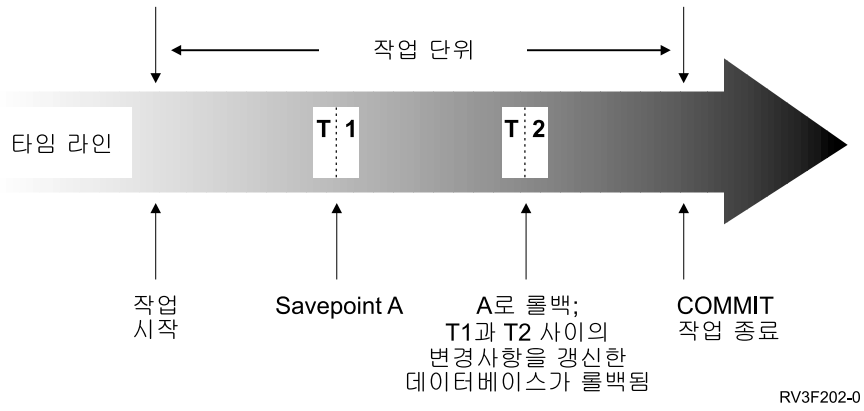


그림 7. Rollback문 및 Savepoint문이 있는 작업 단위

스레드

OS/400에서 어플리케이션 프로세스는 하나 이상의 스레드로 구성될 수도 있습니다. 디폴트로 하나의 스레드는 작업시 다른 스레드와 동일한 확약 정의 및 잠금을 공유합니다. 따라서, 각 스레드는 동일한 작업 단위에서 조작할 수 있으며 하나의 스레드가 확약하거나 롤백할 때 모든 스레드에 의해 수행된 변경을 모두 확약하거나 롤백할 수 있습니다. 이런 유형의 처리는 복수 스레드가 협력하여 병렬로 단일 타스크를 수행하는 경우 유용합니다.

또는 스레드가 작업의 다른 스레드로부터 독립적인 변경을 수행하는 데 유용합니다. 이 때 스레드는 확약 정의 및 잠금을 공유하기를 원하지 않을 수 있습니다. 게다가, 작업은 복수 데이터베이스 연결 및 트랜잭션 정보를 보다 면밀히 제어하기 위해 SQL 서버 모드를 사용할 수 있습니다. 일반적인 복수 스레드 작업은 이런 제어를 필요로 합니다. 이런 유형의 처리를 수행하기 위한 몇 가지 방법이 있습니다.

- 스레드에서 실행되는 프로그램이 분리 활성 그룹을 사용하는지 확인합니다 (ACTGRP(*NEW)를 사용하지 않도록 주의하십시오).
- 첫 번째 SQL문을 발행하기 전에 작업이 SQL 서버 모드로 실행되는지 확인합니다. SQL 서버 모드는 자료 액세스가 어플리케이션에서 발생하기 전에 다음 매커니즘 중 하나를 사용하여 작업에 대해 활성화될 수 있습니다.
  - ODBC API, SQLSetEnvAttr()을 사용하고, 임의의 자료 액세스를 수행하기 전에 SQL_ATTR_SERVER_MODE 속성을 SQL_TRUE로 설정하십시오.
  - Change Job API, QWTCHGJB()를 사용하고 임의의 자료 액세스를 수행하기 전에 'SQL 서버 모드' 키를 설정하십시오.
  - JDBC를 통해 데이터베이스에 액세스하려면 JAVA를 사용하십시오. JDBC는 자동으로 서버 모드를 사용하여 JDBC의 필요한 의미를 보유합니다.

SQL 서버 모드가 설정될 때 모든 SQL문은 요구를 처리할 독립적 서버 작업에 전달됩니다. SQL에 대한 서버 모드 작동은 다음을 포함합니다.

- 삽입 SQL에 대해, 작업의 각 스레드는 내재적으로 데이터베이스에 대한 하나의 연결(따라서, 자신이 소유하는 확약가능한 트랜잭션만 받습니다.)
- ODBC/CLI 및 JDBC의 경우 각 연결은 데이터베이스에 대한 독립형 연결만 표시하며 분리 엔티티로 확약되어 사용될 수 있습니다.

자세한 정보는 SQL 호출 레벨 인터페이스(ODBC) 책을 참조하십시오.

다음 SQL 지원은 스레드 보호가 아닙니다.

- DRDA를 통한 리모트 액세스
- ALTER TABLE
- COMMENT
- CREATE ALIAS
- CREATE DISTINCT TYPE
- CREATE FUNCTION
- CREATE INDEX
- CREATE PROCEDURE
- CREATE SCHEMA
- CREATE TABLE
- CREATE TRIGGER
- CREATE VIEW
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- GRANT
- LABEL
- RENAME
- REVOKE

자세한 내용은 iSeries Information Center의 멀티스레드 어플리케이션을 참조하십시오.

분리 레벨

SQL문 실행 동안 사용된 *분리 레벨*은 활성 그룹이 동시 활성 그룹 실행에서 분리되는 정도를 판별합니다. 따라서, 활성 그룹 P가 SQL문을 실행할 때 분리 레벨은 다음을 판별합니다.

- P에 의해 검색되는 행 및 P에 의한 변경이 다른 동시 실행 활성 그룹에 대해 사용가능한 정도.

- 동시에 실행되는 활성 그룹에 의한 데이터베이스 변경이 P에 영향을 미칠 수 있는 정도.

분리 레벨은 SQL 프로그램의 속성 또는 SQL 패키지로 지정되며 SQL 패키지 또는 SQL 프로그램을 사용하는 활성 그룹에 적용됩니다. iSeries용 DB2 UDB는 분리 레벨을 지정하는 몇 가지 방법을 제공합니다.

- CRTSQLxxx, STRSQL 및 RUNSQLSTM 명령에서 COMMIT 매개변수를 사용하여 디폴트 분리 레벨을 지정합니다.
- SET OPTION문을 사용하여 삽입 SQL이 들어 있는 모듈이나 프로그램 소스 안에서 디폴트 분리 레벨을 지정합니다.
- SET TRANSACTION문을 사용하여 작업 단위 내에서 디폴트 분리 레벨을 대체합니다. 작업 단위가 종료될 때 분리 레벨은 작업 단위 시작시에 가졌던 값으로 리턴됩니다.
- SELECT, SELECT INTO, INSERT, UPDATE, DELETE 및 DECLARE CURSOR문에서 고립절을 사용하여 특정 명령문 또는 커서에 대한 디폴트 분리 레벨을 대체합니다. 분리 레벨은 고립절이 들어 있는 명령문 실행시에만 적용되며 현재 작업 단위에서 지연 중인 변경에는 영향을 미치지 않습니다.

이런 분리 레벨은 해당 자료를 자동으로 잠금으로써 지원됩니다. 잠금 유형에 따라 이것은 다른 확약 정의를 사용하는 동시 활성 그룹에 의한 자료 액세스를 제한하거나 방해합니다. 각 데이터베이스 관리자는 최소 두 가지 유형의 잠금을 지원합니다.

공유 자료에 대한 읽기 전용 조작에 대해 다른 확약 정의를 사용하는 동시 활성 그룹을 제한합니다.

배타적 다른 확약 정의를 사용하는 동시의 활성 그룹이 자료를 갱신하거나 삭제하지 못하도록 합니다. COMM(*RS), COMMIT(*CS) 및 COMMIT(*RR)를 실행하는 다른 확약 정의를 사용하는 동시 활성 그룹이 자료를 읽지 못하도록 합니다. COMMIT(*UR) 및 COMMIT(*NC)를 실행하는 다른 확약 정의를 사용하는 동시 활성 그룹이 자료를 읽도록 허용합니다.

다음 분리 레벨에 대한 설명은 행 단위의 자료 잠금을 참조합니다. 개별 구현은 기준 표 행보다 더 큰 물리 장치(PU)에서 자료를 잠글 수 있습니다. 그러나, 논리적으로 잠금은 모든 제품에 걸쳐 기준 표 행에서 발생합니다. 마찬가지로, 데이터베이스 관리자는 상위 레벨에 대한 잠금을 할 수 있습니다. 활성 그룹은 적어도 최소 요구된 잠금 레벨은 보장받습니다.

iSeries용 DB2 UDB는 5개의 분리 레벨을 지원합니다. No Commit를 제외한 모든 분리 레벨에 대해 데이터베이스 관리자는 삽입, 갱신 또는 삭제되는 모든 행에 배타적 잠금을 합니다. 이것은 작업 단위 동안 변경된 행이 작업 단위가 완료될 때까지 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경되지 않음을 보장합니다. 분리 레벨은 다음과 같습니다.

- 반복가능한 읽기(RR)

  레벨 RR은 다음을 보장합니다.

  – 작업 단위 동안 읽혀진 행이 작업 단위가 완료될 때까지 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경되지 않습니다.[4]

  – 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경된 행(또는 현재 UPDATE 행 잠금으로 잠긴 행)은 확약될 때까지 읽을 수 없습니다.

  배타적 잠금 뿐만 아니라, 레벨 RR에서 실행되는 활성 그룹은 읽는 모든 행에 대하여 최소한 공유 잠금을 예약합니다. 게다가, 잠금은 활성 그룹이 다른 확약 정의를 사용하는 동시 활성 그룹의 영향에서 완전히 분리될 수 있도록 수행됩니다.

  iSeries용 DB2 UDB는 COMMIT(*RR)를 통해 반복가능한 읽기를 지원합니다. 반복가능한 읽기 분리레벨은 읽혀지거나 갱신되는 행이 들어 있는 표를 잠금으로써 지원됩니다. ANS 및 ISO 표준에서 반복가능한 읽기는 일련화가능이라고 합니다.

- 읽기 안정성(RS)

  레벨 RR처럼 레벨 RS는 다음을 보장합니다.

  – 작업 단위 동안 읽혀진 행이 작업 단위가 완료될 때까지 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경되지 않음을 보장합니다.[4]

  – 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경된 행(또는 현재 UPDATE 행 잠금으로 잠긴 행)은 확약될 때까지 읽을 수 없습니다.

  RR과 달리 RS는 다른 확약 정의를 사용하는 동시 활성 그룹의 영향에서 활성 그룹을 완전히 분리시키지는 않습니다. 레벨 RS에서 동일한 조회를 한 번 이상 발행하는 활성 그룹은 추가 행을 참조할 수 있습니다. 이런 추가 행을 *팬텀 행*이라고 합니다.

  예를 들면, 팬텀 행은 다음 상황에서 발생할 수 있습니다.

  1. 활성 그룹 P1이 일부 탐색 조건을 만족하는 행 세트 *n*을 읽습니다.
  2. 활성 그룹 P2가 탐색 조건을 만족하는 하나 이상의 행을 삽입하고 그 삽입을 확약합니다.
  3. P1이 다시 동일한 탐색 조건을 갖는 행 세트를 읽고 원본 행과 P2가 삽입한 행 모두를 받습니다.

  배타적 잠금 뿐만 아니라, 레벨 RS에서 실행되는 활성 그룹은 읽는 모든 행에 대하여 최소한 공유 잠금을 예약합니다.

---

4. **WITH HOLD** 커서의 경우 이 규칙은 행이 실제로 읽혀진 경우에 적용됩니다. 읽기 전용 **WITH HOLD** 커서의 경우 행은 앞의 작업 단위에서 실제로 읽혔을 수 있습니다.

iSeries용 DB2 UDB는 COMMIT(*ALL) 또는 COMMIT(*RS)를 통해 읽기 안정성을 지원합니다. ANS 및 ISO 표준에서 읽기 안정성을 반복가능한 읽기라고 합니다.

- 커서 안정성(CS)

  레벨 RR 및 RS처럼 레벨 CS는 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경된 행(또는 UPDATE 행 잠금으로 현재 잠긴 행)이 확약될 때까지 읽혀질 수 없음을 보장합니다. RR 및 RS와 달리 레벨 CS는 모든 갱신가능한 커서의 현재 행이 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경되지 않는다는 것만 보장합니다. 따라서, 작업 단위 동안 읽혀진 행은 다른 확약 정의를 사용하는 다른 활성 그룹에 의해 변경될 수 있습니다. 배타적 잠금 뿐만 아니라 레벨 CS에서 실행되는 활성 그룹은 모든 커서의 현재 행에 대한 공유 잠금을 확보할 수 있습니다.

  iSeries용 DB2 UDB는 COMMIT(*CS)를 통해 커서 안정성을 지원합니다. ANS 및 ISO 표준에서 커서 안정성을 확약된 읽기라고 합니다.

- 확약되지 않은 읽기(UR)

  INSERT문에서 사용된 읽기 전용 커서, 부속 조회 또는 subselect가 있는 SELECT INTO, FETCH의 경우 레벨 UR은 다음을 허용합니다.

  – 작업 단위 동안 읽혀진 행은 다른 확약 정의하에서 실행되는 다른 활성 그룹에 의해 변경됩니다.
  – 변경이 확약되지 않았다 하더라도 읽혀질 다른 확약 정의하에서 실행되는 다른 활성 그룹에 의해 변경된 행(또는 UPDATE 행 잠금으로 현재 잠긴 행)이 읽혀질 수 있습니다.

  다른 조작의 경우 레벨 CS 규칙이 적용됩니다.

  iSeries용 DB2 UDB는 COMMIT(*CHG) 또는 COMMIT(*UR)를 통해 확약되지 않은 읽기를 지원합니다. ANS 및 ISO 표준에서 확약되지 않은 읽기를 읽기 비확약이라고 합니다.

- 확약 없음(NC)

  모든 조작에 대해, 레벨 UR 규칙이 적용됩니다.

  – 확약(commit) 및 롤백은 SQL문에 영향을 주지 않습니다. 커서는 닫히지 않고 LOCK TABLE 잠금은 해제되지 않습니다. 그러나 해제 중단 상태의 연결은 종료됩니다.
  – 변경은 성공적인 각 조작 끝에 확약되며, 다른 확약 정의를 사용하는 다른 어플리케이션 그룹에 의해 즉시 액세스되거나 변경될 수 있습니다.

  iSeries용 DB2 UDB는 COMMIT(*NONE) 또는 COMMIT(*NC)를 통해 확약 없음을 지원합니다.

레코드 잠금 지속기간에 대한 자세한 설명은 SQL 프로그래밍 개념 책의 확약 제어 주제에 있는 논의와 표를 참조하십시오.

주: *(분산 어플리케이션의 경우)* 서버에서 요구 분리 레벨이 지원되지 않으면 분리 레벨은 지원되는 다음 상위 분리 레벨로 올라갑니다. 예를 들어 서버에서 RS가 지원되지 않으면 RR 분리 레벨이 사용됩니다.

분산 관계형 데이터베이스

*분산 관계형 데이터베이스*는 표 세트 및 상이하지만 상호연결된 컴퓨터 시스템에 걸쳐 퍼져 있는 다른 오브젝트로 이루어집니다. 각 컴퓨터 시스템은 환경에서 표를 관리하는 관계형 데이터베이스(RDB) 관리자를 갖고 있습니다. 데이터베이스 관리자는 데이터베이스 관리자가 다른 컴퓨터 시스템에서 SQL문을 실행할 수 있도록 하는 방법으로 서로 통신하고 협력합니다.

분산 관계형 데이터베이스는 공식 리퀘스터-서버 프로토콜 및 함수으로 구축됩니다. *어플리케이션 리퀘스터*는 연결에 대한 어플리케이션 종료를 지원합니다. 어플리케이션으로부터의 데이터베이스 요구를 분산 데이터베이스 네트워크에서 사용하기 적합한 통신 프로토콜로 변환합니다. 이러한 요구는 연결의 다른 쪽 끝에서 *서버*에 의해 수신되고 처리됩니다.[5] 어플리케이션 리퀘스터와 서버는 함께 통신 및 위치 고려사항을 처리하여 어플리케이션이 이 고려사항으로부터 분리되어 로컬 데이터베이스에 액세스하는 것처럼 운영할 수 있습니다. 간단한 분산 관계형 데이터베이스 환경은 그림 8에서 설명합니다.

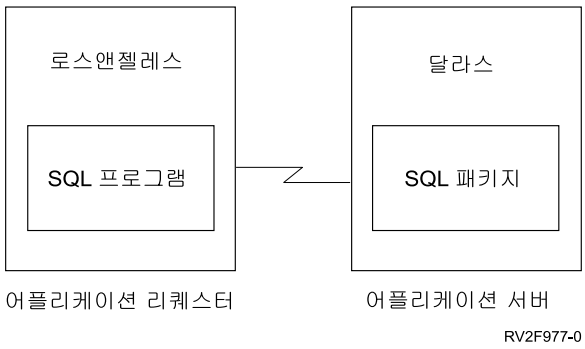


그림 8. 분산 관계형 데이터베이스 환경

분산 관계형 데이터베이스 구조(DRDA)에 대한 자세한 정보는 분산 관계형 데이터베이스 구조 참조서 를 참조하십시오.

5. 이를 *어플리케이션 서버* 라고 합니다.

데이터베이스 서버

활성 그룹은 표나 뷰를 참조하는 SQL문이 실행되기 전에 데이터베이스 관리자의 서버로 연결되어야 합니다.

*연결*이란 활성 그룹과 로컬 또는 리모트 서버 간의 연관입니다. 연결은 어플리케이션에 의해 관리됩니다. CONNECT문은 서버로의 연결을 설정하고 이 서버를 활성 그룹의 현재 서버로 만드는 데 사용할 수 있습니다.

서버는 활성 그룹이 시작되는 환경에 대하여 로컬일 수도 있고 리모트일 수도 있습니다(분산 관계형 데이터베이스를 사용하지 않아도 서버는 존재합니다). 이 환경에는 CONNECT문에서 식별될 수 있는 서버를 설명하는 로컬 디렉토리가 포함됩니다. 디렉토리에 대한 자세한 정보는 다음 iSeries Information Center 주제의 디렉토리 명령(ADDRDBDIRE, CHGRDBDIRE, DSPRDBDIRE, RMVRDBDIRE 및 WRKRDBDIRE)을 참조하십시오.

- SQL 프로그래밍 개념
- 분산 데이터베이스 프로그래밍
- CL 명령

표나 뷰를 참조하는 정적 SQL문을 실행하기 위해 서버는 명령문의 바인드 양식을 사용합니다. 이 바인드 양식은 데이터베이스 관리자가 바인드 조작을 통해 이전에 작성한 패키지에서 가져옵니다. 해당 패키지는 다음 조합으로 판별됩니다.

- CRTSQLxxx 명령에서 SQLPKG 매개변수에 의해 지정된 패키지명. CRTSQLxxx 명령의 설명은 SQL Programming with Host Languages 책을 참조하십시오.
- 패키지 및 프로그램이 동시에 동일한 소스에서 작성되었음을 확인하는 내부 일관성 토큰.

모든 IBM 관계형 데이터베이스(RDB) 제품은 IBM SQL에 대한 부가 제품을 지원합니다. 이들 확장 중 일부는 제품에 특정적이며, 일부는 하나 이상의 제품에 의해 공유됩니다.

대부분의 경우 일부 명령문과 절을 지원하지 않는 데이터베이스 관리자의 어플리케이션 리퀘스터를 통해 어플리케이션이 실행되고 있더라도 어플리케이션은 현재 연결되어 있는 서버의 데이터베이스 관리자가 지원하는 명령문과 절을 사용할 수 있습니다. 제한사항은 913 페이지의 부록 F『SQL문의 특성』에 나열되어 있습니다.

CONNECT(유형 1) 및 CONNECT(유형 2)

구문은 동일하나 의미는 다른 두 가지 유형의 CONNECT문이 있습니다.

- CONNECT(유형 1)은 리모트 작업 단위에 사용됩니다. 421 페이지의『CONNECT(유형 1)』를 참조하십시오.

- CONNECT(유형 2)는 분산 작업 단위에 사용됩니다. 427 페이지의 『CONNECT(유형 2)』를 참조하십시오.

차이점 요약을 보려면 925 페이지의 『CONNECT(유형 1) 및 CONNECT(유형 2) 차이점』을 참조하십시오.

리모트 작업 단위

SQL문 리모트 준비 및 실행을 위해 *리모트 작업 단위* 함수가 제공됩니다. 컴퓨터 시스템 A의 활성 그룹은 컴퓨터 시스템 B의 서버에 연결할 수 있습니다. 그러면 하나 이상의 작업 단위 내에서 그 활성 그룹은 B의 오브젝트를 참조하는 정적 또는 동적 SQL 문을 무제한으로 실행할 수 있습니다. B에서 작업 단위를 종료한 후 활성 그룹은 컴퓨터 시스템 C의 서버에 연결할 수 있으며 이런 방식으로 계속할 수 있습니다.

대부분의 SQL문은 리모트로 준비되어 실행될 수 있는데 다음과 같은 제한사항을 갖습니다.

- 단일 SQL문에서 참조되는 오브젝트는 모두 동일한 서버가 관리해야 합니다.
- 한 작업 단위 내의 SQL문은 모두 동일한 서버에 의해 실행되어야 합니다.

리모트 작업 단위 연결 관리

활성 그룹은 언제든 세 가지 상태 중 하나입니다.

연결가능 및 연결됨
연결 불가능 및 연결됨
연결가능 및 연결되지 않음

다음 도표는 상태 변환을 나타냅니다.

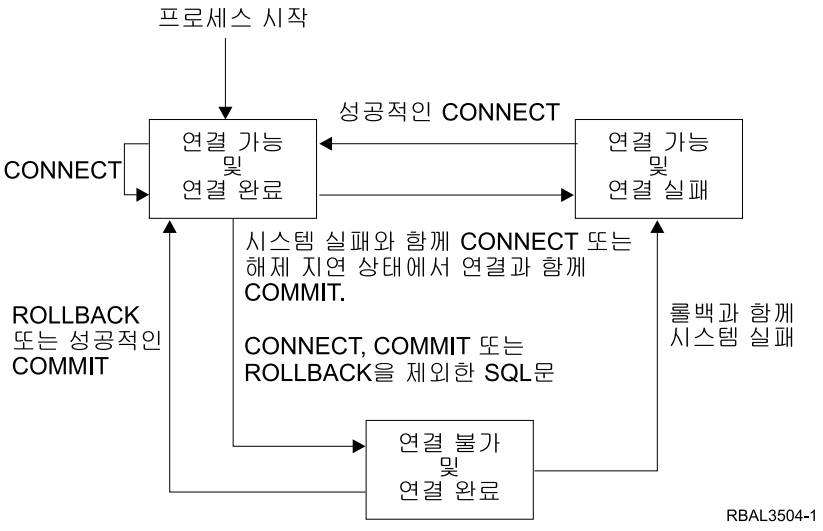


그림 9. 리모트 작업 단위 활성 그룹 연결 상태 변환

활성 그룹의 초기 상태는 *연결가능* 및 *연결된* 상태입니다. 활성 그룹이 연결되어 있는 서버는 CRTSQLxxx 및 STRSQL 명령의 RDB 매개변수에 의해 판별되며 내재적 CONNECT 조작을 수반할 수 있습니다. 내재적 CONNECT 조작은 내재적 또는 명시적 CONNECT 조작이 이미 성공적으로 발생했거나 실패한 경우 발생할 수 없습니다. 따라서 활성 그룹은 두 번 이상 서버에 내재적으로 연결될 수 없습니다.

연결가능 및 연결된 상태: 활성 그룹이 서버에 연결되어 있으며 CONNECT문이 실행될 수 있습니다. 활성 그룹은 연결 불가능 및 연결된 상태로부터 롤백 또는 성공적인 확약을 완료할 때 또는 CONNECT문이 연결가능 및 연결되지 않은 상태로부터 성공적으로 실행될 때 이 상태에 들어갑니다.

연결불가능 및 연결된 상태: 활성 그룹이 서버에 연결되어 있지만 서버를 변경하기 위해 CONNECT문이 정상적으로 실행될 수 없습니다. 활성 그룹은 CONNECT, COMMIT 또는 ROLLBACK이 아닌 SQL문을 실행할 때 연결가능 및 연결된 상태로부터 이 상태에 들어갑니다.

연결가능 및 연결되지 않은 상태: 활성 그룹이 서버에 연결되어 있지 않습니다. 유일하게 실행될 수 있는 SQL문은 CONNECT입니다.

활성 그룹은 다음과 같은 경우 이 상태에 들어갑니다.

- 연결이 이미 해제되었고 성공적인 확약이 실행된 경우
- SQL DISCONNECT문을 사용하여 연결이 단절된 경우
- 연결이 연결가능한 상태에 있지만 CONNECT문이 실패한 경우

CONNECT가 연결가능 상태에서 활성 그룹을 제거하지 않기 때문에 연속 CONNECT 명령문은 성공적으로 실행될 수 있습니다. 활성 그룹이 현재 연결되어 있는 서버로의 CONNECT는 다른 CONNECT문과 마찬가지로 실행됩니다. CONNECT, COMMIT, DISCONNECT, SET CONNECTION, RELEASE 또는 ROLLBACK이 아닌 SQL문이 앞에 올 때 CONNECT는 성공적으로 실행할 수 없습니다(COMMIT(*NC)로 실행되는 경우 제외). 오류를 피하려면 CONNECT문을 실행하기 전에 확약이나 롤백 조작을 실행합니다.

어플리케이션 지시 분산 작업 단위

*어플리케이션 지시 분산 작업 단위 함수*이 리모트 작업 단위와 동일한 방식으로 SQL문 리모트 준비 및 실행을 위해 제공됩니다. 리모트 작업단위와 마찬가지로 컴퓨터 시스템 A의 활성 그룹이 컴퓨터 시스템 B의 서버에 연결될 수 있으며, 작업 단위를 종료하기 전에 B의 오브젝트를 참조하는 정적 또는 동적 SQL문을 무제한 실행할 수 있습니다. 단일 SQL문에서 참조되는 오브젝트는 모두 동일한 서버가 관리해야 합니다. 그러나 리모트 작업 단위와는 달리 서버가 그 수에 관계없이 동일한 작업 단위에 참가할 수 있습니다. 확약 또는 롤백 조작은 작업 단위를 종료합니다.

분산 작업 단위는 APPC 및 TCP/IP 연결에 대하여 완벽하게 지원됩니다.

어플리케이션 지시 분산 작업 단위 연결 관리

언제든 다음과 같습니다.

- 활성 그룹은 항상 *연결된* 또는 *연결되지 않은* 상태이며, 0개 이상의 연결로 이루어진 세트를 갖습니다. 활성 그룹의 각 연결은 연결의 서버명에 의해 고유하게 식별됩니다.
- SQL 연결은 항상 다음 상태 중 하나입니다.
  - 현재 및 보류
  - 현재 및 해제 지연 중
  - 휴지 및 보류
  - 휴지 및 해제 지연 중

활성 그룹의 초기 상태: 활성 그룹은 초기에 연결된 상태이며, 정확히 하나의 연결을 갖습니다. 연결 초기 상태는 *현재 및 보류*입니다.

다음 도표는 상태 변환을 나타냅니다.

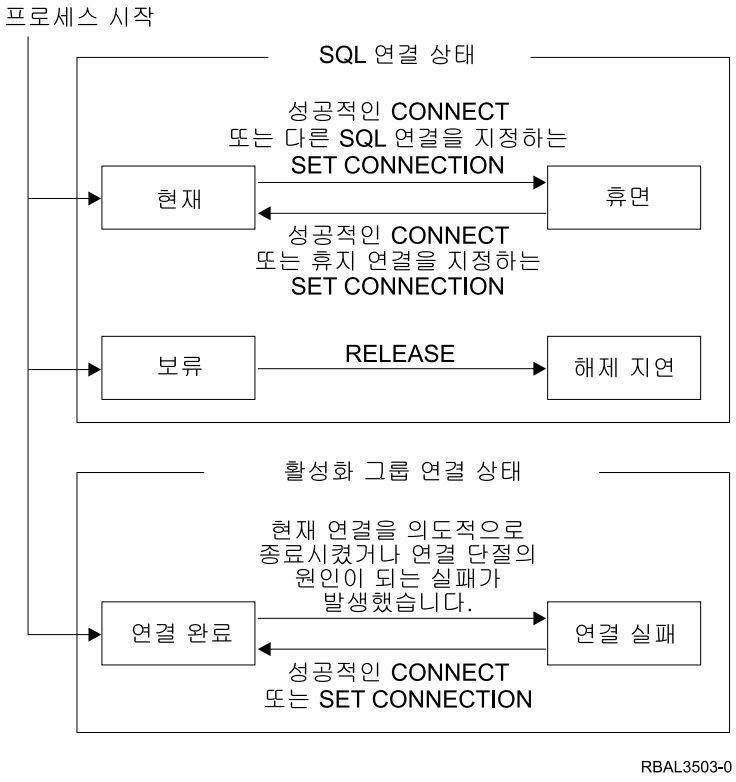


그림 10. 어플리케이션 지시 분산 작업 단위 연결 및 활성 그룹 연결 상태 변환

연결 상태

어플리케이션 프로세스가 CONNECT문을 성공적으로 수행하는 경우 다음과 같습니다.

- 현재 연결이 휴지 상태 및 보류 상태에 놓입니다.
- 서버명이 연결 세트에 추가되고, 새로운 연결은 현재 및 휴지 상태에 놓입니다.

서버명이 이미 활성 그룹 기존 연결 세트에 있는 경우 오류가 발생합니다.

휴지 상태의 연결은 SET CONNECTION문을 통해 현재 상태에 놓입니다. 연결이 현재 상태에 놓일 때 이전 연결이 있는 경우 휴지 상태에 놓입니다. 활성 그룹의 기존 연결 세트에서 하나 이상의 연결이 현재가 될 수 없습니다. 연결 상태를 현재에서 휴지로 또는 휴지에서 현재로 변경하는 것은 보류 또는 해제 지연 중 상태에 영향을 미치지 않습니다.

연결이 RELEASE문에 의해 해제 지연 중 상태에 놓입니다. 활성 그룹이 확약 조작을 실행할 때 모든 해제 지연 중 활성 그룹 연결은 종료됩니다. 연결 상태를 보류에서 해제 지연 중으로 변경하는 것은 현재 또는 휴지 상태에 영향을 미치지 않습니다. 따라서, 해제 지연 중 상태의 연결은 다음 확약 조작까지 사용할 수 있습니다. 연결 상태를 해제 지연 중에서 보류로 변경하는 방법은 없습니다.

활성 그룹 연결 상태

CONNECT문을 명시적으로 또는 내재적으로 실행함으로써 다른 서버를 설정할 수 있습니다. 다음 규칙이 적용됩니다.

- 활성 그룹은 동시에 같은 서버에 대하여 둘 이상의 연결을 가질 수 없습니다.
- 활성 그룹이 SET CONNECTION문을 실행할 때 지정된 위치명은 활성 그룹 연결 세트에서 기존 연결이어야 합니다.
- 활성 그룹이 CONNECT문을 실행할 때 지정된 서버명은 활성 그룹 연결 세트에서 기존 연결이 아니어야 합니다.

활성 그룹이 현재 연결을 갖는 경우, 활성 그룹은 *연결된* 상태입니다. CURRENT SERVER 특수 레지스터에는 현재 연결된 서버명이 들어 있습니다. 활성 그룹은 그 서버에 의해 관리되는 오브젝트를 참조하는 SQL문을 실행할 수 있습니다.

CONNECT 또는 SET CONNECTION문을 성공적으로 실행할 때 연결되지 않은 상태의 활성 그룹은 연결된 상태에 들어갑니다.

활성 그룹이 현재 연결을 갖지 않는 경우, 활성 그룹은 *연결되지 않은* 상태입니다. CURRENT SERVER 특수 레지스터 내용은 공백과 동일합니다. 실행될 수 있는 SQL문은 CONNECT, DISCONNECT, SET CONNECTION, RELEASE, COMMIT 및 ROLLBACK입니다.

현재 연결이 의도적으로 종료되거나 SQL문 실행이 현재 서버에서의 롤백 조작 및 연결 손실을 야기하는 실패로 인해 성공하지 못한 경우 연결된 상태의 활성 그룹이 연결

되지 않은 상태에 들어갑니다. 활성 그룹이 성공적으로 확약 조작을 실행하고 연결이 해제 지연 중 상태이거나 또는 어플리케이션 프로세스가 DISCONNECT문을 성공적으로 실행할 때 연결은 의도적으로 종료됩니다.

연결 종료 시

연결이 종료될 때 연결을 통한 활성 그룹에 의해 예약된 모든 자원 및 연결을 작성하고 유지보수 하는 데 사용되었던 자원은 모두 할당 해제됩니다. 예를 들어, 어플리케이션 프로세스 P가 서버와의 연결을 릴리스 지연 중 상태로 놓은 경우 다음 확약 조작시 연결이 종료될 때 X에서 P의 모든 커서는 닫히고 할당해제됩니다.

활성 그룹이 연결되지 않은 상태에 놓이는 경우의 통신 실패의 결과로 인해 연결이 종료될 수도 있습니다. 활성 그룹의 모든 연결은 활성 그룹이 종료될 때 종료됩니다.

자료 표시 고려사항

서로 다른 시스템은 서로 다른 방식으로 자료를 표시합니다. 한 시스템에서 다른 시스템으로 자료가 이동될 때 자료 변환이 수행되어야 합니다. DRDA를 지원하는 제품은 수신 시스템에서 필요한 변환을 자동으로 수행합니다.

숫자 자료의 경우 변환 수행에 필요한 정보는 자료 유형과 송신 시스템의 환경 유형입니다. 예를 들어 iSeries용 DB2 UDB 어플리케이션 리퀘스터의 부동 소수점 변수가 OS/390 서버에서 표의 열에 할당될 때 이 숫자는 IEEE 형식에서 System/370* 형식으로 변환됩니다.

문자 및 그래픽 자료의 경우 송신 시스템의 자료 유형 및 환경 유형은 충분하지 않습니다. 문자 및 그래픽 스트링을 변환하기 위해 추가 정보가 필요합니다. 스트링 변환은 코드화 자료 문자 세트 및 해당 자료로 수행될 조작에 따라 다릅니다. 스트링 변환은 IBM 문자 자료 표시 구조(CDRA)에 따라 수행됩니다. 문자 변환에 대한 자세한 정보는 *문자 자료 표시 구조 레벨 1 참조서,* SC09-1390을 참조하십시오.

문자 변환

스트링은 문자를 표시할 수 있는 일련의 바이트입니다. 하나의 스트링 내에서 문자는 모두 공통 코딩 표시로 표시됩니다. 이런 문자를 다른 코딩 표시로 변환해야 할 필요가 있는 경우도 있습니다. 변환하는 프로세스를 *문자 변환*이라고 합니다.[6]

문자 변환은 SQL문이 리모트로 실행될 때 발생할 수 있습니다. 예를 들면, 다음 두 가지 경우를 고려하십시오.

- 어플리케이션 리퀘스터로부터 현재 서버로 송신된 호스트 변수 값
- 현재 서버에서 어플리케이션 리퀘스터로 송신된 결과 열의 값

6. 필요한 경우 문자 변환은 자동으로 이루어지며, 성공할 때 어플리케이션에 투명합니다. 그러므로 명령문 실행에 관련된 스트링이 모두 같은 방식으로 표현되는 경우 변환에 관한 지식은 필요하지 않습니다. 따라서, 여러 독자들의 경우 문자 변환은 적절하지 않습니다.

두 가지 경우에서 스트링은 송신 및 수신 시스템에서 서로 다른 표시를 가질 수 있습니다. 변환은 동일한 시스템에서 스트링 조작시에 발생할 수도 있습니다.

다음 리스트는 문자 변환을 설명할 때 사용된 몇 가지 용어를 정의합니다.

문자 세트
정의된 문자 세트. 예를 들면, 다음 문자 세트는 몇 개 코드 페이지에 나타납니다.
- 26개의 악센트가 없는 문자 A - Z
- 26개의 악센트가 없는 문자 a - z
- 숫자 0 - 9
- . , : ; ? ( ) ’ ″ / - _ (밑줄) & + % * = < >

코드 페이지
코드점에 대한 문자의 할당 세트 예를 들어, EBCDIC에서 "A"는 코드점 X'C1'에 지정되고 "B"는 코드점 X'C2'에 지정됩니다. 한 코드 페이지 내에서 각 코드점은 오직 하나의 특정 의미를 갖습니다.

코드점
문자를 표시하는 고유한 비트 패턴

코드화 문자 세트
문자 세트 및 세트의 문자와 코드 표시간의 일대일 관계를 설정하는 명백한 규칙 세트

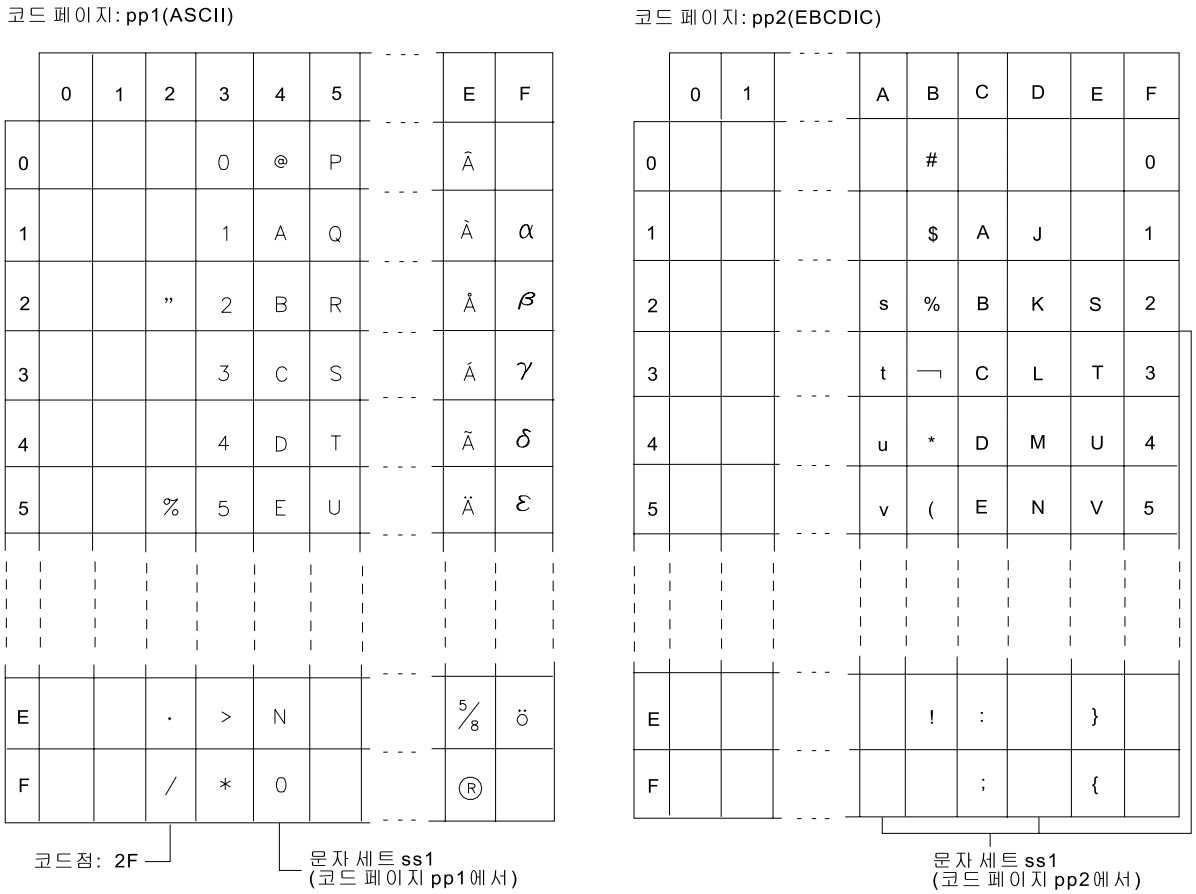
코드화 체계
문자 자료를 표시하는 데 사용되는 규칙 세트. 예를 들면, 다음과 같습니다.
- 단일 바이트 EBCDIC
- 단일 바이트 ASCII[7]
- 혼합 1 및 2바이트 EBCDIC
- 혼합 1 및 2바이트 ASCII
- UCS-2(범용 코드화 문자 세트)

대체 문자
목표 코딩 표시에서 일치되지 않는 소스 코딩 표시의 임의 문자에 대한 문자 변환 동안 대체되는 고유 문자.

문자 세트 및 코드 페이지

다음 예는 일반적인 문자 세트가 두 개의 다른 코드 페이지의 다른 코드점에 맵핑되는 방법을 나타냅니다.

7. ASCII는 이 책에서 IBM-PC 자료 또는 ISO 8 자료를 나타냅니다.



동일한 코드화 체계의 경우 조차 다른 코드화 문자 세트가 많이 있으며, 동일한 코드점은 다른 코드화 문자 세트의 다른 문자를 표시할 수 있습니다. 또한 문자 스트링의 바이트는 1바이트 문자 세트(SBCS)로부터 문자를 반드시 표시할 필요는 없습니다. 문자 스트링은 혼합 자료(1바이트 문자 및 2바이트 문자의 혼합)에 대해서도 사용되며, 어떤 문자 세트(비트 문자라고 함)와 연관되지 않은 자료에 대해서도 사용됩니다. 이것은 그래픽 스트링에 해당되지 않습니다. 데이터베이스 관리자는 모든 그래픽 스트링의 모든 바이트 쌍이 2바이트 문자 세트(DBCS) 또는 범용 코드화 문자 세트(UCS-2)로부터 문자를 표시한다고 가정합니다.

고유 코드화 체계에서 코드화 문자 세트 ID(CCSID)는 자료가 해당 사이트에 저장될 수 있는 코드화 문자 세트 중 하나입니다. 외부 코드화 체계에서 코드화 문자 세트 ID(CCSID)는 자료가 해당 사이트에 저장될 수 없는 코드화 문자 세트입니다. 예를 들어, iSeries용 DB2 UDB는 EBCDIC 코드화 체계로 코드화 문자 세트 ID(CCSID)에 자료를 저장할 수 있지만 ASCII 코드화 체계로는 저장할 수 없습니다.

외부 코드화 체계에서 자료가 들어 있는 호스트 변수는 호스트 변수가 함수 또는 선택 리스트에서 사용될 때 고유 코드화 체계에서 코드화 문자 세트 ID(CCSID)로 변환됩니다. 외부 코드화 체계에서 자료가 들어 있는 호스트 변수는 비교 또는 스트링을 조합하는 조작에서 사용될 때 고유 코드화 체계에서 코드화 문자 세트 ID(CCSID)로 변환됩

니다. 고유 코드화 체계에서 자료가 어느 코드화 문자 세트 ID(CCSID)로 변환되는가는 외부 코드화 문자 세트 ID(CCSID) 및 디폴트 코드화 문자 세트 ID(CCSID)를 근거로 합니다.

코드화 문자 세트 및 CCSID

IBM의 문자 자료 표시 구조(CDRA)에서는 스트링 표시 및 암호화의 차이점을 다룹니다. 코드화 문자 세트 ID(CCSID)는 이 구조의 주요 요소입니다. CCSID는 코드화 체계 및 한 쌍 이상의 문자 세트 및 코드 페이지를 고유하게 식별하는 2바이트(서명되지 않은) 2진 수입니다.

CCSID는 스트링 속성이며, 길이는 스트링 속성과 같습니다. 동일한 스트링 열 값은 모두 동일한 CCSID를 갖습니다.

각 데이터베이스 관리자에서 문자 변환은 *CCSID 변환 선택 표* 사용을 수반합니다. 변환 선택 표에는 유효한 소스 및 목표 조합 리스트가 들어 있습니다. 각 CCSID 쌍에 대해, 변환 선택 표는 한 코드화 문자 세트에서 다른 코드화 문자 세트로 변환을 수행하는 데 사용된 정보를 포함합니다. 이 정보는 변환이 필요한지 여부에 대한 표시를 포함합니다(일부 경우 관련된 스트링이 다른 CCSID를 갖고 있다 하더라도 변환이 필요 없습니다).

디폴트 CCSID

모든 서버와 어플리케이션 리퀘스터는 하나의 디폴트 CCSID 또는 DBCS 자료를 지원하는 설치에서의 복수 디폴트 CCSID를 갖습니다. 다음 스트링 유형의 CCSID는 현재 서버에서 판별됩니다.

- 소스의 CCSID가 외부 코드화 체계로 되어 있을 경우 스트링 상수(날짜시간 값을 표시하는 스트링 상수 포함)
- 스트링 값이 있는 특수 특수 레지스터(예: USER 및 CURRENT SERVER)
- CAST, CHAR, DIGITS 및 HEX 스칼라 함수의 결과[8]
- CCSID가 인수로 지정되지 않을 때 VARCHAR, GRAPHIC 및 VARGRAPHIC 스칼라 함수의 결과
- CCSID가 인수로서 지정되지 않을 때 CLOB 및 DBCLOB 스칼라 함수의 결과
- 명시적 CCSID가 열에 대해 지정되지 않을 때 CREATE TABLE 또는 ALTER TABLE문에 의해 정의된 스트링 열[9]

분산 SQL 프로그램에서 호스트 변수의 디폴트 CCSID는 어플리케이션 리퀘스터에 의해 판별됩니다. 비분산 SQL 프로그램에서 호스트 변수의 디폴트 CCSID는 서버에 의

8. 디폴트 CCSID가 65535이고 함수가 CLOB 또는 DBCLOB에 대한 CAST인 경우 사용되는 CCSID는 DFTCCSID 작업 속성 값이 됩니다.

9. 디폴트 CCSID가 65535이면 문자 스트링 열은 65535를 사용하지 않습니다. 대신, 사용된 CCSID가 DFTCCSID 작업 속성 값이 됩니다.

해 판별됩니다. OS/400에서 디폴트 CCSID는 CCSID 작업 속성에 의해 판별됩니다. CCSID에 대한 자세한 정보는 iSeries Information Center의 국제화 섹션의 CCSID로 작업 주제를 참조하십시오.

정렬 순서

정렬 순서는 문자 세트의 문자가 비교되고 순서화될 때 각각 어떻게 연관되는지를 정의합니다. 서로 다른 정렬 순서는 특정 언어에 대해 자료가 정렬되기를 원하는 사용자에게 유용합니다. 예를 들어, 리스트는 특정 언어에 대해 정상적으로 보여진 대로 정렬될 수 있습니다. 정렬 순서는 어떤 문자들을 동등하게 취급하는 데 사용될 수도 있는데, 예를 들면 **a** 및 **A**)입니다. 정렬 순서는 다음을 포함하는 모든 비교에서 작동합니다.

- SBCS 문자 자료(비트 자료 포함)
- 혼합된 자료의 SBCS 부분
- UCS-2 그래픽 자료

SBCS 정렬 순서 지원은 256 바이트 표를 사용하여 구현됩니다. 표에서 각 바이트는 SBCS 코드 페이지의 코드점 또는 문자에 해당합니다. 정렬 순서는 문자 자료에 적용 가능하기 때문에 CCSID는 표와 연관되어야 합니다. 정렬 순서 표의 바이트는 각 코드점이 해당 코드 페이지의 다른 코드점에 비교되는 방식을 근거로 설정됩니다. 예를 들어, 문자 **a** 및 **A**가 비교될 때 동등하다고 취급되는 경우 코드점에 대한 정렬 순서 표의 바이트는 동일한 값 또는 가중치를 포함합니다.

UCS-2 정렬 순서 지원은 복수 바이트 표를 사용하여 구현됩니다. 표 내의 바이트 쌍은 UCS-2 코드 페이지의 문자에 해당합니다. UCS-2에서 수천개 문자의 서브세트만이 일반적으로 표에 표시됩니다. 다르게 비교할 문자(동일한 감시를 받는 다른 문자)만이 표에 표시됩니다. 정렬 순서 표의 바이트는 각 문자가 UCS-2의 다른 문자와 비교되는 방법을 근거로 설정됩니다.

정렬 순서 표에서 2바이트 이상(또는 UCS-2에 대한 바이트 쌍)이 동일한 값을 가질 때 정렬 순서는 공유 가중치 정렬 순서(shared-weight sort sequence)입니다. 정렬 순서 표에서 모든 바이트(또는 UCS-2에 대한 바이트 쌍)가 고유 값을 가질 때 정렬 순서는 고유 가중치 정렬 순서(unique-weight sort sequence)입니다. 많은 언어에서 고유 및 공유 가중치 정렬 순서가 오퍼레이팅 시스템의 일부로 시스템에 탑재되어 제공됩니다. 다른 언어에 대한 정렬 순서가 필요한 경우 CRTTBL(Creat Table) 명령을 사용하여 정렬 순서를 정의하십시오.

자료 자체는 정렬 순서에 의해 변경되지 않는다는 점을 기억해야 합니다. 자료에 대한 가중치 표시는 비교를 위해서 사용됩니다. SQL에서 CRTSQLxxx, STRSQL 및 RUNSQLSTM 명령에서 정렬 순서가 지정됩니다. SET OPTION문을 사용하여 삽입 SQL이 들어 있는 프로그램 소스 내에서 정렬 순서를 지정할 수 있습니다. 정렬 순서는 SQL문에서 수행된 모든 문자 비교에 적용됩니다. 시스템에서의 디폴트 정렬 순서는 문

자의 16진수 표시가 사용될 때 발생하는 내부 순서입니다. 이것은 SRTSEQ(*HEX)를 지정할 때 사용할 수 있는 순서입니다. 버전 2 릴리스 3 이전의 제품 릴리스로 사전컴파일된 프로그램의 경우 정렬 순서는 *HEX입니다.

정렬 순서는 FOR BIT DATA 또는 BLOB 열에 적용되지 않습니다.

CCSID에 대한 자세한 정보는 iSeries Information Center의 국제화 섹션의 CCSID로 작업 주제를 참조하십시오. 정렬 순서 및 출하시 시스템에서 제공하는 순서에 대한 자세한 정보는 iSeries Information Center의 정렬 순서 표 주제를 참조하십시오.

권한부여 및 권한

사용자는 지정된 함수를 수행하기 위한 권한부여를 가질 때만 SQL문을 성공적으로 실행할 수 있습니다. 표를 작성하려면 사용자는 표를 작성하기 위한 권한을 부여받아야 합니다. 표를 제거하려면 사용자는 표를 제거하기 위한 권한을 부여받아야 합니다.

관리 권한을 갖고 있는 사람들은 데이터베이스 관리자를 제어하는 타스크에 대해 책임을 지며, 자료 안전성 및 무결성에 대한 책임을 집니다. 관리 권한을 갖는 사람들은 데이터베이스 관리자에 대한 액세스 및 해당 액세스 확장을 하는 사람들을 제어합니다. 관리 권한을 갖는 사람들은 특정 권한을 부여받았는지에 상관없이 모든 오브젝트에 대해 모든 조작을 수행할 권한을 갖습니다. 보안 관리자 및 *ALLOBJ 권한이 있는 모든 사용자는 관리 권한을 갖습니다.

*권한*은 관리 권한이 사용자가 수행할 수 있도록 허용한 활동입니다. 권한을 부여받은 사용자는 어떠한 오브젝트도 작성할 수 있으며, 소유한 오브젝트에 액세스할 수 있고, GRANT문을 사용하여 다른 사용자에게 고유 오브젝트에 대한 권한을 전달할 수 있습니다. REVOKE문은 이전에 부여받은 권한을 호출하는 데 사용될 수 있습니다.

오브젝트가 작성될 때 하나의 권한부여 ID가 오브젝트에 대한 *소유권*을 지정받습니다. 소유권은 사용자에게 오브젝트를 제거할 권한을 포함하여 오브젝트에 대한 완전한 제어를 부여합니다. 소유자는 자신이 갖고 있는 오브젝트에 대한 권한을 스스로 호출할 수 있습니다. 이때 소유자는 해당 권한이 필요로 하는 조작을 일시적으로 수행하지 못할 수 있습니다. 그러나, 자신이 소유자이기 때문에 항상 권한을 자신에게 다시 부여할 수 있습니다.

SQL 오브젝트에서 *PUBLIC에 부여된 권한은 오브젝트 작성시에 사용된 명명 규칙에 따라 다릅니다. *SYS 명명 규칙이 사용된 경우 *PUBLIC은 오브젝트가 작성되었던 라이브러리에 대한 작성 권한(CRTAUT)을 예약합니다. *SQL 명명 규칙이 사용된 경우 *PUBLIC은 *EXCLUDE 권한을 예약합니다.

이 책의 권한부여 섹션에서 오브젝트 소유자는 오브젝트가 초기에 작성되었기 때문에 해당 오브젝트로부터 호출된 권한을 갖지 않았다고 가정합니다. 오브젝트가 뷰인 경우 뷰의 소유자는 이 뷰가 직접적으로 또는 간접적으로 종속되어 있는 표 또는 뷰로부터 호

출된 시스템 권한 *READ를 갖지 않았다고 가정합니다. 소유자는 뷰 정의에서 참조된 모든 표 및 뷰에 대해 시스템 권한 *READ를 가지며, 뷰가 참조된 경우 정의에서 참조된 모든 표 및 뷰에 대해 시스템 권한 *READ를 갖습니다. 권한에 대한 자세한 정보는 iSeries 보안 참조서  를 참조하십시오.

기억장치 구조

iSeries 시스템은 오브젝트 기반의 시스템입니다. iSeries용 DB2 UDB의 모든 데이터베이스 오브젝트(예를 들어, 표 및 색인)는 OS/400의 오브젝트입니다. 단일 레벨 기억장치 관리자는 데이터베이스 오브젝트의 모든 기억장치를 관리하므로 데이터베이스 특정 기억장치 구조(예를 들어, 표 공간)는 필요없습니다.

분할된 또는 분배된 표는 자료가 다른 데이터베이스 분할에 걸쳐 퍼질 수 있도록 합니다. 포함된 분할은 표가 작성되거나 변경될 때 지정된 노드 그룹에 의해 판별됩니다. 노드 그룹은 하나 이상의 iSeries 시스템 그룹입니다. 분할 맵은 각 노드 그룹과 관련되어 있습니다. 분할 맵은 데이터베이스 관리자가 노드 그룹의 어떤 시스템이 주어진 자료 열을 저장할 것인지를 판별하는 데 사용됩니다. 노드 그룹 및 자료 분할에 대한 자세한 정보는 DB2 Multisystem 책을 참조하십시오.

표는 외부 파일에 저장되는 자료에 대한 링크를 레지스터하는 열을 포함합니다. 이에 대한 메카니즘은 DataLink 자료 유형입니다. 정규 표에 작성되는 자료 링크 값은 외부 파일 서버에 저장되는 파일을 가리킵니다.

파일 서버에서 DB2 파일 관리자는 DB2와 결합하여 다음 선택적 함수를 제공합니다.

- DB2에 현재 링크된 파일이 삭제 또는 재명명되지 않도록 하는 참조 무결성.
- DataLink 열에 대해 적절한 SQL 권한을 갖는 사람들만 해당 열에 링크된 파일을 읽을 수 있도록 하는 보안.

DataLinker는 다음 함수로 이루어집니다.

DataLinks 파일 관리자

DB2에 링크된 특정 파일 서버의 모든 파일을 등록합니다.

DataLinks 필터

등록된 파일이 삭제 또는 재명명되지 않도록 하기 위해 파일 시스템 명령을 필터링합니다. 선택적으로, 적절한 액세스 권한이 있음을 보장하기 위해 명령을 필터링합니다.

제 2 장 언어 요소

이 장은 많은 SQL문에 공통인 SQL의 기본 구문과 언어 요소를 정의합니다.

자세한 내용은 다음을 참조하십시오.

문자

SQL 언어로 된 키워드와 연산자의 기본 기호는 IBM 관계형 데이터베이스 제품에서 지원되는 1바이트 문자[10]입니다. 언어 문자는 문자, 숫자 또는 특수 문자로 분류됩니다.

*문자*는 영문자 중 26자의 대문자(A - Z)와 26자의 소문자(a - z)입니다.[11]

*숫자*는 0 - 9의 문자입니다.

*특수 문자*는 아래에 나열된 모든 문자입니다.[12]

	간격	-	빼기 부호
"	인용 부호 또는 큰 따옴표	.	마침표
%	퍼센트	/	슬래시(/)
&	앰퍼샌드(&)	:	콜론(:)
'	작은 따옴표	;	세미콜론
(	왼쪽 괄호	<	보다 작음
)	오른쪽 괄호	=	등호
*	별표(*)	>	보다 큼
+	더하기 부호	?	의문부호
,	쉼표	_	밑줄
| [13]	세로줄		

토큰

언어의 기본 구문 단위를 *토큰*이라고 합니다. 토큰은 공백, 제어 문자 및 스트링 상수나 분리 ID 안의 문자를 제외하고 하나 이상의 문자로 구성됩니다(이 용어는 나중에 정의됩니다).

토큰은 *일반* 또는 *분리 문자* 토큰으로 분류됩니다.

- *일반* 토큰은 숫자 상수, 일반 ID, 호스트 ID 또는 키워드입니다.
- *분리 문자* 토큰은 스트링 상수, 분리 ID, 연산자 기호 또는 구문 도표에 표시되는 특수 문자입니다. 의문 부호(?)는 또한 728 페이지의 『PREPARE』에 설명된 대로 매개변수 마커로 사용될 때 분리 문자 토큰입니다.

간격: *간격*은 일련의 공백 문자입니다.

---

10. SQL문이 UCS-2로서 코드화되어 있으면 스트링 상수를 제외한 모든 문자가 처리되기 전에 1바이트 문자로 변환됩니다. 스트링 상수를 나타내는 토큰은 1바이트로 변환되지 않고 UCS-2 그래픽 스트링으로 처리될 수 있습니다.

11. 문자에는 자국어의 영문 확장자로 예약된 세 개의 코드점도 포함됩니다(미국에서는 #, @ 및 $). 이 세 개의 코드점은 CCSID에 따라 다른 문자를 나타내므로 사용하지 않는 것이 좋습니다.

12. 부정 기호(¬) 및 느낌표(!)도 iSeries용 DB2 UDB에서 사용되는 특수 문자입니다. 특수 문자는 변하는 문자이기 때문에 사용을 피해야 합니다.

13. 세로줄(|) 문자를 사용하면 IBM 관계형 데이터베이스 제품 사이에서 코드를 이식할 수 없기도 합니다. 연결 연산자 대신 CONCAT 연산자(||)를 사용하는 것이 좋습니다. 세로줄은 변하는 문자이므로 사용하지 않는 것이 좋습니다.

제어 문자: *제어 문자*는 스트링 정렬에 사용되는 특수 문자입니다. 다음 표에는 데이터베이스 관리자에 의해 핸들되는 제어 문자가 들어 있습니다.

표 1. 제어 문자

제어 문자	EBCDIC 16진 값	UCS-2 16진 값
탭	05	0009
용지 넘김	0C	000C
캐리지 리턴	0D	000D
개행	15	0085
행 진입(개행)	25	000A

스트링 상수와 특정한 분리 ID가 아닌 토큰에는 제어 문자나 간격이 없어야 합니다. 토큰 뒤에 제어 문자나 간격이 올 수 있습니다. 모든 일반 토큰 뒤에 분리 문자 토큰, 제어 문자 또는 간격이 *반드시* 와야 합니다. 구문에서 일반 토큰 뒤에 분리 문자 토큰이 올 수 없으면 그 일반 토큰 뒤에 제어 문자나 간격이 와야 합니다. 다음 예는 이 단락에서 설명한 규칙을 나타냅니다.

일반 토큰의 예입니다.

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

실제 토큰을 변경하는 위의 일반 토큰 조합의 예입니다.

```
1.1       .1+2       SELECTE       .1E       E3       SELECT1
```

일반 토큰 뒤에 분리 문자나 간격이 와야 하는 이유를 나타냅니다.

분리 문자 토큰의 예입니다.

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

실제 토큰을 변경하는 위의 일반 토큰과 분리 문자 토큰 결합의 예입니다.

```
1.       .3
```

마침표(.)는 이름 규정화에서 분리자로 사용될 때 분리 문자 토큰입니다. 여기서 점은 일반 토큰인 숫자 상수와 결합하여 사용됩니다. 이와 같이 구문상 일반 토큰 뒤에 분리 문자 토큰이 올 수 없습니다. 대신, 일반 토큰 뒤에 간격을 두어야 합니다.

102 페이지의『소수점』에서 설명된 대로 소수점이 쉼표로 정의된 경우 쉼표는 숫자 상수에서 소수점으로 해석됩니다. 이런 숫자 상수의 예입니다.

```
1,2       ,1       1,       1,e1
```

'1,2'와 '1,e1'이 두 항목을 의미하면 일반 토큰(1)과 분리 문자 토큰(,) 둘 다 뒤에 간격을 두어야 쉼표가 소수점으로 해석되지 않습니다. 쉼표가 분리 문자 토큰이더라도 소수점으로 해석되면 숫자의 일부입니다. 따라서 구문상 일반 토큰(1) 뒤에 분리 문자 토큰(,)이 올 수 없습니다. 대신, 일반 토큰 뒤에 간격을 두어야 합니다.

주석: 정적 SQL문에 호스트 언어 주석이나 SQL문이 포함될 수 있습니다. 동적 SQL 문은 SQL 주석을 포함할 수 있습니다. 분리 문자 토큰 안이나 키워드 EXEC와 SQL 사이를 제외하고 간격을 둘 수 있는 곳에는 주석 유형 모두를 지정할 수 있습니다. SQL 주석에는 두 가지 유형이 있습니다.

단순 주석

간단한 주석은 두 개의 연속 하이픈(--)으로 시작합니다. 간단한 주석은 행 끝을 지나 계속될 수 없습니다. 자세한 정보는 370 페이지의 『SQL 주석』을 참조하십시오.

브라켓 주석

브라켓 주석은 /*로 시작되고 */로 끝납니다. 브라켓 있는 주석은 행 끝을 지나 계속될 수 있습니다. 자세한 정보는 370 페이지의 『SQL 주석』을 참조하십시오.

대문자와 소문자: C 호스트 변수 이외의 일반 토큰에 사용되는 소문자는 대문자로 처리(fold)됩니다. 분리 문자는 대문자로 처리되지 않습니다. 따라서, 명령문은 다음과 같습니다.

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

위의 명령문은 다음과 동일합니다.

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

ID

*ID*는 이름을 만들 때 사용되는 토큰입니다. SQL문의 식별자는 다음 유형 중 하나입니다.

- 『SQL ID』
- 『시스템 ID』
- 46 페이지의『호스트 ID』

주: $, @, # 및 다른 모든 변하는 문자는 포함하는 스트링의 CCID에 따라 표시하는 데 사용되는 코드점이 달라지므로 ID에 사용하지 말아야 합니다. 사용되면 예측하지 않은 결과가 발생할 수 있습니다. 변형 문자에 대한 자세한 정보는 iSeries Information Center의 변형 문자 주제를 참조하십시오.

SQL ID

두 가지 유형의 SQL ID가 있습니다(*일반 ID*와 *분리 ID*).

- *일반 ID*는 대문자이며 뒤에 0이나 문자가 오는데, 각각 대문자, 숫자 또는 밑줄 문자입니다. 일반 ID는 대문자로 변환됨에 유의하십시오. 예약어는 일반 ID가 될 수 없습니다. 예약어 리스트는 897 페이지의 부록 D『예약어』를 참조하십시오. 예약어가 SQL에서 ID로 사용되면 대문자로 지정하고 SQL 이탈 문자로 묶어야 합니다.
- *분리 ID*는 SQL 이탈 문자로 묶인 일련의 문자입니다. 순서열은 하나 이상의 문자로 구성되어야 하며, 앞의 공백은 유효하나 뒤의 공백은 유효하지 않습니다. 분리 ID의 길이에는 두 개의 SQL 이탈 문자가 포함되지 않습니다. 분리 ID는 대문자로 변환되지 않음에 유의하십시오. 이탈 문자는 작은 따옴표(')인 다음 경우를 제외하고 인용부호(")입니다.
  - SQL 스트링 분리 문자가 COBOL 구문 검사 명령문 모드에서 인용 부호로 설정되는 경우의 대화식 SQL
  - CRTSQLCBL 또는 CRTSQLCBLI 매개변수 OPTION(*QUOTESQL)이 스트링 분리 문자를 인용 부호(")로 지정할 때 COBOL 프로그램의 동적 SQL
  - CRTSQLCBL 또는 CRTSQLCBLI 매개변수 OPTION(*QUOTESQL)이 스트링 분리 문자를 인용 부호(")로 지정할 때 COBOL 어플리케이션 프로그램

  다음 문자는 분리 ID 안에 사용할 수 없습니다.

  - X'00' - X'3F' 및 X'FF'

시스템 ID

시스템 ID는 OS/400의 시스템 오브젝트명에 사용됩니다. 일반 ID와 분리 ID, 이 두 가지 유형의 시스템 ID가 있습니다.

- 시스템 일반 ID를 만드는 규칙은 SQL 일반 ID를 만드는 규칙과 동일합니다.

- 시스템 분리 ID를 만드는 규칙은 다음 경우를 제외하고 SQL 분리 ID를 만드는 규칙과 동일합니다,
  - 다음 특수 문자는 분리 시스템 ID에 사용할 수 없습니다.
    - 공백(X'40')
    - 별표(*)(X'5C')
    - 작은 따옴표(X'7D')
    - 의문 부호(X'6F')
    - 인용 부호(X'7F')
  - 분리 문자 내의 문자가 일반 ID를 형성하지 않는 한 이탈문자에 필요한 바이트는 ID의 길이에 포함됩니다.

    예를 들어, “PRIVILEGES”는 대문자이고 분리 문자 안의 문자가 일반 ID를 만들면 길이가 10바이트이므로 열에 대해 유효한 시스템명입니다. 반면에 “privileges”는 소문자이고 길이가 12바이트이면 분리 문자에 필요한 바이트 수가 ID 길이에 포함되어야 하므로 열에 대해 유효한 시스템명이 아닙니다.

예

```
WKLYSAL     WKLY_SAL     "WKLY_SAL"     "UNION"     "wkly_sal"
```

호스트 ID

*host-identifier*는 호스트 프로그램에서 선언된 이름입니다. 호스트 ID 작성 규칙은 호스트 언어 규칙이나 예외로 DBCS 문자에는 사용될 수 없습니다. 예를 들어, COBOL 프로그램의 호스트 ID 작성 규칙은 COBOL의 사용자 정의 단어 작성 규칙과 같습니다. 사전컴파일러가 'SQ'[14], 'SQL', ’sql', 'RDI' 또는 'DSN' 문자로 시작되는 호스트 변수를 생성하므로 이 문자들로 시작되는 이름은 사용할 수 없습니다.

14. 'SQ'는 C, COBOL 및 PL/I에서 허용되므로 RPG에서는 사용할 수 없습니다.

명명 규칙

이름 작성 규칙은 이름과 명명 옵션(*SQL 또는 *SYS)에 의해 지정된 오브젝트 유형에 따라 달라집니다. 명명 옵션은 CRTSQLxxx, RUNSQLSTM 및 STRSQL 명령에 지정됩니다. SET OPTION문을 사용하여 삽입 SQL이 들어 있는 프로그램의 소스 안에 명명 옵션을 지정할 수 있습니다. 구문 도표에서는 다른 이름 유형에 대해 다른 용어를 사용합니다. 다음 리스트는 이런 용어를 정의합니다.

별명

별명을 지정하는 규정 또는 규정되지 않은 이름. 별명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL 명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

별명은 별명이나 별명의 시스템 오브젝트명을 지정할 수 있습니다.

권한부여명

사용자나 사용자 그룹을 지정하는 시스템 ID. 권한부여명은 서버에서의 사용자 프로파일명입니다. 소문자나 특수 문자가 포함되는 분리 ID는 사용할 수 없습니다. 권한부여명과 권한부여 ID 사이의 차이를 알려면 58 페이지의 『권한부여 ID 및 권한부여명』을 참조하십시오.

열 이름

표나 뷰의 열을 지정하는 규정된 또는 규정되지 않은 이름. 열 이름의 규정되지 않은 형식은 SQL ID입니다. 규정된 형식은 규정자, 마침표(.), SQL ID 순입니다. 규정자는 표 이름, 뷰 이름 또는 상관명입니다.

열 이름은 COMMENT 및 LABEL 명령문을 제외하고 *schema-name*/ *table-name*.*column-name* 형식의 시스템명과 함께 규정될 수 없습니다. 열 이름을 규정하여야 하고 명령문에 상관명이 허용되면 열 규정에 상관명이 사용되어야 합니다.

열 이름은 표 또는 뷰의 열 이름이나 시스템 열 이름을 지정할 수 있습니다. 열 이름이 구분되는 경우 이름 길이를 판별할 때 분리 문자도 이름의 일부로 간주됩니다.

제한조건명

표에 대한 제한조건을 지정하는 규정된 또는 규정되지 않은 이름. 제한조건의 규정된 형식은 명명 옵션에 따

라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), 시스템 ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

콜렉션명 표나 뷰의 표, 뷰 또는 각 행을 지정하는 SQL ID

커서명 SQL 커서를 지정하는 SQL ID

설명자명 다음에 콜론이 오는 SQLDA를 지정하는 호스트 ID 호스트 ID에 대한 자세한 설명은 114 페이지의 『호스트 변수에 대한 참조』를 참조하십시오. SQLDA를 지정하는 호스트 변수에는 인디케이터 변수가 없어야 합니다. 형식 *:host-variable:indicator-variable*이 허용되지 않습니다.

distinct-type-name 고유한 유형을 지정하는 규정된 또는 규정되지 않은 이름. 고유한 유형 이름의 규정된 양식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

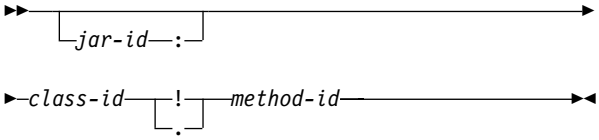
시스템명을 지정하는 경우 고유한 유형 이름이 SQL 루틴의 매개변수 자료 유형에 사용되거나 SQL 함수, SQL 프로시듀어 또는 트리거의 SQL 변수 선언에 사용될 때 규정될 수 없습니다.

외부 프로그램명 규정된 이름, 규정되지 않은 이름 또는 외부 프로그램을 지정하는 문자 스트링. 외부 프로그램명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), 시스템 ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), 시스템 ID 순입니다.

규정되지 않은 형식은 시스템 ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

문자 스트링 형식은 다음 중 하나입니다.

- OS/400 규정된 프로그램명('라이브러리명/ 프로그램명').
- OS/400 규정된 소스 파일명, 왼쪽 괄호, OS/400 멤버명, 오른쪽 괄호 순입니다('라이브러리명/소스파일명(멤버명)'). 이 형식은 REXX 프로시듀어를 호출하는 경우에만 유효합니다.
- OS/400 규정된 서비스 프로그램명, 왼쪽 괄호, OS/400 입력점명, 오른쪽 괄호('라이브러리명/서비스 프로그램명(입력점명)'). 이 형식은 함수에 대해서만 유효합니다 .
- 선택적 jar-id, 클래스 ID, 느낌표 또는 마침표, 메소드 ID('class-id!method-id' 또는 'class-id.method-id') 순서.



  jar-id는 데이터베이스에 설치되었을 때 jar 스키마를 식별합니다. 단순한 ID일 수도 있고 스키마 규정 ID일 수도 있습니다. 예를 들면 'myJar' 또는 'myCollection.myJar'입니다.
  class-id는 Java 오브젝트의 클래스 ID를 식별합니다. 클래스가 패키지의 일부인 경우 클래스 ID는 전체 패키지 접두부를 포함해야 합니다. 예를 들어, 클래스 ID가 'myPackage.StoredProcs'이면 JVM(Java Virtual Machine)은 StoredProcs 클래스에 대한 다음 디렉토리에서 찾을 수 있습니다.

```
'/QIBM/UserData/OS400/SQLLib/
  Function/myPackage/StoredProcs/'
```

  method-id는 호출될 Java 오브젝트의 메소드명을 식별합니다.

  이 형식은 Java 프로시듀어와 Java 함수에서만 유효합니다.

함수명

사용자 정의 함수를 지정하는 규정된 또는 규정되지 않은 이름, 고유한 유형을 작성할 때 생성된 캐스트 함수 또는 내장 함수. 함수명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형

식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템 명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

시스템명을 지정하는 경우 함수명은 이름이 CREATE, COMMENT, DROP, GRANT 또는 REVOKE 명령문에 사용될 때 스키마명/함수명의 형식으로만 규정될 수 있습니다.

호스트 레이블

호스트 프로그램에서 레이블을 지정하는 토큰

호스트 변수

호스트 변수를 지정하는 일련의 토큰. 호스트 변수에는 114 페이지의 『호스트 변수에 대한 참조』에서 설명된 대로 최소한 하나의 호스트 ID가 있습니다.

색인명

색인을 지정하는 규정된 또는 규정되지 않은 이름. 색인명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

노드 그룹명

노드 그룹을 지정하는 규정된 또는 규정되지 않은 이름. 노드 그룹은 표가 분배되는 iSeries 서버 그룹입니다. 분배된 표와 노드그룹에 대한 자세한 정보는 DB2 Multisystem 책을 참조하십시오.

노드 그룹명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), 시스템 ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), 시스템 ID 순입니다.

규정되지 않은 형식은 시스템 ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

패키지명

패키지를 지정하는 규정된 또는 규정되지 않은 이름. 패키지명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표

(.), 시스템 ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), 시스템 ID 순입니다.

규정되지 않은 형식은 시스템 ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

매개변수명

함수나 프로시듀어에 대한 매개변수를 지정하는 일반 ID. 프로시듀어에 대한 매개변수인 경우 ID 앞에 콜론이 올 수 있습니다.

프로시듀어명

프로시듀어를 지정하는 규정된 또는 규정되지 않은 이름. 프로시듀어의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

savepoint-name

저장점을 나타내는 규정되지 않은 ID.

스키마명

SQL 오브젝트에 논리 그룹을 제공하는 규정된 또는 규정되지 않은 이름. 스키마명은 표, 뷰, 색인, 프로시듀어, 함수, 트리거, 제한사항, 별명, 유형 또는 패키지의 이름의 규정자로 사용됩니다. 스키마명의 규정되지 않은 양식이 시스템 ID입니다. 스키마명의 규정된 양식은 명명 옵션에 따라 달라집니다.

SQL명의 경우 SQL문에서 규정되지 않은 스키마명은 서버명에 의해 내재적으로 규정됩니다. 규정된 형식은 서버명 뒤에 마침표(.)가 오고, 시스템 ID 순입니다. 서버명은 현재 서버를 식별하여야 합니다.

시스템명의 경우 SQL문에서 규정되지 않은 스키마명은 서버명에 의해 내재적으로 규정됩니다. 규정된 형식은 서버명, 슬래시(/), 시스템 ID 순입니다. 서버명은 현재 서버를 식별하여야 합니다.

주: 스키마명은 CREATE SCHEMA 명령문에 의해 작성된 스키마나 OS/400 라이브러리를 말합니다.

서버명

서버를 나타내는 SQL ID. ID에는 소문자나 특수 문자가 포함되어야 합니다.

특정명

프로시듀어나 함수를 고유하게 식별하는 규정된 또는 규정되지 않은 이름. 특정명의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

SQL 레이블

SQL 프로시듀어, SQL 함수 또는 트리거 본문에서 레이블을 지정하는 규정되지 않은 이름. SQL 레이블명은 SQL ID입니다.

SQL 매개변수명

SQL 루틴 본문에서 매개변수를 지정하는 규정된 또는 규정되지 않은 이름. SQL 매개변수의 규정되지 않은 형식은 SQL ID입니다. 규정된 형식은 프로시듀어명, 마침표(.), SQL ID 순입니다.

SQL 변수명

SQL 루틴 본문에서 변수를 지정하는 규정된 또는 규정되지 않은 이름. SQL 변수의 규정되지 않은 형식은 SQL ID입니다. 규정된 형식은 SQL 레이블, 마침표(.), SQL ID 순입니다.

명령문명

준비된 SQL문을 지정하는 SQL ID.

시스템 열 이름

표나 뷰의 OS/400열 이름을 지정하는 규정되지 않은 이름. 시스템 열 이름은 시스템 ID입니다. 시스템 열 이름은 분리 문자가 될 수 있지만 분리 문자 내의 문자는 소문자나 특수 문자가 포함될 수 없습니다.

시스템 오브젝트명

표, 뷰, 색인 또는 별명의 OS/400 이름을 지정하는 규정되지 않은 이름. 시스템 오브젝트명은 시스템 ID입니다.

표, 뷰, 색인 또는 별명의 규정되지 않은 이름이 유효한 시스템 ID이면 표, 뷰, 색인 또는 별명의 시스템 오브젝트명은 표, 뷰, 색인 또는 별명의 규정되지 않은 이름입니다.

표 이름

표를 지정하는 규정된 또는 규정되지 않은 이름. 표 이름의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

표 이름은 표 이름이나 표의 시스템 오브젝트명을 지정할 수 있습니다.

트리거명

표에 대한 트리거를 지정하는 규정된 또는 규정되지 않은 이름. 트리거의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), 시스템 ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

뷰 이름

뷰를 지정하는 규정된 또는 규정되지 않은 이름. 뷰 이름의 규정된 형식은 명명 옵션에 따라 달라집니다. SQL명을 지정할 때 규정된 형식은 스키마명, 마침표(.), SQL ID 순입니다. 시스템명을 지정하는 경우 규정된 형식은 스키마명, 슬래시(/), SQL ID 순입니다.

규정되지 않은 형식은 SQL ID입니다. 규정되지 않은 형식은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』에 지정된 규칙에 따라 내재적으로 규정됩니다.

뷰 이름은 뷰 이름이나 뷰의 시스템 오브젝트명을 지정할 수 있습니다.

표 2. ID 길이 한계(바이트)

ID 유형	최대 길이
별명	128
권한부여명	10
상관명	128
커서명	18
호스트 ID	64
저장점명	128
서버명	18
SQL 레이블	128
명령문 이름	18
규정되지 않은 스키마명	10
규정되지 않은 열 이름	30
규정되지 않은 제한사항명	128

표 2. ID 길이 한계(바이트) (계속)

ID 유형	최대 길이
규정되지 않은 고유한 유형 이름	128
규정되지 않은 외부 프로그램명 [15]	10
규정되지 않은 함수명	128
규정되지 않은 노드 그룹명	10
규정되지 않은 패키지명	10
규정되지 않은 매개변수명	128
규정되지 않은 프로시듀어명	128
규정되지 않은 특정 이름	128
규정되지 않은 SQL 매개변수명	128
규정되지 않은 SQL 변수명	128
규정되지 않은 시스템 열 이름	10
규정되지 않은 시스템 오브젝트명	10
규정되지 않은 표, 뷰 및 색인명	128
규정되지 않은 트리거명	128

규정되지 않은 오브젝트명의 규정화

규정되지 않은 오브젝트명은 내재적으로 규정됩니다. 이름 규정화 규칙은 이름으로 식별되는 오브젝트의 유형에 따라 달라집니다.

규정되지 않은 별명, 제한사항, 외부 프로그램, 색인, 노드 그룹, 패키지, 표, 트리거 및 뷰 이름

규정되지 않은 별명, 제한사항, 외부 프로그램, 색인, 노드 그룹, 패키지, 표, 트리거 및 뷰 이름은 다음과 같이 내재적으로 규정됩니다.

- 정적 SQL문의 경우:
  - CRTSQLxxx 명령에 DFTRDBCOL 매개변수가 지정되었거나 SET OPTION문을 함께 사용한 경우내재적 규정자는 이 매개변수에 지정된 스키마명입니다.
  - 기타 모든 경우 내재 규정자는 명명 규칙에 따라 달라집니다.
    - SQL 이름을 지정하는 경우 내재 규정자는 명령문의 권한부여 ID입니다.
    - 시스템 이름을 지정하는 경우 내재 규정자는 작업 라이브러리 리스트(*LIBL)입니다.
- 동적 SQL문의 경우 내재 규정자는 디폴트 스키마명이 명시적으로 지정되었는지 여부에 따라 달라집니다. 이와같은 명시적인 지정의 메카니즘은 SQL문을 동적으로 준비하고 실행하는 데 사용되는 인터페이스에 따라 달라집니다.
  - 디폴트 스키마명은 명시적으로 지정되지 않습니다.
    - SQL 이름을 지정하는 경우 내재 규정자는 실행시 권한부여 ID입니다.

15. REXX 프로시듀어의 경우 한계는 33입니다.

- 시스템 이름을 지정하는 경우 내재 규정자는 작업 라이브러리 리스트(*LIBL) 입니다.

– 디폴트 스키마명이 명시적으로 지정되면 내재 규정자가 해당 디폴트 스키마입니다. 디폴트 스키마명은 다음과 같은 인터페이스로 지정될 수 있습니다.

표 3. 디폴트 스키마 인터페이스

SQL 인터페이스	**스펙**
삽입 SQL	SQL 프로그램 작성(CRTSQLxxx) 및 SQL 패키지 작성(CRTSQLPKG) 명령에 대한 DFTRDBCOL 매개변수 및 DYNDFTCOL(*YES). SET OPTION 명령문도 DFTRDBCOL 및 DYNDFTCOL 값을 설정하는 데 사용될 수 있습니다. (CRTSQLxxx 명령에 대한 자세한 정보는 호스트 언어로 SQL 프로그래밍 책을 참조하십시오).
SQL문 실행	Run SQL문(RUNSQLSTM) 명령의 DFTRDBCOL 매개변수. (RUNSQLSTM 명령에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.)
서버의 호출 레벨 인터페이스	SQL_ATTR_DEFAULT_LIB 또는 SQL_ATTR_DBC_DEFAULT_LIB 환경 또는 연결 변수 (CLI에 대한 자세한 정보는 SQL 호출 레벨 인터페이스(ODBC) 책을 참조하십시오.)
Developer Kit for Java를 사용한 서버의 JDBC 또는 SQLJ	라이브러리 등록 정보 오브젝트 (JDBC와 SQLJ에 대한 자세한 정보는 iSeries Information Center의 IBM Developer Kit for Java 주제를 참조하십시오).
iSeries Access ODBC 드라이버를 사용하는 클라이언트의 ODBC	ODBC 설정의 SQL 디폴트 라이브러리 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries 액세스 범주를 참조하십시오.)
IBM Toolbox for Java를 사용하는 클라이언트의 JDBC	JDBC 설정의 SQL 디폴트 라이브러리 (JDBC에 대한 자세한 정보는 iSeries Information Center의 iSeries 액세스 범주를 참조하십시오.) (IBM Toolbox for Java에 대한 자세한 정보는 iSeries Information Center의 IBM Toolbox for Java 주제를 참조하십시오).
모든 인터페이스	SET SCHEMA 또는 QSQCHGDC(디폴트 콜렉션 동적 변경) API (QSQCHGDC에 대한 자세한 정보는 iSeries Information Center의 파일 API 범주를 참조하십시오).

규정되지 않은 함수, 프로시듀어, 특정 및 고유한 유형 이름

자료 유형(내장 유형 및 고유한 유형), 함수, 프로시듀어 및 특정 이름의 규정화는 규정되지 않은 이름이 나오는 SQL문에 따라 달라집니다.

- 규정되지 않은 이름이 CREATE, COMMENT, DROP, GRANT 또는 REVOKE 명령문의 기본 오브젝트이면 규정되지 않은표 이름을 규정할 때와 같은 규칙을 사용

하여 내재적으로 규정됩니다(54 페이지의 『규정되지 않은 별명, 제한사항, 외부 프로그램, 색인, 노드 그룹, 패키지, 표, 트리거 및 뷰 이름』 참조).

- 그렇지 않으면 내재 스키마명은 다음과 같이 판별됩니다.
  - 고유한 유형 이름의 경우 데이터베이스 관리자는 SQL 경로를 찾아서 해당 자료 유형이 있는 첫 번째 스키마를 선택합니다.
  - 프로시듀어 이름의 경우 데이터베이스 관리자는 SQL 경로를 찾아서 매개변수와 같은 이름 및 수를 갖고 있는 프로시듀어가 있는 첫 번째 스키마를 선택합니다.
  - 함수명과 소스인 함수에 지정된 특정 이름의 경우 데이터베이스 관리자는 123 페이지의 『함수 분석』에서 설명한 대로 함수 분석과 함께 SQL 경로를 사용합니다.

SQL 이름과 시스템명: 특수 고려사항

CL 명령 OVRDBF(데이터베이스 파일 대체)를 지정하여 SQL 또는 시스템명을 로컬 자료 조작 SQL문의 다른 오브젝트명으로 대체할 수 있습니다. 리모트 관계형 데이터베이스에서 실행되는 자료 정의 SQL문과 자료 조작 SQL문에 대한 대체는 무시됩니다. 대체 함수에 대한 자세한 정보는 파일 관리 책을 참조하십시오.

스키마 및 SQL 경로

SQL 경로는 순서대로 나열된 스키마명의 리스트입니다. 데이터베이스 관리자는 경로를 사용하여 CREATE, DROP, COMMENT, GRANT 또는 REVOKE문의 기본 오브젝트를 제외한 모든 문맥에 나타나는 규정되지 않은 고유한 유형 이름(내장 유형 및 고유한 유형 모두), 함수명 및 프로시듀어명에 대한 스키마명을 분석합니다. 데이터베이스 관리자는 왼쪽에서 오른쪽으로 경로를 찾아서 규정되지 않은 동일한 이름의 오브젝트가 들어 있는 첫 번째 스키마를 갖는 오브젝트를 내재적으로 규정합니다. 프로시듀어의 경우 데이터베이스 관리자는 매개변수의 수도 같은 경우에만 대응하는 프로시듀어명을 선택합니다. 함수의 경우 하나의 스키마에 같은 이름의 여러 함수가 있을 수 있으므로 데이터베이스 관리자는 SQL 경로와 함께 함수 분석이라는 프로세스를 사용하여 어떤 함수를 선택할 것인지 판별합니다. (자세한 내용은 123 페이지의 『함수 분석』을 참조하십시오).

예를 들어 SQL 경로가 SMITH, XGRAPHIC, QSYS, QSYS2이고 규정되지 않은 고유한 유형 이름 MYTYPE이 지정된 경우 데이터베이스 관리자는 먼저 SMITH 스키마에서 MYTYPE을 찾고 그 다음 SGRAPHIC, QSYS, QSYS2에서 차례로 찾습니다.

사용된 경로는 다음과 같이 판별됩니다.

- (CALL: *호스트 변수* 명령문을 제외한) 모든 정정 SQL문의 경우 사용되는 경로는 CRTSQLxxx 명령의 SQLPATH 매개변수에 지정된 경로입니다. SQLPATH는 SET OPTION 명령문을 사용하여 설정될 수도 있습니다.
- 동적 SQL문(및 CALL: *호스트 변수* 명령문)의 경우 사용된 경로는 CURRENT PATH 특수 레지스터에 지정된 경로입니다. CURRENT PATH 특수 레지스터에 대

한 자세한 정보는 105 페이지의 『CURRENT PATH, CURRENT_PATH 또는 CURRENT FUNCTION PATH』를 참조하십시오.

별명

데이터베이스 파일의 표, 뷰 또는 멤버에 대한 대체 이름으로 *별명*을 사용합니다. 별명은 파일 대체 속성을 방지할 수 있습니다. 별명은 대체를 더 잘 수행할 뿐만 아니라 한 번만 작성하는 영구 오브젝트입니다.

이름이나 표 별명으로 SQL문의 표 또는 뷰를 참조할 수 있습니다. 별명을 사용하면 SQL문의 데이터베이스 파일 멤버만 참조할 수 있습니다. 별명은 같은 관계형 데이터베이스 내의 표, 뷰 또는 데이터베이스 파일 멤버만을 참조할 수 있습니다.

다음 경우를 제외하고 표 또는 뷰 이름을 사용할 때는 별명을 사용할 수 있습니다.

- CREATE TABLE 또는 CREATE VIEW 명령문과 같이 새로운 표 또는 뷰 이름이 기대되는 별명은 사용하지 마십시오. 예를 들어, PERSONNEL의 별명이 작성되면 CREATE TABLE PERSONNEL과 같은 후속 명령문이 오류 발생합니다.
- 데이터베이스 파일 멤버의 각 멤버를 참조하는 별명은 선택 명령문, DELETE, INSERT, SELECT INTO, SET UPDATE 또는 VALUES INTO 명령문에서만 사용할 수 있습니다.

별명이 참조하는 오브젝트가 없더라도 별명을 작성할 수 있습니다. 그러나 별명을 참조하는 명령문이 실행될 때는 오브젝트가 있어야 합니다. 별명을 작성할 때 오브젝트가 없으면 경고가 리턴됩니다. 별명은 다른 별명을 참조할 수 없습니다. 별명은 같은 관계형 데이터베이스 내의 표, 뷰 또는 데이터베이스 파일 멤버만을 참조할 수 있습니다.

별명으로 표, 뷰 또는 데이터베이스 파일 멤버를 참조하는 옵션은 명시적으로 구문 도표에 표시되거나 SQL문의 설명에 언급되지 않습니다.

새로운 별명으로 기존 표, 뷰, 색인, 파일 또는 별명으로 완전히 규정된 같은 이름을 사용할 수 없습니다.

SQL문에서 별명을 사용하는 효과는 텍스트 대체 효과와 같습니다. SQL문을 실행할 때 정의해야 하는 별명은 규정된 기본 표, 뷰 또는 데이터베이스 파일 멤버명에 의해 대체되지 않습니다. 예를 들어, PBIRD.SALES이 DSPN014.DIST4_SALES_148에 대한 별명이면 아래 나오는 명령문이 그 다음 행에 나오는 명령문으로 대체됩니다.

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

즉, 위의 명령문이 다음과 같이 변경됩니다.

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

별명이 제거되고 다른 표를 참조하기 위해 다시 작성되는 경우 별명을 참조하는 SQL 문은 다음 실행시에 내재적으로 리바인드됩니다. CREATE VIEW 또는 CREATE INDEX 명령문이 별명을 참조하는 경우 별명을 제거하고 다시 작성해도 뷰 또는 색인에 영향을 주지 않습니다.

기존 OS/390 및 z/OS용 DB2 UDB 어플리케이션에 대한 구문 허용의 경우 CREATE ALIAS 및 DROP ALIAS 명령문에서 ALIAS 대신 SYNONYM을 사용할 수 있습니다.

권한부여 ID 및 권한부여명

*권한부여 ID*는 데이터베이스 관리자와 어플리케이션 프로세스 또는 프로그램 준비 프로세스 사이에 연결이 설정될 때 데이터베이스 관리자가 사용하는 문자 스트링입니다. 권한 세트를 지정합니다. 이 등록 정보는 사용자나 사용자 그룹을 지정할 수도 있으나 데이터베이스 관리자에 의해 제어되지 않습니다.

권한부여 ID는 모든 명령문에 적용되고 데이터베이스 관리자에 의해 사용되어 다음을 제공합니다.

- 표, 뷰, 제한사항, 패키지 및 색인명에 대한 내재적인 규정자
- SQL문의 권한부여 검사

권한부여 ID는 모든 SQL문에 적용됩니다. 내재적인 규정화는 정적 또는 동적 SQL 사용 여부에 따라 달라집니다.

- 정적 SQL의 경우 내재 규정자가 프로그램의 소유자입니다.
- 동적 SQL의 경우 내재 규정자는 프로그램을 실행 중인 사용자입니다.

정적 SQL문의 권한부여 검사에 사용되는 권한부여 ID는 사전컴파일러 명령에 지정된 USRPRF 값에 따라 달라집니다.

- USRPRF(*OWNER)가 지정되거나 USRPRF(*NAMING)가 지정되고 SQL 명명 모드가 사용되면 명령문의 권한부여 ID는 분배되지 않은 SQL 프로그램의 소유자입니다. 분배된 SQL 프로그램의 경우 SQL 패키지 소유자입니다.
- USRPRF(*USER)가 지정되거나 USRPRF(*NAMING)가 지정되고 시스템 명명 모드가 사용되면 명령문의 권한부여 ID는 분배되지 않은 SQL 프로그램을 실행 중인 사용자의 권한부여 ID입니다. 분배된 SQL 프로그램의 경우 현재 서버에 있는 사용자의 권한부여 ID입니다.

동적 SQL문의 권한부여 검사에 사용되는 권한부여 ID는 명령문이 실행되는 위치와 방법에 따라서도 달라집니다.

- 분배되지 않는 프로그램에서 명령문이 준비되고 실행되는 경우

  - 프로그램에 대한 USRPRF 값이 *USER이고 DYNUSRPRF 값이 *USER인 경우 적용되는 권한부여 ID는 분배되지 않는 프로그램을 실행 중인 사용자의 ID입니다. 이것을 *실행시 권한부여 ID*라고 합니다.
  - 프로그램에 대한 USRPRF 값이 *OWNER이고 DYNUSRPRF 값이 *USER인 경우 적용되는 권한부여 ID는 분배되지 않는 프로그램을 실행 중인 사용자의 ID입니다.
  - 프로그램에 대한 USRPRF 값이 *OWNER이고 DYNUSRPRF 값이 *OWNER일 경우 적용되는 권한부여 ID는 분배되지 않는 프로그램의 소유자 ID입니다.
- 분배된 프로그램에서 명령문이 준비되고 실행될 경우
  - SQL 패키지에 대한 USRPRF 값이 *USERR이고 DYNUSRPRF 값이 *USER일 경우 적용되는 권한부여 ID는 현재 서버에서 SQL 패키지를 실행 중인 사용자의 ID입니다. 이것을 실행시 권한부여 ID라고도 합니다.
  - SQL 패키지에 대한 USRPRF 값이 *OWNER이고 DYNUSRPRF 값이 *USER일 경우 적용되는 권한부여 ID는 현재 서버에서 SQL 패키지를 실행 중인 사용자의 ID입니다.
  - SQL 패키지에 대한 USRPRF 값이 *OWNER이고 DYNUSRPRF 값이 *OWNER일 경우 적용되는 권한부여 ID는 현재 서버에 있는 SQL 패키지 소유자의 ID입니다.
- 명령문이 대화식으로 발행되는 경우 적용되는 권한부여 ID는 STRSQL(SQL 시작) 명령을 발행한 사용자의 ID입니다.
- RUNSQLSTM 명령의 명령문이 실행되는 경우 적용되는 권한부여 ID는 RUNSQLSTM 명령을 실행하는 사용자의 ID입니다.
- REXX 명령의 명령문이 실행되는 경우 적용되는 권한부여 ID는 STRREXPRC 명령을 발행하는 사용자의 ID입니다.

OS/400에서 실행시 권한부여 ID는 작업의 사용자 프로파일입니다.

SQL문에 지정된 *권한명*을 명령문의 권한부여 ID와 혼동하지 않아야 합니다. 권한부여명은 부여 또는 호출의 목표를 지정하기 위해 GRANT 및 REVOKE 명령문에서 사용되는 ID입니다. 권한 부여 *X*의 전제는 *X*가 계속해서 이런 권한을 요구하는 명령문의 권한부여 ID라는 것입니다. SQL문의 권한을 검사할 때 그룹 사용자 프로파일이 사용

될 수도 있습니다. 그룹 사용자 프로파일에 대한 자세한 정보는 iSeries 보안 참조서  를 참조하십시오.

예

- SMITH가 사용자 ID이고 다음 명령문을 대화식으로 실행할 때의 권한부여 ID라고 가정합니다.

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

권한부여 ID 및 이름

SMITH는 명령문의 권한부여 ID입니다. 따라서 SMITH에 대해 명령문 실행 권한이 검사되고 SMITH는 TDEPT의 내재 규정자입니다.

KEENE은 명령문에 지정된 권한부여명입니다. KEENE는 SMITH.TDEPT에 대한 SELECT 권한이 부여됩니다.

- SMITH에게 관리 권한이 있고 다음 명령문의 권한부여 ID라고 가정합니다.

  ```
  DROP TABLE TDEPT
  ```

  SMITH.TDEPT 표를 제거합니다.

  ```
  DROP TABLE SMITH.TDEPT
  ```

  SMITH.TDEPT 표를 제거합니다.

  ```
  DROP TABLE KEENE.TDEPT
  ```

  KEENE.TDEPT 표를 제거합니다.

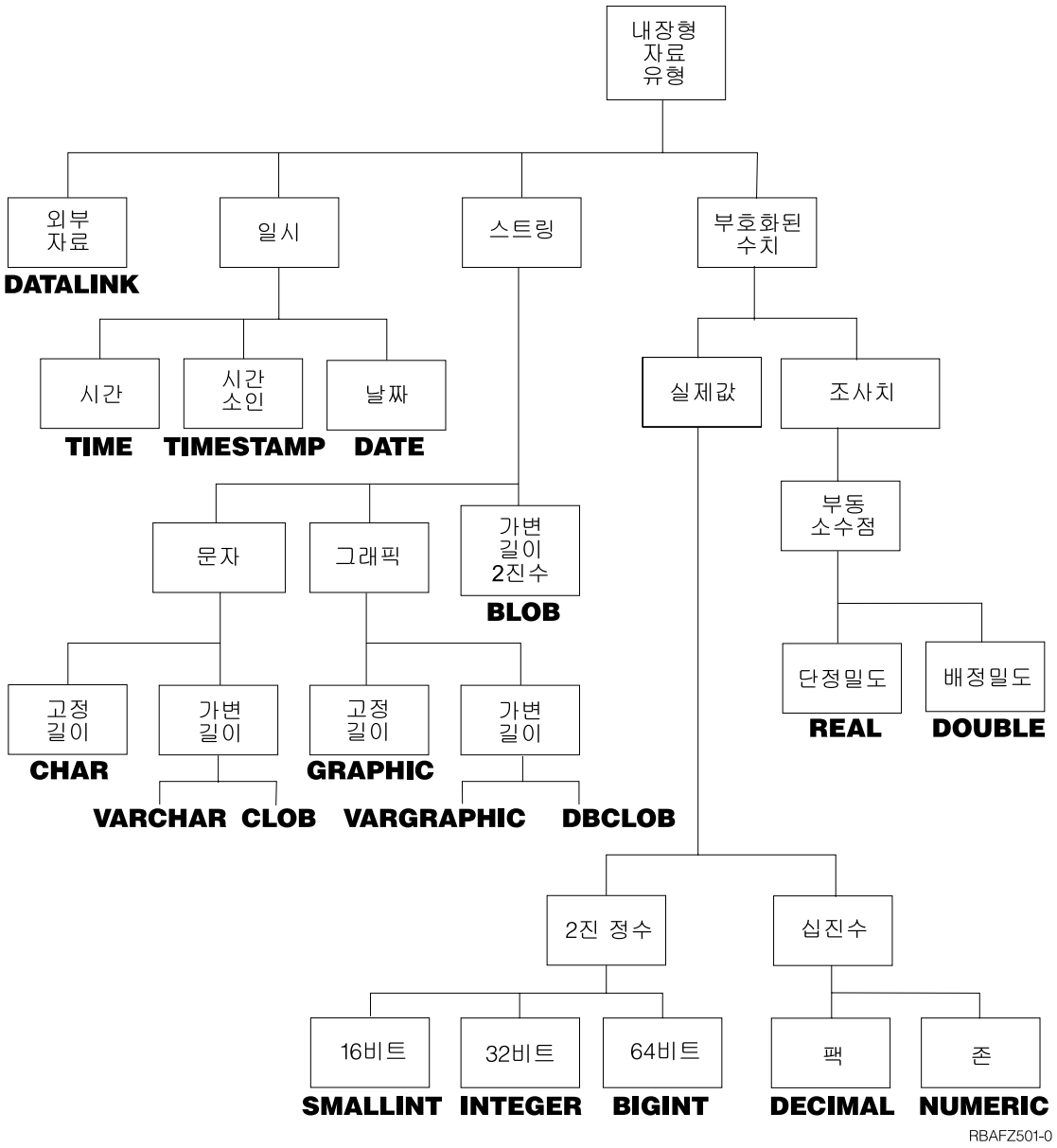
자료 유형

SQL에서 조작할 수 있는 자료의 가장 작은 단위를 값이라고 합니다. 값이 해석되는 방법은 소스의 자료 유형에 따라 달라집니다. 값의 소스는 다음과 같습니다.

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

DB2는 IBM이 제공한 자료 유형(내장 자료 유형)과 사용자 정의 자료 유형(고유 유형)을 모두 지원합니다. 이 섹션은 내장 자료 유형을 설명합니다. 고유한 유형에 대한 설명은 75 페이지의 『사용자 정의 유형』을 참조하십시오.

다음 그림은 iSeries용 DB2 UDB 프로그램에서 지원되는 여러 내장 자료 유형을 나타냅니다.



널: 모든 자료 유형에는 널값이 포함됩니다. 널값은 널이 아닌 값보다 고유한 특수 값이므로(널이 아닌) 값이 없음을 나타냅니다. 모든 자료 유형에 널값이 포함되어 있더라도 값의 소스는 널값이 들어갈 수 없습니다. 예를 들어, 상수, NOT NULL로 정의된 열. 특수 레지스터는 널값을 포함할 수 없고, COUNT 및 COUNT_BIG 함수는 널값을 리턴할 수 없습니다. 조회의 결과 ROWID 열에 대해 널값이 리턴될 수 있지만 ROWID 열은 널값을 저장할 수 없습니다.

자료 유형에 대한 자세한 내용은 다음 주제를 참조하십시오.

- 62 페이지의『2진 스트링』
- 62 페이지의『문자 스트링』
- 63 페이지의『문자 부속유형』

- 64 페이지의『그래픽 스트링』
- 65 페이지의『그래픽 부속유형』
- 66 페이지의『큰 오브젝트(LOB)』
- 67 페이지의『숫자』
- 68 페이지의『Datetime 값』
- 74 페이지의『DataLink 값』
- 75 페이지의『행 ID 값』
- 75 페이지의『사용자 정의 유형』

열의 자료 유형 지정에 대한 내용은 541 페이지의『CREATE TABLE』을 참조하십시오.

2진 스트링

*2진 스트링*은 일련의 바이트입니다. 2진 스트링(BLOB 스트링)의 길이는 연속되는 바이트 수입니다. 2진 스트링에는 65535의 CCSID가 있습니다.

BLOB 열의 경우 길이 속성은 1 - 2 147 483 647 바이트 범위에 있어야 합니다. BLOB에 대한 자세한 내용은 66 페이지의『큰 오브젝트(LOB)』를 참조하십시오.

BLOB 스트링 유형의 호스트 변수는 REXX, RPG/400 및 COBOL/400을 제외한 모든 호스트 언어로 정의할 수 있습니다.

문자 스트링

*문자 스트링*은 일련의 바이트입니다. 스트링의 길이는 연속되는 바이트 수입니다. 길이가 0이면 값을 *빈 스트링*이라고 합니다. 빈 스트링과 널값을 혼동하지 마십시오.

고정 길이 문자 스트링

고정 길이 문자 스트링 열의 모든 값은 길이가 같습니다. 열의 길이 속성에 의해 판별됩니다. 길이 속성은 1 이상 32766 이하이어야 합니다.

가변 길이 문자 스트링

가변 길이 문자 스트링의 유형은 다음과 같습니다.

- VARCHAR(또는 동의어 CHAR VARYING 및 CHARACTER VARYING)
- CLOB(또는 동의어 CHAR LARGE OBJECT 및 CHARACTER LARGE OBJECT)

이들 스트링 유형 중 하나인 열의 값은 다른 길이를 가질 수 있습니다. 열의 길이 속성은 값이 가질 수 있는 최대 길이를 판별합니다.

VARCHAR 열의 경우 길이 속성은 1 - 32740 범위에 있어야 합니다. CLOB 열의 경우 길이 속성은 1 - 2 147 483 647 범위에 있어야 합니다. CLOB에 대한 자세한 정보는 66 페이지의 『큰 오브젝트(LOB)』를 참조하십시오.

문자 스트링 호스트 변수

- 고정 길이 문자 스트링 변수는 REXX를 제외한 모든 호스트 언어에 사용할 수 있습니다(C에서 고정 길이 문자 스트링 변수는 길이가 1로 제한됩니다).
- VARCHAR 가변 길이 문자 스트링 변수는 C, COBOL, PL/I, REXX 및 RPG에서 사용할 수 있습니다.
  - PL/I, REXX 및 ILE RPG에 가변 길이 문자 스트링 자료 유형이 있습니다.
  - COBOL과 C에서 가변 길이 문자 스트링은 구조로 표시됩니다.
  - C에서 가변 길이 문자 스트링 변수도 널 종료 스트링으로 표시될 수 있습니다.
  - RPG/400에서 가변 길이 문자 스트링 변수는 외부 서술 자료 구조의 결과로 포함된 VARCHAR 열에 의해서만 표시될 수 있습니다.
- CLOB 가변 길이 문자 스트링 변수는 REXX, RPG/400 및 COBOL/400을 제외한 모든 호스트 언어에서 정의할 수 있습니다.
  - ILE RPG에서 CLOB 가변 길이 문자 스트링은 SQLTYPE 키워드를 사용하여 선언됩니다.
  - 기타 다른 언어에서는 SQL TYPE IS CLOB 절이 사용됩니다.

문자 부속유형

각 문자 스트링은 다음 중 하나로 더 자세히 정의됩니다.

비트 자료 코드화 문자 세트와 연관되지 않고 변환되지 않는 자료. 비트 자료의 CCSID는 65535입니다.

SBCS 자료 모든 문자가 1바이트로 표시되는 자료. 각 SBCS 자료 문자 스트링에는 연관된 CCSID가 있습니다. 필요하면 SBCS 자료 문자 스트링이 CCSID가 다른 문자 스트링과 조작에 사용되기 전에 변환됩니다.

혼합 자료 1바이트 문자 세트(SBCS)와 2바이트 문자 세트(DBCS)의 문자가 혼합되어 들어 있는 자료. 각 혼합 자료 문자 스트링에는 연관된 CCSID가 있습니다. 필요하면 혼합 자료 문자 스트링이 CCSID가 다른 문자 스트링과 조작에 사용되기 전에 변환됩니다. 혼합 자료에 DBCS 문자가 들어 있으면 SBCS 자료로 변환될 수 없습니다.

데이터베이스 관리자는 2바이트 문자의 서브클래스를 인식하지 않으며 특정 2바이트 코드로 특정한 의미를 지정하지 않습니다. 그러나 혼합 자료를 사용하도록 선택하면 2개의 1바이트 EBCDIC 코드에는 다음과 같은 특별한 의미가 부여됩니다.

- X'0E', “SO” 문자를 사용하여 일련의 2바이트 코드 시작을 표시합니다.
- X'0F', “SI” 문자를 사용하여 일련의 2바이트 코드 끝을 표시합니다.

데이터베이스 관리자가 혼합 자료 문자 스트링의 2바이트 문자를 인식하려면 다음 조건이 맞아야 합니다.

스트링 내의 2바이트 문자는 한 쌍의 SO와 SI 문자로 묶어야 합니다.

스트링이 왼쪽에서 오른쪽으로 읽혀질 때 쌍이 감지됩니다. X'0F'가 나중에 나오면 코드 X'0E'는 SO 문자로 인식되나 그렇지 않은 경우 유효하지 않습니다. 2바이트 경계에 있는 X'0E' 다음의 첫 X'0F'는 쌍이되는 SI 문자입니다. 2바이트 경계에 있지 않는 X'0F'는 인식되지 않습니다.

쌍이 되는 문자 사이에는 짝수 개의 바이트가 있어야 하고 바이트의 각 쌍은 2바이트 문자로 간주됩니다. 스트링에는 2개 이상의 SO와 SI 문자 쌍이 있을 수 있습니다.

혼합 자료 문자 스트링의 길이는 각 2바이트 문자에 2바이트씩, 각 SO 또는 SI 문자에 한 바이트씩 계산한 총 바이트 수입니다.

작업 CCSID에서 DBCS가 허용됨을 지정하면 FOR BIT DATA, FOR SBCS DATA 또는 SBCS CCSID가 지정되지 않는 한 CREATE TABLE은 DBCS 개방 필드로 문자 열을 작성합니다. SQL 사용자에게 문자 필드로 보이지만 시스템 데이터베이스 지원에게는 DBCS 개방 필드로 보입니다. DBCS 개방 필드의 정의에 대해서는 데이터베이스 프로그래밍 책을 참조하십시오.

그래픽 스트링

그래픽 스트링은 일련의 2바이트 문자입니다. 스트링의 길이는 문자 수입니다. 문자 스트링과 같이 그래픽 스트링은 비어 있을 수 있습니다.

고정 길이 그래픽 스트링

고정 길이 그래픽 스트링 열의 모든 값은 열의 길이 속성에 의해 판별되는 길이와 같습니다. 길이 속성은 1 이상 16383 이하이어야 합니다.

가변 길이 그래픽 스트링

가변 길이 그래픽 스트링의 유형은 다음과 같습니다.

- VARGRAPHIC(또는 동의어 GRAPHIC VARYING)
- DBCLOB

이들 스트링 유형 중 하나인 열의 값은 다른 길이를 가질 수 있습니다. 열의 길이 속성은 값이 가질 수 있는 최대 길이를 판별합니다.

VARGRAPHIC 열의 경우 길이 속성은 1 - 16370 범위에 있어야 합니다. DBCLOB 열의 경우 길이 속성은 1 - 1 073 741 823 범위에 있어야 합니다. DBCLOB에 대한 자세한 정보는 66 페이지의 『큰 오브젝트(LOB)』를 참조하십시오.

그래픽 스트링 호스트 변수

- 고정 길이 그래픽 스트링 호스트 변수는 C, ILE COBOL 및 ILE RPG/400으로 정의될 수 있습니다(C에서 고정 길이 그래픽 스트링 호스트 변수는 길이가 1로 제한됩니다).

  고정 길이 그래픽 스트링 호스트 변수가 PL/I, COBOL/400 및 RPG/400으로 정의될 수 없더라도 문자 스트링 호스트 변수가 파일의 외부 정의에 있는 GRAPHIC 열에서 소스에 생성되었으면 고정 길이 그래픽 스트링 호스트 변수와 같이 처리됩니다.
- 가변 길이 그래픽 스트링 호스트 변수는 C, ILE COBOL, REXX 및 ILE RPG로 정의될 수 있습니다.
  - REXX와 ILE RPG에는 가변 길이 그래픽 스트링 자료 유형이 있습니다.
  - C와 ILE COBOL에서 가변 길이 그래픽 스트링은 구조로 표시됩니다.
  - C에서 가변 길이 그래픽 스트링 변수는 널 종료 스트링으로도 표시될 수 있습니다.
  - 가변 길이 그래픽 스트링 호스트 변수가 PL/I, COBOL/400 및 RPG/400으로 정의될 수 없더라도 문자 스트링 호스트 변수가 파일의 외부 정의에 있는 VARGRAPHIC 열에서 소스에 생성되었으면 가변 길이 그래픽 스트링 호스트 변수와 같이 처리됩니다.
- DBCLOB 가변 길이 문자 스트링 변수는 REXX, RPG/400 및 COBOL/400을 제외한 모든 호스트 언어에서 정의할 수 있습니다.
  - ILE RPG에서 DBCLOB 가변 길이 문자 스트링은 SQLTYPE 키워드를 사용하여 선언됩니다.
  - 기타 다른 언어에서는 SQL TYPE IS DBCLOB 절이 사용됩니다.

그래픽 부속유형

각 그래픽 스트링은 DBCS 자료 또는 UCS-2 자료로 더 자세히 정의됩니다.

DBCS 자료 모든 문자가 SO 또는 SI 문자가 포함되지 않은 DBCS(2바이트 문자 세트)의 문자로 표시되는 자료.

모든 DBCS 그래픽 스트링에는 2바이트 코드화 문자 세트를 식별하는 CCSID가 있습니다. 필요하면 DBCS 그래픽 스트링이 DBCS CCSID가 다른 DBCS 그래픽 스트링과 조작에 사용되기 전에 변환됩니다.

UCS-2 자료 모든 문자가 범용 코드화 문자 세트(UCS-2)의 문자로 표시되는 자료.

그래픽 스트링 호스트 변수가 CCSID로 명확하게 태그 표시되지 않으면 작업 CCSID에 관련된 DBCS CCSID가 사용됩니다. 연관된 DBCS CCSID가 없으면 호스트 변수가 65535로 태그 표시됩니다. 그래픽 스트링 호스트 변수는 내재적으로 UCS-2 CCSID로 태그 표시되지 않습니다. CCSID로 그래픽 호스트 변수를 태그 표시하는 방법에 대한 내용은 DECLARE VARIABLE 명령문을 참조하십시오.

큰 오브젝트(LOB)

용어 *큰 오브젝트(LOB)*는 다음 자료 유형을 참조합니다.

2진 큰 오브젝트(BLOB) 스트링

*큰 2진 오브젝트(BLOB)*는 최대 길이가 2 147 483 647인 가변 길이 스트링입니다. BLOB는 영상, 음성 및 혼합 매체와 같은 비전통 자료를 저장하기 위해 설계되었습니다. BLOB는 고유한 유형과 사용자 정의 함수에서 사용할 구조화 자료도 저장할 수 있습니다. BLOB는 2진 스트링으로 간주됩니다.

BLOB 스트링과 FOR BIT DATA 문자 스트링이 유사한 목적을 위해 사용될 수 있더라도 두 가지 자료 유형은 호환될 수 없습니다. BLOB 함수를 사용하여 FOR BIT DATA 문자 스트링을 이진 스트링으로 변경할 수 있습니다.

BLOB의 CCSID는 65535입니다.

문자 큰 오브젝트(CLOB) 스트링

*큰 문자 오브젝트(CLOB)*는 최대 길이가 2 147 483 647인 가변 길이 문자 스트링입니다. CLOB는 긴 문서와 같은 큰 SBCS 자료 또는 혼합 자료를 저장하기 위해 설계되었습니다. 예를 들면, 사원 이력서, 연극 대본 또는 소설의 원문과 같은 정보를 CLOB에 저장할 수 있습니다.

CLOB의 CCSID는 65535일 수 없습니다.

2바이트 문자 큰 오브젝트(DBCLOB) 스트링

*큰 2바이트 문자 오브젝트(DBCLOB)*는 최대 길이가 1 073 741 823 2바이트 문자인 가변 길이 그래픽 스트링입니다. UCS-2의 긴 문서와 같은 큰 DBCS를 저장하기 위해 DBCLOB가 설계되었습니다.

DBCLOB의 CCSID는 65535일 수 없습니다.

로케이터로 큰 오브젝트(LOB) 조작

어플리케이션이 호스트 변수로 이동될 전체 LOB를 요구하지 않을 때 어플리케이션은 LOB(큰 오브제트) 로케이터를 사용하여 LOB 값을 참조할 수 있습니다.

LOB 로케이터는 데이터베이스 서버의 단일 LOB 값을 나타내는 값을 갖는 호스트 변수입니다. LOB 로케이터는 전체 LOB 값이 호스트 변수에 저장 또는 어플리케이션 프로그램이 실행 중일 어플리케이션 리퀘스터(클라이언트)로 전송 요구없이 어플리케이션 프로그램에서 아주 큰 오브젝트가 조작될 수 있는 메카니즘을 제공하기 위한 것입니다.

예를 들어, LOB 값을 선택할 경우 어플리케이션 프로그램은 전체 LOB 값을 선택하여(어플리케이션 프로그램이 한 번에 전체 LOB 값을 처리할 경우 허용될 수 있는) 동등하게 큰 호스트 변수에 위치 지정하거나 LOB 값을 LOB 로케이터로 대신 선택할 수 있습니다. 그런 다음 LOB 로케이터를 사용하여 로케이터 값을 입력 지정하여 어플리

케이션 프로그램은(스칼라 함수 SUBSTR, CONCAT, VALUE, LENGTH 적용, 지정 수행, LIKE 또는 POSSTR로 LOB 탐색 또는 LOB에 대해 UDF 적용하는) LOB 값에 대한 후속 데이터베이스 조작을 발행할 수 있습니다. 클라이언트 호스트 변수에 할당된 자료의 양과 같이 로케이터 조작의 결과 출력은 일반적으로 입력 LOB 값의 작은 서브세트입니다.

LOB 로케이터는 다음과 같은 LOB 표현식을 표시할 수도 있습니다.

```
  SUBSTR((lob1) CONCAT (lob2 )
CONCAT (lob3), start, length )
```

어플리케이션 프로그램의 정상 호스트 변수에 대해, 널값이 호스트 변수로 선택될 때 인디케이터 변수는 값이 널임을 나타내는 -1로 설정됩니다. 그러나 LOB 로케이터의 경우 인디케이터 변수의 의미는 약간 다릅니다. 로케이터 호스트 변수 자체가 널이 될 수 없으므로 음 인디케이터 변수 값은 LOB 로케이터로 표시되는 LOB 값이 널임을 나타냅니다. 널 정보에서는 로컬이 인디케이터 변수 값에 의해 클라이언트로 보유됩니다. 서버는 유효한 로케이터로 널값을 추적하지 않습니다.

LOB 로케이터는 데이터베이스의 행이나 위치가 아니라 값을 나타냄을 이해하는 것이 중요합니다. 일단 값을 선택하여 로케이터에 넣으면 로케이터에 의해 참조되는 값에 영향을 주는 원래 행이나 표에서 수행할 수 있는 조작이 없습니다. 로케이터와 연관된 값은 트랜잭션이 종료할 때까지 또는 로케이터가 명시적으로 해제될 때까지 어떤 경우든지 유효합니다.

LOB 로케이터는 트랜잭션중에 LOB 값 참조에 사용되는 유일한 메카니즘이며 작성된 트랜잭션을 지나면 지속되지 않습니다. 또한 데이터베이스 유형이 아니므로 데이터베이스에 저장되지 않고 따라서 뷰나 검사 제한조건에 참여할 수 없습니다. 그러나, 로케이터가 LOB 유형의 표시이므로 FETCH, OPEN, CALL 및 EXECUTE 명령문에서 사용되는 SQLDA 구조 내에서 설명될 수 있도록 LOB 로케이터에 대해 SQLTYPE이 있습니다.

숫자

모든 숫자에는 *부호*와 *정밀도*가 있습니다. 정밀도는 부호를 제외한 2진 또는 십진수의 전체 자릿수입니다. 값이 0이면 부호는 양입니다.

작은 정수

*작은 정수*는 5 자릿수의 정밀도를 갖는 2바이트(16 비트)로 구성된 2진수입니다. 작은 정수의 범위는 -32768 - +32767입니다.

작은 정수의 경우 소수 정밀도와 스케일은 COBOL, RPG 및 iSeries 시스템 파일에서 지원됩니다. 2진 정수의 정밀도와 스케일에 관한 내용은 DDS 참조서를 참조하십시오.

큰 정수

*큰 정수*는 10 자릿수의 정밀도를 갖는 4바이트(32 비트)로 구성된 2진수입니다. 큰 정수의 범위는 -2147483648 - +2147483647입니다.

큰 정수의 경우 소수 정밀도와 스케일은 COBOL, RPG 및 iSeries 시스템 파일에서 지원됩니다. 2진 정수의 정밀도와 스케일에 관한 내용은 DDS 참조서를 참조하십시오.

큰 정수(BIGINT)

*큰 정수*는 19 자릿수의 정밀도를 갖는 8바이트(64 비트)로 구성된 2진수입니다. 큰 정수의 범위는 -9223372036854775808 - +9223372036854775807입니다.

부동 소수점

단정밀도 부동 소수점 숫자는 실수 대략 32비트를 표시합니다. 크기의 범위는 약 $1.17549436 \times 10^{-38}$에서 $3.40282356 \times 10^{38}$입니다.

배정밀도 부동 소수점 숫자는 실수 대략 IEEE 64 비트를 표시합니다. 크기의 범위는 약 $2.2250738585072014 \times 10^{-308}$ 에서 $1.7976931348623158 \times 10^{308}$입니다.

소수

소수 값은 내재 소수점이 있는 팩 소수 또는 존 소수입니다. 소수점의 위치는 정밀도와 숫자 스케일에 의해 판별됩니다. 숫자의 분수 부분 자릿수인 스케일은 음이 되거나 정밀도 보다 클 수 없습니다. 최대 정밀도는 31 자릿수입니다.

소수 열의 모든 값은 정밀도와 스케일이 같습니다. 소수 변수 또는 소수 열에 있는 숫자의 범위는 *-n* - *+n*이며, 여기서 *n*의 절대값은 적용가능한 정밀도와 스케일로 표시할 수 있는 가장 큰 수입니다. 최대 범위는 음수 $10^{31}+1$ - $10^{31}-1$입니다.

숫자 호스트 변수

작은 정수 및 큰 2진 정수 변수는 모든 호스트 언어에서 사용할 수 있습니다. 큰 정수 변수는 C, C++, ILE COBOL 및 ILE RPG에서만 사용할 수 있습니다. 부동 소수점 변수는 RPG/400 및 COBOL/400을 제외한 모든 호스트 언어에 사용할 수 있습니다. 소수 변수는 지원되는 모든 호스트 언어에 사용할 수 있습니다.

Datetime 값

datetime 값이 특정한 산술 및 스트링 연산에 사용될 수 있고 특정한 스트링과 호환될 수 있더라도 스트링이나 숫자가 아닙니다. 그러나 스트링을 datetime 값으로 나타낼 수 있습니다. 70 페이지의 『Datetime 값의 스트링 표시』를 참조하십시오.

날짜

*날짜*는 A.D. 1년부터 유효했던 것으로 추정되는 그레고리력[16]의 시점을 나타내는 세 부분의 값(년, 월, 일)입니다. 년 부분의 범위는 0001 - 9999입니다. 날짜 형식 *JUL, *MDY, *DMY 및 *YMD는 1940 - 2039 범위의 날짜만 표시할 수 있습니다. 월 부분의 범위는 1 - 12입니다. 일 부분의 범위는 1 - *x*이며 여기서 *x*는 월과 년에 따라 28, 29, 30 또는 31입니다.

날짜의 내부 표시는 정수가 들어 있는 4바이트의 스트링입니다. (Scaliger 숫자라고 하는) 정수는 날짜를 나타냅니다.

SQLDA에 설명한 대로 DATE 열의 길이는 사용되는 형식에 따라 6, 8 또는 10바이트입니다. 값에 대한 문자 스트링 표시에 적절한 길이입니다.

시간

*시간*은 24 시제 시계를 사용하여 하루의 시간을 나타내는 세 부분(시, 분, 초)으로 된 값입니다. 시간 부분의 범위는 0 - 24이고 분과 초 부분의 범위는 0 - 59입니다. 시가 24이면 분과 초 스펙은 모두 0입니다.

시간의 내부 표시는 3바이트의 스트링입니다. 각 바이트는 2개의 팩 소수 자릿수로 구성됩니다. 첫 번째 바이트는 시, 두 번째 바이트는 분 마지막 바이트는 초를 나타냅니다.

SQLDA에 설명한 대로 TIME 열의 길이는 값의 문자 스트링 표시에 적절한 길이인 8바이트입니다.

시간소인

*시간소인*은 마이크로초의 분수 스펙을 포함하는 시간을 제외하고 이전에 정의된 대로 날짜와 시간을 지정하는 7개 부분으로된 값(년 , 월, 일, 시, 분, 초)입니다.

시간소인의 내부 표시는 10바이트 스트링입니다. 처음 4바이트는 날짜, 다음 3바이트는 시간, 마지막 3바이트는 마이크로초를 나타냅니다(마지막 3바이트에는 6 팩자릿수가 들어 있음).

SQLDA에 설명한 대로 TIMESTAMP 열의 길이는 값의 문자 스트링 표시에 적절한 길이인 26바이트입니다.

호스트 변수 결정

문자 스트링 호스트 변수는 보통 날짜, 시간, 시간소인 값을 포함하는 데 사용됩니다. 그러나, 날짜, 시간, 시간소인 호스트 변수는 ILE COBOL 및 ILE RPG에도 지정할 수 있습니다.

16. 역사적인 날짜를 반드시 그레고리력에 따르는 것은 아닙니다. 1582-10-04와 1582-10-15 사이의 날짜는 그레고리력에 없더라도 유효한 날짜로 허용됩니다.

Datetime 값의 스트링 표시

자료 유형이 DATE, TIME 또는 TIMESTAMP인 값은 SQL의 사용자에게 투명한 내부 형식으로 표시됩니다. 그러나 날짜, 시간 및 시간 소인도 문자나 UCS-2 그래픽 스트링으로 표시할 수 있습니다. 이들 표시는 자료 유형이 DATE, TIME 또는 TIMESTAMP인 상수가 없으므로 SQL의 사용자와 직접 관계가 있습니다. ILE RPG와 ILE COBOL만이 datetime 변수를 지원합니다. 검색하기 위해서 character-string 변수에 datetime 값을 할당할 수 있습니다. 결과 스트링의 형식은 명령문이 준비될 때 유효한 디폴트 날짜 형식 및 디폴드 시간 형식에 따라 다릅니다. 디폴트 날짜 및 시간 형식은 날짜 형식(DATFMT), 날짜 분리자(DATSEP), 시간 형식(TIMFMT), 시간 분리자(TIMSEP) 매개변수에 기반한 집합입니다.

datetime 값의 유효한 스트링 표시가 내부 datetime 값 조작에 사용될 때 조작이 수행되기 전에 스트링 표시는 날짜, 시간 또는 시간소인의 내부 형식으로 변환됩니다. 스트링의 CCSID가 외부 코드화 체계를 표시하는 경우(예: ASCII), 스트링이 datetime 값의 내부 형식으로 변환되기 전에 우선 디폴트 CCSID에 의해 식별되는 코드화 문자 세트로 변환됩니다.

다음 섹션은 datetime 값의 유효한 스트링 표시에 대해 정의합니다.

날짜 스트링: 날짜 스트링 표시는 숫자로 시작되고 길이가 최소한 6자인 문자 스트링입니다. 후미 공백이 포함될 수 있습니다. IBM SQL 표준 형식을 사용하는 경우 선행 제로가 월 및 일 부분에서 생략될 수 있습니다. 각 IBM SQL 표준 형식은 이름으로 식별되고 (CHAR 함수가 사용하기 위한) 연관된 약어가 포함됩니다. 다른 형식에는 CHAT 함수에 사용되는 약어가 들어 있지 않습니다. 두 자리 연도 형식의 분리자는 DATSEP(날짜 분리자) 매개변수에 의해 제어됩니다. 날짜에 대해 유효한 스트링 형식은 표 4에 나열됩니다.

다음 중 한 경우일 때 데이터베이스 관리자는 스트링을 날짜로 인식합니다.

- 디폴트 날짜 형식으로 지정된 형식
- IBM SQL 표준 날짜 형식 중 하나
- 형식화되지 않은 율리유스력 형식

표 4. 날짜 스트링 표시에 대한 형식

형식명	약어	날짜 형식	예
국제 표준 기구(*ISO)	ISO	yyyy-mm-dd	1987-10-12
IBM USA 표준(*USA)	USA	mm/dd/yyyy	10/12/1987
IBM 유럽 표준(*EUR)	EUR	dd.mm.yyyy	12.10.1987
일본 산업 표준 Christian Era(*JIS)	JIS	yyyy-mm-dd	1987-10-12
형식화되지 않은 율리유스력	-	yyyyddd	1987285
율리유스력(*JUL)	-	yy/ddd	87/285

표 4. 날짜 스트링 표시에 대한 형식 (계속)

형식명	약어	날짜 형식	예
월, 일, 년(*MDY)	-	mm/dd/yy	10/12/87
일, 월, 년(*DMY)	-	dd/mm/yy	12/10/87
년, 월, 일(*YMD)	-	yy/mm/dd	87/12/10

디폴트 날짜 형식은 다음과 같은 인터페이스로 지정될 수 있습니다.

표 5. 디폴트 날짜 형식 인터페이스

SQL 인터페이스	스펙
삽입 SQL	DATFMT 및 DATSEP 매개변수가 CRTSQLxxx (SQL 프로그램 작성) 명령에 지정되었습니다. SET OPTION문은 삽입 SQL에 들어 있는 프로그램 소스에 내에 DATFMT 및 DATSEP 매개변수를 지정하는 데 사용될 수 있습니다. (CRTSQLxxx 명령에 대한 자세한 정보는 호스트 언어로 SQL 프로그래밍 책을 참조하십시오).
대화식 SQL 및 SQL문 실행	Start SQL (STRSQL) 명령의 DATFMT 및 DATSEP 매개변수 또는 세션 속성을 변경하여. Run SQL (STRSQL) 명령의 DATFMT 및 DATSEP 매개변수. (STRSQL 및 RUNSQLSTM 명령에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.)
서버의 호출 레벨 인터페이스	SQL_ATTR_DATE_FMT 및 SQL_ATTR_DATE_SEP 환경 또는 연결 변수 (CLI에 대한 자세한 정보는 SQL 호출 레벨 인터페이스 (ODBC) 책을 참조하십시오.)
Developer Kit for Java를 사용한 서버의 JDBC 또는 SQLJ	날자 형식 및 날짜 분리자 연결 등록 정보 (JDBC와 SQLJ에 대한 자세한 정보는 iSeries Information Center의 IBM Developer Kit for Java 주제를 참조하십시오).
iSeries Access ODBC 드라이버를 사용하는 클라이언트의 ODBC	ODBC 설정의 고급 서버 옵션의 날짜 형식 및 날짜 분리자 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries 액세스 범주를 참조하십시오.)
IBM Toolbox for Java를 사용하는 클라이언트의 JDBC	JDBC 설정의 형식 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries 액세스 범주를 참조하십시오.) (IBM Toolbox for Java에 대한 자세한 정보는 iSeries Information Center의 IBM Toolbox for Java 주제를 참조하십시오).

시간 스트링: 시간 스트링 표시는 숫자로 시작되고 길이가 최소한 4자인 문자 스트링입니다. 후미 공백이 포함될 수 있고 시간의 시 부분에서 선행 제로가 생략될 수 있으며 초 전체가 생략될 수 있습니다. 초를 생략하도록 선택하면 0초의 내재 스펙이 가정됩니다. 따라서 13.30은 13.30.00과 같습니다.

시간에 대해 유효한 스트링 형식은 72 페이지의 표 6에 나열됩니다. 각 IBM SQL 표준 형식은 이름으로 식별되고 (CHAR 함수가 사용하기 위한) 연관된 약어가 포함됩니

다. 다른 형식(*HMS)에는 CHAT 함수에 사용되는 약어가 들어 있지 않습니다. *HMS 형식의 분리자는 시간 분리자(TIMSEP) 매개변수에 의해 제어됩니다.

다음 중 한 경우일 때 데이터베이스 관리자는 스트링을 시간으로 인식합니다.

- 디폴트 시간 형식으로 지정된 형식
- IBM SQL 표준 시간 형식 중 하나

TIMFMT 및 TIMSEP 매개변수가 CRTSQLxxx, RUNSQLSTM 및 STRSQL 명령에 지정됩니다. SET OPTION 명령문을 사용하여 삽입 SQL이 들어 있는 프로그램의 소스 안에 TIMFMT 및 TIMSEP를 지정할 수 있습니다.

표 6. 시간 스트링 표시에 대한 형식

형식명	약어	시간 형식	예
국제 표준 기구(*ISO)	ISO	hh.mm.ss[17]	13.30.05
IBM USA 표준(*USA)	USA	hh:mm AM 또는 PM	1:30 PM
IBM 유럽 표준(*EUR)	EUR	hh.mm.ss	13.30.05
일본 산업 표준 Christian Era(*JIS)	JIS	hh:mm:ss	13:30:05
시, 분, 초(*HMS)	-	hh:mm:ss	13:30:05

USA 시간 형식에서 시는 12 보다 크지 않아야 하고 00:00 AM과 같이 특수한 경우를 제외하고 0이 될 수 없습니다. 24 시제 시계를 사용하는 경우 USA 형식과 24 시제 시계 사이에서 일치되는 경우는 다음과 같습니다.

표 7. USA 시간 형식

USA 형식	**24** 시제 시계
12:01 AM - 11:59 AM	00.01.00 - 00.59.00
01:00 AM - 11:59 AM	01:00.00 - 11:59.00
12:00 PM(정오) - 11:59 PM	12:00.00 - 23.59.00
12:00 AM(자정)	24.00.00
00:00 AM(자정)	00.00.00

USA 형식에서 하루 시간의 분 부분과 AM 또는 PM 사이에 하나의 간격이 있습니다.

17. ISO 형식의 이전 버전입니다. JIS를 사용하여 현재 ISO 형식을 구할 수 있습니다.

디폴트 시간 형식은 다음과 같은 인터페이스로 지정될 수 있습니다.

표 8. 디폴트 시간 형식 인터페이스

SQL 인터페이스	스펙
삽입 SQL	TIMFMT 및 TIMSEP 매개변수가 CRTSQLxxx(SQL 프로그램 작성) 명령에 지정되었습니다. SET OPTION문은 삽입 SQL에 들어 있는 프로그램 소스에 TIMFMT 및 TIMSEP 매개변수를 지정하는 데 사용될 수 있습니다. (CRTSQLxxx 명령에 대한 자세한 정보는 호스트 언어로 SQL 프로그래밍 책을 참조하십시오).
대화식 SQL 및 SQL문 실행	Start SQL(STRSQL) 명령의 TIMFMT 및 TIMSEP 매개변수 또는 세션 속성을 변경하여. Run SQL (RUNSQLSTM) 명령의 TIMFMT 및 TIMSEP 매개변수. (STRSQL 및 RUNSQLSTM 명령에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.)
서버의 호출 레벨 인터페이스	SQL_ATTR_TIME_FMT 및 SQL_ATTR_TIME_SEP 환경 또는 연결 변수 (CLI에 대한 자세한 정보는 SQL 호출 레벨 인터페이스 (ODBC) 책을 참조하십시오.)
Developer Kit for Java를 사용한 서버의 JDBC 또는 SQLJ	시간 형식 및 시간 분리자 연결 등록 정보 오브젝트 (JDBC와 SQLJ에 대한 자세한 정보는 iSeries Information Center의 IBM Developer Kit for Java 주제를 참조하십시오).
iSeries Access ODBC 드라이버를 사용하는 클라이언트의 ODBC	ODBC 설정의 고급 서버 옵션의 시간 형식 및 시간 분리자 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries Access 범주를 참조하십시오.)
IBM Toolbox for Java를 사용하는 클라이언트의 JDBC	JDBC 설정의 형식 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries Access 범주를 참조하십시오.) (IBM Toolbox for Java에 대한 자세한 정보는 iSeries Information Center의 IBM Toolbox for Java 주제를 참조하십시오).

시간소인 스트링: 시간소인 스트링 표시는 숫자로 시작되고 길이가 최소한 16자인 문자 스트링입니다. 시간소인의 완전한 스트링 표시는 *yyyy-mm-dd-hh.mm.ss.nnnnnn* 또는 *yyyymmddhhmmss* 형식입니다. 후미 공백이 포함될 수 있습니다. 분리자와 함께 시간소인 형식을 사용할 경우 선행 제로가 시간소인의 월, 일 및 시 부분에서 생략될 수 있습니다. 마이크로초에서 후미 제로가 완전히 절단되거나 생략될 수 있습니다. 마이크로 초 부분의 숫자를 생략하도록 선택하면 0의 내재 스텍이 가정됩니다. 따라서 *1990-3-2-8.30.00.10*은 *1990-03-02-08.30.00.100000*과 같습니다.

시간 부분이 24.00.00.000000인 시간소인도 허용됩니다.

DataLink 값

DataLink 값은 데이터베이스에서 외부 데이터베이스가 저장된 파일까지 논리 참조가 들어 있는 캡슐화된 값입니다. 이 캡슐화된 값의 속성은 다음과 같습니다.

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

체계 URL의 경우 HTTP 또는 FILE과 같은 값입니다. 값은 대소문자 입력 여부와 관계없이 대문자로 데이터베이스에 저장됩니다.

파일 서버명
파일 서버의 완전 주소. 값은 대소문자 입력 여부와 관계없이 대문자로 데이터베이스에 저장됩니다.

파일 경로
서버 안의 파일 존재. 값은 대소문자가 구분되므로 데이터베이스에 저장될 때 대문자로 변환되지 않습니다.

액세스 제어 토큰
해당되면 액세스 토큰이 파일 경로 안에 삽입됩니다. 동적으로 생성되며 데이터베이스에 저장되는 Datalink 값의 영구적인 부분이 아닙니다.

주석 최고 254바이트의 서술 정보. 자세한 자료의 위치 또는 대체 ID와 같은 어플리케이션 특정 사용을 위한 것입니다.

DataLink 값에 사용되는 문자는 URL에 정의된 세트로 제한됩니다. 이들 문자에는 대문자(A - Z)와 소문자(a - z), 숫자(0 - 9), 특수 문자($, -, _, @, ., &, +, !, *, ″, ’, (,), =, ;, /, #, ?, :, 간격, 쉼표)의 서브세트 등이 포함됩니다.

처음 4개 속성은 한데 묶어 연결 속성이라고 합니다. DataLink 값에 주석 속성은 포함될 수 있지만 연결 속성은 포함될 수 없습니다. 이런 값은 열에 저장까지 되지만 파일은 해당 열에 연결되지 않습니다.

파일에 대한 DataLink 참조와 118 페이지의 『LOB 파일 참조 변수에 대한 참조』에서 설명한 LOB 파일 참조 변수 사이를 구분하는 것이 중요합니다. 모두 파일 표시가 들어 있다는 점이 비슷합니다. 그러나 다음과 같은 차이가 있습니다.

- DataLink가 데이터베이스에 보유되고 링크와 연결된 파일의 자료 모두 데이터베이스의 자연스러운 자료 부가 제품으로 간주될 수 있습니다.
- 파일 참조 변수가 일시적으로 존재하고 호스트 프로그램 버퍼에 대한 대체로 간주될 수 있습니다.

DataLink 값(DLVALUE)을 설정하고 DataLink 값(DLCOMMENT, DLLINKTYPE, DLURLCOMPLETE, DLURLPATH, DLURLPATHONLY, DLURLSCHEME, DLURLSERVER)에서 캡슐화된 값을 추출하기 위해 내장 스칼라 함수가 제공됩니다.

행 ID 값

*행 ID*는 표의 형을 고유하게 식별하는 값입니다. 열 또는 호스트 변수는 행 ID 자료 유형을 가집니다. ROWID 열을 사용하여 표의 행으로 직접 이동하는 조치를 작성할 수 있습니다. ROWID 열의 값은 고유해야 합니다. 데이터베이스 관리자는 표가 재구성되더라도 값을 영구적으로 보존합니다. 표에 행이 삽입될 때 데이터베이스 관리자는 ROWID 열이 없는 경우 해당 열 값을 생성합니다. 값이 제공될 경우, 그 값은 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에서 이전에 생성된 유효 행 ID 값이어야 합니다.

사용자는 행 ID 값의 내부 표현을 볼 수 있습니다. 값은 CCSID 변환에 의존하지 않는데, BIT 자료를 포함하는 것으로 간주되기 때문입니다. ROWID 행의 길이 속성은 40입니다.

사용자 정의 유형

고유한 유형

*고유한 유형*은 내장된 자료 유형("소스 유형")과 내부자료 표시를 공유하나 분리된 것으로 간주되어 대부분의 조작에 대해 호환되지 않는 사용자 정의 자료 유형입니다. 예를 들어, 모든 내부 표시에 내장된 자료 유형 BLOB를 사용하는 영상 유형, 텍스트 유형 및 오디오 유형의 구문은 상당히 다릅니다. SQL문 CREATE DISTINCT TYPE으로 고유한 유형이 작성됩니다.

예를 들어, 다음 명령문은 AUDIO라는 고유한 유형을 작성합니다.

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

AUDIO가 내장된 자료 유형 BLOB와 같게 표시되더라도 BLOB나 다른 유형과 비교할 수 없는 분리된 유형으로 간주됩니다. AUDIO와 다른 자료 유형을 비교할 수 없으므로 AUDIO에 특정한 함수가 작성되고 이런 함수는 다른 자료 유형에 적용될 수 없습니다.

고유한 유형의 이름은 스키마명으로 규정됩니다. 규정되지 않은 이름에 대한 내재 스키마명은 고유한 유형이 나타나는 문맥에 따라 달라집니다. 규정되지 않은 고유한 유형 이름이 사용되는 경우:

- CREATE DISTINCT TYPE이나 DROP, COMMENT, GRANT 또는 REVOKE 문의 오브젝트에서 데이터베이스 관리자는 권한부여 ID에 의한 규정화의 일반 프로세스를 사용하여 스키마명을 판별합니다. 규정화 규칙에 대한 자세한 내용은 55 페이지의 『규정되지 않은 함수, 프로시듀어, 특정 및 고유한 유형 이름』을 참조하십시오.
- 다른 문맥에서 데이터베이스 관리자는 SQL 경로를 사용하여 스키마명을 판별합니다. 데이터베이스 관리자는 경로에서 순서대로 스키마를 찾아서 일치하는 고유한 유형을

갖는 첫 번째 스키마를 선택합니다. SQL 경로에 대한 설명은 105 페이지의 『CURRENT PATH, CURRENT_PATH 또는 CURRENT FUNCTION PATH』를 참조하십시오.

고유한 유형은 의미가 없으므로 그 소스 유형에 대한 함수와 연산자가 자동으로 사용 지정되지 않습니다(예를 들면, AUDIO 유형의 LENGTH 함수는 오브젝트의 길이를 바이트 수가 아닌 초 수로 리턴합니다). 대신, 고유한 유형은 *강한 유형*을 지원합니다. 강한 유형을 사용하면 고유한 유형에 대해 명시적으로 정의된 함수와 연산자만 그 고유한 유형에 적용될 수 있습니다. 그러나, 소스 유형의 함수 또는 연산자는 해당되는 사용자 정의 함수를 작성하여 고유한 유형에 적용될 수 있습니다. 사용자 정의 함수는 소스 유형이 매개변수인 기존 함수에 근거를 두어야 합니다.

고유한 유형에는 소스 유형과 같은 제한사항이 적용됩니다. 예를 들면, 표는 하나의 ROWID 열만 가질 수 있습니다. 따라서, ROWID 열을 가진 표는 행 ID에서 피소스(sourced)된 고유 유형을 가진 열도 가질 수 없습니다.

Datalink가 소스인 고유한 유형을 제외하고 고유한 유형에 대한 비교 연산자가 자동으로 생성됩니다. 또한 데이터베이스 관리자는 소스 유형에서 고유한 유형으로, 고유한 유형에서 소스 유형으로 캐스트를 지원하는 고유한 유형에 대한 함수를 자동으로 생성합니다. 예를 들어, 위에서 작성된 AUDIO 유형의 경우 다음과 같은 캐스트 함수가 생성됩니다.

```
FUNCTION schema-name.BLOB (schema-name.AUDIO) RETURNS BLOB (1M)

FUNCTION schema-name.AUDIO (BLOB (1M)) RETURNS AUDIO
```

자료 유형의 승격

자료 유형은 관련된 자료 유형의 그룹으로 분류될 수 있습니다. 해당 그룹 내에서 한 자료 유형이 다른 자료 유형 앞에 올 것으로 간주되는 선행 순서가 있습니다. 이런 선행 순서로 데이터베이스 관리자는 선행 순서에서 나중에 표시되는 다른 자료 유형으로 한 자료 유형의 *승격*을 지원할 수 있습니다. 예를 들어, 데이터베이스 관리자는 VARCHAR에 대해 자료 유형 CHAR을, DOUBLE PRECISION에 대해 자료 유형 INTEGER를 승격할 수 있지만 VARCHAR에 대해 CLOB를 승격할 수 없습니다.

데이터베이스 관리자는 다음 경우에 자료의 승격을 고려합니다.

- 함수 분석 수행(123 페이지의 『함수 분석』 참조)
- 고유한 유형 캐스트(77 페이지의 『자료 유형 사이의 캐스트』 참조)
- 내장 자료 유형에 고유한 유형 지정(88 페이지의 『고유한 유형 지정』 참조)

각 자료 유형에 대해, 77 페이지의 표 9에서는 데이터베이스 관리자가 각 자료 유형이 승격될 수 있는 자료 유형을 판별하는 데 사용되는 (순서) 선행 리스트를 보여줍니다. 표에서 최선의 선택은 자료 유형이 같고 다른 자료 유형으로 승격하지 않는 것임을 알

수 있습니다. 또한 표에서 자료 유형이 승격 프로세스 동안 동등한 것으로 간주됨에 유의하십시오. 예를 들면, CHARACTER와 GRAPHIC이 동일한 자료 유형으로 간주됩니다.

표 9. 자료 유형의 선행 순서

자료 유형 *	**(최선-최악의 순서의) 자료 유형 선행 순서 리스트**
CHAR 또는 GRAPHIC	CHAR 또는 GRAPHIC, VARCHAR 또는 VARGRAPHIC, CLOB 또는 DBCLOB
VARCHAR 또는 VARGRAPHIC	VARCHAR 또는 VARGRAPHIC, CLOB 또는 DBCLOB
CLOB 또는 DBCLOB	CLOB 또는 DBCLOB
BLOB	BLOB
SMALLINT	SMALLINT, INTEGER, BIGINT, DECIMAL 또는 NUMERIC, REAL, DOUBLE
INTEGER	INTEGER, BIGINT, DECIMAL 또는 NUMERIC, REAL, DOUBLE
BIGINT	BIGINT, DECIMAL 또는 NUMERIC, REAL, DOUBLE
DECIMAL 또는 NUMERIC	DECIMAL 또는 NUMERIC, REAL, DOUBLE
REAL	REAL, DOUBLE
DOUBLE	DOUBLE
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
ROWID	ROWID
고유한 유형	동일하게 고유한 유형
주: * 나열된 자료 유형에 대한 다른 동의어는 나열된 동의어와 같은 것으로 간주됩니다.	

자료 유형 사이의 캐스트

주어진 자료 유형의 값이 다른 자료 유형이나 길이, 정밀도 또는 스케일이 다른 동일한 자료 유형으로 캐스트(변경)해야 할 경우가 많습니다. 76 페이지의 『자료 유형의 승격』에서 설명한 대로 자료 유형 승격은 하나의 자료 유형을 갖는 값이 새로운 자료 유형으로 캐스트되어야 하는 경우의 예입니다. 다른 자료 유형으로 변경될 수 있는 자료 유형은 소스 자료 유형에서 목표 자료 유형으로 *캐스트할 수 있습니다*.

한 자료 유형에서 다른 자료 유형으로 캐스트는 내재적으로 또는 명시적으로 발생할 수 있습니다. 캐스트 함수나 CAST 스펙을 사용하여 자료 유형을 명시적으로 캐스트할 수 있습니다. 데이터베이스 관리자는 고유한 유형이 포함된 지정을 할 때 내재적으로 자료 유형을 캐스트할 수 있습니다(88 페이지의 『고유한 유형 지정』 참조). 또한 소스 사용자 정의 함수를 작성할 때 소스 함수의 매개변수 자료 유형은 작성중인 함수의 자료 유형으로 캐스트할 수 있어야 합니다(443 페이지의 『CREATE FUNCTION』 참조).

자료 유형 사이의 캐스트

문자나 그래픽 스트링이 다른 자료 유형에 대한 캐스트일 때 절단될 경우 공백이 아닌 문자가 절단되면 경고가 발생합니다. 공백이 아닌 문자가 절단되어 오류가 발생될 때 절단 작동은 목표에 대한 문자나 그랙픽스트링 지정과 같지 않습니다.

서로 캐스트되는 자료 유형으로 고유한 자료 유형이 포함되는 캐스트의 경우 지원된 캐스트는 표 10에서 보여집니다. 내장 자료 유형 간의 캐스트의 경우 지원된 캐스트는 79 페이지의 표 11에서 보여줍니다.

표 10. 고유한 자료 유형이 포함될 때 지원되는 캐스트

자료 유형 ...	자료 유형으로 캐스트할 수 있는...
고유한 유형 DT	고유한 유형 DT의 소스 자료 유형
고유한 유형 DT의 소스 자료 유형	고유한 유형 DT
고유한 유형 DT	고유한 유형 DT
자료 유형 A	A가 고유한 유형 DT의 소스 자료 유형으로 승격될 수 있는 고유한 유형 DT(76 페이지의 『자료 유형의 승격』 참조)
INTEGER	DT 소스 유형이 SMALLINT인 경우 고유한 유형 DT
DOUBLE	DT의 소스 유형이 REAL인 경우 고유한 유형 DT
VARCHAR 또는 VARGRAPHIC	DT 소스 유형이 CHAR 또는 GRAPHIC인 경우 고유한 유형 DT

스키마명으로 명시적으로 규정되지 않은 고유한 유형이 캐스트에 포함되는 경우 데이터베이스 관리자는 SQL 경로를 사용하여 스키마명을 판별합니다. 데이터베이스 관리자는 이름으로 고유한 유형을 포함하는 SQL 경로의 첫 번째 스키마명을 선택합니다. SQL 경로에 대한 자세한 정보는 56 페이지의 『스키마 및 SQL 경로』를 참조하십시오.

다음 표는 자료 유형 사이에 지원되는 캐스트에 대해 설명합니다.

표 11. 내장 자료 유형 사이에서 지원되는 캐스트

소스 자료 유형	SMALLINT INTEGER BIGINT	DECIMAL NUMERIC	REAL DOUBLE	CHAR VARCHAR CLOB	GRAPHIC VARGRAPHIC DBCLOB	DATE	TIME	TIME STAMP	BLOB	ROW ID
SMALLINT	Y	Y	Y	Y	--	--	--	--	--	--
INTEGER	Y	Y	Y	Y	--	--	--	--	--	--
BIGINT	Y	Y	Y	Y	--	--	--	--	--	--
DECIMAL	Y	Y	Y	Y	--	--	--	--	--	--
NUMERIC	Y	Y	Y	Y	--	--	--	--	--	--
REAL	Y	Y	Y	Y	--	--	--	--	--	--
DOUBLE	Y	Y	Y	Y	--	--	--	--	--	--
CHAR	Y	Y	Y	Y	*	Y	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	*	Y	Y	Y	Y	Y
CLOB	Y	Y	Y	Y	*	--	--	--	Y	--
GRAPHIC	--	--	--	Y*	Y	--	--	--	Y	--
VARGRAPHIC	--	--	--	Y*	Y	--	--	--	Y	--
DBCLOB	--	--	--	Y*	Y	--	--	--	Y	--
DATE	--	--	--	Y**	--	Y	--	Y	--	--
TIME	--	--	--	Y**	--	--	Y	Y	--	--
TIMESTAMP	--	--	--	Y**	--	Y	Y	Y	--	--
BLOB	--	--	--	--	--	--	--	--	Y	--
ROWID	--	--	--	Y	--	--	--	--	Y	Y

주: * 변환은 UCS-2 그래픽의 경우에만 지원됩니다.

** DATE, TIME, TIMESTAMP 또는 ROWID에서 CLOB로의 캐스트는 지원되지 않습니다.

DATALINK만 DATALINK 유형으로 캐스트될 수 있습니다.

다음 표에서는 자료 유형에 대한 캐스트 규칙을 설명합니다.

자료 유형 사이의 캐스트

표 12. 자료 유형에 대한 캐스트 규칙

목표 자료 유형	소스 자료 유형	규칙
SMALLINT	임의	SMALLINT 스칼라 함수를 참조하십시오.
INTEGER	임의	INTEGER 스칼라 함수를 참조하십시오.
BIGINT	임의	BIGINT 스칼라 함수를 참조하십시오.
DECIMAL	임의	DECIMAL 스칼라 함수를 참조하십시오.
NUMERIC	임의	ZONED 스칼라 함수를 참조하십시오.
REAL	임의	REAL 스칼라 함수를 참조하십시오.
DOUBLE	임의	DOUBLE 스칼라 함수를 참조하십시오.
CHAR	임의	CHAR 스칼라 함수를 참조하십시오.
VARCHAR	임의	VARCHAR 스칼라 함수를 참조하십시오.
CLOB	임의	CLOB 스칼라 함수를 참조하십시오.
GRAPHIC	임의	호스트 변수에 대한 스트링 지정 규칙을 참조하십시오.
VARGRAPHIC	임의	호스트 변수에 대한 스트링 지정 규칙을 참조하십시오.
DBCLOB	임의	DBCLOB 스칼라 함수를 참조하십시오.
DATE	임의	DATE 스칼라 함수를 참조하십시오.
TIME	임의	TIME 스칼라 함수를 참조하십시오.
TIMESTAMP	CHAR	TIMESTAMP 스칼라 함수를 참조하십시오, 여기서 피연산자가 하나 지정됩니다.
TIMESTAMP	DATE	시간소인은 지정된 날짜와 시간 00:00:00으로 구성됩니다.
TIMESTAMP	TIME	시간소인은 CURRENT_DATE와 지정된 시간으로 구성됩니다.
BLOB	임의	BLOB 스칼라 함수를 참조하십시오.
DATALINK	DATALINK	DataLink 지정 규칙을 참조하십시오.
ROWID	임의	ROWID 스칼라 함수를 참조하십시오.

지정과 비교

SQL의 기본 조작은 지정과 비교입니다. 지정 조작은 CALL, INSERT, UPDATE, FETCH, SELECT, SET 변수 및 VALUES INTO문 실행시 수행됩니다. 비교 조작은 술부와 MAX, MIN, DISTINCT, GROUP BY 및 ORDER BY와 같은 다른 언어 요소를 포함하는 명령문 실행시 수행됩니다.

두 조작의 기본 규칙에 포함되는 피연산자의 자료 유형은 호환되는 것이어야 합니다. 호환성 규칙은 UNION, 연결, CASE 표현식 및 CONCAT, VALUE, COALESCE, IFNULL, MIN 및 MAX 스칼라 함수에도 적용됩니다. 호환성 도표는 다음과 같습니다.

표 13. 자료 유형 호환성

피연산자	2진 정수	십진수[4]	부동 소수점	문자 스트링	그래픽 스트링	2진 스트링	날짜	시간	시간소인	고유한 유형
2진 정수	예	예	예	아니오	아니오	아니오	아니오	아니오	아니오	2
십진수	예	예	예	아니오	아니오	아니오	아니오	아니오	아니오	2
부동 소수점	예	예	예	아니오	아니오	아니오	아니오	아니오	아니오	2
문자 스트링	아니오	아니오	아니오	예	예 5	아니오 3	1	1	1	2
그래픽 스트링	아니오	아니오	아니오	예 5	예	아니오	아니오	아니오	아니오	2
2진 스트링	아니오	아니오	아니오	아니오 3	아니오	예	아니오	아니오	아니오	2
날짜	아니오	아니오	아니오	1	아니오	아니오	예	아니오	아니오	2
시간	아니오	아니오	아니오	1	아니오	아니오	아니오	예	아니오	2
시간소인	아니오	아니오	아니오	1	아니오	아니오	아니오	아니오	예	2
고유한 유형	2	2	2	2	2	2	2	2	2	2

주:

1. 시간소인 값과 문자 스트링의 호환성 규칙은 지정, 비교 및 VALUE, COALESCE, IFNULL, MIN 및 MAX 스칼라 함수에 제한됩니다.
   - Datetime 값은 86 페이지의 『Datetime 지정』에서 설명된 대로 문자 스트링 열과 문자 스트링 변수에 지정될 수 있습니다.
   - 자료의 유효한 스트링 표시는 날짜 열에 지정, 날짜와 비교 또는 날짜와 함께 VALUE, COALESCE, IFNULL, MIN 또는 MAX 스칼라 함수에 사용될 수 있습니다.
   - 시간의 유효한 스트링 표시는 시간 열에 지정, 시간과 비교 또는 시간과 함께 VALUE, COALESCE, IFNULL, MIN 또는 MAX 스칼라 함수에 사용될 수 있습니다.
   - 시간소인의 유효한 스트링 표시는 시간소인 열에 지정, 시간소인과 비교 또는 시간소인과 함께 VALUE, COALESCE, IFNULL, MIN 또는 MAX 스칼라 함수에 사용될 수 있습니다.
2. 고유한 유형을 갖는 값과 동일한 고유한 유형으로 정의된 값만 비교할 수 있습니다. 일반적으로, 데이터베이스 관리자는 고유한 유형 값과 소스 자료 유형 사이에서 지정할 수 있습니다. 자세한 내용은 88 페이지의 『고유한 유형 지정』을 참조하십시오.
3. 부속 유형 FOR BIT DATA 뿐만 아니라 모든 문자 스트링은 2진 스트링과 호환될 수 없습니다.
4. 소수는 팩과 존 소수를 말합니다.
5. 비트 자료와 그래픽 스트링은 호환될 수 없습니다.
6. DATALINK 피연산자만이 다른 DATALINK 피연산자로 지정할 수 있습니다. DATALINK 값은 열이 NO LINK CONTROL로 정의되거나 파일이 존재하여 파일 링크되지 않는 열로만 지정될 수 있습니다. DATALINK 피연산자는 자료 유형과 직접 비교될 수 없습니다. DLCOMMENT, DLLINKTYPE, DLURLCOMPLETE, DLURLPATH, DLURLPATHONLY, DLURLSCHEME 및 DLURLSERVER 스칼라 함수를 사용하여 다른 스트링과 비교할 수 있는 자료 링크로부터 문자 스트링 값을 추출할 수 있습니다.
7. ROWID 피연산자만이 다른 ROWID 피연산자로 지정할 수 있습니다.

지정 조작의 기본 규칙에 따르면 널값을 포함할 수 없는 열과 연관된 인디케이터 변수가 없는 호스트 변수에 널값을 지정할 수 없습니다. 인디케이터 변수에 대한 내용은 114 페이지의 『호스트 변수에 대한 참조』를 참조하십시오.

숫자 지정

숫자 지정 기본 규칙에 따르면 소수 또는 정수의 전부가 절단될 수 없습니다. 필요한 경우 십진수의 분수 부분이 절단됩니다. 호스트 변수에 지정하는 경우 양의 값은 SQLCODE에 리턴될 수 있습니다.

다음 경우 오류가 발생합니다.

- 열에 지정할 때 숫자 전체가 절단되는 경우
- 인디케이터 변수가 없는 호스트 변수에 지정할 때 숫자 전체가 절단되는 경우

다음 경우에 경고가 발생합니다.

인디케이터 변수가 있는 호스트 변수에 지정될 때 숫자 전체가 절단되는 경우. 이때 숫자가 호스트 변수에 지정되지 않고 인디케이터 변수는 -2로 설정됩니다.

주: 소수는 팩과 존 소수를 말합니다.

주: SQL CREATE TABLE문으로 작성되지 않은 파일에서 소수 자료를 페치하는 경우 소수 필드에 유효하지 않은 자료가 포함될 수 있습니다. 이때 경고나 오류 메세지가 발행되지 않고 자료가 저장된 대로 리턴됩니다. SQL CREATE TABLE문으로 작성된 표에는 유효하지 않은 소수 자료가 허용되지 않습니다.

소수 또는 정수에서 부동 소수점

부동 소수점 숫자는 실수의 근사값입니다. 이와 같이 소수 또는 정수가 부동 소수점 열이나 변수에 지정되는 경우 결과는 원래 숫자와 동일하지 않을 수 있습니다.

수신 열이나 변수가 단정밀도(32비트)가 아닌 배정밀도(64비트)로 정의되면 근사값이 더 정확합니다.

부동 소수점이나 소수에서 정수

소수 또는 부동 소수점 수가 2진 정수 열이나 변수로 지정될 때 필요한 경우 숫자는 목표의 정밀도와 스케일로 변환됩니다. 목표의 스케일이 0이면 숫자의 소수 부분이 없어집니다. 필요한 수의 선행 제로가 추가되거나 제거되고 숫자의 분수 부분에서 필요한 수의 후미 제로가 추가되거나 필요한 수의 후미 제로가 제거됩니다.

소수에서 소수

소수를 소수 열이나 변수에 지정되는 경우 필요하면 숫자가 목표의 정밀도와 스케일로 변환됩니다. 필요한 수의 선행 제로가 추가되거나 제거되고 숫자의 분수 부분에서 필요한 수의 후미 제로가 추가되거나 필요한 수의 후미 제로가 제거됩니다.

정수에서 소수

정수가 소수 열이나 변수에 지정되는 경우 숫자가 먼저 임시 소수로 변환된 다음 필요하면 목표의 정밀도와 스케일로 변환됩니다. 정수의 스케일이 0이면 임시 소수의 정밀도가 작은 정수일 경우에는 5,0이고 큰 정수일 경우에는 11,0이며 큰 정수(big integer)의 경우에는 19,0입니다.

부동 소수점에서 소수

부동 소수점 수가 소수 열이나 변수에 지정하는 경우 숫자는 먼저 정밀도 31의 임시 소수로 변환된 다음 필요하면 목표의 정밀도와 스케일로 절단됩니다. 이 변환에서 숫자는 (부동 소수점 산술을 사용하여) 31 소수 자릿수의 정밀도로 반올림됩니다. 따라서 $0.5*10^{-31}$보다 작은 숫자는 0이 됩니다. 스케일에 가능한 가장 큰 값이 지정되어 크게 유실되지 않고 숫자 전체가 표시될 수 있도록 합니다.

COBOL과 RPG 정수로

COBOL 및 RPG 작은 정수 또는 큰 정수 호스트 변수에 할당할 때 호스트 변수에 지정된 모든 스케일을 고려합니다. 그러나 정수 호스트 변수로 지정할 때는 정수의 전체 크기를 사용합니다. 따라서 COBOL 자료 항목이나 RPG 필드의 값은 호스트 변수에 지정된 최대 정밀도 보다 클 수 있습니다.

예를 들어, COBOL에서 COL1에 값 12345가 들어 있으면 명령문은 다음과 같습니다.

```
01  A  PIC  S9999  BINARY.
EXEC SQL SELECT COL1
         INTO :A
         FROM TABLEX
END-EXEC.
```

A가 4 자리로만 정의되었더라도 값 12345가 A에 들어갑니다.

다음 COBOL 명령문을 살펴 봅시다.

```
MOVE 12345 TO A.
```

A에 2345가 들어갑니다.

스트링 지정

2진 스트링 지정

2진 스트링 지정에는 다음의 두 가지 유형이 있습니다.

- *기억장치 할당*은 값이 열이나 함수 또는 저장 프로시듀어의 매개변수에 할당될 때입니다.
- *검색 할당*은 값이 호스트 변수에 할당될 때입니다.

기억장치 할당: 기본 규칙은 열이나 함수 또는 프로시듀어의 매개변수에 할당되는 스트링의 길이가 해당 열 또는 매개변수의 길이 속성보다 크지 않아야 한다는 점입니다.

스트링이 해당 열의 길이 속성보다 더 긴 경우 음수 SQLCODE가 리턴됩니다. SQLCA의 설명은 871 페이지의 부록 B『SQL 통신 영역』을 참조하십시오.

검색 할당: 호스트 변수에 할당되는 스트링의 길이는 호스트 변수의 길이 속성보다 클 수 있습니다. 변수에 스트링이 지정되고 스트링이 변수의 길이 속성 보다 길면 스트링의 오른쪽이 필요한 문자 수로 절단됩니다. 이럴 경우에는 SQLCA의 SQLWARN1 필드에 값 'W'가 지정됩니다.

길이가 *n*인 스트링이 최대 길이가 *n*보다 큰 가변 길이 스트링 변수에 지정된 경우 이 변수의 *n*번째 바이트 이후에 나오는 바이트들은 정의되지 않습니다.

문자 및 그래픽 스트링 지정

소스와 목표 모두가 스트링이면 위의 규칙이 적용됩니다. datetime 자료 유형이 포함되면 86 페이지의『Datetime 지정』을 참조하십시오.

문자 및 그래픽 스트링 지정에는 다음의 두 가지 유형이 있습니다.

- *기억장치 할당*은 값이 열이나 함수 또는 저장 프로시듀어의 매개변수에 할당될 때입니다.
- *검색 할당*은 값이 호스트 변수에 할당될 때입니다.

기억장치 할당: 기본 규칙은 열이나 함수 또는 프로시듀어의 매개변수에 할당되는 스트링의 길이가 해당 열 또는 매개변수의 길이 속성보다 크지 않아야 한다는 점입니다. 스트링이 해당 열의 길이 속성보다 더 긴 경우 음수 SQLCODE가 리턴됩니다(후미 공백은 대개 스트링의 길이에 포함됩니다. 그러나 기억장치 할당의 경우 후미 공백이 스트링 길이에 포함되지 않습니다).

SQLCA의 설명은 871 페이지의 부록 B『SQL 통신 영역』을 참조하십시오.

스트링이 고정 길이 스트링 열이나 매개변수에 할당되고 스트링 길이가 목표의 길이 속성보다 작은 경우 스트링은 필요한 수의 1바이트, 2바이트 또는 UCS-2 공백으로 채워집니다.[18] 비트 자료의 경우에도 채움 문자는 항상 공백입니다.

검색 할당: 호스트 변수에 할당되는 스트링의 길이는 호스트 변수의 길이 속성보다 클 수 있습니다. 변수에 스트링이 지정되고 스트링이 변수의 길이 속성 보다 길면 스트링의 오른쪽이 필요한 문자 수로 절단됩니다. 이럴 경우에는 SQLCA의 SQLWARN1 필드에 값 'W'이 지정됩니다. 또한 인디케이터 변수가 제공되면 스트링의 원래 길이에 설정됩니다. C NULL 종료 호스트 변수에 대해 널(null) 종료자만 절단되고 CRTSQLCI 또는 CRTSQLCPPI 명령에 *NOCNULRQD(또는 SET OPTION 명령문에 CNULRQD(*NO))를 지정하면 'N'의 값이 SQLCA의 SQLWARN1 필드에 지정되고 변수에 NUL이 배치되지 않습니다.

18. UCS-2는 코드점 X‘0020’과 X‘3000’에서 공백 문자를 정의합니다. 데이터베이스 관리자는 코드점 X‘0020’에 공백으로 채웁니다.

스트링이 고정 길이 변수에 할당되고 스트링 길이가 목표의 길이 속성보다 작은 경우 스트링은 1바이트, 2바이트 또는 UCS-2 공백의 필요한 수로 오른쪽에 채워집니다.[18] 비트 자료의 경우에도 채움 문자는 항상 공백입니다.

n 길이의 스트링이 최대 길이 *n* 보다 큰 가변 길이 스트링 변수에 지정되면 변수의 *n* 번째 문자 다음 문자는 정의되지 않습니다.

혼합 스트링을 포함한 할당: 스트링에 혼합 자료가 들어 있으면 할당 규칙은 2바이트 코드의 순서 내에서 절단이 필요할 수 있습니다. 2바이트 순서를 종료하는 SI 문자를 유실하지 않도록 추가 문자가 스트링의 끝에서 절단되고 SI 문자가 추가될 수 있습니다. 절단되면 각각의 SO 문자와 대응하는 SI 문자 사이에 항상 짝수의 바이트가 있습니다.

C NUL 종료 스트링을 포함하는 할당: *n*인 길이의 스트링이 *n+1* 보다 큰 C NUL 종료 스트링 변수에 지정되는 경우:

- CRTSQLCI 또는 CRTSQLCPPI 명령에 *CNULRQD 옵션(또는 SET OPTION 명령문에 CNULRQD(*YES))을 지정하면 스트링의 오른쪽에 x-n-1 공백으로 채워지며, 여기서 x는 변수의 길이입니다. 채워진 스트링이 변수에 지정되고 널(null) 종료자는 다음 문자 위치에 배치됩니다.
- CRTSQLCI 또는 CRTSQLCPPI 명령에 *NOCNULRQD 사전컴파일러 옵션(또는 SET OPTION 명령문에 CNULRQD(*NO))을 지정하면 스트링의 오른쪽에 채워지지 않습니다. 스트링이 변수에 지정되고 널(null) 종료자는 다음 문자 위치에 배치됩니다.

지정을 위한 변환 규칙: 필요하면 열이나 호스트 변수에 지정되는 스트링은 우선 목표의 코드화 문자 세트로 변환됩니다. 다음이 모두 참일 경우에만 문자 변환이 필요합니다.

- CCSID가 다른 경우
- 두 CCSID가 모두 65535가 아닌 경우
- 스트링이 널이 아니고 비어 있지도 않은 경우
- CCSID 변환 선택 표가 변환 필요성을 나타낼 경우

다음 경우 오류가 발생합니다.

- CCSID 변환 선택 표가 사용되지만 CCSID 쌍에 대한 정보는 들어 있지 않는 경우
- 스트링 문자가 변환될 수 없고 조작이 열에 대한 지정이거나 인디케이터없이 호스트 변수에 대한 지정인 경우 예를 들어, DBCS(2바이트 문자)는 SBCS(1바이트 문자) CCSID가 있는 열이나 호스트 변수로 변환될 수 없습니다.

다음 경우에 경고가 발생합니다.

- 스트링 문자가 대체 문자로 변환되는 경우

- 스트링 문자가 변환될 수 없고 조작이 인디케이터 변수없이 호스트 변수에 대한 지정인 경우. 예를 들면, DBCS 문자는 SBCS CCSID가 있는 호스트 변수로 변환될 수 없습니다. 이 경우 스트링은 호스트 변수에 지정되지 않고 인디케이터 변수는 -2로 설정됩니다.

Datetime 지정

DATE 열에 지정된 값은 날짜나 날짜의 유효한 스트링 표시여야 합니다. 날짜는 DATE 열, 문자 스트링 열, 문자 스트링 변수 또는 ILE RPG/400 시간소인 변수에만 지정될 수 있습니다. TIME 열에 지정된 값은 시간이나 시간의 유효한 스트링 표시여야 합니다. 시간은 TIME 열, 문자 스트링 열, 문자 스트링 변수 또는 ILE RPG/400 시간소인 변수에만 지정될 수 있습니다. TIMESTAMP 열에 지정된 값은 시간소인이나 시간소인의 유효한 스트링 표시여야 합니다. 시간소인은 TIMESTAMP 열, 문자 스트링 열, 문자 스트링 변수 또는 ILE RPG/400 시간소인 변수에만 지정될 수 있습니다.

datetime 값이 문자 스트링 변수 또는 열에 지정되면 스트링 표시로 변환됩니다. 날짜, 시간 또는 시간소인 부분에서 선행 제로가 생략되지 않습니다. 목표에 필요한 길이는 스트링 표시의 형식에 따라 달라집니다. 목표의 길이가 필요한 것 보다 크면 오른쪽이 공백으로 채워집니다. 목표 길이가 필요한 것 보다 작으면 결과는 포함된 datetime 값의 유형과 목표 유형에 따라 달라집니다.

- 목표가 문자 스트링 열이면 절단이 허용되지 않습니다. 다음 규칙이 적용됩니다.

  DATE

  날짜 형식이 *ISO, USA, *EUR 또는 *JIS이면 열의 길이 속성이 적어도 10이어야 합니다. 날짜 형식이 *YMD, *MDY 또는 *DMY이면 열의 길이 속성이 적어도 8이어야 하고, 날짜 형식이 *JUL이면 호스트 변수의 길이가 적어도 6이어야 합니다.

  TIME

  열의 길이 속성은 적어도 8이어야 합니다.

  TIMESTAMP

  열의 길이 속성은 적어도 26이어야 합니다.

- 목표가 호스트 변수이면 다음 규칙이 적용됩니다.

  DATE

  날짜 형식이 *ISO, USA, *EUR 또는 *JIS이면 호스트 변수의 길이 속성이 적어도 10 이어야 합니다. 날짜 형식이 *YMD, *MDY 또는 *DMY이면 호스트 변수의 길이 속성은 적어도 8이어야 합니다. 날짜 형식이 *JUL이면 호스트 변수의 길이는 적어도 8이어야 합니다.

  TIME

– *USA 형식이 사용되면 호스트 변수의 길이가 8 이상이어야 합니다. 이 형식에는 초가 포함되지 않습니다.

– *ISO, *EUR, *JIS 또는 *HMS 시간 형식이 사용되면 호스트 변수의 길이가 5 이상이어야 합니다. 길이가 5, 6 또는 7이면 시간의 초 부분이 결과에서 생략되고 SQLWARN1이 'W'로 설정됩니다. 이 경우 시간의 초 부분이 제공되면 인디케이터 변수에 지정되고, 길이가 6 또는 7이면 값이 유효한 시간 스트링 표시가 되도록 공백으로 채워집니다.

TIMESTAMP

호스트 변수의 길이가 19이상이어야 합니다. 길이가 19와 25 사이이면 시간 소인은 마이크로초 부분의 하나 이상의 숫자가 생략되면서, 스트링과 절단됩니다. 길이가 20이면 값이 시간소인의 유효한 스트링 표시가 되도록 후미 소수점이 공백으로 대체됩니다.

DataLink 지정

DataLink 열에 값을 지정하면 값의 연결 속성이 비어 있거나 열이 NO LINK CONTROL로 정의되지 않는 한 파일에 대한 링크가 설정됩니다. 연결된 값이 열에 이미 존재하면 그 파일은 링크되지 않습니다. 연결된 값이 이미 존재하는 위치에 널값을 지정해도 기존 값과 연관된 파일이 링크되지 않습니다.

어플리케이션이 열에 이미 존재하는 것과 같은 자료 위치를 제공하면 링크가 보유됩니다. 그 이유는 두 가지가 있습니다.

- 주석이 변경되지 않음
- 표가 링크 지연 중 상태에 있으면 표의 링크가 열의 링크와 동일한 링크 속성을 제공하여 표의 링크가 회복될 수 없음

DLVALUE 스칼라 함수를 사용하여 DataLink 값을 열에 지정할 수 있습니다. DLVALUE 스칼라 함수는 열에 지정될 수 있는 새로운 DataLink 값을 작성합니다. 값이 주석만 포함하지 않거나 URL이 정확하게 같지 않는 한 지정되어 파일에 연결됩니다.

DataLink 열에 값을 지정할 경우 다음 오류 조건이 발생할 수 있습니다.

- URL(자료 위치) 형식이 유효하지 않음
- 이 데이터베이스에 파일 서버가 등록되어 있지 않음
- 유효하지 않은 링크 유형이 지정됨
- 주석 또는 URL의 길이가 유효하지 않음

  URL 매개변수나 함수 결과의 크기가 입/출력 모두에 대해 같고 DataLink 열의 길이로 바인드됨에 유의하십시오. 그러나 어떤 경우에는 리턴되는 URL 값은 액세스 토큰을 접속합니다. 이때 출력 위치는 액세스 토큰과 DataLink 열의 길이에 충분한

기억장치 공간이 있어야 합니다. 따라서 입력시 제공되는 전체 확장 형식의 주석과 URL의 실제 길이는 출력 기억장치 공간을 수용할 수 있도록 제한되어야 합니다. 제한 길이가 초과되면 오류가 발생합니다.

지정할 때 링크가 작성되어도 다음 오류가 발생합니다.

- 현재 파일 서버를 사용할 수 없음
- 파일이 없음
- 참조된 파일을 연결하기 위해 액세스할 수 없음
- 파일이 이미 다른 열에 연결되어 있음

  다른 관계형 데이터베이스로 연결되어도 해당 오류가 발생함에 유의하십시오.

또한 지정할 때 기존 링크가 제거되면 다음 오류가 발생합니다.

- 현재 파일 서버를 사용할 수 없음
- 참조 무결성 제어가 있는 파일은 DB2 DataLink 파일 관리자에 따라 올바른 상태에 있지 않음

DataLink 값은(DLLINKTYPE 및 DLURLPATH와 같은) 스칼라 함수를 사용하여 데이터베이스에서 검색될 수 있습니다. 그런 다음 스칼라 함수의 결과가 호스트 변수로 지정될 수 있습니다.

보통 검색시에 파일 서버에 액세스를 시도하지 않음에 유의하십시오.[19] 따라서 파일 시스템 명령으로 파일 서버를 액세스하려는 후속 시도가 실패할 수 있습니다.

표가 링크 지연 상태에 있으므로 DataLink 값을 검색할 때 경고가 리턴될 수 있습니다.

행 ID 지정

행 ID 값은 행 ID 자료 유형을 가진 열, 매개변수 또는 호스트 변수에만 지정할 수 있습니다. ROWID 열의 값에 대해, 열은 GENERATED BY DEFAULT로 정의되거나 OVERRIDING SYSTEM VALUE를 지정해야 합니다. 모든 ROWID 값이 고유하다고 보장할 수 있는 column ROWID 열을 가진 모든 표에 고유 제한조건이 암시적으로 추가됩니다. 열에 대해 지정된 값은 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에서 이전에 생성된 유효 행 ID 값이어야 합니다.

고유한 유형 지정

고유한 유형을 호스트 변수에 지정하는 데 적용되는 규칙은 고유한 유형이 포함되는 다른 모든 지정 규칙과 다릅니다.

---

19. 경로와 연관된 접두부명을 판별하려면 파일 서버를 액세스하여야 합니다. 이것은 파일 시스템의 마운트 점이 이동될 때 파일 서버에서 변경될 수 있습니다. 서버에서 처음 파일을 액세스하면 파일 서버에서 필요한 값이 검색되고 그 파일 서버에 대한 DataLink 값의 후속 검색을 위해 데이터베이스 서버에서 캐쉬됩니다. 파일 서버를 액세스할 수 없으면 오류가 리턴됩니다.

호스트 변수에 지정

고유한 유형을 호스트 변수에 지정할 때는 고유한 유형의 소스 자료 유형에 기준을 둡니다. 따라서 고유한 유형의 소스 자료 유형이 호스트 변수에 지정될 수 있는 경우에만 고유한 유형의 값이 호스트 변수에 지정될 수 있습니다.

예: 고유한 유형 AGE가 다음 SQL문으로 작성되고 표 STUDENTS의 열 STU_AGE가 고유한 유형으로 정의되었다고 가정합니다. CL_SCHED 표를 사용하여 오늘 나중에 시작하는(STARTING) 모든 클래스(CLASS_CODE)를 선택합니다. 오늘 클래스의 DAY 열에 값 3이 있습니다.

```
CREATE DISTINCT TYPE AGE AS SMALLINT WITH COMPARISONS
```

다음, INTEGER 자료 유형인 호스트 변수 HV_AGE에 유효한 학생의 나이를 지정한다고 가정합니다.

```
SELECT  STU_AGE INTO :HV_AGE FROM STUDENTS WHERE STU_NUMBER = 200
```

고유한 유형(SMALLINT)의 소스 자료 유형이 호스트 변수(INTEGER)에 지정될 수 있으므로 고유한 유형의 값은 호스트 변수에 HV_AGE로 지정될 수 있습니다. 고유한 유형 AGE가 CHAR(5)와 같은 문자 자료 유형에 대해 소스가 된 경우 문자 유형이 정수 유형으로 지정될 수 없으므로 위의 지정은 유효하지 않습니다.

호스트 변수가 아닌 다른 지정

고유한 유형은 지정의 소스나 목표가 될 수 있습니다. 지정은 지정될 값의 자료 유형이 목표의 자료 유형에 캐스트될 수 있는지의 여부에 따라 달라집니다. 77 페이지의 『자료 유형 사이의 캐스트』는 고유한 유형이 포함되는 경우 지원되는 캐스트를 나타냅니다. 따라서 다음 경우에 고유한 유형 값을 호스트 변수가 아닌 다른 목표에 지정할 수 있습니다.

- 지정 목표가 같은 고유한 유형을 갖는 경우 또는
- 고유한 유형이 목표의 자료 유형으로 캐스트될 수 있는 경우

다음 경우에 값이 고유한 유형으로 지정될 수 있습니다.

- 지정되는 값이 목표와 같은 고유한 유형을 갖는 경우 또는
- 지정된 값의 자료 유형이 목표 고유한 유형에 캐스트될 수 있는 경우

예: 고유한 유형 AGE의 소스 자료 유형이 SMALLINT라고 가정합니다.

```
CREATE DISTINCT TYPE AGE AS SMALLINT WITH COMPARISONS
```

그런 다음, 두 개의 표 TABLE1과 TABLE2가 4개의 식별한 열 설명으로 작성되었다고 가정합니다.

```
AGECOL    AGE
SMINTCOL  SMALLINT
INTCOL    INTEGER
DECCOL    DEC(6,2)
```

다음 SQL문을 사용하고 X와 Y에 대한 여러 값을 대체하여 TABLE2에서 TABLE1의 여러 열로 값을 삽입하면 표 14는 지정이 유효한지를 나타냅니다.

```
INSERT INTO  TABLE1 (Y) SELECT X FROM TABLE2
```

표 14. 지정 유형별 결과(INSERT일 때의 예)

TABLE2.X	TABLE1.Y	유효	이유
AGECOL	AGECOL	예	소스와 목표가 같은 고유한 유형임
SMINTCOL	AGECOL	예	(AGE 소스 유형이 SMALLINT이므로) SMALLINT가 AGE로 캐스트될 수 있음
INTCOL	AGECOL	예	(AGE 소스 유형이 SMALLINT이므로) INTEGER는 AGE로 캐스트될 수 있음
DECCOL	AGECOL	아니오	DECIMAL이 AGE로 캐스트될 수 없음
AGECOL	SMINTCOL	예	AGE가 소스 유형 SMALLINT로 캐스트될 수 있음
AGECOL	INTCOL	아니오	AGE가 INTEGER로 캐스트될 수 있음
AGECOL	DECCOL	아니오	AGE가 DECIMAL로 캐스트될 수 있음

숫자 비교

숫자는 대수적으로 즉 부호에 대해 비교됩니다. 예를 들어, -2는 +1 보다 작습니다.

한 숫자가 정수이고 다른 한 숫자가 소수이면 소수로 변환된 정수의 임시 복사본으로 비교합니다.

소수 또는 스케일이 0이 아닌 2진수를 다른 스케일과 비교할 경우 분수 부분이 다른 숫자와 자릿수가 같도록 후미 제로로 확장된 숫자 중 하나의 임시 복사본을 사용하여 비교합니다.

한 숫자가 부동 소수점이고 다른 한 숫자가 정수, 소수 또는 단정밀도 부동 소수점이면 배정밀도 부동 소수점으로 변환된 두 번째 숫자의 임시 복사본으로 비교합니다. 그러나, 단정밀도 부동 소수점 열이 상수와 비교되고 상수가 단정밀도 부동 소수점 숫자로 표시될 수 있으면 단정밀도 형식의 상수와 비교됩니다.

일반화된 형식의 비트 구성이 동일한 경우에만 두 개의 부동 소수점 숫자가 같습니다.

스트링 비교

2진 스트링 비교

2진 스트링 비교는 항상 *HEX의 정렬 순서를 사용하여 각 스트링의 해당 바이트가 비교됩니다. 또한 두 개의 2진 스트링은 두 스트링의 길이가 동일한 경우에만 같습니다.

문자와 그래픽 스트링 비교

모든 SBCS 자료와 혼합 자료의 1바이트 부분에 대한 명령문이 실행되는 경우 문자와 UCS-2 그래픽 스트링 비교는 유효한 정렬 순서를 사용합니다. 정렬 순서가 *HEX이면

각 스트링의 해당 바이트가 비교됩니다. 다른 모든 정렬 순서의 경우 각 스트링 가중치의 해당 바이트가 비교됩니다. 스트링의 길이가 다르면 비교되기 전에 더 짧은 스트링의 임시 복사본의 오른쪽이 공백으로 채워집니다. 채워져 각 스트링의 길이가 같아집니다. 정렬 순서에 관계없이 채움 문자는 항상 공백입니다. 비트 자료인 경우에도 채움 문자는 공백입니다. DBCS 그래픽 자료의 경우 채움 문자는 DBCS 공백입니다(x'4040'). UCS-2 그래픽 자료일 경우 채움 문자는 UCS-2 공백입니다.[20]

다음 조건이 참이면 두 개의 스트링이 같습니다.

- 두 스트링이 모두 비어 있습니다.
- *HEX 정렬 순서가 사용되고 모든 해당하는 바이트는 같습니다.
- *HEX 이외의 정렬 순서가 사용되고 가중치의 해당되는 모든 바이트가 같습니다.

빈 스트링은 공백 스트링과 같습니다. 같지 않은 두 스트링 사이의 관계는 스트링의 왼쪽 끝에서 첫 번째 같지 않은 바이트(또는 가중치 바이트) 쌍을 비교하여 판별됩니다. 명령문이 실행될 때 유효한 정렬 순서에 따라 비교됩니다.

길이가 다른 두 개의 가변 길이 스트링은 후미 공백의 수가 다른 경우에만 같습니다. 값 세트에서 하나의 값을 선택하는 조작에서 값은 임의로 선택됩니다. 임의 선택에 포함될 수 있는 조작은 DISTINCT, MAX, MIN, UNION 및 열 그룹에 대한 참조입니다. 열 그룹 참조에 포함되는 임의 선택에 대한 자세한 내용은 GROUP BY 설명을 참조하십시오.

비교를 위한 변환 규칙: 두 개의 스트링이 비교되는 경우 필요하면 우선 스트링 중 하나가 다른 스트링의 코드화 문자 세트로 변환됩니다. 다음이 모두 참일 경우에만 문자 변환이 필요합니다.

- 두 스트링의 CCSID가 다른 경우
- 두 CCSID가 모두 65535가 아닌 경우
- 변환을 위해 선택된 스트링이 널도 아니고 비어 있지도 않을 경우
- CCSID 변환 선택 표가 변환 필요성을 나타낼 경우

코드화 체계가 다른 두 스트링을 비교하고 피연산자의 유형이 같으면 다음과 같이 필요한 변환이 스트링에 적용됩니다.

표 15. 문자 변환을 위한 코드화 체계 선택

	두 번째 피연산자			
첫 번째 피연산자	**SBCS 자료**	**DBCS 자료**	**혼합 자료**	**UCS-2 자료**
SBCS 자료	아래 참조	두 번째	두 번째	두 번째
DBCS 자료	첫 번째	아래 참조	두 번째	두 번째
혼합 자료	첫 번째	첫 번째	아래 참조	두 번째

20. UCS-2는 코드점 X'0020'과 X'3000'에서 공백 문자를 정의합니다. 데이터베이스 관리자는 코드점 X'0020'에서 공백으로 채웁니다.

표 15. 문자 변환을 위한 코드화 체계 선택 (계속)

	두 번째 피연산자			
첫 번째 피연산자	**SBCS 자료**	**DBCS 자료**	**혼합 자료**	**UCS-2 자료**
UCS-2 자료	첫 번째	첫 번째	첫 번째	아래 참조

그렇지 않으면 변환을 위해 선택된 스트링은 각 피연산자의 유형에 따라 달라집니다. 다음 표는 피연산자 유형이 주어질 때 변환을 위해 선택되는 피연산자를 나타냅니다.

표 16. 문자 변환을 위한 피연산자 선택

첫 번째 피연산자	두 번째 피연산자				
	열 값	**파생된 값**	**특수 레지스터**	**상수**	**호스트 변수**
열 값	두 번째	두 번째	두 번째	두 번째	두 번째
파생된 값	첫 번째	두 번째	두 번째	두 번째	두 번째
특수 레지스터	첫 번째	첫 번째	두 번째	두 번째	두 번째
상수	첫 번째	첫 번째	첫 번째	두 번째	두 번째
호스트 변수	첫 번째	첫 번째	첫 번째	첫 번째	두 번째

외부 코드화 체계에 자료가 들어 있는 호스트 변수는 조작에 사용되기 전에 항상 고유 코드화 체계로 변환됩니다. 위의 규칙은 변환이 이미 발생된 것으로 가정합니다.

스트링 문자가 변환될 수 없거나 CCSID 변환 선택 표가 사용되지만 CCSID 쌍에 대한 정보는 들어 있지 않는 경우 오류가 발생합니다. 스트링 문자가 대체 문자로 변환되는 경우 경고가 발생합니다.

Datetime 비교

DATE, TIME 또는 TIMESTAMP 값은 같은 자료 유형의 다른 값이나 해당 자료 유형의 스트링 표시로 비교될 수 있습니다. 모든 비교는 연대순이므로 시점이 0001년 1월 1일에서 멀수록 시점의 값은 *커집니다*.

TIME 값과 시간 값의 스프링 표시를 포함하는 비교에는 항상 초가 포함됩니다. 스트링 표시에 초가 생략되면 0초가 내재됩니다. 시간 24:00:00은 시간 00:00:00보다 큰 시간과 비교됩니다.

TIMESTAMP 값이 포함되는 비교는 동일하게 간주되는 표시와 관계없이 연대순입니다. 따라서 다음 전제는 참입니다.

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

고유한 유형 비교

고유한 유형의 값은 같은 고유한 유형의 다른 값에만 비교될 수 있습니다.

예를 들어, 고유한 유형 YOUTH와 표 CAMP_DB2_ROSTER 표가 다음 SQL문으로 작성되었다고 가정합니다.

```
CREATE DISTINCT TYPE YOUTH AS INTEGER WITH COMPARISONS

CREATE TABLE CAMP_DB2_ROSTER
  ( NAME                VARCHAR(20),
    ATTENDEE_NUMBER     INTEGER NOT NULL,
    AGE                 YOUTH,
    HIGH_SCHOOL_LEVEL   YOUTH)
```

AGE와 HIGH_SCHOOL_LEVEL이 같은 고유한 유형이므로 다음 비교가 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, 캐스트 함수나 CAST 스펙을 사용하여 고유한 유형과 소스 유형 사이에서 캐스트하여 AGE를 ATTENDEE_NUMBER와 비교할 수 있습니다. 다음 모든 비교가 유효합니다.

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

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

SELECT * FROM CAMP_DB2_ROSTER
  WHERE INTEGER(AGE) > ATTENDEE_NUMBER

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

결과 자료 유형에 대한 규칙

결과의 자료 유형은 조작에서 피연산자에 적용되는 규칙에 의해 판별됩니다. 이 섹션에서는 해당 규칙에 대해 설명합니다.

이 규칙은 다음에 적용됩니다.

- UNION 또는 UNION ALL 조작에서 해당되는 열
- CASE 표현식의 결과표현식
- 스칼라 함수 COALESCE, IFNULL, MAX, MIN 및 VALUE의 인수
- IN 전제의 IN 리스트 표현식 값

결과의 자료 유형은 피연산자의 자료 유형에 따라 판별됩니다. 처음 두 피연산자의 자료 유형으로 중간 결과 자료 유형이 판별되며, 이 자료 유형과 다음 피연산자의 자료 유형으로 새로운 중간 결과 자료 유형이 판별되며 이런 방식으로 계속 처리됩니다. 마

지막 중간 결과 자료 유형과 마지막 피연산자의 자료 유형으로 결과의 자료 유형이 판별됩니다. 각 쌍의 자료 유형에 대해 결과 자료 유형은 다음 표에 요약된 규칙의 순차적 어플리케이션에 의해 판별됩니다.

두 피연산자 모두 널이 허용되지 않으면 결과도 널이 허용되지 않습니다. 그렇지 않은 경우 결과에 널이 허용됩니다. 피연산자 열의 설명이 결과의 설명과 같지 않으면 결과의 설명에 맞도록 값이 변환됩니다.

변환 조작이 마치 값의 결과에 지정된 것처럼 정확하게 같습니다. 예를 들면, 다음과 같습니다.

- 한 피연산자 열이 CHAR(10)이고 다른 피연산자 열이 CHAR(5)이면 결과는 CHAR(10)이고 CHAR(5)에서 파생된 값은 오른쪽에 5개의 공백으로 채워집니다.
- 숫자의 전체 부분이 보존될 수 없으면 오류가 발생합니다.

2진 스트링 피연산자

BLOB(2진 스트링)은 다른 BLOB(2진 스트링)에만 호환될 수 있습니다. 결과의 자료 유형은 BLOB입니다. 자료 유형을 BLOB로 캐스트하기 위해 다른 자료 유형은 BLOB 스칼라 함수를 사용하여 BLOB 자료 유형으로 처리될 수 있습니다. 결과 BLOB의 길이는 모든 자료 유형 중에서 가장 깁니다.

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
BLOB(x)	BLOB(y)	BLOB(z) where z = max(x,y)

문자와 그래픽 스트링 피연산자

문자 및 그래픽 스트링은 해당 CCSID 간에 정의된 변환이 있을 때 다른 문자 및 그래픽 스트링과 호환됩니다.

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
DBCLOB(x)	CHAR(y), VARCHAR(y), CLOB(y), GRAPHIC(y), VARGRAPHIC(y) 또는 DBCLOB(y)	DBCLOB(z) 여기에서 z = max(x,y)
CLOB(x)	GRAPHIC(y) 또는 VARGRAPHIC(y)	DBCLOB(z) 여기에서 z = max(x,y)
VARGRAPHIC(x)	VARGRAPHIC(y) 또는 GRAPHIC(y) 또는 VARCHAR(y) 또는 CHAR(y)	VARGRAPHIC(z) 여기에서 z = max(x,y)
VARCHAR(x)	GRAPHIC(y)	VARGRAPHIC(z) 여기에서 z = max(x,y)
GRAPHIC(x)	GRAPHIC(y) 또는 CHAR(y)	GRAPHIC(z) 여기에서 z = max(x,y)
CLOB(x)	CLOB(y) 또는 VARCHAR(y) 또는 CHAR(y)	CLOB(z) 여기에서 z = max(x,y)
VARCHAR(x)	VARCHAR(y) 또는 CHAR(y)	VARCHAR(z) 여기에서 z = max(x,y)
CHAR(x)	CHAR(y)	CHAR(z) 여기에서 z = max(x,y)

결과 CCSID는 다음 표에 근거하여 결과 하위 유형을 판별합니다.

한 피연산자 열...	다른 피연산자...	결과 열의 하위 유형은 다음과 같습니다.
UCS-2 자료	DBCS 또는 혼합 또는 SBCS 자료	UCS-2 자료
DBCS 자료	DBCS 또는 혼합 또는 SBCS 자료	DBCS 자료
비트 자료	혼합, SBCS 또는 비트 자료	비트 자료
혼합 자료	혼합 또는 SBCS 자료	혼합 자료
SBCS 자료	SBCS 자료	SBCS 자료

숫자 피연산자

숫자 유형은 다른 숫자 유형에만 호환될 수 있습니다.

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
FLOAT(배정밀도)	숫자 유형	FLOAT(배정밀도)
FLOAT(단정밀도)	FLOAT(단정밀도)	FLOAT(단정밀도)
FLOAT(단정밀도)	DECIMAL, NUMERIC, BIGINT, INTEGER 또는 SMALLINT	FLOAT(배정밀도)

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
DECIMAL(w,x)	DECIMAL(y,z) 또는 NUMERIC(y,z,)	DECIMAL(p,s) 여기에서 p = min(31, max(x,z)+max(w-x,y-z)) s = max(x,z)
DECIMAL(w,x)	BIGINT	DECIMAL(p,x) 여기에서 p = min(31, x+max(w-x,19))
DECIMAL(w,x)	INTEGER	DECIMAL(p,x) 여기에서 p = min(31, x+max(w-x,11))
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x) 여기에서 p = min(31, x+max(w-x,5))
NUMERIC(w,x)	NUMERIC(y,z)	NUMERIC(p,s) 여기에서 p = min(31, max(x,z) + max(w-x, y-z)) s = max(x,z)
NUMERIC(w,x)	BIGINT	NUMERIC(p,x) 여기에서 p = min(31, x + max(w-x,19))
NUMERIC(w,x)	INTEGER	NUMERIC(p,x) 여기에서 p = min(31, x + max(w-x,11))
NUMERIC(w,x)	SMALLINT	NUMERIC(p,x) 여기에서 p = min(31, x + max(w-x,5))
BIGINT	BIGINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	SMALLINT	BIGINT
INTEGER	INTEGER	INTEGER
INTEGER	SMALLINT	INTEGER
SMALLINT	SMALLINT	SMALLINT
NONZERO SCALE BINARY	NONZERO SCALE BINARY	NONZERO SCALE BINARY(하나의 피연산자가 0이 아닌 2진 스케일이면 두 피연산자 모두 같은 스케일의 2진이어야 합니다).

Datetime 피연산자

DATE 유형은 다른 DATE 유형이나 날짜의 유효한 스트링 표시가 들어 있는 CHAR 또는 VARCHAR 표현식과 호환될 수 있습니다. 결과의 자료 유형은 DATE입니다.

TIME 유형은 다른 TIME 유형이나 시간의 유효한 스트링 표시가 들어 있는 CHAR 또는 VARCHAR 표현식과 호환될 수 있습니다. 결과의 자료 유형은 TIME입니다.

TIMESTAMP 유형은 다른 TIMESTAMP 유형이나 시간소인의 유효한 스트링 표시가 들어 있는 CHAR 또는 VARCHAR 표현식과 호환될 수 있습니다. 결과의 자료 유형은 TIMESTAMP입니다.

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
DATE	DATE	DATE
TIME	TIME	TIME
TIMESTAMP	TIMESTAMP	TIMESTAMP

DATALINK 피연산자

DataLink는 다른 DataLink와 호환될 수 있습니다. 그러나, NO LINK CONTROL인 DataLink는 NO LINK CONTROL인 다른 DataLink에만 호환될 수 있고, FILE LINK CONTROL READ PERMISSION FS인 DataLink는 FILE LINK CONTROL READ PERMISSION FS인 다른 DataLink에만 호환될 수 있으며, FILE LINK CONTROL READ PERMISSION DB인 DataLink는 FILE LINK CONTROL READ PERMISSION DB인 다른 DataLink에만 호환될 수 있습니다. 결과의 자료 유형은 DATALINK입니다. 결과 DATALINK의 길이는 모든 자료 유형 중에서 가장 깁니다.

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
DATALINK(x)	DATALINK(y)	DATALINK(z) 여기에서 z = max(x,y)

DISTINCT 유형 피연산자

고유한 유형은 자체에만 호환될 수 있습니다. 결과의 자료 유형은 고유한 유형입니다.

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
고유한 유형	고유한 유형	고유한 유형

스트링 결합 조작을 위한 변환 규칙

스트링을 결합하는 조작은 연결, UNION 및 UNION ALL입니다(이 규칙은 MAX, MIN, VALUE, COALESCE, IFNULL 및 CONCAT 스칼라 함수와 CASE 표현식에도 적용됩니다). 각 경우에 결과의 CCSID는 바인드시 판별되며 조작 실행에는 CCSID로 식별되는 코드화 문자 세트로 스트링 변환이 포함될 수 있습니다.

결과의 CCSID는 피연산자의 CCSID에 의해 판별됩니다. 처음 두 피연산자의 CCSID로 중간 결과 CCSID가 판별되고, 이 CCSID와 다음 피연산자의 CCSID로 새로운 중간 결과 CCSID가 판별되며 이런 방식으로 계속 처리됩니다. 마지막 중간 결과 CCSID와 마지막 피연산자의 CCSID는 결과 스트링이나 열의 CCSID를 판별합니다. 각 쌍의 CCSID에 대해 결과 CCSID는 다음 규칙의 순차적 어플리케이션에 의해 판별됩니다.

- CCSID가 같으면 결과는 해당 CCSID입니다.
- 하나의 CCSID가 65535이면 결과는 65535입니다.[21]
- 한 CCSID가 다른 CCSID와 다른 코드화 체계로 자료를 나타내는 경우 다음 표에 의해 결과가 판별됩니다.

21. 연산자 중 하낙 CLOB나 DBCLOB면 결과 CCSID는 작업 디폴트 CCSID입니다.

표 17. 중간 결과의 코드화 체계 선택

첫 번째 피연산자	두 번째 피연산자			
	SBCS 자료	**DBCS 자료**	**혼합 자료**	**UCS-2 자료**
SBCS 자료	아래 참조	두 번째	두 번째	두 번째
DBCS 자료	첫 번째	아래 참조	두 번째	두 번째
혼합 자료	첫 번째	첫 번째	아래 참조	두 번째
UCS-2 자료	첫 번째	첫 번째	첫 번째	아래 참조

- 그렇지 않은 경우 결과 CCSID가 다음 표에 의해 판별됩니다.

표 18. 중간 결과의 CCSID 선택

첫 번째 피연산자	두 번째 피연산자				
	열 값	**파생된 값**	**상수**	**특수 레지스터**	**호스트 변수**
열 값	첫 번째	첫 번째	첫 번째	첫 번째	첫 번째
파생된 값	두 번째	첫 번째	첫 번째	첫 번째	첫 번째
상수	두 번째	두 번째	첫 번째	첫 번째	첫 번째
특수 레지스터	두 번째	두 번째	첫 번째	첫 번째	첫 번째
호스트 변수	두 번째	두 번째	두 번째	두 번째	첫 번째

그러나, 외부 코드화 체계로 자료가 들어 있는 호스트 변수는 조작에 사용되기 전에 고유 코드화 체계로 변환됩니다. 위의 규칙은 변환이 이미 발생된 것으로 가정합니다.

중간 결과는 파생된 값 피연산자로 간주됨에 유의하십시오. 예를 들어, COLA, COLB 및 COLC는 각각 CCSID가 37, 278 및 500인 열이라고 가정합니다. COLA CONCAT COLB CONCAT COLC의 결과 CCSID는 다음과 같이 판별됩니다.

1. 두 피연산자가 모두 열이기 때문에 COLA CONCAT COLB의 결과 CCSID가 우선 37로 판별되므로 첫 번째 피연산자의 CCSID가 선택됩니다.
2. 단계 1의 결과와 COLC를 연결한 결과 CCSID는 500으로 결정됩니다. 첫 번째 피연산자가 파생된 값이고 두 번째 피연산자가 열이기 때문에 결과 CCSID가 500으로 판별되므로 두 번째 피연산자의 CCSID가 선택됩니다.

필요한 경우 연결의 피연산자나 MAX, MIN, VALUE, COALESCE, IFNULL 및 CONCAT 스칼라 함수의 선택된 인수가 결과 스트링의 코드화 문자 세트로 변환됩니다. 필요한 경우 UNION 또는 UNION ALL의 각 스트링이 결과 열의 코드화 문자 세트로 변환됩니다. 다음이 모두 참일 경우에만 문자 변환이 필요합니다.

- CCSID가 다른 경우
- 두 CCSID가 모두 65535가 아닌 경우
- 스트링이 널이 아니고 비어 있지도 않은 경우
- CCSID 변환 선택 표가 변환 필요성을 나타낼 경우

스트링 문자가 변환될 수 없거나 CCSID 변환 선택 표가 사용되지만 CCSID 쌍에 대한 정보는 들어 있지 않는 경우 오류가 발생합니다. 스트링 문자가 대체 문자로 변환되는 경우 경고가 발생합니다.

상수

(때로 *리터럴*이라고도 하는) *상수*가 값을 지정합니다. 상수는 스트링 상수나 숫자 상수로 분류됩니다. 스트링 상수는 문자나 그래픽으로 더 자세히 분류됩니다. 숫자 상수는 정수, 부동 소수점 또는 소수로 더 자세히 분류됩니다.

모든 상수에는 NOT NULL 속성이 있습니다. 값이 0인 숫자 상수의 음부호는 무시됩니다.

정수 상수

*정수 상수*는 소수점이 없는 최대 19 자릿수의 부호가 있는 또는 부호가 없는 숫자로 정수를 지정합니다. 그 값이 큰 정수 범위 안에 있는 경우 정수 상수의 자료 유형은 큰 정수입니다. 값이 큰 정수 범위 안에 없지만 큰 정수 범위 안에 있는 경우 정수 상수의 자료 유형은 큰 정수입니다. 큰 정수 값 범위를 벗어나서 정의되는 상수는 십진 상수로 간주됩니다.

구문 도표에서 *integer*는 부호가 없는 큰 정수 상수에 사용됩니다.

예

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

부동 소수점 상수

*부동 소수점 상수*는 E로 분리되는 두 숫자로 배정밀도 부동 소수점 숫자를 지정합니다. 첫 번째 숫자에는 부호와 소수점이 포함될 수 있고 두 번째 숫자에는 부호는 포함되지만 소수점은 포함될 수 없습니다. 상수의 값은 첫 번째 숫자와 두 번째 숫자에 의해 지정된 10의 누승을 곱한 값입니다. 부동 소수점 숫자 범위 안에 있어야 합니다. 상수의 문자 수는 24를 넘지 않아야 합니다. 선행 제로를 제외하고 첫 번째 숫자의 자릿수는 17을 넘지 않아야 하고 두 번째 숫자의 자릿수는 3을 넘지 않아야 합니다.

예

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

소수 상수

*소수 상수*는 최대 31자리를 포함하는 부호가 있는 또는 부호가 없는 숫자로 소수를 지정합니다. 상수는 다음과 같아야 합니다.

- 소수점을 포함합니다.
- 2147483647 보다 크거나 -2147483647 보다 작습니다.

정밀도는(선행 제로와 후미 제로를 포함하는) 총 자릿수이며 스케일은(후미 제로를 포함하는) 소수점 오른쪽의 자릿수입니다.

예

```
25.5     1000.     -15.     +37589.3333333333     12345678901
```

2진 스트링 상수

*2진 스트링 상수*는 가변 길이 2진 스트링을 지정합니다. 2진 스트링 상수의 양식은 다음과 같습니다.

- 스트링 분리 문자로 시작하고 끝나는 일련의 문자가 뒤에 붙는 X. 스트링 분리 문자 사이의 문자는 16진 짝수여야 합니다. 16진수는 32740을 넘지 않아야 합니다. 16진수는 숫자나 A - F(대문자 또는 소문자)입니다.

상수에 할당된 CCSID가 65535입니다.

2진 스트링 상수의 구문은 문자 상수의 두 번째 양식과 같음에 주의하십시오. SET OPTION문이 2진 스트링 옵션(SQLCURRULE = *STD)에 지정되었거나 CRTSQLxxx 명령에 SQLCURRULE(*STD) 매개변수가 지정된 경우 이 양식의 상수는 2진 스트링 상수로만 취급됩니다.

예

```
X'FFFF'
```

문자 스트링 상수

*문자 스트링 상수*는 가변 길이 문자 스트링을 지정합니다. 문자 스트링 상수의 두 가지 형식은 다음과 같습니다.

- 스트링 분리 문자로 시작하고 끝나는 일련의 문자. 스트링 분리 문자 사이의 바이트 수는 32740 보다 클 수 없습니다. 두 개의 연속 스트링 분리 문자를 사용하여 문자 스트링 안에 하나의 스트링 분리 문자를 표시합니다. 스트링 안에 들어 있지 않은 두 개의 연속 스트링 분리 문자는 빈 스트링을 나타냅니다.
- 스트링 분리 문자로 시작하고 끝나는 일련의 문자가 뒤에 붙는 X. 스트링 분리 문자 사이의 문자는 16진 짝수여야 합니다. 16진수는 32740을 넘지 않아야 합니다. 16진수는 숫자나 A - F(대문자 또는 소문자)입니다. 16진 표기법의 규칙으로 16진수의 각 쌍은 문자를 나타냅니다. 이런 형식의 스트링 상수를 사용하면 키보드 표시가 없는 문자를 지정할 수 있습니다.

문자 스트링 상수에는 혼합 자료가 들어갈 수 있습니다. 작업 CCSID가 혼합 자료를 지원하는 경우 DBCS 서브스트링이 포함되어 있으면 문자 스트링이 혼합 자료로 분류됩니다. 다른 모든 경우에는 문자 스트링 상수가 SBCS 자료로 분류됩니다.

소스가 (ASCII와 같은) 외부의 코드화 체계로 코드화되지 않는 한 상수에 지정된 CCSID는 상수가 들어 있는 소스의 CCSID입니다. 호스트 변수의 자료는 외부 코드화 체계에서 현재 서버의 디폴트 CCSID로 변환됩니다. 이때 상수에 지정된 CCSID는 현재 서버의 디폴트 CCSID입니다.

소스의 CCSID는 어플리케이션 사용자에 의해 판별됩니다. 소스의 CCSID는 다음과 같습니다.

- STRSQL의 경우 어플리케이션 사용자의 디폴트 CCSID
- RUNSQLSTM 또는 STRREXPRC 명령의 경우 지정된 소스 파일의 CCSID
- CRTSQLxxx의 경우:
  - 정적 SQL의 경우 소스의 CCSID는 CRTSQLxxx 명령에 사용된 소스 파일의 CCSID입니다.
  - 동적 SQL의 경우 소스의 CCSID는 PREPARE 명령문에 지정된 호스트 변수의 CCSID이거나 PREPARE 명령문에 스트링 상수가 지정되어 있으면 현재 서버의 디폴트 CCSID입니다.

예

```
'Peggy'     '14.12.1990'     '32'     'DON''T CHANGE'     ''     X'FFFF'
```

그래픽 스트링 상수

DBCS 그래픽 스트링 상수

*그래픽 스트링 상수*는 가변 길이 그래픽 상수입니다. 지정된 스트링의 길이는 16730 보다 클 수 없습니다. DBCS 그래픽 스트링 상수의 세 가지 형식은 다음과 같습니다.

문맥	그래픽 스트링 상수	빈 스트링	예제
모든 문맥	G'SO dbcs-string SI'	G'SOSI' G'' g'SOSI' g''	G'SO元気SI'
	N'SO dbcs-string SI'	N'SOSI' N'' n'SOSI' n''	
PL/I	SO'dbcs-string'GSI	SO''GSI	SO'元気'GSI

RV3F000-0

일반적인 형식에서 SQL 분리 문자와 G 또는 N은 SBCS 문자입니다. SBCS ’는 EBCDIC 작은 따옴표, X‘7D’입니다.

PL/I 형식에서 작은 따옴표와 G는 DBCS 문자입니다. 두 개의 연속 DBCS 스트링 분리 문자를 사용하여 스트링 안에 하나의 스트링 분리 문자를 표시합니다. 이 PL/I 형식은 PL/I 프로그램에 삽입된 정적 명령문에만 유효함에 유의하십시오.

16진 DBCS 그래픽 상수도 지원됩니다. 16진 DBCS 그래픽 상수의 형식은 다음과 같습니다.

GX'ssss'

상수에서 **ssss**는 0 - 32766 16진수의 스트링을 나타냅니다. 스트링 분리 문자 사이의 문자 수는 4의 짝수 배수여야 합니다. 4자리의 각 그룹은 하나의 DBCS 그래픽 문자를 나타냅니다. SI와 SO('0E'X 및 '0F'X)에 대한 16진수는 스트링에 포함되지 않습니다.

소스가 (ASCII와 같은) 외부의 코드화 체계로 코드화되지 않는 한 상수에 지정된 CCSID는 소스의 CCSID와 연관된 DBCS CCSID입니다. 이때 상수가 들어 있는 SQL문이 준비되면 상수에 지정된 CCSID는 현재 서버의 디폴트 CCSID와 연관된 DBCS CCSID입니다. 소스의 CCSID와 연관된 DBCS CCSID가 없으면 CCSID는 65535입니다.

DBCS CCSID 관련 정보는 iSeries Information Center의 국제화 DBCS CCSID 주제를 참조하십시오. 소스의 CCSID에 대한 자세한 내용은 문자 스트링 상수를 참조하십시오.

UCS-2 그래픽 스트링 상수

16진 UCS-2 그래픽 상수가 지원됩니다. 16진 UCS-2 그래픽 상수의 형식은 다음과 같습니다.

UX'ssss'

상수에서 **ssss**는 0 - 32766 16진수의 스트링을 나타냅니다. 스트링 분리 문자 사이의 문자 수는 4의 짝수 배수여야 합니다. 4자리의 각 그룹은 하나의 UCS-2 그래픽 문자를 나타냅니다.

UCS-2의 CCSID는 13488입니다.

소수점

디폴트 소수점은 다음과 같은 경우 지정될 수 있습니다.

- 숫자 상수를 해석하려는 경우
- 문자 스트링을 숫자로 캐스팅할 때 사용할 소숫점 문자를 결정하려는 경우(예를 들어, DECIMAL, DOUBLE_PRECISION, FLOAT, REAL 스칼라 함수 및 CAST 표현식에서)
- 숫자를 스트링으로 캐스팅할 때 결과에 사용할 소수점 문자를 결정하려는 경우(예를 들어, CHAR, CLOB, VARGRAPHIC 스칼라 함수 및 CAST 표현식에서)

디폴트 소수점은 다음과 같은 인터페이스로 지정될 수 있습니다.

표 19. 디폴트 날짜 형식 인터페이스

SQL 인터페이스	**스펙**
삽입 SQL	OPTION 매개변수의 *JOB, *PERIOD, *COMMA 또는 *SYSVAL 값은 Create SQL Program(CRTSQLxxx) 명령으로 지정됩니다. SET OPTION문은 삽입 SQL에 들어 있는 프로그램 소스에 DECMPT 매개변수를 지정하는 데 사용될 수 있습니다. (CRTSQLxxx 명령에 대한 자세한 정보는 호스트 언어로 SQL 프로그래밍 책을 참조하십시오).
대화식 SQL 및 SQL문 실행	Start SQL (STRSQL) 명령의 DECPNT 매개변수 또는 세션 속성을 변경하여. Run SQL (RUNSQLSTM) 명령의 DECMPT 매개변수. (STRSQL 및 RUNSQLSTM 명령에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.)
서버의 호출 레벨 인터페이스	SQL_ATTR_DATE_FMT 및 SQL_ATTR_DATE_SEP 환경 또는 연결 변수 (CLI에 대한 자세한 정보는 SQL 호출 레벨 인터페이스 (ODBC) 책을 참조하십시오.)
Developer Kit for Java를 사용한 서버의 JDBC 또는 SQLJ	십진 분리자 연결 등록 정보 (JDBC와 SQLJ에 대한 자세한 정보는 iSeries Information Center의 IBM Developer Kit for Java 주제를 참조하십시오).
iSeries Access ODBC 드라이버를 사용하는 클라이언트의 ODBC	ODBC 설정의 고급 서버 옵션의 소숫점 분리자 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries 액세스 범주를 참조하십시오.)
IBM Toolbox for Java를 사용하는 클라이언트의 JDBC	JDBC 설정의 형식 (ODBC에 대한 자세한 정보는 iSeries Information Center의 iSeries 액세스 범주를 참조하십시오.) (IBM Toolbox for Java에 대한 자세한 정보는 iSeries Information Center의 IBM Toolbox for Java 주제를 참조하십시오).

쉼표가 소수점이면 다음 규칙이 적용됩니다.

- 마침표는 소수점으로도 사용됩니다.
- 리스트에서 숫자 상수의 분리자로 사용된 쉼표는 그 뒤에 공백을 두어야 합니다.
- 소수점으로 사용된 쉼표 뒤에는 공백이 올 수 없습니다.

따라서 분수 부분없이 십진 상수를 지정하려면 후미 쉼표 뒤에 공백이 아닌 문자가 와야 합니다. 비공백 문자는 다음과 같이 분리자 쉼표가 될 수있습니다.

```
VALUES(9999999999,, 111)
```

분리 문자

*APOST와 *QUOTE는 COBOL 명령문 안에서 스트링 분리 문자의 이름을 지정하는 상호 배타적인 COBOL 사전컴파일러 옵션입니다. *APOST는 스트링 분리 문자로 작은 따옴표(')를 지정하고 *QUOTE는 인용 부호(")를 지정합니다. *APOSTSQL와 *QUOTESQL는 COBOL 프로그램에 삽입 SQL문에 대해 유사한 역할을 수행하는 상호 배타적인 COBOL 사전컴파일러 옵션입니다. *APOSTSQL는 SQL 스트링 분리 문자로 작은 따옴표(')를 지정하고 이 옵션의 경우 인용부호(")는 SQL 이탈 문자입니다. *QUOTESQL는 SQL 스트링 분리 문자로 인용 부호를 지정하며 이 옵션의 경우 작은 따옴표가 SQL 이탈 문자입니다. *APOSTSQL 및 *QUOTESQL의 값은 각각 *APOST 및 *QUOTE의 값과 같습니다.

COBOL 이외의 호스트 언어는 사용이 고정됩니다. 호스트 언어와 정적 SQL문에 대한 스트링 분리 문자는 작은 따옴표(')이며 SQL 이탈 문자는 인용 부호(")입니다.

특수 레지스터

*특수 레지스터*는 데이터베이스 관리자에 의해 어플리케이션 프로세스에 대해 정의되는 기억장치이며 SQL문에 참조될 수 있는 정보를 저장하는 데 사용됩니다. 특수 레지스터에 대한 참조는 현재 서버에 의해 제공되는 값에 대한 참조입니다. 값이 스트링이면 해당 CCSID는 현재 서버의 디폴드 CCSID입니다. iSeries용 DB2 UDB에는 다음의 특수 레지스터가 포함됩니다.

CURRENT DATE 또는 CURRENT_DATE

CURRENT DATE 특수 레지스터는 SQL문이 현재 서버에서 실행될 때 날짜 시간 시계 읽기의 기준인 날짜를 지정합니다. 다음 상황에서 모든 값은 한번의 시계 읽기를 기초로 나옵니다.

- 하나의 SQL문 안에서 특수 레지스터가 두 번 이상 사용되는 경우
- 하나의 명령문 안에서 특수 레지스터가 CURRENT TIME 또는 CURRENT TIMESTAMP 특수 레지스터나 CURDATE, CURTIME 또는 NOW 스칼라 함수와 함께 사용되는 경우

예

PROJECT 표를 사용하여 MA2111 프로젝트(PROJNO)의 프로젝트 종료 날짜를 현재 날짜로 설정합니다.

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

CURRENT PATH, CURRENT_PATH 또는 CURRENT FUNCTION PATH

CURRENT PATH 특수 레지스터는 동적으로 준비된 SQL문에서 규정되지 않은 고유한 유형 이름(내장 유형 및 고유한 유형 모두), 프로시듀어명 및 함수명을 분석하는 데 사용되는 SQL 경로를 지정합니다. SQL CALL문의 호스트 변수(CALL 호스트 변수)로 지정된 규정되지 않은 프로시듀어명을 해결하는 데도 사용됩니다. 자료 유형은 VARCHAR(3483)입니다.

CURRENT PATH 특수 레지스터에는 하나 이상의 스키마명 리스트가 들어 있는데, 각 스키마명은 분리 문자로 묶여 다음 스키마와는 쉼표로 분리됩니다. 분리 문자와 쉼표는 3483자 길이에 포함됩니다. 경로의 스키마명의 최대 길이는 268입니다.

SQL 경로를 사용하여 동적이고 정적인 SQL문과 해당 값의 결과에서 규정되지 않은 이름을 해결하는 경우에 대한 내용은 56 페이지의 『스키마 및 SQL 경로』를 참조하십시오.

활성 그룹에 있는 CURRENT PATH 특수 레지스터의 초기 값은 실행되는 첫 번째 SQL문으로 설정됩니다.

- 활성 그룹의 첫 번째 SQL문이 SQL 프로그램이나 SQL 패키지에서 실행되고 CRTSQLxxx 명령에 SQLPATH 매개변수가 지정되면 SQLPATH 매개변수에 지정된 값이 경로입니다. SQLPATH 값은 또한 SET OPTION 명령문을 사용하여 지정할 수 있습니다.
- 그렇지 않은 경우:
  - SQL 이름을 지정하는 경우 ″QSYS″, ″QSYS2″, *″명령문의 권한부여 ID 값″*
  - 시스템 이름을 지정하는 경우 ″*LIBL″

명령문 SET PATH를 실행하여 레지스터의 값을 변경할 수 있습니다. 명령문에 대한 자세한 내용은 788 페이지의 『SET PATH』를 참조하십시오.

예

스키마 QSYS와 QSYS2(SYSTEM PATH) 전에 먼저 SMITH를 찾도록 특수 레지스터를 설정합니다.

```
SET CURRENT PATH SMITH, SYSTEM PATH
```

CURRENT SCHEMA

CURRENT SCHEMA 특수 레지스터는 동적으로 준비된 SQL문에 적용 가능한 규정되지 않은 오브젝트 참조를 규정하는 데 사용되는 스키마 이름을 나타내는 VARCHAR(128) 값을 지정합니다.[22] CURRENT SCHEMA는 DYNDFTCOL이 지

22. OS/390 및 z/OS용 DB2 UDB와의 호환을 위해, 특수 레지스터 CURRENT SQLID는 CURRENT SCHEMA와 동의어로 간주됩니다.

정된 프로그램의 이름을 규정하는 데 사용되지 않습니다. DYNDFTCOL이 프로그램에 지정되면, CURRENT SCHEMA 스키마명이 아닌 해당 스키마명이 사용됩니다.

CURRENT SCHEMA의 초기값은 현재 세션 사용자의 권한 ID입니다.

DFTRDBCOL 키워드는 정적 SQL문에 대해 적용 가능한 규정되지 않은 데이터베이스 오브젝트 참조를 규정하는 데 사용되는 스키마명을 제어합니다.

예

오브젝트 규정 스키마를 'D123'으로 설정하십시오.

```
SET CURRENT SCHEMA = 'D123'
```

CURRENT SERVER 또는 CURRENT_SERVER

CURRENT SERVER 특수 레지스터는 현재의 서버를 식별하는 VARCHAR(18) 값을 지정합니다.

CURRENT SERVER는 CONNECT (Type 1), CONNECT (Type 2) 또는 SET CONNECTION 명령문으로 변경될 수 있으나 특정 조건에서만 변경될 수 있습니다. 421 페이지의 『CONNECT(유형 1)』, 427 페이지의 『CONNECT(유형 2)』 및 769 페이지의 『SET CONNECTION』에서 설명을 참조하십시오.

ADDRDBDIRE 또는 WRKRDBDIRE 명령으로 관계형 데이터베이스 디렉토리에 항목을 추가하여 로컬 관계형 데이터베이스명을 지정하지 않으면 CURRENT SERVER를 지정할 수 없습니다.

예

호스트 변수 APPL_SERVE(VARCHAR(18))에 현재 서버명을 설정합니다.

```
SELECT CURRENT SERVER
  INTO :APPL_SERVE
  FROM ROW1_TABLE
```

CURRENT TIME 또는 CURRENT_TIME

CURRENT TIME 특수 레지스터는 현재 서버에서 SQL문이 실행될 때 날짜 시간 시계 읽기를 기준으로 시간을 지정합니다. 다음 상황에서 모든 값은 한번의 시계 읽기를 기초로 나옵니다.

- 하나의 SQL문 안에서 특수 레지스터가 두 번 이상 사용되는 경우.
- 하나의 명령문 내에 특수 레지스터가 CURRENT DATE 또는 CURRENT TIMESTAMP 특수 레지스터나 CURDATE, CURTIME 또는 NOW 스칼라 함수와 함께 사용되는 경우

예

CL_SCHED 표를 사용하여 오늘 나중에 시작하는(STARTING) 모든 클래스(CLASS_CODE)를 선택합니다. 오늘 클래스의 DAY 열에 값 3이 있습니다.

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME  AND DAY = 3
```

CURRENT TIMESTAMP 또는 CURRENT_TIMESTAMP

CURRENT TIMESTAMP 특수 레지스터는 현재 서버에서 SQL문이 실행될 때 날짜 시간 시계 읽기가 기준인 시간소인을 지정합니다. 다음 상황에서 모든 값은 한번의 시계 읽기를 기초로 나옵니다.

- 하나의 SQL문 안에서 특수 레지스터가 두 번 이상 사용되는 경우.
- 하나의 명령문 내에 특수 레지스터가 CURRENT DATE 또는 CURRENT TIME 특수 레지스터나 CURDATE, CURTIME 또는 NOW 스칼라 함수와 함께 사용되는 경우

예

IN_TRAY 표에 행을 삽입합니다. RECEIVED 열의 값은 행이 삽입된 시기를 나타내는 시간소인이어야 합니다. 다른 세 개의 열에 대한 값은 호스트 변수 SRC (CHAR(8)), SUB (CHAR(64)) 및 TXT (VARCHAR(200))에서 나옵니다.

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE 또는 CURRENT_TIMEZONE

CURRENT TIMEZONE 특수 레지스터는 UTC[23]와 현재 서버의 로컬 시간 사이의 차이를 지정합니다. 차이는(처음 두 자리는 시, 다음 두자리는 분 및 마지막 두 자리는 초를 나타내는 십진수) 소요 시간에 의해 표시됩니다. 시는 -24와 24 사이의 값이어야 합니다. 로컬 시간에서 CURRENT TIMEZONE를 빼면 로컬 시간이 UTC로 변환됩니다.

예

IN_TRAY 표를 사용하여 표의 모든 행을 선택하고 값을 UTC로 조정합니다.

```
SELECT RECEIVED - CURRENT TIMEZONE, SOURCE,
      SUBJECT, NOTE_TEXT FROM IN_TRAY
```

USER

USER 특수 레지스터는 현재 서버에서 실행시 권한부여 ID를 지정합니다. 특수 레지스터의 자료 유형은 VARCHAR(18)입니다.

23. 이전에 그리니치 표준 시간(GMT)이라고 함.

예

사용자가 작성한 모든 주를 IN_TRAY 표에서 선택합니다.

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

열 이름

열 이름의 의미는 문맥에 따라 달라집니다. 다음과 같이 열 이름이 사용될 수 있습니다.

- CREATE TABLE 명령문처럼 열 이름을 선언하는 경우
- CREATE INDEX 명령문처럼 열을 식별하는 경우
- 다음 문맥처럼 열의 값을 지정하는 경우
  - *열 함수*에서 열 이름은 함수가 적용되는 그룹이나 중간 결과표의 모든 열 값을 지정합니다. 그룹과 중간 결과표는 766 페이지의 『SELECT INTO』에 설명되어 있습니다. 예를 들어, MAX(SALARY)는 함수 MAX를 그룹에 있는 열 SALARY의 모든 값에 적용합니다.
  - *GROUP BY* 또는 *ORDER BY절*에서 열 이름은 절이 적용되는 중간 결과표의 모든 값을 지정합니다. 예를 들어, ORDER BY DEPT는 열 DEPT 값을 기준으로 중간 결과표의 순서를 정합니다.
  - *표현식*, *탐색 조건* 또는 *스칼라 함수*에서 열 이름은 구성이 적용되는 각 행이나 그룹의 값을 지정합니다. 예를 들어, 탐색 조건 CODE = 20이 행에 적용되는 경우 열 이름 CODE로 지정되는 값은 그 행의 열 CODE의 값입니다.

규정된 열 이름

열 이름의 규정자는 표 이름, 뷰 이름, 별명 또는 상관명입니다.

열 이름의 규정화 가능 여부는 문맥에 따라 달라집니다.

- COMMENT와 LABEL 명령문에서 열 이름은 규정되어야 합니다.
- 열 이름이 열 값을 지정하는 경우 열 이름은 사용자 옵션에 따라 규정될 수 있습니다.
- 다른 모든 문맥에서는 열 이름이 규정될 수 없습니다.

규정자가 선택적이면 두 가지 목적을 수행할 수 있습니다. 자세한 내용은 111 페이지의 『모호함을 피하기 위한 열 이름 규정자』 및 112 페이지의 『상관된 참조의 열 이름 규정자』를 참조하십시오.

상관명

*상관명*은 FROM절과 UPDATE 또는 DELETE 명령문의 첫 번째 절에 정의될 수 있습니다. 예를 들어, 아래 표시된 절은 Z를 X.MYTABLE의 상관명으로 설정합니다.

```
FROM X.MYTABLE Z
```

상관명은 정의되는 문맥 안에서만 표, 뷰 또는 별명과 연관됩니다. 따라서 같은 상관명을 다른 명령문에서 다른 목적으로 또는 같은 명령문의 다른 절에 정의할 수 있습니다.

규정자로서 상관명을 사용하여 모호함을 피하거나 상관된 참조를 설정할 수 있습니다. 표, 뷰 또는 별명에 대해 더 짧은 이름으로서 상관명을 사용할 수도 있습니다. 위에 표시된 예에서 Z는 단순히 X.MYTABLE을 두 번 이상 입력하지 않기 위해 사용되었을 수도 있습니다.

표 이름, 뷰 이름 또는 별명으로 상관명을 지정하면 표, 뷰 또는 별명의 해당 인스턴스 열에 대한 규정된 참조에 표 이름, 뷰 이름 또는 별명 보다 상관명을 사용해야 합니다. 예를 들면, 다음 예에서 EMPLOYEE에 상관명이 지정되었으므로 EMPLOYEE.PROJECT에 대한 참조는 올바르지 않습니다.

```
FROM EMPLOYEE E                          ***INCORRECT***
  WHERE EMPLOYEE.PROJECT='ABC'
```

대신 PROJECT에 대한 규정된 참조에 아래와 같이 상관명 “E”가 사용되어야 합니다.

```
FROM EMPLOYEE E
  WHERE E.PROJECT='ABC'
```

FROM절에 지정된 이름은 *노출되거나 노출되지 않습니다*. 상관명은 항상 노출된 이름입니다. 상관명이 지정되지 않으면 표 이름, 뷰 이름 또는 별명은 해당 FROM절에서 노출되었다고 합니다. 예를 들어, 다음 FROM절에서 상관명이 EMPLOYEE에 지정되고 DEPARTMENT에는 지정되지 않으므로 DEPARTMENT는 노출된 이름이고 EMPLOYEE는 노출된 이름이 아닙니다.

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM절에 노출된 표 이름, 뷰 이름 또는 별명은 해당 FROM절 또는 FROM 구의 상관명에 노출된 다른 표 이름이나 뷰 이름과 같지 않아야 합니다. 규정되지 않은 표나 뷰 이름을 규정한 다음 이름이 비교됩니다.

아래 표시된 첫 번째 두 개의 FROM절은 각 절에 노출된 EMPLOYEE에 대한 참조가 하나씩 들어 있으므로 맞습니다.

1. 주어진 FROM절은 다음과 같습니다.

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   EMPLOYEE.PROJECT와 같은 규정된 참조는 FROM절에서 EMPLOYEE의 두 번째 인스턴스 열을 나타냅니다. EMPLOYEE의 첫 번째 인스턴스에 대한 규정된 참조에는 상관명 “E1”(E1.PROJECT)이 사용되어야 합니다.

2. 주어진 FROM절은 다음과 같습니다.

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

EMPLOYEE.PROJECT와 같은 규정된 참조는 FROM절에서 EMPLOYEE의 첫 번째 인스턴스 열을 나타냅니다. EMPLOYEE의 두 번째 인스턴스에 대한 규정된 참조에는 상관명 “E2”(E2.PROJECT)가 사용되어야 합니다.

3. 주어진 FROM절은 다음과 같습니다.

   ```
   FROM EMPLOYEE, EMPLOYEE                    ***INCORRECT***
   ```

   이 절에 포함된 두 개의 노출된 표 이름(EMPLOYEE와 EMPLOYEE)은 같고 다음은 허용되지 않습니다.

4. 다음 명령문이 제공됩니다.

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

   FROM절의 두 EMPLOYEE 인스턴스 모두에 상관명이 있으므로 규정된 참조 EMPLOYEE.PROJECT는 올바르지 않습니다. 대신, PROJECT에 대한 참조가 두 상관명 중 하나로 규정되어야 합니다(E1.PROJECT 또는 E2.PROJECT).

5. 주어진 FROM절은 다음과 같습니다.

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

   EMPLOYEE의 두 번째 인스턴스에 있는 열에 대한 참조에는 X.EMPLOYEE(X.EMPLOYEE.PROJECT)가 사용되어야 합니다. 이 FROM 절은 명령문의 권한부여 ID가 x가 아닌 경우에만 유효합니다.

FROM절에 지정된 상관명은 다음과 같지 않아야 합니다.

- FROM절의 다른 상관명
- FROM절에 노출된 규정되지 않은 표 이름 또는 뷰 이름
- FROM절에 규정된 표 이름 또는 뷰 이름의 두 번째 SQL ID

예를 들어, 다음 FROM절은 올바르지 않습니다.

```
FROM EMPLOYEE E, EMPLOYEE E
FROM EMPLOYEE DEPARTMENT, DEPARTMENT          ***INCORRECT***
FROM X.T1, EMPLOYEE T1
```

다음 FROM절은 혼동될 수 있더라도 기술적으로는 맞습니다.

```
FROM EMPLOYEE DEPARTMENT, DEPARTMENT EMPLOYEE
```

FROM절에서 상관명을 사용하면 결과표의 열과 연관되는 열 이름 리스트를 지정하는 옵션을 사용할 수도 있습니다. 상관명의 경우 이렇게 나열된 열 이름은 조회의 열 참조에 사용되어야 하는 노출된 열 이름이 됩니다. 열 이름 리스트가 지정되면 기준 표의 열 이름은 노출되지 않습니다.

주어진 FROM절은 다음과 같습니다.

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

D.NUM과 같이 규정된 참조는 표에서 DEPTNO로 정의되는 DEPARTMENT 표의 첫 번째 열을 나타냅니다. FROM절을 사용한 D.DEPTNO에 대한 참조는 열 이름 DEPTNO가 노출되지 않은 열 이름이므로 올바르지 않습니다.

열 리스트를 지정하면 표 참조에 있는 열 만큼의 이름으로 구성되어야 합니다. 각 열 이름은 고유한 것으로서 규정되지 않은 것이어야 합니다.

모호함을 피하기 위한 열 이름 규정자

함수, GROUP BY절, ORDER BY절, 표현식 또는 탐색 조건의 문맥에서 열 이름은 표나 뷰에서 열의 값을 참조합니다. 열이 포함될 수 있는 표와 뷰를 문맥의 *오브젝트 표*라고 합니다. 둘 이상의 오브젝트 표에 이름이 같은 열이 포함될 수 있습니다. 열 이름을 규정하는 이유는 열이 나온 오브젝트를 지정하기 위한 것입니다.

표 지정자

특정한 오브젝트 표를 지정하는 규정자를 표 *규정자*라고 합니다. 오브젝트 표를 식별하는 구도 그에 대한 표 지정자를 설정합니다. 예를 들면, SELECT절에 있는 표현식의 오브젝트 표 이름은 뒤에 나오는 FROM절에서 지정됩니다.

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

다음은 FROM절에서 표 지정자를 설정하는 방법입니다.

- 표나 뷰 이름 뒤에 나오는 이름은 상관명과 표 지정자입니다. 따라서 CORZ는 표 지정자입니다. CORZ는 선택 리스트에서 첫 번째열 이름을 규정하는 데 사용됩니다.
- SQL 이름을 지정할 때는 노출된 표나 뷰 이름이 표 지정자입니다. 따라서 OWNY.MYTABLE은 표 지정자입니다. OWNY.MYTABLE은 선택 리스트에서 두 번째 열 이름을 규정하는 데 사용됩니다.
- 시스템 이름을 지정할 때 노출된 표나 뷰 이름의 표 지정자는 노출되지 않은 표나 뷰 이름입니다. 다음 예에서 MYTABLE은 OWNY/MYTABLE의 표 지정자입니다.

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

정의되지 않거나 모호한 참조 회피

열 이름이 열의 값을 참조할 경우 꼭 하나의 오브젝트 표에 해당 이름을 갖는 열이 포함되어야 합니다. 다음 상황은 오류로 간주됩니다.

- 오브젝트 표에 지정된 이름을 갖는 열이 들어 있지 않은 경우. 참조는 정의되지 않습니다.
- 열 이름이 표 지정자로 규정되어 있지만 지정된 표에 지정된 이름을 갖는 열이 포함되어 있지 않은 경우. 다시 참조가 정의되지 않습니다.
- 이름이 규정되지 않고 둘 이상의 오브젝트 표에 그 이름을 갖는 열이 포함되어 있는 경우. 참조는 모호합니다.

고유하게 정의된 표 지정자로 열 이름을 규정하여 모호한 참조는 피하십시오. 열이 다른 이름으로 여러 오브젝트에 들어 있으면 오브젝트 표 이름은 지정자로 사용될 수 있습니다.

둘 이상의 오브젝트 표가 같은 표의 인스턴스가 될 수 있습니다. 이 경우 고유한 상관명을 사용하여 표의 특정한 인스턴스를 모호하지 않게 지정해야 합니다. 다음의 FROM 절에서 X와 Y는 각각 표 CORPDATA.EMPLOYEE의 첫 번째와 두 번째 인스턴스를 참조하도록 정의됩니다.

```
FROM CORPDATA.EMPLOYEE X, CORPDATA.EMPLOYEE Y
```

열을 표 지정자의 노출된 표 이름 형식으로 규정하는 경우 노출된 표 이름의 규정된 또는 규정되지 않은 형식이 사용될 수 있습니다. 그러나, 표 이름이나 뷰 이름 및 표 규정자를 완전히 규정한 후에는 사용된 규정자와 표가 같아야 합니다.

1. 명령문의 권한부여 ID가 CORPDARA인 경우

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

   가 유효한 명령문입니다.

2. 명령문의 권한부여 ID가 REGION인 경우

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

   는 EMPLOYEE가 표 REGION.EMPLOYEE를 나타내므로 유효하지 않지만 WORKDEPT의 규정자는 다른 표 CORPDATA.EMPLOYEE를 나타냅니다.

상관된 참조의 열 이름 규정자

*subselect*는 여러 SQL문의 구성요소로서 사용될 수 있는 조회 형식입니다. 부속 선택에 대한 자세한 정보는 335 페이지의 제 4 장『조회』를 참조하십시오. *부속 조회*는 괄호로 묶은 fullselect 형식입니다. 예를 들어, *부속 조회*를 탐색 조건에 사용할 수 있습니다.

부속 조회에 자체의 탐색 조건이 포함될 수 있고 이런 탐색 조건은 차례로 부속 조회를 포함할 수 있습니다. 따라서 SQL문에 부속 조회의 계층이 포함될 수 있습니다. 부속 조회가 들어가는 계층의 요소를 요소가 포함되는 부속 조회 보다 상위 레벨에 있다고 합니다.

계층의 모든 요소에는 하나 이상의 표 지정자를 설정하는 절이 있습니다. FROM절로서 UPDATE나 DELETE 명령문의 최상위 레벨은 제외합니다. 부속 조회의 탐색 조건, 선택 리스트, 결합 절 또는 표 기능의 인수는 계층의 자체 요소의 FROM 절에 의해 식별되는 표의 열뿐만 아니라 자체 요소에서 계층의 최상위 레벨에 이르는 경로를 따라 모든 레벨에서 식별되는 표의 열도 참조할 수 있습니다. 상위 레벨에서 식별되는 표 열의 참조를 *상관된 참조*라고 합니다.

Q가 T에 대해 정의된 상관명이면 표 T의 열 C에 대한 상관된 참조는 T.C 또는 Q.C가 될 수 있습니다. 그러나 규정되지 않은 열 이름의 상관된 참조는 올바른 수행이 아닙니다. 다음 설명은 상관된 참조가 항상 규정된 열 이름의 형식으로 되어 있고 규정자는 상관명이라는 가정에 근거합니다.

Q.C는 다음 세 가지 조건이 맞을 때만 상관된 참조입니다.

- Q.C가 부속 조회의 탐색 조건, 선택 리스트, 결합 절 또는 표 기능의 인수에서 사용되는 경우.
- Q에서 해당 부속 조회, 선택 리스트, 결합 절, 또는 표 기능의 인수의 FROM 절에 사용되는 표를 지정하지 않은 경우.
- Q에서 일부 상위 레벨에 사용된 표를 지정하는 경우

Q.C는 레벨에서 표나 뷰의 C 열을 참조하며 Q는 그 표나 뷰의 표 지정자로 사용됩니다. 같은 표나 뷰가 여러 레벨에서 식별될 수 있으므로 표 지정자로 고유한 상관명이 권장됩니다. 둘 이상의 레벨에서 표를 지정하는 데 Q가 사용되면 Q.C는 Q.C를 포함하는 부속조회가 들어 있는 최저 레벨을 참조합니다.

다음 명령문에서 Q는 T1과 T2에 대한 상관명으로 사용되지만 Q.C는 Q.C를 포함하는 부속 조회가 들어 있는 최저 레벨이므로 T2와 연관된 상관명을 참조합니다.

```
SELECT *
  FROM T1 Q
  WHERE A < ALL (SELECT B
                   FROM T2 Q
                   WHERE B < ANY (SELECT D
                                    FROM T3
                                    WHERE D = Q.C))
```

규정되지 않은 열 이름

열이 다음과 같으면 규정되지 않은 열 이름도 상관된 참조가 될 수 있습니다.

- 부속 조회의 탐색 조건에 사용되는 경우
- 해당 부속 조회의 FROM절에 사용되는 표에 들어 있지 않은 경우
- 일부 상위 레벨에 사용되는 표에 들어 있는 경우

규정되지 않은 상관된 참조는 SQL문을 이해하기 어렵게 하므로 권장되지 않습니다. 열이 있는 표에 따라 명령문이 준비될 때 열은 내재적으로 규정됩니다. 일단 내재적인 규정이 판별되면 명령문이 다시 준비될 때까지 변경되지 않습니다. 규정되지 않은 상관된 참조가 있는 SQL문이 준비되고 실행될 때 SQL 사전컴파일러는 사전컴파일러 리스트에 경고 메세지를 발행하고 데이터베이스 관리자는 양의 SQLCODE(+12)와 SQLSTATE(01545)를 발행합니다.

변수 참조

SQL문의 *변수*는 SQL문이 실행될 때 변경될 수 있는 값을 지정합니다. SQL문에는 여러 가지 유형의 *변수*가 사용됩니다.

호스트 변수

호스트 변수는 호스트 언어의 명령문으로 정의됩니다. 호스트 변수 참조 방법에 대한 자세한 정보는 114 페이지의 『호스트 변수에 대한 참조』를 참조하십시오.

전이 변수

전이 변수는 트리거에 정의되며 열의 기존 값이나 새 값을 나타냅니다. 전이 변수 참조에 대한 자세한 정보는 575 페이지의 『CREATE TRIGGER』를 참조하십시오.

SQL 변수

SQL 변수는 SQL 함수, SQL 프로시듀어 또는 트리거의 SQL 복합 명령문에 의해 정의됩니다. SQL 변수에 대한 자세한 정보는 824 페이지의 『SQL 매개변수 및 변수에 대한 참조』를 참조하십시오.

SQL 매개변수

SQL 매개변수는 CREATE FUNCTION(SQL 스칼라), CREATE FUNCTION(SQL 표) 또는 CREATE PROCEDURE(SQL)문에 정의됩니다. SQL 변수에 대한 자세한 정보는 824 페이지의 『SQL 매개변수 및 변수에 대한 참조』를 참조하십시오.

매개변수 마커

변수는 동적 SQL문에서 참조할 수 없습니다. 매개변수 마커는 SQLDA에 정의되며 대신 사용됩니다. 매개변수 마커에 대한 자세한 정보는 731 페이지의 『매개변수 마커』를 참조하십시오.

이 책에서 달리 주가 표시되지 않았다면 구문 도표에서 *호스트 변수*라는 용어는 호스트 변수, 전이 변수, SQL 변수, SQL 매개변수 또는 매개변수 마커를 포괄하여 지칭하는 데 사용됩니다.

호스트 변수에 대한 참조

*호스트 변수*는 COBOL 자료 항목, RPG 필드 또는 SQL문에서 참조되는 PLI, REXX, C++ 또는 C 변수입니다. 호스트 변수는 호스트 언어의 명령문으로 정의됩니다. C, C++, COBOL, PL/I 및 RPG에서 호스트 구조 참조 방법에 대한 자세한 내용은 119 페이지의 『C, C++, COBOL PL/I 및 RPG의 호스트 구조』를 참조하십시오. REXX 호스트 변수에 대한 자세한 정보는 호스트 언어로 SQL 프로그래밍 책을 참조하십시오.

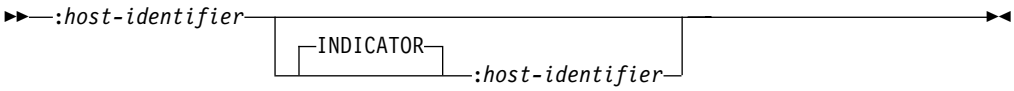
SQL문의 *host-variable*은 호스트 변수 선언에 대한 규칙에 따라 프로그램에 서술된 호스트 변수를 식별하여야 합니다. SQL문에 사용되는 모든 호스트 변수는 REXX 및 RPG

이외의 모든 호스트 언어로 SQL 선언 섹션에서 선언되어야 합니다(변수는 REXX로 선언되지 않아도 됩니다. RPG에는 선언 섹션이 없고 호스트 변수는 프로그램을 통해 선언될 수 있습니다). 변수는 SQL 선언 섹션 내에서 선언된 변수와 동일한 이름으로 SQL 선언 섹션 외부에서 선언될 수 없습니다. SQL 선언 섹션은 BEGIN DECLARE SECTION으로 시작되고 END DECLARE SECTION으로 끝납니다.

호스트 변수 사용에 대한 자세한 내용은 SQL 프로그래밍 개념 책을 참조하십시오.

구문 도표에 사용된 용어 *host-variable*은 호스트 변수에 대한 참조를 나타냅니다. FETCH, SELECT INTO, SET 변수나 VALUES INTO 명령문의 INTO절에 있는 *host-variable*은 행의 열 값이 지정되는 호스트 변수를 식별합니다. CALL 명령문 또는 EXECUTE 명령문의 호스트 변수는 출력 매개변수 값이 지정되는 호스트 변수 중 하나나 둘 모두 어플리케이션 프로그램에서 데이터베이스 관리자로 전달하는 입력 인수 값을 지정하는 호스트 변수를 식별합니다. 다른 모든 문맥에서 *host-variable*은 어플리케이션 프로그램에서 iSeries용 DB2 UDB로 전달되는 값을 지정합니다.

host-variable 참조의 일반 형식은 다음과 같습니다.



각 *host-identifier*는 소스 프로그램에서 선언되어야 합니다. 두 번째 *host-identifier*로 지정된 변수의 자료 유형은 스케일이 0인 *작은 정수*여야 합니다.

첫 번째 *host-identifier*는 *주 변수*를 지정하고 두 번째 *host-identifier*는 *인디케이터 변수*를 지정합니다. 인디케이터 변수의 목적은 다음과 같습니다.

- 널값 지정. 인디케이터 변수의 음수 값은 널값을 지정합니다.
- 다음 자료 맵핑 오류 중 하나 표시
  - 문자를 변환할 수 없음
  - 숫자 변환 오류(부족(underflow) 또는 넘침)
  - 연산식 오류(0으로 나눔)
  - 날짜 또는 시간소인 오류(지정된 형식에 대해 유효한 날짜 범위 안에 있지 않은 날짜나 시간소인)
  - 유효하지 않은 datetime 값의 스트링 표시
  - 제대로 형성되지 않은 혼합된 자료
  - 유효하지 않은 숫자 값
  - 범위를 벗어나는 SUBSTR 스칼라 함수의 인수
- 절단된 스트링의 원래 길이 작성
- 호스트 변수에 지정할 때 시간이 절단되면 시간의 두 번째 부분 작성

예를 들어, :V1:V2가 값 삽입이나 갱신을 지정하는 데 사용되고, V2가 음수이면 지정된 값은 널값입니다. V2가 음수가 아니면 지정된 값은 V1의 값입니다.

마찬가지로, :V1:V2가 CALL, FETCH 또는 SELECT INTO 명령문에 지정되고 리턴되는 값이 널이면 V1은 정의되지 않고 V2는 음수 값으로 설정됩니다. 음수 값은 다음과 같습니다.

- 선택된 값이 널값이면 -1
- 외부 subselect의 선택 리스트에서 자료 맵핑 오류로 인해 널값이 리턴되면 -2[24]

리턴된 값이 널이 아니면 그 값이 V1에 지정되고 V2는 0으로 설정됩니다(V1에 대한 지정으로 스트링이 절단되어야 하지 않는 한 V2는 스트링의 원래 길이로 설정됨). 시간의 두 번째 부분이 절단 요구 지정되면 V2는 초 수로 설정됩니다.

두 번째 호스트 ID가 생략되면 *host variable*은 인디케이터 변수가 없습니다. *host-variable* :V1로 지정된 값은 항상 V1이며, 널값은 변수에 지정될 수 없습니다. 따라서 해당되는 결과 열에 널값이 포함될 수 없는 한 이 형식은 INTO절에 사용될 수 없습니다. 이 형식이 사용되고 열이 널을 포함하면 데이터베이스 관리자는 SQLCA의 SQLCODE 필드에 음의 값(-407)을 리턴합니다. 자료가 절단되고 인디케이터 변수가 없으면, 오류가 발생하지 않습니다.

호스트 변수가 SQL문에서 사용되는 경우에는 항상 콜론이 선행되어야 합니다.

C, C++, ILE RPG 및 PL/I에서 호스트 변수를 참조하는 SQL문은 해당 호스트 변수의 선언 범위 안에 있어야 합니다. 커서 SELECT 명령문에서 참조되는 호스트 변수의 경우 그 규칙은 DECLARE CURSOR 명령문이 아닌 OPEN 명령문에 적용됩니다.

스트링 호스트 변수의 CCSID는 다음 중 하나입니다.

- DECLARE VARIABLE 명령문에 지정된 CCSID 또는
- CCSID절이 있는 DECLARE VARIABLE이 호스트 변수에 대해 지정되지 않으면 CCSID가 외부 코드화 체계(예: ASCII)에 대한 것이 아닌 한, 호스트 변수가 들어 있는 SQL문이 실행될 때 어플리케이션 리퀘스터의 디폴트 CCSID. 이 경우 호스트 변수가 현재 서버의 디폴트 CCSID로 변환됩니다.

예

PROJECT 표를 사용하는 경우 호스트 변수 PNAME(VARCHAR(26))을 프로젝트명(PROJNAME)으로, 호스트 변수 STAFF(DECIMAL(5,2))를 평균사원 레벨(PRSTAFF)로, 호스트 변수 MAJPROJ(CHAR(6))를 프로젝트(PROJNO) ‘IF1000’에 대한 주 프로젝트(MAJPROJ)로 설정합니다. PRSTAFF 및 MAJPROJ 열에는 널 값

24. 자료 맵핑 오류로 리턴되는 널값이 특정한 스칼라 함수 및 연산식에 대해 리턴될 수 있더라도, 결과 열은 연산식 또는 스칼라 함수의 인수가 널 허용이 아니면 널로 허용될 수 없습니다.

이 들어 있을 수 있으므로 인디케이터 변수 STAFF_IND(SMALLINT)와 MAJPROJ_IND(SMALLINT)를 제공합니다.

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
  INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
  FROM PROJECT
  WHERE PROJNO = 'IF1000'
```

동적 SQL의 호스트 변수

동적 SQL문에서 호스트 변수 대신 매개변수 마커가 사용됩니다. 매개변수 마커는 어플리케이션이 값을 제공하는 동적 SQL문에서의 위치 즉, 명령문 스트링이 정적 SQL문일 경우 호스트 변수가 발견된 위치를 나타내는 의문 부호(?)입니다. 다음 예는 호스트 변수를 사용하는 정적 SQL과 매개변수 마커를 사용하는 동적 명령문을 나타냅니다.

```
INSERT INTO DEPT VALUES( :HV_DEPTNO, :HV_DEPTNAME, :HV_MGRNO, :HV_ADMRDEPT)

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

매개변수 마커에 대한 자세한 정보는 731 페이지의 『매개변수 마커』를 참조하십시오.

LOB 호스트 변수에 대한 참조

정상 LOB 변수, LOB 로케이터 변수(118 페이지의 『LOB 로케이터 변수에 대한 참조』 참조) 및 LOB 파일 참조 변수(118 페이지의 『LOB 파일 참조 변수에 대한 참조』 참조)는 다음 호스트 언어로 정의될 수 있습니다.

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I

LOB가 허용되는 경우 구문 도표의 용어 호스트 변수는 정상 호스트 변수, 로케이터 변수 또는 파일 참조 변수를 참조할 수 있습니다. 이들 변수는 호스트 프로그래밍 언어에서 고유한 자료 유형이 아니므로 SQL 부가 제품이 사용되고 사전컴파일러는 각 변수를 표시하는 데 필요한 호스트 언어 구성을 생성합니다.

전체 LOB 값을 유지할 만큼 큰 호스트 변수를 정의할 수 있고 서버에서 자료 전송을 지연하는 성능상의 이점이 필요하지 않는 경우 LOB 로케이터가 필요하지 않습니다. 그러나, 호스트 언어 제한사항, 기억장치 제한사항 또는 성능 요구사항으로 인해 임시 기억장치에 전체 LOB 값을 때론 저장할 수 없습니다. 한 번에 전체 LOB 값을 저장할 수 없는 경우 LOB 로케이터로 LOB 값을 참조하여 LOB 값 부분을 LOB 값 부분만을 포함하는 호스트 변수에서 선택하거나 갱신할 수 있습니다.

다른 모든 호스트 변수와 같이 LOB 로케이터 변수나 LOB 파일 참조 변수는 연관된 인디케이터 변수를 갖을 수 있습니다. LOB 로케이터 변수와 LOB 파일 참조 변수에 대한 인디케이터 변수는 다른 자료 유형의 인디케이터 변수와 같은 방법으로 실행됩니

다. 널값이 데이터베이스에서 리턴되는 경우 인디케이터 변수가 설정되고 호스트 변수는 변경되지 않습니다. 즉, 로케이터는 널값을 가리킬 수 없음을 의미합니다.

LOB 로케이터 변수에 대한 참조

LOB *로케이터 변수*는 서버에서의 LOB 값을 나타내는 로케이터가 들어 있는 호스트 변수로서 다음과 같은 호스트 언어에서 정의할 수 있습니다.

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I

로케이터를 LOB 값 조작에 사용할 수 있는 방법에 대한 자세한 내용은 66 페이지의 『로케이터로 큰 오브젝트(LOB) 조작』을 참조하십시오.

SQL문의 로케이터 변수는 로케이터 변수 선언에 대한 규칙에 따라 프로그램에 서술된 LOB 로케이터 변수를 식별하여야 합니다. 이것은 항상 SQL문을 통하여 간접적으로 이루어집니다. C의 경우 다음과 같습니다.

```
static volatile SQL TYPE IS CLOB_LOCATOR *loc1;
```

구문 도표에서 사용된 용어 *로케이터 변수*는 LOB 로케이터 변수에 대한 참조를 나타냅니다. 메타 변수 *로케이터 변수*는 *호스트 변수*의 경우와 같이 *호스트 ID*를 포함하도록 확장될 수 있습니다.

LOB 로케이터와 연관된 인디케이터 변수가 널이면 참조된 LOB의 값은 널입니다.

로케이터 변수가 현재 값을 나타내지 않으면 로케이터 변수를 참조할 때 오류가 발생합니다.

트랜잭션 확약이나 트랜잭션 종료시 트랜잭션에 의해 예약된 모든 LOB 로케이터가 해제됩니다.

어플리케이션 프로그래머는 LOB 로케이터가 처음 LOB 로케이터를 생성한 것과 같은 서버에서 실행되는 SQL 명령문에서만 사용되도록 해야 합니다. 예를 들어 LOB 로케이터가 한 서버에서 리턴되어 LOB 로케이터 변수에 지정되었다고 가정합니다. LOB 로케이터 변수가 다른 서버에서 실행되는 SQL문에 사용되는 경우 예측할 수 없는 결과가 발생합니다.

LOB 파일 참조 변수에 대한 참조

LOB *파일 참조 변수*는 다음 호스트 언어로 정의될 수 있는 LOB에 대한 직접 파일 입력과 출력에 사용됩니다.

- C

- C++
- ILE RPG
- ILE COBOL
- PL/I

이들은 고유한 자료 유형이 아니므로 SQL 부가 제품이 사용되고 사전컴파일러는 각 변수 표시에 필요한 호스트 언어 구성을 생성합니다.

파일 참조 변수는 LOB 로케이터가 LOB 자료를 포함하는 것이 아니라 나타내는 것과 같이 파일을 (포함하는 것이 아니라) 나타냅니다. 데이터베이스 조회, 갱신 및 삽입에서 파일 참조 변수를 사용하여 하나의 열 값을 저장하거나 검색할 수 있습니다. 참조되는 파일이 어플리케이션 리퀘스터에 반드시 존재해야 합니다.

다른 모든 호스트 변수와 같이 파일 참조 변수에 연관된 인디케이터 변수가 포함될 수 있습니다.

파일 참조 변수의 길이 속성은 LOB의 최대 길이인 것으로 가정됩니다.

파일 참조 변수는 현재 루트(/), QOpenSys 및 UDFS 파일 시스템에서 지원됩니다. 파일이 작성시 파일에 작성 중인 자료의 CCSID를 받습니다. 현재 혼합 CCSID는 지원되지 않습니다. 파일 참조 변수로 작성된 파일을 사용하려면 파일이 2진 모드에서 열려야 합니다.

파일 참조 변수에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.

C, C++, COBOL PL/I 및 RPG의 호스트 구조

호스트 구조는 COBOL 그룹, PL/I, C 또는 C++ 구조 또는 SQL문에서 참조되는 RPG 자료 구조가 될 수 있습니다. 호스트 구조는 호스트 언어로 SQL 프로그래밍 책에서 설명한 대로 호스트 언어의 명령문으로 정의됩니다. 여기서 사용된 대로 용어 호스트 구조에는 SQLCA나 SQLDA가 포함되지 않습니다.

호스트 구조 참조 형식은 호스트 변수 참조 형식과 동일합니다. 참조 :S1:S2는 S1이 호스트 구조인 경우 호스트 구조 참조입니다. S1이 호스트 구조를 지정하면 S2는 작은 정수 변수이거나 작은 정수 변수의 배열이어야 합니다. S1은 호스트 구조이고 S2는 해당 인디케이터 배열입니다.

호스트 구조는 호스트 변수의 리스트가 참조될 수 있는 문맥에서 참조될 수 있습니다. 호스트 구조 참조는 호스트 언어 구조 선언에 정의된 순서로 구조 안에 포함된 각 호스트 변수에 대한 참조와 동일합니다. 인디케이터 배열의 *n*번째 변수는 호스트 구조의 *n*번째 변수에 대한 인디케이터 변수입니다.

C, C++, COBOL, PL/I 및 RPG의 호스트 구조

예를 들어, PL/I에서 V1, V2 및 V3이 구조 S1 안의 변수로 선언되면 명령문은 다음과 같습니다.

```
EXEC SQL FETCH CURSOR1 INTO :S1;
```

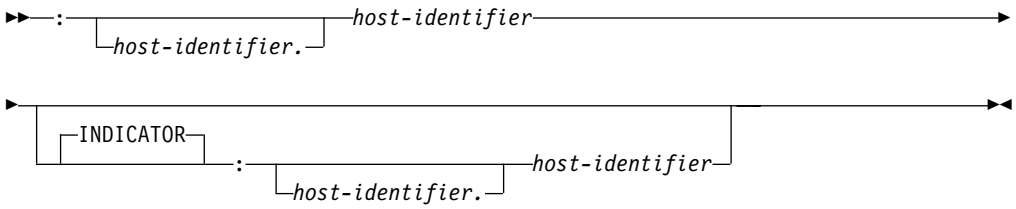
다음과 동일합니다.

```
EXEC SQL FETCH CURSOR1 INTO :V1, :V2, :V3;
```

호스트 구조가 인디케이터 배열 보다 *m*개 많은 변수를 갖으면, 호스트 구조의 마지막 *m*개 변수에는 인디케이터 변수가 없습니다. 호스트 구조가 인디케이터 배열보다 *m*개 적은 변수를 갖는 경우 인디케이터 배열의 마지막 *m*개 변수는 무시됩니다. 호스트 구조에 대한 참조가 인디케이터 변수를 포함하거나 호스트 변수에 대한 참조가 인디케이터 배열을 포함하면 이 규칙이 적용됩니다. 인디케이터 배열이나 인디케이터 변수가 지정되지 않으면 호스트 구조의 변수에 인디케이터 변수가 없습니다.

구조 참조 뿐 아니라 호스트 구조의 각 호스트 변수나 인디케이터 배열의 인디케이터 변수도 규정된 이름으로 참조할 수 있습니다. 규정된 형식은 호스트 ID, 마침표, 다른 호스트 ID 순입니다. 첫 번째 호스트 ID는 호스트 구조를 나타내고 두 번째 호스트 ID는 호스트 구조 안의 호스트 변수를 나타내야 합니다.

다음 도표는 호스트 변수와 호스트 구조에 대한 참조 구문을 지정합니다.



표현식의 호스트 변수는 호스트 변수 선언 규칙에 따라 프로그램에 서술된(구조가 아닌) 변수를 식별해야 합니다.

REXX에서는 호스트 구조가 지원되지 않습니다.

다음 예는 호스트 변수와 호스트 구조에 대한 참조를 나타냅니다.

```
:V1    :S1.V1    :S1.V1:V2    :S1.V2:S2.V4
```

C, C++, COBOL, PL/I 및 RPG의 호스트 구조 배열

PL/I, C++ 및 C에서 호스트 구조 배열은 차원 속성이 있는 구조명입니다. COBOL에서는 1차원 표입니다. RPG에서는 발생 자료 구조입니다. 호스트 구조 배열은 FETCH 명령문에서 복수 행 페치를 사용할 때 또는 INSERT 명령문에서 블록화된 삽입을 사용할 경우에만 참조될 수 있습니다. 호스트 구조 배열은 SQL Programming with Host Languages 책에서 설명한 대로 호스트 언어의 명령문으로 정의됩니다.

C, C++, COBOL, PL/I 및 RPG의 호스트 구조 배열

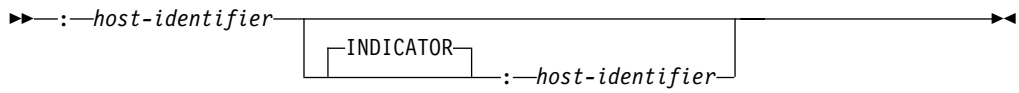
호스트 구조 배열의 형식은 호스트 변수 참조의 형식과 동일합니다. 참조 :S1:S2는 S1이 호스트 구조 배열인 경우 호스트 구조 배열에 대한 참조입니다. S1이 호스트 구조를 지정하면 S2는 작은 정수 호스트 변수, 작은 정수 변수의 배열 또는 작은 정수 호스트 변수의 2차원 배열이어야 합니다. 다음 예에서 S1은 호스트 구조 배열이고 S2는 해당 인디케이터 배열입니다.

```
EXEC  SQL FETCH CURSOR1 FOR 5 ROWS
          INTO :S1:S2;
```

호스트 구조와 인디케이터 배열의 차원은 같아야 합니다.

호스트 구조가 인디케이터 배열 보다 *m*개 많은 변수를 갖으면, 호스트 구조의 마지막 *m*개 변수에는 인디케이터 변수가 없습니다. 호스트 구조가 인디케이터 배열보다 *m*개 적은 변수를 갖는 경우 인디케이터 배열의 마지막 *m*개 변수는 무시됩니다. 인디케이터 배열이나 변수가 지정되지 않으면 호스트 구조 배열의 변수에 인디케이터 변수가 없습니다.

다음 도표는 호스트 구조에 대한 참조 구문을 지정합니다.



REXX에서는 호스트 구조 배열이 지원되지 않습니다.

함수

*함수*는 괄호로 묶인 하나 이상의 피연산자 앞의 함수명에 의해 표시되는 연산입니다. 이는 입력 값 세트와 결과 값 세트의 관계를 나타냅니다. 함수에 대한 입력 값을 *인수*라고 합니다. 예를 들어, 함수는 날짜와 시간 자료 유형인 두 개의 입력 인수를 패스할 수 있으며 결과로서 시간소인 자료 유형의 값을 리턴할 수 있습니다.

함수 유형

함수를 분류하는 여러 방법이 있습니다. 함수를 분류하는 한 가지 방법은 내장, 사용자 정의 또는 고유한 유형에 대해 생성되는 사용자 정의입니다.

- *내장 함수*는 iSeries용 DB2 UDB와 함께 IBM에서 제공하는 함수입니다. 이 함수는 값이 하나인 결과를 제공합니다. 내장 함수에는 연산자 함수(예: ″+″), 열 함수

(예: AVG), 스칼라 함수(예: SUBSTR) 등이 포함됩니다. 내장 열과 스칼라 함수의 리스트 및 이들 함수에 대한 내용은 157 페이지의 제 3 장 『내장 함수』를 참조하십시오.[25]

- *사용자 정의 기능*은 CREATE FUNCTION문을 사용하여 작성되고 카탈로그 표 QSYS2.SYSROUTINES 및 카탈로그 뷰 QSYS2.SYSFUNCS의 데이터베이스 관리자에 등록되는 함수입니다. 이 함수를 사용하여 사용자와 제 3자 판매 함수 정의를 추가하여 데이터베이스 관리자의 함수를 확장할 수 있습니다.

  사용자 정의 함수는 *SQL*, *외부* 또는 소스입니다 SQL 함수는 SQL문만 사용하여 데이터베이스에 대해 정의됩니다. 외부 함수는 함수가 호출될 때 실행되는 외부 프로그램이나 서비스 프로그램에 대한 참조로 데이터베이스에 대해 정의됩니다. 소스 함수는 내장 함수가 다른 사용자 정의 함수에 대한 참조로 데이터베이스에 대해 정의됩니다. 소스 함수는 고유한 유형에 사용할 내장 열 및 스칼라 함수 확장에 사용될 수 있습니다.

  사용자 정의 함수는 작성된 *스키마*에 상주합니다. QSYS, QSYS2 또는 QTEMP는 스키마가 될 수 없습니다.

- 데이터베이스 관리자는 고유한 유형이 CREATE DISTINCT TYPE 명령문을 사용하여 작성되는 경우 일부 사용자 정의 함수를 자동으로 생성합니다. 이들 함수는 고유한 유형에서 소스 유형으로, 소스 유형에서 고유한 유형으로 캐스트를 지원합니다. 자료 유형 사이의 캐스트 함수는 고유한 유형이 자체 유형으로만 호환될 수 있으므로 중요합니다.

  생성된 캐스트 함수는 함수가 작성된 고유한 유형과 같은 스키마에 존재합니다. QSYS, QSYS2 또는 QTEMP는 스키마가 될 수 없습니다. 고유한 유형에 대하여 생성된 함수에 대한 자세한 정보는 435 페이지의 『CREATE DISTINCT TYPE』을 참조하십시오.

함수를 분류하는 다른 방법은 입력 자료 값과 결과 값에 따라 달라지는 열, 스칼라 또는 표 함수에 대한 것입니다.

*열 함수*는 각 인수에 대한 값 세트(예: 열 값)를 받아 값이 하나인 입력 값 세트에 대한 결과를 리턴합니다. 열 함수를 총계 함수라고도 합니다. 내장 함수와 사용자 정의 소스 함수는 열 함수가 될 수 있습니다.

*스칼라 함수*는 각 인수에 대한 값을 받아 값이 하나인 결과를 리턴합니다. 내장 함수와 사용자 정의 함수는 스칼라 함수가 될 수 있습니다. 고유한 유형에 대해 작성된 함수도 스칼라 함수입니다.

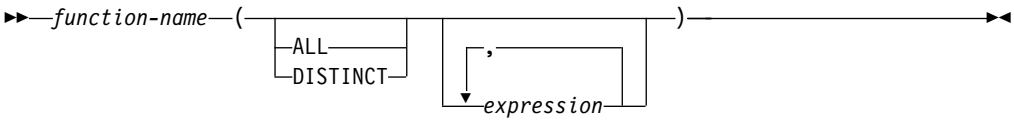
---

25. *내장 함수*는 데이터베이스 관리자에 의해 내부적으로 수행되므로 *내장 함수*에 대해 연관된 프로그램 또는 서비스 프로그램 오브젝트는 없습니다. 또한 카탈로그에는 *내장 함수*에 대한 내용이 포함되지 않습니다. 그러나, *내장 함수*는 QSYS2에 있는 것처럼 처리될 수 있고 *내장 함수*명을 QSYS2로 규정할 수 있습니다.

표 함수는 수신하는 인수 세트에 대한 표를 리턴합니다. 각각의 인수는 단일 값입니다. 표 함수는 부속 선택의 FROM절에만 참조될 수 있습니다. 표 함수는 외부 함수 또는 SQL 함수로만 정의될 수 있습니다(표 함수는 피소스 함수가 될 수 없습니다).

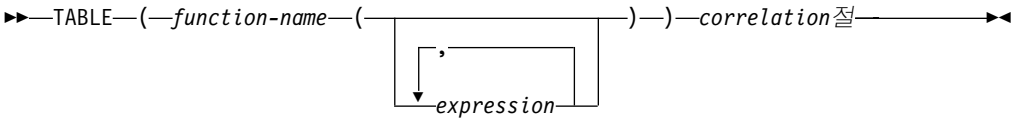
표 함수는 DB2 자료가 아닌 자료에 SQL 언어 처리력을 적용하거나 그러한 자료를 DB2 표로 변환하는 데 사용됩니다. 예를 들어, 표 함수는 파일을 가지고 표로 변환하거나, 웹에서 자료를 가져와서 표 형식으로 정리하거나 Notes 데이터베이스에 액세스하여 메일 메시지에 대한 정보를 리턴할 수 있습니다.

스칼라 또는 열 함수(내장 또는 사용자 정의)에 대한 각 참조는 다음 구문과 일치합니다.



ALL 또는 DISTINCT 키워드는 열 함수 또는 열 함수에 대해 소스인 사용자 정의 함수에 대해 지정될 수 있습니다.

각 표 함수를 언급할 때는 다음 구문을 따라야 합니다.



위의 구문에서, *expression*은 스칼라 또는 열 함수에 대한 것과 같습니다. 표 함수를 나타내는 데 대한 자세한 내용은 340 페이지의 『from절』의 FROM절 설명을 참조하십시오.

함수 분석

함수는 함수명으로 호출되는데, 함수명은 스키마명 및 함수에 대한 인수를 괄호로 묶어 그 다음에 표시하여 내재적 또는 명시적으로 규정됩니다. 데이터베이스 내에서 각 함수는 함수 서명으로 고유하게 식별되는데, 함수 서명은 스키마명, 함수명, 매개변수 수 및 매개변수의 자료 유형입니다. 따라서 스키마에는 이름이 같아도 매개변수의 수가 다르거나 매개변수의 자료 유형이 다른 함수가 여럿 있을 수 있습니다. 또는 이름이 같고 매개변수의 수와 유형이 같은 함수가 여러 스키마에 존재할 수도 있습니다. 함수를 호출하면 데이터베이스 관리자는 실행할 함수를 판별하여야 합니다. 이 프로세스를 함수 분석이라고 합니다.

함수 분석은 규정되지 않은 함수명의 경우 둘 이상의 데이터베이스 관리자가 둘 이상의 스키마를 탐색해야 된다는 점을 제외하고는 규정된 함수명으로 호출되는 함수와 규정되지 않은 함수명으로 호출되는 함수의 경우는 둘 다 유사합니다.

규정된 함수 분석: 함수가 함수명과 스키마명으로 호출될 때 데이터베이스 관리자는 지정된 스키마만을 찾아서 어떤 함수를 실행시킬 것인지 분석합니다. 다음 조건이 모두 참이면 데이터베이스 관리자는 해당되는 함수 인스턴스를 찾습니다.

- 함수 인스턴스 이름이 함수 호출의 이름과 일치합니다.
- 함수 인스턴스에 있는 입력 매개변수의 수가 함수 호출의 인수 수와 일치합니다.
- 함수 호출에 있는 각 입력 인수의 자료 유형이 함수 인스턴스의 해당 매개변수 자료 유형과 일치하거나 *승격*될 수 있습니다.

  이 자료 유형의 비교 결과로 실행에 대한 최선의 선택사항이 선택됩니다(125 페이지의 『가장 적합한 함수를 찾는 방법』 참조). 자료 유형의 승격에 대한 내용은 76 페이지의 『자료 유형의 승격』을 참조하십시오.

  해당 스키마에서 기준에 맞는 함수가 없는 경우 오류가 발생합니다.

규정되지 않은 함수 분석: 함수명만으로 함수를 호출하는 경우 데이터베이스 관리자는 실행할 함수 인스턴스를 분석하기 위해 둘 이상의 스키마를 탐색해야 합니다. SQL 경로에는 탐색할 스키마 리스트가 들어 있습니다. 경로(경로에 대한 내용은 56 페이지의 『스키마 및 SQL 경로』 참조)의 각 스키마에 대해서 데이터베이스 관리자는 다음과 같은 기준에 따라 후보 함수를 선택합니다.

- 함수 인스턴스 이름이 함수 호출의 이름과 일치합니다.
- 함수 인스턴스에 있는 입력 매개변수의 수가 함수 호출의 함수 인수 수와 일치합니다.
- 함수 호출에 있는 각 입력 인수의 자료 유형이 함수 인스턴스의 해당 매개변수 자료 유형과 일치하거나 *승격*될 수 있습니다.

  이 자료 유형의 비교 결과로 실행에 대한 최선의 선택사항이 선택됩니다(125 페이지의 『가장 적합한 함수를 찾는 방법』 참조). 자료 유형의 승격에 대한 내용은 76 페이지의 『자료 유형의 승격』을 참조하십시오.

  해당 스키마에서 기준에 맞는 함수가 없는 경우 오류가 발생합니다.

맞지 않는 기준이 하나라도 있는 경우 스키마에 대하여 후보 함수가 선택되지 않습니다.

데이터베이스 관리자가 후보 함수를 식별한 후 실행할 함수 인스턴스로 가장 적합한 후보를 선택합니다(125 페이지의 『가장 적합한 함수를 찾는 방법』 참조). 가장 적합한(스키마명을 제외한 함수 서명이 동일한) 함수 인스턴스가 있는 스키마가 둘 이상 있는 경우 데이터베이스 관리자는 SQL 경로에서 가장 먼저 나오는 스키마의 함수를 선택합니다.

함수 분석은 내장 함수를 포함하여 모든 함수에 적용됩니다. 내장 함수는 논리적으로 스키마 QSYS2에 있습니다. 스키마 QSYS2가 SQL 경로에 명시적으로 지정되지 않으

면 스키마가 내재적으로 경로의 앞에 있는 것으로 가정됩니다. 따라서, 규정되지 않은 함수명이 지정되면 의도된 함수가 선택되도록 경로를 지정하십시오.

가장 적합한 함수를 찾는 방법

같은 이름을 갖는 실행 후보 함수가 둘 이상 있을 수 있습니다. 그런 경우 데이터베이스 관리자는 인수와 매개변수 자료 유형을 비교하여 호출하기에 가장 적합한 함수를 판별합니다. 고려하는 함수 결과와 함수 유형(열 또는 스칼라)이 이를 판별하기 위해 입력되지 않음에 유의하십시오.

주어진 함수에 대한 모든 매개변수의 자료 유형이 함수 호출 인수의 자료 유형과 같으면 해당 함수가 가장 적합한 함수입니다. 정확히 일치하지 않으면 데이터베이스 관리자는 다음 메소드를 사용하여 왼쪽에서 오른쪽으로 매개변수 리스트의 자료 유형을 비교합니다.

1. 함수 호출의 첫 번째 인수 자료 유형과 각 함수의 첫 번째 매개변수 자료 유형과 비교하십시오(자료 유형의 길이, 정밀도, 스케일 및 CCSID 속성은 비교시 고려되지 않습니다).
2. 인수의 경우 함수에 다른 함수의 자료 유형 보다 함수 호출에 더 적합한 자료 유형이 있으면 해당 함수가 가장 적합한 함수입니다. 76 페이지의 『자료 유형의 승격』의 자료 유형 승격에 대한 선행 리스트에서 가장 적합한 자료 유형에서 가장 적합지 못한 자료 유형까지 각 자료 유형에 맞는 자료 유형을 표시합니다.
3. 둘 이상의 후보 함수에 대한 첫 번째 매개변수의 자료 유형이 함수 호출에 대해 동일하게 잘 맞으면 함수 호출의 다음 인수에 대해 이 프로세스를 반복합니다. 가장 적합한 함수를 찾을 때까지 각 인수에 대해 계속합니다.

다음 예는 함수 분석을 나타냅니다.

예 1: MYSCHEMA가 두 함수를 부분 CREATE FUNCTION 명령문으로 작성한 후 FUNA로 명명한다고 가정합니다.

```
CREATE FUNCTION MYSCHEMA.FUNA (VARCHAR(10), INT, DOUBLE) ...
CREATE FUNCTION MYSCHEMA.FUNA (VARCHAR(10), REAL, DOUBLE) ...
```

또한 자료 유형이 VARCHAR(10), SMALLINT 및 DECIMAL인 세 개의 인수가 있는 함수를 규정된 이름으로 호출한다고 가정합니다.

```
MYSCHEMA.FUNA( VARCHARCOL, SMALLINTCOL, DECIMALCOL ) ...
```

두 MYSCHEMA.FUNA 함수 모두 123 페이지의 『함수 분석』에 지정된 기준과 일치하므로 함수 호출에 대한 후보입니다. 자료 유형이 모두 VARCHAR인 스키마의 두 함수 인스턴스에 대한 첫 번째 매개변수의 자료 유형이 함수 호출 첫 번째 인수의 자료 유형 varchar과 동일하게 일치합니다. 그러나, 두 번째 매개변수의 경우 첫 번째 함수의 자료 유형(INT)이 두 번째 함수의 자료 유형(REAL) 보다 두 번째 인수의 자료 유

형(SMALLINT)에 더 잘 맞습니다. 따라서, 데이터베이스 관리자는 실행할 함수 인스턴스로 첫 번째 MYSCHEMA.FUNA 함수를 선택합니다.

예 2: 부분 CREATE FUNCTION 명령문으로 함수가 작성되었다고 가정합니다.

```
1. CREATE FUNCTION SMITH.ADDIT (CHAR(5), INT, DOUBLE) ...
2. CREATE FUNCTION SMITH.ADDIT (INT, INT, DOUBLE) ...
3. CREATE FUNCTION SMITH.ADDIT (INT, INT, DOUBLE, INT) ...
4. CREATE FUNCTION JOHNSON.ADDIT (INT, DOUBLE, DOUBLE) ...
5. CREATE FUNCTION JOHNSON.ADDIT (INT, INT, DOUBLE) ...
6. CREATE FUNCTION TODD.ADDIT (REAL) ...
7. CREATE FUNCTION TAYLOR.SUBIT (INT, INT, DECIMAL) ...
```

또한 어플리케이션에서 함수를 호출할 때의 SQL 경로가 ″TAYLOR″, ″JOHNSON″, ″SMITH″라고 가정합니다. 다음과 같이 세 개의 자료 유형(INT, INT, DECIMAL)으로 함수가 호출됩니다.

```
SELECT  ... ADDIT(INTCOL1, INTCOL2, DECIMALCOL) ...
```

함수 5가 다음 결과를 기초로 실행할 함수 인스턴스로 선택됩니다.

- 함수 6은 스키마 TODD가 SQL 경로에 없기 때문에 후보에서 제거됩니다.
- 스키마 TAYLOR의 함수 7은 함수명이 올바르지 않으므로 후보에서 제거됩니다.
- 스키마 SMITH의 함수 1은 함수 1 의 첫 번째 매개변수인 CHAR 자료 유형으로 승격시킬 수 없으므로 후보에서 제거됩니다.
- 스키마 SMITH의 함수 3은 매개변수의 수가 틀리므로 후보에서 제거됩니다.
- 함수 2는 매개변수의 자료 유형이 인수의 자료 유형과 일치하거나 승격시킬 수 있으므로 후보가 됩니다.
- 스키마 JOHNSON의 함수 4와 5의 매개변수 자료 유형은 두 유형 모두 인수의 자료 유형과 일치하거나 승격시킬 수 있으므로 후보가 됩니다. 그러나 두 함수의 첫 번째 매개변수 자료 유형(INT)이 첫 번째 인수(INT)와 일치하더라도 함수 5의 두 번째 매개변수 자료 유형(INT)이 함수 4 의 자료 유형(DOUBLE) 보다 두 번째 인수(INT)와 더 일치하므로 더 좋은 후보로 함수 5가 선택됩니다.
- 나머지 후보 즉, 함수 2와 5 중 SQL 경로에서 스키마 JOHNSON이 SMITH 앞에 오므로 데이터베이스 관리자는 함수 5를 선택합니다.

예 3: 부분 CREATE FUNCTION 명령문으로 함수가 작성되었다고 가정합니다.

```
1. CREATE FUNCTION BESTGEN.MYFUNC (INT, DECIMAL(9,0)) ...
2. CREATE FUNCTION KNAPP.MYFUNC (INT, NUMERIC(8,0))...
3. CREATE FUNCTION ROMANO.MYFUNC (INT, FLOAT) ...
```

또한 어플리케이션에서 함수를 호출할 때의 SQL 경로가 ″ROMANO″, ″KNAPP″, ″BESTGEN″이라고 가정합니다. 다음과 같이 두 개의 자료 유형(SMALLNT, DECIMAL)으로 함수가 호출됩니다.

```
SELECT  ... MYFUNC(SINTCOL1, DECIMALCOL) ...
```

함수 2가 다음 결과를 기초로 실행할 함수 인스턴스로 선택됩니다.

- 세 함수 모두 123 페이지의 『함수 분석』에 지정된 기준과 맞으므로 이 함수 호출에 대한 후보입니다.
- 스키마 ROMANO의 함수 3은 두 번째 매개변수(FLOAT)가 함수 1(DECIMAL)이나 함수 2(NUMERIC)의 두 번째 매개변수만큼 두 번째 인수(DECIMAL)에 적합하지 않으므로 제거됩니다.
- 함수 1(DECIMAL)과 함수 2(NUMERIC)의 두 번째 매개변수는 두 번째 인수(DECIMAL)에 동일하게 잘 맞습니다.
- SQL 경로에서 ″KNAPP″가 ″BESTGEN″ 앞에 오므로 함수 2가 마지막으로 선택됩니다.

함수 호출

일단 함수가 선택되어도 함수의 사용을 허용할 수 없는 이유가 계속 있습니다. 각 함수는 특정한 자료 유형으로 결과를 리턴하도록 정의됩니다. 이 결과 자료 유형이 함수가 호출되는 문맥 안에서 호환될 수 없으면 오류가 발생합니다. 예를 들어, STEP이라는 주어진 함수는 다음 결과로서 다른 자료 유형으로 정의됩니다.

```
STEP(SMALLINT) RETURNS CHAR(5)
STEP(DOUBLE) RETURNS INTEGER
```

그런 다음 함수 참조로 정의됩니다(여기서 S는 SMALLINT 열).

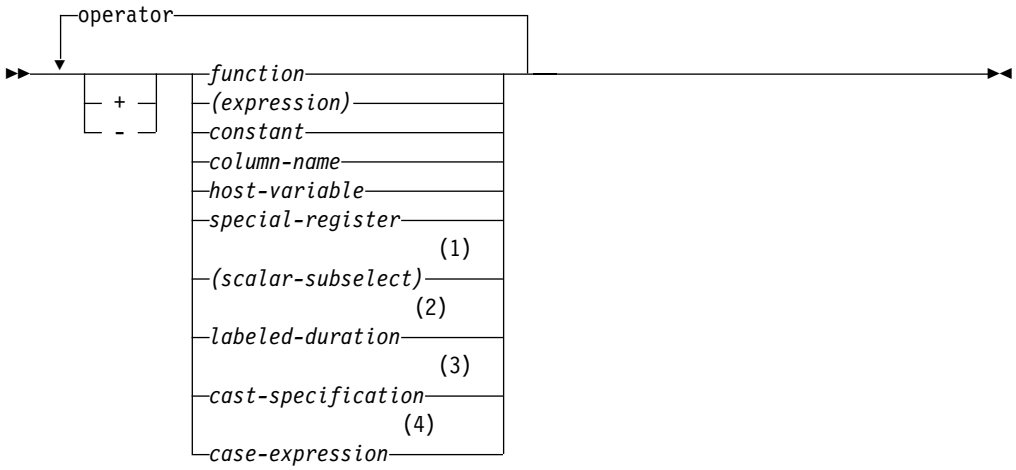
```
SELECT ... 3 +STEP(S)
```

인수 유형에 정확하게 일치하는 것이 있으므로 첫 번째 STEP이 선택됩니다. 추가 연산자의 인수에 필요한 숫자 유형 대신 결과 유형이 CHAR(5)이므로 명령문에 오류가 발생합니다.

함수 호출의 인수가 선택한 함수의 매개변수 자료 유형과 정확하게 일치하지 않는 경우 인수는 열에 대한 지정과 같은 규칙을 사용하여 실행시 매개변수의 자료 유형으로 변환됩니다(80 페이지의 『지정과 비교』 참조). 여기에는 인수와 매개변수 사이의 정밀도, 스케일, 길이 또는 CCSID가 다를 경우가 포함됩니다.

표현식

표현식은 값을 지정합니다.



주:

1 자세한 내용은 133 페이지의 『스칼라 부속 선택』을 참조하십시오.

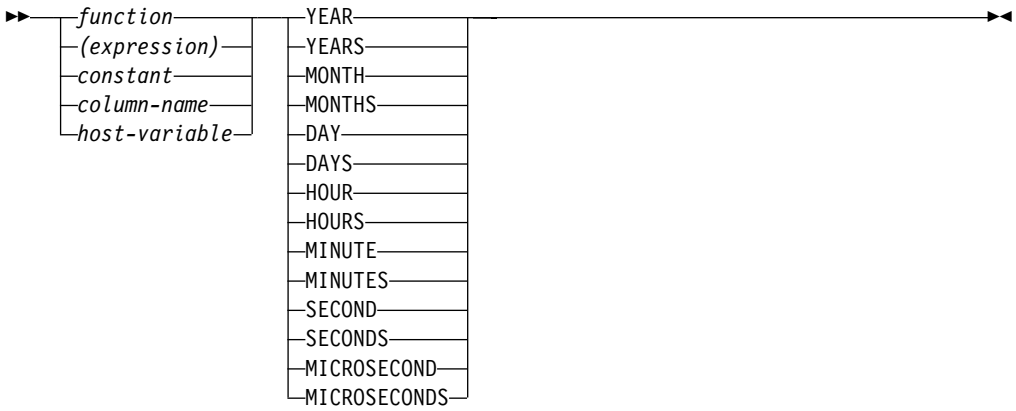
2 자세한 내용은 133 페이지의 『Datetime 피연산자와 기간』을 참조하십시오.

3 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

4 자세한 내용은 139 페이지의 『CASE 표현식』을 참조하십시오.

연산자:

레이블된 기간:



연산자가 없는 경우

연산자가 사용되지 않는 경우 표현식의 결과는 지정된 값입니다.

예

```
SALARY      :SALARY      'SALARY'      MAX(SALARY)
```

연결 연산자를 사용하는 경우

연결 연산자(CONCAT 또는 ||)는 두 개의 스트링을 결합합니다. 표현식의 결과는 스트링입니다.

연결 피연산자는 스트링과 호환될 수 있어야 합니다. 2진 스트링은 다른 2진 스트링과 호환될 수 있습니다.

결과의 자료 유형은 피연산자의 자료 유형에 따라 판별됩니다. 결과의 자료 유형은 다음 표에 요약됩니다.

표 20. 연결된 결과 자료 유형

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
DBCLOB(x)	CHAR(y), VARCHAR(y), CLOB(y), GRAPHIC(y), VARGRAPHIC(y) 또는 DBCLOB(y)	DBCLOB(z) 여기에서 z = MIN(x + y, DBCLOB의 최대 길이)
CLOB(x)	GRAPHIC(y) 또는 VARGRAPHIC(y)	DBCLOB(z) 여기에서 z = MIN(x + y, DBCLOB의 최대 길이)
VARGRAPHIC(x)	CHAR(y), VARCHAR(y), GRAPHIC(y) 또는 VARGRAPHIC(y)	VARGRAPHIC(z) 여기에서 z = MIN(x + y, VARGRAPHIC의 최대 길이)
VARCHAR(x)	GRAPHIC(y)	VARGRAPHIC(z) 여기에서 z = MIN(x + y, VARGRAPHIC의 최대 길이)
GRAPHIC(x)	CHAR(y) 혼합 자료	VARGRAPHIC(z) 여기에서 z = MIN(x + y, VARGRAPHIC의 최대 길이)
GRAPHIC(x)	CHAR(y) SBCS 자료 또는 GRAPHIC(y)	GRAPHIC(z) 여기에서 z = MIN(x + y, GRAPHIC의 최대 길이)
UCS-2 자료	UCS-2, DBCS, 혼합 또는 SBCS 자료	UCS-2 자료
DBCS 자료	DBCS 또는 혼합 또는 SBCS 자료	DBCS 자료
CLOB(x)	CHAR(y), VARCHAR(y) 또는 CLOB(y)	CLOB(z) 여기에서 z = MIN(x + y, CLOB의 최대 길이)
VARCHAR(x)	CHAR(y) 또는 VARCHAR(y)	VARCHAR(z) 여기에서 z = MIN(x + y, VARCHAR의 최대 길이)

표 20. 연결된 결과 자료 유형 (계속)

한 피연산자 열...	다른 피연산자...	결과 열의 자료 유형...
CHAR(x) 혼합 자료	CHAR(y)	VARCHAR(z) 여기에서 z = MIN(x + y, VARCHAR의 최대 길이)
CHAR(x) SBCS 자료	CHAR(y)	CHAR(z) 여기에서 z = MIN(x + y, CHAR의 최대 길이)
비트 자료	혼합, SBCS 또는 비트 자료	비트 자료
혼합 자료	혼합 또는 SBCS 자료	혼합 자료
SBCS 자료	SBCS 자료	SBCS 자료
BLOB(x)	BLOB(y)	BLOB(z) 여기에서 z = MIN(x + y, BLOB의 최대 길이)

피연산자의 길이 합계가 결과 자료 유형의 최대 길이 속성을 초과한다면:

- 결과의 길이 속성은 결과 자료 유형의 최대 길이입니다.[26]
- 공백만 잘린 경우, 경고나 오류가 발생하지 않습니다.
- 공백이 아닌 부분이 잘리면, 오류가 발생합니다.

두 피연산자가 널이 될 수 있으면 결과도 널이 될 수 있고, 둘 다 널이면 결과도 널값입니다. 그렇지 않으면 결과는 첫 번째 피연산자 스트링과 그 뒤의 두 번째로 구성됩니다.

혼합 자료의 경우 결과의 『연결 부분』에 중복되는 시프트(shift) 코드가 없습니다. 따라서, 첫 번째 피연산자가 『SI』 문자로 끝나는 스트링 (X'0F')이고 반면에 두 번째 피연산자가 『SO』 문자로 시작되는 문자 스트링이면(X'0E'), 결과에서 이들 두 바이트가 제거됩니다.

중복되는 시프트가 제거되지 않으면 결과의 실제는 피연산자 길이의 합계이며, 이때 실제 길이는 피연산자의 길이 합계보다 2 작습니다.

CONCAT 연산자는 || 연산자 대신 사용되어야 합니다. | 문자의 코드점은 CCSID에 따라 달라집니다.

결과의 CCSID는 97 페이지의 『스트링 결합 조작을 위한 변환 규칙』에 설명한 대로 피연산자의 CCSID에 의해 판별됩니다. 다음 규칙의 결과에 유의하십시오.

- 피연산자가 비트 자료이면 결과는 비트 자료입니다.
- 한 피연산자가 혼합 자료이고 다른 피연산자가 SBCS 자료이면 결과는 혼합 자료입니다. 그러나 반드시 결과가 잘 형성된 혼합 자료임을 의미하는 것은 아닙니다.

26. 표현식이 선택 리스트에 있는 경우, 길이 속성을 더 줄여서 최대 레코드 크기 내에 맞도록 할 수 있습니다. 자세한 정보는 569 페이지의 『최대 행 크기』를 참조하십시오.

예

공백이 있는 열 FIRSTNME와 열 LASTNAME를 연결합니다.

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

연산자가 있는 경우

연산자가 사용되면 표현식의 결과는 연산자의 어플리케이션에서 피연산자의 값으로 파생된 수입니다.

피연산자가 널이 될 수 있으면 결과는 널이 될 수 있습니다. 피연산자에 널값이 있으면 표현식의 결과는 널값입니다. 연산자가 문자 스트링에 적용될 수는 없습니다. 예를 들면, USER+2는 유효하지 않습니다.

접두부 연산자 +(*단항 더하기*)는 피연산자를 변경하지 않습니다. 접두부 연산자 -(*단항 빼기*)는 0이 아닌 피연산자의 부호를 반전합니다. A의 자료 유형이 작은 정수이면 -A의 자료 유형은 큰 정수입니다. 접두부 뒤에 있는 토큰의 첫 문자는 더하기나 빼기 부호가 될 수 없습니다.

삽입 연산자, +, -, *, / 및 **는 각각 더하기, 빼기, 곱하기, 나누기 및 지수화를 지정합니다. 나눗셈의 두 번째 피연산자의 값은 0이 아니어야 합니다.

지수화(**) 연산자의 결과는 배정밀도 부동 소수점 숫자입니다. 다른 연산자의 결과는 피연산자의 유형에 따라 달라집니다.

두 개의 정수 피연산자

산술 연산자의 피연산자 둘 다 스케일이 0인 정수인 경우 어느 한(또는 둘 다) 피연산자가 큰 정수가 아니면 연산은 2진으로 수행되고 결과는 큰 정수입니다. 큰 정수인 경우에는 결과가 큰 정수입니다. 나눗셈의 나머지는 없어집니다. 정수 연산 조작의 결과(단항 빼기 포함)는 큰 정수의 범위 안에 있어야 합니다. 두 정수 피연산자 중 하나의 스케일이 0이 아니면 정밀도와 스케일이 같은 소수 피연산자로 변환됩니다.

정수와 소수 피연산자

한 피연산자의 스케일이 0인 정수이고 다른 피연산자가 소수이면 다음 표에 정의된 대로 정밀도가 있고 스케일이 0인 소수로 변환된 정수의 임시 사본을 사용하여 연산이 수행됩니다.

피연산자	소수 사본의 정밀도
열 또는 변수: 큰 정수	19
열 또는 변수: 큰 정수	11
열 또는 변수: 작은 정수	5
(선행 제로 포함하는) 상수	상수의 자릿수와 같음

한 피연산자가 스케일이 0이 아닌 정수이면 정밀도와 스케일이 같은 소수 피연산자로 변환됩니다.

두 개의 소수 피연산자

두 피연산자 모두 소수이면 연산이 소수로 수행됩니다. 소수 연산의 결과는 연산과 피연산자의 정밀도 및 스케일에 의존하는 정밀도와 스케일의 소수입니다. 덧셈 또는 뺄셈의 연산이고 같은 스케일의 연산자가 아니면, 피연산자 중의 하나의 임시 사본으로 연산이 수행됩니다. 더 짧은 피연산자의 사본이 후미 제로로 확장되어 분수 부분이 더 긴 피연산자의 자릿수와 같게 됩니다.

다르게 지정되지 않으면 소수를 허용하는 모든 함수와 연산에 최대 31자리의 정밀도가 허용됩니다. 소수 연산의 결과는 31보다 큰 정밀도를 갖지 않아야 합니다.

SQL의 소수 연산

다음 형식은 SQL에서 소수 연산 결과의 정밀도와 스케일을 정의합니다. 기호 *p*와 *s*는 첫 번째 피연산자의 정밀도와 스케일을 나타내고 기호 *p'*와 *s'*는 두 번째 피연산자의 정밀도와 스케일을 나타냅니다.

더하기와 빼기

더하기와 빼기 결과의 스케일은 max(*s,s'*)입니다. 정밀도는 min(31,max(*p-s,p'-s'*) +max(*s,s'*)+1)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 음이 아니어야 합니다.

부동 소수점 피연산자

연산자의 한 피연산자가 부동 소수점이면 부동 소수점으로 연산이 수행됩니다. 필요하면 피연산자가 먼저 배정밀도 부동 소수점 숫자로 변환됩니다. 따라서 표현식의 요소가 부동 소수점 숫자이면 표현식의 결과는 배정밀도 부동 소수점 숫자입니다.

부동 소수점 숫자와 정수를 포함하는 연산은 배정밀도 부동 소수점으로 변환된 정수의 임시 사본으로 수행됩니다. 부동 소수점 숫자와 소수를 포함하는 연산은 배정밀도 부동 소수점으로 변환된 소수의 임시 사본으로 수행됩니다. 부동 소수점 연산의 결과는 부동 소수점 범위 내에 있어야 합니다.

부동 소수점 피연산자(또는 함수의 인수)가 처리되는 순서는 결과에 약간 영향을 줄 수 있습니다. 부동 소수점 피연산자는 실제 숫자를 대략적으로 표현한 것이기 때문입니다. 피연산자가 처리되는 순서는 Optimizer에 의해 내재적으로 수정될 수 있기 때문에(예

를 들어, Optimizer가 사용할 병렬 처리의 정보 및 사용할 액세스 계획을 결정할 수 있기 때문), 어플리케이션은 부동 소수점 피연산자를 사용하는 SQL문이 실행될 때마다 결과가 정확히 같을 것이라고 보장할 수 없습니다.

고유한 유형의 피연산자

고유한 유형은 소스 자료 유형이 숫자라도 연산자와 함께 사용될 수 없습니다. 연산을 수행하려면 소스로 연산자를 사용하여 함수를 작성하십시오. 예를 들어, 둘 다 자료 유형이 DECIMAL(8,2)인 고유한 유형의 INCOME과 EXPENSES가 있으면 다음의 사용자 정의 함수 REVENUE를 사용하여 빼기를 수행할 수 있습니다.

```
CREATE FUNCTION REVENUE ( INCOME, EXPENSES )
  RETURNS DECIMAL(8,2) SOURCE "-" ( DECIMAL, DECIMAL)
```

또는 사용자 정의 함수로 -(빼기 부호) 연산자를 과부하하여 새로운 자료 유형을 뺄 수 있습니다.

```
CREATE FUNCTION "-" ( INCOME, EXPENSES )
  RETURNS DECIMAL(8,2) SOURCE "-" ( DECIMAL, DECIMAL)
```

스칼라 부속 선택

표현식에 지원되는 스칼라 부속 선택은 괄호로 둘러싼 부속 선택으로서 단일 열 값으로 구성된 단일 행을 리턴합니다. 부속 선택이 행을 리턴하지 않는다면 표현식의 결과는 널 값입니다. 선택 리스트 요소가 단지 열 이름인 표현식인 경우, 결과 열 이름은 해당 열의 이름에 기초합니다. 자세한 내용은 336 페이지의 『subselect』를 참조하십시오.

Datetime 피연산자와 기간

Datetime 값은 늘리고 줄이고 뺄 수 있습니다. 이 연산에는 *기간*이라는 소수가 포함될 수 있습니다. *기간*은 시간 간격의 양수 또는 음수 표시입니다. 네 가지 유형의 기간이 있습니다.

레이블된 기간(125 페이지의 다이어그램 참조)

*레이블된 기간*은(표현식의 결과가 될 수 있는) 숫자와 그 뒤에 7개의 경과 키워드(YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS 또는 MICROSECONDS)를 표시한 대로 특정한 시간 단위를 나타냅니다[27]. 지정된 숫자는 DECIMAL(15,0) 숫자에 지정된 것처럼 변환됩니다. 레이블된 기간은 다른 피연산자의 자료 유형 값이 DATE, TIME 또는 TIMESTAMP인 연산자의 피연산자로만 사용될 수 있습니다. 따라서, 표현식 HIREDATE + 2 MONTHS + 14 DAYS가 유효한 반면 표현식 HIREDATE + (2

27. 키워드 YEAR, MONTH, DAY, HOUR, MINUTE, SECOND 및 MICROSECOND의 단일 형식도 허용됩니다.

MONTHS + 14 DAYS)는 유효하지 않습니다. 두 표현식에서 레이블된 기간은 2 MONTHS와 14 DAYS입니다.

날짜 기간

*날짜 기간*은 DECIMAL(8,0) 숫자로 나타낸 년, 월, 일입니다. 적절하게 해석되려면 숫자 형식이 *yyyymmdd*이어야 하며, 여기서 *yyyy*는 년 수, *mm*은 월 수, *dd*는 일 수를 나타냅니다. 표현식 HIREDATE - BRTHDATE 처럼, 한 날짜에서 다른 날짜를 뺀 결과는 날짜 기간입니다.

시간 기간

*시간 기간*은 DECIMAL(6,0) 숫자로 나타낸 시, 분, 초입니다. 적절하게 해석하려면 숫자 형식이 *hhmmss*이어야 하며, 여기서 *hh*는 시, *mm*은 분, *ss*는 초를 나타냅니다. 한 시간 값에서 다른 시간 값을 뺀 결과는 시간 기간입니다.

시간소인 기간

*시간소인 기간*은 DECIMAL(20,6) 숫자로 나타낸 년, 월, 일, 시, 분, 초 및 마이크로초입니다. 적절하게 해석하려면 숫자 형식이 *yyyymmddhhmmsszzzzzz*이어야 하며, 여기서 *yyyy*, *mm*, *dd*, *hh*, *mm*, *ss* 및 *zzzzzz*은 각각 년, 월, 일, 시, 분, 초 및 마이크로초를 나타냅니다. 한 시간소인 값에서 다른 시간소인 값을 뺀 결과는 시간소인 기간입니다.

SQL의 Datetime 산술

datetime 값에서 수행될 수 있는 연산 조작은 더하기와 빼기 뿐입니다. datetime 값이 더하기의 피연산자이면 다른 피연산자는 기간이어야 합니다. datetime 값과 함께 더하기 연산자 사용에 적용되는 특정한 규칙은 다음과 같습니다.

- 한 피연산자가 날짜이면 다른 피연산자는 날짜 기간이나 년, 월 또는 일로 레이블된 기간이어야 합니다.
- 한 피연산자가 시간이면 다른 피연산자는 시간 기간이나 시, 분 또는 초로 레이블된 기간이어야 합니다.
- 한 피연산자가 시간소인이면 다른 피연산자는 기간이어야 합니다. 모든 유형의 기간이 유효합니다.
- 더하기 연산자의 피연산자는 매개변수 마커가 될 수 없습니다.

datetime 값을 기간에서 뺄 수 없고 두 개의 datetime 값을 빼는 연산은 datetime 값에서 기간을 빼는 연산과 같지 않으므로 datetime 값의 빼기 연산자 사용 규칙은 더하기 연산자와 같지 않습니다. datetime 값과 함께 빼기 연산자 사용에 적용되는 특정한 규칙은 다음과 같습니다.

- 첫 번째 피연산자가 날짜이면 두 번째 피연산자는 날짜, 날짜 기간, 날짜의 스트링 표시 또는 년, 월, 일의 레이블된 기간이어야 합니다.
- 두 번째 피연산자가 날짜이면 첫 번째 피연산자는 날짜 또는 날짜의 스트링 표시이어야 합니다.
- 첫 번째 피연산자가 시간이면 두 번째 피연산자는 시간, 시간 기간, 시간의 스트링 표시 또는 시, 분, 초의 레이블된 기간이어야 합니다.
- 두 번째 피연산자가 시간이면 첫 번째 피연산자는 시간 또는 시간의 스트링 표시이어야 합니다.
- 첫 번째 피연산자가 시간소인이면 두 번째 피연산자는 시간소인, 시간소인의 스트링 표시 또는 기간이어야 합니다.
- 두 번째 피연산자가 시간소인이면 첫 번째 피연산자는 시간소인 또는 시간소인의 스트링 표시이어야 합니다.
- 빼기 연산자의 피연산자는 매개변수 마커가 될 수 없습니다.

날짜 산술

날짜는 빼거나 늘리거나 줄일 수 있습니다.

날짜 빼기: 한 날짜(DATE1)에서 다른 날짜(DATE2)를 뺀 결과는 두 날짜 사이의 년, 월, 일의 숫자를 지정하는 날짜 기간입니다. 결과의 자료 유형은 DECIMAL(8,0) 입니다. DATE1이 DATE2 보다 크거나 같으면 DATE1에서 DATE2를 뺍니다. 그러나 DATE1이 DATE2 보다 작으면 DATE2에서 DATE1을 빼고 결과의 부호는 음이 됩니다. 다음 프로시듀어는 연산 RESULT = DATE1 - DATE2에 포함된 단계를 구분하여 설명한 것입니다.

```
DAY(DATE2) <= DAY(DATE1)이면
    DAY(RESULT) = DAY(DATE1) - DAY(DATE2)입니다.

DAY(DATE2) > DAY(DATE1)이면
     DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)이고
       여기서 N = MONTH(DATE2)의 마지막 날입니다.
    MONTH(DATE2)은 1씩 증가합니다.

MONTH(DATE2) <= MONTH(DATE1)이면
    MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2)입니다.

MONTH(DATE2) > MONTH(DATE1)이면
    MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2)이고
    YEAR(DATE2)는 1씩 증가합니다.

YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
```

예를 들어, DATE('3/15/2000') - '12/31/1999'의 결과는 215(또는 기간 0년 2개월 15일)입니다.

날짜 늘리기와 줄이기: 기간을 날짜에 더하거나 날짜에서 기간을 뺀 결과는 날짜입니다(이 연산을 위해 월은 달력 페이지와 같게 표시됩니다. 날짜에 월을 더하면 날짜가 표시되는 페이지부터 시작하여 달력의 페이지를 넘기는 것과 같습니다). 결과는 1년 1월 1일과 9999년 12월 31일 사이(포함)에 있어야 합니다. 년의 기간을 더하거나 빼면 날짜의 년 부분만 영향을 받습니다. 결과가 평년의 2월 29일이 아닌 한, 일과 마찬가지로 월도 변경되지 않습니다. 이 경우 일은 28일로 변경되고 SQLCA의 SQLWARN6은 'W'로 설정되어 월말을 조정하도록 지정합니다.

마찬가지로 월의 기간을 더하거나 빼면 필요한 대로 월과 년만 영향을 받습니다. 결과가 유효하지 않는 한(예: 9월 31일) 날짜의 일 부분은 변경되지 않습니다. 이 경우 일은 그 달의 마지막 날짜로 설정되고 SQLCA의 SQLWARN6은 'W'로 설정되어 월말을 조정하도록 표시합니다.

일의 기간을 더하거나 빼면 날짜의 일 부분은 물론 영향을 받고 월과 년도 영향을 받을 수 있습니다. DAYS의 레이블된 기간을 더하면 월말이 조정되지 않습니다.

날짜 기간이 양 또는 음인지 여부에 따라 날짜에 더하거나 뺄 수 있습니다. 레이블된 기간의 경우 결과는 유효한 날짜이고 월말 조정이 필요할 때마다 SQLCA에서 경고 인디케이터가 설정됩니다.

양의 날짜 기간을 날짜에 더하거나 음의 날짜 기간을 날짜에서 뺄 경우 지정된 수의 년, 월, 일 만큼 그 순서로 날짜가 증가됩니다. 따라서 DATE1 + X는 다음 표현식과 같습니다. 여기서 X는 양의 DECIMAL(8,0) 숫자입니다.

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS
```

양의 날짜 기간을 날짜에서 빼거나 음의 날짜 기간을 날짜에 더할 경우 지정된 수의 일, 월, 년 만큼 그 순서로 날짜가 줄어듭니다. 따라서 DATE1 - X는 다음 표현식과 같습니다. 여기서 X는 양의 DECIMAL(8,0) 숫자입니다.

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS
```

날짜에 기간을 더하는 경우 주어진 날짜에 1개월을 더하면 그 날짜가 다음 달에 없지 않는 한 한 달후의 같은 날짜가 됩니다. 그런 경우 날짜는 다음 달의 마지막 날 날짜로 설정됩니다. 예를 들어, 1월 28일에 한 달을 더하면 2월 28일이 되고 1월 29, 30, 31일에 한 달을 더하면 2월 28일이나 윤년일 경우에는 2월 29일이 됩니다.

주: 주어진 날짜에 한 달 이상을 더하고 같은 달 수 만큼 결과에서 빼면 마지막 날짜가 반드시 처음 날짜가 되는 것은 아닙니다.

시간 산술

시간은 빼거나 늘리거나 줄일 수 있습니다.

시간 빼기: 한 시간(TIME1)에서 다른 시간(TIME2)을 뺀 결과는 두 시간 사이의 시, 분, 초의 숫자를 지정하는 시간 기간입니다. 결과의 자료 유형은 DECIMAL(6,0)입니다. TIME1이 TIME2 보다 크거나 같으면 TIME1에서 TIME2를 뺍니다. 그러나 TIME1이 TIME2 보다 작으면 TIME2에서 TIME1을 빼고 결과의 부호는 음이 됩니다. 다음 프로시듀어는 연산 RESULT = TIME1 - TIME2에 포함된 단계를 설명한 것입니다.

SECOND(TIME2) <= SECOND(TIME1)이면
    SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2)입니다.

SECOND(TIME2) > SECOND(TIME1)이면
    SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2)이고
    MINUTE(TIME2)은 1씩 증가합니다.

MINUTE(TIME2) <= MINUTE(TIME1)이면
    MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2)입니다.

MINUTE(TIME2) > MINUTE(TIME1)이면
    MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2)이고
    HOUR(TIME2)은 1씩 증가합니다.

HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).

예를 들어, TIME('11:02:26') - '00:32:56'의 결과는 102930(10시간, 29분, 30초)입니다.

시간 늘리기와 줄이기: 기간을 시간에 더하거나 시간에서 기간을 뺀 결과는 시간입니다. 시의 넘침이나 부족(underflow)은 삭제되므로 결과는 항상 시간입니다. 시의 기간을 더하거나 빼면 시간의 시 부분만 영향을 받습니다. 분과 초는 변경되지 않습니다.

마찬가지로 분의 기간을 더하거나 빼면 필요한 대로 분과 시만 영향을 받습니다. 시간의 초 부분은 변경되지 않습니다.

초의 기간을 더하거나 빼면 시간의 초 부분은 물론 영향을 받고 분과 시도 영향을 받을 수 있습니다.

시간 기간이 양 또는 음인지 여부에 따라 시간에 더하거나 뺄 수 있습니다. 결과는 시, 분, 초의 지정된 수 만큼 순서대로 늘리거나 줄인 시간입니다. TIME1 + X는 다음 표현식과 같습니다. 여기서 “X”는 DECIMAL(6,0) 숫자입니다.

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

시간소인 산술

시간소인은 빼거나 늘리거나 줄일 수 있습니다.

시간소인 빼기: 한 시간소인(TS1)에서 다른 시간소인(TS2)을 뺀 결과는 두 시간소인 사이의 년, 월, 일, 시, 분, 초의 숫자를 지정하는 시간소인 기간입니다. 결과의 자료 유형은 DECIMAL(20,6)입니다. TS1이 TS2 보다 크거나 같으면 TS1에서 TS2를 뺍니다. 그러나 TS1이 TS2 보다 작으면 TS2에서 TS1을 빼고 결과의 부호는 음이 됩니다. 다음 프로시듀어는 연산 RESULT = TS1 - TS2에 포함된 단계를 설명한 것입니다.

MICROSECOND(TS2) <= MICROSECOND(TS1)이면
MICROSECOND(RESULT) = MICROSECOND(TS1) -
MICROSECOND(TS2)입니다.

MICROSECOND(TS2) >MICROSECOND(TS1)이면
MICROSECOND(RESULT) = 1000000 +
MICROSECOND(TS1) - MICROSECOND(TS2)이고
SECOND(TS2)는 1씩 증가합니다.

시간소인의 초와 분 부분은 시간 빼기 규칙에 지정한 대로 뺍니다.

HOUR(TS2) <= HOUR(TS1)이면
HOUR(RESULT) = HOUR(TS1) - HOUR(TS2)입니다.

HOUR(TS2) > HOUR(TS1)이면
HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)이고
DAY(TS2)는 1씩 증가합니다.

시간소인의 날짜 부분은 날짜 빼기 규칙에 지정한 대로 뺍니다.

시간소인 늘리기와 줄이기: 기간을 시간소인에 더하거나 시간소인에서 기간을 뺀 결과는 시간소인입니다. 날짜와 시간 산술은, 시의 넘침이나 부족(underflow)이 결과의 날짜 부분으로 실행되고 이 결과는 유효한 날짜 범위 안에 있어야 하는 경우를 제외하고, 이전에 정의한 대로 수행됩니다. 마이크로초는 초로 넘침이 발생합니다.

연산 순서

괄호 안의 표현식을 먼저 계산합니다. 괄호로 계산 순서가 지정되지 않은 경우 지수화는 접두부 연산자(예: -, 단항 빼기) 다음에 그리고 곱하기와 나누기 전에 적용됩니다. 곱하기와 나누기는 더하기와 빼기 전에 적용됩니다. 같은 선행 레벨의 연산자는 왼쪽에서 오른쪽으로 적용됩니다. 다음 표에 모든 연산자의 우선순위가 표시됩니다.

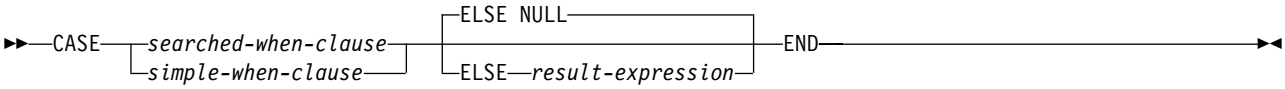
우선순위	연산자
1	+, -(부호 숫자 값으로 사용될 경우)
2	**
3	*, /, CONCAT, ||
4	+, -(두 피연산자 사이에 사용되는 경우)

예

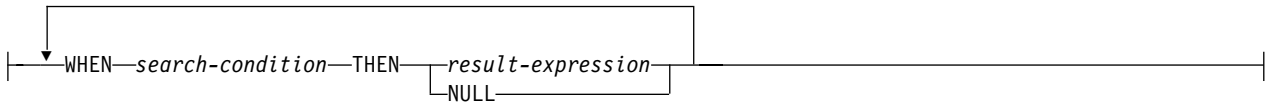
다음 예에서 연산자는 두 번째 행의 숫자 순서로 적용됩니다.

```
1.10 * (SALARY + BONUS) + SALARY / :VAR3
     2         1        4        3
```

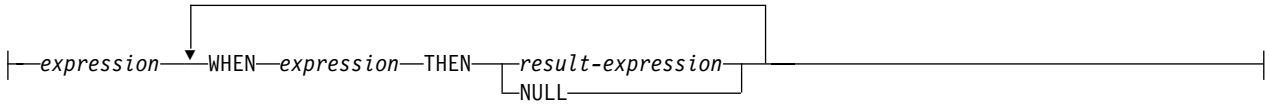
CASE 표현식



searched-when절:



simple-when절:



CASE 표현식을 사용하면 하나 이상의 조건에 따라 표현식을 선택할 수 있습니다. 일반적으로, case-expression의 값은 참으로 계산되는 첫 번째(가장 왼쪽) when절 다음의 *result-expression* 값입니다. when절이 참으로 계산되지 않고 ELSE 키워드가 표시되면 결과는 ELSE *result-expression*의 값이거나 NULL입니다. when절이 참으로 계산되지 않고 ELSE 키워드가 표시되지 않으면 결과는 NULL입니다. when절이(널이므로) 알 수 없는 것으로 계산되면 when절은 참이 아니고 따라서 when절이 거짓인 경우 같은 방법으로 처리됩니다.

*simple-when-clause*를 사용하는 경우 첫 번째 WHEN 키워드 이전의 *expresseion* 값은 WHEN 키워드 다음의 *expression* 값으로 같은지 테스트합니다. 따라서 첫 번째 WHEN 키워드 이전의 *expression* 자료 유형은 WHEN 키워드 다음의 각 *expression* 자료 유형과 호환될 수 있어야 합니다.

*result-expression*은 THEN이나 ELSE 키워드 다음에 오는 *expression*입니다. CASE 표현식에 적어도 하나의 *result-expression*이 있어야 합니다(NULL은 모든 경우에 지정될 수 있는 것이 아님). 모든 *result-expressions*에는 호환될 수 있는 자료 유형이 있어야 하고 여기서 결과의 속성은 93 페이지의 『결과 자료 유형에 대한 규칙』에 따라 판별됩니다.

CASE로 제공되는 함수성의 서브세트를 핸들하도록 특수화된 두 개의 함수, NULLIF와 COALESCE가 있습니다. 다음 표는 CASE 또는 이들 함수를 사용한 동일한 표현식을 나타냅니다.

표 21. 동일한 CASE 표현식

CASE 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)

예

- 부서 번호의 첫 번째 문자가 조직 안에서의 부서를 나타내면 CASE 표현식을 사용하여 각 사원이 속한 부서의 전체 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 년 수가 EMPLOYEE 표에 사용되어 교육 레벨을 알 수 있습니다. CASE 표현식은 이런 사항들을 그룹화하고 교육 레벨을 나타내는 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
```

- CASE 명령문 사용의 다른 공통되는 예는 0으로 나누기 오류에 대한 것입니다. 예를 들어, 다음 코드는 수수료에서 수입의 25% 이상을 벌지만 수수료를 완불하지 않는 사원을 찾습니다.

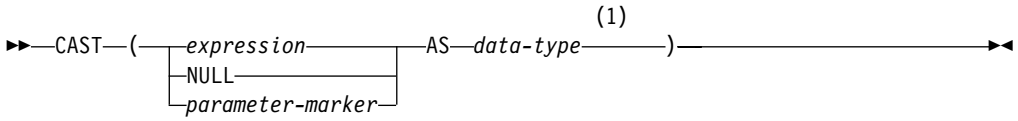
```
SELECT EMPNO, WORKDEPT, SALARY+COMM
FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
            ELSE COMM/SALARY
       END) > 0.25
```

- 다음 CASE 표현식은 동일합니다.

```
SELECT LASTNAME,
 CASE
 WHEN LASTNAME = 'Haas' THEN 'President'
 ...

SELECT LASTNAME,
 CASE LASTNAME
 WHEN 'Haas' THEN 'President'
 ...
```

CAST 스펙



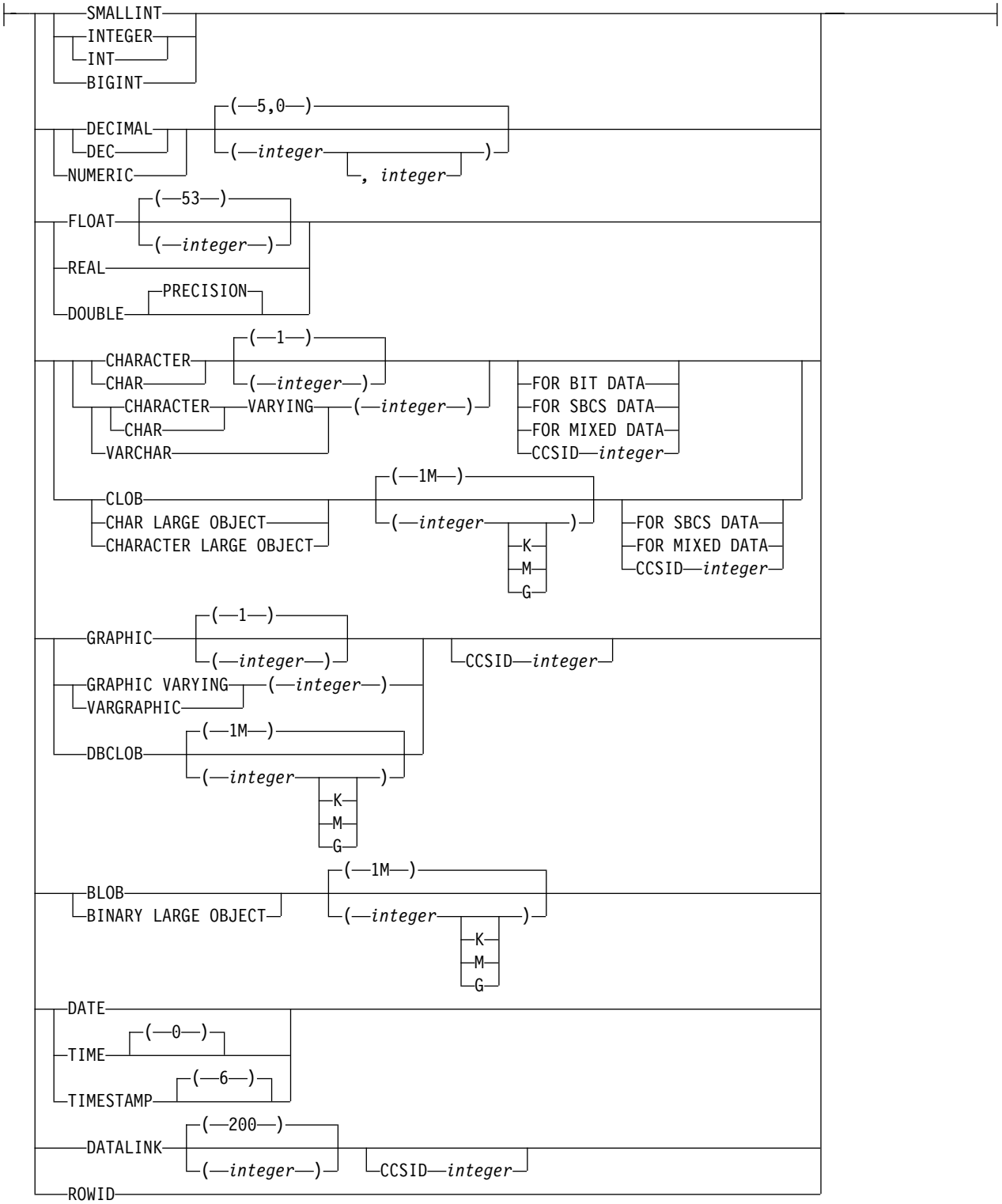
주:

1 자료 유형 이름은 규정될 수 있습니다. 자세한 내용은 47 페이지의 『명명 규칙』을 참조하십시오.

표현식

data-type:

built-in-type:



CAST 스펙은 캐스트 피연산자(첫 번째 피연산자) 캐스트를 *data-type*으로 지정한 유형에 리턴합니다. 두 피연산자의 자료 유형이 고유한 유형이면 명령문의 권한부여 ID로 보유된 권한에 고유한 유형에 대한 USAGE 권한이 포함되어야 합니다.

expression

캐스트 피연산자가(매개변수 마커나 NULL이 아닌) 표현식이면 결과는 지정된 목표 자료 유형으로 변환된 인수 값입니다.

지원되는 캐스트가 79 페이지의 표 11에 표시되고, 여기서 첫 번째 열은 캐스트 피연산자의 자료 유형(소스 자료 유형)을 나타내고 맨 위의 자료 유형은 CAST 스펙의 목표 자료 유형을 나타냅니다. 캐스트가 지원되지 않으면 오류가 발생합니다.

문자나 그래픽 스트링을 길이가 다른 문자나 그래픽 스트링으로 캐스트할 경우 후미 공백 이외의 것이 절단되면 경고가 리턴됩니다.

NULL

캐스트 피연산자가 키워드 NULL이면 결과는 *data-type*이 지정된 널값입니다.

parameter-marker

매개변수 마커(의문 부호 문자로 지정됨)는 표현식으로 정상적으로 간주되지만 특별한 의미가 있으므로 이 경우에는 따로 문서화됩니다. 캐스트 피연산자가 *parameter-marker*이면 지정된 *data-type*은(열에 지정할 때와 같은 규칙을 사용하여) 대체될 수 있는 약속으로 간주됩니다. 이런 매개변수 마커는 *typed-parameter-marker*로 간주됩니다. 입력된 매개변수 마커는 선택 리스트의 DESCRIBE나 열 지정을 위하여 입력된 다른 값과 같이 처리됩니다.

data-type

결과의 자료 유형을 지정합니다. 자료 유형이 규정되지 않으면 SQL 경로를 사용하여 해당되는 자료 유형을 찾습니다. *data-type*에 대한 설명은 541 페이지의 『CREATE TABLE』을 참조하십시오.

길이, 정밀도, 스케일 또는 CCSID 속성이 지정된 경우 지정된 속성이 사용됩니다. 길이, 정밀도 또는 스케일 속성이 지정되지 않은 경우 디폴트 값이 사용됩니다. 예를 들어, CHAR의 디폴트 길이는 1이고 DECIMAL의 디폴트 정밀도는 5이며 스케일은 0입니다. 다른 자료 유형의 디폴트 값은 541 페이지의 『CREATE TABLE』을 참조하십시오(오퍼레이팅 시스템의 이식성에 대해 부동 소수점 자료 유형을 지정할 경우 FLOAT 대신 REAL 또는 DOUBLE을 사용하십시오).

CCSID 속성이 지정되지 않은 경우 다음과 같습니다.

- *자료 유형*이 BLOB면 CCSID 65535가 사용됩니다.
- *expression*이 문자 스트링이고 *자료 유형*이 CHAR, VARCHAR 또는 CLOB인 경우 *expression*의 CCSID가 사용됩니다.
- *expression*이 문자 스트링이고 *data-type*이 GRAPHIC, VARGRAPHIC 또는 DBCLOB인 경우 *expression*의 CCSID가 사용됩니다.
- 그 이외의 경우에는 *data-type*에 대한 디폴트 CCSID가 사용됩니다.

지원되는 자료 유형에 대한 제한사항은 지정된 캐스트 피연산자에 따라 다릅니다.

- *expression*인 캐스트 피연산자의 경우에는 79 페이지의 표 11에서 캐스트 피연산자의 자료 유형에 따라 지원되는 목표 자료 유형을 참조하십시오.
- 키워드 NULL인 캐스트 피연산자의 경우 목표 자료 유형은 어떤 자료 유형도 될 수 있습니다.
- 매개변수 마커인 캐스트 피연산자의 경우에는 목표 자료 유형이 어떤 자료 유형도 될 수 있습니다. 자료 유형이 고유한 유형이면 매개변수 마커를 사용하는 어플리케이션에서 고유한 유형의 소스 자료 유형을 사용합니다.

자료 유형 사이에 지원되는 캐스트와 자료 유형으로의 캐스트 규칙에 대한 내용은 77 페이지의 『자료 유형 사이의 캐스트』를 참조하십시오.

예

- 어플리케이션은 EMPLOYEE 표 에서(DECIMAL(9,2)로 정의된) SALARY 열의 정수 부분에만 관심이 있습니다. 다음 CAST 스펙은 SALARY 열을 INTEGER로 변환합니다.

```
SELECT EMPNO, CAST(SALARY AS INTEGER)
  FROM EMPLOYEE
```

- 두 개의 고유한 유형이 있다고 가정합니다. T_AGE의 소스는 SMALLINT였고 PERSONNEL 표에 있는 AGE 열의 자료 유형입니다. R_YEAR의 소스는 INTEGER였고 같은 표에 있는 RETIRE_YEAR 열의 자료 유형입니다. 다음 UPDATE 명령문이 준비될 수 있습니다.

```
UPDATE PERSONNEL SET RETIRE_YEAR = ?
  WHERE AGE = CAST( ? AS T_AGE )
```

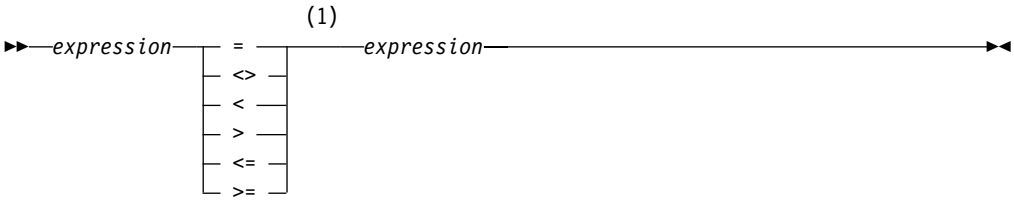
술부

술부는 주어진 행이나 그룹에 대해 참, 거짓 또는 알 수 없는 조건을 지정합니다. 다음 규칙은 모든 유형의 술부에 적용됩니다.

- 술부에 지정된 모든 값은 호환될 수 있어야 합니다.
- 둘 이상의 피연산자가 포함된 술부 피연산자의 CCSID 변환은 91 페이지의 『비교를 위한 변환 규칙』에 따라 수행됩니다.
- 자료 링크 값의 사용은 NULL 술부로 제한됩니다.

기본 술부

28. 다음과 같은 양식의 비교 연산자도 기본 및 규정된 술부에서 지원됩니다. !=, !<, !>, ¬=,¬< 및 ¬>가 지원됩니다. 이러한 각 제품에 고유한 양식의 비교 연산자는 해당 연산자를 사용하는 기존의 SQL문을 지원하는 경우에만 사용하도록 하고 새로운 SQL문을 작성할 때 사용하는 것은 권장되지



주:

1 다른 비교 연산자도 지원됩니다.[28]

*기본 술부*는 두 값을 비교합니다. 술부의 피연산자에 SBCS 자료나 혼합 자료가 들어 있고 명령문이 실행될 때 유효한 정렬 순서가 *HEX가 아니면 피연산자의 가중치를 사용하여 피연산자가 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

둘 중의 한 피연산자의 값이 널이면 술부의 결과는 값이 지정되지 않습니다. 그렇지 않으면 결과가 참이거나 거짓입니다.

값 *x*와 *y*의 경우:

술부 다음 경우에만 참...

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

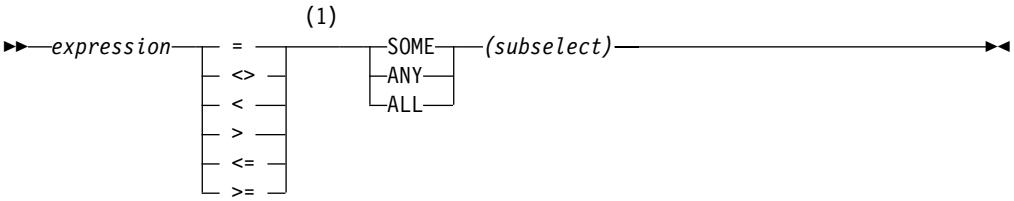
*x***<=** *y* *x*가 *y* 보다 작거나 같음

예

```
EMPNO = '528671'
PRTSTAFF <> :VAR1
SALARY + BONUS + COMM < 20000
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

일정한 양의 술부

않습니다. 일부 키보드는 부정(¬) 기호에 16진 값을 사용해야 합니다. 16진 값은 사용되는 키보드에 따라 다릅니다. 부정 기호(¬) 또는 특정 국가에서 부정 기호로 사용되는 문자는 한 데이터베이스 서버에서 다른 데이터베이스 서버로 전달된 명령문에서 구문 분석 오류를 발생시킬 수 있습니다. 명령문이 소스 CCSID와 목표 CCSID를 결합하여 문자 변환을 하는 경우 문제가 발생합니다. 이러한 문제를 방지하려면 부정 부호가 들어 있는 연산자를 그에 해당되는 다른 연산자로 대체합니다. 예를 들어 '¬='는 '<>'로, '¬>'는 '<='로 그리고 '¬<'는 '>='로 대체합니다.



주:

1 다른 비교 연산자도 지원됩니다. [28]

일정한 양의 술부는 값과 값 세트를 비교합니다.

subselect는 하나의 결과 열을 지정해야 하고 널 또는 널이 아닌 값의 숫자를 리턴할 수 있습니다. 술부의 피연산자에 SBCS 자료나 혼합 자료가 들어 있고 명령문이 실행될 때 유효한 정렬 순서가 *HEX가 아니면 피연산자의 가중치를 사용하여 피연산자가 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

ALL이 지정되면 술부의 결과는 다음과 같습니다.

- subselect의 결과가 비어 있거나 subselect로 리턴되는 모든 값에 대해 지정된 관계가 참이면 참
- subselect로 리턴되는 적어도 하나의 값에 대해 지정된 관계가 거짓이면 거짓
- subselect로 리턴되는 값에 대해 지정된 관계가 거짓이 아니고 널값으로 인해 적어도 하나의 비교가 값이 지정되지 않으면 값이 지정되지 않음

SOME이나 ANY가 지정되면 술부의 결과는 다음과 같습니다.

- subselect로 리턴되는 적어도 하나의 값에 대해 지정된 관계가 참이면 참
- subselect의 결과가 비어 있거나 subselect로 리턴되는 모든 값에 대해 지정된 관계가 거짓이면 거짓
- subselect로 리턴되는 값에 대해 지정된 관계가 참이 아니고 널값으로 인해 적어도 하나의 비교가 값이 지정되지 않으면 값이 지정되지 않음

예

다음 예를 참조할 때는 아래 표를 사용하십시오.

표 22. 표 설명.

TBLA	COLA	TBLB	COLB
	1		2
	2		3
	3		
	4		
	널값		

- 다음 선택 명령문의 결과는 2,3입니다. subselect는 (2,3)을 리턴합니다. 행 2와 3의 COLA는 적어도 이들 값 중 하나와 같습니다.

  ```
  SELECT * FROM TBLA WHERE COLA = ANY(SELECT COLB FROM TBLB)
  ```

- 다음 선택 명령문의 결과는 3,4입니다. subselect는 (2,3)을 리턴합니다. 행 3과 4의 COLA는 적어도 이들 값 중 하나 보다 큽니다.

  ```
  SELECT * FROM TBLA WHERE COLA > ANY(SELECT COLB FROM TBLB)
  ```

- 다음 선택 명령문의 결과는 4입니다. subselect는 (2,3)을 리턴합니다. 행 4의 COLA는 이들 값보다 큰 유일한 값입니다.

  ```
  SELECT * FROM TBLA WHERE COLA > ALL(SELECT COLB FROM TBLB)
  ```

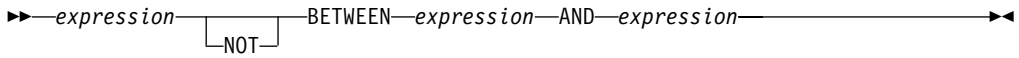
- 다음 선택 명령문의 결과는 1,2,3,4 및 널입니다. subselect의 결과는 비어 있습니다. 따라서 TBLA의 모든 행에 대해 술부는 참입니다.

  ```
  SELECT * FROM TBLA WHERE COLA > ALL(SELECT COLB FROM TBLB WHERE COLB<0)
  ```

- 다음 선택 명령문의 결과는 빈 세트입니다. subselect의 결과는 비어 있습니다. 따라서 TBLA의 모든 행에 대해 술부는 거짓입니다.

  ```
  SELECT * FROM TBLA WHERE COLA > ANY(SELECT COLB FROM TBLB WHERE COLB<0)
  ```

BETWEEN 술부



BETWEEN 술부는 값과 값 범위를 비교합니다. 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 BETWEEN 술부에 SBCS 자료나 혼합 자료가 포함되어 있으면 값 대신 스트링의 가중치가 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

BETWEEN 술부에서:

```
value1 BETWEEN value2 AND value3
```

는 다음의 탐색 조건과 논리적으로 동일합니다.

```
value1 >= value2 AND value1 <= value3
```

BETWEEN 술부에서:

```
value1 NOT BETWEEN value2 AND value3
```

는 다음의 탐색 조건과 동일합니다.

```
NOT(value1 BETWEEN value2 AND value3)은 곧
value1 < value2 OR value1 > value3입니다.
```

BETWEEN 술부의 피연산자가 CCSID가 다른 스트링이면 위의 논리적으로 동일한 탐색 조건이 지정된 것처럼 피연산자가 변환됩니다.

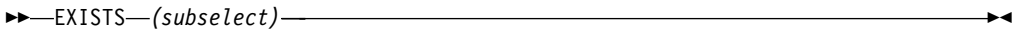
datetime 값의 혼합과 datetime 값의 스트링 표시가 제공되면 모든 값은 datetime 피연산자의 자료 유형으로 변환됩니다.

예

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000

SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

EXISTS 술부



EXISTS 술부는 특정한 행의 존재에 대해 테스트합니다. subselect는 열 수를 지정합니다.

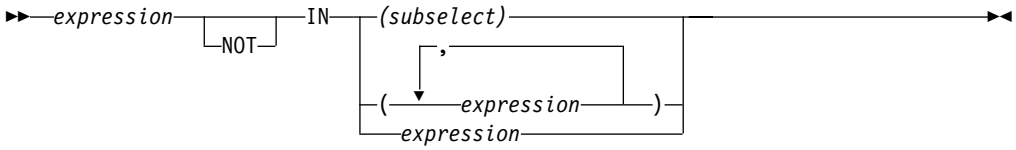
- 결과는 subselect로 지정한 행 수가 0이 아닐 경우에만 참입니다.
- 결과는 subselect로 지정한 행 수가 0일 경우에만 거짓입니다.
- 결과가 값이 지정되지 않음은 될 수 없습니다.

subselect로 리턴되는 값은 무시됩니다.

예

```
EXISTS (SELECT * FROM EMPLOYEE WHERE SALARY > 60000)
```

IN 술부



IN 술부는 값과 값 세트를 비교합니다. 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 IN 술부에 SBCS 자료나 혼합 자료가 포함되어 있으면 실제 값 대신 스트링의 가중치가 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

subselect 형식에서 subselect는 하나의 결과 열을 지정해야 하고 널 또는 널이 아닌 값의 숫자를 리턴할 수 있습니다.

다음 형식의 IN 술부:

expression **IN** (*subselect*)

는 다음 형식의 일정한 양의 술부와 같습니다.

expression = **ANY** (*subselect*)

다음 형식의 IN 술부:

expression **NOT IN** (*subselect*)

는 다음 형식의 일정한 양의 술부와 같습니다.

expression <> **ALL** (*subselect*)

다음 형식의 IN 술부:

expression **IN** *expression*

는 다음 형식의 기본 술부와 같습니다.

expression = *expression*

다음 형식의 IN 술부:

expression **IN** (*value1, value2, ..., valueN*)

는 다음과 논리적으로 동일합니다.

expression **IN** (**SELECT** * **FROM** R)

T가 행이 하나인 표라고 가정합니다. R은 다음 fullselect로 형성된 임시표입니다.

```
SELECT value1 FROM T
  UNION
SELECT value2 FROM T
  UNION
    .
    .
    .
  UNION
SELECT valueN FROM T
```

각 호스트 변수는 호스트 구조나 변수의 선언 규칙에 따라 서술된 구조 또는 변수를 식별해야 합니다.

IN 술부 피연산자의 자료 유형이나 속성이 다르면 IN 술부 평가를 위해 자료 유형 판별에 사용된 규칙은 UNION과 UNION ALL의 규칙입니다. 설명은 93 페이지의 『결과 자료 유형에 대한 규칙』을 참조하십시오.

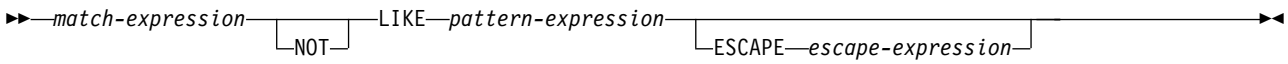
IN 술부의 피연산자가 CCSID가 다른 스트링이면 변환되는 피연산자를 판별하기 위해 사용되는 규칙은 스트링을 결합하는 조작의 규칙입니다. 설명은 97 페이지의 『스트링 결합 조작을 위한 변환 규칙』을 참조하십시오.

예

```
DEPTNO IN ('D01', 'B01', 'C01')

EMPNO IN(SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

LIKE 술부



LIKE 술부는 특정한 패턴의 스트링을 탐색합니다. 패턴은 밑줄이나 퍼센트 기호에 특별한 의미가 있는 스트링에 의해 지정됩니다. 패턴의 후미 공백은 패턴의 일부입니다.

인수 값 중 어느 하나가 널(null)인 경우 LIKE 술부의 결과는 알 수 없습니다.

일치 표현식, *패턴 표현식* 및 *이탈 표현식*은 스트링을 식별해야 합니다. *일치 표현식*, *패턴 표현식* 및 *이탈 표현식*은 모두 2진 스트링이거나 2진 스트링이 아니어야 합니다. 이 세 인수에는 문자 스트링과 그래픽 스트링이 섞여 있을 수 있습니다.

어떠한 표현식도 고유한 유형을 생성할 수 없습니다. 그러나 소스 유형으로 고유한 유형을 캐스트하는 함수일 수는 있습니다.

명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 LIKE 술부에 SBCS 자료나 혼합 자료가 포함되어 있으면 실제 값 대신 스트링의 가중치가 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

문자 스트링의 경우 다음에 나오는 용어 **문자**, **퍼센트 기호** 및 **밑줄**은 1바이트 문자를 참조합니다. 그래픽 스트링의 경우 위의 용어는 2바이트 문자 또는 UCS-2 문자를 참조합니다. 2진 스트링이라는 용어는 1바이트 문자의 코드 포인트를 말합니다.

일치 표현식
: 특정 문자 패턴에 대한 일치 여부를 알기 위해 조사될 스트링을 지정하는 표현식.

LIKE *패턴 표현식*
: 일치될 스트링을 지정하는 표현식.

  패턴에 대한 간단한 설명

  패턴은 다음 경우에서 일치 표현식의 값에 대한 일치 기준을 지정하는 데 사용됩니다.

  - 밑줄 부호(_)는 단일 문자를 나타냅니다.
  - 퍼센트 기호(%)는 0이나 문자 스트링을 나타냅니다.
  - 그외 다른 문자들은 문자 자체를 나타냅니다.

*패턴 표현식*에 밑줄이나 퍼센트 문자가 들어가야 하는 경우 패턴에서 밑줄이나 퍼센트에 선행하는 문자를 지정하는 데 *이탈 표현식*이 사용됩니다.

패턴에 대한 자세한 설명

*이탈 표현식*에 관해서는 뒤에 나오는 설명을 참조하십시오.

*m*은 *일치 표현식* 값을 나타내며 *p*는 *패턴 표현식* 값을 나타냅니다. *p* 스트링은 최소 서브스트링 지정자의 순서로 해석되므로 *p*의 각 문자는 정확히 하나의 서브스트링 지정자의 일부입니다. 서브스트링 지정자는 밑줄, 퍼센트 기호 또는 밑줄이나 퍼센트 기호가 아닌 비어 있지 않은 문자 순서입니다.

*m*이나 *p*가 널 값이면 술부의 결과는 알 수 없으며, 그렇지 않으면 술부의 결과가 참이거나 거짓입니다. *m*과 *p*가 둘 다 빈 스트링이거나 다음과 같이 서브스트링으로 *m*이 분할되면 결과는 참입니다.

- *m*의 서브스트링은 0 또는 그 이상의 연속되는 문자 순서이고 *m*의 각 문자는 정확히 하나의 서브스트링의 일부입니다.
- *n* 번째 서브스트링 지정자가 밑줄이면 *m*의 *n* 번째 서브스트링은 임의의 단일 문자입니다.
- *n* 번째 서브스트링 지정자가 퍼센트 기호이면 *m*의 *n* 번째 서브스트링은 0 또는 그 이상의 문자 순서입니다.
- *n* 번째 서브스트링 지정자가 밑줄도 퍼센트 기호도 아니면 *m*의 *n* 번째 서브스트링은 그 서브스트링 지정자와 동일하고 서브스트링 지정자와 길이도 같습니다.
- *m*의 서브스트링 수는 서브스트링 지정자의 수와 같습니다.

*y*가 빈 스트링이고 *m*이 빈 스트링이 아닌 경우에도 해당되며 결과는 거짓입니다. 마찬가지로 *m*이 빈 스트링이고 *p*가 퍼센트 기호 이외의 것으로 구성되어 빈 스트링이 아닌 경우에도 해당되며 결과는 거짓입니다.

술부 *m* NOT LIKE *p*는 탐색 조건 NOT(*m* LIKE *p*)와 같습니다.

필요에 따라 *일치 표현식*, *패턴 표현식* 및 *이탈 표현식*은 *일치 표현식*과 *패턴 표현식*간의 호환 가능한 CCSID로 변환됩니다.

혼합 자료

표현식이 혼합 자료이면 표현식에 2바이트 문자가 들어갈 수 있고 패턴에 SBCS와 DBCS 문자가 둘 다 포함될 수 있습니다. 이러한 경우 *p*의 특수 문자는 다음과 같이 해석됩니다.

- SBCS 밑줄은 하나의 SBCS 문자를 참조합니다.
- DBCS 밑줄은 하나의 DBCS 문자를 참조합니다.

- 퍼센트 기호(SBCS 또는 DBCS)는 SBCS 또는 DBCS로 된 임의 유형의 임의 숫자의 문자를 참조합니다.
- *일치 표현식*과 *패턴 표현식*의 중복 시프트는 무시됩니다.

UCS-2 자료

표현식이 UCS-2 그래픽 자료이면 패턴에 UCS-2 밑줄과 퍼센트 기호에 지원되는 코드점 중 하나 또는 둘 다를 포함할 수 있습니다. UCS-2 밑줄에 지원되는 코드점은 X'005F'과 X'FF3F'입니다. UCS-2 퍼센트 기호에 지원되는 코드점은 X'0025'와 X'FF05'입니다.

매개변수 마커

LIKE 술부에 지정한 패턴이 매개변수 마커이고 매개변수 마커를 대체하는 데 고정 길이 문자 호스트 변수를 사용하는 경우 호스트 변수에 올바른 길이의 값을 지정하십시오. 올바른 길이를 지정하지 않으면 SELECT가 의도한 결과를 리턴하지 않습니다.

예를 들어, 호스트 변수를 CHAR(10)으로 정의하고, 호스트 변수 값으로 WYSE%를 지정하는 경우 지정에서 호스트 변수는 공백으로 채워집니다. 사용되는 패턴은 다음과 같습니다.

```
'WYSE%     '
```

이 패턴에서는 데이터베이스 관리자가 WYSE로 시작하고 5개의 공백으로 끝나는 모든 값을 탐색해야 합니다. 'WYSE'로 시작하는 값만을 찾으려면 호스트 변수에 'WSYE%%%%%%'를 지정해야 합니다.

ESCAPE *이탈 표현식*

패턴 표현식에서 밑줄(_)과 퍼센트(%) 문자의 특별한 의미를 수정하는 데 사용되는 문자를 지정하는 표현식. ESCAPE절을 사용하여 LIKE 술부를 실제 퍼센트와 밑줄 문자가 들어 있는 값을 일치시키는데 사용할 수 있습니다. 다음 규칙은 ESCAPE 절 및 *이탈 표현식* 사용에 적용됩니다.

- *이탈 표현식*은 길이가 1인 스트링이어야 합니다.[29]
- *패턴 표현식*에는 그 다음에 이탈 문자, 퍼센트 또는 밑줄이 나오는 경우를 제외하고는 이탈 문자가 들어갈 수 없습니다. 예를 들어 '+'가 이탈 문자이면 *패턴 표현식*에 '++', '+_' 또는 '+%' 이외에 '+'가 나오는 경우 오류입니다.
- *이탈 표현식*은 매개변수 마커가 될 수 있습니다.

다음 예에서는 이탈 문자가 연속적으로 나올 때 미치는 영향을 보여줍니다. 여기에서 이탈 문자는 더하기 부호(+)입니다.

패턴 스트링...	실제 패턴...

29. 널로 끝나는 경우 길이가 2인 C 문자 스트링 변수를 지정할 수 있습니다.

+%	퍼센트 기호
++%	더하기 부호와 그 뒤에 0 또는 그 이상의 임의의 문자가 나옴
+++%	더하기 부호와 그 뒤에 퍼센트 기호가 나옴

예

예 1: PROJECT 표의 PROJNAME 열의 어느 곳에나 나오는 'SYSTEMS' 스트링을 탐색합니다.

```
SELECT PROJNAME
  FROM PROJECT
  WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
```

예 2: EMPLOYEE 표의 FIRSTNM 열에서 정확히 첫문자가 두 문자 긴 'J'인 스트링을 탐색합니다.

```
SELECT FIRSTNME
     FROM EMPLOYEE
     WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
```

예 3: EMPLOYEE 표의 FIRSTNME 열에서 첫 문자가 'J'인 임의의 길이의 스트링을 탐색합니다.

```
SELECT FIRSTNME
     FROM EMPLOYEE
     WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
```

예 4: 다음 예를 참조하십시오.

```
SELECT *
     FROM TABLEY
     WHERE C1 LIKE 'AAAA+%BBB%' ESCAPE '+'
```

'+'는 이탈 문자이고 'AAAA%BBB'로 시작되는 스트링 찾기를 나타냅니다. '+%'는 패턴에서 한 번의 '%' 발생으로 해석됩니다.

예 5: 소스 자료 유형이 CHAR(5)인 ZIP_TYPE이라는 이름의 고유한 유형이 있고, 표 TABLEY에 자료 유형이 ZIP_TYPE인 ADDRZIP 열이 있다고 가정합니다. 다음 명령문에서는 우편 번호(ADDRZIP)가 '9555'로 시작되는 행을 선택합니다.

```
SELECT *
  FROM TABLEY
  WHERE CHAR(ADDRZIP) LIKE '9555%'
```

예 6: 샘플 표 EMP_RESUME의 RESUME 열이 CLOB로 정의됩니다. 호스트 변수 LASTNAME에 'JONES'라는 값이 들어 있으면 다음 명령문은 스트링 JONES가 열에 나올 때 RESUME 열을 선택합니다.

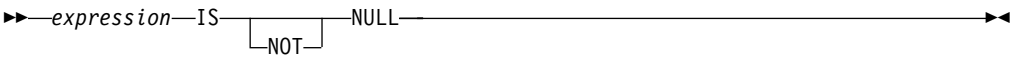
```
SELECT RESUME
  FROM EMP_RESUME
  WHERE RESUME LIKE '%'||LASTNAME||'%'
```

예 7: EBCDIC 예가 나오는 다음 표에서 COL1은 혼합 자료로 가정합니다. 표는 첫 번째 열의 술부가 두 번째 열의 COL1 값을 사용하여 평가되는 경우의 결과를 나타냅니다.

술어	COL1 값	결과
WHERE COL1 LIKE 'aaa SO AB%C SI'	'aaa SO ABDZC SI'	참
	'aaa SO AB SI dzx SO C SI'	참
WHERE COL1 LIKE 'aaa SO AB SI % SO C SI'		
WHERE COL1 LIKE 'a% SO C SI'	'a SO C SI'	참
	'ax SO C SI'	참
	'ab SO DE SI fg SO C SI'	참
WHERE COL1 LIKE 'a _ SO C SI'	'a% SO C SI'	참
	'a SO XC SI'	거짓
WHERE COL1 LIKE 'a SO __ C SI'	'a SO XC SI'	참
	'ax SO C SI'	거짓
WHERE COL1 LIKE ' SO SI'	Empty string	참
WHERE COL1 LIKE 'ab SO C SI _'	'ab SO C SI d'	참
	'ab SO SI SO C SI d'	거짓

RV3F001-0

NULL 술부



NULL 술부는 널값을 테스트합니다.

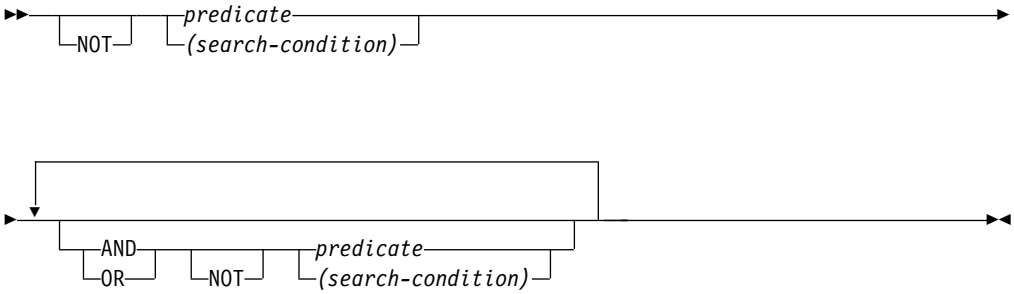
NULL 술부의 결과는 값이 지정될 수 없습니다. 표현식의 값이 널이면 결과는 참입니다. 값이 널이 아니면 결과는 거짓입니다. NOT이 지정되면 결과는 반전됩니다.

예

```
EMPLOYEE.PHONE IS NULL

SALARY IS NOT NULL
```

탐색 조건



*탐색 조건*은 주어진 행이나 그룹에 대해 참, 거짓 또는 알 수 없는 조건을 지정합니다.

탐색 조건의 결과는 지정된 각 술부의 결과에 대해 지정된 *논리 연산자*(AND, OR, NOT)의 어플리케이션에 의해 파생됩니다. 논리 연산자가 지정되지 않으면 탐색 조건의 결과는 지정된 술부의 결과입니다.

AND와 OR는 P와 Q가 술부인 다음 표에 정의되어 있습니다.

표 23. AND와 OR의 진리 표

P	**Q**	**P AND Q**	**P OR Q**
참	참	참	참
참	거짓	거짓	참
참	값이 지정되지 않음	값이 지정되지 않음	참
거짓	참	거짓	참
거짓	거짓	거짓	거짓
거짓	값이 지정되지 않음	거짓	값이 지정되지 않음
값이 지정되지 않음	참	값이 지정되지 않음	참
값이 지정되지 않음	거짓	거짓	값이 지정되지 않음
값이 지정되지 않음	값이 지정되지 않음	값이 지정되지 않음	값이 지정되지 않음

NOT(참)은 거짓이고 NOT(거짓)은 참이며 NOT(값이 지정되지 않음)은 값이 지정되지 않음입니다.

괄호 안의 탐색 조건을 먼저 평가합니다. 평가 순서가 괄호로 지정되지 않으면 AND 앞에 NOT이 적용되고 OR 앞에 AND가 적용됩니다. 같은 선행 레벨의 연산자가 평가되는 순서 탐색 조건을 최적화하기 위해 정의되지 않았습니다.

예

예에서 두 번째 행의 숫자는 연산자가 평가되는 순서를 표시합니다.

예 1

```
MAJPROJ = 'MA2100' AND DEPTNO = 'D11'  OR  DEPTNO = 'B03' OR DEPTNO = 'E11'
                     1                   2 or 3            2 or 3
```

예 2

```
MAJPROJ = 'MA2100'  AND  (DEPTNO = 'D11'  OR  DEPTNO = 'B03')  OR  DEPTNO = 'E11'
                   2                     1                     3
```

제 3 장 내장 함수

*내장 함수*는 iSeries용 DB2 UDB에서 제공되는 함수입니다. 내장 함수는 괄호로 묶인 하나 이상의 피연산자 앞에 함수명으로 표시됩니다. 함수의 피연산자를 *인수*라고 하고, 각각의 인수는 *표현식*에 의해 지정됩니다. 함수의 결과는 인수를 함수의 연산에 적용하여 나온 단일 값입니다.

*내장 함수*는 스키마 QSYS2의 일부입니다. *내장 함수*는 스키마 이름과 함께 또는 스키마 이름 없이 호출할 수 있습니다. 스키마 이름이 함수 이름을 규정하는지 여부와는 상관 없이, 데이터베이스 관리자는 함수 분석을 사용하여 사용할 함수를 결정합니다. 함수 및 함수 분석 프로세스에 대한 자세한 내용은 123 페이지의 『함수 분석』을 참조하십시오.

*내장 함수*는 *열 함수* 또는 *스칼라 함수*로 분류됩니다. 열 함수의 인수는 값의 집합입니다. 스칼라 함수의 인수는 단일 값입니다.

SQL 구문에서 용어 *함수*는 표현식 정의에서만 사용됩니다. 그러므로 함수는 표현식이 사용될 수 있는 곳에서만 사용될 수 있습니다. 추가 제한사항은 다음 섹션과 335 페이지의 제 4 장 『조회』에서 지정된 것처럼 열 함수 사용에 적용합니다.

다음 표는 다양한 유형의 *내장 함수*를 나열합니다.

표 24. 열 함수

163 페이지의 『AVG』	숫자들의 평균 값을 리턴합니다.
165 페이지의 『COUNT』	행이나 값의 집합에 있는 행 또는 값의 수를 리턴합니다.
167 페이지의 『COUNT_BIG』	행이나 값의 집합에 있는 행 또는 값의 수를 리턴합니다. 결과가 정수의 최대값보다 클 수 있는 경우를 제외하고는 COUNT와 비슷합니다.
169 페이지의 『MAX』	그룹의 값 집합 중 최대값을 리턴합니다.
170 페이지의 『MIN』	그룹의 값 집합 중 최소값을 리턴합니다.
171 페이지의 『STDDEV 또는 STDDEV_POP』	수 집합의 표준 편차(/n)를 리턴합니다.
172 페이지의 『SUM』	숫자들의 합계를 리턴합니다.
173 페이지의 『VAR_POP 또는 VARIANCE 또는 VAR』	수 집합의 편차(/n)를 리턴합니다.

표 25. 캐스트 스칼라 함수

182 페이지의 『BIGINT』	수를 큰 정수로 표시하여 리턴합니다.
184 페이지의 『BLOB』	임의의 유형의 스트링을 BLOB로 표시하여 리턴합니다.
187 페이지의 『CHAR』	값을 CHARACTER로 표시하여 리턴합니다.
193 페이지의 『CLOB』	값을 CLOB로 표시하여 리턴합니다.
204 페이지의 『DATE』	값으로부터 DATE를 리턴합니다.
212 페이지의 『DBCLOB』	스트링 표현식을 DBCLOB로 표시하여 리턴합니다.

표 25. 캐스트 스칼라 함수 (계속)

215 페이지의『DECIMAL 또는 DEC』	수를 DECIMAL로 표시하여 리턴합니다.
230 페이지의『DOUBLE_PRECISION 또는 DOUBLE』	수를 DOUBLE_PRECISION으로 표시하여 리턴합니다.
233 페이지의『FLOAT』	수를 FLOAT로 표시하여 리턴합니다.
235 페이지의『GRAPHIC』	스트링 표현식을 GRAPHIC으로 표시하여 리턴합니다.
246 페이지의『INTEGER 또는 INT』	수를 INTEGER로 표시하여 리턴합니다.
284 페이지의『REAL』	수를 REAL로 표시하여 리턴합니다.
287 페이지의『ROWID』	값으로부터 Row ID를 리턴합니다.
294 페이지의『SMALLINT』	수를 SMALLINT로 표시하여 리턴합니다.
305 페이지의『TIME』	값으로부터 TIME을 리턴합니다.
306 페이지의『TIMESTAMP』	값 또는 값 쌍에서 TIMESTAMP를 리턴합니다.
320 페이지의『VARCHAR』	값을 VARCHAR로 표시하여 리턴합니다.
324 페이지의『VARGRAPHIC』	스트링 표현식을 VARGRAPHIC으로 표시하여 리턴합니다.
331 페이지의『ZONED』	수를 존 십진수로 표시하여 리턴합니다.

표 26. 자료 링크 스칼라 함수

221 페이지의『DLCOMMENT』	DataLink 값으로부터 주석 값을 리턴합니다.
222 페이지의『DLLINKTYPE』	DataLink 값으로부터 링크 유형 값을 리턴합니다.
223 페이지의『DLURLCOMPLETE』	링크 유형이 URL인 DataLink 값으로부터 완전한 URL 값을 리턴합니다.
224 페이지의『DLURLPATH』	링크 유형이 URL인 DataLink 값으로부터 주어진 서버 내에서 파일에 액세스하는 데 필요한 경로 및 파일명을 리턴합니다. 해당되는 경우 리턴된 값에는 파일 액세스 토큰이 포함됩니다.
225 페이지의『DLURLPATHONLY』	링크 유형이 URL인 DataLink 값으로부터 주어진 서버 내에서 파일에 액세스하는 데 필요한 경로 및 파일명을 리턴합니다. 리턴된 값은 파일 액세스 토큰을 포함하지 않습니다.
226 페이지의『DLURLSCHEME』	링크 유형이 URL인 DataLink 값으로부터 구조를 리턴합니다.
227 페이지의『DLURLSERVER』	링크 유형이 URL인 DataLink 값으로부터 파일 서버를 리턴합니다.
228 페이지의『DLVALUE』	DataLink 값을 리턴합니다.

표 27. 날짜 시간 스칼라 함수

202 페이지의『CURDATE』	날짜 시간 시계(time-of-day clock)를 읽어서 날짜를 리턴합니다.
203 페이지의『CURTIME』	날짜 시간 시계(time-of-day clock)를 읽어서 시간을 리턴합니다.
211 페이지의『DAYS』	값의 요일 부분을 리턴합니다.
207 페이지의『DAYOFMONTH』	값의 요일 부분을 리턴합니다.
208 페이지의『DAYOFWEEK』	한 주간의 요일을 나타내는 정수를 리턴하며 1은 일요일, 7은 토요일을 나타냅니다.
209 페이지의『DAYOFWEEK_ISO』	한 주간의 요일을 나타내는 정수를 리턴하며 1은 월요일, 7은 일요일을 나타냅니다.
210 페이지의『DAYOFYEAR』	1년 중 요일을 나타내는 정수를 리턴합니다.
211 페이지의『DAYS』	날짜를 정수로 표시하여 리턴합니다.
240 페이지의『HOUR』	값의 시 부분을 리턴합니다.

표 27. 날짜 시간 스칼라 함수 (계속)

248 페이지의 『JULIAN_DAY』	기원전 4712년 1월 1일부터 인수에 지정된 날짜까지의 날 수를 나타내는 정수 값을 리턴합니다.
264 페이지의 『MICROSECOND』	값의 마이크로초 부분을 리턴합니다.
265 페이지의 『MIDNIGHT_SECONDS』	자정과 지정된 시간 사이의 시간을 초 수로 환산한 정수 값을 리턴합니다.
268 페이지의 『MINUTE』	값의 분 부분을 리턴합니다.
271 페이지의 『MONTH』	값의 월 부분을 리턴합니다.
274 페이지의 『NOW』	날짜 시간 시계(time-of-day clock)를 읽어서 시간소인을 리턴합니다.
281 페이지의 『QUARTER』	해당 날짜가 위치하는 1년 중 사분기를 나타내는 정수를 리턴합니다.
290 페이지의 『SECOND』	값의 초 부분을 리턴합니다.
308 페이지의 『TIMESTAMPDIFF』	두 개의 시간 소인간의 차이에 따라 예상 간격 숫자 값을 리턴합니다.
327 페이지의 『WEEK』	1년 중 주를 나타내는 정수를 리턴합니다. 주는 일요일부터 시작합니다.
328 페이지의 『WEEK_ISO』	1년 중 주를 나타내는 정수를 리턴합니다. 주는 월요일부터 시작합니다.
330 페이지의 『YEAR』	값의 연도 부분을 리턴합니다.

표 28. 파티션 스칼라 함수

238 페이지의 『HASH』	값 집합의 구획 번호를 리턴합니다.
272 페이지의 『NODENAME』	행이 위치한 관계형 데이터베이스명을 리턴합니다.
273 페이지의 『NODENUMBER』	행의 노드 번호를 리턴합니다.
276 페이지의 『PARTITION』	행의 구획 번호를 리턴합니다.

표 29. 기타 스칼라 함수

197 페이지의 『COALESCE』	널(null)이 아닌 첫 번째 인수를 리턴합니다.
239 페이지의 『HEX』	값을 16진수로 표시하여 리턴합니다.
241 페이지의 『IDENTITY_VAL_LOCAL』	ID 열에 가장 최근에 지정된 값을 리턴합니다.
245 페이지의 『IFNULL』	널(null)이 아닌 첫 번째 인수를 리턴합니다.
253 페이지의 『LENGTH』	값의 길이를 리턴합니다.
262 페이지의 『MAX』	값 집합에서 최대값을 리턴합니다.
266 페이지의 『MIN』	값 집합에서 최소 값을 리턴합니다.
275 페이지의 『NULLIF』	인수가 동일한 경우 널값을 리턴하고 그렇지 않은 경우 첫 번째 인수 값을 리턴합니다.
288 페이지의 『RRN』	행의 상대 레코드 번호를 리턴합니다.
319 페이지의 『VALUE』	널(null)이 아닌 첫 번째 인수를 리턴합니다.

표 30. 숫자 스칼라 함수

175 페이지의 『ABS』	수의 절대값을 리턴합니다.
176 페이지의 『ACOS』	수의 호(arc) 코사인 값을 라디안으로 리턴합니다.

표 30. 숫자 스칼라 함수 (계속)

177 페이지의『ANTILOG』	수의 진수(기수 10)를 리턴합니다.
178 페이지의『ASIN』	수의 호(arc) 사인 값을 라디안으로 리턴합니다.
179 페이지의『ATAN』	수의 호(arc) 탄젠트 값을 라디안으로 리턴합니다.
180 페이지의『ATANH』	수의 쌍곡선 호(arc)의 탄젠트 값을 라디안으로 리턴합니다.
181 페이지의『ATAN2』	x와 y 좌표의 호(arc) 탄젠트 값을 라디안으로 표시된 각도로 리턴합니다.
186 페이지의『CEILING』	숫자로 표시된 것보다 크거나 같은 가장 작은 정수 값을 리턴합니다.
199 페이지의『COS』	수의 코사인 값을 리턴합니다.
200 페이지의『COSH』	수의 쌍곡선 코사인 값을 리턴합니다.
201 페이지의『COT』	수의 코탄젠트 값을 리턴합니다.
218 페이지의『DEGREES』	각도의 정도를 나타내는 숫자 값을 리턴합니다.
220 페이지의『DIGITS』	수의 절대값을 문자 스트링으로 표시하여 리턴합니다.
232 페이지의『EXP』	인수에 의해 지정된 거듭제곱으로 올림된 자연 로그의 기수 값을 리턴합니다.
234 페이지의『FLOOR』	숫자 표현식보다 작거나 같은 가장 큰 정수 값을 리턴합니다.
255 페이지의『LN』	수의 자연 로그 값을 리턴합니다.
258 페이지의『LOG10』	수의 상용 로그(기수 10) 값을 리턴합니다.
269 페이지의『MOD』	첫 번째 인수를 두 번째 인수로 나누어 그 나머지를 리턴합니다.
277 페이지의『PI』	PI 값을 리턴합니다.
280 페이지의『POWER』	첫 번째 인수에 대한 두 번째 인수의 제곱근의 결과를 리턴합니다.
282 페이지의『RADIANS』	각도로 표시되는 인수에 대한 라디안 값을 리턴합니다.
283 페이지의『RAND』	난수를 리턴합니다.
285 페이지의『ROUND』	지정된 소수 자리수로 반올림된 값을 리턴합니다.
291 페이지의『SIGN』	표현식 부호의 인디케이터를 리턴합니다.
292 페이지의『SIN』	수의 사인(sine) 값을 리턴합니다.
293 페이지의『SINH』	수의 쌍곡선 사인(sine) 값을 리턴합니다.
298 페이지의『SQRT』	수의 제곱근을 리턴합니다.
303 페이지의『TAN』	수의 탄젠트 값을 리턴합니다.
304 페이지의『TANH』	수의 쌍곡선 탄젠트 값을 리턴합니다.
315 페이지의『TRUNCATE 또는 TRUNC』	지정된 소수 자리수에서 절단된 숫자 값을 리턴합니다.

표 31. 문자열 스칼라 함수

192 페이지의『CHARACTER_LENGTH』	스트링으로 표시된 것의 길이를 리턴합니다.
198 페이지의『CONCAT』	두 스트링을 연결합니다.
250 페이지의『LCASE』	모든 문자가 소문자로 변환된 스트링을 리턴합니다.
219 페이지의『DIFFERENCE』	두 스트링의 발음 간의 차이를 나타내는 값을 리턴합니다.
249 페이지의『LAND』	인수 스트링의 논리 'AND'인 스트링을 리턴합니다.
250 페이지의『LCASE』	모든 문자가 소문자로 변환된 스트링을 리턴합니다.
251 페이지의『LEFT』	스트링에서 가장 왼쪽의 문자를 리턴합니다.
256 페이지의『LNOT』	인수 스트링의 논리 'NOT'인 스트링을 리턴합니다.

표 31. 문자열 스칼라 함수 (계속)

257 페이지의 『LOCATE』	한 스트링 안에 포함되어 있는 다른 스트링의 시작 위치를 리턴합니다.
259 페이지의 『LOR』	인수 스트링의 논리 'OR'인 스트링을 리턴합니다.
260 페이지의 『LOWER』	모든 문자가 소문자로 변환된 스트링을 리턴합니다.
261 페이지의 『LTRIM』	스트링 표현식의 앞에서 공백이나 16진수 0을 제거합니다.
278 페이지의 『POSITION 또는 POSSTR』	한 스트링 안에 포함되어 있는 다른 스트링의 시작 위치를 리턴합니다.
299 페이지의 『STRIP』	스트링 표현식의 끝 또는 처음에서 공백이나 다른 지정된 문자를 제거합니다.
289 페이지의 『RTRIM』	스트링 표현식 끝에서 공백이나 16진 0을 제거합니다.
296 페이지의 『SOUNDEX』	인수의 단어의 발음을 나타내는 문자 코드를 리턴합니다.
297 페이지의 『SPACE』	인수가 지정한 공백의 수로 구성된 문자 스트링을 리턴합니다.
300 페이지의 『SUBSTRING 또는 SUBSTR』	스트링의 서브스트링을 리턴합니다.
310 페이지의 『TRANSLATE』	스트링의 하나 이상의 문자를 변환합니다.
313 페이지의 『TRIM』	스트링 표현식의 끝 또는 처음에서 공백이나 다른 지정된 문자를 제거합니다.
317 페이지의 『UCASE』	모든 문자가 대문자로 변환된 스트링을 리턴합니다.
317 페이지의 『UCASE』	모든 문자가 대문자로 변환된 스트링을 리턴합니다.
318 페이지의 『UPPER』	모든 문자가 대문자로 변환된 스트링을 리턴합니다.
329 페이지의 『XOR』	인수 스트링의 논리 'XOR'인 스트링을 리턴합니다.

열 함수

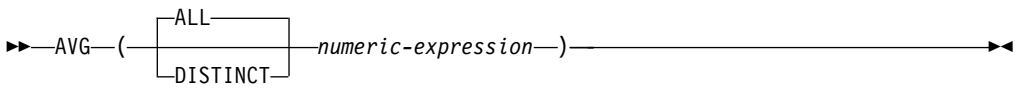
다음 정보는 COUNT(*) 및 COUNT_BIG(*) 이외의 모든 열 함수에 적용합니다.

- 열 함수의 인수는 *expression*에서 나온 값의 집합입니다. *expression*은 열을 포함할 수 있지만 다른 열 함수를 포함할 수는 없습니다. 집합의 범위는 제 4 장 ″조회″에서 설명된 그룹 또는 중간 결과표입니다.
- GROUP BY절이 조회에서 지정되고 FROM, WHERE, GROUP BY 및 HAVING 절의 중간 결과가 공백(empty) 집합이면 열 함수는 적용되지 않고, 조회 결과는 공백 집합이고, SQLCODE는 +100으로, SQLSTATE는 ‘02000’으로 설정됩니다.
- GROUP BY절이 조회에서 지정되지 않고 FROM, WHERE 및 HAVING절의 중간 결과과 공백 집합이면 열 함수는 공백 집합에 적용됩니다.
- 예를 들어, 다음 SELECT문의 결과는 부서 D01의 사원 JOB의 고유한 값입니다.

```
SELECT COUNT(DISTINCT JOB)
     FROM EMPLOYEE
   WHERE WORKDEPT = 'D01'
```

- 키워드 DISTINCT가 함수의 인수를 고려하지 않았지만, 함수 전에 수행된 연산의 스펙이 적용됩니다. DISTINCT가 지정되는 경우 중복 값이 제거됩니다. ALL이 내재적으로 또는 명시적으로 지정된 경우 중복 값은 제거되지 않습니다.
- 열 함수는 WHERE 절이 HAVING 절의 부속 조회이고 표현식의 지정된 열 이름이 그룹에 대한 상관 참조이면 사용될 수 없습니다. 표현식에 둘 이상의 열 이름이 포함되어 있는 경우, 각 열 이름은 동일한 그룹을 상관 참조해야 합니다.

AVG



AVG 함수는 수 집합의 평균을 리턴합니다.

인수 값은 내장 숫자 자료 유형이어야 하고 인수 값의 합은 결과 자료 유형의 범위내에 있어야 합니다.

결과 자료 유형은 다음을 제외하고 인수 값의 자료 유형과 동일합니다.

- 결과는 인수 값이 단정밀도 부동 소수점인 경우 배정밀도 부동 소수점입니다.
- 결과는 인수 값이 작은 정수인 경우 큰 정수입니다.
- 인수 값이 십진 또는 정밀도 *p* 및 스케일 *s*인 0이 아닌 스케일 2진인 경우 결과는 정밀도를 가진 십진수 31이며 스케일은 31-*p*+*s*입니다.

함수는 널값을 제거하여 인수 값에서 구한 값 집합에 적용됩니다. DISTINCT가 사용되는 경우 중복 값이 제거됩니다.

결과는 널(null)이 될 수 있습니다. 값의 집합이 널인 경우 결과는 널값입니다. 그렇지 않으면 결과는 집합의 평균 값입니다.

결합된 값의 순서가 정의되지 않았지만, 모든 중간 결과는 결과 자료 유형의 범위 내에 있어야 합니다.

결과 유형이 정수인 경우 평균 값의 분수 부분은 유실됩니다.

예

- PROJECT 표를 사용하여 호스트 변수 AVERAGE(DECIMAL(5,2))를 부서(DEPTNO) ‘D11’에 프로젝트의 PRSTAFF(평균 사원 레벨)로 설정합니다.

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

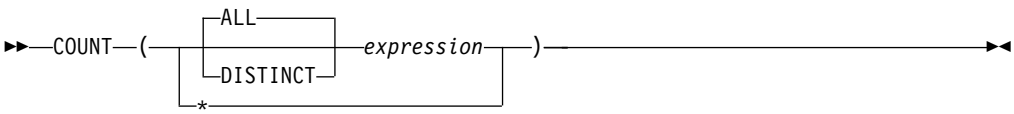
  AVERAGE가 4.25(즉, 17/4)로 설정됩니다.

- PROJECT 표를 사용하여 호스트 변수 ANY_CALC에 부서(DEPTNO) ‘D11’의 프로젝트 각각의 고유 사원 값(PRSTAFF)으로 설정합니다.

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

ANY_CALC는 4.66(즉, 14/3)으로 설정됩니다.

COUNT



COUNT 함수는 행 또는 값 집합의 행 또는 값의 수를 리턴합니다.

함수의 결과는 큰 정수이고, 큰 정수의 범위 내에 있어야 합니다. 결과는 널이 될 수 없습니다. 표가 분배된 표이면 결과는 DECIMAL(15,0)입니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

COUNT(*)의 인수는 행 집합입니다. 결과는 집합의 행 수입니다. 널값만을 포함하는 행은 계수에 포함됩니다.

COUNT(*expression*)의 인수는 값 집합입니다. 함수는 널값을 제거하여 인수 값에서 나온 집합에 적용됩니다. 결과는 집합의 값 수입니다.

COUNT(DISTINCT *expression*)의 인수는 값 집합입니다. 인수 값은 2000보다 큰 길이 속성을 갖는 문자 스트링, 1000 DBCS 또는 UCS-2 문자, LOB 또는 DataLinks 보다 큰 길이 속성을 갖는 그래픽 스트링을 제외한 모든 값이 될 수 있습니다. 함수는 널값과 중복 값을 제거하여 인수 값에서 나온 값 집합에 적용됩니다. 결과는 집합의 값 수입니다.

*HEX 이외의 정렬 순서가 COUNT(DISTINCT *expression*)를 포함하는 명령문이 실행되고 인수에 SBCS, UCS-2 또는 혼합 자료가 들어 있을 때 유효하면 결과는 집합에 있는 각 값에 대한 가중치 값을 비교하여 나옵니다. 가중치는 정렬 순서에 따라 달라집니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 FEMALE(INTEGER)를 SEX 열 값이 'F'인 행 수로 설정합니다.

```
SELECT COUNT(*)
  INTO :FEMALE
  FROM EMPLOYEE
  WHERE SEX = 'F'
```

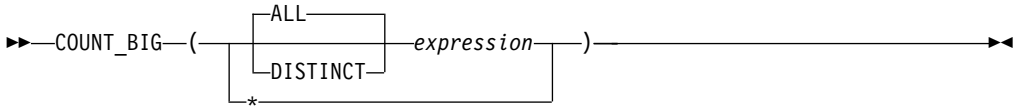
  FEMALE은 13으로 설정됩니다.

- EMPLOYEE 표를 사용하여 호스트 변수 FEMALE_IN_DEPT(INTEGER)를 멤버로 적어도 한 명의 여자가 있는 부서 수(WORKDEPT)로 설정합니다.

```
SELECT COUNT(DISTINCT WORKDEPT)
  INTO :FEMALE_IN_DEPT
  FROM EMPLOYEE
  WHERE SEX='F'
```

FEMALE_IN_DEPT는 5로 설정 됩니다. (부서 A00, C01, D11, D21 및 E11에 적어도 한 명의 여자가 있습니다)

COUNT_BIG



COUNT_BIG 함수는 행이나 값의 집합에 있는 행 또는 값의 수를 리턴합니다. 결과가 정수의 최대값보다 클 수 있는 경우를 제외하고는 COUNT와 비슷합니다.

함수의 결과는 정밀도 31과 스케일 0을 갖는 십진수입니다. 결과는 널이 될 수 없습니다.

COUNT_BIG(*)의 인수는 행 집합입니다. 결과는 집합의 행 수입니다. 널값만을 포함하는 행은 계수에 포함됩니다.

COUNT_BIG(*expression*)의 인수는 값 집합입니다. 함수는 널값을 제거하여 인수 값에서 나온 집합에 적용됩니다. 결과는 집합의 값 수입니다.

COUNT_BIG(DISTINCT *expression*)의 인수는 값 집합입니다. 인수 값은 2000보다 큰 길이 속성을 갖는 문자 스트링, 1000 DBCS 또는 UCS-2 문자, LOB 또는 DataLinks 보다 큰 길이 속성을 갖는 그래픽 스트링을 제외한 모든 값이 될 수 있습니다. 함수는 널값과 중복 값을 제거하여 인수 값에서 나온 값 집합에 적용됩니다. 결과는 집합의 값 수입니다.

*HEX 이외의 정렬 순서가 COUNT_BIG(DISTINCT *expression*)을 포함하는 명령문이 실행되고 인수에 SBCS, UCS-2 또는 혼합 자료가 들어 있을 때 유효하면 결과는 집합에 있는 각 값에 대한 가중치 값을 비교하여 나옵니다. 가중치는 정렬 순서에 따라 달라집니다.

예

- COUNT 예를 참조하고 COUNT 발생에 COUNT_BIG을 대체합니다. 결과는 결과의 자료 유형을 제외하고 동일합니다.
- 특정 열을 계수하려면 소스가 되는 함수가 열의 유형을 지정해야 합니다. 이 예에서 CREATE FUNCTION문은 CHAR로서 정의되는 모든 열을 취하는 소스가 되는 함수를 작성하고, COUNT_BIG을 사용하여 계수를 수행하고 결과를 배정밀도 부동 소수점 숫자로서 리턴합니다. 표시된 조회는 샘플 사원 표에 있는 고유한 부서 수를 계수합니다.

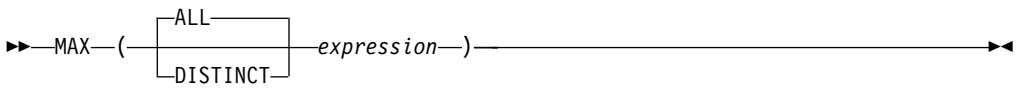
```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE QSYS2.COUNT_BIG(CHAR());
```

```
SET CURRENT PATH RICK, SYSTEM PATH

SELECT COUNT(DISTINCT WORKDEPT FROM EMPLOYEE;
```

새 함수의 매개변수 리스트에 있는 빈 괄호(RICK.COUNT)는 새 함수의 입력 매개변수가 SOURCE절에 지정된 함수의 입력 매개변수와 동일한 유형임을 의미합니다. SOURCE 절의 매개변수 리스트에 있는 빈 괄호(COUNT_BIG)는 DB2가 COUNT_BIG 함수를 찾을 때 COUNT_BIG 함수의 CHAR 매개변수 길이 속성은 무시됨을 의미합니다.

MAX



MAX 열 함수는 그룹의 값 집합에 최대값을 리턴합니다.

인수 값은 LOB 및 DataLink 값을 제외한 모든 내장 자료 유형이 될 수 있습니다.

결과 자료 유형 및 길이 속성은 인수 값의 자료 유형 및 길이 속성과 동일합니다. 인수가 스트링일 때 결과는 인수와 같은 CCSID를 갖습니다. 결과는 널(null)이 될 수 있습니다.

*HEX 이외의 정렬 순서가 MAX 함수가 들어 있는 명령문이 실행될 때 유효하고, 인수에 SBCS, UCS-2 또는 혼합 자료가 있는 경우 결과는 집합의 각 값에 가중치 있는 값을 비교하여 구할 수 있습니다. 가중치는 정렬 순서에 따라 달라집니다.

함수는 널값을 제거하여 인수 값에서 구한 값 집합에 적용됩니다.

함수가 공백 집합에 적용되는 경우 결과는 널값입니다. 그렇지 않으면 결과는 집합의 최대값입니다.

DISTINCT의 스펙은 결과에 영향을 주지 않으며 알려지지 않습니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 MAX_SALARY(DECIMAL(7,2))를 최대 급여(SALARY / 12) 값으로 설정합니다.

```
SELECT MAX(SALARY) /12
  INTO :MAX_SALARY
  FROM EMPLOYEE
```

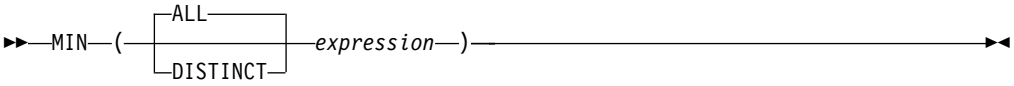
  MAX_SALARY가 4395.83으로 설정됩니다.

- PROJECT 표를 사용하여 호스트 변수 LAST_PROJ(CHAR(24))를 정렬 순서에서 마지막인 프로젝트명(PROJNAME)으로 설정합니다.

```
SELECT MAX(PROJNAME)
  INTO :LAST_PROJ
  FROM PROJECT
```

  LAST_PROJ가 'WELD LINE PLANNING '으로 설정됩니다.

MIN



MIN 열 함수는 그룹의 값 집합에 최소 값을 리턴합니다.

인수 값은 LOB 및 DataLink 값을 제외한 모든 내장 자료 유형이 될 수 있습니다.

결과 자료 유형 및 길이 속성은 인수 값의 자료 유형 및 길이 속성과 동일합니다. 인수가 스트링일 때 결과는 인수와 같은 CCSID를 갖습니다. 결과는 널(null)이 될 수 있습니다.

*HEX 이외의 정렬 순서가 MIN 함수가 들어 있는 명령문이 실행될 때 유효하고, 인수에 SBCS, UCS-2 또는 혼합 자료가 있는 경우 결과는 집합의 각 값에 가중치 있는 값을 비교하여 구할 수 있습니다.

함수는 널값을 제거하여 인수 값에서 구한 값 집합에 적용됩니다.

함수가 공백 집합에 적용되는 경우 결과는 널값입니다. 그렇지 않으면 결과는 집합의 최소 값입니다.

DISTINCT의 스펙은 결과에 영향을 주지 않으며 알려지지 않습니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 COMM_SPREAD(DECIMAL(7,2))를 부서(WORKDEPT) ‘D11’의 멤버에 대한 최대 및 최소 수당(COMM) 간의 차이 값으로 설정합니다.

```
SELECT MAX(COMM) - MIN(COMM)
  INTO :COMM_SPREAD
  FROM EMPLOYEE
  WHERE WORKDEPT  = 'D11'
```

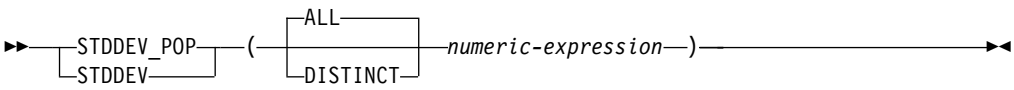
  COMM_SPREAD는 1118(즉, 2580 - 1462)로 설정됩니다.

- PROJECT표를 사용하여 호스트 변수 FIRST_FINISHED(CHAR(10))를 첫 번째 프로젝트 완료가 예정된 예상 최종 날짜(PRENDATE)로 설정합니다.

```
SELECT MIN(PRENDATE)
  INTO :FIRST_FINISHED
  FROM PROJECT
```

  FIRST_FINISHED가 ‘1982-09-15’로 설정됩니다.

STDDEV 또는 STDDEV_POP



STDDEV_POP 함수는 수 집합의 표준 편차(/n)를 리턴합니다. 치우친 표준 편차를 연산하기 위해 사용된 형식은 다음과 같습니다.

```
STDDEV_POP = SQRT(VAR_POP)
```

여기서 SQRT(VAR_POP)는 변수의 제곱근입니다.

인수 값은 내장 숫자 자료 유형이어야 하고 인수 값의 합은 결과 자료 유형의 범위 내에 있어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 결과는 널(null)이 될 수 있습니다.

함수는 널값을 제거하여 인수 값에서 구한 값 집합에 적용됩니다. DISTINCT가 지정되는 경우 중복 값이 제거됩니다.

함수가 공백 집합에 적용되는 경우 결과는 널값입니다. 그렇지 않으면 결과는 집합 값의 표준 편차입니다.

추가된 값의 순서가 정의되지 않았지만, 모든 중간 결과는 결과 자료 유형의 범위 내에 있어야 합니다.

STDDEV를 STDEV_POP와 동의어로 지정할 수 있습니다.

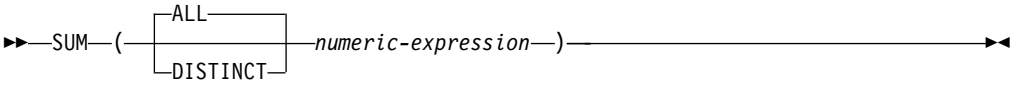
예

- EMPLOYEE 표를 사용하여 호스트 변수 DEV(FLOAT 배정밀도)를 부서 A00에서 각 사원들의 급여의 표준 편차로 설정합니다.

```
SELECT STDDEV_POP(SALARY)
  INTO :DEV
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

  DEV는 대략 9938.00으로 설정됩니다.

SUM



SUM 함수는 수 집합의 합계를 리턴합니다.

인수 값은 내장 숫자 자료 유형이어야 하고 인수 값의 합은 결과 자료 유형의 범위내에 있어야 합니다.

결과 자료 유형은 결과가 다음과 같은 경우를 제외하고 인수 자료 유형과 동일합니다.

- 인수 값이 단정밀도 부동 소수점인 경우 배정밀도 부동 소수점입니다.
- 인수 값이 작은 정수인 경우 큰 정수입니다.
- 인수 값이 0이 아닌 스케일 2진인 경우 십진입니다.

결과는 널(null)이 될 수 있습니다.

인수 값의 자료 유형이 십진 또는 0이 아닌 스케일 2진인 경우 결과의 정밀도는 31이고 스케일은 인수 값의 스케일과 같습니다.

함수는 널값을 제거하여 인수 값에서 구한 값 집합에 적용됩니다. DISTINCT가 지정되는 경우 중복 값이 제거됩니다.

함수가 공백 집합에 적용되는 경우 결과는 널값입니다. 그렇지 않으면 결과는 집합의 값의 합계입니다.

추가된 값의 순서가 정의되지 않았지만, 모든 중간 결과는 결과 자료 유형의 범위 내에 있어야 합니다.

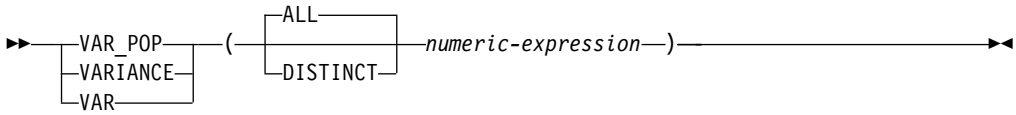
예

- EMPLOYEE 표를 사용하여 호스트 변수 JOB_BONUS(DECIMAL(9,2))를 관리 직원(clerk)(JOB='CLERK')에 지불한 총 보너스 금액(BONUS)으로 설정합니다.

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
  FROM EMPLOYEE
            WHERE JOB = 'CLERK'
```

  JOB_BONUS가 2800으로 설정됩니다.

VAR_POP 또는 VARIANCE 또는 VAR



STDDEV_POP 함수는 수 집합의 편차(/n)를 리턴합니다. 분산을 연산하기 위해 사용된 형식은 다음과 같습니다.

```
VAR_POP = SUM(X**2)/COUNT(X) - (SUM(X)/COUNT(X))**2
```

인수 값은 내장 숫자 자료 유형이어야 하고 인수 값의 합은 결과 자료 유형의 범위 내에 있어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 결과는 널(null)이 될 수 있습니다.

함수는 널값을 제거하여 인수 값에서 구한 값 집합에 적용됩니다. DISTINCT가 지정되는 경우 중복 값이 제거됩니다.

함수가 공백 집합에 적용되는 경우 결과는 널값입니다. 그렇지 않으면 결과는 집합의 값의 편차입니다.

추가된 값의 순서가 정의되지 않았지만, 모든 중간 결과는 결과 자료 유형의 범위 내에 있어야 합니다.

VARIANCE 및 VAR을 VAR_POP과 동의어로 지정할 수 있습니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 DEV(FLOAT 배정밀도)를 부서 A00에서 각 사원들의 급여의 분산으로 설정합니다.

```
SELECT VAR_POP(SALARY)
  INTO :VARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

VARNCE는 대략 98763888.88로 설정됩니다.

스칼라 함수

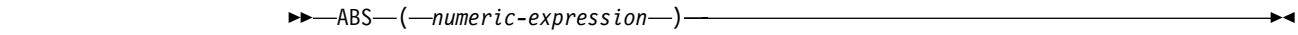
*스칼라 함수*는 표현식이 사용될 수 있는 곳에서 사용될 수 있습니다. 열 함수 사용의 제한사항은 스칼라 함수가 값 집합 보다 단일 매개변수 값에 적용되기 때문에 스칼라 함수를 적용하지 않습니다. 스칼라 함수의 인수는 함수가 될 수 있습니다. 그러나, 표현식과 열 함수 사용에 적용하는 제한사항은 표현식 또는 열 함수가 스칼라 함수 내에 사용될 때도 적용합니다. 예를 들어, 스칼라 함수의 인수는 열 함수가 스칼라 함수가 사용되는 문맥에 허용되는 경우에만 열 함수가 될 수 있습니다.

예

다음 SELECT문의 결과는 부서 D01의 사원이 있는 만큼의 행 수를 갖습니다.

```
 SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BIRTHDATE)
   FROM EMPLOYEE
WHERE WORKDEPT = 'D01'
```

ABS



ABS 함수는 수의 절대 값을 리턴합니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다.

결과 자료 유형 및 길이 속성은 인수 값이 작은 정수인 경우 결과가 큰 정수이고, 인수 값이 단정밀도 부동 소수점인 경우 결과가 배정밀도 부동 소수점인 경우를 제외하고 인수값의 자료유형 및 길이 속성과 동일합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

주

ABSVAL은 ABS와 동의어입니다. 단지 이전 DB2 릴리스와 호환성을 위해 지원됩니다.

예

- 호스트 변수 PROFIT가 값 -50000인 큰 정수라고 가정합니다.

```
SELECT ABS(:PROFIT)
  FROM SYSIBM.SYSDUMMY1
```

  값 50000을 리턴합니다.

ACOS

ACOS 함수는 라디안으로 표현된 각을 인수의 호 코사인(arc cosine)으로 리턴합니다. ACOS 및 COS 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다. 값은 -1 이상 1 이하인 수이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는 -1 이상 PI 이하인 수이어야 합니다.

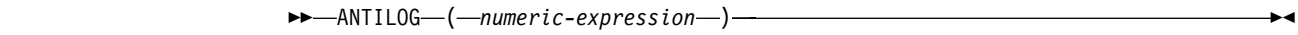
예

- 호스트 변수 ACOSINE을 값 0.070737202인 DECIMAL(10,9) 호스트 변수라고 가정합니다.

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 1.49를 리턴합니다.

ANTILOG



ANTILOG 함수는 수의 진수(기준 10)를 리턴합니다. ANTILOG 및 LOG 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 ALOG를 값 1.499961866인 DECIMAL(10,9) 호스트 변수라고 가정합니다.

```
SELECT ANTILOG(:ALOG)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 31.62를 리턴합니다.

ASIN

ASIN 함수는 라디안으로 표현된 각을 인수의 호 사인(arc sine)으로 리턴합니다. ASIN 및 SIN 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다. 값은 -1 이상 1 이하인 수이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는 Pl/2 이상 PI/2 이하입니다.

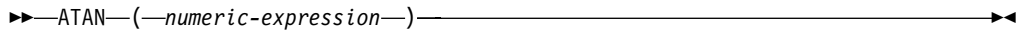
예

- 호스트 변수 ASINE를 값 0.997494987인 DECIMAL(10,9) 호스트 변수라고 가정합니다.

```
SELECT ASIN(:ASINE)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 1.50을 리턴합니다.

ATAN



ATAN 함수는 라디안으로 표현된 각을 인수의 호 탄젠트(arc tangent)로 리턴합니다. ATAN 및 TAN 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는 Pl/2 이상 PI/2 이하입니다.

예

- 호스트 변수 ATANGENT를 값 14.10141995인 DECIMAL(10,8) 호스트 변수라고 가정합니다.

```
SELECT ATAN(:ATANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 1.50을 리턴합니다.

ATANH

ATANH 함수는 수의 쌍곡선의 호 탄젠트(arc tangent)를 라디안으로 리턴합니다. ATANH 및 TANH 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다. 값은 -1 이상 1 이하여야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

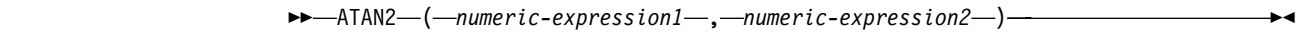
예

- 호스트 변수 HATAN을 값 0.905148254인 DECIMAL(10,9) 호스트 변수라고 가정합니다.

```
SELECT ATANH(:HATAN)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 1.50을 리턴합니다.

ATAN2



ATAN2 함수는 x와 y 좌표의 호(arc) 탄젠트 값을 라디안으로 표시된 각도로 리턴합니다. 첫 번째와 두 번째 인수는 각각 x와 y 좌표를 지정합니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다. 모든 인수는 0이 아니어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 어떤 인수가 널이 될 수 있다면 결과도 널이 될 수 있으며, 어떤 인수가 널이면 결과도 널입니다.

예

- 호스트 변수 HATAN2A 및 HATAN2B가 각각 1과 2 값을 갖는 DOUBLE 호스트 변수라고 가정합니다.

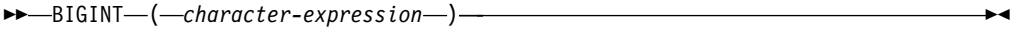
```
SELECT ATAN2(:HATAN2A,:HATAN2B)
  FROM SYSIBM.SYSDUMMY1
```

  대략 1.1071487 값을 갖는 배정밀도 부동 소수점 숫자를 리턴합니다.

BIGINT

숫자 대 큰 정수

문자 대 큰 정수



BIGINT 함수는 다음의 큰 정수 표시를 리턴합니다.

- 수
- 소수의 문자 스트링 표시
- 정수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 큰 정수 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

숫자 대 큰 정수

numeric-expression

내장된 숫자 자료 유형의 숫자 값을 리턴하는 표현식입니다.

인수가 *numeric-expression*인 경우 결과는 인수가 큰 정수 열 또는 변수에 할당된 경우에 발생하는 동일한 숫자입니다. 인수의 전체 부분이 큰 정수 범위 내에 있지 않으면 오류가 발생합니다. 인수의 분수 부분은 절단됩니다.

문자 대 큰 정수

character-expression

표현식은 정수의 문자 스트링 표현을 리턴합니다. 표현식은 CLOB가 아니어야 합니다.

인수가 *character-expression*이면 결과는 CAST(*character-expression* AS BIGINT)로부터 결과된 동일한 숫자입니다. 선행 및 후미 공백은 제거되고 결과 스트링은 부동 소수점, 정수 또는 소수 상수를 형식화하기 위한 규칙을 따릅니다. 인수의 전체 부분이 정수 범위 내에 있지 않으면 오류가 발생합니다. 인수의 분수 부분은 절단됩니다.

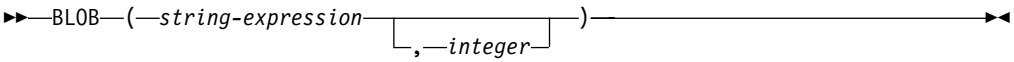
함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- EMPLOYEE 표를 사용하여 어플리케이션에서 추가 처리를 위해 큰 정수 양식으로 EMPNO 열을 선택합니다.

```
SELECT BIGINT(SALARY)
     FROM EMPLOYEE
```

BLOB



BLOB 함수는 스트링 유형의 BLOB 표시로 리턴합니다.

주: 또한 CAST 표현식은 2진 스트링 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수의 결과는 BLOB입니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

string-expression

스트링 표현식의 값은 문자, 그래픽 스트링, 2진 스트링 또는 행 ID가 될 수 있습니다.

integer

결과로 나온 2진 스트링에 대한 길이 속성을 지정합니다. 값은 1 - 2 147 483 647 사이의 값이어야 합니다.

만일 *정수*가 지정되지 않았다면:

- *string-expression*이 빈 스트링 상수이면 결과의 길이 속성은 1입니다.
- 그렇지 않으면 결과의 길이 속성은 인수가 그래픽 스트링이 아닌 한, 첫 번째 인수의 길이 속성과 같습니다. 이 경우 결과의 길이 속성은 인수의 길이 속성의 두 배입니다.

결과의 실제 길이는 결과의 길이 속성과 표현식의 실제 길이의 최소(또는 입력이 그래픽 자료일 때 표현식 길이의 두 배)입니다. *string-expression*의 길이가 결과 길이 속성보다 큰 경우 절단이 수행됩니다. 첫 번째 입력 인수가 문자 스트링이고 잘린 모든 문자가 공백이 아닌 경우, 또는 첫 번째 입력 인수가 그래픽 스트링이고 잘린 모든 문자가 2바이트 공백이 아닌 경우, 경고(SQLSTATE 01004)가 리턴됩니다.

예

- 다음 함수는 스트링 'This is a BLOB'에 대해 BLOB를 리턴합니다.

```
SELECT BLOB('This is a BLOB')
  FROM SYSIBM.SYSDUMMY1
```

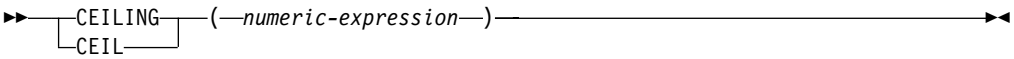
- 다음 함수는 로케이터 myclob_locator에 의해 식별되는 큰 오브젝트에 대해 BLOB를 리턴합니다.

```
SELECT BLOB(:myclob_locator)
  FROM SYSIBM.SYSDUMMY1
```

- 표에 TOPOGRAPHIC_MAP라는 BLOB 열과 MAP_NAME이라는 VARCHAR이 있다고 가정합니다. 스트링 'Pellow Island'가 포함된 맵을 찾아 연결된 맵 이름이 있는 하나의 2진 스트링을 실제 맵 앞으로 리턴합니다. 다음 함수는 로케이터 myclob_locator에 의해 식별되는 큰 오브젝트에 대해 BLOB를 리턴합니다.

```
SELECT BLOB( MAP_NAME CONCAT  ':  ' CONCAT  TOPOGRAPHIC_MAP )
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE '%Pellow Island%'
```

CEILING



CEIL 또는 CEILING 함수는 *numeric-expression*보다 크거나 같은 가장 작은 정수 값을 리턴합니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다.

함수의 결과는 만일 인수가 DECIMAL 또는 NUMERIC일 때 스케일이 0인 경우를 제외하고 인수와 동일한 자료 유형 및 길이 속성을 갖습니다. 예를 들어, DECIMAL(5,5)의 자료 유형을 갖는 인수는 DECIMAL(5,0)의 결과를 가져옵니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

예

- 모든 사원에 있어서 가장 높은 급여를 찾습니다. 결과를 다음 정수로 반올림합니다. SALARY 열은 십진 자료 유형을 갖습니다.

  ```
  SELECT CEIL(MAX(SALARY)/12
       FROM EMPLOYEE
  ```

  가장 높은 급여의 사원이 년간 $52750.00을 받는 Christine Haas이므로 이 예의 결과는 4396.00입니다. 그녀의 평균 급여는 CEIL 함수를 적용하기 전에 4395.83입니다.

- 양수 및 음수 모두에 CEILING을 사용합니다.

  ```
  SELECT CEILING( 3.5),
         CEILING( 3.1),
         CEILING(-3.1),
         CEILING(-3.5),
    FROM SYSIBM.SYSDUMMY1
  ```

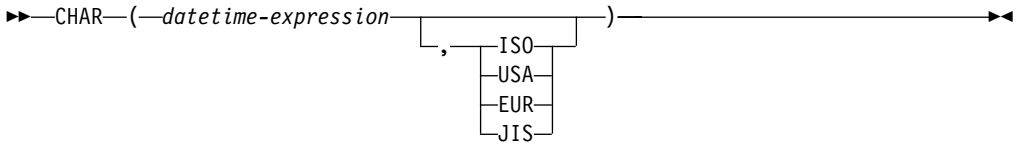
  리턴되는 값은 다음과 같습니다.

  ```
  04.   04.  -03.  -03.
  ```

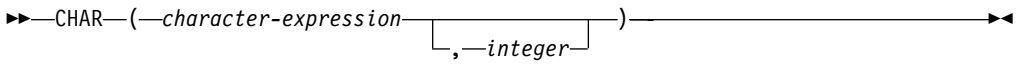
  의 결과를 갖습니다.

CHAR

날짜 시간 대 문자

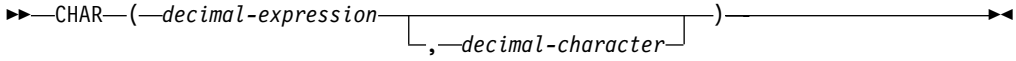


문자 대 문자

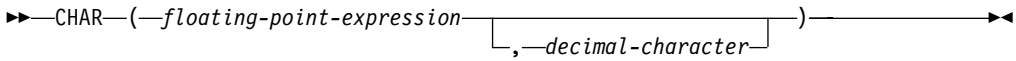


정수 대 문자

소수 대 문자



부동 소수점 대 문자



CHAR 함수는 다음과 같은 고정 길이 문자 스트링 표시를 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우 정수
- 첫 번째 인수가 십진수인 경우 십진수
- 첫 번째 인수가 DOUBLE 또는 REAL인 경우 배정밀도 부동 소수점 수
- 첫 번째 인수가 문자 스트링 유형인 경우 문자 스트링
- 첫번 인수가 DATE인 경우 날짜 값
- 첫 번째 인수가 TIME인 경우 시간 값
- 첫 번째 인수가 TIMESTAMP인 경우 시간소인 값
- 첫 번째 인수가 ROWID이 경우 행 ID 값.

주: 또한 CAST 표현식은 고정 길이 문자 스트링 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

첫 번째 인수는 BLOB, GRAPHIC, VARGRAPHIC 또는 DBCLOB가 아닌 내장 자료 유형이어야 합니다.

함수의 결과는 고정 길이 문자 스트링입니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

날짜 시간 대 문자

datetime-expression
다음 세가지 내장 자료 유형 중에 하나인 표현식

날짜 결과는 두 번째 인수로 지정된 형식에서 날짜의 문자 스트링 표시입니다. 두 번째 인수가 저정되지 않는 경우 형식은 디폴트 날짜 형식을 사용합니다. 형식이 ISO, USA, EUR 또는 JIS인 경우 결과의 길이는 10입니다. 반면 결과의 길이가 디폴트 날짜 형식의 길이입니다. 자세한 내용은 70 페이지의 『Datetime 값의 스트링 표시』를 참조하십시오.

시간 결과는 두 번째 인수로 지정된 형식에서 시간의 문자 스트링 표시입니다. 두 번째 인수가 저정되지 않는 경우 형식은 디폴트 시간 형식을 사용합니다. 결과의 길이는 8입니다.

timestamp
두 번째 인수는 사용될 수 없고 지정되지 않아야 합니다.

결과는 시간소인의 문자 스트링 표시입니다. 결과의 길이는 26입니다.

CCSID의 스트링은 현재 서버에서 디폴트 SBCS CCSID입니다.

ISO, EUR, USA 또는 JIS
결과 문자열의 날짜 또는 시간 형식을 지정합니다. 자세한 내용은 70 페이지의 『Datetime 값의 스트링 표시』를 참조하십시오.

문자 대 문자

character-expression
내장 문자 스트링 값을 리턴하는 표현식.

integer
결과로 나오는 고정 길이 문자 스트링에 길이 속성을 지정합니다. 값은 1 - 32766 (널 가능한 경우 32765) 사이의 값이어야 합니다. 첫 번째 인수가 혼합 자료인 경우 두 번째 인수는 4보다 작을 수 없습니다.

두 번째 인수가 지정되지 않은 경우는 다음과 같습니다.

- *character-expression*이 빈 스트링 상수이면 결과의 길이 속성은 1입니다.
- 그렇지 않으면 결과의 길이 속성은 인수가 첫 번째 인수의 길이 속성과 같습니다.

실제 길이는 결과의 길이 속성과 같습니다. *문자 표현*의 길이가 결과 길이보다 작은 경우 결과 길이까지 공백으로 채워집니다. *문자 표현*의 길이가 결과 길이보다 큰 경우 절단됩니다. 절단된 문자가 모두 공백이 아니면 경고(SQLSTATE 01004)가 리턴됩니다.

CCSID의 스트링은 CCSID의 *character-expression*입니다.

정수 대 문자

integer-expression

정수 자료 유형(SMALLINT, INTEGER 또는 BIGINT 중 하나)인 값을 리턴하는 표현식

결과는 SQL 정수 상수의 형식에서 인수의 고정 길이 문자 스트링 표시입니다. 결과는 인수가 음인 경우 선행 음의 부호를 갖는 인수 값을 표시하는 유효 숫자인 n자로 구성됩니다. 결과는 왼쪽 정렬입니다.

- 인수가 작은 정수인 경우

  결과 길이는 6입니다. 결과의 문자 수가 6 보다 작은 경우 결과의 오른쪽이 공백으로 채워집니다.

- 인수가 큰 정수인 경우

  결과의 길이는 11입니다. 결과의 문자 수가 11 보다 작은 경우 결과의 오른쪽이 공백으로 채워집니다.

- 인수가 큰 정수인 경우

  결과의 길이는 20입니다. 결과에 있는 문자 수가 20보다 작으면, 결과는 오른쪽에서 공백으로 채워집니다.

CCSID의 스트링은 현재 서버에서 디폴트 SBCS CCSID입니다.

소수 대 문자

decimal-expression

내장 십진 자료 유형(DECIMAL 또는 NUMERIC)인 값을 리턴하는 표현식. 다른 정밀도와 스케일이 요구되는 경우 DECIMAL 스칼라 함수는 변경하는 데 사용될 수 있습니다.

decimal-character

결과 문자 스트링에 소수 자리를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 소수점을 사용합니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 인수의 고정 길이 문자 스트링 표시입니다. 결과는 소수 문자를 포함하고 *p* 자릿수까지이며, 여기서 *p*는 인수가 음인 경우 선행 음의 부호를 갖는 *decimal-expression*의 정밀도입니다. 선행 0은 리턴되지 않습니다. 후미 0은 리턴됩니다.

결과 길이는 2+ *p*이며, 여기서 *p*는 *decimal-expression*의 정밀도입니다. 양의 값은 항상 하나의 후미 공백을 포함함을 의미합니다.

CCSID의 스트링은 현재 서버에서 디폴트 SBCS CCSID입니다.

부동 소수점 대 문자

floating-point expression
내장 부동 소수점 값을 리턴하는 표현식.

decimal-character
결과 문자 스트링에 소수 자리를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 소수점을 사용합니다. 자세한 내용은 102 페이지의『소수점』을 참조하십시오.

결과는 부동 소수점 상수의 형식에서 인수의 고정 길이 문자 스트링 표시입니다. 결과의 길이는 24입니다. 인수가 음인 경우 결과의 첫 번째 문자는 음의 부호입니다. 그렇지 않으면 첫 번째 문자는 숫자입니다. 인수가 0인 경우 결과는 0E0입니다. 그렇지 않으면 결과는 0이 아닌 한 자리 숫자 다음에 마침표와 일련의 숫자가 오는 가수와 같은 인수의 값을 표시하는 데 사용될 수 있는 최소의 문자를 포함합니다.

결과에서 문자 수가 24보다 작은 경우 결과의 오른쪽이 공백으로 채워집니다.

CCSID의 스트링은 현재 서버에서 디폴트 SBCS CCSID입니다.

예

- 열 PRSTDATE가 1988-12-25와 같은 내부 값을 갖는다고 가정합니다. 날짜 형식은 *MDY이고 날짜 분리자는 슬래시(/)입니다.

```
SELECT CHAR(PRSTDATE, USA)
     FROM PROJECT
```

  값은 '12/25/1988'이 됩니다.

```
SELECT CHAR(PRSTDATE)
     FROM PROJECT
```

  값은 '12/25/88'이 됩니다.

- STARTING열에 17.12.30의 내부 값이 있고, 호스트 변수 HOUR_DUR (DECIMAL(6,0))에 050000(즉 5시간)의 소요 시간 값이 들어있다고 가정합니다.

```
SELECT CHAR(STARTING, USA)
  FROM CL_SCHED
```

값 '5:12 PM'이 됩니다.

```
SELECT CHAR(STARTING + :HOUR_DUR, JIS)
  FROM CL_SCHED
```

결과는 '10:12:00'이 됩니다.

- 열 RECEIVED (시간 소인)에 PRSTDATE 및 STARTING 열의 조합과 같은 내부 값이 있다고 가정합니다.

```
SELECT CHAR(RECEIVED)
  FROM IN_TRAY
```

값은 '1988-12-25-17.12.30.000000'이 됩니다.

- CHAR 함수를 사용하여 유형 고정 길이 문자를 만들어 EMPLOYEE 표의 (VARCHAR(15)로 정의된) LASTNAME 열에 표시된 결과 길이를 10자로 줄입니다.

```
SELECT CHAR(LASTNAME,10)
     FROM EMPLOYEE
```

(후미 공백을 제외하고) 10자보다 큰 길이의 LASTNAME을 갖는 행의 경우 값이 절단됨을 알리는 경고(SQLSTATE 01004)가 리턴됩니다.

- CHAR 함수를 사용하여 고정 길이 스트링으로(SMALLINT로 정의된) EDLEVEL에 대한 값을 리턴합니다.

```
SELECT CHAR(EDLEVEL)
     FROM EMPLOYEE
```

EDLEVEL 18은 CHAR(6) 값 '18␣␣␣␣'(18 다음에 4개의 공백)로 리턴됩니다.

- STAFF 표에 정밀도 9, 스케일 2인 소수로 정의된 SALARY 열이 있다고 가정합니다. 현재 값은 18357.50이고 소수 문자(18357,50)로서 쉼표로 리턴됩니다.

```
SELECT CHAR(SALARY, ',')
     FROM EMPLOYEE
```

'18357,50␣␣␣' 값을 리턴합니다(18357,50 다음에 3개의 공백이 옴).

- 20000.25에서 뺀 열과 같은 SALARY 열이 디폴트 소수 문자에 대해 리턴되고 디폴트가 마침표라고 가정합니다.

```
SELECT CHAR(20000.25 - SALARY)
     FROM EMPLOYEE
```

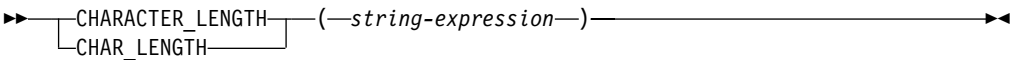
'-1642.75␣␣␣' 값을 리턴합니다(-1642.75 다음에 3개의 공백이 옴).

- 호스트 변수 DOUBLE_NUM에 배정밀도 부동 소수점 자료 유형과 값 -987.654321E-35값이 있다고 가정합니다.

```
SELECT CHAR(:DOUBLE_NUM)
  FROM SYSIBM.SYSDUMMY1
```

문자 값 '-9.8765432100000002E-33'이 됩니다.

CHARACTER_LENGTH



CHARACTER_LENGTH 또는 CHAR_LENGTH 함수는 스트링 표현식의 길이를 리턴합니다. 유사한 함수에 대하여 253 페이지의 『LENGTH』를 참조하십시오.

인수는 모든 내장 스트링 자료 유형의 값을 리턴하는 표현식입니다.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는(바이트 수가 아니라) 인수의 문자 수입니다. 단일 문자는 SBCS 또는 DBCS 문자입니다. 스트링 길이는 후미 공백을 포함합니다. 가변 길이 스트링의 길이는 최대 길이가 아닌 실제 길이입니다.

예

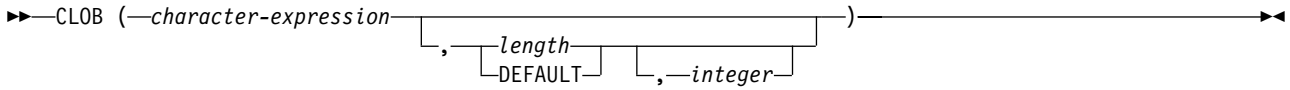
- 호스트 변수 ADDRESS가 값 '895 Don Mills Road'를 갖는 길이 문자 스트링이라고 가정합니다.

```
SELECT CHARACTER_LENGTH(:ADDRESS)
  FROM SYSIBM.SYSDUMMY1
```

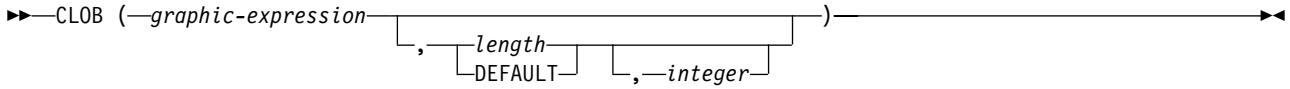
  값 18을 리턴합니다.

CLOB

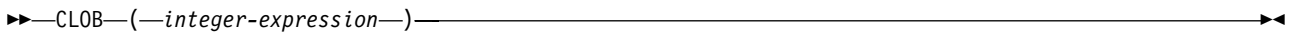
문자 대 CLOB



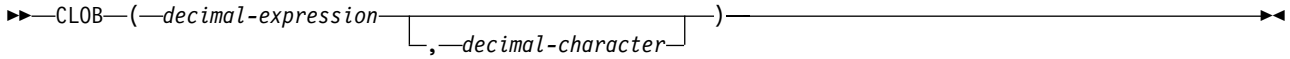
그래픽 대 CLOB



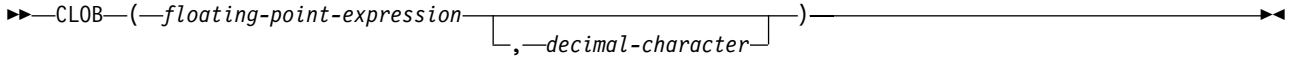
정수 대 CLOB



소수 대 CLOB



부동 소수점 대 CLOB



CLOB 함수는 다음과 같은 문자 스트링 표시를 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우 정수
- 첫 번째 인수가 팩 또는 존 십진수인 경우 십진수
- 첫 번째 인수가 DOUBLE 또는 REAL인 경우 배정밀도 부동 소수점 수
- 첫 번째 인수가 문자 스트링 유형인 경우 문자 스트링
- 첫 번째 인수가 UCS-2 그래픽 스트링인 경우 그래픽 스트링

주: 또한 CAST 표현식은 큰 문자 오브젝트 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수 결과는 CLOB 스트링입니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

문자 대 CLOB

character-expression
내장 문자 스트링 값을 리턴하는 표현식.

CLOB

length

결과 변수 길이 문자 스트링에 길이 속성을 지정합니다. 값은 1 - 2 147 483 647 사이의 값이어야 합니다. 첫 번째 인수가 혼합 자료인 경우 두 번째 인수는 4보다 작을 수 없습니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정되는 경우

- *character-expression*이 빈 스트링 상수이면 결과의 길이 속성은 1입니다.
- 그렇지 않으면 결과의 길이 속성은 인수가 첫 번째 인수의 길이 속성과 같습니다.

결과의 실제 길이는 결과의 길이 속성과 *character-expression*의 실제 길이의 최소값입니다. *문자* 표현의 길이가 결과 길이보다 큰 경우 절단됩니다. 절단된 문자가 모두 공백이 아니면 경고(SQLSTATE 01004)가 리턴됩니다.

integer

결과의 CCSID를 지정합니다. 유효한 SBCS CCSID 또는 혼합 자료 CCSID이어야 합니다. 세 번째 인수가 SBCS CCSID인 경우 결과는 SBCS 자료입니다. 세 번째 인수가 혼합 CCSID인 경우 결과는 혼합 자료입니다. 세 번째 인수가 SBCS CCSID인 경우 첫 번째 인수는 DBCS 선택 또는 DBCS 전용 스트링일 수 없습니다. 세 번째 인수는 65535가 될 수 없습니다.

세 번째 인수가 지정되지 않은 경우 첫 번째 인수는 CCSID 65535를 가져서는 안됩니다.

- 첫 번째 인수가 비트 오류인 경우 오류가 발생합니다.
- 첫 번째 인수가 SBCS 자료인 경우 결과는 SBCS 자료입니다. 결과 CCSID는 첫 번째 인수의 CCSID와 같습니다.
- 첫 번째 인수가 혼합 자료(DBCS 개방, DBCS 전용 또는 DBCS 선택)인 경우 결과는 혼합 자료입니다. 결과 CCSID는 첫 번째 인수의 CCSID와 같습니다.

그래픽 대 CLOB

graphic-expression

내장 그래픽 스트링 값을 리턴하는 표현식. DBCS 그래픽 자료이어서는 안됩니다.

length

결과 변수 길이 문자 스트링에 길이 속성을 지정합니다. 값은 1 - 2 147 483 647 사이의 값이어야 합니다. 결과가 혼합 자료이면 두 번째 인수는 4보다 작을 수 없습니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정되는 경우 결과의 길이 속성은 다음과 같이 판별됩니다(여기서 *n*은 첫 번째 인수의 길이 속성입니다).

- *graphic-expression*이 빈 그래픽 스트링 상수인 경우 결과의 길이 속성은 1입니다.

- 결과가 SBCS 자료인 경우 결과 길이는 *n*입니다.
- 결과가 혼합 자료인 경우 결과 길이는 (2.5*(*n*-1)) + 4입니다.

결과의 실제 길이는 결과의 길이 속성과 *graphic-expression*의 실제 길이의 최소값입니다. *그래픽 표현식*의 길이가 결과 길이 속성보다 큰 경우 절단됩니다. 절단된 문자가 모두 공백이 아니면 경고(SQLSTATE 01004)가 리턴됩니다.

integer

결과의 CCSID를 지정합니다. 유효한 SBCS CCSID 또는 혼합 자료 CCSID이어야 합니다. 세 번째 인수가 SBCS CCSID인 경우 결과는 SBCS 자료입니다. 세 번째 인수가 혼합 CCSID인 경우 결과는 혼합 자료입니다. 세 번째 인수는 65535가 될 수 없습니다.

세 번째 인수가 지정되지 않은 경우 결과의 CCSID는 현재 서버에서 디폴트 CCSID입니다. 디폴트 CCSID가 혼합 자료인 경우 결과는 혼합 자료입니다. 디폴트 CCSID가 SBCS 자료인 경우 결과는 SBCS 자료입니다.

정수 대 CLOB

integer-expression

내장 정수 자료 유형(SMALLINT, INTEGER 또는 BIGINT 중 하나)인 값을 리턴하는 표현식

결과는 SQL 정수 상수의 양식에서 인수의 가변 길이 문자 스트링 표시입니다. 결과는 인수가 음인 경우 선행 음의 부호를 갖는 인수 값을 표시하는 유효 숫자인 n자로 구성됩니다. 결과는 왼쪽 정렬입니다.

- 인수가 작은 정수인 경우 결과의 길이 속성은 6입니다.
- 인수가 큰 정수인 경우 결과의 길이 속성은 11입니다.
- 인수가 큰 정수인 경우 결과의 길이 속성은 20입니다.

결과의 실제 길이는 인수의 값을 표시하는 데 사용될 수 있는 최소 문자입니다. 선행 0은 포함되지 않습니다. 인수가 음인 경우 결과의 첫 번째 문자는 음의 부호입니다. 그렇지 않으면 첫 번째 문자는 숫자입니다.

결과의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

소수 대 CLOB

decimal-expression

내장 십진 자료 유형(DECIMAL 또는 NUMERIC)인 값을 리턴하는 표현식. 다른 정밀도와 스케일이 요구되는 경우 DECIMAL 스칼라 함수는 변경하는 데 사용될 수 있습니다.

decimal-character

결과 문자 스트링에 소수 자리를 분리하는 데 사용되는 1바이트 문자 상수를 지정

합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 소수점을 사용합니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 인수의 가변 길이 문자 스트링 표시입니다. 결과는 소수 문자를 포함하고 *p* 자릿수까지이며, 여기서 *p*는 인수가 음인 경우 선행 음의 부호를 갖는 *decimal-expression*의 정밀도입니다. 선행 0은 리턴되지 않습니다. 후미 0은 리턴됩니다.

결과의 길이 속성은 2+*p*이며, 여기서 *p*는 *decimal-expression*의 정밀도입니다. 결과의 실제 길이는 포함된 후미 문자를 제외한 결과를 표시하는 데 사용될 수 있는 최소 문자입니다. 선행 0은 포함되지 않습니다. 인수가 음인 경우 결과는 음의 부호로 시작합니다. 그렇지 않은 경우 결과는 숫자로 시작합니다.

결과의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

부동 소수점 대 CLOB

floating-point expression
내장 부동 소수점 자료 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식

decimal-character
결과 문자 스트링에 소수 자리를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 소수점을 사용합니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 부동 소수점 상수의 형식에서 인수의 가변 길이 문자 스트링 표시입니다.

결과의 길이 속성은 24입니다. 결과의 실제 길이는 0이 아닌 하나의 숫자 다음에 *decimal-character* 및 일련의 숫자가 오는 가수와 같은 인수 값을 나타내는 최소 문자입니다. 인수가 음인 경우 결과의 첫 번째 문자는 음의 부호이며 그렇지 않은 경우 첫 번째 문자는 숫자입니다. 인수가 0인 경우 결과는 0E0입니다.

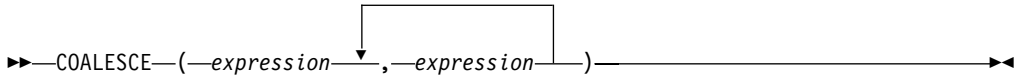
결과의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

예

- 다음의 함수는 스트링 ‘This is a CLOB’에 CLOB를 리턴합니다.

```
SELECT CLOB('This is a CLOB')
  FROM SYSIBM.SYSDUMMY1
```

COALESCE



COALESCE 함수는 널이 아닌 첫 번째 표현식 값을 리턴합니다.

인수는 호환될 수 있어야 합니다. 문자 스트링 인수는 날짜 시간 값으로 호환될 수 있습니다. 자료 유형 호환성에 대한 정보는 80 페이지의 『지정과 비교』를 참조하십시오. 인수는 내장 자료 유형이거나 고유한 유형입니다.[30]

인수는 지정된 순서대로 평가되고, 함수의 결과는 널이 아닌 첫 번째 인수입니다. 결과는 모든 인수가 널이 될 수 있는 경우에만 널이 될 수 있으며, 결과는 모든 인수가 널인 경우에만 널입니다.

선택된 인수는 필요한 경우 결과 속성으로 변환됩니다. 결과 속성은 93 페이지의 『결과 자료 유형에 대한 규칙』에 설명된 대로 모든 피연산자에 의해 판별됩니다.

예

- DEPARTMENT 표의 모든 행에서 모든 값을 선택할 때 부서 관리자(MGRNO)가 누락된 경우(즉, 널인 경우) 값 'ABSENT'를 리턴합니다.

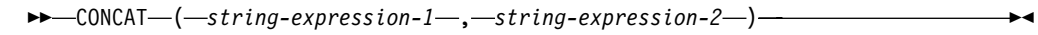
```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- EMPLOYEE 표의 모든 행에서 사원 번호(EMPNO) 및 급여(SALARY)를 선택할 때 급여가 누락된 경우(즉 널인 경우) 값 0을 리턴합니다.

```
SELECT EMPNO, COALESCE(SALARY,0)
     FROM EMPLOYEE
```

30. 이 함수를 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 호환되는 모든 자료 유형이 인수로 승인되기 때문에, 고유한 유형을 지원하기 위한 추가 서명을 작성할 필요가 없습니다.

CONCAT



CONCAT 함수는 두 문자 인수를 결합합니다. 인수는 호환 스트링이어야 합니다. 자료 유형 호환성에 대한 정보는 80 페이지의 『지정과 비교』를 참조하십시오.

함수의 결과는 첫 번째 인수 뒤에 두 번째 인수가 있는 문자열입니다. 인수 중의 하나가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 인수 중 하나가 널인 경우 결과는 널값입니다.

CONCAT 함수는 CONCAT 연산자와 같습니다. 자세한 내용은 129 페이지의 『연결 연산자를 사용하는 경우』를 참조하십시오.

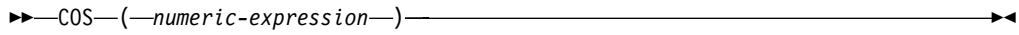
예

- 열 FIRSTNME과 열 LASTNAME을 연결합니다.

```
SELECT CONCAT(FIRSTNME, LASTNAME)
     FROM EMPLOYEE
  WHERE EMPNO ='000010'
```

  값 'CHRISTINEHAAS'를 리턴합니다.

COS



COS 함수는 인수의 코사인을 리턴하는데 인수는 각으로 표현된 라디안 입니다. COS 및 ACOS 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 COSINE이 값 1.5인 DECIMAL(2,1) 호스트 변수라고 가정합니다.

```
SELECT COS(:COSINE)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 0.07을 리턴합니다.

COSH

COSH 함수는 인수의 쌍곡선의 코사인을 리턴하는데 인수는 각으로 표현된 라디안 입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

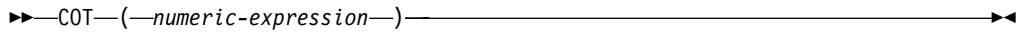
예

- 호스트 변수 HCOS가 값 1.5인 DECIMAL(2,1) 호스트 변수라고 가정합니다.

```
SELECT COSH(:HCOS)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 2.35를 리턴합니다.

COT



COT 함수는 인수의 코탄젠트를 리턴하는데 인수는 각으로 표현된 라디안 입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

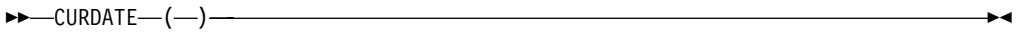
예

- 호스트 변수 COTAN가 값 1.5인 DECIMAL(2,1) 호스트 변수라고 가정합니다.

```
SELECT COT(:COTAN)
  FROM SYSIBM.SYSDUMMY1
```

대략 값 0.07을 리턴합니다.

CURDATE



CURDATE 함수는 SQL문이 현재 서버에서 실행될 때 날짜 시간 시계를 읽어서 날짜를 리턴합니다. CURDATE 함수로 리턴된 값은 CURRENT DATE 특수 레지스터에 의해 리턴된 값과 같습니다.

결과 자료 유형은 날짜입니다. 결과는 널이 될 수 없습니다.

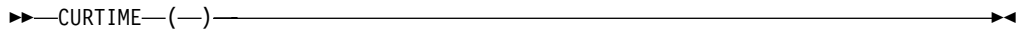
함수가 하나의 SQL문에서 한번 이상 사용되는 경우 또는 하나의 명령문에서 CURTIME 또는 NOW 스칼라 함수 또는 CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP 특수 레지스터가 사용되는 경우 모든 값은 하나의 시계를 읽습니다.

예

- 날짜 시간 시계의 현재 날짜를 리턴합니다.

```
SELECT CURDATE()
  FROM SYSIBM.SYSDUMMY1
```

CURTIME



CURTIME 함수는 SQL문이 현재 서버에서 실행될 때 날짜 시간 시계를 읽어서 시간을 리턴합니다. CURTIME 함수로 리턴된 값은 CURRENT TIME 특수 레지스터로 리턴된 값과 같습니다.

결과 자료 유형은 시간입니다. 결과는 널이 될 수 없습니다.

함수가 하나의 SQL문에서 한번 이상 사용되는 경우 또는 하나의 명령문에서 CURDATE 또는 NOW 스칼라 함수 또는 CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP 특수 레지스터가 사용되는 경우 모든 값은 하나의 시계를 읽습니다.

예

- 날짜 시간 시계의 현재 시간을 리턴합니다.

```
SELECT CURTIME()
  FROM SYSIBM.SYSDUMMY1
```

DATE

DATE 함수는 값에서 날짜를 리턴합니다.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 다음 중 하나여야 합니다.
  - 날짜 또는 시간소인의 유효한 문자 스트링 표현. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
  - 유효한 날짜가 *yyyynnn* 형식으로 표시되는 실제 길이 7의 문자 스트링. 여기서, *yyyy*는 연도를 표시하는 수이고 *nnn*은 해당 년도의 일 수를 표시하는 001 - 366 사이의 숫자입니다.
- *expression*이 숫자인 경우, 3652059 이하의 양수여야 합니다.

주: 또한 CAST 표현식은 날짜 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수 결과는 날짜입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 시간소인인 경우:

  결과는 시간소인의 날짜 부분입니다.
- 인수가 날짜인 경우:

  결과는 해당 날짜입니다.
- 인수가 수인 경우:

  결과는 January 1, 0001다음의 *n*-1일인 날짜이며 여기서 *n*은 수의 정수 부분입니다.
- 인수가 문자 스트링인 경우:

  결과는 스트링으로 표현된 날짜 또는 스트링으로 표현된 시간소인 값의 날짜 부분입니다.

  날짜의 스트링 표시가 SBCS 자료의 디폴트 CCSID와 같지 않은 CCSID를 갖는 SBCS 자료일 때 값은 날짜 값을 해석하고 변환하기 전에 SBCS자료의 디폴트 CCSID로 변환됩니다.

날짜의 스트링 표시가 혼합 자료의 디폴트 CCSID와 같지 않은 CCSID를 갖는 혼합 자료일 때 값은 날짜 값을 해석하고 변환하기 전에 혼합 자료의 디폴트 CCSID로 변환됩니다.

예

- 열 RECEIVED (TIMESTAMP) '1988-12-25-17.12.30.000000'와 같은 내부 값이 있다고 가정합니다.

```
SELECT DATE(RECEIVED)
  FROM IN_TRAY
```

  '1988-12-25'의 내부 표시가 됩니다.

- 날짜의 ISO 스트링 표시에 적용된 DATE 스칼라 함수는 다음과 같습니다.

```
SELECT DATE('1988-12-25')
  FROM SYSIBM.SYSDUMMY1
```

  '1988-12-25'의 내부 표시가 됩니다.

- 날짜의 EUR 스트링 표시에 적용된 DATE 스칼라 함수는 다음과 같습니다.

```
SELECT DATE('25.12.1988')
  FROM SYSIBM.SYSDUMMY1
```

  '1988-12-25'의 내부 표시가 됩니다.

- 양수에 적용된 DATE 스칼라 함수는 다음과 같습니다.

```
SELECT DATE(35)
  FROM SYSIBM.SYSDUMMY1
```

  '0001-02-04'의 내부 표시가 됩니다.

DAY

DAY 함수는 값의 일(day) 부분을 리턴합니다.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 날짜 기간 또는 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 날짜, 시간소인, 날짜 또는 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 일(day) 부분이며 1 - 31 사이의 정수 값입니다.
- 인수가 시간소인 기간 또는 시간소인 기간인 경우:

  결과는 값의 요일 부분이고 -99에서 00 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- PROJECT 표를 사용하여 END_DAY(SMALLINT)를 WELD LINE PLANNING 프로젝트(PROJNAME) 중지 예정일(PRENDATE)로 설정합니다.

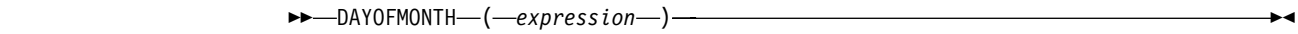
```
SELECT DAY(PRENDATE)
  INTO :END_DAY
  FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  END_DAY가 15로 설정됩니다.
- 두 날짜 사이의 차잇점 중 날짜 부분을 리턴합니다.

```
SELECT DAY( DATE('2000-03-15') - DATE('1999-12-31') )
  FROM SYSIBM.SYSDUMMY1
```

  값은 15가 됩니다.

DAYOFMONTH



DAYOFMONTH 함수는 한 달중 날짜를 표시하는 1-31 사이의 정수를 리턴합니다.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- PROJECT 표를 사용하여 END_DAY(SMALLINT)를 WELD LINE PLANNING 프로젝트(PROJNAME) 중지예정일(PRENDATE)로 설정합니다.

```
SELECT DAYOFMONTH(PRENDATE)
  INTO :END_DAY
  FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  END_DAY가 15로 설정됩니다.

DAYOFWEEK

DAYOFWEEK 함수는 한 주일의 요일을 표시하는 1 - 7 사이의 정수를 리턴하며 1은 일요일, 7은 토요일입니다. 다른 방법은 209 페이지의 『DAYOFWEEK_ISO』를 참조하십시오.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 DAY_OF_WEEK(INTEGER)를 Christine Haas(EMPNO='000010')의 입사일(HIREDATE)로 설정합니다.

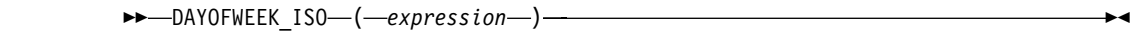
```
SELECT DAYOFWEEK(HIREDATE)
  INTO :DAY_OF_WEEK
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

  DAY_OF_WEEK가 6 금요일로 설정됩니다.

- 다음 조회는 네 가지 값: 1, 2, 1 및 2를 리턴합니다.

```
SELECT DAYOFWEEK(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK(TIMESTAMP('10/12/1998','01.02')),
       DAYOFWEEK(CAST(CAST('10/11/1998' AS DATE)) AS CHAR(20))),
       DAYOFWEEK(CAST(TIMESTAMP('10/12/1998','01.02') AS CHAR(20))),
   FROM SYSIBM.SYSDUMMY1
```

DAYOFWEEK_ISO



DAYOFWEEK_ISO 함수는 한 주일의 요일을 표시하는 1 - 7 사이의 정수를 리턴하며 1은 월요일, 7은 일요일입니다. 다른 방법은 208 페이지의 『DAYOFWEEK』를 참조하십시오.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 DAY_OF_WEEK(INTEGER)를 Christine Haas(EMPNO='000010')의 입사일(HIREDATE)로 설정합니다.

```
SELECT DAYOFWEEK_ISO(HIREDATE)
  INTO :DAY_OF_WEEK
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

  샘플 표를 사용할 때 DAY_OF_WEEK가 5, 금요일로 설정됩니다.

- 다음 조회는 네 가지 값 7, 1, 7 및 1을 리턴합니다.

```
SELECT DAYOFWEEK_ISO(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK_ISO(TIMESTAMP('10/12/1998','01.02')),
       DAYOFWEEK_ISO(CAST(CAST('10/11/1998' AS DATE)) AS CHAR(20))),
       DAYOFWEEK_ISO(CAST(TIMESTAMP('10/12/1998','01.02') AS CHAR(20))),
  FROM SYSIBM.SYSDUMMY1
```

DAYOFYEAR

DAYOFYEAR 함수는 1년 중 날짜를 표시하는 1 - 366 사이의 정수를 리턴하며 1은 1월 1일입니다.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- EMPLOYEE 표를 사용하여 사원들이 입사한 날(HIREDATE)의 평균 값을 호스트 변수 AVG_DAY_OF_YEAR (INTEGER)에 설정합니다.

```
SELECT AVG(DAYOFYEAR(HIREDATE))
  INTO :AVG_DAY_OF_YEAR
    FROM EMPLOYEE
```

  AVG_DAY_OF_YEAR가 202로 설정됩니다.

DAYS

DAYS 함수는 날짜의 정수 표시를 리턴합니다.

인수는 내장 자료 유형 date, timestamp, 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는 January 1, 0001에서 *D*까지 일수보다 1 많으며 여기서, *D*는 DATE 함수가 인수에 적용된 경우 결과로 나오는 날짜입니다.

예

- PROJECT 표를 사용하여 호스트 변수 EDUCATION_DAYS(INTEGER)를 프로젝트(PROJNO) ‘IF2000’의 예상 경과일(PRENDATE - PRSTDATE)로 설정합니다.

```
SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
  INTO :EDUCATION_DAYS
  FROM PROJECT
  WHERE PROJNO = 'IF2000'
```

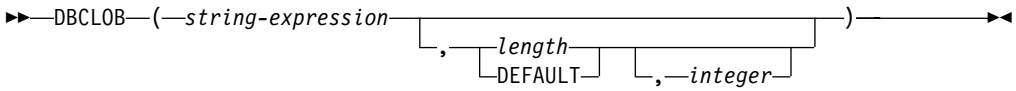
  EDUCATION_DAYS가 396으로 설정됩니다.

- PROJECT 표를 사용하여 호스트 변수 TOTAL_DAYS(INTEGER)를 부서(DEPTNO) ‘E21’의 모든 프로젝트에서 평가된 경과 일수(PRENDATE - PRSTDATE)의 합으로 설정하십시오.

```
SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
  INTO :TOTAL_DAYS
  FROM PROJECT
  WHERE DEPTNO = 'E21'
```

  TOTAL_DAYS가 1484로 설정됩니다.

DBCLOB



DBCLOB 함수는 스트링 표현식의 DBCLOB 표시를 리턴합니다.

주: 또한 CAST 표현식은 2바이트 문자 큰 오브젝트 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

string-expression

문자 스트링 또는 그래픽 스트링 값을 리턴하는 표현식. BLOB일 수 없습니다. CHAR 또는 VARCHAR 비트 자료일 수 없습니다. 세 번째 인수가 지정되지 않으면 GRAPHIC 또는 CCSID가 65535인 VARGRAPHIC일 수 없습니다.

length

결과 가변 길이 그래픽 스트링에 길이 속성을 지정합니다. 값은 1 - 1 073 741 823 사이의 값이어야 합니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정되는 경우

- *expression*이 빈 스트링 상수이면 결과의 길이 속성은 1입니다.
- 그렇지 않으면 결과의 길이 속성은 인수가 첫 번째 인수의 길이 속성과 같습니다.

integer

결과 가변 길이 그래픽 스트링에 CCSID(코드화 문자 세트 ID)를 지정합니다. DBCS 또는 UCS-2 CCSID이어야 합니다. CCSID는 65535가 될 수 없습니다.

다음 규칙에서 S는 다음 중의 하나를 표시합니다.

- 스트링 표현식이 외국어 코드화 체계의 자료가 들어 있는 호스트 변수인 경우 S는 자국어 코드화 체계의 CCSID로 자료를 변환한 다음의 표현식 결과입니다(자세한 내용은 34 페이지의 『문자 변환』을 참조하십시오).
- S는 스트링 표현식이 자국어 코드화 체계 자료인 경우의 스트링 표현식입니다.

세 번째 인수가 지정되지 않고 첫 번째 인수가 문자인 경우 결과의 CCSID는 혼합 CCSID로 판별됩니다. M을 혼합 CCSID로 지정합니다. M은 다음과 같이 판별됩니다.

- S의 CCSID가 혼합 CCSID인 경우 M은 그 CCSID입니다.
- S의 CCSID가 SBCS CCSID인 경우:
  - S의 CCSID가 혼합 CCSID와 연관된 경우 M은 그 CCSID입니다.
  - 그렇지 않으면 연산이 허용되지 않습니다.

다음 표는 M에 따라 CCSID 결과를 요약합니다.

M	결과 CCSID	설명	DBCS 대체 문자
930	300	일본어 EBCDIC	X'FEFE'
933	834	한국어 EBCDIC	X'FEFE'
935	837	중국어 EBCDIC	X'FEFE'
937	835	대만어 EBCDIC	X'FEFE'
939	300	일본어 EBCDIC	X'FEFE'
5026	4396	일본어 EBCDIC	X'FEFE'
5035	4396	일본어 EBCDIC	X'FEFE'

세 번째 인수가 지정되지 않고 첫 번째 인수가 문자가 아닌 경우 결과의 CCSID는 첫 번째 인수의 CCSID와 같습니다.

결과의 실제 길이는 결과의 길이 속성과 *expression*의 실제 길이의 최소값입니다. 결과 DBCLOB의 길이 속성이 첫 번째 인수의 실제 길이보다 작은 경우 절단이 수행되고 경고는 리턴되지 않습니다.

함수의 결과는 DBCLOB 스트링입니다. 표현식이 널이 될 수 있으면, 결과도 널이 될 수 있습니다. 표현식이 널인 경우 결과는 널값입니다. 표현식이 공백 스트링 또는 EBCDIC 스트링 `X'0E0F'`인 경우 결과는 공백 스트링입니다.

결과가 DBCS 그래픽 자료인 경우 SBCS 및 DBCS 문자의 등가가 CCSID에 관계없이 M에 따라 다르고, 인수의 모든 2바이트 코드점은 DBCS 문자로 간주되고, 인수의 모든 1바이트 코드점은 EBCDIC 혼합 자료 시프트 코드 `X'0E'` 및 `X'0F'`를 제외하고 SBCS 문자로 간주됩니다.

- 인수의 n 번째 문자가 DBCS 문자인 경우 결과의 n 번째 문자는 해당 DBCS 문자입니다.
- 인수의 n 번째 문자가 등가의 DBCS 문자를 갖는 SBCS 문자인 경우 결과의 n 번째 문자는 등가의 DBCS 문자입니다.
- 인수의 n 번째 문자가 등가의 DBCS 문자가 아닌 SBCS 문자인 경우 결과의 n 번째 문자는 DBCS 대체 문자입니다.

결과가 UCS-2 그래픽 자료인 경우 인수의 각 문자는 결과 문자를 판별합니다. 결과의 n 번째 문자가 인수의 n 번째 문자와 등가의 UCS-2입니다.

예

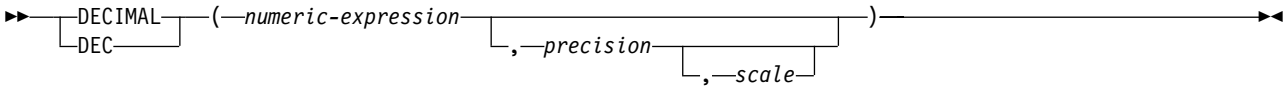
- EMPLOYEE 표를 사용하여 호스트 변수 VAR_DESC(VARGRAPHIC(24))를 사원 번호(EMPNO) '000050'의 이름(FIRSTNME)과 동일한 DBCLOB로 설정합니다.

DBCLOB

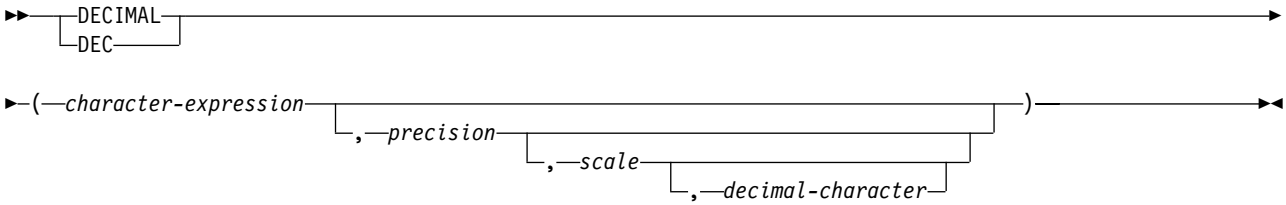
```
SELECT DBCLOB(FIRSTNME)
  INTO :VAR_DESC
     FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

DECIMAL 또는 DEC

숫자 대 소수



문자 대 소수



DECIMAL 함수는 다음의 소수 표시를 리턴합니다.

- 수
- 소수의 문자 스트링 표시
- 정수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 십진 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수의 결과는 정밀도 *p*와 스케일 *s*인 소수이며, 여기서 *p* 및 *s*는 두 번째 및 세 번째 인수입니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

숫자 대 소수

numeric-expression

내장된 숫자 자료 유형의 값을 리턴하는 표현식입니다.

precision

1보다 크거나 같고 31보다 작거나 같은 값으로 설정된 정수 상수.

*precision*의 디폴트는 *numeric-expression*의 자료 유형에 따라 다릅니다.

- 부동 소수점, 소수, 숫자 또는 0이 아닌 스케일 2진인 경우 15
- 큰 정수의 경우 19
- 큰 정수인 경우 11
- 작은 정수인 경우 5

scale

0 이상이고 *precision* 이하인 정수 상수. 지정되지 않은 경우 디폴트는 0입니다.

결과는 첫 번째 인수가 정밀도 *p*와 스케일 *s*인 소수 열 또는 변수에 지정된 경우 발생하는 같은 수입니다. 수 전체를 표시하기 위해 필요한 소수 자릿수가 *p-s*보다 큰 경우 오류가 발생합니다.

문자 대 소수

character-expression

숫자의 문자 스트링 표현을 포함해야 하는 표현식. 선행 및 후미 공백은 제거되고 결과 스트링은 정수 또는 소수 상수를 형식화하기 위한 규칙을 따릅니다. 표현식은 CLOB가 아니어야 합니다.

precision

1보다 크거나 같고 31보다 작거나 같은 정수 상수. 지정되지 않은 경우 디폴트는 15입니다.

scale

0 이상이고 *precision* 이하인 정수 상수. 지정되지 않은 경우 디폴트는 0입니다.

decimal-character

수 전체에서 *문자 표현*의 소수 자리를 분리하는 데 사용된 1바이트 문자 상수를 지정합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않은 경우 소수점은 디폴트 분리 문자입니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 CAST(*character-expression* AS DECIMAL(*p,s*))의 결과로 나오는 수와 같습니다. 자릿수는 소수 문자의 오른쪽 자릿수가 스케일 *s*보다 큰 경우에 끝이 절단됩니다. *character-expression*에서 소수 문자(수 전체 부분) 왼쪽의 유효 자릿수가 *p-s*보다 큰 경우 오류가 발생합니다. 디폴트 소수 문자는 *decimal-character* 인수가 지정된 경우 서브스트링에서 유효하지 않습니다.

예

- DECIMAL 함수를 사용하여(정밀도 5와 스케일 2인) DECIMAL 자료 유형이 EMPLOYEE 표에 있는 EDLEVEL 열(자료 유형 = SMALLINT)의 선택 리스트로 리턴되도록 합니다. EMPNO 열이 반드시 선택 리스트에 있어야 합니다.

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- PROJECT 표를 사용하여 호스트 변수에 지정된 기간으로 증가된 모든 입사일(PRSTDATE)을 선택하십시오. 호스트 변수 PERIOD가 유형 INTEGER라고 가정합니다. 그런 다음, 해당 값을 날짜 기간으로 사용하려면 해당 값은 DECIMAL (8,0)과 같은 "캐스트"이어야 합니다.

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- SALARY 열에 대한 갱신은 소수 문자로 쉼표를 사용하는 문자 스트링으로 창을 통해 입력됩니다(예를 들어, 사용자는 21400,50을 입력합니다). 어플리케이션으로 유효성이 검증되면 CHAR(10)로 정의된 호스트 변수 newsalary에 지정됩니다.

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid
```

SALARY 값은 21400.50이 됩니다.

DEGREES

DEGREES 함수는 라디안으로 표현된 각도 인수의 도수를 리턴합니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

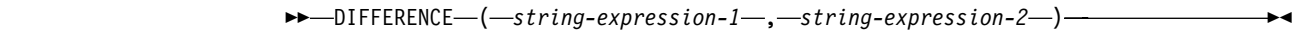
예

- 호스트 변수 RAD를 값 3.142인 DECIMAL(4,3) 호스트 변수라고 가정합니다.

```
SELECT DEGREES(:RAD)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 180.0을 리턴합니다.

DIFFERENCE



DIFFERENCE 함수는 스트링에 SOUNDEX 함수를 적용하여 두 스트링의 발음의 차이를 나타내는 0 - 4 사이의 값을 리턴합니다. 4 값이 가장 잘 일치하는 발음입니다.

인수는 내장 스트링 자료 유형이고 BLOB, CLOB, DBCLOB이 아니어야 합니다.

결과의 자료 유형은 INTEGER입니다. 어떤 인수가 널이 될 수 있다면 결과도 널이 될 수 있으며, 어떤 인수가 널이면 결과도 널입니다.

예

- 한 예로 다음 명령문을 가정합니다.

```
SELECT DIFFERENCE('CONSTRAINT','CONSTANT'),
       SOUNDEX('CONSTRAINT'),
       SOUNDEX('CONSTANT')
  FROM SYSIBM.SYSDUMMY1
```

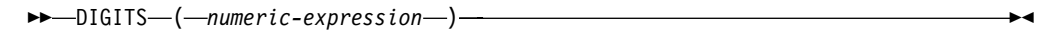
  4, C523 및 C523이 리턴됩니다. 두 개의 스트링이 같은 SOUNDEX 값을 리턴하므로 차이는 4(가장 높은 값)입니다.

- 한 예로 다음 명령문을 가정합니다.

```
SELECT DIFFERENCE('CONSTRAINT','CONTRITE'),
       SOUNDEX('CONSTRAINT'),
       SOUNDEX('CONTRITE')
  FROM SYSIBM.SYSDUMMY1
```

  2, C523 및 C536이 리턴됩니다. 이 경우 두 스트링은 서로 다른 SOUNDEX 값을 리턴하므로 두 값의 차이는 매우 적습니다.

DIGITS



DIGITS 함수는 수 절대값의 문자 스트링을 리턴합니다.

인수는 내장 숫자 자료 유형 SMALLINT, INTEGER, BIGINT 또는 DECIMAL이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 스케일에 관계없이 인수의 절대 값을 표시하는 고정 길이 문자 스트링입니다. 결과는 부호 또는 소수점이 포함되지 않습니다. 대신에 필요한 경우 선행 0을 포함하여 스트링에 채워 자릿수를 구성합니다. 스트링의 길이는 다음과 같습니다.

- 인수가 작은 0 스케일 정수인 경우 5
- 인수가 큰 0 스케일 정수인 경우 10
- 인수가 큰 정수인 경우 19
- 인수가 정밀도 *p*인 소수 또는 0이 아닌 스케일 정수인 경우 *p*

문자 스트링의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

예

- TABLEX 표에 10자리수가 포함된 INTCOL이라는 INTEGER 열이 있는 것으로 가정합니다. 열 INTCOL에 포함된 첫 네자리의 모든 조합을 나열합니다.

```
SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
  FROM TABLEX
```

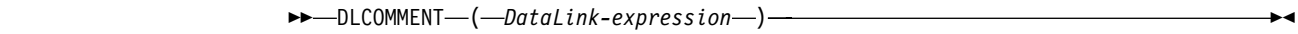
- COLUMNX가 DECIMAL(6,2) 자료 유형을 가지며, 그 값 중의 하나가 -6.28이라고 가정합니다.

```
SELECT DIGITS(COLUMNX)
  FROM TABLEX
```

  값 1123을 리턴합니다.

  결과는 스트링이 이 길이까지 채워진 선행 0이 있는 (열의 정밀도) 길이 6의 스트링입니다. 부호 또는 소수점이 결과에 표시되지 않습니다.

DLCOMMENT



DLCOMMENT 함수는 있는 경우 자료 링크 값에서 주석 값을 리턴합니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 VARCHAR(254)입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 명령문을 준비하여 HOCKEY_GOALS 표에서 ARTICLES 열에 대한 링크로부터 날짜, 설명 및 주석을 선택합니다. 선택되는 행은 Richard 형제(Maurice 또는 Henri)가 구한 결과 행입니다.

```
stmtvar = "SELECT DATE_OF_GOAL, DESCRIPTION, DLCOMMENT(ARTICLES)
                 FROM HOCKEY_GOALS
                 WHERE BY_PLAYER = 'Maurice Richard' OR BY_PLAYER = 'Henri Richard' ";
EXEC SQL PREPARE HOCKEY_STMT FROM :stmtvar;
```

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

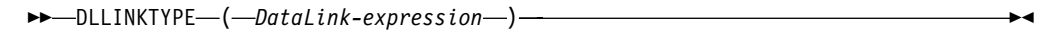
```
INSERT INTO TBLA
  VALUES (DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment'))
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLCOMMENT(COLA)
  FROM TBLA
```

  'A comment' 값을 리턴합니다.

DLLINKTYPE



DLLINKTYPE 함수는 자료 링크 값에서 링크 유형 값을 리턴합니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 VARCHAR(4)입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

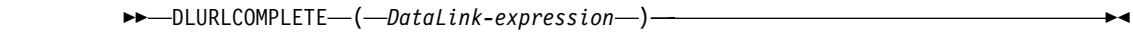
```
INSERT INTO  TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLLINKTYPE(COLA)
  FROM TBLA
```

  값 'URL'을 리턴합니다.

DLURLCOMPLETE



DLURLCOMPLETE 함수는 링크 유형 URL인 DataLink 값에서 완전한 URL을 리턴합니다. 값은 '://'로 DLURLSCHEME를 연결한 다음 DLURLSERVER 그리고 DLURLPATH를 연결하여 리턴되는 값과 같습니다. DataLink가 속성 FILE LINK CONTROL 및 READ PERMISSION DB를 갖는 경우 값은 파일 액세스 토큰을 포함합니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 변수 길이 스트링입니다. 길이 속성은 DataLink 속성에 따라 다릅니다.

- DataLink가 속성 FILE LINK CONTROL 및 READ PERMISSION DB를 갖는 경우 결과의 길이 속성은 인수의 길이 속성 더하기 19입니다.
- 그렇지 않으면 결과의 길이 속성은 인수의 길이 속성입니다.

DataLink 값이 주석만을 포함하는 경우 리턴된 결과는 0 길이 스트링입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

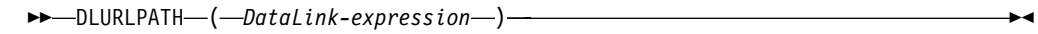
```
INSERT INTO  TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLURLCOMPLETE(COLA)
  FROM TBLA
```

  'HTTP://DLFS.ALMADEN.IBM.COM/x/y/**********;a.b' 값을 리턴합니다. 여기서 **********는 액세스 토큰을 나타냅니다.

DLURLPATH



DLURLPATH 함수는 링크 유형 URL인 DataLink 값에서 주어진 서버 내의 파일을 액세스하기 위해 필요한 경로 및 파일명을 리턴합니다. 해당되는 경우 값은 파일 액세스 토큰을 포함합니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 변수 길이 스트링입니다. 길이 속성은 DataLink 속성에 따라 다릅니다.

- DataLink가 속성 FILE LINK CONTROL 및 READ PERMISSION DB를 갖는 경우 결과의 길이 속성은 인수의 길이 속성 더하기 19입니다.
- 그렇지 않으면 결과의 길이 속성은 인수의 길이 속성입니다.

DataLink 값이 주석만을 포함하는 경우 리턴된 결과는 0 길이 스트링입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

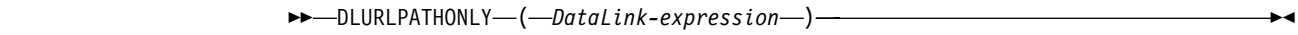
```
INSERT INTO  TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLURLPATH(COLA)
  FROM TBLA
```

  '/x/y/**********;a.b' 값을 리턴합니다. 여기서 **********는 액세스 토큰을 나타냅니다.

DLURLPATHONLY



DLURLPATHONLY 함수는 링크 유형 URL인 DataLink 값에서 주어진 서버의 파일을 액세스하기 위해 필요한 경로 및 파일명을 리턴합니다. 리턴된 값은 파일 액세스 토큰을 포함하지 않습니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 인수의 길이 속성과 같은 길이 속성을 갖는 가변 길이 스트링입니다.

DataLink 값이 주석만을 포함하는 경우 리턴된 결과는 0 길이 스트링입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

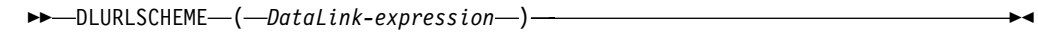
```
INSERT INTO  TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLURLPATHONLY(COLA)
  FROM TBLA
```

  값 '/x/y/a.b'를 리턴합니다.

DLURLSCHEME



DLURLSCHEME 함수는 링크 유형 URL인 DataLink 값에서 스키마를 리턴합니다. 값은 항상 대문자입니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 VARCHAR(20)입니다.

DataLink 값이 주석만을 포함하는 경우 리턴된 결과는 0 길이 스트링입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

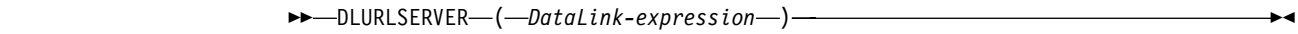
```
INSERT INTO  TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLURLSCHEME(COLA)
  FROM TBLA
```

  값 'HTTP'를 리턴합니다.

DLURLSERVER



DLURLSERVER 함수는 링크 유형 URL인 DataLink 값에서 파일 서버를 리턴합니다. 값은 항상 대문자입니다.

인수는 내장 DataLink 자료 유형이 결과 값이 되는 표현식이어야 합니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

함수의 결과는 인수의 길이 속성과 같은 길이 속성을 갖는 가변 길이 스트링입니다.

DataLink 값이 주석만을 포함하는 경우 리턴된 결과는 0 길이 스트링입니다.

문자 스트링의 CCSID는 *DataLink 표현식*의 CCICD와 같습니다.

예

- 다음은 스칼라 함수를 사용하여 TBLA 표에서 행의 COLA 열에 삽입된 DataLink 값입니다.

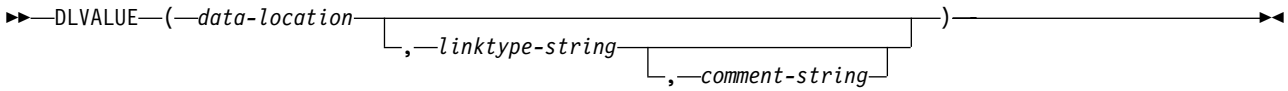
```
INSERT INTO  TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  다음은 해당 값에 대해 연산된 함수입니다.

```
SELECT DLURLSERVER(COLA)
  FROM TBLA
```

  값 ‘DLFS.ALMADEN.IBM.COM’을 리턴합니다.

DLVALUE



DLVALUE 함수는 DataLink 값을 리턴합니다. 함수가 UPDATE문의 SET구의 오른쪽에 있거나 또는 INSERT문의 VALUES절에 있으면, 일반적으로 파일에 대한 링크를 작성합니다. 그러나, (*자료 위치*가 0 길이 스트링인) 주석만이 지정되는 경우 DataLink 값은 파일 링크가 없으므로 공백 링크 속성으로 작성됩니다.

data-location

링크 유형이 URL인 경우 완전한 URL 값을 포함하는 문자 스트링 표현식입니다. 표현식이 공백 스트링이 아닌 경우 URL 스키마 및 URL 서버를 포함하여야 합니다. 문자 스트링 표현식의 실제 길이는 32718자 이하이어야 합니다.

linktype-string

DataLink 값의 링크 유형을 지정하는 선택적 문자 스트링 표현식. 유효한 값은 'URL'입니다.

comment-string

주석 또는 추가 위치 정보를 제공하는 선택적 문자 스트링 표현식. 문자 스트링 표현식의 실제 길이는 254자 이하이어야 합니다.

*comment-string*은 널값일 수 없습니다. *comment-string*이 지정되지 않으면 *comment-string*은 빈 스트링입니다.

첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

함수의 결과는 DataLink 값입니다.

DataLink의 CCSID는 다음 경우를 제외하고 *자료 위치*의 CCSID와 같습니다.

- *주석 스트링*이 혼합 자료이고 *자료 위치*가 혼합 자료가 아닌 경우 결과의 CCSID는 *주석 스트링*의 CCSID가 됩니다.[31]
- *자료 위치*가 비트 자료 CCSID(65535), UCS-2 그래픽 자료(13488), 터어키어 자료(905 또는 1026) 또는 일본어 자료(290, 930 또는 5026)를 갖는 경우 결과의 CCSID는 다음 표에서 설명됩니다.

31. *주석 스트링*의 CCSID가 5026 또는 930인 경우 결과의 CCSID는 939가 됩니다.

*자료 위치*의 **CCSID**	*주석 스트링*의 **CCSID**	**결과 CCSID**
65535	65535	작업 디폴트 CCSID
65535	65535가 아님	*주석 스트링* CCSID(CCSID가 290, 930, 5026, 905, 1026 또는 13488이 아닌 경우 CCSID는 다음 행에서 설명된 것처럼 수정됩니다).
290	모두	4396
930 또는 5026	모두	939
905 또는 1026	모두	500
13488	모두	500

이 함수를 사용하여 DataLink 값을 정의할 때 값의 최대 목표 길이를 고려하십시오. 예를 들어, 열이 DataLink(200)로 정의된 경우 *자료 위치*와 주석의 최대 길이는 200 바이트입니다.

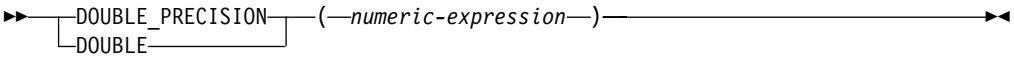
예

- 표에 행을 삽입합니다. 첫 번째 두 개 링크의 URL 값은 url_article 및 url_snapshot 라는 변수에 포함됩니다. url_snapshot_comment 변수에는 주석을 포함하여 스냅샷 링크가 함께 포함되어 있습니다. 아직 movie에 링크되지 않고 url_movie_comment 변수에 주석만 있습니다.

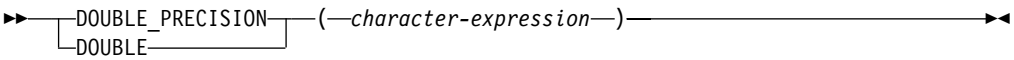
```
INSERT INTO HOCKEY_GOALS
  VALUES('Maurice Richard',
         'Montreal canadian',
         '?',
         'Boston Bruins,
         '1952-04-24',
         'Winning goal in game 7 of Stanley Cup final',
         DLVALUE(:url_article),
         DLVALUE(:url_snapshot, 'URL', :url_snapshot_comment),
         DLVALUE('', 'URL', :url_movie_comment) )
```

DOUBLE_PRECISION 또는 DOUBLE

숫자를 Double로



문자를 Double로



DOUBLE_PRECISION 및 DOUBLE 함수는 다음의 부동 소수점 표시를 리턴합니다.

- 수
- 소수의 문자 스트링 표시
- 정수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 배정밀도 부동 소수점 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

numeric-expression

내장된 숫자 자료 유형의 값을 리턴하는 표현식입니다.

결과는 표현식이 배정밀도 부동 소수점 열 또는 변수에 지정된 경우의 수와 같습니다.

character-expression

문자 스트링 값을 리턴하는 표현식 인수는 CLOB가 아니어야 합니다.

결과는 CAST(*character-expression* AS DOUBLE PRECISION)의 결과와 같습니다. 선행 및 후미 공백은 제거되고 결과 스트링은 부동 소수점, 정수 또는 소수 상수를 형식화하기 위한 규칙을 따릅니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

FLOAT는 DOUBLE_PRECISION 및 DOUBLE과 같습니다.

예

- EMPLOYEE 표를 사용하여 수당이 0이 아닌 사원에 대한 수당과 급여의 비율을 찾습니다. (SALARY 및 COMM)이 포함된 열은 DECIMAL 자료 유형입니다. 범위 밖의 결과가 나올 가능성을 제거하기 위해 나눗셈을 부동 소수점에 실행하도록 DOUBLE_PRECISION이 SALARY가 적용됩니다.

```
SELECT EMPNO, DOUBLE_PRECISION(SALARY)/COMM
     FROM EMPLOYEE
  WHERE COMM > 0
```

EXP

EXP 함수는 자연 대수(e)의 인수에 지정된 값 제곱근을 리턴합니다. EXP 및 LN 함수는 역산입니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 E를 값 3.453789832인 DECIMAL(10,9) 호스트 변수라고 가정합니다.

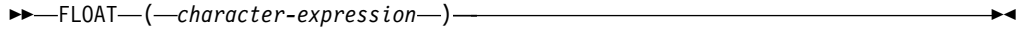
```
SELECT EXP(:E)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 31.62를 리턴합니다.

FLOAT

숫자를 **Float**로

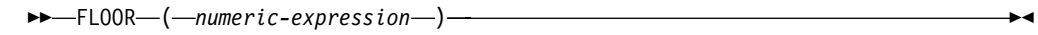
문자를 **Float**로



FLOAT 함수는 숫자의 부동 소수점 표현을 리턴합니다.

FLOAT는 DOUBLE_PRECISION 및 DOUBLE 함수와 같습니다. 자세한 내용은 230 페이지의 『DOUBLE_PRECISION 또는 DOUBLE』을 참조하십시오.

FLOOR



FLOOR 함수는 *numeric-expression*보다 작거나 같은 가장 큰 정수를 리턴합니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다.

함수의 결과는 만일 인수가 십진 수일 때 스케일이 0인 경우를 제외하고 인수와 동일한 자료 유형 및 길이 속성을 갖습니다. 예를 들어, DECIMAL(5,5)의 자료 유형을 갖는 인수는 DECIMAL(5,0)의 결과를 가져옵니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

예

- FLOOR 함수를 사용하여 소수점의 오른쪽 자릿수를 절단합니다.

```
SELECT FLOOR(SALARY)
     FROM EMPLOYEE
```

- 양수 및 음수 모두에 FLOOR를 사용합니다.

```
SELECT FLOOR( 3.5),
       FLOOR( 3.1),
       FLOOR(-3.1),
       FLOOR(-3.5),
  FROM SYSIBM.SYSDUMMY1
```

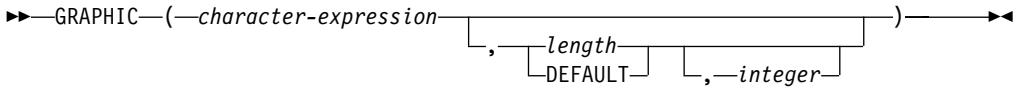
  리턴되는 값은 다음과 같습니다.

```
3.   3.  -4.  -4.
```

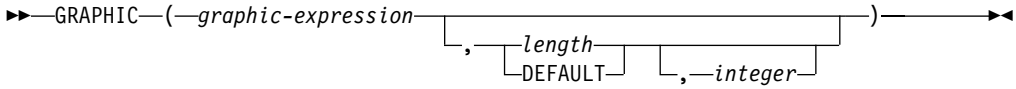
  의 결과를 갖습니다.

GRAPHIC

문자에서 그래픽으로



그래픽에서 그래픽으로



GRAPHIC 함수는 스트링 표현식을 고정 길이 그래픽 스트링으로 표시하여 리턴합니다.

주: 또한 CAST 표현식은 고정 길이 그래픽 스트링 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수의 결과는 고정 길이 그래픽 스트링(GRAPHIC)입니다.

표현식이 널이 될 수 있으면, 결과도 널이 될 수 있습니다. 표현식이 널인 경우 결과는 널값입니다.

문자에서 그래픽으로

character-expression

문자 스트링 표현식을 지정합니다. CHAR 또는 VARCHAR 비트 자료일 수 없습니다. 표현식이 공백 스트링 또는 EBCDIC 스트링 `X'0E0F'`인 경우 결과는 공백 스트링입니다.

length

결과의 길이 속성을 지정하며, 첫 번째 인수가 널이 아닌 경우 1-16383 사이의 정수 상수이고 첫 번째 상수가 널인 경우 1-16382 사이의 정수 상수입니다. *character-expression* 길이가 지정된 길이보다 짧은 경우 결과는 결과 길이까지 2 바이트 공백으로 채워집니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정된 경우 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다.

인수의 각 문자는 결과의 문자를 판별합니다. 결과 고정 길이 스트링의 길이 속성이 첫 번째 인수의 실제 길이보다 짧은 경우 절단이 수행되며 경고는 리턴되지 않습니다.

integer

결과의 CCSID를 지정합니다. DBCS 또는 UCS-2 CCSID이어야 합니다. CCSID는

65535가 될 수 없습니다. CCSID가 UCS-2 그래픽 자료인 경우 인수의 각 문자는 결과 문자를 판별합니다. 결과의 n 번째 문자가 인수의 n 번째 문자와 등가의 UCS-2입니다.

*integer*가 지정되지 않으면 결과의 CCSID는 혼합 CCSID에 의해 판별됩니다. M을 혼합 CCSID로 지정합니다.

다음 규칙에서 S는 다음 중의 하나를 표시합니다.

- 스트링 표현식이 외국어 코드화 체계의 자료가 들어 있는 호스트 변수인 경우 S는 자국어 코드화 체계의 CCSID로 자료를 변환한 다음의 표현식 결과입니다(자세한 내용은 34 페이지의 『문자 변환』을 참조하십시오).
- S는 스트링 표현식이 자국어 코드화 체계 자료인 경우의 스트링 표현식입니다.

M은 다음과 같이 판별됩니다.

- S의 CCSID가 혼합 CCSID인 경우 M은 그 CCSID입니다.
- S의 CCSID가 SBCS CCSID인 경우:
  - S의 CCSID가 혼합 CCSID와 연관된 경우 M은 그 CCSID입니다.
  - 그렇지 않으면 연산이 허용되지 않습니다.

다음 표는 M에 따라 CCSID 결과를 요약합니다.

M	결과 CCSID	설명	DBCS 대체 문자
930	300	일본어 EBCDIC	X'FEFE'
933	834	한국어 EBCDIC	X'FEFE'
935	837	중국어 EBCDIC	X'FEFE'
937	835	대만어 EBCDIC	X'FEFE'
939	300	일본어 EBCDIC	X'FEFE'
5026	4396	일본어 EBCDIC	X'FEFE'
5035	4396	일본어 EBCDIC	X'FEFE'

SBCS와 DBCS 문자의 등가는 M에 따라 다릅니다. CCSID와 관계없이 인수의 모든 2바이트 코드점은 DBCS 문자로 간주되고 모든 1바이트 코드점은 EBCDIC 혼합 자료 시프트 코드 `X'0E'` 및 `X'0F'`를 제외하고는 SBCS 문자로 간주됩니다.

- 인수의 n 번째 문자가 DBCS 문자인 경우 결과의 n 번째 문자는 해당 DBCS 문자입니다.
- 인수의 n 번째 문자가 등가의 DBCS 문자를 갖는 SBCS 문자인 경우 결과의 n 번째 문자는 등가의 DBCS 문자입니다.
- 인수의 n 번째 문자가 등가의 DBCS 문자가 아닌 SBCS 문자인 경우 결과의 n 번째 문자는 DBCS 대체 문자입니다.

그래픽에서 그래픽으로

graphic-expression

그래픽 스트링 표현식을 지정합니다.

length

결과의 길이 속성을 지정하며, 첫 번째 인수가 널이 아닌 경우 1-16383 사이의 정수 상수이고 첫 번째 상수가 널인 경우 1-16382 사이의 정수 상수입니다. *graphic-expression* 길이가 지정된 길이보다 짧은 경우 결과는 결과 길이까지 2바이트 공백으로 채워집니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정된 경우 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다.

*그래픽 표현식*의 길이가 결과 길이 속성보다 큰 경우 절단됩니다.절단된 문자가 모두 공백이 아니면 경고(SQLSTATE 01004)가 리턴됩니다.

integer

결과의 CCSID를 지정합니다. DBCS 또는 UCS-2 CCSID이어야 합니다. CCSID는 65535가 될 수 없습니다.

*integer*가 지정되지 않으면 결과의 CCSID는 첫 번째 인수의 CCSID입니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 DESC(GRAPHIC(24))를 사원 번호(EMPNO) '000050'의 이름(FIRSTNME)과 동일한 GRAPHIC 값으로 설정합니다.

```
SELECT GRAPHIC( VARGRAPHIC(FIRSTNME))
  INTO :DESC
     FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

HASH

HASH 함수는 값 집합의 부분 값을 리턴합니다. 또는 PARTITION 함수를 참조하십시오. 파티션 번호에 대한 자세한 정보는 DB2 Multisystem 책을 참조하십시오.

date, time, timestamp, floating-point 또는 DataLink 값을 제외한 어떤 내장 자료 유형도 인수가 될 수 있습니다.

함수의 결과는 0 - 1023 사이의 값인 큰 정수입니다.

인수가 널인 경우 결과는 0입니다. 결과는 널이 될 수 없습니다.

예

- HASH 함수를 사용하여 파티션 키가 EMPNO 및 LASTNAME로 구성된 파티션을 판별합니다. 이 조회는 EMPLOYEE의 모든 행에 대한 파티션 번호를 리턴합니다.

```
SELECT HASH(EMPNO, LASTNAME)
     FROM EMPLOYEE
```

HEX

HEX 함수는 값의 16진 표시를 리턴합니다.

어떤 내장 자료 유형도 인수가 될 수 있습니다.

함수의 결과는 문자 스트링입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는 16진 자릿수의 스트링이며, 처음 두 자릿수는 인수의 첫 번째 바이트를 표시하고, 다음 두 자릿수는 인수의 두 번째 바이트, 등을 표시합니다. 인수가 datetime 값인 경우, 결과는 인수에 대한 정수 형태의 16진 표시입니다.[32]

인수가 가변 길이 스트링인 경우 결과는 가변 길이 스트링입니다. 그렇지 않으면 결과는 인수의 고정 길이 스트링입니다. 결과의 길이 속성은 인수의 기억장치 길이 속성에 두 배입니다. 기억장치 길이 속성에 대한 자세한 내용은 541 페이지의 『CREATE TABLE』을 참조하십시오.

결과의 길이 속성은 고정 길이 결과 32766보다 클 수 없으며 가변 길이 결과 32740보다 클 수 없습니다.

CCSID의 스트링은 현재 서버에서 디폴트 SBCS CCSID입니다.

예

- HEX 함수를 사용하여 각 사원의 교육 레벨을 16진 표시로 리턴합니다.

```
SELECT FIRSTNME, MIDINIT, LASTNAME, HEX(EDLEVEL)
    FROM EMPLOYEE
```

32. DATE, TIMESTAMP, 및 NUMERIC 자료 유형에 대한 16진 표시는 이들 자료 유형에 대한 내부 양식이 다르기 때문에 다른 데이터베이스 제품과 다릅니다.

HOUR

HOUR 함수는 값의 시간 부분을 리턴합니다.

인수는 내장 자료 유형 time, timestamp, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 시간 기간 또는 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 날짜, 시간소인, 날짜 또는 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 시간 부분이며, 0 - 24 사이의 정수입니다.
- 인수가 소요 시간 또는 시간소인 기간인 경우:

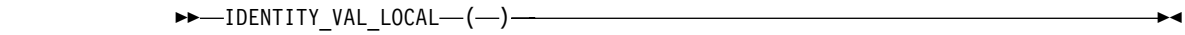
  결과는 값의 시 부분이고 -99에서 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- CL_SCHED 샘플 표를 사용하여 오후에 시작하는 모든 클래스를 선택합니다.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL



IDENTITY_VAL_LOCAL는 ID 열에 가장 최근에 지정된 값을 리턴하는 비결정형 함수입니다.

함수에는 입력 매개변수가 없습니다. 결과 값과 대응되는 ID 열의 실제 자료 유형과 상관 없이 결과는 DECIMAL(31,0)입니다.

리턴된 값은 ID 열이 들어 있는 표의 가장 최신 INSERT문에 지정된 표의 ID 열에 지정된 값입니다. INSERT문은 동일한 레벨에서 실행되어야 합니다. 즉, 값은 지정된 다음 값으로 대체되기 전까지는 해당 값이 지정된 레벨 내에서 로컬로 사용할 수 있어야 합니다. 트리거, 함수 또는 저장 프로시듀어가 호출되면 새로운 레벨이 시작됩니다. 트리거 조건은 트리거된 관련 조치와 동일한 레벨입니다.

지정된 값은 사용자가 제공한 값(ID 열이 GENERATED BY DEFAULT로 정의된 경우)이거나 데이터베이스 관리자에 의해 생성된 ID 값입니다.

결과는 널(null)이 될 수 있습니다. 현재 처리 레벨에서 ID 열이 들어 있는 표에 대해 INSERT문이 실행되지 않은 경우 결과는 널(null)입니다. 여기에는 삽입 트리거 전후의 함수 호출이 포함됩니다.

IDENTITY_VAL_LOCAL 함수의 결과는 다음 명령문의 영향을 받지 않습니다.

- ID 열이 들어 있지 않은 테이블의 INSERT 명령문
- UPDATE 명령문
- COMMIT 명령문
- ROLLBACK 명령문

주

다음은 함수를 여러 가지 상황에서 호출했을 때의 동작입니다.

INSERT문의 VALUES 절 내에서 함수 호출

INSERT문의 표현식은 INSERT문의 목표 열에 값이 지정되기 전에 평가됩니다. 즉, INSERT문에서 IDENTITY_VAL_LOCAL을 호출한 경우, 사용되는 값은 이전 INSERT문에서 ID 열의 값에 가장 최근에 지정된 값입니다. IDENTITY_VAL_LOCAL 함수의 호출과 동일한 레벨 내에서 그러한 INSERT문이 실행되지 않은 경우 함수는 널(null) 값을 리턴합니다.

INSERT문이 실패한 후 함수 호출

ID 열이 있는 표의 INSERT문의 실행이 실패한 후 호출되는 함수가 호출되는 경우 예기치 않은 결과를 리턴합니다. 값은 실패한 INSERT 이전에 호출되었

다면 함수에서 리턴되었을 값 또는 INSERT가 성공했다면 지정되었을 값입니다. 리턴되는 실제 값은 실패 지점에 따라 다르기 때문에 예측할 수 없습니다.

커서의 SELECT문 내에서 함수 호출

IDENTITY_VAL_LOCAL 함수의 결과는 결정적인 것이 아니기 때문에, 커서의 SELECT문 내에서 IDENTITY_VAL_LOCAL 함수를 호출한 결과는 각 FETCH문마다 다릅니다.

insert 트리거의 트리거 조건 내에서 함수 호출

insert 트리거의 조건 내에서 IDENTITY_VAL_LOCAL 함수를 호출한 결과는 널값입니다.

insert 트리거의 트리거 조건 내에서 함수 호출

한 테이블에 대해 여러 사전 또는 사후 insert 트리거가 존재할 수 있습니다. 그러한 경우, 각 트리거는 별도로 처리되며 트리거된 조치 내에서 실행된 SQL 문이 생성한 ID 값은 IDENTITY_VAL_LOCAL 함수를 사용하여 다른 트리거된 조치에서 사용할 수 없습니다. 트리거된 여러 조치가 개념적으로 동일한 레벨에서 정의된 경우에도 그렇습니다.

insert 트리거 이전에 트리거된 조치에서 IDENTITY_VAL_LOCAL 함수를 사용하지 마십시오. 사전 insert 트리거의 트리거 조건 내에서 IDENTITY_VAL_LOCAL 함수를 호출하는 결과는 널값입니다. 트리거가 정의된 표의 ID 열 값은 사전 insert 트리거의 트리거 조건 내에서 IDENTITY_VAL_LOCAL 함수를 호출하여 취득할 수 없습니다. 그러나, ID 열의 트리거 전이 변수를 참조하여 트리거된 조치에서 ID 열 값을 취득할 수 있습니다.

after insert 트리거의 트리거된 조치의 IDENTITY_VAL_LOCAL 함수를 호출한 결과는 ID 열이 들어 있는 표에 대해 트리거된 동일한 조치에 호출된 가장 최신 INSERT문에 지정된 표의 ID 열에 지정된 값입니다. ID 열이 들어 있는 표의 INSERT문이 IDENTITY_VAL_LOCAL 함수를 호출하기 전에 동일한 트리거 조치 내에서 실행되지 않은 경우, 함수는 널 값을 리턴합니다.

트리거된 조치와 함께 INSERT를 수행한 후 함수 호출

트리거를 활성화하는 INSERT 다음에 함수를 호출한 결과는 ID 열에 실제로 지정된 값입니다(즉, 후속 SELECT문에 리턴되었을 값). 이 값은 반드시 INSERT문이 제공하는 값이거나 데이터베이스 관리자에 의해 생성된 값일 필요는 없습니다. 할당된 값은 ID 열과 연관된 트리거 전이 변수에 대한 before insert 트리거의 트리거된 조치 내에 있는 SET 전이 변수 명령문에 지정된 값일 수 있습니다.

예

- EMPLOYEE 표의 ID 열에 지정된 값에 변수 IVAR을 설정하십시오. VALUES문의 함수로부터 리턴된 값은 1입니다.

```
CREATE TABLE EMPLOYEE
  (EMPNO INTEGER GENERATED ALWAYS AS IDENTITY,
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPT SMALLINT)

INSERT INTO EMPLOYEE
  (NAME, SALARY, DEPTNO)
  VALUES('Rupert', 989.99, 50)

VALUES IDENTITY_VAL_LOCAL() INTO  :IVAR
```

- 두 표 T1 및 T2에 ID 열 C1이 있는 경우, 데이터베이스 관리자는 표 T1의 C1 열에 대해 값 1, 2, 3,...을 생성하고 표 T2의 C1 열에 대해 값 10, 11, 12,...를 생성합니다.

```
CREATE TABLE T1
  (C1 SMALLINT GENERATED ALWAYS AS IDENTITY,
   C2 SMALLINT)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY ( START WITH 10 ) ,
   C2 SMALLINT)

INSERT INTO T1 ( C2 ) VALUES(5)

INSERT INTO T1 ( C2 ) VALUES(5)

SELECT  * FROM T1
```

C1	C2
1	5
2	5

```
VALUES IDENTITY_VAL_LOCAL() INTO  :IVAR
```

이 때, IDENTITY_VAL_LOCAL 함수는 IVAR에 값 2를 리턴합니다. 다음 INSERT문은 한 행을 T2에 삽입합니다. 여기서 열 C2는 IDENTITY_VAL_LOCAL 함수로부터 2 값을 가져옵니다.

```
INSERT INTO T2 ( C2 ) VALUES( IDENTITY_VAL_LOCAL() )

SELECT  * FROM T2
  WHERE C1 = DECIMAL( IDENTITY_VAL_LOCAL(), 15, 0)
```

C1	C2
10	2

IDENTITY_VAL_LOCAL 함수를 이 INSERT 다음에 호출하면 결과 값이 10이 됩니다. 이 값은 데이터베이스 관리자에서 T2의 열 C1에 대해 생성한 값입니다. 다른 한 행이 T2에 삽입되는 경우, 다음 INSERT문의 경우, 데이터베이스 관리자는 ID 열 C1에 값 13을 지정하고 C2에 IDENTITY_VAL_LOCAL의 값 10을 제공합니다. 따라서, C2에는 T2에 삽입된 마지막 ID 값이 주어집니다.

```
INSERT INTO T2 ( C2, C1 ) VALUES( IDENTITY_VAL_LOCAL(), 13 )

SELECT  * FROM T2
  WHERE C1 = DECIMAL( IDENTITY_VAL_LOCAL(), 15, 0)
```

C1	C2
13	10

- IDENTITY_VAL_LOCAL 함수는 INSERT문에서도 호출할 수 있습니다. 이 명령문은 IDENTITY_VAL_LOCAL 함수를 호출하고 ID 열에 새 값이 지정되도록 합니다. 리턴되는 다음 값은 INSERT문이 완료된 후 IDENTITY_VAL_LOCAL 함수가 호출될 때 설정됩니다. 예를 들어, 다음 표 정의가 있는 경우,

  ```
  CREATE TABLE T3
    (C1 SMALLINT GENERATED BY DEFAULT AS IDENTITY,
     C2 SMALLINT)
  ```

  다음 INSERT문에서 C2 열에 대해 값 25를 지정하면, 데이터베이스 관리자는 C1 ID 열의 값 1을 생성합니다. 그러면 IDENTITY_VAL_LOCAL 함수가 다음에 호출될 때 리턴되는 값으로 1이 설정됩니다.

  ```
  INSERT INTO  T3 ( C2 ) VALUES( 25 )
  ```

  다음 INSERT문에서, IDENTITY_VAL_LOCAL 함수가 호출되어 2열의 값을 제공합니다. 1 값(첫 번째 행의 C1 열에 지정된 ID 값)은 C2열에 지정됩니다. 데이터베이스 관리자는 C1 ID 열에 2 값을 생성합니다. 그러면 IDENTITY_VAL_LOCAL 함수가 다음에 호출될 때 리턴되는 값으로 2가 설정됩니다.

  ```
  INSERT INTO T3 ( C2 ) VALUES( IDENTITY_VAL_LOCAL() )
  ```

  다음 INSERT문에서, IDENTITY_VAL_LOCAL 함수가 다시 호출되어 C2열의 값을 제공하고, 사용자는 ID 열인 C1에 11 값을 제공합니다. 2 값(두 번째 행의 C1 열에 지정된 ID 값)은 C2열에 지정됩니다. C1에 11을 지정하면 IDENTITY_VAL_LOCAL 함수가 다음에 호출될 때 리턴되는 값으로 11이 설정됩니다.

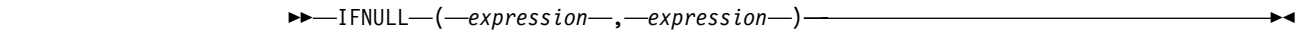
  ```
  INSERT INTO T3 ( C2, C1 ) VALUES( IDENTITY_VAL_LOCAL(), 11 )
  ```

  3 INSERT문이 처리되고 나면, 표 T3에 다음이 포함됩니다.

C1	C2
1	25
2	1
11	2

  T3의 내용은 VALUES의 표현식이 INSERT문의 열을 지정하기 전에 평가됨을 기술합니다. 즉, INSERT문의 VALUES절에서 IDENTITY_VAL_LOCAL을 호출한 경우, 사용되는 값은 이전 INSERT문에서 ID 열의 값에 가장 최근에 지정된 값입니다.

IFNULL



IFNULL 함수는 널이 아닌 첫 번째 표현식 값을 리턴합니다.

IFNULL 함수는 두 개의 인수를 갖는 COALESCE 스칼라 함수와 같습니다. 자세한 내용은 197 페이지의 『COALESCE』를 참조하십시오.

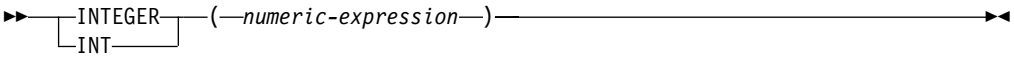
예

- EMPLOYEE 표의 모든 행에서 사원 번호(EMPNO)와 급여(SALARY)를 선택할 때 급여가 누락된 경우(즉, 널인 경우) 값 0을 리턴합니다.

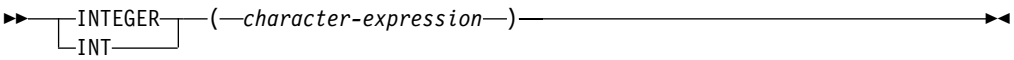
```
SELECT EMPNO, IFNULL(SALARY,0)
     FROM EMPLOYEE
```

INTEGER 또는 INT

숫자를 정수로



문자를 정수로



INTEGER 함수는 다음의 정수 표시를 리턴합니다.

- 수
- 소수의 문자 스트링 표시
- 정수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 정수 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

숫자를 정수로

numeric-expression

내장된 숫자 자료 유형의 숫자 값을 리턴하는 표현식입니다.

인수가 *numeric-expression*인 경우 결과는 인수가 큰 정수 열 또는 변수에 지정된 경우 나온 수와같습니다. 인수의 전체 부분이 정수 범위 내에 있지 않으면 오류가 발생합니다. 인수의 분수 부분은 절단됩니다.

문자를 정수로

character-expression

문자 스트링 값을 리턴하는 표현식

표현식은 정수의 문자 스트링 표현을 리턴합니다. 표현식은 CLOB가 아니어야 합니다.

결과는 CAST(*character-expression* AS INTEGER)의 결과와 같습니다. 선행 및 후미 공백은 제거되고 결과 스트링은 부동 소수점, 정수 또는 소수 상수를 형식화하기 위한 규칙을 따릅니다. 인수의 전체 부분이 정수 범위 내에 있지 않으면 오류가 발생합니다. 인수의 분수 부분은 절단됩니다.

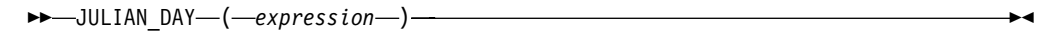
함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- EMPLOYEE 표를 사용하여 교육 레벨(EDLEVEL)로 나눈 급여(SALARY) 리스트를 선택합니다. 연산의 소수 부분은 절단합니다. 리스트에는 또한 연산에 사용된 값과 사원 번호(EMPNO)를 포함시켜야 합니다.

```
SELECT INTEGER(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
    FROM EMPLOYEE
```

JULIAN_DAY



JULIAN_DAY 함수는 기원전 4712년 1월 1일(율리우스력 시작 날짜)부터 인수에 지정된 날짜까지의 날 수를 나타내는 정수 값을 리턴합니다.

인수는 내장 자료 유형 날짜, 시간소인, 날짜나 시간소인의 유효 문자 스트링 표현 중 한 가지 값을 리턴하는 표현식이어야 합니다. 문자 스트링 자료 유형을 가진 인수는 CLOB여서는 안됩니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 결과는 널일 수 있습니다. 인수가 널이면 결과는 널 값입니다.

예

- EMPLOYEE 샘플 표를 사용하여 정수 호스트 변수 JDAY를 율리우스력 날짜를 사용하여 Christine Haas(EMPNO = '000010')의 입사일(HIREDATE = '1965-01-01')로 설정합니다.

```
SELECT JULIAN_DAY(HIREDATE)
  INTO :JDAY
                FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

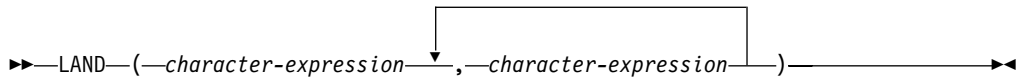
  결과는 JDAY가 2438762로 설정되었습니다.

- 정수 호스트 변수 JDAY를 율리우스력 날짜 1998년 1월 1일로 설정하십시오.

```
SELECT JULIAN_DAY('1998-01-01')
  INTO :JDAY
  FROM SYSIBM.SYSDUMMY1
```

  결과는 JDAY가 2450815로 설정되었습니다.

LAND



LAND 함수는 인수 스트링의 논리 'AND'인 스트링을 리턴합니다. 함수는 처음 인수 스트링을 가지면, 다음 스트링과 AND 비교를 수행하고, 이전 결과를 사용하여 각각의 다음 인수와 AND 비교 수행을 계속합니다. 인수가 이전 결과보다 작으면, 공백으로 채워집니다.

인수는 문자 스트링이어야 하나 LOB가 될 수 없습니다. 인수는 혼합 문자 스트링 또는 그래픽 스트링이 될 수 없습니다.

인수는 필요한 경우 결과 속성으로 변환됩니다. 결과 속성은 다음과 같이 판별됩니다.

- 모든 인수가 고정 길이 스트링인 경우 결과는 길이 *n*인 고정 길이 스트링이며, 여기서 *n*은 가장 긴 인수의 길이입니다.
- 모든 인수가 가변 길이 스트링인 경우 결과는 길이 속성 *n*인 가변 길이 스트링이며, 여기서 *n*은 가장 긴 속성을 갖는 인수의 길이 속성입니다. 결과의 실제 길이는 *m*이며 여기서, *m*은 가장 긴 인수의 실제 길이입니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며, 인수가 널인 경우 결과도 널 값입니다.

결과의 CCSID는 65535입니다.

예

- 호스트 변수 L1은 값 X'A1B1'을 갖는 CHARACTER(2) 호스트 변수, 호스트 변수 L2는 값 X'F0F040'을 갖는 CHARACTER (3) 호스트 변수, 호스트 변수 L3은 값 X'A1B10040'을 갖는 CHARACTER(4) 호스트 변수로 가정합니다.

  ```
  SELECT LAND(:L1,:L2,:L3)
    FROM SYSIBM.SYSDUMMY1
  ```

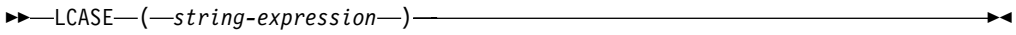
  값 X'A0B00000'을 리턴합니다.

- 이와 같이,

  ```
  SELECT LAND(:L3,:L2,:L1)
    FROM SYSIBM.SYSDUMMY1
  ```

  값 X'A0B00040'을 리턴합니다. 이때 짧은 인수는 공백(X'40')으로 채워져 논리 AND는 처음 예와 달라집니다.

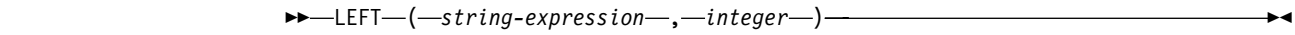
LCASE



LCASE 함수는 인수의 CCSID에 따라 모든 문자를 소문자로 변환된 스트링을 리턴합니다.

LCASE 함수는 LOWER 함수와 같습니다. 자세한 내용은 260 페이지의 『LOWER』을 참조하십시오.

LEFT



LEFT 함수는 가장 왼쪽 *integer*바이트의 *string-expression*을 나타냅니다.

*string-expression*이 문자 스트링인 경우결과는 문자 스트링이고, 각 문자는 1바이트입니다. *string-expression*이 그래픽 스트링인 경우결과는 그래픽 스트링이고, 각 문자는 DBCS 또는 UCS-2 문자입니다. *string-expression*이 2진 스트링인 경우 결과는 2진 스트링이고, 각 문자는 1바이트입니다.

string-expression

결과가 나온 스트링을 지정하는 표현식. *String-expression*의 값은 내장 자료 유형과 문자 스트링, 그래픽 스트링 또는 2진 스트링이어야 합니다.

*string-expression*의 서브스트링은 *string-expression*의 0 바이트 이상의 연속된 바이트입니다. *string-expression*이 그래픽 스트링인 경우 문자는 DBCS 또는 UCS-2 문자입니다. *string-expression*이 문자 스트링인 경우, 결과는 문자 스트링 또는 2진 스트링이고, 각 문자는 1바이트입니다.[33]

integer

결과의 길이를 지정하는 표현식을 나타냅니다. *integer*는 0 이상이고 *n* 이하여야 합니다. 여기서 *n*은 *string-expression*의 길이 속성입니다. 그러나, 정수 상수 0이어서는 안됩니다.

지정된 *string-expression*의 서브스트링이 항상 존재하도록 *string-expression*의 오른쪽이 필요한 수의 공백 문자(또는 2진 스트링의 경우 16진수 0)로 채워집니다.

함수의 결과는 *string-expression*의 길이 속성과 동일한 길이 속성 및 *string-expression*의 자료 유형에 따른 자료 유형을 가지는 가변 길이 스트링입니다.

- *string-expression*이 GRAPHIC 또는 VARGRAPHIC인 경우 VARGRAPHIC
- *string-expression*이 CHAR 또는 VARCHAR인 경우 VARCHAR
- *string-expression*이 DBCLOB인 경우 DBCLOB
- *string-expression*이 CLOB인 경우 CLOB
- *string-expression*이 BLOB인 경우 BLOB

*integer*가 정수이고 인수가 BLOB, CLOB 또는 DBCLOB가 아닌 경우, 함수의 결과는 고정 길이 스트링입니다.

결과의 실제 길이는 *integer*입니다.

---

33. LEFT 함수는 혼합 자료 스트링을 허용합니다. 그러나, LEFT가 엄격한 바이트 계수에 의해 연산되므로 결과는 올바르게 형식화된 혼합 자료 문자 스트링이 아닐 수 있습니다.

어떤 인수가 널이 될 수 있다면 결과도 널이 될 수 있으며, 어떤 인수가 널이면 결과도 널입니다.

결과의 CCSID는 *string-expression*의 CCSID와 같습니다

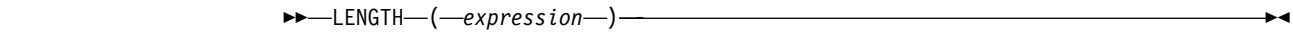
예

- 호스트 변수 NAME(VARCHAR(50))이 'KATIE AUSTIN' 값, 호스트 변수 FIRSTNAME_LEN(int)이 5 값을 갖는다고 가정합니다.

```
SELECT LEFT(:NAME, :FIRSTNAME_LEN)
  FROM SYSIBM.SYSDUMMY1
```

  값 'KATIE'을 리턴합니다.

LENGTH



LENGTH 함수는 값의 길이를 리턴합니다. 유사한 함수에 대하여 192 페이지의 『CHARACTER_LENGTH』를 참조하십시오.

인수는 모든 내장 자료 유형의 값을 리턴하는 표현식입니다.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과는 인수의 길이입니다. 스트링 길이는 후미 공백을 포함합니다. 가변 길이 스트링의 길이는 길이 속성이 아닌 실제 길이입니다.

그래픽 스트링의 길이는 2바이트 문자의 문자 수입니다(바이트 수를 2로 나눈 값). 모든 기타 값의 길이는 값을 표시하기 위해 사용된 바이트 수입니다.

수:

- 작은 정수인 경우 2
- 큰 정수인 경우 4
- 큰 정수의 경우 8
- 정밀도 *p*인 존 소수인 경우 *p*
- 정밀도 *p*인 팩 소수인 경우 (*p*/2)+1의 정수 부분
- 단정밀도 부동인 경우 4
- 배정밀도 부동인 경우 8
- 행 ID의 경우 26

문자 스트링:

- 스트링 길이

그래픽 스트링:

- 스트링에서 DBCS 또는 UCS-2 문자 수

Datetime 값:

- 시간인 경우 3
- 날짜인 경우 4
- 시간소인인 경우 10

DataLink 값:

- DataLink 값을 저장하기 위해 사용된 실제 바이트 수(DataLink가 FILE LINK CONTROL 및 READ PERMISSION DB인 경우 19를 더합니다).

예

- 호스트 변수 ADDRESS가 값 '895 Don Mills Road'를 갖는 길이 문자 스트링이라고 가정합니다.

```
SELECT LENGTH(:ADDRESS)
  FROM SYSIBM.SYSDUMMY1
```

  값 18을 리턴합니다.

- PRSTDATE가 DATE 유형의 열이라고 가정합니다.

```
SELECT LENGTH(PRSTDATE)
     FROM PROJECT
```

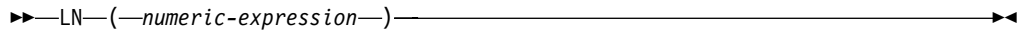
  값 4를 리턴합니다.

- PRSTDATE가 DATE 유형의 열이라고 가정합니다.

```
SELECT LENGTH(CHAR(PRSTDATE, EUR))
     FROM PROJECT
```

  값 10을 리턴합니다.

LN



LN 함수는 수의 자연 대수 값을 리턴합니다. LN 및 EXP 함수는 역산입니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다. 인수 값은 0보다 커야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 NATLOG를 값 31.62인 DECIMAL(4,2) 호스트 변수라고 가정합니다.

```
SELECT LN(:NATLOG)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 3.45를 리턴합니다.

LNOT

LNOT 함수는 인수 스트링의 논리 NOT인 스트링을 리턴합니다.

인수는 문자 스트링이어야 하지만 LOB이 될 수 없습니다. 인수는 MIXED 문자 스트링 또는 그래픽 스트링이 될 수 없습니다.

결과의 자료 유형 및 길이 속성은 인수 값의 자료 유형 및 길이 속성과 같습니다. 인수가 가변 길이 스트링인 경우 결과의 실제 길이는 인수 값의 실제 길이와 같습니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과의 CCSID는 65535입니다.

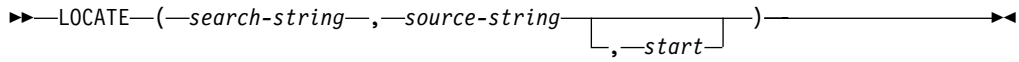
예

- 호스트 변수 L1를 값 X'F0F0'인 CHARACTER(2) 호스트 변수라고 가정합니다.

```
SELECT LNOT(:L1)
  FROM SYSIBM.SYSDUMMY1
```

  값 X'0F0F'을 리턴합니다.

LOCATE



LOCATE 함수는 다른 스트링(*source-string*) 내의 첫 번째 발생 스트링(*search-string*)의 시작 위치를 리턴합니다. *search-string*을 찾을 수 없고 인수도 널인 경우 결과는 0입니다. *search-string*을 찾은 경우 결과는 1에서 *source-string*의 실제 길이 사이의 수입니다. 선택적 *시작*이 지정된 경우, 검색이 시작될 *source-string*의 문자 위치를 나타냅니다.

search-string
: 탐색될 스트링을 지정하는 표현식. *search-string*은 문자 스트링, 그래픽 스트링 또는 2진 스트링 표현식이 될 수 있습니다. *source-string*과 호환될 수 있어야 합니다.

source-string
: 탐색이 일어날 소스 스트링을 지정하는 표현식. *source-string*은 문자 스트링, 그래픽 스트링 또는 2진 스트링 표현식이 될 수 있습니다.

start
: 탐색이 시작되는 *source-string*의 위치를 지정하는 표현식. 양의 정수이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며, 인수가 널인 경우 결과도 널값입니다.

*start*가 지정된 경우 함수는 다음과 같습니다.

```
POSSTR( SUBSTR(source-string,start) , search-string )
```

*start*가 지정되지 않은 경우 함수는 다음과 같습니다.

```
POSSTR( source-string , search-string )
```

자세한 내용은 278 페이지의 『POSITION 또는 POSSTR』을 참조하십시오.

*search-string*의 CCSID가 *source-string*의 CCSID와 다른 경우 *source-string*의 CCSID로 변환됩니다.

예

- 이러한 단어가 들어 있는 IN_TRAY 표의 모든 항목에 대한 NOTE_TEXT 열 내의 단어 'GOOD'의 시작 위치 뿐만 아니라 RECEIVED 및 SUBJECT 열을 선택합니다.

```
SELECT RECEIVED, SUBJECT, LOCATE('GOOD', NOTE_TEXT)
  FROM IN_TRAY
  WHERE LOCATE('GOOD', NOTE_TEXT) <> 0
```

LOG10

LOG10 함수는 수의 공통 대수(기준 10)를 리턴합니다. LOG10 및 ANTILOG 함수는 역산입니다.

어떤 내장 숫자 자료 유형도 인수가 될 수 있습니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

LOG는 LOG10의 동의어입니다. 단지 이전 DB2 릴리스와 호환성을 위해 지원됩니다. LOG10은 일부 데이터베이스 관리자 및 어플리케이션이 대수 대신에 자연 대수로 LOG를 수행하기 때문에 LOG 대신 사용되어야 합니다.

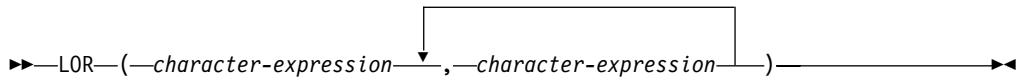
예

- 호스트 변수 L를 값 31.62인 DECIMAL(4,2) 호스트 변수라고 가정합니다.

```
SELECT LOG10(:L)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 1.49를 리턴합니다.

LOR



LOR 함수는 인수 스트링의 논리 OR인 스트링을 리턴합니다. 이 함수가 첫 번째 인수 스트링을 가지면, 다음 스트링과 OR 비교를 수행하고, 이전 결과를 사용하여 각각의 다음 인수와 OR 비교 수행을 계속합니다. 인수가 이전 결과보다 작으면, 공백으로 채워집니다.

인수는 문자 스트링이어야 하나 LOB가 될 수 없습니다. 인수는 혼합 문자 스트링 또는 그래픽 스트링이 될 수 없습니다.

인수는 필요한 경우 결과 속성으로 변환됩니다. 결과 속성은 다음과 같이 판별됩니다.

- 모든 인수가 고정 길이 스트링인 경우 결과는 길이 *n*인 고정 길이 스트링이며, 여기서 *n*은 가장 긴 인수의 길이입니다.
- 모든 인수가 가변 길이 스트링인 경우 결과는 길이 속성 *n*인 가변 길이 스트링이며, 여기서 *n*은 가장 긴 속성을 갖는 인수의 길이 속성입니다. 결과의 실제 길이는 *m*이며 여기서, *m*은 가장 긴 인수의 실제 길이입니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며, 인수가 널인 경우 결과도 널 값입니다.

결과의 CCSID는 65535입니다.

예

- 호스트 변수 L1은 값 X'0101'을 갖는 CHARACTER(2) 호스트 변수, 호스트 변수 L2는 값 X'F0F000'을 갖는 CHARACTER(3) 호스트 변수, 호스트 변수 L3은 값 X'0000000F'를 갖는 CHARACTER(4) 호스트 변수라고 가정합니다.

```
SELECT LOR(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

  값 X'F1F1000F'를 리턴합니다.

- 이와 같이,

```
SELECT LOR(:L3,:L2,:L1)
  FROM SYSIBM.SYSDUMMY1
```

  값 X'F1F1404F'를 리턴합니다. 이때 짧은 인수는 공백(X'40')으로 채워져 논리 OR는 처음 예와 달라집니다.

LOWER

LOWER 함수는 인수의 CCSID에 따라 모든 문자를 소문자로 변환된 스트링을 리턴합니다. SBCS 및 UCS-2 그래픽 문자만이 변환됩니다. 문자 A-Z는 a-z로 변환되고, 발음 구별 마크가 있는 문자는 해당 소문자(있는 경우)로 변환됩니다. 이 변환에 사용되는 모노캐스팅 표에 대한 설명은 iSeries Information Center의 국제화 주제의 UCS-2 레벨 1 맵핑 표 섹션을 참조하십시오.

string-expression

변환될 스트링을 지정하는 표현식. 스트링 표현식은 문자이거나 UCS-2 그래픽 스트링이어야 합니다.

함수의 결과는 동일한 자료 유형, 길이 속성, 실제 길이, 및 인수와 같은 CCSID를 갖습니다. 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있습니다. 인수가 널인 경우 결과는 널값입니다.

LCASE는 LOWER의 동의어입니다.

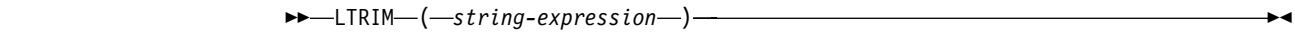
예

- 호스트 변수 NAME 값의 문자는 소문자이어야 합니다. NAME은 자료 유형 VARCHAR(30)과 값 ‘Christine Smith’를 갖습니다.

```
SELECT LOWER(:NAME)
    FROM SYSIBM.SYSDUMMY1
```

결과는 값 ’christine smith’입니다.

LTRIM



LTRIM 함수는 스트링 표현식의 시작에서 공백이나 16진 0들을 제거합니다.[34]

인수는 스트링 표현식이어야 합니다.

- 인수가 2진 스트링이면 선행 16진 0(X'00')이 제거됩니다.
- 인수가 DBCS 그래픽 스트링인 경우 선행 DBCS 공백이 제거됩니다.
- 첫 번째 인수가 UCS-2 그래픽 스트링인 경우 선행 UCS-2 공백이 제거됩니다.
- 그렇지 않으면 선행 SBCS 공백이 제거됩니다.

결과의 자료 유형은 *expression*의 자료 유형에 따라 다릅니다.

*expression*의 자료 유형	결과의 자료 유형
CHAR 또는 VARCHAR	VARCHAR
GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB

결과의 길이 유형은 *string-expression*의 길이 유형과 같습니다.결과의 실제 길이는 *string-expression*의 길이에서 제거된 바이트 수를 뺀 길이입니다. 모든 문자가 제거되면 결과는 빈 스트링입니다.

첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

결과의 CCSID는 스트링의 CCSID와 같습니다.

예

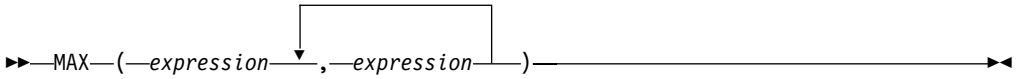
- 유형 CHAR(9)인 호스트 변수 HELLO가 값 ' Hello'를 갖는다고 가정합니다.

```
SELECT LTRIM(:HELLO)
  FROM SYSIBM.SYSDUMMY1
```

  결과는 ' Hello'이 됩니다.

34. LTRIM 함수는 STRIP(expression,LEADING)과 같은 결과를 리턴합니다.

MAX



MAX 스칼라 함수는 값 집합에 최대값을 리턴합니다.

인수는 호환될 수 있어야 합니다. 문자 스트링 인수는 datetime 값으로 호환될 수 있으나, 그래픽 스트링과 호환될 수 없습니다. 인수는 DataLink 값이 될 수 없습니다.

함수의 결과는 가장 큰 인수 값입니다. 결과는 최소한 하나의 인수가 널이 될 수 있으면, 결과도 널이 될 수 있으며, 인수 중의 하나가 널인 경우 결과도 널값입니다. 선택된 인수는 필요한 경우 결과 속성으로 변환됩니다. 결과 속성은 다음과 같이 판별됩니다.

- 인수에 최소한 하나의 날짜가 포함되고, 나머지 인수가 날짜 또는 날짜의 유효한 스트링 표시인 경우 결과는 날짜입니다. 인수에 최소한 하나의 시간이 포함되고, 나머지 인수가 시간 또는 시간의 유효한 스트링 표시인 경우 결과는 시간입니다. 인수에 최소한 하나의 시간소인이 포함되고, 나머지 인수가 시간소인 또는 시간소인의 유효한 스트링 표시인 경우 결과는 시간소인입니다.
- 인수가 스트링인 경우 결과의 CCSID는 인수가 연결된 경우 결과로 나오는 CCSID입니다. 97 페이지의 『스트링 결합 조작을 위한 변환 규칙』을 참조하십시오.
- 모든 인수가 고정 길이 스트링인 경우 결과는 길이 *n*인 고정 길이 스트링이며, 여기서 *n*은 가장 긴 인수의 길이입니다.
- 모든 인수가 가변 길이 스트링인 경우 결과는 길이 속성 *n*인 가변 길이 스트링이며, 여기서 *n*은 가장 긴 속성을 갖는 인수의 길이 속성입니다. 결과의 실제 길이는 *m*이며 여기서, *m*은 가장 긴 인수의 실제 길이입니다.
- 인수가 수인 경우 결과 자료 유형은 추가된 인수와 같습니다. 소수 결과는 다음과 같습니다.
  - 스케일은 *s*이며 여기서, *s*는 가장 큰 스케일을 갖는 인수의 스케일입니다.
  - 정밀도는 최소 31과 *s*+*n*이며 여기서, *n*는 정밀도와 스케일 간의 차이가 가장 큰 인수의 자릿수입니다.
  - 가장 큰 인수의 정수 부분을 표시하는 데 필요한 수는 31-*s*보다 커서는 안됩니다.

명령문이 실행되고 SBCS, UCS-2 또는 혼합 자료가 포함될 때 *HEX가 아닌 정렬 순서가 유효하면 스트링의 가중치가 실제 값 대신에 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

예

- 호스트 변수 M1은 값 5.5인 DECIMAL(2,1) 호스트 변수, 호스트 변수 M2는 값 4.5인 DECIMAL(3,1) 호스트 변수, 호스트 변수 M3은 값 6.5인 DECIMAL(3,2) 호스트 변수라고 가정합니다.

```
SELECT MAX(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  값 6.25를 리턴합니다.

- 호스트 변수 M1은 값 'AA'인 CHARACTER(2) 호스트 변수이고, 호스트 변수 M2는 값 'AA '인 CHARACTER(3) 호스트 변수이고, 호스트 변수 M3은 값 'AA A'인 CHARACTER(4) 호스트 변수라고 가정합니다.

```
SELECT MAX(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  값 'AA A'를 리턴합니다.

MICROSECOND

MICROSECOND 함수는 값의 마이크로초 부분을 리턴합니다.

인수는 내장 자료 유형 시간소인, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 시간소인의 유효 문자 스트링 표현이어야 합니다. 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 시간소인이거나 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 마이크로초 부분이며, 0 - 999999 사이의 정수입니다.
- 인수가 기간인 경우:

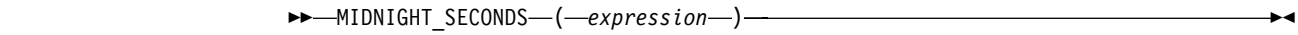
  결과는 값의 마이크로초 부분이며 -999999 - 999999 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- 표 TABLEA가 유형 TIMESTAMP인 두 개의 열, TS1 및 TS2를 포함한다고 가정합니다. TS1의 마이크로초 부분이 0이 아니고 TS1과 TS2의 초 부분이 같은 모든 행을 선택합니다.

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0 AND SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS



MIDNIGHT_SECONDS 함수는 자정에서 인수에 지정된 시간 값 사이의 초 수를 나타내는 0 - 86 400 사이의 정수 값을 리턴합니다.

인수는 내장 자료 유형 시간, 시간소인 또는 시간이나 시간소인의 유효 문자 스트링 표현 중 한 가지 값을 리턴하는 표현식이어야 합니다. 문자 스트링 자료 유형을 가진 인수는 CLOB여서는 안됩니다. 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 결과는 널일 수 있습니다. 인수가 널이면 결과는 널 값입니다.

예

- 자정에서 00:01:00 사이 및 자정에서 13:10:10 사이의 초 수를 구합니다. 호스트 변수 XTIME1에 '00:01:00' 값, XTIME2에 '13:10:10' 값이 들어 있는 것으로 가정합니다.

```
SELECT MIDNIGHT_SECONDS(:XTIME1), MIDNIGHT_SECONDS(:XTIME2)
  FROM SYSIBM.SYSDUMMY1
```

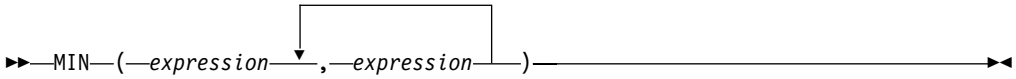
  이 예에서는 60과 47410을 리턴합니다. 1분은 60초이고 1시간은 3600초이므로 00:01:00은 자정 이후 60초((60 * 1) + 0)이고 13:10:10은 47410초((3600 * 13) + (60 * 10) + 10)입니다.

- 자정에서 24:00:00 사이 및 자정에서 00:00:00 사이의 초 수를 구하십시오.

```
SELECT MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00')
  FROM SYSIBM.SYSDUMMY1
```

  이 예에서는 86400과 0을 리턴합니다. 이 두 값은 시간상으로는 같은 시점을 나타내지만 서로 다른 값이 리턴됩니다.

MIN



MIN 스칼라 함수는 값 집합에 최소 값을 리턴합니다.

인수는 호환될 수 있어야 합니다. 문자 스트링 인수는 datetime 값으로 호환될 수 있으나, 그래픽 스트링과 호환될 수 없습니다. 인수는 DataLink 값이 될 수 없습니다.

함수의 결과는 가장 작은 인수 값입니다. 결과는 최소한 하나의 인수가 널이 될 수 있으면, 결과도 널이 될 수 있으며, 인수 중의 하나가 널인 경우 결과도 널값입니다. 선택된 인수는 필요한 경우 결과 속성으로 변환됩니다. 결과 속성은 다음과 같이 판별됩니다.

- 인수에 최소한 하나의 날짜가 포함되고, 나머지 인수가 날짜 또는 날짜의 유효한 스트링 표시인 경우 결과는 날짜입니다. 인수에 최소한 하나의 시간이 포함되고, 나머지 인수가 시간 또는 시간의 유효한 스트링 표시인 경우 결과는 시간입니다. 인수에 최소한 하나의 시간소인이 포함되고, 나머지 인수가 시간소인 또는 시간소인의 유효한 스트링 표시인 경우 결과는 시간소인입니다.
- 인수가 스트링인 경우 결과의 CCSID는 인수가 연결된 경우 결과로 나오는 CCSID입니다. 97 페이지의 『스트링 결합 조작을 위한 변환 규칙』을 참조하십시오.
- 모든 인수가 고정 길이 스트링인 경우 결과는 길이 *n*인 고정 길이 스트링이며, 여기서 *n*은 가장 긴 인수의 길이입니다.
- 모든 인수가 가변 길이 스트링인 경우 결과는 길이 속성 *n*인 가변 길이 스트링이며, 여기서 *n*은 가장 긴 속성을 갖는 인수의 길이 속성입니다. 결과의 실제 길이는 *m*이며, 여기서 *m*은 가장 작은 인수의 실제 길이입니다.
- 인수가 수인 경우 결과의 자료 유형은 인수가 추가된 경우 결과로 나오는 자료 유형입니다. 소수 결과는 다음과 같습니다.
  - 스케일은 *s*이며 여기서, *s*는 가장 큰 스케일을 갖는 인수의 스케일입니다.
  - 정밀도는 최소 31과 *s*+*n*이며 여기서, *n*는 정밀도와 스케일 간의 차이가 가장 큰 인수의 자릿수입니다.
  - 가장 큰 인수의 정수 부분을 표시하는 데 필요한 수는 31-*s*보다 커서는 안됩니다.

명령문이 실행되고 SBCS, UCS-2 또는 혼합 자료가 포함될 때 *HEX가 아닌 정렬 순서가 유효하면 스트링의 가중치가 실제 값 대신에 비교됩니다. 가중치는 정렬 순서에 따라 달라집니다.

예

- 호스트 변수 M1은 값 5.5인 DECIMAL(2,1) 호스트 변수, 호스트 변수 M2는 값 4.5인 DECIMAL(3,1) 호스트 변수, 호스트 변수 M3은 값 6.5인 DECIMAL(3,2) 호스트 변수라고 가정합니다.

```
SELECT MIN(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  값 4.50을 리턴합니다.

- 호스트 변수 M1은 값 'AA'인 CHARACTER(2) 호스트 변수이고, 호스트 변수 M2는 값 'AAA '인 CHARACTER(3) 호스트 변수이고, 호스트 변수 M3은 값 'AAAA'인 CHARACTER(4) 호스트 변수라고 가정합니다.

```
SELECT MIN(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  값 'AA'를 리턴합니다.

MINUTE

MINUTE 함수는 값의 분 부분을 리턴합니다.

인수는 내장 자료 유형 시간, 시간소인, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 시간 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 시간 기간 또는 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 시간, 시간소인, 시간 또는 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 분 부분이며, 0 - 59 사이의 정수입니다.
- 인수가 소요 시간 또는 시간소인 기간인 경우:

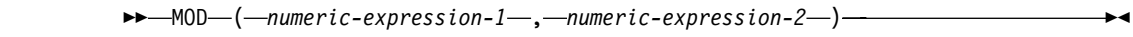
  결과는 값의 분 부분이고 -99 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- CL_SCHED 샘플 표를 사용하여 50분 미만의 클래스를 모두 선택합니다.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0 AND
      MINUTE(ENDING - STARTING) < 50
```

MOD



MOD 함수는 첫 번째 인수를 두 번째 인수로 나누고, 그 나머지를 리턴합니다.

나머지를 연산하기 위해 사용된 형식은 다음과 같습니다.

```
MOD(x,y) = x - (x/y) * y
```

여기서, x/y는 나눗셈하여 절단된 정수입니다. 결과는 첫 번째 인수가 음수인 경우에만 음수입니다.

인수는 각각 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다. *numeric-expression-2*는 0이 될 수 없습니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며, 인수가 널인 경우 결과도 널 값입니다.

결과 속성은 다음과 같이 판별됩니다.

- 양 인수가 모두 0 스케일을 갖는 큰 정수 또는 작은 정수이면 결과의 자료 유형은 큰 정수입니다.
- 양 인수가 모두 0 스케일을 정수이고 인수 중 적어도 하나가 큰 정수이면 결과의 자료 유형은 큰 정수입니다.
- 하나의 인수가 스케일 0을 갖는 정수이고 다른 인수는 소수인 경우 결과는 같은 정밀도와 소수 인수와 같은 스케일을 갖는 소수입니다.
- 모든 인수가 소수 또는 스케일을 갖는 정수인 경우 결과는 소수입니다. 결과의 정밀도는 min (p-s,p'-s') + max (s,s')이고, 결과의 스케일은 max (s,s')이며, 여기서 기호 p와 s는 첫 번째 피연산자의 정밀도와 스케일을 표시하고, 기호 p'와 s'는 두 번째 피연산자의 정밀도와 스케일을 표시합니다.
- 인수 중의 하나가 부동 소수점인 경우 결과의 자료 유형은 배정밀도 부동 소수점입니다.

  연산이 부동 소수점에서 수행되며, 필요한 경우 피연산자는 배정밀도 부동 소수점으로 먼저 변환됩니다.

  부동 소수점과 정수를 포함하는 연산은 배정밀도 부동 소수점으로 변환된 정수의 임시 복사로 수행됩니다. 부동 소수점과 소수를 포함하는 연산은 배정밀도 부동 소수점으로 변환된 소수의 임시 복사로 수행됩니다. 부동 소수점 연산의 결과는 부동 소수점 범위 내에 있어야 합니다.

예

- 호스트 변수 M1은 값 5를 갖는 정수 호스트 변수, 호스트 변수 M2은 값 2를 갖는 정수 호스트 변수로 가정합니다.

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

  값 1을 리턴합니다.

- 호스트 변수 M1은 값 5를 갖는 정수 호스트 변수이고, 호스트 변수 M2은 값 2.20을 갖는 DECIMAL(3,2) 호스트 변수라고 가정합니다.

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

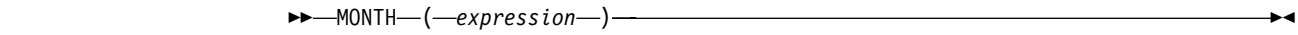
  값 0.60을 리턴합니다.

- 호스트 변수 M1은 값 5.50을 갖는 DECIMAL(4,2) 호스트 변수이고, 호스트 변수 M2은 값 2.0을 갖는 DECIMAL(4,1) 호스트 변수라고가정합니다.

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

  값 1.50을 리턴합니다.

MONTH



MONTH 함수는 값의 월 부분을 리턴합니다.

인수는 내장 자료 유형 날짜, 시간소인, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 날짜 기간 또는 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 날짜, 시간소인, 날짜 또는 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 월 부분이며, 1 - 12 사이의 정수입니다.
- 인수가 시간소인 기간 또는 시간소인 기간인 경우:

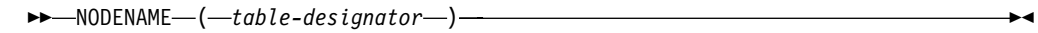
  결과는 값의 월 부분이고 -99 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- EMPLOYEE 표에서 DECEMBER에 태어난 (BIRTHDATE) 사람의 모든 행을 선택합니다.

```
SELECT * FROM EMPLOYEE    WHERE MONTH(BIRTHDATE) = 12
```

NODENAME



NODENAME 함수는 행이 위치한 관계형 데이터베이스(RDB)명을 리턴합니다. 인수가 분배되지 않는 표를 식별하면 CURRENT SERVER 특수 레지스터의 값이 리턴됩니다. 노드에 대한 자세한 정보는 DB2 Multisystem 책을 참조하십시오.

인수는 subselect의 표 지정자입니다. 표 지정자에 대한 자세한 정보는 111 페이지의 『표 지정자』를 참조하십시오.

SQL 명명에서 표 이름은 규정화될 수 있습니다. 시스템 명명에서 표 이름은 규정화될 수 없습니다.

인수가 뷰, 공통 표 표현식 또는 도출된 표를 식별하는 경우, 함수는 기본 표의 관계형 데이터베이스(RDB) 이름을 리턴합니다. 인수가 뷰, 공통 표 표현식 또는 하나 이상의 기본 표에서 도출된 표를 식별하는 경우, 인수는 뷰, 공통 표 표현식 또는 도출된 표의 외부 subselect에서 첫 번째 표의 관계형 데이터베이스 이름을 리턴합니다.

인수는 외부 subselect가 열 함수, GROUP BY절, HAVING절, UNION절 또는 DISTINCT절을 포함하는 뷰, 공통 표 표현식 또는 도출된 표를 식별하지 않습니다. subselect에 GROUP BY 또는 HAVING절이 있는 경우 NODENAME 함수는 WHERE절에 또는 열 함수의 피연산자로서만 지정될 수 있습니다. 인수가 상관명인 경우 상관명은 관련된 참조를 식별하지 않습니다.

결과의 자료 유형은 VARCHAR(18)입니다. 결과는 널(null)이 될 수 있습니다.

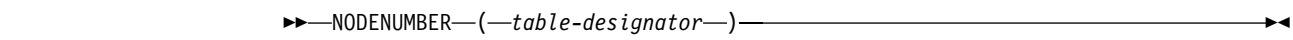
결과의 CCSID는 현재 서버의 CCSID입니다.

예

- EMPLOYEE와 DEPARTMENT 표를 결합하고, 사원 번호(EMPNO)를 선택하여 결합이 이루어진 행의 노드를 판별합니다.

```
SELECT EMPNO, NODENAME(X), NODENAME(Y)
  FROM EMPLOYEE X, DEPARTMENT Y
  WHERE X.DEPTNO=Y.DEPTNO
```

NODENUMBER



NODENUMBER 함수는 행의 노드 번호를 리턴합니다. 인수가 비분산 표를 식별하는 경우 0 값이 리턴됩니다.[35] 노드와 노드 번호에 대한 더 자세한 정보는 DB2 Multisystem 책을 참조하십시오.

인수는 subselect의 표 지정자입니다. 표 지정자에 대한 자세한 정보는 111 페이지의 『표 지정자』를 참조하십시오.

SQL 명명에서 표 이름은 규정화될 수 있습니다. 시스템 명명에서 표 이름은 규정화될 수 없습니다.

인수가 뷰, 공통 표 표현식 또는 도출된 표를 식별하는 경우, 함수는 기본 표의 노드 번호를 리턴합니다. 인수가 뷰, 공통 표 표현식 또는 하나 이상의 기본 표에서 도출된 표를 식별하는 경우, 함수는 뷰, 공통 표 표현식 또는 도출된 표의 외부 subselect에서 첫 번째 표의 노드 번호를 리턴합니다.

인수는 외부 subselect가 열 함수, GROUP BY절, HAVING절, UNION절 또는 DISTINCT절을 포함하는 뷰, 공통 표 표현식 또는 도출된 표를 식별할 수 없습니다. subselect에 GROUP BY 또는 HAVING절이 있는 경우 NODENUMBER 함수는 WHERE절에 또는 열 함수의 피연산자로 지정될 수 있습니다. 인수가 상관명인 경우 상관명은 관련된 참조를 식별하지 않습니다.

결과 자료 유형은 큰 정수입니다. 결과는 널(null)이 될 수 있습니다.

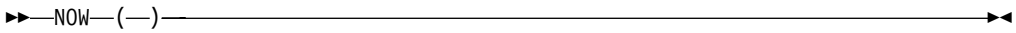
예

- EMPLOYEE 표에서 각 행에 대한 노드 번호 및 사원 이름을 판별합니다. 이것이 분산 표이면 행에 있는 노드 번호가 리턴됩니다.

```
SELECT NODENUMBER(EMPLOYEE), LASTNAME
                FROM EMPLOYEE
```

35. 인수가 둘 이상의 실제 파일 멤버에 근거한 DDS 작성 논리 파일을 식별하는 경우 NODENUMBER는 0을 리턴하지 않고 대신 실제 파일 멤버 번호를 리턴합니다.

NOW



NOW 함수는 SQL문이 현재 서버에서 실행될 때 날짜 시간 시계를 읽어서 시간소인을 리턴합니다. NOW 함수로 리턴된 값은 CURRENT TIMESTAMP 특수 레지스터로 리턴된 값과 같습니다. 함수가 하나의 SQL문에서 한번 이상 사용된 경우 또는 하나의 명령문에서 CURDATE 또는 CURTIME 스칼라 함수 또는 CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP 특수 레지스터가 사용된 경우 모든 값은 하나의 시계를 읽습니다.

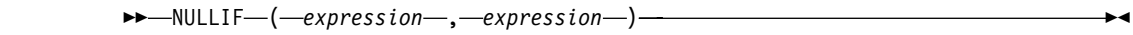
결과 자료 유형은 시간소인입니다. 결과는 널이 될 수 없습니다.

예

- 날짜 시간 시계에 따라 현재 시간소인을 리턴합니다.

```
SELECT NOW()
  FROM SYSIBM.SYSDUMMY1
```

NULLIF



NULLIF 함수는 인수비교가 같은 경우 널값을 리턴하며, 그렇지 않은 경우 첫 번째 인수의 값을 리턴합니다.

인수는 호환 자료 유형이어야 합니다. 문자 스트링 인수는 날짜 시간 값으로 호환될 수 있습니다. 한 피연산자가 고유한 유형이면 다른 피연산자는 고유한 유형이어야 합니다. 인수는 DataLink 값이 될 수 없습니다.

결과 속성은 첫 번째 인수의 속성입니다. 결과는 널(null)이 될 수 있습니다. 결과는 첫 번째 인수가 널이거나 모든 인수가 같은 경우 널입니다.

NULLIF(e1,e2)를 사용한 결과는 다음 표현식을 사용한 것과 같습니다.

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

(하나 또는 두 개의 모든 인수가 NULL이므로) e1=e2가 알 수 없는 것으로 평가될 때 CASE 표현식은 참이 아닌 것으로 간주함에 유의하십시오. 그러므로 이 상황에서 NULLIF은 첫 번째 피연산자 값 e1을 리턴합니다.

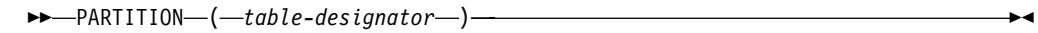
예

- 호스트 변수 PROFIT, CASH, 및 LOSSES이 각각 값 4500.00, 500.00 및 5000.00을 갖는 자료 유형 DECIMAL을 갖는다고 가정합니다.

  ```
  SELECT NULLIF (:PROFIT + :CASH, :LOSSES )
    FROM SYSIBM.SYSDUMMY1
  ```

  널값을 리턴합니다.

PARTITION



PARTITION 함수는 행의 파티션 키 값에 해싱 함수를 적용하여 구한 행의 파티션 번호를 리턴합니다. 또한 HASH 함수를 참조하십시오. 인수가 분배되지 않은 표를 식별하면 값 0이 리턴됩니다. 파티션 번호와 파티션 키에 대한 자세한 정보는 DB2 Multisystem 책을 참조하십시오.

인수는 subselect의 표 지정자입니다. 표 지정자에 대한 자세한 정보는 111 페이지의 『표 지정자』를 참조하십시오.

SQL 명명에서 표 이름은 규정화될 수 있습니다. 시스템 명명에서 표 이름은 규정화될 수 없습니다.

인수가 뷰, 공통 표 표현식 또는 도출된 표를 식별하는 경우, 함수는 기본 표의 파티션 번호를 리턴합니다. 인수가 뷰, 공통 표 표현식 또는 하나 이상의 기본 표에서 도출된 표를 식별하는 경우, 함수는 뷰, 공통 표 표현식 또는 도출된 표의 외부 subselect에서 첫 번째 표의 파티션 번호를 리턴합니다.

인수는 외부 subselect가 열 함수, GROUP BY절, HAVING절, UNION절 또는 DISTINCT절을 포함하는 뷰, 공통 표 표현식 또는 도출된 표를 식별할 수 없습니다. subselect에 GROUP BY 또는 HAVING절이 있는 경우 PARTITION 함수는 WHERE 절에 또는 열 함수의 피연산자로서만 지정될 수 있습니다. 인수가 상관명인 경우 상관명은 관련된 참조를 식별하지 않습니다.

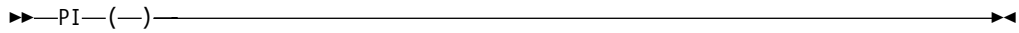
함수의 결과는 0 - 1023 사이의 값인 큰 정수입니다. 결과는 널(null)이 될 수 있습니다.

예

- EMPLOYEE 표에서 파티션 번호가 100인 모든 행에 대한 사원 번호(EMPNO)를 선택합니다.

```
ELECT EMPNO
     FROM EMPLOYEE
  WHERE PARTITION(EMPLOYEE) = 100
```

PI



PI 값 3.141592653589793을 리턴합니다. 인수는 없습니다.

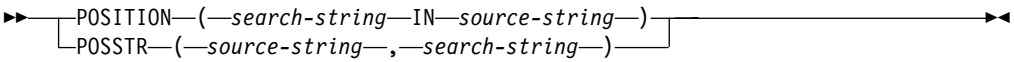
함수 결과는 배정밀도 부동 소수점입니다. 결과는 널이 될 수 없습니다.

예

- 다음은 지름이 10인 원의 원주를 리턴합니다.

```
SELECT PI()*10
  FROM SYSIBM.SYSDUMMY1
```

POSITION 또는 POSSTR



POSITION 및 POSSTR 함수는 다른 스트링(*source string*) 내의 첫 번째 발생 스트링(*search-string*)의 시작 위치를 리턴합니다. *search-string*을 찾을 수 없고 인수도 널인 경우 결과는 0입니다. *search-string*을 찾은 경우 결과는 1에서 *source-string*의 실제 길이 사이의 수입니다. 관련된 함수는 257 페이지의 『LOCATE』를 참조하십시오.

source-string
탐색이 일어날 소스 스트링을 지정하는 표현식. *source-string*은 2진 스트링, 문자 스트링 또는 그래픽 스트링 표현식일 수 있습니다.

search-string
탐색될 스트링을 지정하는 표현식. *search-string*은 2진 스트링, 문자 스트링 또는 그래픽 스트링 표현식일 수 있습니다. *source-string*과 호환될 수 있어야 합니다.

함수의 결과는 큰 정수입니다. 인수 중에 하나가 널이 될 수 있는 경우 결과는 널이 될 수 있습니다. 인수 중에 하나가 널인 경우 결과는 널값입니다.

*search-string*의 CCSID가 *source-string*의 CCSID와 다른 경우 *source-string*의 CCSID로 변환됩니다.

POSITION 함수는 문자를 기초로 연산합니다. POSSTR 함수는 엄격한 바이트 계수에 따라 연산됩니다. *search-string* 또는 *source-string*이 혼합 자료를 포함하는 경우 POSITION을 POSSTR 대신 사용할 것을 권장합니다. POSSTR은 바이트 수에 따라 연산되므로 *search-string* 또는 *source-string*이 혼합 자료를 포함하는 경우 *search-string*은 SI/SO 문자가 정확히 같은 위치의 *source-string*에서 발견되는 경우에만 찾을 수 있습니다. POSITION은 문자 스트링에서 연산되므로 SI/SO 문자가 정확히 같은 위치에 있을 필요가 없으며, SBCS인 문자인지 DBCS인 문자인지 구별하는 것만이 중요합니다.

POSSTR이나 POSITION 함수가 포함된 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 인수에 SBCS, UCS-2 또는 혼합 자료가 들어 있는 경우 결과는 집합의 각 값에 대한 가중치를 비교하여 나옵니다. 가중치는 정렬 순서에 따라 달라집니다.

*search-string*의 길이가 0인 경우 함수에 의해 리턴된 결과는 1입니다.

- *탐색 스트링*의 길이 0인 경우 함수에 의해 리턴된 결과는 0입니다.
- 그렇지 않은 경우:

- *search-string*의 값이 *source-string* 값 내의 연속 위치에 있는 같은 길이의 서브스트링과 같은 경우 함수에 의해 리턴된 결과는 *source-string* 값 내의 서브스트링의 첫 번째 시작 위치입니다.
- 그렇지 않은 경우 함수에 의해 리턴되는 결과는 0입니다.[36]

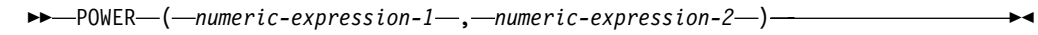
예

- 이러한 단어가 들어 있는 IN_TRAY 표의 모든 항목에 대한 NOTE_TEXT 열 내의 단어 'GOOD'의 시작 위치 뿐만 아니라 RECEIVED 및 SUBJECT 열을 선택합니다.

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD') <> 0
```

36. *search-string*이 *source-string*보다 긴 경우를 포함합니다.

POWER



POWER 함수는 첫 번째 인수에 대한 두 번째 인수의 제곱근의 결과를 리턴합니다.[37]

rkr 인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식이어야 합니다. *numeric-expression-1* 값이 0인 경우, *numeric-expression-2*는 0 이상이어야 합니다. 두 인수 모두 0인 경우, 결과는 1입니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며, 인수가 널인 경우 결과도 널값입니다.

예

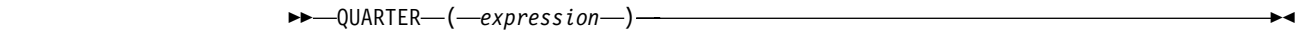
- 호스트 변수 HPOWER가 값 3인 정수라고 가정합니다.

```
SELECT POWER(2,:HPOWER)
  FROM SYSIBM.SYSDUMMY1
```

  값 8을 리턴합니다.

37. POWER 함수의 결과는 *numeric-expression-1* ** *numeric-expression-2*인 지수화 결과와 같습니다.

QUARTER



QUARTER 함수는 해당 날짜가 위치하는 1년 중 사분기를 나타내는 1 - 4 사이의 정수를 리턴합니다. 예를 들어, 1월, 2월 또는 3월의 날짜는 정수 1을 리턴합니다.

인수는 내장 자료 유형 날짜, 시간소인 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다. *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- PROJECT 표를 사용하여 호스트 변수 QUART(INTEGER)를 프로젝트 ‘PL2100’가 끝낸 (PRENDATE) 분기로 설정하십시오.

```
SELECT QUARTER(PRENDATE)
  INTO :QUART
     FROM PROJECT
  WHERE PROJNO = 'PL2100'
```

  QUART가 3으로 설정됩니다.

RADIANS

RADIANS 함수는 각도로 표시되는 인수에 대한 라디안 값을 리턴합니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다. 인수가 배정밀도 부동 소수점 숫자가 아닌 경우 함수가 처리할 수 있도록 배정밀도 부동 소수점 숫자로 변환됩니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

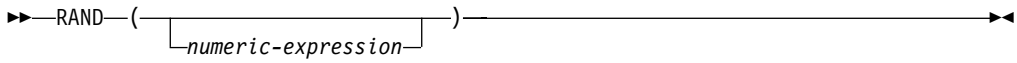
예

- 호스트 변수 HDEG가 값 180을 갖는 INTEGER라고 가정합니다. 다음 명령문을 참조하십시오.

  ```
  SELECT RADIANS(:HDEG)
    FROM SYSIBM.SYSDUMMY1
  ```

  위 명령문은 대략 3.1415926536 값을 갖는 배정밀도 부동 소수점 숫자를 리턴합니다.

RAND



RAND 함수는 0 - 1 사이의 부동 소수점 값을 리턴합니다.

표현식이 지정되면 이것은 시드 값으로 사용됩니다. 표현식은 SMALLINT 또는 INTEGER이어야 합니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 HRAND가 값 180을 갖는 INTEGER라고 가정합니다. 다음 명령문을 참조하십시오.

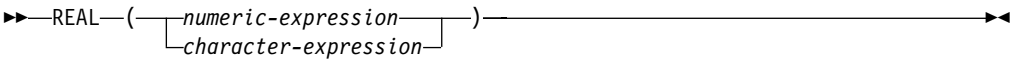
```
SELECT RAND(:HRAND)
  FROM SYSIBM.SYSDUMMY1
```

  대략 .0121398의 0 - 1 사이의 임의의 부동 소수점 값이 리턴됩니다.

- 0 - 1 이 아닌 다른 숫자 간격으로 값을 생성하려면 RAND 함수를 원하는 간격의 크기로 곱하십시오. 예를 들어 5.8731398과 같은 0 - 10 사이의 난수를 구하려는 경우 함수를 10으로 곱합니다.

```
SELECT RAND(:HRAND) * 10
  FROM SYSIBM.SYSDUMMY1
```

REAL



REAL 함수는 다음의 단정밀도 부동 소수점을 리턴합니다.

- 수
- 소수의 문자 스트링 표시
- 정수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 단정밀도 부동 소수점 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

numeric-expression

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다.

결과는 인수가 단정밀도 부동 소수점 열 또는 변수에 지정된 경우의 수와 같습니다. 인수의 숫자 부분이 단정밀도 부동 소수점 내에 있지 않으면 오류가 발생합니다.

character-expression

문자 스트링 값을 리턴하는 표현식

결과는 CAST(*character-expression* AS REAL)의 결과와 같습니다. 선행 및 후미 공백은 제거되고 결과 스트링은 부동 소수점, 정수 또는 소수 상수를 형식화하기 위한 규칙을 따릅니다. 인수의 숫자 부분이 단정밀도 부동 소수점 내에 있지 않으면 오류가 발생합니다.

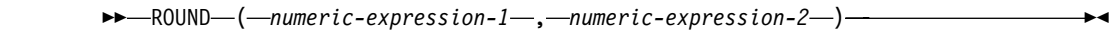
함수의 결과는 단정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- EMPLOYEE 표를 사용하여 수당이 0이 아닌 사원에 대한 수당과 급여의 비율을 찾습니다. (SALARY 및 COMM)이 포함된 열은 DECIMAL 자료 유형입니다. 범위 밖의 결과가 나올 가능성을 제거하려면 나눗셈을 부동 소수점에 실행하도록 REAL이 SALARY에 적용됩니다.

```
SELECT EMPNO, REAL(SALARY)/COMM
    FROM EMPLOYEE
  WHERE COMM > 0
```

ROUND



ROUND 함수는 소수점 왼쪽 및 오른쪽의 특정 자리수에서 반올림된 *numeric-expression-1*을 리턴합니다.

numeric-expression-1
내장된 숫자 자료 유형의 값을 리턴하는 표현식입니다.

numeric-expression-2
작은 정수나 큰 정수를 리턴하는 표현식입니다. 정수의 절대값은 *numeric-expression-2*가 음수가 아닌 경우 결과에 대한 소수점의 오른쪽으로또는 *numeric-expression-2*가 음수인 경우 소수점의 왼쪽으로 자리수를 지정합니다.

*numeric-expression-2*가 음수가 아니면 *numeric-expression-1*은 소수점의 오른쪽 *numeric-expression-2* 자리수로 반올림됩니다. 자리수 5는 다음의 높은 양수로 반올림됩니다.

*numeric-expression-2*가 음수이면 *numeric-expression-1*은 소수점의 왼쪽으로 *numeric-expression-2*+1)의 절대값 자리수로 반올림됩니다. 자리수 5는 다음의 낮은 음수로 반올림됩니다. *numeric-expression-2*의 절대값이 소수점 왼쪽의 자리수 이상이면, 결과는 0입니다.

*numeric-expression-1*이 DECIMAL 또는 NUMERIC이고 정밀도가 31보다 적을 때 정밀도가 1씩 증가하는 경우를 제외하고, 결과의 자료 유형 및 길이 속성은 첫 번째 인수의 자료 유형 및 길이 속성과 같습니다. 예를 들어, 자료 유형 DECIMAL(5,2)를 갖는 인수는 DECIMAL(6,2)의 결과를 가져옵니다. 자료 유형 DECIMAL(31,2)를 갖는 인수는 DECIMAL(31,2)의 결과를 가져옵니다.

한 인수가 널(null)일 수 있으면 결과도 널(null)일 수 있습니다. 한 인수가 널(null)이면 결과는 널값입니다.

예

- 숫자 873.726을 2, 1, 0, -1, -2, -3 및 -4 소수 자리로 반올림하여 각각 연산합니다.

```
SELECT 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)
  FROM SYSIBM.SYSDUMMY1
```

각각 다음 값을 리턴합니다.

```
0873.730   0873.700  0874.000  0870.000  0900.000  1000.000  0000.000
```

- 양수와 음수 모두 연산하십시오.

```
SELECT ROUND( 3.5, 0),
       ROUND( 3.1, 0),
       ROUND(-3.1, 0),
       ROUND(-3.5, 0)
  FROM SYSIBM.SYSDUMMY1
```

각각 다음 값을 리턴합니다.

```
04.0   03.0  -03.0  -04.0
```

ROWID

ROWID 함수는 문자 스트링이나 2진 스트링을 행 ID로 캐스팅합니다.

주: 또한 CAST 표현식은 행 ID 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

string-expression

문자 스트링 또는 2진 스트링 값을 리턴하는 표현식 *string-expression*은 CLOB가 아니어야 합니다. 스트링은 어떤 값을 포함할 수도 있지만, 유효한 ROWID 값이 리턴되도로고 하려면 이전에 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에서 생성한 ROWID 값을 포함하는 것이 좋습니다. 예를 들어, 값이 CHAR 값으로 캐스팅되었다가 다시 ROWID 값으로 캐스팅되는 ROWID를 변환하는 데 이 함수를 사용할 수 있습니다.

*string-expression*의 실제 길이가 40보다 적은 경우, 결과는 채워지지 않습니다. *string-expression*의 실제 길이가 40보다 큰 경우, 결과는 잘립니다. 공백이 아닌 문자가 잘릴 경우, 경고가 리턴됩니다.

결과의 길이 속성은 40입니다. 결과의 실제 길이는 *string-expression*의 길이입니다.

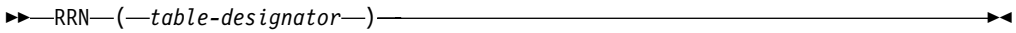
함수의 결과는 행 ID입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 표 EMPLOYEE에 ROWID 열 EMP_ROWID가 포함되며, 표에 행 ID 값 X'F0DFD230E3C0D80D81C201AA0A280100000000000203'로 지정된 행이 있는 경우, 직접 행 액세스를 사용하여 해당 행의 사원 번호를 선택하십시오.

```
ELECT EMPNO
     FROM EMPLOYEE
  WHERE EMP_ROWID = ROWID(X'F0DFD230E3C0D80D81C201AA0A280100000000000203')
```

RRN



RRN 함수는 행의 상대 레코드 번호(RRN)를 리턴합니다.

인수는 subselect의 표 지정자입니다. 표 지정자에 대한 자세한 정보는 111 페이지의 『표 지정자』를 참조하십시오.

SQL 명명에서 표 이름은 규정화될 수 있습니다. 시스템 명명에서 표 이름은 규정화될 수 없습니다.

인수가 뷰, 공통 표 표현식 또는 도출된 표를 식별하는 경우, 함수는 기본 표의 상대 레코드 번호를 리턴합니다. 인수가 뷰, 공통 표 표현식 또는 하나 이상의 기본 표에서 도출된 표를 식별하는 경우, 함수는 뷰, 공통 표 표현식 또는 도출된 표의 외부 subselect에서 첫 번째 표의 상대 레코드 번호를 리턴합니다.

인수가 분산 표를 식별하는 경우 함수는 행이 위치한 노드에서 행의 상대 레코드 번호(RRN)를 리턴합니다. 이것은 RRN이 분산 표의 각 행에 대해 고유하지 않음을 의미합니다.

인수는 외부 subselect가 열 함수, GROUP BY절, HAVING절, UNION절 또는 DISTINCT절을 포함하는 뷰, 공통 표 표현식 또는 도출된 표를 식별할 수 없습니다. RRN 함수는 subselect가 열 함수 GROUP BY절 또는 HAVING절을 포함하는 경우 SELECT절에 지정될 수 없습니다. 인수가 상관명인 경우 상관명은 관련된 참조를 식별하지 않습니다.

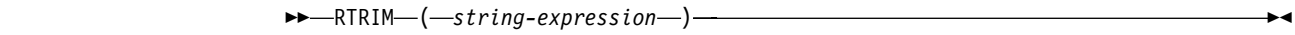
결과의 자료 유형은 정밀도 15 및 스케일 0의 십진수입니다. 결과는 널값일 수 있습니다.

예

- EMPLOYEE 표에서 부서 20에서 사원들에 대한 상대 레코드 번호(RRN)와 사원 이름을 리턴합니다.

```
SELECT RRN(EMPLOYEE), LASTNAME
    FROM EMPLOYEE
  WHERE DEPTNO = 20
```

RTRIM



RTRIM 함수는 스트링 표현식의 끝에서 공백이나 16진 0들을 제거합니다.[38]

인수는 스트링 표현식이어야 합니다.

- 인수가 2진 스트링이면 후미 16진 0(X'00')이 제거됩니다.
- 인수가 DBCS 그래픽 스트링인 경우 후미 DBCS 공백이 제거됩니다.
- 첫 번째 인수가 UCS-2 그래픽 스트링인 경우 후미 UCS-2 공백이 제거됩니다.
- 그렇지 않으면 후미 SBCS 공백이 제거됩니다.

결과의 자료 유형은 *string-expression*의 자료 유형에 따라 다릅니다.

*expression*의 자료 유형	결과의 자료 유형
CHAR 또는 VARCHAR	VARCHAR
GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB

결과의 길이 유형은 *string-expression*의 길이 유형과 같습니다. 결과의 실제 길이는 표현식의 길이에서 제거된 바이트 수를 뺀 길이입니다. 모든 문자가 제거되면 결과는 빈 스트링입니다.

첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

결과의 CCSID는 스트링의 CCSID와 같습니다.

예

- 유형 CHAR(9)인 호스트 변수 HELLO가 값 ' Hello'를 갖는다고 가정합니다.

```
SELECT RTRIM(:HELLO)
    FROM SYSIBM.SYSDUMMY1
```

  결과는 'Hello'가 됩니다.

38. RTRIM 함수는 STRIP(expression,TRAILING)과 같은 결과를 리턴합니다.

SECOND

SECOND 함수는 값의 초 부분을 리턴합니다.

인수는 내장 자료 유형 날짜, 시간소인, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 시간 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 시간 기간 또는 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 시간, 시간소인, 시간 또는 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 초 부분이며, 0 - 59 사이의 정수입니다.
- 인수가 소요 시간 또는 시간소인 기간인 경우:

  결과는 값의 초 부분이고 -99 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- 호스트 변수 TIME_DUR(DECIMAL(6,0))이 값 153045를 갖는다고 가정합니다.

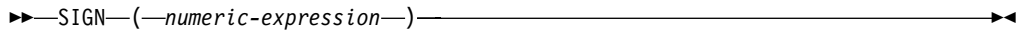
```
SELECT SECOND(:TIME_DUR)
  FROM SYSIBM.SYSDUMMY1
```

  값 45를 리턴합니다.
- 열 RECEIVED(TIMESTAMP) 1988-12-25-17.12.30.000000와 같은 내부 값이 있다고 가정합니다.

```
SELECT SECOND(RECEIVED)
  FROM IN_TRAY
```

  값 30을 리턴합니다.

SIGN



SIGN 함수는 표현식 부호의 인디케이터를 리턴합니다. 리턴되는 값은 다음과 같습니다.

-1 인수가 0보다 작은 경우

0 인수가 0인 경우

1 인수가 0보다 큰 경우

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다.

인수가 DECIMAL 또는 NUMERIC이고 인수의 스케일이 정밀도와 같아서 정밀도가 1씩 증가되는 경우를 제외하고, 결과는 인수와 동일한 자료 유형 및 길이 속성을 가집니다. 예를 들어, 자료 유형 DECIMAL(5,5)를 갖는 인수는 DECIMAL(6,5)의 결과를 가져옵니다. 정밀도가 이미 31이면 스케일이 1씩 감소합니다. 예를 들어, DECIMAL(31,31)은 DECIMAL(31,30)의 결과를 가져옵니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

예

- 호스트 변수 PROFIT가 값 50000인 큰 정수라고 가정합니다.

```
SELECT SIGN(:PROFIT)
     FROM EMPLOYEE
```

  값 1을 리턴합니다.

SIN

SIN 함수는 인수의 사인을 리턴하는데 인수는 라디안으로 표현된 각도입니다. SIN 및 ASIN 함수는 역산입니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 SINE을 값 1.5인 소수(2,1) 호스트 변수라고 가정합니다.

```
SELECT SIN(:SINE)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 0.99를 리턴합니다.

SINH

SINH 함수는 인수의 대칭 사인을 리턴하는데 인수는 라디안으로 표현된 각도입니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 HSINE를 값 1.5인 소수(2,1) 호스트 변수라고 가정합니다.

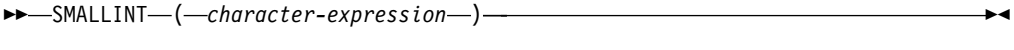
```
SELECT SINH(:HSINE)
  FROM SYSIBM.SYSDUMMY1
```

대략 값 2.12를 리턴합니다.

SMALLINT

숫자를 **Smallint**로

문자를 **Smallint**로



SMALLINT 함수는 다음의 작은 정수 표시를 리턴합니다.

- 수
- 소수의 문자 스트링 표시
- 정수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 작은 정수 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

숫자를 **Smallint**로

numeric-expression

내장된 숫자 자료 유형의 숫자 값을 리턴하는 표현식입니다.

결과는 인수가 작은 정수 열 또는 변수에 지정된 경우와 동일한 수입니다. 인수의 전체 부분이 작은 정수 범위 내에 있지 않으면 오류가 발생합니다. 인수의 분수 부분은 절단됩니다.

문자를 **Smallint**로

character-expression

표현식은 정수의 문자 스트링 표현을 리턴합니다. 표현식은 CLOB가 아니어야 합니다.

결과는 CAST에 기인한 수와 같습니다(*character-expression* AS SMALLINT). 선행 및 후미 공백은 제거되고 결과 스트링은 부동 소수점, 정수 또는 소수 상수를 형식화하기 위한 규칙을 따릅니다. 인수의 전체 부분이 작은 정수 범위 내에 있지 않으면 오류가 발생합니다. 인수의 분수 부분은 절단됩니다. 인수의 분수 부분은 절단됩니다.

함수의 결과는 작은 정수입니다. 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있습니다. 인수가 널인 경우 결과는 널값입니다.

예

- EMPLOYEE 표를 사용하여 교육 레벨(EDLEVEL)로 나눈 급여(SALARY) 리스트를 선택합니다. 연산의 소수 부분은 절단합니다. 리스트에는 또한 연산에 사용된 값과 사원 번호(EMPNO)를 포함시켜야 합니다.

```
SELECT SMALLINT(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
    FROM EMPLOYEE
```

SOUNDEX

SOUNDEX 함수는 인수의 단어의 발음을 나타내는 4문자 코드를 리턴합니다. 결과는 다른 스트링의 발음과 비교하는 데 사용됩니다.

인수는 BLOB, CLOB 또는 DBCLOB 이외의 내장 스트링 자료 유형이 될 수 있습니다.

결과의 자료 유형은 CHAR(4)입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

결과의 CCSID는 현재 서버의 디폴트 CCSID입니다.

SOUNDEX 함수는 발음은 알고 있으나 정확한 철자를 모르는 스트링을 찾을 때 유용합니다. 즉, 이것은 비슷한 발음의 단어를 찾을 때 문자 또는 문자의 조합이 도움이 된다는 것을 전제로 한 것입니다. 비교는 직접 수행되거나 스트링을 인수로 DIFFERENCE 함수에 전달하여 수행할 수도 있습니다. 자세한 내용은 219 페이지의 『DIFFERENCE』를 참조하십시오.

예

- EMPLOYEE 표를 사용하여 성이 ‘Loucesy’처럼 발음되는 사원의 EMPNO 및 LASTNAME을 찾습니다.

```
SELECT EMPNO, LASTNAME
     FROM EMPLOYEE
       WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

  다음 행을 리턴합니다.

```
000110 LUCCHESSI
```

SPACE

SPACE 함수는 인수가 지정한 SBCS 공백 수로 구성된 문자 스트링을 리턴합니다.

인수는 정수를 리턴하는 표현식입니다. 정수는 결과에 대한 SBCS 공백 수를 지정하며 이것은 0 - 32740 사이의 값이어야 합니다. *numeric-expression*이 상수인 경우 상수 0은 안됩니다.

함수 결과는 SBCS 자료를 포함하는 가변 길이 문자 스트링(VARCHAR)입니다.

*numeric-expression*이 상수인 경우 결과의 길이 속성은 상수입니다. 그렇지 않으면 결과의 길이 속성은 4000입니다. 결과의 실제 길이는 *numeric-expression* 값입니다. 결과의 실제 길이는 결과의 길이 속성보다 커서는 안됩니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널 값입니다.

CCSID는 작업의 SBCS 자료에 대한 EBCDIC CCSID입니다.

예

- 다음 명령문은 5개의 공백으로 구성된 문자 스트링을 리턴합니다.

```
SELECT SPACE(5)
  FROM SYSIBM.SYSDUMMY1
```

SQRT

SQRT 함수는 수의 제곱근을 리턴합니다.

인수는 모든 내장 숫자 자료 유형의 값을 리턴하는 표현식입니다. *numeric-expression* 값은 0보다 크거나 같아야 합니다. 인수는 함수가 처리할 배정밀도 부동 소수점으로 변환됩니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

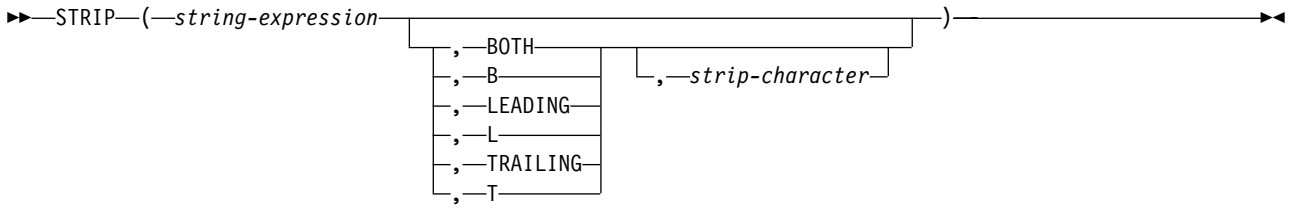
예

- 호스트 변수 SQUARE를 값 9.0인 DECIMAL(2,1) 호스트 변수라고 가정합니다.

```
SELECT SQRT(:SQUARE)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 3.00을 리턴합니다.

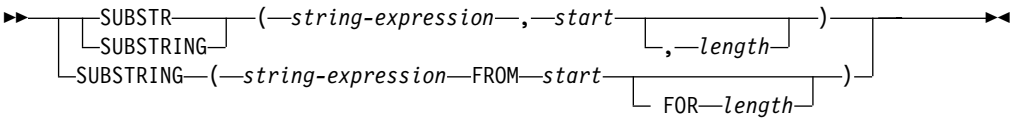
STRIP



STRIP 함수는 스트링 표현식의 끝 또는 시작에서 공백 또는 다르게 지정된 문자를 제거합니다.

STRIP 함수는 TRIM 스칼라 함수와 같습니다. 자세한 내용은 313 페이지의 『TRIM』을 참조하십시오.

SUBSTRING 또는 SUBSTR



SUBSTR 및 SUBSTRING 함수는 스트링의 서브스트링을 리턴합니다.

string-expression

결과가 나온 스트링을 지정하는 표현식.

*String-expression*은 숫자, 그래픽 또는 2진 스트링이어야 합니다. *string-expression*이 문자 스트링인 경우 결과는 문자 스트링입니다. 이 값이 그래픽 스트링인 경우, 함수 결과는 그래픽 스트링입니다. 이 값이 2진 스트링인 경우, 함수 결과는 2진 스트링입니다.

*string-expression*의 서브스트링은 *string-expression*의 0개 이상의 연속 문자입니다. *string-expression*이 그래픽 스트링인 경우, 문자는 DBCS 또는 UCS-2 문자입니다. *string-expression*이 문자 스트링인 경우, 결과는 문자 스트링 또는 2진 스트링인 경우, 문자는 1바이트입니다. SUBSTR 함수는 혼합 자료 스트링을 허용합니다. 그러나, SUBSTR가 엄격한 바이트 계수에 의해 연산되므로 결과는 반드시 올바르게 형식화된 혼합 자료 문자 스트링일 필요는 없습니다.

start

결과의 첫 번째 문자(또는 바이트)의 *string-expression* 내 위치를 지정하는 표현식. 이 값은 2진 정수여야 합니다. *start*는 음수 또는 0입니다. 이 값은 *string-expression*의 길이 속성보다 커야 합니다(가변 길이 스트링의 길이 속성은 스트링의 최대 길이입니다).

length

결과의 길이를 지정하는 표현식을 나타냅니다. 이 값이 지정된 경우, *length*는 2진 정수여야 합니다. *length*는 0 - *n* 범위의 2진 정수이어야 하며 여기서, *n*은 결과 자료 유형의 최대 길이입니다.

SUBSTR이 지정되었으며 *length*가 명시적으로 지정된 경우, 지정된 *string-expression*의 서브스트링이 항상 존재하도록 *string-expression*의 오른쪽이 필요한 수의 공백 문자(또는 2진 스트링의 경우 16진수 0)로 채워집니다.

SUBSTRING이 지정되고 *length*가 명시적으로 지정된 경우, 채우기가 수행되지 않습니다.

*string-expression*이 고정 길이 스트링인 경우 *length*를 생략하는 것은 LENGTH(*string-expression*) - *start* + 1의내재적 스펙으로서 *start* 문자(또는 바이트)에서 *string-expression*의 마지막 문자(또는 바이트)까지의 문자 수입니다.

*string-expression*이 가변 길이 스트링인 경우 *length*를 생략하는 것은 0 또는 LENGTH(*string-expression*) - *start* + 1의 내재적 스펙입니다. 결과 길이가 0인 경우 결과는 공백 스트링입니다.

결과의 자료 유형은 *string-expression*의 자료 유형과 함수가 SUBSTR 또는 SUBSTRING인지 여부에 따라 다릅니다.

*string-expression*의 자료 유형	**SUBSTRING**에 대한 결과의 자료 유형	**SUBSTR**에 대한 결과의 자료 유형
CHAR 또는 VARCHAR	VARCHAR	*length*가 정수 상수에 의해 명시적으로 지정되는 경우 또는 *length*가 명시적으로 지정되지 않지만 *string-expression*이 고정 길이 스트링이고 *start*가 정수 상수인 경우 CHAR. 모든 다른 경우에는 VARCHAR
GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC	*length*가 정수 상수에 의해 명시적으로 지정되는 경우 또는 *length*가 명시적으로 지정되지 않지만 *string-expression*이 고정 길이 스트링이고 *start*가 정수 상수인 경우 GRAPHIC. 모든 다른 경우에는 VARGRAPHIC
BLOB	BLOB	BLOB
CLOB	CLOB	CLOB
DBCLOB	DBCLOB	DBCLOB

SUBSTRING 함수가 지정되면 결과의 길이 속성은 *string-expression*의 길이 속성과 같습니다.

SUBSTR 함수가 지정되고 *string-expression*이 LOB가 아닌 경우, 결과의 길이 속성은 *length*, *start* 및 *string-expression* 속성에 따라 다릅니다.

- *length*가 정수 상수에 의해 명시적으로 지정되는 경우 결과의 길이 속성은 *length*입니다.
- *length*가 명시적으로 지정되지 않지만 *string-expression*이 고정 길이 스트링이고 *start*가 정수 상수인 경우, 결과의 길이 속성은 LENGTH(*string-expression*) - *start* + 1입니다.

모든 다른 경우에는 결과의 길이 속성이 *string-expression*의 길이 속성과 같습니다. (*string-expression*의 실제 길이가 *start* 값보다 작은 경우 서브스트링의 실제 길이는 0임을 기억하십시오).

SUBSTR에 기초한 소스 함수는 항상 가변 길이 스트링인 결과를 갖습니다.

SUBSTR 함수의 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 인수가 널인 경우 결과는 널값입니다.

결과의 CCSID는 *string-expression*의 CCSID와 같습니다.

예

- 호스트 변수 NAME(VARCHAR(50))이 'KATIE AUSTIN' 값을 갖고 호스트 변수 SURNAME_POS(INTERGER)가 7 값을 갖는다고 가정합니다.

```
SELECT SUBSTR(:NAME, :SURNAME_POS)
  FROM SYSIBM.SYSDUMMY1
```

  'AUSTIN' 값을 리턴합니다.

- 이와 같이,

```
SELECT SUBSTR(:NAME, :SURNAME_POS, 1)
  FROM SYSIBM.SYSDUMMY1
```

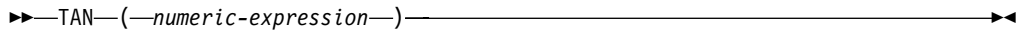
  값 'A'를 리턴합니다.

- PROJECT 표에서 'OPERATION '로 시작하는 프로젝트명(PROJNAME)이 있는 모든 행을 선택하십시오.

```
SELECT * FROM PROJECT
  WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

  상수 끝의 간격은 'OPERATIONS'와 같은 처음 단어를 제외시키는 데 필요합니다.

TAN



TAN 함수는 인수의 탄젠트를 리턴하는데 인수는 각으로 표현된 라디안 입니다. TAN 및 ATAN 함수는 역산입니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 TANGENT가 값 1.5인 DECIMAL(2,1) 호스트 변수라고 가정합니다.

```
SELECT TAN(:TANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 14.10을 리턴합니다.

TANH

TANH 함수는 인수의 대칭 사인을 리턴하는데 인수는 라디안으로 표현된 각도입니다. TANH 및 ATANH 함수는 역산입니다.

인수는 내장 숫자 자료 유형 값을 리턴하는 표현식입니다.

결과 자료 유형은 배정밀도 부동 소수점입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- 호스트 변수 HTANGENT가 값 1.5인 DECIMAL(2,1) 호스트 변수라고 가정합니다.

```
SELECT TANH(:HTANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  대략 값 0.90을 리턴합니다.

TIME

TIME 함수는 값에서 시간을 리턴합니다.

주: 또한 CAST 표현식은 시간 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

인수는 내장 자료 유형 날짜, 시간소인 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다. *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 시간입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 시간인 경우:

  결과는 그 시간입니다.
- 인수가 시간소인인 경우:

  결과는 시간소인의 시간 부분입니다.
- 인수가 문자 스트링인 경우:

  시간의 스트링 표시가 SBCS 자료의 디폴트 CCSID와 같지 않은 CCSID를 갖는 SBCS 자료일 때 값은 시간 값을 해석하고 변환하기 전에 SBCS 자료의 디폴트 CCSID로 변환됩니다.

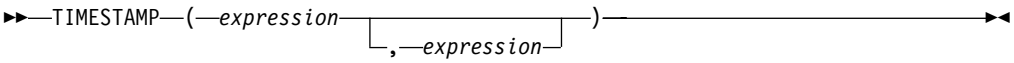
  시간의 스트링 표시가 혼합 자료의 디폴트 CCSID와 같지 않은 CCSID를 갖는 혼합 자료일 때 값은 시간 값을 해석하고 변환하기 전에 혼합 자료의 디폴트 CCSID로 변환됩니다.

예

- IN_TRAY 샘플 표에서 현재 시간보다 최소한 한 시간 뒤에 받은 모든 주를 선택합니다.

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



TIMESTAMP 함수는 해당 인수로부터 시간소인을 리턴합니다.

주: 또한 CAST 표현식은 시간소인을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

인수에 대한 규칙은 두 번째 인수가 지정되었는지의 여부에 따라 다릅니다.

- 하나의 인수만 지정된 경우:

  인수는 내장 자료 유형 시간소인 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

  *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 다음 중 하나여야 합니다.

  – 날짜 또는 시간소인의 유효한 문자 스트링 표현. 날짜 및 시간소인의 유효 스트링 표현은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

  – 유효한 날짜가 yyyynnn 형식으로 표시되는 실제 길이 7의 문자 스트링. 여기서, yyyy는 연도를 표시하는 수이고 nnn은 해당 년도의 일 수를 표시하는 001 - 366 사이의 숫자입니다.

  – 형식 yyyyxxddhhmmss로 유효한 날짜와 시간을 표시하는 실제 길이 14인 문자 스트링으로서, 여기서, yyyy는 연도, xx는 월, dd는 일, hh는 시간, mm은 분 및 ss는 초입니다.

- 모든 인수가 지정된 경우:

  첫 번째 인수는 내장 자료 유형 날짜 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다. 두 번째 인수는 내장 자료 유형 시간 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

  *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜의 유효 문자 스트링 표현이어야 합니다. *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 시간의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 시간소인입니다. 인수 중의 하나가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 인수 중 하나가 널인 경우 결과는 널값입니다.

다른 규칙은 두 번째 인수가 지정되었는지의 여부에 따라 다릅니다.

- 모든 인수가 지정된 경우:

결과는 첫 번째 인수에 지정된 날짜와 두 번째 인수에 지정된 시간을 갖는 시간소인입니다. 시간소인의 마이크로초 부분은 0입니다.

- 하나만 지정된 인수가 시간소인인 경우:

  결과는 해당 시간소인입니다.

- 하나만 지정된 인수가 문자 스트링인 경우:

  결과는 문자 스트링으로 표시된 시간소인입니다. 인수가 길이 14인 문자 스트링인 경우 시간소인의 마이크로초 부분은 0입니다.

시간소인의 스트링 표시가 SBCS 자료의 디폴트 CCSID와 같지 않은 CCSID를 갖는 SBCS 자료일 때 값은 시간소인 값을 해석하고 변환하기 전에 SBCS자료의 디폴트 CCSID로 변환됩니다.

시간소인의 스트링 표시가 혼합 자료의 디폴트 CCSID와 같지 않은 CCSID를 갖는 혼합 자료일 때 값은 시간소인 값을 해석하고 변환하기 전에 혼합 자료의 디폴트 CCSID로 변환됩니다.

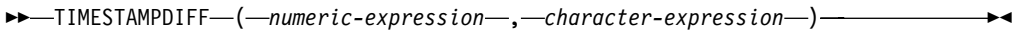
예

- 날짜 및 시간 값이 다음과 같은 경우:

```
SELECT TIMESTAMP( DATE('1988-12-25'), TIME('17.12.30') )
  FROM SYSIBM.SYSDUMMY1
```

  값 '1988-12-25-17.12.30.000000'을 리턴합니다.

TIMESTAMPDIFF



TIMESTAMPDIFF 함수는 두 개의 시간소인 간의 차이에 따라 첫 번째 인수에 의해 정의된 유형의 간격에 대한 예상 값을 리턴합니다.

첫 번째 인수는 내장 자료 유형 INTEGER 또는 SMALLINT여야 합니다. 간격(첫 번째 인수)에 대하여 유효한 값은 다음과 같습니다.

1	1초의 분수
2	초
4	분
8	시
16	요일
32	주
64	월
128	사분기
256	연도

두 번째 인수는 두 개의 시간소인 유형을 뺀 후 그 결과를 CHAR(22)로 변환한 결과입니다.

함수의 결과는 정수입니다. 인수 중의 하나가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 인수 중 하나가 널인 경우 결과는 널값입니다.

다음 가정은 그 차를 예측하는 데 사용될 수 있습니다.

- 1년은 365일
- 1달은 30일
- 1일은 24시간
- 1시간은 60분
- 1분은 60초

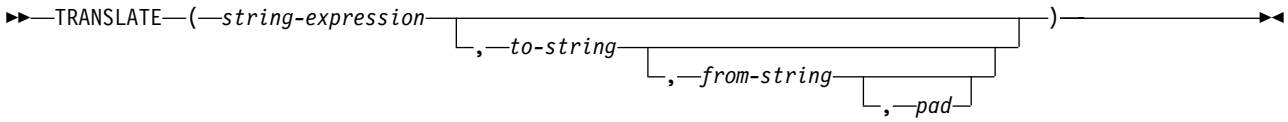
이러한 가정은 두 번째 인수 정보인 시간소인 기간을 첫 번째 인수에 지정된 간격 유형으로 변환할 때 사용됩니다. 리턴된 추정값은 날 수에 따라 다를 수 있습니다. 예를 들어 '1997-03-01-00.00.00'과 '1997-02-01-00.00.00'의 시간소인의 차를 날 수(간격 16)로 구해야 하는 경우 결과는 30입니다. 이것은 시간소인 차가 1달이고 한달은 30일이라는 가정이 적용되었기 때문입니다.

예

- 사원의 나이를 달 수로 산정합니다.

```
SELECT
    TIMESTAMPDIFF(64,
    CAST(CURRENT_TIMESTAMP-CAST(BIRTHDATE AS TIMESTAMP) AS CHAR(22)))
      AS AGE_IN_MONTHS
  FROM EMPLOYEE
```

TRANSLATE



TRANSLATE 함수는 *string-expression*의 하나 이상의 문자가 다른 문자로 변환될 수 있는 값을 리턴합니다.

string-expression

변환된 *string-expression*이 될 스트링을 지정하는 표현식은 문자 스트링 또는 UCS-2 그래픽 스트링이어야 합니다.

to-string

*string-expression*의 특정 문자가 변환될 문자를 지정하는 스트링. 이 스트링을 *출력 변환 표*로 부르는 경우가 있습니다. 스트링은 문자 스트링 상수여야 합니다. 문자 스트링 인수의 크기는 256보다 크지 않아야 합니다.

*to-string*의 길이 속성이 *from-string*의 길이 속성보다 작은 경우 *to-string*은 *pad* 또는 공백을 사용하여 더 긴 길이가 되도록 채웁니다. *to-string*의 길이 속성이 *from-string*의 길이 속성보다 큰 경우 *to-string*의 여분의 문자는 경고없이 무시됩니다.

from-string

*string-expression*에 있는 경우 변환될 문자를 지정하는 스트링. 이 스트링을 *입력 변환 표*로 부르는 경우가 있습니다. *from-string*의 문자가 발견되면, *string-expression*의 문자는 *from-string*의 해당 문자 위치에 있는 *to-string* 문자로 변환됩니다.

스트링은 문자 스트링 상수여야 합니다. 문자 스트링 인수의 실제 길이는 256보다 커서는 안됩니다.

*from-string*에 중복 문자가 있는 경우 먼저 왼쪽에서 스캔이 사용되고 경고는 발행되지 않습니다. *from-string*에 대한 디폴트 값은 문자 X'00'로 시작하고 문자 X'FF'로 끝나는 (십진 255) 스트링입니다.

pad

길이가 *from-string*보다 작은 경우 *to-string*을 채울 문자를 지정합니다. 스트링은 길이 1인 문자 스트링 상수여야 합니다. 디폴트 값은 SBCS 공백입니다.

첫 번째 인수가 UCS-2 그래픽 스트링인 경우 다른 인수를 지정할 수 없습니다.

첫 번째 인수만 지정된 경우 인수의 SBCS 문자는 인수의 CCSID에 따라 대문자로 변환됩니다. SBCS 문자만이 변환됩니다. 문자 a-z는 A-Z로 변환되고, 발음 구별 마크가 있는 문자는 해당 소문자(있는 경우)로 변환됩니다. 첫 번째 인수가 UCS-2 그래픽이면

영문자 UCS-2 문자가 대문자로 변환됩니다. 이 변환에 사용되는 모노캐스팅 표에 대한 설명은 iSeries Information Center의 국제화 주제의 UCS-2 레벨 1 맵핑 표 섹션을 참조하십시오.

둘 이상의 인수가 지정되면, 결과 스트링은 *from-string*의 문자를 *to-string*의 연관된 문자로 변환하여 *expression*의 문자로 구축된 문자입니다. *string-expression*의 각 문자의 경우같은 문자는 *from-string*에서 탐색됩니다. 문자가 *from-string*의 *n*번째 문자로 발견된 경우 결과 스트링은 *to-string*에 *n*번째 문자가 들어 있습니다. *to-string*의 길이가 *n*자보다 작은 경우 결과 스트링에 채움 문자가 들어 있습니다. 문자가 *from-string*에서 발견되지 않은 경우 변환되지 않은 결과 스트링은 이동됩니다.

변환은 바이트에 따라 수행되며, 부적절하게 사용되는 경우 유효하지 않은 혼합 스트링이 결과될 수 있습니다. SRTSEQ 속성은 TRANSLATE 함수에 적용되지 않습니다.

함수의 결과는 동일한 자료 유형, 길이 속성, 실제 길이 및 인수와 같은 CCSID를 갖습니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있습니다. 인수가 널인 경우 결과는 널값입니다.

예

- 스트링 'abcdef'를 모노캐이스로 처리합니다.

```
SELECT TRANSLATE('abcdef')
  FROM SYSIBM.SYSDUMMY1
```

  값 'ABCDEF'를 리턴합니다.

- 혼합 문자 스트링을 모노캐이스 처리하십시오.

```
SELECT TRANSLATE('ab<SO>c<SI>def')

  FROMSYSIBM.SYSDUMMY1
```

  'AB<SO>C<SI>DEF' 값을 리턴합니다.

- 주어진 호스트 변수 SITE는 값 'Pivabiska Lake Place'를 갖는 가변 길이 문자 스트링입니다.

```
SELECT TRANSLATE(:SITE, '$', 'L')
  FROM SYSIBM.SYSDUMMY1
```

  값 'Pivabiska $ake Place'를 리턴합니다.

```
SELECT TRANSLATE(:SITE, '$$', 'Ll')
  FROM SYSIBM.SYSDUMMY1
```

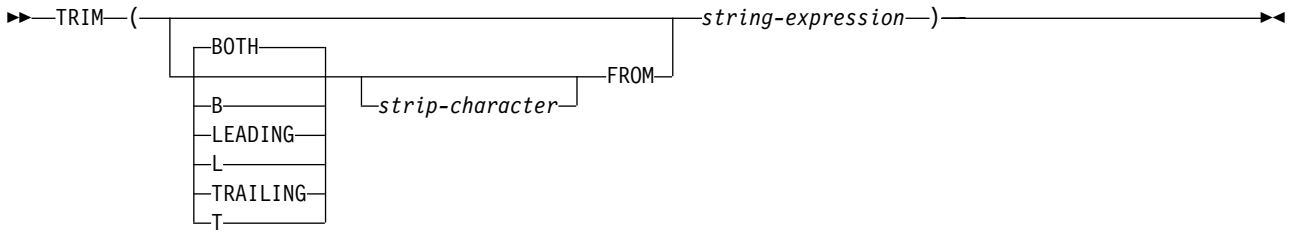
  값 'Pivabiska $ake P$ace'를 리턴합니다.

```
SELECT TRANSLATE(:SITE, 'pLA', 'Place', '.')
  FROM SYSIBM.SYSDUMMY1
```

TRANSLATE

값 ‘pivAbiskA LAk. pLA..’을 리턴합니다.

TRIM



TRIM 함수는 스트링 표현식의 끝 또는 시작에서 공백 또는 다르게 지정된 문자를 제거합니다.

*string-expression*은 스트링 표현식이어야 합니다.

지정된 경우 첫 번째 인수는 스트링의 끝 또는 시작에서 문자를 제거해야 하는지 여부를 표시합니다. 첫 번째 인수가 지정되지 않으면 문자들이 스트링의 끝과 시작에서 제거됩니다.

두 번째 인수는 지정되는 경우 제거될 2진, SBCS 또는 DBCS 문자를 표시하는 단일 문자 상수입니다. *string-expression*이 2진 스트링이면 두 번째 인수는 반드시 2진 스트링 상수이어야 합니다. *string-expression*이 DBCS 그래픽 또는 DBCS 전용 스트링인 경우 두 번째 인수는 하나의 DBCS 문자가 들어 있는 그래픽 상수이어야 합니다. 두 번째 인수가 지정되지 않은 경우:

- *string-expression*이 2진 스트링이면 디폴트스트링 문자는 16진 0(X'00')입니다.
- *string-expression*이 DBCS 그래픽 스트링인 경우 디폴트 스트링 문자는 DBCS 공백입니다.
- *string-expression*이 UCS-2 그래픽 스트링인 경우 디폴트 스트링 문자는 UCS-2 공백입니다.
- 그렇지 않으면 디폴트 스트링 문자는 SBCS 공백입니다.

결과의 자료 유형은 *string-expression*의 자료 유형에 따라 다릅니다.

*expression*의 자료 유형	결과의 자료 유형
CHAR 또는 VARCHAR	VARCHAR
GRAPHIC 또는 VARGRAPHIC	VARGRAPHIC
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB

결과의 길이 유형은 *string-expression*의 길이 유형과 같습니다.결과의 실제 길이는 표현식의 길이에서 제거된 바이트 수를 뺀 길이입니다. 모든 문자가 제거되면 결과는 빈 스트링입니다.

첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

결과의 CCSID는 스트링의 CCSID와 같습니다.

SRTSEQ 속성은 TRIM 함수에 적용되지 않습니다.

예

- 유형 CHAR(9)인 호스트 변수 HELLO가 값 ' Hello'를 갖는다고 가정합니다.

```
SELECT TRIM(:HELLO), TRIM( TRAILING FROM :HELLO)
  FROM SYSIBM.SYSDUMMY1
```

  각각 'Hello' 및 ' Hello'를 나타냅니다.

- 유형 CHAR(9)인 호스트 변수 BALANCE가 값 '000345.50'을 갖는다고 가정합니다.

```
SELECT TRIM( L '0' FROM :BALANCE )
  FROM SYSIBM.SYSDUMMY1
```

  결과는 '345.50'이 됩니다.

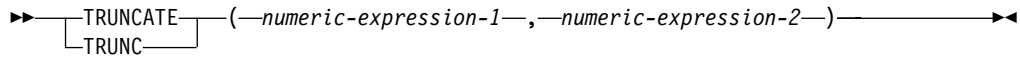
- 스트립된 스트링에 혼합 자료가 있다고 가정합니다.

  SELECT TRIM(BOTH 'SO SI' **FROM** 'SO AB C SI' **)**

  FROMSYSIBM.SYSDUMMY1

  결과: 'SOAB CSI'

TRUNCATE 또는 TRUNC



TRUNCATE 함수는 소수점 왼쪽 및 오른쪽의 특정 자리수에서 잘린 *numeric-expression-1*을 리턴합니다.

numeric-expression1
내장된 숫자 자료 유형의 값을 리턴하는 표현식입니다.

numeric-expression2
작은 정수나 큰 정수를 리턴하는 표현식입니다. 정수의 절대값은 *numeric-expression-2*가 음수가 아닌 경우 결과에 대한 소수점의 오른쪽으로 또는 *numeric-expression-2*가 음수인 경우 소수점의 왼쪽으로 자리수를 지정합니다.

*numeric-expression-2*가 음수가 아니면 *numeric-expression-1*은 소수점의 오른쪽 *numeric-expression-2* 자리수에서 반올림됩니다.

*numeric-expression-2*가 음수이면 *numeric-expression-1*은 소수점의 왼쪽으로 (*numeric-expression-2*+1)의 절대 자리수로 반올림됩니다.

*numeric-expression-2*의 절대값이 소수점 왼쪽의 자리수보다 크면, 결과는 0입니다. 예를 들면, TRUNCATE(748.58,-4) = 0입니다.

결과의 자료 유형과 길이 속성은 첫 번째 인수의 자료 유형 및 길이 속성과 같습니다.

한 인수가 널(null)일 수 있으면 결과도 널(null)일 수 있습니다. 한 인수가 널(null)이면 결과는 널값입니다.

예

- 가장 높은 급여의 사원에 대한 평균 급여를 연산합니다. 결과를 소수점 오른쪽 두 자리로 절단합니다.

```
SELECT TRUNCATE(MAX(SALARY/12, 2)
     FROM EMPLOYEE
```

  샘플 사원 표에서 가장 높은 급여의 사원이 년간 $52750.00을 받기 때문에 예에서는 값 4395.83을 리턴합니다.

- 2, 1, 0, -1, -2 및 -3 소수 자리로 각각 절단하여 숫자 873.726을 연산합니다.

```
SELECT TRUNCATE(873.726,  2),
       TRUNCATE(873.726,  1),
       TRUNCATE(873.726,  0),
       TRUNCATE(873.726, -1),
       TRUNCATE(873.726, -2),
       TRUNCATE(873.726, -3)
  FROM SYSIBM.SYSDUMMY1
```

각각 다음 값을 리턴합니다.

```
0873.720   0873.700  0873.000  0870.000  0800.000  0000.000
```

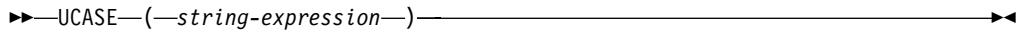
- 양수와 음수 모두 연산하십시오.

```
SELECT TRUNCATE( 3.5, 0),
       TRUNCATE( 3.1, 0),
       TRUNCATE(-3.1, 0),
       TRUNCATE(-3.5, 0)
  FROM SYSIBM.SYSDUMMY1
```

각각 다음 값을 리턴합니다.

```
3.0   3.0  -3.0  -3.0
```

UCASE



UPPER 함수는 인수의 CCSID에 따라 모든 문자를 대문자로 변환된 스트링을 리턴합니다.

UCASE 함수는 UPPER 함수와 같습니다. 자세한 내용은 318 페이지의 『UPPER』을 참조하십시오.

UPPER

UPPER 함수는 인수의 CCSID에 따라 모든 문자를 대문자로 변환된 스트링을 리턴합니다. SBCS 및 UCS-2 그래픽 문자만이 변환됩니다. 문자 a-z는 A-Z로 변환되고, 발음 구별 마크가 있는 문자는 해당 대문자(있는 경우)로 변환됩니다. 이 변환에 사용되는 모노캐스팅 표에 대한 설명은 iSeries Information Center의 국제화 주제의 UCS-2 레벨 1 맵핑 표 섹션을 참조하십시오.

string-expression

변환될 스트링을 지정하는 표현식. *String-expression*은 문자이거나 UCS-2 그래픽 스트링이어야 합니다.

함수의 결과는 동일한 자료 유형, 길이 속성, 실제 길이 및 인수와 같은 CCSID를 갖습니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

UCASE는 UPPER의 동의어입니다.

예

- UPPER 스칼라 함수를 사용하여 스트링 ‘abcdef’를 대문자로 변경하십시오.

```
SELECT UPPER('abcdef')
    FROM SYSIBM.SYSDUMMY1
```

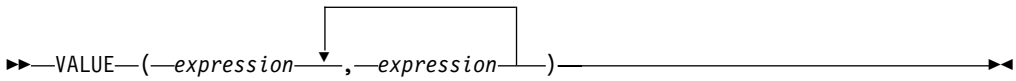
  값 ‘ABCDEF’를 리턴합니다.

- UPPER 스칼라 함수를 사용하여 혼합 문자 스트링을 대문자로 바꾸십시오.

```
SELECT UPPER('ab<SO>c<SI>def')
    FROM SYSIBM.SYSDUMMY1
```

  리턴 값: 'AB<SO>C<SI>DEF'

VALUE

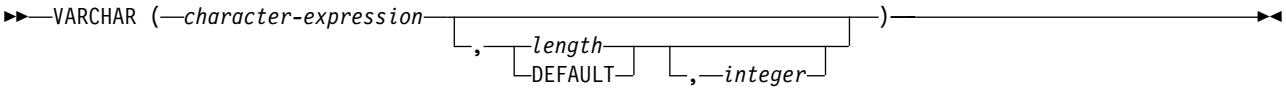


VALUE 함수는 널이 아닌 첫 번째 표현식 값을 리턴합니다.

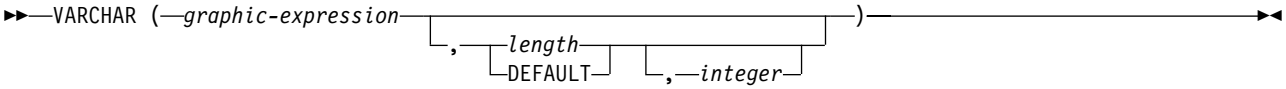
VALUE 함수는 COALESCE 스칼라 함수와 같습니다. 자세한 내용은 197 페이지의 『COALESCE』를 참조하십시오.

VARCHAR

문자 대 Varchar

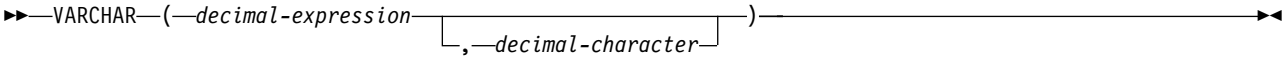


그래픽 대 Varchar

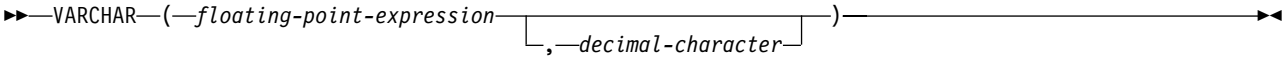


정수 대 Varchar

소수 대 Varchar



부동 소수점 대 Varchar



VARCHAR 함수는 다음과 같은 문자 스트링 표시를 리턴합니다.

- 첫 번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우 정수
- 첫 번째 인수가 팩 또는 존 십진수인 경우 십진수
- 첫 번째 인수가 DOUBLE 또는 REAL인 경우 배정밀도 부동 소수점 수
- 첫 번째 인수가 문자 스트링 유형인 경우 문자 스트링
- 첫 번째 인수가 UCS-2 그래픽 스트링인 경우 그래픽 스트링

주: 또한 CAST 표현식은 가변 길이 문자 스트링 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수의 결과는 변수 길이 스트링입니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

문자 대 Varchar

character-expression

내장 CHAR, VARCHAR 또는 CLOB 자료 유형인 값을 리턴하는 표현식.

length

결과 변수 길이 문자 스트링에 길이 속성을 지정합니다. 값은 1 - 32740(널 가능인 경우 32739) 사이의 값이어야 합니다. 첫 번째 인수가 혼합 자료인 경우 두 번째 인수는 4보다 작을 수 없습니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정되는 경우

- *character-expression*이 빈 스트링 상수이면 결과의 길이 속성은 1입니다.
- 그렇지 않으면 결과의 길이 속성은 인수가 첫 번째 인수의 길이 속성과 같습니다.

결과의 실제 길이는 결과의 길이 속성과 *character-expression*의 실제 길이의 최소값입니다. 문자 표현의 길이가 결과 길이 속성보다 큰 경우 절단이 수행됩니다. 절단된 문자가 모두 공백이 아니면 경고(SQLSTATE 01004)가 리턴됩니다.

integer

결과의 CCSID를 지정합니다. 유효한 SBCS CCSID, 혼합 자료 CCSID 또는 65535(비트 자료)이어야 합니다. 세 번째 인수가 SBCS CCSID인 경우 결과는 SBCS 자료입니다. 세 번째 인수가 혼합 CCSID인 경우 결과는 혼합 자료입니다. 세 번째 인수가 65535인 경우 결과는 비트 자료입니다. 세 번째 인수가 SBCS CCSID인 경우 첫 번째 인수는 DBCS 선택 또는 DBCS 전용 스트링일 수 없습니다.

세 번째 인수는 다음과 같은 경우 지정되지 않습니다.

- 첫 번째 인수가 SBCS 자료인 경우 결과는 SBCS 자료입니다. 결과 CCSID는 첫 번째 인수의 CCSID와 같습니다.
- 첫 번째 인수가 혼합 자료(DBCS 개방, DBCS 전용 또는 DBCS 선택)인 경우 결과는 혼합 자료입니다. 결과 CCSID는 첫 번째 인수의 CCSID와 같습니다.

그래픽 대 Varchar

graphic-expression

GRAPHIC, VARGRAPHIC 또는 DBCLOB 자료 유형 값을 리턴하는 표현식. DBCS 그래픽 자료이어서는 안됩니다.

length

결과 변수 길이 문자 스트링에 길이 속성을 지정합니다. 값은 1 - 32740(널 가능인 경우 32739) 사이의 값이어야 합니다. 첫 번째 인수에 DBCS 자료가 들어 있는 경우, 두 번째 인수는 4보다 작을 수 없습니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정되는 경우 결과의 길이 속성은 다음과 같이 판별됩니다(여기서 *n*은 첫 번째 인수의 길이 속성입니다).

- *graphic-expression*이 빈 그래픽 스트링 상수인 경우 결과의 길이 속성은 1입니다.
- 결과가 SBCS 자료인 경우 결과 길이는 *n*입니다.

- 결과가 혼합 자료인 경우 결과 길이는 (2.5*(*n*-1)) + 4입니다.

결과의 실제 길이는 결과의 길이 속성과 *graphic-expression*의 실제 길이의 최소값입니다. 문자 표현의 길이가 결과 길이 속성보다 큰 경우 절단이 수행됩니다. 절단된 문자가 모두 공백이 아니면 경고(SQLSTATE 01004)가 리턴됩니다.

integer
결과의 CCSID를 지정합니다. 유효한 SBCS CCSID 또는 혼합 자료 CCSID이어야 합니다. 세 번째 인수가 SBCS CCSID인 경우 결과는 SBCS 자료입니다. 세 번째 인수가 혼합 CCSID인 경우 결과는 혼합 자료입니다. 세 번째 인수는 65535가 될 수 없습니다.

세 번째 인수가 지정되지 않은 경우 결과의 CCSID는 현재 서버에서 디폴트 CCSID입니다. 디폴트 CCSID가 혼합 자료인 경우 결과는 혼합 자료입니다. 디폴트 CCSID가 SBCS 자료인 경우 결과는 SBCS 자료입니다.

정수 대 Varchar

integer-expression
정수 자료 유형(SMALLINT, INTEGER 또는 BIGINT 중 하나)인 값을 리턴하는 표현식

결과는 SQL 정수 상수의 양식에서 인수의 가변 길이 문자 스트링 표시입니다. 결과는 인수가 음인 경우 선행 음의 부호를 갖는 인수 값을 표시하는 유효 숫자인 n자로 구성됩니다. 좌측 정렬됩니다.

- 인수가 작은 정수인 경우 결과의 길이 속성은 6입니다.
- 인수가 큰 정수인 경우 결과의 길이 속성은 11입니다.
- 인수가 큰 정수인 경우 결과의 길이 속성은 20입니다.

결과의 실제 길이는 인수의 값을 표시하는 데 사용될 수 있는 최소 문자입니다. 선행 0은 포함되지 않습니다. 인수가 음인 경우 결과의 첫 번째 문자는 음의 부호입니다. 그렇지 않으면 첫 번째 문자는 숫자입니다.

결과의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

소수 대 Varchar

decimal-expression
팩 또는 존 십진 자료 유형(DECIMAL 또는 NUMERIC)인 값을 리턴하는 표현식. 다른 정밀도와 스케일이 요구되는 경우 DECIMAL 스칼라 함수는 변경하는 데 사용될 수 있습니다.

decimal-character
결과 문자 스트링에 소수 자리를 분리하는 데 사용되는 1바이트 문자 상수를 지정

합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 소수점을 사용합니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 인수의 가변 길이 문자 스트링 표시입니다. 결과는 소수 문자를 포함하고 *p* 자릿수까지이며, 여기서 *p*는 인수가 음인 경우 선행 음의 부호를 갖는 *decimal-expression*의 정밀도입니다. 선행 0은 리턴되지 않습니다. 후미 0은 리턴됩니다.

결과의 길이 속성은 2+*p*이며, 여기서 *p*는 *decimal-expression*의 정밀도입니다. 결과의 실제 길이는 포함된 후미 문자를 제외한 결과를 표시하는 데 사용될 수 있는 최소 문자입니다. 선행 0은 포함되지 않습니다. 인수가 음인 경우 결과는 음의 부호로 시작합니다. 그렇지 않은 경우 결과는 숫자로 시작합니다.

결과의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

부동 소수점 대 Varchar

floating-point expression
부동 소수점 자료 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식

decimal-character
결과 문자 스트링에 소수 자리를 분리하는 데 사용되는 1바이트 문자 상수를 지정합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 소수점을 사용합니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 부동 소수점 상수의 형식에서 인수의 가변 길이 문자 스트링 표시입니다.

결과의 길이 속성은 24입니다. 결과의 실제 길이는 0이 아닌 하나의 숫자 다음에 *decimal-character* 및 일련의 숫자가 오는 가수와 같은 인수 값을 나타내는 최소 문자입니다. 인수가 음인 경우 결과의 첫 번째 문자는 음의 부호이며 그렇지 않은 경우 첫 번째 문자는 숫자입니다. 인수가 0인 경우 결과는 0E0입니다.

결과의 CCSID는 현재 서버에서 디폴트 SBCS CCSID입니다.

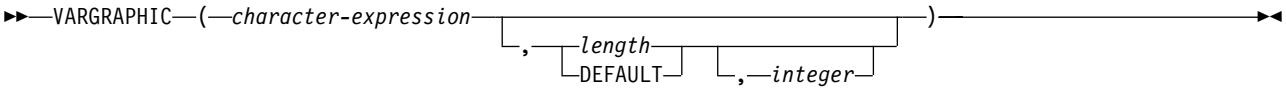
예

- 길이 10인 EMPNO 가변 길이를 작성합니다.

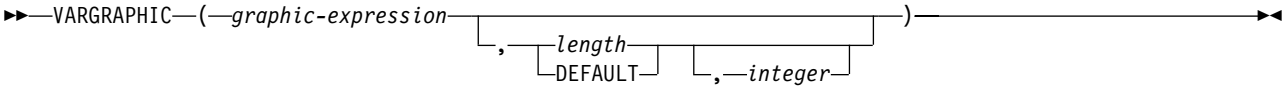
```
SELECT VARCHAR(EMPNO,10)
  INTO :VARHV
                FROM EMPLOYEE
```

VARGRAPHIC

문자에서 그래픽으로



그래픽에서 그래픽으로



VARGRAPHIC 함수는 스트링 표현식의 그래픽 스트링 표시를 리턴합니다.

주: 또한 CAST 표현식은 가변 길이 그래픽 스트링 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수의 결과는 가변 길이 그래픽 스트링(VARGRAPHIC)입니다.

표현식이 널이 될 수 있으면, 결과도 널이 될 수 있습니다. 표현식이 널인 경우 결과는 널값입니다. 표현식이 공백 스트링 또는 EBCDIC 스트링 `X'0E0F'`인 경우 결과는 공백 스트링입니다.

문자에서 그래픽으로

character-expression
문자 스트링 표현식을 지정합니다. CHAR 또는 VARCHAR 비트 자료일 수 없습니다.

length
결과의 길이 속성을 지정하며, 첫 번째 인수가 널이 아닌 경우 1-16370 사이의 정수 상수이고 첫 번째 상수가 널인 경우 1-16369 사이의 정수 상수입니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정된 경우 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다. 단, 표현식이 빈 스트링이거나 EBCDIC 스트링 X'0E0F'인 경우 결과의 길이 속성은 1입니다.

결과의 실제 길이는 인수의 문자 수에 따라 다릅니다. 인수의 각 문자는 결과의 문자를 판별합니다. 결과 가변 길이 스트링의 길이 속성이 첫 번째 인수의 실제 길이보다 작은 경우 절단이 수행되고 경고는 리턴되지 않습니다.

integer
결과의 CCSID를 지정합니다. DBCS 또는 UCS-2 CCSID이어야 합니다. CCSID는

65535가 될 수 없습니다. CCSID가 UCS-2 그래픽 자료인 경우 인수의 각 문자는 결과 문자를 판별합니다. 결과의 n 번째 문자가 인수의 n 번째 문자와 등가의 UCS-2입니다.

*integer*가 지정되지 않으면 결과의 CCSID는 혼합 CCSID에 의해 판별됩니다. M을 혼합 CCSID로 지정합니다.

다음 규칙에서 S는 다음 중의 하나를 표시합니다.

- 스트링 표현식이 외국어 코드화 체계의 자료가 들어 있는 호스트 변수인 경우 S는 자국어 코드화 체계의 CCSID로 자료를 변환한 다음의 표현식 결과입니다(자세한 내용은 34 페이지의 『문자 변환』을 참조하십시오).
- S는 스트링 표현식이 자국어 코드화 체계 자료인 경우의 스트링 표현식입니다.

M은 다음과 같이 판별됩니다.

- S의 CCSID가 혼합 CCSID인 경우 M은 그 CCSID입니다.
- S의 CCSID가 SBCS CCSID인 경우:
  - S의 CCSID가 혼합 CCSID와 연관된 경우 M은 그 CCSID입니다.
  - 그렇지 않으면 연산이 허용되지 않습니다.

다음 표는 M에 따라 CCSID 결과를 요약합니다.

M	결과 CCSID	설명	DBCS 대체 문자
930	300	일본어 EBCDIC	X'FEFE'
933	834	한국어 EBCDIC	X'FEFE'
935	837	중국어 EBCDIC	X'FEFE'
937	835	대만어 EBCDIC	X'FEFE'
939	300	일본어 EBCDIC	X'FEFE'
5026	4396	일본어 EBCDIC	X'FEFE'
5035	4396	일본어 EBCDIC	X'FEFE'

SBCS와 DBCS 문자의 등가는 M에 따라 다릅니다. CCSID와 관계없이 인수의 모든 2바이트 코드점은 DBCS 문자로 간주되고 모든 1바이트 코드점은 EBCDIC 혼합 자료 시프트 코드 `X'0E'` 및 `X'0F'`를 제외하고는 SBCS 문자로 간주됩니다.

- 인수의 n 번째 문자가 DBCS 문자인 경우 결과의 n 번째 문자는 해당 DBCS 문자입니다.
- 인수의 n 번째 문자가 등가의 DBCS 문자를 갖는 SBCS 문자인 경우 결과의 n 번째 문자는 등가의 DBCS 문자입니다.
- 인수의 n 번째 문자가 등가의 DBCS 문자가 아닌 SBCS 문자인 경우 결과의 n 번째 문자는 DBCS 대체 문자입니다.

그래픽에서 그래픽으로

graphic-expression

그래픽 스트링 표현식을 지정합니다.

length

결과의 길이 속성을 지정하며, 첫 번째 인수가 널이 아닌 경우 1-16370 사이의 정수 상수이고 첫 번째 상수가 널인 경우 1-16369 사이의 정수 상수입니다.

두 번째 인수가 지정되지 않거나 DEFAULT가 지정된 경우 결과의 길이 속성은 첫 번째 인수의 길이 속성과 같습니다. 단, 표현식이 빈 스트링인 경우 결과의 길이 속성은 1입니다.

결과의 실제 길이는 인수의 문자 수에 따라 다릅니다. 인수의 각 문자는 결과의 문자를 판별합니다. 결과 가변 길이 스트링의 길이 속성이 첫 번째 인수의 실제 길이보다 작은 경우 절단이 수행되고 경고는 리턴되지 않습니다.

integer

결과의 CCSID를 지정합니다. DBCS 또는 UCS-2 CCSID이어야 합니다. CCSID는 65535가 될 수 없습니다.

*integer*가 지정되지 않으면 결과의 CCSID는 첫 번째 인수의 CCSID입니다.

예

- EMPLOYEE 표를 사용하여 호스트 변수 VAR_DESC(VARGRAPHIC(24))를 사원 번호(EMPNO) '000050'의 이름(FIRSTNME)과 동일한 VARGRAPHIC로 설정합니다.

```
SELECT VARGRAPHIC(FIRSTNME)
  INTO :VAR_DESC
     FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

WEEK

WEEK 함수는 주를 표시하는 1 - 54 사이의 정수를 표시합니다. 일요일부터 시작하고, 1월 1일이 항상 첫 번째 주입니다.

인수는 내장 자료 유형 날짜, 시간소인 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- PROJECT 표를 사용하여 호스트 변수 WEEK (INTERGER)에 프로젝트('PL2100') 를 끝낸 주(week)로 설정합니다.

```
SELECT WEEK(PRENDATE)
  INTO :WEEK
  FROM PROJECT
  WHERE PROJNO = 'PL2100'
```

  WEEK가 38으로 설정됩니다.

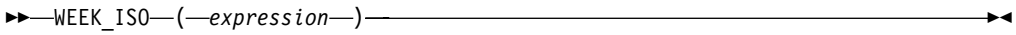
- 표 X에 DATE 열 DATE_1이 있고 이 열에 아래 리스트의 여러 날짜가 있는 경우,

```
SELECT DATE_1, WEEK(DATE_1)
  FROM X
```

  다음 리스트에서는 여러 가지 날짜에 대하여 WEEK 함수가 리턴한 값을 보여줍니다.

```
1997-12-28    53
1997-12-31    53
1998-01-01    1
1999-01-01    1
1999-01-04    2
1999-12-31    53
2000-01-01    1
2000-01-03    2
```

WEEK_ISO



WEEK 함수는 주를 표시하는 1 - 53 사이의 정수를 표시합니다. 주는 월요일부터 시작합니다. 1주차는 목요일이 들어가는 1년 중 첫 번째 주로서 1월 4일이 포함되는 첫 번째 주이기도 합니다. 따라서 최대 3일까지 전년도의 마지막 주를 처음에 표시할 수 있거나 끝에 다음 년도의 첫 주를 최대 3일까지 표시할 수 있습니다.

인수는 내장 자료 유형 날짜, 시간소인 또는 문자 스트링 중 한 가지 값을 리턴하는 표현식이어야 합니다.

*expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현 형식은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

예

- PROJECT 표를 사용하여 호스트 변수 WEEK (INTERGER)에 프로젝트('AD2100')를 끝낸 주(week)로 설정합니다.

```
SELECT WEEK_ISO(PRENDATE)
  INTO :WEEK
  FROM PROJECT
  WHERE PROJNO = 'AD3100'
```

  WEEK가 5으로 설정됩니다.

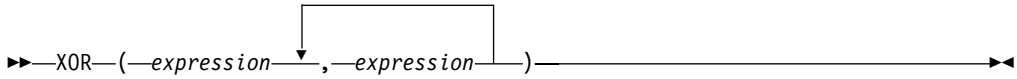
- 표 X에 DATE 열 DATE_1이 있고 이 열에 아래 리스트의 여러 날짜가 있는 경우,

```
SELECT DATE_1, WEEK_ISO(DATE_1)
  FROM X
```

  결과는 다음과 같습니다.

```
1997-12-28    52
1997-12-31    1
1998-01-01    1
1999-01-01    53
1999-01-04    1
1999-12-31    52
2000-01-01    52
2000-01-03    1
```

XOR



XOR 함수는 인수 스트링의 논리 연산 XOR인 스트링을 리턴합니다. 이 함수가 첫 번째 인수 스트링을 가지면, 다음 스트링과 XOR 비교를 수행하고, 이전 결과를 사용하여 각각의 다음 인수와 XOR 비교 수행을 계속합니다. 인수가 이전 결과보다 작으면, 공백으로 채워집니다.

인수는 문자 스트링이어야 하나 LOB가 될 수 없습니다. 인수는 혼합 문자 스트링 또는 그래픽 스트링이 될 수 없습니다.

인수는 필요한 경우 결과 속성으로 변환됩니다. 결과 속성은 다음과 같이 판별됩니다.

- 모든 인수가 고정 길이 스트링인 경우 결과는 길이 *n*인 고정 길이 스트링이며, 여기서 *n*은 가장 긴 인수의 길이입니다.
- 모든 인수가 가변 길이 스트링인 경우 결과는 길이 속성 *n*인 가변 길이 스트링이며, 여기서 *n*은 가장 긴 속성을 갖는 인수의 길이 속성입니다. 결과의 실제 길이는 *m*이며 여기서, *m*은 가장 긴 인수의 실제 길이입니다.

인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며, 인수가 널인 경우 결과도 널 값입니다.

결과의 CCSID는 65535입니다.

예

- 호스트 변수 L1은 값 X'E1E1'을 갖는 CHARACTER(2) 호스트 변수, 호스트 변수 L2는 값 X'F0F000'을 갖는 CHARACTER(3) 호스트 변수, 호스트 변수 L3은 값 X'0000000F'를 갖는 CHARACTER(4) 호스트 변수라고 가정합니다.

```
SELECT XOR(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

  값 X'1111404F'를 리턴합니다. 이때 짧은 결과는 공백(X'40')으로 채워져 논리 XOR는 처음 예와 달라집니다.

```
SELECT XOR(:L3,:L2,:L1)
  FROM SYSIBM.SYSDUMMY1
```

  값 X'1111400F'를 리턴합니다.

YEAR

YEAR 함수는 값의 연도 부분을 리턴합니다.

인수는 내장 자료 유형 날짜, 시간소인, 문자 스트링 또는 숫자 자료 유형 중 한 가지 값을 리턴하는 표현식이어야 합니다.

- *expression*이 문자 스트링인 경우, CLOB여서는 안되고 값은 날짜 또는 시간소인의 유효 문자 스트링 표현이어야 합니다. 날짜 및 시간소인의 유효 스트링 표현은 70 페이지의 『Datetime 값의 스트링 표시』에서 참조하십시오.
- *expression*이 숫자인 경우, 날짜 기간 또는 시간소인 기간이어야 합니다. datetime 기간의 유효 형식은 133 페이지의 『Datetime 피연산자와 기간』에서 참조하십시오.

함수의 결과는 큰 정수입니다. 인수가 널이 될 수 있는 경우 결과도 널이 될 수 있으며 인수가 널인 경우 결과도 널값입니다.

다른 규칙들은 인수의 자료 유형에 따라 다릅니다.

- 인수가 날짜, 시간소인, 날짜 또는 시간소인의 유효 문자 스트링 표현인 경우:

  결과는 값의 연도 부분이며, 1 - 9999 사이의 정수 값입니다.
- 인수가 시간소인 기간 또는 시간소인 기간인 경우:

  결과는 값의 연도 부분이며, -9999 - 9999 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예

- PROJECT 표에서 같은 연도에 시작(PRSTDATE)하고 끝나도록(PRENDATE) 스케줄된 모든 프로젝트를 선택합니다.

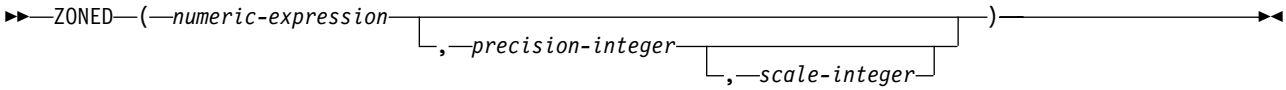
```
SELECT * FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

- PROJECT 표에서 완료할 연도보다 먼저 스케줄된 모든 프로젝트를 선택하십시오.

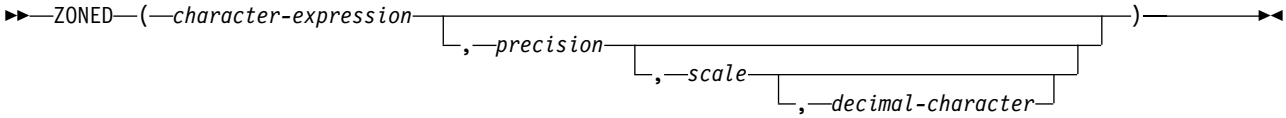
```
SELECT * FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

ZONED

숫자 대 존(zone) 소수



문자 대 존(zone) 소수



ZONED 함수는 다음의 존 소수 표시를 리턴합니다.

- 수
- 정수의 문자 스트링 표시
- 소수의 문자 스트링 표시
- 부동 소수점의 문자 스트링 표시

주: 또한 CAST 표현식은 존 십진 값을 리턴하는 데 사용될 수 있습니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

함수의 결과는 정밀도 *p*와 스케일 *s*인 존 소수이며, 여기서 *p* 및 *s*는 두 번째와 세 번째 인수입니다. 첫 번째 인수가 널이 될 수 있는 경우 결과는 널이 될 수 있으며, 첫 번째 인수가 널인 경우 결과는 널값입니다.

숫자 대 존(zone) 소수

numeric-expression

내장된 숫자 자료 유형의 값을 리턴하는 표현식입니다.

precision

1보다 크거나 같고 31 보다 작거나 같은 값으로 설정된 정수 상수.

*precision*의 디폴트는 *numeric-expression*의 자료 유형에 따라 다릅니다.

- 부동 소수점, 소수, 숫자 또는 0이 아닌 스케일 2진인 경우 15
- 큰 정수의 경우 19
- 큰 정수인 경우 11
- 작은 정수인 경우 5

scale

0 이상이고 *precision* 이하인 정수 상수. 지정되지 않은 경우 디폴트는 0입니다.

결과는 첫 번째 인수가 정밀도 *p*와 스케일 *s*인 소수 열 또는 변수에 지정된 경우 발생하는 같은 수입니다. 수 전체를 표시하기 위해 필요한 소수 자릿수가 *p-s*보다 큰 경우 오류가 발생합니다.

문자 대 존(zone) 소수

character-expression
숫자의 문자 스트링 표현을 포함해야 하는 표현식. 선행 및 후미 공백은 제거되고 결과 스트링은 정수 또는 소수 상수를 형식화하기 위한 규칙을 따라야 합니다. 표현식은 CLOB가 아니어야 합니다.

precision
1보다 크거나 같고 31 보다 작거나 같은 정수 상수. 지정되지 않은 경우 디폴트는 15입니다.

scale
0 이상이고 *precision* 이하인 정수 상수. 지정되지 않은 경우 디폴트는 0입니다.

decimal-character
수 전체에서 *문자 표현*의 소수 자리를 분리하는 데 사용된 1바이트 문자 상수를 지정합니다. 문자는 마침표 또는 쉼표이어야 합니다. 두 번째 인수가 지정되지 않는 경우 소수점은 디폴트 분리 문자를 사용합니다. 자세한 내용은 102 페이지의 『소수점』을 참조하십시오.

결과는 CAST(*character-expression* AS NUMERIC(*p,s*))의 결과가 되는 수와 같습니다. 자릿수는 *decimal-character*의 오른쪽 자릿수가 스케일 *s*보다 큰 경우에 끝이 절단됩니다. *character-expression*에서 *decimal-character*(수 전체 부분) 왼쪽의 유효 자릿수가 *p-s*보다 큰 경우 오류가 발생합니다. 디폴트 소수 분리 문자는 *decimal-character* 인수가 지정된 경우 서브스트링에서 유효하지 않습니다.

예

- 호스트 변수 Z1을 값 1.123인 소수 호스트 변수라고 가정합니다.

```
SELECT ZONED(:Z1,15,14)
  FROM SYSIBM.SYSDUMMY1
```

  값 1.12300000000000을 리턴합니다.

- 호스트 변수 Z1을 값 1123인 소수 호스트 변수라고 가정합니다.

```
SELECT ZONED(:Z1,11,2)
  FROM SYSIBM.SYSDUMMY1
```

  값 1123.00을 리턴합니다.

- 이와 같이,

```
SELECT ZONED(:Z1,4)
  FROM SYSIBM.SYSDUMMY1
```

  값 1123을 리턴합니다.

ZONED

제 4 장 조회

SQL *조회*는 결과표 또는 중간 결과표를 지정합니다.

조회는 SQL문의 구성요소입니다. 세가지 양식의 조회가 있습니다.

- *subselect*
- *fullselect*
- *select*문

다른 선택 양식은 766 페이지의 『SELECT INTO』에서 설명합니다.

또한 『권한부여』를 참조하십시오.

권한부여

조회 양식의 경우 명령문의 권한부여 ID가 보유한 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 각 표 또는 뷰에 대하여
  - 표 또는 뷰에서의 SELECT 권한 및
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에서 시스템 권한 *OBJOPR 및 *READ를 부여받고, 이 뷰가 직접 또는 간접적으로 종속되어 있는 모든 표 및 뷰에서 시스템 권한 *READ를 부여받았을 때 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

*표현식*이 함수를 포함하는 경우 명령문의 권한부여 ID는 각 사용자 정의 함수에 대해 적어도 다음 중 하나를 포함해야 합니다.

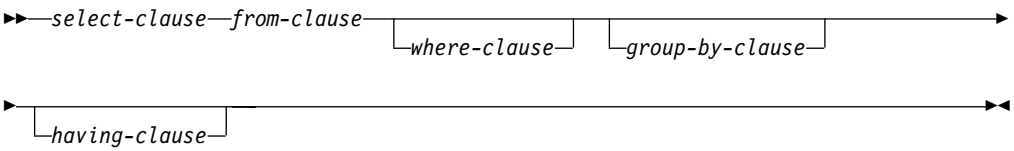
- 함수에서 EXECUTE 권한

- 관리 권한

명령문의 권한부여 ID는 다음 경우에 함수에 대한 EXECUTE 권한을 갖습니다.

- 함수의 소유자입니다.
- 함수에서 EXECUTE 권한을 부여받았습니다.
- 함수에서 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

subselect



*subselect*는 fullselect, CREATE VIEW문 및 INSERT문의 구성요소입니다. 또한 일부 술부의 구성요소이며, 따라서 이들 술부도 subselect의 구성요소입니다.

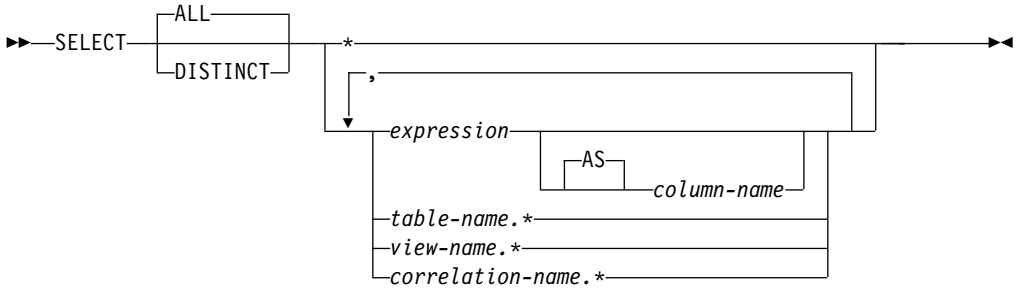
*scalar-subselect*는 부속 선택이며(괄호로 묶여있는) 단일 결과 행 및 단일 결과 열을 리턴합니다. 부속선택의 결과가 행이 아니면 널 값이 리턴됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

subselect는 FROM절에서 식별된 표 또는 뷰에서 파생된 결과표를 지정합니다. 파생은 각 조작의 결과가 다음 조작에 대한 입력인 조작 순서로 설명될 수 있습니다(이것은 subselect를 설명하는 단지 하나의 방법입니다. 파생을 수행하기 위해 사용된 메소드는 이 설명과 다를 수 있습니다).

(가설적인) 조작 순서는 다음과 같습니다.

1. FROM절
2. WHERE절
3. GROUP BY절
4. HAVING절
5. SELECT절

select절



SELECT절은 최종 결과표의 열을 지정합니다. 열 값은 R에 대한 *선택 리스트*의 어플리케이션에 의해 생성됩니다. 선택 리스트는 SELECT절에 지정된 이름 또는 표현식이며, R은 subselect의 이전 조작 결과입니다. 예를 들어, 지정된 절이 SELECT, FROM 및 WHERE이면 R은 해당 WHERE절의 결과입니다.

ALL

최종 결과표의 행 모두를 선택하고 중복을 제거하지 않습니다. 이것이 디폴트 값입니다.

DISTINCT

최종 결과표의 중복 행 각 세트 중 하나를 제외하고 모두 제거합니다.

첫 번째 행에서 각 값이 두 번째 행의 대응값과 같은 경우에만 두 행이 서로 중복됩니다(중복 행을 판별하기 위해 두 널값은 같다고 간주됩니다). 정렬 순서도 고유한 값 판별에 사용됩니다.

선택 리스트가 LOB 또는 DATALINK 열을 포함하는 경우 DISTINCT는 허용되지 않습니다.

선택 리스트 표기법

***** 표 R의 열을 식별하는 이름 리스트를 표시합니다. 리스트에서 첫 번째 이름은 R의 첫 번째 열을 식별하고, 두 번째 이름은 R의 두 번째 열을 식별합니다.

이름 리스트는 SELECT절이 들어 있는 명령문이 준비될 때 설정됩니다. 따라서, * 는 명령문이 준비된 후에 표에 추가된 열은 식별하지 않습니다.

expression

127 페이지의 『표현식』에서 설명된 유형의 표현식은 모두 가능합니다. *expression*에서 각 *column-name*은 R의 열을 분명하게 식별해야 합니다.

colum-name 또는 **AS** *column-name*

결과 열을 명명 또는 재명명합니다. 이름은 규정되지 않아야 하며 고유할 필요는 없습니다.

*name.**

name 열을 식별하는 이름 리스트를 표시합니다. *name*은 표 이름, 뷰 이름 또는 상관명일 수 있으며, FROM절에서 명명된 표 또는 뷰를 지정해야 합니다. 리스트에서 첫 번째 이름은 표 또는 뷰의 첫 번째 열을 식별하고, 리스트의 두 번째 이름은 표 또는 뷰의 두 번째 열을 식별합니다.

이름 리스트는 SELECT절이 들어 있는 명령문이 준비될 때 설정됩니다. 따라서, * 는 명령문이 준비된 후에 표에 추가된 열은 식별하지 않습니다.

보통, SQL문이 내재적으로 리바인드될 때 이름 리스트는 재설정되지 않습니다. 따라서, 명령문에 의해 리턴된 열 수는 변경되지 않습니다. 그러나, 이름 리스트가 다시 설정되고 열 수가 변경될 수 있는 경우가 4가지 있습니다.

- SQL 프로그램 또는 SQL 패키지가 보관되고 보관된 시트템과 동일한 릴리스가 아닌 iSeries 시스템에서 복원될 때
- 명명하는 SQL이 SQL 프로그램 또는 패키지에 대해 지정되고 프로그램 소유자가 SQL 프로그램 또는 패키지가 작성된 이후로 변경될 때
- 더 최근의 OS/400 릴리스가 설치된 후 SQL문이 최초로 실행될 때
- SELECT *가 INSERT문의 subselect에서 발생하거나 술부 내의 subselect에서 발생할 때 그리고 subselect에서 참조된 표 또는 뷰가 삭제되고 추가 열로 재작성될 때

SELECT 결과에서 열 수는 선택 리스트(즉, 준비시에 설정된 리스트) 조작 양식의 표현식 수와 동일하며, 8000을 넘을 수 없습니다. 부속 조회가 EXISTS 술부에서 사용되지 않았으면 부속 조회 결과는 단일 표현식이어야 합니다.

선택 리스트 적용

선택 리스트를 R에 적용한 결과는 GROUP BY 또는 HAVING 사용 여부에 따라 달라집니다. 그런 결과는 별도로 설명됩니다.

GROUP BY 또는 HAVING이 사용된 경우:

- 선택 리스트에서 *column-name*을 포함하는 각 표현식은 그룹화 표현식을 식별하거나 열 함수 내에서 지정되어야 합니다.
  - 그룹화 표현식이 열 이름인 경우 선택 리스트는 열 이름에 추가 연산자를 적용할 수도 있습니다. 예를 들어 그룹화 표현식이 열 C1인 경우 선택 리스트에는 C1+1이 포함될 수 있습니다.
  - 그룹화 표현식이 열 이름이 아닌 경우 선택 리스트는 표현식에 추가적인 연산자를 적용하지 않을 수도 있습니다. 예를 들어 그룹화 표현식이 C1+1인 경우 선택 리스트에는 C1+1이 포함되지만 (C1+1)/8은 포함되지 않습니다.
- RRN, PARTITION, NODENAME 및 NODENUMBER 함수는 선택 리스트에서 지정될 수 없습니다.

- 선택 리스트가 R의 각 그룹에 적용되고, 결과는 R의 그룹 수 만큼의 행을 포함합니다. 선택 리스트가 R의 그룹에 적용될 때 해당 그룹은 선택 리스트에 있는 열 함수의 인수 소스입니다.

GROUP BY도 HAVING도 사용되지 않은 경우:

- 선택 리스트는 열 함수를 포함해서는 안되거나 완전히 열 함수 리스트여야 합니다.
- 선택 리스트가 열 함수를 포함하지 않는 경우 선택 리스트는 R의 각 행에 적용되고, 결과는 R에 있는 행 수 만큼의 행을 포함합니다.
- 선택 리스트가 열 함수 리스트인 경우 R은 함수 인수의 소스이며 선택 리스트를 적용한 결과는 한 행입니다.

두 경우에서 결과의 *n*번째 열은 선택 리스트의 조작 양식에서 *n*번째 표현식을 적용하여 지정된 값을 포함합니다.

결과 열의 널(null) 속성

결과 열은 열이 다음에서 파생된 경우 널값을 허용합니다.

- COUNT 및 COUNT_BIG을 제외한 열 함수
- 널값을 허용하는 열
- 널값을 허용하는 피연산자가 있는 스칼라 함수 또는 표현식
- 인디케이터 변수를 갖고 있는 호스트 변수
- 최소한 선택 리스트의 대응 항목 중 하나가 널이 될 수 있는 경우 UNION 결과
- 연산식
- 스칼라 부속 선택
- 사용자 정의 스칼라 또는 표 함수

결과 열 이름

- AS절이 지정된 경우 결과 열 이름은 AS절에서 지정된 이름입니다.
- 열 리스트가 상관절에 지정된 경우 결과 열 이름은 상관 열 리스트에 있는 대응하는 이름입니다.
- 상관절에서 AS절도 열 리스트도 지정되지 않고 결과 열은 단일 열(함수나 연산자 없이)에서만 파생되는 경우 결과 열 이름은 해당 열의 규정되지 않은 이름입니다.
- 모든 다른 결과 열은 명명되지 않습니다.

결과 열의 자료 유형

SELECT의 각 결과 열은 파생된 표현식에서 자료 유형을 구합니다.

표현식이 다음과 같은 경우	결과 열의 자료 유형은 다음과 같습니다.
숫자 열의 이름	열의 자료 유형과 동일, 소수 열과 동일한 정밀도와 눈금을 가짐

표현식이 다음과 같은 경우	결과 열의 자료 유형은 다음과 같습니다.
정수 상수	INTEGER 또는 BIGINT(상수 값이 INTEGER의 범위를 벗어나지만 BIGINT의 범위 안에 있는 경우)
소수 또는 부동 소수점 상수	상수의 자료 유형과 동일, 소수 상수와 동일한 정밀도와 눈금을 가짐.
숫자 호스트 변수명	변수의 자료 유형과 동일, 소수 변수와 동일한 정밀도와 눈금을 가짐. 변수의 자료 유형이 SQL 자료 유형과 동일하지 않은 경우(예: COBOL의 DISPLAY SIGN LEADING SEPARATE), 결과 열은 소수입니다.
연산식	결과의 자료 유형과 동일, 127 페이지의 『표현식』에서 설명한 소수 결과와 동일한 정밀도와 눈금을 가짐
함수	함수의 결과의 자료 유형. 내장 함수에 대해서 결과의 자료 유형을 판별하려면 제 3 장을 참조하십시오. 사용자 정의 기능에서 결과의 자료 유형은 함수의 CREATE FUNCTION 문에서 정의 됩니다.
스트링 열의 이름	열의 자료 유형과 동일, 동일한 길이 속성을 가짐
스트링 호스트 변수명	변수의 자료 유형과 동일, 변수 길이와 같은 길이 속성을 가짐 변수의 자료 유형이 SQL 자료 유형과 동일하지 않은 경우(예:C의 NUL 종료 스트링), 결과 열은 가변 길이 스트링입니다.
길이 *n*인 문자 스트링 상수	VARCHAR(*n*)
길이 *n*인 그래픽 스트링 상수	VARGRAPHIC(*n*)
날짜 시간 열 또는 ILE RPG 컴파일러이나 ILE COBOL 컴파일러 날짜 시간 호스트 변수의 이름	열 또는 호스트 변수 자료 유형과 동일
자료 링크 열의 이름	동일한 길이 속성을 가진 자료 링크
행 ID 열 또는 행 ID 호스트 변수의 이름	ROWID
scalar-subselect	scalar-subselect의 선택 리스트에 있는 표현식의 자료 유형

from절

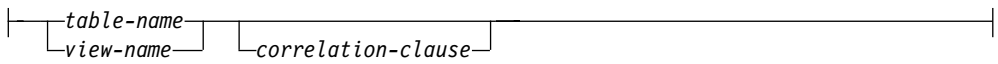
FROM절은 중간 결과표를 지정합니다.

하나의 *table-reference*가 지정된 경우 중간 결과표는 해당 *table-reference*의 결과입니다. 하나 이상의 *table-reference*가 FROM절에서 지정된 경우 중간 결과표는 지정된 table-references 행의 가능한 모든 조합으로 이루어집니다(Cartesian 제품). 결과의 각 행은 두 번째 *table-reference*의 행과 연결된 첫 번째 *table-reference*의 행이며, 두 번째 행은 세 번째 행과 연결되며, 뒤의 다른 행도 이런 식으로 계속 연결됩니다. 결과

행 수는 모든 개별 *table-reference*에 있는 행 수의 곱입니다. *table-reference*에 대한 설명은 『table-reference』를 참조하십시오.

table-reference

single-table:



nested-table 표현식:

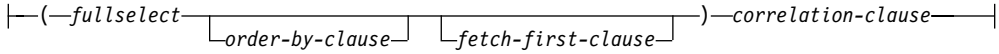
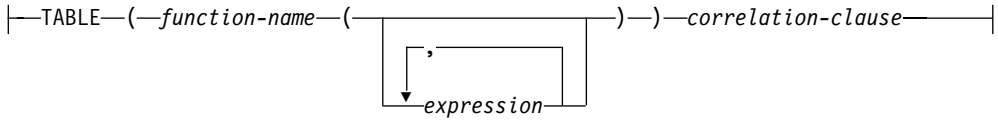


table-function:



correlation절:

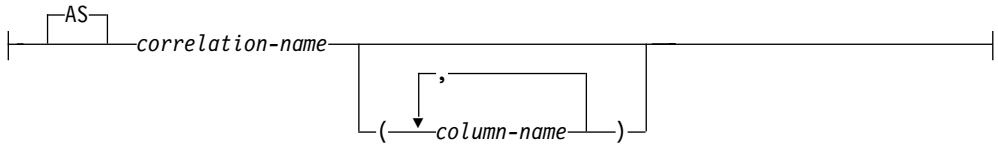


*table-reference*는 중간 결과표를 지정합니다.

- 단일 표 또는 뷰가 식별된 경우 중간 결과표는 단지 해당 표 또는 뷰입니다.
- WITH 절의 괄호 안의 전체 선택을 *내포된 표 표현식*이라고 합니다. [39] 내포된 표 표현식이 지정된 경우 결과표는 해당 내포된 표 표현식의 결과입니다. 결과 열은 고유명을 가질 필요는 없지만, 고유하지 않은 이름을 갖는 열은 참조될 수 없습니다.
- 만일 *function-name*이 지정된 경우 중간 결과 표는 표 함수에 의해 리턴된 행의 집합입니다.
- *joined-table*이 지정된 경우 중간 결과표는 하나 이상의 결합 조작의 결과입니다. 자세한 내용은 343 페이지의 『joined-table』을 참조하십시오.

39. *내포된 표 표현식*은 또한 *파생된 표*라고도 합니다.

FROM절에서 이름 리스트는 다음 규칙에 따라야 합니다.

- 각 *table-name* 및 *view-name*은 현재 서버에 있는 기존 표나 뷰 또는 table-reference가 들어 있는 subselect 이전에 정의된 *common-table 표현식*(353 페이지의 『common-table 표현식』 참조)의 table-name을 명명해야 합니다.
- 노출된 이름은 고유해야 합니다. 노출된 이름은 correlatin-name, 다음에 correlatin-name이 오지 않는 table-name 또는 다음에 correlatin-name이 오지 않는 view-name입니다.
- 해당 인수 유형과 함께 각 *function-name*은 현재 서버에 존재하는 표 함수로 분석되어야 합니다. 123 페이지의 『함수 분석』에 기술된 함수 분석이라는 알고리즘은 함수 이름 및 인수를 사용하여 사용할 정확한 함수를 판별합니다. *correlation-clause*의 해당 열 이름을 제외하고, 표 함수의 열 이름은 CREATE FUNCTION문의 RETURNS절에 지정된 것입니다. 이는 CREATE TABLE에 정의된 열 이름과 유사합니다.

각 *correlation-name*은 바로 앞의 table-reference에 의해 지정된 중간 결과표의 지정자로 정의됩니다. 상관명은 내포된 표 표현식 및 표 함수에 대해 지정되어야 합니다.

모든 표 참조의 노출된 이름은 고유해야 합니다. 노출된 이름은 다음과 같습니다.

- *correlation-name*
- 다음에 *correlation-name*이 오지 않는 *table-name* 또는 *view-name*

표, 뷰, 내포된 표 표현식, 파생된 표 또는 표 함수에 해당하는 열에 대한 규정된 참조는 노출된 이름을 사용해야 합니다. 동일한 표 또는 뷰 이름이 두 번 지정된 경우 적어도 한 스펙 뒤에는 *correlation-name*이 나와야 합니다. *correlation-name*은 표나 뷰의 열에 대한 참조를 규정하는 데 사용됩니다. *correlation-name*이 지정될 때 column-names는 *table-name*, *view-name*, *nested-table 표현식* 또는 *table-function*의 열에 이름을 부여하기 위해 지정될 수도 있습니다. 열 리스트가 지정된 경우 표나 뷰의 각 열, 그리고 *nested-table 표현식* 또는 *table-function*의 각 결과 열에 대한 열 리스트에 이름이 있어야 합니다. 자세한 내용은 108 페이지의 『상관명』을 참조하십시오.

일반적으로, *nested-table 표현식*및 *table-functions*은 *from절*에서 지정될 수 있습니다. 내포된 표 표현식 및 표 함수의 열은 선택 리스트에서 참조될 수 있으며, 지정되어야 하는 상관명을 사용하여 나머지 부속 선택에서도 참조될 수 있습니다. 이런 상관명의 범위는 FROM절에 있는 다른 표나 뷰 이름에 대한 상관명과 동일합니다. 내포된 표 표현식은 다음과 같은 경우에 사용될 수 있습니다.

- 뷰 작성을 피하기 위해 뷰 대신에(일반적인 뷰 사용이 필요하지 않을 때)
- 원하는 결과표가 호스트 변수를 기준으로 할 때

table-reference에서 상관 참조: 상관 참조는 내포된 표 표현식에서 사용할 수 있습니다. 적용되는 기본 규칙은 상관 참조는 부속 조회 계층의 상위 레벨의 *table-reference*에서 참조해야 한다는 것입니다. 이 계층은 FROM절의 왼쪽에서 오른쪽 처리에서 이미 해석된 *table-reference*를 포함합니다.

참조 표가 FROM절의 표에서 왼쪽에서 오른쪽 순의 참조를 선행할 경우 표 함수는 동일한 FROM절의 다른 표로의 하나 이상의 상관 참조를 포함할 수 있습니다. 아니면, 하위 쿼리 계층의 더 높은 레벨에 대한 참조만이 허용됩니다.

예 1: 다음 예는 유효합니다.

```
SELECT D.DEPTNO, D.DEPTNAME, EMPINFO.AVGSAL, EMPINFO.EMPCOUNT
FROM DEPARTMENT D,
  (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT, E.WORKDEPT AS DEPT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT =
       (SELECT X.DEPTNO
         FROM DEPARTMENT X
         WHERE X.DEPTNO = E.WORKDEPT ) GROUP BY E.WORKDEPT)
AS EMPINFO
WHERE D.DEPTNO = EMPINFO.DEPT
```

*nested-table 표현식*의 WHERE절에서 D.DEPTNO에 대한 참조가 부속 조회 계층 밖에 있는 표를 참조하려고 시도하기 때문에 다음 예는 유효하지 않습니다.

```
SELECT D.DEPTNO, D.DEPTNAME, EMPINFO.AVGSAL, EMPINFO.EMPCOUNT
FROM DEPARTMENT D,
  (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT = D.DEPTNO ) AS EMPINFO
```

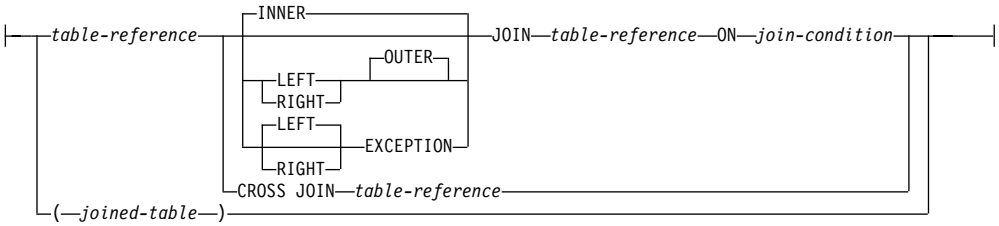
예 2: 다음의 표 함수 예는 유효합니다:

```
SELECT t.c1, z.c5
FROM t, TABLE(tf3 (t.c2 ) ) AS z WHERE t.c3 = z.c4
```

FROM 절에서 표 함수의 오른쪽 표에 대한 참조가 t.c2이기 때문에 다음의 예는 유효하지 않습니다:

```
SELECT t.c1, z.c5
FROM TABLE(tf6 (t.c2 ) ) AS z, t
WHERE t.c3 = z.c4
```

joined-table



*joined-table*은 내부, 외부, 교차 또는 예외 결합의 결과인 중간 결과 표를 지정합니다. 표는 결합 연산자 즉 INNER, LEFT OUTER, RIGHT OUTER, LEFT EXCEPTION, RIGHT EXCEPTION 또는 CROSS 중 하나를 피연산자에 적용하여 파생됩니다.

결합 연산자가 지정되지 않은 경우 INNER가 내재됩니다. 복수 결합이 수행되는 순서는 결과에 영향을 미칠 수 있습니다. 결합은 다른 결합 내에 내포될 수 있습니다. 결합에 대한 처리 순서는 일반적으로 왼쪽에서 오른쪽이지만, 필요한 join-condition 위치를 기준으로 합니다. 내포된 결합 순서를 쉽게 읽을 수 있도록 괄호를 사용하는 것이 좋습니다. 예를 들면, 다음과 같습니다.

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
    LEFT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
          ON TB1.C1=TB3.C1
```

위의 내용은 다음과 동일합니다.

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
     LEFT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
          ON TB1.C1=TB3.C1
```

내부 결합은 왼쪽 표의 각 행을 오른쪽 표의 모든 행과 결합하여 결합 조건이 참인 행만 보유합니다. 따라서 결과 표는 결합된 표 중 어느 하나 또는 둘 다에서 유실되는 행이 생길 수 있습니다. 외부 결합에는 외부 결합 유형에 따라 유실된 행 및 내부 결합에 의해 만들어진 행이 포함됩니다. 예외 결합에는 다음과 같이 예외 결합 유형에 따라 유실된 행만 포함됩니다.

- *왼쪽 외부*. 내부 결합에서 유실된 왼쪽 표의 행이 포함됩니다.
- *오른쪽 외부*. 내부 결합에서 유실된 오른쪽 표의 행이 포함됩니다.
- *왼쪽 예외*. 내부 결합에서 유실된 왼쪽 표의 행만이 포함됩니다.
- *오른쪽 예외*. 내부 결합에서 유실된 오른쪽 표의 행만이 포함됩니다.

결합된 표는 임의의 SELECT문 양식이 사용된 임의의 문맥에서 사용될 수 있습니다. SELECT문에 결합된 표가 포함되는 경우 뷰와 커서는 읽기 전용입니다.

결합 조건: *join-condition*은 다음의 규칙을 따르는 *탐색 조건*입니다.

- 수량화된 부속 조회, 부속 선택이 있는 IN 술어 또는 EXISTS 부속 조회는 포함할 수 없습니다. 기본적인 술부 하위 조회 및 스칼라 부속 선택을 포함할 수 있습니다.
- 각 열 이름은 *from절*의 표 중 하나에 있는 열을 분명하게 식별해야 합니다.

- 열 함수는 *표현식*에서 사용될 수 없습니다.

어떤 유형의 결합에 대해서든 join-condition 표현식의 열 참조는 해당 열이 어떤 표에 속해야 하는가에 대한 규칙이 적용되기 전에 108 페이지의 『열 이름』에 지정된 열 이름 규정자 분석 규칙을 사용하여 분석됩니다.

결합 연산: *join-condition*은 T1과 T2의 짝을 지정하며 T1과 T2는 *join-condition*의 JOIN 연산자의 왼쪽 및 오른쪽 피연산자 표입니다. T1과 T2 행의 모든 가능한 결합에 대하여 *join-condition*이 참인 경우 T1의 행은 T2의 행과 짝을 이룹니다. T1의 행이 T2의 행과 결합되면 결과 행은 T1의 행 값이 T2의 행 값과 연결되어 구성됩니다. 실행시 널(null) 행이 발생할 수도 있습니다. 표의 널(null) 행은 열의 널값 허용 여부에 상관없이 표의 각 열에 대한 널값으로 구성됩니다.

INNER JOIN 또는 **JOIN**

T1 INNER JOIN T2의 결과는 이들의 행이 짝을 이루어 구성됩니다.

*join-condition*이 있는 INNER JOIN 구문을 사용하면 쉼표로 분리된 FROM절에서 두 표를 나열하고 조건을 제공하기 위해 *where절*을 사용하여 결합을 지정하는 것과 동일한 결과가 나타납니다.

LEFT JOIN 또는 **LEFT OUTER JOIN**

T1 LEFT OUTER JOIN T2의 결과는 두 표의 짝을 이룬 행으로 구성되며 T1의 짝이 없는 행은 T2의 널(null) 행과 연결됩니다. T2에서 나온 모든 열에는 널값이 들어갈 수 있습니다.

RIGHT JOIN 또는 **RIGHT OUTER JOIN**

T1 RIGHT OUTER JOIN T2의 결과는 두 표의 짝을 이룬 행으로 구성되며 T2의 짝이 없는 행은 T1의 널(null) 행과 연결됩니다. T1에서 나온 모든 열에는 널값이 들어갈 수 있습니다.

LEFT EXCEPTION JOIN 및 **EXCEPTION JOIN**

T1 LEFT EXCEPTION JOIN T2의 결과는 T1의 짝이 없는 행이 T2의 널(null) 행과 연결되어 구성됩니다. T2에서 나온 모든 열에는 널값이 들어갈 수 있습니다.

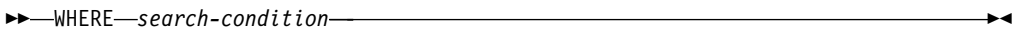
RIGHT EXCEPTION JOIN

T1 RIGHT EXCEPTION JOIN T2의 결과는 T2의 짝이 없는 행이 T1의 널(null) 행과 연결되어 구성됩니다. T1에서 나온 모든 열에는 널값이 들어갈 수 있습니다.

CROSS JOIN

T1 CROSS JOIN T2의 결과는 T1의 각 행이 T2의 각 행과 짝을 이루어 구성됩니다. CROSS JOIN을 Cartesian Product라고도 합니다.

where절



WHERE절은 *search-condition*이 참인 R의 행으로 구성되는 중간 결과표를 지정합니다. R은 명령문의 FROM절의 결과입니다.

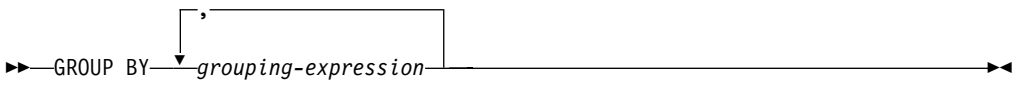
*search-condition*은 다음 규칙에 따라야 합니다.

- 각 *column-name*은 R의 열을 분명하게 식별하거나 상관 참조여야 합니다. *column-name*은 외부 부속 선택에서 식별된 표, 뷰, 공통 표 표현식 또는 파생된 표의 열을 식별하는 경우 상관 참조입니다.
- 열 함수는 WHERE절이 HAVING절의 부속 조회에 지정되지 않고 함수 인수가 그룹에 대한 상관 참조가 아닐 경우 지정되어서는 안됩니다.

WHERE절이 들어 있는 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 탐색 조건이 SBCS, 혼합 또는 UCS-2 자료를 갖는 술부를 포함하는 경우 술부에 대한 비교는 가중치 값을 사용하여 이루어집니다. 가중치 값은 술부의 피연산자에 정렬 순서를 적용하여 구합니다.

*search-condition*의 부속 조회는 R의 각 행에 대해 효율적으로 실행되며, 그 결과는 R의 주어진 행에 대한 *search-condition*의 어플리케이션에서 사용됩니다. 부속 조회는 R의 열에 대한 상관 참조를 포함하는 경우에만 실제로 R의 각 행에 대해 실행됩니다. 사실, 상관 참조가 없는 부속 조회는 단 한번만 실행되는 반면 상관 참조가 있는 부속 조회는 각 행에 대해 한번씩 실행되어야 합니다.

group-by절



GROUP BY절은 R의 행 그룹으로 이루어진 중간 결과표를 지정합니다. R은 이전 subselect절의 결과입니다.

가장 단순한 양식으로, GROUP BY절은 *grouping-expression*을 포함합니다. *grouping-expression*은 R의 그룹화를 지정할 때 사용되는 *표현식*입니다. *grouping-expression*에 포함된 각 column-name은 R의 열을 분명하게 식별해야 합니다. LOB 및 DataLink 열은 *grouping-expression*에서 사용될 수 없습니다. 각 *grouping-expression*의 길이 속성은 2000이상이어서는 안되며, 열이 널이 될 수 있는 경우 1999입니다. *grouping-expression*은 비 판별적 함수나 RRN, PARTITION, NODENAME 또는 NODENUMBER 함수를 포함할 수 없습니다.

GROUP BY절의 결과는 행 그룹 세트입니다. 한 행 이상의 각 그룹에서 각 *grouping-expression*의 값은 모두 같으며, *grouping-expression*의 동일한 세트 값을 갖는 모든 행은 동일한 그룹에 있습니다. 그룹화의 경우 *grouping-expression*에 대한 널 값은 모두 동일한 것으로 간주됩니다.

*HEX 이외의 정렬 순서는 GROUP BY절이 들어 있는 명령문이 실행될 때 유효하며, 행은 가중치 값을 사용하여 그룹으로 배치됩니다. 가중치 값은 정렬 순서를 SBCS 자료 *grouping-expression*, 혼합 자료 *grouping-expression*의 SBCS 자료 및 UCS-2 자료 *grouping-expression*에 적용하여 구합니다.

*grouping-expression*은 HAVING절의 탐색 조건, SELECT절 또는 ORDER BY절의 *sort-key-expression*에서 사용될 수 있습니다(자세한 내용은 order-by절을 참조하십시오). 어느 경우든 참조는 각 그룹에 대해 하나의 값만 지정합니다. 공백은 아무 의미가 없는 것을 제외하고, 이들 절에 지정된 *grouping-expression*은 GROUP BY절의 *grouping-expression*과 정확히 일치해야 합니다. 예를 들면,

```
SALARY*.10
```

의 *grouping-expression*은

```
HAVING SALARY*.10
```

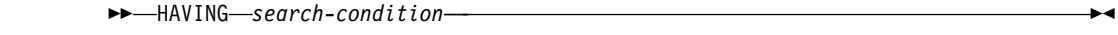
의 *having*절의 표현식과 일치하지만, 다음 절과는 일치하지 않습니다.

```
HAVING .10 *SALARY
```
또는
```
HAVING (SALARY*.10)+100
```

*grouping-expression*이 후행 공백이 있는 가변 길이 스트링을 포함하는 경우 그룹의 값들은 후행 공백의 수가 다를 수 있으며, 모두 동일한 길이를 갖지 않을 수 있습니다. 이때 *grouping-expression*에 대한 참조는 여전히 각 그룹에 대해 오직 하나의 값만 지정하지만, 그룹에 대한 값은 제공된 값 세트에서 임의로 선택됩니다. 따라서, 결과 값의 실제 길이는 예측할 수 없습니다.

GROUP BY절은 최대 120개의 *grouping-expression* 또는 2000 - n 바이트를 포함할 수 있는데, 여기서 n은 널(null)을 허용하는 지정된 *grouping-expression* 수입니다.

having절



having절

HAVING절은 *search-condition*이 참인 R의 그룹으로 구성되는 중간 결과표를 지정합니다. R은 subselect의 이전 구의 결과입니다. 이 절이 GROUP BY가 아닌 경우 R은 그룹화 표현식이 없는 단일 그룹으로 간주됩니다.

탐색 조건에 *column-name*을 포함하는 각 표현식은 다음 중 하나를 수행해야 합니다.

- R의 그룹화 표현식을 분명하게 식별합니다.
- 열 함수 내에서 지정되어야 합니다.
- 상관 참조여야 합니다. *column-name*은 외부 부속 선택에서 식별된 표, 뷰, 공통 표 표현식 또는 파생된 표의 열을 식별하는 경우 상관 참조입니다.

RRN, PARTITION, NODENAME 및 NODENUMBER 함수는 열 함수 내에 있는 경우가 아니면 HAVING절에서 지정될 수 없습니다. 열 함수 사용에 적용되는 제한사항은 제 3 장의 ″함수″를 참조하십시오.

HAVING절이 들어 있는 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 탐색 조건이 SBCS, 혼합 또는 UCS-2 자료를 갖는 술부를 포함하는 경우 술부에 대한 비교는 가중치 값을 사용하여 이루어집니다. 가중치 값은 술부의 피연산자에 정렬 순서를 적용하여 구합니다.

탐색 조건이 적용되는 R 그룹은 탐색 조건의 각 열 함수에 대해 인수를 제공하며 인수가 상관 참조인 함수의 경우는 예외입니다.

각 탐색 조건이 부속 조회를 포함하는 경우 부속 조회는 탐색 조건이 R 그룹에 적용될 때마다 실행되는 것으로 간주될 수 있으며 그 결과는 탐색 조건을 적용할 때 사용됩니다. 실제로, 부속 조회는 상관 참조를 포함하는 경우에만 각 그룹에 대해 실행됩니다. 차이점에 대한 설명은 『subselect의 예』의 예 6 및 7을 참조하십시오.

그룹 R에 대한 상관 참조는 그룹화 열을 식별하거나 열 함수 내에 포함되어야 합니다.

GROUP BY 없이 HAVING이 사용될 때 선택 리스트의 열 이름은 열 함수 내에 나타나야 합니다.

subselect의 예

예 1

EMPLOYEE 표에서 모든 열과 행을 선택합니다.

```
SELECT * FROM EMPLOYEE
```

예 2

EMPPROJACT와 EMPLOYEE 표를 결합하고 EMPPROJACT 표에서 모든 열을 선택한 후 EMPLOYEE 표의 사원 성(LASTNAME)을 결과의 각 행에 추가합니다.

```
SELECT EMPPROJACT.*, LASTNAME
  FROM EMPPROJACT, EMPLOYEE
  WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO
```

예 3

EMPLOYEE 및 DEPARTMENT 표를 결합하고, 사원 번호(EMPNO), 사원 성(LASTNAME), 부서명(EMPLOYEE 표의 WORKDEPT, DEPARTMENT 표의 DEPTNO), 그리고 1930년 이전에 태어난(BIRTHDATE) 모든 사원의 부서명(DEPTNAME)을 선택합니다.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
  FROM EMPLOYEE, DEPARTMENT
  WHERE WORKDEPT = DEPTNO
  AND YEAR(BIRTHDATE) < 1930
```

이 subselect는 다음과 같이 작성될 수도 있습니다.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
  FROM EMPLOYEE INNER JOIN DEPARTMENT
  ON WORKDEPT = DEPTNO
  WHERE YEAR(BIRTHDATE) < 1930
```

예 4

작업(JOB)을 선택하고 EMPLOYEE 표에서 동일한 작업 코드를 갖는 각 행 그룹에 대해 최소 및 최대 급여(SALARY)를 선택합니다. 그러나, 행이 하나 이상이고 최대 급여가 27000 이상인 그룹에 대해서만 선택합니다.

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
       FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1 AND MAX(SALARY) >= 27000
```

예 5

부서(WORKDEPT) 'E11'의 사원(EMPNO)에 대하여 EMPPROJACT 표의 모든 행을 선택합니다(사원 부서 번호는 EMPLOYEE 표에 나타납니다).

```
SELECT * FROM EMPPROJACT
  WHERE EMPNO IN (SELECT EMPNO
                  FROM EMPLOYEE
                  WHERE WORKDEPT = 'E11')
```

예 6

EMPLOYEE 표에서 최대 급여가 모든 사원에 대한 평균 급여 미만인 모든 부서에 대한 부서 번호(WORKDEPT) 및 최대 부서 급여(SALARY)를 선택합니다.

```
SELECT WORKDEPT, MAX(SALARY)
       FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
       FROM EMPLOYEE)
```

HAVING절에서 부속 조회는 이 예에서 단 한번만 실행됩니다.

예 7

EMPLOYEE 표를 사용하여 최대 급여가 모든 다른 부서의 평균 급여 미만인 모든 부서에 대한 부서 번호(WORKDEPT) 및 최대 부서 급여(SALARY)를 선택합니다.

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE EMP_COR
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
       FROM EMPLOYEE
       WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

예 6과 대조적으로, HAVING절에서 부속 조회는 각 그룹에 대해 실행될 필요가 있습니다.

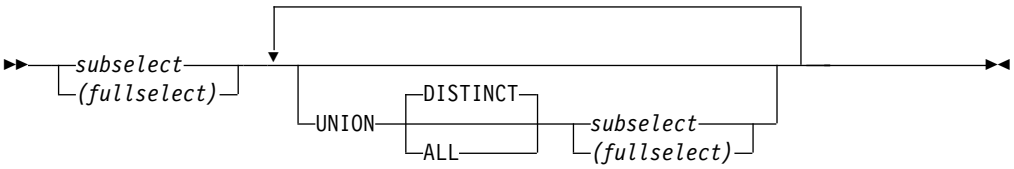
예 8

EMPLOYEE와 EMPPROJACT 표를 결합한 후 모든 사원 및 이들의 프로젝트 번호를 선택합니다. 현재 할당된 프로젝트 번호가 없는 사원들도 리턴합니다.

```
SELECT EMPLOYEE.EMPNO, PROJNO
  FROM EMPLOYEE LEFT OUTER JOIN EMPPROJACT
  ON EMPLOYEE.EMPNO = EMPPROJACT.EMPNO
```

EMPPROJACT 표에 프로젝트 번호가 없는 EMPLOYEE 표의 사원은 결과 표에 EMPNO 값과 PROJNO 열의 널값이 들어 있는 한 행을 리턴합니다.

fullselect



*fullselect*는 결과표를 지정합니다. UNION이 사용되지 않은 경우 fullselect의 결과는 지정된 subselect의 결과입니다.

UNION DISTINCT or **UNION ALL**

두 개의 다른 결과표(R1 및 R2)를 결합하여 결과표를 구합니다. UNION ALL이 지정된 경우 결과는 R1 및 R2의 모든 행으로 이루어집니다. ALL 옵션 없이 UNION이 지정된 경우 결과는 중복 행이 제거된 R1이나 R2에 있는 모든 행 세트입니다. 그러나, 두 경우에서 UNION 표의 각 행은 R1의 행 또는 R2의 행입니다.

결과 열은 다음과 같이 명명됩니다.

- R1의 n 번째 열과 R2의 n 번째 열이 동일한 결과 열 이름을 갖는 경우 결과표의 n 번째 열은 해당 열 이름을 갖습니다.
- R1의 n 번째 열과 R2의 n 번째 열이 동일한 이름을 갖지 않는 경우 결과표의 n 번째 열은 명명되지 않습니다.

첫 번째 행의 각 값이 두 번째 행의 대응값과 같은 경우 두 행은 중복됩니다. UNION 키워드가 들어 있는 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 결과표가 SBCS, UCS-2 또는 혼합 자료를 갖는 열을 포함하는 경우 이러한 열에 대한 비교는 가중치 값을 사용하여 이루어집니다. 가중치 값은 각 값에 정렬 순서를 적용하여 구합니다(중복을 판별하기 위해 두 널값은 동일한 것으로 간주됩니다).

UNION과 UNION ALL은 모두 연관 조작입니다. UNION 연산자와 동일한 SQL문에 UNION ALL 연산자를 포함하는 경우 조작 결과는 평가 순서에 따라 달라집니다. 괄호가 없는 곳에서 평가는 왼쪽에서 오른쪽으로 이루어집니다. 괄호가 있는 곳에서는 괄호 속의 subselect가 먼저 평가되고, 다음에 왼쪽에서 오른쪽으로 명령문의 다른 구성요소에 의해 평가됩니다.

열에 대한 규칙: R1 및 R2는 동일한 수의 열을 가져야 하며, R1의 n 번째 자료 유형은 R2의 n 번째 자료 유형과 호환될 수 있어야 합니다. 문자 스트링 값은 날짜 시간 값과 호환될 수 없습니다.

UNION 및 UNION ALL 결과의 n 번째 열은 R1 및 R2의 n 번째 열에서 파생됩니다. 결과 열의 속성은 결과 열에 대한 규칙을 사용하여 판별됩니다. 자세한 내용은 93 페이지의 『결과 자료 유형에 대한 규칙』을 참조하십시오.

UNION이 지정된 경우 열은 LOB 또는 DATALINK 열이 될 수 없습니다.

괄호 안에 사용된 fullselect를 *부속 조회*라고 합니다. 예를 들어, *부속 조회*를 탐색 조건에 사용할 수 있습니다.

fullselect의 예

예 1

EMPLOYEE 표에서 모든 열과 행을 선택합니다.

```
SELECT * FROM EMPLOYEE
```

예 2

EMPLOYEE 표의 모든 사원들 중에서 부서 번호(WORKDEPT)가 'E'로 시작되거나 또는 EMPPROJACT 표에서 프로젝트 번호(PROJNO)가 'MA2100', 'MA2110' 또는 'MA2112'인 프로젝트에 배정된 사원의 번호(EMPNO)를 모두 나열합니다.

```
SELECT EMPNO FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

예 3

예 2에서와 같은 조회를 수행하며 EMPLOYEE 표에서 가져온 행에는 'emp', EMPPROJACT 표에서 가져온 행에는 'empprojact' 태그를 붙입니다. 예2의 결과와는 달리 이 조회는 같은 어떤 표에서 가져온 것을 태그로 식별하고 EMPNO를 두 번 이상 리턴할 수 있습니다.

```
SELECT EMPNO, 'emp' FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
  UNION
SELECT EMPNO, 'empprojact' FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

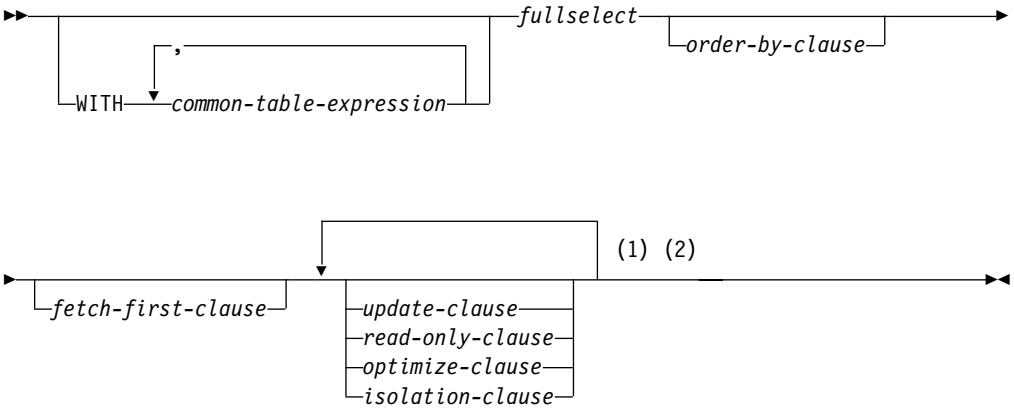
예 4

예 2에서와 동일한 조회를 합니다. UNION ALL만을 사용하여 중복 행이 제거되지 않도록 합니다.

```
SELECT EMPNO FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

---

select문



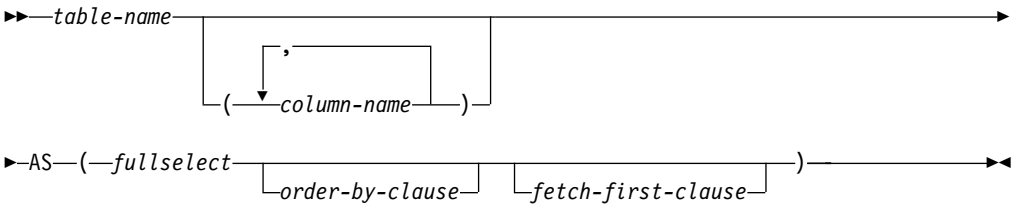
주:

1 update절 및 read-only절이 모두 동일한 select문에서 지정될 수 없습니다.

2 각 절은 한 번만 지정됩니다.

*select*문는 DECLARE CURSOR문에서 직접 지정되거나 준비된 후 DECLARE CURSOR문에서 참조될 수 있는 조회 양식입니다. 이 명령문은 결과표가 사용자의 워크스테이션에 표시되도록 대화 함수(STRSQL 명령)을 사용하여 대화식으로 발행될 수도 있습니다. 두 경우 모두 *select-statement*에 의해 지정된 표는 fullselect의 결과입니다.

common-table 표현식



*common-table-expression*은 이어지는 fullselect의 FROM절의 표 이름으로 지정될 수 있는 *table-name*으로 결과표를 정의할 수 있도록 허용합니다. *table-name*은 규정되지 않아야 합니다. 단일 WITH 키워드 다음에 복수 공통 표 표현식이 지정될 수 있습니다. 지정된 각 공통 표 표현식은 이어지는 공통 표 표현식의 FROM절에 있는 이름으로 참조될 수도 있습니다.

열 리스트가 지정된 경우 리스트는 전체 선택 결과 표의 열 수와 같은 수의 이름으로 구성되어야 합니다. 각 *column-name*은 고유해야 하며 규정되지 않아야 합니다. 이들 열 이름이 지정되지 않은 경우 이름은 공통 표 표현식을 정의하는 데 사용된 subselect의 선택 리스트에서 가져 옵니다.

공통 표 표현식의 *table-name*은 동일한 명령문의 다른 공통 표 표현식 table-name과 달라야 합니다. 공통 표 표현식 *table-name*은 fullselect를 통해 FROM절의 표 이름으로 지정될 수 있습니다. 공통 표 표현식의 *table-name*은 기존 표, 뷰 또는 동일한 규정된 이름을 갖는 별명(카탈로그에 있는)을 대체합니다.

하나 이상의 공통 표 표현식이 동일한 명령문에 정의된 경우 공통 표 표현식 사이의 순회 참조는 허용되지 않습니다. *cyclic reference*는 두 개의 공통 표 표현식 *dt1* 및 *dt2*가, *dt1*이 *dt2*를 참조하고 *dt2*가 *dt1*을 참조하도록 작성될 때 발생합니다.

*common-table-expression*은 CREATE VIEW 및 INSERT문에서의 전체 선택 앞에서 선택적입니다.

*common-table-expression*은 다음과 같은 경우에 사용될 수 있습니다.

- 뷰 작성을 피하기 위해 뷰 대신에(일반적인 뷰 사용이 필요하지 않고 위치지정된 갱신 또는 삭제가 사용되지 않을 때)
- 원하는 결과표가 호스트 변수를 기준으로 할 때

- 동일한 결과표를 *fullselect*에서 공유해야 할 때

공통 표 표현식의 *fullselect*가 FROM절에서 자체 참조를 포함하면 공통 표 표현식은 *순환 표 표현식*입니다. 순환 공통 표 표현식은 iSeries용 DB2 UDB에서 지원되지 않습니다.

order-by절

sort-key:

ORDER BY절은 결과표의 행 순서를 지정합니다. 단일 *sort-key*가 식별되면 행은 해당 *sort-key* 값에 의해 순서화됩니다. 하나 이상의*sort-key*가 식별되면 행은 첫 번째 식별된 *sort-key* 값에 의해 순서화된 다음 두 번째 식별된 *sort-key* 값에 의해 순서화됩니다.

ORDER BY구가 들어 있는 명령문이 실행될 때 *HEX 이외의 정렬 순서가 유효하고 ORDER BY절이 SBCS, UCS-2 또는 혼합 자료를 갖는 *sort-keys*를 포함하는 경우 *sort-keys*에 대한 비교는 가중치 값을 통해 이루어집니다. 가중치 값은 *sort-keys* 값에 정렬 순서를 적용하여 구합니다.

선택 리스트의 명명된 열은 *simple-integer* 또는 *simple-column-name*인 *sort-key*에 의해 식별될 수 있습니다. 선택 리스트의 명명되지 않은 이름은 *simple-integer* 또는 *sort-key-expression*에 의해 식별될 수 있습니다. AS절이 select-list에서 지정되지 않고 상수, 연산자가 있는 표현식 또는 함수에서 파생된 경우 열은 명명되지 않습니다. fullselect가 UNION 연산자를 포함하는 경우 fullselect에서 명명된 열에 대한 규칙은 350 페이지의 『fullselect』를 참조하십시오.

simple-column-name

일반적으로 결과표의 열을 식별합니다. 이 경우, *simple-column-name*은 선택 리스트에 이름이 지정된 열의 이름이어야 합니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. fullselect가 UNION 또는 UNION ALL을 포함하는 경우 열 이름은 규정될 수 없습니다.

*simple-column-name*은 조회가 부속 선택인 경우 FROM절에서 식별된 표, 뷰 또는 내포된 표의 열 이름을 식별할 수도 있습니다. 부속 선택이 그룹화된 결과를 작성하고 *simple-column-name*이 *grouping-expression*이 아니면 오류가 발생합니다.

integer

0 보다 커야 하며, 결과표의 열 수보다 커서는 안됩니다. 정수 *n*은 결과표의 *n*번째 열을 식별합니다. 식별된 열은 LOB 또는 DATALINK 열이어서는 안됩니다.

sort-key-expression

단순히 열 이름 또는 부호 없는 정수 상수가 아닌 표현식. 이 형식의 *sort-key*를 사용하려면 순서화가 적용되는 조회는 subselect이어야 합니다.

*fullselect*가 UNION 또는 UNION ALL을 포함하는 경우 *sort-key-expression*은 RRN, PARTITION, NODENAME 또는 NODENUMBER를 포함할 수 없습니다. *sort-key-expression* 결과는 LOB 또는 DATALINK가 되어서는 안됩니다.

subselect가 그룹화된 경우 *sort-key-expression*은

- subselect의 선택 리스트에 있는 표현식이 될 수 있습니다.
- subselect의 GROUP BY절의 *grouping-expression*을 포함할 수 있습니다.

ASC

오름차순으로 된 열 값을 사용합니다. 이것이 디폴트 값입니다.

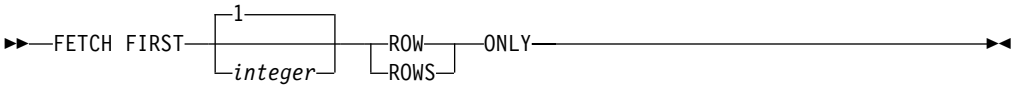
DESC

내림차순으로 된 열 값을 사용합니다.

순서화는 제 2 장에서 설명한 비교 규칙에 따라 수행됩니다. 널값은 다른 값보다 커야 합니다. 순서화 스펙이 완전한 순서화를 판별하지 못하면 마지막 식별된 *sort-key*의 중복값을 갖는 행은 임의 순서를 갖습니다. ORDER BY절이 지정되지 않은 경우 결과표의 행은 임의 순서를 갖습니다.

ORDER BY절은 최대 10000-n *sort-key* 또는 10000-n 바이트를 포함할 수 있습니다 (여기서, n은 널(null)을 허용하는 지정된 *sort-key* 수입니다).

fetch-first절



*fetch-first*절은 검색할 수 있는 최대 행 수를 설정합니다. 그러면 데이터베이스 관리자는 이 절이 지정되지 않았을 때 결과 표에 몇 개의 행이 있는지에 관계없이 어플리케

이션에서 *정수*개의 행을 초과해서는 검색하지 않으려 함을 알 수 있습니다. *정수*개의 행 이상을 페치하려는 시도는 자료의 정상 종료와 같은 방식으로 처리됩니다. *정수* 값은 (0이 아닌) 양의 정수여야 합니다.

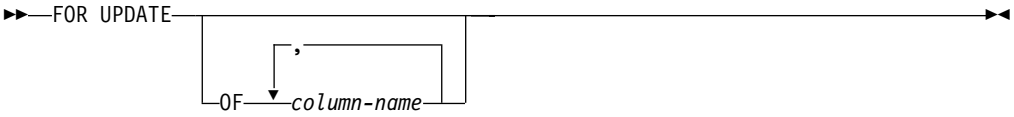
결과표를 첫 번째 *정수* 행으로 제한하면 성능이 향상될 수 있습니다. 데이터베이스 관리자는 첫 번째 *integer* 행을 판별하면 조회 처리를 중단합니다.

select문에 *fetch-first-clause*을 지정하면 결과표가 읽기 전용이 됩니다. 읽기 전용 결과표는 UPDATE나 DELETE문에서 참조되어서는 안됩니다.

*fetch-first-clause*은 UPDATE절이 들어 있는 명령문에 나올 수 없습니다.

*fetch-first-clause*과 *order-by-clause*이 모두 지정된 경우 첫 번째 *integer*의 행을 리턴하기 전에 전체 결과 세트에 대한 정렬이 수행됩니다.

update절



UPDATE절은 뒤에 나오는 UPDATE문에서 갱신될 수 있는 열을 식별합니다. 각 *column-name*은 규정되지 않아야 하며, 전체 선택의 첫 번째 FROM절에서 식별된 표 또는 뷰의 열을 식별해야 합니다. UPDATE절이 열 이름없이 지정된 경우 fullselect의 첫 번째 FROM절에서 식별된 표 또는 뷰의 갱신가능한 열이 모두 포함됩니다. 절은 동반되는 optimize 절 앞 또는 뒤에 나타날 수 있습니다.

fullselect의 결과표가 읽기 전용이거나(자세한 내용은 598 페이지의 『DECLARE CURSOR』 참조), FOR READ ONLY절이 사용된 경우 또는 DECLARE CURSOR 문에서 DYNAMIC 키워드 없이 SCROLL 키워드가 지정된 경우 FOR UPDATE OF 절은 지정되어서는 안됩니다.

select문와 연관된 커서를 식별하는 위치 지정된 UPDATE문은 다음과 같은 경우 갱신가능한 열을 모두 갱신할 수 있습니다.

- select문이 다음 중 하나를 포함하지 않는 경우
  - UPDATE절
  - FOR READ ONLY절
  - ORDER BY절
- DECLARE CURSOR문이 DYNAMIC 키워드 없이 SCROLL 키워드를 포함하지 않는 경우

UPDATE절은 ISO/ANSI SQL의 일부가 아닌 성능 옵션입니다.

read-only절

FOR READ ONLY 또는 FOR FETCH ONLY절은 결과표가 읽기 전용임을 표시합니다. 예를 들면, 커서는 위치지정된 DELETE 또는 UPDATE문에 사용되지 않습니다.

일부 결과표는 본래 읽기 전용입니다(예: 읽기 전용 뷰를 바탕으로 한 표). FOR READ ONLY는 이런 표에 대해 지정될 수는 있지만, 스펙은 영향을 미치지 않습니다.

갱신 및 삭제가 허용되는 결과표에 대해 FOR READ ONLY를 지정하면 데이터베이스 관리자가 블로킹을 수행하여 배타적 잠금을 피할 수 있기 때문에 FETCH 조작 성능을 향상시킬 수 있습니다. 예를 들면, FOR READ ONLY 또는 ORDER BY절이 없는 동적 SQL문을 포함하는 프로그램에서 데이터베이스 관리자는 FOR UPDATE OF 절이 지정된 것처럼 DYNAMIC 키워드 없이 SCROLL을 지정하지 않은 커서를 열 수 있습니다.

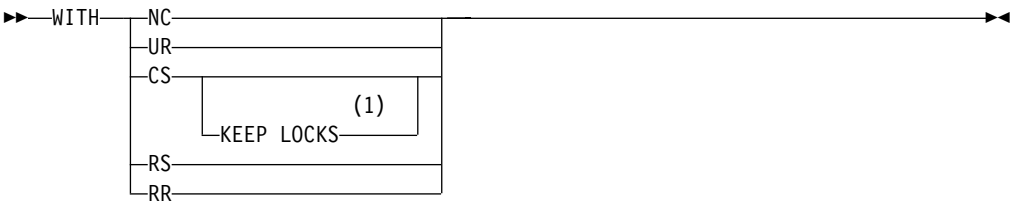
읽기 전용 결과표는 본래 읽기 전용이건 FOR READ ONLY로 지정되었건 UPDATE 또는 DELETE문에서 참조되어서는 안됩니다.

FOR READ ONLY절은 UPDATE절이 들어 있는 명령문에 나타날 수 없습니다. 절은 동반되는 optimize 절 앞 또는 뒤에 나타날 수 있습니다.

optimize절

optimize절은 데이터베이스 관리자에게 프로그램이 결과표의 *integer* 행 이상을 검색하지 않는다고 가정한다는 것을 알려줍니다. 이 절이 없거나 또는 키워드 ALL이 있는 경우 데이터베이스 관리자는 결과표의 모든 행이 검색된다고 가정하고 따라서 최적화합니다. *integer* 행에 대해 또는 최소한 페치된 행에 대해 최적화를 수행하면 성능을 향상시킬 수 있습니다. 이 절은 결과표 또는 행이 페치된 순서를 변경할 수 없습니다. 임의 수의 행을 페치할 수 있지만, 성능은 *integer*번 페치한 다음 저하될 수 있습니다. 정수 값은(0이 아닌) 양의 정수여야 합니다. 이 절은 동반되는 update절 또는 read-only 절 앞 또는 뒤에 나타날 수 있습니다.

isolation절



주:

1 KEEP LOCKS절은 DECLARE CURSOR, SELECT INTO 및 fullselect문에서만 사용될 수 있습니다.

isolation절은 SELECT, SELECT INTO, INSERT, UPDATE, DELETE, 및 DECLARE CURSOR문에서 분리 레벨을 지정합니다. 이 분리 레벨은 분리절이 들어 있는 명령문 실행시에만 적용됩니다. 분리 레벨에 대한 자세한 정보는 분리 레벨을 참조하십시오.

KEEP LOCKS절은 확보한 임의의 읽기 잠금을 더 오래 유지하도록 지정합니다. 보통, 읽기 잠금은 다음 행이 읽혀질 때 해제됩니다. 분리절이 커서와 관련된 경우 커서가 닫히거나 COMMIT 또는 ROLLBACK 명령문이 실행될 때까지 잠금을 유지합니다. 그렇지 않은 경우 잠금은 SQL문 완료시까지 유지됩니다. KEEP LOCKS절은 SQL SELECT, SELECT INTO 또는 DECLARE CURSOR문에서 허용됩니다. 갱신가능한 커서에서는 허용되지 않습니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 NONE은 NC에 대한 동의어로 사용될 수 있습니다.
- 키워드 CHG은 UR에 대한 동의어로 사용될 수 있습니다.
- 키워드 ALL은 RS에 대한 동의어로 사용될 수 있습니다.

select문 예

예 1

EMPLOYEE 표에서 모든 열과 행을 선택합니다.

```
SELECT * FROM EMPLOYEE
```

예 2

PROJECT 표에서 프로젝트명(PROJNAME), 시작 날짜(PRSTDATE), 종료 날짜(PRENDATE)를 선택합니다. 가장 최근의 날짜가 먼저 나타나도록 종료 날짜순으로 결과표를 배열합니다.

```
SELECT PROJNAME, PRSTDATE, PRENDATE
  FROM PROJECT
  ORDER BY PRENDATE DESC
```

예 3

EMPLOYEE 표에서 모든 부서에 대한 평균 부서 급여(SALARY)와 부서 번호(WORKDEPT)를 선택합니다. 결과표를 평균 부서 급여에 따라 오름차순으로 배열합니다.

```
SELECT WORKDEPT, AVG(SALARY)
       FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY 2
```

예 4

PROJECT 표의 시작 날짜(PRSTDATE)와 종료 날짜(PRENDATE)를 갱신하는 C 프로그램에서 사용될 UP_CUR이라는 커서를 선언합니다. 프로그램은 각 행에 대해 프로젝트 번호(PROJNO)와 함께 이 값 모두를 수신해야 합니다. 선언은 조회에 대한 액세스 경로가 최대 2 행 검색을 위해 최적화되도록 지정합니다. 이 경우에도, 프로그램은 결과표에서 2 행 이상을 검색할 수 있습니다. 그러나, 2 행 이상이 검색될 때는 성능이 저하될 수 있습니다.

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
              FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE
              OPTIMIZE FOR 2 ROWS ;
```

예 5

이 예에서는 표현식 SAL+BONUS+COMM을 TOTAL_PAY라고 명명합니다.

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY
  FROM EMPLOYEE
  ORDER BY TOTAL_PAY
```

예 6

평균 급여 및 해당 부서의 인원과 함께 사원 번호 및 영업대표의 급여를 판별합니다. 또한 가장 높은 평균 급여와 함께 부서의 평균 급여를 나열합니다.

이 경우 공통 표 표현식을 사용하면 일반 뷰로 DINFO 뷰를 작성하는 오버헤드를 줄일 수 있습니다. 전체 선택 나머지의 문맥으로 인해 영업 대표의 부서에 대한 행만이 뷰에서 고려해야 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

반복가능한 읽기(RS, ALL) 분리 레벨을 갖는 표에서 항목을 선택합니다.

```
SELECT NAME, SALARY
  FROM PAYROLL
  WHERE DEPT = 704
  WITH RS
```

제 5 장 명령문

이 장에는 구문 도표, 의미 설명, 규칙 및 다음 표에 나열된 SQL문의 사용 예가 포함되어 있습니다.

표 32. SQL 스키마 명령문

SQL문	**설명**	**페이지**
ALTER TABLE	표의 설명 변경	371
COMMENT	별명, 열, 함수, 색인, 패키지, 매개변수, 프로시듀어, 표, 유형이나 뷰의 설명에 대한 주석을 대체하거나 추가	408
CREATE ALIAS	별명 작성	432
CREATE DISTINCT TYPE	고유한 유형 작성	435
CREATE FUNCTION	사용자 정의 함수 작성	443
CREATE FUNCTION(외부 스칼라)	외부 스칼라 함수 작성	447
CREATE FUNCTION(외부 스칼라)	외부 표 함수 작성	465
CREATE FUNCTION(피소스(sourced))	기존의 스칼라나 열 함수를 바탕으로 사용자 정의 함수 작성	480
CREATE FUNCTION(SQL 스칼라)	SQL 스칼라 함수 작성	488
CREATE FUNCTION(SQL 표)	SQL 표 함수 작성	497
CREATE INDEX	표에 색인 작성	506
CREATE PROCEDURE	프로시듀어 작성	510
CREATE PROCEDURE(외부)	외부 프로시듀어 작성	512
CREATE PROCEDURE (SQL)	SQL 프로시듀어 작성	526
CREATE SCHEMA	스키마 및 그 스키마의 오브젝트 세트 작성	536
CREATE TABLE	표 작성	541

표 32. SQL 스키마 명령문 (계속)

SQL문	설명	페이지
CREATE TRIGGER	트리거 작성	575
CREATE VIEW	하나 이상의 표의 뷰나 복수 뷰 작성	590
DROP	별명, 함수, 색인, 패키지, 프로시듀어, 스키마, 표, 트리거, 유형 또는 뷰 삭제	656
GRANT(고유한 유형 권한)	고유한 유형에 권한 부여	684
GRANT(함수 또는 프로시듀어 권한)	함수나 프로시듀어에 권한 부여	687
GRANT(패키지 권한)	패키지에 권한 부여	695
GRANT(표 권한)	표나 뷰에 권한 부여	698
LABEL	별명, 열, 패키지, 표나 뷰의 설명에 레이블을 추가하거나 대체	717
RENAME	표, 뷰 또는 색인 재명명	741
REVOKE(고유한 유형 권한)	권한을 취소하여 고유한 유형 사용	744
REVOKE(함수 또는 프로시듀어 권한)	함수나 프로시듀어의 권한 취소	746
REVOKE(패키지 권한)	권한을 취소하여 패키지에서 명령문 실행	753
REVOKE(표 권한)	표나 뷰의 권한 취소	755

표 33. SQL 자료 변경 명령문

SQL문	설명	페이지
DELETE	표에서 하나 이상의 행 제거	639
INSERT	표에 하나 이상의 행 삽입	708
UPDATE	표에 있는 하나 이상의 행의 하나 이상의 열 값을 갱신	804

표 34. SQL 자료 명령문

SQL문	설명	페이지
	모든 SQL 자료 변경 명령문	표 33
CLOSE	커서 닫기	406
DECLARE CURSOR	SQL 커서 정의	598
FETCH	결과표 행에서 커서의 위치지정. 결과표의 하나 이상의 행의 값을 호스트 변수로 지정할 수도 있습니다.	675
FREE LOCATOR	LOB 로케이터 변수와 값 사이의 연관 제거	683
HOLD LOCATOR	LOB 로케이터 변수가 작업 단위를 넘어서는 값과 연관을 보유하도록 합니다.	704

표 34. SQL 자료 명령문 (계속)

SQL문	설명	페이지
LOCK TABLE	동시 프로세스가 표를 변경하거나 사용하지 못하게 함	721
OPEN	커서 열기	723
SELECT	조회 실행	765
SELECT INTO	값을 호스트 변수에 지정	766
SET 이전 변수	이전 변수에 값을 지정	798
SET 변수	값을 호스트 변수에 지정	801
VALUES	트리거에서 사용자 정의 함수 호출 방법 제공	814
VALUES INTO	한 행만의 결과표를 지정하고 호스트 변수에 값 지정	816

표 35. SQL 트랜잭션 명령문

SQL문	설명	페이지
COMMIT	작업 단위를 종료하고 해당 작업 단위로 수행한 데이터베이스 변경 내용을 확약(commit)	418
RELEASE SAVEPOINT	출시 내에 저장점 릴리스	740
ROLLBACK	작업 단위를 종료하고 해당 작업 단위로 작성한 데이터베이스 변경을 역처리함	759
SAVEPOINT	작업 단위 내에 저장점 설정	763
SET TRANSACTION	현재 작업 단위에 대한 분리 레벨 변경	795

표 36. SQL 연결 명령문

SQL문	설명	페이지
CONNECT(유형 1)	서버에 연결하고 리모트 작업 단위에 대한 규칙 설정	421
CONNECT(유형 2)	서버에 연결하고 어플리케이션 지향 분산 작업 단위에 대한 규칙 설정	427
DISCONNECT	하나 이상의 연결을 즉시 종료	653
RELEASE	하나 이상의 연결을 릴리스 지연 중에 위치시킴	738
SET CONNECTION	기존 연결 기존 연결 중 하나를 식별하여 프로세스의 서버 설정	769

표 37. SQL 동적 명령문

SQL문	설명	페이지
DESCRIBE	준비된 명령문의 결과 열 설명	645
DESCRIBE TABLE	표나 뷰에 대한 정보 획득	650

표 37. SQL 동적 명령문 (계속)

SQL문	설명	페이지
EXECUTE	준비된 SQL문 실행	670
EXECUTE IMMEDIATE	SQL문 준비 및 실행	673
PREPARE	SQL문 실행 준비	728

표 38. SQL 세션 명령문

SQL문	설명	페이지
DECLARE GLOBAL TEMPORARY TABLE	선언된 글로벌 임시 표를 정의	606
SET PATH	CURRENT PATH 특수 레지스터에 값을 할당합니다.	788
SET SCHEMA	CURRENT SCHEMA 특수 레지스터에 값을 할당합니다.	793

표 39. SQL 내장 호스트 언어 명령문

SQL문	설명	페이지
BEGIN DECLARE SECTION	SQL 선언 섹션의 시작 표시	398
DECLARE PROCEDURE	외부 프로시듀어 정의	624
DECLARE STATEMENT	준비된 SQL문을 식별하는 데 사용되는 이름 선언	634
DECLARE VARIABLE	호스트 변수의 디폴드 값 외에 부속 유형이나 코드화 문자 세트 ID(CCSID) 선언	636
END DECLARE SECTION	SQL 선언 섹션의 끝 표시	668
INCLUDE	소스 프로그램에 선언 삽입	706
SET RESULT SET	프로시듀어에서 결과 세트 식별	790
SET OPTION	SQL문을 처리하는 옵션 설정	772
WHENEVER	SQL 리턴 코드를 기준으로 취할 조치 정의	819

표 40. SQL 제어문

SQL문	설명	페이지
assignment문	출력 매개변수 또는 로컬 변수에 값을 지정합니다.	826
CALL	프로시듀어 호출	400

표 40. SQL 제어문 (계속)

SQL문	설명	페이지
CASE	복수 조건을 기준으로 실행 경로를 선택합니다.	829
compound문	SQL 루틴에서 다른 명령문을 함께 그룹화합니다.	831
FOR	표의 각 행에 대해 명령문을 실행합니다.	838
GET DIAGNOSTICS	실행된 이전 SQL문에 관한 정보를 구합니다.	840
GOTO	SQL 루틴 또는 트리거에서 사용자 정의 레이블로 분기하는 데 사용됩니다.	843
IF	조건의 참 값을 기준으로 한 조건부 실행을 제공합니다.	845
ITERATE	제어흐름이 레이블이 설정된 루프 시작 부분으로 리턴되도록 합니다.	847
LEAVE	블록 또는 루프를 나가서 계속 실행합니다.	849
LOOP	명령문의 실행을 반복합니다.	851
REPEAT	명령문의 실행을 반복합니다.	853
RESIGNAL	오류 또는 경고 조건을 다시 신호합니다.	855
RETURN	루틴에서 리턴합니다.	858
SIGNAL	오류 또는 경고 조건을 신호합니다.	861
WHILE	지정된 조건이 참인 동안 명령문 실행을 반복합니다.	864

또한 참조하십시오.

- 『SQL문을 호출하는 방법』
- 368 페이지의 『SQL 리턴 코드』
- 370 페이지의 『SQL 주석』

SQL문을 호출하는 방법

이 장에 설명된 SQL문은 *실행* 또는 *비실행*으로 분류됩니다. 각 명령문의 설명에 있는 호출 섹션은 명령문의 실행가능 여부를 표시합니다.

다음과 같은 방법으로 *실행*문을 호출할 수 있습니다.

- 어플리케이션 프로그램에 삽입
- 동적으로 준비 및 실행
- 대화식으로 발행

주: REXX에 삽입되거나 RUNSQLSTM을 사용하여 처리된 명령문은 동적으로 준비되고 실행됩니다.

명령문에 따라 다음 메소드를 일부 또는 전부 사용할 수 있습니다. 각 명령문의 설명에 있는 호출 섹션은 사용할 수 있는 메소드를 표시합니다.

*비실행*문은 어플리케이션 프로그램에만 삽입될 수 있습니다.

이 장에 설명된 명령문 외에 *select*문이라는 SQL문 구조가 더 있습니다. 352 페이지의 『select문』을 참조하십시오. 이 구조는 다른 명령문에서 다른 방식으로 사용되기 때문에 이 장에는 포함되지 않습니다.

다음과 같은 방법으로 *select*문을 호출할 수 있습니다.

- DECLARE CURSOR에 포함되고 OPEN에 의해 명시적으로 실행
- DECLARE CURSOR를 참조하여 동적으로 준비되고 OPEN에 의해 명시적으로 실행
- 대화식으로 발행

처음의 두 가지 메소드를 각각 *select*문의 *정적* 및 *동적* 호출이라고 합니다.

어플리케이션 프로그램에 명령문 삽입

CRTSQLCBL, CRTSQLCBLI, CRTSQLCI, CRTSQLFTN, CRTSQLCPPI, CRTSQLPLI, CRTSQLRPG, CRTSQLRPG 또는 CRTSQLRPGI 명령을 사용하여 SQL문이 사전컴파일러에 입력될 소스 프로그램에 포함될 수 있습니다. 이러한 명령문을 프로그램에 *삽입*되었다고 합니다. 프로그램에서 호스트 언어가 허용되는 곳이면 어느 곳이나 삽입된 명령문이 있을 수 있습니다. 삽입된 명령문은 모두 EXEC와 SQL 키워드 다음에 옵니다.

실행문

호스트 언어의 명령문이 같은 위치에 지정되어 실행될 때마다 어플리케이션 프로그램에 삽입된 실행문이 실행됩니다. 이는 루프 내의 명령문은 루프가 실행될 때마다 실행되고, 조건 구조 내의 명령문은 조건이 만족될 때만 실행된다는 것을 의미합니다.

삽입된 명령문은 호스트 변수에 대한 참조를 포함할 수 있습니다. 이러한 방식으로 참조된 호스트 변수는 두 가지 방법으로 사용될 수 있습니다.

- 입력으로(호스트 변수의 현재 값은 명령문의 실행시 사용됩니다.)
- 출력으로(명령문 실행의 결과로 변수에 신규 값이 지정됩니다.)

특히, 표현식과 술부의 호스트 변수에 대한 참조는 변수의 현재 값으로 대체됩니다. 즉, 변수는 입력으로 사용됩니다. 기타 참조의 처리는 각 명령문에 대해 개별적으로 설명됩니다.

모든 실행문 다음에 SQL 리턴 코드의 테스트가 나와야 합니다. 또는 WHENEVER문(그 자체는 비실행문)은 삽입된 명령문이 실행된 후 즉시 제어의 흐름을 변경하는 데 사용될 수 있습니다.

SQL문에서 참조된 오브젝트는 명령문을 준비할 때 없어도 됩니다.

비실행문

삽입된 비실행문은 사전컴파일러에 의해서만 처리됩니다. 사전컴파일러는 이러한 명령문에서 발생한 오류를 모두 보고합니다. 명령문은 *절대*로 실행되지 않으며, 어플리케이션 프로그램의 실행문 사이에 위치하면 조작되지 않습니다. 따라서, 이러한 명령문 다음에는 SQL 리턴 코드의 테스트가 나와서는 안됩니다.

동적 준비 및 실행

어플리케이션 프로그램은 호스트 변수에 위치한 문자 스트링 형식으로 SQL문을 동적으로 빌드할 수 있습니다. 일반적으로 명령문은 프로그램에서 사용할 수 있는 일부 자료로 부터 빌드됩니다(예를 들면, 워크스테이션으로부터의 입력). 명령문은 (삽입된) 명령문 PREPARE로 실행을 준비하고 (삽입된) 명령문 EXCUTE에 의해 실행될 수 있습니다. 또는 (삽입된) 명령문 EXECUTE IMMEDIATE를 사용하여 한 단계로 명령문을 준비하고 실행할 수 있습니다.

동적으로 준비된 명령문은 호스트 변수에 대한 참조를 포함할 수 없습니다. 대신 매개변수 마커를 포함할 수 있습니다. 매개변수 마커에 대한 규칙은 728 페이지의 『PREPARE』를 참조하십시오. 준비된 명령문이 실행되면 매개변수 마커는 EXECUTE문에 지정된 호스트 변수의 현재 값으로 대체됩니다. 이러한 대체에 대한 규칙은 670 페이지의 『EXECUTE』를 참조하십시오. 명령문이 준비된 후에는 다른 호스트 변수 값으로 여러번 실행할 수 있습니다. 매개변수 마커는 EXECUTE IMMEDIATE에서는 허용되지 않습니다.

EXECUTE(또는 EXECUTE IMMEDIATE)문 다음의 SQLCA에 있는 SQL 리턴 코드를 설정하여 명령문의 성공적 또는 성공적이지 않은 실행을 표시합니다. 삽입된 명령문의 경우에는 위에서 설명한 대로 SQL 리턴 코드를 검사해야 합니다. 자세한 내용은 368 페이지의 『SQL 리턴 코드』 주제를 참조하십시오.

select문의 정적 호출

select문은 (비실행) 명령문 DECLARE CURSOR의 부분으로 포함될 수 있습니다. 이러한 명령문은 (삽입된) 명령문 OPEN에 의해 커서가 열릴 때마다 실행됩니다. 커서가 열린 후, 결과표는 FETCH문을 연속적으로 실행하여 한 번에 한 행씩 또는 복수 행 FETCH문을 사용하여 한 번에 여러 행씩 검색할 수 있습니다.

이러한 방식으로 *select*문에는 호스트 변수에 대한 참조가 포함될 수 있습니다. 이러한 참조는 OPEN을 실행하는 순간에 변수가 갖는 값으로 대체됩니다.

select문의 동적 호출

어플리케이션 프로그램은 호스트 변수에 위치한 문자 스트링 형식으로 *select*문을 동적으로 빌드할 수 있습니다. 일반적으로 명령문은 프로그램에서 사용할 수 있는 일부 자료로부터 구축됩니다(예를 들면, 워크스테이션에서 획득한 조회). 그런 다음, (삽입된) 명령문 OPEN에 의해 커서가 열릴 때마다 명령문이 실행됩니다. 커서가 열린 후, 결과표는 FETCH문을 연속적으로 실행하여 한 번에 한 행씩 또는 복수 행 FETCH문을 사용하여 한 번에 여러 행씩 검색할 수 있습니다.

이러한 방식으로 *select*문에 호스트 변수에 대한 참조가 포함되면 안됩니다. 대신 매개변수 마커를 포함할 수 있습니다. 매개변수 마커에 대한 규칙은 728 페이지의 『PREPARE』를 참조하십시오. 매개변수 마커는 OPEN문에 지정된 호스트 변수 값으로 대체됩니다. 이러한 대체에 대한 규칙은 723 페이지의 『OPEN』을 참조하십시오.

대화식 호출

워크스테이션에서 SQL문을 입력하는 함수는 데이터베이스 관리자 구조의 일부입니다. iSeries용 DB2 UDB 사용권 프로그램은 SQL 시작(STRSQL) 명령, Query Manager 시작(STRQM), 그리고 이 함수에 대한 iSeries Navigator의 SQL 스크립트 지원을 제공합니다. 다른 제품도 제공됩니다. 이 방식으로 입력된 명령문을 대화식으로 발행되었다고 합니다. 동적으로 준비될 수 없는 명령문은 연결 관리 명령문(CONNECT, DISCONNECT, RELEASE 및 SET CONNECTION)을 제외하고는 대화식으로 발행할 수 없습니다.

매개변수 마커나 호스트 변수에 대한 참조는 어플리케이션 프로그램의 문맥에서만 의미가 있으므로 대화식으로 발행된 명령문은 매개변수 마커나 호스트 변수에 대한 참조가 없는 실행문이어야 합니다.

SQL 리턴 코드

실행가능한 SQL문이 들어 있는 어플리케이션 프로그램은 적어도 다음 중 하나를 제공해야 합니다.

- SQLCA로 명명된 구조
- SQLCODE로 명명된 독립 정수 변수
- SQLSTATE로 명명된 독립 CHAR(5)(C에서는 CHAR(6)) 변수

독립 SQLCODE와 SQLSTATE가 둘 다 제공될 수 있습니다. SQLCA가 제공되면 독립 SQLCODE나 SQLSTATE는 둘 다 제공되지 않습니다. 독립 SQLCODE나 SQLSTATE는 호스트 구조에서 선언되면 안됩니다.

SQLCA는 REXX와 RPG에서 자동으로 제공됩니다. 다른 언어에서는 INCLUDE SQLCA문을 사용하여 SQLCA를 구할 수 있습니다. 독립 SQLCODE 또는 SQLSTATE

가 제공되면 INCLUDE SQLCA를 사용할 수 없습니다. SQLCA는 SQLCODE(RPG에서는 SQLCOD)라고 하는 정수 변수와 SQLSTATE(RPG에서는 SQLSTT)라고 하는 문자 스트링 변수를 갖습니다.

SQLCA 대신 독립 SQLSTATE를 제공하는 옵션은 ISO/ANSI SQL 표준을 따릅니다. 독립 SQLSTATE 대신 독립 SQLCODE를 제공하는 옵션은 ISO/ANSI SQL 표준의 피처와 반대됩니다. ISO/ANSI SQL 표준과 일치하는 것이 바람직한 경우 독립 SQLSTATE를 사용하십시오.

SQLCODE

어플리케이션 프로그램이 SQLCA나 독립 변수를 제공하는지 여부에 상관없이, SQLCODE는 각 SQL문이 실행되고 난 후 데이터베이스 관리자에 의해 설정됩니다. iSeries용 DB2 UDB는 다음과 같이 ISO/ANSI SQL 표준을 따릅니다.

- SQLCODE = 0이고 SQLWARN0이 공백이면 실행은 성공적이었습니다.
- SQLCODE = 100이면 발견된 자료가 없었습니다. 예를 들면, 커서가 결과표의 마지막 행 다음에 위치하였기 때문에 FETCH문은 자료를 리턴하지 않았습니다.
- SQLCODE > 0이고 = 100이 아니면, 실행은 경고가 표시되면서 성공적이었습니다.
- SQLCODE = 0이고 SQLWARN0 = 'W'이면 실행은 경고가 표시되면서 성공적이었습니다.
- SQLCODE < 0이면 실행이 성공적이지 않았습니다.

iSeries용 DB2 UDB SQLCODE 전체 리스트와 해당 SQLSTATE는 iSeries Information Center의 SQL 메세지 및 코드 책에서 제공됩니다.

SQLSTATE

어플리케이션 프로그램이 SQLCA나 독립 변수를 제공하는지 여부에 상관없이 SQLSTATE도 SQL문이 실행되고 난 후 데이터베이스 관리자에 의해서 설정됩니다. 따라서 어플리케이션 프로그램은 SQLCODE 대신 SQLSTATE를 테스트하여 SQL문의 실행을 검사할 수 있습니다.

SQLSTATE는 공통 오류 조건에 대한 공통 코드와 함께 어플리케이션 프로그램을 제공합니다. 또한 SQLSTATE는 어플리케이션 프로그램이 특정 오류나 오류 클래스를 테스트할 수 있도록 설계되었습니다. 구조는 모든 데이터베이스 관리자에서 동일하며 제안된 ISO/ANSI 표준을 기본으로 합니다. 각 SQLCODE와 연관된 SQLSTATE 클래스 및 SQLSTATE의 전체 리스트가 iSeries Information Center의 SQL 메세지 및 코드 책에서 제공됩니다.

SQL 주석

정적 SQL문은 호스트 언어나 SQL 주석을 포함할 수 있습니다. 동적 SQL문은 SQL 주석을 포함할 수 있습니다. SQL 주석에는 두 가지 유형이 있습니다.

단순 주석

단순 주석은 두 개의 연속 하이픈으로 시작됩니다.

브라켓 주석

브라켓 주석은 /*로 시작되고 */로 끝납니다.

다음 규칙은 단순 주석 사용에 적용됩니다.

- 두 개의 하이픈은 같은 행에 있어야 하고 공백에 의해 분리되면 안됩니다.
- 단순 주석은 공백이 유효한 곳이면 어디에서든 시작될 수 있습니다(분리 문자 토큰 안이나 'EXEC'과 'SQL' 사이 제외).
- 단순 주석은 다음 행까지 계속될 수 없습니다.
- COBOL에서는 하이픈 앞에 공백이 와야 합니다.

다음 규칙은 브라켓 주석 사용에 적용됩니다.

- /*는 같은 행에 있어야 하며 공백으로 분리하지 마십시오.
- */는 같은 행에 있어야 하며 공백으로 분리하지 마십시오.
- 브라켓 주석은 공백이 유효한 곳이면 어디에서든 시작될 수 있습니다(분리 문자 토큰 안이나 'EXEC'과 'SQL' 사이 제외).
- 브라켓 주석은 다음 행에 계속될 수 있습니다.
- 다른 브라켓 주석 내에 브라켓 주석이 내포될 수 있습니다.

예

다음 예는 명령문에 단순 주석을 포함시키는 방법을 보여줍니다.

```
CREATE VIEW PRJ_MAXPER         -- PROJECTS WITH MOST SUPPORT PERSONNEL
   AS SELECT PROJNO, PROJNAME  -- NUMBER AND NAME OF PROJECT
      FROM PROJECT
      WHERE DEPTNO = 'E21'     -- SYSTEMS SUPPORT DEPT CODE
      AND PRSTAFF > 1
```

다음 예는 명령문에 브라켓 주석을 포함시키는 방법을 보여줍니다.

```
CREATE VIEW PRJ_MAXPER         /* PROJECTS WITH MOST SUPPORT
                                      PERSONNEL               */
   AS SELECT PROJNO, PROJNAME  /* NUMBER AND NAME OF PROJECT   */
      FROM PROJECT
      WHERE DEPTNO = 'E21'     /* SYSTEMS SUPPORT DEPT CODE    */
      AND PRSTAFF > 1
```

ALTER TABLE

ALTER TABLE문은 표 정의를 변경합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표의 경우
  - 표의 ALTER 권한
  - 표가 들어 있는 라이브러리의 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 표에 대해 ALTER 권한을 갖습니다.

- 표의 소유자입니다.
- 표에 대한 ALTER 권한을 부여받았습니다.
- 표에 대한 *OBJALTER나 *OBJMGT의 시스템 권한을 부여받았습니다.

외부 키를 정의하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 사항 중 하나를 상위 표에 대해 포함해야 합니다.

- 표에 대한 오브젝트 관리 권한이나 REFERENCES 권한
- 지정된 상위 키의 각 열에 대한 REFERENCES 권한
- 표의 소유권
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 표에 대해 REFERENCES 권한을 갖습니다.

- 표의 소유자입니다.
- 표에 대해 REFERENCES 권한을 부여받았습니다.
- 표에 대해 *OBJREF이나 *OBJMGT의 시스템 권한을 부여받았습니다.

고유한 유형이 참조된다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

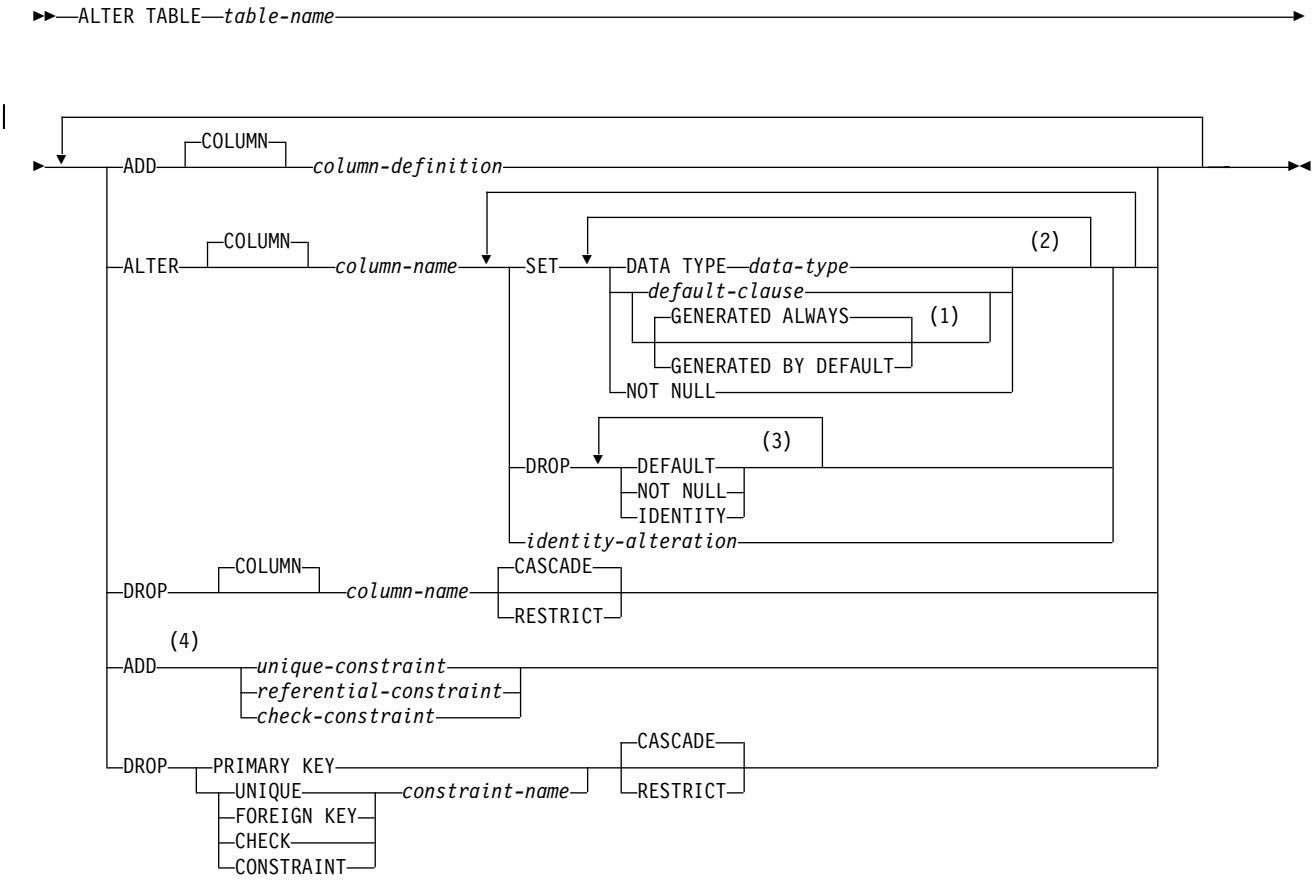
- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE

- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문



주:

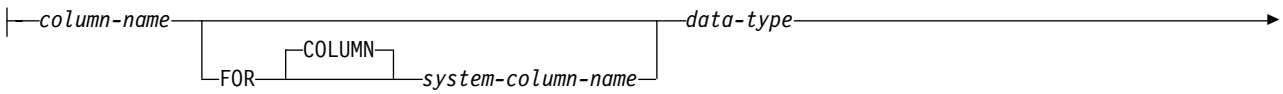
1 GENERATED는 열이 ROWID 자료 유형(또는 ROWID 자료 유형에 기초한 개별 유형)이거나 ID 열일 경우에만 지정할 수 있습니다.

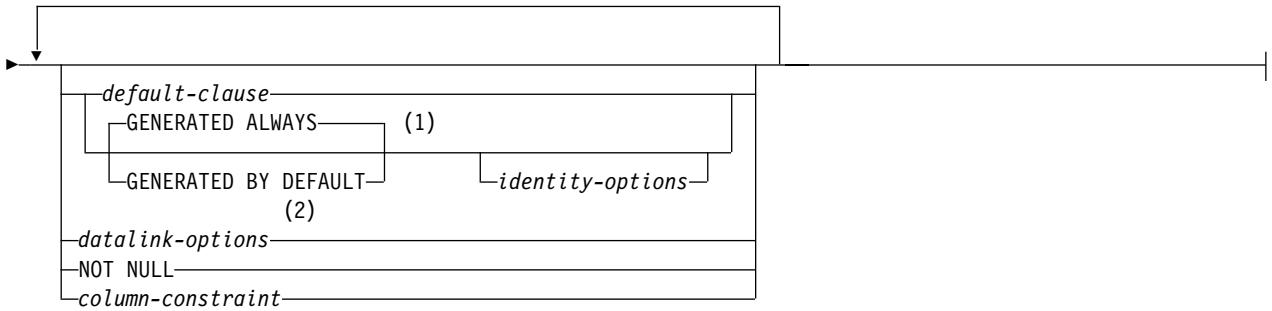
2 각 절은 한 번만 지정됩니다. DATA TYPE이 지정되는 경우 처음에 지정되어야 합니다.

3 각 절은 한 번만 지정됩니다.

4 ALTER TABLE문의 첫 번째 절이면 ADD 키워드는 선택적이지만, 권장됩니다. 그렇지 않은 경우에는 필수입니다.

column-definition:



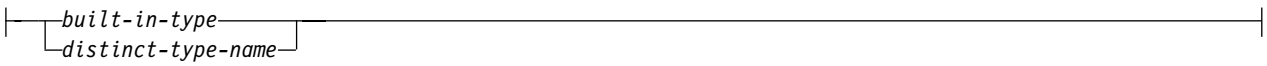


주:

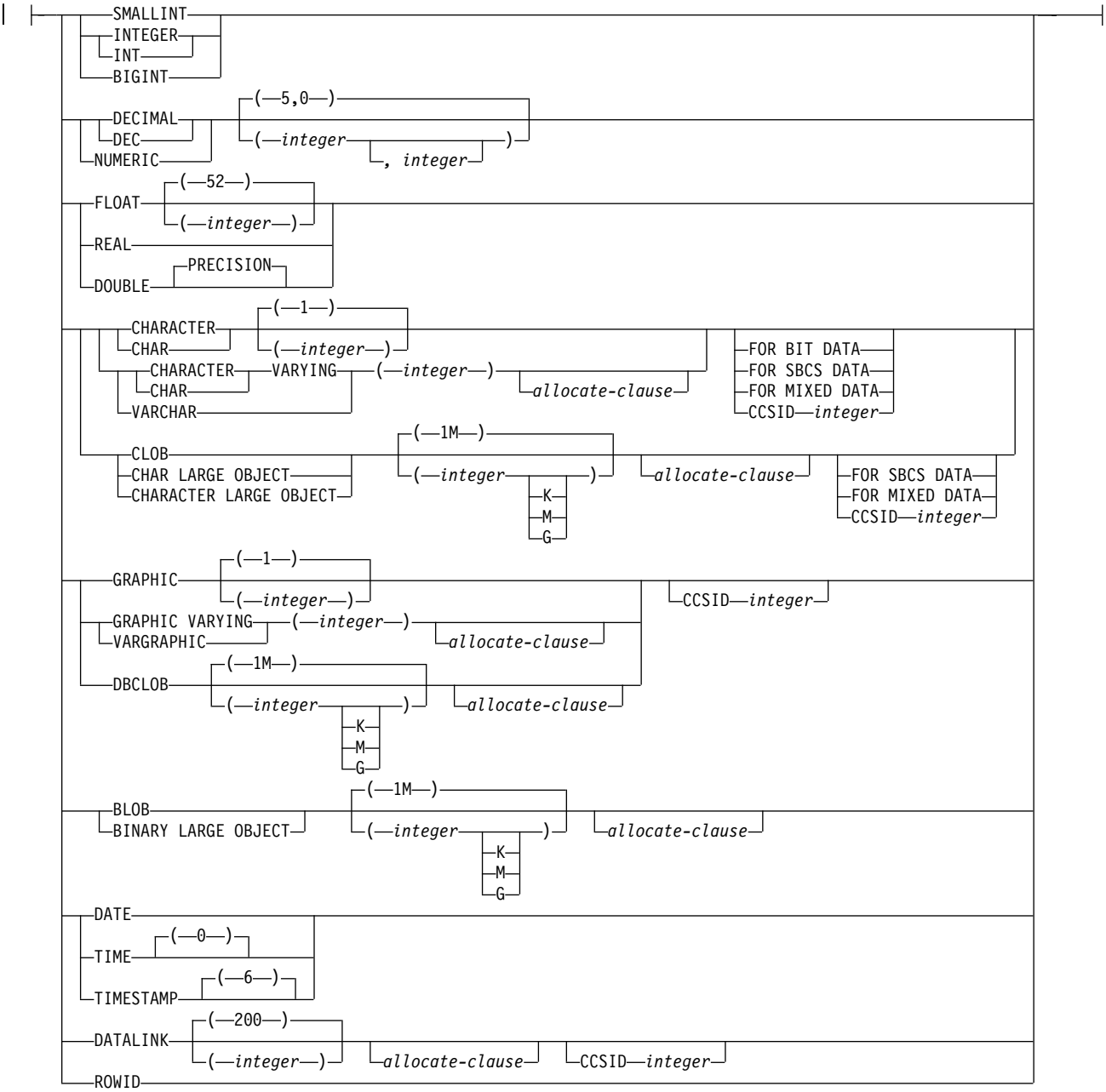
1 GENERATED는 열이 ROWID 자료 유형(또는 ROWID 자료 유형에 기초한 개별 유형)이거나 ID 열일 경우에만 지정할 수 있습니다.

2 datalink-options는 DATALINK와 DATALINK를 소스로 하는 distinct-type에 대해서만 지정될 수 있습니다.

data-type:

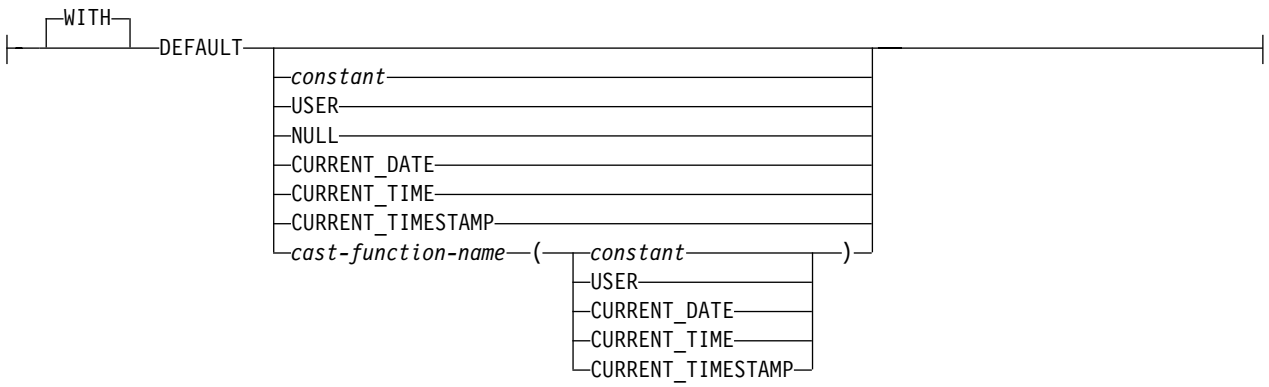


built-in-type:

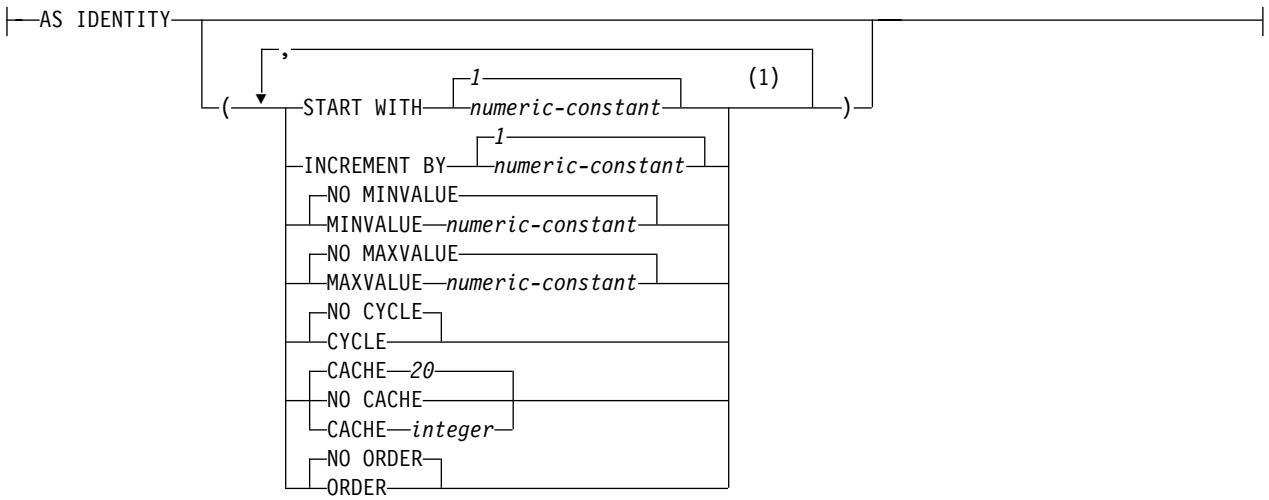


allocate절:

default절:



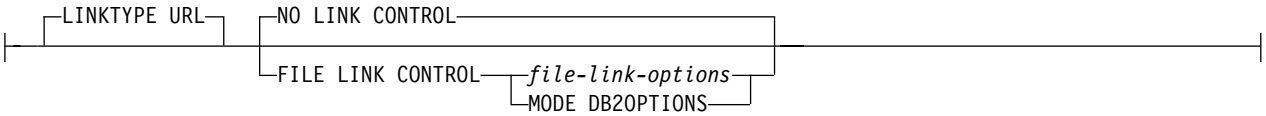
identity-options:



주:

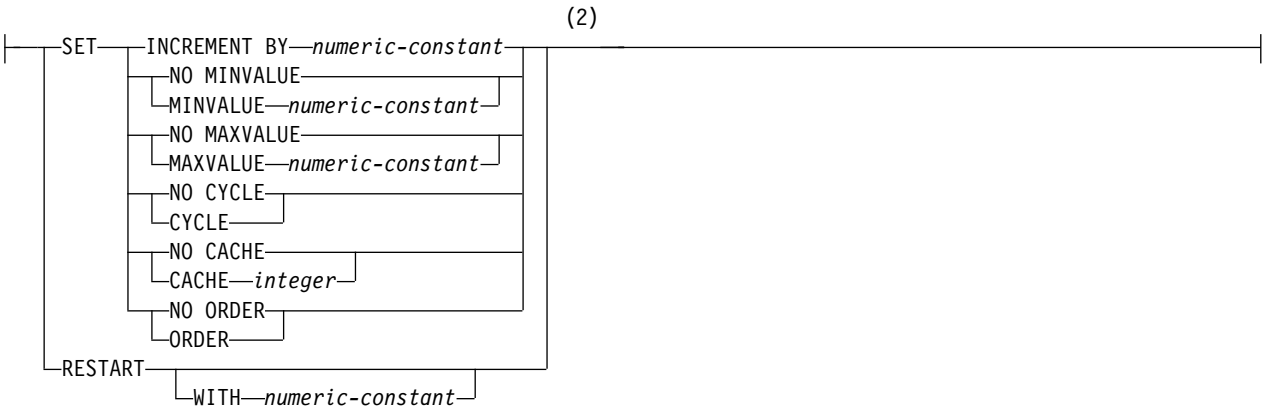
1 각 절은 한 번만 지정됩니다.

datalink-options:



file-link-options:

identity-alteration:



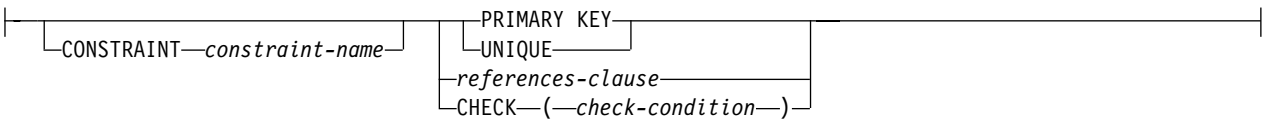
주:

1 모든 다섯 개의 *파일 링크 선택사항*이 지정되어야 합니다. 그러나 순서는 상관없습니다.

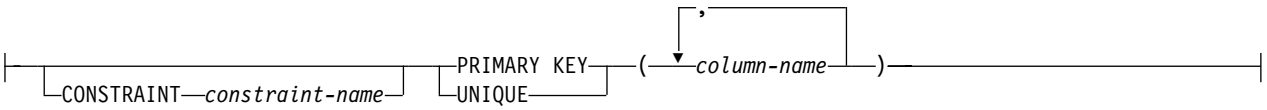
2 각 절은 한 번만 지정됩니다.

ALTER TABLE

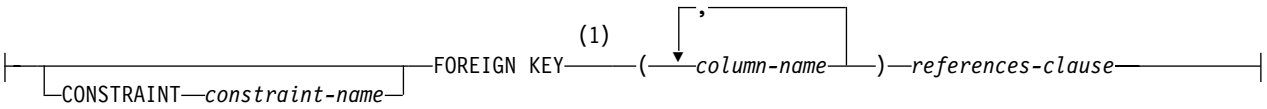
column-constraint:



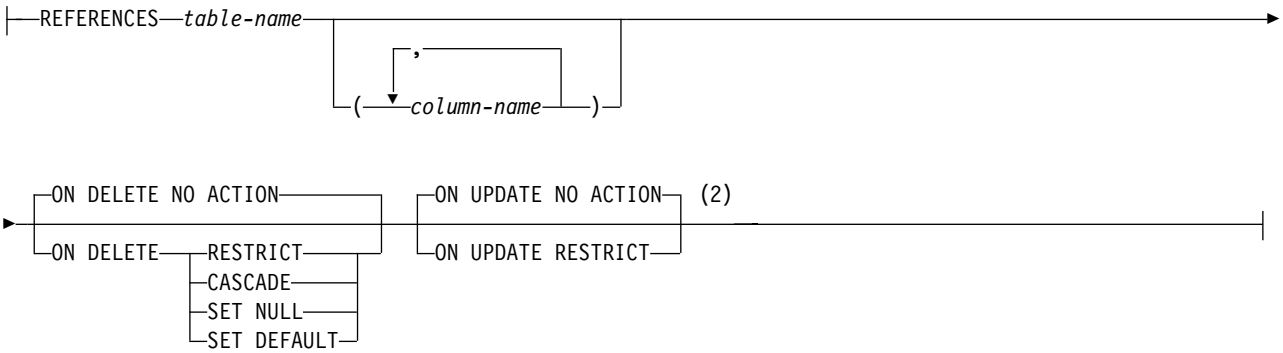
unique-constraint:



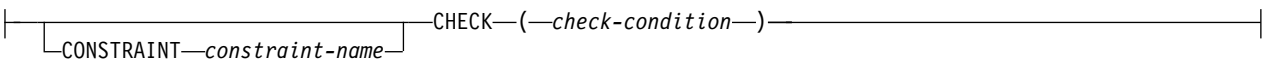
referential-constraint:



references절:



check-constraint:



주:

1 다른 제품과의 호환을 위해 constraint-name(CONSTRAINT 키워드가 없는)을 FOREIGN KEY 다음에 지정할 수 있습니다.

2 ON DELETE와 ON UPDATE절은 순서에 관계 없이 지정할 수 있습니다.

설명

table-name

변경하고자 하는 표를 식별합니다. *table-name* 현재 서버에 있는 표를 식별해야 합니다. 뷰나 카탈로그 표 또는 글로벌 임시 마침표가 되어서는 안됩니다.

ADD COLUMN

column-definition

표에 열을 추가합니다. 표에 행이 있으면 열이 ROWID 열이거나 ID 열(AS IDENTITY로 정의된 열)이 아닌 이상 열의 모든 값은 디폴트 값으로 설정됩니다. 데이터베이스 관리자는 ROWID 열 및 ID 열의 디폴트 값을 생성합니다. 이전에 표에 *n* 열이 포함되어 있었으면 새 열의 순서는 *n*+1입니다. *n* 값은 8000을 넘을 수 없습니다.

표는 하나의 ROWID 열만 가질 수 있습니다. 이미 ID 열을 가진 표에 ID 열을 추가할 수 없습니다.

새 열을 추가했을 때 열의 레코드 행 바이트 수 합계가 32766 보다 커서는 안되고, VARCHAR이나 VARGRAPHIC 열이 지정된 경우에는 32740 보다 크면 안됩니다. 또한 LOB가 지정되면 열의 바이트 수 합계는 15,728,640보다커서는 안됩니다. 자료 유형에 따른 열의 바이트 수에 대한 내용은 569 페이지의 『주』를 참조하십시오.

column-name

표에 추가하고자 하는 열을 명명합니다. 표의 하나 이상의 열이나
표의 system-column-name에 같은 이름을 사용할 수 없습니다. *column-name*을 규정할 수 없습니다.

FOR COLUMN *system-column-name*

열에 대해 OS/400 이름을 제공합니다. 표의 하나 이상의 열이나 system-column-name에 같은 이름을 사용할 수 없습니다.

system-column-name이 지정되지 않고 column-name이 유효한
system-column-name이 아니면, 시스템 열 이름이 생성됩니다. 시스템 열 이름이 생성되는 방법에 대한 자세한 내용은 572 페이지의 『열 이름 생성 규칙』을 참조하십시오.

data-type

열의 자료 유형을 지정합니다.

내장형

내장 자료 유형을 지정합니다. 541 페이지의 『CREATE TABLE』에서 *내장 유형*에 대한 설명을 참조하십시오.

FILE LINK CONTROL을 갖는 DataLink 열은 CASCADE의 삭제 규칙을 갖는 참조 제한조건에서 종속되는 표에 추가할 수 없습니다.

DEFAULT

열에 대한 디폴트 값을 지정합니다. 이 절은 *column-definition*에 한 번 이상 지정될 수 없습니다. ROWID 열이나 ID 열(AS IDENTITY로 정의된 열)에

는 디폴트 값을 지정할 수 없습니다. 데이터베이스 관리자는 ROWID 열 및 ID 열의 디폴트 값을 생성합니다. 디폴트 키워드 다음에 값이 지정되지 않으면 다음이 적용됩니다.

- 열이 널이면 디폴트 값은 널값입니다.
- 열이 널이 아니면 디폴트는 열의 자료 유형에 따라 다릅니다.

자료 유형	디폴트 값
숫자	0
고정 길이 스트링	공백
가변 길이 스트링	길이가 0인 스트링
날짜	기존 행의 경우 0001년 1월 1일에 해당하는 날짜. 추가된 행의 경우는 현재 날짜
시간	기존 행의 경우 0시간, 0분, 0초에 해당하는 시간. 추가된 행의 경우는 현재 시간
시간소인	기존 행의 경우 0001년 1월 1일에 해당하는 날짜와 0시간, 0분, 0초, 0마이크로초에 해당하는 시간. 추가된 행의 경우는 현재 시간소인
Datalink	DLVALUE(‘ ’,‘URL’,‘ ’)에 해당하는 값
distinct-type	고유한 유형의 소스 유형에 대응하는 디폴트 값

*column-definition*에서 NOT NULL과 DEFAULT의 생략은 DEFAULT NULL의 내재적 스펙입니다.

constant

열에 대한 디폴트 값으로 상수를 지정합니다. 지정된 상수는 80 페이지의 『지정과 비교』에 설명된 대로 지정 규칙에 따라 열에 지정될 수 있는 값을 표시해야 합니다. 부동 소수점 상수를 SMALLINT, INTEGER, BIGINT, DECIMAL 또는 NUMERIC 열에 대해 사용해서는 안됩니다. 소수 상수는 열의 지정된 눈금보다 더 많은 소수점 이하의 자릿수를 가질 수 없습니다.

USER

INSERT 또는 UPDATE 시에 USER 특수 레지스터의 값을 열에 대한 디폴트 값으로 지정합니다. 열의 자료 유형은 길이 속성이 USER의 특수 레지스터보다 크거나 같은 CHAR나 VARCHAR가 되어야 합니다. 기존 행의 경우 값은 ALTER TABLE문이 처리될 때 USER 특수 레지스터의 값입니다.

NULL

열에 대한 디폴트 값으로 널을 지정합니다. NOT NULL이 지정되면 동일한 *column-definition* 내에 DEFAULT NULL이 지정되면 안됩니다.

CURRENT_DATE

열에 대한 디폴트 값으로 현재 날짜를 지정합니다. CURRENT_DATE가 지정되면 열의자료 유형은 DATE 또는 DATE에 기초한 고유한 유형이어야 합니다.

CURRENT_TIME

열에 대한 디폴트 값으로 현재 시간을 지정합니다. CURRENT_TIME이 지정되면 열의 자료 유형은 TIME 또는 TIME에 기초한 고유한 유형이어야 합니다.

CURRENT_TIMESTAMP

열에 대한 디폴트 값으로 현재 시간소인을 지정합니다.
CURRENT_TIMESTAMP가 지정되면 열의 자료 유형은 TIMESTAMP이거나 TIMESTAMP에 기초한 고유한 유형이어야 합니다.

cast-function-name

이러한 디폴트 값 형식은 고유한 유형, BLOB, CLOB, DBCLOB, DATE, TIME 또는 TIMESTAMP 자료 유형으로 정의된 열과 함께만 사용될 수 있습니다. 다음 표는 이 *cast-function*이 허용되는 사용법을 설명한 것입니다.

자료 유형	캐스트 함수명
BLOB, CLOB 또는 DBCLOB를 기초로 하는 고유한 유형 N	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP를 기초로 하는 고유한 유형 N	N(N이 생성되었을 때 생성된 사용자 정의 캐스트 함수) ** 또는 DATE, TIME 또는 TIMESTAMP *
기타 자료 유형을 기초로 하는 고유한 유형 N	N(N이 생성되었을 때 생성된 사용자 정의 캐스트 함수) **
BLOB, CLOB 또는 DBCLOB	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP	DATE, TIME 또는 TIMESTAMP *

주:

* 함수 이름은 내재적 또는 명시적 스키마명 QSYS2를 갖는 자료 유형의 이름(또는 고유한 유형의 소스 유형과 일치해야 합니다.

** 함수 이름은 열에 대한 고유한 유형 이름과 일치해야 합니다. 스키마 이름으로 규정했으면 고유한 유형에 대한 스키마 이름과 같아야 합니다. 규정하지 않았으면 함수 해석의 스키마 이름이 고유한 유형에 대한 스키마 이름과 같아야 합니다.

constant

상수를 인수로 지정합니다. 상수는 고유한 유형의 소스 유형 또는 고

유한 유형이 아닌 경우 자료 유형에 대한 상수 규칙을 따라야 합니다. BLOB, CLOB, DBCLOB, DATE, TIME 및 TIMESTAMP 함수의 경우 상수는 스트링 상수이어야 합니다.

USER

INSERT 또는 UPDATE 시에 USER 특수 레지스터의 값을 열에 대한 디폴트 값으로 지정합니다. 열의 고유한 유형의 소스 유형에 대한 자료 유형은 길이 속성이 USER 길이 속성보다 크거나 같은 CHAR나 VARCHAR가 되어야 합니다. 기존 행의 경우 값은 ALTER TABLE 문이 처리될 때 USER 특수 레지스터의 값입니다.

CURRENT_DATE

열에 대한 디폴트 값으로 현재 날짜를 지정합니다. CURRENT_DATE가 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 DATE가 되어야 합니다.

CURRENT_TIME

열에 대한 디폴트 값으로 현재 시간을 지정합니다. CURRENT_TIME이 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 TIME이 되어야 합니다.

CURRENT_TIMESTAMP

열에 대한 디폴트 값으로 현재 시간소인을 지정합니다.

CURRENT_TIMESTAMP가 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 TIMESTAMP가 되어야 합니다.

GENERATED

데이터베이스 관리자가 열의 생성 값을 지정합니다. 열이 ID 열로 간주되는 경우(AS IDENTITY절로 정의) GENERATED를 지정해야 합니다. 열의 자료 유형이 ROWID인 경우(또는 ROWID에 기초한 고유 유형)에도 이 값을 지정해야 합니다. 아니면, 지정할 수 없습니다.

ALWAYS

표에 행이 삽입될 때 데이터베이스 관리자가 항상 열 값을 생성하도록 지정합니다. ALWAYS는 권장 값입니다.

BY DEFAULT

열에 대해 값이 지정되지 않은 경우에만 행이 삽입될 때 데이터베이스 관리자가 항상 열 값을 생성하도록 지정합니다. 값이 지정되면, 데이터베이스 관리자는 값을 사용합니다.

ROWID 열에 대해 데이터베이스 관리자는 지정된 값을 사용하지만, 그 값은 이전에 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에서 생성된 유효한 고유 행 ID여야 합니다.

ID 열에 데이터베이스 관리자는 지정된 값을 삽입하지만 ID 열에 해당 ID 열을 고유하게 지정하는 고유 제한조건이나 고유 인덱스가 있지 않은 경우 해당 값이 열의 고유 값임을 검증하지 않습니다.

AS IDENTITY

열은 표의 ID 열임을 지정합니다. 표는 하나의 ID 열만 가질 수 있습니다. AS IDENTITY는 열의 자료 유형이 스케일 0을 가진 정확한 숫자 유형(스케일 0인 SMALLINT, INTEGER, BIGINT, DECIMAL 또는 NUMERIC, 이러한 유형에 기초한 고유 유형)인 경우에만 지정할 수 있습니다.

ID 열은 묵시적으로 NOT NULL입니다.

START WITH *numeric-constant*

ID 열에 대해 생성된 첫 번째 값을 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 말아야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았다면, 디폴트 값은 오름차순의 경우 MINVALUE, 내림차순의 경우 MAXVALUE입니다. 이 값은 순서의 최대값 또는 최소값에 도달한 후 순서가 순환하는 값일 필요는 없습니다. START WITH절을 사용하면 주기에 사용된 범위의 바깥에 순서를 시작할 수 있습니다. 주기에 사용된 범위는 MINVALUE 및 MAXVALUE로 정의됩니다.

INCREMENT BY *numeric-constant*

ID 열의 연속 값 사이의 간격을 지정합니다. 이 값은 0이 아닌 양수이거나 음수일 수 있지만 큰 정수 상수 값을 초과하지 않으며, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 않는 열에 지정할 수 있습니다. 디폴트는 1입니다.

값이 양수라면, ID 열의 값 순서는 오름차순입니다. 값이 음수라면, ID 열의 값 순서는 내림차순입니다.

MAXVALUE *numeric-constant*

이 ID 열에 대해 생성된 최대값인 숫자 상수를 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 최소값보다 커야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았을 때, 이 값은 오름차순의 경우 자료 유형의 최대값(DECIMAL의 경우 정밀도), 내림차순의 경우 START WITH 값(START WITH가 지정되지 않은 경우 -1)입니다.

MINVALUE *numeric-constant*

이 ID 열에 대해 생성된 최소값인 숫자 상수를 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 최대값보다 적어야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았을 때, 이 값은 오름차순의 경우 START WITH 값(START WITH가 지정되지 않은 경우 -1), 내림차순의 경우 자료 유형의 최소값(DECIMAL의 경우 정밀도)입니다.

CACHE 또는 NO CACHE

사전 할당된 일부 값을 메모리에 보존할 것인지 여부를 지정합니다. 값을 사전 할당하여 캐시에 저장하면 표에 행을 삽입하는 기능이 좋아집니다.

CACHE *integer*

데이터베이스 관리자가 사전에 할당하여 메모리에 보관하는 ID 열의 값 수를 지정합니다. 지정할 수 있는 최소값은 2이고, 최대값은 정수로 표현될 수 있는 가장 큰 값입니다. 디폴트는 20입니다.

시스템 오류가 발생하면, 지정될 캐시된 모든 ID 열 값이 유실되어 다시는 사용되지 않습니다. 따라서, CACHE에 대해 지정된 값은 시스템 오류 중 유실될 수 있는 ID 열의 최대 값 수를 나타냅니다.

NO CACHE

ID 열의 값이 사전에 할당되지 않도록 지정합니다.

CYCLE 또는 CYCLE

순서의 최대값 또는 최소값에 도달한 후 이 ID 열이 계속 값을 생성하는지 여부를 지정합니다.

CYCLE

최대값이나 최소값에 도달한 후 이 열에 대해 값이 계속 생성됨을 지정합니다. 이 옵션이 사용된 경우, 오름차순이 순서의 최대값에 도달한 후 해당 최소값을 생성합니다. 내림차순이 순서의 최소값에 도달하고 나면, 최대값이 생성됩니다. 열의 최대값과 최소값은 주기에 사용된 범위를 판별합니다.

CYCLE이 유효한 경우, 데이터베이스 관리자는 ID 열에 대해 중복값을 생성할 수 있습니다. ID 열에 대해 제한조건 또는 고유 색인이 존재하는데 이 열에 대해 고유하지 않은 값이 생성되면 오류가 발생합니다.

NO CYCLE

순서의 최대값이나 최소값에 도달한 후 ID 열에 대해 값이 생성되지 않도록 지정합니다. 이것이 디폴트 값입니다.

ORDER 또는 NO ORDER

요청 순서대로 ID 값이 생성되도록 지정합니다.

ORDER

요청 순서대로 값이 생성되도록 지정합니다.

NO ORDER

요청 순서대로 값이 생성될 필요가 없도록 지정합니다. 이것이 디폴트 값입니다.

datalink-options

DATALINK 열과 연관된 옵션을 지정합니다. *datalink-options*에 대한 설명은 541 페이지의 『CREATE TABLE』을 참조하십시오.

NOT NULL

열이 널값을 포함하지 않도록 합니다. NOT NULL의 생략은 열이 널값을 가질 수 있다는 것을 의미합니다. NOT NULL이 지정되면 DEFAULT도 지정되어야 합니다.

column-constraint

CONSTRAINT *constraint-name*

제한사항을 명명합니다. constraint-name은 이미 현재 서버에 있는 제한을 식별해서는 안됩니다.

구가 지정되지 않으면 고유 제한 이름은 데이터베이스 관리자에 의해 생성됩니다.

PRIMARY KEY

단일 열로 구성된 1차 키를 정의하는 축약 메소드를 제공합니다. 따라서, PRIMARY KEY가 열 C의 정의에 지정되면 결과는 PRIMARY KEY(C) 절이 분리된 절로 지정된 경우와 동일합니다.

이 절은 하나 이상의 *column-definition*에 지정되어서는 안되며 UNIQUE 절이 열 정의에 지정된 경우에는 절대 지정되어서는 안됩니다. 열은 LOB 또는 DataLink 열이어서는 안됩니다.

1차 키가 추가되면 NULL 값이 1차 키를 구성하는 열에 허용되지 않는 규칙을 강제하기 위해 CHECK 제한이 내재적으로 추가됩니다.

UNIQUE

단일 열로 구성된 고유 키를 정의하는 축약 메소드를 제공합니다. 따라서, UNIQUE가 열 C의 정의에 지정되면 결과는 UNIQUE(C)절이 분리된 절로 지정된 경우와 동일합니다.

이 절은 열 정의에 한 번 이상 지정될 수 없고 PRIMARY KEY가 *column-definition*에 지정된 경우 지정되어서는 안됩니다. 열은 LOB 또는 DataLink 열이어서는 안됩니다.

references절

column-definition의 *reference-clause*는 단일 열로 구성된 외부 키를 정의하는 축약 메소드를 제공합니다. 따라서, reference절이 열 C의 정의에 지

정되면 결과는 C가 유일하게 식별되는 열인 FOREIGN KEY절의 일부로 reference절이 지정되는 경우와 동일합니다.

CHECK(*check-condition*)

*check-condition*만이 단일 열을 참조하는 검사 제한사항을 정의하는 축약 메소드를 제공합니다. 따라서, CHECK가 열 C의 열 정의에 지정되면 C 이외의 어떤 열도 검사 제한사항의 *check-condition*에서 참조될 수 없습니다. 결과는 검사 제한사항이 분리된 절로 지정된 경우와 동일합니다.

FILE LINK CONTROL 열의 ROWID 또는 DATALINK는 CHECK 제한조건에 참조될 수 없습니다. 추가 제한 사항은 391 페이지의 『ADD check-constraint』를 참조하십시오.

ALTER COLUMN

기존 열의 정의를 변경합니다. 지정된 속성만 변경됩니다. 기타 사항은 변경되지 않습니다.

column-name

변경될 열을 식별합니다. 열 이름은 규정되어서는 안됩니다. 이름은 지정된 표의 열을 식별해야 합니다. 이름은 이 ALTER TABLE문에 추가되거나 제거된 열을 식별해서는 안됩니다.

SET DATA TYPE *data-type*

변경될 열의 새로운 자료 유형을 식별합니다. 새로운 자료 유형은 열의 기존 자료 유형과 호환되어야 합니다. 자료 유형의 호환성에 대한 자세한 내용은 80 페이지의 『지정과 비교』를 참조하십시오. 그러나, 일반 규칙에 두 가지 예외가 있습니다.

- 문자와 UCS-2 그래픽 사이의 자료 유형 변경이 허용됩니다.
- datatime 자료 유형에서 문자로의 자료 유형 변경은 허용되지 않습니다.

지정된 길이, 정밀도, 눈금은 기존 길이, 정밀도, 눈금보다 크거나 작거나 또는 같을 수 있습니다. 그러나, 신규 길이, 정밀도 또는 스케일이 더 작은 경우 절단 또는 숫자 변환 오류가 발생할 수 있습니다.

지정된 열이 디폴트 값을 갖고 새로운 디폴트 값이 지정되지 않으면 기존의 디폴트 값은 80 페이지의 『지정과 비교』에 설명된 대로 지정 규칙에 따라 열에 지정될 수 있는 값을 표시해야 합니다.

열이 고유, 1차 또는 외부 키에 지정되면 키의 열 길이의 새로운 합계는 2000-n을 초과하면 안됩니다. 여기서, n은 널을 허용하는지정된 열 수입니다.

속성을 변경하면 열에 있는 모든 기존 값을 스트링 값이 절단될 것을 제외하고 열에 대한 할당 규칙에 따라서 신규 열 속성으로 변환됩니다.

SET *default-clause*

변경될 열의 새로운 디폴트 값을 지정합니다. 지정된 디폴트 값은 80 페이지의 『지정과 비교』에 설명된 대로 지정 규칙에 따라 열에 지정될 수 있는 값을 표시해야 합니다.

SET NOT NULL

열이 널값을 가질 수 없도록 지정합니다. 표의 기존 행에 있는 열의 모든 값은 널이 되어서는 안됩니다. 지정된 열이 디폴트 값을 갖고 새로운 디폴트 값이 지정되지 않은 경우 기존의 디폴트 값은 NULL이 되어서는 안됩니다. 열이 참조 제한의 외부 키에서 SET NULL의 DELETE 규칙으로 식별되고 외부 키에 널이 될 수 있는 다른 열이 없으면, SET NOT NULL은 허용되지 않습니다.

SET GENERATED ALWAYS 또는 GENERATED BY DEFAULT

데이터베이스 관리자가 열의 생성 값을 지정합니다. 열이 ID 열로 간주될 경우(AS IDENTITY절로 정의) 또는 열의 자료 유형이 ROWID인 경우(또는 ROWID에 기초한 고유 유형) GENERATED를 지정해야 합니다. 아니면, 지정할 수 없습니다.

DROP DEFAULT

열의 현재 디폴트 값을 제거합니다. 지정된 열은 디폴트 값을 가져야 하며 널(null) 속성으로 NOT NULL을 가져서는 안됩니다. 새로운 디폴트 값은 널값입니다.

DROP NOT NULL

열이 널값을 가질 수 있도록 허용하면서, 열의 NOT NULL 속성을 제거합니다. 디폴트 값이 지정되지 않거나 이미 존재하지 않으면 새로운 디폴트 값은 널값입니다. 열이 표의 1차 키에서 지정되면 DROP NOT NULL은 허용되지 않습니다.

DROP IDENTITY

열의 ID 속성을 제거하여 해당 열은 단순 숫자 자료 유형 열로 만듭니다. 열이 ID 열이 아닌 경우, DROP IDENTITY는 허용되지 않습니다.

identity-alteration

열의 ID 속성을 변경합니다. 열은 ID열이여야 합니다. 속성의 설명은 383 페이지를 참조하십시오.

RESTART

ID 열의 다음 값을 지정합니다. WITH numeric-constant가 지정되지 않은 경우, 순서는 ID 열이 원래 작성될 때 시작값으로 내재적 또는 명시적으로 지정된 값에서 시작합니다. 열은 ID열이여야 합니다.

WITH *numeric-constant*

*numeric-constant*가 열의 다음 값으로 사용되도록 지정합니다. *numeric-constant*는 정확한 숫자 상수여야 합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 말아야 합니다.

DROP COLUMN

표에서 식별된 열을 제거합니다.

column-name

제거될 열을 식별합니다. 열 이름은 규정되어서는 안됩니다. 이름은 지정된 표의 열을 식별해야 합니다. 이름은 ALTER TABLE문에 이미 추가되었거나 변경된 열을 식별해서는 안됩니다. 이름은 표의 열만 식별해서는 안됩니다.

CASCADE

드롭(drop)되는 열에 종속되는 뷰, 색인, 트리거 또는 제한사항 또한 드롭(drop)되도록 지정합니다.[40]

RESTRICT

열에 종속되는 뷰, 색인, 트리거 또는 제한사항이 있는 경우 해당 열을 삭제할 수 없도록 지정합니다.[40]

제한사항에서 참조되는 모든 열이 같은 ALTER TABLE문에서 제거되는 경우 RESTRICT는 제거를 막지 않습니다.

ADD unique-constraint

CONSTRAINT *constraint-name*

제한사항을 명명합니다. *constraint-name*은 이미 현재 서버에 있는 제한을 식별해서는 안됩니다. *constraint-name*은 스키마 내에서 고유해야 합니다.

지정되지 않은 경우 고유 제한사항 이름은 데이터베이스 관리자에 의해 생성됩니다.

UNIQUE(*column-name***,...)**

식별된 열로 구성된 고유 키를 정의합니다. 각 *column-name*은 표의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 길이의 합계는 2000-n을 초과해서는 안됩니다. 여기서 n은 널(null)을 허용하는 지정된 열의 수입니다. 식별된 열은 표의 다른 UNIQUE 제한사항이나 PRIMARY KEY에 지정된 열과 같을 수 없습니다. 예를 들어, UNIQUE(B,A)나 PRIMARY KEY(A,B)가 이미 표에 있으면 UNIQUE(A,B)는 허용되지 않습니다. 열 세트에 있는 기존의 널이 아닌 값은 고유해야 합니다. 복수 널값이 허용됩니다.

고유 색인이 이미 식별된 열에 있으면, 그 색인은 고유 색인으로 지정됩니다. 그렇지 않은 경우 고유 키의 고유성을 지원하기 위해 고유 색인이 작성됩니다. 고유 색인은 분리된 시스템 논리 파일이 아닌 시스템 실제 파일의 일부로 작성됩니다.

PRIMARY KEY(*column-name***,...)**

식별된 열로 구성된 1차 키를 정의합니다. 각 *column-name*은 표의 열을 식별하는

40. 트리거가 열 리스트의 UPDATE OF나 트리거된 조치의 어느 곳에서든 참조되는 경우 열에 종속됩니다.

규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열 수는 120을 초과해서는 안되고 길이 합계는 2000을 초과해서는 안됩니다. 표에 이미 1차 키가 있으면 안됩니다. 식별된 열은 표의 다른 UNIQUE 제한사항에서 지정된 열과 같을 수 없습니다. 예를 들어, UNIQUE(B,A)가 이미 표에 있는 경우 PRIMARY KEY(A,B)는 허용되지 않습니다. 열 세트에 있는 기존의 값은 고유해야 합니다. 1차 키가 추가되면 NULL 값이 1차 키를 구성하는 열에 허용되지 않는 규칙을 강제하기 위해 CHECK 제한이 내재적으로 추가됩니다.

고유 색인이 식별된 열에 이미 있으면 그 색인은 1차 색인으로 지정됩니다. 그렇지 않은 경우 1차 키의 고유성을 지원하기 위해 1차 색인이 작성됩니다. 고유 색인은 분리된 시스템 논리 파일이 아닌 시스템 실제 파일의 일부로 작성됩니다.

ADD referential-constraint

CONSTRAINT*constraint-name*

제한사항을 명명합니다. *constraint-name*은 이미 현재 서버에 있는 제한을 식별해서는 안됩니다.

지정되지 않은 경우 고유 제한사항 이름은 데이터베이스 관리자에 의해 생성됩니다.

FOREIGN KEY

참조 제한을 정의합니다.

T1은 변경되는 표를 나타냅니다.

(*column-name***,...)**

참조 제한의 외부 키는 식별된 열로 구성됩니다. 각 *column-name*은 T1 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 길이 합계는 2000-n을 초과해서는 안됩니다. 여기서, n은 널을 허용하는 지정된 열 수입니다.

REFERENCES *table-name*

REFERENCES절에 지정된 표 이름은 현재 서버에 있는 기본 표를 식별해야 하지만, 카탈로그 표는 식별해서는 안됩니다. 이 표는 제한 관계의 상위 표로 참조됩니다.

참조 제한의 외부 키, 상위 키, 상위 표가 기존 참조 제한의 외부 키, 상위 키, 상위 표와 같으면 참조 제한이 *duplicate*입니다. 중복 참조 제한은 허용되지만 권장되지는 않습니다.

T2는 식별된 상위 표를 나타냅니다.

(*column-name***,...)**

참조 제한의 상위 키는 식별된 열로 구성됩니다. 각 *column-name*은 T2의 열

을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 길이의 합계는 2000-n을 초과해서는 안됩니다. 여기서 n은 널(null)을 허용하는 지정된 열의 수입니다.

열 이름 리스트는 T2의 1차 키나 T2에 있는 UNIQUE 제한사항에 있는 열 이름 리스트와 동일해야 합니다. 이름은 순서에 관계 없이 지정할 수 있습니다. 예를 들어, (A,B)가 지정되면 UNIQUE(B,A)로 정의된 고유 제한사항은 요구사항을 만족시킵니다. 열 이름 리스트가 지정되지 않으면 T2는 1차 키를 가지고 있어야 합니다. 열 이름 리스트의 생략은 해당 1차 키 열의 내재적 스펙입니다.

지정된 외부 키는 T2의 상위 키와 같은 열 수를 가져야 합니다. 외부 키의 *n* 번째 열과 상위 키의 *n*번째 열의 설명은 동일한 자료 유형과 길이를 가져야 합니다.

표가 비어 있지 않으면 외부 키의 값은 표를 사용할 수 있기 전에 확인해야 합니다. 외부 키의 값은 ALTER TABLE문이 실행되는 동안 확인됩니다. 따라서, 외부 키의 널이 아닌 값은 모두 T2의 상위 키 값의 일부와 일치해야 합니다.

FOREIGN KEY절로 지정한 참조 제한사항은 T2가 상위이고 T1이 종속인 관계를 정의합니다.

ON DELETE

상위 표의 행이 삭제될 때 종속 표에 취할 조치를 지정합니다. 다음과 같은 다섯 개의 조치가 있습니다.

- NO ACTION(디폴트)
- RESTRICT
- CASCADE
- SET NULL
- SET DEFAULT

외부 키의 일부 열이 널값을 허용하지 않으면 SET NULL은 지정되어서는 안됩니다.

T1에 삭제 트리거가 있으면 CASCADE가 지정되어서는 안됩니다. T1에 갱신 트리거가 있으면 SET NULL 및 SET DEFAULT가 지정되어서는 안됩니다.

T1에 FILE LINK CONTROL을 갖는 DataLink 열이 있으면 CASCADE를 지정해서는 안됩니다.

삭제 규칙은 T2의 행이 DELETE의 오브젝트나 전파된 삭제 조작일 때 적용되며 그 행은 T1에 하위 행을 갖습니다. *p*는 T2의 그런 행을 나타냅니다.

- RESTRICT나 NO ACTION이 지정되면 오류가 발생하고 어떤 행도 삭제되지 않습니다.
- CASCADE가 지정되면 삭제 조작은 T1에 있는 *p*의 하위 행에 전파됩니다.
- SET NULL이 지정되면 T1에 있는 *p*의 각 하위 행인 외부 키를 가진 널 가능 열이 널(null)로 설정됩니다.
- SET DEFAULT가 지정되면 T1에 있는 *p*의 각 하위 행인 외부 키를 가진 각 열이 디폴트 값으로 설정됩니다.

ON UPDATE

상위 표의 행이 갱신될 때 종속 표에 취할 조치를 지정합니다.

갱신 규칙은 T2의 행이 UPDATE의 오브젝트나 전파된 삭제 조작일 때 적용되며 그 행은 T1에 하위 행을 갖습니다. *p*는 T2의 그런 행을 나타냅니다.

- RESTRICT나 NO ACTION이 지정되면 오류가 발생하고 어떤 행도 갱신되지 않습니다.

ADD check-constraint

CONSTRAINT *constraint-name*

제한사항을 명명합니다. *constraint-name*은 이미 현재 서버에 있는 제한을 식별해서는 안됩니다. *constraint-name*은 스키마 내에서 고유해야 합니다.

지정되지 않은 경우 고유 제한사항 이름은 데이터베이스 관리자에 의해 생성됩니다.

CHECK(*check-condition***)**

검사 제한사항을 정의합니다. *check-condition*은 표의 모든 행에 대해 참이거나 값이 지정되지 않아야 합니다.

*check-condition*은 다음 사항을 제외하고는 *search-condition*입니다.

- 표의 열만 참조할 수 있습니다.
- FILE LINK CONTROL 열의 ROWID 또는 DATALINK를 참조할 수 없습니다.
- 다음 사항을 포함해서는 안됩니다.
  - 부속 조회
  - Scalar-subselects
  - 열 함수
  - 호스트 변수
  - 매개변수 마커
  - LOB가 들어 잇는 복합 표현식(연결 등)
  - CURRENT TIMEZONE, CURRENT SCHEMA, CURRENT SERVER, CURRENT PATH, 및 USER 특수 레지스터

- NOW, CURDATE, 및 CURTIME 스칼라 함수
- NODENAME 스칼라 함수
- LOB와 관련된 표현식
- 고유한 유형의 작성으로 내재적으로 생성된 사용자 정의 기능
- ATAN2, DIFFERENCE, RAND, RADIANS 및 SOUNDEX 스칼라 함수
- DLVALUE, DLURLPATH, DLURLPATHONLY, DLURLSERVER 또는 DLURLSCHEME 스칼라 함수
- DLURLCOMPLETE 스칼라 함수(FILE LINK CONTROL 및 READ PERMISSION DB 속성을 갖는 자료 링크의 경우)

search-condition에 대한 자세한 정보는 154 페이지의 『탐색 조건』을 참조하십시오.

DROP

PRIMARY KEY
1차 키가 상위 키인 모든 참조 제한과 1차 키의 정의를 제거합니다. 표에는 1차 키가 있어야 합니다.

FOREIGN KEY *constraint-name*
참조 제한 *constraint-name*을 제거합니다. *constraint-name*은 표가 하위 표인 참조 제한을 식별해야 합니다.

UNIQUE *constraint-name*
고유 제한사항 *constraint-name*과 고유 키가 상위 키인 모든 참조 제한사항을 제거합니다. *constraint-name*은 표의 고유 제한사항을 식별해야 합니다. DROP UNIQUE는 PRIMARY KEY 고유 제한사항을 제거하지 않습니다.

CHECK *constraint-name*
검사 제한사항 *constraint-name*을 제거합니다. *constraint-name*은 표의 검사 제한사항을 식별해야 합니다.

CONSTRAINT*constraint-name*
제한사항 *constraint-name*을 제거합니다. *constraint-name*은 표의 검사, 고유 또는 참조 제한을 식별해야 합니다. 제한이 PRIMARY KEY나 UNIQUE 제한이면 1차 키나 고유 키가 상위 키인, 모든 참조 제한사항도 제거됩니다.

CASCADE
제거되는 제한에 종속적인 참조 제한도 제거되도록 고유 제한을 지정합니다.

RESTRICT
참조 제한이 제한에 종속적인 경우 제한이 제거될 수 없도록 고유 제한을 지정합니다.

주

열은 단일 ALTER TABLE문의 ADD, ALTER 또는 DROP COLUMN절에서 *한 번* 만 참조될 수 있습니다. 그러나 같은 열이 같은 ALTER TABLE문에 제한을 추가하거나 제거할 때는 여러 번 참조될 수 있습니다.

ALTER TABLE문 내의 조작 순서는 다음과 같습니다.

- 제한 제거
- RESTRICT 옵션이 지정된 열 제거
- 다른 모든 열 정의 변경
  - CASCADE 옵션이 지정된 열 제거
  - 열 변경 속성 추가
  - 열 추가
- 제한 추가

한 가지를 제외하고, 이 단계에서 사용자가 절을 지정하는 순서는 절이 수행되는 순서입니다. 열이 제거되는 조작은 열 정의가 추가되거나 변경되기 전에 논리적으로 수행됩니다.

변경되는 표에 종속적인 다른 작업의 QTEMP에 있는 뷰나 논리 파일은 ALTER TABLE문의 결과로 제거됩니다.

권한 검사는 변경되는 표에서만 수행됩니다. 기타 오브젝트는 ALTER TABLE문에 의해 액세스될 수 있지만, 해당 오브젝트에 대해 어떤 권한도 필요하지 않습니다. 예를 들면, 변경되는 표에 존재하는 뷰나, 참조 제한을 통해 변경되는 표를 참조하는 종속 표에서는 어떤 권한도 필요하지 않습니다.

표를 변경하기 전에 표와 종속 뷰의 현재 백업이나 논리 파일이 있어야 합니다.

다음 실행시 고려사항은 표에서 열을 추가, 변경 또는 제거할 때 ALTER TABLE문에 적용됩니다.

- 표에 있는 자료가 복사될 수 있습니다.[41]

  열을 추가하고 제거하려면 복사될 자료가 필요합니다.

  열을 변경하려면 일반적으로 복사될 자료가 필요합니다. 그러나, 다음의 변경 사항만이 변경에 포함되면 자료를 복사할 필요가 없습니다.
  - VARCHAR 열의 길이 속성이 증가하고 현재 길이 속성이 20보다 큽니다.
  - VARGRAPHIC 열의 길이 속성이 증가하고 현재 길이 속성이 10보다 큽니다.
  - VARCHAR 열의 할당된 길이가 변경되고 현재 및 새로 할당된 길이 모두 20보다 작거나 같습니다.

41. 전체 사본을 만들기에 충분한 기억장치가 없으면, 약 16 - 32메가바이트(MB)의 기억장치만을 필요로 하는 특수 복사가 수행됩니다.

  - VARGRAPHIC 열의 할당된 길이가 변경되고 현재 및 새로 할당된 길이 모두 10보다 작거나 같습니다.
  - 열의 코드화 문자 세트 ID(CCSID)가 변경되지만 이전의 CCSID에서 새로운 CCSID로 변환할 필요가 없을 때. 예를 들어, 한 코드화 문자 세트 ID(CCSID)가 65535이면 자료 변환은 필요하지 않습니다.
  - 디폴트 값이 변경되고 디폴트 값의 길이는 현재 할당된 길이보다 크지 않습니다.
  - DROP DEFAULT가 지정됩니다.
  - DROP NOT NULL이 지정되지만, 표 변경이 완료된 후에도 널이 가능한 열이 적어도 하나 표에 존재합니다.
- 색인이 재구축되어야 합니다.[42]

  열이 표에 추가될 때나 열이 제거 또는 변경되어 색인 키에서 열이 참조되지 않을 때는 색인을 재구축할 필요가 없습니다.

  제한이나 색인 키에서 사용되는 열을 변경하려면 일반적으로 재구축할 색인이 필요합니다. 그러나 다음 경우에는 색인을 재구축할 필요가 없습니다.
  - VARCHAR 또는 VARGRAPHIC 키의 길이 속성이 증가하는 경우
  - 열의 코드화 문자 세트 ID(CCSID)가 변경되지만 이전의 CCSID에서 새로운 CCSID로 변환할 필요가 없을 때. 예를 들면, 한 CCSID가 65535일 경우

연속 효과

열을 추가해도 SQL 뷰나 대부분의 논리 파일에 연속 효과를 낼 수 없습니다.[43] 예를 들면, 종속 뷰가 SELECT *절로 작성된 경우에도, 표에 열을 추가해도 열은 종속 뷰에 추가되지 않습니다.

열을 제거하거나 변경해도 몇 가지 연속 효과를 낼 수 있습니다. 표 41은 열을 제거하는 연속 효과를 나열합니다.

표 41. 열을 제거하는 연속 효과

조작	**RESTRICT 효과**	**CASCADE 효과**
뷰에 의해 참조되는 열 제거	열 제거가 허용되지 않습니다.	뷰와 그 뷰에 종속된 모든 뷰가 제거됩니다.

42. 재구축되어야 하는 색인은 데이터베이스 서버 작업에 의해 비동기식으로 재구축됩니다.

43. 열이 실제 파일에 추가될 때 실제 파일의 형식을 공유하는 논리 파일에도 추가됩니다(다른 기본 파일이 있는 논리 파일에 형식이 다시 사용되지 않는 경우).

표 41. 열을 제거하는 연속 효과 (계속)

조작	RESTRICT 효과	CASCADE 효과
뷰가 아닌 논리 파일에 의해 참조되는 열 제거	제거가 허용되며, 다음 경우에 열은 논리 파일에서 제거됩니다. • 논리 파일이 변경되는 파일과 형식을 공유하고, • 제거된 열이 키 필드로 또는 선택/생략 스펙에서 사용되지 않으며, • 다른 기본 파일과 함께 논리 파일에서 그 형식이 다시 사용되지 않는 경우 그렇지 않으면 열 제거가 허용되지 않습니다.	제거가 허용되며, 다음 경우에 열은 논리 파일에서 제거됩니다. • 논리 파일이 변경되는 파일과 형식을 공유하고, • 제거된 열이 키 필드로 또는 선택이나 생략 스펙에서 사용되지 않으며, • 다른 기본 파일과 함께 논리 파일에서 그 형식이 다시 사용되지 않는 경우 이 외의 경우에는 논리 파일이 제거됩니다.
색인 키에서 참조되는 열 제거	색인 제거가 허용되지 않습니다.	색인이 제거됩니다.
고유 제한에서 참조되는 열 제거	고유 제한에서 참조되는 모든 열이 같은 ALTER COLUMN문에서 제거되고 고유 제한이 참조 제한에 의해 참조되지 않으면 열과 제한이 제거됩니다(따라서, 제한을 만족시키기 위해 사용된 색인도 제거됩니다). 예를 들면, 열 A가 제거되고 UNIQUE(A)나 PRIMARY KEY(A)의 고유 제한이 존재하고 고유 제한을 참조하는 참조 제한이 없으면, 조작이 허용됩니다. 그렇지 않으면 열 제거가 허용되지 않습니다.	고유 제한은 이 고유 제한을 참조하는 참조 제한이 제거될 경우에 제거됩니다(따라서, 이 제한에 의해 사용된 색인도 제거됩니다).
참조 제한에서 참조되는 열 제거	참조 제한에서 참조되는 모든 열도 동시에 제거되면 열과 제한이 제거됩니다(따라서, 외부 키에 의해 사용된 색인도 제거됩니다). 예를 들어, 열 B가 제거되고 PRIMARY KEY(A) 참조 제한이 존재하면 조작이 허용됩니다. 그렇지 않으면 열 제거가 허용되지 않습니다.	참조 제한이 제거됩니다(따라서, 외부 키에 의해 사용된 색인도 제거됩니다).

표 42는 열을 변경하는 연속 효과를 나열합니다(다음 도표의 열을 변경하는 것은 자료 유형, 정밀도, 눈금, 길이 또는 널이 될 수 있는 특성을 변경하는 것입니다).

표 42. 열 변경의 연속 효과

조작	효과
뷰에 의해 참조되는 열 변경	변경이 허용됩니다. 표에 종속된 뷰는 재작성됩니다. 새로운 열 속성은 뷰를 재작성할 때 사용됩니다.

표 42. 열 변경의 연속 효과 (계속)

조작	효과
뷰가 아닌 논리 파일에 의해 참조되는 열 변경	변경이 허용됩니다. 표에 종속되는 뷰가 아닌 논리 파일이 재작성됩니다. 논리 파일이 변경되는 파일과 형식을 공유하고 그 형식이 다른 기본 파일과 함께 논리 파일에서 다시 사용되지 않으면 새로운 열 속성은 논리 파일이 재작성될 때 사용됩니다. 그렇지 않으면 새로운 열 속성은 논리 파일을 재작성할 때 사용되지 않습니다. 그 대신, 현재 논리 파일 속성이 사용됩니다.
색인 키에서 참조되는 열 변경	변경이 허용됩니다. (따라서, 색인은 일반적으로 재구축됩니다.)
고유 제한에서 참조되는 열 변경	변경이 허용됩니다. (따라서, 색인은 일반적으로 재구축됩니다.) 고유 제한이 참조 제한에 의해 참조되면 외부 키의 속성은 더 이상 고유 제한의 속성과 일치하지 않습니다. 제한은 정의된 그리고 검사 지연 중인 상태에 있게 됩니다.
참조 제한에서 참조되는 열 변경	변경이 허용됩니다. • 참조 제한이 정의된 그러나 검사 지연 중인 상태이면 변경이 허용되고 제한을 작동할 수 있는 상태로 만들려는 시도가 이루어집니다(따라서, 고유 제한을 만족시키기 위해 사용된 색인이 일반적으로 재구축됩니다). • 참조 제한이 작동할 수 있는 상태에 있으면, 제한은 정의된 그리고 검사 지연 중인 상태에 있게 됩니다.

예

예 1

다음 열을 갖는 새로운 표 EQUIPMENT가 작성되었다고 가정합니다.

열 이름	자료 유형
EQUIP_NO	INT
EQUIP_DESC	VARCHAR(50)
LOCATION	VARCHAR(50)
EQUIP_OWNER	CHAR(3)

소유자(EQUIP_OWNER)가 DEPARTMENT 표에 있는 부서 번호(DEPTNO)가 되도록 EQUIPMENT 표에 참조 제한을 추가합니다. 부서가 DEPARTMENT 표에서 제거되면 해당 부서가 소유하고 있는 모든 장비의 소유자(EQUIP_OWNER) 값은 지정되지 않은 상태가 되거나 널(null)로 설정되어야 합니다. 제한의 이름을 DEPTQUIP로 지정합니다. DEPARTMENT 표가 (DEPTNO)로 정의된 1차 키를 가지고 있다고 가정합니다.

```
ALTER TABLE EQUIPMENT
  ADD CONSTRAINT DEPTQUIP
   FOREIGN KEY (EQUIP_OWNER)
   REFERENCES DEPARTMENT
   ON DELETE SET NULL
```

예 2

첫 번째 예처럼 같은 EQUIPMENT 표가 존재한다고 가정합니다.

- 각 장비 번호의 양이 포함된 EQUIPMENT 표에 열을 추가합니다. 열을 QUANTITY 라고 합니다.

```
ALTER TABLE EQUIPMENT
  ADD COLUMN QUANTITY INT
```

- EQUIP_OWNER 열에 대한 디폴트 값을 'ABC'로 변경합니다.

```
ALTER TABLE EQUIPMENT
  ALTER COLUMN EQUIP_OWNER
  SET DEFAULT 'ABC'
```

- LOCATION 열을 제거합니다. 그 열에 구축된 모든 뷰, 색인 또는 제한사항도 제거합니다.

```
ALTER TABLE EQUIPMENT
  DROP COLUMN LOCATION CASCADE
```

- SUPPLIER라고 하는 새로운 열을 추가하고, 기존의 LOCATION 열을 제거하고, 새로운 SUPPLIER 열에 대한 고유 제한을 추가하고, 기존의 EQUIP_NO 열에 1차 키가 구축되도록 표를 변경합니다.

```
ALTER TABLE EQUIPMENT
  ADD COLUMN SUPPLIER INT
  DROP COLUMN LOCATION
  ADD UNIQUE SUPPLIER
  ADD PRIMARY KEY EQUIP_NO
```

- EQUIP_DESC 열은 가변 길이 열입니다. 할당된 길이가 25로 지정되었으면 다음 ALTER TABLE문은 할당 길이를 변경하지 않습니다.

```
ALTER TABLE EQUIPMENT
  ALTER COLUMN EQUIP_DESC
  SET DATA TYPE VARCHAR(60)
```

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION문은 SQL 선언 섹션의 시작을 표시합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. RPG, Java또는 REXX에 지정되어서는 안됩니다.

권한부여

필요한 사항이 없습니다.

구문

설명

변수 선언이 호스트 언어의 규칙에 따라 나타날 수 있는 어플리케이션 프로그램에서 BEGIN DECLARE SECTION문은 코드화될 수 있습니다. 호스트 구조 선언 중에 코드화될 수는 없습니다. 명령문은 SQL 선언 섹션의 시작을 표시하는 데 사용됩니다.

SQL 선언 섹션은 END DECLARE SECTION으로 끝납니다. END DECLARE SECTION문에 대한 자세한 정보는 668 페이지의 『END DECLARE SECTION』을 참조하십시오.

BEGIN DECLARE SECTION과 END DECLARE SECTION문은 쌍을 이루어야 하며, 내포될 수 없습니다.

DECLARE VARIABLE 및 INCLUDE문을 제외한 SQL문은 선언 섹션에 포함되어서는 안됩니다.

SQL 선언 섹션이 프로그램에 지정되면 SQL 선언 섹션 내에 선언된 변수만 호스트 변수로 사용될 수 있습니다. SQL 선언 섹션이 프로그램에 지정되지 않으면 프로그램의 모든 변수는 호스트 변수로 사용하기에 적합합니다.

소스 프로그램이 SQL의 IBM SQL 표준에 따르도록 SQL 선언 섹션은 RPG와 REXX 이외의 호스트 언어에 대해 지정되어야 합니다. SQL 선언 섹션은 C++의 모든 호스트 변수에 대해 필수입니다. SQL 선언 섹션은 변수에 대한 첫 번째 참조 이전에 나타나야 합니다. 호스트 변수는 JAVA 또는 RPG에서 이 명령문을 사용하지 않고 선언되며, REXX에서는 전혀 선언되지 않습니다.

SQL 선언 섹션 외부에 선언된 변수는 SQL 선언 섹션 내부에 선언된 변수와 같은 이름일 수 없습니다.

하나 이상의 SQL 선언 섹션이 프로그램에 지정될 수 있습니다.

예

예 1

C 프로그램에서 호스트 변수 hv_smint(SMALLINT), hv_vchar24(VARCHAR(24)) 및 hv_double(FLOAT)을 정의합니다.

```
EXEC SQL  BEGIN DECLARE SECTION;
  static short                          hv_smint;
  static struct {
           short hv_vchar24_len;
           char  hv_vchar24_value[24];
         }                              hv_vchar24;
  static double                         hv_double;
EXEC SQL  END DECLARE SECTION;
```

예 2

COBOL 프로그램에서 호스트 변수 HV-SMINT(SMALLINT), HV-VCHAR24 (VARCHAR(24)), HV-DEC72(DECIMAL(7,2))를 정의합니다.

```
WORKING-STORAGE SECTION.
EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.
  01  HV-SMINT              PIC S9(4)       BINARY.
  01  HV-VCHAR24.
      49 HV-VCHAR24-LENGTH  PIC S9(4)       BINARY.
      49 HV-VCHAR24-VALUE   PIC X(24).
  01  HV-DEC72              PIC S9(5)V9(2)  PACKED-DECIMAL.
EXEC SQL  END DECLARE SECTION  END-EXEC.
```

CALL

CALL문은 프로시듀어를 호출합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

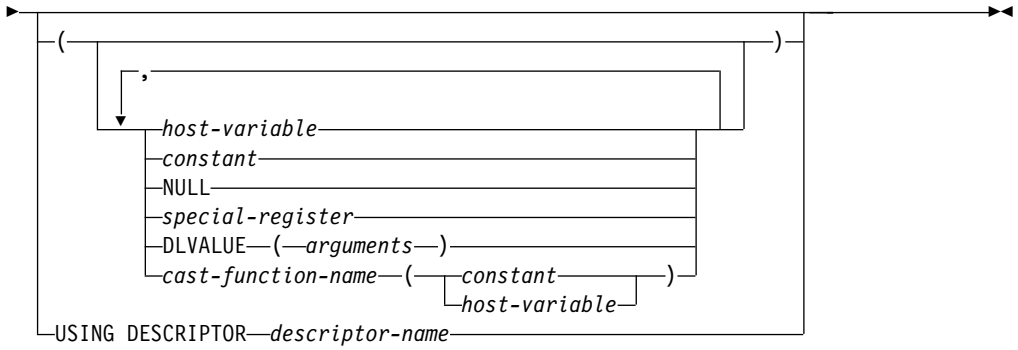
명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 프로시듀어가 REXX 외부 프로시듀어인 경우:
  - 프로시듀어와 연관된 소스 파일의 시스템 권한 *OBJOPR, *READ 및 *EXECUTE
  - 소스 파일이 들어 있는 라이브러리의 시스템 권한 *EXECUTE
  - CL 명령에 대한 시스템 권한 *USE
- 프로시듀어가 Java 외부 프로시듀어인 경우:
  - Java 클래스가 들어있는 통합 파일 시스템 파일에 대한 읽기 권한(*R)
  - 통합 파일 시스템 파일을 찾기 위해 액세스해야 하는 모든 디렉토리에 대한 읽기 및 실행 권한(*RX)
- 프로시듀어가 외부 프로시듀어이지만 REXX 또는 Java 외부 프로시듀어가 아닌 경우
  - 프로시듀어와 연관된 프로그램의 시스템 권한 *EXECUTE
  - 프로시듀어와 연관된 프로그램이 들어 있는 라이브러리의 시스템 권한 *EXECUTE
- 프로시듀어가 SQL 프로시듀어인 경우:
  - 프로시듀어에서 EXECUTE 권한
  - SQL 프로시듀어가 들어 있는 라이브러리의 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 프로시듀어에 대한 EXECUTE 권한을 갖습니다.

- 프로시듀어의 소유자입니다.
- 프로시듀어에서 EXECUTE 권한을 부여받았습니다.
- 프로시듀어에서 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문



설명

procedure-name 또는 *host-variable*

지정된 프로시듀어명이나 호스트 변수에 들어 있는 프로시듀어명에 의해 호출될 프로시듀어를 식별합니다. procedure-name은 현재 서버에 있는 프로시듀어를 식별해야 합니다. 호스트 변수가 지정된 경우

- 문자 스트링 변수 또는 UCS-2 그래픽 스트링이어야 하며 인디케이터 변수를 포함하지 않아야 합니다.
- 호스트 변수에 들어 있는 프로시듀어명은 왼쪽부터 배열되어야 하며 길이가 호스트 변수보다 짧으면 오른쪽은 공백으로 채워져야 합니다.
- 프로시듀어명이 구분된 이름이 아니면 대문자로 표시되어야 합니다.

프로시듀어명이 규정되지 않은 경우 매개변수의 경로와 번호를 기준으로 내재적으로 규정되어야 합니다. 자세한 내용은 54 페이지의 『규정되지 않은 오브젝트명의 규정화』를 참조하십시오.

procedure-name이 CREATE PROCEDURE나 DECLARE PROCEDURE문으로 정의된 프로시듀어를 식별하고 현재 서버가 iSeries용 DB2 UDB 서버인 경우

- CREATE PROCEDURE나 DECLARE PROCEDURE문은 외부 프로그램, 언어 및 호출 규칙의 이름을 판별합니다.
- 프로시듀어 매개변수의 속성은 CREATE PROCEDURE나 DECLARE PROCEDURE문으로 정의됩니다.

그렇지 않은 경우

- 현재 서버는 외부 프로그램, 언어 및 호출 규칙의 이름을 판별합니다.
- 현재 서버가 iSeries용 DB2 UDB인 경우
  - 외부 프로그램명은 외부 프로시듀어명과 동일하다고 가정됩니다.
  - 프로그램과 연관된 프로그램 속성 정보가 인식할 수 있는 언어를 식별하는 경우 그 언어가 사용됩니다. 그렇지 않은 경우에는 언어가 C로 가정됩니다.
  - 호출 규칙은 GENERAL로 가정됩니다.

- 어플리케이션 리퀘스터는 호스트 변수인 모든 매개변수 또는 매개변수 마커가 INOUT인 것으로 가정합니다. 호스트 변수가 아닌 모든 매개변수는 모두 IN으로 가정됩니다.
- 매개변수의 실제 속성은 현재 서버에 의해 판별됩니다.

  현재 서버가 iSeries용 DB2 UDB이면 매개변수의 속성은 CALL문에 지정된 인수의 속성과 같습니다.[44]

host-variable 또는 *constant* 또는 **NULL** 또는 *special-register*

프로시듀어에 매개변수로 전달될 값의 리스트를 식별합니다. n번째 값은 프로시듀어의 n번째 매개변수에 해당합니다.

CALL문이 실행될 때 각 매개변수 값이 프로시듀어의 해당하는 매개변수에 할당됩니다. 호스트 언어의 호출 규칙에 따라서 제어가 프로시듀어로 전달됩니다. 프로시듀어의 실행이 완료될 때 프로시듀어의 각 매개변수 값이 OUT 또는 INOUT으로 정의되는 CALL문의 해당 매개변수에 할당됩니다. 매개변수 지정에 사용되는 규칙에 대한 자세한 정보는 83 페이지의 『스트링 지정』을 참조하십시오.[45]

DLVALUE(*arguments***)**

매개변수 값을 DLVALUE 스칼라 함수의 결과 값으로 지정합니다. DLVALUE 스칼라 함수는 DataLink 매개변수에 대해서만 지정될 수 있습니다. DLVALUE 함수는 스키마, 서버 및 경로/파일 삽입 시 링크 값을 필요로 합니다. DLVALUE의 첫 번째 인수는 상수, 호스트 변수 또는 입력된 매개변수 마커(CAST(? AS 자료 유형))여야 합니다. DLVALUE의 두 번째 및 세 번째 인수는 상수나 host-variables 여야 합니다.

cast-function-name

이러한 인수 형식은 고유한 유형, BLOB, CLOB, DBCLOB, DATE, TIME 또는 TIMESTAMP 자료 유형으로 정의된 매개변수와 함께만 사용할 수 있습니다. 다음 표는 이 *cast-function*이 허용되는 사용법을 설명한 것입니다.

parameter-type	**캐스트 함수명**
BLOB, CLOB 또는 DBCLOB를 기초로 하는 고유한 유형 N	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP를 기초로 하는 고유한 유형 N	DATE, TIME 또는 TIMESTAMP *
BLOB, CLOB 또는 DBCLOB	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP	DATE, TIME 또는 TIMESTAMP *

주:

* 함수 이름은 내재적 또는 명시적 스키마명 QSYS2를 갖는 자료 유형의 이름(또는 고유한 유형의 소스 유형과 일치해야 합니다.

44. 소수 상수 경우에는 인수의 속성을 판별할 때 선행 제로(leading zeros)가 중요합니다. 일반적으로, 선행 제로(leading zeros)는 중요하지 않습니다.

45. CALL문이 준비된 후 삽입 SQL EXECUTE문에 의해 실행되는 경우 OUT 및 INOUT 매개변수는 지정되지 않습니다.

constant

상수를 인수로 지정합니다. 상수는 고유한 유형의 소스 유형 또는 고유한 유형이 아닌 경우 자료 유형에 대한 상수 규칙을 따라야 합니다. BLOB, CLOB, DBCLOB, DATE, TIME 및 TIMESTAMP 함수의 경우 상수는 스트링 상수이어야 합니다.

host-variable

호스트 변수를 인수로 지정합니다. 호스트 변수는 고유한 유형의 소스 유형 또는 고유한 유형이 아닌 경우 자료 유형에 대한 상수 규칙을 따라야 합니다.

USING DESCRIPTOR *descriptor-name*

호스트 변수의 유효한 설명을 포함해야 하는 SQLDA를 식별합니다.

CALL문을 처리하기 전에 SQLDA에 다음 필드를 설정해야 합니다(REXX에 대한 규칙은 다릅니다. 자세한 내용은 SQL Programming with Host Languages 책을 참조하십시오).

- SQLDA에 제공된 SQLVAR 발생 수를 표시하는 SQLN
- SQLDA에 대해 할당된 기억장치의 바이트 수를 표시하는 SQLDABC
- 명령문을 처리할 때 SQLDA에서 사용된 변수 수를 표시하는 SQLD
- 변수 속성을 표시하는 SQLVAR 발생

모든 SQLVAR 발생을 포함할 수 있도록 SQLDA는 충분한 기억장치 공간을 가져야 합니다. 따라서, SQLDABC의 값은 16 + SQLN*(80) 보다 크거나 같아야 합니다. 여기서 80은 SQLVAR 발생의 길이입니다. LOB나 고유한 유형이 지정되는 경우 각 매개변수 마커에 대해 두 SQLVAR 항목이 있어야 하며 SQLN이 매개변수 마커 숫자의 두 배로 설정되어야 합니다.

SQLD는 영(0) 보다 크거나 같고 SQLN 보다 작거나 같은 값으로 설정되어야 합니다. CALL문에 있는 매개변수 마커 수와 같아야 합니다. SQLDA에 의해 설명되는 n 번째 변수는 준비된 명령문에서 n 번째 매개변수 마커에 해당합니다. (SQLDA에 대한 설명은 881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』를 참조하십시오.)

RPG/400은 포인터 설정을 위한 함수를 제공하지 않습니다. SQLDA는 해당 호스트 변수를 위치지정하는 데 포인터를 사용하므로 사용자는 RPG/400 어플리케이션 밖에서 이 포인터를 설정해야 합니다.

주

*procedure-name*이 CREATE PROCEDURE나 DECLARE PROCEDURE문으로 정의된 프로시듀어를 식별하는 경우 OUT이나 INOUT 매개변수는 호스트 변수로 지정되어야 합니다.

*procedure-name*이 CREATE PROCEDURE나 DECLARE PROCEDURE문으로 정의된 프로시듀어를 식별하는 경우 지정된 인수 수는 CREATE PROCEDURE나 DECLARE PROCEDURE문으로 정의된 매개변수 수와 같아야 합니다.

*상수*와 *host-variable*에 대한 설명은 99 페이지의 『상수』 및 114 페이지의 『호스트 변수에 대한 참조』를 참조하십시오. *special-register*에 대한 설명은 104 페이지의 『특수 레지스터』를 참조하십시오. NULL은 널값을 지정합니다.

호출될 외부 프로시듀어가 REXX 프로시듀어이면 CREATE PROCEDURE나 DECLARE PROCEDURE문을 사용하여 그 프로시듀어를 선언해야 합니다.

호스트 변수는 REXX 프로시듀어 내의 CALL문에 사용될 수 없습니다. 대신, 매개변수 마커를 사용하여 CALL을 PREPARE와 EXECUTE의 오브젝트로 만들어야 합니다.

SQL이나 외부 프로시듀어가 호출되면 속성은 프로시듀어가 작성되었을 때 정의된 SQL data-access에 대해 설정됩니다. 속성으로 사용할 수 있는 값은 다음과 같습니다.

```
NONE
CONTAINS
READS
MODIFIES
```

두 번째 프로시듀어가 현재 프로시듀어 실행 내에서 호출되면 다음과 같은 경우에 오류가 발행됩니다.

- 호출된 프로시듀어에는 SQL이 들어 있을 수 있고 호출하는 프로시듀어는 SQL을 허용하지 않는 경우
- 호출된 프로시듀어는 SQL 자료를 읽고 호출하는 프로시듀어는 SQL 자료를 읽도록 허용하지 않는 경우
- 호출된 프로시듀어는 SQL 자료를 수정하고 호출하는 프로시듀어는 SQL 자료를 수정하도록 허용하지 않는 경우

결과 세트는 iSeries Access ODBC(개방 데이터베이스 연결성) 드라이버를 사용하는 클라이언트, iSeries Access Optimized SQL API를 사용하는 클라이언트 또는 SQL 호출 레벨 인터페이스나 JDBC로부터 프로시듀어가 호출될 때 해당 프로시듀어에서만 리턴됩니다. 결과 세트가 프로시듀어로부터 리턴되는 방법에는 다음의 세 가지가 있습니다.

- 프로시듀어에서 SET RESULT SETS문을 실행할 때 SET RESULT SETS문이 결과 세트를 식별합니다. 결과 세트는 SET RESULT SETS문에 지정한 순서로 리턴됩니다.
- SET RESULT SETS문이 프로시듀어에서 실행되지 않는 경우

- WITH RETURN절을 지정한 커서가 없는 경우 프로시듀어가 열고 리턴시 열린 상태로 그대로 놓아 둔 각 커서가 결과 세트를 식별합니다. 결과 세트는 커서가 열린 순서로 리턴됩니다.
- WITH RETURN절을 지정한 커서가 있는 경우 프로시듀어가 열고 리턴시 열린 상태로 그대로 놓아 둔 WITH RETURN절을 사용하여 정의된 각 커서가 결과 세트를 식별합니다. 결과 세트는 커서가 열린 순서로 리턴됩니다.

열린 커서를 사용하여 결과 세트가 리턴될 때 행은 현재 커서 위치에서 시작하여 리턴됩니다.

CALL문 내포

프로시듀어, 사용자 정의 함수 또는 트리거로 실행되는 프로그램은 CALL문을 발행할 수 있습니다. 프로시듀어, 사용자 정의 함수 또는 트리거가 프로시듀어를 호출할 때 호출은 내포된 것으로 간주됩니다. 프로시듀어와 함수가 내포될 수 있는 레벨 수에는 제한이 없지만, 트리거는 300 레벨까지만 내포될 수 있습니다.

프로시듀어가 조회 결과 세트를 리턴하면 결과 세트는 프로시듀어의 호출자에게 리턴됩니다. SQL CALL문이 내포되면 결과 세트는 이전 내포 레벨에 있는 프로그램에서만 볼 수 있습니다. 예를 들어, 클라이언트 프로그램이 프로시듀어 PROCA를 호출하고 이 프로시듀어는 프로시듀어 PROCB를 호출합니다. PROCA만 PROCB가 리턴하는 결과 세트에 액세스할 수 있으며, 클라이언트 프로그램은 조회 결과 세트에 액세스할 수 없습니다.

예

프로시듀어 PGM1을 호출하고 매개변수 두 개를 전달합니다.

```
CALL PGM1 (:hv1,:hv2)
```

CLOSE

CLOSE문은 커서를 닫습니다. 커서가 열렸을 때 결과표가 작성되었으면 그 표는 삭제됩니다.

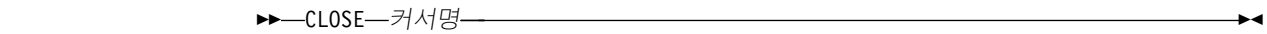
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. JAVA에 지정되어서는 안됩니다.

권한부여

필요한 사항이 없습니다. 커서를 사용하는 데 필요한 권한부여에 대한 내용은 598 페이지의 『DECLARE CURSOR』를 참조하십시오.

구문



설명

cursor-name

닫을 커서를 식별합니다. *cursor-name*은 DECLARE CURSOR문에 설명된 대로 선언된 커서를 식별해야 합니다. CLOSE문이 실행되면 커서는 열린 상태에 있어야 합니다.

주

프로그램에 있는 모든 커서는 다음과 같은 경우에 닫힌 상태에 있습니다.

- 프로그램이 호출된 경우
  - CLOSQLCSR(*ENDPGM)이 지정된 경우 모든 커서는 프로그램이 호출될 때마다 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDSQL)이 지정된 경우 한 SQL 프로그램이 호출 스택에 남아 있는 한, 프로그램이 처음 호출될 때만 모든 커서가 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDJOB)이 지정된 경우 프로그램이 작업에서 처음 호출될 때만 모든 커서가 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDMOD)가 지정된 경우 모듈이 개시될 때마다 모든 커서는 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDACTGRP)가 지정된 경우 프로그램에 있는 모듈이 활성 그룹 내에서 처음 개시될 때 모든 커서가 닫힌 상태에 있습니다.

- 프로그램은 HOLD 옵션없이 COMMIT나 ROLLBACK문을 실행하여 새로운 작업 단위를 시작합니다. HOLD 옵션으로 선언된 커서는 COMMIT문으로 닫히지 않습니다.

주: iSeries용 DB2 UDB 데이터베이스 관리자는 조회를 실행하기 위해 파일을 엽니다. 파일을 닫는 것을 SQL CLOSE문과 분리할 수 있습니다. 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.

가능한 빨리 명시적으로 커서를 닫으면 성능이 향상될 수 있습니다.

예

COBOL 프로그램에서 커서 C1을 사용하여 EMPPROJACT 표의 처음 네 열에서 한 번에 한 행씩 값을 페치하여 그 값을 다음 호스트 변수로 가져갑니다.

- EMP(CHAR(6))
- PRJ(CHAR(6))
- ACT(SMALLINT)
- TIM(DECIMAL(5,2))

마지막으로 커서를 닫습니다.

```
   EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.
     77  EMP              PIC X(6).
     77  PRJ              PIC X(6).
     77  ACT              PIC S9(4) BINARY.
     77  TIM              PIC S9(3)V9(2) PACKED-DECIMAL.
   EXEC SQL  END DECLARE SECTION  END-EXEC.
   .
   .
   .

   EXEC SQL  DECLARE C1 CURSOR FOR
               SELECT EMPNO, PROJNO, ACTNO, EMPTIME
                 FROM EMPPROJACT                        END-EXEC.

   EXEC SQL  OPEN C1  END-EXEC.

   EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM  END-EXEC.

   IF SQLSTATE = '02000'
     PERFORM DATA-NOT-FOUND
ELSE     PERFORM GET-REST UNTIL SQLSTATE IS NOT EQUAL TO '00000'.

   EXEC SQL  CLOSE C1  END-EXEC.

   GET-REST
   EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM  END-EXEC.
   .
   .
   .
```

COMMENT

COMMENT문은 다양한 데이터베이스 오브젝트의 카탈로그 설명에 주석을 추가하거나 대체합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

표, 뷰, 별명, 색인, 열, 고유한 유형, 또는 패키지에 주석을 작성하려면 명령문의 권한부여 ID가 보유하는 권한이 적어도 다음 중 하나를 포함해야 합니다.

- 명령문의 표, 뷰, 별명, 색인, 고유한 유형이나 패키지의 경우
  - 표, 뷰, 별명, 색인, 고유한 유형이나 패키지에 대한 ALTER 권한, 및
  - 표, 뷰, 별명, 색인, 고유한 유형이나 패키지가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

다음 경우에 명령문의 권한부여 ID는 표, 뷰, 별명, 색인, 고유한 유형 또는 패키지에 대한 ALTER 권한을 갖습니다.

- 표, 뷰, 별명, 색인, 고유한 유형이나 패키지의 소유자인 경우
- 표, 뷰, 별명, 고유한 유형이나 패키지에 대해 ALTER 권한을 부여받은 경우
- 표, 뷰, 별명, 색인, 고유한 유형이나 패키지에 대해 *OBJALTER 또는 *OBJMGT의 시스템 권한을 부여받은 경우

트리거에 주석을 작성하려면 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 명령문의 트리거 주제 표의 경우
  - 주제 표의 ALTER 권한 및
  - 주제 표가 들어 있는 라이브러리에 대한 *EXECUTE 시스템 권한
- 관리 권한

함수에 주석을 작성하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSFUNCS 카탈로그 뷰의 경우
  - 뷰에서 UPDATE 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

프로시듀어에 주석을 작성하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSPROCS 카탈로그 뷰의 경우
  - 뷰에서 UPDATE 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

매개변수에 주석을 작성하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

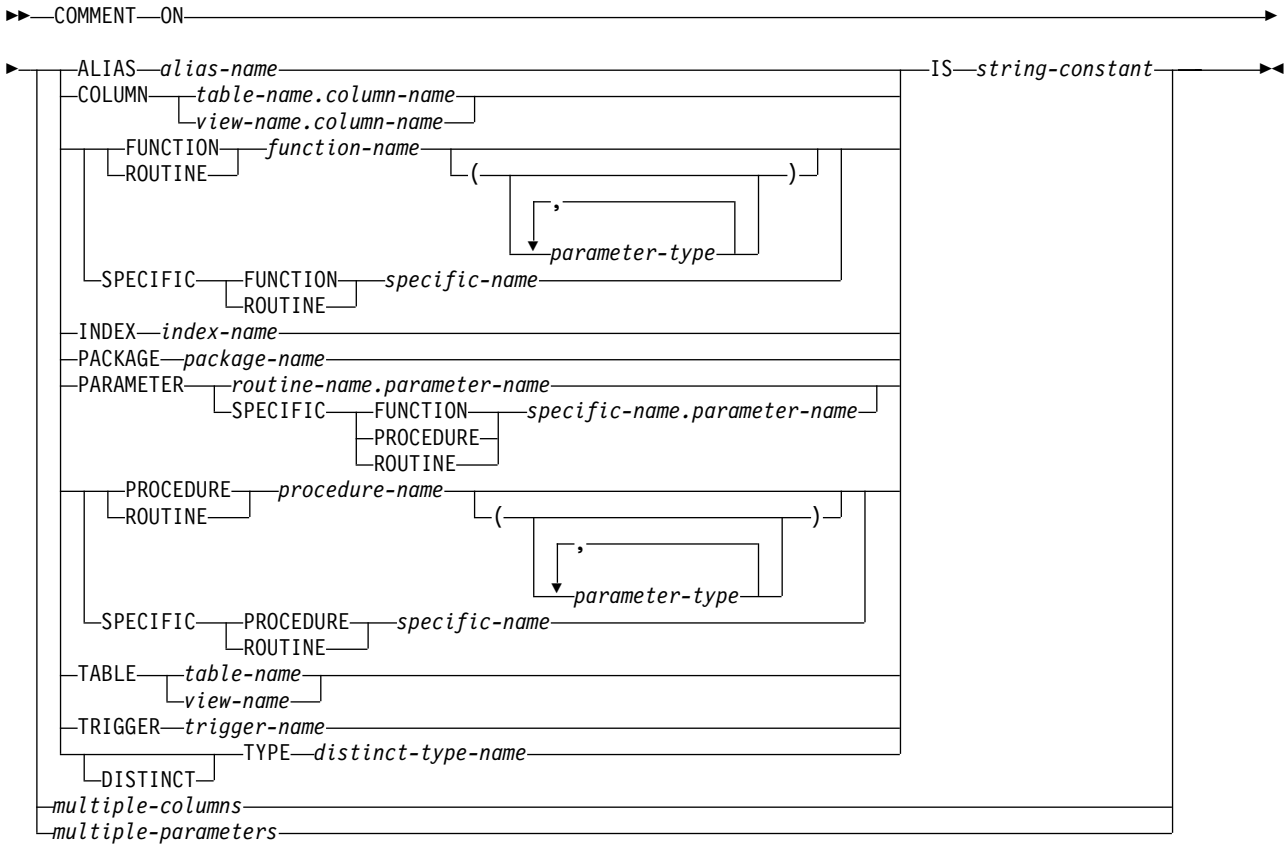
- SYSPARMS 카탈로그 표의 경우
  - 표에서 UPDATE 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 표에 대해 UPDATE 권한을 갖습니다.

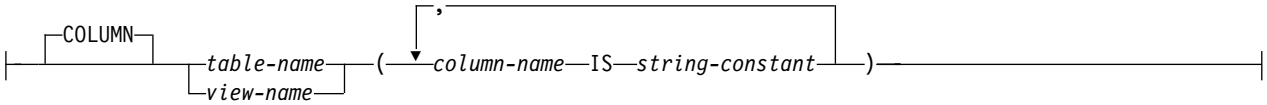
- 표의 소유자인 경우
- 표에 대한 UPDATE 권한을 부여받은 경우
- 표에 대해 *OBJOPR과 *UPD의 시스템 권한을 부여받은 경우

COMMENT

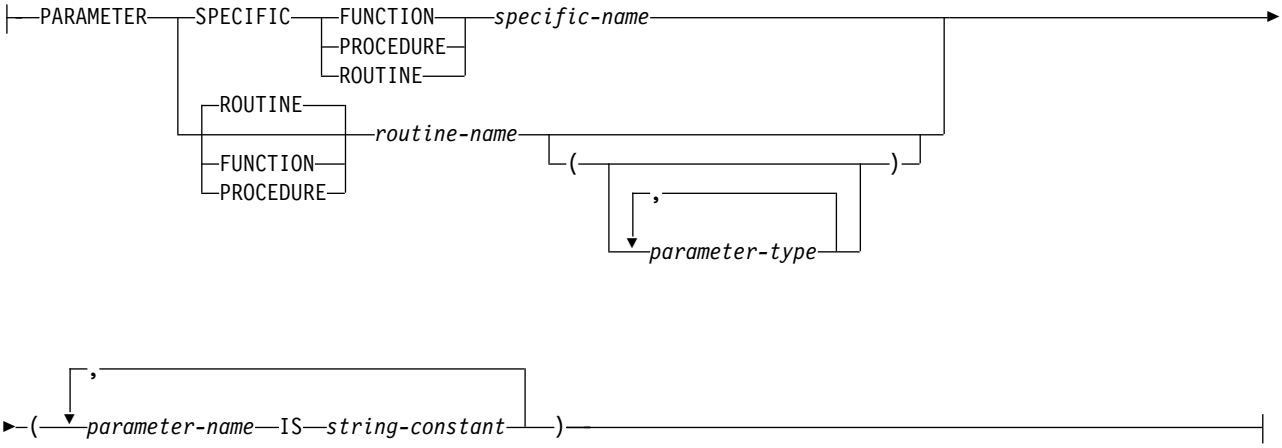
구문


COMMENT ON
ALIAS alias-name
COLUMN table-name.column-name
view-name.column-name
FUNCTION
ROUTINE
function-name
(
,
parameter-type
)
SPECIFIC
FUNCTION
ROUTINE
specific-name
INDEX index-name
PACKAGE package-name
PARAMETER routine-name.parameter-name
SPECIFIC
FUNCTION
PROCEDURE
ROUTINE
specific-name.parameter-name
PROCEDURE
ROUTINE
procedure-name
TABLE table-name
view-name
TRIGGER trigger-name
TYPE distinct-type-name
DISTINCT
IS string-constant
multiple-columns
multiple-parameters

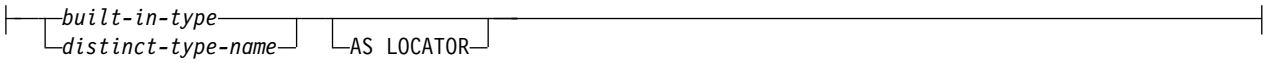
multiple-columns:



multiple-parameters:

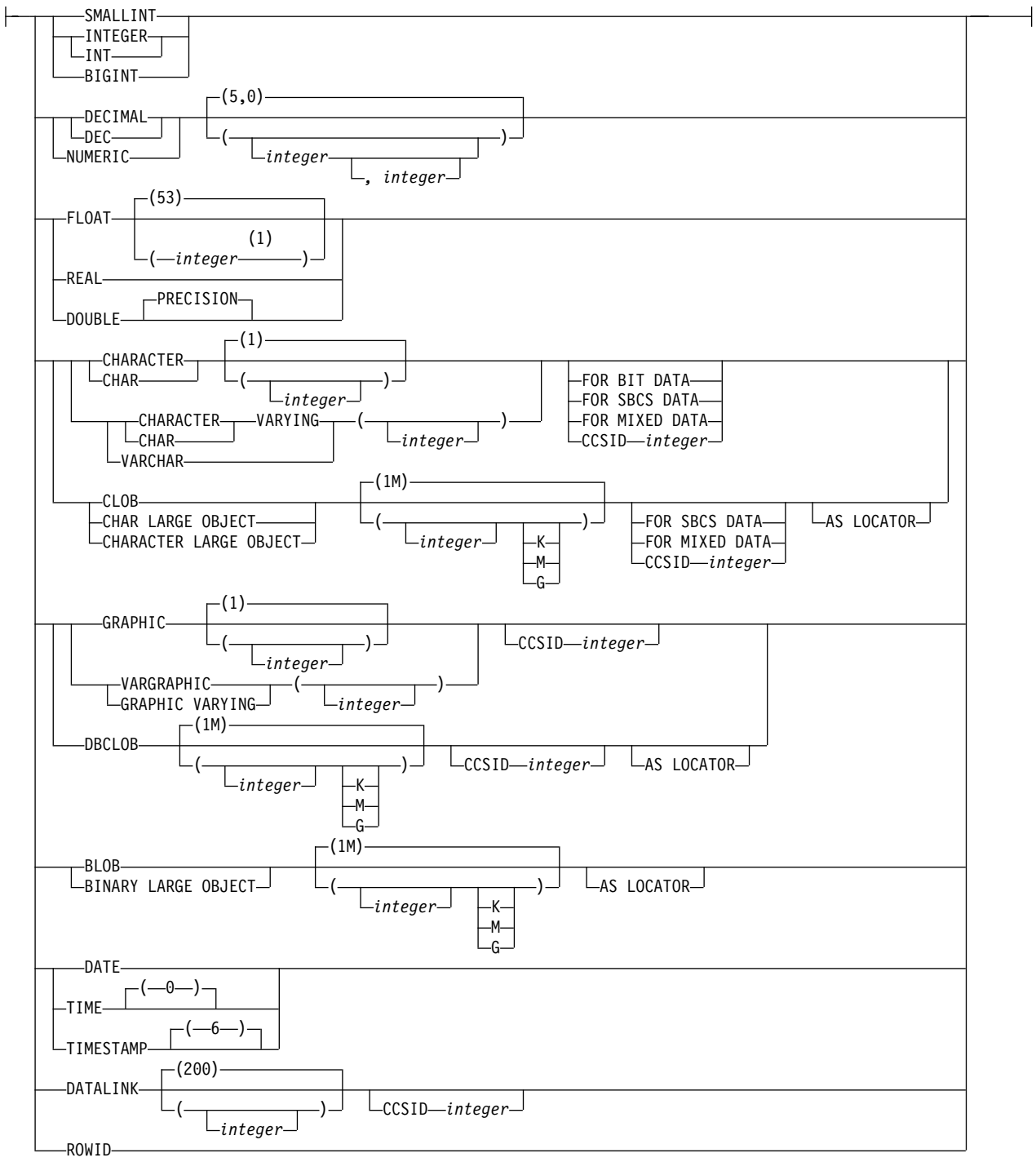


parameter-type:



COMMENT

built-in-type:



주:

1 자료 유형(REAL 또는 DOUBLE)을 기초로 일치가 이루어지므로 정밀도에 대해 지정한 값이 함수를 작성할 때 지정한 값과 일치하지 않아도 됩니다.

설명

ALIAS *alias-name*

주석이 적용되는 별명을 식별합니다. 이름은 현재 서버에 있는 별명을 식별해야 합니다.

COLUMN

주석이 열에 대해 추가되거나 대체됨을 지정합니다.

table-name.column-name 또는 *view-name.column-name*

주석이 적용되는 열을 식별합니다. *table-name* 또는 *view-name*은 현재 서버에 있는 표나 뷰를 식별하지만 글로벌 임시 표를 식별해서는 안됩니다. *column-name*은 그 표나 뷰의 열을 식별해야 합니다.

DISTINCT TYPE *distinct-type-name*

주석이 적용되는 고유한 유형을 식별합니다. 이름은 현재 서버에 있는 고유한 유형을 식별해야 합니다.

FUNCTION

주석이 함수에 대해 추가되거나 대체됨을 지정합니다. 주석이 적용되는 함수를 식별합니다. 이름, 함수 표시나 특정 이름으로 특별한 함수를 식별할 수 있습니다. 함수 해석의 규칙(그리고 SQL 경로)은 사용되지 않습니다.

FUNCTION *function-name*

*function-name*은 현재 서버에 있는 한 함수를 정확히 식별해야 합니다. 함수는 함수에 대해 정의된 매개변수를 가질 수 있습니다. 지정된 또는 내재된 스키마에 지정된 이름의 함수가 둘 이상 있으면 오류가 리턴됩니다.

FUNCTION *function-name(parameter-type, ...)*

function-name(parameter-type, ...) 현재 서버에 있는 지정된 함수 표시로 함수를 식별해야 합니다. 지정된 매개변수는 대응하는 위치의 CREATE FUNCTION문에 지정되어 있는 자료 유형과 일치해야 합니다. 자료 유형의 수와 논리적 연결은 주석이 적용되는 특정 함수 인스턴스를 식별하기 위해 사용됩니다. *function-name()*이 지정되면 식별된 함수는 0개의 매개변수를 가져야 합니다.

function-name

함수 이름을 식별합니다.

(parameter-type, ...)

함수의 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.
- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE FUNCTION문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

  ```
  CHAR      CHAR(1)
  GRAPHIC   GRAPHIC(1)
  DECIMAL   DECIMAL(5,0)
  FLOAT     DOUBLE (길이 8)
  ```

  내재된 길이는 CREATE FUNCTION문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE FUNCTION문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC FUNCTION *specific-name*

*specific-name*은 현재 서버에 있는 특정 함수를 식별해야 합니다.

INDEX

주석이 색인에 대해 추가되거나 대체됨을 지정합니다.

index-name

주석이 적용되는 색인을 식별합니다. 이름은 현재 서버에 있는 색인을 식별해야 합니다.

PACKAGE

주석이 패키지에 대해 추가되거나 대체됨을 지정합니다.

package-name

주석이 적용되는 패키지를 식별합니다. 이름은 현재 서버에 있는 패키지를 식별해야 합니다.

PARAMETER

주석이 매개변수에 대해 추가되거나 대체됨을 지정합니다.

routine-name.parameter-name

주석이 적용되는 매개변수를 식별합니다. 매개변수는 프로시듀어나 함수에 대한

매개변수입니다. *routine-name*은 현재 서버에 있는 프로시듀어나 함수를 식별하고 *parameter-name*은 프로시듀어나 함수의 매개변수를 식별해야 합니다.

specific-name.parameter-name

주석이 적용되는 매개변수를 식별합니다. 매개변수는 프로시듀어나 함수에 대한 매개변수입니다. *specific-name*은 현재 서버에 있는 프로시듀어나 함수를 식별하고 *parameter-name*은 프로시듀어나 함수의 매개변수를 식별해야 합니다.

PROCEDURE

주석이 프로시듀어에 대해 추가되거나 대체됨을 지정합니다. 주석이 적용되는 프로시듀어를 식별합니다. 이름, 프로시듀어 표시 또는 특정 이름으로 특정 프로시듀어를 식별할 수 있습니다. 프로시듀어 해석 규칙(그리고 SQL 경로)은 사용되지 않습니다.

PROCEDURE *procedure-name*

*procedure-name*은 현재 서버에 있는 한 프로시듀어를 정확히 식별해야 합니다. 프로시듀어는 프로시듀어에 대해 정의된 매개변수를 갖습니다. 지정된 또는 내재된 스키마에 지정된 이름의 프로시듀어가 둘 이상 있는 경우 오류가 리턴됩니다.

PROCEDURE *procedure-name(parameter-type, ...)*

The *procedure-name(parameter-type, ...)*은 현재 서버에 있는 지정된 프로시듀어 표시로 프로시듀어를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE PROCEDURE문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수, 자료 유형의 논리 연결이 제거될 특정 프로시듀어 인스턴스를 식별하는 데 사용됩니다. *procedure-name()*이 지정되면 식별된 프로시듀어는 0개의 매개변수를 가져야 합니다.

procedure-name

프로시듀어 이름을 식별합니다.

(parameter-type, ...)

프로시듀어 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.
- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE PROCEDURE문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.

- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

  ```
  CHAR      CHAR(1)
  GRAPHIC   GRAPHIC(1)
  DECIMAL   DECIMAL(5,0)
  FLOAT     DOUBLE (길이 8)
  ```

  내재된 길이는 CREATE PROCEDURE문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE PROCEDURE문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC PROCEDURE *specific-name*

*specific-name*은 현재 서버에 있는 특정 프로시듀어를 식별해야 합니다.

TABLE *table-name* 또는 *view-name*

주석이 적용되는 표나 뷰를 식별합니다. 이름은 현재 서버에 있는 표나 뷰를 식별하지만 글로벌 임시 표를 식별해서는 안됩니다.

TRIGGER *trigger-name*

주석이 적용되는 트리거를 식별합니다. 이름은 현재 서버에 있는 트리거를 식별해야 합니다.

IS

추가되는 주석을 소개합니다.

string-constant

2000자 이하의 문자 스트링 상수가 될 수 있습니다. 상수에는 SBCS 또는 DBCS 문자가 포함될 수 있습니다.

multiple-columns

표나 뷰에 있는 하나 이상의 열에 주석을 작성하려면 표나 뷰 이름을 지정한 다음 괄호에 형식 리스트를 지정합니다.

```
column-name IS string-constant,
column-name IS string-constant, ...
```

열 이름은 규정되어서는 안되고, 각 이름은 지정된 표나 뷰의 열을 식별해야 하며, 그 표나 뷰는 현재 서버에 있어야 합니다.

multiple-parameters

프로시듀어나 함수에 있는 하나 이상의 매개변수에 주석을 작성하려면 프로시듀어명, 함수명 또는 특정 이름을 지정한 다음 괄호에 형식 리스트를 지정합니다.

```
parameter-name IS string-constant,
parameter-name IS string-constant, ...
```

매개변수 이름은 규정되어서는 안되고, 각 이름은 지정된 프로시듀어나 함수의 매개변수를 식별해야 하며, 프로시듀어나 함수는 현재 서버에 있어야 합니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 PROGRAM은 PACKAGE와 동의어로 사용될 수 있습니다.
- 키워드 DATA는 DISTINCT의 동의어로 사용될 수 있습니다.

예

예 1

EMPLOYEE 표에 대한 주석을 삽입합니다.

```
COMMENT ON TABLE EMPLOYEE
  IS 'Reflects first quarter 1981 reorganization'
```

예 2

EMP_VIEW1 뷰에 대한 주석을 삽입합니다.

```
COMMENT ON TABLE EMP_VIEW1
  IS 'View of the EMPLOYEE table without salary information'
```

예 3

EMPLOYEE 표의 EMPNO 열에 대한 주석을 삽입합니다.

```
COMMENT ON COLUMN EMPLOYEE.EMPN
  IS 'Highest grade level passed in school'
```

예 4

DEPARTMENT 표의 두 열에 주석을 입력합니다.

```
COMMENT ON DEPARTMENT
  (MGRNO IS 'EMPLOYEE NUMBER OF DEPARTMENT MANAGER',
   ADMRDEPT IS 'DEPARTMENT NUMBER OF ADMINISTERING DEPARTMENT')
```

예 5

CORPDATA.PAYROLL 패키지에 대한 주석을 삽입합니다.

```
COMMENT ON PACKAGE CORPDATA.PAYROLL
  IS 'This package is used for distributed payroll processing.'
```

COMMIT

COMMIT문은 작업 단위를 종료하고 그 작업 단위로 수행한 데이터베이스 변경을 확약합니다.

호출

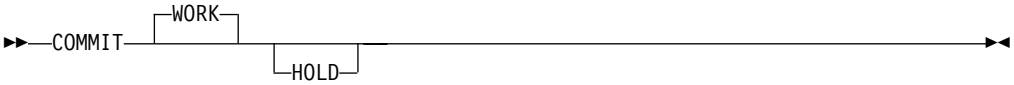
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다. Java에 지정되어서는 안됩니다.

트리거 프로그램과 트리거하는 프로그램이 같은 확약(commit) 정의에서 실행되면 COMMIT를 트리거에서 사용할 수 없습니다. 프로시듀어가 리모트 서버에서 호출되면 COMMIT를 프로시듀어에서 사용할 수 없습니다.

권한부여

필요한 사항이 없습니다.

구문



설명

COMMIT문은 COMMIT문이 실행된 작업 단위를 종료하고 새로운 작업 단위를 시작합니다. 이 명령문은 작업 단위 중의 SQL 스키마문(DROP SCHEMA 제외) 및 SQL 자료 변경문의 모든 변경사항을 확약합니다. SQL 스키마문 및 SQL 자료 변경문에 대해서는 361 페이지의 표 32 및 362 페이지의 표 33을 참조하십시오.

릴리스 지연 중 상태의 연결이 종료됩니다.

WORK

COMMIT WORK는 COMMIT와 같은 효과를 나타냅니다.

HOLD

자원의 보류를 지정합니다. 지정된 경우 현재 열린 커서가 닫히지 않고 작업 단위 동안 예약된 모든 자원이 보류됩니다. 작업 단위가 해제되는 동안 내재적으로 예약된 특정 행과 오브젝트를 잠급니다.

HOLD가 생략된 경우

- 커서가 WITH HOLD절로 선언되지 않았으면, 이 작업 단위 확약 정의에서 열린 커서가 닫힙니다.
- 작업 단위의 확약 정의에서 LOCK TABLE문에 의해 예약된 표 잠금이 해제됩니다.

닫히지 않은 커서에 필요한 오브젝트 레벨 잠금을 제외하고, 내재적으로 예약된 모든 잠금이 해제됩니다.

보유되지 않은 모든 로케이터는 릴리스됩니다. 보류된 로케이터에 대한 자세한 내용은 704 페이지의 『HOLD LOCATOR』를 참조하십시오.

주

내재된 COMMIT는 일정한 환경에서 수행될 수 있습니다. 그러나 어플리케이션이 종료되기 전에 명시적 COMMIT나 ROLLBACK이 발행되도록 권장됩니다.

- 디폴트 활성 그룹의 경우
  - 디폴트 활성 그룹에서 실행되는 어플리케이션이 종료될 때 내재적 COMMIT는 수행되지 않습니다. 대화식 SQL, Query Manager, 비ILE 프로그램은 디폴트 활성 그룹에서 실행되는 프로그램의 예입니다.
  - 작업을 확약하려면 COMMIT를 발행해야 합니다.
- 확약 정의의 범위가 활성 그룹에 해당될 때 디폴트가 아닌 활성 그룹의 경우
  - 활성 그룹이 정상적으로 종료되면 확약 정의는 내재적으로 확약됩니다.
  - 활성 그룹이 비정상적으로 종료되면 확약 정의는 내재적으로 롤백됩니다.
- 활성 그룹의 유형에 상관없이 확약 정의의 범위가 작업이면 내재적 확약은 수행되지 않습니다.

SELECT나 FETCH문[46]에서 검색된 행과 INSERT, DELETE 및 UPDATE문[47]의 일부로 삽입, 삭제 또는 갱신된 행을 포함하여 작업 단위는 4백만 행까지의 처리를 포함할 수 있습니다.

확약과 롤백 조작은 DROP SCHEMA문에 영향을 미치지 않으므로 이 명령문을 COMMIT(*CHG), COMMIT(*CS), COMMIT(*ALL) 또는 COMMIT(*RR)를 지정하는 어플리케이션 프로그램에서 사용할 수 없습니다.

SQL이 사용하는 확약 정의는 다음과 같이 판별됩니다.

- SQL을 호출하는 프로그램의 활성 그룹이 이미 활성 그룹 레벨 확약 정의를 사용 중이면 SQL은 그 확약 정의를 사용합니다.
- SQL을 호출하는 프로그램의 활성 그룹이 이미 작업 레벨 확약 정의를 사용 중이면 SQL은 그 작업 레벨 확약 정의를 사용합니다.

---

46. 이 한계에는 다음 사항이 포함됩니다.
    - 고급 언어 파일 처리를 통해 확약 제어에서 열린 파일을 통해 액세스하거나 변경한 행
    - 트리거나 CASCADE, SET NULL 또는 SET DEFAULT 참조 무결성 삭제 규칙의 결과로 삭제, 갱신 또는 삽입된 행

47. COMMIT(*CHG) 또는 COMMIT(*CS)를 지정하지 않으면 이러한 행은 총계에 포함되지 않습니다.

- SQL을 호출하는 프로그램의 활성 그룹이 현재 확약 정의를 사용하고 있지 않지만 작업 확약 정의가 시작되면 SQL은 그 작업 확약 정의를 사용합니다.
- SQL을 호출하는 프로그램의 활성 그룹이 현재 확약 정의를 사용하지 않고 작업 확약 정의가 시작되지 않으면 SQL은 내재적으로 확약 정의를 시작합니다. SQL은 다음과 함께 확약 제어 시작(STRCMTCTL) 명령을 사용합니다.
  - CMTSCOPE(*ACTGRP) 매개변수
  - CRTSQLxxx, STRSQL 또는 RUNSQLSTM 명령에 지정된 COMMIT 옵션을 기초로 한 LCKLVL 매개변수. REXX에서 LCKLVL 매개변수는 SET OPTION 문의 확약 옵션을 기초로 합니다.

예

COMMIT(*CHG)로 컴파일된 C 프로그램에서는 수수료(COMM)의 일정 금액을 EMPLOYEE 표의 한 사원(EMPNO)에서 다른 사원으로 전송합니다. 한 행에서 그 금액을 빼고 다른 행에 그 금액을 추가합니다. COMMIT WORK문을 사용하여 두 조작이 성공적으로 완료될 때까지는 데이터베이스를 영구적으로 변경하지 않는다는 사실을 확인합니다.

```
void main ()
  {

   EXEC SQL  BEGIN DECLARE SECTION;
     decimal(5,2) AMOUNT;
     char FROM_EMPNO[7];
     char TO_EMPNO[7];
   EXEC SQL  END DECLARE SECTION;
   EXEC SQL  INCLUDE SQLCA;
   EXEC SQL  WHENEVER SQLERROR GOTO SQLERR;
     ...
   EXEC SQL  UPDATE EMPLOYEE
               SET COMM = COMM - :AMOUNT
               WHERE EMPNO = :FROM_EMPNO;
   EXEC SQL  UPDATE EMPLOYEE
               SET COMM = COMM + :AMOUNT
               WHERE EMPNO = :TO_EMPNO;
 FINISHED:
   EXEC SQL  COMMIT WORK;
    return;

 SQLERR:
      ...
     EXEC SQL  WHENEVER SQLERROR CONTINUE;  /* continue if error on rollback */
   EXEC SQL  ROLLBACK WORK;
    return;
 }
```

CONNECT(유형 1)

CONNECT(TYPE 1)문은 리모트 작업 단위 규칙을 사용하여 어플리케이션 프로세스 내의 활성 그룹을 식별된 어플리케이션 서버에 연결합니다. 이 서버는 활성 그룹의 현재 서버입니다. RDBCNNMTH(*RUW)가 CRTSQLxxx 명령에 지정되면 이 유형의 CONNECT문이 사용됩니다. 두 가지 유형의 명령문의 차이에 대해서는 925 페이지의 『CONNECT(유형 1) 및 CONNECT(유형 2) 차이점』에 설명되어 있습니다. 연결 상태에 대한 자세한 내용은 31 페이지의 『어플리케이션 지시 분산 작업 단위』를 참조하십시오.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. Java나 REXX에 지정되어서는 안됩니다.

프로시듀어가 리모트 서버에서 호출되면 CONNECT를 트리거, 함수 또는 프로시듀어에서 사용할 수 없습니다.

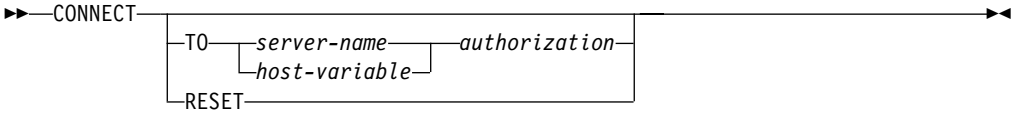
권한부여

명령문의 권한부여 ID가 보유하는 권한에는 통신 레벨 보안이 포함되어 있어야 합니다 (Distributed Database Programming 책의 보안에 관한 섹션을 참조하십시오).

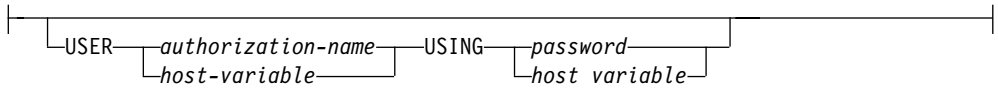
어플리케이션 서버가 iSeries용 DB2 UDB이면 명령문을 발행하는 개인의 사용자 프로파일은 서버 시스템의 유효한 사용자 프로파일이어야 합니다. 그렇지 않으면 다음 사항이 적용됩니다.

- 사용자가 지정됩니다. 이 경우 USER절은 서버 시스템에서 유효한 사용자 프로파일을 지정해야 합니다.
- TCP/IP는 서버에 대한 서버 권한부여 항목과 함께 사용됩니다. 이 경우 서버 권한부여 항목은 서버 시스템에서 유효한 사용자 프로파일을 지정해야 합니다.

구문



권한부여:



설명

TO *server-name* 또는 *host-variable*

지정한 서버명 또는 호스트 변수에 있는 서버명으로 서버를 식별합니다. 호스트 변수가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 서버명은 호스트 변수 내에서 왼쪽부터 배열되어야 하고 일반 ID를 형성하는 규칙을 따라야 합니다.
- 서버명의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

server-name이 로컬 관계형 데이터베이스(RDB)이고 authorization-name이 지정된 경우 server-name은 작업의 authorization-name이어야 합니다. 지정된 authorization-name이 작업의 authorization-name과 다르면, 오류가 발생하고 어플리케이션은 연결되지 않은 상태로 있습니다.

CONNECT문이 실행될 때 호스트 변수에 들어 있는 서버명이나 지정된 서버명은 로컬 디렉토리에 설명된 서버를 식별해야 하고 활성 그룹은 연결할 수 있는 상태에 있어야 합니다.

RESET

CONNECT RESET은 x가 로컬 서버명인 CONNECT TO x와 동등합니다.

CONNECT 피연산자가 없음

이 형식의 CONNECT문은 현재 서버에 대한 정보를 리턴하고 연결 상태, 열린 커서, 준비된 명령문 또는 잠금에 영향을 미치지 않습니다. 정보는 위에 설명된 대로 SQLCA에 리턴됩니다.

USER *authorization-name* 또는 *host-variable*

리모트 작업을 시작하는 데 사용되는 권한부여 이름이 들어 있는 지정된 *authorization-name*이나 *host-variable*로 authorization-name을 식별합니다.

*host-variable*이 지정된 경우

- 문자 스트링 변수여야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 권한부여 이름은 호스트 변수에서 왼쪽부터 배열되어야 하고 권한부여 이름을 형성하는 규칙을 따라야 합니다.
- 권한부여 이름의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

USING *password* 또는 *host-variable*

리모트 작업을 시작하는 데 사용되는 권한부여 이름의 암호가 들어 있는 지정된 *password*나 *host-variable*로 암호를 식별합니다.

암호가 리터럴로 지정되면 문자 스트링이어야 합니다. 최대 길이는 128자입니다. 왼쪽부터 배열되어야 합니다.

*host-variable*가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 암호는 호스트 변수 내에서 왼쪽부터 배열되어야 합니다.
- 암호의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

주

연결 성공: CONNECT문이 성공할 경우:

- 열린 커서가 모두 닫히고, 준비된 모든 명령문이 삭제되며, 현재 연결의 모든 잠금이 해제됩니다.
- 활성 그룹이 모든 현재 및 잠재 연결에서 모두 단절되며, 있는 경우 식별된 서버에 연결됩니다.
- 서버의 이름은 CURRENT SERVER 특수 레지스터에 위치합니다.
- 서버에 대한 정보는 SQLCA의 SQLERRP 및 SQLERRD(4) 필드에 있습니다. 서버가 IBM 관계형 데이터베이스(RDB) 제품이면 SQLERRP 필드에 있는 정보는 *pppvvrrm* 형식입니다.
  - *ppp*는 다음과 같은 제품을 식별합니다.
    ARI for DB2 for VM과 VSE
    DSN for OS/390 및 z/OS용 DB2 UDB
    QSQ for iSeries용 DB2 UDB
    그 외의 모든 DB2 UDB 제품용 SQL
  - *vv*는 '04'와 같은 두 자리의 버전 ID입니다.
  - *rr*은 '01'과 같은 두 자리의 릴리스 ID입니다.
  - *m*은 '0'과 같은 한 자리의 수정 레벨입니다.

  예를 들어 서버가 OS/390 및 z/OS용 DB2 UDB 버전 7이면 SQLERRP의 값은 'DSN07010'입니다.

  SQLCA의 SQLERRD(4) 필드에는 서버가 확약가능한 갱신의 수행을 허용하는지를 표시하는 값이 들어 있습니다. CONNECT(유형 1)문의 경우 SQLERRD(4)는 항상 값 1을 포함합니다. 값 1은 확약가능한 갱신이 수행될 수 있음을 표시합니다.

연결은

- 보호되지 않은 대화를 사용하거나[48] 또는
- *RUW 연결 메소드를 사용하는 어플리케이션 리퀘스터 드라이버 프로그램과의 연결이거나 또는
- *RUW 연결 메소드를 사용하는 로컬 연결입니다.

- 연결에 대한 추가 정보는 SQLCA 필드의 SQLERRMC에 위치합니다. 871 페이지의 부록 B『SQL 통신 영역』을 참조하십시오.

연결 실패: CONNECT문이 성공하지 못하면 SQLCA의 SQLERRP 필드는 오류를 감지한 어플리케이션 리퀘스터에서 모듈 이름으로 설정됩니다. 모듈 이름의 처음 세 자는 제품을 식별합니다. 예를 들어, 어플리케이션 리퀘스터가 NT용 DB2 UDB UWO이면 처음 세 자는 'SQL'입니다.

활성 그룹이 연결가능한 상태가 아니기 때문에 CONNECT문이 성공하지 못하면 활성 그룹의 연결 상태는 변경되지 않습니다. CONNECT문이 다른 이유로 성공하지 못한 경우

- 활성 그룹은 연결가능하지만, 연결되지 않는 상태입니다.
- 열린 커서가 모두 닫히고, 준비된 모든 명령문이 삭제되며, 모든 현재 및 잠재 연결의 모든 잠금이 해제됩니다.

연결가능하지만 연결되지 않는 상태의 어플리케이션만 CONNECT 또는 SET CONNECTION문을 실행할 수 있습니다.

내재적 연결:

- 디폴트 활성 그룹에서 실행될 때 다음 경우에 SQL 프로그램은 리모트 관계형 데이터베이스(RDB)에 내재적으로 연결됩니다.
  - 활성 그룹이 연결가능한 상태일 때
  - 프로그램 스택에 있는 첫 번째 SQL 프로그램의 첫 번째 SQL문이 실행될 때
- 디폴트가 아닌 활성 그룹에서 실행되면 SQL 프로그램은 프로그램 활성 그룹에 대한 첫 번째 SQL 프로그램의 첫 번째 SQL문이 실행될 때 리모트 관계형 데이터베이스(RDB)에 내재적으로 연결됩니다.

주: 활성 그룹에 의해 실행되는 첫 번째 SQL문을 CONNECT문으로 하는 것이 좋습니다.

48. 네트워크 연결과 SQL 연결 사이의 혼동 가능성을 줄이기 위해 이 책에서는 '대화'라는 용어를, 정식으로는 APPC 연결에만 적용됨에도 불구하고, APPC뿐 아니라 TCP/IP의 네트워크 연결에도 적용하여 사용합니다.

RDB에 연결하는 데 APPC가 사용되면 내재적 연결은 어플리케이션 리퀘스터 작업의 *authorization-name*을 항상 전송하지만 암호는 전송하지 않습니다. 서버 작업의 *authorization-name*이 다르거나 암호가 전송되어야 하는 경우 명시적 연결 명령문이 사용되어야 합니다.

TCP/IP가 RDB에 연결하기 위해 사용되면 내재적 연결은 위의 제한사항에 의해 바인드되지 않습니다. ADDSVRAUTE 및 기타 -SVRAUTE 명령을 사용하여 내재적(또는 명시적) CONNECT가 수행된 주어진 사용자에 대해, 주어진 RDB에 연결하기 위해 리모트 authorization-name 및 암호가 사용되도록 지정할 수 있습니다.

암호를 ADDSVRAUTE나 CHGSVRAUTE 명령으로 저장하려면 QRETSVRSEC 시스템 값은 '0'의 디폴트 값보다는 '1'로 설정되어야 합니다. DRDA 연결에 이 명령을 사용할 때 SERVER 매개변수에 입력된 RDB 값은 UPPER CASE이어야 합니다. 자세한 내용은 유형 2 CONNECT의 예 2를 참조하십시오.

내재적 연결에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오. 일단 사용자 프로파일에 대한 관계형 데이터베이스(RDB) 연결이 설정되면 다음에 같은 사용자 프로파일을 사용하여 같은 관계형 데이터베이스(RDB)에 연결할 때 암호는(암호가 지정된 경우) 다시 확인되지 않습니다. 암호의 재확인은 대화가 아직 활동중인지 여부에 따라 달라집니다. 자세한 내용은 Distributed Database Programming 책을 참조하십시오.

연결 상태: 연결 상태에 대한 설명은 30 페이지의 『리모트 작업 단위 연결 관리』를 참조하십시오. CONNECT가 연결가능 상태에서 활성 그룹을 제거하지 않기 때문에 연속 CONNECT 명령문은 성공적으로 실행될 수 있습니다.

어플리케이션 그룹의 현재 또는 잠재 연결에 대한 CONNECT는 다음과 같이 실행됩니다.

- CONNECT(유형 1)문을 사용하여 server-name에 의해 식별된 연결을 설정하였으면, 아무런 조치도 취해지지 않습니다. 커서는 닫히지 않고, 준비된 명령문은 삭제되지 않으며, 잠금은 해제되지 않습니다.
- CONNECT(유형 2)문을 사용하여 server-name에 의해 식별된 연결을 설정하였으면 CONNECT문은 다른 CONNECT문과 같이 실행됩니다.

CONNECT가 CONNECT, COMMIT, DISCONNECT, SET CONNECTION, RELEASE 또는 ROLLBACK 이외의 다른 SQL문 다음에 오면 성공적으로 실행될 수 없습니다. 오류를 피하려면 CONNECT문을 실행하기 전에 확약이나 롤백 조작을 실행합니다.

보호된 대화를 사용하여 현재 또는 잠재 연결을 설정하였으면, CONNECT(유형 1)문은 실패합니다. 또는 CONNECT(유형 2)문이 사용되어야 하거나 연결을 해제하고 성공적으로 확약하여 보호된 대화를 사용하는 연결을 종료해야 합니다.

리모트 관계형 데이터베이스 및 로컬 디렉토리에 연결에 대한 자세한 내용은 SQL 프로그래밍 개념 책과 Distributed Database Programming 책을 참조하십시오.

예

DRDA와 함께 TCP/IP 프로토콜을 사용하여 연결하는 데 사용할 수 있는 부가적 융통성의 예는 CONNECT 유형 2 설명의 예 2를 참조하십시오. 이 예는 CONNECT 유형 1에도 적용됩니다.

예 1

C 프로그램에서 사용자 JOE는 서버 TOROLAB3에 연결합니다. JOE의 암호는 XYZ1입니다. 성공적으로 연결한 후, 서버의 제품 ID 3자를 호스트 변수 *product*로 복사합니다.

```
void main ()
  {
   char product[4] = "   ";
   EXEC SQL  BEGIN DECLARE SECTION;
   char username[11];
   char userpass[129];
   EXEC SQL  END DECLARE SECTION;
   EXEC SQL INCLUDE SQLCA ;

   strcpy(username,"JOE");
   strcpy(userpass,"XYZ1";
   EXEC SQL CONNECT TO  TOROLAB3
              USER :username USING :userpass;
   if (strncmp(SQLSTATE, "00000", 5) )
     { strncpy(product,sqlca.sqlerrp,3); }
   ...
    return;
  }
```

CONNECT(유형 2)

CONNECT(유형 2)문은 어플리케이션 지향 분산 작업 단위 규칙을 사용하여 어플리케이션 프로세스 내의 활성 그룹을 식별된 서버에 연결합니다. 이 서버는 활성 그룹의 현재 서버입니다. RDBCNNMTH(*DUW)가 CRTSQLxxx 명령에 지정되면 이 유형의 CONNECT문이 사용됩니다. 두 가지 유형의 명령문의 차이에 대해서는 925 페이지의 『CONNECT(유형 1) 및 CONNECT(유형 2) 차이점』에 설명되어 있습니다. 연결 상태에 대한 자세한 내용은 31 페이지의 『어플리케이션 지시 분산 작업 단위』를 참조하십시오.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. Java나 REXX에 지정되어서는 안됩니다.

프로시듀어가 리모트 서버에서 호출되면 CONNECT을 프로시듀어에서 사용할 수 없습니다.

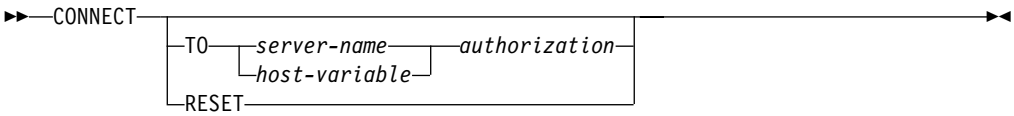
권한부여

명령문의 권한부여 ID가 보유하는 권한에는 통신 레벨 보안이 포함되어 있어야 합니다 (Distributed Database Programming 책의 보안에 관한 섹션을 참조하십시오).

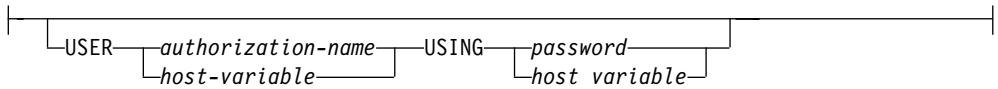
서버가 iSeries용 DB2 UDB이면 명령문을 발행하는 개인의 사용자 프로파일은 서버 시스템의 유효한 사용자 프로파일이어야 합니다. 그렇지 않으면 다음 사항이 적용됩니다.

- USER가 지정됩니다. USER가 지정되면 USER절은 서버 시스템에서 유효한 사용자 프로파일을 지정해야 합니다.
- TCP/IP는 서버에 대한 서버 권한부여 항목과 함께 사용됩니다. 이 경우 서버 권한부여 항목은 서버 시스템에서 유효한 사용자 프로파일을 지정해야 합니다.

구문



권한부여:



설명

TO *server-name* 또는 *host-variable*

지정한 서버명 또는 호스트 변수에 있는 서버명으로 서버를 식별합니다. 호스트 변수가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 서버명은 호스트 변수 내에서 왼쪽부터 배열되어야 하고 일반 ID를 형성하는 규칙을 따라야 합니다.
- 서버명의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

CONNECT문이 실행될 때 호스트 변수에 들어 있는 서버명이나 지정된 서버명은 로컬 디렉토리에 설명된 서버를 식별해야 합니다.

S는 호스트 변수에 포함된 서버명이나 지정된 서버명을 나타냅니다. S가 어플리케이션 프로세스의 기존 연결을 식별하면 안됩니다.

RESET

CONNECT RESET은 *x*가 로컬 서버 이름인 CONNECT TO *x*와 동등합니다.

CONNECT 피연산자가 없음

이 형식의 CONNECT문은 현재 서버에 대한 정보를 리턴하고 연결 상태, 열린 커서, 준비된 명령문 또는 잠금에 영향을 미치지 않습니다. 정보는 위에 설명된 SQLCA의 필드에 리턴됩니다.

또한 SQLCA의 SQLERRD(3) 필드는 이 작업 단위에 대한 연결 상태를 표시합니다. 다음 값 중 하나를 갖게 됩니다.

- 1 - 확약가능한 갱신이 이 작업 단위에 대한 연결에서 수행될 수 있습니다.
- 2 - 확약가능한 갱신이 이 작업 단위에 대한 연결에서 수행될 수 없습니다.

USER *authorization-name* 또는 *host-variable*

리모트 작업을 시작하는 데 사용되는 권한부여 이름이 들어 있는 지정된 *authorization-name*이나 *host-variable*에 의해 authorization-name을 식별합니다.

*host-variable*이 지정된 경우

- 문자 스트링 변수여야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다. 권한부여 이름은 호스트 변수에서 왼쪽부터 배열되어야 하고 권한부여 이름을 형성하는 규칙을 따라야 합니다.
- 권한부여 이름의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

USING *password* 또는 *host-variable*

리모트 작업을 시작하는 데 사용되는 권한부여 이름의 암호가 들어 있는 지정된 *password*나 *host-variable*로 암호를 식별합니다.

암호가 리터럴로 지정되면 문자 스트링이어야 합니다. 최대 길이는 128자입니다. 왼쪽부터 배열되어야 합니다.

*host-variable*가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 암호는 호스트 변수 내에서 왼쪽부터 배열되어야 합니다.
- 암호의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

주

연결 성공: CONNECT문이 성공할 경우:

- 서버 S로의 연결이 작성되며 현재 및 보류 상태가 됩니다. 이전 연결(있는 경우)은 잠재 상태가 됩니다.
- S는 CURRENT SERVER 특수 레지스터에 위치합니다.
- 서버 S에 대한 정보는 SQLCA의 SQLERRP 및 SQLERRD(4) 필드에 위치합니다. 서버가 IBM 관계형 데이터베이스(RDB) 제품이면 SQLERRP 필드에 있는 정보는 *pppvvrrm* 형식입니다.
  - *ppp*는 다음과 같은 제품을 식별합니다.
    ARI for DB2 for VM과 VSE
    DSN for OS/390 및 z/OS용 DB2 UDB
    QSQ for iSeries용 DB2 UDB
    그 외의 모든 DB2 UDB 제품용 SQL
  - *vv*는 '04'와 같은 두 자리의 버전 ID입니다.
  - *rr*은 '01'과 같은 두 자리의 릴리스 ID입니다.
  - *m*은 '0'과 같은 한 자리의 수정 레벨입니다.

  예를 들어 서버가 OS/390 및 z/OS용 DB2 UDB 버전 7이면 SQLERRP의 값은 'DSN07010'입니다.

  SQLCA의 SQLERRD(4) 필드에는 서버 S가 확약가능한 갱신의 수행을 허용하는지를 표시하는 값이 들어 있습니다. 다음은 CONNECT에 있는 SQLCA의 SQLERRD(4) 필드에 대한 값과 의미 리스트입니다.
  - 1 - 확약가능한 갱신이 수행될 수 있습니다. 대화는 보호되지 않습니다.[48]
  - 2 - 확약가능한 갱신이 수행될 수 없습니다. 대화는 보호되지 않습니다.

  - 3 - 확약가능한 갱신이 수행될 수 있는지 알 수 없습니다. 대화는 보호됩니다.
  - 4 - 확약가능한 갱신이 수행될 수 있는지 알 수 없습니다. 대화는 보호되지 않습니다.
  - 5 - 확약가능한 갱신이 수행될 수 있는지 알 수 없습니다. 연결은 로컬 연결 또는 어플리케이션 리퀘스터 드라이버 프로그램 연결입니다.
- 연결에 대한 추가 정보는 SQLCA 필드의 SQLERRMC에 위치합니다. 871 페이지의 부록 B『SQL 통신 영역』을 참조하십시오.

연결 실패: CONNECT문이 성공적으로 수행되지 않으면 활성 그룹의 연결 상태 및 각 연결의 상태가 변하지 않습니다.

내재적 연결:내재적 연결은 항상 어플리케이션 리퀘스터 작업의 *authorization-name*을 전송하고 암호를 전송하지 않습니다. 서버 작업의 *authorization-name*이 다르거나 암호가 전송되어야 하는 경우 명시적 연결 명령문이 사용되어야 합니다.

TCP/IP가 RDB에 연결하기 위해 사용되면 내재적 연결은 위의 제한사항에 의해 바인드되지 않습니다. ADDSVRAUTE 및 기타 -SVRAUTE 명령을 사용하여 내재적(또는 명시적) CONNECT가 수행된 주어진 사용자에 대해, 주어진 RDB에 연결하기 위해 리모트 authorization-name 및 암호가 사용되도록 지정할 수 있습니다.

암호를 ADDSVRAUTE나 CHGSVRAUTE 명령으로 저장하려면 QRETSVRSEC 시스템 값은 '0'의 디폴트 값보다는 '1'로 설정되어야 합니다. DRDA 연결에 이 명령을 사용할 때 SERVER 매개변수에 입력된 RDB 값은 UPPER CASE이어야 합니다. 자세한 내용은 유형 2 CONNECT의 예 2를 참조하십시오.

내재적 연결에 대한 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오. 일단 사용자 프로파일에 대한 관계형 데이터베이스(RDB) 연결이 설정되면 다음에 같은 사용자 프로파일을 사용하여 같은 관계형 데이터베이스(RDB)에 연결할 때 암호는(암호가 지정된 경우) 다시 확인되지 않습니다. 암호의 재확인은 대화가 아직 활동중인지 여부에 따라 달라집니다. 자세한 내용은 Distributed Database Programming 책을 참조하십시오.

예

예 1

TOROLAB1과 TOROLAB2에서 SQL문을 실행합니다. 첫 번째 CONNECT문은 TOROLAB1 연결을 작성하고 두 번째 CONNECT문은 그 연결을 잠재 상태로 만듭니다.

```
EXEC SQL  CONNECT TO TOROLAB1;

(execute statements referencing objects at TOROLAB1)
```

```
EXEC SQL  CONNECT TO TOROLAB2;

(execute statements referencing objects at TOROLAB2)
```

예 2

사용자 JOE는 TOROLAB3에 연결하고 암호가 SHIBBOLETH인 사용자 ID ANONYMOUS로 SQL문을 실행하려고 합니다. TOROLAB3에 대한 RDB 디렉토리 항목은 연결 유형에 대한 *IP를 지정합니다.

어플리케이션을 실행하기 전에 설정이 일부 수행되어야 합니다.

이 명령이 이전에 실행되지 않은 경우 서버 보안 정보를 OS/400에 보관하기 위해 필요합니다.

```
CHGSYSVAL SYSVAL(QRETSVRSEC) VALUE('1')
```

이 명령은 필요한 서버 권한부여 항목을 추가합니다.

```
ADDSVRAUTE USRPRF(JOE) SERVER(TOROLAB3) USRID(ANONYMOUS) +
           PASSWORD(SHIBBOLETH)
```

JOE의 사용자 프로파일에서 실행되는 이 명령문이 연결을 설정합니다.

```
EXEC SQL CONNECT TO TOROLAB3;
(execute statements referencing objects at TOROLAB3)
```

CREATE ALIAS

CREATE ALIAS문은 현재 서버에서 데이터베이스 파일의 표, 뷰 또는 멤버에 별명을 작성합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

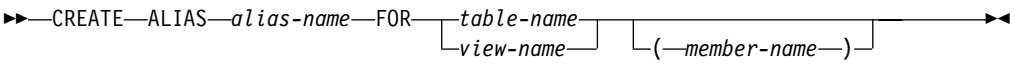
명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - DDM 파일 작성(CRTDDMF) 명령에 대한 *USE
  - 별명이 작성되는 라이브러리에 대한 *EXECUTE, *READ 및 *ADD
- 관리 권한

SQL 이름이 지정되고 별명이 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며, 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

구문



설명

alias-name

별명을 명명합니다. 내재적 또는 명시적 규정자를 가지고 있는 이름은 이미 현재 서버에 있는 색인, 표, 뷰, 별명 또는 파일과 같을 수 없습니다.

SQL 이름이 지정되면 별명은 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템명이 지정되면 별명은 규정자에 의해 지정된 스키마에 작성됩니다. 규정되지 않은 경우 별명은 별명이 작성된 표나 뷰와 동일한 스키마에 작성됩니다. 표가 규정되지 않고 별명이 작성될 때 존재하지 않으면 별명은 현재 라이브러리(*CURLIB)에 작성됩니다.

별명 이름이 유효한 시스템명이 아니면 iSeries용 DB2 UDB는 시스템명을 생성합니다. 이름 생성 규칙에 대한 내용은 572 페이지의 『표 이름 생성 규칙』을 참조하십시오.

FOR *table-name* 또는 *view-name*

별명이 작성되는 현재 서버의 표나 뷰를 정의합니다. 별명 이름은 지정될 수 없습니다(별명은 다른 별명을 참조할 수 없습니다).

*table-name*이나 *view-name*은 별명이 작성될 때 존재하는 표나 뷰를 식별할 필요가 없습니다. 표나 뷰가 별명이 작성될 때 없으면 경고가 리턴됩니다. 표나 뷰가 별명이 사용될 때 없으면 오류가 리턴됩니다.

SQL 이름이 지정되었고 *table-name*이나 *view-name*이 규정되지 않았으면, 규정자는 내재적 규정자입니다. 자세한 내용은 47 페이지의 『명명 규칙』을 참조하십시오.

시스템명이 지정되었고 *table-name*이나 *view-name*이 규정되지 않으며 별명이 작성될 때 없으면 *table-name*이나 *view-name*은 별명이 작성되는 라이브러리에 의해 규정됩니다.

member-name

데이터베이스 파일의 멤버를 식별합니다.

멤버가 지정되면 자료 조작(DML) SQL문의 별명만 사용할 수 있습니다. 멤버 이름이 지정되지 않으면 *FIRST가 사용됩니다.

주

데이터베이스 파일 대체(OVRDBF) CL 명령을 사용하여 데이터베이스 관리자가 데이터베이스 파일의 개별 멤버를 처리할 수 있습니다. 그러나 데이터베이스 파일의 멤버에 대한 별명 작성은 대체를 수행해야 할 필요성을 제거함으로서 더 쉽게 그리고 더 잘 수행됩니다.

시스템명이나 SQL명을 참조하기 위해 별명이 정의될 수 있습니다. 그러나, 작성 처리 중 시스템명이 생성되기 때문에, SQL명을 지정하는 것이 좋습니다.

별명 속성: 별명은 DDM 파일로부터 특별한 형태로 작성됩니다.

분배된 표에 작성된 별명은 현재 서버에서만 작성됩니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

별명 소유권: SQL명이 지정된 경우, 별명의 *소유자*는 별명이 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 별명의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 별명의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

별명 권한: SQL명이 사용되면 뷰는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 별명은 스키마의 작성 권한(CRTAUT) 매개변수에 의해 판별되듯이 *PUBLIC에 대한 권한으로 작성됩니다.

별명의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 별명에 대한 권한도 갖습니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 SYNONYM은 ALIAS의 동의어로 사용될 수 있습니다.

예

예 1

PROJECT 표에 대해 CURRENT_PROJECTS라는 별명을 작성합니다.

```
CREATE ALIAS CURRENT_PROJECTS
  FOR PROJECT
```

예 2

SALES 표의 JANUARY 멤버에 SALES_JANUARY라는 별명을 작성합니다. 판매 표에는 12개의 멤버가 있습니다(매달 1개).

```
CREATE ALIAS SALES_JANUARY
  FOR SALES(JANUARY)
```

CREATE DISTINCT TYPE

CREATE DISTINCT TYPE문은 고유한 유형을 정의합니다. 고유한 유형은 항상 내장 자료 유형 중 하나를 소스로 합니다. 명령문의 성공적인 실행은 고유한 유형과 그 소스 유형 사이를 캐스트하는 함수도 생성하고 고유한 유형과 함께 사용하기 위한 비교 연산자(자료 링크에 대해서는 제외)에 대한 지원도 생성합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 고유한 유형이 작성되는 라이브러리에 대한 시스템 권한 *EXECUTE, *READ 및 *ADD
- 관리 권한

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSTYPES 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

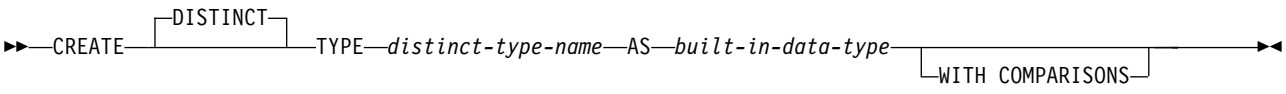
- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

SQL 이름이 지정되고, 고유한 유형이 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며, 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

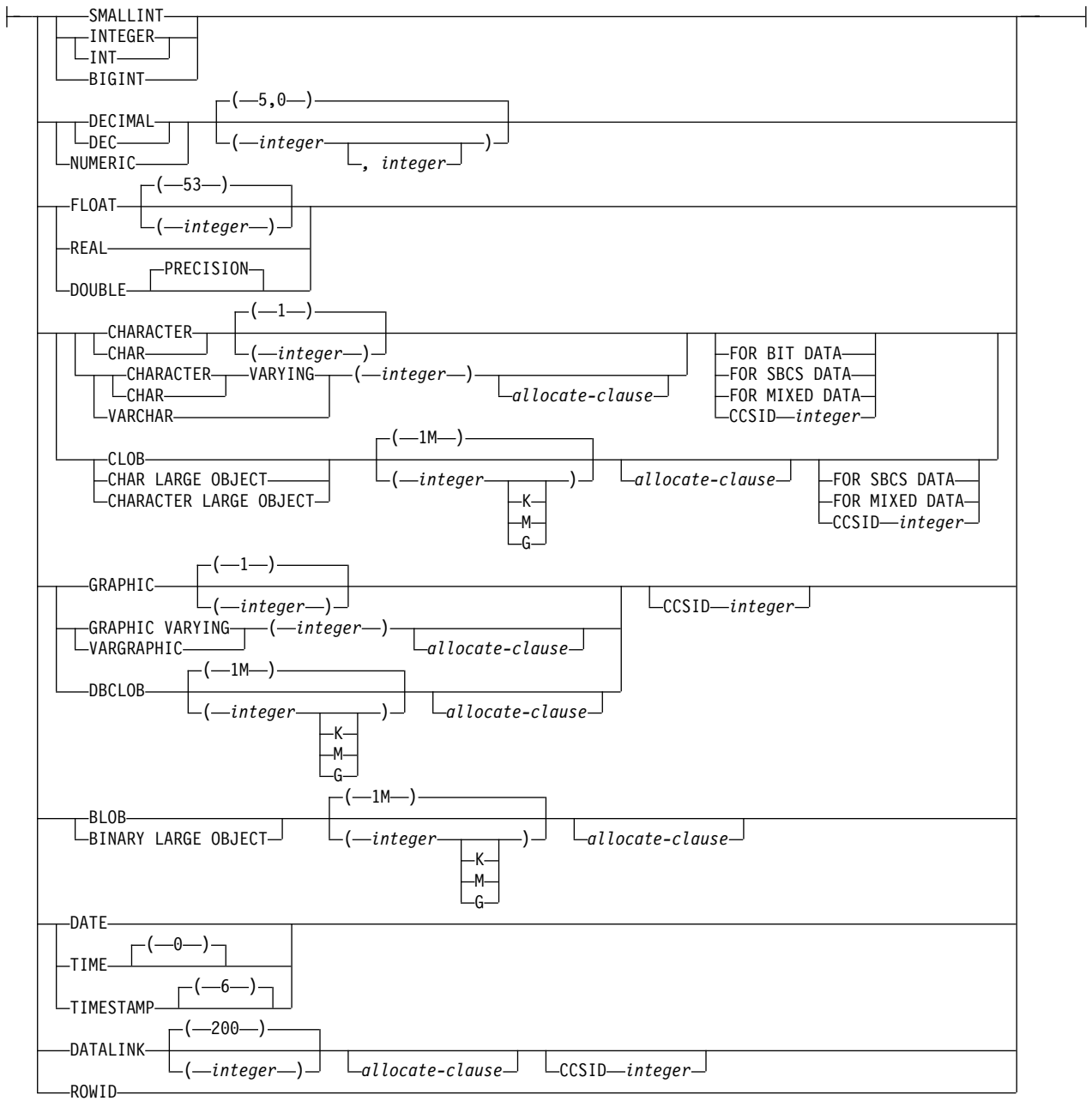
- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

CREATE DISTINCT TYPE

구문



built-in-data-type:



설명

distinct-type-name

고유한 유형을 명명합니다. 내재적 또는 명시적 규정자를 가지고 있는 이름은 이미 현재 서버에 있는 고유한 유형과 같을 수 없습니다.

SQL명이 지정되면 고유한 유형은 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 이름이 지정되었으면 고유한 유형은 규정자에 의해 지정된 스키마에 작성됩니다. 규정되지 않은 경우 고유한 유형은 현재 라이브러리(*CURLIB)에 작성됩니다.

고유한 유형 이름이 유효한 시스템명이 아니면 iSeries용 DB2 UDB는 시스템명을 생성합니다. 이름 생성 규칙에 대한 내용은 572 페이지의 『표 이름 생성 규칙』을 참조하십시오.

*distinct-type-name*은 내장된 자료 유형의 이름이어서는 안되고, 시스템에 예약된 키워드를 분리 식별자로 지정한다고 해도 다음의 키워드여서는 안됩니다.

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!<	!>
ALL	FALSE	ONLY	TABLE
AND	FOR	OR	THEN
ANY	FROM	OVERLAPS	TRIM
BETWEEN	IN	PARTITION	TRUE
BOOLEAN	IS	POSITION	TYPE
CASE	LIKE	RRN	UNIQUE
CAST	MATCH	SELECT	UNKNOWN
CHECK	NODENAME	SIMILAR	WHEN
DISTINCT	NODENUMBER	SOME	
EXCEPT	NOT	STRIP	
EXISTS	NULL	SUBSTRING	

| 규정된 *distinct-type-name*이 지정되면 스키마명은 QSYS, QSYS2,QTEMP 또는
| SYSIBM가 될 수 없습니다.

built-in-data-type

고유한 유형의 내부 표시시 기본으로 사용되는 자료 유형을 지정합니다. 내장된 자료 유형이어야 합니다. LONG VARCHAR 또는 LONG VARGRAPHIC를 제외하고 CREATE TABLE문에서 사용할 수 있는 내장된 자료 유형을 사용할 수 있습니다.

길이, 정밀도 또는 스케일이 명시적으로 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

CHAR	CHAR(1)
GRAPHIC	GRAPHIC(1)
DECIMAL	DECIMAL(5,0)
FLOAT	DOUBLE (길이 8)

고유한 유형이 스트링 자료 유형으로부터 온 것인 경우 고유한 유형이 작성될 때 고유 자료 유형과 CCSID과 연관됩니다. 자료 유형에 대한 자세한 정보는 541 페이지의 『CREATE TABLE』을 참조하십시오.

WITH COMPARISONS

고유한 유형의 두 가지 인스턴스를 비교하기 위해 시스템 생성 비교 함수를 작성하도록 지정합니다. WITH COMPARISONS는 디폴트입니다. 비교는 WITH COMPARISONS의 지정 여부에 상관없이 DATALINK를 제외한 모든 소스 유형에 대해 생성됩니다.[49] 다른 DB2 제품과 호환되기 위해 WITH COMPARISONS가 지정되어야 합니다.

비교 함수는 LIKE 술부를 지원하지 않습니다. 고유한 유형에서 LIKE 술부를 사용하려면 내장 유형으로 캐스트해야 합니다.

주

CREATE DISTINCT TYPE문이 성공적으로 실행되면 데이터베이스 관리자가 다음의 캐스트 함수를 생성합니다.

- 고유한 유형에서 소스 유형으로 변환하는 함수
- 소스 유형에서 고유한 유형으로 변환하는 함수
- 소스 유형이 SMALLINT일 경우 INTEGER에서 고유한 유형으로 변환하는 함수
- 소스 유형이 REAL일 경우 DOUBLE에서 고유한 유형으로 변환하는 함수
- 소스 유형이 CHAR일 경우 VARCHAR에서 고유한 유형으로 변환하는 함수
- 소스 유형이 GRAPHIC일 경우 VARGRAPHIC에서 고유한 유형으로 변환하는 함수

캐스트 함수는 다음 명령문이 실행된 것처럼 작성됩니다(서비스 프로그램이 작성되지 않아서 해당 함수에 특권을 부여하거나 취소할 수 없는 경우를 제외하고).

```
CREATE FUNCTION distinct-type-name (source-type-name)
  RETURNS distinct-type-name

CREATE FUNCTION source-type-name (distinct-type-name)
  RETURNS source-type-name
```

CREATE DISTINCT TYPE문의 소스 자료 유형에 대한 길이, 정밀도 또는 눈금을 지정해도, 고유한 유형을 소스 유형으로 변환하는 캐스트 함수의 이름은 간단히 소스 자

49. 이러한 비교 함수에 대해서는 서비스 프로그램이 작성되지 않습니다. 이 비교 함수는 SYSROUTINES 카탈로그 표에 등록되지 않습니다.

료 유형의 이름입니다. 캐스트 함수가 리턴하는 값의 자료 유형에는 소스 자료 유형에 대해 지정한 길이, 정밀도 또는 눈금 값이 포함됩니다(표 43을 참조하십시오).

소스 유형을 고유한 유형으로 변환하는 캐스트 함수의 이름은 고유한 유형의 이름입니다. 캐스트 함수의 입력 매개변수는 길이, 정밀도 및 눈금을 포함하여 소스 자료 유형과 같은 자료 유형을 갖습니다.

생성된 캐스트 함수는 고유한 유형의 스키마와 같은 스키마에 작성됩니다. 같은 이름과 같은 함수 서명을 갖는 함수가 현재 서버에 있으면 안됩니다.

예를 들면, T_SHOESIZE라는 고유한 유형이 다음 명령문으로 작성된다고 가정합니다.

```
CREATE DISTINCT TYPE CLAIRE.T_SHOESIZE AS VARCHAR(2) WITH COMPARISONS
```

명령문이 실행될 때 데이터베이스 관리자는 다음 캐스트 함수도 생성합니다. VARCHAR는 고유한 유형을 소스 유형으로 변환하고, T_SHOESIZE는 소스 유형을 고유한 유형으로 변환합니다.

```
FUNCTION CLAIRE.VARCHAR (CLAIRE.T_SHOESIZE) RETURNS VARCHAR(2)

FUNCTION CLAIRE.T_SHOESIZE (VARCHAR(2) RETURNS CLAIRE.T_SHOESIZE
```

함수 VARCHAR은 VARCHAR(2)의 자료 유형을 갖는 값을 리턴하고 함수 T_SHOESIZE는 VARCHAR(2)의 자료 유형을 갖는 입력 매개변수를 갖습니다.

생성된 캐스트 함수를 명시적으로 제거할 수 없습니다. 고유한 유형에 대해 생성된 캐스트 함수는 고유한 유형이 DROP문으로 제거될 때 내재적으로 제거됩니다.

고유한 유형에 대한 소스 자료 유형이 될 수 있는 내장된 자료 유형의 경우 다음 표는 생성된 캐스트 함수의 이름, 입력 매개변수의 자료 유형 및 함수가 리턴되는 값의 자료 유형을 제시합니다.

표 43. 고유한 유형에 대한 CAST 함수

소스 유형 이름	함수명	parameter-type	리턴 유형
SMALLINT	*distinct-type-name*	SMALLINT	*distinct-type-name*
	distinct-type-name	INTEGER	*distinct-type-name*
	SMALLINT	*distinct-type-name*	SMALLINT
INTEGER	*distinct-type-name*	INTEGER	*distinct-type-name*
	INTEGER	*distinct-type-name*	INTEGER
BIGINT	*distinct-type-name*	BIGINT	*distinct-type-name*
	BIGINT	*distinct-type-name*	BIGINT
DECIMAL	*distinct-type-name*	DECIMAL(p,s)	*distinct-type-name*
	DECIMAL	*distinct-type-name*	DECIMAL(p,s)
NUMERIC	*distinct-type-name*	NUMERIC(p,s)	*distinct-type-name*
	NUMERIC	*distinct-type-name*	NUMERIC(p,s)
REAL	*distinct-type-name*	REAL	*distinct-type-name*

표 43. 고유한 유형에 대한 CAST 함수 (계속)

소스 유형 이름	함수명	parameter-type	리턴 유형
	distinct-type-name	DOUBLE	*distinct-type-name*
	REAL	*distinct-type-name*	REAL
FLOAT(n), 여기서 n <= 24	*distinct-type-name*	REAL	*distinct-type-name*
	distinct-type-name	DOUBLE	*distinct-type-name*
	REAL	*distinct-type-name*	REAL
FLOAT(n), 여기서 n > 24	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
DOUBLE 또는 DOUBLE PRECISION	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
CHAR	*distinct-type-name*	CHAR(n)	*distinct-type-name*
	distinct-type-name	VARCHAR(n)	*distinct-type-name*
	CHAR	*distinct-type-name*	CHAR(n)
VARCHAR	*distinct-type-name*	VARCHAR(n)	*distinct-type-name*
	VARCHAR	*distinct-type-name*	VARCHAR(n)
CLOB	*distinct-type-name*	CLOB(n)	*distinct-type-name*
	CLOB	*distinct-type-name*	CLOB(n)
GRAPHIC	*distinct-type-name*	GRAPHIC(n)	*distinct-type-name*
	distinct-type-name	VARGRAPHIC (n)	*distinct-type-name*
	GRAPHIC	*distinct-type-name*	GRAPHIC(n)
VARGRAPHIC	*distinct-type-name*	VARGRAPHIC (n)	*distinct-type-name*
	VARGRAPHIC	*distinct-type-name*	VARGRAPHIC (n)
DBCLOB	*distinct-type-name*	DBCLOB(n)	*distinct-type-name*
	DBCLOB	*distinct-type-name*	DBCLOB(n)
BLOB	*distinct-type-name*	BLOB(n)	*distinct-type-name*
	BLOB	*distinct-type-name*	BLOB(n)
DATE	*distinct-type-name*	DATE	*distinct-type-name*
	DATE	*distinct-type-name*	DATE
TIME	*distinct-type-name*	TIME	*distinct-type-name*
	TIME	*distinct-type-name*	TIME
TIMESTAMP	*distinct-type-name*	TIMESTAMP	*distinct-type-name*
	TIMESTAMP	*distinct-type-name*	TIMESTAMP
DATALINK	*distinct-type-name*	DATALINK	*distinct-type-name*
	DATALINK	*distinct-type-name*	DATALINK
ROWID	*distinct-type-name*	ROWID	*distinct-type-name*
	ROWID	*distinct-type-name*	ROWID

표 43. 고유한 유형에 대한 CAST 함수 (계속)

소스 유형 이름	함수명	parameter-type	리턴 유형
주: * 변환은 UCS-2 그래픽의 경우에만 지원됩니다. DATALINK만 DATALINK 유형으로 캐스트될 수 있습니다.			

포팅할 수 있는 어플리케이션에 대한 고유한 유형을 작성할 때 NUMERIC 및 FLOAT는 권장되지 않습니다. 대신 DECIMAL 및 DOUBLE이 사용되어야 합니다.

고유한 유형 속성: 고유한 유형은 *SQLUDT 오브젝트로 작성됩니다.

고유 유형 소유권: SQL명이 지정된 경우, 고유한 유형의 *소유자*는 고유한 유형이 작성되는 *스키마*와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 고유한 유형의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 고유한 유형의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

고유 유형 권한: SQL명이 사용된 경우, 고유한 유형은 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 고유한 유형은 *PUBLIC에 대한 권한으로 작성됩니다.

고유한 유형의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 고유한 유형에 대한 권한도 갖습니다.

예

예 1

INTEGER 자료 유형을 소스로 하는 SHOESIZE라는 고유한 유형을 작성합니다.

```
CREATE DISTINCT TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

이 명령문이 성공적으로 실행되면 두 개의 캐스트 함수도 생성됩니다. 함수 INTEGER(SHOESIZE)는 자료 유형 INTEGER를 갖는 값을 리턴하고 함수 SHOESIZE(INTEGER)는 고유한 유형 SHOESIZE를 갖는 값을 리턴합니다.

예 2

DOUBLE 자료 유형을 소스로 하는 MILES라는 고유한 유형을 작성합니다.

```
CREATE DISTINCT TYPE MILES AS DOUBLE WITH COMPARISONS
```

이 명령문이 성공적으로 실행되면 두 개의 캐스트 함수도 생성됩니다. 함수 DOUBLE(MILES)은 자료 유형 DOUBLE을 갖는 값을 리턴하고 함수 MILES(DOUBLE)는 고유한 유형 MILES를 갖는 값을 리턴합니다.

CREATE FUNCTION

현재 서버에서 등록되는 사용자 정의 함수를 작성하기 위해 CREATE FUNCTION문을 사용할 수 있습니다.

다음 함수 유형을 정의할 수 있습니다.

- 외부 스칼라

  이 함수는 Java나 C와 같은 프로그래밍 언어로 작성되고 스칼라 값을 리턴합니다. 외부 프로그램은 함수의 여러 속성과 함께 현재 서버에 정의된 함수에서 참조합니다. 447 페이지의 『CREATE FUNCTION(외부 스칼라)』을 참조하십시오.

- 외부 표

  이 함수는 Java나 C와 같은 프로그래밍 언어로 작성되고 행 세트를 리턴합니다. 외부 프로그램은 함수의 여러 속성과 함께 현재 서버에 정의된 함수에서 참조합니다. 465 페이지의 『CREATE FUNCTION(외부 스칼라)』을 참조하십시오.

- 소스 함수

  현재 서버에 이미 존재하는 다른 함수(내장, 외부, 소스 또는 SQL)를 호출하여 이 함수를 구현합니다. 소스 함수는 스칼라 결과 또는 열 함수의 결과를 리턴할 수 있습니다. 480 페이지의 『CREATE FUNCTION(피소스(sourced))』을 참조하십시오. 함수는 기본 소스 함수의 속성을 계승합니다.

- SQL 스칼라

  이 함수는 완전 SQL로 작성되고 스칼라 값을 리턴합니다. 함수의 본문은 함수의 여러 속성과 함께 현재 서버에 정의됩니다. 488 페이지의 『CREATE FUNCTION(SQL 스칼라)』을 참조하십시오.

- SQL TABLE

  이 함수는 완전 SQL로 작성되고 행 세트를 리턴합니다. 함수의 본문은 함수의 여러 속성과 함께 현재 서버에 정의됩니다. 497 페이지의 『CREATE FUNCTION(SQL 표)』을 참조하십시오.

주

함수명 선택

규정된 함수 이름이 지정된 경우, 스키마명은 QSYS2, QSYS, QTEMP 또는 SYSIBM이 될 수 없습니다.

함수 이름은 시스템 사용을 위해 예약된 다음 이름이 될 수 없습니다.

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!>	!<
ALL	FALSE	ONLY	TABLE
AND	FOR	OR	THEN

ANY	FROM	OVERLAPS	TRIM
BETWEEN	IN	PARTITION	TRUE
BOOLEAN	IS	POSITION	TYPE
CASE	LIKE	RRN	UNIQUE
CAST	MATCH	SELECT	UNKNOWN
CHECK	NODENAME	SIMILAR	WHEN
DISTINCT	NODENUMBER	SOME	
EXCEPT	NOT	STRIP	
EXISTS	NULL	SUBSTRING	

입력 매개변수의 자료 유형 선택

함수에 대한 입력 매개변수의 자료 유형을 선택할 때 입력 매개변수의 값에 영향을 줄 수 있는 승격 규칙을 고려합니다(76 페이지의 『자료 유형의 승격』을 참조하십시오). 예를 들면, 함수의 입력 인수 중 하나인 상수가 함수가 예상하는 자료 유형과 다른 내장된 자료 유형을 가질 수 있으며, 더구나 그 예상 자료 유형으로 승격시킬수 없을 것입니다. 승격 규칙에 기초하여 매개변수에 대해 다음 자료 유형을 사용할 것을 권장합니다.

- SMALLINT 대신 INTEGER
- REAL 대신 DOUBLE
- CHAR 대신 VARCHAR
- GRAPHIC 대신 VARGRAPHIC

iSeries용 DB2 UDB가 아닌 플랫폼에서의 함수 이식을 위해 다른 플랫폼에서는 다르게 표시되는 다음 자료 유형은 사용하지 않습니다.

- FLOAT. 대신 DOUBLE이나 REAL을 사용합니다.
- NUMERIC. 대신 DECIMAL을 사용합니다.

매개변수에 AS LOCATOR 지정

값 대신 로케이터를 전달하면 함수 내외에서 더 적은 수의 바이트가 전달됩니다. 매개변수 값이 매우 큰 경우 이 기능은 유용할 수 있습니다. AS LOCATOR절은 실제 값 대신 매개변수 값으로의 로케이터가 전달되도록 지정합니다. LOB 자료 유형을 기초로 한 고유한 유형이나 LOB 자료 유형을 가진 매개변수에만 AS LOCATOR를 지정할 수 있습니다.

AS LOCATOR절은 자료 유형이 승격될 수 있는지 여부를 결정하는 데 영향을 주지 않으며, 함수 분석에 사용되는 함수 서명에도 영향을 주지 않습니다.

AS LOCATOR는 SQL 함수에 대해 지정될 수 없습니다.

스키마에서 함수 고유성 판별

현재 서버에서 각 함수의 표시는 고유해야 합니다. 함수의 표시는 입력 매개변수의 자료 유형과 번호를 결합한 규정된 함수명입니다. 이는 두 개의 다른 스키마가 각각, 모든 해당 자료 유형에 대해 같은 자료 유형을 갖는 동일한 이름의 함수를 포함할 수 있

다는 것을 의미합니다. 그러나 하나의 스키마에 모든 해당 자료 유형에 대해 같은 자료 유형을 갖는 동일한 이름의 두 함수가 있을 수는 없습니다.

해당 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자는 비교에 있는 길이, 정밀도, 눈금 또는 CCSID 속성은 고려하지 않습니다. 데이터베이스 관리자는 자료 유형(REAL과 FLOAT 및 DOUBLE과 FLOAT)의 동의어를 일치로 간주합니다. 따라서 DECIMAL(11,2)와 DECIMAL(4,3)이 동일하게 간주되는 것처럼 CHAR(8)과 CHAR(35)도 동일하게 간주됩니다.

동일한 스키마에서 네 개의 함수를 작성하기 위해 다음 명령문이 실행된다고 가정합니다. 두 번째와 네 번째 명령문은 첫 번째와 세 번째 명령문이 작성한 함수와 중복되는 함수를 작성하기 때문에 실패합니다.

```
CREATE FUNCTION PART (INT, CHAR(15) ...
CREATE FUNCTION PART (INTEGER, CHAR(40) ...

CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

함수의 고유명

동일한 이름과 스키마를 가진 여러 함수를 정의할 경우(서로 다른 매개변수 리스트를 가진), 고유명도 지정하는 것이 좋습니다. 고유명을 사용하면 이 함수를 소싱하고, 삭제하고, 권한을 부여하고, 권한을 철회하고, 함수에 주석을 붙일 때 함수를 고유하게 식별할 수 있습니다. 그러나, 함수를 고유명으로 호출할 수 없습니다.

SPECIFIC절이 지정되지 않으면, 고유명이 생성됩니다.

내장 함수 확장 또는 대체

사용자 정의 함수에 내장 함수와 같은 이름을 부여하는 것은 내장 함수의 기능을 확장 또는 대체하려고 하는 경우가 아니면 권장되지 않습니다.

기존 내장 함수의 기능 확장: 내장 함수와 동일한 이름 및 고유 함수 서명을 가진 사용자 정의 함수를 작성하십시오. 예를 들어, 내장 숫자 유형이 아닌 고유 유형 MONEY를 입력으로 받아들이는 내장 ROUND 함수와 유사한 사용자 정의 함수가 필요할 수 있습니다.

이 경우, 새 사용자 정의 함수 ROUND의 서명은 내장 ROUND 함수가 지원하는 모든 함수 서명과 다릅니다.

내장 함수 대체: 기존 내장 함수와 동일한 이름 및 서명을 가진 사용자 정의 함수를 작성하십시오. 새 함수는 내장 함수의 해당 매개변수와 같은 자료 유형과 이름을 가지지만, 다른 논리로 구현됩니다. 예를 들어, 내장 ROUND 함수와 다른 반올림 규칙을 사용하는 내장 ROUND 함수와 유사한 사용자 정의 함수가 필요할 수 있습니다.

이 경우, 새 사용자 정의 함수 ROUND의 서명은 내장 ROUND 함수가 지원하는 서명과 같습니다.

일단 내장 함수가 대체되면, 규정되지 않은 함수명을 사용하며 이전에 해당 이름의 내장 함수를 사용하여 성공했던 어플리케이션이 실패하거나, 심지어는 정상적으로 실행되는 것처럼 보이지만 내장 함수가 아닌 데이터베이스 관리 프로그램에서 사용자 정의 함수가 선택되면 다른 결과를 제공합니다.

CREATE FUNCTION(외부 스칼라)

현재 서버에서 CREATE FUNCTION(외부 스칼라)문은 외부 스칼라 함수를 작성합니다. 함수는 단일 결과를 리턴합니다.

호출

이 명령문을 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSFUNCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

외부 프로그램이나 서비스 프로그램이 존재하면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SQL문에서 참조되는 외부 프로그램이나 서비스 프로그램의 경우
  - 외부 프로그램이나 서비스 프로그램이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
  - 외부 프로그램이나 서비스 프로그램에서 시스템 권한 *EXECUTE
  - 프로그램이나 서비스 프로그램에서 시스템 권한 *CHANGE. 시스템은 함수를 다른 시스템에 보관/저장하는 데 필요한 정보가 들어 있는 프로그램 오브젝트를 갱신하기 위해 이 권한을 필요로 합니다. 사용자에게 이 권한이 없으면 함수는 계속 작성되지만 프로그램 오브젝트는 갱신되지 않습니다.
- 관리 권한

SQL 이름이 지정되고 함수가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

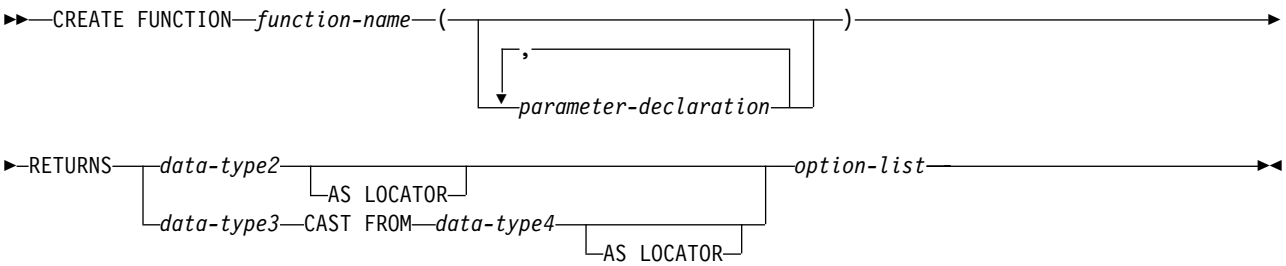
고유한 유형이 참조된다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

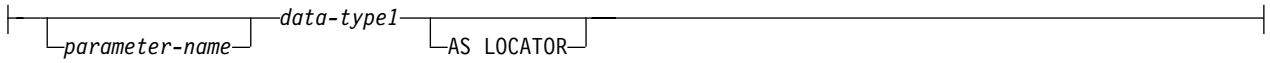
명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

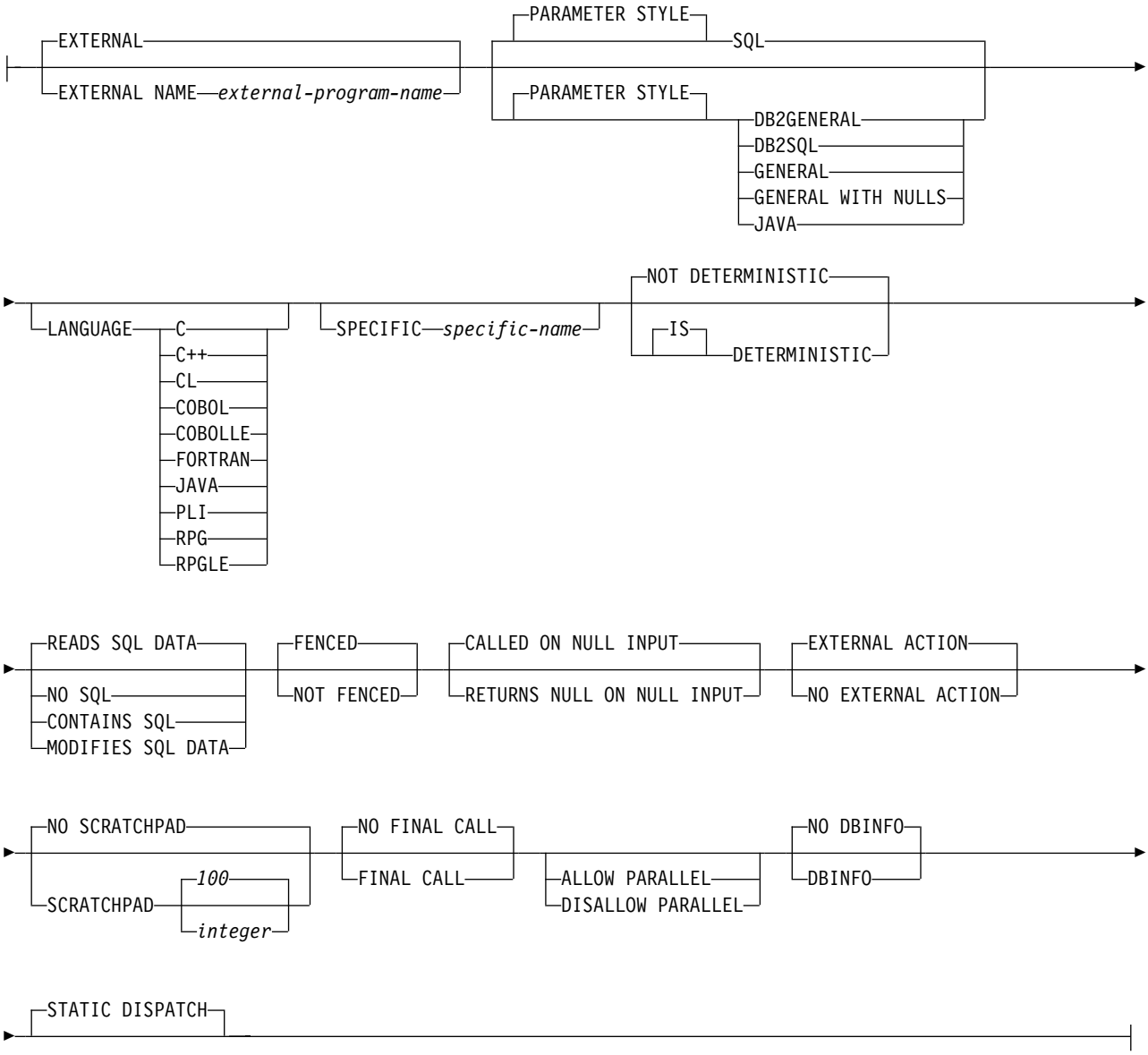
구문



parameter-declaration:

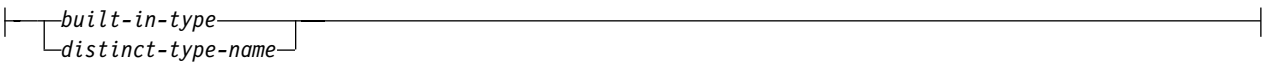


option-list:

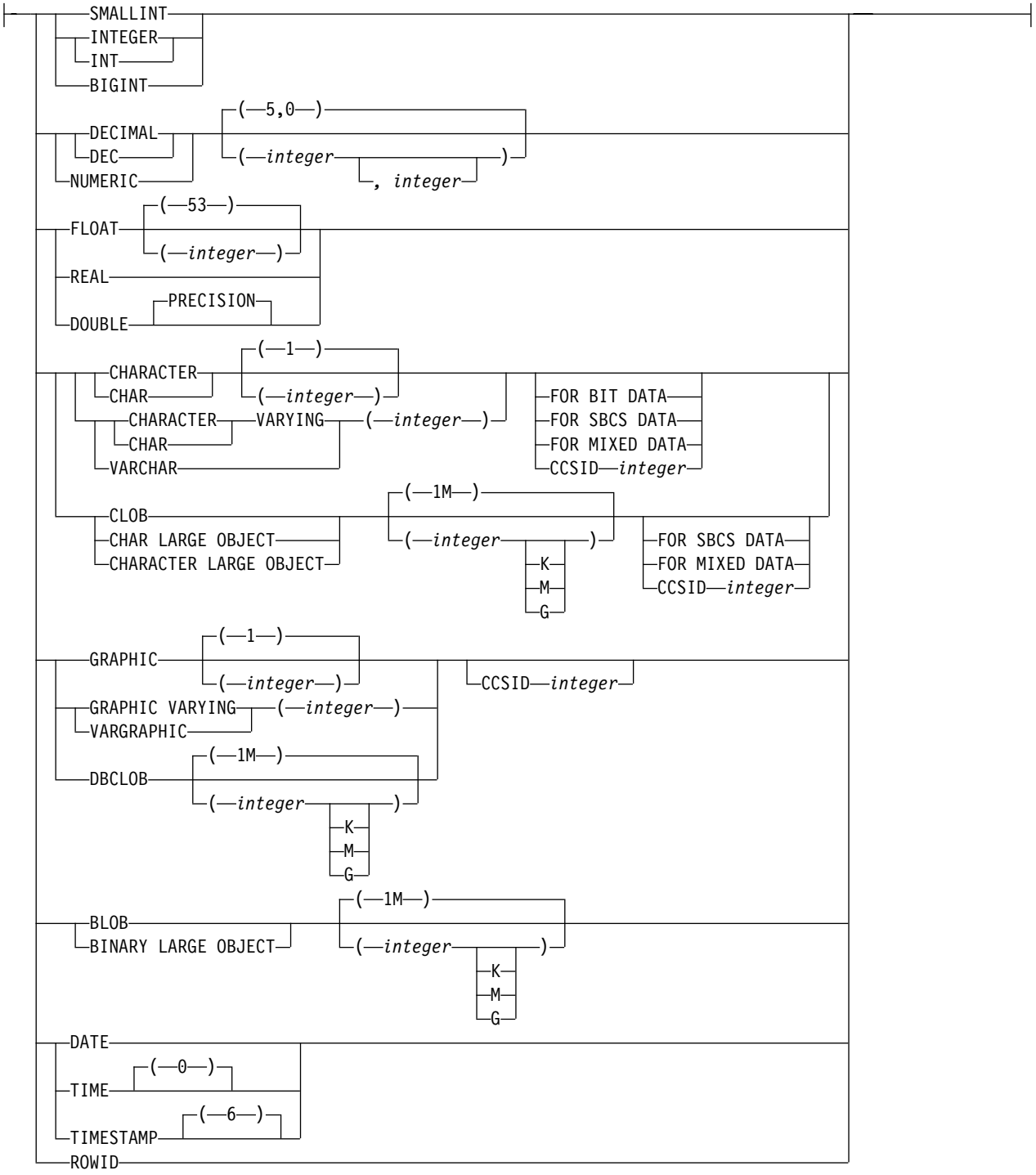

EXTERNAL
EXTERNAL NAME external-program-name
PARAMETER STYLE
SQL
PARAMETER STYLE
DB2GENERAL
DB2SQL
GENERAL
GENERAL WITH NULLS
JAVA
LANGUAGE
C
C++
CL
COBOL
COBOLLE
FORTRAN
JAVA
PLI
RPG
RPGLE
SPECIFIC specific-name
NOT DETERMINISTIC
IS
DETERMINISTIC
READS SQL DATA
NO SQL
CONTAINS SQL
MODIFIES SQL DATA
FENCED
NOT FENCED
CALLED ON NULL INPUT
RETURNS NULL ON NULL INPUT
EXTERNAL ACTION
NO EXTERNAL ACTION
NO SCRATCHPAD
SCRATCHPAD
100
integer
NO FINAL CALL
FINAL CALL
ALLOW PARALLEL
DISALLOW PARALLEL
NO DBINFO
DBINFO
STATIC DISPATCH

CREATE FUNCTION(외부 스칼라)

data-type:



built-in-type:



설명

function-name

사용자 정의 함수를 명명합니다. 이름의 조합, 스키마명, 매개변수 수 및 각 매개변수의 자료 유형(자료 유형의 길이, 정밀도, 눈금이나 CCSID 속성에 관계없이)으로 현재 서버에 있는 사용자 정의 함수를 식별해서는 안됩니다.

SQL 명명을 위해 함수는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 함수는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 함수는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면 함수는 QGPL에 작성됩니다.

일반적으로, 각 함수의 함수 표시가 고유하면 하나 이상의 함수가 같은 이름을 가질 수 있습니다.

특정 함수명은 시스템용으로 예약되어 있습니다. 자세한 내용은 443 페이지의 『함수명 선택』을 참조하십시오.

(*parameter-declaration,...)*

함수의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

CREATE FUNCTION에서 사용할 수 있는 매개변수의 최대수는 90입니다. PARAMETER STYLE SQL로 작성된 외부 함수의 경우 지정된 입력과 결과 매개변수, 인디케이터에 대한 내재적 매개변수, SQLSTATE, 함수명, 특정 이름 및 메세지 텍스트 뿐 아니라 선택적 매개변수도 포함됩니다. 매개변수의 최대수는 외부 프로그램을 컴파일하기 위해 사용되는 사용권 프로그램이 허용하는 매개변수의 최대수에 의해서도 제한됩니다.

parameter-name

입력 매개변수의 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오.

data-type1

함수의 입력 매개변수 수와 각 입력 매개변수의 자료 유형을 지정합니다. 함수의 모든 매개변수는 입력 매개변수입니다. 함수가 수신할 것으로 예상하는 각 매개변수 리스트에는 한 항목만 있어야 합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

PARAMETER STYLE JAVA가 지정된 경우, 대형 오브젝트(LOB) 자료 유형을 가진 매개변수는 지원되지 않습니다.

함수는 매개변수를 갖지 않을 수도 있습니다. 이 경우 빈 괄호 세트를 다음과 같이 코드화해야 합니다.

```
CREATE FUNCTION WOOFER ()
```

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 함수가 호출될 때 현재 서버에서 디폴트 CCSID에 의해 CCSID가 판별됩니다.

AS LOCATOR

입력 매개변수가 실제 값이 아니라 값에 대한 로케이터임을 지정합니다. 입력 매개변수가 LOB 자료 유형이나 LOB 자료 유형을 기초로 한 고유한 유형을 가질 경우에만 AS LOCATOR를 지정할 수 있습니다.

RETURNS

함수의 출력을 지정합니다.

data-type2

출력의 속성과 자료 유형을 지정합니다.

내장 자료 유형(LONG VARCHAR, LONG VARGRAPHIC 또는 DataLink 제외)이나 고유한 유형(자료 링크를 기초로 하지 않는)을 지정할 수 있습니다.

CCSID가 지정된 경우

- AS LOCATOR가 지정되지 않으면 리턴되는 결과는 해당 CCSID로 코드화되는 것으로 가정됩니다.
- AS LOCATOR가 지정되고 로케이터가 가리키는 자료의 CCSID가 다른 CCSID로 코드화된 경우 자료는 지정된 CCSID로 변환됩니다.

CCSID가 지정되지 않은 경우

- AS LOCATOR가 지정되지 않으면 리턴되는 결과는 작업의 CCSID 또는 그래픽 스트링 리턴 값에 대한 연관된 그래픽 CCSID로 코드화되는 것으로 가정됩니다.
- AS LOCATOR가 지정되는 경우 로케이터가 가리키는 자료의 CCSID가 다른 CCSID로 코드화되어 있다면 로케이터가 가리키는 자료가 작업의 CCSID로 변환됩니다. 변환시 문자가 유실될 가능성을 방지하기 위해서는 함수에서 리턴되는 모든 문자를 나타낼 수 있는 CCSID를 명시적으로 지정하는 것을 고려해보십시오. 이것은 자료 유형이 그래픽 스트링 자료일 때 특히 중요합니다. 이 경우 CCSID 13488(UCS-2 그래픽 스트링 자료)를 사용하는 것을 고려해보십시오.

data-type3 **CAST FROM** *data-type4*

출력의 자료 유형과 속성(*data-type4*), 해당 출력이 호출 명령문으로 리턴된 자료 유형(*data-type3*)을 지정합니다. 두 자료 유형은 다를 수 있습니다. 예를 들면, 다음 정의에서 함수는 데이터베이스 관리자가 DATE 값으로 변환하여 함수를 호출했던 명령문으로 전달하는 CHAR(10) 값을 리턴합니다.

```
CREATE FUNCTION GET_HIRE_DATE (CHAR6)
  RETURNS DATE CAST FROM CHAR(10)
```

data-type4 값은 고유한 유형이어서는 안되고 *data-type3*으로 캐스트될 수 있어야 합니다. *data-type3* 값은 내장 자료 유형이나 고유한 유형이 될 수 있습니다(자료 유형 캐스트에 대한 내용은 77 페이지의 『자료 유형 사이의 캐스트』를 참조하십시오).

CCSID 정보의 경우 위의 *data-type2*에 대한 설명을 참조하십시오.

AS LOCATOR

함수가 실제 값이 아니라 값에 대한 로케이터를 리턴함을 지정합니다. 함수의 출력이 LOB 자료 유형을 기초로 한 고유한 유형이나 LOB 자료 유형일 경우에만 AS LOCATOR를 지정할 수 있습니다.

SPECIFIC *specific-name*

함수에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 함수명의 규정자와 같습니다. 규정된 경우 규정자는 함수명의 규정자와 같아야 합니다.

특정 이름이 지정되지 않으면 함수명으로 설정됩니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

LANGUAGE(언어 절)

언어 절은 외부 프로그램의 언어를 지정합니다.

LANGUAGE가 지정되지 않은 경우 LANGUAGE는 함수가 작성될 때 외부 프로그램과 연관된 프로그램 속성 정보로 판별됩니다. 프로그램의 언어는 다음 경우에 C로 가정됩니다.

- 프로그램과 연관된 프로그램 속성 정보가 인식할 수 있는 언어를 식별하지 않을 때
- 프로그램을 찾을 수 없을 때

C

외부 프로그램이 C로 작성됩니다.

C++

외부 프로그램이 C++로 작성됩니다.

CL

외부 프로그램이 CL이나 ILE CL로 작성됩니다.

COBOL

외부 프로그램이 COBOL로 작성됩니다.

COBOLLE

외부 프로그램은 ILE COBOL로 작성됩니다.

FORTRAN
외부 프로그램이 FORTRAN으로 작성됩니다.

JAVA
외부 프로그램이 JAVA로 작성됩니다. 데이터베이스 관리자는 지정된 Java 클래스의 public static 메소드인 사용자 정의 함수를 호출합니다.

PLI
외부 프로그램이 PL/I으로 작성됩니다.

RPG
외부 프로그램이 RPG로 작성됩니다.

RPGLE
외부 프로그램은 ILE RPG로 작성됩니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**
함수가 결정적인지 여부를 지정합니다.

NOT DETERMINISTIC
함수가 동일한 입력 인수로 연속적인 함수 호출로부터 항상 같은 결과를 리턴하지 않음을 지정합니다. NOT DETERMINISTIC은 특수 레지스터에 대한 참조를 포함한 함수 또는 비 판별적 함수일 때 지정되어야 합니다.

DETERMINISTIC
함수가 동일한 입력 인수로 연속적인 호출로부터 항상 같은 결과를 리턴함을 지정합니다.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA 또는 **NO SQL**
함수가 SQL문을 실행할 수 있는지, 그런 경우 어떤 유형인지 지정합니다. 데이터베이스 관리자는 함수에서 실행된 SQL이 이 스펙과 일치하는지 확인합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL
함수는 자료를 읽고 수정하는 SQL문을 실행하지 않습니다.

NO SQL
함수는 실행 SQL문이 아닙니다.

READS SQL DATA
함수는 자료를 수정하는 SQL문을 실행하지 않습니다.

MODIFIES SQL DATA
함수는 함수에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있습니다.

FENCED 또는 **NOT FENCED**

함수가 호출 SQL문과 동일한 스레드에서 실행되는지 별도의 스레드에서 실행되는지 여부를 지정합니다.

FENCED

함수는 별도의 스레드에서 실행될 것입니다.

NOT FENCED

함수가 호출 SQL문과 동일한 스레드에서 실행됩니다. NOT FENCED 함수는 함수로의 각 호출에 대해 SQL 커서를 열어둡니다. 커서는 계속 열려 있으므로, 함수 호출 사이에 커서 위치 또한 보존됩니다.

NULL INPUT

입력 매개변수가 NULL일 때 함수가 호출될 필요가 있는지를 지정합니다.

CALLED ON NULL INPUT

항상 함수를 호출합니다.

RETURNS NULL ON NULL INPUT

널값이 전달되어 함수의 출력이 NULL 값이 되면 함수는 호출될 필요가 없습니다.

EXTERNAL ACTION 또는 **NO EXTERNAL ACTION**

외부 조치가 들어 있는 함수인지 여부를 지정합니다.

EXTERNAL ACTION

함수가 일부 외부 조치(함수 프로그램의 범위밖)를 수행합니다. 따라서 함수는 각각의 연속적인 함수 호출로 호출되어야 합니다. EXTERNAL ACTION은 외부 조치를 하는 다른 함수에 대한 참조를 포함한 함수일 때 지정되어야 합니다.

NO EXTERNAL ACTION

함수는 외부 조치를 수행하지 않습니다. 함수는 각각의 연속적인 함수 호출로 호출될 필요없습니다.

이 매개변수는 함수를 다음과 같이 지정합니다.

SCRATCHPAD

함수에 정적 메모리 영역을 필요로 하는지를 지정합니다.

SCRATCHPAD *integer*

함수가 길이 정수의 지속적인 메모리 영역을 필요로 함을 지정합니다. 정수의 범위는 1부터 16,000,000 사이의 값일 수 있습니다. 메모리 영역이 지정되지 않으면 영역의 크기는 100바이트입니다. 매개변수 양식 DB2SQL이 지정되면 포인터는 정적 기억장치 영역을 가르키는 필수 매개변수 다음에 전달됩니다.

PARALLEL이 지정되면 메모리 영역은 명령문의 각 사용자 정의 함수 참조에 대해 할당됩니다. DISALLOW PARALLEL이 지정되면 1 개의 메모리 영역만 함수에 대해 할당됩니다.

스크래치 패드의 범위는 SQL문입니다. SQL문의 함수에 대한 각각의 참조에는 한 개의 스크래치 패드가 있습니다. 예를 들어, 함수 UDFX가 SCRATCHPAD 키워드로 정의되었다고 가정하면 세 개의 스크래치 패드가 다음 SQL문의 UDFX에 대한 세 개의 참조에 할당됩니다.

```
SELECT A, UDFX(A) FROM TABLEB
  WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

함수가 병렬 타스크에서 실행되면 하나의 스크래치 패드가 SQL문에 있는 함수에 대한 각 참조의 각 병렬 타스크에 대해 할당됩니다. 이는 예측할 수 없는 결과를 가져옵니다. 예를 들어, 함수가 호출된 횟수를 계산하기 위해 스크래치 패드를 사용하면 횟수는 SQL문이 아닌 병렬 타스크에 의해 수행된 호출의 수를 반영합니다. 병렬로는 정확하게 실행되지 않는 함수에 대해서는 DISALLOW PARALLEL절을 지정합니다.

SCRATCHPAD는 PARAMETER STYLE DB2SQL 또는 PARAMETER STYLE DB2GENERAL과 함께만 사용할 수 있습니다.

NO SCRATCHPAD
함수가 지속적인 메모리 영역을 필요로 하지 않음을 지정합니다.

FINAL CALL
함수에 특수 호출 표시가 필요한지를 지정합니다. PARAMETER STYLE DB2SQL이 지정되고, FINAL CALL이 지정되면 추가 매개변수는 첫 번째 호출, 일반 호출 또는 최종 호출을 표시하는 함수에 전달됩니다.

NO FINAL CALL
함수를 최종 호출하지 않음을 지정합니다.

FINAL CALL
함수를 최종 호출함을 지정합니다. 최종 호출 및 기타 호출을 구별하기 위해, 함수는 호출 유형을 지정하는 추가 인수를 수신합니다.

FINAL CALL은 PARAMETER STYLE DB2SQL 또는 PARAMETER STYLE DB2GENERAL과 함께만 사용할 수 있습니다.

호출의 유형은 다음과 같습니다.

처음 호출
이 SQL문의 해당 함수 참조에 대한 함수로의 첫 번째 호출을 지정합니다. 첫 번째 호출은 일반 호출입니다. SQL 인수가 전달되며 함수는 결과를 리턴합니다.

정상 호출

SQL 인수가 전달되며 함수는 결과를 리턴하도록 지정합니다.

마지막 호출

함수가 자원을 비울 수 있도록 함수로의 최종 호출을 지정합니다. 최종 호출은 일반 호출이 아닙니다. 오류가 발생하면, 데이터베이스 관리자는 최종 호출을 합니다.

다음과 같은 경우 최종 호출이 발생합니다.

- *명령문의 끝:* 커서 지향 명령문에서 커서가 닫히거나 명령문 실행이 완료될 경우.
- *병렬 타스크의 끝:* 함수가 병렬 타스크에 의해 실행되는 경우.
- *트랜잭션의 끝:* 일반적인 명령문 처리의 끝이 발생하지 않을 경우. 예를 들어, 어떤 이유로 어플리케이션 논리가 커서 닫기를 생략할 경우.

WITH HOLD로 정의된 커서가 열려 있는 경우 확약 조작이 발생하면, 커서가 닫히거나 어플리케이션이 종료될 때 최종 호출이 이루어집니다. 병렬 타스크의 끝에서 확약이 발생하면, WITH HOLD로 정의된 커서가 열려 있는지 여부에 상관없이 최종 호출이 이루어집니다.

COMMIT 조작의 일부로 호출된 닫기 중에 FINAL CALL이 발생할 수 있으므로 확약가능한 조작은 FINAL CALL 중에 수행할 수 없습니다.

PARALLEL

함수가 병렬로 실행될 수 있는지를 지정합니다.

ALLOW PARALLEL

함수가 병렬로 실행될 수 있는지를 지정합니다.

DISALLOW PARALLEL

함수가 병렬로 실행될 수 없는지를 지정합니다.

다음 절을 둘 이상 지정하는 경우 디폴트는 DISALLOW PARALLEL입니다.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- FINAL CALL
- MODIFIES SQL DATA
- SCRATCHPAD

그렇지 않으면 ALLOW PARALLEL이 디폴트입니다.

DBINFO

함수가 전달될 데이터베이스 정보를 필요로 하는지를 지정합니다.

DBINFO

데이터베이스 관리자가 상태 정보가 들어 있는 구조를 함수에 전달해야 하는지를 지정합니다. 표 44에는 DBINFO 구조에 대한 설명이 나와 있습니다. DBINFO 구조에 대한 자세한 정보는 QSYSINC.H의 포함 파일 SQLUDF에 있습니다.

DBINFO는 PARAMETER STYLE DB2SQL 또는 PARAMETER STYLE DB2GENERAL과 함께만 사용할 수 있습니다.

표 44. DBINFO 필드

필드	자료 유형	설명
관계형 데이터베이스	VARCHAR(128)	현재 서버의 이름.
권한부여 ID	VARCHAR(128)	실행시 권한부여 ID.
CCSID 정보	INTEGER INTEGER INTEGER INTEGER CHAR(8)	작업의 CCSID 정보. 다음 정보로 CCSID를 식별합니다. • SBCS CCSID • DBCS CCSID • Mixed CCSID • 세 개의 CCSID 중 가정 적절한 CCSID 지정 • 예약 CCSID가 CREATE FUNCTION문의 매개변수에 명시적으로 지정되지 않는 경우 입력 스트링은 함수가 실행되는 시점의 작업의 CCSID로 코드화되는 것으로 가정됩니다. 입력 스트링의 CCSID가 매개변수의 CCSID와 같지 않다면 외부 함수로 전달된 입력 스트링은 외부 프로그램을 호출하기 전에 변환될 것입니다.
목표 열	VARCHAR(128) VARCHAR(128) VARCHAR(128)	사용자 정의 함수가 UPDATE문 SET 절 오른쪽에 지정되면 다음 정보로 목표 열을 식별합니다. • 스키마명 • 기본 표 이름 • 열 이름 사용자 정의 함수가 UPDATE문 SET 절의 오른쪽에 없는 경우 이 필드는 공백입니다.
버전 및 릴리스	CHAR(8)	데이터베이스 관리자의 버전 릴리스 및 수정 레벨.
플랫폼	INTEGER	서버의 플랫폼 유형.

NO DBINFO

함수가 전달될 데이터베이스 정보를 필요로 하지 않음을 지정합니다.

STATIC DISPATCH

함수가 정적으로 디스패치됨을 지정합니다. 모든 함수는 정적으로 디스패치됩니다.

EXTERNAL NAME *external-program-name*

함수가 SQL문에서 호출될 때 실행되는 프로그램, 서비스 프로그램 또는 Java 클래스를 지정합니다. 이름은 함수가 호출될 때 서버에 있는 프로그램, 서비스 프로그

램 또는 Java 클래스를 식별해야 합니다. 명명 옵션이 *SYS이고 이름이 규정되지 않으면 현재 경로는 함수가 호출될 때 프로그램이나 서비스 프로그램을 찾기 위해 사용됩니다.

이름의 유효성은 서버에서 검사됩니다. 이름의 형식이 올바르지 않으면 오류가 리턴됩니다.

external-program-name이 지정되지 않으면 외부 프로그램명은 함수명과 동일하다고 가정됩니다.

프로그램, 서비스 프로그램 또는 Java 클래스는 함수를 작성할 때는 없어도 되지만 함수를 호출할 때는 있어야 합니다.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK 및 SET TRANSACTION문은 함수의 외부 프로그램에서 허용되지 않습니다.

PARAMETER STYLE

함수로(부터) 값을 전달하고 리턴하는 데 사용되는 규약을 지정합니다.

SQL

적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.
- 함수의 결과에 대한 하나의 매개변수
- 입력 매개변수의 인디케이터 변수에 대한 N개의 매개변수
- 결과의 인디케이터 변수에 대한 매개변수 한 개
- SQLSTATE에 대한 CHAR(5) 출력 매개변수. 리턴되는 SQLSTATE는 함수의 성공 또는 실패를 표시합니다. 리턴된 SQLSTATE는 외부 프로그램에 의해 할당된 SQLSTATE입니다.

  사용자는 외부 프로그램에 모든 유효한 값으로 SQLSTATE를 설정하여 함수로부터 오류나 경고를 리턴할 수 있습니다.
- 완전 규정된 함수명에 대한 VARCHAR(517) 입력 매개변수.
- 특정 이름에 대한 VARCHAR(128) 입력 매개변수.
- 메세지 텍스트에 대한 VARCHAR(70) 출력 매개변수.

  제어가 호출한 프로그램으로 리턴될 때 메세지 텍스트는 SQLCA의 SQLERRMC 필드의 6번째 토큰에서 찾을 수 있습니다. 메세지 텍스트의 일부만을 사용할 수 있습니다. SQLERRMC의 메세지 자료의 배치에 대한 정보는 메세지 파일 QSQLMSG의 메세지 SQL0443에 대한 대체 자료 설명을 참조하십시오.

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

DB2GENERAL

이 매개변수는 Java 클래스에서 메소드로 정의된 외부 함수로 매개변수를 전달하고 해당 값을 리턴하기 위한 변환을 지정하는 데 사용됩니다. 적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.
- 함수의 결과에 대한 하나의 매개변수

DB2GENERAL은 LANGUAGE가 JAVA인 경우에만 허용됩니다.

GENERAL

적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.

결과는 함수를 리턴하는 값을 통해 리턴됩니다. 예를 들면, 다음과 같습니다.

```
return_val func(parameter-1, parameter-2, ...)
```

GENERAL은 EXTERNAL NAME이 서비스 프로그램을 식별할 때만 사용될 수 있습니다.

GENERAL WITH NULLS

적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.
- 추가 인수가 인디케이터 변수 배열에 대해 전달됩니다.
- 결과의 인디케이터 변수에 대한 매개변수 한 개

결과는 함수를 리턴하는 값을 통해 리턴됩니다. 예를 들면, 다음과 같습니다.

```
return_val func(parameter-1, parameter-2, ...)
```

GENERAL WITH NULLS는 EXTERNAL NAME이 서비스 프로그램을 식별할 때만 허용됩니다.

JAVA

이 매개변수는 Java 언어 및 SQLJ 루틴 스펙을 준수하는 매개변수 전달 규칙을 지정합니다. 적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.

결과는 함수를 리턴하는 값을 통해 리턴됩니다. 예를 들면, 다음과 같습니다.

```
return_val func(parameter-1, parameter-2, ...)
```

JAVA는 LANGUAGE가 JAVA인 경우에만 허용됩니다.

DB2SQL

적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.
- 함수의 결과에 대한 하나의 매개변수
- 입력 매개변수의 인디케이터 변수에 대한 N개의 매개변수
- 결과의 인디케이터 변수에 대한 매개변수 한 개
- SQLSTATE에 대한 CHAR(5) 출력 매개변수. 리턴되는 SQLSTATE는 함수의 성공 또는 실패를 표시합니다. 리턴되는 SQLSTATE는 다음 중 하나일 수 있습니다.
  - 외부 프로그램에서 실행된 마지막 SQL문의 SQLSTATE
  - 외부 프로그램이 할당하는 SQLSTATE

    사용자는 외부 프로그램에 모든 유효한 값으로 SQLSTATE를 설정하여 함수로부터 오류나 경고를 리턴할 수 있습니다.
- 완전 규정된 함수명에 대한 VARCHAR(517) 입력 매개변수.
- 특정 이름에 대한 VARCHAR(128) 입력 매개변수.
- 메세지 텍스트에 대한 VARCHAR(70) 출력 매개변수.
- 0 - 3개의 선택적 매개변수:
  - SCRATCH PAD가 CREATE FUNCTION문에 지정된 경우 스크래치 패드를 위한 VARCHAR(n) 입출력 매개변수.
  - FINAL CALL이 CREATE FUNCTION문에 지정된 경우 호출 유형에 대한 INTEGER 입력 매개변수.
  - DBINFO가 CREATE FUNCTION문에 지정된 경우에는 dbinfo 구조에 대한 구조.

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

외부 함수 언어에 따라 매개변수가 전달되는 방식이 결정됨에 주의하십시오. 예를 들어 C에서는 모든 VARCHAR 또는 CHAR 매개변수가 NUL 종료 스트링으로 전달됩니다. 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.

주

함수 작성

ILE 외부 프로그램이나 서비스 프로그램과 연관된 외부 함수가 작성될 때 함수의 속성을 연관된 프로그램이나 서비스 프로그램 오브젝트에 저장하려고 시도합니다. *PGM이나 *SRVPGM 오브젝트가 저장된 다음 이 시스템이나 다른 시스템으로 복원되면 카탈로그는 그 속성과 함께 자동으로 갱신됩니다.

속성은 다음 제한에 의해 외부 함수에 대해 저장될 수 있습니다.

- 외부 프로그램 라이브러리는 SYSIBM, QSYS 또는 QSYS2이어서는 안됩니다.
- CREATE FUNCTION문이 발행될 때 외부 프로그램이 있어야 합니다.
- 외부 프로그램은 ILE *PGM 또는 *SRVPGM 오브젝트여야 합니다.
- 외부 프로그램이나 서비스 프로그램에는 적어도 하나의 SQL문이 있어야 합니다.

오브젝트가 갱신될 수 없으면 함수는 여전히 작성됩니다.

함수를 복원하는 중에는 다음에 유의하십시오.

- 함수가 처음 작성될 때 특정 이름이 지정되었으나 고유하지 않으면 오류가 발행됩니다.
- 특정 이름이 지정되지 않았으면 필요할 때 고유 이름이 생성됩니다.
- 표시가 고유하지 않으면 함수는 등록될 수 없으며 오류가 발행됩니다.

함수 호출

외부 함수가 호출되면 외부 프로그램이나 서비스 프로그램이 작성되었을 때 지정된 활성 그룹에서 실행됩니다. 그러나, 함수가 호출하는 프로그램과 동일한 활성 그룹에서 실행되도록 일반적으로 ACTGRP(*CALLER)는 사용되어야 합니다. ACTGRP(*NEW)는 허용되지 않습니다.

Java 함수에 대한 주

Java 함수를 실행할 수 있으려면 시스템에 Developer Kit for Java (5722-JV1)을 설치해야 합니다. 그렇지 않으면 -443의 SQLCODE가 리턴되고 CPDB521 메세지가 작업 기록부에 기록됩니다.

Java 프로시듀어를 실행할 때 오류가 발생하면 -443의 SQLCODE가 리턴됩니다. 오류에 따라서 프로시듀어가 실행된 작업의 작업 기록부에 다른 메세지가 있을 수 있습니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.
- 키워드 SIMPLE CALL은 GENERAL의 동의어로 사용될 수 있습니다.
- DB2GENRL 값은 DB2GENERAL의 동의어로 사용될 수 있습니다.

예 1

다음 논리를 구현하는 외부 함수 서비스 프로그램을 C로 작성한다고 가정합니다.

```
output = 2 * input - 4
```

입력 인수 중 하나가 널(null)이면 함수는 널값을 리턴해야 합니다. 입력 값이 널일 때 함수 호출을 하지 않고 널 결과를 구하는 가장 간단한 방법은 CREATE FUNCTION 문에 RETURNS NULL ON NULL INPUT을 지정하는 것입니다. 그러나, 다음 예는 입력 매개변수가 널이면 널을 리턴하는 코드를 포함합니다. 함수를 작성하는 데 필요한 명령문을 특정 이름 MINENULL1을 사용하여 작성합니다.

```
CREATE FUNCTION NTEST1 (INTEGER)
  RETURNS INTEGER
  EXTERNAL NAME 'MYLIB/NTESTMOD(nudft1)'
  SPECIFIC MINENULL1
  LANGUAGE C
  DETERMINISTIC
  NO SQL
  PARAMETER STYLE DB2SQL
  CALLED ON NULL INPUT
  NO EXTERNAL ACTION
```

프로그램 코드는 다음과 같습니다.

```
  void nudft1
    (int *input,              /* ptr to input arg             */
     int *output,             /* ptr to output arg            */
     short *input_ind,          /* ptr to input indicator       */
     short *output_ind,         /* ptr to output indicator      */
     char sqlstate[6], /* sqlstate                      */
     char fname[140],  /* fully qualified function name */
     char finst[129],  /* function specific name       */
     char msgtext[71]) /* msg text buffer              */
  {
    if (*input_ind == -1)
      *output_ind = -1;
    else
  {
      *output = 2*(*input)-4;
      *output_ind = 0;
    }
    return;
  }
```

예 2

사용자 McBride(관리 권한을 가지고 있음)가 SMITH 스키마(schema)에 CENTER라는 외부 함수를 작성한다고 가정합니다. McBride는 함수에 특정 이름 FOCUS98을 지정합니다. 함수 프로그램은 일부 초기화를 수행하고 결과를 저장하기 위해 스크래치 패드를 사용합니다. 함수 프로그램은 FLOAT 자료 유형의 값을 리턴합니다. McBride가 함수를 작성하기 위해 필요한 명령문을 작성하고 함수가 호출될 때 DECIMAL(8,4)의 자료 유형을 갖는 값을 리턴합니다.

```
CREATE FUNCTION SMITH.CENTER (FLOAT, FLOAT, FLOAT)
  RETURNS DECIMAL(8,4) CAST FROM FLOAT
  EXTERNAL NAME CMOD
  SPECIFIC FOCUS98
  LANGUAGE C
  DETERMINISTIC
  NO SQL
  PARAMETER STYLE DB2SQL
  RETURNS NULL ON NULL INPUT
  NO EXTERNAL ACTION
  SCRATCHPAD
  NO FINAL CALL
```

CREATE FUNCTION(외부 스칼라)

현재 서버에서 CREATE FUNCTION(외부 표)문은 외부 표 함수를 작성합니다. 함수는 결과 표를 리턴합니다.

*표 함수*를 SELECT의 FROM절에 사용할 수 있으며 한 번에 한 행을 리턴하여 SELECT에 표를 리턴합니다.

호출

이 명령문을 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSFUNCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

외부 프로그램이나 서비스 프로그램이 존재하면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SQL문에서 참조되는 외부 프로그램이나 서비스 프로그램의 경우
  - 외부 프로그램이나 서비스 프로그램이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
  - 외부 프로그램이나 서비스 프로그램에서 시스템 권한 *EXECUTE
  - 프로그램이나 서비스 프로그램에서 시스템 권한 *CHANGE. 시스템은 함수를 다른 시스템에 보관/저장하는 데 필요한 정보가 들어 있는 프로그램 오브젝트를 갱신하기 위해 이 권한을 필요로 합니다. 사용자에게 이 권한이 없으면 함수는 계속 작성되지만 프로그램 오브젝트는 갱신되지 않습니다.
- 관리 권한

SQL 이름이 지정되고 함수가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

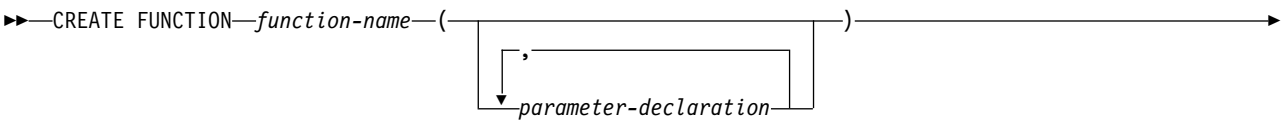
고유한 유형이 참조되다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

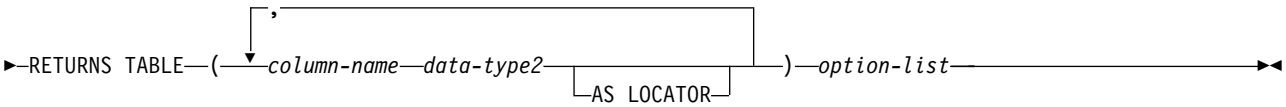
- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

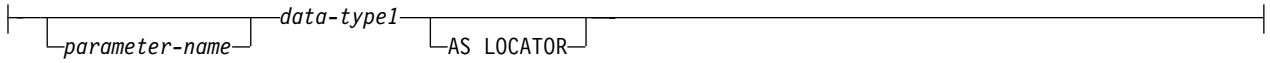
- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문

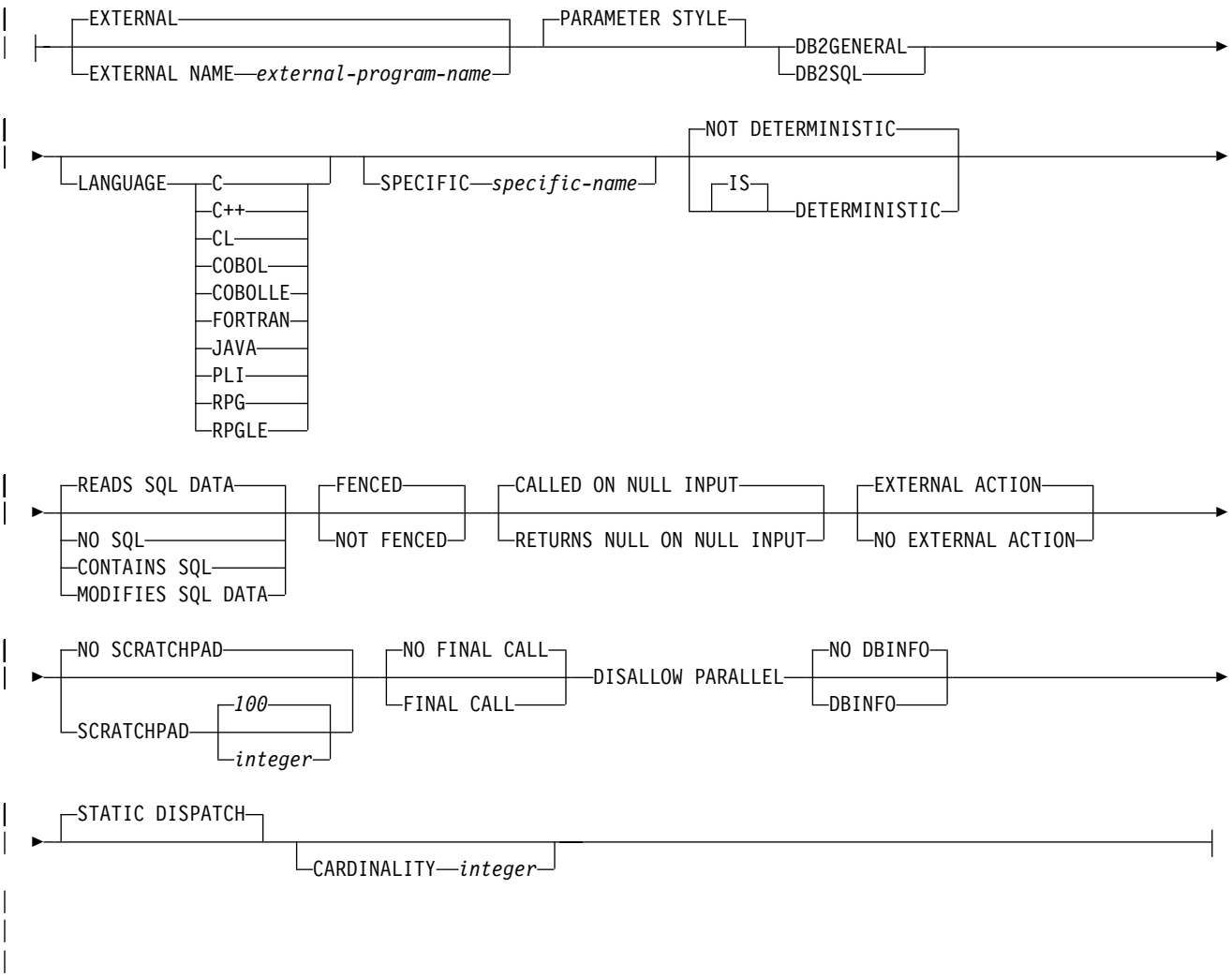




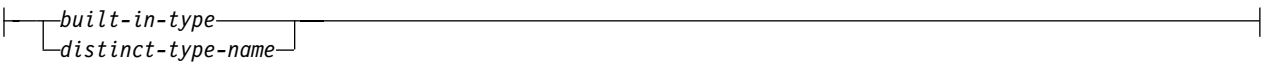
parameter-declaration:



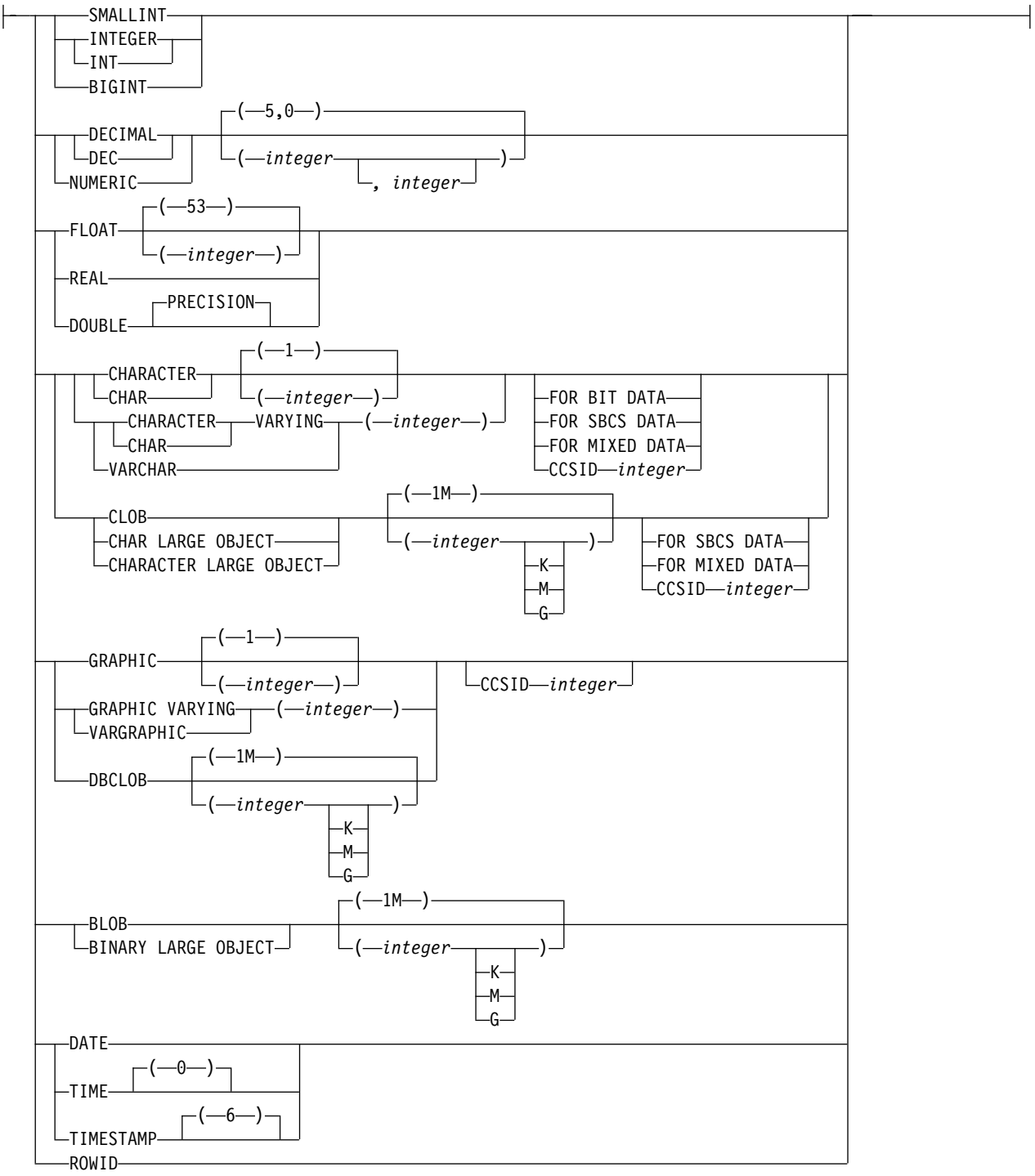
option-list:


EXTERNAL
EXTERNAL NAME external-program-name
PARAMETER STYLE
DB2GENERAL
DB2SQL
LANGUAGE
C
C++
CL
COBOL
COBOLLE
FORTRAN
JAVA
PLI
RPG
RPGLE
SPECIFIC specific-name
NOT DETERMINISTIC
IS
DETERMINISTIC
READS SQL DATA
NO SQL
CONTAINS SQL
MODIFIES SQL DATA
FENCED
NOT FENCED
CALLED ON NULL INPUT
RETURNS NULL ON NULL INPUT
EXTERNAL ACTION
NO EXTERNAL ACTION
NO SCRATCHPAD
SCRATCHPAD
100
integer
NO FINAL CALL
FINAL CALL
DISALLOW PARALLEL
NO DBINFO
DBINFO
STATIC DISPATCH
CARDINALITY integer

data-type:



built-in-type:



설명

function-name

사용자 정의 함수를 명명합니다. 이름의 조합, 스키마명, 매개변수 수 및 각 매개변수의 자료 유형(자료 유형의 길이, 정밀도, 눈금이나 CCSID 속성에 관계없이)으로 현재 서버에 있는 사용자 정의 함수를 식별해서는 안됩니다.

SQL 명명을 위해 함수는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 함수는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 함수는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면 함수는 QGPL에 작성됩니다.

일반적으로, 각 함수의 함수 표시가 고유하면 하나 이상의 함수가 같은 이름을 가질 수 있습니다.

특정 함수명은 시스템용으로 예약되어 있습니다. 자세한 내용은 443 페이지의 『함수명 선택』을 참조하십시오.

(*parameter-declaration,...)*

함수의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

CREATE FUNCTION(외부 표)에서 사용할 수 있는 매개변수의 최대수는 90입니다. 매개변수의 최대수는 외부 프로그램을 컴파일하기 위해 사용되는 사용권 프로그램이 허용하는 매개변수의 최대수에 의해서도 제한됩니다.

parameter-name

입력 매개변수의 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오.

data-type1

함수의 입력 매개변수 수와 각 입력 매개변수의 자료 유형을 지정합니다. 함수의 모든 매개변수는 입력 매개변수입니다. 함수가 수신할 것으로 예상하는 각 매개변수 리스트에는 한 항목만 있어야 합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

PARAMETER STYLE JAVA가 지정된 경우, 대형 오브젝트(LOB) 자료 유형을 가진 매개변수는 지원되지 않습니다.

함수는 매개변수를 갖지 않을 수도 있습니다. 이 경우 빈 괄호 세트를 다음과 같이 코드화해야 합니다.

```
CREATE FUNCTION WOOFER ()
```

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 함수가 호출될 때 현재 서버에서 CCSID에 의해 CCSID가 판별됩니다.

AS LOCATOR

입력 매개변수가 실제 값이 아니라 값에 대한 로케이터임을 지정합니다. 입력 매개변수가 LOB 자료 유형이나 LOB 자료 유형을 기초로 한 고유한 유형을 가질 경우에만 AS LOCATOR를 지정할 수 있습니다.

RETURNS TABLE

함수의 출력 표를 지정합니다.

매개변수 수는 N이라고 할 때, PARAMETER STYLE DB2GENERAL에는 (255-(N*2))/2개 이하의 열이 와야 합니다. PARAMETER STYLE DB2SQL의 경우, (247-(N*2))/2개 이하의 열이 와야 합니다.

열 이름

출력 표의 열 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오.

data-type2

출력의 속성과 자료 유형을 지정합니다.

내장 자료 유형(LONG VARCHAR, LONG VARGRAPHIC 또는 DataLink 제외)이나 고유한 유형(자료 링크를 기초로 하지 않는)을 지정할 수 있습니다.

DATE 또는 TIME이 지정된 경우, 표 함수는 날짜나 시간을 ISO 형식으로 리턴해야 합니다.

CCSID가 지정된 경우

- AS LOCATOR가 지정되지 않으면 리턴되는 결과는 해당 CCSID로 코드화되는 것으로 가정됩니다.
- AS LOCATOR가 지정되고 로케이터가 가리키는 자료의 CCSID가 다른 CCSID로 코드화된 경우 자료는 지정된 CCSID로 변환됩니다.

CCSID가 지정되지 않은 경우

- AS LOCATOR가 지정되지 않으면 리턴되는 결과는 작업의 CCSID 또는 그래픽 스트링 리턴 값에 대한 연관된 그래픽 CCSID로 코드화되는 것으로 가정됩니다.
- AS LOCATOR가 지정되는 경우 로케이터가 가리키는 자료의 CCSID가 다른 CCSID로 코드화되어 있다면 로케이터가 가리키는 자료가 작업의 CCSID로 변환됩니다. 변환시 문자가 유실될 가능성을 방지하기 위해서는 함수에서 리턴되는 모든 문자를 나타낼 수 있는 CCSID를 명시적으로 지정하는 것을 고려해보십시오. 이것은 자료 유형이 그래픽 스트링 자료일 때 특히 중요합니다. 이 경우 CCSID 13488(UCS-2 그래픽 스트링 자료)를 사용하는 것을 고려해보십시오.

AS LOCATOR

함수가 실제 값이 아니라 열의 값에 대한 로케이터를 리턴함을 지정합니다.

LOB 자료 유형을 기초로 한 고유한 유형이나 LOB 자료 유형일 경우에만 AS LOCATOR를 지정할 수 있습니다.

SPECIFIC *specific-name*

함수에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 함수명의 규정자와 같습니다. 규정된 경우 규정자는 함수명의 규정자와 같아야 합니다.

특정 이름이 지정되지 않으면 함수명으로 설정됩니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

LANGUAGE(언어 절)

언어 절은 외부 프로그램의 언어를 지정합니다.

LANGUAGE가 지정되지 않은 경우 LANGUAGE는 함수가 작성될 때 외부 프로그램과 연관된 프로그램 속성 정보로 판별됩니다. 프로그램의 언어는 다음 경우에 C로 가정됩니다.

- 프로그램과 연관된 프로그램 속성 정보가 인식할 수 있는 언어를 식별하지 않을 때
- 프로그램을 찾을 수 없을 때

C

외부 프로그램이 C로 작성됩니다.

C++

외부 프로그램이 C++로 작성됩니다.

CL

외부 프로그램이 CL이나 ILE CL로 작성됩니다.

COBOL

외부 프로그램이 COBOL로 작성됩니다.

COBOLLE

외부 프로그램은 ILE COBOL로 작성됩니다.

FORTRAN

외부 프로그램이 FORTRAN으로 작성됩니다.

JAVA

외부 프로그램이 JAVA로 작성됩니다.데이터베이스 관리자는 Java 클래스의 메소드로 사용자 정의 함수를 호출합니다.

PLI

외부 프로그램이 PL/I로 작성됩니다.

RPG
외부 프로그램이 RPG로 작성됩니다.

RPGLE
외부 프로그램은 ILE RPG로 작성됩니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**
함수가 결정적인지 여부를 지정합니다.

NOT DETERMINISTIC
함수가 동일한 입력 인수로 연속적인 함수 호출로부터 항상 같은 결과를 리턴하지 않음을 지정합니다. NOT DETERMINISTIC은 특수 레지스터에 대한 참조를 포함한 함수 또는 비 판별적 함수 일때 지정되어야 합니다.

DETERMINISTIC
함수가 동일한 입력 인수로 연속적인 호출로부터 항상 같은 결과를 리턴함을 지정합니다.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA 또는 **NO SQL**
함수가 SQL문을 지정할 수 있는지, 그런 경우 어떤 유형인지 지정합니다. 데이터베이스 관리자는 함수에서 실행된 SQL이 이 스펙과 일치하는지 확인합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL
함수는 자료 읽기 또는 수정하는 실행 SQL문이 아닙니다.

NO SQL
함수는 실행 SQL문이 아닙니다.

READS SQL DATA
함수는 자료를 수정하는 실행 SQL문이 아닙니다.

MODIFIES SQL DATA
함수는 함수에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있습니다.

FENCED 또는 **NOT FENCED**
함수가 호출 SQL문과 동일한 스레드에서 실행되는지 별도의 스레드에서 실행되는지 여부를 지정합니다.

FENCED
함수는 분리 스레드에서 실행될 것입니다. 함수에 SQL 커서가 들어 있는 경우 FENCED는 가장 안전한 옵션입니다. 동일한 SQL문에서 동일한 함수를 여러 번 호출할 경우 서로 충돌할 수 있기 때문입니다.

NOT FENCED

함수는 호출 SQL문과 동일한 스레드에서 실행됩니다. NOT FENCED 함수는 함수로의 각 호출에 대해 SQL 커서를 열어둡니다. 커서는 계속 열려 있으므로, 함수 호출 사이에 커서 위치 또한 보존됩니다.

NULL INPUT

입력 매개변수가 NULL일 때 함수가 호출될 필요가 있는지를 지정합니다.

CALLED ON NULL INPUT

항상 함수를 호출합니다.

RETURNS NULL ON NULL INPUT

표 함수 OPEN시, 함수의 인수 중 하나가 널인 경우 사용자 정의 표 함수는 호출되지 않으며 함수 출력은 빈 표(행 없는 표)가 됩니다.

EXTERNAL ACTION 또는 **NO EXTERNAL ACTION**

외부 조치가 들어 있는 함수인지 여부를 지정합니다.

EXTERNAL ACTION

함수가 일부 외부 조치(함수 프로그램의 범위밖)를 수행합니다. 따라서 함수는 각각의 연속적인 함수 호출로 호출되어야 합니다. EXTERNAL ACTION은 외부 조치를 하는 다른 함수에 대한 참조를 포함한 함수일 때 지정되어야 합니다.

NO EXTERNAL ACTION

함수는 외부 조치를 수행하지 않습니다. 함수는 각각의 연속적인 함수 호출로 호출될 필요없습니다.

SCRATCHPAD

함수에 정적 메모리 영역을 필요로 하는지를 지정합니다.

SCRATCHPAD *integer*

함수가 길이 정수의 지속적인 메모리 영역을 필요로 함을 지정합니다. 정수의 범위는 1부터 16,000,000 사이의 값일 수 있습니다. 메모리 영역이 지정되지 않으면 영역의 크기는 100바이트입니다. 매개변수 양식 DB2SQL이 지정되면 포인터는 정적 기억장치 영역을 가르키는 필수 매개변수 다음에 전달됩니다. 함수에 대해서 단 1개의 메모리 영역만 할당될 것입니다.

스크래치 패드의 범위는 SQL문입니다. SQL문의 함수에 대한 각각의 참조에는 한 개의 스크래치 패드가 있습니다. 예를 들어, 함수 UDFX가 SCRATCHPAD 키워드로 정의되었다고 가정하면 두 개의 스크래치 패드가 다음 SQL문의 UDFX에 대한 두 개의 참조에 할당됩니다.

```
SELECT A.C1, B.C1
  FROM TABLE(UDFX(:hv1)) AS A, TABLE(UDFX(:hv1)) AS B
```

NO SCRATCHPAD

함수가 지속적인 메모리 영역을 필요로 하지 않음을 지정합니다.

FINAL CALL

함수에 마지막 호출(및 분리된 처음 호출)을 필요로 하는지를 지정합니다. 표 함수의 경우, 호출 유형 인수는 선택된 FINAL CALL 옵션과는 상관없이 항상 존재합니다. 호출 유형 인수는 첫 번째 호출, 열린 호출, 페치 호출, 닫기 호출 또는 최종 호출을 나타냅니다.

FINAL CALL

함수에 마지막 호출(및 분리된 처음 호출)을 필요로 함을 지정합니다. 또한 스크래치패드가 다시 초기화되는 때를 제어합니다. NO FINAL CALL이 지정된 경우, 데이터베이스 관리자는 표 함수에 세 가지 유형의 호출(열기, 페치, 닫기)만 수행할 수 있습니다. 그러나, FINAL CALL이 지정되면, 열기 외에도 페치, 닫기, 첫 번째 호출 및 최종 호출을 표 함수에 대해 수행할 수 있습니다. 함수에 최종 호출을 수행하도록 지정합니다. 최종 호출 및 기타 호출을 구별하기 위해, 함수는 호출 유형을 지정하는 추가 인수를 수신합니다.

호출의 유형은 다음과 같습니다.

처음 호출
이 SQL문의 해당 함수 참조에 대한 함수로의 첫 번째 호출을 지정합니다.

열기 호출
이 SQL문의 표 함수 결과를 열기 위한 호출을 지정합니다.

페치 호출
이 SQL문의 표 함수로부터 행을 페치하기 위한 호출을 지정합니다.

닫기 호출
이 SQL문의 표 함수 결과를 닫기 위한 호출을 지정합니다.

마지막 호출
함수가 자원을 비울 수 있도록 함수로의 최종 호출을 지정합니다. 오류가 발생하면, 데이터베이스 관리자는 최종 호출을 합니다.

다음과 같은 경우 최종 호출이 발생합니다.

- *명령문의 끝*: 커서 지향 명령문에서 커서가 닫히거나 명령문 실행이 완료될 경우.
- *트랜잭션의 끝*: 일반적인 명령문 처리의 끝이 발생하지 않을 경우. 예를 들어, 어떤 이유로 어플리케이션 논리가 커서 닫기를 생략할 경우.

WITH HOLD로 정의된 커서가 열려 있는 경우 확약 조작이 발생하면, 커서가 닫히거나 어플리케이션이 종료될 때 최종 호출이 이루어집니다.

COMMIT 조작의 일부로 호출된 닫기 중에 FINAL CALL이 발생할 수 있으므로 확약가능한 조작은 FINAL CALL 중에 수행할 수 없습니다.

NO FINAL CALL

함수에 마지막 호출(및 분리된 처음 호출)을 필요로 하지 않음을 지정합니다. 그러나 열린 호출, 페치 호출, 닫기 호출은 여전히 수행됩니다.

DISALLOW PARALLEL

함수가 병렬로 실행될 수 없는지를 지정합니다. 표 함수는 병렬로 실행될 수 없습니다.

DBINFO

함수가 전달될 데이터베이스 정보를 필요로 하는지를 지정합니다.

DBINFO

데이터베이스 관리자가 상태 정보가 들어 있는 구조를 함수에 전달해야 하는지를 지정합니다. 표 45에는 DBINFO 구조에 대한 설명이 나와 있습니다. DBINFO 구조에 대한 자세한 정보는 QSYSINC.H의 포함 파일 SQLUDF에 있습니다.

표 45. DBINFO 필드

필드	자료 유형	설명
관계형 데이터베이스	VARCHAR(128)	현재 서버의 이름.
권한부여 ID	VARCHAR(128)	실행시 권한부여 ID.
CCSID 정보	INTEGER INTEGER INTEGER INTEGER CHAR(8)	작업의 CCSID 정보. 다음 정보로 CCSID를 식별합니다. • SBCS CCSID • DBCS CCSID • Mixed CCSID • 세 개의 CCSID 중 가정 적절한 CCSID 지정 • 예약 CCSID가 CREATE FUNCTION문의 매개변수에 명시적으로 지정되지 않는 경우 입력 스트링은 함수가 실행되는 시점의 작업의 CCSID로 코드화되는 것으로 가정됩니다. 입력 스트링의 CCSID가 매개변수의 CCSID와 같지 않다면 외부 함수로 전달된 입력 스트링은 외부 프로그램을 호출하기 전에 변환될 것입니다.
목표 열	VARCHAR(128) VARCHAR(128) VARCHAR(128)	사용자 정의 함수가 UPDATE문 SET 절 오른쪽에 지정되면 다음 정보로 목표 열을 식별합니다. • 스키마명 • 기본 표 이름 • 열 이름 사용자 정의 함수가 UPDATE문 SET 절의 오른쪽에 없는 경우 이 필드는 공백입니다.
버전 및 릴리스	CHAR(8)	데이터베이스 관리자의 버전 릴리스 및 수정 레벨.
플랫폼	INTEGER	서버의 플랫폼 유형.

표 45. DBINFO 필드 (계속)

필드	자료 유형	설명
표 함수 열 리스트 항목의 수	SMALLINT	아래의 ″표 함수 열 리스트″ 필드에 지정된 표 함수 열 리스트의 0이 아닌 항목 수.
예약	CHAR(24)	향후 용도로 예약되어 있습니다.
표 함수 열 리스트	Pointer(16 Bytes)	이 필드는 데이터베이스 관리자에 의해 동적으로 할당된 짧은 정수 배열을 가리키는 포인터입니다. 처음 n 항목(n은 ″표 함수 열 리스트 항목 수: 필드에 지정됨)만 처리되며, n은 0일 수 있고, RETURNS TABLE절의 함수에 대해 정의된 결과 열 수보다 작거나 같습니다. 값은 이 명령문이 표 함수에서 필요로 하는 열의 서수에 해당합니다. 값 1은 정의된 첫 번째 열을 의미하고, 2는 정의된 결과 열 등을 의미합니다. 값은 순서에 관계 없습니다. SELECT COUNT(*) FROM TABLE(TF(...))과 유사한 명령문의 경우 n은 0과 같음을 유의하십시오. 조회에 실제 열 값이 필요하지 않은 QQ의 경우와 같습니다. 이 배열은 최적화 기회를 나타냅니다. 함수는 표 함수의 모든 결과 열에 대한 모든 값을 리턴하지는 않습니다. 특정 문맥에서는 값의 서브세트만이 필요하며, 이들은 배열에 지정된(번호로) 열입니다. 이 최적화로 인해 함수 논리가 복잡해질 수 있기 때문에, 함수는 정의된 모든 열을 리턴하도록 선택할 수 있습니다.

NO DBINFO

함수가 전달될 데이터베이스 정보를 필요로 하지 않음을 지정합니다.

CARDINALITY *integer*

이 선택적 절은 최적화를 위해 함수가 리턴할 예상 행 수에 대한 예상치를 제공합니다. 정수의 유효 값 범위는 0 - 2 147 483 647입니다.

CARDINALITY절이 표 함수에 대해 지정되지 않은 경우, 데이터베이스 관리자는 유한 값을 디폴트로 가정합니다.

경고: 함수에 실제로 무한 규칙이 설정되어 있지 않은 경우, 즉 호출될 때마다 행을 리턴하고 end-of-table 조건을 리턴하지 않는 경우, end-of-table 조건을 필요로 하는 조회 또한 무한이 되어 중단해야 합니다. 그러한 조회의 예를 들면, GROUP BY 및 ORDER BY와 관련된 것들이 있습니다. 사용자는 그러한 UDF를 작성하지 않는 것이 좋습니다.

STATIC DISPATCH

함수가 정적으로 디스패치됨을 지정합니다. 모든 함수는 정적으로 디스패치됩니다.

EXTERNAL NAME *external-program-name*

함수가 SQL문에서 호출될 때 실행되는 프로그램, 서비스 프로그램 또는 Java 클래스를 지정합니다. 이름은 함수가 호출될 때 서버에 있는 프로그램, 서비스 프로그램, 또는 Java 클래스를 식별해야 합니다. 명명 옵션이 *SYS이고 이름이 규정되지 않으면 현재 경로는 함수가 호출될 때 프로그램이나 서비스 프로그램을 찾기 위해 사용됩니다.

이름의 유효성은 서버에서 검사됩니다. 이름의 형식이 올바르지 않으면 오류가 리턴됩니다.

external-program-name이 지정되지 않으면 외부 프로그램명은 함수명과 동일하다고 가정됩니다.

프로그램, 서비스 프로그램 또는 Java 클래스는 함수를 작성할 때는 없어도 되지만 함수를 호출할 때는 있어야 합니다.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK 및 SET TRANSACTION문은 함수의 외부 프로그램에서 허용되지 않습니다.

PARAMETER STYLE

함수로(부터) 값을 전달하고 리턴하는 데 사용되는 규약을 지정합니다.

DB2GENERAL

이 매개변수는 Java 클래스에서 메소드로 정의된 외부 함수로 매개변수를 전달하고 해당 값을 리턴하기 위한 변환을 지정하는 데 사용됩니다. 적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.
- 다음 M 매개변수는 RETURNS TABLE절에 지정된 함수의 결과 열입니다.

DB2GENERAL은 LANGUAGE가 JAVA인 경우에만 허용됩니다.

DB2SQL

적용할 수 있는 모든 매개변수가 전달됩니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE FUNCTION문에 지정된 입력 매개변수입니다.
- 다음 M 매개변수는 RETURNS TABLE절에 지정된 함수의 결과 열입니다.
- 입력 매개변수의 인디케이터 변수에 대한 N개의 매개변수
- RETURNS TABLE절에 지정된 함수의 결과 열의 인디케이터 변수에 대한 M개의 매개변수
- SQLSTATE에 대한 CHAR(5) 출력 매개변수. 리턴되는 SQLSTATE는 함수의 성공 또는 실패를 표시합니다. 리턴되는 SQLSTATE는 다음 중 하나일 수 있습니다.
  - 외부 프로그램애서 실행된 마지막 SQL문의 SQLSTATE
  - 외부 프로그램이 할당하는 SQLSTATE

    사용자는 외부 프로그램에 모든 유효한 값으로 SQLSTATE를 설정하여 함수으로부터 오류나 경고를 리턴할 수 있습니다.
- 완전 규정된 함수명에 대한 VARCHAR(517) 입력 매개변수.
- 특정 이름에 대한 VARCHAR(128) 입력 매개변수.

- 메세지 텍스트에 대한 VARCHAR(70) 출력 매개변수.
- SCRATCH PAD가 CREATE FUNCTION문에 지정된 경우 스크래치 패드를 위한 VARCHAR(n) 입출력 매개변수.
- 호출 유형에 대한 INTEGER 입력 매개변수.
- DBINFO가 CREATE FUNCTION문에 지정된 경우에는 dbinfo 구조에 대한 구조.

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

외부 함수 언어에 따라 매개변수가 전달되는 방식이 결정됨에 주의하십시오. 예를 들어 C에서는 모든 VARCHAR 또는 CHAR 매개변수가 NUL 종료 스트링으로 전달됩니다. 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.

주

함수 작성

ILE 외부 프로그램이나 서비스 프로그램과 연관된 외부 함수가 작성될 때 함수의 속성을 연관된 프로그램이나 서비스 프로그램 오브젝트에 저장하려고 시도합니다. *PGM이나 *SRVPGM 오브젝트가 저장된 다음 이 시스템이나 다른 시스템으로 복원되면 카탈로그는 그 속성과 함께 자동으로 갱신됩니다.

속성은 다음 제한에 의해 외부 함수에 대해 저장될 수 있습니다.

- 외부 프로그램 라이브러리는 SYSIBM, QSYS 또는 QSYS2이어서는 안됩니다.
- CREATE FUNCTION문이 발행될 때 외부 프로그램이 있어야 합니다.
- 외부 프로그램은 ILE *PGM 또는 *SRVPGM 오브젝트여야 합니다.
- 외부 프로그램이나 서비스 프로그램에는 적어도 하나의 SQL문이 있어야 합니다.

오브젝트가 갱신될 수 없으면 함수는 여전히 작성됩니다.

함수를 복원하는 중에는 다음에 유의하십시오.

- 함수가 처음 작성될 때 특정 이름이 지정되었으나 고유하지 않으면 오류가 발행됩니다.
- 특정 이름이 지정되지 않았으면 필요할 때 고유 이름이 생성됩니다.
- 표시가 고유하지 않으면 함수는 등록될 수 없으며 오류가 발행됩니다.

함수 호출

외부 함수가 호출되면 외부 프로그램이나 서비스 프로그램이 작성되었을 때 지정된 활성 그룹에서 실행됩니다. 그러나, 함수가 호출하는 프로그램과 동일한 활성 그룹에서 실행되도록 일반적으로 ACTGRP(*CALLER)는 사용되어야 합니다. ACTGRP(*NEW)는 허용되지 않습니다.

Java 함수에 대한 주

Java 함수를 실행할 수 있으려면 시스템에 Developer Kit for Java(5722-JV1)를 설치해야 합니다. 그렇지 않으면 -443의 SQLCODE가 리턴되고 CPDB521 메세지가 작업 기록부에 기록됩니다.

Java 프로시듀어를 실행할 때 오류가 발생하면 -443의 SQLCODE가 리턴됩니다. 오류에 따라서 프로시듀어가 실행된 작업의 작업 기록부에 다른 메세지가 있을 수 있습니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.
- DB2GENRL 값은 DB2GENERAL의 동의어로 사용될 수 있습니다.

예 1

다음은 텍스트 관리 시스템에서 알려진 각 문서의 단일 문서 ID로 구성되는 행을 리턴하도록 작성된 표 함수를 작성합니다. 첫 번째 매개변수는 해당 주제 영역에 해당하며 두 번째 매개변수에는 해당 스트링이 포함됩니다.

단일 세션 문맥 내에서 UDF는 항상 동일한 표를 리턴하므로 DETERMINISTIC로 정의됩니다. DOCMATCH로부터의 출력을 정의하는 RETURNS절에 유의하십시오. 각 표 함수마다 FINAL CALL을 지정해야 합니다. 또한, 표 함수로 추가된 DISALLOW PARALLEL 키워드는 병렬로 동작할 수 없습니다. DOCMATCH의 출력 *크기*는 상당히 가변적이지만, CARDINALITY 20이 대표 값으로서 optimizer를 돕기 위해 지정됩니다.

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOCID CHAR(16))
  EXTERNAL NAME 'MYLIB/RAJIV(UDFMATCH)'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

CREATE FUNCTION(피소스(sourced))

CREATE FUNCTION문은 현재 서버에 기존의 다른 스칼라 나열 함수를 기초로 사용자 정의 함수를 작성하기 위해 사용됩니다.

호출

이 명령문을 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSFUNCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

소스 함수가 사용자 정의 함수가면 명령문의 권한부여 ID는 소스 함수에 대해 적어도 다음 중 하나를 포함해야 합니다.

- 함수에서 EXECUTE 권한
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 함수에 대한 EXECUTE 권한을 갖습니다.

- 함수의 소유자입니다.
- 함수에서 EXECUTE 권한을 부여받았습니다.
- 함수에서 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

소스 함수를 작성하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 서비스 프로그램 작성(CRTSRVPGM) 명령에 대해 *USE
  - 프로그램 작성(CRTPGM) 명령에 대해 *USE
  - 함수가 작성되는 라이브러리에 대해 *EXECUTE 및 *ADD
- 관리 권한

SQL 이름이 지정되고 함수가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

고유한 유형이 참조된다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

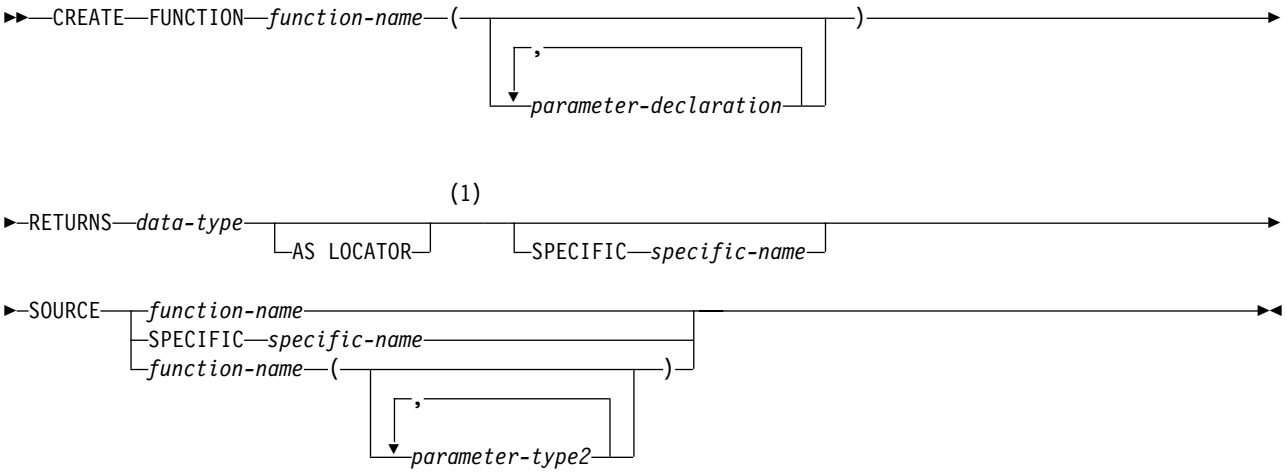
- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

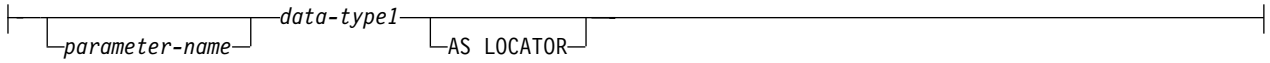
- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

CREATE FUNCTION(피소스(sourced))

구문



parameter-declaration:

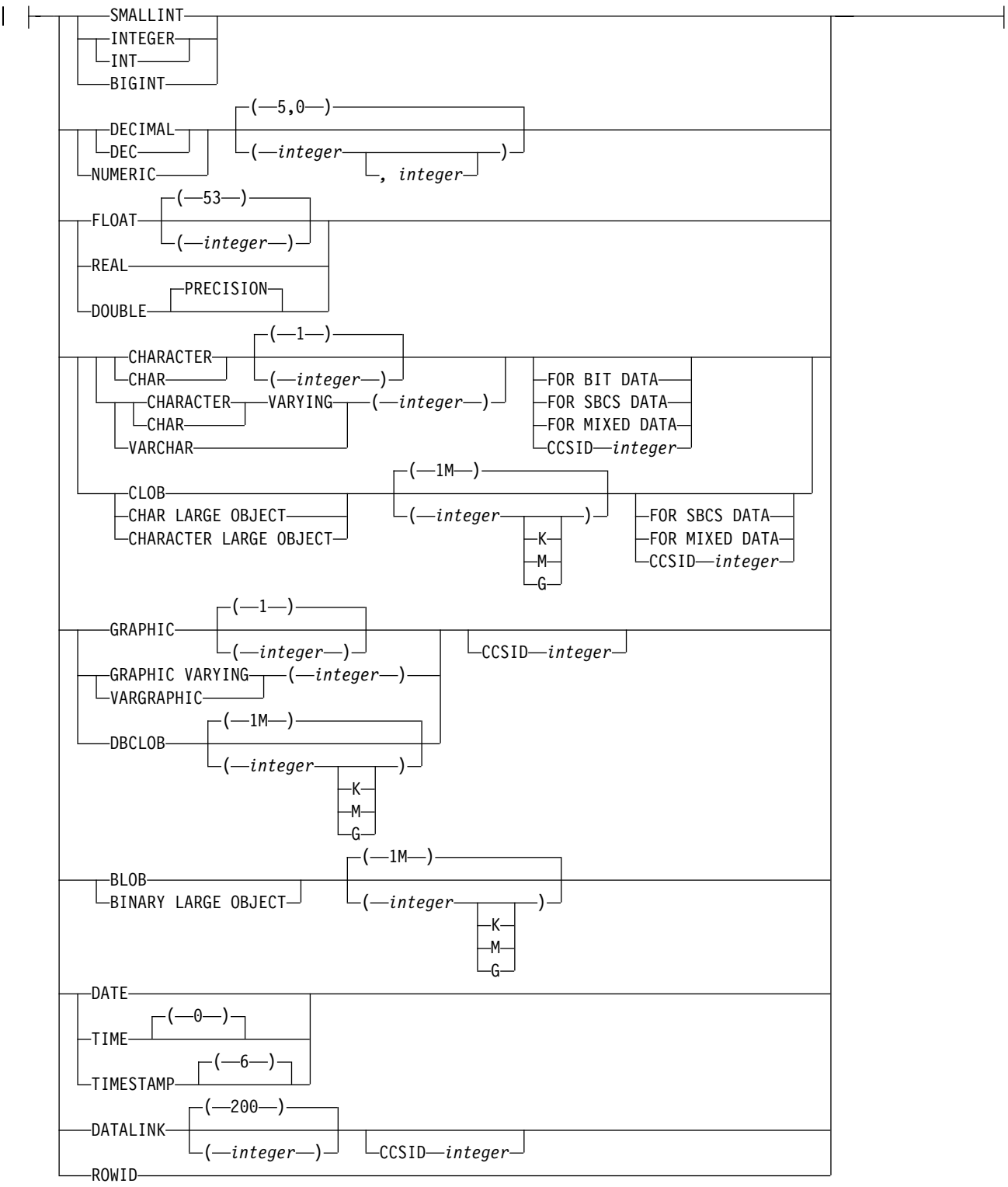


data-type:

주:

1 RETURNS, SPECIFIC 및 SOURCE절은 다른 순서로 지정할 수 있습니다.

built-in-type:



설명

function-name

사용자 정의 함수를 명명합니다. 이름의 조합, 스키마명, 매개변수 수 및 각 매개변

수의 자료 유형(자료 유형의 길이, 정밀도, 눈금이나 CCSID 속성에 관계없이)으로 현재 서버에 있는 사용자 정의 함수를 식별해서는 안됩니다.

SQL 명명을 위해 함수는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 함수는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 함수는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면 함수는 QGPL에 작성됩니다.

고유한 유형으로 기존 함수를 사용할 수 있도록 함수가 기존 함수를 소스로 하는 경우 이름은 기존 함수와 같은 이름을 가질 수 있습니다. 일반적으로, 각 함수의 함수 표시가 고유하면 하나 이상의 함수가 같은 이름을 가질 수 있습니다.

특정 함수명은 시스템용으로 예약되어 있습니다. 자세한 내용은 443 페이지의 『함수명 선택』을 참조하십시오.

(*parameter-declaration,...)*

함수의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

parameter-name

입력 매개변수의 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오.

data-type1

함수의 입력 매개변수 수와 각 입력 매개변수의 자료 유형을 지정합니다. 함수의 모든 매개변수는 입력 매개변수입니다. 함수가 수신할 것으로 예상하는 각 매개변수 리스트에는 한 항목만 있어야 합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다. 자료 링크를 외부 함수에 대해 사용할 수 없습니다.

함수는 매개변수를 갖지 않을 수도 있습니다. 이 경우 빈 괄호 세트를 다음과 같이 코드화해야 합니다.

```
CREATE FUNCTION WOOFER ()
```

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 함수가 호출될 때 현재 서버에서 디폴트 CCSID에 의해 CCSID가 판별됩니다.

AS LOCATOR

입력 매개변수가 실제 값이 아니라 값에 대한 로케이터임을 지정합니다. 입력 매개변수가 LOB 자료 유형이나 LOB 자료 유형을 기초로 한 고유한 유형을 가질 경우에만 AS LOCATOR를 지정할 수 있습니다.

RETURNS

함수의 출력을 지정합니다.

data-type

출력의 속성과 자료 유형을 지정합니다.

소스 함수의 결과 유형에서 캐스트할 수 있으면, 내장 자료 유형(LONG VARCHAR, LONG VARGRAPHIC 또는 DataLink 제외)이나 고유한 유형(자료 링크를 기초로 하지 않는)을 지정할 수 있습니다(자료 유형 캐스트에 대한 내용은 77 페이지의 『자료 유형 사이의 캐스트』를 참조하십시오).

AS LOCATOR

함수가 실제 값이 아니라 값에 대한 로케이터를 리턴함을 지정합니다. 함수의 출력이 LOB 자료 유형을 기초로 한 고유한 유형이나 LOB 자료 유형일 경우에만 AS LOCATOR를 지정할 수 있습니다. AS LOCATOR절은 SQL 함수에서 소스 함수에는 사용할 수 없습니다.

SPECIFIC *specific-name*

함수에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 함수명의 규정자와 같습니다. 규정된 경우 규정자는 함수명의 규정자와 같아야 합니다.

특정 이름이 지정되지 않으면 함수명으로 설정됩니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

SOURCE

작성하는 함수가 소스 함수임을 지정합니다. 피소스(sourced) 함수는 다른 함수(소스 함수)에 의해 구현됩니다. 소스 함수는 내장 스칼라 함수이거나 COALESCE, HASH, IFNULL, LAND, LOR, MAX, MIN, NODENAME, NODENUMBER, NULLIF, PARTITION, POSITION, RRN, STRIP, SUBSTRING, TRIM, VALUE 및 XOR, 또는 이전에 작성된 모든 사용자 정의 함수를 제외한 열 함수입니다. 시스템이 생성한 사용자 정의 함수일 수 있습니다(고유한 유형이 작성되었을 때 생성된).

소스 함수는 하나의 인수가 지정되는 경우에만 BLOB, CHAR, CLOB, DBCLOB, DECIMAL, GRAPHIC, TRANSLATE, VARCHAR, VARGRAPHIC 및 ZONED 내장 함수 중 하나일 수 있습니다.

피소스(sourced) 함수가 직간접으로 스칼라 함수에 기초하고 있으면 피소스(sourced) 함수는 스칼라 함수의 속성을 계승합니다. 피소스(sourced) 함수의 여러 계층과도 관련됩니다. 예를 들어, 함수 A가 함수 B를 소스로 사용하고, B는 함수 C를 소스로 사용하는 것으로 가정합니다. 함수 C는 스칼라 함수입니다. 함수 A와 B는 함수 C의 CREATE FUNCTION문에 지정된 모든 속성을 계승합니다.

규정되지 않은 경우 디폴트 스키마가 함수의 위치를 지정하는 데 사용됩니다.

CREATE FUNCTION(피소스(sourced))

FUNCTION *function-name*

*function-name*은 현재 서버에 있는 한 함수를 정확히 식별해야 합니다. 함수는 함수에 대해 정의된 매개변수를 가질 수 있습니다. 지정된 또는 내재된 스키마에 지정된 이름의 함수가 둘 이상 있으면 오류가 리턴됩니다.

대부분의 내장 함수는 입력 매개변수로 여러 자료 유형을 허용하도록 정의됩니다. 이들 중 하나를 소스로 사용하려면 매개변수 유형이 지정되어야 합니다.

FUNCTION *function-name(parameter-type2, ...)*

function-name(parameter-type2, ...) 현재 서버에 있는 지정된 함수 표시로 함수를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE FUNCTION 문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수와 자료 유형의 논리적 연결은 소스 함수로 사용될 특정 함수 인스턴스를 식별하기 위해 사용됩니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 눈금 속성을 갖는 자료 유형의 경우 값을 지정하거나 빈 괄호 세트를 사용할 수 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.
- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE FUNCTION문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 분명히 지정되지 않고 빈 괄호도 지정되지 않으면 자료 유형의 디폴트 속성이 내재됩니다. 디폴트 속성에 대한 자세한 정보는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE FUNCTION문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC *specific-name*

*specific-name*은 현재 서버에 있는 특정 함수를 식별해야 합니다.

작성되는 함수의 입력 매개변수 수는 소스 함수의 매개변수 수와 같아야 합니다. 각 입력 매개변수의 자료 유형이 소스 함수의 해당 매개변수와 다르거나 캐스트할 수 없으면, 오류가 발생합니다. 소스 함수의 최종 결과의 자료 유형은 피소스(sourced) 함수의 결과와 일치하거나 캐스트할 수 있어야 합니다.

CCSID가 지정되고 리턴 자료의 CCSID가 다른 CCSID에 코드화되는 경우 자료는 지정된 CCSID로 변환됩니다.

CCSID가 지정되지 않은 경우 리턴 자료의 CCSID가 다른 CCSID로 코드화되면 리턴 자료는 작업의 CCSID 또는 그래픽 스트링 리턴 값에 대한 작업의 연관된 그래픽 CCSID로 변환됩니다. 변환시 문자가 유실될 가능성을 방지하기 위해서는 함수에서 리턴되는 모든 문자를 나타낼 수 있는 CCSID를 명시적으로 지정하는 것을 고려해보십시오. 이것은 자료 유형이 그래픽 스트링 자료일 때 특히 중요합니다. 이 경우 CCSID 13488(UCS-2 그래픽 스트링 자료)을 사용하는 것을 고려해보십시오.

주

피소스(sourced) 함수가 작성될 때 함수를 표시하는 작은 서비스 프로그램 오브젝트가 작성됩니다. 이 서비스 프로그램이 저장되고 다른 시스템으로 복원될 때 CREATE FUNCTION문의 속성은 그 시스템의 카탈로그에 자동으로 추가됩니다.

소스 함수에 지정된 모든 속성이 신규 피소스(sourced) 함수(예를 들면, PARAMETER STYLE, STATIC DISPATCH, 등)에 전달됩니다.

예 1

내장 INTEGER 자료 유형을 기초로 하는 고유한 유형 HATSIZE를 작성한다고 가정합니다. AVG 함수를 작성하여 다른 부서의 평균 모자 크기를 계산합니다.

```
EXEC SQL
CREATE FUNCTION AVG (HATSIZE) RETURNS HATSIZE
SOURCE AVG (INTEGER);
```

예 2

Smith가 자신의 스키마에 외부 스칼라 함수인 CENTER를 등록했으며 다른 사용자가 이 함수를 사용할 때 하나의 INTEGER 인수와 하나의 FLOAT 인수 대신 두 개의 INTEGER 인수를 사용하려고 합니다. 이 경우 CENTER를 기초로 소스 함수를 작성합니다.

```
EXEC SQL
CREATE FUNCTION MYCENTERG (INTEGER, INTEGER)
  RETURNS FLOAT
  SOURCE SMITH.CENTER (INTEGER, FLOAT);
```

CREATE FUNCTION(SQL 스칼라)

이 CREATE FUNCTION(SQL 스칼라)문은 현재 서버에서 SQL 함수를 작성합니다. 함수는 단일 결과를 리턴합니다.

호출

이 명령문을 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSFUNCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 서비스 프로그램 작성(CRTSRVPGM) 명령에 대해 *USE
  - 함수가 작성되는 라이브러리에 대해 *EXECUTE 및 *ADD
- 관리 권한

고유한 유형이 참조되다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

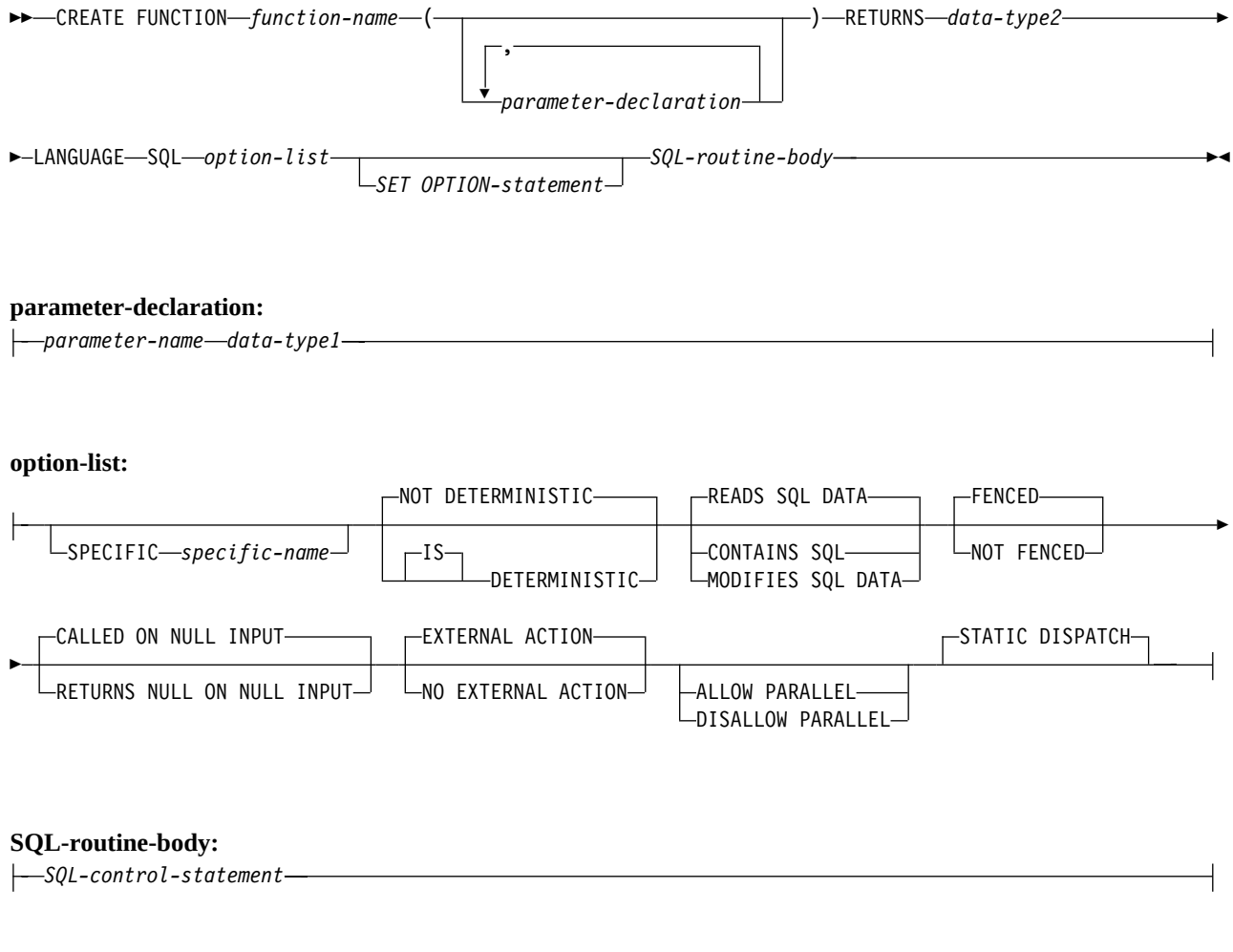
- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

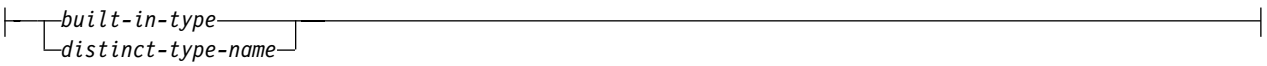
- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.

- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

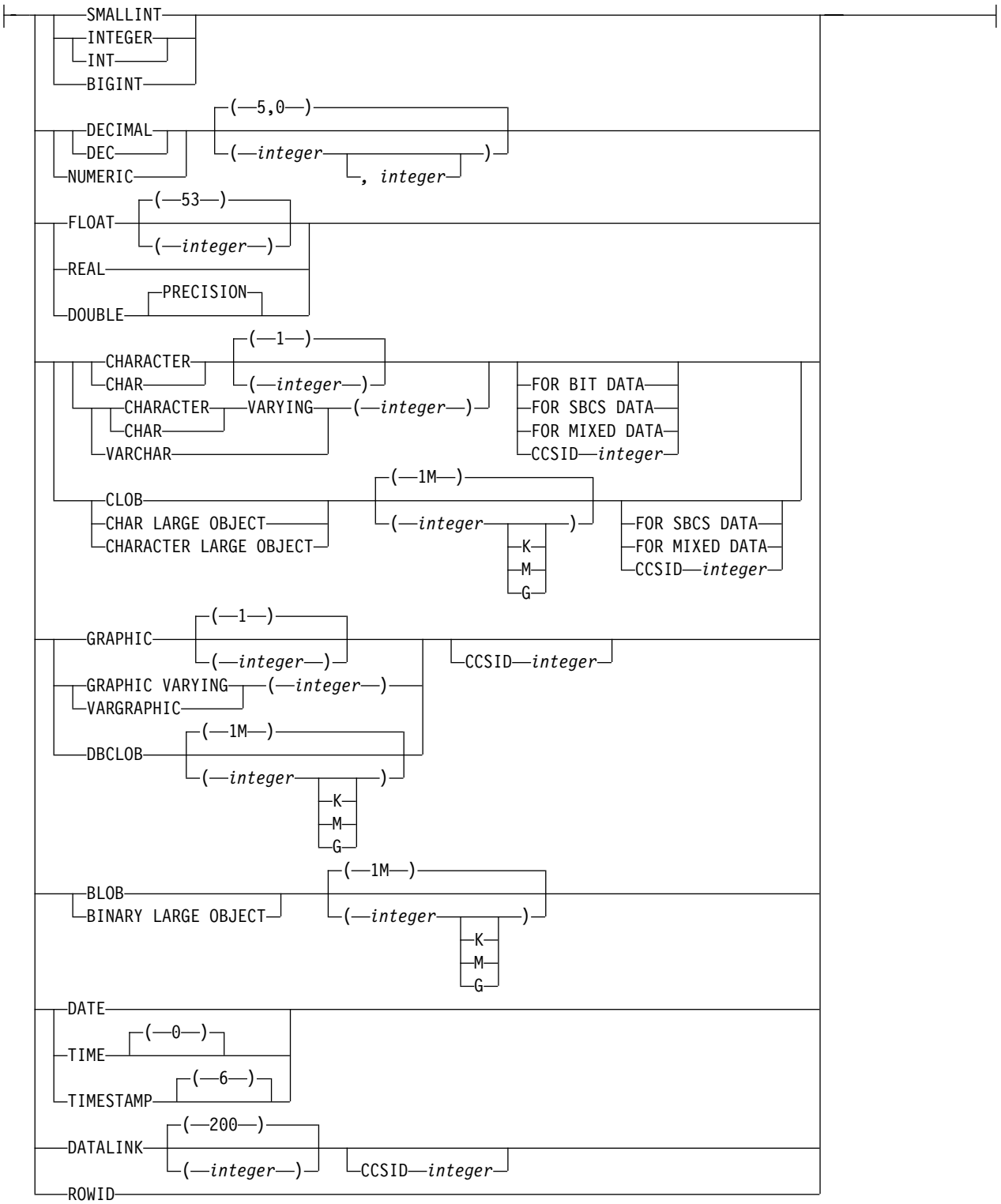
구문



data-type:



built-in-type:



설명

function-name

사용자 정의 함수를 명명합니다. 이름의 조합, 스키마명, 매개변수 수 및 각 매개변수의 자료 유형(자료 유형의 길이, 정밀도, 눈금이나 CCSID 속성에 관계없이)으로 현재 서버에 있는 사용자 정의 함수를 식별해서는 안됩니다.

SQL 명명을 위해 함수는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 함수는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 함수는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면 함수는 QGPL에 작성됩니다.

일반적으로, 각 함수의 함수 표시가 고유하면 하나 이상의 함수가 같은 이름을 가질 수 있습니다.

특정 함수명은 시스템용으로 예약되어 있습니다. 자세한 내용은 443 페이지의 『함수명 선택』을 참조하십시오.

(*parameter-declaration,...)*

함수의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

허용되는 최대 매개변수의 수는 90입니다.

parameter-name

입력 매개변수의 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오. 매개변수 이름은 SQL 함수의 매개변수에 대해 지정되어야 합니다.

data-type1

함수의 입력 매개변수 수와 각 입력 매개변수의 자료 유형을 지정합니다. 함수의 모든 매개변수는 입력 매개변수입니다. 함수가 수신할 것으로 예상하는 각 매개변수 리스트에는 한 항목만 있어야 합니다.

함수는 매개변수를 갖지 않을 수도 있습니다. 이 경우 빈 괄호 세트를 다음과 같이 코드화해야 합니다.

```
CREATE FUNCTION WOOFER ()
```

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 함수가 호출될 때 현재 서버에서 디폴트 CCSID에 의해 CCSID가 판별됩니다.

RETURNS

함수의 출력을 지정합니다.

data-type2

출력의 속성과 자료 유형을 지정합니다.

내장 자료 유형(LONG VARCHAR 또는 LONG VARGRAPHIC 제외)이나 고유한 유형을 지정할 수 있습니다.

CCSID가 지정되고 리턴 자료의 CCSID가 다른 CCSID에 코드화되는 경우 자료는 지정된 CCSID로 변환됩니다.

CCSID가 지정되지 않은 경우 리턴 자료의 CCSID가 다른 CCSID로 코드화되면 리턴 자료는 작업의 CCSID 또는 그래픽 스트링 리턴 값에 대한 작업의 연관된 그래픽 CCSID로 변환됩니다. 변환시 문자가 유실될 가능성을 방지하기 위해서는 함수에서 리턴되는 모든 문자를 나타낼 수 있는 CCSID를 명시적으로 지정하는 것을 고려해보십시오. 이것은 자료 유형이 그래픽 스트링 자료일 때 특히 중요합니다. 이 경우 CCSID 13488(UCS-2 그래픽 스트링 자료)을 사용하는 것을 고려해보십시오.

SPECIFIC *specific-name*

함수에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 함수명의 규정자와 같습니다. 규정된 경우 규정자는 함수명의 규정자와 같아야 합니다.

특정 이름이 지정되지 않으면 함수명으로 설정됩니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

LANGUAGE SQL

이것이 SQL 함수임을 지정합니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

함수가 결정적인지 여부를 지정합니다.

NOT DETERMINISTIC

함수가 동일한 입력 인수로 연속적인 함수 호출로부터 항상 같은 결과를 리턴하지 않음을 지정합니다. NOT DETERMINISTIC은 특수 레지스터에 대한 참조를 포함한 함수 또는 비 판별적 함수일 때 지정되어야 합니다.

DETERMINISTIC

함수가 동일한 입력 인수로 연속적인 호출로부터 항상 같은 결과를 리턴함을 지정합니다.

CONTAINS SQL, READS SQL DATA 또는 **MODIFIES SQL DATA**

함수가 SQL문을 지정할 수 있는지, 그런 경우 어떤 유형인지 지정합니다. 데이터베이스 관리자는 함수에서 실행된 SQL이 이 스펙과 일치하는지 확인합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL
함수는 자료를 읽고 수정하는 SQL문을 실행하지 않습니다.

READS SQL DATA
함수는 자료를 수정하는 SQL문을 실행하지 않습니다.

MODIFIES SQL DATA
함수는 함수에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있습니다.

FENCED 또는 **NOT FENCED**
함수가 호출 SQL문과 동일한 스레드에서 실행되는지 별도의 스레드에서 실행되는지 여부를 지정합니다.

FENCED
함수는 별도의 스레드에서 실행될 것입니다. 함수에 SQL 커서가 들어 있는 경우 FENCED는 가장 안전한 옵션입니다. 동일한 SQL문에서 동일한 함수를 여러 번 호출할 경우 서로 충돌할 수 있기 때문입니다.

NOT FENCED
함수는 호출 SQL문과 동일한 스레드에서 실행됩니다. NOT FENCED 함수는 함수로의 각 호출에 대해 SQL 커서를 열어둡니다. 커서는 계속 열려 있으므로, 함수 호출 사이에 커서 위치 또한 보존됩니다.

NULL INPUT
입력 매개변수가 NULL일 때 함수가 호출될 필요가 있는지를 지정합니다.

CALLED ON NULL INPUT
항상 함수를 호출합니다.

RETURNS NULL ON NULL INPUT
널값이 전달되어 함수의 출력이 NULL 값이 되면 함수는 호출될 필요가 없습니다.

EXTERNAL ACTION 또는 **NO EXTERNAL ACTION**
외부 조치가 들어 있는 함수인지 여부를 지정합니다.

EXTERNAL ACTION
함수가 일부 외부 조치(함수 프로그램의 범위밖)를 수행합니다. 따라서 함수는 각각의 연속적인 함수 호출로 호출되어야 합니다. EXTERNAL ACTION은 외부 조치를 하는 다른 함수에 대한 참조를 포함한 함수일 때 지정되어야 합니다.

NO EXTERNAL ACTION
함수는 외부 조치를 수행하지 않습니다. 함수는 각각의 연속적인 함수 호출로 호출될 필요없습니다.

PARALLEL

함수가 병렬로 실행될 수 있는지를 지정합니다.

ALLOW PARALLEL

함수가 병렬로 실행될 수 있는지를 지정합니다.

DISALLOW PARALLEL

함수가 병렬로 실행될 수 없는지를 지정합니다.

다음 절을 둘 이상 지정하는 경우 디폴트는 DISALLOW PARALLEL입니다.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- MODIFIES SQL DATA

그렇지 않으면 ALLOW PARALLEL이 디폴트입니다.

STATIC DISPATCH

함수가 정적으로 디스패치됨은 지정합니다. 모든 함수는 정적으로 디스패치됩니다.

SET OPTION문

함수 작성에 사용될 옵션을 지정합니다. 예를 들어 디버그를 할 수 있는 함수를 작성하려면 다음 명령문이 포함됩니다.

```
SET OPTION DBGVIEW = *STMT
```

자세한 내용은 772 페이지의 『SET OPTION』을 참조하십시오.

옵션 CLOSQLCSR, CNULRQD, DFTRDBCOL, DYNDFTCOL 및 NAMING은 CREATE FUNCTION문에서 허용되지 않습니다.

SQL-routine-body

복합 명령문을 포함하여 단일 SQL문을 지정합니다. SQL 함수 정의에 대한 자세한 내용은 821 페이지의 제 6 장 『SQL 제어문』을 참조하십시오.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK 및 SET TRANSACTION문을 발행하는 프로시듀어에 대한 호출은 함수에서 허용되지 않습니다.

*SQL-routine-body*가 복합 명령문인 경우, 정확히 하나의 RETURN문을 포함해야 하며 함수가 호출될 때 RETURN문이 실행되어야 합니다.

주

함수 소유권: SQL명이 지정된 경우, 함수의 *소유자*는 함수가 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 함수의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 함수의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

함수 권한: SQL명이 사용되면 함수는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 함수는 *PUBLIC에 대한 권한으로 작성됩니다.

함수의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 함수에 대한 권한도 갖습니다.

함수 작성

SQL 함수가 작성되면 데이터베이스 관리자는 삽입 SQL문과 함께 C 소스 코드가 들어 있는 임시 소스 파일을 작성합니다. *SRVPGM 오브젝트는 CRTSRVPGM 명령을 사용하여 작성됩니다. 서비스 프로그램을 작성하는 데 사용된 SQL 옵션은 CREATE FUNCTION문이 실행될 때 유효한 옵션입니다. 서비스 프로그램은 ACTGRP(*CALLER)를 사용하여 작성됩니다.

특정 이름은 *SRVPGM 오브젝트와 소스 파일 멤버의 이름을 판별하는 데 사용됩니다. 특정 이름이 유효한 시스템명이면 프로그램이 멤버 이름으로 사용됩니다. 멤버가 이미 존재하는 경우 중복됩니다. 프로그램이 지정된 라이브러리에 이미 있으면, 시스템 표 이름을 생성하는 규칙을 사용하여 고유 이름이 생성됩니다. 지정된 이름이 유효한 시스템명이 아니면, 시스템 표 이름을 생성하는 규칙을 사용하여 고유 이름이 생성됩니다.

함수의 속성은 연관된 서비스 프로그램 오브젝트에 저장됩니다. *SRVPGM 오브젝트가 저장된 후 이 시스템이나 다른 시스템으로 복원되면 카탈로그는 그 속성과 함께 자동으로 갱신됩니다.

함수를 복원하는 중에는 다음에 유의하십시오.

- 함수가 처음 작성될 때 특정 이름이 지정되었으나 고유하지 않으면 오류가 발행됩니다.
- 특정 이름이 지정되지 않았으면 필요할 때 고유 이름이 생성됩니다.
- 표시가 고유하지 않으면 함수는 등록될 수 없으며 오류가 발행됩니다.

식별자 분석

루틴 본문에 지정된 표가 있으면, SQL 루틴이 작성될 때 특별한 열, SQL 매개변수 또는 SQL 변수를 식별하기 위해 SQL 루틴의 루틴 본문에 있는 모든 참조가 해석됩니다. 표가 없으면, 함수가 작성될 때 변수나 매개변수를 식별하기 위해 SQL 변수나 매개변수로 존재하는 모든 이름이 해석됩니다. 나머지 이름은 함수가 호출될 때 표에 바인드되는 열로 가정됩니다.

중복 이름이 열, SQL 변수 및 매개변수에 대해 사용되면 열의 표 지정자, 매개변수의 함수명 및 SQL 변수에 대한 레이블 이름을 사용하여 중복 이름을 규정합니다.

함수 호출

SQL 함수가 호출될 때 호출하는 프로그램의 활성 그룹에서 실행됩니다.

함수가 select문의 선택 리스트에 지정되고 이 함수가 EXTERNAL ACTION 또는 MODIFIES SQL DATA를 지정하는 경우 함수는 리턴된 각 행에 대해서만 호출될 수 있습니다. 그렇지 않으면 선택되지 않은 행에 대하여 UDF가 호출될 수 있습니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.

예 1

SQL 함수 NTEST1을 작성하여 규칙을 구현합니다.

```
output = 2 * input - 4
```

```
CREATE FUNCTION NTEST1 (p_input INTEGER)
  RETURNS INTEGER
  LANGUAGE SQL
  SPECIFIC MINENULL1
  func1_lab:
    BEGIN
      DECLARE p_output INT;
      IF p_input IS NULL THEN
        SET p_output = NULL;
      ELSE
        SET p_output = 2 * p_input - 4;
     END IF;
     RETURN p_output;
    END
```

CREATE FUNCTION(SQL 표)

현재 서버에서 CREATE FUNCTION(SQL 표)문은 SQL 표 함수를 작성합니다. 함수는 단일 결과 표를 리턴합니다.

호출

이 명령문을 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSFUNCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 서비스 프로그램 작성(CRTSRVPGM) 명령에 대해 *USE
  - 함수가 작성되는 라이브러리에 대해 *EXECUTE 및 *ADD
- 관리 권한

고유한 유형이 참조된다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

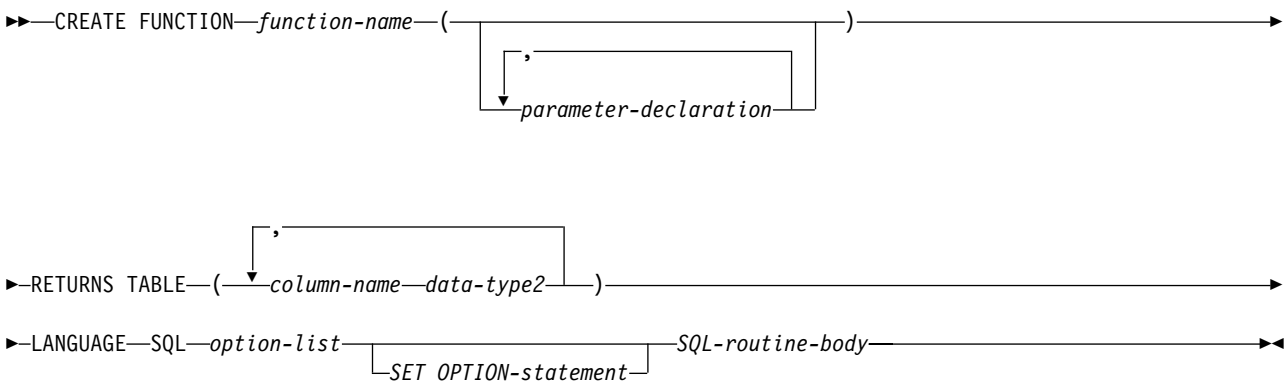
- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.

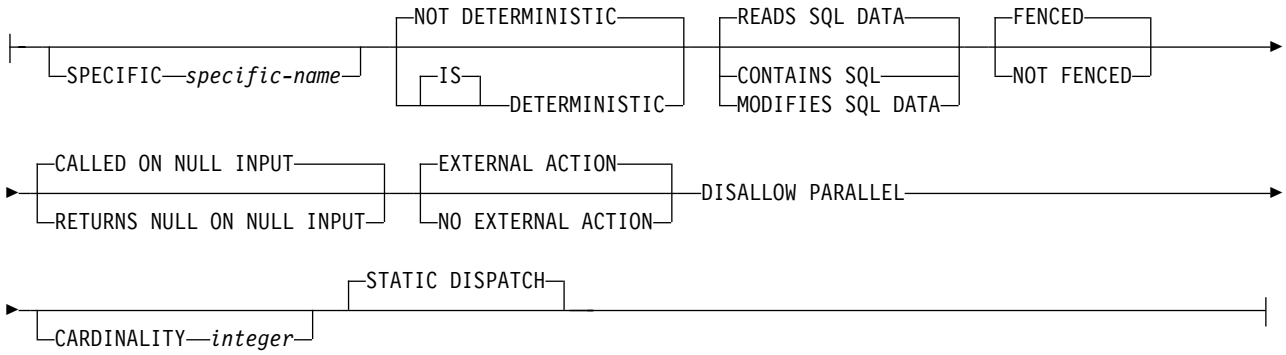
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문

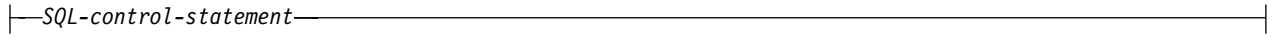


parameter-declaration:

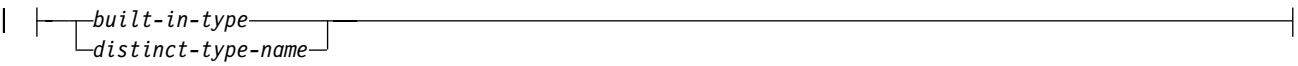
option-list:



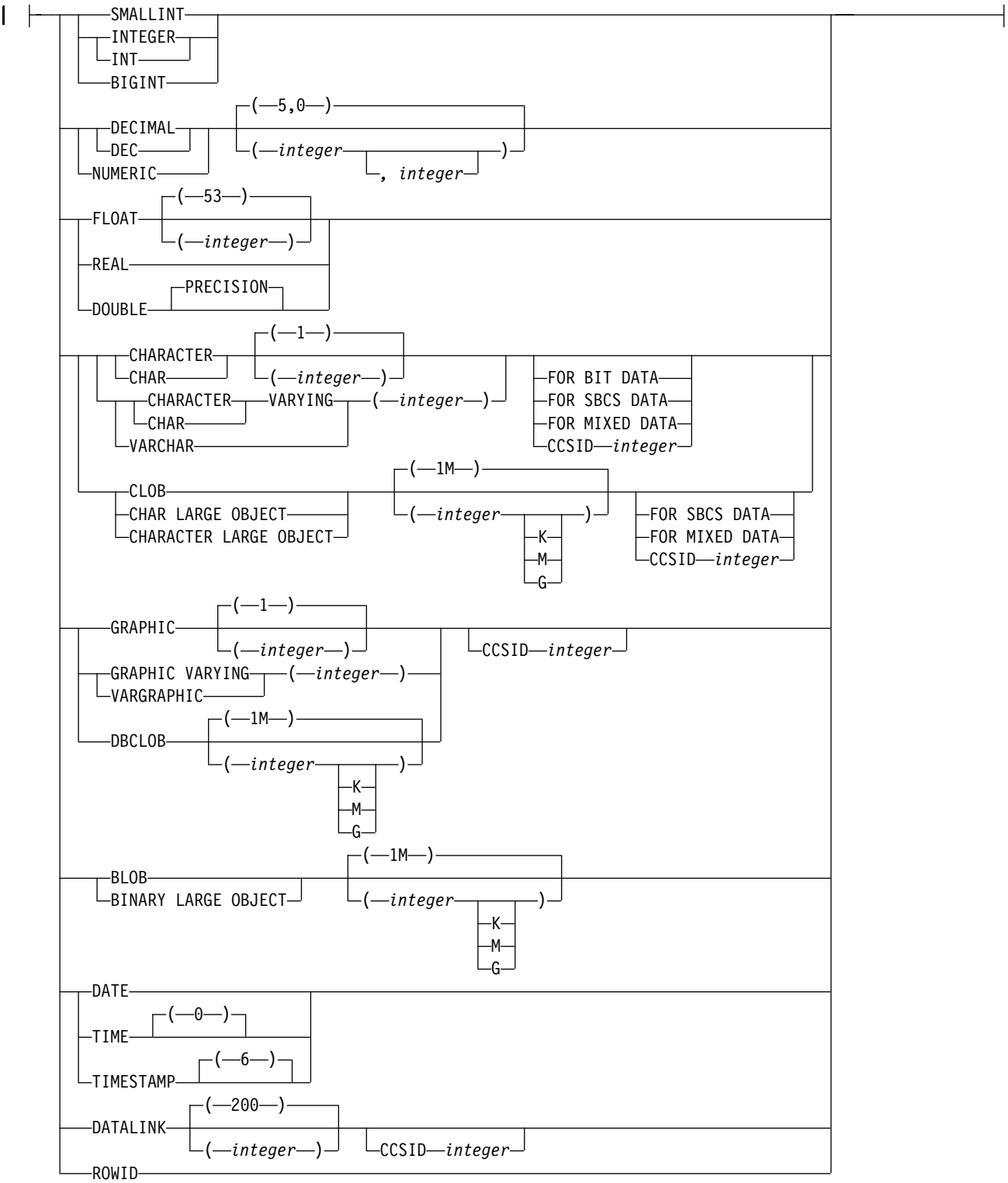
SQL-routine-body:



data-type:



built-in-type:



설명

function-name

사용자 정의 함수를 명명합니다. 이름의 조합, 스키마명, 매개변수 수 및 각 매개변수의 자료 유형(자료 유형의 길이, 정밀도, 눈금이나 CCSID 속성에 관계없이)으로 현재 서버에 있는 사용자 정의 함수를 식별해서는 안됩니다.

SQL 명명을 위해 함수는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 함수는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 함수는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면 함수는 QGPL에 작성됩니다.

일반적으로, 각 함수의 함수 표시가 고유하면 하나 이상의 함수가 같은 이름을 가질 수 있습니다.

특정 함수명은 시스템용으로 예약되어 있습니다. 자세한 내용은 443 페이지의 『함수명 선택』을 참조하십시오.

(*parameter-declaration,...)*

함수의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

매개변수의 최대 수는 90으로 허용됩니다.

parameter-name

입력 매개변수의 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오. 매개변수 이름은 SQL 함수의 매개변수에 대해 지정되어야 합니다.

data-type1

함수의 입력 매개변수 수와 각 입력 매개변수의 자료 유형을 지정합니다. 함수의 모든 매개변수는 입력 매개변수입니다. 함수가 수신할 것으로 예상하는 각 매개변수 리스트에는 한 항목만 있어야 합니다.

함수는 매개변수를 갖지 않을 수도 있습니다. 이 경우 빈 괄호 세트를 다음과 같이 코드화해야 합니다.

```
CREATE FUNCTION WOOFER ()
```

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 함수가 호출될 때 현재 서버에서 CCSID에 의해 CCSID가 판별됩니다.

RETURNS TABLE

함수의 출력 표를 지정합니다.

매개변수 수는 N이라고 할 때, (247-(N*2))/2개 이하의 열이 와야 합니다.

열 이름

출력 표의 열 이름을 지정합니다. 한 번 이상 같은 이름을 지정하지 마십시오.

data-type2

출력의 속성과 자료 유형을 지정합니다.

내장 자료 유형(LONG VARCHAR 또는 LONG VARGRAPHIC 제외)이나 고유한 유형을 지정할 수 있습니다.

CCSID가 지정되고 리턴 자료의 CCSID가 다른 CCSID에 코드화되는 경우 자료는 지정된 CCSID로 변환됩니다.

CCSID가 지정되지 않은 경우 리턴 자료의 CCSID가 다른 CCSID로 코드화되면 리턴 자료는 작업의 CCSID 또는 그래픽 스트링 리턴 값에 대한 작업의 연관된 그래픽 CCSID로 변환됩니다. 변환시 문자가 유실될 가능성을 방지하기 위해서는 함수에서 리턴되는 모든 문자를 나타낼 수 있는 CCSID를 명시적으로 지정하는 것을 고려해보십시오. 이것은 자료 유형이 그래픽 스트링 자료일 때 특히 중요합니다. 이 경우 CCSID 13488(UCS-2 그래픽 스트링 자료)을 사용하는 것을 고려해보십시오.

SPECIFIC *specific-name*

함수에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 함수명의 규정자와 같습니다. 규정된 경우 규정자는 함수명의 규정자와 같아야 합니다.

특정 이름이 지정되지 않으면 함수명으로 설정됩니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

LANGUAGE SQL

이것이 SQL 함수임을 지정합니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

함수가 결정적인지 여부를 지정합니다.

NOT DETERMINISTIC

함수가 동일한 입력 인수로 연속적인 함수 호출로부터 항상 같은 결과를 리턴하지 않음을 지정합니다. NOT DETERMINISTIC은 특수 레지스터에 대한 참조를 포함한 함수 또는 비 판별적 함수 일때 지정되어야 합니다.

DETERMINISTIC

함수가 동일한 입력 인수로 연속적인 호출로부터 항상 같은 결과를 리턴함을 지정합니다.

CONTAINS SQL, READS SQL DATA 또는 **MODIFIES SQL DATA**

함수가 SQL문을 지정할 수 있는지, 그런 경우 어떤 유형인지 지정합니다. 데이터베이스 관리자는 함수에서 실행된 SQL이 이 스펙과 일치하는지 확인합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL

함수는 자료 읽기 또는 수정하는 실행 SQL문이 아닙니다.

READS SQL DATA

함수는 자료를 수정하는 실행 SQL문이 아닙니다.

MODIFIES SQL DATA

함수는 함수에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있습니다.

FENCED 또는 **NOT FENCED**

함수가 호출 SQL문과 동일한 스레드에서 실행되는지 별도의 스레드에서 실행되는지 여부를 지정합니다.

FENCED

함수는 분리 스레드에서 실행될 것입니다. 함수에 SQL 커서가 들어 있는 경우 FENCED는 가장 안전한 옵션입니다. 동일한 SQL문에서 동일한 함수를 여러 번 호출할 경우 서로 충돌할 수 있기 때문입니다.

NOT FENCED

함수가 호출 SQL문과 동일한 스레드에서 실행됩니다. NOT FENCED 함수는 함수로의 각 호출에 대해 SQL 커서를 열어둡니다. 커서는 계속 열려 있으므로, 함수 호출 사이에 커서 위치 또한 보존됩니다.

NULL INPUT

입력 매개변수가 NULL일 때 함수가 호출될 필요가 있는지를 지정합니다.

CALLED ON NULL INPUT

항상 함수를 호출합니다.

RETURNS NULL ON NULL INPUT

표 함수 OPEN시, 함수의 인수 중 하나가 널인 경우 사용자 정의 표 함수는 호출되지 않으며 함수 출력은 빈 표(행 없는 표)가 됩니다.

EXTERNAL ACTION 또는 **NO EXTERNAL ACTION**

외부 조치가 들어 있는 함수인지 여부를 지정합니다.

EXTERNAL ACTION

함수가 일부 외부 조치(함수 프로그램의 범위밖)를 수행합니다. 따라서 함수는

각각의 연속적인 함수 호출로 호출되어야 합니다. EXTERNAL ACTION은 외부 조치를 하는 다른 함수에 대한 참조를 포함한 함수일 때 지정되어야 합니다.

NO EXTERNAL ACTION
함수는 외부 조치를 수행하지 않습니다. 함수는 각각의 연속적인 함수 호출로 호출될 필요없습니다.

DISALLOW PARALLEL
함수가 병렬로 실행될 수 없는지를 지정합니다. 표 함수는 병렬로 실행될 수 없습니다.

CARDINALITY *integer*
이 선택적 절은 최적화를 위해 함수가 리턴할 예상 행 수에 대한 예상치를 제공합니다. 정수의 유효 값 범위는 0 - 2 147 483 647입니다.

CARDINALITY절이 표 함수에 대해 지정되지 않은 경우, 데이터베이스 관리자는 유한 값을 디폴트로 가정합니다.

STATIC DISPATCH
함수가 정적으로 디스패치됨은 지정합니다. 모든 함수는 정적으로 디스패치됩니다.

SET OPTION문
함수 작성에 사용될 옵션을 지정합니다. 예를 들어 디버그를 할 수 있는 함수를 작성하려면 다음 명령문이 포함됩니다.

```
SET OPTION DBGVIEW = *STMT
```

자세한 내용은 772 페이지의 『SET OPTION』을 참조하십시오.

옵션 CLOSQLCSR, CNULRQD, DFTRDBCOL, DYNDFTCOL 및 NAMING은 CREATE FUNCTION문에서 허용되지 않습니다.

SQL-routine-body
복합 명령문을 포함하여 단일 SQL문을 지정합니다. SQL 함수 정의에 대한 자세한 내용은 821 페이지의 제 6 장 『SQL 제어문』을 참조하십시오.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK 및 SET TRANSACTION문을 발행하는 프로시듀어에 대한 호출은 함수에서 허용되지 않습니다.

*SQL-routine-body*가 복합 명령문인 경우, 정확히 하나의 RETURN문을 포함해야 하며 함수가 호출될 때 실행되어야 합니다.

주

함수 소유권: SQL명이 지정된 경우, 함수의 *소유자*는 함수가 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 함수의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 함수의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

함수 권한: SQL명이 사용되면 함수는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 함수는 *PUBLIC에 대한 권한으로 작성됩니다.

함수의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 함수에 대한 권한도 갖습니다.

함수 작성

SQL 함수가 작성되면 데이터베이스 관리자는 삽입 SQL문과 함께 C 소스 코드가 들어 있는 임시 소스 파일을 작성합니다. *SRVPGM 오브젝트는 CRTSRVPGM 명령을 사용하여 작성됩니다. 서비스 프로그램을 작성하는 데 사용된 SQL 옵션은 CREATE FUNCTION문이 실행될 때 유효한 옵션입니다. 서비스 프로그램은 ACTGRP(*CALLER)를 사용하여 작성됩니다.

특정 이름은 *SRVPGM 오브젝트와 소스 파일 멤버의 이름을 판별하는 데 사용됩니다. 특정 이름이 유효한 시스템명이면 프로그램이 멤버 이름으로 사용됩니다. 멤버가 이미 존재하는 경우 중복됩니다. 프로그램이 지정된 라이브러리에 이미 있으면, 시스템 표 이름을 생성하는 규칙을 사용하여 고유 이름이 생성됩니다. 지정된 이름이 유효한 시스템명이 아니면, 시스템 표 이름을 생성하는 규칙을 사용하여 고유 이름이 생성됩니다.

함수의 속성은 연관된 서비스 프로그램 오브젝트에 저장됩니다. *SRVPGM 오브젝트가 저장된 후 이 시스템이나 다른 시스템으로 복원되면 카탈로그는 그 속성과 함께 자동으로 갱신됩니다.

함수를 복원하는 중에는 다음에 유의하십시오.

- 함수가 처음 작성될 때 특정 이름이 지정되었으나 고유하지 않으면 오류가 발행됩니다.
- 특정 이름이 지정되지 않았으면 필요할 때 고유 이름이 생성됩니다.
- 표시가 고유하지 않으면 함수는 등록될 수 없으며 오류가 발행됩니다.

식별자 분석

루틴 본문에 지정된 표가 있으면, SQL 루틴이 작성될 때 특별한 열, SQL 매개변수 또는 SQL 변수를 식별하기 위해 SQL 루틴의 루틴 본문에 있는 모든 참조가 해석됩니다. 표가 없으면, 함수가 작성될 때 변수나 매개변수를 식별하기 위해 SQL 변수나 매개변수로 존재하는 모든 이름이 해석됩니다. 나머지 이름은 함수가 호출될 때 표에 바인드되는 열로 가정됩니다.

중복 이름이 열, SQL 변수 및 매개변수에 대해 사용되면 열의 표 지정자, 매개변수의 함수명 및 SQL 변수에 대한 레이블 이름을 사용하여 중복 이름을 규정합니다.

함수 호출

SQL 함수가 호출될 때 호출하는 프로그램의 활성 그룹에서 실행됩니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.

예

지정된 부서 번호의 직원을 리턴하는 표 함수를 정의합니다.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  DISALLOW PARALLEL
  RETURN
      SELECT EMPNO,LASTNAME,FIRSTNME
                FROM EMPLOYEE
        WHERE EMPLOYEE.WORKDEPT =DEPTEMPLOYEES.DEPTNO
```

CREATE INDEX

현재 서버에서 CREATE INDEX문은 표에 색인을 작성합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 논리 파일 작성(CRTLF) 명령에 대한 *USE
  - 색인이 작성되는 라이브러리에 대한 *EXECUTE 및 *ADD
  - 색인이 작성되는 라이브러리가 자료 사전을 가지고 있는 SQL 스키마인 경우 자료 사전에 대한 *CHANGE
- 관리 권한

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 표에서 INDEX 권한
- 관리 권한

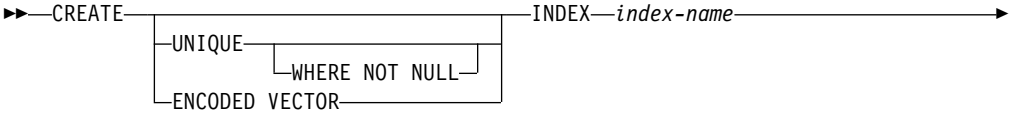
명령문의 권한부여 ID는 다음 경우에 표에 대한 INDEX 권한을 갖습니다.

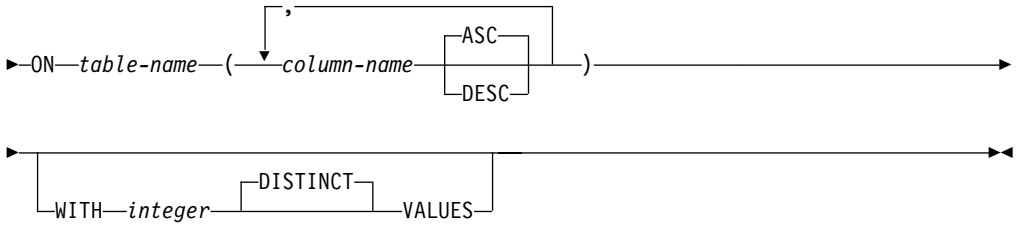
- 표의 소유자인 경우
- 표에서 INDEX 또는 ALTER 권한을 부여받았습니다.
- 표에서 *OBJALTER 또는 *OBJMGT의 시스템 권한을 부여받았습니다.

SQL 이름이 지정되고, 표가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며, 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

구문





설명

UNIQUE

표가 동일한 색인 키 값을 갖는 두 개 이상의 행을 포함하지 않도록 합니다. 제한사항은 표의 행이 갱신되거나 새로운 행이 삽입될 때 강제됩니다.

제한사항은 CREATE INDEX문이 실행되는 동안에도 검사됩니다. 표에 이미 중복 키 값을 갖는 행이 있으면 색인은 작성되지 않습니다.

UNIQUE가 사용되면 널값은 다른 값처럼 처리됩니다. 예를 들어, 키가 널값을 가질 수 있는 단일 열이면, 그 열에는 하나의 널값만 있을 수 있습니다.

UNIQUE WHERE NOT NULL

표가 색인 키와 동일한 널이 아닌 값을 갖는 두 개 이상의 행을 포함하지 않도록 합니다. 여러 개의 널값이 허용됩니다. 그렇지 않으면 UNIQUE와 동일합니다.

ENCODED VECTOR

결과 색인이 인코드된 벡터 색인(EVI)임을 지정표시합니다.

인코드된 벡터 색인은 행의 순서를 확인하는 데 사용될 수 없습니다. 조회의 성능을 개선하기 위해 데이터베이스 관리자에 의해 사용됩니다. 자세한 내용은 Database Performance and Query Optimization 책을 참조하십시오.

index-name

색인을 명명합니다. 내재적 또는 명시적 규정자를 포함하여 이름은 이미 현재 서버에 있는 색인, 표, 뷰, 별명 또는 파일과 같을 수 없습니다.

SQL명이 지정되면 색인은 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템명이 지정되면 색인명은 규정자에 의해 지정된 스키마에 작성됩니다. 규정되지 않은 경우 색인 이름은 색인이 작성되는 표와 같은 콜렉션이나 라이브러리에 작성됩니다.

색인 이름이 유효한 시스템명이 아니면 iSeries용 DB2 UDB는 시스템명을 생성합니다. 이름 생성 규칙에 대한 내용은 572 페이지의 『표 이름 생성 규칙』을 참조하십시오.

ON *table-name*

색인이 작성될 표를 식별합니다. *table-name*은 현재 서버에 있는 기본 표(뷰가 아닌)를 식별해야 합니다.

(column-name, ...)

색인 키의 일부가 될 열의 리스트를 식별합니다.

각 *column-name*은 표의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 하나 이상 지정될 수 있습니다. *column-name*은 LOB나 자료 링크 열에 근거하여 LOB 또는 DATALINK 열 또는 고유한 유형을 식별해서는 안됩니다. 열의 수는 120을 초과해서는 안되고 바이트 길이의 합계는 2000-n을 초과해서는 안됩니다. 여기서, n은 널(null)을 허용하는 지정된 열의 수입니다.

ASC

색인 항목을 열에 의한 오름차순으로 저장합니다. 디폴트입니다.

DESC

색인 항목을 열에 의한 내림차순으로 저장합니다.

WITH *integer* **DISTINCT VALUES**

고유 키 값의 예상 수를 지정합니다. 이 절은 모든 유형의 색인에 대해 지정될 수 있습니다.

인코드된 벡터 색인의 경우 각 고유 키 값에 지정된 코드의 초기 크기를 판별하는 데 사용됩니다. 디폴트 값은 256입니다.

인코드되지 않은 벡터 색인의 경우 최적자에 대한 힌트로 사용됩니다.

주

명명된 표에 이미 자료가 있으면, CREATE INDEX는 이에 대한 색인 항목을 작성합니다. 표에 아직 자료가 없으면 CREATE INDEX는 색인의 설명을 작성합니다. 자료가 표에 삽입될 때 색인 항목이 작성됩니다. 색인은 항상 표의 현재 상태를 반영합니다.

정렬 순서: SBCS나 혼합된 자료가 들어 있는 열에 대해 작성된 색인은 명령문이 실행될 때 유효한 정렬 순서로 작성됩니다. *HEX 이외의 정렬 순서의 경우에는 SBCS 자료나 혼합된 자료에 대한 키가 정렬 순서를 기초로 하는 키의 가중치 값입니다.

색인 속성: 색인은 키순 논리 파일로 작성됩니다. 색인이 작성되면 파일 대기 시간과 레코드 대기 시간 속성은 논리 파일 작성(CRTLF) 명령의 WAITFILE 및 WAITRCD 키워드에 지정된 디폴트 값으로 설정됩니다.

분배된 표에 대해 작성된 색인은 표가 분배되는 서버 전체에서 작성됩니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

함수 소유권: SQL명이 지정된 경우, 인덱스의 *소유자*는 인덱스가 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 색인의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 색인의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

인덱스 권한: SQL 이름이 사용되면 인덱스는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템 이름이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 색인은 *PUBLIC에 대한 권한으로 작성됩니다.

색인의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 색인에 대한 권한도 갖습니다.

예

예 1

PROJECT 표에 UNIQUE_NAM이라는 색인을 작성합니다. 색인의 목적은 표에 프로젝트 이름(PROJNAME)에 대해 같은 값을 갖는 두 개의 항목이 없다는 것을 확인하기 위한 것입니다. 색인 항목은 오름차순으로 정렬됩니다.

```
CREATE UNIQUE INDEX UNIQUE_NAME
  ON PROJECT(PROJNAME)
```

예 2

EMPLOYEE 표에 JOB_BY_DPT라는 색인을 작성합니다. 각 부서(WORKDEPT) 내에서 색인 항목을 작업 제목(JOB)에 의해 오름차순으로 정렬합니다.

```
CREATE INDEX JOB_BY_DPT
  ON EMPLOYEE (WORKDEPT, JOB)
```

CREATE PROCEDURE

CREATE PROCEDURE문은 현재 서버에서 프로시듀어를 정의합니다.

다음 프로시듀어 유형을 정의할 수 있습니다.

- 외부

  프로시듀어 프로그램은 C, COBOL 또는 Java와 같은 프로그래밍 언어로 작성됩니다. 외부 프로그램은 프로시듀어의 여러 속성과 함께 현재 서버에 정의된 프로시듀어에서 참조합니다. 512 페이지의 『CREATE PROCEDURE(외부)』를 참조하십시오.

- SQL

  프로시듀어는 SQL으로만 작성됩니다. 프로시듀어의 본문은 프로시듀어의 여러 속성과 함께 현재 서버에 정의됩니다. 526 페이지의 『CREATE PROCEDURE(SQL)』를 참조하십시오.

주

매개변수의 자료 유형 선택

iSeries용 DB2 UDB가 아닌 플랫폼에서의 프로시듀어 이식을 위해 다른 플랫폼에서는 다르게 표시되는 다음 자료 유형은 사용하지 않습니다.

- FLOAT. 대신 DOUBLE이나 REAL을 사용합니다.
- NUMERIC. 대신 DECIMAL을 사용합니다.

매개변수의 AS LOCATOR 지정

값 대신 로케이터를 전달하면 프로시듀어 내외에서 더 적은 수의 바이트가 전달됩니다. 매개변수 값이 매우 큰 경우 이 기능은 유용할 수 있습니다. AS LOCATOR절은 실제 값 대신 매개변수 값으로의 로케이터가 전달되도록 지정합니다. LOB 자료 유형을 기초로 한 고유한 유형이나 LOB 자료 유형을 가진 매개변수에만 AS LOCATOR를 지정할 수 있습니다.

AS LOCATOR는 SQL 프로시듀어에 대해 지정될 수 없습니다.

스키마에서 프로시듀어 고유성 판별

현재 서버에서 각 프로시듀어의 표시는 고유해야 합니다. 프로시듀어의 표시는 매개변수의 자료 유형과 번호를 결합한 규정된 프로시듀어명입니다(매개변수의 자료 유형은 프로시듀어의 표시 부분이 아닙니다). 이는 두 개의 다른 스키마가 각각, 같은 수의 매개변수를 갖는 동일한 이름의 프로시듀어를 포함할 수 있다는 것을 의미합니다. 그러나 하나의 스키마에 동일한 수의 매개변수를 갖는 동일한 이름의 두 프로시듀어가 있을 수는 없습니다.

프로시듀어의 고유명

동일함 이름과 스키마를 가진 여러 프로시듀어를 정의할 경우(서로 다른 수의 매개변수를 가진), 고유명도 지정하는 것이 좋습니다. 고유명을 사용하면 프로시듀어를 삭제하고, 권한을 부여하고, 권한을 철회하고, 주석을 붙일 때 함수를 해당 프로시듀어를 고유하게 식별할 수 있습니다.

SPECIFIC절이 지정되지 않으면, 고유명이 생성됩니다.

CREATE PROCEDURE(외부)

CREATE PROCEDURE(외부)문은 현재 서버의 외부 프로시듀어를 정의합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSPROCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

외부 프로그램이 존재하면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SQL문에서 참조되는 외부 프로그램의 경우
  - 외부 프로그램에서 시스템 권한 *EXECUTE
  - 외부 프로그램이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

고유한 유형이 참조된다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

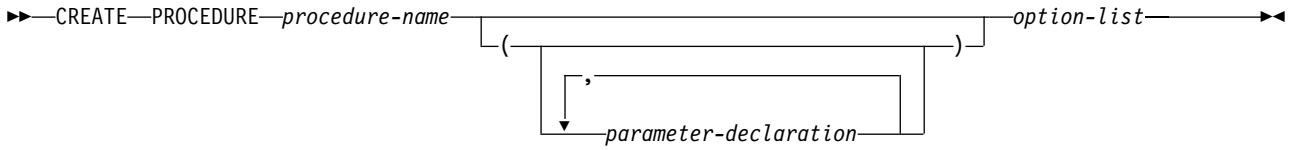
- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

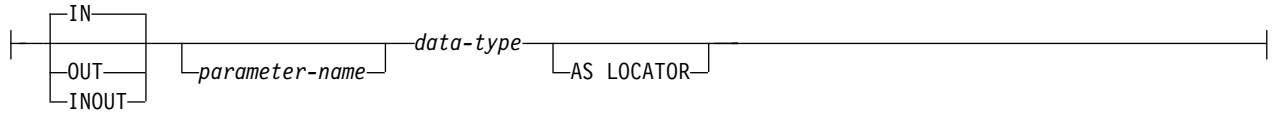
- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.

- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문

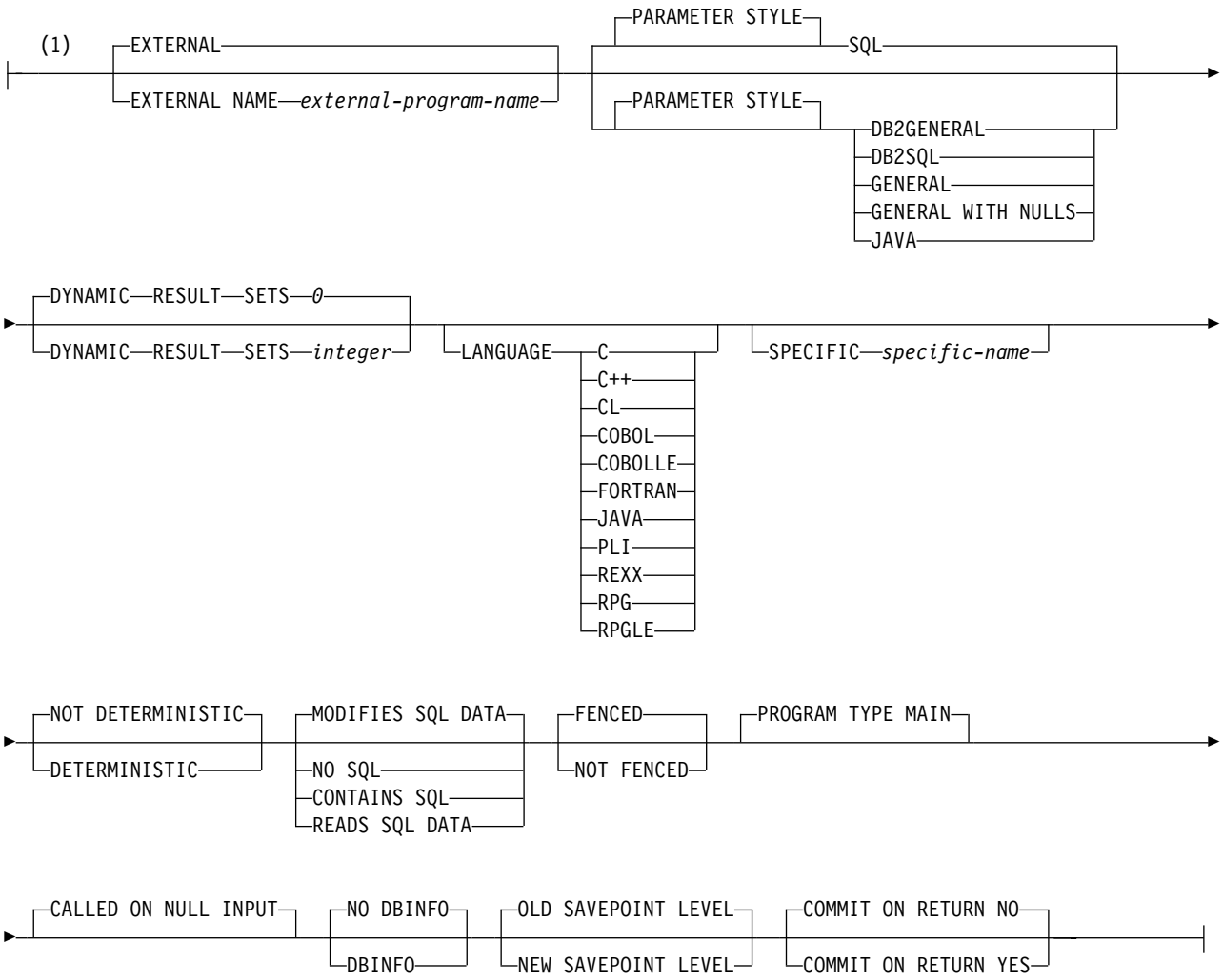


parameter-declaration:



CREATE PROCEDURE(외부)

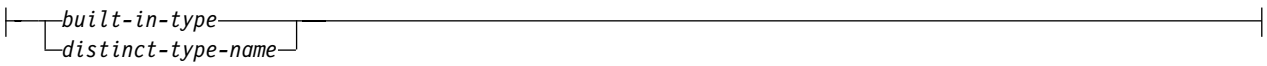
option-list:



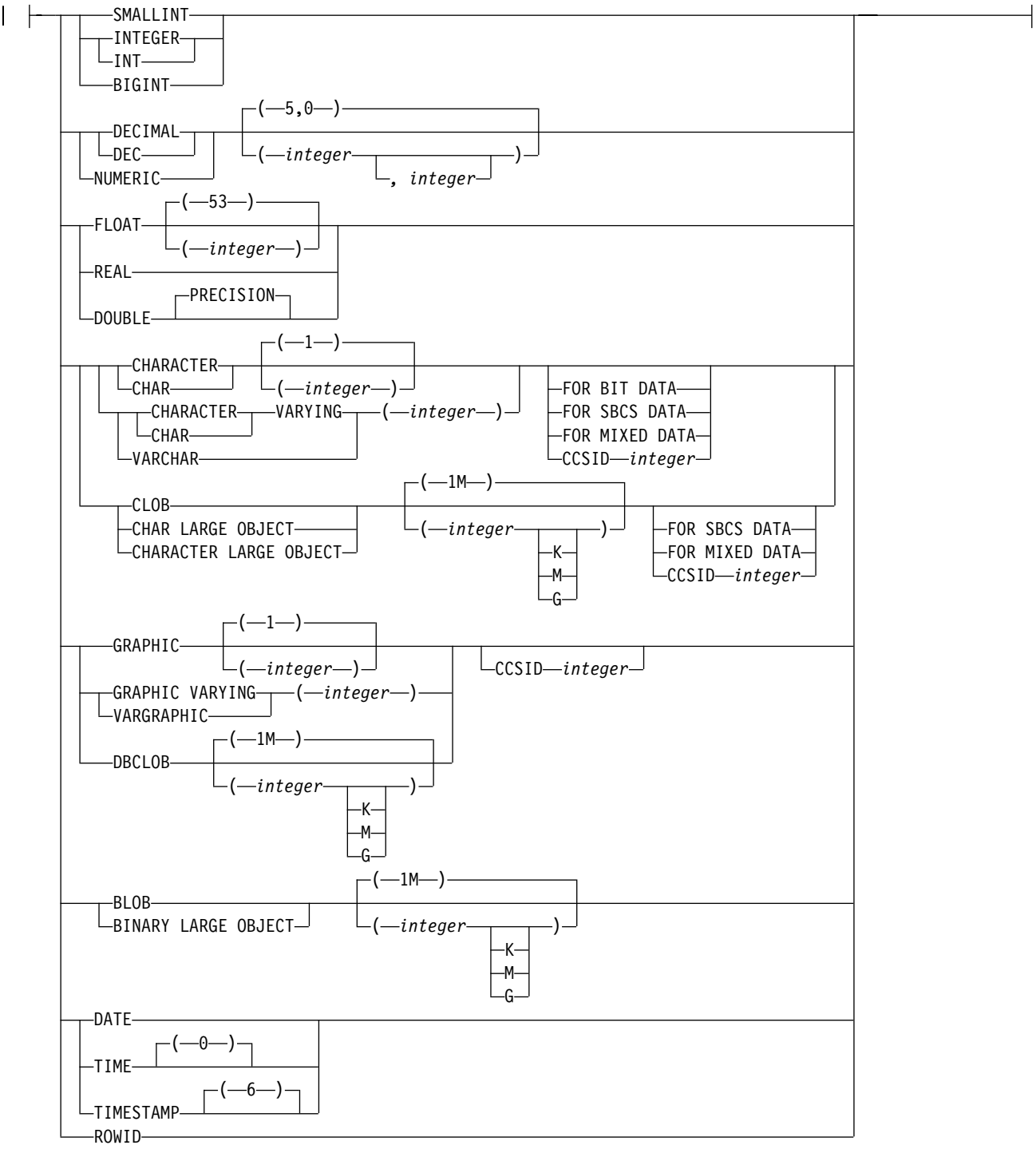
주:

1 옵션 절은 다른 순서로 지정할 수 있습니다.

data-type:



built-in-type:



설명

procedure-name

프로시듀어를 명명합니다. 이름의 조합, 콜렉션 이름, 매개변수 수는 현재 서버에 있는 프로시듀어를 식별해서는 안됩니다.

SQL 명명을 위해 프로시듀어는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 프로시듀어는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 프로시듀어는 현재 라이브러리(*CURLIB)에 작성됩니다.

(*parameter-declaration,...)*

프로시듀어의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 프로시듀어의 매개변수는 입력, 출력, 입출력 모두에 사용할 수 있습니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

CREATE PROCEDURE에 사용할 수 있는 매개변수의 최대수는 255입니다. GENERAL WITH NULLS가 지정되면 최대값은 254입니다. 매개변수 양식 SQL이 지정되면 90개의 매개변수만 허용됩니다. 매개변수의 최대수는 외부 프로그램을 컴파일하기 위해 사용되는 사용권 프로그램이 허용하는 매개변수의 최대수로 제한됩니다.

IN 프로시듀어로의 입력 매개변수로서 매개변수를 지정합니다. 프로시듀어내에서 매개변수의 변경은 제어가 리턴될 때 호출 SQL 어플리케이션에 적용되지 않습니다.[50]

OUT

매개변수를 프로시듀어가 리턴하는 출력 매개변수로 지정합니다.

자료 링크나 자료 링크를 기초로 하는 고유한 유형은 출력 매개변수로 지정되지 않습니다.

INOUT

프로시듀어의 매개변수를 입력 및 출력 매개변수로 지정합니다.

자료 링크나 자료 링크를 기초로 하는 고유한 유형은 입력 및 출력 매개변수로 지정되지 않습니다.

parameter-name

매개변수를 명명합니다. 이름은 프로시듀어에 대한 다른 *parameter-name*과 같아질 수 없습니다.

data-type

매개변수의 자료 유형을 지정합니다.

---

50. 언어 유형이 REXX이면 모든 매개변수는 입력 매개변수이어야 합니다.

자료 유형은 언어 절에 지정된 언어에 대해 유효해야 합니다. 자료 유형은 외부 프로시듀어에 대해 유효하지 않습니다. 자료 유형에 대한 자세한 정보는 541 페이지의『CREATE TABLE』및 SQL 프로그래밍 개념 책을 참조하십시오.

PARAMETER STYLE JAVA가 지정된 경우, 대형 오브젝트(LOB) 자료 유형을 가진 매개변수는 지원되지 않습니다.

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 프로시듀어가 호출될 때 현재 서버에서 디폴트 CCSID에 의해 CCSID가 판별됩니다.

AS LOCATOR

입력 매개변수가 실제 값이 아니라 값에 대한 로케이터임을 지정합니다. 입력 매개변수가 LOB 자료 유형이나 LOB 자료 유형을 기초로 한 고유한 유형을 가질 경우에만 AS LOCATOR를 지정할 수 있습니다.

LANGUAGE

외부 프로그램이 작성되는 언어를 지정합니다. 외부 프로그램이 REXX 프로시듀어이면 언어 절은 필요하지 않습니다.

LANGUAGE가 지정되지 않으면 LANGUAGE는 프로시듀어가 작성될 때 외부 프로그램과 연관된 프로그램 속성 정보에서 판별됩니다. 프로그램과 연관된 프로그램 속성 정보가 인식할 수 있는 언어를 식별하지 못하거나 프로그램을 찾을 수 없으면, 언어는 C로 가정됩니다.

C 외부 프로그램이 C로 작성됩니다.

C++

외부 프로그램이 C++로 작성됩니다.

CL

외부 프로그램이 CL로 작성됩니다.

COBOL

외부 프로그램이 COBOL로 작성됩니다.

COBOLLE

외부 프로그램은 ILE COBOL로 작성됩니다.

FORTRAN

외부 프로그램이 FORTRAN으로 작성됩니다.

JAVA

외부 프로그램이 JAVA로 작성됩니다.

PLI

외부 프로그램이 PL/I로 작성됩니다.

REXX

외부 프로그램은 REXX 프로시듀어입니다.

RPG

외부 프로그램이 RPG로 작성됩니다.

RPGLE

외부 프로그램은 ILE RPG로 작성됩니다.

PARAMETER STYLE

함수로(부터) 값을 전달하고 리턴하는 데 사용되는 규약을 지정합니다.

SQL

CALL문의 매개변수 외에 몇몇 추가 매개변수가 프로시듀어에 전달됨을 지정합니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 CREATE PROCEDURE문에 지정되는 매개변수입니다.
- 매개변수의 인디케이터 변수에 대한 N개의 매개변수
- SQLSTATE에 대한 CHAR(5) 출력 매개변수. 리턴되는 SQLSTATE는 프로시듀어의 성공 또는 실패를 표시합니다. 리턴된 SQLSTATE는 외부 프로그램에 의해 지정됩니다.

  사용자는 외부 프로그램에 모든 유효한 값으로 SQLSTATE를 설정하여 함수으로부터 오류나 경고를 리턴할 수 있습니다.
- 완전 규정된 프로시듀어명에 대한 VARCHAR(517) 입력 매개변수.
- 특정 이름에 대한 VARCHAR(128) 입력 매개변수.
- 메세지 텍스트에 대한 VARCHAR(70) 출력 매개변수.

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

PARAMETER STYLE SQL은 LANGUAGE JAVA와 함께 사용할 수 없습니다.

DB2GENERAL

프로시듀어는 Java 메소드와 함께 사용하도록 정의된 매개변수 전달 규칙을 사용함을 지정합니다.

PARAMETER STYLE DB2GENERAL는 LANGUAGE JAVA와 함께 지정해야 합니다. JAVA에서 매개변수의 전달에 대한 자세한 정보는 'Developer Kit for Java' 책을 참조하십시오.

DB2SQL

CALL문의 매개변수 외에 몇몇 추가 매개변수가 프로시듀어에 전달됨을 지정합니다. DB2SQL은 다음 추가 매개변수가 마지막 매개변수로 전달될 수 있다는 점을 제외하고 SQL 매개변수 스타일과 동일합니다.

- DBINFO가 CREATE PROCEDURE문에 지정된 경우 dbinfo 구조에 대한 매개변수 한 개

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

PARAMETER STYLE DB2SQL은 LANGUAGE JAVA와 함께 사용할 수 없습니다.

GENERAL

프로시듀어가 CALL에 지정된 매개변수를 수신할 때 프로시듀어가 매개변수 전달 메카니즘을 사용함을 지정합니다. 추가 인수는 인디케이터 변수에 대해 전달되지 않습니다.

PARAMETER STYLE GENERAL은 LANGUAGE JAVA와 함께 사용할 수 없습니다.

GENERAL WITH NULLS

CALL문의 매개변수 외에 몇몇 추가 매개변수가 프로시듀어에 전달됨을 지정합니다. 이러한 추가 인수에는 CALL문의 각 매개변수에 대한 요소와 함께 인디케이터 배열이 포함됩니다. C의 경우 짧은 int 배열입니다. 인디케이터 처리 방법에 대한 자세한 내용은 SQL 프로그래밍 개념 책을 참조하십시오.

PARAMETER STYLE GENERAL WITH NULLS는 LANGUAGE JAVA와 함께 사용할 수 없습니다.

JAVA

프로시듀어는 Java 언어와 SQLJ 루틴 스펙을 준수하는 매개변수 전달 규칙을 사용함을 지정합니다. INOUT 및 OUT 매개변수는 리턴하는 값을 이용하기 위해 단일 항목 배열로서 전달됩니다. 증가된 이식성을 위해 PARAMETER STYLE JAVA 규칙을 사용하는 Java 프로시듀어를 작성해야 합니다.

PARAMETER STYLE JAVA는 LANGUAGE JAVA와 함께 지정해야 합니다. JAVA에서 매개변수의 전달에 대한 자세한 정보는 'Developer Kit for Java' 책을 참조하십시오.

외부 함수 언어에 따라 매개변수가 전달되는 방식이 결정됨에 주의하십시오. 예를 들어 C에서는 모든 VARCHAR 또는 CHAR 매개변수가 NUL 종료 스트링으로 전달됩니다. 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.

EXTERNAL NAME *external-program-name*

프로시듀어가 CALL문에 의해 호출될 때 실행될 프로그램을 지정합니다. 프로그램명은 프로시듀어가 호출될 때 서버에 있는 프로그램을 식별해야 합니다. 명명 옵션이 *SYS이고 프로그램명이 규정되지 않은 경우 라이브러리 리스트는 프로시듀어가 호출될 때 프로그램을 찾기 위해 사용됩니다. 프로그램은 ILE 서비스 프로그램이 될 수 없습니다.

이름의 유효성은 서버에서 검사됩니다. 이름의 형식이 올바르지 않으면 오류가 리턴됩니다.

external-program-name이 지정되지 않으면 외부 프로그램명은 프로시듀어명과 동일하다고 가정됩니다.

외부 프로그램은 프로시듀어를 작성할 때는 없어도 되지만 프로시듀어를 호출할 때는 있어야 합니다.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK 및 SET TRANSACTION문은 리모트 서버에서 실행 중인 프로시듀어에서 허용되지 않습니다. COMMIT 및 ROLLBACK문은 ATOMIC SQL 프로시듀어에서 허용되지 않습니다.

DYNAMIC RESULT SETS *integer*

프로시듀어에서 리턴될 수 있는 결과 설정의 최대수를 지정합니다. *integer*는 0보다 크거나 같아야 합니다. 0이 지정되면 결과 세트는 리턴되지 않습니다. 결과 세트는 임의 숫자의 결과 세트를 가질 수 있지만, 언제든지 100개의 프로시듀어만이 페치되기를 기다리는 결과 세트를 가질 수 있습니다. SET RESULT SETS문이 실행된 경우, 리턴되는 결과 수는 이 키워드 및 SET RESULTS SET문에 지정된 최소 결과 세트 수입니다.

결과 세트에서는 화면을 이동할 수 있습니다. 결과 세트를 리턴하는 데 커서가 사용된 경우 결과 세트는 현재 위치에서 시작됩니다. 프로시듀어로부터 리턴되기 전에 5개의 FETCH NEXT 연산이 수행되었다면 결과 세트는 결과 세트의 6번째 행에서 시작됩니다.

결과 세트는 다음 두 조건이 참인 경우에만 리턴됩니다.

- 프로시듀어가 iSeries Access 클라이언트 ODBC 또는 JDBC 드라이버, iSeries 서버 상의 JDBC 또는 SQL 호출 레벨 인터페이스로부터 호출됩니다.
- 외부 프로그램이 ACTGRP(*NEW) 속성을 가지지 않습니다.

결과 세트에 대한 자세한 정보는 790 페이지의 『SET RESULT SETS』를 참조하십시오.

SPECIFIC *specific-name*

프로시듀어에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 프로시듀어명의 규정자와 같습니다. 규정된 경우 규정자는 프로시듀어명의 규정자와 같아야 합니다.

*specific-name*이 지정되지 않으면 프로시듀어명과 같아야 합니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 프로시듀어가 동일한 결과를 리턴하는지 여부를 지정합니다.

NOT DETERMINISTIC

데이터베이스의 참조 데이터가 변경되지 않은 경우, 프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 프로시듀어가 항상 동일한 결과를 리턴합니다.

DETERMINISTIC

데이터베이스의 참조 데이터가 변경되지 않은 경우, 프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 프로시듀어가 동일한 결과를 리턴하지 않습니다.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA 또는 **NO SQL**

프로시듀어 또는 이 프로시듀어에서 호출되는 루틴에서 실행될 수 있는 SQL문(있는 경우)을 지정합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL

프로시듀어에서 SQL 데이터를 읽거나 수정하지 않는 SQL문을 실행할 수 있도록 지정합니다.

NO SQL

프로시듀어가 SQL문을 실행할 수 없도록 지정합니다.

READS SQL DATA

프로시듀어에 SQL 데이터를 수정하지 않는 SQL문을 포함시킬 수 있도록 지정합니다.

MODIFIES SQL DATA

프로시듀어는 프로시듀어에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있음을 지정합니다.

CALLED ON NULL INPUT

매개변수 값이 널(null)인 경우 프로시듀어가 호출되도록 지정합니다.

FENCED 또는 **NOT FENCED**

이 매개변수는 다른 제품과의 호환을 위해 허용되며, iSeries용 DB2 UDB에 의해 사용되지 않습니다.

PROGRAM TYPE MAIN

프로시듀어가 기본 루틴을 실행함을 지정합니다.

DBINFO

데이터베이스 관리자가 상태 정보가 들어 있는 구조를 프로시듀어에 전달해야 하는

지를 지정합니다. 표 46에는 DBINFO 구조에 대한 설명이 나와 있습니다. DBINFO 구조에 대한 자세한 정보는 QSYSINC.H의 포함 파일 SQLUDF에 있습니다.

DBINFO는 PARAMETER STYLE DB2SQL와 함께만 사용될 수 있습니다.

표 46. DBINFO 필드

필드	자료 유형	설명
관계형 데이터베이스	VARCHAR(128)	현재 서버의 이름.
권한부여 ID	VARCHAR(128)	실행시 권한부여 ID.
CCSID 정보	INTEGER INTEGER INTEGER INTEGER CHAR(8)	작업의 CCSID 정보. 다음 정보로 CCSID를 식별합니다. • SBCS CCSID • DBCS CCSID • Mixed CCSID • 세 개의 CCSID 중 가정 적절한 CCSID 지정 • 예약 CCSID가 CREATE PROCEDURE문의 매개변수에 명시적으로 지정되지 않는 경우 입력 스트링은 함수가 실행되는 시점의 작업의 CCSID로 코드화되는 것으로 가정됩니다. 입력 스트링의 CCSID가 매개변수의 CCSID와 같지 않다면 외부 함수로 전달된 입력 스트링은 외부 프로그램을 호출하기 전에 변환될 것입니다.
목표 열	VARCHAR(128) VARCHAR(128) VARCHAR(128)	프로시듀어 호출에 적용될 수 없음.
버전 및 릴리스	CHAR(8)	데이터베이스 관리자의 버전 릴리스 및 수정 레벨.
플랫폼	INTEGER	서버의 플랫폼 유형.

OLD SAVEPOINT LEVEL 또는 NEW SAVEPOINT LEVEL

프로시듀어로의 항목에 대해 새 저장점 레벨을 작성할 것인지 여부를 지정합니다.

OLD SAVEPOINT LEVEL

새 저장점 레벨이 작성되지 않습니다. SAVEPOINT문에 OLD SAVEPOINT LEVEL이 내재적으로 또는 명시적으로 지정되어 있는 프로시듀어 내에서 실행된 SAVEPOINT문은 프로시듀어의 호출자와 같은 저장점 레벨에서 작성됩니다. 이것이 디폴트 값입니다.

NEW SAVEPOINT LEVEL

프로시듀어로의 항목에 대해 새 저장점 레벨이 작성됩니다. 프로시듀어 내에 설정된 모든 저장점은 이 프로시듀어가 호출된 레벨보다 더 깊이 중첩된 저장점 레벨에서 작성됩니다. 따라서, 프로시듀어 내의 새 저장점 세트 이름은 동일한 이름을 가진 더 높은 저장점 레벨(호출 프로그램의 저장점 레벨 등)의 기존 저장점 세트와 충돌하지 않습니다.

COMMIT ON RETURN

데이터베이스 관리자가 프로시듀어로부터 리턴되자마다 트랜잭션을 확약하는지 여부를 지정합니다.

NO

데이터베이스 관리자는 프로시듀어가 리턴될 때 확약을 실행하지 않습니다. NO가 디폴트입니다.

YES

데이터베이스 관리자는 프로시듀어가 정상적으로 리턴되면 확약을 실행합니다. 프로시듀어가 오류를 리턴할 경우, 확약은 실행되지 않습니다.

확약 조작에는 호출 어플리케이션 프로세스 및 프로시듀어에서 수행하는 작업이 포함됩니다.[51]

프로시듀어가 결과 세트를 리턴할 경우, 결과 세트와 연관된 커서는 확약 이후에 사용할 수 있는 WITH HOLD로 정의되어야 합니다.

주

프로시듀어 작성

ILE 외부 프로그램과 연관된 외부 프로시듀어가 작성될 때 프로시듀어의 속성을 연관된 프로그램 오브젝트에 저장하려고 시도합니다. *PGM 오브젝트가 저장된 후 이 시스템이나 다른 시스템으로 복원되면 카탈로그는 그 속성과 함께 자동으로 갱신됩니다.

속성은 다음 제한에 의해 외부 프로시듀어에 대해 저장될 수 있습니다.

- 외부 프로그램 라이브러리는 SYSIBM, QSYS 또는 QSYS2이어서는 안됩니다.
- CREATE PROCEDURE문이 발행될 때 외부 프로그램이 있어야 합니다.
- 외부 프로그램은 ILE *PGM 오브젝트여야 합니다.
- 외부 프로그램에는 적어도 하나의 SQL문이 있어야 합니다.

오브젝트가 갱신될 수 없으면 프로시듀어는 계속 작성됩니다.

프로시듀어를 복원하는 중에는 다음에 유의하십시오.

- 프로시듀어가 처음으로 작성될 때 특정 이름이 지정되었으나 고유하지 않으면 오류가 발행됩니다.
- 특정 이름이 지정되지 않았으면 필요할 때 고유 이름이 생성됩니다.
- 매개변수의 프로시듀어명과 번호가 고유하지 않고 프로시듀어가 등록될 수 없으면 오류가 발행됩니다.

51. 외부 프로그램이 ACTGRP(*NEW)로 작성되었으며 작업 확약 정의가 사용되지 않은 경우, 활성 그룹 종료의 결과 프로시듀어 내에서 수행된 작업은 확약되거나 롤백됩니다.

프로시듀어 호출

DECLARE PROCEDURE문은 작성된 프로시듀어와 같은 이름을 갖는 프로시듀어를 정의하고, 프로시듀어명이 호스트 변수에 의해 식별되지 않는 정적 CALL문이 같은 소스 프로그램에서 실행되는 경우 DECLARE PROCEDURE문의 속성이 CREATE PROCEDURE문의 속성 대신 사용됩니다.

CREATE PROCEDURE문은 프로시듀어명이 호스트 변수에 의해 식별되는 CALL문뿐 아니라 정적 및 동적 CALL문에도 적용됩니다.

외부 프로시듀어가 호출되면 외부 프로그램이 작성될 때 지정된 활성 그룹에서 실행됩니다. 그러나, 프로시듀어가 호출하는 프로그램과 동일한 활성 그룹에서 실행되도록 일반적으로 ACTGRP(*CALLER)는 사용되어야 합니다.

Java 프로시듀어에 대한 주

Java 프로시듀어를 실행할 수 있으려면 시스템에 Developer Kit for Java (5722-JV1)을 설치해야 합니다. 그렇지 않으면 -443의 SQLCODE가 리턴되고 CPDB521 메세지가 작업 기록부에 기록됩니다.

Java 프로시듀어를 실행할 때 오류가 발생하면 -443의 SQLCODE가 리턴됩니다. 오류에 따라서 프로시듀어가 실행된 작업의 작업 기록부에 다른 메세지가 있을 수 있습니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.
- 키워드 SIMPLE CALL은 GENERAL의 동의어로 사용될 수 있습니다.
- DB2GENRL 값은 DB2GENERAL의 동의어로 사용될 수 있습니다.
- DYNAMIC RESULT SET, RESULT SETS, RESULT SET는 DYNAMIC RESULT SETS의 동의어로 사용할 수 있습니다.

예

COBOL 프로그램에서 외부 프로시듀어 PROC1을 작성합니다. CALL문을 사용하여 프로시듀어를 호출하면 라이브러리 LIB1에 있는 PGM1이라는 COBOL 프로그램이 호출됩니다.

```
EXEC SQL
  CREATE PROCEDURE PROC1
        (CHAR(10), CHAR(10))
```

```
        EXTERNAL NAME LIB1.PGM1
        LANGUAGE COBOL GENERAL;

EXEC SQL
CALL PROC1 ('FIRSTNAME ','LASTNAME  ');
```

CREATE PROCEDURE(SQL)

The CREATE PROCEDURE (SQL)문은 현재 서버에서 SQL 프로시듀어를 작성합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- SYSPROCS 카탈로그 뷰 및 SYSPARMS 카탈로그 표의 경우
  - 표에서 INSERT 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 프로그램 작성(CRTPGM) 명령에서 *USE
  - 프로시듀어가 작성되는 라이브러리에서 *EXECUTE 및 *ADD
- 관리 권한

SQL 이름이 지정되고, 프로시듀어가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며, 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

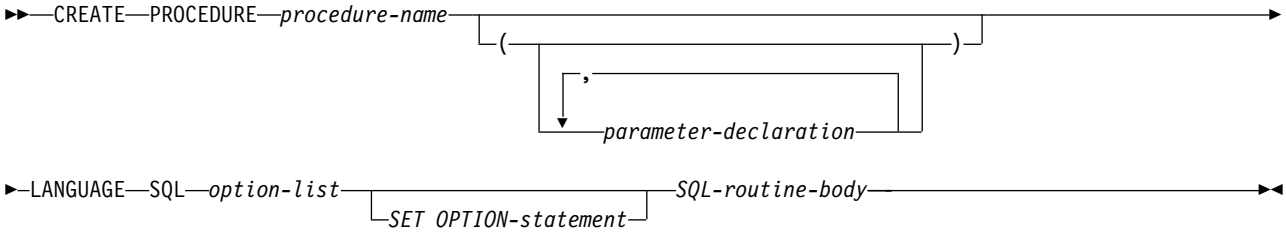
고유한 유형이 참조되다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

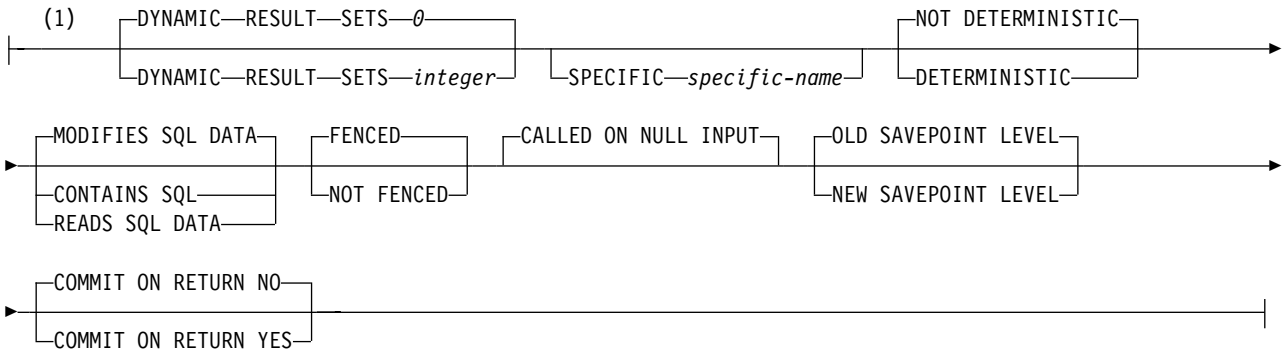
- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문



parameter-declaration:

option-list:

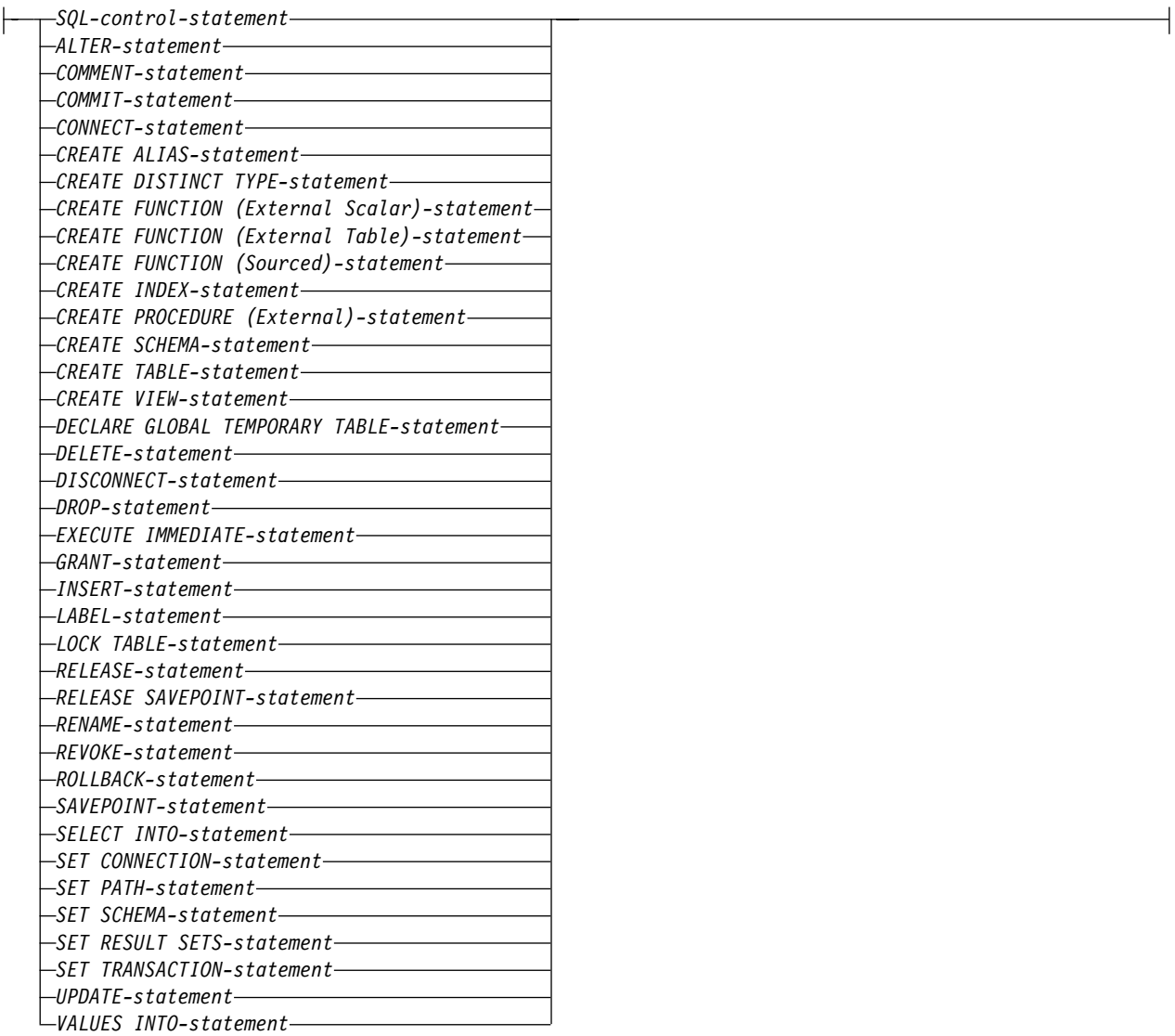


주:

1 옵션 절은 다른 순서로 지정할 수 있습니다.

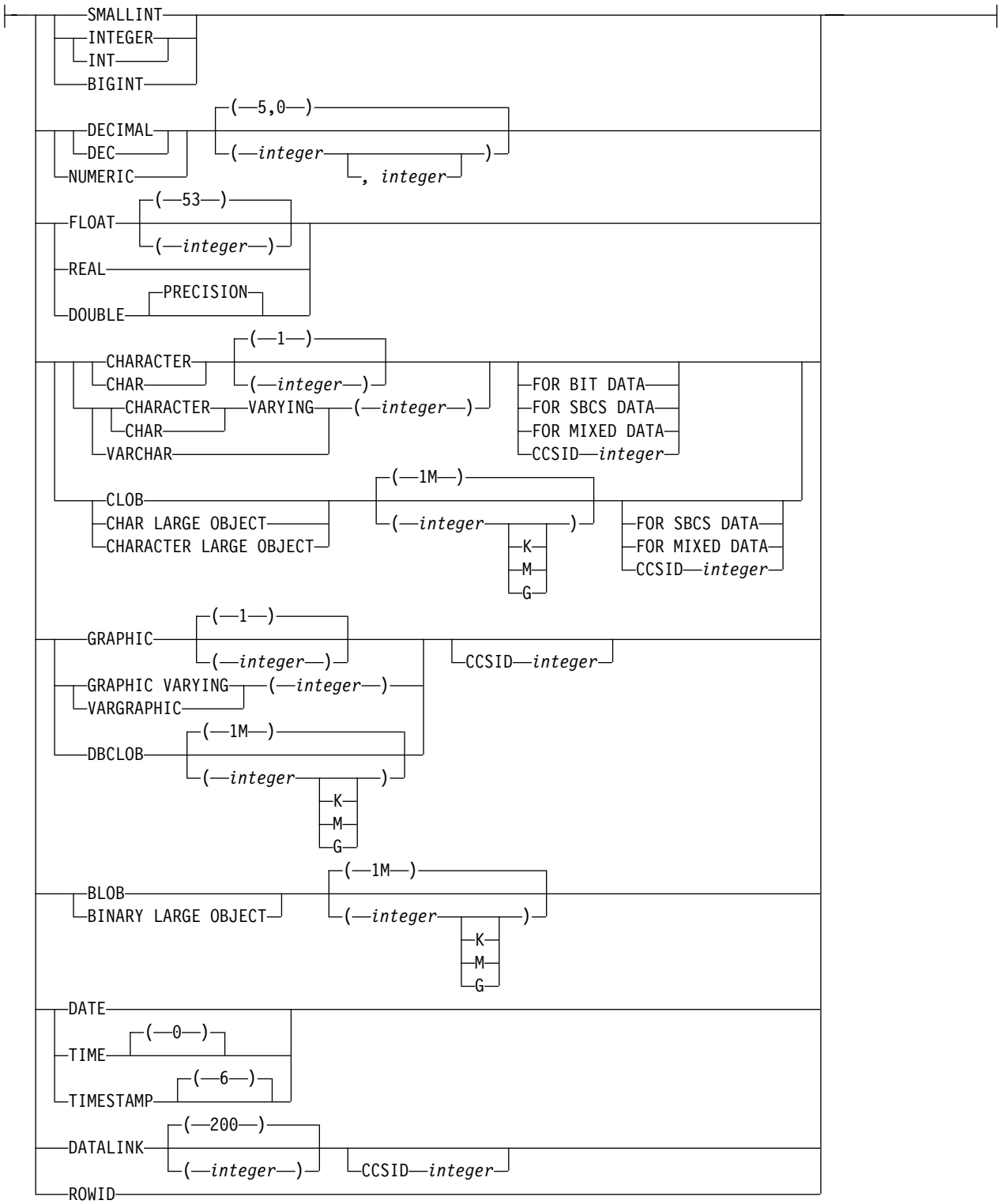
CREATE PROCEDURE(SQL)

SQL-routine-body:



data-type:

built-in-type:



설명

procedure-name

프로시듀어를 명명합니다. 이름의 조합, 콜렉션 이름, 매개변수 수는 현재 서버에 있는 프로시듀어를 식별해서는 안됩니다.

SQL 명명을 위해 프로시듀어는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템 명명을 위해 프로시듀어는 규정자에 의해 지정된 스키마에 작성됩니다. 규정자가 지정되지 않으면 프로시듀어는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면, 프로시듀어는 QGPL에 작성됩니다.

(*parameter-declaration,...)*

프로시듀어의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 프로시듀어의 매개변수는 입력, 출력, 입출력 모두에 사용할 수 있습니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

SQL 프로시듀어에서 사용할 수 있는 매개변수의 최대수는 253입니다.

IN 프로시듀어로의 입력 매개변수로서 매개변수를 지정합니다. 프로시듀어내에서 매개변수의 변경은 제어가 리턴될 때 호출 SQL 어플리케이션에 적용되지 않습니다.

OUT

매개변수를 프로시듀어가 리턴하는 출력 매개변수로 지정합니다. 프로시듀어안에서 매개변수가 설정되지 않으면 널값이 리턴됩니다.

INOUT

프로시듀어의 매개변수를 입력 및 출력 매개변수로 지정합니다.

parameter-name

매개변수를 명명합니다. 이름은 프로시듀어에 대한 다른 *parameter-name*과 같아질 수 없습니다.

data-type

매개변수의 자료 유형을 지정합니다.

자료 유형은 언어 절에 지정된 언어에 대해 유효해야 합니다. 자료 유형에 대한 자세한 정보는 541 페이지의『CREATE TABLE』 및 SQL 프로그래밍 개념 책을 참조하십시오.

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 프로시듀어가 호출될 때 현재 서버에서 디폴트 CCSID에 의해 CCSID가 판별됩니다.

LANGUAGE SQL

이것이 SQL 프로시듀어임을 지정합니다.

DYNAMIC RESULT SETS *integer*

프로시듀어에서 리턴될 수 있는 결과 설정의 최대수를 지정합니다. *integer*는 0보다 크거나 같아야 합니다. 0이 지정되면 결과 세트는 리턴되지 않습니다. 프로시듀어는 임의 숫자의 결과 세트를 가질 수 있지만, 언제든지 100개의 프로시듀어만이 페치되기를 기다리는 결과 세트를 가질 수 있습니다. SET RESULT SETS문이 실행된 경우, 리턴되는 결과 수는 이 키워드 및 SET RESULTS SET문에 지정된 최소 결과 세트 수입니다.

결과 세트에서는 화면을 이동할 수 있습니다. 결과 세트를 리턴하는 데 커서가 사용된 경우 결과 세트는 현재 위치에서 시작됩니다. 프로시듀어로부터 리턴되기 전에 5개의 FETCH NEXT 연산이 수행되었다면 결과 세트는 결과 세트의 6번째 행에서 시작됩니다.

결과 세트는 다음과 같은 경우에만 리턴됩니다.

- 프로시듀어가 iSeries Access 클라이언트 ODBC 또는 JDBC 드라이버, iSeries 서버 상의 JDBC 또는 SQL 호출 레벨 인터페이스로부터 호출됩니다.

결과 세트에 대한 자세한 정보는 790 페이지의 『SET RESULT SETS』를 참조하십시오.

SPECIFIC *specific-name*

프로시듀어에 대한 고유 이름을 제공합니다. 이름은 내재적 또는 명시적으로 스키마명으로 규정됩니다. 스키마명을 포함하는 이름은 현재 서버에 있는 다른 함수나 프로시듀어의 특정 이름을 식별해서는 안됩니다. 규정되지 않은 경우 내재적 규정자는 프로시듀어명의 규정자와 같습니다. 규정된 경우 규정자는 프로시듀어명의 규정자와 같아야 합니다.

*specific-name*이 지정되지 않으면 프로시듀어명과 같아야 합니다. 특정 이름을 갖는 함수나 프로시듀어가 이미 있는 경우 고유한 이름은 고유한 표 이름을 생성하기 위해 사용되는 규칙과 유사하게 생성됩니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 동일한 결과를 리턴하는지 여부를 지정합니다.

NOT DETERMINISTIC

데이터베이스의 참조 데이터가 변경되지 않은 경우, 프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 프로시듀어가 항상 동일한 결과를 리턴합니다.

DETERMINISTIC

데이터베이스의 참조 데이터가 변경되지 않은 경우, 프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 프로시듀어가 동일한 결과를 리턴하지 않습니다.

CONTAINS SQL, READS SQL DATA 또는 **MODIFIES SQL DATA**

프로시듀어 또는 이 프로시듀어에서 호출되는 루틴에서 실행될 수 있는 SQL문(있

는 경우)을 지정합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL
프로시듀어에서 SQL 데이터를 읽거나 수정하지 않는 SQL문을 실행할 수 있도록 지정합니다.

READS SQL DATA
프로시듀어에 SQL 데이터를 수정하지 않는 SQL문을 포함시킬 수 있도록 지정합니다.

MODIFIES SQL DATA
프로시듀어는 프로시듀어에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있음을 지정합니다.

CALLED ON NULL INPUT
매개변수 값이 널(null)인 경우 프로시듀어가 호출되도록 지정합니다.

FENCED 또는 **NOT FENCED**
이 매개변수는 다른 제품과의 호환을 위해 허용되며, iSeries용 DB2 UDB에 의해 사용되지 않습니다.

SET OPTION문
프로시듀어 작성에 사용될 옵션을 지정합니다. 예를 들어 디버그를 할 수 있는 프로시듀어를 작성하려면 다음 명령문이 포함됩니다.

```
SET OPTION DBGVIEW = *STMT
```

자세한 내용은 772 페이지의『SET OPTION』을 참조하십시오.

옵션 CLOSQLCSR, CNULRQD, DFTRDBCOL, DYNDFTCOL 및 NAMING은 CREATE PROCEDURE문에서 허용되지 않습니다.

OLD SAVEPOINT LEVEL 또는 **NEW SAVEPOINT LEVEL**
프로시듀어로의 항목에 대해 새 저장점 레벨을 작성할 것인지 여부를 지정합니다.

OLD SAVEPOINT LEVEL
새 저장점 레벨이 작성되지 않습니다. SAVEPOINT문에 OLD SAVEPOINT LEVEL이 내재적으로 또는 명시적으로 지정되어 있는 프로시듀어 내에서 실행된 SAVEPOINT문은 프로시듀어의 호출자와 같은 저장점 레벨에서 작성됩니다. 이것이 디폴트 값입니다.

NEW SAVEPOINT LEVEL
프로시듀어로의 항목에 대해 새 저장점 레벨이 작성됩니다. 프로시듀어 내에 설정된 모든 저장점은 이 프로시듀어가 호출된 레벨보다 더 깊이 중첩된 저장점 레벨에서 작성됩니다. 따라서, 프로시듀어 내의 새 저장점 세트 이름은 동일한 이름을 가진 더 높은 저장점 레벨(호출 프로그램의 저장점 레벨 등)의 기존 저장점 세트와 충돌하지 않습니다.

COMMIT ON RETURN

데이터베이스 관리자가 프로시듀어로부터 리턴되자마다 트랜잭션을 확약하는지 여부를 지정합니다.

NO

데이터베이스 관리자는 프로시듀어가 리턴될 때 확약을 실행하지 않습니다. NO가 디폴트입니다.

YES

데이터베이스 관리자는 프로시듀어가 정상적으로 리턴되면 확약을 실행합니다. 프로시듀어가 오류를 리턴할 경우, 확약은 실행되지 않습니다.

확약 조작에는 호출 어플리케이션 프로세스 및 프로시듀어를 호출하여 수정되는 작업이 포함됩니다.

프로시듀어가 결과 세트를 리턴할 경우, 결과 세트와 연관된 커서는 확약 이후에 사용할 수 있는 WITH HOLD로 정의되어야 합니다.

SQL-routine-body

복합 명령문을 포함하여 단일 SQL문을 지정합니다. SQL 프로시듀어 정의에 대한 자세한 내용은 821 페이지의 제 6 장『SQL 제어문』을 참조하십시오.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK 및 SET TRANSACTION문은 리모트 서버에서 실행 중인 프로시듀어에서 허용되지 않습니다. COMMIT 및 ROLLBACK문은 ATOMIC SQL 프로시듀어에서 허용되지 않습니다.

주

함수 소유권: SQL명이 지정된 경우, 프로시듀어의 *소유자*는 프로시듀어가 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 프로시듀어의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 프로시듀어의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

프로시듀어 권한: SQL 이름이 사용되면 프로시듀어는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 프로시듀어는 *PUBLIC에 대한 권한으로 작성됩니다.

프로시듀어의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 프로시듀어에 대한 권한도 갖습니다.

프로시듀어 작성

SQL 프로시듀어가 작성되면 SQL은 삽입 SQL문과 함께 C 소스 코드가 들어 있는 임시 소스 파일을 작성합니다. 프로그램 오브젝트는 GRTPGM 명령으로 작성됩니다. 프

로그램을 작성하기 위해 사용된 SQL 옵션은 CREATE PROCEDURE문이 실행될 때 유효한 옵션입니다. 프로그램은 ACTGRP(*CALLER)를 사용하여 작성됩니다.

SQL 프로시듀어가 작성될 때 프로시듀어의 속성은 작성된 프로그램 오브젝트에 저장됩니다. *PGM 오브젝트가 저장된 후 이 시스템이나 다른 시스템으로 복원되면 카탈로그는 그 속성과 함께 자동으로 갱신됩니다.

프로시듀어를 복원하는 중에는 다음에 유의하십시오.

- 프로시듀어가 처음으로 작성될 때 특정 이름이 지정되었으나 고유하지 않으면 오류가 발행됩니다.
- 특정 이름이 지정되지 않았으면 필요할 때 고유 이름이 생성됩니다.
- 매개변수의 프로시듀어명과 번호가 고유하지 않고 프로시듀어가 등록될 수 없으면 오류가 발행됩니다.

프로시듀어명이 유효한 시스템명이면 소스 파일에 있는 멤버 이름과 프로그램 오브젝트 이름으로 사용됩니다. 프로시듀어명이 유효한 시스템명이 아니면 고유 이름이 생성됩니다. 같은 이름을 갖는 소스 파일 멤버가 이미 존재하면 멤버는 중복됩니다. 같은 이름을 갖는 모듈이나 프로그램이 이미 존재하면 오브젝트는 중복되지 않고 고유 이름이 생성됩니다. 고유 이름은 시스템 표 이름을 생성하는 규칙에 따라 생성됩니다.

프로시듀어 호출

DECLARE PROCEDURE문은 작성된 프로시듀어와 같은 이름을 갖는 프로시듀어를 정의하고, 프로시듀어명이 호스트 변수에 의해 식별되지 않는 정적 CALL문이 같은 소스 프로그램에서 실행되는 경우 DECLARE PROCEDURE문의 속성이 CREATE PROCEDURE문의 속성 대신 사용됩니다.

CREATE PROCEDURE문은 프로시듀어명이 호스트 변수에 의해 식별되는 CALL문뿐 아니라 정적 및 동적 CALL문에도 적용됩니다.

SQL 프로시듀어는 SQL CALL문을 사용하여 호출되어야 합니다. 호출될 때 SQL 프로시듀어는 호출하는 프로그램의 활성 그룹에서 실행됩니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.
- DYNAMIC RESULT SET, RESULT SETS, RESULT SET는 DYNAMIC RESULT SETS의 동의어로 사용할 수 있습니다.

예

SQL 프로시듀어에 대한 정의를 작성합니다. 프로시듀어는 입력으로 사원 번호와 급여 인상을 위한 승수를 받습니다. 다음 타스크는 프로시듀어 본문에서 수행됩니다.

- 해당 사원의 새로운 급여를 계산합니다.
- 사원 표를 새로운 급여 값으로 갱신합니다.

```
EXEC SQL
  CREATE PROCEDURE UPDATE_SALARY_1
          (IN EMPLOYEE_NUMBER CHAR(10),
           IN RATE DECIMAL(6,2))
LANGUAGE SQL
          MODIFIES SQL DATA
          UPDATE EMP
            SET SALARY = SALARY + RATE
            WHERE EMPNO = EMPLOYEE_NUMBER
```

CREATE SCHEMA

CREATE SCHEMA문은 현재 서버에서 스키마를 정의하고 선택적으로 표, 뷰, 별명, 색인 및 고유한 유형을 작성합니다. 주석과 레이블은 표, 뷰, 별명, 색인, 열 및 고유한 유형의 카탈로그 설명에 추가될 수 있습니다. 표, 뷰 및 고유한 유형 권한은 사용자에게 부여될 수 있습니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음 CL 명령에 대한 *USE 시스템 권한
  - 라이브러리 작성(CRTLIB)
  - WITH DATA DICTIONARY가 지정된 경우 자료 사전 작성(CRTDTADCT)
- 관리 권한

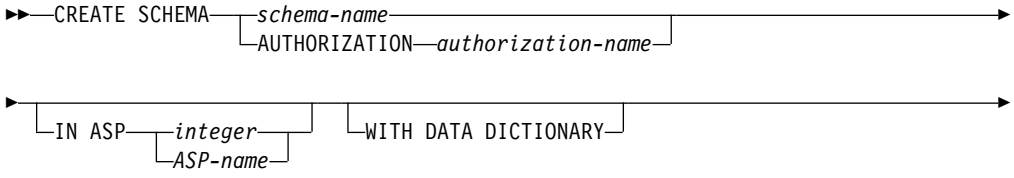
명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

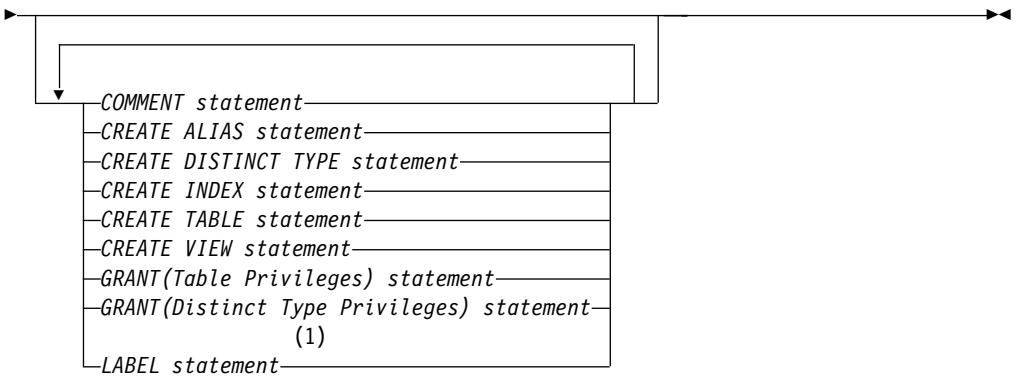
- CREATE SCHEMA(스키마 프로세서)문에 포함된 권한 정의된 각 SQL문
- 관리 권한

AUTHORIZATION절이 지정되면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 권한부여 이름이 식별한 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

구문





주:

1 패키지, 프로시듀어, 함수 및 매개변수의 레이블과 주석은 CREATE SCHEMA문에서 지원되지 않습니다.

설명

schema-name

스키마(schema)를 명명합니다. 이 이름을 사용하여 스키마를 작성합니다. schema-name이 지정된 경우 명령문의 권한부여 ID는 실행시 권한부여 ID입니다. 이름은 현재 서버에 있는 기존의 스키마의 이름과 동일해서는 안됩니다. 스키마의 *소유자*는 명령문을 실행하는 작업의 사용자 프로파일 또는 그룹 사용자 프로파일입니다.

스키마의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 스키마에 대한 권한도 갖습니다.

authorization-name

명령문의 권한부여 ID를 식별합니다. 권한부여 이름도 schema-name입니다. 이름은 현재 서버에 있는 기존의 스키마의 이름과 동일해서는 안됩니다.

IN ASP *integer*

스키마를 작성할 보조 기억장치 풀(ASP)을 지정합니다. 정수는 1과 32 사이의 값이어야 합니다. 1이 지정되면 스키마는 시스템 ASP에 작성됩니다. 구가 생략되면 1의 ASP가 가정됩니다.

IN ASP *ASP-name*

스키마를 작성할 보조 기억장치 풀(ASP)을 지정합니다. 이름은 현재 서버에 있는 보조 기억장치 풀(pool)을 식별해야 합니다.

WITH DATA DICTIONARY

이 절이 지정되면 IDDU 자료 사전이 스키마에 작성됩니다.

자료 사전을 사용하여 작성된 스키마에는 LOB나 DATALINK 열을 갖는 표가 포함될 수 없습니다.

COMMENT 문

표, 뷰 또는 열의 카탈로그 설명에 주석을 추가하거나 대체합니다. 패키지에 대한 주석은 허용되지 않습니다. 408 페이지의 『COMMENT』의 COMMENT문을 참조하십시오.

CREATE ALIAS문

스키마에 별명을 작성합니다. 432 페이지의 CREATE ALIAS문을 참조하십시오.

CREATE DISTINCT TYPE문

스키마에 사용자 정의 고유한 유형을 작성합니다. 435 페이지의 『CREATE DISTINCT TYPE』의 CREATE DISTINCT TYPE문을 참조하십시오.

CREATE INDEX문

스키마에 색인을 작성합니다. 506 페이지의 CREATE INDEX문을 참조하십시오.

CREATE TABLE문

스키마에 표를 작성합니다. 541 페이지의 CREATE INDEX문을 참조하십시오.

CREATE VIEW문

스키마에 뷰를 작성합니다. 590 페이지의 CREATE VIEW문을 참조하십시오 .

GRANT(표 권한)문

스키마의 표와 뷰에 대한 권한을 부여합니다. 698 페이지의 『GRANT(표 권한)』의 GRANT문을 참조하십시오.

GRANT(고유한 유형 권한)문

스키마의 고유한 유형에 대한 권한을 부여합니다. 684 페이지의 『GRANT(고유한 유형 권한)』의 GRANT문을 참조하십시오.

LABEL 문

스키마의 표, 뷰 또는 열의 카탈로그 설명의 레이블을 추가하거나 대체합니다. 패키지에 대한 레이블은 허용되지 않습니다. 717 페이지의 LABEL문을 참조하십시오.

주

스키마 속성: 스키마는 다음과 같이 작성됩니다.

- 라이브러리: 오브젝트와 관련된 라이브러리 그룹이며 이름으로 오브젝트를 찾을 수 있습니다.
- 카탈로그: 카탈로그는 스키마에 표, 뷰, 색인 및 패키지의 설명을 포함합니다. 카탈로그는 뷰 세트로 구성되며, WITH DATA DICTIONARY가 지정된 경우 IDDU 자료 사전으로 구성됩니다. 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.
- 저널과 저널 리시버: 저널 QSQJRN과 저널 리시버 QSQJRN0001이 스키마에 작성되고, 이어서 스키마에 작성되는 모든 표의 변경을 작성하는 데 사용됩니다. 자세한 정보는 iSeries Information Center의 저널 관리 주제를 참조하십시오.

분배된 표에 대해 작성된 색인은 표가 분배되는 서버 전체에서 작성됩니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

오브젝트 소유권: 작성된 오브젝트의 소유자는 다음과 같이 판별됩니다.

- AUTHORIZATION절이 지정되면 지정된 권한부여 ID는 명령문에 의해 작성된 모든 오브젝트를 소유합니다.
- AUTHORIZATION절이 지정되지 않고 SQL 이름이 지정되면 명령문에 의해 작성된 모든 오브젝트의 *소유자*는 schema-name과 같은 이름을 갖는 사용자 프로파일입니다(그 이름을 갖는 사용자 프로파일이 있는 경우).
- 그렇지 않으면 명령문에 의해 작성된 모든 오브젝트의 *소유자*는 명령문을 실행하는 작업의 사용자 프로파일 또는 그룹 사용자 프로파일입니다.

오브젝트 권한: SQL 이름이 사용되면 오브젝트는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성되고 라이브러리 작성 권한 매개변수 CRTAUT(*EXCLUDE)에 의해 작성됩니다. 소유자는 스키마에 대한 권한을 소유하는 유일한 사용자입니다. 다른 사용자가 스키마에 대한 권한을 요구하면 소유자는 오브젝트 권한 부여(GRTOBJAUT) CL 명령을 사용하여 작성된 오브젝트에 권한을 부여할 수 있습니다.

시스템명이 사용되면 스키마 및 다른 오브젝트가 시스템 값 QCRTAUT에 의해 판별된 *PUBLIC에 부여된 시스템 권한으로 작성되고 라이브러리는 CRTAUT(*SYSVAL)로 작성됩니다. 시스템 보안에 대한 자세한 정보는 iSeries 보안 참조서  및 SQL 프로그래밍 개념 책을 참조하십시오.

스키마의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 스키마에 대한 권한도 갖습니다.

오브젝트명: CREATE TABLE, CREATE INDEX, CREATE ALIAS, CREATE DISTINCT TYPE 또는 CREATE VIEW문에 작성되는 표, 색인, 별명, 고유한 유형 또는 뷰에 대하여 규정된 이름이 있으면, 그 규정된 이름에 지정된 스키마 이름은 작성되는 스키마 이름과 같아야 합니다. 스키마(schema) 정의 내에서 참조되는 다른 표나 뷰 이름은 스키마명에 의해 규정될 수 있습니다. SQL문의 규정되지 않은 표, 색인, 별명, 고유한 유형 또는 뷰 이름은 작성된 스키마 이름으로 내재적으로 규정됩니다.

분리 문자는 SQL문 사이에서 사용되지 않습니다.

SQL문 길이:CREATE SCHEMA문의 개별 CREATE TABLE, CREATE INDEX, CREATE DISTINCT TYPE, CREATE VIEW, COMMENT, LABEL 또는 GRANT 문의 최대 길이는 65536입니다.

키워드 동의어: COLLECTION 키워드는 이전에 릴리스의 SCHEMA의 동의어와 호환되어 사용될 수 있습니다. 이 키워드는 비표준이고 사용될 수 없습니다.

예

예 1

재고 부품 표와 부품 번호에 대한 색인을 갖는 스키마를 작성합니다. 사용자 프로파일 JONES에 스키마(schema)에 대한 권한을 부여합니다.

```
CREATE SCHEMA INVENTORY

CREATE TABLE PART (PARTNO SMALLINT NOT NULL,
                   DESCR  VARCHAR(24),
                   QUANTITY INT)

CREATE INDEX PARTIND ON PART (PARTNO)

GRANT ALL ON PART TO JONES
```

예 2

SMITH의 권한부여 ID를 사용하여 스키마를 작성합니다. 학생 번호 열에 대한 주석이 있는 학생 표를 작성합니다.

```
CREATE SCHEMA AUTHORIZATION SMITH

CREATE TABLE SMITH.STUDENT (STUDNBR SMALLINT NOT NULL UNIQUE,
                            LASTNAME CHAR(20),
                            FIRSTNAME CHAR(20),
                            ADDRESS CHAR(50))

COMMENT ON STUDENT (STUDNBR IS 'THIS IS A UNIQUE ID#')
```

CREATE TABLE

CREATE TABLE문은 현재 서버에서 표를 정의합니다. 정의에는 표 이름과 열의 이름 및 속성이 포함되어야 합니다. 정의에는 1차 키와 같은 표의 다른 속성이 포함될 수 있습니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 실제 파일 작성(CRTPF) 명령에 대한 *USE
  - 표가 작성되는 라이브러리에 대한 *EXECUTE 및 *ADD
  - 저널에 대한 *OBJOPR 및 *OBJMGT
  - 표가 작성되는 라이브러리가 자료 사전을 가지고 있는 SQL 스키마인 경우 자료 사전에 대한 *CHANGE
- 관리 권한

SQL 이름이 지정되고, 표가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며, 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 그 이름을 갖는 사용자 프로파일에 대한 시스템 권한 *ADD
- 관리 권한

외부 키를 정의하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 사항 중 하나를 상위 표에 대해 포함해야 합니다.

- 표에 대한 오브젝트 관리 권한이나 REFERENCES 권한
- 지정된 상위 키의 각 열에 대한 REFERENCES 권한
- 표의 소유권
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 표에 대해 REFERENCES 권한을 갖습니다.

- 표의 소유자입니다.
- 표에 대해 REFERENCES 권한을 부여받았습니다.
- 표에 대해 *OBJREF이나 *OBJMGT의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 중 하나가 참일 때 표의 열에 대해 REFERENCES 권한을 갖습니다.

- 표의 소유자입니다.
- 열에 대해 REFERENCES 권한을 부여받았습니다.
- 열에 대해 *OBJREF의 시스템 권한을 부여받았거나 표에 대해 *OBJMGT의 시스템 권한을 부여받았습니다.

LIKE절 또는 AS 선택문이 지정되면 명령문의 권한부여 ID가 보유하는 권한은 이 절에 지정된 표나 뷰에 대하여 적어도 다음 중 하나를 포함해야 합니다.

- 표나 뷰에 대한 SELECT 권한
- 표나 뷰에 대한 소유권
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을, 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

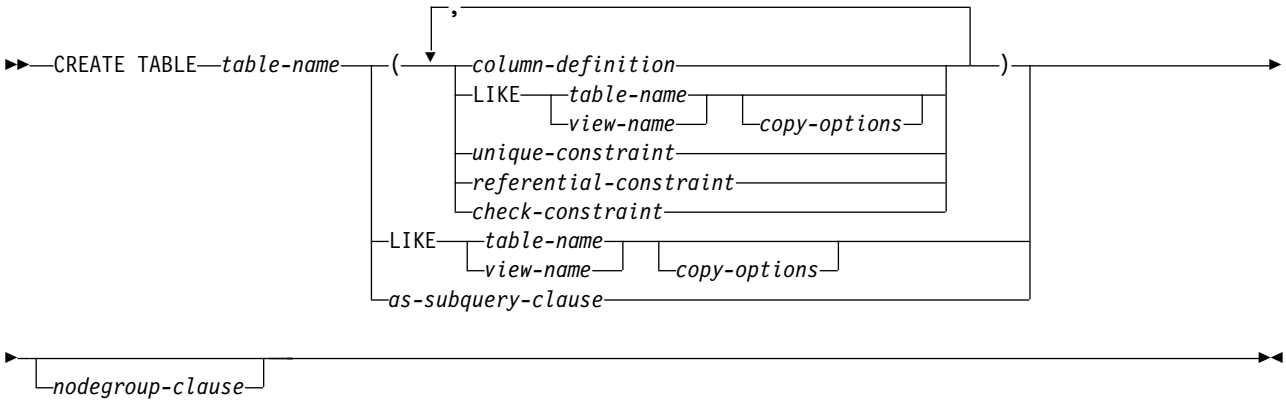
고유한 유형이 참조되다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

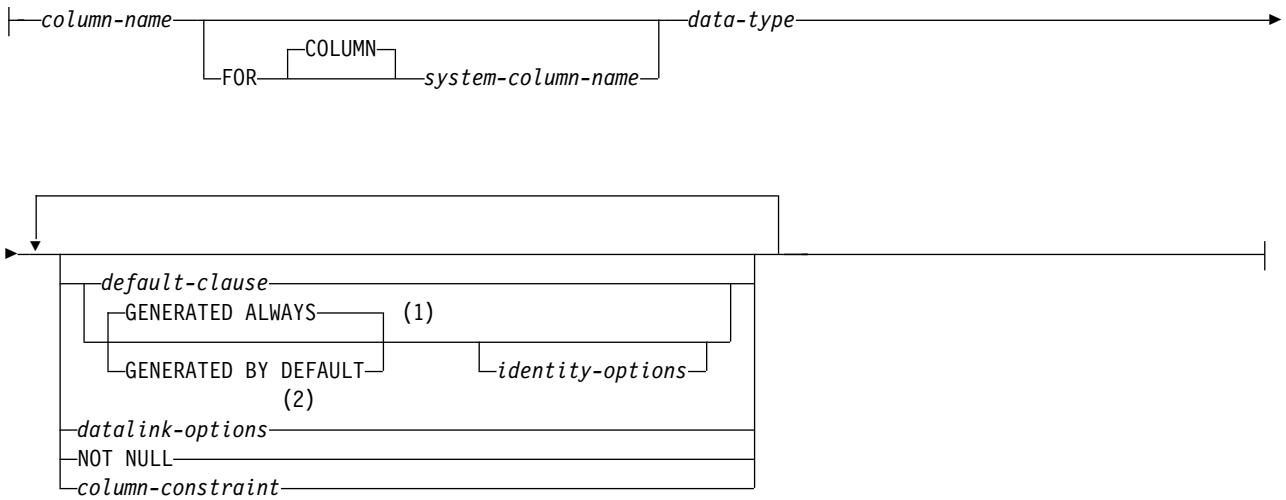
명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

구문



column-definition:



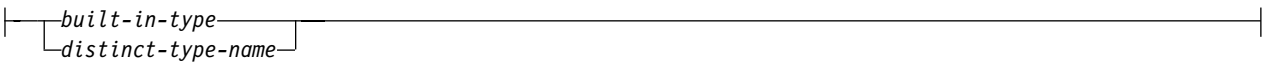
주:

1 GENERATED는 열이 ROWID 자료 유형(또는 ROWID 자료 유형에 기초한 고유 유형)이거나 ID 열일 경우에만 지정할 수 있습니다.

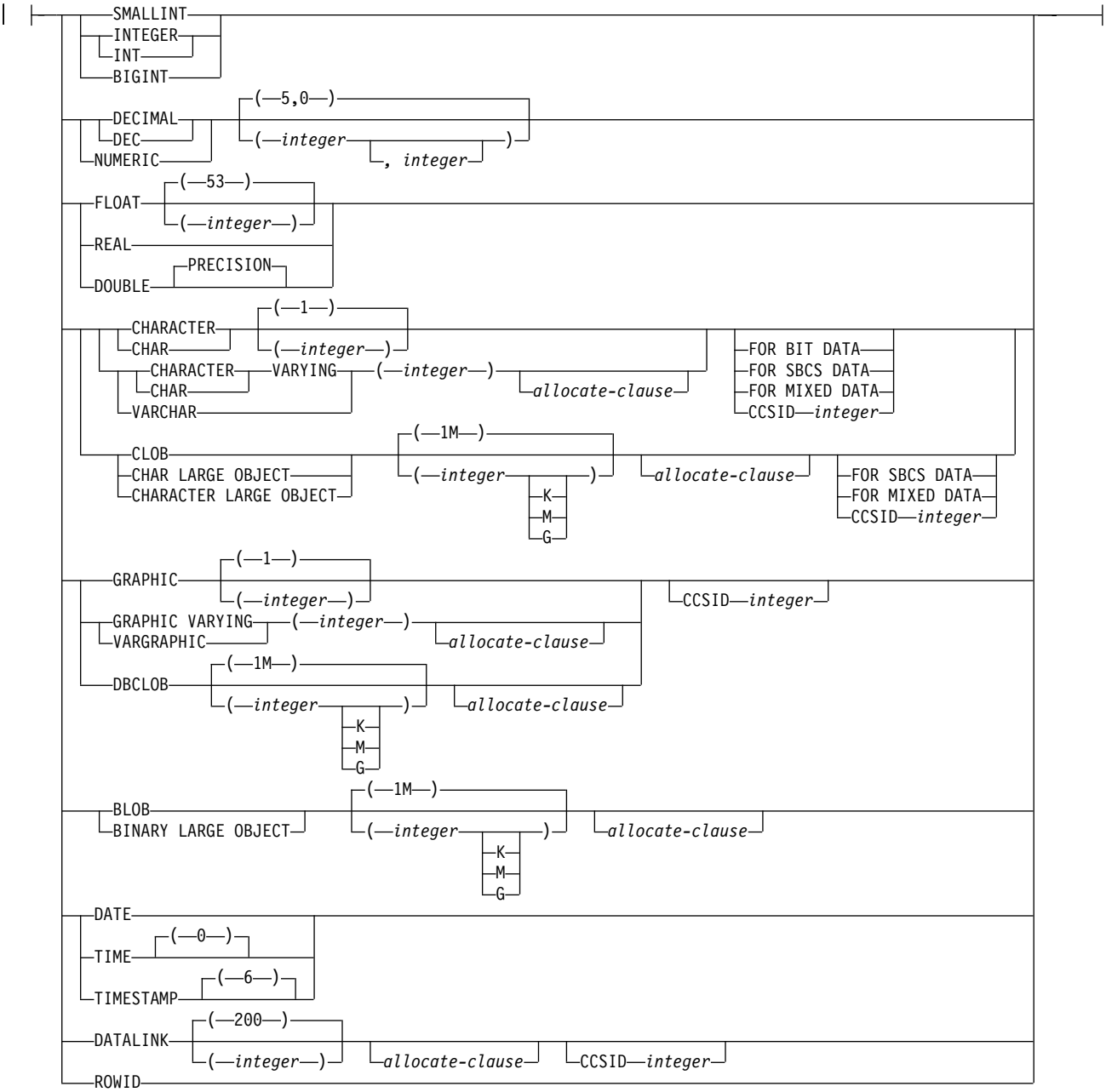
2 datalink-options는 DATALINK와 DATALINK를 소스로 하는 distinct-type에 대해서만 지정될 수 있습니다.

CREATE TABLE

data-type:

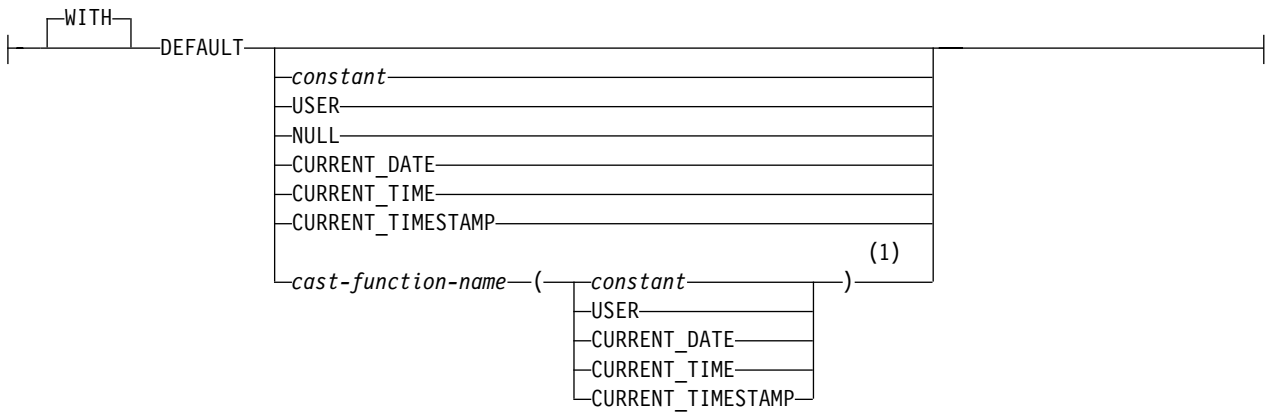


built-in-type:

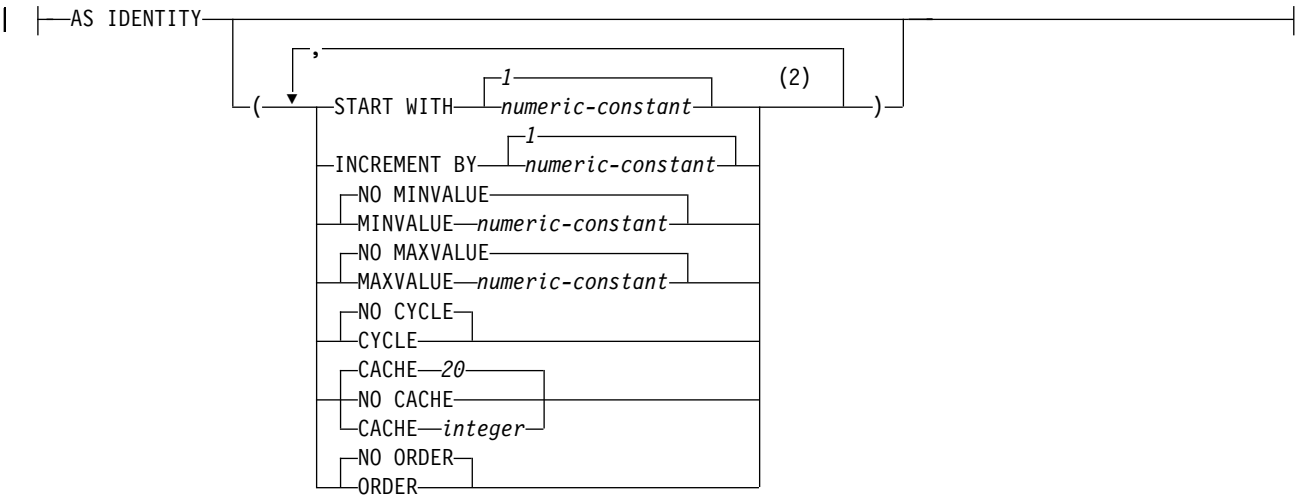


allocate절:

default절:



identity-options:

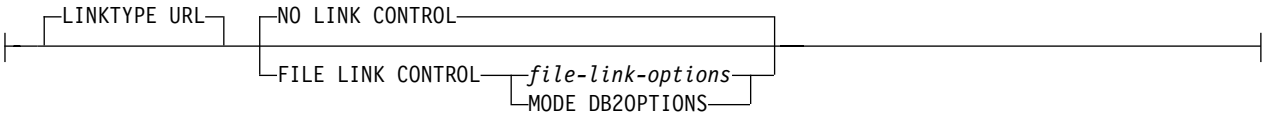


주:

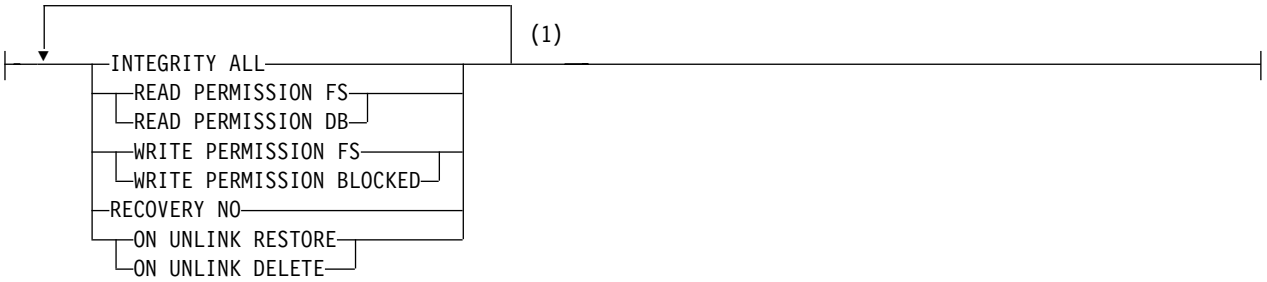
1 이 형식의 DEFAULT 값은 고유한 유형으로 정의된 열과 함께만 사용될 수 있습니다.

2 각 절은 한 번만 지정됩니다.

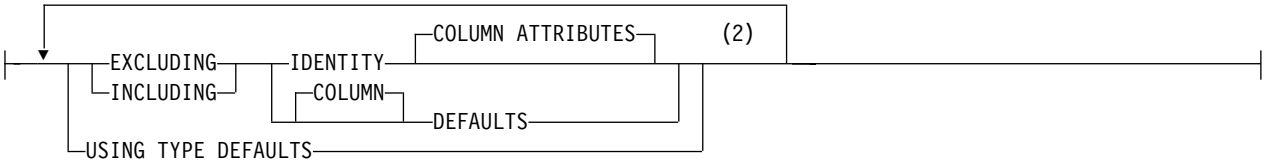
datalink-options:



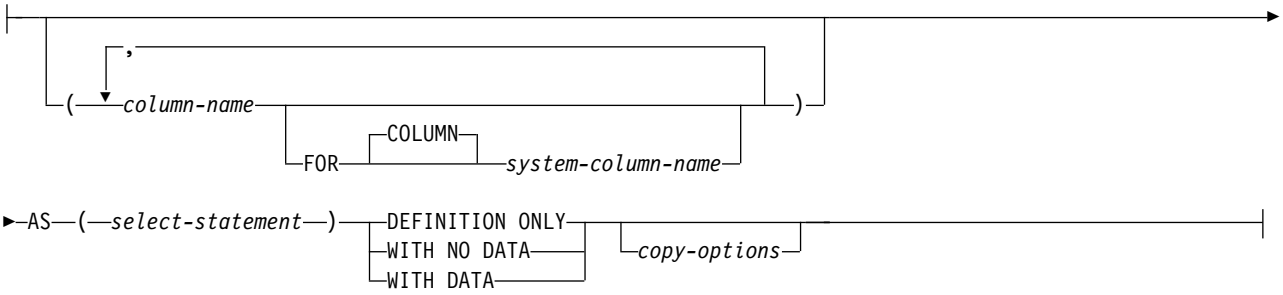
file-link-options:



copy-options:



as-subquery-clause:

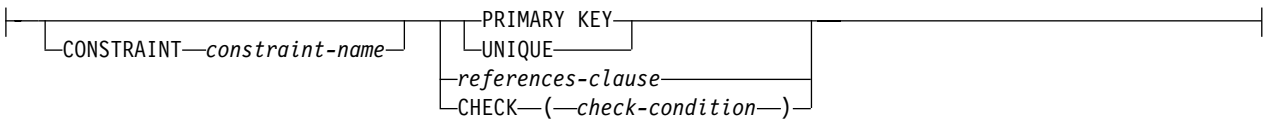


주:

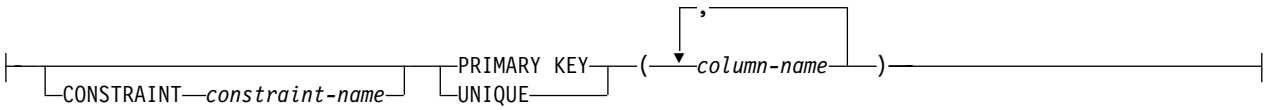
1 모든 다섯 개의 *파일 링크 선택사항*이 지정되어야 합니다. 그러나 순서는 상관없습니다.

2 각 절은 한 번만 지정됩니다.

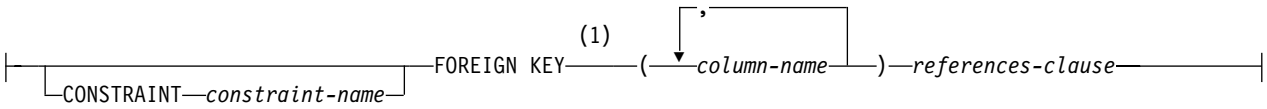
column-constraint:



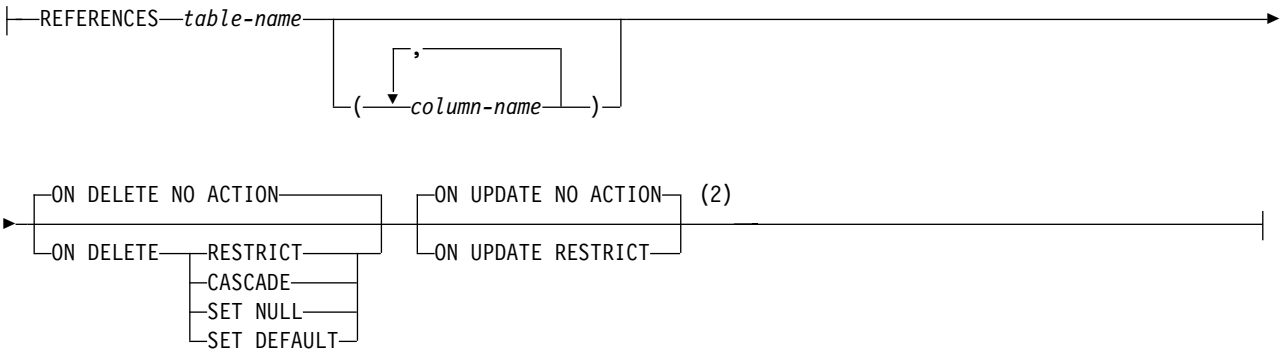
unique-constraint:



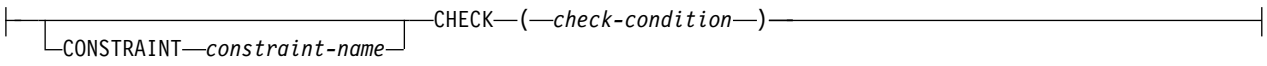
referential-constraint:



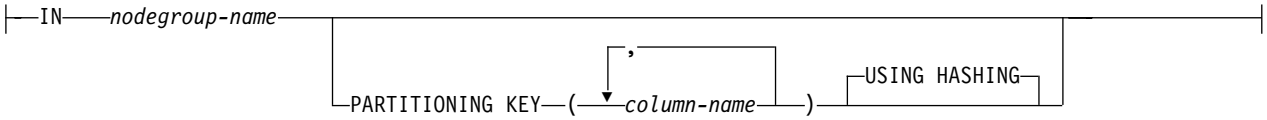
references절:



check-constraint:



nodegroup절:



주:

1 다른 제품과의 호환을 위해 *constraint-name*(CONSTRAINT 키워드가 없는)을 FOREIGN KEY 다음에 지정할 수 있습니다.

2 ON DELETE와 ON UPDATE절은 순서에 관계 없이 지정할 수 있습니다.

설명

table-name

표를 명명합니다. 내재적 또는 명시적 규정자를 포함하여 이름은 이미 현재 서버에 있는 색인, 표, 뷰, 별명 또는 파일과 같을 수 없습니다.

SQL명이 지정되면 표는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템명이 지정되면 표는 규정자에 의해 지정된 스키마에 작성됩니다. 규정되지 않으면 표는 현재 라이브러리(*CURLIB)에 작성됩니다. 현재 라이브러리가 없으면, 표는 QGPL에 작성됩니다.

column-definition

열의 속성을 정의합니다. 하나 이상 8000개 이하의 열 정의가 있어야 합니다.

열의 레코드 행 바이트 수 합계는 32766 보다 커서는 안되고, VARCHAR이나 VARGRAPHIC 열이 지정된 경우에는 32740 보다 크면 안됩니다. 또한 LOB가 지정되면 열의 행 자료 바이트 수 합계는 15,728,640보다 커서는 안됩니다. 자료 유형에 따른 열의 바이트 수에 대한 내용은 569 페이지의 『주』를 참조하십시오.

column-name

표의 열을 명명합니다. *column-name*을 규정해서는 안되면 표의 하나 이상의 열이나 system-column-name에 대해 같은 이름을 사용할 수 없습니다.

FOR COLUMN *system-column-name*

열에 대해 OS/400 이름을 제공합니다. 표의 하나 이상의 열이나 column-name에 대해 같은 이름을 사용할 수 없습니다.

system-column-name이 지정되지 않고 column-name이 유효한 system-column-name이 아니면, 시스템 열 이름이 생성됩니다. 시스템 열 이름이 생성되는 방법에 대한 자세한 내용은 572 페이지의 『열 이름 생성 규칙』을 참조하십시오.

data-type

열의 자료 유형을 지정합니다.

내장형

*내장형*은 다음과 같이 사용합니다.

SMALLINT

작은 정수의 경우

INTEGER 또는 **INT**

큰 정수의 경우

BIGINT

큰 정수의 경우

DECIMAL(*integer***,***integer***)** 또는 **DEC(***integer***,***integer***)**
DECIMAL(*integer***)** 또는 **DEC(***integer***)**
DECIMAL 또는 **DEC**
팩 십진수. 첫 번째 정수는 수의 정밀도 즉, 총 자릿수로, 범위는 1부터 31까지입니다. 두 번째 정수는 수의 눈금입니다(소수점 오른쪽의 자릿수). 범위는 0부터 수의 정밀도까지입니다.

DECIMAL(*p*,0)의 경우는 DECIMAL(*p*)을, 그리고 DECIMAL(5,0)의 경우는 DECIMAL을 사용할 수 있습니다.

NUMERIC(*integer***,***integer***)**
NUMERIC(*integer***)**
NUMERIC
존(zone) 십진수. 첫 번째 정수는 수의 정밀도 즉, 총 자릿수이며 범위는 1부터 31까지입니다. 두 번째 정수는 수의 눈금입니다(소수점 오른쪽의 자릿수). 범위는 0부터 수의 정밀도까지입니다.

NUMERIC(*p*,0)의 경우는 NUMERIC(*p*), NUMERIC(5,0)의 경우는 NUMERIC을 사용할 수 있습니다.

FLOAT
배정밀도 부동 소수점 수의 경우

FLOAT(*integer***)**
정수의 값에 따른 단정밀도 또는 배정밀도 부동 소수점 수의 경우. 정수 값은 1부터 53까지의 범위이어야 합니다. 1부터 24까지의 값은 단정밀도, 25부터 53까지의 값은 배정밀도를 표시합니다. 디폴트는 53입니다.

REAL
단정밀도 부동 소수점의 경우

DOUBLE PRECISION 또는 **DOUBLE**
배정밀도 부동 소수점의 경우

CHARACTER(*integer***)** 또는 **CHAR(***integer***)**
CHARACTER 또는 **CHAR**
integer 길이의 고정 길이 문자 스트링의 경우. 정수의 범위는 1부터 32766까지입니다(널(null)이 허용되는 경우 32765). FOR MIXED DATA나 혼합된 자료 코드화 문자 세트 ID(CCSID)가 지정되면 범위는 4부터 32766까지입니다(널(null)이 허용되는 경우 32765). 길이 스펙이 생략되면 1 문자의 길이가 가정됩니다.

CHARACTER VARYING (*integer***)** 또는 **CHAR VARYING (***integer***)** 또는 **VARCHAR(***integer* **)**
최대 길이 *integer*의 가변 길이 문자 스트링일 경우 범위는 1부터 32740까지

입니다(널(null)이 허용되는 경우 32739). FOR MIXED DATA나 혼합된 자료 코드화 문자 세트 ID(CCSID)가 지정되면 범위는 4부터 32740까지입니다(널(null)이 허용되는 경우 32739).

CLOB(*integer***[K|M|G])** 또는 **CHAR LARGE OBJECT(***integer***[K|M|G])** 또는 **CHARACTER LARGE OBJECT(***integer***[K|M|G])**
CLOB 또는 **CHAR LARGE OBJECT** 또는 **CHARACTER LARGE OBJECT**

지정된 최대 길이의 문자 큰 오브젝트 스트링의 경우. 최대 길이의 범위는 1부터 2 147 483 647까지여야 합니다. FOR MIXED DATA 또는 혼합 자료 CCSID가 지정된 경우 범위는 4에서 2 147 483 647까지입니다. 길이 스펙이 생략되면 1메가바이트(MB)의 길이가 가정됩니다. CLOB은 분배된 표에서 사용될 수 없습니다.

integer

정수의 최대값은 2 147 483 647입니다. 스트링의 최대 길이는 *integer*입니다.

integer **K**

정수의 최대값은 2 097 152입니다. 스트링의 최대 길이는 *integer*의 1,024배입니다.

integer **M**

정수의 최대값은 2 048입니다. 스트링의 최대 길이는 *integer*의 1,048,576배입니다.

integer **G**

정수의 최대값은 2입니다. 스트링의 최대 길이는 *integer*의 1,073,741,824배입니다.

GRAPHIC(*integer***)**
GRAPHIC

integer 길이의 고정 길이 그래픽 스트링의 경우는 범위가 1부터 16383까지입니다(널(null)이 허용되는 경우 16382). 길이 스펙이 생략되면 1 문자의 길이가 가정됩니다.

VARGRAPHIC(*integer***)** 또는 **GRAPHIC VARYING(***integer***)**

*integer*의 최대 길이를 갖는 가변 길이 그래픽 스트링의 경우 범위는 1부터 16370까지입니다(널(null)이 허용되는 경우 16369).

DBCLOB(*integer***[K|M|G])**
DBCLOB

지정된 최대 길이의 2바이트 문자 큰 오브젝트 스트링의 경우.

---

52. 이 옵션은 다른 제품과의 호환을 위해 제공됩니다. 대신 VARCHAR(integer) 또는 VARGRAPHIC(integer)를 지정할 것이 권장됩니다.

최대 길이의 범위는 1부터 1 073 741 823까지여야 합니다. 길이 스펙이 생략되면 1메가바이트(MB)의 길이가 가정됩니다. DBCLOB은 분배된 표에서 사용될 수 없습니다.

integer
정수의 최대값은 1 073 741 823입니다. 스트링의 최대 길이는 *integer*입니다.

integer **K**
정수의 최대값은 1 028 576입니다. 스트링의 최대 길이는 *integer*의 1,024배입니다.

integer **M**
정수의 최대값은 1 024입니다. 스트링의 최대 길이는 *integer*의 1,048,576배입니다.

integer **G**
정수의 최대값은 1입니다. 스트링의 최대 길이는 *integer*의 1,073,741,824배입니다.

BLOB(*integer*[**K**|**M**|**G**]) 또는 **BINARY LARGE OBJECT(***integer***[K|M|G])**
BLOB 또는 **BINARY LARGE OBJECT**
지정된 최대 길이의 2진 큰 오브젝트 스트링의 경우. 최대 길이의 범위는 1부터 2 147 483 647까지여야 합니다. 길이 스펙이 생략되면 1메가바이트(MB)의 길이가 가정됩니다. BLOB는 분배된 표에서는 사용될 수 없습니다.

integer
정수의 최대값은 2 147 483 647입니다. 스트링의 최대 길이는 *integer*입니다.

integer **K**
정수의 최대값은 2 097 152입니다. 스트링의 최대 길이는 *integer*의 1,024배입니다.

integer **M**
정수의 최대값은 2 048입니다. 스트링의 최대 길이는 *integer*의 1,048,576배입니다.

integer **G**
정수의 최대값은 2입니다. 스트링의 최대 길이는 *integer*의 1,073,741,824배입니다.

DATE
날짜의 경우

TIME
시간의 경우

TIMESTAMP
시간소인의 경우

DATALINK(*integer*) 또는 **DATALINK**
지정된 최대 길이의 자료 링크의 경우. 최대 길이의 범위는 1부터 32717까지여야 합니다. FOR MIXED DATA 또는 혼합 자료 CCSID가 지정된 경우 범위는 4에서 32717까지입니다. 지정된 길이는 가장 큰 예상 URL과 DataLink 주석을 둘 다 포함할 정도로 충분해야 합니다. 길이 스펙이 생략되면 200의 길이가 가정됩니다. DATALINK는 분배된 표에서 사용될 수 없습니다.

DATALINK 값은 내장 스칼라 함수 세트로 캡슐화된 값입니다. DLVALUE 함수는 DATALINK 값을 생성합니다. 다음 함수는 DATALINK 값에서 속성을 추출하기 위해 사용될 수 있습니다.

- DLCOMMENT
- DLLINKTYPE
- DLURLCOMPLETE
- DLURLPATH
- DLURLPATHONLY
- DLURLSCHEME
- DLURLSERVER

자료 링크는 색인의 일부가 될 수 없습니다. 따라서 1차 키, 외부 키 또는 고유 제한의 열로 포함될 수 없습니다.

ROWID
행 ID의 경우. 표에서 단 하나의 ROWID 행이 허용됩니다.

distinct-type-name
열의 자료 유형을 고유한 유형(사용자 정의 자료 유형)으로 지정합니다. 열의 길이와 정밀도, 스케일은 각각 고유한 유형의 소스 유형의 길이와 정밀도, 스케일입니다. 고유한 유형 이름이 스키마명 없이 지정되면 고유한 유형 이름은 SQL 경로의 스키마를 찾아서 해석됩니다.

ALLOCATE(*integer*)
VARCHAR, VARGRAPHIC 및 LOB 유형에 대해 각 행의 열에 대해 예약된 공간을 지정합니다. 할당된 값 보다 작거나 같은 길이를 갖는 열 값은 행의 고정 길이 부분에 저장됩니다. 할당된 값 보다 큰 길이를 갖는 열 값은 행의 가변 길이 부분에 저장되며, 검색하기 위해 추가적인 입/출력 조작을 필요로 합니다. 할당된 값의 범위는 1부터, 최대 행 버퍼 크기 한계에 따라 스트링의 최대 길이까지입니다. 최대 행 버퍼 크기에 대한 자세한 정보는 569 페이지의 『최대 행 크기』를 참조하십시오. FOR MIXED나 혼합된 자료 코드화 문자 세트 ID(CCSID)가 지정되면 범위는 4부터 스트링의 최대 길이까지입니다. 할당된 길

이 스펙이 생략되면 0의 할당된 길이가 가정됩니다. VARGRAPHIC의 경우 정수는 DBCS 또는 UCS-2 문자 수입니다. 상수가 디폴트 값에 대해 지정되고 ALLOCATE 길이가 디폴트 값의 길이보다 작으면 ALLOCATE 길이는 디폴트 값의 길이로 가정됩니다.

FOR BIT DATA

열의 값이 코드화된 문자 세트와 연관되지 않고, 결코 변환되지 않음을 지정합니다. FOR BIT DATA는 CHARACTER 또는 VARCHAR 열 대해서만 유효합니다. FOR BIT DATA 열의 코드화 문자 세트 ID(CCSID)는 65535입니다. FOR BIT DATA는 CLOB 열에서는 허용되지 않습니다.

FOR SBCS DATA

열의 값에 SBCS(1바이트 문자 세트) 자료가 들어 있음을 지정합니다. 표가 작성될 때 현재 서버에 있는 디폴트 코드화 문자 세트 ID(CCSID)가 DBCS로 될 수 없거나 열의 길이가 4보다 작으면, FOR SBCS DATA는 CHAR, VARCHAR 및 CLOB 열에 대한 디폴트입니다. FOR SBCS DATA는 CHARACTER, VARCHAR 또는 CLOB 열에 대해서만 유효합니다. FOR SBCS DATA의 코드화 문자 세트 ID(CCSID)는 표가 작성될 때 현재 서버에 있는 디폴트 CCSID에 의해 판별됩니다.

FOR MIXED DATA

열의 값에 SBCS 자료와 DBCS 자료가 둘 다 들어 있음을 지정합니다. 표가 작성될 때 현재 서버에 있는 디폴트 CCSID가 DBCS가 될 수 없고 열의 길이가 3보다 크면, FOR MIXED DATA는 CHAR, VARCHAR 및 CLOB 열에 대한 디폴트입니다. 모든 FOR MIXED DATA 열은 DBCS 개방 데이터베이스 필드입니다. FOR MIXED DATA는 CHARACTER, VARCHAR 또는 CLOB 열에 대해서만 유효합니다. FOR MIXED DATA의 코드화 문자 세트 ID(CCSID)는 표가 작성될 때 현재 서버에 있는 디폴트 CCSID에 의해 판별됩니다.

CCSID 정수

열의 값에 CCSID 정수의 자료가 들어 있음을 지정합니다. 정수가 SBCS 코드화 문자 세트 ID(CCSID)이면 열은 SBCS 자료입니다. 정수가 혼합된 자료 코드화 문자 세트 ID(CCSID)이면 열은 혼합된 자료이고 열의 길이는 3 보다 커야 합니다. 문자 열의 경우 CCSID는 SBCS CCSID이거나 혼합된 자료 CCSID여야 합니다. 그래픽 열의 경우 코드화 문자 세트 ID(CCSID)는 DBCS 또는 UCS-2 CCSID여야 합니다. 코드화 문자 세트 ID(CCSID)가 그래픽 열로 정의되지 않으면 CCSID는 표가 작성될 때 현재 서버에 있는 디폴트 CCSID에 의해 판별됩니다. 유효한 CCSID 리스트는 899 페이지의 부록 E『코드화 문자 세트 ID(CCSID) 값』을 참조하십시오.

DEFAULT

열에 대한 디폴트 값을 지정합니다. 이 절은 *column-definition*에 한 번 이상 지정

될 수 없습니다. ROWID 열이나 ID 열(AS IDENTITY로 정의된 열)에는 디폴트 값을 지정할 수 없습니다. 데이터베이스 관리자는 ROWID 열 및 ID 열의 디폴트 값을 생성합니다. 디폴트 키워드 다음에 값이 지정되지 않으면 다음이 적용됩니다.

- 열이 널이면 디폴트 값은 널값입니다.
- 열이 널이 아니면 디폴트는 열의 자료 유형에 따라 다릅니다.

자료 유형	디폴트 값
숫자	0
고정 길이 스트링	공백
가변 길이 스트링	길이가 0인 스트링
날짜	INSERT 시의 현재 날짜
시간	INSERT 시의 현재 시간
시간소인	INSERT 시의 현재 시간소인
자료 링크	DLVALUE(‘ ’,‘URL’,‘ ’)에 해당하는 값
distinct-type	고유한 유형의 소스 유형에 대응하는 디폴트 값

*column-definition*에서 NOT NULL과 DEFAULT의 생략은 DEFAULT NULL의 내재적 스펙입니다.

constant

열에 대한 디폴트 값으로 상수를 지정합니다. 지정된 상수는 80 페이지의 『지정과 비교』에 설명된 대로 지정 규칙에 따라 열에 지정될 수 있는 값을 표시해야 합니다. 부동 소수점 상수를 SMALLINT, INTEGER, DECIMAL 또는 NUMERIC 열에 대해 사용해서는 안됩니다. 소수 상수는 열의 지정된 눈금보다 더 많은 소수점 이하의 자릿수를 가질 수 없습니다.

USER

INSERT 또는 UPDATE 시의 USER 특수 레지스터의 값을 열의 디폴트 값으로 지정합니다. 열의 자료 유형은 길이 속성이 USER의 특수 레지스터보다 크거나 같은 CHAR나 VARCHAR가 되어야 합니다.

NULL

열에 대한 디폴트 값으로 널을 지정합니다. NOT NULL이 지정되면 동일한 열 정의 내에 DEFAULT NULL이 지정되어서는 안됩니다.

CURRENT_DATE

열에 대한 디폴트 값으로 현재 날짜를 지정합니다. CURRENT_DATE가 지정되면 열의자료 유형은 DATE 또는 DATE에 기초한 고유한 유형이어야 합니다.

CURRENT_TIME

열에 대한 디폴트 값으로 현재 시간을 지정합니다. CURRENT_TIME이 지정되면 열의 자료 유형은 TIME 또는 TIME에 기초한 고유한 유형이어야 합니다.

CURRENT_TIMESTAMP

열에 대한 디폴트 값으로 현재 시간소인을 지정합니다.

CURRENT_TIMESTAMP가 지정되면 열의 자료 유형은 TIMESTAMP이거나 TIMESTAMP에 기초한 고유한 유형이어야 합니다.

cast-function-name

이 형식의 디폴트 값은 고유한 유형, BLOB, CLOB, DBCLOB, DATE, TIME 또는 TIMESTAMP 자료 유형으로 정의된 열과 함께만 사용될 수 있습니다. 다음 표는 이 *cast-function*이 허용되는 사용법을 설명한 것입니다.

자료 유형	캐스트 함수명
BLOB, CLOB 또는 DBCLOB를 기초로 하는 고유한 유형 N	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP를 기초로 하는 고유한 유형 N	N(N이 생성되었을 때 생성된 사용자 정의 캐스트 함수) ** 또는 DATE, TIME 또는 TIMESTAMP *
기타 자료 유형을 기초로 하는 고유한 유형 N	N(N이 생성되었을 때 생성된 사용자 정의 캐스트 함수) **
BLOB, CLOB 또는 DBCLOB	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP	DATE, TIME 또는 TIMESTAMP *

주:

* 함수 이름은 내재적 또는 명시적 스키마명 QSYS2를 갖는 자료 유형의 이름(또는 고유한 유형의 소스 유형)과 일치해야 합니다.

** 함수 이름은 열에 대한 고유한 유형 이름과 일치해야 합니다. 스키마 이름으로 규정했으면 고유한 유형에 대한 스키마 이름과 같아야 합니다. 규정하지 않았으면 함수 해석의 스키마 이름이 고유한 유형에 대한 스키마 이름과 같아야 합니다.

constant

상수를 인수로 지정합니다. 상수는 고유한 유형의 소스 유형 또는 고유한 유형이 아닌 경우 자료 유형에 대한 상수 규칙을 따라야 합니다. BLOB, CLOB, DBCLOB, DATE, TIME 및 TIMESTAMP 함수의 경우 상수는 스트링 상수이어야 합니다.

USER

INSERT 또는 UPDATE 시에 USER 특수 레지스터의 값을 열에 대한 디폴트 값으로 지정합니다. 열의 고유한 유형의 소스 유형에 대한 자료 유형은 길이 속성이 USER 특수 레지스터의 길이 속성보다 크거나 같은 CHAR나 VARCHAR가 되어야 합니다.

CURRENT_DATE

열에 대한 디폴트 값으로 현재 날짜를 지정합니다. CURRENT_DATE가 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 DATE가 되어야 합니다.

CURRENT_TIME

열에 대한 디폴트 값으로 현재 시간을 지정합니다. CURRENT_TIME이 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 TIME이 되어야 합니다.

CURRENT_TIMESTAMP

열에 대한 디폴트 값으로 현재 시간소인을 지정합니다.

CURRENT_TIMESTAMP가 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 TIMESTAMP가 되어야 합니다.

GENERATED

데이터베이스 관리자가 열의 생성 값을 지정합니다. 열이 ID 열로 간주되는 경우 (AS IDENTITY절로 정의) GENERATED를 지정해야 합니다. 열의 자료 유형이 ROWID인 경우(또는 ROWID에 기초한 고유 유형)에도 이 값을 지정해야 합니다. 반면에 지정되어서는 안됩니다.

ALWAYS

표에 행이 삽입될 때 데이터베이스 관리자가 항상 열 값을 생성하도록 지정합니다. ALWAYS는 권장 값입니다.

BY DEFAULT

열에 대해 값이 지정되지 않은 경우에만 행이 삽입될 때 데이터베이스 관리자가 항상 열 값을 생성하도록 지정합니다. 값이 지정되면, 데이터베이스 관리자는 값을 사용합니다.

ROWID 열에 대해 데이터베이스 관리자는 지정된 값을 사용하지만, 그 값은 이전에 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에서 생성된 유효한 고유 행 ID여야 합니다.

ID 열에 데이터베이스 관리자는 지정된 값을 삽입하지만 ID 열에 해당 ID 열을 고유하게 지정하는 고유 제한조건이나 고유 인덱스가 있지 않은 경우 해당 값이 열의 고유 값임을 검증하지 않습니다.

AS IDENTITY

열은 표의 ID 열임을 지정합니다. 표는 하나의 ID 열만 가질 수 있습니다. AS IDENTITY는 열이 자료 유형이 스케일 0을 가진 정확한 숫자 유형(스케일 0인 SMALLINT, INTEGER, BIGINT, DECIMAL 또는 NUMERIC, 이러한 유형에 기초한 고유 유형)인 경우에만 지정할 수 있습니다.

ID 행은 묵시적으로 NOT NULL입니다.

START WITH *numeric-constant*

ID 열에 대해 생성된 첫 번째 값을 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 말아야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았다면, 디폴트 값은 오름차순의 경우 MINVALUE, 내림차순의 경우 MAXVALUE입니다. 이 값은 순서의 최대값 또는 최소값에 도달한 후 순서가 순환하는 값일 필요는 없습니다. START WITH절을 사용하면 주기에 사용된 범위의 바깥에 순서를 시작할 수 있습니다. 주기에 사용된 범위는 MINVALUE 및 MAXVALUE로 정의됩니다.

INCREMENT BY *numeric-constant*

ID 열의 연속 값 사이의 간격을 지정합니다. 이 값은 0이 아닌 양수이거나 음수일 수 있지만 큰 정수 상수 값을 초과하지 않으며, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 않는 열에 지정할 수 있습니다. 디폴트는 1입니다.

값이 양수라면, ID 열의 값 순서는 오름차순입니다. 값이 음수라면, ID 열의 값 순서는 내림차순입니다.

MAXVALUE *numeric-constant*

이 ID 열에 대해 생성된 최대값인 숫자 상수를 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 최소값보다 커야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았을 때, 이 값은 오름차순의 경우 자료 유형의 최대값(DECIMAL의 경우 정밀도), 내림차순의 경우 START WITH 값(START WITH가 지정되지 않은 경우 -1)입니다.

MINVALUE *numeric-constant*

이 ID 열에 대해 생성된 최소값인 숫자 상수를 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 최대값보다 적어야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았을 때, 이 값은 오름차순의 경우 START WITH 값(START WITH가 지정되지 않은 경우 -1), 내림차순의 경우 자료 유형의 최소값(DECIMAL의 경우 정밀도)입니다.

CACHE 또는 NO CACHE

사전 할당된 일부 값을 메모리에 보존할 것인지 여부를 지정합니다. 값을 사전 할당하여 캐시에 저장하면 표에 행을 삽입하는 기능이 좋아집니다.

CACHE *integer*

데이터베이스 관리자가 사전에 할당하여 메모리에 보관하는 ID 열의 순서 값 수를 지정합니다. 지정할 수 있는 최소값은 2이고, 최대값은 정수로 표현될 수 있는 가장 큰 값입니다. 디폴트는 20입니다.

시스템 오류가 발생하면, 지정될 캐시된 모든 ID 열 값이 유실되어 다시는 사용되지 않습니다. 따라서, CACHE에 대해 지정된 값은 시스템 오류 중 유실될 수 있는 ID 열의 최대 값 수를 나타냅니다.

NO CACHE

ID 열의 값이 사전에 할당되지 않도록 지정합니다.

CYCLE 또는 NO CYCLE

순서의 최대값 또는 최소값에 도달한 후 이 ID 열이 계속 값을 생성하는지 여부를 지정합니다.

CYCLE

최대값이나 최소값에 도달한 후 이 열에 대해 값이 계속 생성됨을 지정합니다. 이 옵션이 사용된 경우, 오름차순이 순서의 최대값에 도달한 후 해당 최소값을 생성합니다. 내림차순이 순서의 최소값에 도달하고 나면, 최대값이 생성됩니다. 열의 최대값과 최소값은 주기에 사용된 범위를 결정합니다.

CYCLE이 유효한 경우, 데이터베이스 관리자는 ID 열에 대해 중복값을 생성할 수 있습니다. ID 열에 대해 제한조건 또는 고유 색인이 존재하는데 이 열에 대해 고유하지 않은 값이 생성되면 오류가 발생합니다.

NO CYCLE

순서의 최대값이나 최소값에 도달한 후 ID 열에 대해 값이 생성되지 않도록 지정합니다. 이것이 디폴트 값입니다.

ORDER 또는 NO ORDER

요청 순서대로 ID 값이 생성되도록 지정합니다.

ORDER

요청 순서대로 값이 생성되도록 지정합니다.

NO ORDER

요청 순서대로 값이 생성될 필요가 없도록 지정합니다. 이것이 디폴트 값입니다.

datalink-options

DATALINK 자료 유형과 연관된 옵션을 지정합니다.

LINKTYPE URL

URL(Uniform Resource Locator)로 링크의 유형을 정의합니다.

NO LINK CONTROL

링크된 파일이 존재하는지 판별하기 위해 검사하지 않음을 지정합니다. URL 구문만 검사됩니다. 링크된 파일을 제어하는 데이터베이스 관리자가 없습니다.

FILE LINK CONTROL

링크된 파일이 존재하는지 검사해야함을 지정합니다. 데이터베이스 관리자가 링크된 파일을 더 잘 제어할 수 있도록 추가 옵션이 사용될 수 있습니다.

FILE LINK CONTROL이 지정되면 각 파일은 한 번만 링크될 수 있습니다. 즉, 단일 표에 단일 FILE LINK CONTROL 열에만 URL을 지정할 수 있습니다.

file-link-options

데이터베이스 관리자가 링크된 파일을 제어하는 레벨을 정의하는 추가 옵션

INTEGRITY

DATALINK 값과 실제 파일 사이 링크의 무결성 레벨을 지정합니다.

ALL

DATALINK 값으로 지정된 파일은 데이터베이스 관리자의 제어하에 있고 표준 파일 시스템 프로그래밍 인터페이스를 사용하여 삭제되거나 재명명되지 않을 것입니다.

READ PERMISSION

DATALINK 값에 지정된 파일을 읽을 권한을 판별하는 방법을 지정합니다.

FS

읽기 액세스 권한은 파일 시스템 권한에 의해 판별됩니다. 이러한 파일은 열에서 파일명을 검색하지 않고 액세스될 수 있습니다.

DB

읽기 액세스 권한은 데이터베이스에 의해 판별됩니다. 파일에 대한 액세스는 열기 조작에서 DATALINK 값 검색시 표에서 리턴된 유효한 파일 액세스 토큰을 전달한 경우에만 허용됩니다. READ PERMISSION DB가 지정되면 WRITE PERMISSION BLOCKED가 지정되어야 합니다.

WRITE PERMISSION

DATALINK에 지정된 파일에 작성할 권한을 판별하는 방법을 지정합니다.

FS

쓰기 액세스 권한은 파일 시스템 권한에 의해 판별됩니다. 이러한 파일은 열에서 파일명을 검색하지 않고 액세스될 수 있습니다.

BLOCKED

쓰기 액세스가 블록됩니다. 파일은 인터페이스를 통해서 직접 갱신될 수 없습니다. 대체 메카니즘은 정보를 갱신하기 위해 사용되어야 합니다. 예를 들면, 파일이 복사되고 사본이 갱신된 다음, DATALINK 값이 파일의 새로운 사본을 가리키기 위해 갱신됩니다.

RECOVERY

데이터베이스 관리자가 이 열에 있는 값으로 참조된 파일의 시점 회복을 지원하는지 여부를 지정합니다.

NO

시점 회복이 지원되지 않음을 지정합니다.

ON UNLINK

DATALINK 값이 변경되거나 삭제될 때(링크되지 않을 때) 파일에 취한 조치를 지정합니다. WRITE PERMISSION FS가 사용될 때는 적용되지 않습니다.

RESTORE
파일이 링크되지 않았을 때 자료 링크 파일 관리자가, 파일이 링크되었을 때 가졌던 권한을 갖는 소유자에게 파일을 리턴하려고 시도함을 지정합니다. 사용자가 파일 서버에 더 이상 등록되어 있지 않은 경우 결과는 파일이 들어 있는 파일 시스템에 따라 달라집니다. 파일이 AIX 파일 시스템에 있으면 소유자는 ″dfmunknown″입니다. 파일이 IFS에 있으면 소유자는 QDLFM입니다. INTEGRITY ALL과 WRITE PERMISSION BLOCKED도 지정되어 있을 때만 지정될 수 있습니다.

DELETE
링크되어 있지 않을 때 파일이 삭제됨을 지정합니다. READ PERMISSION DB와 WRITE PERMISSION BLOCKED도 지정되어 있을 때만 지정될 수 있습니다.

MODE DB2OPTIONS
이 모드는 디폴트 파일 링크 옵션 세트를 정의합니다. DB2OPTIONS로 정의한 디폴트 값은 다음과 같습니다.

- INTEGRITY ALL
- READ PERMISSION FS
- WRITE PERMISSION FS
- RECOVERY NO

NOT NULL
열이 널값을 포함하지 않도록 합니다. NOT NULL의 생략은 열이 널(null)이 될 수 있음을 의미합니다.

column-constraint

CONSTRAINT*constraint-name*
제한사항을 명명합니다. *constraint-name*은 CREATE TABLE문에서 이미 지정되었고 이미 현재 서버에 있는 제한사항을 식별해서는 안됩니다.

구가 지정되지 않으면 고유 제한 이름은 데이터베이스 관리자에 의해 생성됩니다.

PRIMARY KEY
단일 열로 구성된 1차 키를 정의하는 축약 메소드를 제공합니다. 따라서, PRIMARY KEY가 열 C의 정의에 지정되면 결과는 PRIMARY KEY(C)절이 분리된 절로 지정된 경우와 동일합니다.

이 절은 하나 이상의 열 정의에 지정되어서는 안되고 UNIQUE 절이 열 정의에 지정된 경우에는 절대로 정의되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다.

1차 키가 추가되면 NULL 값이 1차 키를 구성하는 열에 허용되지 않는 규칙을 강제하기 위해 CHECK 제한이 내재적으로 추가됩니다.

UNIQUE

단일 열로 구성된 고유 키를 정의하는 축약 메소드를 제공합니다. 따라서, UNIQUE가 열 C의 정의에 지정되면 결과는 UNIQUE(C)절이 분리된 절로 지정된 경우와 동일합니다.

이 절은 열 정의에 한 번 이상 지정될 수 없으며 PRIMARY KEY가 열 정의에 지정된 경우에는 지정되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다.

references절

*column-definition*의 *reference-clause*은 단일 열로 구성된 외부 키를 정의하는 축약 메소드를 제공합니다. 따라서, reference절이 열 C의 정의에 지정되면 결과는 C가 유일하게 식별되는 열인 FOREIGN KEY절의 일부로 reference절이 지정되는 경우와 동일합니다.

CHECK(*check-condition***)**

*column-definition*의 CHECK(*check-condition*)가 *check-condition*만이 단일 열을 참조하는 검사 제한사항을 정의하는 축약 메소드를 제공합니다. 따라서, CHECK가 열 C의 열 정의에 지정되면 C 이외의 어떤 열도 검사 제한사항의 *check-condition*에서 참조될 수 없습니다. 결과는 검사 제한사항이 분리된 절로 지정된 경우와 동일합니다.

FILE LINK CONTROL 열의 ROWID 또는 DATALINK는 CHECK 제한조건에 참조될 수 없습니다. 추가 제한 사항은 567 페이지의 『check-constraint』를 참조하십시오.

LIKE

table-name 또는 *view-name*

지정된 표나 뷰에 정의된 열이 이 표에 포함되도록 지정합니다. *table-name* 또는 *view-name*은 이미 서버에 있는 표나 뷰를 식별해야 합니다.

LIKE를 사용하면 n 열을 내재적으로 정의하게 되는데, n은 식별된 표나 뷰의 열의 수입니다. 내재적 정의에는(해당 자료 유형에 적용할 수 있는 경우) 다음과 같은 n 열의 속성이 포함됩니다.

- 열 이름 (및 시스템 열 이름)
- 자료 유형, 길이, 정밀도 및 스케일
- CCSID

LIKE 절이 *table-name* 바로 다음에 지정되고 괄호 안에 들어 있지 않다면 다음의 열 속성도 포함되며, 그렇지 않은 경우 포함되지 않습니다(디폴트 값 및 ID 속성은 또한 *copy-options*를 사용하여 제어될 수 있습니다).

- *table-name*이 지정되고 *view-name*이 지정되지 않은 경우 디폴트 값
- 널 가능성
- ID 속성
- 열 머리말 및 텍스트(717 페이지의 『LABEL』 참조)

내재적 정의에는 식별된 표나 뷰의 다른 선택적 속성을 포함하지 않습니다. 예를 들어 새로운 표는 표의 1차 키, 외부 키난 트리거를 자동으로 포함하지 않습니다. 선택적 절이 명시적으로 지정된 경우에만 새로운 표가 이들 및 기타 선택적 속성을 갖게 됩니다.

지정된 표나 뷰가 비SQL 작성의 실제 파일 또는 논리 파일인 경우 비SQL 속성이 제거됩니다. 예를 들어 날짜와 시간 형식은 ISO로 변환됩니다.

as-subquery-clause

column-name

표의 열을 명명합니다. *column-name*을 규정해서는 안되면 표의 하나 이상의 열이나 system-column-name에 대해 같은 이름을 사용할 수 없습니다.

FOR COLUMN *system-column-name*

열에 대해 OS/400 이름을 제공합니다. 표의 하나 이상의 열이나 column-name에 대해 같은 이름을 사용할 수 없습니다.

system-column-name이 지정되지 않고 column-name이 유효한 system-column-name이 아니면, 시스템 열 이름이 생성됩니다. 시스템 열 이름이 생성되는 방법에 대한 자세한 내용은 572 페이지의 『열 이름 생성 규칙』을 참조하십시오.

select-statement

표의 열이 select-statement가 실행될 경우 select-statement문의 도출된 결과 표에 표시될 열과 동일한 이름과 설명을 가지도록 지정합니다. AS select-statement의 사용은 표에 대한 n 열의 묵시적 정의입니다. 여기서 n은 select-statement로부터 나온 결과 열의 수입니다. 내재적 정의에는(해당 자료 유형에 적용할 수 있는 경우) 다음과 같은 n 열의 속성이 포함됩니다.

- 열 이름 (및 시스템 열 이름)
- 자료 유형, 길이, 정밀도 및 스케일
- CCSID
- 널 가능성
- 열 머리말 및 텍스트(717 페이지의 『LABEL』 참조)

다음 속성은 포함되지 않습니다(*copy-options*를 사용하여 디폴트 값 및 ID 속성을 포함시킬 수 있습니다).

- 디폴트 값
- ID 속성

내재적 정의에는 식별된 표나 뷰의 다른 선택적 속성을 포함하지 않습니다. 예를 들어 새로운 표는 표의 1차 키나 외부 키를 자동으로 포함하지 않습니다. 선택적 절이 명시적으로 지정된 경우에만 새로운 표가 이들 및 기타 선택적 속성을 갖게 됩니다.

표의 묵시적으로 정의된 열은 *select-statement*의 결과 표로부터 열의 이름을 계승합니다. 따라서, 열 이름은 *select-statement* 또는 모든 열과 열의 열 이름 리스트에 지정되어야 합니다. 표현식, 상수, 함수에서 도출된 결과 열의 경우, *select-statement*는 결과 열 바로 다음에 AS column-name절을 포함하거나 열 리스트에서 *select-statement* 앞에 이름이 지정되어야 합니다.

*select-statement*는 호스트 변수를 참조하거나 매개변수 마커(의문 부호)를 참조할 수 없습니다.

WITH DATA

*select-statement*이 실행됨을 지정합니다. 표가 작성된 후, *select-statement*의 결과 표 행은 표에 자동으로 삽입됩니다.

WITH NO DATA 또는 DEFINITION ONLY

*select-statement*이 실행되지 않음을 지정합니다. 따라서, 자동으로 표를 채울 행 세트를 가진 결과 표가 없습니다.

copy-options

INCLUDING IDENTITY COLUMN ATTRIBUTES

표가 *select-statement*, *table-name* 또는 *view-name*에 기인한 열의 ID 속성(있는 경우)을 계승하도록 지정합니다. 일반적으로, 표, 보기 또는 select-statement의 해당 열의 요소가 기본 표 열을 ID 속성과 직접/간접적으로 맵핑하는 보기 열의 이름 또는 표 열의 이름인 경우, ID 열이 복사됩니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES절이 AS *select-statement* 절로 지정된 경우, 다음과 같은 경우 새 표의 열은 ID 속성을 계승하지 않습니다.

- *select-statement*의 선택 리스트에는 ID 열 이름의 복수 인스턴스(즉, 동일한 이름을 두 번 이상 선택)가 포함됩니다.
- *select-statement*의 선택 리스트에는 복수 ID 열이 포함됩니다(즉, 결합 관련).
- ID 열은 선택 리스트에서 표현식에 포함됩니다.
- *select-statement*는 집합 연산(결합)을 포함합니다.

INCLUDING IDENTITY가 지정되지 않은 경우, 표에는 ID 열이 없습니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES

표가 *select-statement, table-name* 또는 *view-name*에 기인한 열의 ID 속성(있는 경우)을 계승하지 않도록 지정합니다.

INCLUDING COLUMN DEFAULTS

표가 *select-statement, table-name* 또는 *view-name*에 기인한 열의 디폴트 값을 계승하도록 지정합니다. 디폴트 값은 INSERT에 값이 지정되지 않을 때 열에 지정되는 값입니다.

USING TYPE DEFAULTS를 지정한 경우, INCLUDING COLUMN DEFAULTS를 지정하지 마십시오.

INCLUDING COLUMN DEFAULTS가 지정되지 않은 경우, 디폴트 값은 계승되지 않습니다.

EXCLUDING COLUMN DEFAULTS

표가 *select-statement, table-name* 또는 *view-name*에 기인한 열의 디폴트 값을 계승하지 않도록 지정합니다.

USING TYPE DEFAULTS

표의 디폴트 값이 *select-statement, table-name* 또는 *view-name*에 기인한 열의 자료 유형에 기초하도록 지정합니다. 열이 널이면 디폴트 값은 널값입니다. 반면에 디폴트 값는 다음과 같습니다.

자료 유형	디폴트 값
숫자	0
고정 길이 스트링	공백
가변 길이 스트링	길이가 0인 스트링
날짜	INSERT 시의 현재 날짜
시간	INSERT 시의 현재 시간
시간소인	INSERT 시의 현재 시간소인
자료 링크	DLVALUE(‘ ’,‘URL’,‘’)에 해당하는 값
distinct-type	고유한 유형의 소스 유형에 대응하는 디폴트 값

INCLUDING COLUMN DEFAULTS가 지정된 경우 USING TYPE DEFAULTS를 지정하지 마십시오.

unique-constraint

CONSTRAINT*constraint-name*

제한사항을 명명합니다. *constraint-name*은 CREATE TABLE문에서 이미 지정되었고 이미 현재 서버에 있는 제한사항을 식별해서는 안됩니다.

구가 지정되지 않으면 고유 제한 이름은 데이터베이스 관리자에 의해 생성됩니다.

PRIMARY KEY(*column-name*,...)

식별된 열로 구성된 1차 키를 정의합니다. 표는 하나의 1차 키만 가질 수 있습니다. 따라서, 이 절은 한 번 이상 지정될 수 없고, 표에 대한 1차 키를 정의하기 위해 축약 양식이 사용되는 경우에는 절대로 지정될 수 없습니다. 식별된 열은 CREATE TABLE문에 이미 지정된 다른 UNIQUE 제한사항에 지정된 열과 같을 수 없습니다. 예를 들어, UNIQUE(B,A)가 이미 지정되어 있으면 PRIMARY KEY(A,B)는 허용되지 않습니다.

각 *column-name*은 표의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 바이트 계수의 합계는 2000-n을 초과해서는 안됩니다. 여기서 n은 널(null)을 허용하는 지정된 열의 수입니다. byte-counts에 대한 자세한 정보는 570 페이지의 표 47을 참조하십시오.

고유 색인은 분리된 시스템 논리 파일이 아닌 시스템 실제 파일의 일부로 작성됩니다. 1차 키가 추가되면 NULL 값이 1차 키를 구성하는 열에 허용되지 않는 규칙을 강제하기 위해 CHECK 제한이 내재적으로 추가됩니다.

UNIQUE(*column-name*,...)

식별된 열로 구성된 고유 키를 정의합니다. UNIQUE절은 한 번 이상 지정될 수 있습니다. 식별된 열은 CREATE TABLE문에 이미 지정된 다른 PRIMARY KEY나 UNIQUE 제한사항에 지정된 열과 같을 수 없습니다. 판별을 위해 고유 제한이 다른 제한 스펙과 같은 경우 열 리스트가 비교됩니다. 예를 들면, UNIQUE(A,B)는 UNIQUE(B,A)와 같습니다.

각 column-name은 표의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 바이트 계수의 합계는 2000-n을 초과해서는 안됩니다. 여기서 n은 널(null)을 허용하는 지정된 열의 수입니다. byte-counts에 대한 자세한 정보는 570 페이지의 표 47을 참조하십시오.

식별된 열의 고유 색인은 CREATE TABLE문이 실행되는 동안 작성됩니다. 고유 색인은 분리된 시스템 논리 파일이 아닌 시스템 실제 파일의 일부로 작성됩니다.

referential-constraint

CONSTRAINT*constraint-name*

제한사항을 명명합니다. *constraint-name*은 CREATE TABLE문에서 이미 지정되었고 이미 현재 서버에 있는 제한사항을 식별해서는 안됩니다.

구가 지정되지 않으면 고유 제한 이름은 데이터베이스 관리자에 의해 생성됩니다.

FOREIGN KEY

FOREIGN KEY절의 각 스펙은 참조 제한을 정의합니다.

(*column-name***,...)**

참조 제한의 외부 키는 식별된 열로 구성됩니다. 각 *column-name*은 표의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 길이의 합계는 2000-n을 초과해서는 안됩니다. 여기서 n은 널(null)을 허용하는 지정된 열의 수입니다.

REFERENCES *table-name*

REFERENCES절에 지정된 *table-name*은 작성되는 표나 서버에 이미 존재하는 기본 표를 식별해야 합니다. 그러나 카탈로그 표 또는 전역 임시 표를 식별해서는 안됩니다.

외부 키, 상위 키 및 상위 표가 이전에 지정된 참조 제한사항의 외부 키, 상위 키 및 상위 표와 같으면 참조 제한은 *duplicate*입니다. 중복 참조 제한은 허용되지만 권장되지는 않습니다.

T2는 식별된 상위 표를 나타내고 T1은 작성되는 표를 나타냅니다.

지정된 외부 키는 T2의 상위 키와 같은 열 수를 가져야 합니다. 외부 키의 *n* 번째 열과 상위 키의 *n*번째 열의 설명은 동일한 자료 유형과 길이를 가져야 합니다.

(*column-name***,...)**

참조 제한의 상위 키는 식별된 열로 구성됩니다. 각 *column-name*은 T2의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 열은 LOB 또는 DATALINK 열이어서는 안됩니다. 식별된 열의 수는 120을 초과해서는 안되고 바이트 계수의 합계는 2000-n을 초과해서는 안됩니다. 여기서 n은 널(null)을 허용하는 지정된 열의 수입니다. byte-counts에 대한 자세한 정보는 570 페이지의 표 47을 참조하십시오.

열 이름 리스트는 T2의 1차 키나 T2에 있는 UNIQUE 제한사항에 있는 열 이름 리스트와 동일해야 합니다. 이름은 1차 키와 같은 순서로 지정될 필요는 없지만, *foreign key*절에 있는 열 리스트의 대응하는 순서로 지정되어야 합니다. 열 이름 리스트가 지정되지 않으면 T2는 1차 키를 가져야 합니다. 열 이름 리스트의 생략은 해당 1차 키 열의 내재적 스펙입니다.

FOREIGN KEY절로 지정한 참조 제한은 T2가 상위이고 T1이 종속인 관계를 정의합니다.

ON DELETE

상위 표의 행이 삭제될 때 종속 표에 취할 조치를 지정합니다. 다섯 개의 조치가 있습니다.

- NO ACTION(디폴트)
- RESTRICT

- CASCADE
- SET NULL
- SET DEFAULT

외부 키의 일부 열이 널값을 허용하지 않으면 SET NULL은 지정되어서는 안 됩니다.

T1에 FILE LINK CONTROL을 갖는 DataLink 열이 있으면 CASCADE를 지정해서는 안됩니다.

삭제 규칙은 T2의 행이 DELETE의 오브젝트나 전파된 삭제 조작일 때 적용되며 그 행은 T1에 하위 행을 갖습니다. *p*는 T2의 그런 행을 나타냅니다.

- RESTRICT나 NO ACTION이 지정되면 오류가 발생하고 어떤 행도 삭제되지 않습니다.
- CASCADE가 지정되면 삭제 조작은 T1에 있는 *p*의 하위 행에 전파됩니다.
- SET NULL이 지정되면 T1에 있는 *p*의 각 하위 행인 외부 키를 가진 널 가능 열이 널(null)로 설정됩니다.
- SET DEFAULT가 지정되면 T1에 있는 *p*의 각 하위 행인 외부 키를 가진 각 열이 디폴트 값으로 설정됩니다.

ON UPDATE

상위 표의 행이 갱신될 때 종속 표에 취할 조치를 지정합니다.

갱신 규칙은 T2의 행이 UPDATE의 오브젝트나 전파된 삭제 조작일 때 적용되며 그 행은 T1에 하위 행을 갖습니다. *p*는 T2의 그런 행을 나타냅니다.

- RESTRICT나 NO ACTION이 지정되면 오류가 발생하고 어떤 행도 갱신되지 않습니다.

check-constraint

CONSTRAINT *constraint-name*

검사 제한사항을 명명합니다. *constraint-name*은 CREATE TABLE문에서 이미 지정되었고 이미 현재 서버에 있는 제한사항을 식별해서는 안됩니다.

구가 지정되지 않으면 고유 제한 이름은 데이터베이스 관리자에 의해 생성됩니다.

CHECK(*check-condition***)**

검사 제한사항을 정의합니다. *check-condition*은 항상 표의 모든 행에 대해 참이거나 값이 지정되지 않아야 합니다.

*check-condition*은 다음 경우를 제외하고는 *search-condition*입니다.

- 표의 열만 참조할 수 있습니다.
- FILE LINK CONTROL 열의 ROWID 또는 DATALINK를 참조할 수 없습니다.

- 다음 사항을 포함해서는 안됩니다.
  - 부속 조회
  - Scalar-subselect
  - 열 함수
  - 호스트 변수
  - 매개변수 마커
  - LOB가 들어 잇는 복합 표현식(연결 등)
  - CURRENT TIMEZONE, CURRENT SCHEMA, CURRENT SERVER, CURRENT PATH, 및 USER 특수 레지스터
  - NOW, CURDATE, 및 CURTIME 스칼라 함수
  - NODENAME 스칼라 함수
  - 고유한 유형의 작성으로 내재적으로 생성된 사용자 정의 기능
  - ATAN2, DIFFERENCE, RADIANS, RAND 및 SOUNDEX 스칼라 함수
  - DLVALUE, DLURLPATH, DLURLPATHONLY, DLURLSERVER 또는 DLURLSCHEME 스칼라 함수
  - DLURLCOMPLETE 스칼라 함수(FILE LINK CONTROL 및 READ PERMISSION DB 속성을 갖는 자료 링크의 경우)

search-condition에 대한 자세한 정보는 154 페이지의 『탐색 조건』을 참조하십시오. LOB 자료 유형 및 표현식을 포함하는 체크 제한조건에 대한 자세한 내용은 Database Programming 책을 참조하십시오.

nodegroup절

IN *nodegroup-name*

표에 있는 자료가 분할될 노드 그룹을 지정합니다. 이름은 현재 서버에 있는 노드 그룹을 식별해야 합니다. 이 절이 지정되면 표는 노드 그룹에 있는 모든 시스템에 분배된 표로 작성됩니다.

LOB나 DATALINK 열은 분배된 표에서 사용될 수 없습니다.

DB2 Multisystem 제품은 분배된 표를 작성하기 위해 설치되어야 합니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

PARTITIONING KEY(*column-name***,...)**

분할 키를 지정합니다. 분할 키는 노드 그룹의 어떤 노드에 행이 위치할지 판별하기 위해 사용됩니다. 각 *column-name*은 표의 열을 식별하는 규정되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. PARTITIONING KEY절이 지정되지 않으면 1차 키의 첫 번째 열이 분할 키로 사용됩니다. 1차 키가 없으면, 부동 소수점, 날짜, 시간 또는 시간소인이 아닌 표의 첫 번째 열이 분할 키로 사용됩니다.

분할 키를 구성하는 열은 표에 대한 고유 제한을 구성하는 열의 서브세트여야 합니다. 부동 소수점, 날짜, 시간 및 시간소인 열은 분할 키에 사용될 수 없습니다.

USING HASHING

행을 노드 그룹의 적합한 서버에 분배하기 위해 분할 키의 자료가 해시되도록 지정합니다.

주

표 속성: 표는 실제 파일로 작성됩니다. 표가 작성되면 파일 대기 시간과 레코드 대기 시간 속성은 실제 파일 작성(CRTLF) 명령의 WAITFILE 및 WAITRCD 키워드에 지정된 디폴트 값으로 설정됩니다.

SQL 표는 삭제된 행에 의해 사용된 공간이 앞으로의 삽입 요구에 의해 재생되도록 작성됩니다. 이 속성은 *CHGPF* 명령 및 *REUSEDLT(*NO)* 매개변수 지정으로 변경될 수 있습니다. *CHGPF* 명령에 대한 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주의 CL 참조 정보를 참조하십시오.

표가 작성될 때 저널링은 스키마의 QSQJRN 저널에서 자동으로 시작됩니다.

분배된 표는 표가 분배되는 서버 전체에서 작성됩니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

표 소유권: SQL명이 지정된 경우, 표의 *소유자*는 표가 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 표의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 표의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

표 권한: SQL 이름이 사용되면 표는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 표는 *PUBLIC에 대한 권한으로 작성됩니다.

표의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 표에 대한 권한도 갖습니다.

최대 행 크기

*column-definition*의 설명에 참조되는 두 가지 최대 행 크기 제한사항이 있습니다.

- 최대 행 버퍼 크기는 32766이거나 VARCHAR, VARGRAPHIC 또는 LOB 열이 지정된 경우 32740입니다.
- LOB이 지정된 경우 최대 행 자료 크기는 3 758 096 383입니다. LOB가 지정되지 않으면 최대 행 자료 크기는 32766이거나 VARCHAR, VARGRAPHIC 또는 VARGRAPHIC 열이 지정된 경우 32740입니다.

행 자료 및/또는 행 버퍼의 길이를 판별하려면 자료 유형의 바이트 수를 기초로 한 행의 각 열의 대응하는 길이를 추가합니다.

다음 표는 널값을 허용하지 않는 열에 대한 자료 유형에 의한 열의 바이트 수입니다. 널값을 허용하는 열이 있으면, 8개의 열마다 1바이트가 필요합니다.

표 47. 자료 유형별 열 바이트 계수

자료 유형	행 버퍼 바이트 수	행 자료 바이트 수
SMALLINT	2	2
INTEGER	4	4
BIGINT	8	8
DECIMAL(*p*, *s*)	(*p*/2) + 1의 정수 부분	(*p*/2) + 1의 정수 부분
NUMERIC(*p*, *s*)	*p*	*p*
FLOAT(단정밀도)	4	4
FLOAT(배정밀도)	8	8
CHAR(*n*)	*n*	*n*
VARCHAR(*n*)	*n*+2	*n*+2
CLOB(*n*)	29+*pad*	*n*+29
GRAPHIC(*n*)	*n**2	*n**2
VARGRAPHIC(n)	*n**2+2	*n**2+2
DBCLOB(*n*)	29+*pad*	*n**2+29
BLOB(*n*)	29+*pad*	*n*+29
DATE	10	4
TIME	8	3
TIMESTAMP	26	10
DATALINK(*n*)	*n*+24	*n*+24
ROWID	42	28
distinct-type	소스 유형에 대한 바이트 수	소스 유형에 대한 바이트 수
주: *pad*는 경계에 맞추기 위해 필요한 1부터 15까지의 값입니다.		

데이터베이스에 대해 설명된 정밀도

- 부동 소수점 필드는 비트 정밀도가 아닌 십진 정밀도가 있는 iSeries 데이터베이스에서 정의됩니다. 비트 수를 십진수로 변환하는 데 사용되는 알고리즘은 *decimal precision = CEILING(n/3.31)*이며, 여기서 *n*은 변환할 비트 수입니다. 십진 정밀도는 대화식 SQL을 사용하여 표시할 자릿수를 판별하기 위해 사용됩니다.
- SMALLINT 필드는 4,0의 십진 정밀도로 저장됩니다.
- INTEGER 필드는 9,0의 십진 정밀도로 저장됩니다.
- BIGINT 필드는 19,0의 십진 정밀도로 저장됩니다.

LONG VARCHAR 및 LONG VARGRAPHIC

표준이 아닌 LONG VARCHAR 및 LONG VARGRAPHIC 구문은 지원되지 않으며 폐기되었습니다. VARCHAR(integer) 및 VARGRAPHIC(integer)의 대체 표준 구문이 우선적입니다. VARCHAR(integer) 및 VARGRAPHIC(integer)가 권장됩니다. CREATE TABLE문이 처리된 후 데이터베이스 관리자는 LONG VARCHAR 열을 VARCHAR로 또 LONG VARGRAPHIC 열을 VARGRAPHIC으로 간주합니다. 최대 길이는 이전되지 않는 제품에 따라 고유한 방식으로 계산됩니다.

LONG VARCHAR[52]
행에서 사용할 수 있는 공간의 양으로 최대 길이가 판별되는 가변 길이 문자 스트링의 경우.

LONG VARGRAPHIC[52]
최대 길이가 행에서 사용할 수 있는 공간의 양으로 판별되는 가변 길이 그래픽 스트링의 경우.

LONG 열의 최대 길이는 다음과 같이 판별됩니다. 다음과 같이 가정합니다.

- m은 최대 행 *크기*입니다.
- i는 LONG VARCHAR나 LONG VARGRAPHIC이 아닌 표의 모든 열의 바이트 수의 합계입니다.
- j는 표에 있는 LONG VARCHAR 및 LONG VARGRAPHIC 열의 수입니다.
- k는 널(null)을 허용하는 행의 열 수입니다.

각각의 LONG VARCHAR 열의 길이는 INTEGER((m-24-i-((k+7)/8))/j)입니다.

LONG VARCHAR 열의 길이를 계산하고 그 길이를 2로 나누어서 각 LONG VARGRAPHIC 열의 길이를 판별합니다. 결과의 정수 부분이 길이입니다.

ID 열 사용

표에 ID 열이 있으면, 데이터베이스 관리자는 표에 행이 삽입될 때 열에 대한 순번 값을 자동으로 생성할 수 있습니다. 따라서, ID 열은 1차 키에 적절합니다. ID 열과 ROWID 열은 두 열 유형 모두 데이터베이스 관리자가 생성한 값을 포함한다는 점에서 유사합니다. ROWID 열은 direct-row 액세스에서 유용합니다. ROWID 열에는 ROWID 자료 유형 값이 들어 있으며 이 값은 일반적으로 오름차순이나 내림차순인 아닌 40바이트 VARCHAR 값을 리턴합니다. 따라서 ROWID 자료 값은 사원 번호나 제품 번호를 생성하는 등의 여러 어플리케이션 용도로 적합하지 않습니다. 직접 행 액세스를 필요로 하는 않는 자료의 경우 일반적으로 ID 열이 적합한 접근법입니다. ID 열에는 기존 숫자 자료 유형이 들어 있으며 ROWID 값이 적절하지 않은 경우 광범위하게 사용될 수 있습니다.

표가 특정 시점(point-in-time)으로 복구된 경우(RMVJRNCHG를 사용하여), ID 열에 대해 생성된 값은 순서에 커다란 편차가 발생할 수 있습니다. 예를 들어, 표에 증가값 1을 가진 ID 열이 있으며 T1시에 마지막으로 생성된 값이 100이며 데이터베이스 관리

자가 이어서 최대 1000까지의 값을 생성한다고 가정할 경우, 표가 T1으로 회복된다면 복구가 완료된 후 삽입되는 다음 행의 생성된 ID 열 값은 1001로서 ID 열의 값에 100 - 1001의 편차가 생깁니다.

CYCLE이 지정되면, 열에 고유 제한조건 또는 고유 색인이 정의되어 있지 않은 경우 열이 GENERATED ALWAYS인 경우에도 중복된 열 값이 생성될 수 있습니다.

시스템명 생성 규칙

시스템이 시스템 표, 뷰, 색인이나 열 이름을 생성할 때 특정 인스턴스가 있습니다. 이러한 인스턴스와 이름 생성 규칙은 다음 섹션에서 설명됩니다.

열 이름 생성 규칙

표나 뷰가 작성되고 column-name이 유효한 system-column-name이 아닐 때 system-column-name이 지정되지 않으면 system-column-name이 생성됩니다.

column-name에 특수 문자가 들어 있지 않고 10자보다 길면, 10자의 system-column-name이 다음과 같이 생성됩니다.

- 이름의 처음 5자는 문자
- 5자리의 고유 숫자

예를 들면, 다음과 같습니다.

```
The system-column-name for LONGCOLUMNNAME would be LONGC00001
```

열 이름이 구분된 경우

- 분리 문자 내의 처음 5자는 system-column-name의 처음 5자로 사용됩니다. 분리 문자 내의 문자가 5자 미만이면 이름의 오른쪽이 밑줄(_) 문자로 채워집니다. 소문자는 대문자로 처리(fold)됩니다. system-column-name에서 유효한 문자는 A-Z, 0-9, @, #, $ 및 _입니다. 다른 문자는 밑줄(_) 문자로 변경됩니다. 첫 문자가 밑줄이 되면 Q자로 변경됩니다.
- 5 자리의 고유 숫자가 5 문자 뒤에 붙습니다.

예를 들면, 다음과 같습니다.

```
The system-column-name for "abc" would be ABC__00001
The system-column-name for "COL2.NAME" would be COL2_00001
The system-column-name for "C 3" would be C_3__00001
The system-column-name for "??" would be Q____00001
The system-column-name for "*column1" would be QCOLU00001
```

표 이름 생성 규칙

표, 뷰, 별명 또는 색인이 다음과 같이 작성될 때 시스템명이 생성됩니다.

- 10문자보다 긴 이름
- 시스템명에 유효하지 않은 문자가 포함된 이름

파일이 일단 작성되면 파일에 액세스하기 위해 SQL 이름이나 대응하는 시스템명이 둘 다 SQL문에 사용될 수 있습니다. 그러나, SQL 이름은 iSeries용 DB2 UDB에 의해서만 인식되며, 시스템명은 다른 환경에서 사용되어야 합니다.

이름에 특수 문자가 들어 있지 않고 10자보다 길면, 10자의 시스템명이 다음과 같이 생성됩니다.

- 이름의 처음 5자는 문자
- 5자리의 고유 숫자

예를 들면, 다음과 같습니다.

```
The system name for LONGTABLENAME would be LONGT00001
```

SQL 이름에 특수 문자가 들어 있으면 시스템명은 다음과 같이 생성됩니다.

- 이름의 처음 4자는 문자
- 4 자리의 고유 숫자

또한

- 모든 특수 문자는 밑줄(_)로 대체됩니다.
- 후행 공백은 이름에서 제거됩니다.
- 유효한 시스템명이 되기 위해 분리 문자가 필요하면 이름은 큰 따옴표(″)로 분리됩니다.

예를 들면, 다음과 같습니다.

```
The system name for "??" would be "__0001"
The system name for "longtablename" would be "long0001"
The system name for "LONGTableName" would be LONG0001
The system name for "A b   " would be "A_b0001"
```

SQL은 교차 참조 파일을 찾아서 시스템명이 고유함을 확인합니다. 이름이 교차 참조 파일에 이미 있으면, 이름이 중복되지 않을 때까지 숫자가 증가됩니다.

위의 규칙을 사용하여 고유한 이름을 판별할 수 없다면 추가 문자가 이름의 카운터에 추가되어 고유한 이름을 찾거나 범위가 다할 때까지 숫자가 증가합니다. 예를 들어 ″longtablename″을 작성하고 ″long0001″에서 ″long9999″까지가 이미 있는 이름이면 이름은 ″lon00001″이 될 것입니다.

예

예 1

관리 권한을 가지고 있다고 가정하고, 다음 열을 갖는 ‘ROSSITER.INVENTORY’라는 이름의 표를 작성합니다.

- 파트 번호: Small integer(널은 허용되지 않음)

- 설명: 길이가 0부터 24인 문자, 널 허용
- 양: 정수, 널 허용

```
CREATE TABLE ROSSITER.INVENTORY
             (PARTNO         SMALLINT     NOT NULL,
              DESCR          VARCHAR(24 ),
              QONHAND        INT )
```

예 2

다음 열을 갖는 DEPARTMENT라는 표를 작성합니다.

- 부서 번호: 길이가 3인 문자, 널이 되어서는 안됩니다.
- 부서 이름: 길이가 0부터 36인 문자, 널이 되어서는 안됩니다.
- 관리자 번호: 길이가 6인 문자, 널 허용
- 관리 부서: 길이가 3인 문자, 널이 되어서는 안됩니다.
- 위치: 길이가 16인 문자, 널 허용

```
CREATE TABLE DEPARTMENT
             (DEPTNO    CHAR(3)     NOT NULL,
              DEPTNAME  VARCHAR(36) NOT NULL,
              MGRNO     CHAR(6),
              ADMRDEPT  CHAR(3)     NOT NULL,
              LOCATION  CHAR(16),
              PRIMARY KEY(DEPTNO) )
```

예 3

뷰 PRJ_LEADER의 열과 동일한 열 정의를 가진 표 REORG_PROJECTS를 작성하십시오.

```
CREATE TABLE REORG_PROJECTS
  LIKE PRJ_LEADER
```

CREATE TRIGGER

CREATE TRIGGER문은 현재 서버에서 트리거를 정의합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표에 대한 권한 및 트리거명 규정자:
  - 주제 표가 들어 있는 라이브러리에 대한 *EXECUTE
  - 주제 표에 대한 ALTER (*OBJALTER) 또는 WITH GRANT OPTION privilege(*OBJMGT)
  - 주제 표에 대한 SELECT(*OBJOPR 및 *READ)
  - BEFORE UPDATE 트리거에 NEW 상관 변수를 수정하는 SET 명령문이 들어 있는 경우 주제 표에 대한 UPDATE (*UPD 및 *OBJOPR)
  - 트리거명 라이브러리에 대한 SELECT(*OBJOPR 및 *READ) 및 INSERT(*OBJOPR 및 *ADD) 권한
  - 실제 파일 트리거(ADDPFTRG) 명령에 대한 *USE
  - SQL 명명이 유효하고 트리거명의 스키마 규정자와 일치하는 사용자 프로파일이 존재하며 그 이름이 명령문의 권한부여 ID와 다를 때 *ALLOBJ 및 *SECADM 특수 권한이 필요합니다.
- 관리 권한

명령문의 권한부여 ID는 다음과 같은 경우에 표에 대한 ALTER 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대한 ALTER 권한을 부여받은 경우
- 표에 대하여 *OBJALTER 또는 *OBJMGT 중 하나의 시스템 권한을 부여받은 경우

명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 표에서 UPDATE 권한을 갖습니다.

- 표의 소유자인 경우

- 표나 열에 대해 UPDATE 권한을 부여받은 경우
- 표에 대해 *OBJOPR과 *UPD의 시스템 권한을 부여받은 경우

또한 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

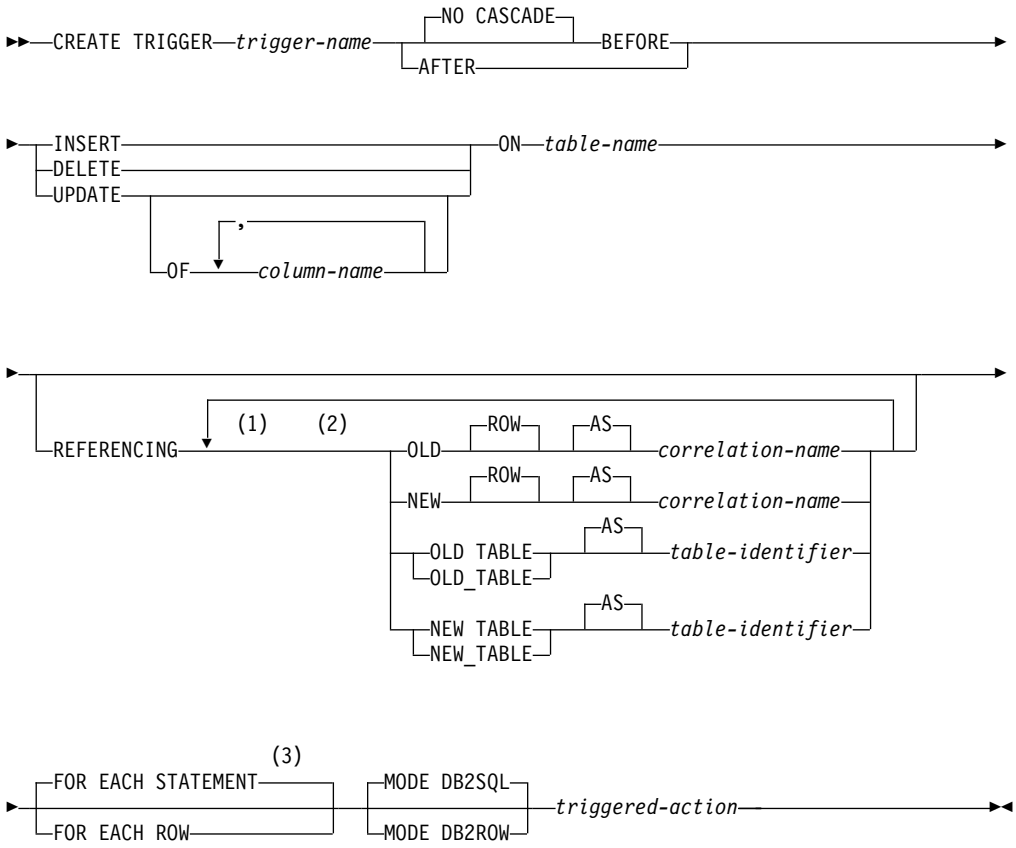
- 다음과 같은 시스템 권한
  - 프로그램 작성(CRTPGM) 명령에서 *USE
- 관리 권한

SQL 이름이 지정되고, 트리거가 작성되는 라이브러리와 같은 이름을 갖는 사용자 프로파일이 있으며, 그 이름이 명령문의 권한부여 ID와 다른 경우 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- *ALLOBJ 및 *SECADM 특수 권한
- 관리 권한

*SQL-trigger-body*의 SQL문에서 식별되는 각 표나 뷰의 경우 권한부여 ID가 보유한 권한에는 해당 SQL문을 수행하는 데 필요한 권한이 포함되어야 합니다.

구문



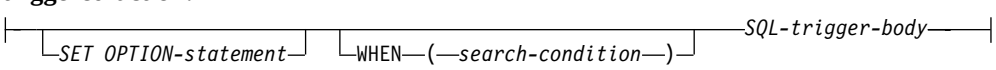
주:

1 같은 절을 두 번 이상 지정할 수 없습니다.

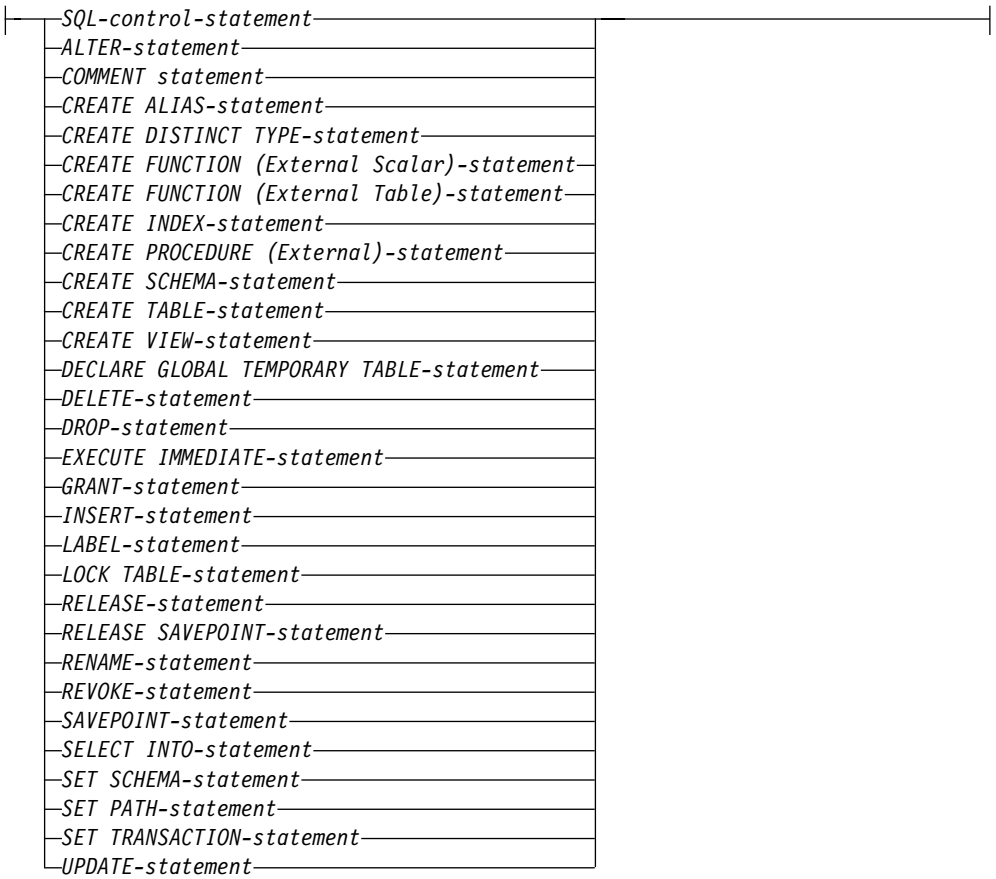
2 OLD TABLE과 NEW TABLE은 AFTER 트리거에 대해서만 각각 한 번씩 지정할 수 있습니다.

3 FOR EACH STATEMENT는 BEFORE 트리거에는 지정할 수 없습니다.

triggered-action:



SQL-trigger-body:



설명

trigger-name

트리거의 이름을 지정합니다. 내재적 또는 명시적 규정자를 가지고 있는 이름은 이미 현재 서버에 있는 트리거와 같을 수 없습니다. QTEMP는 *trigger-name* 스키마 규정자로 사용할 수 없습니다.

SQL명이 지정되면 트리거는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템명이 지정되면 트리거는 규정자에 의해 지정된 스키마에 작성됩니다. 규정되지 않은 경우 트리거는 주제 표와 같은 스키마에 작성됩니다.

트리거명이 유효한 시스템명이 아니거나 같은 이름을 갖는 프로그램이 이미 존재하면 데이터베이스 관리자는 시스템명을 생성합니다. 이름 생성 규칙에 대한 내용은 572 페이지의 『표 이름 생성 규칙』을 참조하십시오.

NO CASCADE

NO CASCADE는 다른 제품과의 호환을 위해 사용되며 iSeries용 DB2 UDB에서는 사용하지 않습니다.

BEFORE

트리거가 이전 트리거(before trigger)임을 지정합니다. 데이터베이스 관리자는 주제 표에서 삽입, 삭제 또는 갱신 조작에 의해 발생된 변경사항을 적용하기 전에 *triggered-action*을 실행합니다. 또한 이전 트리거의 *triggered-action*에 어떤 갱신도 포함할 수 없으므로 *triggered-action*이 다른 트리거를 활성화시키지 않음을 지정합니다.

AFTER

트리거가 이후 트리거(after trigger)임을 지정합니다. 데이터베이스 관리자는 주제 표에서 삽입, 삭제 또는 갱신 조작에 의해 발생된 변경사항을 적용한 후에 *triggered-action*을 실행합니다.

INSERT

트리거가 삽입 트리거임을 지정합니다. 데이터베이스 관리자는 주제 표에서 삽입 조작이 있을 때마다 *triggered-action*을 실행합니다.

DELETE

트리거가 삭제 트리거임을 지정합니다. 데이터베이스 관리자는 주제 표에서 삭제 조작이 있을 때마다 *triggered-action*을 실행합니다.

UPDATE

트리거가 갱신 트리거임을 지정합니다. 데이터베이스 관리자는 주제 표에서 갱신 조작이 있을 때마다 *triggered-action*을 실행합니다.

명시적인 *column-name* 리스트가 지정되지 않은 경우 ALTER TABLE문을 사용하여 나중에 추가된 열을 포함한 주제 표의 모든 열에 대한 갱신 조작은 *triggered-action*을 활성화합니다.

OF *column-name*, **...**

지정된 각각의 *column-name* 은 주제 표의 열이어야 하며 리스트에 단 한번 나와야 합니다. 나열된 모든 열에서의 갱신 조작은 *triggered-action*을 활성화합니다.

ON *table-name*

트리거 정의 주제 표를 식별합니다. table-name은 현재 서버에 있는 기본 표를 식별하지만 카탈로그 표, QTEMP의 표나 글로벌 임시표를 식별해서는 안됩니다.

REFERENCING

*전이 변수*에 대한 상관명 및 *전이 표*에 대한 표 이름을 지정합니다.

*Correlation-name*은 SQL 조작을 트리거함으로써 영향을 받는 행 집합에서 특정 행을 식별합니다. *Table-identifier*는 영향을 받는 행의 전체 집합을 식별합니다.

SQL 조작을 트리거함으로써 영향을 받는 각 행은 다음과 같이 지정된 *correlation-name*으로 열을 규정하여 *triggered-action*에서 사용할 수 있습니다.

OLD ROW AS *correlation-name*
SQL 조작을 트리거하기 전에 행의 값을 식별하는 상관명을 지정합니다.

NEW ROW AS *correlation-name*
SQL 조작 트리거 및 이미 실행된 BEFORE 트리거의 SET 명령문에 의해 수정된 행의 값을 식별하는 상관명을 지정합니다.

SQL 조작을 트리거함으로써 영향을 받는 전체 행 집합은 다음과 같이 지정된 임시표 이름을 사용하여 *triggered-action*에서 사용할 수 있습니다.

OLD TABLE AS *table-identifier*
SQL 조작을 트리거하기 전에 영향을 받는 행 전체 집합의 값을 식별하는 임시표의 이름을 지정합니다. 현재 트리거 활성화가 트리거의 *SQL-trigger-body*에 의해 발생된 경우 OLD TABLE에는 트리거에 의해 영향을 받은 행이 포함됩니다.

NEW TABLE AS *table-identifier*
SQL 조작 트리거 및 이미 실행된 BEFORE 트리거의 SET 명령문에 의해 수정된 영향을 받는 행 전체 집합의 상태를 식별하는 임시표의 이름을 지정합니다.

트리거 정의에는 최대 2개의 상관명 OLD ROW와 NEW ROW 그리고 2개의 표 이름 OLD TABLE과 NEW TABLE을 포함할 수 있습니다. 모든 이름은 서로 고유해야 합니다.

OLD ROW *correlation-name* 및 OLD TABLE *table-identifier*는 트리거하는 이벤트가 DELETE 조작이거나 UPDATE 조작인 경우에만 유효합니다. DELETE 조작의 경우 OLD ROW *correlation-name*에 삭제된 행의 열 값이 들어가고, OLD TABLE *table-identifier*에는 삭제된 행 집합의 값들이 들어갑니다. UPDATE 조작의 경우 OLD ROW *correlation-name*에는 UPDATE 조작이 수행되기 전 행의 열 값이 들어가고 OLD TABLE *table-identifier*에는 갱신된 행 집합의 값이 들어갑니다.

NEW ROW *correlation-name* 및 NEW TABLE *table-identifier*는 트리거하는 이벤트가 INSERT 또는 UPDATE 조작인 경우에만 유효합니다. 두 조작의 경우 모두 NEW ROW *correlation-name*에는 삽입되거나 갱신된 행의 열 값이 들어가고 NEW TABLE *table-identifier*에는 삽입되거나 갱신된 행 집합의 값들이 들어갑니다. BEFORE 트리거의 경우 갱신된 행 값에는 BEFORE 트리거의 *triggered-action*의 SET문으로 발생된 변경사항이 포함됩니다.

OLD TABLE과 NEW TABLE은 BEFORE 트리거나 MODE DB2ROW에 대해서는 지정할 수 없습니다.

OLD ROW와 NEW ROW는 FOR EACH STATEMENT 트리거에 대해서는 지정할 수 없습니다.

OLD ROW와 NEW ROW *correlation-name* 변수는 AFTER 트리거에서는 수정할 수 없습니다.

아래 표에서는 상관 변수와 전이 표의 가능한 조합을 요약해서 보여줍니다.

단위: **FOR EACH ROW**

모드	활성화 시간	트리거하는 조작	가능한 상관 변수	가능한 전이 표
DB2ROW	BEFORE	DELETE	OLD	NONE
		INSERT	NEW	
		UPDATE	OLD, NEW	
	AFTER	DELETE	OLD	
		INSERT	NEW	
		UPDATE	OLD, NEW	
DB2SQL	BEFORE	DELETE	OLD	
		INSERT	NEW	
		UPDATE	OLD, NEW	
	AFTER	DELETE	OLD	OLD TABLE
		INSERT	NEW	NEW TABLE
		UPDATE	OLD, NEW	OLD TABLE, NEW TABLE

단위: **FOR EACH STATEMENT**

모드	활성화 시간	트리거하는 조작	가능한 상관 변수	가능한 전이 표
DB2SQL	AFTER	DELETE	NONE	OLD TABLE
		INSERT		NEW TABLE
		UPDATE		OLD TABLE, NEW TABLE

문자 자료 유형을 갖는 전이 변수는 주제 표의 열의 CCSID를 상속합니다. *triggered-action*를 실행하는 동안 전이 변수는 호스트 변수처럼 처리됩니다. 그러므로 문자 변환이 일어날 수 있습니다.

임시 전이 표는 읽기 전용입니다. 수정할 수 없습니다.

각 *correlation-name*의 범위와 각 *table-identifier*는 전체 트리거 정의입니다.

FOR EACH ROW

데이터베이스 관리자가 트리거하는 조작이 수정하는 주제 표의 각 행에 대하여 *triggered-action*을 실행하도록 지정합니다. 트리거하는 조작이 어떤 행도 수정하지 않은 경우 *triggered-action*은 실행되지 않습니다.

FOR EACH STATEMENT

데이터베이스 관리자가 트리거하는 조작에 대하여 *triggered-action*을 단 한번 실행하도록 지정합니다. UPDATE 또는 DELETE FOR EACH STATEMENT 트리거는 트리거하는 UPDATE나 DELETE문에 의해 어떠한 행도 영향을 받지 않을 때에도 활성화됩니다.

FOR EACH STATEMENT는 BEFORE 트리거에 대하여 지정할 수 없습니다.

FOR EACH STATEMENT는 MODE DB2ROW 트리거에 대하여 지정할 수 없습니다.

MODE DB2SQL

MODE DB2SQL 트리거는 모든 행 조작이 발생한 후에 활성화됩니다.

MODE DB2ROW

MODE DB2ROW 트리거는 각 행 조작시 활성화됩니다.

MODE DB2ROW는 BEFORE 및 AFTER 활성화 시간에 모두 유효합니다.

triggered-action

트리거가 활성화될 때 수행되는 조치를 지정합니다. *triggered-action*은 하나 이상의 SQL문과 이 명령문들의 실행 여부를 제어하는 선택적 조건으로 구성됩니다.

SET OPTION문

트리거 작성에 사용될 옵션을 지정합니다. 예를 들어 디버그를 할 수 있는 트리거를 작성하려면 다음 명령문이 포함됩니다.

```
SET OPTION DBGVIEW = *LIST
```

자세한 내용은 772 페이지의 『SET OPTION』을 참조하십시오.

옵션 CLOSQLCSR, CNULRQD, DFTRDBCOL, DYNDFTCOL 및 NAMING은 CREATE TRIGGER문에서 허용되지 않습니다.

OLD ROW나 NEW ROW가 지정된 경우 DATFMT, DATSEP, TIMFMT 및 TIMSEP 옵션은 사용할 수 없습니다.

WHEN (*search-condition***)**

참, 거짓 또는 알 수 없음으로 평가되는 조건을 지정합니다. 트리거된 SQL문은 *search-condition*이 참으로 평가될 경우에만 실행됩니다. WHEN절이 생략되면 연관된 SQL문은 항상 수행됩니다.

SQL-trigger-body

복합 명령문을 포함하여 단일 SQL문을 지정합니다. SQL 트리거 정의에 대한 자세한 정보는 821 페이지의 제 6 장『SQL 제어문』을 참조하십시오.

CONNECT, SET CONNECTION, RELEASE, DISCONNECT, COMMIT, ROLLBACK, SET TRANSACTION 및 SET RESULT문을 발행하는 프로시듀어에 대한 호출은 트리거의 *triggered-action*에서 허용되지 않습니다.

트리거가 BEFORE 트리거인 경우 *SQL-trigger-body*에는 INSERT, UPDATE, DELETE, ALTER TABLE, COMMENT, 모든 CREATE문, DROP, 모든 GRANT문, LABEL, RENAME 또는 모든 REVOKE문이 포함될 수 없습니다. SQL 자료를 수정하는 프로시듀어나 함수에 대한 참조를 포함해야 합니다.

UNDO 핸들러는 트리거에서 사용할 수 없습니다.

*triggered-action*에서 참조된 모든 표, 뷰, 별명, 고유한 유형, 사용자 정의 함수 및 프로시듀어는 트리거가 작성될 때 현재 서버에 있어야 합니다. 별명이 참조하는 표나 뷰도 트리거가 작성될 때 있어야 합니다. 여기에는 QTEMP 라이브러리의 오브젝트가 포함됩니다. QTEMP의 오브젝트가 *triggered-action*에서 참조될 수 있지만 이 오브젝트를 드롭(drop)한다고 해서 트리거가 드롭(drop)되지는 않습니다.

트리거가 작성될 때 *triggered-action*은 CREATE 트리거 명령문의 결과로 수정됩니다.

- 명명 모드는 SQL 명명으로 전환됩니다.
- 모든 규정되지 않은 오브젝트 참조는 명시적으로 규정됩니다.
- 모든 내재적 열 리스트(예를 들면 SELECT *, 열 리스트가 없는 INSERT, UPDATE SET ROW)는 실제 열 이름 리스트가 되도록 확장됩니다.

수정된 *triggered-action*은 카탈로그에 저장됩니다.

*triggered-action*의 명령문은 프로시듀어나 사용자 정의 함수가 다른 활성화 그룹에서 실행되는 경우 현재 서버 이외의 서버에 액세스할 수 있는 프로시듀어나 사용자 정의 함수를 호출할 수 있습니다.

주

트리거 활성화

삽입, 삭제 또는 갱신 조작만이 트리거를 활성화할 수 있습니다. 참조 제한의 결과 발생한 삭제 조작은 트리거를 활성화하지 않습니다. 따라서,

- DELETE 트리거 이벤트를 갖는 트리거는 ON DELETE CASCADE의 참조 제한을 갖는 표에 추가될 수 없습니다.
- UPDATE 트리거 이벤트를 갖는 트리거는 ON DELETE SET NULL이나 ON DELETE SET DEFAULT의 참조 제한을 갖는 표에 추가될 수 없습니다.

트리거 활성화는 *트리거 직렬*을 유발할 수 있습니다. 이것은 다른 트리거 또는 동일한 트리거를 다시 활성화시키는 SQL문을 실행하는 트리거를 활성화시킨 결과입니다. 트리거된 조치는 원래 수정의 결과로 갱신을 유발하고, 이것이 추가적인 트리거 활성화를 유발할 수 있습니다. 트리거 직렬을 사용하여 한 번의 삭제, 삽입 또는 갱신 명령문으로 데이터베이스에 대한 중요한 변경을 일으키는 트리거를 연속적으로 활성화할 수 있습니다. 직렬 레벨의 수는 200 또는 첫 작업이나 첫 프로세스에서 허용되는 기억장치의 최대 용량에 따라 제한됩니다.

제한사항을 시행하기 위해 트리거 추가

이미 행을 가지고 있는 표에 트리거를 추가한다고 해서 트리거된 조치가 실행되지 않습니다. 따라서 트리거가 표의 자료에 대한 제한사항을 시행하도록 설계되었다면 기존 행의 자료는 그 제한사항을 충족시킬 수 없을 것입니다.

복수 트리거

같은 트리거 SQL 조작과 활성화 시간을 갖는 복수의 트리거를 표에 정의할 수 있습니다. 트리거는 작성된 순서로 활성화됩니다. 예를 들어 맨 처음 작성된 트리거가 맨 처음 실행되고 두 번째로 작성된 트리거가 두 번째로 실행됩니다.

최대 300개의 트리거가 해당 소스 표에 추가될 수 있습니다.

주제 표 또는 트리거된 조치에서 참조된 표에 열 추가

트리거가 정의된 후 열이 주제 표에 추가된 경우 다음과 같은 규칙이 적용됩니다.

- 트리거가 명시적인 열 리스트 없이 정의된 UPDATE 트리거인 경우 새로운 열에 대한 갱신은 트리거 활성화를 유발합니다.
- *triggered-action*의 SQL문이 트리거하는 표를 참조하는 경우 트리거가 재작성될 때까지 SQL문은 새로운 열에 액세스할 수 없습니다.
- OLD_TABLE 및 NEW_TABLE 전이 표에는 새로운 열이 들어가지만 이 열은 트리거가 재작성되지 않는 한 참조될 수 없습니다.

열이 triggered-action의 SQL문에 의해 참조되는 표에 추가되는 경우 트리거가 재작성될 때까지 SQL문은 새로운 열에 액세스할 수 없습니다.

트리거된 조치에서 참조된 표를 재명명 또는 이동

*triggered-action*에서 참조된 모든 표(주제 표 포함)는 이동 또는 재명명될 수 있습니다. 그러나 *triggered-action*는 이전의 이름이나 스키마를 계속 참조할 것입니다. *triggered-action*이 실행될 때 참조된 표를 찾을 수 없으면 오류가 발생합니다. 따라서 트리거를 드롭(drop)한 후 다시 재작성하여 이름 변경 또는 이동된 표를 참조할 수 있도록 해야 합니다.

종속 오브젝트

트리거가 작성될 때, 참조된 모든 오브젝트가 존재해야 합니다.

날짜 시간 고려사항

OLD ROW 또는 NEW ROW가 지정된 경우 *triggered-action*의 SQL문에서 사

용된 날짜나 시간 상수 및 날짜 시간의 스트링 표시는 ISO, EUR, JIS, USA의 형식을 갖거나 DDS와 CRTPF CL 명령을 사용하여 작성되었다면 이 표가 작성될 때 지정된 날짜 및 시간 형식과 일치해야 합니다. DDS 스펙에 서로 다른 여러 날짜 또는 시간 형식이 있는 경우 트리거가 작성되지 못합니다.

트리거된 조치에서 참조된 표를 드롭핑 또는 취소

*triggered-action*에 참조된 표, 보기 또는 별명과 같은 오브젝트가 삭제될 경우, 트리거가 실행될 때 오브젝트를 참조하는 명령문의 액세스 계획이 리빌드됩니다. 오브젝트가 존재하지 않으면 주제표에 대한 INSERT, UPDATE 또는 DELETE 연산이 실패할 것입니다.

트리거를 실행하기 위해 트리거 작성자에게 필요한 권한이 취소된 경우, 트리거가 실행될 때 오브젝트를 참조하는 명령문의 액세스 계획이 리빌드됩니다. 적절한 권한이 존재하지 않으면 주제표에 대한 INSERT, UPDATE 또는 DELETE 연산이 실패할 것입니다.

트리거를 무효화하는 조작

*유효하지 않은 트리거*는 더 이상 활성화시켜 사용할 수 없는 트리거입니다. 트리거가 유효하지 않으면 주제 표에 대하여 INSERT, UPDATE 또는 DELETE 조작을 수행할 수 없습니다. 다음과 같은 경우에 트리거가 유효하지 않습니다.

- *triggered-action*의 SQL문이 주제 표를 참조하고, 트리거가 자신을 참조하는 트리거이며, 그 표가 CRTDUPOBJ CL 명령을 사용하여 복제된 경우
- *triggered-action*의 SQL문이 발신 라이브러리의 표나 뷰를 참조하고, 표가 CRTDUPOBJ CL 명령을 사용하여 복제될 때 새로운 라이브러리에서 오브젝트를 찾을 수 없는 경우
- 표가 RSTOBJ 또는 RSTLIB CL 명령을 사용하여 새로운 라이브러리에 복원되고, *triggered-action*이 주제 표를 참조하며, 트리거가 자신을 참조하는 트리거인 경우

유효하지 않은 트리거는 CREATE TRIGGER문을 사용하여 재작성되기 전에 먼저 드롭(drop)되어야 합니다. 트리거를 드롭하고 재작성하는 것은 동일한 트리거 조작 및 활성화 시간에 대하여 여러 트리거가 주제 표에 정의되어 있을 때 트리거의 활성화 순서에 영향을 미칩니다.

트리거를 실행하는 오류

*SQL-trigger-body*문의 실행시 발생하는 오류는 SQLCODE -723과 SQLSTATE 09000을 사용하여 리턴됩니다.

*SQL-trigger-body*문에는 SIGNAL문이 포함될 수 있습니다. SIGNAL문에 지정된 SQLCODE -438 및 SQLSTATE는 리턴됩니다.

트리거 프로그램 오브젝트

트리거가 작성될 때 SQL은 삽입 SQL문과 함께 C 소스 코드가 들어 있는 임시 소스 파일을 작성합니다. 프로그램 오브젝트는 GRTPGM 명령으로 작성됩니다. 프

로그램을 작성하기 위해 사용된 SQL 옵션은 CREATE TRIGGER문이 실행될 때 유효한 옵션입니다. 프로그램은 ACTGRP(*CALLER)를 사용하여 작성됩니다.

트리거는 트리거 *소유자*의 허용된 권한으로 실행합니다.

트리거 소유권

SQL명이 지정된 경우, 트리거의 *소유자*는 트리거가 작성되는 *스키마*와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 트리거의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 트리거의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

트리거 권한

트리거 프로그램 오브젝트 권한은 다음과 같습니다.

- SQL 명명이 유효할 때 트리거 프로그램은 공용 권한 *EXCLUDE를 가지고 작성되며, 그 이름의 사용자 프로파일이 존재하는 경우 트리거명의 *스키마* 규정자로부터 권한을 허용합니다. *스키마* 규정자의 사용자 프로파일이 있는 경우 트리거 프로그램의 소유자는 이 *스키마* 규정자의 사용자 프로파일이 됩니다. *스키마* 규정자와 같은 이름의 사용자 프로파일이 있고 그 이름이 명령문의 권한 ID와 다른 경우 *스키마* 규정자 라이브러리에 트리거 프로그램 오브젝트를 작성하기 위해 특수 권한인 *ALLOBJ와 *SECADM 이 필요하다는 점에 주의하십시오. 스키마 규정자에 대한 사용자 프로파일이 없는 경우 트리거 프로그램의 소유자는 SQL CREATE TRIGGER문을 실행하는 작업의 사용자 프로파일이거나 그룹 사용자 프로파일입니다. 그룹 사용자 프로파일은 명령문을 실행하는 사용자의 프로파일에 OWNER(*GRPPRF)가 지정된 경우에만 트리거 프로그램 오브젝트의 소유자가 됩니다.트리거 프로그램의 소유자가 그룹 프로파일의 멤버이고 OWNER(*GRPPRF)가 사용자 프로파일에 지정된 경우 프로그램은 그룹 프로파일의 허용된 권한을 사용하여 실행됩니다.
- System 명명이 유효할 때 트리거 프로그램은 공용 권한 *EXCLUDE를 가지고 작성되며, SQL CREATE TRIGGER문을 실행하는 작업의 사용자 또는 그룹 사용자 프로파일로부터 권한을 허용합니다.

트랜잭션 분리

모든 트리거는 활성화될 때 SET TRANSACTION문을 수행하여 트리거에 의한 모든 조작이 그 트리거가 실행되도록 하는 어플리케이션 프로그램과 같은 분리 레벨에서 수행되도록 합니다. 사용자는 자신의 SET TRANSACTION문을 트리거의 *SQL-trigger-body*의 *SQL-control-statement*에 놓을 수 있습니다. 사용자가 SET TRANSACTION문을 트리거의 *SQL-trigger-body*에 두면 트리거는 트리거가 실행되도록 하는 어플리케이션 프로그램의 분리 레벨 대신 SET TRANSACTION문에 지정된 분리 레벨로 실행됩니다.

트리거가 활성화되도록 하는 어플리케이션 프로그램이 No Commit(COMMIT (*NONE) 또는 COMMIT(*NC)) 이외의 분리 레벨로 실행되는 경우 트리거 내에서의 조작은 확약 제어 하에서 실행되며 어플리케이션이 현재의 작업 단위를 확약할 때까지 확약이나 구간복원은 수행하지 않습니다. 트리거의 *SQL-trigger-body*에 ATOMIC이 지정되고 ATOMIC 트리거가 활성화되도록 한 어플리케이션 프로그램이 No Commit(COMMIT(*NONE)이나 COMMIT(*NC))의 분리 레벨로 실행되는 경우 트리거 내의 조작은 확약 제어 하에서 실행되지 않습니다. 트리거를 활성화되도록 한 어플리케이션이 No Commit (COMMIT(*NONE) 또는 COMMIT(*NC))의 분리 레벨로 실행되는 경우 트리거 조작은 즉시 데이터베이스에 작성되고 구간복원될 수 없습니다.

실제 파일 트리거 추가(ADDPFTRG) CL 명령에 의해 정의된 시스템 트리거와 CREATE TRIGGER문에 의해 정의된 SQL 트리거가 모두 하나의 표에 정의된 경우 시스템 트리거가 SET TRANSACTION문을 수행하여 이 트리거들이 활성화되도록 한 원래의 어플리케이션과 동일한 분리레벨에서 실행되도록 하는 것이 좋습니다. 또한 시스템 트리거가 호출한 어플리케이션의 활성 그룹에서 실행되는 것이 좋습니다. 시스템 트리거가 별도의 활성 그룹(ACTGRP(*NEW))에서 실행되는 경우 이 시스템 트리거는 호출한 어플리케이션의 작업 단위에도 참여하지 않고 어떤 SQL 트리거의 작업단위에도 참여하지 않습니다. 별도의 활성 그룹에서 실행되는 시스템 트리거는 확약 제어 하에서 수행한 모든 데이터베이스 조작을 확약하거나 구간복원할 책임이 있습니다. CREATE TRIGGER문에 의해 정의된 SQL 트리거는 항상 호출자의 활성 그룹에서 실행됨을 주의하십시오.

트리거하는 어플리케이션이 확약 제어 하에서 실행되는 경우 SQL 트리거 조작 및 모든 직렬 SQL 트리거는 하위 작업 단위로 캡처됩니다. 트리거 및 모든 직렬 트리거가 성공적인 경우 하위 작업 단위에 캡처된 조작은 트리거하는 어플리케이션이 현재의 작업 단위를 확약하거나 구간복원할 때 확약 또는 구간 복원합니다. 호출 프로그램과 같은 활성 그룹에서 실행되고 호출 프로그램의 분리 레벨에 SET TRANSACTION을 수행하는 시스템 트리거 역시 하위 작업 단위에 참여합니다. 트리거하는 어플리케이션이 확약 제어 없이 실행되는 경우 SQL 트리거 조작 역시 확약 제어 없이 실행됩니다.

트리거가 활성화되도록 하는 어플리케이션이 No Commit(COMMIT(*NONE) 또는 COMMIT(*NC)) 분리 레벨로 실행되고 INSERT, UPDATE 또는 DELETE 문을 발행하여 실행되는 동안 오류가 발생한 경우 해당 조작에 대한 오류 발생 이후의 어떠한 시스템 및 SQL 트리거도 활성화되지 않습니다. 그러나 일부 변경사항은 이미 수행되었습니다. 트리거하는 어플리케이션이 확약 제어 하에서 실행되는 경우 하위 작업 단위에 캡처된 트리거 조작은 첫 번째 오류가 발생했을 때 구간복원되며, 현재의 INSERT, UPDATE 또는 DELETE문에 대하여 추가적인 트리거가 활성화되지 않습니다.

예

예 1

회사가 관리하는 사원 번호를 추적하는 두 개의 트리거를 작성합니다. 트리거하는 표는 EMPLOYEE 표이거 트리거는 COMPANY_STATS 표의 사원 전체 수가 들어 있는 열을 증가 또는 감소시킵니다. COMPANY_STATS 표에는 다음과 같은 등록정보가 있습니다.

```
CREATE TABLE COMPANY_STATS
  (NBEMP INTEGER,
   NBPRODUCT INTEGER,
   REVENUE DECIMAL(15,0))
```

이 예에서는 다른 표의 요약 자료를 유지보수하기 위해 행 트리거를 사용합니다.

첫 번째 트리거 NEW_HIRE를 작성하여 새로운 사람이 고용될 때마다 사원 수를 증가시킵니다. 즉 새로운 행이 EMPLOYEE 표에 삽입될 때마다 COMPANY_STATS 표의 NBEMP 열 값을 1씩 증가시킵니다.

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1;
    END
```

두 번째 트리거 FORM_EMP를 작성하여 사원이 퇴사할 때마다 사원 수를 감소시킵니다. 즉 EMPLOYEE 표에서 행이 삭제될 때마다 COMPANY_STATS 표의 NBEMP 열 값을 1씩 감소시킵니다.

```
CREATE TRIGGER FORM_EMP
  AFTER DELETE ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1;
    END
```

예 2

부품 행이 갱신되고 영향을 받는 부품에 대한 보유 수량이 최대 재고 수량의 10% 미만인 경우에만 출하 요청을 발행하는 사용자 정의 함수 ISSUE_SHIP_REQUEST를 호출하는 트리거인 REORDER를 작성합니다. 사용자 정의 함수 ISSUE_SHIP_REQUEST는 부품의 최대 재고 수량에서 보유 수량을 뺀 나머지 수량을 주문합니다. 이 함수는 또한 요청이 적절한 공급업체로 송신되었는지를 확인합니다.

부품 행은 PARTS 표에 있습니다. 표에 열이 더 있어도 트리거는 ON_HAND나 MAX_STOCKED 열이 갱신될 경우에만 활성화됩니다.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS NROW
  FOR EACH ROW MODE DB2SQL
```

```
  WHEN (NROW.ON_HAND < 0.10 * NROW.MAX_STOCKED)
  BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(NROW.MAX_STOCKED - NROW.ON_HAND, NROW.PARTNO));
    END
```

예 3

사용자 정의 함수를 호출하기 위해 VALUES문 대신 전체 선택을 사용하는 것을 제외한 예2의 시나리오를 반복합니다. 이 예에서는 또는 행 트리거 대신 명령문 트리거로서 트리거를 정의하는 방법을 보여줍니다. WHERE절에 대하여 참인 전이 표의 각 행에 대하여 출하 요구가 해당 부품에 대해 발행됩니다.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW TABLE AS NTABLE
  FOR EACH STATEMENT MODE DB2SQL
  BEGIN ATOMIC
    SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
    FROM NTABLE
    WHERE ON_HAND < 0.10 * MAX_STOCKED;
    END
```

CREATE VIEW

CREATE VIEW문은 현재 서버에서 하나 이상의 표나 뷰에 뷰를 작성합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 논리 파일 작성(CRTLF) CL 명령에 대해 *USE
  - 뷰가 작성되는 라이브러리에 대해 *EXECUTE 및 *ADD
  - 뷰가 작성되는 라이브러리가 자료 사전을 가지고 있는 SQL 스키마인 경우자료 사전에 대한 *CHANGE
- 관리 권한

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 전체 선택을 통해 직접적으로 참조되거나 전체 선택에서 참조된 뷰를 통해 간접적으로 참조된 표나 뷰의 경우
  - 표 또는 뷰에서의 SELECT 권한 및
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

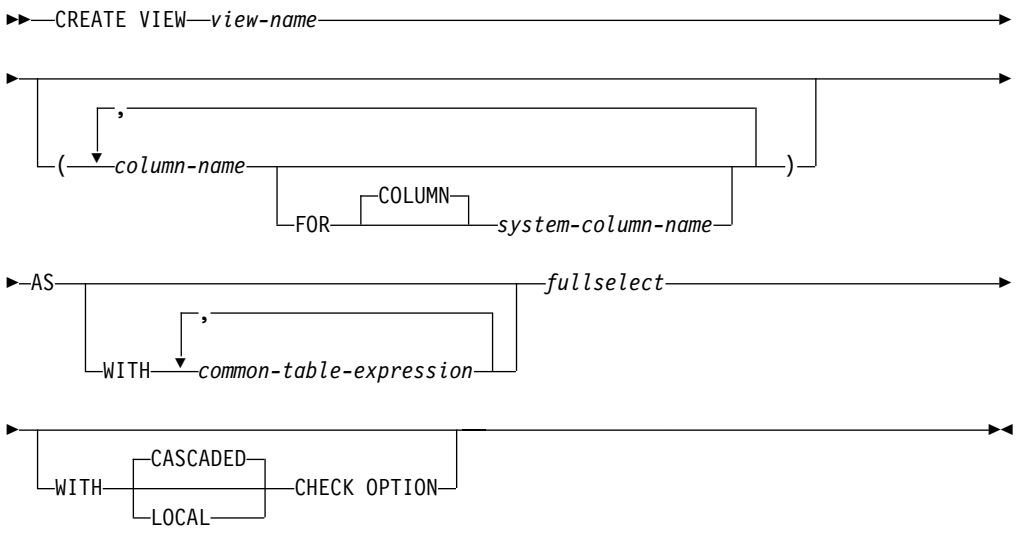
명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을, 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

구문



설명

view-name

뷰를 명명합니다. 내재적 또는 명시적 규정자를 포함하여 이름은 이미 서버에 있는 표, 뷰, 색인, 별명 또는 파일과 같을 수 없습니다.

SQL명이 지정되면 뷰는 내재적 또는 명시적 규정자에 의해 지정된 스키마에 작성됩니다.

시스템명이 지정되면 뷰는 규정자에 의해 지정된 스키마에 작성됩니다. 규정되지 않았으면 뷰 이름은 첫 번째 FROM 절(모든 공통 표 표현식이나 내포된 표 표현식의 FROM 절 포함)에 지정된 첫 번째 표와 같은 스키마에 작성됩니다.

뷰 이름이 유효한 시스템명이 아니면 iSeries용 DB2 UDB SQL은 시스템명을 생성합니다. 이름 생성 규칙에 대한 내용은 572 페이지의 『표 이름 생성 규칙』을 참조하십시오.

(column-name, ...)

뷰의 열을 명명합니다. 열 이름 리스트가 지정된 경우 리스트는 전체 선택 결과 표의 열 수와 같은 수의 이름으로 구성되어야 합니다. 각 *column-name* 및 *system-column-name*은 고유해야 하며 규정되지 않아야 합니다. 열 이름 리스트를 지정하지 않으면 뷰의 열은 전체 선택의 결과표의 열의 시스템명과 열 이름을 계승합니다.

부속 선택의 결과표에 중복된 열 이름, 중복된 시스템 열 이름 또는 명명되지 않은 열이 있으면, 열 이름(및 시스템 열 이름) 리스트를 지정해야 합니다. 명명되지 않은 열에 대한 내용은 339 페이지의 『결과 열 이름』을 참조하십시오.

FOR COLUMN *system-column-name*

열에 대해 OS/400 이름을 제공합니다. 뷰의 하나 이상의 열이나 뷰의 column-name에 대해 같은 이름을 사용할 수 없습니다.

system-column-name이 지정되지 않고 column-name이 유효한 system-column-name이 아니면, 시스템 열 이름이 생성됩니다. 시스템 열 이름이 생성되는 방법에 대한 자세한 내용은 572 페이지의 『열 이름 생성 규칙』을 참조하십시오.

AS *fullselect*

뷰를 정의합니다. 전체 선택이 실행되었으면 뷰는 항상 결과가 되는 행으로 구성됩니다.

*common-table-expression*은 뒤 따르는 전체 선택과 함께 사용하기 위해 공통 표 표현식을 정의합니다. 자세한 내용은 353 페이지의 『common-table 표현식』을 참조하십시오.

*전체선택*은 호스트 변수를 참조하면 안됩니다. *전체선택*에 대한 설명은 350 페이지의 『fullselect』를 참조하십시오.

WITH CASCADED CHECK OPTION

뷰를 통해 삽입되거나 갱신된 모든 행은 뷰의 정의를 따라야 한다것을 지정합니다. 뷰의 정의를 따르지 않는 행은 그 뷰를 사용하여 검색될 수 없는 행입니다.

WITH CHECK OPTION 지정되지 않으면:

- 뷰의 읽기 만이 가능합니다.
- 뷰의 정의는 부속 조회를 포함합니다.
- 뷰 정의의 WHERE절에 scalar-subselect가 포함됩니다.
- 뷰의 정의는 비 판별적 함수를 포함합니다.

WITH CHECK OPTION이 삽입을 허용하지 않는 갱신가능한 뷰에 대해 지정되면 검사 옵션은 갱신에만 적용됩니다.

WITH CHECK OPTION이 생략되면 뷰의 정의는 뷰를 사용하는 삽입 또는 갱신 조작의 검사에 사용되지 않습니다. 뷰가 WITH CHECK OPTION을 포함하는 다른 뷰에 직접적 또는 간접적으로 종속되면 일부 검사는 삽입이나 갱신 조작 중에 발생합니다. 뷰 정의가 사용되지 않기 때문에 뷰의 정의를 따르지 않는 행은 뷰를 통해 삽입되거나 갱신됩니다.

뷰 V에서의 WITH CHECK OPTION은 V에 직접적 또는 간접적으로 종속되는 갱신 가능한 뷰에 의해 계승됩니다. 따라서, 갱신가능한 뷰가 V에서 정의되면 WITH CHECK OPTION이 그 뷰에 지정되지 않더라도 V에 대한 검사 옵션은 그 뷰에 적용됩니다. 예를 들어, 다음의 갱신가능한 뷰를 고려합니다.

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

V1에 WITH CHECK OPTION이 없고 V1이 WITH CHECK OPTION이 있는 다른 뷰에 종속되지 않기 때문에 V1을 사용하는 다음 INSERT문은 성공합니다.

```
INSERT INTO V1 VALUES(5)
```

V2에 WITH CHECK OPTION이 있고 삽입으로 V2 정의에 따르지 않는 행이 작성되므로 V2를 사용하는 다음 INSERT문은 결과적으로 오류를 생성합니다.

```
INSERT INTO V2 VALUES(5)
```

V3가 WITH CHECK OPTION이 없는 V2에 종속되므로 WITH CHECK OPTION이 없더라도 V3를 사용하는 다음 INSERT문은 결과적으로 오류를 생성합니다.

```
INSERT INTO V3 VALUES(5)
```

V3의 정의를 따르지 않더라도(V3에는 WITH CHECK OPTION이 없습니다) V2의 정의를 따르기 때문에(V2에는 WITH CHECK OPTION이 있습니다), V3를 사용하는 다음 INSERT문은 성공합니다.

```
INSERT INTO V3 VALUES(200)
```

WITH LOCAL CHECK OPTION

뷰가 WITH LOCAL CHECK OPTION으로 정의될 때 행이 더 이상 뷰 정의를 따르지 않도록 행을 갱신하는 것이 가능하다는 것을 제외하고는 WITH LOCAL CHECK OPTION은 WITH CASCADED CHECK OPTION과 동일합니다. 이는 뷰가 WITH CASCADED CHECK OPTION이나 WITH LOCAL CHECK OPTION절 없이 정의된 뷰에 직접적 또는 간접적으로 종속되어 있을 때만 발생합니다.

WITH LOCAL CHECK OPTION은 행이 삽입되거나 갱신될 때 WITH LOCAL CHECK OPTION 또는 WITH CASCADED CHECK OPTION이 있는 종속 뷰의 탐색 조건만 검사됨을 지정합니다. 반대로, WITH CASCADED CHECK OPTION은 행이 삽입되거나 갱신될 때 모든 종속 뷰의 탐색 조건이 검사됨을 지정합니다.

CASCADED와 LOCAL의 차이점은 예를 통해 쉽게 이해할 수 있습니다. x와 y가 LOCAL이나 CASCADED를 나타내는 다음의 갱신가능한 뷰를 고려합니다.

- T0에 정의된 V1
- V1 WITH x CHECK OPTION에 정의된 V2
- V2에 정의된 V3

- V3 WITH y CHECK OPTION에 정의된 V4
- V4에 정의된 V5

다음 표는 INSERT나 UPDATE 조작 중에 검사되는 뷰 탐색 조건을 설명합니다.

표 48. INSERT 및 UPDATE 중에 탐색 조건이 검사되는 뷰

INSERT 또는 UPDATE에 사용되는 뷰	**x = LOCAL** **y = LOCAL**	**x = CASCADED** **y = CASCADED**	**x = LOCAL** **y = CASCADED**	**x = CASCADED** **y = LOCAL**
V1	없음	없음	없음	없음
V2	V2	V2 V1	V2	V2 V1
V3	V2	V2 V1	V2	V2 V1
V4	V4 V2	V4 V3 V2 V1	V4 V3 V2 V1	V4 V2 V1
V5	V4 V2	V4 V3 V2 V1	V4 V3 V2 V1	V4 V2 V1

주

제거가능 뷰: 커서는 아래 사항이 참일 경우 *제거가능* 합니다.

- 외부 전체선택은 단 하나의 기본 표 또는 삭제 가능 뷰를 나타냅니다.
- 외부 전체선택은 GROUP BY 절 또는 HAVING 절을 포함하지 않습니다.
- 외부 전체선택은 선택 리스트에서 열 함수를 포함하지 않습니다.
- 외부 전체선택은 UNION 또는 UNION ALL 연산자를 포함하지 않습니다.
- 외부 전체선택은 DISTINCT 절을 포함하지 않습니다.

갱신가능 뷰: 뷰의 열은 다음이 참일 경우 *갱신가능* 합니다.

- 뷰가 제거 가능합니다.
- 열은 표의 열이나 다른 뷰의 갱신가능 열에서만 도출됩니다. 즉, 적어도 하나의 결과 열은 표현식에서 파생된 연산자, 스칼라 함수, 상수 또는 열을 포함하는 표현식에서 파생되어서는 안됩니다.

첫 번째 SELECT절에 열에서 단독으로 파생된 적어도 하나의 결과 열이 있지 않으면 뷰는 UPDATE문의 오브젝트 표가 될 수 없습니다. 즉, 적어도 하나의 결과 열은 표현식에서 파생된 연산자, 스칼라 함수, 상수 또는 열을 포함하는 표현식에서 파생되어서는 안됩니다.

뷰의 열이 *갱신 가능*한 경우 뷰는 *갱신 가능*합니다.

삽입 가능 뷰: 뷰의 적어도 한 열이 갱신 가능한 경우 해당 뷰는 *갱신 가능*합니다.

읽기 전용 뷰 뷰가 삭제 가능하지 않은 경우 *읽기 전용*입니다. 읽기 전용 뷰는 INSERT, UPDATE 또는 DELETE문의 오브젝트가 될 수 없습니다.

커서는 삭제 가능하지 않은 경우 *읽기 전용*입니다.

정렬 순서: 뷰는 CREATE VIEW문이 실행될 때 유효한 정렬 순서로 작성됩니다. 뷰의 정렬 순서는 뷰 전체선택의 SBCS 자료와 혼합된 자료와 관련된 모든 비교에 적용됩니다. 뷰가 조회에 포함될 때 중간 결과표는 뷰 전체선택에서 생성됩니다. 조회가 실행될 때의 정렬 순서는 조회에 지정된 모든 선택에 적용됩니다.

뷰 속성: 뷰는 키순이 아닌 논리 파일로 작성됩니다. 뷰가 작성되면 파일 대기 시간과 레코드 대기 시간 속성은 논리 파일 작성(CRTLF) 명령의 WAITFILE 및 WAITRCD 키워드에 지정된 디폴트 값으로 설정됩니다.

분배된 표에 대해 작성된 뷰는 표가 분배되는 모든 시스템에서 작성됩니다. 뷰가 하나 이상의 분산 표에 대해 작성되고 그 표들이 같은 노드 그룹을 사용하여 분배되지 않으면 뷰는 CREATE VIEW문을 수행하는 시스템에서만 작성됩니다. 분배된 표에 대한 자세한 내용은 DB2 Multisystem 책을 참조하십시오.

뷰 소유권: SQL명이 지정된 경우, 뷰의 *소유자*는 뷰가 작성되는 스키마와 동일한 이름을 가진 사용자 프로파일입니다. 그렇지 않으면 뷰의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

시스템명이 지정되면, 뷰의 *소유자*는 명령문을 실행하는 작업의 그룹 사용자 프로파일이나 사용자 프로파일입니다.

뷰 권한: SQL 이름이 사용되면 뷰는 *PUBLIC에 대한 *EXCLUDE 시스템 권한으로 작성됩니다. 시스템명이 사용되면 스키마의 권한 작성(CRTAUT) 매개변수에 의해 판별되듯이 뷰는 *PUBLIC에 대한 권한으로 작성됩니다.

뷰의 소유자가 그룹 프로파일의 멤버이고(GRPPRF 키워드) 그룹 권한이 지정되면(GRPAUT 키워드), 그 그룹 프로파일은 뷰에 대한 권한도 갖습니다.

소유자는 항상 뷰를 제거하는 권한과 뷰에서 SELECT 권한을 예약합니다. 소유자가 전체선택에 식별된 모든 표나 뷰에서 SELECT 권한을 부여하는 권한도 가지고 있으면 소유자는 SELECT 권한을 다른 사람에게 부여할 수 있습니다.

소유자는 뷰에서 INSERT, UPDATE 및 DELETE 권한을 예약할 수도 있습니다. 뷰가 읽기 전용이 아니면, 소유자가 전체선택의 첫 번째 FROM절에서 식별된 표나 뷰에 대해 가지고 있는 것과 같은 권한을 새로운 뷰에서 예약합니다. 파생된 권한도 부여될 수 있을 경우에만 이러한 권한이 부여될 수 있습니다.

ID 열: 보기 정의의 fullselect의 해당 열 요소가 표의 ID 열 이름이거나 기본 표의 ID 열 이름에 직접/간접적으로 맵핑되는 보기의 열 이름인 경우, 보기 열은 ID 열로 간주됩니다. 다른 모든 경우, 뷰의 열은 ID 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.

- 뷰 정의의 선택 리스트에는 ID 열 이름의 복수 인스턴스(즉, 동일한 열을 두 번 이상 선택)가 포함됩니다.

- 뷰 정의에는 결합이 관련됩니다.
- 뷰 정의의 열에는 ID 열을 나타내는 표현식이 포함됩니다.
- 뷰 정의에는 UNION이 포함됩니다.

뷰 제한사항: 첫 번째 SELECT절에 열에서 단독으로 파생된 적어도 하나의 결과 열이 있지 않으면 뷰는 UPDATE문의 오브젝트 표가 될 수 없습니다. 즉, 적어도 하나의 결과 열은 표현식에서 파생된 연산자, 스칼라 함수, 상수 또는 열을 포함하는 표현식에서 파생되어서는 안됩니다.

하위 뷰에 의해 참조되는 실제 표를 포함하여 뷰는 32개 이상의 실제 표를 참조할 수 없습니다.

뷰는 8000개 이상의 열을 주소지정할 수 없습니다. 뷰에서 참조된 표 수, 열 이름 길이, WHERE절의 길이도 이 수를 줄입니다.

뷰 정의 테스트: SELECT * FROM *view-name*을 실행하여 뷰 정의의 의미를 테스트할 수있습니다.

예

예 1

'MA' 문자로 시작하는 프로젝트 번호(PROJNO)가 있는 행만 들어 있는 PROJECT 표에 대해 MA_PROJ라는 뷰를 작성합니다.

```
CREATE VIEW MA_PROJ
  AS SELECT * FROM PROJECT
        WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 2

예 1과 같이 뷰를 작성합니다. 그러나 프로젝트 번호(PROJNO), 프로젝트 이름(PROJNAME) 및 프로젝트를 책임지는 사원(RESPEMP) 열만 선택합니다.

```
CREATE VIEW MA_PROJ2
  AS SELECT PROJNO, PROJNAME, RESPEMP FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 3

예 2와 같이 뷰를 작성하며 뷰에서 프로젝트 IN_CHARGE를 담당하는 사원 열을 호출합니다.

```
CREATE VIEW MA_PROJ (PROJNO, PROJNAME, IN_CHARGE)
  AS SELECT PROJNO, PROJNAME, RESPEMP FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

주: 열 이름 중 하나만을 변경해도 뷰에 있는 세 열의 이름이 모두 MA_PROJ 뒤에 오는 괄호 안에 나열되어야 합니다.

예 4

PROJECT 표의 처음 네 열(PROJNO, PROJNAME, DEPTNO, RESPEMP)과 프로젝트(RESPEMP)를 담당하는 사원의 성(LASTNAME)이 들어 있는 PRJ_LEADER이라는 뷰를 작성합니다. EMPLOYEE의 EMPNO와 PROJECT의 RESEMP를 일치시켜 EMPLOYEE 표에서 이름을 구합니다.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO
```

예 5

예 4에서처럼 뷰를 작성합니다. 그러나 PROJNO, PROJNAME, DEPTNO, RESEMP 및 LASTNAME 열외에도 책임자의 총 급여(SALARY + BONUS +COMM)를 표시합니다. 또한 1 보다 큰 평균 사원(PRSTAFF)이 있는 프로젝트만 선택합니다.

```
CREATE VIEW PRJ_LEADER (PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY)
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

DECLARE CURSOR

DECLARE CURSOR문은 커서를 정의합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. Java에 지정되어서는 안됩니다.

권한부여

이 명령문을 사용하기 위해서는 권한부여가 필요하지 않습니다. 그러나 커서에 대해 OPEN 이나 FETCH를 사용하려면 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 커서의 SELECT문에 식별된 각 표나 뷰의 경우
  - 표 또는 뷰에서의 SELECT 권한 및
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을, 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

커서의 SELECT문은 다음과 같습니다.

- *statement-name*으로 식별된 준비된 select문
- 지정된 *select-statement*

statement-name이 지정된 경우

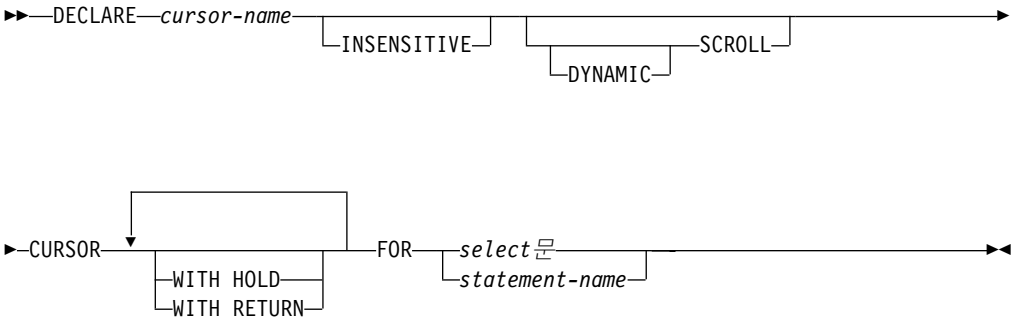
- 프로그램이 작성되었을 때 DYNUSRPRF(*OWNER)가 CRTSQLxxx 명령에 지정되지 않았으면, 명령문의 권한부여 ID는 실행 시간 권한부여 ID입니다. 자세한 내용은 58 페이지의 『권한부여 ID 및 권한부여명』을 참조하십시오.

- DLYPRP(*YES)가 CRTSQLxxx 명령에 지정되지 않은 경우 select문이 준비될 때 권한부여 검사가 수행됩니다.
- DLYPRP(*YES) 매개변수로 컴파일된 프로그램에 대해 커서가 열릴 때 권한부여 검사가 수행됩니다.

***select*문**이 지정된 경우

- SQL 이름을 갖는 USRPRF(*OWNER) 또는 USRPRF(*NAMING)이 CRTSQLxxx 명령에 지정되었으면, 명령문의 권한부여 ID는 SQL 프로그램이나 패키지의 소유자입니다.
- 시스템 이름을 갖는 USRPRF(*OWNER) 또는 USRPRF(*NAMING)이 CRTSQLxxx 명령에 지정되었으면 명령문의 권한부여 ID는 실행 시간 권한부여 ID입니다.
- REXX에서 명령문의 권한부여 ID는 실행 시간 권한부여 ID입니다.
- 커서가 열릴 때 권한부여 검사가 수행됩니다.

구문



설명

cursor-name

커서를 명명합니다. 이름은 소스 프로그램에 선언된 다른 커서의 이름과 동일하면 안됩니다.

INSENSITIVE

일단 커서가 열리면 이 활성 그룹 또는 다른 활성 그룹에 의해 수행되는 삽입, 갱신 또는 삭제에 대한 감도를 가지지 않음을 지정합니다. INSENSITIVE가 지정되면 커서가 열릴 때 커서는 읽기 전용이고 임시 결과가 작성됩니다. 또한 SELECT 문에는 FOR UPDATE절이 들어갈 수 없고 어플리케이션은 자료(ALWCPYDTA (*OPTIMIZE) 또는 ALWCPYDTA(*YES))의 복사를 허용하지 않습니다.

SCROLL

커서를 화면이동할 수 있음을 지정합니다. 커서는 다른 활성 그룹에 의해 수행되는

삽입, 갱신 및 삭제에 대한 즉각적인 감도를 가지거나 가지지 않습니다. DYNAMIC이 지정되지 않으면 커서는 읽기 전용입니다. 또한 SELECT문에는 FOR UPDATE 절이 들어갈 수 없습니다.

DYNAMIC SCROLL

결과표가 갱신가능하고 그 커서가 다른 어플리케이션 프로세스에 의해 수행되는 삽입, 갱신 및 삭제에 대한 즉각적인 감도를 갖는 경우 커서가 갱신가능함을 지정합니다. 그러나 다음 경우에는 키워드 DYNAMIC이 무시되고 커서는 삽입, 갱신 및 삭제에 대한 즉각적인 감도를 가지지 않습니다.

- 임시 결과표로 구현된 조회. 임시 결과표는 다음과 같은 경우에 작성됩니다.
  - INSENSITIVE가 지정되었습니다.
  - ORDER BY절에 지정된 열의 기억장치 총 바이트 수가 2000바이트를 초과할 때
  - ORDER BY 및 GROUP BY절이 다른 열이나 다른 순서로 된 열을 지정할 때
  - ORDER BY 및 GROUP BY절이 사용자 정의 함수나, FILE LINK CONTROL 및 READ PERMISSION DB의 속성을 갖는 자료 링크에 대해 DLVALUE, DLURLPATH, DLURLPATHONLY, DLURLSERVER, DLURLSCHEME 또는 DLURLCOMPLETE 스칼라 함수 중 하나를 포함할 때
  - UNION 또는 DISTINCT절이 지정될 때
  - ORDER BY 또는 GROUP BY절이 모두 같은 표에 있지 않은 열을 지정할 때
  - JOINDFT 자료 정의 스펙(DDS) 키워드에 의해 정의된 논리 파일이 다른 파일과 결합될 때
  - 여러 개의 데이터베이스 파일 멤버를 기초로 한 논리 파일이 지정될 때
  - DECLARE CURSOR의 select문에 GROUP BY절이 있는 경우 CURRENT 또는 RELATIVE 화면이동 옵션이 FETCH문에 지정될 때
- 부속 조회를 포함하는 조회
  - 가장 외부의 조회는 내부 부속 조회에 상관된 값을 제공하지 않습니다.
  - No IN, = ANY, = SOME 또는 <> ALL 부속 조회는 가장 외부의 조회에 의해 참조됩니다.

WITH HOLD

커서가 확약 조작의 결과로 닫히지 않도록 합니다. 커서와 관련된 연결이 확약 조작을 수행하는 동안 종료되면 WITH HOLD 구를 사용하여 선언된 커서는 확약 시간에 내재적으로 닫힙니다.

WITH HOLD가 지정될 때 확약 조작은 현재 작업 단위의 변경을 모두 확약하지만, 커서를 유지관리하는 데 필요하지 않은 잠금만 해제합니다. 나중에는 위치지정된 UPDATE나 DELETE문이 실행될 수 있기 전에 FETCH문이 필요합니다.

모든 커서는 내재적으로 CONNECT(유형 1)나 롤백 조작에 의해 닫힙니다. 연결과 연관된 모든 커서는 연결의 단절에 의해 내재적으로 닫힙니다. WITH HOLD

가 지정되지 않았거나 커서와 연관된 연결이 릴리스 지연 중 상태에 있으면, 커서는 확약 조작에 의해서 내재적으로 닫힙니다.

확약 조작 전에 커서가 닫히면 결과는 커서가 WITH HOLD 옵션 없이 선언된 경우와 동일합니다.

WITH RETURN

이 절은 커서를 프로시듀어로부터의 결과 세트로 사용할 것임을 나타냅니다. WITH RETURN은 DECLARE CURSOR문이 프로시듀어에 대한 소스 코드와 함께 들어 있을 때 적절합니다. 다른 경우에는 사전컴파일러가 절을 받아들이지만 유효하지 않습니다.

SQL 프로시듀어 내에서 SQL 프로시듀어가 종료될 때 여전히 열려 있는 WITH RETURN절을 사용하여 선언된 커서가 SQL 프로시듀어로부터의 결과 세트를 정의합니다. SQL 프로시듀어의 다른 모든 열린 커서들은 SQL 프로시듀어가 종료할 때 닫힙니다. 외부 프로시듀어(LANGUAGE SQL을 사용하여 정의되지 않은 것)에서 WITH RETURN절은 유효하지 않으며 외부 프로시듀어 종료시 열려 있는 커서는 모두 결과 세트로 간주됩니다.

결과 세트는 프로시듀어가 호출 프로그램으로 리턴할 때 현재 커서 위치에서 결과 세트 끝까지의 모든 행으로 구성됩니다.

select-statement

커서의 SELECT문을 지정합니다. 자세한 내용은 352 페이지의 『select문』을 참조하십시오.

*select-statement*은 매개변수 마커(REXX 제외)를 포함해서는 안되지만 호스트 변수에 대한 참조는 포함할 수 있습니다. RPG, PL/I 및 REXX 이외의 호스트 언어에서 호스트 변수의 선언은 소스 프로그램의 DECLARE CURSOR문에 선행되어야 합니다. 호스트 변수 선언은 RPG 및 PL/I의 DECLARE CURSOR문을 따를 수 있습니다. REXX에서 매개변수 마커가 호스트 변수 대신 사용되어야 하며 명령문이 준비되어야 합니다.

statement-name

커서의 SELECT문은 커서가 열릴 때 *statement-name*에 의해 식별된 준비된 select 문입니다. *statement-name*은 소스 프로그램의 다른 DECLARE CURSOR문에 지정된 *statement-name*과 반드시 다른 것이어야 합니다. 준비된 명령문에 대한 설명은 728 페이지의 『PREPARE』를 참조하십시오.

주

DECLARE CURSOR문은 이름으로 커서를 참조하는 모든 명령문에 선행해야 합니다.

커서의 결과 표: 열린 상태의 커서는 *결과표*와 그 표의 행에 상대적인 위치를 지정합니다. 표는 커서의 SELECT문에 의해 지정된 결과표입니다.

커서는 다음의 경우가 모두 참일 경우 *제거가능* 합니다.

- 외부 전체선택은 단 하나의 기본 표 또는 삭제 가능 뷰를 나타냅니다.
- 외부 전체선택은 GROUP BY 절 또는 HAVING 절을 포함하지 않습니다.
- 외부 전체선택은 선택 리스트에서 열 함수를 포함하지 않습니다.
- 외부 전체선택은 UNION 또는 UNION ALL 연산자를 포함하지 않습니다.
- 외부 전체선택은 DISTINCT 절을 포함하지 않습니다.
- select문에는 ORDER BY절이 들어 있고 FOR UPDATE OF절 또는 DYNAMIC SCROLL은 지정됩니다.
- select문에는 FOR READ ONLY 절이 포함되지 않습니다.
- select문에는 FETCH FIRST n ROWS ONLY 절이 포함되지 않습니다.
- 외부 fullselect의 결과는 임시 표를 사용하지 않습니다.
- DYNAMIC 키워드 또한 지정되지 은 경우 select문에는 SCROLL 키워드가 포함되지 않습니다.
- 선택 리스트에는 FOR UPDATE OF절은 지정되지 않는다면 DATALINK 열이 포함되지 않습니다.

커서와 연관된 외부 전체 선택의 선택 리스트에 있는 결과 열은 다음 조건이 모두 충족되면 *갱신가능*합니다.

- 커서가 제거 가능합니다.
- 결과 열은 표의 열이나 뷰의 갱신가능 열에서만 도출됩니다. 즉, 적어도 하나의 결과 열은 표현식에서 파생된 연산자, 스칼라 함수, 상수 또는 열을 포함하는 표현식에서 파생되어서는 안됩니다.

커서는 삭제 가능하지 않은 경우 *읽기 전용*입니다.

ORDER BY가 지정되고 FOR UPDATE OF가 지정되면 FOR UPDATE OF절에 있는 열은 ORDER BY절에 지정된 열과 같을 수 없습니다.

FOR UPDATE OF절이 생략되면 갱신될 수 있는 subselect의 SELECT절에 있는 열만 변경될 수 있습니다.

커서의 범위: *cursor-name*의 범위는 정의된 소스 프로그램 즉, 사전 컴파일러로 입력되는 프로그램입니다. 따라서 커서 선언에 의해 사전컴파일된 명령문에 의해서만 커서를 참조할 수 있습니다. 예를 들면, 다른 별도로 컴파일된 프로그램에서 호출된 프로그램은 호출하는 프로그램에 의해 열린 커서를 사용할 수 없습니다.

cursor-name의 범위는 커서가 있는 프로그램이 실행되는 스레드로 제한됩니다. 예를 들어, 같은 프로그램이 같은 작업에서 두 개의 분리된 스레드로 실행되면 두 번째 스레드는 첫 번째 스레드에 의해 열린 커서를 사용할 수 없습니다.

CLOSQLCSR(*ENDJOB), CLOSQLCSR(*ENDSQL) 또는 CLOSQLCSR (*ENDACTGRP)가 CRTSQLxxx 명령에 지정되지 않으면 커서는 프로그램 스택에 있는 프로그램의 동일한 인스턴스에서만 참조될 수 있습니다.

- CLOSQLCSR(*ENDJOB)이 지정되면 커서는 프로그램 스택에 있는 프로그램의 인스턴스에 의해서 참조될 수 있습니다.
- CLOSQLCSR(*ENDSQL)이 지정되면 프로그램 스택에 있는 마지막 SQL 프로그램이 종료될 때까지 커서는 프로그램 스택에 있는 프로그램의 인스턴스에 의해 참조될 수 있습니다.
- CLOSQLCSR(*ENDACTGRP)가 지정되면 커서는 활성 그룹이 종료될 때까지 활성 그룹에 있는 모듈의 모든 인스턴스에 의해 참조될 수 있습니다.

커서의 범위가 그 커서가 선언된 프로그램이라고 하더라도, 프로그램에서 작성된 각 패키지는 커서의 분리된 인스턴스를 포함하며 하나 이상의 커서가 실행 시간에 존재할 수 있습니다. 예를 들면, CONNECT(유형 2)문을 사용하여 프로그램이 위치 X와 위치 Y에 다음 순서로 연결된다고 가정합니다.

```
EXEC SQL DECLARE C CURSOR FOR...
EXEC SQL CONNECT TO X;
EXEC SQL OPEN C;
EXEC SQL FETCH C INTO...
EXEC SQL CONNECT TO Y;
EXEC SQL OPEN C;
EXEC SQL FETCH C INTO...
```

두 번째 OPEN C문은 커서 C의 다른 인스턴스를 참조하기 때문에 오류를 발생시키지 않습니다.

SELECT문은 커서가 열릴 때 평가됩니다. 같은 커서가 열리고 닫힌 다음, 다시 열리면 결과는 달라질 수 있습니다. 같은 SELECT문을 사용하는 여러 커서는 동시에 열릴 수 있습니다. 이는 독립적 활동으로 간주됩니다.

자료 블록: 자료의 효과적인 처리를 위해 데이터베이스 관리자는 읽기 전용 커서용으로 데이터를 블록할 수 있습니다. 배치된 UPDATE 또는 DELETE문에 커서가 사용되지 않을 경우, FOR READ ONLY로 선언되어야 합니다.

커서 감도: ALWCPYDTA 사전컴파일 옵션은 DYNAMIC SCROLL 커서에 대해 무시됩니다. 삽입, 갱신 및 삭제에 대한 감도가 유지되어야 하는 경우 임시 결과가 조회를 실행하는 데 필요하지 않으면 자료의 임시 사본은 결코 작성되지 않습니다.

REXX 커서: 호스트 변수가 REXX 프로시듀어 내의 DECLARE CURSOR문에 사용되면 DECLARE CURSOR는 PREPARE 및 EXECUTE의 오브젝트가 되어야 합니다.

예

예 1

C1을 표 DEPARTMENT에서 자료를 검색하기 위한 쿼리 커서로 선언합니다. 조회가 DECLARE CURSOR 문에 나타납니다.

```
EXEC SQL  DECLARE C1 CURSOR FOR
    SELECT DEPTNO, DEPTNAME, MGRNO
              FROM DEPARTMENT
              WHERE WHERE ADMRDEPT = 'A00';
```

예 2

C2를 STMT2라는 명령문의 커서로 선언합니다.

```
EXEC SQL  DECLARE C2 CURSOR FOR STMT2;
```

예 3

C3을 표 EMPLOYEE의 배치된 갱신에 사용될 커서로 선언합니다. 때로 커서를 닫지 않고 완료된 갱신을 확약할 수 있도록 합니다.

```
EXEC SQL  DECLARE C3 CURSOR WITH HOLD FOR
            SELECT *
              FROM EMPLOYEE
              FOR UPDATE OF WORKDEPT, PHONENO, JOB, EDLEVEL, SALARY;
```

갱신할 열을 명시적으로 지정하지 않고, 열 이름을 지정하지 않은 채 FOR UPDATE 절을 사용할 수 있습니다. 그러면 표의 모든 갱신가능 열을 갱신할 수 있습니다. 이 커서는 갱신가능하기 때문에, 표에서 행을 삭제하는 데에도 사용할 수 있습니다.

예 4

PL/I 프로그램에서 커서 C1을 사용하여 EMPPROJACT 표의 처음 네 열에서 한 번에 한 행씩 프로젝트(PROJNO)에 대한 값을 페치하여 그 값을 호스트 변수 EMP (CHARACTER(6)), PRJ (CHARACTER(6)), ACT (SMALLINT) 및 TIM (DECIMAL(5,2))에 작성합니다. 프로젝트의 값을 구하여 호스트 변수 SEARCH_PRJ(CHARACTER(6))에서 탐색합니다.

```
EXEC SQL  BEGIN DECLARE SECTION;
  DCL  EMP              CHAR(6);
  DCL  PRJ              CHAR(6);
  DCL  SEARCH_PRJ       CHAR(6);
  DCL  ACT              BINARY     FIXED(15);
  DCL  TIM              DEC        FIXED(5,2);
  DCL  SELECT_STMT      CHAR(200)  VARYING;
EXEC SQL  END DECLARE SECTION;


SELECT_STMT = 'SELECT EMPNO, PROJNO, ACTNO, EMPTIME ' ||
                'FROM EMPPROJACT ' ||
                'WHERE PROJNO = ?';
.
.
.
```

```
   EXEC SQL  PREPARE SELECT_PRJ FROM :SELECT_STMT;

   EXEC SQL  DECLARE C1 CURSOR FOR SELECT_PRJ;

   EXEC SQL  OPEN C1 USING :SEARCH_PRJ;

       EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM;

   IF SQLSTATE = '02000' THEN
     CALL DATA_NOT_FOUND;
ELSE     DO WHILE (SUBSTR(SQLSTATE,1,2) = '00'
     | SUBSTR(SQLSTATE,1,2) = '01');
       EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM;
   END;

   EXEC SQL  CLOSE C1;
   .
   .
   .
```

예 6

DECLARE CURSOR문은 커서 이름 C1과 SELECT의 결과를 연결합니다. C1은 갱신가능한 화면이동 커서입니다.

```
   EXEC SQL DECLARE C1 DYNAMIC SCROLL CURSOR FOR
       SELECT DEPTNO, DEPTNAME, MGRNO
       FROM CORPDATA.TDEPT
       WHERE ADMRDEPT = 'A00';
```

예 7

4 열에서 값을 페치하기 위해 커서를 선언하고 연속하는 (RR) 분리 레벨을 사용하여 값을 호스트 변수에 지정합니다.

```
   DECLARE CURSOR1 CURSOR FOR
     SELECT COL1, COL2, COL3, COL4
     FROM TBLNAME WHERE COL1 = :varname
     WITH RR
```

DECLARE GLOBAL TEMPORARY TABLE

DECLARE GLOBAL TEMPORARY TABLE문은 현재 어플리케이션 프로세스용으로 선언된 임시 표를 정의합니다. 선언된 임시 표 설명은 시스템 카탈로그에 표시되지 않습니다. 이 설명은 지속적이지 않으며 다른 세션과 공유될 수 없습니다. 동일한 이름의 선언된 글로벌 임시 표를 정의하는 각 세션은 임시 표에 대한 자체적인 고유 설명을 가집니다. 어플리케이션 프로세스가 종료되면, 임시 표가 삭제됩니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

LIKE 또는 AS 선택문 절이 지정되면 명령문의 권한부여 ID가 보유하는 권한은 LIKE 또는 AS 선택문 절에 지정된 표나 뷰에 대하여 적어도 다음 중 하나를 포함해야 합니다.

- 표나 뷰에 대한 SELECT 권한
- 표나 뷰에 대한 소유권
- 관리 권한

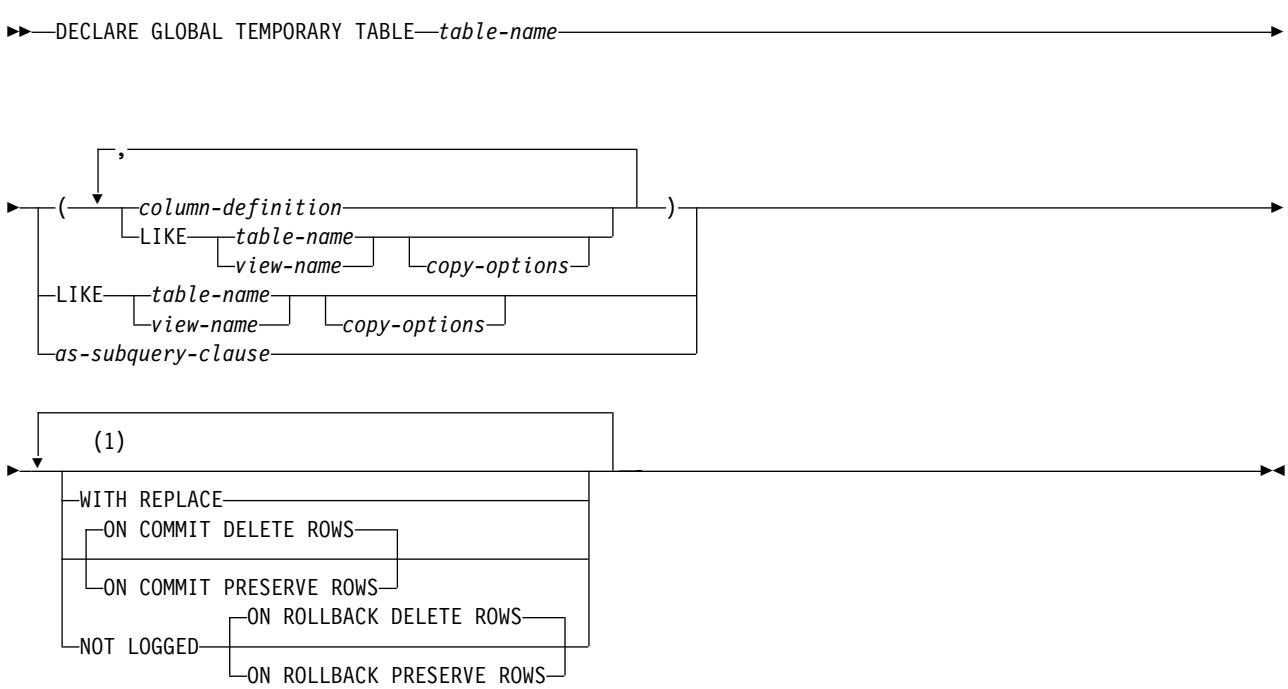
고유한 유형이 참조되다면, 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 명령문에서 식별된 각 고유한 유형의 경우
  - 고유한 유형에 대한 USAGE 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 중 하나가 참일 때 고유한 유형에 대해 USAGE 권한을 갖습니다.

- 고유한 유형의 소유자입니다.
- 고유한 유형에 대해 USAGE 권한을 부여받았습니다.
- 고유한 유형에 대해 *OBJOPR과 *EXECUTE의 시스템 권한을 부여받았습니다.

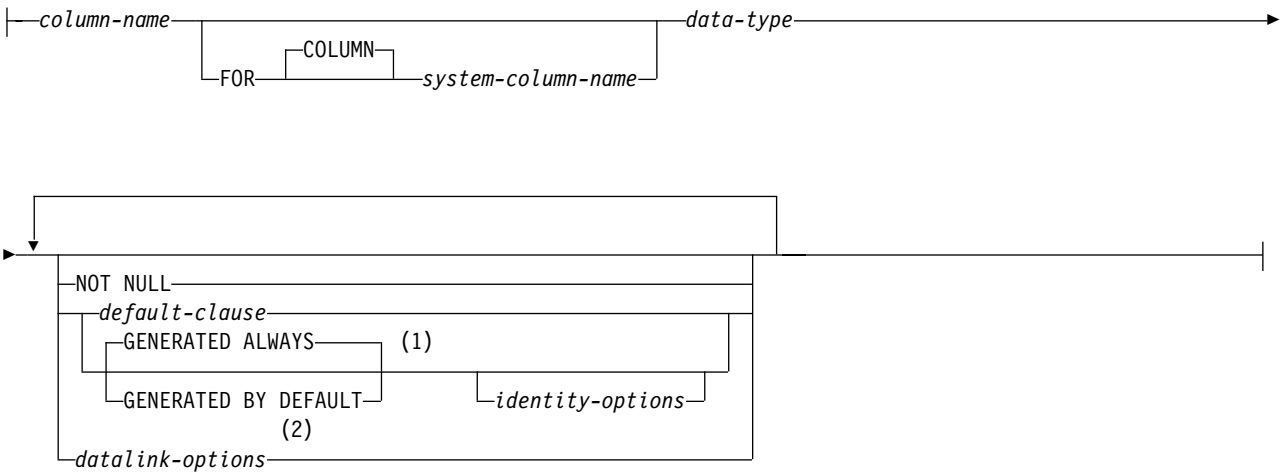
구문



주:

1 각 절은 한 번만 지정됩니다.

column-definition:

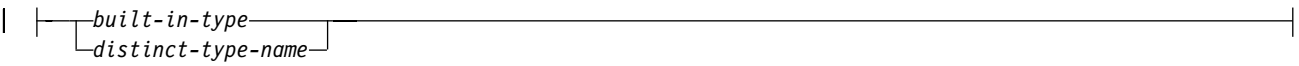


주:

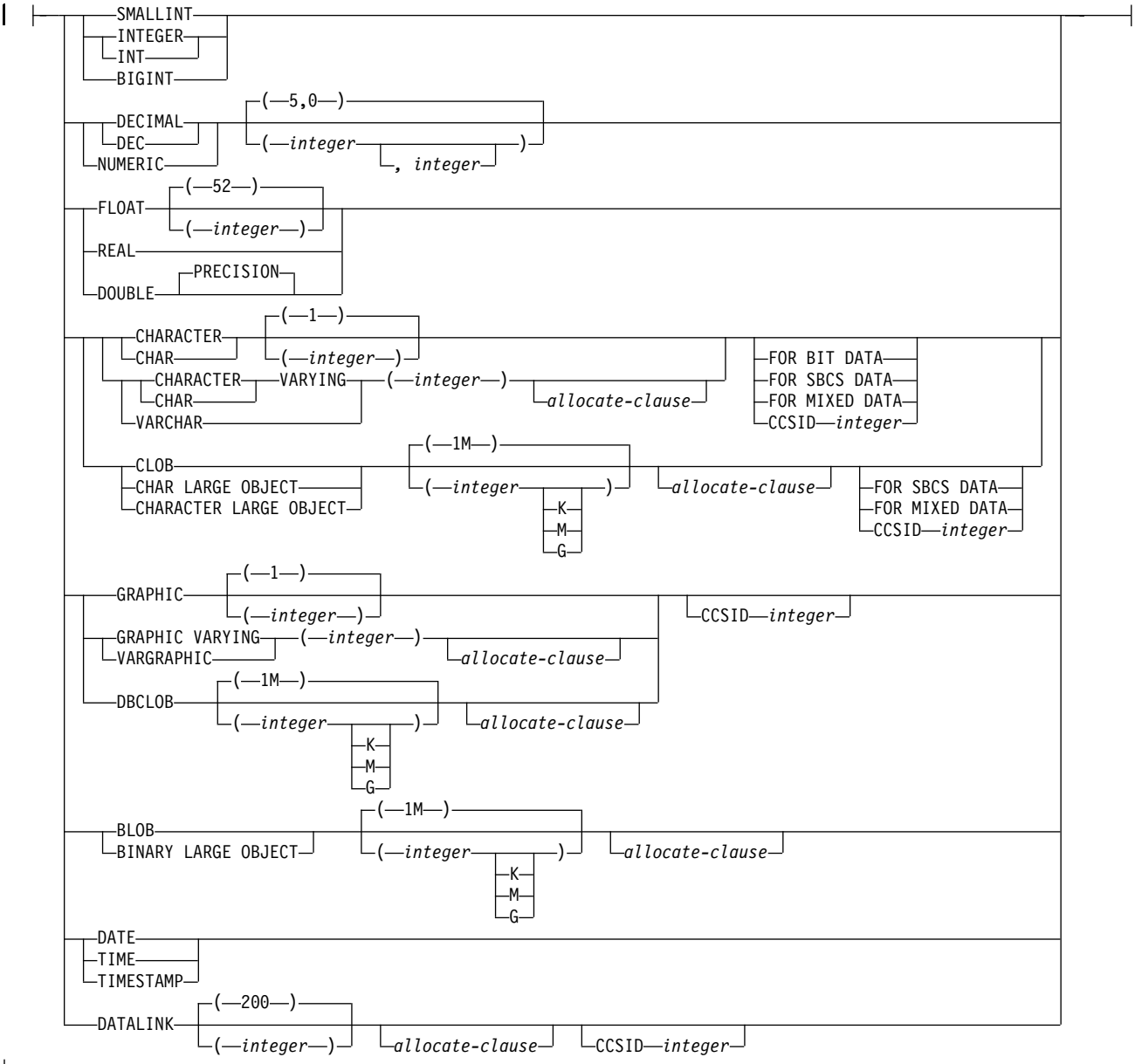
1 GENERATED는 열이 ID 열일 경우에만 지정할 수 있습니다.

2 datalink-options는 DATALINK와 DATALINK를 소스로 하는 distinct-type에 대해서만 지정될 수 있습니다.

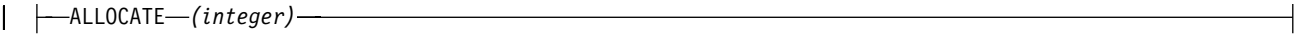
data-type:



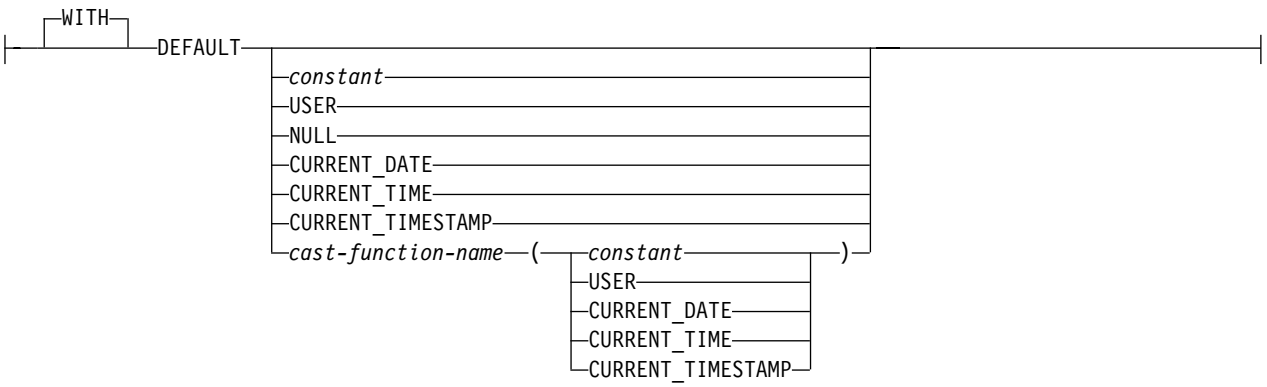
built-in-type:



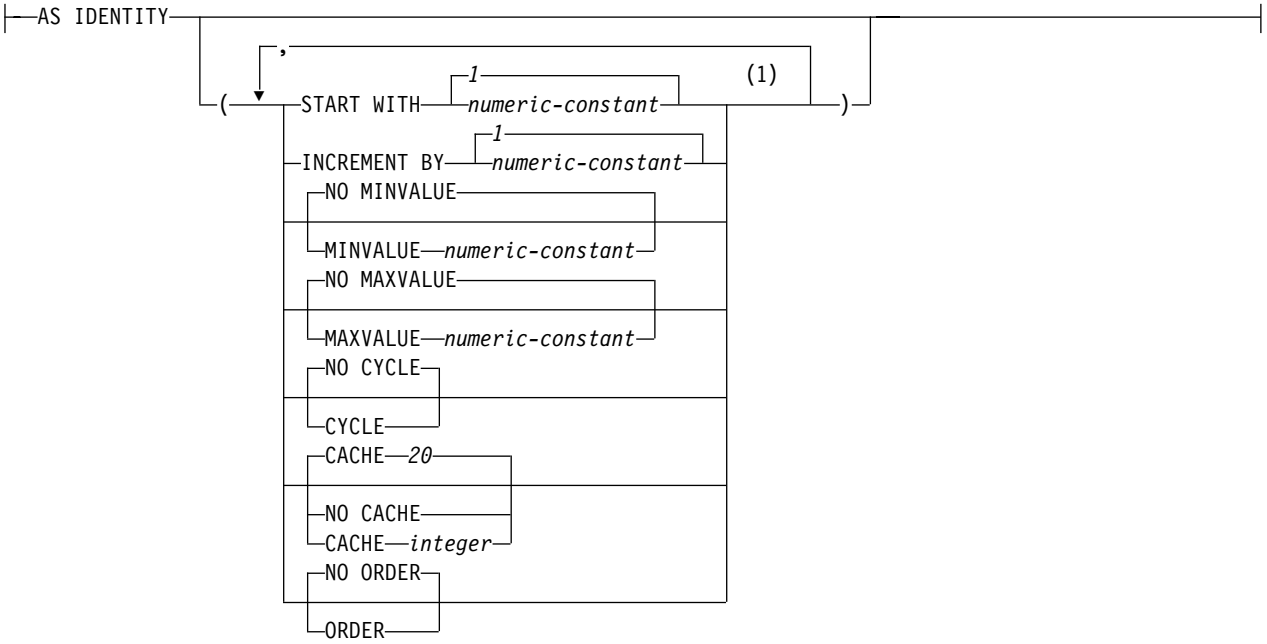
allocate절:



default절:



identity-options:

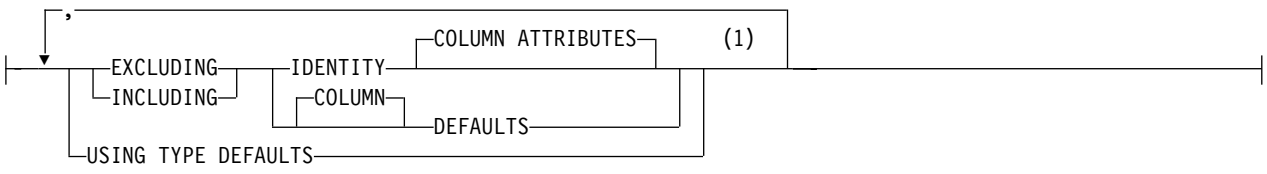


주:

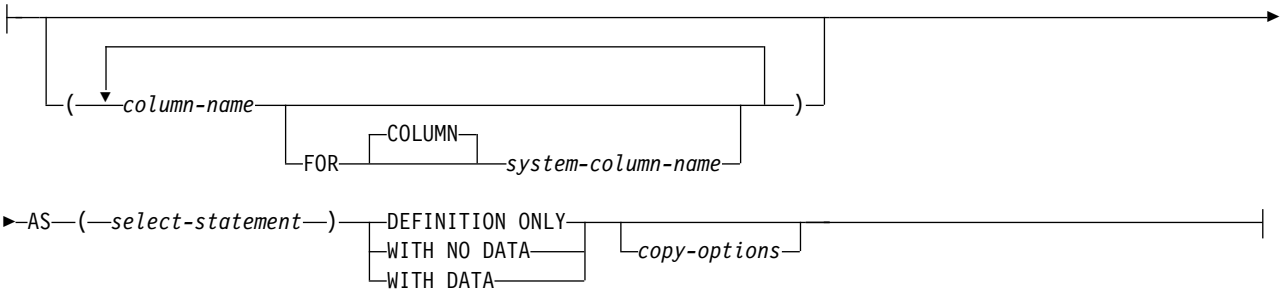
1 각 절은 한 번만 지정됩니다.

datalink-options:

copy-options:



as-subquery-clause:



주:

1 각 절은 한 번만 지정됩니다.

설명

table-name

임시 표를 명명합니다. 규정자를 명시적으로 지정할 경우 SESSION이어야 합니다. 규정자가 지정되지 않은 경우, SESSION에 내부적으로 정의됩니다. 영구 라이브러리에 표, 뷰, 색인 또는 별명이 같은 이름으로 이미 존재한다면 SESSION이라고 합니다.

- 선언된 임시 표는 여전히 SESSION.table-name으로 정의됩니다. 선언된 임시 표의 분석에 영구 라이브러리가 포함되지 않기 때문에 오류는 발생하지 않습니다.
- SESSION.table-name에 대한 어떠한 참조도 영구라기보다 임시라고 선언된 표, 뷰, 색인 또는 별명으로 SESSION.table-name의 이름을 분석할 것입니다.

표는 라이브러리 QTEMP에서 작성될 것입니다.

column-definition

열의 속성을 정의합니다. 하나 이상 8000개 이하의 열 정의가 있어야 합니다.

열의 레코드 행 바이트 수 합계는 32766 보다 커서는 안되고, VARCHAR이나 VARGRAPHIC 열이 지정된 경우에는 32740 보다 크면 안됩니다. 또한 LOB가 지정되면 열의 행 자료 바이트 수 합계는 3.5기가바이트보다 커서는 안됩니다. 자료 유형에 따른 열의 바이트 수에 대한 내용은 569 페이지의 『주』를 참조하십시오.

column-name

표의 열을 명명합니다. *column-name*을 규정해서는 안되면 표의 하나 이상의 열이나 system-column-name에 대해 같은 이름을 사용할 수 없습니다.

FOR COLUMN *system-column-name*

열에 대해 OS/400 이름을 제공합니다. 표의 하나 이상의 열이나 column-name에 대해 같은 이름을 사용할 수 없습니다.

system-column-name이 지정되지 않고 column-name이 유효한 system-column-name이 아니면, 시스템 열 이름이 생성됩니다. 시스템 열 이름이 생성되는 방법에 대한 자세한 내용은 572 페이지의 『열 이름 생성 규칙』을 참조하십시오.

data-type

열의 자료 유형을 지정합니다.

내장형

내장 자료 유형을 지정합니다. *내장 유형*의 설명541 페이지의 『CREATE TABLE』을 참조하십시오.

FILE LINK CONTROL을 가진 ROWID 열 또는 DATALINK 열은 전역 임시 표에 대해 지정될 수 없습니다.

NOT NULL

열이 널값을 포함하지 않도록 합니다. NOT NULL의 생략은 열이 널(null)이 될 수 있음을 의미합니다.

DEFAULT

열에 대한 디폴트 값을 지정합니다. 이 절은 *column-definition*에 한 번 이상 지정될 수 없습니다. ID 열(AS IDENTITY로 정의된 열)에는 디폴트 값을 지정할 수 없습니다. 데이터베이스 관리자는 ID 열의 디폴트 값을 생성합니다. 디폴트 키워드 다음에 값이 지정되지 않으면 다음이 적용됩니다.

- 열이 널이면 디폴트 값은 널값입니다.
- 열이 널이 아니면 디폴트는 열의 자료 유형에 따라 다릅니다.

자료 유형	디폴트 값
숫자	0
고정 길이 스트링	공백
가변 길이 스트링	길이가 0인 스트링
날짜	INSERT 시의 현재 날짜

자료 유형	디폴트 값
시간	INSERT 시의 현재 시간
시간소인	INSERT 시의 현재 시간소인
자료 링크	DLVALUE(‘ ’,‘URL’,‘ ’)에 해당하는 값
distinct-type	고유한 유형의 소스 유형에 대응하는 디폴트 값

*column-definition*에서 NOT NULL과 DEFAULT의 생략은 DEFAULT NULL의 내재적 스펙입니다.

constant

열에 대한 디폴트 값으로 상수를 지정합니다. 지정된 상수는 80 페이지의 『지정과 비교』에 설명된 대로 지정 규칙에 따라 열에 지정될 수 있는 값을 표시해야 합니다. 부동 소수점 상수를 SMALLINT, INTEGER, DECIMAL 또는 NUMERIC 열에 대해 사용해서는 안됩니다. 소수 상수는 열의 지정된 눈금보다 더 많은 소수점 이하의 자릿수를 가질 수 없습니다.

USER

INSERT 또는 UPDATE 시의 USER 특수 레지스터의 값을 열의 디폴트 값으로 지정합니다. 열의 자료 유형은 길이 속성이 USER 특수 레지스터의 길이 속성보다 크거나 같은 CHAR나 VARCHAR가 되어야 합니다.

NULL

열에 대한 디폴트 값으로 널을 지정합니다. NOT NULL이 지정되면 동일한 열 정의 내에 DEFAULT NULL이 지정되어서는 안됩니다.

CURRENT_DATE

열에 대한 디폴트 값으로 현재 날짜를 지정합니다. CURRENT_DATE가 지정되면 열의자료 유형은 DATE 또는 DATE에 기초한 고유한 유형이어야 합니다.

CURRENT_TIME

열에 대한 디폴트 값으로 현재 시간을 지정합니다. CURRENT_TIME이 지정되면 열의 자료 유형은 TIME 또는 TIME에 기초한 고유한 유형이어야 합니다.

CURRENT_TIMESTAMP

열에 대한 디폴트 값으로 현재 시간소인을 지정합니다.

CURRENT_TIMESTAMP가 지정되면 열의 자료 유형은 TIMESTAMP이거나 TIMESTAMP에 기초한 고유한 유형이어야 합니다.

cast-function-name

이 형식의 디폴트 값은 고유한 유형, BLOB, CLOB, DBCLOB, DATE, TIME 또는 TIMESTAMP 자료 유형으로 정의된 열과 함께만 사용될 수 있습니다. 다음 표는 이 *cast-function*이 허용되는 사용법을 설명한 것입니다.

자료 유형	캐스트 함수명
BLOB, CLOB 또는 DBCLOB를 기초로 하는 고유한 유형 N	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP를 기초로 하는 고유한 유형 N	N(N이 생성되었을 때 생성된 사용자 정의 캐스트 함수) ** 또는 DATE, TIME 또는 TIMESTAMP *
기타 자료 유형을 기초로 하는 고유한 유형 N	N(N이 생성되었을 때 생성된 사용자 정의 캐스트 함수) **
BLOB, CLOB 또는 DBCLOB	BLOB, CLOB 또는 DBCLOB *
DATE, TIME 또는 TIMESTAMP	DATE, TIME 또는 TIMESTAMP *

주:

* 함수 이름은 내재적 또는 명시적 스키마명 QSYS2를 갖는 자료 유형의 이름(또는 고유한 유형의 소스 유형)과 일치해야 합니다.

** 함수 이름은 열에 대한 고유한 유형 이름과 일치해야 합니다. 스키마 이름으로 규정했으면 고유한 유형에 대한 스키마 이름과 같아야 합니다. 규정하지 않았으면 함수 해석의 스키마 이름이 고유한 유형에 대한 스키마 이름과 같아야 합니다.

constant

상수를 인수로 지정합니다. 상수는 고유한 유형의 소스 유형 또는 고유한 유형이 아닌 경우 자료 유형에 대한 상수 규칙을 따라야 합니다. BLOB, CLOB, DBCLOB, DATE, TIME 및 TIMESTAMP 함수의 경우 상수는 스트링 상수이어야 합니다.

USER

INSERT 또는 UPDATE 시에 USER 특수 레지스터의 값을 열에 대한 디폴트 값으로 지정합니다. 열의 고유한 유형의 소스 유형에 대한 자료 유형은 길이 속성이 USER 특수 레지스터의 길이 속성보다 크거나 같은 CHAR나 VARCHAR가 되어야 합니다.

CURRENT_DATE

열에 대한 디폴트 값으로 현재 날짜를 지정합니다. CURRENT_DATE가 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 DATE가 되어야 합니다.

CURRENT_TIME

열에 대한 디폴트 값으로 현재 시간을 지정합니다. CURRENT_TIME이 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 TIME이 되어야 합니다.

CURRENT_TIMESTAMP

열에 대한 디폴트 값으로 현재 시간소인을 지정합니다.

CURRENT_TIMESTAMP가 지정되면 열의 고유한 유형의 소스 유형에 대한 자료 유형은 TIMESTAMP가 되어야 합니다.

GENERATED

데이터베이스 관리자가 열의 생성 값을 지정합니다. 열이 ID 열로 간주되는 경우 (AS IDENTITY절로 정의) GENERATED를 지정해야 합니다.

ALWAYS

표에 행이 삽입될 때 데이터베이스 관리자가 항상 열 값을 생성하도록 지정합니다. ALWAYS는 권장 값입니다.

BY DEFAULT

열에 대해 값이 지정되지 않은 경우에만 행이 삽입될 때 데이터베이스 관리자가 열 값을 생성하도록 지정합니다. 값이 지정되면, 데이터베이스 관리자는 값을 사용합니다.

ID 열에 데이터베이스 관리자는 지정된 값을 삽입하지만 ID 열에 해당 ID 열을 고유하게 지정하는 고유 제한조건이나 고유 인덱스가 있지 않은 경우 해당 값이 열의 고유 값임을 검증하지 않습니다.

AS IDENTITY

열은 표의 ID 열임을 지정합니다. 표는 하나의 ID 열만 가질 수 있습니다. AS IDENTITY는 열이 자료 유형이 스케일 0을 가진 정확한 숫자 유형(스케일 0인 SMALLINT, INTEGER, BIGINT, DECIMAL 또는 NUMERIC, 이러한 유형에 기초한 고유 유형)인 경우에만 지정할 수 있습니다.

ID 행은 묵시적으로 NOT NULL입니다.

START WITH *numeric-constant*

ID 열에 대해 생성된 첫 번째 값을 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 말아야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았다면, 디폴트 값은 오름차순의 경우 MINVALUE, 내림차순의 경우 MAXVALUE입니다. 이 값은 순서의 최대값 또는 최소값에 도달한 후 순서가 순환하는 값일 필요는 없습니다. START WITH절을 사용하면 주기에 사용된 범위의 바깥에 순서를 시작할 수 있습니다. 주기에 사용된 범위는 MINVALUE 및 MAXVALUE로 정의됩니다.

INCREMENT BY *numeric-constant*

ID 열의 연속 값 사이의 간격을 지정합니다. 이 값은 0이 아닌 양수이거나 음수일 수 있지만 큰 정수 상수 값을 초과하지 않으며, 소수점 오른쪽에 0이 아닌 숫자가 존재하지 않는 열에 지정할 수 있습니다. 디폴트는 1입니다.

값이 양수라면, ID 열의 값 순서는 오름차순입니다. 값이 음수라면, ID 열의 값 순서는 내림차순입니다.

MAXVALUE *numeric-constant*

이 ID 열에 대해 생성된 최대값인 숫자 상수를 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 최소값보다 커야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았을 때, 이 값은 오름차순의 경우 자료 유형의 최대값(DECIMAL의 경우 정밀도), 내림차순의 경우 START WITH 값(START WITH가 지정되지 않은 경우 -1)입니다.

MINVALUE *numeric-constant*

이 ID 열에 대해 생성된 최소값인 숫자 상수를 지정합니다. 이 값은 이 열에 지정할 수 있는 양수이거나 음수일 수 있지만, 최대값보다 적어야 합니다.

ID 열이 정의된 경우 값이 명시적으로 지정되지 않았을 때, 이 값은 오름차순의 경우 START WITH 값(START WITH가 지정되지 않은 경우 -1), 내림차순의 경우 자료 유형의 최소값(DECIMAL의 경우 정밀도)입니다.

CACHE 또는 NO CACHE

사전 할당된 일부 값을 메모리에 보존할 것인지 여부를 지정합니다. 값을 사전 할당하여 캐시에 저장하면 표에 행을 삽입하는 기능이 좋아집니다.

CACHE *integer*

데이터베이스 관리자가 사전에 할당하여 메모리에 보관하는 ID 열 순서 값 수를 지정합니다. 지정할 수 있는 최소값은 2이고, 최대값은 정수로 표현될 수 있는 가장 큰 값입니다. 디폴트는 20입니다.

시스템 오류가 발생하면, 지정될 캐시된 모든 ID 열 값이 유실되어 다시는 사용되지 않습니다. 따라서, CACHE에 대해 지정된 값은 시스템 오류 중 유실될 수 있는 ID 열의 최대 값 수를 나타냅니다.

NO CACHE

ID 열의 값이 사전에 할당되지 않도록 지정합니다.

CYCLE 또는 NO CYCLE

순서의 최대값 또는 최소값에 도달한 후 이 ID 열이 계속 값을 생성하는지 여부를 지정합니다.

CYCLE

최대값이나 최소값에 도달한 후 이 열에 대해 값이 계속 생성됨을 지정합니다. 이 옵션이 사용된 경우, 오름차순이 순서의 최대값에 도달한 후 해당 최소값을 생성합니다. 내림차순이 순서의 최소값에 도달하고 나면, 최대값이 생성됩니다. 열의 최대값과 최소값은 주기에 사용된 범위를 결정합니다.

CYCLE이 유효한 경우, 데이터베이스 관리자는 ID 열에 대해 중복값을 생성할 수 있습니다. ID 열에 대해 제한조건 또는 고유 색인이 존재하는데 이 열에 대해 고유하지 않은 값이 생성되면 오류가 발생합니다.

NO CYCLE

순서의 최대값이나 최소값에 도달한 후 ID 열에 대해 값이 생성되지 않도록 지정합니다. 이것이 디폴트 값입니다.

ORDER 또는 NO ORDER

요청 순서대로 ID 값이 생성되도록 지정합니다.

ORDER

요청 순서대로 값이 생성되도록 지정합니다.

NO ORDER

요청 순서대로 값이 생성될 필요가 없도록 지정합니다. 이것이 디폴트 값입니다.

datalink-options

DATALINK 자료 유형과 연관된 옵션을 지정합니다.

LINKTYPE URL

URL(Uniform Resource Locator)로 링크의 유형을 정의합니다.

NO LINK CONTROL

링크된 파일이 존재하는지 판별하기 위해 검사하지 않음을 지정합니다. URL 구문만 검사됩니다. 링크된 파일을 제어하는 데이터베이스 관리자가 없습니다.

LIKE

table-name 또는 *view-name*

지정된 표나 뷰에 정의된 열이 이 표에 포함되도록 지정합니다. LIKE 절에 지정된 *table-name* 또는 *view-name*은 이미 서버에 있는 표나 뷰를 식별해야 합니다.

LIKE를 사용하면 n 열을 내재적으로 정의하게 되는데, n은 식별된 표나 뷰의 열의 수입니다. 내재적 정의에는(해당 자료 유형에 적용할 수 있는 경우) 다음과 같은 n 열의 속성이 포함됩니다.

- 열 이름 (및 시스템 열 이름)
- 자료 유형, 길이, 정밀도 및 스케일
- CCSID

LIKE 절이 *table-name* 바로 다음에 지정되고 괄호 안에 들어 있지 않다면 다음의 열 속성도 포함되며, 그렇지 않은 경우 포함되지 않습니다(디폴트 값 및 ID 속성은 또한 *copy-options*를 사용하여 제어될 수 있습니다).

- *table-name*이 지정되고 *view-name*이 지정되지 않은 경우 디폴트 값
- ID 속성
- 널 가능성
- 열 머리말 및 텍스트(717 페이지의 『LABEL』 참조)

지정된 표나 뷰가 비SQL 작성의 실제 파일 또는 논리 파일인 경우 비SQL 속성이 제거됩니다. 예를 들어 날짜와 시간 형식은 ISO로 변환됩니다.

내재적 정의에는 식별된 표나 뷰의 다른 선택적 속성을 포함하지 않습니다. 예를 들어 새로운 표는 표의 1차 키나 외부 키를 자동으로 포함하지 않습니다. 선택적 절이 명시적으로 지정된 경우에만 새로운 표가 이들 및 기타 선택적 속성을 갖게 됩니다.

as-subquery-clause

column-name

표의 열을 명명합니다. *column-name*을 규정해서는 안되면 표의 하나 이상의 열이나 system-column-name에 대해 같은 이름을 사용할 수 없습니다.

FOR COLUMN *system-column-name*

열에 대해 OS/400 이름을 제공합니다. 표의 하나 이상의 열이나 column-name에 대해 같은 이름을 사용할 수 없습니다.

system-column-name이 지정되지 않고 column-name이 유효한 system-column-name이 아니면, 시스템 열 이름이 생성됩니다. 시스템 열 이름이 생성되는 방법에 대한 자세한 내용은 572 페이지의 『열 이름 생성 규칙』을 참조하십시오.

select-statement

표의 열이 select-statement가 실행될 경우 select-statement문의 도출된 결과 표에 표시될 열과 동일한 이름과 설명을 가지도록 지정합니다. AS select-statement의 사용은 표에 대한 n 열의 묵시적 정의입니다. 여기서 n은 select-statement로부터 나온 결과 열의 수입니다. 내재적 정의에는(해당 자료 유형에 적용할 수 있는 경우) 다음과 같은 n 열의 속성이 포함됩니다.

- 열 이름 (및 시스템 열 이름)
- 자료 유형, 길이, 정밀도 및 스케일
- CCSID
- 널 가능성
- 열 머리말 및 텍스트(717 페이지의 『LABEL』 참조)

다음 속성은 포함되지 않습니다(*copy-options*를 사용하여 디폴트 값 및 ID 속성을 포함시킬 수 있습니다).

- 디폴트 값
- ID 속성

내재적 정의에는 식별된 표나 뷰의 다른 선택적 속성을 포함하지 않습니다. 예를 들어 새로운 표는 표의 1차 키나 외부 키를 자동으로 포함하지 않습니다. 선택적 절이 명시적으로 지정된 경우에만 새로운 표가 이들 및 기타 선택적 속성을 갖게 됩니다.

표의 묵시적으로 정의된 열은 *select-statement*의 결과 표로부터 열의 이름을 계승합니다. 따라서, 열 이름은 *select-statement* 또는 모든 결과 열의 열 이름 리스트에 지정되어야 합니다. 표현식, 상수, 함수에서 도출된 결과 열의 경우, *select-statement*는 결과 열 바로 다음에 AS column-name절을 포함하거나 열 리스트에서 *select-statement* 앞에 이름이 지정되어야 합니다.

*select-statement*는 호스트 변수를 참조하거나 매개변수 마커(의문 부호)를 포함할 수 없습니다.

WITH DATA

*select-statement*이 실행됨을 지정합니다. 표가 작성된 후, *select-statement*의 결과 표 행은 표에 자동으로 삽입됩니다.

WITH NO DATA 또는 DEFINITION ONLY

*select-statement*이 실행되지 않음을 지정합니다. 따라서, 자동으로 표를 채울 행 세트를 가진 결과 표가 없습니다.

copy-options

INCLUDING IDENTITY COLUMN ATTRIBUTES

표가 *select-statement*, *table-name* 또는 *view-name*에 기인한 열의 ID 속성(있는 경우)을 계승하도록 지정합니다. 일반적으로, 표, 보기 또는 select-statement의 해당 열의 요소가 기본 표 열을 ID 속성과 직접/간접적으로 맵핑하는 보기 열의 이름 또는 표 열의 이름인 경우, ID 열이 복사됩니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES절이 AS *select-statement* 절로 지정된 경우, 다음과 같은 경우 새 표의 열은 ID 속성을 계승하지 않습니다.

- *select-statement*의 선택 리스트에는 ID 열 이름의 복수 인스턴스(즉, 동일한 열을 두 번 이상 선택)가 포함됩니다.
- *select-statement*의 선택 리스트에는 복수 ID 열이 포함됩니다(즉, 결합 관련).
- ID 열은 선택 리스트에서 표현식에 포함됩니다.
- *select-statement*는 집합 연산(결합)을 포함합니다.

INCLUDING IDENTITY가 지정되지 않은 경우, 표에는 ID 열이 없습니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES

표가 *select-statement*, *table-name* 또는 *view-name*에 기인한 열의 ID 속성(있는 경우)을 계승하지 않도록 지정합니다.

INCLUDING COLUMN DEFAULTS

표가 *select-statement*, *table-name* 또는 *view-name*에 기인한 열의 디폴트 값을 계승하도록 지정합니다. 디폴트 값은 INSERT에 값이 지정되지 않을 때 열에 지정되는 값입니다.

USING TYPE DEFAULTS를 지정한 경우, INCLUDING COLUMN DEFAULTS를 지정하지 마십시오.

INCLUDING COLUMN DEFAULTS가 지정되지 않은 경우, 표는 디폴트 값을 계승하지 않습니다.

EXCLUDING COLUMN DEFAULTS

표가 *select-statement*, *table-name* 또는 *view-name*에 기인한 열의 디폴트 값을 계승하지 않도록 지정합니다.

USING TYPE DEFAULTS

표의 디폴트 값이 *select-statement*, *table-name* 또는 *view-name*에 기인한 열의 자료 유형에 기초하도록 지정합니다. 열이 널이면 디폴트 값은 널값입니다. 반면에 디폴트 값는 다음과 같습니다.

자료 유형	디폴트 값
숫자	0
고정 길이 스트링	공백
가변 길이 스트링	길이가 0인 스트링
날짜	INSERT 시의 현재 날짜
시간	INSERT 시의 현재 시간
시간소인	INSERT 시의 현재 시간소인
자료 링크	DLVALUE(‘ ’,‘URL’,‘ ’)에 해당하는 값
distinct-type	고유한 유형의 소스 유형에 대응하는 디폴트 값

INCLUDING COLUMN DEFAULTS가 지정된 경우 USING TYPE DEFAULTS를 지정하지 마십시오.

WITH REPLACE

선언된 임시 표가 지정된 이름으로 이미 존재할 경우 기존 표를 이 명령문이 정의하는 임시 표로 바꾸도록 지정합니다(그리고 기존 표의 모든 행 삭제).

WITH REPLACE가 지정되지 않은 경우, 지정된 이름은 현재 세션에 이미 존재하는 선언된 글로벌 임시 표를 나타내는 것이 아니어야 합니다.

ON COMMIT

COMMIT 조작이 수행될 때 글로벌 임시 표에 수행된 조치를 지정합니다.

선언된 글로벌 임시 표가 분리 레벨 No Commit(NC)로 열려 있거나 COMMIT HOLD 조작이 수행된 경우 ON COMMIT절은 적용되지 않습니다.

DELETE ROWS

WITH HOLD 커서가 표에서 열려 있는 경우 표의 모든 행이 삭제됩니다. 이것이 디폴트 값입니다.

PRESERVE ROWS

표의 행이 보존될 것입니다.

NOT LOGGED

표 작성을 포함한 표 변경사항이 기록되지 않습니다. ROLLBACK(또는 ROLLBACK TO SAVEPOINT) 조작이 수행되었으며 표가 작업 단위(저장점)에서 변경된 경우, 변경사항은 롤백되지 않습니다. 표가 작업 단위(또는 저장점) 내에서 작성된 경우, 해당 표는 제거됩니다. 표가 작업 단위(또는 저장점)에서 제거된 경우, 해당 표는 행 없이 복원됩니다.

ON ROLLBACK

ROLLBACK 조작이 수행될 때 글로벌 임시 표에 수행된 조치를 지정합니다.

선언된 글로벌 임시 표가 분리 레벨 No Commit(NC)로 열려 있거나 ROLLBACK HOLD 조작이 수행된 경우 ON ROLLBACK절은 적용되지 않습니다.

DELETE ROWS

표의 모든 행이 삭제될 것입니다. 이것이 디폴트 값입니다.

PRESERVE ROWS

표의 행이 보존될 것입니다.

주

- **인스턴스화, 범위 및 종료:** P가 어플리케이션 프로세스를 나타내고 T가 P의 어플리케이션 프로그램에서 선언된 임시 표를 나타낸다고 가정합니다.
  - P의 프로그램이 DECLARE GLOBAL TEMPORARY TABLE문을 실행하면, T의 빔 인스턴스가 작성됩니다.
  - P의 모든 프로그램은 T를 참조할 수 있고 모든 참조는 T의 동일 인스턴스를 참조합니다.(DECLARE GLOBAL TEMPORARY문이 SQL 함수, SQL 프로시듀어 또는 트리거의 복합문내에 지정된 경우, 선언된 임시 표의 범위는 어플리케이션 프로세스이며 복합문이 아닙니다.)

    T가 원격 서버에서 선언된 경우, T의 참조는 T를 선언할 때 사용된 것과 동일한 연결을 사용해야 하며 해당 연결은 T가 선언된 후에 종료되어서는 안됩니다. T가 선언된 데이터베이스 서버로의 연결이 종료되면, T가 삭제되며 인스턴스화된 해당 행이 폐기됩니다.
  - T가 ON COMMIT DELETE ROWS절과 함께 정의되었다면, 확약 조작이 P의 작업 단위를 종료했으며 P의 프로그램 중 어느것도 T에 종속된 WITH HOLD 커서를 열어둔 것이 없는 경우 모든 행이 삭제됩니다.
  - T가 ON ROLLBACK DELETE ROWS절과 함께 정의되었다면, 롤백 조작이 P의 작업 단위를 종료한 경우 모든 행이 삭제됩니다.
  - T를 선언한 어플리케이션 프로세스가 종료되면, T는 제거되고 해당 행은 폐기됩니다.

- **임시 표 소유권:** 표의 *소유자*는 작업이 명령문을 실행하는 작업의 사용자 프로파일입니다.
- **임시 표 권한:** 선언된 임시 표가 정의되었을 때, PUBLIC에는 표에 대한 모든 표 권한 및 표를 삭제할 권한이 암시적으로 부여됩니다.
- **다른 SQL문에서 선언된 임시 표 참조:** 여러 SQL문이 선언된 임시 표를 지원합니다. DECLARE GLOBAL TEMPORARY TABLE 이외의 SQL문에서 선언된 임시 표를 참조하려면, 표는 내재적으로 또는 명시적으로 SESSION으로 규정되어야 합니다.

  SESSION을 표 이름 규정자로 사용하지만 어플리케이션 프로세스에 표 이름에 대한 DECLARE GLOBAL TEMPORARY TABLE문이 포함되어 있지 않은 경우, 데이터베이스 관리자는 사용자가 선언된 임시 표를 참조하지 않는다고 가정합니다. 데이터베이스 관리자는 영구 표에 대한 표 참조를 분석합니다.
- **선언된 임시 표 사용에 대한 제한사항:**
  - 선언된 임시 표는 ALTER TABLE, COMMENT, CREATE TRIGGER, GRANT, LABEL, LOCK, RENAME 또는 REVOKE문에 지정할 수 없습니다.
  - 선언된 임시 표는 참조 제한조건의 상위 표로 지정할 수 없습니다.
  - 선언된 임시 표가 CREATE INDEX 또는 CREATE VIEW문에서 참조된 경우, SESSION에 색인 또는 뷰를 작성해야 합니다(또는 라이브러리 QTEMP).

예

예 1

사원 번호, 임금, 수수료, 보너스에 대한 열 정의가 있는 선언된 임시 표를 정의하십시오.

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.TEMP_EMP
  (EMPNO    CHAR(6)     NOT NULL,
            SALARY    DECIMAL(9, 2),
            BONUS     DECIMAL(9, 2),
            COMM      DECIMAL(9, 2))
  ON COMMIT PRESERVE ROWS
```

예 2

기본 표 USER1.EMPTAB가 존재하고 이 표에 세 개의 열이 들어 있으며 그 중 하나는 ID 열인 경우, 기본 표와 동일한 열 이름 및 속성(ID 속성 포함)을 가진 임시 표를 정의하십시오.

```
DECLARE GLOBAL TEMPORARY TABLE TEMPTAB1
  LIKE USER1.EMPTAB
  INCLUDING IDENTITY
  ON COMMIT PRESERVE ROWS
```

위의 예에서, 데이터베이스 관리자는 SESSION을 TEMPTAB1의 내재적 규정자로 사용합니다.

DECLARE PROCEDURE

DECLARE PROCEDURE문은 외부 프로시듀어를 정의합니다.

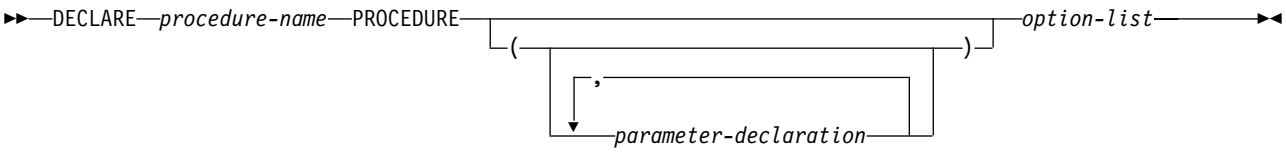
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. REXX에 지정되어서는 안됩니다.

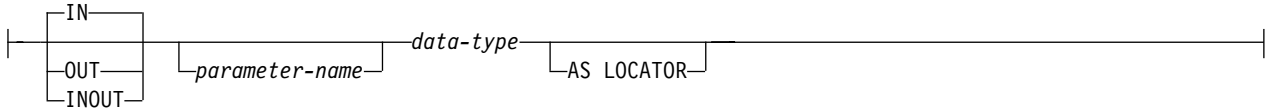
권한부여

없음

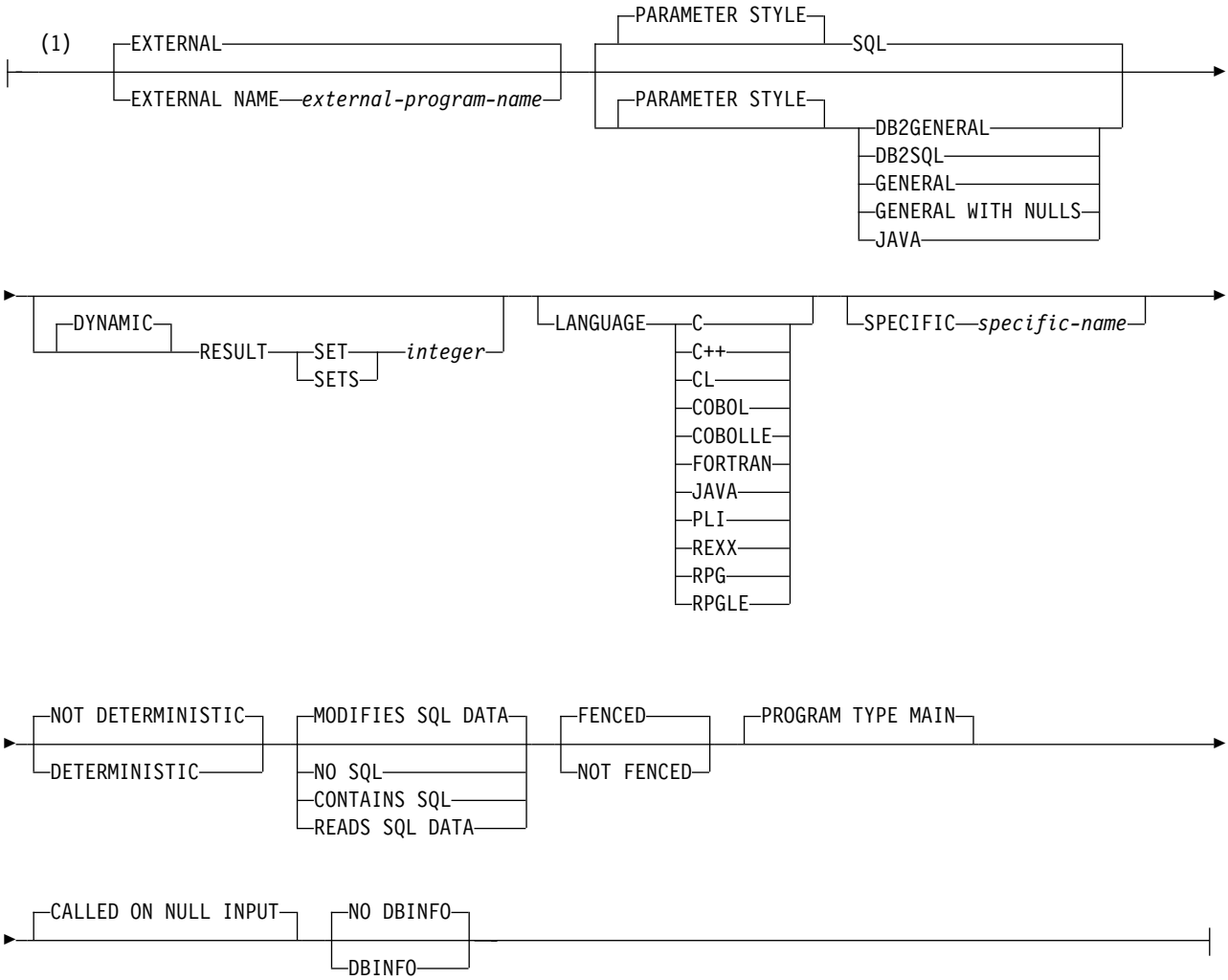
구문



parameter-declaration:



option-list:



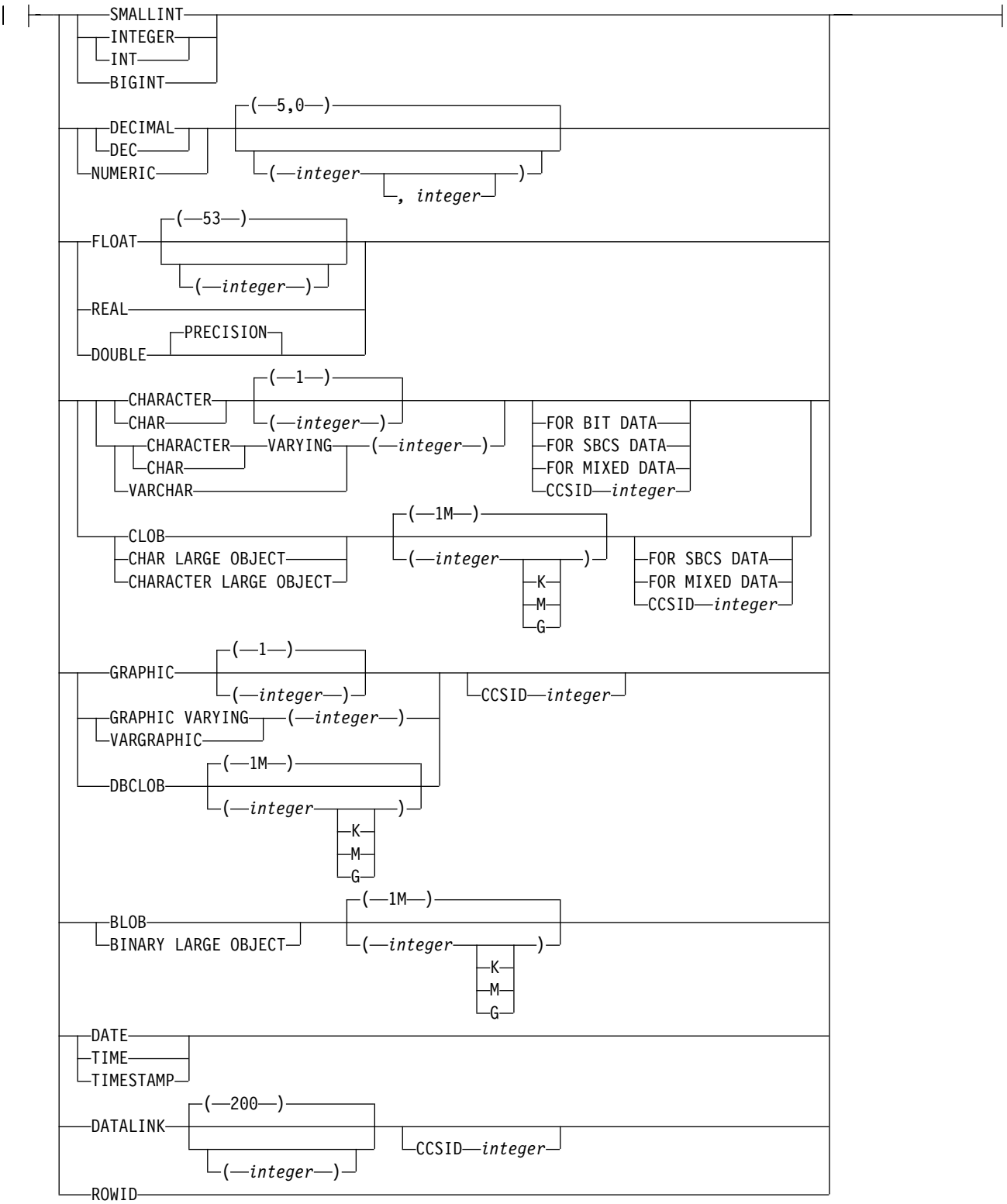
주:

1 옵션 절은 다른 순서로 지정할 수 있습니다.

DECLARE PROCEDURE

data-type:

built-in-type:



설명

procedure-name

프로시듀어를 명명합니다. 이 이름은 소스 프로그램에 선언된 다른 프로시듀어의 이름과 동일해서는 안됩니다.

(*parameter-declaration,...)*

프로시듀어의 매개변수 수와 각 매개변수의 자료 유형을 지정합니다. 프로시듀어의 매개변수는 입력, 출력, 입출력 모두에 사용할 수 있습니다. 필수는 아니지만, 각 매개변수에 이름을 부여할 수 있습니다.

SQL 프로시듀어에서 사용할 수 있는 매개변수의 최대수는 255입니다.

IN

프로시듀어로의 입력 매개변수로서 매개변수를 지정합니다. 프로시듀어내에서 매개변수의 변경은 제어가 리턴될 때 SQL 어플리케이션에 대해서 유효하지 않습니다.[53]

OUT

매개변수를 프로시듀어가 리턴하는 출력 매개변수로 지정합니다.

자료 링크나 자료 링크를 기초로 하는 고유한 유형은 출력 매개변수로 지정되지 않습니다.

INOUT

프로시듀어의 매개변수를 입력 및 출력 매개변수로 지정합니다.

자료 링크나 자료 링크를 기초로 하는 고유한 유형은 입력 및 출력 매개변수로 지정되지 않습니다.

parameter-name

매개변수를 명명합니다. 이름은 프로시듀어에 대한 다른 *parameter-name*과 같아질 수 없습니다.

data-type

매개변수의 자료 유형을 지정합니다.

자료 유형은 언어 절에 지정된 언어에 대해 유효해야 합니다. 모든 자료 유형은 SQL 프로시듀어에 대해 유효합니다. 자료 유형은 외부 프로시듀어에 대해 유효하지 않습니다. 자료 유형에 대한 자세한 정보는 541 페이지의 『CREATE TABLE』 및 SQL 프로그래밍 개념 책을 참조하십시오.

CCSID가 지정되면 매개변수는 함수로 전달되기 전에 해당 CCSID로 변환됩니다. CCSID가 지정되지 않으면 프로시듀어가 호출될 때 현재 서버에서 디폴트 CCSID에 의해 CCSID가 판별됩니다.

53. 언어 유형이 REXX이면 모든 매개변수는 입력 매개변수이어야 합니다.

AS LOCATOR

입력 매개변수가 실제 값이 아니라 값에 대한 로케이터임을 지정합니다. 입력 매개변수가 LOB 자료 유형이나 LOB 자료 유형을 기초로 한 고유한 유형을 가질 경우에만 AS LOCATOR를 지정할 수 있습니다.

DYNAMIC RESULT SETS *integer*

프로시듀어에서 리턴될 수 있는 결과 설정의 최대수를 지정합니다. *integer*는 0보다 크거나 같아야 합니다. 0이 지정되면 결과 세트는 리턴되지 않습니다. 프로시듀어는 임의 숫자의 결과 세트를 가질 수 있지만, 언제든지 100개의 프로시듀어만이 페치되기를 기다리는 결과 세트를 가질 수 있습니다. SET RESULT SETS문이 실행된 경우, 리턴되는 결과 수는 이 키워드 및 SET RESULTS SET문에 지정된 최소 결과 세트 수입니다.

프로시듀어가 iSeries Access 클라이언트나 SQL 호출 레벨 인터페이스에서 호출되면 결과 세트만 리턴됩니다. 결과 세트에 대한 자세한 정보는 790 페이지의 『SET RESULT SETS』를 참조하십시오.

LANGUAGE

외부 프로그램이 작성되는 언어를 지정합니다. 외부 프로그램이 REXX 프로시듀어이면 언어 절은 필요하지 않습니다.

LANGUAGE가 지정되지 않으면 LANGUAGE는 외부 프로그램과 연관된 프로그램 속성 정보에서 판별됩니다. 프로그램과 관련된 프로그램 속성 정보가 인식할 수 있는 언어를 식별하지 못하면 언어는 C로 가정됩니다.

C

외부 프로그램이 C로 작성됩니다.

C++

외부 프로그램이 C++로 작성됩니다.

CL

외부 프로그램이 CL로 작성됩니다.

COBOL

외부 프로그램이 COBOL로 작성됩니다.

COBOLLE

외부 프로그램은 ILE COBOL로 작성됩니다.

FORTRAN

외부 프로그램이 FORTRAN으로 작성됩니다.

JAVA

외부 프로그램이 JAVA로 작성됩니다.

PLI

외부 프로그램이 PL/I으로 작성됩니다.

REXX
외부 프로그램은 REXX 프로시듀어입니다.

RPG
외부 프로그램이 RPG로 작성됩니다.

RPGLE
외부 프로그램은 ILE RPG로 작성됩니다.

SPECIFIC *specific-name*
프로시듀어를 고유하게 식별하는 규정되거나 규정되지 않은 이름을 지정합니다. 내재적 또는 명시적 규정자를 포함하여 *specific-name*은 *procedure-name*과 같아야 합니다.

규정자가 지정되지 않으면 *procedure-name*의 내재적 또는 명시적 규정자가 사용됩니다. 규정자가 지정되면 규정자는 *procedure-name*의 명시적 또는 내재적 규정자와 같아야 합니다.

*specific-name*이 지정되지 않으면 프로시듀어명과 같아야 합니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**
프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 동일한 결과를 리턴하는지 여부를 지정합니다.

NOT DETERMINISTIC
데이터베이스의 참조 데이터가 변경되지 않은 경우, 프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 항상 동일한 결과를 리턴합니다.

DETERMINISTIC
데이터베이스의 참조 데이터가 변경되지 않은 경우, 프로시듀어가 동일한 IN 및 INOUT 인수로 호출될 때마다 동일한 결과를 리턴하지 않습니다.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA 또는 **NO SQL**
프로시듀어 또는 이 프로시듀어에서 호출되는 루틴에서 실행될 수 있는 SQL문(있는 경우)을 지정합니다. 913 페이지의 부록 F『SQL문의 특성』에서 각 자료 액세스 표시 하에 실행할 수 있는 SQL문 리스트에 대한 세부사항을 참조하십시오.

CONTAINS SQL
프로시듀어에서 SQL 데이터를 읽거나 수정하지 않는 SQL문을 실행할 수 있도록 지정합니다.

NO SQL
프로시듀어가 SQL문을 실행하지 않음을 지정합니다.

READS SQL DATA
프로시듀어에 SQL 데이터를 수정하지 않는 SQL문을 포함시킬 수 있도록 지정합니다.

MODIFIES SQL DATA

프로시듀어는 프로시듀어에 지원되지 않는 명령문을 제외한 모든 SQL문을 실행할 수 있음을 지정합니다.

CALLED ON NULL INPUT

매개변수 값이 널(null)인 경우 프로시듀어가 호출되도록 지정합니다.

FENCED 또는 **NOT FENCED**

이 매개변수는 다른 제품과의 호환을 위해 허용되며, iSeries용 DB2 UDB에 의해 사용되지 않습니다.

PROGRAM TYPE MAIN

프로시듀어가 기본 루틴을 실행함을 지정합니다.

DBINFO

데이터베이스 관리자가 상태 정보가 들어 있는 구조를 프로시듀어에 전달해야 하는지를 지정합니다. 표 49에는 DBINFO 구조에 대한 설명이 나와 있습니다. DBINFO 구조에 대한 자세한 정보는 QSYSINC.H의 포함 파일 SQLUDF에 있습니다.

DBINFO는 PARAMETER STYLE DB2SQL와 함께만 사용될 수 있습니다.

표 49. DBINFO 필드

필드	자료 유형	설명
관계형 데이터베이스	VARCHAR(128)	현재 서버의 이름.
권한부여 ID	VARCHAR(128)	실행시 권한부여 ID.
CCSID 정보	INTEGER INTEGER INTEGER INTEGER CHAR(8)	작업의 CCSID 정보. 다음 정보로 CCSID를 식별합니다. • SBCS CCSID • DBCS CCSID • Mixed CCSID • 세 개의 CCSID 중 가정 적절한 CCSID 지정 • 예약 CCSID가 DECLARE PROCEDURE문의 매개변수에 명시적으로 지정되지 않는 경우 입력 스트링은 함수가 실행되는 시점의 작업의 CCSID로 코드화되는 것으로 가정됩니다. 입력 스트링의 CCSID가 매개변수의 CCSID와 같지 않다면 외부 함수로 전달된 입력 스트링은 외부 프로그램을 호출하기 전에 변환될 것입니다.
목표 열	VARCHAR(128) VARCHAR(128) VARCHAR(128)	프로시듀어 호출에 적용될 수 없음.
버전 및 릴리스	CHAR(8)	데이터베이스 관리자의 버전 릴리스 및 수정 레벨.
플랫폼	INTEGER	서버의 플랫폼 유형.

EXTERNAL NAME *external-program-name*

프로시듀어가 CALL문에 의해 호출될 때 실행될 프로그램을 지정합니다. 프로그램 명은 서버에 있는 프로그램을 식별해야 합니다. 프로그램은 ILE 서비스 프로그램이 될 수 없습니다.

이름의 유효성은 서버에서 검사됩니다. 이름의 형식이 올바르지 않으면 오류가 리턴됩니다.

external-program-name이 지정되지 않으면 외부 프로그램명은 프로시듀어명과 동일하다고 가정됩니다.

PARAMETER STYLE

함수로(부터) 값을 전달하고 리턴하는 데 사용되는 규약을 지정합니다.

SQL

CALL문의 매개변수 외에 몇몇 추가 매개변수가 프로시듀어에 전달됨을 지정합니다. 매개변수는 다음 순서로 정의됩니다.

- 처음의 N개의 매개변수는 DECLARE PROCEDURE문에 지정되는 매개변수입니다.
- 매개변수의 인디케이터 변수에 대한 N개의 매개변수
- SQLSTATE에 대한 CHAR(5) 출력 매개변수. 리턴되는 SQLSTATE는 프로시듀어의 성공 또는 실패를 표시합니다. 리턴된 SQLSTATE는 외부 프로그램에 의해 지정됩니다.

  사용자는 외부 프로그램에 모든 유효한 값으로 SQLSTATE를 설정하여 함수로부터 오류나 경고를 리턴할 수 있습니다.
- 완전 규정된 프로시듀어명에 대한 VARCHAR(517) 입력 매개변수.
- 특정 이름에 대한 VARCHAR(128) 입력 매개변수.
- 메세지 텍스트에 대한 VARCHAR(70) 출력 매개변수.

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

PARAMETER STYLE SQL은 LANGUAGE JAVA와 함께 사용할 수 없습니다.

DB2GENERAL

프로시듀어는 Java 메소드와 함께 사용하도록 정의된 매개변수 전달 규칙을 사용함을 지정합니다.

PARAMETER STYLE DB2GENERAL는 LANGUAGE JAVA와 함께 지정해야 합니다. JAVA에서 매개변수의 전달에 대한 자세한 정보는 'Developer Kit for Java' 책을 참조하십시오.

DB2SQL

CALL문의 매개변수 외에 몇몇 추가 매개변수가 프로시듀어에 전달됨을 지정합니다. DB2SQL은 다음 추가 매개변수가 마지막 매개변수로 전달될 수 있다는 점을 제외하고 SQL 매개변수 스타일과 동일합니다.

- DBINFO가 DECLARE PROCEDURE문에 지정된 경우 dbinfo 구조에 대한 매개변수

전달된 매개변수에 대한 자세한 정보는 적절한 소스 파일의 포함 파일 sqludf를 참조하십시오. 예를 들어 C에서는 sqludf가 QSYSINC/H에 있습니다.

PARAMETER STYLE DB2SQL은 LANGUAGE JAVA와 함께 사용할 수 없습니다.

GENERAL

CALL에 대해 지정된 프로시듀어가 매개변수를 수신할때 프로시듀어가 매개변수 전달 메카니즘을 사용함을 지정합니다. 추가 인수는 인디케이터 변수에 대해 전달되지 않습니다.

PARAMETER STYLE GENERAL은 LANGUAGE JAVA와 함께 사용할 수 없습니다.

GENERAL WITH NULLS

GENERAL에 지정된 CALL문의 매개변수 외에 몇몇 추가 매개변수가 프로시듀어에 전달됨을 지정합니다. 이러한 추가 인수에는 CALL문의 각 매개변수에 대한 요소와 함께 인디케이터 배열이 포함됩니다. C의 경우 짧은 int 배열입니다. 인디케이터 처리 방법에 대한 자세한 내용은 SQL 프로그래밍 개념 책을 참조하십시오.

PARAMETER STYLE GENERAL WITH NULLS는 LANGUAGE JAVA와 함께 사용할 수 없습니다.

JAVA

프로시듀어는 Java 언어와 SQLJ 루틴 스펙을 준수하는 매개변수 전달 규칙을 사용함을 지정합니다. INOUT 및 OUT 매개변수는 리턴하는 값을 이용하기 위해 단일 항목 배열로서 전달됩니다. 증가된 이식성을 위해 PARAMETER STYLE JAVA 규칙을 사용하는 Java 프로시듀어를 작성해야 합니다.

PARAMETER STYLE JAVA는 LANGUAGE JAVA와 함께 지정해야 합니다. JAVA에서 매개변수의 전달에 대한 자세한 정보는 'Developer Kit for Java' 책을 참조하십시오.

외부 함수 언어에 따라 매개변수가 전달되는 방식이 결정됨에 주의하십시오. 예를 들어 C에서는 모든 VARCHAR 또는 CHAR 매개변수가 NUL 종료 스트링으로 전달됩니다. 자세한 정보는 SQL 프로그래밍 개념 책을 참조하십시오.

주

*procedure-name*의 범위는 정의된 소스 프로그램 즉, 사전컴파일러에 입력되는 프로그램입니다. 따라서, 다른 분리되어 컴파일된 프로그램에서 호출된 프로그램이나 모듈은 호출하는 프로그램에 있는 DECLARE PROCEDURE문의 속성을 사용하지 않습니다.

DECLARE PROCEDURE문은 프로시듀어를 참조하는 모든 CALL문에 선행합니다.

DECLARE PROCEDURE에 사용할 수 있는 매개변수의 최대수는 255입니다. GENERAL WITH NULLS가 지정되면 최대값은 254입니다. 매개변수 양식 SQL이 지정되면 90개의 매개변수만 허용됩니다. 매개변수의 최대수는 외부 프로그램을 컴파일하기 위해 사용되는 사용권 프로그램이 허용하는 매개변수의 최대수로 제한됩니다.

DECLARE PROCEDURE문은 정적 CALL문에만 적용됩니다. 프로시듀어명이 호스트 변수에 의해 식별되는 CALL문이나 동적으로 준비된 CALL문에는 적용되지 않습니다.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- 키워드 VARIANT와 NOT VARIANT는 NOT DETERMINISTIC과 DETERMINISTIC의 동의어로 사용될 수 있습니다.
- 키워드 NULL CALL과 NOT NULL CALL은 CALLED ON NULL INPUT 및 RETURNS NULL ON NULL INPUT의 동의어로 사용될 수 있습니다.
- 키워드 SIMPLE CALL은 GENERAL의 동의어로 사용될 수 있습니다.
- DB2GENRL 값은 DB2GENERAL의 동의어로 사용될 수 있습니다.

예

C 프로그램에서 외부 프로시듀어 PROC1을 선언합니다. CALL문을 사용하여 프로시듀어를 호출하면 라이브러리 LIB1에 있는 PGM1이라는 COBOL 프로그램이 호출됩니다.

```
EXEC SQL
DECLARE PROC1 PROCEDURE
        (CHAR(10), CHAR(10))
        EXTERNAL NAME LIB1.PGM1
        LANGUAGE COBOL GENERAL;

EXEC SQL
CALL PROC1 ('FIRSTNAME ','LASTNAME  ');
```

DECLARE STATEMENT

DECLARE STATEMENT문은 프로그램 문서화를 위해 사용됩니다. 이 명령문은 준비된 SQL문을 식별하기 위해 사용되는 이름을 선언합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. 이 명령문을 Java 또는 REXX에서는 사용할 수 없습니다.

권한부여

필요한 사항이 없습니다.

구문

설명

statement-name

준비된 SQL문을 식별하기 위해 프로그램에서 사용되는 하나 이상의 이름을 나열합니다.

예

다음 예는 C 프로그램에서 DECLARE STATEMENT문을 사용하는 방법을 보여줍니다.

```
   EXEC SQL  INCLUDE SQLDA;
void main ()
  {
   EXEC SQL  BEGIN DECLARE SECTION;
   char src_stmt[32000];
   char sqlda[32000]
   EXEC SQL  END DECLARE SECTION;
   EXEC SQL INCLUDE SQLCA ;

   strcpy(src_stmt,"SELECT DEPTNO, DEPTNAME, MGRNO \
     FROM DEPARTMENT \
     WHERE ADMRDEPT = 'A00'");

   EXEC SQL DECLARE OBJ_STMT STATEMENT;

   (Allocate storage from SQLDA)

   EXEC SQL DECLARE C1 CURSOR FOR OBJ_STMT;

   EXEC SQL PREPARE OBJ_STMT FROM :src_stmt;
```

```
EXEC SQL DESCRIBE OBJ_STMT INTO :sqlda;

(Examine SQLDA)  (Set SQLDATA pointer addresses)

EXEC SQL  OPEN C1;

while (strncmp(SQLSTATE, "00000", 5) )
  {
  EXEC SQL FETCH C1 USING DESCRIPTOR :sqlda;

(Print results)

  }

EXEC SQL  CLOSE C1;
 return;
}
```

DECLARE VARIABLE

DECLARE VARIABLE문은 디폴트 이외의 부속 유형이나 코드화 문자 세트 ID(CCSID)를 호스트 변수에 지정하기 위해 사용됩니다.

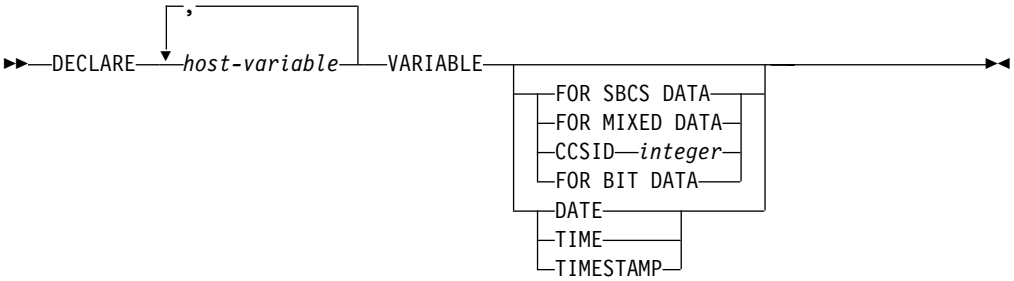
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. Java나 REXX에 지정되어서는 안됩니다.

권한부여

필요한 사항이 없습니다.

구문



설명

host-variable

프로그램에 정의된 문자나 그래픽 스트링 호스트 변수를 명명합니다. 인디케이터 변수는 host-variable에 대해 지정될 수 없습니다. host-variable 정의가 그 변수를 참조하는 DECLARE VARIABLE문에 선행하거나 뒤에 나옵니다.

FOR BIT DATA

host-variable의 값이 코드화 문자 세트와 연관이 없고 따라서 결코 변환되지 않음을 지정합니다. FOR BIT DATA 호스트 변수의 코드화 문자 세트 ID(CCSID)는 65535입니다. FOR BIT DATA는 그래픽 host-variable에 대해 지정될 수 없습니다.

FOR SBCS DATA

호스트 변수의 값에 SBCS(1바이트 문자 세트) 자료가 들어 있음을 지정합니다. 어플리케이션 리퀘스터에 있는 작업의 코드화 문자 세트 ID(CCSID) 속성이 DBCS가 될 수 없거나 호스트 변수의 길이가 4보다 작으면, FOR SBCS DATA가 디폴트입니다. FOR SBCS DATA의 코드화 문자 세트 ID(CCSID)는 어플리케이션 리퀘스터에 있는 작업의 CCSID 속성에 의해 판별됩니다. FOR SBCS DATA는 그래픽 host-variable에 대해 지정될 수 없습니다.

FOR MIXED DATA

호스트 변수의 값에 SBCS 자료와 DBCS 자료가 둘 다 들어 있음을 지정합니다. 어플리케이션 사용자에 있는 작업의 코드화 문자 세트 ID(CCSID) 속성이 DBCS가 될 수 있고 호스트 변수의 길이가 3보다 크면, FOR MIXED DATA가 디폴트입니다. FOR DBCS DATA의 코드화 문자 세트 ID(CCSID)는 어플리케이션 리퀘스터에 있는 작업의 CCSID 속성에 의해 판별됩니다. FOR MIXED DATA는 그래픽 host-variable에 대해 지정될 수 없습니다.

CCSID *integer*

호스트 변수의 값에 코드화 문자 세트 ID(CCSID) 정수 자료가 포함되어 있음을 지정합니다. 정수가 SBCS 코드화 문자 세트 ID(CCSID)이면 호스트 변수는 SBCS 자료입니다. 정수가 혼합된 자료 코드화 문자 세트 ID(CCSID)이면 호스트 변수는 혼합된 자료입니다. 문자 호스트 변수의 경우 지정된 코드화 문자 세트 ID(CCSID)는 SBCS 또는 혼합된 CCSID여야 합니다.

변수가 그래픽 스트링 자료 유형을 가질때 지정된 코드화 문자 세트 ID(CCSID)는 DBCS 또는 UCS-2 CCSID여야 합니다. 유효한 CCSID 리스트는 899 페이지의 부록 E『코드화 문자 세트 ID(CCSID) 값』을 참조하십시오. CCSID 13488을 지정하여 UCS-2 자료를 나타내십시오. CCSID가 지정되지 않았다면 그래픽 스트링 변수의 CCSID 작업에 대한 DBCS CCSID와 연관됩니다.

DATE

호스트 변수의 값에 포함된 자료가 날짜임을 지정합니다.

TIME

호스트 변수의 값에 포함된 자료가 시간임을 지정합니다.

TIMESTAMP

호스트 변수의 값에 포함된 자료가 시간소인임을 지정합니다.

주

다음 경우를 제외하고는 SQL문이 유효한 어플리케이션 프로그램에 DECLARE VARIABLE문이 지정될 수 있습니다.

- 호스트 언어가 COBOL이나 RPG이면 DECLARE VARIABLE문에 지정된 호스트 변수를 참조하는 SQL문 앞에 DECLARE VARIABLE문이 있어야 합니다.
- DATE, TIME 또는 TIMESTAMP가 C에서 널로 종료된 문자 스트링에 대해 지정되면 C 선언의 길이가 하나 감소합니다.

다음 경우 사전컴파일하는 중에 오류 메세지가 발생합니다.

- 참조가 존재하지 않는 변수에 대해 작성된 경우
- 참조가 숫자 변수에 대해 작성된 경우
- 참조가 이미 참조된 변수에 대해 작성된 경우

- 참조가 고유하지 않은 변수에 대해 작성된 경우
- SQL문과 DECLARE VARIABLE문이 같은 변수를 참조하는 SQL문 뒤에 DECLARE VARIABLE문이 있는 경우
- FOR BIT DATA, FOR SBCS DATA 또는 FOR MIXED DATA절이 그래픽 호스트 변수에 대해 지정된 경우
- SBCS 또는 혼합된 코드화 문자 세트 ID(CCSID)가 그래픽 호스트 변수에 대해 지정된 경우
- DBCS 또는 UCS-2 코드화 문자 세트 ID(CCSID)가 문자 호스트 변수에 대해 지정된 경우
- DATE, TIME 또는 TIMESTAMP가 문자가 아닌 호스트 변수에 대해 지정된 경우
- DATE, TIME 또는 TIMESTAMP에 대해 사용된 호스트 변수의 길이가 최소의 날짜, 시간 또는 시간소인 값에 충분한 길이가 아닐 경우

예

예에서는 C 프로그램 변수 *fred* 및 *pete*를 혼합된 자료로 선언하고 *jean* 및 *dave*를 CCSID 37의 SBCS 자료로 선언합니다.

```
void main ()
  {
   EXEC SQL  BEGIN DECLARE SECTION;
     char fred[10];
     EXEC SQL DECLARE :fred VARIABLE  FOR MIXED DATA;

     decimal(6,0) mary;
     char pete[4];
     EXEC SQL DECLARE :pete VARIABLE FOR MIXED DATA;

     char jean[30];
     char dave[9];
     EXEC SQL DECLARE :jean, :dave VARIABLE  CCSID 37;
   EXEC SQL  END DECLARE SECTION;
   EXEC SQL  INCLUDE SQLCA;
   ...

  }
```

DELETE

DELETE문은 표나 뷰에서 행을 삭제합니다. 뷰에서 행을 삭제하는 것은 뷰가 기초로 하는 표에서 행을 삭제하는 것입니다.

이 명령문에는 두 가지 형식이 있습니다.

- *탐색된* DELETE 형식은 하나 이상의 행을 삭제하는 데 사용됩니다(선택적으로 탐색 조건에 의해 판별됨).
- *위치지정된* DELETE 형식은 정확히 한 행을 삭제하는 데 사용됩니다(커서의 현재 위치에 의해 판별됨).

호출

탐색된 DELETE문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 위치지정된 DELETE는 어플리케이션 프로그램에 삽입되어야 합니다. 탐색된 DELETE 및 위치지정된 DELETE는 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표나 뷰의 경우
  - 표나 뷰에 대한 DELETE 권한
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 DELETE 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 DELETE 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *DLT의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 DELETE 권한을 갖습니다.[54]

- 뷰에 대해 DELETE 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *DLT 시스템 권한을, 이 뷰가 직접적 또는 간접적으로 종속되어 있는 첫 번째 표나 뷰에 대해 *DLT 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 첫 번째 표나 뷰 그리고 뷰가 참조된 경우 첫 번째 표나 뷰 정의에 참조된 첫 번째 표나 뷰 등

탐색된 DELETE의 *search-condition*에 표나 뷰의 열에 대한 참조가 들어 있으면, 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

54. 뷰가 작성될 때 소유자는 뷰에 대한 DELETE 권한을 예약할 필요는 없습니다. 뷰가 삭제를 허용하고 소유자가 subselect에 있는 참조된 첫 번째 표에 대한 DELETE 권한도 가지면, 소유자는 DELETE 권한만을 예약합니다.

- 표나 뷰에 대한 SELECT 권한
- 관리 권한

*search-condition*에 부속 조회가 들어 있으면, 명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 부속 조회에 식별된 각 표나 뷰의 경우
  - 표 또는 뷰에서의 SELECT 권한 및
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

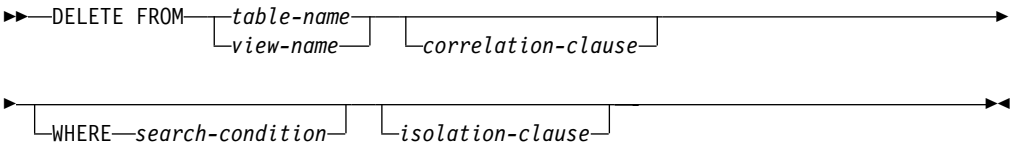
- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

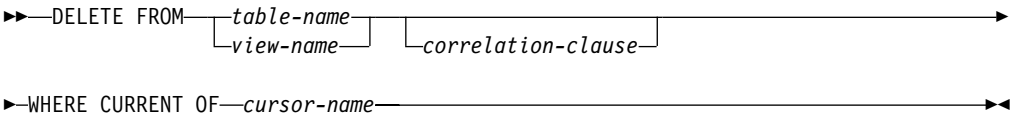
- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을, 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

구문

탐색된 DELETE



위치지정된 DELETE



isolation절:

설명

FROM *table-name* 또는 *view-name*

삭제하려는 행이 있는 표나 뷰를 식별합니다. 이름은 서버에 있는 표나 뷰를 식별해야 하지만, 카탈로그 표, 카탈로그 표의 뷰 또는 읽기 전용 뷰를 식별해서는 안 됩니다. 읽기 전용 뷰에 대한 설명은 594 페이지의 『주』를 참조하십시오.

correlation-clause

표나 뷰 및 표나 뷰의 열 이름을 지정하기 위해 *search-condition* 내에서 사용될 수 있습니다. *correlation-clause*에 대한 설명은 335 페이지의 제 4 장 『조회』를 참조하십시오. *correlation-name*에 대한 설명은 108 페이지의 『상관명』을 참조하십시오.

WHERE

삭제될 행을 지정합니다. 구를 생략하고 탐색 조건을 부여하거나 커서를 명명할 수 있습니다. 구를 생략하면 표나 뷰의 모든 행이 삭제됩니다.

search-condition

154 페이지의 『탐색 조건』에 설명된 탐색 조건입니다. 부속 조회를 제외한 *탐색 조건*의 각 *column-name*은 표나 뷰의 열을 식별해야 합니다.

*search-condition*은 표나 뷰의 각 행에 적용되며, 삭제된 행은 *search-condition*의 결과가 참인 행입니다.

*탐색 조건*에 부속 조회가 있으면, 부속 조회는 *search-condition*이 행과 *search-condition*을 적용할 때 사용된 부속 조회의 결과에 적용될 때마다 실행된다고 간주될 수 있습니다. 실제로, 상호 관련 참조가 없는 부속 조회는 한 번 실행 되는 반면, 상호 관련 조회가 있는 부속 조회는 각 행에 한 번씩 실행되어야 합니다.

부속 조회가 DELETE문의 오브젝트 표나 CASCADE, SET NULL 또는 SET DEFAULT 삭제 규칙이 있는 종속 표를 참조하면 부속 조회는 행이 삭제되기 전에 완전히 평가됩니다.

CURRENT OF *cursor-name*

삭제 조작에 사용될 커서를 식별합니다. *cursor-name*은 DECLARE CURSOR 문의 주에 설명된 대로 선언된 커서를 식별해야 합니다.

명명된 표나 뷰는 커서의 SELECT문의 FROM절에서 식별되어야 하며, 커서의 결과표는 읽기 전용이 되어서는 안됩니다. 읽기 전용 결과표에 대한 설명은 598 페이지의 『DECLARE CURSOR』를 참조하십시오.

DELETE문이 실행되면 커서는 행에 위치되어야 하고 그 행은 삭제됩니다. 삭제 후에 커서는 결과표의 다음 행 앞에 위치합니다. 다음 행이 없으면 커서는 마지막 행 다음에 위치합니다.

isolation-clause

이 명령문에 대해 사용될 분리 레벨을 지정합니다. *isolation-clause*가 지정되지 않은 경우 디폴트 분리가 사용됩니다. *isolation-clause*에 대한 설명은 isolation절을 참조하십시오.

DELETE 규칙

트리거

식별된 표나 식별된 뷰의 기본 표에 삭제 트리거가 있으면 트리거는 활성화됩니다. 트리거는 다른 명령문이 실행되도록 하거나 삭제된 값에 따라 오류 조건을 야기합니다.

참조 무결성

식별된 표나 식별된 표의 기본 표가 상위 표이면 선택 행에는 RESTRICT 또는 NO ACTION의 삭제 규칙과 관련하여 하위 행이 있을 수 없으며 DELETE가 RESTRICT 또는 NO ACTION의 삭제 규칙과 관련하여 종속되는 것이 있는 하위 행과 연속될 수 없습니다.

RESTRICT나 NO ACTION 삭제 규칙에 의해 삭제 조작을 막을 수 없으면, 선택된 행이 삭제됩니다. 선택된 행에 종속된 행에도 영향을 미칩니다.

- SET NULL 삭제 규칙과 관련하여 하위 행인 외부 키를 가진 널 가능 열이 널 값으로 설정됩니다.
- SET DEFAULT 삭제 규칙과 관련하여 하위 행인 외부 키를 가진 열이 디폴트 값으로 설정됩니다.
- CASCADE의 삭제 규칙과 관련하여 하위 행들이 삭제되며 위에 나오는 규칙을 이와 같은 행에 차례로 적용합니다.

참조 제한(RESTRICT 삭제 규칙을 갖는 참조 제한 제외)은 명령문 끝에서 검사됩니다. 여러 행이 삭제되는 경우에는 모든 행이 삭제되고 연관된 트리거가 활성화된 후에 이러한 검사가 이루어집니다.

검사 제한조건

검사 제한조건은 SET NULL 또는 SET DEFAULT의 삭제 규칙과의 관계에 종속 항목이 있는 경우, 상위 표에서 행이 삭제되지 않도록 방지할 수 있습니다. 상위 표의 행을 삭제함으로써 종속 테이블의 열이 널 또는 디폴트 값으로 설정될 수 있으며 널 또는 디폴트 값이 검사 제한조건의 검색 조건이 거짓이 되게 할 수 있다면, 행은 삭제되지 않습니다.

주

삭제 조작을 실행하는 중 오류가 발생할 경우, 이 명령문의 조치 변경사항, 참조 제한 조건, 트리거된 모든 SQL문이 롤백됩니다(분리 레벨이 이 명령문 또는 다른 트리거된 SQL문에 대해 NC가 아닌 경우).

적합한 잠금이 이미 존재하지 않으면 하나 또는 하나 이상의 배타적 잠금을 성공적인 DELETE문의 실행 중에 예약합니다. 확약 또는 롤백 조작으로 잠금이 해제될 때까지는 다음을 통해서만 DELETE 연산을 할 수 있습니다.

- 삭제를 수행한 어플리케이션 프로세스
- 분리 레벨 UR 또는 NC를 사용하는 다른 어플리케이션 프로세스

잠금으로 인해 다른 어플리케이션 프로세스가 표에 대해 조작을 수행할 수 없습니다. 잠금에 대한 자세한 내용은 COMMIT, ROLLBACK 및 LOCK TABLE문에 대한 설명과 24 페이지의 『분리 레벨』을 참조하십시오.

어플리케이션 프로세스가 갱신할 수 없는 커서가 있는 행을 삭제하면 그 커서는 결과표의 다음 행 앞에 위치합니다. C는 다음 행 R 앞에 위치한 커서입니다(OPEN, C를 통한 DELETE, 다른 커서를 통한 DELETE 또는 탐색된 DELETE의 결과로). R이 파생된 기본 표에 영향을 미치는 INSERT, UPDATE 및 DELETE 조작이 있으면, C를 참조하는 다음 FETCH 조작은 R의 C에 위치할 필요는 없습니다. 예를 들면, 조작은 R'의 C에 위치할 수 있는데, R'는 이제 결과표의 다음 행인 새로운 행입니다.

DELETE문이 완료되면 삭제된 행 수는 SQLCA의 SQLERRD(3)에 리턴됩니다. SQLERRD(3)의 값은 CASCADE 삭제 규칙이나 트리거의 결과로 삭제된 행 수를 포함하지 않습니다.

SQLCA의 SQLERRD(5)는 참조 제한에 의해 영향을 받는 행 수를 보여줍니다. CASCADE 삭제 규칙의 결과로 삭제된 행과 SET NULL 또는 SET DEFAULT 삭제 규칙의 결과로 외부 키가 NULL이나 디폴트 값으로 설정된 행이 포함됩니다.

SQLCA의 설명은 871 페이지의 부록 B 『SQL 통신 영역』을 참조하십시오.

COMMIT(*RR), COMMIT(*ALL), COMMIT(*CS) 또는 COMMIT(*CHG)가 지정되었을 때 최대 4000000 행이 단일 DELETE문에서 삭제되거나 변경될 수 있습니다. 변경된 행 수에는 트리거, CASCADE, SET NULL 또는 SET DEFAULT 참조 무결성 삭제 규칙의 결과로 동일한 확약 정의하에서 삽입, 갱신 또는 삭제된 행이 포함됩니다.

호스트 변수는 REXX 프로시듀어의 DELETE문에 사용될 수 없습니다. 그 대신, 매개변수 마커를 사용하여 DELETE는 PREPARE와 EXECUTE의 오브젝트가 되어야 합니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- 키워드 NONE은 NC에 대한 동의어로 사용될 수 있습니다.
- 키워드 CHG은 UR에 대한 동의어로 사용될 수 있습니다.
- 키워드 ALL은 RS에 대한 동의어로 사용될 수 있습니다.

예

예 1

DEPARTMENT 표에서 부서(DEPTNO) 'D11'을 삭제합니다.

```
DELETE FROM DEPARTMENT
  WHERE DEPTNO = 'D11'
```

예 2

DEPARTMENT 표에서 모든 부서를 삭제합니다(즉, 표를 비웁니다).

```
DELETE FROM DEPARTMENT
```

예 3

Java 프로그램 명령문을 사용하여 호스트 변수 HOSTDEPT (java.lang.String)의 값과 같은 부서(DEPTNO)를 갖는 모든 부속 프로젝트(MAJPROJ는 NULL)를 연결 문맥 'ctx'인 PROJECT 표에서 삭제합니다.

```
#sql [ctx] { DELETE FROM PROJECT
               WHERE DEPTNO = :HOSTDEPT AND MAJPROJ IS NULL };
```

예 4

퇴직한 사원(JOB)을 표시하기 위해 사용된 Java 프로그램의 일부를 코드하고 연결 문맥 'ctx'인 EMPLOYEE 표에서 특정 사원을 제거합니다.

```
#sql iterator empIterator implements sqlj.runtime.ForUpdate
     ( ... );
empIterator C1;

#sql [ctx] C1 = { SELECT * FROM EMPLOYEE
                    WHERE JOB = 'RETIRED' };

#sql {  FETCH C1 INTO ...     };
while ( !C1.endFetch() )  {
   System.out.println( ... );
                ...
   if ( condition for deleting row ) {
       #sql [ctx] { DELETE FROM EMPLOYEE
                      WHERE CURRENT OF C1  };
   }
   #sql { FETCH C1 INTO ...     };
}
C1.close();
```

DESCRIBE

DESCRIBE문은 준비된 명령문에 대한 정보를 구합니다. 준비된 명령문에 대한 설명은 728 페이지의 『PREPARE』를 참조하십시오.

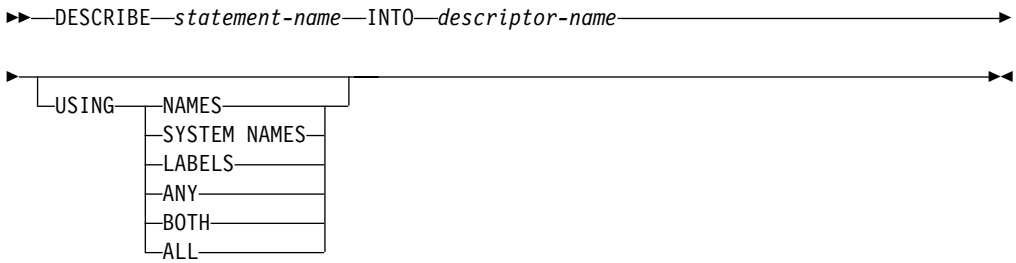
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

필요한 사항이 없습니다. 준비된 명령문을 작성하는 데 필요한 권한부여에 대한 내용은 728 페이지의 『PREPARE』를 참조하십시오.

구문



설명

statement-name

설명하려는 명령문을 식별합니다. DESCRIBE문이 실행되면 이름은 서버에서 준비된 명령문을 식별해야 합니다.

INTO *descriptor-name*

881 페이지의 부록 C 『SQLDA(SQL 설명자 영역)』에 설명되어 있는 SQLDA(SQL 설명자 영역)를 식별합니다. DESCRIBE문이 실행되기 전에 SQLDA의 다음 변수가 설정되어야 합니다(REXX에 대한 규칙은 다릅니다. 자세한 내용은 SQL Programming with Host Languages 책을 참조하십시오).

SQLN SQLDA에 제공된 SQLVAR의 발생 수를 지정합니다. DESCRIBE문이 실행되기 전에 SQLN은 0 보다 크거나 같은 값으로 설정되어야 합니다. 발생 요구의 수를 판별하는 기술에 대한 내용은 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.

DESCRIBE문이 실행될 때 다음과 같이 데이터베이스 관리자는 값을 SQLDA의 변수에 지정합니다.

변수	데이터베이스 관리자에 의해 리턴된 정보

SQLDAID 처음 6바이트가 'SQLDA'(5자 다음에 공백 문자가 나옴)에 설정됩니다.

7번째 바이트 SQLDOUBLED는 SQLDA에 모든 선택 리스트 항목에 대한 둘, 셋, 네개의 SQLVAR가 들어 있는 경우 '2', '3' 또는 '4'로 설정됩니다. 이 기법은 LOB, 고유한 유형, 레이블 및 시스템명을 수용하기 위해 사용됩니다. 그렇지 않으면 SQLDOUBLED가 공백 문자로 설정됩니다.

SQLDA에 DESCRIBE 응답 전체를 가지고 있을 충분한 공간이 없는 경우 2배수 플래그는 공백으로 설정됩니다.

8번째 바이트는 공백 문자로 설정됩니다.

SQLDABC SQLDA의 길이

SQLD 준비된 명령문이 SELECT면, 열 수는 결과표에 있으며, 그렇지 않은 경우에는 0입니다.

SQLVAR SQLD의 값이 0이거나 SQLN의 값보다 크면, SQLVAR 발생에 지정된 값이 없습니다.

SQLD 값이 *n*이고 *n*이 0 보다 크지만 SQLN의 값보다 작거나 같으면, 값은 SQLVAR의 첫 번째 *n* 발생에 지정되어 SQLVAR의 첫 번째 발생에는 결과표의 첫 번째 열의 설명이 포함되고, SQLVAR의 두 번째 발생에는 결과표의 두 번째 열의 설명이 포함되며, 이런 식으로 설명이 포함됩니다. SQLVAR 발생에 지정된 값에 대한 내용은 882 페이지의 『SQLVAR 발생에서의 필드 설명』을 참조하십시오.

USING

SQLDA의 각 SQLNAME 변수에 지정된 값을 지정합니다. 요구된 값이 없으면 SQLNAME은 길이 0으로 설정됩니다.

NAMES

열의 이름을 지정합니다. 이것이 디폴트 값입니다. 이름이 select-list에 명시적으로 나열된 준비된 명령문의 DESCRIBE인 경우 지정된 이름이 리턴됩니다.

SYSTEM NAMES

열의 시스템 열 이름을 지정합니다.

LABELS

열의 레이블을 지정합니다(열 레이블은 LABEL문으로 정의됩니다). 레이블의 처음 20바이트만 리턴됩니다.

ANY

열 레이블을 지정합니다. 열에 레이블이 없으면 열 이름이 대신 사용됩니다.

BOTH

열의 레이블과 이름을 둘다 지정합니다. 이 경우 결과 세트가 고유한 유형에 포

함되는지 여부에 따라, 열 당 SQLVAR의 두 번 또는 세 번의 발생이 추가 정보를 제공하기 위해 필요합니다. SQLVAR 배열의 확장을 지정하려면 SQLN을 2*n 또는 3*n으로 설정합니다(여기서, n은 표나 뷰의 열 수입니다). SQLVAR의 처음 n번 발생에는 열 이름이 포함됩니다. 두 번째 또는 세 번째 n번 발생에는 열 레이블이 포함됩니다. 고유한 유형이 없는 경우 레이블은 SQLVAR 항목의 두 번째 세트에 리턴됩니다. 고유한 유형이 있는 경우 레이블은 SQLVAR 항목의 세 번째 세트에 리턴됩니다.

ALL

레이블, 열 이름 및 시스템 열 이름을 지정합니다. 이 경우 결과 세트가 고유한 유형에 포함되는지 여부에 따라, 열 당 SQLVAR의 세 번 또는 네 번의 발생은 추가 정보를 제공하기 위해 필요합니다. SQLVAR 배열의 확장을 지정하기 위해 SQLN을 3*n 또는 4*n으로 설정합니다(여기서, n은 결과표의 열 수입니다). SQLVAR의 처음 n번 발생에는 시스템 열 이름이 포함됩니다. 두 번째 또는 세 번째 n번 발생에는 열 레이블이 포함됩니다. 세 번째 또는 네 번째 n번 발생에는 열 이름이 포함됩니다. 고유한 유형이 없는 경우 레이블은 SQLVAR 항목의 두 번째 세트에 리턴되고 열 이름은 SQLVAR의 세 번째 세트에 리턴됩니다. 고유한 유형이 있는 경우 레이블은 SQLVAR 항목의 세 번째 세트에 리턴되고 열 이름은 SQLVAR의 네 번째 세트에 리턴됩니다.

주

준비된 명령문에 대한 정보는 PREPARE문의 INTO절을 사용하여 구할 수 있습니다.

SQLDA 할당

DESCRIBE나 PREPARE INTO문이 실행되기 전에 SQLN 값은 SQLDA에 제공되는 SQLVAR의 발생 수를 표시하기 위해 0 보다 크거나 같은 값으로 설정되어야 하며, SQLN 발생을 포함하기 위해 충분한 기억장치가 할당되어야 합니다(REXX에서 SQLDA에 대해 기억장치가 할당될 필요가 없습니다). 준비된 SELECT문의 결과표의 열 설명을 구하려면 SQLVAR의 발생 수가 열의 수보다 작아서는 안됩니다. 뿐만 아니라, USING BOTH나 USING ALL이 지정되거나 열이 LOB나 고유한 유형을 포함하면 SQLVAR의 발생 수는 열 수의 두 배, 세 배 또는 네 배여야 합니다. 자세한 내용은 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.

발생의 모든 세트를 리턴하기 위해 충분한 발생이 제공되지 않으면 SQLN은 모든 정보를 리턴하기 위해 필요한 총 발생 수로 설정됩니다. 그렇지 않으면 SQLN은 열의 수로 설정됩니다.

SQLDA를 할당하는 가능한 방법 중 세 가지가 다음에 설명됩니다.

첫 번째 기술: 충분한 SQLVAR의 발생으로 SQLDA를 할당하여 어플리케이션이 처리해야 하는 모든 선택 리스트를 수용합니다. 최대로, SQLVAR의 수는 결과표에서 허용하는 열의 최대수의 네 배가 될 수 있습니다. 할당한 후, 어플리케이션은 이 SQLDA를 반복적으로 사용할 수 있습니다.

대부분의 기억장치가 특별한 선택 리스트에 대해 사용되지 않는 경우에도, 이 기술은 결코 해제되지 않는 많은 양의 기억장치를 사용합니다.

두 번째 기술: 다음 세 단계를 선택 리스트를 처리할 때마다 반복합니다.

1. SQLVAR의 발생이 없는 SQLDA(즉, SQLN이 0인 SQLDA)로 DESCRIBE문을 실행합니다. SQLD에 대하여 리턴된 값은 결과 표의 열 번호입니다. 이것은 SQLVAR의 필요한 발생 수이거나 그것의 1/2, 1/3 또는 1/4입니다. SQLVAR 항목이 없으므로 경고가 발생합니다. 경고가 발생한 SQLSTATE가 01005인 경우 SQLVAR 항목 수는 SQLD에 리턴된 값의 두 배, 세 배 또는 네 배이어야 합니다. 자세한 내용은 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.
2. SQLD의 리턴된 값을 사용하여 SQLVAR의 충분한 발생과 함께 SQLDA를 할당합니다.
3. 새로운 SQLDA를 사용하여 DESCRIBE문을 다시 실행합니다.

이 기술은 첫 번째 기술보다 기억장치 관리를 더 잘 할 수 있지만 DESCRIBE문의 수가 두 배가 됩니다.

세 번째 기술: 선택 리스트를 거의 전부 처리하기에 충분하지만 적당히 작은 SQLDA를 할당합니다. SQLDA가 너무 작아서 DESCRIBE의 실행이 실패하면 더 큰 SQLDA를 할당하여 DESCRIBE를 다시 실행합니다. 새로운 SQLDA의 경우 SQLVAR의 발생 수에 대해 DESCRIBE의 첫 번째 실행에서 리턴된 SQLD의 값을 사용합니다.

이 기술은 앞의 두 가지 기술을 절충합니다. 효율성은 원래의 SQLDA에 대한 크기 선택에 따라 좌우됩니다.

예

C 프로그램에서 SQLVAR 발생이 없는 SQLDA로 DESCRIBE문을 실행합니다. SQLD가 0보다 크면, 그 값을 사용하여 SQLVAR의 필요한 발생 수가 있는 SQLDA를 할당하고, 그 SQLDA를 사용하여 DESCRIBE문을 실행합니다.

```
EXEC SQL  BEGIN DECLARE SECTION;
 char stmt1_str [200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

... /* code to prompt user for a query, then to generate    */
    /* a select-statement in the stmt1_str                  */
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
```

```
... /* code to set SQLN to zero and to allocate the SQLDA   */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

... /* code to prepare for the use of the SQLDA            */
EXEC SQL  OPEN DYN_CURSOR;

... /* loop to fetch rows from result table                */
EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
.
.
.
```

DESCRIBE TABLE

DESCRIBE TABLE문은 표나 뷰에 대한 정보를 획득합니다.

호출

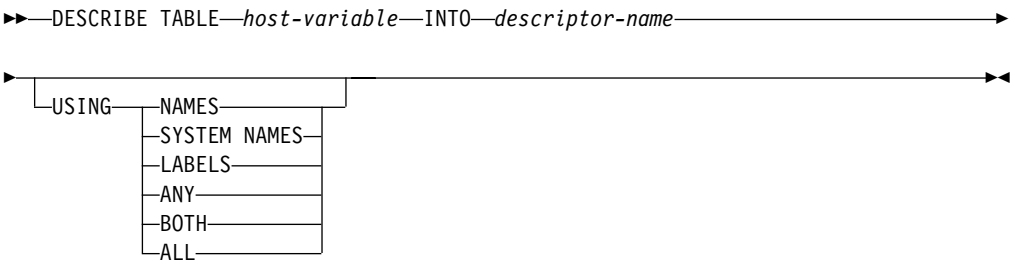
이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표나 뷰의 경우
  - 표나 뷰에 대한 *OBJOPR 시스템 권한
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

구문



설명

host-variable

정보를 구하려는 표나 뷰를 식별합니다. DESCRIBE TABLE문이 실행될 때

- 이름은 서버에 있는 표나 뷰를 식별해야 합니다.
- 호스트 변수는 문자 스트링 또는 UCS-2 그래픽 스트링 변수여야 하며 인디케이터 변수를 포함해서는 안됩니다.
- host-variable에 들어 있는 표 이름은 왼쪽부터 배열되어야 하며, 길이가 호스트 변수보다 짧으면 오른쪽을 공백으로 채워야 합니다.
- 표 이름이 구분된 이름이 아니면 대문자여야 합니다.

DESCRIBE TABLE문이 실행될 때 다음과 같이 데이터베이스 관리자는 값을 SQLDA의 변수에 지정합니다.

INTO *descriptor-name*

881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』에 설명되어 있는 SQLDA(SQL

설명자 영역)를 식별합니다. DESCRIBE TABLE문이 실행되기 전에 SQLDA에 있는 다음 변수가 설정되어야 합니다(REXX에 대한 규칙은 다릅니다. 자세한 내용은 SQL Programming with Host Languages 책을 참조하십시오).

SQLN SQLDA에 제공된 SQLVAR의 발생 수를 지정합니다. DESCRIBE TABLE 문이 실행되기 전에 SQLN은 0 보다 크거나 같은 값으로 설정되어야 합니다. 발생 요구의 수를 판별하는 기술에 대한 내용은 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.

변수	데이터베이스 관리자에 의해 리턴된 정보
SQLDAID	처음 6바이트가 'SQLDA'(5자 다음에 공백 문자가 나옴)에 설정됩니다. 7번째 바이트 SQLDOUBLED는 SQLDA에 모든 선택 리스트 항목에 대한 둘, 셋, 네개의 SQLVAR가 들어 있는 경우 '2', '3' 또는 '4'로 설정됩니다. 이 기법은 LOB, 고유한 유형, 레이블 및 시스템명을 수용하기 위해 사용됩니다. 그렇지 않으면 SQLDOUBLED가 공백 문자로 설정됩니다. SQLDA에 DESCRIBE 응답 전체를 가지고 있을 충분한 공간이 없는 경우 2배수 플래그는 공백으로 설정됩니다.
SQLDABC	SQLDA의 길이
SQLD	참조된 표나 뷰에 있는 열의 수
SQLVAR	SQLD의 값이 0이거나 SQLN의 값보다 크면, SQLVAR 발생에 지정된 값이 없습니다. SQLD 값이 *n*이고 *n*이 0 보다 크지만 SQLN의 값보다 작거나 같으면, 값은 SQLVAR의 처음 *n*번 발생에 지정되어 SQLVAR의 첫 번째 발생에는 표나 뷰의 첫 번째 열의 설명이 포함되고, SQLVAR의 두 번째 발생에는 표나 뷰의 두 번째 열의 설명이 포함되고, 이런 식으로 계속 설명이 포함됩니다. SQLVAR 발생에 지정된 값에 대한 내용은 882 페이지의 『SQLVAR 발생에서의 필드 설명』을 참조하십시오.

USING

SQLDA의 각 SQLNAME 변수에 지정된 값을 지정합니다. 요구된 값이 없으면 SQLNAME은 길이 0으로 설정됩니다.

NAMES

열의 이름을 지정합니다. 이것이 디폴트 값입니다.

SYSTEM NAMES
열의 시스템 열 이름을 지정합니다.

LABELS
열의 레이블을 지정합니다(열 레이블은 LABEL문으로 정의됩니다). 레이블의 처음 20바이트만 리턴됩니다.

ANY
열 레이블을 지정합니다. 열에 레이블이 없으면 열 이름이 대신 사용됩니다.

BOTH
열의 레이블과 이름을 둘다 지정합니다. 이 경우 결과 세트가 고유한 유형에 포함되는지 여부에 따라, 열 당 SQLVAR의 두 번 또는 세 번의 발생이 추가 정보를 제공하기 위해 필요합니다. SQLVAR 배열의 확장을 지정하려면 SQLN을 2**n* 또는 3**n*으로 설정합니다(여기서, *n*은 표나 뷰의 열 수입니다). SQLVAR의 처음 *n*번 발생에는 열 이름이 포함됩니다. 두 번째 또는 세 번째 *n*번 발생에는 열 레이블이 포함됩니다. 고유한 유형이 없는 경우 레이블은 SQLVAR 항목의 두 번째 세트에 리턴됩니다. 고유한 유형이 있는 경우 레이블은 SQLVAR 항목의 세 번째 세트에 리턴됩니다.

ALL
레이블, 열 이름 및 시스템 열 이름을 지정합니다. 이 경우 결과 세트가 고유한 유형에 포함되는지 여부에 따라, 열 당 SQLVAR의 세 번 또는 네 번의 발생은 추가 정보를 제공하기 위해 필요합니다. SQLVAR 배열의 확장을 지정하기 위해 SQLN을 3**n* 또는 4**n*으로 설정합니다(여기서, *n*은 결과표의 열 수입니다). SQLVAR의 처음 *n*번 발생에는 시스템 열 이름이 포함됩니다. 두 번째 또는 세 번째 *n*번 발생에는 열 레이블이 포함됩니다. 세 번째 또는 네 번째 *n*번 발생에는 열 이름이 포함됩니다. 고유한 유형이 없는 경우 레이블은 SQLVAR 항목의 두 번째 세트에 리턴되고 열 이름은 SQLVAR의 세 번째 세트에 리턴됩니다. 고유한 유형이 있는 경우 레이블은 SQLVAR 항목의 세 번째 세트에 리턴되고 열 이름은 SQLVAR의 네 번째 세트에 리턴됩니다.

주

DESCRIBE TABLE문이 실행되기 전에 SQLDA에 제공된 SQLVAR의 발생 수를 표시하기 위해 SQLN 값은 0보다 크거나 같은 값으로 설정되어야 하며, SQLN 발생을 포함하기 위해 충분한 기억장치가 할당되어야 합니다. 표나 뷰의 열 설명을 구하려면 SQLVAR의 발생 수는 열의 수보다 작아서는 안됩니다. 뿐만 아니라, USING BOTH나 USING ALL이 지정되거나 열이 LOB나 고유한 유형을 포함하면 SQLVAR의 발생 수는 열 수의 두 배, 세 배 또는 네 배여야 합니다. 자세한 내용은 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.

발생의 모든 세트를 리턴하기 위해 충분한 발생이 제공되지 않으면 SQLN은 모든 정보를 리턴하기 위해 필요한 총 발생 수로 설정됩니다. 그렇지 않으면 SQLN은 열의 수로 설정됩니다.

SQLDA를 할당하기 위해 사용할 수 있는 기술에 대한 설명은 647 페이지의 『SQLDA 할당』을 참조하십시오.

예

C 프로그램에서 SQLVAR 발생이 없는 SQLDA로 DESCRIBE문을 실행합니다. SQLD가 0보다 크면, 그 값을 사용하여 SQLVAR의 필요한 발생 수가 있는 SQLDA를 할당하고, 그 SQLDA를 사용하여 DESCRIBE문을 실행합니다.

```
 EXEC SQL  BEGIN DECLARE SECTION;
  char table_name[201];
 EXEC SQL  END DECLARE SECTION;
 EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

.../*사용자에게 표나 뷰에 대한 프롬트를 표시하는 코드 */
.../*SQLN을 0으로 설정하고 SQLDA를 할당하는 코드 */
EXEC SQL  DESCRIBE TABLE :table_name INTO :sqlda;

... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
EXEC SQL  DESCRIBE TABLE :table_name INTO :sqlda;

.
.
.
```

DISCONNECT

DISCONNECT문은 보호되지 않은 대화에 대한 하나 이상의 연결을 종료합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. Java나 REXX에 지정되어서는 안됩니다.

DISCONNECT는 트리거에서 사용할 수 없습니다. 외부 프로시듀어가 리모트 서버에서 호출되면 DISCONNECT를 외부 프로시듀어에서 사용할 수 없습니다.

권한부여

필요한 사항이 없습니다.

구문

설명

server-name 또는 *host-variable*

지정한 서버명 또는 호스트 변수에 있는 서버명으로 서버를 식별합니다. 호스트 변수가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 서버명은 호스트 변수 내에서 왼쪽부터 배열되어야 하고 일반 ID를 형성하는 규칙을 따라야 합니다.
- 서버명의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

DISCONNECT문을 실행할 때 지정한 서버명 또는 호스트 변수에 있는 서버명이 활성 그룹의 현재 연결 또는 활동이 정지된 연결을 식별해야 합니다. 식별된 연결은 보호된 대화를 사용할 수 없습니다.

CURRENT

활성 그룹의 현재 연결을 식별합니다. 활성 그룹이 연결된 상태에 있어야 합니다. 현재 연결은 보호된 대화를 사용하지 않아야 합니다.

ALL 또는 **ALL SQL**

활성 그룹의 모든 기존의 연결(리모트 연결과 로컬 연결을 모두 포함)을 식별합니다. 명령문을 실행할 때 어떠한 연결도 없으면, 오류나 경고가 발생하지 않습니다. 모든 연결이 보호된 대화를 사용할 수 없습니다.

주

식별된 연결은 현재 작업 단위중에 SQL문을 실행하는 데 사용했던 연결이 아니어야 하며, 또한 보호된 대화에 대한 연결도 아니어야 합니다. 보호된 대화에서의 연결을 종료하려면 RELEASE문을 사용하십시오. 로컬 연결은 결코 보호된 대화로 간주되지 않습니다.

DISCONNECT문이 성공적으로 수행되면 식별된 각 연결이 종료됩니다. 현재 연결을 차단하면 활성 그룹이 단절 상태가 됩니다.

DISCONNECT문이 성공적으로 수행되지 않으면 활성 그룹의 연결 상태 및 각 연결의 상태가 변하지 않습니다.

CONNECT(유형 1) 구문을 사용한다고 DISCONNECT를 사용할 수 없는 것은 아닙니다.

DISCONNECT는 커서를 닫고 자원을 해제하며, 앞으로의 연결 사용을 막습니다.

ROLLBACK은 DISCONNECT문이 종료시켰던 연결을 재연결하지 않습니다.

리모트 연결을 작성하고 유지보수하는 데는 자원이 필요합니다. 그러므로 다시 사용하지 않을 리모트 연결은 가능한 빨리 종료하고, 다시 사용할 리모트 연결은 차단하지 않아야 합니다.

확약 조작 후에는 즉시 DISCONNECT문을 실행해야 합니다. DISCONNECT문을 사용하여 현재 연결을 종료하는 경우 다음에 실행되는 SQL문은 CONNECT 또는 SET CONNECTION이어야 합니다.

DISCONNECT ALL은 로컬 서버와의 연결을 종료합니다. 연결에 WITH HOLD절로 정의된 열린 커서가 있어도 연결이 종료됩니다.

예

예 1: TOROLAB1과의 연결이 더 이상 필요 없습니다. 확약 조작 후에 다음 명령문을 실행합니다.

```
EXEC SQL DISCONNECT TOROLAB1;
```

예 2: 현재 연결이 더 이상 필요 없습니다. 확약 조작 후에 다음 명령문을 실행합니다.

```
EXEC SQL DISCONNECT CURRENT;
```

예 3: 기존의 연결이 더 이상 필요 없습니다. 확약 조작 후에 다음 명령문을 실행합니다.

```
EXEC SQL DISCONNECT ALL;
```

DROP

DROP문은 오브젝트를 삭제합니다. 삭제되는 오브젝트에 직접 또는 간접적으로 종속되는 모든 오브젝트 또한 삭제됩니다. 오브젝트를 삭제하면 언제나 오브젝트 설명도 카탈로그에서 삭제됩니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

표, 뷰, 색인, 별명 또는 패키지를 제거하려면 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 있어야 합니다.

- 다음과 같은 시스템 권한
  - 제거할 오브젝트에 대한 *OBJOPR 및 *OBJEXIST 시스템 권한
  - 오브젝트가 표 또는 뷰인 경우 해당 표 또는 뷰에 종속되는 모든 뷰, 색인 및 논리 파일에 대한 *OBJOPR 및 *OBJEXIST 시스템 권한
  - 제거할 오브젝트가 들어 있는 라이브러리에 대한 *EXECUTE 시스템 권한
- 관리 권한

스키마를 제거하려면 명령문의 권한부여 ID가 갖는 권한에 다음 중 한 가지 이상의 권한이 있어야 합니다.

- 다음과 같은 시스템 권한
  - 제거할 라이브러리에 대한 *OBJEXIST, *OBJOPR, *EXECUTE 및 *READ 시스템 권한.
  - 스키마에 있는 모든 오브젝트에 대한 *OBJOPR과 *OBJEXIST 시스템 권한 그리고 스키마의 표와 뷰에 종속되는 모든 뷰, 색인 및 논리 파일에 대한 *OBJOPR과 *OBJEXIST 시스템 권한.
  - 스키마에 있는 나머지 오브젝트 유형을 삭제하는 데 필요한 그 밖의 추가 권한. 예를 들어 스키마에 자료 사전이 있는 경우 자료 사전에 대한 *OBJMGT 권한 그리고 저널 리시버에 대한 몇 가지 시스템 자료 권한. 자세한 정보는 iSeries 보안 참조  책을 참조하십시오.
- 관리 권한

고유한 유형을 드롭(drop)하려면 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 다음과 같은 시스템 권한
  - 제거할 고유한 유형에 대한 *OBJOPR 및 *OBJEXIST 시스템 권한

  - SYSTYPES, SYSPARMS 및 SYSROUTINES 카탈로그 표에 대한 DELETE 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

함수를 제거하려면 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 있어야 합니다.

- 다음과 같은 시스템 권한
  - SQL 함수의 경우 해당 함수와 연관된 프로그램 오브젝트에 대한 *OBJEXIST 시스템 권한
  - SYSFUNCS 및 SYSPARMS 카탈로그 표에 대한 DELETE 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

프로시듀어를 제거하려면 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 있어야 합니다.

- 다음과 같은 시스템 권한
  - SQL 프로시듀어의 경우 해당 프로시듀어와 연관된 프로그램 오브젝트에 대한 *OBJEXIST 시스템 권한
  - SYSPROCS 및 SYSPARMS 카탈로그 표에 대한 DELETE 권한
  - 라이브러리 QSYS2에서 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 DELETE 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 DELETE 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *DLT의 시스템 권한을 부여받았습니다.

트리거를 드롭(drop)하려면 명령문의 권한부여 ID가 보유한 권한에 적어도 다음 중 하나가 포함되어야 합니다.

- 권한은 다음과 같습니다.
  - 실제 파일 트리거 제거(RMVPFTRG) 명령에 대한 시스템 권한 *USE
  - 트리거의 주제 표의 경우
    - 주제 표에 대한 ALTER 권한 및
    - 주제 표가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
  - 드롭(drop)되는 트리거가 SQL 트리거인 경우
    - 트리거 프로그램 오브젝트에 대한 시스템 권한 *OBJEXIST 및

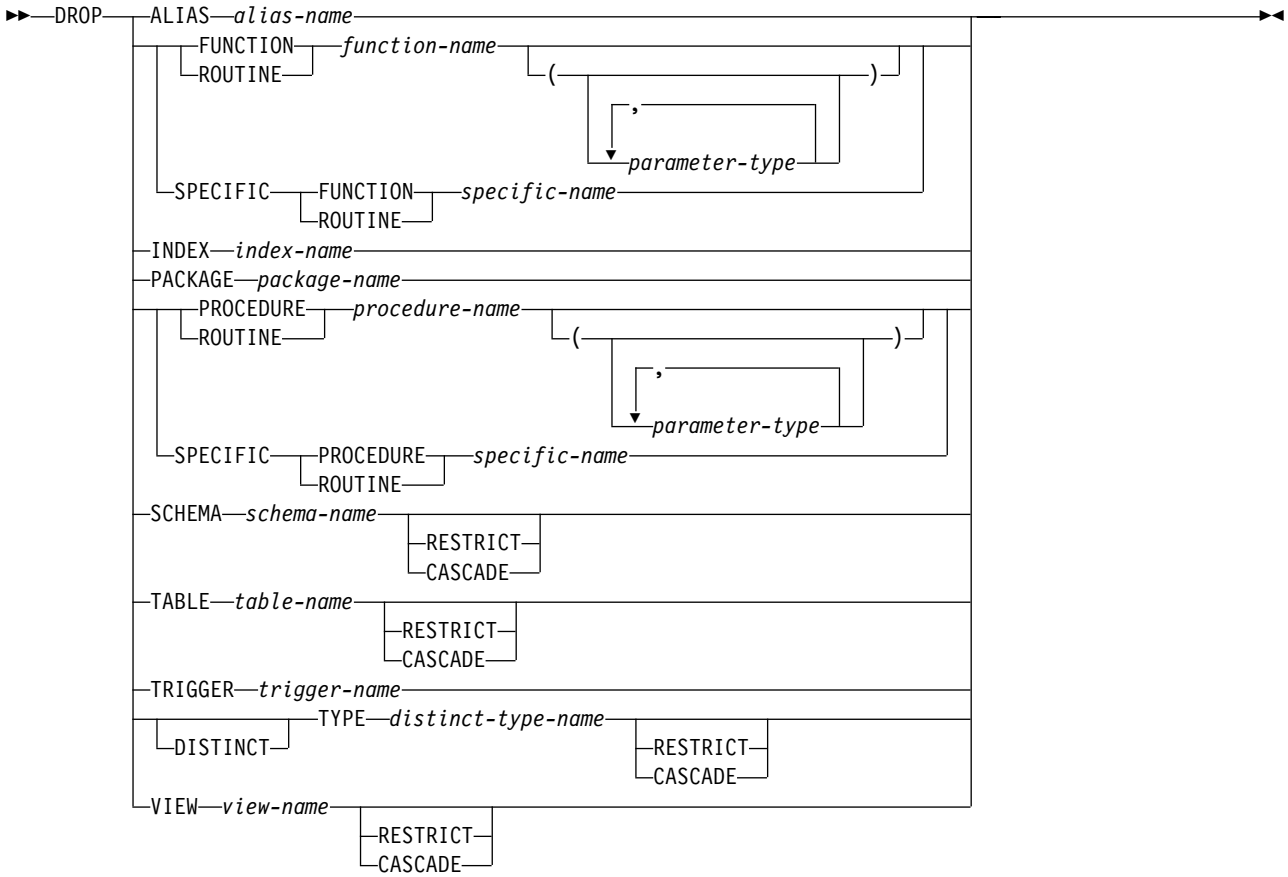
- 트리거가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE

- 관리 권한

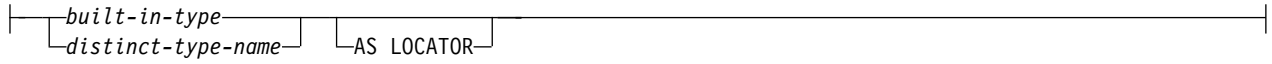
명령문의 권한부여 ID는 다음 중 하나가 참일 때 표에 대해 ALTER 권한을 갖습니다.

- 표의 소유자입니다.
- 표에 대한 ALTER 권한을 부여받았습니다.
- 표에 대한 *OBJALTER나 *OBJMGT의 시스템 권한을 부여받았습니다.

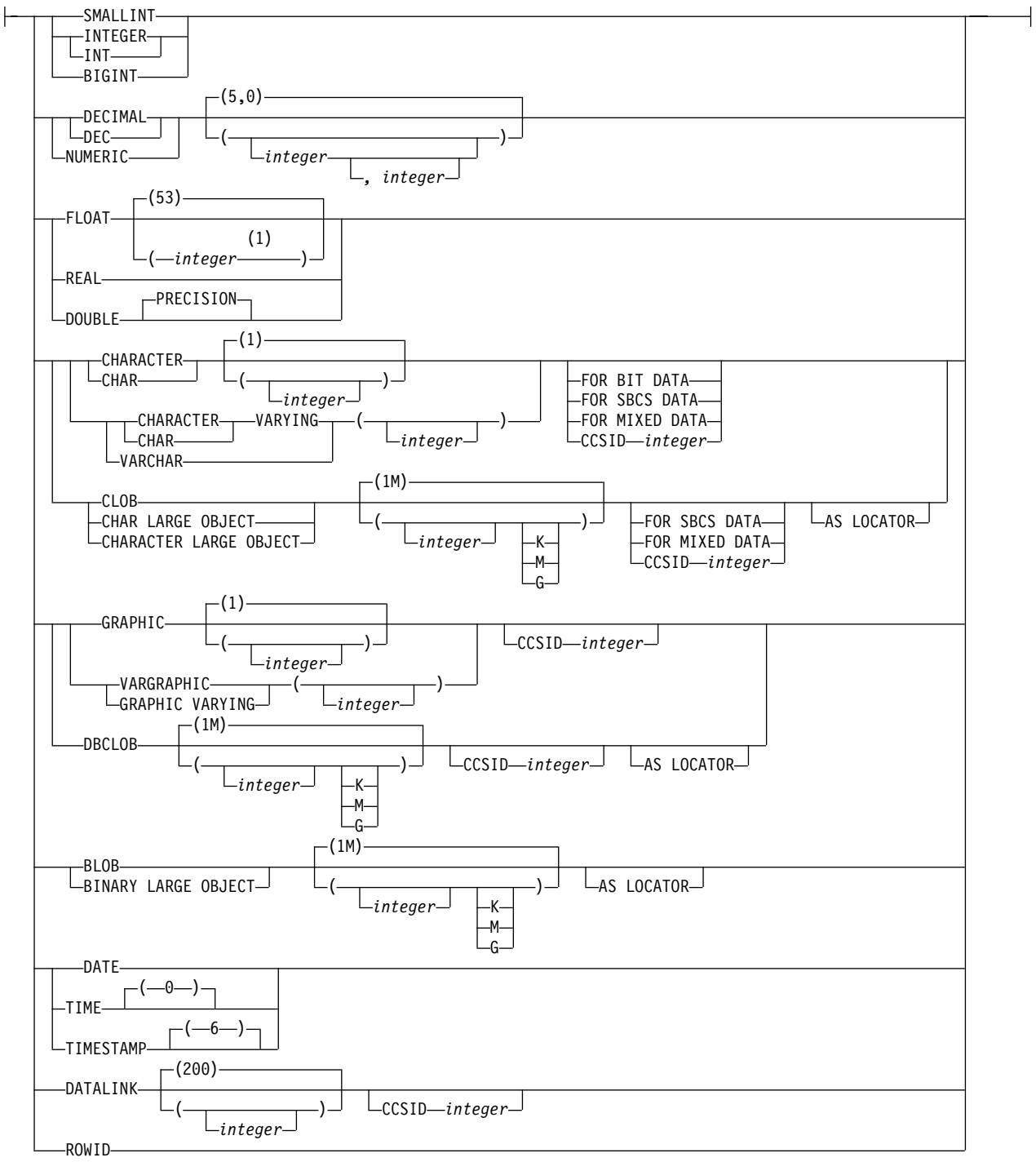
구문



parameter-type:



built-in-type:



주:

1 자료 유형(REAL 또는 DOUBLE)을 기초로 일치가 이루어지므로 정밀도에 대해 지정한 값이 함수를 작성할 때 지정한 값과 일치하지 않아도 됩니다.

설명

ALIAS *alias-name*

제거할 별명을 식별합니다. *alias-name*은 현재 서버에 있는 별명을 식별해야 합니다. 지정된 별명은 스키마에서 삭제됩니다.

별명을 드롭하는 것은 그 별명을 사용하여 정의된 제한사항이나 뷰에는 영향을 미치지 않습니다. 별명이 함수, 패키지, 프로시듀어, 프로그램 또는 트리거에 참조되었는지 여부에 상관없이 그 별명은 드롭될 수 있습니다. 별명을 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 별명이 존재하지 않으면 오류가 리턴됩니다.

FUNCTION

제거할 함수를 식별합니다. 함수의 이름, 함수 표시 또는 특정 이름으로 제거될 특정 함수를 식별할 수 있습니다. 함수 해결(및 경로)에 대한 규칙은 사용되지 않습니다. 지정된 함수는 스키마에서 삭제됩니다. 이것이 SQL 함수이거나 피소스(sourced) 함수이면 그 함수와 연관된 서비스 프로그램(*SRVPGM) 또한 제거됩니다. 외부 함수인 경우에는 CREATE FUNCTION문에 지정된 프로그램 또는 서비스 프로그램에 보관된 정보가 오브젝트에서 제거됩니다. 그 함수에 대한 모든 권한 또한 제거됩니다.

CREATE DISTINCT TYPE문을 사용하여 암시적으로 생성한 함수는 제거할 수 없습니다.

다른 함수가 종속되어 있으면 그 함수는 드롭할 수 없습니다. 함수가 CREATE FUNCTION문의 SOURCE절에서 식별되었다면 다른 함수에 종속된 것입니다. 함수가 함수, 패키지, 프로시듀어, 프로그램, 트리거 또는 뷰에 참조되었는지 여부에 상관없이 그 별명은 드롭될 수 있습니다. 함수를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 함수가 존재하지 않으면 오류가 리턴됩니다.

FUNCTION *function-name*

*function-name*은 현재 서버에 있는 한 함수를 정확히 식별해야 합니다. 지정된 또는 내재된 스키마에 지정된 이름의 함수가 둘 이상 있으면 오류가 리턴됩니다.

FUNCTION *function-name(parameter-type, ...)*

function-name(parameter-type, ...) 현재 서버에 있는 지정된 함수 표시로 함수를 식별해야 합니다. 지정된 매개변수는 대응하는 위치의 CREATE FUNCTION문에 지정되어 있는 자료 유형과 일치해야 합니다. 자료 유형의 수와 자료 유형의 논리적 연결은 제거될 특정 함수 인스턴스를 식별하기 위해 사용됩니다. *function-name()*이 지정되면 식별된 함수는 0개의 매개변수를 가져야 합니다.

function-name

함수 이름을 식별합니다.

(parameter-type, ...)

함수의 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.
- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE FUNCTION문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (길이 8)
```

내재된 길이는 CREATE FUNCTION문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE FUNCTION문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC FUNCTION *specific-name*

*specific-name*은 현재 서버에 있는 특정 함수를 식별해야 합니다.

INDEX *index-name*

제거할 색인을 식별합니다. *index-name*은 현재 서버에 있는 색인을 식별해야 합니다. 지정된 색인은 스키마에서 삭제됩니다.

색인이 함수, 패키지, 프로시듀어, 프로그램 또는 트리거에 참조되었는지 여부에 상관없이 드롭될 수 있습니다. 색인을 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다.

PACKAGE *package-name*

제거할 패키지를 식별합니다. *package-name*은 현재 서버에 있는 패키지를 식별해야 합니다. 지정된 패키지는 스키마에서 삭제됩니다. 해당 패키지에 대한 모든 권한 또한 제거됩니다.

패키지가 함수, 패키지, 프로시듀어, 프로그램 또는 트리거에 참조되었는지 여부에 상관없이 드롭될 수 있습니다. 색인을 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 패키지가 존재하지 않으면 오류가 리턴됩니다.

PROCEDURE

제거할 프로시듀어를 식별합니다. 프로시듀어의 이름, 프로시듀어 표시 또는 특정 이름으로 제거될 특정 프로시듀어를 식별할 수 있습니다. 프로시듀어 해결(및 경로)에 대한 규칙은 사용되지 않습니다.

프로시듀어가 함수, 패키지, 프로시듀어, 프로그램, 트리거 또는 뷰에 참조되었는지 여부에 상관없이 드롭될 수 있습니다. 프로시듀어를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 프로시듀어가 존재하지 않으면 오류가 리턴됩니다.

PROCEDURE *procedure-name*

*procedure-name*은 현재 서버에 있는 한 프로시듀어를 정확히 식별해야 합니다. 프로시듀어는 프로시듀어에 대해 정의된 매개변수를 갖습니다. 지정된 또는 내재된 스키마에 지정된 이름의 프로시듀어가 둘 이상 있는 경우 오류가 리턴됩니다.

PROCEDURE *procedure-name(parameter-type, ...)*

The *procedure-name(parameter-type, ...)*은 현재 서버에 있는 지정된 프로시듀어 표시로 프로시듀어를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE PROCEDURE문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수, 자료 유형의 논리 연결이 제거될 특정 프로시듀어 인스턴스를 식별하는 데 사용됩니다. *procedure-name()*이 지정되면 식별된 프로시듀어는 0개의 매개변수를 가져야 합니다.

procedure-name

프로시듀어 이름을 식별합니다.

(parameter-type, ...)

프로시듀어 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.

- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE PROCEDURE문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

  ```
  CHAR      CHAR(1)
  GRAPHIC   GRAPHIC(1)
  DECIMAL   DECIMAL(5,0)
  FLOAT     DOUBLE (길이 8)
  ```

  내재된 길이는 CREATE PROCEDURE문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE PROCEDURE문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC PROCEDURE *specific-name*

*specific-name*은 현재 서버에 있는 특정 프로시듀어를 식별해야 합니다.

지정한 프로시듀어가 카탈로그 표 SYSPROCS와 SYSPARMS에서 삭제됩니다. 프로시듀어가 SQL 프로시듀어인 경우 지정한 프로시듀어와 연관된 프로그램(*PGM) 오브젝트 또한 제거됩니다. 해당 프로시듀어에 대한 모든 권한도 제거됩니다.

SCHEMA *schema-name*

드롭할 스키마를 식별합니다. *schema-name*은 현재 서버에 있는 스키마를 식별합니다. 지정된 스키마는 삭제됩니다. 스키마의 각 오브젝트는 적절한 DROP문에 드롭 옵션이 지정된 것처럼 드롭됩니다(CASCADE, RESTRICT 또는 어느 것도 아님). 이 오브젝트에 종속되는 오브젝트 처리에 대한 정보는 이 오브젝트 유형의 DROP 설명을 참조하십시오.

DROP SCHEMA는 확약 레벨이 *NONE인 경우에만 유효합니다.

CASCADE도 RESTRICT도 아님

다른 스키마의 함수, 패키지, 프로시듀어, 프로그램, 표 또는 트리거에 참조되었더라도 해당 스키마를 드롭하도록 지정합니다. 이 스키마를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 스키마가 존재하지 않으면 오류가 리턴됩니다.

CASCADE

스키마를 참조하는 모든 트리거가 드롭(drop)되도록 지정합니다. 이 스키마가 다

른 스키마의 함수, 패키지, 프로시듀어 또는 프로그램에 참조된 경우 이 스키마를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 스키마가 존재하지 않으면 오류가 리턴됩니다.

RESTRICT
다른 스키마의 SQL 트리거에 참조되어 있다면 스키마를 드롭할 수 없도록 지정합니다. 이 스키마가 다른 스키마의 함수, 패키지, 프로시듀어 또는 프로그램에 참조된 경우 이 스키마를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 스키마가 존재하지 않으면 오류가 리턴됩니다.

TABLE *table-name*
제거할 표를 식별합니다. *table-name*은 현재 서버에 있는 기본 표를 식별하지만 카탈로그 표를 식별해서는 안됩니다. 지정된 표는 스키마에서 삭제됩니다. 해당 표에 대한 모든 권한, 제한조건 및 트리거도 드롭됩니다.

CASCADE도 RESTRICT도 아님
표가 제한사항, 색인, 트리거 또는 뷰에 참조되어 있어도 드롭되도록 지정합니다. 이 표를 참조하는 모든 색인과 뷰는 드롭됩니다. 표가 함수, 패키지, 프로시듀어, 프로그램 또는 트리거에 참조되는 경우 이 표를 참조하는 액세스 계획은 모두 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 표가 존재하지 않으면 오류가 리턴됩니다.

CASCADE
표가 제한사항, 색인, 트리거 또는 뷰에 참조되어 있어도 드롭되도록 지정합니다. 이 표를 참조하는 모든 색인, 트리거 및 뷰는 드롭됩니다. 표가 함수, 패키지, 프로시듀어 또는 프로그램에 참조되는 경우 이 표를 참조하는 액세스 계획은 모두 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 표가 존재하지 않으면 오류가 리턴됩니다.

RESTRICT
표가 제한사항, 색인, 트리거 또는 뷰에 참조되는 경우 표를 드롭할 수 없도록 지정합니다. 표가 함수, 패키지, 프로시듀어 또는 프로그램에 참조되는 경우 이 표를 참조하는 액세스 계획은 모두 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 표가 존재하지 않으면 오류가 리턴됩니다.

TRIGGER *trigger-name*
드롭할 트리거를 식별합니다. *trigger-name*은 현재 서버에 있는 트리거를 식별해야 합니다. 지정된 트리거는 스키마에서 삭제됩니다. 트리거가 SQL 트리거인 경우 이 트리거와 연관된 프로그램 오브젝트도 스키마에서 삭제됩니다.

DISTINCT TYPE *distinct-type-name*
제거할 고유한 유형을 식별합니다. *distinct-type-name*은 현재 서버에 고유한 유형을 식별해야 합니다. 지정된 유형은 스키마에서 삭제됩니다.

CASCADE도 RESTRICT도 아님

해당 유형을 참조하는 제한사항, 색인, 표 및 뷰가 있는 경우 이 유형은 삭제할 수 없도록 지정됩니다.

또한 드롭될 유형의 매개변수나 리턴 값을 갖는 모든 프로시듀어 또는 함수 R 또는 드롭될 유형에 대한 참조의 경우 다음 DROP문이 실행됩니다.

```
DROP ROUTINE R
```

드롭될 유형을 참조하는 모든 트리거T의 경우 다음과 같은 DROP문이 실행됩니다.

```
DROP TRIGGERT
```

이 명령문이 종속 함수 또는 프로시듀어를 연속적으로 드롭할 수 있습니다. 고유한 유형에 대한 종속성 때문에 모든 함수 또는 프로시듀어가 드롭될 리스트에 있는 경우 고유한 유형이 성공적으로 드롭됩니다. 유형이 패키지나 프로그램에서 참조되는 경우 이 유형을 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 유형이 존재하지 않으면 오류가 리턴됩니다.

CASCADE

유형이 제한사항, 함수, 색인, 프로시듀어, 표, 트리거 또는 뷰에서 참조되었어도 드롭하도록 지정합니다. 이 유형을 참조하는 모든 제한사항, 함수, 색인, 프로시듀어, 표, 트리거 및 뷰도 드롭됩니다. 유형이 패키지나 프로그램에서 참조되는 경우 이 유형을 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 유형이 존재하지 않으면 오류가 리턴됩니다.

RESTRICT

유형이 제한사항, (유형이 작성되었을 때 작성된 함수가 아닌) 함수, 색인, 프로시듀어, 표, 트리거 또는 뷰에서 참조되는 경우 드롭할 수 없도록 지정합니다. 유형이 패키지나 프로그램에서 참조되는 경우 이 유형을 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 유형이 존재하지 않으면 오류가 리턴됩니다.

VIEW *view-name*

제거할 뷰를 식별합니다. *vie-name*은 현재 서버에 있는 뷰를 식별하지만 카탈로그 뷰를 식별해서는 안됩니다. 지정된 뷰는 스키마에서 삭제됩니다. 뷰가 제거될 때 뷰에 대한 모든 권한도 제거됩니다.

CASCADE도 RESTRICT도 아님

뷰가 트리거나 다른 뷰에서 참조되었어도 드롭되도록 지정합니다. 이 뷰를 참조하는 모든 뷰는 드롭됩니다. 뷰가 함수, 패키지, 프로시듀어, 프로그램 또는 트리거에서 참조되는 경우 이 뷰를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 뷰가 존재하지 않으면 오류가 리턴됩니다.

CASCADE

뷰가 트리거나 다른 뷰에서 참조되었어도 드롭되도록 지정합니다. 이 뷰를 참조하는 모든 트리거와 뷰는 드롭됩니다. 뷰가 함수, 패키지, 프로시듀어 또는 프로그램에서 참조되는 경우 이 뷰를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 뷰가 존재하지 않으면 오류가 리턴됩니다.

RESTRICT

뷰가 트리거나 다른 뷰에 참조된 경우 드롭될 수 없도록 지정합니다. 뷰가 함수, 패키지, 프로시듀어 또는 프로그램에서 참조된 경우 표를 참조하는 액세스 계획은 다음 사용시 내재적으로 다시 준비됩니다. 그 때에 뷰가 존재하지 않으면 오류가 리턴됩니다.

주

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 SYNONYM은 ALIAS의 동의어로 사용될 수 있습니다.
- 키워드 DATA는 DISTINCT의 동의어로 사용될 수 있습니다.
- 키워드 PROGRAM은 PACKAGE와 동의어로 사용될 수 있습니다.
- 키워드 COLLECTION은 SCHEMA의 동의어로 사용될 수 있습니다.

예

예 1

MY_IN_TRAY라는 이름의 표를 드롭합니다. 이 표를 통해 뷰나 색인이 작성되는 경우에는 표를 제거할 수 없도록 하십시오.

```
DROP TABLE MY_IN_TRAY RESTRICT
```

예 2

MA_PROJ라는 이름의 뷰를 드롭합니다.

```
DROP VIEW MA_PROJ
```

예 3

이름이 PERS.PACKA인 패키지를 제거합니다.

```
DROP PACKAGE PERS.PACKA
```

예 4

현재 사용하지 않으면 고유한 유형인 DOCUMENT를 드롭합니다.

```
DROP DISTINCT TYPE DOCUMENT RESTRICT
```

예 5

사용자의 이름이 SMITH이고, 스키마 CHEM에서 사용자명으로 된 유일한 함수가 ATOMIC_WEIGHT라고 가정합니다. ATOMIC_WEIGHT를 드롭합니다.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT RESTRICT
```

예 6

사용자의 이름이 SMITH이고, SMITH 스키마에 CENTER 함수를 작성했다고 가정합니다. 드롭할 함수 인스턴스를 식별하는 함수 표시를 사용하여 CENTER를 드롭합니다.

```
DROP FUNCTION CENTER (INTEGER, FLOAT) RESTRICT
```

예 7

사용자의 이름이 SMITH이고, JOHNSON 스키마(shema)에 FOCUS97이란 지정명을 부여하여 또 하나의 CENTER 함수를 작성했다고 가정합니다. 제거할 함수 인스턴스를 식별하는 지정명을 사용하여 CENTER를 드롭합니다.

```
DROP SPECIFIC FUNCTION JOHNSON.FOCUS97
```

예 8

사용자의 이름이 SMITH이고, BIOLOGY 스키마에 OSMOSIS 프로시듀어를 저장했다고 가정합니다. OSMOSIS를 드롭합니다.

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

예 9

사용자의 이름이 SMITH이고, 사용자의 스키마에 BONUS 트리거가 있다고 가정합니다. BONUS를 드롭합니다.

```
DROP TRIGGER BONUS
```

END DECLARE SECTION

END DECLARE SECTION문은 SQL 선언 섹션의 끝을 표시합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. RPG, Java나 REXX에 지정되어서는 안됩니다.

권한부여

필요한 사항이 없습니다.

구문

설명

END DECLARE SECTION문은 호스트 언어의 규칙에 따라서 어플리케이션 프로그램 내에서 선언이 나타날 수 있는 모든 위치에 작성될 수 있습니다. 이 명령문을 사용하여 SQL 선언 섹션의 끝을 표시합니다. SQL 선언 섹션은 BEGIN DECLARE SECTION문으로 시작합니다. BEGIN DECLARE SECTION문에 대한 자세한 정보는 398 페이지의 『BEGIN DECLARE SECTION』을 참조하십시오.

BEGIN DECLARE SECTION과 END DECLARE SECTION문은 쌍을 이루어야 하며, 내포될 수 없습니다.

SQL문이 선언 섹션 안에 들어가서는 안됩니다. 단, DECLARE VARIABLE 문과 INCLUDE문은 예외입니다.

SQL 선언 섹션이 프로그램에 지정되면 SQL 선언 섹션 내에 선언된 변수만 호스트 변수로 사용될 수 있습니다. SQL 선언 섹션을 지정하지 않으면 프로그램에 있는 모든 변수를 호스트 변수로서 사용할 수 있습니다.

RPG와 REXX를 제외한 모든 호스트 언어에 대해 SQL 선언 섹션을 지정하여 소스 프로그램이 IBM SQL 표준을 따르도록 해야 합니다. SQL 선언 섹션은 변수에 대한 첫 번째 참조 이전에 나타나야 합니다. 호스트 변수는 RPG에서 이 명령문을 사용하지 않고 선언되며, REXX에서는 전혀 선언되지 않습니다.

SQL 선언 섹션 외부에 선언된 변수는 SQL 선언 섹션 내부에 선언된 변수와 같은 이름일 수 없습니다.

하나 이상의 SQL 선언 섹션이 프로그램에 지정될 수 있습니다.

예

END DECLARE SECTION문의 사용 예를 보려면 398 페이지의 『BEGIN DECLARE SECTION』을 참조하십시오.

EXECUTE

EXECUTE문은 준비된 SQL문을 실행합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

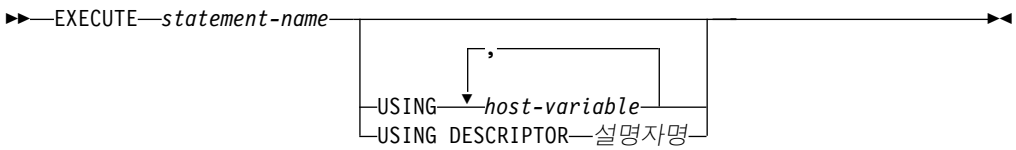
권한부여

권한부여 규칙은 EXECUTE에 의해 지정되는 SQL문에 대해 정의된 규칙입니다. 예를 들어, EXECUTE를 사용하여 INSERT문을 실행할 때 적용되는

권한부여 규칙에 대해서는 INSERT에 대한 설명을 참조하십시오.

프로그램이 작성되었을 때 DYNUSRPRF(*OWNER)가 CRTSQLxxx 명령에 지정되지 않았으면, 명령문의 권한부여 ID는 실행 시간 권한부여 ID입니다. 자세한 내용은 58 페이지의 『권한부여 ID 및 권한부여명』을 참조하십시오.

구문



설명

statement-name

실행할 준비된 명령문을 식별합니다. *state-name*은 이전에 이미 준비된 명령문을 식별해야 합니다. SELECT문은 준비된 명령문일 수 없습니다.

USING

준비된 명령문의 매개변수 마커(의문 부호)로 대체된 값을 갖는 호스트 변수의 리스트를 소개합니다. 매개변수 마커에 대한 설명은 728 페이지의 『PREPARE』를 참조하십시오. 준비된 명령문에 매개변수 마커가 포함되어 있는 경우 USING절을 사용해야 합니다. 매개변수 마커가 없을 때에는 USING을 무시합니다.

host-variable,...

호스트 구조 및 변수의 선언 규칙에 따라서 프로그램에 선언해야 하는 호스트 구조 또는 변수 중 하나를 식별합니다. 호스트 구조에 대한 참조는 해당 변수 각각에 대한 참조로 대체됩니다. 변수의 수는 준비된 명령문에 있는 매개변수 마커의 수와 동일해야 합니다. *n* 번째 변수는 준비된 명령문에서 *n* 번째 매개변수 마커에 해당합니다.

DESCRIPTOR *descriptor-name*

호스트 변수의 유효한 설명을 포함해야 하는 SQLDA를 식별합니다.

EXECUTE문을 처리하기 전에 SQLDA에 다음과 같은 필드를 설정해야 합니다(REXX에 대한 규칙은 다릅니다. 자세한 내용은 SQL Programming with Host Languages 책을 참조하십시오).

- SQLDA에 제공된 SQLVAR의 발생 수를 지정하는 SQLN
- SQLDA에 대해 할당된 기억장치의 바이트 수를 지정하는 SQLDABC
- 명령문을 처리할 때 SQLDA에 사용되는 변수의 수를 지정하는 SQLD
- 변수 속성을 표시하는 SQLVAR 발생

모든 SQLVAR 발생을 포함할 수 있도록 SQLDA는 충분한 기억장치 공간을 가져야 합니다. LOB 또는 고유한 유형이 결과에 있는 경우 추가적인 SQLVAR 항목이 각 매개변수에 대하여 존재해야 합니다. SQLVAR에 대한 설명이 포함된 SQLDA에 대한 자세한 정보 및 SQLVAR 발생 횟수 판별 방법에 대한 설명은 881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』를 참조하십시오.

SQLD는 영(0) 보다 크거나 같고 SQLN 보다 작거나 같은 값으로 설정되어야 합니다. 또한 준비된 명령문에 있는 매개변수 마커의 수와 동일해야 합니다. SQLDA에 의해 설명되는 *n* 번째 변수는 준비된 명령문에서 *n* 번째 매개변수 마커에 해당합니다.

RPG/400은 포인터 설정을 위한 함수를 제공하지 않습니다. SQLDA는 해당 호스트 변수를 위치지정하는 데 포인터를 사용하므로 사용자는 RPG/400 어플리케이션 밖에서 이 포인터를 설정해야 합니다.

주

매개변수 마커 대체

준비된 명령문을 실행하기 전에 명령문에 있는 각 매개변수 마커가 해당 호스트 변수로 대체됩니다. 매개변수 마커의 대체는 소스가 호스트 변수 값이고 목표가 데이터베이스 관리자 내의 변수인 할당 연산입니다. 유형 매개변수 마커의 경우 목표 변수의 속성은 CAST 스펙에서 지정한 것입니다. 비유형 매개변수 마커의 경우 목표 변수의 속성은 매개변수 마커의 문맥에 따라 결정됩니다. 매개변수 마커에 영향을 미치는 규칙은 732 페이지의 표 57을 참조하십시오.

매개변수 마커 P에 해당하는 호스트 변수를 V로 표시하십시오. 열에 값을 지정하는 데 적용되는 규칙에 따라서 값 V가 P에 대한 목표 변수에 지정됩니다. 그러므로 다음이 적용됩니다.

- V는 목표와 호환되어야 합니다.

- V가 숫자인 경우 정수부의 절대 값은 목표의 정수부의 최대 절대값 이하이어야 합니다.
- V의 속성이 목표의 속성과 동일하지 않을 경우 목표의 속성과 일치되도록 값이 변환됩니다.
- 목표에 널(null)이 포함될 수 없을 경우 값 V도 널이 아니어야 합니다.

그러나 열에 값을 지정하는 데 적용되는 규칙과 달리 다음이 적용됩니다.

- V가 스트링인 경우 그 길이가 목표의 길이 속성보다 크면 값이 절단됩니다(오류는 발생하지 않음).

준비된 명령문을 실행할 때 P 대신 사용한 값은 P에 대한 목표 변수의 값입니다. 예를 들어, V가 CHAR(6)이고 목표는 CHAR(8)인 경우 P 대신 사용된 값은 두 개의 공백으로 채워진 V 값입니다.

예

COBOL 프로그램의 부분인 이 예는 매개변수 마커가 있는 INSERT문이 준비되고 실행되는 방법을 보여줍니다.

```
   EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
     77  EMP              PIC X(6).
     77  PRJ              PIC X(6).
     77 ACT              PIC S9(4) COMP-4.
     77 TIM              PIC S9(3)V9(2).
     01 HOLDER.
        49  HOLDER-LENGTH    PIC S9(4) COMP-4.
        49  HOLDER-VALUE     PIC X(80).
   EXEC SQL  END DECLARE SECTION  END-EXEC.
   .
   .
   .
   MOVE 70 TO HOLDER-LENGTH.
   MOVE "INSERT INTO EMPPROJACT (EMPNO, PROJNO, ACTNO, EMPTIME)
 -              "VALUES (?, ?, ?, ?)" TO HOLDER-VALUE.
   EXEC SQL  PREPARE MYINSERT FROM :HOLDER  END-EXEC.

   IF SQLCODE = 0
     PERFORM DO-INSERT THRU END-DO-INSERT
ELSE     PERFORM ERROR-CONDITION.

   DO-INSERT.
     MOVE "000010" TO EMP.
     MOVE "AD3100" TO PRJ.
     MOVE 160      TO ACT.
     MOVE .50      TO TIM.
     EXEC SQL  EXECUTE MYINSERT USING :EMP, :PRJ, :ACT, :TIM  END-EXEC.
   END-DO-INSERT.
   .
   .
   .
```

EXECUTE IMMEDIATE

EXECUTE IMMEDIATE문:

- 명령문의 문자 스트링 형식으로부터 실행가능한 형식의 SQL문을 준비합니다.
- SQL문을 실행합니다.

EXECUTE IMMEDIATE는 PREPARE문과 EXECUTE문의 기본 함수를 결합시킵니다. 이 명령문을 사용하여 호스트 변수와 매개변수 마커가 모두 없는 SQL문을 준비하고 실행할 수 있습니다.

호출

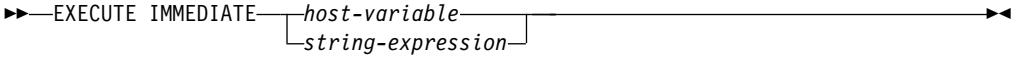
이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

권한부여 규칙은 EXECUTE IMMEDIATE에 의해 지정되는 SQL문에 대해 정의된 규칙입니다. 예를 들어, EXECUTE IMMEDIATE문을 사용하여 INSERT문을 실행할 때 적용되는 권한부여 규칙에 대해서는 708 페이지의 『INSERT』를 참조하십시오.

프로그램이 작성되었을 때 DYNUSRPRF(*OWNER)가 CRTSQLxxx 명령에 지정되지 않았으면, 명령문의 권한부여 ID는 실행 시간 권한부여 ID입니다. 자세한 내용은 58 페이지의 『권한부여 ID 및 권한부여명』을 참조하십시오.

구문



설명

host-variable

문자 스트링 선언 규칙에 따라 선언해야 하는 호스트 변수 또는 UCS-2 그래픽 호스트 변수를 식별합니다. 호스트 변수의 자료 유형은 CLOB나 DBCLOB여야 하고 인디케이터 변수는 지정하지 않아야 합니다.

string-expression

*string-expression*은 문자 스트링을 생성하는 모든 PL/I *string-expression*입니다. 문자 스트링을 생성하는 SQL 표현식은 허용되지 않습니다. *string-expression*은 PL/I에서만 허용됩니다.

식별된 호스트 변수 또는 스트링 표현식의 값을 *statement string*이라고 합니다.

명령문 스트링은 다음 SQL문 중 하나여야 합니다.[55]

ALTER	DROP	REVOKE
CALL	GRANT	ROLLBACK
COMMENT	INSERT	SET PATH
COMMIT	LABEL	SET TRANSACTION
CREATE	LOCK TABLE	UPDATE
DELETE	RENAME	

명령문 스트링에는 다음의 경우가 허용되지 않습니다.

- EXEC SQL로 시작하고 END-EXEC 또는 세미콜론(;)으로 끝나는 경우
- 호스트 변수에 대한 참조를 포함하는 경우
- 매개변수 마커를 포함하는 경우

EXECUTE IMMEDIATE문을 실행할 때 지정한 명령문 스트링이 분석되어 오류 여부가 검사됩니다. 유효하지 않은 SQL문은 실행되지 않으며, 실행을 막는 오류 조건이 SQLCA에 보고됩니다. 유효한 SQL문이지만 실행 중 오류가 발생하면 오류 조건이 SQLCA에 보고됩니다.

주

동일한 SQL문을 두 번 이상 실행하는 경우에는 EXECUTE IMMEDIATE문을 사용하는 것보다 PREPARE문 및 EXECUTE문을 사용하는 것이 더 효과적입니다.

예

호스트 변수 Qstring에서 SQL문을 실행하기 위해 C를 사용합니다.

```
void main ()
  {

   EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.

    char Qstring[100] = "INSERT INTO WORK_TABLE SELECT * FROM EMPPROJACT
     WHERE ACTNO >= 100";

   EXEC SQL  END DECLARE SECTION  END-EXEC.
   EXEC SQL  INCLUDE SQLCA;
  .
  .
  .
    EXEC SQL EXECUTE IMMEDIATE :Qstring;

    return;
  }
```

55. select문은 사용할 수 없습니다. select문을 동적으로 처리하기 위해서는 PREPARE, DECLARE CURSOR 및 OPEN문을 사용하십시오.

FETCH

FETCH문은 결과표의 한 행에 커서의 위치를 지정합니다. 이 명령문은 하나 또는 여러 개의 행을 리턴하거나 한 행도 리턴하지 않을 수 있으며, 호스트 변수에 리턴되는 행의 값을 지정합니다.

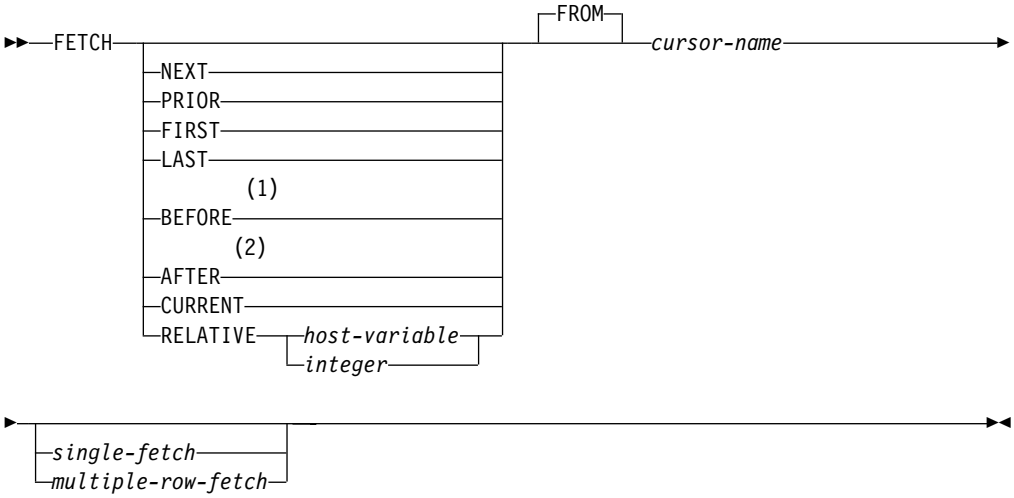
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. REXX 프로시듀어에서는 복수 행 페치가 허용되지 않습니다.

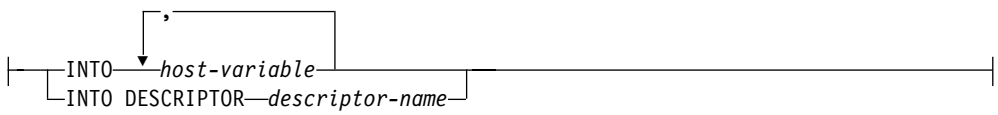
권한부여

커서를 사용하는 데 필요한 권한부여 설명에 대해서는 598 페이지의 『DECLARE CURSOR』를 참조하십시오.

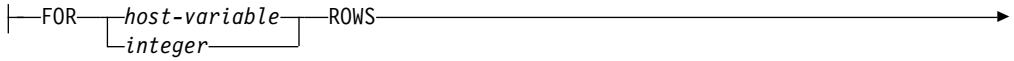
구문

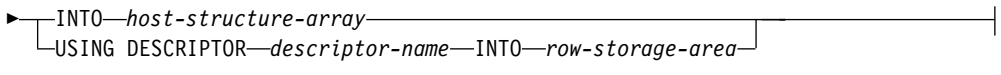


single-fetch:

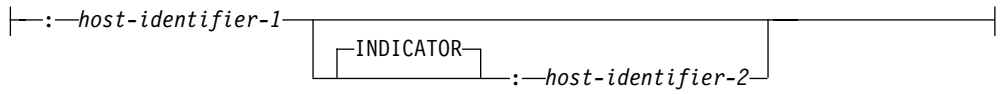


multiple-row-fetch:





row-storage-area:



주:

1 BEFORE를 지정하는 경우 단일 페치절 또는 복수 행 페치절을 지정하지 않아야 합니다.

2 AFTER을 지정하는 경우 단일 페치절 또는 복수 행 페치절을 지정하지 않아야 합니다.

설명

키워드 NEXT, PRIOR, FIRST, LAST, BEFORE, AFTER, CURRENT 및 RELATIVE가 커서의 새로운 위치를 지정합니다. 이 키워드 중 NEXT만을 SCROLL이 선언되지 않은 커서에 사용할 수 있습니다.

NEXT
결과표에서 커서의 현재 위치에서 다음 행으로 커서를 위치시킵니다. 커서의 다른 방향이 지정되지 않은 경우 NEXT가 디폴트입니다.

PRIOR
결과표에서 커서의 현재 위치에서 이전 행으로 커서를 위치시킵니다.

FIRST
결과표의 첫 번째 행으로 커서를 위치시킵니다.

LAST
결과표의 마지막 행으로 커서를 위치시킵니다.

BEFORE
결과표의 첫 번째 행 앞에 커서를 위치시킵니다.

AFTER
결과표의 마지막 행 뒤에 커서를 위치시킵니다.

CURRENT
커서의 위치를 변경시키지 않으며 현재 위치를 유지보수합니다. 커서를 DYNAMIC SCROLL로서 선언하고 현재 행이 갱신하여 결과표의 정렬 순서에서 위치를 변경시킨 경우 오류가 리턴됩니다.

RELATIVE
Host-variable 또는 *integer*에 정수 값 *k*가 지정됩니다. RELATIVE는 *k*>0인 경우 현재 행 다음의 *k* 행 또는 *k*<0인 경우 현재 행 앞의 *k* 행인 결과 표에 있는

행에 커서를 위치시킵니다. *host-variable*을 지정할 경우 반드시 0 배율의 숫자 변수이어야 하며 인디케이터 변수를 포함하지 않는 것이어야 합니다.

표 50. 유사한 화면이동 스펙

스펙	대체
RELATIVE +1	NEXT
RELATIVE -1	PRIOR
RELATIVE 0	CURRENT

FROM

이 키워드는 단지 분명히 하기 위한 것입니다. 스크롤 위치 옵션을 지정할 때는 이 키워드가 필요합니다. 화면이동 옵션을 지정하지 않는 경우 FROM 키워드를 생략할 수 있습니다.

cursor-name

페치 조작에 사용될 커서를 식별합니다. *cursor-name*은 599 페이지의 『설명』에서 DECLARE CURSOR문에 대해 설명한 대로 선언된 커서를 식별해야 합니다. FETCH문을 실행할 때 커서는 열린 상태에 있어야 합니다.

단일 또는 복수 행 페치절을 지정하지 않으면 사용자에게 리턴되는 자료가 없습니다. 그러나 커서의 위치는 지정되며, 행 잠금이 사용지정될 수도 있습니다. 잠금에 대한 자세한 정보는 24 페이지의 『분리 레벨』을 참조하십시오.

single-fetch

INTO *host-variable*,...

호스트 구조 및 변수의 선언 규칙에 따라서 프로그램에 선언해야 하는 하나 이상의 호스트 구조 또는 변수를 식별합니다. INTO의 조작 형식에서 호스트 구조는 해당 변수 각각에 대한 참조로 대체됩니다. 결과 행의 첫 번째 값이 리스트의 첫 번째 호스트 변수에 지정되고, 두 번째 값은 두 번째 호스트 변수에 지정되는 방법으로 행의 값이 변수에 지정됩니다.

INTO DESCRIPTOR *descriptor-name*

0개 또는 여러 개의 호스트 변수에 대한 유효한 설명이 있어야 하는 SQLDA를 식별합니다.

FETCH문을 처리하기 전에 SQLDA에 다음과 같은 필드를 설정해야 합니다(REXX에 대한 규칙은 다릅니다. 자세한 내용은 호스트 언어로 SQL 프로그래밍 책을 참조하십시오).

- SQLDA에 제공된 SQLVAR 발생 수를 표시하는 SQLN
- SQLDA에 대해 할당된 기억장치의 바이트 수를 표시하는 SQLDABC
- 명령문을 처리할 때 SQLDA에서 사용된 변수 수를 표시하는 SQLD
- 변수의 속성을 지정하는 SQLVAR 발생 수

모든 SQLVAR 발생을 포함할 수 있도록 SQLDA는 충분한 기억장치 공간을 가져야 합니다. 따라서, SQLDABC의 값은 16 + SQLN*(80) 보다 크거나 같아야 합니다. 여기서 80은 SQLVAR 발생의 길이입니다.

SQLD는 영(0) 보다 크거나 같고 SQLN 보다 작거나 같은 값으로 설정되어야 합니다. 자세한 내용은 881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』를 참조하십시오.

multiple-row-fetch

FOR *k* **ROWS**

host-variable 또는 *integer*를 정수값 *k*와 비교 평가합니다. *host-variable*을 지정할 경우 반드시 0 배율의 숫자 호스트 변수이어야 하며 인디케이터 변수를 포함하지 않는 것이어야 합니다. *k*는 1 - 32767 사이에 있어야 합니다. 커서는 방향 키워드(NEXT 등)에 의해 지정되는 행에 놓여지고, 커서가 놓여진 행은 페치됩니다. 그런 다음, 커서의 끝에 도달할 때까지 다음 *k*-1 행이 페치됩니다(표에서 정방향으로 이동). 페치 조작 후, 커서는 마지막으로 페치된 행에 놓입니다.

예를 들어, FETCH PRIOR FROM C1 FOR 3 ROWS는 이전 행, 현재 행 및 다음 행을 순서대로 리턴시킵니다. 커서는 다음 행에 놓입니다. FETCH RELATIVE -1 FROM C1 FOR 3 ROWS도 동일한 결과를 리턴합니다. FETCH FIRST FROM C1 FOR :x ROWS는 처음 *x*개의 행을 리턴한 다음, 커서를 행 번호 *x*에 남겨 둡니다.

복수 행 페치가 성공적으로 실행되면 SQLCA에 다음 세 개의 변수가 설정됩니다.

- SQLERRD(3)은 검색되는 행의 수를 나타냅니다.
- SQLERRD(4)에는 검색되는 행의 길이가 들어 있습니다.
- SQLERRD(5)에는 마지막 행이 페치된 경우에 +100이 있습니다.[56]

INTO *host-structure-array*

*host-structure-array*는 호스트 구조 선언 규칙에 따라 정의되는 호스트 구조의 배열을 식별합니다.

배열의 첫 번째 구조가 첫 번째 행에 대응되고, 두 번째 구조는 두 번째 행에 대응되는 방법으로 구조와 행이 대응됩니다. 또한 행의 첫 번째 값은 구조의 첫 번째 항목에 대응되고, 행의 두 번째 값은 구조의 두 번째 항목에 대응되는 방법으로 행의 값과 구조의 항목이 대응됩니다. 페치되는 행의 수는 호스트 구조 배열의 차원 이하이어야 합니다.

USING DESCRIPTOR *descriptor-name*

row-storage-area에 있는 행의 형식을 설명하는 호스트 변수(0 개 또는 여러 개)에 대한 유효한 설명이 있어야 하는 SQLDA를 식별합니다.

FETCH문을 처리하기 전에 SQLDA에 다음과 같은 필드를 설정해야 합니다.

---

56. 리턴된 행 수가 요청된 행 수와 같은 경우, 자료 끝 경고가 발생하지 않았으며 SQLERRD(5)는 +100을 포함하지 않습니다.

- SQLDA에 제공된 SQLVAR의 발생 수를 지정하는 SQLN
- SQLDA에 대해 할당된 기억장치의 바이트 수를 지정하는 SQLDABC
- 명령문을 처리할 때 SQLDA에 사용되는 변수의 수를 지정하는 SQLD
- 호스트 변수의 속성을 지정하는 SQLVAR 발생

SQLDA의 그 밖의 필드의 값(SQLNAME 등)은 FETCH문이 실행된 다음에는 정의하지 못하며, 사용해서도 안됩니다.

모든 SQLVAR 발생을 포함할 수 있도록 SQLDA는 충분한 기억장치 공간을 가져야 합니다. 따라서, SQLDABC의 값은 16 + SQLN*(80) 보다 크거나 같아야 합니다. 여기서 80은 SQLVAR 발생의 길이입니다. LOB나 고유한 유형이 지정되는 경우 각 매개변수 마커에 대해 두 SQLVAR 항목이 있어야 하며 SQLN이 매개변수 마커 숫자의 두 배로 설정되어야 합니다.

SQLD는 영(0) 보다 크거나 같고 SQLN 보다 작거나 같은 값으로 설정되어야 합니다. 자세한 내용은 881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』를 참조하십시오.

FETCH문이 완료될 때 첫 번째 SQLVAR 항목의 SQLDATA 포인터는 첫 번째 행에 할당된 기억장치의 첫 번째 열에 대해 리턴된 값을 제출하고, 두 번째 SQLVAR 항목의 SQLDATA 포인터는 첫 번째 행에 할당된 기억장치의 두 번째 열에 대해 리턴된 값을 제출하는 방법으로 값을 제출합니다. 첫 번째 널(null)가능 SQLVAR 항목의 SQLIND 포인터는 첫 번째 인디케이터 값을 제출하고, 두 번째 널가능 SQLVAR 항목의 SQLIND 포인터는 두 번째 인디케이터 값을 제출하는 방법으로 인디케이터 값을 제출합니다. SQLDA는 16바이트 경계에 할당되어야 합니다.

INTO *row-storage-area*

호스트 변수와 함께 지정되는 *host-idendifier-1*은 행을 리턴하는 기억장치의 할당을 식별합니다. 행은 SQLDA에서 설명되는 형식으로 기억 영역에 리턴됩니다. *host-identifier-1*은 요구된 모든 행을 보유하기에 충분할 만큼 커야 합니다.

*host-identifier-2*는 선택적 인디케이터 영역을 식별합니다. ISQLVAR 발생의 SQLTYPE이 널가능일 경우 이 ID를 반드시 지정해야 합니다. 인디케이터는 작은 정수로서 리턴됩니다. *host-identifier-2*는 리턴되는 각 행의 각 널가능 값에 대한 인디케이터가 들어갈 수 있을 정도로 커야 합니다.

INTO절에 의해 식별되거나 SQLDA에서 설명된 *n 번째* 호스트 변수는 커서의 결과표에서 *n 번째* 열에 해당합니다. 각 호스트 변수의 자료 유형은 대응되는 열과 호환될 수 있어야 합니다.

변수에 대한 각 지정은 41 페이지의 제 2 장『언어 요소』에 설명된 규칙에 따라 이루어집니다. 변수의 수가 행에 있는 값의 수 미만인 경우 SQLCA의 SQLWARN3 필드

가 'W'로 설정됩니다. 결과 열의 수보다 변수가 더 많을 경우 경고가 발행되지 않음에 유의하십시오. 그 값이 널(null)인 경우 인디케이터 변수를 제공해야 합니다. 지정 오류가 발생하면 값이 변수에 지정되지 않으며 더 이상 어떤 값도 변수에 지정되지 않습니다. 변수에 이미 지정된 값은 그대로 남아 있습니다.

외부 SELECT문의 SELECT 리스트에서의 연산식(0으로 나눔 또는 넘침)으로 인한 오류가 발생하거나 문자 변환 오류가 발생하는 경우 결과는 널값입니다. 그 밖의 널값 경우에서는 인디케이터 변수를 제공해야 합니다. 호스트 변수의 값은 정의되지 않습니다. 그러나 이 경우에 인디케이터 변수는 -2로 설정됩니다. 그리고 오류가 발생하지 않았던 것처럼 명령문의 처리가 계속됩니다(그러나 이 오류는 양의 SQLCODE를 유발합니다). 인디케이터 변수를 제공하지 않으면 SQLCA의 SQLCODE 필드에 음의 값이 리턴됩니다. 호스트 변수에 이미 동일한 값이 지정되어 있고, 오류가 발생할 때 그 값이 지정된 상태로 있을 가능성도 있습니다.

LOB인 결과 열이 있거나 현재 리모트 서버에 연결되어 있는 경우 복수 행 페치는 사용할 수 없습니다.

주

열린 커서에 대해 다음 세 위치가 가능합니다.

- 행 앞 위치
- 행 위치
- 마지막 행 다음 위치

커서가 한 행에 놓여 있는 경우 그 행을 커서의 현재 행이라고 합니다. UPDATE문 또는 DELETE문에 참조되는 커서는 행에 놓여 있어야 합니다. FETCH문을 실행한 결과 커서가 행에만 놓여질 수 있습니다.

커서의 상태를 예측할 수 없게 만드는 오류가 발생할 수도 있습니다.

지정한 호스트 변수가 문자이며 결과가 들어갈 수 있을 정도로 크지 않으면 'W'가 SQLCA의 SQLWARN1에 지정됩니다. 인디케이터 변수를 제공하면 결과의 실제 길이가 호스트 변수와 연관된 인디케이터 변수에 리턴됩니다.

지정한 호스트 변수가 C NUL 종료 호스트 변수이고 결과와 NUL 종료자가 들어갈 정도로 크지 않은 경우 다음이 적용됩니다.

- *CNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정(또는 CNULRQD(*YES) 옵션을 SET OPTION문에 지정)하면 다음이 발생합니다.
  - 결과가 절단됩니다.
  - 마지막 문자는 NUL 종료자입니다.
  - 값 ‘W’가 SQLCA의 SQLWARN1에 지정됩니다.

- *NOCNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정(또는 CNULRQD(*NO) 옵션을 SET OPTION문에 지정)하면 다음이 발생합니다.
  - NUL 종료자는 리턴되지 않습니다.
  - 값 'N'이 SQLCA의 SQLWARN1에 지정됩니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- USING DESCRIPTOR을 단일 페치절에서 INTO DESCRIPTOR의 동의어로 사용할 수 있습니다.

예

두 개의 표 FORUM과 ARCHIVE 각각에 다음과 같은 열이 있습니다.

이름:	FORUM	RECEIVED	SOURCE	TOPIC	ENTRY_TEXT
유형:	char(8) 널이 아님	시간소인 널이 아님	char(8) 널이 아님	char(64) 널이 아님	varchar(4000) 널이 아님
설명:	포럼명	수신된 날짜 및 시간 항목	개인 추가 항목에 대한 사용자 ID	포럼 내의 주제	이 항목표에 추가된 텍스트

FORUM 표에는 이름이 지정된 포럼이 여러 개 포함되어 있습니다. 각 포럼은 다시 하나 이상의 주제를 포함하며, 각 주제는 하나 이상의 항목을 포함하고 있습니다. 주제가 더 이상 현재의 것이 아닐 때 해당 항목이 삭제되거나 ARCHIVE 표로 이동됩니다.

다음 PL/I 프로그램은 포럼 표에 대한 유지보수를 수행하는 데 사용됩니다. 사용자는 세 가지 명령 중 하나를 사용하여 프로그램을 호출할 수 있습니다. 주어진 주제(전체 TOPIC 값일 필요는 없음)에 대한 항목의 TOPIC 열 안에서 찾을 수 있는 텍스트의 스트링이 각 명령에 수반됩니다. 세 가지 명령은 다음과 같습니다.

- 1(모든 주제 항목에 대한 TOPIC 값의 내용을 변경합니다.)
- 2(해당 주제의 모든 항목을 ARCHIVE 표로 이동시킵니다.)
- 3(해당 주제의 모든 항목을 *보존하지* 않고 삭제합니다.)

```
 CLEANUP:  PROC OPTIONS(MAIN);
   DCL NOT_END BIT(1);
  EXEC SQL  BEGIN DECLARE SECTION;
   DCL  ACTION       BINARY FIXED(15);   /* 1=chg-topic   2=archive   3=delete */
   DCL  SRCH_FORUM   CHAR(8);
   DCL  SRCH_TOPIC   CHAR(66) VARYING;
   DCL  NEW_TOPIC    CHAR(64) VARYING;
   DCL  FORUM        CHAR(8);
   DCL  TSTMP        CHAR(26);
   DCL  PERSON       CHAR(8);
   DCL  TOPIC        CHAR(64) VARYING;
   DCL  TXT          CHAR(2000) VARYING;
  EXEC SQL  END DECLARE SECTION;
  EXEC SQL  INCLUDE SQLCA;
 EXEC SQL  WHENEVER NOT FOUND CONTINUE;
 EXEC SQL  WHENEVER SQLWARNING CONTINUE;
 EXEC SQL  WHENEVER SQLERROR GOTO ERRCHK;

 EXEC SQL  CONNECT TO TOROLAB3;
 GET LIST (ACTION, SRCH_FORUM, SRCH_TOPIC, NEW_TOPIC);
 SRCH_TOPIC = '%' || SRCH_TOPIC || '%';
 EXEC SQL  DECLARE CUR CURSOR FOR
             SELECT * FROM FORUM
                 WHERE FORUM = :SRCH_FORUM AND TOPIC LIKE :SRCH_TOPIC
                 FOR UPDATE OF TOPIC;
 EXEC SQL  OPEN CUR;

 NOT_END = '1'B;
 DO WHILE (NOT_END);
   EXEC SQL  FETCH CUR INTO :FORUM, :TSTMP, :PERSON, :TOPIC, :TXT;
  IF SQLSTATE = '02000' THEN
     NOT_END = '0'B;
   ELSE DO;
     SELECT;
       WHEN (ACTION = 1)               /* change topic value */
         EXEC SQL  UPDATE FORUM
                     SET TOPIC = :NEW_TOPIC
                     WHERE CURRENT OF CUR;
       WHEN (ACTION = 2)               /* archive entry to another table */
         DO;
           EXEC SQL  INSERT INTO ARCHIVE
                       VALUES (:FORUM, :TSTMP, :PERSON, :TOPIC, :TXT);
         EXEC SQL  DELETE FROM FORUM WHERE CURRENT OF CUR;
  END;
       WHEN (ACTION = 3)               /* delete topic */
         EXEC SQL  DELETE FROM FORUM WHERE CURRENT OF CUR;
     END; /* select */
   END;  /* else do */
 END;  /* do while */

FINISHED:
 EXEC SQL  CLOSE CUR;
  EXEC SQL  COMMIT WORK;
  RETURN;
ERRCHK:
 DISPLAY ('Unexpected Error -changes will be backed out');
 PUT SKIP LIST (SQLCA);
    EXEC SQL  WHENEVER SQLERROR CONTINUE;  /* continue if error on rollback */
  EXEC SQL  ROLLBACK WORK;
  RETURN;
END;  /* CLEANUP */
```

FREE LOCATOR

FREE LOCATOR문은 로케이터 변수와 변수값 사이의 관계를 제거합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 대화식으로는 발행될 수 없습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다. 그러나 준비된 명령문을 실행하는 데는 USING절을 사용한 EXECUTE문을 사용해야 합니다. FREE LOCATOR문은 EXECUTE IMMEDIATE문과 함께 사용할 수 없습니다.

권한부여

필요한 사항이 없습니다.

구문

설명

host-variable,...

로케이터 변수의 선언 규칙에 따라서 프로그램에 선언해야 하는 하나 이상의 호스트 변수를 식별합니다. 인디케이터 변수는 지정하지 않아야 합니다. 로케이터 변수 유형은 2진 큰 오브젝트 로케이터, 문자 큰 오브젝트 로케이터 또는 2바이트 문자 큰 오브젝트 로케이터여야 합니다.

호스트 변수에는 현재 지정된 로케이터가 있어야 합니다. 다시 말해, 로케이터가 이 작업 단위중 CALL, FETCH, SELECT INTO, SET 변수 또는 VALUES INTO 문에 의해 지정되어야 하며, 계속해서 FREE LOCATOR문에 의해 해제되지 않아야 합니다. 그렇지 않으면 오류가 발생합니다.

FREE LOCATOR문에 호스트 변수를 두 개 이상 지정하고 로케이터 중 하나에서 오류가 발생하는 경우 어떠한 로케이터도 해제되지 못합니다.

예

사원표에 RESUME, HISTORY 및 PICTURE 열이 있고 열 값을 표시하기 위해서 프로그램에 로케이터를 설정했다고 가정합니다. COBOL에서 다음 명령으로 CLOB 로케이터 변수 LOCRES와 LOCHIST, 그리고 BLOB 로케이터 변수 LOCPIC을 해제합니다.

```
EXEC SQL FREE LOCATOR :LOCRES, :LOCHIST, :LOCPIC END-EXEC.
```

GRANT(고유한 유형 권한)

이 형식의 GRANT문은 고유한 유형에 대한 권한을 부여합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

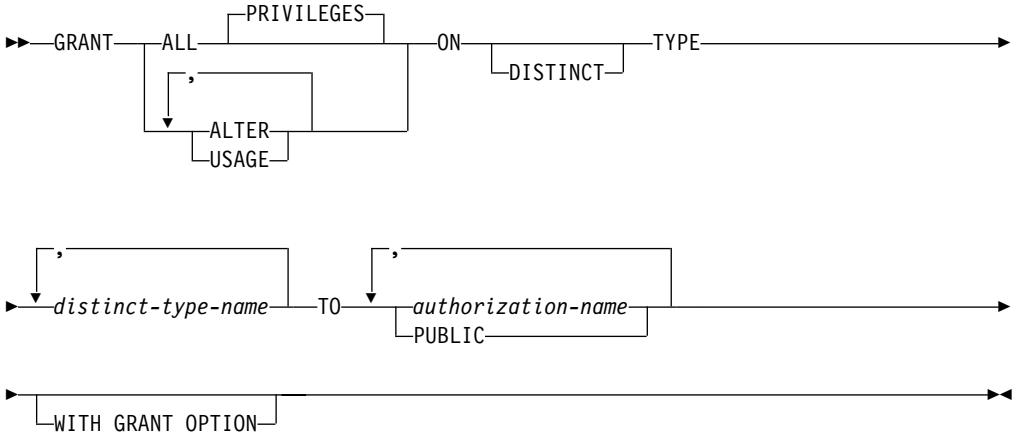
명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 각 고유한 유형의 경우
  - 명령문에 지정된 모든 권한
  - 고유한 유형에 대한 *OBJMGT 시스템 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

WITH GRANT OPTION을 지정하는 경우 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 포함되어 있어야 합니다.

- 고유한 유형의 소유권
- 관리 권한

구문



설명

ALL 또는 **ALL PRIVILEGES**

하나 이상의 권한을 부여합니다. 부여되는 권한은 명령문의 권한부여 ID가 지정된 고유한 유형에 대해 갖는 모든 부여가능한 권한입니다. 고유한 유형에 대해 ALL PRIVILEGES를 부여하는 것은 *ALL의 시스템 권한을 부여하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드는 설명된 권한을 부여합니다.

ALTER

COMMENT문을 사용할 권한을 부여합니다.

USAGE

표, 함수, 프로시듀어에 고유한 유형을 사용할 권한을 부여합니다.

ON DISTINCT TYPE *distinct-type-name*

권한이 부여되고 있는 고유한 유형을 식별합니다. *distinct-type-name*은 현재 서버에 고유한 유형을 식별해야 합니다.

TO

권한을 부여받는 사용자를 지정합니다.

authorization-name,...

하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC

사용자 집합(권한부여 ID)에 권한을 부여합니다.

사용자 집합은 고유한 유형에 대한 권한을 개인적으로 부여받지 않는 사용자로 구성됩니다. 예를 들어, ALTER가 PUBLIC에 부여되었고 USAGE가 HERNANDEZ에 부여되는 경우 이 개인 전용 부여로 인해 HERNANDEZ는 ALTER 권한을 갖지 못합니다.

WITH GRANT OPTION

지정한 *authorization-name*은 ON절에 지정된 고유한 유형에 대한 권한을 다른 사용자에게 부여할 수 있습니다.

WITH GRANT OPTION을 생략하는 경우 지정한 *authorization-name*이 일부 다른 소스로부터(예를들어, 시스템 권한 *OBJMGT로부터) 해당 권한을 받지 않는 한, ON절에 지정된 고유한 유형에 대한 권한을 다른 사용자에게 부여할 수 없습니다.

주

GRANT 및 REVOKE문은 SQL 오브젝트에 대한 시스템 권한을 지정하고 제거합니다. 다음 표에서는 SQL 권한에 해당하는 시스템 권한을 설명합니다.

표 51. 고유한 유형에 부여되거나 그로부터 취소되는 권한

SQL 권한	고유한 유형에 부여하거나 그로부터 취소할 때의 해당 시스템 권한
ALL(ALL의 부여 또는 취소는 명령문의 권한부여 ID가 갖는 권한만을 부여하거나 취소함)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT(취소 전용)
ALTER	*OBJALTER
USAGE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- 키워드 DATA는 DISTINCT의 동의어로 사용될 수 있습니다.

예

고유한 유형 SHOE_SIZE에 대한 USAGE 권한을 JONES에게 부여합니다. 이 GRANT 문은 고유한 유형 SHOE_SIZE와 연관된 캐스트 함수를 실행하는 권한은 JONES에게 제공하지 않습니다.

```
GRANT USAGE
  ON DISTINCT TYPE SHOE_SIZE
  TO JONES
```

GRANT(함수 또는 프로시듀어 권한)

이러한 형식의 GRANT문은 함수 또는 프로시듀어에 대한 권한을 부여합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

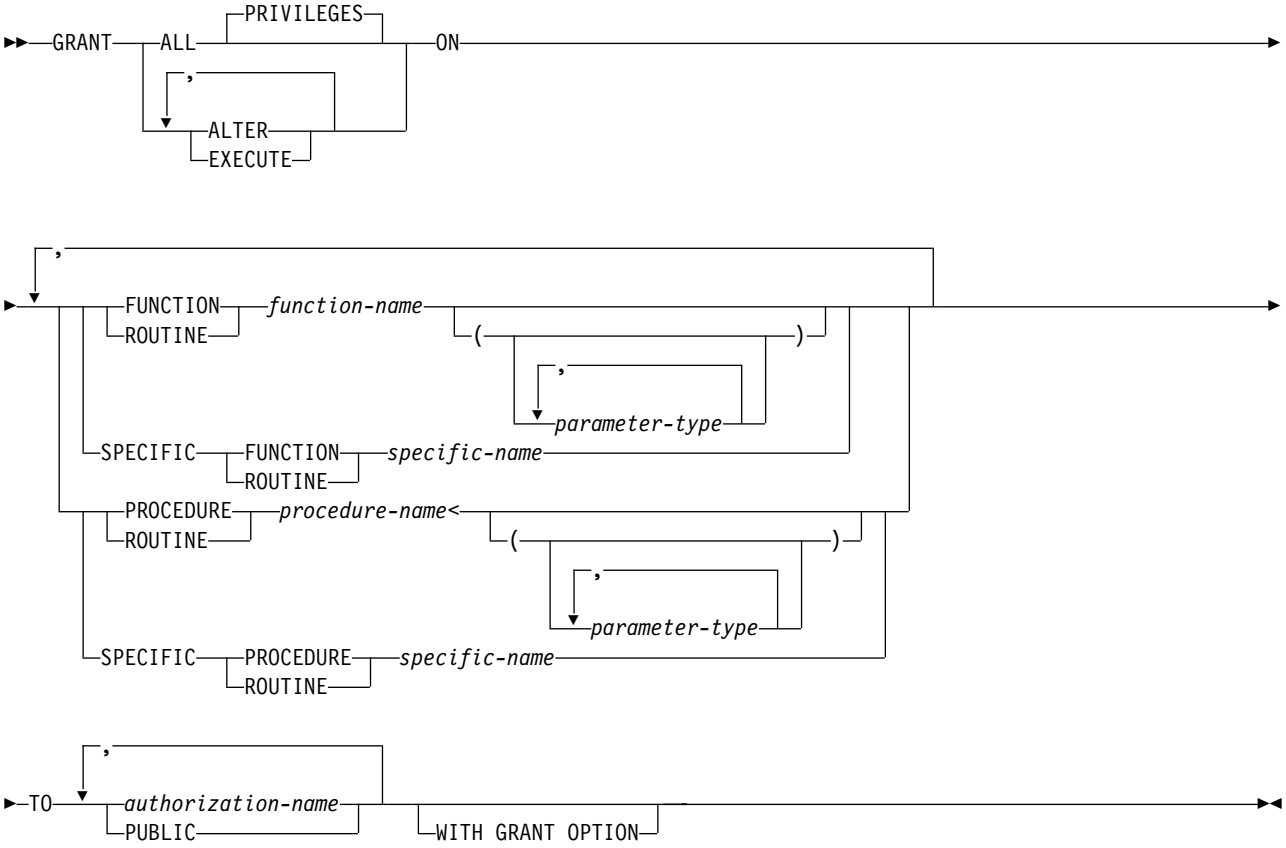
- 명령문에 식별된 각 함수나 프로시듀어에 대해 다음과 같은 권한을 갖습니다.
  - 명령문에 지정된 모든 권한
  - 함수나 프로시듀어에 대한 *OBJMGT 시스템 권한
  - 함수나 프로시듀어가 들어 있는 라이브러리(또는 Java 루틴인 경우 디렉토리)에 대한 *EXECUTE 시스템 권한
- 관리 권한

WITH GRANT OPTION을 지정하는 경우 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 포함되어 있어야 합니다.

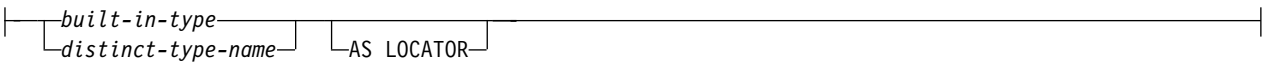
- 함수나 프로시듀어에 대한 소유권
- 관리 권한

GRANT(함수 또는 프로시듀어 권한)

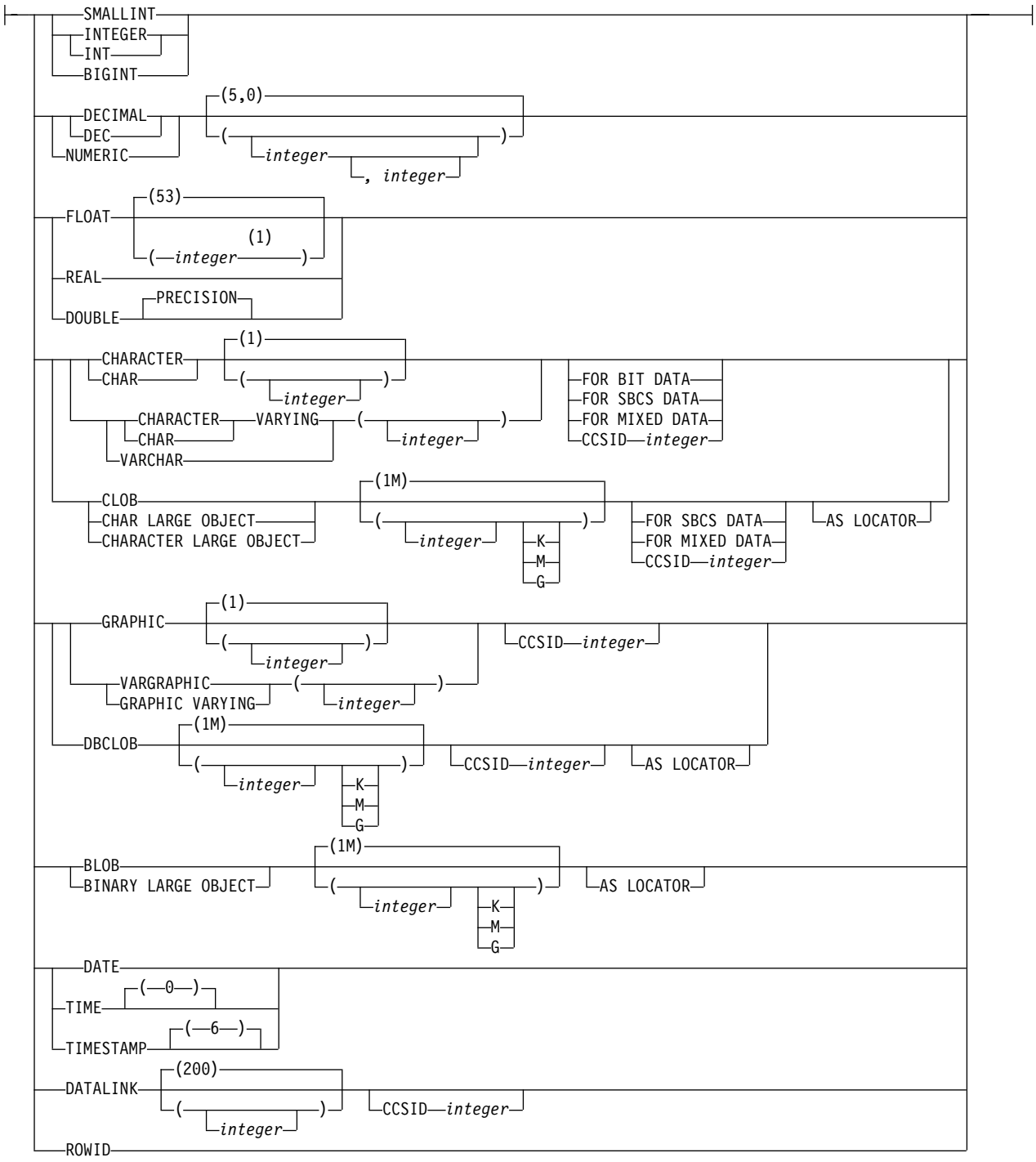
구문



parameter-type:



built-in-type:



주:

1 자료 유형(REAL 또는 DOUBLE)을 기초로 일치가 이루어지므로 정밀도에 대해 지정한 값이 함수를 작성할 때 지정한 값과 일치하지 않아도 됩니다.

설명

ALL 또는 ALL PRIVILEGES

하나 이상의 권한을 부여합니다. 부여되는 권한은 명령문의 권한부여 ID가 지정된 함수나 프로시듀어에 대해 갖는 모든 부여가능한 권한입니다. 함수나 프로시듀어에 대해 ALL PRIVILEGES를 부여하는 것은 *ALL의 시스템 권한을 부여하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드는 설명된 권한을 부여합니다.

ALTER

COMMENT문을 사용할 권한을 부여합니다.

EXECUTE

함수나 프로시듀어를 실행할 권한을 부여합니다.

FUNCTION

권한이 부여되고 있는 함수를 식별합니다. 이름, 함수 표시나 특정 이름으로 특별한 함수를 식별할 수 있습니다. 함수 해결(및 경로)에 대한 규칙은 사용되지 않습니다.

FUNCTION *function-name*

*function-name*은 현재 서버에 있는 한 함수를 정확히 식별해야 합니다. 함수는 함수에 대해 정의된 매개변수를 가질 수 있습니다. 지정된 또는 내재된 스키마에 지정된 이름의 함수가 둘 이상 있으면 오류가 리턴됩니다.

FUNCTION *function-name(parameter-type, ...)*

function-name(parameter-type, ...) 현재 서버에 있는 지정된 함수 표시로 함수를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE FUNCTION 문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수, 자료 유형의 논리 연결이 부여될 특정 함수 인스턴스를 식별하는 데 사용됩니다. *function-name()* 이 지정되면 식별된 함수는 0개의 매개변수를 가져야 합니다.

function-name

함수 이름을 식별합니다.

(parameter-type, ...)

함수의 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.

- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE FUNCTION문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

  ```
  CHAR      CHAR(1)
  GRAPHIC   GRAPHIC(1)
  DECIMAL   DECIMAL(5,0)
  FLOAT     DOUBLE (길이 8)
  ```

  내재된 길이는 CREATE FUNCTION문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE FUNCTION문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC FUNCTION *specific-name*

*specific-name*은 현재 서버에 있는 특정 함수를 식별해야 합니다.

PROCEDURE

권한이 부여되고 있는 프로시듀어를 식별합니다. 이름, 프로시듀어 표시 또는 특정 이름으로 특정 프로시듀어를 식별할 수 있습니다. 프로시듀어 해결(및 경로)에 대한 규칙은 사용되지 않습니다.

PROCEDURE *procedure-name*

*procedure-name*은 현재 서버에 있는 한 프로시듀어를 정확히 식별해야 합니다. 프로시듀어는 프로시듀어에 대해 정의된 매개변수를 갖습니다. 지정된 또는 내재된 스키마에 지정된 이름의 프로시듀어가 둘 이상 있는 경우 오류가 리턴됩니다.

PROCEDURE *procedure-name(parameter-type, ...)*

The *procedure-name(parameter-type, ...)*은 현재 서버에 있는 지정된 프로시듀어 표시로 프로시듀어를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE PROCEDURE문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수, 자료 유형의 논리 연결이 부여될 특정 프로시듀어 인스턴스를 식별하는 데 사용됩니다. *procedure-name()*이 지정되면 식별된 프로시듀어는 0개의 매개변수를 가져야 합니다.

procedure-name

프로시듀어 이름을 식별합니다.

(parameter-type, ...)

프로시듀어 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.
- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE PROCEDURE문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (길이 8)
```

내재된 길이는 CREATE PROCEDURE문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE PROCEDURE문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC PROCEDURE *specific-name*

*specific-name*은 현재 서버에 있는 특정 프로시듀어를 식별해야 합니다.

TO

권한을 부여받는 사용자를 지정합니다.

authorization-name,...

하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC

사용자 집합(권한부여 ID)에 권한을 부여합니다.

사용자 집합은 함수나 프로시듀어에 대한 권한을 개인적으로 부여받지 않는 사용자로 구성됩니다. 예를 들어, ALTER가 PUBLIC에 부여되었고 EXECUTE가 HERNANDEZ에 부여되는 경우 이 개인 전용 부여로 인해 HERNANDEZ는 ALTER 권한을 갖지 못합니다.

WITH GRANT OPTION

지정한 *authorization-name*에서 ON절에 지정된 함수나 프로시듀어에 대한 권한을 다른 사용자에게 부여할 수 있습니다.

WITH GRANT OPTION을 생략하는 경우 지정한 *authorization-name*이 일부 다른 소스로부터(예를 들어, 시스템 권한 *OBJMGT로부터) 해당 권한을 받지 않은 한, ON절에 지정된 함수나 프로시듀어에 대한 권한을 다른 사용자에게 부여할 수 없습니다.

주

SQL 또는 외부 함수나 프로시듀어에 부여된 권한은 관련 프로그램(*PGM) 또는 서비스 프로그램(*SRVPGM) 오브젝트에도 부여됩니다. Java 외부 함수나 프로시듀어에 부여된 권한은 연관된 클래스 파일이나 jar 파일에 부여됩니다.

GRANT 및 REVOKE문은 SQL 오브젝트에 대한 시스템 권한을 지정하고 제거합니다. 다음 표에서는 SQL 권한에 해당하는 시스템 권한을 설명합니다.

표 52. Java가 아닌 함수나 프로시듀어로 부여되고 그로부터 취소되는 권한

SQL 권한	**함수나 프로시듀어에 부여하거나 그로부터 취소할 때의 해당 시스템 권한**
ALL(ALL의 부여 또는 취소는 명령문의 권한부여 ID가 갖는 권한만을 부여하거나 취소함)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT(취소 전용)
ALTER	*OBJALTER
EXECUTE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

표 53. Java 함수나 프로시듀어에 부여되거나 그로부터 취소되는 권한

SQL 권한	**Java 함수나 프로시듀어에 부여하거나 그로부터 취소할 때의 해당 시스템 권한**	**Java 함수나 프로시듀어에 부여하거나 그로부터 취소할 때의 해당 오브젝트 권한**
ALL(ALL의 부여 또는 취소는 명령문의 권한부여 ID가 갖는 권한만을 부여하거나 취소함)	*RWX	*OBJEXIST *OBJALTER *OBJMGT(취소 전용)
ALTER	*R	*OBJALTER
EXECUTE	*RX	*EXECUTE
WITH GRANT OPTION	*RWX	*OBJMGT

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- RUN 키워드를 EXECUTE의 동의어로 사용할 수 있습니다.

예

CORPDATA.PROCA 프로시듀어에 대한 EXECUTE 권한을 PUBLIC에 부여합니다.

```
GRANT EXECUTE
     ON PROCEDURE CORPDATA.PROCA
   TO PUBLIC
```

GRANT(패키지 권한)

이 형식의 GRANT문은 패키지에 대한 권한을 부여합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

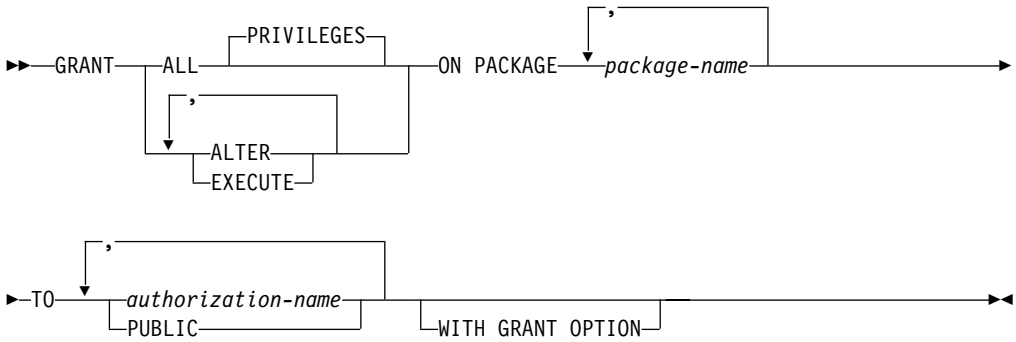
명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에 식별된 각 패키지에 대해 다음과 같은 권한을 갖습니다.
  - 명령문에 지정된 모든 권한
  - 패키지에 대한 *OBJMGT 시스템 권한
  - 패키지가 들어 있는 라이브러리에 대한 *EXECUTE 시스템 권한
- 관리 권한

WITH GRANT OPTION을 지정하는 경우 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 포함되어 있어야 합니다.

- 패키지의 소유권
- 관리 권한

구문



설명

ALL 또는 **ALL PRIVILEGES**

하나 이상의 권한을 부여합니다. 부여되는 권한은 명령문의 권한부여 ID가 지정된

패키지에 대해 갖는 모든 부여가능한 권한입니다. 패키지에 대해 ALL PRIVILEGES를 부여하는 것은 *ALL의 시스템 권한을 부여하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드는 설명된 권한을 부여합니다.

ALTER
COMMENT 및 LABEL문을 사용할 권한을 부여합니다.

EXECUTE
패키지에서 명령문을 실행할 권한을 부여합니다.

ON PACKAGE *package-name*
권한이 부여되고 있는 패키지를 식별합니다. *package-name*은 현재 서버에 있는 패키지를 식별해야 합니다.

TO
권한을 부여받는 사용자를 지정합니다.

authorization-name,...
하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC
사용자 집합(권한부여 ID)에 권한을 부여합니다.

사용자 집합은 사용자 패키지에 대한 권한을 개인적으로 부여받지 않는 사용자로 구성됩니다. 예를 들어, ALTER가 PUBLIC에 부여되었고 EXECUTE가 HERNANDEZ에 부여되는 경우 이 개인 전용 부여로 인해 HERNANDEZ는 ALTER 권한을 갖지 못합니다.

WITH GRANT OPTION
지정한 *authorization-name*은 ON절에 지정된 패키지에 대한 권한을 다른 사용자에게 부여할 수 있습니다.

WITH GRANT OPTION을 생략하는 경우 지정한 *authorization-name*이 일부 다른 소스로부터(예를 들어, 시스템 권한 *OBJMGT로부터) 해당 권한을 받지 않은 한, ON절에 지정된 패키지에 대한 권한을 다른 사용자에게 부여할 수 없습니다.

주

GRANT 및 REVOKE문은 SQL 오브젝트에 대한 시스템 권한을 지정하고 제거합니다. 다음 표에서는 SQL 권한에 해당하는 시스템 권한을 설명합니다.

표 54. 패키지에 부여되거나 패키지로부터 취소되는 권한

SQL 권한	패키지에 부여하거나 그로부터 취소할 때의 해당 시스템 권한
ALL(ALL의 부여 또는 취소는 명령문의 권한부여 ID가 갖는 권한만을 부여하거나 취소함)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT(취소 전용)
ALTER	*OBJALTER
EXECUTE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- RUN 키워드를 EXECUTE의 동의어로 사용할 수 있습니다.
- 키워드 PROGRAM은 PACKAGE와 동의어로 사용될 수 있습니다.

예

CORPDATA.PKGA 패키지에 대한 EXECUTE 권한을 PUBLIC에 부여합니다.

```
GRANT EXECUTE
     ON PACKAGE CORPDATA.PKGA
   TO PUBLIC
```

GRANT(표 권한)

이 형식의 GRANT문은 표나 뷰에 대한 권한을 부여합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

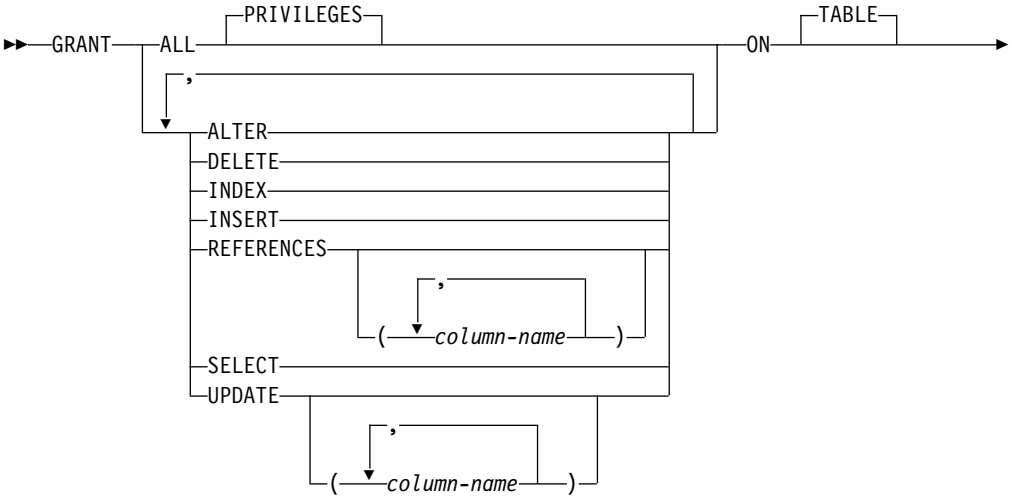
명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

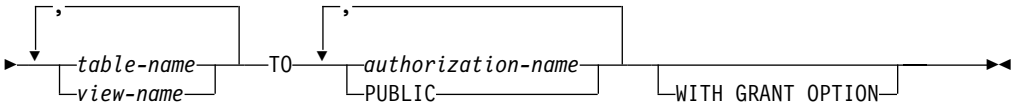
- 명령문에 식별된 각 표나 뷰에 대해 다음과 같은 권한을 갖습니다.
  - 명령문에 지정된 모든 권한
  - 표나 뷰에 대한 *OBJMGT 시스템 권한
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

WITH GRANT OPTION을 지정하는 경우 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 포함되어 있어야 합니다.

- 표의 소유권
- 관리 권한

구문





설명

ALL 또는 ALL PRIVILEGES

하나 이상의 권한을 부여합니다. 부여되는 권한은 명령문의 권한부여 ID가 지정된 표나 뷰에 대해 갖는 모든 부여가능한 권한입니다. 표나 뷰에 대해 ALL PRIVILEGES를 부여하는 것은 *ALL의 시스템 권한을 부여하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드가 설명된 권한을 부여하며 이것은 ON절에 이름이 지정된 표나 뷰에만 적용됩니다. 예를 들어, UPDATE, DELETE 및 INSERT 권한은 읽기 전용 뷰에 적용되지 않습니다.

ALTER

표에 대해 ALTER TABLE 및 DROP TRIGGER문을 사용할 권한을 부여합니다. 또한 표와 뷰에 대해 COMMENT 및 LABEL 문을 사용할 권한을 부여합니다.

DELETE

DELETE문을 사용할 권한을 부여합니다. 읽기 전용 뷰에는 DELETE를 부여할 수 없습니다.

INDEX

CREATE INDEX문을 사용할 권한을 부여합니다. 뷰에는 이 권한을 부여할 수 없습니다.

INSERT

INSERT문을 사용할 권한을 부여합니다. 삽입이 허용되지 않는 뷰에는 INSERT를 부여할 수 없습니다.

REFERENCES

해당 표가 상위가 되는 참조 제한조건을 추가할 권한을 부여합니다. 열 리스트를 지정하지 않거나 ALL PRIVILEGES를 지정하여 표나 뷰의 모든 열에 대해 REFERENCES를 부여하는 경우 부여를 받은 사용자는 ON절에 지정된 각 표의 모든 열을 사용하여 상위 키로서 참조 제한조건을 추가할 수 있습니다. ALTER TABLE문을 통해 나중에 추가할 수도 있습니다. 이 권한을 뷰에 부여할 수는 있지만 뷰에 대해 사용할 수는 없습니다.

REFERENCES(*column-name*,...)

열 리스트에 지정된 열에 대해서만 상위 키로서 참조 제한조건을 추가하는 권한을

부여합니다. 각 *column-name*은 규정화되지 않은 이름으로서 ON절에 지정된 각 표의 열을 식별해야 합니다. 이 권한을 뷰의 열에 부여할 수 있지만 뷰에 대해 사용하지는 않습니다.

SELECT

SELECT문 또는 CREATE VIEW문을 사용할 권한을 부여합니다.

UPDATE

UPDATE문을 사용할 권한을 부여합니다. 열 리스트를 지정하지 않거나 ALL PRIVILEGES를 지정하여 표나 뷰의 모든 열에 대해 UPDATE를 부여하는 경우 부여를 받은 사용자는 ON절에 지정된 각 표에서 갱신가능한 모든 열을 갱신할 수 있습니다. ALTER TABLE문을 통해 나중에 추가할 수도 있습니다. 갱신이 허용되지 않는 뷰에는 UPDATE를 부여할 수 없습니다.

UPDATE(*column-name***,...)**

열 리스트에 식별되어 있는 열만을 갱신하도록 UPDATE문을 사용할 권한을 부여합니다. 각 *column-name*은 규정화되지 않은 이름으로서 ON절에 지정된 각 표와 뷰의 열을 식별해야 합니다. 갱신이 허용되지 않는 열에는 UPDATE를 부여할 수 없습니다.

ON *table-name* 또는 *view-name*,...

권한이 부여되고 있는 표나 뷰를 식별합니다. table-name 또는 view-name은 현재 서버에 있는 표나 뷰를 식별하지만 글로벌 임시 표를 식별해서는 안됩니다.

TO

권한을 부여받는 사용자를 지정합니다.

authorization-name,...

하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC

사용자 집합(권한부여 ID)에 권한을 부여합니다.

사용자 집합은 표나 뷰에 대한 권한을 개인적으로 부여받지 않는 사용자로 구성됩니다. 예를 들어, SELECT가 PUBLIC에 부여되었고 UPDATE가 HERNANDEZ에 부여되는 경우 이 개인 전용 부여로 인해 HERNANDEZ는 SELECT 권한을 갖지 못합니다.

WITH GRANT OPTION

지정한 *authorization-name*은 ON절에 지정된 표와 뷰에 대한 권한을 다른 사용자에게 부여할 수 있습니다.

WITH GRANT OPTION을 생략하는 경우 지정한 *authorization-name*이 일부 다른 소스로부터(예를 들어, 시스템 권한 *OBJMGT로부터) 해당 권한을 받지 않은 한, ON절에 지정된 표와 뷰에 대한 권한을 부여할 수 없습니다.

주

GRANT문과 REVOKE문은 SQL 오브젝트에 대한 시스템 권한을 지정하고 제거합니다. 다음 표는 SQL 권한을 표에 부여할 때 그 권한에 해당하는 시스템 권한을 설명합니다. 왼쪽 열은 SQL 권한을 나열합니다. 오른쪽 열에서는 부여되거나 취소되는 해당 시스템 권한을 나열합니다.

표 55. 표에 부여되거나 표로부터 취소되는 권한

SQL 권한	**표에 부여하거나 그로부터 취소할 때의 해당 시스템 권한**
ALL(ALL의 GRANT 또는 취소는 명령문의 권한 부여 ID가 갖는 권한만을 부여하거나 취소함)	*OBJALTER[57] *OBJMGT(취소 전용) *OBJOPR *OBJREF *ADD *DLT *READ*UPD
ALTER	*OBJALTER[58]
DELETE	*OBJOPR *DLT
INDEX	*OBJALTER[58]
INSERT	*OBJOPR *ADD
REFERENCES	*OBJREF[58]
SELECT	*OBJOPR *READ
UPDATE	*OBJOPR *UPD
WITH GRANT OPTION	*OBJMGT

다음 표는 뷰에 SQL 권한을 부여할 때 그 권한에 해당하는 시스템 권한을 설명합니다. 왼쪽 열은 SQL 권한을 나열합니다. 중간 열은 뷰 자체에 부여되거나 취소되는 동등한 시스템 권한을 나열합니다. 오른쪽 열은 뷰 정의에 참조되는 모든 표와 뷰에 부여되는 시스템 권한을 나열하며, 뷰가 참조되는 경우 그 정의에 참조되는 모든 표와 뷰에 부여되고 취소됩니다.[59]

뷰가 둘 이상의 표나 뷰를 참조하는 경우 *DLT, *ADD 및 *UPD 시스템 권한은 뷰 정의의 subselect에 있는 첫 번째 표나 뷰에만 부여됩니다. *READ 시스템 권한은 뷰 정의에 참조된 모든 표와 뷰에 부여됩니다.

---

57. SQL INDEX 및 ALTER 권한은 동일한 시스템 권한 *OBJALTER에 해당합니다. INDEX와 ALTER 권한을 모두 부여해도 사용자에게 추가의 권한이 제공되지 않습니다.

58. WITH GRANT OPTION이 제공되는 경우 사용자는 ALTER 및 REFERENCES 권한에 의해 주어지는 함수 또한 수행할 수 있습니다.

59. 권리를 부여받을 사용자가 예를 들어 공용 권한과 같은 다른 권한 소스로부터의 권한을 아직 못 갖는 경우 뷰 정의에 참조되는 표와 뷰에만 지정한 권리가 부여됩니다.

SQL 권한을 사용하여 시스템 권한을 두 개 이상 부여하며 그 권한 중 하나가 부여될 수 없는 경우 경고가 발생하고 해당 권한에 대해 어떠한 권한도 부여되지 않습니다. GRANT와 달리 REVOKE는 뷰에 대한 시스템 권한을 취소하기만 합니다. 참조된 표와 뷰로부터 취소되는 시스템 권한은 없습니다.

표 56. 뷰에 부여되거나 뷰로부터 취소되는 권한

SQL 권한	뷰에 부여되거나 뷰로부터 취소되는 해당 시스템 권한	참조된 표와 뷰에 부여되거나 그로부터 취소되는 해당 시스템 권한
ALL(ALL의 GRANT 또는 REVOKE는 명령문의 권한부여 ID가 갖는 권한만을 부여하거나 취소함)	*OBJALTER *OBJMGT(취소 전용) *OBJOPR *OBJREF *ADD *DLT *READ*UPD	*ADD *DLT *READ*UPD
ALTER	*OBJALTER [60]	없음
DELETE	*OBJOPR *DLT	*DLT
INDEX	적용 안됨	적용 안됨
INSERT	*OBJOPR *ADD	*ADD
REFERENCES	*OBJREF [60]	없음
SELECT	*OBJOPR *READ	*READ
UPDATE	*OBJOPR *UPD	*UPD
WITH GRANT OPTION	*OBJMGT	없음

예

예 1

사용자에게 권한이 있다는 가정하에(KATHLEEN 스키마의) 표 WESTERN_CR에 대한 사용자의 모든 권한을 PUBLIC에 부여합니다.

```
GRANT ALL ON KATHLEEN.WESTERN_CR
TO PUBLIC
```

예 2

CALENDAR 표에 적절한 권한을 부여하여 ROANNA와 EMMA가 표를 읽거나 표에 새로운 항목을 삽입할 수 있도록 합니다. ROANNA와 EMMA가 기존의 항목을 변경하거나 제거하는 것은 허용하지 않습니다.

60. WITH GRANT OPTION이 제공되는 경우 사용자는 ALTER 및 REFERENCES 권한에 의해 주어지는 함수 또한 수행할 수 있습니다.

```
GRANT SELECT, INSERT ON CALENDAR
TO ROANNA, EMMA
```

예 3

TABLE1과 VIEW1에 대한 열 권한을 FRED에게 부여합니다. GRANT문에 지정하는 두 열 모두 TABLE1과 VIEW1 모두에 있는 것이어야 합니다.

```
GRANT UPDATE(column_1, column_2)
  ON  TABLE1, VIEW1
  TO FRED WITH GRANT OPTION
```

HOLD LOCATOR

HOLD LOCATOR 명령문은 LOB 로케이터 변수가 작업 단위를 넘어서는 값과 연관을 보유하도록 합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 대화식으로는 발행될 수 없습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다. 그러나 준비된 명령문을 실행하는 데는 USING절을 사용한 EXECUTE문을 사용해야 합니다. HOLD LOCATOR문은 EXECUTE IMMEDIATE문과 함께 사용할 수 없습니다.

권한부여

필요한 사항이 없습니다.

구문

설명

host-variable,...

호스트 변수 로케이터 변수의 선언 규칙에 따라서 선언해야 하는 호스트 변수를 식별합니다. 인디케이터 변수는 지정하지 않아야 합니다. 로케이터 변수 유형은 2진 큰 오브젝트 로케이터, 문자 큰 오브젝트 로케이터 또는 2바이트 문자 큰 오브젝트 로케이터여야 합니다.

HOLD LOCATOR 명령문이 실행된 후, 호스트 변수 리스트의 각 로케이터 변수는 보유 등록 정보를 가집니다.

호스트 변수에는 현재 지정된 로케이터가 있어야 합니다. 다시 말해, 로케이터가 이 작업 단위중 CALL, FETCH, SELECT INTO, SET 변수 또는 VALUES INTO 문에 의해 지정되어야 하며, 계속해서 FREE LOCATOR문에 의해 해제되지 않아야 합니다. 그렇지 않으면 오류가 발생합니다.

HOLD LOCATOR문에 호스트 변수를 두 개 이상 지정하고 로케이터 중 하나에서 오류가 발생하는 경우 어떠한 로케이터도 보유하지 못합니다.

주

보류 등록 정보를 가지는 호스트 변수 LOB 로케이터 변수는 다음과 같은 경우 지워집니다(값과의 연관이 지워집니다).

- SQL FREE LOCATOR문은 로케이터 변수로 실행됩니다.

- SQL ROLLBACK문은 실행됩니다.
- SQL 세션은 종료합니다.

예

사원표에 RESUME, HISTORY 및 PICTURE 열이 있고 열에 의해 표시된 값을 표시하기 위해서 프로그램에 로케이터를 설정했다고 가정합니다. 다음 명령으로 CLOB 로케이터 변수 LOCRES와 LOCHIST, 그리고 BLOB 로케이터 변수 LOCPIC 보유 등록 정보를 줍니다.

```
HOLD LOCATOR :LOCRES,:LOCHIST,:LOCPIC
```

INCLUDE

INCLUDE문은 선언이나 명령문을 소스 프로그램에 삽입합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. Java나 REXX에 지정되어서는 안됩니다.

권한부여

명령문의 권한부여 ID에는 멤버가 들어 있는 파일에 대한 시스템 권한 *OBJOPR 및 *READ가 있어야 합니다.

구문

설명

SQLCA

SQLCA(SQL 통신 영역)이 포함된 설명을 지정합니다. 한 프로그램에 INCLUDE SQLCA를 두 번 이상 지정할 수 없습니다. 프로그램이 독립형 SQLCODE 또는 독립형 SQLSTATE를 포함하는 경우 Include SQLCA를 지정하지 않아야 합니다.

SQLCA는 C, COBOL 및 PL/I에 대해 지정할 수 있습니다. SQLCA를 지정하지 않는 경우 변수 SQLCODE 또는 SQLSTATE가 프로그램에 있어야 합니다. 자세한 내용은 368 페이지의 『SQL 리턴 코드』를 참조하십시오.

SQLCA를 RPG 프로그램에 지정할 수 없습니다. RPG 프로그램에서는 사전컴파일러(pre-compiler)가 자동으로 SQLCA를 포함합니다.

SQLCA의 설명은 871 페이지의 부록 B 『SQL 통신 영역』을 참조하십시오.

SQLDA

SQLCA(SQL 설명자 영역)이 표함된 설명을 지정합니다. INCLUDE SQLDA는 C, COBOL, PL/I 및 ILE RPG/400에 지정할 수 있습니다.

SQLDA에 대한 설명은 881 페이지의 부록 C 『SQLDA(SQL 설명자 영역)』를 참조하십시오.

member-name

CRTSQL*xxx* 명령의 INCFILE 매개변수에 지정된 파일에서 포함시킬 멤버를 식별합니다.

멤버는 모든 호스트 언어 명령문, INCLUDE문을 제외한 모든 SQL문을 포함할 수 있습니다. COBOL에서 INCLUDE *member-name*을 DATA DIVISION 또는 PROCEDURE DIVISION 이외의 영역에 지정해서는 안됩니다.

프로그램이 사전컴파일될 때 INCLUDE문은 소스문으로 대체됩니다.

INCLUDE문은 사용자 프로그램에서 결과 소스문이 컴파일러에 허용되는 위치에 지정해야 합니다.

주

SRCFILE 매개변수에 지정한 소스 파일의 CCSID가 INCFILE 매개변수에 지정한 소스 파일의 CCSID와 다를 경우 INCLUDE문의 소스가 소스 파일의 CCSID로 변환됩니다.

예

C 프로그램에 SQL 통신 영역을 포함시킵니다.

```
EXEC SQL  INCLUDE SQLCA;
```

INSERT

INSERT문은 표나 뷰에 행을 삽입합니다. 뷰에 행을 삽입하면 삽입되는 뷰의 상위 표에도 행이 삽입됩니다.

이 명령문은 다음 세 가지 형식을 갖습니다.

- *VALUES를 사용한 INSERT* 형식은 제공되거나 참조된 값을 사용하여 표 또는 뷰에 하나의 행을 삽입하는 데 사용합니다.
- *SELECT를 사용한 INSERT* 형식은 다른 표나 뷰의 값을 사용하여 하나 이상의 행을 표나 뷰에 삽입하는 데 사용합니다.
- *n ROWS를 사용한 INSERT* 형식은 host-structure-array에 제공된 값을 사용하여 표나 뷰에 여러 개의 행을 삽입하는 데 사용합니다.

호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 n ROWS 형식에 예외가 있지만 동적으로 준비될 수 있는 실행문으로서 어플리케이션 프로그램에 삽입된 정적 명령문이어야 합니다. REXX 프로시듀어에서는 n ROWS 형식이 허용되지 않습니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표나 뷰의 경우
  - 표나 뷰에 대한 INSERT 권한
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 INSERT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에서 INSERT 권한을 부여받았습니다.
- 표에서 *OBJOPR과 *ADD의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID에 뷰에 대한 INSERT 권한이 있으면[61]

- 뷰에 대해 INSERT 권한을 받은 경우
- 뷰에 대해 *OBJOPR 및 *ADD 시스템 권한이 있고, 뷰 정의를 나타내는 첫 번째 FROM 구의 첫 번째 표나 뷰에 대해 *ADD 시스템 권한이 있으며, 이것이 뷰일

61. 뷰를 작성할 때 뷰에 대한 INSERT 권한이 없어도 됩니다. 뷰가 삽입을 허용하며 subselect에서 참조하는 첫 번째 표에 대해 INSERT 권한이 있으면 INSERT 권한만 받습니다.

때 그 뷰 정의의 첫 번째 FROM절에 있는 첫 번째 표나 뷰에 대해 *ADD 시스템 권한을 부여받는 형식으로 권한을 부여받은 경우

subselect를 지정하는 경우 명령문의 권한부여 ID에 의해 보유되는 권한에 다음 중 한 가지 이상의 권한이 포함되어 있어야 합니다.

- subselect에 식별된 각 표나 뷰에 대해 다음과 같은 권한을 갖습니다.
  - 표 또는 뷰에서의 SELECT 권한 및
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

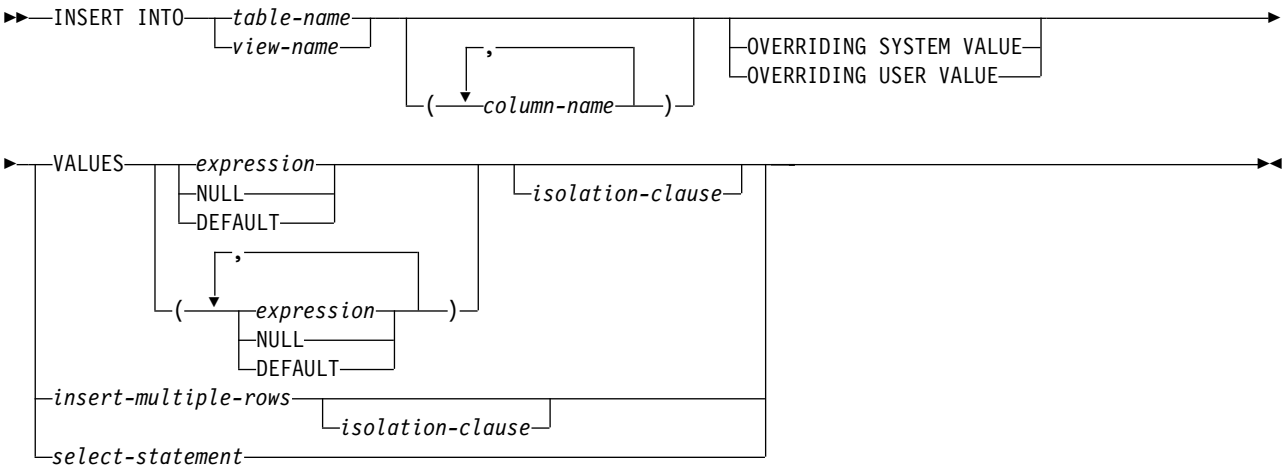
명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

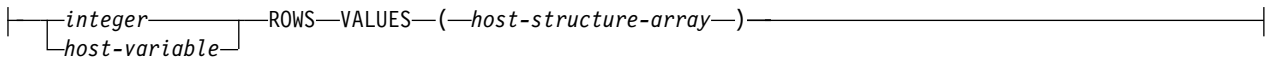
명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

구문



insert-multiple-rows:



isolation–clause:

설명

INTO *table-name* 또는 *view-name*

삽입 조작의 오브젝트를 식별합니다. 이름은 현재 서버에 있는 표나 뷰를 식별하지만 카탈로그 표, 카탈로그 표의 뷰 또는 읽기 전용 뷰를 식별해서는 안됩니다.

다음으로부터 파생된 뷰 열에는 값을 삽입할 수 없습니다.

- 상수, 표현식 또는 스칼라 함수
- 뷰의 일부 다른 열과 동일한 기본 표 열

삽입 조작의 오브젝트가 그러한 열을 갖는 뷰인 경우 열 이름 리스트를 지정해야 하며 그 리스트가 해당 열을 식별해서는 안됩니다.

(*column-name***,...)**

삽입 값이 제공되는 열을 지정합니다. 각 이름은 표나 뷰의 열을 식별하는 규정화

되지 않은 이름이어야 합니다. 같은 열이 한 번 이상 식별되어서는 안됩니다. 삽입 값을 허용하지 않는 뷰 열을 식별해서는 안됩니다.

열 리스트를 생략하면 표나 뷰의 모든 열이 왼쪽에서 오른쪽 순으로 식별되는 리스트가 암시적으로 지정됩니다. 이 리스트는 명령문이 준비될 때 작성되므로 명령문이 준비된 다음에 표에 추가된 열은 리스트에 포함되지 않습니다.

INSERT문이 어플리케이션에 삽입되고 참조된 표나 뷰가 프로그램을 작성할 때 존재하는 경우 이 명령문은 프로그램을 작성할 때 준비됩니다. 그 밖의 경우에는 INSERT문이 처음으로 성공적으로 실행될 때 준비됩니다.

OVERRIDING SYSTEM VALUE 또는 OVERRIDING USER VALUE

ROWID 또는 ID 열에 대해 시스템이 생성한 값이나 사용자 지정 값이 사용되는지 여부를 지정합니다. OVERRIDING SYSTEM VALUE가 지정된 경우, INSERT문의 내재적 또는 명시적 리스트에 GENERATED ALWAYS로 정의된 열이 들어 있어야 합니다. OVERRIDING USER VALUE가 지정된 경우, INSERT문의 내재적 또는 명시적 리스트에 GENERATED ALWAYS 또는 GENERATED BY DEFAULT로 정의된 열이 들어 있어야 합니다.

OVERRIDING SYSTEM VALUE

VALUES 절에 지정된 값 또는 GENERATED ALWAYS로 정의된 열에 대해 전체 선택으로 생성된 값이 사용되도록 지정합니다. 시스템이 생성한 값은 사용되지 않습니다.

OVERRIDING USER VALUE

VALUES 절에 지정된 값 또는 GENERATED ALWAYS 또는 GENERATED BY DEFAULT로 정의된 열에 대해 전체 선택으로 생성된 값이 무시되도록 지정합니다. 대신 시스템이 생성한 값이 삽입되어 사용자 지정 값을 대체합니다.

OVERRIDING SYSTEM VALUE 또는 OVERRIDING USER VALUE가 지정되지 않은 경우,

- ROWID 열이나 ID 열(GENERATED ALWAYS로 정의된 열)에는 값을 지정할 수 없습니다.
- ROWID 열이나 ID 열(GENERATED BY DEFAULT로 정의된 열)에는 값을 지정할 수 있습니다. 값이 지정된 경우, 해당 값이 열에 지정됩니다. 그러나 BY DEFAULT로 정의된 ROWID 열은 지정된 값이 이전에 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에 의해 생성된 유효 행 ID 값인 경우에만 삽입할 수 있습니다. BY DEFAULT로 정의된 ID 열에 값이 삽입될 때, 데이터베이스 관리자는 지정된 ID 열이 고유 제한조건이나 고유 색인의 고유 키가 아닌 경우 해당 값이 열의 고유 값임을 검증하지 않습니다. 고유 제한사항이나 고유 색인이 없는 경우, 데이터베이스 관리자는 NO CYCLE이 유효한 경우에 한해 시스템이 생성된 값 세트 사이에서만 고유 값을 보장할 수 있습니다.

  값이 지정된 데이터베이스 관리자가 아니면 새로운 값이 생성됩니다.

VALUES

값 리스트의 형식으로 새로운 행을 하나 지정합니다.

VALUES에 있는 값의 수는 열 리스트에 있는 이름의 수와 같아야 합니다. 첫 번째 값이 리스트의 첫 번째 열에 삽입되고, 두 번째 값은 두 번째 열에 삽입되는 방법으로 열에 값이 지정됩니다.

expression

열 값을 표현식으로부터 할당하도록 지정합니다. *expression*은 127 페이지의 『표현식』에서 설명한 유형의 모든 표현식입니다. 열 함수나 열 이름을 포함해서는 안됩니다.

*expression*이 단일 호스트 변수인 경우 호스트 변수는 구조를 식별할 수 있습니다. 이 절의 각 호스트 변수는 호스트 구조 및 변수의 선언 규칙에 따라 선언해야 하는 호스트 구조나 호스트 변수를 식별해야 합니다. 명령문의 조작 형식에서 호스트 구조에 대한 참조는 해당 변수 각각에 대한 참조로 대체됩니다. *host-variable*에 대한 설명은 제 2 장을 참조하십시오.

NULL

널값인 열의 값을 지정합니다. NULL은 널가능 열에 대해서만 지정해야 합니다.

DEFAULT

디폴트 값을 열에 지정합니다. 삽입되는 값은 다음과 같이 열이 정의된 방법에 따라 결정됩니다.

- WITH DEFAULT절을 사용하는 경우 삽입되는 디폴트는 열에 대해 정의한 것과 같습니다(541 페이지의 『CREATE TABLE』에 있는 *column-definition*에서 *default절* 참조).
- WITH DEFAULT절 또는 NOT NULL절을 사용하지 않는 경우에 삽입되는 값은 NULL입니다.
- NOT NULL절은 사용하고 WITH DEFAULT절을 사용하지 않거나 DEFAULT NULL을 사용하는 경우 해당 열에 대해 DEFAULT 키워드를 지정할 수 없습니다.
- 열이 ROWID 또는 ID 열이면, 데이터베이스 관리자은 새로운 값을 생성합니다.

GENERATED ALWAYS로 정의된 ROWID 또는 ID 열에 대해, OVERRIDING USER VALUE 값을 지정하여 사용자 지정 값이 무시되며 시스템이 생성한 고유 값이 삽입됨을 나타내지 않는 한 DEFAULT를 지정해야 합니다.

select-statement

선택문의 결과표 형식으로 새로운 행 집합을 지정합니다. INSERT와 함께 사용되는 선택문에서는 FOR READ ONLY, FOR UPDATE 및 OPTIMIZE절이 유효

하지 않습니다. ORDER BY절을 선택문에 지정하는 경우, 행은 ORDER BY절 식별된 열의 값에 따라 삽입됩니다. 선택문에 대한 설명은 352 페이지의『select문』을 참조하십시오.

선택문을 사용할 때 삽입되는 행이 하나 이상 있거나 삽입되는 행이 없을 수도 있습니다. 삽입되는 행이 없는 경우 SQLCODE는 +100으로 설정되고 SQLSTATE는 '02000'으로 설정됩니다.

INSERT의 기본 오브젝트와 선택문에 있는 subselect의 기본 오브젝트가 동일한 표인 경우 행이 삽입되기 전에 선택문이 완전히 평가됩니다.

결과표에 있는 열의 개수는 열 리스트에 있는 이름의 수와 같아야 합니다. 결과의 첫 번째 열 값이 리스트의 첫 번째 열에 삽입되고, 두 번째 값은 두 번째 열에 삽입되는 방법으로 열에 값이 삽입됩니다.

isolation-clause

INSERT문에 사용할 분리 레벨을 지정합니다. isolation절에 대한 설명은 358 페이지의『isolation절』을 참조하십시오.

insert-multiple-rows

integer 또는 *host-variable* **ROWS**

삽입할 행의 수를 지정합니다. 호스트 변수를 지정하는 경우 변수는 배율이 0인 숫자이어야 하며, 인디케이터 변수를 포함할 수 없습니다.

VALUES(*host-structure-array***)**

호스트 구조의 배열 형식으로 새로운 행 집합을 지정합니다. host-structure-array는 호스트 구조 배열의 선언 규칙에 따라 프로그램에 선언해야 합니다.

host-structure-array 이름 대신 매개변수 마커를 사용할 수 있습니다.

호스트 구조에 있는 변수의 수는 열 리스트에 있는 이름의 수와 같아야 합니다. 배열의 첫 번째 호스트 구조는 첫 번째 행에 대응되고, 배열의 두 번째 호스트 구조는 두 번째 행에 대응되는 방법으로 구조와 행이 대응됩니다. 또한 호스트 구조의 첫 번째 변수는 행의 첫 번째 열에 대응되고, 호스트 구조의 두 번째 변수는 행의 두 번째 열에 대응되는 방법으로 구조와 열이 대응됩니다.

호스트 구조의 배열에 대한 설명은 120 페이지의『C, C++, COBOL, PL/I 및 RPG의 호스트 구조 배열』을 참조하십시오.

LOB인 삽입 값이 있거나 현재 비iSeries 리모트 서버에 연결되어 있는 경우 복수 행 삽입은 허용되지 않습니다.

INSERT 규칙

디폴트 값

열 리스트에 없는 열에 삽입되는 값은 해당 열의 디폴트 값입니다. 따라서 디폴트 값이 없는 열은 열 리스트에 포함시켜야 합니다. 뷰의 경우에도 마찬가지로, 뷰에

포함되지 않는 기본 표의 열에는 디폴트 값이 삽입됩니다. 그러므로 뷰에 없는 기본 표의 모든 열은 디폴트 값을 갖고 있어야 합니다.

지정

삽입 값은 제 2 장에 설명된 지정 규칙에 따라서 열에 지정됩니다.

유효성

식별된 표 또는 식별된 뷰의 기본 표에 고유 색인 또는 고유 제한조건이 하나 이상 있는 경우 표에 삽입되는 각 행은 표의 색인에 의해 적용되는 제한조건을 따라야 합니다.

고유 색인 또는 고유 제한조건은 COMMIT(*NONE)를 지정하지 않은 경우에 명령문의 끝에서 검사됩니다. 복수 행 삽입의 경우 이 검사는 모든 행이 삽입되고 모든 관련 트리거가 실행된 다음에 발생합니다. COMMIT(*NONE)를 지정하면 각 행이 삽입될 때마다 검사가 수행됩니다.

식별된 표 또는 식별된 뷰의 기본 표에 하나 이상의 검사 제한조건이 있는 경우 표에 삽입되는 각 행에 대해 각 검사 제한조건이 참이거나 알 수 없는 상태이어야 합니다.

검사 제한조건은 명령문의 끝에서 검사됩니다. 복수 행 삽입의 경우 이 검사는 모든 행이 삽입된 다음에 발생합니다.

뷰가 식별되는 경우에는 삽입된 행이 모든 적용가능한 WITH CHECK OPTION을 따라야 합니다. 자세한 내용은 590 페이지의 『CREATE VIEW』를 참조하십시오.

트리거

식별된 표 또는 식별된 뷰의 기본 표에 삽입 트리거가 있는 경우 트리거는 활성화됩니다. 트리거는 다른 명령문이 실행되도록 하거나 삽입 값에 따라 오류 조건을 야기합니다.

참조 무결성

외부 키에 삽입되는 널(null)이 아닌 각 값은 관계에서 상위 표의 일부 상위 키 값과 같아야 합니다.

참조 제한조건(RESTRICT 삭제 규칙에 따른 참조 제한조건은 제외)은 명령문의 끝에서 검사됩니다. 복수 행 삽입의 경우 이 검사는 모든 행이 삽입되고 모든 관련 트리거가 실행된 다음에 발생합니다.

주

삽입 값이 참조 제한조건을 위배하거나 COMMNIT(*NONE)를 지정하지 않고 INSERT 문을 실행하는 동안 오류가 발생하면 명령문을 실행하는 동안 수행한 모든 변경 내용이 취소됩니다. 그러나, 오류 전에 작업 단위에서 발생한 다른 변경사항은 취소되지 않습니다. COMMIT(*NONE)가 지정되면 변경사항은 취소되지 않습니다.

INSERT문을 실행한 다음, SQLCA의 SQLERRD(3) 값은 데이터베이스 관리자가 삽입한 행의 수입니다. SQLERRD(3)의 값은 트리거의 결과로 삽입된 삭제된 행 수를 포함하지 않습니다.

COMMIT(*RR), COMMIT(*ALL), COMMIT(*CS) 또는 COMMIT(*CHG)를 지정하는 경우 INSERT문이 성공적으로 실행되는 동안 하나 이상의 배타적 잠금이 예약됩니다. 확약 또는 롤백 조작으로 잠금이 해제될 때까지는 다음을 통해서만 삽입된 행에 액세스할 수 있습니다.

- 해당 삽입을 수행한 어플리케이션 프로세스
- 읽기 전용 커서, SELECT INTO문 또는 부속 조회를 통해 COMMIT(*NONE) 또는 COMMIT(*CHG)를 사용하는 다른 어플리케이션 프로세스

잠금으로 인해 다른 어플리케이션 프로세스가 표에 대해 조작을 수행할 수 없습니다. 잠금에 대한 자세한 내용은 COMMIT, ROLLBACK 및 LOCK TABLE문의 설명을 참조하십시오. 또한 24 페이지의 『분리 레벨』 및 데이터베이스 프로그래밍 책을 참조하십시오.

COMMIT(*RR), COMMIT(*ALL), COMMIT(*CS) 또는 COMMIT(*CHG)를 지정한 경우 하나의 INSERT문에서 최대 500,000,000개의 행을 삽입하거나 변경할수 있습니다. 변경된 행의 수에는 동일한 확약 정의 아래 트리거의 결과로서 삽입, 갱신 또는 삭제된 모든 행이 포함됩니다.

REXX 프로시듀어 내에서는 INSERT문에 호스트 변수를 사용할 수 없습니다. 그 대신, INSERT는 매개변수 마커를 사용하는 PREPARE 및 EXECUTE의 오브젝트이어야 합니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- 키워드 NONE은 NC에 대한 동의어로 사용될 수 있습니다.
- 키워드 CHG는 UR에 대한 동의어로 사용될 수 있습니다.
- 키워드 ALL은 RS에 대한 동의어로 사용될 수 있습니다.

예

예 1

DEPARTMENT 표에 다음 부서를 삽입합니다.

- 부서 번호(DEPTNO)는 ‘E31’
- 부서명(DEPTNAME)은 ‘ARCHITECTURE’
- 사원 번호 ‘00390’인 사원(MGRNO)이 관리
- (ADMRDEPT) 부서 ‘E01’로 보고

```
INSERT INTO DEPARTMENT
  VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

예 2

예 1에서와 같이 DEPARTMENT 표에 새로운 부서를 삽입하지만 관리자는 지정하지 않습니다.

```
INSERT INTO  DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('E31', 'ARCHITECTURE', 'E01')
```

예 3

EMPPROJACT 표와 동일한 열을 갖는 표 MA_EMPPROJACT를 작성합니다. 'MA'로 문자로 시작하는 프로젝트 번호(PROJNO)의 EMPPROJACT 표의 행과 함께 MA_EMPPROJACT를 로드합니다.

```
CREATE TABLE MA_EMPPROJACT
           LIKE EMPPROJACT
INSERT INTO MA_EMPPROJACT
  SELECT * FROM EMPPROJACT
    WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 4

C 프로그램 명령문을 사용하여 중추 프로젝트를 PROJECT 표에 추가합니다. 호스트 변수에서 프로젝트 번호(PROJNO), 프로젝트명(PROJNAME), 부서 번호(DEPTNO) 및 담당 사원(RESPEMP)을 구합니다. 현재 날짜를 프로젝트 시작 날짜(PRSTDATE)로 사용합니다. NULL 값을 표의 나머지 열에 지정합니다.

```
EXEC SQL  INSERT INTO PROJECT  (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
            VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);
```

예 5

PL/I 프로그램에서 블록 삽입을 사용하여 DEPARTMENT 표에 10 행을 추가합니다. 호스트 구조 배열 DEPT에는 삽입할 자료가 들어 있습니다.

```
DCL  1  DEPT(10),
        3  DEPT  CHAR(3),
        3  LASTNAME  CHAR(29)  VARYING,
        3  WORKDEPT  CHAR(6),
        3  JOB  CHAR(3);

EXEC SQL INSERT INTO DEPARTMENT 10  ROWS VALUES (:DEPT);
```

예 6

Read Uncommitted(UR, CHG) 옵션을 사용하여 새로운 프로젝트를 EMPPROJACT 표에 삽입합니다.

```
INSERT INTO EMPPROJACT
  VALUES ('000140', 'PL2100', 30)
  WITH CHG
```

LABEL

LABEL문은 표, 뷰, 별명, 패키지 또는 열에 대한 카탈로그 설명에 레이블을 추가하거나 대체합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에 식별된 표, 뷰, 별명 또는 패키지의 경우:
  - 표, 뷰, 별명 또는 패키지에 대한 ALTER 권한
  - 표, 뷰, 별명 또는 패키지에 대한 *EXECUTE 시스템 권한
- 관리 권한

다음과 같은 경우에 명령문의 권한부여 ID는 표, 뷰 또는 패키지에 대한 ALTER 권한을 갖습니다.

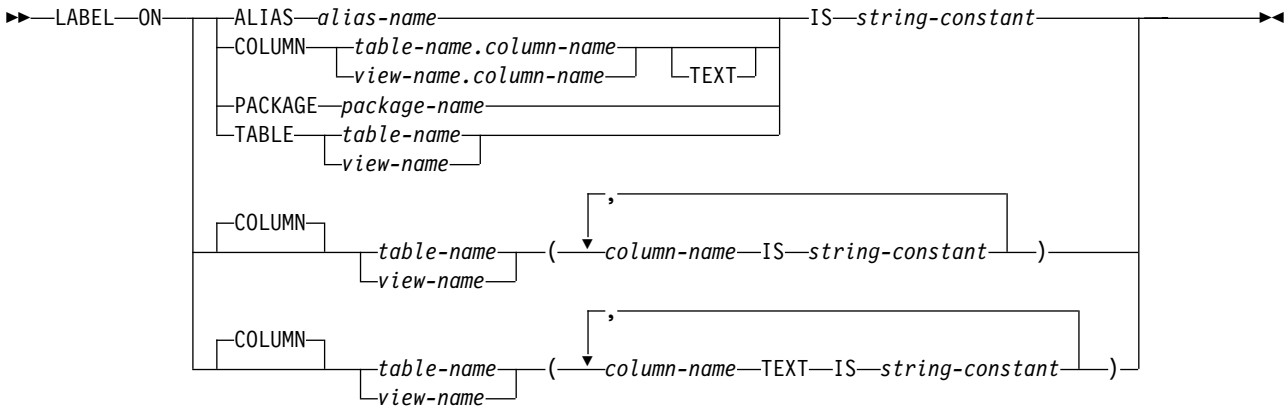
- 표, 뷰 또는 패키지의 소유자인 경우
- 표, 뷰 또는 패키지에 대해 ALTER 권한을 부여받은 경우
- 표, 뷰 또는 패키지에 대해 *OBJALTER 또는 *OBJMGT 중 하나의 시스템 권한을 부여받은 경우

다음과 같은 경우에 명령문의 권한부여 ID는 별명에 대한 ALTER 권한을 갖습니다.

- 해당 별명의 소유자인 경우
- 해당 별명에 대해 *OBJALTER 또는 *OBJMGT 중 하나의 시스템 권한을 부여받은 경우

LABEL

구문



설명

ALIAS

레이블이 별명의 레이블임을 지정합니다. 별명의 레이블은 오브젝트 텍스트로서 구현됩니다.

alias-name

레이블이 적용되는 별명을 식별합니다. 이름은 현재 서버에 있는 별명을 식별해야 합니다.

COLUMN

레이블이 열의 레이블임을 지정합니다. 열의 레이블은 시스템 열 머리말 또는 열 텍스트로서 구현됩니다. 열 머리말은 조회 결과를 표시하거나 인쇄할 때 사용합니다.

table-name.column-name 또는 *view-name.column-name*

레이블이 적용되는 열을 식별합니다. *table-name* 또는 *view-name*은 현재 서버에 있는 표나 뷰를 식별하지만 글로벌 임시 표를 식별해서는 안됩니다. *column-name*은 그 표나 뷰의 열을 식별해야 합니다.

TEXT

OS/400의 열 텍스트를 지정하도록 지정합니다. TEXT를 생략하면 열 머리말이 지정됩니다.

PACKAGE

패키지가 열의 레이블임을 지정합니다. 패키지의 레이블은 시스템 오브젝트 텍스트로서 구현됩니다.

package-name

레이블이 적용되는 패키지를 식별합니다. 이름은 현재 서버에 있는 패키지를 식별해야 합니다.

TABLE

레이블이 표나 뷰의 레이블임을 지정합니다. 표나 뷰의 레이블은 시스템 오브젝트 텍스트로서 구현됩니다.

table-name 또는 *view-name*

레이블을 추가할 표나 뷰를 식별합니다. *table-name* 또는 *view-name*은 현재 서버에 있는 표나 뷰를 식별하지만 글로벌 임시 표를 식별해서는 안됩니다.

IS

제공할 레이블을 소개합니다.

string-constant

표, 뷰, 별명, SQL 패키지 또는 열 텍스트의 경우 최대 50바이트 길이이거나 열 머리말의 경우에는 최대 60바이트 길이의 SQL문자 스트링 상수일 수 있습니다. 상수에는 1바이트와 2바이트 문자가 들어갈 수 있습니다.

열 머리말의 레이블은 세 개의 20바이트 세그먼트로 구성됩니다. 대화식 SQL, Query/400 프로그램, iSeries용 DB2 조회 관리자 및 SQL 개발 킷 및 기타 제품은 20바이트 세그먼트 각각을 개별 행에 표시하거나 인쇄할 수 있습니다. 열의 레이블에 혼합 자료가 있는 경우 20바이트 세그먼트 각각은 유효한 혼합 자료 문자 스트링이어야 합니다. 시프트(shift) 문자는 각 20바이트 세그먼트 안에서 쌍을 이루어야 합니다.

주

열 머리말은 조회 결과를 표시하거나 인쇄할 때 사용합니다. 첫 번째 열 머리말은 첫 번째 행에 표시되거나 인쇄되고, 두 번째 열 머리말은 두 번째 행에 그리고 세 번째 열 머리말은 세 번째 행에 표시되거나 인쇄됩니다. 열 머리말의 길이는 최대 60바이트이며, 그 중 처음 20바이트는 첫 번째 열 머리말이고 두 번째 20바이트는 두 번째 열 머리말, 그리고 세 번째 20바이트는 세 번째 열 머리말입니다. 20바이트 열 머리말 각각의 끝에서 공백이 제거됩니다.

모든 60바이트의 열 머리말 정보를 카탈로그 뷰 SYSCOLUMNS에서 사용할 수 있지만 첫 번째 열 머리말만이 DESCRIBE 또는 DESCRIBE TABLE문의 SQLDA에 리턴됩니다.

열 텍스트는 DESCRIBE 또는 DESCRIBE TABLE문에 리턴되지 않습니다. 데이터베이스 관리자가 공유되는 레코드 형식 설명에서 열 머리말 정보를 변경할 때 변경 내용은 해당 형식 설명을 공유하는 모든 파일에 반영됩니다. 파일이 다른 파일과 형식을 공유하는지 알아보려면 CL 명령 DSPDBR(데이터베이스 관계 표시)에 RCDFMT 매개변수를 사용하십시오.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 PROGRAM은 PACKAGE와 동의어로 사용될 수 있습니다.

예

- DEPARTMENT 표의 DEPTNO 열에 대한 레이블을 입력합니다.

```
LABEL ON COLUMNDEPARTMENT.DEPTNO
  IS 'DEPARTMENT NUMBER'
```

- 열 머리말이 두 행에 걸쳐 표시되는 DEPARTMENT 표의 DEPTNO 열에 대한 레이블을 입력합니다.

```
LABEL ON COLUMN DEPARTMENT.DEPTNO
  IS 'Department          Number'
```

- PAYROLL 패키지에 대한 레이블을 입력합니다.

```
LABEL ON PACKAGE CORPDATA.PAYROLL
  IS 'Payroll Package'
```

LOCK TABLE

LOCK TABLE문은 동시의 어플리케이션 프로세스에서 표를 변경하거나 사용하지 못하게 합니다.

호출

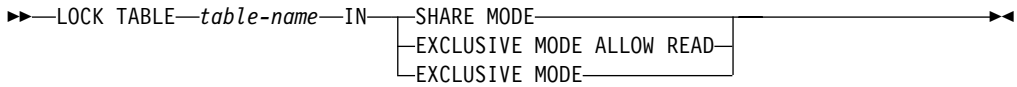
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표의 경우
  - 표에 대한 *OBJMGT 시스템 권한
  - 표가 들어 있는 라이브러리의 시스템 권한 *EXECUTE
- 관리 권한

구문



설명

table-name

잠글 표를 식별합니다. table-name은 현재 서버에 있는 기본 표를 식별하지만 카탈로그 표나 글로벌 임시표를 식별해서는 안됩니다.

IN SHARE MODE

동시의 어플리케이션 프로세스에서 표에 대해 읽기 전용 조작을 제외한 어떠한 조작도 실행하지 못하게 됩니다. 명령문이 실행되는 어플리케이션 프로세스에 대해 공유 잠금(*SHRNUP)을 예약합니다. 기타 어플리케이션 프로세스 또한 공유 잠금(*SHRNUP)을 예약할 수 있으며, 이 잠금으로 어플리케이션 프로세스는 읽기 전용 조작을 제외한 어떠한 조작도 실행할 수 없게 됩니다.

IN EXCLUSIVE MODE ALLOW READ

동시의 어플리케이션 프로세스에서 표에 대해 읽기 전용 조작을 제외한 어떠한 조작도 실행하지 못하게 됩니다. 명령문이 실행되는 어플리케이션 프로세스에 대해 배타적 허용읽기 잠금(*EXCLRD)을 예약합니다. 기타 어플리케이션 프로세스는 공유 잠금(*SHRNUP)을 예약하지 못하며, 그로 인해 어플리케이션 프로세스에서 표에 대한 갱신, 삭제 또는 삽입을 실행하는 것을 막을 수 없습니다.

LOCK TABLE

IN EXCLUSIVE MODE

동시의 어플리케이션 프로세스에서 표에 대해 어떠한 조작도 실행하지 못합니다.명령문이 실행되는 어플리케이션 프로세스에 대해 배타적 잠금(*EXCL)을 예약합니다.

이 잠금은 LOCK TABLE문을 실행할 때 예약됩니다.

잠금은 다음과 같은 때 해제됩니다.

- 작업 단위가 종료될 때 작업 단위가 COMMIT HOLD 또는 ROLLBACK HOLD에 의해 종료될 때는 제외
- 프로그램 스택의 첫 번째 SQL 프로그램이 종료될 때 CRTSQLxxx 명령에 CLOSQLCSR(*ENDJOB) 또는 CLOSQLCSR(*ENDACTGRP)을 지정한 경우는 제외
- 활성 그룹이 종료할 때
- CONNECT(유형 1)문을 사용하여 연결을 변경하는 때
- DISCONNECT문을 사용하여 잠금과 연관된 연결을 단절하는 때
- 연결이 릴리스 지연 중 상태에 있고 성공적인 COMMIT이 발생하는 때

DLCOBJ(오브젝트 할당해제) 명령을 사용하여 표의 잠금을 해제할 수도 있습니다.

명령문은 동기식이므로 다른 어플리케이션 프로세스에서 이미 보유한 상충하는 잠금이 있으면 사용자의 어플리케이션은 디폴트 대기 시간까지 대기하게 됩니다.

예

DEPARTMENT 표에 대한 잠금을 예약합니다. DEPARTMENT 표가 잠겨 있는 동안 다른 사용자가 표를 갱신하거나 읽을 수 없게 합니다.

```
LOCK TABLE DEPARTMENT IN EXCLUSIVE MODE
```

OPEN

OPEN문은 커서를 엽니다.

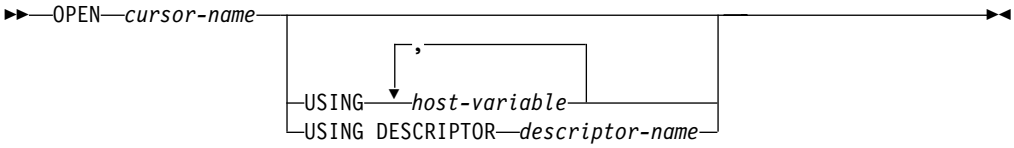
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

커서를 사용하는 데 필요한 권한부여에 대한 내용은 598 페이지의 『DECLARE CURSOR』를 참조하십시오.

구문



설명

cursor-name

열릴 커서를 식별합니다. *cursor-name*은 DECLARE CURSOR문의 주에 설명된 대로 선언된 커서를 식별해야 합니다. OPEN문을 실행할 때 커서는 닫힌 상태에 있어야 합니다.

커서와 연관된 SELECT문은 다음 두 중 하나입니다.

- DECLARE CURSOR문에 지정된 *select-statement* 또는
- DECLARE CURSOR문에 지정된 *select-name*으로 식별되는 준비된 *statement-name*. 명령문이 성공적으로 준비되지 않았거나 *select-statement*가 아닐 경우 커서를 성공적으로 열 수 없습니다.

커서의 결과표는 SELECT문의 평가에 의해 나옵니다. 평가에서는 SELECT문에 지정된 특수 레지스터의 현재 값과 SELELCT문에 지정된 호스트 변수의 현재 값을 사용하거나 OPEN문의 USING절을 사용합니다. 결과표의 행은 OPEN문의 실행 중 생성될 수 있고 그 행을 보유하기 위한 임시표가 작성될 수 있습니다. 또는 후속 FETCH문을 실행하는 동안 행이 생성될 수도 있습니다. 어느 경우에도 커서는 열린 상태로 되며 결과표의 첫 번째 행 앞에 놓여집니다. 표가 비어 있는 경우 커서의 위치는 “마지막 행 다음”입니다.

USING

호스트 변수의 값이 준비된 명령문의 매개변수 마커(의문 부호)를 대체하는 호스트 변수의 리스트를 소개합니다. 매개변수 마커에 대한 설명은 728 페이지의

『PREPARE』를 참조하십시오. DECLARE CURSOR문이 매개변수 마커를 포함하는 준비된 명령문을 명명하는 경우에는 USING을 사용해야 합니다. 준비된 명령문에 매개변수 마커가 포함되지 않은 경우 USING은 무시됩니다.

host-variable,...

호스트 구조 및 호스트 변수의 선언 규칙에 따라서 프로그램에 선언해야 하는 호스트 구조나 변수를 식별합니다. 호스트 구조에 대한 참조는 해당 변수 각각에 대한 참조로 대체됩니다. 변수의 수는 준비된 명령문에 있는 매개변수 마커의 수와 동일해야 합니다. *n* 번째 변수는 준비된 명령문에서 *n* 번째 매개변수 마커에 해당합니다.

DESCRIPTOR*descriptor-name*

호스트 변수의 유효한 설명을 포함해야 하는 SQLDA를 식별합니다.

OPEN문을 처리하기 전에 SQLDA에 다음과 같은 필드를 설정해야 합니다 (REXX에 대한 규칙은 다릅니다. 자세한 내용은 호스트 언어로 SQL 프로그래밍 책을 참조하십시오).

- SQLDA에 제공된 SQLVAR 발생 수를 표시하는 SQLN
- SQLDA에 대해 할당된 기억장치의 바이트 수를 표시하는 SQLDABC
- 명령문을 처리할 때 SQLDA에서 사용된 변수 수를 표시하는 SQLD
- 변수의 속성을 지정하는 SQLVAR 발생 수

모든 SQLVAR 발생을 포함할 수 있도록 SQLDA는 충분한 기억장치 공간을 가져야 합니다. LOB 또는 고유한 유형이 결과에 있는 경우 추가적인 SQLVAR 항목이 각 매개변수에 대하여 존재해야 합니다. SQLVAR에 대한 설명이 포함된 SQLDA에 대한 자세한 정보 및 SQLVAR 발생 횟수 판별 방법에 대한 설명은 881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』를 참조하십시오.

SQLD는 영(0) 보다 크거나 같고 SQLN 보다 작거나 같은 값으로 설정되어야 합니다. 또한 준비된 명령문에 있는 매개변수 마커의 수와 동일해야 합니다. SQLDA에 의해 설명되는 *n* 번째 변수는 준비된 명령문에서 *n* 번째 매개변수 마커에 해당합니다.

RPG/400은 포인터 설정을 위한 함수를 제공하지 않으며 SQLDA가 포인터를 사용하여 적절한 호스트 변수를 찾기 때문에 포인터를 RPG/400 어플리케이션 밖에 설정해야 합니다.

매개변수 마커 대체

커서에 대한 SELECT문이 평가될 때 명령문에 있는 각 매개변수 마커는 해당 호스트 변수로 대체됩니다. 매개변수 마커의 대체는 소스가 호스트 변수 값이고 목표가 데이터베이스 관리자 내의 변수인 할당 연산입니다. 유형 매개변수 마커의 경우 목표 변수의

속성은 CAST 스펙에서 지정한 것입니다. 비유형 매개변수 마커의 경우 목표 변수의 속성은 매개변수 마커의 문맥에 따라 결정됩니다. 매개변수 마커에 영향을 미치는 규칙은 732 페이지의 표 57을 참조하십시오.

매개변수 마커 P에 해당하는 호스트 변수를 V로 표시하십시오. 열에 값을 지정하는 데 적용되는 규칙에 따라서 값 V가 P에 대한 목표 변수에 지정됩니다. 그러므로 다음이 적용됩니다.

- V는 목표와 호환되어야 합니다.
- V가 숫자인 경우 정수부의 절대 값은 목표의 정수부의 최대 절대값 이하이어야 합니다.
- V의 속성이 목표의 속성과 동일하지 않을 경우 목표의 속성과 일치되도록 값이 변환됩니다.
- 목표에 널(null)이 포함될 수 없을 경우 값 V도 널이 아니어야 합니다.

그러나 열에 값을 지정하는 데 적용되는 규칙과 달리 다음이 적용됩니다.

- V가 스트링인 경우 그 길이가 목표의 길이 속성보다 크면 값이 절단됩니다(오류는 발생하지 않음).

커서에 대한 SELECT문이 평가될 때 P 대신 사용한 값은 P에 대한 목표 변수의 값입니다. 예를 들어, V가 CHAR(6)이고 목표는 CHAR(8)인 경우 P 대신 사용되는 값은 두 개의 공백이 채워진 V 값입니다.

USING절은 매개변수 마커가 들어 있는 준비된 SELECT문을 위한 절입니다. 그러나 커서에 대한 SELECT문이 DECLARE CURSOR문의 부분일 경우에도 이 절을 사용할 수 있습니다. 이 경우에 OPEN문은 SELECT문의 각 호스트 변수가 매개변수 마커인 것처럼 실행됩니다. 단, 목표 변수의 속성이 SELECT문에 있는 호스트 변수의 속성과 동일한 경우는 제외됩니다. 그 결과 커서에 대한 SELECT문에 있는 호스트 변수의 값이 USING절에 지정된 호스트 변수의 값으로 대체됩니다.

주

커서의 닫힌 상태

프로그램에 있는 모든 커서는 다음과 같은 경우에 닫힌 상태에 있습니다.

- 프로그램이 호출될 때:
  - CLOSQLCSR(*ENDPGM)이 지정된 경우 모든 커서는 프로그램이 호출될 때마다 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDSQL)이 지정된 경우 한 SQL 프로그램이 호출 스택에 남아 있는 한, 프로그램이 처음 호출될 때만 모든 커서가 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDJOB)을 지정하는 경우 작업이 활동중인 상태로 있는 한 프로그램이 처음으로 호출될 경우에만 모든 커서가 닫힌 상태에 있습니다.

  - CLOSQLCSR(*ENDMOD)가 지정된 경우 모듈이 개시될 때마다 모든 커서는 닫힌 상태에 있습니다.
  - CLOSQLCSR(*ENDACTGRP)을 지정하는 경우 프로그램의 모듈이 처음으로 활성 그룹에서 시작될 경우에만 모든 커서가 닫힌 상태에 있습니다.
- 프로그램은 HOLD 옵션없이 COMMIT나 ROLLBACK문을 실행하여 새로운 작업 단위를 시작합니다. HOLD 옵션으로 선언된 커서는 COMMIT문으로 닫히지 않습니다.
- CONNECT(유형 1)문을 실행하였을 때

커서는 다음과 같은 이유로도 닫힌 상태에 있을 수 있습니다.

- CLOSE문이 실행됨
- DISCONNECT문이 커서가 연관된 연결을 단절함
- 커서가 연관된 연결이 릴리스 지연 중 상태에 있었고 COMMIT가 성공적으로 실행됨.

커서의 결과표에서 행을 검색하기 위해 커서를 열어 놓을 때 FETCH문을 실행해야 합니다. 커서의 상태를 닫힌 상태에서 열린 상태로 변경하는 유일한 방법은 OPEN문을 실행하는 것입니다.

임시표의 영향

커서의 결과표가 읽기 전용이 아닌 경우 후속 FETCH문을 실행하는 동안 표에서 행이 파생됩니다. 동일한 메소드가 읽기 전용 결과표에서도 사용됩니다. 그러나 결과표가 읽기 전용일 때는 iSeries용 DB2 UDB이 대신 임시표 메소드의 사용을 선택할 수 있습니다. 이 메소드를 사용할 경우 OPEN문을 실행하는 동안 전체 결과표가 임시표에 삽입됩니다. 임시표를 사용하는 경우에 프로그램의 결과는 다음의 두 가지 면에서 서로 다를 수 있습니다.

- 나중에 FETCH문을 실행할 때까지 발생하지 않을 오류가 OPEN문을 실행하는 동안 발생할 수 있습니다.
- 커서가 열려 있는 동안 실행되는 INSERT, UPDATE 및 DELETE문은 결과표에 영향을 줄 수 없습니다.

반대로 임시표를 사용하지 않는 경우 커서가 열려 있는 동안 실행되는 INSERT, UPDATE 및 DELETE문은 결과표에 영향을 줄 수 있습니다. 이러한 조작의 영향을 항상 예측할 수 있는 것은 아닙니다. 예를 들어, 커서 C가 SELECT * FROM T로서 정의된 결과표의 한 행에 놓여지고 사용자가 한 행을 T에 삽입하는 경우에는 행이 순서대로 있지 않으므로 결과표에서 삽입의 영향을 예측할 수 없습니다. 후속 FETCH C는 T에서 새로운 행을 검색할 수도 있고 못할 수도 있습니다.

예

예 1

COBOL 프로그램으로 다음에 해당되는 삽입 명령문을 작성합니다.

1. (ADMRDEPT) 부서 'A00'에서 관리하는 부서의 DEPARTMENT 표에서 모든 행을 검색하는 데 사용할 커서 C1을 정의하십시오.
2. 커서 C1을 페치될 첫 번째 행 앞에 놓으십시오.

```
EXEC SQL  DECLARE C1 CURSOR FOR
           SELECT DEPTNO, DEPTNAME, MGRNO FROM DEPARTMENT
              WHERE ADMRDEPT = 'A00'  END-EXEC.

EXEC SQL  OPEN C1  END-EXEC.
```

예 2

커서 DYN_CURSOR을 C 프로그램의 동적으로 정의된 select문과 연관시키기 위한 OPEN문을 작성합니다. 준비된 각 선택문은 항상 선택 리스트에 두 항목을 정의하며, 그 중 첫 번째 항목은 integer의 자료 유형을 갖고 두 번째 항목은 VARCHAR(64)의 자료 유형을 갖는다고 가정합니다. (관련 호스트 변수 정의, PREPARE문 및 DECLARE CURSOR문 또한 아래 예에 나옵니다.)

```
EXEC SQL  BEGIN DECLARE SECTION;
  static short hv_int;
  char hv_vchar64[64];
  char stmt1_str[200];
EXEC SQL  END DECLARE SECTION;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

예 3

이 예에서는 OPEN문을 예 3에서와 같이 작성하되, 선택문에 있는 항목의 수와 자료 유형을 모릅니다.

```
EXEC SQL  BEGIN DECLARE SECTION;
  char stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

PREPARE

PREPARE문은 명령문의 문자 스트링 형식으로부터 실행가능한 형식의 SQL문을 작성합니다. 문자 스트링 형식은 *명령문 스트링*이라고 하며, 실행가능한 형식은 *준비된 명령문*이라고 합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

권한부여 규칙은 PREPARE문에 의해 지정되는 SQL문에 대해 정의된 규칙과 동일합니다. 예를 들어, SELECT문이 준비될 때 적용되는 권한부여 규칙에 대해서는 352 페이지의 『select문』을 참조하십시오.

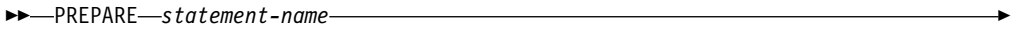
DLYPRP(*NO)를 CRTSQLxxx 명령에 지정하는 경우 명령문이 준비될 때 권한부여 검사가 수행됩니다. 단, 다음은 제외됩니다.

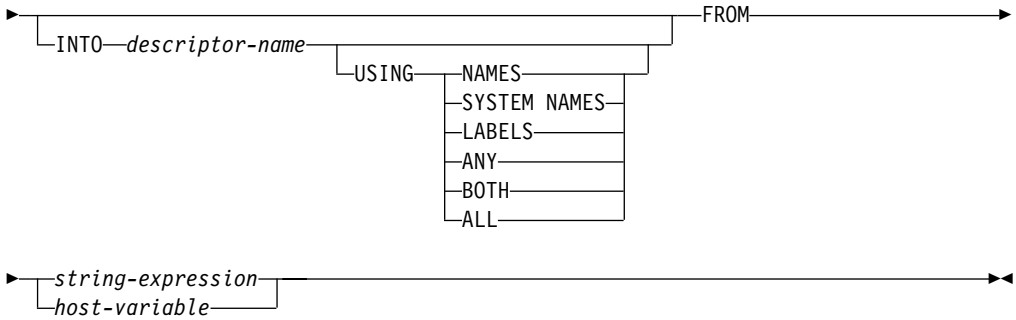
- DROP SCHEMA문이 준비되는 경우 스키마의 모든 오브젝트에 대한 시스템 권한 *OBJEXIST는 이 명령문이 실행될 때까지 검사되지 않습니다.
- DROP TABLE문이 준비되는 경우 표를 참조하는 모든 뷰, 색인 및 논리 파일에 대한 시스템 권한 *OBJEXIST는 이 명령문이 실행될 때까지 검사되지 않습니다.
- DROP VIEW문이 준비되는 경우 뷰를 참조하는 모든 뷰에 대한 시스템 권한 *OBJEXIST는 이 명령문이 실행될 때까지 검사되지 않습니다.

DLYPRP(*YES)를 CRTSQLxxx 명령에 지정하는 경우 모든 권한부여 검사는 명령문이 실행될 때 또는 OPEN문에 사용될 때까지 연기됩니다.

프로그램이 작성되었을 때 DYNUSRPRF(*OWNER)가 CRTSQLxxx 명령에 지정되지 않았으면, 명령문의 권한부여 ID는 실행 시간 권한부여 ID입니다. 자세한 내용은 58 페이지의 『권한부여 ID 및 권한부여명』을 참조하십시오.

구문





설명

statement-name

준비된 명령문의 이름을 지정합니다. 이름이 기존의 준비된 명령문을 식별할 경우 다음 상황에서는 준비된 명령문이 손상됩니다.

- 명령문이 동일한 프로그램의 동일한 인스턴스에서 준비된 경우
- CLOSQLCSR(*ENDJOB), CLOSQLCSR(*ENDACTGRP) 또는 CLOSQLCSR(*ENDSQL)이 두 개의 준비된 명령문 모두와 연관된 CRTSQLxxx 명령에 지정되는 경우

지정하는 이름이 프로그램의 해당 인스턴스에 대한 열린 커서의 SELECT문인 준비된 명령문을 식별해서는 안됩니다.

INTO

INTO를 사용하며 PREPARE문을 성공적으로 실행한 경우 준비된 명령문에 관한 정보가 descriptor-name에서 지정하는 SQLDA에 들어갑니다. 이 PREPARE문은 다음과 같습니다.

```
EXEC SQL PREPARE S1 INTO :SQLDA FROM :V1;
```

다음과 동일합니다.

```
EXEC SQL PREPARE S1 FROM :V1;
EXEC SQL DESCRIBE S1 INTO :SQLDA;
```

descriptor-name

881 페이지의 부록 C『SQLDA(SQL 설명자 영역)』에서 설명한 SQLDA를 식별합니다. PREPARE문이 실행되기 전에 SQLDA의 다음 변수가 설정되어야 합니다(REXX의 규칙은 다릅니다. 자세한 내용은 SQL Programming with Host Languages 책을 참조하십시오).

SQLN

SQLVAR이 나타내는 변수의 수를 지정합니다(SQLN은 SQLVAR 배열의 차원을 제공합니다). PREPARE문을 실행하기 전에 SQLN에 0 이상의 값을 설정해야 합니다. 발생 요구의 수를 판별하는 기술에 대한 내용은 884 페이지의『SQLVAR 필수 발생 수 판별』을 참조하십시오.

SQLDA에 들어가는 정보에 대한 설명은 645 페이지의 『DESCRIBE』를 참조하십시오.

USING

SQLDA의 각 SQLNAME 변수에 지정된 값을 지정합니다. 요구한 값이 없으면, SQLNAME은 길이 0으로 설정됩니다.

NAMES

열의 이름을 지정합니다. 이것이 디폴트 값입니다. 이름이 select-list에 명시적으로 지정되어 있는 준비된 명령문에서는 지정한 이름이 리턴됩니다.

SYSTEM NAMES

열의 시스템 열 이름을 지정합니다.

LABELS

열의 레이블을 지정합니다(열 레이블은 LABEL문으로 정의됩니다). 레이블의 처음 20바이트만 리턴됩니다.

ANY

열 레이블을 지정합니다. 열에 레이블이 없는 경우에 열의 레이블은 열명입니다.

BOTH

열의 레이블과 이름을 둘다 지정합니다. 이 경우 결과 세트가 고유한 유형에 포함되는지 여부에 따라, 열 당 SQLVAR의 두 번 또는 세 번의 발생이 추가 정보를 제공하기 위해 필요합니다. SQLVAR 배열의 확장을 지정하려면 SQLN을 2*n* 또는 3*n*으로 설정합니다(여기서, *n*은 표나 뷰의 열 수입니다). SQLVAR의 처음 *n*번 발생에는 열 이름이 포함됩니다. 두 번째 또는 세 번째 *n*번 발생에는 열 레이블이 포함됩니다. 고유한 유형이 없는 경우 레이블은 SQLVAR 항목의 두 번째 세트에 리턴됩니다. 고유한 유형이 있는 경우 레이블은 SQLVAR 항목의 세 번째 세트에 리턴됩니다.

동일한 SQLDA가 후속 FETCH문에 사용되는 경우 PREPARE가 완료된 다음에 SQLN을 *n*으로 설정하십시오.

ALL

레이블, 열 이름 및 시스템 열 이름을 지정합니다. 이 경우 결과 세트가 고유한 유형에 포함되는지 여부에 따라, 열 당 SQLVAR의 세 번 또는 네 번의 발생은 추가 정보를 제공하기 위해 필요합니다. SQLVAR 배열의 확장을 지정하기 위해 SQLN을 3*n* 또는 4*n*으로 설정합니다(여기서, *n*은 결과표의 열 수입니다). SQLVAR의 처음 *n*번 발생에는 시스템 열 이름이 포함됩니다. 두 번째 또는 세 번째 *n*번 발생에는 열 레이블이 포함됩니다. 세 번째 또는 네 번째 *n*번 발생에는 열 이름이 포함됩니다. 고유한 유형이 없는 경우 레이블은 SQLVAR 항목의 두 번째 세트에 리턴되고 열 이름은 SQLVAR의 세 번째 세트에 리턴

됩니다. 고유한 유형이 있는 경우 레이블은 SQLVAR 항목의 세 번째 세트에 리턴되고 열 이름은 SQLVAR의 네 번째 세트에 리턴됩니다.

동일한 SQLDA가 후속 FETCH문에 사용되는 경우 PREPARE가 완료된 다음에 SQLN을 *n*으로 설정하십시오.

FROM

명령문 스트링을 지정합니다. 명령문 스트링은 지정한 *string-expression* 또는 식별한 *host-variable*의 값입니다.

string-expression

*string-expression*은 문자 스트링을 생성하는 모든 PL/I *string-expression*입니다. 문자 스트링을 생성하는 SQL 표현식은 허용되지 않습니다. *string-expression*은 PL/I에서만 허용됩니다.

host-variable

문자 스트링 또는 UCS-2 그래픽 호스트 변수의 선언 규칙에 따라서 프로그램에 선언하는 호스트 구조를 식별합니다. 호스트 변수의 자료 유형은 CLOB나 DBCLOB여야 하고 인디케이터 변수는 지정하지 않아야 합니다.

명령문 스트링은 다음 SQL문 중 하나이어야 합니다.

ALTER	GRANT	SAVEPOINT
CALL	HOLD LOCATOR	select문
COMMENT	INSERT	SET PATH
COMMIT	LABEL	SET SCHEMA
CREATE	LOCK TABLE	SET TRANSACTION
DECLARE GLOBAL TEMPORARY TABLE	RELEASE SAVEPOINT	UPDATE
DELETE	RENAME	VALUES INTO
DROP	REVOKE	
FREE LOCATOR	ROLLBACK	

명령문 스트링에는 다음의 경우가 허용되지 않습니다.

- EXEC SQL로 시작하고 END-EXEC 또는 세미콜론(;)으로 끝나는 경우
- 호스트 변수에 대한 참조를 포함하는 경우

매개변수 마커

명령문 스트링에 호스트 변수에 대한 참조는 포함될 수 없지만, *매개변수 마커*는 포함될 수 있습니다. 매개변수 마커는 준비된 명령문이 실행될 때 호스트 변수의 값으로 대체됩니다. 매개변수 마커는 명령문 스트링이 정적 SQL문이었을 경우에 호스트 변수가 사용될 수 있는 위치에 사용됩니다. 매개변수 마커를 값으로 대체하는 방법에 대한 설명은 723 페이지의 『OPEN』 및 670 페이지의 『EXECUTE』를 참조하십시오.

매개변수 마커에는 두가지 유형이 있습니다.

유형 매개변수 마커

목표 자료 유형과 함께 지정된 매개변수 마커. 다음은 일반적인 형식입니다.

```
CAST(? AS data-type)
```

이 형식은 함수 호출이 아닌 실행시 매개변수 유형이 지정된 자료 유형 또는 지정된 자료 유형으로 변환될 수 있는 자료 유형이라는 "약속"입니다. 예를 들면 다음과 같습니다.

```
UPDATE EMPLOYEE
  SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
 WHERE EMPNO = ?
```

TRANSLATE 함수의 인수 값은 실행시 제공됩니다. 그 값의 자료 유형은 VARCHAR(12)이거나 VARCHAR(12)로 변환될 수 있는 유형입니다. 자세한 내용은 141 페이지의 『CAST 스펙』을 참조하십시오.

비유형 매개변수 마커

목표 자료 유형 없이 지정된 매개변수 마커. 물음표 하나를 사용하는 형식입니다. 비유형 매개변수 마커의 자료 유형은 문맥에 의해 지정됩니다. 예를 들어 위의 갱신 명령문 술부의 비유형 매개변수 마커는 EMPNO열의 자료 유형과 같습니다.

유형 매개변수 마커는 호스트 변수가 지원되고 자료 유형이 CAST 함수에서 이루어진 약속에 기초한다면 어디에서건 동적 SQL문에서 사용될 수 있습니다.

비유형 매개변수 마커는 호스트 변수가 지원되는 선택된 위치의 동적 SQL문에서 사용할 수 있습니다. 이 위치 및 결과 자료 유형은 표 57에서 찾을 수 있습니다. 이 표에서 위치는 표현식, 술부 및 함수로 그룹화되어 비유형 매개변수 마커의 적용성을 판별할 때 도움이 됩니다.

표 57. 비유형 매개변수 마커 사용법

비유형 매개변수 마커 위치	자료 유형
표현식(선택 리스트, CASE 및 VALUES 포함)	
부속 조회에 있지 않은 선택 리스트에 단독으로	오류
EXISTS 부속 조회에 있는 선택 리스트에 단독으로	오류
부속 조회에 있는 선택 리스트에 단독으로	부속 조회의 다른 피연산자의 자료 유형.[62]
INSERT문의 select문에 있는 선택 리스트에 단독으로	목표 표의 연관된 열의 자료 유형.[62]
연산자 우선 및 작동 규칙 순서를 고려한 후 단일 산술 연산자의 피연산자 모두 다음과 같은 경우가 포함됩니다. `? + ? + 10`	오류

표 57. 비유형 매개변수 마커 사용법 (계속)

비유형 매개변수 마커 위치	자료 유형
산술 표현식(날짜 시간 표현식이 아님)에서 단일 연산자의 단일 피연산자 다음과 같은 경우가 포함됩니다. `? + ? * 10`	다른 피연산자의 자료 유형.
날짜 시간 표현식에서 레이블된 기간(단위 유형을 나타내는 레이블된 기간의 일부는 매개변수 마커가 될 수 없음에 주의하십시오).	DECIMAL(15,0)
날짜 시간 표현식의 다른 피연산자(예를 들면 'timecol + ?' 또는 '? - datecol').	오류
CONCAT 연산자의 피연산자	오류
UPDATE문의 SET 절 오른쪽의 값으로	열의 자료 유형. 열이 사용자 정의 고유한 유형으로 정의되는 경우 사용자 정의 고유한 유형의 소스 자료 유형입니다.[62]
단순한 CASE 표현식의 CASE 키워드 다음에 나오는 표현식	오류
비유형 매개변수 마커이거나 널(null)인 나머지 결과 표현식을 갖는(Simple 및 Searched) CASE 표현식에서 최소 하나의 결과 표현식	오류
단순한 CASE 표현식에서 WHEN 다음에 나오는 임의의 또는 모든 표현식	CASE 다음의 표현식과 비유형 매개변수 마커인 WHEN 다음의 표현식에 93 페이지의 『결과 자료 유형에 대한 규칙』 적용 결과.
적어도 하나의 결과 표현식이 널이 아니고 비유형 매개변수 마커가 아닌 CASE 표현식(Simple 및 Searched)의 결과 표현식	널 또는 비유형 매개변수 마커가 아닌 모든 결과 표현식에 93 페이지의 『결과 자료 유형에 대한 규칙』 적용 결과
INSERT문에 없는 단일 행 VALUES절의 열 표현식으로 단독으로	오류
INSERT문에서 단일 행 VALUES절의 열 표현식으로 단독으로	열의 자료 유형. 열이 사용자 정의 고유한 유형으로 정의되는 경우 사용자 정의 고유한 유형의 소스 자료 유형입니다.[62]
SET 특수 레지스터 명령문의 오른쪽 값으로	특수 레지스터의 자료 유형.
VALUES INTO 명령문의 INTO절의 값으로	연관된 표현식의 자료 유형.[62]
FREE LOCATOR 또는 HOLD LOCATOR 명령문의 값으로서	로케스터
술부	
비교 연산자의 두 피연산자 모두	오류
한 피연산자가 비유형 매개변수 마커이거나 고유한 유형인 비교 연산자의 또다른 피연산자.	다른 피연산자의 자료 유형.[62]
다른 피연산자가 고유한 유형인 비교 연산자의 한 피연산자.	오류
BETWEEN 술부의 모든 피연산자	오류
BETWEEN 술부의 두 피연산자(첫 번째와 두 번째 또는 첫 번째와 세 번째)	유일한 비 매개변수 마커의 것과 같음.

표 57. 비유형 매개변수 마커 사용법 (계속)

비유형 매개변수 마커 위치	자료 유형
BETWEEN 술부의 유일한 피연산자	CCSID 속성이 실행시 지정된 값의 CCSID라는 점을 제외하고 비유형 매개변수 마커를 제외한 모든 피연산자에 대한 93 페이지의 『결과 자료 유형에 대한 규칙』 적용 결과.
IN 술부의 모든 피연산자, 예를 들면 ? IN (?,?,?)	오류
오른쪽이 부속 선택인 IN 술부의 첫 번째 피연산자. 예를 들면 ? IN (subselect).	선택된 열의 자료 유형
오른쪽이 부속 선택이 아닌 IN 술부의 첫 번째 피연산자. 예를 들면 ? IN (?,A,B) 또는 ? IN (A,?,B,?).	CCSID 속성이 실행시 지정된 값의 CCSID라는 점을 제외하고 비유형 매개변수 마커를 제외한 IN 리스트의 모든 피연산자(IN 키워드 오른쪽의 피연산자)에 대한 93 페이지의 『결과 자료 유형에 대한 규칙』 적용 결과.
IN 술부의 IN 리스트의 임의의 또는 모든 피연산자. 예를 들면 A IN (?,B,?).	CCSID 속성이 실행시 지정된 값의 CCSID라는 점을 제외하고 비유형 매개변수 마커를 제외한 IN 술부의 모든 피연산자(IN 술부 왼쪽과 오른쪽의 피연산자)에 대한 93 페이지의 『결과 자료 유형에 대한 규칙』 적용 결과.
LIKE 술부의 세 가지 피연산자 모두	오류
LIKE 술부의 일치 표현식.	오류
LIKE 술부의 패턴 표현식.	일치 표현식의 자료 유형에 따라 VARCHAR(32740) 또는 VARGRAPHIC(16370) 또는 BLOB(32740) 패턴 값에 대한 고정 길이 호스트 변수 사용에 대한 정보는 150 페이지의 『LIKE 술부』 페이지를 참조하십시오..
LIKE 술부의 이탈 표현식.	일치 표현식의 자료 유형에 따라 VARCHAR(1) 또는 VARGRAPHIC(1) 또는 BLOB(1)
널(null) 술부의 피연산자	오류
함수	
COALESCE, IFNULL, LAND, LOR, MIN, MAX, NULLIF, VALUE 또는 XOR의 모든 피연산자	오류
비유형 매개변수 마커 이외에 최소 하나의 피연산자가 있는 COALESCE, IFNULL, LAND, LOR, MIN, MAX, NULLIF, VALUE 또는 XOR 중 임의의 피연산자.	비유형 매개변수 마커를 제외한 모든 피연산자에 93 페이지의 『결과 자료 유형에 대한 규칙』을 적용한 결과
POSITION의 첫 번째 피연산자 또는 POSSTR의 두 번째 피연산자	다른 피연산자의 자료 유형에 따라 VARCHAR(32740) 또는 VARGRAPHIC(16370) 또는 BLOB(32740)
사용자 정의 함수를 포함하여 모든 다른 스칼라 함수의 다른 피연산자.	오류
열 함수의 피연산자	오류

주

오류 검사

PREPARE문을 실행할 때 명령문 스트링이 분석되고 오류 여부가 검사됩니다. 명령문 스트링이 유효하지 않으면 준비된 명령문이 작성되지 않고, 명령문의 작성을 막는 오류 조건이 SQLCA에 보고됩니다.

로컬 및 리모트 처리에서 DLYPREP(*YES) 옵션은 일부 SQL문에서 "지연됨" 오류를 받는 원인이 됩니다. 예를 들어 DESCRIBE, EXECUTE 및 OPEN은 보통 PREPARE 처리시 발생하는 SQLCODE를 받을 수 있습니다.

참조 및 실행 규칙

다음과 같은 제한사항의 적용으로, 준비된 명령문이 다음과 같은 종류의 명령문에 참조될 수 있습니다.

명령문	준비된 명령문 제한
DESCRIBE	None
DECLARE CURSOR	Must be SELECT when the cursor is opened
EXECUTE	Must not be SELECT

준비된 명령문을 여러 번 실행할 수 있습니다. 준비된 명령문이 두 번 이상 실행되지 않고 매개변수 마커를 포함하고 있지 않는 경우 PREPARE문과 EXECUTE문보다는 EXECUTE IMMEDIATE문을 사용하는 것이 더 효과적입니다.

준비된 명령문 지속성

모든 준비된 명령문은[63]

- CONNECT(유형 1)문을 실행할 때
- DISCONNECT문이 준비된 명령문과 관련되는 연결을 단절할 때
- 준비된 명령문이 릴리스 지연 중 연결과 관련되고 성공적인 확약이 발생할 때
- SQL문의 연관된 범위(작업, 활성 그룹 또는 프로그램)가 종료합니다.

명령문 범위

*statement-name*의 범위는 그 이름이 정의되어 있는 소스 프로그램입니다. 사용자는 PREPARE문으로 사전컴파일된 다른 SQL문에 의해 준비된 명령문만을 참조할 수 있습니다. 예를 들어, 별도로 컴파일된 다른 프로그램에서 호출된 프로그램은 호출하는 프로그램이 작성한 준비된 명령문을 사용할 수 없습니다.

---

62. 자료 유형이 DATE, TIME 또는 TIMESTAMP이면 VARCHAR(32766)가 사용됩니다.

63. 준비된 명령문이 캐쉬되어 실제 삭제되지 않을 때 삭제됩니다. 그러나 캐쉬된 명령문은 같은 명령문이 다시 준비될 경우에만 사용할 수 있습니다.

statement-name의 범위는 또한 해당 명령문이 있는 프로그램이 실행되고 있는 스레드로 제한됩니다. 예를 들어, 동일한 작업 내의 두 개의 개별 스레드에서 동일한 프로그램이 실행되고 있는 경우 두 번째 스레드는 첫 번째 스레드에서 준비한 명령문을 사용할 수 없습니다.

명령문의 범위는 명령문이 정의되어 있는 프로그램이지만, 프로그램으로부터 작성된 각 패키지에는 준비된 명령문의 개별 인스턴스가 포함되며 실행시 둘 이상의 준비된 명령문이 존재할 수 있습니다. 예를 들면, CONNECT(유형 2)문을 사용하여 프로그램이 위치 X와 위치 Y에 다음 순서로 연결된다고 가정합니다.

```
EXEC SQL CONNECT TO X;
   EXEC SQL PREPARE S FROM :hv1;
   EXEC SQL EXECUTE S;
   .
   .
   .
EXEC SQL CONNECT TO Y;
   EXEC SQL PREPARE S FROM :hv1;
   EXEC SQL EXECUTE S;
```

S의 두 번째 준비가 Y에서 또 하나의 인스턴스 S를 준비합니다.

CLOSQLCSR(*ENDJOB), CLOSQLCSR(*ENDACTGRP) 또는 CLOSQLCSR(*ENDSQL)을 CRTSQLxxx 명령에 지정하지 않을 경우 준비된 명령은 프로그램 스택에 있는 프로그램의 동일한 인스턴스에만 참조될 수 있습니다.

- CLOSQLCSR(*ENDJOB)을 지정하는 경우 준비된 명령문은 프로그램 스택에 있는 프로그램(명령문을 준비한 프로그램)의 모든 인스턴스에 참조될 수 있습니다. 이 경우에 준비된 명령문은 작업의 끝에서 손상됩니다.
- CLOSQLCSR(*ENDSQL)을 지정하는 경우 준비된 명령문은 프로그램 스택의 마지막 SQL 프로그램이 종료될 때까지 프로그램 스택에 있는 프로그램(명령문을 준비한 프로그램)의 모든 인스턴스에 참조될 수 있습니다. 이 경우에 준비된 명령문은 프로그램 스택의 마지막 SQL 프로그램이 종료될 때 손상됩니다.
- CLOSQLCSR(*ENDACTGRP)을 지정하는 경우 준비된 명령문은 활성 그룹이 종료될 때까지 명령문을 준비한 프로그램에 있는 모듈의 모든 인스턴스에 참조될 수 있습니다. 이 경우에 준비된 명령문은 활성 그룹이 종료될 때 손상됩니다.

예

예 1

SELECT가 아닌 명령문을 COBOL 프로그램에 준비하여 실행합니다. 호스트 변수 HOLDER에 명령문이 들어 있고, 프로그램이 명령문 스트링을 사용자의 일부 지시에 따라서 호스트 변수에 넣는다고 가정합니다. 준비되는 명령문에는 매개변수 마커가 없습니다.

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER  END-EXEC.

EXEC SQL  EXECUTE STMT_NAME  END-EXEC.
```

예 2

예 1에서와 같이 SELECT가 아닌 명령문을 준비하여 실행하되 준비 중인 명령문에 매개변수 마커가 포함될 경우는 제외합니다.

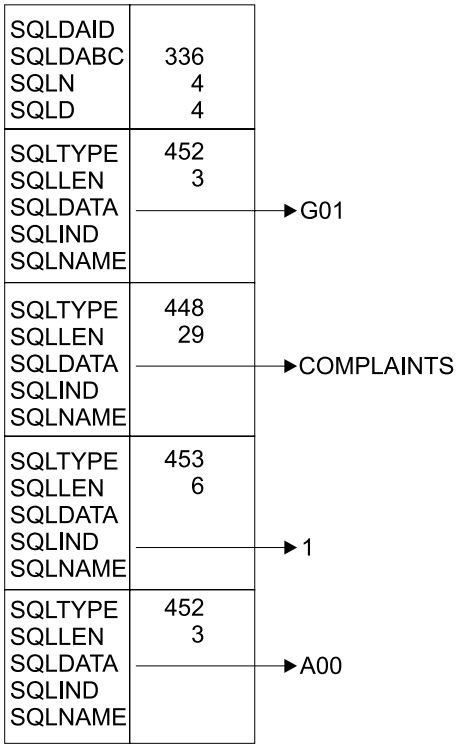
```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER  END-EXEC.

EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :INSERT_DA  END-EXEC.
```

다음과 같은 명령문이 준비된다고 가정합니다.

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

관리자가 없고 부서 A00으로 보고를 하는 COMPAINTS라는 이름의 부서 번호 G01을 삽입하려면 EXECUTE문을 실행하기 전에 구조 INSERT_DA는 다음과 같은 값을 가져야 합니다.



RBAL3501-0

RELEASE

RELEASE문은 하나 이상의 연결을 릴리스 지연 중 상태로 만듭니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. Java나 REXX에 지정되어서는 안됩니다.

RELEASE는 트리거 프로그램에 허용되지 않습니다. 외부 프로시듀어가 리모트 서버에 호출되는 경우에는 외부 프로시듀어에 RELEASE가 허용되지 않습니다.

권한부여

필요한 사항이 없습니다.

구문

설명

server-name 또는 *host-variable*

지정한 서버명 또는 호스트 변수에 있는 서버명으로 서버를 식별합니다. 호스트 변수가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 서버명은 호스트 변수 내에서 왼쪽부터 배열되어야 하고 일반 ID를 형성하는 규칙을 따라야 합니다.
- 서버명의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

RELEASE문을 실행할 때 지정한 서버명 또는 호스트 변수에 있는 서버명이 활성 그룹의 기존 연결을 식별해야 합니다.

CURRENT

활성 그룹의 현재 연결을 식별합니다. 활성 그룹이 연결된 상태에 있어야 합니다.

ALL 또는 **ALL SQL**

활성 그룹의 모든 기존의 연결(리모트 연결과 로컬 연결을 모두 포함)을 식별합니다.

명령문을 실행할 때 어떠한 연결도 없으면, 오류나 경고가 발생하지 않습니다.

RELEASE문이 성공적이면 식별된 각 연결이 릴리스 지연 중 상태에 놓이며, 따라서 다음 확약 조작 중 연결이 종료됩니다. RELEASE문이 성공적이지 않으면 활성 그룹의 연결 상태 및 각 연결의 상태가 변하지 않습니다.

주

CONNECT(유형 1) 구문을 사용하면 RELEASE를 사용하지 못합니다.

RELEASE가 커서를 닫지 않고, 어떠한 자원도 해제하지 않으며, 차후의 연결 사용을 억제하지 않습니다.

ROLLBACK은 연결의 상태를 릴리스 지연 중에서 보류 상태로 재설정하지 않습니다.

리모트 연결을 작성하고 유지보수하는 데는 자원이 필요합니다. 그러므로 다시 사용하지 않을 리모트 연결은 릴리스 지연 중 상태이어야 하며 다시 사용할 리모트 연결은 릴리스 지연 중 상태일 수 없습니다.

확약 조작이 수행될 때 현재 연결이 릴리스 지연 중 상태에 있으면, 그 연결이 종료되고 활성 그룹은 연결되지 않은 상태가 됩니다. 이때 그 다음에 실행되는 SQL문은 CONNECT나 SET CONNECTION이어야 합니다.

RELEASE ALL은 로컬 서버와의 연결이 릴리스 지연 중 상태가 됩니다. 연결에 WITH HOLD절로 정의된 열린 커서가 있어도 확약 조작 중 릴리스 지연 중 상태의 연결이 종료됩니다.

예

예 1: TOROLAB1과의 연결은 다음 작업 단위에 필요 없습니다. 다음 명령문은 다음 확약 조작 중 이러한 연결을 종료시킵니다.

```
EXEC SQL RELEASE TOROLAB1;
```

예 2: 현재 연결이 다음 작업 단위에 필요 없습니다. 다음 명령문은 다음 확약 조작 중 이러한 연결을 종료시킵니다.

```
EXEC SQL RELEASE CURRENT;
```

예 3: 기존의 모든 연결이 다음 작업 단위에 필요 없습니다. 다음 명령문은 다음 확약 조작 중 이러한 연결을 종료시킵니다.

```
EXEC SQL RELEASE ALL;
```

RELEASE SAVEPOINT

RELEASE SAVEPOINT 명령문은 지정된 저장점 및 작업 단위 내에 이어서 설정된 모든 저장점을 릴리스합니다.

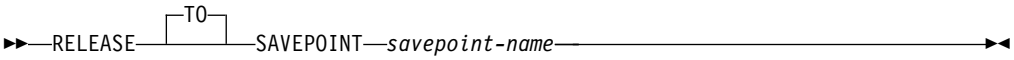
호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

필요한 사항이 없습니다.

구문



설명

savepoint-name

릴리스할 저장점을 식별합니다. 명명된 저장점이 없다면, 오류가 발생합니다. 명명된 저장점 또는 이어서 작업 단위에 설정된 모든 저장점이 릴리스됩니다. 저장점이 릴리스되고 나면, 더 이상 유지보수되지 않으며 더 이상 저장점으로 롤백할 수 없습니다.

주

해당 저장점명을 지정하는 이전의 SAVEPOINT문에 UNIQUE 키워드가 지정되었는지 여부에 상관 없이 릴리스된 저장점명은 다른 SAVEPOINT문에 다시 사용할 수 있습니다.

RELEASE SAVEPOINT 명령문은 활성 그룹에 대해 확약 제어가 작동하지 않는 경우 사용할 수 없습니다. 사용하는 확약 정의 판별에 대한 정보는 419 페이지의 『주』를 참조하십시오.

예

기본 루틴이 저장점 A를 설정하고 저장점 B 및 C를 설정하는 서브루틴을 호출하는 경우, 제어가 기본 루틴으로 돌아가면 저장점 A 및 기타 후속으로 설정된 저장점이 릴리스됩니다. 서브루틴에 의해 설정된 저장점 B 및 C가 A에 이어 릴리스되었습니다.

```
RELEASE SAVEPOINT A
```

RENAME

RENAME문은 표, 뷰 또는 색인을 재명명합니다. 표, 뷰 또는 색인의 이름과 시스템 오브젝트명은 변경할 수 있습니다.

호출

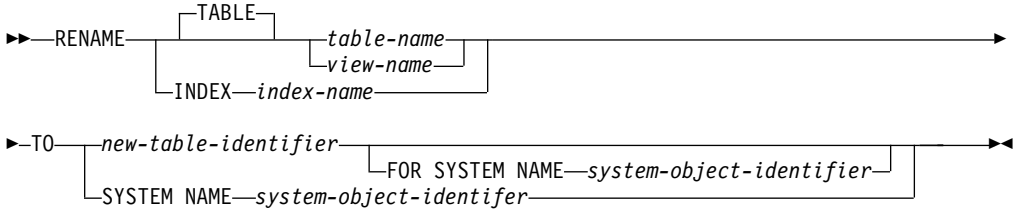
이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 다음과 같은 시스템 권한
  - 오브젝트명을 변경하는 경우:
    - 재명명할 표, 뷰 또는 색인에 대한 *OBJMGT 시스템 권한
    - 재명명할 표, 뷰 또는 색인이 들어 있는 라이브러리에 대한 *EXECUTE 시스템 권한
  - 오브젝트의 시스템명을 변경하는 경우:
    - 재명명할 표, 뷰 또는 색인에 대한 *OBJMGT 시스템 권한
    - 재명명할 표, 뷰 또는 색인이 들어 있는 라이브러리에 대한 *EXECUTE 및 *UPD 시스템 권한
- 관리 권한

구문



설명

TABLE *table-name* 또는 *view-name*

재명명할 표나 뷰를 식별합니다. *table-name* 또는 *view-name*은 현재 서버에 있는 표나 뷰를 식별하지만 카탈로그 표나 글로벌 임시 표를 식별해서는 안됩니다. 지정된 이름이 별명일 수도 있습니다. 지정한 표나 뷰가 새로운 이름으로 재명명됩니다. 표나 뷰에 대한 모든 권한, 제한조건, 색인, 트리거, 뷰 및 논리 파일은 보존됩니다.

액세스 계획을 사용하는 프로그램이 다음 실행 프로그램일 때 표나 뷰를 참조하는 모든 액세스 계획이 다시 암시적으로 준비됩니다. 프로그램은 원래의 이름으로 표나 뷰를 참조하므로 원래의 이름을 갖는 표나 뷰가 없으면 SQLCA의 SQLCODE 필드에 음의 값이 리턴됩니다.

INDEX *index-name*

재명명할 색인을 식별합니다. *index-name*은 현재 서버에 있는 색인을 식별해야 합니다. 지정한 색인이 새로운 이름으로 재명명됩니다.

해당 색인을 참조하는 모든 액세스 계획은 재명명의 영향을 받지 않습니다.

new-table-identifier

표, 뷰 또는 색인의 새로운 *table-name, view-name* 또는 *index-name*을 각각 식별합니다. *new-table-identifier*은 이미 현재 서버에 있는 표, 뷰, 별명 또는 색인과 같지 않아야 합니다. *new-table-identifier*은 규정화되지 않은 SQL ID이어야 합니다.

SYSTEM NAME *system-object-identifier*

표, 뷰 또는 색인의 새로운 *system-object-identifier*을 각각 식별합니다. *system-object-identifier*은 이미 현재 서버에 있는 표, 뷰, 별명 또는 색인과 같지 않아야 합니다. *system-object-identifier*은 규정화되지 않은 시스템 ID이어야 합니다.

오브젝트의 이름과 오브젝트의 시스템명이 같고 *name*을 지정하지 않은 경우 *system-object-identifier*을 지정하면 새로운 이름과 시스템 오브젝트명이 됩니다. 그러나 *system-object-identifier*을 지정하면 오브젝트의 시스템명에만 영향을 미치고 오브젝트명에는 영향을 미치지 않습니다.

*new-table-identifier*과 *system-object-identifier*이 모두 지정되면 둘 다 유효한 시스템 오브젝트명이 될 수 없습니다.

주

수행되는 재명명 조작은 지정한 새로운 이름에 따라 다릅니다.

- 새로운 이름이 유효한 시스템 ID일 경우:
  - 대체명이 있으면 제거되고,
  - 시스템 오브젝트명이 새 이름으로 변경됩니다.
- 새로운 이름이 유효하지 않은 시스템 ID일 경우:
  - 대체명이 추가되거나 새 이름으로 변경되고,
  - 표나 뷰의 시스템 오브젝트명을 재명명할 표, 뷰 또는 색인으로서 지정한 경우 새로운 시스템 오브젝트명이 생성됩니다. 생성된 표 이름 규칙에 대한 정보는 572 페이지의 『표 이름 생성 규칙』을 참조하십시오.

*table-name*의 별명을 지정하는 경우이고 별명이 현재 서버에 존재하면 별명으로 식별된 표가 재명명됩니다. 별명 이름은 변경되지 않으며 기존의 표를 재명명 후에 참조가 계속됩니다. 별명 변경은 지원되지 않습니다.

예

예 1

MY_IN_TRAY 표 이름을 MY_IN_TRAY_94로 변경합니다. 시스템 오브젝트명은 변경되지 않은 그대로입니다(MY_IN_TRAY).

```
RENAME TABLE MY_IN_TRAY TO MY_IN_TRAY_94
  FOR SYSTEM NAME MY_IN_TRAY
```

예 2

MA_PROJ 표 이름을 MA_PROJ_94로 변경합니다.

```
RENAME TABLE MA_PROJ
  TO SYSTEM NAME MA_PROJ_94
```

REVOKE(고유한 유형 권한)

이 형식의 REVOKE문은 고유한 유형에 대한 권한을 제거합니다.

호출

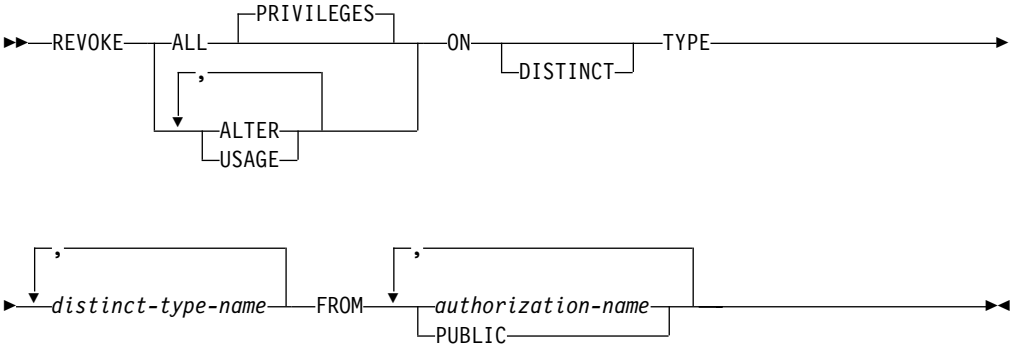
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 각 고유한 유형의 경우
  - 명령문에 지정된 모든 권한
  - 고유한 유형에 대한 *OBJMGT 시스템 권한
  - 고유한 유형이 들어 있는 라이브러리에서 시스템 권한 *EXECUTE
- 관리 권한

구문



설명

ALL 또는 **ALL PRIVILEGES**

각 *authorization-name*으로부터 하나 이상의 고유한 유형 권한을 취소합니다. 취소된 권한은 *authorization-names*에 부여했던 식별된 고유한 유형에 대한 권한입니다. 고유한 유형에 대한 ALL PRIVILEGES를 취소하는 것이 *ALL 시스템 권한을 취소하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드는 설명된 권한을 취소합니다.

ALTER

COMMENT문을 사용할 권한을 취소합니다.

USAGE

표, 함수, 프로시듀어에 고유한 유형을 사용하거나 CREATE DISTINCT TYPE문에 소스 유형으로서 사용자 정의 유형을 사용할 권한을 취소합니다.

ON DISTINCT TYPE *distinct-type-name*

사용자가 권한을 취소하고 있는 고유한 유형을 식별합니다. *distinct-type-name*은 현재 서버에 고유한 유형을 식별해야 합니다.

FROM

권한이 취소되는 사용자를 식별합니다.

authorization-name,...

하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC

지정한 권한을 PUBLIC으로부터 취소합니다.

주

고유한 유형에 대한 권한을 취소하면 권한 부여자에 관계없이 해당 고유한 유형에 대한 모든 권한 부여가 무효화됩니다.

고유한 유형 권한이 취소될 때 그에 해당하는 시스템 권한도 취소됩니다. SQL 권한에 해당하는 시스템 권한에 대한 내용은 684 페이지의 『GRANT(고유한 유형 권한)』를 참조하십시오.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 DATA는 DISTINCT의 동의어로 사용될 수 있습니다.

예

사용자 JONES로부터 고유한 유형 SHOESIZE의 USAGE 권한을 취소합니다.

```
REVOKE USAGE
  ON DISTINCT TYPE SHOESIZE
  FROM JONES
```

REVOKE(함수 또는 프로시듀어 권한)

이 형식의 REVOKE문은 함수나 프로시듀어에 대한 권한을 제거합니다.

호출

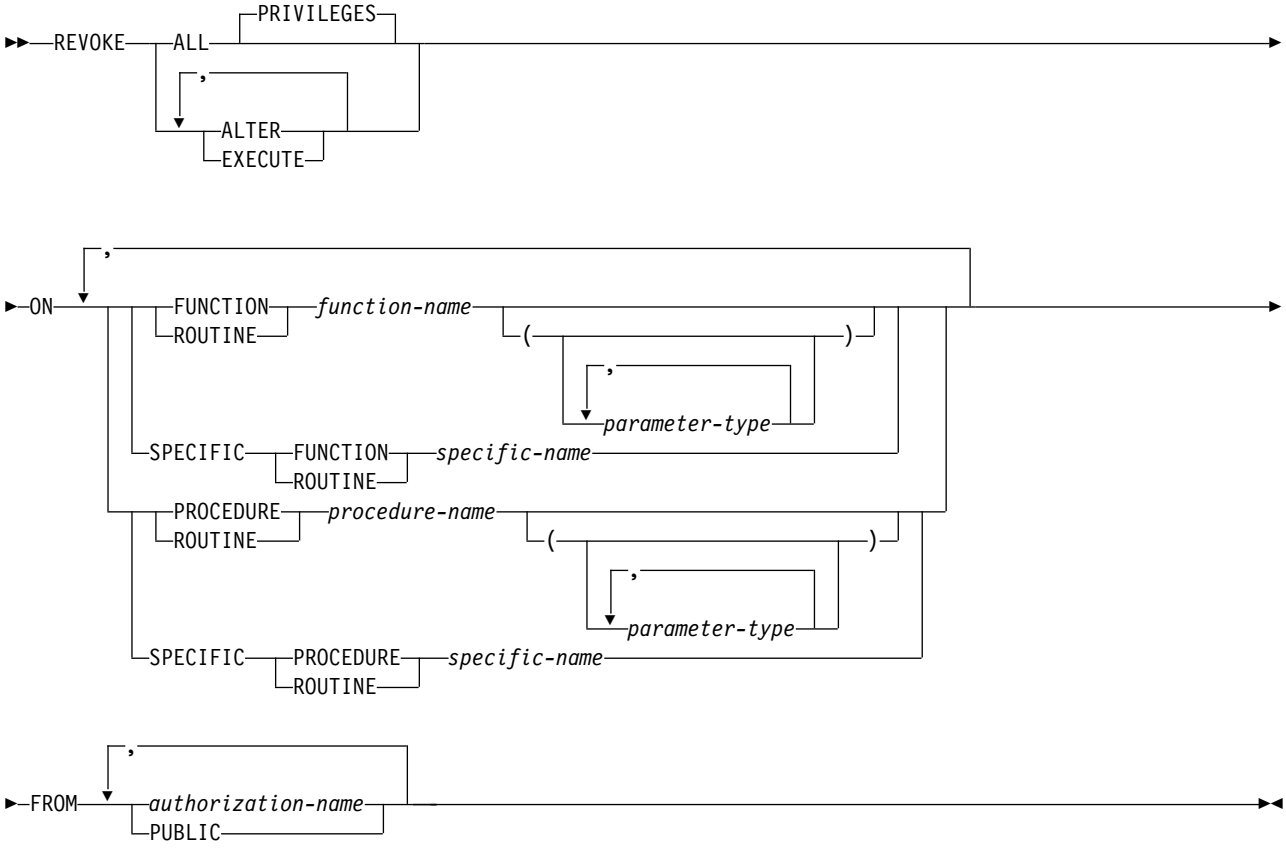
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에 식별된 각 함수나 프로시듀어에 대해 다음과 같은 권한을 갖습니다.
  - 명령문에 지정된 모든 권한
  - 함수나 프로시듀어에 대한 *OBJMGT 시스템 권한
  - 함수나 프로시듀어가 들어 있는 라이브러리(또는 Java 루틴인 경우 디렉토리)에 대한 *EXECUTE 시스템 권한
- 관리 권한

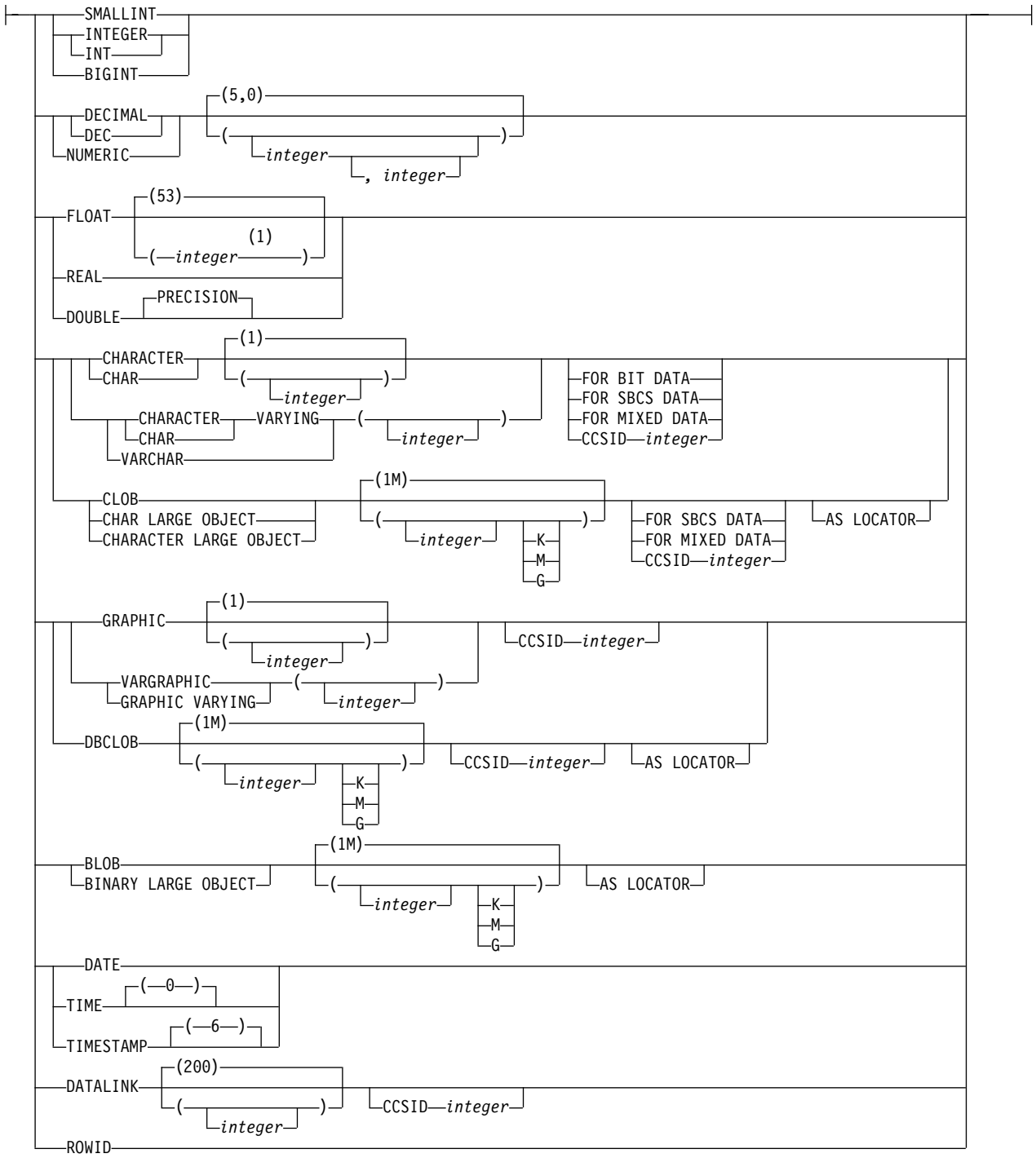
구문



parameter-type:

REVOKE(함수 또는 프로시듀어 권한)

built-in-type:



주:

1 자료 유형(REAL 또는 DOUBLE)을 기초로 일치가 이루어지므로 정밀도에 대해 지정한 값이 함수를 작성할 때 지정한 값과 일치하지 않아도 됩니다.

설명

ALL 또는 **ALL PRIVILEGES**

각 *authorization-name*에서 하나 이상의 함수나 프로시듀어 권한을 취소합니다. 취소되는 권한은 *authorization-names*에 부여했던 식별된 함수 또는 프로시듀어에 대한 권한입니다. 함수나 프로시듀어에 대한 ALL PRIVILEGES를 취소하는 것이 *ALL 시스템 권한을 취소하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드는 설명된 권한을 취소합니다.

ALTER

COMMENT문을 사용할 권한을 취소합니다.

EXECUTE

함수나 프로시듀어를 실행할 권한을 취소합니다.

FUNCTION

사용자가 권한을 취소하고 있는 함수를 식별합니다. 이름, 함수 표시나 특정 이름으로 특별한 함수를 식별할 수 있습니다. 함수 해결(및 경로)에 대한 규칙은 사용되지 않습니다.

FUNCTION *function-name*

*function-name*은 현재 서버에 있는 한 함수를 정확히 식별해야 합니다. 함수는 함수에 대해 정의된 매개변수를 가질 수 있습니다. 지정된 또는 내재된 스키마에 지정된 이름의 함수가 둘 이상 있으면 오류가 리턴됩니다.

FUNCTION *function-name(parameter-type, ...)*

function-name(parameter-type, ...) 현재 서버에 있는 지정된 함수 표시로 함수를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE FUNCTION 문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수와 자료 유형의 논리 연결이 취소될 특정 함수 인스턴스를 식별하는 데 사용됩니다. *function-name()*이 지정되면 식별된 함수는 0개의 매개변수를 가져야 합니다.

function-name

함수 이름을 식별합니다.

(parameter-type, ...)

함수의 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.

- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE FUNCTION문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

  ```
  CHAR      CHAR(1)
  GRAPHIC   GRAPHIC(1)
  DECIMAL   DECIMAL(5,0)
  FLOAT     DOUBLE (길이 8)
  ```

  내재된 길이는 CREATE FUNCTION문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE FUNCTION문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC FUNCTION *specific-name*

*specific-name*은 현재 서버에 있는 특정 함수를 식별해야 합니다.

PROCEDURE

사용자가 권한을 취소하고 있는 프로시듀어를 식별합니다. 이름, 프로시듀어 표시 또는 특정 이름으로 특정 프로시듀어를 식별할 수 있습니다. 프로시듀어 해결(및 경로)에 대한 규칙은 사용되지 않습니다.

PROCEDURE *procedure-name*

*procedure-name*은 현재 서버에 있는 한 프로시듀어를 정확히 식별해야 합니다. 프로시듀어는 프로시듀어에 대해 정의된 매개변수를 갖습니다. 지정된 또는 내재된 스키마에 지정된 이름의 프로시듀어가 둘 이상 있는 경우 오류가 리턴됩니다.

PROCEDURE *procedure-name(parameter-type, ...)*

The *procedure-name(parameter-type, ...)*은 현재 서버에 있는 지정된 프로시듀어 표시로 프로시듀어를 식별해야 합니다. 지정한 매개변수는 해당 위치에서 CREATE PROCEDURE문에 지정된 자료 유형과 일치해야 합니다. 자료 유형의 수, 자료 유형의 논리 연결이 제거될 특정 프로시듀어 인스턴스를 식별하는 데 사용됩니다. *procedure-name()*이 지정되면 식별된 프로시듀어는 0개의 매개변수를 가져야 합니다.

procedure-name

프로시듀어 이름을 식별합니다.

(parameter-type, ...)

프로시듀어 매개변수를 식별합니다.

규정되지 않은 고유한 유형 이름이 지정된 경우 데이터베이스 관리자는 SQL 경로를 찾아서 고유한 유형에 대한 스키마명을 해석합니다.

길이, 정밀도 또는 배율 속성을 갖는 자료 유형의 경우에는 값을 지정할 수도 있고 빈 괄호 세트를 사용할 수도 있습니다.

- 빈 괄호는 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다.
- 길이, 정밀도 또는 눈금 속성에 대해 특정 값을 사용하면 그 값은 CREATE PROCEDURE문에 지정된(내재적 또는 명시적으로) 값과 정확히 일치해야 합니다.
- 길이, 정밀도 또는 눈금이 명시적으로 지정되지 않고 빈 괄호도 지정되지 않은 경우 자료 유형의 디폴트 속성이 내재됩니다. 예를 들면, 다음과 같습니다.

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (길이 8)
```

내재된 길이는 CREATE PROCEDURE문에 내재적으로 또는 명시적으로 지정된 값과 정확히 일치해야 합니다. 자료 유형의 디폴트 길이에 대한 전체 리스트는 541 페이지의 『CREATE TABLE』을 참조하십시오.

부속 유형이나 코드화 문자 세트 ID(CCSID) 속성을 갖는 자료 유형의 경우 FOR DATA절이나 CCSID 절은 선택적입니다. 절의 생략은 자료 유형의 일치 여부를 판별할 때 데이터베이스 관리자가 속성을 무시한다는 것을 표시합니다. 절을 지정하는 경우 CREATE PROCEDURE문에 내재적 또는 명시적으로 지정된 값과 일치해야 합니다.

SPECIFIC PROCEDURE *specific-name*

*specific-name*은 현재 서버에 있는 특정 프로시듀어를 식별해야 합니다.

FROM

권한이 취소되는 사용자를 식별합니다.

authorization-name,...

하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC

지정한 권한을 PUBLIC으로부터 취소합니다.

주

함수나 프로시듀어에 대한 권한을 취소하면 권한 부여자에 관계없이 해당 함수나 프로시듀어에 대한 모든 권한 부여가 무효화됩니다.

SQL 또는 외부 함수나 프로시듀어로부터 취소된 권한은 관련 프로그램(*PGM) 또는 서비스 프로그램(*SRVPGM) 오브젝트에서도 취소됩니다.

함수나 프로시듀어 권한이 취소될 때 그에 해당하는 시스템 권한이 취소됩니다. SQL 권한에 해당하는 시스템 권한에 대한 내용은 687 페이지의 『GRANT(함수 또는 프로시듀어 권한)』를 참조하십시오.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- RUN 키워드를 EXECUTE의 동의어로 사용할 수 있습니다.

예

CORPDATA.PROCA 프로시듀어에 대한 EXECUTE 권한을 PUBLIC으로부터 취소합니다.

```
REVOKE EXECUTE
  ON PROCEDURE CORPDATA.PROCA
  FROM PUBLIC
```

REVOKE(패키지 권한)

이 형식의 REVOKE문은 패키지에 대한 권한을 제거합니다.

호출

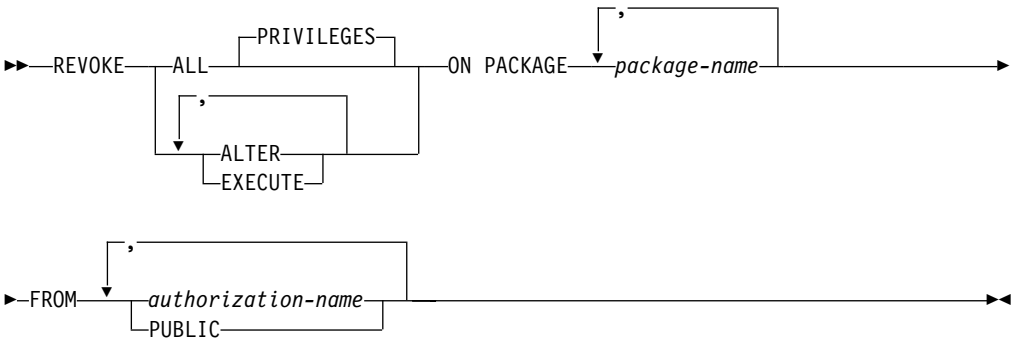
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에 식별된 각 패키지에 대해 다음과 같은 권한을 갖습니다.
  - 명령문에 지정된 모든 권한
  - 패키지에 대한 *OBJMGT 시스템 권한
  - 패키지가 들어 있는 라이브러리에 대한 *EXECUTE 시스템 권한
- 관리 권한

구문



설명

ALL 또는 ALL PRIVILEGES

각 *authorization-name*으로부터 하나 이상의 패키지 권한을 취소합니다. 취소된 권한은 *authorization-names*에 부여했던 식별된 패키지에 대한 권한입니다. 패키지에 대한 ALL PRIVILEGES를 취소하는 것이 *ALL 시스템 권한을 취소하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드는 설명된 권한을 취소합니다.

ALTER

COMMENT 및 LABEL문을 사용할 권한을 취소합니다.

EXECUTE
패키지의 명령문을 실행할 권한을 취소합니다.

ON PACKAGE *package-name*
사용자가 권한을 취소하고 있는 패키지를 식별합니다. *package-name*은 현재 서버에 있는 패키지를 식별해야 합니다.

FROM
권한이 취소되는 사용자를 식별합니다.

authorization-name,...
하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC
지정한 권한을 PUBLIC으로부터 취소합니다.

주

패키지에 대한 권한을 취소하면 권한 부여자에 관계없이 해당 패키지에 대한 모든 권한 부여가 무효화됩니다.

패키지 권한이 취소될 때 그에 해당하는 시스템 권한도 취소됩니다. SQL 권한에 해당되는 시스템 권한에 대한 정보는 695 페이지의 『GRANT(패키지 권한)』를 참조하십시오.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- RUN 키워드를 EXECUTE의 동의어로 사용할 수 있습니다.
- 키워드 PROGRAM은 PACKAGE와 동의어로 사용될 수 있습니다.

예

CORPDATA.PKGA 패키지에 대한 EXECUTE 권한을 PUBLIC으로부터 취소합니다.

```
REVOKE EXECUTE
  ON PACKAGE CORPDATA.PKGA
  FROM PUBLIC
```

REVOKE(표 권한)

이 형식의 REVOKE문은 표에 대한 권한을 제거합니다.

호출

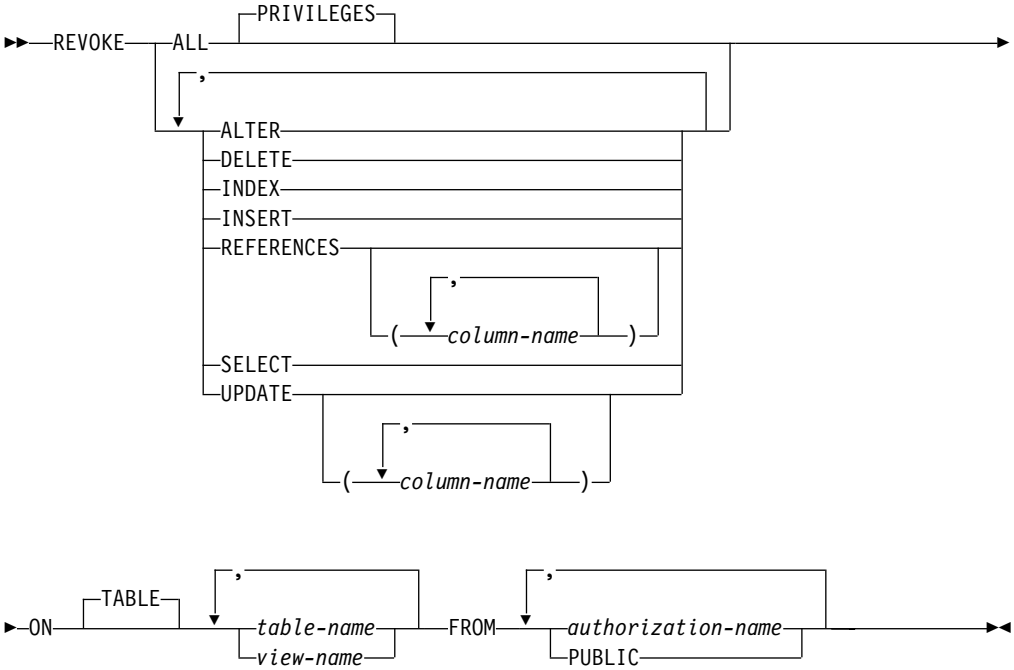
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에 식별된 각 표나 뷰에 대해 다음과 같은 권한을 갖습니다.
  - 명령문에 지정된 모든 권한
  - 표나 뷰에 대한 *OBJMGT 시스템 권한
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

구문



설명

ALL 또는 **ALL PRIVILEGES**

각 *authorization-name*으로부터 하나 이상의 권한을 취소합니다. 취소된 권한은

*authorization-names*에 부여했던 식별된 표와 뷰에 대한 권한입니다. 표나 뷰에 대한 ALL PRIVILEGES를 취소하는 것이 *ALL 시스템 권한을 취소하는 것과 동일하지 않음에 유의하십시오.

ALL을 사용하지 않을 경우 아래 나열된 키워드 중 하나 이상을 사용해야 합니다. 각 키워드가 설명된 권한을 취소하며 ON절에 이름이 지정된 표와 뷰에만 적용합니다.

ALTER
표에 ALTER TABLE문을 사용할 권한을 취소합니다. 또한 표와 뷰에 대해 COMMENT 및 LABEL 문을 사용할 권한을 취소합니다.

DELETE
DELETE문을 사용할 권한을 취소합니다.

INDEX
CREATE INDEX문을 사용할 권한을 취소합니다.

INSERT
INSERT문을 사용할 권한을 취소합니다.

REFERENCES
해당 표가 상위가 되는 참조 제한조건을 추가할 권한을 취소합니다.

REFERENCES(*column-name***,...)**
상위 키에 지정된 열을 사용하여 참조 제한조건을 추가하는 권한을 취소합니다. 각 열 이름은 규정화되지 않은 이름으로서 ON절에 식별된 각 표의 열을 식별해야 합니다.

SELECT
SELECT 또는 CREATE VIEW문을 사용할 권한을 취소합니다.

UPDATE
UPDATE문을 사용할 권한을 취소합니다.

UPDATE(*column-name***,...)**
지정한 열을 갱신할 권한을 취소합니다. 각 열 이름은 규정화되지 않은 이름으로서 ON절에 식별된 각 표의 열을 식별해야 합니다.

ON *table-name* 또는 *view-name,...*
사용자가 권한을 취소하고 있는 표 또는 뷰를 식별합니다. *table-name* 또는 *view-name*은 현재 서버에 있는 표나 뷰를 식별하지만 글로벌 임시 표를 식별해서는 안됩니다.

FROM
권한이 취소되는 사용자를 식별합니다.

authorization-name,...

하나 이상의 권한부여 ID를 나열합니다. 동일한 *authorization-name*을 두 번 이상 지정하지 마십시오.

PUBLIC

지정한 권한을 PUBLIC으로부터 취소합니다.

주

시스템 권한

INDEX 또는 ALTER 권한을 취소하면 시스템 권한 *OBJALTER도 취소됩니다.

표 또는 뷰 권한이 취소될 때 그에 해당하는 시스템 권한도 취소됩니다. 단, 다음 경우는 예외입니다.

- 표나 뷰에 대한 권한을 취소할 때 *ADD, *DLT, *READ 및 *UPD가 모두 이미 취소된 경우에만 *OBJOPR이 취소됩니다.
- 뷰에 대한 권한을 취소할 때 뷰 정의의 subselect에 참조된 표나 뷰로부터는 권한이 취소되지 않습니다.

SQL 권한을 사용하여 둘 이상의 시스템 권한을 취소하며 그 권한 중 취소할 수 없는 권한이 하나라도 있으면 경고가 발생하고 그 권한에 대해서는 어떠한 권한도 취소하지 않습니다.

SQL 권한에 해당하는 시스템 권한에 대한 내용은 698 페이지의 『GRANT(표 권한)』를 참조하십시오.

복수 부여

한 사용자에게 동일한 권한을 두 번 이상 부여한 경우 그 사용자로부터 권한을 취소하면 모든 부여가 무효화됩니다.

권한을 취소하면 권한 부여자에 관계없이 해당 권한에 대한 모든 부여가 무효화됩니다.

WITH GRANT OPTION을 취소하는 유일한 방법은 ALL을 취소하는 것입니다.

예

예 1

사용자 PULASKI로부터 EMPLOYEE 표에 대한 SELECT 권한을 취소합니다.

```
REVOKE SELECT
  ON EMPLOYEE
  FROM PULASKI
```

예 2

이전에 모든 로컬 사용자에게 부여했던 EMPLOYEE 표에 대한 갱신 권한을 취소합니다. 특정 사용자에게 부여한 것은 영향을 받지 않습니다.

```
REVOKE UPDATE
  ON EMPLOYEE
  FROM PUBLIC
```

예 3

사용자 KWAN과 THOMPSON으로부터 EMPLOYEE 표에 대한 모든 권한을 취소합니다.

```
REVOKE ALL
  ON EMPLOYEE
  FROM KWAN,THOMPSON
```

예 4

FRED로부터 view1에 있는 column_1 갱신 권한을 취소합니다.

```
REVOKE UPDATE(column_1)
  ON VIEW1
  FROM FRED
```

ROLLBACK

ROLLBACK 명령문은 다음과 같이 사용될 수 있습니다:

- 단위를 종료하고, 해당 작업 단위가 수행한 모든 관계형 데이터베이스 변경 내용을 역처리합니다. 관계형 데이터베이스가 어플리케이션 프로세스에 사용되는 유일한 복구 가능 자원인 경우, ROLLBACK은 작업 단위 또한 종료합니다.
- 작업 단위를 종료하지 않고 작업 단위 내에 설정된 저장점 이후의 변경사항만을 역처리합니다. 저장점으로 롤백하면 선택된 변경사항이 취소됩니다.

호출

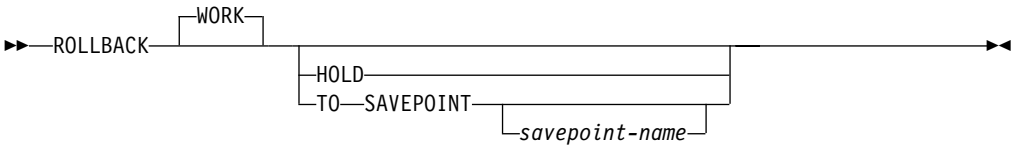
이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

트리거 프로그램과 트리거하는 프로그램이 동일한 확약(commit) 정의 아래 실행되는 경우 ROLLBACK은 트리거 프로그램에 허용되지 않습니다. CALL문을 발행한 외부 프로시듀어 및 프로그램이 동일한 확약 정의 아래 실행되는 경우 ROLLBACK은 외부 프로시듀어에 허용되지 않습니다.

권한부여

필요한 사항이 없습니다.

구문



설명

ROLLBACK이 SAVEPOINT 절 없이 사용될 때, 실행되고 있던 작업 단위는 종료하고 새로운 작업 단위가 시작됩니다. 해당 작업 단위중 실행된 ALTER, CALL, COMMENT, CREATE, DECLARE GLOBAL TEMPORARY TABLE, DELETE, DROP(DROP SCHEMA은 제외), GRANT, INSERT, LABEL, RENAME, REVOKE 및 UPDATE문이 수행한 모든 변경내용이 취소됩니다.

그러나, 다음 명령문은 트랜잭션에서 제어되지 않으며 해당 변경사항은 ROLLBACK 문 실행과 상관 없습니다.

- CONNECT
- DISCONNECT
- RELEASE CONNECTION

- SET CONNECTION
- SET PATH
- SET SCHEMA

ROLLBACK 또는 ROLLBACK TO SAVEPOINT가 선언된 글로벌 임시 표의 내용에 미치는 영향은 DECLARE GLOBAL TEMPORARY TABLE문의 ON ROLLBACK절을 설정하여 결정됩니다.

WORK

ROLLBACK WORK는 ROLLBACK와 동일한 효과를 갖습니다.

HOLD

자원의 보류를 지정합니다. 이 절을 지정하면 현재 열린 커서가 닫히지 않고 작업 단위중 예약한 모든 자원이 계속 보유됩니다. 단, 표의 행에 대한 잠금은 예외입니다. 그러나, 작업 단위중 암시적으로 예약된 특정 행에 대한 잠금은 해제됩니다.

HOLD를 생략하면, TO SAVEPOINT절 없는 ROLLBACK은 이 작업 단위의 확약 정의에 다음과 같은 상황을 발생시킵니다.

- 해당 작업 단위의 확약 정의에서 열린 커서가 닫힙니다.
- 작업 단위의 확약 정의에서 LOCK TABLE문에 의해 예약된 표 잠금이 해제됩니다.
- 보유 중인 것을 포함한 모든 LOB 로케이터가 해제됩니다.

ROLLBACK HOLD의 끝에서 커서의 위치는 커서가 들어 있는 프로그램이나 루틴이 작성될 때 ALWBLK(*ALLREAD)가 지정되지 않는 한 해당 작업 단위를 시작할 때의 위치와 동일합니다.

TO SAVEPOINT

작업 단위가 종료되지 않으며 부분 롤백(저장점으로)만 수행됨을 지정합니다. 저장점 이름이 지정되지 않을 경우, 마지막 활성 저장점으로 롤백됩니다. 예를 들어, 작업 단위, 저장점 A, B, C가 해당 순서로 설정되고 C가 릴리스된 경우, ROLLBACK TO SAVEPOINT는 저장점 B로 롤백이 이루어지도록 합니다.

savepoint-name

롤백할 저장점을 식별합니다. 명명된 저장점이 없다면, 오류가 발생합니다.

ROLLBACK TO SAVEPOINT가 성공한 후에도 저장점이 계속 존재합니다.

저장점이 설정된 이후의 모든 데이터베이스 변경사항(선언된 임시 테이블 변경사항 포함)이 역처리됩니다. 모든 잠금 및 LOB 로케이터가 보유됩니다.

ROLLBACK TO SAVEPOINT로 인해 커서에 미치는 영향은 저장점 내의 명령문에 따라 달라집니다.

- 저장점에 커서가 종속된 DDL이 포함될 경우, 커서는 닫힙니다. ROLLBACK TO SAVEPOINT 이후에 그러한 커서를 사용하려고 하면 오류가 발생합니다.
- 아니면, 커서는 ROLLBACK TO SAVEPOINT의 영향을 받지 않습니다(계속 열려 있고 위치하고 있음).

롤백이 수행된 후 설정된 모든 저장점은 해제됩니다. 롤백이 수행된 저장점은 해제되지 않습니다.

주

디폴트 활성 그룹을 종료하면 암시적 롤백이 발생합니다. 따라서 디폴트 활성 그룹을 종료하기 전에 명시적 COMMIT문이나 ROLLBACK문을 발행해야 합니다.

다음과 같은 경우에 ROLLBACK이 자동으로 수행됩니다.

1. 디폴트 활성 그룹이 마지막 COMMIT 발행 없이 종료될 경우.]
2. 활성 그룹이 작업을 완료하는 것을 막는 장애가 발생할 경우(예를 들어, 전원 장애)

   장애가 발생했을 때 COMMIT가 처리되고 있었기 때문에 작업 단위가 준비된 상태로 있으면, 롤백이 수행되지 않습니다. 롤백이 수행되는 대신, 작업 단위에 포함된 모든 연결에 대한 재동기화가 발생합니다. 자세한 정보는 확약 제어 주제를 참조하십시오.
3. 서버와의 연결이 끊어지는 장애가 발생하는 경우(예를 들어, 통신 회선 장애)

   장애가 발생했을 때 COMMIT가 처리되고 있었기 때문에 작업 단위가 준비된 상태로 있으면, 롤백이 수행되지 않습니다. 롤백이 수행되는 대신, 작업 단위에 포함된 모든 연결에 대한 재동기화가 발생합니다. 자세한 정보는 확약 제어 주제를 참조하십시오.
4. 디폴트가 아닌 활성 그룹이 비정상 종료되는 경우

작업 단위에는 SELECT INTO 또는 FETCH문[64]을 실행하는 동안 검색된 행과 INSERT DELETE 및 UPDATE 조작의 일부로서 삽입, 삭제 또는 갱신된 행을 포함해 최대 4백만 개까지의 행 처리가 포함될 수 있습니다.[65]

확약과 롤백 조작은 DROP SCHEMA문에 영향을 미치지 않으므로 이 명령문을 COMMIT(*CHG), COMMIT(*CS), COMMIT(*ALL) 또는 COMMIT(*RR)을 지정하는 어플리케이션 프로그램에서 사용할 수 없습니다.

---

64. COMMIT(*CHG) 또는 COMMIT(*CS)를 지정하지 않았으면, 그런 경우 이 행들은 총계에 포함되지 않습니다.

65. 이 제한에는 다음도 포함됩니다.

    - 고급 언어 파일 처리를 통해 확약 제어에서 열린 파일을 통해 액세스하거나 변경한 행
    - 트리거나 CASCADE, SET NULL 또는 SET DEFAULT 참조 무결성 삭제 규칙의 결과로 삭제, 갱신 또는 삽입된 행

ROLLBACK 명령문은 활성 그룹에 대해 확약 제어가 작동하지 않는 경우 사용할 수 없습니다. 사용하는 확약 정의 판별에 대한 정보는 COMMIT문의 확약 정의 토론을 참조하십시오.

준비된 명령문이 손상되었을 때 이 명령문과 연관된 커서는 명령문이 다시 준비될 때까지 열지 못합니다. ROLLBACK은 연결의 상태에 영향을 주지 않습니다.

작업 단위 내에서 CLOSE 뒤에 ROLLBACK이 나오는 경우 작업 단위 내에서 이루어진 모든 변경 내용이 취소됩니다. CLOSE 자체는 취소되지 않아서 파일이 다시 열리지 않습니다.

예

예 1

ROOBACK문의 사용 예는 418 페이지의 COMMIT에 나오는 설명을 참조하십시오.

예 2

회복 단위가 시작된 후 세 개의 저장점 A, B, C가 설정되었으며 C가 해제된 경우,

```
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
...
SAVEPOINT B ON ROLLBACK RETAIN CURSORS;
....
SAVEPOINT C ON ROLLBACK RETAIN CURSORS;
...
RELEASE SAVEPOINT C
```

모든 DB2 데이터베이스 변경사항을 저장점 A로만 롤백합니다.

```
ROLLBACK WORK TO SAVEPOINT A
```

저장점명이 지정되지 않은 경우(즉, ROLLBACK WORK TO SAVEPOINT), 설정된 마지막 활성 저장점인 B로 롤백됩니다.

SAVEPOINT

SAVEPOINT문은 작업 단위 내에 저장점을 설정하여 작업 단위 내에서 관계형 데이터베이스 변경사항이 롤백될 수 있는 시점을 나타냅니다.

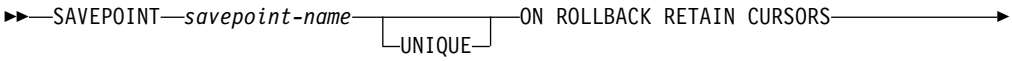
호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

필요한 사항이 없습니다.

구문



주:

1 ROLLBACK 옵션을 다른 순서로 지정할 수 있습니다.

설명

savepoint-name
새 저장점을 식별합니다.

UNIQUE
어플리케이션이 작업 단위 내에서 저장점 이름을 다시 사용할 수 없다고 지정합니다. *savepoint-name*과 동일한 이름의 저장점이 작업 단위 내에 이미 존재할 경우 오류가 발생합니다.

UNIQUE를 생략하면 어플리케이션은 작업 단위 내의 저장점을 재사용할 수 있습니다. *savepoint-name*이 작업 단위 내에 이미 존재하는 저장점을 나타내고 저장점이 UNIQUE 옵션으로 작성되지 않은 경우, 기존 저장점이 폐기되고 새 저장점이 작성됩니다. 저장점을 폐기하고 다른 저장점에 해당 이름을 재사용하는 것은 저장점을 릴리스하는 것과 같지 않습니다. 저장점 이름을 다시 사용하면 단 하나의 저장점만 파기됩니다. RELEASE SAVEPOINT 명령문을 사용하여 저장점을 해제하면 해당 저장점과 이어서 설정된 모든 저장점이 해제됩니다.

ON ROLLBACK RETAIN CURSORS

저장점이 설정된 후 열린 커서가 저장점으로 롤백시 닫히지 않도록 지정합니다.

- 저장점에 커서가 종속된 DDL이 포함될 경우, 커서는 닫힙니다. ROLLBACK TO SAVEPOINT 이후에 그러한 커서를 사용하려고 하면 오류가 발생합니다.
- 아니면, 커서는 ROLLBACK TO SAVEPOINT의 영향을 받지 않습니다(계속 열려 있고 위치하고 있음).

저장점으로 롤백한 후에도 해당 커서는 열려 있지만 사용할 수 없을 수도 있습니다. 예를 들어, 저장점으로 롤백함으로써 커서가 위치한 행의 삽입이 롤백되는 경우, 커서를 사용하여 행을 갱신하거나 삭제하면 오류가 발생합니다.

ON ROLLBACK RETAIN LOCKS

저장점이 설정되기 전에 취득된 모든 잠금이 저장점으로 롤백시 해제되지 않도록 지정합니다.

주

어플리케이션에서, 삽입이 버퍼링되었을 수 있습니다. SAVEPOINT, ROLLBACK 또는 RELEASE TO SAVEPOINT 명령문이 실행되면 버퍼가 플러시됩니다.

SAVEPOINT 명령문은 활성 그룹에 대해 확약 제어가 작동하지 않는 경우 사용할 수 없습니다. 사용하는 확약 정의 판별에 대한 정보는 419 페이지의 『주』를 참조하십시오.

예

작업 단위의 여러 지점에서 세 개의 저장점을 설정하려는 경우, 첫 번째 저장점 A의 이름을 지정하고 저장점 이름을 재사용할 수 있도록 하십시오. 두 번째 저장점 B의 이름을 지정하고 이름을 재사용할 수 없도록 하십시오. 세 번째 저장점을 설정할 준비가 되었을 때 저장점 A가 더 이상 필요하지 않기 때문에 A를 저장점 이름으로 재사용하십시오.

```
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
.
.
.
SAVEPOINT B UNIQUE ON ROLLBACK RETAIN CURSORS;
.
.
.
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
```

SELECT

SELECT 명령문은 쿼리의 한 형태입니다. 대화식으로 발행될 수 있습니다. 자세한 정보는 352 페이지의『select문』 및 335 페이지의 제 4 장『조회』를 참조하십시오.

SELECT INTO

SELECT INTO문은 단 하나의 행으로 구성된 결과표를 작성하며, 해당 행의 값을 호스트 변수에 지정합니다. 표가 비어 있는 경우 이 명령문은 +100을 SQLCODE에 지정하고 '02000'을 SQLSTATE에 지정하며, 호스트 변수에는 값을 지정하지 않습니다. 탐색 조건을 만족하는 행이 두 개 이상이면 명령문의 처리가 종료되면서 오류가 발생합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. REXX에 지정되어서는 안됩니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 각 표 또는 뷰에 대하여
  - 표 또는 뷰에서의 SELECT 권한 및
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

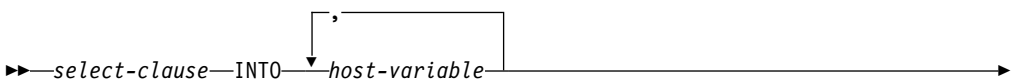
명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

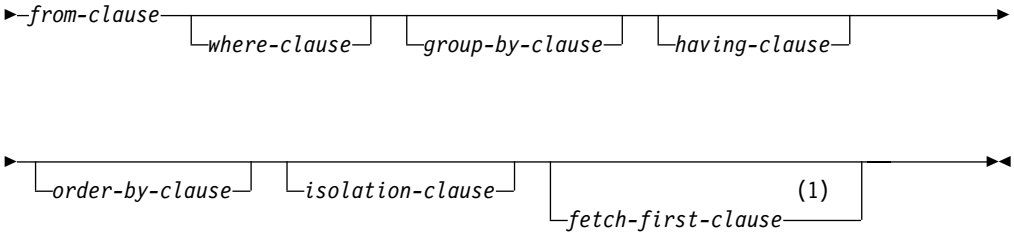
- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

구문





주:

1 fetch-first절에는 한 행만을 지정할 수 있습니다.

설명

결과표는 *isolation-절, from절, where절, group-by절, having절, select절, order-by절,* 및 *fetch-first-절*을 순서대로 평가하여 작성됩니다.

select-clause, *from-clause*, *where-clause*, *group-by-clause*, *having-clause*, *order-by-clause*, *fetch-first-clause* 및 *isolation-clause*에 관한 설명은 335 페이지의 제 4 장 『조회』를 참조하십시오.

*group-by-clause*에 의해 지정된 대로 그룹화하는 것은, 행이 둘 이상인 결과표 작성이 거의 확실하여 *having-clause*에서 표의 행 수를 하나로 줄이는 것이 필요할 수 있음에 유의하십시오.

INTO *host variable*,...

호스트 구조 및 변수의 선언 규칙에 따라서 프로그램에 선언해야 하는 하나 이상의 호스트 구조나 변수를 식별합니다. INTO 절의 조작 형식에서 호스트 구조에 대한 참조는 해당 변수 각각에 대한 참조로 대체됩니다. 결과 행의 첫 번째 값이 리스트의 첫 번째 호스트 변수에 지정되고, 두 번째 값은 두 번째 호스트 변수에 지정되는 방법으로 행의 값이 변수에 지정됩니다. 각 호스트 변수의 자료 유형은 대응되는 열과 호환될 수 있어야 합니다.

호스트 변수에 대한 각 지정은 제 2 장에 설명된 규칙에 따라 수행됩니다. 변수의 수가 행에 있는 값의 수 미만인 경우 SQLCA의 SQLWARN3 필드가 'W'로 설정됩니다. 결과 행의 수보다 호스트 변수가 더 많을 경우 경고가 발행되지 않음에 유의하십시오. 그 값이 널(null)인 경우 인디케이터 변수를 제공해야 합니다. 지정 오류가 발생하면 해당 호스트 변수와 그 뒤에 나오는 호스트 변수들의 값을 예측할 수 없습니다. 변수에 이미 지정된 값은 그대로 남아 있습니다.

select절의 결과 열을 평가할 때 다음 자료 맵핑 오류 중 하나가 발생하면 결과는 널값이 됩니다.

- 문자를 변환할 수 없음
- 숫자 변환 오류(부족(underflow) 또는 넘침)

- 연산식 오류(0으로 나눔)
- 날짜 또는 시간소인 오류(지정된 형식에 대해 유효한 날짜 범위 안에 있지 않은 날짜나 시간소인)
- 유효하지 않은 datetime 값의 스트링 표시
- 제대로 형성되지 않은 혼합된 자료
- 숫자 값이 유효하지 않음
- 범위를 벗어나는 SUBSTR 스칼라 함수의 인수

그 밖의 널값 경우에서는 인디케이터 변수를 제공해야 합니다. 호스트 변수의 값은 정의되지 않습니다. 그러나 이 경우에 인디케이터 변수는 -2로 설정됩니다. 그리고 오류가 발생하지 않았던 것처럼 명령문의 처리가 계속됩니다(그러나 이 오류는 양의 SQLCODE를 유발합니다). 인디케이터 변수를 제공하지 않으면 SQLCA의 SQLCODE 필드에 음의 값이 리턴됩니다. 해당 변수나 그 뒤에 나오는 변수의 값은 예측할 수 없습니다. 변수에 이미 지정된 값은 그대로 남아 있습니다.

결과표에 행이 두 개 이상이어서 오류가 발생하는 경우(SQLCODE 21000), 모든 호스트 변수에 값이 지정되지만 값의 소스가 되는 행은 정의되지 않으며 예측할 수도 없습니다.

예

예 1

COBOL 프로그램 명령문을 사용하여 최대 급여(SALARY)를 분리 레벨 Read Committed(CS)를 사용하여 EMPLOYEE 표에서 호스트 변수 MAX-SALARY (DECIMAL(9,2))로 넣습니다.

```
EXEC SQL  SELECT MAX(SALARY)
            INTO :MAX-SALARY
            FROM EMPLOYEE WITH CS
END-EXEC.
```

예 2

Java 프로그램 명령문을 사용하여 연결 문맥 'ctx'의 EMPLOYEE 표에서 호스트 변수 HOST_EMP (java.lang.Strin)에 저장한 값과 동일한 사원 번호(EMPNO) 값을 갖는 행을 선택합니다. 그 다음, 선택된 행에서 사원의 성(LASTNAME)과 학력(EDLEVEL)을 각각 호스트 변수 HOST_NAME(String)과 HOST_EDUCATE(integer)에 넣습니다.

```
#sql [ctx] {  SELECT LASTNAME, EDLEVEL
            INTO :HOST_NAME, :HOST_EDUCATE
              FROM EMPLOYEE
              WHERE EMPNO = :HOST_EMP   };
```

SET CONNECTION

SET CONNECTION문은 활성 그룹의 기존의 연결 중 하나를 식별하여 활성 그룹의 현재 서버를 구축합니다.

호출

이 명령문은 어플리케이션 프로그램에 삽입되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. Java나 REXX에 지정되어서는 안됩니다.

SET CONNECTION은 트리거 프로그램에 허용되지 않습니다. 외부 프로시듀어가 리모트 서버에 호출되는 경우에는 외부 프로시듀어에 SET COONECTION이 허용되지 않습니다.

권한부여

필요한 사항이 없습니다.

구문

설명

server-name 또는 *host-variable*

지정한 서버명 또는 호스트 변수에 있는 서버명으로 연결을 식별합니다. 호스트 변수가 지정된 경우

- 문자 스트링 변수가 되어야 합니다.
- 다음에 인디케이터 변수가 오면 안됩니다.
- 서버명은 호스트 변수 내에서 왼쪽부터 배열되어야 하고 일반 ID를 형성하는 규칙을 따라야 합니다.
- 서버명의 길이가 호스트 변수의 길이보다 작으면 오른쪽은 공백으로 채워져야 합니다.

S는 호스트 변수에 포함된 서버명이나 지정된 서버명을 나타냅니다. S는 어플리케이션 프로세스의 기존의 연결을 식별해야 합니다. S가 현재 연결을 식별하는 경우 S의 상태와 어플리케이션 프로세스의 다른 모든 연결의 상태가 변하지 않지만, S에 대한 정보가 SQLCA의 SQLERRP 필드로 들어갑니다. S가 활동중이 아닌 연결을 식별할 때 다음과 같은 규칙이 적용됩니다.

SET CONNECTION문이 성공적으로 수행되는 경우:

- 연결 S가 현재 상태로 됩니다.
- S는 CURRENT SERVER 특수 레지스터에 위치합니다.
- 서버 S에 대한 정보는 SQLCA의 SQLERRP 필드에 있습니다. 서버가 IBM 관계형 데이터베이스 제품이면 정보는 *pppvvrrm* 형식입니다.
  여기에서
  - *ppp*는 다음과 같은 제품을 식별합니다.
    ARI for DB2 for VSE와 VM
    DSN for OS/390 및 z/OS용 DB2 UDB
    QSQ for iSeries용 DB2 UDB
    그 외의 모든 DB2 제품용 SQL
  - *vv*는 '04'와 같은 두 자리의 버전 ID입니다.
  - *rr*은 '01'과 같은 두 자리의 릴리스 ID입니다.
  - *m*은 '0'과 같은 한 자리의 수정 레벨입니다.

  예를 들어 서버가 OS/390 및 z/OS용 DB2 UDB 버전 4이면 SQLERRP의 값은 'DSN04010'입니다.
- 연결에 관한 추가 정보는 SQLCA의 SQLERRD(4) 필드에 들어갑니다. SQLERRD(4)에는 서버가 확약가능한 갱신의 수행을 허용하는지 여부를 표시하는 값이 들어 있습니다. 다음은 CONNECT에 있는 SQLCA의 SQLERRD(4) 필드에 대한 값과 의미 리스트입니다.
  - 1 - 확약가능한 갱신이 수행될 수 있으며 연결에서 비보호 대화를 사용하거나 연결이 CONNECT(유형 1)문을 사용하여 대화식 리퀘스터 드라이버 프로그램에 구축한 연결 또는 CONNECT(유형 1)문을 사용하여 구축한 로컬 연결입니다.
  - 2 - 어떠한 확약가능한 갱신도 수행될 수 없으며 대화가 보호되지 않습니다.
  - 3 - 확약가능한 갱신이 수행될 수 있는지 여부를 알 수 없으며 대화는 보호됩니다.
  - 4 - 확약가능한 갱신이 수행될 수 있는지 여부를 알 수 없으며 대화가 보호되지 않습니다.
  - 5 - 확약가능한 갱신이 수행될 수 있는지 여부를 알 수 없으며, 연결이 CONNECT(유형 2)문을 사용하여 구축한 로컬 연결이거나 CONNECT(유형 2)문을 사용하여 어플리케이션 리퀘스터 드라이버 프로그램에 구축한 연결입니다.
- 연결에 대한 추가 정보는 SQLCA 필드의 SQLERRMC에 위치합니다. SQLERRMC 필드에 있는 정보의 설명에 대해서는 부록 B의 ″SQLCA(SQL 통신 영역)″를 참조하십시오.
- 이전의 모든 현재 연결이 비활동 상태로 들어갑니다.

SET CINNECTION문이 성공적으로 수행되지 않으면 활성 그룹의 연결 상태 및 각 연결의 상태가 변하지 않습니다.

주

연결이 사용되고 나서 비활동 상태로 된 다음에 다시 동일한 작업 단위에서 현재 상태로 복원될 때 해당 연결에 대한 잠금, 커서 및 준비된 명령문의 상태가 활성 그룹에 의한 연결의 마지막 사용에 반영됩니다.

현재 독립형 보조 기억장치 풀(IASP) 이름 공간이 로컬 연결의 관계형 데이터베이스와 일치하지 않을 경우, 로컬 연결로의 SET CONNECTION이 실패합니다.

예

TOROLAB1에서 SQL문을 실행하고 TOROLAB2에서 SQL문을 실행한 다음에 TOROLAB1에서 더 많은 SQL문을 실행합니다.

```
EXEC SQL CONNECT TO TOROLAB1;

(TOROLAB1의 오브젝트를 참조하는 명령문 실행)

EXEC SQL CONNECT TO TOROLAB2;

(TOROLAB2의 오브젝트를 참조하는 명령문 실행)

EXEC SQL SET CONNECTION TOROLAB1;

(TOROLAB1의 오브젝트를 참조하는 명령문 실행)
```

첫 번째 CONNECT문은 TOROLAB1 연결을 작성하고, 두 번째 CONNECT문은 작성된 연결을 비활동 상태로 두며, SET CONNECTION문은 연결을 현재 상태로 리턴합니다.

SET OPTION

SET OPTION문은 SQL문에 사용할 처리 옵션을 설정합니다.

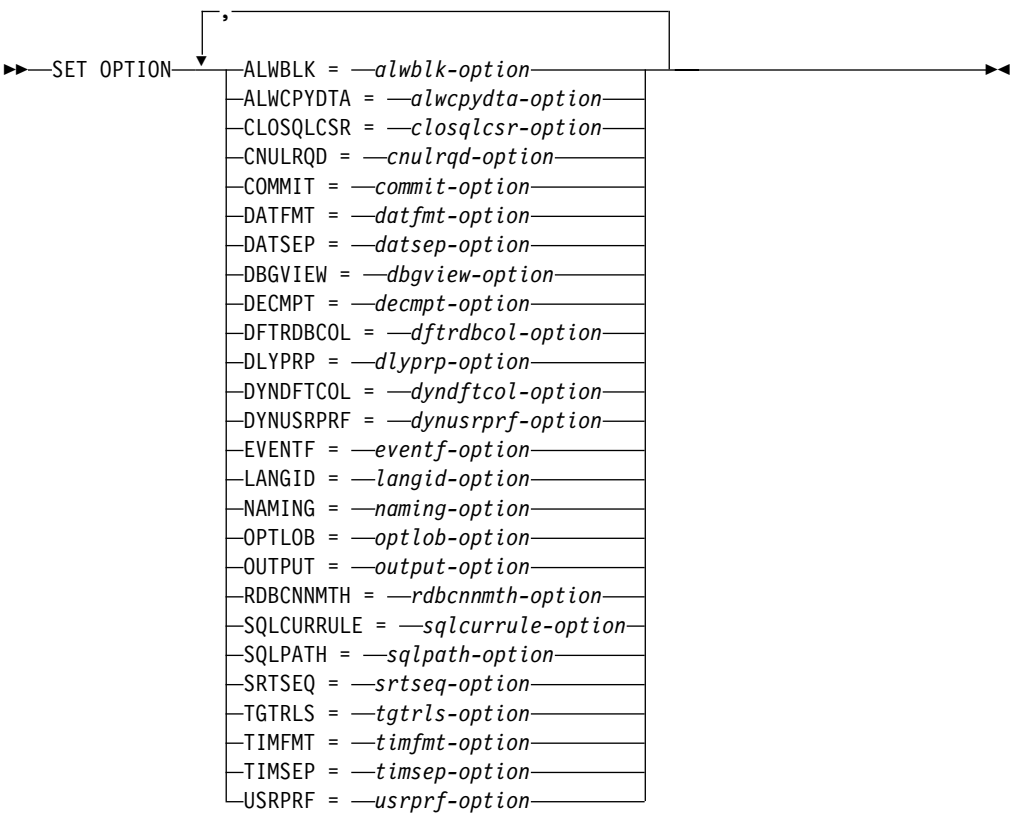
호출

이 명령문은 REXX 프로시듀어에 사용하거나 어플리케이션 프로그램에 삽입할 수 있습니다. REXX 프로시듀어에 사용하는 경우 이 명령문은 실행문입니다. 어플리케이션 프로그램에 삽입하는 경우에는 실행가능하지 않으며 반드시 다른 모든 SQL문 앞에 와야 합니다. 이 명령문은 동적으로 준비될 수 없습니다.

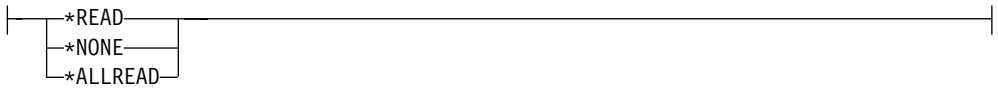
권한부여

필요한 사항이 없습니다.

구문



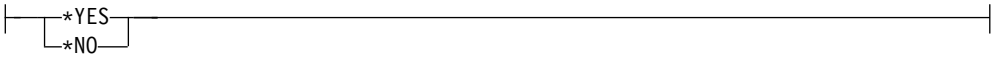
alwblk-option:



alwcpydta-option:

closqlcsr-option:

cnulrqd-option:



commit-option:

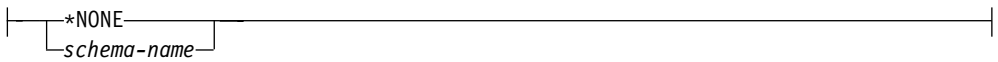
datfmt-option:

datsep-option:

decmpt-option:

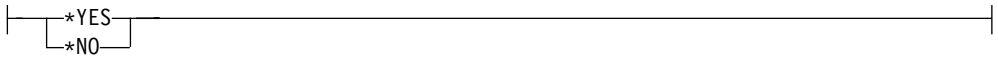
dbgview-option:

dftrdbcol-option:



dlyprp-option:

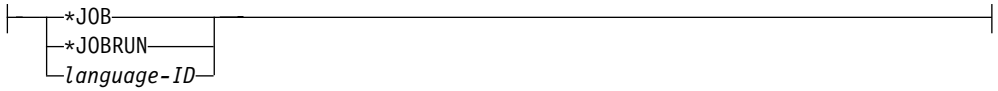
dyndftcol-option:



dynusrprf-option:

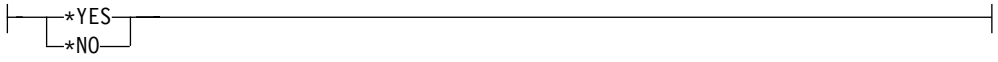
eventf-option:

langid-option:



naming-option:

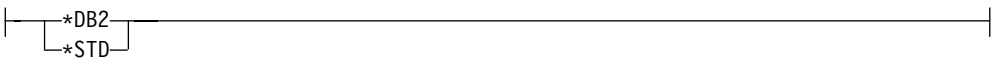
optlob-option:



output-option:

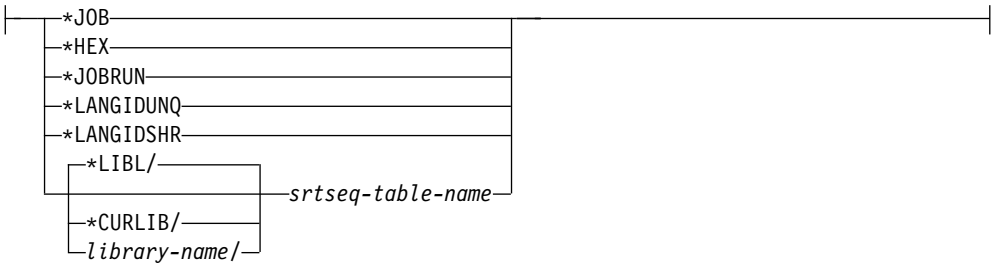
rdbcnnmth-option:

sqlcurrule-option:

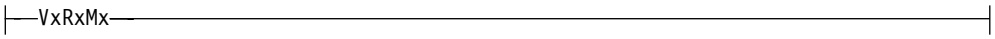


sqlpath-option:

srtseq-option:



tgtrls-option:



timfmt-option:

timsep-option:

usrprf-option:

설명

ALWBLK

데이터베이스 관리자가 행 블록화를 사용할 수 있는지 여부와 읽기 전용 커서에 대해 사용할 수 있는 블록화의 정도를 지정합니다. REXX에서는 이 옵션이 무시됩니다.

***ALLREAD**

COMMIT가 *NONE 또는 *CHG인 경우에 행은 읽기 전용 커서에 대해 블록화됩니다. EXECUTE문이나 EXECUTE IMMEDIATE문이 프로그램에 있는 경우에도, 프로그램에서 명시적으로 갱신될 수 없는 모든 커서가 읽기 전용 처리를 위해 열립니다.

*ALLREAD를 지정하는 경우:

- FRED에 대해 허용되는 블록화 이외에도 확약 제어 레벨 *CHG 아래 행 블록화가 허용됩니다.
- 프로그램에 있는 거의 모든 읽기 전용 커서의 성능이 향상될 수 있지만 다음과 같은 방법의 조회로 제한됩니다.
  - *ALLREAD 옵션이 지정된 경우 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문이 읽기 전용 커서의 위치를 변경하지 않습니다.
  - 커서에 대한 DECLARE문에 FOR UPDATE절이 포함되어 있지 않으면 커서가 위치한 행을 갱신하기 위해 위치지정된 UPDATE문이나 DELETE문의 동적 실행(예를 들어, EXECUTE IMMEDIATE 사용)을 사용할 수 없습니다.

***NONE**

행이 커서에 대한 자료 검색을 위해 블록화되지 않습니다.

*NONE을 지정하는 경우:

- 검색되는 자료가 현재 자료이도록 합니다.
- 조회를 위해 자료의 첫 번째 행을 검색하는 데 걸리는 시간이 줄어들 수 있습니다.
- 데이터베이스 관리자는 조회의 처음 몇 개 행만이 검색되고 나서 조회가 닫혔을 때 프로그램이 사용하지 않은 자료 행 블록의 검색을 중단합니다.
- 많은 수의 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ**

다음과 같은 경우에 행이 커서에 대한 자료의 읽기 전용 검색을 위해 블록화됩니다.

- *NONE를 COMMIT 매개변수에 지정하여 확약 제어를 사용하지 않도록 지정한 경우

- FOR READ ONLY절을 사용하여 커서를 선언하거나 커서에 대해 위치 지정된 UPDATE 또는 DELETE문을 실행할 수 있는 동적 명령문이 없을 경우

*READ를 지정하면 위의 조건을 만족하면서 많은 수의 행을 검색하는 조회의 전반적인 성능을 향상시킬 수 있습니다.

ALWCPYDTA

SELECT문에 자료의 사본을 사용할 수 있는지 여부를 지정합니다. REXX에서는 이 옵션이 무시됩니다.

***OPTIMIZE**

시스템이 데이터베이스에서 직접 검색된 자료를 사용할지 아니면 자료의 사본을 사용할지 여부를 판별합니다. 이 판별은 최상의 성능을 제공하는 방법이 무엇이냐에 좌우됩니다. COMMIT가 *CHG 또는 *CS이고 ALWBLK가 *ALLREAD가 아닌 경우 또는 COMMIT가 *ALL이나 *RR인 경우에는 조회를 실행할 필요가 있을 경우에만 자료의 사본이 사용됩니다.

***YES**

필요할 때만 자료의 사본이 사용됩니다.

***NO**

자료의 사본이 허용되지 않습니다. 조회를 수행하기 위해 자료의 임시 사본이 요구되는 경우 오류 메세지가 리턴됩니다.

CLOSQLCSR

SQL 커서가 암시적으로 닫히는 때 준비된 SQL문이 암시적으로 삭제되는 때 그리고 LOCK TABLE 잠금이 해제되는 때를 지정합니다. SQL 커서는 사용자가 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다. REXX에서는 이 옵션이 무시됩니다. *ENDACTGRP와 *ENDMOD는 ILE(Integrated Language Environment) 프로그램 및 모듈에서 사용하기 위한 것입니다. *ENDPGM, *ENDSQL 및 *ENDJOB은 비ILE 프로그램에서 사용하기 위한 것입니다.

이 옵션은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거에서 허용되지 않습니다.

***ENDACTGRP**

활성 그룹이 종료될 때 SQL 커서가 닫히고, 준비된 SQL문이 암시적으로 삭제되며 LOCK TABLE 잠금이 해제됩니다.

***ENDMOD**

모듈에서 나갈 때 SQL 커서가 닫히고 준비된 SQL문이 암시적으로 삭제됩니다. LOCK TABLE 잠금은 호출 스택의 첫 번째 SQL 프로그램이 종료될 때 해제됩니다.

***ENDPGM**

프로그램이 종료될 때 SQL 커서가 닫히고 준비된 SQL문이 암시적으로 삭제됩니다. LOCK TABLE 잠금은 호출 스택의 첫 번째 SQL 프로그램이 종료될 때 해제됩니다.

***ENDSQL**

SQL 커서가 호출 사이에 열린 상태로 남아 있어 다른 SQL OPEN을 실행하지 않고도 페치될 수 있습니다. 호출 스택에 있는 상위 프로그램 중 하나에서 하나 이상의 SQL문을 실행했어야 합니다. 호출 스택의 첫 번째 SQL 프로그램이 종료될 때 SQL 커서가 닫히고, 준비된 SQL문이 암시적으로 삭제되며 LOCK TABLE 잠금이 해제됩니다. 호출된 첫 번째 SQL 프로그램(호출 스택에서 첫 번째 SQL 프로그램)에 대해 *ENDSQL을 지정하는 경우 프로그램은 *ENDPGM을 지정한 것처럼 처리됩니다.

***ENDJOB**

SQL 커서가 호출 사이에 열린 상태로 남아 있어 다른 SQL OPEN을 실행하지 않고도 페치될 수 있습니다. 호출 스택에 있는 상위 프로그램에서 SQL문을 실행하지 않았어도 됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료될 때 SQL 커서는 열린 상태로 남으며 준비된 SQL이 보존되고 LOCK TABLE 잠금도 보유됩니다. 작업이 종료될 때 SQL 커서가 닫히고 준비된 SQL문이 삭제되며 LOCK TABLE 잠금이 해제됩니다.

CNULRQD

NUL-종료자가 문자 및 그래픽 호스트 변수에 대해 리턴되는지 여부를 지정합니다. 이 옵션은 C 및 C++ 프로그램의 SQL문에만 사용할 수 있습니다.

이 옵션은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거에서 허용되지 않습니다.

***YES**

출력 문자 및 그래픽 호스트 변수에 항상 NUL-종료자가 들어 있습니다. NUL-종료자가 들어가기에 충분한 공간이 없으면, 자료가 절단되고 NUL-종료자가 추가됩니다. 입력 문자 및 그래픽 호스트 변수에 NUL-종료자가 필요합니다.

***NO**

출력 문자 및 그래픽 호스트 변수의 경우 호스트 변수의 길이가 자료의 길이와 정확하게 같을 때 NUL- 종료자가 리턴되지 않습니다. 입력 문자 및 그래픽 호스트 변수에 NUL-종료자가 필요하지 않습니다.

COMMIT

사용할 분리 레벨을 지정합니다. REXX에서는 소스에 참조되는 파일이 이 옵션의 영향을 받지 않습니다. SQL문에 참조되는 표, 뷰 및 패키지만이 영향을 받습니다. 분리 레벨에 대한 자세한 내용은 24 페이지의 『분리 레벨』을 참조하십시오.

***CHG**

비확약 읽기(Uncommitted Read) 분리 레벨을 지정합니다.

***NONE**

확약 안함(No Commit) 분리 레벨을 지정합니다. REXX 프로시듀어에 DROP SCHEMA문이 있는 경우 *NONE을 사용해야 합니다.

***CS**

커서 안전성(Cursor Stability) 분리 레벨을 지정합니다.

***ALL**

읽기 안전성(Read Stability) 분리 레벨을 지정합니다.

***RR**

반복가능 읽기(Repeatable Read) 분리 레벨을 지정합니다.

DATFMT

날짜 결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정한 형식으로 리턴됩니다. 입력 날짜 스트링의 경우에는 지정한 값이 날짜가 유효한 형식으로 지정되는지 판별하는 데 사용됩니다.

주: *USA, *ISO, *EUR 또는 *JIS 형식을 사용하는 입력 날짜 스트링은 항상 유효합니다.

***JOB:**

작업에 지정된 형식을 사용합니다. 작업에 대한 현재 날짜 형식을 판별하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

***ISO**

국제 표준 기구(ISO) 날짜 형식(yyyy-mm-dd)을 사용합니다.

***EUR**

유럽식 날짜 형식(dd.mm.yyyy)을 사용합니다.

***USA**

미국식 날짜 형식(mm/dd/yyyy)을 사용합니다.

***JIS**

일본 산업 표준 날짜 형식(yyyy-mm-dd)을 사용합니다.

***MDY**

날짜 형식(mm/dd/yy)을 사용합니다.

***DMY**

날짜 형식(dd/mm/yy)을 사용합니다.

***YMD**

날짜 형식(yy/mm/dd)을 사용합니다.

***JUL**

율리유스력 형식(yy/ddd)을 사용합니다.

DATSEP

날짜 결과 열에 액세스할 때 사용되는 분리자를 지정합니다.

주: 이 매개변수는 DATFMT 매개변수에 *JOB, *MDY, *DMY, *YMD 또는 *JUL을 지정하는 경우에만 적용됩니다.

***JOB**

작업에 지정된 날짜 분리자를 사용합니다. 작업에 대한 현재 값을 판별하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

***SLASH** 또는 **'/'**

슬래시(/)를 사용합니다.

***PERIOD** 또는 **'.'**

마침표(.)를 사용합니다.

***COMMA** 또는 **','**

쉼표(,)를 사용합니다.

***DASH** 또는 **'-'**

붙임표(-)를 사용합니다.

***BLANK** 또는 **' '**

공백( )을 사용합니다.

DBGVIEW

컴파일러에 의해 디버그 정보 유형이 제공되도록 지정합니다. DBGVIEW 매개변수는 SQL 함수 본문, 프로시듀어 및 트리거에서만 지정할 수 있습니다. 다음과 같은 선택이 가능합니다.

***NONE**

디버그 뷰는 생성되지 않습니다.

***SOURCE**

SQL문을 사용하여 컴파일된 모듈 오브젝트를 디버그하도록 허용합니다.

***STMT**

프로그램 명령문 번호와 기호 식별자를 사용하여 컴파일된 모듈 오브젝트를 디버그하도록 허용합니다.

***LIST**

컴파일된 모듈 오브젝트를 디버깅하기 위해 리스팅 뷰를 생성합니다.

DECMPT

소수점을 표시할 기호를 지정합니다. 다음과 같은 선택이 가능합니다.

***PERIOD**

소수점의 표시가 마침표입니다.

***COMMA**

소수점의 표시가 쉼표입니다.

***SYSVAL**

소수점의 표시가 시스템 값(QDECFMT)입니다.

***JOB**

소수점의 표시가 작업 값(EECFMT)입니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다. REXX에서는 이 옵션이 무시됩니다.

이 옵션은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거에서 허용되지 않습니다.

***NONE**

OPTION 사전컴파일 매개변수에 지정한 명명 규칙이나 SET OPTION NAMING 옵션이 지정한 명명 규칙을 사용합니다.

schema-name

스키마명을 지정합니다. 이 값이 OPTION 사전컴파일 매개변수에 지정한 명명 규칙이나 SET OPTION NAMING 옵션이 지정한 명명 규칙 대신 사용됩니다.

DLYPRP

OPEN, EXECUTE 또는 DESCRIBE문이 실행될 때까지 PREPARE문에 대한 동적 명령문 유효성 검사를 지연할지 여부를 지정합니다. 유효성 검사를 지연하면 여분의 유효성 검사가 제거되어 성능이 향상될 수 있습니다. REXX에서는 이 옵션이 무시됩니다.

***NO**

동적 명령문 유효성 검사를 지연하지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 검사됩니다. 동적 명령문을 OPEN문이나 EXECUTE문에 사용하면 액세스 계획의 유효성이 재검사됩니다. 동적 명령문에 참조되는 오브젝트의 권한이나 오브젝트의 존재 여부가 변경될 수 있기 때문에 OPEN문이나 EXECUTE문을 발행한 다음에는 계속 SQLCODE나 SQLSTATE를 검사하여 동적 명령문이 아직도 유효한지 확인해야 합니다.

***YES**

OPEN, EXECUTE 또는 DESCRIBE SQL문에 동적 명령문이 사용될 때까지 동적 명령문 유효성 검사를 지연합니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 구축됩니다. *YES를 지정하는 경우 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 다음에는 SQLCODE와 SQLSTATE를 검사하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정할 경우 PREPARE문에 INTO절을 사용하거나 동적 명령문에 OPEN을 발행하기 전에 DESCRIBE문이 그 명령문을 사용하면 성능이 향상되지 않습니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정한 콜렉션명을 동적 명령문에 대해서도 사용하도록 지정합니다. REXX에서는 이 옵션이 무시됩니다.

이 옵션은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거에서 허용되지 않습니다.

***NO**

DFTRDBCOL에 대해 지정한 값을 동적 SQL문의 표, 뷰, 색인 및 SQL 패키지의 규정화되지 않은 이름에 사용하지 마십시오. OPTION 사전컴파일 매개변수에 지정한 명명 규칙이나 SET OPTION NAMING 옵션이 지정한 명명 규칙을 사용합니다.

***YES**

DFTRDBCOL에 대해 지정한 스키마명이 동적 SQL문의 표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용됩니다.

DYNUSRPRF

동적 SQL문에 지정할 사용자 프로파일을 지정합니다. REXX에서는 이 옵션이 무시됩니다.

***USER**

로컬 동적 SQL문이 작업에 대한 사용자 프로파일 아래 실행됩니다. 분배된 동적 SQL문이 서버 작업의 사용자 프로파일 아래에서 실행됩니다.

***OWNER**

로컬 동적 SQL문이 프로그램 소유자의 사용자 프로파일 아래 실행됩니다. 분배된 동적 SQL문이 SQL 패키지 소유자의 사용자 프로파일 아래 실행됩니다.

EVENTF

이벤트 파일이 생성되었는지 여부를 지정합니다. CoOperative Development Environment/400(CODE/400)은 이벤트 파일을 사용하여 CODE/400 편집기와 통합된 오류 피드백을 제공합니다.

***YES**

컴파일러는 CoOperative Development Environment/400(CODE/400)이 사용할 이벤트 파일을 생성합니다.

***NO**

컴파일러는 CoOperative Development Environment/400(CODE/400)이 사용할 이벤트 파일을 생성하지 않습니다.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR)를 지정할 때 사용할 언어 ID를 지정합니다.

***JOB** 또는 ***JOBRUN**

작업의 LANGID 값을 사용합니다.

분배된 어플리케이션의 경우 SRTSEQ(*JOBRUN)도 지정해야 LANGID(*JOBRUN)가 유효합니다.

language-id

사용할 언어 ID를 지정합니다. 언어 식별자에 사용할 수 있는 값에 대한 정보는 iSeries Information Center의 언어 식별자 주제를 참조하십시오.

NAMING

SQL 명명 규칙과 시스템 명명 규칙 중 어느쪽을 사용할지 여부를 지정합니다. 이 옵션은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거에서 허용되지 않습니다.

다음과 같은 선택이 가능합니다.

***SYS**

시스템 명명 규칙을 사용합니다.

***SQL**

SQL 명명 규칙을 사용합니다.

OPTLOB

DRDA를 통해 액세스할 때 LOB에의 액세스를 최적화할 수 있는지 여부를 지정합니다. 다음과 같은 선택이 가능합니다.

***YES**

LOB 액세스가 최적화되어야 합니다. 커서에 대한 첫 번째 FETCH가 모든 후속 FETCH에서 LOB에 대해 커서가 사용되는 방법을 결정합니다. 이 옵션은 커서가 닫힐 때까지 유효하게 남아 있습니다.

첫 번째 FETCH가 LOB 로케이터를 사용하여 LOB 열에 액세스하는 경우 해당 커서에 대한 어떠한 후속 FETCH도 해당 LOB 열을 LOB 호스트 변수에 페치할 수 없습니다.

첫 번째 FETCH가 LOB 열을 LOB 호스트 변수에 넣는 경우 해당 커서에 대한 어떠한 후속 FETCH도 해당 열에 대해 LOB 로케이터를 사용할 수 없습니다.

***NO**

LOB 액세스가 최적화되지 않아야 합니다. 열을 LOB 로케이터로 검색할지 또는 LOB 호스트 변수로 검색할지에 대한 제한사항은 없습니다. 이 옵션은 성능 저하를 유발할 수 있습니다.

OUTPUT

사전컴파일러 및 컴파일러 리스팅의 생성 여부를 지정합니다. OUTPUT 매개변수는 SQL 함수 본문, 프로시듀어 및 트리거에서만 지정할 수 있습니다. 다음과 같은 선택이 가능합니다.

***NONE**

사전컴파일러 및 컴파일러 리스팅이 생성되지 않습니다.

***PRINT**

사전컴파일러 및 컴파일러 리스팅이 생성됩니다.

RDBCNNMTH

CONNECT문에 사용하는 구문을 지정합니다. REXX에서는 이 옵션이 무시됩니다.

***DUW**

분배된 작업 단위를 지원하는 데 CONNECT(유형 2) 구문을 사용합니다. 추가 관계형 데이터베이스(RDB)에 대한 연속 CONNECT문이 이전 연결을 단절시키지 않습니다.

***RUW**

리모트 작업 단위를 지원하는 데 CONNECT(유형 1) 구문을 사용합니다. 연속 CONNECT문은 새로운 연결이 구축되기 전에 이전 연결을 단절시킵니다.

SQLCURRULE

SQL문에 사용되는 구문을 지정합니다.

***DB2**

모든 SQL문의 구문이 DB2에 대해 설정된 규칙으로 디폴트 설정됩니다. 다음 구문이 이 옵션에 의해 제어됩니다.

- 16진 상수는 문자 자료로 취급됩니다.

***STD**

모든 SQL문의 구문이 ISO 및 ANSI SQL 표준에 의해 설정된 규칙으로 디폴트 설정됩니다. 다음 구문이 이 옵션에 의해 제어됩니다.

- 16진 상수는 2진 자료로 취급됩니다.

SQLPATH

정적 SQL문의 프로시듀어, 함수 및 사용자 정의 유형을 찾는 데 사용할 경로를 지정합니다. REXX에서는 이 옵션이 무시됩니다.

***LIBL**

사용하는 경로는 실행시 라이브러리 리스트입니다.

character-string

쉼표로 분리한 하나 이상의 스키마명으로 된 문자 상수

SRTSEQ

SQL문의 스트링 비교에 사용할 정렬 순서 표를 지정합니다.

주: REXX 프로시듀어를 iSeries용 DB2 UDB 또는 릴리스 레벨 V2R3M0 이전의 iSeries 시스템이 아닌 서버에 연결하는 경우 *HEX를 지정해야 합니다.

***JOB** 또는 ***JOBRUN**
작업의 SRTSEQ 값을 사용합니다.

***HEX**
정렬 순서 표를 사용하지 않습니다. 문자의 16진 값이 정렬 순서를 판별하는 데 사용됩니다.

***LANGIDUNQ**
정렬 순서 표에 코드 페이지의 각 문자에 대한 고유한 가중치가 있어야 합니다.

***LANGIDSHR**
지정한 LANGID에 대한 공유 가중치 정렬표를 사용합니다.

srtseq-table-name
이 프로그램과 함께 사용할 정렬 순서 표 이름을 지정합니다. 정렬 순서 표의 이름은 다음 라이브러리 값 중 하나로 규정화될 수 있습니다.

***LIBL**
작업의 라이브러리 리스트에서 사용자 및 시스템 부분에 있는 모든 라이브러리가 첫 번째 일치를 찾을 때까지 탐색됩니다.

***CURLIB**
작업의 현재 라이브러리를 탐색합니다. 작업의 현재 라이브러리로서 지정된 라이브러리가 없을 경우 QGPL 라이브러리를 사용합니다.

library-name
탐색할 라이브러리명을 지정하십시오.

TGTRLS
사용자가 작성된 오브젝트를 사용하려는 오퍼레이팅 시스템의 릴리스를 지정합니다. TGTRLS 매개변수는 SQL 함수 본문, 프로시듀어 및 트리거에서만 지정할 수 있습니다. 다음과 같은 선택이 가능합니다.

VxRxMx
릴리스를 VxRxMx 형식으로 지정합니다. Vx는 버전, Rx는 릴리스 및 Mx는 수정 레벨입니다. 예를 들어 V5R1M0는 버전 5 릴리스 1 수정 레벨 0입니다. 이 오브젝트는 지정된 릴리스 또는 설치된 오퍼레이팅 시스템의 후속 릴리스가 설치되어 있는 시스템에서 사용할 수 있습니다.

유효한 값은 현재의 버전, 릴리스 및 수정 레벨에 따라 다르고 각각의 새로운 릴리스에 따라 변합니다. 데이터베이스 관리자가 지원하는 가장 이전의 릴리스 레벨보다 더 이전의 릴리스 레벨을 지정하면 지원할 수 있는 가장 이전의 릴리스를 알려주는 오류 메세지가 송신됩니다.

TGTRLS 옵션은 SQL 함수, SQL 프로시듀어 및 트리거에서만 지정할 수 있습니다.

TIMFMT

시간 결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 시간 필드가 지정한 형식으로 리턴됩니다. 입력 시간 스트링의 경우 지정한 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: *USA, *ISO, *EUR 또는 *JIS 형식을 사용하는 입력 시간 스트링은 항상 유효합니다.

***HMS**

(hh:mm:ss) 형식을 사용합니다.

***ISO**

국제 표준 기구(ISO) 시간 형식(hh.mm.ss)을 사용합니다.

***EUR**

유럽식 시간 형식(hh.mm.ss)을 사용합니다.

***USA**

미국식 시간 형식(hh:mm *xx*)을 사용합니다. 여기서 *xx*는 AM 또는 PM입니다.

***JIS**

일본 산업 표준 시간 형식(hh:mm:ss)을 사용합니다.

TIMSEP

시간 결과 열에 액세스할 때 사용되는 분리자를 지정합니다.

주: 이 매개변수는 TIMFMT 매개변수에 *HMS를 지정한 경우에만 적용됩니다.

***JOB**

작업에 지정된 시간 분리자를 사용합니다. 작업에 대한 현재 값을 판별하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

***COLON** 또는 **':'**

콜론(:)을 사용합니다.

***PERIOD** 또는 **'.'**

마침표(.)를 사용합니다.

***COMMA** 또는 **','**

쉼표(,)를 사용합니다.

***BLANK** 또는 **' '**

공백( )을 사용합니다.

USRPRF

컴파일된 프로그램 오브젝트를 실행할 때 사용하는 사용자 프로파일(해당 프로그램

오브젝트가 정적 SQL문에서 각 오브젝트에 대해 갖는 권한 포함)을 지정합니다. 프로그램 소유자나 프로그램 사용자의 프로파일이 프로그램 오브젝트에서 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다. REXX에서는 이 옵션이 무시됩니다.

***NAMING**
명명 규칙에 따라 사용자 프로파일이 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER**
프로그램 오브젝트를 실행하는 사용자의 프로파일을 사용합니다.

***OWNER**
프로그램을 실행할 때 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일을 사용합니다.

주

REXX 프로시듀어를 시작할 때 옵션이 각각의 디폴트 값으로 설정됩니다. 각 옵션의 디폴트 값은 구문 도표에 나열된 첫 번째 값입니다. SET OPTION문으로 옵션을 변경하는 경우 옵션을 다시 변경하거나 또는 REXX 프로스듀어가 종료될 때까지 신규 값은 유효하게 남아 있습니다.

어플리케이션 프로그램의 경우 처리 옵션은 초기에 CRTSQLxxx 명령에서 지정된 값으로 설정됩니다. 각 옵션은 SET OPTION문에서 나타날 때 갱신됩니다. 모든 SET OPTION문은 삽입된 다른 모든 SQL문 앞에 와야 합니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- *UR을 *CHG의 동의어로 사용할 수 있습니다.
- *NC를 *NONE의 동의어로 사용할 수 있습니다.
- *RS를 *ALL의 동의어로 사용할 수 있습니다.

예

예 1: 분리 레벨을 *ALL로 설정하고 명명 모드는 SQL 이름으로 설정합니다.

```
EXEC SQL SET OPTION COMMIT =*ALL, NAMING =*SQL
```

예 2: 날짜 형식은 유럽식으로, 분리 레벨은 *CS로, 소수점은 쉼표로 설정합니다.

```
EXEC SQL SET OPTION DATFMT = *EUR, COMMIT = *CS, DECMPT = *COMMA
```

SET PATH

SET PATH문은 CURRENT PATH 특수 레지스터의 값을 변경합니다.

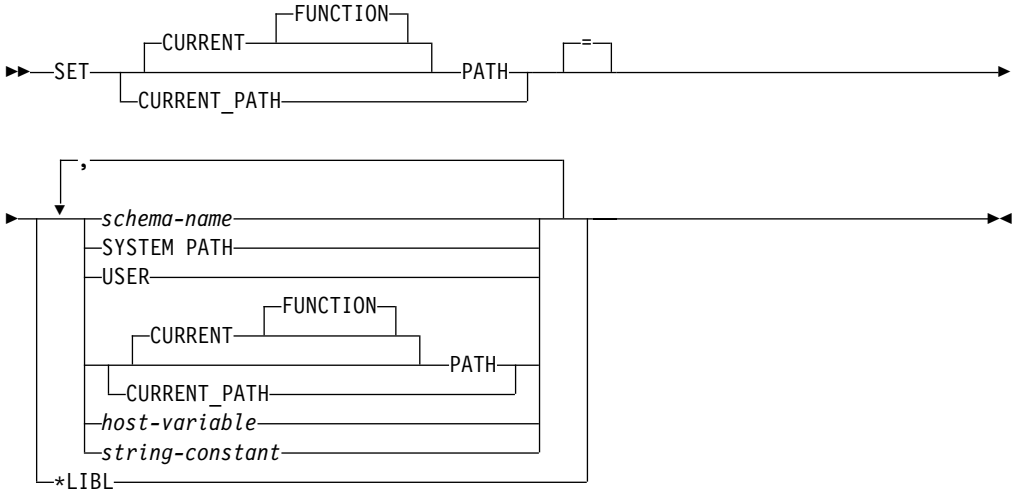
호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

이 명령문을 실행하는 데 필요한 권한이 없습니다.

구문



설명

CURRENT PATH 특수 레지스터의 값을 지정한 값으로 대체합니다.

schema-name

스키마를 식별합니다. 경로 설정시 스키마가 존재하는지에 대한 검증은 하지 않습니다.

SYSTEM PATH

이 값은 스키마명 ″QSYS″,″QSYS2″를 지정하는 것과 같습니다.

USER

이 값은 USER 특수 레지스터입니다.

CURRENT PATH

이 명령문을 실행하기 전의 CURRENT PATH 특수 레지스터의 값

host-variable

쉼표로 분리된 하나 이상의 스키마명이 있는 호스트 변수

호스트 변수에는 다음이 적용됩니다.

- 문자 스트링 변수이어야 합니다.
- 인디케이터 변수가 뒤에 나오지 않아야 합니다.
- 왼쪽 정렬인 스키마를 포함해야 하며, 일반 ID 지정 규칙을 따라야 합니다.
- 오른쪽은 공백으로 채워야 합니다.
- 널값이 아니어야 합니다.

string-constant
쉼표로 분리한 하나 이상의 스키마명으로 된 문자 상수

주

경로에 한 스키마명이 여러 번 나타날 수 없습니다.

SET PATH문은 확약가능한 조작이 아닙니다. ROLLBACK이 CURRENT PATH에 영향을 주지 않습니다.

지정할 수 있는 스키마의 수는 CURRENT PATH 특수 레지스터의 총 길이에 따라 제한됩니다. 특수 레지스터 스트링은, 지정한 각 스키마(schema)의 이름을 취하여 후미 공백을 제거하고 큰 따옴표로 묶은 다음에 각 스키마명을 쉼표로 분리함으로써 구축됩니다. 결과로 생성되는 스트링의 길이가 3483바이트를 넘으면 오류가 리턴됩니다. 최대 268개의 스키마명이 경로에 표시될 수 있습니다.

활성 그룹에서 실행된 첫 번째 SQL문에 시스템 명명을 사용한 경우 CURRENT PATH 특수 레지스터의 초기값은 *LIBL입니다. 첫 번째 SQL문에 SQL 명명을 사용하였으면, 초기 값이 ″QSYS″,″QSYS2″, ″X″(여기서, X는 USER 특수 레지스터의 값)입니다.

스키마 QSYS와 QSYS2는 지정하지 않아도 됩니다. 이 두 스키마를 경로에 포함하지 않으면 첫 번째 스키마로 가정합니다(이 경우에서는 CURRENT PATH 특수 레지스터에 포함되지 않음).

동적 SQL문에 있는 사용자 정의 고유한 유형 및 함수를 해결하는 데 CURRENT PATH 특수 레지스터가 사용됩니다. 자세한 내용은 56 페이지의 『스키마 및 SQL 경로』를 참조하십시오.

예

다음 명령문은 CURRENT PATH 특수 레지스터를 설정합니다.

```
SET PATH = FUNC_XYZ, "NewFun98", QSYS2
```

SET RESULT SETS

SET RESULT SETS문은 iSeries Access 클라이언트나 SQL 호출 레벨 인터페이스에 의해 외부 프로시듀어가 호출될 때 또는 DRDA를 사용하는 리모트 시스템에서 액세스할 때 프로시듀어로부터 리턴될 수 있는 하나 이상의 결과 세트를 식별합니다.

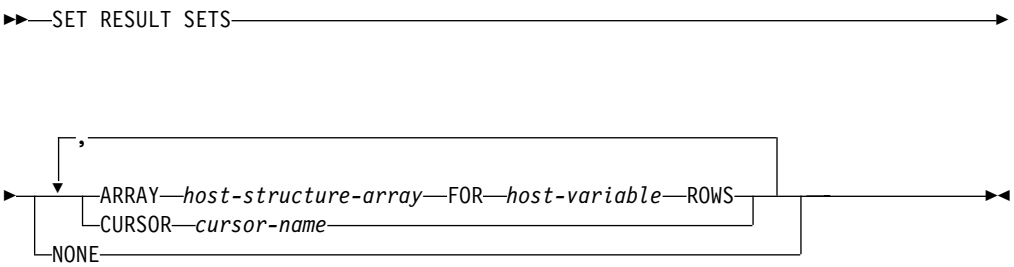
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. Java 또는 REXX 프로시듀어에는 이 명령문이 허용되지 않습니다.

권한부여

필요한 사항이 없습니다.

구문



설명

CURSOR *cursor-name*

프로시듀어로부터 리턴될 수 있는 결과 세트를 정의하는 데 사용할 커서를 식별합니다. *cursor-name*은 599 페이지의 『설명』에서 DECLARE CURSOR문에 대해 설명한 대로 선언된 커서를 식별해야 합니다. SET RESULT SETS문을 실행할 때 커서는 열린 상태에 있어야 합니다.

ARRAY *host-structure-array*

*host-structure-array*는 호스트 구조 선언 규칙에 따라 정의되는 호스트 구조의 배열을 식별합니다. 배열에 C NUL-종료 호스트 변수가 있을 수 없습니다.

배열의 첫 번째 구조가 결과 세트의 첫 번째 행에 대응되고, 배열의 두 번째 구조는 결과 세트의 두 번째 행에 대응되는 방법으로 배열의 구조와 결과 세트의 행이 대응됩니다. 또한 행의 첫 번째 값은 구조의 첫 번째 항목에 대응되고, 행의 두 번째 값은 구조의 두 번째 항목에 대응되는 방법으로 행의 값과 구조의 항목이 대응됩니다.

LOB는 DRDA를 사용할 때 배열에 리턴될 수 없습니다.

SET RESULT SETS문에 하나의 배열만을 지정할 수 있습니다.

FOR *host-variable* **ROWS**

결과 세트의 행 수를 지정합니다. *host-variable*은 0 배율의 숫자 호스트 변수이어야 하며, 인디케이터 변수를 포함하지 않아야 합니다. 행 수는 0 - 32767 범위에 있고 호스트 구조 배열의 차원 이하이어야 합니다.

NONE

리턴되는 결과 세트가 없도록 지정합니다. 프로시듀어가 종료될 때 열린 상태로 있던 커서는 리턴되지 않습니다.

주

결과 세트는 iSeries Access ODBC(개방 데이터베이스 연결성) 드라이버를 사용하는 클라이언트, iSeries Access Optimized SQL API를 사용하는 클라이언트 또는 SQL 호출 레벨 인터페이스나 JDBC로부터 프로시듀어가 호출될 때 해당 프로시듀어에서만 리턴됩니다. 또한 결과 세트는 분산 관계형 데이터베이스 구조(DRDA)를 사용하여 비 iSeries 클라이언트가 iSeries 서버에 액세스할 때마다 리턴됩니다.

외부 프로시듀어: 결과 세트가 외부 프로시듀어로부터 리턴되는 방법에는 다음의 세 가지가 있습니다.

- 프로시듀어에서 SET RESULT SETS문을 실행할 때 SET RESULT SETS문이 결과 세트를 식별합니다. 결과 세트는 SET RESULT SETS문에 지정한 순서로 리턴됩니다.
- SET RESULT SETS문이 프로시듀어에서 실행되지 않는 경우
  - WITH RETURN절을 지정한 커서가 없는 경우 프로시듀어가 열고 리턴시 열린 상태로 그대로 놓아 둔 각 커서가 결과 세트를 식별합니다. 결과 세트는 커서가 열린 순서로 리턴됩니다.
  - WITH RETURN절을 지정한 커서가 있는 경우 프로시듀어가 열고 리턴시 열린 상태로 그대로 놓아 둔 WITH RETURN절을 사용하여 정의된 각 커서가 결과 세트를 식별합니다. 결과 세트는 커서가 열린 순서로 리턴됩니다.

열린 커서를 사용하여 결과 세트가 리턴될 때 행은 현재 커서 위치에서 시작하여 리턴됩니다.

프로시저로부터 결과 세트를 리턴하려면 RESULT SETS절을 CREATE PROCEDURE(외부)문 또는 DECLARE PROCEDURE문에 지정해야 합니다. 리턴되는 최대 결과 세트 수는 CREATE PROCEDURE(외부)문 또는 DECLARE PROCEDURE문에 지정되는 수보다 커서는 안됩니다.

SQL 프로시듀어: SQL 프로시듀어로부터 결과 세트를 리턴하려면, RESULT SETS절을 사용하여 프로시듀어를 작성해야 합니다. 프로시듀어가 열고 리턴시 열린 상태로 그대로 놓아 둔 WITH RETURN절을 사용하여 정의된 각 커서가 결과 세트를 식별합니다.

- 프로시듀어에서 SET RESULT SETS문을 실행할 때 SET RESULT SETS문이 리턴되는 결과 세트를 식별합니다. 결과 세트는 SET RESULT SETS문에 지정한 순서로 리턴됩니다.
- 프로시듀어에서 SET RESULT SETS문이 실행되지 않을 때 커서가 열린 순서대로 결과 세트를 리턴합니다.

열린 커서를 사용하여 결과 세트가 리턴될 때 행은 현재 커서 위치에서 시작하여 리턴됩니다.

SQL 프로시듀어로부터 결과 세트를 리턴하려면 RESULT SETS절을 CREATE PROCEDURE(외부)문에 지정해야 합니다. 리턴되는 최대 결과 세트 수는 CREATE PROCEDURE문에 지정되는 수보다 커서는 안됩니다.

예

다음 SET RESULT SETS문은 커서 X를 프로시듀어가 호출될 때 리턴될 결과 세트로서 지정합니다. ODBC 클라이언트로부터의 결과 세트 사용을 보여주는 자세한 정보 및 예는 iSeries Information Center에서 iSeries Access 범주를 참조하십시오.

```
EXEC SQL SET RESULT SETS CURSOR X;
```

SET SCHEMA

SET SCHEMA문은 CURRENT SCHEMA 특수 레지스터의 값을 변경합니다.

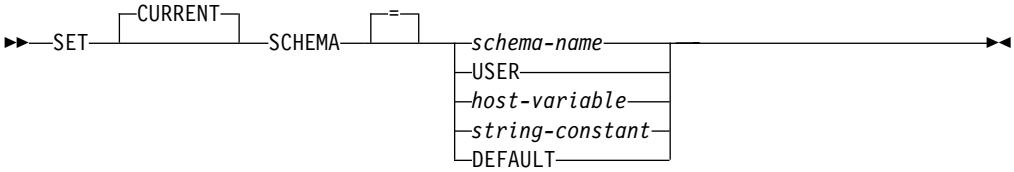
호출

이 명령문은 대화식으로 발행되거나 어플리케이션 프로그램에 삽입될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

이 명령문을 실행하는 데 필요한 권한이 없습니다.

구문



설명

CURRENT SCHEMA 특수 레지스터의 값을 지정한 값으로 대체합니다.

schema-name

스키마를 식별합니다. CURRENT SCHEMA 설정시 스키마가 존재하는지에 대한 검증은 하지 않습니다.

USER

이 값은 USER 특수 레지스터입니다.

host-variable

스키마명이 들어 있는 호스트 변수

호스트 변수에는 다음이 적용됩니다.

- 문자 스트링 변수이어야 합니다.
- 인디케이터 변수가 뒤에 나오지 않아야 합니다.
- 왼쪽 정렬인 스키마를 포함해야 하며, 일반 ID 지정 규칙을 따라야 합니다.
- 오른쪽은 공백으로 채워야 합니다.
- 널값이 아니어야 합니다.

string-constant

스키마 명으로 문자 상수

DEFAULT
CURRENT SCHEMA는 초기 값으로 설정됩니다. SQL 명명의 초기값은 USER입니다. 시스템 명명의 초기값은 *LIBL입니다.

주

CURRENT SCHEMA 특수 레지스터의 값은 DYNDFTCOL이 지정된 프로그램의 경우를 제외하고 모든 동적 SQL문의 규정되지 않은 모든 이름에 대한 규정자로 사용됩니다. DYNDFTCOL이 프로그램에 지정되면, CURRENT SCHEMA 스키마명이 아닌 해당 스키마명이 사용됩니다.

SET SCHEMA문은 확약가능한 조작이 아닙니다. ROLLBACK이 CURRENT SCHEMA에 영향을 주지 않습니다.

SQL명명에서 CURRENT SCHEMA 특수 레지스터의 초기값은 USER와 동등합니다. 시스템 명명에서 CURRENT SCHEMA 특수 레지스터의 초기값은 '*LIBL'입니다.

CURRENT SCHEMA 특수 레지스터를 설정하는 것이 CURRENT PATH 특수 레지스터에 영향을 주지 않습니다. 따라서, CURRENT SCHEMA는 SQL 경로, 함수, 프로시듀어에 포함되지 않으며, 고유한 유형 분석은 해당 오브젝트를 찾지 않습니다. SQL 경로에 현재 스키마값을 포함시키려면, SET SCHEMA문이 실행될 때마다 SET SCHEMA문의 스키마명을 포함한 SET PATH문도 실행하십시오.

CURRENT SQLID는 CURRENT SCHEMA의 동의어로 받아들여지며 SET CURRENT SQLID문의 효과는 SET CURRENT SCHEMA문의 효과와 동일합니다. 명령문 권한 변경과 같은 다른 효과는 발생하지 않습니다.

QSQCHGDC API 호출에 해당하는 SET SCHEMA.

예

예 1

다음 명령문은 CURRENT SCHEMA 특수 레지스터를 설정합니다.

```
SET SCHEMA = RICK
```

예 2

다음 예는 CURRENT SCHEMA 특수 레지스터의 현재 값을 CURSCHEMA 호스트 변수로 검색합니다.

```
EXEC SQL VALUES(CURRENT SCHEMA) INTO :CURSCHEMA
```

값은 앞의 예에서 설정된 RICK입니다.

SET TRANSACTION

SET TRANSACTION문은 현재 작업 단위에 대한 분리 레벨과 읽기 전용 속성을 설정합니다.

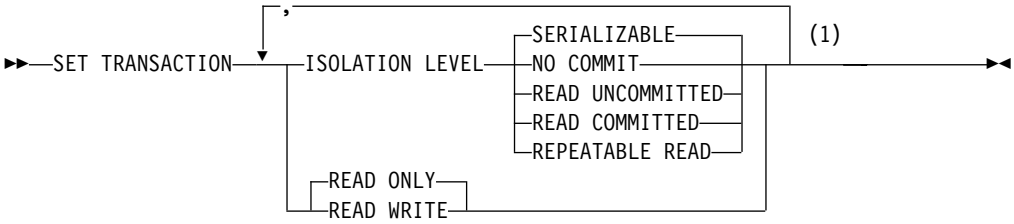
호출

이 명령문은 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

필요한 사항이 없습니다.

구문



주:

1 하나의 ISOLATION LEVEL절 및 하나의 READ WRITE 또는 READ ONLY 절만을 지정할 수 있습니다.

설명

ISOLATION LEVEL

트랜잭션 분리 레벨을 지정합니다. ISOLATION LEVEL절이 지정되지 않은 경우, ISOLATION LEVEL SERIALIZABLE은 내재적입니다.

NO COMMIT

분리 레벨 NC(COMMIT(*NONE))를 지정합니다.

READ UNCOMMITTED

분리 레벨 UR(COMMIT(*CHG))을 지정합니다.

READ COMMITTED

분리 레벨 CS(COMMIT(*CS))를 지정합니다.

REPEATABLE READ[66]
분리 레벨 RS(COMMIT(*ALL))를 지정합니다.

SERIALIZABLE
분리 레벨 RR(COMMIT(*RR))을 지정합니다.

READ WRITE 또는 READ ONLY
트랜잭션의 자료 변경 연산 여부를 지정합니다.

READ WRITE
모든 SQL 연산 허용 여부를 지정합니다. ISOLATION LEVEL READ UNCOMMITTED가 지정되지 않은 경우 이 값이 디폴트 값입니다.

READ ONLY
SQL 자료를 변경하지 않는 SQL 조작망 허용되도록 지정합니다. ISOLATION LEVEL READ UNCOMMITTED가 지정된 경우, 이 값이 디폴트 값입니다.

주

SET TRANSACTION문은 프로세스의 현재 활성 그룹에 대한 SQL문의 분리 레벨을 설정합니다. 그 활성 그룹이 작업 범위에 있는 확약 제어를 갖는 경우 SET TRANSACTION문은 작업 확약 범위의 다른 모든 활성 그룹의 분리 레벨을 설정합니다.

SET TRANSACTION문은 작업 단위의 첫 번째 SQL문일 경우에만 실행할 수 있습니다. 단, 트리거에서 실행할 때는 제외됩니다. 트리거 프로그램에서 SET TRANSACTION은 COMMIT 경계에서만 READ ONLY로 설정할 수 있습니다. SET TRANSACTION문은 트리거에서 언제든지 실행할 수 있지만, 트리거의 첫 번째 명령문으로서 실행하는 것이 바람직합니다. SET TRANSACTION문은 트리거에서 트리거 프로그램의 SQL문에 대한 분리 레벨을 트리거 프로그램의 개시를 유발한 어플리케이션과 동일한 레벨로 설정하는 데 유용한 명령문입니다.

현재 연결이 리모트 서버로 되어 있을 경우 SET TRANSACTION문이 현재 서버의 트리거 프로그램에 있지 않으면 허용되지 않습니다. 일단 SET TRANSACTION문을 실행하고 나면, 그 작업 단위가 확약되거나 롤백될 때까지 CONNECT문과 SET CONNECTION문은 허용되지 않습니다.

SET TRANSACTION문의 범위는 명령문이 실행되는 내용에 좌우됩니다. SET TRANSACTION문이 트리거 프로그램에서 실행되는 경우 지정한 분리 레벨은 다른 SET TRANSACTION문이 발생하거나 트리거 프로그램이 종료될 때까지(둘 중 먼저 발생하는 것), 모든 후속 SQL문에 적용됩니다. SET TRANSACTION문이 트리거 프로그램 밖에서 실행되는 경우 지정한 분리 레벨은 COMMIT나 ROLLBACK 조작이 일어

66. REPEATABLE READ는 iSeries용 DB2 UDB에 대한 분리 레벨 *ALL과 IBM SQL에서의 분리 레벨 RS(읽기 안전성)에 해당하는 ISO와 ANS 표준 용어입니다. IBM SQL이 RR(반복가능 읽기)라고 하는 것에 대해 ISO 및 ANS 표준에서는 SERIALIZABLE을 사용합니다.

날 때까지 모든 후속 SQL(트리거 안에서 SET TRANSACTION문 이 실행된 후 트리거 안의 모든 명령문 제외)문에 적용됩니다.

SET TRANSACTION문은 실행될 때 열려 있는 WITH HOLD 커서에 대하여 아무런 영향을 미치지 않습니다.

분리 레벨에 대한 자세한 정보는 24 페이지의 『분리 레벨』을 참조하십시오.

키워드 동의어

다음 키워드는 이전 릴리스와의 호환을 위해 지원되는 동의어입니다. 이 키워드는 표준이 아니고 사용될 수 없습니다.

- 키워드 NC 또는 NONE을 NO COMMIT의 동의어로 사용할 수 있습니다.
- 키워드 UR 및 CHG를 READ UNCOMMITTED의 동의어로 사용할 수 있습니다.
- 키워드 CS를 READ COMMITTED의 동의어로 사용할 수 있습니다.
- 키워드 RS 또는 ALL을 REPEATABLE READ의 동의어로 사용할 수 있습니다.
- 키워드 RR을 SERIALIZABLE의 동의어로 사용할 수 있습니다.

예

예 1

다음 SET TRANSACTION문은 분리 레벨을 NONE으로 설정합니다(SQL 사전컴파일러 명령에 *NONE을 지정하는 것과 동일).

```
EXEC SQL SET TRANSACTION ISOLATION LEVEL NO COMMIT;
```

예 2

다음 SET TRANSACTION문은 분리 레벨을 SERIALIZABLE로 설정합니다.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

SET 이전 변수

SET transition-variable문은 새 *transition-variable*에 값을 지정합니다.

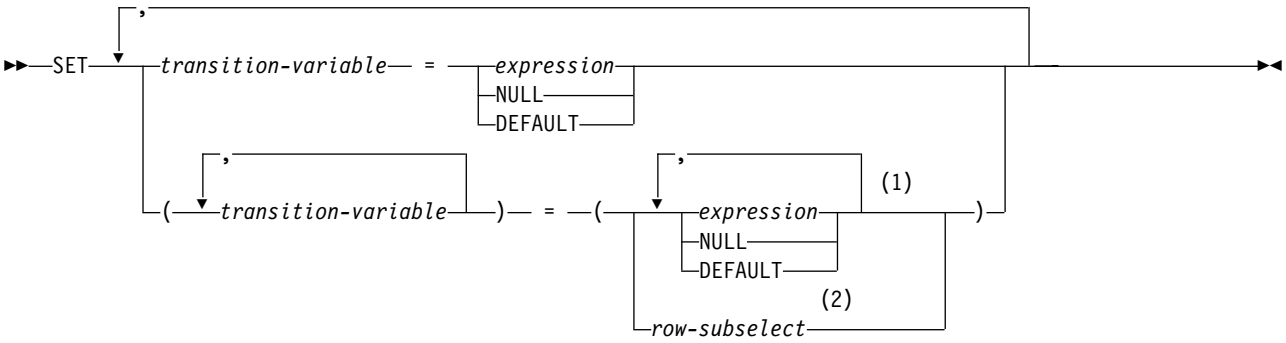
호출

이 명령문은 BEFORE 트리거 안의 SQL문으로만 사용될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

*row-subselect*를 지정하는 경우 각 subselect에 필요한 권한부여에 대해서는 335 페이지의 제 4 장 『조회』를 참조하십시오.

구문



주:

1 *표현식*의 수, NULL 및 DEFAULT는 *transition-variable*의 수와 일치해야 합니다.

2 선택 리스트에 있는 열의 수는 *전이 변수*의 수와 일치해야 합니다.

설명

transition-variable

새 열에서 갱신되는 열을 식별합니다. *transition-variable*는 새 값을 나타내는 상관명으로 선택적으로 규정된 트리거 주제 표의 열을 지정해야 합니다. OLD *transition-variable*은 지정할 수 없습니다.

각 *전이 변수*의 자료 유형은 대응되는 결과 열과 호환될 수 있어야 합니다. 값을 열에 대한 지정 규칙에 따라 *transition-variable*에 지정됩니다. 자세한 내용은 80 페이지의 『지정과 비교』를 참조하십시오.

expression

*전이 변수*의 신규 값을 지정합니다. *expression*은 127 페이지의 『표현식』에서 설명한 유형의 모든 표현식입니다. 표현식은 열 함수를 표함하지 않습니다.

*expression*에는 OLD 및 NEW *transition-variable*에 대한 언급이 포함될 수 있습니다. CREATE TRIGGER문에 OLD 및 NEW 절이 모두 들어 있는 경우, *transition-variable*에 대한 언급은 *correlation-name*으로 규정되어 *transition-variable*을 지정해야 합니다.

NULL

널값을 지정합니다. NULL은 널가능 열에 대해서만 지정될수 있습니다.

DEFAULT

*전이 변수*와 연관된 열에 대한 디폴트 값이 사용될 것임을 지정합니다. IDENTITY 열이 ROWID 자료 유형을 가지는 경우, 값은 데이터베이스 관리자에 의해 생성됩니다.

row-subselect

단일 결과 행을 리턴하는 subselect 결과 열 값이 각 해당 *전이 변수*에 지정됩니다. subselect의 결과에 행이 없으면 널값이 지정됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

주

복수 지정

동일한 SET transition-variable문에 둘 이상의 지정이 포함되는 경우, 지정이 수행되기 전에 모든 *expression*이 평가됩니다. 표현식에서 *transition-variable*에 대한 참조는 항상 단일 SET문에 지정되기 이전의 *transition-variable* 값입니다.

예

예 1

임금 열은 50000보다 커서는 안됩니다. 새 값이 50000보다 큰 경우, 50000으로 설정하십시오.

```
CREATE TRIGGER LIMIT_SALARY
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AS NEW_VAR
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_VAR.SALARY > 50000)
      BEGIN ATOMIC
    SET NEW_VAR.SALARY = 50000;
          END
```

예 2

작업 제목이 갱신된 경우, 새 작업 제목에 의거하여 임금을 증가시키십시오. 연도를 0 위치에 지정하십시오.

SET 이전 변수

```
CREATE TRIGGER SET_SALARY
  BEFORE UPDATE OF JOB ON STAFF
  REFERENCING OLD AS OLD_VAR
              NEW AS NEW_VAR
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    SET (NEW_VAR.SALARY, NEW_VAR.YEARS) =
        (OLD_VAR.SALARY * CASE NEW_VAR.JOB
            WHEN 'Sales' THEN 1.1
            WHEN 'Mgr'   THEN 1.05
            ELSE 1 END ,0);
          END
```

SET 변수

SET 변수 문은 단 하나의 행으로 구성된 결과표를 작성하며, 그 행의 값을 호스트 변수에 지정합니다.

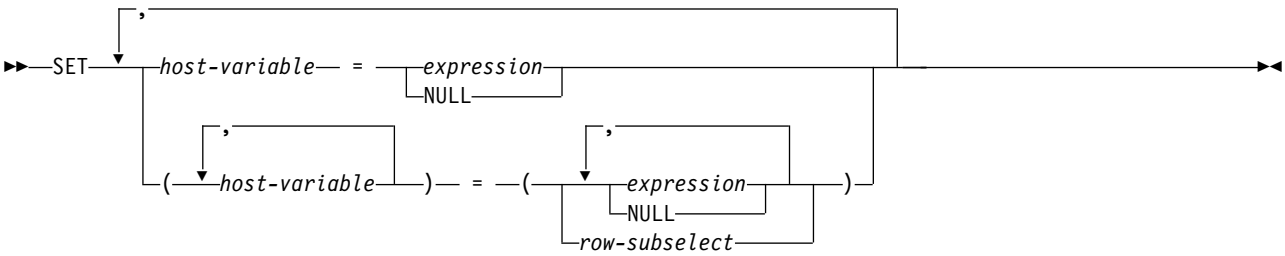
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한부여

*row-subselect*를 지정하는 경우 각 subselect에 필요한 권한부여에 대해서는 335 페이지의 제 4 장『조회』를 참조하십시오.

구문



설명

host-variable, ...

호스트 변수 선언에 대한 규칙에 따라서 선언되어야 하는 하나 이상의 호스트 변수 또는 호스트 구조를 식별합니다(114 페이지의『호스트 변수에 대한 참조』를 참조하십시오). 호스트 구조는 호스트 구조의 각 요소를 대표하는 호스트 변수 리스트에 의해 논리적으로 대체됩니다.

각 *host-variable*에 지정할 값을 *host-variable* 바로 뒤에 지정할 수 있습니다. 예를 들어, *host-variable = expression, host-variable = expression*과 같은 양식입니다. 또는 모든 *host-variable*과 그 값을 지정하는 데 괄호 집합을 사용할 수 있습니다. 예를 들어, (*host-variable, host-variable*) = (*expression, expression*)과 같은 양식입니다.

각 호스트 변수의 자료 유형은 대응되는 결과 열과 호환될 수 있어야 합니다. 각 지정은 80 페이지의『지정과 비교』에서 설명한 규칙에 따라 이루어집니다. 등호 연산자의 왼쪽에 지정하는 *host-variable*의 수는 등호 연산자의 오른쪽에 지정된 대

응 결과에 있는 값의 수와 같아야 합니다. 그 값이 널(null)인 경우 인디케이터 변수를 제공해야 합니다. 지정 오류가 발생하면 값이 변수에 지정되지 않으며 더 이상 어떤 값도 변수에 지정되지 않습니다. 변수에 이미 지정된 값은 그대로 남아 있습니다.

*expression*에 있는 연산식의 결과로 또는 subselect의 SELECT 리스트의 결과로 오류가 발생하거나(0으로 나눔 또는 넘침(overflow)), 문자 변환 오류가 발생하는 경우 결과는 널값입니다. 그 밖의 널값 경우에서는 인디케이터 변수를 제공해야 합니다. 호스트 변수의 값은 정의되지 않습니다. 그러나 이 경우에 인디케이터 변수는 -2로 설정됩니다. 그리고 오류가 발생하지 않았던 것처럼 명령문의 처리가 계속됩니다(그러나 이 오류는 양의 SQLCODE를 유발합니다). 인디케이터 변수를 제공하지 않으면 SQLCA의 SQLCODE 필드에 음의 값이 리턴됩니다. 호스트 변수에 이미 동일한 값이 지정되어 있고, 오류가 발생할 때 그 값이 지정된 상태로 있을 가능성도 있습니다.

expression

호스트 변수의 신규 값을 지정합니다. *expression*은 127 페이지의 『표현식』에서 설명한 유형의 모든 표현식입니다. 열 이름을 포함해서는 안됩니다.

NULL

호스트 변수의 신규 값을 널값으로 지정합니다.

row-subselect

단일 결과 행을 리턴하는 subselect 결과 열 값이 각 해당 *host-variable*에 지정됩니다. subselect의 결과에 행이 없으면 널값이 지정됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

주

지정한 호스트 변수가 문자이며 결과가 들어갈 수 있을 정도로 크지 않으면 'W'가 SQLCA의 SQLWARN1에 지정됩니다. 인디케이터 변수를 제공하면 결과의 실제 길이가 호스트 변수와 연관된 인디케이터 변수에 리턴됩니다.

지정한 호스트 변수가 C NUL 종료 호스트 변수이고 결과와 NUL 종료자가 들어갈 정도로 크지 않은 경우 다음이 적용됩니다.

- *CNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정(또는 CNULRQD(*YES) 옵션을 SET OPTION문에 지정)하면 다음이 발생합니다.
  - 결과가 절단됩니다.
  - 마지막 문자는 NUL 종료자입니다.
  - 값 ‘W’가 SQLCA의 SQLWARN1에 지정됩니다.
- *NOCNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정(또는 CNULRQD(*NO) 옵션을 SET OPTION문에 지정)하면 다음이 발생합니다.
  - NUL 종료자는 리턴되지 않습니다.

– 값 ‘N’이 SQLCA의 SQLWARN1에 지정됩니다.

예

예 1

CURRENT PATH 특수 레지스터의 값을 호스트 변수 HV1에 지정합니다.

```
EXEC SQL SET :HV1 = CURRENT PATH;
```

예 2

LOB 로케이터 LOB1이 CLOB 값과 연관이 있다고 가정합니다. LOB 로케이터를 사용하여 CLOB 값의 부분을 호스트 변수 DETAILS에 지정합니다.

```
EXEC SQL SET :DETAILS = SUBSTR(:LOB1,1,35);
```

UPDATE

UPDATE문은 표나 뷰의 행에서 지정된 열 값을 갱신합니다. 뷰의 행을 갱신하면 뷰의 기본 표에 있는 행도 갱신됩니다.

이 명령문에는 두 가지 형식이 있습니다.

- *탐색* UPDATE 형식은 하나 이상의 행(탐색 조건에 의해 선택적으로 판별)을 갱신하는 데 사용합니다.
- *위치지정* UPDATE 형식은 정확하게 한 행(커서의 현재 위치에 의해 판별)을 갱신하는 데 사용합니다.

호출

탐색 UPDATE문은 어플리케이션 프로그램에 삽입하거나 대화식으로 발행할 수 있습니다. 위치지정 UPDATE문은 어플리케이션 프로그램에 삽입해야만 합니다. 두 형식 모두 동적으로 준비할 수 있는 실행문입니다.

권한부여

명령문의 권한부여 ID가 보유하는 권한은 적어도 다음 중 하나를 포함해야 합니다.

- 명령문에서 식별된 표나 뷰의 경우
  - 표나 뷰에 대한 UPDATE 권한
  - 갱신할 각 열에 대한 UPDATE 권한
  - 표의 소유권
  - 표나 뷰가 들어 있는 라이브러리에 대한 시스템 권한 *EXECUTE
- 관리 권한

다음과 같은 경우에 명령문의 권한부여 ID는 표(또는 표에서 지정된 열)에 대한 UPDATE 권한을 갖습니다.

- 표의 소유자인 경우
- 표나 열에 대해 UPDATE 권한을 부여받은 경우
- 표에 대해 *OBJOPR과 *UPD의 시스템 권한을 부여받은 경우

명령문의 권한부여 ID에 뷰(또는 뷰에서 지정된 열)에 대한 UPDATE 권한이 있을 때[67]

- 뷰나 뷰의 열에 대해 UPDATE 권한을 받은 경우
- 뷰에 대해 *OBJOPR 및 *UPD 시스템 권한이 있고, 뷰 정의를 나타내는 첫 번째 FROM절의 첫 번째 표나 뷰에 대해 *UPD 시스템 권한이 있으며, 이것이 뷰일 때

67. 뷰가 작성되어 있으면 뷰에 대한 UPDATE 권한이 필요 없습니다. 뷰에 갱신이 허용되며 소유자가 subselect에서 참조하는 첫 번째 표에 대해 UPDATE 권한도 가지고 있으면 UPDATE 권한만 받습니다.

해당 뷰 정의의 첫 번째 FROM절에 있는 첫 번째 표나 뷰에 대해 *UPD 시스템 권한을 부여받는 식으로 권한을 받은 경우

*할당 절의 표현식*에 표나 뷰의 열에 대한 참조가 들어 있거나 탐색 UPDATE의 *search-condition*에 표나 뷰의 열에 대한 참조가 들어 있으면, 명령문의 권한부여 ID가 보유하는 권한에는 다음 권한 중 하나가 포함되어야 합니다.

- 표나 뷰에 대한 SELECT 권한
- 관리 권한

명령문의 권한부여 ID는 다음 경우에 표에 대한 SELECT 권한을 갖습니다.

- 표의 소유자인 경우
- 표에 대해 SELECT 권한을 부여받았습니다.
- 표에 대해 *OBJOPR과 *READ의 시스템 권한을 부여받았습니다.

명령문의 권한부여 ID는 다음 경우에 뷰에 대한 SELECT 권한을 갖습니다.

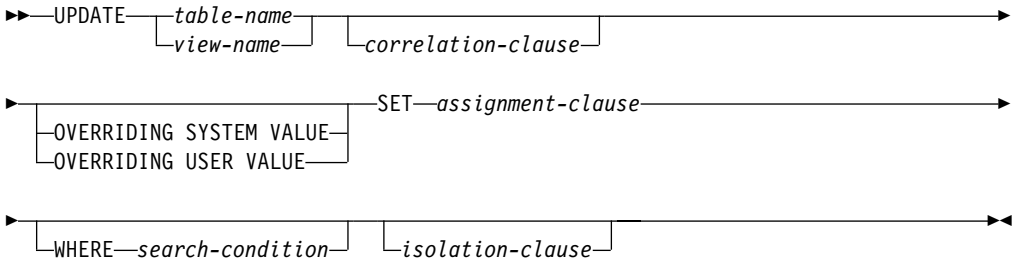
- 뷰의 소유자입니다.
- 뷰에 대해 SELECT 권한을 부여받았습니다.
- 뷰에 대해 *OBJOPR 및 *READ 시스템 권한을 이 뷰가 직접적 또는 간접적으로 종속되어 있는 표나 뷰에 대해 *READ 시스템 권한을 부여받았습니다. 즉, 뷰 정의에 참조된 모든 표나 뷰 그리고 뷰가 참조된 경우 표나 뷰 정의에 참조된 모든 표나 뷰 등

*search-condition*에 부속 조회가 포함되거나 *assignment절*에 *scalar-subselect* 또는 *row-subselect*가 포함되는 경우 각 subselect에 필요한 권한부여에 대해서는 335 페이지의 제 4 장『조회』를 참조하십시오.

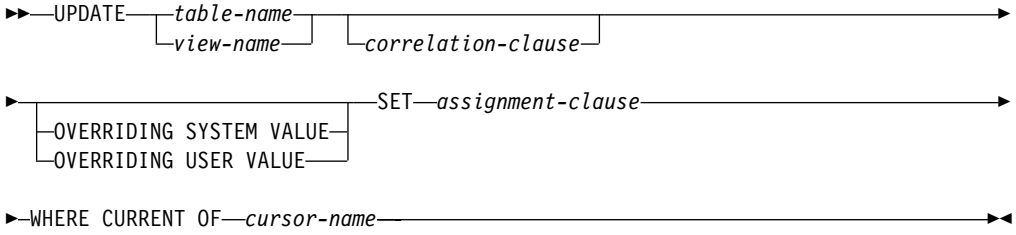
UPDATE

구문

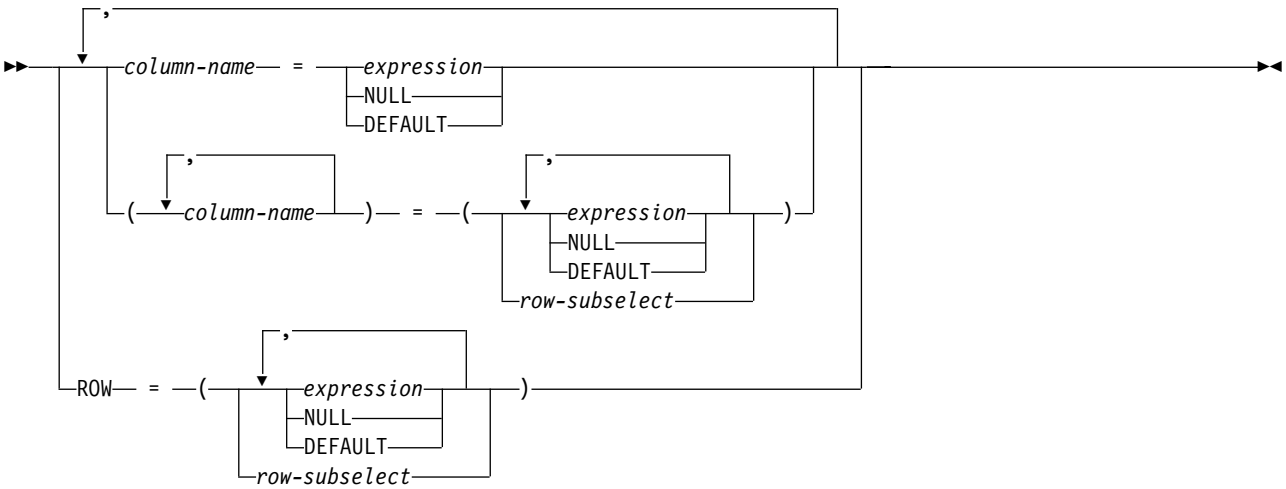
탐색 UPDATE:



위치지정 UPDATE:



assignment절:



isolation절:

설명

table-name 또는 *view-name*

갱신될 표나 뷰를 식별합니다. 이름은 현재 서버에 있는 표나 뷰를 식별하지만 카탈로그 표, 카탈로그 표의 뷰 또는 읽기 전용 뷰를 식별해서는 안됩니다. 읽기 전용 뷰와 갱신가능한 뷰에 대한 설명은 590 페이지의 『CREATE VIEW』를 참조하십시오.

correlation-clause

*search-condition*안에 사용하여 표나 뷰의 이름을 지정할 수 있습니다.
*correlation-clause*에 대한 설명은 341 페이지의 『table-reference』를 참조하십시오
*correlation-name*에 대한 설명은 108 페이지의 『상관명』을 참조하십시오.

OVERRIDING SYSTEM VALUE or OVERRIDING USER VALUE

ROWID 또는 ID 열에 대해 시스템이 생성한 값이나 사용자 지정 값이 사용되는지 여부를 지정합니다. OVERRIDING SYSTEM VALUE가 지정된 경우, SET 절의 내재적 또는 명시적 리스트에 GENERATED ALWAYS로 정의된 열이 들어 있어야 합니다. OVERRIDING USER VALUE가 지정된 경우, INSERT문의 내재적 또는 명시적 리스트는 GENERATED ALWAYS 또는 GENERATED BY DEFAULT로 정의된 열을 포함해야 합니다.

OVERRIDING SYSTEM VALUE

GENERATED ALWAYS로 정의된 열에 대해 SET절로 지정된 값이 사용되도록 지정합니다. system-generated 값은 사용되지 않습니다.

OVERRIDING USER VALUE

GENERATED ALWAYS 또는 GENERATED BY DEFAULT로 정의된 열에 대해 SET절로 지정된 값이 무시되도록 지정합니다. 대신 시스템이 생성한 값이 사용되어 사용자 지정 값을 대체합니다.

만약 OVERRIDING SYSTEM VALUE 또는 OVERRIDING USER VALUE가 지정되지 않았다면:

- ROWID 열이나 ID 열(GENERATED ALWAYS로 정의된 열)에는 값을 지정할 수 없습니다.
- ROWID 열이나 ID 열(GENERATED BY DEFAULT로 정의된 열)에는 값을 지정할 수 있습니다. 값이 지정된 경우, 해당 값이 열에 지정됩니다. 그러나 BY DEFAULT로 정의된 ROWID 열은 지정된 값이 이전에 OS/390 및 z/OS용 DB2 UDB 또는 iSeries용 DB2 UDB에 의해 생성된 유효 행 ID 값인 경우에만 갱신할 수 있습니다. BY DEFAULT로 정의된 ID 열의 값이 갱신될 때, 데이터베이스 관리자는 지정된 ID 열이 고유 제한조건이나 고유 색인의 고유 키가 아닌 경우 해당 값이 열의 고유 값임을 검증하지 않습니다. 고유 제한사항이나 고유 색인이 없는 경우, 데이터베이스 관리자는 NO CYCLE이 유효한 경우에 한해 시스템이 생성된 값 세트 사이에서만 고유 값을 보장할 수 있습니다.

값이 지정된 데이터베이스 관리자가 아니면 새로운 값이 생성됩니다.

SET

열 이름에 할당될 값을 소개합니다.

column-name

갱신할 열을 식별합니다. *column-name*은 지정한 표나 뷰의 열을 식별하지만 스칼라 함수, 상수 또는 표현식에서 파생된 뷰 열을 식별해서는 안됩니다. 열은 두 번 이상 지정할 수 없습니다.

위치지정 UPDATE의 경우 다음이 적용됩니다.

- UPDATE절을 커서의 SELECT문에 지정하였으면 SET 리스트의 각 열명 또한 UPDATE절에 있어야 합니다.
- UPDATE절을 커서의 SELECT문에 지정하지 않았으면, 갱신가능한 열의 이름을 지정할 수 있습니다.

자세한 내용은 356 페이지의 『update절』을 참조하십시오.

뷰의 다른 열과 동일한 열에서 파생된 뷰 열을 갱신할 수 있지만, 두 열 모두를 동일한 UPDATE문에서 갱신할 수는 없습니다.

*column-name*의 리스트를 지정할 경우 *expression*, NULL 및 DEFAULT의 수는 *column-name*의 수와 일치해야 합니다.

ROW

지정한 표나 뷰의 모든 열을 식별합니다. 뷰를 지정하면 스칼라 함수, 상수 또는 표현식으로부터 뷰의 어떠한 열도 파생되지 않을 수 있습니다.

expression, NULL 및 DEFAULT의 수(또는 *row-subselect*로부터의 결과 열의 수)는 행에 있는 열의 수와 일치해야 합니다.

위치지정 UPDATE의 경우 UPDATE절을 커서의 SELECT문에 지정하였으면, 표나 뷰의 각 열 또한 UPDATE절에 있어야 합니다. 자세한 내용은 355 페이지의 ″update절″을 참조하십시오.

뷰의 다른 열과 동일한 열에서 파생된 뷰 열을 포함하는 뷰에 대해 ROW를 지정할 수 없습니다. 그 이유는 두 열 모두를 동일한 UPDATE문에서 갱신할 수 없기 때문입니다.

expression

열의 신규 값을 지정합니다.*expression*은 127 페이지의 『표현식』에서 설명한 유형의 모든 표현식입니다. 열 함수(column function)를 포함해서는 안됩니다.

표현식에 있는 *column-name*은 명명된 표나 뷰의 열을 지정하는 것이어야 합니다. 갱신된 각 행의 경우 표현식에 있는 열의 값은 행이 갱신되기 전에 행에 있는 열의 값입니다.

NULL

열의 신규 값이 널값이도록 지정합니다. NULL은 널가능 열에 대해서만 지정해야 합니다.

DEFAULT

디폴트 값을 열에 지정합니다. 사용되는 값은 다음과 같이 열이 정의된 방법에 따라 다릅니다.

- WITH DEFAULT절을 사용하는 경우 사용되는 디폴트는 열에 대해 정의한 그대로입니다(541 페이지의 『CREATE TABLE』에 있는 *column-definition*에서 *default절* 참조).
- WITH DEFAULT절 또는 NOT NULL절을 사용하지 않는 경우에 사용되는 값은 NULL입니다.
- NOT NULL절은 사용하고 WITH DEFAULT절을 사용하지 않거나 DEFAULT NULL을 사용하는 경우 해당 열에 대해 DEFAULT 키워드를 지정할 수 없습니다.

row-subselect

단일 결과 행을 리턴하는 subselect 선택 리스트에 있는 결과 열의 수는 할당에 지정된 *column-name*의 수(ROW를 지정할 때는 행의 열 수)와 일치해야 합니다. 결과 열 값이 각 해당 *column-name*에 지정됩니다. subselect의 결과에 행이 없으면 널값이 지정됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

*row-subselect*는 UPDATE문의 목표 표에 있는 열에 대한 참조를 포함할 수 있습니다. 갱신된 각 행의 경우 표현식에서 그러한 열의 값은 행이 갱신되기 전에 행에 있는 열의 값입니다.

WHERE

갱신할 행을 지정합니다. 절을 생략할 수 있습니다. *search-condition*을 제공하거나 커서 이름을 지정할 수 있습니다. 이 절을 생략하면 표나 뷰의 모든 행이 갱신됩니다.

search-condition

154 페이지의 『탐색 조건』에 설명된 모든 탐색. 부속 조회의 탐색 조건을 제외한 탐색 조건의 각 *column-name*은 표나 뷰의 열을 지정하는 것이어야 합니다. 동일한 표가 UPDATE와 부속 조회 모두의 기본 오브젝트가 되는 부속 조회가 탐색 조건에 있는 경우 부속 조회는 행이 갱신되기 전에 완전하게 평가됩니다.

*search-condition*은 표나 뷰의 각 행에 적용됩니다. 갱신되는 행은 *search-condition*의 결과가 참인 행입니다.

*search-condition*에 부속 조회가 포함되는 경우에 해당 부속 조회는 *search-condition* 이행에 적용되고 그 *search-condition*을 적용하는 데 부속 조

회의 결과가 사용될 때마다 실행되는 것으로서 생각될 수 있습니다. 실제적으로, 상관 참조가 없는 부속 조회는 한 번만 실행됩니다. 상관 참조가 있는 부속 조회는 각 행에 대해 한 번씩 실행되어야 할 수도 있습니다.

CURRENT OF *cursor-name*

갱신 조작에 사용될 커서를 식별합니다. *cursor-name*은 598 페이지의 『DECLARE CURSOR』에서 설명한 대로 선언된 커서를 식별해야 합니다.

명명된 표나 뷰는 커서에 대한 SELECT문의 FROM절에도 지정해야 하며 커서의 결과표가 읽기 전용이어서는 안됩니다. 읽기 전용 결과표에 대한 설명은 598 페이지의 『DECLARE CURSOR』를 참조하십시오.

UPDATE문을 실행할 때 커서는 행에 위치하고 있어야 합니다. 커서가 있는 행이 갱신됩니다.

isolation-clause

이 명령문에 대해 사용될 분리 레벨을 지정합니다. *isolation-clause*에 대한 설명은 isolation절을 참조하십시오.

UPDATE 규칙

지정

갱신 값이 제 2 장에 설명된 지정 규칙에 따라서 열에 지정됩니다.

유효성

식별된 표 또는 식별된 뷰의 기본 표에 고유 색인 또는 고유 제한조건이 하나 이상 있는 경우 표에 갱신되는 각 행은 표의 색인 때문에 적용되는 제한조건을 따라야 합니다.

고유 색인 또는 고유 제한조건은 COMMIT(*NONE)를 지정하지 않은 경우에 명령문의 끝에서 검사됩니다. 복수 행 갱신의 경우 이러한 검사는 모든 행이 갱신된 다음에 일어납니다. COMMIT(*NONE)를 지정하면 각 행이 갱신될 때마다 검사가 수행됩니다.

식별된 표 또는 식별된 뷰의 기본 표에 하나 이상의 검사 제한조건이 있는 경우 갱신되는 표의 각 행에 대해 각 검사 제한조건은 참이거나 또는 알 수 없는 상태이어야 합니다.

검사 제한조건은 명령문의 끝에서 검사됩니다. 복수 행 갱신의 경우 이러한 검사는 모든 행이 갱신된 다음에 일어납니다.

뷰가 식별되는 경우 갱신된 행이 모든 적용가능한 WITH CHECK OPTION을 따라야 합니다. 자세한 내용은 590 페이지의 『CREATE VIEW』를 참조하십시오.

트리거

식별된 표 또는 식별된 뷰의 기본 표에 갱신 트리거가 있는 경우 트리거는 활성화됩니다. 트리거는 다른 명령문이 실행되도록 하거나 갱신된 값에 따라 오류 조건을 야기합니다.

참조 무결성

상위 행의 상위 키 값은 변경할 수 없습니다.

갱신 값이 널이 아닌 외부 키를 생성하는 경우 외부 키는 관계의 상위 표에서 일부 상위 키 값과 같아야 합니다.

참조 제한조건(RESTRICT 삭제 규칙에 따른 참조 제한조건은 제외)은 명령문의 끝에서 검사됩니다. 복수 행 갱신의 경우 이러한 검사는 모든 행이 갱신된 다음에 일어납니다.

주

갱신 값이 참조 제한조건을 위배하거나 COMMNIT(*NONE)를 지정하지 않고 UPDATE 문을 실행하는 동안 다른 오류가 발생하면 명령문을 실행하는 동안 수행한 모든 변경 내용이 취소됩니다. 그러나, 오류 전에 작업 단위에서 발생한 다른 변경사항은 취소되지 않습니다. COMMIT(*NONE)가 지정되면 변경사항은 취소되지 않습니다.

커서의 상태를 예측할 수 없게 만드는 오류가 발생할 수도 있습니다.

UPDATE문의 실행을 완료한 다음, SQLCA의 SQLERRD(3) 값은 갱신된 행의 수입니다. SQLCA의 설명은 871 페이지의 부록 B『SQL 통신 영역』을 참조하십시오.

적절한 잠금이 아직 없으면, 성공적인 UPDATE문의 실행으로 하나 이상의 배타적 잠금이 예약됩니다. 확약 또는 롤백 조작으로 잠금이 해제될 때까지는 다음을 통해서만 갱신된 행에 액세스할 수 있습니다.

- 갱신을 수행한 어플리케이션 프로세스
- 읽기 전용 커서, SELECT INTO문 또는 부속 조회를 통해 COMMIT(*NONE) 또는 COMMIT(*CHG)를 사용하는 다른 어플리케이션

잠금으로 인해 다른 어플리케이션 프로세스가 표에 대해 조작을 수행할 수 없습니다. 잠금에 대한 자세한 내용은 24 페이지의『분리 레벨』의 COMMIT, ROLLBACK 및 LOCK TABLE문을 참조하십시오. 또한 데이터베이스 프로그래밍 책을 참조하십시오.

COMMIT(*RR), COMMIT(*ALL), COMMIT(*CS) 또는 COMMIT(*CHG)를 지정한 경우 하나의 UPDATE문에서 최대 500 000 000개의 행을 갱신하거나 변경할수 있습니다. 변경된 행의 수에는 동일한 확약 정의 아래 트리거의 결과로서 삽입, 갱신 또는 삭제된 모든 행이 포함됩니다.

REXX 프로시듀어 안의 UPDATE문에 호스트 변수를 사용할 수 없습니다. 그 대신, UPDATE는 매개변수 마커를 사용하는 PREPARE 및 EXECUTE의 오브젝트이어야 합니다.

DATALINK 열의 URL 값이 갱신된 경우, 기존 DATALINK 값을 삭제하고 새 값을 삽입하는 것과 같습니다. 우선, 기존 값이 파일에 링크된 경우, 해당 파일의 링크가 해제됩니다. 그 다음, DATALINK 값의 연계 값이 비어 있지 않은 경우, 지정된 파일은 해당 열에 링크됩니다.

DATALINK 열의 주석 값은 빈 스트링을 URL 경로로 지정하여(예를 들어, DLVALUE 스칼라 함수의 자료 위치 인수로 또는 새 값을 기존 값과 동일하게 설정하여) 파일을 다시 링크하지 않고도 갱신할 수 있습니다. DATALINK 열이 널(null)로 갱신된 경우, 기존 DATALINK 값을 삭제하는 것과 같습니다.

기존 값이나 새 값의 파일 서버가 더 이상 데이터베이스 서버로 등록되지 않은 경우 DATALINK 값을 갱신하려고 할 때 오류가 발생할 수 있습니다.

키워드 동의어: 다음 키워드는 이전 릴리스와 호환을 위해 지원되는 동의어입니다. 이 키워드는 비표준이고 사용될 수 없습니다.

- 키워드 NONE은 NC에 대한 동의어로 사용될 수 있습니다.
- 키워드 CHG는 UR에 대한 동의어로 사용될 수 있습니다.
- 키워드 ALL은 RS에 대한 동의어로 사용될 수 있습니다.

예

예 1

EMPLOYEE 표의 사원 번호(EMPNO) '000290'의 업무(JOB)를 'LABORER'로 변경합니다.

```
UPDATE EMPLOYEE
  SET JOB = 'LABORER'
  WHERE EMPNO = '000290'
```

예 2

PROJECT 표에서 해당 부서(DEPNO) 'D21'이 담당하는 모든 프로젝트에 대해 프로젝트 담당 사원(PRSTAFF)을 1.5씩 증가시킵니다.

```
UPDATE PROJECT
  SET PRSTAFF = PRSTAFF + 1.5
  WHERE DEPTNO = 'D21'
```

예 3

부서(WORKDEPT) 'E21'의 관리자를 제외한 모든 사원이 임시로 재지정되었습니다. EMPLOYEE 표에서 사원의 업무(JOB)를 NULL로 변경하고, 급여(SALARY, BONUS, COMM)를 0으로 변경하여 이를 나타냅니다.

```
UPDATE EMPLOYEE
  SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

예 4

C 프로그램에서 EMPLOYEE 표로부터의 행을 표시한 다음, 요구에 따라 일정 사원의 업무(JOB)를 키순에 있는 새로운 업무로 변경합니다.

```
void main ()
  {
   EXEC SQL  BEGIN DECLARE SECTION;
   char change[4];
   char newjob[20];
   EXEC SQL  END DECLARE SECTION;
   EXEC SQL INCLUDE SQLCA ;

   EXEC SQL  DECLARE C1 CURSOR FOR
               SELECT *
                 FROM EMPLOYEE
                 FOR UPDATE OF JOB;

   EXEC SQL  OPEN C1;

   EXEC SQL  FETCH C1 INTO ...     ;

   getlist(change);
   if (strcmp(change, "YES") )
     {
   EXEC SQL  UPDATE EMPLOYEE
                   SET JOB = :newjob
                 WHERE CURRENT OF C1;
     }

   EXEC SQL  CLOSE C1;
    return;
 }
```

VALUES

VALUES문은 트리거에서 사용자 정의 함수 호출 방법 제공합니다. 전이 변수는 사용자 정의 함수로 전달될 수 있습니다.

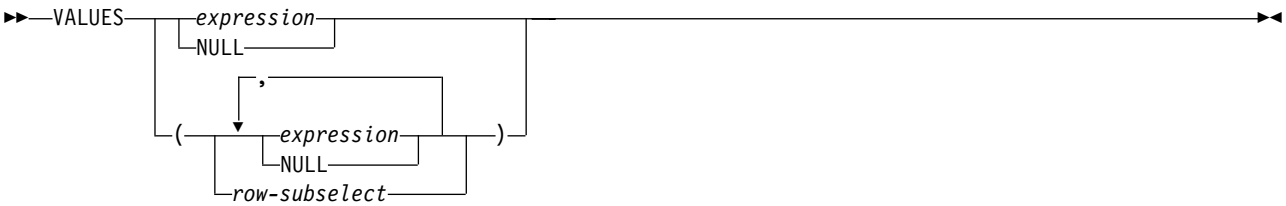
호출

이 명령문은 트리거의 트리거된 조치에서만 사용될 수 있습니다.

권한부여

*row-subselect*를 지정하는 경우 각 subselect에 필요한 권한부여에 대해서는 335 페이지의 제 4 장 『조회』를 참조하십시오.

구문



설명

VALUES

하나 이상의 열로 구성되는 단일 행을 소개합니다.

expression

127 페이지의 『표현식』에서 설명한 유형의 모든 표현식. 호스트 변수를 포함해서는 안됩니다.

NULL

널값을 지정합니다.

row-subselect

단일 결과 행을 리턴하는 subselect subselect의 결과에 행이 없으면 널값이 리턴됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

주

표현식이 평가되고 결과 값은 삭제되며 출력 변수에 할당되지 않습니다. 사용자 정의 함수가 표현식의 일부로 지정되면 사용자 정의 함수가 호출됩니다. 함수가 호출될 때 음

의 SQLCODE가 리턴되면 데이터베이스 관리자는 트리거 실행을 중단하고 트리거가 분리 레벨 *NONE에서 실행되고 있지 않는 한 수행된 트리거된 조치를 구간 복원합니다.

예

예

트리거가 활성화될 때 사용자 정의 함수 NEWEMP를 호출하는 이후(after) 트리거 EMPISRT1을 작성합니다. 표 EMP에 대한 삽입 조작이 트리거를 활성화합니다. 새로운 사원 번호, 성 및 이름에 대한 전이 변수를 사용자 정의 함수로 전달합니다.

```
 CREATE TRIGGER EMPISRT1
AFTER INSERT ON EMPLOYEE
    REFERENCING NEW AS N
    FOR EACH ROW
    MODE DB2SQL
BEGIN ATOMIC
      VALUES( NEWEMP(N.EMPNO, N.LASTNAME, N.FIRSTNAME));
        END
```

VALUES INTO

VALUES INTO문은 단 하나의 행으로 구성된 결과표를 작성하며, 그 행의 값을 호스트 변수에 지정합니다.

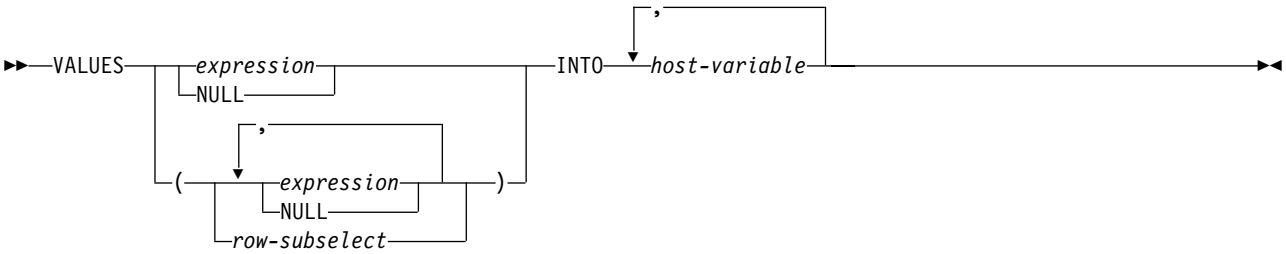
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한부여

*row-subselect*를 지정하는 경우 각 subselect에 필요한 권한부여에 대해서는 335 페이지의 제 4 장 『조회』를 참조하십시오.

구문



설명

VALUES

하나 이상의 열로 구성되는 단일 행을 소개합니다.

expression

호스트 변수의 신규 값을 지정합니다. *expression*은 127 페이지의 『표현식』에서 설명한 유형의 모든 표현식입니다. 열 이름을 포함해서는 안됩니다. 호스트 구조는 지원되지 않습니다.

NULL

호스트 변수의 신규 값을 널값으로 지정합니다.

row-subselect

단일 결과 행을 리턴하는 subselect 결과 열 값이 각 해당 *host-variable*에 지정됩니다. subselect의 결과에 행이 없으면 널값이 지정됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

INTO

호스트 변수 및 호스트 구조의 리스트를 소개합니다. 결과 행의 첫 번째 값이 리스트의 첫 번째 호스트 변수에 지정되고, 두 번째 값은 두 번째 호스트 변수에 지정되는 방법으로 행의 값이 변수에 지정됩니다. 각 지정은 80 페이지의 『지정과 비교』에서 설명한 규칙에 따라 이루어집니다.

호스트 변수의 수가 값의 수보다 작으면, 값 'W'가 SQLCA의 SQLWARN3 필드에 지정됩니다(871 페이지의 부록 B 『SQL 통신 영역』 참조). 결과 열의 수보다 변수가 더 많을 경우 경고가 발행되지 않음에 유의하십시오. 그 값이 널(null)인 경우 인디케이터 변수를 제공해야 합니다. 지정 오류가 발생하면 값이 변수에 지정되지 않으며 더 이상 어떤 값도 변수에 지정되지 않습니다. 변수에 이미 지정된 값은 그대로 남아 있습니다.

*expression*에 있는 연산식의 결과로 또는 subselect의 SELECT 리스트의 결과로 오류가 발생하거나(0으로 나눔 또는 넘침(overflow)), 문자 변환 오류가 발생하는 경우 결과는 널값입니다. 그 밖의 널값 경우에서는 인디케이터 변수를 제공해야 합니다. 호스트 변수의 값은 정의되지 않습니다. 그러나 이 경우에 인디케이터 변수는 -2로 설정됩니다. 그리고 오류가 발생하지 않았던 것처럼 명령문의 처리가 계속됩니다(그러나 이 오류는 양의 SQLCODE를 유발합니다). 인디케이터 변수를 제공하지 않으면 SQLCA의 SQLCODE 필드에 음의 값이 리턴됩니다. 호스트 변수에 이미 동일한 값이 지정되어 있고, 오류가 발생할 때 그 값이 지정된 상태로 있을 가능성도 있습니다.

host-variable, ...

호스트 구조 및 호스트 변수의 선언 규칙에 따라서 프로그램에 선언해야 하는 하나 이상의 호스트 구조나 호스트 변수를 식별합니다. 선언 규칙에 대해서는 114 페이지의 『호스트 변수에 대한 참조』를 참조하십시오. INTO의 조작 형식에서 호스트 구조는 해당 변수 각각에 대한 참조로 대체됩니다.

주

오류가 발생하면 현재 호스트 변수에 지정되는 값이 없습니다. 그러나 LOB 값이 포함될 경우 해당 호스트 변수가 수정되었을 가능성이 있지만 변수의 내용은 예측할 수 없습니다.

지정한 호스트 변수가 문자이며 결과가 들어갈 수 있을 정도로 크지 않으면 'W'가 SQLCA의 SQLWARN1에 지정됩니다. 인디케이터 변수를 제공하면 결과의 실제 길이가 호스트 변수와 연관된 인디케이터 변수에 리턴됩니다.

지정한 호스트 변수가 C NUL 종료 호스트 변수이고 결과와 NUL 종료자가 들어갈 정도로 크지 않은 경우 다음이 적용됩니다.

- *CNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정(또는 CNULRQD(*YES) 옵션을 SET OPTION문에 지정)하면 다음이 발생합니다.

  - 결과가 절단됩니다.
  - 마지막 문자는 NUL 종료자입니다.
  - 값 'W'가 SQLCA의 SQLWARN1에 지정됩니다.
- *NOCNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정(또는 CNULRQD(*NO) 옵션을 SET OPTION문에 지정)하면 다음이 발생합니다.
  - NUL 종료자는 리턴되지 않습니다.
  - 값 'N'이 SQLCA의 SQLWARN1에 지정됩니다.

예

예 1

CURRENT PATH 특수 레지스터의 값을 호스트 변수 HV1에 지정합니다.

```
EXEC SQL VALUES CURRENT PATH INTO :HV1;
```

예 2

LOB 로케이터 LOB1이 CLOB 값과 연관이 있다고 가정합니다. LOB 로케이터를 사용하여 CLOB 값의 부분을 호스트 변수 DETAILS에 지정합니다.

```
EXEC SQL VALUES (SUBSTR(:LOB1,1,35)) INTO  :DETAILS;
```

WHENEVER

WHENEVER문은 지정한 예외 조건이 발생할 때 취할 조치를 지정합니다.

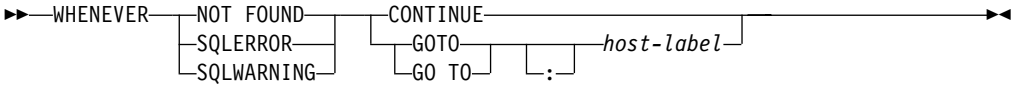
호출

이 명령문은 어플리케이션 프로그램에 삽입될 수만 있습니다. 이 명령문은 실행문이 아닙니다. Java나 REXX에 지정되어서는 안됩니다. REXX에서 오류처리에 대한 내용은 SQL Programming with Host Languages 책을 참조하십시오.

권한부여

필요한 사항이 없습니다.

구문



설명

NOT FOUND, SQLERROR 또는 SQLWARNING절은 예외 조건의 유형을 식별하는 데 사용합니다.

NOT FOUND

SQLCODE +100 또는 SQLSTATE '02000'을 유발하는 모든 조건을 식별합니다.

SQLERROR

음의 SQLCODE를 유발하는 모든 조건을 식별합니다.

SQLWARNING

경고 조건(SQLWARN0이 'W')을 유발하거나 +100이외의 양의 SQLCODE 또는 클래스 코드 01의 SQLSTATE를 유발하는 모든 조건을 식별합니다.

CONTINUE 또는 GO TO절은 식별된 예외 조건 유형이 있을 때 실행될 다음 명령문을 지정하는 데 사용합니다.

CONTINUE

소스 프로그램의 다음 순차 명령어를 지정합니다.

GOTO 또는 **GO TO** *host-label*

*host-label*로 식별되는 명령문을 지정합니다. host-label에 대해, 선택적으로 콜론이 앞에 오는 단일 토큰을 대체하십시오. 토큰의 형식은 호스트 언어에 따라 다릅니다. 예를 들어, COBOL 프로그램에서 토큰은 *section-name* 또는 규정화되지 않은 *paragraph-name*일 수 있습니다.

주

WHENEVER문에는 다음 세 가지 형식이 있습니다.

WHENEVER NOT FOUND
WHENEVER SQLERROR
WHENEVER SQLWARNING

프로그램에 있는 실행가능한 모든 SQL문은 각 유형에서 하나의 암시적 또는 명시적 WHENEVER문의 범위 안에 있습니다. WHENEVER문의 범위는 프로그램에서 명령문의 나열 순서와 관계가 있으며, 실행 순서와는 관계가 없습니다.

SQL문은 소스 프로그램에서 SQL문 앞에 지정되는 각 유형의 마지막 WHENEVER문의 범위 안에 있습니다. 일정한 유형의 WHENEVER문을 SQL문 앞에 지정하지 않을 경우 SQL문은 CONTINUE가 지정되는 유형의 암시적 WHENEVER문의 범위 안에 있습니다.

SQL은 COBOL, C 및 RPG에서 내포 프로그램을 지원합니다. 그러나 SQL은 정상 COBOL, C 및 RPG 범위제한 규칙을 유효한 것으로 간주하지 않습니다. 즉, 프로그램 소스에서 내포 프로시듀어 앞에 지정된 마지막 WHENEVER문은 해당 내포 프로그램에도 계속 유효합니다. WHENEVER문에 참조된 레이블은 그 내부 프로그램에도 중복되어야 합니다. 또는 내부 프로그램이 새로운 WHENEVER문을 지정할 수도 있습니다.

FORTRAN에서 WHENEVER문의 범위는 동일한 서브프로그램 내의 SQL문으로 제한됩니다.

예

COBOL 프로그램에 다음 순서로 삽입해야 하는 명령문을 작성합니다.

1. 오류를 유발하는 명령문의 HANDLER 레이블로 가기

```
EXEC SQL  WHENEVER SQLERROR GOTO HANDLER  END-EXEC.
```

2. 경고를 유발하는 명령문의 처리 계속하기

```
EXEC SQL  WHENEVER SQLWARNING CONTINUE END-EXEC.
```

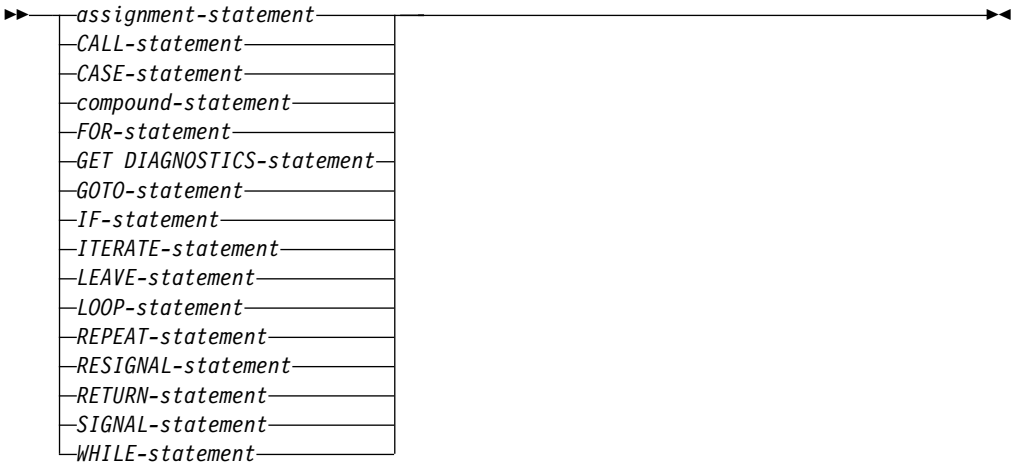
3. 자료의 리턴을 예상했으나 자료를 리턴하지 않은 모든 명령문의 ENDDATA 레이블로 가기

```
EXEC SQL  WHENEVER NOT FOUND GOTO ENDDATA  END-EXEC.
```

제 6 장 SQL 제어문

제어문은 SQL을 구조화된 프로그래밍 언어로 작성하는 것과 비슷한 방식으로 사용할 수 있도록 해주는 SQL문입니다. SQL 제어문은 논리 흐름을 제어하고, 변수를 선언 및 설정하고, 경고 및 예외를 처리할 수 있도록 해줍니다. 일부 SQL 제어문에는 다른 중첩된 SQL문이 포함됩니다.

구문



제어문은 SQL 프로시듀어, SQL 함수, SQL 트리거에서 지원됩니다.

SQL 프로시듀어는 CREATE PROCEDURE문에서 LANGUAGE SQL 및 SQL 루틴을 지정하여 작성됩니다. SQL 함수는 CREATE FUNCTION문에서 LANGUAGE SQL 및 SQL 루틴을 지정하여 작성됩니다. SQL 루틴은 SQL 프로시듀어 또는 SQL 함수가 될 수 있습니다. SQL 트리거는 CREATE TRIGGER문에 SQL 루틴 본문을 지정하여 작성됩니다.

SQL 루틴 본문은 단일 SQL 프로시듀어문이어야 하며 이는 SQL 제어문일 수 있습니다.

SQL 루틴 본문은 프로시듀어, 함수 또는 트리거의 실행가능한 부분이며 데이터베이스 관리자에 의해 프로그램이나 서비스 프로그램으로 변환됩니다. SQL 루틴이나 트리거가 작성될 때 SQL은 삽입 SQL문이 있는 C 소스 코드를 포함할 임시 소스 파일 (QTEMP/QSQLSRC)을 작성합니다. DBGVIEW(*SOURCE)가 지정된 경우, SQL은 임시 소스 파일 QTEMP/QSQDSRC의 루틴 또는 트리거에 대한 루트 소스를 작성합니다.

SQL 프로시듀어나 SQL 트리거는 및 CRTPGM 명령을 사용하여 프로그램(*PGM) 오브젝트로 작성됩니다. SQL 함수는 CRTSRVPGM 명령을 사용하여 서비스 프로그램(*SRVPGM) 오브젝트로 작성됩니다. 프로그램 또는 서비스 프로그램은 프로시듀어, 함수 또는 트리거 이름의 내재적 또는 명시적 규정자인 라이브러리에서 작성됩니다.

프로그램이나 서비스 프로그램이 작성될 때 제어문이 아닌 SQL문은 프로그램이나 서비스 프로그램의 삽입 SQL이 됩니다.

지정된 프로시듀어 또는 함수는 SYSROUTINES 및 SYSPARMS 카탈로그 표에 등록되며, 내부 링크가 SYSROUTINES에서 프로그램으로 작성됩니다. SQL CALL문을 사용하여 프로시듀어가 호출되거나 함수가 SQL문에서 호출될 때 루틴과 연관된 프로그램이 호출됩니다. 지정된 SQL 트리거는 SYSTRIGGER 카탈로그 표에 등록됩니다.

이 장의 나머지 부분에는 구문 도표, 의미 설명, 사용 지침, SQL 루틴 본문을 구성하는 명령문의 사용 예가 있습니다.

824 페이지의 『SQL 매개변수 및 변수에 대한 참조』의 SQL 매개변수 및 변수를 언급하는 섹션도 있습니다. 특정 SQL 제어문을 기술하는 데 사용되는 두 가지 공통 요소는 다음과 같습니다.

- 위에서 기술된 SQL 제어문
- 825 페이지의 『SQL 프로시듀어 명령문』

SQL 제어문 구문 및 추가 정보는 다음 항목을 참조하십시오.

- 826 페이지의 『assignment문』
- 828 페이지의 『CALL문』
- 829 페이지의 『CASE문』
- 831 페이지의 『compound문』
- 845 페이지의 『IF문』
- 838 페이지의 『FOR문』
- 840 페이지의 『GET DIAGNOSTICS문』
- 843 페이지의 『GOTO문』
- 847 페이지의 『ITERATE문』
- 849 페이지의 『LEAVE문』
- 851 페이지의 『LOOP문』
- 853 페이지의 『REPEAT문』
- 855 페이지의 『RESIGNAL문』
- 858 페이지의 『RETURN문』
- 861 페이지의 『SIGNAL문』

- 864 페이지의『WHILE문』

SQL 매개변수 및 변수에 대한 참조

SQL 매개변수 및 SQL 변수는 *host-variable*이 지정될 수 있는 SQL 프로시듀어 명령문 어디에서나 참조될 수 있습니다.

SQL 매개변수는 루틴 어디서든 참조될 수 있으며, 루틴명으로 규정될 수 있습니다. SQL 변수는 선언된 혼합문 어디서나 참조될 수 있으며, 혼합문 시작시에 지정된 레이블명으로 규정될 수 있습니다.

모든 SQL 매개변수 및 SQL 변수는 널이 될 수 있다고 간주됩니다. SQL 변수는 NOT NULL로 명시적으로 선언될 수 있습니다. SQL 매개변수명 또는 SQL 루틴의 SQL 변수명은 루틴에서 참조된 표 또는 뷰의 열 이름과 동일할 수 있습니다. 이때 이름이 열인지, SQL 변수인지, SQL 매개변수인지를 표시하기 위해 명시적으로 규정되어야 합니다.

이름이 규정되지 않은 경우 다음 규칙은 이름이 열을 참조하는지 또는 SQL 변수 또는 매개변수를 참조하는지를 설명합니다.

- SQL 루틴 본문에 지정된 표 및 뷰가 루틴이 작성될 때 존재하면 이름은 먼저 열 이름으로 검사됩니다. 열로 발견되지 않으면 그 다음에는 집합의 SQL 변수로 그 다음에는 SQL 매개변수 이름으로 검사됩니다.
- 참조된 표 또는 뷰가 루틴이 작성될 때 존재하지 않으면 이름은 먼저 SQL 변수 이름으로 검사된 다음 SQL 매개변수 이름으로 검사됩니다. 발견되지 않으면 열로 가정됩니다.

SQL 루틴의 SQL 매개변수나 SQL 변수의 이름은 특정 SQL문에서 사용된 식별자의 이름과 동일할 수 있습니다. 이름이 규정되지 않은 경우 다음 규칙은 그 이름이 식별자 또는 SQL 매개변수나 SQL 변수를 참조하는지 여부를 설명합니다.

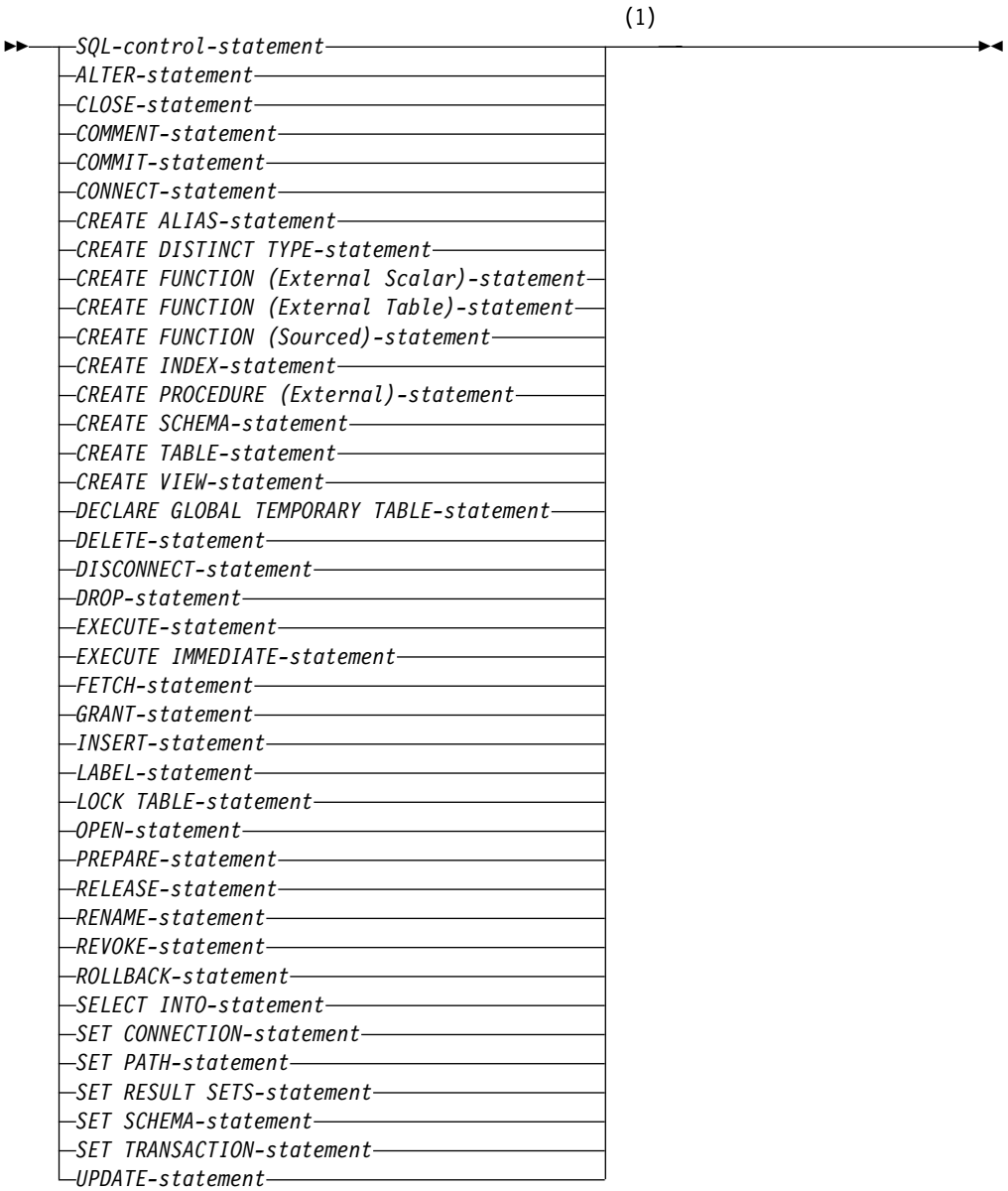
- SET PATH 및 SET SCHEMA문에서 이름은 SQL 매개변수명이나 SQL 변수명으로 검사됩니다. SQL 변수나 SQL 매개변수명으로 발견되지 않으면 식별자로 사용됩니다.
- CONNECT문에서 이름은 식별자로 사용됩니다.

SQL 프로시듀어, 함수, 트리거, SQL 매개변수 및 SQL 변수에 사용되는 이름은 'SQL'로 시작해서는 안됩니다.

SQL 프로시듀어 명령문

SQL 제어문을 사용하면 SQL 제어문 내에 여러 SQL문을 지정할 수 있습니다. 이들 명령문은 SQL 프로시듀어 명령문으로 정의됩니다.

구문



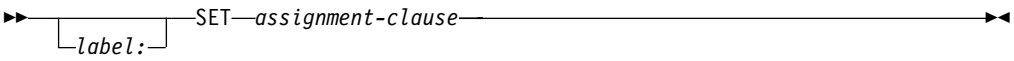
주:

1 COMMIT, ROLLBACK, CONNECT, DISCONNECT, SET CONNECTION 및 SET RESULT SETS문은 SQL 프로시듀어에서만 허용됩니다. SET TRANSACTION문은 SQL 프로시듀어와 트리거에서 허용됩니다.

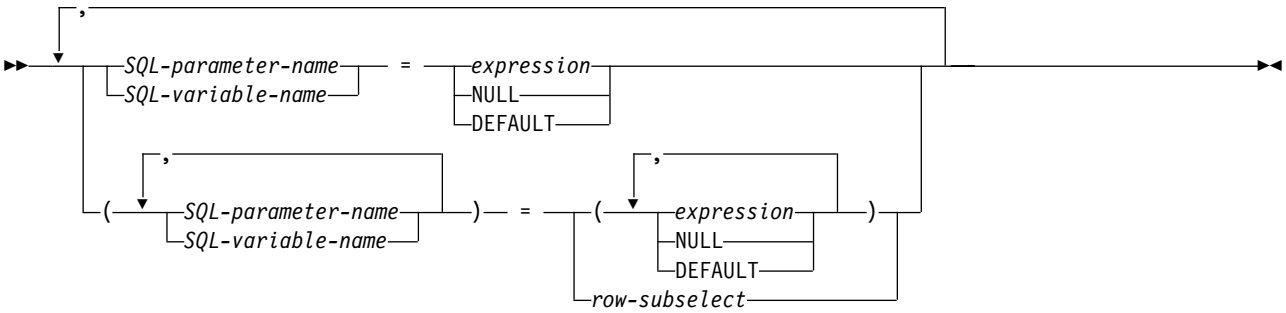
assignment문

assignment문은 SQL 매개변수 또는 SQL 변수에 값을 지정합니다.

구문



assignment절:



설명

label

지정문의 레벨을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

SQL-parameter-name

지정 목표인 SQL 매개변수를 식별합니다. SQL 매개변수는 CREATE PROCEDURE 또는 CREATE FUNCTION문의 *매개변수 선언*에 지정해야 합니다.

SQL-variable-name

지정 목표인 SQL 매개변수를 식별합니다. SQL 변수는 혼합문에서 정의되거나 전이 변수입니다.

표현식 또는 **NULL**

지정 소스가 되는 표현식이나 값을 지정합니다.

DEFAULT

전이 변수와 연관된 열에 대한 디폴트 값이 사용될 것임을 지정합니다. 이것은 전이 변수에 대하여 SQL 트리거에서만 지정할 수 있습니다.

row-subselect

단일 결과 행을 리턴하는 subselect 결과 열 값은 해당 SQL 변수나 매개변수에 지정됩니다. subselect의 결과에 행이 없으면 널값이 지정됩니다. 결과에 하나 이상의 행이 있으면 오류가 리턴됩니다.

주

지정문은 SQL 지정 규칙을 따라야 합니다. 지정 규칙은 80 페이지의 『지정과 비교』를 참조하십시오.

소스와 목표의 자료 유형은 호환될 수 있어야 합니다.

스트링이 고정 길이 변수에 지정되고, 스트링 길이가 목표의 길이 속성보다 작을 경우 스트링의 오른쪽에 필요한 수의 1바이트, 2바이트 또는 UCS-2 공백이 채워집니다.

스트링이 변수에 지정되고 스트링이 변수의 길이 속성보다 긴 경우 음수 SQLCODE가 설정됩니다.

필요한 경우 변수에 지정된 스트링이 먼저 목표의 코드화 문자 세트로 변환됩니다.

숫자 변수 지정시 전체 숫자 부분 절단이 발생하면 음수 SQLCODE가 설정됩니다.

변수가 특수 레지스터(PATH 등)의 이름과 일치하는 ID로 선언된 경우, 변수를 구분하여 특수 레지스터의 지정으로부터 구별해야 합니다(예를 들어, 정수로 선언된 변수 PATH의 경우 SET ″PATH″ = 1).

지정 목표가 변수이고 소스가 변수나 상수인 경우 지정은 인라인으로 수행될 수 있습니다. 이 경우 SQLCODE와 SQLSTATE는 재설정되지 않습니다.

SQL 매개변수 지정 규칙: IN 매개변수는 지정문의 왼쪽 또는 오른쪽에 표시될 수 있습니다. 제어가 호출자로 돌아가면, IN 매개변수의 원래 값이 보존됩니다. OUT 매개변수는 지정문의 왼쪽 또는 오른쪽에 표시될 수 있습니다. 우선 값을 지정하지 않고 사용할 경우, 값이 정의되지 않습니다. 제어가 호출자로 돌아가면, OUT 매개변수에 지정된 마지막 값이 호출자로 돌아갑니다. OUT 매개변수의 경우, 매개변수의 첫 번째 값이 호출자에 의해 결정되며 매개변수에 지정된 마지막 값이 호출자로 돌아갑니다.

예

SQL 변수 p_salary를 10% 증가시킵니다.

```
SET p_salary = p_salary + (p_salary * .10)
```

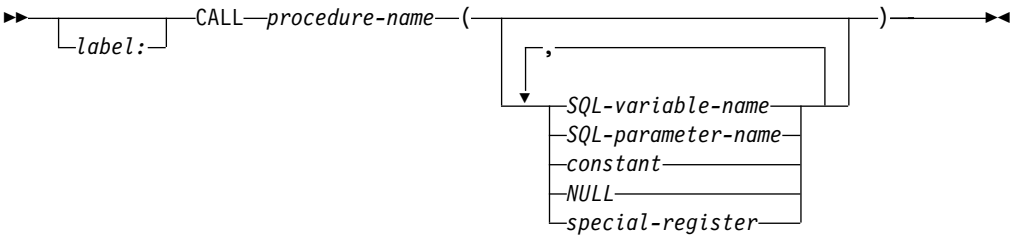
SQL 변수 p_salary를 널값으로 설정합니다.

```
SET p_salary = NULL
```

CALL문

CALL문은 프로시듀어를 호출합니다. 400 페이지의 『CALL』을 참조하십시오.

구문



설명

label

CALL문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

procedure-name

호출할 프로시듀어를 식별합니다. *procedure-name*은 현재 서버에 있는 프로시듀어를 식별해야 합니다.

*SQL-variable-name*또는 *SQL-parameter-name* 또는 *상수* 또는 *NULL* 또는 *special-register*

프로시듀어에 매개변수로 전달할 값 리스트를 식별합니다.

각각의 OUT 또는 INOUT 매개변수는 SQL 매개변수 또는 SQL 변수로 지정되어야 합니다.

주

지정된 인수 수는 해당 프로시듀어에 의해 지정된 매개변수 수와 동일해야 합니다.

프로시듀어의 특수 레지스터의 초기 값은 프로시듀어의 호출자로부터 계승됩니다. 프로시듀어 내에서 특수 레지스터에 지정된 값은 SQL 프로시듀어 전체에 사용되며 해당 프로시듀어로부터 이어서 호출하는 모든 프로시듀어에서 계승합니다. 프로시듀어가 호출자로 돌아가면, 특수 레지스터가 호출자의 원래 값으로 복원됩니다.

자세한 내용은 400 페이지의 『CALL』을 참조하십시오.

예

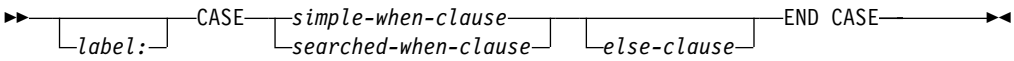
프로시듀어 *proc1*을 호출하고 SQL 변수를 매개변수로 전달합니다.

```
CALL proc1(v_empno, v_salary)
```

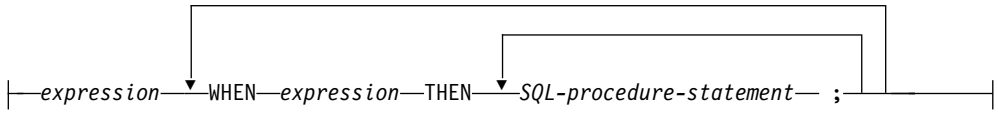
CASE문

CASE문은 복수 조건을 기준으로 실행 경로를 선택합니다.

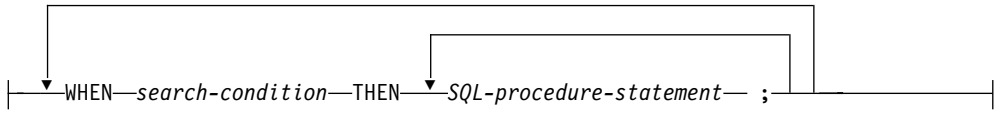
구문



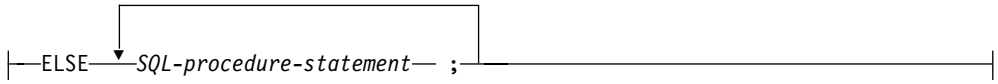
simple-when절:



searched-when절:



else절:



설명

label

CASE문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

simple-when절

첫 번째 WHEN 키워드 앞의 *expresseion* 값은 WHEN 키워드 다음의 각 *expression* 값으로 같은지 여부가 테스트됩니다. 비교가 참이면 THEN문이 실행됩니다. 결과를 알 수 없거나 거짓인 경우 다음 비교가 계속 처리됩니다. 결과가 비교 값과 일치하지 않고 ELSE절이 있는 경우, ELSE절의 명령문이 처리됩니다.

searched-when절

WHEN 키워드 다음의 *search-condition*이 평가됩니다. 참으로 평가되면, 관련 THEN 절의 명령문이 처리됩니다. 거짓으로 평가되거나 알 수 없는 경우, 다음 *search-condition*이 평가됩니다. *search-condition*이 참으로 평가되지 않고 ELSE 절이 있는 경우, ELSE절의 명령문이 처리됩니다.

else-clause

simple-when-clause 또는 *searched-when-clause*가 참인 경우 조건이 지정되지 않으면, *else-clause*의 명령문이 실행됩니다.

WHEN에 지정된 조건이 하나도 참이 아니고 ELSE절이 지정되지 않은 경우 실행 시 오류가 발생하여 CASE문 실행이 종료됩니다(SQLSTATE 20000).

*SQL-PROCEDURE*문

실행되어야 하는 명령문을 지정합니다. 825 페이지의 『SQL 프로시듀어 명령문』을 참조하십시오.

주

CASE문이 가능한 모든 실행 조건을 커버하는지 확인하십시오.

CASE문 네스팅: *simple-when-clause*를 사용하는 CASE문은 최대 3레벨까지 중첩될 수 있습니다. *searched-when-clause*를 사용하는 CASE문의 중첩 레벨 수에는 제한이 없습니다.

예

SQL 변수 v_workdept 값에 따라, 표 DEPARTMENT의 DEPTNAME 열을 적절한 이름으로 갱신합니다.

다음 예는 *simple-when-clause*에 대한 구문을 사용하여 이를 수행하는 방법을 나타냅니다.

```
CASE v_workdept
  WHEN 'A00'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 1';
  WHEN 'B01'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 2';
    ELSE UPDATE department SET
                deptname = 'DATA ACCESS 3';
END CASE
```

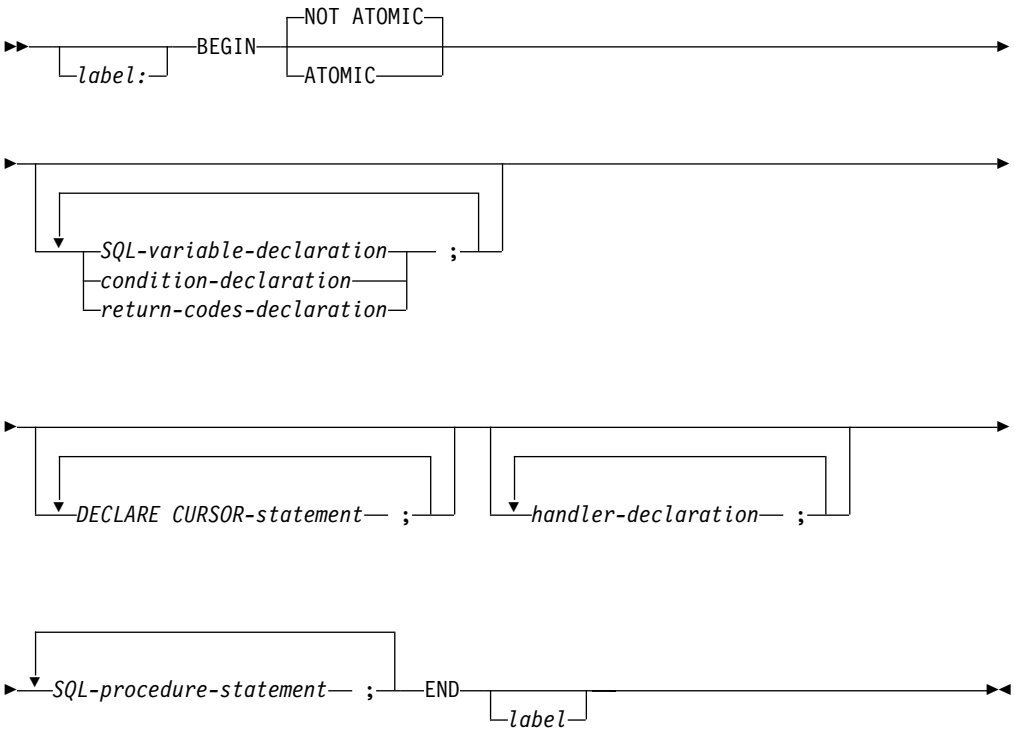
다음 예는 *simple-when-clause*에 대한 구문을 사용하여 이를 수행하는 방법을 나타냅니다.

```
CASE
  WHEN v_workdept = 'A00'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 1';
  WHEN v_workdept = 'B01'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 2';
    ELSE UPDATE department SET
                deptname = 'DATA ACCESS 3';
END CASE
```

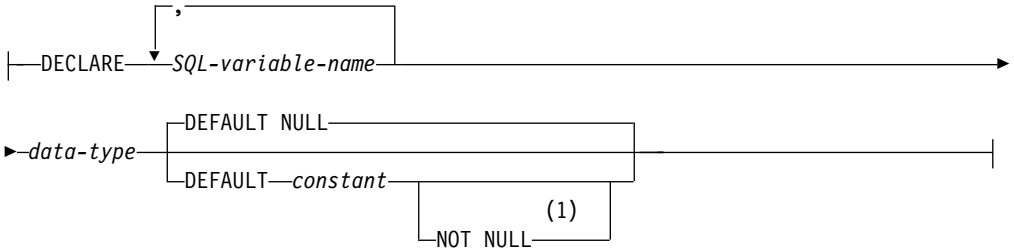
compound문

혼합문은 SQL 루틴에서 다른 명령문을 함께 그룹화합니다. 혼합문 내에서 SQL 변수, 커서 및 핸들러가 선언될 수 있습니다.

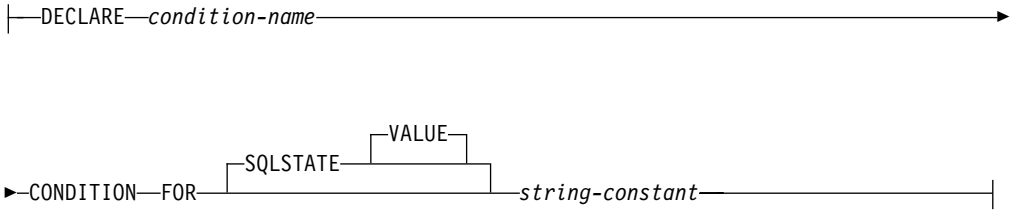
구문



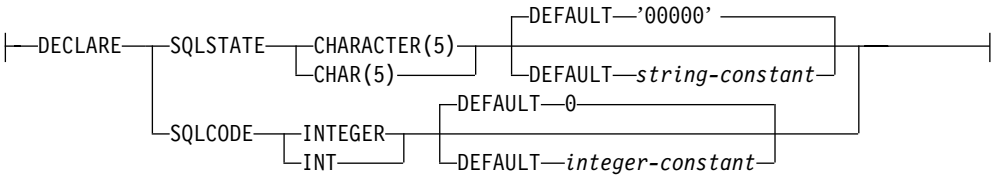
SQL-variable-declaration:



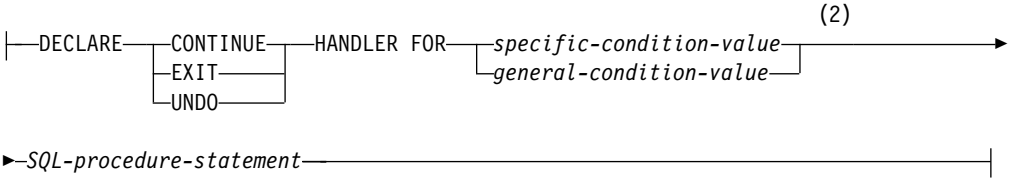
condition-declaration:



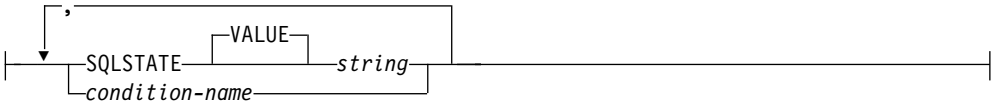
return-codes-declaration:



handler-declaration:

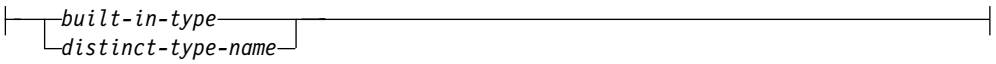


specific-condition-value:



general-condition-value:

data-type:

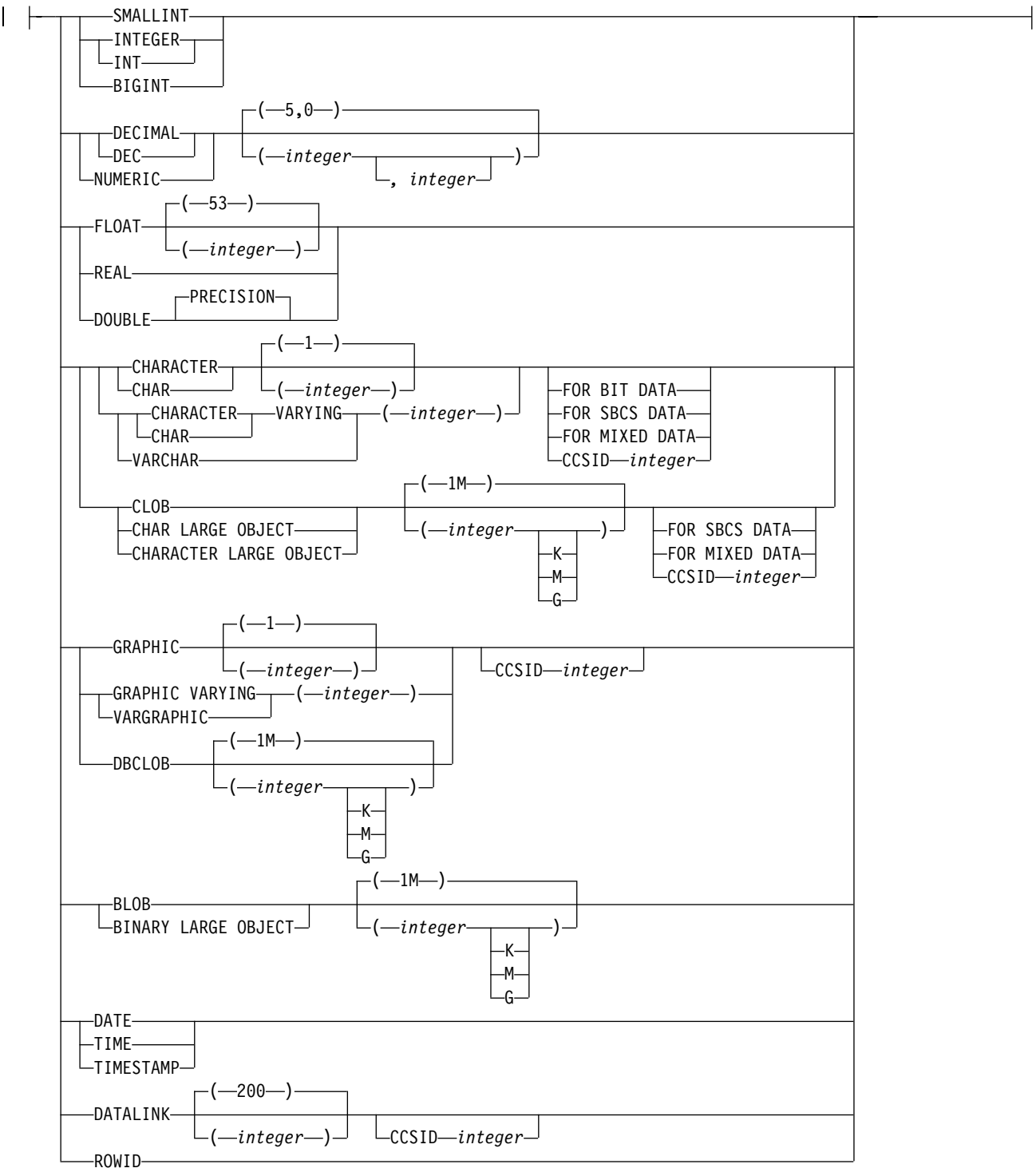


주:

1 DEFAULT 및 NOT NULL절은 어떤 순서로든 지정될 수 있습니다.

2 *specific-condition-value* 및 *general-condition-value*를 동일한 핸들러 선언에 지정할 수 없습니다.

built-in-type:



설명

label

코드 블록에 대해 레이블을 정의합니다. SQL 함수, SQL 프로시듀어 또는 SQL

트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다. 시작 레이블이 지정되면 이 레이블이 혼합문에서 선언된 SQL 변수를 규정하는 데 사용될 수 있으며, LEAVE문에서 지정될 수도 있습니다.

종료 레이블이 지정되는 경우 시작 레이블과 동일해야 합니다.

ATOMIC 또는 **NOT ATOMIC**

ATOMIC은 혼합문에서 오류가 발생하는지, 혼합문의 모든 SQL문이 롤백되는지를 표시합니다. ATOMIC이 지정되면 COMMIT 또는 ROLLBACK문은 혼합문에서 지정될 수 없습니다(ROLLBACK TO SAVEPOINT는 지정할 수 있습니다).

NOT ATOMIC은 혼합문 내의 오류로 인해 혼합문이 롤백되지 않음을 표시합니다. NOT ATOMIC이 SQL 트리거의 외부 복합문에 지정된 경우, ATOMIC으로 처리됩니다.

SQL-variable-declaration

복합문의 로컬 변수를 선언합니다.

SQL-variable-name

로컬 변수명을 정의합니다. 데이터베이스 관리자는 구분되지 않은 모든 SQL 변수명을 대문자로 변환합니다. *SQL-variable-name*은 *compound-statement* 내에서 고유해야 합니다(*compound-statement*에 중첩된 *compound-statements*의 선언 제외). SQL 변수명은 열 이름이나 SQL 매개변수명과 동일해서는 안됩니다. 명령문에 동일한 이름의 열이 있는 경우 SQL 변수명이 분석되는 방식에 대해서는 824 페이지의 『SQL 매개변수 및 변수에 대한 참조』를 참조하십시오. 변수명은 'SQL'로 시작해서는 안됩니다.

*SQL-variable-name*은 선언된 *compound-statement* 내에서만 참조될 수 있습니다(*compound-statement* 내에 중첩된 *compound-statements* 포함).

data-type

변수의 자료 유형을 지정합니다. 자료 유형에 대한 설명은 541 페이지의 『CREATE TABLE』을 참조하십시오.

*data-type*이 그래픽 자료 유형인 경우, CCSID 13488을 지정하여 UCS-2 자료를 나타내십시오. CCSID가 지정되지 않았다면 그래픽 스트링 변수의 CCSID가 작업에 대한 연관 DBCS CCSID입니다.

DEFAULT *상수* 또는 **NULL**

SQL 변수에 대해 디폴트 값을 정의합니다. 변수는 SQL 프로시듀어, SQL 함수 또는 SQL 트리거가 호출될 때 초기화됩니다. 디폴트 값이 지정되지 않으면, SQL 변수는 NULL로 초기화됩니다.

NOT NULL

SQL 변수에 NULL 값이 들어가는 것을 방지합니다. NOT NULL의 생략은 열이 널(null)이 될 수 있음을 의미합니다.

condition-declaration

조건명 및 대응하는 SQLSTATE 값을 선언합니다.

condition-name

조건명을 지정합니다. 조건명은 *compound-statement* 내에서 고유해야 합니다 (*compound-statement*에 중첩된 *compound-statements*의 선언 제외).

*condition-name*은 선언된 *compound-statement* 내에서만 참조될 수 있습니다 (*compound-statement* 내에 중첩된 *compound-statements* 포함).

FOR SQLSTATE *string-constant*

이 조건과 연관된 SQLSTATE를 정의합니다. 스트링 상수는 5 문자로 지정되어야 하며, '00000'은 될 수 없습니다.

return-codes-declaration

SQL문을 실행한 후 리턴된 SQL 리턴 코드에 자동으로 설정되는 SQLSTATE 및 SQLCODE라고 하는 특수 변수를 선언합니다. SQLSTATE 및 SQLCODE 변수 모두 SQL 프로시듀어, SQL 함수 또는 SQL 트리거의 최외단 *compound-statement* 에서만 선언될 수 있습니다.

이 변수에는 지정할 수 있습니다. 그러나, 다음 SQL문이 지정된 값을 대체하기 때문에 지정은 유용하지 않습니다. SQLCODE 및 SQLSTATE 변수는 NULL로 설정될 수 없습니다.

SQLCODE 및 SQLSTATE 변수는 해당 값을 사용할 의도가 있다면 다른 SQL 변수에 즉시 저장해야 합니다. SQLSTATE에 대한 핸들러가 존재하는 경우, 다음 SQL 프로시듀어문에서 값을 대체하지 않도록 하려면 이 지정은 핸들러의 첫 번째 지정이어야 합니다.

declare-CURSOR문

커서를 정의합니다. 커서명은 *compound-statement* 내에서 고유해야 합니다 (*compound-statement*에 중첩된 *compound-statements*의 선언 제외).

*cursor-name*은 선언된 *compound-statement* 내에서만 참조될 수 있습니다 (*compound-statement* 내에 중첩된 *compound-statements* 포함).

커서를 열려면 OPEN문을 지정하고, 커서를 사용하여 행을 읽으려면 FETCH문을 지정하십시오. *declare-cursor-statement*가 SQL 프로시듀어에 있으며 CLOSE문이 지정되지 않았으며 프로시듀어가 작성될 때 RESULT SET가 지정되지 않은 경우, *compound-statement*의 끝에서 커서가 닫힙니다. 자세한 내용은 598 페이지의 『DECLARE CURSOR』를 참조하십시오.

handler-declaration

handler, 즉 복합문에 예외 또는 완료 조건이 발생했을 때 실행할 SQL 프로시듀어문을 지정합니다. 다음과 같은 세 가지 유형의 조건 핸들러가 있습니다.

CONTINUE
일단 핸들러가 성공적으로 호출되면 제어는 예외를 제기한 SQL문 다음의 SQL 문으로 리턴됩니다. IF, CASE, WHILE 또는 REPEAT에서 처럼 비교하는 동안 오류가 발생하면 제어는 해당 END IF, END CASE, END WHILE 또는 END REPEAT 다음의 명령문으로 리턴됩니다.

EXIT
일단 핸들러가 성공적으로 호출되면 제어는 핸들러를 선언한 복합문의 끝으로 리턴됩니다.

UNDO
*compound-statement*에 의한 변경사항을 롤백하고 핸들러를 호출합니다. 일단 핸들러가 올바로 호출되면, 제어는 *compound-statement* 끝으로 리턴됩니다. UNDO가 지정되면 ATOMIC이 지정되어야 합니다.

UNDO는 SQL 함수 또는 SQL 트리거의 외단 *compound-statement*에 지정할 수 없습니다.

핸들러가 활성화되는 조건은 다음과 같습니다.

SQLSTATE *스트링*
특정 SQLSTATE 조건이 발생하면 핸들러가 호출되도록 지정합니다. SQLSTATE는 '00000'이 될 수 없습니다.

condition-name
조건이 발생하면 핸들러가 호출되도록 지정합니다. 조건명은 이전에 *condition-declaration*에 정의되어야 합니다.

SQLEXCEPTION
SQLEXCEPTION이 발생하면 핸들러가 호출되도록 지정합니다. SQLEXCEPTION은 클래스 값이 ″00″, ″01″ 및 ″02″가 아닌 SQLSTATE 값에 대응합니다.

SQLWARNING
SQLWARNING이 발생하면 핸들러가 호출되도록 지정합니다. SQLWARNING은 클래스 값이 ″01″인 SQLSTATE 값에 대응합니다.

NOT FOUND
NOT FOUND 조건이 발생하면 핸들러가 호출되도록 지정합니다. NOT FOUND는 클래스 값이 ″02″인 SQLSTATE 값에 대응합니다.

*handler-declaration*에 동일한 조건을 둘 이상 지정해야 합니다.

주

*handler-declaration*의 규칙:

- 동일한 혼합문 내의 핸들러 선언은 중복 조건을 포함할 수 없습니다.

- 핸들러 선언은 동일한 조건 값 또는 SQLSTATE 값을 한 번 이상 포함할 수 없으며, 동일한 SQLSTATE 값을 표시하는 SQLSTATE 값 및 조건명을 포함할 수 없습니다. SQLSTATE 값 리스트와 자세한 내용은 SQL 프로그래밍 개념 책을 참조하십시오.
- 핸들러는 예외 또는 완료 조건에 대해 가장 적합한 핸들러일 경우에 활성화됩니다. 가장 적절한 핸들러는 예외 또는 완료 조건의 SQLSTATE에 가장 근접하게 일치하는 *compound-statement*에 정의된 핸들러(예외 또는 완료 조건에 대해)입니다. 예를 들어, SQLEXCEPTION의 핸들러 뿐 아니라 SQLSTATE 22001의 핸들러도 존재하는 경우, SQLSTATE 22001의 핸들러는 SQLSTATE 22001이 신호될 때 가장 적합한 핸들러입니다. 핸들러가 없는 경우 오류가 발생하면, *compound-statement* 실행은 종료됩니다. 핸들러가 없는 경우 경고 또는 없음 조건이 발생하면, 다음 명령문으로 처리가 계속됩니다.

예

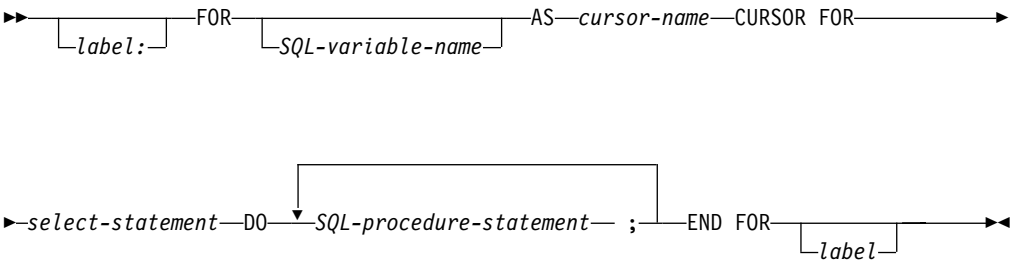
다음 예는 SQL 프로시듀어 PROC1을 작성합니다. 프로시듀어의 루틴 본문은 혼합문입니다. 혼합문은 SQL 변수, SQLSTATE '02000'에 대한 조건, 연속 핸들러 및 선언 커서문을 선언합니다. WHILE문은 FETCH문에서 루프를 돕니다. 파일의 끝, '02000' 조건이 리턴되면 핸들러가 호출되고 SQL 변수 at_end는 1로 설정됩니다. 핸들러에서 제어가 리턴되고, at_end가 더이상 0이 아니기 때문에 WHILE 루프에서 나갑니다.

```
          CREATE PROCEDURE PROC1 () LANGUAGE SQL
BEGIN
          DECLARE v_firstnme VARCHAR(12);
          DECLARE v_midinit CHAR(1);
          DECLARE v_lastname VARCHAR(15);
          DECLARE v_edlevel SMALLINT;
          DECLARE v_salary DECIMAL(9,2);
          DECLARE at_end INT DEFAULT 0;
          DECLARE not_found
                 CONDITION FOR SQLSTATE '02000';
          DECLARE c1 CURSOR FOR
            SELECT firstnme, midinit, lastname,
                              edlevel, salary
            FROM employee;
          DECLARE CONTINUE HANDLER FOR not_found
                      SET at_end = 1;
          OPEN c1;
          FETCH c1 INTO v_firstnme, v_midinit,
                v_lastname, v_edlevel, v_salary;
WHILE at_end = 0 DO
             FETCH c1 INTO
                        v_firstname, v_midinit,
                v_lastname, v_edlevel, v_salary;
         END WHILE;
         CLOSE c1;
          END
```

FOR문

FOR문 표의 각 행에 대해 명령문을 실행합니다.

구문



설명

label

코드 블록에 대해 레이블을 정의합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

종료 레이블이 지정되는 경우 시작 레이블과 동일해야 합니다.

SQL-variable-name

*SQL-variable-name*을 사용하여 명령문의 변수를 규정할 수 있습니다. *SQL-variable-name*은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내의 레이블과 같을 수 없으며, *SQL-variable-name*이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다. *SQL-variable-name* 또는 *label*을 사용하여 명령문 내의 다른 SQL 변수명을 규정할 수 있습니다.

*SQL-variable-name*이 지정된 경우, 이를 사용하여 SQL 함수, SQL 프로시듀어 또는 SQL 트리거를 디버그할 때 명령문의 다른 SQL 변수를 규정할 수 있습니다.

cursor-name

커서를 명명합니다. 지정되지 않은 경우 고유한 커서명이 생성됩니다.

select문

커서의 선택문을 지정합니다.

선택 리스트의 각 표현식은 이름을 가져야 합니다. 표현식이 단순 열 이름이 아닌 경우, AS절을 사용하여 표현식의 이름을 지정해야 합니다. AS절이 지정된 경우 해당 이름은 변수에 대해 사용되며 고유해야 합니다.

SQL-PROCEDURE문

표의 각 행에 대해 실행될 SQL문. SQL문은 FOR문에서 레이블을 지정하는 LEAVE문을 포함할 수 없으며, FOR문의 커서명을 지정하는 OPEN, FETCH 또는 CLOSE를 포함해서는 안됩니다.

주

FOR문은 표의 각 행에 대해 하나 또는 복수 명령문을 실행합니다. 커서는 선택된 열 및 행을 설명하는 선택 리스트를 지정하여 정의됩니다. FOR문 내의 명령문은 선택된 각 행에 대해 실행됩니다.

선택 리스트는 고유한 열 이름으로만 이루어져야 하며, 함수, 프로시듀어 또는 트리거가 작성될 때 선택 리스트에 지정된 표가 있어야 합니다.

FOR문에서 지정된 커서는 FOR문 밖에서 참조될 수 없으며, OPEN, FETCH 또는 CLOSE문에서 지정될 수 없습니다.

예

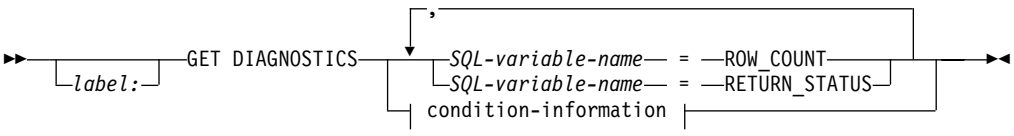
이 예에서 FOR문은 employee 표에서 세 개의 열을 선택하는 커서를 지정하기 위해 사용됩니다. 선택된 모든 행에 대해, SQL 변수 *fullname*은 쉼표, 이름, 공백 및 중간 머릿글자가 뒤에 나오는 성으로 설정됩니다. *fullname*에 대한 각 값은 표 TNAMES에 삽입됩니다.

```
BEGIN
  DECLARE fullname CHAR(40);
  FOR v1 AS
      c1 CURSOR FOR
    SELECT firstnme, midinit, lastname FROM employee
     DO
       SET fullname =
          lastname || ', ' || firstnme ||' ' || midinit;
       INSERT INTO TNAMES VALUE ( fullname );
     END FOR;
END;
```

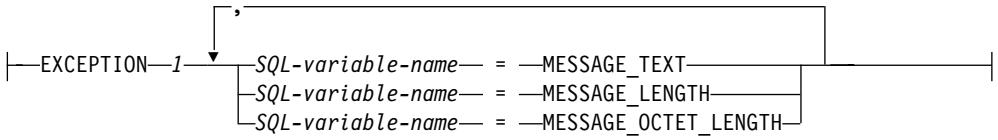
GET DIAGNOSTICS문

GET DIAGNOSTICS문은 실행된 이전 SQL문에 관한 정보를 구합니다.

구문



condition-information:



설명

label

GET DIAGNOSTIC문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

SQL-variable-name

지정 목표인 SQL 변수 또는 SQL 매개변수를 지정합니다. MESSAGE_TEXT가 지정된 경우 변수는 CHAR 또는 VARCHAR이어야 합니다. 그렇지 않으면 SQL 변수는 정수 변수여야 합니다.

ROW_COUNT

실행된 이전 SQL문과 연관된 행 수를 식별합니다. 이전 SQL문이 DELETE, INSERT 또는 UPDATE문인 경우 ROW_COUNT는 트리거나 참조 무결성 제한 조건에 의해 영향 받는 행을 제외하고 해당 명령문에 의해 삭제, 삽입 또는 갱신된 행 수를 지정합니다. 이전 명령문이 PREPARE문인 경우 ROW_COUNT는 준비된 명령문에 있는 결과 행의 예상 수를 식별합니다.

RETURN_STATUS

이전의 SQL CALL문에서 리턴된 상태 값을 식별합니다. 이전 명령문이 CALL문이 아닌 경우 리턴된 값에는 의미가 없으며 예측할 수 없습니다. 자세한 내용은 858 페이지의 『RETURN문』을 참조하십시오.

condition-information

이전 SQL문에 대하여 오류나 경고 정보가 리턴될 것임을 지정합니다.

오류 정보가 필요한 경우 GET DIAGNOSTICS문을 오류를 처리할 핸들러에 맨 처음 지정해야 합니다.

경고 정보가 필요한 경우는 다음과 같이 하십시오.

- 핸들러가 경고 조건을 제어하는 경우 그 핸들러에 GET DIAGNOSTICS문을 맨 처음 지정해야 합니다.
- 핸들러가 경고 조건을 제어하지 않는 경우 GET DIAGNOSTICS문이 이전 명령문 다음에 실행되는 명령문이 되어야 합니다.

MESSAGE_TEXT

실행된 이전 SQL문으로부터 리턴된 오류나 경고 메세지 텍스트를 식별합니다. 이전의 SQL문이 SQLCODE 0로 완료한 경우 빈 스트링이나 공백이 리턴됩니다. 메시지 텍스트가 *SQL-variable-name*의 길이 속성보다 긴 경우, 경고가 리턴되지 않습니다.

MESSAGE_LENGTH 또는 MESSAGE_OCTET_LENGTH

실행된 이전 SQL문으로부터 리턴된 오류나 경고 메세지 텍스트 길이를 식별합니다. 이전의 SQL문이 SQLCODE 0으로 완료한 경우 길이 0이 리턴됩니다.

주

GET DIAGNOSTICS문은 진단 영역(SQLCA)의 내용을 바꾸지 않습니다. SQLSTATE 또는 SQLCODE 특수 변수가 SQL 프로시듀어, SQL 함수 또는 SQL 트리거에 선언된 경우, GET DIAGNOSTICS문을 실행하여 리턴된 SQLSTATE 또는 SQLCODE로 설정됩니다.

예

SQL 프로시듀어에서 GET DIAGNOSTICS문을 실행하여 얼마나 많은 행이 갱신되었는지 판별합니다.

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3)) LANGUAGE SQL
 BEGIN
    DECLARE SQLSTATE CHAR(5);
    DECLARE rcount INTEGER;
    UPDATE CORPDATA.PROJECT
      SET PRSTAFF = PRSTAFF + 1.5
      WHERE DEPTNO = deptnbr;
    GET DIAGNOSTICS rcount = ROW_COUNT;
    /* At this point, rcount contains the number of rows that were updated. */
  END;
```

SQL 프로시듀어 내에서, 저장 프로시듀어 TRYIT를 호출하여 리턴된 상태 값을 처리하십시오. TRYIT는 RETURN문을 사용하여 상태값을 명시적으로 리턴할 수 있으며, 데이터베이스 관리 프로그램은 상태값을 암시적으로 리턴할 수 있습니다. 프로시듀어가 성공하면, 0 값을 리턴합니다.

```
 CREATE PROCEDURE TESTIT ()
LANGUAGE SQL
   A1: BEGIN
```

```
          DECLARE RETVAL INTEGER DEFAULT 0;
          ...
          CALL TRYIT
          GET DIAGNOSTICS RETVAL = RETURN_STATUS;
          IF RETVAL <> 0 THEN
             ...
           LEAVE A1;
       ELSE
             ...
             END IF;     END A1
```

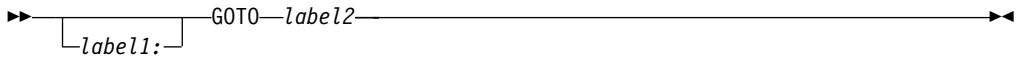
SQL 프로시듀어에서 GET DIAGNOSTICS문을 실행하여 오류 메세지 텍스트를 검색하십시오.

```
   CREATE PROCEDURE divide2 ( IN numerator INTEGER,
                              IN denominator INTEGER,
                         OUT divide_result INTEGER,
                              OUT divide_error VARCHAR(70) )
   LANGUAGE SQL
BEGIN
       DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
          GET DIAGNOSTICS EXCEPTION 1
            divide_error = MESSAGE_TEXT;
           SET divide_result = numerator / denominator;
     END;
```

GOTO문

GOTO문은 SQL 루틴 또는 SQL 트리거에서 사용자 정의 레이블로 분기하는 데 사용됩니다.

구문



설명

label1

코드 블록에 대해 레이블을 정의합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

label2

처리가 계속될 레이블된 명령문을 지정합니다. 레이블된 명령문과 GOTO문은 같은 범위에 있어야 합니다.

- GOTO문이 FOR문에 정의된 경우 *레이블*은 내포된 FOR문을 제외하고 같은 FOR문 안에 정의되어야 합니다.
- GOTO문이 FOR문 외부에 정의되는 경우 *레이블*은 FOR문 안에 정의되어서는 안됩니다.
- GOTO문이 핸들러에 정의된 경우 *레이블*은 같은 핸들러 내에 정의되어야 합니다.
- GOTO문이 핸들러 외부에 정의된 경우 *레이블*은 핸들러 내부에 정의되어서는 안됩니다.

*label2*가 GOTO문이 도달할 수 있는 범위에 정의되지 않은 경우, 오류가 리턴됩니다.

주

GOTO문은 꼭 필요한 경우에만 사용하십시오. 이 명령문은 정상적인 처리 순서를 방해하므로 루틴을 읽고 유지보수하기 더 어려워집니다. IF나 LEAVE와 같은 다른 명령문을 GOTO문 대신 사용하도록 하십시오.

예

다음 명령문에서 매개변수 *rating* 및 *v_empno*가 프로시듀어로 전달됩니다. 근무 기간이 출력 매개변수 *return_parm*에 날짜 기간으로 리턴됩니다. 이 회사에서 근무한 기간이 6개월 미만인 경우 GOTO문이 제어를 프로시듀어 끝으로 전달함으로 *new_salary*는 변경되지 않습니다.

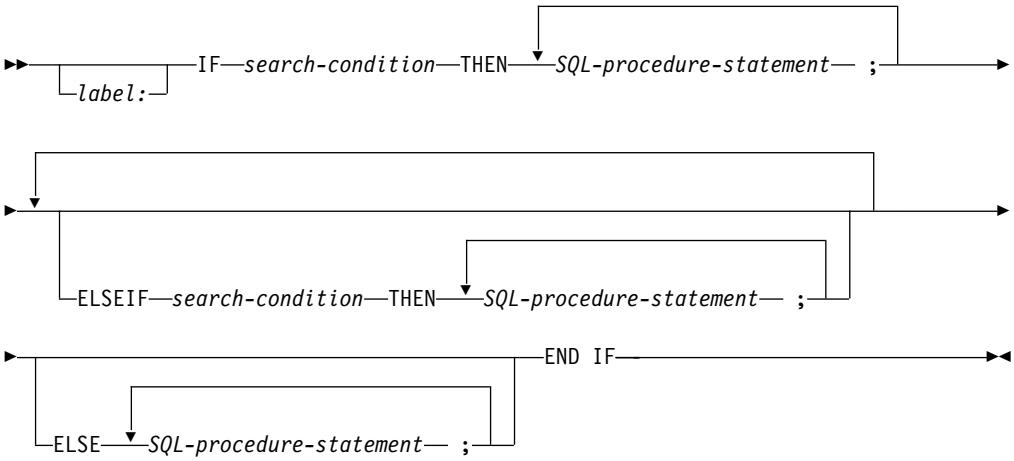
```
CREATE PROCEDURE adjust_salary (IN v_empno CHAR(6),
                                IN rating INTEGER,
                                OUT return_parm DECIMAL(8,2))
  LANGUAGE SQL
  MODIFIES SQL DATA
BEGIN
       DECLARE new_salary DECIMAL(9,2);
       DECLARE service DECIMAL(8,2);
            SELECT salary, current_date - hiredate
               INTO new_salary, service
               FROM employee
               WHERE empno = v_empno;
            IF service < 600
              THEN GOTO exit1;
            END IF;
            IF rating = 1
             THEN SET new_salary =
                          new_salary + (new_salary * .10);
             ELSEIF rating = 2
             THEN SET new_salary =
                          new_salary + (new_salary * .05);
            END IF;
            UPDATE employee
               SET salary = new_salary
               WHERE empno = v_empno;

       exit1: SET return_parm = service;
          END
```

IF문

IF문은 검색 조건의 결과에 따라 서로 다른 SQL문 세트를 실행합니다.

구문



설명

label

IF문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

search-condition

SQL문이 실행되어야 하는 *search-condition*을 지정합니다. 결과를 알 수 없거나 거짓인 경우 다음 탐색 조건 또는 ELSE절이 계속 처리됩니다.

SQL-PROCEDURE문

앞의 *search-condition*이 참인 경우 실행되어야 하는 SQL문을 지정합니다.

예

다음 SQL 프로시듀어는 두 IN 매개변수인 사원 번호와 사원 등급을 받습니다. *rating* 값에 따라, 사원 표의 급여 및 보너스 열은 새 값으로 갱신됩니다.

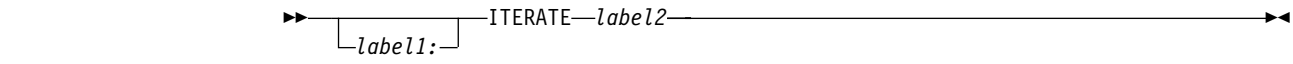
```
CREATE PROCEDURE UPDATE_SALARY_IF
   (IN employee_number CHAR(6), INOUT rating SMALLINT)
   LANGUAGE SQL
             MODIFIES SQL DATA
BEGIN
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE EXIT HANDLER FOR not_found
         SET rating = -1;
```

```
      IF rating = 1
   THEN UPDATE employee
   SET salary = salary * 1.10, bonus = 1000
   WHERE empno = employee_number;
       ELSEIF rating = 2
   THEN UPDATE employee
   SET salary = salary * 1.05, bonus = 500
   WHERE empno = employee_number;
ELSE UPDATE employee
   SET salary = salary * 1.03, bonus = 0
   WHERE empno = employee_number;
      END IF;
    END
```

ITERATE문

ITERATE문은 레이블이 설정된 루프 시작 부분으로 제어 흐름이 리턴되도록 합니다.

구문



설명

label1

코드 블록에 대해 레이블을 정의합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

label2

데이터베이스 관리자가 제어 흐름을 전달하는 FOR, LOOP, REPEAT 또는 WHILE 문의 레이블을 지정합니다.

예

본 예제에서는 커서를 사용하여 새 부서에 대한 정보를 리턴합니다. *not_found* 조건 핸들러가 호출되면, 제어 흐름은 루프를 벗어납니다. *v_dept*의 값이 'D11'인 경우, ITERATE문은 제어 흐름을 LOOP문의 맨 위로 다시 보냅니다. 아니면, 새 행이 DEPARTMENT 표에 삽입됩니다.

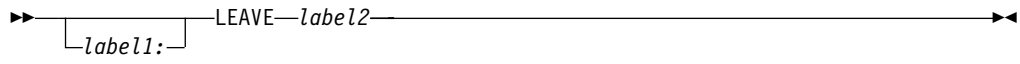
```
    CREATE PROCEDURE ITERATOR ()
  LANGUAGE SQL
            MODIFIES SQL DATA
BEGIN
          DECLARE v_dept CHAR(3);
          DECLARE v_deptname VARCHAR(29);
          DECLARE v_admdept CHAR(3);
          DECLARE at_end INTEGER DEFAULT 0;
          DECLARE not_found CONDITION FOR SQLSTATE '02000';
          DECLARE c1 CURSOR FOR
            SELECT deptno,deptname,admrdept
            FROM department
            ORDER BY deptno;
          DECLARE CONTINUE HANDLER FOR not_found
                      SET at_end = 1;
          OPEN c1;
         ins_loop:
         LOOP
             FETCH c1 INTO v_dept, v_deptname, v_admdept;
             IF at_end = 1 THEN
              LEAVE ins_loop;
             ELSEIF v_dept ='D11' THEN
                ITERATE ins_loop;
            END IF;
```

```
    INSERT INTO department (deptno,deptname,admrdept)
       VALUES('NEW', v_deptname, v_admdept);
END LOOP;
CLOSE c1;
 END
```

LEAVE문

LEAVE문 블록 또는 루프를 나가서 계속 실행합니다.

구문



설명

label1

코드 블록에 대해 레이블을 정의합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

label2

나갈 compound, FOR, LOOP, REPEAT 또는 WHILE문을 지정합니다.

주

LEAVE문이 복합문 바깥으로 제어를 전송하면, 결과 세트를 리턴하는 데 사용된 커서를 제외하고 복합문에서 열려 있는 모든 커서가 닫힙니다.

예

예에는 *c1* 커서에 대해 자료를 페치하는 루프가 포함됩니다. SQL 변수 *at_end*의 값이 0이 아니면, LEAVE문은 제어를 루프 바깥으로 전송합니다.

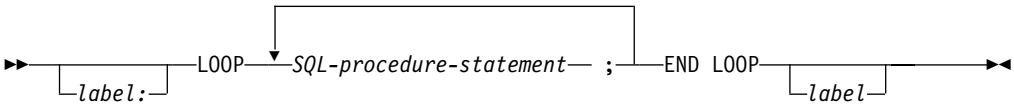
```
 CREATE PROCEDURE LEAVE_LOOP (OUT COUNTER INTEGER)
  LANGUAGE SQL
BEGIN
       DECLARE v_counter INTEGER;
       DECLARE v_firstnme VARCHAR(12);
       DECLARE v_midinit CHAR(1);
       DECLARE v_lastname VARCHAR(15);
       DECLARE at_end SMALLINT DEFAULT 0;
       DECLARE not_found CONDITION FOR SQLSTATE '02000';
          DECLARE c1 CURSOR FOR
          SELECT firstnme, midinit, lastname
            FROM employee;
          DECLARE CONTINUE HANDLER FOR not_found
                      SET at_end = 1;
       SET v_counter = 0;
          OPEN c1;
             fetch_loop:
         LOOP
          FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
          IF at_end <> 0 THEN
               LEAVE fetch_loop;
            END IF;
```

```
    SET v_counter = v_counter + 1;
  END LOOP fetch_loop;
SET counter = v_counter;
  CLOSE c1;
   END
```

LOOP문

LOOP문은 명령문 또는 명령문 그룹의 실행을 반복합니다.

구문



설명

label

LOOP문의 레이블 이름. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다. 시작 레이블이 지정되는 경우 LEAVE문에서 지정될 수 있습니다.

종료 레이블이 지정되는 경우 시작 레이블과 동일해야 합니다.

SQL-procedure statement

루프에서 실행될 SQL문을 지정합니다.

예

이 프로시듀어는 LOOP문을 사용하여 사원 표에서 값을 페치합니다. 루프가 반복될 때마다, OUT 매개변수 *counter*가 증가되며 *v_midinit* 값이 단일 공백(' ')이 아닌지 확인됩니다. *v_midinit*이 단일 공백인 경우, LEAVE문은 제어 흐름을 루프 바깥으로 전달합니다.

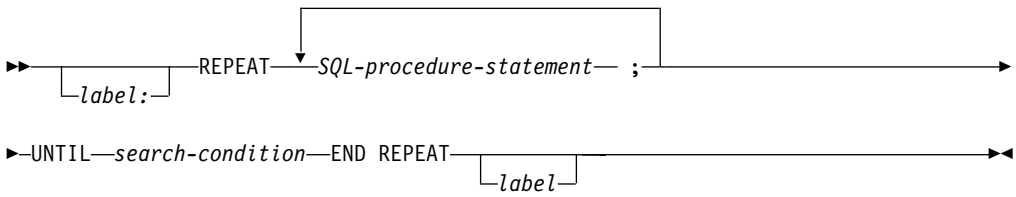
```
 CREATE PROCEDURE LOOP_UNTIL_SPACE (OUT COUNTER INTEGER)
  LANGUAGE SQL
BEGIN
       DECLARE v_counter INTEGER DEFAULT  0;
       DECLARE v_firstnme VARCHAR(12);
       DECLARE v_midinit CHAR(1);
       DECLARE v_lastname VARCHAR(15);
          DECLARE c1 CURSOR FOR
          SELECT firstnme, midinit, lastname
            FROM employee;
       DECLARE CONTINUE HANDLER FOR NOT FOUND
          SET counter = -1;
          OPEN c1;
             fetch_loop:
         LOOP
          FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
          IF v_midinit = ' ' THEN
               LEAVE fetch_loop;
            END IF;
          SET v_counter = v_counter + 1;
```

```
  END LOOP fetch_loop;
SET counter = v_counter;
  CLOSE c1;
   END
```

REPEAT문

REPEAT문은 검색 조건이 참인 경우 명령문 또는 명령문 그룹을 실행합니다.

구문



설명

label

REPEAT문의 레이블 이름. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다. 시작 레이블이 지정된 경우 해당 레이블은 LEAVE문에서 지정될 수 있습니다.

종료 레이블이 지정되는 경우 시작 레이블과 동일해야 합니다.

SQL-PROCEDURE문

REPEAT 루프에서 실행될 SQL문을 지정합니다.

search-condition

*search-condition*은 REPEAT 루프 실행 후에 평가됩니다. 조건이 참인 경우, REPEAT 루프는 종료됩니다. 조건을 알 수 없거나 거짓인 경우 루프 처리가 계속됩니다.

예

REPEAT문은 *not_found* 조건 핸들러가 호출될 때까지 표에서 행을 페치합니다.

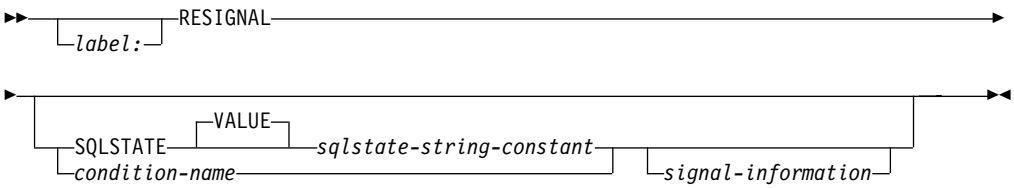
```
 CREATE PROCEDURE REPEAT_STMT (OUT COUNTER INTEGER)
  LANGUAGE SQL
BEGIN
       DECLARE v_counter INTEGER DEFAULT  0;
       DECLARE v_firstnme VARCHAR(12);
       DECLARE v_midinit CHAR(1);
       DECLARE v_lastname VARCHAR(15);
       DECLARE at_end SMALLINT DEFAULT 0;
       DECLARE not_found CONDITION FOR SQLSTATE '02000';
          DECLARE c1 CURSOR FOR
          SELECT firstnme, midinit, lastname
            FROM employee;
          DECLARE CONTINUE HANDLER FOR not_found
```

```
                SET at_end = 1;
   OPEN c1;
      fetch_loop:
      REPEAT
   FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
   SET v_counter = v_counter + 1;
   UNTIL at_end > 0
      END REPEAT fetch_loop;
SET counter = v_counter;
  CLOSE c1;
   END
```

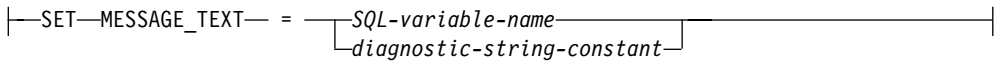
RESIGNAL문

RESIGNAL문은 핸들러 내에 사용되어 오류 조건이나 경고 조건을 리턴합니다.

구문



signal-information:



설명

label

RESIGNAL문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

SQLSTATE VALUE *sqlstate-string-constant*

리턴된 SQLSTATE 오류코드를 지정합니다. 스트링은 정확히 5 문자 스트링 상수로 지정되어야 하며, SQLSTATE에 대한 규칙을 따라야 합니다.

- 각 문자는 숫자 집합((‘0’ - ‘9’) 또는 대문자(‘A’ - ‘Z’)여야 합니다.
- ‘00’은 성공적인 완료를 의미하므로 SQLSTATE 클래스(처음 2자)는 ‘00’이어서는 안됩니다.

SQLSTATE가 해당 규칙을 따르지 않는다면 오류가 표시됩니다.

condition-name

리턴된 조건 이름을 지정합니다. *condition-name*은 복합문 내에 선언되어야 합니다.

MESSAGE_TEXT

오류나 경고를 설명하는 스트링을 지정합니다. 이 스트링은 SQLCA의 SQLERRMC 필드에 리턴됩니다. 스트링의 실제 길이가 70바이트를 넘어가면 경고 없이 절단됩니다.

SQL-variable-name

복합 명령문 내에서 선언되어야 할 SQL 변수를 식별합니다. SQL 변수는 CHAR이나 VARCHAR 자료 유형으로 정의되어야 합니다.

diagnostic-string-constant
메세지 텍스트가 들어 있는 문자 스트링 상수를 지정합니다.

주

유효한 SQLSTATE값은 모두 RESGINAL문에 사용될 수 있습니다. 그러나 어플리케이션에 예약된 범위에 기초하여 프로그래머가 새로운 SQLSTATE를 정의하는 것이 좋습니다. 이렇게 함으로써 차후 릴리스에서 데이터베이스 관리자에 의해 정의될 수도 있는 SQLSTATE값을 의도와 달리 사용하는 것을 막을 수 있습니다. SQLSTATE에 대한 자세한 정보는 iSeries Information Center의 SQL 메세지 및 코드 책을 참조하십시오.

RESIGNAL문이 SQLSTATE절 또는 *condition-name* 없이 지정되면, RESIGNAL문은 핸들러 내에 있어야 합니다. SQL 루틴은 핸들러를 호출한 동일한 조건을 가지고 호출자에게 리턴합니다.

RESIGNAL문이 실행되고 SQLSTATE 또는 *condition-name*이 지정되면, 다음과 같이 SQLCA에 리턴된 SQLCODE는 SQLSTATE 값에 기초하여 설정됩니다.

- 지정된 SQLSTATE 클래스가 '01' 또는 '02'인 경우 경고 또는 찾을 수 없음이 신호되고 SQLCODE가 +438로 설정됩니다.
- 그렇지 않으면 예외가 표시되고 SQLCODE는 -438로 설정됩니다.

RESIGNAL문이 발행되고 SQLSTATE 값 또는 *condition-name* 중 아무 것도 지정되지 않으면 SQLCODE는 변경되지 않습니다.

SQLSTATE나 조건에서 예외('01'이나 '02'가 아닌 SQLSTATE 클래스) 신호가 발생하는 경우 다음 처리가 발생합니다.

- 핸들러가 RESIGNAL문과 같은 복합문에 있고 이 *compound-statement*에 SQLEXCEPTION에 대한 핸들러 또는 지정된 SQLSTATE나 조건이 들어 있는 경우, 예외가 처리되고 제어는 해당 핸들러로 넘어갑니다.
- *compound-statement*가 중첩되어 있고 외부 레벨 *compound-statement*에 SQLEXCEPTION에 대한 핸들러 또는 지정된 SQLSTATE나 조건이 있는 경우, 예외가 처리되고 제어는 해당 핸들러로 넘어갑니다.
- 그렇지 않으면 예외는 처리되지 않고 제어는 즉시 복합문 끝으로 리턴됩니다.

SQLSTATE나 조건에서 경고(SQLSTATE 클래스 '01') 또는 찾을 수 없음(SQLSTATE 클래스 '02') 신호가 발생하는 경우, 다음 처리가 수행됩니다.

- 핸들러가 RESIGNAL문과 같은 복합문에 있고 이 *compound-statement*에 SQLWARNING(SQLSTATE 클래스가 '01'인 경우), NOT FOUND(SQLSTATE 클래스가 '02'인 경우) 또는 지정된 SQLSTATE나 조건에 대한 핸들러가 들어 있는 경우, 경고 또는 찾을 수 없음 조건이 처리되고 제어는 핸들러로 넘어갑니다.

- *compound-statement*가 중첩되어 있고 외부 레벨 복합문에 SQLWARNING (SQLSTATE 클래스가 '01'인 경우), NOT FOUND(SQLSTATE 클래스가 '02'인 경우) 또는 지정된 SQLSTATE나 조건에 대한 핸들러가 들어 있는 경우, 경고 또는 찾을 수 없음 조건이 처리되고 제어는 핸들러로 넘어갑니다.
- 그렇지 않으면 경고는 처리되지 않고 다음 명령문을 계속 처리합니다.

SQLSTATE 값은 2자로 된 클래스 코드 값과 그 뒤에 나오는 3자로 된 서브클래스 코드 값으로 구성됩니다. 클래스 코드 값은 성공한 실행 조건 및 성공하지 못한 실행 조건의 클래스를 나타냅니다.

예

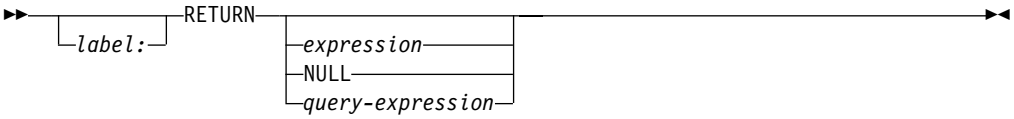
이 예에서는 0으로 나누기 오류가 발생합니다. IF문에서 SIGNAL문을 사용하여 넘침 조건 핸들러를 호출합니다. 조건 핸들러는 RESIGNAL문을 사용하여 다른 SQLSTATE 값을 클라이언트 어플리케이션에 리턴합니다.

```
CREATE PROCEDURE divide ( IN numerator INTEGER,
                          IN denominator INTEGER,
                          OUT divide_result INTEGER )
   LANGUAGE SQL
BEGIN
       DECLARE overflow CONDITION FOR '22003';
       DECLARE CONTINUE HANDLER FOR overflow
           RESIGNAL SQLSTATE '22375';
       IF denominator = 0 THEN
           SIGNAL overflow;
            ELSE
           SET divide_result = numerator / denominator;
     END IF;
     END;
```

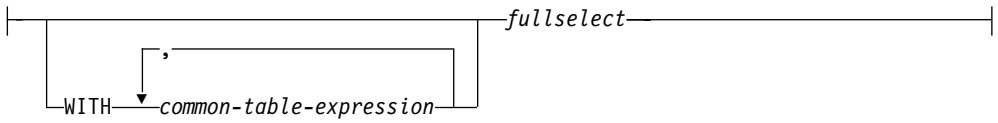
RETURN문

RETURN문은 루틴에서 리턴됩니다. SQL 함수의 경우 함수 결과를 리턴합니다. SQL 프로시듀어의 경우 정수 상태 값을 선택적으로 리턴합니다. SQL 표 함수의 경우 함수 결과로서 표를 리턴합니다.

구문



query-expression:



설명

label

RETURN문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

expression

루틴에서 리턴된 값을 지정합니다.

- 루틴이 함수인 경우 *expression*이 지정되어야 합니다. 이 값은 CREATE FUNCTION문의 RETURNS절에서 지정되는 자료 유형과 호환되어야 합니다.
- 루틴이 프로시듀어인 경우 *expression*의 자료 유형은 INTEGER이어야 합니다. 표현식이 널값인 경우 0 값이 리턴됩니다.

NULL

SQL 함수로부터 널값이 리턴됩니다. 널(null)은 SQL 프로시듀어에서 허용되지 않습니다.

query-expression

SQL 표 함수에서 리턴된 표를 나타냅니다. *fullselect*는 SQL 스칼라 함수 및 SQL 프로시듀어로 허용되지 않습니다.

common-table 표현식

fullselect와 함께 사용할 common-table-expression을 지정합니다.

fullselect

표 함수에 대해 리턴될 행을 지정합니다. fullselect의 열 수는 함수 결과의 열 수와 일치해야 합니다. fullselect는 0개 이상의 열과 함께 0개 이상의 행을 리턴할 수 있습니다.

주

값이 프로시듀어로부터 리턴될 때 호출자는 다음을 사용하여 그 값에 액세스할 수 있습니다.

- SQL 프로시듀어가 다른 SQL 프로시듀어나 SQL 함수로부터 호출될 때 RETURN_STATUS를 검색하기 위한 GET DIAGNOSTICS문
- ODBC 어플리케이션의 escape절 CALL 구문(?=CALL...)의 리턴 값 매개변수 마커용으로 바인드된 매개변수
- SQLCODE가 0보다 작지 않을 때 sqlerrd[0] 값을 검색함으로써 SQL 프로시듀어에 대한 CALL 처리로부터 리턴되는 SQLCA로부터 직접 사용하는 경우. SQLCODE가 0보다 작으면, sqlerrd[0]는 설정되지 않으며 어플리케이션은 RETURN_STATUS 값을 -1이라고 가정합니다.

RETURN문이 프로시듀어로부터 리턴하는 데 사용되지 않았거나 값이 RETURN문에 지정되지 않은 경우

- 프로시듀어가 0 이상의 SQLCODE를 리턴할 경우, RETURN_STATUS는 0 값으로 설정됩니다.
- 프로시듀어가 0보다 작은 SQLCODE를 리턴할 경우, RETURN_STATUS는 -1 값으로 설정됩니다.

지정된 리턴값과 함께 RETURN문이 프로시듀어로부터 리턴할 때 사용된 경우 SQLCA의 SQLCODE, SQLSTATE 및 메세지 텍스트는 0으로 초기화됩니다. 오류는 호출자에게 리턴되지 않습니다.

RETURN은 SQL 트리거에서 사용할 수 없습니다.

SQL 표 함수 명령문 루틴 본문에는 단 하나의 RETURN문만 허용됩니다.

예

RETURN문을 사용하여 SQL 프로시듀어로부터 리턴하며 성공한 경우에는 상태 값이 0, 실패한 경우에는 상태 값이 -200입니다.

```
BEGIN
  ...
  GOTO fail;
  ...
```

```
   success: RETURN 0
   failure: RETURN -200
...
END
```

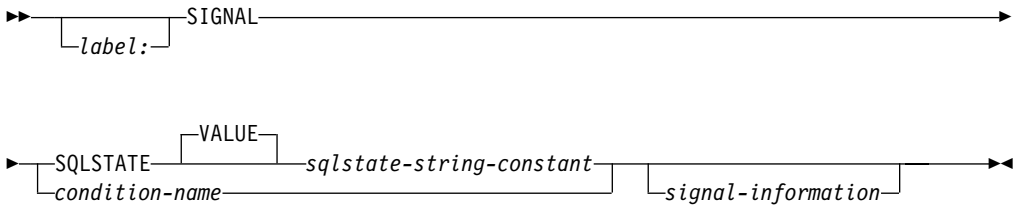
기존의 사인 및 코사인 함수를 사용하여 탄젠트 값을 리턴하는 스칼라 함수를 정의합니다.

```
CREATE FUNCTION mytan (x DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
 NO EXTERNAL ACTION
 DETERMINISTIC
  RETURN SIN(x)/COS(x)
```

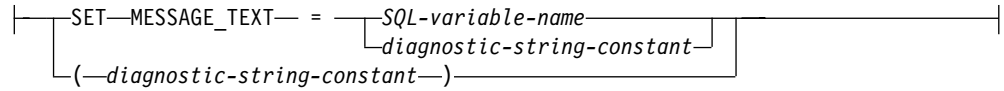
SIGNAL문

SIGNAL문은 오류 조건이나 경고 조건을 신호합니다. 이 명령을 사용하면 오류나 경고가 지정된 SQLSTATE 및 선택적 메세지 텍스트와 함께 리턴됩니다.

구문



signal-information:



설명

label

SIGNAL문의 레이블을 지정합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다.

SQLSTATE VALUE *sqlstate-string-constant*

신호될 SQLSTATE를 지정합니다. 스트링은 정확히 5 문자 스트링 상수로 지정되어야 하며, SQLSTATE에 대한 규칙을 따라야 합니다.

- 각 문자는 숫자 집합((‘0’ - ‘9’) 또는 대문자(‘A’ - ‘Z’)여야 합니다.
- ‘00’은 성공적인 완료를 의미하므로 SQLSTATE 클래스(처음 2자)는 ‘00’이어서는 안됩니다.

SQLSTATE가 해당 규칙을 따르지 않는다면 오류가 표시됩니다.

condition-name

신호될 조건 이름을 지정합니다. *condition-name*은 복합문 내에 선언되어야 합니다.

MESSAGE_TEXT

오류나 경고를 설명하는 스트링을 지정합니다. 이 스트링은 SQLCA의 SQLERRMC 필드에 리턴됩니다. 스트링의 실제 길이가 70바이트를 넘어가면 경고 없이 절단됩니다.

SQL-variable-name
복합 명령문 내에서 선언되어야 할 SQL 변수를 식별합니다. SQL 변수는 CHAR이나 VARCHAR 자료 유형으로 정의되어야 합니다.

diagnostic-string-constant
메세지 텍스트가 들어 있는 문자 스트링 상수를 지정합니다.

(diagnostic-string-constant)
메세지 텍스트가 들어 있는 스트링 상수를 지정합니다. 이 형식은 SQL 트리거 본문에서만 지원됩니다. ANS와 ISO 표준에 따르려면 이 양식은 사용하면 안 됩니다. 이것은 다른 제품과의 호환성을 위해 제공된 것입니다.

주

유효한 SQLSTATE값은 무엇이든 SGINAL문에 사용될 수 있습니다. 그러나 어플리케이션에 예약된 범위에 기초하여 프로그래머가 새로운 SQLSTATE를 정의하는 것이 좋습니다. 이렇게 함으로써 차후 릴리스에서 데이터베이스 관리자에 의해 정의될 수도 있는 SQLSTATE값을 의도와 달리 사용하는 것을 막을 수 있습니다. SQLSTATE에 대한 자세한 정보는 iSeries Information Center의 SQL 메세지 및 코드 책을 참조하십시오.

SIGNAL문이 실행되면, 다음과 같이 SQLCA에 리턴된 SQLCODE는 SQLSTATE 값에 기초하여 설정됩니다.

- 지정된 SQLSTATE 클래스가 '01' 또는 '02'인 경우 경고 또는 찾을 수 없음이 신호되고 SQLCODE가 +438로 설정됩니다.
- 그렇지 않으면 예외는 신호가 되고 SQLCODE는 -438로 설정됩니다.

SQLSTATE나 조건에서 예외('01'이나 '02'가 아닌 SQLSTATE 클래스) 신호가 발생하는 경우 다음 처리가 발생합니다.

- 핸들러가 SIGNAL문과 같은 복합 명령문에 있고 이 복합 명령문에 SQLEXCEPTION에 대한 핸들러 또는 지정된 SQLSTATE나 조건이 들어 있는 경우 예외는 처리되고 제어는 해당 핸들러로 넘어갑니다.
- 그렇지 않으면 예외는 처리되지 않고 제어는 즉시 복합문 끝으로 리턴됩니다.

SQLSTATE나 조건에서 경고(SQLSTATE 클래스 '01') 또는 찾을 수 없음(SQLSTATE 클래스 '02') 신호가 발생하는 경우

- 핸들러가 SIGNAL문과 같은 복합문에 있고 이 복합에 SQLWARNING(SQLSTATE 클래스가 '01'인 경우), NOT FOUND (SQLSTATE 클래스가 '02'인 경우) 또는 지정된 SQLSTATE나 조건에 대한 핸들러가 들어 있는 경우, 경고 또는 찾을 수 없음 조건이 처리되고 제어는 핸들러로 넘어갑니다.
- 그렇지 않으면 경고는 처리되지 않고 다음 명령문을 계속 처리합니다.

SQLSTATE 값은 2자로 된 클래스 코드 값과 그 뒤에 나오는 3자로 된 서브클래스 코드 값으로 구성됩니다. 클래스 코드 값은 성공한 실행 조건 및 성공하지 못한 실행 조건의 클래스를 나타냅니다.

유효한 SQLSTATE값은 무엇이든 SGINAL문에 사용될 수 있습니다. 그러나 어플리케이션에 예약된 범위에 기초하여 프로그래머가 새로운 SQLSTATE를 정의하는 것이 좋습니다. 이렇게 함으로써 차후 릴리스에서 데이터베이스 관리자에 의해 정의될 수도 있는 SQLSTATE값을 의도와 달리 사용하는 것을 막을 수 있습니다.

- '7'에서 '9'까지 또는 'I'에서 'Z'까지의 문자로 시작되는 SQLSTATE 클래스를 정의할 수 있습니다. 이러한 클래스 내에 서브클래스를 정의할 수 있습니다.
- '0'에서 '6'까지 또는 'A'에서 'H'까지의 문자로 시작되는 SQLSTATE 클래스는 데이터베이스 관리자가 사용하도록 예약되어 있습니다. 이 클래스 내에서 '0'에서 'H'까지의 문자로 시작되는 서브클래스는 데이터베이스 관리자가 사용하도록 예약되어 있습니다. 'I'에서 'Z'까지의 문자로 시작되는 서브클래스를 정의할 수 있습니다.

SQLSTATE에 대한 자세한 정보는 iSeries Information Center의 SQL 메세지 및 코드 책을 참조하십시오.

예

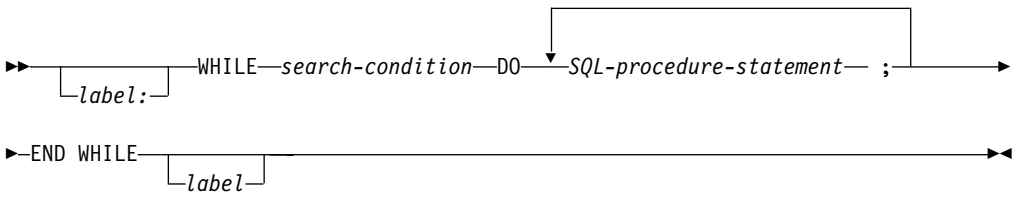
다음의 복합 명령문에서 매개변수 *rating*이 프로시듀어로 전달됩니다. 이 회사에서 근무한 기간이 6개월 미만인 경우 예외 II001이 즉시 호출자에게 리턴됩니다.

```
CREATE PROCEDURE raise ( IN rating INTEGER )
  LANGUAGE SQL
BEGIN
       DECLARE new_salary DECIMAL(9,2);
       DECLARE service DECIMAL(8,0);
            DECLARE v_empno CHAR(6) DEFAULT '123456';
            SELECT salary, current_date - hiredate
               INTO new_salary, service
               FROM employee
               WHERE empno = v_empno;
            IF service < 600
              THEN SIGNAL SQLSTATE 'II001'
                     SET MESSAGE_TEXT = 'Insufficient time in service.';
            END IF;
            IF rating = 1
             THEN SET new_salary =
                          new_salary + (new_salary * .10);
             ELSEIF rating = 2
             THEN SET new_salary =
                          new_salary + (new_salary * .05);
            END IF;
            UPDATE employee
               SET salary = new_salary
               WHERE empno = v_empno;
    END;
```

WHILE문

WHILE문은 지정된 조건이 참인 동안 명령문 실행을 반복합니다.

구문



설명

label

코드 블록에 대해 레이블을 정의합니다. SQL 함수, SQL 프로시듀어 또는 SQL 트리거 내에서 레이블이 고유해야 하며 레이블이 사용된 SQL 함수, SQL 프로시듀어 또는 SQL 트리거의 이름과 같을 수 없습니다. 종료 레이블이 지정되는 경우 LEAVE문에서 지정될 수 있습니다.

종료 레이블이 지정되는 경우 시작 레이블과 동일해야 합니다.

search-condition

*search-condition*은 WHILE 루프 실행 전에 평가됩니다. 조건이 참이면, WHILE 루프의 *SQL-procedure-statements*이 실행됩니다.

*SQL-PROCEDURE*문

WHILE 루프에서 실행될 SQL문을 지정합니다.

예

본 예제는 WHILE문을 사용하여 FETCH 및 SET문을 반복합니다. SQL 변수 *v_counter*의 값이 IN 매개변수 *deptNumber*로 지정된 부서의 사원 수의 반보다 적으면, WHILE문은 FETCH문 및 SET문을 계속 실행합니다. 조건이 더 이상 참이 아닌 경우, 제어 흐름은 WHILE문을 떠나고 커서를 닫습니다.

```
CREATE PROCEDURE dept_median
   (IN deptNumber SMALLINT, OUT medianSalary DECIMAL(7,2))
   LANGUAGE SQL
BEGIN
       DECLARE v_numRecords INTEGER DEFAULT 1;
       DECLARE v_counter INTEGER DEFAULT 0;
          DECLARE c1 CURSOR FOR
            SELECT salary
          FROM staff
          WHERE dept = deptNumber
```

```
   ORDER BY salary;
DECLARE EXIT HANDLER FOR NOT FOUND
   SET medianSalary = 6666;
SET medianSalary = 0;
SELECT COUNT(*) INTO v_numRecords
   FROM staff
   WHERE dept = deptNumber;
   OPEN c1;
WHILE v_counter < (v_numRecords/2 + 1) DO
   FETCH c1 INTO medianSalary;
   SET v_counter = v_counter +1;
  END WHILE;
  CLOSE c1;
   END
```

부록 A. SQL 한계

다음 표는 iSeries용 DB2 UDB데이터베이스 관리자가 지정한 특정 한계를 설명한 것입니다.

표 58. ID 길이 한계

ID 한계	iSeries용 DB2 UDB 한계
가장 긴 별명	128
가장 긴 권한부여명	10
가장 긴 열 레이블	60
가장 긴 상관명	128
가장 긴 커서명	18
가장 긴 호스트 ID	64
가장 긴 저장점명	128
가장 긴 서버명	18
가장 긴 SQL 루틴 레이블	128
가장 긴 명령문 이름	18
가장 긴 표, 패키지 또는 별명 레이블	50
가장 긴 비규정 스키마명	10
가장 긴 비규정 열 이름	30
가장 긴 비규정 제약조건명	128
가장 긴 비규정 자료 유형 이름	128
가장 긴 비규정 외부 프로그램명[68]	10
가장 긴 비규정 함수명	128
가장 긴 비규정 노드 그룹명	10
가장 긴 비규정 패키지명	10
가장 긴 비규정 프로시듀어명	128
가장 긴 비규정 특정명	128
가장 긴 비규정 SQL 매개변수명	128
가장 긴 비규정 SQL 변수명	128
가장 긴 비규정 표, 뷰 및 색인명	128
가장 긴 비규정 트리거명	128
규정되지 않은 시스템 열 이름	10
비규정 시스템 표, 뷰 및 색인명	10

68. 서비스 프로그램 입력점 이름에 대한 한계는 279입니다. REXX 프로시듀어에 대한 한계는 33입니다.

표 59. 숫자 한계

숫자 한계	iSeries용 DB2 UDB 한계
최소 BIGINT 값	-9 223 372 036 854 775 808
가장 긴 BIGINT 값	+9 223 372 036 854 775 807
최소 INTEGER 값	-2 147 483 648
최대 INTEGER 값	+2 147 483 647
최소 SMALLINT 값	-32 768
최대 SMALLINT 값	+32 767
최대 십진수 정밀도	31
최소 FLOAT 값[70]	-1.79×10^{308}
최대 FLOAT 값[70]	$+1.79 \times 10^{308}$
최소 양수 FLOAT 값[70]	$+2.23 \times 10^{-308}$
최대 양수 FLOAT 값[70]	-2.23×10^{-308}
최소 REAL 값[70]	-3.4×10^{38}
최대 REAL 값[70]	$+3.4 \times 10^{38}$
최소 양수 REAL 값[70]	$+1.18 \times 10^{-38}$
최대 양수 REAL 값[70]	-1.18×10^{-38}

표 60. 스트링 한계

스트링 한계	iSeries용 DB2 UDB 한계
BOLB 최대 길이	2 147 483 647
CHAR 최대 길이[71]	32765
VARCHAR 최대 길이[71]	32739
CLOB 최대 길이	2 147 483 647
C NUL-종결 최대 길이[71]	32739
GRAPHIC 최대 길이[71]	16382
VARGRAPHIC 최대 길이[71]	16369
DBCLOB 최대 길이	1 073 741 823
C NUL-종결 그래픽 최대 길이[71]	16369
문자 상수 최대 길이	32740
그래픽 상수 최대 길이	16370
가장 긴 연결 문자 스트링[71]	32765
가장 긴 연결 그래픽 스트링[71]	16369

표 61. 날짜시간 한계

날짜시간 한계	iSeries용 DB2 UDB 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	0001-01-01-00.00.00.000000
최대 TIMESTAMP 값	9999-12-31-24.00.00.000000

표 62. 자료 링크 한계

자료 링크 한계	iSeries용 DB2 UDB 한계
DATALINK 최대 길이	32718
DATALINK 주석 최대 길이	254

표 63. 데이터베이스 관리자 한계

데이터베이스 관리자 한계	iSeries용 DB2 UDB 한계
표 안의 최대 열	8000
뷰 안의 최대 열	8000
함수 안의 매개변수 최대수	90
프로시듀어 안의 매개변수 최대수	254[73]
전체 오버헤드를 포함한 LOB가 없는 행의 최대 길이	32766
전체 오버헤드를 포함한 LOB가 있는 행의 최대 길이	3 758 096 383
표 최대 크기	1 terabyte
색인 최대 크기	1 terabyte
표 안의 최대 행	4 294 967 288
가장 긴 색인 키	2000
색인 키 안의 최대 열	120
테이블에 대한 최대 색인	약 4000
SQL문에서 참조된 최대 표	256
SQL 뷰에서 참조된 최대 표	32
사전컴파일된 프로그램의 최대 호스트 변수 선언[69]	기억장치
SQL문 안의 최대 호스트 변수 및 상수	4096[74]
삽입 또는 갱신에 사용된 가장 긴 호스트 변수	32766
가장 긴 SQL문	65535
선택 리스트의 최대 요소[72]	약 8000
WHERE 또는 HAVING절 안의 최대 술어	4690
GROUP BY절 안의 열 최대수	120
GROUP BY절 안의 열 길이 최대 총계	2000
ORDER BY절 안의 열 최대수	10000
ORDER BY절 안의 열 최대 길이 총계	10000
SQLDA 최대 크기	16 777 215
준비된 명령문 최대수	기억장치
프로그램 안의 최대 선언 커서	기억장치
한 번에 열리는 커서의 최대수	기억장치
관계형 데이터베이스(RDB) 안의 최대 표	기억장치
표에 대한 제약 조건 최대수	300
subselect가 허락된 최대 레벨	32
주석 최대 길이	2000
경로 최대 길이	3483
경로에서 스키마 최대수	268

표 63. 데이터베이스 관리자 한계 (계속)

데이터베이스 관리자 한계	iSeries용 DB2 UDB 한계
한 작업 단위에서 변경된 행 최대수	500 000 000
표에 대한 트리거 최대수	300
내포된 트리거 호출 최대수	200
결과 세트를 갖는 최대 프로시듀어가 페치되기를 대기 중임	100
암호 최대 길이	128
트랜잭션에서 로케이터의 최대 수	250 000
한 번에 활성화된 최대 저장점 수	기억장치
프로세스에서 동시에 할당된 CLI 핸들 최대 수	80 000

69. 이전의 매개변수 전달 기술이 사용할 경우 RPG/400 및 PL/I 프로그램에서는 그 한계가 약 4000개입니다. 한계는 프로그램에 허용되는 포인터의 갯수를 기준으로 한 것입니다. 그 외 모든 경우에서의 한계는 오퍼레이팅 시스템 내의 구조적인 제약 조건에 따른 것입니다.

70. 아래에 보여진 값들은 대략적인 것입니다.

71. 열이 NOT NUL이면, 최대 *크기*는 하나 더 있습니다.

72. 한계는 분석된 SQL문에 대해 생성된 내부 구조의 *크기*를 기준으로 결정됩니다.

73. PARAMETER STYLE SQL 프로시듀어는 90개의 매개변수로 제한됩니다. PARAMETER STYLE GENERAL에 대한 SQL 프로시듀어는 253개로 제한됩니다. PARAMETER STYLE GENERAL WITH NULLS에 대한 프로시듀어는 254로 제한됩니다. PARAMETER STYLE GENERAL에 대한 외부 프로시듀어는 255로 제한됩니다. 매개변수의 최대수는 외부 프로그램을 컴파일하기 위해 사용되는 사용권 프로그램이 허용하는 매개변수의 최대수로 제한됩니다.

74. 명령문이 읽기 전용이 아닌 경우, 한계는 2048입니다. 한계는 대략적 값으로 매우 큰 스트링 상수 또는 스트링 변수가 사용된 경우 더 적을 수 있습니다.

75. DRDA 연결당 할당된 최대 핸들러 수는 500입니다.

부록 B. SQL 통신 영역

SQLCA은 모든 SQL문 실행 말기에 갱신되는 변수들의 집합입니다. 실행가능한 SQL 문을 포함한 프로그램은 반드시 하나의 SQLCA를 정확하게 제공해야만 합니다(그렇지 않으면 독립 SQLCODE 또는 독립 SQLSTATE 변수가 대신 사용됩니다).

SQL INCLUDE 명령문은 RPG 또는 REXX를 제외한 모든 호스트 언어에서 SQLCA의 선언을 제공하는 데 사용할 수 있습니다. REXX 프로시듀어에서 SQLCA 사용에 대한 내용은 SQL Programming with Host Languages 책을 참조하십시오.

C, COBOL, FORTRAN 및 PL/I에서 기억장치 영역명은 반드시 SQLCA이어야 합니다. PL/I과 C에서 구조명은 반드시 SQLCA이어야만 합니다. 모든 SQL문은 반드시 그 선언의 범위 내에 있어야 합니다.

독립 SQLCODE가 프로그램에 명시될 때 SQLCA는 결코 포함되어서는 안됩니다. 사전컴파일러에는 SQLCADE로 변경된 변수 SQLCODE의 이름과 SQLCA가 포함됩니다(또는 SQLCAD로 변경된 SQLCOD). 사전컴파일러는 독립 SQLCODE가 올바른 값을 가지도록 하기 위해 프로그램에 명령문을 추가합니다.

독립 SQLSTATE가 프로그램에 명시된 경우 SQLCA는 절대 포함되어서는 안됩니다. 사전컴파일러는 SQLSTATE로 변경된 변수 SQLSTATE의 이름과 SQLCA를 포함합니다. 사전컴파일러는 독립 SQLSTATE가 올바른 값을 가지도록 하기 위해 프로그램에 명령문을 추가합니다.

독립 SQLCODE 및 독립 SQLSTATE는 절대 RPG 또는 REXX에서 지정하지 마십시오.

필드 설명

다음 표의 이름들은 SQL INCLUDE문이 제공하는 이름들입니다. 대부분의 경우 C(및 C++), COBOL, FORTRAN 및 PL/I는 동일한 이름을 사용합니다. RPG/400에서 이름은 6자로 제한되기 때문에 RPG 명은 다르게 됩니다. PL/I명이 COBOL 명과 구분되는 인스턴스의 경우를 참조하십시오.

표 64. SQL INCLUDE문이 제공하는 이름

C명, COBOL 및 PL/I명	**FORTRAN[1] 명**	**RPG 명**	**필드 자료 유형**	**필드 값**
SQLCAID sqlcaid	사용되지 않는 SQLCAID	SQLAID	CHAR(8)	'SQLCA'가 들어 있는 기억장치 덤프에 대한 『표시』.
SQLCABC sqlcabc	사용되지 않는 SQLCABC	SQLABC	INTEGER	SQLCA의 길이 136이 들어 있습니다.

표 64. SQL INCLUDE문이 제공하는 이름 (계속)

C명, COBOL 및 PL/I명	FORTRAN[1] 명	RPG 명	필드 자료 유형	필드 값
SQLCODE sqlcode	SQLCOD SQLCODE	SQLCOD	INTEGER	SQL 리턴 코드가 들어 있습니다. **코드** **의미** **0** SQLWARN 인디케이터의 설정에 상관없는 성공적인 실행 **양수** 성공적인 실행 그러나 경고 상태 **음수** 오류 상태
SQLERRML[2] sqlerrml	SQLTXL SQLERRML	SQLERL	SMALLINT	0에서 70에 이르는 범위에서 SQLERRMC에 대한 길이 인디케이터 0은 SQLERRMC의 값이 적절치 않음을 의미합니다.
SQLERRMC[2] sqlerrmc	SQLTXT SQLERRMC	SQLERM	CHAR (70)	SQLCODE와 연관된 메세지 대체 텍스트가 들어 있습니다. CONNECT 및 SET CONNECTION의 경우 SQLERRMC 필드에 연결에 대한 정보가 들어 있습니다. 대체 텍스트에 대한 설명은 876 페이지의 표 67을 참조하십시오.
SQLERRP sqlerrp	SQLERP SQLERRP	SQLERP	CHAR(8)	오류를 리턴하는 제품 및 모듈명이 들어 있습니다. 처음 세 문자가 제품을 식별합니다. VM 및 VSE용 DB2에 대한 ARI DSN for OS/390 및 z/OS용 DB2 UDB QSQ for iSeries용 DB2 UDB 그 외의 모든 DB2 제품용 SQL 자세한 내용은 421 페이지의 『CONNECT(유형 1)』 또는 427 페이지의 『CONNECT(유형 2)』를 참조하십시오.
SQLERRD sqlerrd	SQLERR SQLERRD	SQLERR[3]	배열	진단 정보를 제공하는 6개의 INTEGER 변수가 들어 있습니다. 진단 정보에 대한 설명은 874 페이지의 표 66을 참조하십시오.
SQLWARN sqlwarn	SQLWRN SQLWARN	SQLWRN[4]	CHAR(11)	11개 CHAR(1) 경고 인디케이터의 세트로서 각각 공백 또는 'W' 또는 'N'이 들어 있습니다.
SQLSTATE sqlstate	SQLSTT SQLSTATE	SQLSTT	CHAR(5)	가장 최근에 실행된 SQL문의 결과를 표시하는 리턴 코드

주:

1 FORTRAN SQLCA에 대한 IBM SQL SQLCA명을 표시하는 첫 번째 이름. 두 번째 이름은 iSeries용 DB2 UDBFORTRAN에서의 SQLCA의 실행으로 인하여 사용할 수 있게 된 대체 이름을 표시합니다.

2 COBOL에서 SQLERRM은 SQLERRML 및 SQLERRMC을 포함합니다. PL/I에서 가변 길이 스트링 SQLERRM은 SQLERRMC의 접두부인 SQLEERML에 해당합니다.

3 RPG/400 및 ILE RPG/400에서 SQLERR은 SQLER6을 통한 SQLER1 필드에 의해 재정의된 24 문자로 정의됩니다(배열이 아닙니다). 필드는 2진 문자입니다. ILE RPG/400에서 SQLERR 또한 배열로 재정의됩니다. 배열명은 SQLERRD입니다.

4 11문자로 정의됩니다(배열이 아닙니다).

표 65. SQLWARN 진단 정보

C 명 COBOL 명 & PL/I명	FORTRAN[1] 명	RPG 명	필드 값
SQLWARN0 sqlwarn[0]	SQLWRN(0) SQLWARN(1:1)	SQLWN0	다른 모든 인디케이터가 공백일 경우 공백입니다. 만일 적어도 다른 하나의 인디케이터에 'W' 또는 'N'이 들어 있을 경우 'W'가 들어갑니다.
SQLWARN1 sqlwarn[1]	SQLWRN(1) SQLWARN(2:2)	SQLWN1	만일 스트링 열의 값이 호스트 변수에 지정될 때 절단되었다면 'W'가 들어갑니다. 만일 *NOCNULRQD가 CRTSQLCI 또는 CRTSQLCPPI 명령(또는 CNULRQD(*NO)가 SET OPTION문에)에 명시되었거나 스트링 열 값이 C-NUL 종결 호스트 변수에 지정되었고, 호스트 변수가 결과를 포함할 만큼 크지만 NUL 종결자를 포함할 만큼 크지는 않을 경우 ‘N’이 들어갑니다.
SQLWARN2 sqlwarn[2]	SQLWRN(2) SQLWARN(3:3)	SQLWN2	만일 널값이 함수의 인수로부터 제거되었다면 'W'가 들어갑니다. 반드시 MIN 함수에 대해 'W'로 설정될 필요는 없는데, 그 결과가 널값 제거에 영향을 받지 않기 때문입니다.
SQLWARN3 sqlwarn[3]	SQLWRN(3) SQLWARN(4:4)	SQLWN3	열 수가 호스트 변수의 수보다 클 경우에는 'W'가 들어갑니다.
SQLWARN4 sqlwarn[4]	SQLWRN(4) SQLWARN(5:5)	SQLWN4	만일 준비된 UPDATE 또는 DELETE문이 WHERE 절을 포함하지 않는 경우 'W'가 들어갑니다.
SQLWARN5 sqlwarn[5]	SQLWRN(5) SQLWARN(6:6)	SQLWN5	예약
SQLWARN6 sqlwarn[6]	SQLWRN(6) SQLWARN(7:7)	SQLWN6	날짜 산술에서 월말 조정이 일어나는 경우 ‘W’가 들어갑니다.
SQLWARN7 sqlwarn[7]	SQLWRN(7) SQLWARN(8:8)	SQLWN7	예약
SQLWARN8 sqlwarn[8]	SQLWRX(1) SQLWARN(9:9)	SQLWN8	만일 문자 변환 결과에 대체 문자가 들어 있을 경우 ‘W’가 들어갑니다.
SQLWARN9 sqlwarn[9]	SQLWRX(2) SQLWARN(10:10)	SQLWN9	예약
SQLWARNA sqlwarn[10]	SQLWRX(3) SQLWARN(11:11)	SQLWNA	예약

SQLCA

표 66. SQLERRD 진단 정보

C 명 **COBOL 명 &** **PL/I명**	**FORTRAN[1] 명**	**RPG 명**	**필드 값**
SQLERRD(1) sqlerrd[0]	SQLERR(1)	SQLER1	만일 SQLCODE가 0보다 적다면 CPF 이탈 메세지의 마지막 네 문자가 들어갑니다. 예를 들어 메세지가 CPF5715라면, X'F5F7F1F5'이 SQLERRD(1)에 들어갑니다.[1] 프로시듀어 호출의 경우 RETURN문에 지정된 리턴 상태 값이 들어 있습니다. RETURN문에 리턴 상태 값이 지정되지 않은 경우에는 그 결과가 다음과 같습니다. • 호출 명령문이 성공하면 0이 리턴됩니다. • 호출 명령문이 성공하지 못하면 -200이 리턴됩니다.
SQLERRD(2) sqlerrd[1]	SQLERR(2)	SQLER2	만일 SQL 코드가 0보다 적을 경우 CPD 진단 메세지의 마지막 네 문자가 들어갑니다.[1] CALL 명령문의 경우 SQLERRD(2)에는 결과 세트의 수가 들어 있습니다.
SQLERRD(3) sqlerrd[2]	SQLERR(3)	SQLER3	상태 명령문에 대한 CONNERCT에 대해, SQLERRD(3)는 연결 상태에 대한 정보를 포함합니다. 자세한 내용은 427 페이지의 『CONNECT(유형 2)』를 참조하십시오. INSERT, UPDATE 및 DELETE의 경우 해당 행의 수를 보여줍니다. FETCH 명령문의 경우 SQLERRD(3)에는 해당 행의 수가 들어 있습니다. PREPARE 명령문의 경우 선택된 행의 추정 수가 들어갑니다. 행 수가 2 147 483 647보다 크면 2 147 483 647이 리턴됩니다.

표 66. SQLERRD 진단 정보 (계속)

C 명 COBOL 명 & PL/I명	FORTRAN[1] 명	RPG 명	필드 값
SQLERRD(4) sqlerrd[3]	SQLERR(4)	SQLER4	PREPARE 명령문의 경우 각 실행에 필요한 자원의 추정 연관 수가 들어갑니다. 이 수는 색인, 파일 크기, CPU 모델 등의 현재 가용성에 따라 다양합니다. iSeries용 DB2 UDB 조회 최적자에 의해 선택된 액세스 계획에 대한 추정비용입니다. CONNECT 및 SET CONNECTION 명령문에 대해서 SQLERRD(4)에는 사용된 대화식의 유형 및 위탁될 수 있는 갱신의 실행가능 여부가 들어갑니다. 자세한 내용은 427 페이지의 『CONNECT(유형 2)』를 참조하십시오. CALL 명령문에 대해서 SQLERRD(4)에는 프로시듀어의 실패를 야기한 오류의 메세지 키가 들어 있습니다. QMHRTVPM API는 메세지 키에 대한 메세지 설명을 리턴하도록 사용될 수 있습니다. DELETE, INSERT 또는 UPDATE 명령문의 트리거 오류에 대해서 SQLERRD(4)에는 트리거 프로그램으로부터 신호를 받은 오류의 메세지 키가 들어 있습니다. QMHRTVPM API는 메세지 키에 대한 메세지 설명을 리턴하도록 사용될 수 있습니다. FETCH 명령문의 경우 SQLERRD(4)에는 검색된 행의 수가 들어갑니다.
SQLERRD(5) sqlerrd[4]	SQLERR(5)	SQLER5	CALL 명령문인 경우, SQLERRD(5)에는 프로시듀어에서 리턴된 결과 세트의 수가 들어 있습니다. CONNECT 또는 SET CONNECTION 명령문의 경우 SQLERRD(5)에는 다음 중 하나가 들어갑니다. • 연결이 실패한 경우 -1 • 로컬 연결인 경우 0 • 리모트 연결인 경우 1 DELETE 명령문의 경우 참조 제약 조건에 의해 영향받은 행의 수를 보여줍니다. EXECUTE IMMEDIATE 또는 PREPARE 명령문의 경우 구문 오류의 위치가 들어갈 수 있습니다. 복수 행 FETCH문의 경우 현재 표의 마지막 행이 페치되었다면 SQLERRD(5)에는 +100이 들어갑니다. PREPARE 명령문의 경우 SQLERRD(5)에는 준비된 명령문 안의 매개변수 마커 수가 들어갑니다.

표 66. SQLERRD 진단 정보 (계속)

C 명 COBOL 명 & PL/I명	FORTRAN[1] 명	RPG 명	필드 값
SQLERRD(6) sqlerrd[5]	SQLERR(6)	SQLER6	SQLCODE가 0일 때 SQL 완료 메세지 ID가 들어갑니다. 다른 모든 경우에 대해서는 정의되지 않습니다.
주: **1** SQLERRD(1) 및 SQLERRD(2)은 현재 서버가 iSeries용 DB2 UDB이며 적절한 경우에만 설정됩니다.			

표 67. CONNECT 및 SET CONNECTION에 대한 SQLERRMC 대체 텍스트

설명		자료 유형
관계형 데이터베이스(RDB)명		CHAR(18)
제품 ID(SQLERRP와 동일함)		CHAR(8)
서버 작업의 User ID		CHAR(10)
연결 메소드(*DUW 또는 *RUW)		CHAR(10)
DDM 서버 클래스명		CHAR(10)
QAS	iSeries용 DB2 UDB	
QDB2	OS/390 및 z/OS용 DB2 UDB	
QDB2/2	OS/2용 DB2	
QDB2/6000	AIX/6000용 DB2	
QDB2/HPUX	HP-UX**용 DB2	
QDB2/NT	NT용 DB2	
QDB2/SUN	SUN** Solaris**용 DB2	
QSQLDS/VM	VM 및 VSE용 DB2	
QSQLDS/VSE	VM 및 VSE용 DB2	
연결 유형(SQLERRD(4)와 동일함)		SMALLINT

INCLUDE SQLCA 선언

C 및 C++에서 INCLUDE SQLCA 선언은 다음과 같습니다.

```
#ifndef    SQLCODE
struct     sqlca
{
           unsigned char  sqlcaid[8];
           long           sqlcabc;
           long           sqlcode;
           short          sqlerrml;
           unsigned char  sqlerrmc[70];
           unsigned char  sqlerrp[8];
           long           sqlerrd[6];
           unsigned char  sqlwarn[11];
```

```
          unsigned char  sqlstate[5];
};
#define      SQLCODE   sqlca.sqlcode
#define      SQLWARN0  sqlca.sqlwarn[0]
#define      SQLWARN1  sqlca.sqlwarn[1]
#define      SQLWARN2  sqlca.sqlwarn[2]
#define      SQLWARN3  sqlca.sqlwarn[3]
#define      SQLWARN4  sqlca.sqlwarn[4]
#define      SQLWARN5  sqlca.sqlwarn[5]
#define      SQLWARN6  sqlca.sqlwarn[6]
#define      SQLWARN7  sqlca.sqlwarn[7]
#define      SQLWARN8  sqlca.sqlwarn[8]
#define      SQLWARN9  sqlca.sqlwarn[9]
#define      SQLWARNA  sqlca.sqlwarn[10]
#define      SQLSTATE  sqlca.sqlstate
#endif
struct sqlca sqlca;
```

COBOL에서 INCLUDE SQLCA 선언은 다음과 같습니다.

```
01 SQLCA.
   05 SQLCAID      PIC X(8).
   05 SQLCABC      PIC S9(9) BINARY.
   05 SQLCODE      PIC S9(9) BINARY.
   05 SQLERRM.
      49 SQLERRML  PIC S9(4) BINARY.
      49 SQLERRMC  PIC X(70).
   05 SQLERRP      PIC X(8).
   05 SQLERRD      OCCURS 6 TIMES
                             PIC S9(9) BINARY.
   05 SQLWARN.
      10 SQLWARN0  PIC X(1).
      10 SQLWARN1  PIC X(1).
      10 SQLWARN2  PIC X(1).
      10 SQLWARN3  PIC X(1).
      10 SQLWARN4  PIC X(1).
      10 SQLWARN5  PIC X(1).
      10 SQLWARN6  PIC X(1).
      10 SQLWARN7  PIC X(1).
      10 SQLWARN8  PIC X(1).
      10 SQLWARN9  PIC X(1).
      10 SQLWARNA  PIC X(1).
   05 SQLSTATE     PIC X(5).
```

주: COBOL에서는 INCLUDE SQLCA를 Working Storage Section 외부에 지정하면 안됩니다.

FORTRAN에서 INCLUDE SQLCA 선언은 다음과 같습니다.

```
 CHARACTER SQLCA(136)
 CHARACTER SQLCAID*8
 INTEGER*4 SQLCABC
 INTEGER*4 SQLCODE
 INTEGER*2 SQLERRML
 CHARACTER SQLERRMC*70
 CHARACTER SQLERRP*8
 INTEGER*4 SQLERRD(6)
 CHARACTER SQLWARN*11
```

```
 CHARACTER SQLSTOTE*5
 EQUIVALENCE (SQLCA(  1), SQLCAID)
 EQUIVALENCE (SQLCA(  9), SQLCABC)
 EQUIVALENCE (SQLCA( 13), SQLCODE)
 EQUIVALENCE (SQLCA( 17), SQLERRML)
 EQUIVALENCE (SQLCA( 19), SQLERRMC)
 EQUIVALENCE (SQLCA( 89), SQLERRP)
 EQUIVALENCE (SQLCA( 97), SQLERRD)
 EQUIVALENCE (SQLCA(121), SQLWARN)
 EQUIVALENCE (SQLCA(132), SQLSTOTE)

 INTEGER*4 SQLCOD,
C           SQLERR(6)
 INTEGER*2 SQLTXL
 CHARACTER  SQLERP*8,
C           SQLWRN(0:7)*1,
C           SQLWRX(1:3)*1,
C           SQLTXT*70,
C           SQLSTT*5,
C           SQLWRNWK*8,
C           SQLWRXWK*3,
C           SQLERRWK*24,
C           SQLERRDWK*24
 EQUIVALENCE (SQLWRN(1), SQLWRNWK)
 EQUIVALENCE (SQLWRX(1), SQLWRXWK)
 EQUIVALENCE (SQLCA(97), SQLERRDWK)
 EQUIVALENCE (SQLERR(1), SQLERRWK)
 COMMON /SQLCA1/SQLCOD,SQLERR,SQLTXL
 COMMON /SQLCA2/SQLERP,SQLWRN,SQLTXT,SQLWRX,SQLSTT
```

PL/I에서 INCLUDE SQLCA 선언은 다음과 같습니다.

```
DCL 1 SQLCA,
      2 SQLCAID      CHAR(8),
      2 SQLCABC      BIN FIXED(31),
      2 SQLCODE      BIN FIXED(31),
      2 SQLERRM      CHAR(70) VAR,
      2 SQLERRP      CHAR(8),
      2 SQLERRD(6)   BIN FIXED(31),
      2 SQLWARN,
        3 SQLWARN0   CHAR(1),
        3 SQLWARN1   CHAR(1),
        3 SQLWARN2   CHAR(1),
        3 SQLWARN3   CHAR(1),
        3 SQLWARN4   CHAR(1),
        3 SQLWARN5   CHAR(1),
        3 SQLWARN6   CHAR(1),
        3 SQLWARN7   CHAR(1),
        3 SQLWARN8   CHAR(1),
        3 SQLWARN9   CHAR(1),
        3 SQLWARNA   CHAR(1),
      2 SQLSTATE     CHAR(5);
```

RPG/400에서 SQLCA 선언은 다음과 같습니다.

```
ISQLCA          DS
I                                       1   8 SQLAID            SQL
I                                     B   9  120SQLABC            SQL
```

```
I                                     B  13  160SQLCOD            SQL
I                                     B  17  180SQLERL            SQL
I                                        19  88 SQLERM            SQL
I                                        89  96 SQLERP            SQL
I                                        97 120 SQLERR            SQL
I                                     B  97 1000SQLER1            SQL
I                                     B 101 1040SQLER2            SQL
I                                     B 105 1080SQLER3            SQL
I                                     B 109 1120SQLER4            SQL
I                                     B 113 1160SQLER5            SQL
I                                     B 117 1200SQLER6            SQL
I                                       121 131 SQLWRN            SQL
I                                       121 121 SQLWN0            SQL
I                                       122 122 SQLWN1            SQL
I                                       123 123 SQLWN2            SQL
I                                       124 124 SQLWN3            SQL
I                                       125 125 SQLWN4            SQL
I                                       126 126 SQLWN5            SQL
I                                       127 127 SQLWN6            SQL
I                                       128 128 SQLWN7            SQL
I                                       129 129 SQLWN8            SQL
I                                       130 130 SQLWN9            SQL
I                                       131 131 SQLWNA            SQL
I                                       132 136 SQLSTT            SQL
```

ILE RPG/400에서 SQLCA 선언은 다음과 같습니다.

```
D*      SQL Communications area
D SQLCA           DS
D  SQLAID                 1      8A
D  SQLABC                 9     12B 0
D  SQLCOD                13     16B 0
D  SQLERL                17     18B 0
D  SQLERM                19     88A
D  SQLERP                89     96A
D  SQLERRD               97    120B 0 DIM(6)
D  SQLERR                97    120A
D   SQLER1               97    100B 0
D   SQLER2              101    104B 0
D   SQLER3              105    108B 0
D   SQLER4              109    112B 0
D   SQLER5              113    116B 0
D   SQLER6              117    120B 0
D  SQLWRN                121    131A
D   SQLWN0               121    121A
D   SQLWN1               122    122A
D   SQLWN2               123    123A
D   SQLWN3               124    124A
D   SQLWN4               125    125A
D   SQLWN5               126    126A
D   SQLWN6               127    127A
D   SQLWN7               128    128A
D   SQLWN8               129    129A
D   SQLWN9               130    130A
D   SQLWNA               131    131A
D  SQLSTT                132    136A
D*  End of SQLCA
```

부록 C. SQLDA(SQL 설명자 영역)

하나의 SQLDA는 SQL DESCRIBE 명령문의 실행에 필요한 변수 세트이며, PREPARE, OPEN, CALL,FETCH 및 EXECUTE 명령문에 의해 선택적으로 사용될 수 있습니다. 하나의 SQLDA는 DESCRIBE 명령문에서 사용될 수 있으며, 호스트 변수의 주소와 함께 변환될 수 있고, 이후 FETCH 명령문에서 다시 사용될 수 있습니다.

모든 언어가 SQLDA를 지원하지만, 사전 정의된 선언은 C(및 C++), COBOL, ILE RPG/400, PL/I 및 REXX에서만 지원됩니다. REXX에서 SQLDA는 다른 언어에서와 약간 다릅니다. REXX에서 SQLDA 사용에 대한 내용은 호스트 언어로 SQL 프로그래밍 책을 참조하십시오.

하나의 SQLDA에 있는 정보의 의미는 그 사용에 따라 다릅니다. PREPARE 및 DESCRIBE에서 SQLDA는 준비된 명령문에 대한 어플리케이션 프로그램에 정보를 제공합니다. OPEN, CALL, EXECUTE 및 FETCH에서 하나의 SQLDA가 호스트 변수에 관해 데이터베이스 관리자에게 정보를 제공합니다.

필드 설명

하나의 SQLDA에는 묶어서 SQLVAR로 불리우는 일련의 다섯 개 변수가 임의의 수만큼 있고, 이어 있는 헤더 구조 안에는 네 개의 변수가 있습니다. OPEN, CALL, FETCH 및 EXECUTE에서 SQLVAR의 각 발생은 호스트 변수 하나를 나타냅니다. PREPARE 및 DESCRIBE에서 각 발생은 결과표의 열 하나를 나타냅니다.

SQL INCLUDE문은 다음의 필드명을 제공합니다.

76. 이 열에서 소문자명은 C명입니다. 대문자명은 COBOL, PL/I 또는 RPG명입니다.

SQLDA

표 68. SQLDA 헤더에 대한 필드 설명

C명[76] PL/I명 COBOL명	필드 자료 유형	DESCRIBE 및 PREPARE에서 사용 (SQLN을 제외하고 데이터베이스 관리자가 설정)	FETCH, OPEN, CALL 또는 EXECUTE에서 사용(명령문 실행 전에 사용자가 설정)
sqldaid SQLDAID	CHAR(8)	'SQLDA'가 들어 있는 기억장치 덤프에 대한 '표시'. SQLDAID의 일곱 번째 바이트는 각 열에 대해 필요한 SQLVAR 항목이 하나 이상인지의 여부를 결정하도록 사용될 수 있습니다. 자세한 내용은 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.	일곱 번째 바이트 안의 '2'는 각 열에 대해 두 개의 SQLVAR 항목이 할당되었음을 나타냅니다. 일곱 번째 바이트의 '3'은 세 SQLVAR 항목이 각 열에 대해 할당되었음을 나타냅니다. 일곱 번째 바이트의 '4'는 각 열에 대해 4개의 SQLVAR 항목이 할당되었음을 나타냅니다.
sqldabc SQLDABC	INTEGER	SQLDA의 길이	SQLDA에 할당된 기억장치 바이트 수 SQLN 발생이 들어갈 수 있도록 충분한 기억장치가 할당되어야만 합니다. SQLDABC는 16+SQLN*(80)과 같거나 그보다 큰 값으로 설정해야 하며 이때 80은 SQLVAR 발생의 길이입니다. LOB나 고유한 유형이 지정되는 경우 각 매개변수 마커에 대해 두 SQLVAR 항목이 있어야 합니다.
sqln SQLN	SMALLINT	데이터베이스 관리자에 의해 변경되지 않습니다. PREPARE 또는 DESCRIBE 명령문이 실행되기 전에 0보다 크거나 같은 값으로 설정되어야 합니다. 결과의 열의 수보다 크거나 같은 값으로 설정되거나 혹은 SQLVAR 항목의 복수 세트가 필요한 경우에는 결과의 열의 수의 중복으로 설정되어야 합니다. SQLVAR 총 발생 수를 나타냅니다.	SQLDA에 제공되는 SQLVAR 총 발생 수입니다. SQLN는 0보다 크거나 같은 값으로 설정되어야 합니다. LOB나 고유한 유형이 지정되는 경우 각 매개변수 마커에 대해 두 SQLVAR 항목이 있어야 하며 SQLN이 매개변수 마커 숫자의 두 배로 설정되어야 합니다.
sqld SQLD	SMALLINT	SQLVAR의 발생에서 서술된 열 수(서술된 명령문이 SELECT 명령문이 아닐 경우는 0)	이 명령문을 실행할 때 SQLDA에서 사용되는 SQLVAR에 의해 서술되는 호스트 변수의 수. SQLD는 영(0) 보다 크거나 같고 SQLN 보다 작거나 같은 값으로 설정되어야 합니다.

SQLVAR 발생에서의 필드 설명

SQLDA에 의해 서술된 각 열 또는 호스트 변수의 경우 두 가지 유형의 SQLVAR 항목이 있습니다.

기본 SQLVAR 항목

기본 SQLVAR 항목은 항상 있습니다. 이 항목의 필드에는 자료 유형 코드, 길

이 속성(LOB 제외), 열 이름(또는 레이블), 코드화 문자 세트 ID(CCSID), 호스트 변수 주소 및 인디케이터 변수 주소와 같은 열 또는 호스트 변수에 대한 기본 정보가 들어 있습니다.

확장(Extended) SQLVAR 항목

결과가 LOB 또는 고유한 유형의 열을 포함하고 있다면 확장 SQLVAR 항목이 필요합니다(각 열당). 고유한 유형의 경우 확장 SQLVAR에는 고유한 유형 이름이 들어 있습니다. LOB의 경우 확장 SQLVAR에는 호스트 변수와 실제 길이를 포함하는 버퍼의 포인터에 대한 길이 속성이 들어 있습니다. 로케이터나 파일 참조 변수가 LOB를 나타내기 위해 사용되는 경우 확장 SQLVAR이 필요없습니다.

다음의 경우 각 열에 대해 확장 SQLVAR 항목이 필요합니다.

- 열 이름과 레이블이 리턴되었음을 표시하는 USING BOTH이 명시된 경우
- 열 이름, 레이블 및 시스템 열 이름이 리턴되었음을 표시하는 USING ALL이 명시된 경우

LOB 및 고유한 유형 정보를 리턴하는 확장 SQLVAR의 필드는 중복되지 않습니다. LOB 및 레이블 정보를 리턴하는 필드도 중복되지 않습니다. 레이블, LOB 및 고유한 유형의 조합에 따라서, 정보를 리턴하기 위해 열 당 하나 이상의 확장 SQLVAR 항목이 필요할 수 있습니다. 884 페이지의 『SQLVAR 필수 발생 수 판별』을 참조하십시오.

표 69, 884 페이지의 표 70 및 884 페이지의 표 71는 기본 및 확장 SQLVAR 항목을 맵하는 방법을 보여줍니다. 기본 및 확장 SQLVAR 항목이 모두 들어간 SQLDA의 경우 기본 SQLVAR 항목은 첫 번째 블록에 위치하며 확장 SQLVAR 항목이 연이어 위치합니다. 필요할 경우 확장 SQLVAR 항목의 두 번째 혹은 세 번째 블록이 연이어 위치합니다. 확장 SQLVAR 항목의 대다수가 사용되지 않을 수도 있지만, 각 블록에서 SQLVAR 항목의 발생 수는 SQLD에서의 값과 일치합니다.

표 69. USING NAMES, USING SYSTEM NAMES, USING LABELS 또는 USING ANY에 대한 SQLVAR 배열의 내용

LOB	DISTINCT 유형	SQLDAID의 일곱 번째 바이트	SQLN 최소값	첫 번째 세트 (기본)	두 번째 세트 (확장)	세 번째 세트 (확장)	네 번째 세트 (확장)
아니오	아니오	공백	n	열 이름, 시스템 열 이름 또는 레이블	사용되지 않음	사용되지 않음	사용되지 않음
예	아니오	2	2n	열 이름, 시스템 열 이름 또는 레이블	LOB	사용되지 않음	사용되지 않음
아니오	예	2	2n	열 이름, 시스템 열 이름 또는 레이블	Distinct 유형	사용되지 않음	사용되지 않음

표 69. USING NAMES, USING SYSTEM NAMES, USING LABELS 또는 USING ANY에 대한 SQLVAR 배열의 내용 (계속)

LOB	DISTINCT 유형	SQLDAID의 일곱 번째 바이트	SQLN 최소값	첫 번째 세트 (기본)	두 번째 세트 (확장)	세 번째 세트 (확장)	네 번째 세트 (확장)
예	예	2	2n	열 이름, 시스템 열 이름 또는 레이블	LOB 및 고유한 유형	사용되지 않음	사용되지 않음

표 70. USING BOTH에 대한 SQLVAR 배열 내용

LOB	DISTINCT 유형	SQLDAID의 일곱 번째 바이트	SQLN 최소값	첫 번째 세트 (기본)	두 번째 세트 (확장)	세 번째 세트 (확장)	네 번째 세트 (확장)
아니오	아니오	2	2n	열 이름	레이블	사용되지 않음	사용되지 않음
예	아니오	2	2n	열 이름	LOB 및 레이블	사용되지 않음	사용되지 않음
아니오	예	3	3n	열 이름	Distinct 유형	레이블	사용되지 않음
예	예	3	3n	열 이름	LOB 및 고유한 유형	레이블	사용되지 않음

표 71. USING ALL에 대한 SQLVAR 배열 내용

LOB	DISTINCT 유형	SQLDAID의 일곱 번째 바이트	SQLN 최소값	첫 번째 세트 (기본)	두 번째 세트 (확장)	세 번째 세트 (확장)	네 번째 세트 (확장)
아니오	아니오	3	3n	시스템 열 이름	레이블	열 이름	사용되지 않음
예	아니오	3	3n	시스템 열 이름	LOB 및 레이블	열 이름	사용되지 않음
아니오	예	4	4n	시스템 열 이름	Distinct 유형	레이블	열 이름
예	예	4	4n	시스템 열 이름	LOB 및 고유한 유형	레이블	열 이름

SQLVAR 필수 발생 수 판별

SQLVAR 필수 발생 수는 SQLDA가 제공되는 명령문 및 서술되는 열 또는 매개변수의 자료 유형에 따라 다릅니다. 자세한 내용은 위의 표를 참조하십시오.

SQLDAID의 일곱 번째 바이트는 항상 필수적인 SQLVAR 세트 수로 설정됩니다.

SQLD이 SQLVAR 발생 충족 수로 설정되지 않았다면 다음과 같은 경우입니다.

- SQLD는 모든 세트에 필요한 SQLVAR 총 발생 수로 지정되었습니다.
- 적어도 충분한 SQLVAR가 기본 SQLVAR 항목에 대해 명시된 경우 경고 (SQLSTATE 01594)가 SQLCA의 SQLCODE 필드에 리턴됩니다. 기본 SQLVAR 항목이 리턴됩니다. 하지만 확장 SQLVAR는 리턴되지 않습니다.
- 충분한 SQLVAR가 기본 SQLVAR 항목에 대해 명시된 경우 경고(SQLSTATE 01005)가 SQLCA의 SQLCODE 필드에 리턴됩니다. SQLVAR 항목은 리턴되지 않습니다.

표 72. SQLVAR에 대한 필드 설명

C명[77] **COBOL명** **PL/I명** **RPG 명**	필드 자료 유형	**DESCRIBE 및 PREPARE에서 사용(데이터베이스 관리자가 설정)**	**FETCH, OPEN, CALL 및 EXECUTE에서 사용(명령문 실행 전에 사용자가 설정)**
sqltype SQLTYPE	SMALLINT	열의 자료 유형과 널 값을 가지고 있는지 여부를 표시합니다. 유형 코드에 대한 설명은 887 페이지의 표 74를 참조하십시오. 고유한 유형의 경우 고유한 유형이 기준으로 하는 자료 유형이 이 필드에 나타납니다. 기본 SQLVAR에는 이 필드가 고유한 유형에 대한 설명의 일부라는 표시가 없습니다.	호스트 변수의 자료 유형과 인디케이터 변수가 제공되었는지 여부를 표시합니다. 유형 코드에 대한 설명은 887 페이지의 표 74를 참조하십시오.
sqllen SQLLEN	SMALLINT	열의 길이 속성. 날짜시간 열의 경우 값의 스트링 표시의 길이. 887 페이지의 표 74를 참조하십시오. LOB의 경우 LOB 길이 속성과 무관하게 값은 0입니다. 확장 SQLVAR 항목 내의 SQLLONGLEN 필드에는 LOB의 길이 속성이 들어갑니다.	호스트 변수의 길이 속성. 887 페이지의 표 74를 참조하십시오. LOB의 경우 LOB 길이 속성과 무관하게 값은 0입니다. 확장 SQLVAR 항목 내의 SQLLONGLEN 필드에는 LOB의 길이 속성이 들어갑니다.
sqlres SQLRES	CHAR(12)	예약. SQLCATA에 대한 경계 정렬을 제공합니다.	예약. SQLCATA에 대한 경계 정렬을 제공합니다.
sqldata SQLDATA	포인터	889 페이지의 표 75에 서술된 스트링 열의 코드화 문자 세트 ID(CCSID)	호스트 변수의 주소가 들어갑니다. LOB 호스트 변수의 경우 확장 SQLVAR의 SQLDATALEN 필드가 널(null)이면 이것은 LOB 자료를 바로 뒤따르는 4바이트 LOB 길이를 가리킵니다. 확장 SQLVAR의 SQLDATALEN 필드가 널(null)이 아니면 이것은 LOB 자료를 가리키고 SQLDATALEN 필드는 4바이트 LOB 길이를 가리킵니다.
sqlind SQLIND	포인터	예약	인디케이터 변수의 주소가 들어갑니다. 인디케이터 변수가 없을 경우 사용되지 않습니다(SQLTYPE의 짝수 값으로 표시됩니다).
sqlname SQLNAME	VARCHAR (30)	규정되지 않은 열 이름. 열에 이름이 없으면 스트링은 표현식에서 만들어져 리턴됩니다. 이름은 대소문자 구분되며 둘러싼 분리 문자를 포함하지 않습니다.	889 페이지의 표 75에 서술된 대로 호스트 변수의 코드화 문자 세트 ID(CCSID)가 들어갑니다.

77. 이 열에서 소문자명은 C명입니다. 대문자명은 PL/I, COBOL 및 RPG명입니다.

SQLDA

표 73. 확장 SQLVAR에 대한 필드 설명

C명 [78] **COBOL명** **PL/I명** **RPG 명**	**필드 자료 유형**	**DESCRIBE 및 PREPARE에서 사용(데이터베이스 관리자가 설정)**	**FETCH, OPEN, CALL 및 EXECUTE에서 사용(명령문 실행 전에 사용자가 설정)**
len.sqllonglen SQLLONGL SQLLONGLEN	INTEGER	LOB 열의 길이 속성	LOB 호스트 변수의 길이 속성. 데이터베이스 관리자는 이러한 자료 유형에 대한 기본 SQLVAR 내의 SQLLEN 필드를 무시합니다. 길이 속성은 BLOB 또는 CLOB에 대한 바이트 수 및 DBCLOB에 대한 문자 수를 표시합니다.
*	CHAR(12)	예약. SQLDATALEN에 대한 경계 정렬을 제공합니다.	예약. SQLDATALEN에 대한 경계 정렬을 제공합니다.
*	포인터	예약.	예약.
sqldatalen SQLDATAL SQLDATALEN	포인터	사용되지 않음.	LOB 호스트 변수에 대해서만 사용. 이 필드의 값이 널(null)이면 LOB의 실제 길이가 일치하는 기초 SQLVAR의 SQLDATA 필드에 의해 지시되는 첫 번째 4바이트에 저장되며 LOB 자료는 4바이트 길이의 바로 뒤에 옵니다. 실제 길이는 BLOB 또는 CLOB에 대한 바이트 수 및 DBCLOB에 대한 2바이트 문자 수를 표시합니다. 이 필드의 값이 널(null)이 아닌 경우 이 필드는 (DBCLOB의 경우에도) 바이트 단위로 LOB의 실제 길이를 포함하는 4바이트 길이의 버퍼를 가리킵니다. 그런 다음 일치하는 기초 SQLVAR의 SQLDATA 필드가 LOB 자료를 가리킵니다. 이 필드 사용 여부에 관계 없이, 필드 SQLLONGLEN은 반드시 설정되어야 합니다.

표 73. 확장 SQLVAR에 대한 필드 설명 (계속)

C명 [78] COBOL명 PL/I명 RPG 명	필드 자료 유형	DESCRIBE 및 PREPARE에서 사용(데이터베이스 관리자가 설정)	FETCH, OPEN, CALL 및 EXECUTE에서 사용(명령문 실행 전에 사용자가 설정)
sqldatatype_name SQLTNAME SQLDATATYPE-NAME	VARCHAR (30)	확장 SQLVAR의 SQLTNAME 필드는 다음 중 하나로 설정됩니다. • 고유한 유형 열에 대해서, 데이터베이스 관리자는 이 필드를 완전히 규정된 고유한 유형 이름으로 설정합니다. 만일 규정된 이름이 30 바이트보다 길다면 절단됩니다. • 레이블에 대해서, 데이터베이스 관리자는 이 필드를 레이블의 첫 20바이트로 설정합니다. • 열 이름에 대해서, 데이터베이스 관리자는 이 필드를 열 이름으로 설정합니다.	사용되지 않음.

SQLTYPE 및 SQLLEN

다음 표는 SQLDA의 SQLTYPE 및 SQLLEN 필드에 나타날 수 있는 값을 보여줍니다. PREPARE 및 DESCRIBE에서 SQLTYPE의 짝수 값은 열이 널(null) 값을 허용하지 않음을 의미하며, 홀수 값은 열이 널 값을 허용함을 의미합니다.

주: PREPARE 및 DESCRIBE 명령문에서 하나의 피연산자가 널값이 가능하거나 또는 표현식이 -2 맵핑 오류 널값을 초래하는 경우에 그 표현식에 대해 홀수 값이 리턴됩니다.

FETCH, OPEN, CALL 및 EXECUTE에서 SQLTYPE의 짝수 값은 제공된 인디케이터 변수가 없음을 의미하며, 홀수 값은 SQLIND에 인디케이터 변수 주소가 들어 있음을 의미합니다.

표 74. PREPARE, DESCRIBE, FETCH, OPEN, CALL 또는 EXECUTE에 대한 SQLTYPE 및 SQLLEN 값

	PREPARE 및 DESCRIBE의 경우		FETCH, OPEN, CALL 및 EXECUTE의 경우	
SQLTYPE	**열 자료 유형**	**SQLLEN**	**호스트 변수 자료 유형**	**SQLLEN**
384/385	날짜	10	날짜의 고정 길이 문자 스트링 표시	호스트 변수의 길이 속성
388/389	시간	8	시간의 고정 길이 문자 스트링 표시	호스트 변수의 길이 속성
392/393	시간소인	26	시간소인의 고정 길이 문자 스트링 표시	호스트 변수의 길이 속성

78. 이 열에서 소문자명은 C명입니다. 첫 번째 대문자명은 PL/I 및 RPG명입니다. 두 번째 대문자명은 COBOL명입니다.

표 74. PREPARE, DESCRIBE, FETCH, OPEN, CALL 또는 EXECUTE에 대한 SQLTYPE 및 SQLLEN 값 (계속)

	PREPARE 및 DESCRIBE의 경우		FETCH, OPEN, CALL 및 EXECUTE의 경우	
SQLTYPE	**열 자료 유형**	**SQLLEN**	**호스트 변수 자료 유형**	**SQLLEN**
396/397	DataLink	열의 길이 속성	DataLink	호스트 변수의 길이 속성
400/401	적용 안됨	적용 안됨	NUL-종결 그래픽 스트링	호스트 변수의 길이 속성
404/405	BLOB	0[80]	BLOB	사용되지 않음.[80]
408/409	CLOB	0[80]	CLOB	사용되지 않음.[80]
412/413	DBCLOB	0[80]	DBCLOB	사용되지 않음.[80]
448/449	가변 길이의 문자 스트링	열의 길이 속성	가변 길이의 문자 스트링	호스트 변수의 길이 속성
452/453	고정 길이 문자 스트링	열의 길이 속성	고정 길이 문자 스트링	호스트 변수의 길이 속성
456/457	가변 길이의 긴 문자 스트링	열의 길이 속성	가변 길이의 긴 문자 스트링	호스트 변수의 길이 속성
460/461	적용 안됨	적용 안됨	NUL-종결 문자 스트링	호스트 변수의 길이 속성
464/465	가변 길이의 그래픽 스트링	열의 길이 속성	가변 길이의 그래픽 스트링	호스트 변수의 길이 속성
468/469	고정 길이 그래픽 스트링	열의 길이 속성	고정 길이 그래픽 스트링	호스트 변수의 길이 속성
472/473	가변 길이의 긴 그래픽 스트링	열의 길이 속성	긴 그래픽 스트링	호스트 변수의 길이 속성
476/477	적용 안됨	적용 안됨	PASCAL L-스트링	호스트 변수의 길이 속성
480/481	부동 소수점	단정밀도의 경우 4, 배정밀도의 경우 8.	부동 소수점	단정밀도의 경우 4, 배정밀도의 경우 8.
484/485	압축 십진수	바이트 1에 정밀도, 바이트 2에 스케일	압축 십진수	바이트 1에 정밀도, 바이트 2에 스케일
488/489	존(zone) 십진수	바이트 1에 정밀도, 바이트 2에 스케일	존(zone) 십진수	바이트 1에 정밀도, 바이트 2에 스케일
492/493	큰 정수	8[79]	큰 정수	8
496/497	큰 정수	4[79]	큰 정수	4
500/501	작은 정수	2[79]	작은 정수	2
504/505	적용 안됨	적용 안됨	분리되어 표시되는 화면 부호	바이트 1에 정밀도, 바이트 2에 스케일
904/905	ROWID	40	ROWID	40
960/961	적용 안됨	적용 안됨	BLOB 로케이터	4
964/965	적용 안됨	적용 안됨	CLOB 로케이터	4
968/969	적용 안됨	적용 안됨	DBCLOB 로케이터	4
916/917	적용 안됨	적용 안됨	BLOB 파일 참조 변수	267
920/921	적용 안됨	적용 안됨	CLOB 파일 참조 변수	267
924/925	적용 안됨	적용 안됨	DBCLOB 파일 참조 변수	267

79. 2진수는 2, 4 또는 8의 길이로 또는 바이트 1의 정밀도와 바이트 2의 스케일을 갖고 SQLDA에 표시될 수 있습니다. 첫 번째 바이트가 x'00'보다 크면, 정밀도와 스케일을 표시합니다.

80. 확장 SQLVAR 내의 SQLLONGLEN 필드에는 열의 길이 속성이 들어갑니다.

SQLDATA 또는 SQLNAME

OPEN, FETCH, CALL 및 EXECUTE 명령문에서 SQLVAR 요소의 SQLNAME 필드는 스트링 호스트 변수에 대한 코드화 문자 세트 ID(CCSID)를 명시하는 데 사용될 수 있습니다. 만일 SQLNAME 필드를 사용하여 코드화 문자 세트 ID(CCSID)를 지정할 경우 SQLNAME 길이는 8로 설정되어야 합니다. 또한 SQLNAME의 처음 4바이트는 다음 표에 나오는 설명에 따라 설정해야 합니다. 만일 코드화 문자 세트 ID(CCSID)를 지정하지 않으면 작업 코드화 문자 세트 ID(CCSID)가 사용됩니다.

DESCRIBE, DESCRIBE TABLE 및 PREPARE 명령문에서 SQLVAR 요소의 SQLDATA 필드에는 열이 스트링 열인 경우 결과표의 열에 대한 코드화 문자 세트 ID(CCSID)가 들어갑니다. 코드화 문자 세트 ID(CCSID)는 표 75에 서술된 대로 바이트 3 또는 4에 위치가 지정됩니다.

표 75. SQLDATA 또는 SQLNAME에 대한 코드화 문자 세트 ID(CCSID)

자료 유형	하위유형	바이트 1 & 2	바이트 3 & 4
문자	SBCS 자료	X'0000'	코드화 문자 세트 ID(CCSID)
문자	혼합된 자료	X'0000'	코드화 문자 세트 ID(CCSID)
문자	비트 자료	X'0000'	65535
그래픽	적용 안됨	X'0000'	코드화 문자 세트 ID(CCSID)
기타 자료 유형	적용 안됨	적용 안됨	적용 안됨

인식되지 않고 지원되지 않는 SQLTYPES

SQLDA의 SQLTYPE 필드에 나타나는 값은 자료의 수신자뿐 아니라 송신자에서 사용할 수 있는 자료 유형 지원의 레벨에 따라 다릅니다. 이것은 특히 신규 자료 유형이 제품에 추가될 때 중요합니다.

신규 자료 유형은 자료의 송신자 또는 수신자에 의해 지원되거나 지원되지 않을 수 있으며 자료의 송신자 또는 수신자에 의해 인식될 수 있거나 인식될 수 없을 수도 있습니다. 상황에 따라서 신규 자료 유형이 리턴되거나 자료의 송신자와 수신자 모두에 일치되는 호환 자료 유형이 리턴되거나 오류가 발생할 수 있습니다.

송신자와 수신자가 호환 자료 유형 사용에 일치할 때 다음은 발생할 맵핑을 표시합니다. 이 맵핑은 송신자나 수신자 중 적어도 하나가 제공된 자료 유형을 지원하지 않을 때 발생합니다. 지원되지 않는 자료 유형은 어플리케이션이나 데이터베이스 관리자에 의해 제공될 수 있습니다.

표 76. 지원되지 않는 자료 유형에 대한 호환 자료 유형

자료 유형	호환 자료 유형
BIGINT	DECIMAL(19,0)
ROWID	VARCHAR(40) FOR BIT DATA

INCLUDE SQLDA 선언

C 및 C++의 경우

C 및 C++에서 INCLUDE SQLDA 선언은 다음과 같습니다.

```
#ifndef SQLDASIZE
struct sqlda
{
       unsigned char  sqldaid[8];
       long           sqldabc;
       short          sqln;
       short          sqld;
       struct sqlvar
       {
              short          sqltype;
              short          sqllen;
              unsigned char *sqldata;
              short         *sqlind;
              struct sqlname
              {
                     short          length;
                          unsigned char  data[30];
              } sqlname;
       } sqlvar[1];
};

struct sqlvar2
       { struct
                {  long           sqllonglen;
                   char           reserve1[28];
                }  len;
                char *sqldatalen;
                struct sqldistinct_type
                       {  short          length;
                          unsigned char  data[30];
                       } sqldatatype_name;
       };

#define  SQLDASIZE(n) (sizeof(struct sqlda)+(n-1) * sizeof(struct sqlvar))
#endif
```

그림 11. C 및 C++에 대한 INCLUDE SQLDA 선언 (1/3)

```
/*************************************************************************/
/* Macros for using the sqlvar2 fields.                                  */
/*************************************************************************/

/*************************************************************************/
/*   '2' in the 7th byte of sqldaid indicates a doubled number of        */
/*       sqlvar entries.                                                 */
/*   '3' in the 7th byte of sqldaid indicates a tripled number of        */
/*       sqlvar entries.                                                 */
/*************************************************************************/
#define SQLDOUBLED '2'
#define SQLSINGLED ' '

/*************************************************************************/
/* GETSQLDOUBLED(daptr) returns 1 if the SQLDA pointed to by             */
/* daptr has been doubled, or 0 if it has not been doubled.              */
/*************************************************************************/
#define GETSQLDOUBLED(daptr) (((daptr)->sqldaid[6]==  \
 (char) SQLDOUBLED)  ?                                \
    (1)           :      \
    (0)             )

/*************************************************************************/
/* SETSQLDOUBLED(daptr, SQLDOUBLED) sets the 7th byte of sqldaid         */
/* to '2'.                                                               */
/* SETSQLDOUBLED(daptr, SQLSINGLED) sets the 7th byte of sqldaid         */
/* to be a ' '.                                                          */
/*************************************************************************/
#define SETSQLDOUBLED(daptr, newvalue)                \
  (((daptr)->sqldaid[6] =(newvalue)))

/*************************************************************************/
/* GETSQLDALONGLEN(daptr,n) returns the data length of the nth           */
/* entry in the sqlda pointed to by daptr. Use this only if the          */
/* sqlda was doubled or tripled and the nth SQLVAR entry has a           */
/* LOB datatype.                                                         */
/*************************************************************************/
#define GETSQLDALONGLEN(daptr,n) ((long) (((struct sqlvar2 *) \
&((daptr)->sqlvar[(n) +((daptr)->sqld)])) ->len.sqllonglen))

/*************************************************************************/
/* SETSQLDALONGLEN(daptr,n,len) sets the sqllonglen field of the         */
/* sqlda pointed to by daptr to len for the nth entry. Use this only     */
/* if the sqlda was doubled or tripled and the nth SQLVAR entry has      */
/* a LOB datatype.                                                       */
/*************************************************************************/
#define SETSQLDALONGLEN(daptr,n,length) {             \
 struct sqlvar2 *var2ptr;                             \
 var2ptr = (struct sqlvar2 *) &((daptr)->sqlvar[(n)+  \
   ((daptr)->sqld)]);                                 \
 var2ptr->len.sqllonglen = (long) (length);               \
}
```

그림 11. C 및 C++에 대한 INCLUDE SQLDA 선언 (2/3)

```
/*********************************************************************/
/* SETSQLDALENPTR(daptr,n,ptr) sets a pointer to the data length for */
/* the nth entry in the sqlda pointed to by daptr.                   */
/* Use this only if the sqlda has been doubled or tripled.           */
/*********************************************************************/
#define SETSQLDALENPTR(daptr,n,ptr) {                 \
 struct sqlvar2 *var2ptr;                             \
 var2ptr = (struct sqlvar2 *) &((daptr)->sqlvar[(n)+  \
   ((daptr)->sqld)]);                                 \
 var2ptr->sqldatalen = (char *) ptr;                  \
 }

/*********************************************************************/
/* GETSQLDALENPTR(daptr,n) returns a pointer to the data length for  */
/* the nth entry in the sqlda pointed to by daptr. Unlike the inline */
/* value (union sql8bytelen len), which is 8 bytes, the sqldatalen   */
/* pointer field returns a pointer to a long (4 byte) integer.       */
/* If the SQLDATALEN pointer is zero, a NULL pointer is be returned. */
/*                                                                   */
/* NOTE: Use this only if the sqlda has been doubled or tripled.     */
/*********************************************************************/
#define GETSQLDALENPTR(daptr,n) (                     \
  (((struct sqlvar2 *) &(daptr)->sqlvar[(n) +         \
    (daptr)->sqld])->sqldatalen == NULL) ?            \
   ((long *) NULL ) : ((long *) ((struct sqlvar2 *)   \
  &(daptr)->sqlvar[(n) + (daptr) ->sqld])->sqldatalen))
```

그림 11. C 및 C++에 대한 INCLUDE SQLDA 선언 (3/3)

COBOL의 경우

COBOL에서 INCLUDE SQLDA 선언은 다음과 같습니다.

```
1 SQLDA.
  05 SQLDAID     PIC X(8).
  05 SQLDABC     PIC S9(9) BINARY.
  05 SQLN        PIC S9(4) BINARY.
  05 SQLD        PIC S9(4) BINARY.
  05 SQLVAR OCCURS 0 TO 409 TIMES DEPENDING ON SQLD.
     10 SQLTYPE   PIC S9(4) BINARY.
     10 SQLLEN    PIC S9(4) BINARY.
     10 FILLER  REDEFINES SQLLEN.
        15 SQLPRECISION PIC X.
        15 SQLSCALE     PIC X.
     10 SQLRES    PIC X(12).
     10 SQLDATA   POINTER.
     10 SQLIND    POINTER.
     10 SQLNAME.
        49 SQLNAMEL PIC S9(4) BINARY.
        49 SQLNAMEC PIC X(30).
```

그림 12. COBOL에 대한 INCLUDE SQLDA 선언

ILE COBOL의 경우

ILE COBOL에서 INCLUDE SQLDA 선언은 다음과 같습니다.

```
1 SQLDA.
  05 SQLDAID     PIC X(8).
  05 SQLDABC     PIC S9(9) BINARY.
  05 SQLN        PIC S9(4) BINARY.
  05 SQLD        PIC S9(4) BINARY.
  05 SQLVAR OCCURS 0 TO 409 TIMES DEPENDING ON SQLD.
     10 SQLVAR1.
        15 SQLTYPE   PIC S9(4) BINARY.
        15 SQLLEN    PIC S9(4) BINARY.
        15 FILLER  REDEFINES SQLLEN.
           20 SQLPRECISION PIC X.
           20 SQLSCALE     PIC X.
        15 SQLRES    PIC X(12).
        15 SQLDATA   POINTER.
        15 SQLIND    POINTER.
        15 SQLNAME.
        49 SQLNAMEL PIC S9(4) BINARY.
        49 SQLNAMEC PIC X(30).
     10 SQLVAR2 REDEFINES SQLVAR1.
        15 SQLVAR2-RESERVED-1 PIC S9(9) BINARY.
        15 SQLLONGLEN         REDEFINES SQLVAR2-RESERVED-1
                              PIC S9(9) BINARY.
        15 SQLVAR2-RESERVED-2 PIC X(28).
        15 SQLDATALEN         POINTER.
        15 SQLDATATYPE-NAME.
           49 SQLDATATYPE-NAMEL PIC S9(4) BINARY.
           49 SQLDATATYPE-NAMEC PIC X(30).
```

그림 13. ILE COBOL에 대한 INCLUDE SQLDA 선언

PL/I의 경우

PL/I에서 INCLUDE SQLDA 선언은 다음과 같습니다.

```
DCL 1 SQLDA BASED(SQLDAPTR),
      2 SQLDAID     CHAR(8),
      2 SQLDABC     BIN FIXED(31),
      2 SQLN        BIN FIXED,
      2 SQLD        BIN FIXED,
      2 SQLVAR      (99),
        3 SQLTYPE   BIN FIXED,
        3 SQLLEN    BIN FIXED,
        3 SQLRES    CHAR(12),
        3 SQLDATA   PTR,
        3 SQLIND    PTR,
        3 SQLNAME   CHAR(30) VAR,

    1 SQLDA2 BASED(SQLDAPTR),
      2 SQLDAID2    CHAR(8),
      2 SQLDABC2    FIXED(31) BINARY,
      2 SQLN2       FIXED(15) BINARY,
      2 SQLD2       FIXED(15) BINARY,
      2 SQLVAR2     (99),
        3 SQLBIGLEN,
          4 SQLLONGL FIXED(31) BINARY,
          4 SQLRSVDL FIXED(31) BINARY,
        3 SQLDATAL  POINTER,
        3 SQLTNAME  CHAR(30) VAR;

    DECLARE SQLSIZE    FIXED(15) BINARY;
    DECLARE SQLDAPTR   PTR;
    DECLARE SQLDOUBLED CHAR(1)   INITIAL('2') STATIC;
    DECLARE SQLSINGLED CHAR(1)   INITIAL(' ') STATIC;
```

그림 14. PL/I에 대한 INCLUDE SQLDA 선언

ILE(Integrated Language Environment) RPG/400의 경우

ILE(Integrated Language Environment) RPG/400에서 INCLUDE SQLDA 선언은 다음과 같습니다.

```
D*      SQL Descriptor area
D SQLDA           DS
D  SQLDAID                 1      8A
D  SQLDABC                 9     12B 0
D  SQLN                   13     14B 0
D  SQLD                   15     16B 0
D  SQL_VAR                       80A   DIM(SQL_NUM)
D                         17     18B 0
D                         19     20B 0
D                         21     32A
D                         33     48*
D                         49     64*
D                         65     66B 0
D                         67     96A
D*
D SQLVAR          DS
D  SQLTYPE                 1      2B 0
D  SQLLEN                  3      4B 0
D  SQLRES                  5     16A
D  SQLDATA                17     32*
D  SQLIND                 33     48*
D  SQLNAMELEN             49     50B 0
D  SQLNAME                51     80A
D*
D SQLVAR2         DS
D  SQLLONGL                1      4B 0
D  SQLRSVDL                5     32A
D  SQLDATAL               33     48*
D  SQLTNAMELN             49     50B 0
D  SQLTNAME               51     80A
D*  End of SQLDA
```

그림 15. ILE(Integrated Language Environment) RPG/400에 대한 INCLUDE SQLDA 선언

사용자는 SQL_NUM 정의에 대해 책임이 있습니다. SQL_NUM은 SQL_VAR에 대해 필요한 차원과 함께 숫자 상수로 정의되어야 합니다.

RPG가 배열 안에서 구조를 지원하지 않으므로 SQLDA는 세 가지 자료 구조를 생성합니다. 두 번째 및 세 번째 자료 구조는 필드 설명이 들어 있는 SQLDA의 부분을 설정/참조하는 데 사용됩니다.

SQLDA의 필드 설명을 설정하기 위해 프로그램은 SQLVAR(또는 SQLVAR2)의 하위 필드에 필드 설명을 설정한 후 SQL_VAR,n에 SQLVAR(또는 SQLVAR2)의 MOVEA를 수행합니다. 여기서 n은 SQLDA에 있는 필드 수입니다. 이 실행은 모든 필드 설명이 설정될 때까지 반복됩니다.

SQLDA 필드 설명이 참조되어야 할 때 사용자는 SQLVAR(또는 SQLVAR2)에 SQL_VAR,n의 MOVEA를 실행합니다. 여기서 n은 처리될 필드 설명의 수입니다.

부록 D. 예약어

이것은 현재 iSeries용 DB2 UDB 예약 단어의 리스트입니다. 단어들은 언제든지 추가될 수 있습니다. 앞으로 예약어로 사용될 가능성이 추가 단어의 리스트는 *IBM SQL 참조서 버전 1* SC26-3255에서 IBM SQL 및 ANSI 예약어를 참조하십시오.

표 77. SQL 예약어

ADDALIASALLALLOCA	CURRENT_	GENERALGENERATED	MINUTEMINUTESMI
TEALLOW	TIMEZONECURRENT	GET	NVALUE
ALTERANDANYASA	_USER	GLOBAL	MODE
UTHORIZATIONBE	CURSORCYCLE	GO	MODIFIES
GINBETWEENBINARY	DATABASE	GOTOGRANTGRAPHIC	MONTHMONTHSNEW
BY	DAYDAYSDBINFODB2G	GROUP	NEW_TABLE
CACHE	ENERALDB2GENRL	HANDLER	NO
CALLCALLED	DB2SQLDECLAREDEF	HAVINGHOLDHOURH	NOCACHE
CARDINALITY	AULTDEFAULTS	OURSIDENTITY	NOCYCLE
CASECASTCCSIDCH	DEFINITION	IFIMMEDIATEININC	NODENAMENODENU
ARCHARACTER	DELETEDESCRIPTOR	LUDING	MBERNOMAXVALUE
CHECKCLOSECOLLEC	DETERMINISTICDI	INCREMENT	NOMINVALUE
TIONCOLUMNCOMMENT	SALLOW	INDEX	NOORDER
COMMITCONCATCONDI	DISCONNECTDISTI	INDICATORINNER	NOTNULLOFOLD
TIONCONNECT	NCTDODOUBLEDROPDY	INOUTINSENSITIVE	OLD_TABLE
CONNECTION	NAMICEACH	INSERTINTEGRITY	ON OPENOPTIMIZE
CONSTRAINTCONTAINS	ELSEELSEIFENDEND	INTOIS ISOLATION	OPTION
CONTINUECOUNTCOUNT	-EXEC(COBOL의	ITERATE	ORORDER
_BIGCREATECROSS	경우에만 적용)	JAVAJOINKEY	OUTOUTEROVER
CURRENT	ESCAPEEXCEPTIONE	LABEL	RIDING
CURRENT_	XCLUDING	LANGUAGELEAVELEF	PACKAGE
DATECURRENT_	EXECUTEEXISTSEXIT	TLIKELINKTYPE	PARAMETERPARTIT
PATHCURRENT_	EXTERNALFENCED	LOCK	IONPATH
SERVERCURRENT_	FETCHFILE	LONG	POSITIONPREPAR
TIMECURRENT_	FINAL	LOOPMAXVALUE	EPRIMARY
TIMESTAMP	FORFOREIGN	MICROSECONDMICR	PRIVILEGESPROCE
	FREE	OSECONDS	DUREPROGRAM
	FROMFUNCTION		READREADS
			RECOVERY
			REFERENCESREFER
			ENCING

표 78. SQL 예약어 (계속)

RELEASERENAMEREPEA	RRNRUN	SQLID	UNTILUPDATEUSAG
TRESETRESIGNAL	SAVEPOINT	START	EUSERUSINGVALUE
RESTART	SCHEMASCRATCHPADS	STATIC	SVARIABLEVARIANT
RESULT	ECONDSECONDSS	SUBSTRINGSYNONYM	VIEWWHENWHEREWHI
RETURNRETURNSREVO	ELECT	TABLETHENTOTRA	LEWITHWRITE
KERIGHT	SETSIGNAL	NSACTION	YEARYEARS
ROLLBACKROUTINERO	SIMPLE	TRIGGER	
WROWS	SOMESOURCESPECIF	TRIMTYPE	
	ICSQL	UNDOUNIONUNIQUE	

부록 E. 코드화 문자 세트 ID(CCSID) 값

다음 표에서는 코드화 문자 세트 ID(CCSID) 및 IBM 관계형 데이터베이스(RDB) 제품에 의해 제공되는 변환에 대해 설명합니다.

- OS/390 및 z/OS용 DB2 UDB 및 VM 및 VSE용 DB2에 대해서 이 표들은 CCSID와 카탈로그에서 초기에 제공되는 CCSID 변환표 쌍을 표시합니다. 관리자 권한을 갖는 사용자는 언제든지 SBCS 코드화 문자 세트 ID(CCSID) 및 SBCS 변환표를 추가할 수 있습니다. 또한 SBCS 또는 DBCS 변환을 수행하는 사용자 종료 루틴을 제공할 수 있습니다.
- iSeries용 DB2 UDB 및 DB2 UDB UWO에 대해서, 이 도표들은 사용할 수 있는 코드화 문자 세트 ID(CCSID) 및 변환표만을 표시합니다. 부가적인 코드화 문자 세트 ID(CCSID) 또는 변환표를 추가할 방법은 없습니다.

다음 리스트는 다음 표 내의 IBM 관계형 데이터베이스(RDB) 제품 열에서 사용된 기호를 정의한 것입니다.

X 해당 코드화 문자 세트 ID(CCSID)에서 또는 그 CCSID로 변환하기 위한 변환표가 있음을 나타냅니다.

C 해당 코드화 문자 세트 ID(CCSID)에서 다른 코드화 문자 세트 ID(CCSID)로 변환하기 위한 변환표가 있음을 나타냅니다. 또한 C는 해당 코드화 문자 세트 ID(CCSID)가 태그 로컬 자료에 사용될 수 없음을 나타냅니다. 이것은 코드화 문자 세트 ID(CCSID)가 외부 코드화 체계에 속하기 때문입니다(예를 들어 850같은 PC-자료 코드화 문자 세트 ID(CCSID)를 iSeries용 DB2 UDB 내의 태그 로컬 자료에 사용할 수 없습니다).

T 자료가 해당 코드화 문자 세트 ID(CCSID)로 태그 처리되었을 가능성이 있으나 제품과 함께 변환표를 제공하지 않음을 나타냅니다.

이 부록에 나오는 다음 제품 버전에 대한 것입니다.

- OS/390용 DB2 Universal Database 버전 7
- VM 및 VSE용 DB2 버전 7.1
- AS/400용 DB2 Universal Database 버전 5 릴리스 2
- OS/2용 DB2 Universal Database 버전 7
- AIX/6000용 DB2 Universal Database 버전 7
- HP용 DB2 Universal Database 버전 7
- SUN용 DB2 Universal Database 버전 7
- NT 및 Windows 95용 DB2 Universal Database 버전 7

- SCO Open Server용 DB2 Universal Database 버전 7

표 79. 범용 문자 세트 (UCS-2, UTF-16 및 UTF-8)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1200	UTF-16	X		C	C	C	C	C	C	
1208	UTF-8 레벨 3	X		C	X	X	X	X	X	
13488	UCS-2 레벨 1	C		X	X	X	X	X	X	

표 80. EBCDIC 그룹 1(라틴-1) 국가에 대한 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
37	미국, 캐나다 (S/370), 네덜란드, 포르투갈, 브라질, 호주, 뉴질랜드	X	X	X	C	C	C	C	C	C
256	워드 프로세싱, 네덜란드	T	T	X						
273	오스트리아, 독일	X	X	X	C	C	C	C	C	C
277	덴마크, 노르웨이	X	X	X	C	C	C	C	C	C
278	핀란드, 스웨덴	X	X	X	C	C	C	C	C	C
280	이탈리아	X	X	X	C	C	C	C	C	C
284	스페인, 라틴 아메리카(스페인어권)	X	X	X	C	C	C	C	C	C
285	영국	X	X	X	C	C	C	C	C	C
297	프랑스	X	X	X	C	C	C	C	C	C
500	벨기에 캐나다 (AS/400), 스위스, 국제 라틴-1 지역.	X	X	X	C	C	C	C	C	C
871	아이슬랜드	X	X	X	C	C	C	C	C	C
924	라틴-0	T	T	X						
1140	미국, 캐나다 (S/370), 네덜란드, 포르투갈, 브라질, 호주, 뉴질랜드	T	T	X						
1141	오스트리아, 독일	T	T	X						
1142	덴마크, 노르웨이	T	T	X						
1143	핀란드, 스웨덴	T	T	X						
1144	이탈리아	T	T	X						

표 80. EBCDIC 그룹 1(라틴-1) 국가에 대한 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1145	스페인, 라틴 아메리카(스페인어권)	T	T	X						
1146	영국	T	T	X						
1147	프랑스	T	T	X						
1148	벨기에 캐나다 (AS/400), 스위스, 국제 라틴-1 지역.	T	T	X						
1149	아이슬랜드	T	T	X						

표 81. PC-자료 및 ISO 그룹 1(라틴-1) 국가에 대한 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
437	USA	X	C	C	X	C	C	C	C	C
819	라틴-1 국가(ISO 8859-1)	X	C	C	C	X	X	X	C	X
850	라틴 알파벳 번호1; 라틴-1 국가	X	C	C	X	X	C	C	C	C
858	라틴 알파벳 번호1; 라틴-1 국가(Euro 포함)	T	T	C						
860	포르투갈(850 서브세트)	C	C	C	X	C	C	C	C	C
861	아이슬랜드	C		C						
863	캐나다(850 서브세트)	C	C	C	X	C	C	C	C	C
865	덴마크, 노르웨이, 핀란드, 스웨덴	C	C	C						
923	라틴-0			C						
1009	IRV 7-비트			C						
1010	프랑스 7-비트			C						
1011	독일 7-비트			C						
1012	이탈리아 7-비트			C						
1013	영국 7-비트			C						
1014	스페인 7-비트			C						
1015	포르투갈 7-비트			C						
1016	노르웨이 7-비트			C						

표 81. PC-자료 및 ISO 그룹 1(라틴-1) 국가에 대한 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1017	덴마크 7-비트			C						
1018	핀란드 및 스웨덴 7-비트			C						
1019	벨기에 및 네덜란드 7-비트			C						
1051	HP 에뮬레이션	X		C	C	C	X	C	C	
1252	Windows** 라틴-1	X	C	C	C	C	C	C	X	C
1275	Macintosh** 라틴-1	X		C	C	C	C	C	C	C

표 82. EBCDIC 그룹 1a(라틴-1이 아닌 SBCS) 국가에 대한 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
420	아랍어(유형 4)Visual LTR	X	X	X	C	C	C	C	C	C
423	그리스어	X	X	X	C	C	C	C	C	C
424	히브리어(유형 4)	X	X	X	C	C	C	C	C	C
870	라틴-2 복수 언어 사용	X	X	X	C	C	C	C	C	C
875	그리스어	X	X	X	C	C	C	C	C	C
880	러시아어 복수 언어 사용	T	T	X						
905	터키 라틴-3 복수 언어 사용	T	T	X						
918	힌두어	T	T	X						
1025	러시아어 복수 언어 사용	X	X	X	C	C	C	C	C	C
1026	터키 라틴-5	X	T	X	C	C	C	C	C	C
1097	이란어	T	T	X						
1112	발틱 복수 언어 사용	X	X	X	C	C	C	C	C	C
1122	에스토니아어	T	X	X	C	C	C	C	C	C
1123	우크라이나어	T	X	X	C	C	C	C	C	C
1137	Devanagari	T	T	X						
1153	라틴-2(Euro 포함)	T	T	X						

표 82. EBCDIC 그룹 1a(라틴-1이 아닌 SBCS) 국가에 대한 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1154	러시아어(Euro 포함)	T	T	X						
1155	터키 라틴-5(Euro 포함)	T	T	X						
1156	발리어(Euro 포함)	T	T	X						
1157	에스토니아어(Euro 포함)	T	T	X						
1158	우크라이나어(Euro 포함)	T	T	X						
4971	그리스어(Euro 포함)	T	T	X						
8612	아랍어(유형 5)			X						
8616	히브리어(유형 6)			X						
12708	아랍어(유형 7)			X						
62211	히브리어(유형 5)			X	C	C	C	C	C	C
62224	아랍어(유형 6)			X	C	C	C	C	C	C
62229	히브리어(유형 8)				C	C	C	C	C	C
62233	아랍어(유형 8)				C	C	C	C	C	C
62234	아랍어(유형 9)				C	C	C	C	C	C
62235	히브리어(유형 10)			X	C	C	C	C	C	C
62240	히브리어(유형 11)				C	C	C	C	C	C
62245	히브리어(유형 10)			X						

스트링 유형:

4 Visual / Left-to-Right / Shaped

5 Implicit / Left-to-Right / Unshaped

6 Implicit / Right-to-Left / Unshaped

7 Visual / Contextual / Unshaped

8 Visual / Right-to-Left / Shaped

9 Visual / Right-to-Left / Shaped

10 Implicit / Contextual-Left

11 Implicit / Contextual-Right

코드화 문자 세트 ID(CCSID) 값

표 83. PC-자료 및 ISO 그룹 1a(라틴-1이 아닌 SBCS) 국가에 대한 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
720	아랍어(MS-Dos)			C						
737	그리스어(MS-Dos)			C	C	C	C	C	C	C
775	발틱어(MS-Dos)			C						
813	그리스어/라틴어 (ISO 8859-7)	C	C	C	X	X	X	C	C	X
851	그리스어			C						
852	라틴-2 복수 언어 사용	C	C	C	X	C	C	C	C	C
855	러시아어 복수 언어 사용		C	C	X	C	C	C	C	C
856	아랍어(유형 5)			C						
857	터키 라틴-5	C		C	X	C	C	C	C	C
862	히브리어(유형 10)	C	C	C	X	C	C	C	C	C
864	아랍어(유형 5)	C	C	C	X	C	C	C	C	C
866	러시아어		C	C	X	C	C	C	C	C
868	힌두어			C						
869	그리스어	C	C	C	X	C	C	C	C	C
878	러시아 인터넷			C						
912	라틴-2(ISO 8859-2)	C	C	C	C	X	X	C	C	X
914	라틴-4(ISO 8859-4)			C						
915	러시아어 복수 언어 사용(ISO 8859-5)		C	C	X	X	X	C	C	X
916	히브리어/라틴(ISO 8859-8) (유형 5)	C	C	C	C	X	C	C	C	C
920	터키 라틴-5 (ISO 8859-9)	C		C	C	X	X	C	C	X
921	발틱 8-비트	C		C	X	X	C	C	X	C
922	에스토니아 8-비트			C	X	X	C	C	X	C
1008	아랍어 8-비트 ISO			C						
1046	아랍어(유형 5)	C		C	C	X	C	C	C	C
1089	아랍어(ISO 8859-6) (유형 5)	C		C	C	X	X	C	C	C
1098	이란어			C						
1124	우크라이나 8-비트 ISO			C						
1125	우크라이나			C	X	C	C	C	C	C

표 83. PC-자료 및 ISO 그룹 1a(라틴-1이 아닌 SBCS) 국가에 대한 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1131	벨라러스어			C	X	C	C	C	C	C
1250	Windows 라틴-2	C	C	C	C	C	C	C	X	C
1251	Windows 러시아어	C	C	C	C	C	C	C	X	C
1253	Windows 그리스어	C	C	C	C	C	C	C	X	C
1254	Windows 터키어	C		C	C	C	C	C	X	C
1255	Windows 히브리어 (유형 5)	C	C	C	C	C	C	C	X	C
1256	Windows 아랍어 (유형 5)	C	C	C	C	C	C	C	X	C
1257	Windows 발틱어	C		C						
1280	Macintosh** 그리스어	C		C	C	C	C	C	C	C
1281	Macintosh** 터키어	C		C	C	C	C	C	C	C
1282	Macintosh** 라틴-2	C		C	C	C	C	C	C	C
1283	Macintosh** 러시아어	C		C	C	C	C	C	C	C
4948	라틴-2 복수 언어 사용			C						
4951	러시아어 복수 언어 사용			C						
4952	히브리어			C						
4953	터키 라틴-5			C						
9056	아랍어(기억장치 교환)			C						
4960	아랍어			C						
4965	그리스어			C						
62208	히브리어(유형 4)				X	X	X	X	X	X
62209	히브리어(유형 4)			C	X	C	C	C	C	C
62210	히브리어/라틴(ISO 8859-8) (유형 4)			C	C	X	X	C	C	C
62213	히브리어(유형 5)			C	X	C	C	C	C	C
62215	Windows 히브리어 (유형 4)			C	C	C	C	C	X	C
62218	아랍어(유형 4)			C	X	C	C	C	C	C
62220	히브리어(유형 6)				X	X	X	X	X	X
62221	히브리어(유형 6)			C	X	C	C	C	C	C

코드화 문자 세트 ID(CCSID) 값

표 83. PC-자료 및 ISO 그룹 1a(라틴-1이 아닌 SBCS) 국가에 대한 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
62222	히브리어/라틴(ISO 8859-8) (유형 6)			C	C	X	X	C	C	C
62223	Windows 히브리어 (유형 6)			C	C	C	C	C	X	C
62225	아랍어(유형 6)				X	C	C	C	C	C
62226	아랍어(유형 6)				C	X	C	C	C	C
62227	아랍어(ISO 8859-6) (유형 6)				C	X	X	C	C	C
62228	Windows 아랍어 (유형 6)			C	C	C	C	C	X	C
62230	히브리어(유형 8)				X	X	X	X	X	X
62231	히브리어(유형 8)				X	C	C	C	C	C
62232	히브리어/라틴(ISO 8859-8) (유형 8)				C	X	X	C	C	C
62236	히브리어(유형 10)				X	X	X	X	X	X
62238	히브리어/라틴(ISO 8859-8) (유형 10)			C	C	X	X	C	C	C
62239	Windows 히브리어 (유형 10)			C	C	C	C	C	X	C
62241	히브리어(유형 11)				X	X	X	X	X	X
62242	히브리어(유형 11)				X	C	C	C	C	C
62243	히브리어/라틴(ISO 8859-8) (유형 11)				C	X	X	C	C	C
62244	Windows 히브리어 (유형 11)				C	C	C	C	X	C

스트링 유형:

4 Visual / Left-to-Right / Shaped

5 Implicit / Left-to-Right / Unshaped

6 Implicit / Right-to-Left / Unshaped

7 Visual / Contextual / Unshaped

8 Visual / Right-to-Left / Shaped

9 Visual / Right-to-Left / Shaped

10 Implicit / Contextual-Left

11 Implicit / Contextual-Right

표 84. EBCDIC 그룹 2(DBCS) 국가에 대한 SBCS 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
290	일본어 가타카나(확장)	X	X	X	C	C	C	C	C	C
833	한글(확장)	X	X	X	C	C	C	C	C	C
836	중국어(확장)	X	X	X	C	C	C	C	C	C
838	타이어(확장)	X	X	X	C	C	C	C	C	C
1027	일본어 영어(확장)	X	X	X	C	C	C	C	C	C
1130	베트남어	T	X	X						
1132	라오스어	T	X	X						
1160	타이어 (Euro 포함)	T	T	X						
1164	베트남어 (Euro 포함)	T	T	X						
5123	일본어 (Euro 포함)	T	T	X						
9030	타이어(확장)	T	T	X						
13121	한글 Windows	T	T	X						
13124	대만어	T	T	X						
28709	대만어(확장)	X	X	X	C	C	C	C	C	C

표 85. PC-자료 그룹 2(DBCS) 국가에 대한 SBCS 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
367	한글 및 중국어 EUC	X	C	C		X			C	
874	타이어(확장)	C	C	C	X	X			X	
891	한글(확장 없음)		C	C						
895	일본어 EUC - JISX201 Roman 세트	C	C							
896	일본어 EUC - JISX201 가타카나 세트		C							
897	일본어(확장 없음)	X	C	C						
903	중국어(확장 없음)		C	C						
904	대만어(확장 없음)	C	C	C						

표 85. PC-자료 그룹 2(DBCS) 국가에 대한 SBCS 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1040	한글(확장)		C	C						
1041	일본어(확장)	X	C	C						
1042	중국어(확장)		C	C						
1043	대만어(확장)	C	C	C						
1088	한글(KS 코드 5601-89)	X	C	C						
1114	대만어(Big-5)	C	C	C						
1115	중국어 GB-코드	C	C	C						
1126	한글 Windows		C	C						
1129	베트남어			C						
1133	라오스어 ISO			C						
1258	베트남어			C						
4970	타이어(확장)			C	X	X			X	
5210	대만어			C						
9066	타이어(확장)			C						

표 86. EBCDIC 그룹 2(DBCS) 국가에 대한 DBCS 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
300	일본어 - 4370 사용자 정의 문자 (UDC) 포함	X	X	X	C	C	C	C	C	C
834	한글 - 1880 사용자 정의 문자 (UDC) 포함	X	X	X	C	C	C	C	C	C
835	대만어 - 6204 사용자 정의 문자 (UDC) 포함	X	X	X	C	C	C	C	C	C
837	중국어 - 1880 사용자 정의 문자 (UDC) 포함	X	X	X	C	C	C	C	C	C
4396	일본어 - 1880 사용자 정의 문자 (UDC) 포함	X	X	X	C	C	C	C	C	C
4930	한글 Windows			X	C	C	C	C	C	C
4933	중국어			X	C	C	C	C	C	C

표 87. PC-자료 그룹 2(DBCS) 국가에 대한 DBCS 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
301	일본어 - 1880 사용자 정의 문자 (UDC) 포함	X	C	C	X	X	C	C	C	C
926	한글 - 1880 사용자 정의 문자 (UDC) 포함		C	C						
927	대만어 - 6204 사용자 정의 문자 (UDC) 포함	C	C	C	X	C	C	C	C	C
928	중국어 - 1880 사용자 정의 문자 (UDC) 포함		C	C						
941	일본어 Windows	X	C	C	C	C	C	C	X	C
947	대만어(Big-5)	C	C	C	X	X	C	C	X	C
951	한글(KS 코드 5601-89) - 1880 사용자 정의 문자 (UDC) 포함	X	C	C	X	C	C	C	X	C
952	일본어(EUC) X208-1990 세트		C							
953	일본어(EUC) X212-1990 세트		C							
971	한글(EUC) - 1880 사용자 정의 문자 (UDC) 포함	X	C	C	C	X	X	X	C	C
1351	일본어 HP-UX(J15)	X		C	C	C	X	C	C	C
1362	한글 Windows		C	C	C	C	C	C	X	C
1380	중국어(GB-코드) - 1880 사용자 정의 문자(UDC) 포함	C	C	C	X	C	C	C	X	X
1382	중국어(EUC) - 1360 사용자 정의 문자(UDC) 포함	C	C	C	C	X	X	X	C	X
1385	대만어			C	C	C	C	C	X	C

코드화 문자 세트 ID(CCSID) 값

표 88. EBCDIC 그룹 2(DBCS) 국가에 대한 혼합 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
930	일본어 가타카나/간지(확장) - 4370 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C
933	한글(확장) - 1880 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C
935	중국어(확장) - 1880 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C
937	대만어(확장) - 4370 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C
939	일본어 영어/간지(확장) - 4370 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C
1364	한글(확장)			X	C	C	C	C	C	C
1388	중국어			X	C	C	C	C	C	C
5026	일본어 가타카나/간지(확장)- 1880 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C
5035	일본어 영어/간지(확장) - 1880 사용자 정의 문자(UDC) 포함	X	X	X	C	C	C	C	C	C

표 89. PC-자료 그룹 2(DBCS) 국가에 대한 혼합 코드화 문자 세트 ID(CCSID)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
932	일본어(확장 없음) - 1880 사용자 정의 문자(UDC) 포함	X	C	C	X	X	C	C	C	C
934	한글(확장 없음) 1880 사용자 정의 문자(UDC) 포함		C	C						

표 89. PC-자료 그룹 2(DBCS) 국가에 대한 혼합 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
936	중국어(확장 없음) - 1880 사용자 정의 문자(UDC) 포함		C	C						
938	대만어(확장 없음) - 6204 사용자 정의 문자(UDC) 포함	C	C	C	X	C	C	C	C	C
942	일본어(확장) - 1880 사용자 정의 문자(UDC) 포함	X	C	C	X	C	C	C	C	C
943	일본어 NT	X	C	C	X	C	C	C	X	C
944	한글(확장) - 1880 사용자 정의 문자(UDC) 포함	C	C	C						
946	중국어(확장) - 1880 사용자 정의 문자(UDC) 포함		C	C						
948	대만어(확장) - 6204 사용자 정의 문자(UDC) 포함	C	C	C	X	C	C	C	C	C
949	한글(KS 코드 5601-89) - 1880 사용자 정의 문자(UDC) 포함	X	C	C	X	C	C	C	C	C
950	대만어(Big-5)	C	C	C	X	X	X	X	X	C
954	일본어(EUC)	C	C	C	C	X	X	X	C	X
956	일본어 2022 TCP			C						
957	일본어 2022 TCP			C						
958	일본어 2022 TCP			C						
959	일본어 2022 TCP			C						
964	대만어(EUC)	C	C	C	C	X	X	X	C	C
965	대만어 2022 TCP			C						
970	한글 EUC	X	C	C	C	X	X	X	C	C
1363	한글 Windows		C	C	C	C	C	C	X	C
1381	중국어 GB-코드	C	C	C	X	C	C	C	X	C
1383	중국어 EUC	C	C	C	C	X	X	X	C	X
1386	중국어			C	X	X	C	C	X	C
1392	중국어 GB18030			C						

표 89. PC-자료 그룹 2(DBCS) 국가에 대한 혼합 코드화 문자 세트 ID(CCSID) (계속)

CCSID	설명	DB2 UDB UWO (OS/390)	VM 및 VSE용 DB2	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/ 6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
5039	일본어 HP-UX(J15)	C			C	C	X	C	C	C
5050	일본어(EUC)			C						
5052	일본어 2022 TCP			C						
5053	일본어 2022 TCP			C						
5054	일본어 2022 TCP			C						
5055	일본어 2022 TCP			C						
17354	한글 2022 TCP			C						
25546	한글 2022 TCP			C						
33722	일본어 EUC			C						

부록 F. SQL문의 특성

이 부록에는 SQL문이 사용되는 여러 위치와 관련하여 SQL문의 특성에 대한 정보가 있습니다.

- SQL문에 허용되는 조치는 SQL문을 실행할 수 있으며, 대화식으로나 동적으로 준비할 수 있으며, 리퀘스터, 서버 또는 사전컴파일러로 처리할 수 있는지 여부를 보여줍니다. 『SQL문의 허용된 조치』를 참조하십시오.
- 자료 액세스 표시 표는 루틴의 SQL문을 사용하기 위해 지정해야 하는 SQL 자료 액세스 레벨을 보여줍니다. 915 페이지의 『루틴에서 SQL문 자료 액세스 표시』를 참조하십시오.
- 분산 관계형 데이터베이스 사용을 고려하면 어플리케이션 서버가 어플리케이션 리퀘스터와 같지 않을 때의 SQL문 사용에 대한 정보를 얻을 수 있습니다. 부록 F 『SQL문의 특성』을 참조하십시오.

SQL문의 허용된 조치

표 90에서는 특정 DB2문을 실행할 수 있으며, 대화식으로나 동적으로 준비할 수 있으며, 리퀘스터, 서버 또는 사전컴파일러로 처리할 수 있는지 여부를 보여줍니다. **Y** 글자는 *yes*를 의미합니다.

표 90. SQL문의 허용된 조치

		대화식으로 또는	처리		
SQL문	실행문	동적으로 준비	요구 시스템	서버	사전컴파일러
ALTER	**Y**	**Y**		**Y**	
BEGIN DECLARE SECTION					**Y**
CALL	**Y**	**Y**		**Y**	
CLOSE	**Y**			**Y**	
COMMENT	**Y**	**Y**		**Y**	
COMMIT	**Y**	**Y**		**Y**	
CONNECT(유형 1 및 유형 2)	**Y**		**Y**		
CREATE ...	**Y**	**Y**		**Y**	
DECLARE CURSOR					**Y**
DECLARE GLOBAL TEMPORARY TABLE	**Y**	**Y**		**Y**	
DECLARE PROCEDURE					**Y**
DECLARE STATEMENT					**Y**
DECLARE VARIABLE					**Y**
DELETE	**Y**	**Y**		**Y**	
DESCRIBE	**Y**			**Y**	

표 90. SQL문의 허용된 조치 (계속)

SQL문	실행문	대화식으로 또는 동적으로 준비	처리		
			요구 시스템	서버	사전컴파일러
DESCRIBE TABLE	Y			Y	
DISCONNECT	Y		Y		
DROP ...	Y	Y		Y	
END DECLARE SECTION					Y
EXECUTE	Y			Y	
EXECUTE IMMEDIATE	Y			Y	
FETCH	Y			Y	
FREE LOCATOR	Y	Y		Y	
GET DIAGNOSTICS[2]	Y			Y	
GRANT ...	Y	Y		Y	
HOLD LOCATOR	Y	Y		Y	
INCLUDE					Y
INSERT	Y	Y		Y	
LABEL	Y	Y		Y	
LOCK TABLE	Y	Y		Y	
OPEN	Y			Y	
PREPARE	Y			Y	
RELEASE CONNECTION	Y		Y		
RELEASE SAVEPOINT	Y	Y		Y	
RENAME	Y	Y		Y	
REVOKE ...	Y	Y		Y	
ROLLBACK	Y	Y		Y	
SAVEPOINT	Y	Y		Y	
SELECT INTO	Y			Y	
SET CONNECTION	Y		Y		
SET OPTION					Y
SET PATH	Y	Y		Y	
SET RESULT SETS[3]	Y			Y	
SET SCHEMA	Y	Y		Y	
SET TRANSACTION	Y	Y		Y	
SET 변수	Y		Y		
SIGNAL SQLSTATE[2]	Y			Y	
UPDATE	Y	Y		Y	
VALUES[1]	Y			Y	
VALUES INTO	Y	Y		Y	
WHENEVER					Y

표 90. SQL문의 허용된 조치 (계속)

SQL문	실행문	대화식으로 또는 동적으로 준비	처리		
			요구 시스템	서버	사전컴파일러

주:

1. 이 명령문은 트리거의 트리거된 조치에서만 사용될 수 있습니다.
2. 이 명령문은 SQL 함수, SQL 프로시듀어 또는 SQL 트리거에서만 사용할 수 있습니다. SQL trigger.
3. 이 명령문은 프로시듀어에서만 사용할 수 있습니다.

루틴에서 SQL문 자료 액세스 표시

다음 표는 SQL문(첫 열에 지정된)을 지정된 SQL 자료 액세스 표시로 함수나 프로시듀어에서 실행할 수 있는지 여부를 나타냅니다. NO SQL로 정의된 함수나 프로시듀어에 실행 가능 SQL문이 있는 경우, SQLSTATE 38001이 리턴됩니다. 다른 실행 문맥에서, 문맥에서 지원되지 않은 SQL문은 SQLSTATE 38003을 리턴합니다. CONTAINS SQL 문맥에서 허용되지 않는 다른 SQL문의 경우 SQLSTATE 38004가 리턴되고 READS SQL DATA 문맥에서 SQLSTATE 38002가 리턴됩니다. SQL 함수나 SQL 프로시듀어를 작성하는 동안, SQL 자료 액세스 표시에 맞지 않는 명령문이 있으면 SQLSTATE 42895가 리턴됩니다.

표 91. SQL문 및 SQL 자료 액세스 표시

SQL문	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER TABLE	N	N	N	Y
BEGIN DECLARE SECTION	Y[1]	Y	Y	Y
CALL	N	Y	Y	Y
CLOSE	N	N	Y	Y
COMMENT	N	N	N	Y
COMMIT	N	N	N	N
CONNECT(유형 1 및 유형 2)[3]	N	N	N	N
CREATE ...	N	N	N	Y
DECLARE CURSOR	Y[1]	Y	Y	Y
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	Y
DECLARE PROCEDURE	Y[1]	Y	Y	Y
DECLARE STATEMENT	Y[1]	Y	Y	Y
DECLARE VARIABLE	Y[1]	Y	Y	Y
DELETE	N	N	N	Y
DESCRIBE	N	N	Y	Y
DESCRIBE TABLE	N	N	Y	Y
DISCONNECT[3]	N	N	N	N
DROP ...	N	N	N	Y

표 91. SQL문 및 SQL 자료 액세스 표시 (계속)

SQL문	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
END DECLARE SECTION	Y[1]	Y	Y	Y
EXECUTE	N	Y[2]	Y[2]	Y
EXECUTE IMMEDIATE	N	Y[2]	Y[2]	Y
FETCH	N	N	Y	Y
FREE LOCATOR	N	Y	Y	Y
GRANT ...	N	N	N	Y
HOLD LOCATOR	N	Y	Y	Y
INCLUDE	Y[1]	Y	Y	Y
INSERT	N	N	N	Y
LABEL	N	N	N	Y
LOCK TABLE	N	Y	Y	Y
OPEN	N	N	Y	Y
PREPARE	N	Y	Y	Y
RELEASE CONNECTION[3]	N	N	N	N
RELEASE SAVEPOINT	N	N	N	Y
RENAME	N	N	N	Y
REVOKE ...	N	N	N	Y
ROLLBACK	N	Y	Y	Y
ROLLBACK TO SAVEPOINT	N	N	N	Y
SAVEPOINT	N	N	N	Y
SELECT INTO	N	N	Y	Y
SET CONNECTION[3]	N	N	N	N
SET OPTION	Y[1]	Y	Y	Y
SET PATH	N	N	Y	Y
SET RESULT SETS	N	Y	Y	Y
SET SCHEMA	N	N	Y	Y
SET TRANSACTION	N	Y	Y	Y
SET 변수	N	Y	Y	Y
UPDATE	N	N	N	Y
VALUES	N	Y	Y	Y
VALUES INTO	N	N	Y	Y
WHENEVER	Y[1]	Y	Y	Y

주:

1. NO SQL 옵션은 SQL문을 지정할 수 없음을 의미하지만, 실행 가능하지 않은 명령문은 제한되지 않습니다.

2. 실행되는 명령문에 의존합니다. EXECUTE문에 대해 지정된 명령문은 유효한 특정 SQL 액세스 레벨의 문맥에서 허용되는 명령문이어야 합니다. 예를 들어, 유효 SQL 액세스 레벨이 READS SQL DATA인 경우, 명령문은 INSERT, UPDATE 또는 DELETE여야 합니다.
3. 저장 프로시듀어 실행 문맥에서는 연결 관리 명령문이 허용되지 않습니다.

분산 관계형 데이터베이스(DRDB) 사용시 고려사항

이 표에서는 어플리케이션 리퀘스터와 똑같은 제품이 아닌 서버를 사용하는 어플리케이션 개발에 유용한 정보를 제공합니다.

모든 IBM 관계형 데이터베이스(RDB) 제품은 IBM SQL에 대한 부가 제품을 지원합니다. 이들 확장 중 일부는 제품에 특정적이며, 일부는 하나 이상의 제품에 의해 공유됩니다.

비록 어플리케이션이 몇몇의 명령문 및 절을 지원하지 않는 데이터베이스 관리자의 어플리케이션 리퀘스터를 통하여 실행될 수도 있으나 대부분의 경우 어플리케이션은 현재 서버의 데이터베이스 관리자가 지원하는 명령문 및 절을 사용할 수 있습니다. 이 일반적인 규칙에 대한 제약은 918 페이지의 표 92, 919 페이지의 표 93, 921 페이지의 표 94 및 923 페이지의 표 95에 명시되어 있습니다.

표 안에서 이 SQL 함수가 특정 환경에서 지원되지 않음을 표시하는 'R'에 주의하십시오. 같은 행의 모든 열 안의 'R'은 이 함수가 단지 현재 서버 및 리퀘스터가 똑같은 제품일때만 사용할 수 있음을 의미합니다.

다음 표에서 DB2 UDB UWO가 OS/390 및 z/OS용 DB2 UDB, VM 및 VSE용 DB2 또는 iSeries용 DB2 UDB 외에 어느 DB2 제품에 적용되는지 자세히 살표보십시오.

이 부록에 나오는 다음 제품 버전에 대한 것입니다.

- OS/390 및 z/OS용 DB2 UDB 버전 7
- VM 및 VSE용 DB2 버전 7.1
- iSeries용 DB2 UDB 버전 5 릴리스 2
- OS/2용 DB2 UDB UWO 버전 7
- AIX/6000용 DB2 UDB UWO 버전 7
- HP용 DB2 UDB UWO 버전 7
- NT용 DB2 UDB UWO 버전 7
- SUN용 DB2 UDB UWO 버전 7
- SCO Open Server용 DB2 UDB UWO 버전 7

표 92. OS/390 및 z/OS용 DB2 UDB 어플리케이션 리퀘스터

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버
결과 세트와 함께 사용되는 CALL				R
COMMIT HOLD	R	R	R	R
COMMIT RELEASE	R	R	R	R
CONNECT(유형 2)		주 1		주 1
DECLARE CURSOR WITH HOLD		R		
DECLARE STATEMENT				
DECLARE TABLE				
DECLARE VARIABLE				
지연 존재(주 3)				R
DESCRIBE TABLE		R		R
USING절과 함께 사용되는 DESCRIBE				R
DISCONNECT	R	R	R	R
확장(Extended) 동적 명령문	R	R	R	R
호스트 구조				
호스트 변수 - 선택적 콜론(:)		R	R	R
LOB(Large Object) 자료 유형		R		
DATALINK 자료 유형	R	R	R	R
고유한 자료 유형		R		R
ROWID 자료 유형		R		R
IBM SQL이 아닌 호스트 선언		주 2	주 2	주 2
INTO절과 함께 사용되는 PREPARE				
USING절과 함께 사용되는 PREPARE				R
PUT	R	R	R	R
RELEASE				
ROLLBACK HOLD	R	R	R	R
ROLLBACK RELEASE	R	R	R	R
SET CONNECTION				
SET CURRENT PACKAGESET				
SET 호스트 변수		R	R	R
SET TRANSACTION	R	R	R	R
화면이동 커서 명령문	R	R	R	R
UPDATE 커서 - FOR UPDATE OF절은 명시되지 않음				
STOP과 함께 사용되는 WHENEVER	R	R	R	R

주:

1 서버는 다른 모든 연결이 이미 읽기 전용이 아닌 한 읽기 전용 연산에 대해서만 허용될 것입니다. 서버가 VM 및 VSE용 DB2 또는 DB2 UDB UWO인 경우 이것은 TCP/IP 연결이 사용되는 경우의 유일한 제약입니다.

2 이 명령문은 어플리케이션 리퀘스터가 명령문을 이해하는 경우 지원됩니다.

3 오브젝트는 오브젝트가 자료 조작 명령문 안에서 참조되는 바인드 시간에는 없어도 됩니다.

표 93. VM 및 VSE용 DB2 어플리케이션 리퀘스터

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버
결과 세트와 함께 사용되는 CALL				R
COMMIT HOLD	R	R	R	R
COMMIT RELEASE				
CONNECT(유형 2)	R	R	R	R
DECLARE CURSOR WITH HOLD		R		
DECLARE STATEMENT	R	R	R	R
DECLARE TABLE	R	R	R	R
DECLARE VARIABLE	R	R	R	R
지연 존재(주 2)			R	
DESCRIBE TABLE	R	R	R	R
USING절과 함께 사용되는 DESCRIBE			R	
DISCONNECT	R	R	R	R
확장(Extended) 동적 명령문	R 주 3		R 주 3	R 주 3
호스트 변수 - 선택적 콜론(:)	R	R	R	R
호스트 구조				
LOB(Large Object) 자료 유형	R	R	R	R
DATALINK 자료 유형	R	R	R	R
고유한 자료 유형	R	R	R	R
ROWID 자료 유형	R	R	R	R
IBM SQL이 아닌 호스트 선언	주 1		주 1	주 1
INTO절과 함께 사용되는 PREPARE	R	R	R	R
USING절과 함께 사용되는 PREPARE	R	R	R	R
PUT				
RELEASE	R	R	R	R
ROLLBACK HOLD	R	R	R	R
ROLLBACK RELEASE				
SET CONNECTION	R	R	R	R
SET CURRENT PACKAGESET	R	R	R	R
SET 호스트 변수	R	R	R	R
SET TRANSACTION	R	R	R	R
화면이동 커서 명령문	R	R	R	R
UPDATE 커서 - FOR UPDATE OF절은 명시되지 않음				
STOP과 함께 사용되는 WHENEVER				

표 93. VM 및 VSE용 DB2 어플리케이션 리퀘스터 (계속)

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버

주:

1 이 명령문은 어플리케이션 리퀘스터가 명령문을 이해하는 경우 지원됩니다.

2 오브젝트는 오브젝트가 자료 조작 명령문 안에서 참조되는 바인드 시간에는 없어도 됩니다.

3 변경 불가능한 패키지와 관련된 대부분의 확장(Extended) 동적 명령문은 작동합니다. 자세한 내용은 VM 및 VSE용 DB2 제품 라이브러리를 참조하십시오.

표 94. iSeries용 DB2 UDB어플리케이션 리퀘스터

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버
결과 세트와 함께 사용되는 CALL	R	R		R
COMMIT HOLD	R	R		R
COMMIT RELEASE	R	R	R	R
CONNECT(유형 2)		주 1		주 1
DECLARE CURSOR WITH HOLD		R		
DECLARE PROCEDURE				
DECLARE STATEMENT				
DECLARE TABLE				
DECLARE VARIABLE				
지연 존재(주 3)				R
DESCRIBE TABLE		R		R
USING절과 함께 사용되는 DESCRIBE				R
DISCONNECT				
확장(Extended) 동적 명령문	R	R	R	R
호스트 변수 - 선택적 콜론(:)	R	R	R	R
호스트 구조				
LOB(Large Object) 자료 유형		R		R
DATALINK 자료 유형	R	R		R
고유한 자료 유형		R		R
ROWID 자료 유형		R		R
IBM SQL이 아닌 호스트 선언	주 2	주 2		주 2
INTO절과 함께 사용되는 PREPARE				
USING절과 함께 사용되는 PREPARE				R
PUT	R	R	R	R
RELEASE				
ROLLBACK HOLD	R	R		R
ROLLBACK RELEASE	R	R	R	R
SET CONNECTION				
SET CURRENT PACKAGESET	R	R	R	R
SET 호스트 변수	R	R	R	R
SET TRANSACTION	R	R		R
화면이동 커서 명령문	R	R		R
UPDATE 커서 - FOR UPDATE OF절은 명시되지 않음	R	R		
STOP과 함께 사용되는 WHENEVER	R	R	R	R

표 94. iSeries용 DB2 UDB어플리케이션 리퀘스터 (계속)

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버

주:

1 서버는 다른 모든 연결이 이미 읽기 전용이 아닌 한 읽기 전용 연산에 대해서만 허용될 것입니다. 서버가 VM 및 VSE용 DB2 또는 DB2 UDB UWO인 경우 이것은 TCP/IP 연결이 사용되는 경우의 유일한 제약입니다.

2 이 명령문은 어플리케이션 리퀘스터가 명령문을 이해하는 경우 지원됩니다.

3 오브젝트는 오브젝트가 자료 조작 명령문 안에서 참조되는 바인드 시간에는 없어도 됩니다.

표 95. DB2 UDB UWO 어플리케이션 리퀘스터

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버
결과 세트와 함께 사용되는 CALL				
COMMIT HOLD	R	R	R	R
COMMIT RELEASE	R	R	R	R
CONNECT(유형 2)		주 1		주 1
DECLARE CURSOR WITH HOLD		R		
DECLARE STATEMENT	R	R	R	R
DECLARE TABLE	R	R	R	R
DECLARE VARIABLE	R	R	R	R
지연 존재(주 3)				R
DESCRIBE TABLE	R	R	R	R
USING절과 함께 사용되는 DESCRIBE	R	R	R	R
DISCONNECT				
확장(Extended) 동적 명령문	R	R	R	R
호스트 변수 - 선택적 콜론(:)	R	R	R	R
호스트 구조	주 4	주 4	주 4	주 4
LOB(Large Object) 자료 유형		R		
DATALINK 자료 유형	R	R	R	R
고유한 자료 유형		R		
ROWID 자료 유형	R	R	R	R
IBM SQL이 아닌 호스트 선언	주 2	주 2	주 2	
INTO절과 함께 사용되는 PREPARE				
USING절과 함께 사용되는 PREPARE	R	R	R	R
PUT	R	R	R	R
RELEASE				
ROLLBACK HOLD	R	R	R	R
ROLLBACK RELEASE	R	R	R	R
SET CONNECTION				
SET CURRENT PACKAGESET				
SET 호스트 변수	R	R	R	R
SET TRANSACTION	R	R	R	R
화면이동 커서 명령문	R	R	R	R
UPDATE 커서 - FOR UPDATE OF절은 명시되지 않음	R	R		
STOP과 함께 사용되는 WHENEVER	R	R	R	R

표 95. DB2 UDB UWO 어플리케이션 리퀘스터 (계속)

SQL문 또는 함수	OS/390 및 z/OS용 DB2 UDB 서버	VM 및 VSE용 DB2 서버	iSeries용 DB2 UDB 서버	DB2 UDB UWO 서버

주:

1 서버는 다른 모든 연결이 이미 읽기 전용이 아닌 한 읽기 전용 연산에 대해서만 허용될 것입니다. 만일 서버가 VM 및 VSE용 DB2라면, 이 제약은 TCP/IP 연결이 사용되는 경우의 유일한 제약입니다.

2 이 명령문은 어플리케이션 리퀘스터가 명령문을 이해하는 경우 지원됩니다.

3 오브젝트는 오브젝트가 자료 조작 명령문 안에서 참조되는 바인드 시간에는 없어도 됩니다.

4 DB2 UDB UWO에서 호스트 구조는 COBOL 내에서만 지원됩니다.

CONNECT(유형 1) 및 CONNECT(유형 2) 차이점

CONNECT 명령문에는 다음 두 가지 유형이 있습니다. 두 유형은 똑같은 구문을 갖지만 그 의미는 다릅니다.

- CONNECT(유형 1)은 리모트 작업 단위에 사용됩니다. 30 페이지의 『리모트 작업 단위』를 참조하십시오.
- CONNECT(유형 2)는 분산 작업 단위에 사용됩니다. 427 페이지의 『CONNECT(유형 2)』를 참조하십시오.

다음 표는 CONNECT(유형 1) 및 CONNECT(유형 2) 규칙 사이의 차이점을 요약한 것입니다.

표 96. CONNECT(유형 1) 및 CONNECT(유형 2) 차이점

유형 1 규칙	유형 2 규칙
CONNECT 명령문은 활성 그룹이 연결 가능한 상태에 있을 때 실행만 될 수 있습니다. 동일한 작업 단위 내에서 하나 이상의 CONNECT 명령문이 실행될 수 없습니다.	연결 가능한 상태에 대한 규칙은 없습니다. 동일한 작업 단위 내에서 하나 이상의 CONNECT 명령문이 실행될 수 있습니다.
로컬 디렉토리에 서버명이 나열되지 않았기 때문에 CONNECT 명령문이 실패한 경우 활성 그룹의 연결 상태는 변경되지 않습니다. 활성 그룹이 연결 가능한 상태가 아니기 때문에 CONNECT 명령문이 실패한 경우 활성 그룹의 SQL 연결 상태는 변경되지 않습니다. CONNECT 명령문이 기타 다른 이유로 실패한 경우 활성 그룹은 연결이 해제된 상태가 됩니다.	CONNECT 명령문이 실패한 경우 현재 SQL 연결에는 변함이 없으며 다른 이후의 SQL 명령문이 현재 서버에 의해 실행됩니다.
CONNECT는 활성 그룹의 현존하는 연결 모두를 종료합니다. 따라서 CONNECT는 활성 그룹에 대해 열린 커서 모두를 닫습니다.	CONNECT는 연결을 종료하지 않으며 커서를 닫지 않습니다.
어플리케이션 그룹이 연결 상태인 경우 현재 서버에 대한 CONNECT는 성공할 것입니다.	활성 그룹의 현존 SQL 연결에 대한 CONNECT는 오류입니다. 따라서 현재 서버에 대한 CONNECT는 오류입니다.

적용할 CONNECT 규칙 판별

프로그램 준비 옵션은 하나의 프로그램에 의해 수행될 CONNECT 유형을 명시하는 데 사용됩니다. 프로그램 준비 옵션은 CRTSQLxxx 명령에서 RDBCNNMTH 매개변수를 사용하여 명시됩니다.

리모트 작업 단위만 지원하는 서버로의 연결

리모트 작업 단위만 지원하는 서버로의 CONNECT(유형 2) 연결은 읽기 전용 연결을 초래할 수 있습니다.

CONNECT(유형 2)가 리모트 작업 단위 지원 전용 서버로 수행된 경우[81]

- 이 연결은 연결 당시 갱신이 가능한 비활성 연결이 있는 경우 읽기 전용 연산만을 허용합니다. 이러한 경우 연결 상태에서 갱신은 불가능합니다.
- 이러한 경우가 아니라면, 연결 상태에서 갱신이 가능합니다.

CONNECT(유형 2)가 분산 작업 단위를 지원하는 서버로 수행된 경우

- 리모트 작업 단위 지원 전용 서버에 갱신을 허용하는 비활성 연결이 있을 경우 이 연결은 읽기 전용 연산만을 허용합니다. 이러한 경우 비활성 연결이 종료되자마자 연결 상태에서 갱신을 허용합니다.
- 이러한 경우가 아니라면, 연결 상태에서 갱신이 가능합니다.

81. 본래 TCP/IP에 대한 초기 DRDA 지원을 사용하는 iSeries용 DB2 UDB는 리모트 작업 단위만 지원하는 서버의 예입니다.

부록 G. iSeries용 DB2 UDB 카탈로그 뷰

이 섹션에서는 iSeries용 DB2 UDB 카탈로그 내에 들어 있는 뷰를 설명합니다. 각 관계형 데이터베이스 관리자는 데이터베이스 내의 자료에 대한 정보가 들어 있는 표의 세트를 유지보수합니다. 이 표들을 집합적으로 *카탈로그*라고 합니다. *카탈로그 표*에는 iSeries용 DB2 UDB에 의해 지원되는 표, 사용자 정의 함수, 고유한 유형, 매개 변수, 프로시듀어, 패키지, 뷰, 색인, 별명, 제약 조건, 트리거 및 언어에 대한 정보가 들어 있습니다. 카탈로그는 이 시스템으로 부터 액세스할 수 있는 모든 관계형 데이터베이스에 대한 정보가 들어 있습니다.

카탈로그 뷰의 세 가지 클래스가 있습니다.

- iSeries 카탈로그 표 및 뷰

  iSeries 카탈로그 표 및 뷰는 ANS 및 ISO 카탈로그 뷰를 따라 모델링되지만, ANS 및 ISO 카탈로그 뷰와는 같지 않습니다. 이 표와 뷰는 이전 iSeries용 DB2 UDB 릴리스와 호환됩니다.

  이 표 및 뷰는 스키마 QSYS와 QSYS2에서 존재합니다.

  카탈로그 표와 뷰에는 전체 관계형 데이터베이스 상의 모든 표, 매개변수, 프로시듀어, 함수, 고유한 유형, 패키지, 뷰, 색인, 트리거, 및 제한사항에 대한 정보가 들어 있습니다. SQL 스키마가 작성될 때, 또한 SQL 스키마에는 스키마 내의 표, 패키지, 뷰, 색인 및 제한사항에 대한 정보만을 수록한 이 뷰들의 세트가 들어갑니다 (SYSPARMS, SYSPROCS, SYSFUNCS, SYSROUTINES, SYSROUTINEDEP 및 SYSTYPES 제외).

- ODBC 및 JDBC 카탈로그 뷰

  ODBC 및 JDBC 카탈로그 뷰는 ODBC 및 JDBC 메타데이터 API 요청을 충족하도록 설계되었습니다. 예를 들면, SQLColumns입니다. 이 뷰는 OS/390 및 z/OS용 DB2 UDB 및 DB2 UDB UWO 버전 8의 뷰와 호환됩니다. 이 뷰는 ODBC 또는 JDBC가 해당 메타데이터 API를 개선하거나 수정하면 수정됩니다.

  이 뷰는 스키마 SYSIBM에 존재합니다.

- ANS 및 ISO 카탈로그 뷰

  ANS 및 ISO 카탈로그 뷰는 ANS 및 ISO SQL 표준(Information Schema 카탈로그 뷰)에 맞게 설계되었습니다. 이 뷰는 DB2 UDB UWO 버전 8의 뷰와 호환됩니다. 이 뷰는 ANS 및 ISO 표준이 개선되거나 수정되면 수정됩니다.

  이 뷰는 스키마 QSYS2 및 SYSIBM에 존재합니다.

  이 뷰에는 향후 표준 개선을 위해 예약된 여러 열이 있습니다.

주: 이들 뷰 중 일부는 뷰 정의의 일부로 특수 카탈로그 함수를 사용합니다. 이들 함수는 SYSIBM에 존재하지만, 어플리케이션에 직접 사용할 수는 없습니다. 함수는 특정 독립형 보조 기억장치 풀(IASP)에 대해 작성되고 향후 릴리스에서 변경될 수 있습니다.

주

카탈로그의 이름: 일반적으로, 카탈로그 표의 열에 저장된 모든 이름은 분리 문자 없이 사용되고 대소문자 구분됩니다. 예를 들어, 다음 테이블이 작성되었다고 가정합니다.

```
CREATE TABLE "colname"/"long_table_name"
                        ("long_column_name" CHAR(10),
                         INTCOL INTEGER)
```

다음 SELECT문은 SQL명과 시스템명 사이의 맵핑에 대한 정보를 리턴하기 위해 사용하고, 다음 SELECT문은 다음과 같이 사용됩니다.

```
SELECT TABLE_NAME, SYSTEM_TABLE_NAME, COLUMN_NAME, SYSTEM_COLUMN_NAME
  FROM QSYS2/SYSCOLUMNS
  WHERE TABLE_NAME = 'long_table_name'    AND
        TABLE_SCHEMA = 'colname'
```

다음 행을 리턴합니다.

TABLE_NAME	SYSTEM_TABLE_NAME	COLUMN_NAME	SYSTEM_COLUMN_NAME
long_table_name	″long0001″	long_column_name	LONG_00001
long_table_name	″long0001″	INTCOL	INTCOL

카탈로그의 이름: 일반적으로, 짧은 시스템 열 이름보다는 긴 SQL 열 이름을 사용해야 합니다. iSeries 카탈로그 표 및 뷰의 짧은 시스템 열 이름은 이전 릴리스 및 다른 DB2 UDB 제품과의 호환을 위해 명시적으로 관리됩니다. ODBC 및 JDBC 카탈로그 뷰, ANS 및 ISO 카탈로그 뷰의 짧은 시스템 열 이름은 명시적으로 관리되지 않으며 릴리스 간에 변경될 수 있습니다.

카탈로그에서 널값: 열의 정보가 적용 불가능한 경우 널값이 리턴됩니다. 위에 작성된 표를 사용하면 NUMERIC_SCALE 및 CHARACTER_MAXIMUM_LENGTH를 조회하는 다음의 선택 명령문은 열의 자료 유형에 자료의 적용이 불가능할 때 널값을 리턴합니다.

```
SELECT COLUMN_NAME, NUMERIC_SCALE, CHARACTER_MAXIMUM_LENGTH
  FROM QSYS2/SYSCOLUMNS
  WHERE TABLE_NAME = 'long_table_name'    AND
        TABLE_SCHEMA = 'colname'
```

다음 행을 리턴합니다.

COLUMN_NAME	NUMERIC_SCALE	CHARACTER_MAXIMUM_LENGTH
long_column_name	?	10
INTCOL	0	?

숫자 스케일이 문자 열에 대해 유효하지 않기 때문에 ″long_column_name″ 열에 대한 NUMERIC_SCALE에 대해서 널값이 리턴됩니다. 문자 길이가 숫자 열에 대해 유효하지 않기 때문에 INTCOL 열에 대한 CHARACTER_MAXIMUM_LENGTH에 대해서 널값이 리턴됩니다.

설치 및 백업 고려사항: 카탈로그 표 및 뷰에 대해 작성된 특정 카탈로그 표 및 모든 뷰는 정기적으로 저장해야 합니다.

- 카탈로그 표 QSYS.QADBXRDBD는 관계형 데이터베이스 정보가 들어 있습니다. 이 표를 정기적으로 저장해야 합니다.
- ILE 외부 함수나 프로시듀어 또는 SQL 함수나 프로시듀어가 복원되면, 해당 카탈로그 표에 정보가 자동으로 삽입됩니다. 비ILE 외부 함수 및 프로시듀어의 경우 이러한 상황이 발생하지 않습니다. 비ILE 외부 함수 또는 프로시듀어의 정의를 백업하려면, 카탈로그 표 SYSROUTINES 및 SYSPARMS가 저장되었거나 이러한 함수 및 프로시듀어를 작성하는 데 사용되는 SQL 소스문의 백업을 가지고 있는지 확인하십시오.
- QSYS2 또는 SYSIBM 스키마 내의 모든 카탈로그 뷰는 시스템 오브젝트입니다. 다시 말해 카탈로그 뷰를 통해 작성된 모든 사용자 뷰는 오퍼레이팅 시스템이 설치될 때 반드시 삭제되어야 합니다. 모든 의존적인 오브젝트 또한 삭제되어야 합니다. 이 요구사항을 피하기 위해서는 설치 이전에 뷰를 저장하고 그 후 다시 뷰를 복원할 수 있습니다.
- QSYS 라이브러리의 카탈로그 표는 또한 시스템 오브젝트입니다. QSYS 라이브러리 내의 카탈로그 표는 설치 중 삭제되지 않습니다. 따라서, 설치 프로세스 전체에 걸쳐 이 표에 대해 작성된 모든 뷰는 보존됩니다.

카탈로그 뷰로 권한 부여: 카탈로그 내의 표 및 뷰는 기타 다른 데이터베이스 표 및 뷰와 비슷합니다. 권한이 있다면 SQL문을 사용하여 다른 표에서 자료를 검색하는 것과 같은 방법으로 자료를 볼 수 있습니다. 카탈로그의 표 및 뷰는 SELECT 권한이 PUBLIC으로 설정된 채 제공됩니다. 이 권한을 취소하고 SELECT 권한을 개별 사용자에게 부여할 수 있습니다.

QSYS 카탈로그 표: 대부분의 카탈로그 뷰는 QSYS 라이브러리의 다음 표에 기초합니다(때로는 데이터베이스 상호 참조 파일이라 함). 이 표는 SELECT 특권을 가지고 PUBLIC에게 제공되지 않으며 직접 사용할 수 없습니다.

QADBCCST	QADBKFLD	QADBXSFLD
QADBFDEP	QADBPKG	QADBXTRIGB
QADBFCST	QADBXRDBD	QADBXTRIGC
QADBIFLD	QADBXREF	QADBXTRIGD

iSeries 카탈로그 표 및 뷰

iSeries 카탈로그에는 QSYS2 스키마 내의 다음 뷰 및 표가 있습니다.

iSeries용 DB2 UDB 이름	해당 ANSI/ISO 명	설명
932 페이지의『SYSCATALOGS』	CATALOGS	관계형 데이터베이스에 대한 정보
934 페이지의『SYSCHKCST』	CHECK_CONSTRAINTS	검사 제약 조건에 대한 정보
935 페이지의『SYSCOLUMNS』	COLUMNS	열 속성에 대한 정보
944 페이지의『SYSCST』	TABLE_CONSTRAINTS	모든 제약 조건에 대한 정보
945 페이지의『SYSCSTCOL』	CONSTRAINT_COLUMN_USAGE	제약 조건에서 참조된 열에 대한 정보
946 페이지의『SYSCSTDEP』	CONSTRAINT_TABLE_USAGE	표에서의 제약 조건 의존성에 대한 정보
947 페이지의『SYSFUNCS』	ROUTINES	사용자 정의 함수에 대한 정보
953 페이지의『SYSINDEXES』		색인에 대한 정보
954 페이지의『SYSJARCONTENTS』		Java 루틴의 jars에 대한 정보.
955 페이지의『SYSJAROBJECTS』		Java 루틴의 jars에 대한 정보.
956 페이지의『SYSKEYCST』	KEY_COLUMN_USAGE	고유, 1차 및 외부 키에 대한 정보
957 페이지의『SYSKEYS』		색인 키에 대한 정보
958 페이지의『SYSPACKAGE』		패키지에 대한 정보
960 페이지의『SYSPARMS』	PARAMETERS	루틴 매개변수에 대한 정보
964 페이지의『SYSPROCS』	ROUTINES	프로시듀어에 대한 정보
969 페이지의『SYSREFCST』	REFERENTIAL_CONSTRAINTS	참조 제약 조건에 대한 정보
971 페이지의『SYSROUTINES』	ROUTINES	함수 및 프로시듀어에 대한 정보
970 페이지의『SYSROUTINEDEP』	ROUTINE_TABLE_USAGE	기능 및 프로시듀어 종속성에 대한 정보
979 페이지의『SYSTABLES』	TABLES	표 및 뷰에 대한 정보
981 페이지의『SYSTRIGCOL』	TRIGGER_COLUMN_USAGE	트리거에서 사용된 열 정보
982 페이지의『SYSTRIGDEP』	TRIGGER_TABLE_USAGE	트리거에서 사용된 오브젝트 정보
983 페이지의『SYSTRIGGERS』	TRIGGERS	트리거 정보
987 페이지의『SYSTRIGUPD』	TRIGGERED_UPDATE_COLUMNS	트리거의 WHEN절의 열 정보
988 페이지의『SYSTYPES』	USER_DEFINED_TYPES	내장 자료 유형과 고유한 유형에 대한 정보
994 페이지의『SYSVIEWDEP』	VIEW_TABLE_USAGE	표에 대한 뷰 의존성에 대한 정보
996 페이지의『SYSVIEWS』	VIEWS	뷰의 정의에 대한 정보

SYSCATALOGS

SYSCATALOGS 뷰는 사용자가 연결할 수 있는 관계형 데이터베이스에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSCATALOGS 뷰의 열을 설명합니다.

표 97. SYSCATALOGS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CATALOG_NAME	LOCATION	VARCHAR(18)	관계형 데이터베이스명
CATALOG_STATUS	RDBASPSTAT	CHAR(10)	관계형 데이터베이스 상태 **ACTIVE** 관계형 데이터베이스는 활성화된 독립형 보조 기억장치 풀(IASP)과 연관되어 있지만 아직 사용할 수 없습니다. **AVAILABLE** 관계형 데이터베이스는 사용가능합니다. **VARYOFF** 관계형 데이터베이스는 단절변환된 독립 보조 기억장치 풀(IASP)과 연관되어 있습니다. **VARYON** 관계형 데이터베이스는 연결변환된 독립 보조 기억장치 풀(IASP)과 연관되어 있지만 아직 사용할 수 없습니다. **UNKNOWN** 관계형 데이터베이스 상태는 알 수 없습니다.리모트 관계형 데이터베이스 상태는 항상 알 수 없습니다.
CATALOG_TYPE	RDBTYPE	CHAR(7)	관계형 데이터베이스 유형 **LOCAL** 관계형 데이터베이스는 이 시스템에 로컬로 있습니다. **REMOTE** 관계형 데이터베이스는 리모트 시스템에 있습니다.
CATALOG_ASPGRP	RDBASPGRP	VARCHAR(10) 널값 가능	독립 보조 기억장치 풀(IASP) 이름 관계형 데이터베이스 상태가 UNKNOWN인 경우 널 값이 들어갑니다.

표 97. SYSCATALOGS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CATALOG_ASPNUM	RDBASPNUM	VARCHAR(10) 널값 가능	독립 보조 기억장치 풀(IASP) 수 관계형 데이터베이스 상태가 UNKNOWN인 경우 널 값이 들어갑니다.
CATALOG_TEXT	RDBTEXT	CHAR(50)	관계형 데이터베이스 텍스트 설명.

SYSCHKCST

SYSCHKCST 뷰에는 SQL 스키마의 각 검사 제한사항에 대하여 하나의 행이 들어 있습니다. 다음 표는 SYSCHKCST 뷰의 열을 설명합니다.

표 98. SYSCHKCST 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CONSTRAINT_SCHEMA	DBNAME	VARCHAR(128)	제한사항이 들어 있는 스키마명.
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	제약 조건 명
CHECK_CLAUSE	CHECK	VARCHAR(2000) 널값 가능	검사 제한사항 절의 텍스트 검사절이 절단 없이 표현할 수 없는 경우 널 값이 들어갑니다.

SYSCOLUMNS

SYSCOLUMNS 뷰에는 SQL 스키마의 각 표 및 뷰의 모든 열에 대한 하나의 행이 들어 있습니다(SQL 카탈로그의 열 포함). 다음 표에서는 SYSCOLUMNS 뷰의 열을 설명합니다.

표 99. SYSCOLUMNS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
COLUMN_NAME	NAME	VARCHAR(128)	열의 이름. 이름이 있으면 그 이름이 SQL 열 이름이 됩니다. 아니면 그 이름이 시스템 열 이름이 됩니다.
TABLE_NAME	TBNAME	VARCHAR(128)	열이 들어 있는 표 또는 뷰 이름. 이름이 하나이면 그 이름이 SQL 표 또는 뷰 이름이 됩니다. 아니면 그 이름이 시스템 표 또는 뷰 이름이 됩니다.
TABLE_OWNER	TBCREATOR	VARCHAR(128)	표 또는 뷰의 소유자
ORDINAL_POSITION	COLNO	INTEGER	좌에서 우로 순서가 지정된 표 또는 뷰의 열에 대한 숫자 위치

SYSCOLUMNS

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명	
DATA_TYPE	COLTYPE	VARCHAR(8)	열 유형	
			BIGINT	큰 수
			INTEGER	큰 수
			SMALLINT	작은 수
			DECIMAL	압축 십진수
			NUMERIC	존(zone) 십진수
			FLOAT	부동 소수점. FLOAT, REAL 또는 DOUBLE PRECISION
			CHAR	고정 길이 문자 스트링
			VARCHAR	가변 길이의 문자 스트링
			CLOB	문자 LOB(Large Object) 스트링
			GRAPHIC	고정 길이 그래픽 스트링
			VARG	가변 길이의 그래픽 스트링
			DBCLOB	2바이트 문자 LOB(Large Object) 스트링
			BLOB	2진 LOB(Large Object) 스트링
			DATE	날짜
			TIME	시간
			TIMESTMP	시간소인
			DATALINK	자료 링크
			ROWID	행 ID
			DISTINCT	고유한 유형

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
LENGTH	LENGTH	INTEGER	열의 길이 속성. 또는 십진수, 숫자 또는 0이 아닌 정밀도 2진 열의 경우 그 정밀도. **8바이트** BIGINT **4바이트** INTEGER **2바이트** SMALLINT **수 정밀도** DECIMAL **수 정밀도** NUMERIC **8바이트** FLOAT, FLOAT(n) 여기서 n은 25 - 53 또는 DOUBLE PRECISION **4바이트** FLOAT, FLOAT(n) 여기서 n은 1 - 24 사이 또는 REAL **스트링 길이** CHAR **스트링 최대 길이** VARCHAR 또는 CLOB **그래픽 스트링 길이** GRAPHIC **그래픽 스트링 최대 길이** VARGRAPHIC 또는 DBCLOB **2진 스트링의 최대 길이** BLOB **4바이트** DATE **3바이트** TIME **10바이트** TIMESTAMP **자료 링크 URL 및 주석의 최대 길이** DATALINK **40 바이트** ROWID **소스 유형과 동일한 값** DISTINCT
NUMERIC_SCALE	SCALE	INTEGER 널값 가능	숫자 자료의 스케일 열이 십진, 숫자 또는 2진이 아닌 경우 널값이 들어갑니다.

SYSCOLUMNS

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
IS_NULLABLE	NULLS	CHAR(1)	열에 널값이 들어가는 경우 **N** 아니오 **Y** 예
IS_UPDATABLE	UPDATES	CHAR(1)	열을 갱신할 수 있는 경우 **N** 아니오 **Y** 예
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
HAS_DEFAULT	DEFAULT	CHAR(1)	열에 디폴트 값(DEFAULT 절 또는 널(null) 허용)이 들어갈 경우 **N** 아니오 **Y** 예 **A** 열은 ROWID 자료 유형과 GENERATED ALWAYS 속성이 있습니다. **D** 열은 ROWID 자료 유형과 GENERATED BY DEFAULT 속성이 있습니다. **I** 열은 AS IDENTITY 및 GENERATED ALWAYS 속성으로 정의됩니다. **J** 열은 AS IDENTITY 및 GENERATED BY DEFAULT 속성으로 정의됩니다.
COLUMN_HEADING	LABEL	VARCHAR(60) 널값 가능	문자 스트링은 LABEL문으로 지원됩니다(열 머리말). 열 머리말이 없을 경우 널값이 들어갑니다.

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
STORAGE	STORAGE	INTEGER	열에 대한 기억장치 요구사항 **8바이트** BIGINT **4바이트** INTEGER **2바이트** SMALLINT **(정밀도/2) + 1** DECIMAL **수 정밀도** NUMERIC **8바이트** FLOAT, FLOAT(n) 여기서 n은 25 - 53 또는 DOUBLE PRECISION **4바이트** FLOAT, FLOAT(n) 여기서 n은 1 - 24 사이 또는 REAL **스트링 길이** CHAR **스트링 최대 길이 + 2** VARCHAR **스트링 최대 길이 + 29** CLOB **스트링 길이 * 2** GRAPHIC **스트링 최대 길이 * 2 + 2** VARGRAPHIC **스트링 최대 길이 * 2 + 29** DBCLOB **4바이트** DATE **3바이트** TIME **10바이트** TIMESTAMP **자료 링크 URL 및 주석의 최대 길이 + 24** DATALINK **42바이트** ROWID **소스 유형과 동일한 값** DISTINCT **주:** 이 열은 모든 자료 유형에 대한 기억장치 요구사항을 지원합니다.

SYSCOLUMNS

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
NUMERIC_PRECISION	PRECISION	INTEGER 널값 가능	전체 숫자 열 정밀도 **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 열이 숫자가 아닌 경우 널값이 들어갑니다.
CCSID	CCSID	INTEGER 널값 가능	CHAR, VARCHAR, CLOB, DATE, TIME, TIMESTAMP, GRAPHIC, VARGRAPHIC, DBCLOB 및 DATALINK 열에 대한 코드화 문자 세트 ID(CCSID) 값 열이 BLOB 또는 ROWID일 경우 65535가 들어갑니다. 열이 숫자 자료 유형일 경우 널값이 들어갑니다.
TABLE_SCHEMA	DBNAME	VARCHAR(128)	표 또는 뷰가 들어 있는 SQL 스키마명
COLUMN_DEFAULT	DFTVALUE	VARCHAR(2000) 널값 가능	만일 하나라도 존재할 경우 열의 디폴트 값. 만일 열의 디폴트 값을 절단시켜야만 볼 수 있으면 열의 값은 'TRUNCATED' 스트링이 됩니다. 디폴트 값은 문자 양식으로 저장됩니다. 다음과 같은 특수 값도 있습니다. **CURRENT_DATE** 디폴트 값은 현재 날짜입니다. **CURRENT_TIME** 디폴트 값은 현재 시간입니다. **CURRENT_TIMESTAMP** 디폴트 값은 현재 시간소인입니다. **NULL** 디폴트 값은 널값입니다. **USER** 디폴트 값은 현재의 작업 사용자입니다. 열에 디폴트 값이 없을 경우 널값이 들어갑니다. 예를 들어, 열이 IDENTITY 속성을 가지거나 행 ID인 경우입니다.
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 스트링 최대 길이 열이 스트링이 아닌 경우 널값이 들어갑니다.

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 바이트 수. 열이 스트링이 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진수 또는 십진수의 수로 명시되었는지 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 열이 숫자가 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	DATPRC	INTEGER 널값 가능	날짜, 시간 또는 시간소인의 분수 부분. **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 열이 날짜, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.
COLUMN_TEXT	LABELTEXT	VARCHAR(50) 널값 가능	문자 스트링은 LABEL문으로 지원됩니다(열 텍스트). 열에 열 텍스트가 없는 경우 널값이 들어갑니다.
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	열의 시스템명
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	표 또는 뷰의 시스템명
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	스키마의 시스템명
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) 널값 가능	이것이 고유한 유형인 경우 스키마명. 열이 고유한 유형이 아닌 경우 널값이 들어갑니다.
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128) 널값 가능	고유한 유형 이름. 열이 고유한 유형이 아닌 경우 널값이 들어갑니다.
IS_IDENTITY	IDENTITY	VARCHAR(3)	이 열은 열이 ID 열인지 여부를 식별합니다. **NO** 열은 ID 열이 아닙니다. **YES** 열은 ID 열입니다.

SYSCOLUMNS

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
IDENTITY_GENERATION	GENERATED	VARCHAR(10) 널값 가능	이 열은 GENERATED ALWAYS 또는 GENERATED BY DEFAULT인지 여부를 실별합니다. **ALWAYS** 열 값은 항상 생성됩니다. **BY DEFAULT** 열 값은 디폴트로 생성됩니다. 열이 ROWID 또는 IDENTITY 열이 아닌 경우 널값이 들어갑니다.
IDENTITY_START	START	DECIMAL(31,0) 널값 가능	ID 열의 시작 값. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.
IDENTITY_INCREMENT	INCREMENT	DECIMAL(31,0) 널값 가능	ID 열의 증가 값. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.
IDENTITY_MINIMUM	MINVALUE	DECIMAL(31,0) 널값 가능	ID 열의 최소 값. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.
IDENTITY_MAXIMUM	MAXVALUE	DECIMAL(31,0) 널값 가능	ID 열의 최대 값. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.
IDENTITY_CYCLE	CYCLE	VARCHAR(3) 널값 가능	이 열은 최소값 또는 최대값에 도달한 후 ID 열 값이 계속 생성되는지 여부를 나타냅니다. **NO** 값은 계속 생성되지 않습니다. **YES** 값은 계속 생성됩니다. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.
IDENTITY_CACHE	CACHE	INTEGER 널값 가능	보다 빠른 액세스를 위해 사전 할당되는 ID 값의 수를 지정합니다. 0은 값이 사전 할당되지 않음을 나타냅니다. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.

표 99. SYSCOLUMNS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
IDENTITY_ORDER	ORDER	VARCHAR(3) 널값 가능	요청 순서대로 ID 값이 생성되어야 하는지 여부를 지정합니다. **NO** 요청 순서대로 값이 생성될 필요가 없습니다. **YES** 요청 순서대로 값이 생성되어야 합니다. 열이 IDENTITY 열이 아닌 경우 널값이 들어갑니다.

SYSCST

SYSCST 뷰에는 SQL 스키마의 각 제한사항에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSCST 뷰의 열을 설명합니다.

표 100. SYSCST 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	제한사항이 들어 있는 스키마명.
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	제약 조건 명
CONSTRAINT_TYPE	TYPE	VARCHAR(11)	제약 조건 유형 CHECK UNIQUE PRIMARY KEY FOREIGN KEY
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	표가 들어 있는 스키마명.
TABLE_NAME	TBNAME	VARCHAR(128)	제약 조건 작성이 완료된 표 이름. 이름이 있으면 그 표 이름이 SQL 표 이름이 됩니다. 아니면 그 이름이 시스템 표 이름이 됩니다.
IS_DEFERRABLE	ISDEFER	VARCHAR(3)	제한사항 검사가 지연될 수 있는지 여부를 표시합니다. 항상 'NO'입니다.
INITIALLY_DEFERRED	INITDEFER	VARCHAR(3)	제한사항이 초기에 지연되었는지 여부를 표시합니다. 항상 'NO'입니다.
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	표의 시스템명
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	표가 들어 있는 스키마의 시스템명
CONSTRAINT_KEYS	COLCOUNT	SMALLINT 널값 가능	이 값이 UNIQUE, PRIMARY KEY 또는 FOREIGN KEY 제한조건인 경우 키 열 수를 지정합니다. 제한사항이 CHECK 제한사항인 경우 널값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSCSTCOL

SYSCSTCOL 뷰는 제약 조건이 정의된 열을 작성합니다. 고유 또는 1차 키 제약 조건 내의 각 열 및 참조 제약 조건의 참조 열에 대해 하나의 행이 존재합니다. 다음 표에서는 SYSCSTCOL 뷰의 열을 설명합니다.

표 101. SYSCSTCOL 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	제한사항이 종속된 표가 들어 있는 SQL 스키마명
TABLE_NAME	TBNAME	VARCHAR(128)	제약 조건이 의존하는 표 이름. 이름이 있으면 그 이름이 SQL 표 이름이 됩니다. 아니면 그 이름이 시스템 표 이름이 됩니다.
COLUMN_NAME	COLUMN	VARCHAR(128)	제약 조건이 작성된 열. 만일 존재하면 그것이 SQL 열 이름이 됩니다. 그렇지 않다면 시스템 열 이름이 될 것입니다.
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	제한사항이 들어 있는 스키마명
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	제약 조건 명
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	열의 시스템명
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	표의 시스템명
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	표가 들어 있는 스키마의 시스템명

SYSCSTDEP

SYSCSTDEP 뷰는 제약 조건이 정의된 표를 작성합니다. 다음 표에서는 SYSCSTDEP 뷰의 열을 설명합니다.

표 102. SYSCSTDEP 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	제한사항이 종속된 표가 들어 있는 SQL 스키마명
TABLE_NAME	TBNAME	VARCHAR(128)	제약 조건이 의존하는 표 이름. 만일 존재하면 그것이 SQL 표 이름이 됩니다. 그렇지 않다면 시스템 표 이름이 될 것입니다.
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	제한사항이 들어 있는 스키마명
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	제약 조건 명
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	표의 시스템명
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	표가 들어 있는 스키마의 시스템명
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSFUNCS

SYSFUNCS 뷰에는 CREATE FUNCTION 명령문에 의해 작성된 각 함수에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSFUNCS 뷰의 열을 설명합니다.

표 103. SYSFUNCS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	루틴(함수) 인스턴스의 스키마(schema)명
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	루틴 인스턴스의 고유 이름
ROUTINE_SCHEMA	FUNCSCHEMA	VARCHAR(128)	루틴이 들어 있는 SQL 스키마(schema)명
ROUTINE_NAME	FUNCNAME	VARCHAR(128)	루틴 명
ROUTINE_CREATED	FUNCCREATE	TIMESTAMP	루틴이 작성된 시간소인을 식별합니다.
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	루틴을 정의한 사용자 명
ROUTINE_BODY	BODY	VARCHAR(8)	루틴 본문의 유형 **EXTERNAL** 외부 루틴입니다. **SQL** SQL 루틴입니다.
EXTERNAL_NAME	EXTNAME	VARCHAR(279) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. • ILE 서비스 프로그램의 경우 외부 프로그램명은 *schema-name/service-program-name(entry-point-name)*입니다. • Java 프로그램의 경우 외부 프로그램명은 선택적인 jar-id 다음에 *fully-qualified-class-name!method-name* 또는 *fully-qualified-class-name.method-name*이 나옵니다. • 다른 모든 언어의 경우 외부 프로그램명은 *schema-name/program-name*입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

SYSFUNCS

표 103. SYSFUNCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. **C** 외부 프로그램이 C로 작성됩니다. **C++** 외부 프로그램이 C++로 작성됩니다. **CL** 외부 프로그램이 CL로 작성됩니다. **COBOL** 외부 프로그램이 COBOL로 작성됩니다. **COBOLLE** 외부 프로그램은 ILE COBOL로 작성됩니다. **JAVA** 외부 프로그램이 JAVA로 작성됩니다. **PLI** 외부 프로그램이 PL/I으로 작성됩니다. **RPG** 외부 프로그램이 RPG로 작성됩니다. **RPGLE** 외부 프로그램은 ILE RPG로 작성됩니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

표 103. SYSFUNCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) 널값 가능	외부 루틴인 경우 이 열은 매개변수 유형을 식별합니다(호출 규칙). **DB2SQL** DB2SQL 호출 규칙입니다. **DB2GNRL** DB2GENERAL 호출 규칙입니다. **GENERAL** GENERAL 호출 규칙입니다. **JAVA** JAVA 호출 규칙입니다. **NULLS** GENERAL WITH NULLS 호출 규칙입니다. **SQL** SQL 표준 호출 규칙입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	이 열은 루틴이 결정적인지 여부를 식별합니다. 다시 말해, 동일한 인수로 루틴을 호출하는 경우 항상 동일한 결과가 리턴되는지의 여부를 식별하는 것입니다. **NO** 루틴이 결정적이 아닙니다. **YES** 루틴이 결정적입니다.
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	이 열은 SQL이 들어 있는 루틴인지의 여부 및 자료를 읽고 변경하는지 여부를 식별합니다. **NONE** 루틴에 SQL문이 없습니다. **CONTAINS** 루틴에 SQL문이 있습니다. **READS** 루틴이 표 또는 뷰로부터 자료를 읽을 수 있습니다. **MODIFIES** 루틴이 표 또는 뷰 내의 자료를 변경하거나 SQL DDL(data definition language) 명령문을 생성할 수 있습니다.

SYSFUNCS

표 103. SYSFUNCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SQL_PATH	SQL_PATH	VARCHAR(3483) 널값 가능	SQL 루틴인 경우 이 열은 경로를 식별합니다. 이것이 외부 루틴인 경우 널 값이 들어갑니다.
PARM_SIGNATURE	SIGNATURE	VARCHAR(510)	이 열은 루틴 서명을 식별합니다.
NUMBER_OF_RESULTS	NUMRESULTS	SMALLINT	결과의 수를 식별합니다.
IN_PARMS	IN_PARMS	SMALLINT	입력 매개변수의 수를 식별합니다. 0은 입력 매개변수가 없음을 의미합니다.
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
ROUTINE_DEFINITION	ROUTINEDEF	VARCHAR(24000) 널값 가능	SQL 루틴인 경우 이 열에는 SQL 루틴 본문이 들어갑니다. 이것이 SQL 루틴이 아니거나 루틴 본문이 절단 없이 이 열에 들어있을 수 없는 경우에 널 값이 들어갑니다.
FUNCTION_ORIGIN	ORIGIN	CHAR(1)	함수 유형을 식별합니다. 만일 프로시듀어이면 이 열은 공백으로 남습니다. **B** 내장 함수(built-in function)입니다 (iSeries용 DB2 UDB에서 정의되었습니다). **E** 사용자 정의 함수입니다. **U** 다른 함수에 기반한 사용자 정의 함수입니다. **S** 시스템 생성 함수입니다.
FUNCTION_TYPE	TYPE	CHAR(1)	함수 형식을 식별합니다. 만일 프로시듀어이면 이 열은 공백으로 남습니다. **S** 스칼라 함수입니다. **C** 열 함수(column function)입니다. **T** 표 함수입니다.
EXTERNAL_ACTION	EXTACTION	CHAR(1) 널값 가능	함수 호출로 인한 외부 영향이 있을지 여부를 식별합니다. **E** 외부 부작용이 있습니다. **N** 외부 부작용이 없습니다.

표 103. SYSFUNCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
IS_NULL_CALL	NULL_CALL	VARCHAR(3) 널값 가능	입력 매개변수가 널값인 경우 함수를 호출할 필요가 있는지의 여부를 식별합니다. **NO** 입력 매개변수가 널값인 경우 함수를 호출할 필요가 없습니다. 이것이 스칼라 함수라면, 만일 피연산자 중 하나가 널값인 경우 함수의 결과는 내재적으로 널값이 됩니다. 이것이 표 함수라면, 만일 피연산자 중 하나가 널값인 경우 함수의 결과는 빈 표가 됩니다. **YES** 입력 연산자가 널(null)인 경우라도 이 함수를 반드시 호출하여야 합니다.
SCRATCH_PAD	SCRATCHPAD	INTEGER 널값 가능	정적 메모리 영역(스크래치 패드)의 주소가 함수에 전달되는지의 여부를 식별합니다. **0** 이 함수는 스크래치 패드가 없습니다. **정수** 함수에 전달되는 스크래치 패드의 크기를 표시합니다.
FINAL_CALL	FINAL_CALL	VARCHAR(3) 널값 가능	함수의 최종 호출이 함수 작업 영역(스크래치 패드) 제거를 허용하는지의 여부를 표시합니다. **NO** 최종 호출이 작성되지 않습니다. **YES** 명령문이 완료되면 함수에 대한 최종 호출이 작성됩니다.
PARALLELIZABLE	PARALLEL	VARCHAR(3) 널값 가능	함수의 병행 실행이 가능한지 여부를 식별합니다. **NO** 함수의 병행 실행이 불가능합니다. **YES** 함수의 병행 실행이 가능합니다.
DBINFO	DBINFO	VARCHAR(3) 널값 가능	데이터베이스에 관한 정보가 함수에 전달되는지의 여부를 식별합니다. **NO** 데이터베이스 정보가 함수에 전달되지 않습니다. **YES** 데이터베이스에 관한 정보가 함수에 전달됩니다.

SYSFUNCS

표 103. SYSFUNCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SOURCE_ SPECIFIC_SCHEMA	SRCSCHEMA	VARCHAR(128) 널값 가능	이것이 피소스(sourced) 함수이며 소스가 사용자 정의된 것이면 이 열에는 소스 스키마의 이름이 들어갑니다. 이것이 피소스(sourced) 함수이며 소스가 내장된 것이면 이 열에는 'QSYS2'가 들어갑니다. 피소스(sourced) 함수가 아닌 경우 널값이 들어갑니다.
SOURCE_SPECIFIC_NAME	SRCNAME	VARCHAR(128) 널값 가능	이것이 피소스(sourced) 함수이며 소스가 사용자 정의된 것이면 이 열에는 소스 함수명의 고유 이름이 들어갑니다. 피소스(sourced) 함수가 아닌 경우 널값이 들어갑니다.
IS_USER_DEFINED_CAST	CAST_FUNC	VARCHAR(3) 널값 가능	함수가 고유한 유형이 작성될 때 작성된 캐스트 함수인지의 여부를 식별합니다. **NO** 캐스트 함수가 아닙니다. **YES** 캐스트 함수입니다.
CARDINALITY	CARD	BIGINT 널값 가능	표 함수 중요성을 지정합니다. 표 함수가 아니고 중요성이 지정되지 않은 경우 널값을 포함합니다.
FENCED	FENCED	VARCHAR(3) 널값 가능	함수가 분리되었는지 여부를 식별합니다. **NO** 함수가 분리되지 않습니다. **YES** 함수가 분리됩니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSINDEXES

SYSINDEXES 뷰에는 SQL 카탈로그 상의 색인을 포함하여 SQL CREATE INDEX 명령문을 사용하여 작성된 SQL 스키마 내의 모든 색인에 대해 하나의 행이 들어 있습니다. 다음 표에서는 SYSINDEXES 뷰의 열을 설명합니다.

표 104. SYSINDEXES 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
INDEX_NAME	NAME	VARCHAR(128)	색인명. 만일 하나가 있으면, 그것이 SQL 색인명이 됩니다. 그렇지 않다면 시스템 색인명이 될 것입니다.
INDEX_OWNER	CREATOR	VARCHAR(128)	색인 소유자
TABLE_NAME	TBNAME	VARCHAR(128)	색인이 정의된 표 이름 이름이 있으면 그 이름이 SQL 표 이름이 됩니다. 아니면 그 이름이 시스템 표 이름이 됩니다.
TABLE_OWNER	TBCREATOR	VARCHAR(128)	표 소유자
TABLE_SCHEMA	TBDBNAME	VARCHAR(128)	색인이 정의된 표가 들어 있는 SQL 스키마명
IS_UNIQUE	UNIQUERULE	CHAR(1)	색인이 고유한 경우 **D** No(중복을 허가함) **V** Yes(중복 널(duplicate null)값 허용) **U** 예 **E** 코드화 벡터 색인
COLUMN_COUNT	COLCOUNT	INTEGER	키 안의 열 수
INDEX_SCHEMA	DBNAME	VARCHAR(128)	색인이 들어 있는 SQL 스키마명
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	시스템 색인명
SYSTEM_INDEX_SCHEMA	SYS_IDNAME	CHAR(10)	시스템 색인 스키마명
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	시스템 표 이름
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	시스템 표 스키마명
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSJARCONTENTS

SYSJARCONTENTS 표에는 SQL 스키마의 jarid에 의해 정의된 각 클래스에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSJARCONTENTS 뷰의 열을 설명합니다.

표 105. SYSJARCONTENTS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
JARSCHEMA	JARSCHEMA	VARCHAR(128)	jar_id가 들어 있는 스키마명
JAR_ID	JAR_ID	VARCHAR(128)	jar_id의 이름
CLASS	CLASS	VARCHAR(128)	클래스명
CLASS_SOURCE	CLASSSRC	DBCLOB(10485760) 널값 가능	예약. 널 값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSJAROBJECTS

SYSJAROBJECTS 표에는 SQL 스키마의 각 jarid에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSJAROBJECTS 뷰의 열을 설명합니다.

표 106. SYSJAROBJECTS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
JARSCHEMA	JARSCHEMA	VARCHAR(128)	jar_id가 들어 있는 스키마명
JAR_ID	JAR_ID	VARCHAR(128)	jar_id의 이름
DEFINER	DEFINER	VARCHAR(128)	jarid 소유자명
JAR_DATA	JAR_DATA	BLOB(104857600) 널값 가능	jar에 대한 바이트 코드
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.
JAR_CREATED	CREATEDTS	TIMESTAMP	Jar 작성 시간소인
LAST_ALTERED	ALTEREDTS	TIMESTAMP 널값 가능	예약. 널 값이 들어갑니다.
DEBUG_MODE	DEBUG_MODE	CHAR(1)	함수가 디버그 가능한지 여부를 지정합니다. **0** 함수가 디버그 가능하지 않습니다. **2** 함수가 디버그 가능합니다.
DEBUG_DATA	DEBUG_DATA	CLOB(1048576) 널값 가능	예약. 널 값이 들어갑니다.

SYSKEYCST

SYSKEYCST 뷰에는 SQL 스키마의 각 UNIQUE KEY, PRIMARY KEY 또는 FOREIGN KEY에 대한 하나 이상의 행이 들어 있습니다. 고유하거나 1차인 키의 제약 조건 내의 각 열 및 참조 제약 조건의 참조 열에 대해 하나의 행이 존재합니다. 다음 표에서는 SYSKEYCST 뷰의 열을 설명합니다.

표 107. SYSKEYCST 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	제한사항이 들어 있는 스키마명
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	제약 조건 명
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	표가 들어 있는 스키마명
TABLE_NAME	TBNAME	VARCHAR(128)	표 이름
COLUMN_NAME	COLNAME	VARCHAR(128)	열의 이름
ORDINAL_POSITION	COLSEQ	INTEGER	키 내의 열 위치
COLUMN_POSITION	COLNO	INTEGER	행 내의 열 위치
TABLE_OWNER	CREATOR	VARCHAR(128)	표 소유자
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	열의 시스템명
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	표의 시스템명
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	스키마 표가 들어 있는 스키마의 시스템명

SYSKEYS

SYSKEYS 뷰에는 SQL 카탈로그 상의 색인에 대한 키를 포함하여 SQL 스키마의 색인 하나에 대한 모든 열에 대해 하나의 행이 들어 있습니다. 다음 표에서는 SYSKEYS 뷰의 열을 설명합니다.

표 108. SYSKEYS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
INDEX_NAME	IXNAME	VARCHAR(128)	색인명. 만일 하나가 있으면, 그것이 SQL 색인명이 됩니다. 그렇지 않다면 시스템 색인명이 될 것입니다.
INDEX_OWNER	IXCREATOR	VARCHAR(128)	색인 소유자
COLUMN_NAME	COLNAME	VARCHAR(128)	키의 열 이름. 이름이 있으면 그 이름이 SQL 열 이름이 됩니다. 아니면 그 이름이 시스템 열 이름이 됩니다.
COLUMN_POSITION	COLNO	INTEGER	행 내의 열에 대한 숫자 위치
ORDINAL_POSITION	COLSEQ	INTEGER	키 내의 열에 대한 숫자 위치
ORDERING	ORDERING	CHAR(1)	키 내의 열 순서 **A** 오름차순 **D** 내림차순
INDEX_SCHEMA	IXDBNAME	VARCHAR(128)	색인이 들어 있는 스키마명
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	열의 시스템명
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	색인의 시스템명
SYSTEM_INDEX_SCHEMA	SYS_IDNAME	CHAR(10)	색인이 들어 있는 스키마의 시스템명

SYSPACKAGE

SYSPACKAGE 뷰에는 SQL 스키마의 각 SQL 패키지에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSPACKAGE 뷰의 열을 설명합니다.

표 109. SYSPACKAGE 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
PACKAGE_CATALOG	LOCATION	VARCHAR(128)	SQL 패키지의 관계형 데이터베이스명 (RDBNAME)
PACKAGE_SCHEMA	COLLID	VARCHAR(128)	스키마명
PACKAGE_NAME	NAME	VARCHAR(128)	SQL 패키지 명
PACKAGE_OWNER	OWNER	VARCHAR(128)	SQL 패키지 소유자
PACKAGE_CREATOR	CREATOR	VARCHAR(128)	SQL 패키지 작성자
CREATION_TIMESTAMP	TIMESTAMP	CHAR(26)	SQL 패키지가 작성된 때의 시간소인
DEFAULT_SCHEMA	QUALIFIER	VARCHAR(128)	비규정 표, 뷰 및 색인에 대한 내재명
PROGRAM_NAME	PROGNAME	VARCHAR(128)	패키지가 작성된 프로그램명
PROGRAM_SCHEMA	LIBRARY	VARCHAR(128)	프로그램이 들어 있는 스키마명
PROGRAM_CATALOG	RDB	VARCHAR(128)	프로그램이 상주하는 관계형 데이터베이스 (RDB)명
ISOLATION	ISOLATION	CHAR(2)	분리 옵션 스펙 RR 반복 읽기 RS 읽기 안정성(*ALL) CS 커서 안정성(*CS) UR 미확정 읽기(*CHG) NO 없음(*NONE)
QUOTE	QUOTE	CHAR(1)	이탈 문자(escape character) 스펙(Y/N): Y = 인용 부호 마크 N = 어포스트로피
COMMA	COMMA	CHAR(1)	쉼표 옵션 스펙(Y/N): Y = 쉼표 N = 마침표
PACKAGE_TEXT	LABEL	VARCHAR(50)	LABEL문으로 지원하는 문자 스트링
LONG_COMMENT	REMARKS	VARCHAR(2000)	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
CONSISTENCY_TOKEN	CONTOKEN	CHAR(8) FOR BIT DATA	패키지에 대한 일관성 토큰
SYSTEM_PACKAGE_NAME	SYS_NAME	CHAR(10)	패키지의 시스템명
SYSTEM_PACKAGE_SCHEMA	SYS_DNAME	CHAR(10)	패키지가 들어 있는 스키마의 시스템명
SYSTEM_DEFAULT_SCHEMA	SYS_DDNAME	CHAR(10)	비규정 표, 뷰, 색인 및 패키지에 대한 내재 규정자의 시스템명
SYSTEM_PROGRAM_NAME	SYS_PNAME	CHAR(10)	프로그램의 시스템명
SYSTEM_PROGRAM_SCHEMA	SYS_PDNAME	CHAR(10)	프로그램이 들어 있는 스키마의 시스템명

표 109. SYSPACKAGE 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSPARMS

SYSPARMS 표에는 CREATE PROCEDURE 명령문에 의해 작성된 프로시듀어 또는 CREATE FUNCTION 명령문에 의해 작성된 함수의 각 매개변수에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSPARMS 표 내의 열을 설명합니다.

표 110. SYSPARMS 표

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	루틴 인스턴스의 스키마(schema) 이름
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	루틴 인스턴스의 고유 이름
ORDINAL_POSITION	PARMNO	INTEGER	좌에서 우로 순서를 매기는 매개변수 리스트 내의 매개변수의 숫자 위치
PARAMETER_MODE	PARMMODE	VARCHAR(5)	매개변수 유형: **IN** 입력 매개변수입니다. **OUT** 출력 매개변수입니다. **INOUT** 입/출력(I/O) 매개변수입니다.
PARAMETER_NAME	PARMNAME	VARCHAR(128) 널값 가능	매개변수의 이름. 매개변수가 이름을 가지지 않는 경우 널값이 들어갑니다.

표 110. SYSPARMS 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
DATA_TYPE	DATA_TYPE	VARCHAR(128)	열 유형 **BIGINT** 큰 수 **INTEGER** 큰 수 **SMALLINT** 작은 수 **DECIMAL** 압축 십진수 **NUMERIC** 존(zone) 십진수 **DOUBLE PRECISION** 부동 소수점. 배정밀도 **REAL** 부동 소수점. REAL **CHARACTER** 고정 길이 문자 스트링 **CHARACTER VARYING** 가변 길이의 문자 스트링 **CHARACTER LARGE OBJECT** 문자 LOB(Large Object) 스트링 **GRAPHIC** 고정 길이 그래픽 스트링 **GRAPHIC VARYING** 가변 길이의 그래픽 스트링 **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2바이트 문자 LOB(Large Object) 스트링 **BINARY LARGE OBJECT** 2진 LOB(Large Object) 스트링 **DATE** 날짜 **TIME** 시간 **TIMESTAMP** 시간소인 **DATALINK** 자료 링크 **ROWID** 행 ID **DISTINCT** 고유한 유형
NUMERIC_SCALE	SCALE	INTEGER 널값 가능	숫자 자료의 스케일 매개변수가 십진, 숫자 또는 2진이 아닌 경우 널값이 들어갑니다.

SYSPARMS

표 110. SYSPARMS 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
NUMERIC_PRECISION	PRECISION	INTEGER 널값 가능	전체 숫자 매개변수의 정밀도 **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 매개변수가 숫자가 아닌 경우 널값이 들어갑니다.
CCSID	CCSID	INTEGER 널값 가능	CHAR, VARCHAR, CLOB, DATE, TIME, TIMESTAMP, GRAPHIC, VARGRAPHIC, DBCLOB 및 DATALINK 매개변수에 대한 코드화 문자 세트 ID(CCSID) 값 매개변수가 숫자인 경우 널값이 들어갑니다.
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 스트링 최대 길이 매개변수가 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 바이트 수 매개변수가 스트링이 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 매개변수가 숫자가 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	DATPRC	INTEGER 널값 가능	날짜, 시간 또는 시간소인의 분수 부분. **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 매개변수가 날짜, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.

표 110. SYSPARMS 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
IS_NULLABLE	NULLS	VARCHAR(3)	매개변수의 널값 가능 여부를 표시합니다. **NO** 매개변수에서 널값을 허용하지 않습니다. **YES** 매개변수에서 널값을 허용합니다.
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
ROW_TYPE	ROWTYPE	CHAR(1)	행의 유형을 표시합니다. 프로시듀어에 대한 매개변수의 경우 이 열에는 널값이 들어갑니다. **P** 매개변수 **R** 캐스트 전의 결과 **C** 캐스트 후의 결과
DATA_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) 널값 가능	고유한 유형인 경우 자료 유형의 스키마 (schema). 매개변수가 고유한 유형이 아닌 경우 널값이 들어갑니다.
DATA_TYPE_NAME	TYPENAME	VARCHAR(128) 널값 가능	고유한 유형인 경우 자료 유형의 이름. 매개변수가 고유한 유형이 아닌 경우 널값이 들어갑니다.
AS_LOCATOR	ASLOCATOR	VARCHAR(3)	매개변수가 로케이터로서 명시되었는지의 여부를 표시합니다. **NO** 매개변수가 로케이터로 명시되지 않았습니다. **YES** 매개변수가 로케이터로 명시되었습니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSPROCS

SYSPROCS 뷰에는 CREATE PROCEDURE 명령문에 의해 작성된 각 프로시듀어에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSPROCS 뷰의 열을 설명합니다.

표 111. SYSPROCS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	루틴(프로시듀어) 인스턴스의 스키마(schema) 이름
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	루틴 인스턴스의 고유 이름
ROUTINE_SCHEMA	PROCSCHEMA	VARCHAR(128)	루틴이 들어 있는 SQL 스키마(schema)명
ROUTINE_NAME	PROCNAME	VARCHAR(128)	루틴 명
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	루틴이 작성된 시간소인을 식별합니다.
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	루틴을 정의한 사용자 명
ROUTINE_BODY	BODY	VARCHAR(8)	루틴 본문의 유형 **EXTERNAL** 외부 루틴입니다. **SQL** SQL 루틴입니다.
EXTERNAL_NAME	EXTNAME	VARCHAR(279) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. • REXX의 경우 외부 프로그램명은 *schema-name/source-file-name(member-name)*입니다. • Java 프로그램의 경우 외부 프로그램명은 선택적인 jar-id 다음에 *fully-qualified-class-name!method-name* 또는 *fully-qualified-class-name.method-name*이 나옵니다. • 다른 모든 언어의 경우 외부 프로그램명은 *schema-name/program-name*입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

표 111. SYSPROCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. **C** 외부 프로그램이 C로 작성됩니다. **C++** 외부 프로그램이 C++로 작성됩니다. **CL** 외부 프로그램이 CL로 작성됩니다. **COBOL** 외부 프로그램이 COBOL로 작성됩니다. **COBOLLE** 외부 프로그램은 ILE COBOL로 작성됩니다. **FORTRAN** 외부 프로그램이 FORTRAN으로 작성됩니다. **JAVA** 외부 프로그램이 JAVA로 작성됩니다. **PLI** 외부 프로그램이 PL/I으로 작성됩니다. **REXX** 외부 프로그램은 REXX 프로시듀어입니다. **RPG** 외부 프로그램이 RPG로 작성됩니다. **RPGLE** 외부 프로그램은 ILE RPG로 작성됩니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

SYSPROCS

표 111. SYSPROCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) 널값 가능	외부 루틴인 경우 이 열은 매개변수 유형을 식별합니다(호출 규칙). **DB2GNRL** DB2GENERAL 호출 규칙입니다. **DB2SQL** DB2SQL 호출 규칙입니다. **GENERAL** GENERAL 호출 규칙입니다. **JAVA** JAVA 호출 규칙입니다. **NULLS** GENERAL WITH NULLS 호출 규칙입니다. **SQL** SQL 표준 호출 규칙입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	이 열은 루틴이 결정적인지 여부를 식별합니다. 다시 말해, 동일한 인수로 루틴을 호출하는 경우 항상 동일한 결과가 리턴되는지의 여부를 식별하는 것입니다. **NO** 루틴이 결정적이 아닙니다. **YES** 루틴이 결정적입니다.
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	이 열은 SQL이 들어 있는 루틴인지의 여부 및 자료를 읽고 변경하는지 여부를 식별합니다. **NONE** 루틴에 SQL문이 없습니다. **CONTAINS** 루틴에 SQL문이 있습니다. **READS** 루틴이 표 또는 뷰로부터 자료를 읽을 수 있습니다. **MODIFIES** 루틴이 표 또는 뷰 내의 자료를 변경하거나 SQL DDL(data definition language) 명령문을 생성할 수 있습니다.

표 111. SYSPROCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SQL_PATH	SQL_PATH	VARCHAR(3483) 널값 가능	SQL 루틴인 경우 이 열은 경로를 식별합니다. 이것이 SQL 루틴이 아닌 경우 널 값이 들어갑니다.
PARM_SIGNATURE	SIGNATURE	VARCHAR(510)	이 열은 루틴 서명을 식별합니다.
RESULT_SETS	RESULTS	SMALLINT	리턴된 결과 세트의 최대수를 식별합니다. 0은 결과 세트가 없음을 의미합니다.
IN_PARMS	IN_PARMS	SMALLINT	입력 매개변수의 수를 식별합니다. 0은 입력 매개변수가 없음을 의미합니다.
OUT_PARMS	OUT_PARMS	SMALLINT	출력 매개변수의 수를 식별합니다. 0은 출력 매개변수가 없음을 의미합니다.
INOUT_PARMS	INOUT_PARM	SMALLINT	입/출력(I/O) 매개변수의 수를 식별합니다. 0은 입/출력(I/O) 매개변수가 없음을 의미합니다.
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
ROUTINE_DEFINITION	ROUTINEDEF	VARCHAR(24000) 널값 가능	SQL 루틴인 경우 이 열에는 SQL 루틴 본문이 들어갑니다. 이것이 SQL 루틴이 아니거나 루틴 본문이 절단 없이 이 열에 들어있을 수 없는 경우에 널 값이 들어갑니다.
DBINFO	DBINFO	VARCHAR(3) 널값 가능	데이터베이스에 대한 정보가 프로시듀어에 전달되는지 여부를 식별합니다. **NO** 데이터베이스 정보가 프로시듀어에 전달되지 않습니다. **YES** 데이터베이스에 관한 정보가 프로시듀어에 전달됩니다.
COMMIT_ON_RETURN	CMTONRET	VARCHAR(3) 널값 가능	이 열은 프로시듀어로부터 성공적으로 리턴될 때 프로시듀어가 확약하는지 여부를 지정합니다. **NO** 프로시듀어로부터 성공적으로 리턴될 때 확약이 수행되지 않습니다. **YES** 프로시듀어로부터 성공적으로 리턴될 때 확약이 수행됩니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

SYSPROCS

표 111. SYSPROCS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
NEW_SAVEPOINT_LEVEL	NEWSAVEPTL	VARCHAR(3) 널값 가능	이 열은 루틴이 새 저장점 레벨을 시작하는지 여부를 지정합니다. **NO** 새 저장점 레벨이 시작되지 않습니다. **YES** 새 저장점 레벨이 시작됩니다.

SYSREFCST

SYSREFCST 뷰에는 SQL 스키마의 각 외부 키에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSREFCST 뷰의 열을 설명합니다.

표 112. SYSREFCST 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	제한사항이 들어 있는 스키마명.
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	제약 조건 명
UNIQUE_CONSTRAINT_SCHEMA	UNQDBNAME	VARCHAR(128)	참조 제한사항에 의해 참조되는 고유한 제한사항이 들어 있는 SQL 스키마명
UNIQUE_CONSTRAINT_NAME	UNQNAME	VARCHAR(128)	참조 제약 조건에 의해 참조되는 고유한 제약 조건 명
MATCH_OPTION	MATCH	VARCHAR(7)	일치 옵션. 항상 없음(none)입니다.
UPDATE_RULE	UPDATE	VARCHAR(11)	갱신 규칙 • NO ACTION • RESTRICT
DELETE_RULE	DELETE	VARCHAR(11)	삭제 규칙 • NO ACTION • CASCADE • SET NULL • SET DEFAULT • RESTRICT
COLUMN_COUNT	COLCOUNT	INTEGER	외부 키 안의 열의 수

SYSROUTINEDEP

SYSROUTINEDEP 뷰는 루틴의 종속성을 기록합니다. 다음 표에서는 SYSROUTINEDEP 뷰의 열을 설명합니다.

표 113. SYSROUTINEDEP 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	루틴 인스턴스의 스키마(schema) 이름
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	루틴 인스턴스의 고유 이름
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	오브젝트가 들어 있는 SQL 스키마명.
OBJECT_NAME	BNAME	VARCHAR(128)	루틴이 의존하는 오브젝트 이름.
OBJECT_TYPE	BTYPE	CHAR(10)	루틴에서 참조된 오브젝트의 오브젝트 유형을 나타냅니다. **ALIAS** 오브젝트는 별명입니다. **FUNCTION** 오브젝트는 함수입니다. **INDEX** 오브젝트는 색인입니다. **PROCEDURE** 오브젝트는 프로시듀어입니다. **SCHEMA** 오브젝트는 스키마입니다. **TABLE** 오브젝트는 표입니다. **TYPE** 오브젝트는 고유한 유형입니다. **VIEW** 오브젝트는 뷰입니다.
PARM_SIGNATURE	SIGNATURE	VARCHAR(10000) 널값 가능	이 열은 루틴 서명을 식별합니다. 오브젝트가 루틴이 아닌 경우 널값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	오브젝트의 독립 보조 기억장치 풀(IASP) 수를 지정합니다.
NUMBER_OF_PARMS	NUMPARMS	SMALLINT 널값 가능	매개변수의 수를 식별합니다. 오브젝트가 루틴이 아닌 경우 널값이 들어갑니다.

SYSROUTINES

SYSROUTINES 표에는 CREATE PROCEDURE 명령문에 의해 작성된 프로시듀어 또는 CREATE FUNCTION 명령문에 의해 작성된 함수의 각 매개변수에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSROUTINES 뷰의 열을 설명합니다.

표 114. SYSROUTINES 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	루틴 인스턴스의 스키마(schema) 이름
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	루틴 인스턴스의 고유 이름
ROUTINE_SCHEMA	RTNSCHEMA	VARCHAR(128)	루틴이 들어 있는 SQL 스키마(schema)명
ROUTINE_NAME	RTNNAME	VARCHAR(128)	루틴 명
ROUTINE_TYPE	RTNTYPE	VARCHAR(9)	루틴 유형 **PROCEDURE** 프로시듀어입니다. **FUNCTION** 함수입니다.
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	루틴이 작성된 시간소인을 식별합니다.
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	루틴을 정의한 사용자 명
ROUTINE_BODY	BODY	VARCHAR(8)	루틴 본문의 유형 **EXTERNAL** 외부 루틴입니다. **SQL** SQL 루틴입니다.
EXTERNAL_NAME	EXTNAME	VARCHAR(279) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. • REXX의 경우 외부 프로그램명은 *schema-name/source-file-name(member-name)*입니다. • ILE 서비스 프로그램의 경우 외부 프로그램명은 *schema-name/service-program-name(entry-point-name)*입니다. • Java 프로그램의 경우 외부 프로그램명은 선택적인 jar-id 다음에 *fully-qualified-class-name!method-name* 또는 *fully-qualified-class-name.method-name*이 나옵니다. • 다른 모든 언어의 경우 외부 프로그램명은 *schema-name/program-name*입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. **C** 외부 프로그램이 C로 작성됩니다. **C++** 외부 프로그램이 C++로 작성됩니다. **CL** 외부 프로그램이 CL로 작성됩니다. **COBOL** 외부 프로그램이 COBOL로 작성됩니다. **COBOLLE** 외부 프로그램은 ILE COBOL로 작성됩니다. **FORTRAN** 외부 프로그램이 FORTRAN으로 작성됩니다. **JAVA** 외부 프로그램이 JAVA로 작성됩니다. **PLI** 외부 프로그램이 PL/I으로 작성됩니다. **REXX** 외부 프로그램은 REXX 프로시듀어입니다. **RPG** 외부 프로그램이 RPG로 작성됩니다. **RPGLE** 외부 프로그램은 ILE RPG로 작성됩니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) 널값 가능	외부 루틴인 경우 이 열은 매개변수 유형을 식별합니다(호출 규칙). **DB2GNRL** DB2GENERAL 호출 규칙입니다. **DB2SQL** DB2SQL 호출 규칙입니다. **GENERAL** GENERAL 호출 규칙입니다. **JAVA** JAVA 호출 규칙입니다. **NULLS** GENERAL WITH NULLS 호출 규칙입니다. **SQL** SQL 표준 호출 규칙입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	이 열은 루틴이 결정적인지 여부를 식별합니다. 다시 말해, 동일한 인수로 루틴을 호출하는 경우 항상 동일한 결과가 리턴되는지의 여부를 식별하는 것입니다. **NO** 루틴이 결정적이 아닙니다. **YES** 루틴이 결정적입니다.
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	이 열은 SQL이 들어 있는 루틴인지의 여부 및 자료를 읽고 변경하는지 여부를 식별합니다. **NONE** 루틴에 SQL문이 없습니다. **CONTAINS** 루틴에 SQL문이 있습니다. **READS** 루틴이 표 또는 뷰로부터 자료를 읽을 수 있습니다. **MODIFIES** 루틴이 표 또는 뷰 내의 자료를 변경하거나 SQL DDL(data definition language) 명령문을 생성할 수 있습니다.

SYSROUTINES

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SQL_PATH	SQL_PATH	VARCHAR(3483) 널값 가능	SQL 루틴인 경우 이 열은 경로를 식별합니다. 이것이 SQL 루틴이 아닌 경우 널 값이 들어갑니다.
PARM_SIGNATURE	SIGNATURE	VARCHAR(510)	이 열은 루틴 서명을 식별합니다.
NUMBER_OF_RESULTS	NUMRESULTS	SMALLINT	결과의 수를 식별합니다.
MAX_DYNAMIC_RESULT_SETS	RESULTS	SMALLINT	리턴된 결과 세트의 최대수를 식별합니다. 0은 결과 세트가 없음을 의미합니다.
IN_PARMS	IN_PARMS	SMALLINT	입력 매개변수의 수를 식별합니다. 0은 입력 매개변수가 없음을 의미합니다.
OUT_PARMS	OUT_PARMS	SMALLINT	출력 매개변수의 수를 식별합니다. 0은 출력 매개변수가 없음을 의미합니다.
INOUT_PARMS	INOUT_PARM	SMALLINT	입/출력(I/O) 매개변수의 수를 식별합니다. 0은 입/출력(I/O) 매개변수가 없음을 의미합니다.
PARSE_TREE	PARSE_TREE	VARCHAR(666) FOR BIT DATA	루틴인 경우 이 열은 CREATE FUNCTION 또는 CREATE PROCEDURE 명령문의 분석 트리를 식별합니다. 이것은 내부에서만 사용됩니다.
PARM_ARRAY	PARM_ARRAY	VARCHAR(10008) FOR BIT DATA	외부 루틴인 경우 이 열은 CREATE FUNCTION 또는 CREATE PROCEDURE 명령문으로부터 빌드된 매개변수 배열을 식별합니다. 이것은 내부에서만 사용됩니다.
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
ROUTINE_DEFINITION	ROUTINEDEF	DBCLOB(1048576) 널값 가능	SQL 루틴인 경우 이 열에는 SQL 루틴 본문이 들어갑니다. 이것이 SQL 루틴이 아니거나 루틴 본문이 절단 없이 이 열에 들어있을 수 없는 경우에 널 값이 들어갑니다.

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
FUNCTION_ORIGIN	ORIGIN	CHAR(1)	함수 유형을 식별합니다. 만일 프로시듀어이면 이 열은 공백으로 남습니다. **B** 내장 함수(built-in function)입니다(iSeries용 DB2 UDB에서 정의되었습니다). **E** 사용자 정의 함수입니다. **U** 다른 함수에 기반한 사용자 정의 함수입니다. **S** 시스템 생성 함수입니다.
FUNCTION_TYPE	TYPE	CHAR(1)	함수 형식을 식별합니다. 만일 프로시듀어이면 이 열은 공백으로 남습니다. **S** 스칼라 함수입니다. **C** 열 함수(column function)입니다. **T** 표 함수입니다.
EXTERNAL_ACTION	EXTACTION	CHAR(1) 널값 가능	함수 호출이 외부에 영향을 미칠지의 여부를 식별합니다. **E** 외부 부작용이 있습니다. **N** 외부 부작용이 없습니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
IS_NULL_CALL	NULL_CALL	VARCHAR(3) 널값 가능	입력 매개변수가 널값인 경우 함수를 호출할 필요가 있는지의 여부를 식별합니다. **NO** 입력 매개변수가 널값인 경우 함수를 호출할 필요가 없습니다. 이것이 스칼라 함수라면, 만일 피연산자 중 하나가 널값인 경우 함수의 결과는 내재적으로 널값이 됩니다. 이것이 표 함수라면, 만일 피연산자 중 하나가 널값인 경우 함수의 결과는 빈 표가 됩니다. **YES** 입력 연산자가 널(null)인 경우라도 이 함수를 반드시 호출하여야 합니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.

SYSROUTINES

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SCRATCH_PAD	SCRATCHPAD	INTEGER 널값 가능	정적 메모리 영역(스크래치 패드)의 주소가 함수에 전달되는지의 여부를 식별합니다. **0** 이 함수는 스크래치 패드가 없습니다. **정수** 함수에 전달되는 스크래치 패드의 크기를 표시합니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
FINAL_CALL	FINAL_CALL	VARCHAR(3) 널값 가능	함수의 최종 호출이 함수 작업 영역(스크래치 패드) 제거를 허용하는지의 여부를 표시합니다. **NO** 최종 호출이 작성되지 않습니다. **YES** 명령문이 완료되면 함수에 대한 최종 호출이 작성됩니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
PARALLELIZABLE	PARALLEL	VARCHAR(3) 널값 가능	함수의 병행 실행이 가능한지 여부를 식별합니다. **NO** 함수의 병행 실행이 불가능합니다. **YES** 함수의 병행 실행이 가능합니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
DBINFO	DBINFO	VARCHAR(3) 널값 가능	데이터베이스에 대한 정보가 루틴에 전달되는지 여부를 식별합니다. **NO** 데이터베이스 정보가 루틴에 전달되지 않습니다. **YES** 데이터베이스에 관한 정보가 루틴에 전달됩니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
SOURCE_SPECIFIC_SCHEMA	SRCSCHEMA	VARCHAR(128) 널값 가능	이것이 피소스(sourced) 함수이며 소스가 사용자 정의된 것이면 이 열에는 소스 스키마의 이름이 들어갑니다. 이것이 피소스(sourced) 함수이며 소스가 내장된 것이면 이 열에는 ‘QSYS2’가 들어갑니다. 루틴이 피소스(sourced) 함수가 아닌 경우 널값이 들어갑니다.

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
SOURCE_SPECIFIC_NAME	SRCNAME	VARCHAR(128) 널값 가능	이것이 피소스(sourced) 함수이며 소스가 사용자 정의된 것이면 이 열에는 소스 함수명의 고유 이름이 들어갑니다. 루틴이 피소스(sourced) 함수가 아닌 경우 널값이 들어갑니다.
IS_USER_ DEFINED_CAST	CAST_FUNC	VARCHAR(3) 널값 가능	함수가 고유한 유형이 작성될 때 작성된 캐스트 함수인지의 여부를 식별합니다. **NO** 캐스트 함수가 아닙니다. **YES** 캐스트 함수입니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
CARDINALITY	CARD	BIGINT 널값 가능	표 함수 중요성을 지정합니다. 표 함수가 아니고 중요성이 지정되지 않은 경우 널값을 포함합니다.
FENCED	FENCED	VARCHAR(3) 널값 가능	함수가 분리되었는지 여부를 식별합니다. **NO** 함수가 분리되지 않습니다. **YES** 함수가 분리됩니다. 루틴이 프로시듀어인 경우 널값이 들어갑니다.
COMMIT_ON_RETURN	CMTONRET	VARCHAR(3) 널값 가능	이 열은 프로시듀어로부터 성공적으로 리턴될 때 프로시듀어가 확약하는지 여부를 지정합니다. **NO** 프로시듀어로부터 성공적으로 리턴될 때 확약이 수행되지 않습니다. **YES** 프로시듀어로부터 성공적으로 리턴될 때 확약이 수행됩니다. 루틴이 함수인 경우 널값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.
NEW_SAVEPOINT_LEVEL	NEWSAVEPTL	VARCHAR(3) 널값 가능	이 열은 루틴이 새 저장점 레벨을 시작하는지 여부를 지정합니다. **NO** 새 저장점 레벨이 시작되지 않습니다. **YES** 새 저장점 레벨이 시작됩니다. 루틴이 함수인 경우 널값이 들어갑니다.
LAST_ALTERED	ALTEREDTS	TIMESTAMP 널값 가능	루틴이 마지막으로 변경한 시간소인 널 값이 들어갑니다.

표 114. SYSROUTINES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
DEBUG_MODE	DEBUG_MODE	CHAR(1)	루틴이 디버그 가능한지 여부를 지정합니다. **0** 루틴이 디버그 가능하지 않습니다. **2** 루틴이 디버그 가능합니다.
DEBUG_DATA	DEBUG_DATA	CLOB(1048576) 널값 가능	예약. 널 값이 들어갑니다.

SYSTABLES

SYSTABLES 뷰에는 SQL 카탈로그의 표 및 뷰를 포함하여 SQL 스키마의 각 표, 뷰 또는 별명에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSTABLES 뷰의 열을 설명합니다.

표 115. SYSTABLES 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TABLE_NAME	NAME	VARCHAR(128)	표, 뷰 또는 별명 이름. 이름이 있으면 그 이름이 SQL 표, 뷰 또는 별명의 이름이 됩니다. 아니면 그 이름이 시스템 표, 뷰 또는 별명 이름이 됩니다.
TABLE_OWNER	CREATOR	VARCHAR(128)	표, 뷰 또는 별명 소유자
TABLE_TYPE	TYPE	CHAR(1)	행에서 표, 뷰 또는 별명을 설명하는 경우. **A** 별명 **L** 논리 파일 **P** 실제 파일(PF) **T** 표 **V** 뷰
COLUMN_COUNT	COLCOUNT	INTEGER	표 또는 뷰 안의 열의 수. 별명의 경우 제로값입니다.
ROW_LENGTH	RECLENGTH [82]	INTEGER	표 내의 레코드 최대 길이. 별명의 경우 제로값입니다.
TABLE_TEXT	LABEL	VARCHAR(50)	문자 스트링은 LABEL문으로 제공됩니다.
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
TABLE_SCHEMA	DBNAME	VARCHAR(128)	표, 뷰 또는 별명이 들어 있는 SQL 스키마명
LAST_ALTERED_TIMESTAMP	ALTEREDTS	TIMESTAMP	표 최종 변경 시간소인
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	시스템 표 이름
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	시스템 스키마명
FILE_TYPE	FILETYPE	CHAR(1)	파일 유형 **D** 자료 파일 또는 별명 **S** 소스 파일
BASE_TABLE_SCHEMA	TBDBNAME	VARCHAR(128) 널값 가능	별명의 경우 별명이 기초로 하고 있는 표 또는 뷰가 들어 있는 SQL 스키마명이 됩니다. 표가 별명이 아닌 경우 널값이 들어갑니다.

SYSTABLES

표 115. SYSTABLES 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
BASE_TABLE_NAME	TBNAME	VARCHAR(128) 널값 가능	별명의 경우 별명이 기초하는 표 또는 뷰의 이름이 됩니다. 표가 별명이 아닌 경우 널값이 들어갑니다.
BASE_TABLE_MEMBER	TBMEMBER	VARCHAR(10) 널값 가능	별명의 경우 별명이 기초하는 파일 멤버명이 됩니다. 만일 별명이면 *FIRST가 들어갑니다. 하지만 멤버명이 지정되지는 않습니다. 표가 별명이 아닌 경우 널값이 들어갑니다.
SYSTEM_TABLE	SYSTABLE	CHAR(1)	시스템 표 **N** 표는 시스템 표가 아닙니다. **Y** 표는 시스템 표입니다.
SELECT_OMIT	SELECTOMIT	CHAR(1)	선택/생략 논리 파일 **N** 표는 선택/생략 논리 파일이 아닙니다. **Y** 표는 선택/생략 논리 파일입니다.
IS_INSERTABLE_INTO	INSERTABLE	VARCHAR(3)	표에서 INSERT가 허용되는지 여부를 식별합니다. **NO** INSERT는 이 표에서 허용되지 않습니다. **YES** INSERT는 이 표에서 허용됩니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

82. 길이는 데이터베이스 버퍼에 전달된 바이트 수이며, 내부 기억장치의 길이가 아닙니다.

SYSTRIGCOL

SYSTRIGCOL 뷰에는 WHEN절에 내재적으로든 명시적으로든 참조되어 있는 각 열 또는 트리거의 트리거된 SQL문에 대하여 하나의 행이 들어 있습니다. 다음 표에서는 SYSTRIGCOL 뷰의 열을 설명합니다.

표 116. SYSTRIGCOL 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	트리거가 들어 있는 스키마명
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	트리거명
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	트리거에서 참조된 열을 포함하는 표나 뷰가 들어 있는 스키마명
TABLE_NAME	TABNAME	VARCHAR(128)	트리거에서 참조된 열이 들어 있는 표나 뷰의 이름
COLUMN_NAME	TABCOLUMN	VARCHAR(128)	트리거에서 참조된 열의 이름
OBJECT_TYPE	BTYPE	VARCHAR(10)	트리거에서 참조된 열이 들어 있는 오브젝트의 오브젝트 유형을 나타냅니다. **FUNCTION** 오브젝트는 함수입니다. **TABLE** 오브젝트는 표입니다. **VIEW** 오브젝트는 뷰입니다.

SYSTRIGDEP

SYSTRIGDEP 뷰에는 WHEN절에 참조되어 있는 각 오브젝트 또는 트리거의 트리거된 SQL문에 대하여 하나의 행이 들어 있습니다. 다음 표에서는 SYSTRIGDEP 뷰의 열을 설명합니다.

표 117. SYSTRIGDEP 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	트리거가 들어 있는 스키마명
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	트리거명
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	트리거에서 참조된 오브젝트가 들어 있는 스키마명
OBJECT_NAME	BNAME	VARCHAR(128)	트리거에서 참조된 오브젝트명
OBJECT_TYPE	BTYPE	CHAR(10)	트리거에서 참조된 오브젝트의 오브젝트 유형을 나타냅니다. **ALIAS** 오브젝트는 별명입니다. **FUNCTION** 오브젝트는 함수입니다. **INDEX** 오브젝트는 색인입니다. **PACKAGE** 오브젝트는 패키지입니다. **PROCEDURE** 오브젝트는 프로시듀어입니다. **SCHEMA** 오브젝트는 스키마입니다. **TABLE** 오브젝트는 표입니다. **TYPE** 오브젝트는 고유한 유형입니다. **VIEW** 오브젝트는 뷰입니다.
PARM_SIGNATURE	SIGNATURE	VARCHAR(10000) 널값 가능	이 열은 루틴 서명을 식별합니다. 오브젝트가 루틴이 아닌 경우 널값이 들어갑니다.

SYSTRIGGERS

SYSTRIGGERS 뷰에는 SQL 스키마의 각 트리거에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSTRIGGERS 뷰의 열을 설명합니다.

표 118. SYSTRIGGERS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	트리거가 들어 있는 스키마명
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	트리거명
EVENT_MANIPULATION	TRIGEVENT	VARCHAR(6)	트리거가 실행되도록 하는 이벤트를 나타냅니다. **DELETE** DELETE시 트리거 실행. **INSERT** INSERT시 트리거 실행. **UPDATE** DELETE시 트리거 실행. **READ** 행이 읽혀질 때 트리거가 실행됩니다. 이것은 ADDPFTRG 명령으로 작성된 트리거에서만 사용될 수 있습니다.
EVENT_OBJECT_SCHEMA	TABSCHEMA	VARCHAR(128)	트리거의 주제표가 들어 있는 스키마명
EVENT_OBJECT_TABLE	TABNAME	VARCHAR(128)	트리거의 주제표 이름
ACTION_ORDER	ORDERSEQNO	INTEGER	해당 표에 대한 트리거 리스트에서 이 트리거의 위치. 이것은 트리거가 실행될 순서를 나타냅니다.
ACTION_CONDITION	CONDITION	DBCLOB(1048576) 널값 가능	이 트리거에 대한 WHEN절의 텍스트 WHEN절이 없는 경우 널값이 들어갑니다.
ACTION_STATEMENT	TEXT	DBCLOB(1048576) 널값 가능	트리거 조치의 SQL문 텍스트 ADDPFTRG 명령으로 작성된 트리거인 경우 널값이 들어갑니다.
ACTION_ORIENTATION	GRANULAR	VARCHAR(9)	ROW 트리거인지 STATEMENT 트리거인지를 나타냅니다. **ROW** 각 행에 대하여 트리거가 실행됩니다. **STATEMENT** 각 명령문에 대하여 트리거가 실행됩니다.

SYSTRIGGERS

표 118. SYSTRIGGERS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
ACTION_TIMING	TRIGTIME	VARCHAR(6)	이전(BEFORE) 트리거인지 이후(AFTER) 트리거인지를 나타냅니다. **BEFORE** 트리거가 실행된 후 트리거 이벤트가 실행됩니다. **AFTER** 트리거 이벤트가 실행되어야 트리거가 실행됩니다.
TRIGGER_MODE	TRIGMODE	VARCHAR(6)	트리거 실행 모드를 나타냅니다. **DB2SQL** 트리거 모드는 DB2SQL입니다. **DB2ROW** 트리거 모드는 DB2ROW입니다.
ACTION_REFERENCE_OLD_ROW	OLD_ROW	VARCHAR(128) 널값 가능	OLD ROW 상관명의 이름 OLD ROW 상관명이 지정되지 않았으면 널값이 들어갑니다.
ACTION_REFERENCE_NEW_ROW	NEW_ROW	VARCHAR(128) 널값 가능	NEW ROW 상관명의 이름 NEW ROW 상관명이 지정되지 않았으면 널값이 들어갑니다.
ACTION_REFERENCE_OLD_TABLE	OLD_TABLE	VARCHAR(128) 널값 가능	OLD TABLE 상관명의 이름 OLD TABLE 상관명이 지정되지 않았으면 널값이 들어갑니다.
ACTION_REFERENCE_NEW_TABLE	NEW_TABLE	VARCHAR(128) 널값 가능	NEW TABLE 상관명의 이름 NEW TABLE 상관명이 지정되지 않았으면 널값이 들어갑니다.
SQL_PATH	SQL_PATH	VARCHAR(3483) 널값 가능	트리거가 작성될 때 작성된 SQL 경로 ADDPFTRG 명령으로 작성된 트리거인 경우 널값이 들어갑니다.
CREATED	CREATE_DTS	TIMESTAMP	트리거가 작성되었을 때의 시간소인
TRIGGER_PROGRAM_NAME	TRIGPGM	VARCHAR(128)	트리거 프로그램명
TRIGGER_PROGRAM_LIBRARY	TRIGPGMLIB	VARCHAR(128)	트리거 프로그램이 들어 있는 스키마의 시스템명
OPERATIVE	OPERATIVE	VARCHAR(1)	트리거가 작동할 수 있는지(멤버가 있는 파일과 연관되어 있는지) 여부를 나타냅니다. **Y** 트리거가 작동가능합니다. **N** 트리거가 작동가능하지 않습니다.

표 118. SYSTRIGGERS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
ENABLED	ENABLED	VARCHAR(1)	트리거가 작동할 수 있는지 여부를 나타냅니다(CL 명령 CHGPFTRG 참조). **Y** 트리거가 작동가능하지 않습니다. **N** 트리거가 작동불가능합니다.
THREADSAFE	THDSAFE	VARCHAR(8)	트리거가 스레드 세이프한지 여부를 나타냅니다. **YES** 트리거가 스레드 세이프합니다. **NO** 트리거가 스레드 세이프하지 않습니다. **UNKNOWN** 알 수 없는의 스레드 안정성을 알 수 없습니다.
MULTITHREADED_JOB_ACTION	MLTTHDACN	VARCHAR(8)	멀티스레드 작업에서 트리거 프로그램이 호출될 때 취해야 할 조치를 나타냅니다. **SYSVAL** QMLTTHDACN 시스템 값을 사용하여 취해야 할 조치를 판별합니다. **MSG** 멀티스레드 작업에서 트리거 프로그램을 실행하지만 진단 메세지를 송신합니다. **NORUN** 멀티스레드 작업에서 트리거 프로그램을 실행하지 마십시오. **RUN** 멀티스레드 작업에서 트리거 프로그램을 실행합니다.
ALLOW_REPEATED_CHANGE	ALWREPCHG	VARCHAR(8)	갱신 이벤트가 트리거를 실행시킬 수 있는 조건을 나타냅니다. **YES** 트리거를 사용하여 같은 행에 대한 변경을 여러번 반복할 수 있습니다. **NO** 트리거가 같은 행에 대한 반복된 변경을 허용하지 않습니다.

SYSTRIGGERS

표 118. SYSTRIGGERS 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TRIGGER_UPDATE_CONDITION	TRGUPDCND	CHAR(8) 널값 가능	UPDATE 트리거가 갱신 이벤트시 항상 실행될 것인지 또는 열 값이 실제 변경되었을 경우에만 실행할 것인지 여부를 나타냅니다. **ALWAYS** 트리거는 갱신 이벤트시 항상 실행됩니다. **CHANGE** 트리거는 열 값이 실제 변경되었을 경우에만 갱신 이벤트시 실행됩니다. 트리거가 UPDATE 트리거가 아닌 경우 널값이 들어갑니다.
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.

SYSTRIGUPD

SYSTRIGUPD 뷰에는 UPDATE 열 리스트에서 식별된 각 열에 대한 하나의 행이 들어갑니다. 다음 표에서는 SYSTRIGUPD 뷰의 열을 설명합니다.

표 119. SYSTRIGUPD 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	트리거가 들어 있는 스키마명
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	트리거명
EVENT_OBJECT_SCHEMA	TABSCHEMA	VARCHAR(128)	트리거의 주제표가 들어 있는 스키마명
EVENT_OBJECT_TABLE	TABNAME	VARCHAR(128)	트리거의 주제표 이름
TRIGGERED_UPDATE_COLUMNS	TABCOLUMN	VARCHAR(128)	트리거의 UPDATE 열 리스트에 지정된 열 이름.

SYSTYPES

SYSTYPES 표에는 CREATE DISTINCT TYPE 명령문에 의해 작성된 각 내장 자료 유형과 각 고유한 유형에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSTYPES 표의 열에 대하여 설명합니다.

표 120. SYSTYPES 표

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128)	자료 유형의 스키마 이름
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128)	자료 유형 이름
USER_DEFINED_TYPE_DEFINER	DEFINER	VARCHAR(128)	자료 유형을 작성한 사용자명
SOURCE_SCHEMA	SRCSCHEMA	VARCHAR(128) 널값 가능	이 자료 유형에 대한 소스 자료 유형에 대한 스키마. 이 내장 자료 유형인 경우 널값이 들어갑니다.
SOURCE_TYPE	SRCTYPE	VARCHAR(128) 널값 가능	이 자료 유형의 소스 자료 유형 이름 이 내장 자료 유형인 경우 널값이 들어갑니다.
SYSTEM_TYPE_SCHEMA	SYSTSCHEMA	CHAR(10)	자료 정의 유형에 대한 시스템 스키마 이름
SYSTEM_TYPE_NAME	SYSTNAME	CHAR(10)	자료 정의 유형에 대한 시스템 이름
METATYPE	METATYPE	CHAR(1)	자료 유형의 유형을 표시합니다. **S** 시스템 사전정의 자료 유형 **T** 사용자 정의의 고유한 유형

표 120. SYSTYPES 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
LENGTH	LENGTH	INTEGER	자료 유형의 길이 속성 또는 십진수, 숫자나 0이 아닌 정밀도 2진 열의 경우 그 정밀도. **8바이트** BIGINT **4바이트** INTEGER **2바이트** SMALLINT **수 정밀도** DECIMAL **수 정밀도** NUMERIC **8바이트** FLOAT, FLOAT(n) 여기서 n은 25 - 53 또는 DOUBLE PRECISION **4바이트** FLOAT, FLOAT(n) 여기서 n은 1 - 24 사이 또는 REAL **스트링 길이** CHARACTER **스트링 최대 길이** VARCHAR 또는 CLOB **그래픽 스트링 길이** GRAPHIC **그래픽 스트링 최대 길이** VARGRAPHIC 또는 DBCLOB **2진 스트링의 최대 길이** BLOB **4바이트** DATE **3바이트** TIME **10바이트** TIMESTAMP **자료 링크 URL 및 주석의 최대 길이** DATALINK **40바이트** ROWID **소스 유형과 동일한 값** DISTINCT
NUMERIC_SCALE	SCALE	INTEGER 널값 가능	숫자 자료의 스케일 자료 유형이 십진, 숫자 또는 2진이 아닌 경우 널값이 들어갑니다.

SYSTYPES

표 120. SYSTYPES 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CCSID	CCSID	INTEGER 널값 가능	CHAR, VARCHAR, CLOB, DATE, TIME, TIMESTAMP, GRAPHIC, VARGRAPHIC, DBCLOB 및 DATALINK 자료 유형에 대한 CCSID 값. 자료 유형이 숫자인 경우 널값이 들어갑니다.

표 120. SYSTYPES 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명	
STORAGE	STORAGE	INTEGER	열에 대한 기억장치 요구사항	
			8바이트	BIGINT
			4바이트	INTEGER
			2바이트	SMALLINT
			(정밀도/2) + 1	DECIMAL
			수 정밀도	NUMERIC
			8바이트	FLOAT, FLOAT(n) 여기서 n은 25 - 53 또는 DOUBLE PRECISION
			4바이트	FLOAT, FLOAT(n) 여기서 n은 1 - 24 사이 또는 REAL
			스트링 길이	CHAR
			스트링 최대 길이 + 2	VARCHAR
			스트링 최대 길이 + 29	CLOB
			스트링 길이 * 2	GRAPHIC
			스트링 최대 길이 * 2 + 2	VARGRAPHIC
			스트링 최대 길이 * 2 + 29	DBCLOB
			4바이트	DATE
			3바이트	TIME
			10바이트	TIMESTAMP
			자료 링크 URL 및 주석의 최대 길이 + 24	DATALINK
			42바이트	ROWID
			소스 유형과 동일한 값	DISTINCT
			주: 이 열은 모든 자료 유형에 대한 기억장치 요구사항을 지원합니다.	

SYSTYPES

표 120. SYSTYPES 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
NUMERIC_PRECISION	PRECISION	INTEGER 널값 가능	모든 숫자 자료 유형의 정밀도. **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 자료 유형이 숫자가 아닌 경우 널값이 들어갑니다.
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 스트링 최대 길이 자료 유형이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 바이트 수 자료 유형이 스트링이 아닌 경우 널값이 들어갑니다.
ALLOCATE	ALLOCATE	INTEGER 널값 가능	2진수, 가변 길이의 문자 및 가변 길이의 그래픽 스트링 자료 유형의 스트링의 할당 길이 자료 유형이 숫자 또는 고정 길이인 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 자료 유형이 숫자가 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	DATPRC	INTEGER 널값 가능	날짜, 시간 또는 시간소인의 분수 부분 **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 자료 유형이 날짜, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.

표 120. SYSTYPES 표 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
CREATE_TIME	CRTTIME	TIMESTAMP	자료 유형이 작성될 때의 시간소인을 식별합니다.
LONG_COMMENT	REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	자료 유형의 독립 보조 기억장치 풀(IASP) 수를 지정합니다.
LAST_ ALTERED	ALTEREDTS	TIMESTAMP 널값 가능	예약. 널 값이 들어갑니다.

SYSVIEWDEP

SYSVIEWDEP 뷰에서는 SQL 카탈로그의 뷰를 포함하여 표에 대한 뷰의 의존성을 작성합니다. 다음 표에서는 SYSVIEWDEP 뷰의 열을 설명합니다.

표 121. SYSVIEWDEP 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
VIEW_NAME	DNAME	VARCHAR(128)	뷰 이름. 만일 있다면 SQL 뷰 이름이 됩니다. 그렇지 않다면 시스템 뷰 이름이 될 것입니다.
VIEW_OWNER	DCREATOR	VARCHAR(128)	뷰 소유자
OBJECT_NAME	ONAME	VARCHAR(128)	뷰가 의존하는 오브젝트 이름.
OBJECT_SCHEMA	OSCHEMA	VARCHAR(128)	뷰가 종속된 오브젝트가 들어 있는 SQL 스키마명
OBJECT_TYPE	OTYPE	CHAR(10)	뷰가 기초하는 오브젝트의 유형 **FUNCTION** 함수 **TABLE** 표 **TYPE** 고유한 유형 **VIEW** 뷰
VIEW_SCHEMA	DDBNAME	VARCHAR(128)	뷰의 스키마명
SYSTEM_VIEW_NAME	SYS_VNAME	CHAR(10)	시스템 뷰 이름
SYSTEM_VIEW_SCHEMA	SYS_VDNAME	CHAR(10)	시스템 뷰 스키마
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10) 널값 가능	시스템 표 이름 오브젝트가 함수 또는 고유한 유형인 경우 널값이 들어갑니다.
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10) 널값 가능	시스템 표 스키마 오브젝트가 함수 또는 고유한 유형인 경우 널값이 들어갑니다.
TABLE_NAME	BNAME	VARCHAR(128) 널값 가능	뷰가 의존하는 표 또는 뷰의 이름. 만일 있다면 SQL 뷰 이름이 됩니다. 그렇지 않다면 시스템 뷰 이름이 될 것입니다. 오브젝트가 함수 또는 고유한 유형인 경우 널값이 들어갑니다.
TABLE_OWNER	BCREATOR	VARCHAR(128) 널값 가능	뷰가 의존하는 표 또는 뷰의 소유자. 오브젝트가 함수 또는 고유한 유형인 경우 널값이 들어갑니다.
TABLE_SCHEMA	BDBNAME	VARCHAR(128) 널값 가능	뷰가 종속된 표 또는 뷰가 수록된 SQL 스키마명 오브젝트가 함수 또는 고유한 유형인 경우 널값이 들어갑니다.

표 121. SYSVIEWDEP 뷰 (계속)

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TABLE_TYPE	BTYPE	CHAR(1) 널값 가능	뷰가 기초하는 오브젝트의 유형. **T** 표 **P** 실제 파일(PF) **V** 뷰 **L** 논리 파일 오브젝트가 함수 또는 고유한 유형인 경우 널값이 들어갑니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.
PARM_SIGNATURE	SIGNATURE	VARCHAR(10000) 널값 가능	이 열은 루틴 서명을 식별합니다. 오브젝트가 루틴이 아닌 경우 널값이 들어갑니다.

SYSVIEWS

SYSVIEWS 뷰에는 SQL 카탈로그의 뷰를 포함하여 SQL 스키마의 각 뷰에 대한 하나의 행이 들어 있습니다. 다음 표에서는 SYSVIEWS 뷰의 열을 설명합니다.

표 122. SYSVIEWS 뷰

열 이름	시스템 열 이름 (Column Name)	자료 유형	설명
TABLE_NAME	NAME	VARCHAR(128)	뷰 이름. 만일 있다면 SQL 뷰 이름이 됩니다. 그렇지 않다면 시스템 뷰 이름이 될 것입니다.
VIEW_OWNER	CREATOR	VARCHAR(128)	뷰 소유자
SEQNO	SEQNO	INTEGER	이 행의 순번입니다. 항상 1입니다.
CHECK_OPTION	CHECK	CHAR(1)	뷰 상에 사용된 검사 옵션 **N** 검사 옵션이 지정되지 않았습니다. **Y** 로컬 옵션이 명시되었습니다. **C** 직렬 옵션이 명시되었습니다.
VIEW_DEFINITION	TEXT	VARCHAR(10000) 널값 가능	CREATE VIEW 명령문의 조회 표현식 부분. 뷰 정의가 열에 잘리지 않고 포함될 수 없는 경우 널값을 포함합니다.
IS_UPDATABLE	UPDATES	CHAR(1)	뷰의 갱신가능 여부를 명시합니다. **Y** 뷰의 갱신이 가능합니다. **N** 뷰의 읽기 만이 가능합니다.
TABLE_SCHEMA	DBNAME	VARCHAR(128)	뷰가 들어 있는 SQL 스키마명.
SYSTEM_VIEW_NAME	SYS_VNAME	CHAR(10)	시스템 뷰 이름
SYSTEM_VIEW_SCHEMA	SYS_VDNAME	CHAR(10)	시스템 뷰 스키마명
IS_INSERTABLE_INTO	INSERTABLE	VARCHAR(3)	뷰에서 INSERT가 허용되는지 여부를 식별합니다. **NO** INSERT는 이 뷰에서 허용되지 않습니다. **YES** INSERT는 이 뷰에서 허용됩니다.
IASP_NUMBER	IASPNUMBER	SMALLINT	독립 보조 기억장치 풀(IASP) 수를 지정합니다.

ODBC 및 JDBC 카탈로그 뷰

카탈로그에는 SYSIBM 라이브러리 내의 다음 뷰 및 표가 있습니다.

View Name	설명
998 페이지의 『SQLCOLPRIVILEGES』	열에 부여된 특권에 대한 정보
999 페이지의 『SQLCOLUMNS』	열 속성에 대한 정보
1004 페이지의 『SQLFOREIGNKEYS』	외부 키에 대한 정보
1005 페이지의 『SQLPRIMARYKEYS』	1차 키에 대한 정보
1006 페이지의 『SQLPROCEDURECOLS』	프로시듀어 매개변수에 대한 정보
1012 페이지의 『SQLPROCEDURES』	프로시듀어에 대한 정보
1013 페이지의 『SQLSCHEMAS』	스키마에 대한 정보
1014 페이지의 『SQLSPECIALCOLUMNS』	행을 고유하게 식별하는 데 사용할 수 있는 표 열에 대한 정보
1016 페이지의 『SQLSTATISTICS』	표에 대한 통계 정보
1017 페이지의 『SQLTABLEPRIVILEGES』	표에 부여된 특권에 대한 정보
1018 페이지의 『SQLTABLES』	표에 대한 정보
1019 페이지의 『SQLTYPEINFO』	표의 유형에 대한 정보
1024 페이지의 『SQLUDTS』	내장 자료 유형과 고유한 유형에 대한 정보

SQLCOLPRIVILEGES

SQLCOLPRIVILEGES 뷰는 열의 모든 권한 부여에 대한 하나의 행이 들어 있습니다. 이 카탈로그 뷰는 사용자가 열에 대한 권한을 가지고 있는지 여부를 판별하는 데 사용할 수 없습니다. 열을 사용할 특권은 그룹 사용자 프로파일 또는 특수한 권한(*ALLOBJ 등)을 통해 취득되기 때문입니다. 다음 표에서는 뷰의 열을 설명합니다.

표 123. SQLCOLPRIVILEGES 뷰

열 이름	자료 유형	설명
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	표가 들어 있는 SQL 스키마명.
TABLE_NAME	VARCHAR(128)	표 이름
COLUMN_NAME	VARCHAR(128)	열 이름
GRANTOR	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
GRANTEE	VARCHAR(128)	특권이 부여된 사용자 프로파일
PRIVILEGE	VARCHAR(10)	부여된 특권: **UPDATE** 권한은 열을 갱신 선택합니다. **REFERENCES** 권한은 참조된 제한 사항에서 열을 참조합니다.
IS_GRANTABLE	VARCHAR(3)	다른 사용자에게 특권을 부여할 수 있는지 여부를 나타냅니다. **NO** 특권을 부여할 수 없습니다. **YES** 특권을 부여할 수 있습니다.
DBNAME	VARCHAR(8) 널값 가능	예약. 열은 널 값이 들어갑니다.

SQLCOLUMNS

SQLCOLUMNS 뷰는 표, 뷰 또는 별명의 모든 열에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 124. SQLCOLUMNS 뷰

열 이름	자료 유형	설명	
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명	
TABLE_SCHEM	VARCHAR(128)	표가 들어 있는 SQL 스키마명.	
TABLE_NAME	VARCHAR(128)	표 이름	
COLUMN_NAME	VARCHAR(128)	열 이름	
DATA_TYPE	SMALLINT	열의 자료 유형:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
		17	DISTINCT

표 124. SQLCOLUMNS 뷰 (계속)

열 이름	자료 유형	설명
TYPE_NAME	VARCHAR(128)	열의 자료 유형 이름: **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **규정된 Type Name** DISTINCT
COLUMN_SIZE	INTEGER	열의 길이.
BUFFER_LENGTH	INTEGER	버퍼에서 열의 길이를 표시합니다.
DECIMAL_DIGITS	SMALLINT 널값 가능	숫자 열의 자리수를 나타냅니다. 오브젝트가 숫자가 아닌 경우 널값이 들어갑니다.
NUM_PREC_RADIX	SMALLINT 널값 가능	숫자 열의 radix를 나타냅니다. 오브젝트가 숫자가 아닌 경우 널값이 들어갑니다.

표 124. SQLCOLUMNS 뷰 (계속)

열 이름	자료 유형	설명
NULLABLE	SMALLINT	열이 널 값이 들어가는지 여부를 표시합니다. **0** 열은 널을 허용하지 않습니다. **1** 열은 널을 허용합니다.
REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
COLUMN_DEF	VARCHAR(2000) 널값 가능	열의 디폴트 값. 디폴트 값이 없을 경우 널값이 들어갑니다.
SQL_DATA_TYPE	SMALLINT	열의 SQL 자료 유형을 표시합니다.
SQL_DATETIME_SUB	SMALLINT 널값 가능	자료 유형의 날짜 시간 부속 유형 **1** DATE **2** TIME **3** TIMESTAMP 열이 자료시간 자료 유형이 아닌 경우 널값이 들어갑니다.
CHAR_OCTET_LENGTH	INTEGER 널값 가능	열의 문자 길이를 표시합니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
ORDINAL_POSITION	INTEGER	표에서 열의 원래 위치를 표시합니다.
IS_NULLABLE	VARCHAR(3)	열이 널 값이 들어가는지 여부를 표시합니다. **NO** 열은 널값 가능 여부가 없습니다. **YES** 열은 널값 가능 여부가 있습니다.

표 124. SQLCOLUMNS 뷰 (계속)

열 이름	자료 유형	설명	
JDBC_DATA_TYPE	SMALLINT	열의 JDBC 자료 유형을 표시합니다.	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		2005	CLOB
		1	GRAPHIC
		12	VARGRAPHIC
		1111	DBCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		1111	ROWID
		2001	DISTINCT
SCOPE_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.	
SCOPE_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.	
SCOPE_TABLE	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.	
SOURCE_DATA_TYPE	VARCHAR(128) 널값 가능	열의 자료 유형이 고유 유형인 경우 소스 자료 유형 자료 유형이 고유한 유형이 아닌 경우 널값이 들어갑니다.	
DBNAME	VARCHAR(8) 널값 가능	예약. 널 값이 들어갑니다.	

표 124. SQLCOLUMNS 뷰 (계속)

열 이름	자료 유형	설명
COLUMN_TEXT	VARCHAR(50) 널값 가능	열의 텍스트. 열에 열 텍스트가 없는 경우 널값이 들어갑니다.
PSEUDO_COLUMN	SMALLINT	이것이 ROWID 열인지 ID 열인지 여부를 나타냅니다. **1** 열은 ROWID 또는 ID 열이 아닙니다. **2** 열은 ROWID 또는 ID 열입니다.

SQLFOREIGNKEYS

SQLFOREIGNKEYS 뷰는 표의 모든 참조된 제한사항키에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 125. SQLFOREIGNKEYS 뷰

열 이름	자료 유형	설명
PKTABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
PKTABLE_SCHEM	VARCHAR(128)	상위 표가 들어 있는 SQL 스키마명.
PKTABLE_NAME	VARCHAR(128)	상위 표 이름
PKCOLUMN_NAME	VARCHAR(128)	상위 키 열 이름
FKTABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
FKTABLE_SCHEM	VARCHAR(128)	참조 제한조건의 종속 표가 들어 있는 SQL 스키마명
FKTABLE_NAME	VARCHAR(128)	참조 제한조건의 종속 표 이름
FKCOLUMN_NAME	VARCHAR(128)	종속 키 이름
KEY_SEQ	SMALLINT	키 내의 열 위치.
UPDATE_RULE	SMALLINT	갱신 규칙 **1** RESTRICT **3** NO ACTION
DELETE_RULE	SMALLINT	규칙 삭제: **0** CASCADE **1** RESTRICT **2** SET NULL **3** NO ACTION **4** SET DEFAULT
FK_NAME	VARCHAR(128)	참조된 제약 조건 명
PK_NAME	VARCHAR(128)	고유 제한사항 이름
DEFERRABILITY	SMALLINT	제한사항 검사가 지연될 수 있는지 여부를 표시합니다. 항상 7입니다.

SQLPRIMARYKEYS

SQLPRIMARYKEYS 뷰는 표의 모든 1차 제한사항 키에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 126. SQLPRIMARYKEYS 뷰

열 이름	자료 유형	설명
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	1차 키를 가진 표가 들어 있는 스키마명
TABLE_NAME	VARCHAR(128)	1차 키를 가진 표의 이름
COLUMN_NAME	VARCHAR(128)	1차 키 열 이름
KEY_SEQ	SMALLINT	키 내의 열 위치
PK_NAME	VARCHAR(128)	1차 키 제한사항 이름

SQLPROCEDURECOLS

SQLPROCEDURECOLS 뷰는 프로시듀어의 모든 매개변수에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 127. SQLPROCEDURECOLS 뷰

열 이름	자료 유형	설명
PROCEDURE_CAT	VARCHAR(128)	관계형 데이터베이스명
PROCEDURE_SCHEM	VARCHAR(128)	프로시듀어 인스턴스의 스키마(schema) 이름
PROCEDURE_NAME	VARCHAR(128)	프로시듀어 인스턴스의 이름
COLUMN_NAME	VARCHAR(128) 널값 가능	프로시듀어 매개변수의 이름 매개변수가 이름을 가지지 않는 경우 널값이 들어갑니다.
COLUMN_TYPE	SMALLINT	매개변수 유형: **1** IN **2** INOUT **4** OUT

표 127. SQLPROCEDURECOLS 뷰 (계속)

열 이름	자료 유형	설명	
DATA_TYPE	SMALLINT	매개변수의 자료 유형:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
		17	DISTINCT

SQLPROCEDURECOLS

표 127. SQLPROCEDURECOLS 뷰 (계속)

열 이름	자료 유형	설명
TYPE_NAME	VARCHAR(260)	매개변수의 자료 유형 이름: **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **규정된 Type Name** DISTINCT
COLUMN_SIZE	INTEGER	매개변수의 길이.
BUFFER_LENGTH	INTEGER	버퍼에서 매개변수의 길이를 표시합니다.
DECIMAL_DIGITS	SMALLINT 널값 가능	숫자 또는 날짜시간 자료의 스케일 매개변수가 십진, 숫자, 2진, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.

표 127. SQLPROCEDURECOLS 뷰 (계속)

열 이름	자료 유형	설명
NUM_PREC_RADIX	SMALLINT 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로 서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 매개변수가 숫자가 아닌 경우 널값이 들어갑니다.
NULLABLE	SMALLINT	매개변수의 널값 가능 여부를 표시합니다. **0** 매개변수에서 널값을 허용하지 않습니다. **1** 매개변수에서 널값을 허용합니다.
REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
COLUMN_DEF	VARCHAR(1) 널값 가능	열에 대한 디폴트 값. 디폴트 값이 없는 경우 널 값을 포함합니다.

SQLPROCEDURECOLS

표 127. SQLPROCEDURECOLS 뷰 (계속)

열 이름	자료 유형	설명
SQL_DATA_TYPE	SMALLINT	매개변수의 SQL 자료 유형: **-5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1** CHARACTER **-2** CHARACTER FOR BIT DATA **12** VARCHAR **-3** VARCHAR FOR BIT DATA **-99** CLOB **-95** GRAPHIC **-96** VARGRAPHIC **-350** DBCLOB **-98** BLOB **9** DATE **10** TIME **11** TIMESTAMP **70** DATALINK **-100** ROWID **17** DISTINCT
SQL_DATETIME_SUB	SMALLINT 널값 가능	매개변수의 날짜 시간 부속 유형: **1** DATE **2** TIME **3** TIMESTAMP 자료 유형이 날짜 시간 자료 유형이 아닌 경우 널값이 들어갑니다.
CHAR_OCTET_LENGTH	INTEGER 널값 가능	매개변수의 문자 길이를 표시합니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
ORDINAL_POSITION	INTEGER	좌에서 우로 순서를 매기는 매개변수 리스트 내의 매개변수의 숫자 위치

표 127. SQLPROCEDURECOLS 뷰 (계속)

열 이름	자료 유형	설명
IS_NULLABLE	VARCHAR(3)	매개변수의 널값 가능 여부를 표시합니다.
		NO 매개변수에서 널값을 허용하지 않습니다.
		YES 매개변수에서 널값을 허용합니다.
JDBC_DATA_TYPE	SMALLINT	매개변수의 JDBC 자료 유형:
		-5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		1 CHARACTER
		-2 CHARACTER FOR BIT DATA
		12 VARCHAR
		-3 VARCHAR FOR BIT DATA
		2005 CLOB
		1 GRAPHIC
		12 VARGRAPHIC
		1111 DBCLOB
		2004 BLOB
		91 DATE
		92 TIME
		93 TIMESTAMP
		70 DATALINK
		1111 ROWID
		2001 DISTINCT

SQLPROCEDURES

SQLPROCEDURES 뷰는 모든 프로시듀어에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 128. SQLPROCEDURES 뷰

열 이름	자료 유형	설명
PROCEDURE_CAT	VARCHAR(128)	관계형 데이터베이스명
PROCEDURE_SCHEM	VARCHAR(128)	프로시듀어 인스턴스의 스키마명
PROCEDURE_NAME	VARCHAR(128)	프로시듀어 이름
NUM_INPUT_PARAMS	SMALLINT	입력 매개변수의 수를 식별합니다. 0은 입력 매개변수가 없음을 의미합니다.
NUM_OUTPUT_PARAMS	SMALLINT	출력 매개변수의 수를 식별합니다. 0은 출력 매개변수가 없음을 의미합니다.
NUM_RESULT_SETS	SMALLINT	리턴된 결과 세트의 최대수를 식별합니다. 0은 결과 세트가 없음을 의미합니다.
REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
PROCEDURE_TYPE	SMALLINT	예약. 0을 포함합니다.
NUM_INOUT_PARAMS	SMALLINT	입/출력(I/O) 매개변수의 수를 식별합니다. 0은 입/출력(I/O) 매개변수가 없음을 의미합니다.

SQLSCHEMAS

SQLSCHEMAS 뷰는 모든 스키마에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 129. SQLSCHEMAS 뷰

열 이름	자료 유형	설명
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	스키마 이름
TABLE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TABLE_TYPE	VARCHAR(128)	예약. 널 값이 들어갑니다. 널값 가능
REMARKS	VARCHAR(2000)	예약. 널 값이 들어갑니다. 널값 가능
DBNAME	VARCHAR(8)	예약. 널 값이 들어갑니다. 널값 가능
SCHEMA_TEXT	VARCHAR(50)	스키마를 설명하는 문자 스트링. 텍스트가 없을 경우 널값이 들어갑니다.

SQLSPECIALCOLUMNS

SQLSPECIALCOLUMNS 뷰에는 1차 키의 모든 열에 대한 한 행 또는 표의 행을 나타낼 수 있는 고유 색인이 포함됩니다. 다음 표에서는 뷰의 열을 설명합니다.

표 130. SQLSPECIALCOLUMNS 뷰

열 이름	자료 유형	설명	
SCOPE	SMALLINT	예약. 0을 포함합니다.	
COLUMN_NAME	VARCHAR(128)	열 이름	
DATA_TYPE	SMALLINT	열의 자료 유형:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
		17	DISTINCT
TYPE_NAME	VARCHAR(260)	열의 자료 유형 이름	
COLUMN_SIZE	INTEGER	열의 길이	
BUFFER_LENGTH	INTEGER	버퍼에서 열의 길이를 표시합니다.	
DECIMAL_DIGITS	SMALLINT 널값 가능	숫자 열의 자리수를 나타냅니다. 열이 숫자가 아닌 경우 널값이 들어갑니다.	

표 130. SQLSPECIALCOLUMNS 뷰 (계속)

열 이름	자료 유형	설명
PSEUDO_COLUMN	SMALLINT	ROWID 열인지 ID 열인지 여부를 나타냅니다. **1** 열은 ROWID 또는 ID 열이 아닙니다. **2** 열은 ROWID 또는 ID 열입니다.
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	표가 들어 있는 SQL 스키마명
TABLE_NAME	VARCHAR(128)	표 이름
NULLABLE	SMALLINT	열이 널 값이 들어가는지 여부를 표시합니다. **0** 열은 널값 가능 여부가 없습니다. **1** 열은 널값 가능 여부가 있습니다.
JDBC_DATA_TYPE	SMALLINT	열의 JDBC 자료 유형을 표시합니다. **-5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1** CHARACTER **-2** CHARACTER FOR BIT DATA **12** VARCHAR **-3** VARCHAR FOR BIT DATA **2005** CLOB **1** GRAPHIC **12** VARGRAPHIC **1111** DBCLOB **2004** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **1111** ROWID **2001** DISTINCT

SQLSTATISTICS

SQLSTATISTICS 뷰에는 표에 대한 통계 정보가 포함됩니다. 다음 표에서는 뷰의 열을 설명합니다.

표 131. SQLSTATISTICS 뷰

열 이름	자료 유형	설명
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	표의 SQL 스키마 이름
TABLE_NAME	VARCHAR(128)	표 이름
NON_UNIQUE	SMALLINT 널값 가능	색인으로 인해 표의 중복 키가 없어지는지 여부를 나타냅니다. TYPE 이 0인 경우 널 값이 들어갑니다.
INDEX_QUALIFIER	VARCHAR(128) 널값 가능	색인의 스키마명 TYPE 이 0인 경우 널 값이 들어갑니다.
INDEX_NAME	VARCHAR(128) 널값 가능	색인명. TYPE 이 0인 경우 널 값이 들어갑니다.
TYPE	SMALLINT	리턴되는 정보의 유형을 나타냅니다. **0** 표에서 행 수. **3** 표에서 색인
ORDINAL_POSITION	SMALLINT 널값 가능	색인에서 키의 원래 위치를 표시합니다. TYPE 이 0인 경우 널 값이 들어갑니다.
COLUMN_NAME	VARCHAR(128) 널값 가능	색인에서 키에 대한 열 이름. TYPE 이 0인 경우 널 값이 들어갑니다.
ASC_OR_DESC	CHAR(1) 널값 가능	키 내의 열 순서 **A** 오름차순 **D** 내림차순 TYPE 이 0인 경우 널 값이 들어갑니다.
CARDINALITY	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
PAGES	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
FILTER_CONDITION	VARCHAR(128) 널값 가능	색인이 선택/생략 색인인지 여부를 나타냅니다. **empty-string** 선택/생략 색인입니다. TYPE 이 0이거나 선택/생략 색인이 아닌 경우 널 값이 들어갑니다.

SQLTABLEPRIVILEGES

SQLTABLEPRIVILEGES 뷰는 표의 모든 권한 부여에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 132. SQLTABLEPRIVILEGES 뷰

열 이름	자료 유형	설명
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	표의 SQL 스키마 이름
TABLE_NAME	VARCHAR(128)	표 이름
GRANTOR	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
GRANTEE	VARCHAR(128)	특권이 부여된 사용자 프로파일
PRIVILEGE	VARCHAR(10)	부여된 특권: **ALTER** 표를 변경할 특권. **DELETE** 권한은 표에서 행을 삭제합니다. **INDEX** 권한은 표에서 색인을 작성합니다. **INSERT** 권한은 표로 행을 삽입합니다. **REFERENCES** 권한은 참조된 제한 사항에서 표를 참조합니다. **SELECT** 권한은 표에서 행을 선택합니다. **UPDATE** 권한은 표를 갱신 선택합니다.
IS_GRANTABLE	VARCHAR(3)	다른 사용자에게 특권을 부여할 수 있는지 여부를 나타냅니다. **NO** 특권을 부여할 수 없습니다. **YES** 특권을 부여할 수 있습니다.
DBNAME	VARCHAR(8) 널값 가능	예약. 널 값이 들어갑니다.

SQLTABLES

SQLTABLES 뷰는 모든 표, 뷰, 및 별명에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 133. SQLTABLES 뷰

열 이름	자료 유형	설명
TABLE_CAT	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEM	VARCHAR(128)	표가 들어 있는 스키마명.
TABLE_NAME	VARCHAR(128)	표 이름
TABLE_TYPE	VARCHAR(10)	표의 유형을 표시합니다. **ALIAS** 표는 별명입니다. **TABLE** 표는 SQL 표 또는 실제 파일(PF)입니다. **VIEW** 표는 SQL 뷰 또는 논리 파일입니다.
REMARKS	VARCHAR(128) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 긴 주석이 없는 경우 널값이 들어갑니다.
TYPE_CAT	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TYPE_SCHEM	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TYPE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SELF_REF_COL_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
REF_GENERATION	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
DBNAME	VARCHAR(8) 널값 가능	예약. 널 값이 들어갑니다.
TABLE_TEXT	VARCHAR(50)	문자 스트링은 LABEL문으로 제공됩니다.

SQLTYPEINFO

SQLTYPEINFO 뷰는 내장 자료 유형에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 134. SQLTYPEINFO 뷰

열 이름	자료 유형	설명	
TYPE_NAME	VARCHAR(128)	내장 자료 유형 이름:	
		BIGINT	BIGINT
		INTeger	INTEGER
		SMALLINT	SMALLINT
		DECIMAL	DECIMAL
		NUMERIC	NUMERIC
		FLOAT	DOUBLE PRECISION
		REAL	REAL
		CHARacter	CHARACTER
		CHARacter FOR BIT DATA	CHARACTER FOR BIT DATA
		VARCHAR	VARCHAR
		VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
		CLOB	CLOB
		GRAPHIC	GRAPHIC
		VARGRAPHIC	VARGRAPHIC
		DBCLOB	DBCLOB
		BLOB	BLOB
		DATE	DATE
		TIME	TIME
		TIMESTAMP	TIMESTAMP
		DATALINK	DATALINK
		ROWID	ROWID

SQLTYPEINFO

표 134. SQLTYPEINFO 뷰 (계속)

열 이름	자료 유형	설명	
DATA_TYPE	SMALLINT	열의 자료 유형:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
COLUMN_SIZE	INTEGER	자료 유형의 최대 길이.	
LITERAL_PREFIX	VARCHAR(128) 널값 가능	스트링 리터럴에 대한 접두부를 표시합니다. 자료 유형이 스트링이 아닌 경우 널값이 들어갑니다.	
LITERAL_SUFFIX	VARCHAR(128) 널값 가능	스트링 리터럴에 대한 접미부를 표시합니다. 자료 유형이 스트링이 아닌 경우 널값이 들어갑니다.	

표 134. SQLTYPEINFO 뷰 (계속)

열 이름	자료 유형	설명
CREATE_PARAMS	VARCHAR(128) 널값 가능	자료 유형으로 지원된 매개변수를 표시합니다. **LENGTH** 매개변수는 길이입니다. 모든 스트링 자료 유형 및 DATALINK의 경우 리턴됩니다. **PRECISION, SCALE** 매개변수는 정밀도와 스케일을 포함합니다. DECIMAL 및 NUMERIC 자료 유형의 경우 리턴됩니다. 모든 다른 자료 유형에 대한 널 값이 들어갑니다.
NULLABLE	SMALLINT	자료 유형의 널값 가능 여부를 표시합니다. **0** 자료 유형은 널을 허용하지 않습니다. **1** 자료 유형은 널을 허용합니다.
CASE_SENSITIVE	SMALLINT	자료 유형의 대소문자 구분 여부를 표시합니다. **0** 자료 유형은 대소문자 구분이 없습니다. **1** 자료 유형은 대소문자 구분이 있습니다.
SEARCHABLE	SMALLINT	자료 유형을 술부에 사용할 수 있는지 여부를 나타냅니다. **0** 자료 유형을 술부에 사용할 수 없습니다. **2** LIKE 술부를 제외한 모든 술부에 자료 유형을 사용할 수 있습니다. **3** LIKE 술부를 포함한 모든 술부에 자료 유형을 사용할 수 있습니다.
UNSIGNED_ATTRIBUTE	SMALLINT 널값 가능	숫자 자료 유형에 부호가 있는지 여부를 나타냅니다. **0** 자료 유형이 부호화되었습니다. **1** 자료 유형이 부호화되지 않았습니다. 자료 유형이 숫자가 아닌 경우 널값이 들어갑니다.
FIXED_PREC_SCALE	SMALLINT	자료 유형에 고정 정밀도 및 스케일이 있는지 여부를 나타냅니다. **0** 자료 유형에 고정 정밀도 및 스케일이 없습니다. **1** 자료 유형에 고정 정밀도 및 스케일이 있습니다.
AUTO_UNIQUE_VALUE	SMALLINT 널값 가능	숫자 자료 유형이 자동 증가되는지 여부를 나타냅니다. **0** 자료 유형은 자동 증가되지 않습니다. **1** 자료 유형이 자동 증가됩니다. 자료 유형이 숫자가 아닌 경우 널값이 들어갑니다.
LOCAL_TYPE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
MINIMUM_SCALE	SMALLINT 널값 가능	숫자 자료 유형의 최소 스케일을 표시합니다. 자료 유형이 숫자가 아닌 경우 널값이 들어갑니다.

SQLTYPEINFO

표 134. SQLTYPEINFO 뷰 (계속)

열 이름	자료 유형	설명
MAXIMUM_SCALE	SMALLINT 널값 가능	숫자 자료 유형의 최대 스케일을 표시합니다. 자료 유형이 숫자가 아닌 경우 널값이 들어갑니다.
SQL_DATA_TYPE	SMALLINT	자료 유형의 SQL 자료 유형을 표시합니다. **-5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1** CHARACTER **-2** CHARACTER FOR BIT DATA **12** VARCHAR **-3** VARCHAR FOR BIT DATA **-99** CLOB **-95** GRAPHIC **-96** VARGRAPHIC **-350** DBCLOB **-98** BLOB **9** DATE **10** TIME **11** TIMESTAMP **70** DATALINK **-100** ROWID
SQL_DATETIME_SUB	SMALLINT 널값 가능	자료 유형의 날짜 시간 부속 유형: **1** DATE **2** TIME **3** TIMESTAMP 자료 유형이 날짜 시간 자료 유형이 아닌 경우 널값이 들어갑니다.

표 134. SQLTYPEINFO 뷰 (계속)

열 이름	자료 유형	설명
NUM_PREC_RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로 서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 매개변수가 숫자가 아닌 경우 널값이 들어갑니다.
INTERVAL_PRECISION	SMALLINT 널값 가능	예약. 널 값이 들어갑니다.
JDBC_DATA_TYPE	SMALLINT	자료 유형의 JDBC 자료 유형 값: **-5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1** CHARACTER **-2** CHARACTER FOR BIT DATA **12** VARCHAR **-3** VARCHAR FOR BIT DATA **2005** CLOB **1** GRAPHIC **12** VARGRAPHIC **1111** DBCLOB **2004** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **1111** ROWID

SQLUDTS

SQLUDTS 뷰는 보든 고유한 유형에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 135. SQLUDTS 뷰

열 이름	자료 유형	설명
TYPE_CAT	VARCHAR(128)	관계형 데이터베이스명
TYPE_SCHEM	VARCHAR(128)	사용자 정의 유형이 들어 있는 스키마명
TYPE_NAME	VARCHAR(128)	사용자 정의 유형 이름
CLASS_NAME	VARCHAR(20)	사용자 정의 유형에 대한 Java 클래스명
		java.math.BigInteger BIGINT
		java.lang.Integer INTEGER
		java.lang.Short SMALLINT
		java.math.BigDecimal DECIMAL
		java.sql.BigDecimal NUMERIC
		java.lang.Double DOUBLE PRECISION
		java.lang.Float REAL
		java.lang.String CHARACTER
		byte[] CHARACTER FOR BIT DATA
		java.lang.String VARCHAR
		byte[] VARCHAR FOR BIT DATA
		java.sql.Clob CLOB
		java.lang.String GRAPHIC
		java.lang.String VARGRAPHIC
		java.sql.Clob DBCLOB
		java.sql.Blob BLOB
		java.sql.Date DATE
		java.sql.Time TIME
		java.sql.Timestamp TIMESTAMP
		java.net.URL DATALINK
		byte[] ROWID
DATA_TYPE	SMALLINT	예약. 2001을 포함합니다.

표 135. SQLUDTS 뷰 (계속)

열 이름	자료 유형	설명	
BASE_TYPE	SMALLINT	사용자 정의 자료 유형의 소스 자료 유형:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		2005	CLOB
		1	GRAPHIC
		12	VARGRAPHIC
		1111	DBCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		1111	ROWID
REMARKS	VARCHAR(2000) 널값 가능	문자 스트링은 COMMENT문으로 지원됩니다. 주석이 없을 경우 널값이 들어갑니다.	

ANS 및 ISO 카탈로그 뷰

일부 ANS 및 ISO 카탈로그 뷰에는 두 가지 버전이 있습니다. 기술된 버전은 ANS 및 ISO 뷰의 일반 세트입니다. 두 번째 뷰 세트 이름은 18자 이하로 제한되며 이 책에는 뷰 이름만 기술됩니다.

ANS 및 ISO 카탈로그는 QSYS2 라이브러리의 다음 표를 포함합니다.

View Name	Shorter View Name	설명
1048 페이지의 『SQL_FEATURES』		데이터베이스 관리자에 의해 지원된 피처에 대한 정보
1049 페이지의 『SQL_LANGUAGES』	SQL_LANGUAGES_S	지원 언어에 대한 정보
1051 페이지의 『SQL_SIZING』		데이터베이스 관리자에 의해 지원된 제한에 대한 정보

ANS 및 ISO 카탈로그는 SYSIBM 라이브러리의 다음 뷰와 표를 포함합니다.

View Name	Shorter View Name	설명
1027 페이지의 『CHARACTER_SETS』	CHARACTER_SETS_S	지원되는 CCSID에 대한 정보
1028 페이지의 『CHECK_CONSTRAINTS』		검사 제약 조건에 대한 정보
1029 페이지의 『COLUMNS』	COLUMNS_S	열에 대한 정보
1033 페이지의 『INFORMATION_SCHEMA_CATALOG_NAME』	CATALOG_NAME	관계형 데이터베이스에 대한 정보
1034 페이지의 『PARAMETERS』	PARAMETERS_S	프로시듀어 매개변수에 대한 정보
1038 페이지의 『REFERENTIAL_CONSTRAINTS』	REF_CONSTRAINTS	참조 제약 조건에 대한 정보
1039 페이지의 『ROUTINES』	ROUTINES_S	루틴에 대한 정보
1047 페이지의 『SCHEMATA』	SCHEMATA_S	스키마에 대한 통계 정보
1052 페이지의 『TABLE_CONSTRAINTS』		제한사항에 대한 정보
1053 페이지의 『TABLES』	TABLES_S	표에 대한 정보
1054 페이지의 『USER_DEFINED_TYPES』	UDT_S	고유한 유형에 대한 정보
1058 페이지의 『VIEWS』		뷰에 대한 정보

CHARACTER_SETS

CHARACTER_SETS 뷰는 지원된 모든 CCSID에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 136. CHARACTER_SETS 뷰

열 이름	자료 유형	설명
CHARACTER_SET_CATALOG	VARCHAR(128)	관계형 데이터베이스명
CHARACTER_SET_SCHEMA	VARCHAR(128)	문자 세트의 스키마 이름. 'SYSIBM'을 포함합니다.
CHARACTER_SET_NAME	VARCHAR(128)	문자 세트 이름
FORM_OF_USE	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
NUMBER_OF_CHARACTERS	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
DEFAULT_COLLATE_CATALOG	VARCHAR(128)	예약. 관계형 데이터베이스명이 들어갑니다.
DEFAULT_COLLATE_SCHEMA	VARCHAR(128)	예약. SYSIBM을 포함합니다.
DEFAULT_COLLATE_NAME	VARCHAR(128)	예약. IBMDEFAULT를 포함합니다.

CHECK_CONSTRAINTS

CHECK_CONSTRAINTS 뷰는 모든 검사 제한 사항에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 137. CHECK_CONSTRAINTS 뷰

열 이름	자료 유형	설명
CONSTRAINT_CATALOG	VARCHAR(128)	관계형 데이터베이스명
CONSTRAINT_SCHEMA	VARCHAR(128)	제한사항이 들어 있는 스키마명
CONSTRAINT_NAME	VARCHAR(128)	제약 조건 명
CHECK_CLAUSE	VARCHAR(2000) 널값 가능	검사 제한사항 절의 텍스트 검사 절이 열에 잘리지 않고 포함될 수 없는 경우 널값을 포함합니다.

COLUMNS

COLUMNS 뷰는 모든 열에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 138. COLUMNS 뷰

열 이름	자료 유형	설명
TABLE_CATALOG	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEMA	VARCHAR(128)	표 또는 뷰가 들어 있는 SQL 스키마명
TABLE_NAME	VARCHAR(128)	열이 들어 있는 표 또는 뷰의 이름
COLUMN_NAME	VARCHAR(128)	열의 이름
ORDINAL_POSITION	INTEGER	좌에서 우로 순서가 지정된 표 또는 뷰의 열에 대한 숫자 위치
COLUMN_DEFAULT	VARCHAR(2000) 널값 가능	만일 하나라도 존재할 경우 열의 디폴트 값. 만일 열의 디폴트 값을 절단시켜야만 볼 수 있으면 열의 값은 'TRUNCATED' 스트링이 됩니다. 디폴트 값은 문자 양식으로 저장됩니다. 다음과 같은 특수 값도 있습니다. **CURRENT_DATE** 디폴트 값은 현재 날짜입니다. **CURRENT_TIME** 디폴트 값은 현재 시간입니다. **CURRENT_TIMESTAMP** 디폴트 값은 현재 시간소인입니다. **NULL** 디폴트 값은 널값입니다. **USER** 디폴트 값은 현재의 작업 사용자입니다. 열에 디폴트 값이 없을 경우 널값이 들어갑니다. 예를 들어, 열이 IDENTITY 속성을 가지거나 행 ID인 경우입니다.
IS_NULLABLE	VARCHAR(3)	열이 널 값이 들어가는지 여부를 표시합니다. **NO** 열은 널 값을 가질 수 없습니다. **YES** 열은 널 값을 가질 수 있습니다.

COLUMNS

표 138. COLUMNS 뷰 (계속)

열 이름	자료 유형	설명
DATA_TYPE	VARCHAR(128)	열 유형 **BIGINT** 큰 수 **INTEGER** 큰 수 **SMALLINT** 작은 수 **DECIMAL** 압축 십진수 **NUMERIC** 존(zone) 십진수 **DOUBLE PRECISION** 배정밀도 부동 소수점 **REAL** 단정밀도 부동 소수점 **CHARACTER** 고정 길이 문자 스트링 **CHARACTER VARYING** 가변 길이의 문자 스트링 **CHARACTER LARGE OBJECT** 문자 LOB(Large Object) 스트링 **GRAPHIC** 고정 길이 그래픽 스트링 **GRAPHIC VARYING** 가변 길이의 그래픽 스트링 **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2바이트 문자 LOB(Large Object) 스트링 **BINARY LARGE OBJECT** 2진 LOB(Large Object) 스트링 **DATE** 날짜 **TIME** 시간 **TIMESTAMP** 시간소인 **DATALINK** 자료 링크 **ROWID** 행 ID **USER-DEFINED** 고유한 유형
CHARACTER_MAXIMUM_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 스트링 최대 길이 열이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_OCTET_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 바이트 수. 열이 스트링이 아닌 경우 널값이 들어갑니다.

표 138. COLUMNS 뷰 (계속)

열 이름	자료 유형	설명
NUMERIC_PRECISION	INTEGER 널값 가능	전체 숫자 열 정밀도 **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 열이 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진수 또는 십진수의 수로 명시되었는지 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 열이 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_SCALE	INTEGER 널값 가능	숫자 자료의 스케일 열이 십진, 숫자 또는 2진이 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	INTEGER 널값 가능	날짜, 시간 또는 시간소인의 분수 부분. **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 열이 날짜, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.
INTERVAL_TYPE	VARCHAR(128) 널값 가능	예약 널 값이 들어갑니다.
INTERVAL_PRECISION	INTEGER 널값 가능	예약 널 값이 들어갑니다.
CHARACTER_SET_CATALOG	VARCHAR(128) 널값 가능	관계형 데이터베이스명 열이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_SCHEMA	VARCHAR(128) 널값 가능	문자 세트의 스키마 이름. SYSIBM을 포함합니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_NAME	VARCHAR(128) 널값 가능	문자 세트 이름. 열이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_CATALOG	VARCHAR(128) 널값 가능	관계형 데이터베이스명 열이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_SCHEMA	VARCHAR(128) 널값 가능	배열의 스키마. SYSIBM을 포함합니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.

COLUMNS

표 138. COLUMNS 뷰 (계속)

열 이름	자료 유형	설명
COLLATION_NAME	VARCHAR(128) 널값 가능	배열 이름. IBMBINARY를 포함합니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
DOMAIN_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
DOMAIN_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
DOMAIN_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
UDT_CATALOG	VARCHAR(128) 널값 가능	이것이 고유한 유형인 경우 관계형 데이터베이스명. 이것이 고유한 유형이 아닌 경우 널 값이 들어갑니다.
UDT_SCHEMA	VARCHAR(128) 널값 가능	이것이 고유한 유형인 경우 스키마명. 이것이 고유한 유형이 아닌 경우 널 값이 들어갑니다.
UDT_NAME	VARCHAR(128) 널값 가능	고유한 유형 이름. 이것이 고유한 유형이 아닌 경우 널 값이 들어갑니다.
SCOPE_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SCOPE_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SCOPE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
MAXIMUM_CARDINALITY	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
DTD_IDENTIFIER	VARCHAR(128) 널값 가능	열의 고유 내부 ID.
IS_SELF_REFERENCING	VARCHAR(3)	예약. ‘NO’를 포함합니다.

INFORMATION_SCHEMA_CATALOG_NAME

INFORMATION_SCHEMA_CATALOG_NAME 뷰는 관계형 데이터베이스에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 139. INFORMATION_SCHEMA_CATALOG_NAME 뷰

열 이름	자료 유형	설명
CATALOG_NAME	VARCHAR(128)	관계형 데이터베이스명

PARAMETERS

PARAMETERS 뷰는 관계형 데이터베이스에서 루틴의 각 매개변수에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 140. PARAMETERS 뷰

열 이름	자료 유형	설명
SPECIFIC_CATALOG	VARCHAR(128)	관계형 데이터베이스명
SPECIFIC_SCHEMA	VARCHAR(128)	루틴 인스턴스의 스키마(schema) 이름
SPECIFIC_NAME	VARCHAR(128)	루틴 인스턴스의 고유 이름
ORDINAL_POSITION	INTEGER	좌에서 우로 순서를 매기는 매개변수 리스트 내의 매개변수의 숫자 위치
PARAMETER_MODE	VARCHAR(5)	parameter-type **IN** 입력 매개변수입니다. **OUT** 출력 매개변수입니다. **INOUT** 입/출력(I/O) 매개변수입니다.
IS_RESULT	VARCHAR(3)	예약. 'NO'를 포함합니다.
AS_LOCATOR	VARCHAR(3)	매개변수가 로케이터로서 명시되었는지의 여부를 표시합니다. **NO** 매개변수가 로케이터로 명시되지 않았습니다. **YES** 매개변수가 로케이터로 명시되었습니다.
PARAMETER_NAME	VARCHAR(128) 널값 가능	매개변수명 매개변수가 이름을 가지지 않는 경우 널값이 들어갑니다.
FROM_SQL_SPECIFIC_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
FROM_SQL_SPECIFIC_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
FROM_SQL_SPECIFIC_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TO_SQL_SPECIFIC_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TO_SQL_SPECIFIC_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TO_SQL_SPECIFIC_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.

표 140. PARAMETERS 뷰 (계속)

열 이름	자료 유형	설명
DATA_TYPE	VARCHAR(128) 널값 가능	매개변수 유형: **BIGINT** 큰 수 **INTEGER** 큰 수 **SMALLINT** 작은 수 **DECIMAL** 압축 십진수 **NUMERIC** 존(zone) 십진수 **DOUBLE PRECISION** 부동 소수점. 배정밀도 **REAL** 부동 소수점. REAL **CHARACTER** 고정 길이 문자 스트링 **CHARACTER VARYING** 가변 길이의 문자 스트링 **CHARACTER LARGE OBJECT** 문자 LOB(Large Object) 스트링 **GRAPHIC** 고정 길이 그래픽 스트링 **GRAPHIC VARYING** 가변 길이의 그래픽 스트링 **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2바이트 문자 LOB(Large Object) 스트링 **BINARY LARGE OBJECT** 2진 LOB(Large Object) 스트링 **DATE** 날짜 **TIME** 시간 **TIMESTAMP** 시간소인 **DATALINK** 자료 링크 **ROWID** 행 ID **USER-DEFINED** 고유한 유형
CHARACTER_MAXIMUM_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 스트링 최대 길이 매개변수가 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_OCTET_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형에 대한 바이트 수 매개변수가 스트링이 아닌 경우 널값이 들어갑니다.

PARAMETERS

표 140. PARAMETERS 뷰 (계속)

열 이름	자료 유형	설명
CHARACTER_SET_CATALOG	VARCHAR(128) 널값 가능	관계형 데이터베이스명 열이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_SCHEMA	VARCHAR(128) 널값 가능	문자 세트의 스키마 이름. 'SYSIBM'을 포함합니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_NAME	VARCHAR(128) 널값 가능	문자 세트 이름. 열이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_CATALOG	VARCHAR(128) 널값 가능	관계형 데이터베이스명 열이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_SCHEMA	VARCHAR(128) 널값 가능	배열의 스키마. SYSIBM은 리턴됩니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_NAME	VARCHAR(128) 널값 가능	배열 이름. IBMBINARY은 리턴됩니다. 열이 스트링이 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION	INTEGER 널값 가능	전체 숫자 매개변수의 정밀도 **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 매개변수가 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 매개변수가 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_SCALE	INTEGER 널값 가능	숫자 자료의 스케일 십진수, 숫자 또는 2진수 매개변수가 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	INTEGER 널값 가능	날짜, 시간 또는 시간소인의 분수 부분. **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 매개변수가 날짜, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.

표 140. PARAMETERS 뷰 (계속)

열 이름	자료 유형	설명
INTERVAL_TYPE	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
INTERVAL_PRECISION	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
UDT_CATALOG	VARCHAR(128) 널값 가능	이것이 고유한 유형인 경우 관계형 데이터베이스명. 이것이 고유한 유형이 아닌 경우 널 값이 들어갑니다.
UDT_SCHEMA	VARCHAR(128) 널값 가능	이것이 고유한 유형인 경우 스키마명. 이것이 고유한 유형이 아닌 경우 널 값이 들어갑니다.
UDT_NAME	VARCHAR(128) 널값 가능	고유한 유형 이름. 이것이 고유한 유형이 아닌 경우 널 값이 들어갑니다.
SCOPE_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SCOPE_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SCOPE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
MAXIMUM_CARDINALITY	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
DTD_IDENTIFIER	VARCHAR(128) 널값 가능	매개변수의 고유 내부 ID.

REFERENTIAL_CONSTRAINTS

REFERENTIAL_CONSTRAINTS 뷰는 각 참조된 제한사항에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 141. REFERENTIAL_CONSTRAINTS 뷰

열 이름	자료 유형	설명
CONSTRAINT_CATALOG	VARCHAR(128)	관계형 데이터베이스명
CONSTRAINT_SCHEMA	VARCHAR(128)	제한사항이 들어 있는 스키마명.
CONSTRAINT_NAME	VARCHAR(128)	제약 조건 명
UNIQUE_CONSTRAINT_CATALOG	VARCHAR(128)	참조 제한사항에 의해 참조되는 고유한 제한사항이 들어 있는 관계형 데이터베이스명
UNIQUE_CONSTRAINT_SCHEMA	VARCHAR(128)	참조 제한사항에 의해 참조되는 고유한 제한사항이 들어 있는 SQL 스키마명
UNIQUE_CONSTRAINT_NAME	VARCHAR(128)	참조 제약 조건에 의해 참조되는 고유한 제약 조건 명
MATCH_OPTION	VARCHAR(7)	예약. 'NONE'을 포함합니다.
UPDATE_RULE	VARCHAR(11)	갱신 규칙 • NO ACTION • RESTRICT
DELETE_RULE	VARCHAR(11)	삭제 규칙 • NO ACTION • CASCADE • SET NULL • SET DEFAULT • RESTRICT

ROUTINES

ROUTINES 뷰는 각 루틴에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 142. ROUTINES 뷰

열 이름	자료 유형	설명
SPECIFIC_CATALOG	VARCHAR(128)	관계형 데이터베이스명
SPECIFIC_SCHEMA	VARCHAR(128)	루틴 인스턴스의 스키마(schema) 이름
SPECIFIC_NAME	VARCHAR(128)	루틴의 고유 이름
ROUTINE_CATALOG	VARCHAR(128)	관계형 데이터베이스명
ROUTINE_SCHEMA	VARCHAR(128)	루틴이 들어 있는 SQL 스키마명.
ROUTINE_NAME	VARCHAR(128)	루틴 명
ROUTINE_TYPE	VARCHAR(15)	루틴 유형 **PROCEDURE** 프로시듀어입니다. **FUNCTION** 함수입니다. **INSTANCE METHOD** 고유 유형에 대해 작성된 내장 자료 유형 함수입니다.
MODULE_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
MODULE_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
MODULE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
UDT_CATALOG	VARCHAR(128) 널값 가능	관계형 데이터베이스명 이것이 INSTANCE METHOD이 아닌 경우 널 값이 들어갑니다.
UDT_SCHEMA	VARCHAR(128) 널값 가능	이 함수와 관련된 고유한 유형이 들어 있는 SQL 스키마명. 이것이 INSTANCE METHOD이 아닌 경우 널 값이 들어갑니다.
UDT_NAME	VARCHAR(128) 널값 가능	이 함수와 관련된 고유 유형 이름. 이것이 INSTANCE METHOD이 아닌 경우 널 값이 들어갑니다.

ROUTINES

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
DATA_TYPE	VARCHAR(128) 널값 가능	함수의 결과 유형: **BIGINT** 큰 수 **INTEGER** 큰 수 **SMALLINT** 작은 수 **DECIMAL** 압축 십진수 **NUMERIC** 존(zone) 십진수 **DOUBLE PRECISION** 부동 소수점. 배정밀도 **REAL** 부동 소수점. REAL **CHARACTER** 고정 길이 문자 스트링 **CHARACTER VARYING** 가변 길이의 문자 스트링 **CHARACTER LARGE OBJECT** 문자 LOB(Large Object) 스트링 **GRAPHIC** 고정 길이 그래픽 스트링 **GRAPHIC VARYING** 가변 길이의 그래픽 스트링 **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2바이트 문자 LOB(Large Object) 스트링 **BINARY LARGE OBJECT** 2진 LOB(Large Object) 스트링 **DATE** 날짜 **TIME** 시간 **TIMESTAMP** 시간소인 **DATALINK** 자료 링크 **ROWID** 행 ID **USER-DEFINED** 고유한 유형 스칼라 함수가 아닌 경우 널값이 들어갑니다.
CHARACTER_MAXIMUM_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 함수 결과 스트링 최대 길이 스칼라 함수가 아닌 경우 또는 매개변수가 스트링이 아닌 경우 널값이 들어갑니다.

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
CHARACTER_OCTET_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 함수 결과 스트링 바이트 수 스칼라 함수가 아닌 경우 또는 매개변수가 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_CATALOG	VARCHAR(128) 널값 가능	함수 결과의 관계형 데이터베이스명. 스칼라 함수가 아닌 경우 또는 결과가 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_SCHEMA	VARCHAR(128) 널값 가능	함수 결과의 문자 세트 스키마명. 'SYSIBM'을 포함합니다. 스칼라 함수가 아닌 경우 또는 결과가 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_NAME	VARCHAR(128) 널값 가능	함수 결과의 문자 세트명. 스칼라 함수가 아닌 경우 또는 결과가 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_CATALOG	VARCHAR(128) 널값 가능	함수 결과의 관계형 데이터베이스명. 스칼라 함수가 아닌 경우 또는 결과가 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_SCHEMA	VARCHAR(128) 널값 가능	함수 결과의 배열 스키마. SYSIBM은 리턴됩니다. 스칼라 함수가 아닌 경우 또는 결과가 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_NAME	VARCHAR(128) 널값 가능	함수 결과의 배열명. IBMBINARY은 리턴됩니다. 스칼라 함수가 아닌 경우 또는 결과가 스트링이 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION	INTEGER 널값 가능	함수 결과의 정밀도. **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 스칼라 함수가 아닌 경우 또는 결과가 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 스칼라 함수가 아닌 경우 또는 결과가 숫자가 아닌 경우 널값이 들어갑니다.

ROUTINES

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
NUMERIC_SCALE	INTEGER 널값 가능	함수의 숫자 결과 스케일. 스칼라 함수가 아닌 경우 또는 결과가 숫자가 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	INTEGER 널값 가능	함수 결과의 날짜, 시간 또는 시간소인 분수 부분. **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 스칼라 함수가 아닌 경우 또는 결과가 시간소인이 아닌 경우 널값이 들어갑니다.
INTERVAL_TYPE	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
INTERVAL_PRECISION	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
TYPE_UDT_CATALOG	VARCHAR(128) 널값 가능	함수 결과가 고유한 유형인 경우의 관계형 데이터베이스명. 스칼라 함수가 아닌 경우 또는 결과가 고유한 유형이 아닌 경우 널값이 들어갑니다.
TYPE_UDT_SCHEMA	VARCHAR(128) 널값 가능	함수 결과가 고유한 유형인 경우의 스키마명. 스칼라 함수가 아닌 경우 또는 결과가 고유한 유형이 아닌 경우 널값이 들어갑니다.
TYPE_UDT_NAME	VARCHAR(128) 널값 가능	함수 결과가 고유한 유형인 경우의 고유한 유형 이름. 스칼라 함수가 아닌 경우 또는 결과가 고유한 유형이 아닌 경우 널값이 들어갑니다.
SCOPE_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SCOPE_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
SCOPE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
MAXIMUM_CARDINALITY	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
DTD_IDENTIFIER	VARCHAR(128) 널값 가능	함수의 결과에 대한 하나의 고유 내부 식별자
ROUTINE_BODY	VARCHAR(8)	루틴 본문의 유형 **EXTERNAL** 외부 루틴입니다. **SQL** SQL 루틴입니다.
ROUTINE_DEFINITION	DBCLOB 널값 가능	SQL 루틴인 경우 이 열에는 SQL 루틴 본문이 들어갑니다. 이것이 SQL 루틴이 아니거나 루틴 본문이 절단 없이 이 열에 들어 있을 수 없는 경우에 널값이 들어갑니다.

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
EXTERNAL_NAME	VARCHAR(279) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. • REXX의 경우 외부 프로그램명은 *schema-name/source-file-name(member-name)*입니다. • ILE 서비스 프로그램의 경우 외부 프로그램명은 *schema-name/service-program-name(entry-point-name)* 입니다. • Java 프로그램의 경우 외부 프로그램명은 선택적인 jar-id 다음에 *fully-qualified-class-name!method-name* 또는 *fully-qualified-class-name.method-name*이 나옵니다. • 다른 모든 언어의 경우 외부 프로그램명은 *schema-name/program-name*입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.
EXTERNAL_LANGUAGE	VARCHAR(8) 널값 가능	외부 루틴인 경우 이 열은 외부 프로그램명을 식별합니다. **C** 외부 프로그램이 C로 작성됩니다. **C++** 외부 프로그램이 C++로 작성됩니다. **CL** 외부 프로그램이 CL로 작성됩니다. **COBOL** 외부 프로그램이 COBOL로 작성됩니다. **COBOLLE** 외부 프로그램은 ILE COBOL로 작성됩니다. **FORTRAN** 외부 프로그램이 FORTRAN으로 작성됩니다. **JAVA** 외부 프로그램이 JAVA로 작성됩니다. **PLI** 외부 프로그램이 PL/I으로 작성됩니다. **REXX** 외부 프로그램은 REXX 프로시듀어입니다. **RPG** 외부 프로그램이 RPG로 작성됩니다. **RPGLE** 외부 프로그램은 ILE RPG로 작성됩니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.

ROUTINES

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
PARAMETER_STYLE	VARCHAR(18) 널값 가능	외부 루틴인 경우 이 열은 매개변수 유형을 식별합니다(호출 규칙). **DB2GENERAL** DB2GENERAL 호출 규칙입니다. **DB2SQL** DB2SQL 호출 규칙입니다. **GENERAL** GENERAL 호출 규칙입니다. **JAVA** JAVA 호출 규칙입니다. **GENERAL WITH NULLS** GENERAL WITH NULLS 호출 규칙입니다. **SQL** SQL 표준 호출 규칙입니다. 이것이 외부 루틴이 아닌 경우 널 값이 들어갑니다.
IS_DETERMINISTIC	VARCHAR(3)	이 열은 루틴이 결정적인지 여부를 식별합니다. 다시 말해, 동일한 인수로 루틴을 호출하는 경우 항상 동일한 결과가 리턴되는지의 여부를 식별하는 것입니다. **NO** 루틴이 결정적이 아닙니다. **YES** 루틴이 결정적입니다.
SQL_DATA_ACCESS	VARCHAR(17)	이 열은 SQL이 들어 있는 루틴인지의 여부 및 자료를 읽고 변경하는지 여부를 식별합니다. **NO SQL** 루틴에 SQL문이 없습니다. **CONTAINS SQL** 루틴에 SQL문이 있습니다. **READS SQL DATA** 루틴이 표 또는 뷰로부터 자료를 읽을 수 있습니다. **MODIFIES SQL DATA** 루틴이 표 또는 뷰 내의 자료를 변경하거나 SQL DDL(data definition language) 명령문을 생성할 수 있습니다.

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
IS_NULL_CALL	VARCHAR(3) 널값 가능	입력 매개변수가 널값인 경우 함수를 호출할 필요가 있는지의 여부를 식별합니다. **NO** 입력 매개변수가 널값인 경우 함수를 호출할 필요가 없습니다. 이것이 스칼라 함수라면, 만일 피연산자 중 하나가 널값인 경우 함수의 결과는 내재적으로 널값이 됩니다. 이것이 표 함수라면, 피연산자중 하나라도 널값을 가지는 경우 함수의 결과는 빈 표가 됩니다. **YES** 입력 연산자가 널(null)인 경우라도 이 함수를 반드시 호출하여야 합니다. 함수가 아닌 경우 널값이 들어갑니다.
SQL_PATH	VARCHAR(3483) 널값 가능	SQL 루틴인 경우 이 열은 경로를 식별합니다. 이것이 SQL 루틴이 아닌 경우 널 값이 들어갑니다.
SCHEMA_LEVEL_ROUTINE	VARCHAR(3)	예약. 'YES'를 포함합니다.
MAX_DYNAMIC_RESULT_SETS	SMALLINT	리턴된 결과 세트의 최대수를 식별합니다. 0은 결과 세트가 없음을 의미합니다.
IS_USER_DEFINED_CAST	VARCHAR(3) 널값 가능	함수가 고유한 유형이 작성될 때 작성된 캐스트 함수인지의 여부를 식별합니다. **NO** 캐스트 함수가 아닙니다. **YES** 캐스트 함수입니다. 루틴이 함수가 아닌 경우 널값이 들어갑니다.
IS_IMPLICITLY_INVOCABLE	VARCHAR(3) 널값 가능	이 함수가 고유한 유형이 작성될 때 작성된 캐스트 함수이며 내재적으로 호출할 수 있는지의 여부를 식별합니다. **NO** 캐스트 함수가 아닙니다. **YES** 이 함수는 캐스트 함수이며 내재적으로 호출될 수 있습니다. 루틴이 함수가 아닌 경우 널값이 들어갑니다.
SECURITY_TYPE	VARCHAR(22) 널값 가능	예약. 이것이 외부 루틴인 경우 'IMPLEMENTATION DEFINED'가 들어갑니다. 루틴이 외부 루틴이 아닌 경우 널값이 들어갑니다.
TO_SQL_SPECIFIC_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TO_SQL_SPECIFIC_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
TO_SQL_SPECIFIC_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.

ROUTINES

표 142. ROUTINES 뷰 (계속)

열 이름	자료 유형	설명
AS_LOCATOR	VARCHAR(3) 널값 가능	결과가 로케이터로서 명시되었는지의 여부를 표시합니다. **NO** 매개변수가 로케이터로 명시되지 않았습니다. **YES** 매개변수가 로케이터로 명시되었습니다. 스칼라 함수가 아닌 경우 널값이 들어갑니다.
CREATED	TIMESTAMP	루틴이 작성된 시간소인을 식별합니다.
LAST_ALTERED	TIMESTAMP	예약. 'CREATED'를 포함합니다.

SCHEMATA

SCHEMATA 뷰는 각 스키마에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 143. SCHEMATA 뷰

열 이름	자료 유형	설명
CATALOG_NAME	VARCHAR(128)	관계형 데이터베이스명
SCHEMA_NAME	VARCHAR(128)	스키마명
SCHEMA_OWNER	VARCHAR(128)	스키마 소유자
DEFAULT_CHARACTER_SET_CATALOG	VARCHAR(128)	관계형 데이터베이스명
DEFAULT_CHARACTER_SET_SCHEMA	VARCHAR(128)	디폴트 문자 세트의 스키마명. 'SYSIBM'을 포함합니다.
DEFAULT_CHARACTER_SET_NAME	VARCHAR(128)	디폴트 문자 세트 이름.
SQL_PATH	VARCHAR(3483) 널값 가능	예약. 널 값이 들어갑니다.

SQL_FEATURES

SQL_FEATURES 뷰는 데이터베이스 관리자에 의해 지원된 각 피처에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 144. SQL_FEATURES 뷰

열 이름	자료 유형	설명
FEATURE_ID	VARCHAR(7)	ANS 및 ISO 피처 ID
FEATURE_NAME	VARCHAR(128)	ANS 및 ISO 피처 이름.
SUB_FEATURE_ID	VARCHAR(7)	ANS 및 ISO subfeature ID
SUB_FEATURE_NAME	VARCHAR(256)	ANS 및 ISO subfeature 이름
IS_SUPPORTED	VARCHAR(3)	피처가 지원되었는지 여부를 표시합니다: **YES** 피처는 지원됩니다. **NO** 피처는 지원되지 않습니다.
IS_VERIFIED_BY	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
COMMENTS	VARCHAR(2000) 널값 가능	예약. 널 값이 들어갑니다.

SQL_LANGUAGES

SQL_LANGUAGES(시스템명 SYSLANGS) 표에는 모든 SQL 언어 바인딩 및 일치성이 요구되는 프로그래밍 언어에 대한 하나의 행이 들어 있습니다. 다음 표는 SQL_LANGUAGES 뷰의 열을 설명합니다.

표 145. SQL_LANGUAGES 뷰

열 이름	자료 유형	설명
SQL_LANGUAGE_SOURCE	VARCHAR(254)	표준 이름
SQL_LANGUAGE_YEAR	VARCHAR(254)	표준 승인 연도
SQL_LANGUAGE_CONFORMANCE	VARCHAR(254) 널값 가능	일치 레벨. **2** 1987 및 1989 표준에서, 레벨 2를 준수하도록 규정합니다. **ENTRY** 1992 표준에서, 초기 레벨을 준수하도록 규정합니다. **CORE** 1999 표준에서, 핵심 레벨을 준수하도록 규정합니다. 준수가 규정되지 않은 경우 널값을 포함합니다.
SQL_LANGUAGE_INTEGRITY	VARCHAR(254) 널값 가능	무결성 피처 지원 **YES** 무결성 피처에 대해 일치성이 요구됩니다. **NO** 무결성 피처에 대해 일치성이 요구되지 않습니다. 표준에 별도의 무결성 기능이 없는 경우 널값이 들어 있습니다.
SQL_LANGUAGE_IMPLEMENTATION	VARCHAR(254) 널값 가능	예약. 널 값이 들어갑니다.
SQL_LANGUAGE_BINDING_STYLE	VARCHAR(254)	SQL 언어의 바인딩 스타일 **EMBEDDED** 내부 언어에 대해 삽입 SQL 지원 SQL_LANGUAGE_PROGRAMMING_LANG **DIRECT** DIRECT SQL이 지원됩니다(예를 들면 대화식 SQL). **CLI** 내부 언어에 대한 CLI 지원 SQL_LANGUAGE_PROGRAMMING_LANG

SQL_LANGUAGES

표 145. SQL_LANGUAGES 뷰 (계속)

열 이름	자료 유형	설명
SQL_LANGUAGE_PROGRAMMING_LANG	VARCHAR(254) 널값 가능	EMBEDDED 또는 CLI에 의해 지원되는 언어 **C** C 언어는 지원됩니다. **COBOL** COBOL 언어는 지원됩니다. **PLI** PL/I 언어는 지원됩니다. SQL_LANGUAGE_BINDING_STYLE 이 DIRECT인 경우 널값이 들어갑니다.

SQL_SIZING

SQL_SIZING 뷰는 데이터베이스 관리자에 의해 지원된 각 제한에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 146. SQL_SIZING 뷰

열 이름	자료 유형	설명
SIZING_ID	INTEGER	ANS 및 ISO 크기 지정 ID
SIZING_NAME	VARCHAR(128)	ANS 이름 및 ISO 크기 지정
SUPPORTED_VALUE	INTEGER 널값 가능	크기 지정 한계를 나타냅니다. 크기 지정 한계가 적용 가능하지 않은 경우 널값을 포함합니다.
COMMENTS	VARCHAR(2000) 널값 가능	예약. 널 값이 들어갑니다.

TABLE_CONSTRAINTS

TABLE_CONSTRAINTS 뷰는 각 제한 사항에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 147. TABLE_CONSTRAINTS 뷰

열 이름	자료 유형	설명
CONSTRAINT_CATALOG	VARCHAR(128)	관계형 데이터베이스명
CONSTRAINT_SCHEMA	VARCHAR(128)	제한사항이 들어 있는 스키마명
CONSTRAINT_NAME	VARCHAR(128)	제약 조건 명
TABLE_CATALOG	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEMA	VARCHAR(128)	표가 들어 있는 스키마명
TABLE_NAME	VARCHAR(128)	제약 조건 작성이 완료된 표 이름
CONSTRAINT_TYPE	VARCHAR(11)	제약 조건 유형 CHECK UNIQUE PRIMARY KEY FOREIGN KEY
IS_DEFERRABLE	VARCHAR(3)	제한사항 검사가 지연될 수 있는지 여부를 표시합니다. 'NO'를 포함합니다.
INITIALLY_DEFERRED	VARCHAR(3)	제한사항이 초기에 지연되었는지 여부를 표시합니다. 'NO'를 포함합니다.

TABLES

TABLES 뷰에는 각 표, 뷰, 별명에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 148. TABLES 뷰

열 이름	자료 유형	설명
TABLE_CATALOG	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEMA	VARCHAR(128)	표, 뷰 또는 별명이 들어 있는 SQL 스키마명
TABLE_NAME	VARCHAR(128)	표, 뷰 또는 별명 이름
TABLE_TYPE	VARCHAR(10)	표의 유형을 표시합니다. **ALIAS** 표는 별명입니다. **BASE_TABLE** 표는 SQL 표 또는 실제 파일(PF)입니다. **VIEW** 표는 SQL 뷰 또는 논리 파일입니다.
SELF_REFERENCING_COLUMN_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
REFERENCE_GENERATION	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
USER_DEFINED_TYPE_CATALOG	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
USER_DEFINED_TYPE_NAME	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
IS_INSERTABLE_INTO	VARCHAR(3)	표에서 INSERT가 허용되는지 여부를 식별합니다. **NO** INSERT는 이 표에서 허용되지 않습니다. **YES** INSERT는 이 표에서 허용됩니다.

USER_DEFINED_TYPES

USER_DEFINED_TYPES 뷰는 각 고유한 유형에 대한 하나의 행이 들어 있습니다.[83] 다음 표에서는 뷰의 열을 설명합니다.

표 149. USER_DEFINED_TYPES 뷰

열 이름	자료 유형	설명
USER_DEFINED_TYPE_CATALOG	VARCHAR(128)	관계형 데이터베이스명
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128)	고유한 유형의 스키마명
USER_DEFINED_TYPE_NAME	VARCHAR(128)	고유한 유형을 작성한 사용자의 이름
USER_DEFINED_TYPE_CATEGORY	VARCHAR(128)	사용자 정의 유형을 표시합니다. 'DISTINCT'를 포함합니다.
IS_INSTANTIABLE	VARCHAR(3)	예약. 'YES'를 포함합니다.
IS_FINAL	VARCHAR(3)	예약. 'YES'를 포함합니다.
ORDERING_FORM	VARCHAR(4)	이 고유한 유형이 비교 연산자(comparand)인 경우 허용되는 술부의 종류: **FULL** 모든 술부는 허용됩니다. **NONE** 술부는 허용되지 않습니다.
ORDERING_CATEGORY	VARCHAR(8)	예약. 'MAP'을 포함합니다.
ORDERING_ROUTINE_CATALOG	VARCHAR(128) 널값 가능	관계형 데이터베이스명 ORDERING_FORM이 'NONE'인 경우 널값이 들어갑니다.
ORDERING_ROUTINE_SCHEMA	VARCHAR(128) 널값 가능	예약. 'SYSIBM'을 포함합니다. ORDERING_FORM이 'NONE'인 경우 널값이 들어갑니다.
ORDERING_ROUTINE_NAME	VARCHAR(128) 널값 가능	예약. 자료 유형 이름이 들어갑니다. ORDERING_FORM이 'NONE'인 경우 널값이 들어갑니다.
REFERENCE_TYPE	VARCHAR(16) 널값 가능	예약. 널 값이 들어갑니다.

83. 이 뷰에서는 내장 자료 유형에 대한 정보가 들어 있지 않습니다.

표 149. USER_DEFINED_TYPES 뷰 (계속)

열 이름	자료 유형	설명
DATA_TYPE	VARCHAR(128) 널값 가능	고유한 유형의 소스 자료 유형: **BIGINT** 큰 수 **INTEGER** 큰 수 **SMALLINT** 작은 수 **DECIMAL** 압축 십진수 **NUMERIC** 존(zone) 십진수 **DOUBLE PRECISION** 부동 소수점. 배정밀도 **REAL** 부동 소수점. REAL **CHARACTER** 고정 길이 문자 스트링 **CHARACTER VARYING** 가변 길이의 문자 스트링 **CHARACTER LARGE OBJECT** 문자 LOB(Large Object) 스트링 **GRAPHIC** 고정 길이 그래픽 스트링 **GRAPHIC VARYING** 가변 길이의 그래픽 스트링 **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2바이트 문자 LOB(Large Object) 스트링 **BINARY LARGE OBJECT** 2진 LOB(Large Object) 스트링 **DATE** 날짜 **TIME** 시간 **TIMESTAMP** 시간소인 **DATALINK** 자료 링크 **ROWID** 행 ID **USER-DEFINED** 고유한 유형
CHARACTER_MAXIMUM_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 고유한 유형 최대 길이 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_OCTET_LENGTH	INTEGER 널값 가능	2진수, 문자, 그래픽 스트링 자료 유형의 고유한 유형 바이트 수 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.

USER_DEFINED_TYPES

표 149. USER_DEFINED_TYPES 뷰 (계속)

열 이름	자료 유형	설명
CHARACTER_SET_CATALOG	VARCHAR(128) 널값 가능	고유한 유형의 관계형 데이터베이스명 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_SCHEMA	VARCHAR(128) 널값 가능	고유한 유형 문자 세트의 스키마 이름. 'SYSIBM'을 포함합니다. 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
CHARACTER_SET_NAME	VARCHAR(128) 널값 가능	고유한 유형 문자 세트 이름. 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_CATALOG	VARCHAR(128) 널값 가능	고유한 유형의 관계형 데이터베이스명 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_SCHEMA	VARCHAR(128) 널값 가능	고유한 유형 배열의 스키마. SYSIBM은 리턴됩니다. 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
COLLATION_NAME	VARCHAR(128) 널값 가능	고유한 유형의 배열 이름. IBMBINARY은 리턴됩니다. 고유한 유형이 스트링이 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION	INTEGER 널값 가능	고유한 유형 정밀도. **주:** 이 열은 단정밀도 및 배정밀도 부동 소수점을 포함하여 모든 숫자 자료 유형의 정밀도를 지원합니다. NUMERIC_PRECISION_RADIX 열은 이 열의 값이 2진 자릿수인지 소수 자릿수인지를 표시합니다. 고유한 유형이 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_PRECISION_RADIX	INTEGER 널값 가능	NUMERIC_PRECISION 열 내에 지정된 정밀도가 2진 또는 십진 자릿수의 수로서 지정되었는지의 여부를 표시합니다. **2** 2진. 부동 소수점 정밀도는 2진 자릿수로 명시됩니다. **10** 십진. 다른 모든 숫자 유형은 십진 자릿수로 명시됩니다. 고유한 유형이 숫자가 아닌 경우 널값이 들어갑니다.
NUMERIC_SCALE	INTEGER 널값 가능	숫자 고유한 유형의 스케일. 고유한 유형이 십진, 숫자 또는 2진이 아닌 경우 널값이 들어갑니다.
DATETIME_PRECISION	INTEGER 널값 가능	날짜, 시간 또는 시간소인 고유한 유형의 분수 부분. **0** DATE 및 TIME 자료 유형의 경우 **6** TIMESTAMP 자료 유형의 경우(마이크로 초의 수) 고유한 유형이 날짜, 시간 또는 시간소인이 아닌 경우 널값이 들어갑니다.

표 149. USER_DEFINED_TYPES 뷰 (계속)

열 이름	자료 유형	설명
INTERVAL_TYPE	VARCHAR(128) 널값 가능	예약. 널 값이 들어갑니다.
INTERVAL_PRECISION	INTEGER 널값 가능	예약. 널 값이 들어갑니다.
SOURCE_DTD_IDENTIFIER	VARCHAR(128) 널값 가능	소스 자료 유형에 대한 하나의 고유 내부 식별자 고유한 유형이 다른 고유한 유형의 소스가 되지 않은 경우 널값을 포함합니다.
REF_DTD_IDENTIFIER	VARCHAR(256) 널값 가능	예약. 널 값이 들어갑니다.

VIEWS

VIEWS 뷰는 각 뷰에 대한 하나의 행이 들어 있습니다. 다음 표에서는 뷰의 열을 설명합니다.

표 150. VIEWS 뷰

열 이름	자료 유형	설명
TABLE_CATALOG	VARCHAR(128)	관계형 데이터베이스명
TABLE_SCHEMA	VARCHAR(128)	뷰가 들어 있는 SQL 스키마명
TABLE_NAME	VARCHAR(128)	뷰 이름
VIEW_DEFINITION	VARCHAR (10000) 널값 가능	CREATE VIEW 명령문의 조회 표현식 부분 뷰 정의가 열에 잘리지 않고 포함될 수 없는 경우 널값을 포함합니다.
CHECK_OPTION	VARCHAR(8)	뷰 상에 사용된 검사 옵션 **NONE** 검사 옵션이 지정되지 않았습니다. **LOCAL** 로컬 옵션이 명시되었습니다. **CASCADED** 직렬 옵션이 명시되었습니다.
IS_UPDATABLE	VARCHAR(3)	뷰의 갱신가능 여부를 명시합니다. **YES** 뷰의 갱신이 가능합니다. **NO** 뷰의 읽기 만이 가능합니다.

참고 문헌

여기에 나열된 책들은 이 책에서 서술되고 참조된 주제들에 대한 추가적인 정보를 제공합니다. 리스트에는 전체 제목 및 주문 번호가 함께 나옵니다. 본문에서는 짧은 제목을 사용합니다.

- 백업 및 회복 

  백업 및 회복 전략 수립, 프로시듀어를 저장하고 복원하기 위해 사용가능한 다양한 유형의 매체 및 디스크 회복 프로시듀어에 대한 정보가 있습니다. 또한 백업으로부터 시스템을 재설치하는 방법도 설명되어 있습니다.

- ILE COBOL Programmer's Guide 

  iSeries 400 시스템에서 COBOL/400 프로그램을 설계, 작성, 테스트 및 유지보수하기 위해 필요한 정보를 제공합니다.

- ILE RPG Programmer's Guide 

  iSeries 400 시스템에서 ILE RPG/400 프로그램을 설계, 작성, 테스트 및 유지보수하기 위해 필요한 정보를 제공합니다.

- REXX/400 Programmer's Guide 

  iSeries 400 시스템에서 REXX/400 프로그램을 설계, 작성, 테스트 및 유지보수하기 위해 필요한 정보를 제공합니다.

- CL Programming 

  오브젝트와 라이브러리, CL 프로그래밍, 프로그램 간 흐름 제어 및 통신, CL 프로그램에서 오브젝트에 대한 작업 및 CL 프로그램 작성 등에 대한 일반적인 정보를 포함하여 iSeries 400 프로그래밍 주제에 대한 광범위한 정보를 제공합니다. 기타 주제로는 사전정의되었거나 즉흥적인 메세지 및 사용자 정의 명령과 메뉴의 처리, 정의 및 작성, 어플리케이션 테스팅, 디버그 모드, 중단점, 추적, 및 표시 함수들이 있습니다.

- 파일 관리

  어플리케이션 프로그램의 파일을 사용하는 방법에 대한 정보를 제공합니다.

- 데이터베이스 프로그래밍

  시스템에서 데이터베이스 파일을 작성, 설명 및 갱신하는 방법에 대한 정보를 포함하여 iSeries 데이터베이스 조직에 대한 상세한 설명을 제공합니다.

- 분산 데이터베이스 프로그래밍

  분산 관계형 데이터베이스 구조(DRDA)를 사용하는 분산 관계형 데이터베이스 (DRDB) 내의 iSeries 시스템의 준비와 관리에 대한 정보를 제공합니다. 유사 시스템 환경에서 둘 이상의 iSeries 시스템 상에서 분산 관계형 데이터베이스를 계획, 설정, 프로그래밍, 관리 및 운영에 대하여 설명합니다.

- iSeries Security Reference 

  시스템 보안 개념, 보안 계획 수립 및 시스템 상에 보안 설정에 대한 정보를 제공합니다. 또한 자료를 의도적이거나 실수에 의한 손상 또는 파괴로부터 보호하고, 보안을 항시 최신으로 유지하며, 시스템 상에 보안을 설정하면서, 적절한 권한이 없는 사용자이 시스템 및 자료를 사용하지 못하도록 보호하는 것에 대한 정보도 제공합니다.

- SQL 프로그래밍 개념

  iSeries용 DB2 UDB 명령문을 설계하고, 작성하고, 실행하며 테스트하는 방법에 대한 개요를 제공합니다. 또한 대화식 SQL(Structured Query Language)에 대해서도 설명합니다.

- 호스트 언어로 SQL 프로그래밍

COBOL, ILE COBOL/400, ILE RPG/400, ILE C/400 및 PL/I 프로그램에서 SQL문을 작성하는 방법의 예를 제공합니다.

- 데이터베이스 성능 및 조회 최적화

  사용 가능한 툴과 기법을 사용한 조회 성능 최적화에 대한 정보를 제공합니다.

- IDDU Use 

  자료 사전, 파일 및 레코드를 시스템에 설명하기 위해 iSeries 대화식 자료 정의 유틸리티(IDDU)를 사용하는 방법에 대하여 설명합니다.

- SQL 호출 레벨 인터페이스 (ODBC)

  iSeries용 DB2 UDB가 제공하는 서비스 프로그램에 대한 프로시듀어 호출을 통하여 SQL 함수에 직접 액세스하기 위해 X/Open SQL 호출 레벨 인터페이스를 사용하는 방법이 있습니다.

- iSeries Information Center의 Client Access Express 범주

  Client Access ODBC를 사용하여 클라이언트에 ODBC 어플리케이션을 설정하고 실행하는 방법을 설명합니다. 또한 성능, 예 및 Client Access ODBC로 실행하기 위한 특정 어플리케이션 구성에 대한 장이 포함되어 있습니다.

- IBM Toolbox for Java

  IBM Toolbox for Java를 사용하여 클라이언트에서 JDBC 어플리케이션을 설정하고 실행하는 방법을 설명합니다. 또한 실행, 예 및 IBM Toolbox for Java와 함께 실행하기 위한 특정 어플리케이션 구성에 대한 장이 포함되어 있습니다.

- IBM Developer Kit for Java

  iSeries 시스템에서 JAVA 프로그램을 설계, 작성, 테스트 및 유지보수하는 데 필요한 세부사항을 설명합니다. 또한 IBM Developer Kit for Java JDBC 드라이버에 대한 내용도 제공합니다.

- DB2 Multisystem

  분산 관계형 데이터베이스(DRDB) 파일, 노드 그룹 및 분할에 대한 기본적인 개념을 설명합니다. 또한 복수 시스템에 걸쳐 구획되어 있는 데이터베이스 파일을 작성하고 사용하는 데 필요한정보를 제공합니다. 시스템 구성 방법, 파일 작성 방법 및 어플리케이션 내에서 파일의 사용 방법에 관한 정보가 제공됩니다.

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

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

[특수 문자]

Printed in U.S.A