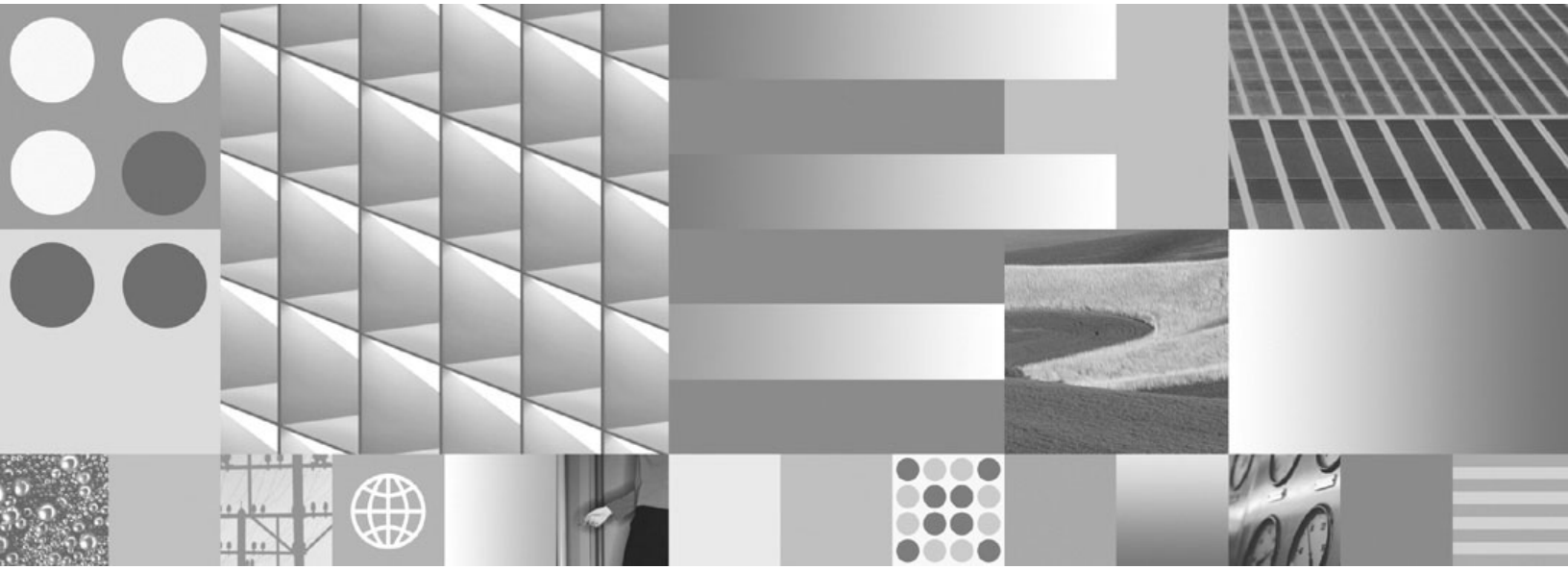
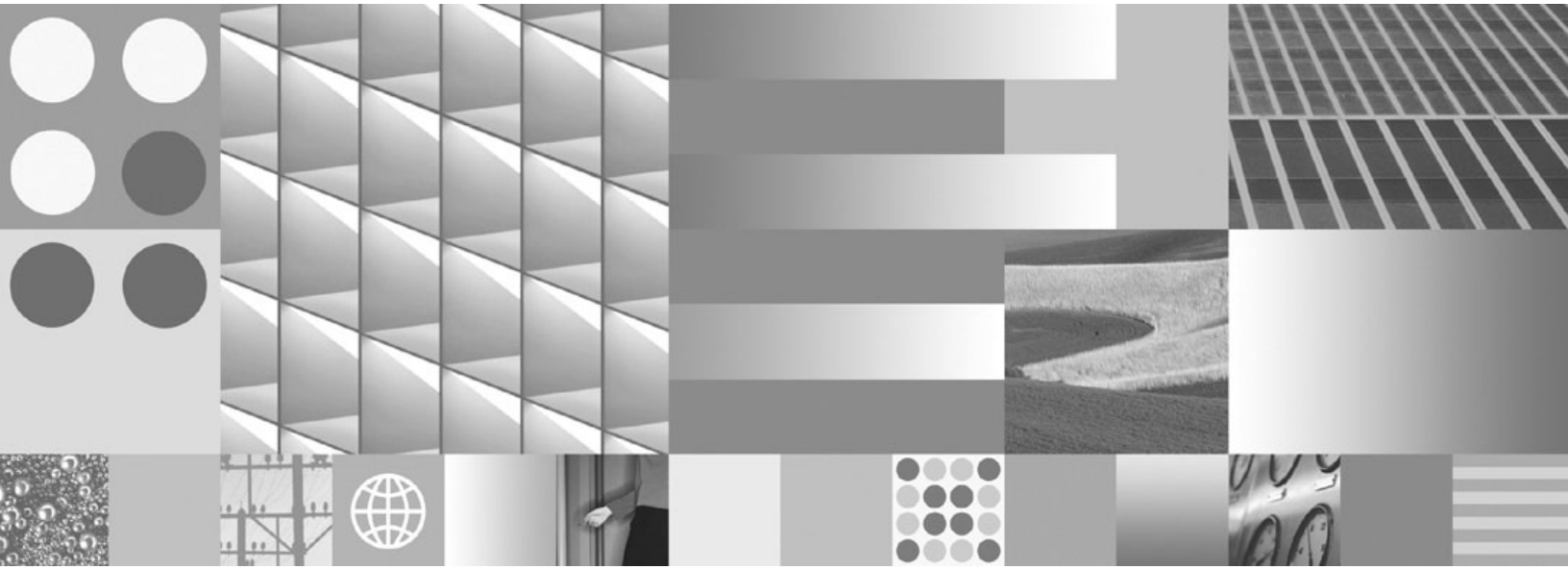
Linux, UNIX 및 Windows용
IBM DB2 9.7



SQL 참조서, 볼륨 2

SA30-3957-00

Linux, UNIX 및 Windows용
IBM DB2 9.7



SQL 참조서, 볼륨 2

SA30-3957-00

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 1377 페이지의 부록 B 『주의사항』의 정보를 읽으십시오.

개정판 주의사항

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

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

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

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

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

목차

이 책에 대한 정보

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

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

이 책의 사용자

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

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

이 책의 구성

SQL 참조서의 두 번째 볼륨에는 SQL문의 구문 및 시맨틱에 관한 정보가 포함됩니다.

- 『명령문』은 구문 다이어그램, 시맨틱 설명, 규칙 및 SQL 프로시저 명령문을 포함한 모든 SQL문의 예가 포함됩니다.

구문 다이어그램을 읽는 방법

구문은 다음과 같은 구조를 사용하여 설명합니다.

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

►►── 기호는 구문 다이어그램의 시작을 표시합니다.

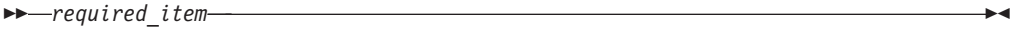
──► 기호는 구문이 다음 라인에서 계속됨을 표시합니다.

►── 기호는 구문이 이전 라인에서 계속됨을 표시합니다.

──►◄ 기호는 구문 다이어그램의 끝을 표시합니다.

구문 조각은 ├── 기호로 시작하고 ──┤ 기호로 끝납니다.

필수 항목은 수평선(주 경로)에 나타납니다.



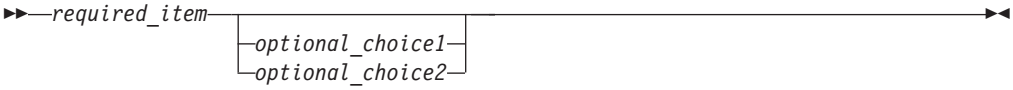
선택 항목은 기본 경로 아래에 나타납니다.

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

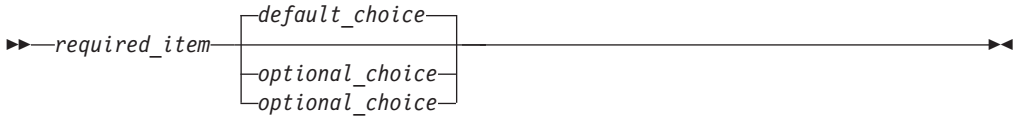
둘 이상의 항목에서 선택할 수 있는 경우 해당 항목은 스택으로 나타납니다.

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

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



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



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

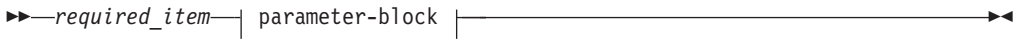
반복 화살표가 쉼표를 포함하는 경우 반복되는 항목을 쉼표로 구분해야 합니다.

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

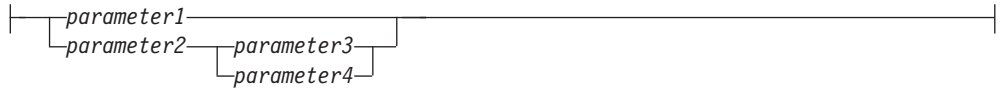
키워드는 대문자로 나타납니다(예: FROM). 키워드는 표시된 대로 정확하게 입력해야 합니다. 변수는 소문자로 나타납니다(예: column-name). 구문에서 사용자가 제공하는 이름이나 값을 나타냅니다.

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

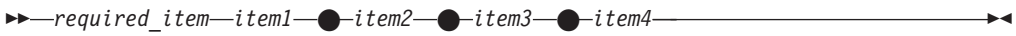
가끔 단일 변수가 구문의 더 큰 조각을 나타냅니다. 예를 들어 다음 다이어그램에서 parameter-block 변수는 **parameter-block**으로 레이블되는 전체 구문 조각을 나타냅니다.



parameter-block:



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



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

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

이 매뉴얼에서 사용된 규칙

오류 조건

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

시그니처가 중복되면 SQL 오류가 리턴됩니다(SQLSTATE 42723).

강조표시 규칙

이 책에서 사용되는 규칙은 다음과 같습니다.

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

관련 문서

다음은 응용프로그램을 준비할 때 유용한 책입니다.

- *Getting Started with Database Application Development*
  - 플랫폼 전제조건, 지원되는 개발 소프트웨어, 지원되는 프로그래밍 API의 장점과 제한사항을 포함한 DB2 응용프로그램 개발에 대한 개요를 제공합니다.
- *i5/OS용 DB2 SQL 참조서*
  - 이 책은 System i®에서 DB2 쿼리 관리 프로그램 및 SQL 개발 킷에서 지원하는 SQL을 정의합니다. 시스템 관리, 데이터베이스 관리, 응용프로그램 프로그래밍 및 조작 태스크에 대한 참조 정보를 포함합니다. 이 매뉴얼에는 DB2를 실행하는 i5/OS® 시스템에서 사용되는 구문, 사용법, 키워드 및 각 SQL문의 예가 포함됩니다.
- *z/OS용 DB2 SQL 참조서*

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

명령문

SQL문

다음 표는 종류별로 분류된 지원되는 SQL문을 나열합니다.

- SQL 스키마 명령문(표 1)
- SQL 데이터 변경 명령문(6 페이지의 표 2)
- SQL 데이터 명령문(6 페이지의 표 3)
- SQL 트랜잭션 명령문(6 페이지의 표 4)
- SQL 연결 명령문(7 페이지의 표 5)
- SQL 동적 명령문(7 페이지의 표 6)
- SQL 세션 명령문(7 페이지의 표 7)
- SQL 임베디드(embedded) 호스트 언어 명령문(8 페이지의 표 8)
- SQL 제어 명령문(8 페이지의 표 9)

표 1. SQL 스키마 명령문

SQL문	목적
25 페이지의 『ALTER AUDIT POLICY』	현재 서버에서 감사 규정의 정의를 수정합니다.
29 페이지의 『ALTER BUFFERPOOL』	버퍼 풀의 정의를 변경합니다.
37 페이지의 『ALTER DATABASE』	자동 스토리지 테이블 스페이스에 사용되는 경로 콜렉션에 새 스토리지 경로를 추가합니다.
32 페이지의 『ALTER DATABASE PARTITION GROUP』	데이터베이스 파티션 그룹의 정의를 변경합니다.
43 페이지의 『ALTER FUNCTION』	함수의 등록 정보를 변경하여 함수 기능을 수정합니다.
46 페이지의 『ALTER HISTOGRAM TEMPLATE』	서비스 클래스 또는 작업 클래스의 하나 이상의 디폴트 막대 그래프를 겹쳐쓰는 데 사용할 수 있는 막대 그래프 유형을 설명하는 템플리트를 수정합니다.
48 페이지의 『ALTER INDEX』	인덱스의 정의를 변경합니다.
50 페이지의 『ALTER METHOD』	메소드와 연관된 메소드 본문을 변경하여 기존 메소드를 수정합니다.
52 페이지의 『ALTER MODULE』	모듈의 정의를 변경합니다.
60 페이지의 『ALTER NICKNAME』	별칭의 정의를 변경합니다.
69 페이지의 『ALTER PACKAGE』	패키지를 바인드하거나 리바인드할 필요 없이 현재 서버에서 패키지의 바인드 옵션을 변경합니다.
72 페이지의 『ALTER PROCEDURE(외부)』	프로시저의 등록 정보를 변경하여 기존 외부 프로시저를 수정합니다.
75 페이지의 『ALTER PROCEDURE(전래)』	하나 이상의 소싱된 프로시저 매개변수의 데이터 유형을 변경하여 기존 소싱된 프로시저를 수정합니다.
77 페이지의 『ALTER PROCEDURE(SQL)』	프로시저의 등록 정보를 변경하여 기존 SQL 프로시저를 수정합니다.
79 페이지의 『ALTER SECURITY LABEL COMPONENT』	보안 레이블 구성요소를 수정합니다.
83 페이지의 『ALTER SECURITY POLICY』	보안 규정을 수정합니다.
88 페이지의 『ALTER SEQUENCE』	시퀀스 정의를 변경합니다.
92 페이지의 『ALTER SERVER』	페더레이티드 시스템에서 데이터 소스의 정의를 변경합니다.
96 페이지의 『ALTER SERVICE CLASS』	서비스 클래스의 정의를 변경합니다.
105 페이지의 『ALTER TABLE』	테이블 정의를 변경합니다.

표 1. SQL 스키마 명령문 (계속)

SQL문	목적
159 페이지의 『ALTER TABLESPACE』	테이블 스페이스 정의를 변경합니다.
174 페이지의 『ALTER THRESHOLD』	임계값의 정의를 변경합니다.
186 페이지의 『ALTER TRUSTED CONTEXT』	현재 서버에서 트러스트된 컨텍스트의 정의를 변경합니다.
196 페이지의 『ALTER TYPE(구조화)』	구조화된 유형의 정의를 변경합니다.
204 페이지의 『ALTER USER MAPPING』	사용자 권한 부여 맵핑의 정의를 변경합니다.
207 페이지의 『ALTER VIEW』	범위를 추가하도록 참조 유형 컬럼을 변경하여 뷰 정의를 변경합니다.
210 페이지의 『ALTER WORK ACTION SET』	작업 조치 세트에서 작업 조치를 추가, 변경 또는 삭제(drop)합니다.
225 페이지의 『ALTER WORK CLASS SET』	작업 클래스 세트에서 작업 클래스를 추가, 변경 또는 삭제(drop)합니다.
231 페이지의 『ALTER WORKLOAD』	워크로드를 변경합니다.
247 페이지의 『ALTER WRAPPER』	랩퍼 모듈과 함께 특정 유형의 데이터 소스에 액세스하기 위해 사용되는 옵션을 갱신합니다.
249 페이지의 『ALTER XSROBJECT』	특정 XML 스키마에 대한 작성 취소 지원을 사용하거나 사용하지 않습니다.
254 페이지의 『AUDIT』	현재 서버에서 특정 데이터베이스 또는 데이터베이스 오브젝트에 사용될 감사 규정을 판별합니다.
275 페이지의 『COMMENT』	오브젝트의 설명을 주석으로 대체하거나 주석을 추가합니다.
333 페이지의 『CREATE ALIAS』	모듈, 별칭, 시퀀스, 테이블, 뷰 또는 다른 별명에 대한 별명을 정의합니다.
337 페이지의 『CREATE AUDIT POLICY』	현재 서버에서 감사 규정을 정의합니다.
341 페이지의 『CREATE BUFFERPOOL』	새로운 버퍼 풀을 생성합니다.
345 페이지의 『CREATE DATABASE PARTITION GROUP』	데이터베이스 파티션 그룹을 정의합니다.
348 페이지의 『CREATE EVENT MONITOR』	모니터할 데이터베이스의 이벤트를 지정합니다.
370 페이지의 『CREATE EVENT MONITOR(활동)』	모니터할 데이터베이스에 활동 이벤트를 지정합니다.
383 페이지의 『CREATE EVENT MONITOR(잠금)』	모니터할 데이터베이스에 잠금 이벤트를 지정합니다.
388 페이지의 『CREATE EVENT MONITOR(통계)』	모니터할 데이터베이스에 통계 이벤트를 지정합니다.
402 페이지의 『CREATE EVENT MONITOR(임계값 위반)』	모니터할 데이터베이스에 임계값 위반 이벤트를 지정합니다.
416 페이지의 『CREATE EVENT MONITOR(작업 단위)』	모니터할 데이터베이스에 작업 단위(UOW) 이벤트를 지정합니다.
422 페이지의 『CREATE FUNCTION』	사용자 정의 함수(UDF)를 등록합니다.
423 페이지의 『CREATE FUNCTION(외부 스칼라)』	사용자 정의 외부 스칼라 함수를 등록합니다.
453 페이지의 『CREATE FUNCTION(외부 테이블)』	사용자 정의 외부 테이블 함수를 등록합니다.
476 페이지의 『CREATE FUNCTION(OLE DB 외부 테이블)』	사용자 정의 OLE DB 외부 테이블 함수를 등록합니다.
485 페이지의 『CREATE FUNCTION(소스 또는 템플리트)』	사용자 정의 전래 함수를 등록합니다.
501 페이지의 『CREATE FUNCTION(SQL 스칼라, 테이블 또는 행)』	사용자 정의 SQL 함수를 등록 및 정의합니다.
516 페이지의 『CREATE FUNCTION MAPPING』	함수 맵핑을 정의합니다.
521 페이지의 『CREATE GLOBAL TEMPORARY TABLE』	작성된 임시 테이블을 정의합니다.
535 페이지의 『CREATE HISTOGRAM TEMPLATE』	서비스 클래스 또는 작업 클래스의 하나 이상의 디폴트 막대 그래프를 겹쳐쓰는 데 사용할 수 있는 막대 그래프 유형을 설명하는 템플리트를 정의합니다.

표 1. SQL 스키마 명령문 (계속)

SQL문	목적
537 페이지의『CREATE INDEX』	테이블에 대한 인덱스를 정의합니다.
559 페이지의『CREATE INDEX EXTENSION』	구조화된 유형 또는 구별 유형의 컬럼이 있는 테이블에서 인덱스와 함께 사용할 확장 오브젝트를 정의합니다.
567 페이지의『CREATE METHOD』	메소드 본문을 이전에 정의된 메소드 스펙과 연관시킵니다.
573 페이지의『CREATE MODULE』	모듈을 정의합니다.
575 페이지의『CREATE NICKNAME』	별칭을 정의합니다.
589 페이지의『CREATE PROCEDURE』	프로시저를 등록합니다.
590 페이지의『CREATE PROCEDURE(외부)』	외부 프로시저를 등록합니다.
608 페이지의『CREATE PROCEDURE(전래)』	다른 프로시저(소싱된 프로시저)를 기반으로 프로시저(목표 프로시저)를 등록합니다. 페더레이티드 시스템에서 페더레이티드 프로시저는 소싱된 프로시저가 지원되는 데이터 소스에 있는 목표 프로시저입니다.
615 페이지의『CREATE PROCEDURE(SQL)』	SQL 프로시저를 등록합니다.
626 페이지의『CREATE ROLE』	현재 서버에서 역할을 정의합니다.
627 페이지의『CREATE SCHEMA』	스키마를 정의합니다.
630 페이지의『CREATE SECURITY LABEL COMPONENT』	보안 규정의 일부로 사용할 구성요소를 작성합니다.
633 페이지의『CREATE SECURITY LABEL』	보안 레이블을 작성합니다.
635 페이지의『CREATE SECURITY POLICY』	보안 규정을 작성합니다.
637 페이지의『CREATE SEQUENCE』	시퀀스를 정의합니다.
652 페이지의『CREATE SERVER』	데이터 소스를 페더레이티드 데이터베이스에 정의합니다.
642 페이지의『CREATE SERVICE CLASS』	서비스 클래스를 정의합니다.
656 페이지의『CREATE SYNONYM』	모듈, 별칭, 시퀀스, 테이블, 뷰 또는 다른 동의어에 대한 동의어를 정의합니다.
657 페이지의『CREATE TABLE』	테이블을 정의합니다.
741 페이지의『CREATE TABLESPACE』	테이블 스페이스를 정의합니다.
757 페이지의『CREATE THRESHOLD』	임계값을 정의합니다.
772 페이지의『CREATE TRANSFORM』	변환 함수를 정의합니다.
776 페이지의『CREATE TRIGGER』	트리거를 정의합니다.
791 페이지의『CREATE TRUSTED CONTEXT』	현재 서버에서 트러스트된 컨텍스트를 정의합니다.
799 페이지의『CREATE TYPE(배열)』	배열 유형을 정의합니다.
805 페이지의『CREATE TYPE(커서)』	커서 유형을 정의합니다.
808 페이지의『CREATE TYPE(구별)』	구별 데이터 유형을 정의합니다.
816 페이지의『CREATE TYPE(행)』	행 유형을 정의합니다.
821 페이지의『CREATE TYPE(구조화)』	구조화된 데이터 유형을 정의합니다.
849 페이지의『CREATE TYPE MAPPING』	데이터 유형 간의 맵핑을 정의합니다.
857 페이지의『CREATE USER MAPPING』	사용자 권한 부여 간의 맵핑을 정의합니다.
859 페이지의『CREATE VARIABLE』	전역 변수를 정의합니다.
869 페이지의『CREATE VIEW』	하나 이상의 테이블, 뷰 또는 별칭의 뷰를 정의합니다.
886 페이지의『CREATE WORK ACTION SET』	작업 조치 세트에서 작업 조치 세트 및 작업 조치를 정의합니다.
896 페이지의『CREATE WORK CLASS SET』	작업 클래스 세트를 정의합니다.
902 페이지의『CREATE WORKLOAD』	워크로드를 정의합니다.
920 페이지의『CREATE WRAPPER』	랩퍼를 등록합니다.

표 1. SQL 스키마 명령문 (계속)

SQL문	목적
964 페이지의 『DROP』	데이터베이스에서 오브젝트를 삭제합니다.
1040 페이지의 『GRANT(데이터베이스 권한)』	전체 데이터베이스에 대해 권한을 부여합니다.
1046 페이지의 『GRANT(면제)』	지정된 레이블 기반 액세스 제어(LBAC) 보안 규정의 액세스 규칙에 대한 면제를 허용합니다.
1049 페이지의 『GRANT(전역 변수 특권)』	작성된 전역 변수에 대한 하나 이상의 특권을 부여합니다.
1052 페이지의 『GRANT(인덱스 권한)』	데이터베이스에 있는 인덱스에 대해 CONTROL 특권을 부여합니다.
1054 페이지의 『GRANT(모듈 특권)』	모듈에 대해 특권을 부여합니다.
1056 페이지의 『GRANT(패키지 특권)』	데이터베이스에 있는 패키지에 대해 특권을 부여합니다.
1060 페이지의 『GRANT(역할)』	사용자, 그룹 또는 기타 역할에 역할을 부여합니다.
1063 페이지의 『GRANT(루틴 특권)』	루틴(기능, 메소드 또는 프로시저)에 대해 특권을 부여합니다.
1068 페이지의 『GRANT(스키마 특권)』	스키마에 대해 특권을 부여합니다.
1071 페이지의 『GRANT(보안 레이블)』	레이블 기반 액세스 제어(LBAC) 보안 레이블에 읽기 액세스, 쓰기 액세스 또는 읽기 및 쓰기 액세스를 모두 허용합니다.
1074 페이지의 『GRANT(시퀀스 특권)』	시퀀스에 대해 특권을 부여합니다.
1077 페이지의 『GRANT(서버 특권)』	특정 데이터 소스를 쿼리할 수 있는 특권을 부여합니다.
1080 페이지의 『GRANT(SETSESSIONUSER 특권)』	SET SESSION AUTHORIZATION문을 사용할 특권을 부여합니다.
1082 페이지의 『GRANT(테이블 스페이스 특권)』	테이블 스페이스에 대해 특권을 부여합니다.
1085 페이지의 『GRANT(테이블, 뷰 또는 별칭 특권)』	테이블, 뷰 및 별칭에 대해 특권을 부여합니다.
1093 페이지의 『GRANT(워크로드 특권)』	워크로드에 대한 USAGE 특권을 부여합니다.
1096 페이지의 『GRANT(XSR 오브젝트 특권)』	XSR 오브젝트에 대한 USAGE 특권을 부여합니다.
1144 페이지의 『REFRESH TABLE』	구체화된 쿼리 테이블의 데이터를 새로 고칩니다.
1151 페이지의 『RENAME』	기존 테이블의 이름을 바꿉니다.
1154 페이지의 『RENAME TABLESPACE』	기존 테이블 스페이스의 이름을 바꿉니다.
1164 페이지의 『REVOKE(데이터베이스 권한)』	전체 데이터베이스에서 권한을 취소합니다.
1169 페이지의 『REVOKE(면제)』	지정된 레이블 기반 액세스 제어(LBAC) 보안 규정의 액세스 규칙에 대한 면제를 취소합니다.
1172 페이지의 『REVOKE(전역 변수 특권)』	작성된 전역 변수에 대한 하나 이상의 특권을 취소합니다.
1175 페이지의 『REVOKE(인덱스 권한)』	제공된 인덱스에 대해 CONTROL 특권을 취소합니다.
1177 페이지의 『REVOKE(모듈 특권)』	모듈에 대해 특권을 취소합니다.
1179 페이지의 『REVOKE(패키지 특권)』	데이터베이스의 주어진 패키지에서 특권을 취소합니다.
1182 페이지의 『REVOKE(역할)』	사용자, 그룹 또는 기타 역할에서 역할을 취소합니다.
1185 페이지의 『REVOKE(루틴 특권)』	루틴(기능, 메소드 또는 프로시저)에 대해 특권을 취소합니다.
1190 페이지의 『REVOKE(스키마 특권)』	스키마에 대해 특권을 취소합니다.
1193 페이지의 『REVOKE(보안 레이블)』	레이블 기반 액세스 제어(LBAC) 보안 레이블에서 읽기 액세스, 쓰기 액세스 또는 읽기 및 쓰기 액세스를 모두 취소합니다.
1195 페이지의 『REVOKE(시퀀스 특권)』	시퀀스에 대해 특권을 취소합니다.
1198 페이지의 『REVOKE(서버 특권)』	특정 데이터 소스를 쿼리할 수 있는 특권을 취소합니다.
1200 페이지의 『REVOKE(SETSESSIONUSER 특권)』	SET SESSION AUTHORIZATION문을 사용할 특권을 취소합니다.
1202 페이지의 『REVOKE(테이블 스페이스 특권)』	주어진 테이블 스페이스에 대해 USE 특권을 취소합니다.

표 1. SQL 스키마 명령문 (계속)

SQL문	목적
1204 페이지의 『REVOKE(테이블, 뷰 또는 별칭 특권)』	제공된 테이블, 뷰 또는 별칭으로부터 특권을 취소합니다.
1210 페이지의 『REVOKE(워크로드 특권)』	워크로드에 대한 USAGE 특권을 취소합니다.
1212 페이지의 『REVOKE(XSR 오브젝트 특권)』	XSR 오브젝트에 대한 USAGE 특권을 취소합니다.
1274 페이지의 『SET INTEGRITY』	무결성 설정 보류 상태를 설정하고 제한사항 위반에 대해 데이터를 점검합니다.
1325 페이지의 『TRANSFER OWNERSHIP』	데이터베이스 오브젝트의 소유권을 전송합니다.

표 2. SQL 데이터 변경 명령문

SQL문	목적
944 페이지의 『DELETE』	테이블에서 두 개 이상의 행을 삭제합니다.
1101 페이지의 『INSERT』	테이블에 한 개 이상의 행을 삽입합니다.
1120 페이지의 『MERGE』	소스의 데이터를 사용하여(테이블 참조 결과) 목표(테이블 또는 뷰)를 갱신합니다.
1342 페이지의 『TRUNCATE』	테이블에서 행을 모두 삭제합니다.
1345 페이지의 『UPDATE』	테이블의 하나 이상의 행에서 하나 이상의 컬럼에 대한 값을 갱신합니다.

표 3. SQL 데이터 명령문

SQL문	목적
23 페이지의 『ALLOCATE CURSOR』	결과 세트 로케이터 변수로 식별되는 결과 세트에 대한 커서를 할당합니다.
251 페이지의 『ASSOCIATE LOCATORS』	프로시저가 리턴하는 결과 세트 각각에 대한 결과 세트 로케이터 값을 가져옵니다.
272 페이지의 『CLOSE』	커서를 닫습니다.
922 페이지의 『DECLARE CURSOR』	SQL 커서를 정의합니다.
1022 페이지의 『FETCH』	호스트 변수에 행 값을 지정합니다.
1027 페이지의 『FLUSH EVENT MONITOR』	이벤트 모니터의 사용 중인 내부 버퍼를 기록합니다.
1030 페이지의 『FLUSH PACKAGE CACHE』	패키지 캐시에 현재 있는 모든 캐시 동적 SQL문을 제거합니다.
1034 페이지의 『FREE LOCATOR』	로케이터 변수와 로케이터 값 사이의 연관을 제거합니다.
1116 페이지의 『LOCK TABLE』	동시 프로세스로 인해 테이블이 변경되거나 테이블이 사용되지 않도록 합니다.
1131 페이지의 『OPEN』	FETCH문이 발행될 때 값을 검색하기 위해 사용되는 커서를 준비합니다.
1220 페이지의 『SELECT INTO』	한 행의 결과 테이블을 지정하고 호스트 변수에 값을 지정합니다.
1309 페이지의 『SET variable』	NEW 전이 변수에 값을 지정합니다.
1358 페이지의 『VALUES INTO』	한 행의 결과 테이블을 지정하고 호스트 변수에 값을 지정합니다.

표 4. SQL 트랜잭션 명령문

SQL문	목적
289 페이지의 『COMMIT』	작업 단위(UOW)를 종료하고 해당 작업 단위(UOW)에서 수행한 데이터베이스 변경을 커미트합니다.
1150 페이지의 『RELEASE SAVEPOINT』	트랜잭션 내에서 세이브포인트를 해제합니다.
1213 페이지의 『ROLLBACK』	작업 단위(UOW)를 종료하고 작업 단위(UOW)에서 수행한 데이터베이스 변경사항을 취소합니다.
1216 페이지의 『SAVEPOINT』	트랜잭션 내에 세이브포인트를 설정합니다.

표 5. SQL 연결 명령문

SQL문	목적
317 페이지의 『CONNECT(유형 1)』	리모트 작업 단위의 규칙에 따라 응용프로그램 서버(AS)에 연결합니다.
325 페이지의 『CONNECT(유형 2)』	응용프로그램 지향 분산 작업 단위(DUOW)의 규칙에 따라 응용프로그램 서버(AS)에 연결합니다.
961 페이지의 『DISCONNECT』	사용 중인 작업 단위가 없을 때, 하나 이상의 연결을 종료합니다.
1148 페이지의 『RELEASE(연결)』	하나 이상의 연결을 해제 보류 상태로 둡니다.
1226 페이지의 『SET CONNECTION』	연결 상태를 유휴에서 현재로 변경하여 지정된 위치를 현재 서버로 지정합니다.

표 6. SQL 동적 명령문

SQL문	목적
951 페이지의 『DESCRIBE』	오브젝트에 대한 정보를 얻습니다.
952 페이지의 『DESCRIBE INPUT』	PREPARE문의 입력 매개변수 표시문자에 대한 정보를 얻습니다.
956 페이지의 『DESCRIBE OUTPUT』	PREPARE문에 대한 정보 또는 Prepared SELECT문의 선택 목록 컬럼에 대한 정보를 얻습니다.
1004 페이지의 『EXECUTE』	Prepared SQL문을 실행합니다.
1013 페이지의 『EXECUTE IMMEDIATE』	SQL문을 준비하고 실행합니다.
1137 페이지의 『PREPARE』	실행을 위해 SQL문(선택적 매개변수와 함께)을 준비합니다.

표 7. SQL 세션 명령문

SQL문	목적
929 페이지의 『DECLARE GLOBAL TEMPORARY TABLE』	선언된 임시 테이블을 정의합니다.
1016 페이지의 『EXPLAIN』	선택된 액세스 플랜에 관한 정보를 캡처합니다.
1224 페이지의 『SET COMPILATION ENVIRONMENT』	교착 상태 이벤트 모니터에서 제공하는 컴파일 환경에 포함된 값과 일치하도록 연결된 현재 컴파일 환경을 변경합니다.
1228 페이지의 『SET CURRENT DECFLOAT ROUNDING MODE』	지정된 반올림 모드가 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터에 대해 현재 설정된 값인지 검증합니다.
1230 페이지의 『SET CURRENT DEFAULT TRANSFORM GROUP』	CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터의 값을 변경합니다.
1232 페이지의 『SET CURRENT DEGREE』	CURRENT DEGREE 특수 레지스터의 값을 변경합니다.
1234 페이지의 『SET CURRENT EXPLAIN MODE』	CURRENT EXPLAIN MODE 특수 레지스터의 값을 변경합니다.
1237 페이지의 『SET CURRENT EXPLAIN SNAPSHOT』	CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 값을 변경합니다.
1240 페이지의 『SET CURRENT FEDERATED ASYNCHRONY』	CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 값을 변경합니다.
1242 페이지의 『SET CURRENT IMPLICIT XMLPARSE OPTION』	CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 값을 변경합니다.
1243 페이지의 『SET CURRENT ISOLATION』	CURRENT ISOLATION 특수 레지스터의 값을 변경합니다.
1244 페이지의 『SET CURRENT LOCALE LC_TIME』	CURRENT LOCALE LC_TIME 특수 레지스터 값을 변경합니다.
1246 페이지의 『SET CURRENT LOCK TIMEOUT』	CURRENT LOCK TIMEOUT 특수 레지스터의 값을 변경합니다.

표 7. SQL 세션 명령문 (계속)

SQL문	목적
1248 페이지의『SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION』	CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터의 값을 변경합니다.
1251 페이지의『SET CURRENT MDC ROLLOUT MODE』	CURRENT MDC ROLLOUT MODE 특수 레지스터에 값을 지정합니다.
1253 페이지의『SET CURRENT OPTIMIZATION PROFILE』	CURRENT OPTIMIZATION PROFILE 특수 레지스터에 값을 지정합니다.
1257 페이지의『SET CURRENT PACKAGE PATH』	CURRENT PACKAGE PATH 특수 레지스터에 값을 지정합니다.
1262 페이지의『SET CURRENT PACKAGESET』	선택한 패키지에 대한 스키마 이름을 설정합니다.
1264 페이지의『SET CURRENT QUERY OPTIMIZATION』	CURRENT QUERY OPTIMIZATION 특수 레지스터의 값을 변경합니다.
1267 페이지의『SET CURRENT REFRESH AGE』	CURRENT REFRESH AGE 특수 레지스터의 값을 변경합니다.
1269 페이지의『SET ENCRYPTION PASSWORD』	암호화를 위해 암호를 설정합니다.
1271 페이지의『SET EVENT MONITOR STATE』	이벤트 모니터를 활성화하거나 비활성화합니다.
1295 페이지의『SET PASSTHRU』	데이터 소스의 원시(native) SQL을 데이터 소스에 직접 제출하기 위한 세션을 엽니다.
1297 페이지의『SET PATH』	CURRENT PATH 특수 레지스터의 값을 변경합니다.
1300 페이지의『SET ROLE』	세션의 권한 부여 ID가 특정 역할의 구성원인지 검증합니다.
1301 페이지의『SET SCHEM』	CURRENT SCHEMA 특수 레지스터의 값을 변경합니다.
1304 페이지의『SET SERVER OPTION』	서버 옵션 설정값을 설정합니다.
1306 페이지의『SET SESSION AUTHORIZATION』	SESSION USER 특수 레지스터의 값을 변경합니다.

표 8. SQL 임베디드(embedded) 호스트 언어 명령문

SQL문	목적
258 페이지의『BEGIN DECLARE SECTION』	호스트 변수 선언 섹션의 시작 부분을 표시합니다.
1003 페이지의『END DECLARE SECTION』	호스트 변수 선언 섹션의 끝을 표시합니다.
1035 페이지의『GET DIAGNOSTICS』	이전에 실행된 SQL문에 대한 정보를 얻는 데 사용됩니다.
1099 페이지의『INCLUDE』	소스 프로그램에 코드나 선언을 삽입합니다.
1158 페이지의『RESIGNAL』	오류 또는 경고 조건을 다시 알리는 데 사용됩니다.
1322 페이지의『SIGNAL』	오류 또는 경고 조건을 알리는 데 사용됩니다.
1361 페이지의『WHENEVER』	SQL 리턴 코드에 따라 취해질 조치를 정의합니다.

표 9. SQL 제어 명령문

SQL문	목적
260 페이지의『CALL』	프로시저를 호출합니다.
269 페이지의『CASE』	여러 조건에 따라 실행 경로를 선택합니다.
291 페이지의『복합 SQL(인라인된)』	하나 이상의 SQL문을 동적 블록에 조인합니다.
297 페이지의『복합 SQL(임베디드)』	하나 이상의 SQL문을 실행 가능한 블록에 조인합니다.
301 페이지의『복합 SQL(컴파일된)』	SQL 프로시저에서 다른 명령문들을 함께 그룹으로 묶습니다.
1031 페이지의『FOR』	테이블의 각 행에 대해 하나의 명령문이나 명령문 그룹을 실행합니다.

표 9. SQL 제어 명령문 (계속)

SQL문	목적
1038 페이지의 『GOTO』	SQL 프로시저 내에서 사용자 정의 레이블로 분기하는 데 사용됩니다.
1097 페이지의 『IF』	조건의 평가에 따라 실행 경로를 선택합니다.
1112 페이지의 『ITERATE』	제어 흐름을 레이블이 붙은 루프의 시작 부분으로 리턴합니다.
1114 페이지의 『LEAVE』	프로그램 제어를 루프나 복합 명령문 밖으로 전송합니다.
1118 페이지의 『LOOP』	명령문이나 명령문 그룹의 실행을 반복합니다.
1156 페이지의 『REPEAT』	검색 조건이 참이 될 때까지 하나의 명령문이나 명령문 그룹을 실행합니다.
1158 페이지의 『RESIGNAL』	오류 또는 경고 조건을 다시 알리는 데 사용됩니다.
1161 페이지의 『RETURN』	루틴에서부터 리턴하는 데 사용됩니다.
1322 페이지의 『SIGNAL』	오류 또는 경고 조건을 알리는 데 사용됩니다.
1363 페이지의 『WHILE』	지정된 조건이 참인 동안 하나의 명령문이나 명령문 그룹의 실행을 반복합니다.

SQL문 호출 방법

SQL문은 실행 가능한 명령문과 실행 불가능한 명령문으로 구분됩니다.

*실행*문은 네 가지 방법으로 호출할 수 있습니다. 다음과 같은 이름이 가능합니다.

- 응용프로그램에 임베드
- SQL 프로시저에 임베드
- 동적으로 준비 및 실행
- 대화식 실행

명령문에 따라 이 방법 중 일부 또는 전체를 사용할 수 있습니다. (REXX™에 임베드된 명령문은 동적으로 준비되어 실행됩니다.)

*비실행*문은 응용프로그램에만 임베드할 수 있습니다.

다른 SQL문 구문은 select문입니다. *select-statement*는 세 가지 방식으로 호출할 수 있습니다. 다음과 같은 이름이 가능합니다.

- DECLARE CURSOR에 포함되어 OPEN, FETCH 및 CLOSE로 내재적으로 실행(정적 호출)
- 동적으로 준비, DECLARE CURSOR에서 참조 및 OPEN, FETCH 및 CLOSE로 내재적으로 실행(동적 호출)
- 대화식 실행

응용프로그램에 명령문 임베드

SQL문은 프리컴파일러에 제출되는 소스 프로그램에 포함될 수 있습니다. 이런 명령문을 프로그램에 *임베디드*된다고 합니다. 호스트 언어 명령문이 허용되는 프로그램의 모든 위치에 임베디드 명령문을 배치할 수 있습니다. 각 임베디드 명령문 앞에는 키워드 EXEC SQL이 표시되어 있어야 합니다.

응용프로그램에 임베드된 실행문은 호스트 언어 명령문이 동일한 위치에 지정된 경우, 해당 명령문이 실행될 때마다 실행됩니다. 따라서 루프에 있는 명령문은 루프가 실행될 때마다 실행되고 조건부 구문에 있는 명령문은 조건이 충족될 때만 실행됩니다.

임베디드 명령문은 호스트 변수에 대한 참조를 포함할 수 있습니다. 이 방법으로 참조된 호스트 변수는 두 가지 방식으로 사용할 수 있습니다. 다음과 같이 사용됩니다.

- 입력으로 사용(호스트 변수의 현재 값이 명령문 실행에 사용됩니다.)
- 출력으로 사용(명령문 실행 결과, 변수는 새 값으로 지정됩니다.)

특히, 표현식 및 술어에서 호스트 변수에 대한 모든 참조가 변수의 현재 값으로 적절하게 대체됩니다. 즉, 변수가 입력으로 사용됩니다.

모든 실행문 뒤에는 SQL 리턴 코드 테스트가 수행되어야 합니다. 또는 WHENEVER문(자체적으로는 실행할 수 없음)을 사용하여 임베디드 명령문 실행 후에 즉시 제어 순서를 변경할 수도 있습니다.

데이터 처리 언어(DML) 명령문에서 참조된 모든 오브젝트는 명령문이 데이터베이스에 바운드될 때 존재해야 합니다.

임베디드 비실행문은 프리컴파일러로만 처리됩니다. 프리컴파일러가 명령문에서 발생한 모든 오류를 보고합니다. 명령문은 프로그램 실행 중에는 처리되지 *않으며* 이런 명령문 실행 후에는 SQL 리턴 코드 테스트가 수행되지 않아야 합니다.

명령문은 CREATE PROCEDURE문의 SQL-procedure-body 부분에 포함될 수 있습니다. 이런 명령문을 SQL 프로시저에서 임베디드라고 합니다. SQL문 설명이 *host-variable*을 참조할 때마다 명령문이 SQL 프로시저에 임베드된 경우 *SQL-variable*을 사용할 수 있습니다.

동적 준비 및 실행

응용프로그램은 호스트 변수에 배치되는 문자열 양식으로 SQL문을 동적으로 빌드할 수 있습니다. 일반적으로 명령문은 프로그램에 사용 가능한 일부 데이터를 사용하여 빌드됩니다(예: 워크스테이션에서 입력). 명령문(select문이 아님) 구문은 (임베디드) PREPARE문으로 실행할 수 있도록 준비되어 (임베디드) EXECUTE문으로 실행할 수 있습니다. 또는 임베디드 EXECUTE IMMEDIATE문을 사용하면 한 단계로 명령문을 준비하여 실행할 수 있습니다.

동적으로 준비되는 명령문은 호스트 변수에 대한 참조를 포함하면 안됩니다. 대신 매개변수 표시문자를 포함할 수는 있습니다. (매개변수 표시문자와 관련된 규칙은 『PREPARE』를 참조하십시오.) PREPARE문이 실행될 때 매개변수 표시문자는 EXECUTE문에 지정된 호스트 변수의 현재 값으로 적절하게 교체됩니다. 준비되면 명령문은 호스트 변수에 대한 여러 값으로 여러 번 실행할 수 있습니다. 매개변수 표시문자는 EXECUTE IMMEDIATE문에서 허용되지 않습니다.

명령문 실행 성공이나 실패는 EXECUTE(또는 EXECUTE IMMEDIATE)문 완료 후 SQLCA에 SQL 리턴 코드 설정으로 표시됩니다. SQL 리턴 코드는 위에서 설명한 대로 점검해야 합니다. 자세한 정보는 12 페이지의 『SQL 리턴 코드(SQLCODE 및 SQLSTATE)』를 참조하십시오.

select문의 정적 호출

select문은 DECLARE CURSOR문(실행할 수 없음) 파트로 포함될 수 있습니다. 이런 명령문은 커서를 (임베디드) OPEN문으로 열 때마다 실행됩니다. 커서를 연 후 결과 테이블은 연속된 FETCH문 실행으로 한번에 한 행씩 검색할 수 있습니다.

이런 방식으로 사용하여 select문은 호스트 변수에 대한 참조를 포함할 수 있습니다. 이 참조는 OPEN문 실행 시 변수가 포함하는 값으로 적절히 교체됩니다.

select문의 동적 호출

응용프로그램은 호스트 변수에 배치되는 문자열 양식으로 select문을 동적으로 빌드할 수 있습니다. 일반적으로 명령문은 프로그램에 사용 가능한 일부 데이터를 사용하여 빌드됩니다(예: 워크스테이션에서 가져온 조회). 구문화된 명령문은 (임베디드) PREPARE 문으로 실행하기 위해 준비되어 (실행할 수 없는) DECLARE CURSOR문으로 참조됩니다. 그러면 명령문은 (임베디드) OPEN문으로 커서를 열 때마다 실행됩니다. 커서를 연 후 결과 테이블은 연속된 FETCH문 실행으로 한번에 한 행씩 검색할 수 있습니다.

이런 방식으로 사용하여 select문은 호스트 변수에 대한 참조를 포함하면 안됩니다. 대신 매개변수 표시문자를 포함할 수는 있습니다. 매개변수 표시문자는 OPEN문에 지정된 호스트 변수 값으로 적절하게 교체됩니다.

대화식 호출

워크스테이션에서 SQL문을 입력하는 기능은 데이터베이스 관리 프로그램 아키텍처의 파트입니다. 이런 방식으로 입력하는 명령문을 대화식으로 실행된다고 합니다. 이런 명령문은 응용프로그램 컨텍스트에서만 의미가 있기 때문에 매개변수 표시문자 또는 호스트 변수에 대한 참조를 포함하지 않는 실행문이어야 합니다.

다른 호스트 시스템에서 SQL 사용

SQL문 구문은 여러 유형의 호스트 시스템(예: z/OS용 DB2, System i용 DB2, Linux, UNIX 및 Windows용 DB2 데이터베이스)에서 약간 변형된 형태로 사용됩니다. 응용프로그램의 SQL문이 정적 또는 동적인지에 상관없이, 응용프로그램이 다양한 데이터베이스 호스트 시스템에 액세스하는 경우 응용프로그램이 액세스하는 데이터베이스 시스템에서 반드시 SQL문 및 프리컴파일/바인드 옵션이 지원되어야 합니다.

다른 호스트 시스템에서 사용되는 SQL문에 대한 추가 정보는 *System i용 DB2 SQL 참조* 및 *z/OS용 DB2 SQL 참조*에서 제공됩니다.

SQL 리턴 코드(SQLCODE 및 SQLSTATE)

실행할 수 있는 SQL문이 포함된 응용프로그램은 SQLCODE 또는 SQLSTATE 값을 사용하여 SQL문의 리턴 코드를 처리할 수 있습니다. 두 가지 방법으로 응용프로그램이 이 값에 액세스할 수 있습니다.

- SQLCA 이름의 구조 포함. SQLCA에는 이름이 SQLCODE인 정수 변수 및 이름이 SQLSTATE인 문자열 변수가 포함됩니다. REXX에서 SQLCA는 자동으로 제공됩니다. 다른 언어에서는 INCLUDE SQLCA문을 사용하여 SQLCA를 가져올 수 있습니다.

- 프리컴파일 옵션으로 LANGLEVEL SQL92E가 지정된 경우 SQLCODE 또는 SQLSTATE 이름의 변수가 프로그램의 SQL 선언 섹션에 선언될 수 있습니다. SQL 선언 섹션에 이 변수가 모두 선언되지 않은 경우 프로그램 어딘가에 SQLCODE 이름의 변수가 선언된 것으로 간주됩니다. LANGLEVEL SQL92E를 사용하면 프로그램에는 INCLUDE SQLCA문을 포함하면 안됩니다.

SQLCODE는 각 SQL문이 실행된 후에 데이터베이스 관리 프로그램이 설정합니다. 모든 데이터베이스 관리 프로그램은 다음과 같이 ISO/ANSI SQL 표준을 준수합니다.

- SQLCODE = 0 및 SQLWARN0가 공백인 경우 실행이 성공입니다.
- SQLCODE = 100, ″no data″인 경우, 예를 들어 결과 테이블의 마지막 행 뒤에 커서가 있기 때문에 FETCH문이 데이터를 리턴하지 않습니다.
- SQLCODE > 0 및 not = 100인 경우 실행은 성공했으나 경고가 발생합니다.
- SQLCODE = 0 및 SQLWARN0 = 'W'인 경우 실행은 성공했지만 하나 이상의 경고 표시기가 설정되었습니다.
- SQLCODE < 0인 경우 실행은 실패합니다.

0 및 100 이외의 SQLCODE 값은 제품에 고유하게 적용되는 값입니다.

SQLSTATE는 각 SQL문이 실행된 후에 데이터베이스 관리 프로그램이 설정합니다. 응용프로그램은 SQLCODE가 아닌 SQLSTATE를 테스트하여 SQL문 실행을 점검할 수 있습니다. SQLSTATE는 공통 오류 조건에 대한 공통 코드를 제공합니다. 응용프로그램은 특정 오류나 오류 클래스에 대해 테스트할 수 있습니다. 코드 스킴은 모든 IBM 데이터베이스 관리 프로그램에 대해 동일하며 ISO/ANSI SQL92 표준을 기반으로 합니다.

SQL 주석

정적 SQL문에는 호스트 언어 또는 SQL 주석이 포함될 수 있습니다. 동적 SQL문에는 SQL 주석이 포함될 수 있습니다. SQL 주석에는 두 가지 유형이 있습니다.

단순 주석

단순 주석은 두 개의 연속 하이픈으로 시작되고 행 끝에서 끝납니다.

대괄호로 표시된 주석

브래킷(Bracketed) 주석은 /*로 시작하여 */로 끝납니다.

단순 주석에는 다음과 같은 규칙이 적용됩니다.

- 두 개의 하이픈은 동일한 행에 있어야 하며 사이에 공백이 있으면 안됩니다.
- 단순 주석은 공백이 유효할 때마다 시작됩니다(분리문자 토큰 내 또는 'EXEC' 및 'SQL' 사이는 제외).
- 단순 주석은 다음 행으로 계속될 수 없습니다.
- COBOL에서 하이픈 앞에는 공백이 있어야 합니다.

대괄호로 표시되는 주석에는 다음 규칙이 적용됩니다.

- /*는 동일한 행에 있어야 하며 사이에 공백이 있으면 안 됩니다.
- */는 동일한 행에 있어야 하며 사이에 공백이 있으면 안 됩니다.
- 대괄호로 표시되는 주석은 공백이 유효할 때마다 시작됩니다(분리문자 토큰 내 또는 'EXEC' 및 'SQL' 사이는 제외).
- 대괄호로 표시되는 주석은 다음 행으로 계속될 수 있습니다.

예 1: 이 예에서는 명령문에 단순 주석을 포함시키는 방법에 대해 설명합니다.

```
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
```

예 2: 이 예에서는 명령문에 대괄호로 표시되는 주석을 포함시키는 방법에 대해 설명합니다.

```
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
```

SQL 제어 명령문 정보

SQL 제어 명령문(SQL PL(SQL 프로시저 언어)이라고도 함)은 구조화된 프로그래밍 언어로 프로그램을 작성하는 것과 유사한 방식으로 SQL을 사용할 수 있게 하는 SQL 문입니다. SQL 제어 명령문은 논리 흐름 제어, 선언, 변수 설정, 경고 및 예외 처리 등의 기능을 제공합니다. 일부 SQL 제어 명령문은 다른 중첩된 SQL문을 포함합니다. SQL 제어 명령문은 루틴, 트리거 또는 복합 명령문 내용에 사용할 수 있습니다.

SQL 매개변수, SQL 변수 및 전역 변수에 대한 참조

SQL 매개변수, SQL 변수 및 전역 변수는 표현식이나 변수를 지정할 수 있는 SQL 프로시저 명령문의 모든 위치에서 참조할 수 있습니다. SQL 루틴, SQL 트리거 또는 동적 복합 명령문에는 호스트 변수를 지정할 수 없습니다. SQL 매개변수는 루틴의 모든 위치에서 참조될 수 있고 루틴 이름으로 규정될 수 있습니다. SQL 변수는 변수가 선언된 복합 명령문의 모든 위치에서 참조될 수 있고 복합 명령문을 시작할 때 지정한 레이블 이름으로 규정될 수 있습니다. SQL 매개변수나 SQL 변수에 행 데이터 유형이 있는 경우 SQL 매개변수 또는 SQL 변수가 참조될 수 있는 모든 위치에서 필드를 참조할 수 있습니다. 전역 변수는 표현식이 결정적일 필요가 없는 경우에는 모든 표현식 내에서 참조할 수 있습니다. 다음 시나리오의 경우 전역 변수 사용을 방지하는 결정적 표현식이 필요합니다.

- 점검 제한조건

- 생성된 컬럼 정의
- MQT 즉시 새로 고침

모든 SQL 매개변수, SQL 변수, 행 변수 필드 및 전역 변수는 널(NULL) 입력이 가능합니다. SQL 루틴의 SQL 매개변수, SQL 변수, 행 변수 필드 또는 전역 변수 이름은 루틴에서 참조되는 테이블이나 뷰의 컬럼 이름과 같은 이름일 수 있습니다. SQL 변수 또는 행 변수 필드 이름은 동일한 루틴에 선언된 다른 SQL 변수 또는 행 변수 필드 이름과 동일할 수 있습니다. 이는 다른 복합 명령문에 두 개의 SQL 변수를 선언할 때 발생할 수 있습니다. SQL 변수 선언을 포함하는 복합 명령문은 해당 변수의 범위를 판별합니다. 자세한 정보는 『복합 SQL(프로시저)』을 참조하십시오.

SQL 루틴에 있는 SQL 변수 또는 SQL 매개변수의 이름은 특정 SQL문에 사용된 ID의 이름과 동일할 수 있습니다. 이름을 규정하지 않은 경우에는 다음 규칙을 통해 이름이 ID를 참조하는지 아니면 SQL 매개변수나 SQL 변수를 참조하는지 알 수 있습니다.

- SET PATH 및 SET SCHEMA문에서는 SQL 매개변수 또는 SQL 변수로서 이름을 점검합니다. SQL 변수 또는 SQL 매개변수로서 찾지 못한 경우, 이름을 ID로 사용합니다.
- CONNECT, DISCONNECT, RELEASE 및 SET CONNECTION문에서는 이름을 ID로 사용합니다.

동일한 이름은 명시적으로 규정되어야 합니다. 이름을 명확히 규정한다는 것은 이름이 컬럼, SQL 변수, SQL 매개변수, 행 변수 필드또는 전역 변수를 참조하는지 여부를 나타냅니다. 이름이 규정되어 있지 않거나 규정은 되었지만 아직 애매한 경우, 다음 규칙을 통해 이름이 컬럼, SQL 변수, SQL 매개변수 또는 전역 변수를 참조하는지 알 수 있습니다.

- SQL 루틴 본문에 지정된 테이블이나 뷰가 루틴을 작성할 때 있었던 경우에는 이름이 먼저 컬럼 이름으로 지정됩니다. 컬럼으로 찾지 못한 경우, 복합 명령문의 SQL 변수로서 점검한 다음 SQL 매개변수로서 점검하고 마지막으로 전역 변수로 점검합니다.
- 루틴을 작성할 때 참조된 테이블 또는 뷰가 존재하지 않으면 먼저 이름을 복합 명령문의 SQL 변수로서 점검한 다음 SQL 매개변수로서 점검하고 전역 변수로 점검합니다. 참조를 포함하는 복합 명령문 또는 해당 복합 명령문이 중첩된 복합 명령문 내에 변수를 선언할 수 있습니다. 두 개의 SQL 변수가 동일한 범위 내에 있고 동일한 이름을 갖는 경우(이는 다른 복합 명령문에 선언될 경우에 발생함) 가장 안쪽의 복합 명령문에 선언된 SQL 변수가 사용됩니다. SQL 변수나 SQL 매개변수로 찾지 못하면 컬럼으로 간주됩니다.

레이블에 대한 참조

대부분의 SQL 프로시저 명령문에 레이블을 지정할 수 있습니다. 레이블을 정의하는 명령문을 포함하는 복합 명령문이 해당 레이블 이름의 범위를 판별합니다. 레이블 이름은 해당 복합 명령문 내에 중첩된 복합 명령문에 정의된 모든 레이블을 비롯하여 레이블이 정의된 복합 명령문 내에서 고유해야 합니다(SQLSTATE 42734). 레이블은 복합 명령문 자체에 지정된 레이블과 동일하거나(SQLSTATE 42734) SQL 프로시저 명령문을 포함하는 루틴의 이름과 동일해서는 안 됩니다(SQLSTATE 42734).

레이블 이름은 해당 복합 명령문 내에 중첩된 모든 복합 명령문을 비롯하여 레이블 이름이 정의되어 있는 복합 명령문 내에서만 참조될 수 있습니다. 레이블을 사용하여 SQL 변수의 이름을 규정하거나 GOTO, LEAVE 또는 ITERATE문의 목표로 지정할 수 있습니다.

SQL 조건 이름에 대한 참조

SQL 조건의 이름은 동일한 루틴에 선언된 다른 SQL 조건의 이름과 동일할 수 있습니다. 이는 다른 복합 명령문에 두 개의 SQL 조건을 선언할 때 발생할 수 있습니다. SQL 조건 이름 선언을 포함하는 복합 명령문은 해당 조건 이름의 범위를 판별합니다. 조건 이름은 조건 이름이 선언된 복합 명령문 내에서 고유해야 합니다. 단, 해당 복합 명령문 내에 중첩된 복합 명령문 내의 선언은 제외됩니다(SQLSTATE 42734). 조건 이름은 해당 복합 명령문 내에 중첩된 모든 복합 명령문을 비롯하여 조건 이름이 정의되어 있는 복합 명령문 내에서만 참조될 수 있습니다. 조건 이름에 대한 참조가 있는 경우, 가장 안쪽의 복합 명령문에 선언된 조건이 사용되는 조건입니다. 자세한 정보는 『복합 SQL(프로시저)』을 참조하십시오.

SQL문 이름에 대한 참조

SQL문의 이름은 동일한 루틴에 선언된 다른 SQ문의 이름과 동일할 수 있습니다. 이는 다른 복합 명령문에 두 개의 SQL문을 선언할 때 발생할 수 있습니다. SQL문 이름 선언을 포함하는 복합 명령문은 해당 명령문 이름의 범위를 판별합니다. 명령문 이름은 명령문 이름이 선언된 복합 명령문 내에서 고유해야 합니다. 단, 해당 복합 명령문 내에 중첩된 복합 명령문 내의 선언은 제외됩니다(SQLSTATE 42734). 명령문 이름은 해당 복합 명령문 내에 중첩된 모든 복합 명령문을 비롯하여 명령문 이름이 정의되어 있는 복합 명령문 내에서만 참조될 수 있습니다. 명령문 이름에 대한 참조가 있는 경우, 가장 안쪽의 복합 명령문에 선언된 명령문이 사용되는 명령문입니다. 자세한 정보는 『복합 SQL(프로시저)』을 참조하십시오.

SQL 커서 이름에 대한 참조

커서 이름에는 선언된 커서 이름 및 커서 변수 이름이 포함됩니다.

SQL 커서의 이름은 동일한 루틴에 선언된 다른 SQL 커서의 이름과 동일할 수 있습니다. 이는 다른 복합 명령문에 두 개의 SQL 커서를 선언할 때 발생할 수 있습니다.

SQL 커서 선언을 포함하는 복합 명령문은 해당 커서 이름의 범위를 판별합니다. 커서 이름은 커서 이름이 선언된 복합 명령문 내에서 고유해야 합니다. 단, 해당 복합 명령문 내에 중첩된 복합 명령문 내의 선언은 제외됩니다(SQLSTATE 42734). 커서 이름은 해당 복합 명령문 내에 중첩된 모든 복합 명령문을 비롯하여 커서 이름이 정의되어 있는 복합 명령문 내에서만 참조될 수 있습니다. 커서 이름에 대한 참조가 있는 경우, 가장 안쪽의 복합 명령문에 선언된 커서가 사용되는 커서입니다. 자세한 정보는 『복합 SQL(프로시저)』을 참조하십시오.

커서 변수에 지정된 커서 컨스트럭터가 로컬 SQL 변수에 대한 참조를 포함하면, 커서 변수를 사용하는 OPEN문은 로컬 SQL 변수가 선언되는 범위 내에 있어야 합니다.

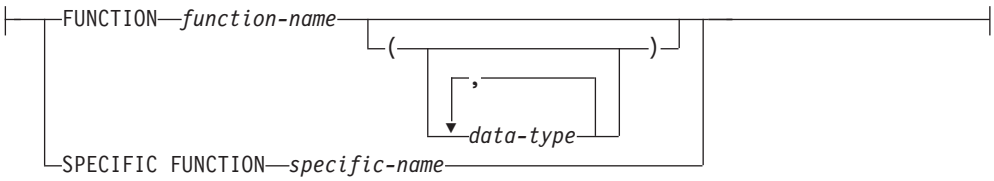
함수, 메소드 및 프로시저 지정자

다음 섹션에서는 모듈에 정의되지 않은 함수, 메소드 또는 프로시저를 고유하게 식별하는 데 사용되는 구문을 부분별로 설명합니다.

함수 지정자

함수 지정자는 단일 함수를 고유하게 식별합니다. 함수 지정자는 일반적으로 함수의 DDL 문에 표시됩니다(예: DROP 또는 ALTER). 함수 지정자는 모듈 함수를 식별하면 안됩니다(SQLSTATE 42883).

function-designator:



FUNCTION *function-name*

특정 함수를 식별하며, 이름으로 *function-name*을 갖는 함수 인스턴스가 스키마에 정확히 한 개만 있을 경우에만 유효합니다. 식별된 함수에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 이름이 지정된 스키마 또는 내재적 스키마에 이 이름을 가진 함수가 없으면, 오류가 발생합니다(SQLSTATE 42704). 이름이 지정된 스키마나 내재적 스키마에 함수의 인스턴스가 둘 이상 있는 경우, 오류가 발생합니다(SQLSTATE 42725).

FUNCTION *function-name (data-type,...)*

함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 결정 알고리즘은 사용되지 않습니다.

function-name

함수 이름을 지정합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

(data-type,...)

값은 CREATE FUNCTION문의(해당 위치에) 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합이 특정 함수 인스턴스를 식별하는 데 사용됩니다.

데이터 유형이 규정되어 있지 않은 경우, SQL 경로에서 스키마를 검색하여 유형 이름이 해석됩니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE FUNCTION 문에 지정된 값과 정확히 일치해야 합니다.

0 < *n* < 25는 REAL을 의미하고, 24 < *n* < 54는 DOUBLE을 의미하므로, FLOAT(*n*) 유형은 *n*에 정의된 값과 일치하지 않아도 됩니다. 일치는 유형이 REAL인지 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 가진 함수가 이름이 지정된 스키마나 내재적 스키마의 유형으로 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

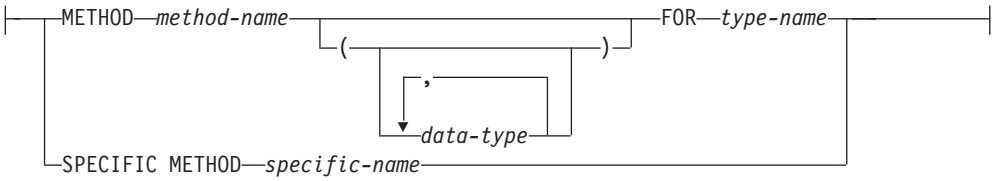
SPECIFIC FUNCTION *specific-name*

함수 작성 시 지정되거나 디폴트로 설정된 이름을 사용하여 특정 사용자 정의 함수(UDF)를 식별합니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 함수 인스턴스를 식별해야 합니다. 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

메소드 지정자

메소드 지정자는 단일 메소드를 고유하게 식별합니다. 메소드 지정자는 대개 메소드에 대한 DDL문(예: DROP 또는 ALTER)에 나타납니다.

method-designator:



METHOD *method-name*

특정 메소드를 식별하며, *type-name* 유형에 대해 이름이 *method-name*인 메소드 인스턴스가 정확히 한 개만 있을 경우에만 유효합니다. 식별된 메소드에는 그에 대

해 정의된 임의 수의 매개변수가 있을 수 있습니다. 유형에 대해 이 이름의 메소드가 없는 경우에는 오류가 발생합니다(SQLSTATE 42704). 유형에 대해 메소드의 인스턴스가 둘 이상 있는 경우에는 오류가 발생합니다(SQLSTATE 42725).

METHOD *method-name (data-type,...)*

메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 분석 알고리즘은 사용되지 않습니다.

method-name

type-name 유형에 대한 메소드 이름을 지정합니다.

(data-type,...)

값은 CREATE TYPE문의 해당 위치에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합은 특정 메소드 인스턴스를 식별하는 데 사용됩니다.

데이터 유형이 규정되어 있지 않은 경우, SQL 경로에서 스키마를 검색하여 유형 이름이 해석됩니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE TYPE문에 지정된 값과 정확히 일치해야 합니다.

$0 < n < 25$는 REAL을 의미하고, $24 < n < 54$는 DOUBLE을 의미하므로, FLOAT(*n*) 유형은 *n*에 정의된 값과 일치하지 않아도 됩니다. 일치는 유형이 REAL인지 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 가진 메소드가 이름이 지정된 스키마나 내재적 스키마의 유형으로 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

FOR *type-name*

지정된 메소드가 연관될 유형의 이름을 지정합니다. 이 이름은 카탈로그에 기술되어 있는 유형을 식별해야 합니다(SQLSTATE 42704). 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

SPECIFIC METHOD *specific-name*

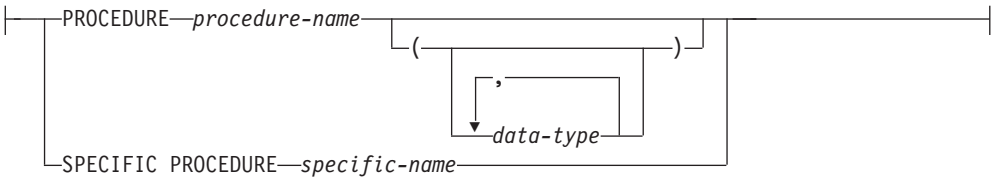
메소드 작성시 디폴트로 설정된 이름이나 지정된 이름을 사용하여 특정 메소드를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않

은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 메소드 인스턴스를 식별해야 합니다. 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

프로시저 지정자

프로시저 지정자는 단일 프로시저를 고유하게 식별합니다. 프로시저 지정자는 일반적으로 프로시저의 DDL문에 표시됩니다(예: DROP 또는 ALTER). 프로시저 지정자는 모듈 프로시저를 식별하면 안됩니다(SQLSTATE 42883).

procedure-designator:



PROCEDURE *procedure-name*

특정 프로시저를 식별하며, 이름으로 *procedure-name*인 프로시저 인스턴스가 스키마에 정확히 한 개만 있을 경우에만 유효합니다. 식별된 프로시저에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 이름이 지정된 스키마 또는 내재적 스키마에 이 이름의 프로시저가 존재하지 않는 경우, 오류(SQLSTATE 42704)가 발생합니다. 이름이 지정된 또는 내재적 스키마에 하나 이상의 프로시저 인스턴스가 있는 경우, 오류가 발생합니다(SQLSTATE 42725).

PROCEDURE *procedure-name (data-type,...)*

프로시저를 고유하게 식별하는 프로시저 시그니처를 제공합니다. 프로시저 분석 알고리즘은 사용하지 않습니다.

procedure-name

프로시저 이름을 지정합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

(data-type,...)

값은 CREATE PROCEDURE문의(해당 위치에) 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합이 특정 프로시저 인스턴스를 식별하는 데 사용됩니다.

데이터 유형이 규정되어 있지 않은 경우, SQL 경로에서 스키마를 검색하여 유형 이름이 해석됩니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE PROCEDURE 문에 지정된 값과 정확히 일치해야 합니다.

$0 < n < 25$는 REAL을 의미하고, $24 < n < 54$는 DOUBLE을 의미하므로, FLOAT(*n*) 유형은 *n*에 정의된 값과 일치하지 않아도 됩니다. 일치는 유형이 REAL인지 DOUBLE인지에 따라 발생합니다.

지정된 시그니처의 프로시저가 이름 지정된 스키마나 내재적 스키마에 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC PROCEDURE *specific-name*

프로시저 작성 시 지정되거나 디폴트로 설정된 이름을 사용하여 특정 프로시저를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 프로시저 인스턴스를 식별해야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42704).

ALLOCATE CURSOR

ALLOCATE CURSOR문은 결과 세트 로케이터 변수로 식별되는 결과 세트에 대한 커서를 할당합니다. 결과 세트 로케이터 변수에 대한 자세한 정보는 ASSOCIATE LOCATORS문의 설명을 참조하십시오.

호출

이 명령문은 SQL 프로시저에만 임베드될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

권한 부여

필요한 권한 없음

구문

설명

cursor-name

커서 이름을 지정합니다. 이름은 소스 SQL 프로시저에서 이미 선언된 커서를 식별해서는 안됩니다(SQLSTATE 24502).

CURSOR FOR RESULT SET *rs-locator-variable*

결과 세트 로케이터 변수 선언에 대한 규칙에 따라서 소스 SQL 프로시저에서 선언된 결과 세트 로케이터 이름을 지정합니다. SQL 변수 선언에 대한 자세한 정보는 『복합 SQL(프로시저)문』을 참조하십시오.

결과 세트 로케이터 변수는 ASSOCIATE LOCATORS SQL문이 리턴하는 대로 유효한 결과 세트 로케이터 값을 포함해야 합니다(SQLSTATE 0F001).

규칙

- 할당된 커서를 사용할 때 다음 규칙이 적용됩니다.
  - 할당된 커서를 OPEN문으로 열 수 없습니다(SQLSTATE 24502).
  - 할당된 커서를 위치 지정된 UPDATE 또는 DELETE문에서 사용할 수 없습니다(SQLSTATE 42828).
  - 할당된 커서는 CLOSE문으로 닫을 수 있습니다. 할당된 커서를 닫으면 연관된 커서가 닫힙니다.
  - 각 결과 세트에 커서를 하나만 할당할 수 있습니다.
- 할당된 커서는 롤백 조작, 내재된 닫기 또는 명시적 닫기까지 지속됩니다.
- 커미트 조작은 WITH HOLD가 정의되지 않은 할당된 커서를 삭제합니다.

- 할당된 커서를 삭제하면 SQL 프로시저에서 연관된 커서가 닫힙니다.

예:

이 SQL 프로시저 예는 커서 C1을 작성하고 결과 세트 로케이터 변수 LOC1 및 SQL 프로시저가 리턴하는 관련 결과 세트와 연관시킵니다.

```
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
```

ALTER AUDIT POLICY

ALTER AUDIT POLICY문은 현재 서버에서 감사 규정의 정의를 수정합니다.

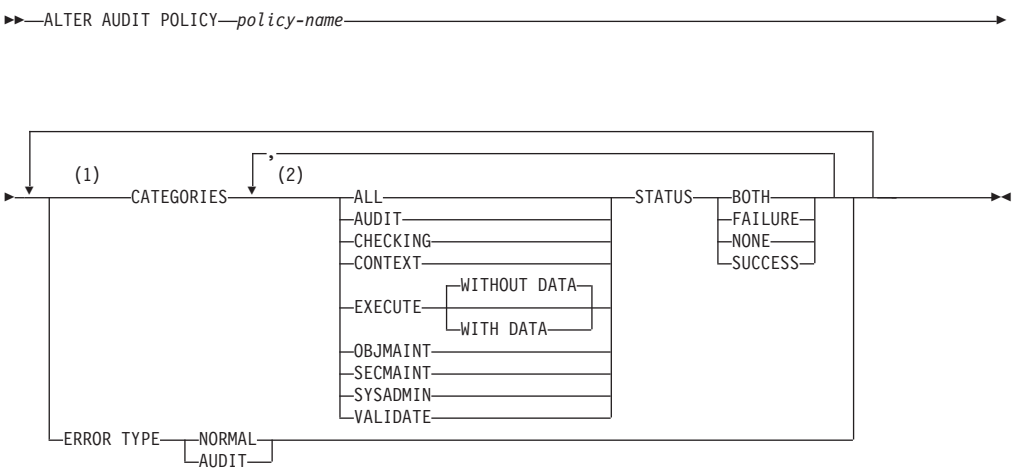
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



주:

1 CATEGORIES 및 ERROR TYPE 절 각각은 최대 한 번 지정할 수 있습니다(SQLSTATE 42614).

2 각 범주는 최대 한 번 지정할 수 있으며(SQLSTATE 42614), ALL이 지정된 경우 다른 어떤 범주도 지정할 수 없습니다(SQLSTATE 42601).

설명

policy-name

변경될 감사 규정을 식별합니다. 이는 한 부분으로 된 이름입니다. SQL ID(일반 또는 구분 ID)입니다. 이름은 현재 서버의 기존 감사 규정을 고유하게 식별해야 합니다(SQLSTATE 42704).

CATEGORIES

상태 값이 지정된 하나 이상의 감사 범주 목록입니다. ALL을 지정하지 않은 경우 명시적으로 지정되지 않은 범주의 STATUS는 변경되지 않고 유지됩니다.

ALL
모든 범주를 동일한 상태로 설정합니다. EXECUTE 범주는 WITHOUT DATA입니다.

AUDIT
감사 설정이 변경될 때 또는 감사 로그에 액세스할 때 레코드를 생성합니다.

CHECKING
데이터베이스 오브젝트 또는 함수에 액세스하거나 처리하기 위한 권한 부여 검사 시도 중 레코드를 생성합니다.

CONTEXT
데이터베이스 조작이 수행될 때 조작 컨텍스트를 표시하기 위해 레코드를 생성합니다.

EXECUTE
SQL문의 실행을 표시하기 위해 레코드를 생성합니다.

WITHOUT DATA 또는 WITH DATA
호스트 변수나 매개변수 표시문자에 대해 제공된 입력 데이터 값이 EXECUTE 범주 일부로 로그해야 하는지 여부를 지정합니다.

WITHOUT DATA
호스트 변수와 매개변수 표시문자에 대해 제공된 입력 데이터 값이 EXECUTE 범주 일부로 로그해야 하는지 여부를 지정합니다.

WITH DATA
호스트 변수나 매개변수 표시문자에 대해 제공된 입력 데이터 값이 EXECUTE 범주 일부로 로그됨을 지정합니다. 모든 입력 값이 로그되는 것은 아닙니다. 특히 LOB, LONG, XML 및 구조화 유형 매개변수는 널(NULL) 값으로 표시됩니다. 날짜, 시간 및 시간소인 필드는 ISO 형식으로 로그됩니다. 입력 데이터 값은 로그되기 전에 데이터베이스 코드 페이지로 변환됩니다. 코드 페이지 변환이 실패하는 경우 어떤 오류도 리턴되지 않으며 변환되지 않은 데이터가 로그됩니다.

OBJMAINT
데이터 오브젝트가 작성되거나 삭제될 때 레코드를 생성합니다.

SECMAINT
오브젝트 특권, 데이터베이스 특권 또는 DBADM 권한이 부여되거나 취소될 때 레코드를 생성합니다. 데이터베이스 관리 프로그램 보안 구성 매개변수 **sysadm_group**, **sysctrl_group** 또는 **sysmaint_group**이 수정될 때도 레코드가 생성됩니다.

SYSADMIN
SYSADM, SYSMAINT 또는 SYSCTRL 권한이 필요한 조작이 수행될 때 레코드를 생성합니다.

VALIDATE
사용자가 인증될 때 또는 사용자에 관련된 시스템 보안 정보가 검색될 때 레코드를 생성합니다.

STATUS
지정된 범주에 대한 상태를 지정합니다.

BOTH
성공 및 실패 이벤트를 감사합니다.

FAILURE
실패 이벤트만 감사합니다.

SUCCESS
성공 이벤트만 감사합니다.

NONE
이 범주의 이벤트는 감사하지 않습니다.

ERROR TYPE
감사 오류를 리턴할 것인지 또는 무시할 것인지 여부를 지정합니다.

NORMAL
감사에서 생성된 오류가 무시되고 수행되는 조작과 연관된 오류에 대한 SQLCODE만 응용프로그램에 리턴됩니다.

AUDIT
감사 기능 자체에서 발생하는 오류를 포함한 모든 오류가 응용프로그램에 리턴됩니다.

규칙

- AUDIT 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다 (SQLSTATE 5U021). AUDIT 독점 SQL문은 다음과 같습니다.
  - AUDIT
  - CREATE AUDIT POLICY, ALTER AUDIT POLICY 또는 DROP(AUDIT POLICY)
  - 역할 또는 트러스트된 컨텍스트가 감사 규정과 연관되는 경우 DROP(ROLE) 또는 DROP(TRUSTED CONTEXT)
- AUDIT 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 모든 데이터베이스 파티션 사이에서 한 번에 단 하나의 언커미트된 AUDIT 독점 SQL 문만 허용됩니다. 언커미트된 AUDIT 독점 SQL문이 실행 중인 경우, 연속 AUDIT 독점 SQL문은 현재 AUDIT 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 발행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.
- 변경되는 감사 규정이 현재 데이터베이스 오브젝트와 연관되는 경우, 변경의 영향을 받는 응용프로그램의 다음 작업 단위까지 변경사항은 적용되지 않습니다. 예를 들어, 데이터베이스에 대해 감사 규정을 사용 중인 경우 현재 작업 단위는 해당 작업 단위의 COMMIT 또는 ROLLBACK 문이 완료될 때까지 규정 변경을 보지 못합니다.

예 :

감사 규정 DBAUDPRF의 SECMAINT, CHECKING 및 VALIDATE 범주를 성공 및 실패 모두를 감사하도록 변경하십시오.

```
ALTER AUDIT POLICY DBAUDPRF
  CATEGORIES SECMAINT STATUS BOTH,
             CHECKING STATUS BOTH,
             VALIDATE STATUS BOTH
```

ALTER BUFFERPOOL

ALTER BUFFERPOOL문은 다음을 수행할 때 사용됩니다.

- 모든 데이터베이스 파티션 또는 단일 데이터베이스 파티션의 버퍼 풀 *크기* 수정
- 버퍼 풀의 자동 *크기* 조정 기능 사용 가능하게 하기 및 불가능하게 하기
- 해당 버퍼 풀 정의를 새 데이터베이스 파티션 그룹에 추가
- 블록 기반 입출력에 대한 버퍼 풀의 블록 영역 수정

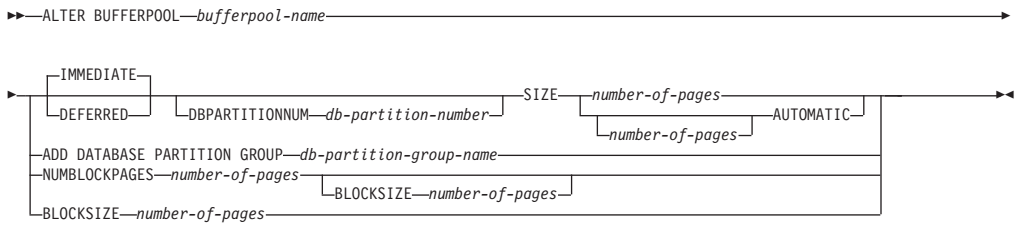
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 SYSCTRL 권한을 포함해야 합니다.

구문



설명

bufferpool-name

버퍼 풀의 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이것은 SQL ID(일반 또는 분리 ID)입니다. 카탈로그에 기술된 버퍼 풀이어야 합니다.

IMMEDIATE 또는 DEFERRED

버퍼 풀 *크기*가 즉시 변경되는지 표시합니다.

IMMEDIATE

버퍼 풀 *크기*가 즉시 변경됩니다. 새 스페이스를 할당하기 위해 데이터베이스 공유 메모리에 예약된 스페이스가 충분하지 않으면(SQLSTATE 01657) 이 명령문이 DEFERRED로 실행됩니다.

DEFERRED

데이터베이스가 다시 활성화되면 버퍼 풀의 *크기*가 변경됩니다. (모든 응용프로

그램이 데이터베이스에서 연결을 끊어야 합니다.) DB2 데이터베이스는 활성 시에 필요한 메모리를 시스템에서 할당하므로 예약 메모리 스페이스가 필요하지 않습니다.

DBPARTITIONNUM *db-partition-number*
버퍼 풀 크기가 수정될 데이터베이스 파티션을 지정합니다. 예외 항목이 SYSCAT.BUFFERPOOLDBPARTITIONS 시스템 카탈로그 뷰에 작성됩니다. 데이터베이스 파티션은 버퍼 풀에 대한 데이터베이스 파티션 그룹 중 하나여야 합니다(SQLSTATE 42729). 이 절을 지정하지 않으면 버퍼 풀 크기가 SYSCAT.BUFFERPOOLDBPARTITIONS에서 예외 항목이 있는 경우를 제외하고 모든 데이터베이스 파티션에서 수정됩니다.

SIZE
버퍼 풀의 크기를 새로 지정하거나, 해당 버퍼 풀의 자체 성능 조정 기능을 사용 가능하게 하거나 불가능하게 합니다.

number-of-pages
새 버퍼 풀 크기에 대한 페이지 수. 버퍼 풀이 자체 성능 조정 버퍼 풀이고 SIZE *number-of-pages* 절이 지정된 경우, 변경 조작은 해당 버퍼 풀의 자체 성능 조정 기능을 사용하지 않습니다.

AUTOMATIC
해당 버퍼 풀의 자체 성능 조정 기능을 사용 가능하게 합니다. 데이터베이스 관리 프로그램은 워크로드 요구사항에 맞춰 버퍼 풀 크기를 조정합니다. AUTOMATIC이 지정되면 DBPARTITIONNUM절을 지정할 수 없습니다 (SQLSTATE 42601).

ADD DATABASE PARTITION GROUP *db-partition-group-name*
버퍼 풀 정의를 적용할 수 있는 데이터베이스 파티션 그룹 목록에 이 데이터베이스 파티션 그룹을 추가합니다. 아직 버퍼 풀이 정의되지 않은 데이터베이스 파티션 그룹에 있는 데이터베이스 파티션의 경우, 버퍼 풀은 버퍼 풀에 대해 지정된 디폴트 크기를 사용하여 데이터베이스 파티션에서 작성됩니다. *db-partition-group-name*의 테이블 스페이스가 이 버퍼 풀을 지정할 수도 있습니다. 데이터베이스 파티션 그룹이 현재 데이터베이스에 있어야 합니다(SQLSTATE 42704).

NUMBLOCKPAGES *number-of-pages*
블록 기반 영역에 있어야 할 페이지 수를 지정합니다. 이 페이지 수는 버퍼 풀에 대한 총 페이지 수의 98%를 넘지 않아야 합니다(SQLSTATE 54052). 0 값을 지정하면 블록화 입출력을 사용할 수 없게 됩니다. 실제로 사용되는 NUMBLOCKPAGES 값은 BLOCKSIZE의 배수입니다.

BLOCKSIZE *number-of-pages*
블록에 페이지 수를 지정합니다. 블록 크기는 2 - 256 사이의 값이어야 합니다 (SQLSTATE 54053). 디폴트값은 32입니다.

주

- 버퍼 풀 *크기*만 동적으로 (즉시) 변경할 수 있습니다. 다른 변경사항은 모두 지연되어, 데이터베이스를 다시 활성화한 후에만 적용됩니다.
- 명령문이 지연되어 실행되는 경우, 다음과 같은 사항이 적용됩니다. 버퍼 풀 정의가 트랜잭션이 될 수 있으며, 버퍼 풀 정의에 대한 변경사항은 커미트시 카탈로그 테이블에 반영되며, 다음 번에 데이터베이스를 시작할 때 실제 버퍼 풀에 대한 변경사항이 적용됩니다. 버퍼 풀에 대한 현재 속성은 그때까지 존재하며, 그 동안에는 버퍼 풀에 어떤 영향도 미치지 않습니다. 새 데이터베이스 파티션 그룹의 테이블 스페이스에 작성된 테이블은 버퍼 풀 디폴트값을 사용합니다. 이 키워드가 적용될 때 명령문은 디폴트로 IMMEDIATE입니다.
- 데이터베이스 관리 프로그램과 응용프로그램의 나머지 요구사항에 대해서뿐만 아니라 전체 버퍼 풀에 대해 머신에 충분한 실제 메모리가 있어야 합니다.
- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.

ALTER DATABASE PARTITION GROUP

ALTER DATABASE PARTITION GROUP문은 다음을 수행하는 데 사용됩니다.

- 하나 이상의 데이터베이스 파티션을 데이터베이스 파티션 그룹에 추가
- 하나 이상의 데이터베이스 파티션을 데이터베이스 파티션 그룹에서 삭제

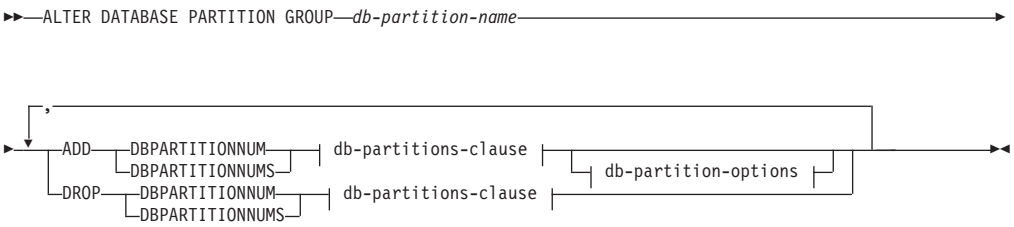
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

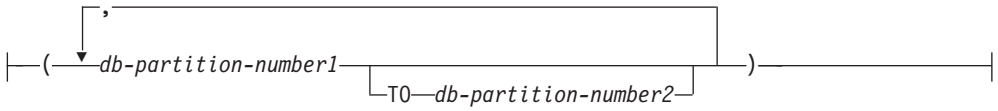
권한 부여

명령문의 권한 부여 ID는 SYSCTRL 또는 SYSADM 권한을 가지고 있어야 합니다.

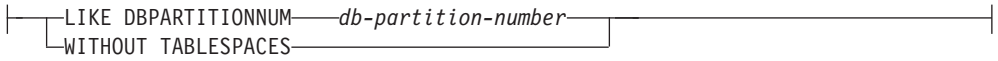
구문



db-partitions-clause:



db-partition-options:



설명

db-partition-name

데이터베이스 파티션 그룹의 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이것은 SQL ID(일반 또는 분리 ID)입니다. 카탈로그에 기술된 데이터베이스 파티션 그룹이어야 합니다. IBMCATGROUP과 IBMTEMPGROUP은 지정할 수 없습니다(SQLSTATE 42832).

ADD DBPARTITIONNUM

데이터베이스 파티션 그룹에 추가할 특정 데이터베이스 파티션을 지정합니다. DBPARTITIONNUMS는 DBPARTITIONNUM과 동의어입니다. 지정된 데이터베이스 파티션은 데이터베이스 파티션 그룹에 정의되어 있지 않아야 합니다 (SQLSTATE 42728).

DROP DBPARTITIONNUM

데이터베이스 파티션 그룹에서 삭제할 특정 데이터베이스 파티션을 지정합니다. DBPARTITIONNUMS는 DBPARTITIONNUM과 동의어입니다. 지정된 데이터베이스 파티션은 데이터베이스 파티션 그룹에 정의되어 있어야 합니다(SQLSTATE 42729).

db-partitions-clause

데이터베이스 파티션 그룹에 추가 또는 삭제할 데이터베이스 파티션을 지정합니다.

db-partition-number1

특정 데이터베이스 파티션 번호를 지정하십시오.

TO *db-partition-number2*

데이터베이스 파티션 번호의 범위를 지정하십시오. *db-partition-number2* 값은 *db-partition-number1* 값보다 크거나 같아야 합니다(SQLSTATE 428A9).

db-partition-options

LIKE DBPARTITIONNUM *db-partition-number*

데이터베이스 파티션 그룹에 있는 기존 테이블 스페이스에 대한 컨테이너가 지정된 *db-partition-number*의 컨테이너와 같도록 지정합니다. 지정된 데이터베이스 파티션은 이 명령문에 우선하는 데이터베이스 파티션 그룹에 있는 파티션이어야 하며, 같은 명령문의 DROP DBPARTITIONNUM절에는 포함되지 않아야 합니다.

자동 스토리지를 사용하도록 지정된 테이블 스페이스(즉 CREATE TABLESPACE문의 MANAGED BY AUTOMATIC STORAGE절로 작성된 테이블 스페이스 또는 MANAGED BY절이 지정되지 않은 테이블 스페이스)의 경우, 컨테이너는 지정된 파티션의 컨테이너와 반드시 일치할 필요는 없습니다. 대신에 컨테이너는 데이터베이스에 연결된 스토리지 경로를 기반으로 하는 데이터베이스 관리 프로그램에 의해 자동으로 할당되고, 사용되고 있는 것과 동일한 컨테이너를 결과로 생성할 수도 있고 그렇지 않을 수도 있습니다. 각 테이블 스페이스의 크기는 테이블 스페이스가 작성되었을 때 지정되었던 최초의 크기에 기반하고, 지정된 파티션에 있는 테이블 스페이스의 현재 크기와 일치하지 않을 수도 있습니다.

WITHOUT TABLESPACES

데이터베이스 파티션 그룹에 있는 기존 테이블 스페이스에 대한 컨테이너가 새로 추가된 데이터베이스 파티션에 작성되지 않도록 지정합니다. 이 데이터베이

스 파티션 그룹에 정의된 테이블 스페이스에 사용할 컨테이너를 정의하려면 *db-partitions-clause*를 사용하는 ALTER TABLESPACE문을 사용해야 합니다. 이 옵션을 지정하지 않을 경우, 데이터베이스 파티션 그룹에 정의된 각 테이블 스페이스에 대해 새로 추가된 데이터베이스 파티션에 디폴트 컨테이너가 지정됩니다.

이 옵션은 자동 스토리지를 사용하도록 지정된 테이블 스페이스(즉 CREATE TABLESPACE문의 MANAGED BY AUTOMATIC STORAGE절로 작성된 테이블 스페이스 또는 MANAGED BY절이 지정되지 않은 테이블 스페이스)에 대해 무시됩니다. 해당 테이블 스페이스에 대한 컨테이너 작성을 지연할 수 없습니다. 컨테이너는 데이터베이스에 연결된 스토리지 경로를 기반으로 하는 데이터베이스 관리 프로그램에 의해 자동으로 할당됩니다. 각 테이블 스페이스의 *크기*는 테이블 스페이스가 작성되었을 때 지정되었던 최초의 *크기*를 기반으로 합니다.

규칙

- 번호 순으로 지정된 각 데이터베이스 파티션은 `db2nodes.cfg` 파일에 정의되어야 합니다(SQLSTATE 42729).
- *db-partitions-clause*절에 나열된 각 *db-partition-number*는 고유한 데이터베이스 파티션이어야 합니다(SQLSTATE 42728).
- 유효한 파티션 번호는 0 - 999 사이입니다(SQLSTATE 42729).
- 데이터베이스 파티션은 ADD 및 DROP절 모두에 표시될 수 없습니다(SQLSTATE 42728).
- 데이터베이스 파티션 그룹에 최소한 하나의 데이터베이스 파티션이 남아 있어야 합니다. 마지막 데이터베이스 파티션은 데이터베이스 파티션 그룹에서 삭제할 수 없습니다(SQLSTATE 428C0).
- 데이터베이스 파티션을 추가할 때 LIKE DBPARTITIONNUM절 또는 WITHOUT TABLESPACES절을 지정하지 않은 경우, 디폴트값은 데이터베이스 파티션 그룹에서 기존 데이터베이스 파티션의 최하위 데이터베이스 파티션 번호(예를 들어 *2*인 경우)를 사용하여 LIKE DBPARTITIONNUM *2*가 지정된 것처럼 진행하는 것입니다. 디폴트값으로 사용되는 기존 데이터베이스 파티션의 경우, 데이터베이스 파티션 그룹의 모든 테이블 스페이스에 대해 정의된 컨테이너가 있어야 합니다(SYSCAT.DBPARTITIONGROUPDEF의 IN_USE 컬럼은 'T'가 아님).
- 데이터베이스 서버 요청이 보류 또는 진행 중인 경우 ALTER DATABASE PARTITION GROUP문이 실패할 수 있습니다(SQLSTATE 55071). 새 데이터베이스 파티션 서버가 온라인으로 인스턴스에 추가되고 모든 응용프로그램이 새 데이터베이스 파티션 서버를 인식하지 않는 경우, 이 명령문도 실패할 수 있습니다(SQLSTATE 55077).

주

- 데이터베이스 파티션이 데이터베이스 파티션 그룹에 추가될 때, 데이터베이스 파티션에 대한 카탈로그 항목이 작성됩니다(SYSCAT.DBPARTITIONGROUPDEF 참조). 다음 중 어느 한 경우에 해당될 때, 분산 맵은 즉시 변경되어 데이터베이스 파티션이 분산 맵에 있음을 나타내는 표시기(IN_USE)와 함께 새 데이터베이스 파티션을 포함합니다.
  - 데이터베이스 파티션 그룹에 테이블 스페이스가 정의되지 않은 경우
  - 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 테이블이 정의되지 않고 WITHOUT TABLESPACES절이 지정되지 않은 경우

  다음 중 어느 한 경우에 해당될 때, 분산 맵은 변경되지 않고 표시기(IN_USE)는 데이터베이스 파티션이 분산 맵에 포함되지 않음을 표시하도록 설정됩니다.
  - 테이블이 데이터베이스 파티션 그룹의 테이블 스페이스에 존재하는 경우 또는
  - 테이블 스페이스가 데이터베이스 파티션 그룹에 존재하고 WITHOUT TABLESPACES절이 지정된 경우 (모든 테이블 스페이스가 자동 스토리지를 사용하도록 지정되지 않는 경우 WITHOUT TABLESPACES절은 무시됨)

  분산 맵을 변경하려면 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용해야 합니다. 이는 데이터를 다시 분산하고 분산 맵을 변경한 후 표시기를 변경합니다. WITHOUT TABLESPACES절을 지정한 경우 데이터 재분배를 시도하기 전에 테이블 스페이스 컨테이너를 추가해야 합니다.
- 데이터베이스 파티션이 데이터베이스 파티션 그룹에서 삭제될 때 데이터베이스 파티션에 대한 카탈로그 항목이 갱신됩니다(SYSCAT.DBPARTITIONGROUPDEF 참조). 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 테이블이 정의되지 않은 경우, 분산 맵이 즉시 변경되어 삭제된 데이터베이스 파티션을 제외시키고 데이터베이스 파티션 그룹의 데이터베이스 파티션에 대한 항목이 삭제됩니다. 테이블이 존재할 경우 파티션 분산 맵은 변경되지 않고 표시기(IN_USE)는 데이터베이스 파티션이 삭제되기를 기다리고 있음을 나타내도록 설정됩니다. REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 데이터를 재분배하고 데이터베이스 파티션에 대한 항목을 데이터베이스 파티션 그룹에서 삭제해야 합니다.
- *호환성*: 이전 버전의 DB2 제품과의 호환성
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - DBPARTITIONNUMS 대신 NODES를 지정할 수 있습니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.

예 :

데이터베이스에 6개의 데이터베이스 파티션 0, 1, 2, 5, 7, 8이 있다고 가정하십시오. 두 데이터베이스 파티션 3과 6이 시스템에 추가되었습니다.

- 사용자가 데이터베이스 파티션 3과 6을 MAXGROUP이라는 데이터베이스 파티션 그룹에 추가하려 하고 데이터베이스 파티션 2에 있는 것과 같은 테이블 스페이스 컨테이너가 있다고 가정하십시오. 이 경우 명령문은 다음과 같습니다.

```
ALTER DATABASE PARTITION GROUP MAXGROUP
  ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- 사용자가 데이터베이스 파티션 1을 삭제하고 데이터베이스 파티션 6을 데이터베이스 파티션 그룹 MEDGROUP에 추가하려 한다고 가정하십시오. ALTER TABLESPACE를 사용하여 데이터베이스 파티션 6에 대해 별도의 테이블 스페이스 컨테이너를 정의하고자 합니다. 명령문은 다음과 같습니다.

```
ALTER DATABASE PARTITION GROUP MEDGROUP
 ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
 DROP DBPARTITIONNUM(1)
```

ALTER DATABASE

ALTER DATABASE문은 자동 스토리지 테이블 스페이스에 사용되는 경로 콜렉션에 새 스토리지 경로를 추가하거나 기존 스토리지 경로를 제거합니다. 자동 스토리지 테이블 스페이스는 자동 스토리지를 사용하여 작성된 테이블 스페이스입니다. 즉, MANAGED BY AUTOMATIC STORAGE절이 CREATE TABLESPACE문에 지정되었거나 MANAGED BY절이 전혀 지정되지 않은 것입니다. 데이터베이스가 자동 스토리지에 사용 가능한 경우, 해당 테이블 스페이스의 컨테이너 및 스페이스 관리 등록 정보는 데이터베이스 관리 프로그램에 의해 전적으로 결정될 수 있습니다. 자동 스토리지에 대해 데이터베이스를 현재 사용할 수 없는 경우, 스토리지 경로를 추가하면 사용할 수 있습니다.

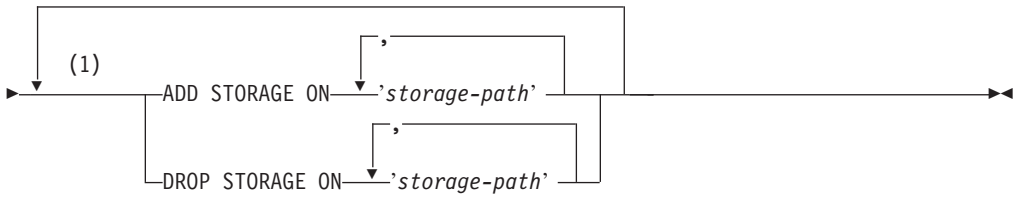
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유한 특권에는 SYSADM 또는 SYSCTRL 권한이 포함되어야 합니다.

구문



주:

1 각 절을 한 번만 지정할 수 있습니다.

설명

database-name

변경되는 데이터베이스 이름을 지정하는 선택적 값. 지정된 경우 값은 응용프로그램이 현재 연결된 데이터베이스 이름(클라이언트가 카탈로그된 별명이 아니라)과 일치해야 합니다. 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 42961).

ADD STORAGE ON

하나 이상의 새 스토리지 경로가 자동 스토리지 테이블 스페이스에 사용되는 스토리지 경로의 콜렉션에 추가되도록 지정합니다.

'storage-path'

자동 스토리지 테이블 스페이스를 작성할 컨테이너의 절대 경로 또는 드라이브 이름(Windows 운영 체제 전용)을 지정하는 문자열 상수.

DROP STORAGE ON

하나 이상의 스토리지 경로가 자동 스토리지 테이블 스페이스에 사용되는 스토리지 경로 콜렉션에서 제거되도록 지정합니다. 테이블 스페이스가 삭제될 스토리지 경로를 활발하게 사용 중인 경우, 스토리지 경로의 상태가 『사용 중』에서 『삭제 보류』로 변경되고 스토리지 경로의 추후 사용이 금지됩니다.

'storage-path'

드라이브의 절대 경로 또는 문자 이름(Windows 운영 체제만 해당함)을 지정하는 문자열 상수.

규칙

- 추가될 스토리지 경로는 경로의 이름 지정 규칙에 따라 유효해야 하며 액세스 가능해야 합니다(SQLSTATE 57019). 마찬가지로 파티션된 데이터베이스 환경에서 스토리지 경로가 존재하고 모든 데이터베이스 파티션에서 액세스 가능해야 합니다(SQLSTATE 57019).
- 삭제될 스토리지 경로가 현재 데이터베이스에 존재해야 하고(SQLSTATE 57019) 이미 『삭제 보류』 상태에 있을 수 없습니다(SQLSTATE 55073).
- 자동 스토리지에 사용 가능한 데이터베이스가 최소 하나의 스토리지 경로를 가져야 합니다. 데이터베이스에서 모든 스토리지 경로를 삭제하는 것은 허용되지 않습니다(SQLSTATE 428HH).
- 데이터베이스 파티션 서버가 추가되는 중에는 ALTER DATABASE문을 실행할 수 없습니다(SQLSTATE 55071).

주

- 새 스토리지 경로를 추가하는 경우:
  - 자동 스토리지를 사용하는 기존의 일반 크기 및 대형 테이블 스페이스는 처음에 이러한 새 경로를 사용하지 않습니다. 데이터베이스 관리 프로그램은 스페이스 부족 조건이 발생하는 경우에만 이러한 경로에서 새 테이블 스페이스 컨테이너를 작성하도록 선택할 수 있습니다.
  - 기존 임시 테이블 스페이스는 새 스토리지 경로를 자동으로 사용하지 않는 자동 스토리지에 의해 관리됩니다. 데이터베이스가 정상적으로 중지된 후 해당 테이블 스페이스에서 컨테이너를 다시 시작하여 새 스토리지 경로 또는 경로를 사용해야

합니다. 대안으로, 임시 테이블 스페이스가 삭제 및 재작성될 수 있습니다. 작성 시, 이러한 테이블 스페이스는 여유 공간이 충분한 모든 스토리지 경로를 자동으로 사용할 수 있습니다.

- 자동 스토리지를 사용할 수 있도록 데이터베이스에 스토리지 경로를 추가하면 자동 스토리지를 사용하도록 데이터베이스가 기존 비자동 스토리지 사용 가능 테이블 스페이스를 변환하지 않습니다.
- ADD STORAGE 및 DROP STORAGE가 로그되는 조작이지만, 롤 포워드 조작 중에 재실행되는지 여부는 데이터베이스가 리스토어된 방법에 따라 다릅니다. 리스토어 조작이 데이터베이스와 연관된 스토리지 경로를 재정의하지 않는 경우, 스토리지 경로 변경을 포함하는 로그 레코드가 재실행되고 로그 레코드에 설명된 스토리지 경로가 롤 포워드 조작 중에 추가 또는 삭제됩니다. 그러나 스토리지 경로가 리스토어 조작 중에 *재정의되는* 경우 이미 스토리지 경로를 설정했다고 가정하기 때문에 롤 포워드 조작이 ADD STORAGE 또는 DROP STORAGE 로그 레코드를 재실행하지 않습니다. 이 동작은 고가용성 재해 복구(HADR) 환경에도 적용됩니다. 보조 시스템이 설정되었을 때 스토리지 경로가 재정의되면 로그 레코드가 재실행되지 않습니다.
- 데이터베이스 파티션의 스토리지 경로에 대한 여유 공간을 계산할 때, 데이터베이스 관리 프로그램은 다음 디렉토리 또는 마운트 지점이 존재하는지 점검한 후 발견된 첫 번째 디렉토리 또는 마운트 지점을 사용합니다.

  ```
  <storage path>/<instance name>/NODE####/<database name>
  <storage path>/<instance name>/NODE####
  <storage path>/<instance name>
  <storage path>
  ```

  여기서:

  – <storage path>는 데이터베이스와 연관된 스토리지 경로입니다.

  – <instance name>은 데이터베이스가 있는 인스턴스입니다.

  – NODE####는 데이터베이스 파티션 번호에 해당합니다(예를 들어, NODE0000 또는 NODE0001).

  – <database name>은 데이터베이스의 이름입니다.

  파일 시스템은 스토리지 경로 아래 임의의 위치에서 마운트할 수 있으며, 데이터베이스 관리 프로그램은 테이블 스페이스 컨테이너에 사용 가능한 실제 양이 스토리지 경로 디렉토리 자체와 연관된 양과 동일하지 않을 수도 있음을 인식합니다.

  두 개의 논리적 데이터베이스 파티션이 하나의 물리적 시스템에 존재하며 단일 스토리지 경로(/db2data)가 있는 예를 고려하십시오. 각각의 데이터베이스 파티션은 해당 스토리지 경로를 사용하지만, 자체 파일 시스템 내의 각 파티션에서 데이터를 분리하고자 할 수도 있습니다. 이 경우, 각각의 파티션에 대해 별도의 파일 시스템을 작성하여 /db2data/<instance>/NODE####에 마운트할 수 있습니다. 스토리지 경

로에 컨테이너를 작성하고 여유 공간을 판별할 때, 데이터베이스 관리 프로그램은 /db2data에 대한 여유 공간 정보를 검색하지 않지만 대신 해당하는 /db2data/<instance>/NODE#### 디렉토리에 대한 여유 공간을 검색합니다.

- 일반적으로 멀티파티션 데이터베이스의 각 파티션에 대해 동일한 스토리지 경로를 사용해야 합니다. 한 가지 예외는 데이터베이스 파티션 표현식을 스토리지 경로 내에서 사용하는 경우입니다. 이를 수행하여 데이터베이스 파티션 번호를 스토리지 경로에 반영할 수 있습니다. 결과로 생성되는 경로 이름은 파티션마다 다릅니다.

  『 $N』([blank]$N) 인수를 사용하여 데이터베이스 파티션 표현식을 나타낼 수 있습니다. 데이터베이스 파티션 표현식은 스토리지 경로의 어디에서든 사용할 수 있으며 데이터베이스 파티션 표현식을 여러 개 지정할 수도 있습니다. 데이터베이스 파티션 표현식은 스페이스 문자로 끝내십시오. 스페이스 다음에 오는 문자는 데이터베이스 파티션 표현식을 평가한 후 스토리지 경로에 추가됩니다. 스토리지 경로에서 데이터베이스 파티션 표현식 뒤에 스페이스 문자가 없으면, 나머지 문자열은 표현식의 일부로 간주됩니다. 인수는 다음 형식 중 한 가지 형식으로만 사용할 수 있습니다.

표 10. 스토리지 경로 작성에 필요한 인수. 연산자는 왼쪽에서 오른쪽으로 계산됩니다. 예에서 데이터베이스 파티션 번호는 10으로 가정합니다.

구문	예 :	값
[blank]$N	″ $N″	10
[blank]$N+[number]	″ $N+100″	110
[blank]$N%[number]	″ $N%5″ [a]	0
[blank]$N+[number]%[number]	″ $N+1%5″	1
[blank]$N%[number]+[number]	″ $N%4+2″	4

[a] %는 모듈러스 연산자를 나타냅니다.

- 하나 이상의 테이블 스페이스가 사용 중인 스토리지 경로를 삭제할 때 경로의 상태가 『사용 중』에서 『삭제 보류』로 변경됩니다. 경로의 추후 증가는 발생하지 않습니다. 경로가 데이터베이스에서 완전히 제거될 수 있기 전에 컨테이너 데이터가 스토리지 경로로 이동되도록 영향을 받는 각 테이블 스페이스가(ALTER TABLESPACE 문의 REBALANCE절을 사용하여) 재조정을 이뤄야 합니다. 재조정은 일반 및 대형 테이블 스페이스의 경우에만 지원됩니다. 임시 테이블 스페이스는 컨테이너가 삭제된 경로에서 제거되도록 삭제 및 재작성되어야 합니다. 경로는 테이블 스페이스에 의해 더 이상 사용 중이 아닐 때 데이터베이스에서 물리적으로 제거됩니다.

  파티션된 데이터베이스의 경우 경로는 각 파티션에 독립적으로 유지됩니다. 경로는 주어진 데이터베이스 파티션에서 더 이상 사용 중이 아닐 때 해당 파티션에서 물리적으로 제거됩니다. 기타 파티션은 여전히 경로를 『삭제 보류』 상태에 있는 것으로 표시할 수 있습니다.

  삭제 보류 스토리지 경로를 사용하는 자동 스토리지 테이블 스페이스의 목록은 다음 SQL문을 발행하여 판별할 수 있습니다.

```
SELECT DISTINCT A.TBSP_NAME, A.TBSP_ID, A.TBSP_CONTENT_TYPE
  FROM SYSIBMADM.SNAPTBSP A, SYSIBMADM.SNAPTBSP_PART B
  WHERE A.TBSP_ID = B.TBSP_ID AND B.TBSP_PATHS_DROPPED = 1
```

- 원래 데이터베이스 파티션 표현식을 사용하여 지정된 스토리지 경로를 삭제할 때, 데이터베이스 파티션 표현식을 포함한 동일한 스토리지 경로 문자열이 삭제(drop)에서 사용되어야 합니다. 데이터베이스 파티션 표현식이 지정된 경우 이 경로 문자열은 데이터베이스 스냅샷의 『db 파티션 표현식을 갖는 경로』 요소 (db_storage_path_with_dpe)에서 찾을 수 있습니다. 데이터베이스 파티션 표현식이 지정된 원래 경로에 포함되지 않은 경우 이 요소는 표시되지 않습니다.
- 주어진 스토리지 경로가 여러 번 데이터베이스에 추가될 수 있습니다. DROP STORAGE ON절을 사용할 때 해당 특정 경로를 한 번 지정하면 데이터베이스에서 경로의 모든 인스턴스가 삭제됩니다.

예:

예 1: /db2 디렉토리 아래 두 개의 경로(/db2/filesystem1 및 /db2/filesystem2)와 세 번째 경로 /filesystem3를 현재 연결된 데이터베이스와 연관된 자동 스토리지 테이블 스페이스의 스페이스에 추가합니다.

```
ALTER DATABASE ADD STORAGE ON '/db2/filesystem1', '/db2/filesystem2',
       '/filesystem3'
```

예 2: SAMPLE 데이터베이스와 연관된 자동 스토리지 테이블 스페이스의 스페이스에 드라이브 D 및 E를 추가합니다.

```
ALTER DATABASE SAMPLE ADD STORAGE ON 'D:', 'E:₩'
```

예 3: F:₩DB2DATA 디렉토리 및 G 드라이브를 현재 연결된 데이터베이스와 연관된 자동 스토리지 테이블 스페이스의 스페이스에 추가합니다.

```
ALTER DATABASE ADD STORAGE ON 'F:₩DB2DATA', 'G:'
```

예 4: 데이터베이스 파티션 표현식을 사용하여 각각의 데이터베이스 파티션의 스토리지 경로를 구분하는 스토리지 경로를 추가합니다.

```
ALTER DATABASE ADD STORAGE ON '/dataForPartition $N'
```

데이터베이스 파티션 0에서 사용되는 스토리지 경로는 /dataForPartition0이며, 데이터베이스 파티션 1에서는 /dataForPartition1입니다.

예 5: 데이터베이스에 대한 자동 스토리지를 사용 가능하도록 하기 위해 자동 스토리지가 가능하지 않은 데이터베이스에 스토리지 경로를 추가하십시오.

```
CREATE DATABASE MYDB AUTOMATIC STORAGE NO
CONNECT TO MYDB
ALTER DATABASE ADD STORAGE ON '/db2/filesystem1', '/db2/filesystem2'
```

MYDB 데이터베이스가 이제 자동 스토리지에 사용 가능합니다.

예 6: 현재 연결된 데이터베이스에서 /db2/filesystem1 및 /db2/filesystem2 경로를 제거하십시오.

```
ALTER DATABASE DROP STORAGE ON '/db2/filesystem1', '/db2/filesystem2'
```

스토리지가 성공적으로 삭제되면, 이러한 스토리지 경로를 사용한 각 테이블 스페이스에 대하여 REBALANCE 절과 함께 ALTER TABLESPACE 명령문을 사용하여 테이블 스페이스를 재조정하십시오.

예 7: 데이터베이스 파티션 표현식(/dataForPartition $N)을 갖는 스토리지 경로가 이전에 데이터베이스에 추가되었으며 이제 제거해야 합니다.

```
ALTER DATABASE DROP STORAGE ON '/dataForPartition $N'
```

스토리지가 성공적으로 삭제되면, 이러한 스토리지 경로를 사용한 각 테이블 스페이스에 대하여 REBALANCE 절과 함께 ALTER TABLESPACE 명령문을 사용하여 테이블 스페이스를 재조정하십시오.

ALTER FUNCTION

ALTER FUNCTION문은 기존 함수의 등록 정보를 수정합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 함수 스키마에 대한 ALTERIN 특권
- SYSCAT.ROUTINES 카탈로그 뷰의 OWNER 컬럼에 기록된 함수의 소유자
- DBADM 권한

함수의 EXTERNAL NAME을 변경하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

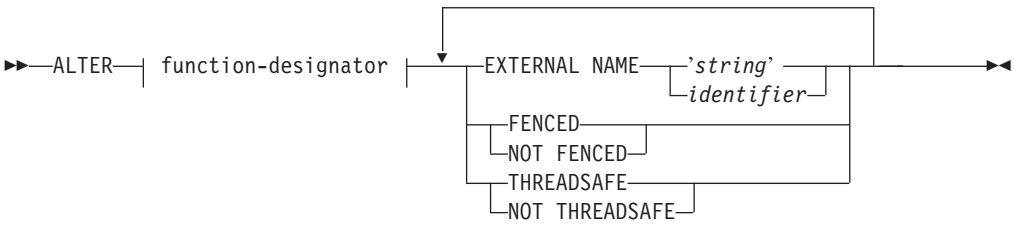
- 데이터베이스에 있는 CREATE_EXTERNAL_ROUTINE 권한
- DBADM 권한

함수를 분리(fenced)되지 않게 변경하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스에서의 CREATE_NOT_FENCED_ROUTINE 권한
- DBADM 권한

함수가 분리되도록 변경하려면 추가 권한이나 특권이 필요하지 않습니다.

구문



설명

function-designator

변경할 함수를 고유하게 식별하십시오. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

EXTERNAL NAME *'string'* **또는** *identifier*

함수를 구현하는 사용자 작성 코드의 이름을 식별하십시오. 이 옵션은 외부 함수를 변경할 때만 지정할 수 있습니다(SQLSTATE 42849).

FENCED 또는 NOT FENCED

함수가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 어드레스 스페이스에서 실행되어도 안전한지(NOT FENCED) 아니면 안전하지 않은지(FENCED) 여부를 지정하십시오. 대부분의 함수는 FENCED나 NOT FENCED로 실행 중인 옵션을 갖고 있습니다.

함수가 FENCED로 변경된 경우, 데이터베이스 관리 프로그램은 내부 자원(예: 데이터 버퍼)을 메소드에 의한 액세스로부터 단절시킵니다. 일반적으로 FENCED로 수행되는 함수는 NOT FENCED로 실행되는 함수와는 다르게 실행됩니다.

주의:

제대로 코딩, 검토 및 테스트되지 않은 함수에 NOT FENCED를 사용하면 DB2 데이터베이스의 무결성이 손상될 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 사용자 정의 함수가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

NOT THREADSAFE로 선언된 함수는 NOT FENCED로 변경할 수 없습니다(SQLSTATE 42613).

함수에 AS LOCATOR로 정의된 매개변수가 있고 함수가 NO SQL 옵션을 사용하여 정의된 경우에는 함수를 FENCED로 변경할 수 없습니다(SQLSTATE 42613).

LANGUAGE OLE, OLEDB 또는 CLR 함수의 경우 이 옵션을 변경할 수 없습니다(SQLSTATE 42849).

THREADSAFE 또는 NOT THREADSAFE

함수가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 아니면 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

함수가 OLE 및 OLEDB가 아닌 다른 LANGUAGE로 정의된 경우에는 다음과 같습니다.

- 함수가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 함수를 호출할 수 있습니다. 일반적으로 스레드가 안전하려면 함수가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED와 NOT FENCED 함수는 둘다 THREADSAFE될 수 있습니다.
- 함수가 NOT THREADSAFE로 정의되어 있으면, 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 함수를 동시에 호출할 수 없습니다. 분리(fenced) 함수만이 NOT THREADSAFE될 수 있습니다(SQLSTATE 42613).

LANGUAGE OLE 또는 OLEDB 함수의 경우 이 옵션을 변경할 수 없습니다 (SQLSTATE 42849).

주

- SYSIBM, SYSFUN 또는 SYSPROC 스키마의 함수를 변경할 수 없습니다 (SQLSTATE 42832).
- LANGUAGE SQL, 전래 함수 또는 템플리트 함수로 선언된 함수는 변경될 수 없습니다(SQLSTATE 42917).

예 :

함수 MAIL()가 완전하게 테스트되었습니다. 이 기능을 향상시키려면 함수를 분리(fenced)가 아닌 함수로 변경하십시오.

```
ALTER FUNCTION MAIL() NOT FENCED
```

ALTER HISTOGRAM TEMPLATE

ALTER HISTOGRAM TEMPLATE문은 서비스 클래스나 작업 클래스의 디폴트 막대 그래프 중 하나 이상의 대체하기 위해 사용할 수 있는 막대 그래프의 유형을 설명하는 템플리트를 수정하기 위해 사용됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

구문

설명

template-name

막대 그래프 템플리트의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. 이름은 현재 서버에서 기존 막대 그래프 템플리트를 식별해야 합니다(SQLSTATE 42704). 템플리트 이름은 디폴트 시스템 막대 그래프 템플리트 SYSDEFAULTHISTOGRAM이 될 수 있습니다.

HIGH BIN VALUE *bigint-constant*

마지막 바이너리에 대한 최상위 초 값을 지정합니다(마지막 바이너리는 바운드되지 않은 최상위 값을 가짐). 단위는 막대 그래프 사용 방법에 따라 결정됩니다. 최대 값은 268 435 456입니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)

  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 모든 파티션 사이에서 한 번에 하나의 언커미트된 WLM 독점 SQL문만 허용됩니다. 언커미트된 WLM 독점 SQL문이 실행 중인 경우, 연속 WLM 독점 SQL문은 현재 WLM 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 발행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예 :

막대 그래프 템플리트 LIFETIMETEMP의 상위 바이너리 값을 변경하십시오.

```
ALTER HISTOGRAM TEMPLATE LIFETIMETEMP
  HIGH BIN VALUE 90000
```

ALTER INDEX

ALTER INDEX문은 인덱스의 정의를 변경합니다.

호출

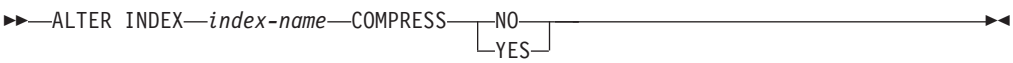
이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 인덱스의 스키마에 대한 ALTERIN 특권
- 인덱스가 정의되는 테이블에 대한 ALTER 특권
- 인덱스에 대한 CONTROL 특권
- DBADM 권한

구문



설명

INDEX *index-name*

변경될 인덱스를 식별합니다. 이름은 현재 서버에 존재하는 인덱스를 식별해야 합니다(SQLSTATE 42704).

COMPRESS

인덱스 압축의 사용 가능 여부를 지정합니다. 인덱스는 MDC 블록 인덱스, 카탈로그 인덱스, XML 경로 인덱스, 인덱스 스펙 또는 작성된 임시 테이블이나 선언된 임시 테이블의 인덱스여서는 안됩니다(SQLSTATE 42995).

NO

인덱스 압축이 사용 불가능함을 지정합니다. 압축된 인덱스는 인덱스 재구성 또는 재작성으로 인덱스가 재빌드될 때까지 압축된 상태로 남아 있습니다.

YES

인덱스 압축이 사용 가능함을 지정합니다. 압축 해제된 인덱스는 인덱스 재구성 또는 재작성으로 인덱스가 재빌드될 때까지 압축 해제된 상태로 남아 있습니다.

예

예 1: JOB_BY_DPT 인덱스를 압축된 인덱스로 변경.

```
ALTER INDEX JOB_BY_DPT
COMPRESS YES
```

ALTER METHOD

ALTER METHOD문은 메소드와 연관된 메소드 본문을 변경하여 기존 메소드를 수정합니다.

호출

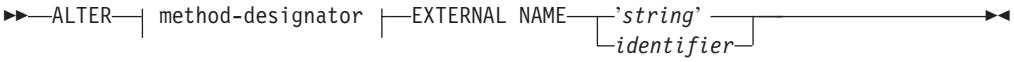
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스의 CREATE_EXTERNAL_ROUTINE 권한은 다음 중 최소한 하나를 포함해야 합니다.
  - 유형 스키마에 대한 ALTERIN 특권
  - SYSCAT.DATATYPES 카탈로그 뷰의 OWNER 컬럼에 기록된 유형의 소유자
- DBADM 권한

구문



설명

method-designator

변경할 메소드를 고유하게 식별하십시오. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

EXTERNAL NAME *'string'* **또는** *identifier*

메소드를 구현하는 사용자 작성 코드의 이름을 식별하십시오. 이 옵션은 외부 메소드를 변경할 때만 지정할 수 있습니다(SQLSTATE 42849).

주

- SYSIBM, SYSFUN 또는 SYSPROC 스키마의 메소드를 변경할 수 없습니다(SQLSTATE 42832).
- LANGUAGE SQL로 선언된 메소드는 변경할 수 없습니다(SQLSTATE 42917).
- LANGUAGE CLR로 선언된 메소드는 변경할 수 없습니다(SQLSTATE 42849).
- 지정한 메소드에 본문이 있어야 메소드를 변경할 수 있습니다(SQLSTATE 42704).

예 :

newaddresslib 라이브러리를 사용하도록 구조화된 유형 ADDRESS_T의 DISTANCE() 메소드를 변경하십시오.

```
ALTER METHOD DISTANCE()
  FOR TYPE ADDRESS_T
  EXTERNAL NAME 'newaddresslib!distance2'
```

ALTER MODULE

ALTER MODULE 명령문은 모듈 정의를 변경합니다.

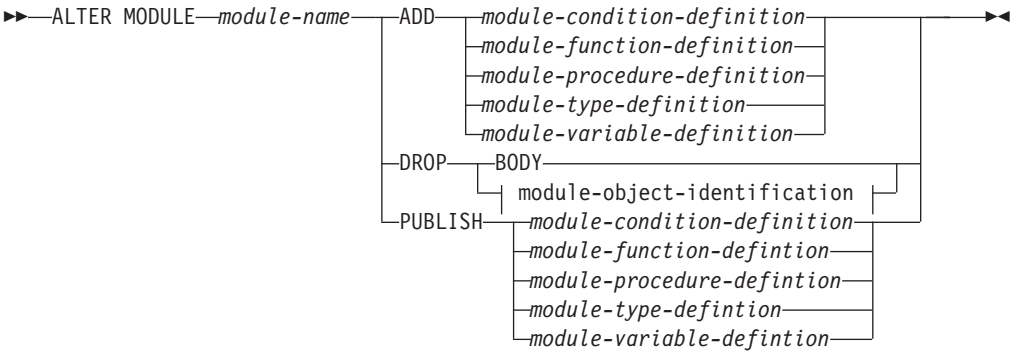
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

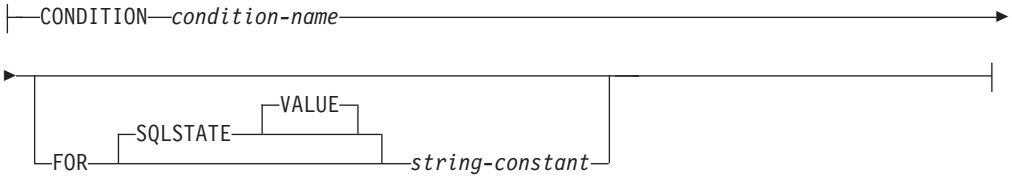
권한 부여

명령문의 권한 부여 ID에서 보유한 특권은 모듈 소유권을 포함해야 하며 ALTER MODULE 명령문 내에 지정된 SQL문을 호출하기 위해 필요한 모든 특권도 포함해야 합니다.

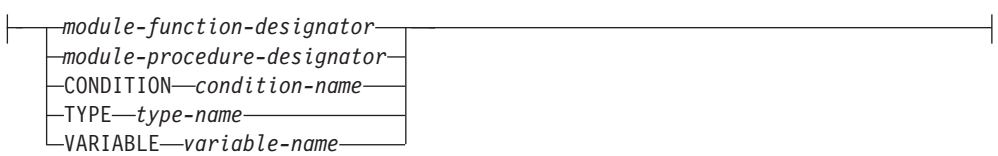
구문



module-condition-definition:



module-object-identification:



설명

module-name

변경할 모듈을 식별합니다. module-name은 현재 서버에 있는 모듈을 식별해야 합니다(SQLSTATE 42704).

ADD

오브젝트를 모듈에 추가하거나 본문이 없는 모듈에 이미 존재하는 루틴 정의에 본문을 추가합니다. 사용자 정의 유형이나 전역 변수를 추가하면 오브젝트는 모듈에 이미 존재하는 전역 변수나 사용자 정의 유형을 식별해서는 안됩니다. 사용자 정의 유형이나 전역 변수가 존재하지 않았다면 모듈 내에서만 하도록 모듈에 추가됩니다.

추가 루틴과 지정된 루틴이 없으면 루틴이 추가됩니다. 추가 루틴과 지정된 루틴이 다음 중 하나와 같이 존재하면,

- 동일한 특정 이름과 동일한 루틴 이름.
- 특정 이름과 상관 없이 동일한 서명(module-identification-rules 사용).

기존 루틴 정의는 루틴 본문(SQLSTATE 42723)을 포함해서는 안됩니다. 루틴 프로토타입은 루틴 속성과 루틴 분문을 포함하며 보유한 발행된 속성을 제외하고 새 루틴 정의로 완전히 대체됩니다.

module-condition-definition

모듈 조건을 추가합니다.

condition-name

조건 이름. 이름은 모듈의 기존 조건을 식별해서는 안됩니다. condition-name은 자격 없이 지정되어야 합니다(SQLSTATE 42601). 조건 이름은 모듈 내에서 고유해야 합니다.

FOR SQLSTATE *string-constant*

조건과 연관된 SQLSTATE를 지정합니다. string-constant는 작은따옴표로 묶어 5자로 지정해야 하며, SQLSTATE 클래스(처음 2자)는 '00'이 아니어야 합니다. 이것은 선택적 절입니다.

module-function-definition

함수를 추가할 구문은 CREATE 키워드가 없는 CREATE FUNCTION문과 동일하며 function-name과 specific-name 모두는 자격 없이 지정되어야 합니다(SQLSTATE 42601). 함수가 모듈 내에서 고유하면 새 함수가 추가됩니다. 함수가 분문(SQL-routine-body 또는 EXTERNAL NAME 절)을 포함하지 않은 기존 함수와 일치하면, 이 함수 프로토타입은 보유된 발행된 속성을 제외하고 새 정의로 대체됩니다. 복합 SQL(컴파일된)문이 사용된 것과 같이 모든 SQL 함수가 모듈에 추가됩니다.

모듈 함수 정의는 SOURCE절, TEMPLATE절 또는 LANGUAGE OLEDEB 옵션을 지정해서는 안 됩니다(SQLSTATE 42613).

module-procedure-definition

프로시저를 정의할 구문은 CREATE 키워드를 제외한 CREATE PROCEDURE 문과 동일하며 procedure-name과 specific-name 모두가 자격 없이 지정되어야 합니다(SQLSTATE 42601). 프로시저 시그니처가 모듈 내에서 고유하면 새 프로시저가 추가됩니다. 프로시저가 본문(SQL-routine-body 또는 EXTERNAL NAME 절)을 포함하지 않는 기존 프로시저와 일치하면 이 프로시저 프로토타입은 발행된 속성을 보유한다는 점을 제외하고는 새 정의로 대체됩니다. 프로시저 이름은 SYS_INIT이라는 모듈 초기화 프로시저를 추가하기 위해서 『SYS_』만으로 시작할 수 있습니다. 세부사항은 주를 참조하십시오.

module-type-definition

사용자 정의 유형을 정의할 구문은 CREATE 키워드를 제외하고는 CREATE TYPE문과 동일하며 type-name은 자격 없이 지정되어야 합니다(SQLSTATE 42601). 사용자 정의 유형 이름은 모듈 내에서 고유해야 합니다.구조화된 유형을 모듈에서 정의할 수 없습니다. 유형 정의를 지원하기 위해 생성된 함수도 모듈에서 정의됩니다. 모듈 사용자 정의 유형이 발행되면 생성된 함수도 발행됩니다.

module-variable-definition

변수를 정의할 구문은 CREATE 키워드를 제외하고는 CREATE VARIABLE 문과 동일하며 변수 이름은 자격 없이 지정됩니다(SQLSTATE 42601). 변수 이름은 모듈 내에서 고유해야 합니다.

DROP

모듈의 지정된 파트를 삭제하십시오. 모듈의 본문이 삭제되지 않는 한 module-object-identification 구문은 삭제될 오브젝트를 식별하기 위해 사용됩니다.

BODY

다음을 포함한 모듈 본문을 삭제하십시오.

- 발행되지 않은 모든 오브젝트.
- 발행된 SQL 루틴의 루틴 본문
- 발행된 외부 루틴을 위한 EXTERNAL 참조.

PUBLISH

새 오브젝트를 모듈에 추가하고 모듈 밖에서 사용하도록 유효하게 하십시오. 루틴의 경우, 루틴의 실행 가능한 본문을 포함하지 않도록 루틴 프로토타입을 지정할 수 있습니다.

module-condition-definition

모듈 밖에서 사용하도록 유효한 모듈 조건을 추가하십시오.

condition-name

조건 이름. 이름은 모듈의 기존 조건을 식별해서는 안됩니다. condition-name은 자격 없이 지정되어야 합니다(SQLSTATE 42601). 조건 이름은 모듈 내에서 고유해야 합니다.

FOR SQLSTATE *string-constant*

조건과 연관된 SQLSTATE를 지정합니다. string-constant는 작은따옴표로 묶어 5자로 지정해야 하며, SQLSTATE 클래스(처음 2자)는 '00'이 아니어야 합니다. 이것은 선택적 절입니다.

module-function-definition

함수를 정의할 구문은 CREATE 키워드가 없는 CREATE FUNCTION문과 동일하며 function-name과 specific-name 모두는 자격 없이 지정되어야 합니다(SQLSTATE 42601). 함수 정의에는 함수 이름, 매개변수의 전체 스펙 및 리턴 절을 포함됩니다. 발행되지 않은 모듈 사용자 정의 데이터 유형은 매개변수 데이터 유형이나 RETURNS절 데이터 유형에 대한 후보가 아닙니다. 발행되지 않은 모듈 변수는 리턴 데이터 유형이나 매개변수 데이터 유형의 ANCHOR절에서 앵커 오브젝트에 대한 후보가 아닙니다. LANGUAGE절(또는 LANGUAGE SQL 지정) 및 SQL-routine-body를 생략하여 함수 프로토타입을 지정할 수 있습니다. 함수 시그니처는 모듈 내에서 고유해야 합니다. 함수 이름은 "SYS_"로 시작하면 안됩니다(SQLSTATE 42939). 복합 SQL(컴파일된)문이 사용된 것과 같이 모든 SQL 함수가 모듈에 추가됩니다.

모듈 함수 정의는 SOURCE절, TEMPLATE절 또는 LANGUAGE OLEDEB 옵션을 지정해서는 안 됩니다(SQLSTATE 42613).

module-procedure-definition

프로시저를 정의할 구문은 CREATE 키워드를 제외한 CREATE PROCEDURE 문과 동일하며 procedure-name과 specific-name 모두가 자격 없이 지정되어야 합니다(SQLSTATE 42601). 프로시저 정의는 프로시저 이름과 매개변수의 전체 스펙을 포함해야 합니다. 발행되지 않은 모듈 사용자 정의 데이터 유형은 매개변수 데이터 유형에 대한 후보가 아닙니다. 발행되지 않은 모듈 변수는 매개변수 정의의 ANCHOR 절에서 앵커 오브젝트에 대한 후보가 아닙니다. LANGUAGE절(또는 LANGUAGE SQL 지정) 및 SQL-routine-body를 생략하여 함수 프로토타입을 지정할 수 있습니다. 프로시저 시그니처는 모듈 내에서 고유해야 합니다. 프로시저 이름은 "SYS_"로 시작하면 안됩니다(SQLSTATE 42939).

module-type-definition

사용자 정의 유형을 정의할 구문은 CREATE 키워드를 제외하고는 CREATE TYPE문과 동일하며 type-name은 자격 없이 지정되어야 합니다(SQLSTATE 42601). 발행되지 않은 모듈 사용자 정의 데이터 유형은 모듈 사용자 정의 데이터 유형 정의에서 참조된 데이터 유형에 대한 후보가 아닙니다. 발행되지 않

은 모듈 변수는 ANCHOR절의 앵커 오브젝트에 대한 후보가 아닙니다. 사용자 정의 유형의 이름은 “SYS_”(SQLSTATE 42939)로 시작할 수 없으며 모듈 내에서 고유해야 합니다. 구조화된 유형을 모듈에서 정의할 수 없습니다. 또한 유형 정의를 지원하기 위해 생성된 함수는 발행된 함수로서 모듈에서 정의됩니다.

module-variable-definition

변수를 정의할 구문은 CREATE 키워드를 제외하고는 CREATE VARIABLE 문과 동일하며 변수 이름은 자격 없이 지정됩니다(SQLSTATE 42601). 발행되지 않은 모듈 사용자 정의 데이터 유형은 변수 정의에서 참조된 데이터 유형에 대해 후보가 아닙니다. 발행되지 않은 모듈 변수는 ANCHOR절의 앵커 오브젝트에 대한 후보가 아닙니다. 변수 이름은 “SYS_”(SQLSTATE 42939)로 시작할 수 없으며 모듈 내에서 고유해야 합니다.

module-object-identification

고유한 모듈 오브젝트를 식별합니다.

module-function-designator

단순 모듈 함수를 고유하게 식별합니다.

FUNCTION *unqualified-function-name*

특정 함수를 식별하여 모듈에 unqualified-function-name 이름으로 정확히 하나의 함수 인스턴스가 있는 경우에만 유효합니다. 식별된 함수에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 이 이름의 함수가 모듈에 존재하지 않으면 오류(SQLSTATE 42704)가 발생합니다. 모듈에 함수의 인스턴스가 둘 이상 있는 경우 오류(SQLSTATE 42725)가 발생합니다.

FUNCTION *unqualified-function-name (data type,...)*

함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 결정 알고리즘은 사용되지 않습니다.

unqualified-function-name

함수 이름을 지정합니다.

(data-type,...)

값은 함수가 원래 정의되었을 때 지정된(해당 위치의) 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합이 특정 함수 인스턴스를 식별하는 데 사용됩니다.

데이터 유형이 규정되어 있지 않은 경우, SQL 경로에서 스키마를 검색하여 유형 이름이 해석됩니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다. 매개변수 값이 서로 다

른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601). 길이, 정밀도 또는 스케일을 코딩하는 경우 값은 함수가 원래 정의될 때 지정된 값과 정확히 일치해야 합니다.

0 < n < 25는 REAL을 의미하고 24 < n < 54는 DOUBLE을 의미하기 때문에 FLOAT(n) 유형은 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 DOUBLE인지에 따라 발생합니다. 지정된 시그니처의 함수가 모듈에 존재하지 않은 경우 오류(SQLSTATE 42883)가 발행합니다.

SPECIFIC FUNCTION *unqualified-specific-name*

함수 정의 시 지정되거나 디폴트로 설정된 이름을 사용하여 특정 사용자 정의 함수를 식별합니다. unqualified-specific-name은 모듈의 특정 함수 인스턴스를 식별해야 합니다. 식별하지 않으면 오류가 리턴됩니다(SQLSTATE 42704).

module-procedure-designator

단순 모듈 프로시저를 고유하게 식별합니다.

PROCEDURE *unqualified-procedure-name*

특정 프로시저를 식별하여 모듈에서 unqualified-procedure-name 이름으로 정확히 하나의 프로시저 인스턴스가 있는 경우에만 유효합니다. 식별된 프로시저에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 모듈에 이 이름의 프로시저가 없는 경우에는 오류가 리턴됩니다(SQLSTATE 42704). 모듈에 프로시저의 인스턴스가 둘 이상 있는 경우 오류가 리턴됩니다(SQLSTATE 42725).

PROCEDURE *unqualified-procedure-name (data-type,...)*

프로시저를 고유하게 식별하는 프로시저 시그니처를 제공합니다. 프로시저 분석 알고리즘은 사용하지 않습니다.

unqualified-procedure-name

프로시저 이름을 지정합니다.

(data-type,...)

값은 프로시저가 원래 정의되었을 때 지정된(해당 위치의) 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합이 특정 프로시저 인스턴스를 식별하는 데 사용됩니다.

데이터 유형이 규정되어 있지 않은 경우, SQL 경로에서 스키마를 검색하여 유형 이름이 해석됩니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601). 길이, 정밀도 또는 스케일을 코딩하는 경우 값은 프로시저가 정의될 때 지정된 값과 정확히 일치해야 합니다.

0 < n < 25는 REAL을 의미하고 24 < n < 54는 DOUBLE을 의미하기 때문에 FLOAT(n) 유형은 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 DOUBLE인지에 따라 발생합니다.

모듈에 지정한 서명의 프로시저가 없는 경우 오류가 리턴됩니다(SQLSTATE 42883).

SPECIFIC PROCEDURE *unqualified-specific-name*
프로시저 정의 시 지정되거나 디폴트로 설정된 이름을 사용하여 특정 프로시저를 식별합니다. unqualified-specific-name은 모듈의 특정 프로시저 인스턴스를 식별해야 합니다. 식별하지 않으면 오류가 리턴됩니다(SQLSTATE 42704).

TYPE *type-name*
모듈에서 사용자 정의 유형을 식별합니다. type-name은 자격(SQLSTATE 42601) 없이 지정되어야 하며 모듈에 존재하는 사용자 정의 유형을 식별해야 합니다(SQLSTATE 42704).

VARIABLE *variable-name*
모듈에서 전역 변수를 식별합니다. 변수 이름은 자격(SQLSTATE 42601) 없이 지정되어야 하며 모듈에 존재하는 전역 변수를 식별해야 합니다(SQLSTATE 42704).

CONDITION *condition-name*
모듈에서 조건을 식별합니다. condition-name은 자격 없이 지정되어야 하며 모듈에 존재하는 조건을 식별해야 합니다(SQLSTATE 42737).

규칙

- 모듈의 오브젝트 이름은 특별히 지정된 SYS_INIT 프로시저 이름을 제외하고 “SYS_”로 시작할 수 없습니다(SQLSTATE 42939).

주

- 모듈 초기화: 첫 번째 참조가 모듈 루틴이나 모듈 전역 변수로 이루어진 경우에 내재적으로 실행된 SYS_INIT이라는 특수 초기화 프로시저가 모듈에 있을 수 있습니다. SYS_INIT 프로시저가 매개변수 없이 구현되어야 하며, 결과 세트를 리턴할 수

없으며 발행될 수 없는 SQL 또는 외부 프로시저일 수 있습니다(SQLSTATE 428HP). SYS_INIT 프로시저가 실패하면 모듈 초기화를 유발하는 명령문에 대해 오류가 리턴됩니다(SQLSTATE 56098).

- 모듈 조건 사용: 모듈 조건은 SIGNAL문, RESIGNAL문 또는 복합 SQL(컴파일된) 문에 있는 핸들러 선언과 반드시 함께 사용해야 합니다.
- 무효화: 루틴 프로토타입이 ADD 조치를 사용하여 대체되는 경우 발행된 모듈 루틴에 종속된 모든 오브젝트는 무효화됩니다. DROP BODY가 발행되는 경우 발행된 모듈 루틴에 따라 모든 오브젝트는 무효화됩니다.

예 :

다음 명령문은 연관된 배열 유형을 포함한 INVENTORY라는 모듈, 해당 데이터 유형의 변수, 배열에서 요소를 추출한 함수와 요소를 배열에 추가한 프로시저를 생성합니다. 함수와 프로시저만이 해당 ALTER MODULE문의 PUBLISH 키워드를 기반으로 하는 모듈 밖에서 참조될 수 있습니다. 데이터 유형과 변수만이 모듈의 다른 오브젝트에서 참조될 수 있습니다.

```
CREATE MODULE INVENTORY

ALTER MODULE INVENTORY ADD
TYPE ITEMLIST AS INTEGER ARRAY[VARCHAR(100)]

ALTER MODULE INVENTORY ADD
VARIABLE ITEMS ITEMLIST

ALTER MODULE INVENTORY PUBLISH
PROCEDURE UPDATE_ITEM(NAME VARCHAR(100), QUANTITY INTEGER)
BEGIN
SET ITEMS[NAME] = QUANTITY;
END

ALTER MODULE INVENTORY PUBLISH
FUNCTION CHECK_ITEM(NAME VARCHAR(100) RETURNS INTEGER
RETURN ITEMS[NAME]
```

ALTER NICKNAME

ALTER NICKNAME문은 데이터 소스 오브젝트(예:테이블, 뷰 또는 파일)와 연관된 별칭 정보를 수정합니다. 이 명령문은 다음을 수행하여 페더레이티드 데이터베이스에 저장된 정보를 수정합니다.

- 데이터 소스 오브젝트의 컬럼에 대한 로컬 컬럼 이름 변경
- 데이터 소스 오브젝트의 컬럼에 대한 로컬 데이터 유형 변경
- 별칭 및 컬럼 옵션 추가, 설정 또는 삭제
- 기본 키 추가 또는 삭제
- 하나 이상의 고유, 참조 또는 점검 제한조건 추가 또는 삭제
- 하나 이상의 참조 또는 점검 제한조건 속성 변경
- 페더레이티드 서버의 데이터 캐싱 변경

호출

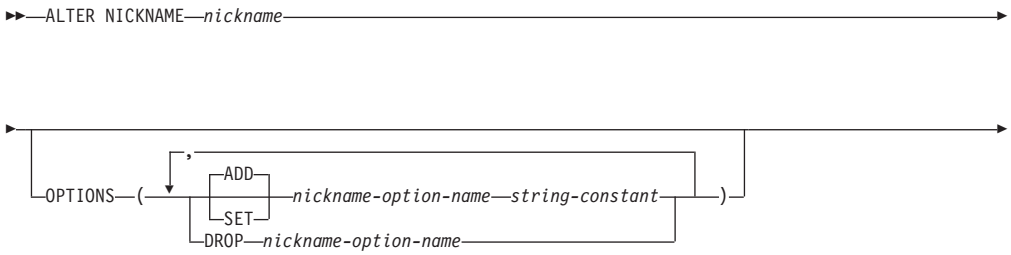
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

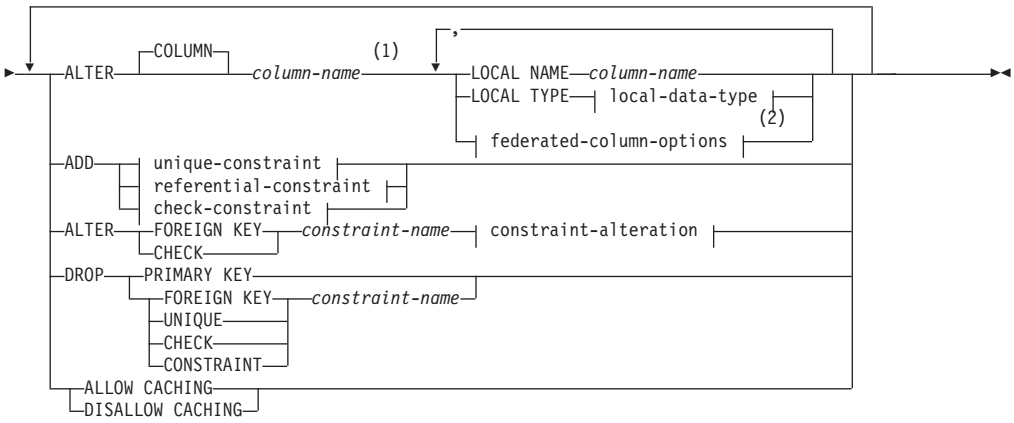
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 명령문에 지정된 별칭에 대한 ALTER 특권
- 명령문에 지정된 별칭에 대한 CONTROL 특권
- 스키마에 대한 ALTERIN 특권(별칭의 스키마 이름이 존재하는 경우)
- SYSCAT.TABLES 카탈로그 뷰의 OWNER 컬럼에 기록된 별칭의 소유자
- DBADM 권한

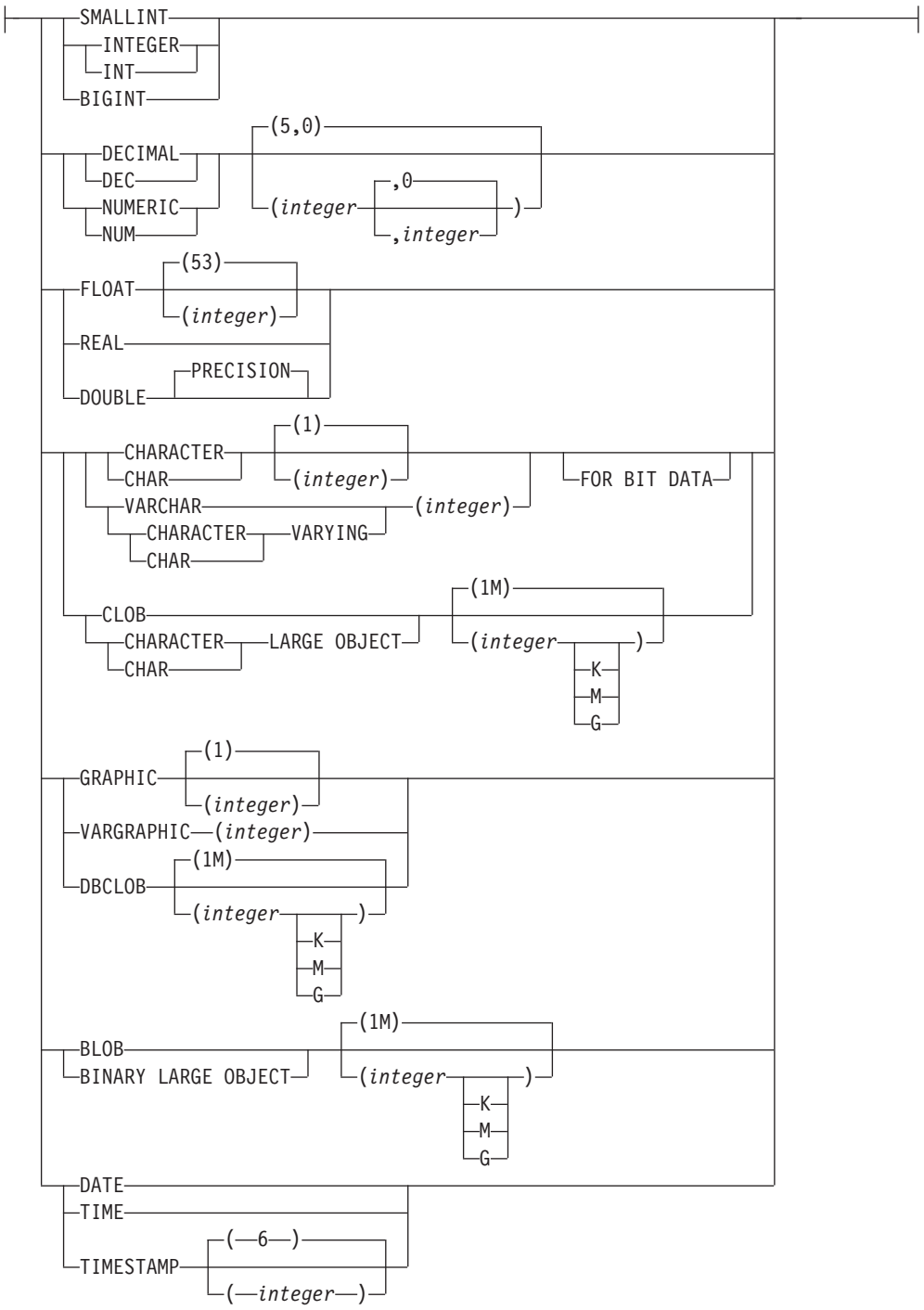
구문



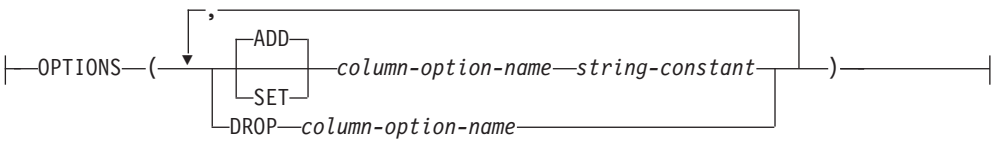


local-data-type:

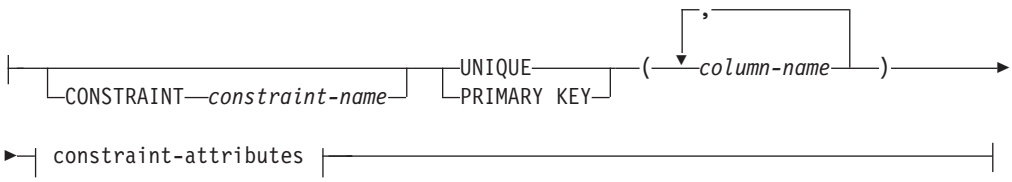
내장 유형:



federated-column-options:

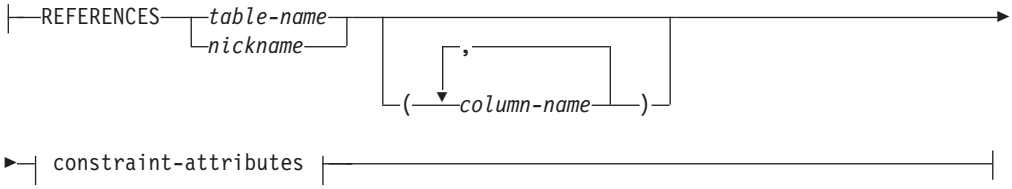


unique-constraint:

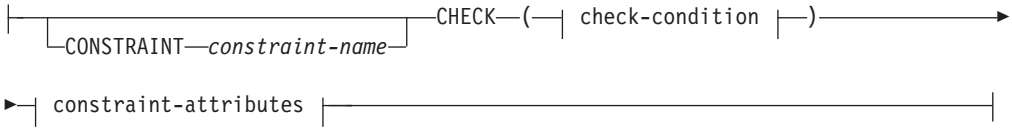


referential-constraint:

references-clause:

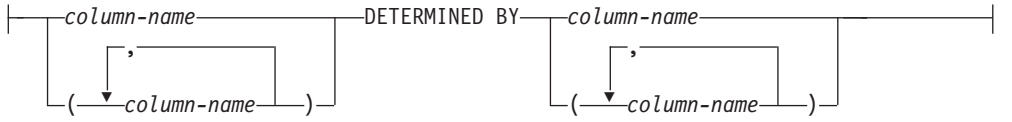


check-constraint:

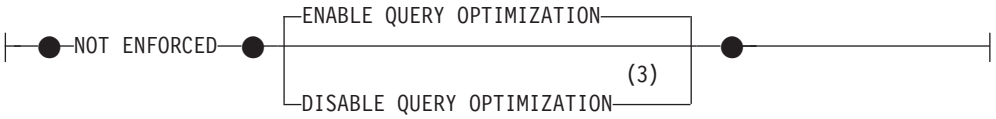


check-condition:

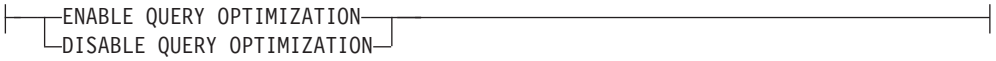
functional-dependency:



constraint-attributes:



constraint-alteration:



주:

1 ALTER COLUMN절 및 ADD, ALTER 또는 DROP 정보용 제한조건 절 모두를 동일한 ALTER NICKNAME문에 지정할 수 없습니다.

2 사용자가 LOCAL NAME 매개변수나 LOCAL TYPE 매개변수 또는 이 두 매개변수를 모두 지정하는 것 외에, federated-column-options절을 지정해야 하는 경우에는 federated-column-options절을 마지막에 지정해야 합니다.

3 DISABLE QUERY OPTIMIZATION은 고유 또는 기본 키 제한조건에 대해 지원되지 않습니다.

설명

별칭

변경될 컬럼이 포함된 데이터 소스 오브젝트(예: 테이블, 뷰 또는 파일)의 별칭을 식별합니다. 카탈로그에 기술된 별칭이어야 합니다.

OPTIONS

별칭이 변경될 때, 추가, 설정 또는 삭제되는 별칭 옵션을 표시합니다.

ADD

별칭 옵션을 추가합니다.

SET

별칭 옵션의 설정을 변경합니다.

nickname-option-name

추가 또는 설정될 별칭 옵션의 이름을 지정합니다.

string-constant

*nickname-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

DROP *nickname-option-name*

별칭 옵션을 삭제합니다.

ALTER COLUMN *column-name*

변경될 컬럼의 이름을 지정합니다. *column-name*은 데이터 소스에 있는 테이블이나

뷰의 컬럼에 대한 페더레이티드 서버의 현재 이름입니다. *column-name*은 별칭의 기존 컬럼을 식별해야 합니다(SQLSTATE 42703). 동일한 ALTER NICKNAME문에서 동일한 컬럼 이름을 여러 번 참조할 수 없습니다(SQLSTATE 42711).

LOCAL NAME *column-name*

페더레이티드 서버가 변경된 컬럼을 참조할 때 사용하는 새 이름인 *column-name*을 지정합니다. 새 이름을 규정할 수 없으며, 별칭의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

LOCAL TYPE *local-data-type*

변경될 컬럼의 데이터 유형이 맵핑되는 새 로컬 데이터 유형을 지정합니다. 새 유형은 *local-data-type*으로 표시됩니다.

대부분의 랩퍼는 SQL 데이터 유형 중 일부만 지원합니다. 특정 데이터 유형에 대한 설명은『CREATE TABLE』문을 참조하십시오.

OPTIONS

COLUMN 키워드 뒤에 지정된 컬럼에 대해 추가, 설정 또는 삭제될 컬럼 옵션을 나타냅니다.

ADD

컬럼 옵션을 추가합니다.

SET

컬럼 옵션의 설정값을 변경합니다.

column-option-name

추가되거나 설정될 컬럼 옵션의 이름을 지정합니다.

string-constant

*column-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

DROP *column-option-name*

컬럼 옵션을 삭제합니다.

ADD *unique-constraint*

고유 제한조건을 정의합니다.『CREATE NICKNAME』문에 대한 설명을 참조하십시오.

ADD *referential-constraint*

참조 제한조건을 정의합니다.『CREATE NICKNAME』문에 대한 설명을 참조하십시오.

ADD *check-constraint*

점검 제한조건을 정의합니다.『CREATE NICKNAME』문에 대한 설명을 참조하십시오.

ALTER FOREIGN KEY *constraint-name*

참조 제한조건 *constraint-name*의 제한조건 속성을 변경합니다. 제한조건 속성에 대

한 설명은 『CREATE NICKNAME』문을 참조하십시오. *constraint-name*은 기존의 참조 제한조건을 식별해야 합니다(SQLSTATE 42704).

ALTER CHECK *constraint-name*

점검 제한조건 *constraint-name*의 제한조건 속성을 변경합니다. *constraint-name*은 기존의 점검 제한조건을 식별해야 합니다(SQLSTATE 42704).

constraint-alteration

참조 또는 점검 제한조건과 연관된 속성을 변경하기 위한 옵션을 제공합니다.

ENABLE QUERY OPTIMIZATION

적절한 환경에서 쿼리 최적화를 위해 제한조건을 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

제한조건을 쿼리 최적화에 사용할 수 없습니다.

DROP PRIMARY KEY

기본 키 정의 및 이 기본 키에 종속되는 모든 참조 제한조건을 삭제합니다. 별칭에는 기본 키가 있어야 합니다.

DROP FOREIGN KEY *constraint-name*

참조 제한조건의 *constraint-name*을 삭제합니다. *constraint-name*은 별칭에 정의된 기존의 참조 제한조건을 식별해야 합니다.

DROP UNIQUE *constraint-name*

고유 제한조건 *constraint-name*의 정의 및 이 고유 제한조건에 종속되는 모든 참조 제한조건을 삭제합니다. *constraint-name*은 기존의 고유 제한조건을 식별해야 합니다.

DROP CHECK *constraint-name*

점검 제한조건의 *constraint-name*을 삭제합니다. *constraint-name*은 별칭에 정의된 기존의 점검 제한조건을 식별해야 합니다.

DROP CONSTRAINT *constraint-name*

제한조건의 *constraint-name*을 삭제합니다. *constraint-name*은 별칭에 정의된 기존의 점검 제한조건, 참조 제한조건, 기본 키 또는 고유 제한조건을 식별해야 합니다.

ALLOW CACHING 또는 DISALLOW CACHING

페더레이티드 서버에서 해당 별칭에 대한 데이터가 캐시될 수 있는지 지정합니다.

ALLOW CACHING

페더레이티드 서버에서 해당 별칭에 대한 데이터가 캐시될 수 있도록 지정합니다.

DISALLOW CACHING

페더레이티드 서버에서 해당 별칭에 대한 데이터가 캐시될 수 없도록 지정합니다. 구체화된 쿼리 테이블 정의는 캐싱을 허용하지 않는 별칭을 참조할 수 없

으며 해당 절은 구체화된 쿼리 테이블 정의의 fullselect에서 참조되는 별칭에 대해 지정될 수 없습니다(SQLSTATE 42917).

규칙

- 뷰, SQL 메소드 또는 SQL 함수에 별칭이 사용되거나 정보용 제한조건이 정의된 경우, ALTER NICKNAME문을 사용하여 별칭에 있는 컬럼의 로컬 이름 또는 데이터 유형을 변경할 수 없습니다(SQLSTATE 42893). 그러나 이 명령문을 사용하여 컬럼 옵션, 별칭 옵션 또는 정보용 제한조건을 추가, 설정 또는 삭제할 수 있습니다.
- 구체화된 쿼리 테이블 정의가 별칭을 참조할 경우, ALTER NICKNAME문을 사용하여 로컬 이름, 데이터 유형, 컬럼 옵션 또는 별칭 옵션을 변경할 수 없습니다(SQLSTATE 42893). 또한 캐싱 사용을 불가능하게 하는 데 해당 명령문을 사용할 수 없습니다(SQLSTATE 42917). 그러나 이 명령문을 사용하여 정보용 제한조건을 추가, 변경 또는 삭제할 수 있습니다.
- 동일한 ALTER NICKNAME문에 컬럼 옵션을 두 번 이상 지정할 수 없습니다(SQLSTATE 42853). 컬럼 옵션이 활성화, 재설정 또는 삭제될 때 사용 중인 다른 컬럼 옵션은 영향을 받지 않습니다.
- 관계형 별칭의 경우, 다음과 같은 상황에서는 주어진 작업 단위(UOW) 내에서 ALTER NICKNAME문을 처리할 수 없습니다(SQLSTATE 55007).
  - 이 명령문에서 참조하는 별칭의 동일한 UOW에서 커서가 열려 있는 경우
  - 이 명령문에서 참조하는 별칭에 대해 동일한 UOW에서 이미 INSERT, DELETE 또는 UPDATE문이 발행된 경우
- 비관계형 별칭의 경우, 다음과 같은 상황에서는 주어진 작업 단위(UOW) 내에서 ALTER NICKNAME문을 처리할 수 없습니다(SQLSTATE 55007).
  - 이 명령문에서 참조하는 별칭의 동일한 UOW에서 커서가 열려 있는 경우
  - 이 명령문에서 참조하는 별칭이 동일한 UOW에 있는 SELECT문에 의해 이미 참조된 경우
  - 이 명령문에서 참조하는 별칭에 대해 동일한 UOW에서 이미 INSERT, DELETE 또는 UPDATE문이 발행된 경우

주

- ALTER NICKNAME문을 사용하여 별칭의 컬럼에 대한 로컬 이름을 변경하면, 해당 컬럼에 대한 쿼리는 새로운 이름으로 컬럼을 참조해야 합니다.
- 컬럼 데이터 유형의 로컬 스펙이 변경되면, 데이터베이스 관리 프로그램은 해당 컬럼에 대해 수집된 모든 통계(HIGH2KEY, LOW2KEY 등)를 무효화합니다.
- 데이터 소스 오브젝트가 보호되는 별칭에 대해 DISALLOW CACHING절을 지정하십시오. 이 지정은 별칭이 사용될 때마다 적절한 권한 부여 ID에 대한 데이터가 쿼리 실행시간에 데이터 소스로부터 리턴되게끔 합니다.

예:

예 1: 별칭 NICK1은 T1이라는 System i용 DB2 테이블을 참조합니다. 또한 COL1은 이 테이블의 첫 번째 컬럼인 C1을 참조하는 로컬 이름입니다. C1의 로컬 이름을 COL1에서 NEWCOL로 변경하십시오.

```
ALTER NICKNAME NICK1
  ALTER COLUMN COL1
  LOCAL NAME NEWCOL
```

*예 2:*별칭 EMPLOYEE는 EMP라는 z/OS용 DB2 테이블을 참조합니다. 또한 SALARY는 이 테이블의 컬럼 중 하나인 EMP_SAL을 참조하는 로컬 이름입니다. 컬럼의 데이터 유형 FLOAT는 로컬 데이터 유형 DOUBLE로 맵핑됩니다. FLOAT가 DECIMAL (10, 5)로 맵핑되도록 맵핑을 변경하십시오.

```
ALTER NICKNAME EMPLOYEE
  ALTER COLUMN SALARY
  LOCAL TYPE DECIMAL(10,5)
```

예 3: Oracle 테이블에서 데이터 유형이 VARCHAR인 컬럼은 뒤 공백을 가지지 않도록 나타내십시오. 테이블의 별칭은 NICK2이며, 컬럼의 로컬 이름은 COL1입니다.

```
ALTER NICKNAME NICK2
  ALTER COLUMN COL1
  OPTIONS (ADD VARCHAR_NO_TRAILING_BLANKS 'Y')
```

예 4: 별칭 DRUGDATA1의 테이블 구조 파일 drugdata1.txt에 대한 완전한 경로를 변경하십시오. FILE_PATH 별칭 옵션을 사용하여 경로를 현재 값인 '/user/pat/drugdata1.txt'에서 '/usr/kelly/data/drugdata1.txt'로 변경하십시오.

```
ALTER NICKNAME DRUGDATA1
  OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

ALTER PACKAGE

ALTER PACKAGE 명령문은 패키지를 바인드하거나 리바인드할 필요 없이 현재 서버에서 패키지의 바인드 옵션을 변경합니다.

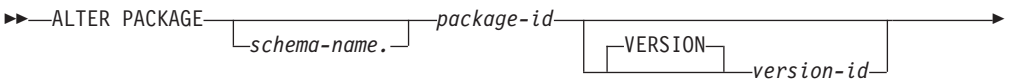
호출

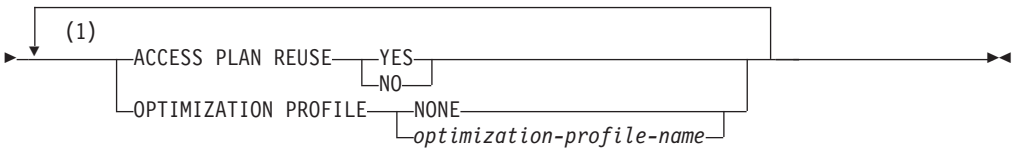
이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 스키마에서 ALTERIN 특권
- 패키지에 대한 BIND 특권
- DBADM 권한





주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

설명

schema-name.package-id

변경할 패키지를 식별합니다. 스키마 이름이 지정되지 않은 경우 디폴트 스키마가 패키지 ID를 내재적으로 규정합니다. 스키마 이름과 패키지 ID는 내재적으로 또는 명시적으로 지정된 버전 ID와 함께 현재 서버에 존재하는 패키지를 식별해야 합니다(SQLSTATE 42704).

VERSION *version-id*

변경할 패키지 버전을 식별합니다. 이 값을 지정하지 않은 경우, 버전은 빈 문자열로 디폴트됩니다. 패키지 이름은 동일하지만 버전이 다른 여러 개의 패키지가 있을 경우에는 ALTER PACKAGE문을 한 번 호출할 때마다 한 개의 패키지 버전만 변경할 수 있습니다. 다음과 같은 경우 큰따옴표를 사용하여 버전 ID를 구분하십시오.

- VERSION(AUTO) 프리컴파일러 옵션을 사용하여 생성한 경우
- 숫자로 시작되는 경우
- 소문자 또는 대소문자 혼용 문자가 있는 경우

운영 체제 명령 프롬프트에서 명령문을 호출한 경우, 각 큰따옴표 분리문자 앞에 백슬래시 문자를 사용하여 운영 체제가 분리문자를 없애지 못하게 하십시오.

ACCESS PLAN REUSE

쿼리 컴파일러가 추후 내재된 리바인드 및 명시적 리바인드 동안 패키지에서 정적 명령문의 액세스 플랜을 재사용하도록 해야 하는지 여부를 표시합니다.

NO

액세스 플랜을 재사용하지 않도록 지정합니다.

YES

액세스 플랜을 재사용하도록 지정합니다.

OPTIMIZATION PROFILE

최적화 프로파일이 있는 경우 패키지와 연관시킬 최적화 프로파일을 표시합니다.

NONE

최적화 프로파일을 패키지와 연관시키지 않습니다. 최적화 프로파일이 이미 패키지와 연관되 경우 연관이 제거됩니다.

optimization-profile-name

최적화 프로파일 *optimization-profile-name*을 패키지와 연관시킵니다. 최적화 프로파일은 두 부분으로 구성된 이름입니다. 지정된 *optimization-profile-name*이 규정되지 않은 경우, CURRENT DEFAULT SCHEMA 특수 레지스터의 값은 내재된 규정자로 사용됩니다. 최적화 프로파일이 이미 패키지와 연관된 경우 연관이 *optimization-profile-name*로 교체됩니다.

ALTER PACKAGE 명령문으로 리바인드가 발생되지 않으므로 명령문의 쿼리 실행 계획은 다음 내재적 리바인드 또는 명시적 리바인드까지 영향받지 않습니다. 그러나 일단 ALTER PACKAGE 명령문이 커미트되면 최적화 프로파일의 값은 패키지를 사용하여 발행된 동적 DML 명령문을 컴파일하기 위한 CURRENT OPTIMIZATION PROFILE 특수 레지스터의 디폴트값으로 사용됩니다. ALTER PACKAGE문은 최적화 프로파일을 처리하지 않지만, 이름이 구문적으로 유효한지만 확인합니다. 최적화 프로파일은 패키지를 사용하여 발행된 DML 명령문이 최적화되는 다음 번에 처리됩니다.

주

- **카탈로그 뷰 값은 패키지에 적용되는 설정을 반영하지 않을 수도 있습니다.** 이 명령문은 패키지 리바인드를 트리거하지 않으므로 SYSCAT.PACKAGES 카탈로그 뷰

에 표시된 대로 패키지 설정에 마지막 BIND 또는 REBIND 동안 실제로 적용된 것이 반영되지 않을 수 있습니다. ALTER_TIME이 LAST_BIND_TIME보다 크면 이것이 case가 될 수 있습니다.

- *호환성:* BIND 및 REBIND 명령과의 호환성:
  - ACCESS PLAN REUSE 대신 APREUSE를 지정할 수 있습니다.
  - OPTIMIZATION PROFILE 대신 OPTPROFILE을 지정할 수 있습니다.

예

*예 1:*TRUUVERT.EMPADMIN 패키지에 액세스 플랜 재사용이 가능하도록 합니다.

```
ALTER PACKAGE TRUUVERT.EMPADMIN ACCESS PLAN REUSE YES
```

*예 2:*TRUUVERT.EMPADMIN 패키지에 액세스 플랜 재사용이 가능하게 된 것으로 가정합니다. 또한 AYYANG.INDEXHINTS 최적화 프로파일에 패키지 내의 특정 명령문의 명령문 프로파일이 포함된 것으로 가정합니다. 최적화 프로파일을 이 패키지와 연관시키면 최적화 프로파일이 명령문의 액세스 플랜 재사용을 겹쳐씁니다.

```
ALTER PACKAGE TRUUVERT.EMPADMIN OPTIMIZATION PROFILE AYYANG.INDEXHINTS
```

명령문이 커미트된 후 동적 명령문이 적용되고 정적 명령문은 다음 리바인드에서 적용됩니다. 패키지가 리바인드되면, 쿼리 컴파일러는 최적화 프로파일로 식별되는 명령문을 제외한 패키지에서 모든 정적 명령문의 액세스 플랜을 재사용하려고 시도합니다. 이 명령문을 재컴파일하면 쿼리 컴파일러가 대신 명령문 프로파일 적용을 시도합니다.

예 3: 다음의 결과로 TRUUVERT.EMPADMIN 패키지와 연관된 최적화 프로파일이 없게 됩니다.

```
ALTER PACKAGE TRUUVERT.EMPADMIN OPTIMIZATION PROFILE NONE
```

ALTER PROCEDURE(외부)

ALTER PROCEDURE(외부)문은 프로시저의 등록 정보를 변경하여 기존 외부 프로시저를 수정합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 프로시저 스키마에 대한 ALTERIN 특권
- SYSCAT.ROUTINES 카탈로그 뷰의 OWNER 컬럼에 기록된 프로시저의 소유자
- DBADM 권한

프로시저의 EXTERNAL NAME을 변경하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

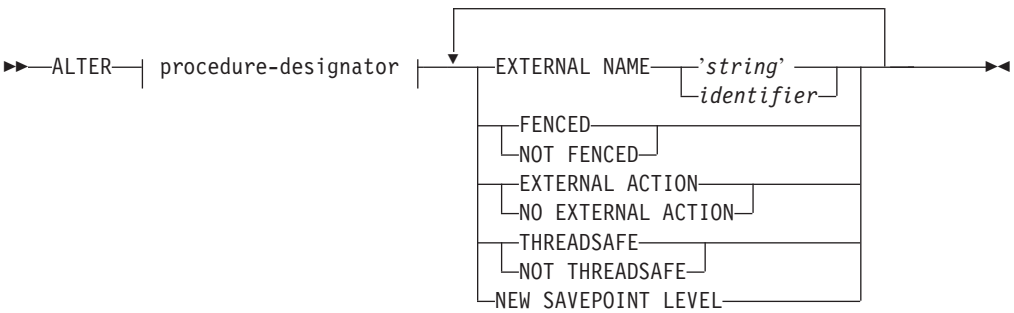
- 데이터베이스에 있는 CREATE_EXTERNAL_ROUTINE 권한
- DBADM 권한

프로시저를 분리(fenced)되지 않게 변경하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스에서의 CREATE_NOT_FENCED_ROUTINE 권한
- DBADM 권한

프로시저가 분리되도록 변경하려면 추가 권한이나 특권이 필요하지 않습니다.

구문



설명

procedure-designator

변경할 프로시저를 식별합니다. *procedure-designator*는 현재 서버에 존재하는 프로시저를 식별해야 합니다. 프로시저 소유자와 프로시저에 대한 모든 특권이 보존됩니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

EXTERNAL NAME *'string'* **또는** *identifier*

프로시저를 구현하는 사용자 작성 코드의 이름을 식별합니다.

FENCED 또는 NOT FENCED

프로시저가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 어드레스 스페이스에서 실행되어도 안전한지(FENCED) 또는 안전하지 않은지(NOT FENCED) 여부를 지정합니다. 대부분의 프로시저는 FENCED나 NOT FENCED로 실행될 수 있습니다.

프로시저가 FENCED로 변경된 경우, 데이터베이스 관리 프로그램은 내부 자원(예: 데이터 버퍼)을 프로시저가 액세스하지 못하도록 분리시킵니다. 일반적으로 FENCED로 수행되는 프로시저는 NOT FENCED로 실행되는 프로시저와는 다르게 실행됩니다.

주의:

제대로 코딩, 검토 및 테스트되지 않은 프로시저에 NOT FENCED를 사용하면 DB2 데이터베이스의 무결성이 손상될 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 스토어드 프로시저가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

NOT THREADSAFE로 선언된 프로시저는 NOT FENCED로 변경할 수 없습니다(SQLSTATE 42613).

프로시저에 AS LOCATOR로 정의된 매개변수가 있고 프로시저가 NO SQL 옵션을 사용하여 정의된 경우에는 프로시저를 FENCED로 변경할 수 없습니다(SQLSTATE 42613).

LANGUAGE OLE 또는 CLR 프로시저의 경우 이 옵션을 변경할 수 없습니다(SQLSTATE 42849).

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

프로시저가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지(EXTERNAL ACTION) 아니면 취하지 않는지(NO EXTERNAL ACTION) 여부를 지정합니다. NO EXTERNAL ACTION을 지정하면, 시스템은 프로시저에 외부적 영향이 없는 것으로 간주하는 특정 최적화를 사용할 수 있습니다.

THREADSAFE 또는 NOT THREADSAFE

프로시저가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 아니면 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

프로시저가 OLE가 아닌 다른 LANGUAGE로 정의된 경우에는 다음과 같습니다.

- 프로시저가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 프로시저를 호출할 수 있습니다. 일반적으로 Threadsafe 프로시저가 되려면 프로시저가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED 및 NOT FENCED 프로시저는 둘 다 THREADSAFE될 수 있습니다.
- 프로시저가 NOT THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 동일한 프로세스에서 프로시저를 호출할 수 없습니다. 분리(fenced) 프로시저만 NOT THREADSAFE될 수 있습니다(SQLSTATE 42613).

LANGUAGE OLE 프로시저의 경우 이 옵션을 변경할 수 없습니다(SQLSTATE 42849).

NEW SAVEPOINT LEVEL

프로시저에 대해 새 세이브포인트 레벨을 작성하도록 지정합니다. 세이브포인트 레벨은 세이브포인트 관련 명령문에 대한 참조 범위는 물론 세이브포인트 이름의 참조 및 비교에 사용되는 이름 스페이스를 의미합니다.

프로시저의 세이브포인트 레벨은 NEW SAVEPOINT LEVEL로만 변경할 수 있습니다.

규칙

- SYSIBM, SYSFUN 또는 SYSPROC 스키마에 있는 프로시저는 변경할 수 없습니다(SQLSTATE 42832).

예 :

프로시저 PARTS_ON_HAND()를 분리(fenced)되지 않도록 변경하십시오.

```
ALTER PROCEDURE PARTS_ON_HAND() NOT FENCED
```

ALTER PROCEDURE(전래)

ALTER PROCEDURE(전래)문은 소싱된 프로시저의 하나 이상의 매개변수 데이터 유형을 변경하여 기존 소싱된 프로시저를 수정합니다.

호출

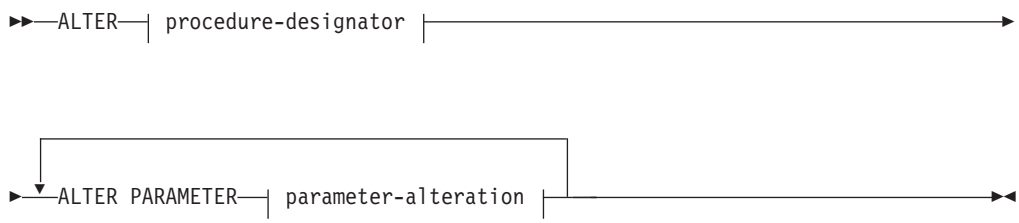
이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

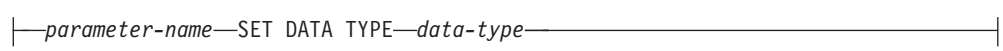
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 프로시저 스키마에 대한 ALTERIN 특권
- SYSCAT.ROUTINES 카탈로그 뷰의 OWNER 컬럼에 기록된 프로시저의 소유자
- DBADM 권한

구문



parameter-alteration:



설명

procedure-designator

변경할 프로시저를 고유하게 식별합니다. 식별된 프로시저는 소싱된 프로시저여야 합니다(SQLSTATE 42849). 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

parameter-name

변경될 매개변수를 식별합니다. *parameter-name*은 프로시저의 기존 매개변수를 식별해야 합니다(SQLSTATE 42703). 이름은 동일한 ALTER PROCEDURE문에서 변경되는 매개변수를 식별해서는 안됩니다(SQLSTATE 42713).

data-type

매개변수의 새 로컬 데이터 유형을 지정합니다. CREATE TABLE문의 *data-type*

정의에 대해 유효한 SQL 데이터 유형 스펙 및 약어를 지정할 수 있습니다. BLOB, CLOB, DBCLOB, DECFLOAT, XML, REFERENCE 및 사용자 정의 유형은 지원되지 않습니다(SQLSTATE 42815).

예 :

리모트 Oracle 프로시저인 'EMPLOYEE'에 대해 페더레이티드 프로시저 FEDEMPLOYEE가 작성되었다고 가정합니다. 입력 매개변수 SALARY의 데이터 유형은 DB2에서 DOUBLE(8)에 맵핑됩니다. 이 매개변수의 데이터 유형을 DECIMAL(5,2)로 변경하십시오.

```
ALTER PROCEDURE FEDEMPLOYEE
  ALTER PARAMETER SALARY
  SET DATA TYPE DECIMAL(5,2)
```

ALTER PROCEDURE(SQL)

ALTER PROCEDURE(SQL)문은 프로시저의 등록 정보를 변경하여 기존 SQL 프로시저를 수정합니다.

호출

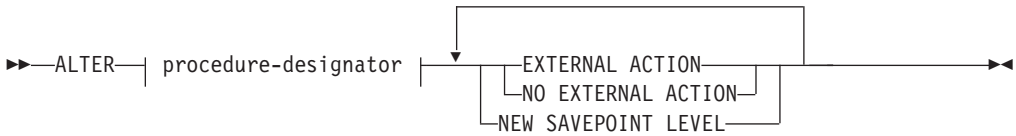
이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 프로시저 스키마에 대한 ALTERIN 특권
- SYSCAT.ROUTINES 카탈로그 뷰의 OWNER 컬럼에 기록된 프로시저의 소유자
- DBADM 권한

구문



설명

procedure-designator

변경할 프로시저를 식별합니다. *procedure-designator*는 현재 서버에 있는 프로시저를 식별해야 합니다. 프로시저의 소유자와 프로시저의 모든 특권은 보존됩니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

프로시저가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지(EXTERNAL ACTION) 아니면 취하지 않는지(NO EXTERNAL ACTION) 여부를 지정합니다. NO EXTERNAL ACTION을 지정하면, 시스템은 프로시저에 외부적 영향이 없는 것으로 간주하는 특정 최적화를 사용할 수 있습니다.

NEW SAVEPOINT LEVEL

프로시저에 대해 새 세이브포인트 레벨을 작성하도록 지정합니다. 세이브포인트 레벨은 세이브포인트 관련 명령문에 대한 참조 범위는 물론 세이브포인트 이름의 참조 및 비교에 사용되는 이름 스페이스를 의미합니다.

프로시저의 세이브포인트 레벨은 NEW SAVEPOINT LEVEL로만 변경할 수 있습니다.

규칙

- SYSIBM, SYSFUN 또는 SYSPROC 스키마에 있는 프로시저는 변경할 수 없습니다(SQLSTATE 42832).

예 :

외부 조치가 없음을 표시하기 위해 프로시저 MEDIAN_RESULT_SET를 변경하십시오.

```
ALTER PROCEDURE MEDIAN_RESULT_SET(DOUBLE)
  NO EXTERNAL ACTION
```

ALTER SECURITY LABEL COMPONENT

ALTER SECURITY LABEL COMPONENT문은 보안 레이블 구성요소를 수정합니다.

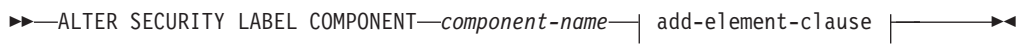
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

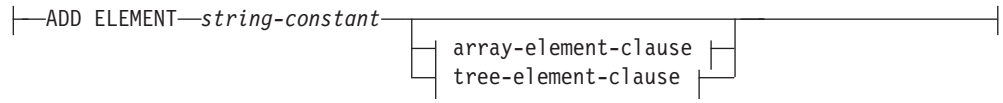
권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

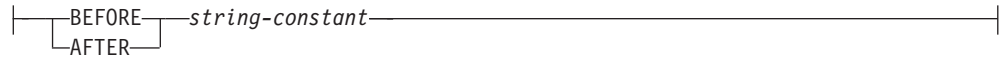
구문



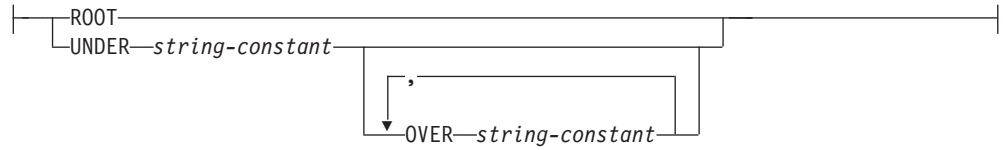
add-element-clause:



array-element-clause:



tree-element-clause:



설명

component-name

변경할 보안 레이블 구성요소의 이름을 지정합니다. 이름 지정된 구성요소는 현재 서버에 존재해야 합니다(SQLSTATE 42704).

ADD ELEMENT

보안 레이블 구성요소에 추가될 요소를 지정합니다. *array-element-clause* 및 *tree-element-clause*를 지정하지 않으면 요소는 세트 구성요소에 추가됩니다.

string-constant

보안 레이블 구성요소에 대한 유효값 세트에 추가될 문자열 상수 값입니다. 값은 보안 레이블 구성요소의 유효값 세트에 있는 다른 값과 같으면 안됩니다(SQLSTATE 42713).

BEFORE 또는 AFTER

배열 구성요소의 경우, 보안 레이블 구성요소에 대한 순서화된 요소 값 세트에서 요소가 추가될 위치를 지정합니다.

BEFORE

추가될 요소는 식별된 기존 요소 바로 앞으로 순위가 지정됩니다.

AFTER

추가될 요소는 식별된 기존 요소 바로 다음으로 순위가 지정됩니다.

string-constant

배열 구성요소에서 기존 요소의 문자열 상수 값을 지정합니다(SQLSTATE 42704).

ROOT 또는 UNDER

트리 구성요소의 경우, 보안 레이블 구성요소에 대한 노드 요소 값 구조에서 요소가 추가될 위치를 지정합니다.

ROOT

추가될 요소는 식별된 트리의 루트 노드로 간주됩니다.

UNDER *string-constant*

추가될 요소는 *string-constant*로 식별되는 요소의 인접 하위 요소입니다. *string-constant* 값은 트리 구성요소의 기존 요소여야 합니다(SQLSTATE 42704).

OVER *string-constant*,...

추가될 요소는 *string-constant* 값 목록으로 식별되는 모든 요소의 인접 하위 요소입니다. 각 *string-constant* 값은 트리 구성요소의 기존 요소여야 합니다(SQLSTATE 42704).

규칙

- 요소 이름은 다음 문자를 포함할 수 없습니다(SQLSTATE 42601).
  - 여는 괄호 - (
  - 닫는 괄호 -)
  - 쉼표 - ,
  - 콜론 - :
- 요소 이름은 32바이트를 초과할 수 없습니다(SQLSTATE 42622).

- 보안 레이블 구성요소가 세트 또는 트리이면, 이 구성요소에 포함될 수 있는 요소가 64개 이하입니다.
- 구성요소가 배열이면, 이 구성요소는 총 요소 수가 배열 유형의 보안 레이블 구성요소를 작성할 때 지정할 수 있는 총 요소 수와 일치하는 배열에 도달하거나 도달하지 못할 수 있습니다(65 535). DB2는 새 요소가 추가되는 간격 내에서 새 요소에 인코드된 값을 지정합니다. 배열 구성요소에 요소를 추가할 때 따르는 패턴에 따라, 특정 간격 내에서 지정될 수 있는 가능한 값 수는 몇 개의 요소가 해당 간격에 삽입되는 경우 빠르게 소비될 수 있습니다.
- BEFORE 및 AFTER는 배열인 보안 레이블 구성요소에 대해서만 지정해야 합니다 (SQLSTATE 42613).
- ROOT 및 UNDER는 트리인 보안 레이블 구성요소에 대해서만 지정해야 합니다 (SQLSTATE 42613).

주

- 세트 구성요소의 경우 세트에서 요소 순서가 지정되지 않습니다.

예:

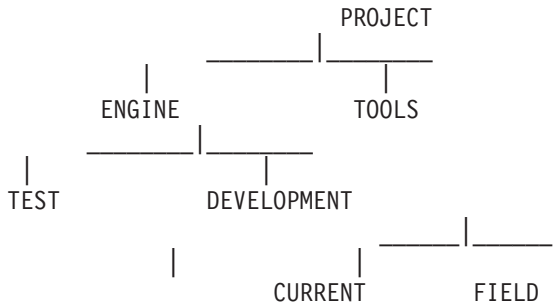
예 1: 요소 'Secret' 및 'Classified' 사이에 'High classified' 요소를 LEVEL 보안 배열 구성요소에 추가하십시오.

```
ALTER SECURITY LABEL COMPONENT LEVEL
  ADD ELEMENT 'High classified' BEFORE 'Classified'
```

예 2: 요소 'Funding'을 COMPARTMENTS 보안 레이블 세트 구성요소에 추가하십시오.

```
ALTER SECURITY LABEL COMPONENT COMPARTMENTS
  ADD ELEMENT 'Funding'
```

예 3: 요소 'ENGINE' 및 'TOOLS'를 GROUPS 보안 레이블 배열 구성요소에 추가하십시오. 다음 다이어그램은 새 요소가 배치될 위치를 표시합니다.




```
ALTER SECURITY LABEL COMPONENT GROUPS
  ADD ELEMENT 'TOOLS' UNDER 'PROJECT'
```

```
ALTER SECURITY LABEL COMPONENT GROUPS
  ADD ELEMENT 'ENGINE' UNDER 'PROJECT'
    OVER 'TEST', 'DEVELOPMENT'
```

ALTER SECURITY POLICY

ALTER SECURITY POLICY문은 보안 규정을 수정합니다.

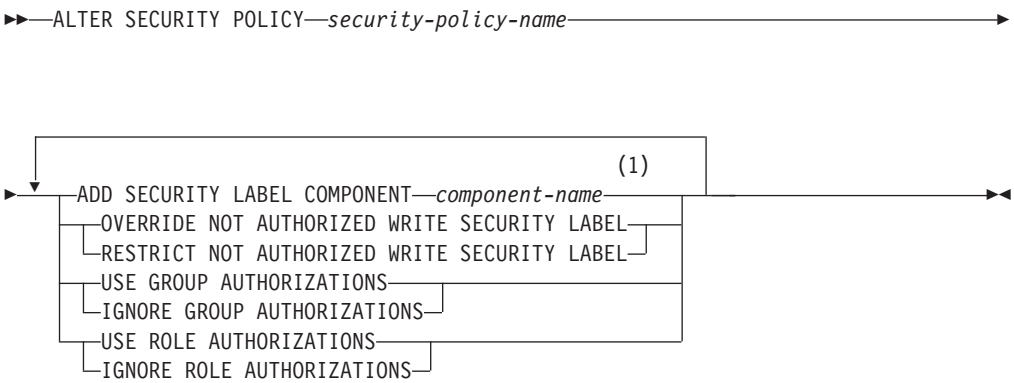
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



주:

1 ADD SECURITY LABEL COMPONENT 절만 두 번 이상 지정할 수 있습니다.

설명

security-policy-name

변경할 보안 규정의 이름을 지정합니다. 이름은 현재 서버의 기존 보안 규정을 식별해야 합니다(SQLSTATE 42710).

ADD SECURITY LABEL COMPONENT *component-name*

보안 규정에 보안 레이블 구성요소를 추가합니다. 동일한 보안 구성요소를 보안 규정에 대해 두 번 이상 지정할 수 없습니다(SQLSTATE 42713). 보안 규정은 현재 테이블에서 사용될 수 없습니다(SQLSTATE 42893).

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 또는 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

사용자에게 해당 보안 규정으로 보호되는 테이블에 대해 실행된 INSERT 또는 UPDATE 문에 제공된 명시적으로 지정된 보안 레이블을 작성할 권한이 없을 때

취할 조치를 지정합니다. 사용자의 보안 레이블 및 면제 증명서는 명시적으로 제공된 보안 레이블을 작성하는 사용자의 권한을 판별합니다.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
명시적으로 지정된 보안 레이블이 아닌 사용자의 보안 레이블 값을 삽입 또는 갱신 조작 도중 쓰기 액세스에 사용됨을 표시합니다.

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
사용자에게 INSERT 또는 UPDATE문에 제공된 명시적으로 지정된 보안 레이블을 작성할 권한이 없을 경우 삽입 또는 갱신 조작이 실패함을 표시합니다 (SQLSTATE 42519).

USE GROUP AUTHORIZATION 또는 IGNORE GROUP AUTHORIZATION
그룹에 직접 또는 간접적으로 부여된 면제 및 보안 레이블이 액세스 시도에 고려되는지 여부를 지정합니다.

USE GROUP AUTHORIZATION
그룹에 직접 또는 간접적으로 부여된 면제 및 보안 레이블이 고려됨을 표시합니다.

IGNORE GROUP AUTHORIZATION
그룹에 직접 또는 간접적으로 부여된 면제 및 보안 레이블이 고려되지 않음을 표시합니다.

USE ROLE AUTHORIZATION 또는 IGNORE ROLE AUTHORIZATION
역할에 직접 또는 간접적으로 부여된 면제 및 보안 레이블이 액세스 시도에 고려되는지 여부를 지정합니다.

USE ROLE AUTHORIZATION
역할에 직접 또는 간접적으로 부여된 면제 및 보안 레이블이 고려됨을 표시합니다.

IGNORE ROLE AUTHORIZATION
역할에 직접 또는 간접적으로 부여된 면제 및 보안 레이블이 고려되지 않음을 표시합니다.

규칙

- 사용자가 직접 쓰기 액세스에 대한 보안 레이블을 보유하지 않는 경우 다음 상황에서 오류가 리턴됩니다(SQLSTATE 42519).
  - 행 보안 레이블 컬럼의 값이 SQL문의 일부로 명시적으로 제공되지 않습니다.
  - 보안 규정에 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션이 적용 중인데 사용자가 제공된 보안 레이블로 데이터 오브젝트를 작성할 수 없습니다.

주

- 새 구성요소는 수정된 규정으로 포함된 기존 보안 레이블 정의의 끝에 논리적으로 추가됩니다. 해당 보안 규정에 대해 정의된 기존 보안 레이블은 구성요소에 대한 해당 값에 요소가 없는 정의의 일부로 새 구성요소를 포함하도록 수정됩니다.
- ***NOT AUTHORIZED WRITE SECURITY LABEL 변경 시 캐시 무효화:*** NOT AUTHORIZED WRITE SECURITY LABEL을 새 값으로 변경하면 변경되는 보안 규정으로 보호받는 테이블에 종속하는 캐시된 동적 또는 정적 SQL문이 무효화됩니다.
- 세션 권한 부여 ID는 레이블 기반 액세스 제어(LBAC)의 촛점 권한 부여 ID이므로, 그룹을 통해 액세스 가능한 역할이나 그룹에 부여된 보안 레이블이 정적 SQL을 비롯한 모든 유형의 SQL문에 대해 고려할 수 있습니다.
- 읽기 또는 쓰기 액세스 시도 시 연관된 그룹 또는 역할을 가지고 있는 사용자가 두 개 이상의 보안 레이블이나 면제를 사용할 수 있는 경우 다음 규칙을 기초로 해당 보안 레이블 및 면제의 적합성을 평가합니다.
  - 보안 규정이 역할 권한만 고려할 수 있는 경우, 권한 부여 ID가 직접 또는 간접 구성원인 역할에 부여된 모든 보안 레이블 및 면제가 고려됩니다. 멤버쉽이 사용자 권한 부여 ID와 연관된 그룹을 통해서만 액세스 가능한 역할에 부여된 보안 레이블 및 면제는 고려되지 않습니다.
  - 보안 규정이 그룹 권한만 고려할 수 있는 경우, 사용자 권한 부여 ID와 연관된 그룹에 부여된 모든 보안 레이블 및 면제가 고려됩니다. 멤버쉽이 사용자 권한 부여 ID와 연관된 그룹을 통해서만 액세스 가능한 역할에 부여된 보안 레이블 및 면제는 고려되지 않습니다.
  - 보안 규정이 그룹 및 역할 권한 모두를 고려할 수 있는 경우, 사용자 권한 부여 ID와 연관된 그룹을 통해 간접적으로 사용자에 액세스할 수 있는 역할에 부여된 모든 보안 레이블 및 면제가 고려됩니다.
  - PUBLIC을 통해서만 사용자에 액세스할 수 있는 역할 권한은 언제나 고려되지 않습니다.
- 액세스 시도 중 두 개 이상의 보안 레이블을 고려할 수 있는 경우, 보안 레이블마다 제공되는 값은 개별 구성요소 레벨에서 병합되어 보안 규정의 각 구성요소 조각에서 사용 가능한 모든 값의 조합을 반영하는 보안 레이블을 형성합니다. 이는 액세스 시도에 사용될 보안 레이블 값입니다.

  보안 레이블 조합 메커니즘은 구성요소 유형별로 다릅니다. 결과 보안 레이블은 다음과 같습니다.
  - 세트 구성요소에는 적합한 보안 레이블에서 발견된 모든 고유한 값의 통합이 포함됩니다.
  - 배열 구성요소는 적합한 보안 레이블에서 발견된 최상위 순서 요소를 포함합니다.

  - 트리 구성요소에는 적합한 보안 레이블에서 발견된 모든 고유한 값의 통합이 포함됩니다.
- 액세스 시도 중 두 개 이상의 면제를 고려할 수 있는 경우, 발견된 모든 면제가 액세스 시도에 적용됩니다.

예:

예 1: 새 구성요소 REGION을 추가하도록 보안 규정 DATA_ACCESS를 변경하십시오.

```
ALTER SECURITY POLICY DATA_ACCESS
  ADD COMPONENT REGION
```

예 2: 역할에 부여된 보안 레이블을 통해 액세스할 수 있도록 보안 규정 DATA_ACCESS를 변경하십시오.

```
ALTER SECURITY POLICY DATA_ACCESS
  USE ROLE AUTHORIZATIONS
```

예 3: 보안 규정에서의 그룹 또는 역할 권한 설정에 따라 고려될 보안 레이블을 표시하십시오. 보안 규정 SECUR_POL에는 다음 요소로 구성되는 배열 구성요소와 세트 구성요소가 있습니다.

배열 = {TS, S, C, U}

세트 = {A, B, X, Y}

다음 보안 레이블이 SECUR_POL에 대해 정의됩니다.

보안 레이블 L1 = C:A

보안 레이블 L2 = S:B

보안 레이블 L3 = TS:X

보안 레이블 L4 = U:Y

사용자 Paul은 역할 R1 및 그룹 G1의 구성원입니다. 그룹 G1은 역할 R2의 구성요소입니다. 보안 레이블 L1이 Paul에 부여됩니다. 보안 레이블 L2는 역할 R1에 부여됩니다. 보안 레이블 L3은 그룹 G1에 부여됩니다. 보안 레이블 L4는 역할 R2에 부여됩니다. 다음 표는 보안 규정 SECUR_POL의 가능한 여러 설정에 따라 Paul이 액세스 시도에 대해 고려할 보안 레이블을 표시합니다.

표 11. 보안 규정 설정의 함수로 고려할 보안 레이블

	역할 사용 가능	역할 사용 불가능
그룹 사용 가능	L1, L2, L3, L4	L1, L3
그룹 사용 불가능	L1, L2	L1

다음 표는 보안 규정 SECUR_POL의 여러 설정에 따라 Paul의 액세스 시도에 대해 조합된 보안 레이블의 값을 표시합니다.

표 12. 보안 규정 설정의 함수로서 조합된 보안 레이블

	역할 사용 가능	역할 사용 불가능
그룹 사용 가능	TS:(A, B, X, Y)	TS:(A, X)
그룹 사용 불가능	S:(A, B)	C:A

ALTER SEQUENCE

ALTER SEQUENCE문을 사용하여 다음 방법 중 하나로 시퀀스를 변경할 수 있습니다.

- 시퀀스 재시작
- 이후 시퀀스 값 간의 증분 변경
- 최소값 또는 최대값 설정 또는 제거
- 캐시 시퀀스 번호 수 변경
- 시퀀스의 순환 여부를 결정하는 속성 변경
- 시퀀스 번호를 요청 순서대로 생성할지 여부 변경

호출

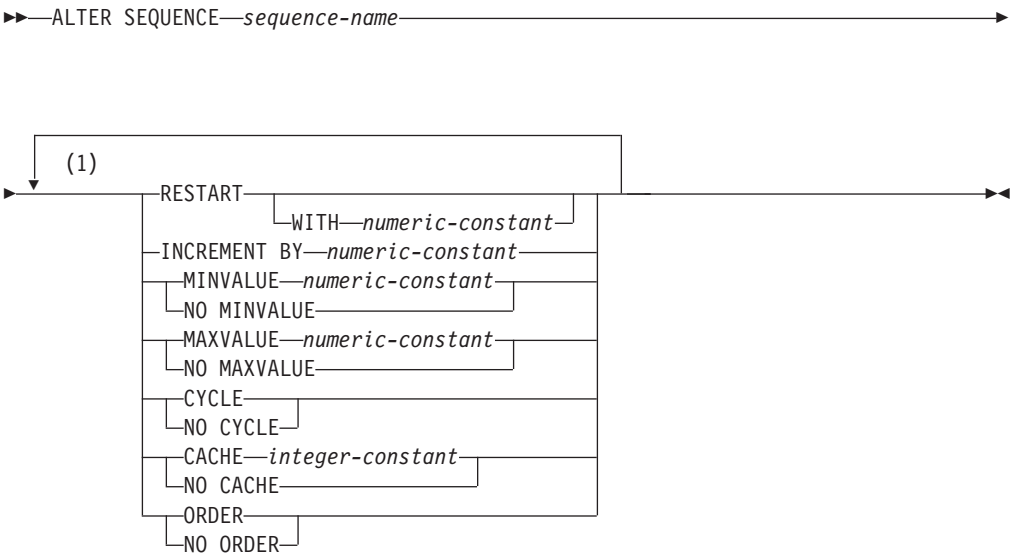
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 변경될 시퀀스에 대한 ALTER 특권
- 내재적 또는 명시적으로 지정된 스키마에 대한 ALTERIN 특권
- DBADM 권한

구문



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

설명

sequence-name

변경할 시퀀스를 식별합니다. 내재적 또는 명시적 스키마 규정자를 비롯한 이름은 현재 서버에 있는 기존 시퀀스를 고유하게 식별해야 합니다. 명시적 또는 내재적으로 지정된 스키마에 이 이름의 시퀀스가 없으면, 오류가 리턴됩니다(SQLSTATE 42704). *sequence-name*은 식별 컬럼에 대해 시스템이 생성하는 시퀀스가 아니어야 합니다(SQLSTATE 428FB).

RESTART

시퀀스를 재시작합니다. *numeric-constant*를 지정하지 않으면, 시퀀스를 작성한 원래 CREATE SEQUENCE문에 내재적으로 또는 명시적으로 시작값으로 지정된 값에서 시퀀스가 재시작됩니다.

WITH *numeric-constant*

지정된 값으로 시퀀스를 재시작합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않습니다(SQLSTATE 428FA).

INCREMENT BY *numeric-constant*

시퀀스의 연속되는 값의 간격을 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있습니다(SQLSTATE 42815). 이 값은 큰 정수 상수의 값을 초과하지 않고(SQLSTATE 42820) 소수점 오른쪽에 0이 아닌 자릿수를 갖지 않아야 합니다(SQLSTATE 428FA).

이 값이 음수이면, 내림차순입니다. 이 값이 0이거나 양수인 경우, 값은 ALTER문 이후에 오름차순이 됩니다.

MINVALUE 또는 NO MINVALUE

내림차순 시퀀스가 값 생성을 순환하거나 중지하여 최소값을 지정합니다. 또는 오름차순 시퀀스가 최대값에 도달한 후 순환하여 최소값을 지정합니다.

MINVALUE *numeric-constant*

최소값인 숫자 상수를 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최대값보다 작거나 같아야 합니다(SQLSTATE 42815).

NO MINVALUE

오름차순 시퀀스의 경우, 원래의 시작값입니다. 내림차순 시퀀스의 경우, 시퀀스와 연관된 데이터 유형의 최소값입니다.

MAXVALUE 또는 NO MAXVALUE

오름차순 시퀀스가 값 생성을 순환하거나 중지하여 최대값을 지정합니다. 또는 내림차순 시퀀스가 최소값에 도달한 후 순환하여 최대값을 지정합니다.

MAXVALUE *numeric-constant*

최대값인 숫자 상수를 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최소값보다 크거나 같아야 합니다(SQLSTATE 42815).

NO MAXVALUE

오름차순 시퀀스의 경우, 시퀀스와 연관된 데이터 유형의 최대값입니다. 내림차순 시퀀스의 경우, 원래의 시작값입니다.

CYCLE 또는 NO CYCLE

시퀀스가 최대값 또는 최소값에 도달한 이후에도 계속 값을 생성할지 여부를 지정합니다. 시퀀스의 바운더리는 바운더리 조건에 정확하게 도착하는 다음 값이나 값을 지나쳐서 도달될 수 있습니다.

CYCLE

최대값 또는 최소값에 도달된 이후에도 이 시퀀스에 대해 값을 계속 생성하도록 지정합니다. 이 옵션을 사용할 경우, 오름차순 시퀀스가 최대값에 도달하면 최소값이 생성되고, 내림차순 시퀀스가 최소값에 도달하면 최대값이 생성됩니다. 시퀀스의 최대값과 최소값에 따라 순환에 사용되는 범위가 결정됩니다.

CYCLE이 효력이 있으면 DB2가 시퀀스에 대해 중복 값을 생성할 수 있습니다.

NO CYCLE

시퀀스의 최대값이나 최소값에 도달한 이후에는 시퀀스에 대해 값이 생성되지 않도록 지정합니다.

CACHE 또는 NO CACHE

보다 빠른 액세스를 위해 메모리에 사전 할당된 값 일부를 보존할지 여부를 지정합니다. 이것은 성능 및 조정 옵션입니다.

CACHE *integer-constant*

사전 할당되고 메모리에 보존되는 최대 시퀀스 값을 지정합니다. 캐시에 값을 사전 할당하고 저장하면 시퀀스에 대한 값이 생성될 때 로그에 대한 동기 입출력이 줄어듭니다.

시스템 오류가 발생할 경우, 커미트된 명령문에서 사용되지 않은 캐시된 모든 시퀀스 값이 없어집니다. 즉, 사용되지 않습니다. CACHE 옵션에 대해 지정된 값은 시스템 오류 시 없어질 수 있는 최대 시퀀스 값입니다.

최소값은 2입니다(SQLSTATE 42815).

NO CACHE

시퀀스 값이 사전 할당되지 않도록 지정합니다. 따라서 시스템 오류, 시스템 종료 또는 데이터베이스 비활성화의 경우에 값의 유실되지 않습니다. 이 옵션을 지정하면 시퀀스의 값이 캐시에 저장되지 않습니다. 이 경우, 새 시퀀스 값에 대한 요청이 있을 때마다 동기 입출력이 로그에 기록됩니다.

ORDER 또는 NO ORDER

시퀀스 번호를 요청 순서대로 생성할지 여부를 지정합니다.

ORDER

시퀀스 번호가 요청 순서대로 생성되도록 지정합니다.

NO ORDER

시퀀스 번호가 요청 순서대로 생성되지 않도록 지정합니다.

주

- 이후 시퀀스 번호만 ALTER SEQUENCE문의 영향을 받습니다.
- 시퀀스의 데이터 유형은 변경할 수 없습니다. 대신, 새 시퀀스에 대해 원하는 데이터 유형을 지정하는 시퀀스를 삭제 및 재작성하십시오.
- 모든 캐시 값은 시퀀스가 변경될 때 유실됩니다.
- 시퀀스를 재시작하거나 CYCLE로 변경한 이후, 시퀀스 번호가 이전 시퀀스에 의해 생성된 값과 중복될 수 있습니다.
- *호환성*: 이전 버전 DB2와의 호환성:
  - 여러 옵션을 구분하는 데 쉼표(,)를 사용할 수 있습니다.

  다음 구문도 지원됩니다.

  - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE 및 NOORDER

예:

예 1: 숫자 값없이 RESTART를 지정하는 이유는 시퀀스를 START WITH 값으로 재설정하기 위한 것입니다. 이 예의 목표는 1부터 테이블의 행 수까지 숫자를 생성한 후 임시 테이블을 사용하여 테이블에 추가된 컬럼에 숫자를 삽입하는 것입니다. 다른 용도는 모든 결과 행에 번호가 지정되는 결과를 다시 얻는 것입니다.

```
ALTER SEQUENCE ORG_SEQ RESTART
SELECT NEXT VALUE FOR ORG_SEQ, ORG.* FROM ORG
```

ALTER SERVER

ALTER SERVER문은 다음을 수행하는 데 사용됩니다.

- 특정 데이터 소스의 정의나 데이터 소스의 범주 정의 수정
- 특정 데이터 소스의 구성이나 데이터 소스의 범주 구성 변경. 변경사항은 페더레이티드 데이터베이스에 대한 여러 연결에서 유지됩니다.

이 명령문에서 SERVER라는 단어와 *server*-로 시작하는 매개변수 이름은 페더레이티드 시스템에 있는 데이터 소스만 참조하며, 페더레이티드 시스템에 있는 페더레이티드 서버나 DRDA® 응용프로그램 서버는 참조하지 않습니다.

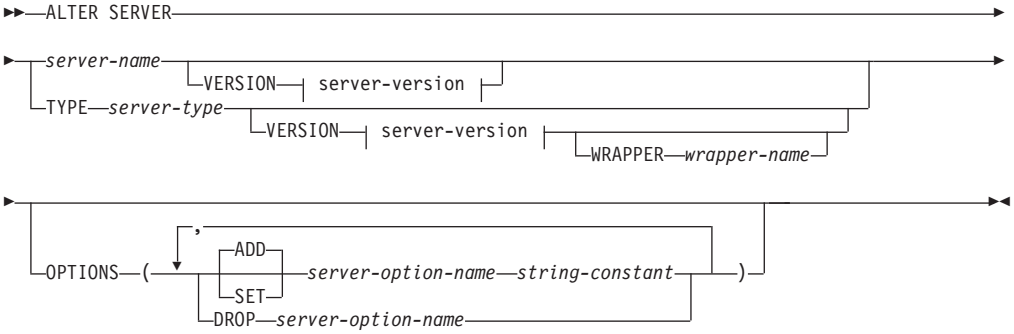
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

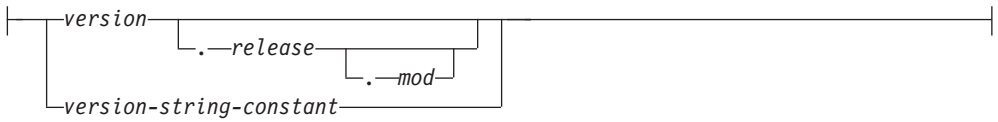
권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

구문



server-version:



설명

server-name

요청되고 있는 변경사항이 적용될 데이터 소스에 대한 페더레이티드 서버의 이름을 식별합니다. 데이터 소스는 카탈로그에 기술되어 있어야 합니다.

VERSION

server-name 뒤에 오는 VERSION 및 매개변수는 *server-name*이 설명하는 데이터 소스의 새 버전을 지정합니다.

version

버전 번호를 지정합니다. 이 값은 정수여야 합니다.

release

*version*으로 표시되는 버전의 릴리스 번호를 지정합니다. 이 값은 정수여야 합니다.

mod

*release*로 표시되는 릴리스의 수정 번호를 지정합니다. 이 값은 정수여야 합니다.

version-string-constant

버전의 전체 명칭을 지정합니다. *version-string-constant*는 단일 값(예: '8i')이거나 *version*, *release* 및 *mod*(적용 가능한 경우)의 조인된 값(예: '8.0.3')일 수 있습니다.

TYPE *server-type*

요청되고 있는 변경사항이 적용될 데이터 소스의 유형을 식별합니다.

VERSION

server-type 뒤에 오는 VERSION 및 매개변수는 어느 서버 옵션에 대한 데이터 소스의 버전이 활성화, 재설정 또는 삭제될 것인지를 지정합니다.

WRAPPER *wrapper-name*

server-type 및 *server-version*에 의해 표시된 유형 및 버전의 데이터 소스와 상호작용하기 위해 페더레이티드 서버가 사용하는 랩퍼의 이름을 지정합니다. 랩퍼는 카탈로그에 나열되어야 합니다.

OPTIONS

*server-name*으로 표시되는 데이터 소스나, *server-type* 및 연관 매개변수로 표시되는 데이터 소스의 범주에 대해 활성화, 재설정 또는 삭제할 서버 옵션을 나타냅니다.

ADD

서버 옵션을 활성화합니다.

SET

서버 옵션의 설정값을 변경합니다.

server-option-name

활성화되거나 재설정될 서버 옵션의 이름을 지정합니다.

string-constant

*server-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

DROP *server-option-name*
서버 옵션을 삭제합니다.

주

- 서버 옵션은 동일한 ALTER SERVER문에서 두 번 이상 지정할 수 없습니다 (SQLSTATE 42853). 서버 옵션이 활성화, 재설정 또는 삭제될 때, 사용 중인 다른 서버 옵션은 영향을 받지 않습니다.
- 다음과 같은 상황에서는 주어진 작업 단위(UOW) 내에서 ALTER SERVER문을 처리할 수 없습니다(SQLSTATE 55007).
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 내에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문
  - 명령문은 데이터 소스의 범주(예: 특정 유형 및 버전의 모든 데이터 소스)를 참조하며, UOW는 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 중 하나에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문
- 서버 옵션이 데이터 소스의 한 유형에 대해 어느 한 값으로 설정되고, 이 유형의 한 인스턴스에 대해 또 다른 값으로 설정되면, 두 번째 값이 인스턴스의 첫 번째 값을 겹쳐씁니다. 예를 들어, PLAN_HINTS가 서버 유형 ORACLE에 대해 'Y'로 설정되고 Oracle 데이터 소스 DELPHI에 대해 'N'으로 설정되었다고 가정해 보십시오. 이 구성은 DELPHI를 제외한 모든 Oracle 데이터 소스에서 플랜 힌트가 사용되도록 합니다.
- 이전 서버 옵션 추가 변경 조작에 의해 사용 가능하게 된 데이터 소스 범주에 대해서만 서버 옵션 설정 또는 서버 옵션 삭제를 변경할 수 있습니다(SQLSTATE 42704).
- 서버 버전을 변경하면 DB2는 지정된 서버 버전이 리모트 서버 버전과 일치한다는 것을 확인하지 않습니다. 부정확한 서버 버전을 지정하면 사용자가 DB2 서버 정의에 속한 별칭에 액세스할 때 SQL 오류가 나타날 수 있습니다. 이는 리모트 서버 버전보다 상위 서버 버전을 지정할 경우 나타날 확률이 높습니다. 이 경우 서버 정의에 속한 별칭에 액세스할 때 DB2는 리모트 서버가 인식하지 못하는 SQL을 보낼 수 있습니다.

예:

예 1: 권한 부여 ID가 Oracle 8.0.3 데이터 소스로 전송될 때, ID의 대소문자가 변경되지 않게 하십시오. 또한 로컬 페더레이티드 서버 CPU가 데이터 소스 CPU보다 두 배 빠르다고 간주하십시오. 옵티마이저에게 이 상태를 알리십시오.

```
ALTER SERVER
  TYPE ORACLE
  VERSION 8.0.3
  OPTIONS
     (ADD FOLD_ID 'N',
     SET CPU_RATIO '2.0')
```

예 2: DCTM_SVR_ASIA라는 Documentum 데이터 소스가 버전 4로 변경되었음을 표시하십시오.

```
ALTER SERVER DCTM_SVR_ASIA
  VERSION 4
```

ALTER SERVICE CLASS

ALTER SERVICE CLASS문은 서비스 클래스의 정의를 변경합니다.

호출

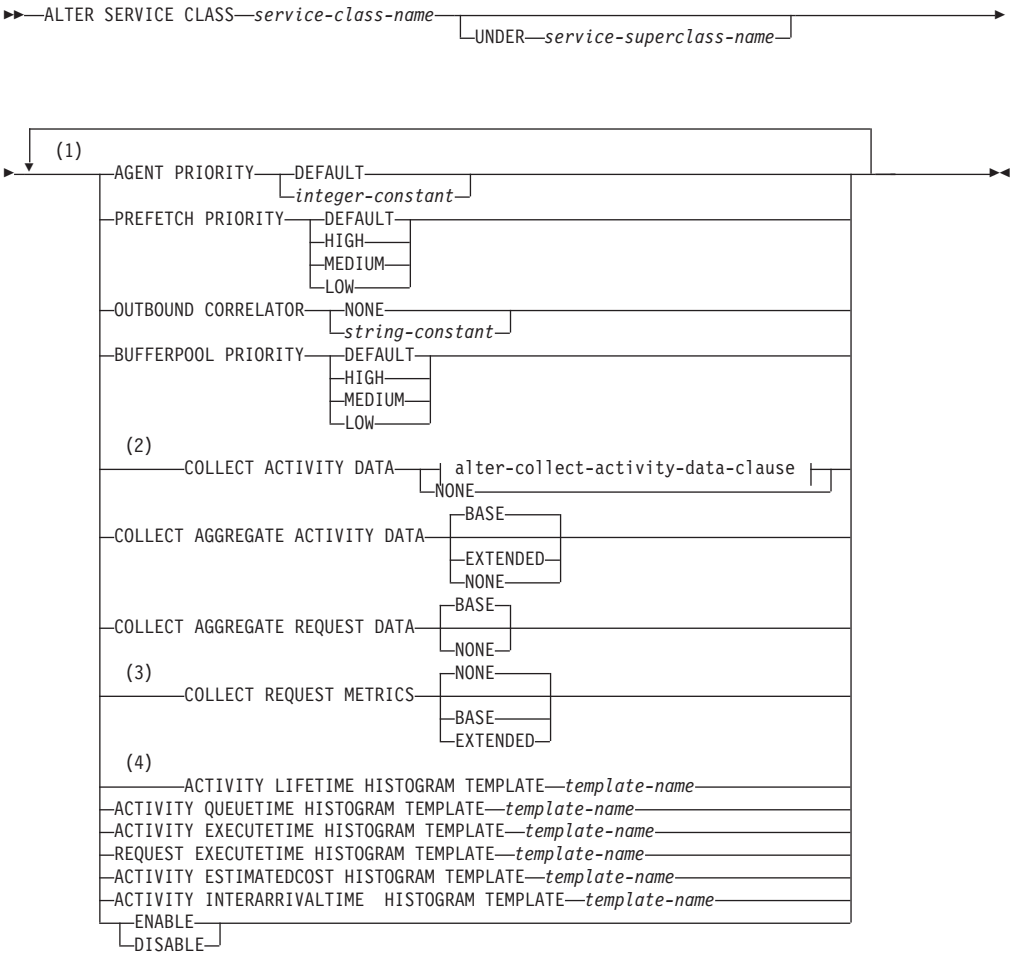
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

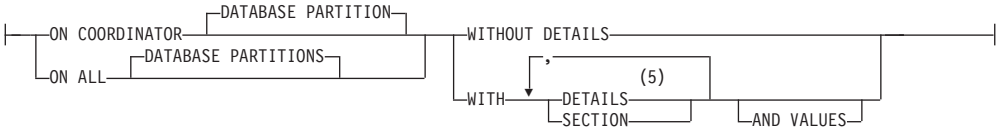
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- SQLADM 권한(모든 변경 절이 COLLECT 절인 경우에만)
- WLMADM 권한
- DBADM 권한

구문



alter-collect-activity-data-clause:



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

2 COLLECT REQUEST METRICS를 제외하고 모든 COLLECT 절은 서비스 서브클래스에 대해서만 유효합니다.

3 COLLECT REQUEST METRICS 절은 서비스 수퍼 클래스에 대해서만 유효합니다.

4 HISTOGRAM TEMPLATE 절은 서비스 서브클래스에 대해서만 유효합니다.

5 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

service-class-name

변경될 서비스 클래스를 식별합니다. 이는 한 부분으로 된 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *service-class-name*은 데이터베이스에 존재하는 서비스 클래스를 식별해야 합니다(SQLSTATE 42704). 서비스 서브클래스를 변경하려면 UNDER 절을 사용하여 *service-superclass-name*을 지정해야 합니다.

UNDER *service-superclass-name*

이 절은 서비스 서브클래스를 변경할 경우에만 사용됩니다. *service-superclass-name*은 서비스 서브클래스의 서비스 수퍼 클래스를 식별하므로 데이터베이스에 존재하는 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704).

AGENT PRIORITY DEFAULT 또는 AGENT PRIORITY *integer-constant*

서비스 클래스에서 실행 중인 에이전트의 상대적(델타) 운영 체제 우선순위나 DB2에서 실행 중인 스레드의 일반 우선순위를 지정합니다. 디폴트값은 DEFAULT입니다. DEFAULT로 설정한 경우, 어떤 특수 조치도 수행되지 않고 서비스 클래스의 에이전트는 운영 체제가 모든 DB2 스레드를 스케줄링하는 일반 우선순위에 따라 스케줄됩니다. 이 매개변수가 DEFAULT가 아닌 다른 값으로 설정되는 경우, 에이전트는 일반 우선순위에 다음 활동 시작 시 AGENT PRIORITY를 더한 우선순위로 설정됩니다. 예를 들어, 일반 우선순위가 20이고 AGENT PRIORITY가 -10으로 설정되면, 서비스 클래스에서 에이전트의 우선순위가 20 – 10 = 10으로 설정됩니다.

UNIX 운영 체제와 Linux에서, 유효한 값은 DEFAULT와 -20 ~ 20입니다(SQLSTATE 42615). 음수 값은 상대적으로 더 높은 우선순위를 나타냅니다. 양수 값은 상대적으로 낮은 우선순위를 나타냅니다.

Windows 운영 체제에서, 유효한 값은 DEFAULT와 -6 ~ 6입니다(SQLSTATE 42615). 음수 값은 상대적으로 더 낮은 우선순위를 나타냅니다. 양수 값은 상대적으로 더 높은 우선순위를 나타냅니다.

AGENT PRIORITY가 서비스 서브클래스에 대해 DEFAULT이면, 상위 서브클래스의 AGENT PRIORITY 값을 상속합니다. AGENT PRIORITY는 디폴트 서브클래스의 경우 변경할 수 없습니다. OUTBOUND CORRELATOR가 설정된 경우 AGENT PRIORITY는 DEFAULT로 설정해야 합니다(SQLSTATE 42613).

주: AIX®에서, AGENT PRIORITY를 사용하는 서비스 클래스에서 에이전트에 대해 상대적으로 더 높은 우선순위를 설정하려면 인스턴스 소유자가 CAP_NUMA_ATTACH 및 CAP_PROPAGATE 성능을 가지고 있어야 합니다. 이 성능을 부여하려면 루트로 로그온하고 다음 명령을 실행하십시오.

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

PREFETCH PRIORITY DEFAULT | HIGH | MEDIUM | LOW

이 매개변수는 서비스 클래스의 에이전트가 프리페치 요청을 제출할 수 있는 우선순위를 제어합니다. 올바른 값은 HIGH, MEDIUM, LOW 또는 DEFAULT입니다(SQLSTATE 42615). HIGH, MEDIUM 및 LOW는 프리페치 요청이 높은, 중간 및 낮은 우선순위 큐에 제출됨을 의미합니다. 프리페치는 높음에서 낮음 순서로 우선순위 큐를 비웁니다. 서비스 클래스의 에이전트는 다음 활동이 시작할 때 PREFETCH PRIORITY 레벨에서 해당되는 프리페치 요청을 제출합니다. 프리페치 요청이 제출된 후 PREFETCH PRIORITY가 변경되어도 요청 우선순위는 변경되지 않습니다. 디폴트 값은 DEFAULT입니다. 이는 서비스 수퍼 클래스의 경우 내부적으로 MEDIUM에 맵핑됩니다. 서비스 서브클래스에 대해 DEFAULT가 지정되면, 해당되는 상위 수퍼 클래스의 PREFETCH PRIORITY를 상속합니다.

PREFETCH PRIORITY는 디폴트 서브클래스의 경우 변경할 수 없습니다(SQLSTATE 5U032).

OUTBOUND CORRELATOR NONE 또는 OUTBOUND CORRELATOR *string-constant*

서비스 클래스의 스레드를 외부 워크로드 관리 프로그램 서비스 클래스에 연관시킬 것인지 여부를 지정합니다.

OUTBOUND CORRELATOR가 서비스 수퍼 클래스에 대해 *string-constant*로 설정되고 OUTBOUND CORRELATOR NONE이 서비스 서브클래스에 대해 설정된 경우, 서비스 서브클래스는 상위의 OUTBOUND CORRELATOR를 상속합니다. AGENT PRIORITY가 DEFAULT로 설정되지 않으면 OUTBOUND CORRELATOR는 NONE으로 설정해야 합니다(SQLSTATE 42613).

OUTBOUND CORRELATOR NONE
서비스 수퍼 클래스의 경우 이 서비스 클래스와의 외부 워크로드 관리 프로그램 서비스 클래스 연관이 없음을 지정하고, 서비스 서브클래스의 경우 외부 워크로드 관리 프로그램 서비스 클래스 연관이 상위와 동일함을 지정합니다.

OUTBOUND CORRELATOR *string-constant*
서비스 클래스의 스레드를 외부 워크로드 관리 프로그램 서비스 클래스에 연관시키기 위한 상관자로 사용할 *string-constant*를 지정합니다. 외부 워크로드 관리 프로그램은 활성 상태여야 합니다(SQLSTATE 5U030). 외부 워크로드 관리 프로그램은 *string-constant* 값을 인식하도록 설정해야 합니다.

BUFFERPOOL PRIORITY DEFAULT | HIGH | MEDIUM | LOW
이 매개변수는 서비스 클래스에서 활동에 의해 페치된 페이지의 버퍼 풀 우선순위를 제어합니다. 유효한 값은 HIGH, MEDIUM, LOW 또는 DEFAULT입니다(SQLSTATE 42615). 버퍼 풀 우선순위가 높은 서비스 클래스에서 활동에 의해 페치된 페이지 수는 버퍼 풀 우선순위가 낮은 서비스 클래스에서 활동에 의해 페치된 페이지 수보다 적게 스왑 아웃됩니다. 서비스 서브클래스에 대해 DEFAULT가 지정되면 해당되는 상위 수퍼 클래스의 BUFFERPOOL PRIORITY를 상속합니다.

BUFFERPOOL PRIORITY는 디폴트 서브클래스의 경우 변경할 수 없습니다(SQLSTATE 5U032).

COLLECT ACTIVITY DATA
해당 서비스 클래스에서 실행되는 각 활동에 대한 데이터가 활동 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. COLLECT ACTIVITY DATA 절은 서비스 서브클래스에 대해서만 유효합니다.

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION
활동 데이터가 활동 코디네이터의 데이터베이스 파티션에서만 수집되도록 지정합니다.

ON ALL DATABASE PARTITIONS
활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. SECTION이 지정된 경우, 활동 세부사항 및 섹션 환경은 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우, 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다.

WITHOUT DETAILS
서비스 클래스에서 실행되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내져야 함을 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 해당 활동에 대한 활성 활동 이벤트 모니터로 전송되도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

NONE
해당 서비스 클래스에서 실행되는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA
해당 서비스 클래스에 대한 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. 디폴트는 COLLECT AGGREGATE ACTIVITY DATA BASE입니다. COLLECT AGGREGATE ACTIVITY DATA 절은 서비스 서브클래스에 대해서만 유효합니다.

BASE
해당 서비스 클래스에 대한 기본 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 실행 시간 막대 그래프

EXTENDED
해당 서비스 클래스에 대한 모든 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 데이터 처리 언어(DML) 계산 비용 막대 그래프
- 활동 DML 도착 간 시간 막대 그래프

NONE
어떤 집계 활동 데이터도 해당 서비스 클래스에 대해 수집되지 않음을 지정합니다.

COLLECT AGGREGATE REQUEST DATA
해당 서비스 클래스에 대한 집계 요청 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. 디폴트는 COLLECT AGGREGATE REQUEST DATA NONE입니다. COLLECT AGGREGATE REQUEST DATA 절은 서비스 서브클래스에 대해서만 유효합니다.

BASE
해당 서비스 클래스에 대한 기본 집계 요청 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다.

NONE
어떤 집계 요청 데이터도 해당 서비스 클래스에 대해 수집되지 않음을 지정합니다.

COLLECT REQUEST METRICS
지정된 서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 모니터 메트릭을 수집해야 함을 지정합니다. 디폴트는 COLLECT REQUEST METRICS NONE입니다. COLLECT REQUEST METRICS절은 서비스 수퍼 클래스에 대해서만 유효합니다(SQLSTATE 50U44).

주: 유효한 요청 메트릭 콜렉션 설정은 요청을 제출하는 워크로드에서 COLLECT REQUEST METRICS절로 지정된 속성 및 **mon_req_metrics** 데이터베이스 구성 매개변수의 조합입니다. 워크로드 속성 또는 구성 매개변수에 NONE이 아닌 다른 값이 있는 경우, 메트릭이 요청에 대해 수집됩니다.

BASE
서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 기본 메트릭을 수집할 것을 지정합니다.

NONE
서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 어떤 메트릭도 수집하지 않을 것을 지정합니다.

EXTENDED
서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 확장 메트릭을 수집할 것을 지정합니다.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*
특정 간격 동안 서비스 클래스에서 실행 중인 DB2 활동의 지속기간(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 이 절은 서비스 서브클래스에 대해서만 유효합니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*
서비스 클래스에서 실행 중인 DB2 활동이 특정 간격 동안 큐에서 대기하는 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 이 절은 서비스 서브클래스에 대해서만 유효합니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*
서비스 클래스에서 실행 중인 DB2 활동이 특정 간격 동안 실행 중인 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 코디네이터 데이터베이스 파티션에서만 막대 그래프에서 수집됩니다. 시간에는 유휴 시간이 포함되지 않습니다. 유휴 시간은 어떤 작업도 완료되지 않은 경우에 동일한 활동에 속하는 요청의 실행 사이 시간입니다. 유휴 시간의 예로, 커서 열기가 종료되고 그 커서로부터 페치가 시작되는 시간 사이의 시간을 들 수 있습니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 이 절은 서비스 서브클래스에 대해서만 유효합니다.

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *template-name*
서비스 클래스에서 실행 중인 DB2 요청이 특정 간격 동안 실행 중인 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 요청 실행 시간은 요청이 실행되는 데이터베이스 파티션마다 이 막대 그래프에서 수집됩니다. 이 정보는 BASE 옵션을 사용하여 COLLECT AGGREGATE REQUEST DATA 절을 지정한 경우에만 수집됩니다. 이 절은 서비스 서브클래스에 대해서만 유효합니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*
서비스 클래스에서 실행 중인 DML 활동의 계산된 비용(timeron 단위)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 이 절은 서비스 서브클래스에 대해서만 유효합니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*
하나의 DML 활동이 도착하고 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 이 절은 서비스 서브클래스에 대해서만 유효합니다.

ENABLE 또는 DISABLE
연결 또는 활동을 서비스 클래스에 맵핑할 수 있는지 여부를 지정합니다.

ENABLE
연결 또는 활동을 서비스 클래스에 맵핑할 수 있습니다.

DISABLE
연결 또는 활동을 서비스 클래스에 맵핑할 수 없습니다. 사용 불가능한 서비스 클래스에 맵핑되는 연결 또는 활동은 거부됩니다(SQLSTATE 5U028). 서비스 수퍼 클래스가 사용 불가능한 경우 해당 서비스 서브클래스도 사용 불가능합니다. 서비스 수퍼 클래스가 다시 사용 가능하게 되면 해당되는 서비스 서브클래스는 시스템 카탈로그에 정의된 상태로 리턴됩니다. 디폴트 서비스 클래스는 사용 불가능하도록 설정할 수 없습니다(SQLSTATE 5U032).

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 모든 파티션 사이에서 한 번에 하나의 언커미트된 WLM 독점 SQL문만 허용됩니다. 언커미트된 WLM 독점 SQL문이 실행 중인 경우, 연속 WLM 독점 SQL문은 현재 WLM 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 발행하는 연결에 대해서도 COMMIT문이 실행될 때까지 적용되지 않습니다.
- ALTER SERVICE CLASS문이 커미트되면, AGENT PRIORITY, PREFETCH PRIORITY, OUTBOUND CORRELATOR 및 COLLECT에 대한 변경사항은 서비스 클래스에서 다음 새 활동에 적용됩니다. 서비스 클래스의 기존 활동은 계속 이전 설정을 사용하여 작업을 완료합니다.

예:

예 1: 서비스 수퍼 클래스 PETSALES에서 에이전트의 에이전트 우선순위를 DEFAULT에서 가능한 최상위 값(UNIX 및 Linux 운영 체제에 대해 표시된 값. Windows 운영 체제에서는 6을 대신 사용)으로 높이도록 변경하십시오.

```
ALTER SERVICE CLASS PETSALES AGENT PRIORITY -20
```

예 2: 서비스 수퍼 클래스 BARNSALES를 변경하고 아웃바운드 상관자 'osLowPriority'를 추가하십시오. 서비스 수퍼 클래스 및 해당되는 서비스 서브클래스에서 실행 중인 스레드에는 아웃바운드 상관자 'osLowPriority'가 연관됩니다.

```
ALTER SERVICE CLASS BARNSALES OUTBOUND CORRELATOR 'osLowPriority'
```

ALTER TABLE

ALTER TABLE문은 테이블의 정의를 변경합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509) 에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 변경될 테이블에 대한 ALTER 특권
- 변경될 테이블에 대한 CONTROL 특권
- 테이블의 스키마에 대한 ALTERIN 특권
- DBADM 권한

외부 키를 작성하거나 삭제하려면, 명령문의 권한 부여 ID가 상위 테이블에 대해 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블에 대해 REFERENCES 특권
- 지정된 상위 키의 각 컬럼에 대한 REFERENCES 특권
- 테이블에 대한 CONTROL 특권
- DBADM 권한

테이블 T의 고유 제한조건 또는 기본 키를 삭제하려면, 명령문의 권한 부여 ID가 갖고 있는 특권이 T의 상위 키에 종속된 모든 테이블에 대해 적어도 다음 중 하나가 포함되어야 합니다.

- 테이블에 대한 ALTER 특권
- 테이블에 대한 CONTROL 특권
- 테이블의 스키마에 대한 ALTERIN 특권
- DBADM 권한

구체화된 쿼리 테이블이 되도록 테이블을 변경하려면(fullselect 사용), 명령문의 권한 부여 ID가 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블에 대한 CONTROL 특권
- DBADM 권한

그리고 fullselect에서 식별되는 각 테이블이나 뷰에 대해 적어도 다음 중 하나(그룹 특권 제외):

- 테이블 또는 뷰에 대한 SELECT 특권 및 ALTER 특권(그룹 특권 포함)

- 테이블 또는 뷰에 대한 CONTROL 특권
- 테이블 또는 뷰에 대한 SELECT 특권 및 테이블 또는 뷰 스키마에 대한 ALTERIN 특권(그룹 특권 포함)
- DATAACCESS 권한

더 이상 구체화된 쿼리 테이블이 되지 않도록 테이블을 변경하려면, 명령문의 권한 부여 ID가 구체화된 쿼리 테이블을 정의하기 위해 사용되는 fullselect에서 식별되는 각 테이블이나 뷰에 대해 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블 또는 뷰에 대한 ALTER 특권
- 테이블 또는 뷰에 대한 CONTROL 특권
- 테이블 또는 뷰 스키마에 대한 ALTERIN 특권
- DBADM 권한

DB2SECURITYLABEL 유형 컬럼을 테이블에 추가하려면, 명령문의 권한 부여 ID가 보유한 특권이 테이블과 연결된 보안 규정에서 최소한 하나의 보안 레이블을 포함해야 합니다.

테이블에서 보안 규정을 제거하려면 명령문의 권한 부여 ID가 보유한 특권은 SECADM 권한을 포함해야 합니다.

테이블을 변경하여 데이터 파티션을 접속하려면, 명령문의 권한 부여 ID가 보유한 특권은 소스 테이블의 다음 권한 중 최소 하나를 포함해야 합니다.

- 테이블의 SELECT 특권 및 테이블 스키마에 대한 DROPIN 특권
- 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

그리고 최소한 목표 테이블의 다음 권한 중 하나를 포함해야 합니다.

- 테이블에 대한 ALTER 및 INSERT 특권
- 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

테이블을 변경하여 데이터 파티션을 접속 해제하려면, 명령문의 권한 부여 ID가 보유한 특권은 접속 해제된 파티션의 목표 테이블의 다음 권한 중 최소 하나도 포함해야 합니다.

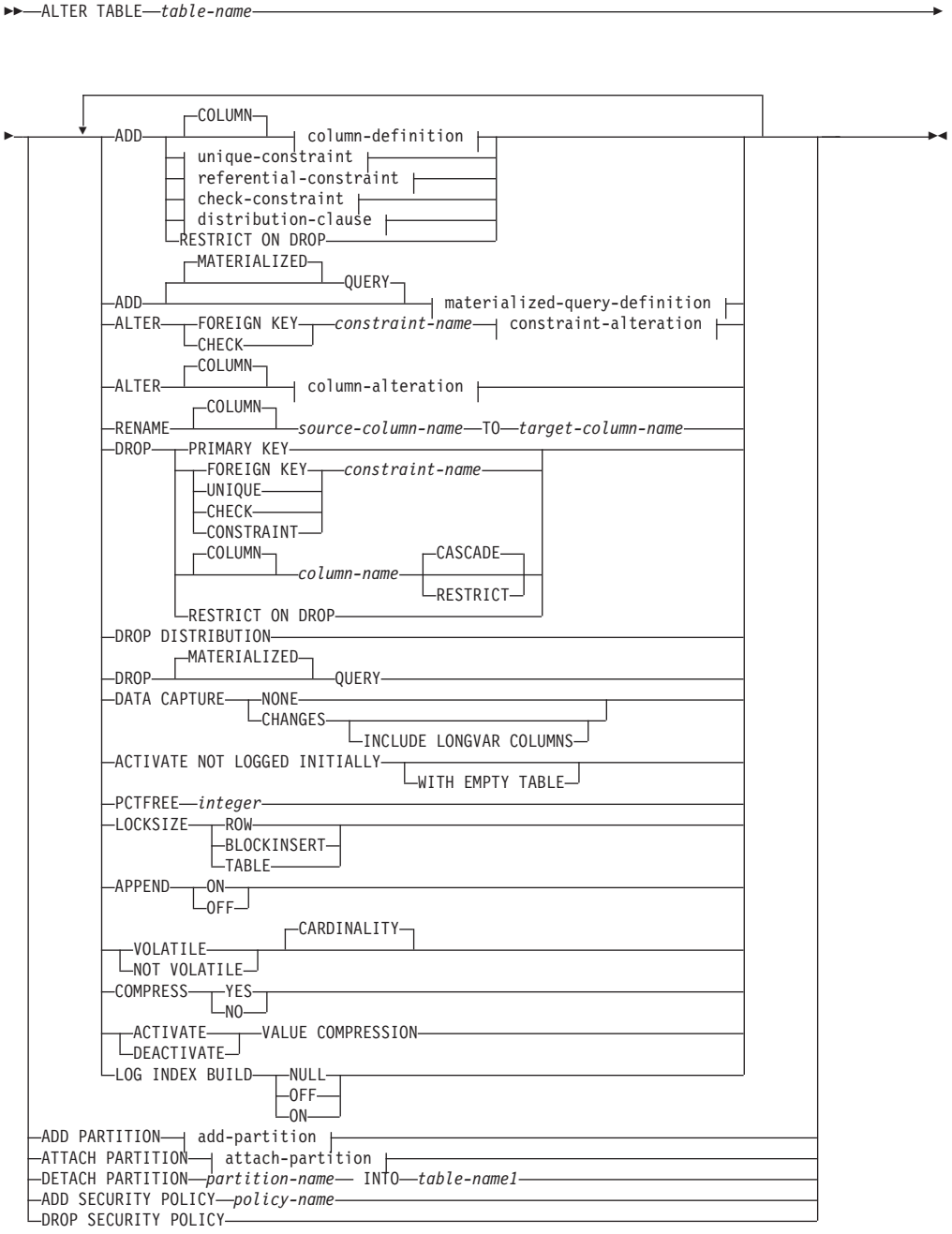
- 데이터베이스의 CREATETAB 권한 및 테이블이 사용하는 테이블 스페이스의 USE 특권 및 다음 중 하나를 포함해야 합니다.
  - 새 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않는 경우 데이터베이스의 IMPLICIT_SCHEMA 권한
  - 새 테이블의 스키마 이름이 기존의 스키마를 참조하는 경우 스키마에 대한 CREATEIN 특권

- DBADM 권한

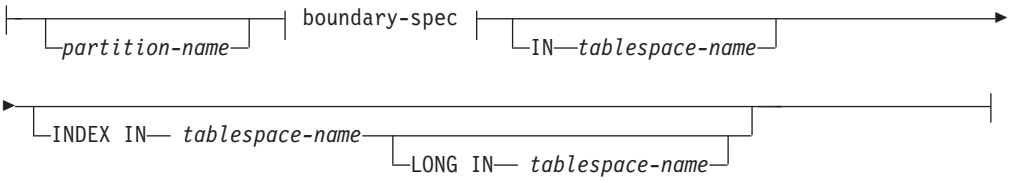
그리고 최소한 소스 테이블의 다음 권한 중 하나를 포함해야 합니다.

- 테이블의 SELECT, ALTER 및 DELETE 특권
- 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

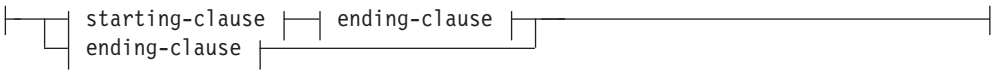
구문



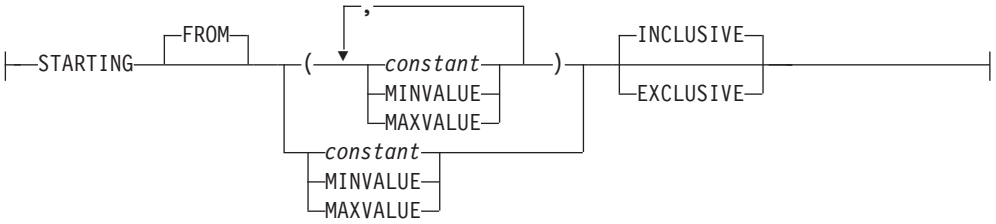
add-partition:



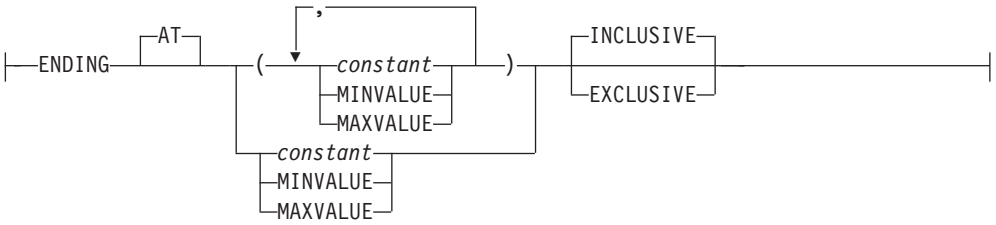
boundary-spec:



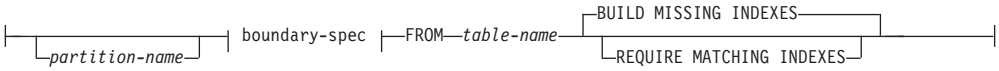
starting-clause:



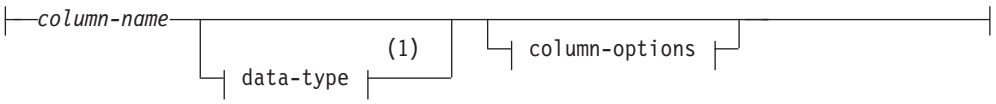
ending-clause:



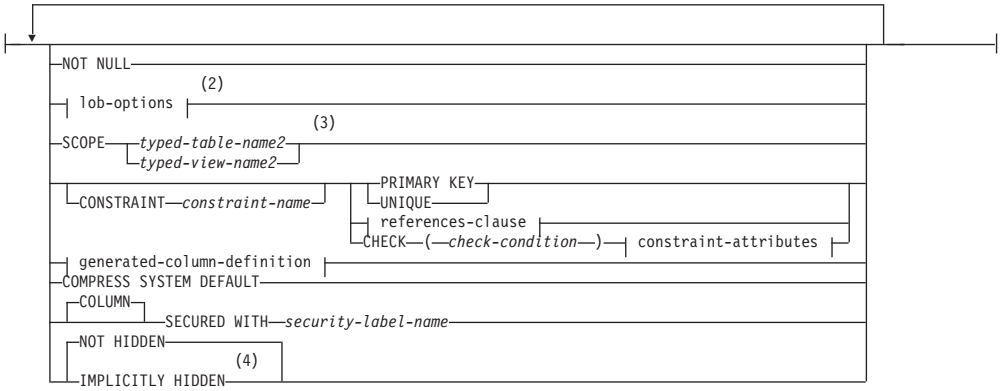
attach-partition:



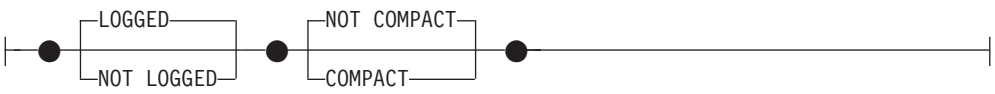
column-definition:



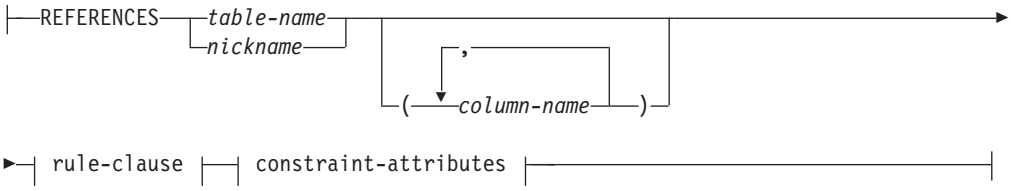
column-options:



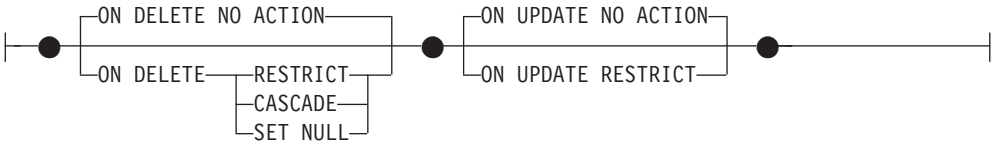
lob-options:



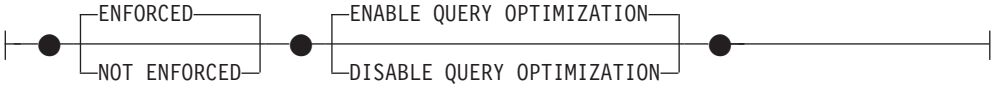
references-clause:



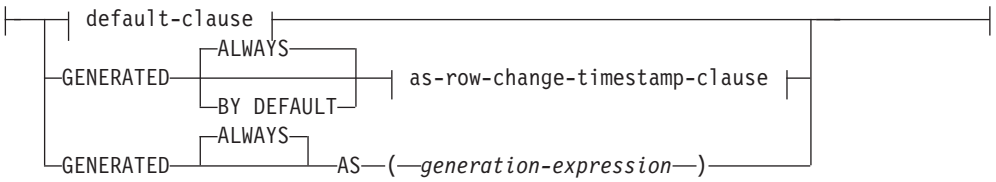
rule-clause:



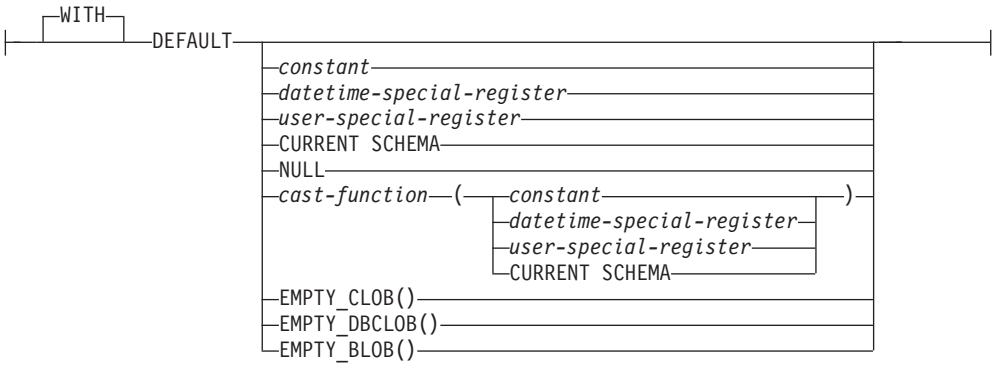
constraint-attributes:



generated-column-definition:



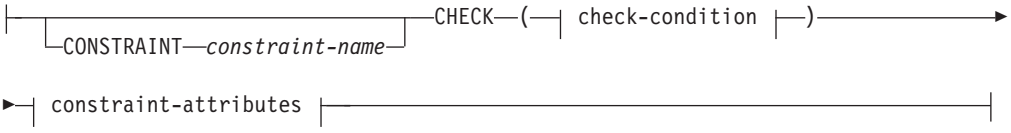
default-clause:



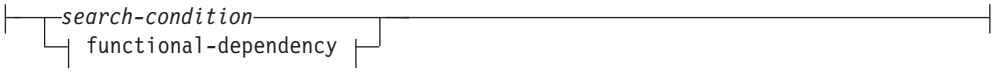
unique-constraint:

referential-constraint:

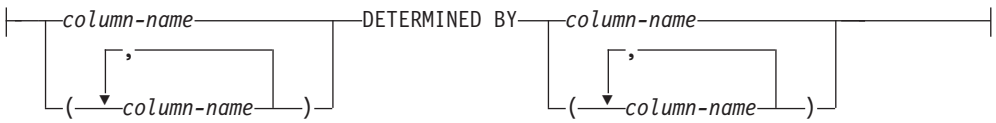
check-constraint:



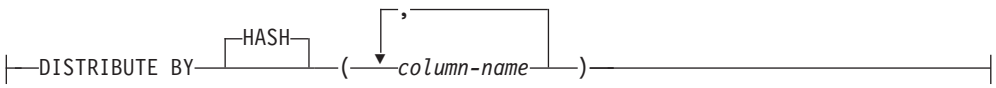
check-condition:



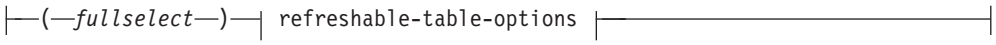
functional-dependency:



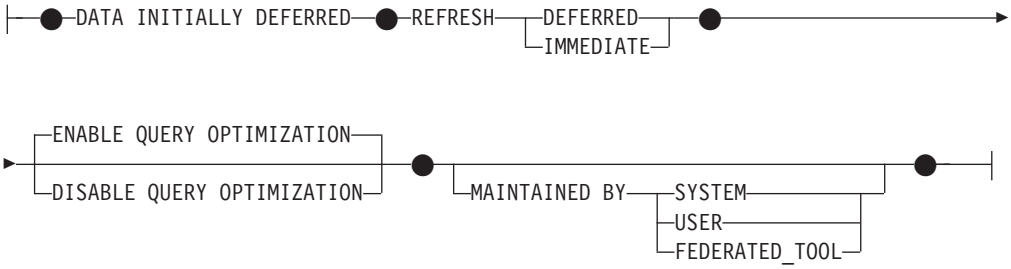
distribution-clause:



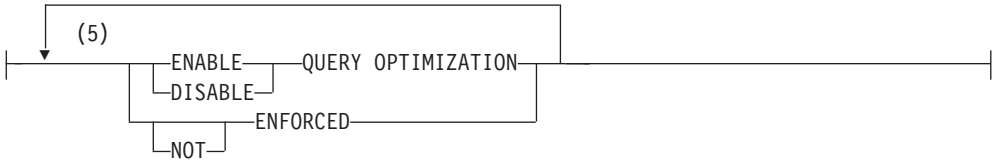
materialized-query-definition:



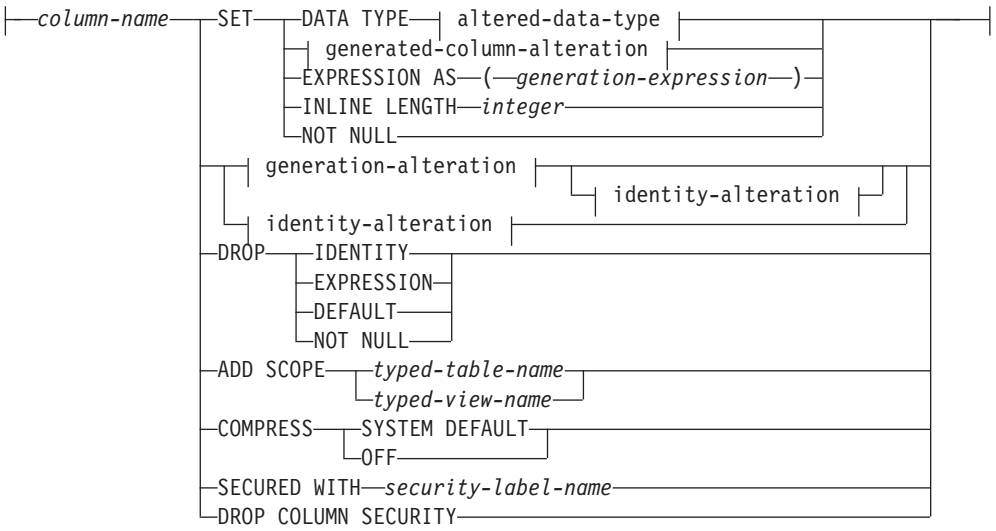
refreshable-table-options:



constraint-alteration:

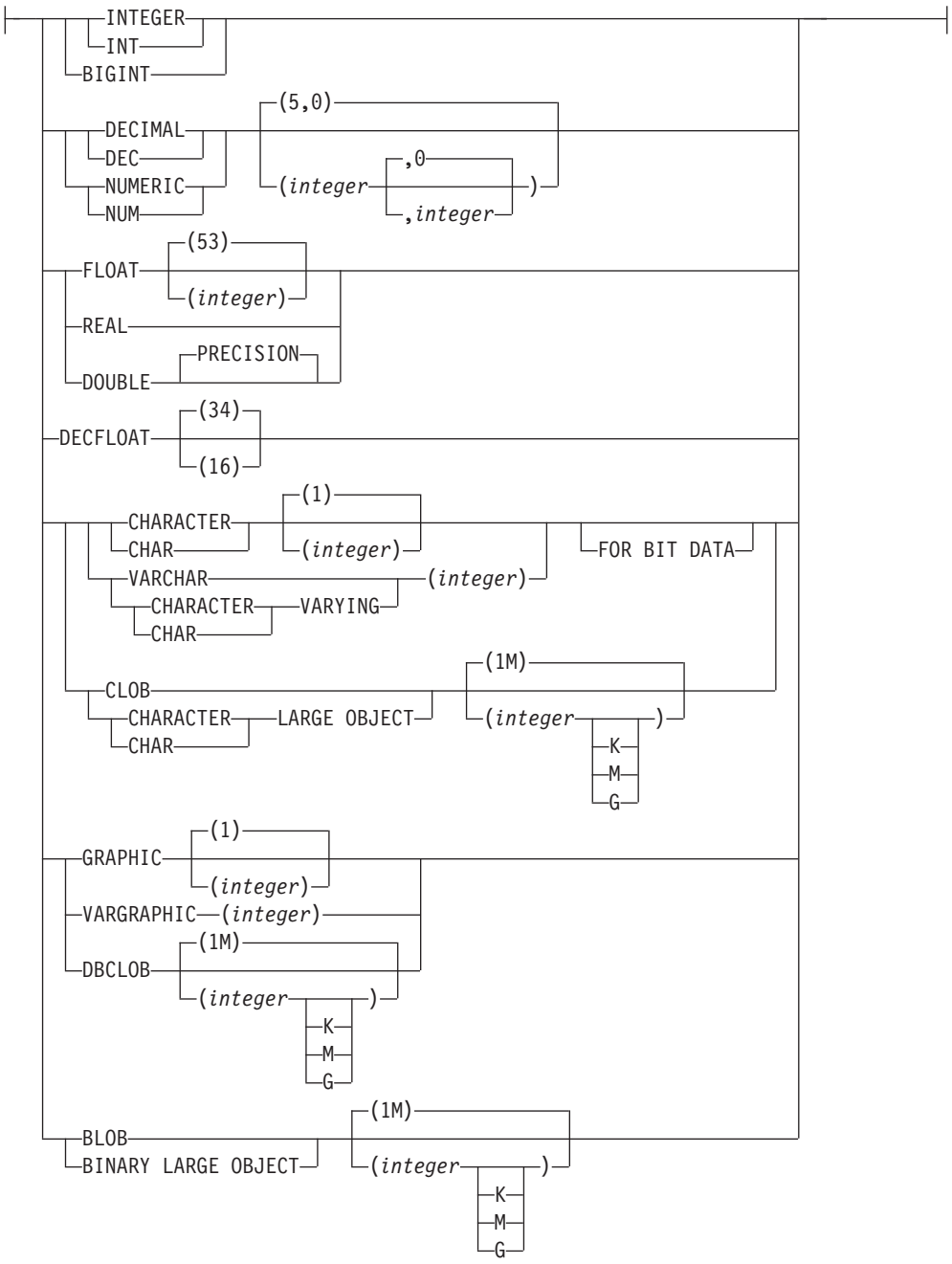


column-alteration:

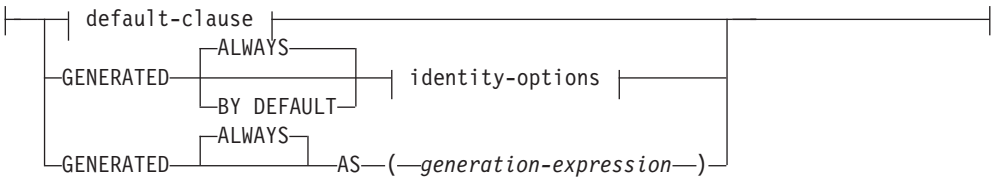


altered-data-type:

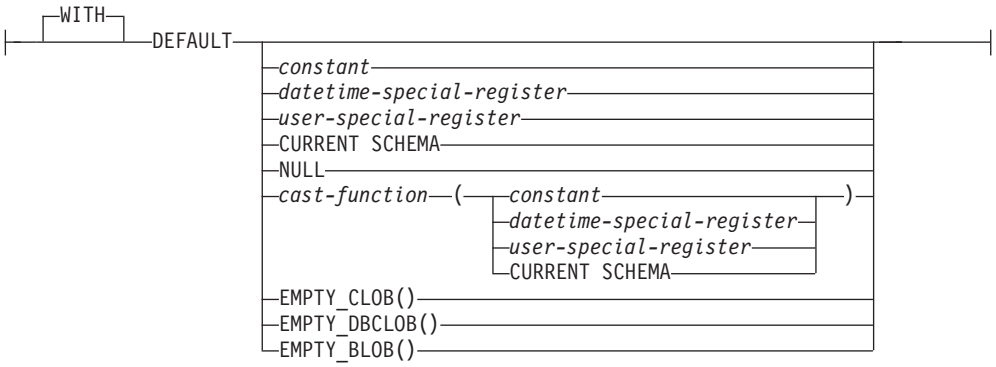
내장 유형:



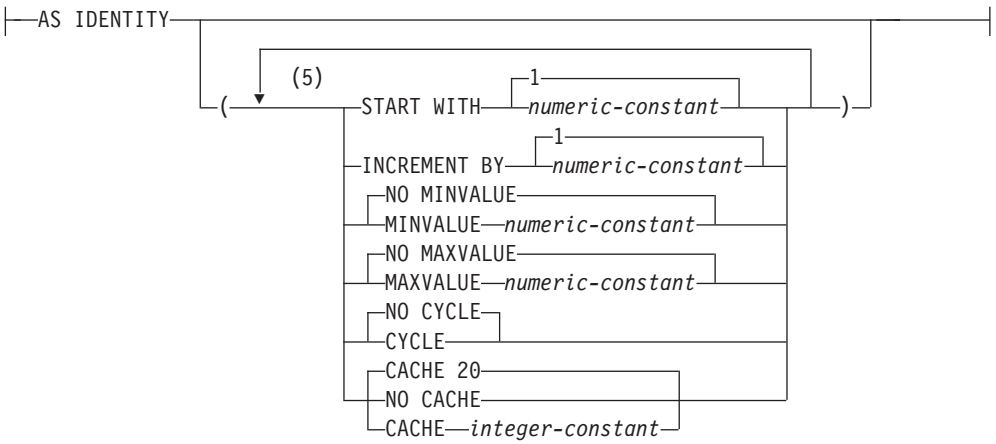
generated-column-alteration:



default-clause:



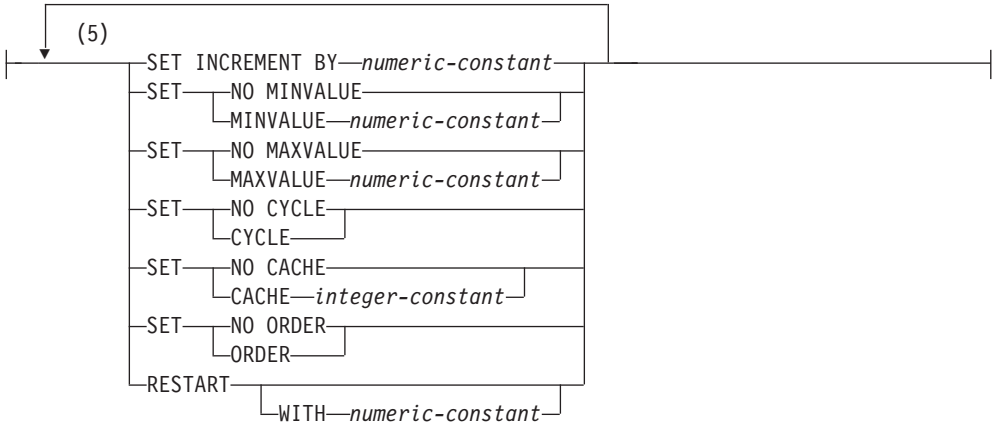
identity-options:



as-row-change-timestamp-clause:

generation-alteration:

identity-alteration:



주:

1 선택된 첫 번째 컬럼 옵션이 *generated-column-definition*인 경우 *data-type*은 생략될 수 있고, 생성 표현식에 의해 계산됩니다.

2 *lob-options*절은 대형 오브젝트 유형(BLOB, DBCLOB 및 BLOB)과 대형 오브젝트 유형을 기반으로 하는 구별 유형에만 적용됩니다.

3 SCOPE절은 REF 유형에만 적용됩니다.

4 IMPLICITLY HIDDEN은 ROW CHANGE TIMESTAMP도 지정된 경우에만 지정할 수 있습니다.

5 같은 절은 두 번 이상 지정될 수 없습니다.

6 지정된 첫 번째 컬럼 옵션이 생성된 컬럼 정의인 경우, 데이터 유형은 행 변경 시간소인 컬럼에 대해 선택적이며 데이터 유형 디폴트는 TIMESTAMP(6)입니다.

설명

table-name

*table-name*은 현재 서버에 있는 테이블을 식별해야 합니다. 이 이름은 별칭이 될 수 없으며(SQLSTATE 42809), 뷰, 카탈로그 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블이 될 수 없습니다(SQLSTATE 42995).

*table-name*이 구체화된 쿼리 테이블을 식별할 경우, 구체화된 쿼리 테이블 추가 또는 삭제, 초기에 로그되지 않은 테이블 활성화, RESTRICT ON DROP 추가 또는 삭제, PCTFREE, 잠금 크기, 추가 또는 VOLATILE 변경으로 변경 작업이 제한됩니다.

*table-name*이 범위 클러스터 테이블을 식별하는 경우, 제한조건 추가, 변경 또는 삭제, 초기에 로그되지 않은 테이블 활성화, RESTRICT ON DROP 추가 또는 삭제, 잠금 크기, 데이터 캡처 또는 VOLATILE 변경 및 컬럼 디폴트값 설정으로 변경 작업이 제한됩니다.

ADD PARTITION *add-partition*

하나 이상의 데이터 파티션을 파티션된 테이블에 추가합니다. 지정된 테이블이 파티션된 테이블이 아니면 오류가 리턴됩니다(SQLSTATE 428FT). 데이터 파티션 번호는 32 767을 초과하지 않아야 합니다.

partition-name

데이터 파티션의 이름을 지정합니다. 테이블의 다른 데이터 파티션 이름과 달라야 합니다(SQLSTATE 42710). 이 절이 지정되지 않는 경우 테이블의 고유 이름을 작성하기 위해 정수 값의 문자 양식이 뒤에 붙는 'PART'가 이름이 됩니다.

boundary-spec

새 데이터 파티션에 대한 값의 범위를 지정합니다. 이 범위는 기존 데이터 파티션의 범위와 겹쳐서는 안됩니다(SQLSTATE 56016). starting-clause 및 ending-clause에 대한 설명은 『CREATE TABLE』을 참조하십시오.

starting-clause가 생략되는 경우 새 데이터 파티션은 테이블의 끝에 있는 것으로 가정됩니다. ending-clause가 생략되는 경우 새 데이터 파티션은 테이블의 시작에 있는 것으로 가정됩니다. 파티션 키의 첫 번째 컬럼이 DESC이면 이 가정들은 거꾸로가 됩니다.

IN *tablespace-name*

데이터 파티션이 저장될 테이블 스페이스를 지정합니다. 이름 지정된 테이블 스페이스는 동일한 페이지 크기여야 하고, 동일한 데이터베이스 파티션 그룹에 속해야 하며, 파티션된 테이블의 다른 테이블 스페이스와 동일한 방법으로 관리되어야 합니다(SQLSTATE 42838). 이 테이블 스페이스는 동일한 테이블의 다른 데이터 파티션에 대해 이미 사용 중인 테이블 스페이스 또는 해당 테이블에서 현재 사용되고 있지 않은 테이블 스페이스일 수 있습니다. 그러나 이 테이블 스페이스는 명령문의 권한 부여 ID에 USE 특권(SQLSTATE 42727)이 있는 테이블 스페이스여야 합니다. 이 절이 지정되지 않으면 테이블의 첫 번째 보이는 데이터 파티션 또는 접속된 데이터 파티션의 테이블 스페이스가 쓰입니다.

INDEX IN *tablespace-name*

데이터 파티션에서 파티션된 인덱스가 저장되는 테이블 스페이스를 지정합니다. INDEX IN절이 지정되지 않은 경우, 데이터 파티션에서 파티션된 인덱스는 데이터 파티션과 동일한 테이블 스페이스에 저장됩니다.

디폴트값이거나 INDEX IN절에서 지정된 새 인덱스 파티션이 사용한 테이블 스페이스는 다른 모든 인덱스 파티션이 사용한 테이블 스페이스의 유형(SMS 또는 DMS), 페이지 크기 및 Extent 크기와 일치해야 합니다(SQLSTATE 42838).

LONG IN *tablespace-name*

긴 컬럼 데이터를 포함하는 데이터 파티션이 저장될 테이블 스페이스를 지정합니다. 이름 지정된 테이블 스페이스는 동일한 페이지 크기여야 하고, 동일한 데

이터베이스 파티션 그룹에 속해야 하며, 파티션된 테이블의 다른 테이블 스페이스 및 데이터 파티션과 동일한 방법으로 관리되어야 합니다(SQLSTATE 42838). 또한 이 테이블 스페이스는 명령문의 권한 부여 ID에 USE 특권이 있는 테이블 스페이스여야 합니다. 이름 지정된 테이블 스페이스에 대한 페이지 크기 및 Extent 크기는 파티션된 테이블의 기타 데이터 파티션의 페이지 크기와 Extent 크기와 다를 수 있습니다.

파티션된 테이블이 있는 LONG IN절의 사용에 관한 규칙에 대해서는 『파티션된 테이블에서의 대형 오브젝트(LOB) 동작』을 참조하십시오.

ATTACH PARTITION *attach-partition*

다른 테이블을 새 데이터 파티션으로 접속시킵니다. 접속 중인 테이블의 데이터 오브젝트는 접속의 상대인 테이블의 새 파티션이 됩니다. 관련된 데이터 이동이 없습니다. 테이블은 무결성 설정 보류 상태에 놓이고, SET INTEGRITY문이 실행될 때까지 참조 무결성 점검이 지연됩니다. ALTER TABLE ATTACH 조작은 IN 또는 LONG IN 절의 사용을 허용하지 않습니다. 해당 데이터 파티션에 대한 LOB의 배치가 소스 테이블이 작성될 때 판별됩니다. 파티션된 테이블이 있는 LONG IN 절의 사용에 관한 규칙에 대해서는 『파티션된 테이블에서의 대형 오브젝트(LOB) 동작』을 참조하십시오.

partition-name

데이터 파티션의 이름을 지정합니다. 테이블의 다른 데이터 파티션 이름과 달라야 합니다(SQLSTATE 42710). 이 절이 지정되지 않는 경우 테이블의 고유 이름을 작성하기 위해 정수 값의 문자 양식이 뒤에 붙는 'PART'가 이름이 됩니다.

boundary-spec

새 데이터 파티션에 대한 값의 범위를 지정합니다. 이 범위는 기존 데이터 파티션의 범위와 겹쳐서는 안됩니다(SQLSTATE 56016). starting-clause 및 ending-clause에 대한 설명은 『CREATE TABLE』을 참조하십시오.

starting-clause가 생략되는 경우 새 데이터 파티션은 테이블의 끝에 있는 것으로 가정됩니다. ending-clause가 생략되는 경우 새 데이터 파티션은 테이블의 시작에 있는 것으로 가정됩니다.

FROM *table-name1*

새 파티션에 대한 데이터 소스로 사용될 테이블을 지정하십시오. *table-name1*의 테이블 정의에는 다중 데이터 파티션이 있을 수 없고, 다음의 항목에 대해 변경된 테이블과 일치해야 합니다(SQLSTATE 428G3).

- 컬럼 수가 동일해야 합니다.
- 테이블의 동일한 순서 위치에 있는 컬럼의 데이터 유형이 동일해야 합니다.
- 테이블의 동일한 순서 위치에 있는 컬럼의 널(NULL) 가능성 등록 정보가 동일해야 합니다.

- 데이터가 또한 분산되어 있으면 동일한 분산 메소드를 사용하여 동일한 데이터베이스 파티션 그룹 전체에 분산되어야 합니다.
- 두 테이블에서 한 테이블의 데이터가 구성된 경우, 구성은 일치해야 합니다.
- 구조화된 경우, XML 또는 LOB 데이터 유형, INLINE LENGTH가 동일해야 합니다.

*table-name1*의 데이터가 성공적으로 접속된 후 DROP TABLE *table-name1*과 동일한 조작이 수행되어 데이터베이스에서 더 이상 데이터가 없는 해당 테이블을 제거합니다.

BUILD MISSING INDEXES

소스 테이블에 목표 테이블에서 파티션된 인덱스에 해당하는 인덱스가 없는 경우에 지정하며, SET INTEGRITY 조작은 기존 데이터 파티션에서 파티션된 인덱스에 해당하는 새 데이터 파티션에서 파티션된 인덱스를 빌드합니다. 목표 테이블에서 파티션된 인덱스와 일치하지 않는 소스 테이블의 인덱스는 접속 처리 중 삭제됩니다.

REQUIRE MATCHING INDEXES

소스 테이블에 목표 테이블에서 파티션된 인덱스와 일치하는 인덱스가 있어야 함을 지정합니다. 그렇지 않으면 오류가 리턴되며(SQLSTATE 428GE), 일치하지 않는 인덱스에 대한 관리 로그에 정보가 작성됩니다.

REQUIRE MATCHING INDEXES 절이 지정되지 않고 소스 테이블의 인덱스가 목표 테이블에서 파티션된 모든 인덱스와 일치하지 않는 경우, 다음 동작이 발생합니다.

1. 소스 테이블과 일치하지 않는 목표 테이블에 있는 인덱스이고, 고유 인덱스이거나 REJECT INVALID VALUES로 정의된 XML 인덱스인 경우, ATTACH 조작이 실패합니다(SQLSTATE 428GE).
2. 소스 테이블과 일치하지 않는 목표 테이블에 있는 다른 모든 인덱스의 경우, 소스 테이블의 인덱스 오브젝트는 접속 조작 중 유효하지 않은 것으로 표시됩니다. 소스 테이블에 인덱스가 없는 경우, 비어 있는 인덱스 오브젝트는 유효하지 않은 것으로 작성 및 표시됩니다. ATTACH 조작이 성공하지만 새 데이터 파티션의 인덱스 오브젝트가 유효하지 않음으로 표시됩니다. 일반적으로 SET INTEGRITY는 데이터 파티션에 대해 실행하기 위한 다음 조작입니다. 필요한 경우 SET INTEGRITY는 최근 접속된 데이터 파티션에서 인덱스 오브젝트의 재빌드를 강제 실행합니다. 인덱스 재빌드는 새 데이터 온라인을 가져오는 데 필요한 시간을 늘릴 수 있습니다.
3. 일치하지 않는 인덱스에 대한 관리 로그에 정보가 작성됩니다.

DETACH PARTITION *partition-name* **INTO** *table-name1*

변경된 테이블에서 데이터 파티션 *partition-name*을 접속 해제하고 데이터 파티션을 사용하여 *table-name1*이라는 이름의 새 테이블을 작성합니다. 데이터 파티션은

논리적으로 데이터 이동 없이 새 테이블에 접속됩니다. 지정된 데이터 파티션은 변경 중인 테이블의 마지막 남은 데이터 파티션이 될 수 없습니다(SQLSTATE 428G2).

ADD SECURITY POLICY *policy-name*

테이블에 보안 규정을 추가합니다. 보안 규정은 현재 서버에 존재해야 합니다(SQLSTATE 42704). 테이블에 이미 보안 규정(SQLSTATE 55065)이 있어서는 안 되며 유형이 지정된 테이블(SQLSTATE 428DH), 구체화된 쿼리 테이블(MQT) 또는 스테이징 테이블(SQLSTATE 428FG)이 아니어야 합니다.

DROP SECURITY POLICY

테이블에서 보안 규정 및 모든 LBAC 보호를 삭제합니다. 보안 규정은 *table-name*이 지정한 테이블을 보호해야 합니다(SQLSTATE 428GT). 테이블에 데이터 유형 DB2SECURITYLABEL을 가진 컬럼이 있는 경우 데이터 유형은 VARCHAR(128) FOR BIT DATA로 변경됩니다. 테이블에 하나 이상의 보호된 컬럼이 있는 경우 그 컬럼은 비보호 상태가 됩니다.

ADD *column-definition*

테이블에 컬럼을 추가합니다. 테이블은 유형이 지정된 테이블이어서는 안됩니다(SQLSTATE 428DH). 테이블에 있는 모든 기존 행의 경우 새 컬럼의 값은 디폴트값으로 설정됩니다. 새 컬럼은 테이블의 마지막 컬럼이 되는데, 이는 초기에 *n*개의 컬럼이 있으면 추가되는 컬럼은 컬럼 *n*+1임을 의미합니다.

새 컬럼을 추가해도 모든 컬럼의 총 바이트 수가 최대 레코드 크기를 초과해서는 안됩니다.

column-name

테이블에 추가될 컬럼의 이름입니다. 이름은 규정할 수 없습니다. 테이블에 있는 기존 컬럼 이름은 사용할 수 없습니다(SQLSTATE 42711).

data-type

『CREATE TABLE』 아래에 나열된 데이터 유형 중 하나입니다.

NOT NULL

컬럼에 널(NULL)값이 포함되지 못하도록 합니다. *default-clause*도 지정해야 합니다(SQLSTATE 42601).

NOT HIDDEN 또는 IMPLICITLY HIDDEN

컬럼이 숨김으로 정의되어 있는지 여부를 지정합니다. 숨겨진 속성은 컬럼이 테이블에 대한 내재적 참조에 포함되어 있는지 여부 또는 명시적으로 SQL문을 참조할 수 있는지 여부를 판별합니다. 디폴트값은 NOT HIDDEN입니다.

NOT HIDDEN

컬럼이 테이블에 대한 내재적 참조에 포함되는지, 그리고 컬럼이 명시적으로 참조될 수 있는지를 지정합니다.

IMPLICITLY HIDDEN

컬럼이 명시적으로 이름으로 참조되지 않는 경우 해당 컬럼이 SQL문에서 볼 수 없음을 지정합니다. 예를 들어, 테이블에 IMPLICITLY HIDDEN 절로 정의된 컬럼이 포함되어 있다고 가정하면, SELECT *에는 내재적으로 숨겨진 컬럼이 포함되지 않습니다. 그러나 내재적으로 숨겨진 컬럼의 이름을 명시적으로 참조하는 SELECT의 결과에 결과 테이블의 해당 컬럼이 포함됩니다.

IMPLICITLY HIDDEN은 ROW CHANGE TIMESTAMP 컬럼에 대해서만 지정되어야 합니다(SQLSTATE 42867). ROW CHANGE TIMESTAMP FOR *table-designator* 표현식은 IMPLICITLY HIDDEN ROW CHANGE TIMESTAMP 컬럼으로 해결합니다. 그러므로 ROW CHANGE TIMESTAMP 컬럼은 IMPLICITLY HIDDEN으로 테이블에 추가될 수 있으며, 컬럼을 처리하기 위해 이 테이블의 SELECT *에 추가되는 기존 응용프로그램을 수정할 필요가 없습니다. 표현식을 사용하여, 컬럼 이름을 알지 않아도 새 응용프로그램은 항상 컬럼에 액세스할 수 있습니다.

lob-options

LOB 데이터 유형에 대한 옵션을 지정합니다. 『CREATE TABLE』에서 *lob-options*를 참조하십시오.

SCOPE

참조 유형 컬럼의 범위를 지정합니다.

typed-table-name2

유형이 지정된 테이블의 이름입니다. *column-name*의 데이터 유형은 REF(*S*)이어야 합니다. 여기서, *S*는 *typed-table-name2* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name2*의 기존 행을 참조하는지 확인하기 위해 *column-name*의 디폴트값을 점검하지는 않습니다.

typed-view-name2

유형이 지정된 뷰의 이름입니다. *column-name*의 데이터 유형은 REF(*S*)이어야 합니다. 여기서, *S*는 *typed-view-name2* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-view-name2*의 기존 행을 참조하는지 확인하기 위해 *column-name*의 디폴트값을 점검하지는 않습니다.

CONSTRAINT *constraint-name*

제한조건이 이름을 지정합니다. *constraint-name*은 같은 ALTER TABLE문 내에 지정된 제한조건 또는 다른 기존의 테이블 제한조건 이름으로 이미 지정된 제한조건과 동일해서는 안됩니다(SQLSTATE 42710).

사용자가 제한조건 이름을 지정하지 않은 경우, 테이블에 정의된 기존 제한조건 ID 내에 18바이트의 고유한 긴 ID가 생성됩니다. ID는 ″SQL″과 ″SQL″ 뒤에 시간소인 기반 함수가 생성하는 15개의 숫자 시퀀스로 이루어집니다.

PRIMARY KEY 또는 UNIQUE 제한조건과 함께 사용할 경우, 제한조건을 지원하기 위해 작성된 인덱스의 이름으로 *constraint-name*을 사용할 수 있습니다. 고유 제한조건과 연관된 인덱스 이름에 대한 자세한 내용은 주를 참조하십시오.

PRIMARY KEY

단일 컬럼으로 구성되는 기본 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 PRIMARY KEY가 C 컬럼의 정의에 지정된 경우, PRIMARY KEY(C)절이 별도의 절로 지정된 경우와 결과가 같습니다. 컬럼에는 널(NULL) 값이 포함될 수 없으므로, NOT NULL 속성도 지정해야 합니다(SQLSTATE 42831).

다음의 *unique-constraint* 설명에 있는 PRIMARY KEY를 참조하십시오.

UNIQUE

단일 컬럼으로 구성되는 고유 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 UNIQUE가 C 컬럼의 정의에 지정된 경우, UNIQUE(C)절이 별도의 절로 지정된 경우와 결과가 같습니다.

다음의 *unique-constraint* 설명에 있는 UNIQUE를 참조하십시오.

references-clause

단일 컬럼으로 구성되는 외부 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 references-clause가 C 컬럼의 정의에 지정된 경우, C가 유일하게 식별되는 컬럼인 FOREIGN KEY절의 일부로 references-clause를 지정한 것과 그 결과가 같습니다.

『CREATE TABLE』에서 *references-clause*를 참조하십시오.

CHECK (*check-condition***)**

단일 컬럼에 적용되는 점검 제한조건을 정의함에 있어 간편한 방법을 제공합니다. 『CREATE TABLE』에서 *check-condition*을 참조하십시오.

generated-column-definition

컬럼 생성에 대한 자세한 내용은 『CREATE TABLE』을 참조하십시오.

default-clause

컬럼의 디폴트값을 지정합니다.

WITH

선택적 키워드입니다.

DEFAULT

값이 INSERT에 제공되지 않거나 INSERT 또는 UPDATE의 DEFAULT로 지정된 이벤트에서 디폴트값을 제공합니다. 특정 디폴트값을 DEFAULT 키워드 뒤에 지정하지 않으면, 디폴트값은 122 페이지의

지의 표 13에 표시된 컬럼의 데이터 유형에 따라 달라집니다. 컬럼이 XML 또는 구조화된 유형으로 정의된 경우 DEFAULT절을 지정할 수 없습니다.

컬럼이 구별 유형을 사용하여 정의되면, 컬럼의 디폴트값은 구별 유형으로 캐스트되는 소스 데이터 유형의 디폴트값입니다.

표 13. 디폴트값(값을 지정하지 않은 경우)

데이터 유형	디폴트값
숫자	0
고정 길이 문자열	공백
가변 길이 문자열	길이가 0인 문자열
고정 길이 그래픽 문자열	2바이트 공백
가변 길이 그래픽 문자열	길이가 0인 문자열
날짜	기존의 행의 경우, 1월 1일(0001)에 해당되는 날짜. 추가된 행의 경우, 현재 날짜
시간	기존의 행의 경우, 0시, 0분, 0초에 해당되는 시간. 추가된 행의 경우, 현재 시간
시간소인	기존 행의 경우, 1월 1일(0001)에 해당되는 날짜와, 0시, 0분, 0초에 해당되는 날짜. 추가된 행의 경우, 현재 시각
실행 파일 문자열(BLOB)	길이가 0인 문자열

*column-definition*에서 DEFAULT를 생략하면, 컬럼의 디폴트값으로 널(NULL)값이 사용됩니다.

DEFAULT 키워드와 함께 지정할 수 있는 특정 값 유형은 다음과 같습니다.

constant

컬럼의 디폴트값으로 상수를 지정합니다. 지정된 상수는 다음과 같아야 합니다.

- 제 3 장에 설명된 지정 규칙에 따라 컬럼에 지정할 수 있는 값을 나타내야 합니다.
- 컬럼이 부동 소수점 데이터 유형으로 정의되어 있지 않는 한, 부동 소수점 상수가 아니여야 합니다.
- 컬럼의 데이터 유형이 10진수 부동 소수점인 경우 숫자 상수 또는 10진수 부동 소수점 특수 값이어야 합니다. 부동 소수점 상수는 먼저 DOUBLE로 해석된 다음 10진수 부동 소수점으로 변환됩니다. DECFLOAT(16) 컬럼의 경우, 10진 상수는 16 이하의 정밀도여야 합니다.

- 상수가 10진 상수인 경우 컬럼 데이터 유형의 전체 자릿수를 초과하는 0이 아닌 자릿수를 갖지 않아야 합니다(예를 들면, 1.234는 DECIMAL(5,2) 컬럼에 대한 디폴트값이 될 수 없음).
- 따옴표, 16진 상수를 나타내는 X와 같은 도입 문자 및 상수가 *cast-function*의 인수일 때 완전한 함수 이름의 문자 및 괄호를 포함하여 254바이트 이하로 표시됩니다.

datetime-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 날짜 시간 특수 레지스터의 값(CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP)을 지정합니다. 컬럼의 데이터 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다. 예를 들면, CURRENT DATE를 지정할 경우 데이터 유형은 DATE여야 합니다. 기존 행의 경우, 이 값은 ALTER TABLE 문이 처리되는 현재 날짜, 현재 시간 또는 현재 시각입니다.

user-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시에 사용자 특수 레지스터의 값(CURRENT USER, SESSION_USER, SYSTEM_USER)을 지정합니다. 컬럼의 데이터 유형은 사용자 특수 레지스터의 길이 속성보다 짧지 않은 길이의 문자열이어야 합니다. SESSION_USER 대신 USER를, CURRENT USER 대신 CURRENT_USER를 지정할 수 있습니다. 기존 행의 경우, 이 값은 ALTER TABLE문의 CURRENT USER, SESSION_USER 또는 SYSTEM_USER입니다.

CURRENT SCHEMA

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 CURRENT SCHEMA 특수 레지스터값을 지정합니다. CURRENT SCHEMA를 지정할 경우, 컬럼의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성과 크거나 같은 길이의 문자열이어야 합니다. 기존 행의 경우, CURRENT SCHEMA 특수 레지스터의 값은 ALTER TABLE문이 처리되는 시간입니다.

NULL

컬럼의 디폴트값으로 널(NULL)을 지정합니다. NOT NULL이 지정된 경우, 동일한 컬럼 정의 내에 DEFAULT NULL을 지정해서는 안됩니다.

cast-function

이 형식의 디폴트값은 구별 유형, BLOB 또는 날짜 시간(DATE, TIME 또는 TIMESTAMP) 데이터 유형으로 정의된 컬럼과 함께 사용할 수 있습니다. 구별 유형의 경우, BLOB 또는 날짜 시간 유

형을 따르는 구별 유형을 제외하고는 함수 이름이 컬럼의 구별 유형 이름과 일치해야 합니다. 스키마 이름으로 규정된 경우, 함수 이름이 구별 유형에 대한 스키마 이름과 같아야 합니다. 규정되지 않은 경우, 함수 결정의 스키마 이름은 구별 유형에 대한 스키마 이름과 같아야 합니다. 디폴트값이 상수인 날짜 시간 유형을 따르는 구별 유형의 경우, 함수가 사용되어야 하며, 함수 이름은 내재적 또는 명시적 스키마 이름이 SYSIBM인 구별 유형의 소스 유형 이름과 일치해야 합니다. 다른 날짜 시간 컬럼의 경우, 해당하는 날짜 시간 함수가 사용될 수도 있습니다. BLOB 또는 BLOB을 기반으로 하는 구별 유형의 경우, 함수가 사용되어야 하며 함수 이름은 내재적 또는 명시적 스키마 이름으로 SYSIBM을 사용하는 BLOB이어야 합니다.

constant
상수를 인수로 지정합니다. 상수는 구별 유형의 소스 유형 또는 데이터 유형(구별 유형이 아닌 경우)에 대한 상수 규칙을 따라야 합니다. *cast-function*이 BLOB인 경우, 상수는 문자열 상수여야 합니다.

datetime-special-register
CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다.

user-special-register
CURRENT USER, SESSION_USER 또는 SYSTEM_USER를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 최소한 8바이트 길이의 문자열 데이터 유형이어야 합니다. *cast-function*이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

CURRENT SCHEMA
CURRENT SCHEMA 특수 레지스터의 값을 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성보다 크거나 같은 문자열이어야 합니다. *cast-function*이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

EMPTY_CLOB(), EMPTY_DBCLOB() 또는 EMPTY_BLOB()
컬럼의 디폴트값으로 영(0) 길이의 문자열을 지정합니다. 컬럼의 데이터 유형은 함수의 결과 데이터 유형을 기반으로 해야 합니다.

지정된 값이 유효하지 않으면, 오류가 발생합니다(SQLSTATE 42894).

GENERATED

DB2가 컬럼의 값을 생성하도록 지정합니다.

ALWAYS

행이 테이블에 삽입될 때 또는 *generation-expression*의 결과 값이 변경될 수 있을 때마다 DB2에서 항상 컬럼 값을 생성하도록 지정합니다. 표현식의 결과는 테이블에 저장됩니다. GENERATED ALWAYS는 데이터 전파, 언로드 또는 다시 로드 조작을 수행하지 않는 경우에 권장하는 옵션입니다. GENERATED ALWAYS는 생성된 컬럼의 필수 값입니다.

BY DEFAULT

명시적 값이 지정되어 있지 않으면 컬럼에 DEFAULT를 지정하여 테이블에 행을 삽입하거나 테이블을 갱신할 때 DB2가 컬럼에 대한 값을 생성하도록 지정합니다. BY DEFAULT는 데이터 전파 사용시 또는 언로드와 재로드 조작 수행시 권장되는 옵션입니다.

AS (*generation-expression***)**

컬럼의 정의가 표현식을 기반으로 하도록 지정합니다. OFF 옵션과 함께 SET INTEGRITY문을 사용하여 테이블이 무결성 설정 보류 상태가 되도록 요구합니다. ALTER TABLE문 다음에, IMMEDIATE CHECKED 및 FORCE GENERATED 옵션이 있는 SET INTEGRITY문을 사용하여 새 표현식에 대해 해당 컬럼의 모든 값을 갱신하고 점검해야 합니다. *generation-expression* 컬럼의 지정에 대한 자세한 내용은 『CREATE TABLE』을 참조하십시오.

COMPRESS SYSTEM DEFAULT

시스템 디폴트값(특정 값을 지정하지 않았을 때 데이터 유형에 사용되는 디폴트값)을 최소한의 스페이스를 사용하여 저장하도록 지정합니다. VALUE COMPRESSION절을 지정하지 않으면 경고가 리턴되고(SQLSTATE 01648) 시스템 디폴트값이 최소한의 스페이스를 사용하여 저장되지 않습니다.

이와 같은 방식으로 시스템 디폴트값이 저장되도록 하면 추가 점검이 수행되기 때문에 컬럼에 대해 삽입 및 갱신 조작을 수행하는 동안 성능이 약간 저하될 수 있습니다.

기본 데이터 유형은 DATE, TIME, TIMESTAMP, XML 또는 구조화된 데이터 유형일 수 없습니다(SQLSTATE 42842). 기본 데이터 유형이 가변 길이 문자열일 경우에는 이 절이 무시됩니다. VALUE COMPRESSION을 사용하여 테이블을 설정한 경우에는 길이가 0인 문자열 값이 자동으로 압축됩니다.

COLUMN SECURED WITH *security-label-name*

테이블과 연결된 보안 규정에 대해 존재하는 보안 레이블을 식별합니다. name은 규정하지 않아야 합니다(SQLSTATE 42601). 테이블에는 테이블과 연결된 보안 규정이 있어야 합니다(SQLSTATE 55064).

ADD *unique-constraint*

고유 제한조건이나 기본 키 제한조건을 정의합니다. 서브테이블인 테이블에는 기본 키 또는 고유 제한조건을 추가할 수 없습니다(SQLSTATE 429B3). 테이블이 계층 맨 위에 있는 슈퍼 테이블인 경우, 제한조건이 해당 테이블 및 모든 서브테이블에 적용됩니다.

CONSTRAINT *constraint-name*

기본 키 또는 고유 제한조건의 이름을 지정합니다. 자세한 내용은 『CREATE TABLE』에서 *constraint-name*을 참조하십시오.

UNIQUE (*column-name*...,)

식별된 컬럼으로 구성되는 고유 키를 정의합니다. 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별하면 안됩니다.이름은 규정할 수 없습니다. 식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오. 컬럼의 길이 속성이 페이지 크기에 대한 인덱스 키 길이 한계를 초과하지 않아도 LOB, 이러한 유형 중 하나라도 기반으로 하는 구별 유형 또는 구조화된 유형은 고유 키의 일부로 사용될 수 없습니다(SQLSTATE 54008). 고유 키의 컬럼 세트는 기본 키 또는 다른 고유 키의 컬럼 세트와 동일할 수 없습니다(SQLSTATE 01543). LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 발생합니다(SQLSTATE 42891). 식별된 컬럼 세트에 있는 기존의 모든 값은 고유해야 합니다(SQLSTATE 23515).

기존 인덱스가 인덱스의 모든 INCLUDE 컬럼은 무시하고 고유한 키 정의와 일치하는지 판별하기 위해 점검이 수행됩니다. 컬럼의 순서나 방향(ASC/DESC) 스펙에 관계없이 동일한 컬럼 세트를 식별할 경우, 인덱스 정의가 일치하는 것입니다. 그러나 파티션된 테이블의 경우에는 테이블 파티션 키 컬럼의 수퍼 세트가 아닌 고유하지 않은 파티션된 인덱스가 일치하는 인덱스로 간주되지 않습니다.

일치하는 인덱스 정의가 있는 경우, 시스템에 필요한 인덱스임을 나타내도록 인덱스 설명이 변경되고, 고유하지 않은 인덱스일 경우 고유성을 확인한 후 고유한 인덱스로 변경됩니다. 테이블에 일치하는 인덱스가 두 개 이상 있으면 기존의 고유 인덱스가 선택됩니다. 여러 개의 고유 인덱스가 있는 경우 다음과 같이 선택에 임의의 예외가 한 가지 있습니다.

- 파티션된 테이블의 경우 파티션된 일치하는 고유 인덱스가 파티션되지 않은 일치하는 고유 인덱스나 파티션되거나 파티션되지 않은 일치하는 비고유 인덱스보다 우선시됩니다.

일치하는 인덱스가 없는 경우 CREATE TABLE에 기술된 것과 같이 컬럼에 대해 고유한 양방향 인덱스가 자동으로 작성됩니다. 고유 제한조건과 연관된 인덱스 이름에 대한 자세한 내용은 주를 참조하십시오.

PRIMARY KEY *...(column-name,)*

식별된 컬럼으로 구성되는 기본 키를 정의합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼이 두 번 이상 식별되어서는 안됩니다. 이름은 규정할 수 없습니다. 식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오. 테이블에는 기본 키가 없어야 하고 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 컬럼의 길이 속성이 페이지 크기에 대한 인덱스 키 길이 한계를 초과하지 않아도 LOB, 이러한 유형 중 하나라도 기반으로 하는 구별 유형 또는 구조화된 유형은 기본 키의 일부로 사용될 수 없습니다(SQLSTATE 54008). 기본 키의 컬럼 세트는 고유 키의 컬럼 세트와 같을 수 없습니다(SQLSTATE 01543). LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 발생합니다(SQLSTATE 42891). 식별된 컬럼 세트에 있는 기존의 모든 값은 고유해야 합니다(SQLSTATE 23515).

기존 인덱스가 기본 키 정의와 일치하는지 판별하기 위해 점검이 수행됩니다. 인덱스의 모든 INCLUDE 컬럼은 무시합니다. 컬럼의 순서나 방향(ASC/DESC) 스펙에 관계없이 동일한 컬럼 세트를 식별할 경우, 인덱스 정의가 일치하는 것입니다. 그러나 파티션된 테이블의 경우에는 테이블 파티션 키 컬럼의 수퍼 세트가 아닌 고유하지 않은 파티션된 인덱스가 일치하는 인덱스로 간주되지 않습니다.

일치하는 인덱스 정의가 있는 경우, 시스템에서 필요한 1차 인덱스임을 나타내도록 인덱스의 설명이 변경되고, 고유하지 않은 인덱스일 경우 고유성을 확인한 후 고유한 인덱스로 변경됩니다. 테이블에 일치하는 인덱스가 두 개 이상 있을 경우, 기존의 고유 인덱스가 선택됩니다. 여러 개의 고유 인덱스가 있는 경우 다음과 같이 선택에 임의의 하나의 예외가 있습니다.

- 파티션된 테이블의 경우 파티션된 일치하는 고유 인덱스가 파티션되지 않은 일치하는 고유 인덱스나 (파티션되거나 파티션되지 않은) 일치하는 고유하지 않은 인덱스보다 선호됩니다.

일치하는 인덱스가 없는 경우 CREATE TABLE에 기술된 것과 같이 컬럼에 대해 고유한 양방향 인덱스가 자동으로 작성됩니다. 고유 제한조건과 연관된 인덱스 이름에 대한 자세한 내용은 주를 참조하십시오.

하나의 테이블에서 하나의 기본 키만 정의할 수 있습니다.

ADD *referential-constraint*

참조 제한조건을 정의합니다. 『CREATE TABLE』에서 *referential-constraint*를 참조하십시오.

ADD *check-constraint*

점검 제한조건 또는 함수적 종속성을 정의합니다. 『CREATE TABLE』에서 *check-constraint*를 참조하십시오.

ADD *distribution-clause*

분산 키를 정의합니다. 테이블은 단일 파티션 데이터베이스 파티션 그룹의 테이블 스페이스에 정의되어 있어야 하며(SQLSTATE 55037) 분산 키를 이미 가지고 있어서는 안됩니다(SQLSTATE 42889) 분산 키가 이미 테이블에 존재할 경우 기존 키는 새 분산 키를 추가하기 전에 삭제되야 합니다. 서브테이블인 테이블에는 분산 키를 추가할 수 없습니다(SQLSTATE 428DH). .

DISTRIBUTE BY HASH (*column-name***...)**

지정된 컬럼을 사용하여 분산 키를 정의합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼이 두 번 이상 식별되어서는 안됩니다. 이름은 규정할 수 없습니다. 컬럼의 데이터 유형이 BLOB, CLOB, DBCLOB, XML, 이 유형 중 하나 이상에 대한 구별 유형 또는 구조화된 유형일 경우, 컬럼은 분산 키의 일부로 사용될 수 없습니다.

ADD RESTRICT ON DROP

테이블과 테이블이 포함된 테이블 스페이스를 삭제할 수 없도록 지정합니다.

ADD MATERIALIZED QUERY

materialized-query-definition

쿼리 최적화 중에 사용하기 위해 일반 테이블을 구체화된 쿼리 테이블로 변경합니다. *table-name*으로 지정된 테이블은 다음과 같아야 합니다.

- 이전에 구체화된 쿼리 테이블로 정의된 적이 없어야 합니다.
- 유형이 지정된 테이블이 아니어야 합니다.
- 제한조건, 고유 인덱스 또는 트리거가 정의되어 있지 않아야 합니다.
- 캐싱 사용 불가능으로 표시된 별칭을 참조하지 않아야 합니다.
- 다른 구체화된 쿼리 테이블의 정의에서 참조되지 않아야 합니다.
- 쿼리 최적화의 사용이 가능한 뷰의 정의에서 참조되지 않아야 합니다.

*table-name*이 이 기준과 일치하지 않을 경우, 오류가 발생합니다(SQLSTATE 428EW).

fullselect

테이블이 기반으로 하는 쿼리를 정의합니다. 기존 테이블의 컬럼은

fullselect의 결과 컬럼과 다음 사항이 같아야 합니다(SQLSTATE 428EW).

- 컬럼 수가 같아야 합니다.
- 데이터 유형이 정확하게 같아야 합니다.
- 같은 순서 위치에 같은 컬럼 이름이 있어야 합니다.

*fullselect*의 결과 컬럼으로(SQLSTATE 428EW). 구체화된 쿼리 테이블에 대해 *fullselect*를 지정하는 자세한 내용은 『CREATE TABLE』을 참조하십시오. 하나의 추가 제한사항은 fullselect에서 *table-name*을 직접 또는 간접적으로 참조할 수 없다는 것입니다.

refreshable-table-options

구체화된 쿼리 테이블 변경을 위한 새로 고침 옵션을 지정합니다.

DATA INITIALLY DEFERRED

테이블의 데이터는 REFRESH TABLE이나 SET INTEGRITY문을 사용하여 유효성이 확인되어야 합니다.

REFRESH

테이블의 데이터가 유지보수되는 방식을 나타냅니다.

DEFERRED

테이블의 데이터가 REFRESH TABLE문을 사용하여 언제라도 새로 고쳐질 수 있습니다. 테이블의 데이터는 REFRESH TABLE문이 처리될 때의 스냅샷인 쿼리 결과에만 영향을 미칩니다. 이 속성을 사용하여 정의되어 있는 구체화된 쿼리 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다(SQLSTATE 42807).

IMMEDIATE

DELETE, INSERT 또는 UPDATE의 일부로서 하위 테이블에 대해 수행된 변경사항은 구체화된 쿼리 테이블에 연쇄됩니다. 이 경우, 특정 시점에 테이블의 내용은 지정된 subselect가 처리될 경우와 같습니다. 이 속성을 사용하여 정의된 구체화된 쿼리 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다(SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

쿼리 최적화를 위해 구체화된 쿼리 테이블을 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

구체화된 쿼리 테이블을 쿼리 최적화에 사용할 수 없습니다. 해당 테이블은 여전히 직접 쿼리됩니다.

MAINTAINED BY

구체화된 쿼리 테이블의 데이터가 시스템, 사용자 또는 복제 도구에 의해 유지보수될지 여부를 지정합니다.

SYSTEM

구체화된 쿼리 테이블의 데이터를 시스템에서 유지보수하도록 지정합니다.

USER

구체화된 쿼리 테이블의 데이터를 사용자가 유지보수하도록 지정합니다. 사용자는 사용자가 유지보수하는 구체화된 쿼리 테이블에 대해 갱신, 삭제, 삽입 조작을 수행할 수 있습니다. 시스템에서 유지보수하는 구체화된 쿼리 테이블에 사용되는 REFRESH TABLE문은 사용자가 유지보수하는 구체화된 쿼리 테이블에 대하여 호출할 수 없습니다. REFRESH DEFERRED 구체화된 쿼리 테이블만 MAINTAINED BY USER로 정의할 수 있습니다.

FEDERATED_TOOL

구체화된 쿼리 테이블의 데이터를 복제 도구에서 유지보수하도록 지정합니다. 시스템에서 유지보수하는 구체화된 쿼리 테이블에 사용되는 REFRESH TABLE문은 페더레이티드 도구가 유지보수하는 구체화된 쿼리 테이블에 대하여 호출할 수 없습니다. REFRESH DEFERRED 구체화된 쿼리 테이블만 MAINTAINED BY FEDERATED_TOOL로 정의할 수 있습니다.

ALTER FOREIGN KEY *constraint-name*

참조 제한조건 *constraint-name*의 제한조건 속성을 변경합니다. *constraint-name*은 기존의 참조 제한조건을 식별해야 합니다(SQLSTATE 42704).

ALTER CHECK *constraint-name*

점검 제한조건 또는 함수적 종속성 *constraint-name*의 제한조건 속성을 변경합니다. *constraint-name*은 기존 점검 제한조건 또는 함수적 종속성을 식별해야 합니다 (SQLSTATE 42704).

constraint-alteration

참조 제한조건이나 점검 제한조건과 연관된 속성을 변경하기 위한 옵션입니다.

ENABLE QUERY OPTIMIZATION 또는 DISABLE QUERY OPTIMIZATION

적절한 환경에서 쿼리 최적화를 위해 제한조건 또는 함수적 종속성을 사용할 수 있는지 여부를 지정합니다.

ENABLE QUERY OPTIMIZATION

제한조건을 참으로 가정하며 쿼리 최적화에 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

제한조건을 쿼리 최적화에 사용할 수 없습니다.

ENFORCED 또는 NOT ENFORCED

제한조건이 삽입, 갱신 또는 삭제 등 정상 조작 중에 데이터베이스 관리 프로그램에 의해 시행되는지 여부를 지정합니다.

ENFORCED

제한조건을 ENFORCED로 변경합니다. ENFORCED를 함수적 종속성에 대해서는 지정할 수 없습니다(SQLSTATE 42621).

NOT ENFORCED

제한조건을 NOT ENFORCED로 변경합니다. 이 옵션은 각 테이블 데이터가 개별적으로 제한조건을 확인해야 할 경우에만 지정해야 합니다. 데이터가 실제로 제한조건을 준수하지 않는 경우 쿼리 결과는 예상할 수 없습니다.

ALTER *column-alteration*

컬럼의 정의를 변경합니다. 지정된 속성만 변경되고 다른 속성은 변경되지 않습니다. 유형이 지정된 테이블의 컬럼은 변경될 수 없습니다(SQLSTATE 428DH).

column-name

변경할 컬럼의 이름을 지정합니다. *column-name*은 테이블의 기존 컬럼을 식별해야 합니다(SQLSTATE 42703). 이름을 규정해서는 안됩니다. 이름은 동일한 ALTER TABLE문에서 추가, 변경 또는 삭제되는 컬럼을 식별해서는 안됩니다(SQLSTATE 42711).

SET DATA TYPE *altered-data-type*

컬럼의 새 데이터 유형을 지정합니다. 새 데이터 유형은 컬럼의 기존 데이터 유형과 호환 가능해야 합니다(SQLSTATE 42837). 기존 데이터를 절단하지 않고 VARCHAR 또는 VARGRAPHIC 컬럼의 길이를 변경하면 테이블의 연속 재구성이 필요하지 않습니다. 뒤 공백만 절단되면 테이블이 완전히 액세스되기 전에 테이블 재구성이 필요합니다(SQLSTATE 57016). 관리 루틴 SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS를 호출하여 필요한 경우 테이블 재구성을 수행할 수 있습니다. 비공백 문자의 절단은 허용되지 않습니다(SQLSTATE 42837).

컬럼이 데이터 파티션 키의 컬럼인 경우 문자열 데이터 유형을 변경할 수 없습니다.

컬럼이 분산 키의 컬럼이면 DECFLOAT(16)가 DECFLOAT(34)로 변경된 경우를 제외하고는 새 데이터 유형이 REAL, DOUBLE, DECFLOAT(16), DECFLOAT(34), 또는 DECIMAL(*p*, *m*)(*p* − *m* > 4인 경우)이어서는 안됩니다.

데이터 유형이 LOB인 경우 지정된 길이는 기존 길이보다 작을 수 없습니다(SQLSTATE 42837).

식별 컬럼의 데이터 유형은 변경될 수 없습니다(SQLSTATE 42997).

해당 테이블은 데이터 캡처 기능을 사용 가능하게 할 수 없습니다(SQLSTATE 42997).

컬럼을 변경해도 모든 컬럼의 총 바이트 수가 지정된 최대 레코드 *크기*를 초과해서는 안됩니다(SQLSTATE 54010). 고유 제한조건 또는 인덱스에서 컬럼이 사용되는 경우, 새로운 길이는 고유 제한조건 또는 인덱스에 대해 저장된 길이의 합이 페이지 *크기*에 대한 인덱스 키 길이 한계를 초과하면 안됩니다(SQLSTATE 54008). 컬럼의 저장된 길이에 대해서는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오.

auto_reval이 DISABLED로 설정된 경우, 컬럼 변경의 연쇄 효과가 표 14에 표시됩니다.

표 14. 컬럼 변경의 연쇄 효과

조작	효과
뷰 또는 점검 제한조건이 참조하는 컬럼 변경	오브젝트는 변경 처리 중에 다시 생성됩니다. 뷰의 경우 오브젝트에 대한 함수 또는 메소드 분석은 변경 조작, 즉 오브젝트의 시맨틱 변경 이후 달라질 수 있습니다. 점검 제한조건의 경우 변경 조작의 결과로 오브젝트의 시맨틱이 변경되면 해당 조작을 할 수 없습니다.
종속 패키지, 트리거 또는 SQL 루틴이 있는 테이블의 컬럼 변경	오브젝트는 유효하지 않음으로 표시되고 다음 사용 시 유효성이 다시 확인됩니다.
분석 기능 사용이 가능한 XSROBJECT가 참조하는 테이블의 컬럼 유형 변경	XSROBJECT는 분석 작동 불능으로 표시됩니다. XSROBJECT를 다시 사용 가능하게 하려면 맵핑의 재조정이 필요하며, XSROBJECT에 대해 ALTER XSROBJECT ENABLE DECOMPOSITION문을 발행하십시오.
전역 변수의 디폴트 표현식에서 참조되는 컬럼 변경	변경 처리 동안 전역 변수의 디폴트 표현식의 유효성이 확인됩니다. 디폴트 표현식에서 사용된 사용자 정의 함수(UDF)를 해결할 수 없으면 연산이 실패합니다.

SET *generated-column-alteration*

컬럼의 값을 생성하는 데 사용되는 기술을 지정합니다. 이것은 특정 디폴트값 또는 표현식 형태이거나 컬럼을 식별 컬럼으로 정의할 수 있습니다. 컬럼의 기존 디폴트값이 다른 생성 기술에서 작성된 경우, 그 디폴트값은 삭제해야 하는데 이는 DROP절 중 하나를 사용하여 동일한 *column-alteration*에서 실행할 수 있습니다.

default-clause

변경할 컬럼의 새 디폴트값을 지정합니다. 컬럼이 이미 식별 컬럼으로 정의되어 있거나 컬럼에 생성 표현식이 정의되어 있으면 안됩니다(SQLSTATE 42837). 지정된 디폴트값은 『지정 및 비교』에 설명된 지정 규칙에 따라 컬럼에 지정할 수 있는 값을 나타내야 합니다. 디폴트값을 변경해도 이 컬럼과 연관된 기존 행의 값은 변경되지 않습니다.

GENERATED ALWAYS 또는 GENERATED BY DEFAULT

데이터베이스 관리 프로그램이 컬럼 값을 생성하는 시기를 지정합니다. GENERATED BY DEFAULT는 값을 제공하지 않았거나 컬럼 지정에 DEFAULT 키워드를 사용하는 경우에만 값을 생성하도록 지정합니다. GENERATED ALWAYS는 데이터베이스 관리 프로그램이 항상 컬럼 값을 생성하도록 지정합니다. GENERATED BY DEFAULT는 *generation-expression*과 함께 지정할 수 없습니다.

identity-options

컬럼이 테이블의 식별 컬럼이 되도록 지정합니다. 컬럼이 이미 식별 컬럼으로 정의되어 있거나, 컬럼에 생성 표현식이 있거나 명시된 디폴트 값이 있으면 안됩니다(SQLSTATE 42837). 하나의 테이블은 하나의 식별 컬럼만 가질 수 있습니다(SQLSTATE 428C1). 컬럼은 널(NULL)이 허용되지 않도록 지정되고(SQLSTATE 42997), 컬럼과 연관된 데이터 유형이 소수점 이하 자릿수가 0인 정확한 숫자 데이터 유형이어야 합니다(SQLSTATE 42815). 정확한 숫자 데이터 유형은 소수점 이하 자릿수가 0인 SMALLINT, INTEGER, BIGINT, DECIMAL 또는 NUMERIC이거나 이러한 유형 중 하나를 기반으로 하는 구별 유형입니다. 식별 옵션에 대한 자세한 내용은 『CREATE TABLE』을 참조하십시오.

AS (*generation-expression*)

컬럼의 정의가 표현식을 기반으로 하도록 지정합니다. 컬럼이 이미 생성 표현식으로 정의되어 있거나, 식별 컬럼이거나 컬럼에 명시된 디폴트값이 있으면 안됩니다(SQLSTATE 42837). *generation-expression*은 생성된 컬럼을 정의할 때 적용되는 것과 같은 규칙을 따라야 합니다. *generation-expression*의 결과 데이터 유형은 컬럼의 데이터 유형에 지정할 수 있어야 합니다(SQLSTATE 42821). 컬럼을 분산 키 컬럼 또는 ORGANIZE BY절에서 참조해서는 안됩니다(SQLSTATE 42997).

SET EXPRESSION AS (*generation-expression*)

컬럼에 대한 표현식을 지정된 *generation-expression*으로 변경합니다. SET EXPRESSION AS는 OFF 옵션과 함께 SET INTEGRITY문을 사용하여 테이블이 무결성 설정 보류 상태가 되도록 요구합니다. ALTER TABLE문 다음에, IMMEDIATE CHECKED 및 FORCE GENERATED 옵션이 있는 SET INTEGRITY문을 사용하여 새 표현식에 대해 해당 컬럼의 모든 값을 갱신하고 점검해야 합니다. 컬럼은 이 표현식에 기반하여 생성된 컬럼으로 이미 정의되어 있어야 하고(SQLSTATE 42837), 테이블의 PARTITIONING KEY, DIMENSIONS 또는 KEY SEQUENCE절에는 나타나지 않아야 합니다(SQLSTATE 42997). generation-expression은 생성된 컬럼을 정의할 때 적

용되는 것과 같은 규칙을 따라야 합니다. generation-expression의 결과 데이터 유형은 컬럼의 데이터 유형에 지정할 수 있어야 합니다(SQLSTATE 42821).

SET INLINE LENGTH *integer*

기존 구조화된 유형, XML 또는 LOB 데이터 유형 컬럼의 인라인 길이를 변경합니다. 인라인 길이는 기본 테이블 행에 저장하기 위한 구조화된 유형, XML 또는 LOB 데이터 유형의 인스턴스의 최대 바이트 *크기*를 표시합니다. 기본 테이블 행의 인라인으로 저장할 수 없는 구조화된 유형의 인스턴스 또는 XML 데이터 유형은 LOB 값이 저장된 방법과 유사하게 별도로 저장됩니다.

*column-name*의 데이터 유형은 구조화된 유형, XML 또는 LOB 데이터 유형이어야 합니다(SQLSTATE 42842).

구조화된 유형 컬럼의 디폴트 인라인 길이는 해당 데이터 유형의 인라인 길이로서 CREATE TYPE문에 명시적으로 또는 디폴트로 지정됩니다. 구조화된 유형의 인라인 길이가 292보다 작은 경우, 값 292가 컬럼의 인라인 길이로 사용됩니다.

명시적 인라인 길이 값은 증가할 수만 있으며(SQLSTATE 429B2), 32673을 초과할 수 없습니다(SQLSTATE 54010). 구조화된 유형이나 XML 데이터 유형 컬럼의 경우 적어도 292여야 합니다. LOB 데이터 유형 컬럼의 경우, INLINE LENGTH는 최대 LOB 디스크립터 *크기*보다 작아서는 안됩니다.

컬럼을 변경해도 모든 컬럼의 총 바이트 수가 행 *크기* 한계를 초과해서는 안됩니다(SQLSTATE 54010).

행의 나머지 부분과 별도로 이미 저장된 데이터는 이 명령문으로도 인라인을 기본 테이블로 이동하지 않습니다. 구조화된 유형 컬럼의 변경된 인라인 길이를 사용하려면 컬럼의 인라인 길이를 변경한 후 지정한 테이블에 대해 REORG 명령을 호출하십시오. 기존 테이블의 XML 데이터 유형 컬럼의 변경된 인라인 길이를 사용하려면, UPDATE문으로 모든 행을 갱신해야 합니다. REORG 명령은 XML 문서의 행 스토리지에 영향을 주지 못합니다. LOB 데이터 유형 컬럼의 변경된 인라인 길이를 사용하려면, LONGLOBDATA 옵션이 있는 REORG 명령을 사용하거나 LOB 컬럼에 해당하는 UPDATE를 사용하십시오. 예를 들어, 다음과 같습니다.

```
UPDATE table-name SET lob-column = lob-column
   WHERE LENGTH(lob-column) <= chosen-inline-length - 4
```

*table-name*이 변경된 LOB 데이터 유형 컬럼의 인라인 길이가 있는 테이블인 경우, *lob-column*은 변경된 LOB 데이터 유형 컬럼이며 *chosen-inline-length*는 INLINE LENGTH에 선택된 새 값입니다.

SET NOT NULL

컬럼이 널(NULL) 값을 포함할 수 없도록 지정합니다. 테이블의 기존 행에 있는 이 컬럼의 값은 널(NULL) 값이 될 수 없습니다(SQLSTATE 23502). 컬

럼이 SET NULL의 DELETE 규칙을 가진 참조 제한조건의 외부 키에 지정된 경우 SET NOT NULL절은 허용되지 않으며, 외부 키에 널(NULL) 입력 가능한 다른 컬럼이 존재할 수 없습니다(SQLSTATE 42831). 컬럼의 이 속성을 변경하려면 추가로 테이블 액세스를 허용하기 전에 테이블을 재구성해야 합니다(SQLSTATE 57016). 이 조작은 테이블 데이터의 유효성 확인을 필요로 하기 때문에 테이블이 REORG 보류 상태에 있으면 해당 조작을 수행할 수 없다는 점에 유의하십시오(SQLSTATE 57016). 해당 테이블은 데이터 캡처 기능을 사용 가능하게 할 수 없습니다(SQLSTATE 42997).

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

컬럼이 테이블의 시간소인 컬럼이 되도록 지정합니다. 값은 삽입된 각 행의 컬럼 및 컬럼이 갱신된 행으로 생성됩니다. ROW CHANGE TIMESTAMP 컬럼에 대해 생성된 값은 해당 행에 대한 삽입 또는 갱신 시간에 해당되는 시간소인입니다. 다중 행이 단일 명령문으로 삽입되거나 갱신되면, ROW CHANGE TIMESTAMP 컬럼의 값은 각 행에 대해 다를 수 있습니다.

하나의 테이블은 하나의 ROW CHANGE TIMESTAMP 컬럼만 가질 수 있습니다(SQLSTATE 428C1). *data-type*이 지정된 경우 이는 TIMESTAMP 또는 TIMESTAMP(6)여야 합니다(SQLSTATE 42842). ROW CHANGE TIMESTAMP 컬럼은 DEFAULT절을 포함할 수 없습니다(SQLSTATE 42623). NOT NULL은 ROW CHANGE TIMESTAMP 컬럼에 대해서 지정되어서는 안됩니다(SQLSTATE 42831).

SET GENERATED ALWAYS 또는 GENERATED BY DEFAULT

데이터베이스 관리 프로그램이 컬럼 값을 생성하는 시기를 지정합니다. GENERATED BY DEFAULT는 값을 제공하지 않았거나 컬럼 지정에 DEFAULT 키워드를 사용하는 경우에만 값을 생성하도록 지정합니다. GENERATED ALWAYS는 데이터베이스 관리 프로그램이 항상 컬럼 값을 생성하도록 지정합니다. 컬럼은 이미 식별 컬럼을 기반으로 생성된 컬럼으로 정의되어 있어야 합니다. 즉, AS IDENTITY절을 사용하여 정의되어야 합니다(SQLSTATE 42837).

identity-alteration

컬럼의 식별 속성을 변경합니다. 컬럼은 식별 컬럼이어야 합니다.

SET INCREMENT BY *numeric-constant*

식별 컬럼의 연속 값 간격을 지정합니다. 식별 컬럼에 대해 생성될 다음 값은 증분이 적용된 마지막으로 지정된 값에서 결정됩니다. 컬럼은 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837).

이 값은 이 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 큰 정수 상수의 값을 초과하지 않고(SQLSTATE 42820), 소수점 오른쪽에 0이 아닌 자릿수를 갖지 않습니다(SQLSTATE 428FA).

이 값이 음수인 경우, 값은 ALTER문 이후에 내림차순이 됩니다. 이 값이 0이거나 양수인 경우, 값은 ALTER문 이후에 오름차순이 됩니다.

SET NO MINVALUE 또는 MINVALUE *numeric-constant*

내림차순 식별 컬럼이 값 생성을 순환하거나 중지하여 최소값을 지정합니다. 또는 오름차순 식별 컬럼이 최대값에 도달한 후 순환하여 최소값을 지정합니다. 컬럼은 지정된 테이블에 있어야 하며(SQLSTATE 42703), 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837).

NO MINVALUE

오름차순 시퀀스의 경우, 원래의 시작값입니다. 내림차순의 경우 값은 컬럼 데이터 유형의 최소값입니다.

MINVALUE *numeric-constant*

최소값인 숫자 상수를 지정합니다. 이 값은 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최대값보다 작거나 같아야 합니다(SQLSTATE 42815).

SET NO MAXVALUE 또는 MAXVALUE *numeric-constant*

오름차순 식별 컬럼이 값 생성을 순환하거나 중지하여 최대값을 지정합니다. 또는 내림차순 식별 컬럼이 최소값에 도달한 후 순환하여 최대값을 지정합니다. 컬럼은 지정된 테이블에 있어야 하며(SQLSTATE 42703), 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837).

NO MAXVALUE

오름차순의 경우, 값은 컬럼 데이터 유형의 최대값입니다. 내림차순 시퀀스의 경우, 원래의 시작값입니다.

MAXVALUE *numeric-constant*

최대값인 숫자 상수를 지정합니다. 이 값은 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최소값보다 크거나 같아야 합니다(SQLSTATE 42815).

SET NO CYCLE 또는 CYCLE

이 식별 컬럼이 최대값 또는 최소값을 생성한 이후 계속 값을 생성할지 여부를 지정합니다. 컬럼은 지정된 테이블에 있어야 하며(SQLSTATE 42703), 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837).

NO CYCLE

일단 최대값이나 최소값에 도달된 후에는 식별 컬럼에 대해 값이 생성되지 않도록 지정합니다.

CYCLE

최대값 또는 최소값에 도달된 이후 이 컬럼에 대해 값이 계속 생성되도록 지정합니다. 이 옵션을 사용하면, 오름차순 식별 컬럼이 최대값에 도달한 후 최소값을 생성합니다. 또는 내림차순이 최소값에 도달한 후 최대값을 생성합니다. 식별 컬럼에 대한 최대값과 최소값에 따라 순환에 사용되는 범위가 결정됩니다.

CYCLE이 효력을 가질 때 식별 컬럼에 대해 중복 값이 생성될 수 있습니다. 필수는 아니지만 고유한 값을 원할 경우, 식별 컬럼을 사용하여 단일 컬럼 고유 인덱스를 정의하면 고유성이 보장됩니다. 고유 인덱스가 이러한 식별 컬럼에 존재하고 고유하지 않은 값이 생성되면, 오류가 발생합니다(SQLSTATE 23505).

SET NO CACHE 또는 CACHE *integer-constant*

보다 빠른 액세스를 위해 메모리에 사전 할당된 값 일부를 저장할지 여부를 지정합니다. 이것은 성능 및 조정 옵션입니다. 컬럼은 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837).

NO CACHE

식별 컬럼의 값이 사전 할당되지 않도록 지정합니다. 데이터 공유 환경에서 식별 값을 요청 순서대로 *생성해야* 하는 경우에는 NO CACHE 옵션을 사용해야 합니다.

이 옵션을 지정하면 식별 컬럼의 값이 캐시에 저장되지 않습니다. 이 경우, 새 식별 값에 대한 모든 요청이 로그에 동시에 입출력됩니다.

CACHE *integer-constant*

사전 할당되고 메모리에 보존되는 식별 시퀀스 값의 수를 지정합니다. 식별 컬럼에 대한 값이 생성될 때 값을 사전할당하고 캐시에 저장하면 로그에 대한 동기 입출력이 줄어듭니다.

식별 컬럼에 새 값이 필요한데 캐시에 사용되지 않는 값이 없는 경우, 값을 할당하려면 로그에 대한 입출력을 기다려야 합니다. 그러나 식별 컬럼에 새 값이 필요한데 캐시에 사용되지 않는 값이 있는 경우, 로그에 대한 입출력이 수행되지 않으므로 해당 식별 값이 보다 빨리 할당될 수 있습니다.

정상 작업 중 또는 시스템 오류로 인해 데이터베이스 비활성화가 발생한 경우, 커미트 명령문에서 사용되지 않은 캐시된 모든 시퀀스 값이 유실됩니다. 즉, 이 값은 사용되지 않습니다. CACHE 옵션에 대해 지정된 값은 시스템 오류 시 유실될 수 있는 식별 컬럼에 대한 최대값입니다.

최소값은 2입니다(SQLSTATE 42815).

SET NO ORDER 또는 ORDER

식별 컬럼 값을 요청 순서대로 생성할지 여부를 지정합니다. 컬럼은 지정된 테이블에 있어야 하며(SQLSTATE 42703), 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837).

NO ORDER

식별 컬럼 값이 요청 순서대로 생성되지 않도록 지정합니다.

ORDER

식별 컬럼 값이 요청 순서대로 생성되도록 지정합니다.

RESTART 또는 RESTART WITH *numeric-constant*

식별 컬럼과 연관된 시퀀스의 상태를 재설정합니다. WITH *numeric-constant*를 지정하지 않으면, 식별 컬럼에 대한 시퀀스가 식별 컬럼이 원래 작성될 때 시작값으로 지정된 값으로, 다시 내재적 또는 명시적으로 지정되지 않습니다.

컬럼은 지정된 테이블에 있어야 하며(SQLSTATE 42703), 이미 IDENTITY 속성을 사용하여 정의되어 있어야 합니다(SQLSTATE 42837). RESTART는 원래의 START WITH 값을 변경하지 *않습니다*.

*numeric-constant*는 이 컬럼에 지정할 수 있는 양수 또는 음수의 정확한 숫자 상수일 수 있으며(SQLSTATE 42815), 소수점 오른쪽에 0이 아닌 자릿수를 갖지 않습니다(SQLSTATE 428FA). *numeric-constant*는 컬럼에 대한 다음 값으로 사용됩니다.

DROP IDENTITY

컬럼의 식별 속성을 삭제하여 컬럼을 단순 숫자 데이터 유형 컬럼으로 작성합니다. 컬럼이 식별 컬럼이 아니면 DROP IDENTITY를 사용할 수 없습니다(SQLSTATE 42837).

DROP EXPRESSION

컬럼의 생성된 표현식 속성을 삭제하여 비생성 컬럼으로 설정합니다. 컬럼이 생성된 표현식 컬럼이 아니면 DROP EXPRESSION을 사용할 수 없습니다(SQLSTATE 42837).

DROP DEFAULT

컬럼에 대한 현재 디폴트값을 삭제합니다. 지정된 컬럼에 디폴트값이 있어야 합니다(SQLSTATE 42837).

DROP NOT NULL

컬럼의 NOT NULL 속성을 삭제하여 컬럼이 널(NULL) 값을 가질 수 있도록 합니다. 이 절은 컬럼이 기본 키 또는 테이블의 고유 제한조건에 지정되어 있으면 사용할 수 없습니다(SQLSTATE 42831). 컬럼의 이 속성을 변경하려면

추가로 테이블 액세스를 허용하기 전에 테이블을 재구성해야 합니다(SQLSTATE 57016). 해당 테이블은 데이터 캡처 기능을 사용 가능하게 할 수 없습니다 (SQLSTATE 42997).

ADD SCOPE

아직 범위가 정의되어 있지 않은 기존의 참조 유형 컬럼에 범위를 추가합니다 (SQLSTATE 428DK). 변경할 테이블이 유형이 지정된 테이블인 경우, 이 컬럼은 슈퍼 테이블로부터 상속되어서는 안됩니다(SQLSTATE 428DJ).

typed-table-name

유형이 지정된 테이블의 이름입니다. *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-table-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하는지 확인하기 위해 *column-name*에 있는 기존 값을 점검하지 않습니다.

typed-view-name

유형이 지정된 뷰의 이름입니다. *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서, *S*는 *typed-view-name* 유형입니다(SQLSTATE 428DM). 값이 *typed-view-name*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *column-name*에 있는 기존 값들에 대해 점검을 하지 않습니다.

COMPRESS

이 컬럼의 디폴트값을 보다 효율적으로 저장할지 여부를 지정합니다.

SYSTEM DEFAULT

시스템 디폴트값(특정 값을 지정하지 않았을 때 데이터 유형에 사용되는 디폴트값)을 최소한의 스페이스를 사용하여 저장하도록 지정합니다. VALUE COMPRESSION 속성을 활성화하여 테이블이 설정되어 있지 않으면 경고가 리턴되고(SQLSTATE 01648) 시스템 디폴트값이 최소한의 스페이스를 사용하여 저장되지 않습니다.

이와 같은 방식으로 시스템 디폴트값이 저장되도록 하면 추가 점검이 수행되기 때문에 컬럼에 대해 삽입 및 갱신 조작을 수행하는 동안 성능이 약간 저하될 수 있습니다.

컬럼에 있는 기존의 데이터는 변경되지 않습니다. 오프라인에서 테이블을 재구성하여 기존 데이터가 최소한의 스페이스를 사용하여 시스템 디폴트값을 저장하는 방법을 고려해 보십시오.

OFF

시스템 디폴트값이 일반 값으로 컬럼에 저장되도록 지정합니다. 컬럼에 있는 기존의 데이터는 변경되지 않습니다. 기존의 데이터를 변경하기 위해서는 오프라인 재구성이 권장됩니다.

기본 데이터 유형은 DATE, TIME 또는 TIMESTAMP일 수 없습니다 (SQLSTATE 42842). 기본 데이터 유형이 가변 길이 문자열일 경우에는 이 절

이 무시됩니다. VALUE COMPRESSION을 사용하여 테이블을 설정한 경우에는 길이가 0인 문자열 값이 자동으로 압축됩니다.

변경할 테이블이 유형이 지정된 테이블인 경우, 이 컬럼은 슈퍼 테이블로부터 상속되어서는 안됩니다(SQLSTATE 428DJ).

SECURED WITH *security-label-name*
테이블과 연결된 보안 규정에 대해 존재하는 보안 레이블을 식별합니다. 이름이 규정되어 있지 않아야 합니다(SQLSTATE 42601). 테이블에는 테이블과 연관된 보안 규정이 있어야 합니다(SQLSTATE 55064).

DROP COLUMN SECURITY
컬럼을 변경하여 비보호 컬럼으로 만듭니다.

RENAME COLUMN *source-column-name* **TO** *target-column-name*
*source-column-name*에서 지정된 컬럼의 이름을 *target-column-name*에서 지정한 이름으로 바꿉니다. **auto_reval** 데이터베이스 구성 매개변수가 DISABLED로 설정된 경우, ALTER TABLE 명령문의 RENAME COLUMN 옵션은 REVALIDATION IMMEDIATE 시맨틱의 제어를 받는 경우와 동일하게 작동합니다.

source-column-name
이름을 변경할 컬럼의 이름을 지정합니다. *source-column-name*은 테이블의 기존 컬럼을 식별해야 합니다(SQLSTATE 42703). 이름을 규정해서는 안됩니다. 이름은 동일한 ALTER TABLE문에서 추가, 변경 또는 삭제되는 컬럼을 식별해서는 안됩니다(SQLSTATE 42711).

target-column-name
컬럼의 새 이름입니다. 이름을 규정해서는 안됩니다. 테이블의 기존 컬럼 이름은 사용하지 마십시오(SQLSTATE 42711).

DROP PRIMARY KEY
기본 키 정의와 이 기본 키에 종속되는 모든 참조 제한조건을 삭제합니다. 테이블에는 기본 키가 있어야 합니다(SQLSTATE 42888).

DROP FOREIGN KEY *constraint-name*
참조 제한조건의 *constraint-name*을 삭제합니다. *constraint-name*은 참조 제한조건을 식별해야 합니다(SQLSTATE 42704). 참조 제한조건 삭제에 대한 자세한 내용은 주를 참조하십시오.

DROP UNIQUE *constraint-name*
고유성 *constraint-name*의 정의와 이 고유 제한조건에 종속되는 모든 참조 제한조건을 삭제합니다. *constraint-name*은 참조 제한조건을 식별해야 합니다(SQLSTATE 42704). 고유 제한조건 삭제에 대한 자세한 내용은 주를 참조하십시오.

DROP CHECK *constraint-name*

점검 제한조건의 *constraint-name*을 삭제합니다. *constraint-name*은 테이블에 정의된 기존의 점검 제한조건을 식별해야 합니다(SQLSTATE 42704).

DROP CONSTRAINT *constraint-name*

제한조건의 *constraint-name*을 삭제합니다. *constraint-name*은 테이블에 정의된 기존의 점검 제한조건, 참조 제한조건, 기본 키 또는 고유 제한조건을 식별해야 합니다(SQLSTATE 42704). 제한조건 삭제에 대한 자세한 내용은 주를 참조하십시오.

DROP COLUMN

테이블에서 식별된 컬럼을 삭제합니다. 테이블은 유형이 지정된 테이블이어서는 안됩니다(SQLSTATE 428DH). 해당 테이블은 데이터 캡처 기능을 사용 가능하게 할 수 없습니다(SQLSTATE 42997). 컬럼이 삭제된 경우 테이블에서 update, insert 또는 delete 조작 또는 인덱스 스캔을 수행하기 전에 테이블을 재구성해야 합니다(SQLSTATE 57016). XML 컬럼은 테이블의 다른 모든 XML 컬럼이 동시에 삭제된 경우에만 삭제할 수 있습니다.

column-name

삭제될 컬럼을 식별합니다. 컬럼 이름을 규정해서는 안됩니다. 해당 이름은 지정된 테이블의 컬럼을 식별해야 합니다(SQLSTATE 42703). 이름은 테이블의 단 하나뿐인 컬럼을 식별해서는 안됩니다(SQLSTATE 42814). 이름은 해당 테이블의 분산 키, 테이블 파티션 키 또는 구성 차원의 일부인 컬럼을 식별해서는 안됩니다(SQLSTATE 42997).

CASCADE

삭제 중인 컬럼에 종속되는 뷰, 인덱스, 트리거, SQL 함수, 제한조건 또는 전역 변수도 삭제되도록 지정하거나 또는 해당 컬럼을 포함한 테이블에 종속되는 분석 가능한 XSROBJECT가 분석 작동 불능이 되도록 지정합니다. 트리거는 UPDATE OF 컬럼 목록 또는 트리거 조치에서 참조되는 경우 해당 컬럼에 종속됩니다. 분석 가능한 XSROBJECT는 테이블에 대한 속성 또는 XML 요소의 맵핑을 포함하는 경우 테이블에 종속됩니다. SQL 함수 또는 전역 변수가 다른 데이터베이스 오브젝트에 종속된 경우, CASCADE 옵션으로 함수를 삭제하지 못할 수 있습니다. CASCADE는 디폴트값입니다.

RESTRICT

뷰, 인덱스, 트리거, 제한조건 또는 전역 변수가 컬럼에 종속되거나 분석 가능한 XSROBJECT가 컬럼을 포함하는 테이블에 종속되는 경우 해당 컬럼을 삭제할 수 없도록 지정합니다(SQLSTATE 42893). 트리거는 UPDATE OF 컬럼 목록 또는 트리거 조치에서 참조되는 경우 해당 컬럼에 종속됩니다. 분석 가능한 XSROBJECT는 테이블에 대한 속성 또는 XML 요소의 맵핑을 포함하는 경우 테이블에 종속됩니다. 발견된 첫 번째 종속 오브젝트는 관리 로그에서 식별됩니다.

표 15. 컬럼 삭제의 연쇄 효과

조작	RgESTRICT 효과	CASCADE 효과
뷰 또는 트리거가 참조하는 컬럼 삭제	컬럼 삭제를 할 수 없습니다.	오브젝트 및 그 오브젝트에 종속된 모든 오브젝트를 삭제합니다.
인덱스의 키에서 참조되는 컬럼 삭제	인덱스에서 참조되는 모든 컬럼이 동일한 ALTER TABLE문에서 삭제되면 해당 인덱스를 삭제할 수 있습니다. 그렇지 않으면 컬럼 삭제를 할 수 없습니다.	해당 인덱스가 삭제됩니다.
고유 제한조건에서 참조되는 컬럼 삭제	고유 제한조건에서 참조되는 모든 컬럼이 동일한 ALTER TABLE문에서 삭제되고 참조 제한조건이 고유 제한조건을 참조하지 않으면, 컬럼 및 제한조건이 삭제됩니다. (제한조건을 만족시키기 위해 사용된 인덱스도 삭제됩니다.) 그렇지 않으면 컬럼 삭제를 할 수 없습니다.	고유 제한조건 및 그 고유 제한조건을 참조하는 참조 제한조건이 삭제됩니다. (제한조건이 사용한 모든 인덱스도 삭제됩니다.)
참조 제한조건에서 참조되는 컬럼 삭제	참조 제한조건에서 참조되는 모든 컬럼이 동일한 ALTER TABLE문에서 삭제되면 해당 인덱스 및 제한조건을 삭제할 수 있습니다. 그렇지 않으면 컬럼 삭제를 할 수 없습니다.	해당 참조 제한조건이 삭제됩니다.
삭제되지 않는 시스템 생성 컬럼이 참조하는 컬럼을 삭제합니다.	컬럼 삭제를 할 수 없습니다.	컬럼 삭제를 할 수 없습니다.
점검 제한조건에서 참조되는 컬럼 삭제	컬럼 삭제를 할 수 없습니다.	해당 점검 제한조건이 삭제됩니다.
분석 가능한 XSROBJECT에서 참조되는 컬럼 삭제	컬럼 삭제를 할 수 없습니다.	XSROBJECT는 분석 작동 불능으로 표시됩니다. XSROBJECT를 다시 사용 가능하게 하려면 맵핑의 재조정이 필요하며, XSROBJECT에 대해 ALTER XSROBJECT ENABLE DECOMPOSITION문을 발행하십시오.
전역 변수의 디폴트 표현식에서 참조되는 컬럼 삭제	컬럼 삭제를 할 수 없습니다.	다른 오브젝트가 있어 전역 변수의 삭제가 승인 불가되지 않는 한 전역 변수는 삭제되며, 전역 변수에 따라 연쇄를 허용하지 않습니다.

DROP RESTRICT ON DROP

테이블 및 테이블이 포함된 테이블 스페이스 삭제에 대한 제한이 있는 경우 이를 제거합니다.

DROP DISTRIBUTION

테이블에 대한 분산 정의를 삭제합니다. 테이블에는 분산 정의가 있어야 합니다 (SQLSTATE 428FT). 테이블에 대한 테이블 스페이스는 단일 파티션 데이터베이스 파티션 그룹에 정의되어야 합니다.

DROP MATERIALIZED QUERY

더 이상 구체화된 쿼리 테이블로 간주되지 않도록 구체화된 쿼리 테이블을 변경합니다. *table-name*으로 지정된 테이블은 복제되지 않는 구체화된 쿼리 테이블로 정의해야 합니다(SQLSTATE 428EW). *table-name*의 컬럼에 대한 정의는 변경되지 않지만, 해당 테이블은 쿼리 최적화에 더 이상 사용될 수 없으며 REFRESH TABLE 문을 더 이상 사용할 수 없습니다.

DATA CAPTURE

데이터 복제에 대한 추가 정보를 로그에 기록할지 여부를 나타냅니다.

테이블이 유형이 지정된 테이블이라면, 이 옵션이 지원되지 않습니다(루트 테이블의 경우에는 SQLSTATE 428DH, 기타 서브테이블의 경우에는 428DR).

NONE

추가 정보가 기록되지 않음을 나타냅니다.

CHANGES

이 테이블에 대한 SQL 변경사항에 대한 추가 정보가 로그에 기록됨을 나타냅니다. 이 옵션은 이 테이블이 복제되며 캡처 프로그램을 사용하여 이 테이블에 대한 변경사항을 로그에서 캡처할 경우에 필요합니다.

테이블이 카탈로그 파티션(카탈로그 파티션 이외의 다른 데이터베이스 파티션을 가진 다중 파티션 데이터베이스 파티션 그룹 또는 데이터베이스 파티션 그룹) 이외의 다른 데이터베이스 파티션에서 데이터를 허용하도록 정의된 경우는 이 옵션이 지원되지 않습니다(SQLSTATE 42997).

테이블의 (내재적 또는 명시적) 스키마 이름이 18바이트보다 크면, 이 옵션이 지원되지 않습니다(SQLSTATE 42997).

INCLUDE LONGVAR COLUMNS

데이터 복제 유틸리티가 LONG VARCHAR 또는 LONG VARGRAPHIC 컬럼에 대한 변경사항을 캡처할 수 있도록 합니다. 이 절은 LONG VARCHAR 또는 LONG VARGRAPHIC 컬럼이 없는 테이블에 대해 지정할 수 있습니다. 왜냐하면 이러한 컬럼을 포함하도록 테이블을 변경할 수 있기 때문입니다.

ACTIVATE NOT LOGGED INITIALLY

현재 작업 단위에 대한 테이블의 NOT LOGGED INITIALLY 속성을 활성화합니다.

이 명령문에 의해 테이블이 교체된 후 동일한 작업 단위에서 INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX 또는 ALTER TABLE에 의한 테이블 변경사항은 기록되지 않습니다. NOT LOGGED INITIALLY 속성이 활성화된 ALTER문에 의한 시스템 카탈로그 변경사항은 기록되지 않습니다. 동일한 작업 단위 내에서 이후의 시스템 카탈로그 정보 변경사항은 기록됩니다.

현재 작업 단위가 완료되면 NOT LOGGED INITIALLY 속성은 비활성화되고, 이후의 작업 단위에서 테이블에서 수행되는 모든 조작이 기록됩니다.

데이터를 삽입하면서 카탈로그 테이블에 대한 잠금을 방지하기 위해 이 기능을 사용하는 경우, ALTER TABLE문에서 이 절만 지정해야 합니다. ALTER TABLE문에서 다른 절을 사용하면 카탈로그가 잠깁니다. ALTER TABLE문에 대해 다른 절을 지정하지 않은 경우, 시스템 카탈로그 테이블에서는 SHARE 잠금만 사용할 수 있습니다. 이렇게 하면 이 명령문이 실행되어 실행된 작업 단위가 종료될 때까지의 시간 동안 동시성 충돌이 일어날 가능성이 현저히 줄어들게 됩니다.

테이블이 유형이 지정된 테이블이라면, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

NOT LOGGED INITIALLY 속성에 대한 자세한 내용은 『CREATE TABLE』을 참조하십시오.

주: NOT LOGGED INITIALLY 속성이 활성화된 테이블에 대하여 비로그가 수행되고, 명령문이 실패하여 롤백이 발생하거나 ROLLBACK TO SAVEPOINT가 실행될 경우에는 전체 작업 단위가 롤백됩니다(SQL1476N). 또한 롤백이 발생한 후에는 NOT LOGGED INITIALLY 속성이 활성화된 테이블은 액세스 불가로 표시되어 삭제만 가능합니다. 따라서 NOT LOGGED INITIALLY 속성이 사용 중인 작업 단위 내에서 오류가 발생할 가능성은 최소로 해야 합니다.

WITH EMPTY TABLE

테이블에 현재 있는 모든 데이터가 제거됩니다. 일단 데이터가 제거되면 RESTORE 기능으로만 복구할 수 있습니다. 이 ALTER문이 발행된 작업 단위가 롤백되는 경우, 테이블 데이터는 원래 상태로 돌아오지 않습니다.

이 조치가 요청되면 영향을 받는 테이블에 정의된 DELETE 트리거는 실행되지 않습니다. 테이블에 있는 인덱스도 삭제됩니다.

접속된 데이터 파티션이 있는 파티션된 테이블은 비울 수가 없습니다 (SQLSTATE 42928).

PCTFREE *integer*

로드 또는 테이블 재구성 작업시 여유 공간으로 남게 될 각 페이지의 백분율을 지정합니다. 제한없이 각 페이지의 첫 번째 행이 추가됩니다. 페이지에 행이 추가될 때 해당 페이지에 적어도 *integer* 퍼센트의 여유 공간이 남게 됩니다. PCTFREE 값은 로드 및 테이블 재구성 유틸리티에 의해서만 고려됩니다. *integer* 값의 범위는 0에서 99까지입니다. 시스템 카탈로그(SYSCAT.TABLES)에서 -1의 PCTFREE 값은 디폴트값으로 해석됩니다. 테이블 페이지에 대한 디폴트 PCTFREE 값은 0입니다. 유형이 지정된 테이블의 경우, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

LOCKSIZE

테이블이 액세스될 때 사용되는 잠금 크기(세분화도)를 나타냅니다. 테이블 정의에

서 이 옵션을 사용하면 정상적인 잠금 에스컬레이션이 발생합니다. 테이블이 유형이 지정된 테이블이라면, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

ROW

행 잠금 사용을 나타냅니다. 이는 테이블이 작성될 때의 디폴트 잠금 *크기*입니다.

BLOCKINSERT

삽입 조작 중 블록 잠금을 표시합니다. 삽입 전에 적절한 배타적 잠금이 블록에 확보되고 행 잠금이 삽입된 행에 이루어지지 않음을 의미합니다. 별도의 트랜잭션이 테이블의 별도 셀에 삽입되는 경우 유용한 옵션입니다. 동일한 셀로의 트랜잭션 삽입을 동시에 진행할 수 있지만 구별 블록으로 삽입될 것이고 이는 더 많은 블록이 필요한 경우 셀 *크기*에 영향을 줄 수 있습니다. 이 옵션은 MDC 테이블에만 사용할 수 있습니다(SQLSTATE 42613).

TABLE

테이블 잠금 사용을 나타냅니다. 즉, 테이블에 대한 적절한 공유 또는 배타적 잠금은 확보되지만 의도 잠금(의도 없음 제외)은 사용되지 않습니다. 파티션된 테이블의 경우 이 잠금 전략은 액세스되는 모든 데이터 파티션에 대한 테이블 잠금 및 데이터 파티션 잠금에 적용됩니다. 이 값을 사용하면 확보되어야 할 잠금 수를 제한하여 쿼리 성능을 향상시킬 수 있습니다. 그러나 전체 테이블에서 모든 잠금이 유지되므로 동시성도 감소됩니다.

APPEND

데이터 페이지에서 여유 공간이 있는 곳에 데이터를 저장할 것인지 데이터를 테이블 데이터 끝부분에 추가할 것인지 여부를 나타냅니다. 테이블이 유형이 지정된 테이블이라면, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

ON

테이블 데이터가 추가된다는 것과 페이지의 여유 공간에 대한 정보가 보존되지 않을 것임을 나타냅니다. 테이블에는 클러스터 인덱스가 없어야 합니다(SQLSTATE 428CA).

OFF

테이블 데이터가 여유 공간에 저장될 것임을 나타냅니다. 이는 테이블이 작성될 때 디폴트값입니다.

사용 가능한 빈 스페이스에 대한 정보가 정확하지 않아 삽입시 성능이 저하될 수 있으므로 APPEND OFF를 설정한 후에는 테이블을 재구성해야 합니다.

VOLATILE CARDINALITY 또는 NOT VOLATILE CARDINALITY

테이블 *table-name*의 카디널리티가 런타임시 *크게* 다를 수 있는지 여부를 옵티마

이저에 나타냅니다. 유동성은 테이블 자체가 아닌 테이블에 있는 행 수에 적용됩니다. CARDINALITY는 선택적 키워드입니다. 디폴트값은 NOT VOLATILE입니다.

VOLATILE

런타임시 테이블 *table-name*의 카디낼리티가 빈 것에서 아주 큰 것에 이르기까지 크게 다를 수 있도록 지정합니다. 테이블에 액세스하려면 옵티마이저는 해당 인덱스가 인덱스만이거나(참조된 모든 컬럼이 인덱스에 있음) 해당 인덱스가 인덱스 스캔에서 술어를 적용할 수 있다면, 통계에 관계없이 테이블 스캔보다는 인덱스 스캔을 사용합니다. 리스트 프리페치 액세스 메소드는 테이블 액세스에 사용되지 않습니다. 테이블이 유형이 지정된 테이블이라면, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

NOT VOLATILE

*table-name*의 카디낼리티가 일시적이 아니도록 지정합니다. 이 테이블에 대한 액세스 플랜은 계속 기존 통계 및 현재 최적화 레벨을 기반으로 합니다.

COMPRESS

데이터 압축을 테이블의 행에 적용할지 여부를 지정합니다.

YES

데이터 행 압축을 사용 가능하도록 지정합니다. 테이블에 대한 삽입 및 갱신 조작은 압축의 영향을 받습니다. 테이블에 대한 압축 사전이 없으면, 압축 사전이 자동으로 작성되며 테이블이 데이터로 채워진 다음 압축의 영향을 받습니다. 테이블의 기존 압축 사전이 있으면 압축이 다시 활성화되어 이 사전을 사용하며 행은 압축의 영향을 받습니다. XML 스토리지 오브젝트의 데이터에도 적용됩니다. XML 스토리지 오브젝트에 충분한 데이터가 있는 경우, 압축 사전이 자동으로 작성되며 XML 문서가 압축의 영향을 받습니다. CREATE INDEX 문에서 명시적으로 사용하지 않도록 설정하지 않는 한 새 인덱스에 대해 인덱스 압축이 사용됩니다. 기존 인덱스는 ALTER INDEX 명령문을 사용하여 압축될 수 있습니다.

NO

데이터 행 압축을 사용 가능하도록 지정합니다. 테이블에 대한 삽입 및 갱신 조작은 더 이상 압축에 영향을 받지 않습니다. 압축 형식으로 된 테이블의 모든 행은 갱신시 압축되지 않은 형식으로 변환될 때까지 압축 형식으로 남습니다. 테이블의 non-inplace 재구성은 압축된 모든 행을 압축 해제합니다. 압축 사전은 테이블 다시 초기화 또는 절단 중에 버려집니다(예: 바꾸기 조작). CREATE INDEX문에서 명시적으로 사용하도록 설정하지 않는 한 새 인덱스에 대해 인덱스 압축이 사용되지 않습니다. ALTER INDEX 명령문을 사용하면 기존 인덱스에 대한 인덱스 압축이 명시적으로 사용 불가능할 수 있습니다.

VALUE COMPRESSION

이것은 사용할 행 형식을 결정합니다. 각 데이터 유형은 사용되는 행 형식에 따라 서로 다른 바이트 수를 갖습니다. 자세한 정보는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 갱신 조작을 수행하면 기존 행이 새 행 형식으로 변경됩니다. 기존 행에 대한 갱신 조작의 성능을 향상시키려면 오프라인에서 테이블을 재구성하는 것이 좋습니다. 이로 인해 테이블 스페이스에 여유가 생길 수 있습니다. 『데이터 유형별 컬럼의 바이트 수』(CREATE TABLE 참조)라는 테이블에서 해당 컬럼을 사용하여 계산된 행 *크기*가 『각 테이블 스페이스 페이지 *크기*에서 행 *크기* 및 컬럼의 수에 대한 한계』라는 테이블에 지정된 행 *크기* 한계를 넘어서는 경우 오류가 리턴됩니다(SQLSTATE 54010). 테이블이 유형이 지정된 테이블이라면, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

ACTIVATE

널(NULL) 값은 3 바이트를 사용하여 저장됩니다. 이것은 CHAR(1)을 제외한 모든 데이터 유형의 컬럼에 대해 VALUE COMPRESSION이 활성화되지 않았을 때보다 작거나 같은 스페이스입니다. 컬럼이 널(NULL) 입력 가능으로 정의되었는지 여부가 행 *크기* 계산에 영향을 주지 않습니다. 데이터 유형이 VARCHAR, VARGRAPHIC, LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB 또는 BLOB인 컬럼의 0 길이 데이터 값은 VALUE COMPRESSION이 활성화되지 않았을 때 필요한 스토리지보다 작은 2바이트만 사용하여 저장됩니다. 컬럼을 COMPRESS SYSTEM DEFAULT 옵션으로 정의하면 컬럼의 시스템 디폴트 값이 전체 스토리지의 3바이트를 사용하여 저장될 수 있습니다. 이를 지원하는 데 사용되는 행 형식은 각 데이터 유형에 대한 바이트 수를 결정하고 널(NULL), 0길이 값 또는 시스템 디폴트 값으로 갱신하는 동안 데이터를 단편화합니다.

DEACTIVATE

추후 갱신을 위해 개별적으로 설정된 스페이스를 사용하여 널(NULL) 값이 저장됩니다. 가변 길이 컬럼에 대해서는 이 스페이스가 개별적으로 설정되지 않습니다. 뿐만 아니라 이 옵션은 컬럼에 대한 시스템 디폴트값이 효율적으로 저장되는 것을 지원하지 않습니다. 따라서 컬럼에 COMPRESS SYSTEM DEFAULT 속성이 이미 설정되어 있으면 경고가 리턴됩니다(SQLSTATE 01648).

LOG INDEX BUILD

이 테이블에 대한 인덱스 재구성, 재작성 또는 작성 조작 중에 수행되는 로깅 레벨을 지정합니다.

NULL

logindexbuild 데이터베이스 구성 매개변수의 값을 사용하여 인덱스 빌드 조작을 완전하게 로그할지 여부를 판별하도록 지정합니다. 이는 테이블이 작성될 때 디폴트값입니다.

OFF

이 테이블에 대한 모든 인덱스 빌드 조작이 최소한으로 로그되도록 지정합니다. 이 값은 **logindexbuild** 데이터베이스 구성 매개변수의 설정값을 겹쳐씁니다.

ON

이 테이블에 대한 모든 인덱스 빌드 조작이 완전하게 로그되도록 지정합니다. 이 값은 **logindexbuild** 데이터베이스 구성 매개변수의 설정값을 겹쳐씁니다.

규칙

- 테이블에 정의된 기본 키 또는 고유 키 제한조건이 한 가지일 경우 분산 키의 수퍼세트여야 합니다(SQLSTATE 42997).
- 기본 키 또는 고유 키는 차원의 서브 세트가 될 수 없습니다(SQLSTATE 429BE).
- 단일 ALTER TABLE문에 있는 하나의 ADD, ALTER 또는 DROP COLUMN 절에서 컬럼은 참조만 될 수 있습니다(SQLSTATE 42711).
- 테이블에 종속된 구체화된 쿼리 테이블이 있으면 컬럼 길이 또는 데이터 유형을 변경할 수 없고 컬럼을 삭제할 수도 없습니다(SQLSTATE 42997).
- 각각 4000과 2000을 초과하도록 변경된 VARCHAR 및 VARGRAPHIC 컬럼은 SYSFUN 스키마에서 함수의 입력 매개변수로 사용해서는 안 됩니다(SQLSTATE 22001).
- 테이블에 종속된 쿼리 최적화 사용 가능 뷰가 있으면 테이블의 컬럼 길이는 변경될 수 없습니다(SQLSTATE 42997).
- 다음 조작이 수행되기 전에 OFF 옵션과 함께 SET INTEGRITY문을 사용하여 테이블이 무결성 설정 보류 상태가 되도록 해야 합니다(SQLSTATE 55019).
  - 생성 표현식을 사용하여 컬럼 추가
  - 컬럼의 생성된 표현식 변경
  - 컬럼에 생성된 표현식이 포함되도록 변경
- 기존의 컬럼은 변경되어 유형 DB2SECURITYLABEL이 될 수 없습니다(SQLSTATE 42837).
- 테이블과 관련된 보안 규정이 테이블에 없는 경우 유형 DB2SECURITYLABEL 컬럼을 정의할 수 없습니다(SQLSTATE 55064).
- 유형 DB2SECURITYLABEL 컬럼은 변경 또는 삭제될 수 없습니다(SQLSTATE 42817).
- 테이블에 종속된 MQT가 있는 경우 테이블을 보호되게 표시하기 위한 ALTER TABLE 조작을 할 수 없습니다(SQLSTATE 55067).

- 소스 테이블 및 목표 테이블이 동일한 보안 규정을 사용하여 보호되지 않고 두 테이블에 동일한 행 보안 레이블이 없으며 동일한 보호된 컬럼 세트가 없는 경우, 파티션을 보호된 파티션된 테이블에 접속할 수 없습니다(SQLSTATE 428GE).
- 생성된 컬럼이 테이블 파티션 키에서 참조되는 경우 생성된 컬럼 표현식은 변경될 수 없습니다(SQLSTATE 42837).
- *isolation-clause*는 *materialized-query-definition*의 *fullselect*에 지정될 수 없습니다(SQLSTATE 42601).

주

- *REORG 권장 조작*은 ALTER TABLE문이 만든 변경이 데이터의 행 형식에 영향을 주는 경우 발생합니다. 이 조작이 발생하면 테이블의 다음 조작 대부분이 테이블 재구성 조작이 완료될 때까지 제한을 받습니다. 재구성이 필요한 상황이 오기 전에 이 유형의 ALTER TABLE문을 하나의 테이블에 대해 3개까지 실행할 수 있습니다(SQLSTATE 57016). REORG 권장 조작을 이루는 복수의 조작은 단일 ALTER TABLE문(컬럼당 1개)의 일부로 작성될 수 있으며 단일 REORG 권장 조작으로 간주됩니다. 예를 들어 단일 ALTER TABLE문에서 2개의 컬럼을 삭제하는 것은 2개의 REORG 권장 조작으로 간주되지 않습니다. 그러나 2개의 개별 ALTER TABLE문에서 2개의 컬럼을 삭제하는 것은 REORG 권장 옵션을 포함하는 2개의 명령문으로 간주됩니다.
- REORG 권장 조작이 발생한 후 다음의 테이블 조작을 할 수 있습니다.
  - 행 데이터 유효성 확인이 필요하지 않은 경우의 ALTER TABLE이 해당됩니다. 그러나 다음 조작은 허용되지 않습니다(SQLSTATE 57007).
    - ADD CHECK CONSTRAINT
    - ADD REFERENTIAL CONSTRAINT
    - ADD UNIQUE CONSTRAINT
    - ALTER COLUMN SET NOT NULL
  - DROP TABLE
  - RENAME TABLE
  - REORG TABLE
  - TRUNCATE TABLE
  - 테이블 데이터의 테이블 스캔 액세스
- 테이블을 구체화된 쿼리 테이블로 변경하면 테이블이 무결성 설정 보류 상태에 놓입니다. 테이블이 REFRESH IMMEDIATE로 정의될 경우, 테이블은 INSERT, DELETE 또는 UPDATE 명령이 fullselect에 의해 참조되는 테이블에서 호출되기 전에 무결성 설정 보류 상태에서 벗어나야 합니다. 테이블은 IMMEDIATE CHECKED 옵션과 함께 REFRESH TABLE 또는 SET INTEGRITY를 사용하여 무결성 설정 보류 상태를 벗어나서, fullselect를 기초로 테이블의 데이터를 완전하게

새로 고칠 수 있습니다. 테이블의 데이터가 fullselect의 결과를 정확하게 반영할 경우, SET INTEGRITY의 IMMEDIATE UNCHECKED 옵션을 사용하여 테이블 상태를 무결성 설정 보류 상태에서 벗어나게 할 수 있습니다.

- 테이블을 REFRESH IMMEDIATE 구체화된 쿼리 테이블로 변경하면 fullselect에 의해 참조되는 테이블에서 INSERT, DELETE 또는 UPDATE가 사용되는 패키지가 무효화됩니다.
- 테이블을 구체화된 쿼리 테이블에서 일반 테이블로 변경하면, 그 테이블에 종속되는 패키지가 무효화됩니다.
- 테이블을 MAINTAINED BY FEDERATED_TOOL 구체화된 쿼리 테이블에서 일반 테이블로 변경해도 복제 도구의 서브스크립션 설정은 변경되지 않습니다. MAINTAINED BY SYSTEM 구체화된 쿼리 테이블에 대한 이후 변경사항으로 인해 복제 도구가 실패하게 되므로 MAINTAINED BY FEDERATED_TOOL 구체화된 쿼리 테이블을 변경할 때 서브스크립션 설정도 변경해야 합니다.
- 지연된 구체화된 쿼리 테이블이 스테이징 테이블과 연관되어 있을 경우 구체화된 쿼리 테이블이 일반 테이블로 변경되면 스테이징 테이블이 삭제됩니다.
- ADD 컬럼 절은 모든 다른 절보다 먼저 처리됩니다. 다른 절은 지정된 순서대로 처리됩니다.
- 테이블 변경 조작을 통해 추가되는 컬럼은 테이블의 기존 뷰에 자동으로 추가되지 않습니다.
- 데이터 파티션을 파티션된 테이블에 추가 또는 접속하거나 데이터 파티션을 파티션된 테이블에서 접속 해제하면 해당 테이블에 종속된 패키지가 무효화됩니다.
- 테이블에 대한 파티션을 삭제하려면 테이블이 삭제되고 다시 작성되야 합니다.
- 테이블에 대한 구성을 삭제하려면 테이블이 삭제되고 다시 작성되야 합니다.
- 고유 제한조건이나 기본 키 제한조건에 대해 인덱스가 자동으로 작성되는 경우, 데이터베이스 관리 프로그램은 테이블의 스키마 이름과 일치하는 스키마 이름과 더불어 지정된 제한조건 이름을 인덱스 이름으로 사용하려 합니다. 이름이 기존의 인덱스 이름과 일치하거나 제한조건에 대한 이름을 지정하지 않은 경우, SYSIBM 스키마에 인덱스가 작성되는데, 시스템에서 생성하는 이름은 ″SQL″과 그 뒤에 오는 시각 기반 함수에 의해 생성된 15자리 숫자 문자로 이루어집니다.
- 접속된 데이터 파티션이 있는 파티션된 테이블에서 파티션되지 않은 인덱스가 작성된 경우, 인덱스는 접속된 데이터 파티션에서 데이터를 포함하지 않습니다. SET INTEGRITY문을 사용하여 모든 접속된 데이터 파티션에 대한 모든 인덱스를 유지보수하십시오.
- 접속된 파티션이 있는 파티션된 인덱스를 작성하는 경우(SYSDATAPARTITIONS에서 'A'의 STATUS), 각 접속 파티션에 대한 인덱스 파티션도 작성됩니다. 파티션된 인덱스가 고유하게 작성되었거나 REJECT INVALID VALUES로 작성된 XML

인덱스이면, 접속된 파티션에 위반사항(고유 인덱스에 중복되거나 XML 인덱스에 대해 올바르지 않은 값인 경우)이 있는 경우 인덱스 작성이 실패할 수 있습니다.

- 테이블 T에 대한 DELETE 조작에 수반될 수도 있는 테이블을 T에 *delete-connected* 된다고 말합니다. 그러므로 T에 종속되거나 T 연쇄로부터 삭제되는 테이블에 종속되는 테이블은 T에 연속 삭제됩니다.
- 레코드가 패키지 내의 명령문에 의해 직접적으로 삽입되거나 명령문 중 하나를 대신해서 패키지에 의해 실행되는 제한조건 또는 트리거를 통해 간접적으로 T에 삽입(갱신/삭제)되는 경우, 패키지에서는 테이블 T에 대한 삽입(갱신/삭제)을 수행할 수 있습니다. 마찬가지로, 컬럼이 패키지 내의 명령문에 의해 직접 수정되거나 패키지 명령문 중 하나를 대신하여 패키지에 의해 실행되는 제한조건 또는 트리거를 통해 간접적으로 수정되는 경우, 패키지에서는 컬럼에 대한 갱신을 수행할 수 있습니다.
- 페더레이티드 시스템에서 투명한 DDL을 사용하여 작성된 리모트 기본 테이블을 변경할 수 있습니다. 그러나 투명한 DDL은 수정과 관련한 몇 가지 제한사항이 있습니다.
  - 리모트 기본 테이블은 새 컬럼을 추가하거나 기본 키를 지정할 때만 변경할 수 있습니다.
  - 투명 DDL이 지원하는 특정 절은 다음을 포함합니다.
    - ADD COLUMN *column-definition*
    - *column-options*절의 NOT NULL 및 PRIMARY KEY
    - ADD *unique-constraint*(PRIMARY KEY만 해당)
  - 리모트 기본 테이블에 있는 기존 컬럼에 주석을 지정할 수 없습니다.
  - 리모트 기본 테이블의 기존 기본 키는 변경하거나 삭제할 수 없습니다.
  - 리모트 기본 테이블을 변경하면, 리모트 기본 테이블과 연관된 별칭에 종속된 모든 패키지가 무효화됩니다.
  - 리모트 데이터 소스는 ALTER TABLE문을 통해 요청되는 변경사항을 지원해야 합니다. 데이터 소스가 지원되지 않는 요청에 응답하는 방식에 따라, 오류가 리턴되거나 요청이 무시될 수 있습니다.
  - 투명한 DDL을 사용하여 작성되지 않은 리모트 기본 테이블을 변경하려고 하면 오류가 발생합니다.
- 기본 키, 고유 키 또는 외부 키에 대한 내재적 또는 명시적 변경은 패키지, 인덱스 및 기타 외부 키에 다음과 같은 영향을 미칠 수 있습니다.
  - 기본 키나 고유 키가 추가되는 경우
    - 패키지, 외부 키 또는 기존 고유 키에는 아무 영향을 주지 않습니다. 기본 키 또는 고유 키가 지연된 고유성을 지원하도록 변환되지 않아 이전 버전에서 작성된 기존의 고유 인덱스를 사용하는 경우, 인덱스는 변환되며 관련 테이블에 대한 갱신 사용을 갖는 패키지는 무효화됩니다.
  - 기본 키나 고유 키가 삭제(drop)되는 경우

- 제한조건에 대해 인덱스가 자동으로 작성된 경우, 인덱스가 삭제됩니다. 해당 인덱스에 종속된 모든 패키지는 무효화됩니다.
- 인덱스가 제한조건에 대해 고유하도록 변환된 경우 이 인덱스는 다시 비고유 인덱스로 설정되며, 더 이상 시스템 필수 인덱스가 아닙니다. 해당 인덱스에 종속된 모든 패키지는 무효화됩니다.
- 인덱스가 제한조건에 대해 사용된 기존의 고유 인덱스인 경우, 이 인덱스는 더 이상 시스템 필수 인덱스로 설정되지 않습니다. 패키지에는 아무 영향을 주지 않습니다.
- 모든 종속 외부 키가 삭제됩니다. 다음 항목에 지정된 대로, 각 종속 외부 키에 대해 추가 조치가 취해집니다.

– 외부 키가 NOT ENFORCED에서 ENFORCED로 (또는 ENFORCED에서 NOT ENFORCED로) 추가, 삭제(drop) 또는 변경되는 경우

- 오브젝트 테이블에 대해 삽입 사용을 갖고 있는 모든 패키지가 무효화됩니다.
- 외부 키 내에서 최소한 한 컬럼에 대해 갱신 사용을 갖고 있는 모든 패키지가 무효화됩니다.
- 상위 테이블에서 삭제 사용을 갖고 있는 모든 패키지가 무효화됩니다.
- 상위 키 내에서 적어도 한 컬럼에 대해 갱신 사용을 갖고 있는 모든 패키지가 무효화됩니다.

– 외부 키 또는 함수적 종속성이 ENABLE QUERY OPTIMIZATION에서 DISABLE QUERY OPTIMIZATION로 변경되는 경우

- 최적화를 위해 제한조건에 종속된 모든 패키지가 무효화됩니다.

- 테이블에 컬럼을 추가하면 변경된 테이블에 대해 삽입 사용을 갖고 있는 모든 패키지가 무효화됩니다. 추가된 컬럼이 테이블에서 사용자가 정의한 첫 번째 구조화된 유형 컬럼일 경우, 변경된 테이블에 대해 DELETE 사용을 갖고 있는 패키지도 무효화됩니다.
- 이미 존재하고 무결성 설정 보류 상태에 있지 않는 테이블에 점검 또는 참조 제한조건을 추가하거나, 무결성 설정 보류 상태에 있지 않은 기존 테이블에 대해 기존의 점검 또는 참조 제한조건을 NOT ENFORCED에서 ENFORCED로 변경하면, 테이블에 있는 기존 행이 제한조건에 대해 즉시 평가됩니다. 확인을 할 수 없으면, 오류가 리턴됩니다(SQLSTATE 23512). 테이블이 무결성 설정 보류 상태에 있을 경우, 점검 또는 참조 제한조건을 추가하거나 제한조건을 NOT ENFORCED에서 ENFORCED로 변경하면 제한조건의 강제 실행이 즉시 수행되지 않습니다. 제한조건의 강제 실행을 시작하려면 IMMEDIATE CHECKED 옵션과 함께 SET INTEGRITY문을 발행하십시오.

- 점검 제한조건을 추가, 변경 또는 삭제하면 오브젝트 테이블에 대해 삽입을 사용할 수 있거나, 제한조건과 관련된 컬럼 중 적어도 한 컬럼에 대해 갱신을 수행할 수 있거나, 성능 향상을 위해 제한조건을 사용하는 선택을 수행할 수 있는 모든 패키지가 무효화됩니다.
- 분산 키를 추가하면 분산 키의 컬럼 중 최소한 한 컬럼에서 갱신을 사용하는 모든 패키지가 무효화됩니다.
- 기본 키의 첫 번째 컬럼이 디폴트값으로 정의된 분산 키는 기본 키를 삭제하거나 다른 기본 키를 추가하는 작업의 영향을 받지 않습니다.
- 컬럼을 삭제하거나 컬럼의 데이터 유형을 변경하면 변경 중인 테이블에서 모든 Runstats 정보가 제거됩니다. Runstats이 다시 액세스 가능해지기 전에 테이블에서 Runstats이 실행되어야 합니다. 테이블이 명시적으로 삭제된 컬럼을 포함하지 않은 경우 테이블의 통계 프로파일이 보존됩니다.
- 컬럼의 길이를 증가시키거나 데이터 유형 또는 널(null) 가능성 속성을 변경하기 위해 컬럼을 변경하거나, 컬럼을 삭제하면 참조 제한조건을 통해 직접적 또는 간접적으로 테이블을 참조하는 모든 패키지가 무효화됩니다.
- 컬럼의 길이를 증가시키거나 데이터 유형 또는 널(null) 가능성 속성을 변경하기 위해 컬럼을 변경하면 유형이 지정된 뷰를 제외한 테이블에 종속된 뷰를 재생성합니다. 뷰를 재생성하는 중에 문제점이 발생하면 오류가 리턴됩니다(SQLSTATE 56098). 테이블에 종속된 유형이 지정된 뷰는 모두 작동하지 않는 것으로 표시됩니다.
- 컬럼 길이를 증가시키기 위해 컬럼을 변경하거나 테이터 유형을 변경하면 모든 종속 트리거 및 SQL 함수가 올바르지 않은 것으로 표시되고, 다음 사용시 내재적으로 다시 컴파일됩니다. 오브젝트를 재생성하는 중에 문제점이 발생하면 오류가 리턴됩니다 (SQLSTATE 56098).
- 트리거 또는 SQL 함수를 포함하는 명령문이 준비되거나 바운드되는 경우, 컬럼의 길이를 증가시키거나 데이터 유형 또는 널(null) 가능성 속성을 변경하기 위해 컬럼을 변경하면 트리거 또는 SQL 함수를 처리하는 중에 오류가 발생할 수 있습니다 (SQLSTATE 54010). 전이 변수 및 전이 테이블 컬럼 길이의 합계에 기반하는 행 길이가 너무 길 때 이 오류가 발생할 수 있습니다. 이러한 트리거 또는 SQL 함수가 삭제된 후 이를 다시 작성하려고 하면 오류가 발생합니다(SQLSTATE 54040).
- 인라인 길이를 증가시키기 위해 구조화된 유형 컬럼 또는 XML 유형 컬럼을 변경하면, 참조 제한조건 및 트리거를 통해 직접 또는 간접적으로 테이블을 참조하는 모든 패키지가 무효화됩니다.
- 인라인 길이를 증가시키기 위해 구조화된 유형 컬럼 또는 XML 유형 컬럼을 변경하면, 테이블에 종속된 뷰가 재생성됩니다.
- XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 온라인 테이블 이동을 사용하여 테이블을 이주한 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전을 작성할 수 있습니다.

- 테이블의 LOCKSIZE를 변경하면 교체된 테이블에 종속된 모든 패키지가 무효화됩니다.
- VOLATILE이나 NOT VOLATILE CARDINALITY를 변경하면 변경된 테이블에 종속된 모든 패키지가 무효화됩니다.
- ***복제:*** 컬럼의 길이를 증가하거나 데이터 유형을 변경하는 경우 주의를 실행하십시오. 응용프로그램 테이블과 연결된 데이터 변경 테이블이 이미 DB2 행 *크기* 한계에 있거나 이 한계에 근접했을 수 있습니다. 두 테이블 모두에 대해 변경이 완료될 수 있으려면, 응용프로그램 테이블에 앞서 데이터 변경 테이블이 변경되거나 동일한 작업 단위 내에서 두 테이블이 변경되어야 합니다. 행 *크기* 한계에 있거나 이에 근접하거나 또는 기존 컬럼의 길이를 증가시키는 기능이 부족한 플랫폼에 있는 사본에 대해서는 주의를 기울여야 합니다.

  Capture 프로그램이 변경된 속성으로 로그 레코드를 처리하기 전에 데이터 변경 테이블이 변경되지 않으면, Capture 프로그램이 실행되지 않을 수도 있습니다. 변경된 컬럼이 있는 사본이 사본 실행을 유지보수하는 서브스크립션에 앞서 변경되지 않으면, 서브스크립션이 실행되지 않을 수도 있습니다.
- 보호된 테이블에서 파티션을 접속 해제하는 경우 DB2가 자동으로 생성한 목표 테이블은 소스 테이블이 보호된 것과 동일한 방식으로 보호됩니다.
- 행 레벨 단위로 보호되는 것처럼 테이블을 변경하는 경우 테이블에 종속된 모든 캐시된 동적 SQL 섹션이 무효화됩니다. 마찬가지로 테이블에 종속된 모든 패키지도 무효화됩니다.
- 보호된 컬럼이 되는 것처럼 테이블 T의 컬럼이 변경되는 경우 테이블 T에 종속된 모든 캐시된 동적 SQL 섹션이 무효화됩니다. 마찬가지로 테이블 T에 종속된 모든 패키지도 무효화됩니다.
- 보호되지 않은 컬럼이 되는 것처럼 테이블 T의 컬럼이 변경되는 경우 테이블 T에 종속된 모든 캐시된 동적 SQL 섹션이 무효화됩니다. 마찬가지로 테이블 T에 종속된 모든 패키지도 무효화됩니다.
- 테이블의 기존 행의 경우, 보안 레이블 컬럼의 값은 행 보안 레이블을 추가하는 ALTER문 실행시 세션 권한 부여 ID의 *쓰기* 액세스에 대한 보안 레이블에 대한 디폴트입니다.
- ***내재적으로 숨겨진 컬럼 고려사항:*** 내재적으로 숨김으로 정의된 컬럼은 ALTER TABLE 명령문에서 명시적으로 참조할 수 있습니다. 예를 들어, 내재적으로 숨겨진 컬럼을 변경하거나 참조 제한조건의 파트로 지정하고, 제한조건이나 구체화된 쿼리 테이블 정의를 점검합니다.
- ***호환성***: 이전 버전의 DB2 제품과의 호환성을 위해,
  - 다음의 경우 ADD 키워드는 선택적입니다.
    - 이름이 지정되지 않은 PRIMARY KEY 제한조건
    - 이름이 지정되지 않은 참조 제한조건

- 이름이 FOREIGN KEY 구문 뒤에 오는 참조 제한조건

– references-clause를 정의하는 *column-definition*에서 CONSTRAINT 키워드를 생략할 수 있습니다.

– *constraint-name*을 FOREIGN KEY 다음에 지정할 수 있습니다(CONSTRAINT 키워드없이).

– SET MATERIALIZED QUERY AS 대신 SET SUMMARY AS를 지정할 수 있습니다.

– DROP MATERIALIZED QUERY 대신 SET MATERIALIZED QUERY AS DEFINITION ONLY를 지정할 수 있습니다.

– ADD MATERIALIZED QUERY(fullselect) 대신 SET MATERIALIZED QUERY AS(fullselect)를 지정할 수 있습니다.

– ADD DISTRIBUTE BY HASH를 대신해서 ADD PARTITIONING KEY를 지정할 수 있으며 이 경우 선택적 USING HASHING절도 지정될 수 있음

– DROP PARTITIONING KEY가 DROP DISTRIBUTION 대신 지정될 수 있습니다.

– LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 계속 지원되지만 사용되지 않으며 권장되지 않습니다(특히 휴대용 응용프로그램의 경우).

이전 버전의 DB2 제품과의 호환성 및 일관성을 위해,

– *identity-alteration*절에서 여러 옵션을 구분하는 데 쉼표를 사용할 수 있습니다.

z/OS용 DB2와의 호환성을 위해,

– P ARTITION 대신 PART 지정 가능

– ENDING AT 대신 VALUES 지정 가능

다음 구문도 지원됩니다.

– NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE 및 NOORDER

예:

예 1: 한 문자 길이의 새 컬럼 RATING을 DEPARTMENT 테이블에 추가합니다.

```
ALTER TABLE DEPARTMENT
   ADD RATING CHAR(1)
```

예 2: SITE_NOTES라는 새 컬럼을 PROJECT 테이블에 추가하십시오. 최대 길이가 1000바이트인 가변 길이 컬럼으로 SITE_NOTES를 작성하십시오. 컬럼의 값은 연관 문자 세트를 갖고 있지 않으므로 변환될 수 없습니다.

```
ALTER TABLE PROJECT
   ADD SITE_NOTES  VARCHAR(1000) FOR BIT DATA
```

예 3: EQUIPMENT라는 테이블이 다음 컬럼으로 정의되어 존재한다고 가정하십시오.

```
Column Name         Data Type
EQUIP_NO            INT
EQUIP_DESC          VARCHAR(50)
   LOCATION            VARCHAR(50)
EQUIP_OWNER         CHAR(3)
```

소유자(EQUIP_OWNER)가 DEPARTMENT 테이블에 있는 부서 번호(DEPTNO)가 되도록 참조 제한조건을 EQUIPMENT 테이블에 추가하십시오. DEPTNO는 DEPARTMENT 테이블의 기본 키입니다. 부서가 DEPARTMENT 테이블에서 제거된 경우, 그 부서에서 소유하는 모든 장비에 대한 소유자(EQUIP_OWNER) 값은 지정되지 않아야 합니다(또는 널(NULL)로 설정). 제한조건에 DEPTQUIP라는 이름을 지정하십시오.

```
ALTER TABLE EQUIPMENT
     ADD CONSTRAINT DEPTQUIP
      FOREIGN KEY (EQUIP_OWNER)
        REFERENCES DEPARTMENT
           ON DELETE SET NULL
```

또한 추가 컬럼에서는 이 장비 레코드와 연관된 양을 기록하는 것을 허용해야 합니다. 달리 지정하지 않으면, EQUIP_QTY 컬럼의 값은 1이어야 하며 널(NULL)이 아니어야 합니다.

```
ALTER TABLE EQUIPMENT
  ADD COLUMN EQUIP_QTY
  SMALLINT NOT NULL DEFAULT 1
```

예 4: EMPLOYEE 테이블을 변경하십시오. 각 사원에 대해 급여와 수당의 합계가 $30,000을 초과하도록 정의된 점검 제한조건 REVENUE를 추가하십시오.

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT REVENUE
  CHECK (SALARY + COMM > 30000)
```

예 5: EMPLOYEE 테이블을 변경하십시오. 이전에 정의된 제한조건 REVENUE를 삭제하십시오.

```
ALTER TABLE EMPLOYEE
  DROP CONSTRAINT REVENUE
```

예 6: SQL 변경사항을 디폴트 형식으로 로그하도록 테이블을 변경하십시오.

```
ALTER TABLE SALARY1
  DATA CAPTURE NONE
```

예 7: SQL 변경사항을 확장 형식으로 로그하도록 테이블을 변경하십시오.

```
ALTER TABLE SALARY2
  DATA CAPTURE CHANGES
```

예 8: 디폴트값을 갖는 4개의 새 컬럼을 추가하도록 EMPLOYEE 테이블을 변경하십시오.

```
ALTER TABLE EMPLOYEE
   ADD COLUMN HEIGHT MEASURE   DEFAULT MEASURE(1)
   ADD COLUMN BIRTHDAY BIRTHDATE DEFAULT DATE('01-01-1850')
   ADD COLUMN FLAGS BLOB(1M)  DEFAULT BLOB(X'01')
   ADD COLUMN PHOTO PICTURE   DEFAULT BLOB(X'00')
```

디폴트값은 디폴트값 지정시 다양한 함수 이름을 사용합니다. MEASURE는 INTEGER를 기반으로 하는 구별 유형이므로, MEASURE 함수가 사용됩니다. HEIGHT 컬럼 디폴트값은 MEASURE의 소스 유형이 BLOB 또는 날짜 시간 데이터 유형이 아니므로, 함수없이 지정할 수 있습니다. BIRTHDATE는 DATE를 기반으로 하는 구별 유형이므로, DATE 함수가 사용됩니다(여기서는 BIRTHDATE를 사용할 수 없음). FLAGS 및 PHOTO 컬럼의 경우, PHOTO가 구별 유형이더라도 디폴트값은 BLOB 함수를 사용하여 지정됩니다. BIRTHDAY, FLAGS 및 PHOTO 컬럼의 디폴트값을 지정하려면, 유형이 BLOB이거나 BLOB 또는 날짜 시간 데이터 유형을 기반으로 하는 구별 유형이므로 함수를 사용해야 합니다.

예 9: CUSTOMERS라는 테이블이 다음 컬럼으로 정의됩니다.

```
Column Name        Data Type
BRANCH_NO          SMALLINT
CUSTOMER_NO        DECIMAL(7)
CUSTOMER_NAME      VARCHAR(50)
```

이 테이블에서 기본 키는 BRANCH_NO 및 CUSTOMER_NO 컬럼으로 구성됩니다. 테이블을 분산하려면 테이블에 대한 분산 키를 작성해야 합니다. 테이블은 단일 노드 데이터베이스 파티션 그룹의 테이블 스페이스에 정의해야 합니다. 기본 키는 분산 키 컬럼의 수퍼 세트여야 합니다. 기본 키의 컬럼 중 최소한 하나가 분산 키로 사용되어야 합니다. 다음과 같이 BRANCH_NO를 분산 키로 만드십시오.

```
ALTER TABLE CUSTOMERS
  ADD DISTRIBUTE BY HASH (BRANCH_NO)
```

예 10: 리모트 테이블 EMPLOYEE가 투명한 DDL을 사용하는 페더레이티드 시스템에서 작성되었습니다. 리모트 테이블 EMPLOYEE를 변경하여 컬럼 PHONE_NO 및 WORK_DEPT를 추가하십시오. 그리고 기본 키를 기존 컬럼 EMP_NO 및 새 컬럼 WORK_DEPT에 추가하십시오.

```
ALTER TABLE EMPLOYEE
  ADD COLUMN PHONE_NO CHAR(4) NOT NULL
  ADD COLUMN WORK_DEPT CHAR(3)
  ADD PRIMARY KEY (EMP_NO, WORK_DEPT)
```

예 11: 함수적 종속성 FD1을 추가하도록 DEPARTMENT 테이블을 변경한 후 DEPARTMENT 테이블에서 함수적 종속성 FD1을 삭제하십시오.

```
ALTER TABLE DEPARTMENT
  ADD CONSTRAINT FD1
    CHECK ( DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED

ALTER TABLE DEPARTMENT
  DROP CHECK FD1
```

예 12: EMPLOYEE 테이블에 있는 WORKDEPT 컬럼의 디폴트값을 123으로 변경하십시오.

```
ALTER TABLE EMPLOYEE
  ALTER COLUMN WORKDEPT
    SET DEFAULT '123'
```

예 13: EMPLOYEE 테이블에 DATA_ACCESS 보안 규정을 연관시키십시오.

```
ALTER TABLE EMPLOYEE
  ADD SECURITY POLICY DATA_ACCESS
```

예 14: SALARY 컬럼을 보호하도록 EMPLOYEE 테이블을 변경하십시오.

```
ALTER TABLE EMPLOYEE
  ALTER COLUMN SALARY
  SECURED WITH EMPLOYEESECLABEL
```

예 15: 다음 컬럼으로 정의된 SALARY_DATA라는 이름의 테이블이 있다고 가정하십시오.

```
Column Name            Data Type
-----------            ---------
EMP_NAME               VARCHAR(50) NOT NULL
EMP_ID                 SMALLINT NOT NULL
EMP_POSITION           VARCHAR(100) NOT NULL
SALARY                 DECIMAL(5,2)
PROMOTION_DATE         DATE NOT NULL
```

해당 테이블을 변경하여 급여를 DECIMAL(6,2) 컬럼에 저장할 수 있도록 하고 PROMOTION_DATE를 널(NULL) 값으로 설정될 수 있는 선택적 필드로 만든 후 EMP_POSITION 컬럼을 제거하십시오.

```
ALTER TABLE SALARY_DATA
  ALTER COLUMN SALARY SET DATA TYPE DECIMAL(6,2)
  ALTER COLUMN PROMOTION_DATE DROP NOT NULL
       DROP COLUMN EMP_POSITION
```

ALTER TABLESPACE

ALTER TABLESPACE문은 다음과 같은 방법으로 기존의 테이블 스페이스를 수정하는 데 사용됩니다.

- 컨테이너를 DMS 테이블 스페이스에 추가하거나 삭제합니다. 즉, 테이블 스페이스가 MANAGED BY DATABASE 옵션을 사용하여 작성됩니다.
- DMS 테이블 스페이스의 컨테이너 크기를 수정합니다.
- Extent 이동을 통해 DMS 테이블 스페이스의 상위 워터 마크(water mark)를 낮춥니다.
- 현재 컨테이너가 없는 데이터베이스 파티션의 SMS 테이블 스페이스에 컨테이너를 추가합니다.
- 테이블 스페이스에 대한 PREFETCHSIZE 설정값을 수정합니다.
- 테이블 스페이스에서 테이블에 사용된 BUFFERPOOL을 수정합니다.
- 테이블 스페이스에 대한 OVERHEAD 설정값을 수정합니다.
- 테이블 스페이스에 대한 TRANSFERRATE 설정값을 수정합니다.
- 테이블 스페이스에 대한 파일 시스템 캐싱 규정을 수정합니다.
- DMS 또는 자동 스토리지 테이블 스페이스의 자동 크기 조정을 사용하거나 사용하지 않도록 설정합니다.
- 일반 또는 대형 자동 스토리지 테이블 스페이스를 재조정합니다.

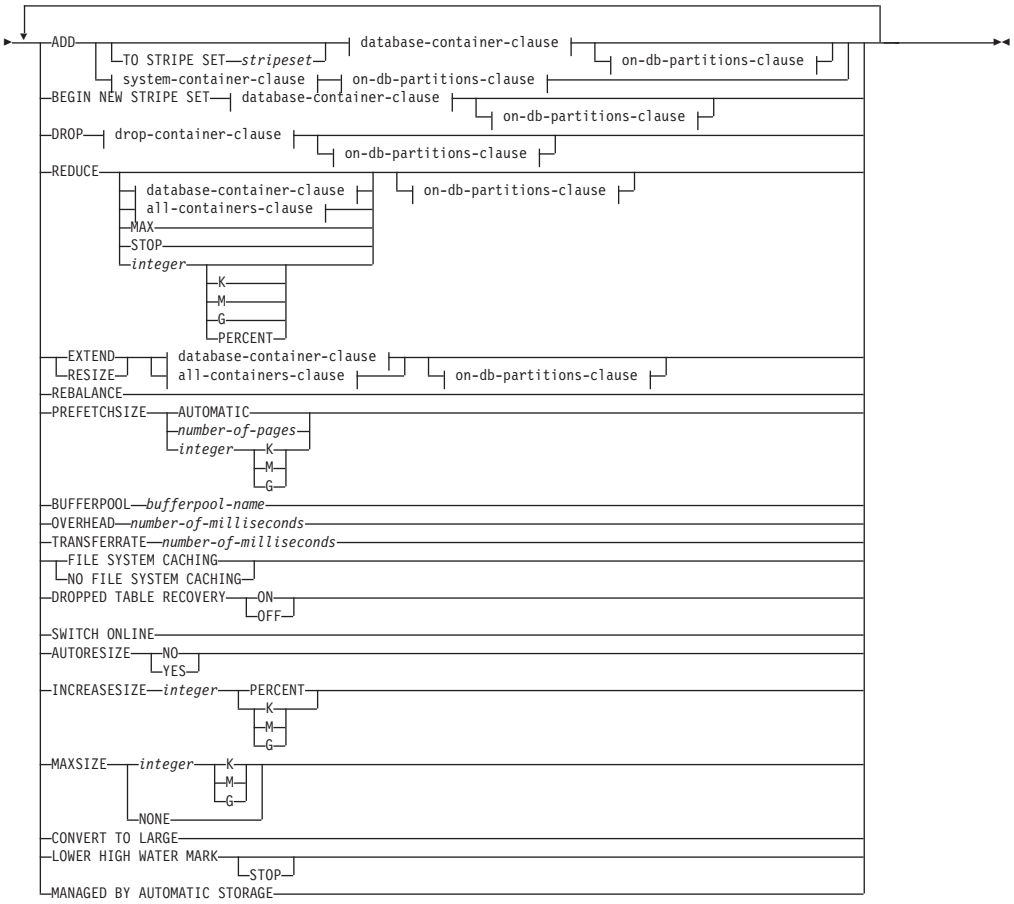
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

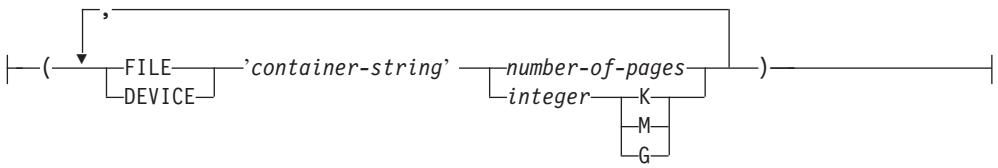
권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 SYSCTRL 권한을 포함해야 합니다.

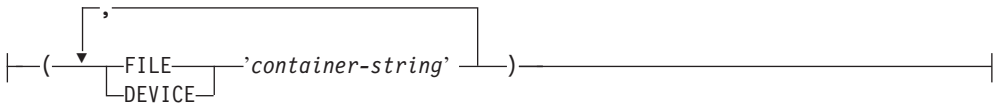
구문



database-container-clause:

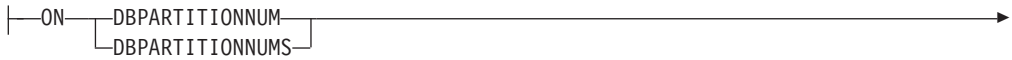


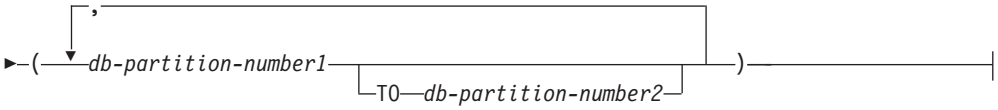
drop-container-clause:



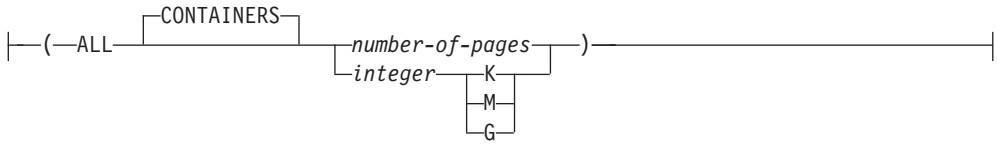
system-container-clause:

on-db-partitions-clause:





all-containers-clause:



설명

tablespace-name

테이블 스페이스 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 긴 SQL ID(일반 ID 또는 분리 ID)입니다.

ADD

하나 이상의 새 컨테이너가 테이블 스페이스에 추가되도록 지정합니다.

TO STRIPE SET *stripeset*

하나 이상의 새 컨테이너가 테이블 스페이스에 추가되도록 지정하고, 새 컨테이너가 지정한 스트라이프 세트에 놓이도록 지정합니다.

BEGIN NEW STRIPE SET

테이블 스페이스에 새 스트라이프 세트가 작성되도록 지정하고, 하나 이상의 컨테이너가 이 새 스트라이프 세트에 추가되도록 지정합니다. TO STRIPE SET를 지정하지 않으면 ADD 옵션을 사용하여 이후에 추가되는 컨테이너는 이 새 스트라이프 세트에 추가됩니다.

DROP

하나 이상의 컨테이너가 테이블 스페이스에서 삭제되도록 지정합니다.

REDUCE

자동 스토리지 테이블 스페이스가 아닌 경우, 기존 컨테이너의 *크기*가 감소하도록 지정합니다. 지정된 *크기*는 기존 컨테이너가 감소되는 단위 *크기*입니다. *all-containers-clause*를 지정할 경우, 테이블 스페이스에 있는 모든 컨테이너가 이 *크기*만큼 감소합니다. *크기* 감소가 현재 상위 워터 마크(water mark)보다 작은 테이블 스페이스 *크기*에 영향을 주면 컨테이너 감소 시도 전에 상위 워터 마크 감소를 시도하

게 됩니다. 자동 스토리지 테이블 스페이스가 아닌 경우 REDUCE절 다음에 *database-container-clause* 또는 *all-containers-clause*가 와야 합니다.

자동 스토리지 테이블 스페이스의 경우 현재 상위 워터 마크(water mark)가 감소하도록 지정하고, 가능하면 테이블 스페이스의 크기를 새 상위 워터 마크(water mark)로 감소할 수 있도록 지정합니다. 자동 스토리지 테이블 스페이스인 경우 REDUCE절 다음에 *database-container-clause* 또는 *all-containers-clause*가 와야 합니다.

주: MAX, 숫자 값, PERCENT 또는 STOP절이 있는 REDUCE 옵션과, STOP 절을 포함하는 LOWER HIGH WATER MARK 옵션은 재개 가능한 스토리지 속성을 가지고 있는 데이터베이스 관리 및 자동 스토리지 관리 테이블 스페이스에만 사용 가능합니다. 또한, 이러한 옵션은 다른 옵션(두 옵션 중 나머지 옵션을 포함하여) 없이 지정하고 실행해야 합니다.

database-container-clause
하나 이상의 컨테이너를 DMS 테이블 스페이스에 추가합니다. 테이블 스페이스는 응용프로그램 서버(AS)에 이미 존재하는 DMS 테이블 스페이스를 식별해야 합니다.

all-containers-clause
DMS 테이블 스페이스에 있는 모든 컨테이너를 확장하거나 줄이거나 크기를 조정합니다. 테이블 스페이스는 응용프로그램 서버(AS)에 이미 존재하는 DMS 테이블 스페이스를 식별해야 합니다.

MAX
재개 가능한 스토리지가 있는 자동 스토리지 테이블 스페이스의 경우 최대 개수의 Extent를 테이블 스페이스의 시작 부분으로 이동하여 상위 워터 마크(water mark)를 낮추도록 지정합니다. 또한 테이블 스페이스의 크기가 새 상위 워터 마크(water mark)로 감소됩니다. 이는 자동이 아닌 스토리지 테이블 스페이스에는 적용되지 않습니다.

STOP
재개 가능한 스토리지가 있는 자동 스토리지 테이블 스페이스의 경우, Extent 이동 조작이 진행 중이면 이 조작을 인터럽트합니다. 이 옵션은 자동이 아닌 스토리지 테이블 스페이스에 사용할 수 없습니다.

integer **[K | M | G]** 또는 *integer* **PERCENT**
재개 가능한 스토리지가 있는 자동 스토리지 테이블 스페이스의 경우 Extent 이동을 통해 테이블 스페이스가 감소되는 정도를 나타내는 숫자 값을 지정합니다. 값은 몇 가지의 방법으로 표현할 수 있습니다.

- K, M, G 또는 PERCENT 없이 지정된 정수는 숫자 값이 테이블 스페이스가 감소될 페이지 수임을 표시합니다.

- K, M 또는 G와 함께 지정된 정수는 각각 KB, MB 또는 GB 단위의 감소 *크기*를 표시합니다. 값은 우선 테이블 스페이스의 페이지 *크기*를 기초로 바이트에서 페이지 수로 변환됩니다.
- PERCENT와 함께 지정된 정수는 이동할 Extent 수를 테이블 스페이스의 현재 *크기* 백분율로 표시한 것입니다.

Extent 이동이 완료되면, 테이블 스페이스 *크기*가 새 상위 워터 마크(water mark)로 감소됩니다. 이 옵션은 자동이 아닌 스토리지 테이블 스페이스에 사용할 수 없습니다.

on-db-partitions-clause

해당 컨테이너 조작에 대해 하나 이상의 데이터베이스 파티션을 지정합니다.

EXTEND

기존 컨테이너의 *크기*가 증가하도록 지정합니다. 지정된 *크기*는 기존 컨테이너가 증가되는 단위 *크기*입니다. *all-containers-clause*를 지정할 경우, 테이블 스페이스에 있는 모든 컨테이너가 이 *크기*만큼 증가합니다.

RESIZE

기존 컨테이너의 *크기*가 변경되도록 지정합니다. 지정된 *크기*는 컨테이너의 새 *크기*입니다. *all-containers-clause*을 지정할 경우, 테이블 스페이스에 있는 모든 컨테이너가 이 *크기*로 변경됩니다. *크기* 조정 조작이 여러 개의 컨테이너에 영향을 줄 경우에는 영향을 받는 컨테이너의 *크기*를 모두 늘리거나 모두 줄여야 합니다. 따라서 어떤 컨테이너의 *크기*는 늘리면서 다른 컨테이너의 *크기*는 줄일 수 없습니다 (SQLSTATE 429BC).

database-container-clause

하나 이상의 컨테이너를 DMS 테이블 스페이스에 추가합니다. 테이블 스페이스는 응용프로그램 서버(AS)에 이미 존재하는 DMS 테이블 스페이스를 식별해야 합니다.

drop-container-clause

하나 이상의 컨테이너를 DMS 테이블 스페이스에서 삭제합니다. 테이블 스페이스는 응용프로그램 서버(AS)에 이미 존재하는 DMS 테이블 스페이스를 식별해야 합니다.

system-container-clause

하나 이상의 컨테이너를 지정된 데이터베이스 파티션의 SMS 테이블 스페이스에 추가합니다. 테이블 스페이스는 응용프로그램 서버(AS)에 이미 존재하는 SMS 테이블 스페이스를 식별해야 합니다. 테이블 스페이스에 대해 지정된 데이터베이스 파티션에 컨테이너가 있으면 안됩니다(SQLSTATE 42921).

on-db-partitions-clause

해당 컨테이너 조작에 대해 하나 이상의 데이터베이스 파티션을 지정합니다.

all-containers-clause

DMS 테이블 스페이스에 있는 모든 컨테이너를 확장하거나 줄이거나 크기를 조정합니다. 테이블 스페이스는 응용프로그램 서버(AS)에 이미 존재하는 DMS 테이블 스페이스를 식별해야 합니다.

REBALANCE

일반 및 대형 자동 스토리지 테이블 스페이스의 경우, 최근에 추가된 스토리지 경로에서의 컨테이너 작성이나 『삭제 보류』 상태의 스토리지 경로에서의 컨테이너 삭제 또는 둘 다를 시작합니다. 재조정 중에, 데이터는 새 경로의 컨테이너로 이동하고 삭제된 경로의 컨테이너 밖으로 이동됩니다. 재조정은 백그라운드에서 비동기식으로 실행되므로 데이터의 사용 가능성에는 영향을 주지 않습니다.

PREFETCHSIZE

쿼리에서 참조되기 전에 쿼리에서 필요한 데이터를 읽도록 지정하여, 입출력 수행시 쿼리가 지연되지 않도록 합니다.

AUTOMATIC

테이블 스페이스의 프리페치 크기가 자동으로 갱신되도록 지정하며, 즉 프리페치 크기는 다음 공식을 사용하여 DB2가 관리합니다.

```
프리페치 크기 =
  (컨테이너 수) *
  (컨테이너당 실제 디스크 수) *
  (Extent 크기)
```

DB2_PARALLEL_IO 레지스트리 변수를 통해 값을 지정하지 않을 경우, 컨테이너당 실제 디스크 수의 디폴트값은 1입니다.

DB2 데이터베이스는 하나 이상의 컨테이너를 추가 또는 삭제하는 ALTER TABLESPACE문의 성공적인 실행 후 테이블 스페이스의 컨테이너 수가 변경될 때마다 프리페치 크기를 자동으로 갱신합니다. 프리페치 크기는 데이터베이스가 시작될 때 갱신됩니다.

프리페치 크기의 자동 갱신은 PREFETCHSIZE절에 숫자 값을 지정하여 해제할 수 있습니다.

number-of-pages

데이터 프리페치가 수행되는 동안 테이블 스페이스로부터 읽어 들일 PAGESIZE 페이지의 수를 지정합니다. 프리페치 크기 값은 정수로도 지정할 수 있으며 숫자 뒤에 K(킬로바이트의 경우), M(메가바이트의 경우) 또는 G(기가바이트의 경우)가 붙습니다. 이런 방식으로 지정될 경우, 바이트 수를 페이지 크기로 나눈 최저값은 프리페치 크기에 대한 페이지 수를 결정하는 데 사용됩니다.

BUFFERPOOL *bufferpool-name*

이 테이블 스페이스의 테이블에 사용된 버퍼 풀의 이름입니다. 버퍼 풀이 현재 데이터베이스에 있어야 합니다(SQLSTATE 42704). 테이블 스페이스의 데이터베이스 파티션 그룹은 버퍼 풀에 대해 정의되어야 합니다(SQLSTATE 42735).

OVERHEAD *number-of-milliseconds*

입출력 제어기 오버헤드 및 디스크 탐색 및 대기 시간(밀리초)을 지정하는 숫자 리터럴(정수, 소수 또는 부동 소수점)입니다. 이 숫자는 모든 컨테이너에 대해 같지 않을 경우, 테이블 스페이스에 속하는 모든 컨테이너에 대한 평균이어야 합니다. 이 값은 쿼리 최적화 시 I/O 비용을 계산할 때 사용됩니다.

TRANSFERRATE *number-of-milliseconds*

한 페이지(4K 또는 8K)를 메모리로 읽어들이는 시간(밀리초)을 지정하는 숫자 리터럴(정수, 소수 또는 부동 소수점)입니다. 이 숫자는 모든 컨테이너에 대해 같지 않을 경우, 테이블 스페이스에 속하는 모든 컨테이너에 대한 평균이어야 합니다. 이 값은 쿼리 최적화 시 I/O 비용을 계산할 때 사용됩니다.

FILE SYSTEM CACHING 또는 NO FILE SYSTEM CACHING

I/O 조작이 파일 시스템 레벨에서 캐시되는지의 여부를 지정합니다. 새 캐싱 규정이 적용되기 전에 데이터베이스 연결을 종료해야 합니다. Long 또는 LOB 데이터에 대한 I/O 액세스는 SMS 및 DMS 컨테이너 모두에 대해 버퍼링됨을 유의하십시오.

FILE SYSTEM CACHING

목표 테이블 스페이스의 모든 I/O 조작이 파일 시스템 레벨에서 캐시됩니다.

NO FILE SYSTEM CACHING

모든 I/O 조작은 파일 시스템 레벨 캐시를 생략합니다.

DROPPED TABLE RECOVERY

*tablespace-name*에서 삭제(drop)된 테이블은 ROLLFORWARD DATABASE 명령의 RECOVER DROPPED TABLE ON 옵션을 사용하여 복구 여부를 지정합니다. 하나 이상의 테이블 스페이스의 파티션되지 않은 테이블에 대해서는 삭제된 테이블 복구 기능이 꺼져 있지만, 파티션된 테이블의 경우는 항상 켜져 있습니다.

ON

삭제된 테이블을 복구할 수 있도록 지정합니다.

OFF

삭제된 테이블을 복구할 수 없도록 지정합니다.

SWITCH ONLINE

OFFLINE 상태의 테이블 스페이스는 컨테이너가 액세스 가능하게 될 경우에 온라인 상태가 되도록 지정합니다. 컨테이너가 액세스 가능해지지 않으면 오류가 리턴됩니다(SQLSTATE 57048).

AUTORESIZE

데이터베이스 관리 스페이스(DMS) 테이블 스페이스 또는 자동 스토리지 테이블 스페이스의 자동 크기 조정 성능의 사용 가능 여부를 지정합니다. 자동 크기 조정 가능 테이블 스페이스는 가득 차게 되면 자동으로 크기가 증가됩니다.

NO

DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스의 자동 크기 조정 성능의 사용 안함 여부를 지정합니다. 자동 크기 조정 성능을 사용하지 않으면 이전에 INCREASESIZE 또는 MAXSIZE에 지정한 값이 보존되지 않습니다.

YES

DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스의 자동 크기 조정 성능의 사용을 지정합니다.

INCREASESIZE *integer* **PERCENT 또는 INCREASESIZE** *integer* **K | M | G**

테이블 스페이스가 가득 차고 스페이스 요청이 있으면 자동 크기 조정이 사용 가능한 테이블 스페이스가 자동으로 증가되는 양을 데이터베이스 파티션별로 지정합니다. 정수 값은 다음 앞에 나와야 합니다.

- 스페이스 요청시 테이블 스페이스 크기 백분율로 양을 지정하는 PERCENT. PERCENT가 지정되면 정수 값은 0 - 100 사이에 있어야 합니다(SQLSTATE 42615).
- 바이트 단위로 양을 지정하는 K(킬로바이트), M (메가바이트) 또는 G(기가바이트)

데이터베이스 관리 프로그램이 테이블 스페이스의 컨테이너에서 증가량을 일관적으로 유지하려고 하기 때문에 사용되는 실제 값은 지정한 값보다 약간 작거나 클 수 있습니다.

MAXSIZE *integer* **K | M | G 또는 MAXSIZE NONE**

자동 크기 조정이 사용 가능한 테이블 스페이스가 자동으로 증가될 수 있도록 최대 크기를 지정합니다.

integer

DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스가 자동으로 증가될 수 있도록 데이터베이스 파티션별로 크기에 대한 하드 한계를 지정합니다. 정수 값은 K(킬로바이트), M(메가바이트) 또는 G(기가바이트) 앞에 나와야 합니다. 데이터베이스 관리 프로그램이 테이블 스페이스의 컨테이너에서 증가량을 일관적으로 유지하려고 하기 때문에 사용되는 실제 값은 지정한 값보다 약간 작을 수 있습니다.

NONE

테이블 스페이스가 파일 시스템 용량 또는 최대 테이블 스페이스 크기(『SQL 한계』에서 설명)까지 증가될 수 있도록 지정합니다.

CONVERT TO LARGE

기존의 일반 DMS 테이블 스페이스를 대형 DMS 테이블 스페이스가 되도록 수정합니다. 변환 중에 테이블 스페이스 및 내용이 잠깁니다. 이 옵션은 일반 DMS 테이블 스페이스에서만 사용 가능합니다. SMS 테이블 스페이스, 임시 테이블 스페이스 또는 시스템 카탈로그 테이블 스페이스가 지정되면 오류가 리턴됩니다

(SQLSTATE 560CF). 사용자는 다른 테이블 스페이스의 데이터 파티션을 가진 파티션된 테이블의 데이터 파티션을 포함하는 테이블 스페이스를 변환할 수 없습니다(SQLSTATE 560CF). 변환은 커미트 후 거꾸로 변환될 수 없습니다. 테이블 스페이스의 테이블이 DATA CAPTURE CHANGES로 정의된 경우, 목표 테이블 및 테이블 스페이스의 스토리지 및 용량 한계를 고려하십시오.

LOWER HIGH WATER MARK

재개 가능한 스토리지가 있는 자동 스토리지 및 자동이 아닌 스토리지 테이블 스페이스 둘 다에 대해, Extent 이동 조작을 트리거하여 테이블 스페이스에서 최대 개수의 Extent를 낮게 이동시킵니다. 상위 워터 마크(water mark)가 낮아져도, 테이블 스페이스 *크기*는 감소되지 않습니다. 이 옵션 다음에는 자동 스토리지 테이블 스페이스의 경우 ALTER TABLESPACE REDUCE가, 자동이 아닌 스토리지 테이블 스페이스의 경우에는 *database-container-clause* 또는 *all-containers-clause*가 있는 ALTER TABLESPACE REDUCE가 있어야 합니다.

주: STOP절을 포함하는 LOWER HIGH WATER MARK 옵션과 MAX, 숫자 값, PERCENT 또는 STOP절이 있는 REDUCE 옵션은 재개 가능한 스토리지 속성을 가지고 있는 데이터베이스 관리 및 자동 스토리지 관리 테이블 스페이스에만 사용 가능합니다. 또한, 이러한 옵션은 다른 옵션(두 옵션 중 나머지 옵션을 포함하여) 없이 지정하고 실행해야 합니다.

STOP

재개 가능한 스토리지가 있는 자동 스토리지 및 자동이 아닌 스토리지 테이블 스페이스 둘 다에 대해, Extent 이동 조작이 진행 중이면 이 조작을 인터럽트합니다.

MANAGED BY AUTOMATIC STORAGE

데이터베이스 관리 스페이스(DMS) 테이블 스페이스에 자동 스토리지를 사용 가능하게 합니다. 자동 스토리지가 사용 가능하면, 테이블 스페이스에서 추가 컨테이너 조작을 실행할 수 없습니다. 변환되는 테이블 스페이스는 RAW(DEVICE) 컨테이너를 사용할 수 없습니다.

규칙

- ADD, DROP, EXTEND, REDUCE 및 RESIZE절이 다른 데이터베이스 파티션으로 방향 지정되지 않으면 BEGIN NEW STRIPE SET절은 ADD, DROP, EXTEND, REDUCE 및 RESIZE와 동일한 명령문에 지정될 수 없습니다(SQLSTATE 429BC).
- TO STRIPE SET절을 사용하여 지정한 스트라이프 세트 값은 변경할 테이블 스페이스에 대하여 유효한 범위 내에 있어야 합니다(SQLSTATE 42615).
- 테이블 스페이스에 스페이스를 추가하거나 제거할 때는 다음 규칙을 따라야 합니다.
  - 각 컨테이너의 *크기*가 증가할 경우 EXTEND와 RESIZE를 같은 명령문에서 사용할 수 있습니다(SQLSTATE 429BC).

  - 각 컨테이너의 *크기*가 감소할 경우 REDUCE와 RESIZE를 같은 명령문에서 사용할 수 있습니다(SQLSTATE 429BC).
  - EXTEND 및 REDUCE가 다른 데이터베이스 파티션으로 방향 지정되지 않으면 EXTEND 및 REDUCE는 동일한 명령문에서 쓰일 수 없습니다(SQLSTATE 429BC).
  - ADD 및 REDUCE 또는 DROP이 다른 데이터베이스 파티션으로 방향 지정되지 않으면 ADD는 REDUCE 또는 DROP과 동일한 명령문에서 쓰일 수 없습니다(SQLSTATE 429BC).
  - DROP 및 EXTEND 또는 ADD가 다른 데이터베이스 파티션으로 방향 지정되지 않으면 DROP은 EXTEND 또는 ADD와 동일한 명령문에서 쓰일 수 없습니다(SQLSTATE 429BC).

- 시스템 관리 스페이스(SMS) 테이블 스페이스, 자동 스토리지를 사용하여 작성된 임시 테이블 스페이스 또는 원시 디바이스 컨테이너를 사용하기 위해 정의된 DMS 테이블 스페이스에는 AUTORESIZE, INCREASESIZE 또는 MAXSIZE절을 지정할 수 없습니다(SQLSTATE 42601).
- 테이블 스페이스의 자동 *크기* 조정이 불가능한 경우 INCREASESIZE 또는 MAXSIZE절을 지정할 수 없습니다(SQLSTATE 42601).
- 테이블 스페이스에 새로운 최대 *크기*를 지정할 때 값은 각 데이터베이스 파티션에 대한 현재 *크기*보다 커야 합니다(SQLSTATE 560B0).
- 데이터베이스 관리 프로그램이 이러한 테이블 스페이스에 대한 스페이스 관리를 제어 하고 있기 때문에 자동 스토리지 테이블 스페이스에서 컨테이너 조작(ADD, EXTEND, RESIZE, DROP 또는 BEGIN STRIPE SET)을 수행할 수 없습니다(SQLSTATE 42858).
- 자동 *크기* 조정 가능 DMS 테이블 스페이스에 원시 디바이스 컨테이너를 추가할 수 없습니다(SQLSTATE 42601).
- CONVERT TO LARGE절은 다른 절과 동일한 명령문에서 지정될 수 없습니다(SQLSTATE 429BC).
- REBALANCE절은 다른 절과 함께 지정할 수 없습니다(SQLSTATE 429BC).
- REBALANCE절은 일반 및 대형 자동 스토리지 테이블 스페이스에 대해서만 유효합니다(SQLSTATE 42601). 임시 자동 스토리지 테이블 스페이스는 삭제하고 다시 작성하여 최근에 추가된 스토리지 경로를 이용하거나 해당 컨테이너가 삭제되는 스토리지 경로에서 제거되도록 해야 합니다.
- 컨테이너 조작 및 REBALANCE절은 테이블 스페이스가 『DMS 재조정 프로그램 사용 중』 상태에 있는 경우 지정할 수 없습니다(SQLSTATE 55041).

주

- 각 컨테이너 정의에는 컨테이너 이름을 저장하는 데 필요한 바이트 수 더하기 53바이트가 필요합니다. 테이블 스페이스의 모든 컨테이너 이름 길이의 합계가 20 480바이트를 초과할 수 없습니다(SQLSTATE 54034).
- 디폴트 컨테이너 조작은 ALTER TABLESPACE문에 지정된 컨테이너 조작이지만 특정 데이터베이스 파티션을 명시적으로 가리키지는 않습니다. 이러한 컨테이너 조작은 명령문에 나열되지 않은 데이터베이스 파티션으로 보내집니다. 이러한 디폴트 컨테이너 조작을 어떤 데이터베이스 파티션으로도 보내지 않으면 컨테이너 조작에 대하여 모든 데이터베이스 파티션이 명시적으로 언급되었기 때문에 경고가 리턴됩니다(SQLSTATE 01589).
- 테이블 스페이스에 스페이스가 추가되거나 제거되고 트랜잭션이 커미트되면 테이블 스페이스의 내용이 모든 컨테이너에서 재조정됩니다. 재조정되는 동안 테이블 스페이스에 대한 액세스는 제한되지 않습니다.
- 테이블 스페이스가 OFFLINE 상태에 있으며 컨테이너 액세스가 가능해지는 경우, 사용자가 모든 응용프로그램의 연결을 끊고 다시 데이터베이스에 연결하여 테이블 스페이스를 OFFLINE 상태에서 해제할 수 있습니다. 또는, SWITCH ONLINE 옵션을 사용하여 데이터베이스의 나머지 부분은 계속 실행하면서 테이블 스페이스를 OFFLINE에서 해제할 수 있습니다.
- 둘 이상의 컨테이너를 테이블 스페이스에 추가할 경우, 재조정 비용이 한 번만 발생하도록 컨테이너를 동일한 명령문에 추가하는 것이 좋습니다. 단일 트랜잭션 내에서 별도의 ALTER TABLESPACE문으로 동일한 테이블 스페이스에 컨테이너를 추가하면 오류가 발생합니다(SQLSTATE 55041).
- 존재하지 않는 컨테이너를 확장하거나, 줄이거나, *크기*를 조정하거나, 삭제하려고 하면 오류가 발생합니다(SQLSTATE 428B2).
- 컨테이너를 확장하거나, 줄이거나, *크기*를 조정할 때, 컨테이너 유형은 컨테이너가 작성될 때 사용된 유형과 일치해야 합니다(SQLSTATE 428B2).
- 단일 트랜잭션 내에서 별도의 ALTER TABLESPACE문으로 동일한 테이블 스페이스에서 컨테이너 *크기*를 변경하려고 하면 오류가 발생합니다(SQLSTATE 55041).
- 파티션된 데이터베이스에서 여러 개의 데이터베이스 파티션이 물리적으로 동일한 노드에 상주할 경우, 그러한 데이터베이스 파티션에 대해 동일한 디바이스나 특정 경로를 지정할 수 없습니다(SQLSTATE 42730). 이러한 환경에서는 각 데이터베이스 파티션에 대해 고유한 *container-string*을 지정하거나 상대 경로 이름을 사용하십시오.
- 테이블 스페이스 정의가 트랜잭션이 가능하며 커미트시 테이블 스페이스 정의에 대한 변경사항이 카탈로그 테이블에 반영된다고 하더라도, 다음 번에 데이터베이스가 시작될 때까지는 새로운 정의를 가진 버퍼 풀을 사용할 수 없습니다. ALTER TABLESPACE문이 발행될 때 사용 중이었던 버퍼 풀은 중간에 계속 사용됩니다.
- 필요하면 DMS 테이블 스페이스의 경우 REDUCE, RESIZE 또는 DROP 옵션이 사용되지 않는 Extent를 사용 가능하도록 시도하며, 자동 스토리지 테이블 스페이스

의 경우 REDUCE 옵션은 사용되지 않는 Extent를 사용 가능하도록 시도합니다. 사용되지 않는 Extent를 제거하면 상위 워터 마크(water mark)를 사용된 스페이스 양을 정확히 나타내는 값으로 줄일 수 있으며, 테이블 스페이스 공간에서 더 많이 줄일 수 있습니다.

- ***대용량 DMS 테이블 스페이스로 변환:***: 변환 후, COMMIT문을 발행하여 테이블 스페이스의 스토리지 용량을 증가시킬 것을 권장합니다.
  - 테이블 스페이스가 자동 크기 조정 기능 사용이 가능한 경우, MAXSIZE 테이블 스페이스 속성이 NONE으로 설정되어 있지 않으면 속성을 늘려야 합니다.
  - 테이블 스페이스가 자동 크기 조정 기능 사용이 불가능한 경우는
    - AUTORESIZE YES 옵션으로 ALTER TABLESPACE문을 발행하여 자동 크기 조정 기능 사용을 가능하게 하거나
    - 스트라이프 세트를 추가하여 스토리지를 추가하여 기존의 컨테이너 크기를 확장하거나, 두 가지 모두를 실행하십시오.

  변환된 테이블 스페이스의 테이블에 대한 인덱스가 대형 레코드 ID(RID)를 지원하기 전에 해당 인덱스가 재구성 또는 재빌드되어야합니다.
  - 인덱스는 CLEANUP ONLY절 없이 REORG INDEXES ALL 명령을 사용하여 재구성될 수 있습니다. 파티션된 테이블에 대해 ALLOW NO ACCESS 옵션을 지정하십시오.
  - 그 외의에 테이블을 재구성(INPLACE 아님)하여 모든 인덱스를 재빌드하고 테이블이 페이지당 255행 초과를 지원할 수 있도록 하는 방법이 있습니다.

  대형 RID를 지원하지 않는 테이블을 판별하려면 ADMIN_GET_TAB_INFO 테이블 함수를 사용하십시오.
- 『삭제 보류』 상태에 있는 스토리지 경로에 컨테이너가 있는 자동 스토리지 테이블 스페이스를 재조정하면 해당 컨테이너가 삭제됩니다. 삭제된 컨테이너 외부로 이동되는 데이터를 보유하기 위해 새 컨테이너를 작성해야 합니다. 컨테이너가 작성될 수 있도록 데이터베이스의 다른 스토리지 경로에 충분한 여유 공간이 있어야 합니다. 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 57011). 필요한 실제 여유 공간 양은 상위 워터 마크(water mark) Extent 및 변경되는 스트라이프 세트와 같은 많은 요소에 따라 결정됩니다. 그러나 조작이 성공했는지 확인하려면 남아 있는 스토리지 경로에 최소한의 충분한 여유 공간이 있어야 합니다. 삭제되는 컨테이너가 스페이스를 소비하기 때문입니다.
- REBALANCE절을 지정했지만 데이터 서버가 새 컨테이너를 작성하거나 기존 컨테이너를 삭제할 필요가 없다고 판단하면, 재조정은 발생하지 않고 명령문은 경고와 함께 성공합니다(SQLSTATE 01690).
- 최근에 추가된 경로에서 컨테이너를 추가하는 것 외에, REBALANCE 조작은 기존 스토리지 경로에서 컨테이너를 추가하는 경우에도 사용할 수 있습니다. 테이블 스페이스에 설정된 각 스트라이프 세트를 조사하고 특정 스트라이프 세트가 사용하고 있

지 않은 스토리지 경로가 식별됩니다. 식별된 스토리지 경로마다, 충분한 여유 공간이 있으면 컨테이너가 새로 작성됩니다. 컨테이너는 스트라이프 세트에 있는 다른 컨테이너와 동일한 *크기*를 갖습니다. 이는 제공된 스토리지 경로에 공간이 부족한데, 테이블 스페이스가 이 경로를 사용하여 중지되었으며(다른 경로에서 스트라이프 세트를 작성하여) 추가 스토리지가 경로에 제공된 경우에 유익합니다. 이와 같은 경우, 새 경로가 추가되지 않지만 이전에 포함되지 않은 스트라이프 세트에 해당 스토리지 경로를 포함시키기 위해 재조정이 시도됩니다.

- 자동 스토리지 테이블 스페이스의 재조정이 진행되는 동안 자동 *크기* 조정이 계속 발생할 수 있습니다.
- MANAGED BY AUTOMATIC STORAGE절에 의해 자동 스토리지에 대해 DMS 테이블 스페이스가 사용 가능한 경우, 테이블 스페이스는 사용자 정의(자동이 아닌 스토리지) 컨테이너의 하나 이상의 스트라이프 세트와 자동 스토리지 컨테이너의 하나 이상의 스트라이프 세트를 갖게 됩니다. 테이블 스페이스 재조정을 수행하면 (REBALANCE절을 사용하여) 모든 사용자 정의 컨테이너가 제거됩니다. 데이터베이스 관리 프로그램은 기존의 자동 스토리지 컨테이너를 확장하거나 자동 스토리지 컨테이너를 새로 작성하여 사용자 정의 컨테이너에서 제거되는 데이터를 보유합니다.
- *호환성*: 버전 8 이전 버전과의 호환성을 위해 다음 키워드를 사용할 수 있습니다.
  - DBPARTITIONNUM 대신 NODE를 사용할 수 있습니다.
  - DBPARTITIONNUMS 대신 NODES를 사용할 수 있습니다.

예:

*예 1:*디바이스를 PAYROLL 테이블 스페이스에 추가합니다.

```
ALTER TABLESPACE PAYROLL
  ADD (DEVICE '/dev/rhdisk9' 10000)
```

*예 2:*ACCOUNTING 테이블 스페이스에 대한 프리페치 *크기*와 입출력 오버헤드를 변경합니다.

```
ALTER TABLESPACE ACCOUNTING
  PREFETCHSIZE 64
  OVERHEAD 19.3
```

예 3: 테이블 스페이스 TS1을 작성한 후, 모든 컨테이너의 페이지 수가 2000이 되도록 *크기*를 조정합니다. (이 *크기* 조정을 수행할 세 가지의 다른 ALTER TABLESPACES문이 제공됩니다.)

```
CREATE TABLESPACE TS1
  MANAGED BY DATABASE
  USING (FILE '/conts/cont0' 1000,
         DEVICE '/dev/rcont1' 500,
         FILE 'cont2' 700)
```

```
ALTER TABLESPACE TS1
  RESIZE (FILE '/conts/cont0' 2000,
            DEVICE '/dev/rcont1' 2000,
            FILE 'cont2' 2000)
```

또는

```
ALTER TABLESPACE TS1
  RESIZE (ALL 2000)
```

또는

```
ALTER TABLESPACE TS1
  EXTEND (FILE '/conts/cont0' 1000,
            DEVICE '/dev/rcont1' 1500,
            FILE 'cont2' 1300)
```

예 4: DATA_TS 테이블 스페이스에서 모든 컨테이너를 1000 페이지씩 확장합니다.

```
ALTER TABLESPACE DATA_TS
  EXTEND (ALL 1000)
```

예 5: INDEX_TS 테이블 스페이스에서 모든 컨테이너의 *크기*를 100메가바이트(MB)로 조정합니다.

```
ALTER TABLESPACE INDEX_TS
  RESIZE (ALL 100 M)
```

예 6: 세 개의 새 컨테이너를 추가합니다. 첫 번째 컨테이너를 확장한 다음 두 번째 컨테이너의 *크기*를 조정하십시오.

```
ALTER TABLESPACE TS0
  ADD (FILE 'cont2' 2000, FILE 'cont3' 2000)
  ADD (FILE 'cont4' 2000)
  EXTEND (FILE 'cont0' 100)
  RESIZE (FILE 'cont1' 3000)
```

예 7: 테이블 스페이스 TSO는 데이터베이스 파티션 0, 1 및 2에 존재합니다. 데이터베이스 파티션 0에 새 컨테이너를 추가하십시오. 데이터베이스 파티션 1의 모든 컨테이너를 확장하십시오. 명시적으로 지정된 데이터베이스 파티션 0과 1이 아닌 다른 모든 데이터베이스 파티션의 컨테이너의 *크기*를 조정합니다.

```
ALTER TABLESPACE TS0
  ADD (FILE 'A' 200) ON DBPARTITIONNUM (0)
  EXTEND (ALL 200) ON DBPARTITIONNUM (1)
  RESIZE (FILE 'B' 500)
```

이 예에서 RESIZE절은 디폴트 컨테이너 절이며, 다른 조작은 명시적으로 데이터베이스 파티션 0 및 1로 보내지기 때문에 RESIZE절은 데이터베이스 파티션 2에서 실행됩니다. 그러나 데이터베이스 파티션 0과 1만 있을 경우에는 명령문은 성공하지만 디폴트 컨테이너가 지정되었으나 사용되지 않았음을 알리는 경고(SQL1758W)가 리턴됩니다.

예 8: 테이블 스페이스 DMS_TS1에 대한 자동 크기 조정 옵션을 사용 가능하게 하고 최대 크기를 256MB로 설정합니다.

```
ALTER TABLESPACE DMS_TS1
  AUTORESIZE YES MAXSIZE 256 M
```

예 9 테이블 스페이스 AUTOSTORE1에 대한 자동 크기 조정 옵션을 사용 가능하게 하고 증가 비율을 5%로 변경합니다.

```
ALTER TABLESPACE AUTOSTORE1
  AUTORESIZE YES INCREASESIZE 5 PERCENT
```

예 10: 자동 크기 조정 가능 테이블 스페이스 MY_TS의 증가 비율을 512KB로 변경하고 최대 크기를 가능한 크게 설정합니다.

```
ALTER TABLESPACE MY_TS
  INCREASESIZE 512 K MAXSIZE NONE
```

예 11: 데이터베이스 관리 대상 테이블 스페이스 DMS_TS10에 대한 자동 스토리지 사용 가능

```
ALTER TABLESPACE DMS_TS10
  MANAGED BY AUTOMATIC STORAGE
```

예 12: ALTER DATABASE 문이 현재 연결된 데이터베이스에서 경로 /db2/filesystem1 및 /db2/filesystem2을 제거하였습니다. PRODTS1, PRODTS2 및 PRODTS3 테이블 스페이스만 제거된 경로를 사용합니다. 다음 테이블 스페이스를 재조정하십시오. 세 가지 ALTER TABLESPACE 문이 사용되어야 합니다.

```
ALTER TABLESPACE PRODTS1 REBALANCE
ALTER TABLESPACE PRODTS2 REBALANCE
ALTER TABLESPACE PRODTS3 REBALANCE
```

예 13: 데이터베이스 관리 대상 테이블 스페이스 DATA1에 대해 자동 스토리지가 사용 가능하도록 설정하고 테이블 스페이스에서 자동이 아닌 기존 스토리지 컨테이너를 모두 제거하십시오. 첫 번째 명령문은 두 번째 명령문이 실행되기 전에 커미트해야 합니다.

```
ALTER TABLESPACE DATA1 MANAGED BY AUTOMATIC STORAGE
ALTER TABLESPACE DATA1 REBALANCE
```

예 14: 재개 가능한 스토리지 속성으로 자동 스토리지 테이블 스페이스의 Extent 이동을 트리거하여 컨테이너의 크기를 10MB로 줄이십시오.

```
ALTER TABLESPACE REDUCE 10 M
```

예 15: 재개 가능한 스토리지 속성으로 비자동 스토리지 테이블 스페이스의 EXTENT 이동을 트리거해서 각 컨테이너의 크기를 10MB로 줄이십시오.

```
ALTER TABLESPACE LOWER HIGH WATER MARK
ALTER TABLESPACE REDUCE (ALL CONTAINERS 10 M)
```

ALTER THRESHOLD

ALTER THRESHOLD문은 임계값 정의를 변경합니다.

호출

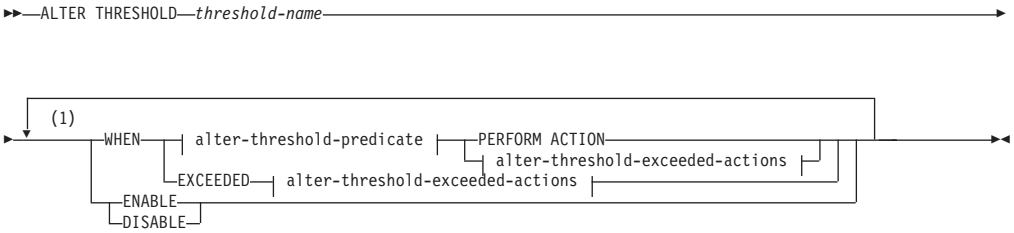
이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

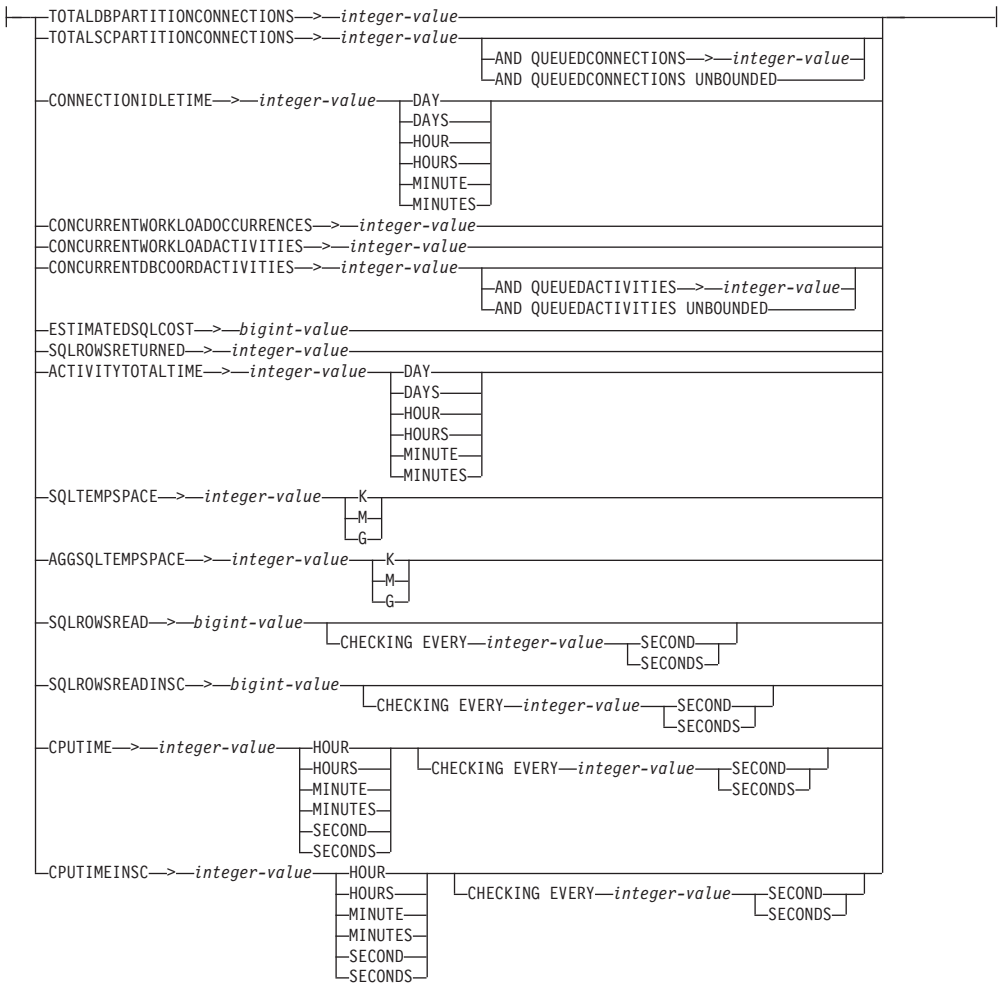
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- SQLADM 권한(모든 변경 절이 COLLECT절인 경우에만 해당)
- WLMADM 권한
- DBADM 권한

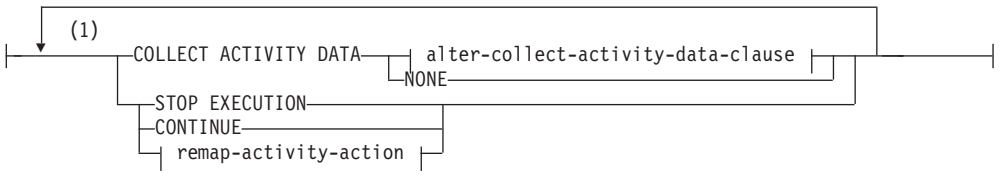
구문



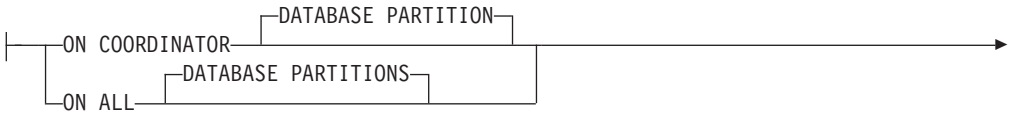
alter-threshold-predicate:

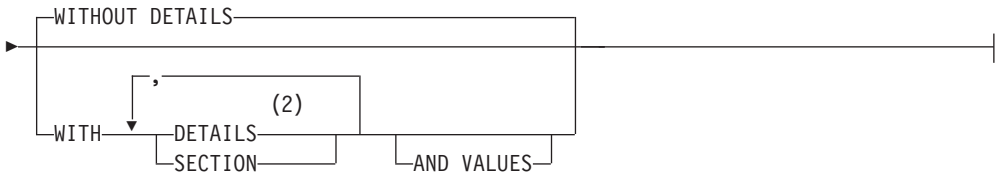


alter-threshold-exceeded-actions:

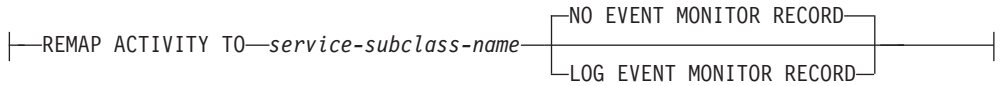


alter-collect-activity-data-clause:





remap-activity-action:



주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

2 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

threshold-name

변경될 임계값을 식별합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL ID(일반 또는 분리 ID)입니다. 이름은 현재 서버에서 기존 임계값을 고유하게 식별해야 합니다(SQLSTATE 42704).

WHEN *alter-threshold-predicate* **또는 WHEN EXCEEDED**

임계값 술어 조건에서 기존 상한 값을 새 상한 값으로 교체합니다. 임계값 조건은 다른 조건으로 변경될 수 없습니다.

PERFORM ACTION

임계값 술어 조건 값을 변경할 때 임계값 초과 조치가 변경되지 않음을 지정합니다.

EXCEEDED

변경된 임계값에 대해 원래 지정된 동일한 임계값 술어를 보존하도록 지정합니다.

alter-threshold-predicate

TOTALDBPARTITIONCONNECTIONS > *integer-value*

이 조건은 데이터베이스 파티션에서 동시에 실행할 수 있는 코디네이터 연결 수에 대한 상한을 정의합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 새 코디네이터 연결이 연결되지 않음을 의미합니다. 현재 실행 중이거나 큐에서 대기 상태인 모든 연결이 계속됩니다.

TOTALSCPARTITIONCONNECTIONS > *integer-value*

이 조건은 특정 서비스 수퍼 클래스의 데이터베이스 파티션에서 동시에 실행할 수 있는 코디네이터 연결 수에 대한 상한을 정의합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 새 연결이 서버 클래스를 조인하지 않음을 의미합니다. 현재 실행 중이거나 큐에서 대기 상태인 모든 연결이 계속됩니다.

AND QUEUEDCONNECTIONS > *integer-value* **또는 AND QUEUEDCONNECTIONS UNBOUNDED**

최대 코디네이터 연결 수를 초과할 때 큐 *크기*를 지정합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 코디네이터 연결이 큐에 대기되지 않음을 의미합니다. UNBOUNDED를 지정하면 지정된 최대 코디네이터 연결 수를 초과하는 모든 연결을 큐에 넣지만 *threshold-exceeded-actions*는 실행되지 않습니다.

CONNECTIONIDLETIME > *integer-value* **DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES**

이 조건은 데이터베이스 관리 프로그램이 연결이 유휴 상태로 유지되도록 허용하는 시간의 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820). 유효한 지속기간 키워드를 사용하여 *integer-value*에 적절한 시간 단위를 지정하십시오. 이 조건은 코디네이터 데이터베이스 파티션에서 논리적으로 시행됩니다.

CONNECTIONIDLETIME 임계값을 사용하여 STOP EXECUTION 조치를 지정하는 경우 응용프로그램의 연결은 임계값이 초과될 때 삭제됩니다. 응용프로그램이 데이터 서버에 더 이상 연결되지 않기 때문에 응용프로그램이 데이터 서버에 액세스하기 위한 다음 시도에서는 SQLSTATE 5U026이 수신되지 않습니다.

이 임계값의 최대값은 2,147,483,640초입니다. 2,147,483,640초보다 많은 초 값을 지정하면 이 초 수로 설정됩니다.

CONCURRENTWORKLOADOCCURRENCES > *integer-value*

이 조건은 각 데이터베이스 파티션에서 워크로드에 대한 동시 어커런스 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

CONCURRENTWORKLOADACTIVITIES > *integer-value*

이 조건은 각 데이터베이스 파티션에서 워크로드에 대한 동시 코디네이터 활동 및 중첩 활동 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

중첩된 각 활동은 다음 조건을 충족해야 합니다.

- 인식되는 코디네이터 활동이어야 합니다. 인식되는 활동 유형 내에 속하지 않는 중첩된 코디네이터 활동은 계산되지 않습니다. 마찬가지로, 리모트 노드 요청과 같은 중첩된 서브에이전트 활동은 계산되지 않습니다.
- SQL문을 발행하는 사용자 작성 프로시저와 같이, 사용자 논리를 통해 직접 호출해야 합니다.

결과적으로, SYSIBM, SYSFUN 또는 SYSPROC 스키마의 DB2 유틸리티 또는 루틴 호출 하에 자동으로 시작된 중첩된 코디네이터 활동은 이 임계값에 지정된 상한에 계산되지 않습니다.

제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 생성된 활동과 같은 내부 SQL 활동도 이 임계값에서 계산되지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

CONCURRENTDBCOORDACTIVITIES > *integer-value*

이 조건은 지정된 도메인의 모든 데이터베이스 파티션에서 동시에 실행할 수 있는 인식된 데이터베이스 코디네이터 활동 수에 대한 상한을 정의합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 새 데이터베이스 코디네이터 활동이 실행되지 않음을 의미합니다. 현재 실행 중이거나 큐에서 대기 상태인 모든 데이터베이스 코디네이터 활동이 계속됩니다.

중요사항: CONCURRENTDBCOORDACTIVITIES 임계값의 결과 외부 개입이 필요한 큐 기반 경쟁이 발생할 수 있습니다. 자세한 정보는 "CONCURRENTDBCOORDACTIVITIES 임계값"을 참조하십시오.

AND QUEUEDACTIVITIES > *integer-value* **또는 AND QUEUEDACTIVITIES UNBOUNDED**

최대 데이터베이스 코디네이터 활동 수를 초과할 때 큐 크기를 지정합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 데이터베이스 코디네이터 활동이 큐에 대기되지 않음을 의미합니다. UNBOUNDED를 지정하면 지정된 최대 데이터베이스 코디네이터 활동 수를 초과하는 모든 데이터베이스 코디네이터 활동을 큐에 넣지만 *threshold-exceeded-actions*는 실행되지 않습니다.

ESTIMATEDSQLCOST > *bigint-value*

이 조건은 활동의 옵티마이저 지정 비용(timeron 단위)의 상한을 정의합니다. 이 값은 0이 아닌 양의 큰 정수가 될 수 있습니다(SQLSTATE 42820). 이 조건은 코디네이터 데이터베이스 파티션에서 시행됩니다. 이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 데이터 처리 언어(DML) 유형의 코디네이터 활동.
- 사용자 논리를 통해 호출되는 중첩된 DML 활동. 결과적으로, 데이터베이스 관리 프로그램에 의해 시작될 수 있는 DML 활동(예: 유틸리티, 프로시저 또

는 내부 SQL)은 이 조건에 의해 추적되지 않습니다(간접적으로 추적되는 경우 상위의 추정에 비용이 포함되는 경우는 제외).

SQLROWSRETURNED > *integer-value*

이 조건은 응용프로그램 서버(AS)에서 클라이언트 응용프로그램으로 리턴되는 행 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 정수가 될 수 있습니다(SQLSTATE 42820). 이 조건은 코디네이터 데이터베이스 파티션에서 시행됩니다. 이 조건에 의해 추적되는 활동은 다음과 같습니다.

- DML 유형의 코디네이터 활동.
- 사용자 논리를 통해 파생되는 중첩된 DML 활동. 유틸리티, 프로시저 또는 내부 SQL을 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건의 영향을 받지 않습니다.

프로시저 내에서 리턴된 결과 세트는 개별적 활동으로 별도로 처리됩니다. 프로시저 자체에서 리턴되는 행의 집계는 없습니다.

ACTIVITYTOTALTIME > *integer-value* **DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES**

이 조건은 활동이 큐에서 대기한 시간을 포함하여, 데이터베이스 관리 프로그램이 연결 실행을 허용할 시간의 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820). 유효한 지속기간 키워드를 사용하여 *integer-value*에 적절한 시간 단위를 지정하십시오. 이 조건은 코디네이터 데이터베이스 파티션에서 논리적으로 시행됩니다.

이 임계값의 최대값은 2,147,483,640초입니다. 2,147,483,640초보다 많은 초 값을 지정하면 이 초 수로 설정됩니다.

SQLTEMPSPACE > *integer-value* **K | M | G**

이 조건은 데이터베이스 파티션에서 시스템 임시 테이블 스페이스 크기에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

integer-value K(대문자 또는 소문자)를 지정한 경우, 최대 크기는 1024 x *integer-value*입니다. *integer-value M*을 지정한 경우, 최대 크기는 1,048,576 x *integer-value*입니다. *integer-value G*를 지정한 경우, 최대 크기는 1 073 741 824 x *integer-value*입니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당 서브에이전트 작업(서브섹션 실행).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행). 유틸리티, 프로시저 또는 내부 SQL을 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건의 영향을 받지 않습니다.

AGGSQLTEMPSPACE > *integer-value* **K | M | G**

이 조건은 데이터베이스 파티션의 정의 도메인에서 모든 활동 사이에 소비될 수 있는 시스템 임시 스페이스의 최대량을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

integer-value K(대문자 또는 소문자)를 지정한 경우, 최대 크기는 1024 x *integer-value*입니다. *integer-value* M을 지정한 경우, 최대 크기는 1,048,576 x *integer-value*입니다. *integer-value* G를 지정한 경우, 최대 크기는 1 073 741 824 x *integer-value*입니다.

이 조건에 의해 추적되는 집계에 제공되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(예: 서브섹션 실행과 같은). 유틸리티, 프로시저 또는 내부 SQL문을 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건의 영향을 받지 않습니다.

SQLROWSREAD > *bigint-value*

이 조건은 특정 데이터베이스 파티션에서 수명 동안 활동에 의해 읽을 수 있는 행 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 큰 정수가 될 수 있습니다(SQLSTATE 42820). 읽혀진 행 수는 리턴된 행 수와 다릅니다. 리턴된 행 수는 SQLROWSRETURNED 조건의 제어를 받습니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.
- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. 임계값은 각 요청(페치 조작과 같은)의 끝에서, CHECKING 절에 정의된 간격으로 점검됩니다. CHECKING 절은 임계값 위반이 발견되지 않을 수 있는 시간의 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

SQLROWSREADINSC > *bigint-value*

이 조건은 서비스 서브클래스에서 실행 중인 동안 특정 데이터베이스 파티션에서 활동에 의해 읽을 수 있는 행 수에 대한 상한을 정의합니다. 지정된 서비스 서브클래스에서 실행하기 전에 읽은 행 수는 계산되지 않습니다. 이 값은 0이 아닌 양의 큰 정수가 될 수 있습니다(SQLSTATE 42820). 읽혀진 행 수는 리턴된 행 수와 다릅니다. 리턴된 행 수는 SQLROWSRETURNED 조건의 제어를 받습니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.
- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. 임계값은 각 요청(페치 조작과 같은)의 끝에서, CHECKING 절에 정의된 간격으로 점검됩니다. CHECKING 절은 임계값 위반이 발견되지 않을 수 있는 시간의 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

CPUTIME > *integer-value* **DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS**

이 조건은 특정 데이터베이스 파티션에서 수명 동안 활동에 소비될 수 있는 프로세서 시간에 대한 상한을 정의합니다. 이 임계값에 의해 추적되는 프로세서 시간은 활동이 실행을 시작하는 시간부터 측정됩니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.

- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.
- 유형 CALL의 활동. CALL 활동의 경우, 스토어드 프로시저에 대해 추적되는 프로세서 시간에는 하위 활동이나 분리 모드 프로세스에 사용되는 프로세서 시간이 포함되지 않습니다. 임계값 조건은 사용자 논리에서 데이터베이스 엔진으로 리턴될 때만 점검됩니다. 예를 들어, 트러스트된 루틴 실행 중, 임계값 조건은 루틴이 데이터베이스 엔진에 요청을 발행하는 경우에만 점검됩니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. CPUTIME 임계값의 세분화도는 대략 이 숫자에 활동의 병렬 처리 수준을 곱한 값입니다. 예를 들어, 임계값이 60초마다 점검되고 병렬 처리 수준이 2이면 활동은 임계값 위반이 발견되기 전에 1분 대신 여분의 프로세서 시간 2분을 사용할 수 있습니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

CPUTIMEINSC > *integer-value* **DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS**

이 조건은 서비스 서브클래스에서 실행 중인 동안 특정 데이터베이스 파티션에서 활동에 소비될 수 있는 프로세스 시간에 대한 상한을 정의합니다. 이 임계값에 의해 추적되는 프로세서 시간은 활동이 임계값 도메인에서 식별된 서비스 서브클래스에서 실행을 시작하는 시간부터 측정됩니다. 해당 시점 이전에 사용된 프로세서 시간은 이 임계값에 의해 부과되는 한계에 대해 계산되지 않습니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.
- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

- 유형 CALL의 활동. CALL 활동의 경우, 스토어드 프로시저에 대해 추적되는 프로세서 시간에는 하위 활동이나 분리 모드 프로세스에 사용되는 프로세서 시간이 포함되지 않습니다. 임계값 조건은 사용자 논리에서 데이터베이스 엔진으로 리턴될 때만 점검됩니다. 예를 들어, 트러스트된 루틴 실행 중, 임계값 조건은 루틴이 데이터베이스 엔진에 요청을 발행하는 경우에만 점검됩니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. CPUTIMEINSC 임계값의 세분화도는 대략 이 숫자에 활동의 병렬 처리 수준을 곱한 값입니다. 예를 들어, 임계값이 60초마다 점검되고 병렬 처리 수준이 2이면 활동은 임계값 위반이 발견되기 전에 1분 대신 여분의 프로세서 시간 2분을 사용할 수 있습니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

alter-threshold-exceeded-actions

조건을 초과할 때 수행할 조치를 지정합니다. 조건을 초과할 때마다 이벤트가 모든 활성 상태의 임계값 위반 이벤트 모니터에에 기록됩니다.

COLLECT ACTIVITY DATA

임계값을 초과한 각 활동에 대한 데이터가 활동 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

활동 데이터가 활동 코디네이터의 데이터베이스 파티션에서만 수집되도록 지정합니다.

ON ALL DATABASE PARTITIONS

활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. 예측적 임계값의 경우, 초과된 임계값에 CONTINUE 조치도 지정하는 경우에만 모든 파티션에서 활동 정보가 수집됩니다. SECTION이 지정된 경우 활동 세부사항 및 섹션 환경도 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 반응적 임계값의 경우, ON ALL DATABASE PARTITIONS절은 효과 및 활동, 활동 세부사항, 섹션 정보가 없으며, 값은 항상 코디네이터 파티션에서만 수집됩니다.

WITHOUT DETAILS

작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가

활동 실행 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

NONE
임계값을 초과하는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

STOP EXECUTION
활동 실행이 중지되고 오류가 리턴됩니다(SQLSTATE 5U026).

CONTINUE
활동 실행이 중지되지 않습니다. 조건에 큐도 있는 경우, 이 옵션을 사용하면 큐에서 대기되어 큐 *크기*를 넘어 확장됩니다.

remap-activity-action

REMAP ACTIVITY TO *service-subclass-name*
활동은 *service-subclass-name*에 맵핑됩니다.활동 실행이 중지되지 않습니다. 이 조치는 CPUTIMEINSC 및 SQLROWSREADINSC 임계값과 같은 in-service-class 임계값에만 유효합니다(SQLSTATE 5U037). *service-subclass-name*은 임계값과 연관되는 동일한 수퍼 클래스 아래에 있는 기존 서비스 서브클래스를 식별해야 합니다(SQLSTATE 5U037). *service-subclass-name*은 임계값의 연관된 서비스 서브클래스와 동일할 수 없습니다(SQLSTATE 5U037).

NO EVENT MONITOR RECORD
임계값 위반 레코드가 기록되지 않도록 지정합니다.

LOG EVENT MONITOR RECORD
THRESHOLD VIOLATIONS 이벤트 모니터가 존재하고 활성 상태인 경우 임계값 위반 레코드가 이 이벤트 모니터에 기록되도록 지정합니다.

ENABLE 또는 DISABLE

데이터베이스 관리 프로그램에 의한 사용에 임계값이 사용 가능한지 여부를 지정합니다.

ENABLE

데이터베이스 활동 실행을 제한하기 위해 데이터베이스 관리 프로그램에서 임계값이 사용됩니다. 현재 실행 중인 데이터베이스 활동은 이 임계값의 제한 없이 계속 실행됩니다.

DISABLE

데이터베이스 활동 실행을 제한하기 위해 데이터베이스 관리 프로그램에서 임계값이 사용되지 않습니다. 새 데이터베이스 활동은 이 임계값으로 제한되지 않습니다. 큐에 대한 임계값(예: TOTALSCPARTITIONCONNECTIONS 또는 CONCURRENTDBCOORDACTIVITIES)은 사용 불가능하게 된 후에 삭제해야 합니다.

주

- 임계값의 새 값은 변경 조작이 커미트된 후 실행을 시작하는 DB2 활동에만 영향을 줍니다.

예 :

임계값 MAXBIGQUERIESCONCURRENCY를 두 개가 아닌 최대 세 개의 활동으로 변경하십시오.

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY
  WHEN CONCURRENTDBCOORDACTIVITIES > 3
  STOP EXECUTION
```

큐에 대한 임계값이므로 다음과 같이 사용 불가능 상태가 아니면 임계값을 삭제할 수 없습니다.

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY DISABLE
```

ALTER TRUSTED CONTEXT

ALTER TRUSTED CONTEXT문은 현재 서버에서 트러스트된 컨텍스트의 정의를 수정합니다.

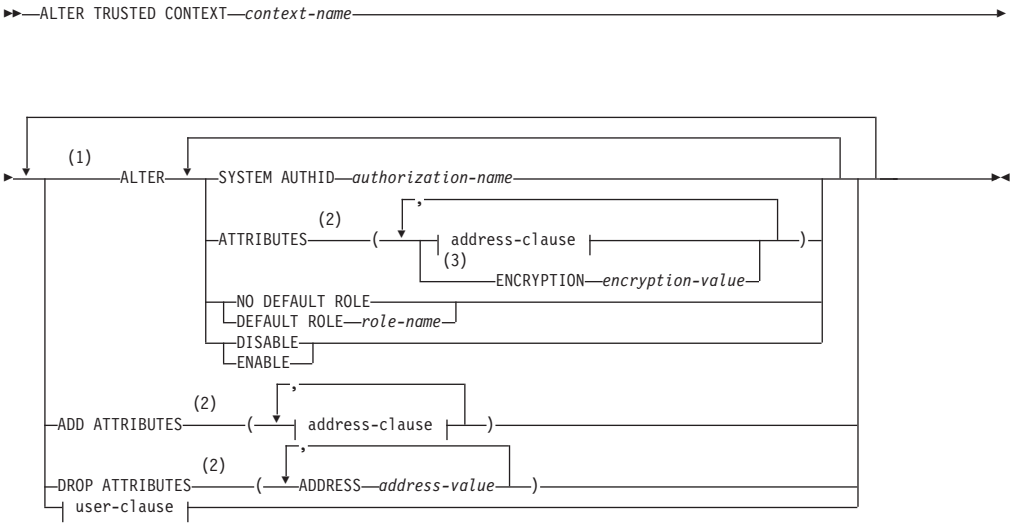
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

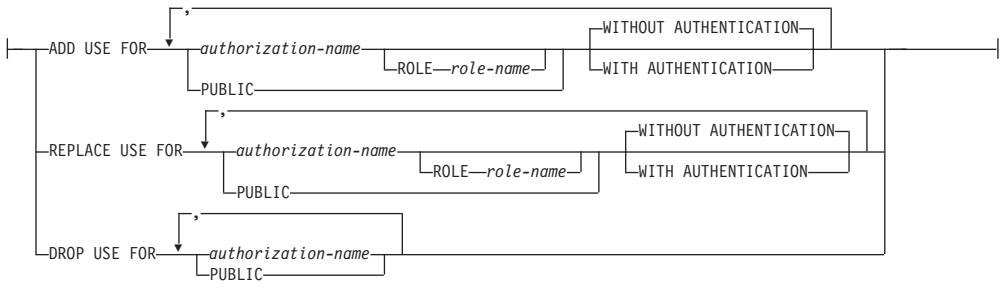
명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



address-clause:

user-clause:



주:

1 ATTRIBUTES, DEFAULT ROLE, ENABLE 및 WITH USE 절 각각은 최대 한 번 지정할 수 있습니다(SQLSTATE 42614).

2 각각의 속성 이름과 해당 이름은 고유해야 합니다(SQLSTATE 4274D).

3 ENCRYPTION은 두 번 이상 지정할 수 없습니다(SQLSTATE 42614). 그러나 지정된 ADRESS마다 WITH ENCRYPTION을 지정할 수 있습니다.

설명

context-name

변경될 트러스트된 컨텍스트를 식별합니다. 이는 한 부분으로 된 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *context-name*은 현재 서버에 존재하는 트러스트된 컨텍스트를 식별해야 합니다(SQLSTATE 42704).

ALTER

트러스트된 컨텍스트의 옵션 및 속성을 변경합니다.

SYSTEM AUTHID *authorization-name*

컨텍스트가 시스템 권한 부여 ID *authorization-name*에 의해 설정된 연결임을 지정합니다. 이 ID는 기존의 트러스트된 컨텍스트와 연관될 수 없습니다 (SQLSTATE 428GL). 이는 명령문의 권한 부여 ID가 될 수 없습니다 (SQLSTATE 42502).

ATTRIBUTES (...)

트러스트된 컨텍스트가 정의되고 수정될 하나 이상의 연결 트러스트 속성 목록을 지정합니다. 지정된 속성의 기존 값은 새 값으로 교체됩니다. 속성이 현재 트러스트된 컨텍스트 정의의 일부가 아니면 오류가 리턴됩니다(SQLSTATE 4274C). 지정되지 않은 속성은 이전 값을 보유합니다.

ADDRESS *address-value*

클라이언트가 데이터베이스 서버와 통신하기 위해 사용하는 실제 통신 주소를 지정합니다. 지원되는 유일한 프로토콜은 TCP/IP입니다. 지정된 트러스트된 컨텍스트의 이전 ADDRESS 값은 제거됩니다. ADDRESS 속성을 여러 번 지정할 수 있지만, 각 *address-value* 쌍은 속성 세트에 대해 고유해야 합니다(SQLSTATE 4274D).

트러스트된 연결을 설정할 때 트러스트된 컨텍스트의 ADDRESS 속성에 복수의 값이 정의되면, 연결에 사용되는 주소가 트러스트된 컨텍스트의 ADDRESS 속성에 대해 정의된 값 중 하나와 일치하는 경우 후보 연결이 이 속성과 일치하는 것으로 간주됩니다.

address-value

ADDRESS 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다. *address-value*는 IPv4 주소, IPv6 주소 또는 보안 도메인 이름이어야 합니다.

- IPv4 주소는 앞 공백을 포함할 수 없으며 점 10진 주소로 나타냅니다. IPv4 주소의 예로 9.112.46.111이 있습니다. 값 'localhost' 또는 해당되는 표현 '127.0.0.1'은 결과적으로 일치하지 않습니다. 대신 호스트의 실제 IPv4 주소를 지정해야 합니다.
- IPv6 주소는 앞 공백을 포함할 수 없으며 콜론 16진 주소로 나타냅니다. IPv6 주소의 예로 2001:0DB8:0000:0000:0008:0800:200C:417A가 있습니다. IPv4 맵핑 IPv6 주소(예, ::ffff:192.0.2.128)는 결과적으로 일치하지 않습니다. 마찬가지로, 'localhost' 또는 해당되는 IPv6 축약 표현 '::1'은 결과적으로 일치하지 않습니다.
- 도메인 이름은 도메인 이름 서버에 의해 결과 IPv4 또는 IPv6 주소가 판별되는 IP 주소로 변환됩니다. 도메인 이름의 예로 corona.torolab.ibm.com이 있습니다. 도메인 이름이 IP 주소로 변환될 때 이 변환의 결과는 하나 이상의 IP 주소로 설정될 수 있습니다. 이 경우, 연결이 시작되는 IP 주소가 도메인 이름이 변환된 IP 주소 중 하나와 일치하면 트러스트된 컨텍스트 오브젝트의 ADDRESS 속성과 일치함을 의미합니다. 트러스트된 컨텍스트 오브젝트를 작성할 때 정적 IP 주소 대신 ADDRESS 속성의 도메인 이름 값을 제공하는 것이 좋습니다(특히 DHCP(Dynamic Host Configuration Protocol) 환경에서). DHCP를 사용하는 경우, 디바이스는 네트워크에 연결할 때마다 다른 IP 주소를 가질 수 있습니다. 따라서, 트러스트된 컨텍스트 오브젝트의 ADDRESS 속성에 대해 정적 IP 주소가 제공되는 경우 일부 디바이스는 의도하지 않게 트러스트된 연결을 획득할 수 있습니다. 트러스트된 컨텍스트 오브젝트의 ADDRESS 속성에 도메인 이름을 제공하면 DHCP 환경에서 이러한 문제점이 방지됩니다.

WITH ENCRYPTION *encryption-value*

특정 *address-value*에 대한 네트워크 암호화 또는 데이터 스트림 암호화의 최소 레벨을 지정합니다. 이 *encryption-value*는 특정 *address-value*에 대한 전역 ENCRYPTION 속성을 대체합니다.

encryption-value

특정 *address-value*에 대한 ENCRYPTION 트러스트 속성과

연관될 값을 포함하는 문자열 상수를 지정합니다.
*encryption-value*는 다음 중 하나여야 합니다(SQLSTATE 42615).

- NONE: 암호화의 특정 레벨이 필요하지 않습니다.
- LOW: 최소한의 가벼운 암호화가 필요합니다. 데이터베이스 관리 프로그램에 대한 인증 유형은 수신 연결이 이 특정 주소에 대한 암호화 설정과 일치할 경우 DATA_ENCRYPT 여야 합니다.
- HIGH, 수신 연결이 이 특정 주소에 대한 암호화 설정과 일치하는 경우 DB2 클라이언트와 DB2 서버 사이의 데이터 통신에 SSL(Secure Sockets Layer) 암호화를 사용해야 합니다.

ENCRYPTION *encryption-value*

네트워크 암호화 또는 데이터 스트림 암호화의 최소 레벨을 지정합니다. 디폴트는 NONE입니다.

encryption-value

특정 *address-value*에 대한 ENCRYPTION 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다. *encryption-value*는 다음 중 하나여야 합니다(SQLSTATE 42615).

- NONE: 수신 연결이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치하도록 하는 데 암호화의 특정 레벨이 필요하지 않습니다.
- LOW: 최소한의 가벼운 암호화가 필요합니다. 데이터베이스 관리 프로그램에 대한 인증 유형은 수신 연결이 이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치할 경우 DATA_ENCRYPT 여야 합니다.
- HIGH, 수신 연결이 이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치할 경우 DB2 클라이언트와 DB2 서버 사이의 데이터 통신에 SSL(Secure Sockets Layer) 암호화를 사용해야 합니다.

ENCRYPTION 트러스트 속성에 대한 세부사항은 『CREATE TRUSTED CONTEXT』를 참조하십시오.

NO DEFAULT ROLE 또는 DEFAULT ROLE *role-name*

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결과 디폴트 역할이 연관되는지 여부를 지정합니다. 해당 컨텍스트에 대한 트러스트된 연결이 활성 상태이면, 변경은 다음 사용자 전환 요청 시 또는 새 연결 요청 시 적용됩니다.

NO DEFAULT ROLE

트러스트된 컨텍스트가 디폴트 역할을 가지고 있지 않음을 지정합니다.

DEFAULT ROLE *role-name*

*role-name*이 트러스트된 컨텍스트의 디폴트 역할임을 지정합니다. *role-name*은 현재 서버에 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). 이 역할은 사용자가 트러스트된 컨텍스트 정의의 일부로 정의된 사용자 특정 역할을 가지고 있지 않은 경우 이 트러스트된 컨텍스트를 기초로, 트러스트된 연결에서 사용자에 대해 사용됩니다.

ENABLE 또는 DISABLE

트러스트된 컨텍스트가 사용 가능 또는 사용 불가능한지 여부를 지정합니다.

ENABLE

트러스트된 컨텍스트가 사용 가능함을 지정합니다.

DISABLE

트러스트된 컨텍스트가 사용 불가능함을 지정합니다. 사용 불가능한 트러스트된 컨텍스트는 트러스트된 연결이 설정될 때 고려되지 않습니다.

ADD ATTRIBUTES

트러스트된 컨텍스트가 정의되는 하나 이상의 추가적인 트러스트 속성의 목록을 지정합니다.

ADDRESS *address-value*

클라이언트가 데이터베이스 서버와 통신하기 위해 사용하는 실제 통신 주소를 지정합니다. 지원되는 유일한 프로토콜은 TCP/IP입니다. ADDRESS 속성을 여러 번 지정할 수 있지만, 각 *address-value* 쌍은 속성 세트에 대해 고유해야 합니다(SQLSTATE 4274D).

트러스트된 연결을 설정할 때 트러스트된 컨텍스트의 ADDRESS 속성에 복수의 값이 정의되면, 연결에 사용되는 주소가 트러스트된 컨텍스트의 ADDRESS 속성에 대해 정의된 값 중 하나와 일치하는 경우 후보 연결이 이 속성과 일치하는 것으로 간주됩니다.

address-value

ADDRESS 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다. *address-value*는 IPv4 주소, IPv6 주소 또는 보안 도메인 이름이어야 합니다.

- IPv4 주소는 앞 공백을 포함할 수 없으며 점 10진 주소로 나타냅니다. IPv4 주소의 예로 9.112.46.111이 있습니다. 값 'localhost' 또는 해당되는 표현 '127.0.0.1'은 결과적으로 일치하지 않습니다. 대신 호스트의 실제 IPv4 주소를 지정해야 합니다.
- IPv6 주소는 앞 공백을 포함할 수 없으며 콜론 16진 주소로 나타냅니다. IPv6 주소의 예로

2001:0DB8:0000:0000:0008:0800:200C:417A가 있습니다. IPv4 맵핑 IPv6 주소(예, ::ffff:192.0.2.128)는 결과적으로 일치하지 않습니다. 마찬가지로, 'localhost' 또는 해당되는 IPv6 축약 표현 '::1'은 결과적으로 일치하지 않습니다.

- 도메인 이름은 도메인 이름 서버에 의해 결과 IPv4 또는 IPv6 주소가 판별되는 IP 주소로 변환됩니다. 도메인 이름의 예로 corona.torolab.ibm.com이 있습니다.

WITH ENCRYPTION *encryption-value*

특정 *address-value*에 대한 네트워크 암호화 또는 데이터 스트림 암호화의 최소 레벨을 지정합니다. 이 *encryption-value*는 특정 *address-value*에 대한 전역 ENCRYPTION 속성을 대체합니다.

encryption-value

특정 *address-value*에 대한 ENCRYPTION 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다.

*encryption-value*는 다음 중 하나여야 합니다(SQLSTATE 42615).

- NONE: 암호화의 특정 레벨이 필요하지 않습니다.
- LOW: 최소한의 가벼운 암호화가 필요합니다. 데이터베이스 관리 프로그램에 대한 인증 유형은 수신 연결이 이 특정 주소에 대한 암호화 설정과 일치할 경우 DATA_ENCRYPT여야 합니다.
- HIGH, 수신 연결이 이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치할 경우 DB2 클라이언트와 DB2 서버 사이의 데이터 통신에 SSL(Secure Sockets Layer) 암호화를 사용해야 합니다.

DROP ATTRIBUTES

하나 이상의 속성이 트러스트된 컨텍스트의 정의에서 삭제되도록 지정합니다. 속성 및 속성 값 쌍이 현재 트러스트된 컨텍스트 정의의 일부가 아니면 오류가 리턴됩니다(SQLSTATE 4274C).

ADDRESS *address-value*

식별된 통신 주소가 트러스트된 컨텍스트의 정의에서 삭제되도록 지정합니다. *address-value*는 기존 ADDRESS 트러스트 속성의 값을 포함하는 문자열 상수를 지정합니다.

ADD USE FOR

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결을 사용할 수 있는 추가 사용자를 지정합니다. 트러스트된 컨텍스트의 정의에서 PUBLIC 및 사용자 목록 둘 다의 액세스를 허용하는 경우, 사용자 스펙이 PUBLIC 스펙보다 우선합니다.

authorization-name

지정된 *authorization-name*이 트러스트된 연결을 사용할 수 있음을 지정합니다. *authorization-name*은 트러스트된 컨텍스트를 사용하도록 이미 정의되어 있는 권한 부여 ID를 식별하면 안되며, ADD USE FOR 절에서 두 번 이상 지정할 수 없습니다(SQLSTATE 428GM). 이는 명령문의 권한 부여 ID여도 안됩니다(SQLSTATE 42502).

ROLE *role-name*

*role-name*이 사용자에 대해 사용할 역할임을 지정합니다. *role-name*은 현재 서버에 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). 사용자에 대해 명시적으로 지정된 역할이 트러스트된 컨텍스트와 연관되는 디폴트 역할보다 우선합니다.

PUBLIC

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결을 모든 사용자가 사용할 수 있음을 지정합니다. PUBLIC은 트러스트된 컨텍스트를 사용하도록 이미 정의되어 있으면 안되며 PUBLIC은 ADD USE FOR 절에서 두 번 이상 지정할 수 없습니다(SQLSTATE 428GM).

WITHOUT AUTHENTICATION 또는 WITH AUTHENTICATION

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자를 전환하는 데 인증이 필요한지 여부를 지정합니다.

WITHOUT AUTHENTICATION

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자를 전환하는 데 인증이 필요하지 않음을 지정합니다.

WITH AUTHENTICATION

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자를 전환하는 데 인증이 필요함을 지정합니다.

REPLACE USE FOR

특정 사용자 또는 PUBLIC이 트러스트된 컨텍스트를 사용하는 방식을 변경할 것을 지정합니다.

authorization-name

트러스트된 연결 사용을 변경할 사용자의 *authorization-name*을 지정합니다. 트러스트된 컨텍스트는 *authorization-name*이 사용할 수 있도록 이미 정의되어 있어야 하고(SQLSTATE 428GN), *authorization-name*은 REPLACE USE FOR 절에서 두 번 이상 지정할 수 없습니다(SQLSTATE 428GM). 이는 명령문의 권한 부여 ID여도 안됩니다(SQLSTATE 42502).

ROLE *role-name*

*role-name*이 사용자의 역할임을 지정합니다. *role-name*은 현재 서버에 존

재하는 역할을 식별해야 합니다(SQLSTATE 42704). 사용자에 대해 명시적으로 지정된 역할이 트러스트된 컨텍스트와 연관되는 디폴트 역할보다 우선합니다.

PUBLIC

PUBLIC에 의한 트러스트된 연결 사용의 속성을 변경할 것을 지정합니다. 트러스트된 컨텍스트는 PUBLIC이 사용할 수 있도록 이미 정의되어 있어야 하고(SQLSTATE 428GN), PUBLIC은 REPLACE USE FOR 절에서 두 번 이상 지정할 수 없습니다(SQLSTATE 428GM).

WITHOUT AUTHENTICATION 또는 WITH AUTHENTICATION

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자를 전환하는 데 인증이 필요한지 여부를 지정합니다.

WITHOUT AUTHENTICATION

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자를 전환하는 데 인증이 필요하지 않음을 지정합니다.

WITH AUTHENTICATION

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자를 전환하는 데 인증이 필요함을 지정합니다.

DROP USE FOR

트러스트된 컨텍스트를 더 이상 사용할 수 없는 사용자를 지정합니다. 트러스트된 컨텍스트의 정의에서 제거되는 사용자는 현재 트러스트된 컨텍스트를 사용할 수 있는 사용자입니다. 전체는 아니지만 한 명 이상의 사용자를 트러스트된 컨텍스트 정의에서 제거할 수 있는 경우, 지정된 사용자가 제거되고 경고가 리턴됩니다(SQLSTATE 01682). 트러스트된 컨텍스트 정의에서 지정된 사용자를 전혀 제거할 수 없는 경우 오류가 리턴됩니다(SQLSTATE 428GN).

authorization-name

트러스트된 컨텍스트를 사용할 수 있는 지정된 권한 부여 ID의 능력을 제거합니다.

PUBLIC

트러스트된 컨텍스트를 사용할 수 있는 모든 사용자의 능력(명시적으로 사용 기능하게 된 개인 권한 부여 ID 및 시스템 권한 부여 ID 제외)을 제거합니다.

규칙

- 트러스트된 컨텍스트 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). 트러스트된 컨텍스트 독점 SQL문은 다음과 같습니다.
  - CREATE TRUSTED CONTEXT, ALTER TRUSTED CONTEXT 또는 DROP(TRUSTED CONTEXT)

- 트러스트된 컨텍스트 독점 SQL문은 페더레이티드 트랜잭션에 대한 2단계 커미트의 일부로 시작되는 전역 트랜잭션이나 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 트러스트된 컨텍스트 정의의 일부로 IP 주소를 제공할 때 주소는 네트워크에 적용되는 형식이어야 합니다. 예를 들어, 네트워크가 IPv4일 때 IPv6 형식으로 주소를 제공하면 일치하지 않게 됩니다. 혼합 환경에서는, 주소의 IPv4 및 IPv6 표시를 둘 다 지정하거나, 더 낫게는 주소 형식 세부사항을 숨기는 보안 도메인 이름(예: corona.torolab.ibm.com)을 지정하는 것이 좋습니다.
- 모든 데이터베이스 파티션 사이에서 한 번에 단 하나의 언커미트된 트러스트된 컨텍스트 독점 SQL문만 허용됩니다. 언커미트된 트러스트된 컨텍스트 독점 SQL문이 실행 중인 경우, 연속 트러스트된 컨텍스트 독점 SQL문은 현재 트러스트된 컨텍스트 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.
- *조작 순서:* ALTER TRUSTED CONTEXT문 내에서의 조작 순서는 다음과 같습니다.
  - DROP
  - ALTER
  - ADD ATTRIBUTES
  - ADD USE FOR
  - REPLACE USE FOR
- *기존의 트러스트된 연결에 대한 변경사항의 효과:* 변경되는 트러스트된 컨텍스트에 대해 트러스트된 연결이 존재하는 경우, 연결은 다음 사용자 전환 요청이 있거나 연결이 종료할 때까지 ALTER TRUSTED CONTEXT문 이전에 적용 중인 정의를 사용하여 계속 트러스트된 상태로 유지됩니다. 트러스트된 컨텍스트에 대한 트러스트된 연결이 활성 상태인 동안 이 컨텍스트가 사용 불가능하게 되면, 연결은 다음 사용자 전환 요청이 있거나 연결이 종료할 때까지 계속 트러스트된 상태로 유지됩니다. ALTER TRUSTED CONTEXT문으로 트러스트 속성이 변경되는 경우, 트러스트된 컨텍스트를 사용하는, ALTER TRUSTED CONTEXT문 실행 시 존재하는 트러스트된 연결은 계속 허용됩니다.
- *역할 특권:* 사용자 또는 트러스트된 컨텍스트와 연관되는 역할이 없는 경우 사용자와 연관되는 특권만 적용 가능합니다. 이는 트러스트된 컨텍스트에서 안되는 것과 같습니다.

예:

예 1: 트러스트된 컨텍스트 APPSERVER가 존재하고 사용 가능하다고 가정하십시오. Bill이 트러스트된 컨텍스트 APPSERVER를 사용할 수 있지만 트러스트된 컨텍스트를 사용 불가능 상태로 만들기 위한 ALTER TRUSTED CONTEXT문을 실행하십시오.

```
ALTER TRUSTED CONTEXT APPSERVER
  DISABLE
  ADD USE FOR BILL
```

예 2: 트러스트된 컨텍스트 SECUREROLE이 존재한다고 가정하십시오. 기존 사용자 Joe가 인증을 사용하여 트러스트된 컨텍스트를 사용하도록 수정하고 그 외의 모든 사용자가 인증 없이 트러스트된 컨텍스트를 사용할 수 있도록 추가하기 위한 ALTER TRUSTED CONTEXT문을 실행하십시오.

```
ALTER TRUSTED CONTEXT SECUREROLE
  REPLACE USE FOR JOE WITH AUTHENTICATION
  ADD USE FOR PUBLIC WITHOUT AUTHENTICATION
```

예 3: ADDRESS 속성 값이 '9.13.55.100' 및 '9.12.30.112'이고 ENCRYPTION 속성 값이 'NONE'인 트러스트된 컨텍스트 SECUREROLEENCRYPT가 존재합니다. ADDRESS 속성 값과 암호화 속성을 'LOW'로 수정하기 위한 ALTER문을 실행하십시오.

```
ALTER TRUSTED CONTEXT SECUREROLEENCRYPT
  ALTER ATTRIBUTES (ADDRESS '9.12.155.200',
    ENCRYPTION 'LOW')
```

ALTER TYPE(구조화)

ALTER TYPE문은 사용자 정의 구조화된 유형의 속성 또는 메소드 스펙을 추가하거나 삭제(drop)하는 데 사용됩니다. 기존 메소드의 등록 정보도 변경할 수 있습니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 권한 중 하나를 가지고 있어야 합니다.

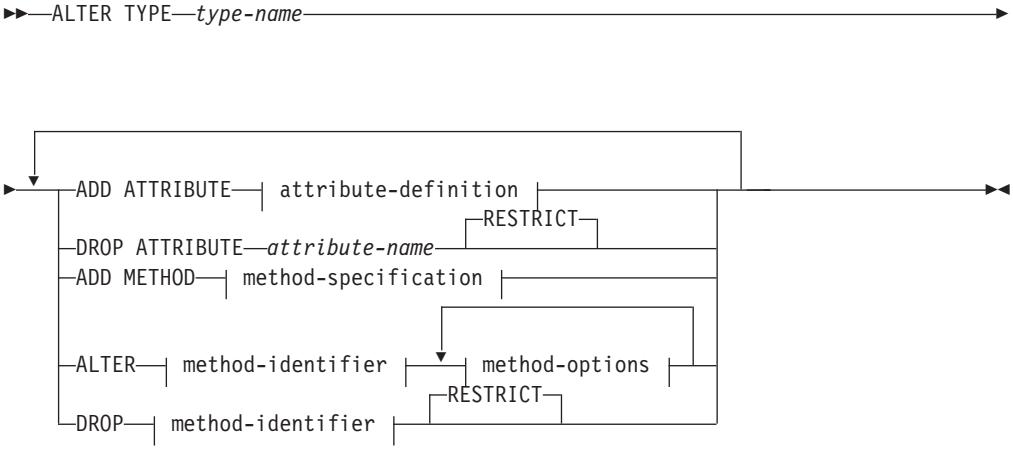
- 유형 스키마에 대한 ALTERIN 권한
- SYSCAT.DATATYPES 카탈로그 뷰의 OWNER 컬럼에 기록된 유형의 소유자
- DBADM 권한

메소드가 분리(fenced)되지 않게 변경하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 권한 중 하나를 가지고 있어야 합니다.

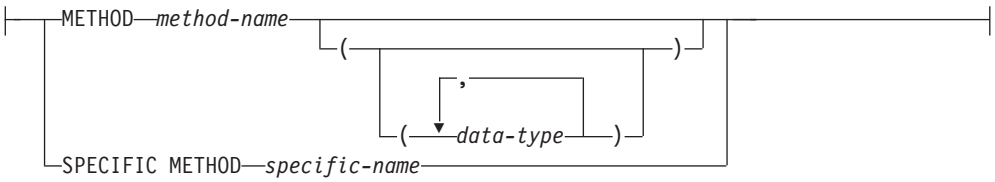
- 데이터베이스에서의 CREATE_NOT_FENCED_ROUTINE 권한
- DBADM 권한

메소드가 분리되도록 변경하는 데에는 추가 권한이나 권한이 필요하지 않습니다.

구문



method-identifier:



method-options:

설명

type-name

변경될 구조화된 유형을 식별합니다. 카탈로그에 정의된 기존의 유형(SQLSTATE 42704)이어야 하고, 유형은 구조화된 유형(SQLSTATE 428DP)이어야 합니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

ADD ATTRIBUTE

기존의 구조화된 유형의 최종 속성 다음에 속성을 추가합니다.

attribute-definition

구조화된 유형의 속성을 정의합니다.

attribute-name

속성에 대한 이름을 지정합니다. 이름은 해당되는 구조화된 유형의 다른 속성(상속된 속성을 포함하여)이나, 해당되는 구조화된 유형의 부속 유형에 대한 다른 속성과 같으면 안됩니다(SQLSTATE 42711).

술어에서 키워드로 사용되는 여러 이름은 시스템에서 사용하도록 예약되어 있으므로, *attribute-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름들이 있고, 비교 연산자도 포함됩니다.

data-type 1

속성의 데이터 유형을 지정합니다. CREATE TABLE 아래에 나열된 데이터 유형 중 하나입니다(XML(SQLSTATE 42601)이 아닌). 데이터 유형은 기존의 데이터 유형을 식별해야 합니다(SQLSTATE 42704). 스키마 이름 없이 *data-type*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형이 해석됩니다. 다양한 데이터 유형에 대한 설명은 『CREATE TABLE』에 있

습니다. 속성 데이터 유형이 참조 유형일 경우, 참조의 목표 유형은 존재하는 구조화된 유형이어야 합니다(SQLSTATE 42704).

런타임에서 유형의 인스턴스에 직접 또는 간접적으로 같은 유형의 다른 인스턴스나 해당되는 부속 유형 중 하나가 포함되도록 하는 유형 정의를 금지하기 위해, 해당되는 속성 유형 중 하나가 직접 또는 간접적으로 자체를 사용하는 것과 같이 유형이 정의되지 않도록 하는 제한사항이 있습니다(SQLSTATE 428EP).

lob-options

LOB 유형과 연관된 옵션(또는 LOB 유형을 기반으로 하는 구별 유형)을 지정합니다. lob-options에 대한 자세한 내용은 『CREATE TABLE』을 참조하십시오.

DROP ATTRIBUTE

기존의 구조화된 유형 속성을 삭제합니다.

attribute-name

속성의 이름입니다. 속성은 유형의 속성으로 존재해야 합니다(SQLSTATE 42703).

RESTRICT

기존의 테이블, 뷰, 컬럼, 컬럼 유형 내에 중첩된 속성 또는 인덱스 확장의 유형으로 *type-name*을 사용할 경우 어떤 속성도 삭제될 수 없게 하는 규칙이 적용됩니다.

ADD METHOD *method-specification*

*type-name*으로 식별되는 유형에 메소드 스펙을 추가합니다. 별도의 CREATE METHOD문을 사용하여 메소드에 본문을 제공할 때까지 메소드를 사용할 수 없습니다. *method-specification*에 대한 자세한 내용은 『CREATE TYPE(구조화)』을 참조하십시오.

ALTER *method-identifier*

변경될 메소드의 인스턴스를 고유하게 식별합니다. 지정한 메소드에 기존 메소드 본문이 있는 경우도 있고 없는 경우도 있습니다. LANGUAGE SQL로 선언된 메소드는 변경할 수 없습니다(SQLSTATE 42917).

method-identifier

METHOD *method-name*

특정 메소드를 식별하며, *type-name* 유형에 대해 이름이 *method-name*인 메소드 인스턴스가 정확히 한 개만 있을 경우에만 유효합니다. 식별된 메소드에는 그에 대해 정의된 임의 수의 매개변수가 있을 수 있습니다. 유형에 대해 이 이름의 메소드가 없는 경우에는 오류가 발생합니다(SQLSTATE 42704). 유형에 대해 메소드의 인스턴스가 둘 이상 있는 경우에는 오류가 발생합니다(SQLSTATE 42725).

METHOD *method-name (data-type,...)*

메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 분석 알고리즘은 사용되지 않습니다.

method-name

type-name 유형에 대한 메소드 이름을 지정합니다.

(data-type,...)

값은 CREATE TYPE문의 해당 위치에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합은 특정 메소드 인스턴스를 식별하는 데 사용됩니다.

데이터 유형이 규정되어 있지 않은 경우, SQL 경로에서 스키마를 검색하여 유형 이름이 해석됩니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대한 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE TYPE 문에 지정된 값과 정확히 일치해야 합니다.

$0 < n < 25$는 REAL을 의미하고, $24 < n < 54$는 DOUBLE을 의미하므로, FLOAT(*n*) 유형은 *n*에 정의된 값과 일치하지 않아도 됩니다. 일치는 유형이 REAL인지 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 가진 메소드가 이름이 지정된 스키마나 내재적 스키마의 유형으로 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC METHOD *specific-name*

메소드 작성시 디폴트로 설정된 이름이나 지정된 이름을 사용하여 특정 메소드를 식별합니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 메소드 인스턴스를 식별해야 합니다. 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

method-options

메소드에 대해 변경할 옵션을 지정합니다.

FENCED 또는 NOT FENCED

메소드가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 어드레스 스페이스에서 실행되어도 안전한지(NOT FENCED) 아니면 안전하지 않은지(FENCED) 여부를 지정합니다. 대부분의 메소드는 FENCED나 NOT FENCED로 실행될 수 있습니다.

메소드가 FENCED로 변경된 경우, 데이터베이스 관리 프로그램은 내부 자원(예: 데이터 버퍼)을 메소드가 액세스하지 못하도록 분리시킵니다. 일반적으로 NOT FENCED로 실행되는 메소드는 FENCED로 실행되는 메소드 만큼 잘 수행되지 않습니다.

주의:
제대로 코딩, 검토 및 테스트되지 않은 메소드에 NOT FENCED를 사용하면 DB2 데이터베이스의 무결성이 손상될 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 메소드가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

NOT THREADSAFE로 선언된 메소드는 NOT FENCED로 변경될 수 없습니다(SQLSTATE 42613).

메소드에 AS LOCATOR로 정의된 매개변수가 있고 매개변수가 NO SQL 옵션을 사용하여 정의된 경우에는 메소드를 FENCED로 변경할 수 없습니다(SQLSTATE 42613).

LANGUAGE OLE 메소드의 경우 이 옵션을 변경할 수 없습니다(SQLSTATE 42849).

THREADSAFE 또는 NOT THREADSAFE

메소드가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 아니면 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

메소드가 OLE가 아닌 다른 LANGUAGE로 정의된 경우에는 다음과 같습니다.

- 메소드가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 메소드를 호출할 수 있습니다. 일반적으로 Threadsafe 함수가 되려면 메소드가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED와 NOT FENCED 메소드는 둘 다 THREADSAFE될 수 있습니다. 이 메소드가 LANGUAGE OLE로 정의된 경우, THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

- 메소드가 NOT THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 메소드를 호출할 수 없습니다. 분리(fenced) 메소드만 NOT THREADSAFE될 수 있습니다(SQLSTATE 42613).

DROP *method-identifier*

삭제될 메소드의 인스턴스를 고유하게 식별합니다. 지정된 메소드에는 기존 메소드 본문이 없어야 합니다(SQLSTATE 428ER). ALTER TYPE DROP METHOD를 사용하기 이전에 DROP METHOD문을 사용하여 메소드 본문을 삭제하도록 하십시오. CREATE TYPE문에 의해 내재적으로 생성되는 메소드(예: mutator 및 observer)는 삭제할 수 없습니다(SQLSTATE 42917).

RESTRICT

지정된 메소드가 기존 메소드 본문을 갖는 데 있어서 제한됨을 나타냅니다. ALTER TYPE DROP METHOD를 사용하기 이전에 DROP METHOD문을 사용하여 메소드를 삭제하십시오.

규칙

- 다음 중 한 경우에 해당될 때 *type-name* 유형에 대해 속성을 추가하거나 삭제할 수 없습니다(SQLSTATE 55043).
  - 유형이나 부속 유형 중 하나가 기존 테이블이나 뷰의 유형인 경우
  - 직접 또는 간접으로 *type-name*을 사용하는 유형을 갖는 테이블의 컬럼이 존재하는 경우 *directly uses*과 *indirectly uses*이라는 용어는 『구조화된 유형』에 정의되어 있습니다.
  - 유형이나 부속 유형 중 하나가 인덱스 확장에서 사용되는 경우
- 유형이나 부속 유형 중 어느 하나의 총 속성 수가 4082를 초과하도록 속성을 추가하여 유형을 변경할 수 없습니다(SQLSTATE 54050).
- ADD ATTRIBUTE 옵션:
  - ADD ATTRIBUTE는 새 속성에 대한 observer 및 mutator 메소드를 생성합니다. 이 메소드는 구조화된 유형이 작성될 때 생성된 것과 유사합니다(『CREATE TYPE(구조화)』 참조). 이 메소드가 기존 메소드나 함수와 충돌하거나 이를 대체할 경우, ALTER TYPE문이 실패합니다(SQLSTATE 42745).
  - 유형(또는 부속 유형)에 대한 INLINE LENGTH가 사용자에 의해 292 이하의 값으로 명시적으로 지정되었고, 추가된 속성으로 인해 지정된 인라인 길이가 변경된 유형에 대한 컨스트럭터 함수의 결과 크기(속성당 32바이트 + 10바이트)보다 작게 될 경우, 오류가 발생합니다(SQLSTATE 42611).
- DROP ATTRIBUTE 옵션:
  - 기존의 슈퍼 유형에서 상속된 속성은 삭제할 수 없습니다(SQLSTATE 428DJ).

  - DROP ATTRIBUTE는 삭제된 속성의 mutator 및 observer 메소드를 삭제한 후 삭제된 메소드에 대한 종속성을 확인합니다.
- DROP METHOD 옵션
  - 다른 메소드가 겹쳐쓴 원래 메소드는 삭제할 수 없습니다(SQLSTATE 42893).

주

- SYSIBM, SYSFUN 또는 SYSPROC 스키마의 메소드를 변경할 수 없습니다 (SQLSTATE 42832).
- 속성 추가 또는 삭제로 유형이 변경될 경우, 이 유형이나 이 유형의 부속 유형을 매개변수나 결과로 사용하는 함수나 메소드에 종속된 모든 패키지가 무효화됩니다.
- 속성이 구조화된 유형에 추가되거나 구조화된 유형에서 삭제될 경우:
  - 유형이 작성될 때 유형의 INLINE LENGTH가 시스템에서 계산된 경우, INLINE LENGTH 값은 변경된 유형과 변경하려는 모든 부속 유형에 대해 자동으로 수정됩니다. INLINE LENGTH 값은 또한 시스템에서 INLINE LENGTH가 계산되었고 변경된 INLINE LENGTH가 있는 유형의 속성이 유형에 포함되는 모든 구조화된 유형에 대해 자동으로(반복적으로) 수정됩니다.
  - 속성 추가 또는 삭제의 영향을 받는 유형의 INLINE LENGTH를 사용자가 명시적으로 지정하였으면, 그 특정 유형에 대한 INLINE LENGTH는 변경되지 않습니다. 명시적으로 지정된 인라인 길이에 대해서는 특히 주의해야 합니다. 나중에 유형에 속성이 추가될 가능성이 있으면, 컬럼 정의에서 유형이나 이에 해당되는 부속 유형의 사용을 위해, 인라인 길이는 인스턴스화된 오브젝트의 길이에서 증가 가능성을 고려할 만큼 충분히 커야 합니다.
  - 새로운 속성이 응용프로그램에 보여지게 되면, 기존 변환 함수는 데이터 유형의 새 구조와 일치하도록 수정해야 합니다.
- 파티션된 데이터베이스 환경에서는 외부 사용자 정의 함수 또는 메소드에서 SQL을 사용할 수 없습니다(SQLSTATE 42997).
- *권한*: CREATE METHOD문을 사용하여 메소드 본문을 정의한 후에만 ALTER TYPE문에 명시적으로 지정된 메소드에 대하여 EXECUTE 권한을 부여할 수 있습니다. 사용자 정의 유형의 소유자는 ALTER TYPE문을 사용하여 메소드 스펙을 삭제할 수 있습니다.

예:

예 1: ALTER TYPE문을 사용하여 상호 참조 유형 및 테이블의 순환을 허용하도록 하십시오. EMPLOYEE 및 DEPARTMENT 상호 참조 테이블을 사용하십시오.

다음 시퀀스는 유형 및 테이블이 작성되도록 합니다.

```
CREATE TYPE DEPT ...
CREATE TYPE EMP ... (including attribute named DEPTREF of type REF(DEPT))
ALTER TYPE DEPT ADD ATTRIBUTE MANAGER REF(EMP)
```

```
CREATE TABLE DEPARTMENT OF DEPT ...
CREATE TABLE EMPLOYEE OF EMP (DEPTREF WITH OPTIONS SCOPE DEPARTMENT)
ALTER TABLE DEPARTMENT ALTER COLUMN MANAGER ADD SCOPE EMPLOYEE
```

다음 시퀀스는 이들 테이블 및 유형이 삭제(drop)되도록 합니다.

```
DROP TABLE EMPLOYEE   (the MANAGER column in DEPARTMENT becomes unscoped)
DROP TABLE DEPARTMENT
ALTER TYPE DEPT DROP ATTRIBUTE MANAGER
DROP TYPE EMP
DROP TYPE DEPT
```

예 2: ALTER TYPE문을 사용하여 부속 유형을 참조하는 속성을 가진 유형을 작성할 수 있습니다.

```
CREATE TYPE EMP ...
CREATE TYPE MGR UNDER EMP ...
ALTER TYPE EMP ADD ATTRIBUTE MANAGER REF(MGR)
```

예 3: ALTER TYPE문을 사용하여 속성을 추가할 수 있습니다. 다음 명령문은 SPECIAL 속성을 EMP 유형에 추가합니다. 원래의 CREATE TYPE문에 인라인 길이가 지정되지 않았기 때문에 DB2 데이터베이스는 13을 더하여 인라인 길이를 다시 계산합니다(새 속성에 대한 10바이트 + 속성 길이 + 비LOB 속성에 대한 2바이트).

```
ALTER TYPE EMP ...
  ADD ATTRIBUTE SPECIAL CHAR(1)
```

예 4: ALTER TYPE문을 사용하여 속성을 추가할 수 있습니다. 다음 명령문은 BONUS라는 메소드를 추가합니다.

```
ALTER TYPE EMP ...
  ADD METHOD BONUS (RATE DOUBLE)
    RETURNS INTEGER
    LANGUAGE SQL
    CONTAINS SQL
         NO EXTERNAL ACTION
       DETERMINISTIC
```

BONUS 메소드는 CREATE METHOD문을 발행하여 메소드 본문을 작성할 때까지 사용할 수 없습니다. 유형 EMP에 SALARY라는 속성이 포함되어 있다고 가정할 경우, 다음은 메소드 본문 정의의 예입니다.

```
CREATE METHOD BONUS(RATE DOUBLE) FOR EMP
  RETURN CAST(SELF.SALARY * RATE AS INTEGER)
```

ALTER USER MAPPING

ALTER USER MAPPING문은 지정된 페더레이티드 서버의 권한 부여 ID에 대한 데이터 소스에서 사용되는 권한 부여 ID 또는 암호를 변경하는 데 사용됩니다.

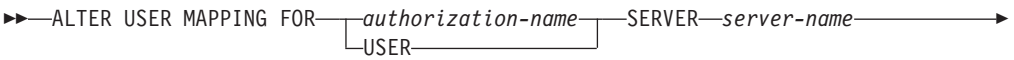
호출

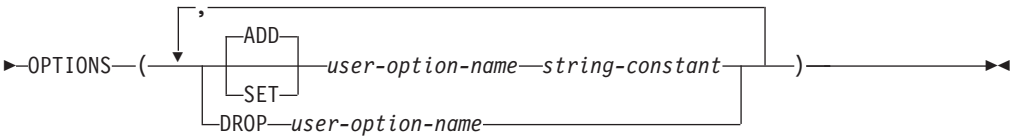
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 데이터 소스에 맵핑되는 권한 부여 이름과 다르면, 명령문의 권한 부여 ID가 보유한 특권은 DBADM 권한을 포함해야 합니다. 그렇지 않으면, 권한 부여 ID가 권한 부여 이름과 일치하는 경우 어떠한 권한이나 특권도 필요하지 않습니다.

구문





설명

authorization-name

사용자나 응용프로그램이 페더레이티드 데이터베이스에 연결하는 권한 부여 이름을 지정하십시오.

USER

특수 레지스터 USER에 있는 값. USER가 지정될 때에는, ALTER USER MAPPING문의 권한 부여 ID가 REMOTE_AUTHID 사용자 옵션에서 지정되는 데이터 소스 권한 부여 ID로 맵핑됩니다.

SERVER *server-name*

*authorization-name*으로 표시되거나 USER가 참조하는 로컬 권한 부여 ID로 맵핑하는 리모트 권한 부여 ID하에서 액세스 가능한 데이터 소스를 식별합니다.

OPTIONS

변경되는 맵핑에 대해 어느 사용자 옵션이 작동 가능, 재설정 또는 삭제될 지를 나타냅니다.

ADD

사용자 옵션을 작동 가능하게하십시오.

SET

사용자 옵션의 설정값을 변경하십시오.

user-option-name

작동 가능화되거나 재설정될 사용자 옵션에 이름을 지정하십시오.

string-constant

*user-option-name*에 대한 설정값을 문자열 상수로서 지정하십시오.

DROP *user-option-name*

사용자 옵션을 삭제(Drop)하십시오.

주

- 사용자 옵션은 동일한 ALTER USER문에서 두 번 이상 지정될 수 없습니다 (SQLSTATE 42853). 사용자 옵션이 작동 가능화, 재설정 또는 삭제될 때, 사용 중인 다른 서버 옵션들에는 영향이 미치지 않습니다.
- 주어진 작업 단위(UOW) 내의 ALTER USER MAPPING문은 UOW에 다음 중 하나가 이미 포함된 경우에는 처리될 수 없습니다(SQLSTATE 55007).
  - 맵핑에 포함될 데이터 소스에서 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
  - 맵핑에 포함될 데이터 소스에서 테이블이나 뷰에 대한 별칭의 열린 커서
  - 맵핑에 포함될 데이터 소스에서 테이블이나 뷰의 별칭에 대해 INSERT, DELETE 또는 UPDATE문 발행

예:

예 1: Jim은 로컬 데이터베이스를 사용하여 ORACLE1이라는 Oracle 데이터 소스에 연결합니다. 그는 권한 부여 ID KLEEWEIN으로 로컬 데이터베이스를 액세스합니다. KLEEWEIN은 CORONA Jim이 ORACLE1을 액세스하는 권한 부여 ID로 맵핑됩니다. Jim은 새 ID인 JIMK로 ORACLE1을 액세스하기 시작합니다. KLEEWEIN이 이제 JIMK로 맵핑되어야 합니다.

```
ALTER USER MAPPING FOR KLEEWEIN
  SERVER ORACLE1
  OPTIONS ( SET REMOTE_AUTHID 'JIMK'  )
```

예 2: Mary는 페더레이티드 데이터베이스를 사용하여 DORADO라는 z/OS용 DB2 데이터 소스에 연결합니다. 그녀는 권한 부여 ID를 사용하여 DB2에 액세스하고, 또 다른 권한 부여 ID를 사용하여 DORADO에 액세스합니다. 그리고 이 두 ID간에 맵핑

을 작성했습니다. 이 두 ID에 동일한 암호를 사용하고 있었으나, 이제 DORADO에 대한 ID에 대해 별도의 암호인 ZNYQ를 사용하려고 합니다. 따라서 그녀는 자신의 페더레이티드 데이터베이스 암호를 ZNYQ에 맵핑되도록 하십시오.

```
ALTER USER MAPPING FOR MARY
  SERVER DORADO
  OPTIONS ( ADD REMOTE_PASSWORD 'ZNYQ'  )
```

ALTER VIEW

ALTER VIEW문은 다음과 같은 방법으로 기존 뷰를 수정합니다.

- 참조 유형 컬럼을 변경하여 영역에 추가
- 쿼리 최적화에서 뷰 사용 가능하게 하기 또는 사용 불가능하게 하기

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509) 에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

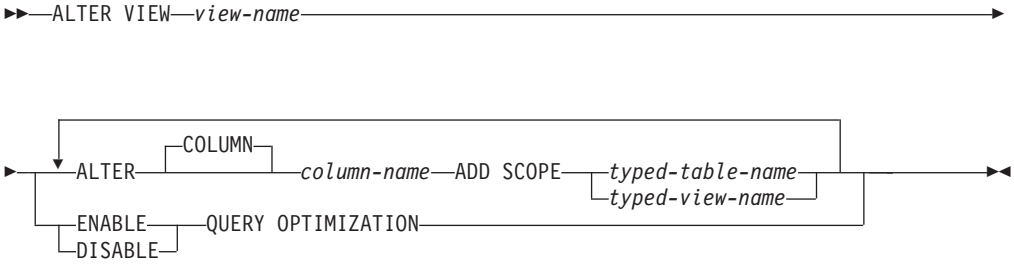
명령문의 권한 부여 ID는 적어도 다음과 같은 권한 중 하나를 가지고 있어야 합니다.

- 뷰의 스키마에 대한 ALTERIN 권한
- 변경할 뷰의 소유자
- 변경할 뷰에 대한 CONTROL 권한
- DBADM 권한

쿼리 최적화에서 뷰를 사용 가능하게 하거나 사용 불가능하게 하려면, 명령문의 권한 부여 ID가 보류된 권한에는 뷰 fullselect의 FROM절에서 참조하는 각 뷰의 하위 테이블 또는 테이블에 대해 다음 중 적어도 하나가 포함되어야 합니다.

- 테이블에 대한 ALTER 권한
- 테이블의 스키마에 대한 ALTERIN 권한
- DBADM 권한

구문



설명

view-name

변경될 뷰를 지정합니다. 카탈로그에 기술된 뷰여야 합니다.

ALTER COLUMN *column-name*

변경할 컬럼의 이름을 지정합니다. *column-name*은 뷰의 기존 컬럼을 식별해야 합니다(SQLSTATE 42703). 이름은 규정할 수 없습니다.

ADD SCOPE

아직 영역이 정의되어 있지 않은 기존의 참조 유형 컬럼에 영역을 추가합니다(SQLSTATE 428DK). 컬럼은 슈퍼 뷰로부터 상속되어서는 안됩니다(SQLSTATE 428DJ).

typed-table-name

유형이 지정된 테이블의 이름을 지정합니다. *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-table-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하는지 확인하기 위해 *column-name*에 있는 기존 값을 점검하지 않습니다.

typed-view-name

유형이 지정된 뷰의 이름을 지정합니다. *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서, *S*는 *typed-view-name* 유형입니다(SQLSTATE 428DM). 값이 *typed-view-name*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *column-name*에 있는 기존 값들에 대해 점검을 하지 않습니다.

ENABLE QUERY OPTIMIZATION 또는 DISABLE QUERY OPTIMIZATION

쿼리 최적화를 증진하기 위해 뷰 및 모든 연관 통계를 사용할 것인지 지정합니다. DISABLE QUERY OPTIMIZATION는 뷰가 작성될 때 디폴트값입니다.

ENABLE QUERY OPTIMIZATION

뷰가 자신의 fullselect와 유사한 서브쿼리를 포함하고 있는 해당 뷰 또는 쿼리를 수반하는 쿼리의 최적화를 증진하는 데 사용할 수 있는 통계를 포함하도록 지정합니다.

DISABLE QUERY OPTIMIZATION

쿼리 최적화를 증진하기 위해 뷰 및 모든 연관 통계를 사용하지 않을 것을 지정합니다.

규칙

- 다음의 경우 쿼리 최적화에 뷰를 사용할 수 없습니다.
  - 해당 뷰는 직접 또는 간접적으로 구체화된 쿼리 테이블(MQT)을 참조합니다. MQT 또는 통계 뷰가 통계 뷰를 참조할 수 있음을 유의하십시오.
  - 유형이 지정된 뷰입니다.

주

- 쿼리 최적화를 위해 뷰는 다음을 포함할 수 없습니다.
  - 집계 또는 구별 조작
  - union, except 또는 intersect 조작

  – 스칼라 집계(OLAP) 함수

- 뷰를 변경하여 쿼리 최적화의 사용을 불가능하게 하면, 쿼리 최적화를 위해 해당 뷰를 사용했던 캐시 쿼리 플랜이 무효화됩니다. 뷰를 변경하여 쿼리 최적화의 사용을 가능하게 경우 캐시 쿼리 플랜이 새로이 사용 가능하게 된 뷰 참조와 동일한 테이블을 직접 또는 간접적으로 다른 뷰를 통해 참조하게 되면 무효화됩니다. 이렇게 캐시 쿼리 플랜을 무효화하면 뷰의 변경된 쿼리 최적화 등록 정보를 고려하는 내재적 유효성 다시 확인이 발생합니다.

  뷰의 쿼리 최적화 등록 정보는 정적 Embedded SQL문에 영향을 주지 않습니다.

ALTER WORK ACTION SET

ALTER WORK ACTION SET문은 작업 조치 세트 내에서 작업 조치를 추가, 변경 또는 삭제하여 작업 조치 세트를 변경합니다.

호출

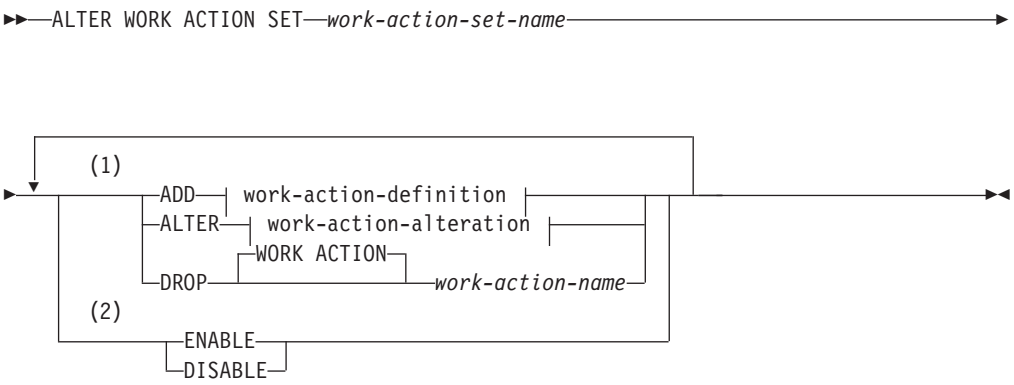
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

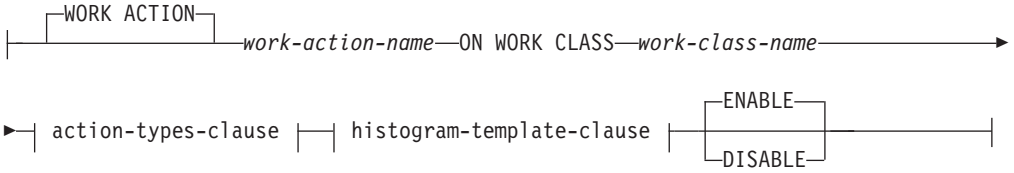
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- SQLADM 권한, 모든 변경 절이 COLLECT 절인 경우에만
- WLMADM 권한
- DBADM 권한

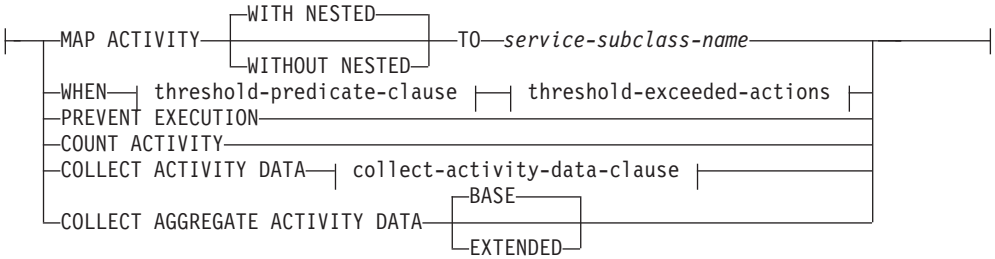
구문



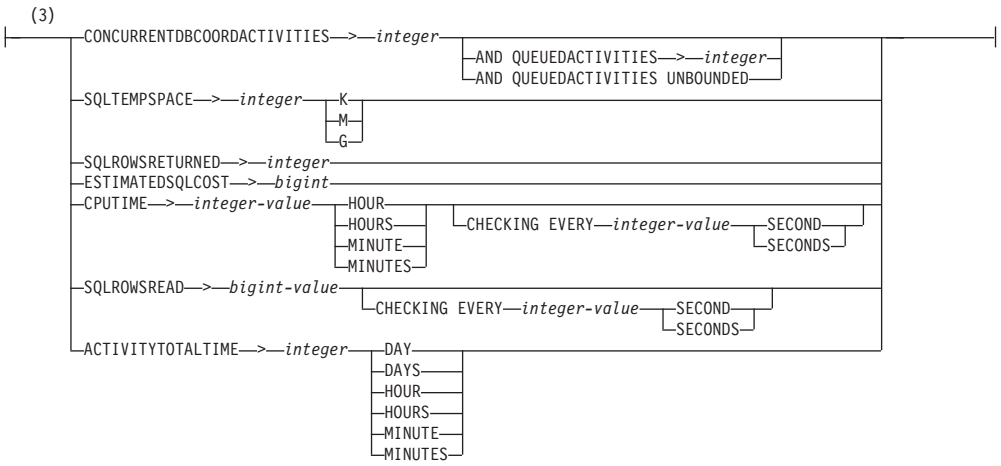
work-action-definition:



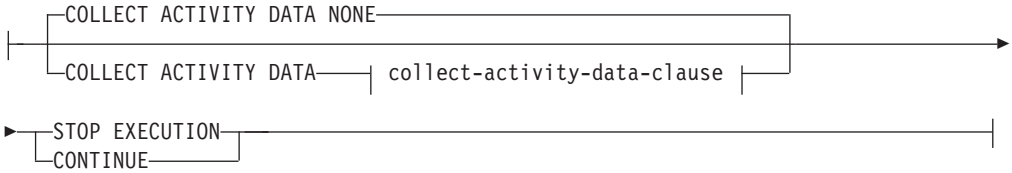
action-types-clause:



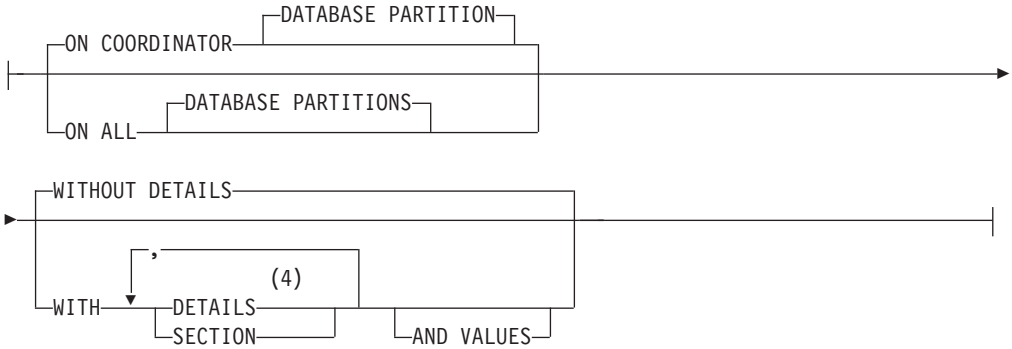
threshold-predicate-clause:



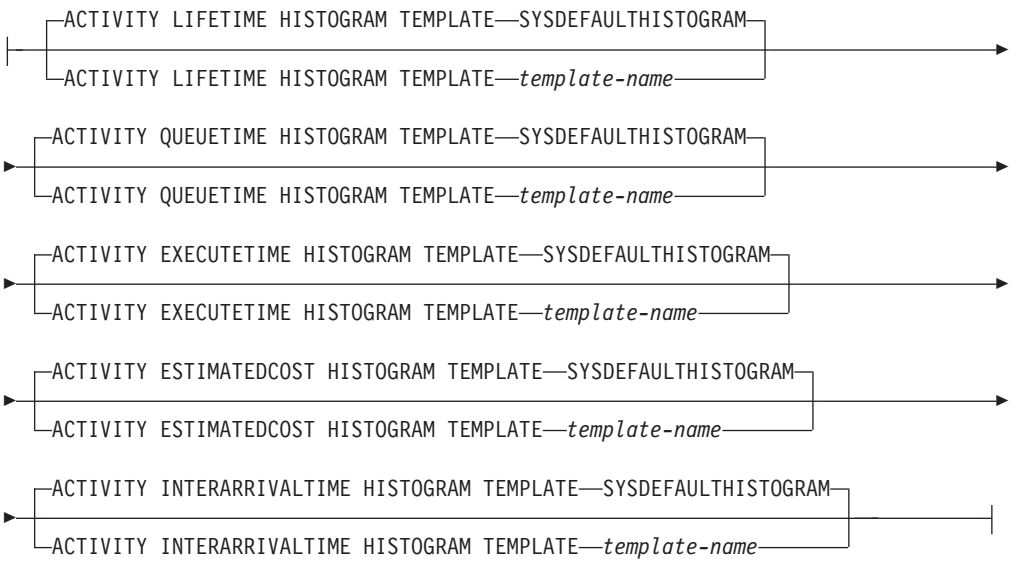
threshold-exceeded-actions:



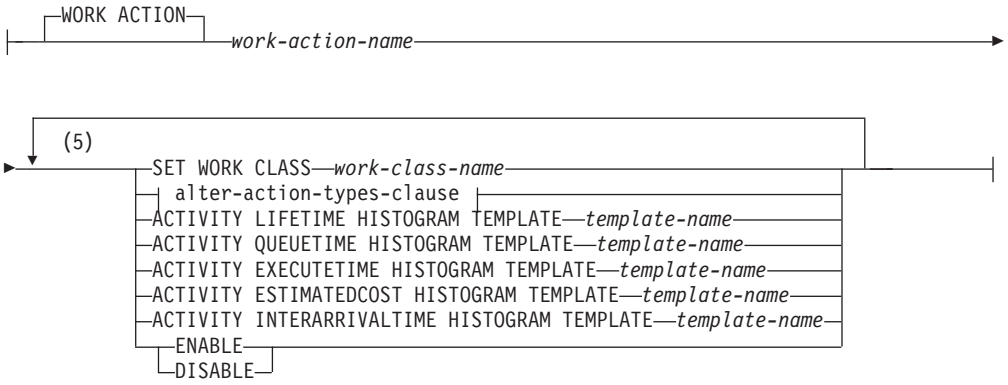
collect-activity-data-clause:



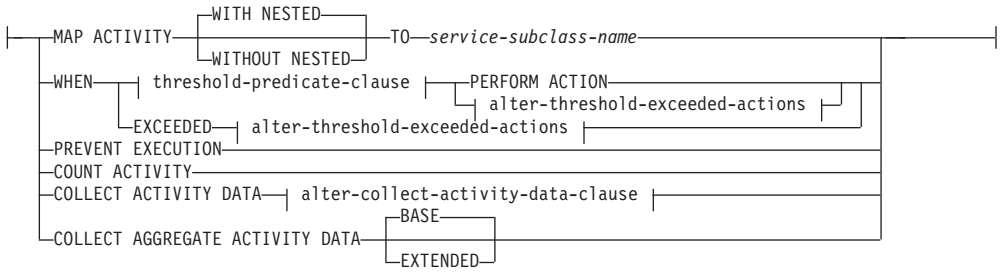
histogram-template-clause:



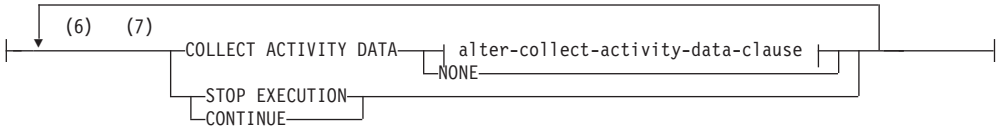
work-action-alteration:



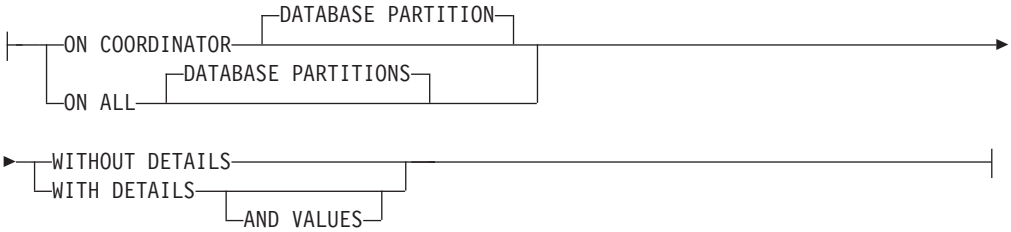
alter-action-types-clause:



alter-threshold-exceeded-actions:



alter-collect-activity-data-clause:



주:

1 ADD, ALTER 및 DROP 절은 지정된 순서대로 처리됩니다.

2 ENABLE 또는 DISABLE 절은 동일한 명령문에서 한 번만 지정할 수 있습니다.

3 한 번에 임계값 유형이 동일한 단 하나의 작업 조치만 단일 작업 클래스에 적용할 수 있습니다. 임계값 작업 조치를 변경할 때 임계값 술어는 변경할 수 없습니다.

4 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

5 같은 절은 두 번 이상 지정될 수 없습니다.

6 같은 절은 두 번 이상 지정될 수 없습니다.

7 기존 작업 조치에, 이에 대하여 정의된 임계값 초과 조치가 없고 임계값 작업 조치로 변경 중인 경우, STOP EXECUTION 또는 CONTINUE가 지정되어야 하며, COLLECT ACTIVITY DATA가 지정되지 않은 경우, COLLECT ACTIVITY DATA NONE이 디폴트여야 합니다.

설명

work-action-set-name

변경될 작업 조치 세트를 식별합니다. 이는 한 부분으로 된 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *work-action-set-name*은 현재 서버에 존재하는 작업 조치 세트를 식별해야 합니다(SQLSTATE 42704).

ADD

작업 조치 세트에 작업 조치를 추가합니다.

WORK ACTION *work-action-name*

작업 조치의 이름을 지정합니다. *work-action-name*은 현재 서버에서 해당 작

업 조치 세트 아래에 이미 존재하는 작업 클래스 세트를 식별하면 안됩니다 (SQLSTATE 42710). *work-action-name*은 'SYS'로 시작할 수 없습니다 (SQLSTATE 42939).

ON WORK CLASS *work-class-name*
작업 조치를 적용할 데이터베이스 활동을 식별하는 작업 클래스를 지정합니다. *work-class-name*은 현재 서버에서 *work-class-set-name*에 존재해야 합니다 (SQLSTATE 42704).

MAP ACTIVITY
활동을 맵핑하는 작업 조치를 지정합니다. 이 조치는 작업 조치 세트가 정의된 오브젝트가 서비스 수퍼 클래스인 경우에만 지정할 수 있습니다(SQLSTATE 5U034).

WITH NESTED 또는 WITHOUT NESTED
해당 활동 아래에 중첩되는 활동이 서비스 서브클래스에 맵핑되는지 여부를 지정합니다. 디폴트는 WITH NESTED입니다.

WITH NESTED
작업 클래스 아래에 분류되어 있는 중첩 레벨 0의 모든 데이터베이스 활동과 이 활동 아래에 중첩된 모든 데이터베이스 활동은 서비스 서브클래스에 맵핑됩니다. 즉, 중첩 레벨이 0보다 큰 활동은 중첩 레벨이 0인 활동과 같은 서비스 클래스 아래에서 실행됩니다.

WITHOUT NESTED
작업 클래스 아래에 분류되어 있는 중첩 레벨 0의 데이터베이스 활동만 서비스 서브클래스에 맵핑됩니다. 해당 활동 아래에 중첩되는 데이터베이스 활동은 해당 활동 유형에 따라 처리됩니다.

TO *service-subclass-name*
활동이 맵핑될 서비스 서브클래스를 지정합니다. *service-subclass-name*은 현재 서버에서 *service-superclass-name*에 이미 존재해야 합니다 (SQLSTATE 42704). *service-subclass-name*은 디폴트 서비스 서브클래스 SYSDEFAULTSUBCLASS가 될 수 없습니다(SQLSTATE 5U018).

WHEN
작업 조치가 정의된 작업 클래스와 연관되는 데이터베이스 활동에 적용될 임계값을 지정합니다. 임계값은 작업 조치 세트가 정의된 데이터베이스 관리 프로그램 오브젝트가 데이터베이스인 경우에만 지정할 수 있습니다(SQLSTATE 5U034). 관리 SQL 루틴에서 생성된 데이터베이스 활동이나 데이터베이스 관리 프로그램에서 시작된 내부 데이터베이스 활동에는 이 임계값이 전혀 적용되지 않습니다.

threshold-predicate-clause

유효한 임계값 유형에 대한 설명은 『CREATE THRESHOLD』문을 참조하십시오.

threshold-exceeded-actions

유효한 임계값 초과 조치에 대한 설명은 『CREATE THRESHOLD』문을 참조하십시오.

PREVENT EXECUTION

작업 조치가 정의된 작업 클래스와 연관되는 데이터베이스 활동 중 어떤 것도 실행할 수 없음을 지정합니다(SQLSTATE 5U033).

COUNT ACTIVITY

작업 클래스와 연관되는 모든 데이터베이스 활동이 실행되고 활동이 실행될 때마다 작업 클래스의 카운터가 증가함을 지정합니다.

COLLECT ACTIVITY DATA

작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

활동 코디네이터의 데이터베이스 파티션에서만 수집될 활동 데이터를 지정합니다.

ON ALL DATABASE PARTITIONS

활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. 예측적 임계값의 경우, 초과된 임계값에 CONTINUE 조치도 지정하는 경우에만 모든 파티션에서 활동 정보가 수집됩니다. SECTION이 지정된 경우 활동 세부사항 및 섹션 환경도 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 반응적 임계값의 경우 ON ALL DATABASE PARTITIONS 절은 효과 및 활동, 활동 세부사항, 섹션 정보가 없으며 값은 항상 코디네이터 파티션에서만 수집됩니다.

WITHOUT DETAILS

작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

NONE
작업 조치가 정의된 작업 클래스와 연관되는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA
작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. 디폴트는 COLLECT AGGREGATE ACTIVITY DATA BASE입니다. 이 절은 데이터베이스에 적용되는 작업 조치 세트에 정의된 작업 조치에 지정할 수 없습니다.

BASE
작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 기본 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 실행 시간 막대 그래프

EXTENDED
작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 모든 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 데이터 처리 언어(DML) 계산 비용 막대 그래프

- 활동 DML 도착 간 시간 막대 그래프

ENABLE 또는 DISABLE

데이터베이스 활동이 제출될 때 작업 조치가 고려되는지 여부를 지정합니다. 디폴트는 ENABLE입니다.

ENABLE

작업 조치가 사용 가능하고 데이터베이스 활동이 제출될 때 고려되도록 지정합니다.

DISABLE

작업 조치가 사용 불가능하고 데이터베이스 활동이 제출될 때 고려되지 않도록 지정합니다.

histogram-template-clause

작업 조치가 지정된 작업 클래스와 연관되는 활동에 대한 집계 활동 데이터를 수집할 때 사용할 막대 그래프 템플리트를 지정합니다. 집계 활동 데이터는 작업 조치 유형이 COLLECT AGGREGATE ACTIVITY DATA인 경우에만 작업 클래스에 대해 수집됩니다.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동의 지속기간(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동이 큐에 대기된 시간 길이(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동이 실행 중인 시간 길이(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 활동이 실행되는 데이터베이스 파티션마다 이 막대 그래프에서 수집됩니다. 활동의 코디네이터 데이터베이스 파티션에서, 이 시간은 엔드-투-엔드 실행 시간입니다(즉, 수명이 큐에 대기하며 소비된 시간보다 적음). 코디네이터가 아닌 데이터베이스 파티션에서 이 시간은 파티션이 활동 대신 작업에 소비한 시간

입니다. 제공된 활동 실행 중, DB2는 리모트 데이터베이스 파티션에 두 번 이상 작업을 제시할 수 있으며, 리모트 파티션은 매번 활동의 해당 어커런스에 대해 실행 시간을 수집합니다. 따라서, 실행 시간 막대 그래프의 계수는 데이터베이스 파티션에서 실행한 실제 고유 활동 수를 나타내지 못할 수도 있습니다. 디폴트는
SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*

작업 조치가 지정된 작업 클래스와 연관되는 DML 활동의 계산된 비용(timeron)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*

작업 조치가 지정된 작업 클래스와 연관되는 활동에 대해, 하나의 DML 활동이 도착하고 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ALTER

작업 조치의 정의를 변경합니다. 작업 조치가 적용되는 작업 클래스와, 작업 클래스 내에 속하는 데이터베이스 활동에 적용될 조치를 변경할 수 있습니다.

WORK ACTION *work-action-name*

작업 조치를 식별합니다. *work-action-name*은 현재 서버에서 해당 작업 조치 세트 아래에 존재하는 작업 조치를 식별해야 합니다(SQLSTATE 42704).

SET WORK CLASS *work-class-name*

작업 조치를 적용할 데이터베이스 활동을 식별하는 작업 클래스를 지정합니다. *work-class-name*은 현재 서버에서 *work-class-set-name*에 존재해야 합니다(SQLSTATE 42704).

MAP ACTIVITY

활동을 맵핑하는 작업 조치를 지정합니다. 이 조치는 작업 조치 세트가 정의된 오브젝트가 서비스 수퍼 클래스인 경우에만 지정할 수 있습니다(SQLSTATE 5U034).

WITH NESTED 또는 WITHOUT NESTED

해당 활동 아래에 중첩되는 활동이 서비스 서브클래스에 맵핑되는지 여부를 지정합니다. 디폴트는 WITH NESTED입니다.

WITH NESTED

작업 클래스 아래에 분류되어 있는 중첩 레벨 0의 모든 데이터베이스 활동과, 이 활동 아래에 중첩된 모든 데이터베이스 활동이 서비스 서브클래스에 맵핑됩니다.

WITHOUT NESTED

작업 클래스 아래에 분류되어 있는 중첩 레벨 0의 데이터베이스 활동만 서비스 서브클래스에 맵핑됩니다. 해당 활동 아래에 중첩되는 데이터베이스 활동은 해당 활동 유형에 따라 처리됩니다.

TO *service-subclass-name*

활동이 맵핑될 서비스 서브클래스를 지정합니다. *service-subclass-name*은 현재 서버에서 *service-superclass-name*에 이미 존재해야 합니다(SQLSTATE 42704). *service-subclass-name*은 디폴트 서비스 서브클래스 SYSDEFAULTSUBCLASS가 될 수 없습니다(SQLSTATE 5U018).

WHEN

작업 조치가 정의된 작업 클래스와 연관되는 데이터베이스 활동에 대해 변경될 임계값을 지정합니다.

threshold-predicate-clause

유효한 임계값 유형에 대한 설명은 『CREATE THRESHOLD』문을 참조하십시오.

PERFORM ACTION

임계값 술어 조건 값을 변경할 때 임계값 초과 조치가 변경되지 않음을 지정합니다. 작업 조치는 임계값이어야 합니다(SQLSTATE 42613).

alter-threshold-exceeded-actions

유효한 alter-threshold-exceeded-actions에 대한 설명은 『CREATE THRESHOLD』문에서 threshold-exceeded-actions를 참조하십시오.

EXCEEDED

변경된 임계값에 대해 원래 지정된 동일한 임계값 술어를 보존하도록 지정합니다. 작업 조치는 임계값이어야 합니다(SQLSTATE 42613).

PREVENT EXECUTION

작업 조치가 정의된 작업 클래스와 연관되는 데이터베이스 활동 중 어떤 것도 실행할 수 없음을 지정합니다(SQLSTATE 5U033).

COUNT ACTIVITY

작업 클래스와 연관되는 모든 데이터베이스 활동이 실행되고 활동이 실행될 때마다 작업 클래스의 카운터가 증가함을 지정합니다.

COLLECT ACTIVITY DATA

작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가 활동 완료 시 활동 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다.

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

활동 데이터가 활동 코디네이터의 데이터베이스 파티션에서만 수집되도록 지정합니다.

ON ALL DATABASE PARTITIONS

활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. 예측적 임계값의 경우, 초과된 임계값에 CONTINUE 조치도 지정하는 경우에만 모든 파티션에서 활동 정보가 수집됩니다. 반응적 임계값의 경우, ON ALL DATABASE PARTITIONS 절은 효과가 없으며 활동 정보는 항상 코디네이터 파티션에서만 수집됩니다. 예측적 및 반응적 임계값 모두에 대해, 활동 세부사항 또는 값은 코디네이터 파티션에서만 수집됩니다.

WITHOUT DETAILS

작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가 활동 완료 시 활동 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 명령문 및 컴파일 환경 정보는 이벤트 모니터로 보내지 않습니다.

WITH DETAILS

명령문과 컴파일 환경 정보는 이 정보를 가지고 있는 활동에 대한 활동 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다.

AND VALUES

입력 데이터 값은 이 값을 가지고 있는 활동에 대한 활동 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다.

NONE

작업 조치가 정의된 작업 클래스와 연관되는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA

작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. 디폴트는 COLLECT AGGREGATE ACTIVITY DATA BASE입니다. 이 절은 데이터베이스에 적용되는 작업 조치 세트에 정의된 작업 조치에 지정할 수 없습니다.

BASE

작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 기본 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 실행 시간 막대 그래프

EXTENDED

작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 모든 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 DML 계산된 비용 막대 그래프
- 활동 DML 도착 간 시간 막대 그래프

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동의 지속기간(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동이 큐에 대기된 시간 길이(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동이 실행 중인 시간 길이(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 활동이 실행되는 데이터베이스 파티션마다 이 막대 그래프에서 수집됩니다. 활동의 코디네이터 데이터베

이스 파티션에서, 이 시간은 엔드-투-엔드 실행 시간입니다(즉, 수명이 큐에 대기하며 소비된 시간보다 적음). 코디네이터가 아닌 데이터베이스 파티션에서 이 시간은 파티션이 활동 대신 작업에 소비한 시간입니다. 제공된 활동 실행 중, DB2는 리모트 데이터베이스 파티션에 두 번 이상 작업을 제시할 수 있으며, 리모트 파티션은 매번 활동의 해당 어커런스에 대해 실행 시간을 수집합니다. 따라서, 실행 시간 막대 그래프의 계수는 데이터베이스 파티션에서 실행한 실제 고유 활동 수를 나타내지 못할 수도 있습니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*
작업 조치가 지정된 작업 클래스와 연관되는 데이터 처리 언어(DML) 활동의 계산된 비용(timeron 단위)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*
작업 조치가 지정된 작업 클래스와 연관되는 활동에 대해, 하나의 DML 활동이 도착하고 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ENABLE 또는 DISABLE
데이터베이스 활동이 제출될 때 작업 조치가 고려되는지 여부를 지정합니다.

ENABLE
작업 조치가 사용 가능하고 데이터베이스 활동이 제출될 때 고려되도록 지정합니다.

DISABLE
작업 조치가 사용 불가능하고 데이터베이스 활동이 제출될 때 고려되지 않도록 지정합니다.

DROP *work-action-name*
작업 조치 세트에서 작업 조치를 삭제(drop)합니다. *work-action-name*은 현재 서버에서 해당 작업 조치 세트 아래에 존재하는 작업 조치를 식별해야 합니다 (SQLSTATE 42704).

ENABLE 또는 DISABLE

데이터베이스 활동이 제출될 때 작업 조치 세트가 고려되는지 여부를 지정합니다.

ENABLE

작업 조치 세트가 사용 가능하고 데이터베이스 활동이 제출될 때 고려되도록 지정합니다.

DISABLE

작업 조치 세트가 사용 불가능하고 데이터베이스 활동이 제출될 때 고려되지 않도록 지정합니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예:

예 1: DATABASE_ACTIONS 작업 조치 세트를 변경하고 작업 클래스 LARGE_SELECTS를 사용하는 두 개의 작업 조치를 추가하십시오. 작업 조치 ONE_CONCURRENT_SELECT에 대해, 한 번에 실행될 수 있는 활동 수를 제어하기 위해 1 동시성 임계값을 적용하고 최대 세 개까지 큐에 대기될 수 있도록 허용하십시오. 작업 조치 BIG_ROWS_RETURNED에 대해, 해당 클래스 내에 속하는 데이터베이스 활동에서 리턴할 수 있는 행 수를 1 000 000개로 제한하십시오.

```
ALTER WORK ACTION SET DATABASE_ACTIONS
     ADD WORK ACTION ONE_CONCURRENT_SELECT ON WORK CLASS LARGE_SELECTS
        WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 3
 STOP EXECUTION
        ADD WORK ACTION BIG_ROWS_RETURNED ON WORK CLASS LARGE_SELECTS
        WHEN SQLROWSRETURNED > 1000000 STOP EXECUTION
```

예 2: 작업 클래스 SELECT_CLASS 아래의 수퍼 서비스 클래스 ADMIN_APPS에서 실행되는 모든 활동을 서비스 서브클래스 ALL_SELECTS에 맵핑하도록 MAP_SELECTS 작업 조치를 변경하기 위해 ADMIN_APPS_ACTIONS 작업 조치 세트를 변경하십시오. 또한 작업 클래스 UPDATE_CLASS에서 실행되는 모든 활동을 서비스 서브클래스 ALL_SELECT에 맵핑하는 새 작업 조치 MAP_UPDATES도 추가하십시오.

```
ALTER WORK ACTION SET ADMIN_APPS_ACTIONS
  ALTER WORK ACTION MAP_SELECTS MAP ACTIVITY TO ALL_SELECTS
  ADD WORK ACTION MAP_UPDATES ON WORK CLASS UPDATE_CLASS
    MAP ACTIVITY TO ALL_SELECTS
```

ALTER WORK CLASS SET

ALTER WORK CLASS SET문은 작업 클래스 세트 내에서 작업 클래스를 추가, 변경 또는 삭제(drop)합니다.

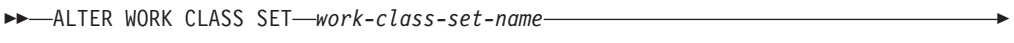
호출

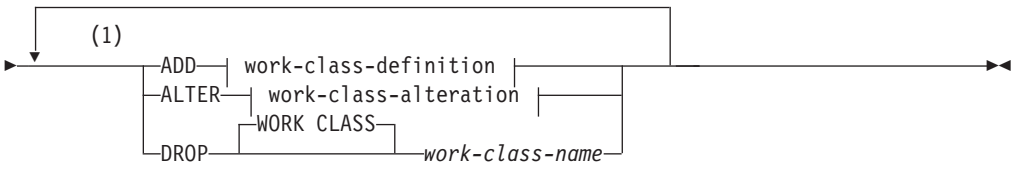
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

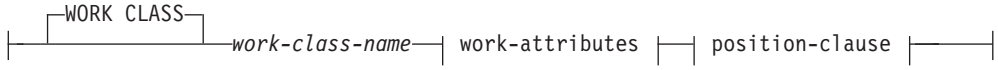
명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

구문

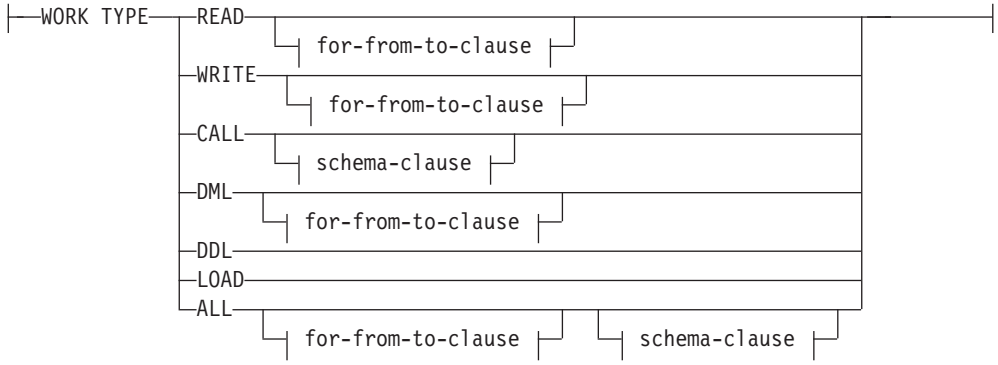




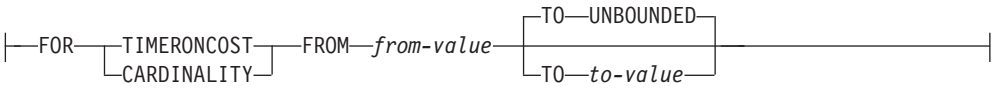
work-class-definition:



work-attributes:

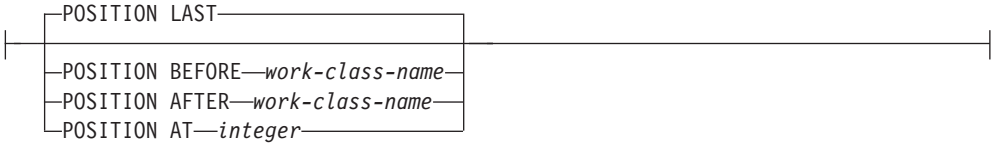


for-from-to-clause:

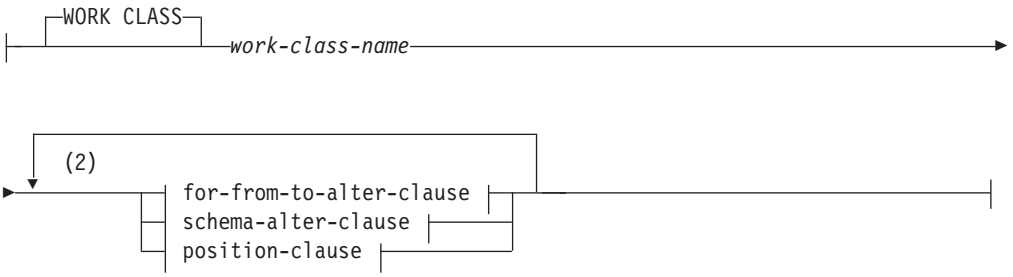


schema-clause:

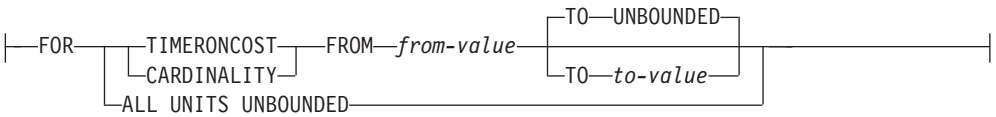
position-clause:



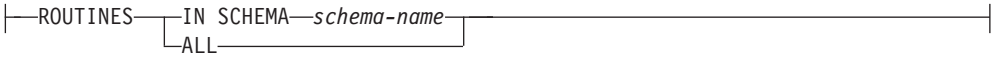
work-class-alteration:



for-from-to-alter-clause:



schema-alter-clause:



주:

1 ADD, ALTER 및 DROP 절은 지정된 순서대로 처리됩니다.

2 같은 절은 두 번 이상 지정될 수 없습니다.

설명

work-class-set-name

변경될 작업 클래스 세트를 식별합니다. 이는 한 부분으로 된 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *work-class-set-name*은 현재 서버에 이미 존재하는 작업 클래스 세트를 식별해야 합니다(SQLSTATE 42704).

ADD

작업 클래스 세트에 작업 클래스를 추가합니다. 세부사항은 『CREATE WORK CLASS SET』를 참조하십시오.

ALTER

작업 클래스 세트 내에서 특정 작업 클래스의 위치와 데이터베이스 활동 속성을 변경합니다.

WORK CLASS *work-class-name*

변경될 작업 클래스를 식별합니다. *work-class-name*은 현재 서버에 있는 작업 클래스 세트 내에 존재하는 작업 클래스를 식별해야 합니다(SQLSTATE 42704).

DROP

작업 클래스 세트에서 작업 클래스를 삭제(drop)합니다.

WORK CLASS *work-class-name*

삭제할 작업 클래스를 식별합니다. *work-class-name*은 현재 서버에 있는 작업 클래스 세트 내에 존재하는 작업 클래스를 식별해야 합니다(SQLSTATE 42704). 작업 클래스 세트와 연관되는 작업 조치 세트 중 하나에 작업 클래스에 종속되는 작업 조치가 있는 경우 작업 클래스를 삭제할 수 없습니다(SQLSTATE 42893).

for-to-from-alter-clause

FOR

FROM *from-value* TO *to-value* 절에 지정되는 정보의 유형을 표시합니다. FOR 절은 다음 작업 유형에 대해서만 사용됩니다.

- READ
- WRITE
- DML
- ALL

TIMERONCOST

작업의 계산된 비용(timeron 단위)입니다. 이 값은 작업이 FROM *from-value* TO *to-value* 절에 지정된 범위 내에 속하는지 여부를 판별하기 위해 사용됩니다.

CARDINALITY

작업의 계산된 카디낼리티(cardinality)입니다. 이 값은 작업이 FROM *from-value* TO *to-value* 절에 지정된 범위 내에 속하는지 여부를 판별하기 위해 사용됩니다.

FROM *from-value* **TO UNBOUNDED 또는 FROM** *from-value* **TO** *to-value*

데이터베이스 활동이 해당 작업 클래스의 일부가 되는 경우 이 활동이 속해야 하는 timeron 값(계산된 비용의 경우) 또는 카디낼리티(cardinality)의 범위를 지정합니다. 범위는 *from-value* - *to-value*입니다. 이 범위는 다음 작업 유형에 대해서만 사용됩니다.

- READ
- WRITE
- DML
- ALL

FROM *from-value* **TO UNBOUNDED**

*from-value*는 0 또는 양수 DOUBLE 값이어야 합니다(SQLSTATE 5U019). 범위에 상한은 없습니다.

FROM *from-value* **TO** *to-value*

*from-value*는 0 또는 양수 DOUBLE 값이어야 하고 *to-value*는 양수 DOUBLE 값이어야 합니다. *from-value*는 *to-value* 이하여야 합니다(SQLSTATE 5U019).

ALL UNITS UNBOUNDED

FROM *from-value* TO *to-value* 절에 범위가 지정되지 않고 지정된 작업 유형에 속하는 모든 작업이 포함됨을 표시합니다.

schema-alter-clause

ROUTINES

이 절은 작업 유형이 CALL 또는 ALL이고 데이터베이스 활동이 CALL 문인 경우에만 사용됩니다.

IN SCHEMA *schema-name*

CALL문이 호출될 프로시저의 스키마 이름을 지정합니다.

ALL

모든 스키마가 포함됨을 지정합니다.

position-clause

POSITION

작업 클래스 세트 내에서 해당 작업 클래스가 배치될 위치를 지정합니다. 이 위치에 따라 작업 클래스가 평가되는 순서가 결정됩니다. 런타임 시 작

업 클래스 지정을 수행하는 경우, 데이터베이스 관리 프로그램은 먼저 오브젝트(데이터베이스 또는 서비스 수퍼 클래스)와 연관되는 작업 클래스 세트를 판별합니다. 작업 클래스 세트에서 첫 번째 일치하는 작업 클래스가 선택됩니다. 이 키워드를 지정하지 않는 경우 작업 클래스는 마지막 위치에 배치됩니다.

LAST

작업 클래스 세트 내에서 순서화된 작업 클래스 목록에서 마지막에 작업 클래스를 배치함을 지정합니다.

BEFORE *work-class-name*

목록에서 작업 클래스 *work-class-name* 앞에 작업 클래스를 배치할 것을 지정합니다. *work-class-name*은 현재 서버에 존재하는 작업 클래스 세트의 작업 클래스를 식별해야 합니다(SQLSTATE 42704).

AFTER *work-class-name*

목록에서 작업 클래스 *work-class-name* 다음에 작업 클래스를 배치할 것을 지정합니다. *work-class-name*은 현재 서버에 존재하는 작업 클래스 세트의 작업 클래스를 식별해야 합니다(SQLSTATE 42704).

AT *position*

작업 클래스 세트 내의 순서화된 작업 클래스 목록에서 작업 클래스가 배치될 절대 위치를 지정합니다. 이 값은 어떤 양수 값도 될 수 있습니다(SQLSTATE 42615). *position*이 기존 작업 클래스 수 + 1보다 클 경우 작업 클래스는 작업 클래스 세트 내의 마지막 위치에 배치됩니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)

- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예:

예 1: 작업 클래스 세트 LARGE_QUERIES를 변경하고 두 개의 기존 작업 클래스가 각각 바운드되지 않은 범위를 유지하면서 범위가 100 000에서 시작하도록 설정하십시오. 10,000 이상의 계산된 timeron 비용을 가지고 있는 모든 SELECT문에 대한 세 번째 작업 클래스를 추가하고 이 작업 클래스가 기존의 두 작업 클래스보다 우선순위가 높도록 위치시키십시오.

```
ALTER WORK CLASS SET LARGE_QUERIES
  ALTER WORK CLASS LARGE_ESTIMATED_COST
   FOR TIMERONCOST FROM 100000 TO UNBOUNDED
  ALTER WORK CLASS LARGE_CARDINALITY
   FOR CARDINALITY FROM 100000 TO UNBOUNDED
  ADD WORK CLASS LARGE_SELECTS WORK TYPE READ
   FOR TIMERONCOST FROM 10000 TO UNBOUNDED POSITION AT 1
```

예 2: DELETE, INSERT, MERGE 또는 UPDATE 문을 포함하는 모든 DML SELECT문을 나타내는 작업 클래스를 추가하도록 작업 클래스 세트 DML_STATEMENTS를 변경하십시오.

```
ALTER WORK CLASS SET DML_STATEMENTS
  ADD WORK CLASS UPDATE_CLASS WORK TYPE WRITE
```

ALTER WORKLOAD

ALTER WORKLOAD문은 워크로드를 변경합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

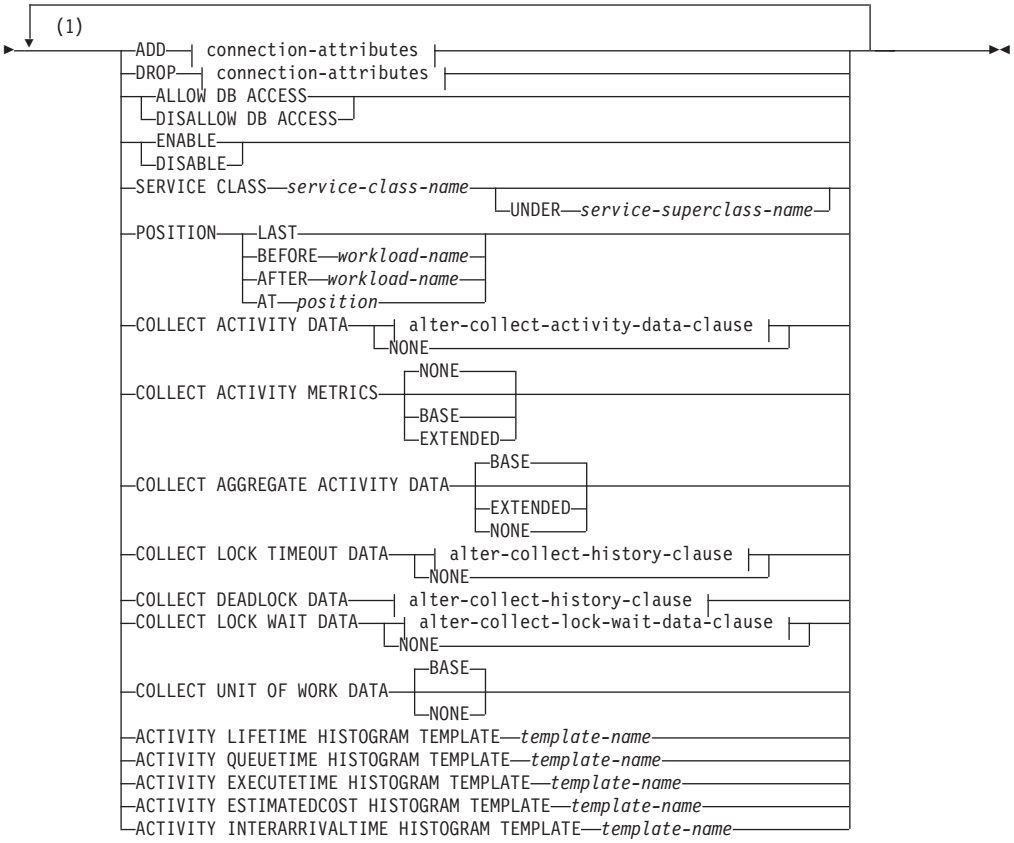
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

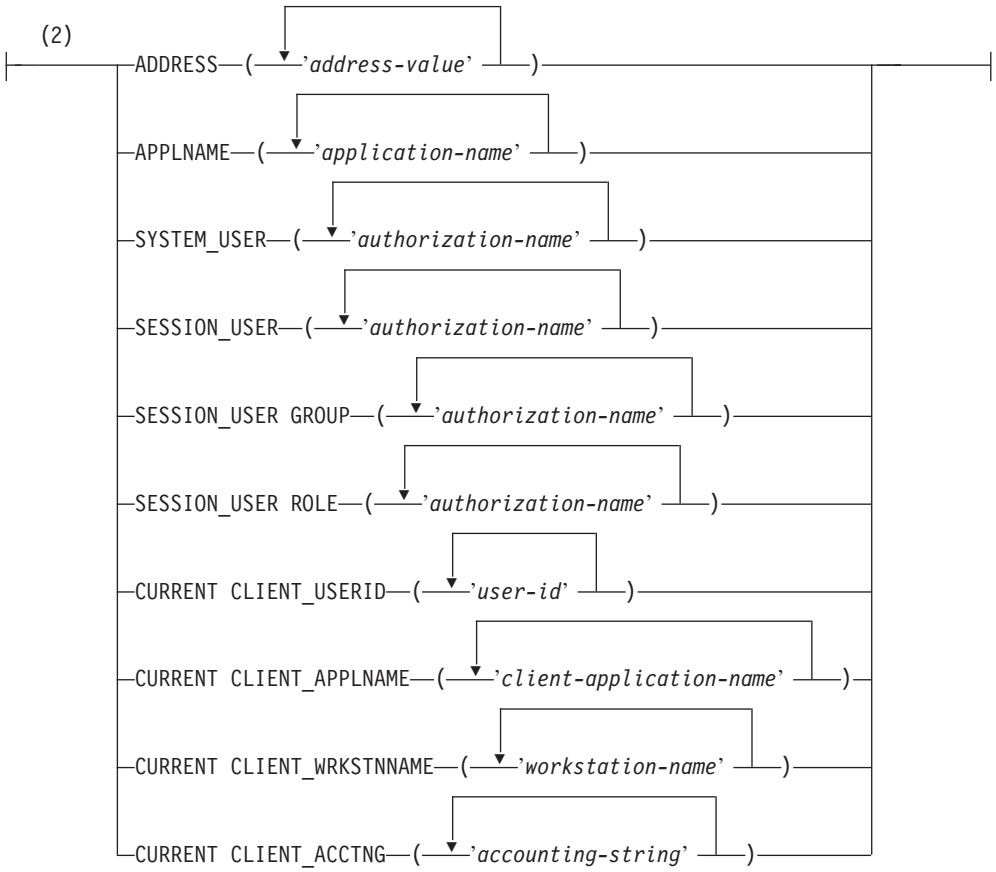
- SQLADM 권한, 모든 변경 절이 COLLECT절인 경우에만
- WLMADM 권한
- DBADM 권한

COLLECT 절이 아닌 다음 절을 지정하려면 명령문의 권한 부여 ID에 DBADM 또는 WLMADM 권한이 포함되어야 합니다.

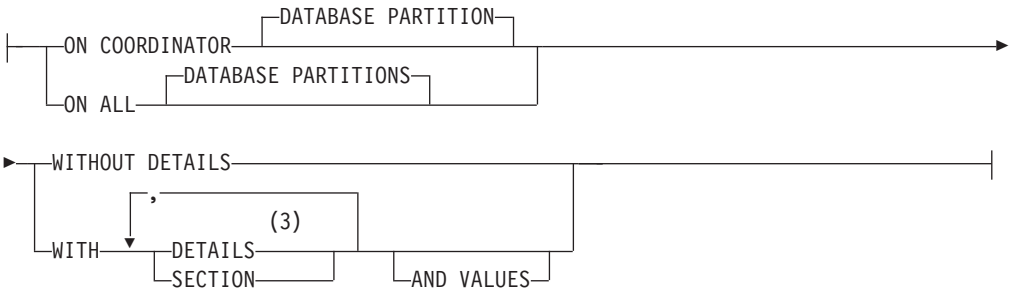
구문



connection-attributes:

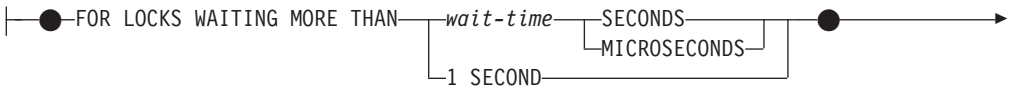


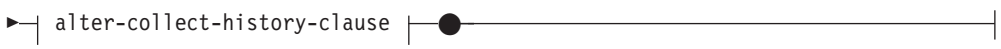
alter-collect-activity-data-clause:



alter-collect-history-clause:

alter-collect-lock-wait-data-clause:





주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

2 Each connection attribute clause can only be specified once.

3 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

workload-name

변경될 워크로드를 식별합니다. 이는 한 부분으로 된 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704).

ADD *connection-attributes*

하나 이상의 연결 속성 값을 워크로드 정의에 추가합니다. 지정된 각각의 연결 속성 값은 워크로드에 대해 이미 정의되어 있으면 안됩니다(SQLSTATE 5U039). *workload-name*이 'SYSDEFAULTUSERWORKLOAD' 또는 'SYSDEFAULTADMWORKLOAD'인 경우 ADD 옵션을 지정할 수 없습니다(SQLSTATE 42832).

DROP *connection-attributes*

하나 이상의 연결 속성 값을 워크로드 정의에서 삭제(drop)합니다. 지정된 각각의 연결 속성 값은 워크로드에 대해 정의되어 있어야 합니다(SQLSTATE 5U040). *workload-name*이 'SYSDEFAULTUSERWORKLOAD' 또는 'SYSDEFAULTADMWORKLOAD'인 경우 DROP 옵션을 지정할 수 없습니다(SQLSTATE 42832). 최소한 하나의 정의된 연결 속성 값이 있어야 합니다. 마지막 연결 속성 값은 삭제할 수 없습니다(SQLSTATE 5U022).

connection-attributes

워크로드의 연결 속성 값을 지정합니다.

ADDRESS ('*address-value***', ...)**

ADDRESS 연결 속성에 대해 하나 이상의 IPv4 주소, IPv6 주소 또는 보안 도메인 이름을 지정합니다. 주소 값은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). 각 주소 값은 IPv4 주소, IPv6 주소 또는 보안 도메인 이름이어야 합니다.

IPv4 주소는 앞 공백을 포함할 수 없으며 점 10진 주소로 나타냅니다. IPv4 주소의 예로 `9.112.46.111`이 있습니다. 값 `localhost` 또는 해당되는 표현 `127.0.0.1`은 결과적으로 일치하지 않습니다. 대신 호스트의 실제 IPv4 주소를 지정해야 합니다. IPv6 주소는 앞 공백을 포함할 수 없으며 콜론 16진 주소로 나타냅니다. IPv6 주소의 예로

2001:0DB8:0000:0000:0008:0800:200C:417A가 있습니다. IPv4 맵핑 IPv6 주소(예: ::ffff:192.0.2.128)는 결과적으로 일치하지 않습니다. 마찬가지로, localhost 또는 해당되는 IPv6 축약 표현 ::1은 결과적으로 일치하지 않습니다. 도메인 이름은 도메인 이름 서버에 의해 결과 IPv4 또는 IPv6 주소가 판별되는 IP 주소로 변환됩니다. 도메인 이름의 예로 corona.torolab.ibm.com이 있습니다. 도메인 이름이 IP 주소로 변환될 때 이 변환의 결과는 하나 이상의 IP 주소로 설정될 수 있습니다. 이 경우, 연결이 시작되는 IP 주소가 도메인 이름이 변환된 IP 주소 중 하나와 일치하면 워크로드 오브젝트의 ADDRESS 속성과 일치함을 의미합니다.

워크로드 오브젝트를 작성할 때 정적 IP 주소 대신 ADDRESS 속성의 도메인 이름 값을 지정해야 합니다(특히 디바이스가 네트워크에 연결할 때마다 다른 IP 주소를 가질 수 있는 DHCP(Dynamic Host Configuration Protocol) 환경에서).

APPLNAME ('*application-name*', ...)

APPLNAME 연결 속성에 대한 하나 이상의 응용프로그램을 지정합니다. 응용프로그램 이름은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *application-name*은 대소문자를 구분하며, 단일 별표 문자(*)를 포함하지 않는 경우 시스템 모니터 출력과 LIST APPLICATIONS 명령의 출력에서 『응용프로그램 이름』 필드에 표시되는 값과 같습니다. *application-name*이 단일 별표 문자(*)를 포함하는 경우, 값은 응용프로그램 이름 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 응용프로그램 이름에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

SYSTEM_USER ('*authorization-name*', ...)

SYSTEM USER 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

SESSION_USER ('*authorization-name*', ...)

SESSION USER 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

SESSION_USER GROUP ('*authorization-name*', ...)

SESSION_USER GROUP 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

SESSION_USER ROLE ('*authorization-name*', ...)

SESSION_USER ROLE 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 이 컨텍스트에서 세션 권한 부여 ID의 역할은 역할을 얻은 방법에 관계없이 세션 권한 부여 ID에 사용 가능한 모든 역할을 의미합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

CURRENT CLIENT_USERID ('*user-id*', ...)

CURRENT CLIENT_USERID 연결 속성에 대한 하나 이상의 클라이언트 사용자 ID를 지정합니다. 클라이언트 사용자 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *user-id*에 단일 별표 문자(*)가 포함되는 경우, 값은 사용자 ID 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 사용자 ID에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

CURRENT CLIENT_APPLNAME ('*client-application-name*', ...)

CURRENT CLIENT_APPLNAME 연결 속성에 대한 하나 이상의 응용프로그램을 지정합니다. 응용프로그램 이름은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *client-application-name*은 대소문자를 구분하며, 단일 별표 문자(*)를 포함하지 않는 경우 시스템 모니터 출력에서 『TP 모니터 클라이언트 응용프로그램 이름』 필드에 표시되는 값과 같습니다. *client-application-name*이 단일 별표 문자(*)를 포함하는 경우, 값은 응용프로그램 이름 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 응용프로그램 이름에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

CURRENT CLIENT_WRKSTNNAME ('*workstation-name*', ...)

CURRENT CLIENT_WRKSTNNAME 연결 속성에 대한 하나 이상의 클라이언트 워크스테이션 이름을 지정합니다. 클라이언트 워크스테이션 이름은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *workstation-name*에 단일 별표 문자(*)가 포함되는 경우, 값은 워크스테이션 이름 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 워크스테이션 이름에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

CURRENT CLIENT_ACCTNG ('*accounting-string*', ...)

CURRENT CLIENT_ACCTNG 연결 속성에 대한 하나 이상의 클라이언트 어카운팅 문자열을 지정합니다. 클라이언트 어카운팅 문자열은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *accounting-string*에 단일 별표 문자(*)가 포함되는 경우, 값은 어카운팅 문자열 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 어카운팅 문자열에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

ALLOW DB ACCESS 또는 DISALLOW DB ACCESS

워크로드와 연관된 워크로드 어커런스가 데이터베이스에 액세스할 수 있는지 여부를 지정합니다.

ALLOW DB ACCESS

워크로드와 연관된 워크로드 어커런스가 데이터베이스에 액세스할 수 있음을 지정합니다.

DISALLOW DB ACCESS

워크로드와 연관된 워크로드 어커런스가 데이터베이스에 액세스할 수 없음을 지정합니다. 워크로드와 연관된 다음 작업 단위가 거부됨을 지정합니다(SQLSTATE 5U020). 이미 실행 중인 워크로드 어커런스는 완료될 수 있습니다. *workload-name*이 'SYSDEFAULTUSERWORKLOAD'인 경우 이 옵션을 지정할 수 없습니다(SQLSTATE 42832).

ENABLE 또는 DISABLE

워크로드가 선택될 때 워크로드가 고려되는지 여부를 지정합니다.

ENABLE

워크로드가 사용 가능하고 워크로드가 선택될 때 고려됨을 지정합니다.

DISABLE

워크로드가 사용 불가능하고 워크로드가 선택될 때 고려되지 않음을 지정합니다. *workload-name*이 'SYSDEFAULTUSERWORKLOAD'인 경우 이 옵션을 지정할 수 없습니다(SQLSTATE 42832).

SERVICE CLASS *service-class-name*

워크로드와 연관된 요청이 *service-class-name* 서비스 클래스에서 실행됨을 지정합니다. *service-class-name*은 현재 서버에 존재하는 서비스 클래스를 식별해야 합니다(SQLSTATE 42704). *service-class-name*은 'SYSDEFAULTSUBCLASS', 'SYSDEFAULTSYSTEMCLASS' 또는 'SYSDEFAULTMAINTENANCECLASS'가 될 수 없습니다(SQLSTATE 5U032). *workload-name*이 'SYSDEFAULTUSERWORKLOAD'인 경우 이 옵션을 지정할 수 없습니다(SQLSTATE 42832).

UNDER *service-superclass-name*

이 절은 서비스 서브클래스를 지정할 때 사용됩니다. *service-superclass-name*은 *service-class-name*의 서비스 수퍼 클래스를 식별합니다. *service-superclass-name*은 현재 서버에 존재하는 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704). *service-superclass-name*은 'SYSDEFAULTSYSTEMCLASS' 또는 'SYSDEFAULTMAINTENANCECLASS'가 될 수 없습니다(SQLSTATE 5U032).

POSITION

순서화된 워크로드 목록에서 해당 워크로드를 배치할 위치를 지정합니다. 런타임 시, 이 목록을 순서대로 검색하여 필수 연결 속성과 일치하는 첫 번째 워크로드를 찾습

니다. *workload-name*이 'SYSDEFAULTUSERWORKLOAD' 또는 'SYSDEFAULTADMWORKLOAD'인 경우 이 옵션을 지정할 수 없습니다 (SQLSTATE 42832).

LAST
디폴트 워크로드 SYSDEFAULTUSERWORKLOAD 및 SYSDEFAULTADMWORKLOAD 이전에, 목록에서 워크로드가 마지막이 됨을 지정합니다.

BEFORE *relative-workload-name*
목록에서 워크로드 *relative-workload-name* 앞에 워크로드를 배치할 것을 지정합니다. *relative-workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). *relative-workload-name*이 'SYSDEFAULTUSERWORKLOAD' 또는 'SYSDEFAULTADMWORKLOAD'인 경우 BEFORE 옵션을 지정할 수 없습니다(SQLSTATE 42832).

AFTER *relative-workload-name*
목록에서 워크로드 *relative-workload-name* 다음에 워크로드를 배치할 것을 지정합니다. *relative-workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). *relative-workload-name*이 'SYSDEFAULTUSERWORKLOAD' 또는 'SYSDEFAULTADMWORKLOAD'인 경우 AFTER 옵션을 지정할 수 없습니다(SQLSTATE 42832).

AT *position*
목록에서 워크로드가 배치될 절대 위치를 지정합니다. 이 값은 어떤 양수 값도 될 수 있습니다(SQLSTATE 42615). *position*이 기존 워크로드 수 + 1보다 클 경우 워크로드는 마지막 위치, SYSDEFAULTUSERWORKLOAD 및 SYSDEFAULTADMWORKLOAD 직전에 위치됩니다.

COLLECT ACTIVITY DATA
해당 워크로드와 연관된 각 활동에 대한 데이터가 활동 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION
활동 데이터가 활동 코디네이터의 데이터베이스 파티션에서만 수집되도록 지정합니다.

ON ALL DATABASE PARTITIONS
활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. 섹션이 지정된 경우, 활동 세부사항 및 섹션 환경은 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. 섹션이 지정되지 않은 경

우 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다.

WITHOUT DETAILS
서비스 클래스에서 실행되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내져야 함을 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

NONE
해당 워크로드와 연관되는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

COLLECT ACTIVITY METRICS
워크로드 어커런스에서 제출된 활동에 대해 모니터 메트릭을 수집해야 함을 지정합니다. 디폴트는 COLLECT ACTIVITY METRICS NONE입니다.

주: 유효한 활동 메트릭 콜렉션 설정은 활동을 제출하는 워크로드에서 COLLECT ACTIVITY METRICS 절로 지정된 속성의 조합이며 MON_ACT_METRICS 데이터베이스 구성 매개변수입니다. 워크로드 속성 또는 구성 매개변수에 NONE이 아닌 다른 값이 있는 경우 메트릭이 활동에 대해 수집됩니다.

NONE
워크로드 어커런스에서 제출된 활동에 대해 메트릭이 수집되지 않음을 지정합니다.

BASE
워크로드 어커런스에서 제출된 활동에 대해 기본 메트릭이 수집됨을 지정합니다.

EXTENDED

워크로드 어커런스에서 제출된 활동에 대해 확장 메트릭이 수집됨을 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA

해당 워크로드와 연관된 활동에 대한 집계 활동 데이터가 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. COLLECT AGGREGATE ACTIVITY DATA를 지정하지 않은 경우 디폴트는 COLLECT AGGREGATE ACTIVITY DATA NONE입니다. COLLECT AGGREGATE ACTIVITY DATA를 지정한 경우 디폴트는 COLLECT AGGREGATE ACTIVITY DATA BASE입니다.

BASE

해당 워크로드와 연관된 활동에 대한 기본 집계 활동 데이터가 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 실행 시간 막대 그래프

EXTENDED

해당 워크로드와 연관된 활동에 대한 모든 집계 활동 데이터가 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 데이터 처리 언어(DML) 계산 비용 막대 그래프
- 활동 DML 도착 간 시간 막대 그래프

NONE

어떤 집계 활동 데이터도 해당 워크로드에 대해 수집되지 않음을 지정합니다.

COLLECT LOCK TIMEOUT DATA

이 워크로드 내에서 발생하는 잠금 시간종료 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 잠금 시간종료 데이터는 모든 파티션에서 수집됩니다. 이 설정은 **MON_LOCKTIMEOUT** 데이터베이스 구성 설정과 결합하여 작동합니다. 가장 상세한 출력을 생성하는 설정이 적용됩니다.

alter-collect-history-clause

WITHOUT HISTORY

이 워크로드 내에서 발생하는 잠금 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 지나간 활동 실행기록 및 입력 값은 이벤트 모니터로 보내지 않습니다.

WITH HISTORY

해당 유형의 모든 잠금 이벤트에 대해 현재 작업 단위(UOW)에 있는 지나간 활동 실행기록을 수집할 것을 지정합니다. 활동 실행기록 버퍼는 최대 크기 한계가 사용된 후 래핑됩니다.

하나의 응용프로그램이 보존할 지나간 활동 수에 대한 디폴트 한계는 250입니다. 지나간 활동 수가 한계보다 큰 경우 최신 활동만 보고됩니다. 이 디폴트값은 다른 값을 지정하기 위한 레지스트리 변수 DB2_MAX_INACT_STMTS를 사용하여 대체할 수 있습니다. 지나간 활동 정보에 사용되는 시스템 모니터 힙 양을 늘리거나 줄이기 위해 한계로 다른 값을 선택할 수 있습니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대한 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 데이터 값에는 LOB 데이터, LONG VARCHAR 데이터, LONG VARGRAPHIC 데이터, 구조화된 유형 데이터 또는 XML 데이터가 포함되지 않습니다. REOPT ALWAYS 바인드 옵션을 사용하여 컴파일된 SQL문의 경우 이벤트 정보에는 REOPT 컴파일 또는 명령문 실행 데이터 값이 제공되지 않습니다.

NONE

워크로드에 대한 잠금 시간종료 데이터가 어떤 파티션에서도 수집되지 않도록 지정합니다.

COLLECT DEADLOCK DATA

이 워크로드 내에서 발생하는 교착 상태 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 교착 상태 데이터는 모든 파티션에서 수집됩니다. 이 설정은 **MON_DEADLOCK** 데이터베이스 구성 매개변수가 NONE으로 설정되지 않는 경우에만 적용됩니다.

alter-collect-history-clause

WITHOUT HISTORY

이 워크로드 내에서 발생하는 잠금 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 지나간 활동 실행기록 및 입력 값은 이벤트 모니터로 보내지 않습니다.

WITH HISTORY

해당 유형의 모든 잠금 이벤트에 대해 현재 작업 단위에 있는 지나간 활동 실행기록을 수집할 것을 지정합니다. 활동 실행기록 버퍼는 최대 *크기* 한계가 사용된 후 래핑됩니다.

하나의 응용프로그램이 보존할 지나간 활동 수에 대한 디폴트 한계는 250입니다. 지나간 활동 수가 한계보다 큰 경우 최신 활동만 보고됩니다. 이 디폴트값은 다른 값을 지정하기 위한 레지스트리 변수 DB2_MAX_INACT_STMTS를 사용하여 대체할 수 있습니다. 지나간 활동 정보에 사용되는 시스템 모니터 힙 양을 늘리거나 줄이기 위해 한계로 다른 값을 선택할 수 있습니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대한 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 데이터 값에는 LOB 데이터, LONG VARCHAR 데이터, LONG VARGRAPHIC 데이터, 구조화된 유형 데이터 또는 XML 데이터가 포함되지 않습니다. REOPT ALWAYS 바인드 옵션을 사용하여 컴파일된 SQL문의 경우 이벤트 정보에는 REOPT 컴파일 또는 명령문 실행 데이터 값이 제공되지 않습니다.

COLLECT LOCK WAIT DATA

해당 워크로드 내에서 발생하는 잠금 대기 이벤트에 대한 데이터가 *wait-time* 내에 잠금이 획득되지 않은 경우 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 설정은 **MON_LOCKWAIT** 및 **MON_LW_THRESH** 데이터베이스 구성 매개변수와 결합하여 작동합니다. 가장 상세한 출력을 생성하는 설정이 적용됩니다.

alter-collect-lock-wait-data-clause

FOR LOCKS WAITING MORE THAN *대기 시간* **SECONDS | MICROSECONDS) | 1 SECOND**

해당 워크로드 내에서 발생하는 잠금 대기 이벤트에 대한 데이터가 *wait-time* 내에 잠금이 획득되지 않은 경우 적용 가능한 이벤트 모니터로 보내지도록 지정합니다.

이 값은 어떤 음수가 아닌 정수도 될 수 있습니다. 유효한 지속기간 키워드를 사용하여 *wait-time*으로 적절한 시간 단위를 지정하십시오. *wait-time* 매개변수의 유효한 최소값은 1000마이크로초입니다.

WITH HISTORY

해당 유형의 모든 잠금 이벤트에 대해 현재 작업 단위(UOW)에 있는 지나간 활동 실행기록을 수집할 것을 지정합니다. 활동 실행기록 버퍼는 최대 *크기* 한계가 사용된 후 래핑됩니다.

하나의 응용프로그램이 보존할 지나간 활동 수에 대한 디폴트 한계는 250입니다. 지나간 활동 수가 한계보다 큰 경우 최신 활동만 보고됩니다. 이 디폴트값은 다른 값을 지정하기 위한 레지스트리 변수 DB2_MAX_INACT_STMTS를 사용하여 대체할 수 있습니다. 지나간 활동 정보에 사용되는 시스템 모니터 힙 양을 늘리거나 줄이기 위해 한계로 다른 값을 선택할 수 있습니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대한 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 데이터 값에는 LOB 데이터, LONG VARCHAR 데이터, LONG VARGRAPHIC 데이터, 구조화된 유형 데이터 또는 XML 데이터가 포함되지 않습니다. REOPT ALWAYS 바인드 옵션을 사용하여 컴파일된 SQL문의 경우 이벤트 정보에는 REOPT 컴파일 또는 명령문 실행 데이터 값이 제공되지 않습니다.

NONE

워크로드에 대한 잠금 대기 이벤트가 어떤 파티션에서도 수집되지 않도록 지정합니다.

COLLECT UNIT OF WORK DATA

해당 워크로드와 연관된 각 작업 단위(트랜잭션이라고도 함)에 대한 데이터가 작업 단위 종료 시 작업 단위 이벤트 모니터(작성된 경우)로 보내지도록 지정합니다. 디폴트값은 COLLECT UNIT OF WORK BASE입니다. **mon_uow_data** 데이터베이스 구성 매개변수가 BASE로 설정된 경우, COLLECT UNIT OF WORK DATA 매개변수보다 우선합니다. **mon_uow_data**의 NONE 값은 개별 워크로드의 COLLECT UNIT OF WORK DATA 매개변수를 사용함을 표시합니다.

BASE

이 워크로드와 연관된 트랜잭션에 대한 데이터의 기본 레벨이 작업 단위(UOW) 이벤트 모니터로 보내지도록 지정합니다.

작업 단위 이벤트로 보고되는 일부 정보는 시스템 레벨 요청 메트릭입니다. 이 메트릭의 콜렉션은 작업 단위 데이터 콜렉션과 독립적으로 제어됩니다. 요청 메트릭은 수퍼 클래스에서 COLLECT REQUEST METRICS절 또는 **mon_req_metrics** 데이터베이스 구성 매개변수를 사용하여 제어됩니다. 워크로드가 연관되는 서비스 수퍼 클래스나, 워크로드가 연관되는 서비스 수퍼 클래스의 서비스 수퍼 클래스는 작업 단위 이벤트에서 있을 요청 메트릭을 위해 요청 메트릭 콜렉션이 사용 가능해야 합니다. 요청 메트릭 콜렉션이 사용 가능하지 않으면 요청 메트릭 값은 영(0)이 됩니다.

NONE

해당 워크로드와 연관된 트랜잭션에 대한 어떤 작업 단위 데이터도 작업 단위 이벤트 모니터로 보내지지 않도록 지정합니다.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*
특정 간격 동안 워크로드에서 실행 중인 DB2 활동의 지속기간(마이크로초)에 대한 통계 데이터를 수집하는 데 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*
워크로드에서 실행 중인 DB2 활동이 특정 간격 동안 큐에서 대기하는 시간 길이(마이크로초)에 대한 통계 데이터를 수집하는 데 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*
워크로드에서 실행 중인 DB2 활동이 특정 간격 동안 실행되는 시간 길이(마이크로초)에 대한 통계 데이터를 수집하는 데 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 코디네이터 데이터베이스 파티션에서만 막대 그래프에서 수집됩니다. 시간에는 유휴 시간이 포함되지 않습니다. 유휴 시간은 어떤 작업도 완료되지 않은 경우에 동일한 활동에 속하는 요청의 실행 사이 시간입니다. 유휴 시간의 예로, 커서 열기가 종료되고 그 커서로부터 페치가 시작되는 시간 사이의 시간을 들 수 있습니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*
워크로드에서 실행 중인 DML 활동의 계산된 비용(timeron 단위)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*
워크로드에 하나의 DML 활동이 도착하고 이 워크로드에 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.

  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다. 새로 제출되는 워크로드 어커런스의 경우 변경사항은 ALTER WORKLOAD문이 커미트된 후에 적용됩니다. 활성 워크로드 어커런스에 대해, 변경사항은 다음 작업 단위가 시작될 때 적용됩니다.
- DISABLE 옵션이 지정된 경우 워크로드는 명령문이 커미트된 후에 사용 불가능하게 됩니다. 워크로드는 다음에 워크로드가 선택될 때 고려되지 않습니다. ALTER WORKLOAD문이 커미트될 때 해당 워크로드와 연관되는 활성 워크로드 어커런스가 있으면 현재 작업 단위가 끝날 때까지 계속 실행됩니다. 다음 작업 단위가 시작될 때, 워크로드 재평가가 발생하고 연결은 다른 워크로드와 연관됩니다.

예:

예 1: DB2가 런타임 시 워크로드를 선택할 때 워크로드 INVENTORY가 먼저 고려되도록 워크로드 PAYROLL이 현재 위치 지정되어 있습니다. PAYROLL이 먼저 고려되도록 평가 순서를 변경하십시오.

```
ALTER WORKLOAD PAYROLL
  POSITION BEFORE INVENTORY
```

예 2: 카탈로그에 있는 다른 워크로드보다 이전에 워크로드 BENCHMARK가 DB2에서 평가되도록 평가 순서를 변경하십시오.

```
ALTER WORKLOAD BENCHMARK
  POSITION AT 1
```

예 3: APPLNAME이 appl1, appl2, appl3으로 설정되고 SYSTEM_USER가 BOB 및 MARY로 설정된 워크로드 REPORTS가 작성되었습니다. 새 응용프로그램 appl4를 응용프로그램 이름 목록에 추가하고 appl2를 제거하도록(더 이상 REPORTS에 맵핑되지 않아서) 워크로드를 변경하십시오.

```
ALTER WORKLOAD REPORTS
  ADD APPLNAME ('appl4')
  DROP APPLNAME ('appl2')
```

예 4: LOCK이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 워크로드 APP 내에서 발생하는 잠금 시간종료 이벤트의 명령문 실행기록으로 잠금 이벤트 레코드를 작성하십시오.

```
ALTER WORKLOAD APP
  COLLECT LOCK TIMEOUT DATA WITH HISTORY
```

예 5: LOCK이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 모든 파티션에서 워크로드 PAYROLL 내에 발생하는 교착 상태 및 잠금 시간종료 이벤트에 대한 잠금 이벤트 레코드를 작성하십시오.

```
ALTER WORKLOAD PAYROLL
  COLLECT DEADLOCK DATA
  COLLECT LOCK TIMEOUT DATA WITHOUT HISTORY
```

예 6: LOCK이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 워크로드 INVOICE 내에서 발생하는 교착 상태 이벤트의 명령문 실행기록 및 값으로 잠금 이벤트 레코드를 작성하십시오.

```
ALTER WORKLOAD INVOICE
  COLLECT DEADLOCK DATA WITH HISTORY AND VALUES
```

예 7: LOCK이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 워크로드 INVOICE 내에서 발생하는, 150밀리초보다 오래 기다린 후 획득된 잠금에 대한 명령문 실행기록 및 값으로 잠금 이벤트 레코드를 작성하십시오.

```
ALTER WORKLOAD INVOICE
  COLLECT LOCK WAIT DATA FOR LOCKS WAITING MORE THAN 150000
  MICROSECONDS WITH HISTORY AND VALUES
```

예 8: 작업 단위 데이터를 수집하여 작업 단위 이벤트 모니터로 보내도록 워크로드 REPORTS를 변경하십시오.

```
ALTER WORKLOAD REPORTS
  COLLECT UNIT OF WORK DATA BASE
```

ALTER WRAPPER

ALTER WRAPPER문은 랩퍼의 등록 정보를 갱신하는 데 사용됩니다.

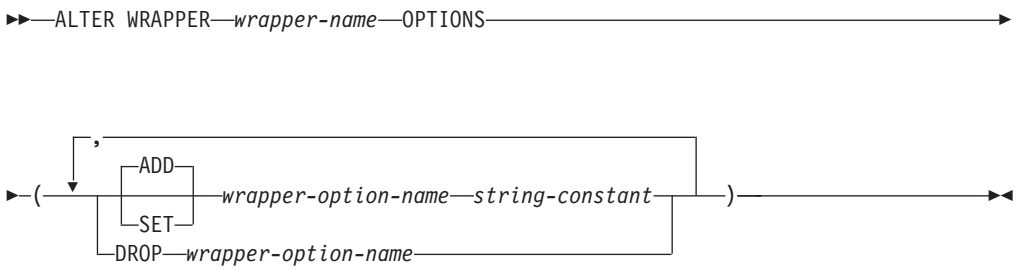
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

구문



설명

wrapper-name
랩퍼 이름을 지정하십시오.

OPTIONS
사용 가능, 재설정 또는 삭제(drop)될 랩퍼 옵션을 나타내십시오.

ADD
서버 옵션을 활성화합니다.

SET
랩퍼 옵션의 설정값을 변경하십시오.

wrapper-option-name
작동 가능화되거나 재설정될 랩퍼 옵션에 이름을 지정하십시오. 현재 유일하게 지원되는 랩퍼 옵션 이름은 DB2_FENCED입니다.

string-constant
*wrapper-option-name*에 대한 설정값을 문자열 상수로서 지정하십시오. 유효한 값은 'Y' 또는 'N'입니다. 관계형 랩퍼의 디폴트값은 'N'이고 비관계형 랩퍼의 디폴트값은 'Y'입니다.

DROP *wrapper-option-name*
랩퍼 옵션을 삭제(drop)하십시오.

주

- ALTER WRAPPER문의 실행에는 랩퍼 특정 옵션에 대한 유효성 점검이 포함되지 않습니다.
- 주어진 작업 단위(UOW) 내의 ALTER WRAPPER문은 UOW에 다음 중 하나가 이미 포함된 경우에는 처리될 수 없습니다(SQLSTATE 55007).
  - 랩퍼에 속하는 별칭을 참조하는 SELECT문.
  - 랩퍼에 속하는 별칭의 열린 커서.
  - 랩퍼에 속하는 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문.

예:

예 1: DB2_FENCED 옵션을 랩퍼 NET8에 대해 설정합니다.

```
ALTER WRAPPER NET8 OPTIONS (SET DB2_FENCED 'Y')
```

ALTER XSROBJECT

이 명령문은 특정 XML 스키마에 대한 분석 지원을 사용 또는 사용 불가능하게 하는 데 사용됩니다. 해당 XML 스키마에 대해 분석을 사용할 수 있는 경우 XML 문서를 관계형 테이블로 분석하기 위해 주석이 표시된 XML 스키마를 사용할 수 있습니다.

호출

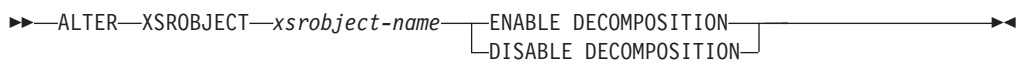
ALTER XSROBJECT문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때만 (SQLSTATE 42509) 동적으로 준비될 수 있는 실행문입니다.

권한 부여

다음 권한 중 하나가 필요합니다.

- DBADM
- SQL 스키마에 대한 ALTERIN
- 변경될 XSR 오브젝트의 소유권

구문



설명

xsrobject-name

변경될 XSR 오브젝트를 식별합니다. 내재적 또는 명시적 스키마 규정자를 포함하는 *xsrobject-name*은 현재 서버에 있는 기존 XSR 오브젝트를 고유하게 식별해야 합니다. 이 ID의 XSR 오브젝트가 없는 경우 오류가 리턴됩니다(SQLSTATE 42704).

ENABLE DECOMPOSITION 또는 DISABLE DECOMPOSITION

XSR 오브젝트를 분석에 사용 또는 사용할 수 없도록 합니다. 식별된 XSR 오브젝트는 XML 스키마여야 합니다(SQLSTATE 42809). 분석을 사용할 수 있도록 하려면 XML 스키마에 분석 규칙에 대한 주석이 표시되어야 하며(SQLSTATE 225DE) 분석 규칙에서 참조되는 오브젝트가 현재 서버에 존재해야 합니다 (SQLSTATE 42704).

참고:

- XSR 오브젝트에 대해 분석을 사용할 수 없을 때 모든 관련 카탈로그 항목은 제거됩니다.

- XSR 오브젝트가 종속된 오브젝트(예: 테이블)이 삭제되거나 XSR 오브젝트와 호환되지 않도록 변경될 경우 XSR 오브젝트에 대해 분석 지원을 사용할 수 없게 됩니다.
- 파티션된 데이터베이스 환경에서 임의의 파티션에 연결하여 이 명령문을 실행할 수 있습니다.

ASSOCIATE LOCATORS

ASSOCIATE LOCATORS문은 스토어드 프로시저가 리턴하는 각 결과 세트에 대한 결과 세트 로케이터 값을 가져옵니다.

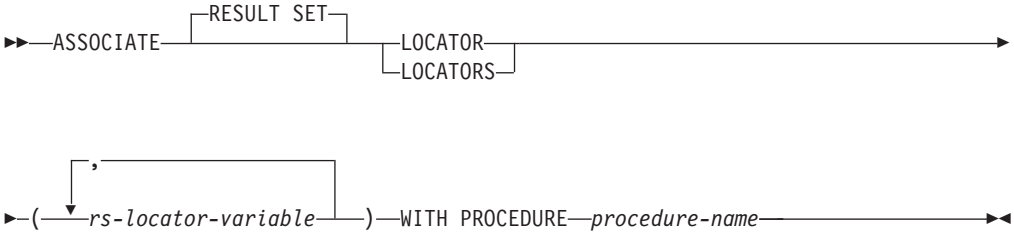
호출

이 명령문은 SQL 프로시저에만 임베드될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

권한 부여

필요한 권한 없음

구문



설명

rs-locator-variable

복합 SQL(프로시저)문에 선언된 결과 세트 로케이터 변수를 지정합니다.

WITH PROCEDURE

지정한 프로시저 이름별로 결과 세트 로케이터를 리턴하는 스토어드 프로시저를 식별합니다.

procedure-name

프로시저 이름은 규정화된 또는 규정되지 않은 이름입니다.

완전한 프로시저 이름은 두 파트로 구성된 이름입니다. 첫 번째 파트는 스토어드 프로시저의 스키마 이름이 포함된 ID입니다. 마지막 파트는 스토어드 프로시저 이름이 포함된 ID입니다. 마침표로 각 파트를 구분해야 합니다. 파트 일부 또는 전체는 분리 ID일 수 있습니다.

규정되지 않은 프로시저 이름의 경우 내재적 스키마 이름이 프로시저 이름에 규정자로 추가되지 않기 때문에 한 개의 이름만 갖습니다. ASSOCIATE LOCATORS문을 제대로 실행하려면 명령문에서 규정되지 않은 프로시저 이름이 규정되지 않은 프로시저 이름으로 지정한 최근에 실행한 CALL문의 프로시

저 이름과 동일해야 합니다. CALL문에서 규정되지 않은 이름에 대한 내재된 스키마 이름은 일치하는 것으로 고려되지 않습니다. 프로시저 이름 지정 방법에 대한 규칙은 아래에서 설명됩니다.

ASSOCIATE LOCATORS문이 실행되면 프로시저 이름 또는 스펙이 CALL문을 사용하여 리퀘스터를 이미 호출한 스토어드 프로시저를 식별해야 합니다. ASSOCIATE LOCATORS문의 프로시저 이름은 CALL문에서 지정된 동일한 방식으로 지정되어야 합니다. 예를 들어 두 파트 이름이 CALL문에 지정된 경우 ASSOCIATE LOCATORS문에도 두 파트의 이름을 사용해야 합니다.

주

- ASSOCIATE LOCATORS문에 나열된 결과 세트 로케이터 변수의 개수가 스토어드 프로시저에서 리턴하는 로케이터 수보다 적은 경우, 명령문의 모든 변수에 하나의 값이 지정되고 경고가 발행됩니다.
- ASSOCIATE LOCATORS문에 나열된 결과 세트 로케이터 변수의 개수가 스토어드 프로시저에서 리턴하는 로케이터 수보다 큰 경우, 여분의 변수에 0 값이 지정됩니다.
- 동일한 호출자에서 두 번 이상 스토어드 프로시저가 호출된 경우 최근의 결과 세트에만 액세스할 수 있습니다.
- 결과 세트 로케이터 값은 이전에 PREPARE문으로 준비한 CALL문을 실행하는 EXECUTE문을 사용하여 호출되는 프로시저에 대해 사용할 수 있습니다. 그렇지만 결과 세트 로케이터 값은 EXECUTE IMMEDIATE문을 사용하여 호출되는 프로시저에 대해서는 사용할 수 없습니다.
- ASSOCIATE LOCATORS문에서 참조되는 module-procedure 이름은 1 파트 또는 2 파트의 규정된 이름 참조만 될 수 있습니다. 3 파트의 이름 참조는 허용되지 않습니다(SQLSTATE 42601). ASSOCIATE LOCATORS문에서 참조한 module-procedure를 참조하는 모든 CALL문은 ASSOCIATE LOCATORS문에서 사용되는 동일한 1 파트 또는 2 파트의 규정된 이름이 있는 module-procedure를 지정해야 합니다.

예

다음 예의 명령문은 SQL 프로시저에 임베드되는 것으로 간주됩니다.

예 1: 결과 세트 로케이터 변수 LOC1 및 LOC2를 사용하여 스토어드 프로시저 P1이 리턴한 두 개의 결과 세트에 대한 결과 세트 로케이터 값을 가져오십시오. 1 파트 이름으로 스토어드 프로시저가 호출되는 것으로 가정합니다.

```
CALL P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
  WITH PROCEDURE P1;
```

예 2: 예 1의 시나리오를 반복하지만 2 파트의 이름을 사용하여 MYSCHEMA 스키마의 스토어드 프로시저 P1이 사용되도록 스토어드 프로시저에 대한 명시적 스키마 이름을 지정하십시오.

```
CALL MYSCHEMA.P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
  WITH PROCEDURE MYSCHEMA.P1;
```

AUDIT

AUDIT문은 현재 서버에서 특정 데이터베이스나 데이터베이스 오브젝트에 사용되는 감사 규정을 판별합니다. 오브젝트가 사용 중일 때마다 해당 규정에 따라 감사를 받습니다.

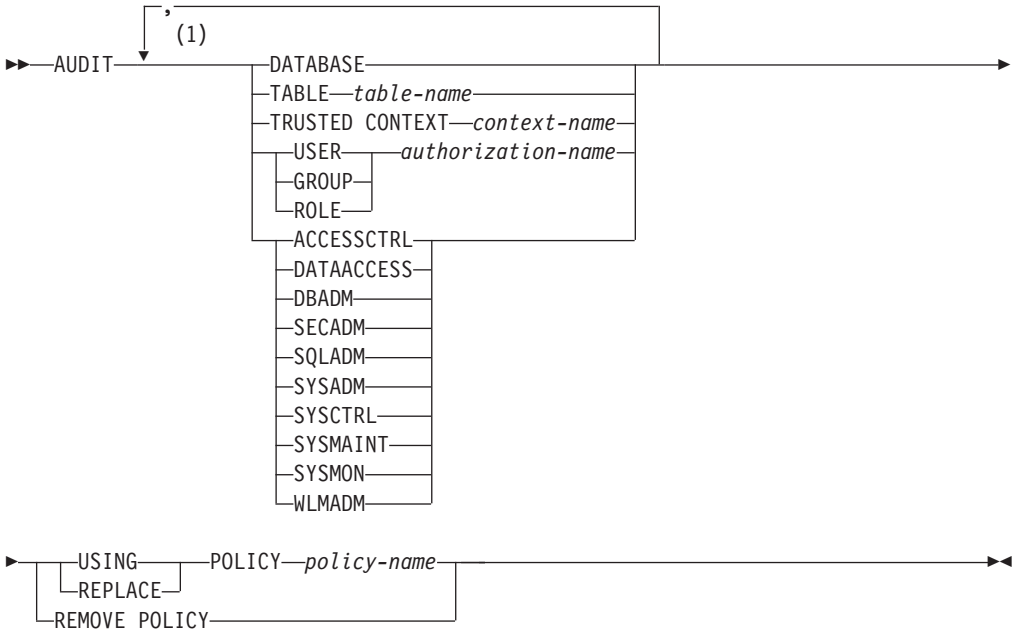
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



주:

1 각각의 절(적용 가능한 경우 오브젝트 이름이 동일한)은 최대한 한 번 지정할 수 있습니다(SQLSTATE 42713).

설명

ACCESSCTRL, DATAACCESS, DBADM, SECADM, SQLADM, SYSADM, SYSCTRL, SYSMAINT, SYSMON 또는 WLMADM

감사 규정이 지정된 권한과 연관되거나 지정된 권한에서 제거됨을 지정합니다. 지정

된 권한이 이벤트에 필요하지 않은 경우에도 해당 권한을 보유하는 사용자가 시작하는 모든 감사 가능한 이벤트는 연관된 감사 규정에 따라 감사를 받습니다.

DATABASE

감사 규정이 현재 서버에서 데이터베이스와 연관되거나 데이터베이스에서 제거됨을 지정합니다. 데이터베이스 내에서 발생하는 모든 감사 가능한 이벤트는 연관된 감사 규정에 따라 감사를 받습니다.

TABLE *table-name*

감사 규정이 *table-name*과 연관되거나 table-name에서 제거됨을 지정합니다. *table-name*은 현재 서버에 존재하는 테이블, 구체화된 쿼리 테이블(MQT) 또는 별칭을 식별해야 합니다(SQLSTATE 42704). 이름은 뷰, 카탈로그 테이블, 작성된 임시 테이블, 선언된 임시 테이블(SQLSTATE 42995) 또는 유형이 지정된 테이블(SQLSTATE 42997)이 될 수 없습니다. 규정에 다른 범주를 감사해야 된다고 표시되어 있어도, 테이블에 액세스할 때 데이터가 있거나 없는 EXECUTE 범주 감사 이벤트만 생성됩니다.

TRUSTED CONTEXT *context-name*

감사 규정이 *context-name*과 연관되거나 context-name에서 제거됨을 지정합니다. *context-name*은 현재 서버에 존재하는 트러스트된 컨텍스트를 식별해야 합니다(SQLSTATE 42704). 트러스트된 컨텍스트 *context-name*으로 정의된 트러스트된 연결 내에서 발생하는 모든 감사 가능한 이벤트는 연관된 감사 규정에 따라 감사를 받습니다.

USER *authorization-name*

감사 규정이 *authorization-name* 권한 부여 ID를 가지고 있는 사용자와 연관되거나 사용자에서 제거됨을 지정합니다. *authorization-name*이 시작하는 모든 감사 가능한 이벤트는 연관된 감사 규정에 따라 감사를 받습니다.

GROUP *authorization-name*

감사 규정이 *authorization-name* 권한 부여 ID를 가지고 있는 그룹과 연관되거나 그룹에서 제거됨을 지정합니다. *authorization-name*의 구성원인 사용자가 시작하는 모든 감사 가능한 이벤트는 연관된 감사 규정에 따라 감사를 받습니다. 그룹에서의 사용자 멤버쉽을 판별할 수 없는 경우 규정은 해당 사용자에게 적용되지 않습니다.

ROLE *authorization-name*

감사 규정이 *authorization-name* 권한 부여 ID를 가지고 있는 역할과 연관되거나 역할에서 제거됨을 지정합니다. *authorization-name*은 현재 서버에 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). *authorization-name*의 구성원인 사용자가 시작하는 모든 감사 가능한 이벤트는 연관된 감사 규정에 따라 감사를 받습니다. 다른 역할 또는 그룹을 통한 간접 역할 멤버쉽이 유효합니다.

USING, REMOVE 또는 REPLACE

지정된 오브젝트에 대해 감사 규정을 사용, 제거 또는 교체해야 함을 지정합니다.

USING
감사 규정이 지정된 오브젝트에 사용됨을 지정합니다. 기존의 감사 규정은 오브젝트에 대해 이미 정의되어 있으면 안됩니다(SQLSTATE 5U041). 감사 규정이 이미 존재하면 제거하거나 교체해야 합니다.

REMOVE
감사 규정이 지정된 오브젝트에서 제거됨을 지정합니다. 오브젝트 사용은 더 이상 감사 규정에 따라 감사를 받지 않습니다. 감사 규정이 오브젝트에서 제거될 때 카탈로그에서 연관이 삭제됩니다.

REPLACE
감사 규정이 지정된 오브젝트에 대한 기존 감사 규정을 교체함을 지정합니다. 이는 REMOVE 및 USING 옵션 둘 다를 하나의 단계로 조합하여 감사 규정이 지정된 오브젝트에 적용되지 않는 시간이 없도록 합니다. 지정된 오브젝트에 규정이 사용되고 있지 않는 경우 REPLACE는 USING과 같습니다.

POLICY *policy-name*
감사 설정을 판별하기 위해 사용될 감사 규정을 지정합니다. *policy-name*은 현재 서버에서 기존 감사 규정을 식별해야 합니다(SQLSTATE 42704).

규칙

- AUDIT 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). AUDIT 독점 SQL문은 다음과 같습니다.
  - AUDIT
  - CREATE AUDIT POLICY, ALTER AUDIT POLICY 또는 DROP(AUDIT POLICY)
  - DROP(감사 규정과 연관되는 경우 ROLE 또는 TRUSTED CONTEXT)
- AUDIT 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).
- 하나의 오브젝트는 단 하나의 규정과 연관될 수 있습니다(SQLSTATE 5U042).

주

- 변경사항은 카탈로그에 기록되지만, COMMIT문이 실행될 때까지 적용되지 않습니다.
- 변경사항은 감사 규정이 적용되는 오브젝트를 참조하는 다음 작업 단위가 발생할 때까지 적용되지 않습니다. 예를 들어, 데이터베이스에 대해 감사 규정을 사용 중인 경우 현재 작업 단위는 COMMIT 또는 ROLLBACK문이 완료될 때까지 규정에 따라 감사를 시작하지 않습니다.
- 감사 규정과 연관되는 테이블에 액세스하는 뷰는 기본 테이블 규정에 따라 감사를 받습니다.

- 테이블에 적용되는 감사 규정은 해당 테이블을 기초로 하는 구체화된 쿼리 테이블(MQT)에 적용되지 않습니다. 감사 규정을 테이블과 연관시키는 경우 해당 테이블을 기초로 하는 MQT와도 규정을 연관시키도록 하십시오. SQL문이 기본 테이블을 참조해도 컴파일러가 자동으로 MQT를 사용할 수 있습니다. 그러나 기본 테이블에 사용 중인 감사 규정은 계속 적용됩니다.
- 트러스트된 컨텍스트에서 사용자 전환 조작이 수행되는 경우, 모든 감사 규정은 새 사용자에 따라 다시 평가되고, 이전 사용자의 규정은 현재 세션에서 사용되지 않습니다. 이는 특히 사용자, 사용자의 그룹 또는 역할, 멤버쉽 및 사용자의 권한과 직접 연관되는 감사 규정에 적용됩니다. 예를 들어, 이전 사용자가 감사 대상 역할의 구성원이어서 현재 세션이 감사를 받았지만 전환 대상 사용자가 해당 역할의 구성원이 아닌 경우 규정은 더 이상 세션에 적용되지 않습니다.
- SET SESSION USER문이 실행될 때 원래 사용자(및 해당 사용자의 그룹 및 역할 멤버쉽과 권한)와 연관되는 감사 규정은 SET SESSION USER문에 지정된 사용자와 연관되는 규정과 조합됩니다. 원래 사용자와 연관되는 감사 규정은 계속 적용됩니다. SET SESSION USER문에 지정된 사용자에 대한 규정이기 때문입니다. 하나의 세션 내에서 여러 개의 SET SESSION USER문이 실행되는 경우 원래 사용자와 현재 세션에 연관되는 감사 규정만 고려됩니다.
- 감사 규정이 연관되는 오브젝트가 삭제되면, 감사 추적에 대한 연관성이 카탈로그에서 제거되어 더 이상 존재하지 않습니다. 해당 오브젝트가 나중에 다시 작성되어도, 오브젝트가 삭제될 때 연관되었던 규정에 따라 감사하지 않습니다.

예:

예 1: 현재 서버에서 데이터베이스에 대한 감사 설정을 판별하기 위해 감사 규정 DBAUDPRF를 사용하십시오.

```
AUDIT DATABASE USING POLICY DBAUDPRF
```

예 2: EMPLOYEE 테이블에서 감사 규정을 제거하십시오.

```
AUDIT TABLE EMPLOYEE REMOVE POLICY
```

예 3: 권한 SYSADM, DBADM 및 SECADM과 그룹 DBAS에 대한 감사 설정을 판별하기 위해 감사 규정 POWERUSERS를 사용하십시오.

```
AUDIT SYSADM, DBADM, SECADM, GROUP DBAS USING POLICY POWERUSERS
```

예 4: 역할 TELLER에 대한 감사 규정을 새 규정 TELLERPRF로 교체하십시오.

```
AUDIT ROLE TELLER REPLACE POLICY TELLERPRF
```

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION 문은 호스트 변수 선언 섹션의 시작을 표시합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이는 실행문이 아닙니다. REXX에 지정해서는 안됩니다.

권한 부여

필요한 권한 없음

구문

설명

BEGIN DECLARE SECTION 문은 호스트 언어의 규칙에 따라 변수 선언이 나타날 수 있는 응용프로그램에서 코딩할 수 있습니다. 호스트 변수 선언 섹션의 시작을 표시하기 위해 사용됩니다. 호스트 변수 섹션은 END DECLARE SECTION 문으로 종료됩니다.

규칙

- BEGIN DECLARE SECTION 및 END DECLARE SECTION 문은 쌍이어야 하며 중첩될 수 없습니다.
- SQL문은 선언 섹션에 포함될 수 없습니다.
- SQL문에 참조된 호스트 변수는 REXX가 아닌 다른 모든 호스트 언어로 호스트 변수 선언 섹션에서 선언해야 합니다. 또한 섹션은 변수에 대한 첫 번째 참조 이전에 있어야 합니다. 일반적으로, 호스트 변수는 LOB 로케이터 및 파일 참조 변수는 제외하고 REXX에서 선언되지 않습니다. 이 경우에, 호스트 변수는 BEGIN DECLARE SECTION 내에 선언되지 않습니다.
- 선언 섹션 밖에서 선언된 변수는 선언 섹션 내에서 선언된 변수와 동일한 이름을 가질 수 없습니다.
- LOB 데이터 유형에는 데이터 유형 및 길이가 수반되고 앞에 SQL TYPE IS 키워드가 붙습니다.

예:

예 1: 호스트 변수 hv_smint(smallint), hv_vchar24(varchar(24)), hv_double (double), hv_blob_50k (blob(51200)), hv_struct(blob(10240)로서 구조화된 유형 ″struct_type″의)를 C 프로그램으로 정의하십시오.

```
 EXEC SQL  BEGIN DECLARE SECTION;
short  hv_smint;
      struct {
short hv_vchar24_len;
char  hv_vchar24_value[24];
}                              hv_vchar24;
double hv_double;
SQL TYPE IS BLOB(50K) hv_blob_50k;
SQL TYPE IS struct_type AS BLOB(10k) hv_struct;
 EXEC SQL  END DECLARE SECTION;
```

예 2: 호스트 변수 HV-SMINT(smallint), HV-VCHAR24(varchar(24)), HV-DEC72 (dec(7,2)) 및 HV-BLOB-50k(blob(51200))를 COBOL 프로그램으로 정의하십시오.

```
WORKING-STORAGE SECTION.
      EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
01  HV-SMINT              PIC S9(4)       COMP-4.
01  HV-VCHAR24.
    49 HV-VCHAR24-LENGTH  PIC S9(4)       COMP-4.
    49 HV-VCHAR24-VALUE   PIC X(24).
01  HV-DEC72              PIC S9(5)V9(2)  COMP-3.
01 HV-BLOB-50K            USAGE SQL TYPE IS BLOB(50K).
      EXEC SQL END DECLARE SECTION  END-EXEC.
```

예 3: 호스트 변수 HVSMINT(smallint), HVVCHAR24(char(24)), HVDOUBLE(double) 및 HVBLOB50k(blob(51200))를 Fortran 프로그램으로 정의하십시오.

```
EXEC SQL  BEGIN DECLARE SECTION
  INTEGER*2      HVSMINT
  CHARACTER*24   HVVCHAR24
  REAL*8         HVDOUBLE
  SQL TYPE IS BLOB(50K)  HVBLOB50K
EXEC SQL END DECLARE SECTION
```

주: Fortran에서, 예상 값이 254바이트보다 큰 경우, CLOB 호스트 변수를 사용해야 합니다.

예 4: 호스트 변수 HVSMINT (smallint), HVBLOB50K(blob(51200)) 및 HVCLOBLOC(CLOB 로케이터)를 REXX 프로그램으로 정의하십시오.

```
DECLARE :HVCLOBLOC  LANGUAGE TYPE CLOB LOCATOR
call sqlexec 'FETCH c1 INTO :HVSMINT, :HVBLOB50K'
```

변수 HVSMINT 및 HVBLOB50K는 FETCH문에서 사용하여 내재적으로 정의되었다는 점에 유의하십시오.

CALL

CALL문은 프로시저 또는 외부 프로시저를 호출합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다. 명령행 처리기를 사용하여 호출 시, 프로시저의 인수를 지정하는 데에는 일부 추가된 규칙이 있습니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

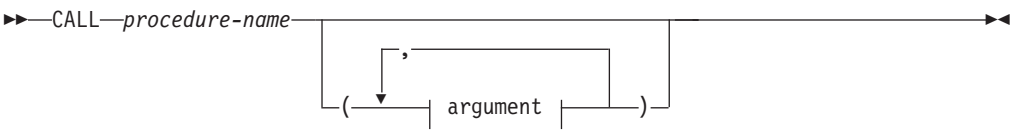
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 권한 중 하나를 가지고 있어야 합니다.

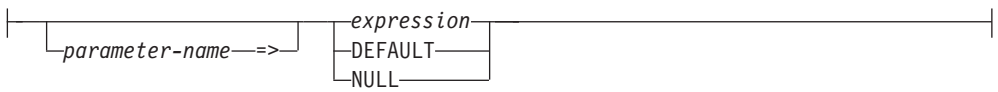
- 프로시저에 대한 EXECUTE 권한
- DATAACCESS 권한

명령문의 권한 부여 ID에 실행 권한이 없는데 일치하는 프로시저가 있으면 오류가 리턴됩니다(SQLSTATE 42501).

구문



인수:



설명

procedure-name

호출될 프로시저를 지정합니다. 이는 카탈로그에 기술된 프로시저여야 합니다. 호출할 특정 프로시저는 프로시저 분석을 사용하여 선택됩니다. 자세한 내용은 이 명령문의 『주』를 참조하십시오.

argument

parameter-name

인수가 지정된 매개변수의 이름입니다. 인수를 이름으로 매개변수에 지정하면, 다음의 모든 인수도 이름으로 지정되어야 합니다(SQLSTATE 4274J).

프로시저 호출에서 이름 지정된 매개변수 모두가 프로시저 정의에 존재해야 합니다(SQLSTATE 4274J).

이름 지정된 매개변수는 한 번만 지정되어야 합니다(내재적으로나 명시적으로나)(SQLSTATE 4274J).

이름 지정된 매개변수는 카탈로그 해제된 프로시저에 대한 호출에서 지원되지 않습니다(SQLSTATE 4274J).

expression 또는 **DEFAULT** 또는 **NULL**

expression, DEFAULT 키워드 또는 NULL 키워드의 각 스펙은 CALL의 인수입니다. CALL문의 *n*번째 이름 지정되지 않은 인수는 프로시저에 대한 CREATE PROCEDURE문에 정의된 *n*번째 매개변수에 해당합니다.

이름 지정된 인수는 지정된 순서와 상관 없이 같은 이름의 매개변수에 해당됩니다.

DEFAULT 키워드가 지정되면, CREATE PROCEDURE문에서 정의된 디폴트가 사용됩니다(있는 경우). 그렇지 않은 경우 디폴트로 널(NULL) 값이 사용됩니다.

NULL 키워드가 지정되면, 널(NULL) 값이 매개변수 값으로 전달됩니다.

CALL문의 각 인수는 다음과 같이 프로시저 정의의 해당 매개변수와 호환되어야 합니다.

- IN 매개변수
  - 인수는 매개변수에 지정 가능해야 합니다.
  - 문자열 인수를 지정하면 저장 지정 규칙이 사용됩니다.
- OUT 매개변수
  - 인수는 단일 변수 또는 매개변수 표시문자여야 합니다(SQLSTATE 42886).
  - 인수는 매개변수에 지정 가능해야 합니다.
  - 문자열 인수를 지정하면 검색 지정 규칙이 사용됩니다.
- INOUT 매개변수
  - 인수는 단일 변수 또는 매개변수 표시문자여야 합니다(SQLSTATE 42886).
  - 인수는 매개변수에 지정 가능해야 합니다.
  - 문자열 인수를 지정하면 호출할 때는 저장 지정 규칙이 사용되고 리턴할 때는 검색 지정 규칙이 사용됩니다.

규칙

- 프로시저 호출 시 디폴트 값을 가지도록 정의되지 않은 모든 매개변수에 대해 인수가 지정되어 있어야 합니다(SQLSTATE 428HF).

주

매개변수 지정: CALL문이 실행될 때 각 인수 값은 스토리지 지정을 사용하여 해당 프로시저 매개변수에 지정됩니다. 프로시저 호출 시, 디폴트 값을 가지도록 정의된 매개변수 값은 인수 목록에서 생략할 수 있습니다.

CALL문이 실행되면 호출하는 호스트 언어 규칙에 따라 제어권이 프로시저에 넘겨집니다. 프로시저 실행이 완료되면 프로시저의 각 매개변수 값이 OUT 또는 INOUT으로 정의된 CALL문의 해당 인수에 스토리지 지정을 사용하여 지정됩니다. 프로시저에서 오류가 리턴되면 OUT 인수의 정의가 취소되고 INOUT 인수는 변경되지 않습니다. 지정 규칙에 대한 자세한 내용은 『지정 및 비교』를 참조하십시오.

CALL statement문이 SQL 프로시저에 있고 다른 SQL 프로시저를 호출하고 있는 경우, XML 매개변수 할당이 참조 순으로 이루어집니다. XML 인수가 참조 순으로 전달되는 경우, 입력 노드 트리가 있으면 XML 인수로부터 직접 사용되고 문서 순서, 원본 노드 식별 및 모든 상위 등록 정보를 포함하는 모든 등록 정보를 보존합니다.

프로시저 서명: 프로시저는 스키마, 프로시저 이름 및 매개변수 개수로 식별됩니다. 이를 프로시저 시그니처라고 하는데, 데이터베이스 내에서 고유해야 합니다. 매개변수 수가 각 프로시저마다 서로 다르게 제공되면, 스키마에는 같은 이름으로 된 프로시저가 여러 개 있을 수 있습니다.

SQL 경로: 프로시저는 괄호로 묶인 선택적 인수 목록이 뒤에 오는 규정된 이름(스키마 및 프로시저 이름)을 참조하여 호출될 수 있습니다. 스키마 이름없이 프로시저를 호출할 수도 있는데, 이 때는 동일한 개수의 매개변수가 있는 다른 스키마에서 가능한 프로시저가 선택됩니다. 이 경우, 프로시저 분석을 지원하는 데 SQL 경로가 사용됩니다. SQL 경로는 이름과 매개변수 개수가 동일한 프로시저를 식별하기 위해 검색되는 스키마 목록입니다. 정적 CALL문의 경우, SQL 경로는 FUNCPATH 바인드 옵션을 사용하여 지정됩니다. 동적 CALL문의 경우, SQL 경로는 CURRENT PATH 특수 레지스터의 값입니다.

프로시저 분석: 프로시저가 호출되면, 데이터베이스 관리 프로그램에서는 동일한 이름의 가능한 프로시저 중 어느 것이 최적인지 결정해야 합니다.

- *A*는 프로시저 호출의 인수 수입니다.
- *P*는 프로시저 서명에서 매개변수의 수입니다.
- *N*은 디폴트가 없는 매개변수의 수입니다.

프로시저 호출의 분석을 위한 후보 프로시저에는 일치하는 이름 및 *적용 가능한* 매개변수의 수가 있습니다. 적용 가능한 매개변수의 수는 $N \le A \le P$라는 조건에 부합합니다. 후보 프로시저 세트는 프로시저가 호출되는 환경 및 프로시저 이름이 규정되는 방법에 따라 다릅니다.

- 프로시저 이름이 규정되지 않은 경우, 프로시저 분석은 다음 단계를 사용하여 수행됩니다.
  1. 규정되지 않은 프로시저가 모듈 오브젝트 내에서 호출되면, 후보 프로시저의 모듈 내에서 검색하십시오. 하나 이상의 후보 프로시저가 컨텍스트 모듈에서 발견되면 SQL 경로의 스키마에 있는 후보 프로시저와 함께 이 후보 프로시저가 포함됩니다.(다음 항목 참조)
  2. 후보 프로시저의 SQL 경로에 있는 스키마와 함께 CALL 명령문의 권한 부여 ID에 EXECUTE 특권이 없는 후보 프로시저를 제외한 모든 스키마 프로시저를 검색하십시오. 하나 이상의 후보 프로시저가 SQL 경로의 스키마에서 발견되면 컨텍스트 모듈의 후보 프로시저와 함께 이 후보 프로시저가 포함됩니다.(이전 항목 참조) 단일 후보 프로시저가 남아 있으면 분석이 완료됩니다. 다중 후보 프로시저가 있으면, 낮은 수의 매개변수가 있는 후보 프로시저를 판별하고 많은 수의 매개변수가 있는 후보 프로시저를 제거하십시오. 여러 개의 후보 프로시저가 있는 경우 프로시저가 계속 후보 프로시저이면 컨텍스트 모듈에서 프로시저를 선택하고 그렇지 않은 경우 SQL 경로에서 가장 첫 번째 스키마의 프로시저를 선택합니다.

  2단계 이후에 남아 있는 후보 함수가 없는 경우, 오류가 리턴됩니다(SQLSTATE 42884).
- 프로시저 이름이 규정되지 않으면, 프로시저 분석은 다음 단계에 따라 수행됩니다.
  1. 프로시저가 모듈 내에서 호출되고 규정자가 프로시저가 호출한 모듈의 이름과 일치하면, 후보 프로시저의 모듈 내에서 검색하십시오. 규정자가 단일 ID이면, 모듈 이름을 일치시킬 때 모듈의 스키마 이름이 무시됩니다. 규정자가 두 파트의 ID이면, 일치를 판별할 때 스키마 규정 모듈 이름과 비교합니다. 단일 후보 프로시저가 존재하면 분석이 완료됩니다. 다중 후보 프로시저가 있으면, 가장 작은 수의 매개변수가 있는 후보 프로시저를 선택하십시오. 규정자가 일치하지 않거나 후보 프로시저가 없는 경우, 다음 단계를 계속하십시오.
  2. 규정자를 스키마 이름으로 간주하고 CALL 명령문의 권한 부여 ID에 EXECUTE 특권이 없는 후보 프로시저를 제외한 후보 프로시저의 해당 스키마를 검색하십시오. 단일 후보 프로시저가 존재하면 분석이 완료됩니다. 다중 후보 프로시저가 있으면 가장 작은 수의 매개변수가 있는 후보 프로시저를 선택하고 분석이 완료됩니다. 스키마가 존재하지 않거나 권한이 부여된 후보 프로시저가 없는 경우 그리고 첫 번째 단계에서 규정자가 모듈의 이름과 일치하는 경우 오류를 리턴합니다. 그렇지 않으면, 다음 단계를 계속하십시오.
  3. 모듈에 대한 EXECUTE 특권을 고려하지 않고 모듈 이름으로 규정자를 고려하십시오.
     - 모듈 이름이 스키마 이름으로 규정된 경우, 후보 프로시저에 대해 이 모듈 내에서 발행된 프로시저를 검색하십시오.

- 모듈 이름이 스키마 이름으로 규정되지 않으면, 모듈의 스키마는 모듈 이름과 일치하는 SQL 경로에 있는 첫 번째 스키마입니다. 프로시저가 발견되면, 후보 프로시저에 대해 이 모듈 내에서 발행된 프로시저를 검색하십시오.
- SQL 경로를 사용하여 모듈을 찾을 수 없으면, 프로시저 규정자의 이름과 일치하는 모듈 공용 별명을 점검하십시오. 발견되면, 후보 프로시저에 대해 이 모듈 내에서 발행된 프로시저를 검색하십시오.

일치하는 모듈을 찾을 수 없으며 일치하는 모듈에 후보 프로시저가 없으면, 프로시저 찾을 수 없음 오류가 리턴됩니다(SQLSTATE 42884). 다중 후보 프로시저가 있으면, 가장 작은 수의 매개변수가 있는 후보 프로시저를 선택하십시오. CALL문의 권한 부여 ID에 나머지 후보 프로시저에 대한 EXECUTE 특권이 있는 경우, 분석이 완료됩니다. 그렇지 않으면, 권한 부여 오류가 리턴됩니다 (SQLSTATE 42501).

프로시저가 분석되는 동안 호출 시 지정되는 *parameter-names*가 고려되지 않음을 유의하십시오. 분석된 프로시저에 지정된 이름의 매개변수가 없거나 일치하는 이름의 매개변수가 이름이 지정되지 않은 매개변수가 사용하는 위치 다음에 있으면, 오류가 리턴됩니다(SQLSTATE 4274J).

SQL 프로시저의 DB2_RETURN_STATUS 검색: SQL 프로시저가 상태 값과 함께 RETURN문을 성공적으로 발행할 경우, 이 값은 SQLCA의 첫 번째 SQLERRD 필드에 리턴됩니다. CALL문이 SQL 프로시저에서 실행되면, GET DIAGNOSTICS문을 사용하여 DB2_RETURN_STATUS 값을 검색하십시오. SQLSTATE가 오류를 나타낼 경우, 값은 -1입니다. 오류가 리턴되지 않고 프로시저에 RETURN문이 지정되지 않은 경우에는 값이 0입니다.

프로시저의 결과 세트 리턴: 호출 프로그램이 CLI, JDBC 또는 SQLJ를 사용하여 작성되거나 호출자가 SQL 프로시저인 경우, 결과 세트는 직접 호출자로 리턴될 수 있습니다. 프로시저는 해당 결과 세트에서 커서를 선언하고 결과 세트에서 커서를 연 다음 프로시저를 종료할 때 커서를 열린 상태로 유지함으로써 결과 세트가 리턴될 것임을 나타냅니다.

프로시저 끝에서:

- 열린 상태인 모든 커서의 경우 결과 세트가 호출자 또는 클라이언트(WITH RETURN TO CLIENT 커서의 경우)에게 직접 리턴됩니다.
- 읽어들이지 않은 행만 다시 전달됩니다. 예를 들어, 커서의 결과 세트에 500개의 행이 있는데 이 행 중에서 150개의 행이 프로시저에 의해 읽힌 경우, 프로시저가 종료될 때 151부터 500까지의 행이 호출자 또는 응용프로그램(해당하는 경우)에 리턴됩니다.

CLI 또는 JDBC에서 프로시저가 호출되고 둘 이상의 커서가 열려 있을 경우, 결과 세트는 커서가 열린 순서대로만 처리될 수 있습니다.

성능 개선: 모든 인수 값은 응용프로그램에서 프로시저로 전달됩니다. 이 조작의 성능을 향상시키려면 CALL문을 실행하기 전에 OUT 매개변수에 해당하고 길이가 몇 바이트 이상인 호스트 변수를 널(NULL)로 설정해야 합니다.

CALL문 중첩: 프로시저는 응용프로그램은 물론 루틴에서도 호출할 수 있습니다. 루틴에서 프로시저를 호출할 경우, 호출이 중첩된 것으로 간주됩니다.

프로시저가 쿼리 결과 세트를 리턴할 경우에는 결과 세트가 다음과 같이 리턴됩니다.

- RETURN TO CALLER 결과 세트는 이전의 중첩 레벨에 있는 프로그램에만 표시됩니다.
- RETURN TO CLIENT 결과 세트는 프로시저를 중첩 프로시저 세트에서 호출한 경우에만 표시됩니다. 호출 체인 내에서 함수 또는 메소드가 발생하면 결과 세트가 표시되지 않습니다. 결과 세트가 표시되더라도 프로시저를 처음 호출한 클라이언트 응용프로그램에만 표시됩니다.

다음 예를 고려하십시오.

```
Client program:
EXEC SQL CALL PROCA;

    PROCA:
    EXEC SQL CALL PROCB;

        PROCB:
        EXEC SQL DECLARE B1 CURSOR WITH RETURN TO CLIENT ...;
        EXEC SQL DECLARE B2 CURSOR WITH RETURN TO CALLER ...;
        EXEC SQL DECLARE B3 CURSOR FOR SELECT UDFA FROM T1;

            UDFA:
            EXEC SQL CALL PROCC;

                PROCC:
                EXEC SQL DECLARE C1 CURSOR WITH RETURN TO CLIENT ...;
                EXEC SQL DECLARE C2 CURSOR WITH RETURN TO CALLER ...;
```

PROCB 프로시저에서:

- 커서 B1은 클라이언트 응용프로그램에만 표시되고 PROCA 프로시저에는 표시되지 않습니다.
- 커서 B2는 PROCA에 표시되고 클라이언트에는 표시되지 않습니다.

PROCC 프로시저에서

- 커서 C1은 UDFA나 클라이언트 응용프로그램 모두에 표시되지 않습니다. UDFA는 클라이언트와 PROCC 사이의 호출 체인에 나타나므로 결과 세트가 클라이언트에 리턴되지 않습니다.
- 커서 C2는 UDFA에만 표시되고 상위 프로시저에는 표시되지 않습니다.

트리거, 복합 명령문, 함수 또는 메소드 내에서의 프로시저 중첩: 트리거, 복합 명령문, 함수 또는 메소드 내에서 프로시저를 호출할 경우

- 프로시저는 COMMIT 또는 ROLLBACK문을 발행하면 안됩니다.
- 프로시저에서 리턴된 결과 세트에는 액세스할 수 없습니다.
- 프로시저가 READS SQL DATA 또는 MODIFIES SQL DATA로 정의되어 있으면, 프로시저의 어떤 명령문도 프로시저를 호출한 명령문에서 수정 중인 테이블에 액세스할 수 없습니다(SQLSTATE 57053). 프로시저가 MODIFIES SQL DATA로 정의되어 있으면, 프로시저의 어떤 명령문도 프로시저를 호출한 명령문에서 읽거나 수정 중인 테이블을 수정할 수 없습니다(SQLSTATE 57053).

함수 또는 메소드에서 프로시저를 호출할 경우

- 호출 함수나 메소드와 동일한 테이블 액세스 제한사항이 프로시저에 적용됩니다.
- 함수나 메소드를 호출하기 전에 정의된 세이브포인트는 프로시저에 표시되지 않고, 프로시저 내에 정의된 세이브포인트는 함수나 메소드 외부에서 볼 수 없습니다.
- 프로시저에서 리턴된 RETURN TO CLIENT 결과 세트는 클라이언트에서 액세스할 수 없습니다.

호환성: CALL_RESOLUTION DEFERRED 옵션으로 응용프로그램을 프리컴파일하여 응용프로그램에 임베디드될 수 있는 CALL문의 이전 양식이 있습니다. 이 옵션은 SQL 프로시저 및 페더레이티드 프로시저에서 사용할 수 없습니다.

예:

예 1: Java™ 프로시저는 다음 명령문을 사용하는 데이터베이스에서 정의됩니다.

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                OUT COST DECIMAL(7,2),
                OUT QUANTITY INTEGER)
                EXTERNAL NAME 'parts!onhand'
                LANGUAGE JAVA
                PARAMETER STYLE DB2GENERAL;
```

Java 응용프로그램은 다음과 같은 코드 조각을 사용하여 이 프로시저를 호출합니다.

```
...
   CallableStatement stpCall;

   String sql = "CALL PARTS_ON_HAND (?, ?, ?)";
stpCall = con.prepareCall(sql); /*con is the connection */

   stpCall.setInt(1, hvPartnum);
   stpCall.setBigDecimal(2, hvCost);
   stpCall.setInt(3, hvQuantity);

   stpCall.registerOutParameter(2, Types.DECIMAL, 2);
   stpCall.registerOutParameter(3, Types.INTEGER);

   stpCall.execute();
```

```
    hvCost = stpCall.getBigDecimal(2);
    hvQuantity = stpCall.getInt(3);
...
```

외부 이름 parts!onhand를 갖는 CALL문에 지정된 프로시저 이름을 데이터베이스에서 찾게 되므로, 해당 응용프로그램 코드 조각은 클래스 parts에서 Java 메소드 onhand를 호출합니다.

예 2: 네 개의 서로 다른 스키마에 6개의 FOO 프로시저가 다음과 같이 등록된 상태입니다(모든 필수 키워드를 표시한 것은 아님).

```
CREATE PROCEDURE AUGUSTUS.FOO (INT) SPECIFIC FOO_1 ...
   CREATE PROCEDURE AUGUSTUS.FOO (DOUBLE, DECIMAL(15, 3)) SPECIFIC FOO_2 ...
   CREATE PROCEDURE JULIUS.FOO (INT) SPECIFIC FOO_3 ...
   CREATE PROCEDURE JULIUS.FOO (INT, INT, INT) SPECIFIC FOO_4 ...
   CREATE PROCEDURE CAESAR.FOO (INT, INT) SPECIFIC FOO_5 ...
   CREATE PROCEDURE NERO.FOO (INT,INT) SPECIFIC FOO_6 ...
```

프로시저 참조는 다음과 같습니다. 여기서, I1 및 I2는 INTEGER 값입니다.

```
   CALL FOO(I1, I2)
```

이 참조를 구성하는 응용프로그램에 다음과 같이 설정될 수 있는 SQL 경로가 있다고 가정해 보십시오.

```
    "JULIUS","AUGUSTUS","CAESAR"
```

위 알고리즘에 따르면...

스키마 "NERO"가 SQL 경로에 포함되지 않으므로, 구체적인 이름이 FOO_6인 프로시저는 후보에서 제거됩니다. FOO_1, FOO_3, FOO_4에는 잘못된 개수의 매개변수가 있으므로 후보에서 제거됩니다. 나머지 후보는 SQL 경로에 나타난 대로 순서가 지정됩니다. 인수 및 매개변수의 유형은 무시됩니다. 매개변수 FOO_5가 CALL의 인수와 정확하게 일치하지만 SQL 경로에서 "AUGUSTUS"가 "CAESAR" 앞에 나타나므로 FOO_2가 선택됩니다.

예 3: 다음 프로시저가 존재한다고 가정하십시오.

```
  CREATE PROCEDURE update_order(
                      IN IN_POID BIGINT,
                      IN IN_CUSTID BIGINT DEFAULT GLOBAL_CUST_ID,
                      IN NEW_STATUS VARCHAR(10) DEFAULT NULL,
                      IN NEW_ORDERDATE DATE DEFAULT NULL,
                      IN NEW_COMMENTS VARCHAR(1000)DEFAULT NULL)...
```

전역 변수 GLOBAL_CUST_ID가 값 1002로 설정되었다고도 가정하십시오. 프로시저를 호출하여 고객 1002에 대한 주문 5000의 상태를 'Shipped'로 변경합니다. 디폴트에 대한 나머지 인수를 널 값으로 허용하여 나머지 데이터 순서를 있는 그대로 둡니다.

```
   CALL update_order(5000, NEW_STATUS => 'Shipped')
```

ID 1001인 고객이 호출되고 구입 주문 5002에 대한 출시를 수신하여 충족되었다는 것을 표시합니다. 주문을 갱신합니다.

```
CALL update_order(5002,
    IN_CUSTID => 1001,
    NEW_STATUS => 'Received',
    NEW_COMMENTS => 'Customer satisfied with the order.')
```

CASE

CASE문은 여러 조건을 기준으로 실행 경로를 선택합니다. 이 명령문을 CASE 표현식과 혼동해서는 안됩니다. CASE 표현식은 하나 이상의 조건을 평가하여 그것을 기준으로 표현식을 선택할 수 있습니다.

호출

이 명령문은 다음과 같이 임베드될 수 있습니다.

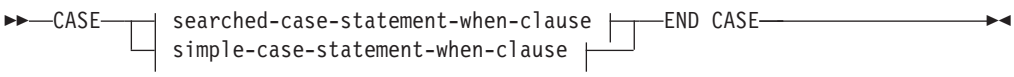
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 SQL문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베드될 수 있습니다. CASE문은 실행문이 아니며 동적으로 준비될 수 없습니다.

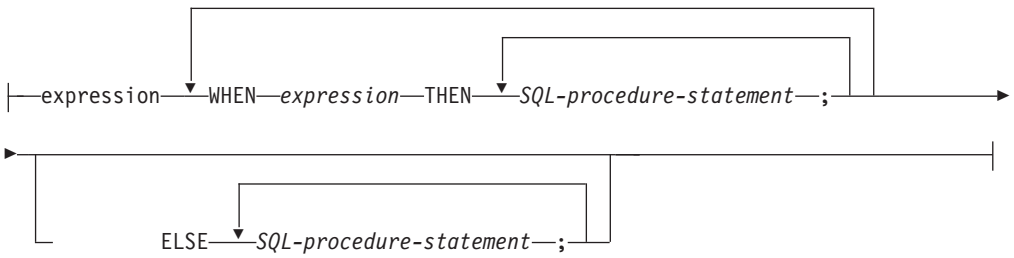
권한 부여

CASE문을 호출하기 위해 필요한 특권은 없습니다. 단, CASE문으로 임베드되는 SQL문을 호출하기 위해 필요한 모든 특권이 명령문의 권한 부여 ID에 의해 보유된 특권에 포함되어야 합니다.

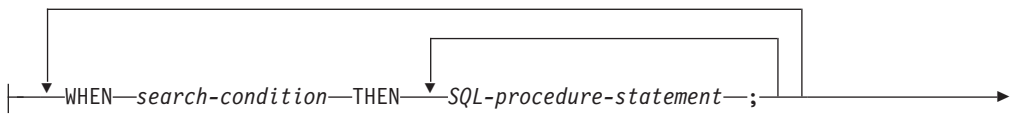
구문

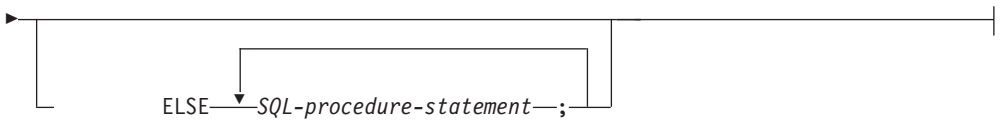


simple-case-statement-when-clause:



searched-case-statement-when-clause:





설명

CASE

*case-statement*를 시작합니다.

simple-case-statement-when-clause

첫 번째 WHEN 키워드 이전의 *expression* 값은 WHEN 키워드 뒤에 오는 각 *expression* 값과 함께 등식으로 테스트됩니다. 검색 조건이 참이면, THEN문이 실행됩니다. 결과가 알 수 없음 또는 거짓이면, 다음 검색 조건으로 처리를 계속합니다. 결과가 검색 조건과 일치하지 않고, ELSE절이 존재하면, ELSE절의 명령문이 처리됩니다.

searched-case-statement-when-clause

WHEN 키워드 뒤에 오는 *search-condition*이 평가됩니다. 참으로 평가되면, 연관된 THEN절의 명령문이 처리됩니다. 거짓 또는 알 수 없음으로 평가되면, 다음 *search-condition*이 평가됩니다. *search-condition* 이 참으로 평가되지 않거나 ESLE 절이 존재하면, ELSE절의 명령문이 처리됩니다.

SQL-procedure-statement

호출해야 할 명령문을 지정합니다. 『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

END CASE

*case-statement*를 종료합니다.

주

- WHEN에 지정된 조건이 참이 아니고, ELSE절이 지정되지 않으면, 런타임 시 오류가 발행되고, CASE문의 실행이 종료됩니다(SQLSTATE 20000).
- CASE문이 실행 가능 조건을 모두 다루는지 확인하십시오.

예

SQL 변수 `v_workdept`의 값에 따라, DEPARTMENT의 DEPTNAME 컬럼을 적절한 이름으로 갱신하십시오.

다음 예는 *simple-case-statement-when-clause*의 구문을 사용하여 이를 수행하는 방법을 표시합니다.

```
CASE v_workdept
  WHEN'A00'
    THEN UPDATE department
    SET deptname = 'DATA ACCESS 1';
  WHEN 'B01'
    THEN UPDATE department
    SET deptname = 'DATA ACCESS 2';
  ELSE UPDATE department
    SET deptname = 'DATA ACCESS 3';
END CASE
```

다음 예는 *searched-case-statement-when-clause*의 구문을 사용하여 이를 수행하는 방법을 표시합니다.

```
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
```

CLOSE

CLOSE문은 커서를 닫습니다. 커서가 열려 있을 때 결과 테이블이 작성되면, 해당 테이블은 삭제됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 명령행 처리기를 사용하여 호출 시, 일부 옵션을 지정할 수 없습니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

권한 부여

전역 변수가 참조되면, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 READ 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

커서를 사용하기 위해 필요한 권한 부여에 대해서는 『DECLARE CURSOR』를 참조하십시오.

구문

설명

cursor-name

닫으려는 커서를 식별합니다. *cursor-name*은 DECLARE CURSOR문에 설명된 대로, 선언된 커서를 식별해야 합니다. 따라서 CLOSE문이 실행될 때 커서는 열린 상태여야 합니다.

cursor-variable-name

닫으려는 커서를 식별합니다. *cursor-variable-name*은 커서 변수를 식별해야 합니다. CLOSE문이 실행될 때, *cursor-variable-name*의 기본 커서는 열린 상태여야 합니다(SQLSTATE 24501). *cursor-variable-name*을 사용한 CLOSE문은 복합 SQL(컴파일된) 명령문 내에서만 사용될 수 있습니다.

WITH RELEASE

커서에 대해 설정된 모든 잠금의 해제가 시도됩니다. 모든 잠금이 반드시 해제될 필요는 없습니다. 이들 잠금은 다른 조작이나 활동에 대해 설정될 수도 있습니다.

주

- 작업 단위 마지막 부분에서, WITH HOLD 옵션없이 선언된 응용프로그램 프로세스에 속하는 모든 커서는 내재적으로 닫힙니다.
- 사용되지 않는 커서가 될 때 커서 변수의 기반 커서는 내재적으로 닫힙니다. 더 이상 커서 변수의 기본 커서가 되지 못할 때 기본 커서는 사용되지 못하게 됩니다. 예를 들어, 기본 커서의 모든 커서 변수가 동일 범위에 있고 모두가 동시에 범위 밖에 있으면 이러한 상황이 발생합니다.
- 함수나 메소드에 정의된 커서를 닫을 때 WITH RELEASE절의 영향을 받지 않습니다. 그리고 함수나 메소드에서 호출한 프로시저에 정의된 커서를 닫을 때도 이 절이 적용되지 않습니다.
- WITH RELEASE절은 분리 레벨 CD나 UR에서 작동되는 커서에 대해 영향을 미치지 않습니다. 분리 레벨 RS나 RR에서 작동되는 커서에 대해 지정된 경우, WITH RELEASE는 해당 분리 레벨에서 보장하는 내용 중 일부를 종료합니다. 특히, 그 커서가 다시 열리면, RS 커서의 경우 '반복 불가능한 읽기' 현상이 발생할 수 있으며, RR 커서의 경우 '반복 불가능한 읽기' 또는 '팬텀' 현상이 발생할 수도 있습니다.

  원래 RR 또는 RS인 커서가 WITH RELEASE절을 사용하여 닫힌 후에 다시 열린 경우, 새 잠금이 설정됩니다.
- 호출 프로그램으로 리턴되기 전에 닫히지 않은 프로시저 내에 있는 커서에 특별한 규칙이 적용됩니다.
- 커서가 열려 있을 때(즉, 커서가 아직 닫히지 않았을 때) 이 커서와 관련된 명령문의 결과로 시퀀스 값이 변경되더라도(예: 시퀀스에 대한 NEXT VALUE 표현식을 포함하는 커서를 사용하는 FETCH 또는 UPDATE) 이 커서가 표시하는 해당 시퀀스에 대하여 PREVIOUS VALUE로 갱신되지 않습니다. 영향을 받는 이러한 시퀀스의 PREVIOUS VALUE 값은 CLOSE문을 사용하여 커서가 명시적으로 닫힐 때 갱신됩니다. 파티션된 데이터베이스 환경에서 커서가 커미트 또는 롤백을 통해 암시적으로 닫히면, PREVIOUS VALUE는 시퀀스의 가장 최근에 생성된 값으로 갱신되지 않을 수도 있습니다.

예 :

커서는 한 번에 한 행을 C 프로그램 변수 dnum, dname 및 mnum로 페치할 때 사용됩니다. 마지막으로 커서가 닫힙니다. 커서가 다시 열리면, 커서는 페치할 행의 맨 앞에 다시 위치합니다.

```
 EXEC SQL DECLARE C1 CURSOR FOR
SELECT DEPTNO, DEPTNAME, MGRNO
FROM TDEPT
WHERE ADMRDEPT = 'A00';

 EXEC SQL OPEN C1;

 while (SQLCODE==0) {                                          .
```

```
        EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
      .
      .
}
   EXEC SQL CLOSE C1;
```

COMMENT

COMMENT문은 다양한 오브젝트의 카탈로그 설명에 주석을 추가하거나 대체합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

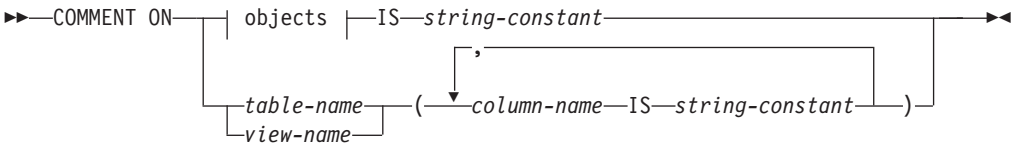
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 권한 중 하나를 가지고 있어야 합니다.

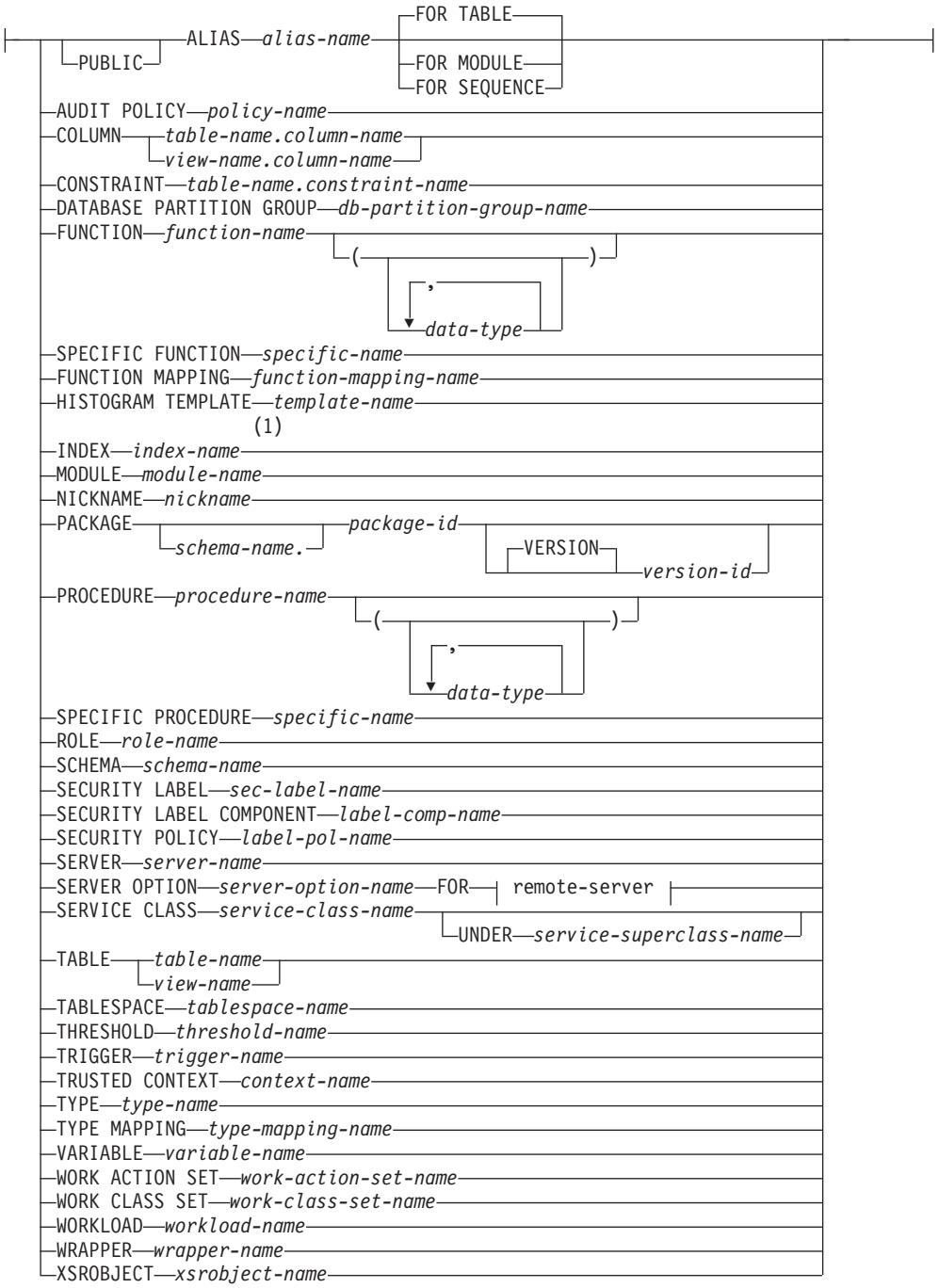
- 오브젝트에 대한 카탈로그 뷰의 소유자 컬럼에 기록된 오브젝트의 소유자(컬럼 또는 제한조건에 대한 하부 테이블)
- 스키마에 대한 ALTERIN 특권(두 개 이상의 부분으로 구성되는 이름을 허용하는 오브젝트에만 적용됨)
- 오브젝트에 대한 CONTROL 특권(인덱스, 패키지, 테이블 또는 뷰 오브젝트에만 적용 가능)
- 오브젝트에 대한 ALTER 특권(테이블 오브젝트에만 적용 가능)
- WITH ADMIN OPTION(역할에만 적용 가능)
- WLMADM 권한(워크로드 관리 프로그램 오브젝트에만 적용 가능)
- SECADM 권한(감사 규정, 역할, 보안 레이블, 보안 레이블 구성요소, 보안 규정 또는 트러스트된 컨텍스트 오브젝트에만 적용 가능)
- DBADM 권한(감사 규정, 역할, 보안 레이블, 보안 레이블 구성요소, 보안 규정 또는 트러스트된 컨텍스트 오브젝트를 제외한 모든 오브젝트에 적용 가능)

테이블 스페이스나 데이터베이스 파티션 그룹 및 버퍼 풀의 경우, 권한 부여 ID는 SYSADM 또는 SYSCTRL 권한이 있어야 함에 유의하십시오.

구문

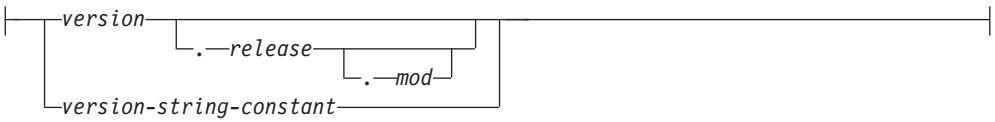


오브젝트:



remote-server:

server-version:



주:

1 *Index-name*은 인덱스 또는 인덱스 스펙의 이름일 수 있습니다.

설명

ALIAS *alias-name*

별명에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *alias-name*은 현재 서버에 존재하는 별명을 식별해야 합니다(SQLSTATE 42704).

FOR TABLE, FOR MODULE 또는 FOR SEQUENCE

별명에 대한 오브젝트 유형을 지정합니다.

FOR TABLE

테이블, 뷰 또는 별칭에 대한 별명입니다. 주석은별명을 기술하는 행에 대한 SYSCAT.TABLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

FOR MODULE

모듈에 대한 별명입니다. 주석은 별명을 설명하는 행에 대한 SYSCAT.MODULES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

FOR SEQUENCE

시퀀스에 대한 별명입니다. 주석은 별명을 설명하는 행에 대한 SYSCAT.SEQUENCES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

PUBLIC이 지정되면 *alias-name*은 현재 서버에 존재하는 공용 별명을 식별해야 합니다(SQLSTATE 42704).

AUDIT POLICY *policy-name*

감사 규정에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *policy-name*은 현재 서버에 존재하는 별명을 식별해야 합니다(SQLSTATE 42704). 주석은 감사 규정을 설명하는 행에 대한 SYSCAT.AUDITPOLICIES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

COLUMN *table-name.column-name* **또는** *view-name.column-name*

컬럼에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *table-name.column-name* 또는 *view-name.column-name* 조합은 현재 서버에 존재하는 컬럼과 테이블 조합을 식별해야 하지만(SQLSTATE 42704), 전역 임시 테이블은 식별하지 않아야 합니다(SQLSTATE 42995). 주석은 컬럼을 설명하는 행에 대한 SYSCAT.COLUMNS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

CONSTRAINT *table-name.constraint-name*

제한조건에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *table-name.constraint-name* 조합은 제한조건과 그 제한조건이 제한하는 테이블을 식별해야 하며, 현재 서버에 존재해야 합니다(SQLSTATE 42704). 주석은 제한조건을 기술하는 행에 대한 SYSCAT.TABCONST 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

DATABASE PARTITION GROUP *db-partition-group-name*

데이터베이스 파티션 그룹에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *db-partition-group-name*은 현재 서버에 존재하는 구별 데이터베이스 파티션 그룹을 식별해야 합니다(SQLSTATE 42704). 주석은 데이터베이스 파티션 그룹을 기술하는 행에 대한 SYSCAT.DBPARTITIONGROUPS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

FUNCTION

함수에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. 지정된 함수 인스턴스는 현재 서버에 있는 사용자 정의 함수(UDF) 또는 함수 템플리트여야 합니다. 사용자 정의 함수(UDF)는 모듈 함수를 식별하면 안됩니다(SQLSTATE 42883).

다음과 같은 여러 가지 방법으로 함수 인스턴스를 식별할 수 있습니다.

FUNCTION *function-name*

특정 함수를 식별하며, *function-name*을 갖는 함수가 정확히 하나가 있는 경우에만 유효합니다. 그러므로 식별되는 함수는 이에 대해 정의된 임의 수의 매개변수를 가질 수 있습니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. 이름이 지정된 스키마 또는 내재적 스키마에 이 이름을 가진 함수가 없으면, 오류가 발생합니다(SQLSTATE 42704). 이름이 지정된 스키마나 내재적 스키마에 함수의 특정 인스턴스가 둘 이상 있는 경우, 오류가 발생합니다(SQLSTATE 42725).

FUNCTION *function-name* **(***data-type***,...)**

주석을 작성할 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 선택 알고리즘은 사용되지 *않습니다*.

function-name

주석을 작성할 함수의 이름을 제공합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

(*data-type***,...)**

해당 위치에 있는 CREATE FUNCTION문에 지정된 데이터 유형과 일치

해야 합니다. 데이터 유형 수 및 데이터 유형의 논리적 병합은 주석을 추가하거나 대체할 특정 함수를 식별하는 데 사용됩니다.

*data-type*이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해서는 정밀도나 스케일 및 길이를 지정하지 않아도 됩니다. 대신 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다.

0<n<25는 REAL을 의미하고 24<n<54는 DOUBLE을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

FOR BIT DATA 속성은 일치를 위해 시그니처의 일부분으로 고려되지 않습니다. 그러므로 예를 들어, 시그니처에 지정된 CHAR FOR BIT DATA는 CHAR를 사용하여 정의된 함수와만 일치합니다.

지정된 시그니처를 가진 함수가 이름이 지정된 스키마나 내재적 스키마의 유형으로 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC FUNCTION *specific-name*

함수에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. (함수를 식별하는 다른 방법에 대해서는 FUNCTION을 참조하십시오.) 함수 작성시 지정되거나 디폴트값인 특정 이름을 사용하여 주석이 붙여질 특정 사용자 정의 함수를 식별합니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 함수 인스턴스를 식별해야 합니다. 그렇지 않은 경우 오류가 발생합니다 (SQLSTATE 42704).

SYSIBM, SYSFUN 또는 SYSPROC 스키마의 함수에는 주석을 달 수 없습니다 (SQLSTATE 42832).

주석은 함수를 기술하는 행에 대한 SYSCAT.ROUTINES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

FUNCTION MAPPING *function-mapping-name*

함수 맵핑에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *function-*

*mapping-name*은 현재 서버에 존재하는 함수 맵핑을 식별해야 합니다(SQLSTATE 42704). 주석은 함수 맵핑을 기술하는 행에 대한 SYSCAT.FUNCMAPPINGS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

HISTOGRAM TEMPLATE *template-name*
주석이 막대 그래프 템플리트에 추가되거나 대체될 것임을 나타냅니다. *template-name*은 현재 서버에 존재하는 막대 그래프 템플리트를 식별해야 합니다(SQLSTATE 42704). 주석은 막대 그래프 템플리트를 설명하는 행에 대한 SYSCAT.HISTOGRAMTEMPLATES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

INDEX *index-name*
인덱스나 인덱스 스펙에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *index-name*은 현재 서버에 존재하는 인덱스 스펙이나 구별 인덱스를 식별해야 합니다(SQLSTATE 42704). 주석은 인덱스 또는 인덱스 스펙을 기술하는 행에 대한 SYSCAT.INDEXES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

MODULE *module-name*
모듈에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *module-name*은 현재 서버에 존재하는 모듈을 식별해야 합니다(SQLSTATE 42704).

NICKNAME *nickname*
별칭에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *nickname*은 현재 서버에 존재하는 별칭이어야 합니다(SQLSTATE 42704). 주석은 별칭을 기술하는 행에 대한 SYSCAT.TABLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

PACKAGE *schema-name.package-id*
패키지에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. 스키마 이름이 지정되지 않은 경우 디폴트 스키마가 패키지 ID를 내재적으로 규정합니다. 스키마 이름과 패키지 ID는 내재적으로 또는 명시적으로 지정된 버전 ID와 함께 현재 서버에 존재하는 패키지를 식별해야 합니다(SQLSTATE 42704). 주석은 패키지를 기술하는 행에 대한 SYSCAT.PACKAGES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

VERSION *version-id*
주석을 표기할 패키지 버전을 식별합니다. 이 값을 지정하지 않은 경우, 버전은 빈 문자열로 디폴트됩니다. 이름은 같지만 버전이 다른 여러 개의 패키지가 있을 경우에는 COMMENT문을 한 번 호출할 때마다 하나의 패키지 버전에만 주석을 표기할 수 있습니다. 큰따옴표를 사용하여 버전 ID를 구분하는 경우는 다음과 같습니다.

- VERSION(AUTO) 프리컴파일러 옵션을 사용하여 생성한 경우
- 숫자로 시작되는 경우
- 소문자 또는 대소문자 혼용 문자가 있는 경우

운영 체제 명령 프롬프트에서 명령문을 호출한 경우, 각 큰따옴표 분리문자 앞에 백슬래시 문자를 사용하여 운영 체제가 분리문자를 없애지 못하게 하십시오.

PROCEDURE

프로시저에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. 지정된 프로시저 인스턴스는 현재 서버에 있는 프로시저여야 합니다. 프로시저는 모듈 프로시저를 식별하면 안됩니다(SQLSTATE 42883).

프로시저 인스턴스를 식별하는 데 사용할 수 있는 방법이 여러 가지 있습니다.

PROCEDURE *procedure-name*

특정 프로시저를 식별하며, 이름이 *procedure-name*인 프로시저가 스키마에 정확히 한 개만 있을 경우에만 유효합니다. 그러므로 식별된 프로시저에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. 이름이 지정된 스키마 또는 내재적 스키마에 이 이름의 프로시저가 존재하지 않는 경우, 오류(SQLSTATE 42704)가 발생합니다. 이름이 지정된 또는 내재적 스키마에 하나 이상의 특정 프로시저 인스턴스가 있는 경우, 오류가 발생합니다(SQLSTATE 42725).

PROCEDURE *procedure-name* **(***data-type***,...)**

주석을 작성할 프로시저를 고유하게 식별하는 함수 시그니처를 제공하는 데 사용됩니다.

procedure-name

주석을 작성할 프로시저의 프로시저 이름을 제공합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

(*data-type***,...)**

CREATE PROCEDURE문의 해당 위치에 지정된 데이터 유형과 일치해야 합니다. 페더레이티드 프로시저의 경우 데이터 유형은 로컬 카탈로그 정보와 일치해야 합니다. 데이터 유형의 수 및 데이터 유형의 논리적 병합은 주석을 추가하거나 대체할 특정 프로시저를 식별하는 데 사용됩니다.

*data-type*이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 정밀도 또는 스케일, 길이를 지정할 필요가 없습니다. 대신 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일이 코딩된 경우 해당 값은 CREATE PROCEDURE문 또는 페더레이티드 프로시저의 경우 로컬 카탈로그 정보에 지정된 값과 정확히 일치해야 합니다.

0<n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처의 프로시저가 이름 지정된 스키마나 내재적 스키마에 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC PROCEDURE *specific-name*

주석이 프로시저에 대해 추가되거나 대체될 것임을 나타냅니다. 프로시저를 식별하는 다른 방법에 대해서는 PROCEDURE를 참조하십시오. 프로시저 작성 시 지정되거나 디폴트값으로 설정되는 특정 이름을 사용하여 주석이 붙여질 특정 프로시저를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 프로시저 인스턴스를 식별해야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42704).

SYSIBM, SYSFUN 또는 SYSPROC 스키마의 프로시저에는 주석을 달 수 없습니다(SQLSTATE 42832).

주석은 프로시저를 기술하는 행에 대한 SYSCAT.ROUTINES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

ROLE *role-name*

역할에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *role-name*은 현재 서버에 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). 주석은 역할을 설명하는 행에 대한 SYSCAT.ROLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SCHEMA *schema-name*

스키마에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *schema-name*은 현재 서버에 존재하는 스키마를 식별해야 합니다(SQLSTATE 42704). 주석은 스키마를 기술하는 행에 대한 SYSCAT.SCHEMATA 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SECURITY LABEL *sec-label-name*

*sec-label-name*이라는 보안 레이블에 대해 주석이 추가되거나 대체될 것임을 표시합니다. 이름은 보안 규정으로 규정해야 하며 현재 서버에 존재하는 보안 레이블을 식별해야 합니다(SQLSTATE 42704). 주석은 보안 레이블을 기술하는 행에 대한 SYSCAT.SECURITYLABELS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SECURITY LABEL COMPONENT *label-comp-name*

*label-comp-name*이라는 보안 레이블 구성요소에 대해 주석이 추가되거나 대체될 것임을 표시합니다. *label-comp-name*은 현재 서버에 존재하는 보안 레이블 구성요소를 식별해야 합니다(SQLSTATE 42704). 주석은 보안 레이블을 기술하는 행에 대한 SYSCAT.SECURITYLABELCOMPONENTS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SECURITY POLICY *label-pol-name*

*label-pol-name*이라는 보안 규정에 대해 주석이 추가되거나 대체될 것임을 표시합니다. *label-pol-name*은 현재 서버에 존재하는 보안 규정을 식별해야 합니다(SQLSTATE 42704). 주석은 보안 규정을 기술하는 행에 대한 SYSCAT.SECURITYPOLICIES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SERVER *server-name*

데이터 소스에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *server-name*은 현재 서버에 존재하는 데이터 소스를 식별해야 합니다(SQLSTATE 42704). 주석은 데이터 소스를 기술하는 행에 대한 SYSCAT.SERVERS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SERVER OPTION *server-option-name* **FOR** *remote-server*

서버 옵션에 대해 주석이 추가되거나 대체될 것임을 나타냅니다.

server-option-name

서버 옵션을 식별합니다. 이 옵션은 현재 서버에 존재하는 옵션이어야 합니다(SQLSTATE 42704). 주석은 서버 옵션을 기술하는 행에 대한 SYSCAT.SERVEROPTIONS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

remote-server

*서버 옵션*이 적용하는 데이터 소스를 기술합니다.

SERVER *server-name*

*server-option*이 적용되는 데이터 소스의 이름을 지정합니다. *server-name*은 현재 서버에 존재하는 데이터 소스를 식별해야 합니다.

TYPE *server-type*

*server-option*이 적용되는 데이터 소스의 유형(예: z/OS 또는 Oracle용 DB2)

을 지정합니다. *server-type*은 대문자나 소문자 어느 것으로든 지정될 수 있습니다. 카탈로그에는 대문자로 저장됩니다.

VERSION

*server-name*으로 식별된 데이터 소스의 버전을 지정합니다.

version

버전 번호를 지정합니다. *version*은 정수여야 합니다.

release

*version*으로 표시되는 버전의 릴리스 번호를 지정합니다. *release*는 정수여야 합니다.

mod

*release*로 표시되는 릴리스의 수정 번호를 지정합니다. *mod*는 정수여야 합니다.

version-string-constant

버전의 전체 명칭을 지정합니다. *version-string-constant*는 단일 값일 수 있습니다(예; '8i'). 또는 *version*, *release* 및 적용 가능한 경우 *mod*의 연속 값일 수 있습니다(예: '8.0.3').

WRAPPER *wrapper-name*

*server-name*으로 참조되는 데이터 소스에 액세스하는 데 사용되는 랩퍼를 식별합니다.

SERVICE CLASS *service-class-name*

주석이 서비스 클래스에 추가되거나 대체될 것임을 나타냅니다. *service-class-name*은 현재 서버에 존재하는 서비스 클래스를 식별해야 합니다(SQLSTATE 42704). 서비스 서브클래스에 대한 주석을 추가하거나 대체하려면 *service-superclass-name*이 UNDER절을 사용하여 지정되어야 합니다. 주석은 서비스 클래스를 설명하는 행에 대한 SYSCAT.SERVICECLASSES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

UNDER *service-superclass-name*

서비스 서브클래스에 대한 주석을 추가하거나 대체할 때 서비스 서브클래스의 서비스 수퍼 클래스를 지정합니다. *service-superclass-name*은 현재 서버에 존재하는 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704).

TABLE *table-name* 또는 *view-name*

테이블 또는 뷰에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *table-name*이나 *view-name*은 현재 서버에 존재하는 테이블이나 뷰(별칭 또는 별명이 아님)를 식별해야 하며(SQLSTATE 42704), 선언된 임시 테이블은 식별해서는 안됩니다(SQLSTATE 42995). 주석은 테이블이나 뷰를 기술하는 행에 대한 SYSCAT.TABLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TABLESPACE *tablespace-name*

테이블 스페이스에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *tablespace-name*은 현재 서버에 존재하는 구별 테이블 스페이스를 식별해야 합니다(SQLSTATE 42704). 주석은 테이블 스페이스를 설명하는 행에 대한 SYSCAT.TABLESPACES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

THRESHOLD *threshold-name*

임계값에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *threshold-name*은 현재 서버에 존재하는 임계값을 식별해야 합니다(SQLSTATE 42704). 주석은 임계값을 설명하는 행에 대한 SYSCAT.THRESHOLDS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TRIGGER *trigger-name*

트리거에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *trigger-name*은 현재 서버에 존재하는 구별 트리거를 식별해야 합니다(SQLSTATE 42704). 주석은트리거를 기술하는 행에 대한 SYSCAT.TRIGGERS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TRUSTED CONTEXT *context-name*

주석이 트러스트된 컨텍스트에 추가되거나 대체될 것임을 나타냅니다. *context-name*은 현재 서버에 존재하는 트러스트된 컨텍스트를 식별해야 합니다(SQLSTATE 42704). 주석은 트러스트된 컨텍스트를 설명하는 행에 대한 SYSCAT.CONTEXTS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TYPE *type-name*

사용자 정의 유형에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *type-name*은 현재 서버에 존재하는 사용자 정의 유형을 식별해야 합니다(SQLSTATE 42704). 주석은 사용자 정의 유형을 기술하는 행에 대한 SYSCAT.DATATYPES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

TYPE MAPPING *type-mapping-name*

사용자 정의 데이터 유형 맵핑에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *type-mapping-name*은 현재 서버에 존재하는 데이터 유형 맵핑을 식별해야 합니다(SQLSTATE 42704). 주석은 맵핑을 기술하는 행에 대한 SYSCAT.TYPEMAPPINGS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

VARIABLE *variable-name*

전역 변수에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *variable-name*은 현재 서버에 존재하는 전역 변수를 식별해야 합니다(SQLSTATE 42704). 주석은 변수를 설명하는 행에 대한 SYSCAT.VARIABLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

WORK ACTION SET *work-action-set-name*

주석이 작업 조치 세트에 대해 추가되거나 대체될 것임을 나타냅니다. *work-action-set-name*은 현재 서버에 존재하는 작업 조치 세트를 식별해야 합니다(SQLSTATE 42704). 주석은 작업 조치 세트를 설명하는 행에 대한 SYSCAT.WORKACTIONSETS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

WORK CLASS SET *work-class-set-name*

주석이 작업 클래스 세트에 대해 추가되거나 대체될 것임을 나타냅니다. *work-class-set-name*은 현재 서버에 존재하는 작업 클래스 세트를 식별해야 합니다(SQLSTATE 42704). 주석은 작업 클래스 세트를 설명하는 행에 대한 SYSCAT.WORKCLASSSETS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

WORKLOAD *workload-name*

워크로드에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). 주석은 워크로드를 설명하는 행에 대한 SYSCAT.WORKLOADS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

WRAPPER *wrapper-name*

랩퍼에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *wrapper-name*은 현재 서버에 존재하는 랩퍼를 식별해야 합니다(SQLSTATE 42704). 주석은 랩퍼를 기술하는 행에 대한 SYSCAT.WRAPPERS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

XSROBJECT *xsrobject-name*

XSR 오브젝트에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *xsrobject-name*은 현재 서버에 존재하는 XSR 오브젝트를 식별해야 합니다(SQLSTATE 42704). 주석은 XSR 오브젝트를 기술하는 행에 대한 SYSCAT.XSROBJECTS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

IS *string-constant*

추가되거나 대체될 주석을 지정합니다. *string-constant*는 최대 254바이트의 문자열 상수일 수 있습니다. 캐리지 리턴(CR)과 줄 바꾸기(LF)는 각각 1바이트로 계산됩니다.

table-name|*view-name* **(** **{** *column-name* **IS** *string-constant* **}** **...** **)**

이 형식의 COMMENT문은 테이블이나 뷰의 여러 컬럼에 대해 주석을 지정할 수 있습니다. 컬럼 이름은 규정되지 않아야 하고, 각 이름은 지정된 테이블이나 뷰의 컬럼을 식별해야 하며, 테이블이나 뷰는 현재 서버에 존재해야 합니다. *table-name*은 임시 테이블로 선언될 수 없습니다(SQLSTATE 42995).

작동 불능 뷰의 컬럼에 대해서는 주석을 작성할 수 없습니다(SQLSTATE 51024).

주

- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.
  - TYPE *type-name* 대신 DISTINCT TYPE *type-name*을 지정할 수 있습니다.
  - TYPE *type-name* 대신 DATA TYPE *type-name*을 지정할 수 있습니다.
  - ALIAS 대신 SYNONYM을 지정할 수 있습니다.

예:

예 1: EMPLOYEE 테이블에 주석을 추가하십시오.

```
COMMENT ON TABLE EMPLOYEE
  IS 'Reflects first quarter reorganization'
```

예 2: EMP_VIEW1 뷰에 주석을 추가하십시오.

```
COMMENT ON TABLE EMP_VIEW1
  IS 'View of the EMPLOYEE table without salary information'
```

예 3: EMPLOYEE 테이블의 EDLEVEL 컬럼에 주석을 추가하십시오.

```
COMMENT ON COLUMN EMPLOYEE.EDLEVEL
  IS 'highest grade level passed in school'
```

예 4: EMPLOYEE 테이블의 서로 다른 두 컬럼에 주석을 추가하십시오.

```
COMMENT ON EMPLOYEE
       (WORKDEPT IS 'see DEPARTMENT table for names',
       EDLEVEL IS 'highest grade level passed in school' )
```

예 5: Pellow는 주석을 작성할 특정 함수를 식별하는 시그니처를 사용하여 자신의 PELLOW 스키마에 작성한 CENTRE 함수에 주석을 추가하십시오.

```
COMMENT ON FUNCTION CENTRE (INT,FLOAT)
  IS 'Frank''s CENTRE fctn, uses Chebychev method'
```

예 6: McBride는 주석을 작성할 함수 인스턴스를 식별하는 특정 이름을 사용하여 자신이 PELLOW 스키마에 작성한 또 다른 CENTRE 함수에 주석을 추가하십시오.

```
COMMENT ON SPECIFIC FUNCTION PELLOW.FOCUS92 IS
  'Louise''s most triumphant CENTRE function, uses the
       Brownian fuzzy-focus technique'
```

예 7: CHEM 스키마에서 함수 AUTOMIC_WEIGHT에 주석을 추가하십시오. CHEM 스키마에는 이 이름을 갖는 함수가 하나뿐입니다.

```
COMMENT ON FUNCTION CHEM.ATOMIC_WEIGHT
  IS 'takes atomic nbr, gives atomic weight'
```

예 8: Eigler는 주석을 작성할 특정 프로시저를 식별하는 시그니처를 사용하여 자신의 EIGLER 스키마에 작성한 SEARCH 프로시저에 주석을 추가하십시오.

```
COMMENT ON PROCEDURE SEARCH (CHAR,INT)
  IS 'Frank''s mass search and replace algorithm'
```

예 9: Macdonald는 주석을 작성할 프로시저 인스턴스를 식별하는 특정 이름을 사용하여 EIGLER 스키마에 작성한 또 다른 SEARCH 함수에 주석을 추가하십시오.

```
COMMENT ON SPECIFIC PROCEDURE EIGLER.DESTROY IS
  'Patrick''s mass search and destroy algorithm'
```

예 10: BIOLOGY 스키마에서 프로시저 OSMOSIS에 주석을 추가하십시오. 이 스키마에는 이 이름을 갖는 프로시저가 하나뿐입니다.

```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS
  IS 'Calculations modelling osmosis'
```

예 11: INDEXSPEC이라는 인덱스 스펙에 주석을 추가하십시오.

```
COMMENT ON INDEX INDEXSPEC
  IS 'An index specification that indicates to the optimizer
       that the table referenced by nickname NICK1 has an index.'
```

예 12: 디폴트 이름이 NET8인 랩퍼에 주석을 추가하십시오.

```
COMMENT ON WRAPPER NET8
  IS 'The wrapper for data sources associated with
      Oracle's Net8 client software.'
```

예 13: XML 스키마 HR.EMPLOYEE에 주석을 작성하십시오.

```
COMMENT ON XSROBJECT HR.EMPLOYEE
  IS '직원 데이터에 대한 기본 XML 스키마입니다.'
```

예 14: 트러스트된 컨텍스트 APPSERVER의 주석을 작성합니다.

```
COMMENT ON TRUSTED CONTEXT APPSERVER
  IS 'WebSphere Server'
```

COMMIT

COMMIT문은 작업 단위(UOW)를 종료하고 해당 작업 단위에서 수행한 데이터베이스 변경사항을 커미트합니다.

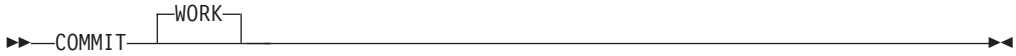
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

COMMIT문이 실행된 작업 단위는 종료되고 새로운 작업 단위가 시작됩니다. 작업 단위(UOW) 중에 실행된 다음 명령문에 의해 수행된 모든 변경사항은 커미트됩니다. ALTER, COMMENT, CREATE, DROP, GRANT, LOCK TABLE, REVOKE, SET INTEGRITY, SET 변수 및 데이터 변경 명령문(INSERT, DELETE, MERGE, UPDATE)(쿼리에 중첩된 명령문 포함)

그러나 다음 명령문은 트랜잭션 제어하에 있지 않으므로, 이러한 명령문에 의한 변경은 COMMIT문과 무관합니다.

- SET CONNECTION
- SET PASSTHRU

  주: SET PASSTHRU문이 트랜잭션의 제어를 받지 않아도 명령문이 트랜잭션 제어하에 있으면 passthru 세션은 초기화됩니다.
- SET SERVER OPTION
- 갱신 가능한 특수 레지스터에 지정

작업 단위를 시작한 이후 작업 단위에서 설정한 모든 잠금은 WITH HOLD로 선언된 열린 커서에 필요한 잠금을 제외하고 모두 해제됩니다. WITH HOLD로 정의되지 않은 모든 열린 커서는 닫힙니다. WITH HOLD로 정의된 열린 커서는 열린 상태로 남아 있고, 그 커서는 결과 테이블의 다음 논리 행 앞에 놓입니다. 위치가 지정된 UPDATE 또는 DELETE문을 발행하기 전에 FETCH를 수행해야 합니다. LOB 로케

이터는 모두 해제됩니다. 이것은 로케이터가 WITH HOLD 등록 정보를 갖는 커서를 통해 검색된 LOB 값과 연관될 때에도 성립합니다.

트랜잭션 내에 설정된 모든 세이브포인트는 해제됩니다.

다음 명령문은 데이터 정의 언어(DDL)와 데이터 제어 언어(DCL) 명령문과 다르게 동작합니다. 명령문을 발행하는 현재 연결에 대해서도 명령문이 커미트될 때까지 이 명령문에서 변경한 내용은 영향을 주지 않습니다. 한 번에 이들 명령문의 하나만 응용프로그램에서 발행될 수 있으며 이들 명령문 중 하나만 하나의 작업 단위 내에서 허용됩니다. 각 명령문 다음에는 COMMIT 또는 ROLLBACK 명령문이 와야 이들 명령문 중 다른 명령문을 발행할 수 있습니다.

- CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
- CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
- CREATE WORK ACTION, ALTER WORK ACTION 또는 DROP(WORK ACTION)
- CREATE WORK CLASS, ALTER WORK CLASS 또는 DROP(WORK CLASS)
- CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
- GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)

주

- 종료 전에 각 응용프로그램 프로세스가 명시적으로 작업 단위를 종료하는 것이 매우 바람직합니다. 응용프로그램이 COMMIT나 ROLLBACK문없이 정상적으로 종료되면, 데이터베이스 관리 프로그램은 응용프로그램 환경에 따라 커미트 또는 롤백을 시도합니다.
- 캐시된 동적 SQL문에 대해 COMMIT가 미치는 영향에 대한 자세한 내용은 『EXECUTE』를 참조하십시오.
- 임시 테이블에서 작성된 COMMIT의 잠재적 영향에 대한 자세한 내용은 『CREATE GLOBAL TEMPORARY TABLE』을 참조하십시오.
- 선언된 임시 테이블에 대해 COMMIT가 잠재적으로 미치는 영향에 대한 자세한 내용은 『DECLARE GLOBAL TEMPORARY TABLE』을 참조하십시오.

예 :

최종 커미트 이후로 수행된 데이터베이스에 대해 변경사항을 커미트합니다.

```
COMMIT WORK
```

복합 SQL(인라인된)

복합 SQL(인라인된) 명령문은 런타임 시 다른 SQL문에서 인라인된 복합 SQL문입니다. 복합 SQL(인라인된) 명령문은 자동으로 실행되는 등록 정보를 갖습니다. 명령문 실행으로 인해 오류가 발생하면, 전체 명령문이 롤백됩니다.

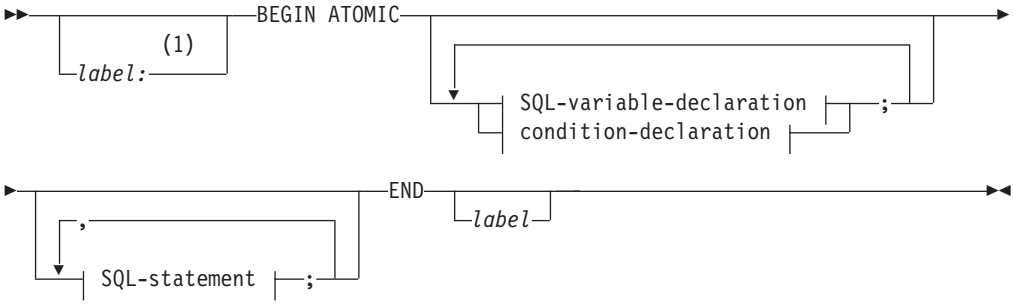
호출

이 명령문은 트리거, SQL 함수 또는 SQL 메소드에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

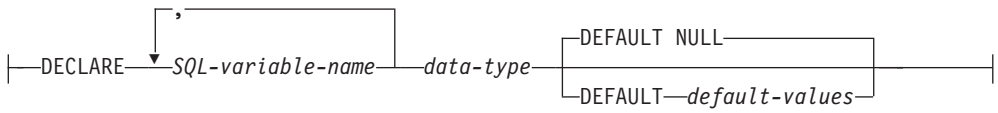
권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권에는 복합 명령문에 지정된 SQL문을 호출하는 데 필요한 모든 특권도 포함해야 합니다.

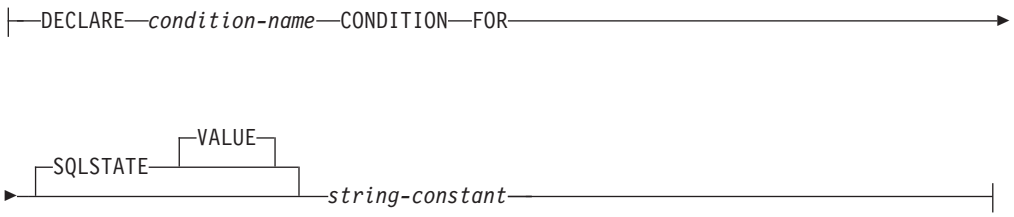
구문



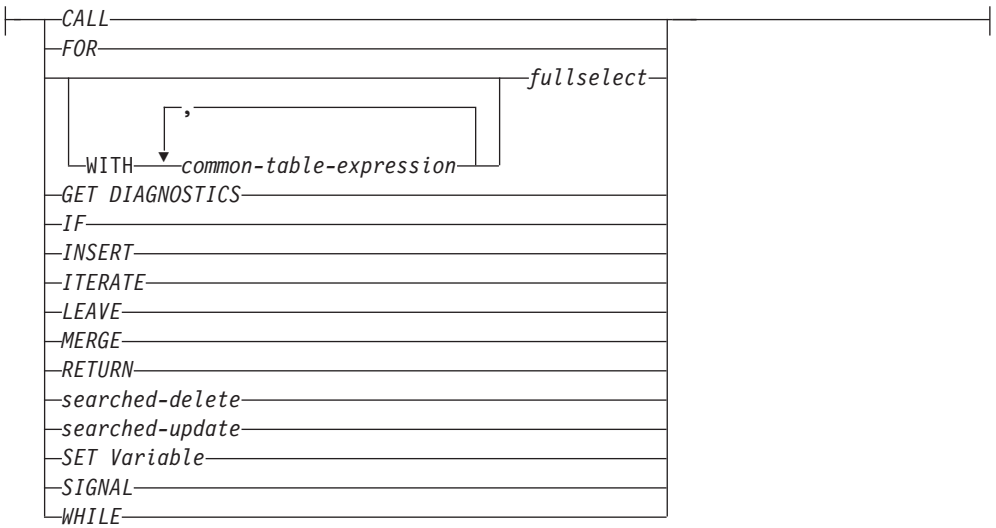
SQL-variable-declaration:



condition-declaration:



SQL-statement:



주:

1 명령문이 함수, 메소드 또는 트리거 정의 내에 있을 때만 레이블을 지정할 수 있습니다.

설명

label

코드 블록에 대한 레이블을 정의합니다. 시작 레이블이 지정된 경우, 복합 SQL(인라인된) 명령문에 선언된 SQL 변수를 규정하는 데 사용할 수 있으며 LEAVE문에도 지정할 수 있습니다. 끝 레이블이 지정된 경우 이 레이블은 시작 레이블과 같아야 합니다.

ATOMIC

ATOMIC은 복합 명령문에서 오류가 발생할 경우, 복합 명령문의 모든 SQL문이 롤백되며 복합 명령문의 나머지 SQL문은 처리되지 않음을 나타냅니다.

ATOMIC 키워드가 모듈이나 SQL 프로시저에서 SQL 함수에 지정된 경우 복합 명령문은 복합 SQL(컴파일된)문으로 처리됩니다.

SQL-statement

복합 SQL(인라인된) 명령문에서 실행될 SQL문을 지정합니다.

SQL-variable-declaration

복합 SQL(인라인된) 명령문에 로컬인 변수를 선언합니다.

SQL-variable-name

로컬 변수의 이름을 정의합니다. DB2 데이터베이스는 모든 SQL 변수 이름을 대문자로 변환합니다. 이 이름은 다음과 같을 수 없습니다.

- 복합 명령문 내의 또 다른 SQL 변수
- 매개변수 이름

SQL문에 SQL 변수 및 컬럼 참조와 동일한 이름의 ID가 포함되는 경우, ID를 컬럼으로 해석합니다.

data-type
변수의 데이터 유형을 지정합니다. XML 데이터 유형은 트리거나 메소드에서 사용되었거나 독립형 명령문으로 사용된 복합 SQL(인라인된) 명령문에서 지원되지 않습니다(SQLSTATE 429BB). XML 데이터 유형은 복합 SQL(인라인된) 명령문이 SQL 함수 본문에서 사용될 때 지원됩니다.

DEFAULT *default-values* **또는 NULL**
SQL 변수의 디폴트값을 정의합니다. 변수는 복합 SQL(인라인된) 명령문이 호출될 때 초기화됩니다. 디폴트값을 지정하지 않을 경우 이 변수는 널(NULL)로 초기화됩니다.

condition-declaration
조건 이름 및 해당되는 SQLSTATE 값을 선언합니다.

condition-name
조건 이름을 지정합니다. 조건 이름은 선언된 복합 명령문 내에서 고유해야 하며 해당 복합 명령문 내에서 중첩된 복합 명령문의 선언을 제외합니다(SQLSTATE 42734). 조건 이름은 선언된 복합 명령문 내에서만 참조될 수 있으며 복합 명령문 내에서 중첩된 복합 명령문을 포함합니다(SQLSTATE 42737).

FOR SQLSTATE *string-constant*
조건과 연관된 SQLSTATE를 지정합니다. 작은따옴표로 묶어 5자로 *string-constant*를 지정해야 하며, 및 SQLSTATE 클래스(처음 2자)는 '00'이 아니어야 합니다.

주

- 복합 SQL(인라인된)문은 한 개의 명령문으로 컴파일됩니다. 이 명령문은 제어 흐름 논리는 거의 포함하지 않지만 중요한 데이터 흐름을 포함하는 짧은 스크립트에 효과적입니다. 중첩된 복합 제어 흐름이나 조건 처리에 대한 요구사항이 있는 대형 구문의 경우, 복합 SQL(컴파일된) 명령문이나 SQL 프로시저를 사용하는 것이 최상의 선택입니다.
- 복합 명령문 내에서 호출된 프로시저는 COMMIT 또는 ROLLBACK문을 발행할 수 없습니다(SQLSTATE 42985).
- ***테이블 액세스 제한사항:*** 프로시저가 READS SQL DATA 또는 MODIFIES SQL DATA로 정의된 경우, 프로시저의 어떤 명령문도 프로시저를 호출한 복합 명령문에서 수정 중인 테이블에 액세스할 수 없습니다(SQLSTATE 57053). 프로시저가 MODIFIES SQL DATA로 정의되어 있으면, 프로시저의 어떤 명령문도 프로시저를 호출한 복합 명령문에서 읽거나 수정 중인 테이블을 수정할 수 없습니다(SQLSTATE 57053).

- ***XML 지정:*** XML 데이터 유형의 매개변수 및 변수에 대한 지정은 SQL 함수 본문에서 참조별 수행됩니다.

  XML 값이 참조에 의해 전달되는 경우, 입력 노드 트리가 직접 사용됩니다. 이러한 직접 사용은 문서 순서, 원래 노드 ID 및 모든 상위 등록 정보를 포함하여 모든 등록 정보를 보존합니다.
- ***분리 레벨:*** *select-statement*, *fullselect* 또는 *subselect*가 *isolation-clause*를 지정한 경우, 절이 무시되고 경고가 리턴됩니다.

예:

예 1:

이 예는 데이터 정리를 위해 데이터 웨어하우징 시나리오에서 인라인 SQL PL을 사용하는 방법을 나타냅니다.

이 예에서는 세 가지 테이블을 사용합니다. ″target″ 테이블은 정리된 데이터를 포함합니다. ″except″ 테이블은 정리될 수 없는 (예외) 행을 저장하고, ″source″ 테이블은 정리될 행 데이터를 포함합니다.

″discretize″라는 단순 SQL 함수가 데이터를 분류하고 수정하는 데 사용됩니다. 잘못된 모든 데이터에 대해 NULL을 리턴합니다. 그런 다음 복합 SQL(인라인된) 명령문이 데이터를 정리합니다. FOR-Loop에 있는 소스 테이블의 모든 행을 조사하고 ″discretize″ 함수의 결과에 따라 현재 행이 ″target″ 또는 ″except″ 테이블에 삽입되는지 여부를 결정합니다. 이 기술을 사용하면 보다 정교한 메커니즘(다단계 정리)이 가능합니다.

SQL 프로시저 또는 호스트 언어로 된 다른 프로시저나 응용프로그램을 사용하여 동일한 코드를 작성할 수 있습니다. 그러나 복합 SQL(인라인된) 명령문을 사용할 때만 얻을 수 있는 이점은 FOR-Loop가 커서를 열지 않으며 단일 행 삽입이 실제로는 단일 행 삽입이 아니라는 점입니다. 사실, 이 논리는 실제적으로 공유 선택을 통한 다중 테이블 삽입입니다.

이를 위해서는 복합 SQL(인라인된) 명령문을 단일 명령문으로 컴파일하면 됩니다. 본문을 사용하는 쿼리에 본문이 통합된 후 쿼리 컨텍스트 내에서 전체적으로 컴파일되고 최적화되는 뷰와 유사하게, DB2 옵티마이저는 제어 및 데이터 흐름 모두를 함께 컴파일하고 최적화합니다. 따라서 전체 논리가 DB2의 런타임 내에서 실행됩니다. 프로시저의 경우와 같이 코어 DB2 엔진 외부로 데이터가 이동되지 않습니다.

첫 단계는 필수 테이블을 작성하는 것입니다.

```
CREATE TABLE target
       (pk INTEGER NOT NULL
       PRIMARY KEY, c1 INTEGER)
```

정리된 데이터를 포함하는 TARGET이라는 테이블을 작성하십시오.

```
CREATE TABLE except
       (pk INTEGER NOT NULL
       PRIMARY KEY, c1 INTEGER)
```

예외를 포함하는 EXCEPT라는 테이블을 작성하십시오.

```
CREATE TABLE source
       (pk INTEGER NOT NULL
       PRIMARY KEY, c1 INTEGER)
```

정리할 데이터를 포함하는 SOURCE라는 테이블을 작성하십시오.

다음으로 [0..1000] 밖에 있는 모든 값을 버리고 10단계로 정렬하여 데이터를 정리하는 "discretize" 함수를 작성하십시오.

```
CREATE FUNCTION discretize(raw INTEGER) RETURNS INTEGER
       RETURN CASE
           WHEN raw < 0 THEN CAST(NULL AS INTEGER)
    WHEN raw > 1000 THEN NULL
           ELSE ((raw / 10) * 10) + 5
       END
```

그런 다음, 값을 삽입하십시오.

```
INSERT INTO source (pk, c1)
       VALUES (1,   -5),
                    (2, NULL),
                    (3, 1200),
                    (4,   23),
                    (5,   10),
                    (6,  876)
```

다음 함수를 호출하십시오.

```
        BEGIN ATOMIC
            FOR row AS
                SELECT pk, c1, discretize(c1) AS d FROM source
            DO
                IF row.d is NULL THEN
                    INSERT INTO except VALUES(row.pk, row.c1);
                ELSE
                    INSERT INTO target VALUES(row.pk, row.d);
                END IF;
            END FOR;
        END
```

결과를 확인하십시오.

```
SELECT * FROM except ORDER BY 1
PK          C1
----------- -----------
          1          -5
          2           -
          3        1200
  3 record(s) selected.
```

```
SELECT * FROM target ORDER BY 1
PK          C1
----------- -----------
          4          25
          5          15
          6         875
  3 record(s) selected.
```

마지막 단계는 정리를 수행하는 것입니다.

```
DROP FUNCTION discretize
DROP TABLE source
DROP TABLE target
DROP TABLE except
```

복합 SQL(임베디드)

하나 이상의 다른 SQL문(*부속 명령문*)을 실행 가능한 블록에 조인합니다.

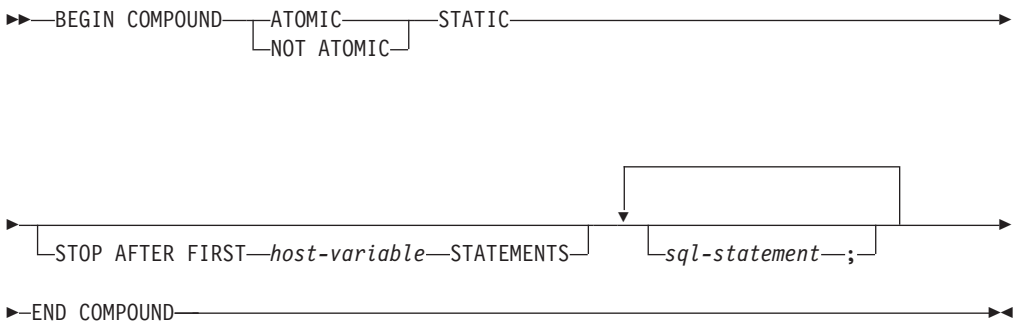
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 전체 복합 SQL(임베디드)문 구문은 동적으로 준비될 수 없는 실행문입니다. 이 명령문은 REXX에서 지원되지 않습니다.

권한 부여

복합 SQL(임베디드)을 호출하는 데 필요한 특권은 없습니다. 단, 복합 명령문에 임베디드(embedded)되는 SQL문을 호출하기 위해 필요한 특권이 명령문의 권한 부여 ID에 의해 보유된 특권에 포함되어야 합니다.

구문



설명

ATOMIC

복합 SQL(임베디드)문에 포함된 부속 명령문 중 하나가 실패하면, 성공적인 부속 명령문으로 수행된 변경사항을 비롯하여 부속 명령문 중 하나로 데이터베이스에 수행된 모든 변경사항을 취소하도록 지정합니다.

NOT ATOMIC

부속 명령문의 실패 여부와 관계없이, 복합 SQL(임베디드)문은 다른 부속 명령문에 의한 데이터베이스에 실행된 변경사항을 취소하지 않도록 지정합니다.

STATIC

모든 부속 명령문에 대한 입력 변수가 원래 값을 갖도록 지정합니다. 예를 들어, 아래와 같은 명령문 다음에

```
SELECT ... INTO :abc ...
```

다음 명령문이 오면,

```
UPDATE T1 SET C1 = 5 WHERE C2 = :abc
```

UPDATE문은 SELECT INTO 다음에 오는 값이 아니라, 복합 SQL(임베디드)문의 실행을 시작할 때 :abc가 가지고 있었던 값을 사용합니다.

두 개 이상의 부속 명령문에 의해 같은 변수가 설정되면, 복합 SQL(임베디드)문 뒤에 오는 해당 변수의 값은 마지막 부속 명령문에 의해 설정된 값입니다.

주: 동적 동작은 지원되지 않습니다. 즉, 부속 명령문은 실행되는 대로 비순차적으로 열람되어야 하고, 상호 종속성을 갖고 있어서는 안됩니다.

STOP AFTER FIRST
특정 개수의 부속 명령문만 실행되도록 지정합니다.

host-variable
실행될 부속 명령문의 수를 지정하는 작은 정수(small integer)입니다.

STATEMENTS
STOP AFTER FIRST *host-variable*절을 완료합니다.

sql-statement
다음을 제외한 모든 실행문이 임베디드 정적 복합 SQL(임베디드)문 내에 포함될 수 있습니다.

```
   CALL                FETCH
   CLOSE               OPEN
   CONNECT             PREPARE
   Compound SQL        RELEASE (Connection)
 DESCRIBE            ROLLBACK
   DISCONNECT          SET CONNECTION
 EXECUTE IMMEDIATE   SET 변수
```

주: Compound SQL에서는 INSERT, UPDATE, DELETE를 별칭과 함께 사용할 수 없습니다.

COMMIT문이 포함된 명령문은 마지막 부속 명령문이어야 합니다. COMMIT가 이 위치에 있으면, STOP AFTER FIRST *host-variable* STATEMENTS절에 모든 부속 명령문이 실행되는 것은 아님을 나타낼 경우에도 이 명령문이 발행됩니다. 예를 들어, COMMIT가 100개의 부속 명령문으로 된 Compound SQL 블록에서 마지막 부속 명령문이라고 할 경우, STOP AFTER FIRST STATEMENTS절에 50개의 부속 명령문만 실행되도록 표시되면 COMMIT는 51번째 부속 명령문입니다.

CONNECT 유형 2를 사용하거나 XA 분산 트랜잭션 프로세싱 환경에서 실행되고 있을 때 COMMIT를 포함시키면 오류가 발생합니다(SQLSTATE 25000).

규칙

- DB2® Connect™는 복합 SQL 블록에서 LOB 컬럼을 선택하는 SELECT문을 지원하지 않습니다.

- 복합 SQL(임베디드)문 내에서는 어떠한 호스트 언어 코드도 사용할 수 없습니다. 즉, 복합 SQL(임베디드)문을 구성하는 부속 명령문 간에 호스트 언어 코드를 사용할 수 없습니다.
- NOT ATOMIC 복합 SQL(임베디드)문만 DB2 Connect에 의해 승인될 수 있습니다.
- 복합 SQL(임베디드)문은 중첩될 수 없습니다.
- 준비된 COMMIT문은 ATOMIC 복합 SQL(임베디드)문에서 사용할 수 없습니다.

주

전체 복합 SQL(임베디드)문에 대해 하나의 SQLCA가 리턴됩니다. 해당 SQLCA에 있는 대부분의 정보는 마지막 부속 명령문을 처리할 때 응용프로그램 서버(AS)가 설정한 값을 반영합니다. 예를 들면 다음과 같습니다.

- SQLCODE 및 SQLSTATE는 보통 마지막 부속 명령문에 대한 것입니다(예외에 대해서는 다음 항목에 설명되어 있음).
- '데이터 없음' 경고(SQLSTATE 02000)가 발생할 경우, WHENEVER NOT FOUND 예외가 발생할 수 있으므로 이 경고는 다른 경고보다 우선합니다. 즉, 결국 응용프로그램에 리턴되는 SQLCODE, SQLERRML, SQLERRMC, SQLERRP 필드는 '데이터 없음' 경고를 트리거한 부속 명령문의 SQLCODE, SQLERRML, SQLERRMC, SQLERRP입니다. 복합 SQL(임베디드)문 내에 '데이터 없음' 경고가 여러 개 있으면, 마지막 부속 명령문에 대한 필드가 리턴되는 필드가 됩니다.
- SQLWARN 표시기는 모든 부속 명령문에 대한 표시기 세트의 누적입니다.

NOT ATOMIC 복합 SQL 실행 중 하나 이상의 오류가 발생했고 이들 중 어떤 것도 심각한 오류가 아니면, SQLERRMC에는 이 오류에 대한 정보가 최대 7개까지 포함됩니다. SQLERRMC의 첫 번째 토큰은 발생한 총 오류 수를 나타냅니다. 나머지 토큰은 순서 위치와 복합 SQL(임베디드)문 내의 실패한 부속 명령문의 SQLSTATE를 각각 포함합니다. 형식은 다음과 같은 문자열입니다.

```
nnnXsssccccc
```

여기서, 7번까지 반복하는 X로 시작하는 부속 문자열이 있는데 문자열 요소는 다음과 같이 정의됩니다.

nnn 오류가 발생한 총 명령문 수입니다. 번호가 999를 초과할 경우, 0부터 다시 계산됩니다. 이 필드는 왼쪽으로 정렬되고 공백으로 채워집니다.

X 토큰 구분자 X'FF'입니다.

sss 오류가 발생한 명령문의 원래 위치입니다. 번호가 999를 초과할 경우, 0부터 다시 계산됩니다. 예를 들어, 첫 번째 명령문이 실패하면, 이 필드에는 왼쪽으로 정렬된 1('1 ')이 포함됩니다.

ccccc 오류의 SQLSTATE입니다.

두 번째 SQLERRD 필드에는 실패한 명령문 수가 포함됩니다(음수의 SQLCODE가 리턴됨).

SQLCA의 세 번째 SQLERRD 필드는 모든 부속 명령문의 영향을 받는 행 수의 누적입니다.

SQLCA의 네 번째 SQLERRD 필드는 성공한 부속 명령문 수입니다. 예를 들면, 복합 SQL(임베디드)문의 세 번째 부속 명령문이 실패하면, 네 번째 SQLERRD 필드는 2로 설정되며, 이는 두 개의 부속 명령문이 오류를 발견하기 전에 성공적으로 처리되었음을 표시합니다.

SQLCA의 다섯 번째 SQLERRD 필드는 제한조건 활동을 트리거한 모든 부속 명령문에 대한 참조 무결성 제한조건의 적용으로 갱신되거나 삭제된 행 수의 누적 값입니다.

예:

예 1: C 프로그램에서 ACCOUNTS 테이블과 TELLERS 테이블을 모두 갱신하는 복합 SQL(임베디드)문을 발행하십시오. 명령문 중에 오류가 있으면, 모든 명령문의 결과를 실행 취소합니다(ATOMIC). 오류가 없으면, 현재 작업 단위를 커미트합니다.

```
 EXEC SQL BEGIN COMPOUND ATOMIC STATIC
UPDATE ACCOUNTS SET ABALANCE = ABALANCE + :delta
  WHERE AID = :aid;
UPDATE TELLERS SET TBALANCE = TBALANCE + :delta
  WHERE TID = :tid;
INSERT INTO TELLERS (TID, BID, TBALANCE) VALUES (:i, :branch_id, 0);
COMMIT;
END COMPOUND;
```

예 2: C 프로그램에서 10개의 데이터 행을 데이터베이스에 삽입하십시오. 호스트 변수 :nbr에 값 10이 있고 S1이 준비된 INSERT문이라고 가정하십시오. 또한 오류에 관계없이 모든 삽입이 시도된다고 가정합니다(NOT ATOMIC).

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC STOP AFTER FIRST  :nbr STATEMENTS
     EXECUTE S1 USING DESCRIPTOR :*sqlda0;
     EXECUTE S1 USING DESCRIPTOR :*sqlda1;
     EXECUTE S1 USING DESCRIPTOR :*sqlda2;
     EXECUTE S1 USING DESCRIPTOR :*sqlda3;
     EXECUTE S1 USING DESCRIPTOR :*sqlda4;
     EXECUTE S1 USING DESCRIPTOR :*sqlda5;
     EXECUTE S1 USING DESCRIPTOR :*sqlda6;
     EXECUTE S1 USING DESCRIPTOR :*sqlda7;
     EXECUTE S1 USING DESCRIPTOR :*sqlda8;
     EXECUTE S1 USING DESCRIPTOR :*sqlda9;
   END COMPOUND;
```

복합 SQL(컴파일된)

변수, 조건, 커서 및 핸들러의 로컬 범위로 실행되는 SQL문의 시퀀스.

호출

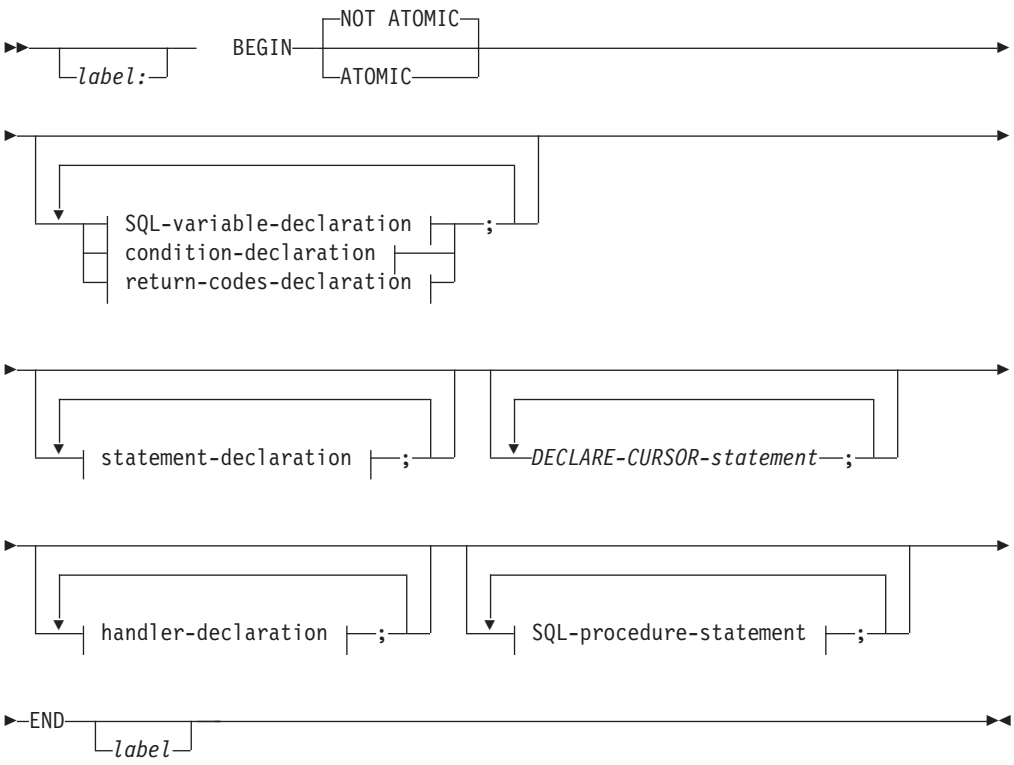
이 명령문은 트리거, SQL 프로시저, SQL 함수에 임베디드(embedded)하거나 독립형 명령문으로 동적으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 문입니다.

권한 부여

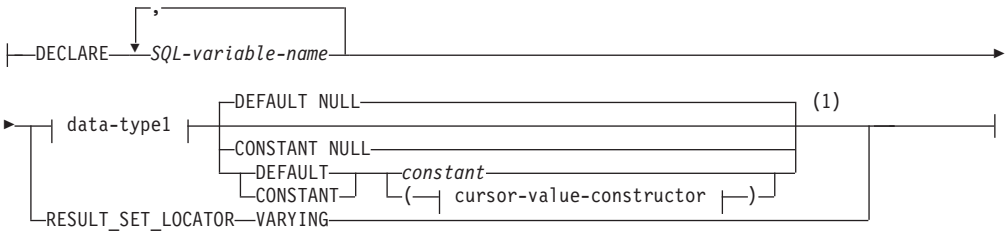
*select-statement*를 사용하는 *cursor-value-constructor*를 지정하는 *SQL-variable-declaration*의 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 *select-statement*를 실행하는 데 필요한 특권을 포함해야 합니다. "SQL 쿼리"의 권한 부여 섹션을 참조하십시오.

명령문의 권한 부여 ID에 의해 보유된 특권에는 복합 명령문에 지정된 SQL문을 호출하는 데 필요한 모든 특권도 포함해야 합니다.

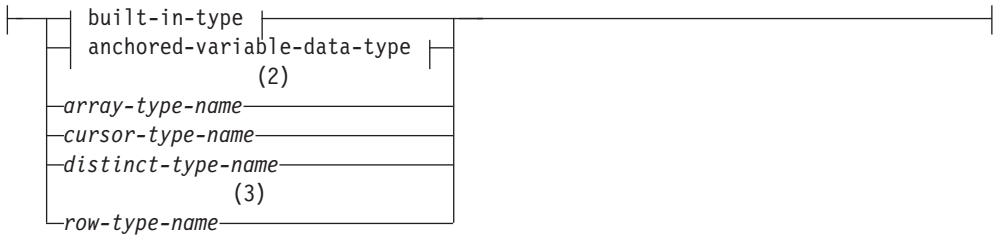
구문



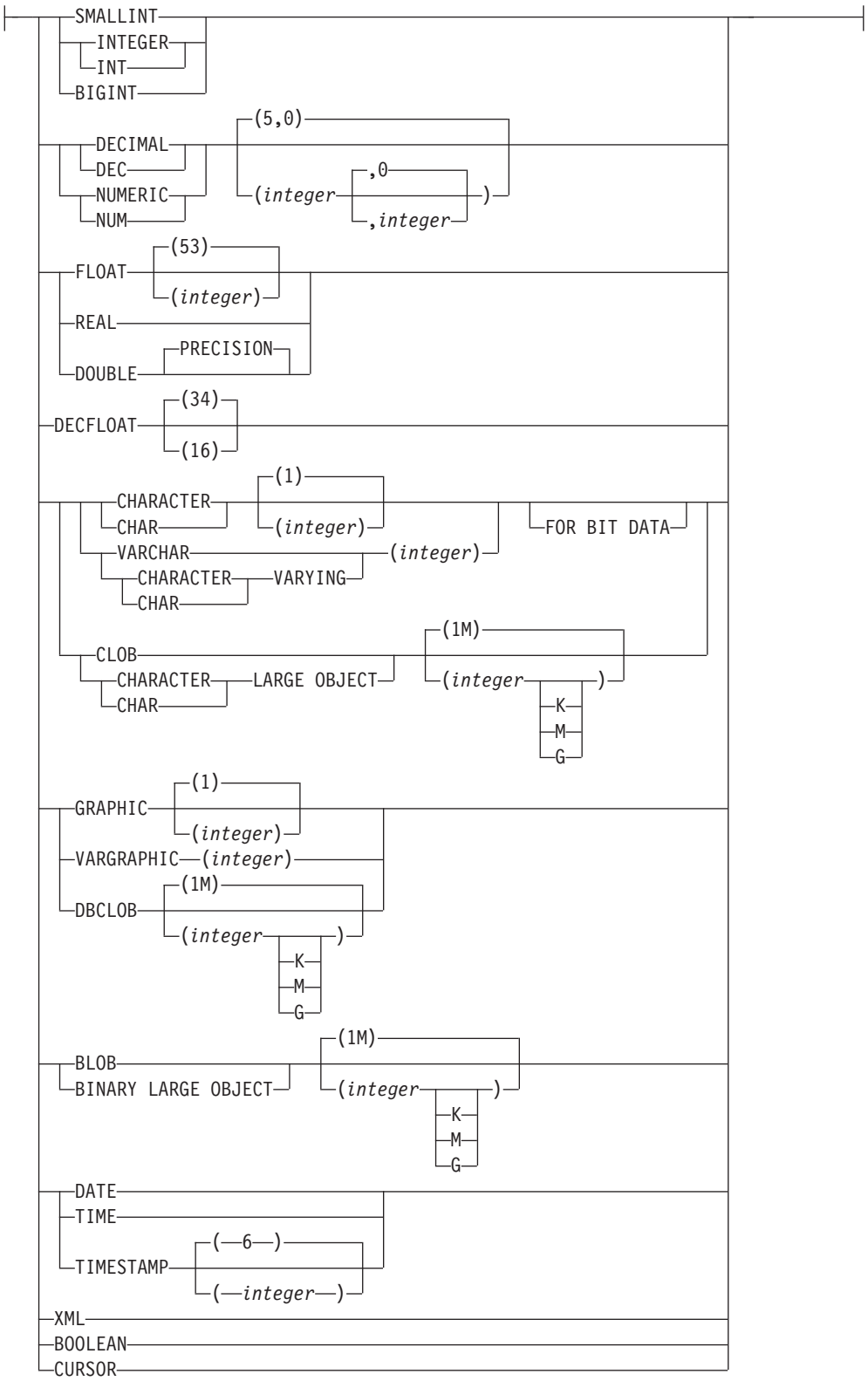
SQL-variable-declaration:



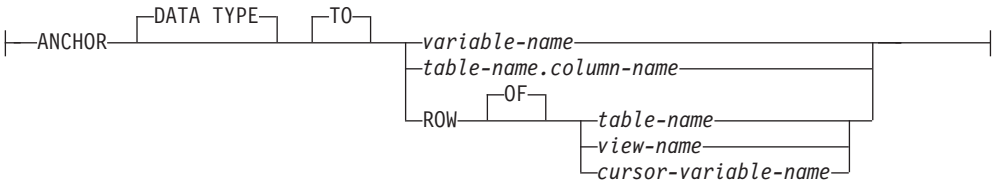
data-type1:



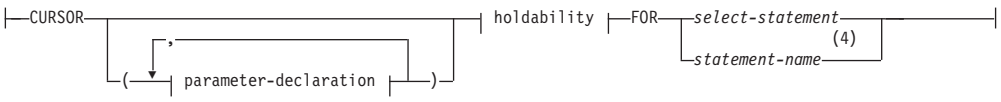
built-in-type:



anchored-variable-data-type:

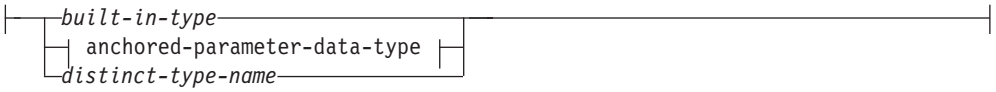


cursor-value-constructor:

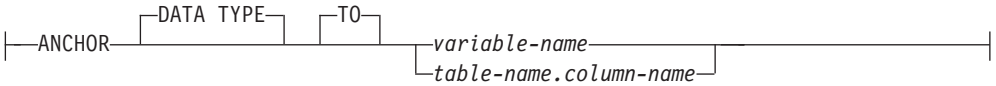


parameter-declaration:

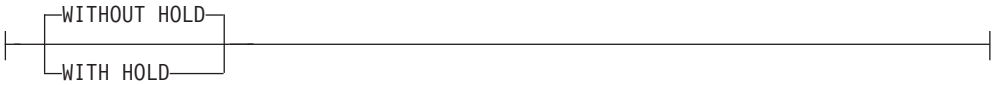
data-type2:



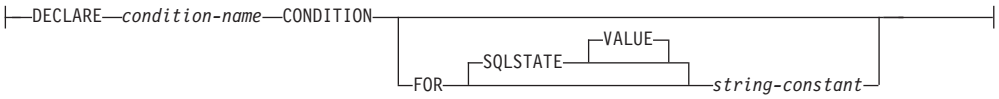
anchored-parameter-data-type:



holdability:

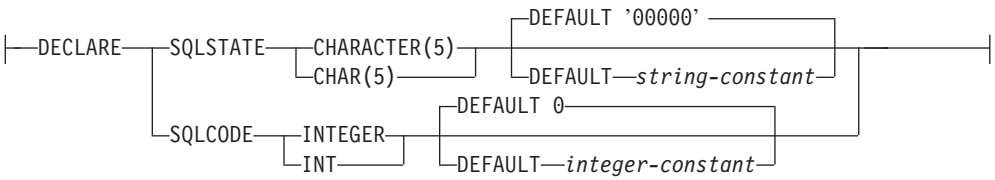


condition-declaration:

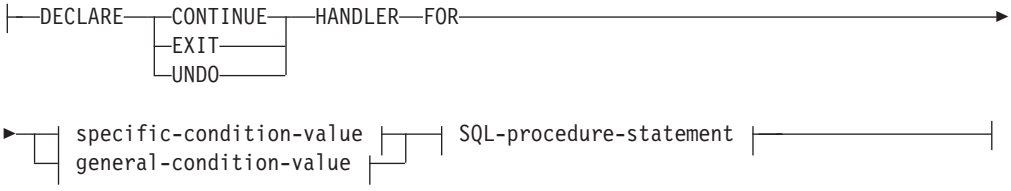


statement-declaration:

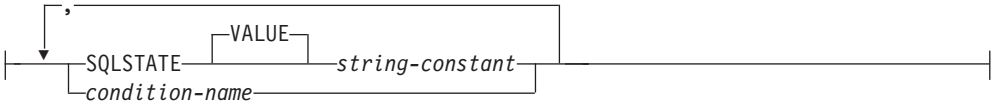
return-codes-declaration:



handler-declaration:



specific-condition-value:



general-condition-value:

SQL-procedure-statement:

주:

1 *data-type1*이 **CURSOR** 내장 유형 또는 *cursor-type-name*을 지정하는 경우, **NULL** 또는 *cursor-value-constructor*만 지정할 수 있습니다. *array-type-name* 또는 *row-type-name*에 대해서는 **DEFAULT NULL**만 명시적으로 지정할 수 있습니다.

2 *array-type-name*에 대해서는 DEFAULT NULL만 명시적으로 지정할 수 있습니다.

3 *row-type-name*에 대해서는 DEFAULT NULL만 명시적으로 지정할 수 있습니다.

4 *parameter-declaration*이 지정된 경우, *statement-name*을 지정할 수 없습니다.

설명

label

코드 블록에 대한 레이블을 정의합니다. 시작 레이블을 지정한 경우, 이 레이블은 복합 명령문에 선언된 SQL 변수를 규정하는 데 사용할 수 있으며, LEAVE문에 지정할 수도 있습니다. 끝 레이블이 지정된 경우 이 레이블은 시작 레이블과 같아야 합니다.

ATOMIC 또는 NOT ATOMIC

ATOMIC은 복합 명령문에서 처리되지 않은 예외 조건이 발생하는 경우, 복합 명령문의 모든 SQL문이 롤백될 것임을 나타냅니다.

NOT ATOMIC은 복합 명령문 내의 처리되지 않은 예외 조건으로 인해 복합 명령문이 롤백되지 않음을 나타냅니다.

ATOMIC 키워드가 모듈내에 없는 SQL 함수에 지정된 경우 복합 명령문은 복합 SQL(인라인된)문으로 처리됩니다.

SQL-variable-declaration

복합 명령문에 대한 로컬 변수를 선언합니다.

SQL-variable-name

로컬 변수의 이름을 정의합니다. 모든 SQL 변수 이름이 대문자로 변환됩니다. 이 이름은 같은 복합 명령문 내의 다른 SQL 변수와 같을 수 없으며 매개변수 이름과 같을 수도 없습니다. SQL 변수 이름은 컬럼 이름과 달라야 합니다. SQL 문에 SQL 변수 및 컬럼 참조와 동일한 이름의 ID가 포함되는 경우, ID를 컬럼으로 분석합니다. 변수가 선언된 복합 명령문에 레이블이 지정되면, 레이블을 사용하여 변수에 대한 참조를 규정할 수 있습니다. 예를 들어, 레이블 C가 있는 복합 명령문에서 선언된 변수 V는 C.V로 참조할 수 있습니다.

data-type1

변수의 데이터 유형을 지정합니다. 구조화된 유형 또는 참조 유형을 지정할 수 없습니다(SQLSTATE 429BB).

built-in-type

내장 데이터 유형을 지정합니다. BOOLEAN 및 CURSOR를 제외한 각 내장 데이터 유형에 대한 자세한 설명은 “CREATE TABLE”을 참조하십시오. 트리거 또는 함수에서 사용되거나 독립형 명령문으로 사용되는 복합 SQL(컴파일된)문에는 XML 데이터 유형을 지정할 수 없습니다(SQLSTATE 429BB). XML 데이터 유형은 복합 SQL(컴파일된)문이 SQL 프로시저 본문에서 사용될 때 지정할 수 있습니다.

BOOLEAN
부울

CURSOR
커서

anchored-variable-data-type
SQL 변수의 데이터 유형을 판별하는 데 사용되는 다른 오브젝트를 식별합니다. 고정 오브젝트의 데이터 유형은 데이터 유형을 직접 지정하기 위해, 또는 행의 경우 행 유형을 작성하기 위해 적용되는 동일한 제한사항을 수반합니다.

ANCHOR DATA TYPE TO
데이터 유형을 지정하는 데 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name
SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다. 참조된 변수의 데이터 유형이 *SQL-variable-name*의 데이터 유형으로 사용됩니다.

table-name.column-name
기존 테이블이나 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형이 *SQL-variable-name*의 데이터 유형으로 사용됩니다.

ROW OF *table-name* **또는** *view-name*
*table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뷰의 컬럼 이름 및 컬럼 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. *SQL-variable-name*의 데이터 유형은 unnamed row 자료형입니다.

ROW OF *cursor-variable-name*
*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 지정된 커서 변수는 다음 중 하나여야 합니다 (SQLSTATE 428HS).

- 강하게 유형이 지정된 커서 데이터 유형의 SQL 변수 또는 전역 변수
- 모든 결과 컬럼의 이름이 지정된 *select-statement*를 지정하는 CONSTANT절로 작성 또는 선언된 약하게 유형이 지정된 커서 데이터 유형이 포함된 SQL 변수 또는 전역 변수.

커서 변수의 커서 유형이 named row 자료형을 사용하여 강하게 유형 지정되지 않은 경우, *SQL-variable-name*의 데이터 유형은 unnamed row 자료형입니다.

array-type-name

사용자 정의 배열 유형의 이름을 지정합니다. 스키마 이름 없이 *array-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 배열 유형이 분석됩니다.

cursor-type-name

커서 유형의 이름을 지정합니다. 스키마 이름 없이 *cursor-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 커서 유형이 분석됩니다.

distinct-type-name

구별 유형의 이름을 지정합니다. 선언된 변수의 길이, 정밀도 및 스케일은 각각 구별 유형의 소스 유형에 대한 길이, 정밀도 및 스케일입니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

row-type-name

사용자 정의 행 유형의 이름을 지정합니다. 변수 필드는 행 유형의 필드입니다. 스키마 이름 없이 *row-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 행 유형이 분석됩니다.

DEFAULT 또는 CONSTANT

복합 SQL(컴파일된)문이 참조될 때 SQL 변수의 값을 지정합니다. 어느 것도 지정하지 않으면, SQL 변수의 디폴트값은 널(NULL) 값입니다. *array-type-name* 또는 *row-type-name*이 지정되는 경우, DEFAULT NULL만 명시적으로 지정할 수 있습니다.

DEFAULT

SQL 변수의 디폴트값을 정의합니다. 변수는 복합 SQL(컴파일된)문이 참조될 때 초기화됩니다. 디폴트값은 변수의 데이터 유형과 호환 가능한 지정이어야 합니다.

CONSTANT

SQL 변수가 변경할 수 없는 고정 값을 가짐을 지정합니다. CONSTANT를 사용하여 정의된 SQL 변수를 지정 조작의 목표로 사용할 수는 없습니다. 고정 값은 변수의 데이터 유형과 호환 가능한 지정이어야 합니다.

NULL

SQL 변수의 디폴트값으로 널(NULL)을 지정합니다.

constant

SQL 변수의 디폴트값으로 상수를 지정합니다. *data-type1*이 CURSOR 내장 유형 또는 *cursor-type-name*을 지정하는 경우, *constant*를 지정할 수 없습니다(SQLSTATE 42601).

cursor-value-constructor

*cursor-value-constructor*는 SQL 변수와 연관된 *select-statement*를 지정합니다. *cursor-value-constructor*를 커서 변수에 지정하면 해당 커서 변수의 기본 커서를 정의합니다.

(parameter-declaration, ...)

각 매개변수의 이름 및 데이터 유형을 포함하는 커서의 입력 매개변수를 지정합니다. *cursor-value-constructor*에 *select-statement*도 지정되면 이름 지정된 입력 매개변수만 지정할 수 있습니다(SQLSTATE 428HU).

parameter-name

*select-statement*내에서 SQL 변수로 사용되는 cursor 매개변수의 이름을 지정합니다. 이름은 커서의 다른 매개변수 이름과 동일할 수 없습니다. 또한 매개변수 이름 전에 컬럼 이름이 분석되므로 *select-statement*에서 사용할 수 있는 컬럼 이름을 피하도록 이름을 선택해야 합니다.

data-type2

*select-statement*내에서 사용되는 cursor 매개변수의 데이터 유형을 지정합니다. 구조화된 유형 및 참조 유형을 지정할 수 없습니다(SQLSTATE 429BB).

built-in-type

내장 데이터 유형을 지정합니다. 각 내장 데이터 유형에 대한 자세한 설명은 CREATE TABLE"을 참조하십시오. BOOLEAN 및 CURSOR 내장 유형을 지정할 수 없습니다(SQLSTATE 429BB).

anchored-parameter-data-type

cursor 매개변수의 데이터 유형을 판별하는 데 사용되는 다른 오브젝트를 식별합니다. 앵커 오브젝트의 데이터 유형에는 데이터 유형을 직접 지정할 때 적용되는 것과 동일한 제한사항이 있습니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하는 데 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

로컬 SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다. 참조된 변수의 데이터 유형은 cursor 매개변수의 데이터 유형으로 사용됩니다.

table-name.column-name
기존 테이블이나 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 cursor 매개변수의 데이터 유형으로 사용됩니다.

distinct-type-name
구별 유형의 이름을 지정합니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

holdability
커서가 커미트 조작의 결과로 닫히지 않도록 할지 여부를 지정합니다. 자세한 정보는 "DECLARE CURSOR"를 참조하십시오. 디폴트값은 WITHOUT HOLD입니다.

WITHOUT HOLD
커서가 커미트 조작의 결과로 닫히도록 합니다.

WITH HOLD
여러 작업 단위에서 자원을 유지보수합니다. 커서가 커미트 조작의 결과로 닫히지 않도록 합니다.

SELECT
커서의 SELECT문을 지정합니다. 자세한 정보는 "select-statement"를 참조하십시오. *parameter-declaration*이 *cursor-value-constructor*에 포함되면, *select-statement*가 로컬 SQL 변수 또는 루틴 SQL 매개변수를 포함해서는 안 됩니다(SQLSTATE 42704).

statement-name
커서의 준비된 *select-statement*을 지정합니다. PREPARE문에 대한 설명은 "PREPARE"를 참조하십시오. 목표 커서 변수에 강하게 유형 지정된 사용자 정의 커서 유형인 데이터 유형이 있어서는 안됩니다(SQLSTATE 428HU). *statement-name*이 지정된 경우, 이름 지정된 입력 매개변수는 *cursor-value-constructor*에 지정되어서는 안 됩니다(SQLSTATE 428HU).

RESULT_SET_LOCATOR VARYING
결과 세트 로케이터 변수의 데이터 유형을 지정합니다.

condition-declaration
선택적 연관된 SQLSTATE 값이 있는 조건 이름을 선언합니다.

condition-name
조건 이름을 지정합니다. 조건 이름은 선언된 복합 명령문 내에서 고유해야 하며 해당 복합 명령문 내에서 중첩된 복합 명령문의 선언을 제외합니다

(SQLSTATE 42734). 조건 이름은 선언된 복합 명령문 내에서만 참조될 수 있으며 복합 명령문 내에서 중첩된 복합 명령문을 포함합니다(SQLSTATE 42737).

CONDITION FOR SQLSTATE VALUE*string-constant*
조건과 연관된 SQLSTATE를 지정합니다. 문자열 상수는 작은따옴표로 묶어 5자로 지정해야 하며, SQLSTATE 클래스(처음 2자)는 '00'이 아니어야 합니다. 이 절을 지정하지 않으면, 조건이 SQLSTATE 값과 연관되지 않습니다.

statement-declaration
복합 명령문에 대해 로컬인 하나 이상의 이름 목록을 선언합니다. *statement-name*의 각 이름은 동일한 복합 명령문에 선언된 다른 명령문 이름과 달라야 합니다.

return-codes-declaration
SQL문 처리 후 리턴되는 값으로 자동 설정되는 특수 변수 SQLSTATE와 SQLCODE를 선언합니다. SQLSTATE 및 SQLCODE 변수는 둘 다 중첩된 복합 SQL(컴파일된) 명령문이 있을 때(예를 들어, SQL 프로시저 본문에서) 가장 외부 복합 명령문에서만 선언될 수 있습니다. 이러한 변수는 SQL 프로시저당 한 번만 선언 가능합니다.

declare-cursor-statement
프로시저 본문에서 시스템 정의 커서를 선언합니다. 사용자 정의 커서 데이터 유형의 변수는 *SQL-variable-declaration*문을 사용하여 선언됩니다.

각 선언된 커서는 선언된 복합 명령문에서 고유한 이름이 있어야 하며, 해당 복합 명령문 안에 중첩되는 복합 명령문에서의 선언은 제외됩니다(SQLSTATE 42734). 커서는 선언된 복합 명령문 내에서만 참조될 수 있으며 복합 명령문 내에서 중첩된 복합 명령문을 포함합니다(SQLSTATE 34000).

OPEN문을 사용하여 커서를 열고 FETCH문을 사용하여 커서로 행을 읽으십시오. 결과 세트를 SQL 프로시저에서 클라이언트 응용프로그램으로 리턴하려면, WITH RETURN절을 사용하여 커서를 선언해야 합니다. 다음 예는 하나의 결과 세트를 클라이언트 응용프로그램으로 리턴합니다.

```
CREATE PROCEDURE RESULT_SET()
  LANGUAGE SQL
  RESULT SETS 1
  BEGIN
    DECLARE C1 CURSOR WITH RETURN FOR
      SELECT id, name, dept, job
        FROM staff;
    OPEN C1;
END
```

주: 결과 세트를 처리하려면 DB2 콜 레벨 인터페이스(CLI)(DB2 콜 레벨 인터페이스(CLI)), ODBC(Open Database Connectivity), JDBC(Java Database Connectivity) 또는 SQLJ(Embedded SQL for Java) API 중 하나를 사용하여 클라이언트 응용프로그램을 작성해야 합니다.

커서 선언에 대한 자세한 정보는 "DECLARE CURSOR"를 참조하십시오.

handler-declaration

핸들러 및 복합 명령문에서 예외 또는 완료 조건이 발생할 때 실행할 하나 이상의 *SQL-procedure-statements* 세트를 지정합니다. *SQL-procedure-statement*는 핸들러가 제어를 수신할 때 실행하는 명령문입니다.

핸들러는 핸들러가 모든 중첩된 복합 명령문을 포함하여 선언되는 복합 명령문 안에서 *handler-declarations* 세트 뒤에 오는 *SQL-procedure-statements* 세트의 실행 지속기간 동안 활성 상태에 있습니다.

조건 핸들러의 유형에는 다음 세 가지가 있습니다.

CONTINUE

핸들러가 정상적으로 호출된 후, 예외를 일으킨 명령문 다음에 오는 SQL문으로 제어가 리턴됩니다. 예외를 일으킨 오류가 FOR, IF, CASE, WHILE 또는 REPEAT문(이들 중 하나에 있는 SQL 프로시저 명령문은 아님)인 경우에는 제어가 END FOR, END IF, END CASE, END WHILE 또는 END REPEAT 다음에 오는 명령문으로 리턴됩니다.

EXIT

핸들러가 정상적으로 호출된 후, 핸들러를 선언한 복합 명령문의 끝으로 제어가 리턴됩니다.

UNDO

핸들러가 호출되기 전에, 복합 명령문에 작성된 모든 SQL 변경사항이 롤백됩니다. 핸들러가 정상적으로 호출된 후, 핸들러를 선언한 복합 명령문의 끝으로 제어가 리턴됩니다. UNDO를 지정한 경우에는 핸들러가 선언되는 복합 명령문이 ATOMIC이어야 합니다.

핸들러 활성화 조건이 다음과 같이 핸들러 선언에 정의됩니다.

specific-condition-value

핸들러가 *특정 조건 핸들러*가 되도록 지정합니다.

SQLSTATE VALUE*string-constant*

핸들러가 호출되는 SQLSTATE를 지정합니다. SQLSTATE 값의 처음 두 자는 '00'이 아니어야 합니다.

condition-name

핸들러가 호출되는 조건 이름을 지정합니다. 조건 이름은 사전에 조건 선언에 정의되거나 현재 서버에 있는 조건을 식별해야 합니다.

general-condition-value

핸들러가 *일반 조건 핸들러*가 되도록 지정합니다.

SQLEXCEPTION

예외 조건 발생 시 핸들러가 호출되도록 지정합니다. 예외 조건은 처음 두 자가 '00', '01' 또는 '02'가 아닌 SQLSTATE 값으로 표시됩니다.

SQLWARNING

경고 조건 발생 시 핸들러가 호출되도록 지정합니다. 경고 조건은 처음 두 자가 '01'인 SQLSTATE 값으로 표시됩니다.

NOT FOUND

NOT FOUND 조건 발생 시 핸들러가 호출되도록 지정합니다. NOT FOUND 조건은 처음 두 자가 '02'인 SQLSTATE 값으로 표시됩니다.

SQL-procedure-statement

SQL 프로시저 명령문을 지정합니다.

label

SQL 프로시저 명령문에 대한 레이블을 지정합니다. 레이블은 SQL 프로시저 명령문 목록 내에서 고유해야 합니다(이 목록에 중첩된 복합 명령문 포함). 중첩되지 않은 복합 명령문은 같은 레이블을 사용할 수도 있음에 유의하십시오. SQL 프로시저 목록은 수많은 SQL 제어 명령문에서 가능합니다.

SQL-statement

다음을 제외한 모든 실행할 수 있는 SQL문:

- ALTER
- CONNECT
- CREATE
- DESCRIBE
- DISCONNECT
- DROP
- FLUSH EVENT MONITOR
- GRANT
- REFRESH TABLE
- RELEASE(연결에서만)
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- SET CONNECTION
- SET INTEGRITY
- SET PASSTHRU
- SET SERVER OPTION
- TRANSFER OWNERSHIP

다음 실행문은 독립형 복합 SQL(컴파일된)문에서 지원되지 않지만, SQL 함수, SQL 프로시저 또는 트리거 내에서 사용되는 복합 SQL(컴파일된)문에서는 지원됩니다.

- 인덱스, 테이블 또는 뷰의 CREATE
- 인덱스, 테이블 또는 뷰의 DROP
- GRANT
- ROLLBACK

ROLLBACK문도 (컴파일된) 독립형 복합 SQL문 내에 호출된 중첩된 명령문에 지원되지 않습니다.

실행문이 아닌 다음 명령문은 복합 SQL(컴파일된)문에서 지원됩니다.

- ALLOCATE CURSOR
- ASSOCIATE LOCATORS

규칙

- ATOMIC 복합 명령문은 중첩될 수 없습니다.
- 핸들러 선언에는 다음과 같은 규칙이 적용됩니다.
  - 핸들러 선언에는 같은 *condition-name* 또는 SQLSTATE 값이 두 번 이상 포함될 수 없으며, 같은 SQLSTATE 값을 나타내는 *condition-name*과 SQLSTATE 값이 포함될 수 없습니다.
  - 다음과 같은 경우에 복합 명령문에서 두 개 이상의 조건 핸들러가 선언됩니다.
    - 두 핸들러 선언이 같은 일반 조건 범주(SQLEXCEPTION, SQLWARNING, NOT FOUND)를 지정하지 않은 경우
    - 두 핸들러 선언이 같은 특정 조건을 SQLSTATE 값이나 같은 값을 나타내는 *condition-name*으로 지정하지 않은 경우
  - 예외나 완료 조건에 가장 적합한 핸들러가 활성화됩니다. 가장 적합한 핸들러는 다음을 고려하여 판별됩니다.
    - 핸들러 선언 *H*의 범위는 *H*가 나타나는 복합 명령문 내에 포함된 핸들러 선언 다음에 나오는 *SQL-procedure-statement* 목록입니다. 즉, *H*의 범위에 조건 핸들러 *H*의 본문에 포함된 명령문이 포함되지 않습니다. 이는 조건 핸들러가 자체 본문 내에 발생하는 조건을 처리할 수 없음을 의미합니다. 마찬가지로 같은 복합 명령문에 선언된 두 핸들러 *H1*과 *H2*의 경우, *H1*은 *H2*의 본문에서 발생하는 조건을 처리하지 않고, *H2*는 *H1*의 본문에서 발생하는 조건을 처리하지 않습니다.
    - 내부 범위에서 선언된 *specific-condition-value* 또는 *general-condition-value* C에 대한 핸들러가 포함 범위에서 선언된 C에 대한 다른 핸들러보다 우선합니다.

- 조건 *C*에 대한 특정 핸들러와 *C*를 처리하는 일반 핸들러를 같은 범위에서 선언하면 특정 핸들러가 일반 핸들러보다 우선합니다.
- SQLSTATE 값과 연관되지 않은 모듈 조건 핸들러 및 SQLSTATE 45000의 핸들러는 동일한 범위로 선언되며 모듈 조건의 핸들러가 SQLSTATE 45000의 핸들러보다 우선합니다.

적합한 핸들러가 없는 예외 조건이 발생하면 오류 명령문이 포함된 SQL 프로시저가 처리되지 않은 예외 조건으로 종료됩니다. 적합한 핸들러가 없는 완료 조건이 발생하면 다음 SQL문으로 실행이 계속됩니다.

- 커미트 또는 롤백 조작이 발생한 후, 새 변수를 SQL 프로시저의 XML 데이터 유형 매개변수 또는 변수에 할당하지 않은 상태에서 SQL 프로시저의 XML 데이터 유형 매개변수 또는 변수를 참조하는 것은 지원되지 않습니다(SQLSTATE 560CE).
- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.

주

- ***XML 지정:*** XML 데이터 유형의 매개변수 및 변수에 지정은 참조에 의해 수행됩니다.

  CALL문의 XML 데이터 유형의 매개변수를 SQL 프로시저로 전달하는 것은 참조에 의해 수행됩니다. XML 값이 참조에 의해 전달되는 경우, 입력 노드 트리가 있으면 XML 인수로부터 직접 사용됩니다. 이러한 직접 사용은 문서 순서, 원래 노드 ID 및 모든 상위 등록 정보를 포함하여 모든 등록 정보를 보존합니다.

예:

다음 조치를 수행하는 복합 SQL(컴파일된) 명령문을 갖는 프로시저를 작성하십시오.

1. SQL 변수를 선언합니다.
2. 커서를 선언하여 IN 매개변수로 결정된 부서의 직원 급여를 리턴합니다. SELECT 문에서 *salary* 컬럼의 데이터 유형을 DECIMAL에서 DOUBLE로 캐스트합니다.
3. '6666' 값을 OUT 매개변수 medianSalary에 지정하는 NOT FOUND(파일의 끝) 조건에 대한 EXIT 핸들러를 선언합니다.
4. SQL 변수 numRecords로 지정 부서의 직원 수를 선택합니다.
5. 직원의 50% + 1이 검색될 때까지 WHILE 루프에서 커서의 행을 페치합니다.
6. 평균 급여를 리턴합니다.

```
CREATE PROCEDURE DEPT_MEDIAN
  (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
  LANGUAGE SQL
```

```
BEGIN
  DECLARE v_numRecords INTEGER DEFAULT 1;
  DECLARE v_counter INTEGER DEFAULT 0;
  DECLARE c1 CURSOR FOR
    SELECT CAST(salary AS DOUBLE) FROM staff
      WHERE DEPT = deptNumber
      ORDER BY salary;
  DECLARE EXIT HANDLER FOR NOT FOUND
    SET medianSalary = 6666;
 -- initialize OUT parameter
    SET medianSalary = 0;
    SELECT COUNT(*) INTO v_numRecords FROM staff
      WHERE DEPT = deptNumber;
    OPEN c1;
    WHILE v_counter < (v_numRecords / 2 + 1) DO
      FETCH c1 INTO medianSalary;
      SET v_counter = v_counter + 1;
    END WHILE;
    CLOSE c1;
END
```

다음 예는 UNDO 핸들러가 RESIGNAL의 결과와 다른 조건에서 활성화되었다고 가정했을 경우의 실행 흐름을 나타냅니다.

```
CREATE PROCEDURE A()
LANGUAGE SQL
CS1: BEGIN ATOMIC
  DECLARE C CONDITION FOR SQLSTATE '12345';
  DECLARE D CONDITION FOR SQLSTATE '23456';

  DECLARE UNDO HANDLER FOR C
  H1: BEGIN
          -- Rollback after error, perform final cleanup, and exit
          -- procedure A.

    -- ...

          -- When this handler completes, execution continues after
          -- compound statement CS1; procedure A will terminate.

  END;

      -- Perform some work here ...
  CS2: BEGIN
          DECLARE CONTINUE HANDLER FOR D
    H2: BEGIN
              -- Perform local recovery, then forward the error
              -- condition to the outer handler for additional
              -- processing.

      -- ...

      RESIGNAL C; -- will activate UNDO handler H1; execution
                                    -- WILL NOT return here.  Any local cursors
                -- declared in H2 and CS2 will be closed.
    END;

          -- Perform some more work here ...

          -- Simulate raising of condition D by some SQL statement
          -- in compound statement CS2:
    SIGNAL D; -- will activate H2
  END;
END
```

CONNECT(유형 1)

CONNECT(유형 1)문은 리모트 작업 단위(UOW)에 대한 규칙에 따라 응용프로그램 프로세스를 식별된 응용프로그램 서버(AS)에 연결합니다.

응용프로그램 프로세스는 한 번에 하나의 응용프로그램에만 연결될 수 있는데, 이를 *현재 서버(current server)*라고 합니다. 디폴트 응용프로그램 서버(AS)는 응용프로그램 리퀘스터가 초기화될 때 설정됩니다. 내재적 연결이 사용 가능하고 응용프로그램 프로세스가 시작된 경우, 디폴트 응용프로그램 서버(AS)에 내재적으로 연결됩니다. 응용프로그램 프로세스는 CONNECT문을 발행하여 다른 응용프로그램 서버(AS)에 명시적으로 연결할 수 있습니다. 연결은 CONNECT RESET문 또는 DISCONNECT문이 발행되거나, 다른 CONNECT문이 응용프로그램 서버(AS)를 변경할 때까지 지속됩니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베디드할 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 명령행 처리기를 사용하여 호출 시, 추가 옵션을 지정할 수 있습니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

권한 부여

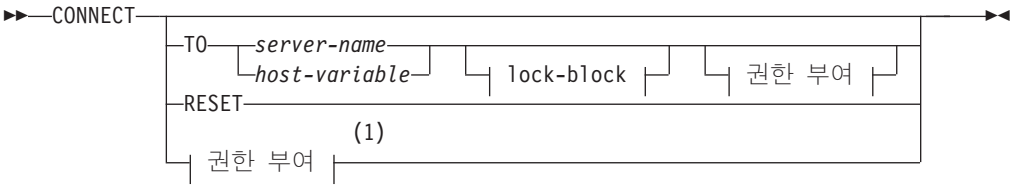
CONNECT 처리는 두 레벨의 액세스 제어를 통해 이루어집니다. 두 레벨은 연결이 성공적이도록 충족되어야 합니다.

첫 번째 레벨의 액세스 제어는 인증이며, 여기서 연결과 연관된 사용자 ID는 서버에 대해 설정된 인증 메소드에 따라 성공적으로 인증되어야 합니다. 인증이 성공하면 서버에 적용되는 인증 플러그인에 따라 DB2 권한 부여 ID가 연결 사용자 ID에서 파생됩니다. 그러면, 이 DB2 권한 부여 ID는 연결에 대한 두 번째 레벨의 액세스 제어, 즉 인증을 전달해야 합니다. 전달하려면, 이 권한 부여 ID에는 적어도 다음 중 하나의 권한이 있어야 합니다.

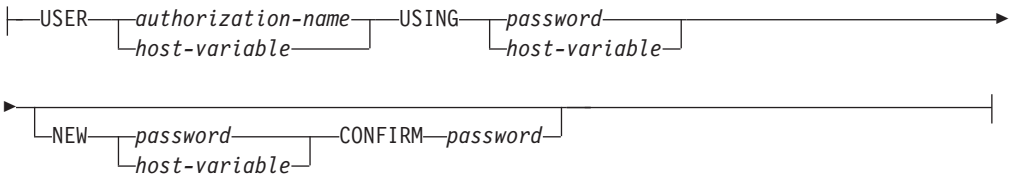
- CONNECT 권한
- SECADM 권한
- DBADM 권한
- SYSADM 권한
- SYSCTRL 권한
- SYSMAINT 권한
- SYSMON 권한

주: 파티션된 데이터베이스의 경우, 사용자와 그룹 정의는 데이터베이스 파티션 전반에 걸쳐 동일해야 합니다.

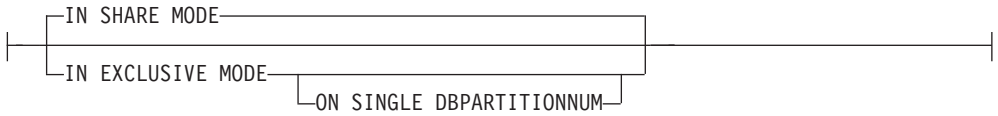
구문



권한 부여:



lock-block:



주:

1 이 형식은 내재적 연결이 설정된 경우에만 유효합니다.

설명

CONNECT(피연산자 없음)

현재 서버에 대한 정보를 리턴합니다. "성공적인 연결"에 기술된 것과 같이 정보는 SQLCA의 SQLERRP 필드에 리턴됩니다.

연결 상태가 존재하면, 권한 부여 ID와 데이터베이스 별명이 SQLCA의 SQLERRMC 필드에 위치합니다. 권한 부여 ID가 8바이트보다 길 경우, ID는 8 바이트로 절단되고,그 절단은 SQLCA의 SQLWARN0 및 SQLWARN1 필드에 각각 'W' 및 'A'를 사용하여 플래그 처리됩니다. 데이터베이스 구성 매개변수 **dyn_query_mgmt**가 사용 가능하면, SQLCA의 SQLWARN0 및 SQLWARN7 필드는 'W' 및 'E'를 각각 사용하여 플래그 처리됩니다.

어떤 연결도 존재하지 않고 내재적 연결이 가능하면, 내재적 연결이 수행됩니다. 내재적 연결이 사용 가능하지 않으면, 오류가 발생합니다(기존 연결 없음). 연 결이 없으면 SQLERRMC 필드는 공백입니다.

응용프로그램 서버(AS)의 지역 코드와 코드 페이지는 SQLERRMC 필드에 위치합니다(성공한 CONNECT문의 경우에도 마찬가지임).

이 형식의 CONNECT가 수행될 때, 다음과 같은 상황이 발생합니다.

- 응용프로그램 프로세스가 연결 가능한 상태일 필요가 없습니다.
- 연결되더라도, 연결 상태가 변경되지 않습니다.
- 연결되지 않고 내재적 연결이 가능한 경우, 디폴트 응용프로그램 서버(AS)에 대한 연결이 만들어집니다. 이 경우, 응용프로그램 서버(AS)의 국가 또는 지역 코드와 코드 페이지는 성공한 CONNECT문과 같이 SQLERRMC 필드에 위치합니다.
- 연결되지 않은 상태이면서 내재적 연결이 가능하지 않으면, 응용프로그램 프로세스는 연결되지 않은 상태로 남아 있습니다.
- 커서가 닫히지 않습니다.

TO *server-name* 또는 *host-variable*

서버 이름이 포함된 *host-variable* 또는 지정된 *server-name*으로 응용프로그램 서버(AS)를 식별합니다.

*host-variable*을 지정할 경우 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안됩니다. *host-variable* 내에 들어 있는 *server-name*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안됩니다.

*server-name*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 로컬 디렉토리에 나열되어야 합니다.

CONNECT문이 실행될 때 응용프로그램 프로세스는 연결 가능한 상태여야 합니다.

성공적인 연결:

CONNECT문이 성공할 경우:

- 열려 있던 커서가 모두 닫히고, 준비된 명령문이 모두 삭제되며, 모든 잠금이 이전 응용프로그램 서버로부터 해제됩니다.
- 응용프로그램 프로세스는 이전 응용프로그램 서버(AS)에서 연결이 끊기고(연결되어 있는 경우) 식별된 응용프로그램 서버(AS)에 연결됩니다.
- 응용프로그램 서버(AS)의 실제 이름(별명 아님)은 CURRENT SERVER 특수 레지스터에 있습니다.
- 응용프로그램 서버(AS)에 대한 자세한 내용은 SQLCA의 SQLERRP 필드에 있습니다. 응용프로그램 서버가 IBM 제품이면, 정보는 *pppvvrrm* 형식으로 되어 있습니다.
  - *ppp*는 다음과 같은 제품을 식별합니다.
    - z/OS용 DB2에 대한 DSN
    - VSE & VM용 DB2 서버에 대한 ARI
    - i5/OS용 DB2에 대한 QSQ
    - Linux, UNIX 및 Windows용 DB2 데이터베이스에 대한 SQL
  - *vv*는 '08'과 같은 두 자리의 버전 ID입니다.

– *rr*은 '01'과 같은 두 자리의 릴리스 ID입니다.

– *m*은 '0'과 같은 한 자리의 수정 레벨 ID입니다.

Linux, UNIX 및 Windows용 DB2 Database의 이 릴리스(버전 9.5)는 'SQL09050'으로 식별됩니다.

- SQLCA의 SQLERRMC 필드는 다음 값(X'FF'로 구분됨)을 포함하도록 설정됩니다.
  1. 응용프로그램 서버의 국가 또는 지역 코드(또는 DB2 Connect를 사용할 경우 공백)
  2. 응용프로그램 서버의 코드 페이지(또는 DB2 Connect를 사용할 경우 CCSID)
  3. 권한 부여 ID(처음 8바이트까지만)
  4. 데이터베이스 별명
  5. 응용프로그램 서버의 플랫폼 유형. 현재 식별되는 값은 다음과 같습니다.

     토큰 서버

     QAS System i용 DB2

     QDB2 z/OS용 DB2

     QDB2/6000
     AIX용 DB2 Database

     QDB2/HPUX
     HP-UX용 DB2 Database

     QDB2/LINUX
     Linux용 DB2 Database

     QDB2/NT
     Windows용 DB2 Database

     QDB2/SUN
     Solaris 운영 체제용 DB2 Database

     QSQLDS/VM
     VM용 DB2 서버

     QSQLDS/VSE
     VSE용 DB2 서버
  6. 에이전트 ID. 응용프로그램 대신 데이터베이스 관리 프로그램 내에서 실행되는 에이전트를 식별합니다. 이 필드는 데이터베이스 모니터에 의해 리턴되는 **agent_id** 요소와 같습니다.
  7. 에이전트 인덱스. 에이전트의 인덱스를 식별하고 서비스에 사용됩니다.

8. 데이터베이스 파티션 번호. 파티션되지 않은 데이터베이스의 파티션 번호(있는 경우)는 항상 0입니다.
9. 응용프로그램 클라이언트의 코드 페이지
10. 파티션된 데이터베이스의 데이터베이스 파티션 수. 데이터베이스를 분산할 수 없는 경우, 값은 0입니다. 토큰은 버전 5 이상부터 제공됩니다.

- SQLCA의 SQLERRD(1) 필드는 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환시 예상되는 최대 차이를 혼합 문자 데이터(CHAR 데이터 유형)의 길이로 나타냅니다. 값 0 또는 1은 확장이 없음을 나타내고 1보다 큰 값은 확장이 가능함을 나타내며 음수 값은 축소가 가능함을 나타냅니다.
- SQLCA의 SQLERRD(2) 필드는 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 변환시 예상되는 최대 차이를 혼합 문자 데이터(CHAR 데이터 유형)의 길이로 나타냅니다. 값 0 또는 1은 확장이 없음을 나타내고 1보다 큰 값은 확장이 가능함을 나타내며 음수 값은 축소가 가능함을 나타냅니다.
- SQLCA의 SQLERRD(3) 필드는 연결시 데이터베이스를 갱신할 수 있는지 여부를 나타냅니다. 데이터베이스는 처음에는 갱신 가능하지만, 작업 단위(UOW)에서 권한 부여 ID가 갱신을 수행할 수 없다고 판별할 경우 읽기 전용으로 변경됩니다. 값은 다음 중 하나입니다.
  - 1: 갱신 가능
  - 2: 읽기 전용
- SQLCA의 SQLERRD(4) 필드는 특정 연결 등록 정보를 리턴합니다. 값은 다음 중 하나입니다.

  0 N/A(1단계 커미트면서 갱신자인 하위 레벨 클라이언트로부터 실행 중인 경우에만 가능함)

  1 1단계 커미트

  2 1단계 커미트. 읽기 전용(TP 모니터 환경에서의 DRDA1 데이터베이스에 대한 연결의 경우에만 해당됨)

  3 2단계 커미트
- SQLCA의 SQLERRD(5) 필드는 연결의 인증 유형을 리턴합니다. 값은 다음 중 하나입니다.

  0 서버에 대해 인증됨

  1 클라이언트에 대해 인증됨

  2 DB2 Connect를 사용하여 인증됨

  4 암호화를 사용하는 서버에 대해 인증됨

  5 암호화를 사용하는 DB2 Connect를 사용하여 인증됨

  7 외부 Kerberos 보안 메커니즘을 사용하여 인증됨

9 외부 GSS API 플러그인 보안 메커니즘을 사용하여 인증됨

11 암호화된 데이터를 승인하는 서버에 대해 인증됨

255 인증이 지정되지 않음

- SQLCA의 SQLERRD(6) 필드는 데이터베이스가 분산될 때 연결된 데이터베이스 파티션의 데이터베이스 파티션 번호를 리턴합니다. 그렇지 않으면, 0 값이 리턴됩니다.
- 성공적인 연결의 권한 부여 ID가 8바이트보다 클 경우, SQLCA의 SQLWARN1 필드는 'A'로 설정됩니다. 이는 절단이 발생하였음을 나타냅니다. SQLCA의 SQLWARN0 필드는 이러한 경고를 나타내기 위해 'W'로 설정됩니다.
- 데이터베이스에 대한 데이터베이스 구성 매개변수 **dyn_query_mgmt**가 사용 가능할 경우 SQLCA의 SQLWARN7 필드는 'E'로 설정됩니다. SQLCA의 SQLWARN0 필드는 이러한 경고를 나타내기 위해 'W'로 설정됩니다.

실패한 연결:

CONNECT문이 실패할 경우:

- SQLCA의 SQLERRP 필드가 오류를 발견한 응용프로그램 리퀘스터에 있는 모듈의 이름으로 설정됩니다. 모듈 이름의 처음 세 자로 제품을 식별할 수 있습니다.
- 응용프로그램 프로세스가 연결 가능한 상태에 있지 않아서 CONNECT문이 실패할 경우, 응용프로그램 프로세스의 연결 상태는 변경되지 않습니다.
- *서버 이름*이 로컬 디렉토리에 나열되어 있지 않아서 CONNECT문이 실패할 경우, 오류 메시지(SQLSTATE 08001)가 발행되고 응용프로그램 프로세스의 연결 상태는 변경되지 않고 그대로 유지됩니다.
  - 응용프로그램 리퀘스터가 응용프로그램 서버(AS)에 연결되지 않은 경우, 응용프로그램 프로세스는 연결되지 않은 상태로 유지됩니다.
  - 응용프로그램 리퀘스터가 응용프로그램 서버(AS)에 이미 연결되어 있는 경우, 응용프로그램 프로세스는 해당 응용프로그램 서버(AS)에 연결된 상태로 유지됩니다. 임의의 추가 명령문이 응용프로그램 서버(AS)에서 실행됩니다.
- 어떤 이유로 CONNECT문이 실패하면, 응용프로그램 프로세스는 연결되지 않은 상태가 됩니다.

IN SHARE MODE

데이터베이스에 대해 또 다른 동시 연결을 허용하며, 다른 사용자가 독점 모드로 데이터베이스에 연결하지 못하도록 합니다.

IN EXCLUSIVE MODE

배타적 잠금을 보유하는 사용자와 동일한 권한 부여 ID를 갖고 있지 않는 한, 동시 응용프로그램 프로세스가 응용프로그램 서버(AS)에서 조작을 실행하지 못하도록 합니다. DB2 Connect는 이 옵션을 지원하지 않습니다.

ON SINGLE DBPARTITIONNUM

코디네이터 데이터베이스 파티션은 독점 모드로 연결되도록 하고, 다른 모든 데이터베이스 파티션은 공유 모드로 연결되도록 지정합니다. 이 옵션은 파티션된 데이터베이스에서만 유효합니다.

RESET

현재 서버와 응용프로그램 프로세스의 연결을 끊습니다. 커미트 조작이 수행됩니다. 내재적 응용프로그램이 사용 가능하면, 응용프로그램 프로세스는 SQL문이 발행될 때까지 연결되지 않은 상태로 유지됩니다.

USER *authorization-name/host-variable*

응용프로그램 서버(AS)에 연결하려고 하는 사용자 ID를 식별합니다. 호스트 변수가 지정되면, 표시기 변수를 포함하지 않는 문자열 변수여야 합니다. *host-variable* 내에 포함된 사용자 ID는 왼쪽으로 정렬되어야 하며, 따옴표로 구분해서는 안됩니다.

USING *password/host-variable*

응용프로그램 서버(AS)에 연결하려고 하는 사용자 ID의 암호를 식별합니다. *password* 또는 *host-variable*에는 최대 14바이트까지 사용 가능합니다. 호스트 변수를 지정할 경우, 이는 길이 속성이 14바이트 이하인 문자열 변수이어야 하고 표시기 변수를 포함해서는 안 됩니다.

NEW *password/host-variable* **CONFIRM** *password*

USER 옵션으로 식별된 사용자 ID에 지정할 새 암호를 식별합니다. *password* 또는 *host-variable*에는 최대 14바이트까지 사용 가능합니다. 호스트 변수를 지정할 경우, 이는 길이 속성이 14바이트 이하인 문자열 변수이어야 하고 표시기 변수를 포함해서는 안 됩니다. 암호를 변경할 시스템은 사용자 인증이 설정된 방식에 따라 다릅니다. 표시된 릴리스(그리고 추후 릴리스)로 다음 서버에서 이 절을 사용하여 새 암호를 지정할 수 있습니다. AIX 및 Windows 운영 체제의 DB2® Universal Database™ 버전 8, Linux 운영 체제의 DB2 버전 9.1 FixPack 3 이상, z/OS 버전 7용 DB2, i5/OS V6R1용 DB2. Linux에서 DB2 데이터베이스 제품의 암호 변경이 가능하도록 하려면 보안 플러그인 IBMOSchgpwdclient 및 IBMOSchgpwdserver가 사용되도록 DB2 인스턴스를 구성해야 합니다.

주

- 응용프로그램 프로세스에 의해 실행되는 첫 번째 SQL문이 CONNECT문이 되도록 하는 것이 좋습니다.
- CONNECT문이 다른 사용자 ID와 암호를 사용하여 현재 응용프로그램 서버(AS)에 대해 발행되면, 대화가 할당 해제되었다가 다시 할당됩니다. 데이터베이스 관리 프로그램에 의해 모든 커서가 닫힙니다(WITH HOLD 옵션이 사용된 경우 커서 위치가 유실됨).

- CONNECT문이 같은 사용자 ID와 암호를 사용하여 현재 응용프로그램 서버(AS)에 대해 발행되면, 대화가 할당 해제되었다가 다시 할당되지 않습니다. 이 경우, 커서는 닫히지 않습니다.
- 파티션된 다중 파티션 데이터베이스 환경을 사용하려면, 사용자 또는 응용프로그램을 db2nodes.cfg 파일에 나열된 데이터베이스 파티션 중 하나에 연결해야 합니다. 모든 사용자가 코디네이터 파티션과 동일한 데이터베이스 파티션을 사용하는 일이 없도록 해야 합니다.
- *authorization-name* SYSTEM은 CONNECT문에 명시적으로 지정할 수 없습니다. 그러나 Windows 운영 체제에서는 로컬 시스템 어카운트에서 실행되는 로컬 응용프로그램이 SYSTEM이라는 사용자 ID를 데이터베이스에 내재적으로 연결할 수 있습니다.
- Windows 서버에 명시적으로 연결할 경우 *authorization-name* 또는 사용자 *host-variable*은 Microsoft® Windows Security Account Manager(SAM)와 호환 가능한 이름을 사용하여 지정될 수 있습니다. 규정자는 최대 길이가 15바이트인 NetBIOS 양식의 이름이어야 합니다. 예: 'Domain₩User'.
- *호환성*: 이전 버전의 DB2 제품과의 호환성
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.

예:

예 1: C 프로그램에서 데이터베이스 별명 TOROLAB, 사용자 ID FERMAT 및 암호 THEOREM을 사용하여 응용프로그램 서버 TOROLAB에 연결하십시오.

```
EXEC SQL  CONNECT TO TOROLAB USER FERMAT USING THEOREM;
```

예 2: C 프로그램에서 해당되는 데이터베이스 별명이 호스트 변수 APP_SERVER (varchar(8))에 저장되는 응용프로그램 서버(AS)에 연결하십시오. 연결에 성공한 후 응용프로그램 서버(AS)의 3자로 된 제품 ID를 변수 PRODUCT(char(3))에 복사하십시오.

```
EXEC SQL  CONNECT TO :APP_SERVER;
if (strncmp(SQLSTATE,'00000',5))
    strncpy(PRODUCT,sqlca.sqlerrp,3);
```

CONNECT(유형 2)

CONNECT(유형 2)문은 응용프로그램 프로세스를 식별된 응용프로그램 서버(AS)에 연결하고 응용프로그램 지향 분산 작업 단위(DUOW)에 대한 규칙을 설정합니다. 그러면, 이 서버는 이 프로세스에 대한 현재 서버가 됩니다.

CONNECT(유형 1)문의 대부분의 사항이 CONNECT(유형 2)문에도 적용됩니다. 여기서는 그 내용을 반복하지 않고, 유형 2의 요소 중에서 유형 1과 다른 부분에 대해서만 설명합니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베디드할 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 명령행 처리기를 사용하여 호출 시 추가 옵션을 지정할 수 있습니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

권한 부여

CONNECT 처리는 두 레벨의 액세스 제어를 통해 이루어집니다. 두 레벨은 연결이 성공적이도록 충족되어야 합니다.

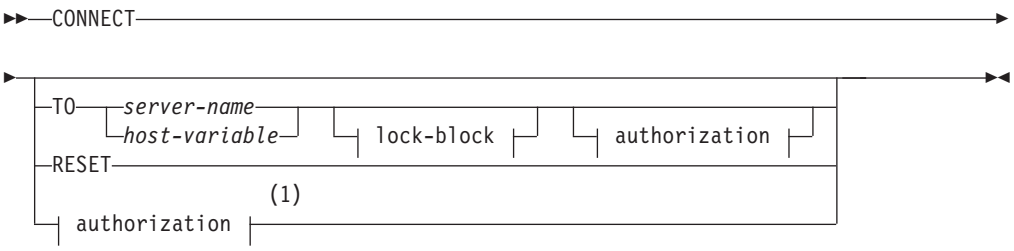
첫 번째 레벨의 액세스 제어는 인증이며, 여기서 연결과 연관된 사용자 ID는 서버에 대해 설정된 인증 메소드에 따라 성공적으로 인증되어야 합니다. 인증이 성공하면 서버에 적용되는 인증 플러그인에 따라 DB2 권한 부여 ID가 연결 사용자 ID에서 파생됩니다. 그러면, DB2 권한 부여 ID는 연결에 대한 두 번째 레벨의 액세스 제어, 즉 인증을 패스해야 합니다. 패스하려면 이 권한 부여 ID에는 다음 중 하나 이상의 권한이 있어야 합니다.

- CONNECT 권한
- SECADM 권한
- DBADM 권한
- SYSADM 권한
- SYSCTRL 권한
- SYSMAINT 권한
- SYSMON 권한

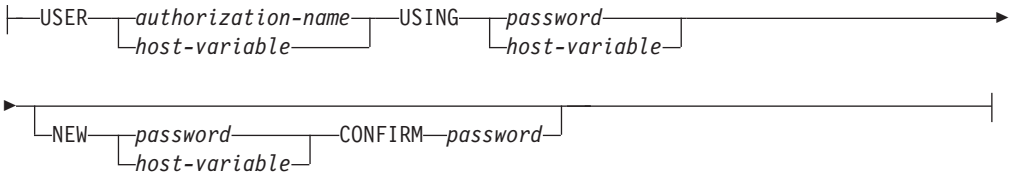
주: 파티션된 데이터베이스의 경우, 사용자와 그룹 정의는 데이터베이스 파티션 전반에 걸쳐 동일해야 합니다.

구문

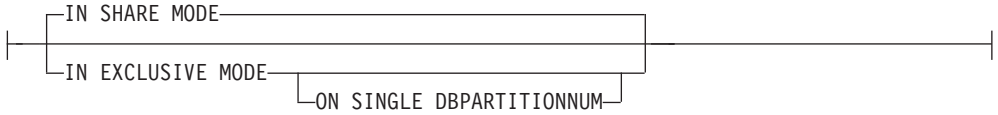
프리컴파일러 옵션에 따라 유형 1과 유형 2 중에서 선택할 수 있습니다. 이 옵션에 대한 개요는 『분산 관계형 데이터베이스 연결』을 참조하십시오.



권한 부여:



lock-block:



주:

1 이 형식은 내재적 연결이 설정된 경우에만 유효합니다.

설명

TO *server-name/host-variable*

서버 이름의 코딩 규칙은 유형 1에서와 같습니다.

SQLRULES(STD) 옵션이 유효할 경우, *server-name*이 응용프로그램 프로세스의 기존 연결을 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 08002).

SQLRULES(DB2) 옵션이 유효하고 *server-name*이 응용프로그램 프로세스의 기존 연결을 식별할 경우, 그 연결이 현재 연결이 되고, 이전 연결은 유휴 상태가 됩니다. 즉, 이 상황에서 CONNECT문을 사용하면 SET CONNECTION문을 사용한 경우와 동일한 결과가 나타납니다.

SQLRULES 스펙에 대한 자세한 내용은 『분산 작업 단위(DUOW) 시맨틱 제어 옵션』을 참조하십시오.

성공적인 연결:

CONNECT문이 성공할 경우:

- 응용프로그램 서버(AS)에 대한 연결이 작성되어(또는 비유휴 상태가 되어) 현재 및 보류 상태가 됩니다.
- CONNECT TO가 현재 서버가 아닌 다른 서버로 지정되면, 현재 연결은 유휴 상태가 됩니다.
- CURRENT SERVER 특수 레지스터와 SQLCA는 CONNECT(유형 1)에서와 같은 방법으로 갱신됩니다.

실패한 연결

CONNECT문이 실패할 경우:

- 실패한 이유가 무엇이든지 응용프로그램 프로세스의 연결 상태 및 연결 상태는 변경되지 않습니다.
- 실패한 CONNECT(유형 1)에서와 마찬가지로, SQLCA의 SQLERRP 필드는 오류를 발견한 응용프로그램 리퀘스터 또는 서버의 모듈 이름으로 설정됩니다.

CONNECT(피연산자 없음), IN SHARE/EXCLUSIVE MODE, USER 및 USING

연결이 존재하면, 유형 2는 유형 1처럼 작동합니다. 권한 부여 ID와 데이터베이스 별명이 SQLCA의 SQLERRMC 필드에 위치합니다. 연결이 존재하지 않을 경우, 내재적 연결에 대한 어떠한 시도도 수행되지 않고 SQLERRP 및 SQLERRMC 필드가 공백을 리턴합니다. 응용프로그램은 이 필드를 점검하여 현재 연결이 존재하는지 점검할 수 있습니다.

USER 및 USING을 포함하는 피연산자가 없는 CONNECT는 DB2DBDFT 환경 변수를 사용하여 데이터베이스에 응용프로그램 프로세스를 계속 연결할 수 있습니다. 이 방법은 유형 2 CONNECT RESET에 해당되는 방법이나, 사용자 ID와 암호를 사용할 수 있습니다.

RESET

디폴트 데이터베이스(사용 가능한 경우)에 대한 내재적 연결과 같습니다. 디폴트 데이터베이스를 사용할 수 없으면, 응용프로그램 프로세스의 연결 상태와 해당되는 연결 상태는 변경되지 않습니다.

디폴트 데이터베이스의 사용 가능성은 설치 옵션, 환경 변수, 인증 설정값에 따라 결정됩니다.

규칙

- 『분산 작업 단위(DUOW) 시맨틱 제어 옵션』에 설명된 대로 일련의 연결 옵션이 연결 관리 시맨틱을 제어합니다. 디폴트값은 처리되는 모든 소스 파일에 지정됩니다. 응용프로그램은 다른 연결 옵션으로 프리컴파일된 여러 개의 소스 파일로 구성될 수 있습니다.

SET CLIENT 명령이나 API를 먼저 실행하지 않으면, 런타임시 실행된 첫 번째 SQL 문을 포함하는 소스 파일을 사전에 처리할 때 사용되는 연결 옵션이 유효한 연결 옵션이 됩니다.

SET CLIENT 명령이나 API를 중간에 실행하지 않고, 다른 연결 옵션으로 사전 처리된 소스 파일의 CONNECT문을 다음으로 실행하면, 오류가 발생합니다(SQLSTATE 08001). SET CLIENT 명령이나 API가 실행된 후에는 응용프로그램의 모든 소스 파일을 사전 처리할 때 사용된 연결 옵션은 무시됩니다.

이러한 규칙은 이 명령문의 『예』 섹션에 설명되어 있습니다.

- CONNECT문을 사용하여 연결을 설정하거나 전환할 수 있더라도, USER/USING절이 있는 CONNECT는 이름이 지정된 서버에 대한 현재 또는 유휴 연결이 없을 때만 사용할 수 있습니다. USER/USING절을 사용하여 같은 서버에 대한 연결을 발행하기 전에 연결을 해제해야 합니다. 그렇지 않으면, 연결이 거부됩니다(SQLSTATE 51022). DISCONNECT문 또는 RELEASE문과 그 다음에 COMMIT문을 발행하여 연결을 해제하십시오.

주

- 내재적 연결은 유형 2의 연결을 사용하는 응용프로그램의 첫 번째 SQL문에 대해 지원됩니다. 디폴트 데이터베이스에서 SQL문을 실행하려면, 먼저 연결을 설정하기 위해 CONNECT RESET 또는 CONNECT USER/USING문을 사용해야 합니다. 피연산자가 없는 CONNECT문은 현재 연결이 있는 경우에는 그 연결에 대한 정보를 표시하고, 현재 연결이 없으면 디폴트 데이터베이스에 연결하지 않습니다.
- *authorization-name* SYSTEM은 CONNECT문에 명시적으로 지정할 수 없습니다. 그러나 Windows 운영 체제에서는 로컬 시스템 어카운트에서 실행되는 로컬 응용프로그램이 SYSTEM이라는 사용자 ID를 데이터베이스에 내재적으로 연결할 수 있습니다.
- Windows 서버에 명시적으로 연결할 경우 *authorization-name* 또는 사용자 *host-variable*은 Microsoft Windows Security Account Manager(SAM)와 호환 가능한 이름을 사용하여 지정될 수 있습니다. 규정자는 최대 길이가 15바이트인 NetBIOS 양식의 이름이어야 합니다. (예: 'Domain₩User')

유형 1과 유형 2의 CONNECT문 비교:

CONNECT문의 시맨틱은 CONNECT 프리컴파일러 옵션이나 SET CLIENT API가 결정합니다(『분산 작업 단위(DUOW) 시맨틱 제어 옵션』 참조). CONNECT 유형 1 또는 CONNECT 유형 2를 지정할 수 있으며, 프로그램의 CONNECT문은 각각 유형 1 및 유형 2 CONNECT문으로 알려집니다. 그 시맨틱은 아래에 설명되어 있습니다.

CONNECT의 사용:

유형 1	유형 2
각 작업 단위는 하나의 응용프로그램 서버에 대한 연결만 설정할 수 있습니다.	각 작업 단위는 복수의 응용프로그램 서버에 대한 연결을 설정할 수 있습니다.
현재 작업 단위는 다른 응용프로그램 서버에 대한 연결을 허용하기 전에 커미트되거나 롤백되어야 합니다.	현재 작업 단위는 다른 응용프로그램 서버에 연결하기 전에 커미트되거나 롤백될 필요가 없습니다.
CONNECT문은 현재 연결을 설정합니다. 후속 SQL 요구는 다른 CONNECT에 의해 변경될 때까지 이 연결에 포워드됩니다.	첫 번째 연결을 설정중이면, 유형 1 CONNECT와 같습니다. 유휴 연결로 전환 중이고 SQLRULES가 STD로 설정되어 있으면, SET CONNECTION문이 대신에 사용되어야 합니다.
현재 연결에 연결하는 것은 유효하며 현재 연결은 변경되지 않습니다.	SQLRULES 프리컴파일러 옵션이 DB2로 설정되면, 유형 1 CONNECT와 동일합니다. SQLRULES가 STD로 설정되면, SET CONNECTION문을 대신 사용해야 합니다.
다른 응용프로그램 서버에 연결하면 현재 연결이 단절됩니다. 새로운 연결이 현재 연결이 됩니다. 하나의 작업 단위(UOW)에서 하나의 연결만 유지보수됩니다.	다른 응용프로그램 서버에 연결하면 현재 연결이 *유휴 상태*가 됩니다. 새로운 연결이 현재 연결이 됩니다. 복수의 연결이 하나의 작업 단위에서 유지보수될 수 있습니다. CONNECT가 유휴 상태의 연결에 있는 응용프로그램 서버(AS)에 대한 것이면, 그것은 현재 연결이 됩니다. CONNECT를 사용하여 유휴 연결에 연결하는 것은 SQLRULES(DB2)가 지정된 경우에만 허용됩니다. SQLRULES(STD)가 지정되면, SET CONNECTION문이 대신 사용되어야 합니다.
SET CONNECTION문은 유형 1 연결을 지원하지만 유일하게 유효한 목표는 현재 연결입니다.	SET CONNECTION문은 연결을 유휴에서 현재 상태로 변경하기 위한 유형 2 연결에 대해 지원됩니다.

CONNECT...USER...USING의 사용:

유형 1	유형 2
USER...USING절을 사용하여 연결하면 현재 연결이 끊어지고, 주어진 권한 부여 이름과 암호를 갖는 새로운 연결을 설정합니다.	USER/USING절은 이름 지정된 동일한 서버에 대해 현재 또는 유휴 연결이 전혀 없을 경우에만 허용됩니다.

내재된 CONNECT, CONNECT RESET 및 연결 끊기 사용:

유형 1	유형 2
CONNECT RESET은 현재 연결을 끊을 때 사용할 수 있습니다.	CONNECT RESET은 디폴트 응용프로그램 서버가 시스템에 정의되어 있을 경우 명시적으로 그 서버에 연결하는 것과 같습니다. 연결은 응용프로그램의 정상적인 COMMIT에 의해 끊어질 수 있습니다. 커미트 전에, RELEASE문을 사용하여 연결을 해제 보류로 표시하십시오. 그러한 모든 연결은 다음 COMMIT에서 끊어집니다. 또 다른 방법은 RELEASE문 대신에 프리컴파일러 옵션인 DISCONNECT(EXPLICIT), DISCONNECT(CONDITIONAL), DISCONNECT(AUTOMATIC) 또는 DISCONNECT문을 사용하는 것입니다.
CONNECT RESET을 사용하여 현재 연결을 끊은 후, 다음 SQL문이 CONNECT문이 아니면, 디폴트 응용프로그램 서버가 시스템에 연결되어 있으면 그 서버에 대한 내재적 연결을 수행합니다.	CONNECT RESET은 디폴트 응용프로그램 서버가 시스템에 정의되어 있는 경우 그 서버에 대한 명시적 연결과 같습니다.
연속 CONNECT RESET을 발행하면 오류가 발생합니다.	SQLRULES(STD)를 지정한 경우에만, 이 옵션은 기존 연결에 대해 CONNECT를 사용하는 것을 허용하지 않으므로 연속 CONNECT RESET을 발행하면 오류가 발생합니다.
CONNECT RESET도 현재 작업 단위를 명시적으로 커미트합니다.	CONNECT RESET은 현재 작업 단위를 커미트하지 않습니다.
기존 연결이 어떠한 이유로든 시스템에서 끊어지면, 그 다음에 오는 데이터베이스에 대한 비CONNECT SQL문은 SQLSTATE 08003을 수신하게 됩니다.	기존 연결이 시스템에 의해 끊어지면, COMMIT, ROLLBACK과 SET CONNECTION문은 계속 사용할 수 있습니다.
작업 단위(UOW)는 응용프로그램 프로세스가 성공적으로 종료될 때 명시적으로 커미트됩니다.	유형 1과 같습니다.
모든 연결(하나만)은 응용프로그램 프로세스가 종료될 때 끊어집니다.	모든 연결(현재, 유휴 및 해제 보류에 대해 표시된 연결)은 응용프로그램 프로세스가 종료되면 끊어집니다.

CONNECT 실패:

유형 1	유형 2
현재 연결의 유무에 관계없이, CONNECT가 실패하면(로컬 디렉토리에 정의되지 않은 서버 이름 이외의 오류로 인해), 응용프로그램 프로세스는 연결되지 않은 상태가 됩니다. 그 다음의 비CONNECT문은 SQLSTATE 08003을 수신합니다.	현재 연결이 있을 때 CONNECT가 실패하면, 현재 연결은 영향을 받지 않습니다. CONNECT 실패시 현재 연결이 전혀 없으면, 프로그램은 연결되지 않은 상태가 됩니다. 그 다음의 비CONNECT문은 SQLSTATE 08003을 수신합니다.

예:

예 1:

이 예는 다중 소스 프로그램(상자안에 표시)의 사용을 보여주는 데, 일부는 다중 연결 옵션으로 사전 처리되었으며(코드위에 표시), 이들 중 하나에는 SET CLIENT API 호출이 들어 있습니다.

PGM1: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
   exec sql CONNECT TO OTTAWA;
   exec sql SELECT col1 INTO :hv1
FROM tbl1;
...
```

PGM2: CONNECT(2) SQLRULES(STD) DISCONNECT(AUTOMATIC)

```
...
   exec sql CONNECT TO QUEBEC;
   exec sql SELECT col1 INTO :hv1
FROM tbl2;
...
```

PGM3: CONNECT(2) SQLRULES(STD) DISCONNECT(EXPLICIT)

```
...
   SET CLIENT CONNECT 2  SQLRULES DB2  DISCONNECT EXPLICIT [1]
   exec sql CONNECT TO LONDON;
   exec sql SELECT col1 INTO :hv1
FROM tbl3;
...
```

참고 1: SET CLIENT API의 실제 구문은 아닙니다.

PGM4: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
   exec sql CONNECT TO REGINA;
   exec sql SELECT col1 INTO :hv1
FROM tbl4;
...
```

응용프로그램이 PGM1을 실행하고 나서 PGM2를 실행할 경우

- OTTAWA에 대한 연결이 실행됩니다. connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- QUEBEC에 대한 연결이 SQLSTATE 08001과 함께 실패합니다. SQLRULES와 DISCONNECT가 서로 다르기 때문입니다.

응용프로그램이 PGM1을 실행하고 나서 PGM3을 실행할 경우

- OTTAWA에 대한 연결이 실행됩니다. connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- LONDON에 대한 연결이 실행됩니다. connect=2, sqlrules=DB2, disconnect=EXPLICIT

이것은 SET CLIENT API가 두 번째 CONNECT문 전에 실행되므로 성립합니다.

응용프로그램이 PGM1을 실행하고 나서 PGM4를 실행할 경우

- OTTAWA에 대한 연결이 실행됩니다. connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- REGINA에 대한 연결이 실행됩니다. connect=2, sqlrules=DB2, disconnect=CONDITIONAL

이것은 PGM1에 대한 프리컴파일러 옵션이 PGM4에 대한 것과 같으므로 성립합니다.

예 2:

이 예는 CONNECT(유형 2), SET CONNECTION, RELEASE, DISCONNECT문의 상호 관계를 나타냅니다. S0, S1, S2, S3는 네 개의 서버를 나타냅니다.

순서	명령문	현재 서버	유휴 연결	해제 보류
0	• 명령문 없음	• 없음	• 없음	• 없음
1	• SELECT * FROM TBLA	• S0(디폴트값)	• 없음	• 없음
2	• CONNECT TO S1	• S1	• S0	• 없음
	• SELECT * FROM TBLB	• S1	• S0	• 없음
3	• CONNECT TO S2	• S2	• S0, S1	• 없음
	• UPDATE TBLC SET ...	• S2	• S0, S1	• 없음
4	• CONNECT TO S3	• S3	• S0, S1, S2	• 없음
	• SELECT * FROM TBLD	• S3	• S0, S1, S2	• 없음
5	• SET CONNECTION S2	• S2	• S0, S1, S3	• 없음
6	• RELEASE S3	• S2	• S0, S1	• S3
7	• COMMIT	• S2	• S0, S1	• 없음
8	• SELECT * FROM TBLE	• S2	• S0, S1	• 없음
9	• DISCONNECT S1	• S2	• S0	• 없음
	• SELECT * FROM TBLF	• S2	• S0	• 없음

CREATE ALIAS

CREATE ALIAS문은 모듈, 별칭, 시퀀스, 테이블, 뷰 또는 다른 별명에 대한 별명을 정의합니다. 별명은 동의어로도 알려졌습니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 별명의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 별명의 스키마 이름이 기존의 스키마를 참조할 경우 스키마에 대한 CREATEIN 특권, 또는 공용 별명이 작성될 경우 SYSPUBLIC의 CREATEIN 특권
- DBADM 권한

별명을 통해 참조되는 오브젝트를 사용하는 데 필요한 특권은 오브젝트를 직접 사용하는 데 필요한 특권과 동일합니다.

기존 별명을 교체하려면 명령문의 권한 부여 ID가 기존 별명의 소유자여야 합니다(SQLSTATE 42501).

구문

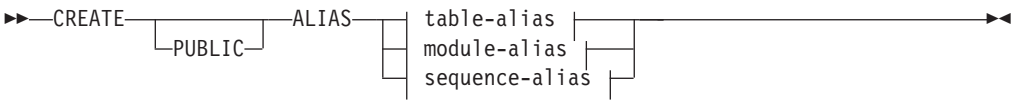
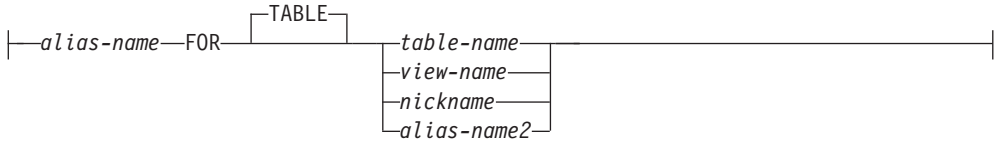
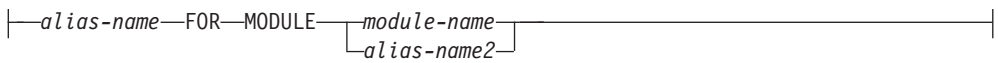


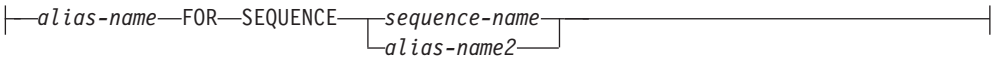
table-alias:



module-alias:



sequence-alias:



설명

PUBLIC

별명이 시스템 스키마 SYSPUBLIC의 오브젝트임을 지정합니다.

alias-name

별명의 이름을 지정합니다. 테이블 별명의 경우, 이름은 현재 서버에 존재하는 별칭, 테이블, 뷰 또는 테이블 별명을 식별해서는 안됩니다. 모듈 별명의 경우, 이름은 현재 서버에 존재하는 모듈 또는 모듈 별명을 식별해서는 안됩니다. 시퀀스 별명의 경우, 이름은 현재 서버에 존재하는 시퀀스 또는 시퀀스 별명을 식별해서는 안됩니다.

두 부분으로 구성된 이름이 지정된 경우, 스키마 이름은 'SYS'로 시작될 수 없으며, PUBLIC이 지정된 경우는 예외로 스키마 이름이 SYSPUBLIC이어야 합니다 (SQLSTATE 428EK).

FOR TABLE *table-name, view-name, nickname,* **또는** *alias-name2*

*alias-name*이 정의되는 테이블, 뷰, 별칭 또는 테이블 별명을 식별합니다. 다른 별명(*alias-name2*)이 제공되면, 정의될(완전한 형식으로) 새 *alias-name*과 같아서는 안됩니다. *table-name*은 임시 테이블로 선언될 수 없습니다(SQLSTATE 42995).

FOR MODULE *module-name*, **or** *alias-name2*

*alias-name*이 정의된 모듈 또는 모듈 별명을 식별합니다. 다른 별명(*alias-name2*)이 제공된 경우, 정의될(완전한 형식으로) 새 alias-name과 같아서는 안됩니다.

FOR SEQUENCE *sequence-name*, **or** *alias-name2*

*alias-name*이 정의된 시퀀스 또는 시퀀스 별명을 식별합니다. 다른 별명(*alias-name2*)이 제공되면, 정의될(완전한 형식으로) 새 *alias-name*과 같아서는 안됩니다. *sequence-name*은 식별 컬럼에 대해 시스템이 생성하는 시퀀스가 아니어야 합니다 (SQLSTATE 428FB).

주

- 키워드 PUBLIC은 공용 별명(공용 동의어라고도 알려진)을 작성하는 데 사용됩니다. 키워드 PUBLIC이 사용되지 않으면 별명 유형은 개인용 별명(개인용 동의어라고도 알려진)입니다.
- 새로 작성된 테이블 별명에 대한 정의는 SYSCAT.TABLES에 저장됩니다. 새로 작성된 모듈 별명에 대한 정의는 SYSCAT.MODULES에 저장됩니다. 새로 작성된 시퀀스 별명에 대한 정의는 SYSCAT.SEQUENCES에 저장됩니다.

- 별명은 정의시 존재하지 않는 오브젝트에 대해 정의될 수 있습니다. 오브젝트가 존재하지 않으면 경고가 발행됩니다(SQLSTATE 01522). 그러나 별명을 포함하는 SQL 문이 컴파일될 때 참조된 오브젝트가 있어야 하며, 그 외에는 오류가 발생합니다 (SQLSTATE 52004).
- 별명 체인의 다른 별명 부분으로 참조하기 위해 별명을 정의할 수 있으나, 이 체인은 SQL문에서 사용될 때 단일 별명과 같은 제한사항을 갖습니다. 별명 체인은 단일 별명과 같은 방법으로 해결됩니다. 패키지의 명령문, SQL 루틴, 트리거, 전역 변수에 대한 기본 표현식 또는 뷰 정의에 사용된 별명이 별명 체인에 포인터를 지정하면, 패키지, SQL 루틴, 트리거, 전역 변수 또는 체인의 각 별명에서의 뷰에 대한 종속성이 기록됩니다. 별명은 별명 체인의 별명 자체를 참조할 수 없으며 이러한 순환은 별명 정의 시에 발견됩니다(SQLSTATE 42916).
- **규정되지 않은 별명 이름 분석:** 규정되지 않은 이름 분석 시, 공용 별명 전에 개인용 별명이 고려됩니다.
- **공용 별명에 대한 제한적 바인딩 유지:** 공용 별명이 패키지의 명령문, SQL 루틴, 트리거, 전역 변수에 대한 기본 표현식 또는 뷰 정의에 사용되면, 같은 이름의 다른 오브젝트가 연속해서 작성되는 것과 상관 없이 공용 별명은 계속해서 이들 오브젝트에서 사용됩니다.
- 아직 존재하지 않는 스키마 이름을 사용하여 별명을 작성하면, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- *호환성*:
  - ALIAS 대신 SYNONYM을 지정할 수 있습니다.

예:

예 1: HEDGES가 테이블 T1에 대해 별명을 작성합니다(둘 다 규정되지 않음).

```
CREATE ALIAS A1 FOR T1
```

별명 HEDGES.A1이 HEDGES.T1에 대해 작성됩니다.

예 2: HEDGES가 테이블에 대해 별명을 작성합니다(둘 다 규정화됨).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

별명 HEDGES.A1이 MCKNIGHT.T1에 대해 작성됩니다.

예 3: HEDGES가 테이블에 대해 별명을 작성합니다(서로 다른 스키마의 별명). HEDGES는 DBADM이 아니며, HEDGES는 스키마 MCKNIGHT에 대해 CREATEIN을 가지고 있지 않습니다.

```
CREATE ALIAS  MCKNIGHT.A1 FOR MCKNIGHT.T1
```

이 예는 실패합니다(SQLSTATE 42501).

예 4: HEDGES가 정의되지 않은 테이블에 대해 별명을 작성합니다(둘 다 규정됨). 참고로 FUZZY.WUZZY는 이전에 없었습니다.

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

이 명령문은 성공하나 경고가 발행됩니다(SQLSTATE 01522).

예 5: HEDGES가 별명에 대해 별명을 작성합니다(둘 다 규정됨).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
CREATE ALIAS HEDGES.A2 FOR HEDGES.A1
```

첫 번째 명령문은 성공합니다(예 2).

두 번째 명령문은 성공하고 MCKNIGHT.T1을 참조하는 HEDGES.A1, 또 다시 HEDGES.A1을 참조하는 HEDGES.A2로 구성되는 별명 체인이 작성됩니다. HEDGES가 MCKNIGHT.T1에 대해 특권을 갖고 있는지 여부는 중요하지 않습니다. 테이블 특권에 관계없이 별명이 작성됩니다.

예 6: A1을 별칭 FUZZYBEAR에 대한 별명으로 지정합니다.

```
CREATE ALIAS A1 FOR FUZZYBEAR
```

예 7: 대규모 조직에 재정 부서 D108과 인사 부서 D577이 있습니다. D108은 DB2 RDBMS에 있는 테이블에 특정 정보를 보관합니다. D577은 Oracle RDBMS에 있는 테이블에 특정 레코드를 보관합니다. DBA는 페더레이티드 시스템 내에서 두 개의 RDBMS를 데이터 소스로 정의하며, 테이블에 각각 DEPTD108과 DEPTD577이라는 별칭을 부여합니다. 페더레이티드 시스템 사용자는 이 테이블 간의 조인을 작성해야 하는데, 영숫자로 된 별칭보다 더 의미있는 이름으로 이들을 참조하고자 할 것입니다. 따라서 사용자는 DEPTD108에 대한 별명으로 FINANCE를, DEPTD577에 대한 별명으로 PERSONNEL을 정의합니다.

```
CREATE ALIAS FINANCE FOR DEPTD108
CREATE ALIAS PERSONNEL FOR DEPTD577
```

*예 8:*카탈로그 뷰 SYSCAT.TABLES에 대해 TABS라는 공용 별명을 작성합니다.

```
CREATE PUBLIC ALIAS TABS FOR SYSCAT.TABLES
```

CREATE AUDIT POLICY

CREATE AUDIT POLICY문은 현재 서버에서 감사 규정을 정의합니다. 규정은 감사할 범주를 판별합니다. 범주는 다른 데이터베이스 오브젝트에 적용되어 해당 오브젝트의 사용을 감사할 수 있는 방법을 판별할 수 있습니다.

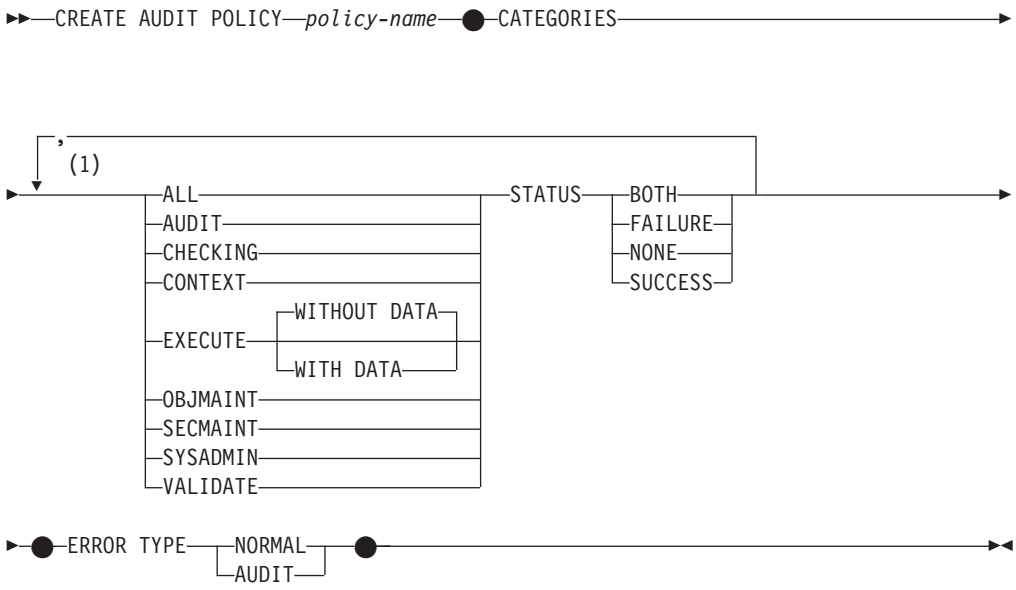
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



주:

1 각 범주는 최대 한 번 지정할 수 있으며(SQLSTATE 42614), ALL이 지정된 경우 다른 어떤 범주도 지정할 수 없습니다(SQLSTATE 42601).

설명

policy-name

감사 규정의 이름. 이 이름은 한 부분의 이름입니다. 이것은 SQL ID(일반 또는 분

리 ID)입니다. *policy-name*은 카탈로그에 이미 설명되어 있는 감사 규정을 식별해서는 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

CATEGORIES
상태가 지정된 하나 이상의 감사 범주 목록입니다. ALL을 지정하지 않은 경우 명시적으로 지정되지 않은 범주의 STATUS는 NONE으로 설정됩니다.

ALL
모든 범주를 동일한 상태로 설정합니다. EXECUTE 범주는 WITHOUT DATA입니다.

AUDIT
감사 설정이 변경될 때 또는 감사 로그에 액세스할 때 레코드를 생성합니다.

CHECKING
데이터베이스 오브젝트 또는 함수에 액세스하거나 처리하기 위한 권한 부여 검사 시도 중 레코드를 생성합니다.

CONTEXT
데이터베이스 조작이 수행될 때 조작 컨텍스트를 표시하기 위해 레코드를 생성합니다.

EXECUTE
SQL문의 실행을 표시하기 위해 레코드를 생성합니다.

WITHOUT DATA 또는 WITH DATA
호스트 변수나 매개변수 표시문자에 대해 제공된 입력 데이터 값이 EXECUTE 범주 일부로 로그해야 하는지 여부를 지정합니다.

WITHOUT DATA
호스트 변수와 매개변수 표시문자에 대해 제공된 입력 데이터 값이 EXECUTE 범주 일부로 로그해야 하는지 여부를 지정합니다. WITHOUT DATA는 디폴트입니다.

WITH DATA
호스트 변수나 매개변수 표시문자에 대해 제공된 입력 데이터 값이 EXECUTE 범주 일부로 로그됨을 지정합니다. 모든 입력 값이 로그되는 것은 아닙니다. 특히 LOB, LONG, XML 및 구조화 유형 매개변수는 널(NULL) 값으로 표시됩니다. 날짜, 시간 및 시간소인 필드는 ISO 형식으로 로그됩니다. 입력 데이터 값은 로그되기 전에 데이터베이스 코드 페이지로 변환됩니다. 코드 페이지 변환이 실패하는 경우 어떤 오류도 리턴되지 않으며 변환되지 않은 데이터가 로그됩니다.

OBJMAINT
데이터 오브젝트가 작성되거나 삭제될 때 레코드를 생성합니다.

SECMAINT

오브젝트 특권, 데이터베이스 특권 또는 DBADM 권한이 부여되거나 취소될 때 레코드를 생성합니다. 데이터베이스 관리 프로그램 보안 구성 매개변수 **sysadm_group**, **sysctrl_group** 또는 **sysmaint_group**이 수정될 때도 레코드가 생성됩니다.

SYSADMIN

SYSADM, SYSMAINT 또는 SYSCTRL 권한이 필요한 조작이 수행될 때 레코드를 생성합니다.

VALIDATE

사용자가 인증될 때 또는 사용자에 관련된 시스템 보안 정보가 검색될 때 레코드를 생성합니다.

STATUS

지정된 범주에 대한 상태를 지정합니다.

BOTH

성공 및 실패 이벤트를 감사합니다.

FAILURE

실패 이벤트만 감사합니다.

SUCCESS

성공 이벤트만 감사합니다.

NONE

이 범주의 이벤트는 감사하지 않습니다.

ERROR TYPE

감사 오류를 리턴할 것인지 또는 무시할 것인지 여부를 지정합니다.

NORMAL

감사에서 생성된 오류가 무시되고 수행되는 조작과 연관된 오류에 대한 SQLCODE만 응용프로그램에 리턴됩니다.

AUDIT

감사 기능 자체에서 발생하는 오류를 포함한 모든 오류가 응용프로그램에 리턴됩니다.

규칙

- AUDIT 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다 (SQLSTATE 5U021). AUDIT 독점 SQL문은 다음과 같습니다.
  - AUDIT
  - CREATE AUDIT POLICY, ALTER AUDIT POLICY 또는 DROP(AUDIT POLICY)

  – DROP(감사 규정과 연관되는 경우 ROLE 또는 TRUSTED CONTEXT)
- AUDIT 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 모든 데이터베이스 파티션 사이에서 한 번에 단 하나의 언커미트된 AUDIT 독점 SQL문만 허용됩니다. 언커미트된 AUDIT 독점 SQL문이 실행 중인 경우, 연속 AUDIT 독점 SQL문은 현재 AUDIT 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 발행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예 :

AUDIT 및 OBJMAINT 범주에 대해서는 성공 및 실패를 감사하고 SECMAINT, CHECKING, VALIDATE 범주에 대해서는 실패만 감사하며 다른 범주에 대해서는 어떤 이벤트도 감사하지 않는 감사 규정을 작성하십시오.

```
CREATE AUDIT POLICY DBAUDPRF
  CATEGORIES AUDIT    STATUS BOTH,
             SECMAINT STATUS FAILURE,
             OBJMAINT STATUS BOTH,
             CHECKING STATUS FAILURE,
             VALIDATE STATUS FAILURE
  ERROR TYPE NORMAL
```

CREATE BUFFERPOOL

CREATE BUFFERPOOL문은 데이터베이스 관리 프로그램이 사용할 새 버퍼 풀을 정의합니다.

특정 데이터베이스 파티션에서 *크기*를 무시할 수 있는 경우, 파티션된 데이터베이스에서는 각 데이터베이스 파티션에 대한 디폴트 버퍼 풀 정의가 지정됩니다. 또한 파티션된 데이터베이스에서 데이터베이스 파티션 그룹이 지정되어 있지 않은 경우, 모든 데이터베이스에 버퍼 풀이 정의됩니다. 데이터베이스 파티션 그룹이 지정되어 있는 경우, 해당 데이터베이스 파티션 그룹에 있는 데이터베이스 파티션에만 버퍼 풀이 작성됩니다.

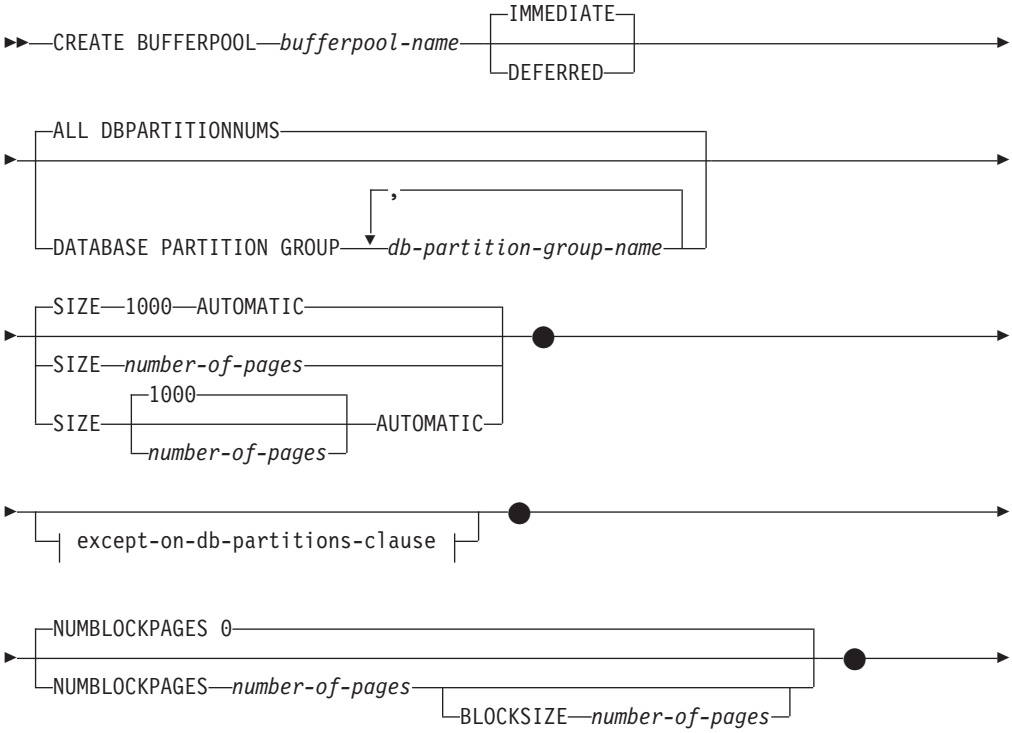
호출

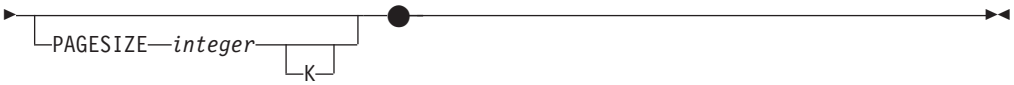
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

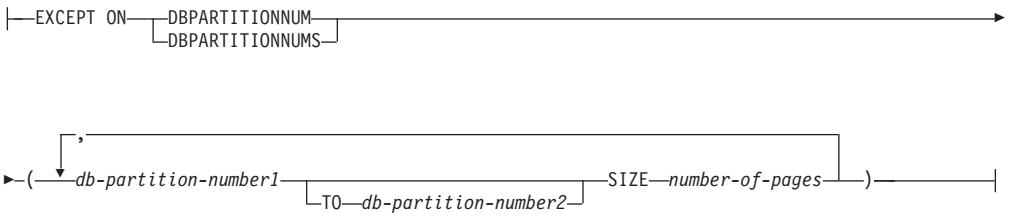
명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 SYSCTRL 권한을 포함해야 합니다.

구문





except-on-db-partitions-clause:



설명

bufferpool-name

버퍼 풀의 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이것은 SQL ID(일반 또는 분리 ID)입니다. *bufferpool-name*은 카탈로그에 이미 존재하는 버퍼 풀을 식별하면 안됩니다(SQLSTATE 42710). *bufferpool-name*은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

IMMEDIATE 또는 DEFERRED

버퍼 풀이 즉시 작성되는지 표시합니다.

IMMEDIATE

버퍼 풀이 즉시 작성됩니다. 데이터베이스 공유 메모리에 새 버퍼 풀을 할당하기 위해 예약된 스페이스가 충분하지 않으면(SQLSTATE 01657) 이 명령문이 DEFERRED로 실행됩니다.

DEFERRED

데이터베이스가 비활성화되면 버퍼 풀이 작성됩니다. 데이터베이스에서 모든 응용프로그램의 연결을 끊어야 합니다. DB2는 필요한 메모리를 시스템에서 할당하므로 예약 메모리 스페이스가 필요하지 않습니다.

ALL DBPARTITIONNUMS

데이터베이스의 모든 데이터베이스 파티션에 해당 버퍼 풀이 작성됩니다.

DATABASE PARTITION GROUP *db-partition-group-name, ...*

버퍼 풀 정의가 적용될 데이터베이스 파티션 그룹을 식별합니다. 해당 매개변수가 지정되어 있는 경우, 해당 데이터베이스 파티션 그룹에 있는 데이터베이스 파티션에만 버퍼 풀이 작성됩니다. 각 데이터베이스 파티션 그룹은 현재 데이터베이스에 있어야 합니다(SQLSTATE 42704). DATABASE PARTITION GROUP절이 지정되지 않은 경우, 이 버퍼 풀은 모든 데이터베이스 파티션(및 해당 데이터베이스에 이어서 추가되는 모든 데이터베이스 파티션)에 작성됩니다.

SIZE

버퍼 풀의 *크기*를 지정합니다. 파티션된 데이터베이스에서 이 *크기*는 버퍼 풀이 있는 모든 데이터베이스 파티션의 디폴트 *크기*가 됩니다. 디폴트는 1000페이지입니다.

number-of-pages

새 버퍼 풀에 대한 페이지 수.

AUTOMATIC

해당 버퍼 풀의 자체 성능 조정 기능을 사용 가능하게 합니다. 데이터베이스 관리 프로그램은 워크로드 요구사항에 맞춰 버퍼 풀 *크기*를 조정합니다. 지정된 내재적 또는 명시적 페이지 수가 버퍼 풀의 최초 *크기*로 쓰입니다.

NUMBLOCKPAGES *number-of-pages*

블록 기반 영역에 있어야 할 페이지 수를 지정합니다. 이 페이지 수는 버퍼 풀에 대한 총 페이지 수의 98%를 넘지 않아야 합니다(SQLSTATE 54052). 0 값을 지정하면 블록화 입출력을 사용할 수 없게 됩니다. 실제로 사용되는 NUMBLOCKPAGES 값은 BLOCKSIZE의 배수입니다.

BLOCKSIZE *number-of-pages*

블록에 페이지 수를 지정합니다. 블록 *크기*는 2 - 256 사이의 값이어야 합니다 (SQLSTATE 54053). 디폴트값은 32입니다.

except-on-db-partitions-clause

버퍼 풀의 *크기*가 디폴트 *크기*와 다른 데이터베이스 파티션 또는 데이터베이스 파티션들을 지정합니다. 이 절이 지정되지 않은 경우, 모든 데이터베이스 파티션이 이 버퍼 풀에 대해 지정된 것과 같은 *크기*의 버퍼 풀을 갖게 됩니다.

EXCEPT ON DBPARTITIONNUMS

지정된 특정 데이터베이스 파티션을 가리키는 키워드. DBPARTITIONNUM은 DBPARTITIONNUMS와 동의어입니다.

db-partition-number1

버퍼 풀이 작성된 데이터베이스 파티션에 포함되어 있는 특정 데이터베이스 파티션 번호를 지정합니다.

TO *db-partition-number2*

데이터베이스 파티션 번호의 범위를 지정하십시오. *db-partition-number2* 값은 *db-partition-number1* 값보다 크거나 같아야 합니다(SQLSTATE 428A9). 지정된 데이터베이스 파티션 번호 사이 및 번호를 포함하는 모든 데이터베이스 파티션은 버퍼 풀이 작성되는 데이터베이스 파티션에 포함되어야 합니다(SQLSTATE 42729).

SIZE *number-of-pages*

페이지 수로 지정되는 버퍼 풀의 *크기*입니다.

PAGESIZE *integer* **[K]**

버퍼 풀에 사용되는 페이지 *크기*를 정의합니다. 접미부 K가 없는 *integer*에 유효한

값은 4096, 8192, 16 384, 또는 32 768입니다. 접미부 K가 있는 *integer*에 유효한 값은 4, 8, 16 또는 32입니다. *integer*와 K 사이에는 공백이 없을 수도 있고 임의의 수의 공백이 들어갈 수도 있습니다. 페이지 크기가 이러한 값 중 하나가 아니면 오류가 리턴됩니다(SQLSTATE 428DE).

디폴트값은 데이터베이스 작성시 설정된 **pagesize** 데이터베이스 구성 매개변수에 의해 제공됩니다.

주

- DEFERRED 옵션을 사용하여 버퍼 풀을 작성한 경우에는 이 버퍼 풀에 작성된 테이블 스페이스는 다음에 데이터베이스가 활성화될 때까지 같은 페이지 크기의 작은 시스템 버퍼 풀을 사용합니다. 버퍼 풀을 활성화하고 새 버퍼 풀에 테이블 스페이스를 지정하려면 데이터베이스를 재시작해야 합니다. 디폴트 옵션은 IMMEDIATE입니다.
- 데이터베이스 관리 프로그램과 응용프로그램의 나머지 부분에 대해서뿐 아니라 전체 버퍼 풀에 대해 머신상에 충분한 실제 메모리가 있어야 합니다. DB2가 일반 버퍼 풀에 대한 메모리를 확보할 수 없는 경우에는 페이지 크기(4K, 8K, 16K, 32K) 각각에 대하여 작은 시스템 버퍼 풀을 사용하여 시작하려고 시도합니다. 이 경우, 경고가 사용자에게 리턴되어(SQLSTATE 01626) 모든 테이블 스페이스의 페이지가 시스템 버퍼 풀을 사용하게 됩니다.
- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - DBPARTITIONNUMS 대신 NODES를 지정할 수 있습니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.

CREATE DATABASE PARTITION GROUP

CREATE DATABASE PARTITION GROUP문은 데이터베이스 내에 새로운 데이터베이스 파티션 그룹을 정의한 후 데이터베이스 파티션 그룹에 데이터베이스 파티션을 지정하고, 시스템 카탈로그에 데이터베이스 파티션 그룹 정의를 기록합니다.

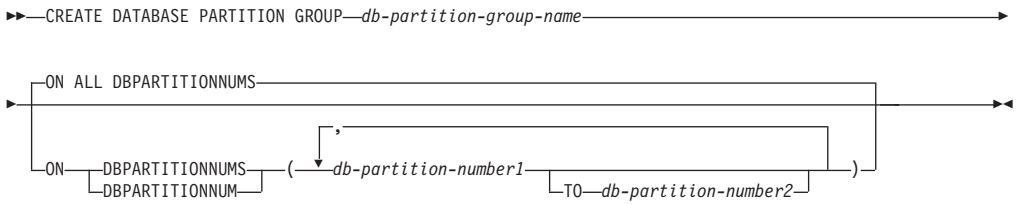
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 SYSCTRL 권한을 포함해야 합니다.

구문



설명

db-partition-group-name

데이터베이스 파티션 그룹의 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이것은 SQL ID(일반 또는 분리 ID)입니다. *db-partition-group-name*은 카탈로그에 이미 존재하는 데이터베이스 파티션 그룹을 식별하면 안됩니다(SQLSTATE 42710). *db-partition-group-name*은 'SYS' 또는 'IBM'으로 시작하면 안됩니다(SQLSTATE 42939).

ON ALL DBPARTITIONNUMS

데이터베이스 파티션 그룹 작성시 데이터베이스에 대해 정의된 모든 데이터베이스 파티션(db2nodes.cfg 파일)에 데이터베이스 파티션 그룹이 정의되도록 지정합니다.

데이터베이스 파티션이 데이터베이스 시스템에 추가되면, ALTER DATABASE PARTITION GROUP문은 데이터베이스 파티션 그룹(IBMDEFAULTGROUP 포함)에 새 데이터베이스 파티션을 포함하도록 실행되어야 합니다. 또한 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 발행하여 데이터를 데이터베이스 파티션으로 이동시켜야 합니다.

ON DBPARTITIONNUMS

데이터베이스 파티션 그룹에 있는 데이터베이스 파티션을 지정합니다. DBPARTITIONNUM은 DBPARTITIONNUMS와 동의어입니다.

db-partition-number1

데이터베이스 파티션 번호를 지정합니다. (이전 버전과 호환되도록 NODE*nnnnn* 형식의 *node-name*을 지정할 수 있습니다.)

TO *db-partition-number2*

데이터베이스 파티션 번호의 범위를 지정하십시오. *db-partition-number2* 값은 *db-partition-number1* 값보다 크거나 같아야 합니다(SQLSTATE 428A9). 지정된 데이터베이스 파티션 번호 사이 및 번호를 포함하는 모든 데이터베이스 파티션은 데이터베이스 파티션 그룹에 포함됩니다.

규칙

- 번호 순으로 지정된 각 데이터베이스 파티션은 db2nodes.cfg 파일에 정의되어야 합니다(SQLSTATE 42729).
- ON DBPARTITIONNUMS절에 나열된 각 *db-partition-number*는 한 번만 표시되어야 합니다(SQLSTATE 42728).
- 유효한 *db-partition-number*는 0과 999 사이입니다(SQLSTATE 42729).
- 데이터베이스 파티션 서버 요청이 보류 또는 진행 중인 경우 CREATE DATABASE PARTITION GROUP문이 실패할 수 있습니다(SQLSTATE 55071). 새 데이터베이스 파티션 서버가 온라인으로 인스턴스에 추가되고 모든 응용프로그램이 새 데이터베이스 파티션 서버를 인식하지 않는 경우, 이 명령문도 실패할 수 있습니다(SQLSTATE 55077).

주

- 이 명령문은 데이터베이스 파티션 그룹에 대한 분산 맵을 작성합니다. 각 분산 맵에 대해 분산 맵 ID(PMAP_ID)가 생성됩니다. 이 정보는 카탈로그에 기록되며, SYSCAT.DBPARTITIONGROUPS와 SYSCAT.PARTITIONMAPS에서 검색할 수 있습니다. 분산 맵의 각 항목은 해시된 모든 행이 있는 목표 데이터베이스 파티션을 지정합니다. 단일 파티션 데이터베이스 파티션 그룹의 경우, 해당하는 분산 맵에는 단 한 개의 항목만 있습니다. 다중 파티션 데이터베이스 파티션 그룹의 경우, 해당하는 분산 맵의 32768 항목이 디폴트값이며 데이터베이스 파티션 번호가 라운드 로빈 방식으로 맵 항목에 할당됩니다.
- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - DBPARTITIONNUMS 대신 NODES를 지정할 수 있습니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.

예:

0, 1, 2, 5, 7 및 8로 정의된 6개의 데이터베이스 파티션이 있는 파티션 데이터베이스가 있다고 가정하십시오.

- 여섯 개의 모든 데이터베이스 파티션에 MAXGROUP이라는 데이터베이스 파티션 그룹을 작성할 경우, 명령문은 다음과 같습니다.

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- 데이터베이스 파티션 0, 1, 2, 5 및 8에 MEDGROUP이라는 데이터베이스 파티션 그룹을 작성할 경우, 명령문은 다음과 같습니다.

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- 데이터베이스 파티션 7에 MINGROUP이라는 단일 파티션 데이터베이스 파티션 그룹을 작성할 경우, 명령문은 다음과 같습니다.

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

CREATE EVENT MONITOR

CREATE EVENT MONITOR문은 데이터베이스를 사용할 때 발생하는 특정 이벤트를 기록할 모니터를 정의합니다. 각 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.

이 명령문을 사용하여 여러 다른 유형의 이벤트 모니터를 작성할 수 있습니다. 다음 유형 중 다섯 가지는 따로 설명되어 있으며(관련 링크 참조) 나머지 유형은 여기에 설명되어 있습니다. 이벤트 모니터의 유형은 다음에 따로 설명되어 있습니다.

- 활동. 이벤트 모니터는 데이터베이스를 사용할 때 발생하는 활동 이벤트를 기록합니다. 통계 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.
- 잠금. 이벤트 모니터는 데이터베이스를 사용할 때 발생하는 잠금 관련 이벤트를 기록합니다. 모든 레코드는 비형식화 이벤트 테이블에서 수집됩니다.
- 통계. 이벤트 모니터는 데이터베이스를 사용할 때 발생하는 통계 이벤트를 기록합니다. 통계 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.
- 임계값 위반. 이벤트 모니터는 데이터베이스를 사용할 때 발생하는 임계값 위반 이벤트를 기록합니다. 통계 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.
- 작업 단위(UOW). 이벤트 모니터는 작업 단위(UOW)가 완료될 때 이벤트를 기록합니다. 모든 레코드는 비형식화 이벤트 테이블에서 수집됩니다.

호출

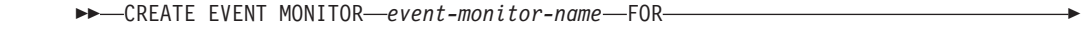
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

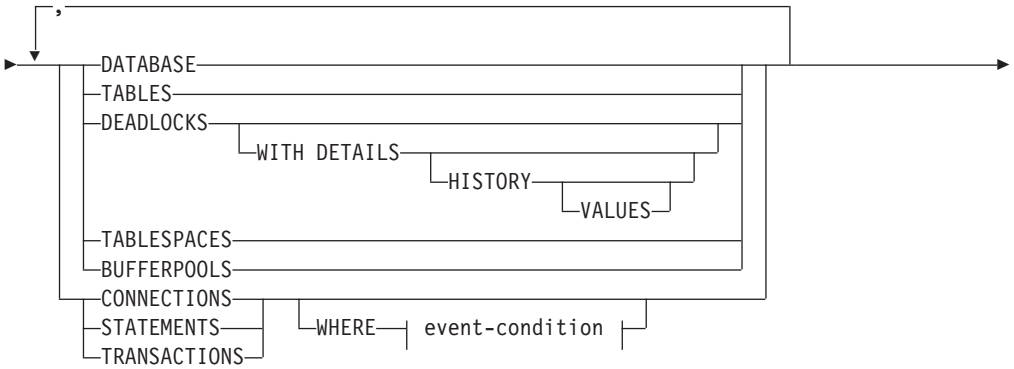
권한 부여

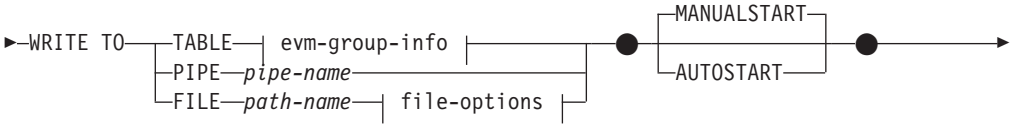
명령문의 권한 부여 ID는 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

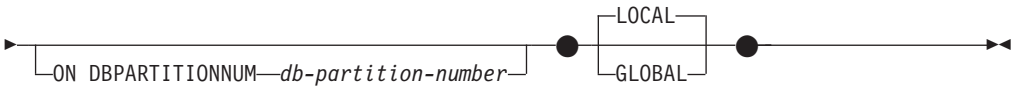
- DBADM 권한
- SQLADM 권한

구문

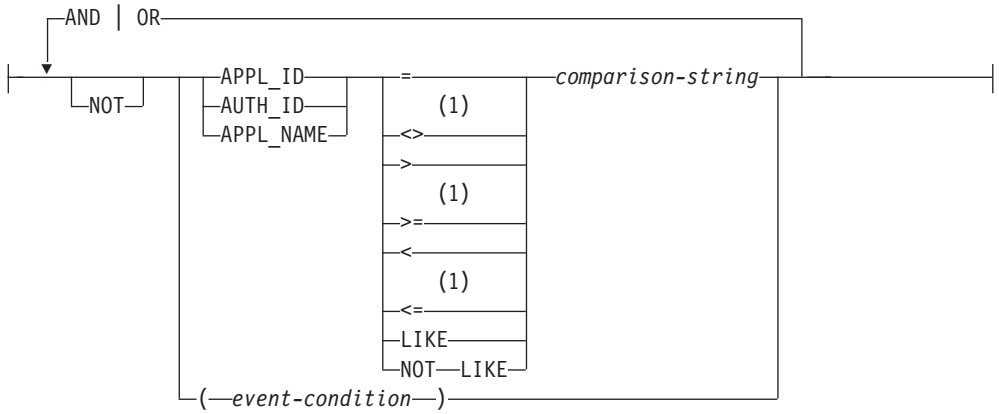




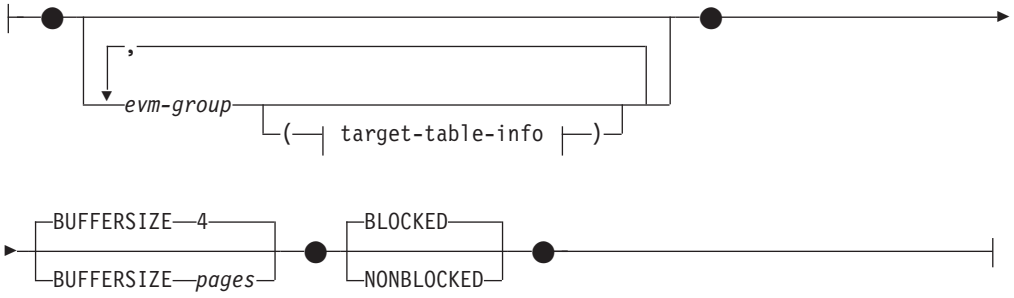




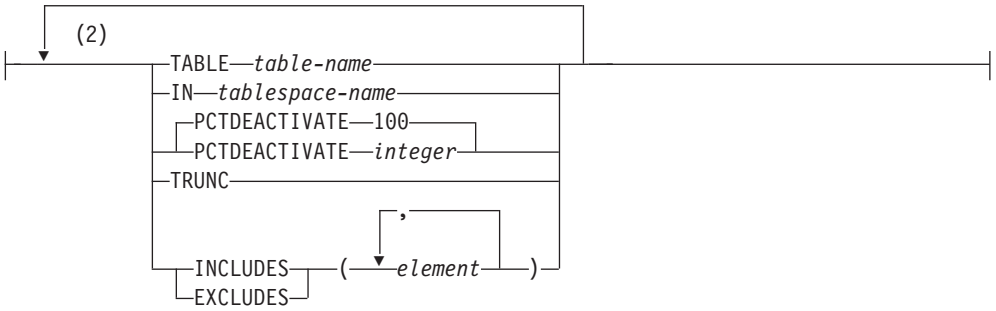
event-condition:



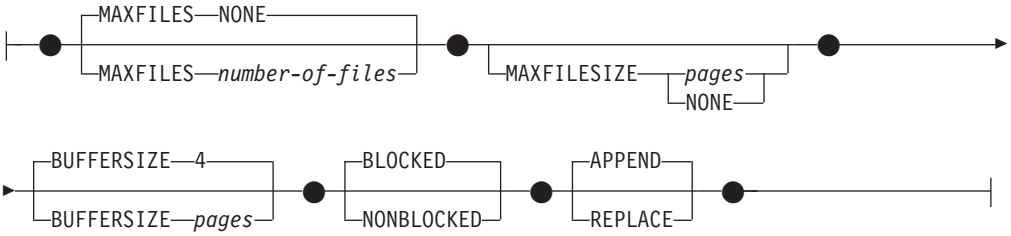
evm-group-info:



target-table-info:



file-options:



주:

1 다른 형태의 연산자도 지원됩니다.

2 각 절을 한 번만 지정할 수 있습니다.

설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름으로, SQL ID(일반 또는 분리)입니다. *event-monitor-name*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 레코드의 유형을 소개합니다.

DATABASE

마지막 응용프로그램이 데이터베이스에서 연결을 끊을 때 이벤트 모니터가 데이터베이스 이벤트를 기록합니다.

TABLES

마지막 응용프로그램이 데이터베이스에서 연결을 끊을 때 사용 중인 각 테이블에 대해 이벤트 모니터가 테이블 이벤트를 기록합니다. 파티션된 테이블의 경우 테이블 이벤트는 각 활성화된 테이블의 각 데이터 파티션에 대해 기록됩니다. 활성 테이블은 데이터베이스에 처음 연결한 이후 변경된 테이블입니다.

DEADLOCKS

주: 이 옵션은 사용되지 않습니다. 더 이상 사용할 것을 권장하지 않으며 추후 릴리스에서 제거될 수 있습니다. 잠금 시간종료, 잠금 대기 및 교착 상태와 같은 잠금 관련 이벤트를 모니터하려면 CREATE EVENT MONITOR FOR LOCKING 문을 사용하십시오.

교착 상태가 발생할 때마다 이벤트 모니터가 교착 상태를 기록하도록 저장합니다.

WITH DETAILS

이벤트 모니터가 교착 상태와 관련된 각 응용프로그램에 대하여 매우 자세한 교착 상태 연결 이벤트를 생성하도록 지정합니다. 여기에 포함되는 세부 정보는 다음과 같습니다.

- 교착 상태가 발생했을 때 응용프로그램이 실행하고 있던 명령문에 대한 정보(예: 명령문 텍스트)
- 교착 상태가 발생했을 때 응용프로그램이 보유한 잠금. 파티션된 데이터베이스 환경에서는 교착 상태가 발생했을 때 응용프로그램이 잠금을 대기하고 있던 데이터베이스 파티션의 잠금만 잠금에 포함됩니다. 파티션된 테이블의 경우 데이터 파티션 ID가 잠금에 포함됩니다.

HISTORY

이벤트 모니터가 다음 사항도 포함하도록 지정합니다.

- 참여 노드의 현재 작업 단위(UOW)(이전 작업 단위(UOW)에서 열린 WITH HOLD 커서 포함)에 있는 모든 명령문의 실행기록. 언커미트 읽기(UR) 분리 레벨에서 실행된 SELECT문은 명령문 실행 기록에 포함되지 않습니다.
- 2진 형식에서 각 SQL문에 대한 명령문 컴파일 환경(해당되는 경우)

VALUES

이벤트 모니터가 다음 사항도 포함하도록 지정합니다.

- 각 SQL문에 대한 입력 변수로 사용되는 데이터 값. 해당 데이터 값은 LOB 데이터, 긴 데이터, 구조화된 유형 데이터 또는 XML 데이터를 포함하지 않습니다.

DEADLOCKS, DEADLOCKS WITH DETAILS, DEADLOCKS WITH DETAILS HISTORY 또는 DEADLOCKS WITH DETAILS HISTORY VALUES 중 하나만 단일 CREATE EVENT MONITOR문에 지정할 수 있습니다(SQLSTATE 42613).

TABLESPACES

마지막 응용프로그램이 데이터베이스에서 연결을 끊을 때 각 테이블 스페이스에 대해 이벤트 모니터가 테이블 스페이스 이벤트를 기록하도록 지정합니다.

BUFFERPOOLS
마지막 응용프로그램이 데이터베이스에서 연결을 끊을 때 이벤트 모니터가 버퍼 풀 이벤트를 기록하도록 지정합니다.

CONNECTIONS
마지막 응용프로그램이 데이터베이스에서 연결을 끊을 때 이벤트 모니터가 연결 이벤트를 기록하도록 지정합니다.

STATEMENTS
SQL문이 실행을 종료할 때마다 이벤트 모니터가 명령문을 기록하도록 지정합니다.

TRANSACTIONS

주: 이 옵션은 사용되지 않습니다. 더 이상 사용할 것을 권장하지 않으며 추후 릴리스에서 제거될 수 있습니다. 트랜잭션 이벤트를 모니터하려면 CREATE EVENT MONITOR FOR UNIT OF WORK 문을 사용하십시오.

트랜잭션이 종료될 때마다(즉, 커미트 또는 롤백 조작이 있을 때마다) 이벤트 모니터가 트랜잭션 이벤트를 기록하도록 지정합니다.

WHERE event-condition
CONNECTION, STATEMENT 또는 TRANSACTION 이벤트를 발생시키는 연결을 판별하는 필터를 정의합니다. 이벤트 조건의 결과가 특정 연결에 대해 참이면, 그 연결은 요청된 이벤트를 생성합니다.

이 절은 WHERE절의 특별한 형태로, 표준 검색 조건과 혼돈해서는 안됩니다.

응용프로그램이 특정 이벤트 모니터에 대한 이벤트를 생성하는지 판별하기 위해 WHERE절은 다음에 대해 평가됩니다.

- 이벤트 모니터가 처음 작동될 때 각 활성 연결에 대해
- 그런 다음 연결 시 데이터베이스에 대한 새로운 각 연결에 대해

WHERE절은 각 이벤트에 대해 평가되지 않습니다.

WHERE절을 지정하지 않을 경우, 지정된 이벤트 유형의 모든 이벤트가 모니터됩니다.

데이터베이스 코드 페이지에서 이 event-condition의 길이는 32 678바이트를 초과할 수 없습니다(SQLSTATE 22001).

APPL_ID
연결시 CONNECTION, STATEMENT 또는 TRANSACTION 이벤트(지정된 것)가 발생하는지 판별하기 위해 각 연결의 응용프로그램 ID를 *comparison-string*과 비교할지 여부를 지정합니다.

AUTH_ID
연결시 CONNECTION, STATEMENT 또는 TRANSACTION 이벤트(지

정된 것)가 발생하는지 판별하기 위해 각 연결의 권한 부여 ID를 *comparison-string*과 비교할지 여부를 지정합니다.

APPL_NAME

연결시 CONNECTION, STATEMENT 또는 TRANSACTION 이벤트(지정된 것)가 발생하는지 판별하기 위해 각 연결의 응용프로그램 이름을 *comparison-string*과 비교할지 여부를 지정합니다.

응용프로그램 이름은 마지막 경로 분리문자 다음에 오는 응용프로그램 파일 이름의 처음 20바이트입니다.

comparison-string

데이터베이스에 연결된 각 응용프로그램의 APPL_ID, AUTH_ID 또는 APPL_NAME으로 비교되는 문자열입니다. *comparison-string*은 문자열 상수여야 합니다. 즉, 호스트 변수와 다른 문자열 표현식은 허용되지 않습니다.

WRITE TO

데이터에 대한 목표를 사용합니다.

TABLE

이벤트 모니터 데이터의 목표가 데이터베이스 테이블 세트임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 하나 이상의 논리 데이터 그룹에 나눈 다음 각 그룹을 별도의 테이블에 삽입합니다. 목표 테이블이 있는 그룹의 데이터는 보존되고 목표 테이블이 없는 그룹의 데이터는 버립니다. 그룹에 포함된 각 모니터 요소는 같은 이름으로 된 테이블 컬럼에 맵핑됩니다. 해당하는 테이블 컬럼이 있는 요소만 테이블에 삽입되고, 다른 요소들은 버립니다.

evm-group-info

논리 데이터 그룹에 대한 목표 테이블을 정의합니다. 이 절은 기록할 각 그룹화에 대하여 지정해야 합니다. 그러나 evm-group-info절을 지정하지 않으면 해당 이벤트 모니터 유형에 대한 모든 그룹이 기록됩니다.

evm-group

목표 테이블을 정의할 논리 데이터 그룹을 식별합니다. 이 값은 다음 표에서와 같이 이벤트 모니터의 유형에 따라 다릅니다.

이벤트 모니터 유형	**evm-group** 값
데이터베이스	• DB • CONTROL[1] • DBMEMUSE
테이블	• TABLE • CONTROL[1]

이벤트 모니터 유형	evm-group 값
교착 상태	• CONNHEADER • DEADLOCK • DLCONN • CONTROL[1]
세부사항이 있는 교착 상태	• CONNHEADER • DEADLOCK • DLCONN[2] • DLLOCK[3] • CONTROL[1]
세부 실행기록이 있는 교착 상태	• CONNHEADER • DEADLOCK • DLCONN[2] • DLLOCK[3] • STMTHIST • CONTROL[1]
세부 실행기록 값이 있는 교착 상태	• CONNHEADER • DEADLOCK • DLCONN[2] • DLLOCK[3] • STMTHIST • STMTVALS • CONTROL[1]
테이블 스페이스	• TABLESPACE • CONTROL[1]
버퍼 풀	• BUFFERPOOL • CONTROL[1]
연결	• CONNHEADER • CONN • CONTROL[1] • CONNMEMUSE
명령문	• CONNHEADER • STMT • SUBSECTION[4] • CONTROL[1]

이벤트 모니터 유형	evm-group 값
트랜잭션	• CONNHEADER • XACT • CONTROL[1]

[1] 논리 데이터 그룹의 dbheader(conn_time 요소만 해당), 시작 및 오버플로우가 모두 CONTROL 그룹에 기록됩니다. 이벤트 모니터가 블록화되어 있지 않고 이벤트를 버린 경우에는 오버플로우 그룹이 기록됩니다.

[2] DETAILED_DLCONN 이벤트에 해당합니다.

[3] 각 DETAILED_DLCONN 이벤트 내에서 발생하는 LOCK 논리 데이터 그룹에 해당합니다.

[4] 파티션된 데이터베이스 환경에 대해서만 작성됩니다.

target-table-info

그룹에 대한 목표 테이블을 식별합니다. *target-table-info* 값을 지정하지 않으면 CREATE EVENT MONITOR 처리가 다음과 같이 진행됩니다.

- 파생된 테이블 이름이 사용됩니다(아래에 설명되어 있음).
- 디폴트 테이블 스페이스가 선택됩니다(아래에 설명되어 있음).
- 모든 요소가 포함됩니다.
- PCTDEACTIVATE와 TRUNC가 지정되지 않습니다.

TABLE *table-name*

목표 테이블의 이름을 지정합니다. 목표 테이블은 파티션되지 않은 테이블이어야 합니다. 규정되지 않은 이름일 경우에는 디폴트 테이블 스키마는 현재 권한 부여 ID에 대한 스키마입니다. 이름을 제공하지 않을 경우, 규정되지 않은 이름이 다음과 같이 *evm-group* 및 *event-monitor-name*으로부터 파생됩니다.

```
substring(evm-group CONCAT "_"
  CONCAT event-monitor-name,1,128)
```

IN *tablespace-name*

테이블이 작성될 테이블 스페이스를 정의합니다. 테이블 스페이스 이름을 제공하지 않으면 다음과 같이 테이블 스페이스가 선택됩니다.

```
IF table space IBMDEFAULTGROUP over which the user
        has USE privilege exists
         THEN choose it
      ELSE IF a table space over which the user
        has USE privilege exists
         THEN choose it
      ELSE issue an error (SQLSTATE 42727)
```

PCTDEACTIVATE *integer*

테이블이 DMS 테이블 스페이스에 작성되는 경우 PCTDEACTIVATE 매개변수는 이벤트 모니터가 자동으로 비활성화될 때까지 테이블 스페이스가 얼마만큼 차야 하는지 지정합니다. 0부터 100까지의 값을 지정할 수 있으며 이 값은 백분율을 나타냅니다. 디폴트값은 100입니다. 100은 테이블 스페이스가 완전히 찼을 때 이벤트 모니터가 비활성화됨을 의미합니다. 이 옵션은 SMS 테이블 스페이스의 경우 무시됩니다.

목표 테이블 스페이스에서 자동 크기 조정이 사용 가능하면 PCTDEACTIVATE 매개변수를 100으로 설정할 것을 권장합니다.

TRUNC

STMT_TEXT 및 STMT_VALUE_DATA 컬럼이 VARCHAR(*n*)로 정의되도록 지정합니다. 여기서, *n*은 테이블 행에 맞는 최대 크기입니다. 이 경우 *n* 바이트보다 긴 데이터는 절단됩니다. 다음 예는 *n*의 값을 계산하는 방법을 나타냅니다. 다음과 같이 가정합니다.

- 테이블이 32K 페이지를 사용하는 테이블 스페이스에 작성됩니다.
- 테이블에 있는 다른 모든 컬럼의 총 길이가 357바이트입니다.

이 경우 테이블 행의 최대 크기는 32677바이트입니다. 따라서 해당 요소는 VARCHAR(32316) 즉, 32677 - 357 - 4로 정의됩니다. TRUNC가 지정되지 않으면 컬럼이 CLOB(64K)로 정의됩니다. 세부사항이 있는 교착 상태 이벤트 모니터의 경우에는 STMT 이벤트 그룹, STMT_HISTORY 이벤트 그룹 및 DLCONN 이벤트 그룹에서 STMT_TEXT를 찾을 수 있습니다. STMT_VALUE_DATA는 DATA_VALUE 이벤트 그룹에서 찾을 수 있습니다.

INCLUDES

다음 요소가 테이블에 포함되도록 지정합니다.

EXCLUDES

다음 요소가 테이블에 포함되지 않도록 지정합니다.

element

모니터 요소를 식별합니다. 요소 정보는 다음 중 한 가지 형식으로 제공할 수 있습니다.

- 요소 정보를 지정하지 않습니다. 이 경우 모든 요소가 CREATE TABLE문에 포함됩니다.
- INCLUDES (element1, element2, ..., element*n*) 형식으로 포함할 요소를 지정합니다. 이런 요소에 대해서는 테이블 컬럼만 작성됩니다.
- EXCLUDES (element1, element2, ..., element*n*) 형식으로 제외시킬 요소를 지정합니다. 이런 요소를 제외한 모든 요소에 대한 테이블 컬럼만 작성됩니다.

그룹에 대한 전체 요소 목록을 포함하는 CREATE EVENT MONITOR문을 빌드하려면 db2evtbl 명령을 사용하십시오.

BUFFERSIZE *pages*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 단위). 테이블 이벤트 모니터는 버퍼의 모든 데이터를 삽입하고 버퍼가 모두 처리되면 COMMIT를 발행합니다. 버퍼가 클수록 이벤트 모니터가 사용하는 커미트 범위도 큽니다. 활발하게 활동 중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 가지고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두 배의 버퍼링을 사용합니다.

각 버퍼의 디폴트 *크기*는 4페이지입니다(두 개의 16K 버퍼가 할당됨). 최소 *크기*는 1페이지입니다. 버퍼는 힙으로부터 할당되므로 버퍼의 최대 *크기*는 모니터 힙의 *크기*에 따라 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 *크기*를 늘리십시오.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

NONBLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 기다리지 않도록 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤트 모니터만큼 데이터베이스 작업의 속도를 저하시키지는 않습니다. 그러나 NONBLOCKED 이벤트 모니터는 활발히 사용 중인 시스템에서 데이터가 유실될 수 있습니다.

PIPE

이벤트 모니터 데이터의 목표가 Named Pipe임을 지정합니다. 이벤트 모니터

는 데이터를 단일 스트림의 파이프(즉, 하나인 것처럼 보이는 무한정으로 긴 파일)에 기록합니다. 데이터를 파이프에 기록할 때, 이벤트 모니터는 블록화된 쓰기를 수행하지 않습니다. 파이프 버퍼에 스페이스가 없으면, 이벤트 모니터에서는 데이터를 버립니다. 데이터 유실이 없도록 하려면, 즉시 데이터를 읽도록 해야 합니다.

pipe-name

이벤트 모니터가 데이터를 기록할 파이프의 이름(AIX에서 FIFO)입니다.

파이프의 이름 지정 규칙은 플랫폼에 따라 다릅니다. UNIX 운영 체제의 경우, 파이프 이름은 파일 이름과 같이 취급됩니다. 그러므로 상대 파이프 이름이 허용되고, 상대 경로 이름과 같이 취급됩니다(아래의 *path-name* 참조). 그러나 Windows에는 파이프 이름에 대한 특수한 구문이 있으며 그 결과로 절대 파이프 이름이 요구됩니다.

파이프의 존재 여부는 이벤트 모니터 작성시 점검되지 않습니다. 이벤트 모니터가 활성화될 때 읽을 파이프를 작성하고 여는 것은 모니터링 응용프로그램에서 수행됩니다. 이 때 파이프를 읽을 수 없으면, 이벤트 모니터 자체가 작동 중지되고 오류가 기록됩니다. 즉, 이벤트 모니터가 데이터베이스 시작 시 AUTOSTART 옵션에 의해 활성화된 경우, 이벤트 모니터는 시스템 오류 로그에 오류를 기록합니다. 이벤트 모니터가 SET EVENT MONITOR STATE SQL문으로 활성화된 경우, 해당 명령문은 실패합니다(SQLSTATE 58030).

FILE

이벤트 모니터 데이터의 목표가 파일(또는 파일 세트)임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 확장자가 “evt”인 일련의 8문자 순서화 파일로 작성합니다 (예: 00000000.evt, 00000001.evt 및 00000002.evt). 데이터는 더 작은 여러 조각으로 분리되더라도 하나의 논리 파일로 간주됩니다. 즉, 데이터 스트림의 시작은 00000000.evt 파일의 첫 번째 바이트이고, 데이터 스트림의 끝은 nnnnnnnn.evt 파일의 마지막 바이트입니다.

각 파일의 최대 *크기*뿐 아니라 최대 파일 수도 정의할 수 있습니다. 이벤트 모니터는 단일 이벤트 레코드를 두 파일에 걸쳐 분리시킬 수 없습니다. 그러나 이벤트 모니터는 관련 레코드들을 두 개의 다른 파일로 작성할 수 있습니다. 이벤트 파일을 처리할 때 그러한 관련 정보를 추적하는 것은 이 데이터를 사용하는 응용프로그램에서 수행됩니다.

path-name

이벤트 모니터가 이벤트 파일 데이터를 기록해야 하는 디렉토리의 이름입니다. 경로는 서버에 알려져 있어야 하지만 경로 그 자체는 다른 파티션에 상주할 수 있습니다(예: UNIX 시스템에서는 NFS 마운트 파일이 될 수 있음). *path-name*을 지정할 때 문자열 상수를 사용해야 합니다.

CREATE EVENT MONITOR를 실행할 때 디렉토리는 없어도 됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 경로의 존재 여부에 대한 점검이 수행됩니다. 이 때 목표 경로가 존재하지 않으면 오류가 발생합니다 (SQLSTATE 428A3).

AIX에서는 루트 디렉토리로 시작하고 Windows에서는 디스크 ID로 시작하는 절대 경로가 지정되는 경우, 지정된 경로가 사용되는 경로가 됩니다. 상대 경로(루트로 시작하지 않는 경로)를 지정할 경우, 데이터베이스 디렉토리에 있는 DB2EVENT 디렉토리에 대한 상대 경로가 사용됩니다.

상대 경로를 지정하면, DB2EVENT 디렉토리를 사용하여 그 경로를 절대 경로로 변환할 수 있습니다. 그러면, 절대 및 상대 경로 사이에 어떤 구별도 만들어지지 않습니다. 절대 경로는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 저장됩니다.

같은 목표 경로를 갖는 두 개 이상의 이벤트 모니터를 지정할 수 있습니다. 그러나 처음에 이벤트 모니터 중 하나가 활성화되고 대상 디렉토리가 비어 있지 않으면, 다른 이벤트 모니터 중 어떤 것도 활성화시킬 수 없습니다.

file-options

파일 형식에 대한 옵션을 지정합니다.

MAXFILES NONE

이벤트 모니터가 작성하는 이벤트 파일 수에 제한이 없음을 지정합니다. 이는 디폴트값입니다.

MAXFILES *number-of-files*

언제든지 특정 이벤트 모니터에 대해 존재할 이벤트 모니터 파일 수에 제한이 있음을 지정합니다. 이벤트 모니터는 다른 파일을 작성해야 할 때마다, 디렉토리에 있는 .evt 파일의 수가 *number-of-files*보다 적은지 확인합니다. 이미 이 한계에 도달해 있으면, 이벤트 모니터는 스스로 작동 중지됩니다.

응용프로그램이 이벤트 파일이 작성된 후에 디렉토리에서 파일을 제거할 경우, 이벤트 모니터가 작성할 수 있는 총 파일 수는 *number-of-files*를 초과할 수 있습니다. 이 옵션은 이벤트 데이터가 지정된 디스크 스페이스의 양보다 많이 사용되지 않도록 사용자가 보증할 수 있도록 하기 위해 제공됩니다.

MAXFILESIZE *pages*

각 이벤트 모니터 파일의 크기에 제한이 있음을 지정합니다. 이벤트 모니터는 새 이벤트 레코드를 파일에 기록할 때마다, 파일이 *Pages*(4K

페이지 단위)보다 *크지* 않은지 확인합니다. 결과 파일이 너무 크면, 이벤트 모니터는 다음 파일로 전환합니다. 이 옵션의 디폴트값은 다음과 같습니다.

- Windows - 200 4K 페이지
- UNIX - 1000 4K 페이지

페이지 수는 최소한 이벤트 버퍼의 *크기*(페이지 단위)보다 많아야 합니다. 이 요건이 만족되지 않으면, 오류가 발생합니다(SQLSTATE 428A4).

MAXFILESIZE NONE

파일 *크기*에 제한이 없음을 지정합니다. MAXFILESIZE NONE을 지정할 경우, MAXFILES 1도 함께 지정해야 합니다. 이 옵션은 한 파일에 특정 이벤트 모니터에 대한 모든 이벤트 데이터가 포함된다는 것을 의미합니다. 이 경우 이벤트 파일은 00000000.evt입니다.

BUFFERSIZE *pages*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 단위). 모든 이벤트 모니터 파일 I/O는 이벤트 모니터의 성능이 향상되도록 버퍼화됩니다. 버퍼가 *크면* 클수록, 이벤트 모니터에 의해 수행되는 I/O는 적어집니다. 활발하게 활동 중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 가지고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두 배의 버퍼링을 사용합니다.

각 버퍼의 디폴트 *크기*는 4페이지입니다(두 개의 16K 버퍼가 할당됨). 최소 *크기*는 1페이지입니다. 버퍼가 힙으로부터 할당되므로 버퍼의 최대 *크기*는 모니터 힙의 *크기*뿐 아니라 MAXFILESIZE 매개변수 값으로 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 *크기*를 늘리십시오.

해당 데이터를 파이프에 기록하는 이벤트 모니터도 *크기*가 각각 2 페이지인 (구성 가능하지 않은) 버퍼를 갖고 있습니다. 이 버퍼들은 또한 모니터 힙(MON_HEAP)으로부터 할당됩니다. 파이프 목표를 갖는 각 활성 이벤트 모니터에 대해, 데이터베이스 힙의 *크기*를 2페이지씩 증가시키십시오.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

NONBLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 기다리지 않도록 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤트 모니터만큼 데이터베이스 작업의 속도를 저하시키지는 않습니다. 그러나 NONBLOCKED 이벤트 모니터는 활발히 사용 중인 시스템에서 데이터가 유실될 수 있습니다.

APPEND

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 새 이벤트 데이터를 데이터 파일의 기존 스트림에 추가하도록 지정합니다. 이벤트 모니터가 다시 활성화될 경우, 작동이 중지되지 않았던 것처럼 이벤트 파일에 다시 기록하기 시작합니다. APPEND는 디폴트 옵션입니다.

새로 작성된 이벤트 모니터가 이벤트 데이터를 기록할 디렉토리에 기존의 이벤트 데이터가 있을 경우, APPEND 옵션은 CREATE EVENT MONITOR 수행시 적용되지 않습니다.

REPLACE

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 모든 이벤트 파일을 지우고 00000000.evt 파일에 데이터를 기록하기 시작하도록 지정합니다.

MANUALSTART

이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화해야 한다는 것을 지정합니다. MANUALSTART 이벤트 모니터가 활성화된 다음, SET EVENT MONITOR STATE문을 사용하거나 인스턴스를 중지해야 비활성화될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 제외한 비WLM 이벤트 모니터의 디폴트 동작입니다.

AUTOSTART

이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화될 때마다 이벤트 모니터가 자동으로 활성화되도록 지정합니다. WLM 이벤트 모니터 및 DB2DETAILDEADLOCK 이벤트 모니터의 디폴트 동작입니다.

ON DBPARTITIONNUM *db-partition-number*

파일이나 파이프 이벤트 모니터를 실행할 데이터베이스 파티션을 지정합니다. 모니터링 범위가 LOCAL로 정의된 경우, 데이터는 지정된 파티션에서만 수집됩니다. 모니터 범위가 GLOBAL로 정의된 경우, 모든 데이터베이스 파티션이 데이터를 수집하고 지정된 번호로 데이터베이스 파티션에 보고합니다. I/O 구성요소는 지정된 데이터베이스 파티션에서 실제로 실행되며 지정된 파일 또는 파이프에 레코드를 씁니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다. 파티션된 데이터베이스 환경의 경우, 테이블에 기록 이벤트 모니터는 목표 테이블에 대한 테이블 스페이스가 정의된 모든 데이터베이스 파티션에서 이벤트를 실행하고 기록합니다.

GLOBAL절이 지정되지 않으면 응용프로그램에 대해 현재 연결된 데이터베이스 파티션 번호가 사용됩니다.

LOCAL

이벤트 모니터를 실행 중인 데이터베이스 파티션에만 보고합니다. 데이터베이스 활동에 대한 부분적 추적을 제공합니다. 이는 디폴트값입니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다.

GLOBAL

이벤트 모니터를 모든 데이터베이스 파티션에 보고합니다. 파티션된 데이터베이스의 경우, DEADLOCKS 이벤트 모니터만이 GLOBAL로 정의될 수 있습니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다.

규칙

- 이벤트 유형(DATABASE, TABLES, DEADLOCK,...) 각각은 특정 이벤트 모니터 정의에 한 번만 지정될 수 있습니다.

주

- 이벤트 모니터 정의는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 기록됩니다. 이벤트 자체는 SYSCAT.EVENTS 카탈로그 뷰에 기록됩니다. 목표 테이블의 이름은 SYSCAT.EVENTTABLES 카탈로그 뷰에 기록됩니다.
- DEADLOCKS보다 DEADLOCKS WITH DETAILS를 사용할 때 성능이 저하될 수 있습니다. 교착 상태가 발생하면 데이터베이스 관리 프로그램이 교착 상태에 대한 추가 정보를 기록하기 위해서는 추가 시간이 필요하기 때문입니다.
- CONNHEADER 이벤트는 일반적으로 연결이 설정될 때마다 기록됩니다. 그러나 DEADLOCKS WITH DETAILS에 대해서만 이벤트 모니터가 작성되었을 경우에는 연결이 처음 교착 상태로 들어갈 때만 CONNHEADER 이벤트가 기록됩니다.
- 여러 개의 데이터베이스 파티션이 있는 데이터베이스에서 ON DBPARTITIONNUM 절은 이벤트 모니터가 있어야 하는 위치를 나타내기 위해 DEADLOCKS 이벤트 유형이 있는 FILE 및 PIPE 이벤트 모니터와 함께 사용될 수 있습니다. 관련성이 있는 다른 데이터베이스 파티션의 정보가 해당 위치에 전달되어 처리됩니다.
- 여러 개의 데이터베이스 파티션이 있는 데이터베이스에서 교착 상태 이벤트 모니터는 참여 잠금이 존재했던 모든 데이터베이스 파티션으로부터 교착 상태에 참여하는 잠금이 있는 응용프로그램에 대한 정보를 수신합니다. 응용프로그램이 연결되는 데이터베이스 파티션(응용프로그램 코디네이터 파티션)이 참여 데이터베이스 파티션 중 하나가 아닌 경우 해당 파티션에서는 교착 상태 이벤트에 대한 정보가 수신되지 않습니다.

- BUFFERSIZE 매개변수는 STMT, STMT_HISTORY, DATA_VALUE 및 DETAILED_DLCONN 이벤트의 *크기*를 제한합니다. STMT 또는 STMT_HISTORY 이벤트의 *크기*가 버퍼에 맞지 않으면 명령문 텍스트를 절단하여 절단됩니다. DETAILED_DLCONN 이벤트의 *크기*가 버퍼에 맞지 않으면 잠금을 제거하여 해당 이벤트가 절단됩니다. 그래도 맞지 않으면 명령문 텍스트를 절단합니다. DATA_VAL 이벤트의 *크기*가 버퍼에 맞지 않으면 데이터 값이 절단됩니다.

  이벤트 모니터 WITH DETAILS HISTORY VALUES(및 보다 작은 범위의 WITH DETAILS HISTORY)는 상당한 양의 모니터 힙 스페이스를 사용하여 명령문과 해당 데이터 값을 추적합니다. 자세한 정보는 **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 설명을 참조하십시오.
- 이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화되지 않은 경우, 그 다음 데이터베이스 파티션이 활성화되면 이벤트 모니터가 활성화됩니다.
- 이벤트 모니터는 일단 활성화되면 이벤트 모니터가 명시적으로 비활성화되거나 인스턴스 처리가 되풀이될 때까지 자동 시작 이벤트 모니터처럼 작동합니다. 즉 데이터베이스 파티션이 비활성화되었을 때 이벤트 모니터가 활성화되고 그후에 그 데이터베이스 파티션이 다시 활성화되면, 이벤트 모니터도 명시적으로 재활성화됩니다.
- *테이블 이벤트 모니터에 기록:*: 일반 메모:
  - 모든 목표 테이블은 CREATE EVENT MONITOR문을 실행할 때 작성됩니다.
  - 어떤 이유로든 테이블 작성에 실패하면 응용프로그램에 오류가 전달되고 CREATE EVENT MONITOR문이 실패합니다.
  - 목표 테이블은 한 개의 이벤트 모니터만 사용할 수 있습니다. CREATE EVENT MONITOR를 처리하는 동안 다른 이벤트 모니터에서 사용하도록 목표 테이블이 이미 정의되어 있음이 발견되면 CREATE EVENT MONITOR문이 실패하고 응용프로그램에 오류가 전달됩니다. 테이블 이름이 SYSCAT.EVENTTABLES 카탈로그 뷰에 있는 값과 일치하면 다른 이벤트 모니터가 테이블을 사용하도록 테이블이 정의됩니다.
  - CREATE EVENT MONITOR를 처리하는 동안 테이블은 있지만 다른 이벤트 모니터에서 사용하도록 테이블이 정의되어 있지 *않으면* 테이블이 작성되지 않고 처리가 계속됩니다. 응용프로그램에 경고가 전달됩니다.
  - 모든 테이블 스페이스는 CREATE EVENT MONITOR문이 실행되기 전에 존재해야 합니다. CREATE EVENT MONITOR문은 테이블 스페이스를 작성하지 않습니다.
  - LOCAL 키워드와 GLOBAL 키워드를 지정할 경우 무시됩니다. WRITE TO TABLE 이벤트 모니터의 경우에는 이벤트 모니터 출력 프로세스나 스레드가 이 인스턴스의 각 데이터베이스 파티션에서 시작되고 이러한 각 프로세스는 프로세스가 실행되는 데이터베이스 파티션에 대해서만 데이터를 보고합니다.

– 테이블에 기록 이벤트 모니터는 다음과 같은 플랫 모니터 로그 파일이나 파이프 형식의 이벤트 유형은 기록하지 않습니다.
  - LOG_STREAM_HEADER
  - LOG_HEADER
  - DB_HEADER(db_name 및 db_path 요소는 기록되지 않습니다. conn_time 요소는 CONTROL에 기록됩니다.)
– 파티션된 데이터베이스 환경의 경우, 데이터는 목표 테이블의 테이블 스페이스가 존재하는 데이터베이스 파티션에서만 목표 테이블에 기록됩니다. 일부 데이터베이스 파티션에 목표 테이블에 대한 테이블 스페이스가 존재하지 않을 경우, 해당 목표 테이블에 대한 데이터가 무시됩니다. 이 때문에 사용자는 특정 데이터베이스 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터링할 데이터베이스 파티션의 서브세트를 선택할 수 있습니다.

  파티션된 데이터베이스 환경의 경우, 일부 목표 테이블이 데이터베이스 파티션에 상주하지 않지만 다른 목표 테이블이 동일한 데이터베이스 파티션에 상주하면 데이터베이스 파티션에 있는 목표 테이블에 대한 데이터만 기록됩니다.
– 사용자는 모든 목표 테이블을 직접 프룬해야 합니다.

테이블 컬럼:

– 테이블의 컬럼 이름은 이벤트 모니터의 요소 ID와 일치합니다. sqlm_time(경과 시간) 유형의 모니터 변수는 예외입니다. 이러한 유형에 대한 컬럼 이름은 TYPE_NAME_S와 TYPE_NAME_MS이고, 이 두 이름은 각각 초와 마이크로초로 시간을 저장하는 컬럼을 나타냅니다. 일치하는 목표 테이블 컬럼이 없는 이벤트 모니터 요소는 무시됩니다.
– 그룹에 대한 전체 요소 목록을 포함하는 CREATE EVENT MONITOR 명령을 빌드하려면 db2evtbl 명령을 사용하십시오.
– 모니터 요소에 사용되는 컬럼 유형은 다음 맵핑과 상호 관련이 있습니다.

```
SQLM_TYPE_STRING                    CHAR[n], VARCHAR[n] or CLOB(n)
                                    (이벤트 모니터 레코드의 데이터가
                          n 바이트를 초과할 경우,
                                     데이터는 절단됩니다.)
SQLM_TYPE_U8BIT 및 SQLM_TYPE_8BIT   SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_16BIT 및 SQLM_TYPE_U16BIT SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_32BIT 및 SQLM_TYPE_U32BIT INTEGER 또는 BIGINT
SQLM_TYPE_U64BIT 및 SQLM_TYPE_64BIT  BIGINT
sqlm_timestamp                      TIMESTAMP
sqlm_time(경과 시간)                BIGINT
sqlca:
    sqlerrmc                          VARCHAR[72]
    sqlstate                          CHAR[5]
    sqlwarn                           CHAR[11]
    기타 파일                         INTEGER 또는 BIGINT
```

– 컬럼은 NOT NULL로 정의됩니다.

– CLOB 컬럼이 있는 테이블은 VARCHAR 컬럼이 있는 테이블에 비해 성능이 떨어지므로 STMT *evm-group* 값(또는 DEADLOCKS WITH DETAILS 이벤트 유형을 사용할 경우에는 DLCONN *evm-group* 값)을 지정할 때 TRUNC 키워드를 사용하는 것이 좋습니다.

– 다른 목표 테이블과 달리 CONTROL 테이블의 컬럼은 모니터 요소 ID와 일치하지 않습니다. 컬럼은 다음과 같이 정의됩니다.

```
컬럼 이름         데이터 유형   널(NULL)  설명
                                입력 가능
------------      ---------     --------  -----------
PARTITION_KEY     INTEGER       N         분산 키
                                          (파티션된 데이터베이스만 해당)
PARTITION_NUMBER  INTEGER       N         데이터베이스 파티션 번호
                                          (파티션된 데이터베이스만 해당)
EVMONNAME         VARCHAR(128)  N         이벤트 모니터 이름
MESSAGE           VARCHAR(128)  N         MESSAGE_TIME 컬럼의
                                          등록 정보를 기술합니다.
                                          다음 중 하나가
                                          될 수 있습니다.
                                          - FIRST_CONNECT(데이터베이스
                                          활성화 이후 처음
                                          데이터베이스에 연결한 시간)
                                          - EVMON_START(EVMONNAME에
      나열된 이벤트 모니터가
                                          시작된 시간)
                                          - OVERFLOWS:n(버퍼 오버플로우로 인해
                          n개의 레코드를
                  버린다는 것을 나타냄)
                                          - LAST_DROPPED_RECORD
                                            (오버플로우가 발생된
        마지막 시간)
MESSAGE_TIME    TIMESTAMP       N         시간소인
```

– 파티션된 데이터베이스 환경에서 각 테이블의 첫 번째 컬럼의 이름은 PARTITION_KEY이고, NOT NULL이며 INTEGER 유형입니다. 이 컬럼은 테이블의 분산 키로 사용됩니다. 각 이벤트 모니터 프로세스에서 프로세스가 실행되는 데이터베이스 파티션에 데이터를 삽입할 수 있도록 이 컬럼의 값이 선택됩니다. 즉, 삽입 조작은 이벤트 모니터 프로세스가 실행되는 데이터베이스 파티션에서 로컬로 수행됩니다. 모든 데이터베이스 파티션에서 PARTITION_KEY 필드는 같은 값을 포함합니다. 즉 데이터베이스 파티션이 삭제되고 데이터 재분배가 수행될 경우 삭제된 데이터베이스 파티션에 있는 모든 데이터는 고르게 분배되지 않고 다른 데이터베이스 파티션으로 이동됩니다. 따라서 데이터베이스 파티션을 제거하기 전에 제거할 데이터베이스 파티션에 있는 모든 테이블 행을 삭제하는 것이 좋습니다.

– 파티션된 데이터베이스 환경에서는 각 테이블에 대해 PARTITION_NUMBER 컬럼을 정의할 수 있습니다. 이 컬럼은 INTEGER 유형인 NOT NULL입니다. 이 컬럼에는 데이터가 삽입된 데이터베이스 파티션의 번호가 포함합니다. PARTITION_KEY 컬럼과 달리 PARTITION_NUMBER 컬럼은 필수가 아닙니다. 파티션되지 않은 데이터베이스 환경에서는 PARTITION_NUMBER 컬럼을 정의할 수 없습니다.

테이블 속성

– 디폴트 테이블 속성이 사용됩니다. 테이블을 작성할 때 분산 키(파티션된 데이터베이스에만 해당)를 제외하고 추가 옵션은 지정하지 않습니다.

– 테이블에 대한 인덱스를 작성할 수 있습니다.
– VOLATILE, RI, 트리거 및 제한조건 등과 같은 추가 테이블 속성을 추가할 수는 있지만 이벤트 모니터 프로세스(또는 스레드)는 이런 속성을 무시합니다.
– ″not logged initially″를 테이블 속성으로 추가하면 처음 COMMIT를 실행할 때 이 속성이 해제되고 다시 설정되지 않습니다.

이벤트 모니터 활성화

– 이벤트 모니터가 활성화되면 SYSCAT.EVENTTABLES 카탈로그 뷰에서 모든 목표 테이블 이름이 검색됩니다.
– 파티션된 데이터베이스 환경에서는 인스턴스의 모든 데이터베이스 파티션에서 활성화 처리가 이루어집니다. 특정 데이터베이스 파티션에서는 활성화 처리가 각 목표 테이블의 테이블 스페이스 및 데이터베이스 파티션 그룹을 판별합니다. 이벤트 모니터는 데이터베이스 파티션에 최소한 하나의 목표 테이블이 존재할 경우에만 데이터베이스 파티션에 활성화됩니다. 또한 데이터베이스 파티션에 일부 목표 테이블이 없을 경우, 해당 테이블에 대한 데이터가 런타임 처리시 삭제되도록 해당 목표 테이블에 대한 플래그가 표시됩니다.
– 이벤트 모니터가 활성화될 때 목표 테이블이 없거나, 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주하지 않을 경우에도 활성화가 계속됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 테이블이 있거나 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주할 경우 이 테이블에 삽입될 데이터는 무시됩니다.
– 활성화 프로세스는 각 목표 테이블의 유효성을 확인합니다. 유효성 확인이 실패하면 이벤트 모니터가 활성화되지 않고 관리 로그에 메시지가 기록됩니다.
– 파티션된 데이터베이스 환경에서 활성화하는 동안에는 FIRST_CONNECT와 EVMON_START에 대한 CONTROL 테이블의 행만 카탈로그 데이터베이스 파티션에 삽입됩니다. 이 때 제어 테이블에 대한 테이블 스페이스가 카탈로그 데이터베이스 파티션에 있어야 합니다. 테이블 스페이스가 카탈로그 데이터베이스 파티션에 없으면 이런 삽입 조작이 수행되지 않습니다.
– 파티션된 데이터베이스 환경에서는 테이블에 기록 이벤트 모니터가 활성화될 때 파티션이 활성화되어 있지 않으면, 다음에 파티션이 활성화될 때 이벤트 모니터가 활성화됩니다.

런타임

– 이벤트 모니터가 DATAACCESS 권한으로 실행됩니다.
– 이벤트 모니터가 활성화되어 있는 동안 목표 테이블에 대한 삽입 조작이 실패하면 다음 작업이 수행됩니다.
  - 언커미트된 변경사항을 롤백합니다.
  - 관리 로그에 메시지가 기록됩니다.
  - 이벤트 모니터가 비활성화됩니다.

  - 이벤트 모니터가 활성화되어 있을 경우 이벤트 모니터 버퍼에 대한 처리가 완료되면 로컬 COMMIT가 수행됩니다.
  - 파티션된 데이터베이스 환경에서는 65,535바이트까지 가능한 실제 명령문 텍스트는 응용프로그램 코디네이터 데이터베이스 파티션에서 실행되는 이벤트 모니터에 의해 STMT 테이블이나 DLCONN 테이블에 저장됩니다. 다른 데이터베이스 파티션에서는 이 값이 0입니다.
  - 파티션되지 않은 데이터베이스 환경에서는 마지막 응용프로그램이 종료되고 데이터베이스가 명시적으로 활성화되지 않았을 때 모든 테이블에 기록 이벤트 모니터가 비활성화됩니다. 파티션된 데이터베이스 환경에서는 카탈로그 파티션이 비활성화될 때 테이블에 기록 이벤트 모니터가 비활성화됩니다.
  - DROP EVENT MONITOR문은 목표 테이블을 삭제하지 않습니다.
  - 테이블에 기록 이벤트 모니터가 활성화될 때마다, 이벤트 모니터가 사용 중인 동안 수정되지 않도록 IN 테이블이 각 목표 테이블을 잠급니다. 이벤트 모니터가 사용 중인 동안 테이블 잠금은 모든 테이블에서 유지됩니다. 독점 액세스가 목표 테이블에 필요하면(예를 들어, 유틸리티를 실행하려고 할 때) 이러한 액세스를 시도하기 전에 먼저 이벤트 모니터를 비활성화하여 테이블 잠금을 해제합니다.
- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - 쉼표는 *target-table-info* 절에서 여러 옵션을 구분할 때 사용됩니다.

예:

예 1: SMITHPAY라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 데이터베이스에 대해, 그리고 JSMITH 권한 부여 ID가 소유하는 PAYROLL 응용프로그램에서 수행되는 SQL문에 대해 이벤트 데이터를 수집합니다. 데이터는 절대 경로 /home/jsmith/event/smithpay/에 추가됩니다. 최대 25개의 파일이 작성됩니다. 각 파일은 최대 1 024 4K 페이지가 됩니다. 파일 I/O는 비블록화됩니다.

```
CREATE EVENT MONITOR SMITHPAY
  FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
  WRITE TO FILE '/home/jsmith/event/smithpay'
  MAXFILES 25
  MAXFILESIZE 1024
  NONBLOCKED
  APPEND
```

예 2: DEADLOCKS_EVTS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 교착 상태 이벤트를 수집한 다음 이를 상대 경로 DLOCKS에 기록합니다. 한 개의 파일이 작성되며, 최대 파일 크기는 없습니다. 이벤트 모니터가 활성화될 때마다, 이벤트 데이터를 00000000.evt 파일(있는 경우)에 추가합니다. 이벤트 모니터는 데이터베이스가 시작될 때마다 시작됩니다. I/0는 디폴트로 블록화됩니다.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
  FOR DEADLOCKS
       WRITE TO FILE 'DLOCKS'
  MAXFILES 1
  MAXFILESIZE NONE
  AUTOSTART
```

예 3: DB_APPLS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 연결 이벤트를 수집하고 데이터를 Named Pipe /home/jsmith/applpipe에 기록합니다.

```
CREATE EVENT MONITOR DB_APPLS
  FOR CONNECTIONS
  WRITE TO PIPE '/home/jsmith/applpipe'
```

예 4: 파티션된 데이터베이스 환경을 가정하고 FOO라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트를 수집하고 수집한 이벤트를 다음 파생 이름을 사용하여 SQL 테이블에 기록합니다.

- CONNHEADER_FOO
- STMT_FOO
- SUBSECTION_FOO
- CONTROL_FOO

테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 모든 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 모든 테이블에는 테이블 그룹에 대한 모든 요소가 포함됩니다. 즉, 요소 이름과 같은 이름을 갖는 컬럼이 정의됩니다.

```
CREATE EVENT MONITOR FOO
  FOR STATEMENTS
  WRITE TO TABLE
```

예 5: 파티션된 데이터베이스 환경을 가정하고 BAR라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트와 트랜잭션 이벤트를 수집하고 수집한 이벤트를 다음과 같이 테이블에 기록합니다.

- STMT 그룹의 데이터는 모두 MYDEPT.MYSTMTINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. ROWS_READ, ROWS_WRITTEN 및 STMT_TEXT 요소에 대해서만 컬럼을 작성하고, 그룹의 다른 요소는 버립니다.
- SUBSECTION 그룹의 데이터는 모두 MYDEPT.MYSUBSECTIONINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. 이 테이블에는 START_TIME, STOP_TIME 및 PARTIAL_RECORD를 제외한 모든 컬럼이 포함됩니다.

- XACT 그룹의 데이터는 모두 XACT_BAR 테이블에 기록됩니다. 테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 이 테이블에는 XACT 그룹에 있는 모든 요소가 포함됩니다.
- connheader나 control에 대해서는 테이블이 작성되지 않습니다. 따라서 이 두 그룹에 대한 데이터는 모두 버립니다.

```
CREATE EVENT MONITOR BAR
  FOR STATEMENTS, TRANSACTIONS
  WRITE TO TABLE
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
  XACT
```

CREATE EVENT MONITOR(활동)

CREATE EVENT MONITOR(활동) 명령문은 데이터베이스를 사용할 때 발생하는 활동 이벤트를 기록할 모니터를 정의합니다. 활동 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.

호출

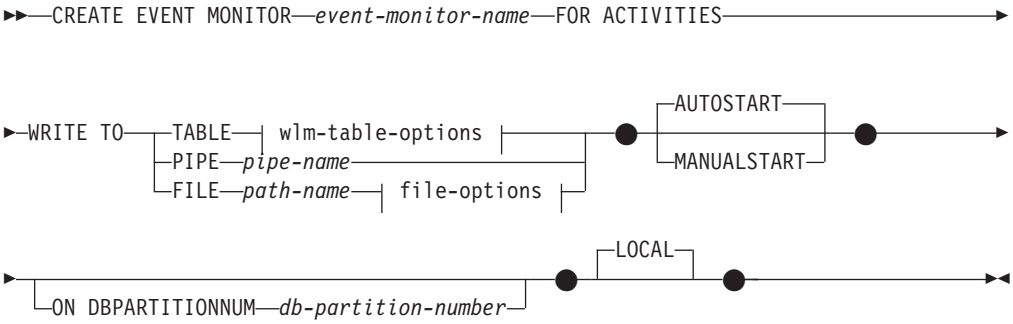
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

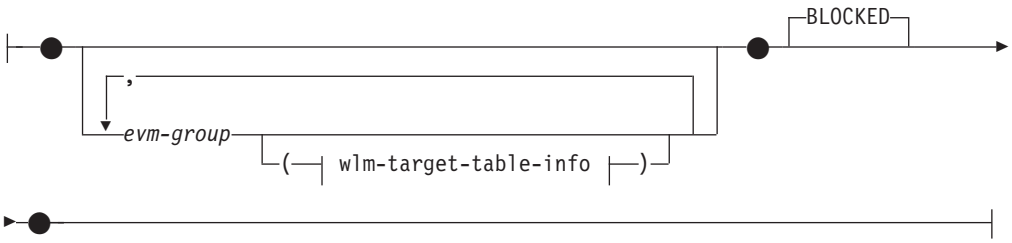
명령문의 권한 부여 ID는 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- DBADM 권한
- SQLADM 권한
- WLMADM 권한

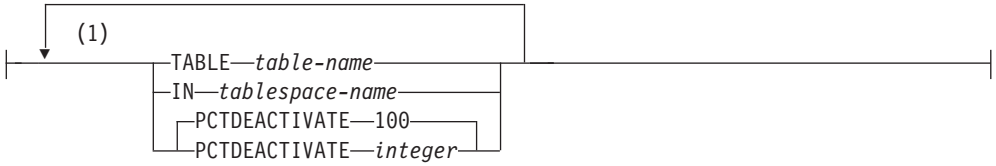
구문



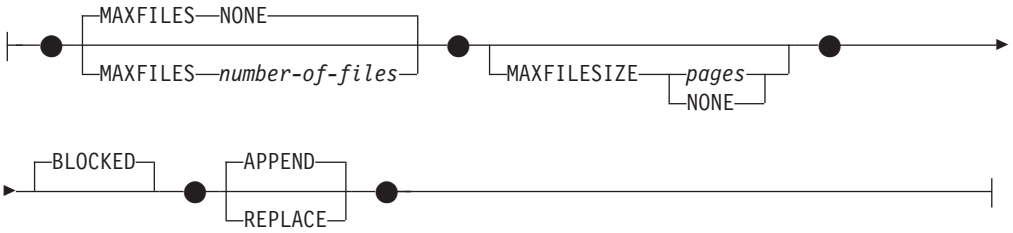
wlm-table-options:



wlm-target-table-info:



file-options:



주:

1 각 절을 한 번만 지정할 수 있습니다.

설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *event-monitor-name*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 레코드의 유형을 소개합니다.

ACTIVITIES

이벤트가 WLM_CAPTURE_ACTIVITY_IN_PROGRESS 프로시저에서 트리거되면, 활동이 실행을 완료하거나 실행 완료 전에 이벤트 모니터가 활동을 기록하도록 지정합니다. 활동은 다음과 같아야 합니다.

- COLLECT ACTIVITY DATA 세트가 있는 서비스 클래스나 워크로드에 속합니다.
- 연관된 작업 조치가 COLLECT ACTIVITY DATA인 작업 클래스에 속합니다.
- COLLECT ACTIVITY DATA절이 지정된 임계값을 위반한 활동으로 식별됩니다.
- 완료 전에 호출에서 WLM_CAPTURE_ACTIVITY_IN_PROGRESS 프로시저로 식별됩니다.

WRITE TO

데이터에 대한 목표를 사용합니다.

TABLE

이벤트 모니터 데이터의 목표가 데이터베이스 테이블 세트임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 하나 이상의 논리 데이터 그룹에 나눈 다음 각 그룹을 별도의 테이블에 삽입합니다. 목표 테이블이 있는 그룹의 데이터는 보존되고 목표 테이블이 없는 그룹의 데이터는 버립니다. 그룹에 포함된 각 모니터 요소는 같은 이름으로 된 테이블 컬럼에 맵핑됩니다. 해당하는 테이블 컬럼이 있는 요소만 테이블에 삽입되고, 다른 요소들은 버립니다.

wlm-table-options

논리 데이터 그룹에 대한 목표 테이블을 정의합니다. 이 절은 기록할 각 그룹화에 대하여 지정해야 합니다. 그러나 evm-group-info절을 지정하지 않으면 해당 이벤트 모니터 유형에 대한 모든 그룹이 기록됩니다.

evm-group

목표 테이블을 정의할 논리 데이터 그룹을 식별합니다. 이 값은 다음 표에서와 같이 이벤트 모니터의 유형에 따라 다릅니다.

이벤트 모니터 유형	evm-group 값
활동	• ACTIVITY • ACTIVITYSTMT • ACTIVITYVALS • CONTROL

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

PIPE

이벤트 모니터 데이터의 목표가 Named Pipe임을 지정합니다. 이벤트 모니터는 데이터를 단일 스트림의 파이프(즉, 하나인 것처럼 보이는 무한정으로 긴 파일)에 기록합니다. 데이터를 파이프에 기록할 때, 이벤트 모니터는 블록화된 쓰기를 수행하지 않습니다. 파이프 버퍼에 스페이스가 없으면, 이벤트 모니터에서는 데이터를 버립니다. 데이터 유실이 없도록 하려면, 즉시 데이터를 읽도록 해야 합니다.

pipe-name

이벤트 모니터가 데이터를 기록할 파이프의 이름(AIX에서 FIFO)입니다.

파이프의 이름 지정 규칙은 플랫폼에 따라 다릅니다. UNIX 운영 체제의 경우, 파이프 이름은 파일 이름과 같이 취급됩니다. 그러므로 상대 파이프 이름이 허용되고, 상대 경로 이름과 같이 취급됩니다(아래의 *path-name* 참

조). 그러나 Windows에는 파이프 이름에 대한 특수한 구문이 있으며 그 결과로 절대 파이프 이름이 요구됩니다.

파이프의 존재 여부는 이벤트 모니터 작성시 점검되지 않습니다. 이벤트 모니터가 활성화될 때 읽을 파이프를 작성하고 여는 것은 모니터링 응용프로그램에서 수행됩니다. 이 때 파이프를 읽을 수 없으면, 이벤트 모니터 자체가 작동 중지되고 오류가 기록됩니다. 즉, 이벤트 모니터가 데이터베이스 시작 시 AUTOSTART 옵션에 의해 활성화된 경우, 이벤트 모니터는 시스템 오류 로그에 오류를 기록합니다. 이벤트 모니터가 SET EVENT MONITOR STATE SQL문으로 활성화된 경우, 해당 명령문은 실패합니다(SQLSTATE 58030).

FILE

이벤트 모니터 데이터의 목표가 파일(또는 파일 세트)임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 확장자가 “evt”인 일련의 8문자 순서화 파일로 작성합니다 (예: 00000000.evt, 00000001.evt 및 00000002.evt). 데이터는 더 작은 여러 조각으로 분리되더라도 하나의 논리 파일로 간주됩니다. 즉, 데이터 스트림의 시작은 00000000.evt 파일의 첫 번째 바이트이고, 데이터 스트림의 끝은 nnnnnnnn.evt 파일의 마지막 바이트입니다.

각 파일의 최대 *크기*뿐 아니라 최대 파일 수도 정의할 수 있습니다. 이벤트 모니터는 단일 이벤트 레코드를 두 파일에 걸쳐 분리시킬 수 없습니다. 그러나 이벤트 모니터는 관련 레코드들을 두 개의 다른 파일로 작성할 수 있습니다. 이벤트 파일을 처리할 때 그러한 관련 정보를 추적하는 것은 이 데이터를 사용하는 응용프로그램에서 수행됩니다.

path-name

이벤트 모니터가 이벤트 파일 데이터를 기록해야 하는 디렉토리의 이름입니다. 경로는 서버에 알려져 있어야 하지만 경로 그 자체는 다른 파티션에 상주할 수 있습니다(예: UNIX 시스템에서는 NFS 마운트 파일이 될 수 있음). *path-name*을 지정할 때 문자열 상수를 사용해야 합니다.

CREATE EVENT MONITOR를 실행할 때 디렉토리는 없어도 됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 경로의 존재 여부에 대한 점검이 수행됩니다. 이 때 목표 경로가 존재하지 않으면 오류가 발생합니다 (SQLSTATE 428A3).

AIX에서는 루트 디렉토리로 시작하고 Windows에서는 디스크 ID로 시작하는 절대 경로가 지정되는 경우, 지정된 경로가 사용되는 경로가 됩니다. 상대 경로(루트로 시작하지 않는 경로)를 지정할 경우, 데이터베이스 디렉토리에 있는 DB2EVENT 디렉토리에 대한 상대 경로가 사용됩니다.

상대 경로를 지정하면, DB2EVENT 디렉토리를 사용하여 그 경로를 절대 경로로 변환할 수 있습니다. 그러면, 절대 및 상대 경로 사이에 어떤 구별도 만들어지지 않습니다. 절대 경로는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 저장됩니다.

같은 목표 경로를 갖는 두 개 이상의 이벤트 모니터를 지정할 수 있습니다. 그러나 처음에 이벤트 모니터 중 하나가 활성화되고 대상 디렉토리가 비어 있지 않으면, 다른 이벤트 모니터 중 어떤 것도 활성화시킬 수 없습니다.

file-options

파일 형식에 대한 옵션을 지정합니다.

MAXFILES NONE

이벤트 모니터가 작성하는 이벤트 파일 수에 제한이 없음을 지정합니다. 이는 디폴트입니다.

MAXFILES *number-of-files*

언제든지 특정 이벤트 모니터에 대해 존재할 이벤트 모니터 파일 수에 제한이 있음을 지정합니다. 이벤트 모니터는 다른 파일을 작성해야 할 때마다, 디렉토리에 있는 .evt 파일의 수가 *number-of-files*보다 적은지 확인합니다. 이미 이 한계에 도달해 있으면, 이벤트 모니터는 스스로 작동 중지됩니다.

응용프로그램이 이벤트 파일이 작성된 후에 디렉토리에서 파일을 제거할 경우, 이벤트 모니터가 작성할 수 있는 총 파일 수는 *number-of-files*를 초과할 수 있습니다. 이 옵션은 이벤트 데이터가 지정된 디스크 스페이스의 양보다 많이 사용되지 않도록 사용자가 보증할 수 있도록 하기 위해 제공됩니다.

MAXFILESIZE *pages*

각 이벤트 모니터 파일의 크기에 제한이 있음을 지정합니다. 이벤트 모니터는 새 이벤트 레코드를 파일에 기록할 때마다, 파일이 *Pages*(4K 페이지 단위)보다 크지 않은지 확인합니다. 결과 파일이 너무 크면, 이벤트 모니터는 다음 파일로 전환합니다. 이 옵션의 디폴트값은 다음과 같습니다.

- Windows - 200 4K 페이지
- UNIX - 1000 4K 페이지

페이지 수는 최소한 이벤트 버퍼의 크기(페이지 단위)보다 많아야 합니다. 이 요건이 만족되지 않으면, 오류가 발생합니다(SQLSTATE 428A4).

MAXFILESIZE NONE

파일 크기에 제한이 없음을 지정합니다. MAXFILESIZE NONE을 지

정할 경우, MAXFILES 1도 함께 지정해야 합니다. 이 옵션은 한 파일에 특정 이벤트 모니터에 대한 모든 이벤트 데이터가 포함된다는 것을 의미합니다. 이 경우 이벤트 파일은 00000000.evt입니다.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

APPEND

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 새 이벤트 데이터를 데이터 파일의 기존 스트림에 추가하도록 지정합니다. 이벤트 모니터가 다시 활성화될 경우, 작동이 중지되지 않았던 것처럼 이벤트 파일에 다시 기록하기 시작합니다. APPEND는 디폴트 옵션입니다.

새로 작성된 이벤트 모니터가 이벤트 데이터를 기록할 디렉토리에 기존의 이벤트 데이터가 있을 경우, APPEND 옵션은 CREATE EVENT MONITOR 수행시 적용되지 않습니다.

REPLACE

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 모든 이벤트 파일을 지우고 00000000.evt 파일에 데이터를 기록하기 시작하도록 지정합니다.

MANUALSTART

이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화해야 한다는 것을 지정합니다. MANUALSTART 이벤트 모니터가 활성화된 다음, SET EVENT MONITOR STATE문을 사용하거나 인스턴스를 중지해야 비활성화될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 제외한 비WLM 이벤트 모니터의 디폴트 동작입니다.

AUTOSTART

이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화될 때마다 이벤트 모니터가 자동으로 활성화되도록 지정합니다. WLM 이벤트 모니터 및 DB2DETAILDEADLOCK 이벤트 모니터의 디폴트 동작입니다.

ON DBPARTITIONNUM *db-partition-number*

파일이나 파이프 이벤트 모니터를 실행할 데이터베이스 파티션을 지정합니다. 모니터링 범위가 LOCAL로 정의된 경우, 데이터는 지정된 파티션에서만 수집됩니다. 모니터 범위가 GLOBAL로 정의된 경우, 모든 데이터베이스 파티션이 데이터를 수집하고 지정된 번호로 데이터베이스 파티션에 보고합니다. I/O 구성요소는 지정된 데이터베이스 파티션에서 실제로 실행되며 지정된 파일 또는 파이프에 레코드를 씁니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다. 파티션된 데이터베이스 환경의 경우, 테이블에 기록 이벤트 모니터는 목표 테이블에 대한 테이블 스페이스가 정의된 모든 데이터베이스 파티션에서 이벤트를 실행하고 기록합니다.

GLOBAL절이 지정되지 않으면 응용프로그램에 대해 현재 연결된 데이터베이스 파티션 번호가 사용됩니다.

LOCAL

이벤트 모니터를 실행 중인 데이터베이스 파티션에만 보고합니다. 데이터베이스 활동에 대한 부분적 추적을 제공합니다. 이는 디폴트값입니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다.

규칙

- ACTIVITIES 이벤트 유형은 특정 이벤트 모니터 정의에 있는 다른 이벤트 유형과 결합될 수 없습니다.

주

- 이벤트 모니터 정의는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 기록됩니다. 이벤트 자체는 SYSCAT.EVENTS 카탈로그 뷰에 기록됩니다. 목표 테이블의 이름은 SYSCAT.EVENTTABLES 카탈로그 뷰에 기록됩니다.
- 이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화되지 않은 경우, 그 다음 데이터베이스 파티션이 활성화되면 이벤트 모니터가 활성화됩니다.
- 이벤트 모니터는 일단 활성화되면 이벤트 모니터가 명시적으로 비활성화되거나 인스턴스 처리가 되풀이될 때까지 자동 시작 이벤트 모니터처럼 작동합니다. 즉 데이터베이스 파티션이 비활성화되었을 때 이벤트 모니터가 활성화되고 그후에 그 데이터베이스 파티션이 다시 활성화되면, 이벤트 모니터도 명시적으로 재활성화됩니다.
- *테이블 이벤트 모니터에 기록:*: 일반 메모:
  - 모든 목표 테이블은 CREATE EVENT MONITOR문을 실행할 때 작성됩니다.
  - 어떤 이유로든 테이블 작성에 실패하면 응용프로그램에 오류가 전달되고 CREATE EVENT MONITOR문이 실패합니다.
  - 목표 테이블은 한 개의 이벤트 모니터만 사용할 수 있습니다. CREATE EVENT MONITOR를 처리하는 동안 다른 이벤트 모니터에서 사용하도록 목표 테이블이 이미 정의되어 있음이 발견되면 CREATE EVENT MONITOR문이 실패하고 응용프로그램에 오류가 전달됩니다. 테이블 이름이 SYSCAT.EVENTTABLES 카탈로그 뷰에 있는 값과 일치하면 다른 이벤트 모니터가 테이블을 사용하도록 테이블이 정의됩니다.
  - CREATE EVENT MONITOR를 처리하는 동안 테이블은 있지만 다른 이벤트 모니터에서 사용하도록 테이블이 정의되어 있지 않으면 테이블이 작성되지 않고 처리가 계속됩니다. 응용프로그램에 경고가 전달됩니다.

– 모든 테이블 스페이스는 CREATE EVENT MONITOR문이 실행되기 전에 존재해야 합니다. CREATE EVENT MONITOR문은 테이블 스페이스를 작성하지 않습니다.
– LOCAL 키워드와 GLOBAL 키워드를 지정할 경우 무시됩니다. WRITE TO TABLE 이벤트 모니터의 경우에는 이벤트 모니터 출력 프로세스나 스레드가 이 인스턴스의 각 데이터베이스 파티션에서 시작되고 이러한 각 프로세스는 프로세스가 실행되는 데이터베이스 파티션에 대해서만 데이터를 보고합니다.
– 테이블에 기록 이벤트 모니터는 다음과 같은 플랫 모니터 로그 파일이나 파이프 형식의 이벤트 유형은 기록하지 않습니다.
  - LOG_STREAM_HEADER
  - LOG_HEADER
  - DB_HEADER(db_name 및 db_path 요소는 기록되지 않습니다. conn_time 요소는 CONTROL에 기록됩니다.)
– 파티션된 데이터베이스 환경의 경우, 데이터는 목표 테이블의 테이블 스페이스가 존재하는 데이터베이스 파티션에서만 목표 테이블에 기록됩니다. 일부 데이터베이스 파티션에 목표 테이블에 대한 테이블 스페이스가 존재하지 않을 경우, 해당 목표 테이블에 대한 데이터가 무시됩니다. 이 때문에 사용자는 특정 데이터베이스 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터링할 데이터베이스 파티션의 서브세트를 선택할 수 있습니다.

  파티션된 데이터베이스 환경의 경우, 일부 목표 테이블이 데이터베이스 파티션에 상주하지 않지만 다른 목표 테이블이 동일한 데이터베이스 파티션에 상주하면 데이터베이스 파티션에 있는 목표 테이블에 대한 데이터만 기록됩니다.
– 사용자는 모든 목표 테이블을 직접 프룬해야 합니다.

테이블 컬럼:

– 테이블의 컬럼 이름은 이벤트 모니터의 요소 ID와 일치합니다. sqlm_time(경과 시간) 유형의 모니터 변수는 예외입니다. 이러한 유형에 대한 컬럼 이름은 TYPE_NAME_S와 TYPE_NAME_MS이고, 이 두 이름은 각각 초와 마이크로초로 시간을 저장하는 컬럼을 나타냅니다. 일치하는 목표 테이블 컬럼이 없는 이벤트 모니터 요소는 무시됩니다.
– 그룹에 대한 전체 요소 목록을 포함하는 CREATE EVENT MONITOR 명령을 빌드하려면 db2evtbl 명령을 사용하십시오.
– 모니터 요소에 사용되는 컬럼 유형은 다음 맵핑과 상호 관련이 있습니다.

```
SQLM_TYPE_STRING                    CHAR[n], VARCHAR[n] or CLOB(n)
                                    (이벤트 모니터 레코드의 데이터가
                          n 바이트를 초과할 경우,
                                     데이터는 절단됩니다.)
SQLM_TYPE_U8BIT 및 SQLM_TYPE_8BIT   SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_16BIT 및 SQLM_TYPE_U16BIT SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_32BIT 및 SQLM_TYPE_U32BIT INTEGER 또는 BIGINT
SQLM_TYPE_U64BIT 및 SQLM_TYPE_64BIT  BIGINT
sqlm_timestamp                      TIMESTAMP
sqlm_time(경과 시간)                 BIGINT
```

```
sqlca:
    sqlerrmc                        VARCHAR[72]
    sqlstate                        CHAR[5]
    sqlwarn                         CHAR[11]
    기타 파일                       INTEGER 또는 BIGINT
```

- 컬럼은 NOT NULL로 정의됩니다.
- CLOB 컬럼이 있는 테이블은 VARCHAR 컬럼이 있는 테이블에 비해 성능이 떨어지므로 STMT *evm-group* 값(또는 DEADLOCKS WITH DETAILS 이벤트 유형을 사용할 경우에는 DLCONN *evm-group* 값)을 지정할 때 TRUNC 키워드를 사용하는 것이 좋습니다.
- 다른 목표 테이블과 달리 CONTROL 테이블의 컬럼은 모니터 요소 ID와 일치하지 않습니다. 컬럼은 다음과 같이 정의됩니다.

```
컬럼 이름         데이터 유형   널(NULL)  설명
                               입력 가능
-----------       ---------     --------  -----------
PARTITION_KEY     INTEGER       N         분산 키
                                          (파티션된 데이터베이스만 해당)
PARTITION_NUMBER  INTEGER       N         데이터베이스 파티션 번호
                                          (파티션된 데이터베이스만 해당)
EVMONNAME         VARCHAR(128)  N         이벤트 모니터 이름
MESSAGE           VARCHAR(128)  N         MESSAGE_TIME 컬럼의
                                          등록 정보를 기술합니다.
                                          다음 중 하나가
                                          될 수 있습니다.
                                          - FIRST_CONNECT(데이터베이스
                                          활성화 이후 처음
                                          데이터베이스에 연결한 시간)
                                          - EVMON_START(EVMONNAME에
      나열된 이벤트 모니터가
                                          시작된 시간)
                                          - OVERFLOWS:n
      (버퍼 오버플로우로 인해
                          n개의 레코드를
                 버린다는 것을나타냄)
                                          - LAST_DROPPED_RECORD
                                            (오버플로우가 발생된
        마지막 시간)
MESSAGE_TIME      TIMESTAMP     N         시간소인
```

- 파티션된 데이터베이스 환경에서 각 테이블의 첫 번째 컬럼의 이름은 PARTITION_KEY이고, NOT NULL이며 INTEGER 유형입니다. 이 컬럼은 테이블의 분산 키로 사용됩니다. 각 이벤트 모니터 프로세스에서 프로세스가 실행되는 데이터베이스 파티션에 데이터를 삽입할 수 있도록 이 컬럼의 값이 선택됩니다. 즉, 삽입 조작은 이벤트 모니터 프로세스가 실행되는 데이터베이스 파티션에서 로컬로 수행됩니다. 모든 데이터베이스 파티션에서 PARTITION_KEY 필드는 같은 값을 포함합니다. 즉 데이터베이스 파티션이 삭제되고 데이터 재분배가 수행될 경우 삭제된 데이터베이스 파티션에 있는 모든 데이터는 고르게 분배되지 않고 다른 데이터베이스 파티션으로 이동됩니다. 따라서 데이터베이스 파티션을 제거하기 전에 제거할 데이터베이스 파티션에 있는 모든 테이블 행을 삭제하는 것이 좋습니다.
- 파티션된 데이터베이스 환경에서는 각 테이블에 대해 PARTITION_NUMBER 컬럼을 정의할 수 있습니다. 이 컬럼은 INTEGER 유형인 NOT NULL입니다. 이 컬럼에는 데이터가 삽입된 데이터베이스 파티션의 번호가 포함합니다.

PARTITION_KEY 컬럼과 달리 PARTITION_NUMBER 컬럼은 필수가 아닙니다. 파티션되지 않은 데이터베이스 환경에서는 PARTITION_NUMBER 컬럼을 정의할 수 없습니다.

테이블 속성

- 디폴트 테이블 속성이 사용됩니다. 테이블을 작성할 때 분산 키(파티션된 데이터베이스에만 해당)를 제외하고 추가 옵션은 지정하지 않습니다.
- 테이블에 대한 인덱스를 작성할 수 있습니다.
- VOLATILE, RI, 트리거 및 제한조건 등과 같은 추가 테이블 속성을 추가할 수는 있지만 이벤트 모니터 프로세스(또는 스레드)는 이런 속성을 무시합니다.
- ″not logged initially″를 테이블 속성으로 추가하면 처음 COMMIT를 실행할 때 이 속성이 해제되고 다시 설정되지 않습니다.

이벤트 모니터 활성화

- 이벤트 모니터가 활성화되면 SYSCAT.EVENTTABLES 카탈로그 뷰에서 모든 목표 테이블 이름이 검색됩니다.
- 파티션된 데이터베이스 환경에서는 인스턴스의 모든 데이터베이스 파티션에서 활성화 처리가 이루어집니다. 특정 데이터베이스 파티션에서는 활성화 처리가 각 목표 테이블의 테이블 스페이스 및 데이터베이스 파티션 그룹을 판별합니다. 이벤트 모니터는 데이터베이스 파티션에 최소한 하나의 목표 테이블이 존재할 경우에만 데이터베이스 파티션에 활성화됩니다. 또한 데이터베이스 파티션에 일부 목표 테이블이 없을 경우, 해당 테이블에 대한 데이터가 런타임 처리시 삭제되도록 해당 목표 테이블에 대한 플래그가 표시됩니다.
- 이벤트 모니터가 활성화될 때 목표 테이블이 없거나, 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주하지 않을 경우에도 활성화가 계속됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 테이블이 있거나 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주할 경우 이 테이블에 삽입될 데이터는 무시됩니다.
- 활성화 프로세스는 각 목표 테이블의 유효성을 확인합니다. 유효성 확인이 실패하면 이벤트 모니터가 활성화되지 않고 관리 로그에 메시지가 기록됩니다.
- 파티션된 데이터베이스 환경에서 활성화하는 동안에는 FIRST_CONNECT와 EVMON_START에 대한 CONTROL 테이블의 행만 카탈로그 데이터베이스 파티션에 삽입됩니다. 이 때 제어 테이블에 대한 테이블 스페이스가 카탈로그 데이터베이스 파티션에 있어야 합니다. 테이블 스페이스가 카탈로그 데이터베이스 파티션에 없으면 이런 삽입 조작이 수행되지 않습니다.
- 파티션된 데이터베이스 환경에서는 테이블에 기록 이벤트 모니터가 활성화될 때 파티션이 활성화되어 있지 않으면, 다음에 파티션이 활성화될 때 이벤트 모니터가 활성화됩니다.

런타임

  - 이벤트 모니터가 DATAACCESS 권한으로 실행됩니다.
  - 이벤트 모니터가 활성화되어 있는 동안 목표 테이블에 대한 삽입 조작이 실패하면 다음 작업이 수행됩니다.
    - 언커미트된 변경사항을 롤백합니다.
    - 관리 로그에 메시지가 기록됩니다.
    - 이벤트 모니터가 비활성화됩니다.
  - 이벤트 모니터가 활성화되어 있을 경우 이벤트 모니터 버퍼에 대한 처리가 완료되면 로컬 COMMIT가 수행됩니다.
  - 파티션된 데이터베이스 환경에서는 65,535바이트까지 가능한 실제 명령문 텍스트는 응용프로그램 코디네이터 데이터베이스 파티션에서 실행되는 이벤트 모니터에 의해 STMT 테이블이나 DLCONN 테이블에 저장됩니다. 다른 데이터베이스 파티션에서는 이 값이 0입니다.
  - 파티션되지 않은 데이터베이스 환경에서는 마지막 응용프로그램이 종료되고 데이터베이스가 명시적으로 활성화되지 않았을 때 모든 테이블에 기록 이벤트 모니터가 비활성화됩니다. 파티션된 데이터베이스 환경에서는 카탈로그 파티션이 비활성화될 때 테이블에 기록 이벤트 모니터가 비활성화됩니다.
  - DROP EVENT MONITOR문은 목표 테이블을 삭제하지 않습니다.
  - 테이블에 기록 이벤트 모니터가 활성화될 때마다, 이벤트 모니터가 사용 중인 동안 수정되지 않도록 IN 테이블이 각 목표 테이블을 잠급니다. 이벤트 모니터가 사용 중인 동안 테이블 잠금은 모든 테이블에서 유지됩니다. 독점 액세스가 목표 테이블에 필요하면(예를 들어, 유틸리티를 실행하려고 할 때) 이러한 액세스를 시도하기 전에 먼저 이벤트 모니터를 비활성화하여 테이블 잠금을 해제합니다.
- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - 쉼표를 사용하여 *wlm-target-table-info*절에서 여러 옵션을 구분할 수 있습니다.

예

예 1: SMITHPAY라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 데이터베이스에 대해, 그리고 JSMITH 권한 부여 ID가 소유하는 PAYROLL 응용프로그램에서 수행되는 SQL문에 대해 이벤트 데이터를 수집합니다. 데이터는 절대 경로 /home/jsmith/event/smithpay/에 추가됩니다. 최대 25개의 파일이 작성됩니다. 각 파일은 최대 1 024 4K 페이지가 됩니다. 파일 I/O는 비블록화됩니다.

```
CREATE EVENT MONITOR SMITHPAY
  FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
  WRITE TO FILE '/home/jsmith/event/smithpay'
```

```
  MAXFILES 25
  MAXFILESIZE 1024
  NONBLOCKED
  APPEND
```

예 2: DEADLOCKS_EVTS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 교착 상태 이벤트를 수집한 다음 이를 상대 경로 DLOCKS에 기록합니다. 한 개의 파일이 작성되며, 최대 파일 크기는 없습니다. 이벤트 모니터가 활성화될 때마다, 이벤트 데이터를 00000000.evt 파일(있는 경우)에 추가합니다. 이벤트 모니터는 데이터베이스가 시작될 때마다 시작됩니다. I/0는 디폴트로 블록화됩니다.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
  FOR DEADLOCKS
       WRITE TO FILE 'DLOCKS'
  MAXFILES 1
  MAXFILESIZE NONE
  AUTOSTART
```

예 3: DB_APPLS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 연결 이벤트를 수집하고 데이터를 Named Pipe /home/jsmith/applpipe에 기록합니다.

```
CREATE EVENT MONITOR DB_APPLS
  FOR CONNECTIONS
  WRITE TO PIPE '/home/jsmith/applpipe'
```

예 4: 파티션된 데이터베이스 환경을 가정하고 FOO라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트를 수집하고 수집한 이벤트를 다음 파생 이름을 사용하여 SQL 테이블에 기록합니다.

- CONNHEADER_FOO
- STMT_FOO
- SUBSECTION_FOO
- CONTROL_FOO

테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 모든 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 모든 테이블에는 테이블 그룹에 대한 모든 요소가 포함됩니다. 즉, 요소 이름과 같은 이름을 갖는 컬럼이 정의됩니다.

```
CREATE EVENT MONITOR FOO
  FOR STATEMENTS
  WRITE TO TABLE
```

예 5: 파티션된 데이터베이스 환경을 가정하고 BAR라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트와 트랜잭션 이벤트를 수집하고 수집한 이벤트를 다음과 같이 테이블에 기록합니다.

- STMT 그룹의 데이터는 모두 MYDEPT.MYSTMTINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. ROWS_READ, ROWS_WRITTEN 및 STMT_TEXT 요소에 대해서만 컬럼을 작성하고, 그룹의 다른 요소는 버립니다.
- SUBSECTION 그룹의 데이터는 모두 MYDEPT.MYSUBSECTIONINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. 이 테이블에는 START_TIME, STOP_TIME 및 PARTIAL_RECORD를 제외한 모든 컬럼이 포함됩니다.
- XACT 그룹의 데이터는 모두 XACT_BAR 테이블에 기록됩니다. 테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 이 테이블에는 XACT 그룹에 있는 모든 요소가 포함됩니다.
- connheader나 control에 대해서는 테이블이 작성되지 않습니다. 따라서 이 두 그룹에 대한 데이터는 모두 버립니다.

```
CREATE EVENT MONITOR BAR
  FOR STATEMENTS, TRANSACTIONS
  WRITE TO TABLE
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
  XACT
```

CREATE EVENT MONITOR(잠금)

CREATE EVENT MONITOR(잠금)문은 데이터베이스를 사용할 때 발생하는 잠금 관련 이벤트를 기록하는 이벤트 모니터를 작성합니다.

호출

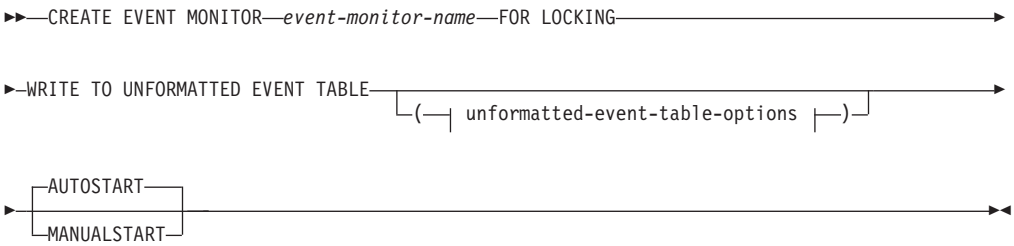
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

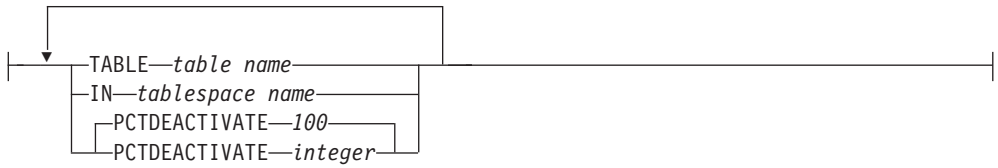
명령문의 권한 부여 ID는 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- DBADM 권한
- SQLADM 권한

구문



unformatted-event-table-options:



설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *event-monitor-name*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 이벤트의 유형을 소개합니다.

LOCKING

이 수동 이벤트 모니터는 DB2가 다음 조건 중 하나 이상의 상태가 될 때 생성되는 잠금 이벤트를 기록합니다.

- LOCKTIMEOUT: 잠금 시간종료가 발생했습니다.
- DEADLOCK: 교착 상태(희생(victim) 및 구성원(participant))에 잠금이 포함되었습니다.
- LOCKWAIT: 지정된 지속기간에 획득되지 않은 잠금입니다.

잠금 이벤트 모니터가 작성된다고 해서 잠금 데이터가 즉시 수집되는 것은 아닙니다. 관심있는 실제 잠금 이벤트는 워크로드 레벨 또는 데이터베이스 레벨에서 제어됩니다.

WRITE TO

데이터의 목표를 지정합니다.

UNFORMATTED EVENT TABLE

이벤트 모니터의 목표가 비형식화 이벤트 테이블임을 지정합니다. 비형식화 이벤트 테이블은 수집된 잠금 이벤트 모니터 데이터를 저장하는 데 사용됩니다. 데이터는 인라인된 BLOB 컬럼에 내부 2진 형식으로 저장됩니다. 각 이벤트는 이 테이블에 다중 레코드를 삽입할 수 있으며 각 삽입된 레코드는 연관된 BLOB 컨텐츠도 변하는 다른 유형일 수 있습니다. BLOB 컬럼의 데이터는 읽을 수 있는 형식이 아니며 db2evmonfmt Java 기반 도구, EVMON_FORMAT_UE_TO_XML 테이블 함수 또는 EVMON_FORMAT_UE_TO_TABLES 프로시저를 사용하여 XML 문서 또는 관계형 테이블 같이 이용 가능한 형식으로 변환되어야 합니다.

(unformatted-event-table-options)

비형식화 이벤트 테이블을 식별합니다. unformatted-event-table-options에 대한 값이 지정되지 않는 경우, CREATE EVENT MONITOR FOR LOCKING 처리는 다음과 같이 진행됩니다.

- 파생된 테이블 이름이 사용됩니다(아래에 설명되어 있음).
- 디폴트 테이블 스페이스가 선택됩니다(아래에 설명되어 있음).
- PCTDEACTIVATE가 100으로 설정됩니다.

TABLE *table-name*

비형식화 이벤트 테이블의 이름을 지정합니다. 이름을 제공하지 않는 경우 규정되지 않은 이름은 *event-monitor-name*과 같습니다. 즉, 비형식화 이벤트 테이블은 이벤트 모니터 이후에 이름이 지정됩니다.

다음 사항에 유의하십시오.

- 비형식화 이벤트 테이블은 CREATE EVENT MONITOR FOR LOCKING 문이 실행될 때 작성됩니다(테이블이 아직 존재하지 않는 경우).

- CREATE EVENT MONITOR FOR LOCKING 처리 중에, 비형식화 이벤트 테이블이 다른 이벤트 모니터가 사용하도록 이미 정의되지 않은 것으로 확인되는 경우 CREATE EVENT MONITOR FOR LOCKING문은 실패하고 오류가 다시 응용프로그램으로 전달됩니다. 비형식화 이벤트 테이블 이름이 SYSCAT.EVENTTABLES 카탈로그 뷰에 있는 값과 일치하면 다른 이벤트 모니터가 사용하도록 비형식화 이벤트 테이블이 정의됩니다. 비형식화 이벤트 테이블이 존재하고 다른 이벤트 모니터가 사용하도록 정의되지 않은 경우 이벤트 모니터는 비형식화 이벤트 테이블을 다시 사용합니다.
- 이벤트 모니터를 삭제해도 비형식화 이벤트 테이블은 삭제되지 않습니다. 연관된 비형식화 이벤트 테이블은 이벤트 모니터가 삭제된 후에 명시적으로 삭제해야 합니다.
- 비형식화 이벤트 테이블은 수동으로 프룬되어야 합니다.

IN *tablespace-name*

비형식화 이벤트 테이블이 작성될 테이블 스페이스를 정의합니다. CREATE EVENT MONITOR FOR LOCKING문은 테이블 스페이스를 작성하지 않습니다.

테이블 스페이스 이름을 제공하지 않으면 다음과 같이 테이블 스페이스가 선택됩니다.

```
  IF table space IBMDEFAULTGROUP over which the user
      has USE privilege exists
  THEN choose it
ELSE IF table space over which the user
      has USE privilege exists
  THEN choose it
ELSE return an error (SQLSTATE 42727)
```

PCTDEACTIVATE *integer*

비형식화 이벤트 테이블이 DMS 테이블 스페이스에 작성되는 경우 PCTDEACTIVATE 매개변수는 이벤트 모니터가 자동으로 비활성화되기 전에 테이블 스페이스가 채워져야 하는 정도를 지정합니다. 0부터 100까지의 값을 지정할 수 있으며 이 값은 백분율을 나타냅니다. 디폴트값은 100입니다. 100은 테이블 스페이스가 완전히 찼을 때 이벤트 모니터가 비활성화됨을 의미합니다. 테이블 스페이스에서 자동 크기 조정이 가능한 경우 PCTDEACTIVATE를 100으로 설정하도록 하십시오. 이 옵션은 SMS 테이블 스페이스의 경우 무시됩니다.

AUTOSTART

이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화될 때마다 이벤트 모니터가 자동으로 활성화되도록 지정합니다. WLM 이벤트 모니터 및 DB2DETAILDEADLOCK 이벤트 모니터의 디폴트 동작입니다.

MANUALSTART

이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화해야 한다는 것을 지정합니다. MANUALSTART 이벤트 모니터가 활성화된 다음, SET EVENT MONITOR STATE문을 사용하거나 인스턴스를 중지해야 비활성화될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 제외한 비WLM 이벤트 모니터의 디폴트 동작입니다.

주

- 이벤트 데이터는 비형식화 이벤트 테이블에서 인라인된 BLOB 데이터 컬럼에 삽입됩니다. 일반적으로 BLOB 데이터는 개별 LOB 테이블 스페이스에 저장되며 결과적으로 추가 성능 오버헤드를 겪을 수 있습니다. 기본 테이블의 데이터 페이지에 인라인될 때 BLOB 데이터는 이 오버헤드를 겪지 않습니다. BLOB 데이터의 *크기*가 테이블 스페이스 페이지 *크기*에서 레코드 접두부를 뺀 것보다 작은 경우 DB2 데이터베이스 관리 프로그램이 비형식화 이벤트 테이블 레코드의 BLOB 데이터 부분을 자동으로 인라인합니다. 그러므로 높은 효율 및 응용프로그램 처리량을 달성하려면 32KB 테이블 스페이스 및 연관된 버퍼 풀을 포함하여 가능한 큰 테이블 스페이스에 이벤트 모니터를 작성할 것을 권장합니다.

  예 : 잠금 이벤트 모니터는 현재 다음 두 레코드 유형을 갖습니다.

  – 응용프로그램 정보 레코드

  – 응용프로그램 활동 레코드

  응용프로그램 정보 레코드 = 최대 *크기* 3.5KB

  응용프로그램 활동 레코드 = 3KB + SQL문 텍스트 *크기*(여기서 SQL문 텍스트 *크기*는 최대 2MB임)

  응용프로그램 정보 레코드는 매우 작으며 4KB 페이지 *크기*가 사용되는 중에는 항상 인라인되어야 합니다. 응용프로그램 활동 레코드는 다음 공식을 기반으로 인라인됩니다.

```
Application Activity Record < inline length (Pagesize - overhead non-LOB columns (0.5KB))
   3KB + SQL statement text < inline length (Pagesize - overhead non-LOB columns (0.5KB))

        SQL statement text < Pagesize - nonLOB overhead (1K) - 3KB
        SQL statement text < 16KB - 1KB – 3KB
                          < 12KB
```

  그러므로 16KB 페이지 *크기*를 사용할 때 잠금 이벤트 모니터 레코드는 캡처되는 SQL문의 *크기*가 12KB 미만인 경우에만 인라인됩니다.

- 데이터베이스별로 잠금 이벤트 모니터를 하나만 작성하십시오. 모든 데이터베이스는 전이 목적을 위해 DB2DETAILDEADLOCK 이벤트 모니터가 사용 가능한 상태로 작성되는데, 이것은 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 사용 불가능하게 하고 제거해야 합니다. 그렇지 않으면 사용되지 않는 이벤트 모니터와 새 이벤트 모니터가 둘 다 데이터를 수집합니다. DB2DETAILDEADLOCK 이벤트 모니터를 제거하려면 다음 SQL문을 발행하십시오.

```
SET EVENT MONITOR DB2DETAILDEADLOCK state 0
DROP EVENT MONITOR DB2DETAILDEADLOCK
```

- 파티션된 데이터베이스 환경에서, 데이터는 테이블 스페이스가 존재하는 데이터베이스 파티션에서만 목표 비형식화 이벤트 테이블에 기록됩니다. 일부 데이터베이스 파티션에 목표 비형식화 이벤트 테이블에 대한 테이블 스페이스가 존재하지 않을 경우, 해당되는 목표 비형식화 이벤트 테이블에 대한 데이터가 무시됩니다. 이 때문에 사용자는 특정 데이터베이스 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터링 대상으로 선택할 데이터베이스 파티션의 서브세트를 선택할 수 있습니다.
- 파티션된 데이터베이스 환경에서, 일부 목표 비형식화 이벤트 테이블이 데이터베이스 파티션에 상주하지 않지만 다른 목표 비형식화 이벤트 테이블이 동일한 데이터베이스 파티션에 상주하면 데이터베이스 파티션에 있는 목표 비형식화 이벤트 테이블에 대한 데이터만 기록됩니다.

예:

예 1: 이 예는 작성 데이터베이스에 대해 발생하는 잠금 이벤트를 수집하지만 디폴트 비형식화 이벤트 테이블 LOCKEVMON에 데이터를 기록할 잠금 이벤트 모니터 LOCKEVMON을 작성합니다.

```
CREATE EVENT MONITOR LOCKEVMON
  FOR LOCKING
  WRITE TO UNFORMATTED EVENT TABLE
```

예 2: 이 예는 작성 데이터베이스에 대해 발생하는 잠금 이벤트를 수집하여 비형식화 이벤트 테이블 IMRAN.LOCKEVENTS에서 저장할 잠금 이벤트 모니터 LOCKEVMON을 작성합니다.

```
CREATE EVENT MONITOR LOCKEVMON
  FOR LOCKING
  WRITE TO UNFORMATTED EVENT TABLE (TABLE IMRAN.LOCKEVENTS)
```

예 3: 이 예는 작성 데이터베이스에 대해 발생하는 잠금 이벤트를 수집하여 테이블 스페이스 APPSPACE의 비형식화 이벤트 테이블 IMRAN.LOCKEVENTS에서 저장할 잠금 이벤트 모니터 LOCKEVMON을 작성합니다. 이벤트 모니터는 테이블 스페이스가 85% 찰 때 비활성화됩니다.

```
CREATE EVENT MONITOR LOCKEVMON
FOR LOCKING
WRITE TO UNFORMATTED EVENT TABLE
(TABLE IMRAN.LOCKEVENTS IN APPSPACE PCTDEACTIVATE 85)
```

CREATE EVENT MONITOR(통계)

CREATE EVENT MONITOR(통계) 명령문은 데이터베이스를 사용할 때 발생하는 통계 이벤트를 기록할 모니터를 정의합니다. 통계 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.

호출

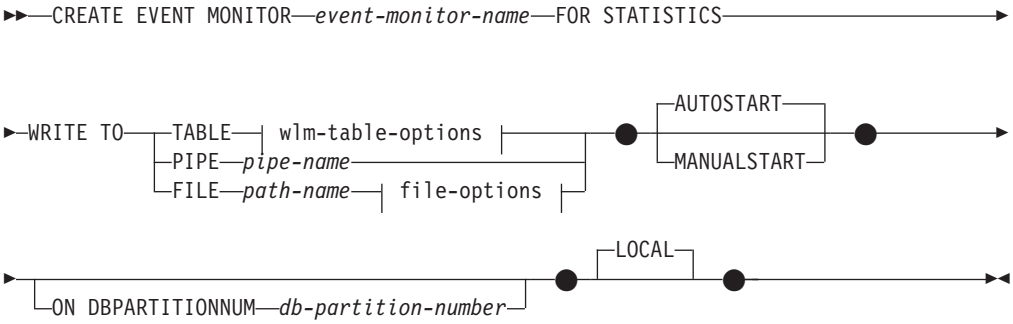
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

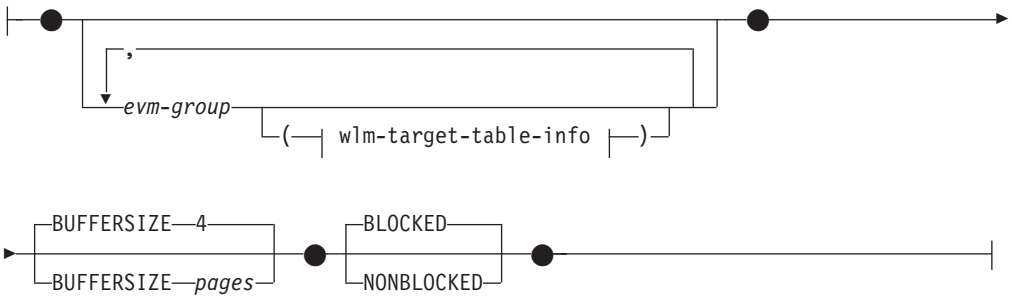
명령문의 권한 부여 ID는 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- DBADM 권한
- SQLADM 권한
- WLMADM 권한

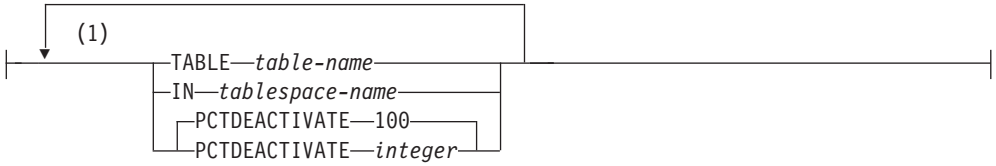
구문



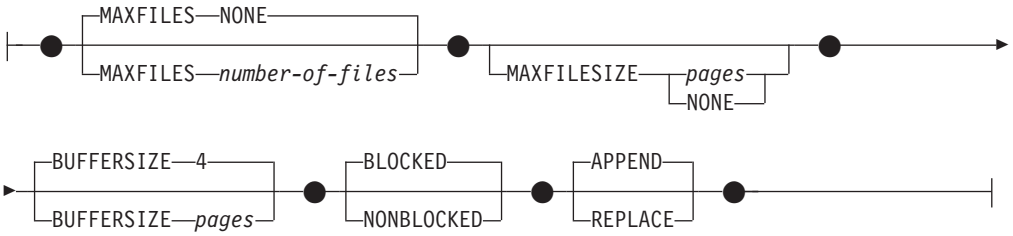
wlm-table-options:



wlm-target-table-info:



file-options:



주:

1 각 절을 한 번만 지정할 수 있습니다.

설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *event-monitor-name*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 레코드의 유형을 소개합니다.

STATISTICS

이벤트 모니터가 서비스 클래스, 워크로드 또는 작업 클래스 이벤트를 기록하도록 지정합니다.

- *period*분마다(여기서 *period*는 **wlm_collect_int** 데이터베이스 구성 매개변수의 값)
- wlm_collect_stats 프로시저가 호출된 경우

WRITE TO

데이터에 대한 목표를 사용합니다.

TABLE

이벤트 모니터 데이터의 목표가 데이터베이스 테이블 세트임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 하나 이상의 논리 데이터 그룹에 나눈 다음 각 그룹을 별도의 테이블에 삽입합니다. 목표 테이블이 있는 그룹의 데이터는 보존되고 목표 테이블이 없는 그룹의 데이터는 버립니다. 그룹에 포함된 각 모니터 요소는 같은 이름으로 된 테이블 컬럼에 맵핑됩니다. 해당하는 테이블 컬럼이 있는 요소만 테이블에 삽입되고, 다른 요소들은 버립니다.

wlm-table-options

논리 데이터 그룹에 대한 목표 테이블을 정의합니다. 이 절은 기록할 각 그룹화에 대하여 지정해야 합니다. 그러나 evm-group-info절을 지정하지 않으면 해당 이벤트 모니터 유형에 대한 모든 그룹이 기록됩니다.

evm-group

목표 테이블을 정의할 논리 데이터 그룹을 식별합니다. 이 값은 다음 표에서와 같이 이벤트 모니터의 유형에 따라 다릅니다.

이벤트 모니터 유형	**evm-group** 값
통계	• QSTATS • SCSTATS • WCSTATS • WLSTATS • HISTOGRAMBIN • CONTROL

BUFFERSIZE *pages*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 단위). 테이블 이벤트 모니터는 버퍼의 모든 데이터를 삽입하고 버퍼가 모두 처리되면 COMMIT를 발행합니다. 버퍼가 클수록 이벤트 모니터가 사용하는 커미트 범위도 큽니다. 활발하게 활동 중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 가지고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두 배의 버퍼링을 사용합니다.

각 버퍼의 디폴트 *크기*는 4페이지입니다(두 개의 16K 버퍼가 할당됨). 최소 *크기*는 1페이지입니다. 버퍼는 힙으로부터 할당되므로 버퍼의 최대 *크기*는 모니터 힙의 *크기*에 따라 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 *크기*를 늘리십시오.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

NONBLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 기다리지 않도록 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤

트 모니터만큼 데이터베이스 작업의 속도를 저하시키지는 않습니다. 그러나 NONBLOCKED 이벤트 모니터는 활발히 사용 중인 시스템에서 데이터가 유실될 수 있습니다.

PIPE

이벤트 모니터 데이터의 목표가 Named Pipe임을 지정합니다. 이벤트 모니터는 데이터를 단일 스트림의 파이프(즉, 하나인 것처럼 보이는 무한정으로 긴 파일)에 기록합니다. 데이터를 파이프에 기록할 때, 이벤트 모니터는 블록화된 쓰기를 수행하지 않습니다. 파이프 버퍼에 스페이스가 없으면, 이벤트 모니터에서는 데이터를 버립니다. 데이터 유실이 없도록 하려면, 즉시 데이터를 읽도록 해야 합니다.

pipe-name

이벤트 모니터가 데이터를 기록할 파이프의 이름(AIX에서 FIFO)입니다.

파이프의 이름 지정 규칙은 플랫폼에 따라 다릅니다. UNIX 운영 체제의 경우, 파이프 이름은 파일 이름과 같이 취급됩니다. 그러므로 상대 파이프 이름이 허용되고, 상대 경로 이름과 같이 취급됩니다(아래의 *path-name* 참조). 그러나 Windows에는 파이프 이름에 대한 특수한 구문이 있으며 그 결과로 절대 파이프 이름이 요구됩니다.

파이프의 존재 여부는 이벤트 모니터 작성시 점검되지 않습니다. 이벤트 모니터가 활성화될 때 읽을 파이프를 작성하고 여는 것은 모니터링 응용프로그램에서 수행됩니다. 이 때 파이프를 읽을 수 없으면, 이벤트 모니터 자체가 작동 중지되고 오류가 기록됩니다. 즉, 이벤트 모니터가 데이터베이스 시작 시 AUTOSTART 옵션에 의해 활성화된 경우, 이벤트 모니터는 시스템 오류 로그에 오류를 기록합니다. 이벤트 모니터가 SET EVENT MONITOR STATE SQL문으로 활성화된 경우, 해당 명령문은 실패합니다(SQLSTATE 58030).

FILE

이벤트 모니터 데이터의 목표가 파일(또는 파일 세트)임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 확장자가 “evt”인 일련의 8문자 순서화 파일로 작성합니다 (예: 00000000.evt, 00000001.evt 및 00000002.evt). 데이터는 더 작은 여러 조각으로 분리되더라도 하나의 논리 파일로 간주됩니다. 즉, 데이터 스트림의 시작은 00000000.evt 파일의 첫 번째 바이트이고, 데이터 스트림의 끝은 nnnnnnnn.evt 파일의 마지막 바이트입니다.

각 파일의 최대 *크기*뿐 아니라 최대 파일 수도 정의할 수 있습니다. 이벤트 모니터는 단일 이벤트 레코드를 두 파일에 걸쳐 분리시킬 수 없습니다. 그러나 이벤트 모니터는 관련 레코드들을 두 개의 다른 파일로 작성할 수 있습니다. 이벤트 파일을 처리할 때 그러한 관련 정보를 추적하는 것은 이 데이터를 사용하는 응용프로그램에서 수행됩니다.

path-name

이벤트 모니터가 이벤트 파일 데이터를 기록해야 하는 디렉토리의 이름입니다. 경로는 서버에 알려져 있어야 하지만 경로 그 자체는 다른 파티션에 상주할 수 있습니다(예: UNIX 시스템에서는 NFS 마운트 파일이 될 수 있음). *path-name*을 지정할 때 문자열 상수를 사용해야 합니다.

CREATE EVENT MONITOR를 실행할 때 디렉토리는 없어도 됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 경로의 존재 여부에 대한 점검이 수행됩니다. 이 때 목표 경로가 존재하지 않으면 오류가 발생합니다 (SQLSTATE 428A3).

AIX에서는 루트 디렉토리로 시작하고 Windows에서는 디스크 ID로 시작하는 절대 경로가 지정되는 경우, 지정된 경로가 사용되는 경로가 됩니다. 상대 경로(루트로 시작하지 않는 경로)를 지정할 경우, 데이터베이스 디렉토리에 있는 DB2EVENT 디렉토리에 대한 상대 경로가 사용됩니다.

상대 경로를 지정하면, DB2EVENT 디렉토리를 사용하여 그 경로를 절대 경로로 변환할 수 있습니다. 그러면, 절대 및 상대 경로 사이에 어떤 구별도 만들어지지 않습니다. 절대 경로는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 저장됩니다.

같은 목표 경로를 갖는 두 개 이상의 이벤트 모니터를 지정할 수 있습니다. 그러나 처음에 이벤트 모니터 중 하나가 활성화되고 대상 디렉토리가 비어 있지 않으면, 다른 이벤트 모니터 중 어떤 것도 활성화시킬 수 없습니다.

file-options

파일 형식에 대한 옵션을 지정합니다.

MAXFILES NONE

이벤트 모니터가 작성하는 이벤트 파일 수에 제한이 없음을 지정합니다. 이는 디폴트입니다.

MAXFILES *number-of-files*

언제든지 특정 이벤트 모니터에 대해 존재할 이벤트 모니터 파일 수에 제한이 있음을 지정합니다. 이벤트 모니터는 다른 파일을 작성해야 할 때마다, 디렉토리에 있는 .evt 파일의 수가 *number-of-files*보다 적은지 확인합니다. 이미 이 한계에 도달해 있으면, 이벤트 모니터는 스스로 작동 중지됩니다.

응용프로그램이 이벤트 파일이 작성된 후에 디렉토리에서 파일을 제거할 경우, 이벤트 모니터가 작성할 수 있는 총 파일 수는 *number-of-files*를 초과할 수 있습니다. 이 옵션은 이벤트 데이터가 지정된 디스크 스페이스의 양보다 많이 사용되지 않도록 사용자가 보증할 수 있도록 하기 위해 제공됩니다.

MAXFILESIZE *pages*

각 이벤트 모니터 파일의 *크기*에 제한이 있음을 지정합니다. 이벤트 모니터는 새 이벤트 레코드를 파일에 기록할 때마다, 파일이 *Pages*(4K 페이지 단위)보다 크지 않은지 확인합니다. 결과 파일이 너무 크면, 이벤트 모니터는 다음 파일로 전환합니다. 이 옵션의 디폴트값은 다음과 같습니다.

- Windows - 200 4K 페이지
- UNIX - 1000 4K 페이지

페이지 수는 최소한 이벤트 버퍼의 *크기*(페이지 단위)보다 많아야 합니다. 이 요건이 만족되지 않으면, 오류가 발생합니다(SQLSTATE 428A4).

MAXFILESIZE NONE

파일 *크기*에 제한이 없음을 지정합니다. MAXFILESIZE NONE을 지정할 경우, MAXFILES 1도 함께 지정해야 합니다. 이 옵션은 한 파일에 특정 이벤트 모니터에 대한 모든 이벤트 데이터가 포함된다는 것을 의미합니다. 이 경우 이벤트 파일은 00000000.evt입니다.

BUFFERSIZE *pages*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 단위). 모든 이벤트 모니터 파일 I/O는 이벤트 모니터의 성능이 향상되도록 버퍼화됩니다. 버퍼가 크면 클수록, 이벤트 모니터에 의해 수행되는 I/O는 적어집니다. 활발하게 활동 중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 가지고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두 배의 버퍼링을 사용합니다.

각 버퍼의 디폴트 *크기*는 4페이지입니다(두 개의 16K 버퍼가 할당됨). 최소 *크기*는 1페이지입니다. 버퍼가 힙으로부터 할당되므로 버퍼의 최대 *크기*는 모니터 힙의 *크기*뿐 아니라 MAXFILESIZE 매개변수 값으로 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 *크기*를 늘리십시오.

해당 데이터를 파이프에 기록하는 이벤트 모니터도 *크기*가 각각 2 페이지인 (구성 가능하지 않은) 버퍼를 갖고 있습니다. 이 버퍼들은 또한 모니터 힙(MON_HEAP)으로부터 할당됩니다. 파이프 목표를 갖는 각 활성 이벤트 모니터에 대해, 데이터베이스 힙의 *크기*를 2페이지씩 증가시키십시오.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는

지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

NONBLOCKED
이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 기다리지 않도록 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤트 모니터만큼 데이터베이스 작업의 속도를 저하시키지는 않습니다. 그러나 NONBLOCKED 이벤트 모니터는 활발히 사용 중인 시스템에서 데이터가 유실될 수 있습니다.

APPEND
이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 새 이벤트 데이터를 데이터 파일의 기존 스트림에 추가하도록 지정합니다. 이벤트 모니터가 다시 활성화될 경우, 작동이 중지되지 않았던 것처럼 이벤트 파일에 다시 기록하기 시작합니다. APPEND는 디폴트 옵션입니다.

새로 작성된 이벤트 모니터가 이벤트 데이터를 기록할 디렉토리에 기존의 이벤트 데이터가 있을 경우, APPEND 옵션은 CREATE EVENT MONITOR 수행시 적용되지 않습니다.

REPLACE
이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 모든 이벤트 파일을 지우고 00000000.evt 파일에 데이터를 기록하기 시작하도록 지정합니다.

MANUALSTART
이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화해야 한다는 것을 지정합니다. MANUALSTART 이벤트 모니터가 활성화된 다음, SET EVENT MONITOR STATE문을 사용하거나 인스턴스를 중지해야 비활성화될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 제외한 비WLM 이벤트 모니터의 디폴트 동작입니다.

AUTOSTART
이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화될 때마다 이벤트 모니터가 자동으로 활성화되도록 지정합니다. WLM 이벤트 모니터 및 DB2DETAILDEADLOCK 이벤트 모니터의 디폴트 동작입니다.

ON DBPARTITIONNUM *db-partition-number*
파일이나 파이프 이벤트 모니터를 실행할 데이터베이스 파티션을 지정합니다. 모니터링 범위가 LOCAL로 정의된 경우, 데이터는 지정된 파티션에서만 수집됩니다. 모니터 범위가 GLOBAL로 정의된 경우, 모든 데이터베이스 파티션이 데이터를 수집

하고 지정된 번호로 데이터베이스 파티션에 보고합니다. I/O 구성요소는 지정된 데이터베이스 파티션에서 실제로 실행되며 지정된 파일 또는 파이프에 레코드를 씁니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다. 파티션된 데이터베이스 환경의 경우, 테이블에 기록 이벤트 모니터는 목표 테이블에 대한 테이블 스페이스가 정의된 모든 데이터베이스 파티션에서 이벤트를 실행하고 기록합니다.

GLOBAL절이 지정되지 않으면 응용프로그램에 대해 현재 연결된 데이터베이스 파티션 번호가 사용됩니다.

LOCAL

이벤트 모니터를 실행 중인 데이터베이스 파티션에만 보고합니다. 데이터베이스 활동에 대한 부분적 추적을 제공합니다. 이는 디폴트값입니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다.

규칙

- STATISTICS 이벤트 유형은 특정 이벤트 모니터 정의에 있는 다른 이벤트 유형과 결합될 수 없습니다.

주

- 이벤트 모니터 정의는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 기록됩니다. 이벤트 자체는 SYSCAT.EVENTS 카탈로그 뷰에 기록됩니다. 목표 테이블의 이름은 SYSCAT.EVENTTABLES 카탈로그 뷰에 기록됩니다.
- BUFFERSIZE 매개변수는 STMT, STMT_HISTORY, DATA_VALUE 및 DETAILED_DLCONN 이벤트의 *크기*를 제한합니다. STMT 또는 STMT_HISTORY 이벤트의 *크기*가 버퍼에 맞지 않으면 명령문 텍스트를 절단하여 절단됩니다. DETAILED_DLCONN 이벤트의 *크기*가 버퍼에 맞지 않으면 잠금을 제거하여 해당 이벤트가 절단됩니다. 그래도 맞지 않으면 명령문 텍스트를 절단합니다. DATA_VAL 이벤트의 *크기*가 버퍼에 맞지 않으면 데이터 값이 절단됩니다.
- 이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화되지 않은 경우, 그 다음 데이터베이스 파티션이 활성화되면 이벤트 모니터가 활성화됩니다.
- 이벤트 모니터는 일단 활성화되면 이벤트 모니터가 명시적으로 비활성화되거나 인스턴스 처리가 되풀이될 때까지 자동 시작 이벤트 모니터처럼 작동합니다. 즉 데이터베이스 파티션이 비활성화되었을 때 이벤트 모니터가 활성화되고 그후에 그 데이터베이스 파티션이 다시 활성화되면, 이벤트 모니터도 명시적으로 재활성화됩니다.
- *테이블 이벤트 모니터에 기록:*: 일반 메모:
  - 모든 목표 테이블은 CREATE EVENT MONITOR문을 실행할 때 작성됩니다.
  - 어떤 이유로든 테이블 작성에 실패하면 응용프로그램에 오류가 전달되고 CREATE EVENT MONITOR문이 실패합니다.

- 목표 테이블은 한 개의 이벤트 모니터만 사용할 수 있습니다. CREATE EVENT MONITOR를 처리하는 동안 다른 이벤트 모니터에서 사용하도록 목표 테이블이 이미 정의되어 있음이 발견되면 CREATE EVENT MONITOR문이 실패하고 응용프로그램에 오류가 전달됩니다. 테이블 이름이 SYSCAT.EVENTTABLES 카탈로그 뷰에 있는 값과 일치하면 다른 이벤트 모니터가 테이블을 사용하도록 테이블이 정의됩니다.
- CREATE EVENT MONITOR를 처리하는 동안 테이블은 있지만 다른 이벤트 모니터에서 사용하도록 테이블이 정의되어 있지 않으면 테이블이 작성되지 않고 처리가 계속됩니다. 응용프로그램에 경고가 전달됩니다.
- 모든 테이블 스페이스는 CREATE EVENT MONITOR문이 실행되기 전에 존재해야 합니다. CREATE EVENT MONITOR문은 테이블 스페이스를 작성하지 않습니다.
- LOCAL 키워드와 GLOBAL 키워드를 지정할 경우 무시됩니다. WRITE TO TABLE 이벤트 모니터의 경우에는 이벤트 모니터 출력 프로세스나 스레드가 이 인스턴스의 각 데이터베이스 파티션에서 시작되고 이러한 각 프로세스는 프로세스가 실행되는 데이터베이스 파티션에 대해서만 데이터를 보고합니다.
- 테이블에 기록 이벤트 모니터는 다음과 같은 플랫 모니터 로그 파일이나 파이프 형식의 이벤트 유형은 기록하지 않습니다.
  - LOG_STREAM_HEADER
  - LOG_HEADER
  - DB_HEADER(db_name 및 db_path 요소는 기록되지 않습니다. conn_time 요소는 CONTROL에 기록됩니다.)
- 파티션된 데이터베이스 환경의 경우, 데이터는 목표 테이블의 테이블 스페이스가 존재하는 데이터베이스 파티션에서만 목표 테이블에 기록됩니다. 일부 데이터베이스 파티션에 목표 테이블에 대한 테이블 스페이스가 존재하지 않을 경우, 해당 목표 테이블에 대한 데이터가 무시됩니다. 이 때문에 사용자는 특정 데이터베이스 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터링할 데이터베이스 파티션의 서브세트를 선택할 수 있습니다.

  파티션된 데이터베이스 환경의 경우, 일부 목표 테이블이 데이터베이스 파티션에 상주하지 않지만 다른 목표 테이블이 동일한 데이터베이스 파티션에 상주하면 데이터베이스 파티션에 있는 목표 테이블에 대한 데이터만 기록됩니다.
- 사용자는 모든 목표 테이블을 직접 프룬해야 합니다.

테이블 컬럼:

- 테이블의 컬럼 이름은 이벤트 모니터의 요소 ID와 일치합니다. sqlm_time(경과 시간) 유형의 모니터 변수는 예외입니다. 이러한 유형에 대한 컬럼 이름은

TYPE_NAME_S와 TYPE_NAME_MS이고, 이 두 이름은 각각 초와 마이크로초로 시간을 저장하는 컬럼을 나타냅니다. 일치하는 목표 테이블 컬럼이 없는 이벤트 모니터 요소는 무시됩니다.

- 그룹에 대한 전체 요소 목록을 포함하는 CREATE EVENT MONITOR 명령을 빌드하려면 db2evtbl 명령을 사용하십시오.
- 모니터 요소에 사용되는 컬럼 유형은 다음 맵핑과 상호 관련이 있습니다.

```
SQLM_TYPE_STRING                    CHAR[n], VARCHAR[n] or CLOB(n)
                                              (이벤트 모니터 레코드의 데이터가
                                    n 바이트를 초과할 경우,
                                              데이터는 절단됩니다.)
SQLM_TYPE_U8BIT 및 SQLM_TYPE_8BIT   SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_16BIT 및 SQLM_TYPE_U16BIT SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_32BIT 및 SQLM_TYPE_U32BIT INTEGER 또는 BIGINT
SQLM_TYPE_U64BIT 및 SQLM_TYPE_64BIT  BIGINT
sqlm_timestamp                      TIMESTAMP
sqlm_time(경과 시간)                 BIGINT
sqlca:
    sqlerrmc                          VARCHAR[72]
    sqlstate                          CHAR[5]
    sqlwarn                           CHAR[11]
    기타 파일                          INTEGER 또는 BIGINT
```

- 컬럼은 NOT NULL로 정의됩니다.
- CLOB 컬럼이 있는 테이블은 VARCHAR 컬럼이 있는 테이블에 비해 성능이 떨어지므로 STMT *evm-group* 값(또는 DEADLOCKS WITH DETAILS 이벤트 유형을 사용할 경우에는 DLCONN *evm-group* 값)을 지정할 때 TRUNC 키워드를 사용하는 것이 좋습니다.
- 다른 목표 테이블과 달리 CONTROL 테이블의 컬럼은 모니터 요소 ID와 일치하지 않습니다. 컬럼은 다음과 같이 정의됩니다.

```
컬럼 이름         데이터 유형    널(NULL)  설명
                                입력 가능
-------------     ---------     --------  -----------
PARTITION_KEY     INTEGER       N         분산 키
                                          (파티션된 데이터베이스만 해당)
PARTITION_NUMBER  INTEGER       N         데이터베이스 파티션 번호
                                          (파티션된 데이터베이스만 해당)
EVMONNAME         VARCHAR(128)  N         이벤트 모니터 이름
MESSAGE           VARCHAR(128)  N         MESSAGE_TIME 컬럼의
                                          등록 정보를 기술합니다.
                                          다음 중 하나가
                                          될 수 있습니다.
                                          - FIRST_CONNECT(데이터베이스
                                          활성화 이후 처음
                                          데이터베이스에 연결한 시간)
                                          - EVMON_START(EVMONNAME에
      나열된 이벤트 모니터가
                                          시작된 시간)
                                          - OVERFLOWS:n(버퍼 오버플로우로 인해
                                            n개의 레코드를
        버린다는 것을 나타냄)
                                          - LAST_DROPPED_RECORD
                                            (오버플로우가 발생된
        마지막 시간)
MESSAGE_TIME    TIMESTAMP       N         시간소인
```

- 파티션된 데이터베이스 환경에서 각 테이블의 첫 번째 컬럼의 이름은 PARTITION_KEY이고, NOT NULL이며 INTEGER 유형입니다. 이 컬럼은 테이블의 분산 키로 사용됩니다. 각 이벤트 모니터 프로세스에서 프로세스가 실행되는 데이터베이스 파티션에 데이터를 삽입할 수 있도록 이 컬럼의 값이 선택됩니다. 즉, 삽입 조작은 이벤트 모니터 프로세스가 실행되는 데이터베이스 파티션에서 로컬로 수행됩니다. 모든 데이터베이스 파티션에서 PARTITION_KEY 필드는

같은 값을 포함합니다. 즉 데이터베이스 파티션이 삭제되고 데이터 재분배가 수행될 경우 삭제된 데이터베이스 파티션에 있는 모든 데이터는 고르게 분배되지 않고 다른 데이터베이스 파티션으로 이동됩니다. 따라서 데이터베이스 파티션을 제거하기 전에 제거할 데이터베이스 파티션에 있는 모든 테이블 행을 삭제하는 것이 좋습니다.

– 파티션된 데이터베이스 환경에서는 각 테이블에 대해 PARTITION_NUMBER 컬럼을 정의할 수 있습니다. 이 컬럼은 INTEGER 유형인 NOT NULL입니다. 이 컬럼에는 데이터가 삽입된 데이터베이스 파티션의 번호가 포함합니다. PARTITION_KEY 컬럼과 달리 PARTITION_NUMBER 컬럼은 필수가 아닙니다. 파티션되지 않은 데이터베이스 환경에서는 PARTITION_NUMBER 컬럼을 정의할 수 없습니다.

테이블 속성

– 디폴트 테이블 속성이 사용됩니다. 테이블을 작성할 때 분산 키(파티션된 데이터베이스에만 해당)를 제외하고 추가 옵션은 지정하지 않습니다.
– 테이블에 대한 인덱스를 작성할 수 있습니다.
– VOLATILE, RI, 트리거 및 제한조건 등과 같은 추가 테이블 속성을 추가할 수는 있지만 이벤트 모니터 프로세스(또는 스레드)는 이런 속성을 무시합니다.
– ″not logged initially″를 테이블 속성으로 추가하면 처음 COMMIT를 실행할 때 이 속성이 해제되고 다시 설정되지 않습니다.

이벤트 모니터 활성화

– 이벤트 모니터가 활성화되면 SYSCAT.EVENTTABLES 카탈로그 뷰에서 모든 목표 테이블 이름이 검색됩니다.
– 파티션된 데이터베이스 환경에서는 인스턴스의 모든 데이터베이스 파티션에서 활성화 처리가 이루어집니다. 특정 데이터베이스 파티션에서는 활성화 처리가 각 목표 테이블의 테이블 스페이스 및 데이터베이스 파티션 그룹을 판별합니다. 이벤트 모니터는 데이터베이스 파티션에 최소한 하나의 목표 테이블이 존재할 경우에만 데이터베이스 파티션에 활성화됩니다. 또한 데이터베이스 파티션에 일부 목표 테이블이 없을 경우, 해당 테이블에 대한 데이터가 런타임 처리시 삭제되도록 해당 목표 테이블에 대한 플래그가 표시됩니다.
– 이벤트 모니터가 활성화될 때 목표 테이블이 없거나, 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주하지 않을 경우에도 활성화가 계속됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 테이블이 있거나 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주할 경우 이 테이블에 삽입될 데이터는 무시됩니다.
– 활성화 프로세스는 각 목표 테이블의 유효성을 확인합니다. 유효성 확인이 실패하면 이벤트 모니터가 활성화되지 않고 관리 로그에 메시지가 기록됩니다.

- 파티션된 데이터베이스 환경에서 활성화하는 동안에는 FIRST_CONNECT와 EVMON_START에 대한 CONTROL 테이블의 행만 카탈로그 데이터베이스 파티션에 삽입됩니다. 이 때 제어 테이블에 대한 테이블 스페이스가 카탈로그 데이터베이스 파티션에 있어야 합니다. 테이블 스페이스가 카탈로그 데이터베이스 파티션에 없으면 이런 삽입 조작이 수행되지 않습니다.
- 파티션된 데이터베이스 환경에서는 테이블에 기록 이벤트 모니터가 활성화될 때 파티션이 활성화되어 있지 않으면, 다음에 파티션이 활성화될 때 이벤트 모니터가 활성화됩니다.

런타임

- 이벤트 모니터가 DATAACCESS 권한으로 실행됩니다.
- 이벤트 모니터가 활성화되어 있는 동안 목표 테이블에 대한 삽입 조작이 실패하면 다음 작업이 수행됩니다.
  - 언커미트된 변경사항을 롤백합니다.
  - 관리 로그에 메시지가 기록됩니다.
  - 이벤트 모니터가 비활성화됩니다.
- 이벤트 모니터가 활성화되어 있을 경우 이벤트 모니터 버퍼에 대한 처리가 완료되면 로컬 COMMIT가 수행됩니다.
- 파티션된 데이터베이스 환경에서는 65,535바이트까지 가능한 실제 명령문 텍스트는 응용프로그램 코디네이터 데이터베이스 파티션에서 실행되는 이벤트 모니터에 의해 STMT 테이블이나 DLCONN 테이블에 저장됩니다. 다른 데이터베이스 파티션에서는 이 값이 0입니다.
- 파티션되지 않은 데이터베이스 환경에서는 마지막 응용프로그램이 종료되고 데이터베이스가 명시적으로 활성화되지 않았을 때 모든 테이블에 기록 이벤트 모니터가 비활성화됩니다. 파티션된 데이터베이스 환경에서는 카탈로그 파티션이 비활성화될 때 테이블에 기록 이벤트 모니터가 비활성화됩니다.
- DROP EVENT MONITOR문은 목표 테이블을 삭제하지 않습니다.
- 테이블에 기록 이벤트 모니터가 활성화될 때마다, 이벤트 모니터가 사용 중인 동안 수정되지 않도록 IN 테이블이 각 목표 테이블을 잠급니다. 이벤트 모니터가 사용 중인 동안 테이블 잠금은 모든 테이블에서 유지됩니다. 독점 액세스가 목표 테이블에 필요하면(예를 들어, 유틸리티를 실행하려고 할 때) 이러한 액세스를 시도하기 전에 먼저 이벤트 모니터를 비활성화하여 테이블 잠금을 해제합니다.

- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - 쉼표를 사용하여 *wlm-target-table-info*절에서 여러 옵션을 구분할 수 있습니다.

예

예 1: SMITHPAY라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 데이터베이스에 대해, 그리고 JSMITH 권한 부여 ID가 소유하는 PAYROLL 응용프로그램에서 수행되는 SQL문에 대해 이벤트 데이터를 수집합니다. 데이터는 절대 경로 /home/jsmith/event/smithpay/에 추가됩니다. 최대 25개의 파일이 작성됩니다. 각 파일은 최대 1 024 4K 페이지가 됩니다. 파일 I/O는 비블록화됩니다.

```
CREATE EVENT MONITOR SMITHPAY
  FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
  WRITE TO FILE '/home/jsmith/event/smithpay'
  MAXFILES 25
  MAXFILESIZE 1024
  NONBLOCKED
  APPEND
```

예 2: DEADLOCKS_EVTS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 교착 상태 이벤트를 수집한 다음 이를 상대 경로 DLOCKS에 기록합니다. 한 개의 파일이 작성되며, 최대 파일 크기는 없습니다. 이벤트 모니터가 활성화될 때마다, 이벤트 데이터를 00000000.evt 파일(있는 경우)에 추가합니다. 이벤트 모니터는 데이터베이스가 시작될 때마다 시작됩니다. I/0는 디폴트로 블록화됩니다.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
  FOR DEADLOCKS
       WRITE TO FILE 'DLOCKS'
  MAXFILES 1
  MAXFILESIZE NONE
  AUTOSTART
```

예 3: DB_APPLS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 연결 이벤트를 수집하고 데이터를 Named Pipe /home/jsmith/applpipe에 기록합니다.

```
CREATE EVENT MONITOR DB_APPLS
  FOR CONNECTIONS
  WRITE TO PIPE '/home/jsmith/applpipe'
```

예 4: 파티션된 데이터베이스 환경을 가정하고 FOO라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트를 수집하고 수집한 이벤트를 다음 파생 이름을 사용하여 SQL 테이블에 기록합니다.

- CONNHEADER_FOO
- STMT_FOO
- SUBSECTION_FOO
- CONTROL_FOO

테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 모든 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 모든 테이블에는 테이블 그룹에 대한 모든 요소가 포함됩니다. 즉, 요소 이름과 같은 이름을 갖는 컬럼이 정의됩니다.

```
CREATE EVENT MONITOR FOO
  FOR STATEMENTS
  WRITE TO TABLE
```

예 5: 파티션된 데이터베이스 환경을 가정하고 BAR라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트와 트랜잭션 이벤트를 수집하고 수집한 이벤트를 다음과 같이 테이블에 기록합니다.

- STMT 그룹의 데이터는 모두 MYDEPT.MYSTMTINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. ROWS_READ, ROWS_WRITTEN 및 STMT_TEXT 요소에 대해서만 컬럼을 작성하고, 그룹의 다른 요소는 버립니다.
- SUBSECTION 그룹의 데이터는 모두 MYDEPT.MYSUBSECTIONINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. 이 테이블에는 START_TIME, STOP_TIME 및 PARTIAL_RECORD를 제외한 모든 컬럼이 포함됩니다.
- XACT 그룹의 데이터는 모두 XACT_BAR 테이블에 기록됩니다. 테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 이 테이블에는 XACT 그룹에 있는 모든 요소가 포함됩니다.
- connheader나 control에 대해서는 테이블이 작성되지 않습니다. 따라서 이 두 그룹에 대한 데이터는 모두 버립니다.

```
CREATE EVENT MONITOR BAR
  FOR STATEMENTS, TRANSACTIONS
  WRITE TO TABLE
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
  XACT
```

CREATE EVENT MONITOR(임계값 위반)

CREATE EVENT MONITOR(임계값 위반) 명령문은 데이터베이스를 사용할 때 발생하는 임계값 위반 이벤트를 기록할 모니터를 정의합니다. 임계값 위반 이벤트 모니터의 정의는 데이터베이스에서 이벤트를 기록해야 하는 위치도 지정합니다.

호출

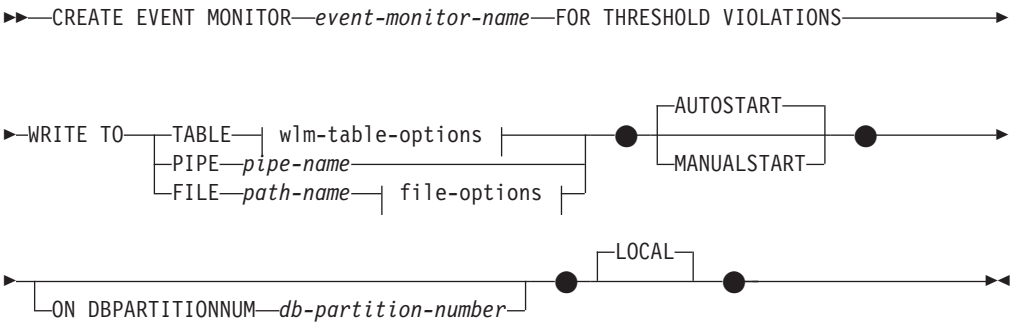
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

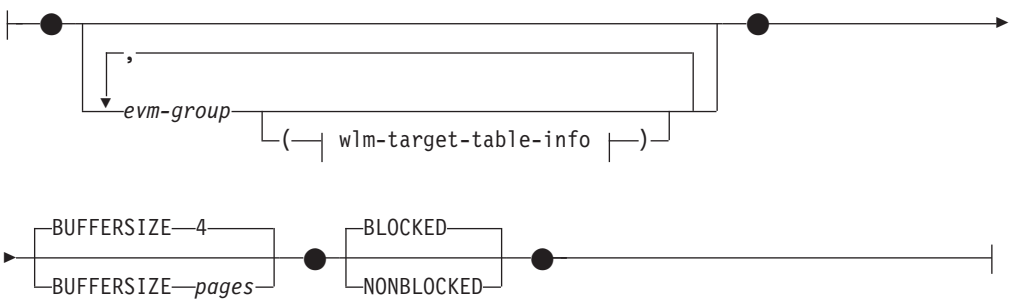
명령문의 권한 부여 ID는 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- DBADM 권한
- SQLADM 권한
- WLMADM 권한

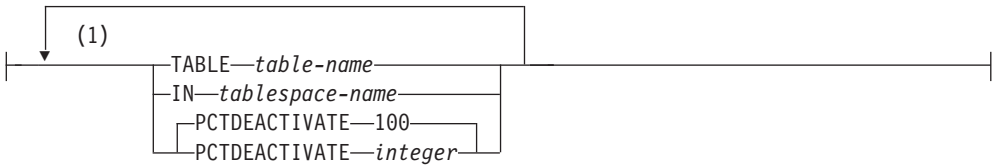
구문



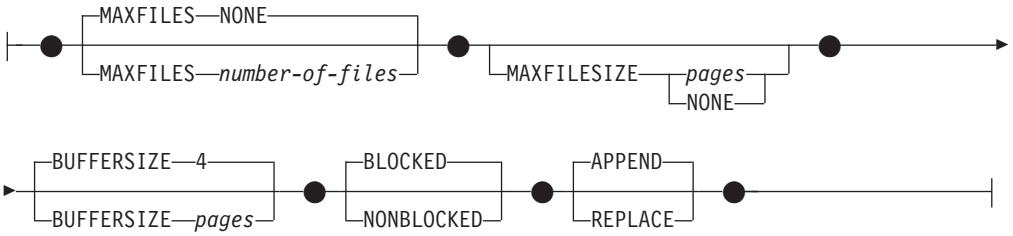
wlm-table-options:



wlm-target-table-info:



file-options:



주:

1 각 절을 한 번만 지정할 수 있습니다.

설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *event-monitor-name*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 레코드의 유형을 소개합니다.

THRESHOLD VIOLATIONS

임계값이 위반될 때 이벤트 모니터가 임계값 위반 이벤트를 기록하도록 지정합니다. 이러한 이벤트는 완료 시점이 아닌 활동 중에 기록될 수 있습니다.

WRITE TO

데이터에 대한 목표를 사용합니다.

TABLE

이벤트 모니터 데이터의 목표가 데이터베이스 테이블 세트임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 하나 이상의 논리 데이터 그룹에 나눈 다음 각 그룹을 별도의 테이블에 삽입합니다. 목표 테이블이 있는 그룹의 데이터는 보존되고 목표 테이블이 없는 그룹의 데이터는 버립니다. 그룹에 포함된 각 모니터 요소는 같은 이름으로 된 테이블 컬럼에 맵핑됩니다. 해당하는 테이블 컬럼이 있는 요소만 테이블에 삽입되고, 다른 요소들은 버립니다.

wlm-table-options

논리 데이터 그룹에 대한 목표 테이블을 정의합니다. 이 절은 기록할 각 그

룹화에 대하여 지정해야 합니다. 그러나 evm-group-info절을 지정하지 않으면 해당 이벤트 모니터 유형에 대한 모든 그룹이 기록됩니다.

evm-group

목표 테이블을 정의할 논리 데이터 그룹을 식별합니다. 이 값은 다음 표에서와 같이 이벤트 모니터의 유형에 따라 다릅니다.

이벤트 모니터 유형	**evm-group** 값
임계값 위반	• THRESHOLDVIOLATIONS • CONTROL

BUFFERSIZE *pages*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 단위). 테이블 이벤트 모니터는 버퍼의 모든 데이터를 삽입하고 버퍼가 모두 처리되면 COMMIT를 발행합니다. 버퍼가 클수록 이벤트 모니터가 사용하는 커미트 범위도 큽니다. 활발하게 활동 중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 가지고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두 배의 버퍼링을 사용합니다.

각 버퍼의 디폴트 *크기*는 4페이지입니다(두 개의 16K 버퍼가 할당됨). 최소 *크기*는 1페이지입니다. 버퍼는 힙으로부터 할당되므로 버퍼의 최대 *크기*는 모니터 힙의 *크기*에 따라 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 *크기*를 늘리십시오.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

NONBLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 기다리지 않도록 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤트 모니터만큼 데이터베이스 작업의 속도를 저하시키지는 않습니다. 그러나 NONBLOCKED 이벤트 모니터는 활발히 사용 중인 시스템에서 데이터가 유실될 수 있습니다.

PIPE

이벤트 모니터 데이터의 목표가 Named Pipe임을 지정합니다. 이벤트 모니터는 데이터를 단일 스트림의 파이프(즉, 하나인 것처럼 보이는 무한정으로 긴 파일)에 기록합니다. 데이터를 파이프에 기록할 때, 이벤트 모니터는 블록화된 쓰

기를 수행하지 않습니다. 파이프 버퍼에 스페이스가 없으면, 이벤트 모니터에서는 데이터를 버립니다. 데이터 유실이 없도록 하려면, 즉시 데이터를 읽도록 해야 합니다.

pipe-name

이벤트 모니터가 데이터를 기록할 파이프의 이름(AIX에서 FIFO)입니다.

파이프의 이름 지정 규칙은 플랫폼에 따라 다릅니다. UNIX 운영 체제의 경우, 파이프 이름은 파일 이름과 같이 취급됩니다. 그러므로 상대 파이프 이름이 허용되고, 상대 경로 이름과 같이 취급됩니다(아래의 *path-name* 참조). 그러나 Windows에는 파이프 이름에 대한 특수한 구문이 있으며 그 결과로 절대 파이프 이름이 요구됩니다.

파이프의 존재 여부는 이벤트 모니터 작성시 점검되지 않습니다. 이벤트 모니터가 활성화될 때 읽을 파이프를 작성하고 여는 것은 모니터링 응용프로그램에서 수행됩니다. 이 때 파이프를 읽을 수 없으면, 이벤트 모니터 자체가 작동 중지되고 오류가 기록됩니다. 즉, 이벤트 모니터가 데이터베이스 시작 시 AUTOSTART 옵션에 의해 활성화된 경우, 이벤트 모니터는 시스템 오류 로그에 오류를 기록합니다. 이벤트 모니터가 SET EVENT MONITOR STATE SQL문으로 활성화된 경우, 해당 명령문은 실패합니다(SQLSTATE 58030).

FILE

이벤트 모니터 데이터의 목표가 파일(또는 파일 세트)임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 확장자가 “evt”인 일련의 8문자 순서화 파일로 작성합니다 (예: 00000000.evt, 00000001.evt 및 00000002.evt). 데이터는 더 작은 여러 조각으로 분리되더라도 하나의 논리 파일로 간주됩니다. 즉, 데이터 스트림의 시작은 00000000.evt 파일의 첫 번째 바이트이고, 데이터 스트림의 끝은 nnnnnnnn.evt 파일의 마지막 바이트입니다.

각 파일의 최대 *크기*뿐 아니라 최대 파일 수도 정의할 수 있습니다. 이벤트 모니터는 단일 이벤트 레코드를 두 파일에 걸쳐 분리시킬 수 없습니다. 그러나 이벤트 모니터는 관련 레코드들을 두 개의 다른 파일로 작성할 수 있습니다. 이벤트 파일을 처리할 때 그러한 관련 정보를 추적하는 것은 이 데이터를 사용하는 응용프로그램에서 수행됩니다.

path-name

이벤트 모니터가 이벤트 파일 데이터를 기록해야 하는 디렉토리의 이름입니다. 경로는 서버에 알려져 있어야 하지만 경로 그 자체는 다른 파티션에 상주할 수 있습니다(예: UNIX 시스템에서는 NFS 마운트 파일이 될 수 있음). *path-name*을 지정할 때 문자열 상수를 사용해야 합니다.

CREATE EVENT MONITOR를 실행할 때 디렉토리는 없어도 됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 경로의 존재 여부에 대한 점검이 수행됩니다. 이 때 목표 경로가 존재하지 않으면 오류가 발생합니다 (SQLSTATE 428A3).

AIX에서는 루트 디렉토리로 시작하고 Windows에서는 디스크 ID로 시작하는 절대 경로가 지정되는 경우, 지정된 경로가 사용되는 경로가 됩니다. 상대 경로(루트로 시작하지 않는 경로)를 지정할 경우, 데이터베이스 디렉토리에 있는 DB2EVENT 디렉토리에 대한 상대 경로가 사용됩니다.

상대 경로를 지정하면, DB2EVENT 디렉토리를 사용하여 그 경로를 절대 경로로 변환할 수 있습니다. 그러면, 절대 및 상대 경로 사이에 어떤 구별도 만들어지지 않습니다. 절대 경로는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 저장됩니다.

같은 목표 경로를 갖는 두 개 이상의 이벤트 모니터를 지정할 수 있습니다. 그러나 처음에 이벤트 모니터 중 하나가 활성화되고 대상 디렉토리가 비어 있지 않으면, 다른 이벤트 모니터 중 어떤 것도 활성화시킬 수 없습니다.

file-options

파일 형식에 대한 옵션을 지정합니다.

MAXFILES NONE

이벤트 모니터가 작성하는 이벤트 파일 수에 제한이 없음을 지정합니다. 이는 디폴트입니다.

MAXFILES *number-of-files*

언제든지 특정 이벤트 모니터에 대해 존재할 이벤트 모니터 파일 수에 제한이 있음을 지정합니다. 이벤트 모니터는 다른 파일을 작성해야 할 때마다, 디렉토리에 있는 .evt 파일의 수가 *number-of-files*보다 적은지 확인합니다. 이미 이 한계에 도달해 있으면, 이벤트 모니터는 스스로 작동 중지됩니다.

응용프로그램이 이벤트 파일이 작성된 후에 디렉토리에서 파일을 제거할 경우, 이벤트 모니터가 작성할 수 있는 총 파일 수는 *number-of-files*를 초과할 수 있습니다. 이 옵션은 이벤트 데이터가 지정된 디스크 스페이스의 양보다 많이 사용되지 않도록 사용자가 보증할 수 있도록 하기 위해 제공됩니다.

MAXFILESIZE *pages*

각 이벤트 모니터 파일의 크기에 제한이 있음을 지정합니다. 이벤트 모니터는 새 이벤트 레코드를 파일에 기록할 때마다, 파일이 *Pages*(4K

페이지 단위)보다 *크기* 않은지 확인합니다. 결과 파일이 너무 *크면*, 이벤트 모니터는 다음 파일로 전환합니다. 이 옵션의 디폴트값은 다음과 같습니다.

- Windows - 200 4K 페이지
- UNIX - 1000 4K 페이지

페이지 수는 *최소한* 이벤트 버퍼의 *크기*(페이지 단위)보다 많아야 합니다. 이 요건이 만족되지 않으면, 오류가 발생합니다(SQLSTATE 428A4).

MAXFILESIZE NONE

파일 *크기*에 제한이 없음을 지정합니다. MAXFILESIZE NONE을 지정할 경우, MAXFILES 1도 함께 지정해야 합니다. 이 옵션은 한 파일에 특정 이벤트 모니터에 대한 모든 이벤트 데이터가 포함된다는 것을 의미합니다. 이 경우 이벤트 파일은 00000000.evt입니다.

BUFFERSIZE *pages*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 단위). 모든 이벤트 모니터 파일 I/O는 이벤트 모니터의 성능이 향상되도록 버퍼화됩니다. 버퍼가 *크면 클수록*, 이벤트 모니터에 의해 수행되는 I/O는 적어집니다. 활발하게 활동 중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 가지고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두 배의 버퍼링을 사용합니다.

각 버퍼의 디폴트 *크기*는 4페이지입니다(두 개의 16K 버퍼가 할당됨). 최소 *크기*는 1페이지입니다. 버퍼가 힙으로부터 할당되므로 버퍼의 최대 *크기*는 모니터 힙의 *크기*뿐 아니라 MAXFILESIZE 매개변수 값으로 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, **mon_heap_sz** 데이터베이스 관리 프로그램 구성 매개변수의 *크기*를 늘리십시오.

해당 데이터를 파이프에 *기록*하는 이벤트 모니터도 *크기*가 각각 2 페이지인 (구성 가능하지 않은) 버퍼를 갖고 있습니다. 이 버퍼들은 또한 모니터 힙(MON_HEAP)으로부터 할당됩니다. 파이프 목표를 갖는 각 활성 이벤트 모니터에 대해, 데이터베이스 힙의 *크기*를 2페이지씩 증가시키십시오.

BLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 각 에이전트가 기다리도록 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 디폴트 옵션입니다.

NONBLOCKED

이벤트를 생성하는 각 에이전트에서 두 이벤트 버퍼가 모두 가득 찼는지를 판별할 경우 이벤트 버퍼가 디스크에 기록될 때까지 기다리지 않도록 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤트 모니터만큼 데이터베이스 작업의 속도를 저하시키지는 않습니다. 그러나 NONBLOCKED 이벤트 모니터는 활발히 사용 중인 시스템에서 데이터가 유실될 수 있습니다.

APPEND

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 새 이벤트 데이터를 데이터 파일의 기존 스트림에 추가하도록 지정합니다. 이벤트 모니터가 다시 활성화될 경우, 작동이 중지되지 않았던 것처럼 이벤트 파일에 다시 기록하기 시작합니다. APPEND는 디폴트 옵션입니다.

새로 작성된 이벤트 모니터가 이벤트 데이터를 기록할 디렉토리에 기존의 이벤트 데이터가 있을 경우, APPEND 옵션은 CREATE EVENT MONITOR 수행시 적용되지 않습니다.

REPLACE

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 모든 이벤트 파일을 지우고 00000000.evt 파일에 데이터를 기록하기 시작하도록 지정합니다.

MANUALSTART

이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화해야 한다는 것을 지정합니다. MANUALSTART 이벤트 모니터가 활성화된 다음, SET EVENT MONITOR STATE문을 사용하거나 인스턴스를 중지해야 비활성화될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 제외한 비WLM 이벤트 모니터의 디폴트 동작입니다.

AUTOSTART

이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화될 때마다 이벤트 모니터가 자동으로 활성화되도록 지정합니다. WLM 이벤트 모니터 및 DB2DETAILDEADLOCK 이벤트 모니터의 디폴트 동작입니다.

ON DBPARTITIONNUM *db-partition-number*

파일이나 파이프 이벤트 모니터를 실행할 데이터베이스 파티션을 지정합니다. 모니터링 범위가 LOCAL로 정의된 경우, 데이터는 지정된 파티션에서만 수집됩니다. 모니터 범위가 GLOBAL로 정의된 경우, 모든 데이터베이스 파티션이 데이터를 수집하고 지정된 번호로 데이터베이스 파티션에 보고합니다. I/O 구성요소는 지정된 데이터베이스 파티션에서 실제로 실행되며 지정된 파일 또는 파이프에 레코드를 씁니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다. 파티션된 데이터베이스 환경의 경우, 테이블에 기록 이벤트 모니터는 목표 테이블에 대한 테이블 스페이스가 정의된 모든 데이터베이스 파티션에서 이벤트를 실행하고 기록합니다.

GLOBAL절이 지정되지 않으면 응용프로그램에 대해 현재 연결된 데이터베이스 파티션 번호가 사용됩니다.

LOCAL

이벤트 모니터를 실행 중인 데이터베이스 파티션에만 보고합니다. 데이터베이스 활동에 대한 부분적 추적을 제공합니다. 이는 디폴트값입니다.

이 절은 테이블 이벤트 모니터에 대해 유효하지 않습니다.

규칙

- THRESHOLD VIOLATIONS 이벤트 유형은 특정 이벤트 모니터 정의에 있는 다른 이벤트 유형과 결합될 수 없습니다.

주

- 이벤트 모니터 정의는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 기록됩니다. 이벤트 자체는 SYSCAT.EVENTS 카탈로그 뷰에 기록됩니다. 목표 테이블의 이름은 SYSCAT.EVENTTABLES 카탈로그 뷰에 기록됩니다.
- BUFFERSIZE 매개변수는 STMT, STMT_HISTORY, DATA_VALUE 및 DETAILED_DLCONN 이벤트의 *크기*를 제한합니다. STMT 또는 STMT_HISTORY 이벤트의 *크기*가 버퍼에 맞지 않으면 명령문 텍스트를 절단하여 절단됩니다. DETAILED_DLCONN 이벤트의 *크기*가 버퍼에 맞지 않으면 잠금을 제거하여 해당 이벤트가 절단됩니다. 그래도 맞지 않으면 명령문 텍스트를 절단합니다. DATA_VAL 이벤트의 *크기*가 버퍼에 맞지 않으면 데이터 값이 절단됩니다.
- 이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화되지 않은 경우, 그 다음 데이터베이스 파티션이 활성화되면 이벤트 모니터가 활성화됩니다.
- 이벤트 모니터는 일단 활성화되면 이벤트 모니터가 명시적으로 비활성화되거나 인스턴스 처리가 되풀이될 때까지 자동 시작 이벤트 모니터처럼 작동합니다. 즉 데이터베이스 파티션이 비활성화되었을 때 이벤트 모니터가 활성화되고 그후에 그 데이터베이스 파티션이 다시 활성화되면, 이벤트 모니터도 명시적으로 재활성화됩니다.
- *테이블 이벤트 모니터에 기록:*: 일반 메모:
  - 모든 목표 테이블은 CREATE EVENT MONITOR문을 실행할 때 작성됩니다.
  - 어떤 이유로든 테이블 작성에 실패하면 응용프로그램에 오류가 전달되고 CREATE EVENT MONITOR문이 실패합니다.
  - 목표 테이블은 한 개의 이벤트 모니터만 사용할 수 있습니다. CREATE EVENT MONITOR를 처리하는 동안 다른 이벤트 모니터에서 사용하도록 목표 테이블이 이미 정의되어 있음이 발견되면 CREATE EVENT MONITOR문이 실패하고 응

용프로그램에 오류가 전달됩니다. 테이블 이름이 SYSCAT.EVENTTABLES 카탈로그 뷰에 있는 값과 일치하면 다른 이벤트 모니터가 테이블을 사용하도록 테이블이 정의됩니다.

- CREATE EVENT MONITOR를 처리하는 동안 테이블은 있지만 다른 이벤트 모니터에서 사용하도록 테이블이 정의되어 있지 않으면 테이블이 작성되지 않고 처리가 계속됩니다. 응용프로그램에 경고가 전달됩니다.
- 모든 테이블 스페이스는 CREATE EVENT MONITOR문이 실행되기 전에 존재해야 합니다. CREATE EVENT MONITOR문은 테이블 스페이스를 작성하지 않습니다.
- LOCAL 키워드와 GLOBAL 키워드를 지정할 경우 무시됩니다. WRITE TO TABLE 이벤트 모니터의 경우에는 이벤트 모니터 출력 프로세스나 스레드가 이 인스턴스의 각 데이터베이스 파티션에서 시작되고 이러한 각 프로세스는 프로세스가 실행되는 데이터베이스 파티션에 대해서만 데이터를 보고합니다.
- 테이블에 기록 이벤트 모니터는 다음과 같은 플랫 모니터 로그 파일이나 파이프 형식의 이벤트 유형은 기록하지 않습니다.
  - LOG_STREAM_HEADER
  - LOG_HEADER
  - DB_HEADER(db_name 및 db_path 요소는 기록되지 않습니다. conn_time 요소는 CONTROL에 기록됩니다.)
- 파티션된 데이터베이스 환경의 경우, 데이터는 목표 테이블의 테이블 스페이스가 존재하는 데이터베이스 파티션에서만 목표 테이블에 기록됩니다. 일부 데이터베이스 파티션에 목표 테이블에 대한 테이블 스페이스가 존재하지 않을 경우, 해당 목표 테이블에 대한 데이터가 무시됩니다. 이 때문에 사용자는 특정 데이터베이스 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터링할 데이터베이스 파티션의 서브세트를 선택할 수 있습니다.

  파티션된 데이터베이스 환경의 경우, 일부 목표 테이블이 데이터베이스 파티션에 상주하지 않지만 다른 목표 테이블이 동일한 데이터베이스 파티션에 상주하면 데이터베이스 파티션에 있는 목표 테이블에 대한 데이터만 기록됩니다.
- 사용자는 모든 목표 테이블을 직접 프룬해야 합니다.

테이블 컬럼:

- 테이블의 컬럼 이름은 이벤트 모니터의 요소 ID와 일치합니다. sqlm_time(경과 시간) 유형의 모니터 변수는 예외입니다. 이러한 유형에 대한 컬럼 이름은 TYPE_NAME_S와 TYPE_NAME_MS이고, 이 두 이름은 각각 초와 마이크로초로 시간을 저장하는 컬럼을 나타냅니다. 일치하는 목표 테이블 컬럼이 없는 이벤트 모니터 요소는 무시됩니다.
- 그룹에 대한 전체 요소 목록을 포함하는 CREATE EVENT MONITOR 명령을 빌드하려면 db2evtbl 명령을 사용하십시오.

– 모니터 요소에 사용되는 컬럼 유형은 다음 맵핑과 상호 관련이 있습니다.

```
SQLM_TYPE_STRING                    CHAR[n], VARCHAR[n] or CLOB(n)
                                   (이벤트 모니터 레코드의 데이터가
                        n 바이트를 초과할 경우,
                                    데이터는 절단됩니다.)
SQLM_TYPE_U8BIT 및 SQLM_TYPE_8BIT   SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_16BIT 및 SQLM_TYPE_U16BIT SMALLINT, INTEGER 또는 BIGINT
SQLM_TYPE_32BIT 및 SQLM_TYPE_U32BIT INTEGER 또는 BIGINT
SQLM_TYPE_U64BIT 및 SQLM_TYPE_64BIT  BIGINT
sqlm_timestamp                      TIMESTAMP
sqlm_time(경과 시간)                 BIGINT
sqlca:
    sqlerrmc                          VARCHAR[72]
    sqlstate                          CHAR[5]
    sqlwarn                           CHAR[11]
    기타 파일                          INTEGER 또는 BIGINT
```

– 컬럼은 NOT NULL로 정의됩니다.

– CLOB 컬럼이 있는 테이블은 VARCHAR 컬럼이 있는 테이블에 비해 성능이 떨어지므로 STMT *evm-group* 값(또는 DEADLOCKS WITH DETAILS 이벤트 유형을 사용할 경우에는 DLCONN *evm-group* 값)을 지정할 때 TRUNC 키워드를 사용하는 것이 좋습니다.

– 다른 목표 테이블과 달리 CONTROL 테이블의 컬럼은 모니터 요소 ID와 일치하지 않습니다. 컬럼은 다음과 같이 정의됩니다.

```
컬럼 이름         데이터 유형   널(NULL)  설명
                              입력 가능
-----------      ---------    --------  -----------
PARTITION_KEY    INTEGER      N         분산 키
                                        (파티션된 데이터베이스만 해당)
PARTITION_NUMBER INTEGER      N         데이터베이스 파티션 번호
                                        (파티션된 데이터베이스만 해당)
EVMONNAME        VARCHAR(128) N         이벤트 모니터 이름
MESSAGE          VARCHAR(128) N         MESSAGE_TIME 컬럼의
                                        등록 정보를 기술합니다.
                                        다음 중 하나가
                                        될 수 있습니다.
                                        - FIRST_CONNECT(데이터베이스
                                        활성화 이후 처음
                                        데이터베이스에 연결한 시간)
                                        - EVMON_START(EVMONNAME에
      나열된 이벤트 모니터가
                                        시작된 시간)
                                        - OVERFLOWS:n(버퍼 오버플로우로 인해
                                          n개의 레코드를
      버린다는 것을 나타냄)
                                        - LAST_DROPPED_RECORD
                                          (오버플로우가 발생된
        마지막 시간)
MESSAGE_TIME   TIMESTAMP      N         시간소인
```

– 파티션된 데이터베이스 환경에서 각 테이블의 첫 번째 컬럼의 이름은 PARTITION_KEY이고, NOT NULL이며 INTEGER 유형입니다. 이 컬럼은 테이블의 분산 키로 사용됩니다. 각 이벤트 모니터 프로세스에서 프로세스가 실행되는 데이터베이스 파티션에 데이터를 삽입할 수 있도록 이 컬럼의 값이 선택됩니다. 즉, 삽입 조작은 이벤트 모니터 프로세스가 실행되는 데이터베이스 파티션에서 로컬로 수행됩니다. 모든 데이터베이스 파티션에서 PARTITION_KEY 필드는 같은 값을 포함합니다. 즉 데이터베이스 파티션이 삭제되고 데이터 재분배가 수행될 경우 삭제된 데이터베이스 파티션에 있는 모든 데이터는 고르게 분배되지 않고 다른 데이터베이스 파티션으로 이동됩니다. 따라서 데이터베이스 파티션을 제거하기 전에 제거할 데이터베이스 파티션에 있는 모든 테이블 행을 삭제하는 것이 좋습니다.

- 파티션된 데이터베이스 환경에서는 각 테이블에 대해 PARTITION_NUMBER 컬럼을 정의할 수 있습니다. 이 컬럼은 INTEGER 유형인 NOT NULL입니다. 이 컬럼에는 데이터가 삽입된 데이터베이스 파티션의 번호가 포함합니다. PARTITION_KEY 컬럼과 달리 PARTITION_NUMBER 컬럼은 필수가 아닙니다. 파티션되지 않은 데이터베이스 환경에서는 PARTITION_NUMBER 컬럼을 정의할 수 없습니다.

테이블 속성

- 디폴트 테이블 속성이 사용됩니다. 테이블을 작성할 때 분산 키(파티션된 데이터베이스에만 해당)를 제외하고 추가 옵션은 지정하지 않습니다.
- 테이블에 대한 인덱스를 작성할 수 있습니다.
- VOLATILE, RI, 트리거 및 제한조건 등과 같은 추가 테이블 속성을 추가할 수는 있지만 이벤트 모니터 프로세스(또는 스레드)는 이런 속성을 무시합니다.
- ″not logged initially″를 테이블 속성으로 추가하면 처음 COMMIT를 실행할 때 이 속성이 해제되고 다시 설정되지 않습니다.

이벤트 모니터 활성화

- 이벤트 모니터가 활성화되면 SYSCAT.EVENTTABLES 카탈로그 뷰에서 모든 목표 테이블 이름이 검색됩니다.
- 파티션된 데이터베이스 환경에서는 인스턴스의 모든 데이터베이스 파티션에서 활성화 처리가 이루어집니다. 특정 데이터베이스 파티션에서는 활성화 처리가 각 목표 테이블의 테이블 스페이스 및 데이터베이스 파티션 그룹을 판별합니다. 이벤트 모니터는 데이터베이스 파티션에 최소한 하나의 목표 테이블이 존재할 경우에만 데이터베이스 파티션에 활성화됩니다. 또한 데이터베이스 파티션에 일부 목표 테이블이 없을 경우, 해당 테이블에 대한 데이터가 런타임 처리시 삭제되도록 해당 목표 테이블에 대한 플래그가 표시됩니다.
- 이벤트 모니터가 활성화될 때 목표 테이블이 없거나, 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주하지 않을 경우에도 활성화가 계속됩니다. 그러나 이벤트 모니터가 활성화될 때 목표 테이블이 있거나 파티션된 데이터베이스 환경에서 테이블 스페이스가 데이터베이스 파티션에 상주할 경우 이 테이블에 삽입될 데이터는 무시됩니다.
- 활성화 프로세스는 각 목표 테이블의 유효성을 확인합니다. 유효성 확인이 실패하면 이벤트 모니터가 활성화되지 않고 관리 로그에 메시지가 기록됩니다.
- 파티션된 데이터베이스 환경에서 활성화하는 동안에는 FIRST_CONNECT와 EVMON_START에 대한 CONTROL 테이블의 행만 카탈로그 데이터베이스 파티션에 삽입됩니다. 이 때 제어 테이블에 대한 테이블 스페이스가 카탈로그 데이터베이스 파티션에 있어야 합니다. 테이블 스페이스가 카탈로그 데이터베이스 파티션에 없으면 이런 삽입 조작이 수행되지 않습니다.

– 파티션된 데이터베이스 환경에서는 테이블에 기록 이벤트 모니터가 활성화될 때 파티션이 활성화되어 있지 않으면, 다음에 파티션이 활성화될 때 이벤트 모니터가 활성화됩니다.

런타임

– 이벤트 모니터가 DATAACCESS 권한으로 실행됩니다.
– 이벤트 모니터가 활성화되어 있는 동안 목표 테이블에 대한 삽입 조작이 실패하면 다음 작업이 수행됩니다.
  - 언커미트된 변경사항을 롤백합니다.
  - 관리 로그에 메시지가 기록됩니다.
  - 이벤트 모니터가 비활성화됩니다.
– 이벤트 모니터가 활성화되어 있을 경우 이벤트 모니터 버퍼에 대한 처리가 완료되면 로컬 COMMIT가 수행됩니다.
– 파티션된 데이터베이스 환경에서는 65,535바이트까지 가능한 실제 명령문 텍스트는 응용프로그램 코디네이터 데이터베이스 파티션에서 실행되는 이벤트 모니터에 의해 STMT 테이블이나 DLCONN 테이블에 저장됩니다. 다른 데이터베이스 파티션에서는 이 값이 0입니다.
– 파티션되지 않은 데이터베이스 환경에서는 마지막 응용프로그램이 종료되고 데이터베이스가 명시적으로 활성화되지 않았을 때 모든 테이블에 기록 이벤트 모니터가 비활성화됩니다. 파티션된 데이터베이스 환경에서는 카탈로그 파티션이 비활성화될 때 테이블에 기록 이벤트 모니터가 비활성화됩니다.
– DROP EVENT MONITOR문은 목표 테이블을 삭제하지 않습니다.
– 테이블에 기록 이벤트 모니터가 활성화될 때마다, 이벤트 모니터가 사용 중인 동안 수정되지 않도록 IN 테이블이 각 목표 테이블을 잠급니다. 이벤트 모니터가 사용 중인 동안 테이블 잠금은 모든 테이블에서 유지됩니다. 독점 액세스가 목표 테이블에 필요하면(예를 들어, 유틸리티를 실행하려고 할 때) 이러한 액세스를 시도하기 전에 먼저 이벤트 모니터를 비활성화하여 테이블 잠금을 해제합니다.

• *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  – DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  – 쉼표를 사용하여 *wlm-target-table-info*절에서 여러 옵션을 구분할 수 있습니다.

예

예 1: SMITHPAY라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 데이터베이스에 대해, 그리고 JSMITH 권한 부여 ID가 소유하는 PAYROLL 응용프로그램에서 수행되는 SQL문에 대해 이벤트 데이터를 수집합니다. 데이터는 절대 경로 /home/jsmith/event/smithpay/에 추가됩니다. 최대 25개의 파일이 작성됩니다. 각 파일은 최대 1 024 4K 페이지가 됩니다. 파일 I/O는 비블록화됩니다.

```
CREATE EVENT MONITOR SMITHPAY
  FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
  WRITE TO FILE '/home/jsmith/event/smithpay'
  MAXFILES 25
  MAXFILESIZE 1024
  NONBLOCKED
  APPEND
```

예 2: DEADLOCKS_EVTS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 교착 상태 이벤트를 수집한 다음 이를 상대 경로 DLOCKS에 기록합니다. 한 개의 파일이 작성되며, 최대 파일 크기는 없습니다. 이벤트 모니터가 활성화될 때마다, 이벤트 데이터를 00000000.evt 파일(있는 경우)에 추가합니다. 이벤트 모니터는 데이터베이스가 시작될 때마다 시작됩니다. I/0는 디폴트로 블록화됩니다.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
  FOR DEADLOCKS
       WRITE TO FILE 'DLOCKS'
  MAXFILES 1
  MAXFILESIZE NONE
  AUTOSTART
```

예 3: DB_APPLS라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 연결 이벤트를 수집하고 데이터를 Named Pipe /home/jsmith/applpipe에 기록합니다.

```
CREATE EVENT MONITOR DB_APPLS
  FOR CONNECTIONS
  WRITE TO PIPE '/home/jsmith/applpipe'
```

예 4: 파티션된 데이터베이스 환경을 가정하고 FOO라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트를 수집하고 수집한 이벤트를 다음 파생 이름을 사용하여 SQL 테이블에 기록합니다.

- CONNHEADER_FOO
- STMT_FOO
- SUBSECTION_FOO
- CONTROL_FOO

테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 모든 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 모든 테이블에는 테이블 그룹에 대한 모든 요소가 포함됩니다. 즉, 요소 이름과 같은 이름을 갖는 컬럼이 정의됩니다.

```
CREATE EVENT MONITOR FOO
  FOR STATEMENTS
  WRITE TO TABLE
```

예 5: 파티션된 데이터베이스 환경을 가정하고 BAR라는 이벤트 모니터를 작성하십시오. 이 이벤트 모니터는 SQL문 이벤트와 트랜잭션 이벤트를 수집하고 수집한 이벤트를 다음과 같이 테이블에 기록합니다.

- STMT 그룹의 데이터는 모두 MYDEPT.MYSTMTINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. ROWS_READ, ROWS_WRITTEN 및 STMT_TEXT 요소에 대해서만 컬럼을 작성하고, 그룹의 다른 요소는 버립니다.
- SUBSECTION 그룹의 데이터는 모두 MYDEPT.MYSUBSECTIONINFO 테이블에 기록됩니다. 이 테이블은 테이블 스페이스 MYTABLESPACE에 작성됩니다. 이 테이블에는 START_TIME, STOP_TIME 및 PARTIAL_RECORD를 제외한 모든 컬럼이 포함됩니다.
- XACT 그룹의 데이터는 모두 XACT_BAR 테이블에 기록됩니다. 테이블 스페이스 정보를 제공하지 않았으므로 IN *tablespace-name*절에 설명된 규칙에 따라 테이블이 시스템에서 선택한 테이블 스페이스에 작성됩니다. 이 테이블에는 XACT 그룹에 있는 모든 요소가 포함됩니다.
- connheader나 control에 대해서는 테이블이 작성되지 않습니다. 따라서 이 두 그룹에 대한 데이터는 모두 버립니다.

```
CREATE EVENT MONITOR BAR
  FOR STATEMENTS, TRANSACTIONS
  WRITE TO TABLE
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
  XACT
```

CREATE EVENT MONITOR(작업 단위)

CREATE EVENT MONITOR(작업 단위)문은 작업 단위(UOW)가 완료할 때 이벤트를 기록할 이벤트 모니터를 작성합니다.

호출

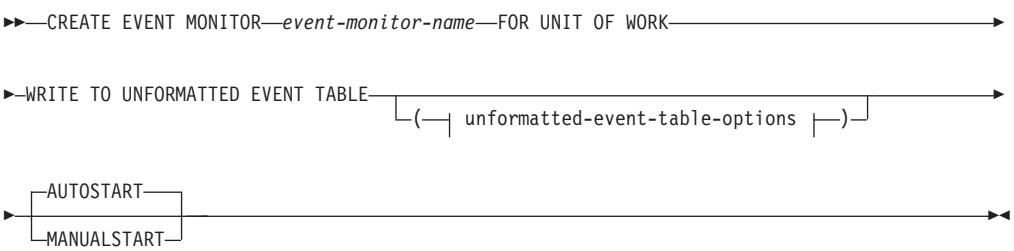
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

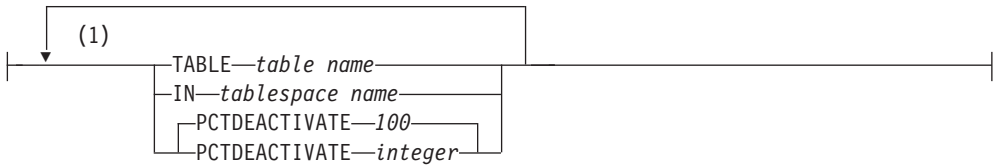
명령문의 권한 부여 ID는 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- DBADM 권한
- SQLADM 권한

구문



unformatted-event-table-options:



주:

1 각 비형식화 이벤트 테이블 옵션은 최대 1번 지정할 수 있습니다(SQLSTATE 42613).

설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *event-monitor-name*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 이벤트의 유형을 소개합니다.

UNIT OF WORK

작업 단위(UOW)가 완료될 때마다(즉, 커미트 또는 롤백이 있을 때마다) 이 수동 이벤트 모니터가 이벤트를 기록하도록 지정합니다.

작업 단위 이벤트 모니터가 작성된다고 해서 작업 단위 데이터가 즉시 수집되는 것은 아닙니다. 관심있는 실제 작업 단위 이벤트는 워크로드 레벨에서 제어됩니다.

WRITE TO

데이터의 목표를 지정합니다.

UNFORMATTED EVENT TABLE

이벤트 모니터의 목표가 비형식화 이벤트 테이블임을 지정합니다. 비형식화 이벤트 테이블은 수집된 작업 단위(UOW) 이벤트 모니터 데이터를 저장하는 데 사용됩니다. 모든 삽입 트랜잭션이 이 데이터베이스 테이블에 로그되지 않습니다. 데이터는 인라인되고 로그되지 않는 BLOB 컬럼에 원래 2진 형식으로 저장됩니다. BLOB 컬럼은 다른 유형의 여러 2진 레코드를 포함할 수 있습니다. BLOB 컬럼의 데이터는 읽을 수 있는 형식이 아니며 db2evmonfmt Java 기반 도구,
EVMON_FORMAT_UE_TO_XML 테이블 함수 또는
EVMON_FORMAT_UE_TO_TABLES 프로시저를 사용하여 XML 문서 또는 관계형 테이블 같이 이용 가능한 형식으로 변환되어야 합니다.

(unformatted-event-table-options)

비형식화 이벤트 테이블을 식별합니다. unformatted-event-table-options 값을 지정하지 않으면 CREATE EVENT MONITOR 처리가 다음과 같이 진행됩니다.

- 파생된 테이블 이름이 사용됩니다(아래에 설명되어 있음).
- 디폴트 테이블 스페이스가 선택됩니다(아래에 설명되어 있음).
- PCTDEACTIVATE가 100으로 설정됩니다.

TABLE *table-name*

비형식화 이벤트 테이블의 이름을 지정합니다. 이름을 제공하지 않는 경우 규정되지 않은 이름은 *event-monitor-name*과 같습니다. 즉, 비형식화 이벤트 테이블은 이벤트 모니터 이후에 이름이 지정됩니다.

다음을 주의하십시오.

- 비형식화 이벤트 테이블은 CREATE EVENT MONITOR FOR UNIT OF WORK문이 실행될 때 작성됩니다(아직 존재하지 않는 경우).

- CREATE EVENT MONITOR FOR UNIT OF WORK 처리 중에, 비형식화 이벤트 테이블이 다른 이벤트 모니터가 사용하기 위해 이미 정의된 것으로 확인되는 경우 CREATE EVENT MONITOR FOR UNIT OF WORK문은 실패하며 오류가 응용프로그램으로 다시 전달됩니다. 비형식화 이벤트 테이블 이름이 SYSCAT.EVENTTABLES 카탈로그 뷰에 있는 값과 일치하면 다른 이벤트 모니터가 사용하도록 비형식화 이벤트 테이블이 정의됩니다. 비형식화 이벤트 테이블이 존재하고 다른 이벤트 모니터가 사용하도록 정의되지 않는 경우, 테이블은 작성되지 않고 다른 모든 테이블 unformatted-event-table-options 매개변수는 무시되며 처리는 계속됩니다. 응용프로그램에 경고가 전달됩니다.
- 이벤트 모니터를 삭제해도 비형식화 이벤트 테이블은 삭제되지 않습니다. 연관된 비형식화 이벤트 테이블은 이벤트 모니터가 삭제된 후에 명시적으로 삭제해야 합니다.
- 비형식화 이벤트 테이블은 수동으로 프룬되어야 합니다.

IN *tablespace-name*

비형식화 이벤트 테이블이 작성될 테이블 스페이스를 지정합니다. CREATE EVENT MONITOR FOR UNIT OR WORK문은 테이블 스페이스를 작성하지 않습니다.

테이블 스페이스 이름을 제공하지 않으면 다음과 같이 테이블 스페이스가 선택됩니다.

```
IF table space IBMDEFAULTGROUP over which the user
        has USE privilege exists
   THEN choose it
 ELSE IF table space over which the user
        has USE privilege exists
   THEN choose it
 ELSE return an error (SQLSTATE 42727)
```

PCTDEACTIVATE *integer*

비형식화 이벤트 테이블이 DMS 테이블 스페이스에 작성되는 경우 PCTDEACTIVATE 매개변수는 이벤트 모니터가 자동으로 비활성화되기 전에 테이블 스페이스가 채워지는 정도를 지정합니다. 0부터 100까지의 값을 지정할 수 있으며 이 값은 백분율을 나타냅니다. 디폴트값은 100입니다. 100은 테이블 스페이스가 완전히 찼을 때 이벤트 모니터가 비활성화됨을 의미합니다. 테이블 스페이스에서 자동 크기 조정이 가능한 경우 PCTDEACTIVATE를 100으로 설정하도록 하십시오. 이 옵션은 SMS 테이블 스페이스의 경우 무시됩니다.

AUTOSTART

이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화될 때마다 이벤트 모니터

가 자동으로 활성화되도록 지정합니다. WLM 이벤트 모니터 및 DB2DETAILDEADLOCK 이벤트 모니터의 디폴트 동작입니다.

MANUALSTART
이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화해야 한다는 것을 지정합니다. MANUALSTART 이벤트 모니터가 활성화된 다음, SET EVENT MONITOR STATE문을 사용하거나 인스턴스를 중지해야 비활성화될 수 있습니다. DB2DETAILDEADLOCK 이벤트 모니터를 제외한 비WLM 이벤트 모니터의 디폴트 동작입니다.

주

- 이벤트 데이터는 비형식화 이벤트 테이블에서 인라인된 BLOB 데이터 컬럼에 삽입됩니다. 일반적으로 BLOB 데이터는 개별 LOB 테이블 스페이스에 저장되며 결과적으로 추가 성능 오버헤드를 겪을 수 있습니다. 기본 테이블의 데이터 페이지에 인라인될 때 BLOB 데이터는 이 오버헤드를 겪지 않습니다. BLOB 데이터의 *크기*가 테이블 스페이스 페이지 *크기*에서 레코드 접두부를 뺀 것보다 작은 경우 DB2 데이터베이스 관리 프로그램이 비형식화 이벤트 테이블 레코드의 BLOB 데이터 부분을 자동으로 인라인합니다. 그러므로 높은 효율 및 응용프로그램 처리량을 달성하려면 32KB 테이블 스페이스 및 연관된 버퍼 풀을 포함하여 가능한 큰 테이블 스페이스에 이벤트 모니터를 작성할 것을 권장합니다.

  예 : 잠금 이벤트 모니터는 현재 다음 두 레코드 유형을 갖습니다.

  – 응용프로그램 정보 레코드

  – 응용프로그램 활동 레코드

  응용프로그램 정보 레코드 = 최대 *크기* 3.5KB

  응용프로그램 활동 레코드 = 3KB + SQL문 텍스트 *크기*(여기서 SQL문 텍스트 *크기*는 최대 2MB임)

  응용프로그램 정보 레코드는 매우 작으며 4KB 페이지 *크기*가 사용되는 중에는 항상 인라인되어야 합니다. 응용프로그램 활동 레코드는 다음 공식을 기반으로 인라인됩니다.

```
Application Activity Record < inline length (Pagesize - overhead non-LOB columns (0.5KB))
   3KB + SQL statement text < inline length (Pagesize - overhead non-LOB columns (0.5KB))

        SQL statement text < Pagesize - nonLOB overhead (1K) - 3KB
        SQL statement text < 16KB - 1KB – 3KB
                          < 12KB
```

  그러므로 16KB 페이지 *크기*를 사용할 때 잠금 이벤트 모니터 레코드는 캡처되는 SQL문의 *크기*가 12KB 미만인 경우에만 인라인됩니다.

- 데이터베이스별로 작업 단위(UOW) 이벤트 모니터를 하나만 작성하고 동일한 데이터베이스에 여러 개의 작업 단위 이벤트 모니터를 작성하지 마십시오.
- 파티션된 데이터베이스 환경에서, 데이터는 테이블 스페이스가 존재하는 데이터베이스 파티션에서만 목표 비형식화 이벤트 테이블에 기록됩니다. 일부 데이터베이스 파

티션에 목표 비형식화 이벤트 테이블에 대한 테이블 스페이스가 존재하지 않을 경우, 해당되는 목표 비형식화 이벤트 테이블에 대한 데이터가 무시됩니다. 이 때문에 사용자는 특정 데이터베이스 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터링 대상으로 선택할 데이터베이스 파티션의 서브세트를 선택할 수 있습니다.

- 다중 구성원 환경에서 데이터는 작업이 작업 단위(UOW) 안에서 발생하는 구성원의 목표 비형식화 이벤트 테이블에만 기록됩니다.
- 파티션된 데이터베이스 환경에서, 일부 목표 비형식화 이벤트 테이블이 데이터베이스 파티션에 상주하지 않지만 다른 목표 비형식화 이벤트 테이블이 동일한 데이터베이스 파티션에 상주하면 데이터베이스 파티션에 있는 목표 비형식화 이벤트 테이블에 대한 데이터만 기록됩니다.
- 작업 단위(UOW) 이벤트 모니터는 단위 또는 작업 이벤트 모니터 스위치의 영향을 받지 않습니다. 작업 단위(UOW) 이벤트 모니터 스위치는 작업 단위 이벤트 모니터가 작성될 때 변경되지 않으며 단위 또는 작업 이벤트 모니터의 컨텐츠는 작업 단위 이벤트 모니터 스위치의 변경사항에 영향을 받지 않습니다.
- FLUSH EVENT MONITOR문은 이벤트가 UOW 이벤트 모니터에 기록되도록 하지 않습니다.
- 작업 단위 이벤트 모니터를 작성해도 이벤트가 이벤트 모니터에 기록되지 않습니다. 작업 단위 이벤트 모니터는 SET EVENT MONITOR STATE를 사용하여 활성화되어야 하며, 작업 단위 데이터가 적합한 워크로드를 변경하여 COLLECT UNIT OF WORK DATA를 지정하거나 **UOW_DATA DB** 구성 매개변수를 NONE이 아닌 값으로 설정하여 수집되어야 합니다.

예:

예 1: 이 예는 작성 데이터베이스에 대해 발생하는 작업 단위 이벤트를 수집하지만 디폴트 비형식화 이벤트 테이블 UOWEVMON에 데이터를 기록할 작업 단위 이벤트 모니터 UOWEVMON을 작성합니다.

```
CREATE EVENT MONITOR UOWEVMON
  FOR UNIT OF WORK
  WRITE TO UNFORMATTED EVENT TABLE
```

예 2: 이 예는 작성 데이터베이스에 대해 발생하는 작업 단위 이벤트를 수집하여 비형식화 이벤트 테이블 GREG.UOWEVENTS에서 저장할 작업 단위 이벤트 모니터 UOWEVMON을 작성합니다.

```
CREATE EVENT MONITOR UOWEVMON
  FOR UNIT OF WORK
  WRITE TO UNFORMATTED EVENT TABLE (TABLE GREG.UOWEVENTS)
```

예 3: 이 예는 작성 데이터베이스에 대해 발생하는 작업 단위 이벤트를 수집하여 테이블 스페이스 APPSPACE의 비형식화 이벤트 테이블 GREG.UOWEVENTS에서 저장할 작업 단위 이벤트 모니터 UOWEVMON을 작성합니다. 이벤트 모니터는 테이블 스페이스가 85% 찰 때 비활성화됩니다.

```
CREATE EVENT MONITOR UOWEVMON
FOR UNIT OF WORK
WRITE TO UNFORMATTED EVENT TABLE
(TABLE GREG.UOWEVENTS IN APPSPACE PCTDEACTIVATE 85)
```

CREATE FUNCTION

CREATE FUNCTION 문은 현재 서버에 사용자 정의 함수나 함수 템플리트를 등록 또는 정의하는데 사용됩니다.

이 명령문을 사용하여 다른 5가지 유형의 함수를 작성할 수 있습니다. 각 유형에 대해서는 별도로 설명합니다.

- 외부 스칼라. 함수가 프로그래밍 언어로 작성되고 스칼라 값을 리턴합니다. 외부 실행 파일은 다양한 함수 속성과 함께 데이터베이스에 등록됩니다.
- 외부 테이블. 함수가 프로그래밍 언어로 작성되고 전체 테이블을 리턴합니다. 외부 실행 파일은 다양한 함수 속성과 함께 데이터베이스에 등록됩니다.
- OLE DB 외부 테이블. OLE DB Provider로부터 데이터에 액세스할 수 있도록 사용자 정의 OLE DB 외부 테이블 함수가 데이터베이스에 등록됩니다.
- 소스 또는 템플리트. 소스 함수는 이미 데이터베이스에 등록된 다른 함수(내장, 외부, SQL 또는 소스)를 호출하여 구현됩니다.

  리턴될 값 유형을 정의하지만 실행 가능 코드를 포함하지 않는 *함수 템플리트*라고 하는 부분 함수를 작성할 수 있습니다. 사용자는 이를 페더레이티드 시스템 내에 있는 데이터 소스 함수에 맵핑하여 페더레이티드 데이터베이스에서 데이터 소스 함수가 호출될 수 있도록 합니다. 함수 템플리트는 페더레이티드 서버로 지정된 응용프로그램 서버(AS)에만 등록할 수 있습니다.
- SQL 스칼라, 테이블 또는 행. 함수 내용은 SQL로 작성되고 데이터베이스에서의 등록과 함께 정의됩니다. 스칼라 값, 테이블 또는 단일 행을 리턴합니다.

CREATE FUNCTION(외부 스칼라)

CREATE FUNCTION(외부 스칼라)문은 현재 서버에 사용자 정의 외부 스칼라 함수를 등록하는 데 사용됩니다. *스칼라 함수*는 호출될 때마다 단일 값을 리턴하며, 일반적으로 SQL 표현식이 유효하면 이 함수도 유효합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스의 CREATE_EXTERNAL_ROUTINE 권한은 다음 중 최소한 하나를 포함해야 합니다.
  - 함수의 스키마 이름이 기존의 스키마를 가리키지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 함수의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

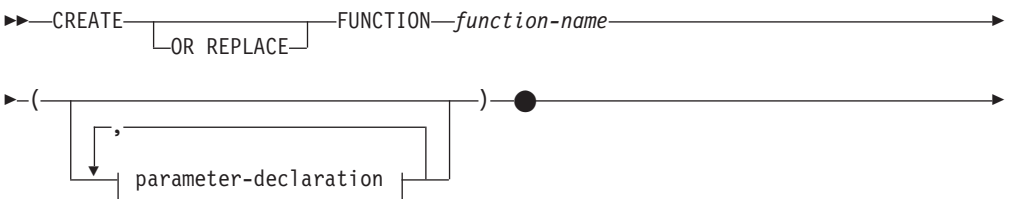
비분리 함수를 작성하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

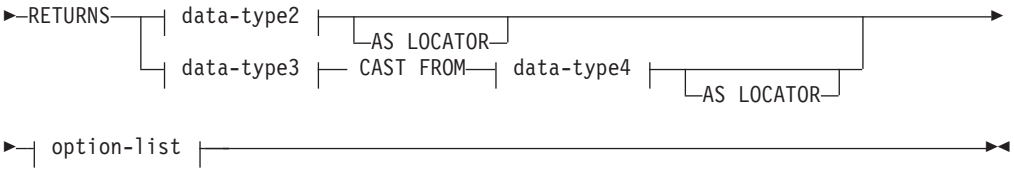
- 데이터베이스에서의 CREATE_NOT_FENCED_ROUTINE 권한
- DBADM 권한

분리 함수를 작성하려면, 추가 권한이나 특권이 필요하지 않습니다.

기존 함수를 교체하려면, 명령문의 권한 부여 ID가 기존 함수의 소유자여야 합니다(SQLSTATE 42501).

구문

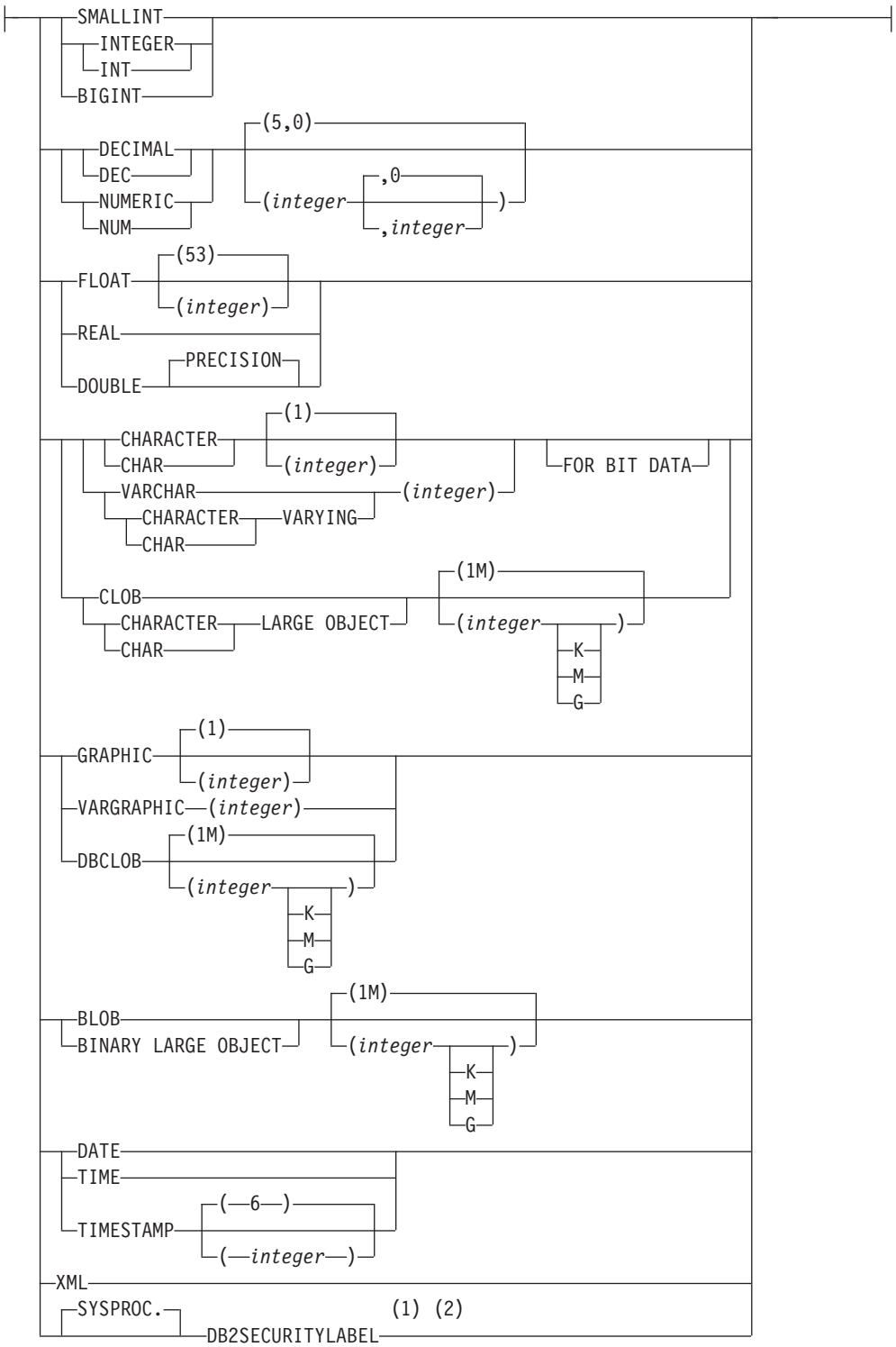




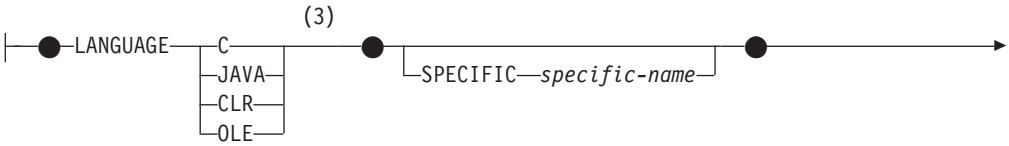
parameter-declaration:

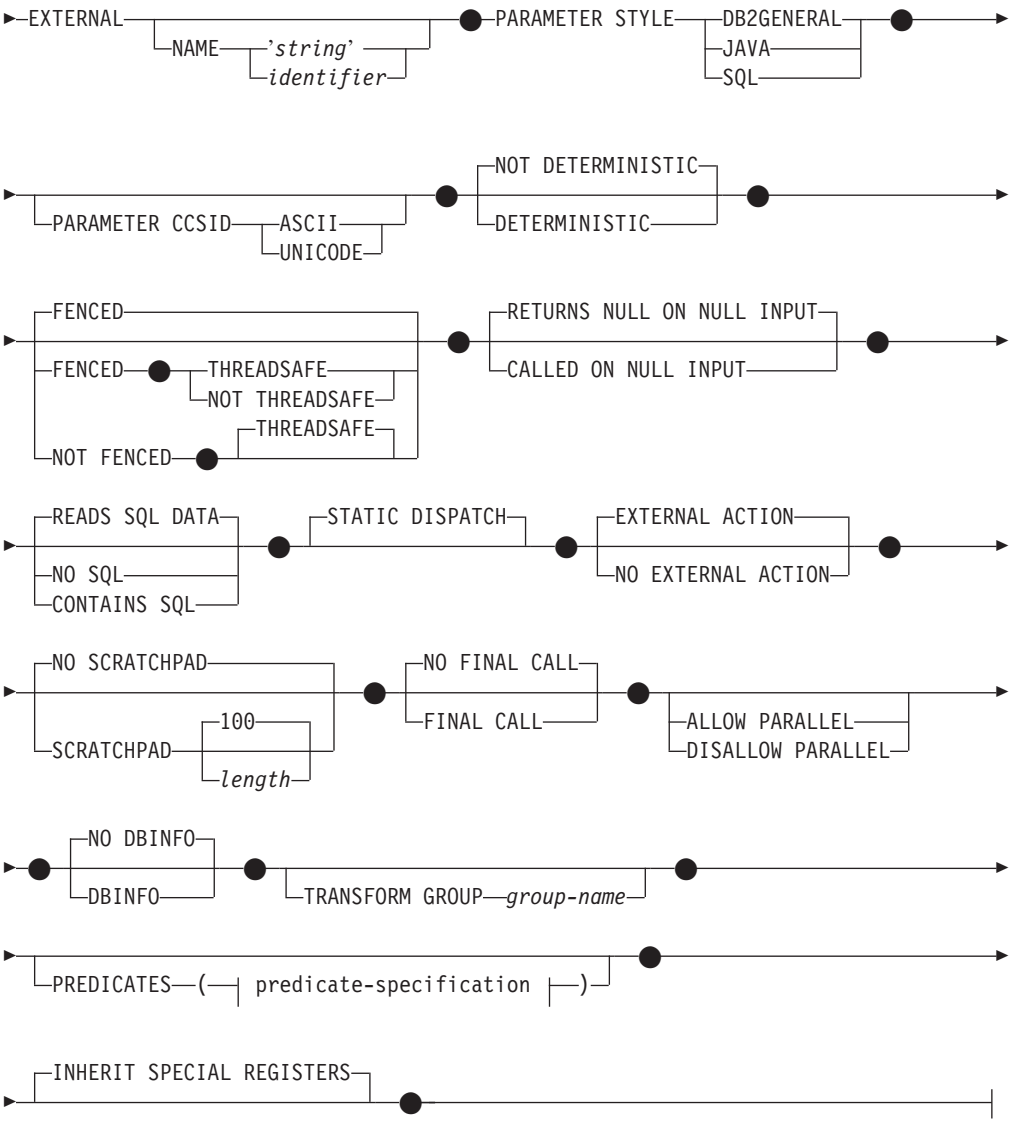
data-type1, data-type2, data-type3, data-type4:

built-in-type:

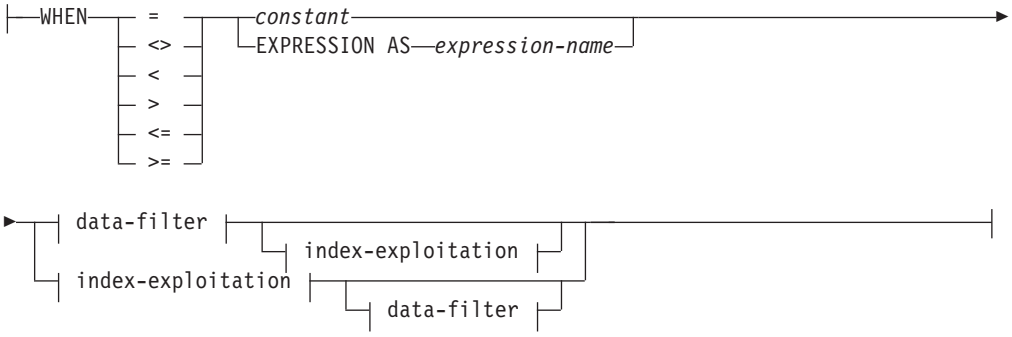

SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer)
,0
,integer
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
CHARACTER
CHAR
(1)
(integer)
FOR BIT DATA
VARCHAR
(integer)
CHARACTER
VARYING
CHAR
CLOB
CHARACTER
LARGE OBJECT
CHAR
(1M)
(integer)
K
M
G
GRAPHIC
(1)
(integer)
VARGRAPHIC
(integer)
DBCLOB
(1M)
(integer)
K
M
G
BLOB
BINARY LARGE OBJECT
(1M)
(integer)
K
M
G
DATE
TIME
TIMESTAMP
(6)
(integer)
XML
SYSPROC.
DB2SECURITYLABEL
(1) (2)

option-list:

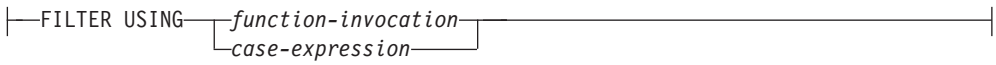

LANGUAGE
C
JAVA
CLR
OLE
(3)
SPECIFIC
specific-name



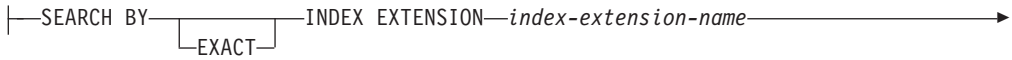
predicate-specification:



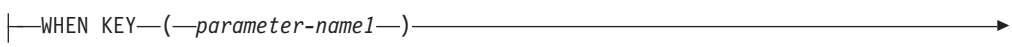
data-filter:

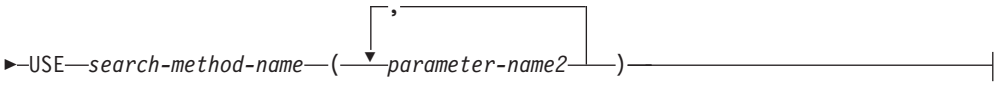


index-exploitation:



exploitation-rule:





주:

1 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

2 DB2SECURITYLABEL 유형의 컬럼의 경우, NOT NULL WITH DEFAULT는 내재적 및 명시적으로 지정될 수 없습니다(SQLSTATE 42842). DB2SECURITYLABEL 유형의 컬럼 디폴트값은 쓰기 액세스에 대한 세션 권한 부여 ID의 보안 레이블입니다.

3 LANGUAGE SQL도 지원됩니다.

설명

OR REPLACE

함수 정의가 현재 서버에 존재하는 경우 이를 교체하도록 지정합니다. 함수에 대해 부여된 특권에 영향을 주지 않는다는 점을 제외하고 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 효과적으로 삭제됩니다. 함수 정의가 현재 서버에 없으면 이 옵션은 무시됩니다. 기존 함수를 교체하려면, 새 정의의 특정 이름 및 함수 이름이 이전 정의의 특정 이름 및 함수 이름과 동일하거나 새 정의의 시그니처가 이전 정의의 시그니처와 일치해야 합니다. 그렇지 않으면, 새 함수가 작성됩니다.

function-name

정의되는 함수의 이름을 지정하십시오. 이 이름은 함수를 지정하는 규정화되거나 규정되지 않은 이름입니다. 규정되지 않은 *function-name* 양식은 SQL ID입니다(최대 길이는 128). 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지

정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 규정된 이름은 첫 번째 매개변수가 구조화된 유형일 경우 그 첫 번째 매개변수의 데이터 유형과 같으면 안됩니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수나 메소드를 식별하지 않아야 합니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 이루어진 이름이 지정된 경우, *schema-name*은 'SYS'로 시작될 수 없습니다. 그렇지 않으면, 오류(SQLSTATE 42939)가 발생합니다.

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *function-name*으로 사용될 수 없습니다. SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름 및 비교 연산자가 이에 해당합니다. 이 규칙을 지키지 못하면 오류가 발생됩니다(SQLSTATE 42939).

일반적으로 함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다.

이것에 대한 제한이 없다고 하여도, 겹쳐쓸 의도가 없으면 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안됩니다. 내장 스칼라 또는 집계 함수와 동일한 이름(예: LENGTH, VALUE, MAX)을 갖고 일관성 있는 인수를 갖지만 의미는 다른 함수를 제공하면, 동적 SQL문 또는 정적 SQL 응용프로그램을 다시 바인드할 때 오류가 발생할 수 있습니다. 응용프로그램이 실패하거나, 더 심한 경우 잘못된 결과를 제공하면서 성공적으로 실행된 것처럼 나타날 수도 있습니다.

(*parameter-declaration***,...)**

함수의 입력 매개변수 수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수가 받게 될 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 90개의 매개변수만 허용됩니다(SQLSTATE 54023).

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예를 들면, 다음과 같습니다.

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 이름이 지정된 두 함수는 모든 해당 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 따라서 CHAR(8) 및 CHAR(35)는 동일한 유형으로 간주되며, DECIMAL(11,2) 및 DECIMAL(4,3)도 마찬가지입니다. 유니코드 데이터베이스의 경우, CHAR(13) 및 GRAPHIC(8)은 같은 유형으로 간주됩니다. DECIMAL 및 NUMERIC과 같이 이러한 목적을 위해 동일한 유형으로 처리되도록 하는 유형들이 더 있습니다. 시그니처가 중복되면 오류가 리턴됩니다(SQLSTATE 42723).

parameter-name

입력 매개변수의 선택적 이름을 지정합니다. 매개변수 이름은 술어 스펙의 *인덱스 노출* 절에서 함수의 매개변수를 참조하기 위해 필요합니다. 이름은 매개변수 목록의 기타 *parameter-name*과 동일할 수 없습니다(SQLSTATE 42734).

data-type1

입력 매개변수의 데이터 유형을 지정합니다. 데이터 유형은 내장 데이터 유형, 구별 유형, 구조화된 유형 또는 참조 유형일 수 있습니다. 각 내장 데이터 유형에 대한 자세한 설명은 『CREATE TABLE』을 참조하십시오. 일부 데이터 유형은 모든 언어로 지원되지는 않습니다. SQL 데이터 유형과 호스트 언어 데이터 유형 간 맵핑에 대한 자세한 내용은 『Embedded SQL 응용프로그램에서 SQL 데이터 유형에 맵핑되는 데이터 유형』을 참조하십시오.

- 날짜 시간 유형 매개변수는 문자 데이터 유형으로 전달되며 데이터는 ISO 형식으로 전달됩니다.
- DECIMAL(및 NUMERIC)은 LANGUAGE C와 OLE에 대해 유효하지 않습니다(SQLSTATE 42815).
- DECFLOAT는 LANGUAGE C, COBOL, CLR, JAVA 및 OLE에는 유효하지 않습니다(SQLSTATE 42815).
- XML은 LANGUAGE OLE에서 유효하지 않습니다.
- 함수 내에서 보이는 XML 값은 함수 호출의 매개변수로 전달되는 XML 값의 병렬 버전이므로 XML 유형의 매개변수는 `XML AS CLOB(`*n*`)` 구문을 사용하여 선언되어야 합니다.
- CLR은 28보다 큰 DECIMAL 스케일은 지원하지 않습니다(SQLSTATE 42613).
- 배열 유형을 지정할 수 없습니다(SQLSTATE 42815).

사용자 정의 구별 유형의 경우, 매개변수의 길이, 정밀도 또는 스케일 속성은 구별 유형의 소스 유형의 속성입니다(CREATE TYPE에 지정된 속성). 구별 유형 매개변수는 구별 유형의 소스 유형으로 전달됩니다. 구별 유형의 이름이 규정되지 않은 경우, 데이터베이스 관리 프로그램은 SQL 경로의 스키마를 검색하여 스키마 이름을 분석합니다.

사용자 정의 구조화된 유형의 경우, 연관된 변환 그룹에 해당 변환 함수가 존재해야 합니다.

참조 유형의 경우, 매개변수 범위가 지정되지 않으면 매개변수를 REF(*type-name*)로 지정할 수 있습니다.

AS LOCATOR

매개변수 값의 로케이터가 실제 값 대신 함수로 전달되도록 지정합니다. LOB 데이터 유형 또는 LOB 데이터 유형을 기반으로 하는 구별 유형 매개변수에 대해서만 AS LOCATOR를 지정하십시오(SQLSTATE 42601).

값 대신 로케이터를 전달하면 함수로 전달되는 바이트 수를 줄일 수 있습니다(특히, 매개변수 값이 아주 큰 경우).

AS LOCATOR절은 데이터 유형을 승격할 수 있는지 여부를 판별하는 데 영향을 주지 않으며, 함수 결정에서 사용되는 함수 시그니처에도 영향을 주지 않습니다.

함수가 FENCED이고 NO SQL 옵션을 가질 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

RETURNS

이 필수 절은 함수의 출력을 식별합니다.

data-type2

출력의 데이터 유형을 지정합니다.

이 경우, 함수 매개변수 *data-type1*에 기술된 외부 함수의 매개변수에 대한 것과 동일한 고려사항이 적용됩니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR 절을 추가할 수 있습니다. 이는 LOB 로케이터가 실제 값 대신 UDF로부터 전달됨을 나타냅니다.

data-type3 **CAST FROM** *data-type4*

출력의 데이터 유형을 지정합니다.

이 형태의 RETURNS절은 함수 코드에 의해 리턴된 데이터 유형에서 호출 명령문으로 서로 다른 데이터 유형을 리턴하는 데 사용됩니다. 예를 들어, 다음 명령문을 살펴보십시오.

```
CREATE FUNCTION GET_HIRE_DATE(CHAR(6))
  RETURNS DATE CAST FROM CHAR(10)
  ...
```

함수 코드는 데이터베이스 관리 프로그램으로 CHAR(10)을 보낸 후, 이를 DATE로 변환하여 그 값을 호출 명령문에 전달합니다. *data-type4*는 *data-type3* 매개변수로 캐스트할 수 있어야 합니다. 캐스트할 수 없는 경우, 오류가 발생합니다(SQLSTATE 42880).

*data-type3*의 길이, 정밀도 또는 스케일은 *data-type4*에서 추론할 수 있으므로, *data-type3*에 대해 지정된 매개변수화된 유형의 길이, 정밀도 또는 스케일은 지정하지 않아도 됩니다(지정할 수도 있음). 대신 빈 괄호가 사용됩니다(예: VARCHAR()). 매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

구별 유형, 배열 유형 및 구조화된 유형은 *data-type4*에 지정되는 유형으로 유효하지 않습니다(SQLSTATE 42815).

캐스트 조작도 변환 오류를 야기할 수 있으므로 런타임 점검사항입니다.

AS LOCATOR

LOB 유형 또는 LOB 유형을 기반으로 하는 구별 유형인 *data-type4* 스펙의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 LOB 로케이터가 실제 값 대신 UDF로부터 전달됨을 나타냅니다.

SPECIFIC *specific-name*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. 규정되지 않은 *specific-name* 양식은 SQL ID입니다(최대 길이는 128). 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함한 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스나 메소드 스펙을 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *function-name*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *function-name*에 대해 사용된 규정자가 사용됩니다. 규정자를 지정할 경우, 규정자는 *function-name*의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

EXTERNAL

이 절은 CREATE FUNCTION문이 내부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 링크 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됩니다.

NAME절을 지정하지 않을 경우, "NAME *function-name*"이 사용됩니다.

NAME *'string'*

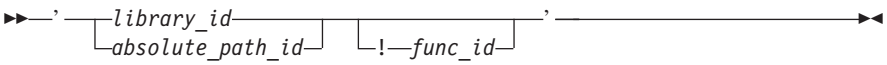
이 절은 정의되는 함수를 구현하는 사용자 작성 코드의 이름을 식별합니다.

'string' 옵션은 최대 길이가 254바이트인 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다.

- LANGUAGE C의 경우:

  지정된 *string*은 라이브러리 이름 및 라이브러리 내의 함수이고, 이것은 데이터베이스 관리 프로그램이 작성한 사용자 정의 함수를 실행하기 위해 호출하는 것입니다. CREATE FUNCTION문이 실행될 때 라이브러리(및 라이브러리 내의 함수)는 존재할 필요가 없습니다. 그러나 함수가 SQL문에서 사용될 경우 라이브러리와 그 라이브러리 내의 함수가 존재해야 하고 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42724).

*string*을 다음과 같이 정의할 수 있습니다.



작은따옴표 내에서 공백은 허용되지 않습니다.

library_id

함수를 포함하는 라이브러리 이름을 식별합니다. 데이터베이스 관리 프로그램은 다음과 같이 라이브러리를 찾습니다.

- UNIX 시스템에서 'myfunc'가 *library_id*로 지정되고 데이터베이스 관리 프로그램이 /u/production에서 실행 중인 경우, 데이터베이스 관리 프로그램은 /u/production/sqllib/function/myfunc 라이브러리에서 함수를 찾습니다.
- Windows 운영 체제에서 데이터베이스 관리 프로그램은 LIBPATH 또는 PATH 환경 변수에서 지정하는 디렉토리에서 함수를 찾습니다.

absolute_path_id

함수를 포함하는 파일의 전체 경로 이름을 나타냅니다.

예를 들어, UNIX 시스템에서 '/u/jchui/mylib/myfunc'는 데이터베이스 관리 프로그램이 /u/jchui/mylib에서 myfunc 공유 라이브러리를 찾게 합니다.

Windows 운영 체제에서 'd:₩mylib₩myfunc.dll'은 데이터베이스 관리 프로그램이 d:₩mylib 디렉토리에서 동적 링크 라이브러리 myfunc.dll을 로드하게 합니다. 절대 경로 ID를 사용하여 루틴 본문을 식별할 경우, .dll 확장자를 추가해야 합니다.

! func_id

호출될 함수의 시작점 이름을 식별합니다. !은 library ID와 function ID 사이의 분리 문자로 사용됩니다.

예를 들어, UNIX 시스템에서 'mymod!func8'은 데이터베이스 관리 프로그램이 $inst_home_dir/sqllib/function/mymod 라이브러리를 찾아 해당 라이브러리에서 시작점 func8을 사용하도록 지시합니다.

Windows 운영 체제에서 'mymod!func8'은 데이터베이스 관리 프로그램이 mymod.dll 파일을 로드하고 동적 링크 라이브러리(DLL)의 func8() 함수를 호출하게 합니다.

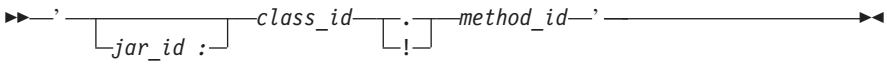
문자열이 적절히 구성되지 않으면, 오류가 발생합니다(SQLSTATE 42878).

모든 외부 함수의 본문이 모든 데이터베이스 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

- LANGUAGE JAVA의 경우:

  지정된 *string*에는 선택적 jar 파일 ID, 클래스 ID 및 메소드 ID가 포함되는데, 이는 작성되는 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 것입니다. CREATE FUNCTION문이 실행될 때 클래스 ID 및 메소드 ID는 존재하지 않아도 됩니다. *jar_id*를 지정할 경우, CREATE FUNCTION문이 실행될 때 존재해야 합니다. 그러나 함수가 SQL 문에서 사용될 경우 메소드 ID가 존재해야 하고 데이터베이스 서버 머신에서 액세스할 수 있어야 하며, 그렇지 않을 경우 오류가 리턴됩니다 (SQLSTATE 42724).

  *string*을 다음과 같이 정의할 수 있습니다.



  작은따옴표 내에서 공백은 허용되지 않습니다.

  jar_id

  데이터베이스에 설치될 때, jar 콜렉션에 제공된 jar ID를 식별합니다. 간단한 ID 또는 스키마 규정된 ID일 수 있습니다. 'myJar' 및 'mySchema.myJar'가 그 예입니다.

  class_id

  Java 오브젝트의 클래스 ID를 나타냅니다. 클래스가 패키지의 일부일 경우, 클래스 ID 부분에 완전한 패키지 접두어(예: 'myPacks.UserFuncs')가 포함되어야 합니다. JVM(Java Virtual Machine)은 클래스를 디렉토리 '.../myPacks/UserFuncs/'에서 찾게 됩니다. Windows 운영 체제에서 Java Virtual Machine은 '...₩myPacks₩UserFuncs₩' 디렉토리에서 찾습니다.

  method_id

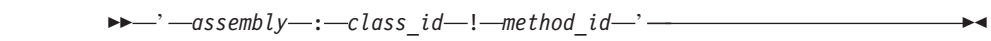
  호출할 Java 오브젝트의 메소드 이름을 식별합니다.

- LANGUAGE CLR의 경우:

  지정된 *string*은 .NET 어셈블리(라이브러리 또는 실행 파일), 해당 어셈블리에 있는 클래스 및 작성되는 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 클래스에 있는 메소드를 표시합니다. CREATE FUNCTION 문이 실행될 때 모듈, 클래스 및 메소드는 없어도 됩니다. 그러나 함수가 SQL 문에서 사용될 경우 모듈, 클래스 및 메소드가 존재해야 하고 데이터베이스 서버 머신에서 액세스할 수 있어야 하며, 그렇지 않을 경우 오류가 리턴됩니다(SQLSTATE 42724).

관리 코드 확장자가 포함된 것을 표시하도록 '/clr' 컴파일러 옵션으로 컴파일된 C++ 루틴은 'LANGUAGE C'가 아닌 'LANGUAGE CLR'로 카탈로그되어야 합니다. DB2는 필요한 런타임 결정을 하기 위해 .NET 인프라스트럭처가 사용자 정의 함수에서 활용되고 있는지 알아야 합니다. .NET 인프라스트럭처를 사용하는 모든 사용자 정의 함수(UDF)는 'LANGUAGE CLR'로 카탈로그되어야 합니다.

*string*을 다음과 같이 정의할 수 있습니다.



이름은 작은따옴표로 묶어야 합니다. 공백은 허용되지 않습니다.

assembly
클래스가 있는 DLL 또는 다른 어셈블리 파일을 식별합니다. 파일 확장자(예: .dll)를 지정해야 합니다. 전체 경로 이름을 지정하지 않은 경우, 파일은 DB2 설치 경로의 함수 디렉토리(예: c:₩sqllib₩function)에 상주해야 합니다. 파일이 설치 함수 디렉토리의 서브디렉토리에 상주할 경우, 전체 경로를 지정하는 대신에 서브디렉토리를 파일 이름 전에 지정할 수 있습니다. 예를 들어, 설치 디렉토리가 c:₩sqllib이고 어셈블리 파일이 c:₩sqllib₩function₩myprocs₩mydotnet.dll일 경우, 어셈블리에 대해 'myprocs₩mydotnet.dll'만 지정하면 됩니다. 이 매개변수의 대소문자 구분은 파일 시스템의 대소문자 구분과 동일합니다.

class_id
호출될 메소드가 있는 지정된 어셈블리 내에 있는 클래스의 이름을 지정합니다. 클래스가 이름 스페이스에 상주할 경우, 전체 이름 스페이스를 클래스와 같이 지정해야 합니다. 예를 들어, EmployeeClass 클래스가 MyCompany.ProcedureClasses 이름 스페이스에 있으면 클래스에 대해 MyCompany.ProcedureClasses.EmployeeClass를 지정해야 합니다. 일부 .NET 언어용 컴파일러는 프로젝트 이름을 클래스의 이름 스페이스로 추가하는데, 명령행 컴파일러가 사용되었는지 또는 GUI 컴파일러가 사용되었는지에 따라 동작이 다를 수 있습니다. 이 매개변수는 대소문자가 구분됩니다.

method_id
호출될 지정된 클래스 내에 있는 메소드를 지정합니다. 이 매개변수는 대소문자가 구분됩니다.

- LANGUAGE OLE의 경우:

지정된 *stringv*은 OLE 프로그램 가능 ID(progid) 또는 클래스 ID(clsid) 및 메소드 ID로서, 작성중인 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출한 것입니다. CREATE FUNCTION문이 실행될 때 프

로그램가능 ID 또는 클래스 ID 및 메소드 ID는 존재하지 않아도 됩니다. 그러나 함수가 SQL문에서 사용될 경우 메소드 ID가 존재해야 하고 데이터베이스 서버 머신에서 액세스할 수 있어야 하며, 그렇지 않을 경우 오류가 리턴됩니다(SQLSTATE 42724).

*string*을 다음과 같이 정의할 수 있습니다.

작은따옴표 내에서 공백은 허용되지 않습니다.

progid

OLE 오브젝트의 프로그램가능 ID를 식별합니다.

*progid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임시 OLE API로 전달됩니다. 지정된 OLE 오브젝트는 작성 가능해야 하며 후기 바인딩(IDispatch형 바인딩이라고도 함)을 지원해야 합니다.

clsid

작성할 OLE 오브젝트의 클래스 ID를 식별합니다. OLE 오브젝트가 progid를 사용하여 등록되지 않은 경우 *progid*를 지정하기 위한 대안으로 사용할 수 있습니다. *clsid*의 형식은 다음과 같습니다.

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

여기서, 'n'은 영숫자입니다. *clsid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임시 OLE API로 전달됩니다.

method_id

호출할 OLE 오브젝트의 메소드 이름을 식별합니다.

NAME *identifier*

지정된 이 *identifier*는 SQL ID입니다. SQL ID는 문자열에서 *library id*로 사용됩니다. 분리 ID가 아닐 경우, ID는 대문자로 겹쳐집니다. ID에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 형식의 이름은 LANGUAGE C에서만 사용할 수 있습니다.

LANGUAGE

이 필수 절은 사용자 정의 함수의 본문이 작성되는 언어 인터페이스 규칙을 지정하는 데 사용됩니다.

C 데이터베이스 관리 프로그램은 사용자 정의 함수를 C 함수인 것처럼 호출합니다. 사용자 정의 함수는 표준 ANSI C 프로토타입에 의해 정의된 대로 C 언어 호출 규칙과 링크 규칙을 따라야 합니다.

JAVA 데이터베이스 관리 프로그램이 사용자 정의 함수를 Java 클래스의 메소드로 호출함을 의미합니다.

CLR 데이터베이스 관리 프로그램이 사용자 정의 함수를 .NET 클래스의 한 메소드로 호출합니다. 이 시점에서 LANGUAGE CLR은 Windows 운영 체제에서 실행되는 사용자 정의 함수에 대해서만 지원됩니다. CLR 루틴에 대해서는 NOT FENCED를 지정할 수 없습니다(SQLSTATE 42601).

OLE 데이터베이스 관리 프로그램은 사용자 정의 함수를 OLE 자동화 오브젝트에 의해 표시되는 메소드인 것처럼 호출합니다. 사용자 정의 함수(UDF)는 *OLE 자동화 프로그래머 참조서*에 설명된 대로 OLE 자동화 데이터 유형 및 호출 메커니즘에 따라야 합니다.

LANGUAGE OLE는 Windows 운영 체제용 DB2에 저장된 사용자 정의 함수에 대해서만 지원됩니다. LANGUAGE OLE로 정의된 UDF에 대해서는 THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

PARAMETER STYLE

이 절은 함수로부터 값을 리턴하고 매개변수를 전달하기 위한 규칙을 지정하는 데 사용됩니다.

DB2GENERAL

Java 클래스의 메소드로 정의된 외부 함수에 매개변수를 전달하고 함수로부터 값을 리턴하는 규칙을 지정하는 데 사용됩니다. 이것은 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

값 DB2GENRL은 DB2GENERAL의 동의어로 사용할 수 있습니다.

JAVA

함수가 Java 언어 및 SQLJ 루틴 스펙을 준수하는 규칙을 전달하는 매개변수를 사용할 것임을 의미합니다. LANGUAGE JAVA가 사용되는 경우에만 이를 지정할 수 있고 구조화된 데이터 유형은 매개변수로 지정되지 않으며, CLOB, BLOB 또는 DBCLOB 데이터 유형은 리턴 유형으로 지정되지 않습니다(SQLSTATE 429B8). PARAMETER STYLE JAVA 함수는 FINAL CALL, SCRATCHPAD 또는 DBINFO절을 지원하지 않습니다.

SQL

C 언어 호출 및 연결 규칙, OLE 자동화 오브젝트에 의해 표시되는 메소드 또는 .NET 오브젝트의 공용 정적 메소드를 준수하는 외부 함수에 매개변수를 전달하고 외부 함수로부터 값을 리턴하기 위해 사용하는 규칙을 지정하는 데 사용됩니다. 이는 LANGUAGE C, LANGUAGE CLR 또는 LANGUAGE OLE를 사용할 때 지정해야 합니다.

PARAMETER CCSID

함수에 전달되거나 함수에서 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII

문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031). 함수가 호출될 때 함수의 응용프로그램 코드 페이지는 데이터베이스 코드 페이지입니다.

UNICODE

문자열 데이터를 유니코드로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, 문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있습니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, 문자 데이터는 UTF-8로 되어 있습니다. 어떠한 경우라도 함수가 호출될 때 함수의 응용프로그램 코드 페이지는 1208입니다.

데이터베이스가 유니코드 데이터베이스가 아니고 PARAMETER CCSID UNICODE를 사용하여 함수가 작성되는 경우, 함수에는 그래픽 유형, XML 유형 또는 사용자 정의 유형이 있을 수 없습니다(SQLSTATE 560C1).

데이터베이스가 유니코드 데이터베이스가 아니고 데이터베이스 구성에 대체 조합 시퀀스가 지정된 경우, 함수는 PARAMETER CCSID ASCII 또는 PARAMETER CCSID UNICODE로 작성될 수 있습니다. 함수에 전달되거나 함수로부터 전달되는 모든 문자열 데이터는 해당 코드 페이지로 변환됩니다.

이 절은 LANGUAGE OLE, LANGUAGE JAVA 또는 LANGUAGE CLR로 지정할 수 없습니다(SQLSTATE 42613).

DETERMINISTIC 또는 NOT DETERMINISTIC

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC을 지정하여 방지할 수 있습니다. NOT DETERMINISTIC 함수의 예는 난수(random-number) 생성 프로그램입니다. DETERMINISTIC 함수의 예는 입력의 제곱근을 판별하는 함수입니다.

FENCED 또는 NOT FENCED

이 절은 함수가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 어드레스 스페이스에서 실행하기에 FENCED 또는 NOT FENCED를 지정합니다.

함수가 FENCED로 등록된 경우, 데이터베이스 관리 프로그램은 함수에서 액세스하지 못하도록 내부 자원(예: 데이터 버퍼)을 보호합니다. 대부분의 함수는 FENCED나 NOT FENCED로 실행됩니다. 일반적으로 FENCED로 수행되는 함수는 NOT FENCED로 실행되는 함수와는 다르게 실행됩니다.

주의:

제대로 코딩, 검토 및 테스트되지 않은 함수에 NOT FENCED를 사용하면 DB2 데이터베이스의 무결성이 손상될 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 사용자 정의 함수가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

LANGUAGE OLE 또는 NOT THREADSAFE의 함수에 대해서는 FENCED만 지정할 수 있습니다(SQLSTATE 42613).

함수가 FENCED이고 NO SQL 옵션을 가질 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

함수를 NOT FENCED로 등록하기 위해서는 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED_ROUTINE)이 필요합니다.

NOT FENCED절을 지정하면 LANGUAGE CLR 사용자 정의 함수(UDF)를 작성할 수 없습니다(SQLSTATE 42601).

THREADSAFE 또는 NOT THREADSAFE

함수가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 아니면 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

함수가 OLE가 아닌 다른 LANGUAGE를 사용하여 정의된 경우에는 다음과 같습니다.

- 함수가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 함수를 호출할 수 있습니다. 일반적으로 스레드가 안전하려면 함수가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED와 NOT FENCED 함수는 둘다 THREADSAFE될 수 있습니다.
- 함수가 NOT THREADSAFE로 정의되어 있으면, 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 함수를 동시에 호출할 수 없습니다.

FENCED 함수의 경우, LANGUAGE가 JAVA 또는 CLR이면 THREADSAFE가 디폴트값입니다. 다른 모든 언어에서는 NOT THREADSAFE가 디폴트값입니다. 이 함수가 LANGUAGE OLE로 정의된 경우, THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

NOT FENCED 함수의 경우 THREADSAFE가 디폴트값입니다. NOT THREADSAFE는 지정될 수 없습니다(SQLSTATE 42613).

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 널(NULL) 값인 인수가 있을 경우 외부 함수로의 호출을 방지하기

위해 사용됩니다. 사용자 정의 함수(UDF)가 매개변수를 갖지 않는 것으로 정의되면, 이러한 널(NULL) 인수 조건이 발생할 수 없으며 이 스펙의 코딩 방법이 문제가 되지 않습니다.

RETURNS NULL ON NULL INPUT이 지정되고 실행시 함수 인수 중 어느 하나가 널(NULL)일 경우, 사용자 정의 함수는 호출되지 않으며 결과는 널(NULL) 값이 됩니다.

CALLED ON NULL INPUT이 지정될 경우, 인수가 널(NULL)인지에 관계없이 사용자 정의 함수가 호출됩니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나 널(NULL) 값 인수 값에 대한 테스트 책임은 UDF에 있습니다.

이전 버전과의 호환성 및 제품군 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로 NULL CALL을 사용할 수도 있습니다. 마찬가지로, NOT NULL CALL을 RETURNS NULL ON NULL INPUT에 대한 동의어로 사용할 수도 있습니다.

NO SQL, CONTAINS SQL, READS SQL DATA

함수가 SQL문을 발행하는지 여부 및 발행할 경우 SQL문의 유형을 나타냅니다.

NO SQL

함수가 SQL문을 실행할 수 없음을 나타냅니다(SQLSTATE 38001).

CONTAINS SQL

SQL 데이터를 읽지도 수정하지도 않는 SQL문이 함수에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 38004 또는 42985). 함수에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문이 이 함수에 포함될 수 있음을 나타냅니다(SQLSTATE 38002 또는 42985). 함수에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

STATIC DISPATCH

이 선택적 절은 함수 결정시 DB2가 함수 매개변수의 정적 유형(선언된 유형)에 기반하는 함수를 선택한다는 것을 나타냅니다.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하는지의 여부를 지정합니다. 외부 조치의 예로는 메시지 전송 또는 파일에 레코드 쓰기를 들 수 있습니다. 디폴트값은 EXTERNAL ACTION입니다.

EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하도록 지정합니다.

외부 조치가 있는 함수는 병렬 태스크가 해당 함수를 실행할 경우 부정확한 결과를 리턴할 수 있습니다. 예를 들어 함수가 자신에 대한 각 초기 호출에 대해 메모를 전송할 경우, 함수에 대해 한 번의 메모를 전송하는 대신에 각 병렬 태스크에 대해 하나의 메모를 전송합니다. 정확하게 병렬 처리와 작동하지 않는 함수에 대해 DISALLOW PARALLEL절을 지정하십시오.

NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하지 않는다는 것을 지정합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다.

NO SCRATCHPAD 또는 SCRATCHPAD *length*

이 선택적 절은 외부 함수에 대해 스크래치 패드가 제공되는지 여부를 지정할 때 사용할 수 있습니다. 사용자 정의 함수는 재입력이 가능하도록 적극 권장되므로, 스크래치 패드는 함수가 한 호출에서 다음 호출로의 "상태를 저장"하기 위한 수단을 제공합니다.

SCRATCHPAD를 지정하면, 사용자 정의 함수를 처음 호출할 때 외부 함수에서 사용할 스크래치 패드에 대해 메모리가 할당됩니다. 이 스크래치 패드는 다음 등록 정보를 갖습니다.

- *length* - 지정될 경우, 스크래치 패드의 크기를 바이트 단위로 설정합니다. 이 값은 1 - 32 767이어야 합니다(SQLSTATE 42820). 디폴트값 크기는 100바이트입니다.
- 모두 X'00''으로 초기화됩니다.
- 범위는 SQL문입니다. SQL문에서 외부 함수에 대한 참조당 하나의 스크래치 패드가 있습니다. 그래서 다음 명령문의 UDFX 함수가 SCRATCHPAD 키워드를 사용하여 정의된 경우, 세 개의 스크래치 패드가 지정됩니다.

```
SELECT A, UDFX(A) FROM TABLEB
  WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

  ALLOW PARALLEL이 지정되거나 디폴트 설정되면, 범위가 위와 달라집니다. 함수가 다중 데이터베이스 파티션에서 실행될 경우, SQL문에서의 각 함수 참조에 대해 스크래치패드는 함수가 처리되는 각 데이터베이스 파티션에 지정됩니다. 마찬가지로 파티션 내 병렬 처리를 사용하여 쿼리가 실행되면 네 개 이상의 스크래치 패드가 지정될 수 있습니다.
- 지속적입니다. 내용이 하나의 외부 함수 호출에서 다음 호출로 보존됩니다. 한 호출상의 외부 함수에서 변경한 스크래치 패드는 다음 호출에도 있게 됩니다. 데이터베이스 관리 프로그램은 각 SQL문 실행이 시작될 때 스크래치 패드를 초기화하며, 각 서브쿼리 실행이 시작될 때 스크래치 패드를 재설정합니다. FINAL CALL 옵션을 지정하면, 시스템에서 스크래치 패드를 재설정하기 전에 마지막 호출을 발행합니다.

- 외부 함수가 획득할 수 있는 시스템 자원(예: 메모리)에 대한 중심점으로 사용됩니다. 함수는 첫 번째 호출에서 메모리를 획득하고, 그 주소를 스크래치 패드에 보관한 후 다음 호출에서 이를 참조할 수 있습니다.

  (시스템 자원을 확보한 경우에는 FINAL CALL 키워드를 지정해야 합니다. 그러면, 명령문 끝에서 특수 호출이 이루어져 외부 함수가 확보한 시스템 자원이 해제됩니다.)

SCRATCHPAD를 지정하면, 사용자 정의 함수를 호출할 때마다 추가 인수가 스크래치 패드를 지정하는 외부 함수로 전달됩니다.

NO SCRATCHPAD를 지정하면, 어떤 스크래치 패드도 외부 함수에 할당되거나 전달되지 않습니다.

SCRATCHPAD는 PARAMETER STYLE JAVA 함수에 지원되지 않습니다.

FINAL CALL 또는 NO FINAL CALL

이 선택적 절은 외부 함수에 대해 마지막 호출이 수행되는지 여부를 지정합니다. 마지막 호출의 목적은 외부 함수가 획득한 시스템 자원을 해제할 수 있도록 하는 것입니다. 이것은 외부 함수가 메모리와 같은 시스템 자원을 확보하여 이들을 스크래치 패드에 저장하는 경우에 SCRATCHPAD 키워드와 함께 사용하면 유용할 수 있습니다. FINAL CALL을 지정하면 실행시, 다음과 같은 작업이 수행됩니다.

- 호출 유형을 지정하는 추가 인수가 외부 함수에 전달됩니다. 호출 유형은 다음과 같습니다.
  - 정상 호출: SQL 인수가 전달되고 결과 리턴이 예상됩니다.
  - 첫 번째 호출: 이 SQL문의 사용자 정의 함수(UDF)에 대한 이 참조의 외부 함수에 대한 첫 번째 호출입니다. 첫 번째 호출은 정상적인 호출입니다.
  - 마지막 호출: 함수가 자원에 여유 공간을 확보할 수 있도록 하는 외부 함수에 대한 마지막 호출입니다. 마지막 호출은 정상적인 호출이 아닙니다. 이 마지막 호출은 다음 상황에서 발생합니다.
    - 명령문의 끝: 이 경우는 커서가 커서 지향 명령문에 대해 닫혀 있거나, 명령문이 다른 명령문을 실행할 때 발생합니다.
    - 병렬 태스크의 끝: 이 경우는 병렬 태스크에 의해 함수가 실행될 때 발생합니다.
    - 트랜잭션의 끝 또는 인터럽트: 정상적인 명령문의 끝이 발생하지 않을 때 발생합니다. 예를 들어, 응용프로그램의 논리는 어떤 이유로 커서 닫기를 생략할 수 있습니다. 이러한 유형의 마지막 호출이 일어나는 동안에는 CLOSE 커서에 대해서만 SQL문을 발행할 수 있고 다른 커서에 대해서는 SQL문을 발행할 수 없습니다(SQLSTATE 38505). 이런 유형의 마지막 호출은 ″호출 유형″ 인수에 특수한 값을 사용하여 표시됩니다.

WITH HOLD로 정의된 커서가 열려 있는 동안 커미트 조작이 발생하면, 그 다음의 커서를 닫을 때 또는 응용프로그램 종료시 마지막 호출이 수행됩니다.

NO FINAL CALL을 지정하면 "호출 유형" 인수가 외부 함수에 전달되지 않으며 마지막 호출도 이루어지지 않습니다.

FINAL CALL은 PARAMETER STYLE JAVA 함수에 지원되지 않습니다.

ALLOW PARALLEL 또는 DISALLOW PARALLEL

이 선택적 절은 함수에 대한 단일 참조에 대해 함수 호출을 병렬화할 수 있는지 여부를 지정합니다. 일반적으로 대부분의 스칼라 함수 호출은 병렬화가 가능해야 하지만, 병렬화할 수 없는 함수(스크래치 패드의 단일본에 근거한 함수)도 있을 수 있습니다. 스칼라 함수에 대해 ALLOW PARALLEL 또는 DISALLOW PARALLEL을 지정하는 경우 DB2는 이 스펙을 승인합니다. 함수에 적합한 키워드를 결정할 때 다음 사항을 고려해야 합니다.

- 모든 UDF 호출이 서로 완전히 독립되어 있는지 여부. 그럴 경우, ALLOW PARALLEL을 지정하십시오.
- 각 UDF 호출이 스크래치 패드를 갱신하여 다음 호출과 관련한 값을 제공하는지 여부 (예: 카운터 증가). 외부 조치가 있을 경우 DISALLOW PARALLEL을 지정하거나 디폴트값을 사용하십시오.
- 하나의 데이터베이스 파티션에서만 발생해야 하는, UDF에 의해 수행되는 일부 외부 조치가 있습니까? 외부 조치가 있을 경우 DISALLOW PARALLEL을 지정하거나 디폴트값을 사용하십시오.
- 비용이 많이 드는 초기화 프로세스를 최소한의 횟수로 수행할 수 있도록 스크래치 패드가 사용되었는지 여부. 그럴 경우, ALLOW PARALLEL을 지정하십시오.

모든 경우, 모든 외부 함수의 본문이 모든 데이터베이스 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

단, 다음 옵션 중 하나 이상이 명령문에 지정되어 있는 경우는 예외입니다. 디폴트값은 ALLOW PARALLEL입니다.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

이들 옵션 중 하나가 지정되거나 내재되는 경우, 디폴트값은 DISALLOW PARALLEL입니다.

INHERIT SPECIAL REGISTERS

이 선택적 절은 함수에 있는 갱신 가능한 특수 레지스터가 레지스터의 초기값을 호출 명령문의 환경에서 상속하도록 지정합니다. 커서의 select문에서 호출되는 함수

의 경우에는 초기값을 커서가 열려 있는 환경에서 상속합니다. 중첩 오브젝트(예: 트리거 또는 뷰)에서 호출되는 루틴의 경우에는 초기값을 오브젝트 정의에서 상속하지 않고 런타임 환경에서 상속합니다.

특수 레지스터에 대한 변경사항은 함수 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 갱신할 수 없는 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 디폴트값으로 설정됩니다.

NO DBINFO 또는 DBINFO

이 선택적 절은 DB2에 의해 알려진 특정 정보가 UDF에 추가적인 호출 시간 인수(DBINFO)로 전달되는지 아니면 전달되지 않는지(NO DBINFO) 여부를 지정합니다. NO DBINFO가 디폴트값입니다. DBINFO는 LANGUAGE OLE(SQLSTATE 42613) 또는 PARAMETER STYLE JAVA에서는 지원되지 않습니다.

DBINFO를 지정할 경우, 다음 정보가 포함된 구조가 UDF에 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름입니다.
- 응용프로그램 ID - 각 데이터베이스 연결에 대해 설정되는 고유한 응용프로그램 ID입니다.
- 응용프로그램 권한 부여 ID - 이 UDF와 응용프로그램 사이의 중첩 UDF에 관계없이 응용프로그램 런타임 권한 부여 ID
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 스키마 이름 - 테이블 이름과 완전히 같은 조건하에서 스키마의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 테이블 이름 - UDF 참조가 UPDATE문의 SET절 오른쪽에 있거나 INSERT문의 VALUES 목록에 있는 항목인 경우에만 갱신하거나 삽입할 테이블의 규정되지 않은 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 컬럼 이름 - 테이블 이름과 완전히 같은 조건하에서, 갱신하거나 삽입할 컬럼의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 데이터베이스 버전/릴리스 - UDF를 호출한 데이터베이스 서버의 버전, 릴리스 및 개정 레벨을 나타냅니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.
- 테이블 함수 결과 컬럼 수 - 외부 스칼라 함수에는 해당되지 않습니다.

TRANSFORM GROUP *group-name*

함수를 호출할 때 사용자 정의된, 구조화된 유형 변환에 사용할 변환 그룹을 나타냅니다. 변환은 함수 정의가 사용자 정의 구조화된 유형을 매개변수로 포함하거나 데이터 유형을 리턴할 경우에 필요합니다. 이 절을 지정하지 않으면, 디폴트 그룹 이름인 DB2_FUNCTION이 사용됩니다. 지정된 (또는 디폴트값인) *group-name*이 참조된 구조화된 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE

42741). 필수 FROM SQL 또는 TO SQL 변환 함수가 제공된 그룹 이름과 구조화된 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42744).

지정되었는지 또는 내재되었는지 여부와 관계없이 변환 함수 FROM SQL 및 TO SQL은 구조화된 유형 및 해당되는 내장 유형 속성 사이에 적절하게 변환하는 SQL 함수여야 합니다.

PREDICATES

이 함수가 술어에서 사용될 경우, 수행되는 필터링 또는 인덱스 확장 이용을 정의합니다. 술어 스펙은 검색 조건의 선택적 SELECTIVITY절을 지정할 수 있도록 허용합니다. PREDICATES절을 지정할 경우, 함수는 NO EXTERNAL ACTION을 사용하여 DETERMINISTIC으로 정의되어야 합니다(SQLSTATE 42613). PREDICATES절을 지정했는데 데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE를 지정하면 안됩니다(SQLSTATE 42613).

WHEN *comparison-operator*

비교 연산자("=", "<", ">", ">=", "<=", "<>")를 사용하여 술어에 특정 함수 사용을 소개합니다.

constant

데이터 유형이 함수의 RETURNS 유형에 해당하는 상수 값을 지정합니다(SQLSTATE 42818). 술어가 같은 비교 연산자와 이 상수를 사용하여 이 함수를 사용할 때, 지정된 필터링과 인덱스 이용이 옵티마이저에 의해 고려됩니다.

EXPRESSION AS *expression-name*

표현식에 대한 이름을 제공합니다. 술어가 같은 비교 연산자와 표현식을 사용하여 이 함수를 사용할 때, 필터링과 인덱스 노출을 사용할 수도 있습니다. 표현식에는 검색 함수 인수로 사용될 수 있도록 표현식 이름이 지정됩니다. *expression-name*은 작성되는 함수의 *parameter-name*과 같을 수 없습니다(SQLSTATE 42711). 표현식을 지정할 경우, 그 표현식의 유형이 식별됩니다.

FILTER USING

결과 테이블의 추가 필터링에 외부 함수나 CASE 표현식의 스펙을 사용할 수 있게 합니다.

function-invocation

결과 테이블의 추가 필터링을 수행하는 데 사용할 수 있는 필터 함수를 지정합니다. 이것은 행 규정 여부를 판별하기 위해 사용자 정의 술어가 실행되어야 하는 행 수를 감소하는 정의된 함수(술어에 사용됨)의 버전입니다. 인덱스에 의해 생성되는 결과가 사용자 정의 술어에 대해 예상된 결과에 근접할 경우, 필터링 함수를 적용하는 것은 불필요할 수도 있습니다. 지정하지 않으면, 데이터 필터링은 수행되지 않습니다.

이 함수는 *parameter-name*, *expression-name* 또는 상수를 인수로 사용하고(SQLSTATE 42703), 정수를 리턴합니다(SQLSTATE 428E4). 리턴 값 1은 행이 보존됨을 의미하고, 그렇지 않으면 행은 버려집니다.

이 함수는 다음을 만족해야 합니다.

- LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 429B4).
- NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의하지 않아야 합니다(SQLSTATE 42845).
- 매개변수 중 어느 하나의 데이터 유형으로 구조화된 데이터 유형을 가지고 있지 않아야 합니다(SQLSTATE 428E3).
- 서브쿼리를 포함하고 있지 않아야 합니다(SQLSTATE 428E4).
- XMLQUERY 또는 XMLEXISTS 표현식을 포함하고 있지 않아야 합니다(SQLSTATE 428E4).

인수가 다른 함수나 메소드를 호출할 경우, 해당 중첩 함수나 메소드에 대해서도 이 규칙이 시행됩니다. 그러나 시스템에서 생성되는 관찰자 메소드는 인수가 내장 데이터 유형을 평가하면 필터 함수(또는 인수로 사용되는 함수나 메소드)에 대한 인수로 허용됩니다.

함수 정의자에게는 지정한 필터 함수에 대한 EXECUTE 특권이 있어야 합니다.

*function-invocation*절의 길이는 데이터베이스 코드 페이지에서 65,536바이트를 초과할 수 없습니다(SQLSTATE 22001).

case-expression

결과 테이블의 추가 필터링에 대한 CASE 표현식을 지정합니다. *searched-when-clause* 및 *simple-when-clause*는 *parameter-name*, *expression-name* 또는 상수를 사용할 수 있습니다(SQLSTATE 42703). FILTER USING *function-invocation*에 규칙이 지정된 외부 함수를 결과 표현식으로 사용할 수 있습니다. *case-expression*에서 참조되는 함수나 메소드는 *function-invocation* 아래에 나열된 네 가지 규칙도 따라야 합니다.

서브쿼리 및 XMLQUERY 또는 XMLEXISTS 표현식은 *case-expression*에서 쓰일 수 없습니다(SQLSTATE 428E4).

CASE 표현식은 정수를 리턴해야 합니다(SQLSTATE 428E4). 결과 표현식에서 리턴 값 1은 행이 보존됨을 의미하며, 그렇지 않으면 행은 버려집니다.

*case-invocation*절의 길이는 데이터베이스 코드 페이지에서 65,536바이트를 초과할 수 없습니다(SQLSTATE 22001).

index-exploitation

인덱스 이용에 사용할 수 있는 인덱스 확장의 검색 메소드 측면에서 규칙 세트를 정의합니다.

SEARCH BY INDEX EXTENSION *index-extension-name*

인덱스 확장을 식별합니다. *index-extension-name*은 기존의 인덱스 확장을 식별해야 합니다.

EXACT

인덱스 찾아보기가 술어 평가 측면에서 정확함을 나타냅니다. 원래 사용자 정의 술어 함수나 필터가 인덱스 찾아보기 후에 적용되지 않도록 DB2에 지시하려면 EXACT를 사용하십시오. EXACT 술어는 인덱스 찾아보기가 술어와 같은 결과를 리턴할 때 유용합니다.

EXACT를 지정하지 않을 경우, 원래의 사용자 정의 술어는 인덱스 찾아보기 후에 적용됩니다. 인덱스가 술어의 예측만을 제공할 것으로 예상되면, EXACT 옵션을 지정하지 마십시오.

인덱스 찾아보기를 사용하지 않으면, 필터 함수와 원래의 술어를 적용해야 합니다.

exploitation-rule

검색 목표 및 검색 인수, 그리고 이들을 사용하여 인덱스 확장에서 정의된 검색 메소드를 통해 인덱스 검색을 수행하는 방법에 대해 설명합니다.

WHEN KEY (*parameter-name1*)

검색 목표를 정의합니다. 하나의 키에 대해 하나의 검색 목표만 지정할 수 있습니다. *parameter-name1* 값은 정의된 함수의 매개변수 이름을 식별합니다(SQLSTATE 42703 또는 428E8).

*parameter-name1*의 데이터 유형은 인덱스 확장에 지정된 소스 키의 데이터 유형과 일치해야 합니다(SQLSTATE 428EY). 내장 및 구별 데이터 유형에 대해, 그리고 구조화된 유형에 대한 동일한 구조화된 자료형 계층 내에서 정확하게 일치해야 합니다.

이 절은 이름이 지정된 매개변수의 값이 지정된 인덱스 확장을 기반으로 하는 인덱스에서 다뤄지는 컬럼들일 경우에 유용합니다.

USE *search-method-name*(*parameter-name2,...*)

검색 인수를 정의하며, 인덱스 확장에 정의된 검색 메소드에서 사용할 검색 메소드를 식별합니다. *search-method-name*은 인덱스 확장에 정의된 검색 메소드와 일치해야 합니다(SQLSTATE 42743). *parameter-name2* 값은 EXPRESSION AS절에서 정의된 함수의 매개변수 이름이나 *expression-name*을 식별합니다. 이것은 검색 목표에 지정된 매개변수 이름과 달라야 합니다(SQLSTATE 428E9). 매개변수 수와 각 *parameter-name2*의 데이터 유형은 인덱스 확장에서 검색 메소드에 대해 정의된 매개변수와

일치해야 합니다. 내장 및 구별 데이터 유형에 대해, 그리고 구조화된 유형에 대한 동일한 구조화된 자료형 계층 내에서 정확하게 일치해야 합니다.

주

- 하나의 데이터 유형이 다른 데이터 유형으로 캐스트 가능한지를 판별할 때 CHAR 및 DECIMAL과 같은 매개변수화된 데이터 유형에 대해 길이나 정밀도, 스케일은 고려되지 않습니다. 그러므로 함수를 사용할 때 소스 데이터 유형의 값을 목표 데이터 유형의 값으로 캐스트하려고 하면 오류가 발생할 수 있습니다. 예를 들어, VARCHAR은 DATE로 캐스트 가능하나 소스 유형이 실제로 VARCHAR(5)로 정의되면, 함수를 사용할 때 오류가 발생합니다.
- 사용자 정의 함수(UDF)의 매개변수에 대한 데이터 유형을 선택할 때, 입력 값에 영향을 줄 승격 규칙을 고려하십시오(『데이터 유형 승격』 참조). 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀 더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음과 같은 데이터 유형을 사용하는 것이 좋습니다.
  - SMALLINT 대신 INTEGER
  - REAL 대신 DOUBLE
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 플랫폼 간의 UDF 이식성을 위해 다음과 같은 데이터 유형을 사용해서는 안됩니다.
  - FLOAT: DOUBLE 또는 REAL을 대신 사용
  - NUMERIC: DECIMAL을 대신 사용
  - LONG VARCHAR: CLOB(또는 BLOB)를 대신 사용
- 함수와 메소드는 중첩 관계가 될 수 없습니다(SQLSTATE 42745). 겹쳐쓰기에 대한 자세한 내용은 『CREATE TYPE(구조화된)』을 참조하십시오.
- 함수는 메소드와 같은 시그니처를 가질 수 없습니다(함수의 첫 번째 *parameter-type*을 메소드의 *subject-type*과 비교할 경우)(SQLSTATE 42723).
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성할 경우, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가지고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 부여됩니다.
- 파티션된 데이터베이스 환경에서는 외부 사용자 정의 함수 또는 메소드에서 SQL을 사용할 수 없습니다(SQLSTATE 42997).
- NO SQL로 정의된 루틴만 인덱스 확장자를 정의하는 데 사용할 수 있습니다(SQLSTATE 428F8).
- 함수가 SQL을 허용한다면 외부 프로그램이 페더레이티드 오브젝트에 대해 액세스를 시도해서는 안됩니다(SQLSTATE 55047).

- NOT FENCED로 정의된 Java 루틴은 FENCED THREADSAFE로 정의된 것처럼 호출됩니다.
- PARAMETER STYLE DB2GENERAL절이 지정되는 경우, XML 매개변수는 LANGUAGE JAVA 외부 함수에서만 지원됩니다.
- *테이블 액세스 제한사항*

  함수가 READS SQL DATA로 정의되어 있으면 함수의 어떤 명령문도 함수를 호출한 명령문에서 수정 중인 테이블에 액세스할 수 없습니다(SQLSTATE 57053). 예를 들어, 사용자 정의 함수(UDF) BONUS()가 READS SQL DATA로 정의되어 있다고 가정해 보십시오. UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO) 명령문을 호출하면 BONUS 함수에 있는 어떤 SQL문도 EMPLOYEE 테이블에서 데이터를 읽을 수 없습니다.
- *특권*: 함수 정의자는 함수 삭제 권한은 물론 함수에 대한 EXECUTE 특권 WITH GRANT OPTION도 항상 수신합니다.

  SQL문에 함수를 사용할 때는 함수 정의자에게 함수가 사용하는 모든 패키지에 대한 EXECUTE 특권이 있어야 합니다.
- *호환성*: z/OS용 DB2와의 호환성:
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - ASUTIME NO LIMIT
    - NO COLLID
    - PROGRAM TYPE SUB
    - STAY RESIDENT NO
    - 유니코드 데이터베이스의 경우 CCSID UNICODE
    - CCSID ASCII(PARAMETER CCSID UNICODE가 지정되지 않은 경우 비유니코드 데이터베이스)

  이전 버전 DB2와의 호환성을 위해,
  - PARAMETER STYLE SQL 대신 PARAMETER STYLE DB2SQL을 지정할 수 있습니다.
  - DETERMINISTIC 대신 NOT VARIANT를, NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.
  - CALLED ON NULL INPUT 대신 NULL CALL을, RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

예:

예 1: Pellow가 자신의 PELLOW 스키마에 CENTRE 함수를 등록하는 중입니다. 이 키워드를 디폴트값으로 하고, 시스템이 함수에 특정 이름을 제공하십시오.

```
CREATE FUNCTION CENTRE (INT,FLOAT)
  RETURNS FLOAT
  EXTERNAL NAME 'mod!middle'
  LANGUAGE C
       PARAMETER STYLE SQL
       DETERMINISTIC
       NO SQL
  NO EXTERNAL ACTION
```

예 2: McBride(DBADM 권한 보유)가 다른 CENTRE 함수를 PELLOW 스키마에 등록하여, 후속 데이터 정의 언어(DDL)용으로 명시적인 특정 이름을 제공하고 명시적으로 모든 키워드 값을 제공합니다. 이 함수는 스크래치 패드를 사용하며 후속되는 결과에 영향을 주는 데이터를 누적합니다. DISALLOW PARALLEL가 지정되었으므로 함수에 대한 모든 참조는 병렬화되지 않습니다. 따라서 한 번만 초기화를 수행하고 그 결과를 저장하는 데 단일 스크래치 패드가 사용됩니다.

```
CREATE FUNCTION PELLOW.CENTRE (FLOAT, FLOAT, FLOAT)
  RETURNS DECIMAL(8,4) CAST FROM FLOAT
  SPECIFIC FOCUS92
  EXTERNAL NAME 'effects!focalpt'
  LANGUAGE C    PARAMETER STYLE SQL
  DETERMINISTIC   FENCED   NOT NULL CALL   NO SQL   NO EXTERNAL ACTION
  SCRATCHPAD   NO FINAL CALL
  DISALLOW PARALLEL
```

예 3: 다음은 규칙을 구현하기 위해 작성된 C 언어의 사용자 정의 함수 프로그램입니다.

```
output = 2 * input - 4
```

입력이 널(NULL)인 경우, 널(NULL)을 리턴합니다. CREATE FUNCTION문이 NOT NULL CALL을 사용하면, 더 간단하게(즉, 널(NULL) 점검없이) 작성할 수 있습니다. CREATE FUNCTION문은 다음과 같습니다.

```
CREATE FUNCTION ntest1 (SMALLINT)
  RETURNS SMALLINT
  EXTERNAL NAME 'ntest1!nudft1'
  LANGUAGE C    PARAMETER STYLE SQL
  DETERMINISTIC   NOT FENCED   NULL CALL
  NO SQL   NO EXTERNAL ACTION
```

프로그램 코드는 다음과 같습니다.

```
#include "sqlsystm.h"
/* NUDFT1 IS A USER_DEFINED SCALAR FUNCTION */
/* udft1 accepts smallint input
and produces smallint output
implementing the rule:
if (input is null)
set output = null;
else
set output = 2 * input - 4;
*/
void SQL_API_FN nudft1
(short *input,      /* ptr to input arg */
```

```
short *output,     /* ptr to where result goes */
short *input_ind,  /* ptr to input indicator var */
short *output_ind, /* ptr to output indicator var */
char sqlstate[6],  /* sqlstate, allows for null-term */
char fname[28],    /* fully qual func name, nul-term */
char finst[19],    /* func specific name,  null-term */
char msgtext[71])  /* msg text buffer,     null-term */
{
/* first test for null input */
if (*input_ind == -1)
{
/* input is null, likewise output */
*output_ind = -1;
}
else
{
/* input is not null.  set output to 2*input-4 */
*output = 2 * (*input) - 4;
/* and set out null indicator to zero */
*output_ind = 0;
}
/* signal successful completion by leaving sqlstate as is */
/* and exit */
return;
}
/* end of UDF: NUDFT1 */
```

예 4: 다음은 문자열에서 첫 번째 모음의 위치를 리턴하는 Java UDF를 등록합니다. Java에 기록된 UDF는 분리된 채로 실행되며 클래스 javaUDF의 findvwl 메소드입니다.

```
CREATE FUNCTION findv ( CLOB(100K))
  RETURNS INTEGER
  FENCED
  LANGUAGE JAVA
  PARAMETER STYLE JAVA
  EXTERNAL NAME 'javaUDFs.findvwl'
       NO EXTERNAL ACTION
  CALLED ON NULL INPUT
  DETERMINISTIC
       NO SQL
```

예 5: 다음은 SHAPE 유형의 두 매개변수인 g1 및 g2를 입력으로 사용하는 사용자 정의 술어 WITHIN을 간략하게 나타냅니다.

```
CREATE FUNCTION within (g1 SHAPE, g2 SHAPE)
RETURNS INTEGER
LANGUAGE C
PARAMETER STYLE SQL
DETERMINISTIC
NOT FENCED
NO SQL
     NO EXTERNAL ACTION
EXTERNAL NAME 'db2sefn!SDESpatilRelations'
PREDICATES
WHEN = 1
FILTER USING mbrOverlap(g1..xmin, g1..ymin, g1..xmax, g1..max,
```

```
g2..xmin, g2..ymin, g2..xmax, g2..ymax)
     SEARCH BY INDEX EXTENSION gridIndex
WHEN KEY(g1) USE withinExplRule(g2)
WHEN KEY(g2) USE withinExplRule(g1)
```

WITHIN 함수의 설명은 사용자 정의 함수의 설명과 유사하지만, 다음의 추가사항은 이 함수가 사용자 정의 술어에서 사용될 수 있음을 나타냅니다.

- **PREDICATES WHEN = 1**은 이 함수가 DML문의 WHERE절에서 다음과 같이 나타날 때,

  ```
  within(g1, g2) = 1
  ```

  술어가 사용자 정의 술어로 간주되고 인덱스 확장 *gridIndex*에 의해 정의된 인덱스를 사용하여 이 술어를 만족하는 행을 검색해야 함을 나타냅니다. 상수를 지정하면, DML문에서 지정된 상수는 인덱스 작성 명령문에 지정된 상수와 정확하게 일치해야 합니다. 이 조건은 보통 결과 유형이 1 또는 0인 BOOLEAN 표현식을 다루기 위해 제공됩니다. 다른 경우, EXPRESSION절을 선택하는 것이 더 좋습니다.
- **FILTER USING mbrOverlap**은 보다 경제적인 WITHIN 술어 버전인 필터링 함수 mbrOverlap을 의미합니다. 위의 예에서 mbrOverlap 함수는 입력으로 최소 바운드 직사각형을 사용하여 이들이 신속하게 겹쳐지는지 여부를 판별합니다. 두 입력 형태의 최소 바운드 직사각형이 겹쳐지지 않으면, g1은 g2와 함께 포함되지 않습니다. 그러므로 튜플을 안전하게 버릴 수 있어서 비경제적인 WITHIN 술어를 적용하지 않아도 됩니다.
- **SEARCH BY INDEX EXTENSION**절은 인덱스 확장과 검색 목표의 조합이 이 사용자 정의 술어에 사용될 수 있음을 나타냅니다.

예 6: 다음은 유형 POINT의 두 매개변수인 P1 및 P2를 입력으로 사용하는 사용자 정의 술어 DISTANCE를 간략하게 나타냅니다.

```
CREATE FUNCTION distance (P1 POINT, P2 POINT)
  RETURNS INTEGER
  LANGUAGE C
  PARAMETER STYLE SQL
  DETERMINISTIC
  NOT FENCED
  NO SQL
       NO EXTERNAL ACTION
  EXTERNAL NAME 'db2sefn!SDEDistances'
  PREDICATES
  WHEN > EXPRESSION AS distExpr
       SEARCH BY INDEX EXTENSION gridIndex
  WHEN KEY(P1) USE distanceGrRule(P2, distExpr)
  WHEN KEY(P2) USE distanceGrRule(P1, distExpr)
```

DISTANCE 함수의 설명은 사용자 정의 함수의 설명과 유사하지만, 다음의 추가사항은 이 함수가 사용자 정의 술어에서 사용될 수 있음을 나타냅니다.

- **PREDICATES WHEN > EXPRESSION AS distExpr**은 또 다른 유효한 술어 스펙입니다. WHEN절에 표현식이 지정된 경우, 그 표현식의 결과 유형은 술어가 DML문에서 사용자 정의 술어인지를 판별하는 데 사용됩니다. 예를 들면, 다음과 같습니다.

```
SELECT T1.C1
  FROM T1, T2
  WHERE distance (T1.P1, T2.P1) > T2.C2
```

  술어 스펙 distance는 입력으로 두 개의 매개변수를 취하여 결과를 유형이 INTEGER인 T2.C2와 비교합니다. 오른쪽 표현식의 데이터 유형만 문제가 되므로(특정 상수를 사용하는 것과는 반대로), 비교 값으로 와일드카드를 지정하기 위해 CREATE FUNCTION DDL에서 EXPRESSION절을 선택하는 것이 좋습니다.

  또한 다음도 유효한 사용자 정의 술어입니다.

```
SELECT T1.C1
  FROM T1, T2
  WHERE distance(T1.P1, T2.P1) > distance (T1.P2, T2.P2)
```

  현재, 오른쪽만 표현식으로 처리되는 제한사항이 있습니다. 왼쪽의 용어는 사용자 술어에 대한 사용자 정의 함수입니다.

- **SEARCH BY INDEX EXTENSION**절은 인덱스 확장과 검색 목표의 조합이 이 사용자 정의 술어에 사용될 수 있음을 나타냅니다. distance 함수의 경우, distExpr로 식별되는 표현식도 range-producer 함수(인덱스 확장의 일부로 정의됨)에 전달되는 검색 인수 중 하나입니다. 표현식 ID는 range-producer 함수에 인수로 전달되도록 표현식의 이름을 정의하는 데 사용됩니다.

CREATE FUNCTION(외부 테이블)

CREATE FUNCTION(외부 테이블)문은 현재 서버에 사용자 정의 외부 테이블 함수를 등록하는 데 사용됩니다.

*테이블 함수*는 SELECT의 FROM절에서 사용할 수 있으며, 한 번에 한 행씩 리턴하여 SELECT로 테이블을 리턴합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스의 CREATE_EXTERNAL_ROUTINE 권한은 다음 중 최소한 하나를 포함해야 합니다.
  - 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 함수의 스키마 이름이 존재할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

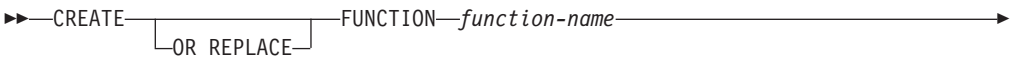
비분리 함수를 작성하려면, 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

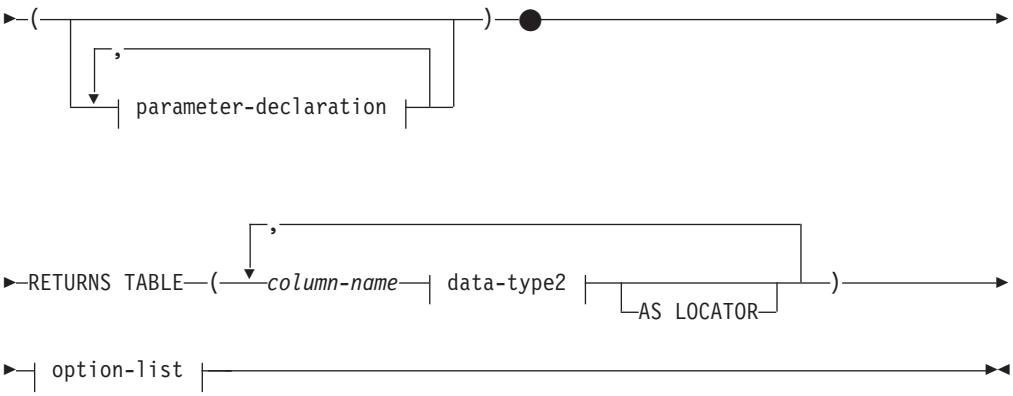
- 데이터베이스에서의 CREATE_NOT_FENCED_ROUTINE 권한
- DBADM 권한

분리 함수를 작성하려면, 추가 권한이나 특권이 필요하지 않습니다.

기존 함수를 교체하려면, 명령문의 권한 부여 ID가 기존 함수의 소유자여야 합니다(SQLSTATE 42501).

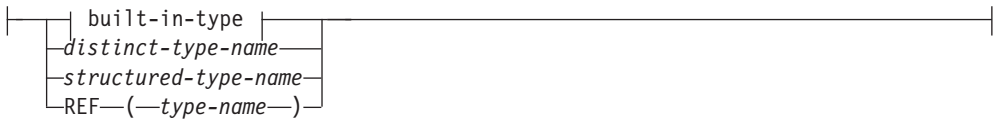
구문



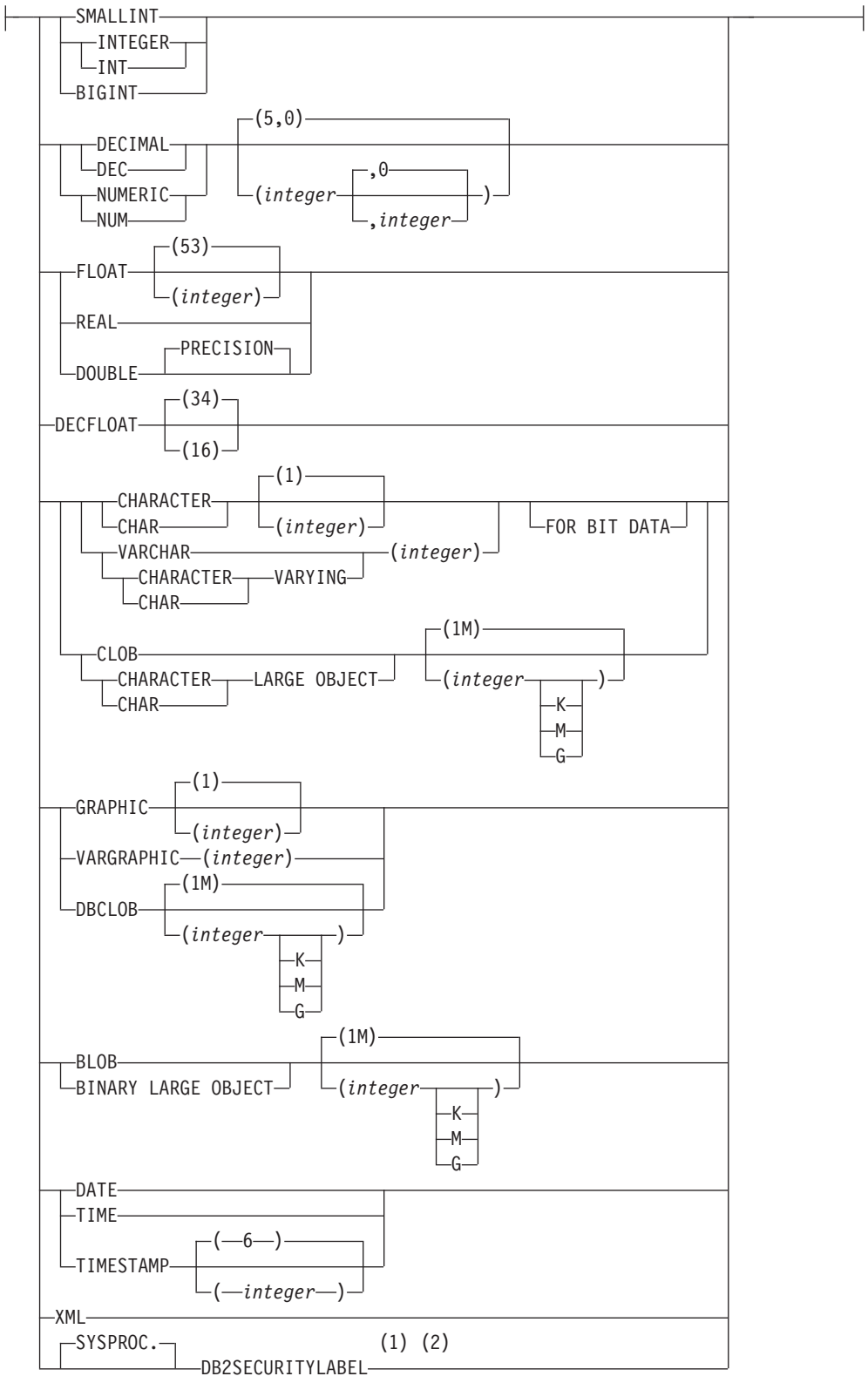


parameter-declaration:

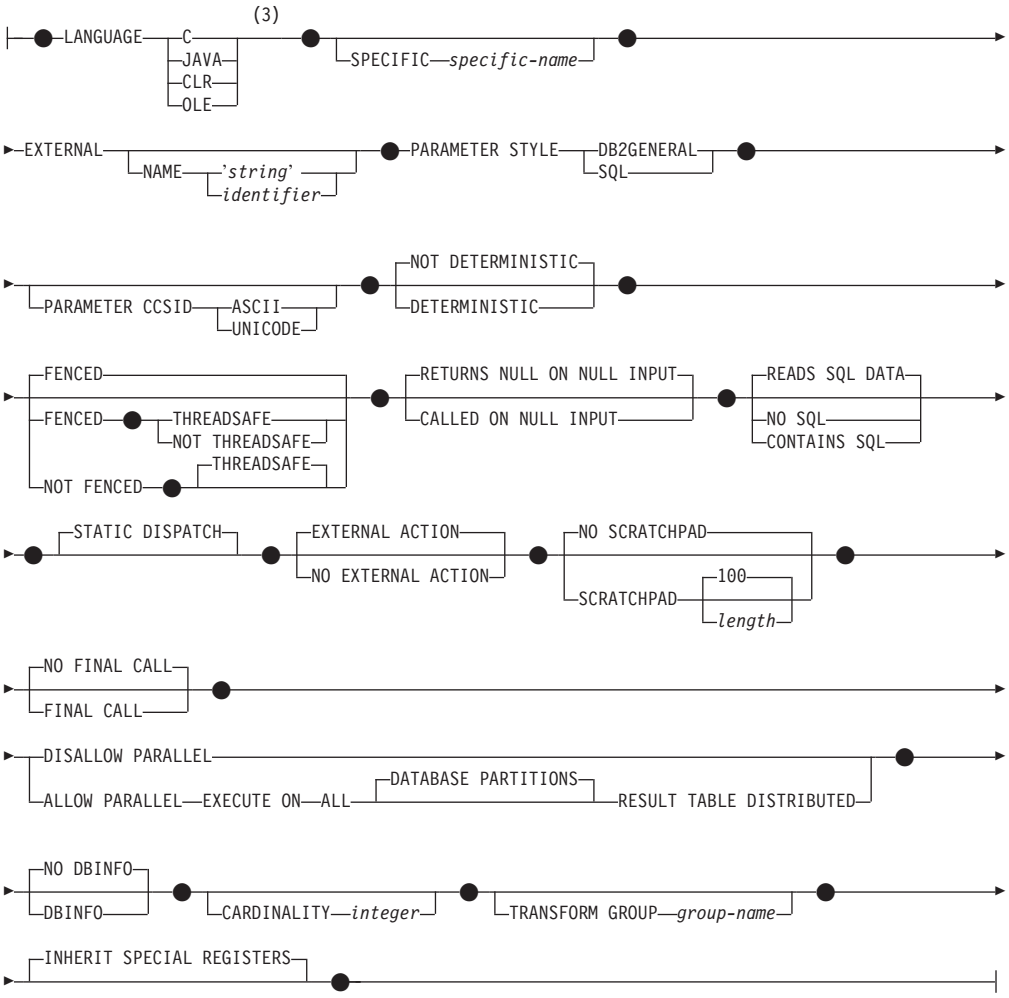
data-type1, data-type2:



built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
,integer)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
FOR BIT DATA
VARCHAR
(integer)
CHARACTER
VARYING
CHAR
CLOB
LARGE OBJECT
(1M)
K
M
G
GRAPHIC
VARGRAPHIC
DBCLOB
BLOB
BINARY LARGE OBJECT
DATE
TIME
TIMESTAMP
(6)
XML
SYSPROC.
DB2SECURITYLABEL
(1) (2)

option-list:



주:

1 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

2 DB2SECURITYLABEL 유형의 컬럼의 경우, NOT NULL WITH DEFAULT는 내재적 및 명시적으로 지정될 수 없습니다(SQLSTATE 42842). DB2SECURITYLABEL 유형의 컬럼 디폴트값은 쓰기 액세스에 대한 세션 권한 부여 ID의 보안 레이블입니다.

3 LANGUAGE OLE DB 외부 테이블 함수 작성에 대한 정보는 『CREATE FUNCTION(OLE DB 외부 테이블)』을 참조하십시오. LANGUAGE SQL 테이블 함수 작성에 대한 정보는 『CREATE FUNCTION(SQL 스칼라, 테이블 또는 행)』을 참조하십시오.

설명

OR REPLACE

함수 정의가 현재 서버에 존재하는 경우 이를 교체하도록 지정합니다. 함수에 대해 부여된 특권에 영향을 주지 않는다는 점을 제외하고 새 정의가 카탈로그에서 교체

되기 전에 기존 정의가 효과적으로 삭제됩니다. 함수 정의가 현재 서버에 없으면 이 옵션은 무시됩니다. 기존 함수를 교체하려면, 새 정의의 특정 이름 및 함수 이름이 이전 정의의 특정 이름 및 함수 이름과 동일하거나 새 정의의 시그니처가 이전 정의의 시그니처와 일치해야 합니다. 그렇지 않으면, 새 함수가 작성됩니다.

function-name

정의되는 함수의 이름을 지정하십시오. 이 이름은 함수를 지정하는 규정화되거나 규정되지 않은 이름입니다. 규정되지 않은 *function-name* 양식은 SQL ID입니다(최대 길이는 128). 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 규정된 이름은 첫 번째 매개변수가 구조화된 유형일 경우 그 첫 번째 매개변수의 데이터 유형과 같으면 안됩니다.

매개변수 수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고) 내재적 또는 명시적 규정자를 포함한 이름은 카탈로그에 기술된 함수를 식별해서는 안됩니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 구성된 이름을 지정할 경우, *schema-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *function-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름 및 비교 연산자가 이에 해당합니다.

함수 시그니처에 약간의 차이가 있으면, 여러 함수에 동일한 이름을 사용할 수 있습니다. 이에 대한 제한이 없다고 하여도, 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안됩니다.

(*parameter-declaration***,...)**

함수의 입력 매개변수 수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수가 받게 될 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 90개의 매개변수만 허용됩니다(SQLSTATE 54023).

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예를 들면, 다음과 같습니다.

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 이름이 지정된 두 함수는 모든 해당 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서

고려되지 않습니다. 따라서 CHAR(8) 및 CHAR(35)는 동일한 유형으로 간주되며, DECIMAL(11,2) 및 DECIMAL(4,3)도 마찬가지입니다. 유니코드 데이터베이스의 경우, CHAR(13) 및 GRAPHIC(8)은 같은 유형으로 간주됩니다. 유형을 번들하여 위의 목적용으로 이를 동일한 유형으로 처리할 수 있습니다(예: DECIMAL 및 NUMERIC) 시그니처가 중복되면 오류가 리턴됩니다(SQLSTATE 42723).

parameter-name

입력 매개변수의 선택적 이름을 지정합니다. 이름은 매개변수 목록의 기타 *parameter-name*과 동일할 수 없습니다(SQLSTATE 42734).

data-type1

입력 매개변수의 데이터 유형을 지정합니다. 데이터 유형은 내장 데이터 유형, 구별 유형, 구조화된 유형 또는 참조 유형일 수 있습니다. 각 내장 데이터 유형에 대한 자세한 설명은 『CREATE TABLE』을 참조하십시오. 일부 데이터 유형은 모든 언어로 지원되지는 않습니다. SQL 데이터 유형과 호스트 언어 데이터 유형 간 맵핑에 대한 자세한 내용은 『Embedded SQL 응용프로그램에서 SQL 데이터 유형에 맵핑되는 데이터 유형』을 참조하십시오.

- 날짜 시간 유형 매개변수는 문자 데이터 유형으로 전달되며 데이터는 ISO 형식으로 전달됩니다.
- DECIMAL(및 NUMERIC)은 LANGUAGE C와 OLE에 대해 유효하지 않습니다(SQLSTATE 42815).
- XML은 LANGUAGE OLE에서 유효하지 않습니다.
- 함수 내에서 보이는 XML 값은 함수 호출의 매개변수로 전달되는 XML 값의 병렬 버전이므로 XML 유형의 매개변수는 `XML AS CLOB(n)` 구문을 사용하여 선언되어야 합니다.
- CLR은 28보다 큰 DECIMAL 스케일은 지원하지 않습니다(SQLSTATE 42613).
- 배열 유형을 지정할 수 없습니다(SQLSTATE 42815).

사용자 정의 구별 유형의 경우, 매개변수의 길이, 정밀도 또는 스케일 속성은 구별 유형의 소스 유형의 속성입니다(CREATE TYPE에 지정된 속성). 구별 유형 매개변수는 구별 유형의 소스 유형으로 전달됩니다. 구별 유형의 이름이 규정되지 않은 경우, 데이터베이스 관리 프로그램은 SQL 경로의 스키마를 검색하여 스키마 이름을 분석합니다.

사용자 정의 구조화된 유형의 경우, 연관된 변환 그룹에 해당 변환 함수가 존재해야 합니다.

참조 유형의 경우, 매개변수 범위가 지정되지 않으면 매개변수를 REF(*type-name*)로 지정할 수 있습니다.

AS LOCATOR

매개변수 값의 로케이터가 실제 값 대신 함수로 전달되도록 지정합니다.

LOB 데이터 유형 또는 LOB 데이터 유형을 기반으로 하는 구별 유형 매개변수에 대해서만 AS LOCATOR를 지정하십시오(SQLSTATE 42601). 값 대신 로케이터를 전달하면 함수로 전달되는 바이트 수를 줄일 수 있습니다(특히, 매개변수 값이 아주 큰 경우).

AS LOCATOR절은 데이터 유형을 승격할 수 있는지 여부를 판별하는 데 영향을 주지 않으며, 함수 결정에서 사용되는 함수 시그니처에도 영향을 주지 않습니다.

함수가 FENCED이고 NO SQL 옵션을 가질 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

RETURNS TABLE

함수의 결과가 테이블이 되도록 지정합니다. 이 키워드 뒤에 오는 괄호는 테이블 컬럼의 이름과 유형의 목록을 구분하므로, 추가 스펙(예: 제한사항)이 없는 단순 CREATE TABLE문의 양식과 유사합니다. 최대 255개의 컬럼이 허용되지 않습니다(SQLSTATE 54011).

column-name

이 컬럼의 이름을 지정합니다. 이름은 규정할 수 없으며 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

data-type2

컬럼의 데이터 유형을 지정하며, 구조화된 유형을 제외하고 특정 언어로 작성된 UDF의 매개변수에 대해 지원되는 데이터 유형이면 됩니다(SQLSTATE 42997).

AS LOCATOR

*data-type2*가 LOB 유형이거나 LOB 유형을 기반으로 하는 구별 유형일 경우, 이 옵션을 사용하면 함수는 결과 테이블에서 인스턴스화된 LOB 값에 대한 로케이터를 리턴함을 나타냅니다.

이 절에 사용할 수 있는 유효한 유형은 『CREATE FUNCTION(외부 스칼라)』에 설명되어 있습니다.

SPECIFIC *specific-name*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. 규정되지 않은 *specific-name* 양식은 SQL ID입니다(최대 길이는 128). 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함한 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *function-name*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *function-name*에 대해 사용된 규정자가 사용됩니다. 규정자를 지정할 경우, 규정자는 *function-name*의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

EXTERNAL

이 절은 CREATE FUNCTION문이 내부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 링크 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됩니다.

NAME절을 지정하지 않을 경우, "NAME *function-name*"이 사용됩니다.

NAME *'string'*

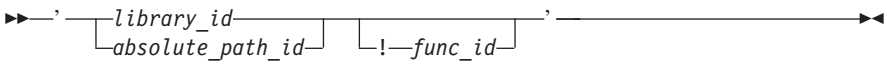
이 절은 정의되는 함수를 구현하는 사용자 작성 코드를 식별합니다.

'string' 옵션은 최대 길이가 254바이트인 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다.

- LANGUAGE C의 경우:

  지정된 *string*은 라이브러리 이름 및 라이브러리 내의 함수이고, 이것은 데이터베이스 관리 프로그램이 작성한 사용자 정의 함수를 실행하기 위해 호출하는 것입니다. CREATE FUNCTION문이 실행될 때 라이브러리(및 라이브러리 내의 함수)는 존재할 필요가 없습니다. 그러나 함수가 SQL문에서 사용될 때, 라이브러리와 해당 라이브러리 내의 함수가 존재해야 하고 데이터베이스 서버 머신에서 접근할 수 있어야 합니다.

  *string*을 다음과 같이 정의할 수 있습니다.



  작은따옴표 내에서 공백은 허용되지 않습니다.

  library_id

  함수를 포함하는 라이브러리 이름을 식별합니다. 데이터베이스 관리 프로그램은 다음과 같이 라이브러리를 찾습니다.

  - UNIX 시스템에서 'myfunc'가 *library_id*로 지정되고 데이터베이스 관리 프로그램이 /u/production에서 실행 중인 경우, 데이터베이스 관리 프로그램은 /u/production/sqllib/function/myfunc 라이브러리에서 함수를 찾습니다.
  - Windows 운영 체제에서 데이터베이스 관리 프로그램은 LIBPATH 또는 PATH 환경 변수에서 지정하는 디렉토리에서 함수를 찾습니다.

absolute_path_id
함수를 포함하는 파일의 전체 경로 이름을 나타냅니다.

예를 들어, UNIX 시스템에서 '/u/jchui/mylib/myfunc'는 데이터베이스 관리 프로그램이 /u/jchui/mylib에서 myfunc 공유 라이브러리를 찾게 합니다.

Windows 운영 체제에서 'd:₩mylib₩myfunc.dll'은 데이터베이스 관리 프로그램이 d:₩mylib 디렉토리에서 동적 링크 라이브러리 myfunc.dll을 로드하게 합니다. 절대 경로 ID를 사용하여 루틴 본문을 식별할 경우, .dll 확장자를 추가해야 합니다.

! func_id
호출될 함수의 시작점 이름을 식별합니다. !은 library ID와 function ID 사이의 분리 문자로 사용됩니다.

예를 들어, UNIX 시스템에서 'mymod!func8'은 데이터베이스 관리 프로그램이 $inst_home_dir/sqllib/function/mymod 라이브러리를 찾아 해당 라이브러리에서 시작점 func8을 사용하도록 지시합니다.

Windows 운영 체제에서 'mymod!func8'은 데이터베이스 관리 프로그램이 mymod.dll 파일을 로드하고 동적 링크 라이브러리(DLL)의 func8() 함수를 호출하게 합니다.

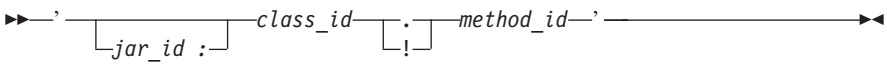
문자열이 적절히 구성되지 않으면, 오류가 발생합니다(SQLSTATE 42878).

모든 경우, 모든 외부 함수의 본문이 모든 데이터베이스 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

- LANGUAGE JAVA의 경우:

  지정된 *string*에는 선택적 jar 파일 ID, 클래스 ID 및 메소드 ID가 포함되는데, 이는 작성되는 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 것입니다. CREATE FUNCTION문이 실행될 때 클래스 ID 및 메소드 ID는 존재하지 않아도 됩니다. *jar_id*를 지정할 경우, CREATE FUNCTION문이 실행될 때 존재해야 합니다. 그러나 함수가 SQL 문에서 사용될 경우, 메소드 ID가 존재해야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다.

  *string*을 다음과 같이 정의할 수 있습니다.



  작은따옴표 내에서 공백은 허용되지 않습니다.

jar_id

데이터베이스에 설치될 때, jar 콜렉션에 제공된 jar ID를 식별합니다. 간단한 ID 또는 스키마 규정된 ID일 수 있습니다. 'myJar' 및 'mySchema.myJar'가 그 예입니다.

class_id

Java 오브젝트의 클래스 ID를 나타냅니다. 클래스가 패키지의 일부일 경우, 클래스 ID 부분에 완전한 패키지 접두어(예: 'myPacks.UserFuncs') 가 포함되어야 합니다. JVM(Java Virtual Machine)은 클래스를 디렉토리 '.../myPacks/UserFuncs/'에서 찾게 됩니다. Windows 32비트 운영 체제에서 Java Virtual Machine은 '...₩myPacks₩UserFuncs₩' 디렉토리에서 찾습니다.

method_id

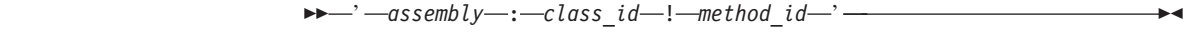
호출할 Java 오브젝트의 메소드 이름을 식별합니다.

- LANGUAGE CLR의 경우:

지정된 *string*은 .NET 어셈블리(라이브러리 또는 실행 파일), 해당 어셈블리에 있는 클래스 및 작성되는 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 클래스에 있는 메소드를 표시합니다. CREATE FUNCTION 문이 실행될 때 모듈, 클래스 및 메소드는 없어도 됩니다. 그러나 함수가 SQL 문에서 사용될 경우 모듈, 클래스 및 메소드가 존재해야 하고 데이터베이스 서버 머신에서 액세스할 수 있어야 하며, 그렇지 않을 경우 오류가 리턴됩니다(SQLSTATE 42724).

관리 코드 확장자가 포함된 것을 표시하도록 '/clr' 컴파일러 옵션으로 컴파일된 C++ 루틴은 'LANGUAGE C'가 아닌 'LANGUAGE CLR'로 카탈로그되어야 합니다. DB2는 필요한 런타임 결정을 하기 위해 .NET 인프라스트럭처가 사용자 정의 함수에서 활용되고 있는지 알아야 합니다. .NET 인프라스트럭처를 사용하는 모든 사용자 정의 함수(UDF)는 'LANGUAGE CLR'로 카탈로그되어야 합니다.

*string*을 다음과 같이 정의할 수 있습니다.



이름은 작은따옴표로 묶어야 합니다. 공백은 허용되지 않습니다.

assembly

클래스가 있는 DLL 또는 다른 어셈블리 파일을 식별합니다. 파일 확장자(예: .dll)를 지정해야 합니다. 전체 경로 이름을 지정하지 않은 경우, 파일은 DB2 설치 경로의 함수 디렉토리(예: c:₩sqllib₩function)에 상주해야 합니다. 파일이 설치 함수 디렉토리의 서브디렉토리에 상주할 경우,

전체 경로를 지정하는 대신에 서브디렉토리를 파일 이름 전에 지정할 수 있습니다. 예를 들어, 설치 디렉토리가 c:₩sqllib이고 어셈블리 파일이 c:₩sqllib₩function₩myprocs₩mydotnet.dll일 경우, 어셈블리에 대해 'myprocs₩mydotnet.dll'만 지정하면 됩니다. 이 매개변수의 대소문자 구분은 파일 시스템의 대소문자 구분과 동일합니다.

class_id

호출될 메소드가 있는 지정된 어셈블리 내에 있는 클래스의 이름을 지정합니다. 클래스가 이름 스페이스에 상주할 경우, 전체 이름 스페이스를 클래스와 같이 지정해야 합니다. 예를 들어, EmployeeClass 클래스가 MyCompany.ProcedureClasses 이름 스페이스에 있으면 클래스에 대해 MyCompany.ProcedureClasses.EmployeeClass를 지정해야 합니다. 일부 .NET 언어용 컴파일러는 프로젝트 이름을 클래스의 이름 스페이스로 추가하는데, 명령행 컴파일러가 사용되었는지 또는 GUI 컴파일러가 사용되었는지에 따라 동작이 다를 수 있습니다. 이 매개변수는 대소문자가 구분됩니다.

method_id

호출될 지정된 클래스 내에 있는 메소드를 지정합니다. 이 매개변수는 대소문자가 구분됩니다.

- LANGUAGE OLE의 경우:

지정된 *stringv*은 OLE 프로그램 가능 ID(progid) 또는 클래스 ID(clsid) 및 메소드 ID로서, 작성중인 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출한 것입니다. CREATE FUNCTION문이 실행될 때 프로그램가능 ID 또는 클래스 ID 및 메소드 ID는 존재하지 않아도 됩니다. 그러나 함수가 SQL문에서 사용될 경우 메소드 ID가 존재해야 하고 데이터베이스 서버 머신에서 액세스할 수 있어야 하며, 그렇지 않을 경우 오류가 리턴됩니다(SQLSTATE 42724).

*string*을 다음과 같이 정의할 수 있습니다.

작은따옴표 내에서 공백은 허용되지 않습니다.

progid

OLE 오브젝트의 프로그램가능 ID를 식별합니다.

*progid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 시 OLE API로 전달됩니다. 지정된 OLE 오브젝트는 작성 가능해야 하며 후기 바인딩(IDispatch형 바인딩이라고도 함)을 지원해야 합니다.

clsid

작성할 OLE 오브젝트의 클래스 ID를 식별합니다. OLE 오브젝트가 progid를 사용하여 등록되지 않은 경우 *progid*를 지정하기 위한 대안으로 사용할 수 있습니다. *clsid*의 형식은 다음과 같습니다.

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

여기서, 'n'은 영숫자입니다. *clsid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임시 OLE API로 전달됩니다.

method_id

호출할 OLE 오브젝트의 메소드 이름을 식별합니다.

NAME *identifier*

이 절은 정의되는 함수를 구현하는 사용자 작성 코드의 이름을 식별합니다. 지정된 *identifier*는 SQL ID입니다. SQL ID는 문자열에서 *library id*로 사용됩니다. 분리 ID가 아닐 경우, ID는 대문자로 겹쳐집니다. ID에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 형식의 이름은 LANGUAGE C에서만 사용할 수 있습니다.

LANGUAGE

이 필수 절은 사용자 정의 함수의 본문이 작성되는 언어 인터페이스 규칙을 지정하는 데 사용됩니다.

C 데이터베이스 관리 프로그램은 사용자 정의 함수를 C 함수인 것처럼 호출합니다. 사용자 정의 함수는 표준 ANSI C 프로토타입에 의해 정의된 대로 C 언어 호출 규칙과 링크 규칙을 따라야 합니다.

JAVA 데이터베이스 관리 프로그램이 사용자 정의 함수를 Java 클래스의 메소드로 호출함을 의미합니다.

CLR 데이터베이스 관리 프로그램이 사용자 정의 함수를 .NET 클래스의 한 메소드로 호출합니다. 이 시점에서 LANGUAGE CLR은 Windows 운영 체제에서 실행되는 사용자 정의 함수에 대해서만 지원됩니다. CLR 루틴에 대해서는 NOT FENCED를 지정할 수 없습니다(SQLSTATE 42601).

OLE 데이터베이스 관리 프로그램은 사용자 정의 함수를 OLE 자동화 오브젝트에 의해 표시되는 메소드인 것처럼 호출합니다. 사용자 정의 함수(UDF)는 *OLE 자동화 프로그래머 참조서*에 설명된 대로 OLE 자동화 데이터 유형 및 호출 메커니즘에 따라야 합니다.

LANGUAGE OLE는 Windows 32비트 운영 체제용 DB2에 저장된 사용자 정의 함수에 대해서만 지원됩니다.

LANGUAGE OLE DB 외부 테이블 함수 작성에 대한 자세한 내용은 『CREATE FUNCTION(OLE DB 외부 테이블)』을 참조하십시오.

PARAMETER STYLE

이 절은 함수로부터 값을 리턴하고 매개변수를 전달하기 위한 규칙을 지정하는 데 사용됩니다.

DB2GENERAL

Java 클래스의 메소드로 정의된 외부 함수에 매개변수를 전달하고 함수로부터 값을 리턴하는 규칙을 지정하는 데 사용됩니다. 이는 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

SQL

C 언어 호출 및 연결 규칙, OLE 자동화 오브젝트에 의해 표시되는 메소드 또는 .NET 오브젝트의 공용 정적 메소드를 준수하는 외부 함수에 매개변수를 전달하고 외부 함수로부터 값을 리턴하기 위해 사용하는 규칙을 지정하는 데 사용됩니다. 이는 LANGUAGE C, LANGUAGE CLR 또는 LANGUAGE OLE를 사용할 때 지정해야 합니다.

PARAMETER CCSID

함수에 전달되거나 함수에서 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII

문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031). 함수가 호출될 때 함수의 응용프로그램 코드 페이지는 데이터베이스 코드 페이지입니다.

UNICODE

문자열 데이터를 유니코드로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, 문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있습니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, 문자 데이터는 UTF-8로 되어 있습니다. 어떠한 경우라도 함수가 호출될 때 함수의 응용프로그램 코드 페이지는 1208입니다.

데이터베이스가 유니코드 데이터베이스가 아니고 PARAMETER CCSID UNICODE가 있는 함수가 작성될 경우, 함수에는 그래픽 유형 또는 사용자 정의 유형이 있을 수 없습니다(SQLSTATE 560C1).

데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE로 테이블 함수를 작성할 수 있지만 다음 규칙이 적용됩니다.

- 테이블 함수를 작성하기 전에 데이터베이스 구성에 대체 조합 시퀀스를 지정해야 합니다(SQLSTATE 56031). PARAMETER CCSID UNICODE 테이블 함수는 데이터베이스 구성에 지정된 대체 조합 시퀀스와 조합됩니다.

- CCSID ASCII로 작성된 테이블 또는 테이블 함수와 CCSID UNICODE로 작성된 테이블 또는 테이블 함수 모두 단일 SQL문에서는 사용할 수 없습니다(SQLSTATE 53090). 이 사항은 명령문에서 직접 참조된 테이블 및 테이블 함수는 물론 간접적으로 참조된(예: 참조 무결성 제한조건, 트리거, 구체화된 쿼리 테이블 및 뷰 본문에 있는 테이블을 통해서) 테이블 및 테이블 함수에도 적용됩니다.
- PARAMETER CCSID UNICODE로 작성된 테이블 함수는 SQL 함수 또는 SQL 메소드에서 참조할 수 없습니다(SQLSTATE 560C0).
- PARAMETER CCSID UNICODE로 작성된 테이블 함수를 참조하는 SQL 문은 SQL 함수 또는 SQL 메소드를 호출할 수 없습니다(SQLSTATE 53090).
- 그래픽 유형, XML 유형 및 사용자 정의 유형은 PARAMETER CCSID UNICODE 테이블 함수에 대한 매개변수로 사용할 수 없습니다(SQLSTATE 560C1).
- PARAMETER CCSID UNICODE 테이블 함수를 참조하는 명령문은 DB2 버전 8.1 이상 클라이언트에서만 호출할 수 있습니다(SQLSTATE 42997).
- SQL문은 언제나 데이터베이스 코드 페이지에서 해석됩니다. 특히, 이것은 리터럴, 16진 리터럴 및 분리 ID의 모든 문자에 대한 표현이 데이터베이스 코드 페이지에 있어야 한다는 것을 의미합니다. 그렇지 않을 경우, 문자는 대체 문자로 교체됩니다.

데이터베이스가 유니코드 데이터베이스가 아니고 데이터베이스 구성에 대체 조합 시퀀스가 지정된 경우, 함수는 PARAMETER CCSID ASCII 또는 PARAMETER CCSID UNICODE로 작성될 수 있습니다. 함수에 전달되거나 함수로부터 전달되는 모든 문자열 데이터는 해당 코드 페이지로 변환됩니다.

이 절은 LANGUAGE OLE, LANGUAGE JAVA 또는 LANGUAGE CLR로 지정할 수 없습니다(SQLSTATE 42613).

DETERMINISTIC 또는 NOT DETERMINISTIC

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC을 지정하여 방지할 수 있습니다. 비결정 테이블 함수의 한 예는 테이블 함수 결과 테이블에 영향을 주는 방식으로 특수 레지스터, 전역 변수, 비결정 함수 또는 시퀀스를 참조하는 함수입니다.

FENCED 또는 NOT FENCED

이 절은 함수가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 어드레스 스페이스에서 실행하기에 FENCED 또는 NOT FENCED를 지정합니다.

함수가 FENCED로 등록된 경우, 데이터베이스 관리 프로그램은 함수에서 액세스하지 못하도록 내부 자원(예: 데이터 버퍼)을 보호합니다. 대부분의 함수는 FENCED나 NOT FENCED로 실행됩니다. 일반적으로 FENCED로 수행되는 함수는 NOT FENCED로 실행되는 함수와는 다르게 실행됩니다.

주의:

제대로 코딩, 검토 및 테스트되지 않은 함수에 NOT FENCED를 사용하면 DB2 데이터베이스의 무결성이 손상될 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 사용자 정의 함수가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

LANGUAGE OLE 또는 NOT THREADSAFE의 함수에 대해서는 FENCED만 지정할 수 있습니다(SQLSTATE 42613).

함수가 FENCED이고 NO SQL 옵션을 가질 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

함수를 NOT FENCED로 등록하기 위해서는 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED_ROUTINE)이 필요합니다.

NOT FENCED절을 지정하면 LANGUAGE CLR 사용자 정의 함수(UDF)를 작성할 수 없습니다(SQLSTATE 42601).

THREADSAFE 또는 NOT THREADSAFE

함수가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 아니면 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

함수가 OLE가 아닌 다른 LANGUAGE를 사용하여 정의된 경우에는 다음과 같습니다.

- 함수가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 함수를 호출할 수 있습니다. 일반적으로 스레드가 안전하려면 함수가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED와 NOT FENCED 함수는 둘다 THREADSAFE될 수 있습니다.
- 함수가 NOT THREADSAFE로 정의되어 있으면, 데이터베이스 관리 프로그램이 다른 루틴과 동일한 프로세스에서 함수를 동시에 호출할 수 없습니다.

FENCED 함수의 경우, LANGUAGE가 JAVA 또는 CLR이면 THREADSAFE가 디폴트값입니다. 다른 모든 언어에서는 NOT THREADSAFE가 디폴트값입니다. 이 함수가 LANGUAGE OLE로 정의된 경우, THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

NOT FENCED 함수의 경우 THREADSAFE가 디폴트값입니다. NOT THREADSAFE는 지정될 수 없습니다(SQLSTATE 42613).

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 널(NULL) 값인 인수가 있을 경우 외부 함수로의 호출을 방지하기 위해 사용됩니다. 사용자 정의 함수(UDF)가 매개변수를 갖지 않는 것으로 정의되면, 이러한 널(NULL) 인수 조건이 발생할 수 없으며 이 스펙의 코딩 방법이 문제가 되지 않습니다.

RETURNS NULL ON NULL INPUT이 지정되고 테이블 함수 OPEN 시기에 함수 인수 중 어느 하나가 널(NULL)일 경우, 사용자 정의 함수는 호출되지 않습니다. 시도된 테이블 함수 스캔의 결과는 빈 테이블(행이 없는 테이블)입니다.

CALLED ON NULL INPUT이 지정될 경우, 인수가 널(NULL)인지에 관계없이 사용자 정의 함수가 호출됩니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나 널(NULL) 값 인수 값에 대한 테스트 책임은 UDF에 있습니다.

이전 버전과의 호환성 및 제품군 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로 NULL CALL을 사용할 수도 있습니다. 마찬가지로, NOT NULL CALL을 RETURNS NULL ON NULL INPUT에 대한 동의어로 사용할 수도 있습니다.

NO SQL, CONTAINS SQL, READS SQL DATA

함수가 SQL문을 발행하는지 여부 및 발행할 경우 SQL문의 유형을 나타냅니다.

NO SQL

함수가 SQL문을 실행할 수 없음을 나타냅니다(SQLSTATE 38001). ALLOW PARALLEL, EXECUTE ON ALL DATABASE PARTITIONS 및 RESULT TABLE DISTRIBUTED문이 모두 지정되는 경우, 유일하게 허용되는 옵션은 NO SQL입니다.

CONTAINS SQL

SQL 데이터를 읽지도 수정하지도 않는 SQL문이 함수에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 38004 또는 42985). 함수에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문이 이 함수에 포함될 수 있음을 나타냅니다(SQLSTATE 38002 또는 42985). 함수에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

STATIC DISPATCH

이 선택적 절은 함수 결정시 DB2가 함수 매개변수의 정적 유형(선언된 유형)에 기반하는 함수를 선택한다는 것을 나타냅니다.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하는지의 여부를 지정합니다. 외부 조치의 예로는 메시지 전송 또는 파일에 레코드 쓰기를 들 수 있습니다. 디폴트값은 EXTERNAL ACTION입니다.

EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하도록 지정합니다.

외부 조치가 있는 함수는 병렬 태스크가 해당 함수를 실행할 경우 부정확한 결과를 리턴할 수 있습니다. 예를 들어 함수가 자신에 대한 각 초기 호출에 대해 메모를 전송할 경우, 함수에 대해 한 번의 메모를 전송하는 대신에 각 병렬 태스크에 대해 하나의 메모를 전송합니다. 정확하게 병렬 처리와 작동하지 않는 함수에 대해 DISALLOW PARALLEL절을 지정하십시오.

NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하지 않는다는 것을 지정합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다.

NO SCRATCHPAD 또는 SCRATCHPAD *length*

이 선택적 절은 외부 함수에 대해 스크래치 패드가 제공되는지 여부를 지정할 때 사용할 수 있습니다. 사용자 정의 함수는 재입력이 가능하도록 적극 권장되므로, 스크래치 패드는 함수가 한 호출에서 다음 호출로의 "상태를 저장"하기 위한 수단을 제공합니다.

SCRATCHPAD를 지정하면, 사용자 정의 함수를 처음 호출할 때 외부 함수에서 사용할 스크래치 패드에 대해 메모리가 할당됩니다. 이 스크래치 패드는 다음 등록 정보를 갖습니다.

- *length* - 지정될 경우, 스크래치 패드의 크기를 바이트 단위로 설정합니다. 이 값은 1 - 32 767이어야 합니다(SQLSTATE 42820). 디폴트값은 100입니다.
- 모두 X'00''으로 초기화됩니다.
- 범위는 SQL문입니다. SQL문에서 외부 함수에 대한 참조당 하나의 스크래치 패드가 있습니다. 그래서 다음 명령문의 UDFX 함수가 SCRATCHPAD 키워드를 사용하여 정의된 경우, 두 개의 스크래치 패드가 지정됩니다.

```
SELECT A.C1, B.C2
  FROM TABLE (UDFX(:hv1)) AS A,
       TABLE (UDFX(:hv1)) AS B
    WHERE ...
```

- 지속적입니다. 이는 명령문의 실행 초기에 초기화되며, 호출간에 스크래치 패드의 상태를 유지하기 위해 외부 테이블 함수에 의해 사용될 수 있습니다. 사용자 정의 함수에 대해서도 FINAL CALL 키워드가 지정되면, 스크래치 패드는 절대 DB2에 의해 변경되지 않으며 스크래치패드에 할당된 모든 자원은 특수 FINAL 호출이 발행되면 해제되야 합니다.

  DB2가 OPEN 호출시마다 스크래치패드를 다시 초기화하기 때문에, NO FINAL CALL이 지정되었거나 디폴트값으로 설정되면 외부 테이블 함수는 CLOSE 호출시 이러한 자원을 지웁니다. 테이블 함수가 서브쿼리나 조인에 사용될 경우에는 FINAL CALL 또는 NO FINAL CALL 및 스크래치 패드의 관련 동작을 판별하는 것이 매우 중요한데, 이유는 이 때가 명령문 실행 중에 여러 개의 OPEN 호출이 발생할 수 있기 때문입니다.
- 외부 함수가 획득할 수 있는 시스템 자원(예: 메모리)에 대한 중심점으로 사용됩니다. 함수는 첫 번째 호출에서 메모리를 획득하고, 그 주소를 스크래치 패드에 보관한 후 다음 호출에서 이를 참조할 수 있습니다.

  (위에서 간단히 설명했듯이, FINAL CALL/NO FINAL CALL 키워드는 스크래치 패드의 재 초기화를 제어하는 데 사용되며, 외부 테이블 함수가 스크래치 패드에 할당된 자원을 해제하는 시기도 말해줍니다.)

SCRATCHPAD를 지정하면, 사용자 정의 함수를 호출할 때마다 추가 인수가 스크래치 패드를 지정하는 외부 함수로 전달됩니다.

NO SCRATCHPAD를 지정하면, 어떤 스크래치 패드도 외부 함수에 할당되거나 전달되지 않습니다.

FINAL CALL 또는 NO FINAL CALL

이 선택적 절은 외부 함수에 대해 마지막 호출(및 별도의 첫 번째 호출)이 수행되는지 여부를 지정합니다. 또한 스크래치 패드가 다시 초기화되는 시기도 제어합니다. NO FINAL CALL을 지정하면, DB2는 테이블 함수에 대해 세 가지 유형의 호출(열기, 페치 및 닫기)만 수행할 수 있습니다. 단, FINAL CALL을 지정하면, 테이블 함수에 대해 열기, 페치 및 닫기 외에도 첫 번째 호출 및 마지막 호출을 수행할 수 있습니다.

외부 테이블 함수의 경우, 어떤 옵션을 선택하는지에 관계없이 호출 유형 인수는 항상 존재합니다.

인터럽트나 트랜잭션의 끝으로 인해 마지막 호출이 이루어질 경우 UDF는 CLOSE 커서를 제외한 다른 커서에 대해 SQL문을 발행할 수 없습니다(SQLSTATE 38505). 특수하게 마지막 호출이 이루어지는 이런 상황에서는 ″호출 유형″ 인수에 특수 값이 전달됩니다.

DISALLOW PARALLEL 또는 ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

함수에 대한 단일 참조에 대해 함수의 호출이 병렬화될 수 있는지 지정합니다.

DISALLOW PARALLEL

함수 호출시마다 DB2가 단일 데이터베이스 파티션의 함수를 호출하도록 지정합니다.

ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

함수 호출시마다 DB2가 모든 데이터베이스 파티션의 함수를 호출하도록 지정합니다. 각 데이터베이스 파티션의 확보된 결과 세트의 통합이 리턴됩니다. 함수는 SQL문을 실행할 수 없으며 NO SQL절도 지정되어야 합니다.

NO DBINFO 또는 DBINFO

이 선택적 절은 DB2에 알려진 고유한 특정 정보가 추가 호출시간 인수로 함수에 전달되는 지(DBINFO) 또는 전달되지 않는 지(NO DBINFO)를 지정합니다. NO DBINFO가 디폴트값입니다. DBINFO는 LANGUAGE OLE에 대해 지원하지 않습니다(SQLSTATE 42613).

DBINFO를 지정하면 다음 정보가 있는 구조가 함수에 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름
- 응용프로그램 ID - 각 데이터베이스 연결에 대해 설정되는 고유한 응용프로그램 ID
- 응용프로그램 권한 부여 ID - 이 함수와 응용프로그램간의 중첩된 함수에 관계없는 응용프로그램 런타임 권한 부여 ID
- 코드 페이지 - 데이터베이스 코드 페이지
- 스키마 이름 - 외부 테이블 함수에는 적용되지 않음
- 테이블 이름 - 외부 테이블 함수에는 적용되지 않음
- 컬럼 이름 - 외부 테이블 함수에는 적용되지 않음
- 데이터베이스 버전 또는 릴리스 - 함수를 호출하는 데이터베이스 서버의 버전, 릴리스 버전 또는 수정 레벨
- 플랫폼 - 서버의 플랫폼 유형
- 테이블 함수 결과 컬럼 번호 - 함수를 참조하는 명령문이 사용하는 결과 컬럼 번호의 배열로, 이 정보는 함수가 모든 컬럼 값 대신 필요한 컬럼 값만을 리턴하도록 함
- 데이터베이스 파티션 번호 - 외부 테이블 함수가 호출되는 외부 테이블 함수의 데이터베이스 파티션 번호로, 단일 데이터베이스 파티션 환경에서는 0값이 됨

CARDINALITY *integer*

이 선택적 절은 최적화를 위해 함수가 리턴하는 예상 행의 수를 제공합니다. 유효한 *integer* 범위 값은 0 - 9,223,372,036,854,775,807입니다.

테이블 함수에 대해 CARDINALITY절이 지정되지 않은 경우 DB2는 유한 값을 디폴트값으로 가정하는 데, 이 값은 RUNSTATS 유틸리티가 통계를 수집하지 않은 테이블에 추정되는 값과 동일합니다.

경고: 함수가 무한 카디낼리티(cardinality)를 가지고 있는 경우, 즉, 요청될 때마다 행을 리턴하고 절대 ″테이블의 끝″ 조건을 리턴하지 않는 경우, 올바르게 작동하기 위해 ″테이블의 끝″ 조건이 필요한 쿼리는 무한하게 되어 인터럽트되어야 합니다. 이러한 쿼리의 예는 GROUP BY 또는 ORDER BY절이 포함된 쿼리입니다. 이러한 UDF의 작성은 권장되지 않습니다.

TRANSFORM GROUP *group-name*

함수를 호출할 때 사용자 정의된, 구조화된 유형 변환에 사용할 변환 그룹을 나타냅니다. 변환은 함수 정의가 사용자 정의 구조화된 유형을 매개변수로 포함할 경우에 필요합니다. 이 절을 지정하지 않으면, 디폴트 그룹 이름인 DB2_FUNCTION이 사용됩니다. 지정된 (또는 디폴트값인) *group-name*이 참조된 구조화된 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42741). 필수 FROM SQL 변환 함수가 제공된 그룹 이름과 구조화된 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42744).

INHERIT SPECIAL REGISTERS

이 선택적 절은 함수에 있는 갱신 가능한 특수 레지스터가 레지스터의 초기값을 호출 명령문의 환경에서 상속하도록 지정합니다. 커서의 select문에서 호출되는 함수의 경우에는 초기값을 커서가 열려 있는 환경에서 상속합니다. 중첩 오브젝트(예: 트리거 또는 뷰)에서 호출되는 루틴의 경우에는 초기값을 오브젝트 정의에서 상속하지 않고 런타임 환경에서 상속합니다.

특수 레지스터에 대한 변경사항은 함수 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 갱신할 수 없는 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 디폴트값으로 설정됩니다.

규칙

- 파티션된 데이터베이스 환경에서는 외부 사용자 정의 함수 또는 메소드에서 SQL을 사용할 수 없습니다(SQLSTATE 42997).
- NO SQL로 정의된 루틴만 인덱스 확장자를 정의하는 데 사용할 수 있습니다(SQLSTATE 428F8).
- 함수가 SQL을 허용한다면 외부 프로그램이 페더레이티드 오브젝트에 대해 액세스를 시도해서는 안됩니다(SQLSTATE 55047).
- ***테이블 액세스 제한사항*** 함수가 READS SQL DATA로 정의되어 있으면 함수의 명령문은 함수를 호출한 명령문이 수정하고 있는 테이블에 액세스할 수 없습니다

(SQLSTATE 57053). 예를 들어, 사용자 정의 함수(UDF) BONUS()가 READS SQL DATA로 정의되어 있다고 가정해 보십시오. UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO) 명령문을 호출하면 BONUS 함수에 있는 어떤 SQL문도 EMPLOYEE 테이블에서 데이터를 읽을 수 없습니다.

주

- 사용자 정의 함수(UDF)의 매개변수에 대한 데이터 유형을 선택할 때, 입력 값에 영향을 주게 될 승격 규칙을 고려하십시오. 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀 더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음과 같은 데이터 유형을 사용하는 것이 좋습니다.
  - SMALLINT 대신 INTEGER
  - REAL 대신 DOUBLE
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 플랫폼 간의 UDF 이식성을 위해 다음 데이터 유형을 사용할 것을 권장합니다.
  - FLOAT 대신 DOUBLE 또는 REAL
  - NUMERIC 대신 DECIMAL
  - LONG VARCHAR 대신 CLOB(또는 BLOB)
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성할 경우, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가지고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 부여됩니다.
- NOT FENCED로 정의된 Java 루틴은 FENCED THREADSAFE로 정의된 것처럼 호출됩니다.
- *특권*: 함수 정의자는 함수 삭제 권한은 물론 함수에 대한 EXECUTE 특권 WITH GRANT OPTION도 항상 수신합니다. SQL문에 함수를 사용할 때는 함수 정의자에게 함수가 사용하는 모든 패키지에 대한 EXECUTE 특권이 있어야 합니다.
- *호환성*: z/OS용 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - ASUTIME NO LIMIT
    - NO COLLID
    - PROGRAM TYPE SUB
    - STAY RESIDENT NO
    - 유니코드 데이터베이스의 경우 CCSID UNICODE

- CCSID ASCII(PARAMETER CCSID UNICODE가 지정되지 않은 경우 비 유니코드 데이터베이스)

이전 버전 DB2와의 호환성을 위해,

– PARAMETER STYLE SQL 대신 PARAMETER STYLE DB2SQL을 지정할 수 있습니다.

– DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있습니다.

– NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.

– CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.

– RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

– DB2GENERAL 대신 DB2GENRL을 지정할 수 있습니다.

예:

예 1: 텍스트 관리 시스템에 알려진 문서에 대해 단일 문서 ID 컬럼으로 이루어진 행을 리턴하는 데 기록된 테이블 함수를 등록하십시오. 첫 번째 매개변수는 주어진 주제 영역과 일치하고 두 번째 매개변수는 주어진 문자열을 포함합니다.

단일 세션의 컨텍스트에서 UDF는 항상 같은 테이블을 리턴합니다. 그래서 DETERMINISTIC으로 정의됩니다. DOCMATCH에서 출력을 정의한 RETURNS절에 유의하십시오. FINAL CALL은 각 테이블 함수에 대해 지정되어야 합니다. 또한 테이블 함수는 병렬로 작동할 수 없으므로 DISALLOW PARALLEL 키워드가 추가됩니다. DOCMATCH에 대한 출력 크기가 매우 가변적일지라도 CARDINALITY 20이 대표 값이므로 DB2 옵티마이저를 지원하기 위해 지정됩니다.

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOC_ID CHAR(16))
  EXTERNAL NAME '/common/docfuncs/rajiv/udfmatch'
  LANGUAGE C
  PARAMETER STYLE SQL
  NO SQL
  DETERMINISTIC
       NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

예 2: 다음은 Microsoft Exchange 메시지의 부분 텍스트 메시지 및 메시지 헤더 정보를 검색하는 데 사용된 OLE 테이블 함수를 등록합니다.

```
CREATE FUNCTION MAIL()
  RETURNS TABLE (TIMERECEIVED DATE,
                                      SUBJECT VARCHAR(15),
                                      SIZE INTEGER,
                                      TEXT VARCHAR(30))
```

```
EXTERNAL NAME 'tfmail.header!list'
LANGUAGE OLE
PARAMETER STYLE SQL
NOT DETERMINISTIC
FENCED
CALLED ON NULL INPUT
SCRATCHPAD
FINAL CALL
NO SQL
EXTERNAL ACTION
DISALLOW PARALLEL
```

CREATE FUNCTION(OLE DB 외부 테이블)

CREATE FUNCTION(OLE DB 외부 테이블)문은 OLE DB Provider로부터 데이터를 액세스할 수 있게 사용자 정의 OLE DB 외부 테이블 함수를 등록하는 데 사용됩니다.

*테이블 함수*는 SELECT의 FROM절에서 사용될 수도 있습니다.

호출

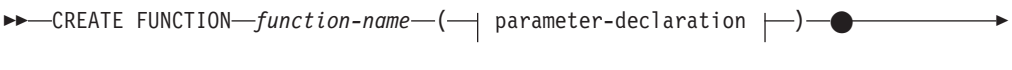
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

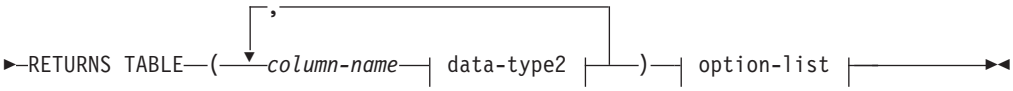
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스의 CREATE_EXTERNAL_ROUTINE 권한은 다음 중 최소한 하나를 포함해야 합니다.
  - 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 함수의 스키마 이름이 존재할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

구문

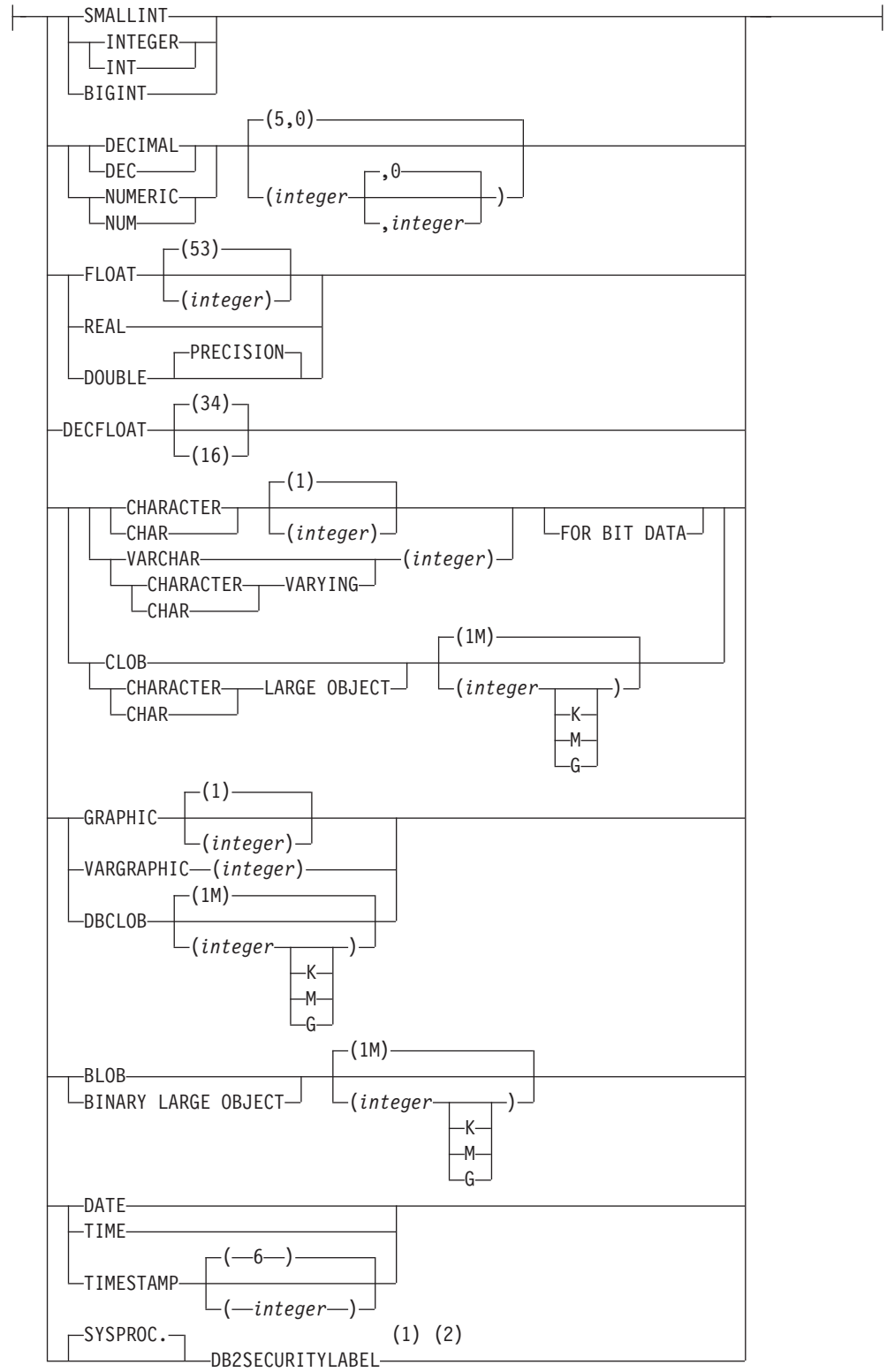




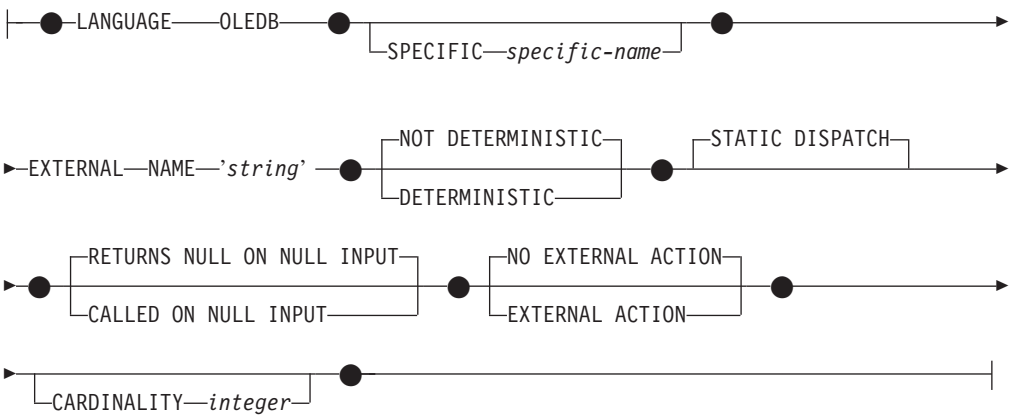
parameter-declaration:

data-type1, data-type2:

내장 유형:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
,integer)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
FOR BIT DATA
VARCHAR
(integer)
CHARACTER
VARYING
CHAR
CLOB
CHARACTER
LARGE OBJECT
CHAR
(1M)
(integer
K
M
G)
GRAPHIC
(1)
(integer)
VARGRAPHIC
(integer)
DBCLOB
(1M)
(integer
K
M
G)
BLOB
BINARY LARGE OBJECT
(1M)
(integer
K
M
G)
DATE
TIME
TIMESTAMP
(6)
(integer)
SYSPROC.
DB2SECURITYLABEL
(1) (2)

option-list:



주:

1 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

2 DB2SECURITYLABEL 유형의 컬럼의 경우, NOT NULL WITH DEFAULT는 내재적 및 명시적으로 지정될 수 없습니다(SQLSTATE 42842). DB2SECURITYLABEL 유형의 컬럼 디폴트값은 쓰기 액세스에 대한 세션 권한 부여 ID의 보안 레이블입니다.

설명

function-name

정의될 함수의 이름을 지정합니다. 이 이름은 함수를 지시하는 규정화되거나 규정되지 않는 이름입니다. *function-name*의 규정되지 않은 형식은 SQL ID(최대 길이 : 18)입니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다.

매개변수 수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고) 내재적 또는 명시적 규정자를 포함한 이름은 카탈로그에 기술된 함수를 식별해서는 안됩니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 구성된 이름을 지정할 경우, *schema-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *function-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL,

NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

함수 시그니처에 약간의 차이가 있으면, 여러 함수에 동일한 이름을 사용할 수 있습니다. 이에 대한 제한이 없다고 하여도, 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안됩니다.

(*parameter-declaration***,...)**

함수의 입력 매개변수를 식별하며, 매개변수의 데이터 유형을 지정합니다. 입력 매개변수가 지정되지 않으면, 쿼리 최적화를 통해 서브세트화된 외부 소스로부터 데이터가 검색됩니다. 입력 매개변수는 명령 텍스트를 OLE DB Provider에 전달합니다.

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예를 들면, 다음과 같습니다.

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 이름이 지정된 두 함수는 모든 해당 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 그러므로 CHAR(8) 및 CHAR(35)이 같은 유형으로 간주됩니다. 유니코드 데이터베이스의 경우, CHAR(13) 및 GRAPHIC(8)은 같은 유형으로 간주됩니다. 시그니처가 중복되면 오류가 리턴됩니다(SQLSTATE 42723).

parameter-name

입력 매개변수에 대한 선택적 이름을 지정합니다.

data-type1

입력 매개변수의 데이터 유형을 지정합니다. 데이터 유형은 임의의 문자 또는 그래픽 문자열 데이터 유형 또는 문자나 그래픽 문자열 데이터 유형을 기반으로 하는 구별 유형일 수 있습니다. XML 유형의 매개변수는 지원되지 않습니다(SQLSTATE 42815).

각 내장 데이터 유형에 대한 자세한 설명은 『CREATE TABLE』을 참조하십시오.

사용자 정의 구별 유형의 경우, 매개변수의 길이, 정밀도나 스케일 속성은 구별 유형의 소스 유형의 것과 같습니다(CREATE TYPE에서 지정됨). 구별 유형 매개변수는 구별 유형의 소스 유형으로 전달됩니다. 구별 유형의 이름이 규정되지 않으면, 데이터베이스 관리 프로그램은 SQL 경로에서 스키마를 검색하여 스키마 이름을 분석합니다.

RETURNS TABLE

함수의 결과가 테이블이 되도록 지정합니다. 이 키워드 뒤에 오는 괄호는 테이블 컬럼의 이름과 유형의 목록을 구분하므로, 추가 스펙(예: 제한사항)이 없는 단순 CREATE TABLE문의 양식과 유사합니다.

column-name

해당 행 세트 컬럼 이름과 동일해야 하는 컬럼의 이름을 지정합니다. 이름은 규정할 수 없으므로 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

data-type2

컬럼의 데이터 유형을 지정합니다. XML은 유효하지 않습니다(SQLSTATE 42815).

SPECIFIC *specific-name*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *specific-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 18)입니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함한 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *function-name*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *function-name*에 대해 사용된 규정자가 사용됩니다. 규정자를 지정할 경우, 규정자는 *function-name*의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

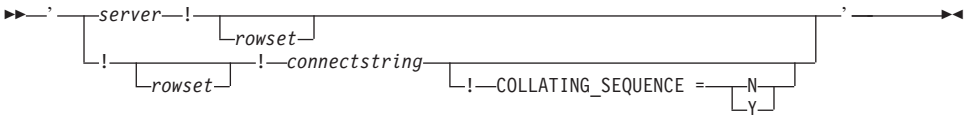
EXTERNAL NAME *'string'*

이 절은 외부 테이블 및 OLE DB Provider를 식별합니다.

'string' 옵션은 최대 길이가 254바이트인 문자열 상수입니다.

지정된 문자열은 OLE DB Provider와의 연결 및 세션을 형성하고, rowset으로부터 데이터를 검색하는 데 사용됩니다. CREATE FUNCTION문이 실행될 때 OLE DB Provider와 데이터 소스가 존재할 필요는 없습니다.

*string*을 다음과 같이 정의할 수 있습니다.



server

『CREATE SERVER』에서 정의한 대로 데이터 소스의 로컬 이름을 식별합니다.

rowset

OLE DB Provider에 의해 표시되는 rowset(테이블)을 식별합니다. 카탈로그나 스키마 이름을 지원하는 OLE DB Provider에 완전한 테이블 이름을 제공해야 합니다.

connectstring

데이터 소스에 연결하는 데 필요한 초기화 등록 정보의 문자열 버전입니다. 연결 문자열의 기본 형식은 ODBC 연결 문자열에 기초합니다. 문자열은 세미콜론으로 구분되는 일련의 키워드/값 쌍을 포함합니다. 등호(=)는 각 키워드와 해당되는 값을 구분합니다. 키워드는 OLE DB 초기화 등록 정보(등록 정보 세트 DBPROPSET_DBINIT) 또는 공급자 고유 키워드에 대한 설명입니다.

COLLATING_SEQUENCE

데이터 소스가 Linux, UNIX 및 Windows용 DB2 Database와 동일한 조합 시퀀스를 사용하는지의 여부를 지정합니다. 자세한 내용은 『CREATE SERVER』를 참조하십시오. 유효한 값은 다음과 같습니다.

- Y = 동일한 조합 시퀀스
- N = 다른 조합 시퀀스

COLLATING_SEQUENCE가 지정되지 않으면, 데이터 소스에 Linux, UNIX 및 Windows용 DB2 Database와 다른 조합 시퀀스가 있는 것으로 가정합니다.

*server*를 제공하면, *connectstring* 또는 COLLATING_SEQUENCE를 외부 이름에서 사용할 수 없습니다. 이들은 서버 옵션 CONNECTSTRING 및 COLLATING_SEQUENCE로 정의됩니다. *server*를 제공하지 않으면, *connectstring*을 제공해야 합니다. *rowset*을 제공하지 않으면, 명령 텍스트를 통해 OLE DB Provider에 전달하는 입력 매개변수가 테이블 함수에 있어야 합니다.

LANGUAGE OLEDB

데이터베이스 관리 프로그램이 OLE DB Provider로부터 데이터를 검색하는 내장 일반 OLE DB 사용자를 전개할 것임을 의미합니다. 개발자는 테이블 함수를 구현할 필요가 없습니다.

LANGUAGE OLEDB 테이블 함수는 모든 플랫폼에서 작성될 수 있으나 Microsoft OLE DB에 의해 지원되는 플랫폼에서만 실행될 수 있습니다.

DETERMINISTIC 또는 NOT DETERMINISTIC

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다.

STATIC DISPATCH

이 선택적 절은 함수 결정시 DB2가 함수 매개변수의 정적 유형(선언된 유형)에 기반하는 함수를 선택한다는 것을 나타냅니다.

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 널(NULL) 값인 인수가 있을 경우 외부 함수로의 호출을 방지하기 위해 사용됩니다. 사용자 정의 함수(UDF)가 매개변수를 갖지 않는 것으로 정의되면, 이 널(NULL) 인수 조건이 발생할 수 없습니다.

RETURNS NULL ON NULL INPUT이 지정되고 실행시 함수 인수 중 하나가 널(NULL)일 경우, 사용자 정의 함수는 호출되지 않으며 결과는 빈 테이블(즉, 행이 없는 테이블)이 됩니다.

CALLED ON NULL INPUT이 지정될 경우, 실행시 인수가 널(NULL)인지에 관계없이 사용자 정의 함수가 호출됩니다. 자체 논리에 따라 빈 테이블을 리턴하거나 그렇지 않을 수 있습니다. 그러나 널(NULL) 값 인수 값에 대한 테스트 책임은 UDF에 있습니다.

이전 버전과의 호환성 및 제품군 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로 NULL CALL을 사용할 수도 있습니다. 마찬가지로, NOT NULL CALL을 RETURNS NULL ON NULL INPUT에 대한 동의어로 사용할 수도 있습니다.

NO EXTERNAL ACTION 또는 EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하는지의 여부를 지정합니다. 외부 조치의 예로는 메시지 전송 또는 파일에 레코드 쓰기를 들 수 있습니다. 디폴트값은 NO EXTERNAL ACTION입니다.

NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하지 않는다는 것을 지정합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다.

EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하도록 지정합니다.

CARDINALITY *integer*

이 선택적 절은 최적화를 위해 함수가 리턴하는 예상 행의 수를 제공합니다. *integer*에 대한 유효한 값의 범위는 0 - 2,147,483,647까지입니다.

테이블 함수에 대해 CARDINALITY절이 지정되지 않은 경우 DB2는 유한 값을 디폴트값으로 가정하는 데, 이 값은 RUNSTATS 유틸리티가 통계를 수집하지 않은 테이블에 추정되는 값과 동일합니다.

경고: 함수가 무한 카디널리티(cardinality)를 가지고 있는 경우, 즉, 요청될 때마다 행을 리턴하고 절대 "테이블의 끝" 조건을 리턴하지 않는 경우, 올바르게 작동하기 위해 "테이블의 끝" 조건이 필요한 쿼리는 무한하게 되어 인터럽트되어야 합니다. 이러한 쿼리의 예는 GROUP BY 또는 ORDER BY절이 포함된 쿼리입니다. 이러한 UDF의 작성은 권장되지 않습니다.

주

- FENCED, FINAL CALL, SCRATCHPAD, PARAMETER STYLE SQL, DISALLOW PARALLEL, NO DBINFO, NOT THREADSAFE 및 NO SQL은 명령문에 내재되어 있으며, 지정할 수 있습니다.
- 사용자 정의 함수(UDF)의 매개변수에 대한 데이터 유형을 선택할 때, 입력 값에 영향을 주게 될 승격 규칙을 고려하십시오. 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀 더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음과 같은 데이터 유형을 사용하는 것이 좋습니다.
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 플랫폼 간의 UDF 이식성을 위해 다음 데이터 유형을 사용할 것을 권장합니다.
  - FLOAT 대신 DOUBLE 또는 REAL
  - NUMERIC 대신 DECIMAL
  - LONG VARCHAR 대신 CLOB(또는 BLOB)
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성할 경우, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가지고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- *특권*: 함수 정의자는 항상 함수 삭제 권한 뿐만 아니라, 함수에 대한 EXECUTE 특권 WITH GRANT OPTION도 부여받습니다.
- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  - DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있습니다.
  - NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.
  - CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.
  - RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

예:

예 1: 다음은 Microsoft Access 데이터베이스로부터 정보를 검색하는 OLE DB 테이블 함수를 등록합니다. 연결 문자열은 외부 이름에 정의됩니다.

```
CREATE FUNCTION orders ()
  RETURNS TABLE (orderid INTEGER,
                                    customerid CHAR(5),
                                    employeeid INTEGER,
                                    orderdate TIMESTAMP,
                                    requireddate TIMESTAMP,
                                    shippeddate TIMESTAMP,
                                    shipvia INTEGER,
                                    freight dec(19,4))
  LANGUAGE OLEDB
  EXTERNAL NAME '!orders!Provider=Microsoft.Jet.OLEDB.3.51;
                        Data Source=c:WsqllibWsamplesWoledbWnwind.mdb
  !COLLATING_SEQUENCE=Y';
```

예 2: Oracle 데이터베이스에서 고객 정보를 검색하는 OLE DB 테이블 함수를 등록하십시오. 연결 문자열은 서버 정의를 통해 제공됩니다. 테이블 이름은 외부 이름에 완전하게 규정됩니다. 로컬 사용자 john이 리모트 사용자 dave에 맵핑됩니다. 다른 사용자들은 연결 문자열에 게스트 사용자 ID를 사용합니다.

```
CREATE SERVER spirit
  WRAPPER OLEDB
  OPTIONS (CONNECTSTRING 'Provider=MSDAORA;Persist Security Info=False;
                                User ID=guest;password=pwd;Locale Identifier=1033;
                                OLE DB Services=CLIENTCURSOR;Data Source=spirit');

CREATE USER MAPPING FOR john
  SERVER spirit
  OPTIONS (REMOTE_AUTHID 'dave', REMOTE_PASSWORD 'mypwd');

CREATE FUNCTION customers ()
  RETURNS TABLE (customer_id INTEGER,
                                    name  VARCHAR(20),
                                    address VARCHAR(20),
                                    city VARCHAR(20),
                                    state VARCHAR(5),
                zip_code INTEGER)
  LANGUAGE OLEDB
  EXTERNAL NAME 'spirit!demo.customer';
```

예 3: MS SQL Server 7.0 데이터베이스에서 점포 정보를 검색하는 OLE DB 테이블 함수를 등록하십시오. 연결 문자열은 외부 이름에서 제공됩니다. 테이블 함수는 명령 텍스트를 통해 OLE DB Provider에게 전달하는 입력 매개변수를 가집니다. rowset 이름은 외부 이름에 지정할 필요가 없습니다. 쿼리 예는 상위 세 개의 점포를 검색하는 SQL문 텍스트에 전달됩니다.

```
CREATE FUNCTION favorites (varchar(600))
  RETURNS TABLE (store_id CHAR (4),
                                    name  VARCHAR (41),
                                    sales INTEGER)
  SPECIFIC favorites
  LANGUAGE OLEDB
  EXTERNAL NAME '!!Provider=SQLOLEDB.1;Persist Security Info=False;
                          User ID=sa;Initial Catalog=pubs;Data Source=WALTZ;
                          Locale Identifier=1033;Use Procedure for Prepare=1;
                          Auto Translate=False;Packet Size=4096;Workstation ID=WALTZ;
                          OLE DB Services=CLIENTCURSOR;';

SELECT *
  FROM TABLE (favorites
             (' select top 3 sales.stor_id as store_id, ' CONCAT
                                  ' stores.stor_name as name, ' CONCAT
                                  ' sum(sales. qty) as sales ' CONCAT
              ' from sales, stores ' CONCAT
                             ' where sales.stor_id = stores.stor_id ' CONCAT
                             ' group by sales.stor_id, stores.stor_name ' CONCAT
                             ' order by sum(sales.qty) desc ')) as f;
```

CREATE FUNCTION(소스 또는 템플리트)

CREATE FUNCTION(소스 또는 템플리트)문은 다음을 수행하는 데 사용됩니다.

- 기존의 다른 스칼라 함수나 집계 함수를 기반으로 현재 서버에서 사용자 정의 함수(UDF)를 등록합니다.
- 함수 템플리트를 페더레이티드 서버로 지정된 응용프로그램 서버(AS)에 등록합니다. *함수 템플리트*는 실행 가능 코드를 포함하고 있지 않는 부분적인 함수입니다. 사용자는 데이터 소스 함수에 맵핑하기 위해 이를 작성합니다. 맵핑이 작성된 후, 사용자는 페더레이티드 서버로 제출된 쿼리에서 함수 템플리트를 지정할 수 있습니다. 그러한 쿼리가 처리될 때, 페더레이티드 서버는 템플리트가 맵핑되는 데이터 소스 함수를 호출하고, 그 데이터 유형이 템플리트 정의의 RETURNS 부분에 해당하는 값을 리턴합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

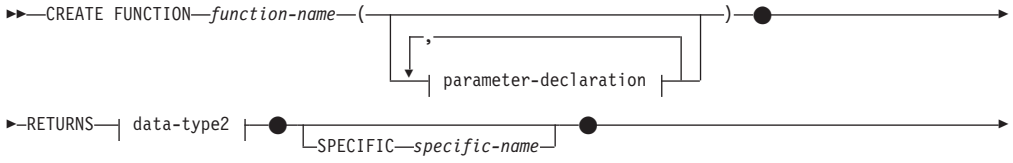
권한 부여

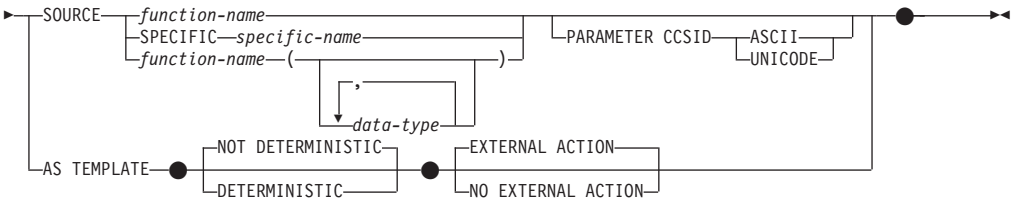
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 존재할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

명령문의 권한 부여 ID에 DATAACCESS 권한이 없고 SOURCE절이 지정된 경우 명령문의 권한 부여 ID가 보유한 특권에는 소스 함수에 대한 EXECUTE 특권도 포함되어야 합니다.

구문

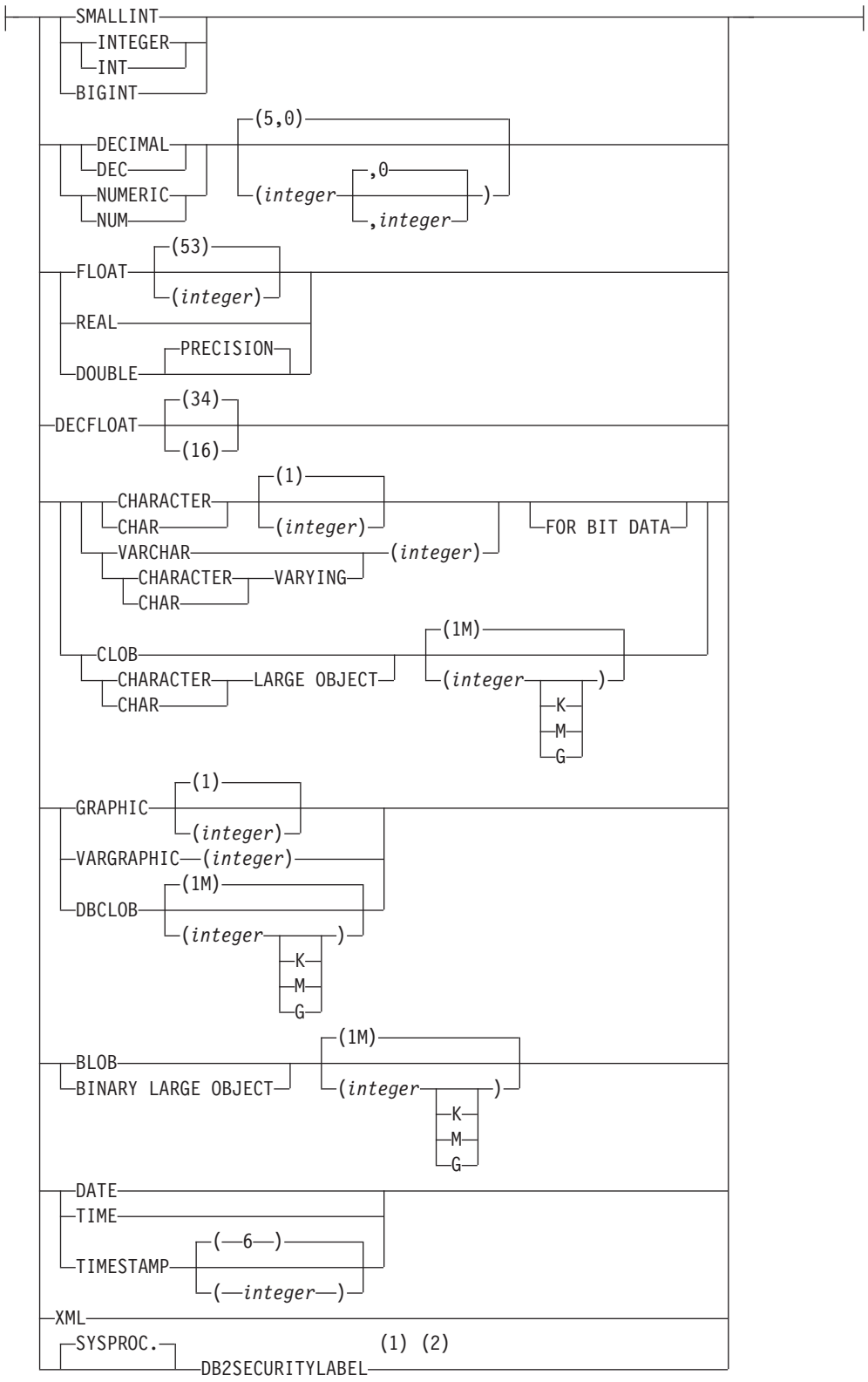




parameter-declaration:

data-type1, data-type2:

built-in-type:



주:

1 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

2 DB2SECURITYLABEL 유형의 컬럼의 경우, NOT NULL WITH DEFAULT는 내재적 및 명시적으로 지정될 수 없습니다(SQLSTATE 42842). DB2SECURITYLABEL 유형의 컬럼 디폴트값은 쓰기 액세스에 대한 세션 권한 부여 ID의 보안 레이블입니다.

설명

function-name

함수 또는 정의될 함수 템플리트 이름. 이 이름은 함수를 지시하는 규정화되거나 규정되지 않는 이름입니다. *function-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 18)입니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수나 함수 템플리트를 식별하지 않아야 합니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 구성된 이름을 지정할 경우, *schema-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *function-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름 및 비교 연산자가 이에 해당합니다.

사용자 정의 구별 유형을 갖는 함수를 지원하기 위해 기존 함수를 근거로 하는 사용자 정의 함수를 이름 지정할 때, 전래 함수와 같은 이름이 사용될 수 있습니다. 이로 인해, 사용자는 추가 정의가 필요하다는 것을 인식하지 않고 사용자가 정의하는 구별 유형을 갖는 동일한 함수를 사용할 수 있습니다. 일반적으로 함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다.

(*parameter-declaration***,...)**

함수 또는 함수 템플리트의 입력 매개변수 개수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수 또는 함수 템플리트 받기를 기대하는 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 90개의 매개변수만 허용됩니다(SQLSTATE 54023).

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예를 들면, 다음과 같습니다.

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 이름이 지정된 두 함수는 모든 해당 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 이 제한사항은 동일한 스키마에서 동일한 이름을 가진 함수 및 함수 템플리트에도 적용됩니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 따라서 CHAR(8) 및 CHAR(35)는 동일한 유형으로 간주되며, DECIMAL(11,2) 및 DECIMAL(4,3)도 마찬가지입니다. 유니코드 데이터베이스의 경우, CHAR(13) 및 GRAPHIC(8)은 같은 유형으로 간주됩니다. 유형을 번들하여 위의 목적용으로 이를 동일한 유형으로 처리할 수 있습니다(예: DECIMAL 및 NUMERIC) 시그니처가 중복되면 오류가 리턴됩니다(SQLSTATE 42723).

parameter-name

입력 매개변수의 선택적 이름을 지정합니다. 이름은 매개변수 목록의 기타 *parameter-name*과 동일할 수 없습니다(SQLSTATE 42734).

data-type1

입력 매개변수의 데이터 유형을 지정합니다. 데이터 유형은 내장 데이터 유형, 구별 유형 또는 구조화된 유형일 수 있습니다.

SOURCE절에서 식별된 함수의 해당 매개변수 유형으로 캐스트할 수 있으면 모든 유효한 SQL 데이터 유형을 사용할 수 있습니다(정보는 『데이터 유형 간 캐스팅』 참조). 그러나 이 검사는 함수가 호출될 때 오류가 발생하지 않음을 보장하지는 않습니다.

각 내장 데이터 유형에 대한 자세한 설명은 『CREATE TABLE』을 참조하십시오.

- 날짜 시간 유형 매개변수는 문자 데이터 유형으로 전달되며 데이터는 ISO 형식으로 전달됩니다.
- 배열 유형을 지정할 수 없습니다(SQLSTATE 42879).
- REF(*type-name*)로 지정된 참조 유형을 지정할 수 없습니다(SQLSTATE 42879).

사용자 정의 구별 유형의 경우, 매개변수의 길이, 정밀도 또는 스케일 속성은 구별 유형의 소스 유형의 속성입니다(CREATE TYPE에 지정된 속성). 구별 유형 매개변수는 구별 유형의 소스 유형으로 전달됩니다. 구별 유형의 이름이 규정되지 않은 경우, 데이터베이스 관리 프로그램은 SQL 경로의 스키마를 검색하여 스키마 이름을 분석합니다.

사용자 정의 구조화된 유형의 경우, 연관된 변환 그룹에 해당 변환 함수가 존재해야 합니다.

함수가 전래되었으므로, 매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다(지정할 수도 있음). 그 대신, 빈 괄호를 사용할 수 있습니다(예: CHAR()). *매개변수화된 데이터 유형*은 특정 길이, 정밀도 또

는 스케일을 사용하여 정의할 수 있는 데이터 유형 중 하나로서, 매개변수화된 데이터 유형은 문자열 데이터 유형, 10진수 데이터 유형 및 TIMESTAMP 데이터 유형입니다.

함수 템플리트의 경우, 매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정하는 대신 빈 괄호를 사용할 수도 있습니다. 매개변수화된 데이터 유형에 대해서는 빈 괄호를 사용할 것을 권장합니다. 빈 괄호를 사용하면 길이, 정밀도 또는 스케일은 함수 맵핑을 작성하여 함수 템플리트를 리모트 함수에 맵핑할 때 판별되는 리모트 함수의 것과 같습니다. 괄호 전체를 생략할 경우, 데이터 유형의 디폴트 길이가 사용됩니다(『CREATE TABLE』 참조).

RETURNS

이 필수 절은 함수 또는 함수 템플리트의 출력을 식별합니다.

data-type2

출력의 데이터 유형을 지정합니다.

전래 스칼라 함수의 경우, 소스 함수의 결과 유형에서 캐스트할 수 있다면 구별 유형에서와 마찬가지로 모든 유효한 SQL 데이터 유형을 사용할 수 있습니다. 배열 유형을 매개변수의 데이터 유형으로 지정할 수 없습니다(SQLSTATE 42879).

위에 설명된 전래 함수의 매개변수에서처럼, 매개변수화된 유형의 매개변수를 지정할 필요가 없습니다. 그 대신, 빈 괄호를 사용할 수 있습니다(예: VARCHAR()).

함수가 다른 함수의 소스가 될 때 RETURNS절의 데이터 유형 스펙에 적용되는 추가 고려사항 및 규칙에 대해서는 이 명령문의 『규칙』 절을 참조하십시오.

함수 템플리트의 경우, 빈 괄호는 허용되지 않습니다(SQLSTATE 42611). 매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정해야 합니다. 리모트 함수와 동일한 길이, 정밀도 또는 스케일을 지정할 것을 권장합니다.

SPECIFIC *specific-name*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *specific-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 18)입니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함한 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *function-name*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *function-name*에 대해 사용된 규정자가 사용됩니다. 규정자가 지정된 경우, 규정자는 *function-name*의 명시적 또는 내재된 규정자와 동일해야 합니다. 그렇지 않으면 오류(SQLSTATE 42882)가 리턴됩니다.

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

SOURCE

전래 함수로 정의되어 있는 새 함수를 지정합니다. *전래 함수*는 다른 함수로 구현됩니다(*전래 함수*). 함수는 현재 서버에 있는 스칼라 또는 집계 함수여야 하며 다음 유형의 함수 중 하나여야 합니다.

- CREATE FUNCTION문으로 정의된 함수
- CREATE TYPE문으로 생성된 캐스트 함수
- 내장 함수

전래 함수가 내장 함수가 아닌 경우 특정 함수는 해당 이름, 함수 시그니처 또는 특정 이름으로 식별할 수 있습니다.

전래 함수가 내장 함수인 경우 SOURCE절은 내장 함수에 대한 함수 시그니처를 포함해야 합니다. 소스 함수는 다음 내장 함수 중 하나여서는 안됩니다(특정 구문이 표시된 경우 표시된 양식은 지정할 수 없습니다).

- CARDINALITY
- CHAR - 2개 이상의 인수가 지정되고 첫 번째 인수가 datetime 데이터 유형인 경우
- CHARACTER_LENGTH
- COALESCE
- CONTAINS
- CURSOR_ROWCOUNT
- DATAPARTITIONNUM
- DBPARTITIONNUM
- DEREF
- EXTRACT
- GRAPHIC - 2개 이상의 인수가 지정되고 첫 번째 인수가 datetime 데이터 유형인 경우
- GREATEST
- HASHEDVALUE
- INSERT - 5개 이상의 인수가 지정된 경우
- INSTR - 5개 이상의 인수가 지정된 경우
- LCASE - 4개 이상의 인수가 지정된 경우
- LEAST
- LEFT - 3개 이상의 인수가 지정된 경우

- LENGTH - 2개 이상의 인수가 지정된 경우
- LOCATE - 4개 이상의 인수가 지정된 경우
- LOCATE_IN_STRING - 5개 이상의 인수가 지정된 경우
- LOWER - 4개 이상의 인수가 지정된 경우
- MAX
- MAX_CARDINALITY
- MIN
- NODENUMBER
- NULLIF
- NVL
- OVERLAY
- PARAMETER
- POSITION
- RAISE_ERROR
- REC2XML
- RID
- RID_BIT
- RIGHT - 3개 이상의 인수가 지정된 경우
- SCORE
- STRIP
- SUBSTRING
- TRIM
- TRIM_ARRAY
- TYPE_ID
- TYPE_NAME
- TYPE_SCHEMA
- UCASE - 4개 이상의 인수가 지정된 경우
- UPPER - 4개 이상의 인수가 지정된 경우
- VALUE
- VARCHAR - 2개 이상의 인수가 지정되고 첫 번째 인수가 datetime 데이터 유형인 경우
- VARGRAPHIC - 2개 이상의 인수가 지정되고 첫 번째 인수가 datetime 데이터 유형인 경우
- XMLATTRIBUTES

- XMLCOMMENT
- XMLCONCAT
- XMLDOCUMENT
- XMLELEMENT
- XMLFOREST
- XMLNAMESPACES
- XMLPARSE
- XMLPI
- XMLQUERY
- XMLROW
- XMLSERIALIZE
- XMLTEXT
- XMLVALIDATE
- XMLXSROBJECTID
- XSLTRANSFORM

function-name

소스로 사용될 특정 함수를 식별하며, 명령문의 권한 부여 ID가 EXECUTE 특권을 갖는 이 *function-name*의 스키마에 정확히 하나의 특정 함수가 있을 경우에만 유효합니다. 이 구문의 변형은 내장 함수인 소스 함수에 대해서는 유효하지 않습니다.

규정되지 않은 이름이 제공될 경우, 현재 SQL 경로(CURRENT PATH 특수 레지스터의 값)가 함수를 찾는 데 사용됩니다. EXECUTE 특권이 있는 명령문의 권한 부여 ID에 대한 해당 이름을 가진 함수가 있는 SQL 경로의 첫 번째 스키마가 선택됩니다.

이름 지정된 스키마에 해당 이름의 함수가 없거나, 해당 이름이 규정화되지 않고 해당 이름을 가진 함수가 SQL 경로에 없을 경우 오류가 리턴됩니다(SQLSTATE 42704). 이름 지정되거나 위치된 스키마에 함수에 대한 권한 부여된 특정 인스턴스가 여러 개 있으면, 오류가 발생합니다(SQLSTATE 42725). 이 이름의 함수가 있는데 명령문의 권한 부여 ID에 이 함수에 대한 EXECUTE 특권이 없으면 오류가 발생합니다(SQLSTATE 42501).

SPECIFIC *specific-name*

함수 작성시 지정되거나 디폴트값인 *specific-name*을 사용하여 소스로 사용할 특수한 사용자 정의 함수를 식별합니다. 이 구문의 변형은 내장 함수인 소스 함수에 대해서는 유효하지 않습니다.

규정되지 않은 이름이 제공될 경우, 현재 SQL 경로가 함수를 찾는 데 사용됩니다. EXECUTE 특권이 있는 명령문의 권한 부여 ID에 대한 해당 특정 이름을 가진 함수가 있는 SQL 경로의 첫 번째 스키마가 선택됩니다.

이름이 지정된 스키마에 이 *specific-name*을 갖는 함수가 없거나, 이름이 규정되지 않았으며 SQL 경로에 이 *specific-name*을 갖는 함수가 없으면, 오류가 발생합니다(SQLSTATE 42704). *specific-name*을 갖는 함수가 있는데 명령문의 권한 부여 ID에 이 함수에 대한 EXECUTE 특권이 없으면 오류가 발생합니다(SQLSTATE 42501).

function-name (data-type,...)

소스 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 이것은 내장 함수인 소스 함수에 대해서만 유효한 구문 변형입니다.

함수 결정 규칙은 SOURCE절에 지정된 데이터 유형이 제공될 경우 함수 이름이 같은 함수 중에서 하나의 함수를 선택할 때 적용됩니다. 그러나 선택된 함수의 각 매개변수에 대한 데이터 유형은 소스 함수에 지정된 해당 데이터 유형과 정확히 같은 유형을 가지고 있어야 합니다.

function-name

소스 함수의 함수 이름을 제공합니다. 규정되지 않은 이름을 제공할 경우, 사용자의 SQL 경로에 있는 스키마가 사용됩니다.

data-type

CREATE FUNCTION문의 해당 위치에 쉼표로 구분하여 지정한 데이터 유형과 일치해야 합니다.

매개변수화된 데이터 유형에 대해 정밀도 또는 스케일, 길이를 지정할 필요가 없습니다. 대신 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다. 예를 들어, DECIMAL()은 데이터 유형이 DECIMAL(7,2)로 정의된 매개변수와 일치해야 합니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다. 이것은 의도한 함수가 사용되도록 하려는 경우에 유용할 수 있습니다. 데이터 유형에 대한 동의어도 일치하는 것으로 간주됩니다(예를 들어, DEC와 NUMERIC는 일치함).

0<n<25는 REAL을 의미하고, 24<n<54는 DOUBLE을 의미하므로, FLOAT(n) 유형은 n에 대해 지정된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 가진 함수가 이름이 지정된 스키마나 내재적 스키마에 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

PARAMETER CCSID

함수에 전달되거나 함수에서 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII

문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031). 함수가 호출될 때 함수의 응용프로그램 코드 페이지는 데이터베이스 코드 페이지입니다.

UNICODE

문자열 데이터를 유니코드로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, 문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있습니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, 문자 데이터는 UTF-8로 되어 있습니다. 어떠한 경우라도 함수가 호출될 때 함수의 응용프로그램 코드 페이지는 1208입니다.

PARAMETER CCSID절은 소스 함수와 동일한 코드화 체계를 지정해야 합니다(SQLSTATE 53090).

AS TEMPLATE

이 명령문이 실행 가능 코드를 가진 함수가 아니라, 함수 템플리트를 작성하는 데 사용될 것임을 나타냅니다.

NOT DETERMINISTIC 또는 DETERMINISTIC

함수가 동일한 입력 인수에 대해 동일한 결과를 리턴하는지 여부를 지정합니다. 디폴트값은 NOT DETERMINISTIC입니다.

NOT DETERMINISTIC

동일한 입력 인수로 함수를 호출할 때마다 함수가 동일한 결과를 리턴하지 않을 수 있다는 것을 지정합니다. 함수는 결과에 영향을 주는 일부 상태 값에 의존합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다. 결정적이 아닌 함수의 예는 난수를 생성하는 함수입니다.

결정적이 아닌 함수는 병렬 태스크에 의해 실행될 경우 틀린 결과를 받을 수 있습니다.

DETERMINISTIC

동일한 입력 인수로 함수를 호출할 때마다 함수가 언제나 동일한 결과를 리

턴한다는 것을 지정합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다. 결정적인 함수의 예는 입력 인수의 제곱근을 계산하는 함수입니다.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하는지의 여부를 지정합니다. 외부 조치의 예로는 메시지 전송 또는 파일에 레코드 쓰기를 들 수 있습니다. 디폴트값은 EXTERNAL ACTION입니다.

EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하도록 지정합니다. SQL 루틴 본문이 EXTERNAL ACTION으로 정의된 함수를 호출할 경우 EXTERNAL ACTION을 명시적이나 내재적으로 지정해야 합니다(SQLSTATE 428C2).

외부 조치가 있는 함수는 병렬 태스크가 해당 함수를 실행할 경우 부정확한 결과를 리턴할 수 있습니다. 예를 들어 함수가 자신에 대한 각 초기 호출에 대해 메모를 전송할 경우, 함수에 대해 한 번의 메모를 전송하는 대신에 각 병렬 태스크에 대해 하나의 메모를 전송합니다.

NO EXTERNAL ACTION

함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하지 않는다는 것을 지정합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다.

규칙

- 편의상, 이 절에서는 허용되는 세 가지 구문 중 어떤 것이 SF를 식별하는 데 사용되었는지에 관계없이 CF를 작성하는 함수와 SOURCE절 SF에 식별된 함수를 호출합니다.
  - CF의 규정되지 않은 이름과 SF의 규정되지 않은 이름은 다를 수 있습니다.
  - 다른 함수의 소스로 이름 지정된 함수는 다른 함수를 소스로 사용할 수 있습니다. 간접적으로 호출된 함수에 오류가 발생하면 응용프로그램을 디버그하기가 매우 어려울 수 있으므로, 이 기능을 이용할 때 주의해야 합니다.
  - 다음 절은 SOURCE절과 함께 지정할 경우 유효하지 않습니다. 왜냐하면 CF가 SF로부터 속성들을 상속하기 때문입니다.
    - CAST FROM ...,
    - EXTERNAL ...,
    - LANGUAGE ...,
    - PARAMETER STYLE ...,
    - DETERMINISTIC / NOT DETERMINISTIC,

- FENCED / NOT FENCED,
- RETURNS NULL ON NULL INPUT / CALLED ON NULL INPUT
- EXTERNAL ACTION / NO EXTERNAL ACTION
- NO SQL / CONTAINS SQL / READS SQL DATA
- SCRATCHPAD / NO SCRATCHPAD
- FINAL CALL / NO FINAL CALL
- RETURNS TABLE (...)
- CARDINALITY ...
- ALLOW PARALLEL / DISALLOW PARALLEL
- DBINFO / NO DBINFO
- THREADSAFE / NOT THREADSAFE
- INHERIT SPECIAL REGISTERS

이 규칙을 위반하면 오류가 발생합니다(SQLSTATE 42613).

- CF의 입력 매개변수 수는 SF에서의 입력 매개변수 수와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42624).
- 다음의 경우에 CF에서 매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정하지 않아도 됩니다.
  - 함수의 입력 매개변수
  - RETURNS 매개변수

  그 대신, 길이/정밀도/스케일이 소스 함수와 같음을 나타내기 위해 데이터 유형의 일부로 빈 괄호를 지정하거나(예: VARCHAR()) 캐스팅에 의해 빈 괄호가 판별될 수 있습니다.

  그러나 길이, 정밀도 또는 스케일이 지정되면, 입력 매개변수와 리턴 값에 대해 아래에 간략히 설명된 대로 SF의 해당 값에 대해 CF의 값이 검사됩니다.
- CF 입력 매개변수의 스펙은 SF 입력 매개변수의 스펙에 대해 점검됩니다. CF의 각 매개변수의 데이터 유형은 SF의 해당 매개변수에 대한 데이터 유형과 같거나 *캐스트 가능*해야 합니다. 매개변수가 같은 유형이 아니거나 캐스트 가능하지 않으면, 오류가 발생합니다(SQLSTATE 42879).

  이 규칙은 CF 사용시 발생하는 오류에 대해서는 보장되지 않는다는 점에 유의하십시오. 데이터 유형 및 CF 매개변수의 길이 또는 정밀도와 일치하는 인수는 해당 SF 매개변수의 길이가 더 짧고, 그 정밀도가 더 낮을 경우에는 지정할 수 없는 경우도 있습니다. 일반적으로 CF 매개변수에는 해당 SF 매개변수의 속성보다 더 큰 길이 또는 정밀도 속성이 있으면 안됩니다.

- CF의 RETURNS 데이터 유형에 대한 스펙은 SF의 이 스펙에 대해 점검됩니다. SF의 마지막 RETURNS 데이터 유형은 캐스팅 이후에 CF의 RETURNS 데이터 유형과 같거나 캐스트 가능해야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42866).

  이 규칙은 CF 사용시 발생하는 오류에 대해서는 보장되지 않는다는 점에 유의하십시오. 데이터 유형 및 SF RETURNS 데이터 유형의 길이 또는 정밀도 속성과 일치하는 결과 값은 CF RETURNS 데이터 유형의 길이가 더 짧거나 정밀도가 더 낮은 경우 지정할 수 없는 수도 있습니다. SF RETURNS 데이터 유형의 속성보다 적은 길이 또는 정밀도 속성을 가지므로, CF의 RETURNS 데이터 유형 지정 선택 시 주의해야 합니다.

주

- 하나의 데이터 유형이 다른 데이터 유형으로 캐스트 가능한지를 판별할 때 CHAR 및 DECIMAL과 같은 매개변수화된 데이터 유형에 대해 길이나 정밀도, 스케일은 고려되지 않습니다. 그러므로 함수를 사용할 때 소스 데이터 유형의 값을 목표 데이터 유형의 값으로 캐스트하려고 하면 오류가 발생할 수 있습니다. 예를 들어, VARCHAR은 DATE로 캐스트 가능하나 소스 유형이 실제로 VARCHAR(5)로 정의되면, 함수를 사용할 때 오류가 발생합니다.
- 사용자 정의 함수(UDF)의 매개변수에 대한 데이터 유형을 선택할 때, 입력 값에 영향을 줄 승격 규칙을 고려하십시오(『데이터 유형 승격』 참조). 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀 더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음과 같은 데이터 유형을 사용하는 것이 좋습니다.
  - SMALLINT 대신 INTEGER
  - REAL 대신 DOUBLE
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성할 경우, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가지고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 부여됩니다.
- 페더레이티드 서버가 데이터 소스 함수를 인식하기 위해서는 함수가 페더레이티드 데이터베이스에 있는 상대편에 맵핑되어야 합니다. 데이터베이스에 상대편이 없으면, 사용자가 상대편을 작성한 후 맵핑을 작성해야 합니다.

상대편은 함수(스칼라나 소스) 또는 함수 템플리트일 수 있습니다. 사용자가 함수 및 필요한 맵핑을 작성하면 함수를 지정하는 쿼리가 처리될 때마다 DB2가 (1) 호출하는 전략을 데이터 소스 함수를 호출하는 전략과 비교하고 (2) 오버헤드가 덜 들어가는 것으로 예상되는 함수를 호출합니다.

사용자가 함수 템플리트 및 맵핑을 작성하는 경우 그 템플리트를 지정하는 쿼리가 처리될 때마다, 해당 함수를 호출하기 위해 액세스 플랜이 존재하면 DB2는 맵핑되는 데이터 소스 함수를 호출합니다.

- *특권*: 함수 정의자는 함수 삭제 권한은 물론 함수에 대한 EXECUTE 특권도 항상 수신합니다. 다음 중 하나에 해당하면 함수 정의자에게 WITH GRANT OPTION 도 함께 부여됩니다.
  - 소스 함수가 내장 함수입니다.
  - 함수 정의자가 소스 함수에 대해 EXECUTE WITH GRANT OPTION을 가집니다.
  - 함수가 템플리트입니다.

예:

예 1: Pellow가 원래 CENTRE 외부 스칼라 함수를 작성한 이후에, 다른 사용자가 그 함수를 작성하려고 합니다. 단, 이 함수는 정수 인수만 허용합니다.

```
CREATE FUNCTION MYCENTRE (INTEGER, INTEGER)
  RETURNS FLOAT
       SOURCE PELLOW.CENTRE (INTEGER, FLOAT)
```

예 2: 구별 유형 HATSIZE가 내장 INTEGER 데이터 유형을 기반으로 작성되었습니다. 서로 다른 부서의 평균 모자 *크기*를 계산하려면 AVG 함수가 있는 것이 유용합니다. 다음과 같이 간편하게 수행할 수 있습니다.

```
CREATE FUNCTION AVG (HATSIZE) RETURNS HATSIZE
  SOURCE SYSIBM.AVG (INTEGER)
```

구별 유형을 작성하면 필수 캐스트 함수가 작성되며 인수에 대해 HATSIZE에서 INTEGER로 그리고 INTEGER에서 HATSIZE로의 캐스트가 허용됩니다.

예 3: 페더레이티드 시스템에서, 사용자가 배정밀도 부동 소수점의 형태로 값을 형성하는 테이블 통계를 리턴하는 Oracle UDF를 호출하려고 합니다. 페더레이티드 서버는 함수와 페더레이티드 데이터베이스 상대편간의 맵핑이 있는 경우에만 이 함수를 인식할 수 있습니다. 그러나 그러한 상대편이 존재하지 않습니다. 사용자는 함수 템플리트의 형태로 하나 제공하고 NOVA라고 하는 스키마에 이 템플리트를 지정하기로 결정합니다. 사용자는 다음 코드를 사용하여 템플리트를 페더레이티드 서버에 등록합니다.

```
CREATE FUNCTION NOVA.STATS (DOUBLE, DOUBLE)
  RETURNS DOUBLE
  AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

예 4: 페더레이티드 시스템에서 사용자가 특정 조직의 직원이 보너스로서 버는 달러 금액을 리턴하는 Oracle UDF를 호출하려고 합니다. 페더레이티드 서버는 함수와 페더레이티드 데이터베이스 상대편간의 맵핑이 있는 경우에만 이 함수를 인식할 수 있습니다. 그러한 상대편이 존재하지 않습니다. 함수 템플리트의 형태로 하나를 작성하려는 경우 다음 코드를 사용하여 이 템플리트를 페더레이티드 서버에 등록하십시오.

```
CREATE FUNCTION BONUS ()
  RETURNS DECIMAL (8,2)
  AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

CREATE FUNCTION(SQL 스칼라, 테이블 또는 행)

CREATE FUNCTION(SQL 스칼라, 테이블 또는 행)문을 사용하여 사용자 정의 SQL 스칼라, 테이블 또는 행 함수를 정의합니다. *스칼라 함수*는 호출될 때마다 단일 값을 리턴하며, 일반적으로 SQL 표현식이 유효하면 이 함수도 유효합니다. *테이블 함수*는 FROM절에 사용할 수 있으며 테이블을 리턴합니다. *행 함수*는 전송 함수로 사용할 수 있으며 행을 리턴합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

또한 fullselect에서 식별되는 각 테이블, 뷰 또는 별칭에 대해 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- 테이블, 뷰 또는 별칭에 대한 SELECT 특권
- DATAACCESS 권한

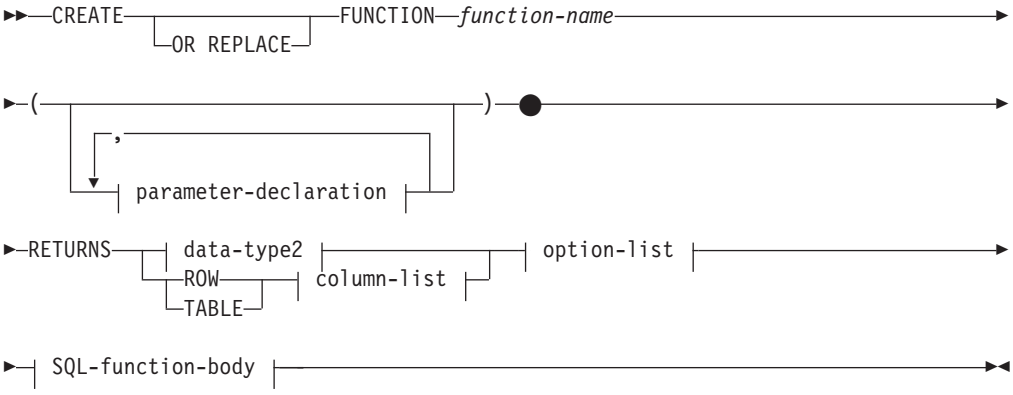
CREATE FUNCTION문에 지정된 테이블이나 뷰에 대해서는 PUBLIC 이외의 그룹 특권이 고려되지 않습니다.

별칭으로 참조되는 테이블이나 뷰에 대한 데이터 소스의 권한 부여 요건은 함수가 호출될 때 적용됩니다. 연결의 권한 부여 ID는 서로 다른 리모트 권한 부여 ID로 맵핑될 수 있습니다.

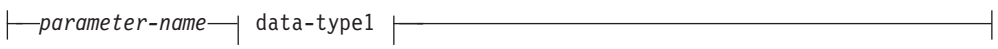
명령문의 권한 부여 ID에서 보유한 특권은 함수 본문에서 지정된 SQL문을 호출할 때 필요한 특권 모두도 포함해야 합니다.

기존 함수를 교체하려면 명령문의 권한 부여 ID가 기존 함수의 소유자여야 합니다 (SQLSTATE 42501).

구문

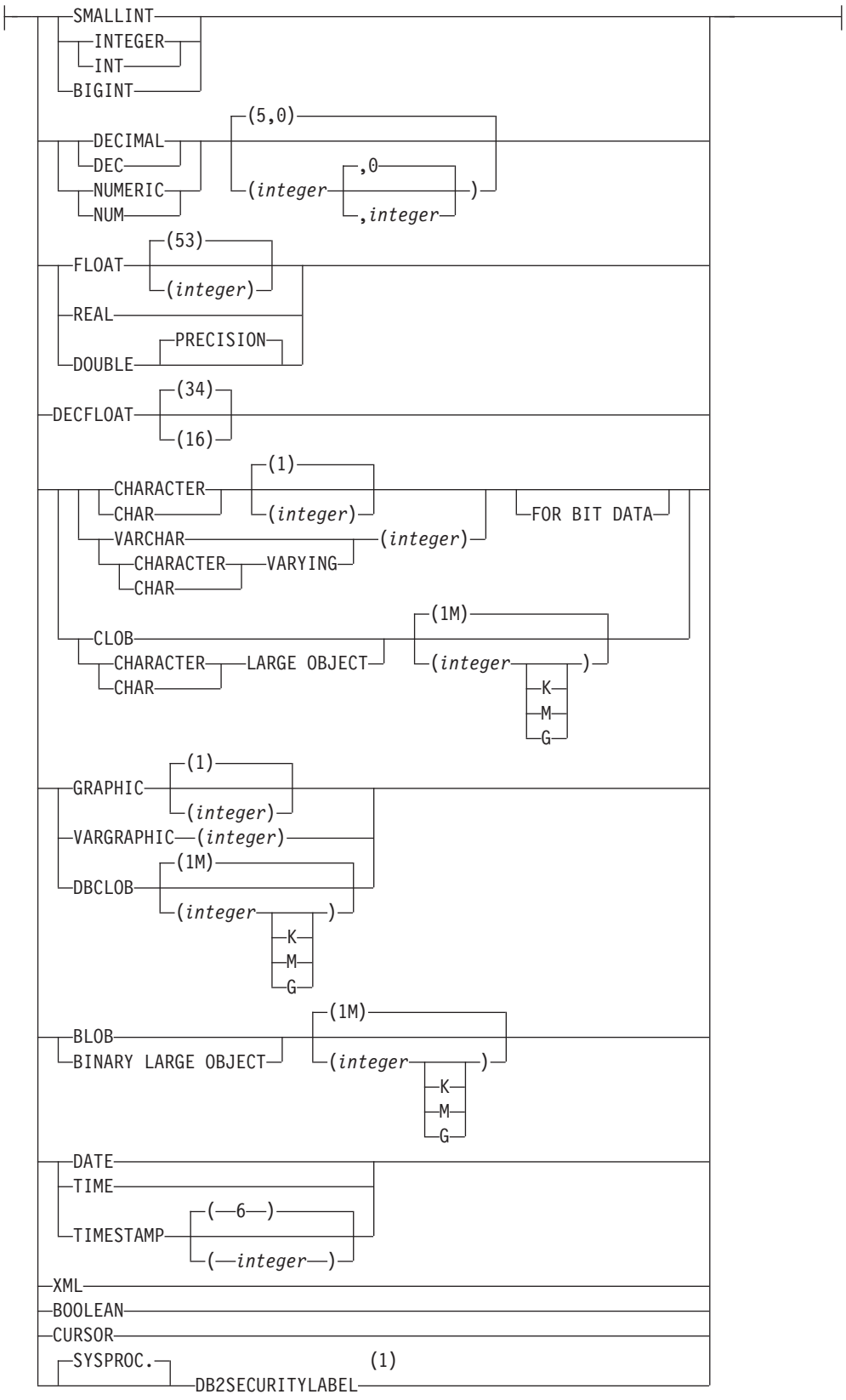


parameter-declaration:

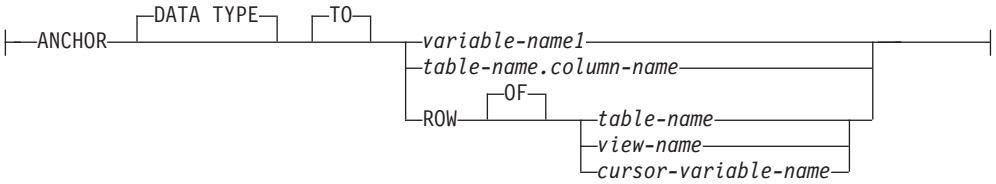


data-type1, data-type2:

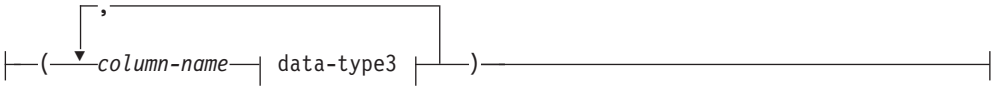
내장 유형:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
,integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
FOR BIT DATA
VARCHAR
(integer)
CHARACTER
VARYING
CHAR
CLOB
LARGE OBJECT
(1M)
K
M
G
GRAPHIC
VARGRAPHIC
DBCLOB
BLOB
BINARY LARGE OBJECT
DATE
TIME
TIMESTAMP
(6)
XML
BOOLEAN
CURSOR
SYSPROC.
DB2SECURITYLABEL
(1)

anchored-data-type:

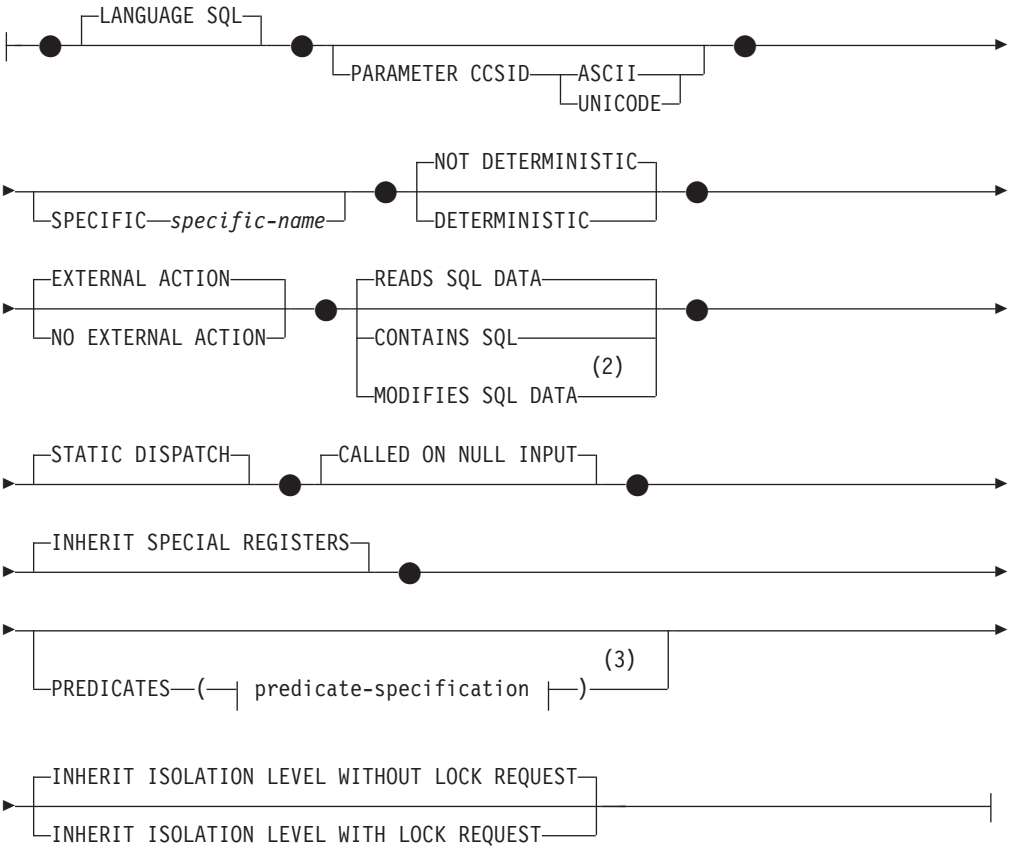


column-list:

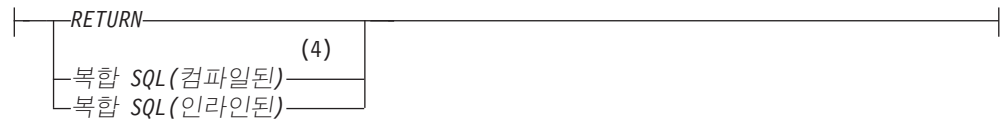


data-type3:

option-list:



SQL-function-body:



주:

1 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

2 RETURNS가 테이블(TABLE *column-list*)을 지정한 경우에 유효합니다. RETURNS가 스칼라 결과를 리턴하고 SQL-function-body가 복합 SQL(컴파일된) 문인 경우에도 유효합니다. 이 경우, 결과 함수는 복합 SQL(컴파일된)문에 있는 지정 명령문의 오른쪽에 있는 요소만으로 사용할 수 있습니다.

3 RETURNS가 스칼라 결과(*data-type2*)를 지정할 경우에만 유효합니다.

4 복합 SQL(컴파일된)문은 파티션되지 않은 데이터베이스에서, SQL 스칼라 함수 정의의 SQL-function-body에 대해서만 지원됩니다. SQL 테이블 함수 정의의 경우 또는 파티션된 데이터베이스 환경에서는 지원되지 않습니다.

설명

OR REPLACE

현재 서버에 정의가 존재하면 함수의 정의를 교체하도록 지정합니다. 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 삭제되며, 함수에 부여되었던 특권에는 영향을 주지 않습니다. 함수에 대한 정의가 현재 서버에 존재하지 않으면 이 옵션이 무시됩니다.기존 함수를 교체하려면, 새 정의의 함수 이름과 특정 이름이 이전 정의의 함수 이름 및 특정 이름과 같거나, 새 정의의 시그니처가 이전 정의의 시그니처와 일치해야 합니다. 그렇지 않으면 새 함수가 작성됩니다.

function-name

정의될 함수의 이름을 지정합니다. 이 이름은 함수를 지시하는 규정화되거나 규정되지 않는 이름입니다. *function-name*의 규정되지 않은 형식은 SQL ID(최대 길이 : 18)입니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다.

매개변수 수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고) 내재적 또는 명시적 규정자를 포함한 이름은 카탈로그에 기술된 함수를 식별해서는 안됩니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 구성된 이름을 지정할 경우, *schema-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *function-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

함수 시그니처에 약간의 차이가 있으면, 여러 함수에 동일한 이름을 사용할 수 있습니다. 이에 대한 제한이 없다고 하여도, 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안됩니다.

(*parameter-declaration***,...)**

함수의 입력 매개변수 수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수가 받게 될 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 매개변수 수는 90개를 초과할 수 없습니다(SQLSTATE 54023).

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예를 들면, 다음과 같습니다.

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 이름이 지정된 두 함수는 모든 해당 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 그러므로, CHAR(8) 및 CHAR(35)은 DECFLOAT(16) 및 DECFLOAT(34) 뿐 아니라 DECIMAL(11,2) 및 DECIMAL(4,3)과 같이, 동일한 유형으로 처리됩니다. 유니코드 데이터베이스의 경우, CHAR(13) 및 GRAPHIC(8)은 같은 유형으로 간주됩니다. DECIMAL 및 NUMERIC과 같이 이러한 목적을 위해 동일한 유형으로 처리되도록 하는 유형들이 더 있습니다. 시그니처가 중복되면 오류가 리턴됩니다(SQLSTATE 42723).

매개변수에 대한 데이터 유형이 부울 데이터 유형, 배열 유형, 커서 유형 또는 행 유형인 경우, SQL 함수 본문은 복합 SQL(컴파일된) 명령문 내에서만 매개변수를 참조할 수 있습니다(SQLSTATE 428H2). 매개변수에 대한 데이터 유형이 XML인 경우, 매개변수는 복합 SQL(컴파일된) 명령문인 SQL 함수 본문에서 참조될 수 없습니다(SQLSTATE 429BB).

parameter-name

입력 매개변수에 대한 이름을 지정합니다. 이름은 매개변수 목록의 다른 *parameter-name*과 같을 수 없습니다(SQLSTATE 42734).

data-type1

입력 매개변수의 데이터 유형을 지정합니다.

built-in-type

내장 데이터 유형을 지정합니다. 테이블에 대해 지정할 수 없는 각 내장 데

이터 유형(BOOLEAN 및 CURSOR 제외)에 대한 자세한 설명은 "CREATE TABLE"을 참조하십시오.

BOOLEAN
부울

CURSOR
기반 커서에 대한 참조.

anchored-data-type
매개변수 데이터 유형을 정의하는 데 사용되는 다른 오브젝트를 식별합니다. 앵커 오브젝트의 데이터 유형은 *data-type1*로 명시적으로 허용된 모든 데이터 유형이 가능합니다. 앵커 오브젝트의 데이터 유형에는 직접 데이터 유형을 지정하거나 행의 경우 행 유형 작성에 적용되는 동일한 제한사항이 있습니다.

ANCHOR DATA TYPE TO
데이터 유형을 지정하는 데 사용되는 앵커된 데이터 유형을 표시합니다.

variable-name1
전역 변수를 식별합니다. 전역 변수의 데이터 유형은 *parameter-name*의 데이터 유형으로 사용됩니다.

table-name.column-name
기존 테이블 또는 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 *parameter-name*의 데이터 유형으로 사용됩니다.

ROW OF *table-name* **또는** *view-name*
*table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뷰의 컬럼 이름 및 컬럼 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. *parameter-name*의 데이터 유형은 unnamed row 자료형입니다.

ROW OF *cursor-variable-name*
*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 지정된 커서 변수는 다음 중 하나여야 합니다 (SQLSTATE 428HS).

- 강하게 유형 지정된 커서 데이터 유형이 있는 전역 변수
- 모든 결과 컬럼이 이름 지정된 *select-statement*를 지정하는 CONSTANT절로 작성되거나 선언되어 약하게 유형 지정된 커서 데이터 유형이 있는 전역 변수

커서 변수의 커서 유형이 named row 자료형을 사용하여 강하게 유형 지정되지 않은 경우, *parameter-name*의 데이터 유형은 unnamed row 자료형입니다.

array-type-name
사용자 정의 배열 유형의 이름을 지정합니다. 스키마 이름 없이 *array-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 배열 유형이 분석됩니다.

cursor-type-name
커서 유형의 이름을 지정합니다. 스키마 이름 없이 *cursor-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 커서 유형이 분석됩니다.

distinct-type-name
구별 유형의 이름을 지정합니다. 매개변수의 길이, 정밀도 및 스케일은 각각 구별 유형의 소스 유형에 대한 길이, 정밀도 및 스케일입니다. 구별 유형 매개변수는 구별 유형의 소스 유형으로 전달됩니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

REF (*type-name***)**
범위가 없는 참조 유형을 지정합니다. 지정된 *type-name*은 사용자 정의한 구조화된 유형을 식별해야 합니다(SQLSTATE 428DP). 시스템은 매개변수나 결과의 범위를 판단하려고 하지 않습니다. 함수 내용에서, 참조 유형은 범위를 수반하도록 처음으로 캐스팅을 수행할 경우에만 비참조 조작에서 사용할 수 있습니다. 마찬가지로 SQL 함수에서 리턴되는 참조는 범위를 수반하도록 처음으로 캐스팅을 수행할 경우에만 비참조 조작에서 사용할 수 있습니다. 유형 이름이 스키마 이름 없이 지정된 경우, SQL 경로에서 스키마를 검색하여 *type-name*이 분석됩니다.

row-type-name
사용자 정의 행 유형의 이름을 지정합니다. 매개변수의 필드는 행 유형의 필드입니다. 스키마 이름 없이 *row-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 행 유형이 분석됩니다.

structured-type-name
사용자 정의한 구조화된 유형 이름을 지정합니다. 스키마 이름을 포함하지 않은 채로 *structured-type-name*을 지정한 경우 구조화된 유형은 SQL 경로에서 스키마를 검색하여 해결됩니다.

RETURNS
이 필수 절은 함수의 출력 유형을 식별하십시오.

함수 출력의 데이터 유형이 부울 데이터 유형, 배열 유형, 커서 유형 또는 행 유형인 경우, SQL 함수 본문은 복합 SQL(컴파일된) 명령문이어야 합니다(SQLSTATE

428H2). 함수 출력의 데이터 유형이 XML인 경우, SQL 함수 본문은 복합 SQL(컴파일된) 명령문이어야 합니다(SQLSTATE 429BB).

data-type2
출력의 데이터 유형을 지정합니다.

이 명령문에서, 함수 매개변수에 대해 *data-type1*에서 설명한 SQL 함수의 매개변수에 대해 적용한 것과 같은 고려사항이 적용됩니다.

TABLE *column-list*
이 함수의 결과는 테이블입니다.

column-list
ROW 또는 TABLE 함수에 대해 리턴되는 컬럼 이름 및 데이터 유형 목록

column-name
이 컬럼의 이름을 지정하십시오. 이름을 규정화할 수 없으며 행의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

data-type3
컬럼의 데이터 유형을 지정하며, SQL 함수의 매개변수에서 지원되는 데이터 유형이면 됩니다.

함수 매개변수에 대해 *data-type1*에서 설명한 SQL 함수의 매개변수에 대해 적용한 것과 같은 고려사항이 적용됩니다. 그러나 *data-type3*은 다음을 지원하지 않습니다(*anchored-data-type*, *array-type-name*, *cursor-type-name* 및 *row-type-name*).

SPECIFIC *specific-name*
정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *specific-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 18)입니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함하여, 이 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별하지 않아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *function-name*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *function-name*에 대해 사용된 규정자가 사용됩니다. 규정자가 지정되면, 함수 이름의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

LANGUAGE SQL
함수가 SQL을 사용하여 작성됨을 지정합니다.

PARAMETER CCSID
함수에 전달되거나 함수에서 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII
문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031).

UNICODE
문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있도록 지정합니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE를 지정할 수 없습니다(SQLSTATE 56031).

DETERMINISTIC 또는 NOT DETERMINISTIC
이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION
함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하는지의 여부를 지정합니다. 외부 조치의 예로는 메시지 전송 또는 파일에 레코드 쓰기를 들 수 있습니다. 디폴트값은 EXTERNAL ACTION입니다.

EXTERNAL ACTION
함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하도록 지정합니다.

NO EXTERNAL ACTION
함수가 데이터베이스 관리 프로그램이 관리하지 않는 오브젝트의 상태를 변경하는 조치를 수행하지 않는다는 것을 지정합니다. 데이터베이스 관리 프로그램은 SQL문 최적화 시 이 정보를 사용합니다.

CONTAINS SQL, READS SQL DATA 또는 MODIFIES SQL DATA
실행할 수 있는 SQL문의 유형을 나타냅니다.

CONTAINS SQL
SQL 데이터를 읽지도 수정하지도 않는 SQL문이 함수에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 42985).

READS SQL DATA
SQL 데이터를 수정하지 않는 SQL문이 함수에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 42985).

MODIFIES SQL DATA
SQL-function-body에서 지원되는 모든 SQL문은 함수로 실행할 수 있음을 나타냅니다.

STATIC DISPATCH
이 선택적 절은 함수 결정시 DB2가 함수 매개변수의 정적 유형(선언된 유형)에 기반하는 함수를 선택한다는 것을 나타냅니다.

CALLED ON NULL INPUT
이 절은 해당되는 인수 중 어느 하나가 널(NULL)인지에 관계없이 함수가 호출됨을 나타냅니다. 이 절은 널(NULL) 값이나 널(NULL)이외의 값을 리턴할 수 있습니다. 널(NULL) 인수 값에 대한 테스트 책임은 사용자 정의 함수에 있습니다.

CALLED ON NULL INPUT 대신 NULL CALL을 사용할 수 있습니다.

INHERIT SPECIAL REGISTERS
이 선택적 절은 함수에 있는 갱신 가능한 특수 레지스터가 레지스터의 초기값을 호출 명령문의 환경에서 상속한다는 것을 나타냅니다. 커서의 select문에서 호출되는 함수의 경우에는 초기값을 커서가 열려 있는 환경에서 상속합니다. 중첩 오브젝트(예: 트리거 또는 뷰)에서 호출되는 루틴의 경우에는 초기값을 오브젝트 정의에서 상속하지 않고 런타임 환경에서 상속합니다.

특수 레지스터에 대한 변경사항은 함수 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 호출자로부터 상속하지 않습니다.

PREDICATES
이 함수를 사용하는 술어의 경우, 이 절은 인덱스 확장을 이용할 수 있는 술어들을 식별하고, 술어의 검색 조건에 대해 선택적 SELECTIVITY절을 사용할 수 있습니다. PREDICATES절을 지정할 경우, 함수는 NO EXTERNAL ACTION을 사용하여 DETERMINISTIC으로 정의되어야 합니다(SQLSTATE 42613). PREDICATES절을 지정했는데 데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE를 지정하면 안됩니다(SQLSTATE 42613). PREDICATES는 SQL-function-body가 복합 SQL(컴파일된)문인 경우에는 지정할 수 없습니다(SQLSTATE 42613).

predicate-specification

술어 스펙에 대한 자세한 내용은 『CREATE FUNCTION(외부 스칼라)』을 참조하십시오.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST 또는 INHERIT ISOLATION LEVEL WITH LOCK REQUEST

함수가 함수를 호출하는 명령문의 분리 레벨을 상속할 경우, 잠금 요청이 명령문의 분리 절과 연관될 수 있는지 여부를 지정합니다. 디폴트값은 INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST입니다.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST

함수가 호출하는 명령문의 분리 레벨을 상속하므로 잠금 요청 절을 지정된 분리 절의 일부로 포함하는 SQL문의 컨텍스트에서 함수를 호출할 수 없음을 지정합니다(SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST

함수가 호출하는 명령문의 분리 레벨을 상속하므로 지정된 잠금 요청 절도 상속함을 지정합니다.

SQL-function-body

함수의 본문을 지정합니다. 매개변수 이름은 SQL-function-body에서 참조될 수 있습니다. 매개변수 이름은 불명확한 참조를 피하기 위해 함수 이름으로 규정될 수 있습니다.

RETURN문의 경우 RETURN문을 참조하십시오.

*복합 SQL(컴파일된)*의 경우, 복합 SQL(컴파일된) 명령문을 참조하십시오.

*복합 SQL(인라인된)*의 경우, 복합 SQL(인라인된) 명령문을 참조하십시오.

규칙

- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.
- ***커서 및 행 유형 사용:*** 매개변수에 대한 커서 유형 또는 행 유형을 사용하거나 커서 유형 또는 행 유형을 리턴하는 함수는 복합 SQL(컴파일된)문 내에서만 호출할 수 있습니다(SQLSTATE 428H2).
- ***테이블 액세스 제한사항***: 함수가 READS SQL DATA로 정의되어 있으면 함수의 명령문은 함수를 호출한 명령문이 수정하고 있는 테이블에 액세스할 수 없습니다(SQLSTATE 57053). 예를 들어, 사용자 정의 함수(UDF) BONUS()가 READS SQL DATA로 정의되어 있다고 가정해 보십시오. UPDATE EMPLOYEE SET SALARY

= SALARY + BONUS(EMPNO) 명령문을 호출하면 BONUS 함수에 있는 어떤 SQL문도 EMPLOYEE 테이블에서 데이터를 읽을 수 없습니다.

MODIFIES SQL DATA로 정의된 함수에 중첩된 CALL문이 포함될 경우, 함수(함수 정의 또는 함수를 호출하는 명령문)에 의해 수정되고 있는 테이블에 대한 읽기 액세스는 허용되지 않습니다(SQLSTATE 57053).

주

- 함수 본문 내의 함수 호출 분석은 CREATE FUNCTION문에 대해 효율적인 SQL 경로에 따라 수행되고 그 함수가 작성되고 나면 변경되지 않습니다.
- SQL 함수에 날짜 및 시간 특수 레지스터 중 하나에 대한 여러 참조가 포함될 경우, 모든 참조는 같은 값을 리턴하며, 이것은 그 함수를 호출한 명령문에서의 레지스터 호출에 의해 리턴된 것과 같은 값이어야 합니다.
- SQL 함수의 본문은 그 자체나 이를 호출하는 다른 함수 또는 메소드에 대한 재귀 호출을 포함할 수 없습니다. 그런 함수는 호출되기 위해 존재할 수 없기 때문입니다.
- SQL 함수 본문에서 참조된 오브젝트가 존재하지 않거나 유효하지 않은 것으로 표시되거나 정의자에 임시로 오브젝트 액세스 특권이 없고 데이터베이스 구성 매개변수 **auto_reval**이 DISABLED로 설정되어 있지 않으면, SQL 함수는 성공적으로 여전히 작성됩니다. SQL 함수는 유효하지 않은 것으로 표시되며 다음에 호출될 때 유효성이 다시 확인됩니다.
- 다음 규칙은 함수나 메소드를 작성하는 모든 명령문에 의해 시행됩니다.
  - 함수는 메소드와 같은 시그니처를 가질 수 없습니다(함수의 첫 번째 *parameter-type*을 메소드의 *subject-type*과 비교할 경우).
  - 함수와 메소드는 중첩 관계가 될 수 없습니다. 즉, 함수가, 첫 번째 매개변수가 주제인 메소드일 경우, 다른 메소드를 겹쳐쓰기하거나 다른 메소드에 의해 겹쳐쓰기되어서는 안됩니다. 메소드 겹쳐쓰기에 대한 자세한 내용은 『CREATE TYPE(구조화)』을 참조하십시오.
  - 겹쳐쓰기는 함수에 적용되지 않으므로, 두 함수가 메소드여서 하나가 다른 메소드를 겹쳐쓰기할 경우와 같이 두 함수가 존재할 수 있습니다.

  위의 규칙에서의 매개변수 유형 비교를 위해 다음이 수행됩니다.
  - 매개변수 이름, 길이, AS LOCATOR 및 FOR BIT DATA가 무시됩니다.
  - 부속 유형은 해당되는 슈퍼 유형과 다른 것으로 간주됩니다.
- *특권*: 함수 정의자는 항상 함수 삭제 권한 뿐만 아니라, 함수에 대한 EXECUTE 특권도 부여받습니다.함수 정의자에게 함수 정의에 필요한 모든 특권으로 WITH GRANT OPTION이 있거나 정의자에게 SYSADM이나 DBADM 권한이 있으면 함수 정의자는 함수에 대한 WITH GRANT OPTION도 함께 부여됩니다.

함수 정의자는 함수 작성시 파생되는 특권들이 존재할 경우, 특권을 획득할 수 있습니다. 정의자는 PUBLIC이 특권을 갖기 때문에 또는 직접 특권을 가져야 합니다. 함수 정의자가 구성원인 그룹에서 보유하는 특권은 고려되지 않습니다. 함수를 사용할 때는 연결된 사용자의 권한 부여 ID에 별칭이 데이터 소스에서 참조하는 테이블이나 뷰에 대한 유효한 특권이 있어야 합니다.

- *호환성*: z/OS용 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - 유니코드 데이터베이스의 경우 CCSID UNICODE
    - 비유니코드 데이터베이스의 경우 CCSID ASCII

  이전 버전 DB2와의 호환성을 위해,
  - CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.

예:

예 1: 기존의 사인 및 코사인 함수를 사용하여 값의 탄젠트를 리턴하는 스칼라 함수를 정의하십시오.

```
CREATE FUNCTION TAN (X DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
       NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

예 2: 구조화된 유형 PERSON에 대한 변환 함수를 정의하십시오.

```
CREATE FUNCTION FROMPERSON (P PERSON)
  RETURNS ROW (NAME VARCHAR(10), FIRSTNAME VARCHAR(10))
  LANGUAGE SQL
  CONTAINS SQL
       NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN VALUES (P..NAME, P..FIRSTNAME)
```

예 3: 지정된 부서 번호의 사원을 리턴하는 테이블 함수를 정의하십시오.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                                     LASTNAME VARCHAR(15),
                                     FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
       NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN
    SELECT EMPNO, LASTNAME, FIRSTNME
      FROM EMPLOYEE
      WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

예 4: 문자열을 역방향으로 바꾸는 스칼라 함수를 정의하십시오.

```
CREATE FUNCTION REVERSE(INSTR VARCHAR(4000))
  RETURNS VARCHAR(4000)
  DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
  BEGIN ATOMIC
  DECLARE REVSTR, RESTSTR VARCHAR(4000) DEFAULT '';
  DECLARE LEN INT;
  IF INSTR IS NULL THEN
  RETURN NULL;
  END IF;
  SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
  WHILE LEN > 0 DO
  SET (REVSTR, RESTSTR, LEN)
           = (SUBSTR(RESTSTR, 1, 1) CONCAT REVSTR,
           SUBSTR(RESTSTR, 2, LEN - 1),
           LEN - 1);
  END WHILE;
  RETURN REVSTR;
END
```

예 4: 감사를 사용하여 예 4의 테이블 함수를 정의하십시오.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                                    LASTNAME VARCHAR(15),
                                    FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
    MODIFIES SQL DATA
       NO EXTERNAL ACTION
     DETERMINISTIC
  BEGIN ATOMIC
          INSERT INTO AUDIT
    VALUES (USER,
                          'Table: EMPLOYEE Prd: DEPTNO = ' CONCAT DEPTNO);
    RETURN
              SELECT EMPNO, LASTNAME, FIRSTNME
        FROM EMPLOYEE
        WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
  END
```

CREATE FUNCTION MAPPING

CREATE FUNCTION MAPPING문은 다음을 수행하는 데 사용됩니다.

- 페더레이티드 데이터베이스 함수나 함수 템플리트와 데이터 소스 함수 사이에 맵핑을 정의합니다. 맵핑은 다음에서 페더레이티드 데이터베이스 함수나 템플리트를 함수와 연관지을 수 있습니다.
  - 지정된 데이터 소스
  - 데이터 소스의 범위(예를 들어, 특정 유형 및 버전의 모든 데이터 소스)
- 페더레이티드 데이터베이스 함수와 데이터 소스 함수 간의 디폴트 맵핑을 비활성화합니다.

다중 함수 맵핑을 함수에 적용할 수 있는 경우, 가장 최근 맵핑이 적용됩니다.

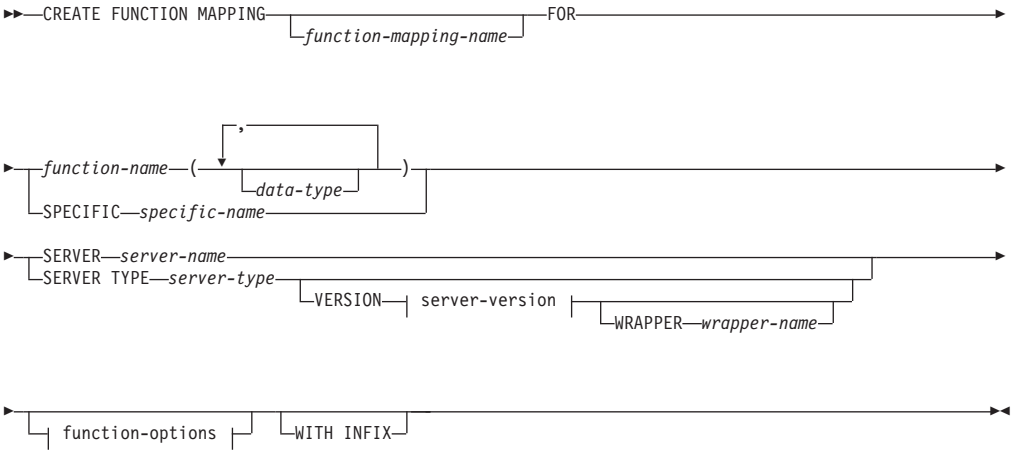
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

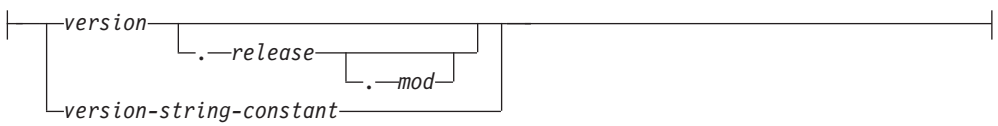
권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

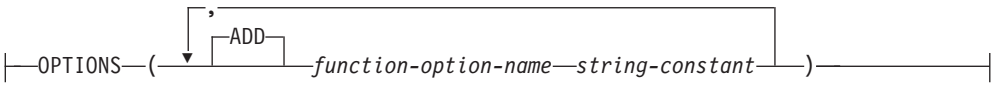
구문



server-version:



function-options:



설명

function-mapping-name

함수 맵핑에 이름을 지정합니다. 이름은 카탈로그에 이미 기술되어 있는 함수 맵핑을 식별해서는 안됩니다(SQLSTATE 42710).

*function-mapping-name*을 생략하면, 시스템에서 생성하는 고유 이름이 지정됩니다.

function-name

맵핑할 페더레이티드 데이터베이스 함수나 페더레이티드 데이터베이스 함수 템플리트의 규정된 이름 또는 규정되지 않은 이름을 지정합니다.

data-type

입력 매개변수가 있는 함수나 함수 템플리트의 경우, *data-type*은 각 매개변수의 데이터 유형을 지정합니다. *data type*은 XML 또는 사용자 정의 유형일 수 없습니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정하는 대신에 빈 괄호를 사용할 수 있습니다. 매개변수화된 데이터 유형에 대해서는 빈 괄호를 사용할 것을 권장합니다(예: CHAR()). 매개변수화된 데이터 유형은 특정 길이, 정밀도 또는 스케일을 사용하여 정의할 수 있는 데이터 유형 중 하나입니다. 매개변수화된 데이터 유형은 문자열 데이터 유형과 10진수 데이터 유형입니다. 길이, 정밀도 또는 스케일을 지정할 경우, 함수 템플리트의 것과 동일해야 합니다. 괄호 전체를 생략할 경우, 데이터 유형의 디폴트 길이가 사용됩니다(CREATE TABLE문의 설명 참조).

SPECIFIC *specific-name*

맵핑할 함수나 함수 템플리트를 식별합니다. 일반 함수 이름을 작성하도록 *specific-name*을 지정하십시오.

SERVER *server-name*

맵핑할 함수를 포함하는 데이터 소스에 이름을 지정합니다.

SERVER TYPE *server-type*

맵핑할 함수를 포함하는 데이터 소스의 유형에 이름을 지정합니다.

VERSION

*server-type*으로 표시되는 데이터 소스의 버전을 지정합니다.

version

버전 번호를 지정합니다. 이 값은 정수여야 합니다.

release

*version*으로 표시된 버전의 릴리스 번호를 지정합니다. 이 값은 정수여야 합니다.

mod

*release*로 표시된 릴리스의 수정 번호를 지정합니다. 이 값은 정수여야 합니다.

version-string-constant

버전의 전체 명칭을 지정합니다. *version-string-constant*는 단일 값(예: '8i')이거나 *version*, *release* 및 *mod*(적용 가능한 경우)의 조인된 값(예: '8.0.3')일 수 있습니다.

WRAPPER *wrapper-name*

server-type 및 *server-version*에 의해 표시된 유형 및 버전의 데이터 소스와 상호작용하기 위해 페더레이티드 서버가 사용하는 랩퍼의 이름을 지정합니다.

OPTIONS

어느 함수 맵핑 옵션이 작동 가능한지를 지정합니다.

ADD

하나 이상의 함수 맵핑 옵션을 활성화합니다.

function-option-name

함수 맵핑이나 맵핑에 포함된 데이터 소스 함수에 적용하는 함수 맵핑 옵션에 이름을 지정합니다.

string-constant

*function-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

WITH INFIX

데이터 소스 함수가 인픽스 형식으로 생성되도록 지정합니다. 페더레이티드 데이터베이스 시스템은 접두부 표기(prefix notation)를 리모트 데이터 소스가 사용하는 인픽스 표기(infix notation)로 변환합니다.

주

- 다음과 같은 경우에 페더레이티드 데이터베이스 함수나 함수 템플리트가 데이터 소스 함수에 맵핑될 수 있습니다.
  - 페더레이티드 데이터베이스 함수나 템플리트에는 데이터 소스 함수와 동일한 개수의 입력 매개변수가 있습니다.
  - 페더레이티드 함수나 템플리트에 정의되는 데이터 유형은 데이터 소스 함수에 대해 정의된 해당 데이터 유형과 호환됩니다.
- 분산 요청이 데이터 소스 함수에 맵핑되는 DB2 함수를 참조하면, 옵티마이저가 요청 처리시 함수를 호출하는 전략을 개발합니다. 그렇게 하여 DB2 함수를 호출하면 데이터 소스 함수를 호출하는 것보다 오버헤드가 덜 발생합니다. 그렇지 않고 DB2 함수 호출에 오버헤드가 더 많이 필요하게 되면 데이터 소스 함수가 호출됩니다.

- 분산 요청이 데이터 소스 함수에 맵핑되는 DB2 함수 템플리트를 참조하면, 요청 처리시 데이터 소스 함수만 호출될 수 있습니다. 템플리트는 실행 가능 코드가 없기 때문에 호출될 수 없습니다.
- 디폴트 함수 맵핑을 사용 안 함으로 설정하여(삭제할 수는 없음) 디폴트 함수 맵핑을 작동 불능으로 표시할 수 있습니다. 디폴트 함수 맵핑을 사용하지 않으려면 맵핑 내에 DB2 함수 이름을 지정하고 DISABLE 옵션이 'Y'로 설정되도록 CREATE FUNCTION MAPPING문을 코딩하십시오.
- SYSIBM 스키마에 있는 함수에는 특정 이름이 없습니다. SYSIBM 스키마에 있는 함수에 대한 디폴트 함수 맵핑을 겹쳐쓰려면, SYSIBM.LENGTH()와 같은 명시적 규정자 SYSIBM을 사용하여 *function-name*을 지정합니다.
- 다음 조건에서는 주어진 작업 단위(UOW) 내에서 CREATE FUNCTION MAPPING 문을 처리할 수 없습니다(SQLSTATE 55007).
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 내에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문
  - 명령문은 데이터 소스의 범주(예: 특정 유형 및 버전의 모든 데이터 소스)를 참조하며, UOW는 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 중 하나에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문

예:

예 1: 함수 템플리트를 모든 Oracle 데이터 소스가 액세스할 수 있는 UDF에 맵핑하십시오. 템플리트는 STATS라고 하며 NOVA라는 스키마에 속합니다. Oracle UDF는 STATISTICS라고 하며 STAR라는 스키마에 속합니다.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
  FOR NOVA.STATS (DOUBLE, DOUBLE)
  SERVER TYPE ORACLE
  OPTIONS (REMOTE_NAME 'STAR.STATISTICS')
```

예 2: BONUS라는 함수 템플리트를 ORACLE1이라는 Oracle 데이터 소스에서 사용되는 BONUS라고도 하는 UDF에 맵핑하십시오.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN2
  FOR BONUS()
  SERVER ORACLE1
  OPTIONS (REMOTE_NAME 'BONUS')
```

예 3: 페더레이티드 데이터베이스에 정의되는 WEEK 시스템 함수와 Oracle 데이터 소스에 정의되는 유사한 함수 간에 디폴트 함수 맵핑이 있다고 가정하십시오. Oracle 데이터를 요청하고 WEEK를 참조하는 쿼리가 처리될 때, 옵티마이저가 어느 것이 오버헤드가 덜 든다고 추정하는지에 따라 WEEK 또는 이것의 Oracle 상대편이 호출될 것입니다. DBA는 그러한 쿼리에 대해 WEEK만 호출된 경우 성능에 미치는 영향에 대해 알고 싶어합니다. 매번 WEEK가 호출되게 하려면, DBA가 맵핑을 사용 안 함으로 설정하십시오.

```
CREATE FUNCTION MAPPING
  FOR SYSFUN.WEEK(INT)
  SERVER TYPE ORACLE
  OPTIONS (DISABLE 'Y')
```

예 4: 페더레이티드 함수 UCASE(CHAR)를 ORACLE2라는 Oracle 데이터 소스에서 사용되는 UDF에 맵핑하십시오. Oracle UDF의 호출당 예상되는 명령어 수를 포함시키십시오.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4
  FOR SYSFUN.UCASE(CHAR)
  SERVER ORACLE2
  OPTIONS
    (REMOTE_NAME 'UPPERCASE',
    INSTS_PER_INVOC '1000')
```

CREATE GLOBAL TEMPORARY TABLE

CREATE GLOBAL TEMPORARY TABLE 문은 현재 서버에 있는 임시 테이블의 설명을 작성합니다. 작성된 임시 테이블에서 선택하는 각 세션은 동일한 세션이 삽입된 행만 검색합니다. 세션이 종료되면 세션과 연관된 테이블의 행이 삭제됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

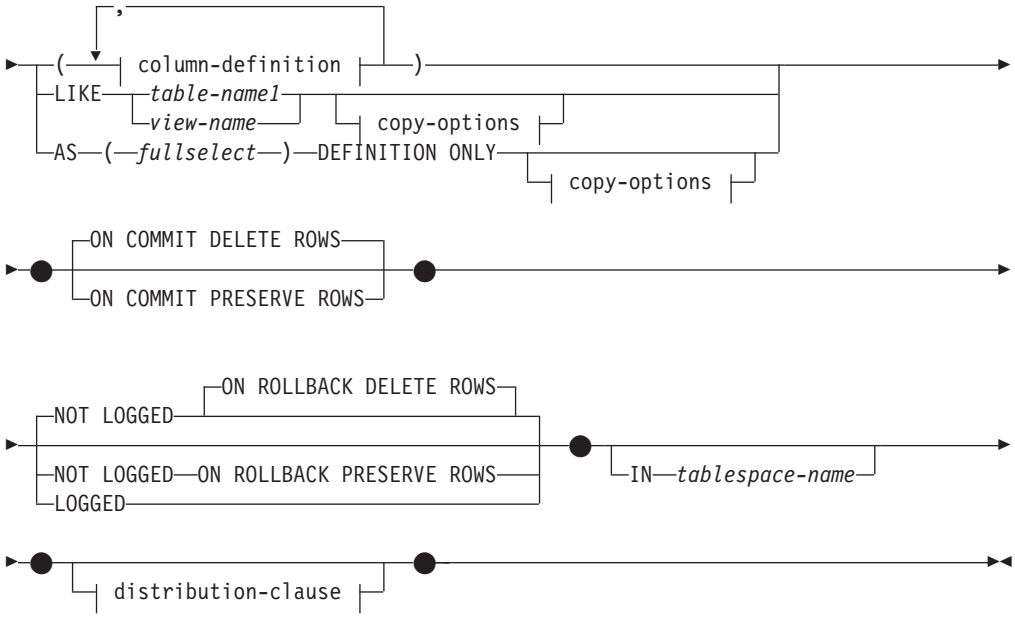
명령문의 권한 부여 ID가 보유하는 특권은 여기에 설명된 내용과 같이 추가 권한이 있는 조합에서 DBADM 권한 또는 CREATETAB 권한 중 하나를 포함해야 합니다.

- 다음 특권 및 권한 중 하나입니다.
  - 테이블 스페이스에 대한 USE 특권
  - SYSADM
  - SYSCTRL
- 다음 특권 및 권한에 하나를 더합니다.
  - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

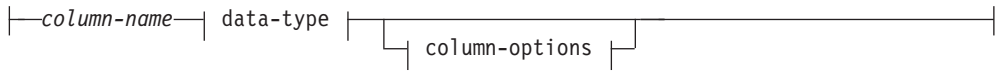
LIKE 또는 fullselect를 사용하여 테이블을 정의하는 경우 명령문의 권한 부여 ID가 보유하는 특권은 각각의 식별된 테이블이나 뷰에 대해 최소한 다음 중 하나를 포함해야 합니다.

- 테이블 또는 뷰에 대한 SELECT 특권
- 테이블 또는 뷰에 대한 CONTROL 특권
- DATAACCESS 권한

구문

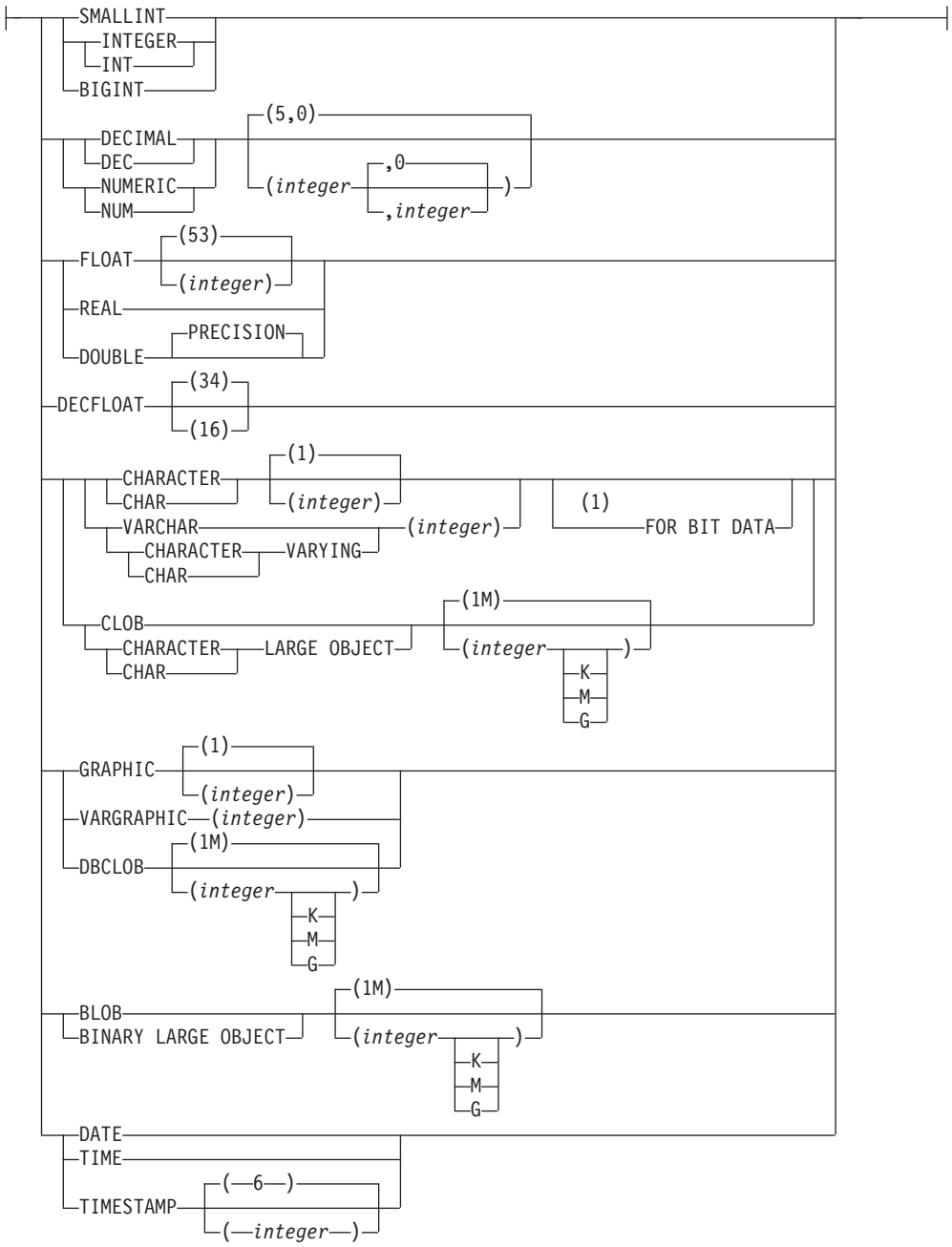


column-definition:

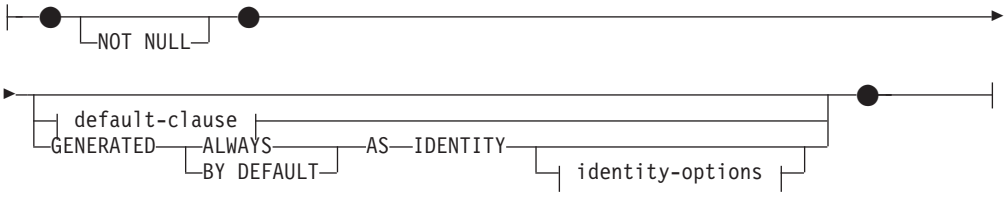


data-type:

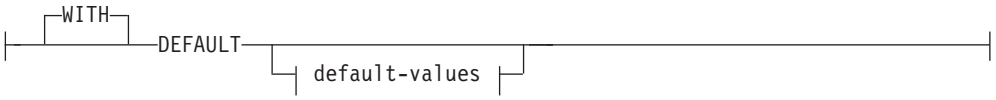
built-in-type:



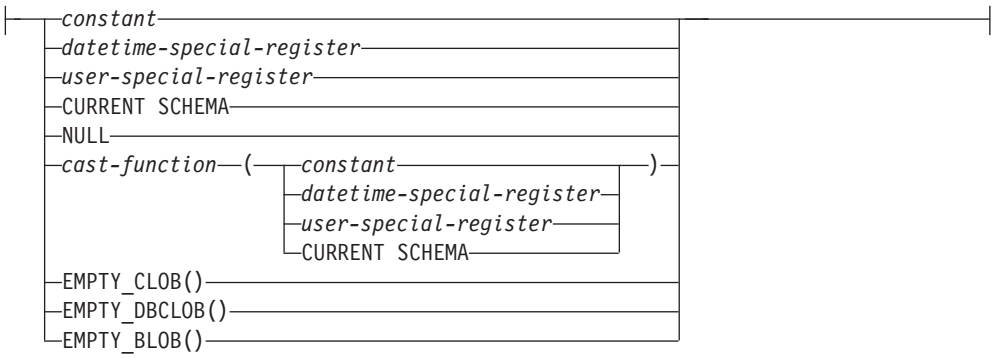
column-options:



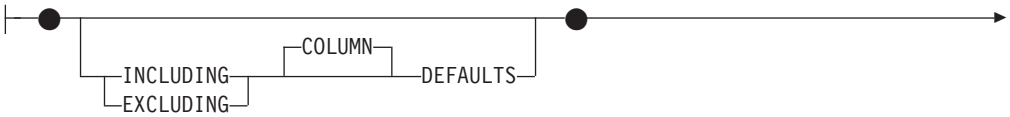
default-clause:

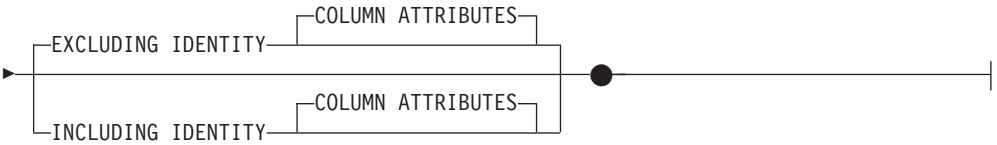


default-values:

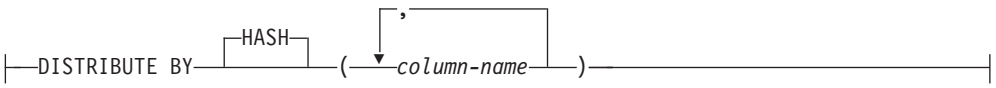


copy-options:





distribution-clause:



주:

1 FOR BIT DATA절은 다른 컬럼 제한조건에 대해 임의의 순서로 지정될 수 있습니다.

설명

table-name

테이블의 이름을 지정합니다. 내재적 또는 명시적인 규정자를 포함한 이름은 카탈로그에 기술된 테이블, 뷰, 별칭 또는 별명을 식별해서는 안됩니다. 두 부분으로 구성된 이름이 지정된 경우, 스키마 이름은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

column-definition

임시 테이블의 컬럼 속성을 정의합니다.

column-name

테이블의 컬럼을 이름 지정합니다. 이름을 규정할 수 없으며 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

테이블은 다음 사항을 수반할 수 있습니다.

- 최대 500개 컬럼의 4K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 4 005 이하여야 합니다.
- 최대 1 012개 컬럼의 8K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 8 101 이하여야 합니다.
- 최대 1 012개 컬럼의 16K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 16 293 이하여야 합니다.
- 최대 1 012개 컬럼의 32K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 32 677 이하여야 합니다.

세부사항은 『CREATE TABLE』의 『행 *크기*』를 참조하십시오.

data-type

컬럼의 데이터 유형을 지정합니다.

built-in-type

내장 데이터 유형을 지정합니다. *built-in-type*의 설명은 『CREATE TABLE』을 참조하십시오.

XML 및 SYSPROC.DB2SECURITYLABEL 데이터 유형은 작성된 임시 테이블에 지정될 수 없습니다.

distinct-type-name

구별 유형인 사용자 정의 유형의 경우. 구별 유형 이름을 *스키마* 이름없이 지정하면, 구별 유형 이름은 SQL 경로(정적 SQL의 경우 FUNCPATH 사전 처리 옵션에 의해 정의되고, 동적 SQL의 경우 CURRENT PATH 레지스터에 의해 정의됨)의 *스키마*를 검색하여 분석됩니다.

구별 유형을 사용하여 정의된, 컬럼의 데이터 유형은 구별 유형입니다. 컬럼의 길이와 *스케일*은 각각 구별 유형의 소스 유형의 길이와 *스케일*입니다.

column-options

테이블 컬럼에 관련된 추가 옵션을 정의합니다.

NOT NULL

컬럼에 널(NULL)값이 포함되지 못하도록 합니다. 널(NULL) 값 스펙은 『CREATE TABLE』의 NOT NULL을 참조하십시오.

default-clause

컬럼의 디폴트값을 지정합니다.

WITH

선택적 키워드입니다.

DEFAULT

값이 INSERT에 제공되지 않거나 INSERT 또는 UPDATE에서 DEFAULT로 지정된 경우 디폴트값을 제공합니다. 디폴트값을 DEFAULT 키워드 다음에 지정하지 않은 경우, 디폴트값은 『ALTER TABLE』에 표시된 컬럼의 데이터 유형에 따라 달라집니다.

컬럼이 유형이 지정된 테이블 컬럼을 기반으로 하는 경우, 디폴트값 정의시 특정 디폴트값을 지정해야 합니다. 유형이 지정된 테이블의 오브젝트 ID 컬럼에 대한 디폴트값은 지정할 수 없습니다(SQLSTATE 42997).

컬럼이 구별 유형을 사용하여 정의되면, 컬럼의 디폴트값은 구별 유형으로 캐스트되는 소스 데이터 유형의 디폴트값입니다.

구조화된 유형을 사용하여 컬럼을 정의한 경우, *default-clause*를 지정할 수 없습니다(SQLSTATE 42842).

*column-definition*에서 DEFAULT를 생략하면, 컬럼의 디폴트값으로 널(NULL)값이 사용됩니다. 그러한 컬럼은 NOT NULL로 정의할 경우 유효 디폴트값을 갖지 않습니다.

default-values

지정될 수 있는 디폴트값의 특정 유형은 다음과 같습니다.

constant

컬럼의 디폴트값으로 상수를 지정합니다. 지정된 상수는 다음과 같아야 합니다.

- 지정된 규칙에 따라 컬럼에 지정할 수 있는 값을 나타내야 합니다.
- 컬럼이 부동 소수점 데이터 유형으로 정의되어 있지 않는 한, 부동 소수점 상수가 아니여야 합니다.
- 컬럼의 데이터 유형이 10진수 부동 소수점인 경우 숫자 상수 또는 10진수 부동 소수점 특수 값이어야 합니다. 목표 컬럼이 DECFLOAT이면 부동 소수점 상수는 먼저 DOUBLE로 해석된 다음 10진수 부동 소수점으로 변환됩니다. DECFLOAT(16) 컬럼의 경우, 16보다 정밀도가 큰 10진 상수는 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터에서 지정된 반올림 모드를 사용하여 반올림됩니다.
- 상수가 10진 상수인 경우 컬럼 데이터 유형의 전체 자릿수를 초과하는 0이 아닌 자릿수를 갖지 않아야 합니다(예를 들면, 1.234는 DECIMAL(5,2) 컬럼에 대한 디폴트값이 될 수 없음).

- 따옴표, 16진 상수를 나타내는 X와 같은 도입 문자 및 상수가 *cast-function*의 인수일 때 완전한 함수 이름의 문자 및 괄호를 포함하여 254바이트 이하로 표시됩니다.

datetime-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 날짜 시간 특수 레지스터의 값(CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP)을 지정합니다. 컬럼의 데이터 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다(예: CURRENT DATE 지정시 DATE여야 함).

user-special-register

INSERT, UPDATE 또는 LOAD 사용시 사용자 특수 레지스터(CURRENT USER, SESSION_USER, SYSTEM_USER)의 값을 컬럼의 디폴트값으로 지정합니다. 컬럼의 데이터 유형은 사용자 특수 레지스터의 길이 속성보다 짧지 않은 길이의 문자열이어야 합니다. SESSION_USER 대신 USER를, CURRENT USER 대신 CURRENT_USER를 지정할 수 있습니다.

CURRENT SCHEMA

INSERT, UPDATE 또는 LOAD 사용시 CURRENT SCHEMA 특수 레지스터값을 컬럼의 디폴트값으로 지정합니다. CURRENT SCHEMA가 지정되면, 컬럼의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성과 같거나 그 이상인 문자열이어야 합니다.

NULL

컬럼의 디폴트값으로 널(NULL)을 지정합니다. NOT NULL이 지정되었으면, DEFAULT NULL은 동일한 컬럼 정의 내에 지정될 수 있으나, 이로 인해 디폴트값으로 컬럼을 설정하려고 시도할 때 오류가 발생합니다.

cast-function

이 형식의 디폴트값은 구별 유형, BLOB 또는 날짜 시간(DATE, TIME 또는 TIMESTAMP) 데이터 유형으로 정의된 컬럼과 함께 사용할 수 있습니다. 구별 유형의 경우, BLOB 또는 날짜 시간 유형을 따르는 구별 유형을 제외하고는 함수 이름이 컬럼의 구별 유형 이름과 일치해야 합니다. 스키마 이름으로 규정된 경우, 함수 이름이 구별 유형에 대한 스키마 이름과 같아야 합니다. 규정되지 않은 경우, 함수 결정의 스키마 이름은 구별 유형에 대한 스키마 이름과 같아야 합니다. 디폴트값이 상수인 날짜 시간 유형을 따르는 구별 유형의 경우, 함수가 사용되어야 하며, 함수 이름은 내재적 또는 명시적 스키마 이름이 SYSIBM인 구별 유형의 소스 유형 이름과 일치해야 합니다. 다른 날짜 시간 컬럼의 경우, 해당하는 날짜 시간 함수가 사용될 수도 있습니다. BLOB 또

는 BLOB을 기반으로 하는 구별 유형의 경우, 함수가 사용되어야 하며 함수 이름은 내재적 또는 명시적 스키마 이름으로 SYSIBM을 사용하는 BLOB이어야 합니다.

constant
상수를 인수로 지정합니다. 상수는 구별 유형의 소스 유형 또는 데이터 유형(구별 유형이 아닌 경우)에 대한 상수 규칙을 따라야 합니다. *cast-function*이 BLOB인 경우, 상수는 문자열 상수여야 합니다.

datetime-special-register
CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다.

user-special-register
CURRENT USER, SESSION_USER 또는 SYSTEM_USER를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 최소한 8바이트 길이의 문자열 데이터 유형이어야 합니다. *cast-function*이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

CURRENT SCHEMA
CURRENT SCHEMA 특수 레지스터의 값을 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성보다 크거나 같은 문자열이어야 합니다. cast-function이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

EMPTY_CLOB(), EMPTY_DBCLOB() 또는 EMPTY_BLOB()
컬럼의 디폴트값으로 영(0) 길이의 문자열을 지정합니다. 컬럼의 데이터 유형은 함수의 결과 데이터 유형을 기반으로 해야 합니다.

지정된 값이 유효하지 않으면, 오류가 발생합니다(SQLSTATE 42894).

IDENTITY 및 *identity-options*
ID 컬럼의 스펙은 『CREATE TABLE』의 IDENTITY 및 *identity-options*를 참조하십시오.

LIKE *table-name1* **또는** *view-name* **또는** *nickname*
테이블 컬럼이 식별된 테이블(*table-name1*), 뷰(*view-name*) 또는 별칭(*nickname*)의 컬럼과 정확하게 같은 이름 및 설명을 가지도록 지정합니다. LIKE 다음에 지정된 이름은 카탈로그 또는 선언된 임시 테이블에 있는 테이블, 뷰 또는 별칭을 식별해야 합니다. 유형이 지정된 테이블 또는 유형이 지정된 뷰는 지정할 수 없습니다(SQLSTATE 428EC). 보호 테이블은 지정할 수 없습니다(SQLSTATE 42962).

LIKE의 사용은 *n* 컬럼의 내재된 정의입니다. 여기서, *n*은 식별된 테이블(내재적으로 숨겨진 컬럼 포함), 뷰 또는 별칭의 컬럼 수입니다. 기존 테이블에서 내재적으로 숨겨진 컬럼에 해당되는 새 테이블의 컬럼도 내재적으로 숨겨진 컬럼으로 정의됩니다. 내재된 정의는 LIKE 이후에 식별되는 것에 따라 다릅니다.

- 식별된 테이블인 경우 내재된 정의에는 *table-name1*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다. EXCLUDING COLUMN DEFAULTS가 지정되지 않을 경우에는 컬럼 디폴트값도 포함됩니다.
- 식별된 뷰인 경우 내재된 정의에는 *view-name*에 정의된 fullselect의 각 결과 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다.
- 식별된 별칭인 경우 내재된 정의에는 *nickname*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다.

copy-attributes 절에 따라, 컬럼 디폴트값 및 식별 컬럼 속성을 포함하거나 제외시킬 수 있습니다. 내재된 정의에는 식별된 테이블, 뷰 또는 별칭의 다른 어느 속성도 포함되지 않습니다. 그러므로 새 테이블이 고유한 제한조건, 외부 키 제한조건, 트리거 또는 인덱스를 가지지 않습니다. 테이블은 IN절에 의해 내재적 또는 명시적으로 지정된 테이블 스페이스에 작성되며, 선택적 절이 지정되는 경우에만 테이블이 다른 선택적 절을 가집니다.

테이블이 LIKE절에서 식별되고 이 테이블에 ROW CHANGE TIMESTAMP 컬럼이 포함되는 경우, 새 테이블의 해당 컬럼은 ROW CHANGE TIMESTAMP 컬럼의 데이터 유형만 상속합니다. 새 컬럼은 생성된 컬럼으로 간주되지 않습니다.

AS (*fullselect*) DEFINITION ONLY

fullselect가 실행되는 경우 테이블 컬럼이 fullselect의 도출된 결과 테이블에 나타난 컬럼과 동일한 이름 및 설명을 갖도록 지정하십시오. AS(*fullselect*)의 사용은 작성된 임시 테이블에 대한 *n* 컬럼의 내재된 정의입니다. 여기서, *n*은 fullselect 결과 컬럼의 수입니다.

내재된 정의에는 *n* 컬럼에 대한 다음 속성이 포함됩니다(데이터 유형에 해당되는 경우).

- 컬럼 이름
- 데이터 유형, 길이, 정밀도 및 스케일
- 널(null) 가능성

다음 속성은 포함되지 않습니다(디폴트값 및 ID 속성은 *copy-options*을 사용하여 포함될 수 있습니다).

- 디폴트값
- ID 속성
- ROW CHANGE TIMESTAMP

내재된 정의에는 fullselect에서 참조되는 테이블 또는 뷰의 다른 선택적 속성은 포함되지 않습니다.

모든 선택 목록 요소는 고유한 이름이어야 합니다(SQLSTATE 42711). AS절은 고유한 이름을 제공하는 SELECT절에서 사용할 수 있습니다. fullselect는 호스트 변수나 포함 매개변수 표시문자를 참조해서는 안됩니다.

copy-options

이러한 옵션은 소스 결과 테이블 정의(테이블, 뷰 또는 fullselect)의 추가 속성을 복사할지 여부를 지정합니다.

INCLUDING COLUMN DEFAULTS

소스 결과 테이블 정의를 갱신할 수 있는 각 컬럼의 컬럼 디폴트값이 복사됩니다. 갱신 가능하지 않은 컬럼은 작성된 테이블의 해당 컬럼에 정의된 디폴트값을 가지지 않습니다.

LIKE *table-name1*이 지정되고 *table-name1*이 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별하는 경우에는 INCLUDING COLUMN DEFAULTS가 디폴트값입니다.

EXCLUDING COLUMN DEFAULTS

컬럼 디폴트값은 소스 결과 테이블 정의로부터 복사되지 않습니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별하는 경우를 제외하고는 이 절이 디폴트값입니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES

가능한 경우 식별 컬럼 속성(START WITH, INCREMENT BY 및 CACHE 값)이 소스 결과 테이블 정의로부터 복사됩니다. 테이블, 뷰 또는 fullselect의 해당 컬럼 요소가 테이블 컬럼의 이름이거나 직접 또는 간접적으로 식별 등록 정보를 갖는 기본 테이블 또는 작성된 임시 테이블의 컬럼 이름에 맵핑되는 뷰의 컬럼 이름인 경우 이 속성을 복사할 수 있습니다. 다른 모든 경우에는 새 임시 테이블의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.

- fullselect의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함됩니다(즉, 두 번 이상 동일 컬럼 선택).
- fullselect의 선택 목록에 여러 식별 컬럼이 포함됩니다(즉, 하나의 조인 포함).
- 식별 컬럼이 선택 목록의 표현식에 포함됩니다.
- fullselect에 집합 연산(union, except 또는 intersect)이 포함됩니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES

식별 컬럼 속성이 소스 결과 테이블 정의로부터 복사되지 않습니다.

ON COMMIT

COMMIT 조작 수행 시 작성된 임시 테이블에 대해 취하는 조치를 지정합니다. 디폴트값은 DELETE ROWS입니다.

DELETE ROWS

테이블에 대해 열려 있는 WITH HOLD 커서가 없는 경우 테이블의 모든 행이 삭제됩니다.

PRESERVE ROWS

테이블 행은 유지됩니다.

LOGGED 또는 NOT LOGGED

테이블에 대한 조작이 로깅되는지 여부를 지정합니다. 디폴트는 NOT LOGGED ON ROLLBACK DELETE ROWS입니다.

NOT LOGGED

테이블에 대한 삽입, 갱신 또는 삭제 조작은 로그되지 않지만 테이블의 작성 또는 삭제(drop)는 로그되도록 지정합니다. ROLLBACK(또는 ROLLBACK TO SAVEPOINT) 조작 중에 다음을 수행하십시오.

- 테이블이 작업 단위(UOW)(또는 세이브포인트) 내에 작성된 경우 테이블은 삭제(drop)됩니다.
- 테이블이 작업 단위(UOW)(또는 세이브포인트) 내에 작성된 경우 테이블은 재작성되지만 데이터는 포함되지 않습니다.

ON ROLLBACK

ROLLBACK(또는 ROLLBACK TO SAVEPOINT) 조작이 수행될 때 로그되지 않은 작성된 임시 테이블에 취할 조치를 지정합니다. 디폴트값은 DELETE ROWS입니다.

DELETE ROWS

테이블 데이터가 변경되었으면 모든 행은 삭제됩니다.

PRESERVE ROWS

테이블 행은 유지됩니다.

LOGGED

테이블에 대한 삽입, 갱신 또는 삭제 조작과 테이블의 작성 또는 삭제(drop)도 로그되도록 지정합니다.

IN *tablespace-name*

작성된 임시 테이블의 인스턴스가 작성될 테이블 스페이스를 식별합니다. 테이블 스페이스가 있어야 하고 이 테이블 스페이스는 명령문의 권한 부여 ID가 USE 특권을 가지는 사용자 임시 테이블 스페이스이어야 합니다(SQLSTATE 42501). 이 절이 지정되지 않은 경우 해당 명령문의 권한 부여 ID가 USE 특권을 가지는 가장 작은 충분한 페이지 크기의 사용자 임시 테이블 스페이스를 선택함으로써 이 테이

블의 테이블 스페이스를 결정합니다. 둘 이상의 테이블 스페이스가 규정되었다면 USE 특권이 부여된 사용자에 따라 환경설정이 제공됩니다.

1. 권한 부여 ID
2. 권한 부여 ID가 속한 그룹
3. PUBLIC

여전히 둘 이상의 테이블 스페이스가 규정되었다면 데이터베이스 관리 프로그램에서 최종 선택을 합니다. 사용자 임시 테이블 스페이스가 규정된 경우에는 오류가 발생합니다(SQLSTATE 42727).

테이블 스페이스의 판별은 다음 경우에 변경될 수 있습니다.

- 테이블 스페이스가 삭제(drop) 또는 작성되었을 때
- USE 특권이 부여되었거나 취소되었을 때

테이블의 페이지 크기가 충분한지 여부는 행의 바이트 합계 또는 컬럼 수에 의해 결정됩니다. 세부사항은 『CREATE TABLE』의 『행 크기』를 참조하십시오.

distribution-clause

데이터베이스 파티션 또는 다중 데이터베이스 파티션에 데이터를 분배하는 방법을 지정합니다.

DISTRIBUTE BY HASH (*column-name***,...)**

분산 키라고 불리는 지정 컬럼의 디폴트 해싱 함수 사용을 데이터베이스 파티션에 분산 메소드로 지정합니다. *column-name*은 테이블의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다(SQLSTATE 42703). 동일한 컬럼은 두 번 이상 식별될 수 없습니다(SQLSTATE 42709). 데이터 유형이 BLOB, CLOB, DBCLOB, XML, 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형인 컬럼은 분산 키의 일부로 사용될 수 없습니다(SQLSTATE 42962).

해당 절이 지정되지 않고 테이블이 여러 데이터베이스 파티션이 있는 다중 파티션 데이터베이스 파티션 그룹에 상주할 경우, 분산 키는 해당 데이터 유형이 분산 키에 대해 유효한 첫 번째 컬럼으로 정의됩니다.

디폴트 분산 키 요건을 충족시키는 컬럼이 없을 경우, 분산 키 없이 테이블이 작성됩니다. 이러한 테이블은 단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에서만 사용할 수 있습니다.

단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 있는 테이블의 경우, 분산 키에 대해 유효한 데이터 유형을 가진 컬럼의 콜렉션은 분산 키를 정의하는 데 사용될 수 있습니다. 이 절을 지정하지 않으면 분산 키가 작성되지 않습니다.

주

- 사용자 임시 테이블 스페이스는 작성된 임시 테이블이 작성되기 전에 존재해야 합니다(SQLSTATE 42727).

- *인스턴스화 및 종료:* 아래 설명에서 P는 세션이고 T는 세션 P에서 작성된 임시 테이블입니다.
  - 빈 인스턴스 T가 P에서 실행되는 T에 대한 첫 번째 참조의 결과로 작성됩니다.
  - P의 SQL문은 T에 대한 참조를 작성할 수 있습니다. P에서 T에 대한 모든 참조는 T의 동일한 인스턴스를 참조합니다.
  - ON COMMIT DELETE ROWS절이 내재적으로 또는 명시적으로 지정되었다고 가정할 때, 커미트 조작이 P에서 작업 단위(UOW)를 종료하고 T에 종속적인 P에 열려 있는 WITH HOLD 커서가 없다면 커미트에 DELETE FROM T 조작이 포함됩니다.
  - 롤백 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 해당 작업 단위 또는 세이브포인트에 T의 수정사항이 포함되는 경우,
    - NOT LOGGED가 지정되면 롤백에는 ON ROLLBACK PRESERVE ROWS도 지정되지 않은 경우 DELETE from T 조작이 포함됩니다. 이와 같은 경우 DELETE 조작은 T에서 모든 행을 삭제합니다.
    - NOT LOGGED가 지정되지 않으면 T는 변경되지 않습니다.

    롤백 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 T의 작성이 포함되면 롤백에 DROP T 조작이 포함됩니다.

    롤백 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 작성된 임시 테이블 T의 삭제(drop)가 포함되면 롤백이 테이블 삭제 실행을 취소합니다. NOT LOGGED가 지정되면 테이블도 비워집니다.
  - T를 참조한 응용프로그램 프로세스가 종료되거나 데이터베이스와의 연결이 끊기면 T의 개인용 인스턴스가 삭제되고 인스턴스화된 행이 파괴됩니다.
  - T가 참조된 서버로의 연결이 종료되면, T의 개인용 인스턴스가 삭제되고 인스턴스화된 행이 파괴됩니다.
- *작성된 임시 테이블 사용의 제한사항:* 작성된 임시 테이블에서는 다음 사항이 불가능합니다.
  - ALTER, LOCK 또는 RENAME문에 지정(SQLSTATE 42995)
  - 참조 제한조건에 지정(SQLSTATE 42995)
- *호환성:*
  - 이전 버전 DB2와의 호환성을 위해,
    - DISTRIBUTE BY절 대신 PARTITIONING KEY절 지정 가능
  - z/OS용 DB2와의 호환성을 위해,
    - 다음 구문이 디폴트 동작으로 허용됩니다.
      - CCSID ASCII
      - CCSID UNICODE

예 :

예 1: 임시 테이블 CURRENTMAP을 작성하십시오. 두 개 컬럼에 이름 CODE 및 MEANING을 지정하십시오. 둘 다 NULL 값을 포함할 수 없습니다. CODE에는 숫자 데이터가 포함되고 MEANING은 문자 데이터를 수반합니다.

```
CREATE GLOBAL TEMPORARY TABLE CURRENTMAP
   (CODE        INTEGER      NOT NULL,
    MEANING     VARCHAR(254) NOT NULL)
```

예 2: 임시 테이블 TMPDEPT를 작성하십시오.

```
CREATE GLOBAL TEMPORARY TABLE TMPDEPT
   (TMPDEPTNO   CHAR(3)      NOT NULL,
    TMPDEPTNAME VARCHAR(36)  NOT NULL,
    TMPMGRNO    CHAR(6),
    TMPLOCATION CHAR(16) )
```

CREATE HISTOGRAM TEMPLATE

CREATE HISTOGRAM TEMPLATE문은 서비스 클래스나 작업 클래스의 디폴트 막대 그래프 중 하나 이상의 대체하기 위해 사용할 수 있는 막대 그래프의 유형을 설명하는 템플리트를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

구문

설명

template-name

막대 그래프 템플리트의 이름입니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. 이름은 현재 서버에서 기존 막대 그래프 템플리트를 식별해서는 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

HIGH BIN VALUE *bigint-constant*

마지막 바이너리에 대한 최상위 초 값을 지정합니다(마지막 바이너리는 바운드되지 않은 최상위 값을 가짐). 단위는 막대 그래프 사용 방법에 따라 결정됩니다. 최대 값은 268 435 456입니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)

  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 모든 파티션 사이에서 한 번에 하나의 언커미트된 WLM 독점 SQL문만 허용됩니다. 언커미트된 WLM 독점 SQL문이 실행 중인 경우, 연속 WLM 독점 SQL문은 현재 WLM 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 발행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예 :

디폴트 활동 수명 막대 그래프 템플리트를 새로운 상위 바이너리 값인 90 000(90 000 마이크로초를 나타냄)으로 대체할 LIFETIMETEMP 막대 그래프 템플리트를 서비스 수퍼 클래스 ADMIN의 서비스 클래스 PAYROLL에 작성하십시오. 그러면 바이너리 범위가 지수 방식으로 증가하고 범위가 90 000에서 무제한까지인 바이너리로 끝나는 막대 그래프가 생성됩니다.

```
CREATE HISTOGRAM TEMPLATE LIFETIMETEMP
  HIGH BIN VALUE 90000

CREATE SERVICE CLASS PAYROLL
  UNDER ADMIN ACTIVITY LIFETIME HISTOGRAM TEMPLATE LIFETIMETEMP
```

CREATE INDEX

CREATE INDEX문은 다음을 수행하는 데 사용됩니다.

- DB2 테이블에 인덱스를 정의하십시오. 인덱스는 XML 데이터 또는 관계형 데이터에 정의될 수 있습니다.
- 인덱스 스펙(옵티마이저에게 데이터 소스 테이블에 인덱스가 있음을 나타내는 메타데이터) 작성

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

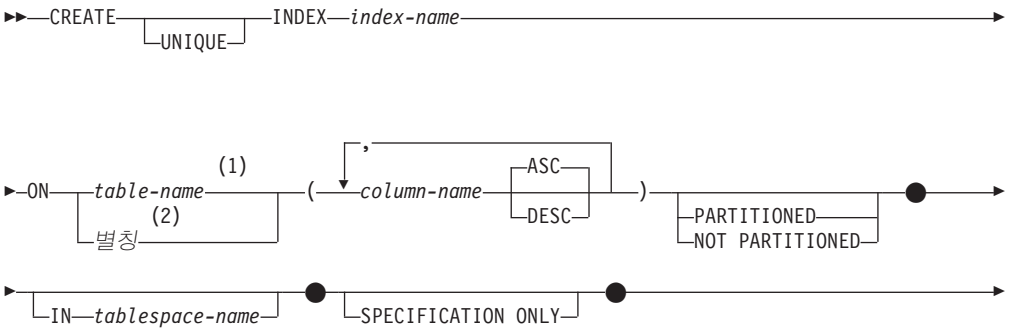
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

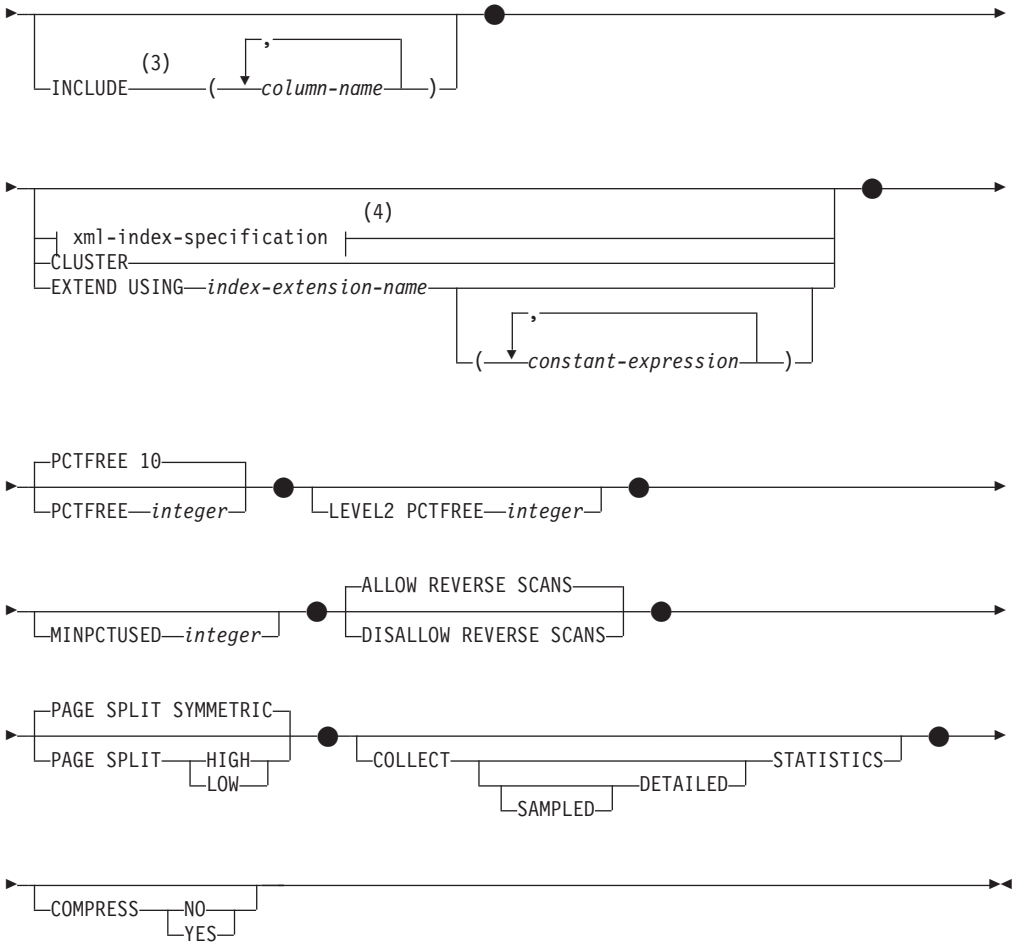
- 다음 중 하나
  - 인덱스가 정의되는 테이블이나 별칭에 대한 CONTROL 특권
  - 인덱스가 정의되는 테이블이나 별칭에 대한 INDEX 특권

  다음 중 하나
  - 인덱스의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 인덱스의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

선언된 임시 테이블에 인덱스를 작성할 때는 명시적 특권이 필요하지 않습니다.

구문





주:

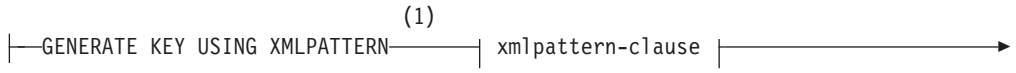
1 페더레이티드 시스템에서 *table-name*은 페더레이티드 데이터베이스의 테이블을 식별해야 합니다. 데이터 소스 테이블을 식별할 수는 없습니다.

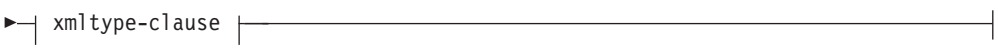
2 *nickname*이 지정되면, CREATE INDEX문이 인덱스 스펙을 작성합니다. 이 경우에는, INCLUDE, *xml-index-specification*, CLUSTER, EXTEND USING, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, ALLOW REVERSE SCANS, PAGE SPLIT 또는 COLLECT STATISTICS는 지정할 수 없습니다.

3 INCLUDE절은 UNIQUE가 지정된 경우에만 지정할 수 있습니다.

4 *xml-index-specification*을 지정하면 *column-name* DESC, INCLUDE 또는 CLUSTER를 지정할 수 없습니다.

xml-index-specification:

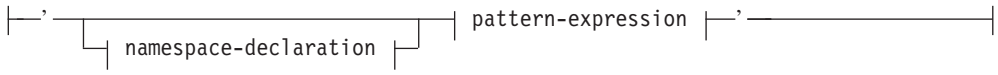




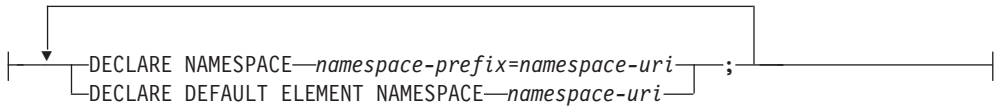
주:

1 대안 구문 GENERATE KEYS USING XMLPATTERN을 사용할 수 있습니다.

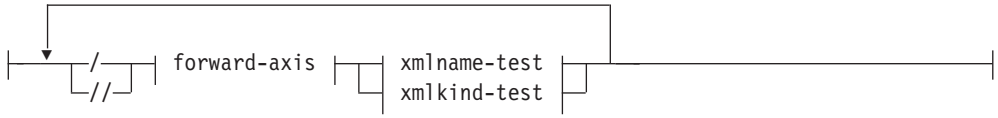
xmlpattern-clause:



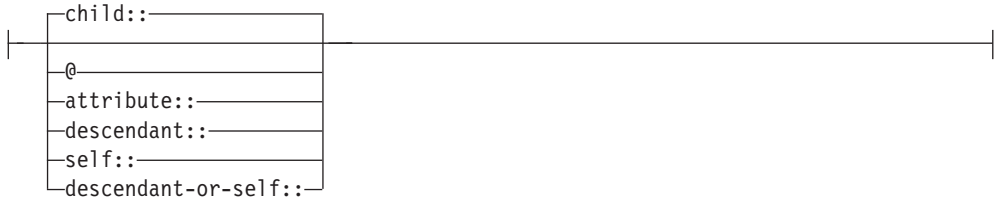
namespace-declaration:



pattern-expression:



forward-axis:

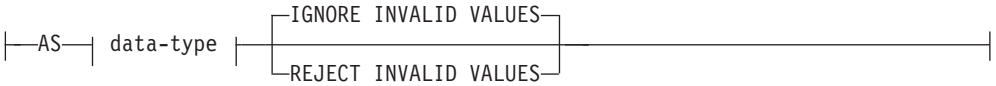


xmlname-test:

xml-wildcard:

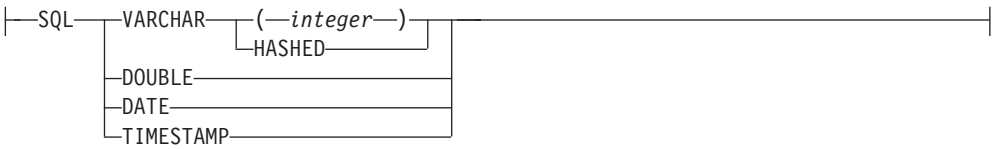
xmlkind-test:

xmltype-clause:



data-type:

sql-data-type:



설명

UNIQUE

ON *table-name*을 지정할 경우, UNIQUE는 테이블에 인덱스 키와 같은 값을 갖는 둘 이상의 행이 포함되지 못하도록 합니다. 고유성은 행을 갱신하거나 새로운 행을 삽입하는 SQL문의 끝에서 적용됩니다.

CREATE INDEX문이 실행될 때에도 고유성이 점검됩니다. 테이블에 중복되는 키 값을 갖는 행을 포함할 경우 인덱스는 작성되지 않습니다.

XML 데이터를 초과하는 인덱스가 XML 컬럼에 있는 경우 테이블의 모든 행에 대해 지정된 *pattern-expression*을 가진 값에 고유성이 적용됩니다. 값을 지정된 *sql-data-type*으로 변환한 후 각 값에 대해 고유성이 적용됩니다. 지정된 *sql-data-type*으로의 변환이 정밀도 또는 범위의 손실을 가져올 수 있거나 다른 값이 동일한 키 값으로 해시될 수 있기 때문에, XML 문서에서 고유하게 보이는 복수의 값은 중복 키 오류를 발생시킬 수 있습니다. 문자열의 고유성은 뒤 공백이 유효한 XQuery 시맨틱에 따라 다릅니다. 따라서 뒤 공백이 다른 값은 SQL에서는 중복되나 XML 데이터에 있는 인덱스에서는 고유하다고 간주됩니다.

UNIQUE가 사용되면, 널(NULL) 값이 다른 값처럼 처리됩니다. 예를 들어, 키가 널(NULL) 값을 포함하는 단일 컬럼일 경우, 그 컬럼에는 두 개 이상의 널(NULL) 값을 포함할 수 없습니다.

UNIQUE 옵션이 지정되고 테이블에 분산 키가 있는 경우, 인덱스 키 컬럼은 분산 키의 수퍼 세트여야 합니다. 즉 고유 인덱스 키에 대해 지정된 컬럼에는 분산 키의 컬럼이 모두 포함되어야 합니다(SQLSTATE 42997).

UNIQUE 옵션이 지정되고 테이블에 테이블 파티션 키가 있는 경우 인덱스 키의 컬럼은 테이블 파티셔닝 키의 수퍼 세트여야 합니다. 즉, 고유 인덱스 키에 대해 지정된 컬럼에는 테이블 파티셔닝 키의 컬럼이 모두 포함되어야 합니다(SQLSTATE 42990).

기본 키 또는 고유 키는 차원의 서브 세트가 될 수 없습니다(SQLSTATE 429BE).

ON *nickname*이 지정되면, 인덱스 키에 대한 데이터가 데이터 소스 테이블의 모든 행에 대해 고유한 값을 포함하는 경우에만 UNIQUE를 지정해야 합니다. 고유성이 점검되지 않습니다.

XML 데이터에 있는 인덱스의 경우, 지정된 *pattern-expression*이 단일 전체 경로를 지정하고 하위 구성원 또는 descendant-or-self 축, *"//"*, *xml-wildcard*, *node()*, 또는 *processing-instruction()*을 포함하지 않을 경우에만 UNIQUE를 지정할 수 있습니다(SQLSTATE 429BS).

파티션된 데이터베이스 환경에서, 하나 이상의 XML 컬럼이 포함된 테이블에는 다음 규칙이 적용됩니다.

- 분산 테이블의 경우 XML 데이터에 고유 인덱스를 포함할 수 없습니다.
- XML 데이터의 고유 인덱스는 분산 키가 없고 단일 노드의 멀티파티션 데이터베이스에서만 지원됩니다.
- XML 데이터의 고유 인덱스가 테이블에 있는 경우 분산 키를 추가하기 위해 테이블을 변경할 수 없습니다.

INDEX *index-name*

인덱스나 인덱스 스펙에 이름을 지정합니다. 내재적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 기술된 인덱스 또는 인덱스 스펙이나 선언된 임시 테이블의 기존 인덱스를 식별해서는 안됩니다. 규정자는 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT(SQLSTATE 42939)이 될 수 없습니다.

선언된 전역 임시 테이블의 인덱스에 대한 내재적 규정자나 명시적 규정자는 SESSION이어야 합니다(SQLSTATE 428EK).

table-name 또는 *nickname*

*table-name*은 인덱스가 작성될 테이블을 식별합니다. 테이블은 현재 서버 또는 선언된 임시 테이블에 존재하는 기본 테이블(보기 아님), 작성된 임시 테이블, 선언된 임시 테이블, 구체화된 쿼리 테이블이어야 합니다. 선언된 임시 테이블의 이름은 SESSION을 사용하여 규정해야 합니다. *table-name*은 카탈로그 테이블을 식별해서는 안됩니다(SQLSTATE 42832). UNIQUE가 지정되어 있고 *table-name*이 유형이 지정된 테이블인 경우, 서브테이블이어서는 안됩니다(SQLSTATE 429B3).

*nickname*은 인덱스 스펙이 작성될 테이블에 대한 별칭입니다. *nickname*은 인덱스 스펙에 의해 인덱스가 설명되는 데이터 소스나 그러나 테이블에 기초하는 데이터 소스 뷰를 참조합니다. *nickname*은 카탈로그에 나열되어야 합니다.

column-name

인덱스의 경우 *column-name*은 인덱스 키의 일부가 될 컬럼을 식별합니다. 인덱스 스펙의 경우 *column-name*은 페더레이티드 서버가 데이터 소스 테이블의 컬럼을 참조하는 이름입니다.

각 *column-name*은 테이블의 컬럼을 식별하는 규정된 이름이어야 합니다. 최대 64개의 컬럼을 지정할 수 있습니다. *table-name*이 유형이 지정된 테이블인 경우, 최대 63개의 컬럼을 지정할 수 있습니다. *table-name*이 서브테이블일 경우, 최소한 하나의 *column-name*이 서브테이블에 도입되어야 합니다, 즉, 슈퍼 테이블로부터 상속되지 않아야 합니다(SQLSTATE 428DS). *column-name*은 반복할 수 없습니다(SQLSTATE 42711).

지정된 컬럼의 저장된 길이의 합은 페이지 *크기*의 인덱스 키 길이 한계보다 작아야 합니다. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오. *table-name*이 유형이 지정된 테이블인 경우, 인덱스 길이 한계는 4바이트 단위로 추가로 감소합니다. 컬럼의 데이터 유형과 널(NULL) 입력 가능 여부에 따라 변하는 시스템 오버헤드에 의해 이 길이 한계가 감소할 수 있다는 점에 유의하십시오. 이 한계에 영향을 주는 오버헤드에 대한 자세한 정보는 『CREATE TABLE』의 『바이트 수』를 참조하십시오.

컬럼의 데이터 유형과 널(NULL) 입력 가능 여부에 따라 시스템 오버헤드에 의해 이 길이가 감소할 수 있습니다. 이 한계에 영향을 주는 오버헤드에 대한 자세한 정보는 『CREATE TABLE』의 『바이트 수』를 참조하십시오.

컬럼의 길이 속성이 페이지 *크기*에 대한 인덱스 키 길이 한계를 초과하지 않아도 LOB 컬럼 또는 LOB를 기반으로 하는 구별 유형은 인덱스의 일부로 사용될 수 없습니다(SQLSTATE 54008). 구조화된 유형 컬럼은 EXTEND USING절도 지정할 경우에만 지정할 수 있습니다(SQLSTATE 42962). EXTEND USING절을 지정할 경우, 하나의 컬럼만 지정할 수 있으며 컬럼의 유형이 LOB를 기반으로 하지 않는 구조화된 유형이나 구별 유형이어야 합니다(SQLSTATE 42997).

인덱스에 하나의 컬럼만 있고 해당 컬럼의 데이터 유형이 XML이며 GENERATE KEY USING XMLPATTERN 절도 지정된 경우, 인덱스는 XML 데이터에 있는 인덱스입니다. GENERATE KEY USING XMLPATTERN절도 지정된 경우에만 XML 데이터 유형이 있는 컬럼이 지정될 수 있습니다(SQLSTATE 42962). GENERATE KEY USING XMLPATTERN절이 지정된 경우 하나의 컬럼만이 지정될 수 있고 컬럼 유형은 XML이어야 합니다.

ASC

인덱스 항목이 컬럼 값의 오름차순으로 보존되도록 지정합니다. 이것이 디폴트 설정입니다. ASC는 EXTEND USING으로 정의된 인덱스에 대해 지정할 수 없습니다(SQLSTATE 42601).

DESC

인덱스 항목이 컬럼 값의 내림차순으로 보존되도록 지정합니다. 인덱스가 EXTEND USING을 사용하여 정의되었거나 인덱스가 XML 데이터에 있는 인덱스인 경우에는 DESC를 지정할 수 없습니다(SQLSTATE 42601).

PARTITIONED

파티션된 인덱스를 작성해야 함을 표시합니다. *table-name*은 데이터 파티션으로 정의된 테이블을 식별해야 합니다(SQLSTATE 42601).

테이블이 파티션되고 PARTITIONED 및 NOT PARTITIONED가 지정되지 않은 경우 인덱스는 파티션되어 지정됩니다. 이 경우 몇 가지 예외는 있습니다. 다음 상황 중 하나인 경우 파티션된 인덱스 대신 파티션되지 않은 인덱스가 작성됩니다.

- UNIQUE가 지정되고 인덱스 키가 모든 테이블 파티션 키 컬럼을 포함하지 않습니다.
- 공간 인덱스가 작성됩니다.
- 인덱스가 XML 데이터에 대해 정의됩니다.

파티션되지 않은 인덱스 정의와 중복되는 정의가 포함된 파티션된 인덱스는 중복 인덱스로 고려되지 않습니다. 자세한 내용은 이 주제의 554 페이지의 『규칙』 섹션을 참조하십시오.

PARTITIONED 키워드는 다음 인덱스에 지정할 수 없습니다.

- 파티션되지 않은 테이블의 인덱스(SQLSTATE 42601)
- XML 데이터에 대해 정의된 인덱스(SQLSTATE 42613)
- 인덱스 키가 모든 테이블 파티셔닝 키 컬럼을 포함하지 않는 고유 인덱스(SQLSTATE 42990)
- 공간 인덱스(SQLSTATE 42997)

파티션된 인덱스는 접속 해제된 종속 테이블이 있는 파티션된 테이블에 작성될 수 없습니다(예: MQTs)(SQLSTATE 55019).

파티션된 인덱스의 인덱스 파티션에 대한 테이블 스페이스 배치는 다음 규칙으로 판별됩니다.

- CREATE TABLE문의 partition-tablespace-options INDEX IN절을 사용하여 인덱스화 중인 테이블을 작성한 경우, 인덱스 파티션은 INDEX IN절에 지정된 테이블 스페이스에 작성됩니다.

- 인덱스화 중인 테이블의 CREATE TABLE문이 partition-tablespace-options INDEX IN절을 지정하지 않는 경우 인덱스 파티션은 인덱스화한 해당 데이터 파티션과 동일한 테이블 스페이스에 작성됩니다.

CREATE INDEX문의 IN절은 파티션된 인덱스에서 지원되지 않습니다(SQLSTATE 42601). CREATE TABLE문의 tablespace-clauses INDEX IN절이 파티션된 인덱스에 대해 무시됩니다.

NOT PARTITIONED

테이블에 대해 정의된 모든 데이터 파티션을 포함하는 파티션되지 않은 인덱스가 작성되어야 함을 표시합니다. *table-name*은 데이터 파티션으로 정의된 테이블을 식별해야 합니다(SQLSTATE 42601).

파티션된 인덱스 정의와 중복되는 정의가 포함된 파티션되지 않은 인덱스는 중복 인덱스로 고려되지 않습니다. 자세한 내용은 이 주제의 554 페이지의 『규칙』 섹션을 참조하십시오.

파티션되지 않은 인덱스의 테이블 스페이스 배치는 다음 규칙으로 판별됩니다.

- CREATE INDEX문에 IN절을 지정하는 경우 파티션되지 않은 인덱스는 해당 IN절에 지정된 테이블 스페이스에 배치됩니다.
- CREATE INDEX문의 IN절을 지정하지 않으면, 다음 규칙이 파티션되지 않은 인덱스의 테이블 스페이스 배치를 판별합니다.
  - CREATE TABLE문의 tablespace-clauses INDEX IN절을 사용하여 인덱스화 중인 테이블을 작성한 경우, 파티션되지 않은 인덱스는 INDEX IN절에 지정된 테이블 스페이스에 작성됩니다.
  - CREATE TABLE문의 tablespace-clauses INDEX IN절을 사용하지 않고 인덱스화 중인 테이블을 작성한 경우, 파티션되지 않은 인덱스는 테이블의 접속된 데이터 파티션 또는 첫 번째 보이는 데이터 파티션의 테이블 스페이스에 작성됩니다. 테이블의 접속된 데이터 파티션 또는 첫 번째 보이는 데이터 파티션은 데이터 파티션 목록의 첫 번째 파티션입니다. 또한 명령문의 권한 부여 ID는 디폴트 테이블 스페이스에 대한 USE 특권을 가질 필요가 없습니다.

IN *tablespace-name*

IN절은 파티션되지 않은 인덱스에 대해서만 지원됩니다. 파티션된 인덱스에 대해 IN절을 지정하면 SQLSTATE 42601이 발생합니다.

인덱스가 작성되는 테이블 스페이스를 지정합니다. 이 절은 작성된 임시 테이블이나 선언된 임시 테이블의 인덱스를 지원하지 않습니다(SQLSTATE 42601). 테이블 작성시 INDEX IN절이 지정되었다 해도 이 절을 지정할 수 있습니다. 해당 절이 INDEX IN절을 겹쳐씁니다.

*tablespace-name*에서 지정된 테이블 스페이스는 테이블에 대한 데이터 테이블 스페이스와 동일한 데이터베이스 파티션 그룹에 속해야 하고, 파티션된 테이블의 다

른 테이블 스페이스와 동일한 방법으로 관리되어야 합니다(SQLSTATE 42838). 또한 이 테이블 스페이스는 명령문의 권한 부여 ID에 USE 특권이 있는 테이블 스페이스여야 합니다.

IN절이 지정되지 않은 경우 CREATE TABLE문의 INDEX IN절이 지정한 테이블 스페이스에 인덱스가 작성됩니다. INDEX IN절이 지정되지 않은 경우 테이블의 첫 번째 보이는 데이터 파티션 또는 접속된 데이터 파티션의 테이블 스페이스가 쓰입니다. 범위 스펙의 기초에 정렬된 데이터 파티션 목록의 첫 번째 파티션입니다. IN 절이 지정되지 않으면 명령문의 권한 부여 ID는 기본 테이블 스페이스에서 USE 특권을 가질 필요가 없습니다.

SPECIFICATION ONLY

이 명령문이 *nickname*에 의해 참조된 데이터 소스 테이블에 적용되는 인덱스 스펙을 작성하는 데 사용될 것임을 나타냅니다. *nickname*을 지정할 경우에는 SPECIFICATION ONLY를 지정해야 합니다(SQLSTATE 42601). *table-name*을 지정할 경우에는 이를 지정하면 안됩니다(SQLSTATE 42601).

인덱스 스펙이 고유한 인덱스에 적용되는 경우 DB2는 리모트 테이블의 컬럼 값이 고유한지 확인하지 않습니다. 리모트 컬럼 값이 고유하지 않은 경우 인덱스 컬럼을 포함하는 별칭에 반하는 쿼리는 틀린 데이터 또는 오류를 리턴할 수 있습니다.

작성된 임시 테이블이나 선언된 임시 테이블에 인덱스를 작성할 때 이 절을 사용할 수 없습니다(SQLSTATE 42995).

INCLUDE

이 키워드는 인덱스 키 컬럼 세트에 추가될 컬럼을 지정하는 절을 사용합니다. 고유성을 강화하기 위해 이 절에 포함된 컬럼이 사용되지는 않습니다. 포함된 컬럼은 인덱스 전용 액세스를 통해서 일부 쿼리 성능을 향상시킬 수 있습니다. 컬럼들은 고유성을 강화시키는 데 사용되는 컬럼과 구별되어야 합니다(SQLSTATE 42711). INCLUDE 지정시 UNIQUE가 지정되야 합니다(SQLSTATE 42613). 컬럼 수 및 길이 속성의 합계에 대한 한계는 고유 키 및 인덱스의 모든 컬럼에 적용됩니다.

이 절은 작성된 임시 테이블이나 선언된 임시 테이블에는 사용할 수 없습니다 (SQLSTATE 42995).

column-name

인덱스에 포함되지만 고유 인덱스 키의 일부는 아닌 컬럼을 식별합니다. 고유한 인덱스 키 컬럼에 적용된 것과 동일한 규칙이 적용됩니다. column-name 다음에 ASC 또는 DESC 키워드를 지정할 수 있으나 순서에는 영향을 미치지 않습니다.

*nickname*이 지정된 경우 또는 XML 값 인덱스의 경우, EXTEND USING으로 정의된 인덱스에 대해 INCLUDE를 지정할 수 없습니다(SQLSTATE 42601).

xml-index-specification

XML 컬럼에서 정렬된 XML 문서로부터 인덱스 키가 생성되는 방법을 지정합니

다. 두 개 이상의 인덱스 컬럼이 있거나 컬럼에 XML 데이터 유형이 없는 경우 *xml-index-specification*을 지정할 수 없습니다.

이 절은 XML 컬럼에만 적용됩니다(SQLSTATE 429BS).

GENERATE KEY USING XMLPATTERN *xmlpattern-clause*

인덱스될 XML 문서의 부분을 지정합니다. XML 패턴 값은 *xmlpattern-clause*에서 생성된 인덱스된 값입니다. 인덱스에서는 목록 데이터 유형 노드가 지원되지 않습니다. 노드가 *xmlpattern-clause*에 의해 규정되었고 노드가 목록 데이터 유형임을 지정하는 XML 스키마가 존재할 경우에는 목록 데이터 유형 노드를 인덱스화할 수 없습니다 (CREATE INDEX문의 경우에는 SQLSTATE 23526, INSERT 및 UPDATE문의 경우에는 SQLSTATE 23525).

xmlpattern-clause

인덱스될 노드를 식별하는 패턴 표현식을 포함합니다. 이 절은 선택적 *namespace-declaration* 및 필수 *pattern-expression*으로 구성됩니다.

namespace-declaration

패턴 표현식이 규정된 이름을 포함하는 경우 *namespace-declaration*은 이름 스페이스 접두어를 정의하도록 지정되어야 합니다. 규정되지 않은 이름에 대해 디폴트 이름 스페이스를 정의할 수 있습니다.

DECLARE NAMESPACE *namespace-prefix*=*namespace-uri*

NCName인 *namespace-prefix*를 문자열 리터럴인 *namespace-uri*에 맵핑합니다. *namespace-declaration*은 다중 *namespace-prefix*-to-*namespace-uri* 맵핑을 포함할 수 있습니다. *namespace-prefix*는 *namespace-declaration* 목록에서 고유해야 합니다(SQLSTATE 10503).

DECLARE DEFAULT ELEMENT NAMESPACE *namespace-uri*

규정되지 않은 요소 이름 또는 유형에 대한 디폴트 이름 스페이스 URI를 선언합니다. 디폴트 이름 스페이스가 선언되지 않으면 요소 및 유형의 규정되지 않은 이름이 이름 스페이스에 존재하지 않습니다. 한 개의 디폴트 이름 스페이스만이 선언될 수 있습니다 (SQLSTATE 10502).

pattern-expression

인덱스된 XML 문서의 노드를 지정합니다. *pattern-expression*은 패턴 일치 문자(*)를 포함할 수 있습니다. 이 절은 XQuery의 경로 표현식과 유사하지만 DB2가 지원하는 XQuery 언어의 서브세트를 지원합니다.

/(정방향 슬래시)

경로 표현식 단계를 분리합니다.

///(정방향 슬래시 두 개)

이 기호는 /descendant-or-self::node()/에 대한 약어 구문입니다. UNIQUE도 지정한 경우 *///(정방향 슬래시 두 개)* 구문을 사용할 수 없습니다.

forward-axis

child::

컨텍스트 노드의 하위를 지정합니다. 다른 정방향 축이 지정되어 있지 않으면 디폴트값이 됩니다.

@ 컨텍스트 노드의 속성을 지정합니다. attribute::의 약어 구문입니다.

attribute::

컨텍스트 노드의 속성을 지정합니다.

descendant::

컨텍스트 노드의 하위 구성원을 지정합니다. UNIQUE도 지정한 경우 *descendant::*를 사용할 수 없습니다.

self::

컨텍스트 노드 자체를 지정합니다.

descendant-or-self::

컨텍스트 노드 및 컨텍스트 노드의 하위 구성원을 지정합니다. UNIQUE를 지정할 경우에는 descendant-or-self::를 사용할 수 없습니다.

xmlname-test

규정된 XML 이름(xml-qname) 또는 와일드 카드(xml-wildcard)를 사용하여 경로의 단계에 노드 이름을 지정합니다.

xml-ncname

XML 1.0이 정의한 XML 이름입니다. 콜론을 포함할 수 없습니다.

xml-qname

두 가지 양식을 가질 수 있는 규정된 XML 이름을 지정합니다(QName이라고 함).

- xml-nsprefix가 유효 범위 안에 있는 이름 스페이스를 식별하는 xml-ncname인 xml-nsprefix:xml-ncname
- 디폴트 이름 스페이스가 내재적 xml-nsprefix로 적용되어야 함을 표시하는 xml-ncname

xml-wildcard

xml-qname을 세 가지 양식을 가질 수 있는 와일드 카드로 지정합니다.

- *(단일 별표 문자)는 모든 xml-qname 표시
- *xml-nsprefix*:*는 지정된 이름 스페이스의 xml-ncname를 표시
- *:*xml-ncname*은 유효 범위 안에 있는 이름 스페이스의 지정된 XML 이름을 표시

UNIQUE를 지정한 경우 *xml-wildcard*를 사용할 수 없습니다.

xmlkind-test

이 옵션을 사용하여 어떤 노드 유형을 패턴 일치할지 지정하십시오. 다음 옵션을 사용할 수 있습니다.

node()

모든 노드를 일치시킵니다. UNIQUE도 지정한 경우 *node()*를 사용할 수 없습니다.

text()

모든 텍스트 노드를 일치시킵니다.

comment()

모든 주석 노드를 일치시킵니다.

processing-instruction()

모든 명령어 노드 처리를 일치시킵니다. UNIQUE도 지정한 경우 processing-instruction()을 사용할 수 없습니다.

xmltype-clause

AS *data-type*

저장하기 전에 인덱스된 값을 변환할 데이터 유형을 지정합니다. 값은 지정된 인덱스 SQL 데이터 유형에 해당하는 인덱스 XML 데이터 유형으로 변환됩니다.

표 16. 해당 인덱스 데이터 유형

인덱스 **XML** 데이터 유형	인덱스 **SQL** 데이터 유형
xs:string	VARCHAR(*integer*), VARCHAR HASHED
xs:double	DOUBLE
xs:date	DATE
xs:dateTime	TIMESTAMP

VARCHAR(*integer*) 및 VARCHAR HASHED의 경우, 값은 XQuery 함수 fn:string을 사용하여 xs:string 값으로 변환됩니다. VARCHAR(*integer*)의 길이 속성은 제한조건으로서 결과 xs:string 값

에 적용됩니다. VARCHAR HASHED의 인덱스 SQL 데이터 유형은 해시 알고리즘을 결과 xs:string 값에 적용하여 인덱스에 삽입되는 해시 코드를 생성합니다.

DOUBLE, DATE 및 TIMESTAMP 데이터 유형을 사용한 인덱스의 경우 XQuery 캐스트 표현식을 사용하여 값을 인덱스 XML 데이터 유형으로 변환합니다.

인덱스가 고유한 경우, 값을 인덱스로 변환해도 값의 고유성이 적용됩니다.

data-type

다음의 데이터 유형이 지원됩니다.

sql-data-type

지원되는 SQL 데이터 유형입니다.

VARCHAR(*integer*)

VARCHAR의 해당 형식이 지정되면 DB2는 *정수*를 제한 조건으로 사용합니다. 인덱스될 문서 노드에 *정수*보다 긴 값이 있는 경우, 인덱스가 이미 존재하면 문서는 테이블에 삽입되지 않습니다. 인덱스가 존재하지 않으면 작성되지 않습니다. *정수*는 1과 페이지 *크기* 종속 최대값 사이의 값입니다. 표 17은 각 페이지 *크기*의 최대값을 나타냅니다.

표 17. 페이지 크기순 문서 노드의 최대 길이

페이지 *크기*	문서 노드의 최대 길이(바이트)
4KB	817
8KB	1841
16KB	3889
32KB	7985

문자열 비교에 XQuery 시맨틱이 사용되며, 이 경우에는 뒤 공백이 유효합니다. 이는 SQL 시맨틱과는 다르며 비교시 뒤 공백이 유효하지 않습니다.

VARCHAR HASHED

임의 길이 문자열을 인덱스하기 위해 VARCHAR HASHED를 지정하십시오. 인덱스된 문자열의 길이는 한계가 없습니다. DB2는 전체 문자열에 대해 8바이트 해시 코드를 생성합니다. 이 해시 문자열을 사용하는 인덱스는 동일성 검색용으로만 사용될 수 있습니다. 문자열 등식 비교에 XQuery 시맨틱이 사용되며, 이 경우에는 뒤 공백이 유효합니다. 이는 SQL 시맨틱과는 다르며 비교시 뒤 공백

이 유효하지 않습니다. 문자열에 대한 해시는 등식의 SQL 시맨틱이 아닌 XQuery 시맨틱을 보존합니다.

DOUBLE

DOUBLE 데이터 유형이 숫자 값을 인덱스하는 데 쓰이도록 지정합니다. 바인드되지 않은 10진수 유형 및 64비트 정수를 DOUBLE 값으로 저장할 경우 정밀도를 유실할 수 있습니다. SQL 데이터 유형 DOUBLE 자체는 특수 숫자 값인 *NaN*, *INF*, *-INF*, *+0* 및 *-0*을 지원하지 않으나 DOUBLE 값은 이러한 값을 포함할 수 있습니다.

DATE

XML 값을 인덱스화할 때 DATE 데이터 유형을 사용하도록 지정합니다. xs:date의 XML 스키마 데이터 유형은 SQL 데이터 유형에 해당되는 DB2 pureXML® xs:date 데이터 유형보다 더 큰 값의 범위를 허용합니다. 범위를 벗어난 값이 발생하면 오류가 리턴됩니다.

TIMESTAMP

XML 값을 인덱스화할 때 TIMESTAMP 데이터 유형을 사용하도록 지정합니다. xs:dateTime의 XML 스키마 데이터 유형은 SQL 데이터 유형에 해당되는 DB2 pureXML xs:dateTime 데이터 유형보다 더 큰 값의 범위와 소수 초 정밀도를 허용합니다. 범위를 벗어난 값이 발생하면 오류가 리턴됩니다.

IGNORE INVALID VALUES

목표 인덱스 XML 데이터 유형에 대해 유효하지 않은 XML 패턴 값은 무시되며 저장된 XML 문서에서의 해당 값이 CREATE INDEX 문에서 인덱스화되지 않는다는 것을 지정합니다. 디폴트로 유효하지 않은 값이 무시됩니다. 삽입 및 갱신 조작 시 유효하지 않은 XML 패턴 값이 인덱스화되지 않지만 XML 문서는 테이블에 삽입됩니다. 이 데이터 유형을 지정하는 것은 XML 패턴 값에 대한 제한조건으로 간주되지 않기 때문에 오류 또는 경고가 발생하지 않습니다(특정 XML 인덱스 데이터 유형을 검색하는 XQuery 표현식은 이 값을 고려하지 않음).

인덱스는 인덱스 XML 데이터 유형에 대해 유효하지 않은 XML 패턴 값만 무시할 수 있습니다. 올바른 값은 인덱스 XML 데이터 유형 값의 DB2 표현을 따라야 합니다. 그렇지 않으면 오류가 리턴됩니다. 인덱스 XML 데이터 유형 xs:string과 연관된 XML 패턴 값이 항상 유효합니다. 그러나 최대 길이를 초과하면 연관된 인덱스 SQL 데이터 유형 VARCHAR(*integer*) 데이터 유형의 추가 길이 제한조건은 오류를

발생할 수 있습니다. 오류가 리턴되면, 인덱스가 이미 존재하는 경우 XML 데이터가 테이블에 삽입되거나 갱신됩니다(SQLSTATE 23525). 인덱스가 존재하지 않으면 인덱스가 작성되지 않습니다(SQLSTATE 23526).

REJECT INVALID VALUES

모든 XML 패턴 값은 인덱스 XML 데이터 유형에 대해 유효해야 한다는 것을 지정합니다. XML 패턴 값을 인덱스 XML 데이터 유형으로 캐스트할 수 없으면 오류가 리턴됩니다. 인덱스가 이미 존재하는 경우 XML 데이터가 테이블에 삽입되거나 갱신되지 않습니다(SQLSTATE 23525). 인덱스가 존재하지 않으면 인덱스가 작성되지 않습니다 (SQLSTATE 23526).

CLUSTER

인덱스가 테이블의 클러스터링 인덱스가 되도록 지정합니다. 클러스터링 인덱스의 클러스터 요소는 데이터가 관련 테이블로 삽입됨에 따라 이 인덱스의 키 값이 동일한 범위 내에 있는 행에 근접하도록 새로운 행을 물리적으로 삽입하여 동적으로 유지보수되거나 향상됩니다. 테이블 하나에 대해 하나의 클러스터링 인덱스만 존재할 수 있으므로, CLUSTER가 테이블에 있는 기존 인덱스 정의에 사용된 경우 지정할 수 없습니다. 추가 모드를 사용하도록 정의된 테이블에는 클러스터 인덱스를 작성할 수 없습니다(SQLSTATE 428D8).

별명을 지정하거나 인덱스가 XML 데이터에 있는 인덱스인 경우에는 CLUSTER가 허용되지 않습니다 (SQLSTATE 42601). 이 절은 작성된 테이블 또는 선언된 임시 테이블(SQLSTATE 42995) 또는 범위 클러스터 테이블(SQLSTATE 429BG)에 사용할 수 없습니다.

EXTEND USING *index-extension-name*

이 인덱스를 관리하기 위해 사용되는 *index-extension*에 이름을 지정합니다. 이 절을 지정할 경우, 하나의 *column-name*만 지정해야 하고 그 컬럼은 구조화된 유형이나 구별 유형이어야 합니다(SQLSTATE 42997). *index-extension-name*은 카탈로그에 설명된 인덱스 확장에 이름을 지정해야 합니다(SQLSTATE 42704). 구별 유형의 경우, 컬럼은 인덱스 확장에 있는 해당 소스 키 매개변수의 유형과 일치해야 합니다. 구조화된 유형 컬럼의 경우, 해당 소스 키 매개변수 유형은 컬럼 유형과 같은 유형이거나 슈퍼 유형이어야 합니다(SQLSTATE 428E0).

이 절은 작성된 임시 테이블이나 선언된 임시 테이블에는 사용할 수 없습니다 (SQLSTATE 42995).

constant-expression

인덱스 확장에 대한 필수 인수 값을 식별합니다. 각 표현식은 길이나 정밀도, 스케일을 포함하여, 해당되는 인덱스 확장 매개변수의 정의된 데이터 유형과 정

확하게 일치하는 데이터 유형을 갖는 상수 값이어야 합니다(SQLSTATE 428E0). 데이터베이스 코드 페이지에서 이 절의 길이는 32,768바이트를 초과할 수 없습니다(SQLSTATE 22001).

PCTFREE *integer*

인덱스 작성시, 빈 스페이스로 남겨 둘 인덱스 페이지 비율을 지정합니다. 페이지의 첫 번째 행은 제한없이 추가됩니다. 인덱스 페이지에 추가 항목이 있는 경우, 적어도 *integer*%의 빈 스페이스가 각 페이지에 남게 됩니다. *integer* 값의 범위는 0에서 99까지입니다. 10보다 큰 값이 지정되는 경우, 비리프(NON-leaf) 페이지에는 10%의 빈 스페이스만 남게 됩니다. 디폴트값은 10입니다.

*nickname*을 지정할 경우, PCTFREE는 허용되지 않습니다(SQLSTATE 42601). 이 절은 작성된 임시 테이블이나 선언된 임시 테이블에는 사용할 수 없습니다(SQLSTATE 42995).

LEVEL2 PCTFREE *integer*

인덱스 레벨 2 작성시, 빈 스페이스로 남겨 둘 인덱스 페이지 비율을 지정합니다. *integer* 값의 범위는 0에서 99까지입니다. LEVEL2 PCTFREE가 설정되어 있지 않은 경우 여유 공간의 최소 10% 또는 PCTFREE%를 모든 비리프(non-leaf) 페이지에 남겨 둡니다. LEVEL2 PCTFREE가 설정되어 있는 경우 여유 공간의 *integer*%를 레벨 2 중간 페이지에 남겨 두며 여유 공간의 최소 10% 또는 *integer*%를 레벨 3 이상의 중간 페이지에 남겨 둡니다.

*nickname*을 지정할 경우, LEVEL2 PCTFREE는 허용되지 않습니다(SQLSTATE 42601). 이 절은 작성된 임시 테이블이나 선언된 임시 테이블에는 사용할 수 없습니다(SQLSTATE 42995).

MINPCTUSED *integer*

인덱스 리프 페이지가 온라인으로 병합되는지와 인덱스 리프 페이지에 사용되는 스페이스의 최소 비율에 대한 임계값을 나타냅니다. 인덱스 리프 페이지에서 키가 삭제되어 페이지에서 사용되는 스페이스의 비율이 *integer*% 이하인 경우, 이 페이지에 있는 나머지 키를 이웃하는 페이지의 키와 병합하려고 합니다. 이들 페이지 중 하나에 충분한 스페이스가 있으면, 병합이 수행되며 페이지들 중 하나가 삭제됩니다. *integer* 값은 0에서 99까지입니다. 성능상의 이유로 50 이하의 값이 권장됩니다. 이 옵션을 지정하면 갱신 및 삭제 조작을 수행할 때 성능이 저하됩니다. 독점 테이블 잠금이 있을 때만 갱신 및 삭제 조작 중 병합이 이루어집니다. 독점 테이블 잠금이 없으면 갱신 및 삭제 조작 동안 키가 허위로 삭제된 것으로 표시되고 병합이 이루어지지 않습니다. 이 때는 CREATE INDEX의 MINPCTUSED 옵션 대신 REORG INDEXES의 CLEANUP ONLY ALL 옵션을 사용하여 리프 페이지를 병합해 보십시오.

*nickname*을 지정할 경우, MINPCTUSED는 허용되지 않습니다(SQLSTATE 42601). 이 절은 작성된 임시 테이블이나 선언된 임시 테이블에는 사용할 수 없습니다(SQLSTATE 42995).

DISALLOW REVERSE SCANS

인덱스가 작성시에 정의된 순서로 스캔하는 것 또는 정방향 스캔만 지원하도록 지정합니다.

DISALLOW REVERSE SCANS는 *nickname*과 함께 지정될 수 없습니다 (SQLSTATE 42601).

ALLOW REVERSE SCANS

인덱스가 작성시 정의된 순서 및 그 반대 순서로 스캔하는 것 즉, 정방향 및 역방향 스캔을 모두 지원할 수 있도록 지정합니다.

ALLOW REVERSE SCANS는 *nickname*과 함께 지정될 수 없습니다(SQLSTATE 42601).

PAGE SPLIT

인덱스 분할 동작을 지정하며 디폴트값은 SYMMETRIC입니다.

SYMMETRIC

페이지가 중간쯤에서 분할되도록 지정합니다.

HIGH

인덱스 키 값이 특정 패턴을 따라 삽입되는 경우, 효율적으로 인덱스 페이지의 스페이스를 사용하는 인덱스 페이지 분할 동작을 지정합니다. 인덱스 키 값의 서브세트일 경우, 인덱스의 가장 왼쪽 컬럼에 동일한 값이 있어야 하고 인덱스의 가장 오른쪽 컬럼에 각 삽입시 증가되는 값이 있어야 합니다. 자세한 내용은 『CREATE INDEX문의 옵션』을 참조하십시오.

LOW

인덱스 키 값이 특정 패턴을 따라 삽입되는 경우, 효율적으로 인덱스 페이지의 스페이스를 사용하는 인덱스 페이지 분할 동작을 지정합니다. 인덱스 키 값의 서브세트일 경우, 인덱스의 가장 왼쪽 컬럼에 동일한 값이 있어야 하고 인덱스의 가장 오른쪽 컬럼에 각 삽입시 감소되는 값이 있어야 합니다. 자세한 내용은 『CREATE INDEX문의 옵션』을 참조하십시오.

COLLECT STATISTICS

인덱스를 작성하는 동안 기본 인덱스 통계가 수집되도록 지정합니다.

DETAILED

인덱스를 작성하는 동안 확장 인덱스 통계(CLUSTERFACTOR 및 PAGE_FETCH_PAIRS)도 함께 수집되도록 지정합니다.

SAMPLED

확장 인덱스 통계를 컴파일할 때 샘플링을 사용할 수 있도록 지정합니다.

COMPRESS

인덱스 압축의 사용 가능 여부를 지정합니다. 디폴트로, 데이터 행 압축이 사용 가능한 경우 인덱스 압축이 사용 가능하고, 데이터 행 압축이 사용 불가능한 경우 인

덱스 압축이 사용 불가능합니다. 이 옵션은 디폴트 동작 겹쳐쓰기에 사용할 수 있습니다. *nickname*을 지정할 경우, COMPRESS는 허용되지 않습니다(SQLSTATE 42601).

YES
인덱스 압축이 사용 가능함을 지정합니다. 인덱스에 대한 삽입 및 갱신 조작은 압축의 영향을 받습니다.

NO
인덱스 압축이 사용 불가능함을 지정합니다.

규칙

- CREATE INDEX문은 기존 인덱스와 일치하는 인덱스를 작성하려고 하는 경우에 실패합니다(SQLSTATE 01550).

  두 개의 인덱스가 일치하는 경우에는 몇 가지 인수를 사용하여 판별합니다. 두 개의 인덱스가 일치하는 경우 이 인수는 몇 개의 다른 방식을 통해 규칙으로 조합됩니다. 두 개의 인수가 일치하는 경우 다음과 같은 인수를 사용하여 판별합니다.

  1. 임의의 INCLUDE 컬럼을 포함하는 인덱스 컬럼 세트가 두 인덱스에서 동일합니다.
  2. 임의의 INCLUDE 컬럼을 포함하는 인덱스 키 컬럼 순서가 두 인덱스에서 동일합니다.
  3. 새 인덱스의 키 컬럼은 기존 인덱스의 키 컬럼과 동일하거나 수퍼 세트입니다.
  4. 컬럼의 순서 지정 속성이 두 인덱스에서 동일합니다.
  5. 기존 인덱스가 고유합니다.
  6. 두 인덱스가 고유하지 않습니다.

  이 인수의 다음과 같은 조합은 두 인덱스가 중복된 것으로 판별하는 규칙을 형식화합니다.

  – 1 + 2 + 4 + 5
  – 1 + 2 + 4 + 6
  – 1 + 2 + 3 + 5

  예외:

  – 비교된 인덱스 중 하나는 파티션되어 있고 다른 하나는 파티션되지 않은 경우 일치하는 다른 인덱스 조건이 충족된다고 해도 인덱스 이름이 다른 경우에는 중복된 것으로 간주되지 않습니다.
  – XML 데이터를 초과하는 인덱스의 경우 인덱스된 XML 컬럼, XML 패턴 및 데이터 유형(옵션 포함)이 동일하다 해도 인덱스 이름이 다르면, 인덱스 설명이 중복된 것으로 간주하지 않습니다.

- 시스템 유지보수 MQT의 고유 인덱스는 지원되지 않습니다(SQLSTATE 42809).
- COLLECT STATISTICS 옵션은 별칭이 지정된 경우에는 지원되지 않습니다 (SQLSTATE 42601).

주

- 인덱스가 작성되는 동안 테이블에 대한 동시 읽기/쓰기 액세스가 허용됩니다. 그렇지만 파티션되지 않은 테이블, 파티션되지 않은 인덱스 및 파티션된 인덱스의 인덱스에 대한 디폴트 인덱스 작성 동작은 서로 다릅니다.
  - 파티션되지 않은 테이블에 대한 인덱스의 경우, 인덱스를 빌드하면 인덱스를 작성할 때 테이블에 대해 수행한 변경사항이 새 인덱스에 맞게 전달됩니다. 그러면 인덱스 작성이 완료되는 동안 테이블에 대한 쓰기 액세스가 잠시 차단된 후 새 인덱스가 사용 가능해집니다.
  - 파티션되지 않은 인덱스의 경우, 인덱스를 빌드하면 인덱스를 작성할 때 테이블에 대해 수행한 변경사항이 새 인덱스에 맞게 전달됩니다. 그러면 인덱스 작성이 완료되는 동안 테이블에 대한 쓰기 액세스가 잠시 차단된 후 새 인덱스가 사용 가능해집니다.
  - 파티션된 인덱스의 경우, 인덱스 파티션을 빌드하면 해당 인덱스 파티션을 작성할 때 파티션에 대해 수행한 변경사항이 새 인덱스 파티션에 맞게 전달됩니다. 그러면 나머지 데이터 파티션에서 인덱스 작성이 완료되는 동안 데이터 파티션에 대한 쓰기 액세스가 차단됩니다. 마지막 데이터 파티션에 인덱스 파티션이 빌드되고 트랜잭션이 커미트되면 모든 데이터 파티션에서 읽기 및 쓰기가 가능해집니다.

  이 디폴트 동작을 사용하지 않으려면 CREATE INDEX문을 발행하기 전에 LOCK TABLE문을 사용하여 테이블을 명시적으로 잠그십시오. 읽기 액세스가 허용되는지 여부에 따라 테이블을 SHARE 모드나 EXCLUSIVE 모드로 잠글 수 있습니다.
- 이름이 지정된 테이블에 데이터가 이미 포함되어 있는 경우, CREATE INDEX는 이 데이터에 대한 인덱스 항목을 작성합니다. 테이블에 아직 데이터가 포함되어 있지 않으면, CREATE INDEX는 인덱스의 설명을 작성하고 인덱스 항목은 데이터가 그 테이블에 삽입될 때 작성됩니다.
- 일단 인덱스가 작성되어 데이터가 테이블에 로드되면, RUNSTATS 명령을 실행하는 것이 좋습니다. RUNSTATS 명령은 데이터베이스 테이블, 컬럼 및 인덱스에 대해 수집된 통계를 갱신합니다. 이 통계는 테이블에 대한 최적의 액세스 경로를 판별할 때 사용됩니다. RUNSTATS 명령을 실행하여, 데이터베이스 관리 프로그램이 새로운 인덱스의 등록 정보를 판별할 수 있습니다. CREATE INDEX문을 발행하기 전에 데이터가 로드된 경우에는 RUNSTATS 명령 대신 CREATE INDEX문에 COLLECT STATISTICS 옵션을 사용하는 것이 좋습니다.

- 아직 존재하지 않는 스키마 이름을 사용하여 인덱스를 작성하면, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 옵티마이저는 실제 인덱스 작성에 앞서 인덱스를 권장할 수 있습니다.
- 인덱스가 있는 데이터 소스에 대해 인덱스 스펙을 정의하는 경우, 인덱스 스펙의 이름이 인덱스의 이름과 일치하지 않아도 됩니다.
- 옵티마이저는 인덱스 스펙을 사용하여 스펙이 적용하는 데이터 소스 테이블에 대한 액세스를 향상시킵니다.
- *호환성*: z/OS용 DB2와의 호환성:
  - 다음 구문은 허용되기도 하고 무시되기도 합니다.
    - CLOSE
    - DEFINE
    - FREEPAGE
    - GBPCACHE
    - PIECESIZE
    - TYPE 2
    - 사용 중인 블록
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - COPY NO
    - DEFER NO

예:

예 1: PROJECT 테이블에 UNIQUE_NAM이라는 인덱스를 작성하십시오. 인덱스의 목적은 프로젝트 이름(PROJNAME)에 대해 같은 값을 갖고 있는 항목이 테이블에 두 개가 되지 않도록 하는 것입니다. 인덱스 항목은 오름차순으로 정렬됩니다.

```
CREATE UNIQUE INDEX UNIQUE_NAM
   ON PROJECT(PROJNAME)
```

예 2: EMPLOYEE 테이블에 JOB_BY_DPT라는 인덱스를 작성하십시오. 각 부서(WORKDEPT) 내에서 인덱스 항목을 직책(JOB)별로 오름차순으로 배열합니다.

```
CREATE INDEX JOB_BY_DPT
   ON EMPLOYEE (WORKDEPT, JOB)
```

예 3: 별칭 EMPLOYEE는 CURRENT_EMP라는 데이터 소스 테이블을 참조합니다. 이 별칭이 작성된 후 CURRENT_EMP에 인덱스가 정의되었습니다. 인덱스 키에 대해 선택된 컬럼은 WORKDEBT와 JOB이었습니다. 이 인덱스를 기술하는 인덱스 스펙을

작성하십시오. 이 스펙을 통해, 옵티마이저는 인덱스가 존재하는지 여부와 인덱스 키가 무엇인지를 알 수 있습니다. 이 정보로 옵티마이저는 테이블 액세스 전략을 향상시킬 수 있습니다.

```
CREATE UNIQUE INDEX JOB_BY_DEPT
  ON EMPLOYEE (WORKDEPT, JOB)
  SPECIFICATION ONLY
```

예 4: 구조화된 유형 컬럼 위치에 SPATIAL_INDEX라는 확장된 인덱스 유형을 작성하십시오. 인덱스 확장 GRID_EXTENSION에 있는 설명은 PATIAL_INDEX의 유지보수에 사용됩니다. 이 리터럴은 인덱스 격자 크기를 작성하기 위해 GRID_EXTENSION에 제공됩니다.

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)
  EXTEND USING (GRID_EXTENSION (x'000100100010001000400010'))
```

예 5: TAB1 테이블에 IDX1 인덱스를 작성하고 IDX1 인덱스에 대한 기본 인덱스 통계를 수집하십시오.

```
CREATE INDEX IDX1 ON TAB1 (col1) COLLECT STATISTICS
```

예 6: TAB1 테이블에 IDX2 인덱스를 작성하고 IDX2 인덱스에 대한 세부 인덱스 통계를 수집하십시오.

```
CREATE INDEX IDX2 ON TAB1 (col2) COLLECT DETAILED STATISTICS
```

예 7: TAB1 테이블에 IDX3 인덱스를 작성하고 샘플링을 사용하여 IDX3 인덱스에 대한 세부 인덱스 통계를 수집하십시오.

```
CREATE INDEX IDX3 ON TAB1 (col3) COLLECT SAMPLED DETAILED STATISTICS
```

예 8: 테이블 스페이스 IDX_TBSP의 MYNUMBERDATA라는 이름의 파티션된 테이블에 있는 A_IDX라는 이름의 고유 인덱스를 작성하십시오.

```
CREATE UNIQUE INDEX A_IDX ON MYNUMBERDATA (A) IN IDX_TBSP
```

예 9: 테이블 스페이스 IDX_TBSP의 MYNUMBERDATA라는 이름의 파티션된 테이블에 있는 B_IDX라는 이름의 비고유 인덱스를 작성하십시오.

```
CREATE INDEX B_IDX ON MYNUMBERDATA (B)
  NOT PARTITIONED IN IDX_TBSP
```

예 10: COMPANYDOCS라는 이름의 XML 컬럼이 있는 COMPANYINFO라는 이름의 테이블에 XML 데이터에 있는 인덱스를 작성하십시오. XML 컬럼 COMPANYDOCS는 다음과 유사한 XML 문서를 많이 포함하고 있습니다.

```
<company name="Company1">
    <emp id="31201" salary="60000" gender="Female">
        <name>
            <first>Laura</first>
            <last>Brown</last>
        </name>
        <dept id="M25">
```

```
            Finance
        </dept>
    </emp>
</company>
```

COMPANYINFO 테이블의 사용자는 직원 ID를 사용하여 직원 정보를 자주 검색할 필요가 있습니다. 다음과 같은 인덱스를 사용하면 해당 검색을 보다 효율적으로 할 수 있습니다.

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
  GENERATE KEY USING XMLPATTERN '/company/emp/@id'
    AS SQL DOUBLE
```

예 11: 다음 인덱스는 약어로 표시되지 않은 구문을 사용하는 경우를 제외하고 이전 예제에서 작성한 인덱스와 논리적으로 동일합니다.

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
  GENERATE KEY USING XMLPATTERN '/child::company/child::emp/attribute::id'
    AS SQL DOUBLE
```

예 12: 책 제목만을 VARCHAR(100)로 인덱스하여 DOC이라는 이름의 컬럼에 인덱스를 작성하십시오. 해당 책 제목이 모든 책에서 고유해야 하기 때문에 인덱스는 고유해야 합니다.

```
CREATE UNIQUE INDEX MYDOCSIDX ON MYDOCS(DOC)
  GENERATE KEY USING XMLPATTERN '/book/title'
    AS SQL VARCHAR(100)
```

예 13: 장 번호를 DOUBLE로 인덱스하여 DOC이라는 이름의 컬럼에 인덱스를 작성하십시오. 이 예는 이름 스페이스 선언을 포함합니다.

```
CREATE INDEX MYDOCSIDX ON MYDOCS(DOC)
  GENERATE KEY USING XMLPATTERN
           'declare namespace b="http://www.foobar.com/book/";
               declare namespace c="http://acme.org/chapters";
         /b:book/c:chapter/@number'
    AS SQL DOUBLE
```

CREATE INDEX EXTENSION

CREATE INDEX EXTENSION문은 구조화된 유형이나 구별 유형 컬럼을 가지고 있는 테이블에서 인덱스와 함께 사용하기 위해 확장 오브젝트를 정의합니다.

호출

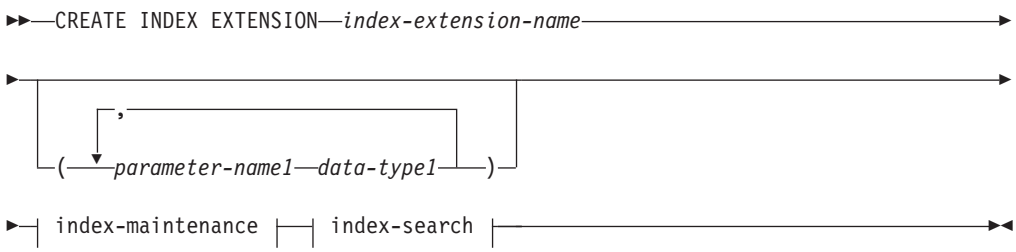
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

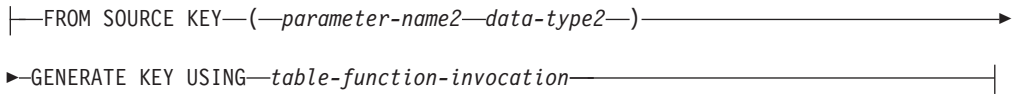
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 인덱스 확장의 스키마 이름이 기존 스키마를 참조하지 않는 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 인덱스 확장의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

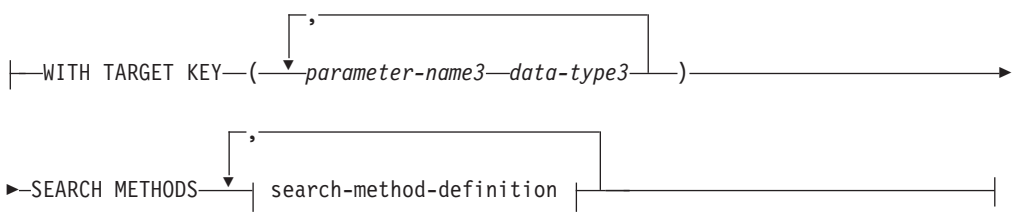
구문



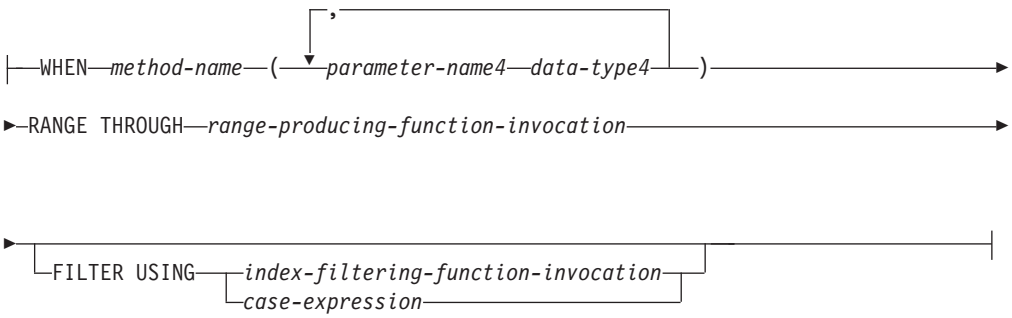
index-maintenance:



index-search:



search-method-definition:



설명

index-extension-name

인덱스 확장에 이름을 지정합니다. 내재적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 설명된 인덱스 확장을 식별해서는 안됩니다. 두 부분으로 구성된 *index-extension-name*을 지정하면, 스키마 이름은 'SYS'로 시작할 수 없습니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42939).

parameter-name1

인덱스 확장의 실제 동작을 정의하기 위해 CREATE INDEX를 실행할 때 인덱스 확장에 전달되는 매개변수를 식별합니다. 인덱스 확장에 전달되는 매개변수의 값이 인덱스 확장의 새 인스턴스를 정의하기 때문에 이러한 매개변수를 *인스턴스 매개변수*라고 합니다.

*parameter-name1*은 인덱스 확장의 정의 내에서 고유해야 합니다. 매개변수 수는 90개를 초과할 수 없습니다. 이 한계를 초과하면, 오류가 발생합니다(SQLSTATE 54023).

data-type1

각 매개변수의 데이터 유형을 지정합니다. 인덱스 확장에서 받기를 기대하는 각 매개변수에 대해 목록에서 한 항목만 지정해야 합니다. 지정할 수 있는 유일한 SQL 데이터 유형은 VARCHAR, INTEGER, DECIMAL, DOUBLE 또는 VARGRAPHIC과 같이, 상수로 사용할 수 있는 데이터 유형입니다(SQLSTATE 429B5). 10진수 부동 소수점 데이터 유형을 지정할 수 없습니다(SQLSTATE 429B5). CREATE INDEX를 실행할 때 인덱스 확장에서 받는 매개변수 값은 길이, 정밀도 및 스케일을 포함하여, 정확하게 *data-type1*과 일치해야 합니다(SQLSTATE 428E0).

index-maintenance

구조화된 또는 구별 유형 컬럼의 인덱스 키가 유지보수되는지 방식을 지정합니다. 인덱스 유지보수는 소스 컬럼을 목표 키로 변환하는 프로세스입니다. 변환 프로세스는 이전에 데이터베이스에서 정의된 테이블 함수를 사용하여 정의됩니다.

FROM SOURCE KEY (*parameter-name2 data-type2*)

인덱스 확장에서 지원되는 소스 키 컬럼에 대한 구조화된 유형 또는 구별 유형을 지정합니다.

parameter-name2

소스 키 컬럼과 연관되는 매개변수를 식별합니다. 소스 키 컬럼은 데이터 유형이 *data-type2*와 같은 인덱스 키 컬럼(CREATE INDEX문에 정의됨)입니다.

data-type2

*parameter-name2*의 데이터 유형을 정의하고, *data-type2*는 LOB, XML 또는 DECFLOAT에서 전래되지 않은 사용자 정의 구조화된 유형이나 구별 유형이어야 합니다(SQLSTATE 42997). 인덱스 확장이 CREATE INDEX를 실행할 때 인덱스와 연관될 경우, 인덱스 키 컬럼의 데이터 유형은 다음과 같아야 합니다.

- 구별 유형일 경우 정확하게 *data-type2*와 일치하거나
- 구조화된 유형일 경우 *data-type2*와 같은 유형이거나 부속 유형이어야 함

그렇지 않으면 오류가 발생합니다(SQLSTATE 428E0).

GENERATE KEY USING *table-function-invocation*

사용자 정의 테이블 함수를 사용하여 인덱스 키가 생성되는 방법을 지정합니다. 단일 소스 키 데이터 값에 대해 여러 개의 인덱스 항목이 생성될 수도 있습니다. 인덱스 항목은 단일 소스 키 데이터 값에서 중복될 수 없습니다(SQLSTATE 22526). 함수는 *parameter-name1*, *parameter-name2* 또는 상수를 인수로 사용할 수 있습니다. *parameter-name2*의 데이터 유형이 구조화된 데이터 유형일 경우, 그 구조화된 유형의 observer 메소드만 해당되는 인수에서 사용될 수 있습니다(SQLSTATE 428E3). GENERATE KEY 함수의 출력은 TARGET KEY 스펙에서 지정해야 합니다. 함수 출력은 FILTER USING절에 지정된 인덱스 필터링 함수에 대한 입력으로 사용할 수도 있습니다.

*테이블 함수 호출*에서 사용되는 함수는 다음과 같아야 합니다.

- 테이블 함수로 분석되어야 합니다(SQLSTATE 428E4).
- 이 데이터베이스가 유니코드 데이터베이스가 아니면 PARAMETER CCSID UNICODE로 정의하지 마십시오(SQLSTATE 428E4).
- LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 428E4).
- NOT DETERMINISTIC(SQLSTATE 428E4) 또는 EXTERNAL ACTION(SQLSTATE 428E4)으로 정의하지 않아야 합니다.
- NO SQL로 정의되어 있습니다(SQLSTATE 428E4).
- 시스템에서 생성하는 관찰자 메소드를 제외하고, 매개변수의 데이터 유형으로 구조화된 데이터 유형, LOB 또는 XML이면 안됩니다(SQLSTATE 428E3).

- 서브쿼리를 포함하고 있지 않아야 합니다(SQLSTATE 428E3).
- XMLQUERY 또는 XMLEXISTS 표현식을 포함하고 있지 않아야 합니다 (SQLSTATE 428E3).
- EXTEND USING절 없이 정의된 인덱스 컬럼의 데이터 유형에 대한 제한사항을 따르는 데이터 유형의 컬럼을 리턴해야 합니다.

인수가 다른 조작이나 루틴을 호출할 경우, 이 인수는 observer 메소드여야 합니다(SQLSTATE 428E3).

인덱스 확장자 정의자에게는 이 함수에 대한 EXECUTE 특권이 있어야 합니다.

index-search

검색 인수의 맵핑을 검색 범위에 제공하여 검색이 수행되는 방식을 지정합니다.

WITH TARGET KEY

GENERATE KEY USING절에 지정된 키 생성 함수의 출력인 목표 키 매개변수를 지정합니다.

parameter-name3

주어진 목표 키와 연관되는 매개변수를 지정합니다. *parameter-name3*는 GENERATE KEY USING절의 테이블 함수에 지정된 RETURNS 테이블의 컬럼에 해당됩니다. 지정된 매개변수 수는 테이블 함수에서 리턴되는 컬럼 수와 일치해야 합니다(SQLSTATE 428E2).

data-type3

해당되는 각 *parameter-name3*에 대한 데이터 유형을 지정합니다. *data-type3*는 길이, 정밀도 및 유형을 포함하여 GENERATE KEY USING절의 테이블 함수에 지정된 대로 RETURNS 테이블의 각 해당 출력 데이터 유형과 정확히 일치해야 합니다(SQLSTATE 428E2).

SEARCH METHODS

인덱스에 대해 정의된 검색 메소드를 사용합니다.

search-method-definition

인덱스 검색에 대한 메소드 세부사항을 지정하며, 메소드 이름, 검색 인수, 범위 생성 함수, 선택적 인덱스 필터 함수로 구성됩니다.

WHEN *method-name*

검색 메소드의 이름으로, 인덱스 노출 규칙(사용자 정의 함수의 PREDICATES 절에 있음)에 지정된 메소드 이름과 관련된 SQL ID입니다. *search-method-name*은 검색 메소드 정의에서 단 하나의 WHEN절에서만 참조될 수 있습니다(SQLSTATE 42713).

parameter-name4

검색 인수의 매개변수를 식별합니다. 이 이름들은 RANGE THROUGH 및 FILTER USING절에서 사용됩니다.

data-type4

검색 매개변수와 연관된 데이터 유형입니다.

RANGE THROUGH *range-producing-function-invocation*

검색 범위를 생성하는 외부 테이블 함수를 지정합니다. 이 함수는 인수로 *parameter-name1*, *parameter-name4* 또는 상수를 사용하여 검색 범위 세트를 리턴합니다.

*range-producing-function-invocation*에 사용되는 테이블 함수는 다음과 같아야 합니다.

- 테이블 함수로 분석되어야 합니다(SQLSTATE 428E4).
- 서브쿼리(SQLSTATE 428E3) 또는 SQL 함수(SQLSTATE 428E4)를 포함하고 있지 않아야 합니다.
- 인수에 XMLQUERY 또는 XMLEXISTS 표현식을 포함하고 있지 않아야 합니다(SQLSTATE 428E3).
- 이 데이터베이스가 유니코드 데이터베이스가 아니면 PARAMETER CCSID UNICODE로 정의하지 마십시오(SQLSTATE 428E4).
- LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 428E4).
- NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의하지 않아야 합니다(SQLSTATE 428E4).
- NO SQL로 정의되어 있습니다(SQLSTATE 428E4).

이 함수 결과의 번호와 유형은 다음에 의해 GENERATE KEY USING절에 지정된 테이블 함수의 결과와 관련되어야 합니다(SQLSTATE 428E1).

- 키 변환 함수가 리턴하는 컬럼 수의 두 배까지 리턴합니다.
- 짝수 개수의 컬럼이 있어 리턴 컬럼의 처음 반은 범위의 시작(시작 키 값)을 정의하고 리턴 컬럼의 두 번째 반은 범위의 끝(중지 키 값)을 정의합니다.
- 해당되는 중지 키 컬럼과 동일한 유형의 각 시작 키 컬럼이 있습니다.
- 각 시작 키 컬럼의 유형을 해당되는 키 변환 함수 컬럼과 동일하게 합니다.

더 명확하게, $_1$:t_1, ..., a_n:t_n을 함수 결과 컬럼과 키 변환 함수의 데이터 유형이라고 합니다. *range-producing-function-invocation*의 함수 결과 컬럼은 b_1:t_1, ..., b_m:t_m, c_1:t_1, ..., c_m:t_m여야 하며 여기서 m <= n 및 "b" 컬럼은 시작 키 컬럼이며 "c" 컬럼은 중지 키 컬럼입니다.

*range-producing-function-invocation*이 시작 및 중지 키 값으로 널(NULL) 값을 리턴할 경우, 시맨틱은 정의되지 않습니다.

인덱스 확장자 정의자에게는 이 함수에 대한 EXECUTE 특권이 있어야 합니다.

FILTER USING

외부 함수나 CASE 표현식의 스펙이 범위 생성 함수 적용 후 리턴된 인덱스 항목을 필터링하는 데 사용되도록 합니다.

index-filtering-function-invocation

인덱스 항목 필터링에 사용될 외부 함수를 지정합니다. 이 함수는 *parameter-name1*, *parameter-name3*, *parameter-name4* 또는 상수를 인수로 사용하고(SQLSTATE 42703), 정수를 리턴합니다(SQLSTATE 428E4). 리턴된 값이 1일 경우, 인덱스 항목에 해당되는 행은 테이블에서 검색됩니다. 그렇지 않으면, 인덱스 항목은 추가 처리에 고려되지 않습니다.

지정하지 않으면, 인덱스 필터링이 수행되지 않습니다.

*index-filtering-function-invocation*에 사용되는 함수는 다음과 같아야 합니다.

- 이 데이터베이스가 유니코드 데이터베이스가 아니면 PARAMETER CCSID UNICODE로 정의하지 마십시오(SQLSTATE 428E4).
- LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 429B4).
- NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의하지 않아야 합니다(SQLSTATE 42845).
- NO SQL로 정의되어 있습니다(SQLSTATE 428E4).
- 매개변수 중 어느 하나의 데이터 유형으로 구조화된 데이터 유형을 가지고 있지 않아야 합니다(SQLSTATE 428E3).
- 서브쿼리를 포함하고 있지 않아야 합니다(SQLSTATE 428E3).
- XMLQUERY 또는 XMLEXISTS 표현식을 포함하고 있지 않아야 합니다(SQLSTATE 428E3).

인수가 다른 함수나 메소드를 호출할 경우, 해당 중첩 함수나 메소드에 대해서도 이 규칙이 시행됩니다. 그러나 시스템에서 생성되는 관찰자 메소드는 인수가 내장 데이터 유형을 야기하면 필터 함수(또는 인수로 사용되는 함수나 메소드)에 대한 인수로 허용됩니다.

인덱스 확장자 정의자에게는 이 함수에 대한 EXECUTE 특권이 있어야 합니다.

case-expression

인덱스 항목 필터링에 대한 CASE 표현식을 지정합니다. *parameter-name1*, *parameter-name3*, *parameter-name4* 또는 상수(SQLSTATE 42703)를 *searched-when-clause* 및 *simple-when-clause*에 사용할 수 있습니다. FILTER USING *index-filtering-function-invocation*에 규칙이 지정된 외부 함수를 *result-expression*으로 사용할 수 있습니다. *case-expression*에서 참조되는 함

수도 *index-filtering-function-invocation*에 나열된 규칙을 따라야 합니다. 그리고 서브쿼리 및 XMLQUERY 또는 XMLEXISTS 표현식은 *case-expression*의 다른 곳에서는 사용할 수 없습니다(SQLSTATE 428E4). case 표현식은 정수를 리턴해야 합니다(SQLSTATE 428E4). *result-expression*에서 리턴 값 1은 인덱스 항목이 보존됨을 의미하고, 1 이외의 값의 경우 인덱스 항목은 버려집니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 인덱스 확장자를 작성하면, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 부여됩니다.

예:

예 1: 다음은 *gridEntry*라는 테이블 함수에서 구조화된 SHAPE 컬럼을 사용하여 7개의 인덱스 목표 키를 생성하는 *grid_extension*이라는 인덱스 표현식을 작성합니다. 이 인덱스 확장은 또한 검색 인수가 제공될 때 검색 범위를 생성하기 위해 두 개의 인덱스 검색 메소드를 제공합니다.

```
CREATE INDEX EXTENSION GRID_EXTENSION (LEVELS VARCHAR(20) FOR BIT DATA)
  FROM SOURCE KEY (SHAPECOL SHAPE)
  GENERATE KEY USING GRIDENTRY(SHAPECOL..MBR..XMIN,
                                                          SHAPECOL..MBR..YMIN,
                                                          SHAPECOL..MBR..XMAX,
                                                          SHAPECOL..MBR..YMAX,
                                                         LEVELS)
  WITH TARGET KEY (LEVEL INT, GX INT, GY INT,
                                    XMIN INT, YMIN INT, XMAX INT, YMAX INT)
  SEARCH METHODS
  WHEN SEARCHFIRSTBYSECOND (SEARCHARG SHAPE)
  RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                                                           SEARCHARG..MBR..YMIN,
                                                           SEARCHARG..MBR..XMAX,
                                                           SEARCHARG..MBR..YMAX,
                                                           LEVELS)
  FILTER USING
    CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
       SEARCHARG..MBR..YMAX < YMIN) THEN 0
    ELSE CHECKDUPLICATE(LEVEL, GX, GY,
                                                         XMIN, YMIN, XMAX, YMAX,
                                                          SEARCHARG..MBR..XMIN,
                                                         SEARCHARG..MBR..YMIN,
                                                         SEARCHARG..MBR..XMAX,
                                                         SEARCHARG..MBR..YMAX,
                                                         LEVELS)
    END
  WHEN SEARCHSECONDBYFIRST (SEARCHARG SHAPE)
  RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                                                           SEARCHARG..MBR..YMIN,
                                                           SEARCHARG..MBR..XMAX,
                                                           SEARCHARG..MBR..YMAX,
                                                           LEVELS)
  FILTER USING
    CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
       SEARCHARG..MBR..YMAX < YMIN) THEN 0
    ELSE MBROVERLAP(XMIN, YMIN, XMAX, YMAX,
                                                            SEARCHARG..MBR..XMIN,
```

```
                                                SEARCHARG..MBR..YMIN,
                                                SEARCHARG..MBR..XMAX,
                                                 SEARCHARG..MBR..YMAX)
    END
```

CREATE METHOD

CREATE METHOD문은 이미 사용자가 정의하는 구조화된 유형 정의의 일부인 메소드 스펙과 메소드 본문을 연관시키는 데 사용됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- CREATE METHOD문에서 참조된 구조화된 유형의 스키마에 대한 CREATEIN 특권
- CREATE METHOD문에서 참조된 구조화된 유형의 소유자
- DBADM 권한

외부 메소드 본문을 메소드 스펙과 연관시키려면 명령문의 권한 부여 ID가 보유한 특권에 다음 중 적어도 하나가 포함되어야 합니다.

- 데이터베이스에 있는 CREATE_EXTERNAL_ROUTINE 권한
- DBADM 권한

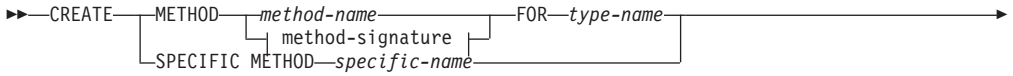
SQL 메소드를 작성할 때 명령문의 권한 부여 ID에서 보유한 특권은 모든 fullselect에서 식별되는 각 테이블, 뷰 또는 별칭에 대해 다음 특권 중 적어도 하나의 특권을 가지고 있어야 합니다.

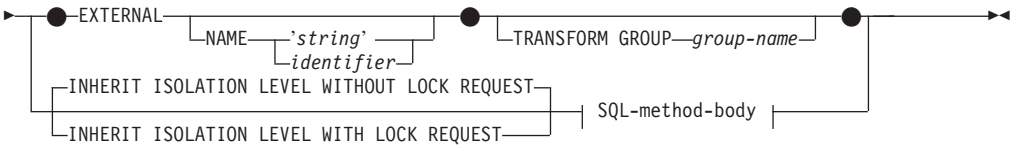
- 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- 테이블, 뷰 또는 별칭에 대한 SELECT 특권
- DATAACCESS 권한

CREATE METHOD문에 지정된 테이블이나 뷰에 대해서는 PUBLIC 이외의 그룹 특권이 고려되지 않습니다.

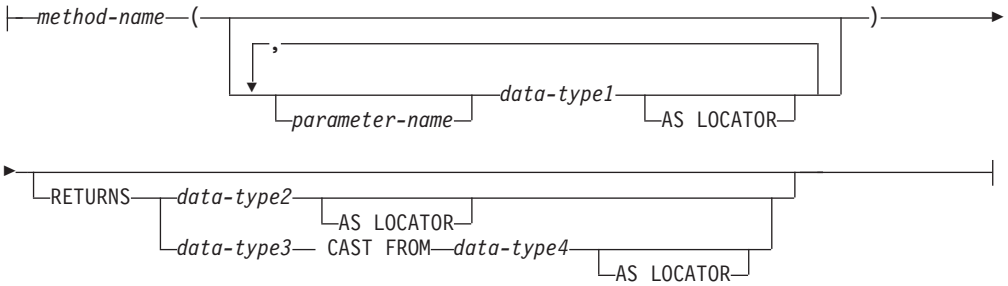
별칭으로 참조된 테이블이나 뷰에 대한 데이터 소스의 권한 부여 요건은 메소드가 호출될 때 적용됩니다. 연결의 권한 부여 ID는 서로 다른 리모트 권한 부여 ID로 맵핑될 수 있습니다.

구문

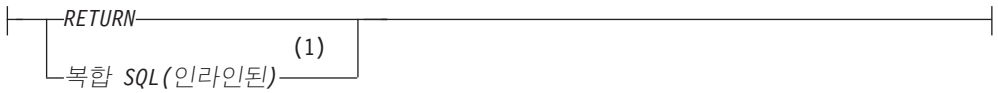




method-signature:



SQL-method-body:



주:

1 복합 SQL(인라인)문은 파티션되지 않은 데이터베이스에서 SQL 메소드 정의의 SQL-method-body에 대해서만 지원됩니다.

설명

METHOD

사용자가 정의하는 구조화된 유형과 연관되는 기존의 메소드 스펙을 식별합니다. 메소드 스펙은 다음 수단 중 한 가지 방법으로 식별될 수 있습니다.

method-name

메소드 본문이 정의되는 메소드 스펙에 이름을 지정하십시오. 내재적 스키마는 주제 유형(*type-name*)의 스키마입니다. 이 *method-name*을 가지고 있는 *type-name*에 대해 하나의 메소드 스펙만 있어야 합니다(SQLSTATE 42725).

method-signature

정의할 메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 시그니처는 CREATE TYPE 또는 ALTER TYPE 명령문에 제공된 메소드 스펙과 일치해야 합니다(SQLSTATE 42883).

method-name

메소드 본문이 정의되는 메소드 스펙에 이름을 지정합니다. 내재적 스키마는 주제 유형(*type-name*)의 스키마입니다.

parameter-name

매개변수 이름을 식별하십시오. 메소드 시그니처에서 매개변수 이름이 제공되면, 그 이름은 일치하는 메소드 스펙의 해당되는 부분과 정확하게 같아야 합니다. 매개변수 이름은 문서화를 위해서만 이 명령문에서 지원됩니다.

data-type1

각 매개변수의 데이터 유형을 지정합니다. 배열 유형은 지원되지 않습니다(SQLSTATE 42815).

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다.

RETURNS

이 절은 메소드의 출력을 식별합니다. 메소드 시그니처에서 RETURNS 절이 제공되면, 이는 CREATE TYPE에서 일치하는 메소드 스펙의 해당되는 부분과 정확하게 같아야 합니다. RETURNS절은 문서화를 위해서만 이 명령문에서 지원됩니다.

data-type2

출력의 데이터 유형을 지정합니다. 배열 유형은 지원되지 않습니다(SQLSTATE 42815).

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값 대신 LOB 로케이터가 메소드에 의해 리턴됨을 나타냅니다.

data-type3 **CAST FROM** *data-type4*

이 형태의 RETURNS절은 함수 코드에 의해 리턴된 데이터 유형에서 호출 명령문으로 서로 다른 데이터 유형을 리턴하는 데 사용됩니다.

AS LOCATOR

LOB 유형이나 LOB 유형을 기초로 하는 구별 유형의 경우, AS LOCATOR절은 실제 값 대신 메소드로부터 LOB 로케이터가 리턴됨을 나타내기 위해 사용할 수 있습니다.

FOR *type-name*

지정된 메소드가 연관될 유형의 이름을 지정하십시오. 이 이름은 카탈로그에 이미 설명된 유형을 식별해야 합니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

SPECIFIC METHOD *specific-name*

CREATE TYPE시에 지정하였거나 디폴트로 설정된 특정 이름을 사용하여 특정 메소드를 식별합니다. 특정 이름은 이름 지정된 스키마 또는 내재된 스키마에서 메소드 스펙을 식별해야 합니다. 그렇지 않으면, 오류가 발생합니다(SQLSTATE 42704).

EXTERNAL

이 절은 CREATE METHOD문이 외부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 링크 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됨을 나타냅니다. CREATE TYPE에서 일치하는 메소드 스펙은 SQL 이외의 LANGUAGE를 지정해야 합니다. 메소드가 호출될 경우, 메소드의 주제는 내재된 첫 번째 매개변수로서 구현에 전달됩니다.

NAME절이 지정되지 않았을 경우, "NAME *method-name*"이 사용됩니다.

NAME

이 절은 정의되는 메소드를 구현하는 사용자 작성 코드의 이름을 식별합니다.

'*string***'**

'string' 옵션은 최대 길이가 254바이트인 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다. 특정 언어 규칙에 대한 정보는 『CREATE FUNCTION(외부 스칼라)문』을 참조하십시오.

identifier

지원된 이 ID는 SQL ID입니다. SQL ID는 문자열에서 library-id로 사용됩니다. 분리 ID가 아닐 경우, ID는 대문자로 겹쳐집니다. ID에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 NAME 양식은 LANGUAGE C에서만 사용할 수 있습니다(CREATE TYPE의 메소드 스펙에서 정의된 대로).

TRANSFORM GROUP *group-name*

메소드를 호출할 때 사용자가 정의하는 구조화된 유형 변환에 대해 사용되는 변환 그룹을 나타냅니다. 변환은 메소드 정의에 사용자가 정의하는 구조화된 유형이 포함되므로 반드시 필요합니다.

변환 그룹 이름을 지정하는 것이 아주 좋습니다. 이 절을 지정하지 않을 경우, 사용되는 디폴트 그룹 이름은 DB2_FUNCTION입니다. 지정된 (또는 디폴트값인) 그룹 이름이 참조된 구조화된 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42741). 마찬가지로, 필수 FROM SQL 또는 TO SQL 변환 함수가 제공된 그룹 이름과 구조화된 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42744).

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST 또는 INHERIT ISOLATION LEVEL WITH LOCK REQUEST

메소드가 메소드를 호출하는 명령문의 분리 레벨을 상속할 경우, 잠금 요청이 명령

문의 분리 절과 연관될 수 있는지의 여부를 지정합니다. 디폴트값은 INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST입니다.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST
메소드가 호출하는 명령문의 분리 레벨을 상속하므로 잠금 요청 절을 지정된 분리 절의 일부로 포함하는 SQL문의 컨텍스트에서 메소드를 호출할 수 없음을 지정합니다(SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST
메소드가 호출하는 명령문의 분리 레벨을 상속하므로 지정된 잠금 요청 절도 상속함을 지정합니다.

SQL-method-body
SQL-method-body는 CREATE TYPE의 메소드 스펙이 LANGUAGE SQL인 경우 메소드가 구현되는 방법을 정의합니다.

SQL-method-body는 메소드 스펙의 다음 부분을 따라야 합니다.

- DETERMINISTIC 또는 NOT DETERMINISTIC (SQLSTATE 428C2)
- EXTERNAL ACTION 또는 NO EXTERNAL ACTION (SQLSTATE 428C2)
- CONTAINS SQL 또는 READS SQL DATA (SQLSTATE 42985)

매개변수 이름은 SQL-method-body에서 참조될 수 있습니다. 메소드의 주제는 내재된 SELF라는 첫 번째 매개변수로 메소드 구현에 전달됩니다.

더 자세한 내용은 ″복합 SQL(인라인)문″ 및 ″RETURN문″을 참조하십시오.

규칙

- 메소드 스펙은 CREATE METHOD를 사용하기 전에 CREATE TYPE이나 ALTER TYPE문을 사용하여 미리 정의되어 있어야 합니다(SQLSTATE 42723).
- 작성되는 메소드가 겹쳐쓰는 메소드일 경우에는 다음 메소드에 종속된 패키지가 무효화됩니다.
  - 원래 메소드
  - 메소드의 주제가 작성되는 메소드의 슈퍼 유형인 기타 겹쳐쓰는 메소드
- XML 데이터 유형은 메소드에서 쓰일 수 없습니다.

주

- 메소드에 SQL을 사용할 수 있는 경우, 외부 프로그램에서 임의의 페더레이티드 오브젝트에 액세스해서는 안됩니다(SQLSTATE 55047).
- *특권*: 메소드의 정의자는 항상 메소드 삭제 권한뿐 아니라 메소드의 EXECUTE 특권을 수신합니다.

  EXTERNAL 메소드를 작성할 경우 메소드 정의자는 항상 EXECUTE 특권 WITH GRANT OPTION을 받습니다.

SQL 메소드를 작성할 경우, 정의자는 메소드 정의에 필요한 모든 특권으로 WITH GRANT OPTION이 있거나 정의자에게 SYSADM이나 DBADM 권한이 있으면 메소드에 대한 EXECUTE 특권 WITH GRANT OPTION을 가집니다. SQL 메소드의 정의자는 메소드 작성시 파생되는 특권들이 존재할 경우 특권들을 획득할 수 있습니다. 정의자는 PUBLIC이 특권을 갖기 때문에 또는 직접 특권을 가져야 합니다. 메소드 정의자가 구성원인 그룹에서 보유하는 특권은 고려되지 않습니다. 메소드를 사용할 때는 연결된 사용자의 권한 부여 ID에 별칭이 데이터 소스에서 참조되는 테이블이나 뷰에 대한 유효한 특권이 있어야 합니다.

- *테이블 액세스 제한사항*: 메소드가 READS SQL DATA로 정의되어 있으면 메소드의 명령문은 메소드를 호출한 명령문이 수정하고 있는 테이블에 액세스할 수 없습니다(SQLSTATE 57053).

예:

예 1:

```
CREATE METHOD BONUS (RATE DOUBLE)
  FOR EMP
  RETURN SELF..SALARY * RATE
```

예 2:

```
CREATE METHOD SAMEZIP (addr address_t)
  RETURNS INTEGER
  FOR address_t
  RETURN
    (CASE
      WHEN (self..zip = addr..zip)
        THEN 1
      ELSE 0
    END)
```

예 3:

```
CREATE METHOD DISTANCE (address_t)
  FOR address_t
  EXTERNAL NAME 'addresslib!distance'
  TRANSFORM GROUP func_group
```

CREATE MODULE

CREATE MODULE문은 응용프로그램 서버(AS)에 모듈을 작성합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

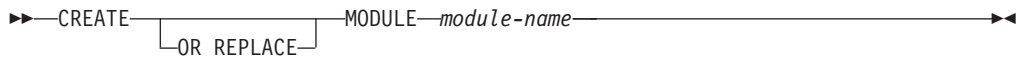
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 모듈의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 모듈의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

기존 모듈을 교체하려면 명령문의 권한 부여 ID는 기존 모듈의 소유자여야 합니다(SQLSTATE 42501).

구문



설명

OR REPLACE

현재 서버에 모듈 정의가 있는 경우 모듈 정의가 교체되도록 지정합니다. 모듈의 모든 오브젝트를 포함하여 기존 모듈 정의는 카탈로그에서 새 정의가 교체되기 전에 적절하게 삭제됩니다. 예외로, 모듈에 대한 권한이 부여된 특권에는 영향을 주지 않습니다. 이 옵션은 모듈의 정의가 현재 서버에 존재하지 않는 경우 무시됩니다.

module-name

모듈의 이름을 지정합니다. 내재적 또는 명시적 규정자를 포함하는 이름은 현재 서버에서 기존 모듈을 식별하면 안됩니다. 모듈 이름 및 스키마 이름은 'SYS'(SQLSTATE 42939)로 시작할 수 없으며 SESSION 사용도 권장되지 않습니다.

주

- 모듈은 다른 데이터베이스 오브젝트 콜렉션으로 사용됩니다. 모듈이 작성되면 모듈의 오브젝트는 ALTER MODULE문을 사용하여 관리됩니다. 모듈에는 함수, 프로시저, 유형, 전역 변수 및 조건이 포함될 수 있습니다. 모듈의 오브젝트는 모듈 밖에서 참

조할 수 있도록 발행할 수 있습니다. 오브젝트를 발행하지 않으면 모듈에서만 참조할 수 있습니다. 모듈은 2 파트로 구성될 수 있습니다.

- 모듈 스펙은 공개된 모든 오브젝트로 구성됩니다(모든 루틴의 본문은 제외).
- 발행되지 않은 모든 오브젝트 및 공개된 모든 루틴의 본문으로 구성되는 모듈 본문.

모듈 관리 조치 포함

- ADD - 오브젝트를 발행하지 않고 모듈에 오브젝트를 추가하거나 루틴 프로토타입을 구현된 루틴 정의로 교체합니다.
- PUBLISH - 오브젝트를 모듈에 추가하고 발행합니다.
- COMMENT - 모듈의 오브젝트에 주석을 추가합니다.
- DROP - 모듈에서 오브젝트를 삭제(drop)하거나 모듈 본문을 삭제합니다.

모듈을 참조하려면 모듈에 하나 이상의 발행된 오브젝트가 있어야 합니다.

예 :

다음은 *SalesModule* 이름의 모듈을 작성하는 데 사용된 CREATE MODULE문의 예입니다.

```
CREATE MODULE salesModule
```

CREATE NICKNAME

CREATE NICKNAME문은 데이터 소스 오브젝트에 대한 별칭을 정의하십시오.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

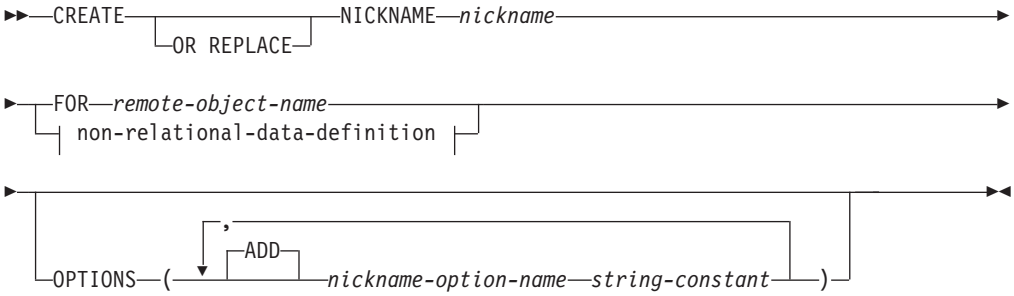
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 페더레이티드 데이터베이스에 대한 CREATETAB 권한은 다음 경우 중 하나입니다.
  - 별칭의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 페더레이티드 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 별칭의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

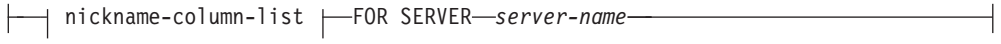
사용자 맵핑이 필요한 데이터 소스의 경우, 데이터 소스의 권한 부여 ID는 별칭이 나타내는 오브젝트에서 데이터를 선택할 수 있는 특권을 가지고 있어야 합니다.

기존 별칭을 교체하려면 명령문의 권한 부여 ID가 기존 별칭의 소유자여야 합니다(SQLSTATE 42501).

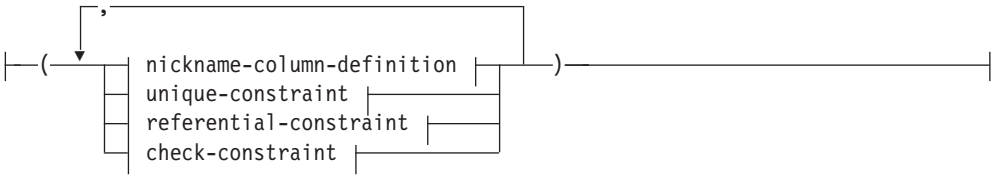
구문



non-relational-data-definition:

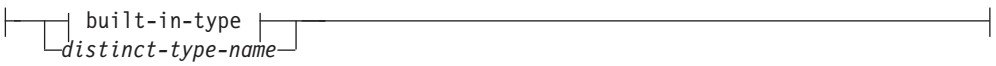


nickname-column-list:

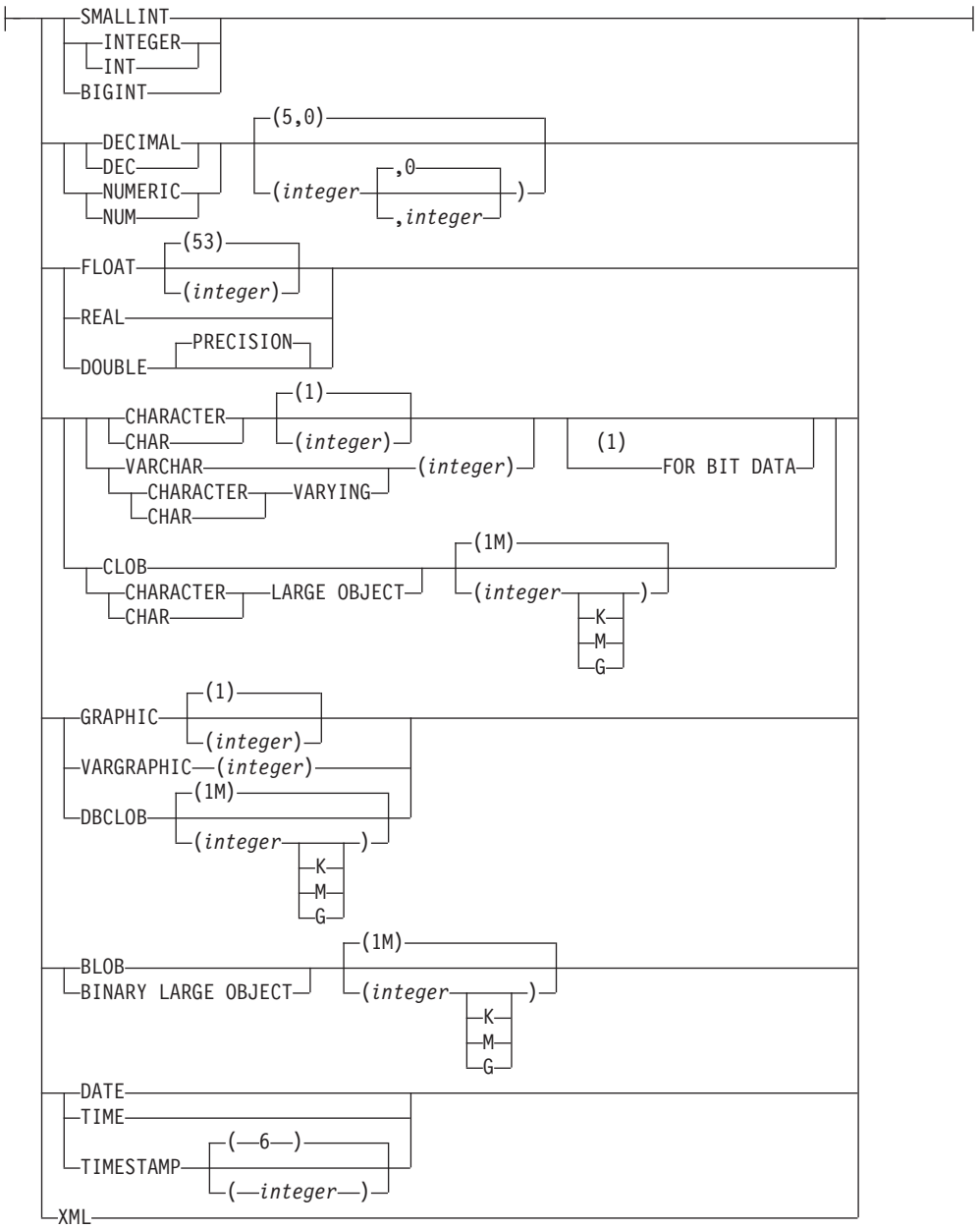


nickname-column-definition:

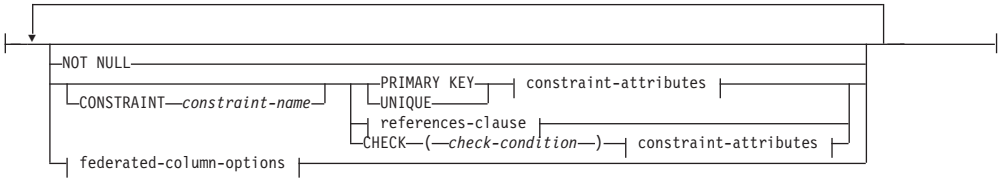
local-data-type:



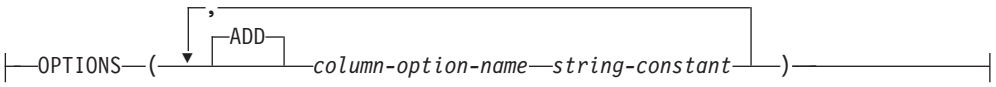
내장 유형:



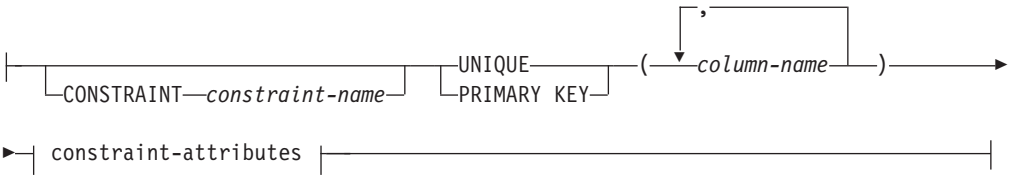
nickname-column-options:



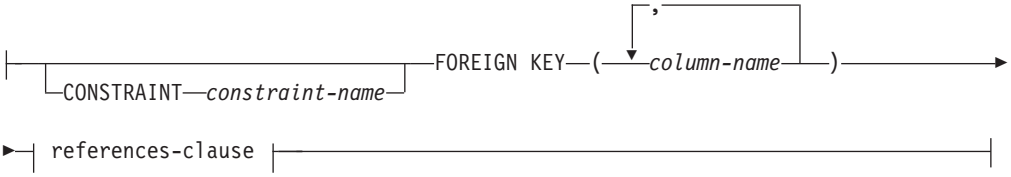
federated-column-options:



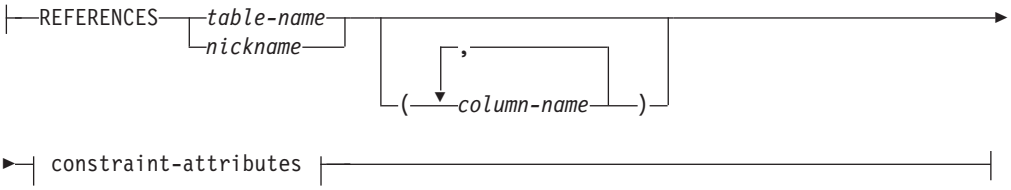
unique-constraint:



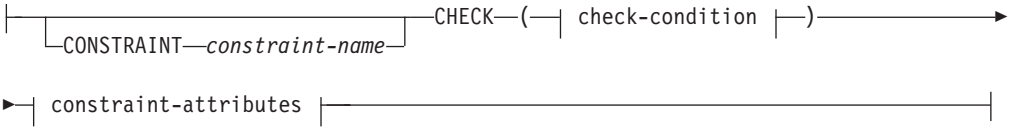
referential-constraint:



references-clause:

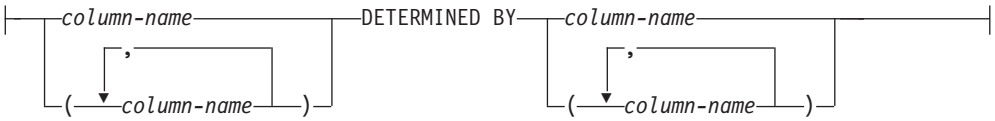


check-constraint:

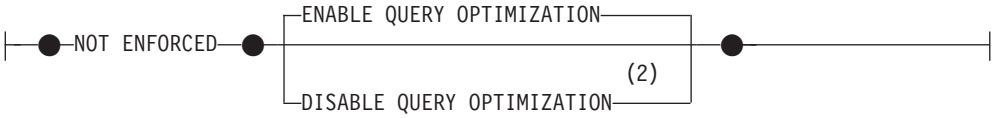


check-condition:

functional-dependency:



constraint-attributes:



주:

1 FOR BIT DATA절은 다른 컬럼 제한조건에 대해 임의의 순서로 지정될 수 있습니다.

2 DISABLE QUERY OPTIMIZATION은 고유 또는 기본 키 제한조건에 대해 지원되지 않습니다.

설명

OR REPLACE

현재 서버에 별칭이 존재하면 별칭의 정의를 교체하도록 지정합니다. 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 삭제되며, 별칭에 부여되었던 특권에는 영향을 주지 않습니다. 별칭의 정의가 현재 서버에 존재하지 않으면 이 옵션이 무시됩니다.

별칭

데이터 소스 오브젝트에 대해 페더레이티드 서버가 사용하는 별칭(ID)을 지정합니다. 내재적 또는 명시적인 규정자를 포함한 별칭은 카탈로그에 기술된 테이블, 뷰, 별칭 또는 별명을 식별해서는 안됩니다. 데이터 소스 오브젝트는 DB2 별명이 될 수 없습니다. 스키마 이름은 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

FOR *remote-object-name*

ID를 지정합니다. 스키마 이름을 지원하는 데이터 소스의 경우, 이 ID는 *data-source-name.remote-schema-name.remote-table-name* 형식의 세 부분으로 구성된 ID입니다. 스키마 이름을 지원하지 않는 데이터 소스의 경우, 이 ID는 *data-source-name.remote-table-name* 형식의 두 부분으로 구성된 ID입니다.

data-source-name

별칭을 작성할 테이블이나 뷰를 포함하는 데이터 소스의 이름을 지정합니다. *data-source-name*은 CREATE SERVER문에서 *server-name*에 지정된 것과 같은 이름입니다.

remote-schema-name

테이블이나 뷰가 속하는 스키마의 이름을 지정합니다. 리모트 스키마 이름에 특수 문자나 소문자가 포함되어 있으면 큰따옴표로 묶어야 합니다.

remote-table-name

별칭을 작성할 특정 데이터 소스 오브젝트(테이블 또는 뷰)의 이름을 지정합니다. 테이블 이름은 선언된 임시 테이블이 될 수 없습니다(SQLSTATE 42995). 리모트 테이블 이름에 특수 문자나 소문자가 포함되어 있으면 큰따옴표로 묶어야 합니다.

non-relational-data-definition

비관계형 랩퍼를 통해 액세스되는 데이터를 정의합니다.

nickname-column-definition

별칭의 컬럼에 대한 로컬 속성을 정의합니다. 일부 랩퍼에는 이런 속성을 지정해야 하지만 다른 랩퍼의 경우에는 데이터 소스를 통해 이런 속성을 판별합니다.

column-name

컬럼에 대한 로컬 이름을 지정합니다. 이 이름은 *remote-object-name*의 해당 컬럼의 이름과 다를 수 있습니다.

local-data-type

컬럼에 대한 로컬 데이터 유형을 지정합니다. 대부분의 랩퍼는 SQL 데이터 유형 중 일부만 지원합니다. 특정 데이터 유형에 대한 설명은 『CREATE TABLE』을 참조하십시오.

nickname-column-options

별칭 컬럼과 관련된 추가 옵션을 지정합니다.

NOT NULL

컬럼에 널(NULL) 값을 허용하지 않도록 지정합니다.

CONSTRAINT *constraint-name*

제한조건이 이름을 지정합니다. *constraint-name*은 같은 CREATE NICKNAME문 내에 이미 지정된 제한조건을 식별해서는 안됩니다(SQLSTATE 42710).

이 절이 생략되는 경우, 시스템은 별칭에 정의된 기존의 제한조건 ID 가운데 고유한 18바이트의 ID를 생성합니다. 이 ID는 'SQL'과 그 뒤에 오는 시간소인 함수가 생성한 15개의 숫자로 구성됩니다.

PRIMARY KEY 또는 UNIQUE 제한조건과 함께 사용할 경우, 제한조건을 지원하기 위해 작성된 인덱스 스펙의 이름으로 *constraint-name*을 사용할 수 있습니다.

PRIMARY KEY

단일 컬럼으로 구성되는 기본 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 PRIMARY KEY가 C 컬럼 정의에 지정된 경우, PRIMARY KEY(C)를 별도의 절로 지정한 경우와 그 결과가 같습니다.

다음의 *unique-constraint* 아래에 있는 PRIMARY KEY를 참조하십시오.

UNIQUE

단일 컬럼으로 구성되는 고유 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 UNIQUE가 C 컬럼 정의에 지정된 경우, UNIQUE(C)를 별도의 절로 지정한 경우와 그 결과가 같습니다.

다음 *unique-constraint* 아래에 있는 UNIQUE를 참조하십시오.

references-clause

단일 컬럼으로 구성되는 외부 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 references-clause가 C 컬럼의 정의에 지정된 경우, C가 유일하게 식별되는 컬럼인 FOREIGN KEY절의 일부로 references-clause를 지정한 것과 그 결과가 같습니다.

다음의 *referential-constraint* 아래에 있는 *references-clause*를 참조하십시오.

CHECK (*check-condition*)

단일 컬럼에 적용되는 점검 제한조건을 정의함에 있어 간편한 방법을 제공합니다. 아래의 CHECK (*check-condition*)을 참조하십시오.

OPTIONS

별칭이 작성될 때 추가되는 컬럼 옵션을 표시합니다. 일부 랩퍼의 경우 특정 컬럼 옵션을 지정해야 합니다.

ADD

컬럼 옵션을 추가합니다.

column-option-name

옵션 이름을 지정합니다.

string-constant

*column-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

unique-constraint

고유 제한조건이나 기본 키 제한조건을 정의합니다.

CONSTRAINT *constraint-name*

기본 키 또는 고유 제한조건의 이름을 지정합니다.

UNIQUE (*column-name*,...)

식별된 컬럼으로 구성되는 고유 키를 정의합니다. 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 별칭의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안됩니다.

식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 키 길이 한계에 대해서는 『SQL 및 XQuery 한계』를 참조하십시오. 컬럼의 길이 속성이 페이지 크기의 인덱스 키 길이 한계를 초과하지 않아도 LOB 컬럼, LOB를 기반으로 하는 구별 유형 컬럼 또는 구조화된 유형 컬럼은 고유 키의 일부로 사용할 수 없습니다(SQLSTATE 54008).

고유 키의 컬럼 세트는 기본 키 또는 다른 고유 키에 있는 컬럼 세트와 동일할 수 없습니다(SQLSTATE 01543). LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 발생합니다(SQLSTATE 42891).

카탈로그에 기록된 대로 별칭의 설명에는 고유 키와 해당 인덱스 스펙이 포함됩니다. 인덱스 스펙은 컬럼에 대해 자동으로 작성됩니다(각 컬럼에 대해 오름차순으로). 인덱스 스펙의 이름은 별칭이 작성된 스키마의 기존 인덱스 또는 인덱스 스펙과 상충하지 않을 경우, *constraint-name*과 동일합니다. 인덱스 스펙의 이름이 상충하는 경우, 이름은 'SQL'과 그 뒤에 오는 문자 시간소인(*yymmddhhmmssxxx*)으로 이루어지며, 스키마 이름은 SYSIBM이 됩니다.

PRIMARY KEY (*column-name*,...)

식별된 컬럼으로 구성되는 기본 키를 정의합니다. 절은 두 번 이상 지정되어서는 안되며, 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 별칭의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안됩니다.

식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 키 길이 한계에 대해서는 『SQL 및 XQuery 한계』를 참조하십시오. 컬럼의 길이 속성이 페이지 크기의 인덱스 키 길이 한계를 초과하지 않아도 LOB 컬럼, LOB를 기반으로 하는 구별 유형 컬럼 또는 구조화된 유형 컬럼은 기본 키의 일부로 사용할 수 없습니다(SQLSTATE 54008).

기본 키의 컬럼 세트는 고유 키의 컬럼 세트와 같을 수 없습니다(SQLSTATE 01543). LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 발생합니다(SQLSTATE 42891).

하나의 별칭에 단 하나의 기본 키만 정의할 수 있습니다.

카탈로그에 기록된 대로 별칭의 설명에는 기본 키와 해당 인덱스 스펙이 포함됩니다. 인덱스 스펙은 컬럼에 대해 자동으로 작성됩니다(각 컬럼에 대해 오름

차순으로). 인덱스 스펙의 이름은 별칭이 작성된 스키마의 기존 인덱스 또는 인덱스 스펙과 상충하지 않을 경우, *constraint-name*과 동일합니다. 인덱스 스펙의 이름이 상충하는 경우, 이름은 'SQL'과 그 뒤에 오는 문자 시간소인(*yymmddhhmmssxxx*)으로 이루어지며, 스키마 이름은 SYSIBM이 됩니다.

referential-constraint

참조 제한조건을 정의합니다.

CONSTRAINT *constraint-name*

참조 제한조건의 이름을 지정합니다.

FOREIGN KEY (*column-name***,...)**

지정된 *constraint-name*을 가진 참조 제한조건을 정의합니다.

N1은 명령문의 오브젝트 별칭을 나타냅니다. 참조 제한조건의 외부 키는 식별되는 컬럼으로 구성됩니다. 컬럼 이름 목록의 각 이름은 N1의 컬럼을 식별하고, 같은 컬럼은 두 번 이상 식별되어서는 안됩니다.

식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 『CREATE TABLE』의 『바이트 수』를 참조하십시오. 키 길이 한계에 대해서는 『SQL 및 XQuery 한계』를 참조하십시오. 외부 키는 길이가 255바이트 이상인 가변 길이 컬럼에 정의할 수 있습니다. LOB 컬럼, LOB를 기반으로 하는 구별 유형 또는 구조화된 유형 컬럼은 외부 키의 일부로 사용될 수 없습니다(SQLSTATE 42962). 상위 키에 있는 것과 동일한 수의 외부 키 컬럼이 있어야 하며, 해당 컬럼의 데이터 유형은 호환 가능해야 합니다(SQLSTATE 42830). 데이터 유형이 호환 가능할 경우 두 컬럼 설명이 호환 가능합니다(두 컬럼이 숫자, 문자열, 그래픽, 날짜 시간 또는 동일한 구별 유형일 경우).

references-clause

참조 제한조건에 대한 상위 테이블 또는 상위 별칭 및 상위 키를 지정합니다.

REFERENCES *table-name* 또는 *nickname*

REFERENCES절에 지정된 테이블 또는 별칭은 카탈로그에 기술된 기본 테이블 또는 별칭을 식별해야 하나, 카탈로그 테이블을 식별해서는 안됩니다.

참조 제한조건의 외부 키, 상위 키, 상위 테이블 또는 상위 별칭이 이전에 지정된 참조 제한조건의 외부 키, 상위 키, 상위 테이블 또는 상위 별칭과 동일할 경우, 참조 제한조건이 중복됩니다. 중복 참조 제한조건은 무시되며 경고가 리턴됩니다(SQLSTATE 01543).

아래 설명에서 N2는 식별된 상위 테이블 또는 상위 별칭을 나타내고, N1은 작성되거나 변경될 별칭을 나타냅니다. N1과 N2는 동일한 별칭일 수 있습니다.

지정되는 외부 키는 N2의 상위 키와 컬럼 수가 같아야 하며, 외부 키의 *n* 번째 컬럼의 설명은 상위 키의 *n*번째 컬럼의 설명에 해당합니다. 날짜 시간 컬럼은 이 규칙의 목적상 문자열 컬럼과 비교 가능한 것으로 간주되지 않습니다.

FOREIGN KEY절에 의해 지정된 참조 제한조건은 N2가 상위이고 N1이 종속인 관계를 정의합니다.

(*column-name***,...)**

참조 제한조건의 상위 키는 식별된 컬럼으로 구성됩니다. 각 *column-name*은 N2의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다. 같은 컬럼은 한 번 이상 식별될 수 없습니다.

컬럼 이름의 목록은 N2에 존재하는 기본 키나 고유 제한조건의 컬럼 세트(순서는 관계없음)와 일치해야 합니다(SQLSTATE 42890). 컬럼 이름 목록을 지정하지 않을 경우, N2에 기본 키가 있어야 합니다(SQLSTATE 42888). 컬럼 이름 목록을 생략하면 원래 지정된 순서로 해당 기본 키의 컬럼 스펙이 내재됩니다.

constraint-attributes

참조 무결성이나 점검 제한조건과 연관된 속성을 정의합니다.

NOT ENFORCED

제한조건이 삽입, 갱신 또는 삭제 등 정상 조작 중에 데이터베이스 관리 프로그램에 의해 시행되지 않습니다.

ENABLE QUERY OPTIMIZATION

제한조건을 참으로 가정하며 적절한 환경에서 쿼리 최적화에 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

제한조건을 쿼리 최적화에 사용할 수 없습니다.

check-constraint

점검 제한조건을 정의합니다. *check-constraint*는 참으로 평가되어야 하는 *search-condition*이거나 컬럼 간의 함수적 종속성을 정의하는 search-condition입니다.

CONSTRAINT *constraint-name*

점검 제한조건의 이름을 지정합니다.

CHECK (*check-condition***)**

점검 제한조건을 정의합니다. *check-condition*은 별칭의 모든 행에 대해 참이거나 알 수 없어야 합니다.

search-condition

*search-condition*에는 다음과 같은 제한사항이 있습니다.

- 컬럼 참조는 작성할 별칭의 컬럼이어야 합니다.
- *search-condition*에는 TYPE 술어가 포함될 수 없습니다.
- 다음 중 어느 것도 포함될 수 없습니다(SQLSTATE 42621).
  - 서브쿼리
  - 범위 지정된 참조 인수가 오브젝트 ID(OID) 컬럼이 아닌 비참조 조작 또는 DEREF 함수
  - SCOPE절이 있는 CAST 권장 스펙
  - 컬럼 함수
  - 결정적이지 않은 함수
  - 외부 조치를 갖도록 정의된 함수
  - CONTAINS SQL 또는 READS SQL DATA로 정의된 사용자 정의 함수(UDF)
  - 호스트 변수
  - 매개변수 표시문자
  - 특수 레지스터의 값에 따라 다른 특수 레지스터 및 내장 함수
  - 전역 변수
  - 식별 컬럼이 아닌 다른 생성된 컬럼에 대한 참조

functional-dependency

컬럼 간의 함수적 종속성을 정의합니다.

컬럼의 수퍼 세트에는 DETERMINED BY절 바로 앞에 오는 식별된 컬럼이 포함됩니다. 컬럼의 하위 세트에는 DETERMINED BY절 바로 뒤에 오는 식별된 컬럼이 포함됩니다. *search-condition*의 모든 제한사항은 수퍼 세트 및 하위 세트 컬럼에 적용되며 컬럼 세트에는 단순 컬럼 참조만 허용됩니다(SQLSTATE 42621). 동일한 컬럼은 함수적 종속성에서 여러 번 식별될 수 없습니다(SQLSTATE 42709). 컬럼의 데이터 유형은 LOB 데이터 유형, LOB 데이터 유형을 기반으로 하는 구별 유형 또는 구조화된 유형일 수 없습니다(SQLSTATE 42962). 컬럼의 하위 세트에 있는 컬럼은 널(NULL) 입력 가능 컬럼일 수 없습니다(SQLSTATE 42621).

*column-definition*의 일부분으로 점검 제한조건이 지정된 경우, 컬럼 참조는 같은 컬럼에 대해서만 이루어질 수 있습니다. 별칭 정의의 일부분으로 지정된 점검 제한조건은 이전에 CREATE NICKNAME문에 정의한 컬럼을 식별하는 컬럼 참조를 가질 수 있습니다. 비일관성, 중복 조건 또는 동등한 조건에 대해 점검 제한조건이 점검되지 않습니다. 그러므로 모순되거나 불필요한 점검 제한조건이 정의되면 실행시 오류가 발생할 수 있습니다.

FOR SERVER *server-name*

CREATE SERVER문을 사용하여 등록된 서버를 지정합니다. 이 서버는 별칭 데이터에 액세스하는 데 사용됩니다.

OPTIONS

별칭이 작성될 때 사용 가능하게 되는 별칭 옵션을 표시합니다.

ADD

별칭 옵션을 추가합니다.

nickname-option-name

옵션 이름을 지정합니다.

string-constant

*nickname-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

주

- 관계형 데이터 소스 오브젝트의 예는 테이블 및 뷰입니다. 비관계형 데이터 소스 오브젝트의 예는 Documentum 오브젝트나 등록된 테이블, 텍스트 파일(.txt) 및 Microsoft Excel 파일(.xls)입니다.
- 별칭이 참조하는 데이터 소스 오브젝트가 이미 *remote-object-name*의 첫 번째 규정자에 의해 표시되는 데이터 소스에 존재해야 합니다.
- 지원되는 데이터 소스 데이터 유형 목록은 랩퍼에 따라 다릅니다. XML 및 REF 데이터 소스 데이터 유형은 모든 랩퍼에서 지원되지 않습니다. DECFLOAT 데이터 소스 데이터 유형은 Linux, UNIX 및 Windows용 IBM DB2 버전 9.5 이상의 DB2 랩퍼에서만 지원합니다. CREATE NICKNAME문에서 지원되지 않는 데이터 유형을 가진 컬럼이 있는 *remote-object-name*을 지정하면 오류가 발생합니다.

  LONG VARCHAR 및 LONG VARGRAPHIC 데이터 소스 데이터 유형은 CLOB 및 DBCLOB 데이터 유형에 각각 맵핑됩니다. LONG VARCHAR FOR BIT DATA는 BLOB에 맵핑됩니다.
- DB2 인덱스 이름의 최대 허용 길이는 128바이트입니다. 이름이 이 길이를 초과하는 인덱스를 가지는 관계형 테이블에 대해 별칭이 작성되고 있다면, 전체 이름이 카탈로그화되지 않습니다. 대신 DB2는 이름을 128바이트로 절단합니다. 해당 문자로 구성된 문자열이 인덱스가 속하는 스키마 내에서 고유하지 않으면 DB2가 마지막 문자를 0으로 대체하여 이를 고유하게 만들려고 시도합니다. 결과가 여전히 고유하지 않으면 DB2가 마지막 문자를 1로 바꿉니다. DB2는 해당 이름의 127번째 문자, 126번째 문자 등에 대해 2에서 9까지의 숫자를, 필요하면 0에서 9까지의 숫자를 사용하여 고유한 이름이 생성될 때까지 이 프로세스를 반복합니다. 데이터 소스 테이블의 인덱스의 130바이트 이름은 AREALLY...REALLYLONGNAME입니다. 이름 AREALLY...REALLYLONGNA 및 AREALLY...REALLYLONGN0는 이미 이 인덱스가 속하는 스키마에 존재합니다. 새 이름은 128바이트를 초과하므로 DB2가 이를 AREALLY...REALLYLONGNA로 절단합니다. 해당 이름이 이미 스키마에 존

재하기 때문에 DB2는 절단된 버전을
AREALLY...REALLYLONGN0로 변경합니다. 또한 이 이름도 존재하기 때문에 DB2는 절단된 버전을 AREALLY...REALLYLONGN1로 변경합니다. 이 이름은 스키마에 아직 없으므로 DB2는 이것을 새 이름으로 받아들입니다.

- 데이터 소스 오브젝트에 대한 별칭이 작성될 때 DB2는 별칭 컬럼의 이름을 카탈로그에 저장합니다. 데이터 소스 오브젝트가 테이블이나 뷰일 경우 DB2는 별칭 컬럼 이름을 테이블이나 뷰의 컬럼 이름과 동일하게 만듭니다. 이름이 DB2 컬럼 이름에 대해 허용되는 최대 길이를 초과할 경우 DB2는 이 길이로 이름을 절단합니다. 절단된 버전이 테이블이나 뷰의 다른 컬럼 이름들 사이에서 고유하지 않을 경우 DB2는 이전 단락에 기술된 프로시저에 따라 이름을 고유하게 만듭니다.
- 데이터 소스 오브젝트에 정의된 인덱스가 있는 경우, 별칭 작성시 각 인덱스에 대한 인덱스 스펙이 작성됩니다. 다음 특징을 가진 인덱스에 대한 데이터 소스에서는 인덱스 스펙이 작성되지 않습니다.
  - 중복 컬럼 이름
  - 64컬럼 초과
  - 인덱스 키 파트 길이의 전체 합이 1024바이트를 초과함
- 리모트 데이터 소스 오브젝트의 정의가 변경되면(예: 컬럼이 삭제되거나 데이터 유형이 변경되면), 별칭을 삭제하고 다시 작성해야 합니다. 그렇지 않으면 SQL문에서 별칭을 사용할 때 오류가 발생할 수 있습니다.

예:

예 1: HEDGES라는 스키마에 있는 뷰 DEPARTMENT에 대한 별칭을 작성하십시오. 이 뷰는 OS390A라는 z/OS용 DB2 데이터 소스에 저장됩니다.

```
CREATE NICKNAME DEPT
  FOR OS390A.HEDGES.DEPARTMENT
```

예 2: 예 1에서 작성된 별칭에 대한 뷰에서 모든 레코드를 선택하십시오. 뷰는 별칭으로 참조되어야 합니다. 리모트 뷰는 pass-through 세션에 있는 데이터 소스에서만 아는 이름을 사용하여 참조할 수 있습니다.

```
SELECT * FROM DEPT                          Valid after nickname DEPT is created

  SELECT * FROM OS390A.HEDGES.DEPARTMENT     Invalid
```

예 3: salesdata라는 스키마에 있는 리모트 테이블 JAPAN에 대한 별칭을 작성하십시오. 데이터 소스의 스키마 이름 및 테이블 이름은 소문자로 저장되기 때문에 리모트 스키마 이름 및 테이블 이름을 큰따옴표로 묶어서 지정하십시오.

```
CREATE NICKNAME JPSALES
  FOR asia."salesdata"."japan"
```

예 4: 테이블 구조 파일 DRUGDATA1.TXT에 대한 별칭을 작성하십시오. 명령문에 FILE_PATH, COLUMN DELIMITER, KEY_COLUMN 및 VALIDATE_DATA_FILE 별칭 옵션을 포함시키십시오.

```
CREATE NICKNAME DRUGDATA1
       (Dcode         INTEGER,
       DRUG           CHAR(20),
       MANUFACTURER   CHAR(20))
  FOR SERVER biochem_lab
  OPTIONS
           (FILE_PATH '/usr/pat/DRUGDATA1.TXT',
     COLUMN_DELIMITER ',',
     KEY_COLUMN 'DCODE',
     SORTED 'Y',
     VALIDATE_DATA_FILE 'Y')
```

예 5: 지정된 디렉토리 경로 /home/db2user 아래에 있는 다중 XML 파일에 대해 상위 별칭 CUSTOMERS를 작성하십시오. 다음 옵션을 포함시키십시오.

- 컬럼 옵션:
  - 이름이 ID인 VARCHAR(5) 컬럼에 대한 XPATH 컬럼 옵션. 이 옵션은 컬럼 데이터가 추출되는 XML 파일에 있는 요소 또는 속성을 나타냅니다.
  - 이름이 NAME인 VARCHAR(16) 컬럼에 대한 XPATH 컬럼 옵션. 이 옵션은 컬럼 데이터가 추출되는 XML 파일에 있는 요소 또는 속성을 나타냅니다.
  - 이름이 ADDRESS인 VARCHAR(30) 컬럼에 대한 XPATH 컬럼 옵션. 이 옵션은 컬럼 데이터가 추출되는 XML 파일에 있는 요소 또는 속성을 나타냅니다.
  - 이름이 CID인 VARCHAR(16) 컬럼에 대한 PRIMARY_KEY 컬럼 옵션. 이 옵션은 고객 별칭을 별칭 계층 구조에서 상위 별칭으로 식별합니다.
- 별칭 옵션:
  - 데이터를 제공하는 XML 파일의 위치를 표시하는 DIRECTORY_PATH 별칭 옵션
  - 데이터가 시작되는 XML 파일의 요소를 표시하는 XPATH 별칭 옵션
  - XML 소스 데이터가 요소 단위로 구분되고 처리되는 것을 표시하는 STREAMING 별칭 옵션. 이 예에서는 고객 레코드가 요소입니다.

```
CREATE NICKNAME customers
       (id      VARCHAR(5)   OPTIONS(XPATH './@id'),
         name    VARCHAR(16)  OPTIONS(XPATH './/name'),
         address VARCHAR(30)  OPTIONS(XPATH './/address/@street'),
         cid     VARCHAR(16)  OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS
            (DIRECTORY_PATH '/home/db2user',
      XPATH '//customer',
      STREAMING 'YES')
```

CREATE PROCEDURE

CREATE PROCEDURE 문은 현재 서버에 프로시저를 정의합니다.

이 명령문을 사용하여 다른 세 가지 유형의 프로시저를 작성할 수 있습니다. 각 유형은 별도로 선언됩니다.

- 외부. 프로시저 본문이 프로그래밍 언어로 작성됩니다. 외부 실행 파일은 다양한 프로시저 속성과 함께 현재 서버에 정의된 프로시저에서 참조됩니다.
- 전래. 프로시저 본문은 다양한 프로시저 속성과 함께 현재 서버에 정의된 소싱된 프로시저에서 참조되는 소싱된 프로시저의 일부입니다. 해당 소싱된 프로시저가 데이터 소스에 있는 소싱된 프로시저를 *페더레이티드 프로시저*라고도 합니다.
- SQL. 프로시저 본문이 SQL로 작성되고 다양한 프로시저 속성과 함께 현재 서버에서 정의됩니다.

CREATE PROCEDURE(외부)

CREATE PROCEDURE(외부)문은 현재 서버에서 외부 프로시저를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 데이터베이스의 CREATE_EXTERNAL_ROUTINE 권한은 다음 중 최소한 하나를 포함해야 합니다.
  - 프로시저의 스키마 이름이 기존의 스키마를 참조하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 프로시저의 스키마 이름이 기존 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

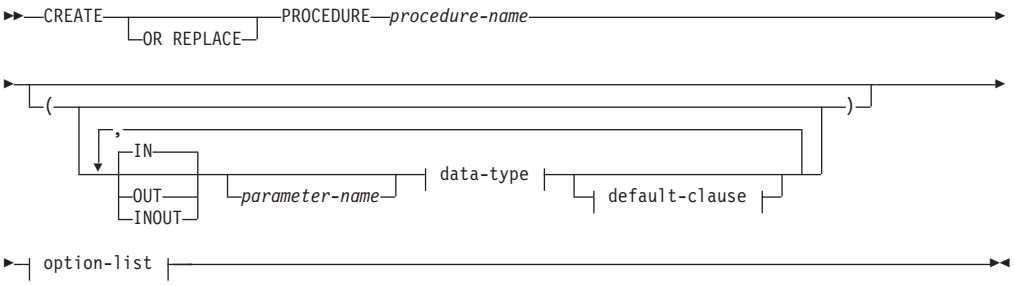
비분리 프로시저를 작성하려면, 명령문의 권한 부여 ID가 보유한 특권으로 다음 중 적어도 하나가 포함되어야 합니다.

- 데이터베이스에서의 CREATE_NOT_FENCED_ROUTINE 권한
- DBADM 권한

분리 프로시저를 작성하는 데 추가 권한이나 특권이 필요하지는 않습니다.

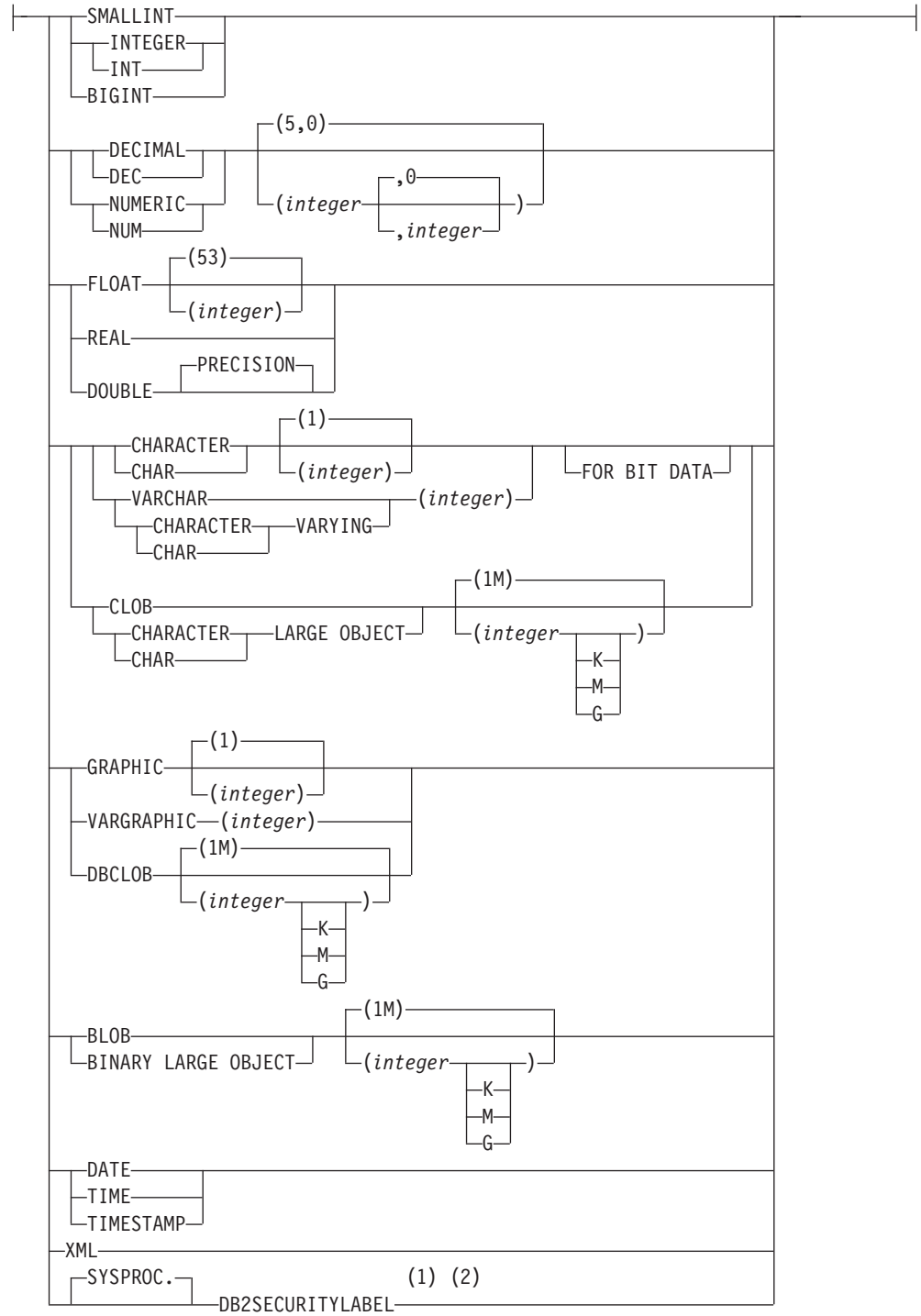
기존 프로시저를 교체하려면, 명령문의 권한 부여 ID가 기존 프로시저의 소유자여야 합니다(SQLSTATE 42501).

구문



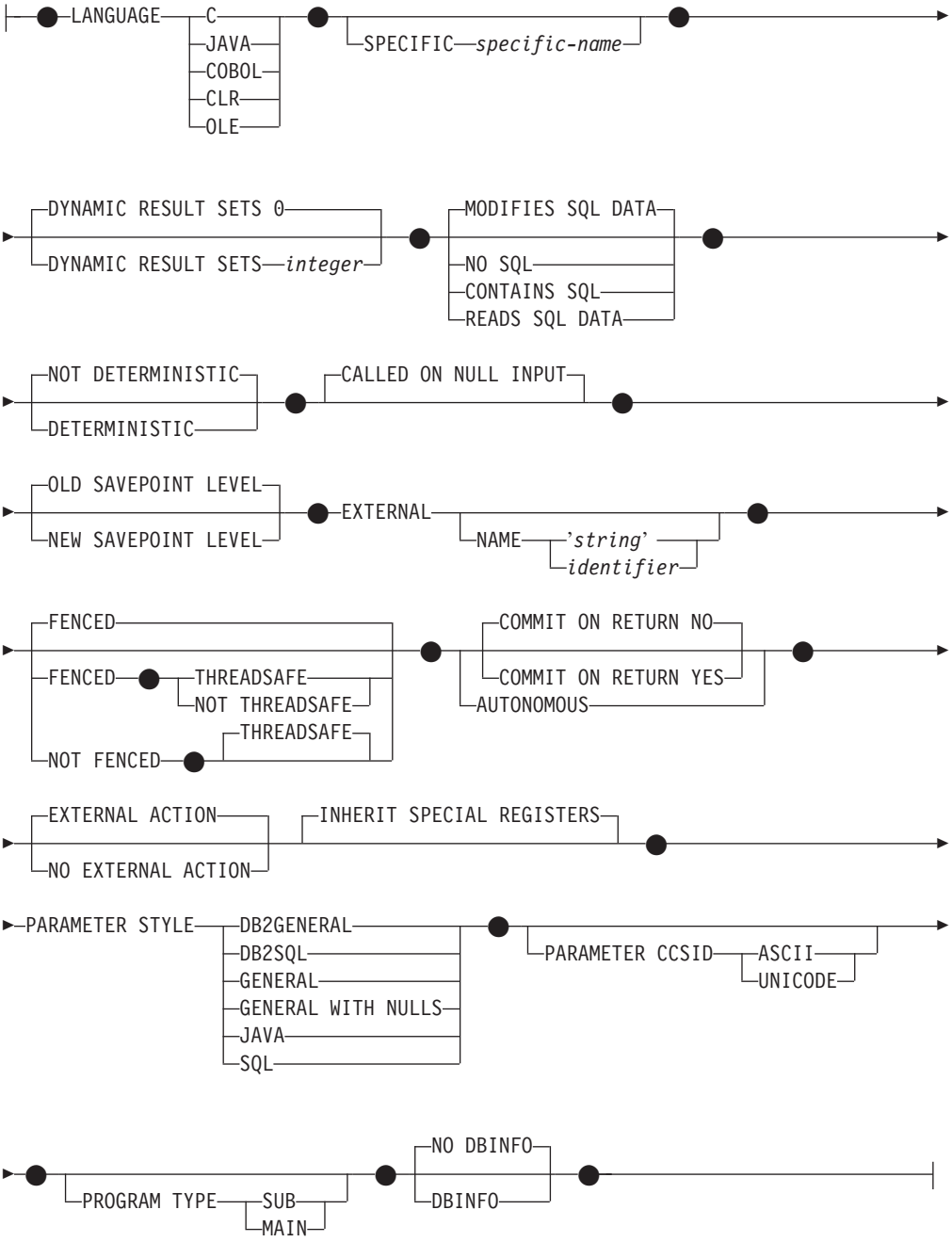
data-type:

built-in type:



default-clause:

option-list:



주:

1 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

2 DB2SECURITYLABEL 유형의 컬럼의 경우, NOT NULL WITH DEFAULT는 내재적 및 명시적으로 지정될 수 없습니다(SQLSTATE 42842). DB2SECURITYLABEL 유형의 컬럼 디폴트값은 쓰기 액세스에 대한 세션 권한 부여 ID의 보안 레이블입니다.

설명

OR REPLACE

프로시저 정의가 현재 서버에 존재하는 경우 이를 교체하도록 지정합니다. 프로시저에 대해 부여된 특권에 영향을 주지 않는다는 점을 제외하고 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 효과적으로 삭제됩니다. 프로시저 정의가 현재 서버에 없으면 이 옵션은 무시됩니다. 기존 프로시저를 교체하려면, 새 정의의 특정 이름 및 프로시저 이름이 이전 정의의 특정 이름 및 프로시저 이름과 동일하거나 새 정의의 시그니처가 이전 정의의 시그니처와 일치해야 합니다. 그렇지 않으면, 새 프로시저가 작성됩니다.

procedure-name

정의되고 있는 프로시저에 이름을 지정하십시오. 이는 프로시저를 지정하는 규정화되거나 규정되지 않은 이름입니다. *procedure-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 128)입니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다.

매개변수 수와 더불어 내재적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 기술된 프로시저를 나타내면 안됩니다(SQLSTATE 42723). 규정되지 않은 이름은 매개변수의 개수와 함께 여러 스키마에서 고유하지 않아도 됩니다.

두 부분으로 구성된 이름을 지정할 경우, *schema-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

*(***IN | OUT | INOUT** *parameter-name data-type default-clause,...)*

프로시저 매개변수를 식별하고 각 매개변수의 모드, 선택적 매개변수 이름, 데이터 유형 및 선택적 디폴트값을 지정합니다. 프로시저에 예상되는 각 매개변수에 대해 목록에 있는 한 항목이 지정되어야 합니다.

한 스키마 내에서 동일하게 이름이 지정된 두 프로시저는 같은 수의 매개변수를 가질 수 없습니다. 시그니처가 중복되면 SQL 오류가 리턴됩니다(SQLSTATE 42723).

예를 들어, 다음 명령문이 제공되면, 다음과 같습니다.

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

데이터 유형은 달라도 프로시저에 있는 매개변수 수가 동일하므로 두 번째 명령문은 실패합니다.

프로시저에서 오류가 리턴되면 OUT 매개변수의 정의가 취소되고 INOUT 매개변수는 변경되지 않습니다.

IN 매개변수를 프로시저에 대한 입력 매개변수로 식별합니다. 프로시저 내에서 이뤄진 매개변수 변경사항은 제어가 리턴될 때 호출 SQL 응용프로그램에서 사용할 수 없습니다. 디폴트값은 IN입니다.

OUT
매개변수를 프로시저에 대한 출력 매개변수로 식별합니다.

INOUT
매개변수를 프로시저에 대한 입력 매개변수와 출력 매개변수로 식별합니다.

parameter-name
매개변수의 이름을 선택적으로 지정합니다. 이 매개변수 이름은 프로시저에 대해 고유해야 합니다(SQLSTATE 42734).

data-type
매개변수의 데이터 유형을 지정합니다. 구조화된 유형은 지정할 수 없습니다(SQLSTATE 429BB).

built-in-type
내장 데이터 유형을 지정합니다. 각 내장 데이터 유형에 대한 자세한 설명은 “CREATE TABLE”을 참조하십시오. 프로시저 작성에 사용 중인 언어에 해당 항목을 포함하는 내장 데이터 유형만 지정할 수 있습니다.

- 날짜 시간 유형 매개변수는 문자 데이터 유형으로 전달되며 데이터는 ISO 형식으로 전달됩니다.
- XML은 LANGUAGE OLE에서 유효하지 않습니다.
- 프로시저 내부에 표시되는 XML 값이 프로시저 호출에서 매개변수로 전달되는 XML 값의 직렬화된 버전이므로, `XML AS CLOB(n)` 구문을 사용하여 XML 유형 매개변수를 선언해야 합니다.
- CLR은 28보다 큰 DECIMAL 스케일은 지원하지 않습니다(SQLSTATE 42613).
- 10진 부동 소수점은 C, Java COBOL, CLR 및 OLE 언어에는 지원되지 않습니다(SQLSTATE 42613).

array-type-name
사용자 정의 배열 유형의 이름을 지정합니다. 스키마 이름을 포함하지 않고 *array-type-name*을 지정하는 경우 SQL 경로에서 스키마를 검색하여 배열 유

형이 해결됩니다. 배열은 일반 배열이어야 하며 프로시저는 PARAMETER STYLE Java절을 포함하여 정의된 Java 프로시저여야 합니다(SQLSTATE 428H2).

DEFAULT

매개변수의 디폴트값을 지정합니다. 디폴트값은 상수, 특수 레지스터, 전역 변수, 표현식 또는 NULL 키워드일 수 있습니다. 디폴트로 지정할 수 있는 특수 레지스터는 컬럼 디폴트에 지정할 수 있는 특수 레지스터와 동일합니다(″CREATE TABLE″문의 *″default-clause″* 참조). 표현식을 사용하여 기타 특수 레지스터를 디폴트값으로 지정할 수 있습니다.

*expression*은 ″표현식″에서 설명되는 유형의 모든 표현식이 가능합니다. 디폴트값을 지정하지 않으면, 매개변수가 디폴트값을 갖지 않으며 프로시저 호출 시 대응하는 인수를 생략할 수 없습니다. *expression*의 최대 크기는 64K바이트입니다.

디폴트 표현식은 SQL 데이터를 수정하지 않거나(SQLSTATE 428FL 또는 SQLSTATE 429BL) 외부 조치를 수행하지 않아야 합니다(SQLSTATE 42845). 매개변수 데이터 유형과 호환 가능한 표현식을 지정해야 합니다(SQLSTATE 42821).

다음 상황에서는 디폴트값을 지정할 수 없습니다.

- INOUT 또는 OUT 매개변수의 경우(SQLSTATE 42601)
- ARRAY, ROW 또는 CURSOR 매개변수 유형의 경우(SQLSTATE 429BB)

디폴트값이 없는 매개변수를 디폴트값이 있는 매개변수 뒤에 정의할 수 없습니다(SQLSTATE 428HG).

SPECIFIC *specific-name*

정의할 프로시저의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 프로시저를 삭제할 때 또는 프로시저에 주석을 붙일 때 사용할 수 있지만, 프로시저를 호출하는 데에는 사용할 수 없습니다. 규정되지 않은 *specific-name* 양식은 SQL ID입니다(최대 길이는 128). 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버에 존재하는 다른 루틴 인스턴스를 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *procedure-name*과 같을 수 있습니다.

규정자를 지정하지 않으면, *procedure-name*에 대해 사용된 규정자가 사용됩니다. 규정자를 지정하면, 규정자는 *procedure-name*의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 뒤에 문자 시간소인이 오는 'SQL'입니다('SQL*yymmddhhmmssxxx*').

DYNAMIC RESULT SETS *integer*
프로시저에 대한 리턴된 결과 세트의 측정된 상위 바운드를 표시하십시오.

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA
프로시저가 SQL문과 유형을 실행하는지 여부를 표시하십시오.

NO SQL
프로시저가 SQL문을 실행할 수 없음을 나타내십시오(SQLSTATE 38001).

CONTAINS SQL
SQL 데이터를 읽지도 수정하지도 않는 SQL문이 프로시저에 의해 실행될 수 있음을 나타내십시오(SQLSTATE 38004). 프로시저에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003).

READS SQL DATA
SQL 데이터를 수정하지 않는 일부 SQL문이 이 프로시저에 포함될 수 있음을 나타내십시오(SQLSTATE 38002 또는 42985). 프로시저에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003).

MODIFIES SQL DATA
프로시저에서 지원되지 않는 명령문을 제외하고, 프로시저가 SQL문을 실행할 수 있음을 나타냅니다(SQLSTATE 38003).

DETERMINISTIC 또는 NOT DETERMINISTIC
이 절은 프로시저가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 프로시저가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 프로시저는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다.

이 절은 현재 프로시저의 처리에는 영향을 주지 않습니다.

CALLED ON NULL INPUT
CALLED ON NULL INPUT은 항상 프로시저에 적용됩니다. 즉, 프로시저는 널(NULL) 인수가 있는지 여부에 상관없이 호출됩니다. 모든 OUT 또는 INOUT 매개변수는 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 널(NULL) 인수 값에 대한 테스트 책임은 프로시저에 있습니다.

OLD SAVEPOINT LEVEL 또는 NEW SAVEPOINT LEVEL
이 프로시저가 세이브포인트 이름 및 효과에 대한 새 세이브포인트 레벨을 작성하는지의 여부를 지정합니다. OLD SAVEPOINT LEVEL이 디폴트 동작입니다. 세이브포인트 레벨에 대한 자세한 정보는 SAVEPOINT문 설명에서 ″규칙″ 섹션을 참조하십시오.

LANGUAGE
이 필수 절은 프로시저의 본문이 작성되는 언어 인터페이스 규칙을 지정하는 데 사용됩니다.

C 데이터베이스 관리 프로그램은 C 프로시저인 것처럼 프로시저를 호출합니다. 프로시저는 표준 ANSI C 프로토타입에 의해 정의된대로 C 언어 호출 규칙과 링크 규칙에 따라야 합니다.

JAVA
데이터베이스 관리 프로그램이 Java 클래스의 메소드로 프로시저를 호출합니다.

COBOL
데이터베이스 관리 프로그램이 스토어드 프로시저를 COBOL 프로시저인 것처럼 호출합니다.

CLR
이것은 데이터베이스 관리 프로그램이 .NET 클래스의 한 메소드로 프로시저를 호출하는 것을 의미합니다. 이 때 LANGUAGE CLR은 Windows 운영 체제에서 실행 중인 프로시저에 대해서만 지원됩니다. CLR 루틴에 대해서는 NOT FENCED를 지정할 수 없습니다(SQLSTATE 42601).

OLE
데이터베이스 관리 프로그램은 OLE 자동화 오브젝트에 의해 노출된 메소드인 것처럼 프로시저를 호출합니다. 스토어드 프로시저는 OLE 자동화 데이터 및 호출 메커니즘과 일치해야 합니다. 또한 OLE 자동화 오브젝트는 처리 중인 서버로 구현되어야 합니다(DLL). 이러한 제한사항은 *OLE 자동화 프로그래머 참조서*에 개괄적으로 설명되어 있습니다.

LANGUAGE OLE는 Windows 운영 체제용 DB2에 저장된 프로시저에 대해서만 지원됩니다. THREADSAFE는 LANGUAGE OLE로 정의된 프로시저에 대해서는 지정할 수 없습니다(SQLSTATE 42613).

EXTERNAL
이 절은 CREATE PROCEDURE문이 외부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 링크 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됨을 나타냅니다.

NAME절을 지정하지 않을 경우, 『NAME *procedure-name*』이 사용됩니다. NAME절이 올바르게 형식화되지 않은 경우, 오류가 발생합니다(SQLSTATE 42878).

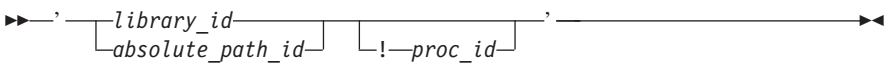
NAME *'string'*
이 절은 정의되는 프로시저를 구현하는 사용자 작성 코드의 이름을 식별합니다.

'string' 옵션은 최대 길이가 254바이트인 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다.

- LANGUAGE C의 경우:

  지정된 *string*은 라이브러리 이름과 이 라이브러리 내의 프로시저로써, 작성 중인 프로시저를 실행하기 위해 데이터베이스 관리 프로그램이 호출합니다. CREATE PROCEDURE문이 수행될 때 라이브러리(및 해당 라이브러리 내

의 프로시저)가 없어도 됩니다. 그러나 프로시저가 호출될 때 라이브러리와 해당 라이브러리 내의 프로시저가 존재해야 하며 데이터베이스 서버 머신에 액세스할 수 있어야 합니다.



이름은 작은따옴표로 묶어야 합니다. 공백은 허용되지 않습니다.

library_id

프로시저를 포함하는 라이브러리 이름을 식별합니다. 데이터베이스 관리 프로그램은 다음과 같이 라이브러리를 찾습니다.

- UNIX 시스템에서 'myfunc'가 *library_id*로 지정되고 데이터베이스 관리 프로그램이 /u/production에서 실행 중인 경우, 데이터베이스 관리 프로그램은 FENCED가 지정된 경우 /u/production/sqllib/function/myproc 라이브러리에서 또는 NOT FENCED가 지정된 경우 /u/production/sqllib/function/unfenced/myproc에서 프로시저를 찾습니다.
- Windows 운영 체제에서 데이터베이스 관리 프로그램은 LIBPATH 또는 PATH 환경 변수에서 지정하는 디렉토리에서 함수를 찾습니다.

이 디렉토리에 있는 스토어드 프로시저는 등록된 속성을 사용하지 않습니다.

absolute_path_id

프로시저의 전체 경로 이름을 식별합니다.

예를 들어, UNIX 시스템에서 '/u/jchui/mylib/myproc'는 데이터베이스 관리 프로그램이 /u/jchui/mylib에서 myproc 프로시저를 찾게 합니다.

Windows 운영 체제에서 'd:₩mylib₩myproc.dll'은 데이터베이스 관리 프로그램이 operatingdirectory에서 myproc.dll 파일을 로드하게 합니다. 절대 경로 ID를 사용하여 루틴 본문을 식별할 경우, .dll 확장자를 추가해야 합니다.

! proc_id

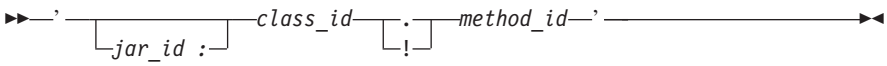
호출될 프로시저의 시작점 이름을 식별합니다. 느낌표(!)는 라이브러리 ID와 프로시저 ID 간 분리 문자의 역할을 합니다. '!proc8'은 데이터베이스 관리 프로그램이 *absolute_path_id*가 지정한 위치에서 라이브러리를 찾아 해당 라이브러리에서 시작점 proc8을 사용하도록 지시합니다.

문자열이 적절히 구성되지 않으면, 오류가 발생합니다(SQLSTATE 42878).

모든 프로시저 본문은 마운트된 디렉토리에 있어야 하며 모든 데이터베이스 파티션에서 사용 가능해야 합니다.

- LANGUAGE JAVA의 경우:

  지정된 *string*에는 선택적 jar 파일 ID, 클래스 ID 및 메소드 ID가 포함됩니다. 이는 작성되는 프로시저를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 것입니다. CREATE PROCEDURE문이 수행될 때 클래스 ID와 방법 ID는 없어도 됩니다. *jar_id*를 지정할 경우, CREATE PROCEDURE 문이 수행될 때 이것이 존재해야 합니다. 그러나 프로시저가 호출될 경우, 클래스 ID와 메소드 ID가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42884).



  이름은 작은따옴표로 묶어야 합니다. 공백은 허용되지 않습니다.

  jar_id

  데이터베이스에 설치될 때, jar 콜렉션에 제공된 jar ID를 식별합니다. 간단한 ID 또는 스키마 규정된 ID일 수 있습니다. 'myJar' 및 'mySchema.myJar'가 그 예입니다.

  class_id

  Java 오브젝트의 클래스 ID를 나타냅니다. 클래스가 패키지의 일부인 경우, 클래스 ID 일부에 전체 패키지 접두어를 포함해야 합니다(예: 'myPacks.StoredProcs'). JVM은 '../myPacks/StoredProcs/' 디렉토리에서 클래스를 찾게 됩니다. Windows 운영 체제에서 JVM은 '..₩myPacks₩StoredProcs₩' 디렉토리에서 찾게 됩니다.

  method_id

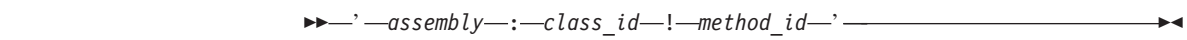
  호출할 Java 클래스를 사용하여 메소드 이름을 식별합니다.

- LANGUAGE CLR의 경우:

  지정된 *string*은 .NET 어셈블리(라이브러리 또는 실행 파일), 해당 어셈블리에 있는 클래스 및 작성되는 프로시저를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 클래스에 있는 메소드를 표시합니다. CREATE PROCEDURE문이 실행될 때 모듈, 클래스 및 메소드는 없어도 됩니다. 그러나 프로시저가 호출될 경우, 모듈, 클래스 및 메소드가 존재해야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42284).

  관리 코드 확장자가 포함된 것을 표시하도록 '/clr' 컴파일러 옵션으로 컴파일된 C++ 루틴은 'LANGUAGE C'가 아닌 'LANGUAGE CLR'로 카탈로그되어야 합니다. DB2는 필요한 런타임 결정을 위해 .NET 인프라스트럭

처가 프로시저에서 활용되고 있는지 알아야 합니다. .NET 인프라스트럭처를 사용하는 모든 프로시저는 'LANGUAGE CLR'로 카탈로그되어야 합니다.



이름은 작은따옴표로 묶어야 합니다. 공백은 허용되지 않습니다.

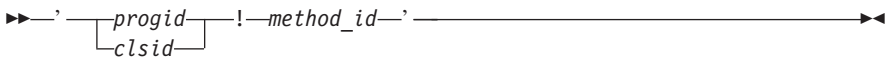
assembly
클래스가 있는 DLL 또는 다른 어셈블리 파일을 식별합니다. 파일 확장자(예: .dll)를 지정해야 합니다. 전체 경로 이름을 지정하지 않은 경우, 파일은 DB2 인스턴스 경로의 함수 디렉토리(예: c:₩DB2₩function)에 상주해야 합니다. 파일이 인스턴스 함수 디렉토리의 서브디렉토리에 상주할 경우 전체 경로를 지정하는 대신에 서브디렉토리를 파일 이름 전에 지정할 수 있습니다. 예를 들어, 인스턴스 디렉토리가 c:₩DB2이고 어셈블리 파일이 c:₩DB2₩function₩myprocs₩mydotnet.dll일 경우 어셈블리에 대해 'myprocs₩mydotnet.dll'만 지정하면 됩니다. 이 매개변수의 대소문자 구분은 파일 시스템의 대소문자 구분과 동일합니다.

class_id
호출될 메소드가 있는 지정된 어셈블리 내에 있는 클래스의 이름을 지정합니다. 클래스가 이름 스페이스에 상주할 경우, 전체 이름 스페이스를 클래스와 같이 지정해야 합니다. 예를 들어, EmployeeClass 클래스가 MyCompany.ProcedureClasses 이름 스페이스에 있으면 클래스에 대해 MyCompany.ProcedureClasses.EmployeeClass를 지정해야 합니다. 일부 .NET 언어용 컴파일러는 프로젝트 이름을 클래스의 이름 스페이스로 추가하는데, 명령행 컴파일러가 사용되었는지 또는 GUI 컴파일러가 사용되었는지에 따라 동작이 다를 수 있습니다. 이 매개변수는 대소문자가 구분됩니다.

method_id
호출될 지정된 클래스 내에 있는 메소드를 지정합니다. 이 매개변수는 대소문자가 구분됩니다.

- LANGUAGE OLE의 경우:

지정된 문자열은 OLE 프로그램 가능 ID(*progid*) 또는 클래스 ID(*clsid*)와 메소드 ID(*method_id*)로서, 명령문에 의해 작성 중인 프로시저를 실행하기 위해 데이터베이스 관리 프로그램이 호출한 것입니다. CREATE PROCEDURE문이 실행될 때 프로그램가능 ID, 클래스 ID 및 메소드 ID는 없어도 됩니다. 그러나 프로시저가 CALL문에서 사용될 경우, 메소드 ID가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42724).



이름은 작은따옴표로 묶어야 합니다. 공백은 허용되지 않습니다.

progid

OLE 오브젝트의 프로그램가능 ID를 식별합니다.

*progid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 시 OLE 자동 제어기로 포워드됩니다. 지정된 OLE 오브젝트는 작성 가능해야 하며 후기 바인딩(IDispatch형 바인딩이라고도 함)을 지원해야 합니다. 규정에 의해, progids의 형식은 다음과 같습니다.

```
<program_name>.<component_name>.<version>
```

이것이 유일한 규정으로 반드시 지켜야 할 규칙은 아니므로, 사실상 *progids*는 다른 형식을 취할 수도 있습니다.

clsid

작성할 OLE 오브젝트의 클래스 ID를 식별합니다. OLE 오브젝트가 progid를 사용하여 등록되지 않은 경우 progid를 지정하기 위한 대안으로 사용할 수 있습니다. *clsid*의 형식은 다음과 같습니다.

```
{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}
```

여기서, 'n'은 영숫자입니다. *clsid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으나, 런타임시 OLE API로 전달됩니다.

method_id

호출할 OLE 오브젝트의 메소드 이름을 식별합니다.

NAME *identifier*

지정된 이 *identifier*는 SQL ID입니다. SQL ID는 문자열에서 *library id*로 사용됩니다. 분리 ID가 아닐 경우, ID는 대문자로 겹쳐집니다. ID에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 형식의 이름은 LANGUAGE C에서만 사용할 수 있습니다.

FENCED 또는 NOT FENCED

이 절은 프로시저가 데이터베이스 관리 프로그램 운영 환경의 프로세스 또는 어드레스 스페이스에서 실행되어도『안전』한지(FENCED) 또는 안전하지 않은지(NOT FENCED) 여부를 지정합니다.

프로시저가 FENCED로 등록된 경우, 데이터베이스 관리 프로그램은 프로시저에서 액세스를 못하도록 내부 자원(예: 데이터 버퍼)을 보호합니다. 모든 프로시저는 FENCED 또는 NOT FENCED로 실행됩니다. 일반적으로 FENCED로 수행되는 프로시저는 NOT FENCED로 실행되는 프로시저와는 다르게 실행됩니다.

주의:

제대로 검사하지 않은 프로시저에 NOT FENCED를 사용하면 DB2 데이터베이스의 무결성이 손상될 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 프로시저가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

프로시저를 NOT FENCED로 등록하려면 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED)이 필요합니다. LANGUAGE OLE 또는 NOT THREADSAFE의 프로시저에 대해서는 FENCED만을 지정할 수 있습니다.

NOT FENCED 절을 지정할 때 LANGUAGE CLR 프로시저를 작성할 수 없습니다(SQLSTATE 42601).

THREADSAFE 또는 NOT THREADSAFE

프로시저가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 아니면 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

프로시저가 OLE가 아닌 다른 LANGUAGE로 정의된 경우에는 다음과 같습니다.

- 프로시저가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 프로시저를 호출할 수 있습니다. 일반적으로 Threadsafe 프로시저가 되려면 프로시저가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED 및 NOT FENCED 프로시저는 둘 다 THREADSAFE될 수 있습니다.
- 프로시저가 NOT THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 동일한 프로세스에서 프로시저를 호출할 수 없습니다.

FENCED 프로시저의 경우, LANGUAGE가 JAVA 또는 CLR이면 THREADSAFE가 디폴트값입니다. 다른 모든 언어에서는 NOT THREADSAFE가 디폴트값입니다. 이 프로시저가 LANGUAGE OLE로 정의된 경우, THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

NOT FENCED 프로시저의 경우 THREADSAFE가 디폴트값입니다. NOT THREADSAFE는 지정될 수 없습니다(SQLSTATE 42613).

COMMIT ON RETURN

프로시저에서 리턴 시 커미트가 발행될지 여부를 표시합니다. 디폴트값은 NO입니다.

NO

프로시저가 리턴할 때 커미트가 발행되지 않습니다.

YES

CALL문이 양수 SQLCODE를 리턴하면 프로시저가 리턴할 때 커미트가 발행됩니다.

커미트 조작은 응용프로그램 프로세스 및 프로시저를 호출하여 수행되는 작업을 포함합니다.

프로시저가 결과 세트를 리턴하는 경우, 결과 세트와 연관된 커서는 커미트 후에 사용할 수 있도록 WITH HOLD로 정의되어야 합니다.

AUTONOMOUS

자체 자동 트랜잭션 범위에서 프로시저를 실행해야 하는지 여부를 표시합니다.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

프로시저가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지(EXTERNAL ACTION) 아니면 취하지 않는지(NO EXTERNAL ACTION) 여부를 지정합니다. 디폴트값은 EXTERNAL ACTION입니다. NO EXTERNAL ACTION을 지정하면, 시스템은 프로시저에 외부적 영향이 없는 것으로 간주하는 특정 최적화를 사용할 수 있습니다.

INHERIT SPECIAL REGISTERS

이 선택적 절은 프로시저에 있는 갱신 가능한 특수 레지스터가 레지스터의 초기값을 호출 명령문의 환경에서 상속하도록 지정합니다.

특수 레지스터에 대한 변경사항은 프로시저 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 갱신할 수 없는 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 디폴트값으로 설정됩니다.

PARAMETER STYLE

이 절은 프로시저로부터 값을 리턴하고 매개변수를 전달하기 위한 규칙을 지정하는 데 사용됩니다.

DB2GENERAL

프로시저가 Java 메소드에서 사용하도록 정의된 매개변수 전달 규칙을 사용합니다. 이는 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

DB2SQL

CALL문의 매개변수 외에도, 다음 인수들이 프로시저에 전달됩니다.

- CALL문의 각 매개변수에 대한 널(NULL) 표시기가 포함된 벡터
- DB2에 리턴할 SQLSTATE
- 프로시저의 규정된 이름
- 프로시저의 특정 이름
- DB2에 리턴할 SQL 진단 문자열

이것은 LANGUAGE C, COBOL, CLR 또는 OLE를 사용할 때만 지정할 수 있습니다.

GENERAL

이는 프로시저가 CALL에서 지정된 매개변수를 수신하는 매개변수 전달 메카

니즘을 사용할 것임을 의미합니다. 매개변수는 언어에 의해 예상된 대로 직접 전달되며, SQLDA 구조는 사용되지 않습니다. 이는 LANGUAGE C, COBOL 또는 CLR을 사용할 때만 지정할 수 있습니다.

널(NULL) 표시기는 프로그램에 직접 전달되지 *않습니다*.

GENERAL WITH NULLS

GENERAL 아래에 지정된 대로 CALL문의 매개변수 외에, 또 다른 인수가 프로시저에 전달됩니다. 이 추가 인수는 CALL문의 각 매개변수에 대해 하나씩 지정되는 널(NULL) 표시기의 벡터로서, C에서는 short integer의 배열이 됩니다. 이는 LANGUAGE C, COBOL 또는 CLR을 사용할 때만 지정할 수 있습니다.

JAVA

프로시저가 Java 언어 및 SQLJ 루틴 스펙을 준수하는 매개변수 전달 규칙을 사용합니다. 값 리턴을 쉽게하기 위해 IN/OUT 및 OUT 매개변수가 단일 항목 배열로 전달됩니다. 이는 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

PARAMETER STYLE JAVA 프로시저는 DBINFO 또는 PROGRAM TYPE 절을 지원하지 않습니다.

SQL

CALL문의 매개변수 외에도, 다음 인수들이 프로시저에 전달됩니다.

- CALL문의 각 매개변수에 대한 널(NULL) 표시기
- DB2에 리턴할 SQLSTATE
- 프로시저의 규정된 이름
- 프로시저의 특정 이름
- DB2에 리턴할 SQL 진단 문자열

이것은 LANGUAGE C, COBOL, CLR 또는 OLE를 사용할 때만 지정할 수 있습니다.

PARAMETER CCSID

프로시저에 전달되거나 프로시저로부터 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII

문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031). 프로시저가 호출될 때 프로시저의 응용프로그램 코드 페이지가 데이터베이스 코드 페이지입니다.

UNICODE

문자열 데이터를 유니코드로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, 문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있습니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, 문자 데이터는 UTF-8로 되어 있습니다. 어떠한 경우라도 프로시저가 호출될 때 프로시저의 응용프로그램 코드 페이지는 1208입니다.

데이터베이스가 유니코드 데이터베이스가 아니고 PARAMETER CCSID UNICODE를 사용하여 프로시저가 작성되는 경우, 프로시저에는 그래픽 유형, XML 유형 또는 사용자 정의 유형이 있을 수 없습니다(SQLSTATE 560C1).PARAMETER CCSID UNICODE 프로시저는 DB2 버전 8.1 이상 클라이언트에서만 호출할 수 있습니다(SQLSTATE 42997).

데이터베이스가 유니코드 데이터베이스가 아니고 데이터베이스 구성에 대체 조합 시퀀스가 지정된 경우, 프로시저는 PARAMETER CCSID ASCII 또는 PARAMETER CCSID UNICODE로 작성될 수 있습니다. 프로시저에 전달되거나 프로시저로부터 전달되는 모든 데이터는 해당 코드 페이지로 변환됩니다.

이 절은 LANGUAGE OLE, LANGUAGE JAVA 또는 LANGUAGE CLR로 지정할 수 없습니다(SQLSTATE 42613).

PROGRAM TYPE

프로시저가 기본 루틴이나 서브루틴 양식의 매개변수를 기대하는지 여부를 지정합니다. 디폴트값은 SUB입니다.

SUB

프로시저는 서로 다른 인수로 전달될 매개변수를 예상합니다.

MAIN

프로시저는 인수 카운터 및 인수 벡터(argc, argv)로서 전달될 매개변수를 기대합니다. 호출될 프로시저의 이름도 ″main″이 되어야 합니다. 이런 유형의 스토어드 프로시저는 독립형 실행 파일이 아닌 공유 라이브러리와 같은 방식으로 내장되어야 합니다. PROGRAM TYPE MAIN은 LANGUAGE절이 C, COBOL 또는 CLR 중 하나를 지정할 경우에만 유효합니다.

DBINFO 또는 NO DBINFO

DB2에 알려진 특정 정보가 추가 호출시간 인수로 호출될 때 프로시저에 전달되는지(DBINFO) 또는 전달되지 않는지(NO DBINFO) 여부를 지정합니다. NO DBINFO가 디폴트값입니다. DBINFO는 LANGUAGE OLE에 대해 지원하지 않습니다(SQLSTATE 42613). 또한 PARAMETER STYLE JAVA 또는 DB2GENERAL에 대해서도 지원되지 않습니다.

DBINFO를 지정하면 다음 정보가 있는 구조가 프로시저에 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름입니다.

- 응용프로그램 ID - 각 데이터베이스 연결에 대해 설정되는 고유한 응용프로그램 ID입니다.
- 응용프로그램 권한 부여 ID - 데이터베이스(SYSTEM_USER 특수 레지스터)에 연결된 사용자의 권한 부여 ID입니다.
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 데이터베이스 버전/릴리스 - 프로시저를 호출하는 데이터베이스 서버의 버전, 릴리스 및 수정 레벨을 나타냅니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.

DBINFO 구조는 모든 외부 루틴에 공통되며, 프로시저와 관련이 없는 추가 필드가 포함되어 있습니다.

SET SESSION AUTHORIZATION 명령문을 사용하여 세션 권한 부여 ID(SESSION_USER 특수 레지스터)를 변경한 경우 응용프로그램 권한 부여 ID는 여전히 SYSTEM_USER 특수 레지스터 값을 리턴합니다.

규칙

- ***자동 루틴 제한사항:*** 자동 루틴은 결과 세트를 리턴할 수 없으며 다음 매개변수 데이터 유형을 지원하지 않습니다(SQLSTATE 428H2).
  - 커서 유형
  - 구조화된 유형
  - XML

  전역 변수는 자동 범위 내에서 참조할 수 없습니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 인덱스를 작성하면, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 부여됩니다.
- NOT FENCED로 정의된 Java 루틴은 FENCED THREADSAFE로 정의된 것처럼 호출됩니다.
- 복합 SQL(인라인된) 명령문 내에서 호출되는 프로시저는 프로시저가 작성될 때 OLD SAVEPOINT LEVEL이 지정되거나 디폴트값이 되더라도 이 프로시저가 NEW SAVEPOINT LEVEL을 지정하면서 작성된 것과 동일하게 실행됩니다.
- PARAMETER STYLE DB2GENERAL절이 지정되는 경우, XML 매개변수는 LANGUAGE JAVA 외부 프로시저에서만 지원됩니다.
- ***디폴트값 설정:*** 디폴트값을 사용하여 정의된 프로시저 매개변수는 프로시저가 호출될 때 디폴트값으로 설정됩니다(프로시저가 호출될 때 대응하는 인수에 값이 제공되지 않거나 값이 DEFAULT로 지정된 경우만).

- *특권*: 프로시저 정의자는 프로시저 삭제 권한은 물론 프로시저에 대한 EXECUTE 특권 WITH GRANT OPTION도 항상 수신합니다. SQL문에 프로시저를 사용할 때는 프로시저 정의자에 프로시저가 사용하는 모든 패키지에 대한 EXECUTE 특권이 있어야 합니다.
- *호환성*: z/OS용 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - ASUTIME NO LIMIT
    - NO COLLID
    - STAY RESIDENT NO
    - 유니코드 데이터베이스의 경우 CCSID UNICODE
    - CCSID ASCII(PARAMETER CCSID UNICODE가 지정되지 않은 경우 비 유니코드 데이터베이스)

  이전 버전 DB2와의 호환성을 위해,
  - DYNAMIC RESULT SETS 대신 RESULT SETS를 지정할 수 있습니다.
  - CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.
  - DB2GENERAL 대신 DB2GENRL을 지정할 수 있습니다.
  - GENERAL 대신 SIMPLE CALL을 지정할 수 있습니다.
  - GENERAL WITH NULLS 대신 SIMPLE CALL WITH NULLS를 지정할 수 있습니다.
  - PARAMETER STYLE DB2DARI가 지정됩니다.

예:

예 1: 부품 번호를 받아서 부품을 비용 및 현재 사용 가능한 수량을 전달하는 Java로 작성된 프로시저에 대한 프로시저 정의를 작성합니다.

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
    OUT COST DECIMAL(7,2),
    OUT QUANTITY INTEGER)
EXTERNAL NAME 'parts.onhand'
LANGUAGE JAVA PARAMETER STYLE JAVA
```

예 2: 어셈블리 번호를 전달하며 어셈블리를 구성하는 부품 수, 부품의 총 비용 그리고 부품 번호, 수량, 각 부품의 단가를 나열하는 결과 세트를 리턴하는 C로 작성된 프로시저에 대해 프로시저 정의를 작성합니다.

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN ASSEMBLY_NUM INTEGER,
    OUT NUM_PARTS INTEGER,
    OUT COST DOUBLE)
  EXTERNAL NAME 'parts!assembly'
  DYNAMIC RESULT SETS 1 NOT FENCED
  LANGUAGE C PARAMETER STYLE GENERAL
```

CREATE PROCEDURE(전래)

CREATE PROCEDURE(전래) 명령문은 다른 프로시저(*소싱된 프로시저*)에 기초한 프로시저(*소싱된 프로시저*)를 등록합니다. 페더레이티드 시스템에서 *페더레이티드 프로시저*는 해당 소싱된 프로시저가 지원되는 데이터 소스에 있는 소싱된 프로시저입니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

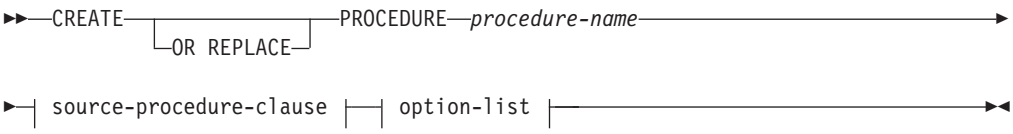
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 프로시저의 스키마 이름이 기존의 스키마를 참조하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 프로시저의 스키마 이름이 기존 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

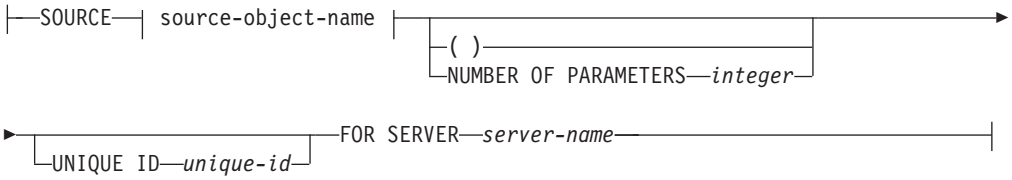
사용자 맵핑이 필요한 데이터 소스의 경우 명령문의 권한 부여 ID가 데이터 소스에 보유하고 있는 특권은 리모트 카탈로그 테이블에서 프로시저의 설명을 선택할 수 있는 특권을 포함해야 합니다.

기존 프로시저를 교체하려면 명령문의 권한 부여 ID가 기존 프로시저의 소유자여야 합니다(SQLSTATE 42501).

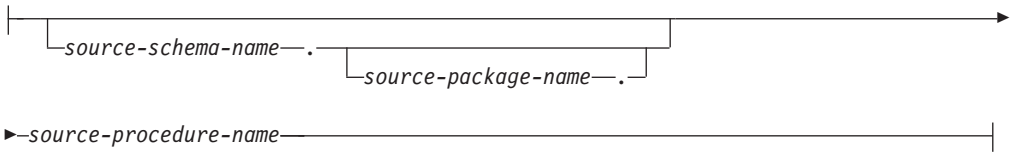
구문



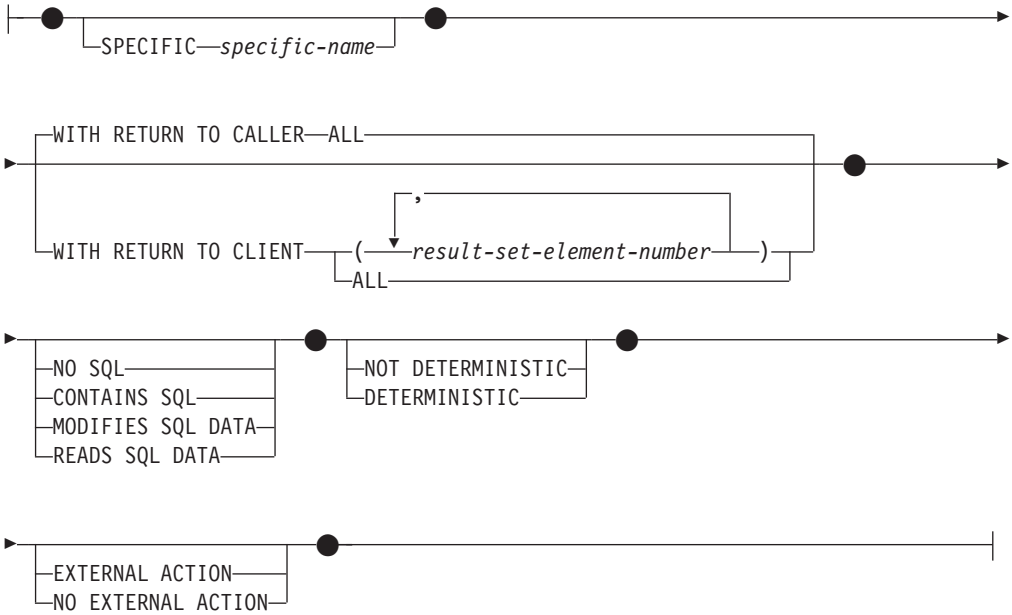
source-procedure-clause:



source-object-name:



option-list:



설명

OR REPLACE

프로시저의 정의가 현재 서버에 존재하는 경우 정의를 교체할 것을 지정합니다. 기존 정의는 카탈로그에서 새 정의가 교체되기 전에 효율적으로 삭제됩니다. 예외로, 프로시저에 대한 권한이 부여된 특권에는 영향이 미치지 않습니다. 이 옵션은 프로시저의 정의가 현재 서버에 존재하지 않는 경우 무시됩니다. 기존 프로시저를 교체하려면 새 정의의 특정 이름 및 프로시저 이름이 이전 정의의 특정 이름 및 프로시저 이름과 동일하거나, 새 정의의 서명이 이전 정의의 서명과 일치해야 합니다. 그렇지 않으면 새 프로시저가 작성됩니다.

procedure-name

소싱된 프로시저가 정의되고 있는 이름. 이는 프로시저를 지정하는 규정화되거나 규정되지 않은 이름입니다. *procedure-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 128)입니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER

프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다.

매개변수 수와 함께 내재적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 기술된 프로시저를 식별해서는 안됩니다(SQLSTATE 42723). 매개변수 수와 함께 규정되지 않은 이름은 스키마 중에서 고유하지 않아도 됩니다.

두 부분으로 구성된 이름을 지정할 경우, *schema-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

페더레이티드 시스템에서 *procedure-name*은 페더레이티드 서버에 있는 프로시저의 이름입니다.

SOURCE *source-object-name*

정의 중인 프로시저에서 사용되는 소싱된 프로시저를 지정합니다. 페더레이티드 시스템에서 소싱된 프로시저는 지원되는 데이터 소스에 있는 프로시저 입니다.

source-schema-name

소싱된 프로시저의 스키마 이름을 식별합니다. 스키마 이름을 사용하여 소싱된 프로시저를 식별하는 경우 CREATE PROCEDURE(전래)문에 *source-schema-name*을 지정해야 합니다. *source-schema-name*에 특수 문자나 소문자가 있는 경우 이를 큰따옴표로 묶어야 합니다.

source-package-name

소싱된 프로시저의 패키지 이름을 식별합니다. *source-package-name*은 Oracle 데이터 소스에만 적용됩니다. 패키지 이름을 사용하여 소싱된 프로시저를 식별하는 경우 CREATE PROCEDURE(전래)문에 *source-package-name*을 지정해야 합니다. *source-package-name*에 특수 문자나 소문자가 있는 경우 이를 큰따옴표로 묶어야 합니다.

source-procedure-name

소싱된 프로시저의 프로시저 이름을 식별합니다. *source-procedure-name*에 특수 문자나 소문자가 있는 경우 이를 큰따옴표로 묶어야 합니다.

( )

매개변수 수가 0임을 표시합니다.

NUMBER OF PARAMETERS *integer*

소싱된 프로시저에 대한 매개변수 수를 지정합니다. *integer*의 최소값은 0이고 최대값은 32 767입니다.

UNIQUE ID *string-constant*

데이터 소스에 동일한 이름, 스키마 및 매개변수 수를 갖는 프로시저가 여러 개 있을 때 소싱된 프로시저를 고유하게 식별하는 방법을 제공합니다. 최대 길이가 128 문자인 *string-constant* 값은 각 데이터 소스별로 고유하게 식별됩니다.

FOR SERVER *server-name*

CREATE SERVER문을 사용하여 등록된 서버 정의를 지정합니다.

SPECIFIC *specific-name*

정의되고 있는 소싱된 프로시저의 인스턴스에 대한 고유 이름을 제공합니다. 이 특정 이름은 소싱된 프로시저 또는 소싱된 프로시저의 주석 표시를 삭제할 때 사용할 수 있습니다. 소싱된 프로시저를 호출하는 데는 이 이름을 사용할 수 없습니다. *specific-name*의 규정되지 않은 양식은 최대 길이가 18인 SQL ID입니다. *specific-name*의 규정된 양식은 뒤에 마침표와 SQL ID가 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함하는 *specific-name* 값은 응용프로그램 서버에 존재하는 다른 프로시저 인스턴스를 식별해서는 안됩니다. 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 42710).

*specific-name*은 기존의 *procedure-name*과 같을 수 있습니다.

규정자를 지정하지 않으면, *procedure-name*에 대해 사용된 규정자가 사용됩니다. 큐정자를 지정하면 *procedure-name*의 명시적 또는 내재적 규정자와 동일해야 합니다. 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 뒤에 문자 시간소인이 오는 'SQL'입니다('SQL*yymmddhhmmssxxx*').

WITH RETURN TO CALLER 또는 WITH RETURN TO CLIENT

소싱된 프로시저의 결과 세트가 처리되는 위치를 표시합니다. 소싱된 프로시저가 Oracle 데이터 소스가 아닌 경우, 하나의 결과 세트만이 호출자 또는 클라이언트에게 리턴됩니다. 소싱된 프로시저가 두 개 이상의 결과 세트를 리턴하도록 코딩된 경우, 호출자 또는 클라이언트에게는 첫 번째 결과 세트만 리턴됩니다. 디폴트값은 WITH RETURN TO CALLER입니다.

WITH RETURN TO CALLER ALL

소싱된 프로시저의 모든 결과 세트가 호출자에게 리턴되도록 지정합니다.

WITH RETURN TO CLIENT

클라이언트 응용프로그램으로 직접 리턴되는 소싱된 프로시저의 결과 세트를 표시합니다. 데이터 소스의 동적 결과 세트 값은 리턴될 결과 세트의 0보다 커야 합니다.

(result-set-element-number, ...)

결과 세트의 비어 있지 않은 목록을 클라이언트 응용프로그램으로 리턴하도록 지정합니다(SQLSTATE 42601). *result-set-element-number*가 결과 세트가 리턴되는 순서를 기반으로 결과 세트를 식별합니다. 여기에서 1은 첫 번째 결과 세트를 식별하고 2는 두 번째 결과 세트를 식별하는 식으로 진행됩니다. 총 결과 세트 수보다 큰 *result-set-element-number*는 무시됩니다. 각 *result-set-element-number*는 0보다 큰 정수값이어야 하며

(SQLSTATE 42815), 작은 정수 상수의 값을 초과하면 안 됩니다 (SQLSTATE 42820). 클라이언트 응용프로그램에 리턴할 결과 세트의 목록에는 중복 값이 포함될 수 없으며 오름 차순으로 지정되어야 합니다 (SQLSTATE 42815). 결과 세트는 항상 소싱된 프로시저에서 리턴되는 순서대로 처리됩니다.

클라이언트 응용프로그램에 리턴할 목록에 식별되지 않은 결과 세트는 호출자에게 리턴됩니다.

주: 클라이언트 응용프로그램에 리턴할 이 결과 세트 목록은 알려진 소싱된 프로시저와 함께 사용될 때만 실행될 때마다 결과 세트 목록의 같은 위치에 있는 클라이언트의 결과 세트를 일관되게 리턴합니다. 프로시저의 내부 논리에 따라 소싱된 프로시저가 실행될 때마다 다른 결과 세트를 리턴할 수 있습니다. 이것이 case이면 WITH RETURN TO CALLER ALL 또는 WITH RETURN TO CLIENT ALL을 대신 지정하고 이 case를 처리하도록 응용프로그램을 코드하십시오.

ALL

소싱된 프로시저의 모든 결과 세트가 클라이언트에게 리턴되도록 지정합니다.

NO SQL, CONTAINS SQL, MODIFIES SQL DATA, READS SQL DATA

소싱된 프로시저에 포함된 SQL문에 대한 데이터 액세스 레벨을 나타냅니다. 소싱된 프로시저에 대한 소싱된 프로시저가 페더레이티드 서버에 위치하지 않으므로 데이터 소스에서 소싱된 프로시저를 실행하는 동안 지정된 레벨이 강제실행되지 않습니다. 소싱된 프로시저에 대해 지정된 레벨과 데이터 소스에서 소싱된 프로시저가 실제로 수행하는 레벨에 차이가 있으면 데이터 불일치가 발생할 수 있습니다. 이 옵션을 명시적으로 지정하지 않으면 소싱된 프로시저에 대한 값을 사용합니다. 데이터 소스에서 이 옵션을 사용할 수 없는 경우 디폴트값은 MODIFIES SQL DATA입니다. 이 옵션을 명시적으로 지정했지만 소싱된 프로시저에 대한 값과 일치하지 않으면 오류가 리턴됩니다(SQLSTATE 428GS).

DETERMINISTIC 또는 NOT DETERMINISTIC

소싱된 프로시저가 항상 주어진 인수 값에 대해 동일한 결과를 리턴하는지 (DETERMINISTIC) 또는 소싱된 프로시저가 결과에 영향을 주는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. DETERMINISTIC 소싱된 프로시저는 항상 동일한 입력으로 연속되는 호출로부터 동일한 결과를 리턴해야 합니다. 이 절은 현재 프로시저의 처리에는 영향을 주지 않습니다. 이 옵션을 명시적으로 지정하지 않으면 소싱된 프로시저에 대한 값을 사용합니다. 데이터 소스에서 이 옵션을 사용할 수 없는 경우 디폴트값은 NOT DETERMINISTIC입니다. 이 옵션을 명시적으로 지정했지만 소싱된 프로시저에 대한 값과 일치하지 않으면 오류가 리턴됩니다(SQLSTATE 428GS).

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

소싱된 프로시저가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 몇 가지 조치를 취하는지(EXTERNAL ACTION) 또는 취하지 않는지(NO EXTERNAL ACTION)를 지정합니다. NO EXTERNAL ACTION절을 지정할 경우 페더레이티드 데이터베이스는 소싱된 프로시저가 외부 영향을 받지 않는다는 가정하에 최적화를 사용합니다. 이 옵션을 명시적으로 지정하지 않으면 소싱된 프로시저에 대한 값을 사용합니다. 데이터 소스에서 이 옵션을 사용할 수 없는 경우 디폴트값은 EXTERNAL ACTION입니다. 이 옵션을 명시적으로 지정했지만 소싱된 프로시저에 대한 값과 일치하지 않으면 오류가 리턴됩니다(SQLSTATE 428GS).

규칙

- *source-object-name*이 NUMBER OF PARAMETERS 및 UNIQUE ID절을 사용하여 데이터 소스에서 프로시저를 식별하지 않을 경우 오류가 리턴됩니다(SQLSTATE 42883). 프로시저가 두 개 이상 식별된 경우에도 오류가 리턴됩니다. (SQLSTATE 42725).
- UNIQUE ID절을 지정했지만 데이터 소스가 고유 ID를 지원하지 않을 경우 오류가 리턴됩니다(SQLSTATE 42883).

주

- 페더레이티드 프로시저를 데이터 소스에 등록하려면 먼저 해당 데이터 소스에 액세스하도록 페더레이티드 서버를 구성해야 합니다. 이 구성에는 데이터 소스에 랩퍼 등록, 데이터 소스에 대한 서버 정의 작성, 그리고 사용자 맵핑이 필요한 데이터 소스에 대한 데이터 소스 서버와 페더레이티드 서버 사이의 사용자 맵핑 작성이 포함됩니다.
- ***처음에 유효하지 않은 프로시저 작성:*** 프로시저 본문에서 참조된 오브젝트가 존재하지 않거나 유효하지 않은 것으로 표시된 경우 또는 정의자가 오브젝트에 액세스할 수 있는 특권을 일시적으로 갖고 있지 않고 데이터베이스 구성 매개변수 **auto_reval**이 DISABLED로 설정되지 않은 경우, 프로시저는 계속 성공적으로 작성됩니다. 프로시저는 유효하지 않은 것으로 표시되고 다음에 호출될 때 유효성이 다시 확인됩니다.
- 페더레이티드 서버에 정의된 SQL 및 외부 프로시저와는 달리 페더레이티드 프로시저는 해당 *remote-object-name*이 DB2 데이터 소스에 있는 프로시저를 참조하더라도 호출자의 특수 레지스터를 상속하지 않습니다.
- 소싱된 프로시저의 정의가 변경되면(예를 들어, 매개변수 데이터 유형이 변경됨) 페더레이티드 프로시저를 삭제하고 재작성해야 합니다. 그렇지 않으면 페더레이티드 프로시저를 호출할 때 오류가 발생할 수 있습니다.
- 소싱된 프로시저 매개변수의 길이가 128을 초과할 경우 페더레이티드 프로시저의 매개변수 이름은 128바이트로 잘려집니다.

- *호환성*: 스토어드 프로시저 별칭 작성에 대한 DataJoiner® 구문이 지원되지 않습니다. 새 버전 9의 구문에서는 매개변수 유형 맵핑이 별칭과 유사하게 처리되며, 카탈로그 찾아보기가 리모트 데이터 유형을 판별합니다. 로컬 매개변수 유형은 정방향 유형 맵핑을 통해 판별됩니다.

예:

예 1: 페더레이티드 서버 S1에서 리모트 스키마 이름 USER1, 리모트 패키지 이름 P1을 사용하고 결과 세트를 클라이언트로 리턴하는 Oracle 프로시저 EMPLOYEE에 대한 페더레이티드 프로시저 FEDEMPLOYEE를 작성하십시오.

```
CREATE PROCEDURE FEDEMPLOYEE SOURCE USER1.P1.EMPLOYEE
  FOR SERVER S1 WITH RETURN TO CLIENT ALL
```

예 2: 페더레이티드 서버 S1에서 리모트 스키마 이름 USER1, 리모트 패키지 이름 P1을 사용하고 첫 번째 및 세 번째 결과 세트를 클라이언트에게 리턴하고 나머지 결과 세트는 호출자에게 리턴하는 Oracle 프로시저 SALARYSTAT에 대한 페더레이티드 프로시저 FEDSALARYSTAT를 작성하십시오.

```
CREATE OR REPLACE PROCEDURE FEDSALARYSTAT SOURCE USER1.P1.SALARYSTAT
FOR SERVER S1 WITH RETURN TO CLIENT(1,3)
```

CREATE PROCEDURE(SQL)

CREATE PROCEDURE(SQL)문은 현재 서버에 SQL 프로시저를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

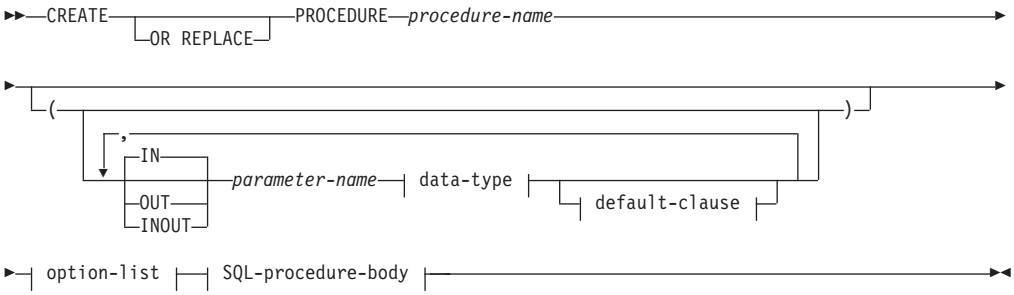
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 프로시저의 내재적 또는 명시적 스키마 이름이 없는 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 프로시저의 스키마 이름이 기존 스키마를 참조하는 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

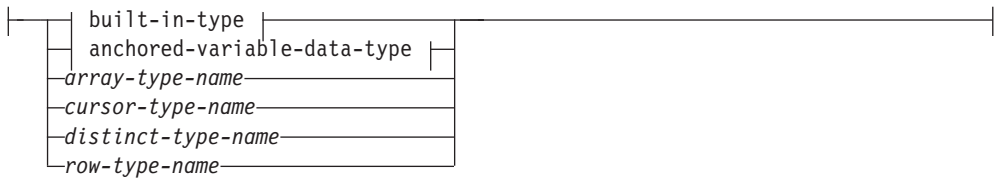
명령문의 권한 부여 ID가 보유한 특권에는 프로시저 본문에 지정된 SQL문을 호출하는 데 필요한 모든 권한도 포함되어야 합니다.

기존 프로시저를 교체하려면 명령문의 권한 부여 ID가 기존 프로시저의 소유자여야 합니다(SQLSTATE 42501).

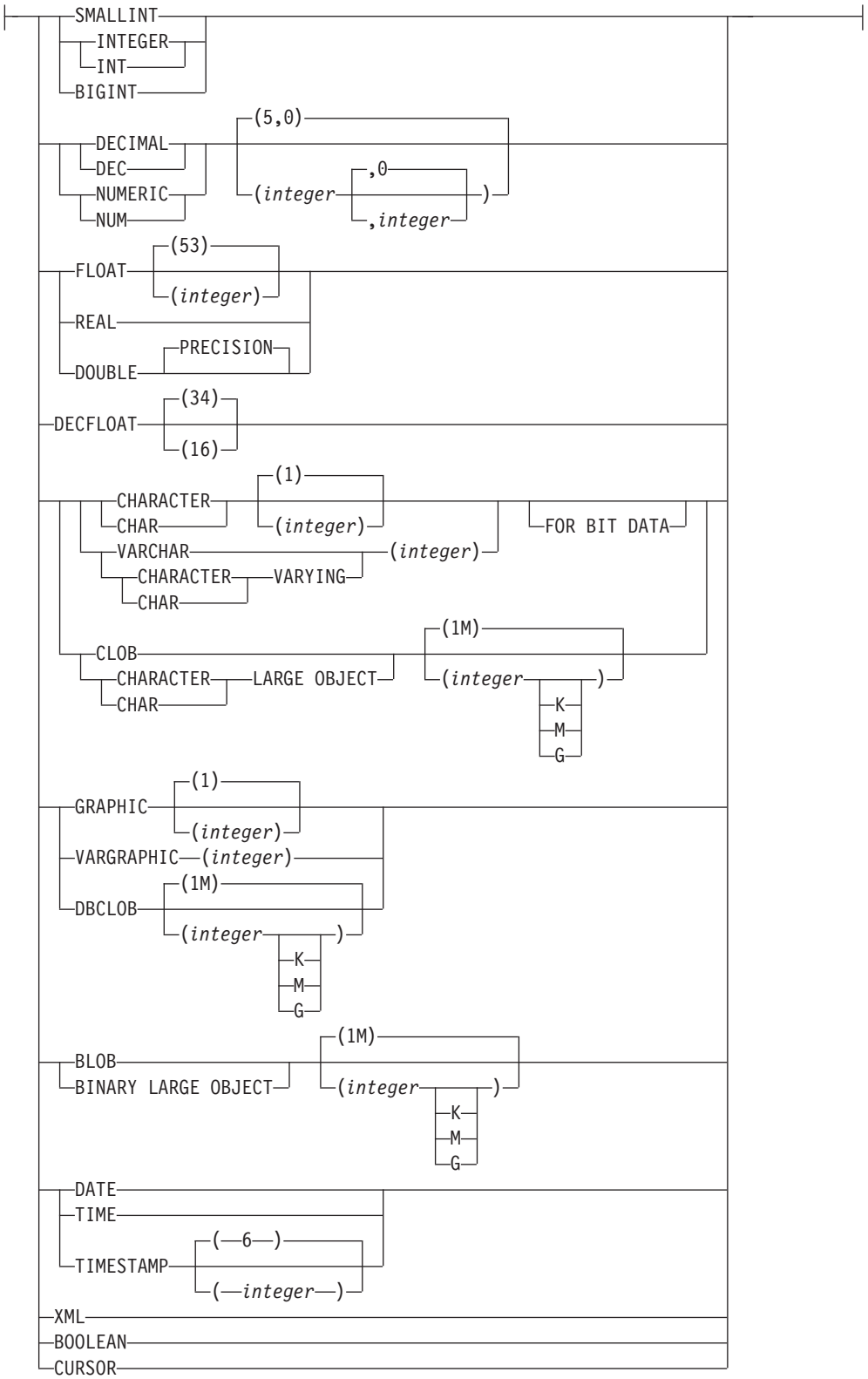
구문



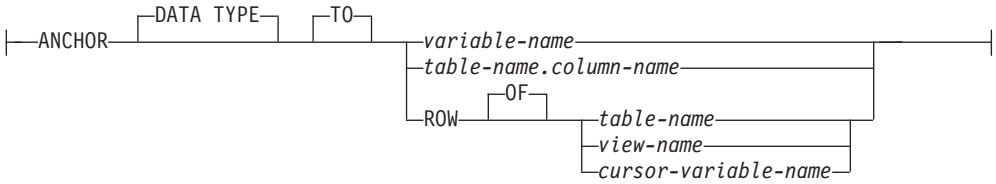
data-type:



built-in-type:

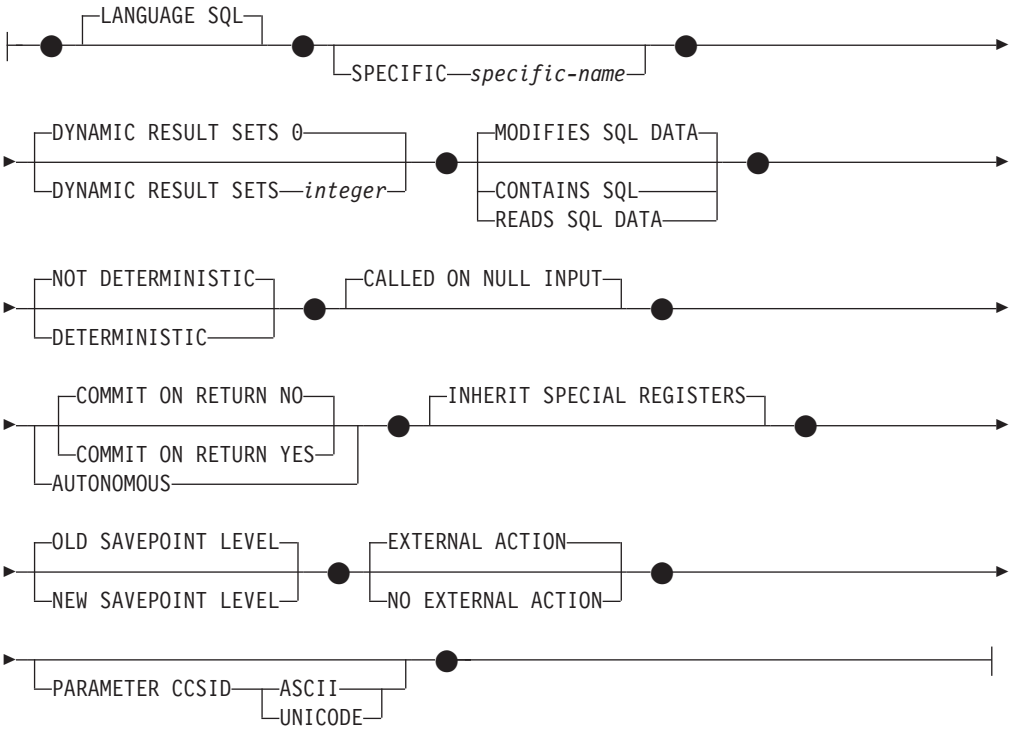


anchored-data-type:

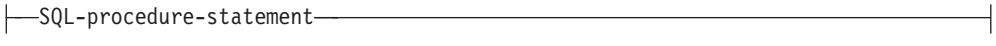


default-clause:

option-list:



SQL-procedure-body:



설명

OR REPLACE

프로시저 정의가 현재 서버에 존재하는 경우 이를 교체하도록 지정합니다. 프로시

저에 대해 부여된 특권에 영향을 주지 않는다는 점을 제외하고 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 효과적으로 삭제됩니다. 프로시저 정의가 현재 서버에 없으면 이 옵션은 무시됩니다. 기존 프로시저를 교체하려면, 새 정의의 특정 이름 및 프로시저 이름이 이전 정의의 특정 이름 및 프로시저 이름과 동일하거나 새 정의의 시그니처가 이전 정의의 시그니처와 일치해야 합니다. 그렇지 않으면, 새 프로시저가 작성됩니다.

procedure-name

정의되고 있는 프로시저에 이름을 지정하십시오. 이는 프로시저를 지정하는 규정화되거나 규정되지 않은 이름입니다. *procedure-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 128)입니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name* 입니다.

매개변수 수와 더불어 내재적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 기술된 프로시저를 식별해서는 안됩니다(SQLSTATE 42723). 매개변수 수를 비롯한 규정되지 않은 이름이 해당 스키마 내에서는 고유해야 하지만 여러 스키마에서는 고유하지 않아도 됩니다.

두 부분으로 구성된 이름이 지정된 경우, *schema-name*은 'SYS'로 시작할 수 없습니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42939).

*(***IN | OUT | INOUT** *parameter-name data-type default-clause,...)*

프로시저 매개변수를 식별하고 각 매개변수의 모드, 이름, 데이터 유형 및 선택적 디폴트값을 지정합니다. 프로시저에 예상되는 각 매개변수에 대해 목록에 있는 한 항목이 지정되어야 합니다.

매개변수가 없는 프로시저를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예를 들면, 다음과 같습니다.

```
CREATE PROCEDURE SUBWOOFER() ...
```

한 스키마 내에서 동일하게 이름이 지정된 두 프로시저는 같은 수의 매개변수를 가질 수 없습니다. 시그니처가 중복되면 SQL 오류가 발생합니다(SQLSTATE 42723).

예를 들어, 다음 명령문이 제공되면, 다음과 같습니다.

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

데이터 유형은 달라도 프로시저에 있는 매개변수 수가 동일하므로 두 번째 명령문은 실패합니다.

IN | OUT | INOUT

매개변수의 모드를 지정합니다.

프로시저에서 오류가 리턴되면 OUT 매개변수의 정의가 취소되고 INOUT 매개변수는 변경되지 않습니다.

IN 매개변수를 프로시저에 대한 입력 매개변수로 식별합니다. 프로시저 내에서 이뤄진 매개변수 변경사항은 제어가 리턴될 때 호출 SQL 응용프로그램에서 사용할 수 없습니다. 디폴트값은 IN입니다.

OUT 매개변수를 프로시저에 대한 출력 매개변수로 식별합니다.

INOUT
매개변수를 프로시저에 대한 입력 매개변수와 출력 매개변수로 식별합니다.

parameter-name
매개변수의 이름을 지정합니다. 이 매개변수 이름은 프로시저에 대해 고유해야 합니다(SQLSTATE 42734).

data-type
매개변수의 데이터 유형을 지정합니다. 구조화된 유형 또는 참조 유형을 지정할 수 없습니다(SQLSTATE 429BB).

built-in-type
내장 데이터 유형을 지정합니다. BOOLEAN 및 CURSOR를 제외한 각 내장 데이터 유형에 대한 자세한 설명은 “CREATE TABLE”을 참조하십시오.

BOOLEAN
부울

CURSOR
기반 커서에 대한 참조.

anchored-data-type
데이터 유형을 정의하는 데 사용되는 다른 오브젝트를 식별합니다. 고정 오브젝트의 데이터 유형은 데이터 유형을 직접 지정하기 위해, 또는 행의 경우 행 유형을 작성하기 위해 적용되는 동일한 제한사항을 수반합니다.

ANCHOR DATA TYPE TO
데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name
전역 변수를 식별합니다. 전역 변수의 데이터 유형은 *parameter-name*의 데이터 유형으로 사용됩니다.

table-name.column-name
기존 테이블이나 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 *parameter-name*의 데이터 유형으로 사용됩니다.

ROW OF *table-name* 또는 *view-name*

*table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뷰의 컬럼 이름 및 컬럼 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. *parameter-name*의 데이터 유형은 unnamed row 자료형입니다.

ROW OF *cursor-variable-name*

*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 지정된 커서 변수는 다음 중 하나여야 합니다 (SQLSTATE 428HS).

- 강하게 유형 지정된 커서 데이터 유형이 있는 전역 변수
- 모든 결과 컬럼이 이름 지정된 *select-statement*를 지정하는 CONSTANT절로 작성되거나 선언되어 약하게 유형 지정된 커서 데이터 유형이 있는 전역 변수

커서 변수의 커서 유형이 named row 자료형을 사용하여 강하게 유형 지정되지 않은 경우, *parameter-name*의 데이터 유형은 unnamed row 자료형입니다.

array-type-name

사용자 정의 배열 유형의 이름을 지정합니다. 스키마 이름 없이 *array-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 배열 유형이 분석됩니다.

cursor-type-name

커서 유형의 이름을 지정합니다. 스키마 이름 없이 *cursor-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 커서 유형이 분석됩니다.

distinct-type-name

구별 유형의 이름을 지정합니다. 매개변수의 길이, 정밀도 및 스케일은 각각 구별 유형의 소스 유형에 대한 길이, 정밀도 및 스케일입니다. 구별 유형 매개변수는 구별 유형의 소스 유형으로 전달됩니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

row-type-name

사용자 정의 행 유형의 이름을 지정합니다. 매개변수의 필드는 행 유형의 필드입니다. 스키마 이름 없이 *row-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 행 유형이 분석됩니다.

DEFAULT

매개변수의 디폴트값을 지정합니다. 디폴트값은 상수, 특수 레지스터, 전역 변수, 표현식 또는 NULL 키워드일 수 있습니다. 디폴트값으로 지정할 수 있는 특수

레지스터는 컬럼 디폴트값에 지정할 수 있는 특수 레지스터와 동일합니다 (CREATE TABLE문의 *default-clause* 참조). 표현식을 사용하여 기타 특수 레지스터를 디폴트값으로 지정할 수 있습니다.

*expression*은 『표현식』에 설명된 표현식 유형일 수 있습니다. 디폴트값을 지정하지 않으면, 매개변수가 디폴트값을 갖지 않으며 프로시저 호출 시 대응하는 인수를 생략할 수 없습니다. *expression*의 최대 크기는 64K바이트입니다.

디폴트 표현식은 SQL 데이터를 수정하지 않거나(SQLSTATE 428FL 또는 SQLSTATE 429BL) 외부 조치를 수행하지 않아야 합니다(SQLSTATE 42845). 매개변수 데이터 유형과 호환 가능한 표현식을 지정해야 합니다(SQLSTATE 42821).

다음 상황에서는 디폴트값을 지정할 수 없습니다.

- INOUT 또는 OUT 매개변수의 경우(SQLSTATE 42601)
- ARRAY, ROW 또는 CURSOR 매개변수 유형의 경우(SQLSTATE 429BB)

디폴트값이 없는 매개변수를 디폴트값이 있는 매개변수 뒤에 정의할 수 없습니다(SQLSTATE 428HG).

SPECIFIC *specific-name*

정의할 프로시저의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 프로시저를 삭제할 때 또는 프로시저에 주석을 붙일 때 사용할 수 있지만, 프로시저를 호출하는 데에는 사용할 수 없습니다. *specific-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 18)입니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다. 내재적 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버에 존재하는 다른 프로시저 인스턴스를 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *procedure-name*과 같을 수 있습니다.

규정자를 지정하지 않으면, *procedure-name*에 대해 사용된 규정자가 사용됩니다. 규정자를 지정하면, 규정자는 *procedure-name*의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 뒤에 문자 시간소인이 오는 'SQL'입니다 ('SQL*yymmddhhmmssxxx*').

DYNAMIC RESULT SETS *integer*

프로시저에 대한 리턴된 결과 세트의 측정된 상위 바운드를 표시하십시오.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

프로시저에 포함된 SQL문에 대한 데이터 액세스 레벨을 나타냅니다.

CONTAINS SQL

SQL 데이터를 읽지도 수정하지도 않는 SQL문이 프로시저에 의해 실행될 수

있음을 나타냅니다(SQLSTATE 38004 또는 42985). 프로시저에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

READS SQL DATA
SQL 데이터를 수정하지 않는 일부 SQL문이 이 프로시저에 포함될 수 있음을 나타냅니다(SQLSTATE 38002 또는 42985). 프로시저에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

MODIFIES SQL DATA
프로시저에서 지원되지 않는 명령문을 제외하고, 프로시저가 SQL문을 실행할 수 있음을 나타냅니다(SQLSTATE 38003 또는 42985).

BEGIN ATOMIC절이 복합 SQL 프로시저에서 사용될 경우, MODIFIES SQL DATA로 정의된 경우에만 프로시저를 작성할 수 있습니다.

DETERMINISTIC 또는 NOT DETERMINISTIC
이 절은 프로시저가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 프로시저가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 프로시저는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다.

이 절은 현재 프로시저의 처리에는 영향을 주지 않습니다.

CALLED ON NULL INPUT
CALLED ON NULL INPUT은 항상 프로시저에 적용됩니다. 즉, 프로시저는 널(NULL) 인수가 있는지 여부에 상관없이 호출됩니다. 모든 OUT 또는 INOUT 매개변수는 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 널(NULL) 인수 값에 대한 테스트 책임은 프로시저에 있습니다.

COMMIT ON RETURN
프로시저에서 리턴 시 커미트가 발행될지 여부를 표시합니다. 디폴트값은 NO입니다.

NO
프로시저가 리턴할 때 커미트가 발행되지 않습니다.

YES
CALL문이 양수 SQLCODE를 리턴하면 프로시저가 리턴할 때 커미트가 발행됩니다.

커미트 조작은 응용프로그램 프로세스 및 프로시저를 호출하여 수행되는 작업을 포함합니다.

프로시저가 결과 세트를 리턴하는 경우, 결과 세트와 연관된 커서는 커미트 후에 사용할 수 있도록 WITH HOLD로 정의되어야 합니다.

AUTONOMOUS

자체 자동 트랜잭션 범위에서 프로시저를 실행해야 하는지 여부를 표시합니다.

INHERIT SPECIAL REGISTERS

이 선택적 절은 프로시저에 있는 갱신 가능한 특수 레지스터가 레지스터의 초기값을 호출 명령문의 환경에서 상속하도록 지정합니다. 중첩 오브젝트(예: 트리거 또는 뷰)에서 호출되는 루틴의 경우에는 초기값을 오브젝트 정의에서 상속하지 않고 런타임 환경에서 상속합니다.

특수 레지스터에 대한 변경사항은 프로시저 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 갱신할 수 없는 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 디폴트값으로 설정됩니다.

OLD SAVEPOINT LEVEL 또는 NEW SAVEPOINT LEVEL

이 프로시저가 세이브포인트 이름 및 효과에 대한 새 세이브포인트 레벨을 작성하는지의 여부를 지정합니다. OLD SAVEPOINT LEVEL이 디폴트 동작입니다. 세이브포인트 레벨에 대한 자세한 내용은 『SAVEPOINT』의 『규칙』을 참조하십시오.

LANGUAGE SQL

이 절은 프로시저 본문이 SQL 언어로 작성되도록 지정하는 데 사용됩니다.

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

프로시저가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지(EXTERNAL ACTION) 아니면 취하지 않는지(NO EXTERNAL ACTION) 여부를 지정합니다. 디폴트값은 EXTERNAL ACTION입니다. NO EXTERNAL ACTION을 지정하면, 시스템은 프로시저에 외부적 영향이 없는 것으로 간주하는 특정 최적화를 사용할 수 있습니다.

PARAMETER CCSID

프로시저에 전달되거나 프로시저로부터 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII

문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031).

UNICODE

문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있도록 지정합니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE를 지정할 수 없습니다(SQLSTATE 56031).

SQL-procedure-body

SQL 프로시저의 본문인 SQL문을 지정합니다.

『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

규칙

- ***자동 루틴 제한사항:*** 자동 루틴은 결과 세트를 리턴할 수 없으며 다음을 지원하지 않습니다(SQLSTATE 428H2).
  - 사용자 정의 커서 유형
  - 사용자 정의 구조화 유형
  - IN, OUT 및 INOUT 매개변수로서의 XML

  세션 변수는 자동 범위 내에서 참조될 수 없습니다.
- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.
- ***커서 및 행 유형 사용:*** 매개변수에 대해 커서 유형이나 행 유형을 사용하는 프로시저는 복합 SQL문(컴파일된) (SQLSTATE 428H2)내에서만 호출할 수 있습니다(커서 유형을 포함하는 OUT 매개변수를 사용하여 프로시저를 호출할 수 있는 JDBC는 제외).

주

- 아직 존재하지 않는 스키마 이름을 사용하여 테이블을 작성할 경우 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 갖고 있으면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 복합 SQL(인라인된) 명령문 안에서 호출되는 프로시저는 OLD SAVEPOINT LEVEL이 지정되었거나 프로시저가 작성될 때 디폴트로 지정되는 경우에도 NEW SAVEPOINT LEVEL을 지정하여 작성된 것처럼 실행됩니다.
- ***처음에 유효하지 않은 프로시저 작성:*** 프로시저 본문에서 참조된 오브젝트가 존재하지 않거나 유효하지 않은 것으로 표시된 경우 또는 정의자가 오브젝트에 액세스할 수 있는 특권을 일시적으로 갖고 있지 않고 데이터베이스 구성 매개변수 **auto_reval**이 DISABLED로 설정되지 않은 경우, 프로시저는 계속 성공적으로 작성됩니다. 프로시저는 유효하지 않은 것으로 표시되며 다음 번에 호출될 때 유효성이 다시 확인됩니다.
- ***디폴트값 설정:*** 디폴트값을 사용하여 정의된 프로시저 매개변수는 프로시저가 호출될 때 디폴트값으로 설정됩니다(프로시저가 호출될 때 대응하는 인수에 값이 제공되지 않거나 값이 DEFAULT로 지정된 경우만).
- ***특권***: 프로시저 정의자는 프로시저 삭제 권한은 물론 프로시저에 대한 EXECUTE 특권 WITH GRANT OPTION도 항상 수신합니다.

- *호환성*: z/OS용 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - ASUTIME NO LIMIT
    - NO COLLID
    - STAY RESIDENT NO

  이전 버전 DB2와의 호환성을 위해,
  - DYNAMIC RESULT SETS 대신 RESULT SETS를 지정할 수 있습니다.
  - CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.

예:

예 1: 중간 직원 급여를 리턴하는 SQL 프로시저를 작성하십시오. 중간 급여보다 많이 받는 모든 사원들의 이름, 지위 및 급여를 포함하는 결과 세트를 리턴하십시오.

```
CREATE PROCEDURE MEDIAN_RESULT_SET (OUT medianSalary DOUBLE)
  RESULT SETS 1
  LANGUAGE SQL
   BEGIN
       DECLARE v_numRecords INT DEFAULT 1;
  DECLARE v_counter INT DEFAULT 0;

  DECLARE c1 CURSOR FOR
           SELECT CAST(salary AS DOUBLE)
      FROM staff
      ORDER BY salary;
  DECLARE c2 CURSOR WITH RETURN FOR
           SELECT name, job, CAST(salary AS INTEGER)
      FROM staff
      WHERE salary > medianSalary
      ORDER BY salary;

  DECLARE EXIT HANDLER FOR NOT FOUND
           SET medianSalary = 6666;

  SET medianSalary = 0;
  SELECT COUNT(*) INTO v_numRecords
    FROM STAFF;
  OPEN c1;
  WHILE v_counter < (v_numRecords / 2 + 1)
  DO
           FETCH c1 INTO medianSalary;
    SET v_counter = v_counter + 1;
  END WHILE;
  CLOSE c1;
  OPEN c2;
END
```

CREATE ROLE

CREATE ROLE문은 현재 서버에서 역할을 정의합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문

설명

role-name

역할 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. 이름은 현재 서버에서 기존 역할을 식별해서는 안됩니다(SQLSTATE 42710).이름은 문자 'SYS'로 시작할 수 없고 'ACCESSCTRL', 'DATAACCESS', 'DBADM', 'NONE', 'NULL', 'PUBLIC', 'SECADM', 'SQLADM' 또는 'WLMADM'이 될 수 없습니다(SQLSTATE 42939).

예 :

이름이 DOCTOR인 역할을 작성하십시오.

```
CREATE ROLE DOCTOR
```

CREATE SCHEMA

CREATE SCHEMA문은 스키마를 정의합니다. 또한 일부 오브젝트를 작성하고 명령문 내에서 오브젝트에 대한 특권을 부여할 수도 있습니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

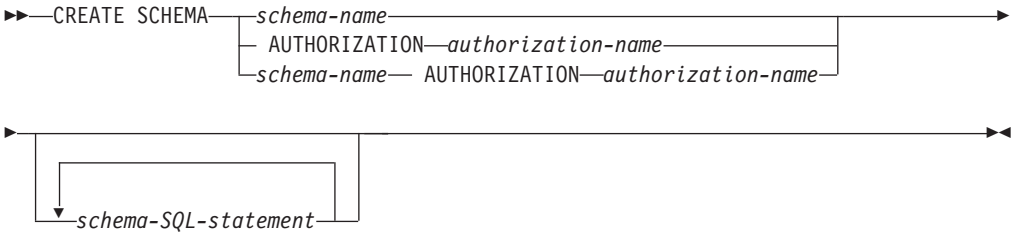
DBADM 권한을 보유한 권한 부여 ID는 유효한 *schema-name* 또는 *authorization-name*을 사용하여 스키마를 작성할 수 있습니다.

DBADM 권한을 갖고 있지 않은 권한 부여 ID는 명령문의 권한 부여 ID에 부합하는 *schema-name* 또는 *authorization-name*을 사용하여 스키마를 작성할 수 있습니다.

명령문에 *schema-SQL-statement*가 들어 있는 경우, *authorization-name*(지정되지 않은 경우, 명령문의 권한 부여 ID로 디폴트 설정됨)이 보유한 특권에는 다음 중 적어도 하나가 포함되어야 합니다.

- 각각의 *schema-SQL-statement*를 수행하는 데 필요한 특권
- DBADM 권한

구문



설명

schema-name

스키마의 이름을 지정합니다. 이 이름은 카탈로그에 이미 기술되어 있는 스키마를 식별해서는 안됩니다(SQLSTATE 42710). 이름은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939). 스키마의 소유자는 명령문을 발행한 권한 부여 ID입니다.

AUTHORIZATION *authorization-name*

스키마의 소유자인 사용자를 식별합니다. *authorization-name* 값은 스키마 이름을 지정하는 데에도 사용됩니다. *authorization-name*은 카탈로그에 이미 기술되어 있는 스키마를 식별해서는 안됩니다(SQLSTATE 42710).

schema-name **AUTHORIZATION** *authorization-name*

소유자가 *authorization-name*인 *schema-name*이라는 스키마를 식별합니다. *schema-name*은 카탈로그에 이미 기술되어 있는 스키마를 식별해서는 안됩니다 (SQLSTATE 42710). *schema-name*은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

schema-SQL-statement

CREATE SCHEMA문의 일부로 포함시킬 수 있는 SQL문은 다음과 같습니다.

- CREATE TABLE문: 유형이 지정된 테이블과 구체화된 쿼리 테이블 제외
- CREATE VIEW문: 유형이 지정된 뷰 제외
- CREATE INDEX문
- COMMENT문
- GRANT문

주

- 스키마의 소유자는 다음과 같이 결정됩니다.
  - AUTHORIZATION절이 지정된 경우, 지정된 *authorization-name*이 스키마 소유자입니다.
  - AUTHORIZATION절이 지정되지 않은 경우, CREATE SCHEMA문을 발행한 권한 부여 ID가 스키마 소유자입니다.
- 스키마 소유자는 사용자(그룹이 아님)로 간주됩니다.
- CREATE SCHEMA문을 사용하여 스키마가 명시적으로 작성된 경우, 스키마 소유자에게는 스키마에 대한 CREATEIN, DROPIN 및 ALTERIN 특권이 부여되며, 이 특권을 다른 사용자에게 부여할 수도 있습니다.
- CREATE SCHEMA문의 일부로 작성된 오브젝트의 정의자가 스키마 소유자입니다. 스키마 소유자는 CREATE SCHEMA문의 일부로 부여된 특권을 부여한 사용자이기도 합니다.
- CREATE SCHEMA문의 SQL문에 있는 규정되지 않은 오브젝트 이름은 작성된 스키마 이름에 의해 내재적으로 규정화됩니다.
- CREATE문에 작성될 오브젝트에 대한 규정된 이름이 포함된 경우, 규정된 이름에 지정된 스키마 이름은 작성될 스키마의 이름과 같아야 합니다(SQLSTATE 42875). 명령문 내에서 참조되는 다른 오브젝트는 다른 유효한 스키마 이름을 사용하여 규정할 수 있습니다.
- 스키마 이름으로 ″SESSION″을 사용하지 않도록 하십시오. 선언된 임시 테이블은 ″SESSION″에 의해 규정되어야 하므로, 영구 테이블의 이름과 같은 이름의 임시 테이블을 응용프로그램에서 선언되도록 할 수 있습니다. 스키마 이름이 ″SESSION″인 테이블을 참조하는 SQL문은 같은 이름의 영구적 테이블이 아니라 선언된 임시 테이블로 분석됩니다(명령문이 컴파일될 때). SQL문은 정적 Embedded SQL과 동적

Embedded SQL문에 대해 서로 다른 시간에 컴파일되므로, 결과는 선언된 임시 테이블이 정의되는 시기에 따라 다릅니다. 지속적 테이블, 뷰 또는 별명이 "SESSION"이라는 스키마 이름으로 정의되지 않은 경우, 이 사항은 고려하지 않아도 됩니다.

예:

예 1: DBADM 권한을 가진 사용자로서, 사용자 RICK을 소유자로 하여 RICK이라는 스키마를 작성하십시오.

```
CREATE SCHEMA RICK AUTHORIZATION RICK
```

예 2: 재고 부품표와 부품 번호에 대한 인덱스를 갖는 스키마를 작성하십시오. 테이블에 대한 권한을 JONES에게 부여하십시오.

```
CREATE SCHEMA INVENTRY

  CREATE TABLE PART (PARTNO   SMALLINT NOT NULL,
                                        DESCR    VARCHAR(24),
                                        QUANTITY INTEGER)

  CREATE INDEX PARTIND ON PART (PARTNO)

  GRANT ALL ON PART TO JONES
```

예 3: 다른 테이블을 참조하는 각자의 외부 키를 갖는 두 개의 테이블을 사용하여 PERS라는 스키마를 작성하십시오. 이것은 ALTER TABLE문을 사용하지 않고 테이블을 작성할 수 있도록 하는 CREATE SCHEMA문의 등록 정보를 보여주는 예입니다.

```
CREATE SCHEMA PERS

  CREATE TABLE ORG (DEPTNUMB  SMALLINT NOT NULL,
                                        DEPTNAME VARCHAR(14),
                                        MANAGER  SMALLINT,
                                        DIVISION VARCHAR(10),
                                        LOCATION VARCHAR(13),
                    CONSTRAINT PKEYDNO
                      PRIMARY KEY (DEPTNUMB),
                    CONSTRAINT FKEYMGR
                      FOREIGN KEY (MANAGER)
                      REFERENCES STAFF (ID) )

  CREATE TABLE STAFF (ID        SMALLINT NOT NULL,
                                          NAME     VARCHAR(9),
                                          DEPT     SMALLINT,
                                          JOB      VARCHAR(5),
                                          YEARS    SMALLINT,
                                          SALARY   DECIMAL(7,2),
                                          COMM     DECIMAL(7,2),
                     CONSTRAINT PKEYID
                       PRIMARY KEY (ID),
                     CONSTRAINT FKEYDNO
                       FOREIGN KEY (DEPT)
                       REFERENCES ORG (DEPTNUMB) )
```

CREATE SECURITY LABEL COMPONENT

CREATE SECURITY LABEL COMPONENT 명령문은 보안 규정의 일부로 사용할 구성요소를 정의합니다.

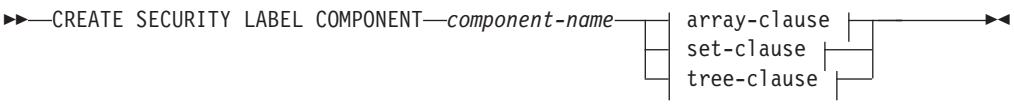
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

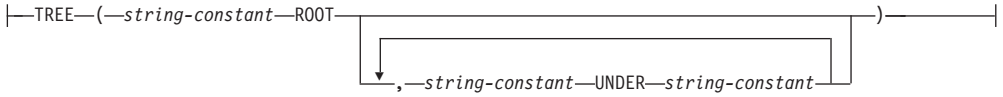
구문



array-clause:

set-clause:

tree-clause:



설명

component-name

보안 레이블 구성요소의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이 이름은 현재 서버의 기존 보안 레이블 구성요소를 식별해서는 안됩니다(SQLSTATE 42710).

ARRAY

요소의 순서 세트를 지정합니다.

string-constant,...

해당 보안 레이블 구성요소에 대한 유효한 값 세트를 구성하는 하나 이상의 문자열 상수 값. 배열 요소가 표시되는 순서는 중요합니다. 첫 번째 요소는 두 번째 요소보다 높은 순위를 갖고, 두 번째 요소는 세 번째 요소보다 높은 순위를 갖는 등의 방식입니다.

SET

요소의 무순 세트를 지정합니다.

string-constant,...

해당 보안 레이블 구성요소에 대한 유효한 값 세트를 구성하는 하나 이상의 문자열 상수 값. 요소의 순서는 중요하지 않습니다.

TREE

노드 요소의 트리 구조를 지정합니다.

string-constant

해당 보안 레이블 구성요소에 대한 유효한 값 세트를 구성하는 하나 이상의 문자열 상수 값.

ROOT

키워드 다음에 나오는 *string-constant*가 트리의 루트 노드 요소임을 지정합니다.

UNDER

UNDER 키워드 앞에 나오는 *string-constant*가 **UNDER** 키워드 다음에 나오는 *string-constant*의 하위 요소임을 지정합니다. 상위 요소로 사용하려면 요소를 루트 요소 또는 다른 요소의 하위 요소로 정의해야 하며, 그렇지 않을 경우 오류(SQLSTATE 42704)가 리턴됩니다.

규칙

다음 규칙은 세 가지 유형의 구성요소(ARRAY, SET 및 TREE) 모두에 적용됩니다.

- 요소 이름은 다음 문자를 포함할 수 없습니다.
  - 여는 괄호 - (
  - 닫는 괄호 -)
  - 쉼표 - ,
  - 콜론 - :
- 요소 이름은 32바이트를 초과할 수 없습니다(SQLSTATE 42622).
- 보안 레이블 구성요소가 세트 또는 트리이면, 해당 구성요소에 포함될 수 있는 요소는 64개를 넘을 수 없습니다.

- CREATE SECURITY LABEL COMPONENT문은 배열 유형의 보안 레이블 구성요소에 대해 최대 65 535개 요소를 지정할 수 있습니다.
- 동일한 구성요소에서 요소 이름을 두 번 이상 사용할 수 없습니다(SQLSTATE 42713).

예:

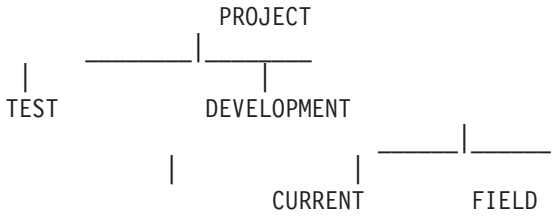
예 1: LEVEL이라는 ARRAY 유형 보안 레이블을 작성합니다. 구성요소에는 Top Secret, Secret, Classified 및 Unclassified 요소가 있습니다(높은 순위부터 나열됨).

```
CREATE SECURITY LABEL COMPONENT LEVEL
  ARRAY ['Top Secret', 'Secret', 'Classified', 'Unclassified']
```

예 2: COMPARTMENTS라는 SET 유형 보안 레이블 구성요소를 작성합니다. 구성요소에는 Research, Analysis 및 Collection 요소가 있습니다.

```
CREATE SECURITY LABEL COMPONENT COMPARTMENTS
  SET {'Collection', 'Research', 'Analysis'}
```

예 3: GROUPS라는 TREE 유형 보안 레이블 구성요소를 작성합니다. GROUPS에는 PROJECT, TEST, DEVELOPMENT, CURRENT 및 FIELD라는 5개의 요소가 있습니다. 다음 다이어그램은 해당 요소의 서로에 대한 관계를 나타냅니다.




```
CREATE SECURITY LABEL COMPONENT GROUPS
  TREE (
           'PROJECT' ROOT,
           'TEST' UNDER 'PROJECT',
           'DEVELOPMENT' UNDER 'PROJECT',
           'CURRENT' UNDER 'DEVELOPMENT',
           'FIELD' UNDER 'DEVELOPMENT'
  )
```

CREATE SECURITY LABEL

CREATE SECURITY LABEL문은 보안 레이블을 정의합니다.

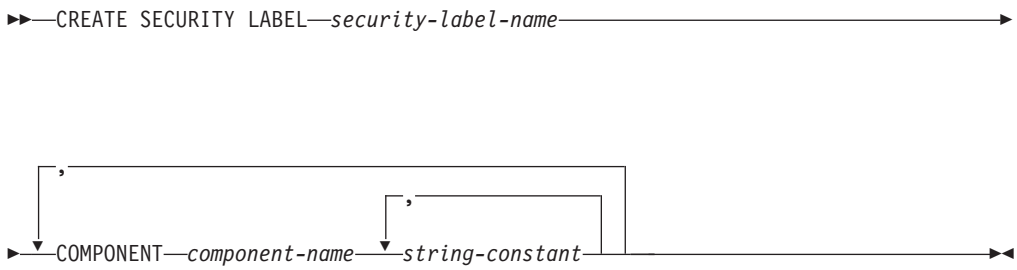
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



설명

security-label-name

보안 레이블의 이름을 지정합니다. 이름은 보안 규정을 사용하여 규정해야 하며(SQLSTATE 42704) 해당 보안 규정의 기존 보안 레이블을 식별하지 않아야 합니다(SQLSTATE 42710).

COMPONENT *component-name*

보안 레이블 구성요소의 이름을 지정합니다. 구성요소가 보안 규정 *security-policy-name*의 파트가 아니면 오류가 리턴됩니다(SQLSTATE 4274G). 구성요소가 동일한 명령문에서 두 번 지정되면 오류가 리턴됩니다(SQLSTATE 42713).

string-constant,...

보안 구성요소의 유효한 요소를 지정합니다. 유효한 요소는 보안 구성요소가 작성될 때 지정된 요소입니다. 요소가 유효하지 않을 경우, 오류가 리턴됩니다(SQLSTATE 4274F).

예:

예 1: DATA_ACCESS 보안 규정의 파트이고 LEVEL 구성요소의 Top Secret 요소와 COMPARTMENTS 구성요소의 Research 및 Analysis 요소를 갖는 EMPLOYEESECLABEL 보안 레이블을 작성하십시오.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research', 'Analysis'
```

예 2: LEVEL 구성요소의 Top Secret 요소와 COMPARTMENTS 구성요소의 Research 요소를 갖는 EMPLOYEESECLABELREAD 보안 레이블을 작성하십시오.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research'
```

예 3: COMPARTMENTS 구성요소에 대한 Analysis 요소와 LEVEL 구성요소에 대한 널(NULL) 값을 갖는 EMPLOYEESECLABELWRITE 보안 레이블을 작성하십시오. DATA_ACCESS 보안 규정이 예 1 및 2에서 사용되는 것과 동일한 보안 규정이라고 가정하십시오.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
  COMPONENT COMPARTMENTS 'Analysis'
```

예 4: 기존 CLASSPOLICY 보안 규정의 파트이며 TRUST 구성요소에 대한 Trainee 요소 및 SECTIONS 구성요소에 대한 Morning 요소를 갖는 보안 레이블 BEGINNER를 작성하십시오.

```
CREATE SECURITY LABEL CLASSPOLICY.BEGINNER
  COMPONENT TRUST 'Trainee',
  COMPONENT SECTIONS 'Morning'
```

CREATE SECURITY POLICY

CREATE SECURITY POLICY문은 보안 규정을 정의합니다.

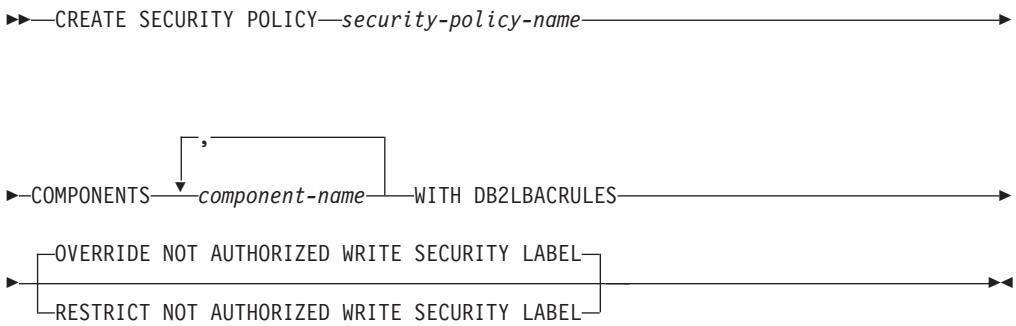
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



설명

security-policy-name

보안 규정의 이름을 지정합니다. 이 이름은 한 부분으로 구성된 이름입니다. 이름은 현재 서버의 기존 보안 규정을 식별해서는 안됩니다(SQLSTATE 42710).

COMPONENTS *component-name*,...

보안 레이블 구성요소를 식별합니다. 이름은 현재 서버에 이미 존재하는 보안 레이블 구성요소를 식별해야 합니다(SQLSTATE 42704). 동일한 보안 구성요소를 보안 규정에 대해 두 번 이상 지정할 수 없습니다(SQLSTATE 42713). 17개 이상의 보안 레이블 구성요소를 보안 규정에 대해 지정할 수 없습니다(SQLSTATE 54062).

WITH DB2LBACRULES

해당 보안 규정의 일부인 보안 레이블을 비교할 때 사용할 규칙을 표시합니다. 현재 DB2LBACRULES와 같은 하나의 규칙 세트가 존재합니다.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 또는 **RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL**

사용자에게 해당 보안 규정으로 보호된 테이블에 대해 발행된 INSERT 또는

UPDATE문에 제공된 명시적으로 지정된 보안 레이블을 작성할 권한이 없을 때 취할 조치를 지정합니다. 사용자의 보안 레이블 및 면제 증명서는 명시적으로 제공된 보안 레이블을 작성하는 사용자의 권한을 판별합니다. 디폴트값은 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL입니다.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
명시적으로 지정된 보안 레이블이 아닌 사용자의 보안 레이블 값을 삽입 또는 갱신 조작 도중 쓰기 액세스에 사용하도록 표시합니다.

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
사용자에게 INSERT 또는 UPDATE문에 제공된 명시적으로 지정된 보안 레이블을 작성할 권한이 없을 경우 삽입 또는 갱신 조작이 실패함을 표시합니다 (SQLSTATE 42519).

주

- ***DB2LBACRULES 규칙 세트:*** DB2LBACRULES는 다음 규칙을 포함하는 규칙의 미리 정의된 세트입니다. DB2LBACREADARRAY, DB2LBACREADSET, DB2LBACREADTREE, DB2LBACWRITEARRAY, DB2LBACWRITESET, DB2LBACWRITETREE.
- 그룹 및 역할 권한 부여는 보안 규정이 작성될 때 기본적으로 고려되지 않습니다. 이 동작을 변경하여 고려하려면 ALTER SECURITY POLICY문을 사용하십시오.

예:

예 1: DB2LBACRULES 규칙 세트를 사용하며 LEVEL과 COMPARTMENTS 구성요소를 순서대로 갖는 DATA_ACCESS이라는 보안 규정을 작성합니다. 두 개의 구성요소가 이미 있다고 가정합니다.

```
CREATE SECURITY POLICY DATA_ACCESS
  COMPONENTS LEVEL, COMPARTMENTS
  WITH DB2LBACRULES
```

예 2: MEMBER 및 BADGE 구성요소(이미 존재한다고 가정)를 갖는 CONTRIBUTIONS라는 보안 규정을 작성합니다.

```
CREATE SECURITY POLICY CONTRIBUTIONS
  COMPONENTS MEMBER, BADGE
  WITH DB2LBACRULES
```

CREATE SEQUENCE

CREATE SEQUENCE문은 응용프로그램 서버에 시퀀스를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

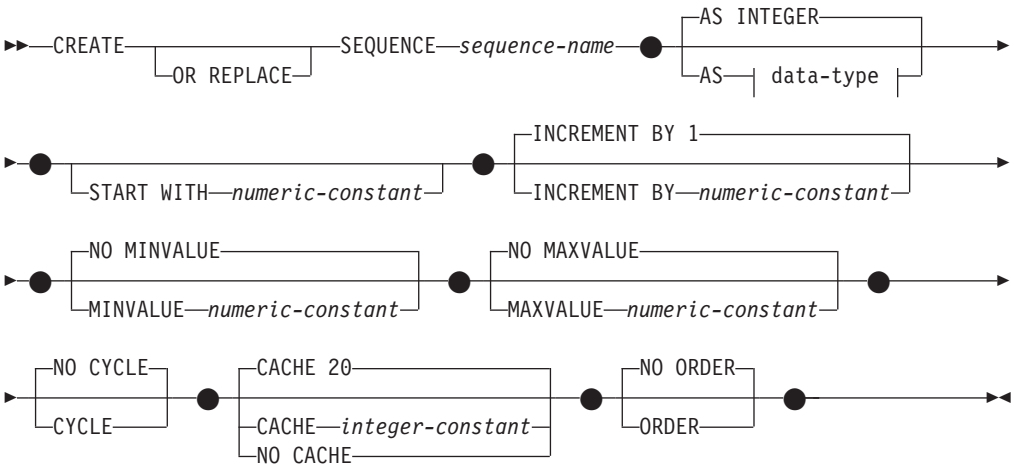
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

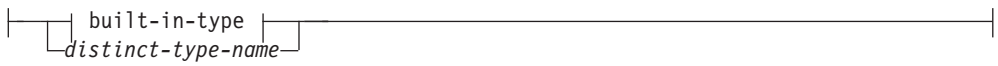
- 시퀀스의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 시퀀스의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

기존 시퀀스를 교체하려면 명령문의 권한 부여 ID가 기존 시퀀스의 소유자여야 합니다(SQLSTATE 42501).

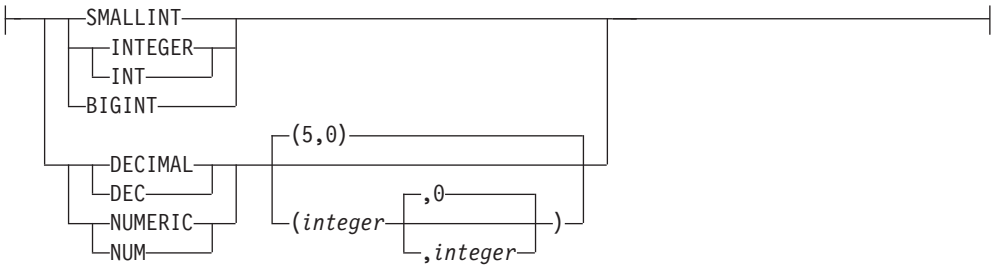
구문



data-type:



내장 유형:



설명

OR REPLACE

현재 서버에 정의가 존재하면 시퀀스의 정의를 교체하도록 지정합니다. 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 삭제되며, 시퀀스에 부여되었던 특권에는 영향을 주지 않습니다. 시퀀스의 정의가 현재 서버에 존재하지 않으면 이 옵션이 무시됩니다.

sequence-name

시퀀스 이름을 지정합니다. 이름 조합과 내재적 또는 명시적 스키마 이름은 현재 서버에 있는 기존 시퀀스를 식별해서는 안됩니다(SQLSTATE 42710).

시퀀스 이름의 규정되지 않은 형식은 SQL ID입니다. 규정된 형식은 뒤에 마침표와 SQL ID가 오는 규정자입니다. 규정자는 스키마 이름입니다.

시퀀스 이름이 스키마 이름을 사용하여 명시적으로 규정될 경우에 스키마 이름은 'SYS'로 시작할 수 없습니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42939).

AS *data-type*

시퀀스 값에 사용할 데이터 유형을 지정합니다. 데이터 유형은 스케일이 0인 정확한 숫자 유형(SMALLINT, INTEGER, BIGINT 또는 DECIMAL)이거나 소스 유형이 0의 스케일을 갖는 정확한 숫자 유형인 사용자 정의 구별 유형 또는 참조 유형일 수 있습니다(SQLSTATE 42815). 디폴트값은 INTEGER입니다.

START WITH *numeric-constant*

시퀀스의 맨 처음 값을 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않습니다(SQLSTATE 428FA). 디폴트값은 오름차순 시퀀스의 경우 MINVALUE이고, 내림차순 시퀀스의 경우 MAXVALUE입니다.

이 값이 반드시 시퀀스의 최대값 또는 최소값에 도달한 후 시퀀스가 순환하는 값일 필요는 없습니다. START WITH절을 사용하여 순환에 사용되는 범위 밖에서 시퀀스를 시작할 수 있습니다. 순환에 사용되는 범위는 MINVALUE 및 MAXVALUE에 의해 정의됩니다.

INCREMENT BY *numeric-constant*

시퀀스의 연속되는 값의 간격을 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있습니다(SQLSTATE 42815). 이 값은 큰 정수 상수의 값을 초과하지 않고(SQLSTATE 42820) 소수점 오른쪽에 0이 아닌 자릿수를 갖지 않아야 합니다(SQLSTATE 428FA).

이 값이 음수인 경우, 값은 내림차순입니다. 이 값이 0이거나 양수인 경우, 값은 오름차순입니다. 디폴트값은 1입니다.

MINVALUE 또는 NO MINVALUE

내림차순 시퀀스가 값 생성을 순환하거나 중지하여 최소값을 지정합니다. 또는 오름차순 시퀀스가 최대값에 도달한 후 순환하여 최소값을 지정합니다.

MINVALUE *numeric-constant*

최소값인 숫자 상수를 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최대값보다 작거나 같아야 합니다(SQLSTATE 42815).

NO MINVALUE

오름차순 시퀀스의 경우, START WITH 값으로, START WITH 값이 지정되지 않은 경우 1입니다. 내림차순 시퀀스의 경우, 값은 시퀀스와 연관된 데이터 유형의 최소값입니다. 이는 디폴트값입니다.

MAXVALUE 또는 NO MAXVALUE

오름차순 시퀀스가 값 생성을 순환하거나 중지하여 최대값을 지정합니다. 또는 내림차순 시퀀스가 최소값에 도달한 후 순환하여 최대값을 지정합니다.

MAXVALUE *numeric-constant*

최대값인 숫자 상수를 지정합니다. 이 값은 시퀀스와 연관된 데이터 유형의 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최소값보다 크거나 같아야 합니다(SQLSTATE 42815).

NO MAXVALUE

오름차순 시퀀스의 경우, 시퀀스와 연관된 데이터 유형의 최대값입니다. 내림차순 시퀀스의 경우, START WITH 값으로, START WITH 값이 지정되지 않은 경우 1입니다.

CYCLE 또는 NO CYCLE

시퀀스가 최대값 또는 최소값에 도달한 이후에도 계속 값을 생성할지 여부를 지정합니다. 시퀀스의 바운더리는 바운더리 조건에 정확하게 도착하는 다음 값이나 바운더리 조건을 지나쳐서 도달될 수 있습니다.

CYCLE

최대값 또는 최소값에 도달된 이후에도 이 시퀀스에 대해 값을 계속 생성하도

록 지정합니다. 이 옵션을 사용할 경우, 오름차순 시퀀스가 최대값에 도달하면 최소값이 생성되고, 내림차순 시퀀스가 최소값에 도달하면 최대값이 생성됩니다. 시퀀스의 최대값과 최소값에 따라 순환에 사용되는 범위가 결정됩니다.

CYCLE이 효력을 가질 때, 시퀀스에 대해 중복 값이 생성될 수 있습니다.

NO CYCLE
시퀀스의 최대값이나 최소값에 도달한 이후에는 시퀀스에 대해 값이 생성되지 않도록 지정합니다. 이는 디폴트값입니다.

CACHE 또는 NO CACHE
보다 빠른 액세스를 위해 메모리에 사전 할당된 값 일부를 보존할지 여부를 지정합니다. 이것은 성능 및 조정 옵션입니다.

CACHE *integer-constant*
사전 할당되고 메모리에 보존되는 최대 시퀀스 값을 지정합니다. 캐시에 값을 사전 할당하고 저장하면 시퀀스에 대한 값이 생성될 때 로그에 대한 동기 입출력이 줄어듭니다.

시스템 오류가 발생할 경우, 커미트된 명령문에서 사용되지 않은 캐시된 모든 시퀀스 값이 없어집니다. 즉, 사용되지 않습니다. CACHE 옵션에 대해 지정된 값은 시스템 오류 시 없어질 수 있는 최대 시퀀스 값입니다.

최소값은 2입니다(SQLSTATE 42815). 디폴트값은 CACHE 20입니다.

NO CACHE
시퀀스 값이 사전 할당되지 않도록 지정합니다. 따라서 시스템 오류, 시스템 종료 또는 데이터베이스 비활성화의 경우에 값의 유실되지 않습니다. 이 옵션을 지정하면 시퀀스의 값이 캐시에 저장되지 않습니다. 이 경우, 새 시퀀스 값에 대한 요청이 있을 때마다 동기 입출력이 로그에 기록됩니다.

NO ORDER 또는 ORDER
시퀀스 번호를 요청 순서대로 생성할지 여부를 지정합니다.

ORDER
시퀀스 번호가 요청 순서대로 생성되도록 지정합니다.

NO ORDER
시퀀스 번호가 요청 순서대로 생성되지 않도록 지정합니다. 이는 디폴트값입니다.

주

- 상수 시퀀스, 즉 항상 상수 값을 리턴하는 시퀀스를 정의할 수 있습니다. INCREMENT 값을 0으로 지정하고 START WITH 값을 MAXVALUE를 초과하지 않는 값으로 지정하거나, START WITH, MINVALUE 및 MAXVALUE에 모두 같은 값을 지정하면 됩니다. 상수 시퀀스의 경우에는 시퀀스에 대해 NEXT

VALUE를 호출할 때마다 같은 값이 리턴됩니다. 상수 시퀀스는 숫자 전역 변수로 사용할 수 있습니다. ALTER SEQUENCE를 사용하여 상수 시퀀스에 대해 생성될 값을 조정할 수 있습니다.

- ALTER SEQUENCE문을 사용하여 시퀀스를 수동으로 순환시킬 수 있습니다. NO CYCLE을 내재적 또는 명시적으로 지정하면, ALTER SEQUENCE문을 사용하여 시퀀스를 재시작 또는 확장하여 시퀀스에 대한 최대값이나 최소값에 도달된 후에 값이 계속 생성되도록 할 수 있습니다.
- CYCLE 키워드를 지정하여 시퀀스가 순환하도록 명시적으로 정의할 수 있습니다. 바운더리에 도달하면 생성된 값이 순환해야 한다는 것을 나타내는 시퀀스를 정의할 때 CYCLE 옵션을 사용하십시오. 자동으로 순환하도록 시퀀스를 정의할 경우(즉, CYCLE을 명시적으로 지정하면), 증분 값이 1이나 -1이 아닌 다른 값이면 시퀀스에 대해 생성된 최대값이나 최소값이 지정된 실제 MAXVALUE나 MINVALUE가 아닐 수 있습니다. 예를 들어, `START WITH=1, INCREMENT=2, MAXVALUE=10`으로 정의한 시퀀스가 최대값으로 10이 아닌 9를 생성합니다. CYCLE을 사용하여 시퀀스를 정의할 때는 MINVALUE, MAXVALUE, START WITH 값이 미치는 영향을 고려해야 합니다.
- 시퀀스 번호 캐싱은 빠른 액세스를 위해 시퀀스 번호의 범위를 메모리에 저장할 수 있음을 의미합니다. 응용프로그램이 캐시로부터 다음 시퀀스 번호를 할당할 수 있는 시퀀스에 액세스하면, 시퀀스 번호 할당이 빨리 이뤄질 수 있습니다. 그러나 응용프로그램이 캐시로부터 다음 시퀀스 번호를 할당할 수 없는 시퀀스에 액세스하는 경우, 시퀀스 번호 할당은 지속적 스토리지에 대한 입출력 조작을 기다려야 합니다. CACHE 값은 성능 및 응용프로그램 요구사항이 제공하는 이점을 따져본 후 선택해야 합니다.
- 시퀀스 정의자에게는 권한 부여 옵션과 함께 ALTER 특권과 USAGE 특권이 부여됩니다. 시퀀스의 소유자는 시퀀스를 삭제할 수 있습니다.
- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 여러 시퀀스 옵션을 구분하는 데 쉼표(,)를 사용할 수 있습니다.

  다음 구문도 지원됩니다.

  - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE 및 NOORDER

예:

예 1: 1에서 시작하고 1씩 증가하며 순환하지 않으며 한 번에 24개의 값을 캐시하는 ORG_SEQ 시퀀스를 작성하십시오.

```
CREATE SEQUENCE ORG_SEQ
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

CREATE SERVICE CLASS

CREATE SERVICE CLASS문은 서비스 클래스를 정의합니다.

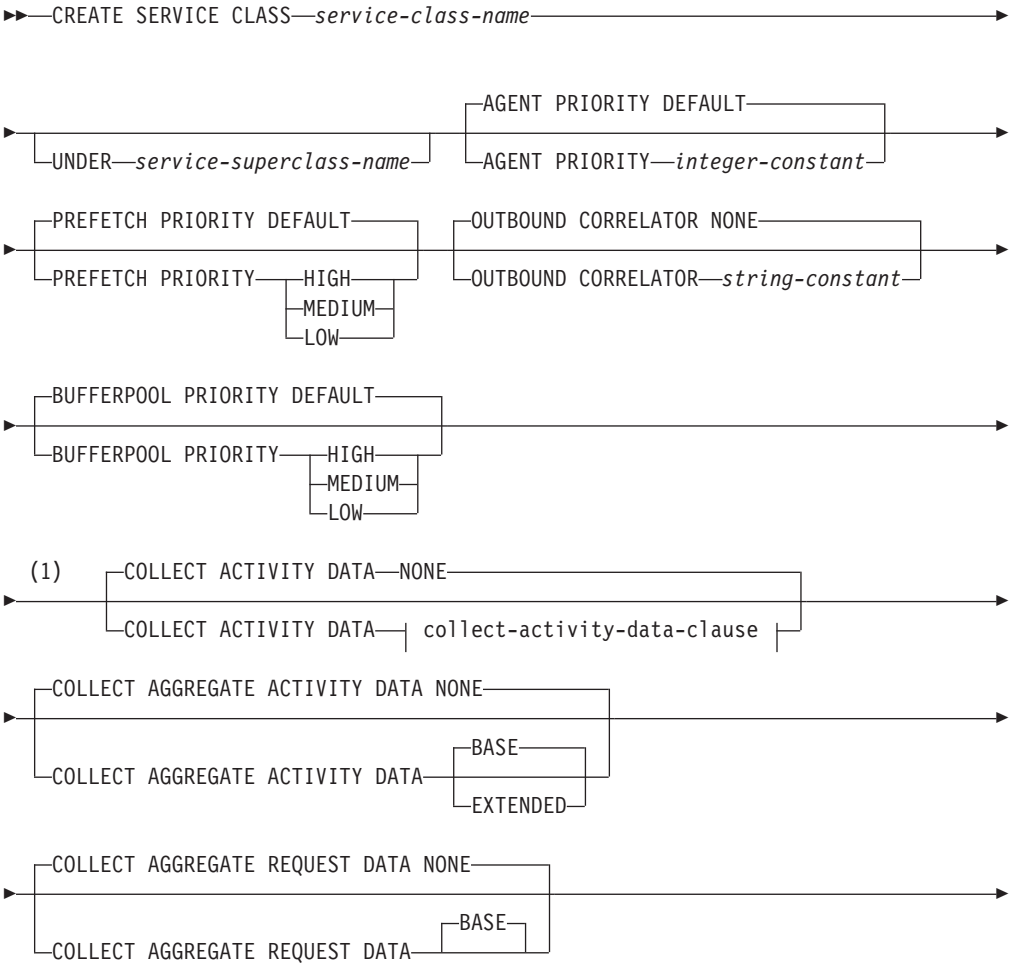
호출

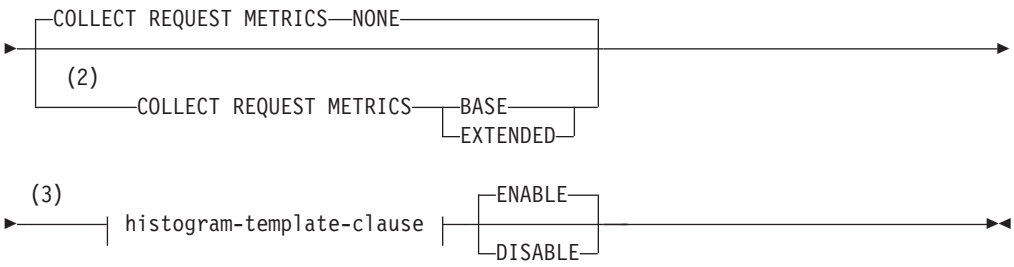
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

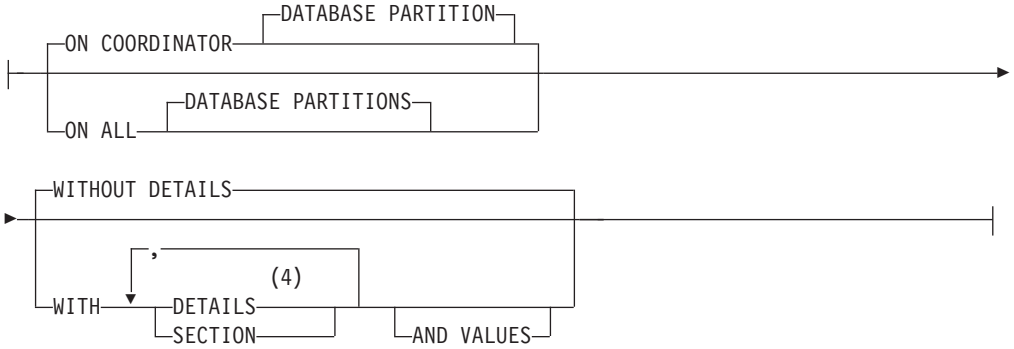
명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

구문

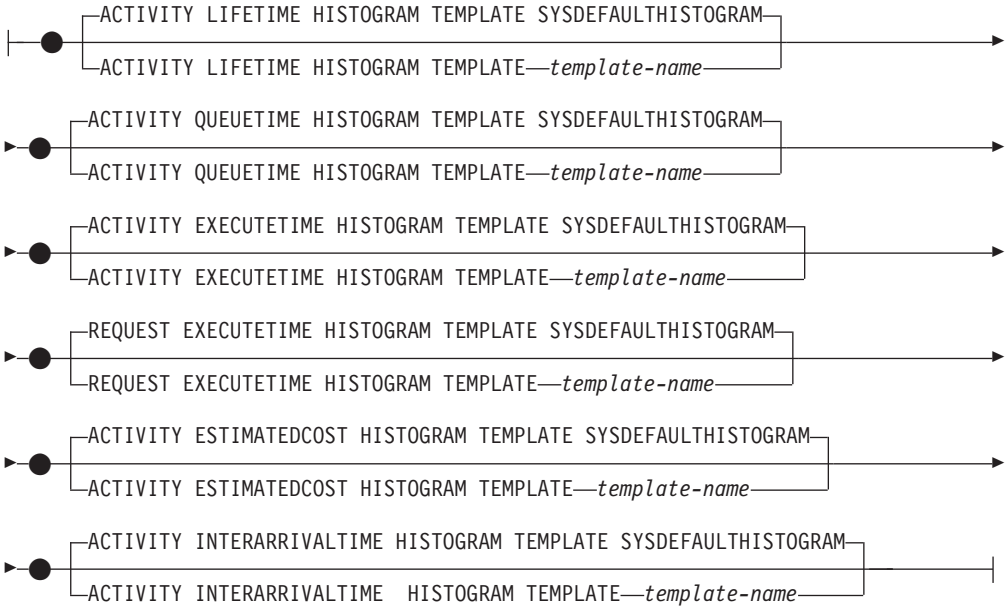




collect-activity-data-clause:



histogram-template-clause:



주:

1 COLLECT REQUEST METRICS를 제외한 모든 COLLECT절은 서비스 서브클래스에 대해서만 유효합니다.

2 COLLECT REQUEST METRICS절은 서비스 수퍼 클래스에 대해서만 유효합니다.

3 HISTOGRAM TEMPLATE절은 서비스 서브클래스에 대해서만 유효합니다.

4 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

service-class-name

서비스 클래스의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. 서비스 클래스가 서비스 수퍼 클래스인 경우 *service-class-name*은 카탈로그에 이미 존재하는 서비스 수퍼 클래스를 식별하면 안됩니다(SQLSTATE 42710). 서비스 클래스가 서비스 서브클래스인 경우 *service-class-name*은 서비스 수퍼 클래스에 이미 존재하는 서비스 서브클래스를 식별하면 안됩니다(SQLSTATE 42710). 서비스 클래스가 서비스 서브클래스인 경우 *service-class-name*은 서비스 수퍼 클래스와 같을 수 없습니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

UNDER *service-superclass-name*

서비스 클래스가 서비스 수퍼 클래스 *service-superclass-name*의 서브클래스임을 지정합니다. UNDER를 지정하지 않은 경우 서비스 클래스는 서비스 수퍼 클래스입니다. *service-superclass-name*은 데이터베이스에 대해 존재하는 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704). 서비스 수퍼 클래스는 디폴트 서비스 클래스가 될 수 없습니다(SQLSTATE 5U029).

AGENT PRIORITY DEFAULT 또는 AGENT PRIORITY *integer-constant*

서비스 클래스에서 실행 중인 에이전트의 상대적(델타) 운영 체제 우선순위나 DB2에서 실행 중인 스레드의 일반 우선순위를 지정합니다. 디폴트값은 DEFAULT입니다. DEFAULT로 설정한 경우, 어떤 특수 조치도 수행되지 않고 서비스 클래스의 에이전트는 운영 체제가 모든 DB2 스레드를 스케줄링하는 일반 우선순위에 따라 스케줄됩니다. 이 매개변수가 DEFAULT가 아닌 다른 값으로 설정되는 경우, 에이전트는 일반 우선순위에 다음 활동 시작 시 AGENT PRIORITY를 더한 우선순위로 설정됩니다. 예를 들어, 일반 우선순위가 20이고 AGENT PRIORITY가 -10으로 설정되면, 서비스 클래스에서 에이전트의 우선순위가 20 – 10 = 10으로 설정됩니다.

UNIX 운영 체제와 Linux에서, 유효한 값은 DEFAULT와 -20 ~ 20입니다(SQLSTATE 42615). 음수 값은 상대적으로 더 높은 우선순위를 나타냅니다. 양수 값은 상대적으로 낮은 우선순위를 나타냅니다.

Windows 운영 체제에서, 유효한 값은 DEFAULT와 -6 ~ 6입니다(SQLSTATE 42615). 음수 값은 상대적으로 더 낮은 우선순위를 나타냅니다. 양수 값은 상대적으로 더 높은 우선순위를 나타냅니다.

AGENT PRIORITY가 서비스 서브클래스에 대해 DEFAULT이면, 상위 서브클래스의 AGENT PRIORITY 값을 상속합니다. AGENT PRIORITY는 디폴트 서브클래스의 경우 변경할 수 없습니다. OUTBOUND CORRELATOR가 설정된 경우 AGENT PRIORITY는 DEFAULT로 설정해야 합니다(SQLSTATE 42613).

주: AIX에서, AGENT PRIORITY를 사용하는 서비스 클래스에서 에이전트에 대해 상대적으로 더 높은 우선순위를 설정하려면 인스턴스 소유자가 CAP_NUMA_ATTACH 및 CAP_PROPAGATE 성능을 가지고 있어야 합니다. 이 성능을 부여하려면 루트로 로그온하고 다음 명령을 실행하십시오.

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

PREFETCH PRIORITY DEFAULT | HIGH | MEDIUM | LOW

이 매개변수는 서비스 클래스의 에이전트가 프리페치 요청을 제출할 수 있는 우선순위를 제어합니다. 올바른 값은 HIGH, MEDIUM, LOW 또는 DEFAULT입니다(SQLSTATE 42615). HIGH, MEDIUM 및 LOW는 프리페치 요청이 높은, 중간 및 낮은 우선순위 큐에 제출됨을 의미합니다. 프리페치는 높음에서 낮음 순서로 우선순위 큐를 비웁니다. 서비스 클래스의 에이전트는 다음 활동이 시작될 때 PREFETCH PRIORITY 레벨에서 해당되는 프리페치 요청을 제출합니다. 프리페치 요청이 제출된 후 PREFETCH PRIORITY가 변경되어도 요청 우선순위는 변경되지 않습니다. 디폴트 값은 DEFAULT입니다. 이는 서비스 수퍼 클래스의 경우 내부적으로 MEDIUM에 맵핑됩니다. 서비스 서브클래스에 대해 DEFAULT가 지정되면, 해당되는 상위 수퍼 클래스의 PREFETCH PRIORITY를 상속합니다.

PREFETCH PRIORITY는 디폴트 서브클래스의 경우 변경할 수 없습니다(SQLSTATE 5U032).

OUTBOUND CORRELATOR NONE 또는 OUTBOUND CORRELATOR *string-constant*

서비스 클래스의 스레드를 외부 워크로드 관리 프로그램 서비스 클래스에 연관시킬 것인지 여부를 지정합니다.

OUTBOUND CORRELATOR가 서비스 수퍼 클래스에 대해 *string-constant*로 설정되고 OUTBOUND CORRELATOR NONE이 서비스 서브클래스에 대해 설정된 경우, 서비스 서브클래스는 상위의 OUTBOUND CORRELATOR를 상속합니다. AGENT PRIORITY가 DEFAULT로 설정되지 않으면 OUTBOUND CORRELATOR는 NONE으로 설정해야 합니다(SQLSTATE 42613). 디폴트는 OUTBOUND CORRELATOR NONE입니다.

OUTBOUND CORRELATOR NONE

서비스 수퍼 클래스의 경우 이 서비스 클래스와의 외부 워크로드 관리 프로그램 서비스 클래스 연관이 없음을 지정하고, 서비스 서브클래스의 경우 외부 워크로드 관리 프로그램 서비스 클래스 연관이 상위와 동일함을 지정합니다.

OUTBOUND CORRELATOR *string-constant*

서비스 클래스의 스레드를 외부 워크로드 관리 프로그램 서비스 클래스에 연관시키기 위한 상관자로 사용할 *string-constant*를 지정합니다. 외부 워크로드 관리 프로그램은 활성 상태여야 합니다(SQLSTATE 5U030). 외부 워크로드 관리 프로그램은 *string-constant* 값을 인식하도록 설정해야 합니다.

BUFFERPOOL PRIORITY DEFAULT | HIGH | MEDIUM | LOW

이 매개변수는 서비스 클래스에서 활동에 의해 페치된 페이지의 버퍼 풀 우선순위를 제어합니다. 유효한 값은 HIGH, MEDIUM, LOW 또는 DEFAULT입니다(SQLSTATE 42615). 버퍼 풀 우선순위가 높은 서비스 클래스에서 활동에 의해 페치된 페이지 수는 버퍼 풀 우선순위가 낮은 서비스 클래스에서 활동에 의해 페치된 페이지 수보다 적게 스왑 아웃됩니다. 디폴트 값은 DEFAULT입니다. 이는 서비스 수퍼 클래스의 경우 내부적으로 LOW에 맵핑됩니다. 서비스 서브클래스에 대해 DEFAULT가 지정되면 해당되는 상위 수퍼 클래스의 BUFFERPOOL PRIORITY를 상속합니다.

BUFFERPOOL PRIORITY는 디폴트 서브클래스의 경우 변경할 수 없습니다(SQLSTATE 5U032).

COLLECT ACTIVITY DATA

해당 서비스 클래스에서 실행되는 각 활동에 대한 데이터가 활동 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 디폴트는 COLLECT ACTIVITY DATA NONE입니다. COLLECT ACTIVITY DATA절은 서비스 서브클래스에 대해서만 유효합니다.

NONE

해당 서비스 클래스에서 실행되는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

ON COORDINATOR DATABASE PARTITION

활동 데이터가 활동 코디네이터의 데이터베이스 파티션에서만 수집되도록 지정합니다.

ON ALL DATABASE PARTITIONS

활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. SECTION이 지정된 경우, 활동 세부사항 및 섹션 환경은 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우, 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다.

WITHOUT DETAILS

서비스 클래스에서 실행되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내져야 함을 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS

명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION

명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA

해당 서비스 클래스에 대한 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. COLLECT AGGREGATE ACTIVITY DATA를 지정하지 않은 경우 디폴트는 COLLECT AGGREGATE ACTIVITY DATA NONE입니다. COLLECT AGGREGATE ACTIVITY DATA를 지정한 경우 디폴트는 COLLECT AGGREGATE ACTIVITY DATA BASE입니다. COLLECT AGGREGATE ACTIVITY DATA 절은 서비스 서브클래스에 대해서만 유효합니다.

BASE

해당 서비스 클래스에 대한 기본 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 실행 시간 막대 그래프

EXTENDED

해당 서비스 클래스에 대한 모든 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 데이터 처리 언어(DML) 계산 비용 막대 그래프

- 활동 DML 도착 간 시간 막대 그래프

NONE
어떤 집계 활동 데이터도 해당 서비스 클래스에 대해 수집되지 않음을 지정합니다.

COLLECT AGGREGATE REQUEST DATA
해당 서비스 클래스에 대한 집계 요청 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. 디폴트는 COLLECT AGGREGATE REQUEST DATA NONE입니다. COLLECT AGGREGATE REQUEST DATA 절은 서비스 서브클래스에 대해서만 유효합니다.

BASE
해당 서비스 클래스에 대한 기본 집계 요청 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다.

NONE
어떤 집계 요청 데이터도 해당 서비스 클래스에 대해 수집되지 않음을 지정합니다.

COLLECT REQUEST METRICS
지정된 서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 모니터 메트릭을 수집해야 함을 지정합니다. 디폴트는 COLLECT REQUEST METRICS NONE입니다. COLLECT REQUEST METRICS절은 서비스 수퍼 클래스에 대해서만 유효합니다(SQLSTATE 50U44).

주: 유효한 요청 메트릭 콜렉션 설정은 요청을 제출하는 워크로드에서 COLLECT REQUEST METRICS절로 지정된 속성 및 **mon_req_metrics** 데이터베이스 구성 매개변수의 조합입니다. 워크로드 속성 또는 구성 매개변수에 NONE이 아닌 다른 값이 있는 경우, 메트릭이 요청에 대해 수집됩니다.

NONE
서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 어떤 메트릭도 수집하지 않을 것을 지정합니다.

BASE
서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 기본 메트릭을 수집할 것을 지정합니다.

EXTENDED
서비스 수퍼 클래스와 연관되는 연결에서 제출된 요청에 대해 확장 메트릭을 수집할 것을 지정합니다.

histogram-template-clause

서비스 클래스에서 실행 중인 활동에 대한 집계 활동 데이터를 수집할 때 사용할 막대 그래프 템플리트를 지정합니다. HISTOGRAM TEMPLATE절은 서비스 서브클래스에 대해서만 유효합니다.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 서비스 클래스에서 실행 중인 DB2 활동의 지속기간(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

서비스 클래스에서 실행 중인 DB2 활동이 특정 간격 동안 큐에서 대기하는 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

서비스 클래스에서 실행 중인 DB2 활동이 특정 간격 동안 실행 중인 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 코디네이터 데이터베이스 파티션에서만 막대 그래프에서 수집됩니다. 시간에는 유휴 시간이 포함되지 않습니다. 유휴 시간은 어떤 작업도 완료되지 않은 경우에 동일한 활동에 속하는 요청의 실행 사이 시간입니다. 유휴 시간의 예로, 커서 열기가 종료되고 그 커서로부터 페치가 시작되는 시간 사이의 시간을 들 수 있습니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 중첩 레벨 0의 활동만 막대 그래프에 포함되는 것으로 간주됩니다.

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *template-name*

서비스 클래스에서 실행 중인 DB2 요청이 특정 간격 동안 실행 중인 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 요청 실행 시간은 요청이 실행되는 데이터베이스 파티션마다 이 막대 그래프에서 수집됩니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 옵션을 사용하여 COLLECT AGGREGATE REQUEST DATA절을 지정한 경우에만 수집됩니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*
서비스 클래스에서 실행 중인 DML 활동의 계산된 비용(timeron 단위)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 중첩 레벨 0의 활동만 막대 그래프에 포함되는 것으로 간주됩니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*
하나의 DML 활동이 도착하고 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ENABLE 또는 DISABLE
연결 또는 활동을 서비스 클래스에 맵핑할 수 있는지 여부를 지정합니다. 디폴트는 ENABLE입니다.

ENABLE
연결 또는 활동을 서비스 클래스에 맵핑할 수 있습니다.

DISABLE
연결 또는 활동을 서비스 클래스에 맵핑할 수 없습니다. 사용 불가능한 서비스 클래스에 맵핑되는 연결 또는 활동은 거부됩니다(SQLSTATE 5U028). 서비스 수퍼 클래스가 사용 불가능한 경우 해당 서비스 서브클래스도 사용 불가능합니다. 서비스 수퍼 클래스가 다시 사용 가능하게 되면 해당되는 서비스 서브클래스는 시스템 카탈로그에 정의된 상태로 리턴됩니다. 디폴트 서비스 클래스는 사용 불가능하도록 설정할 수 없습니다(SQLSTATE 5U032).

규칙

- 서비스 수퍼 클래스 아래에서 작성될 수 있는 최대 서비스 서브클래스 수는 61입니다(SQLSTATE 5U027).
- 데이터베이스에 대해 작성될 수 있는 최대 서비스 수퍼 클래스 수는 64입니다(SQLSTATE 5U027).
- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U027). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)

  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 디폴트 서브클래스 SYSDEFAULTSUBCLASS는 모든 서비스 수퍼 클래스에 대해 자동으로 작성됩니다.
- 모든 파티션 사이에서 한 번에 하나의 언커미트된 WLM 독점 SQL문만 허용됩니다. 언커미트된 WLM 독점 SQL문이 실행 중인 경우, 연속 WLM 독점 SQL문은 현재 WLM 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 시스템 카탈로그에 기록되지만, 명령문을 발행하는 연결에 대해서도 COMMIT문이 실행될 때까지 적용되지 않습니다.

예:

예 1: 서비스 수퍼 클래스 PETSALES를 작성하십시오. PETSALES의 디폴트 서브클래스는 자동으로 작성됩니다.

```
CREATE SERVICE CLASS PETSALES
```

예 2: 서비스 수퍼 클래스 PETSALES 아래에 서비스 서브클래스 DOGSALES를 작성하십시오. 서비스 클래스 DOGSALES를 사용 불가능 상태로 설정하십시오.

```
CREATE SERVICE CLASS DOGSALES UNDER PETSALES DISABLE
```

예 3: 프리페처 우선순위가 LOW인 서비스 수퍼 클래스 BARNSALES를 작성하십시오. BARNSALES의 디폴트 서브클래스는 자동으로 작성됩니다. BARNSALES 서비스 클래스에서 에이전트가 제출한 프리페치 요청은 낮은 우선순위 프리페치 큐로 이동됩니다.

```
CREATE SERVICE CLASS BARNSALES PREFETCH PRIORITY LOW
```

CREATE SERVER

CREATE SERVER문은 데이터 소스를 페더레이티드 데이터베이스에 정의합니다. 이 명령문에서 SERVER라는 용어와 *server*로 시작하는 매개변수 이름은 페더레이티드 시스템에 있는 데이터 소스만 참조합니다. 페더레이티드 시스템에 있는 페더레이티드 서버나 DRDA 응용프로그램 서버는 참조하지 않습니다.

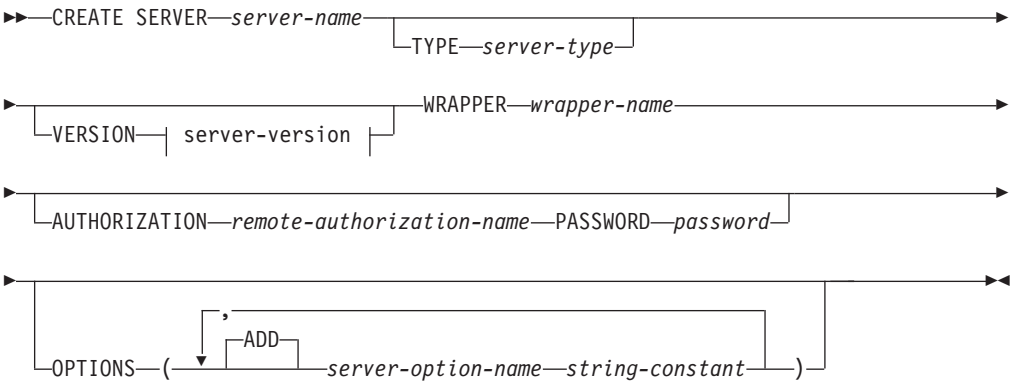
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

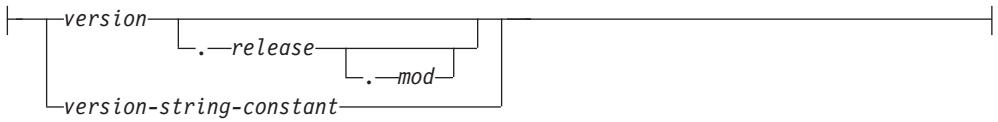
권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

구문



server-version:



설명

server-name

페더레이티드 데이터베이스에 정의될 데이터 소스의 이름을 지정합니다. 이름은 카탈로그에 기술되어 있는 데이터 소스를 식별해서는 안됩니다. *server-name*은 페더레이티드 데이터베이스에 있는 모든 테이블 스페이스 이름과 동일해서는 안됩니다.

관계형 데이터 소스에 대한 서버 정의는 일반적으로 리모트 데이터베이스를 나타냅니다. Oracle과 같은 일부 관계형 데이터베이스 관리 시스템은 각 인스턴스에 다중 데이터베이스를 허용하지 않습니다. 대신에 각 인스턴스는 페더레이티드 시스템 내에 있는 하나의 서버를 나타냅니다.

비관계형 데이터 소스의 경우, 서버 정의의 목적은 데이터 소스에 따라 다릅니다. 일부 서버 정의는 검색 유형과 디먼, 웹 사이트 또는 웹 서버에 맵핑됩니다. 다른 비관계형 데이터 소스의 경우, 데이터 소스 파일(별칭으로 식별됨)이 특정 서버 오브젝트와 연관되도록 요구하는 페더레이티드 오브젝트의 계층 구조 때문에 서버 정의가 작성됩니다.

TYPE *server-type*

*server-name*으로 표시되는 데이터 소스의 유형을 지정합니다. 일부 랩퍼에서 이 매개변수는 필수입니다.

VERSION

*server-name*으로 표시되는 데이터 소스의 버전을 지정합니다. 일부 랩퍼에서 이 매개변수는 필수입니다.

version

버전 번호를 지정합니다. 이 값은 정수여야 합니다.

release

*version*으로 표시된 버전의 릴리스 번호를 지정합니다. 이 값은 정수여야 합니다.

mod

*release*로 표시된 릴리스의 수정 번호를 지정합니다. 이 값은 정수여야 합니다.

version-string-constant

버전의 전체 명칭을 지정합니다. *version-string-constant*는 단일 값(예: '8i')이거나 *version*, *release* 및 *mod*(적용 가능한 경우)의 조인된 값(예: '8.0.3')일 수 있습니다.

WRAPPER *wrapper-name*

*server-name*으로 지정된 서버 오브젝트와 상호 작용하기 위해 페더레이티드 서버가 사용하는 랩퍼 이름입니다.

AUTHORIZATION *remote-authorization-name*

DB2 계열 데이터 소스에 대해서만 필수입니다. CREATE SERVER문이 처리될 때 데이터 소스에서 필요한 조치를 수행하는 데 사용되는 권한 부여 ID를 지정합니다. 이 권한 부여 ID는 서버에 다음 연결을 설정할 때 쓰이지 않습니다.

이 ID는 필요한 조치가 요구하는 권한(BINDADD 또는 이에 상응하는 것)을 가지고 있어야 합니다. *remote-authorization-name*을 대소문자와 함께 사용해서 지정하거나 소문자로 지정하였으나 리모트 데이터 소스에 대소문자를 구분하는 권한 부여 이름이 있을 경우에는 *remote-authorization-name*을 큰따옴표로 묶어야 합니다.

PASSWORD *password*

DB2 계열 데이터 소스에 대해서만 필수입니다. *remote-authorization-name*으로 표시되는 권한 부여 ID와 연관된 암호를 지정합니다. *password*를 대소문자를 함께 사용해서 지정하거나 소문자로 지정했는데 리모트 데이터 소스에 대소문자를 구분하는 암호가 있을 경우에는 *password*를 큰따옴표로 묶어야 합니다.

OPTIONS

서버 정의가 작성될 때 사용 가능하게 되는 옵션을 나타냅니다. 서버 옵션은 서버 정의를 구성하는 데 사용됩니다. 일부 서버 옵션은 모든 데이터 소스에 대한 서버 정의를 작성하는 데 사용될 수 있습니다. 일부 서버 옵션은 특정 데이터 소스에 대해서만 사용됩니다.

ADD

하나 이상의 서버 옵션을 활성화합니다.

server-option-name

*server-name*으로 표시되는 데이터 소스에 대한 정보를 구성하거나 제공하는 데 사용될 서버 옵션의 이름을 지정합니다.

string-constant

*server-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

주

- 데이터 소스에 암호가 필요할 때에는 *password*를 지정해야 합니다. *password*에 있는 문자는 소문자여야 하며, *password*는 인용부호로 묶어야 합니다.
- CREATE SERVER문을 사용하여 DB2 계열 인스턴스를 데이터 소스로 정의한 경우, DB2는 특정 패키지를 해당 인스턴스에 바인드해야 합니다. 바인딩이 필요할 경우 명령문의 *remote-authorization-name*은 BIND 권한을 가지고 있어야 합니다. 바인드 조작을 완료하는 데 걸리는 시간은 데이터 소스 속도와 네트워크 연결 속도에 따라 결정됩니다.
- DB2는 지정된 서버 버전이 리모트 서버 버전과 일치한다는 것을 확인하지 않습니다. 부정확한 서버 버전을 지정하면 사용자가 DB2 서버 정의에 속한 별칭에 액세스할 때 SQL 오류가 나타날 수 있습니다. 이는 리모트 서버 버전보다 상위 서버 버전을 지정할 경우 나타날 확률이 높습니다. 이 경우 서버 정의에 속한 별칭에 액세스할 때 DB2는 리모트 서버가 인식하지 못하는 SQL을 보낼 수 있습니다.

예:

예 1: 서버 정의를 등록하여 z/OS 및 OS/390®용 DB2 버전 7.1 데이터 소스에 액세스하십시오. CRANDALL은 z/OS 및 OS/390용 DB2 서버 정의에 할당된 이름입니다. DRDA는 이 데이터 소스를 액세스하는 데 사용되는 랩퍼 이름입니다. 또한 다음과 같이 지정하십시오.

- GERALD 및 drowssap는 이 명령문이 처리될 때 CRANDALL에서 패키지를 바인드하는 데 사용되는 권한 부여 ID 및 암호입니다.
- CATALOG DATABASE문으로 지정된 z/OS 및 OS/390용 DB2 데이터베이스의 별칭은 CLIENTS390입니다.
- CRANDALL 액세스에 사용할 수 있는 권한 부여 ID 및 암호가 대문자로 CRANDALL에 전송됩니다.
- CLIENTS390 및 페더레이티드 데이터베이스는 동일한 조합 시퀀스를 사용합니다.

```
CREATE SERVER CRANDALL
  TYPE DB2/ZOS
  VERSION 7.1
  WRAPPER DRDA
  AUTHORIZATION "GERALD"
  PASSWORD drowssap
  OPTIONS
      (DBNAME 'CLIENTS390',
              FOLD_ID 'U',
              FOLD_PW 'U',
      COLLATING_SEQUENCE 'Y')
```

예 2: Oracle 9 데이터 소스에 액세스하는 서버 정의를 등록하십시오. Oracle 서버 정의에 지정된 이름은 CUSTOMERS입니다. NET8은 이 데이터 소스를 액세스하는 데 사용되는 랩퍼 이름입니다. 또한 다음과 같이 지정하십시오.

- ABC는 Oracle 데이터베이스 서버가 있는 노드 이름입니다.
- 페더레이티드 서버에 대한 CPU는 CUSTOMERS를 지원하는 CPU보다 두 배 빨리 실행됩니다.
- 페더레이티드 서버에 있는 입출력 디바이스는 CUSTOMERS에 있는 입출력 디바이스보다 1.5배 신속하게 데이터를 처리합니다.

```
CREATE SERVER CUSTOMERS
  TYPE ORACLE
  VERSION 9
  WRAPPER NET8
  OPTIONS
      (NODE 'ABC',
              CPU_RATIO '2.0',
      IO_RATIO '1.5')
```

예 3: Excel 랩퍼에 대한 서버 정의를 등록하십시오. 페더레이티드 오브젝트의 계층 구조를 보존하려면 서버 정의가 필요합니다. BIOCHEM_LAB은 Excel 서버 정의에 지정된 이름입니다. EXCEL_2000_WRAPPER는 이 데이터 소스에 액세스하는 데 사용되는 랩퍼 이름입니다.

```
CREATE SERVER BIOCHEM_DATA
  WRAPPER EXCEL_2000_WRAPPER
```

CREATE SYNONYM

CREATE SYNONYM문은 모듈, 별칭, 시퀀스, 테이블, 뷰 또는 다른 동의어에 대한 동의어를 정의합니다.

설명

SYNONYM은 ALIAS의 동의어입니다.

CREATE TABLE

CREATE TABLE문은 테이블을 정의합니다. 정의에는 컬럼의 이름과 속성이 포함되어야 합니다. 정의에는 기본 키나 점검 제한조건과 같은 테이블의 다른 속성이 포함될 수도 있습니다.

작성된 임시 테이블을 작성하려면 CREATE GLOBAL TEMPORARY TABLE 명령문을 사용하십시오. 선언된 임시 테이블을 선언하려면 DECLARE GLOBAL TEMPORARY TABLE 명령문을 사용하십시오.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권은 여기에 설명된 내용과 같이 추가 권한이 있는 조합에서 DBADM 권한 또는 CREATETAB 권한 중 하나를 포함해야 합니다.

- 다음 특권 및 권한 중 하나입니다.
  - 테이블 스페이스에 대한 USE 특권
  - SYSADM
  - SYSCTRL
- 다음 특권 및 권한에 하나를 더합니다.
  - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

서브테이블이 정의되고 있는 경우, 권한 부여 ID는 테이블 계층의 루트 테이블의 소유자와 동일해야 합니다.

외부 키를 정의하려면, 명령문의 권한 부여 ID에 보유된 특권은 상위 테이블에서 다음 중 하나를 포함해야 합니다.

- 테이블에 대해 REFERENCES 특권
- 지정된 상위 키의 각 컬럼에 대한 REFERENCES 특권
- 테이블에 대한 CONTROL 특권
- DBADM 권한

구체화된 쿼리 테이블을 정의하려면(fullselect 사용), 명령문의 권한 부여 ID가 보유하는 특권에 fullselect에서 식별된 각 테이블이나 뷰에 대한 최소한 다음 중 하나가 포함되어야 합니다(그룹 특권 제외).

- 테이블 또는 뷰에 대한 SELECT 특권
- 테이블 또는 뷰에 대한 CONTROL 특권
- DATAACCESS 권한

구체화된 쿼리 테이블을 정의하고 CREATE TABLE문의 특정 절을 지정하는 경우 추가적인 권한이 필요하거나 다음 대신 사용할 수 있습니다.

- WITH NO DATA가 지정되면, 다음 권한 중 최소한 하나 이상을 충족해야 합니다.
  - DBADM
  - SQLADM
  - EXPLAIN
- REFRESH DEFERRED 또는 REFRESH IMMEDIATE가 지정되면, fullselect에서 식별되는 각 테이블 또는 뷰에 대한 다음 특권 또는 권한 중 최소한 하나 이상이 필요합니다.
  - 테이블 또는 뷰에 대한 ALTER 특권
  - 테이블 또는 뷰에 대한 CONTROL 특권
  - DBADM 권한

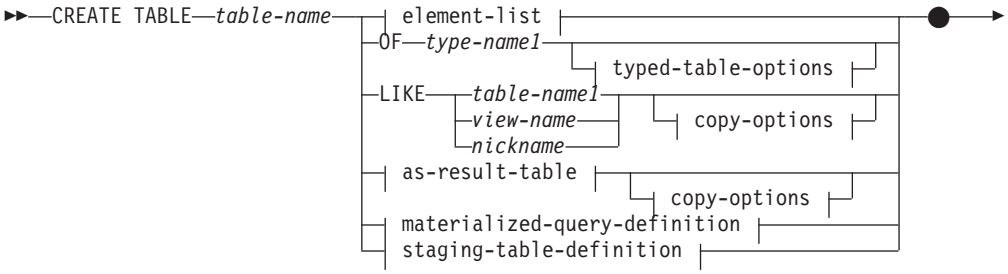
구체화된 쿼리 테이블과 연관된 스테이징 테이블을 정의하려면, 해당 명령문의 권한 부여 ID에서 갖고 있는 특권에는 구체화된 쿼리 테이블에 대해 적어도 다음 중 하나가 포함되어야 합니다.

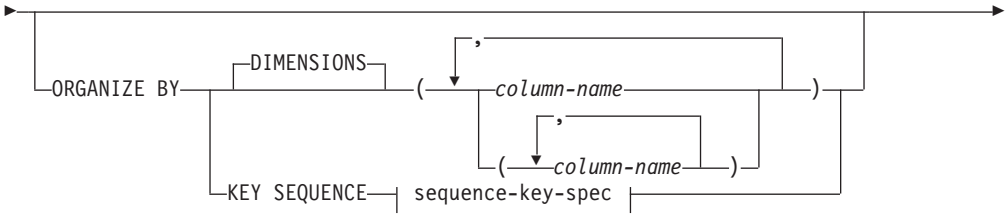
- 구체화된 쿼리 테이블에 대한 ALTER 특권
- 구체화된 쿼리 테이블에 대한 CONTROL 특권
- DBADM 권한

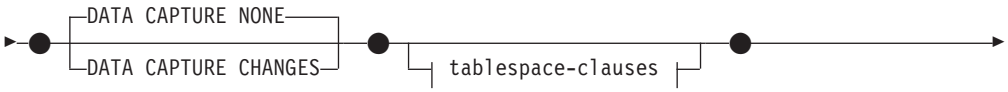
또한 구체화된 쿼리 테이블의 fullselect에서 식별되는 각 테이블이나 뷰에 대한 다음 특권 중 적어도 한 특권:

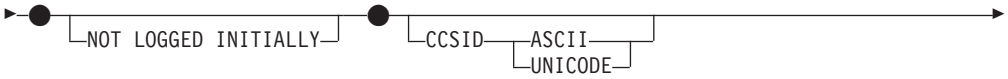
- 테이블이나 뷰에 대한 SELECT 특권이나 DATAACCESS 권한 및 다음 특권 중 적어도 한 특권:
  - 테이블 또는 뷰에 대한 ALTER 특권
  - DBADM 권한
- 테이블 또는 뷰에 대한 CONTROL 특권

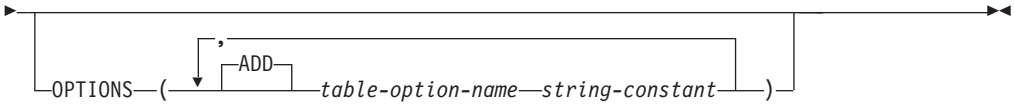
구문



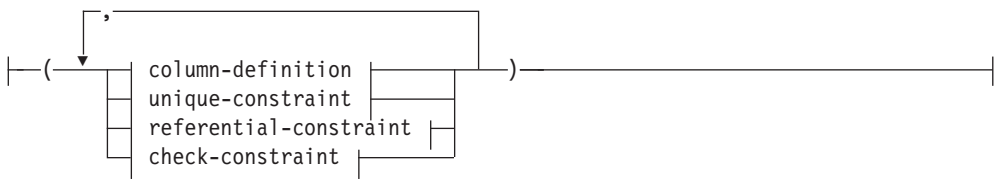




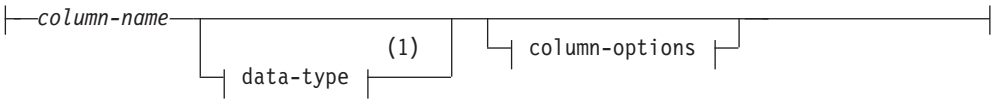




element-list:

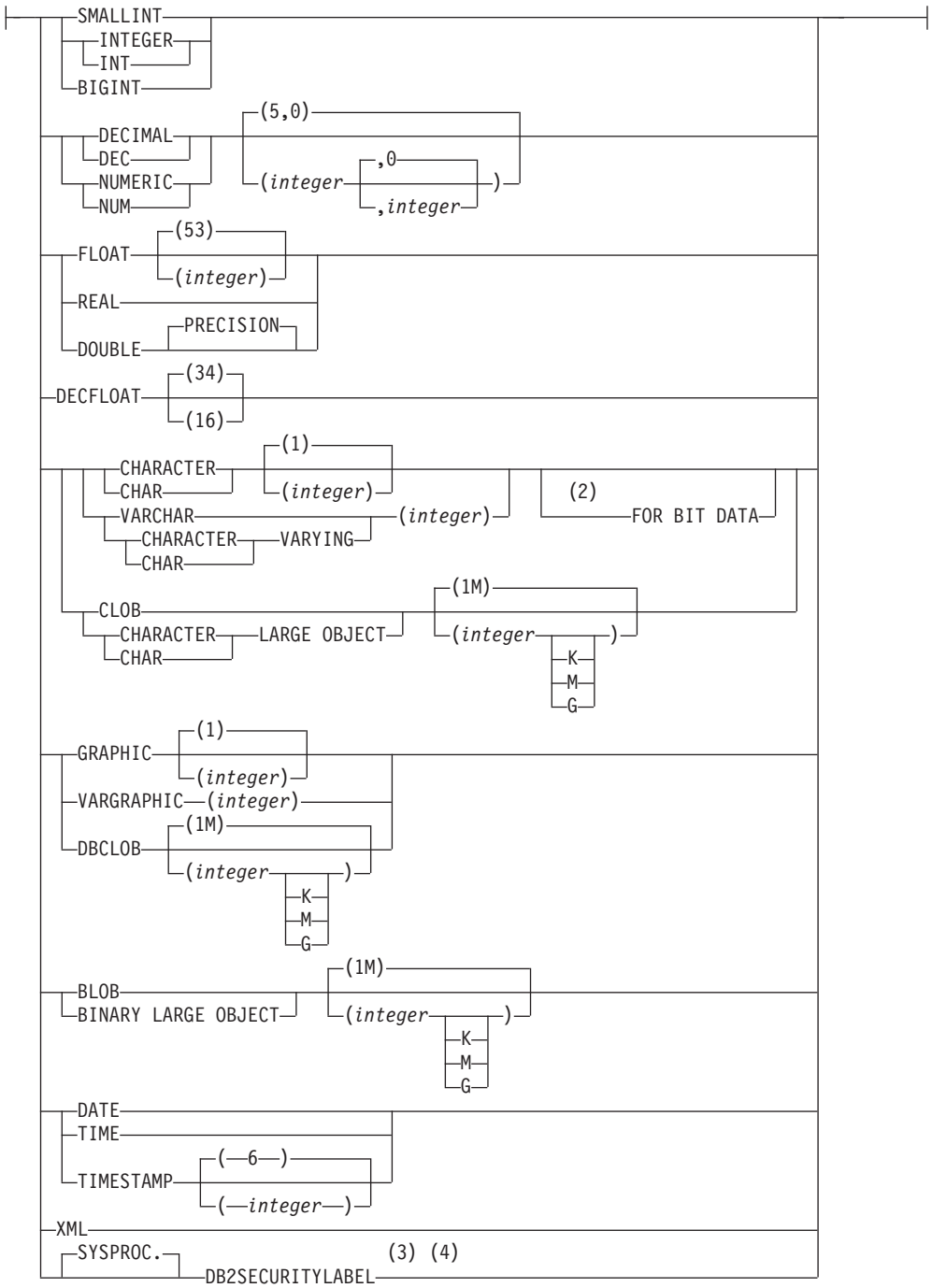


column-definition:

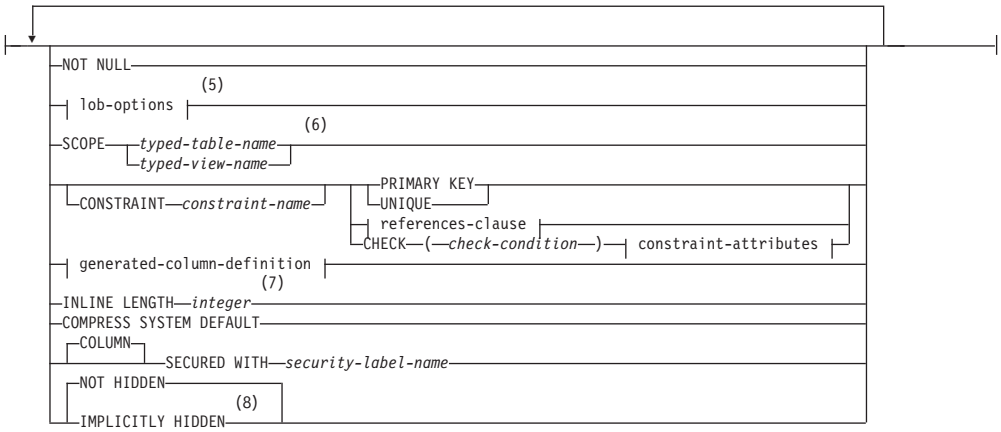


data-type:

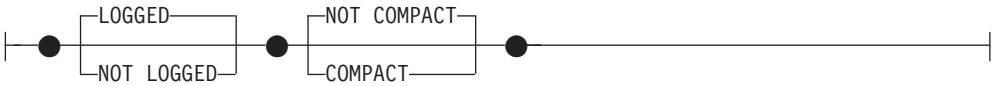
built-in-type:



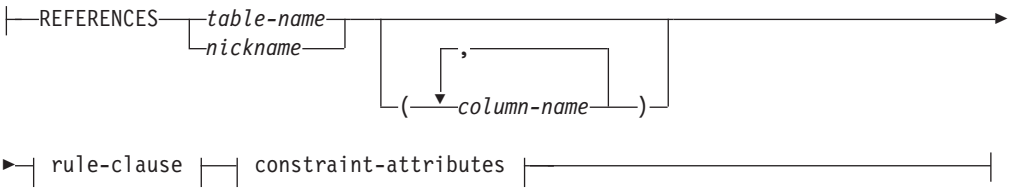
column-options:



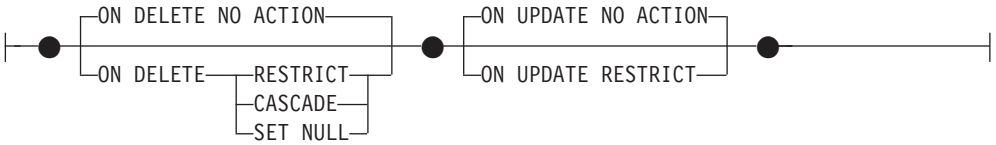
lob-options:



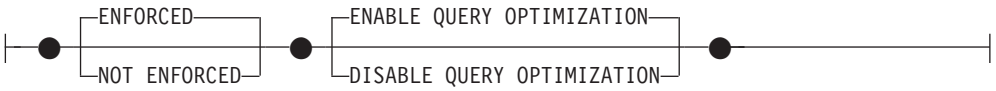
references-clause:



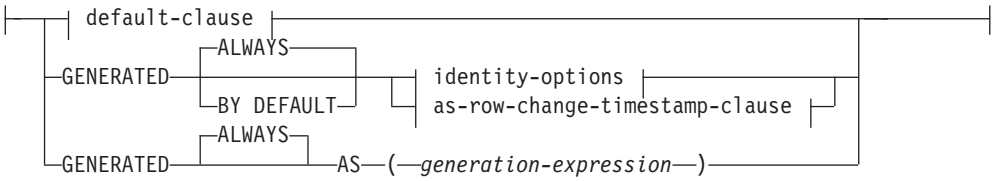
rule-clause:



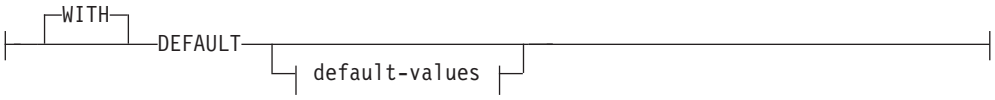
constraint-attributes:



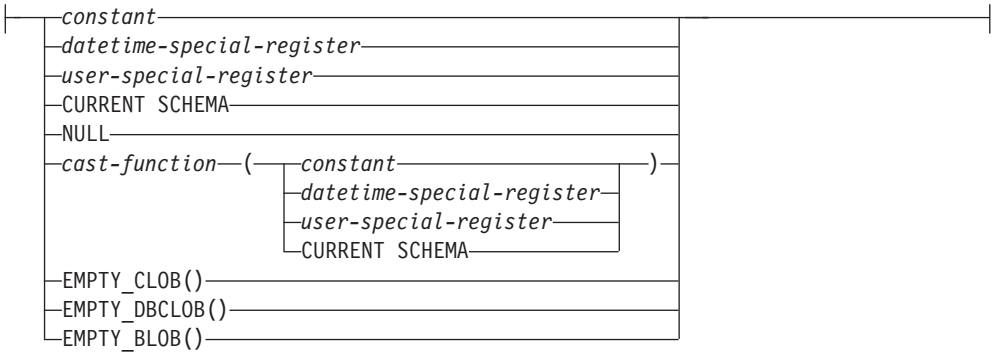
generated-column-definition:



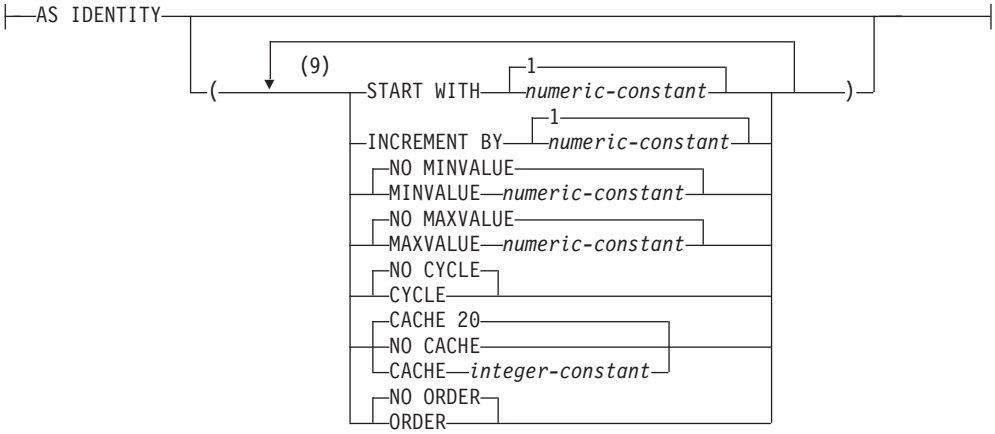
default-clause:



default-values:

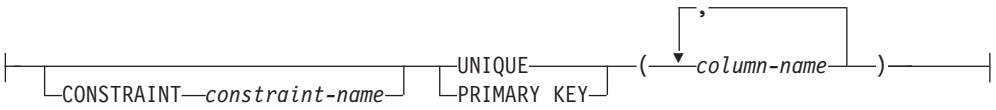


identity-options:



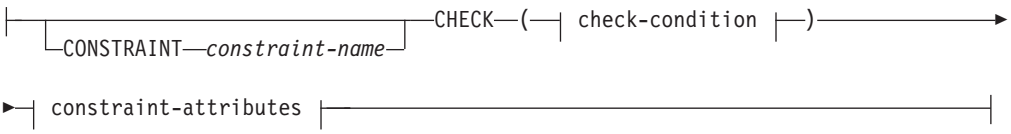
as-row-change-timestamp-clause:

unique-constraint:



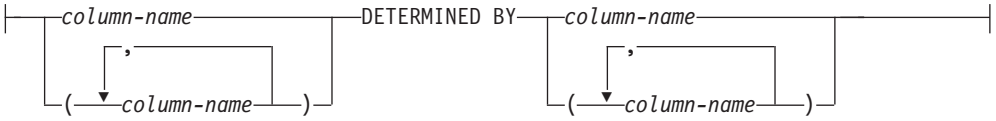
referential-constraint:

check-constraint:

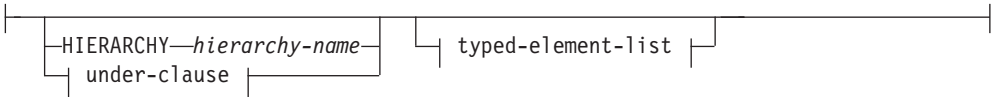


check-condition:

functional-dependency:

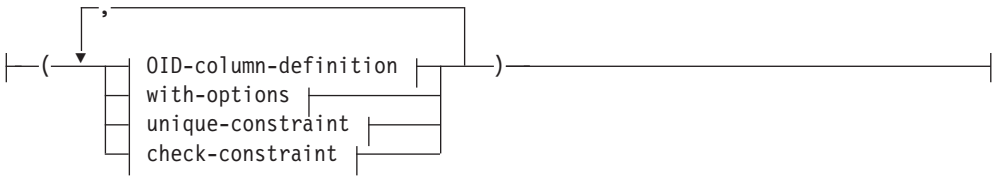


typed-table-options:



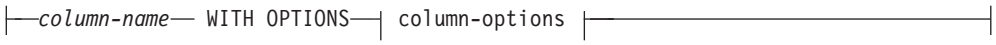
under-clause:

typed-element-list:

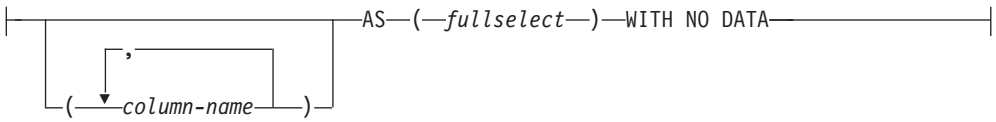


OID-column-definition:

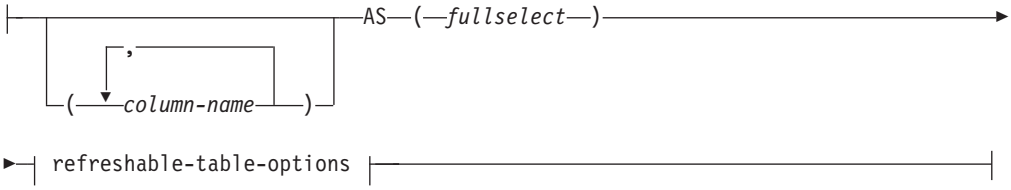
with-options:



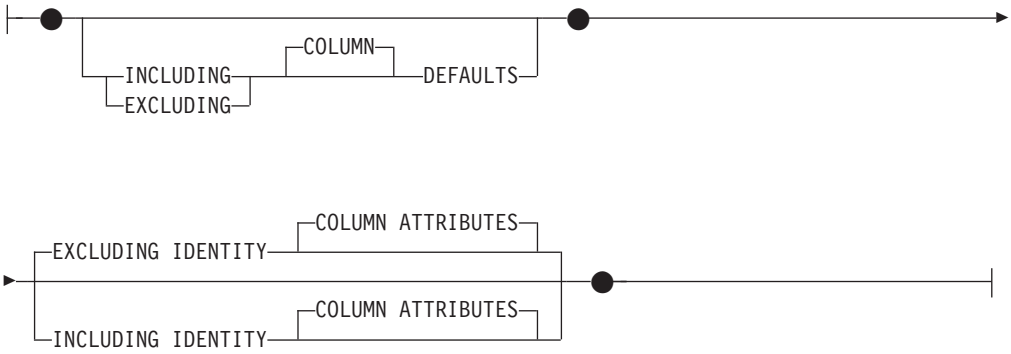
as-result-table:



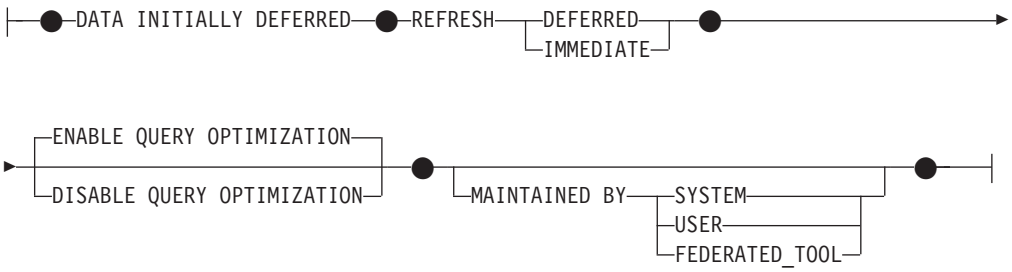
materialized-query-definition:



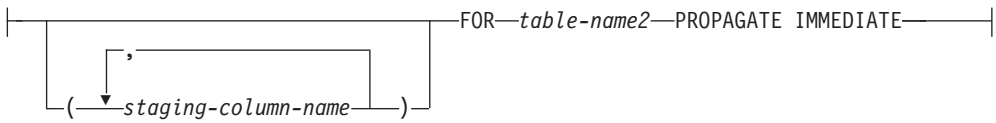
copy-options:



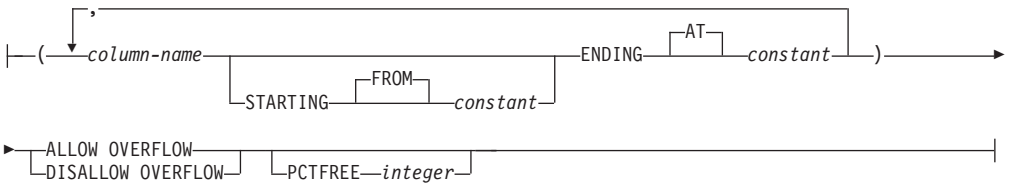
refreshable-table-options:



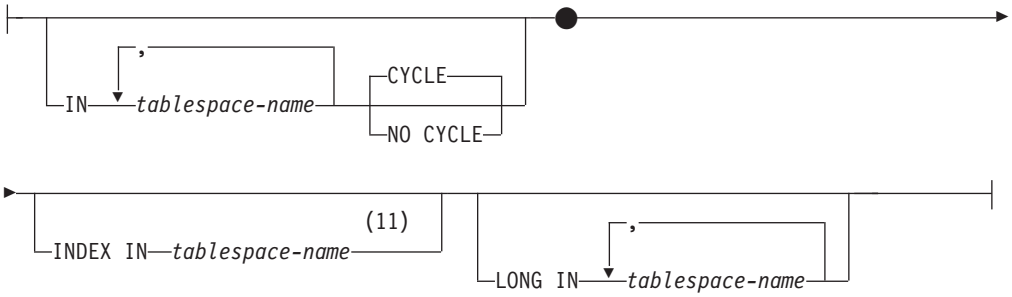
staging-table-definition:



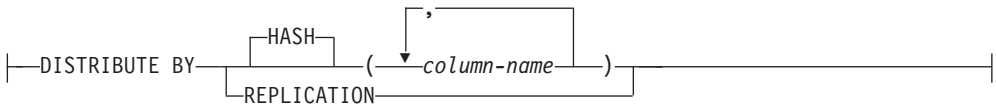
sequence-key-spec:



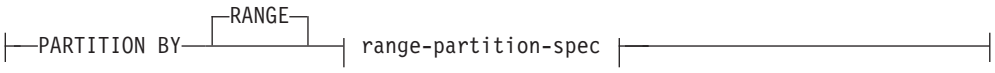
tablespace-clauses:



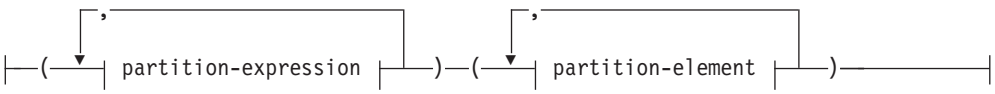
distribution-clause:



partitioning-clause:

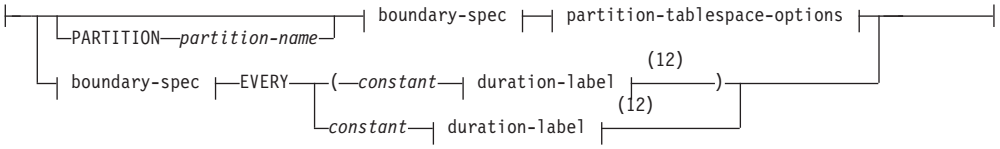


range-partition-spec:



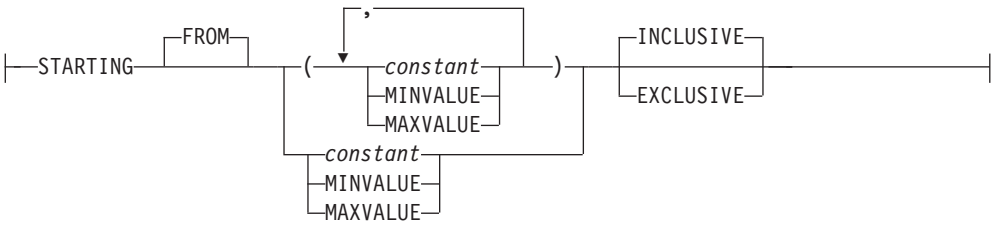
partition-expression:

partition-element:

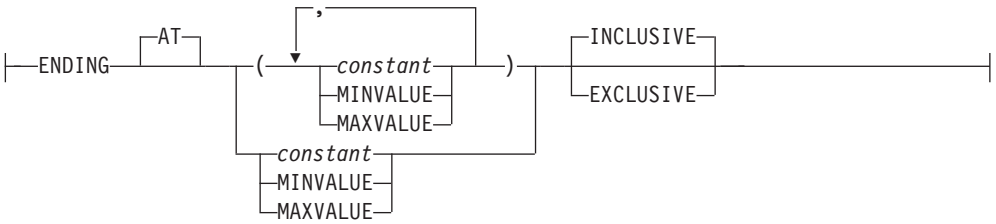


boundary-spec:

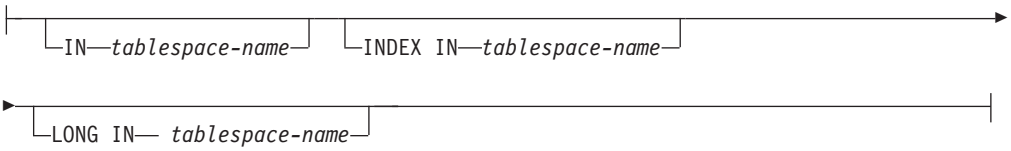
starting-clause:



ending-clause:



partition-tablespace-options:



duration-label:

주:

1 선택된 첫 번째 column-option이 generation-expression을 갖는 generated-column-definition인 경우에는 data-type을 생략할 수 있습니다. 이는 generation-expression의 결과 데이터 유형으로부터 결정됩니다.

2 FOR BIT DATA절은 뒤에 오는 다른 컬럼 제한조건과 함께 임의의 순서로 지정될 수 있습니다.

3 DB2SECURITYLABEL은 보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다.

4 DB2SECURITYLABEL 유형의 컬럼의 경우, NOT NULL WITH DEFAULT는 내재적 및 명시적으로 지정될 수 없습니다(SQLSTATE 42842). DB2SECURITYLABEL 유형의 컬럼 디폴트값은 쓰기 액세스에 대한 세션 권한 부여 ID의 보안 레이블입니다.

5 lob-options절은 대형 오브젝트 유형(BLOB, CLOB 및 DBCLOB)과 대형 오브젝트(LOB) 유형을 기반으로 하는 구별 유형에만 적용됩니다.

6 SCOPE절은 REF 유형에만 적용됩니다.

7 INLINE LENGTH는 구조화된 XML이나 LOB 유형으로 정의된 컬럼에만 적용됩니다.

8 IMPLICITLY HIDDEN은 ROW CHANGE TIMESTAMP도 지정된 경우에만 지정할 수 있습니다.

9 같은 절은 두 번 이상 지정될 수 없습니다.

10 지정된 첫 번째 컬럼 옵션이 생성된 컬럼 정의인 경우 데이터 유형은 행 변경 시간소인 컬럼에 대해 선택적입니다. 데이터 유형 디폴트는 TIMESTAMP(6)입니다.

11 테이블 작성시 테이블 인덱스를 포함할 테이블 스페이스를 지정할 수 있습니다. 테이블이 파티션된 테이블인 경우 파티션되지 않은 인덱스에 대한 인덱스 테이블 스페이스는 CREATE INDEX 명령문의 IN절로 지정될 수 있습니다.

12 partition-element에 대한 이 구문은 숫자 또는 날짜 시간 데이터 유형을 가진 하나의 partition-expression만 있는 경우 유효합니다.

13 첫 번째 partition-element는 starting-clause를 포함해야 하고 마지막 partition-element는 ending-clause를 포함해야 합니다.

설명

시스템에서 유지보수하는 구체화된 쿼리 테이블과 사용자가 유지보수하는 구체화된 쿼리 테이블은 두 테이블을 구분할 필요가 없다면 일반적으로 둘 다 *구체화된 쿼리 테이블*이라고 합니다.

table-name

테이블의 이름을 지정합니다. 내재적 또는 명시적인 규정자를 포함한 이름은 카탈로그에 기술된 테이블, 뷰, 별칭 또는 별명을 식별해서는 안됩니다. 스키마 이름은 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT이면 안됩니다(SQLSTATE 42939).

element-list

테이블의 요소를 정의합니다. 여기에는 테이블에 있는 컬럼 및 제한조건 정의가 포함됩니다.

column-definition

컬럼의 속성을 정의합니다.

column-name

테이블의 컬럼을 이름 지정합니다. 이름을 규정할 수 없으며 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

테이블은 다음 사항을 포함합니다.

- 최대 500개 컬럼의 4K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 4,005 이하여야 합니다.
- 최대 1,012개 컬럼의 8K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 8,101 이하여야 합니다.
- 최대 1,012개 컬럼의 16K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 16,293 이하여야 합니다.
- 최대 1,012개 컬럼의 32K 페이지 *크기*. 여기서, 컬럼의 바이트 수는 32,677 이하여야 합니다.

자세한 내용은 행 *크기* 제한을 참조하십시오.

data-type

컬럼의 데이터 유형을 지정합니다.

built-in-type

내장 유형의 경우 다음 유형 중 하나를 사용합니다.

SMALLINT

작은 정수

INTEGER 또는 INT

큰 정수

BIGINT

큰 정수(big integer)

DECIMAL*(precision-integer, scale-integer)* **또는 DEC***(precision-integer, scale-integer)*

10진수. 첫 번째 정수는 숫자의 정밀도(즉, 총 자릿수)로, 그 범위는 1 - 31이 될 수 있습니다. 두 번째 정수는 숫자의 스케일, 즉 소수점의 오른쪽에 있는 자릿수로, 그 범위는 0 - 숫자의 정밀도가 될 수 있습니다.

정밀도나 스케일을 지정하지 않으면, 디폴트값인 5,0이 사용됩니다. **NUMERIC**과 **NUM**은 **DECIMAL**과 **DEC**의 동의어로 사용될 수 있습니다.

FLOAT(*integer***)**

단정밀도 또는 배정밀도의 부동 소수점 숫자. *integer*의 값에 따릅니다. 정수 값은 1 - 53 범위 내에 있어야 합니다. 1 - 24의 값은 단정밀도를 나타내며, 25 - 53의 값은 배정밀도를 나타냅니다.

다음과 같이 지정할 수도 있습니다.

REAL

단정밀도 부동 소수점

DOUBLE

배정밀도 부동 소수점

DOUBLE PRECISION

배정밀도 부동 소수점

FLOAT

배정밀도 부동 소수점

DECFLOAT*(precision-integer)*

10진수 부동 소수점 숫자의 경우. *precision-integer*의 값은 숫자의 정밀도(즉, 총 자릿수)이며, 그 값은 16 또는 34입니다.

정밀도가 지정되지 않으면 디폴트값 34가 사용됩니다.

CHARACTER*(integer)* **또는 CHAR***(integer)* **또는 CHARACTER 또는 CHAR**

길이 *integer* 바이트의 고정 길이 문자열의 경우 범위는 1 - 254가 될 수 있습니다. 길이를 지정하지 않으면, 1자로 간주됩니다.

VARCHAR*(integer)* **또는 CHARACTER VARYING***(integer)***, 또는 CHAR VARYING***(integer)*

최대 길이 *integer* 바이트의 고정 길이 문자열의 경우 범위는 1 - 32,672가 될 수 있습니다.

FOR BIT DATA

비트(2진) 데이터로 취급되는 컬럼의 내용을 지정합니다. 다른 시스템과의 데이터 교환 중에는 코드 페이지 변환이 수행되지 않습니다. 비교는 데이터베이스 배열 순서와 상관없이 2진으로 수행됩니다.

CLOB 또는 CHARACTER (CHAR) LARGE OBJECT*(integer [K | M | G])*

지정된 최대 길이(바이트)의 문자 대형 오브젝트(CLOB) 문자열.

*integer K | M | G*의 의미는 BLOB과 유사합니다.

길이 스펙이 생략되면 길이는 1,048,576(1MB)으로 간주됩니다.

1GB 이상의 CLOB 문자열을 작성하려면, NOT LOGGED 옵션을 지정해야 합니다.

CLOB 컬럼에 FOR BIT DATA절을 지정할 수 없습니다. 그러나 CHAR FOR BIT DATA 문자열은 CLOB 컬럼에 지정할 수 있으며, CHAR FOR BIT DATA 문자열을 CLOB 문자열과 연결할 수 있습니다.

GRAPHIC*(integer)*

1 - 127 범위에 있을 수 있는 *integer* 길이의 고정 길이 그래픽 문자열. 길이를 지정하지 않으면, 1자로 간주됩니다.

VARGRAPHIC*(integer)*

최대 길이 *integer*의 가변 길이 그래픽 문자열의 경우, 범위는 1 - 16,336이 될 수 있습니다.

DBCLOB*(integer [K | M | G])*

지정된 최대 길이(2바이트 문자)의 2바이트 문자 대형 오브젝트(LOB) 문자열.

*integer K | M | G*의 의미는 BLOB과 유사합니다. 차이점은 지정된 숫자가 2바이트 문자의 수인 것과 최대 크기가 1,073,741,823개의 2 바이트 문자라는 점입니다.

길이 스펙이 생략되면, 길이는 1,048,576개의 2바이트 문자로 간주됩니다.

1GB 이상의 DBCLOB 문자열을 지정하려면 NOT LOGGED 옵션을 지정해야 합니다.

BLOB 또는 BINARY LARGE OBJECT*(integer [K | M | G])*

지정된 최대 길이(바이트)의 실행 파일 대형 오브젝트(BLOB) 문자열.

길이는 1 - 2,147,483,647바이트 범위일 수 있습니다.

*integer*만 지정된 경우, 최대 길이를 나타냅니다.

integer K(대문자 또는 소문자)를 지정한 경우, 최대 길이는 1,024와 *integer*를 곱한 값입니다. *integer*의 최대 값은 2,097,152입니다.

*integer M*을 지정한 경우, 최대 길이는 1,048,576과 *integer*를 곱한 값입니다. *integer*의 최대값은 2,048입니다.

*integer G*를 지정한 경우 최대 길이는 1,073,741,824와 *integer*를 곱한 값입니다. *integer*의 최대값은 2입니다.

2,147,483,648로 계산되는 K, M 또는 G의 배수를 지정한 경우, 사용되는 실제 값은 2,147,483,647(또는 2GB에서 1바이트를 뺀 값)이고, 이것이 LOB 컬럼의 최대 길이입니다.

길이 스펙이 생략되면 길이는 1,048,576(1MB)으로 간주됩니다.

1GB 이상의 BLOB 문자열을 작성하려면, NOT LOGGED 옵션을 지정해야 합니다.

정수와 K, M, G 사이에 임의 수의 스페이스가 허용되는데, 스페이스는 필수적이지 않습니다. 예를 들어, 다음은 모두 유효합니다.

```
BLOB(50K)    BLOB(50 K)    BLOB (50   K)
```

DATE

날짜

TIME
시간

TIMESTAMP(*정수*) 또는 TIMESTAMP
시간소인*정수*는 0과 12 사이여야 하며 0(초)부터 12(피코초)까지의 소수 초 정밀도를 지정해야 합니다. 디폴트는 6(마이크로초)입니다.

XML
XML 문서에 해당합니다. 바르게 구성된 XML 문서만이 XML 컬럼에 삽입될 수 있습니다.

XML 컬럼에는 다음과 같은 제한사항이 있습니다.

- 컬럼은 XML 데이터에 있는 인덱스를 제외하고 인덱스의 일부가 될 수 없습니다. 따라서 기본 키나 고유 제한조건 컬럼으로 포함될 수 없습니다(SQLSTATE 42962).
- 컬럼은 참조 제한조건의 외부 키가 될 수 없습니다(SQLSTATE 42962).
- 컬럼에 대해 디폴트값(WITH DEFAULT)이 지정될 수 없습니다(SQLSTATE 42613). 널(NULL) 입력 가능한 컬럼의 경우, 컬럼의 디폴트값은 널(NULL)입니다.
- 컬럼은 분산 키로 사용할 수 없습니다(SQLSTATE 42997).
- 컬럼은 데이터 파티셔닝 키로 사용할 수 없습니다(SQLSTATE 42962).
- 컬럼은 다차원적으로 클러스터된(MDC) 테이블 구성에 사용할 수 없습니다(SQLSTATE 42962).
- 컬럼은 범위 클러스터된 테이블에서 사용될 수 없습니다(SQLSTATE 429BG).
- 컬럼은 VALIDATED 술어의 경우를 제외한 점검 제한조건에서 참조될 수 없습니다(SQLSTATE 42621).

XML 유형의 컬럼이 작성되면 XML 경로 인덱스가 해당 컬럼에 작성됩니다. XML 유형의 첫 번째 컬럼이 작성되면 테이블 레벨 XML 지역 인덱스도 작성됩니다. 해당 인덱스의 이름은 'SQL' 다음에 문자 시간소인(*yymmddhhmmssxxx*)이 오는 형식입니다. 스키마 이름은 SYSIBM입니다.

SYSPROC.DB2SECURITYLABEL
보호된 테이블의 행 보안 레이블 컬럼을 정의하는 데 사용해야 하는 내장 구별 유형입니다. DB2SECURITYLABEL 내장 구별 유형의 컬럼에 있는 하부 데이터 유형은 VARCHAR(128) FOR BIT DATA입니다. 테이블에는 많아야 DB2SECURITYLABEL 유형 컬럼 하나가 있을 수 있습니다(SQLSTATE 428C1).

distinct-type-name

구별 유형인 사용자 정의 유형의 경우. 구별 유형 이름을 스키마 이름없이 지정하면, 구별 유형 이름은 SQL 경로(정적 SQL의 경우 FUNCPATH 사전 처리 옵션에 의해 정의되고, 동적 SQL의 경우 CURRENT PATH 레지스터에 의해 정의됨)의 스키마를 검색하여 분석됩니다.

구별 유형을 사용하여 정의된, 컬럼의 데이터 유형은 구별 유형입니다. 컬럼의 길이와 스케일은 각각 구별 유형의 소스 유형의 길이와 스케일입니다.

구별 유형을 사용하여 정의된 컬럼이 참조 제한조건의 외부 키이면, 기본 키의 해당 컬럼에 대한 데이터 유형이 같은 구별 유형을 가지고 있어야 합니다.

structured-type-name

구조화된 유형인 사용자 정의 유형의 경우. 구조화된 유형 이름을 스키마 이름없이 지정하면, 구조화된 유형 이름은 SQL 경로(정적 SQL의 경우 FUNCPATH 사전 처리 옵션에 의해 정의되고, 동적 SQL의 경우 CURRENT PATH 레지스터에 의해 정의됨)에서 스키마를 검색하여 분석됩니다.

구조화된 유형을 사용하여 정의된, 컬럼의 정적 데이터 유형은 구조화된 유형입니다. 컬럼은 *structured-type-name*의 부속 유형인 동적 유형 값을 포함할 수 있습니다.

구조화된 유형을 사용하여 정의된 컬럼은 기본 키, 고유 제한조건, 외부 키, 인덱스 키 또는 분산 키에 사용될 수 없습니다(SQLSTATE 42962).

컬럼이 구조화된 유형을 사용하여 정의되고 중첩 레벨에서 참조 유형 속성을 포함하는 경우, 해당 참조 유형 속성의 범위가 지정되지 않습니다. 비참조 조작에 이러한 속성을 사용하려면, CAST 스펙을 사용하여 명시적으로 SCOPE를 지정해야 합니다.

REF (*type-name2*)

유형이 지정된 테이블에 대한 참조의 경우. *type-name2*를 스키마 이름없이 지정하면, 유형 이름은 SQL 경로(정적 SQL의 경우 FUNCPATH 사전 처리 옵션에 의해 정의되고, 동적 SQL의 경우 CURRENT PATH 레지스터에 의해 정의됨)에서 스키마를 검색하여 분석됩니다. 컬럼의 하위 데이터 유형은 *type-name2*에 대한 CREATE TYPE문의 REF USING절에 지정된 표현 데이터 유형이나 *type-name2*를 포함하는 데이터 유형 계층의 루트 유형을 기초로 합니다.

column-options

테이블 컬럼과 관련된 추가 옵션을 정의합니다.

NOT NULL

컬럼에 널(NULL)값이 포함되지 못하도록 합니다.

NOT NULL을 지정하지 않으면 컬럼에 널(NULL) 값이 포함될 수 있습니다. 디폴트값은 널(NULL) 값이거나 WITH DEFAULT절에서 제공하는 값입니다.

NOT HIDDEN 또는 IMPLICITLY HIDDEN

컬럼이 숨김으로 정의되어 있는지 여부를 지정합니다. 숨겨진 속성은 컬럼이 테이블에 대한 내재적 참조에 포함되어 있는지 여부 또는 명시적으로 SQL문을 참조할 수 있는지 여부를 판별합니다. 디폴트값은 NOT HIDDEN입니다.

NOT HIDDEN

컬럼이 테이블에 대한 내재적 참조에 포함되는지, 그리고 컬럼이 명시적으로 참조될 수 있는지를 지정합니다.

IMPLICITLY HIDDEN

컬럼이 명시적으로 이름으로 참조되지 않는 경우 해당 컬럼이 SQL문에서 볼 수 없음을 지정합니다. 예를 들어, 테이블에 IMPLICITLY HIDDEN 절로 정의된 컬럼이 포함되어 있다고 가정하면, SELECT * 에는 내재적으로 숨겨진 컬럼이 포함되지 않습니다. 그러나 내재적으로 숨겨진 컬럼의 이름을 명시적으로 참조하는 SELECT의 결과에 결과 테이블의 해당 컬럼이 포함됩니다.

IMPLICITLY HIDDEN은 ROW CHANGE TIMESTAMP 컬럼에 대해서만 지정되어야 합니다(SQLSTATE 42867). ROW CHANGE TIMESTAMP FOR *table-designator* 표현식은 IMPLICITLY HIDDEN ROW CHANGE TIMESTAMP 컬럼을 해결합니다.

IMPLICITLY HIDDEN은 테이블의 모든 컬럼에 대해 지정되지 않아야 합니다(SQLSTATE 428GU).

lob-options

LOB 데이터 유형에 대한 옵션을 지정합니다.

LOGGED

컬럼에 대한 변경사항이 로그에 기록되도록 지정합니다. 이러한 컬럼의 데이터는 데이터베이스 유틸리티(예: RESTORE DATABASE)를 사용하여 복구 가능합니다. 디폴트값 LOGGED입니다.

1GB보다 큰 LOB은 기록할 수 없습니다(SQLSTATE 42993).

NOT LOGGED

컬럼에 대한 변경사항이 로그에 기록되지 않도록 지정합니다. 인라인되지 않는 LOB 데이터에만 적용됩니다.

NOT LOGGED는 커미트 또는 롤백 조작에 영향을 주지 않습니다. 즉, 트랜잭션이 롤백되는 경우에도 LOB 값의 기록 여부에 관계없이 데이터베이스의 일관성이 유지됩니다. 기록하지 않는다는 것은 백업이나 로

드 작업 다음에 롤 포워드 작업을 수행하는 동안 LOB 데이터가 롤 포워드 도중에 재생된 로그 레코드를 갖고 있었던 LOB 값에 대해 0으로 대체됨을 의미합니다. 응급 복구 시, 모든 커미트된 변경사항과 롤백된 변경사항은 예상되는 결과를 반영합니다.

COMPACT
후속 추가 조작이 용이하도록 LOB 스토리지 끝에 스페이스를 남겨두는 대신 LOB 컬럼의 값이 최소한의 디스크 스페이스를 차지하도록 지정합니다(LOB 값에 사용되는 마지막 그룹의 추가 디스크 페이지를 해제함). 이 방법으로 데이터를 저장하면 컬럼에 대한 추가(길이 증가) 조작 시 성능이 저하됩니다.

NOT COMPACT
나중에 컬럼의 LOB 값을 쉽게 변경할 수 있도록 삽입 스페이스를 지정합니다. 이는 디폴트값입니다.

SCOPE
참조 유형 컬럼 영역을 식별합니다.

비참조 연산자의 왼쪽 피연산자 또는 DEREF 함수의 인수로서 사용될 예정인 컬럼에 대한 영역이 지정되어야 합니다. 일반적으로 상호 참조 테이블인 경우, 참조 유형 컬럼의 범위를 지정하면 이후의 ALTER TABLE문을 따르게 되어 목표 테이블이 정의될 수 있도록 합니다.

typed-table-name
유형이 지정된 테이블의 이름입니다. 이미 테이블이 있어야 하고, 그렇지 않은 경우 작성되는 테이블 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-table-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하는지 확인하기 위해 *column-name*에 지정된 값을 점검하지 않습니다.

typed-view-name
유형이 지정된 뷰의 이름입니다. 이미 뷰가 있어야 하고, 그렇지 않은 경우 작성되는 뷰 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서, *S*는 *typed-view-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하는지 확인하기 위해 *column-name*에 지정된 값을 점검하지 않습니다.

CONSTRAINT *constraint-name*
제한조건이 이름을 지정합니다. *constraint-name*은 같은 CREATE TABLE 문 내에 이미 지정된 제한조건을 식별해서는 안됩니다 (SQLSTATE 42710).

이 절이 생략되는 경우, 시스템은 테이블에 정의된 기존의 제한조건 ID 가운데 고유한 18바이트의 ID를 생성합니다. 이 ID는 ″SQL″과 그 뒤에 오는 시간소인 함수가 생성한 15개의 숫자로 구성됩니다.

PRIMARY KEY 또는 UNIQUE 제한조건과 함께 사용할 경우, 제한조건을 지원하기 위해 작성된 인덱스의 이름으로 *constraint-name*을 사용할 수 있습니다.

PRIMARY KEY

단일 컬럼으로 구성되는 기본 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 PRIMARY KEY가 C 컬럼 정의에 지정된 경우, PRIMARY KEY(C)를 별도의 절로 지정한 경우와 그 결과가 같습니다.

테이블이 서브테이블인 경우, 기본 키가 슈퍼 테이블로부터 상속하게 되므로 기본 키를 지정할 수 없습니다(SQLSTATE 429B3).

ROW CHANGE TIMESTAMP 컬럼은 기본 키의 일부로 사용될 수 없습니다(SQLSTATE 429BV).

다음의 *unique-constraint* 설명에 있는 PRIMARY KEY를 참조하십시오.

UNIQUE

단일 컬럼으로 구성되는 고유 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 UNIQUE가 C 컬럼 정의에 지정된 경우, UNIQUE(C)를 별도의 절로 지정한 경우와 그 결과가 같습니다.

테이블이 서브테이블인 경우, 고유 제한조건은 슈퍼 테이블로부터 상속되므로 고유 제한조건을 지정할 수 없습니다(SQLSTATE 429B3).

다음의 *unique-constraint* 설명에 있는 UNIQUE를 참조하십시오.

references-clause

단일 컬럼으로 구성되는 외부 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로 references-clause가 C 컬럼의 정의에 지정된 경우, C가 유일하게 식별되는 컬럼인 FOREIGN KEY절의 일부로 references-clause를 지정한 것과 그 결과가 같습니다.

다음의 *referential-constraint* 아래에 있는 *references-clause*를 참조하십시오.

CHECK (*check-condition***)**

단일 컬럼에 적용되는 점검 제한조건을 정의함에 있어 간편한 방법을 제공합니다. 아래의 CHECK (*check-condition*)을 참조하십시오.

generated-column-definition

컬럼에 대해 생성된 값을 지정합니다.

default-clause

컬럼의 디폴트값을 지정합니다.

WITH
선택적 키워드입니다.

DEFAULT
값이 INSERT에 제공되지 않거나 INSERT 또는 UPDATE의 DEFAULT로 지정된 이벤트에서 디폴트값을 제공합니다. 디폴트값을 DEFAULT 키워드 다음에 지정하지 않은 경우, 디폴트값은 『ALTER TABLE』에 표시된 컬럼의 데이터 유형에 따라 달라집니다.

컬럼이 XML로 정의되는 경우 디폴트값을 지정할 수 없습니다 (SQLSTATE 42613). 가능한 디폴트값은 NULL뿐입니다.

컬럼이 유형이 지정된 테이블 컬럼을 기반으로 하는 경우, 디폴트값 정의시 특정 디폴트값을 지정해야 합니다. 유형이 지정된 테이블의 오브젝트 ID 컬럼에 대한 디폴트값은 지정할 수 없습니다 (SQLSTATE 42997).

컬럼이 구별 유형을 사용하여 정의되면, 컬럼의 디폴트값은 구별 유형으로 캐스트되는 소스 데이터 유형의 디폴트값입니다.

구조화된 유형을 사용하여 컬럼을 정의한 경우, *default-clause*를 지정할 수 없습니다(SQLSTATE 42842).

*column-definition*에서 DEFAULT를 생략하면, 컬럼의 디폴트값으로 널(NULL)값이 사용됩니다. 그러한 컬럼은 NOT NULL로 정의할 경우 유효 디폴트값을 갖지 않습니다.

default-values
지정될 수 있는 디폴트값의 특정 유형은 다음과 같습니다.

constant
컬럼의 디폴트값으로 상수를 지정합니다. 지정된 상수는 다음과 같아야 합니다.

- 지정된 규칙에 따라 컬럼에 지정할 수 있는 값을 나타내야 합니다.
- 컬럼이 부동 소수점 데이터 유형으로 정의되어 있지 않는 한, 부동 소수점 상수가 아니여야 합니다.
- 컬럼의 데이터 유형이 10진수 부동 소수점인 경우 숫자 상수 또는 10진수 부동 소수점 특수 값이어야 합니다. 목표 컬럼이 DECFLOAT이면 부동 소수점 상수는 먼저 DOUBLE로 해석된 다음 10진수 부동 소수점으로 변환됩니다. DECFLOAT(16) 컬럼의 경우, 16보다 정밀도가 큰 10진 상수는 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터에서 지정된 근사값 모드를 사용하여 반올림됩니다.

- 상수가 10진 상수인 경우 컬럼 데이터 유형의 전체 자릿수를 초과하는 0이 아닌 자릿수를 갖지 않아야 합니다(예를 들면, 1.234는 DECIMAL(5,2) 컬럼에 대한 디폴트값이 될 수 없음).
- 따옴표, 16진 상수를 나타내는 X와 같은 도입 문자 및 상수가 *cast-function*의 인수일 때 완전한 함수 이름의 문자 및 괄호를 포함하여 254바이트 이하로 표시됩니다.

datetime-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 날짜 시간 특수 레지스터의 값(CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP)을 지정합니다. 컬럼의 데이터 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다(예: CURRENT DATE 지정시 DATE여야 함).

user-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시에 사용자 특수 레지스터의 값(CURRENT USER, SESSION_USER, SYSTEM_USER)을 지정합니다. 컬럼의 데이터 유형은 사용자 특수 레지스터의 길이 속성보다 짧지 않은 길이의 문자열이어야 합니다. SESSION_USER 대신 USER를, CURRENT USER 대신 CURRENT_USER를 지정할 수 있습니다.

CURRENT SCHEMA

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 CURRENT SCHEMA 특수 레지스터값을 지정합니다. CURRENT SCHEMA가 지정되면, 컬럼의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성과 같거나 그 이상인 문자열이어야 합니다.

NULL

컬럼의 디폴트값으로 널(NULL)을 지정합니다. NOT NULL이 지정되었으면, DEFAULT NULL은 동일한 컬럼 정의 내에 지정될 수 있으나, 이로 인해 디폴트값으로 컬럼을 설정하려고 시도할 때 오류가 발생합니다.

cast-function

이 형식의 디폴트값은 구별 유형, BLOB 또는 날짜 시간(DATE, TIME 또는 TIMESTAMP) 데이터 유형으로 정의된 컬럼과 함께 사용할 수 있습니다. 구별 유형의 경우, BLOB 또는 날짜 시간 유형을 따르는 구별 유형을 제외하고는 함수 이

름이 컬럼의 구별 유형 이름과 일치해야 합니다. 스키마 이름으로 규정된 경우, 함수 이름이 구별 유형에 대한 스키마 이름과 같아야 합니다. 규정되지 않은 경우, 함수 결정의 스키마 이름은 구별 유형에 대한 스키마 이름과 같아야 합니다. 디폴트 값이 상수인 날짜 시간 유형을 따르는 구별 유형의 경우, 함수가 사용되어야 하며, 함수 이름은 내재적 또는 명시적 스키마 이름이 SYSIBM인 구별 유형의 소스 유형 이름과 일치해야 합니다. 다른 날짜 시간 컬럼의 경우, 해당하는 날짜 시간 함수가 사용될 수도 있습니다. BLOB 또는 BLOB을 기반으로 하는 구별 유형의 경우, 함수가 사용되어야 하며 함수 이름은 내재적 또는 명시적 스키마 이름으로 SYSIBM을 사용하는 BLOB이어야 합니다.

constant

상수를 인수로 지정합니다. 상수는 구별 유형의 소스 유형 또는 데이터 유형(구별 유형이 아닌 경우)에 대한 상수 규칙을 따라야 합니다. *cast-function*이 BLOB인 경우, 상수는 문자열 상수여야 합니다.

datetime-special-register

CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다.

user-special-register

CURRENT USER, SESSION_USER 또는 SYSTEM_USER를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 최소한 8바이트 길이의 문자열 데이터 유형이어야 합니다. *cast-function*이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

CURRENT SCHEMA

CURRENT SCHEMA 특수 레지스터의 값을 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성보다 크거나 같은 문자열이어야 합니다. cast-function이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

EMPTY_CLOB(), EMPTY_DBCLOB() 또는 EMPTY_BLOB()

컬럼의 디폴트값으로 영(0) 길이의 문자열을 지정합니다. 컬럼의 데이터 유형은 함수의 결과 데이터 유형을 기반으로 해야 합니다.

지정된 값이 유효하지 않으면, 오류가 발생합니다(SQLSTATE 42894).

GENERATED

DB2가 컬럼의 값을 생성함을 표시합니다. 컬럼이 IDENTITY 컬럼 또는 ROW CHANGE TIMESTAMP 컬럼으로 간주되는 경우, GENERATED를 지정해야 합니다.

ALWAYS

행이 테이블에 삽입될 때 또는 *generation-expression*의 결과 값이 변경될 때마다 DB2에서 항상 컬럼의 값을 생성하도록 지정합니다. 표현식의 결과는 테이블에 저장됩니다. GENERATED ALWAYS는 데이터 전파 또는 언로드/다시 로드 조작을 수행하지 않는 경우에 권장하는 값입니다. GENERATED ALWAYS는 생성된 컬럼의 필수 값입니다.

BY DEFAULT

명시적 값이 지정되어 있지 않은 경우 DEFAULT절을 지정하여 행을 삽입하거나 갱신할 때 DB2에서 컬럼의 값을 생성하도록 지정합니다. BY DEFAULT는 데이터 전파 사용시 또는 언로드와 재로드 조작 수행시 권장되는 값입니다.

명시적으로 요구되지는 않지만 값의 고유성을 보장하려면 생성된 IDENTITY 컬럼에 대해 고유한 단일 컬럼 인덱스를 정의합니다.

AS IDENTITY

컬럼이 이 테이블의 식별 컬럼이 되도록 지정합니다. 하나의 테이블은 하나의 IDENTITY 컬럼만 가질 수 있습니다(SQLSTATE 428C1). IDENTITY 키워드는 해당 컬럼과 연관된 데이터 유형이 스케일 0을 갖는 정확한 숫자 유형이거나, 소스 유형이 스케일 0을 갖는 정확한 숫자 유형인 사용자 정의 구별 유형인 경우에만 지정할 수 있습니다(SQLSTATE 42815). 스케일 0을 갖는 SMALLINT, INTEGER, BIGINT 또는 DECIMAL이나 이러한 유형 중 하나를 기반으로 하는 구별 유형은 정확한 숫자 유형으로 고려됩니다. 대조적으로, 단정밀도 또는 배정밀도 부동 소수점은 근사치의 숫자 데이터 유형으로 간주됩니다. 참조 유형은 정확한 숫자로 표시되더라도 식별 유형으로 정의할 수 없습니다.

식별 컬럼은 내재적으로 NOT NULL입니다. 식별 컬럼은 DEFAULT 절을 포함할 수 없습니다(SQLSTATE 42623).

START WITH *numeric-constant*

식별 컬럼의 첫 번째 값을 지정합니다. 이 값은 컬럼에 지정될 수 있는 모든 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않습니다(SQLSTATE 428FA). 디폴트값은 오름차순 시퀀스의 경우 MINVALUE이고, 내림차순 시퀀스의 경우 MAXVALUE입니다.

INCREMENT BY *numeric-constant*

식별 컬럼의 연속 값 간격을 지정합니다. 이 값은 이 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 큰 정수 상수의 값을 초과하지 않고(SQLSTATE 42820), 소수점 오른쪽에 0이 아닌 자릿수를 갖지 않습니다(SQLSTATE 428FA).

이 값이 음수인 경우, 값은 내림차순입니다. 이 값이 0이거나 양수인 경우, 값은 오름차순입니다. 디폴트값은 1입니다.

NO MINVALUE 또는 MINVALUE

내림차순 식별 컬럼이 순환하거나 값 생성을 중지하거나, 오름차순 식별 컬럼이 최대값에 도달한 후 순환하는 최소값을 지정합니다.

NO MINVALUE

오름차순의 경우 값은 START WITH 값으로, START WITH 값이 지정되지 않았으면 1입니다. 내림차순의 경우 값은 컬럼 데이터 유형의 최소값입니다. 이는 디폴트값입니다.

MINVALUE *numeric-constant*

최소값인 숫자 상수를 지정합니다. 이 값은 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최대값보다 작거나 같아야 합니다(SQLSTATE 42815).

NO MAXVALUE 또는 MAXVALUE

오름차순 식별 컬럼이 순환하거나 값 생성을 중지하는 최대값, 또는 내림차순 식별 컬럼이 최소값에 도달한 이후 순환하는 값을 지정합니다.

NO MAXVALUE

오름차순의 경우, 값은 컬럼 데이터 유형의 최대값입니다. 내림차순의 경우 값은 START WITH 값으로, START WITH 값이 지정되지 않았으면 -1입니다. 이는 디폴트값입니다.

MAXVALUE *numeric-constant*

최대값인 숫자 상수를 지정합니다. 이 값은 컬럼에 지정할 수 있는 양수 또는 음수 값일 수 있으며(SQLSTATE 42815), 소수점의 오른쪽에 0이 아닌 자릿수를 갖지 않지만(SQLSTATE 428FA), 값이 최소값보다 크거나 같아야 합니다(SQLSTATE 42815).

NO CYCLE 또는 CYCLE

이 식별 컬럼이 최대값 또는 최소값을 생성한 이후 계속 값을 생성할지 여부를 지정합니다.

NO CYCLE

일단 최대값이나 최소값에 도달된 후에는 식별 컬럼에 대해 값이 생성되지 않도록 지정합니다. 이는 디폴트값입니다.

CYCLE

최대값 또는 최소값에 도달된 이후 이 컬럼에 대해 값이 계속 생성되도록 지정합니다. 이 옵션을 사용하는 경우, 오름차순 식별 컬럼이 최대값에 도달하면 최소값을 생성합니다. 또는 내림차순이 최소값에 도달하면 최대값을 생성합니다. 식별 컬럼에 대한 최대값과 최소값에 따라 순환에 사용되는 범위가 결정됩니다.

CYCLE이 유효하면 DB2는 식별 컬럼에 대해 중복 값을 생성할 수 있습니다. 명시적으로 요구되지는 않지만, 고유 값을 원할 경우, 값의 고유성을 보장하기 위해 생성된 컬럼에 대해 고유한 단일 컬럼 인덱스를 정의해야 합니다. 고유 인덱스가 이러한 식별 컬럼에 존재하고 고유하지 않은 값이 생성되면, 오류가 발생합니다(SQLSTATE 23505).

NO CACHE 또는 CACHE

보다 빠른 액세스를 위해 메모리에 사전 할당된 값 일부를 저장할지 여부를 지정합니다. 식별 컬럼에 새 값이 필요한데 캐시에 어떤 값도 없는 경우, 새 캐시 블록의 끝을 기록해야 합니다. 그러나 식별 컬럼에 새 값이 필요한데 캐시에 사용되지 않는 값이 있다면, 로깅이 필요하지 않으므로 식별 값을 보다 빨리 할당할 수 있습니다. 이것은 성능 및 조정 옵션입니다.

NO CACHE

식별 컬럼의 값이 사전 할당되지 않도록 지정합니다.

이 옵션을 지정하면 식별 컬럼의 값이 캐시에 저장되지 않습니다. 이 경우, 새 식별 값에 대한 모든 요청이 로그에 동시에 입출력됩니다.

CACHE *integer-constant*

사전 할당되어 메모리에 보존되는 식별 시퀀스 값의 수를 지정합니다. 식별 컬럼에 대한 값이 생성될 때 값을 사전할당하고 캐시에 저장하면 로그에 대한 동기 입출력이 줄어듭니다.

식별 컬럼에 새 값이 필요한데 캐시에 사용되지 않는 값이 없는 경우, 값을 할당하려면 로그에 대한 입출력을 기다려야 합니다. 그러나 식별 컬럼에 새 값이 필요한데 캐시에 사용되지 않는 값이 있는 경우, 로그에 대한 입출력이 수행되지 않으므로 해당 식별 값이 보다 빨리 할당될 수 있습니다.

정상 작업 중 또는 시스템 오류로 인해 데이터베이스 비활성화가 발생한 경우, 커미트 명령문에서 사용되지 않은 캐시된 모든 시퀀스 값이 *유실됩니다*. 즉, 이 값은 사용되지 않습니다. CACHE 옵션에 대해 지정된 값은 데이터베이스 비활성화 시 유실될 수 있는 식별 컬럼에 대한 최대값입니다. (데이터베이스가 명시적으로 활성화되지 않았는데 ACTIVATE 명령 또는 API를 사용하여 마지막 응용프로그램이 데이터베이스로부터 연결 끊기되면, 내재된 비활성화가 발생합니다.)

최소값은 2입니다(SQLSTATE 42815). 디폴트값은 CACHE 20입니다.

NO ORDER 또는 ORDER

식별 값을 요청 순서대로 생성할지 여부를 지정합니다.

NO ORDER

값이 요청 순서대로 생성되지 않도록 지정합니다. 이는 디폴트 값입니다.

ORDER

값이 요청 순서대로 생성되도록 지정합니다.

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

컬럼이 테이블의 시간소인 컬럼이 되도록 지정합니다. 값은 삽입된 각 행의 컬럼 및 컬럼이 갱신된 행으로 생성됩니다. ROW CHANGE TIMESTAMP 컬럼에 대해 생성된 값은 해당 행에 대한 삽입 또는 갱신 시간에 해당되는 시간소인입니다. 다중 행이 단일 명령문으로 삽입되거나 갱신되면, ROW CHANGE TIMESTAMP 컬럼의 값은 각 행에 대해 다를 수 있습니다.

하나의 테이블은 하나의 ROW CHANGE TIMESTAMP 컬럼만 가질 수 있습니다(SQLSTATE 428C1). *data-type*이 지정된 경우 이는 TIMESTAMP 또는 TIMESTAMP(6)여야 합니다(SQLSTATE 42842).

ROW CHANGE TIMESTAMP 컬럼은 DEFAULT절을 포함할 수 없습니다(SQLSTATE 42623). NOT NULL은 ROW CHANGE TIMESTAMP 컬럼에 대해서 지정되어서는 안됩니다(SQLSTATE 42831).

GENERATED ALWAYS AS (*generation-expression***)**

컬럼의 정의가 표현식을 기반으로 하도록 지정합니다. GENERATED ALWAYS 컬럼에 대한 표현식에 사용자 정의 외부 함수가 포함되는 경우, 제공된 인수에 따라 결과가 변하도록 이 함수의 실행 가능 파일을 변경하면 데이터 불일치가 발생할 수 있습니다. 이러한 문제는 SET INTEGRITY문을 사용하여 강제로 새 값을 생성함으로써 방지할 수 있습니다. *generation-expression*에는 다음 중 어느 하나도 포함될 수 없습니다(SQLSTATE 42621).

- 서브쿼리
- XMLQUERY 또는 XMLEXISTS 표현식
- 컬럼 함수
- 비참조 연산 또는 DEREF 함수
- 비결정적인 사용자 정의 또는 내장 함수
- EXTERNAL ACTION 옵션을 사용하는 사용자 정의 함수
- NO SQL로 정의되지 않은 사용자 정의 함수
- 호스트 변수 또는 매개변수 표시문자
- 특수 레지스터의 값에 따라 다른 특수 레지스터 및 내장 함수
- 전역 변수
- 컬럼 목록에서 나중에 정의된 컬럼에 대한 참조
- 기타 생성된 컬럼에 대한 참조
- XML 유형 컬럼에 대한 참조

컬럼의 데이터 유형은 *generation-expression*의 결과 데이터 유형을 기반으로 합니다. CAST 스펙은 특정 데이터 유형을 강제로 적용하고 범위(참조 유형의 경우에만)를 제공하는 데 사용할 수 있습니다. *data-type*이 지정된 경우, 값은 적절한 지정 규칙에 따라 컬럼에 지정됩니다. 생성된 컬럼은 NOT NULL 컬럼 옵션이 사용되지 않는 한 내재적으로 널(NULL) 입력 가능 컬럼으로 고려됩니다. *generation-expression*의 결과 데이터 유형 및 생성된 컬럼의 데이터 유형은 동일하게 정의되어야 합니다(『지정 및 비교』 참조). LOB 데이터 유형, XML, 구조화된 유형 및 이러한 유형을 기본으로 하는 구별 유형의 컬럼과 표현식 생성은 제외됩니다(SQLSTATE 42962).

INLINE LENGTH *integer*

이 옵션은 구조화된 유형, XML 또는 LOB 데이터 유형을 사용하여 정의된 컬럼에 대해서만 유효합니다(SQLSTATE 42842).

XML 또는 LOB 데이터 유형 컬럼의 경우 *integer*는 기본 테이블 행에 저장하기 위한 XML 문서나 LOB 데이터 내부 표현의 최대 바이트 *크기*를 나타냅니다. 내부 표현이 더 큰 XML 문서는 기본 테이블 행에서 보조 기억장치 오브젝트로 별도로 저장됩니다. 이 태스크는 자동으로 발생합니다. XML 유형 컬럼에 대한 디폴트 인라인 길이가 없습니다. XML 문서나 LOB 데이터는 기본 테이블 행에 인라인으로 저장되면 추가적인 오버헤드가 있습니다. LOB 데이터의 경우 오버헤드는 4바이트입니다.

LOB 데이터 유형 컬럼의 경우 절이 지정되지 않으면 디폴트 인라인 길이는 최대 *크기*의 LOB 디스크립터로 설정됩니다. 명시적인 INLINE LENGTH는 최대 LOB 디스크립터 *크기*여야 합니다. 다음 표는 LOB 디스크립터 *크기*를 요약합니다.

표 18. 다양한 LOB 길이에 대한 LOB 디스크립터 크기

최대 LOB 길이(바이트)	명시적인 최소 INLINE LENGTH
1,024	68
8,192	92
65,536	116
524,000	140
4,190,000	164
134,000,000	196
536,000,000	220
1,070,000,000	252
1,470,000,000	276
2,147,483,647	312

구조화된 유형 컬럼의 경우 *integer*는 행에 값의 나머지 부분과 함께 인라인을 저장하기 위해 구조화된 유형 인스턴스의 최대 바이트 *크기*를 표시합니다. 인라인을 저장할 수 없는 구조화된 유형의 인스턴스는 LOB 값의 저장 방법과 유사하게 기본 테이블 행과는 별도로 저장됩니다. 이 태스크는 자동으로 발생합니다. 구조화된 유형 컬럼의 디폴트 INLINE LENGTH는 해당 유형의 인라인 길이입니다(CREATE TYPE문에 명시적으로 또는 디폴트값으로 지정됨). 구조화된 유형의 INLINE LENGTH가 292보다 작은 경우, 292 값이 컬럼의 INLINE LENGTH로 사용됩니다.

주: 부속 유형의 인라인 길이는 디폴트 인라인 길이에 계수되지 않습니다. 즉, 기존의 부속 유형 및 후속 부속 유형을 고려하기 위해 명시적 INLINE LENGTH를 CREATE TABLE 수행시 지정하지 않은 경우에는 부속 유형의 인스턴스가 인라인으로 적합하지 않을 수 있음을 의미합니다.

명시적 INLINE LENGTH 값은 32673을 초과할 수 없습니다. 구조화된 유형이나 XML 데이터 유형의 경우 적어도 292여야 합니다(SQLSTATE 54010).

COMPRESS SYSTEM DEFAULT
시스템 디폴트값이 최소 스페이스를 사용하여 저장되도록 지정합니다. VALUE COMPRESSION절을 지정하지 않으면 경고가 리턴되고 (SQLSTATE 01648) 시스템 디폴트값이 최소 스페이스를 사용하여 저장되지 않습니다.

이와 같은 방식으로 시스템 디폴트값이 저장되도록 하면 추가 점검이 수행되기 때문에 컬럼에 대해 삽입 및 갱신 조작을 수행하는 동안 성능이 약간 저하될 수 있습니다.

기본 데이터 유형은 DATE, TIME, TIMESTAMP, XML 또는 구조화된 데이터 유형일 수 없습니다(SQLSTATE 42842). 기본 데이터 유형이 가변 길이 문자열일 경우에는 이 절이 무시됩니다. VALUE COMPRESSION을 사용하여 테이블을 설정한 경우에는 길이가 0인 문자열 값이 자동으로 압축됩니다.

COLUMN SECURED WITH *security-label-name*
테이블과 연결된 보안 규정에 대해 존재하는 보안 레이블을 식별합니다. name은 규정하지 않아야 합니다(SQLSTATE 42601). 테이블에는 테이블과 연결된 보안 규정이 있어야 합니다(SQLSTATE 55064).

unique-constraint
고유 제한조건이나 기본 키 제한조건을 정의합니다. 테이블에 분산 키가 있는 경우, 고유 키 또는 기본 키는 분산 키의 수퍼 세트여야 합니다. 서브테이블인 테이블에 대해 고유 키 또는 기본 키 제한조건을 지정할 수 없습니다(SQLSTATE 429B3). 기본 키 또는 고유 키는 차원의 서브 세트가 될 수 없습니다(SQLSTATE 429BE). 테이블이 루트 테이블인 경우, 제한조건은 테이블 및 모든 서브테이블에 적용됩니다.

CONSTRAINT *constraint-name*
기본 키 또는 고유 제한조건의 이름을 지정합니다.

UNIQUE (*column-name***,...)**
식별된 컬럼으로 구성되는 고유 키를 정의합니다. 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별하면 안됩니다.

식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 바이트 수를 참조하십시오. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오. 컬럼의 길이 속성이 페이지 크기에 대한 인덱스 키 길이 한계를 초

과하지 않아도 LOB, XML, 이러한 유형 중 하나를 기반으로 하는 구별 유형 또는 구조화된 유형은 고유 키의 일부로 사용될 수 없습니다(SQLSTATE 54008).

고유 키의 컬럼 세트는 기본 키 또는 다른 고유 키에 있는 컬럼 세트와 동일할 수 없습니다(SQLSTATE 01543). LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 발생합니다(SQLSTATE 42891).

테이블이 서브테이블인 경우 고유 제한조건은 슈퍼 테이블로부터 상속되므로, 고유 제한조건을 지정할 수 없습니다(SQLSTATE 429B3).

카탈로그에 기록된 대로 테이블의 설명에는 고유 키와 고유 인덱스가 포함됩니다. 각 컬럼에 대해 오름차순으로 지정된 시퀀스의 컬럼에 대해 정방향 및 역방향 스캔이 가능한 고유 양방향 인덱스가 자동 작성됩니다. 인덱스의 이름은 테이블이 작성된 스키마의 기존 인덱스와 상충하지 않을 경우, *constraint-name*과 동일합니다. 인덱스 이름이 상충하는 경우, 이름은 SQL과 그 뒤에 오는 문자 시간소인(*yymmddhhmmssxxx*)으로 이뤄지며, 스키마 이름은 SYSIBM입니다.

PRIMARY KEY (*column-name***,...)**

식별된 컬럼으로 구성되는 기본 키를 정의합니다. 절은 두 번 이상 지정되어서는 안되며, 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼이 두 번 이상 식별되어서는 안됩니다.

식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 바이트 수를 참조하십시오. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오. 컬럼의 길이 속성이 페이지 크기에 대한 인덱스 키 길이 한계를 초과하지 않아도 LOB, XML, 이러한 유형 중 하나라도 기반으로 하는 구별 유형 또는 구조화된 유형은 기본 키의 일부로 사용될 수 없습니다(SQLSTATE 54008).

기본 키의 컬럼 세트는 고유 키의 컬럼 세트와 같을 수 없습니다(SQLSTATE 01543). LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 발생합니다(SQLSTATE 42891).

하나의 테이블에서 하나의 기본 키만 정의할 수 있습니다.

테이블이 서브테이블인 경우, 기본 키가 슈퍼 테이블로부터 상속하게 되므로 기본 키를 지정할 수 없습니다(SQLSTATE 429B3).

카탈로그에 기록된 대로 테이블의 설명에는 기본 키와 해당되는 1차 인덱스가 포함됩니다. 각 컬럼에 대해 오름차순으로 지정된 시퀀스의 컬럼에 대해 정방향 및 역방향 스캔이 가능한 고유 양방향 인덱스가 자동 작성됩니다. 인덱스의 이름은 테이블이 작성된 스키마의 기존 인덱스와 상충하지 않을 경우,

*constraint-name*과 동일합니다. 인덱스 이름이 상충하는 경우, 이름은 SQL과 그 뒤에 오는 문자 시간소인(*yymmddhhmmssxxx*)으로 이뤄지며, 스키마 이름은 SYSIBM입니다.

테이블에 분산 키가 있는 경우 *unique-constraint*의 컬럼은 분산 키 컬럼의 수퍼 세트여야 하며, 컬럼 순서는 중요하지 않습니다.

referential-constraint

참조 제한조건을 정의합니다.

CONSTRAINT *constraint-name*

참조 제한조건의 이름을 지정합니다.

FOREIGN KEY (*column-name***,...)**

지정된 *constraint-name*을 가진 참조 제한조건을 정의합니다.

T1은 명령문의 오브젝트 테이블을 나타냅니다. 참조 제한조건의 외부 키는 식별되는 컬럼으로 구성됩니다. 컬럼 이름 목록의 각 이름은 T1의 컬럼을 식별하고, 같은 컬럼은 두 번 이상 식별되어서는 안됩니다.

식별된 컬럼의 수는 64를 초과할 수 없으며 저장된 길이의 합은 페이지 크기에 대한 인덱스 키 길이 한계를 초과할 수 없습니다. 컬럼의 저장된 길이에 대해서는 바이트 수를 참조하십시오. 키 길이 한계에 대해서는 『SQL 한계』를 참조하십시오. LOB, XML, 이러한 유형 중 하나를 기반으로 하는 구별 유형 또는 구조화된 유형 컬럼은 외부 키의 일부로 사용될 수 없습니다(SQLSTATE 42962). 상위 키에 있는 것과 동일한 수의 외부 키 컬럼이 있어야 하며, 해당 컬럼의 데이터 유형은 호환 가능해야 합니다(SQLSTATE 42830). 데이터 유형이 호환 가능할 경우 두 컬럼 설명이 호환 가능합니다(두 컬럼이 숫자, 문자열, 그래픽, 날짜/시간 또는 동일한 구별 유형일 경우).

references-clause

참조 제한조건에 대한 상위 테이블 또는 상위 별칭 및 상위 키를 지정합니다.

REFERENCES *table-name* 또는 *nickname*

REFERENCES절에 지정된 테이블 또는 별칭은 카탈로그에 기술된 기본 테이블 또는 별칭을 식별해야 하나, 카탈로그 테이블을 식별해서는 안됩니다.

참조 제한조건의 외부 키, 상위 키, 상위 테이블 또는 상위 별칭이 이전에 지정된 참조 제한조건의 외부 키, 상위 키, 상위 테이블 또는 상위 별칭과 동일할 경우, 참조 제한조건이 중복됩니다. 중복 참조 제한조건은 무시되며 경고가 리턴됩니다(SQLSTATE 01543).

아래 설명에서 T2는 식별된 상위 테이블을 나타내고, T1은 작성되거나 변경될 테이블을 나타냅니다. T1과 T2는 같은 테이블일 수 있습니다.

지정되는 외부 키는 T2의 상위 키와 컬럼 수가 같아야 하며, 외부 키의 *n* 번째 컬럼의 설명은 상위 키의 *n*번째 컬럼의 설명에 해당합니다. 날짜 시간 컬럼은 이 규칙의 목적상 문자열 컬럼과 비교 가능한 것으로 간주되지 않습니다.

(*column-name***,...)**

참조 제한조건의 상위 키는 식별된 컬럼으로 구성됩니다. 각 *column-name*은 T2의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다. 같은 컬럼은 한 번 이상 식별될 수 없습니다.

컬럼 이름의 목록은 T2에 존재하는 기본 키나 고유 제한조건의 컬럼 세트(순서는 관계없음)와 일치해야 합니다(SQLSTATE 42890). 컬럼 이름 목록을 지정하지 않을 경우, T2에 기본 키가 있어야 합니다 (SQLSTATE 42888). 컬럼 이름 목록을 생략하면 원래 지정된 순서로 해당 기본 키의 컬럼 스펙이 내재됩니다.

FOREIGN KEY절에 의해 지정된 참조 제한조건은 T2가 상위이고 T1이 종속인 관계를 정의합니다.

rule-clause

종속 테이블에서 취할 조치를 지정합니다.

ON DELETE

상위 테이블의 행이 삭제될 때 종속 테이블에서 취할 조치를 지정합니다. 네 가지의 가능한 조치는 다음과 같습니다.

- NO ACTION(디폴트값)
- RESTRICT
- CASCADE
- SET NULL

삭제 규칙은 T2의 행이 DELETE의 오브젝트이거나 전파된 삭제 조작이고 해당 행이 T1에 종속될 경우에 적용됩니다. *p*는 T2의 그러한 행을 나타냅니다.

- RESTRICT나 NO ACTION을 지정한 경우, 오류가 발생하여 어떤 행도 삭제되지 않습니다.
- CASCADE를 지정할 경우, 삭제 작업은 T1에 있는 *p*의 종속사항에 전파됩니다.
- SET NULL을 지정할 경우, T1에 있는 *p*의 각 종속사항에 대한 외부 키의 널(NULL) 값 입력 가능한 각 컬럼은 널(NULL)로 설정됩니다.

외부 키의 일부 컬럼이 널(NULL) 값을 허용하지 않으면 SET NULL을 지정해서는 안됩니다. 절을 생략하면 ON DELETE NO ACTION이 내재적으로 지정됩니다.

T1이 여러 경로를 통해 T2에 삭제 연결되는 경우, 외부 키 정의를 겹쳐 2개의 SET NULL 규칙을 정의하는 것은 허용되지 않습니다 (예: T1 (i1, i2, i3)). 외부 키(i1, i2)가 있는 Rule1 및 외부 키(i2, i3)가 있는 Rule2는 허용되지 않습니다.

규칙의 시작 순서는 다음과 같습니다.

1. RESTRICT
2. SET NULL OR CASCADE
3. NO ACTION

T1의 임의 행이 다른 규칙에 의해 영향을 받는 경우, 오류가 발생하며 어떠한 행도 삭제되지 않습니다.

둘 이상의 테이블이 순환하여 테이블 자체가 연속 삭제가 되는 경우 및 삭제 규칙에 RESTRICT 또는 SET NULL이 있는 경우에는 참조 제한조건을 정의할 수 없습니다(SQLSTATE 42915).

다중 경로에 의해 테이블이 자기 자신 또는 다른 테이블이 연속 삭제가 되는 경우 참조 제한조건을 정의할 수는 있으나 다음의 경우에는 정의할 수 없습니다(SQLSTATE 42915).

- 테이블은 CASCADE 관계(자체 참조 또는 다른 테이블 참조)에 있는 종속 테이블이 될 수 없고 삭제 규칙이 RESTRICT 또는 SET NULL인 자체 참조 관계를 가질 수 없습니다.
- 키에 있는 최소한 하나의 컬럼이 다른 키의 컬럼과 동일한 경우 키는 다른 키와 겹쳐집니다. 겹치는 외부 키를 사용하여 테이블과 다른 테이블이 연속 삭제의 다중 관계에 있는 경우 이러한 관계에서는 동일한 삭제 규칙이 있어야 하며 삭제 규칙은 SET NULL이 될 수 없습니다.
- 테이블이 다중 관계를 통해 다른 테이블과 연속 삭제되고 최소한 하나의 관계가 SET NULL 삭제 규칙으로 지정된 경우, 다중 관계의 외부 키 정의에는 분산 키 또는 다차원 클러스터링(MDC) 키 컬럼이 없어야 합니다.
- 두 테이블이 CASCADE 관계를 통해 동일한 테이블로 연속 삭제되면, 각 삭제 연결된 경로에서 마지막 관계의 삭제 규칙이 RESTRICT 또는 SET NULL인 두 테이블이 서로 연속 삭제되어서는 안됩니다.

T1의 행이 다른 규칙의 영향을 받는 경우, 결과는 이 규칙에서 지정된 모든 조치의 영향을 받습니다. T1에 대한 AFTER 트리거 및 점검 제한조건에서도 모든 조치가 발효됩니다. 이에 대한 예로, 상위 테이블에

연결된 하나의 연속 삭제 경로에서 널(NULL)로 설정되고 동일한 상위 테이블에 연결된 두 번째 연속 삭제 경로를 통해 삭제되도록 한 행을 들 수 있습니다. 그 결과로 행이 삭제됩니다. 하위 테이블의 AFTER DELETE 트리거는 활성화되지만 AFTER UPDATE 트리거는 활성화되지 않습니다.

위의 규칙을 참조 제한조건에 적용하는 경우 상위 테이블이나 종속 테이블이 유형이 지정된 테이블 계층의 구성원이라면, 각 계층의 테이블에 적용되는 모든 참조 제한조건이 고려됩니다.

ON UPDATE
상위 테이블의 행이 갱신될 때 종속 테이블에서 취할 조치를 지정합니다. 이 절 은 생략 가능합니다. ON UPDATE NO ACTION이 디폴트값이며, ON UPDATE RESTRICT는 유일한 대체값입니다.

NO ACTION과 RESTRICT의 차이점은『참고』절에 설명되어 있습니다.

check-constraint
점검 제한조건을 정의합니다. *check-constraint*는 참으로 평가되어야 하는 *search-condition*이거나 컬럼 간에 정의된 함수적 종속성입니다.

CONSTRAINT *constraint-name*
점검 제한조건의 이름을 지정합니다.

CHECK (*check-condition*)
점검 제한조건을 정의합니다. *search-condition*은 테이블의 모든 행에 대해 참이거나 알 수 없어야 합니다.

search-condition
*search-condition*에는 다음과 같은 제한사항이 있습니다.

- 컬럼 참조는 작성할 테이블의 컬럼이어야 합니다.
- *search-condition*에는 TYPE 술어가 포함될 수 없습니다.
- *search-condition*은 다음을 포함할 수 없습니다(SQLSTATE 42621).
  - 서브쿼리
  - XMLQUERY 또는 XMLEXISTS 표현식
  - 범위 지정된 참조 인수가 오브젝트 ID(OID) 컬럼이 아닌 비참조 조작 또는 DEREF 함수
  - SCOPE절이 있는 CAST 권장 스펙
  - 컬럼 함수
  - 결정적이지 않은 함수
  - 외부 조치를 갖도록 정의된 함수

- CONTAINS SQL 또는 READS SQL DATA로 정의된 사용자 정의 함수(UDF)
- 호스트 변수
- 매개변수 표시문자
- *sequence-references*
- OLAP 스펙
- 특수 레지스터의 값에 따라 다른 특수 레지스터 및 내장 함수
- 전역 변수
- 식별 컬럼이 아닌 다른 생성된 컬럼에 대한 참조
- XML 유형 컬럼 참조(VALIDATED 술어의 경우 제외)
- 오류가 허용되는 *nested-table-expression*

functional-dependency

컬럼 간의 함수적 종속성을 정의합니다.

column-name **DETERMINED BY** *column-name* 또는 **(***column-name***,...) DETERMINED BY (***column-name***,...)**

컬럼의 수퍼 세트에는 DETERMINED BY절 바로 앞에 오는 식별된 컬럼이 포함됩니다. 컬럼의 하위 세트에는 DETERMINED BY절 바로 뒤에 오는 식별된 컬럼이 포함됩니다. *search-condition*의 모든 제한사항은 수퍼 세트 및 하위 세트 컬럼에 적용되며 컬럼 세트에는 단순 컬럼 참조만 허용됩니다(SQLSTATE 42621). 동일한 컬럼은 함수적 종속성에서 여러 번 식별될 수 없습니다(SQLSTATE 42709). 컬럼의 데이터 유형은 LOB 데이터 유형, LOB 데이터 유형에 기반한 구별 유형, XML 데이터 유형 또는 구조화된 유형일 수 없습니다(SQLSTATE 42962). ROW CHANGE TIMESTAMP 컬럼은 기본키의 일부로 사용될 수 없습니다(SQLSTATE 429BV). 컬럼의 하위 세트에 있는 컬럼은 널(NULL) 입력 가능 컬럼일 수 없습니다(SQLSTATE 42621).

*column-definition*의 일부분으로 점검 제한조건이 지정된 경우, 컬럼 참조는 같은 컬럼에 대해서만 이루어질 수 있습니다. 테이블 정의의 일부분으로 지정된 점검 제한조건은 이전에 CREATE TABLE문에 정의한 컬럼을 식별하는 컬럼 참조를 가질 수 있습니다. 비일관성, 중복 조건 또는 동등한 조건에 대해 점검 제한조건이 점검되지 않습니다. 그러므로 모순되거나 불필요한 점검 제한조건이 정의되면 실행시 오류가 발생할 수 있습니다.

search-condition 『IS NOT NULL』을 지정할 수는 있으나, 컬럼의 NOT NULL 속성을 사용하여 널(null) 가능성을 직접 적용할 것을 권장합니다. 예를 들어, 급여가 널(NULL)로 설정된 경우, CHECK 제한조건을 만족시켜야 하는데 급여를 알 수 없으므로, `CHECK (salary + bonus > 30000)`가 허용됩니다. 그

러나 급여가 널(NULL)로 설정된 경우, CHECK (salary IS NOT NULL)는 거짓으로 간주되어 제한조건에 위반됩니다.

*search-condition*을 사용하는 점검 제한조건은 테이블의 행이 삽입되거나 갱신될 [008b]시행됩니다. 테이블에 지정된 점검 제한조건은 해당 테이블의 모든 서브테이블에 자동으로 적용됩니다.

삽입, 갱신, 삭제 또는 무결성 설정과 같은 일반 조작 중에 데이터베이스 관리 프로그램은 함수적 종속성을 시행하지 않습니다. 함수적 종속성은 쿼리를 최적화하기 위해 쿼리 재작성 중에 사용될 수 있습니다. 함수적 종속성의 무결성을 유지하지 않으면 잘못된 결과가 리턴될 수 있습니다.

constraint-attributes

참조 무결성이나 점검 제한조건과 연관된 속성을 정의합니다.

ENFORCED 또는 NOT ENFORCED

제한조건이 삽입, 갱신 또는 삭제 등 정상 조작 중에 데이터베이스 관리 프로그램에 의해 시행되는지 여부를 지정합니다. 디폴트값은 ENFORCED입니다.

ENFORCED

제한조건이 데이터베이스 관리 프로그램에 의해 시행됩니다. ENFORCED는 함수적 종속성에 대해 지정할 수 없습니다(SQLSTATE 42621). 참조 제한조건이 별칭을 참조할 경우 ENFORCED를 지정할 수 없습니다 (SQLSTATE 428G7).

NOT ENFORCED

제한조건이 데이터베이스 관리 프로그램에 의해 시행되지 않습니다. 이 옵션은 각 테이블 데이터가 개별적으로 제한조건을 확인해야 할 경우에만 지정해야 합니다.

ENABLE QUERY OPTIMIZATION 또는 DISABLE QUERY OPTIMIZATION

적절한 환경에서 쿼리 최적화를 위해 제한조건 또는 함수적 종속성을 사용할 수 있는지 여부를 지정합니다. 디폴트값은 ENABLE QUERY OPTIMIZATION입니다.

ENABLE QUERY OPTIMIZATION

제한조건을 참으로 가정하며 쿼리 최적화에 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

제한조건을 쿼리 최적화에 사용할 수 없습니다.

OF *type-name1*

테이블의 컬럼이 *type-name1*으로 식별되는 구조화된 유형 속성에 기초하도록 지정합니다. 스키마 이름없이 *type-name1*을 지정하면, 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 사전 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의됨)에서 스키마를 검색하여 분석됩니다. 유형 이름은 기

존 사용자 정의 유형의 이름이어야 하고(SQLSTATE 42704) 최소한 하나의 속성을 갖는(SQLSTATE 42997) 인스턴스화되는 구조화된 유형이어야 합니다(SQLSTATE 428DP).

UNDER가 지정되지 않은 경우 오브젝트 ID 컬럼을 지정해야 합니다(*OID-column-definition* 참조). 이 오브젝트 ID 컬럼이 테이블의 첫 번째 컬럼입니다. 오브젝트 ID 컬럼 뒤에는 *type-name1* 속성에 기초한 컬럼이 옵니다.

HIERARCHY *hierarchy-name*

테이블 계층과 연관된 계층 테이블의 이름을 지정합니다. 계층의 루트 테이블과 동시에 작성됩니다. 유형이 지정된 테이블 계층에 있는 모든 서브테이블에 대한 데이터는 계층 테이블에 저장됩니다. 계층 테이블은 SQL문에서 직접 참조될 수 없습니다. *hierarchy-name*은 *table-name*입니다. 내재적 또는 명시적인 스키마 이름을 포함한 *hierarchy-name*은 카탈로그에 기술된 테이블, 별칭, 뷰 또는 별명을 식별해서는 안됩니다. 스키마 이름을 지정할 경우, 작성되는 테이블의 스키마 이름과 동일해야 합니다. 루트 테이블을 정의할 때 이 절이 생략된 경우 시스템에 의해 이름이 생성됩니다. 이 이름은 ID가 기존 테이블, 뷰 및 별칭의 ID 내에서 고유하도록 고유한 접미부가 뒤따르는 작성중인 테이블의 이름으로 구성됩니다.

UNDER *supertable-name*

테이블이 *supertable-name*의 서브테이블임을 나타냅니다. 슈퍼 테이블은 기존 테이블이어야 하고(SQLSTATE 42704), 이 테이블은 *type-name1*의 직접 슈퍼 테이블인 구조화된 유형을 사용하여 정의되어야 합니다(SQLSTATE 428DB). *table-name*과 *supertable-name*은 동일해야 합니다(SQLSTATE 428DQ). *supertable-name*으로 식별되는 테이블에는 *type-name1*(SQLSTATE 42742)을 사용하여 이미 정의된 서브테이블이 없어야 합니다.

테이블 컬럼에는 REF(*type-name1*)로 수정될 유형을 가진 슈퍼 테이블 오브젝트 ID 컬럼이 포함되며, 그 뒤에는 *type-name1* 속성에 기초한 컬럼들이 옵니다. (이 유형에는 슈퍼 유형 속성이 포함된다는 점에 유의하십시오.) 속성 이름이 OID 컬럼 이름과 동일해서는 안됩니다(SQLSTATE 42711).

테이블 스페이스, 데이터 캡처, 초기에 기록되지 않은 테이블 옵션 및 분산 키 옵션과 같은 다른 테이블 옵션은 지정될 수 없습니다. 이러한 옵션들은 슈퍼 테이블로부터 상속됩니다(SQLSTATE 42613).

INHERIT SELECT PRIVILEGES

슈퍼 테이블에 대한 SELECT 특권을 보유하고 있는 사용자나 그룹에게는 새로 작성된 서브테이블에 대해서도 그에 상응하는 특권이 부여됩니다. 서브테이블 정의자가 이 특권을 부여한 사용자로 간주됩니다.

typed-element-list

유형이 지정된 테이블의 추가 요소를 정의합니다. 여기에는 컬럼에 대한 추가 옵션, 오브젝트 ID 컬럼 추가(루트 테이블만), 테이블에 대한 제한조건 등이 포함됩니다.

OID-column-definition

유형이 지정된 테이블에 대한 오브젝트 ID 컬럼을 정의합니다.

REF IS *OID-column-name* **USER GENERATED**

오브젝트 ID(OID) 컬럼이 첫 번째 컬럼으로 테이블에 정의되도록 지정합니다. OID는 테이블 계층의 루트 테이블에 필수적입니다(SQLSTATE 428DX). 테이블은 서브테이블이 아닌 유형이 지정된 테이블이(OF절이 있어야 함)어야 합니다(SQLSTATE 42613). 컬럼 이름은 *OID-column-name*으로 정의되며, 구조화된 유형 *type-name1*의 속성 이름과 같을 수 없습니다(SQLSTATE 42711). 컬럼은 REF(*type-name1*) 유형, NOT NULL을 사용하여 정의되며, 시스템에 필요한 고유 인덱스가 (디폴트 인덱스 이름으로) 생성됩니다. 이 컬럼을 *오브젝트 ID 컬럼* 또는 *OID 컬럼*이라고 합니다. 키워드 USER GENERATED는 행을 삽입할 때 사용자가 OID 컬럼에 대한 초기 값을 제공해야 한다는 것을 나타냅니다. 일단 행이 삽입되면 OID 컬럼은 갱신할 수 없습니다(SQLSTATE 42808).

with-options

유형이 지정된 테이블 컬럼에 적용되는 추가 옵션을 정의합니다.

column-name

추가 옵션이 지정될 컬럼의 이름을 지정합니다. *column-name*은 슈퍼 테이블의 컬럼이 아닌 테이블의 컬럼 이름과 일치해야 합니다(SQLSTATE 428DJ). 컬럼 이름은 명령문에서 하나의 WITH OPTIONS절에만 나타날 수 있습니다(SQLSTATE 42613).

(CREATE TYPE에서) 유형 정의의 일부로 이미 옵션이 지정되어 있는 경우, 여기에 지정된 옵션은 CREATE TYPE에 있는 옵션을 대체합니다.

WITH OPTIONS *column-options*

지정된 컬럼에 대한 옵션을 정의합니다. 앞에서 설명한 *column-options*를 참조하십시오. 테이블이 서브테이블인 경우, 기본 키 또는 고유 제한조건을 지정할 수 없습니다(SQLSTATE 429B3).

LIKE *table-name1* **또는** *view-name* **또는** *nickname*

테이블 컬럼에 식별된 테이블(*table-name1*), 뷰(*view-name*) 또는 별칭(*nickname*)의 컬럼과 정확히 같은 이름과 설명이 있음을 지정합니다. LIKE 다음에 지정된 이름은 카탈로그 또는 선언된 임시 테이블에 있는 테이블, 뷰 또는 별칭을 식별해야 합니다. 유형이 지정된 테이블 또는 유형이 지정된 뷰는 지정할 수 없습니다(SQLSTATE 428EC).

LIKE의 사용은 *n* 컬럼의 내재된 정의입니다. 여기서, *n*은 식별된 테이블(내재적으로 숨겨진 컬럼 포함), 뷰 또는 별칭의 컬럼 수입니다. 기존 테이블에서 내재적으로 숨겨진 컬럼에 해당되는 새 테이블의 컬럼은 내재적인 숨김으로 정의됩니다. 내재적인 정의는 LIKE 이후에 식별되는 내용에 따라 다릅니다.

- 식별된 테이블인 경우 내재된 정의에는 *table-name1*의 각 컬럼에 대한 컬럼 이름, 데이터 유형, 숨겨진 속성 및 널(null) 가능성 특성이 포함됩니다. EXCLUDING COLUMN DEFAULTS가 지정되지 않을 경우에는 컬럼 디폴트값도 포함됩니다.
- 식별된 뷰인 경우 내재된 정의에는 *view-name*에 정의된 fullselect의 각 결과 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다.
- 식별된 별칭인 경우 내재된 정의에는 *nickname*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다.
- 보호된 테이블이 LIKE절에서 식별되는 경우 새 테이블은 식별된 테이블과 동일한 보안 규정 및 보호된 컬럼을 상속받습니다.

복사 속성 절을 기반으로 컬럼 디폴트값 및 식별 컬럼 속성을 포함하거나 제외시킬 수 있습니다. 내재된 정의에는 식별된 테이블, 뷰 또는 별칭의 다른 어느 속성도 포함되지 않습니다. 그러므로 새 테이블이 고유한 제한조건, 외부 키 제한조건, 트리거 또는 인덱스를 가지지 않습니다. 테이블은 IN절에 의해 내재적 또는 명시적으로 지정된 테이블 스페이스에 작성되며, 선택적 절이 지정되는 경우에만 테이블이 다른 선택적 절을 가집니다.

테이블이 LIKE절에서 식별되고 테이블에 ROW CHANGE TIMESTAMP 컬럼이 포함되어 있으면, 새 테이블의 해당 컬럼이 ROW CHANGE TIMESTAMP 컬럼의 데이터 유형만을 상속합니다. 새 컬럼이 생성된 컬럼이 되지 않습니다.

copy-options

이러한 옵션은 소스 결과 테이블 정의(테이블, 뷰 또는 fullselect)의 추가 속성을 복사할지 여부를 지정합니다.

INCLUDING COLUMN DEFAULTS

소스 결과 테이블 정의를 갱신할 수 있는 각 컬럼의 컬럼 디폴트값이 복사됩니다. 갱신 가능하지 않은 컬럼은 작성된 테이블의 해당 컬럼에 정의된 디폴트값을 가지지 않습니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별하면, INCLUDING COLUMN DEFAULTS가 디폴트값입니다.

EXCLUDING COLUMN DEFAULTS

컬럼 디폴트값이 소스 결과 테이블 정의로부터 복사되지 않습니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별하는 경우를 제외하고는 이 절이 디폴트값입니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES

가능한 경우 식별 컬럼 속성이 소스 결과 테이블 정의로부터 복사됩니다. 테이블, 뷰 또는 fullselect의 해당 컬럼 요소가 테이블 컬럼의 이름이거나 식별 등

록 정보를 가지는 기본 테이블 컬럼의 이름에 직접 또는 간접적으로 맵핑되는 뷰 컬럼의 이름인 경우 식별 컬럼 속성을 복사할 수 있습니다. 다른 모든 경우에는 새 테이블의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.

- fullselect의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함됩니다(즉, 두 번 이상 같은 컬럼 선택).
- fullselect의 선택 목록에 여러 식별 컬럼이 포함됩니다(즉, 하나의 조인 포함).
- 식별 컬럼이 선택 목록의 표현식에 포함됩니다.
- fullselect에 하나의 설정 연산(union, except 또는 intersect)이 포함됩니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES
식별 컬럼 속성이 소스 결과 테이블 정의로부터 복사되지 않습니다.

as-result-table

column-name
테이블에 있는 컬럼의 이름을 지정합니다. 컬럼 이름 목록을 지정한 경우, *fullselect*의 결과 테이블에 있는 컬럼 수와 동일해야 합니다. 각 *column-name*은 고유해야 하고 규정화되어서는 안됩니다. 컬럼 이름 목록을 지정하지 않은 경우, 테이블의 컬럼은 *fullselect*의 결과 컬럼에 대한 이름을 상속합니다.

fullselect의 결과 테이블에 있는 이름이 지정되지 않은 컬럼에 중복된 컬럼 이름이 있는 경우, 컬럼 이름 목록을 반드시 지정해야 합니다(SQLSTATE 42908). 이름 지정되지 않은 컬럼은 컨테이너, 함수, 표현식 또는 세트 조작으로부터 유래된 컬럼으로서, 선택 목록의 AS절을 사용하여 이름 지정되지 않았습니다.

AS
테이블의 정의에 사용되는 쿼리를 소개합니다.

fullselect
테이블을 기반으로 하는 쿼리를 정의합니다. 결과 컬럼 정의는 동일한 쿼리로 정의된 뷰에 대한 정의와 동일합니다. *fullselect*에서 참조하는 기본 테이블에서 내재적으로 숨겨진 컬럼에 해당되는 새 테이블의 컬럼은 새 테이블에 숨겨진 것으로 간주되지 않습니다.

모든 선택 목록 요소에는 이름(표현식에 AS절 사용)이 있어야 합니다. *as-result-table*은 테이블의 속성을 정의합니다.

*fullselect*는 *data-change-table-reference*절을 포함할 수 없습니다(SQLSTATE 428FL).

유형이 지정된 테이블이나 유형이 지정된 뷰를 참조하지 않는 유효한 *fullselect*를 지정할 수 있습니다.

WITH NO DATA

쿼리가 테이블을 정의하는 데에만 사용됩니다. 테이블은 쿼리 결과를 사용하여 채워지지 않습니다.

테이블의 컬럼은 *fullselect*의 결과인 컬럼의 정의를 기본으로 정의됩니다. *fullselect*가 FROM절에서 하나의 테이블을 참조하는 경우, 해당 테이블의 컬럼인 선택 목록 항목은 컬럼 이름, 데이터 유형 및 참조된 테이블의 널(null) 가능성 특성을 사용하여 정의됩니다.

materialized-query-definition

column-name

테이블에 있는 컬럼의 이름을 지정합니다. 컬럼 이름의 목록이 지정되면, 이 목록에 포함된 이름 수는 fullselect의 결과 테이블에 있는 컬럼 수와 동일해야 합니다. 각 *column-name*은 고유해야 하고 규정화되어서는 안됩니다. 컬럼 이름 목록을 지정하지 않은 경우, 테이블의 컬럼은 fullselect의 결과 컬럼에 대한 이름을 상속합니다.

*fullselect*의 결과 테이블에 있는 이름이 지정되지 않은 컬럼에 중복된 컬럼 이름이 있는 경우, 컬럼 이름 목록을 지정해야 합니다(SQLSTATE 42908). 이름 지정되지 않은 컬럼은 컨테이너, 함수, 표현식 또는 세트 조작으로부터 유래된 컬럼으로서, 선택 목록의 AS절을 사용하여 이름 지정되지 않았습니다.

AS

테이블 정의에 사용되는 쿼리 및 테이블에 포함되는 데이터를 판별하는 쿼리를 사용합니다.

fullselect

테이블을 기반으로 하는 쿼리를 정의합니다. 결과 컬럼 정의는 동일한 쿼리로 정의된 뷰에 대한 정의와 동일합니다. fullselect에서 참조하는 기본 테이블에서 내재적으로 숨겨진 컬럼에 해당되는 새 테이블의 컬럼은 새 테이블에 숨겨진 것으로 간주되지 않습니다.

모든 선택 목록 요소에는 이름(표현식에 AS절 사용)이 있어야 합니다. *materialized-query-definition*은 구체화된 쿼리 테이블의 속성을 정의합니다. 선택된 옵션은 다음과 같이 fullselect의 내용도 정의합니다.

fullselect는 *data-change-table-reference*절을 포함할 수 없습니다(SQLSTATE 428FL).

REFRESH DEFERRED 또는 REFRESH IMMEDIATE가 지정되어 있을 경우, fullselect는 다음을 포함할 수 없습니다(SQLSTATE 428EC).

- FROM절에서 구체화된 쿼리 테이블, 작성된 임시 테이블, 선언된 임시 테이블 또는 유형이 지정된 테이블에 대한 참조
- 뷰의 fullselect가 구체화된 쿼리 테이블의 fullselect에 대해 나열된 제한사항을 위반하는 뷰에 대한 참조

- 참조 유형인 표현식(또한 이 유형에 기반한 구별 유형)
- 다음 속성을 갖는 함수
  - EXTERNAL ACTION
  - LANGUAGE SQL
  - CONTAINS SQL
  - READS SQL DATA
  - MODIFIES SQL DATA
- 물리적 특성에 의존하는 함수(예: DBPARTITIONNUM, HASHEDVALUE, RID_BIT, RID)
- ROW CHANGE 표현식 또는 행의 ROW CHANGE TIMESTAMP 컬럼 참조
- 시스템 오브젝트에 대한 테이블 또는 뷰 참조(Explain 테이블도 지정하지 않아야 함)
- 구조화된 유형, LOB 유형(또는 LOB 유형에 기반한 구별 유형) 또는 XML 유형인 표현식
- 보호된 테이블 또는 보호된 별칭 참조

DISTRIBUTE BY REPLICATION이 지정된 경우 다음 제한사항이 적용됩니다.

- GROUP BY절이 허용되지 않습니다.
- 구체화된 쿼리 테이블은 단일 테이블만 참조해야 합니다. 즉, 조인을 포함할 수 없습니다.

REFRESH IMMEDIATE가 지정될 때 다음과 같은 제한사항이 적용됩니다.

- 쿼리는 subselect이어야 합니다. 단, GROUP BY의 입력 테이블 표현식에서 UNION ALL을 지원할 경우는 예외입니다.
- 쿼리는 재귀될 수 없습니다.
- 쿼리는 다음을 포함할 수 없습니다.
  - 별칭에 대한 참조
  - 결정적이지 않은 함수
  - 스칼라 fullselect
  - fullselect를 포함한 술어
  - 특수 레지스터의 값에 따라 다른 특수 레지스터 및 내장 함수
  - 전역 변수
  - SELECT DISTINCT
  - 오류가 허용되는 *nested-table-expression*

- FROM절이 두 개 이상의 테이블이나 뷰를 참조할 경우, 명시적 INNER JOIN 구문을 사용하지 않고 내부 조인을 정의할 수 있습니다.
- GROUP BY절을 지정할 경우, 다음과 같은 고려사항이 적용됩니다.
  - 지원되는 컬럼 함수는 SUM, COUNT, COUNT_BIG 및 GROUPING (DISTINCT없이)입니다. 선택 목록은 COUNT(*) 또는 COUNT_BIG(*) 컬럼을 포함해야 합니다. 구체화된 쿼리 테이블 선택 목록이 SUM(X)(여기서, X는 널(NULL) 입력 가능 인수)를 포함하는 경우, 구체화된 쿼리 테이블의 선택 목록에 COUNT(X)도 있어야 합니다. 이들 컬럼 함수는 표현식의 일부가 될 수 없습니다.
  - HAVING절이 허용되지 않습니다.
  - 다중 파티션된 데이터베이스 파티션 그룹의 경우, 분산 키는 GROUP BY 항목의 서브세트여야 합니다.
- 구체화된 쿼리 테이블에는 중복 행이 없어야 하며 GROUP BY절의 지정 여부에 따라 이 고유성 요구사항에 해당되는 다음과 같은 제한사항이 적용됩니다.
  - GROUP BY절이 지정된 경우, 다음 고유성 관련 제한사항이 적용됩니다.
    - 모든 GROUP BY 항목이 선택 목록에 포함되어야 합니다.
    - GROUP BY에 GROUPING SETS, CUBE 또는 ROLLUP이 포함되어 있을 경우, 선택 목록에 있는 GROUP BY 항목 및 관련 GROUPING 컬럼 함수는 결과 세트의 고유 키를 형성합니다. 이와 같이 하려면, 다음 제한사항을 충족시켜야 합니다.
      - 그룹화 세트는 반복될 수 없습니다. 예를 들어, `ROLLUP(X,Y),X`는 `GROUPING SETS((X,Y),(X),(X))`와 동등하지 않으므로, 허용되지 않습니다.
      - X가 GROUPING SETS, CUBE 또는 ROLLUP 내에 나타나는 널(NULL) 입력 가능 GROUP BY 항목인 경우, GROUPING(X)가 선택 목록에 나타나야 합니다.
  - GROUP BY절을 지정하지 않은 경우, 다음과 같은 고유성 관련 제한사항이 적용됩니다.
    - 구체화된 쿼리 테이블의 고유성 요구사항은 각각의 하위 테이블에 정의된 고유 키 제한조건 중 하나에서 구체화된 뷰의 고유 키를 추출하여 충족시킵니다. 그러므로 하위 테이블에는 최소한 하나의 고유 키 제한조건이 정의되어 있어야 하며 이 키의 컬럼은 구체화된 쿼리 테이블 정의의 선택 목록에 나타나야 합니다.
- MAINTAINED BY FEDERATED_TOOL이 지정된 경우, FROM절에서는 별칭에 대한 참조만 사용할 수 있습니다.

REFRESH DEFERRED가 지정될 때 다음 사항이 적용됩니다.

- 연결된 스테이징 테이블을 제공하려는 목적으로 구체화된 쿼리 테이블을 뒤의 명령문에서 작성한 경우, 구체화된 쿼리 테이블의 fullselect는 REFRESH IMMEDIATE 옵션이 지정된 구체화된 쿼리 테이블을 작성하는 데 사용되는 fullselect와 동일한 제한사항 및 규칙을 따라야 합니다.
- 쿼리가 순환하는 경우 구체화된 쿼리 테이블은 쿼리 처리를 최적화하는 데 사용되지 않습니다.

fullselect에 GROUP BY절이 포함된 구체화된 쿼리 테이블에는 fullselect에서 참조한 테이블의 데이터를 요약합니다. 이러한 구체화된 쿼리 테이블을 *요약 테이블*이라고도 합니다. 요약 테이블은 구체화된 쿼리 테이블의 특수한 유형입니다.

refreshable-table-options

구체화된 쿼리 테이블 속성의 새로 고침 옵션을 정의합니다.

DATA INITIALLY DEFERRED

데이터가 테이블에 CREATE TABLE문의 일부로 삽입되지 않습니다. *table-name*을 지정하는 REFRESH TABLE문은 데이터를 테이블에 삽입하는 데 사용됩니다.

REFRESH

테이블의 데이터가 유지보수되는 방식을 나타냅니다.

DEFERRED

테이블의 데이터가 REFRESH TABLE문을 사용하여 언제라도 새로 고쳐질 수 있습니다. 테이블의 데이터는 REFRESH TABLE문이 처리될 때의 스냅샷인 쿼리 결과에만 영향을 미칩니다. 이 속성이 정의되어 있는 시스템 유지보수 구체화된 쿼리 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다(SQLSTATE 42807). 이 속성이 정의되어 있는 사용자 유지보수 구체화된 쿼리 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 있습니다(SQLSTATE 42807).

IMMEDIATE

DELETE, INSERT 또는 UPDATE의 일부로서 하위 테이블에 대해 수행된 변경사항은 구체화된 쿼리 테이블에 연쇄됩니다. 이런 경우, 테이블의 내용은 지정된 *subselect*가 처리될 경우와 항상 동일합니다. 이 속성을 사용하여 정의되어 있는 구체화된 쿼리 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다(SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

적절한 환경에서 쿼리 최적화에 구체화된 쿼리 테이블을 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

구체화된 쿼리 테이블을 쿼리 최적화에 사용할 수 없습니다. 해당 테이블은 여전히 직접 쿼리됩니다.

MAINTAINED BY

구체화된 쿼리 테이블의 데이터가 시스템, 사용자 또는 복제 도구에 의해 유지보수될지 여부를 지정합니다. 디폴트값은 SYSTEM입니다.

SYSTEM

구체화된 쿼리 테이블의 데이터를 시스템에서 유지보수하도록 지정합니다.

USER

구체화된 쿼리 테이블의 데이터를 사용자가 유지보수하도록 지정합니다. 사용자는 사용자가 유지보수하는 구체화된 쿼리 테이블에 대해 갱신, 삭제, 삽입 조작을 수행할 수 있습니다. 시스템에서 유지보수하는 구체화된 쿼리 테이블에 사용되는 REFRESH TABLE문은 사용자가 유지보수하는 구체화된 쿼리 테이블에 대하여 호출할 수 없습니다. REFRESH DEFERRED 구체화된 쿼리 테이블만 MAINTAINED BY USER로 정의할 수 있습니다.

FEDERATED_TOOL

구체화된 쿼리 테이블의 데이터를 복제 도구에서 유지보수하도록 지정합니다. 시스템에서 유지보수하는 구체화된 쿼리 테이블에 사용되는 REFRESH TABLE문은 페더레이티드 도구가 유지보수하는 구체화된 쿼리 테이블에 대하여 호출할 수 없습니다. REFRESH DEFERRED 구체화된 쿼리 테이블만 MAINTAINED BY FEDERATED_TOOL로 정의할 수 있습니다.

staging-table-definition

스테이징 테이블이 지원하는 쿼리를 연관된 구체화된 쿼리 테이블을 통하여 간접적으로 정의합니다. 구체화된 쿼리 테이블의 하위 테이블은 또한 이 테이블과 관련된 스테이징 테이블의 하위 테이블이기도 합니다. 스테이징 테이블은 구체화된 쿼리 테이블을 하위 테이블의 내용과 동기화하기 위해 구체화된 쿼리 테이블에 적용해야 할 변경사항을 수집합니다.

staging-column-name

스테이징 테이블 컬럼의 이름을 지정합니다. 컬럼 이름 목록을 지정할 경우에는 스테이징 테이블을 정의할 구체화된 쿼리 테이블에 있는 실제 컬럼보다 *두 개* 많은 컬럼 이름으로 목록을 구성해야 합니다. 구체화된 쿼리 테이블이 복제된 구체화된 쿼리 테이블이거나 구체화된 쿼리 테이블을 정의하는 쿼리에 GROUP BY절이 없으면 스테이징 테이블을 정의할 구체화된 쿼리 테이블에 있는 실제 컬럼보다 *세 개* 많은 컬럼 이름으로 목록을 구성해야 합니다. 각 컬럼 이름은 고유해야 하며 규정되어서는 안됩니다. 컬럼 이름의 목록을 지정하지 않은 경

우, 테이블의 컬럼은 연관된, 구체화된 쿼리 테이블의 컬럼에 대한 이름을 상속합니다. 추가 컬럼의 이름은 GLOBALTRANSID와 GLOBALTRANSTIME이고 세 번째 컬럼이 필요할 경우 이 컬럼의 이름은 OPERATIONTYPE입니다.

표 19. 스테이징 테이블에 추가된 임시 컬럼

컬럼 이름	데이터 유형	컬럼 설명
GLOBALTRANSID	CHAR(8) FOR BIT DATA	전파된 각 행에 대한 전역 트랜잭션 ID
GLOBALTRANSTIME	CHAR(13) FOR BIT DATA	트랜잭션의 시간소인
OPERATIONTYPE	INTEGER	전파된 행에 대한 조작(예: 삽입, 갱신, 삭제)

연관된 구체화된 쿼리 테이블의 컬럼 중 생성된 컬럼 이름을 복제하는 컬럼이 있으면 컬럼 이름 목록을 반드시 지정해야 합니다(SQLSTATE 42711).

FOR *table-name2*

스테이징 테이블의 정의에 사용되는 구체화된 쿼리 테이블을 지정합니다. 내재적 또는 명시적 스키마를 포함한 이름은 REFRESH DEFERRED로 정의되어 현재 서버에 존재하는 구체화된 쿼리 테이블을 식별해야 합니다. 연관된 구체화된 쿼리 테이블의 fullselect는 REFRESH IMMEDIATE 옵션을 사용하여 구체화된 쿼리 테이블을 작성하는 데 사용되는 fullselect와 동일한 제한사항과 규칙을 따라야 합니다.

스테이징 테이블의 내용이 연관된 구체화된 쿼리 테이블 및 기본 소스 테이블과 일치할 경우에는 REFRESH TABLE문을 호출하여 구체화된 쿼리 테이블을 스테이징 테이블의 내용으로 새로 고칠 수 있습니다.

PROPAGATE IMMEDIATE

삭제, 삽입, 갱신 조작의 일부로서 하위 테이블에 대해 수행된 변경사항은 같은 삭제, 삽입, 갱신 조작의 스테이징 테이블에 연결됩니다. 스테이징 테이블이 일치하지 않은 것으로 표시되어 있지 않으면 구체화된 쿼리 테이블을 마지막으로 새로 고친 후 언제든지 스테이징 테이블의 내용이 하위 테이블에 대한 델타 변경이 됩니다.

ORGANIZE BY DIMENSIONS (*column-name***,...)**

테이블 데이터를 클러스터하는 데 사용되는 각 컬럼이나 컬럼 그룹에 대한 차원을 지정합니다. 차원 목록 내에서 괄호를 사용하면 컬럼 그룹이 한 차원으로 처리됩니다. DIMENSIONS 키워드는 선택적입니다. 정의가 이 절을 지정하는 테이블을 다차원적으로 클러스터된(MDC) 테이블이라고 합니다.

클러스터링 블록 인덱스는 지정한 각 차원에 대하여 자동으로 유지보수됩니다. 그리고 이 절에 사용되는 모든 컬럼으로 구성된 블록 인덱스는 클러스터링 블록 인덱

스 중 어떤 것에도 모든 컬럼이 포함되지 않은 경우에 유지보수됩니다. ORGANIZE BY절에 사용된 컬럼 세트는 CLUSTER를 지정하는 CREATE INDEX문에 대한 규칙을 따라야 합니다.

ORGANIZE BY절에 지정된 각 컬럼 이름은 테이블에 대해 정의해야 합니다 (SQLSTATE 42703). 차원은 차원 목록에 두 번 이상 나올 수 없습니다 (SQLSTATE 42709). 차원은 ROW CHANGE TIMESTAMP 컬럼(SQLSTATE 429BV) 또는 XML 컬럼(SQLSTATE 42962)을 포함할 수 없습니다.

테이블의 페이지는 동일한 *크기*의 블록에 배열됩니다. 이 *크기*는 테이블 스페이스의 Extent *크기*입니다. 그리고 각 블록의 모든 행에는 동일한 차원 값의 조합이 포함됩니다.

테이블은 다차원적으로 클러스터된(MDC) 테이블 및 파티션된 테이블 모두일 수 있습니다. 이러한 테이블의 컬럼은 *range-partition-spec* 및 MDC 키에서 쓰일 수 있습니다. 테이블 파티션은 다중 컬럼이지만 다차원이 아니라는 점을 유의하십시오.

ORGANIZE BY KEY SEQUENCE *sequence-key-spec*

키 시퀀스 값의 지정된 범위를 기반으로, *크기*가 고정된 오름차순 키 시퀀스로 테이블이 구성되도록 지정합니다. 이런 방법으로 구성된 테이블을 *범위 클러스터 테이블*이라고 합니다. 정의된 범위의 각 가능한 키 값에는 물리 테이블에서의 위치가 미리 정의되어 있습니다. 테이블 작성 시, 범위 클러스터 테이블에 필요한 스토리지가 사용 가능해야 하고 행 *크기*를 곱한 지정된 범위 내에서 행 수를 포함하기에 충분한 스토리지가 있어야 합니다(스페이스 요구사항 판별에 대한 자세한 내용은 행 크기 제한 및 바이트 수를 참조).

column-name

범위 클러스터 테이블의 시퀀스를 결정하는 고유 키에 포함될 하나의 테이블 컬럼을 지정합니다. 컬럼의 데이터 유형은 SMALLINT, INTEGER 또는 BIGINT이어야 하고(SQLSTATE 42611) 해당 컬럼은 NOT NULL로 정의되어야 합니다(SQLSTATE 42831). 같은 컬럼은 시퀀스 키에서 한 번 이상 식별될 수 없습니다. 식별된 컬럼의 수는 64를 초과할 수 없습니다(SQLSTATE 54008).

고유 인덱스 항목은 컬럼에 대해 자동으로 카탈로그에 작성됩니다(각 컬럼에 대해 오름차순으로). 인덱스 이름은 SQL 다음에 시간소인이 오는, SQL(*yymmddhhmmssxxx*)이 되며 스키마 이름은 SYSIBM이 됩니다. 실제 인덱스 오브젝트가 스토리지에 작성되지 않는 이유는 테이블 구성이 이 키로 정렬되기 때문입니다. 기본 키 또는 고유 제한조건이 범위 클러스터 테이블 시퀀스 키와 같은 컬럼에 정의되어 있는 경우 이 동일한 인덱스 항목은 제한조건에 사용됩니다.

키 시퀀스 스펙의 경우 점검 제한조건이 있어서 컬럼 제한조건을 반영합니다. DISALLOW OVERFLOW절을 지정한 경우 점검 제한조건의 이름은 RCT가

되며 점검 제한조건이 적용됩니다. ALLOW OVERFLOW절을 지정한 경우 점검 제한조건의 이름은 RCT_OFLOW가 되며 점검 제한조건이 적용되지는 않습니다.

STARTING FROM *constant*

column-name 범위의 아래쪽 끝에 상수 값을 지정합니다. 지정한 상수 미만의 값은 ALLOW OVERFLOW 옵션이 지정된 경우에만 허용됩니다. *column-name* 이 SMALLINT 또는 INTEGER 컬럼인 경우 상수는 INTEGER 상수이어야 합니다. *column-name*이 BIGINT 컬럼인 경우 상수는 INTEGER 또는 BIGINT 상수이어야 합니다(SQLSTATE 42821).시작 상수가 지정되지 않은 경우, 디폴트값은 1입니다.

ENDING AT *constant*

column-name 범위의 위쪽 끝에 상수 값을 지정합니다. 지정한 상수를 초과하는 값은 ALLOW OVERFLOW 옵션이 지정된 경우에만 허용됩니다. 종료 상수 값은 시작 상수보다 커야 합니다. *column-name*이 SMALLINT 또는 INTEGER 컬럼인 경우 상수는 INTEGER 상수이어야 합니다. *column-name* 이 BIGINT 컬럼인 경우 상수는 INTEGER 또는 BIGINT 상수이어야 합니다(SQLSTATE 42821).

ALLOW OVERFLOW

범위 클러스터 테이블이 정의된 값 범위 밖에 있는 키 값을 값는 행을 허용하도록 지정합니다. 오버플로우를 허용하는 범위 클러스터 테이블 작성시 범위 밖에 있는 키 값을 갖는 행은 미리 예정된 순서없이 정의된 범위 끝에 배치됩니다. 이러한 오버플로우 행이 발생하는 조작의 경우 정의된 범위 내의 키 값을 갖는 행에 대한 조작보다 효율성이 떨어집니다.

DISALLOW OVERFLOW

범위 클러스터 테이블이 정의된 값 범위 밖에 있는 키 값을 갖는 행을 허용하지 않도록 지정합니다. 오버플로우를 허용하지 않는 범위 클러스터 테이블은 모든 행을 키 시퀀스의 오름차순으로 유지보수합니다.

PCTFREE *integer*

여유 공간으로 남게 될 각 페이지의 백분율을 지정합니다. 제한없이 각 페이지의 첫 번째 행이 추가됩니다. 페이지에 행이 추가될 때 해당 페이지에 적어도 *integer* 퍼센트의 여유 공간이 남게 됩니다. *integer* 값의 범위는 0에서 99까지입니다. 시스템 카탈로그(SYSCAT.TABLES)에서 -1의 PCTFREE 값은 디폴트값으로 해석됩니다. 테이블 페이지에 대한 PCTFREE 디폴트값은 0입니다.

DATA CAPTURE

데이터베이스 간 데이터 복제에 대한 추가 정보를 로그에 기록할지 여부를 나타냅니다. 이 절은 서브테이블을 작성할 때에는 지정할 수 없습니다(SQLSTATE 42613).

테이블이 유형이 지정된 테이블이면, 이 옵션이 지원되지 않습니다(SQLSTATE 428DH 또는 42HDR).

NONE
추가 정보가 기록되지 않음을 나타냅니다.

CHANGES
이 테이블에 대한 SQL 변경사항에 대한 추가 정보가 로그에 기록됨을 나타냅니다. 이 옵션은 이 테이블이 복제되며 캡처 프로그램을 사용하여 이 테이블에 대한 변경사항을 로그에서 캡처할 경우에 필요합니다.

테이블의 (내재적 또는 명시적) 스키마 이름이 18바이트보다 크면, 이 옵션이 지원되지 않습니다(SQLSTATE 42997).

IN *tablespace-name*,...
테이블이 작성될 테이블 스페이스를 식별합니다. 테이블 스페이스는 존재해야 하고, 동일한 데이터베이스 파티션 그룹에 위치해야 하며, 명령문의 권한 부여 ID가 USE 특권을 보유하고 있는 모든 일반 DMS, 모든 대형 DMS 또는 모든 SMS 테이블 스페이스(SQLSTATE 42838)여야 합니다.

테이블 레벨에서는 하나의 IN절의 최대값을 사용할 수 있습니다. 테이블이 사용하는 모든 데이터 테이블 스페이스에는 동일한 페이지 *크기* 및 Extent *크기*가 있어야 합니다. 모두 동일한 프리페치 *크기*가 아닌 경우 경고가 리턴됩니다. 모든 테이블 스페이스가 AUTOMATIC 프리페치 *크기*인 경우 경고가 리턴되지 않습니다.

하나의 테이블 스페이스만 지정되면 모든 테이블 부분이 이 테이블 스페이스에 저장됩니다. 테이블 스페이스가 테이블 계층의 루트 테이블로부터 물려받은 것이므로, 이 절은 서브테이블을 작성할 때에는 지정할 수 없습니다(SQLSTATE 42613). 이 절을 지정하지 않으면, 테이블에 대한 테이블 스페이스는 다음과 같이 결정됩니다.

```
IF table space IBMDEFAULTGROUP (over which the user
    has USE privilege) exists with sufficient page size
        THEN choose it
ELSE IF a table space (over which the user has USE privilege)
    exists with sufficient page size (see below when
    multiple table spaces qualify)
        THEN choose it
ELSE return an error (SQLSTATE 42727)
```

둘 이상의 테이블 스페이스가 ELSE IF 조건으로 식별되는 경우, 가장 작은 *크기*의 충분한 페이지가 있는 테이블 스페이스를 선택하십시오. 둘 이상의 테이블 스페이스가 적격인 경우, USE 특권이 부여된 것에 따라 다음 우선순위대로 테이블 스페이스를 선택하십시오.

1. 권한 부여 ID
2. 권한 부여 ID가 속한 그룹
3. PUBLIC

여전히 둘 이상의 테이블 스페이스가 규정되었다면 데이터베이스 관리 프로그램에서 최종 선택을 합니다.

다음과 같은 경우 테이블 스페이스 결정사항이 변경될 수 있습니다.

- 테이블 스페이스가 삭제(drop) 또는 작성되었을 때
- USE 특권이 부여되었거나 취소되었을 때

파티션된 테이블은 다중 테이블 스페이스에 데이터 파티션을 분배할 수 있습니다. 다중 테이블 스페이스가 지정되면, 모든 테이블 스페이스가 SMS, 일반 DMS 또는 대형 DMS 테이블 스페이스로 존재해야 합니다(SQLSTATE 42838). 명령문의 권한 부여 ID는 모든 지정된 테이블 스페이스에 USE 특권을 보유하고 있어야 합니다.

테이블의 페이지 *크기*가 충분한지 여부는 행의 바이트 합계 또는 컬럼 수에 의해 결정됩니다. 자세한 정보는 행 *크기* 제한을 참조하십시오.

테이블이 대형 테이블 스페이스에 위치한 경우는 다음과 같은 등록 정보를 나타냅니다.

- 일반 테이블 스페이스의 테이블보다 클 수 있습니다. 테이블 및 테이블 스페이스 한계에 대한 자세한 내용은 『SQL 한계』를 참조하십시오.
- 테이블은 데이터 페이지당 255행 이상을 지원할 수 있으며 이는 데이터 페이지의 스페이스 사용도를 개선할 수 있습니다.
- 테이블에 정의된 인덱스는 일반 테이블 스페이스에 있는 테이블에 정의된 인덱스에 비해 행 항목당 2바이트가 더 필요합니다.

CYCLE 또는 NO CYCLE

명시적 테이블 스페이스가 없는 테이블 스페이스 수가 지정된 데이터 파티션의 수를 넘을 수 있는지 지정합니다.

CYCLE

명시적 테이블 스페이스가 없는 데이터 파티션의 수가 지정된 테이블 스페이스의 수를 초과하는 경우, 테이블 스페이스가 라운드 로빈 방식으로 데이터 파티션에 할당되도록 지정합니다.

NO CYCLE

명시적 테이블 스페이스가 없는 데이터 파티션의 수가 지정된 테이블 스페이스의 수를 초과할 수 없도록 지정합니다(SQLSTATE 428G1). 이 옵션은 테이블 스페이스가 라운드 로빈 방식으로 데이터 파티션에 할당되는 것을 방지합니다.

tablespace-options

인덱스나 긴 컬럼 값이 저장되는 테이블 스페이스를 지정합니다. 테이블 스페이스의 유형에 대한 자세한 내용은 『CREATE TABLESPACE』를 참조하십시오.

INDEX IN *tablespace-name*

파티션되지 않은 테이블의 인덱스 또는 파티션된 테이블의 파티션되지 않은 인덱스가 작성될 테이블 스페이스를 식별합니다. 지정된 테이블 스페이스가 있어야 하고, 해당 테이블에 DMS 테이블 스페이스의 데이터가 있는 경우

에는 DMS 테이블 스페이스여야 하거나 파티션된 테이블에 SMS 테이블 스페이스의 데이터가 있는 경우에는 SMS 테이블 스페이스여야 합니다. 또한 이 테이블 스페이스는 명령문의 권한 부여 ID에 USE 특권이 있는 테이블 스페이스여야 하며 *tablespace-name*과 동일한 데이터베이스 파티션 그룹에 속해야 합니다(SQLSTATE 42838).

테이블 작성시 또는 파티션된 테이블의 경우는 파티션되지 않은 인덱스에 대하여 CREATE INDEX문의 IN절을 지정하여 인덱스가 있는 테이블 스페이스를 지정할 수 있습니다. 테이블 스페이스의 USE 특권에 대한 점검은 나중에 인덱스 작성시가 아니라 테이블 작성시에 이루어집니다.

파티션된 테이블의 파티션되지 않은 인덱스의 경우 인덱스의 스토리지는 다음과 같습니다.

- CREATE INDEX 명령문의 IN절에 의한 테이블 스페이스
- 테이블 레벨 테이블 스페이스가 CREATE TABLE 명령문의 INDEX IN 절에 대해 지정하였음
- 위에 표시된 내용 중 하나도 지정되지 않은 경우 인덱스는 첫 번째로 접속한 테이블 스페이스에 저장되거나 표시되는 데이터 파티션에 저장됨

파티션된 테이블에서 파티션된 인덱스에 대한 정보는 파티션 요소 INDEX IN절의 설명을 참조하십시오.

LONG IN *tablespace-name*

긴 컬럼 값이 저장되는 테이블 스페이스를 식별합니다. Long 컬럼은 LOB 데이터 유형, XML 유형, 소스 유형으로 이들 유형을 포함한 구별 유형 또는 인라인에 저장할 수 없는 값을 가진 사용자 정의 구조화된 유형인 컬럼을 포함합니다. 이 옵션은 IN절이 DMS 테이블 스페이스를 식별하는 경우에만 사용할 수 있습니다.

지정된 테이블 스페이스가 존재해야 합니다. 데이터가 저장된 곳이 같은 테이블 스페이스인 경우 일반 테이블 스페이스일 수 있습니다. 그 외의 경우 명령문의 권한 부여 ID에 USE 특권이 있으면 큰 DMS 테이블 스페이스여야 합니다. *tablespace-name*과 동일한 데이터베이스 파티션 그룹이어야 합니다(SQLSTATE 42838).

긴 컬럼, LOB 컬럼 또는 XML 컬럼이 있는 테이블 스페이스의 지정은 테이블 작성시에만 가능합니다. USE 특권에 대한 점검은 나중에 긴 컬럼 또는 LOB 컬럼을 추가할 때가 아니라 테이블 작성시에 이루어집니다.

파티션된 테이블이 있는 LONG IN 절의 사용에 관한 규칙에 대해서는 『파티션된 테이블에서의 대형 오브젝트(LOB) 동작』을 참조하십시오.

distribution-clause

데이터베이스 파티션 또는 다중 데이터베이스 파티션에 데이터를 분배하는 방법을 지정합니다.

DISTRIBUTE BY HASH (*column-name*,...)

분산 *키*라고 불리는 지정 컬럼의 디폴트 해싱 함수 사용을 데이터베이스 파티션에 분산 메소드로 지정합니다. *column-name* 은 테이블의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다(SQLSTATE 42703). 동일한 컬럼은 두 번 이상 식별될 수 없습니다(SQLSTATE 42709). 데이터 유형이 BLOB, CLOB, DBCLOB, XML, 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형을 갖는 컬럼은 분산 키의 일부로 사용될 수 없습니다(SQLSTATE 42962). 분산 키는 ROW CHANGE TIMESTAMP 컬럼을 포함할 수 없습니다(SQLSTATE 429BV). 분산 키는 테이블 계층의 루트 테이블 또는 XML 데이터 유형 컬럼을 가진 테이블(SQLSTATE 42997)로부터 물려받은 것이므로, 서브테이블인 테이블에 대해 분산 키를 지정할 수 없습니다(SQLSTATE 42613). 해당 절이 지정되지 않고 테이블이 다중 데이터베이스 파티션이 있는 다중 파티션 데이터베이스 파티션 그룹에 상주할 경우, 분산 키는 다음과 같이 정의됩니다.

- 유형이 지정된 테이블인 경우, 오브젝트 ID 컬럼이 분산 키입니다
- 기본 키가 정의되는 경우, 기본 키의 첫 번째 컬럼이 분산 키입니다.
- 기본 키가 정의되지 않는 경우, 분산 키에 유효한 데이터 유형을 가진 첫 번째 컬럼이 분산 키가 됩니다.

분산 키의 컬럼은 고유 제한조건을 구성하는 컬럼의 서브세트여야 합니다.

디폴트 분산 키 요건을 충족시키는 컬럼이 없을 경우, 분산 키 없이 테이블이 작성됩니다. 이러한 테이블은 단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에서만 사용할 수 있습니다.

단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 있는 테이블의 경우, 분산 키에 대해 유효한 데이터 유형을 가진 컬럼의 콜렉션은 분산 키를 정의하는 데 사용될 수 있습니다. 이 절을 지정하지 않으면 분산 키가 작성되지 않습니다.

분산 키와 관련된 제한사항에 대해서는 규칙을 참조하십시오.

DISTRIBUTE BY REPLICATION

테이블에 저장된 데이터가 테이블이 정의된 테이블 스페이스의 데이터베이스 파티션 그룹에 있는 각 데이터베이스 파티션에 실제로 복제되도록 지정합니다. 이는 테이블에 있는 모든 데이터의 사본이 각각의 데이터베이스 파티션에 존재한다는 뜻입니다. 이 옵션은 구체화된 쿼리 테이블에 대해서만 지정할 수 있습니다(SQLSTATE 42997).

partitioning-clause

데이터가 데이터베이스 파티션에서 파티션되는 방법을 지정합니다.

PARTITION BY RANGE *range-partition-spec*

테이블에 대한 테이블 파티션 스킴을 지정합니다.

partition-expression

해당 데이터의 목표 데이터 파티션을 판별하도록 정의된 범위에 적용되는 키 데이터를 지정합니다.

column-name

테이블 파티셔닝 키 컬럼을 식별합니다. *column-name* 은 테이블의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다(SQLSTATE 42703). 동일한 컬럼은 두 번 이상 식별될 수 없습니다(SQLSTATE 42709). 데이터 유형이 BLOB, CLOB, DBCLOB, XML 중 하나를 기반으로 하는 구별 유형 또는 구조화 유형인 컬럼은 테이블 파티셔닝 키의 일부로 사용될 수 없습니다(SQLSTATE 42962).

범위 스펙에서 사용된 숫자 리터럴은 숫자 리터럴에 대한 규칙에 의해 제어됩니다. 숫자 상수에 대해 지정된 규칙에 따라, 숫자 컬럼에 해당되는 범위에서 사용된 모든 숫자 리터럴은(10진수 부동 소수점 특수 값 제외) 정수, 부동 소수점 또는 10진 상수로 해석됩니다. 결과로 10진수 부동 소수점 컬럼의 경우, 데이터 파티션의 범위 스펙에서 사용될 수 있는 최소 및 최대 숫자 상수 값은 각각 가장 작은 DOUBLE 값과 가장 큰 DOUBLE 값입니다. 10진수 부동 소수점 특수 값은 범위 스펙에서 사용될 수 있습니다. 모든 10진수 부동 소수점 특수 값은 MINVALUE 보다 크고 MAXVALUE 보다 작은 것으로 해석됩니다.

테이블 파티션 컬럼은 ROW CHANGE TIMESTAMP 컬럼을 포함할 수 없습니다(SQLSTATE 429BV). 식별된 컬럼의 수는 16을 초과할 수 없습니다(SQLSTATE 54008).

NULLS LAST

널(Null) 값은 상위 비교함을 표시합니다.

NULLS FIRST

널(Null) 값은 하위 비교함을 표시합니다.

partition-element

데이터 파티션 키 범위 및 해당 범위의 테이블 행이 저장될 테이블 스페이스를 지정합니다.

PARTITION *partition-name*

데이터 파티션의 이름을 지정합니다. 테이블의 다른 데이터 파티션 이름과 달라야 합니다(SQLSTATE 42710). 이 절이 지정되지 않는 경우 테이블의 고유 이름을 작성하기 위해 정수 값의 문자 양식이 뒤에 붙는 'PART'가 이름이 됩니다.

boundary-spec

범위 파티션의 바운더리를 지정합니다. 최하위 범위 파티션은 starting-clause를 포함해야 하고 최상위 범위 파티션은 ending-clause

를 포함해야 합니다(SQLSTATE 56016). 최하위 범위 파티션과 최상위 범위 파티션 사이에는 starting-clause, ending-clause 또는 두 가지 모두가 있을 수 있습니다. ending-clause만 지정된 경우 이전 범위 파티션에도 ending-clause이 존재했어야 합니다(SQLSTATE 56016).

starting-clause
데이터 파티션 범위의 하위 끝을 지정합니다. 최소 하나의 지정된 시작값 및 데이터 파티션 키의 컬럼 수보다 적은 수의 값이 있어야 합니다(SQLSTATE 53038). 컬럼 수보다 적은 수의 값이 있으면 나머지 값은 내재적으로 MINVALUE이 됩니다.

STARTING FROM
*starting-clause*를 시작합니다.

constant
해당 *column-name*의 데이터 유형에 할당 가능한 데이터 유형을 가진 상수 값을 지정합니다(SQLSTATE 53045). 테이블에 대한 다른 바운더리 스펙의 범위 내에 값이 있으면 안됩니다(SQLSTATE 56016).

MINVALUE
해당하는 *column-name*의 데이터 유형에 대한 최하위 가능 값보다 더 낮은 값을 지정합니다.

MAXVALUE
해당하는 *column-name*의 데이터 유형에 대한 가장 큰 가능 값보다 더 큰 값을 지정합니다.

INCLUSIVE
지정된 범위 값이 데이터 파티션에 포함됨을 표시합니다.

EXCLUSIVE
지정된 *constant* 값이 데이터 파티션에서 제외됨을 표시합니다. 이 스펙은 MINVALUE 또는 MAXVALUE가 지정되면 무시됩니다.

ending-clause
데이터 파티션 범위의 상위 끝을 지정합니다. 최소 하나의 지정된 시작값 및 데이터 파티션 키의 컬럼 수보다 적은 수의 값이 있어야 합니다(SQLSTATE 53038). 컬럼 수보다 적은 수의 지정된 값이 있으면 나머지 값은 내재적으로 MAXVALUE가 됩니다.

ENDING AT
*ending-clause*를 시작합니다.

constant
해당 *column-name*의 데이터 유형에 할당 가능한 데이터

유형을 가진 상수 값을 지정합니다(SQLSTATE 53045). 테이블에 대한 다른 바운더리 스펙의 범위 내에 값이 있으면 안됩니다(SQLSTATE 56016).

MINVALUE
해당하는 *column-name*의 데이터 유형에 대한 최하위 가능 값보다 더 낮은 값을 지정합니다.

MAXVALUE
해당하는 *column-name*의 데이터 유형에 대한 가장 큰 가능 값보다 더 큰 값을 지정합니다.

INCLUSIVE
지정된 범위 값이 데이터 파티션에 포함됨을 표시합니다.

EXCLUSIVE
지정된 *constant* 값이 데이터 파티션에서 제외됨을 표시합니다. 이 스펙은 MINVALUE 또는 MAXVALUE가 지정되면 무시됩니다.

IN *tablespace-name*
데이터 파티션이 저장될 테이블 스페이스를 지정합니다. 이름 지정된 테이블 스페이스는 동일한 페이지 크기여야 하고, 동일한 데이터베이스 파티션 그룹에 속해야 하며, 파티션된 테이블의 다른 테이블 스페이스와 동일한 방법으로 관리되어야 합니다(SQLSTATE 42838). 또한 이 테이블 스페이스는 명령문의 권한 부여 ID에 USE 특권이 있는 테이블 스페이스여야 합니다. 이 절이 지정되지 않으면, 테이블 스페이스는 해당 테이블에 지정된 테이블 스페이스 목록에서 라운드 로빈 방식을 사용하여 할당되는 것이 디폴트값입니다. 테이블 스페이스가 LONG IN 절을 사용하여 대형 오브젝트에 대해 지정되지 않은 경우, 대형 오브젝트는 데이터 파티션의 나머지 행이 위치하는 동일한 테이블 스페이스에 위치합니다. 파티션된 테이블의 경우, LONG IN절이 테이블 스페이스 목록을 제공하는 데 쓰일 수 있습니다. 이 목록은 각 데이터 파티션의 대형 오브젝트를 라운드 로빈 방식으로 위치시키는 데 쓰입니다. 파티션된 테이블이 있는 LONG IN 절의 사용에 관한 규칙에 대해서는 『파티션된 테이블에서의 대형 오브젝트(LOB) 동작』을 참조하십시오.

INDEX IN절이 CREATE TABLE 또는 CREATE INDEX문에 지정되지 않으면, 인덱스는 테이블의 첫 번째 보이는 파티션 또는 접속된 파티션과 동일한 테이블 스페이스에 위치합니다.

INDEX IN *tablespace-name*
파티션된 테이블에서 파티션된 인덱스가 저장될 테이블 스페이스를 지정합니다.

파티션 요소 레벨 INDEX IN절은 파티션된 인덱스의 스토리지에만 영향을 줍니다. 인덱스의 스토리지는 다음과 같습니다.

- 테이블이 작성될 때 INDEX IN절이 파티션 레벨에서 지정된 경우 파티션된 인덱스는 지정된 테이블 스페이스에 저장됩니다.
- 테이블이 작성될 때 INDEX IN절이 파티션 레벨에 지정되지 않은 경우 파티션된 인덱스는 해당 데이터 파티션의 테이블 스페이스에 저장됩니다.

INDEX IN절은 데이터 테이블 스페이스가 DMS 테이블 스페이스이고 INDEX IN절에 지정된 테이블 스페이스가 DMS 테이블 스페이스인 경우에만 지정될 수 있습니다. 데이터 테이블 스페이스가 SMS 테이블 스페이스인 경우 오류가 리턴됩니다(SQLSTATE 42839).

LONG IN *tablespace-name*

긴 컬럼 값이 저장되는 테이블 스페이스를 식별합니다. Long 컬럼은 LOB 데이터 유형, XML 유형, 소스 유형으로 이들 유형을 포함한 구별 유형 또는 인라인에 저장할 수 없는 값을 가진 사용자 정의 구조화된 유형으로 정의된 컬럼을 포함합니다. 이 옵션은 IN절이 DMS 테이블 스페이스를 식별하는 경우에만 사용할 수 있습니다.

지정된 테이블 스페이스가 존재해야 합니다. 데이터가 저장된 곳이 같은 테이블 스페이스인 경우 일반 테이블 스페이스일 수 있습니다. 그 외의 경우 명령문의 권한 부여 ID에 USE 특권이 있으면 큰 DMS 테이블 스페이스여야 합니다. *tablespace-name*과 동일한 데이터베이스 파티션 그룹이어야 합니다(SQLSTATE 42838).

Long, LOB 또는 XML 컬럼이 있는 테이블 스페이스의 지정은 테이블 작성 시에만 가능합니다. USE 특권에 대한 점검은 나중에 긴 컬럼 또는 LOB 컬럼을 추가할 때가 아니라 테이블 작성시에 이루어집니다.

파티션된 테이블이 있는 LONG IN 절의 사용에 관한 규칙에 대해서는 『파티션된 테이블에서의 대형 오브젝트(LOB) 동작』을 참조하십시오.

EVERY (*constant*)

자동으로 생성된 구문 양식을 사용하는 경우 각 데이터 파티션 범위의 너비를 지정합니다. 데이터 파티션이 작성되고 STARTING FROM 값에서 시작되며 범위에 있는 값의 해당 번호를 포함합니다. 구문의 해당 형식은 단일 숫자 컬럼 또는 날짜 시간 컬럼이 파티션한 테이블에 대해서만 지원됩니다(SQLSTATE 53038).

파티션 키 컬럼이 숫자 유형이면 첫 번째 파티션의 시작값은 starting-clause에 지정된 값입니다. 첫 번째 및 다른 파티션 모두에 대한 종료값은 파티션의 시작값을 EVERY절에 있는 *constant*로 지정된

증분값에 추가함으로써 계산됩니다. 다른 파티션 모두에 대한 시작값은 이전 파티션의 시작값을 가져옴으로써, 그리고 EVERY절에 있는 *constant*로 지정된 증분값을 추가함으로써 계산됩니다.

파티션 키 컬럼이 DATE 또는 TIMESTAMP이면 첫 번째 파티션의 시작값은 starting-clause에 지정된 값입니다. 첫 번째 및 다른 파티션 모두에 대한 종료값은 파티션의 시작값을 EVERY절에 있는 레이블된 지속 기간으로 지정된 증분값에 추가함으로써 계산됩니다. 다른 파티션 모두에 대한 시작값은 이전 파티션의 시작값을 가져옴으로써, 그리고 EVERY절에 있는 레이블된 지속 기간으로 지정된 증분값을 추가함으로써 계산됩니다.

숫자 컬럼의 경우 EVERY 값은 양의 숫자 상수여야 하고, 날짜 시간 컬럼의 경우는 레이블된 지속 기간이어야 합니다(SQLSTATE 53045).

COMPRESS

데이터 압축을 테이블의 행에 적용할지 여부를 지정합니다.

YES

데이터 행 압축을 사용 가능하도록 지정합니다. 테이블에 대한 삽입 및 갱신 조작은 압축의 영향을 받습니다. 테이블이 데이터로 채워진 다음, 압축 사전이 자동으로 작성되며 행이 압축의 영향을 받습니다. XML 스토리지 오브젝트의 데이터에도 적용됩니다. XML 스토리지 오브젝트에 충분한 데이터가 있는 경우, 압축 사전이 자동으로 작성되며 XML 문서가 압축의 영향을 받습니다.

NO

데이터 행 압축을 사용 가능하도록 지정합니다.

VALUE COMPRESSION

이것은 사용할 행 형식을 결정합니다. 각 데이터 유형은 사용되는 행 형식에 따라 서로 다른 바이트 수를 갖습니다. 자세한 정보는 바이트 수를 참조하십시오. 테이블이 유형이 지정된 테이블이라면, 이 옵션은 유형이 지정된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

널(NULL) 값은 3 바이트를 사용하여 저장됩니다. 이것은 CHAR(1)을 제외한 모든 데이터 유형의 컬럼에 대해 VALUE COMPRESSION이 활성화되지 않았을 때보다 작거나 같은 스페이스입니다. 컬럼이 널(NULL) 입력 가능으로 정의되었는지 여부가 행 크기 계산에 영향을 주지 않습니다. 데이터 유형이 VARCHAR, VARGRAPHIC, LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB, BLOB 또는 XML인 컬럼의 0 길이 데이터 값은 VALUE COMPRESSION이 비활성화일 때 필요한 스토리지보다 작은 2바이트만 사용하여 저장됩니다. 컬럼을 COMPRESS SYSTEM DEFAULT 옵션으로 정의하면 컬럼의 시스템 디폴트 값이 전체 스토리지의 3바이트를 사용하여 저장될 수 있습니다. 이를 지원하는 데 사용되는 행 형식은 각 데이터 유형에 대한 바이트 수를 결정하고 널(NULL), 0길이 값 또는 시스템 디폴트 값으로 갱신하는 동안 데이터를 단편화합니다.

WITH RESTRICT ON DROP
테이블과 테이블이 포함된 테이블 스페이스를 삭제할 수 없도록 지정합니다.

NOT LOGGED INITIALLY
테이블이 작성된 동일한 작업 단위(UOW)에서 삽입, 삭제, 갱신, 인덱스 작성, 인덱스 삭제 또는 테이블 변경 조작으로 인해 생긴 테이블 변경사항이 기록되지 않습니다. 이 옵션 사용시 기타 고려사항에 대해서는 이 명령문의 『주』를 참조하십시오.

모든 카탈로그 변경사항 및 스토리지 관련 정보는 후속되는 단위 작업에서 테이블에 대해 수행되는 모든 조작과 마찬가지로 기록됩니다.

주: NOT LOGGED INITIALLY 속성이 활성화된 테이블에 대하여 비로그가 수행되고, 명령문이 실패하여 롤백이 발생하거나 ROLLBACK TO SAVEPOINT가 실행될 경우에는 전체 작업 단위가 롤백됩니다(SQL1476N). 또한 롤백이 발생한 후 NOT LOGGED INITIALLY 속성이 활성화된 테이블은 액세스 불가로 표시되어 삭제만 가능합니다. 따라서 NOT LOGGED INITIALLY 속성이 사용 중인 작업 단위 내에서 오류가 발생할 가능성은 최소로 해야 합니다.

CCSID
테이블에 저장된 문자열 데이터에 대한 코드화 체계를 지정합니다. CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 CCSID UNICODE이고 다른 모든 데이터베이스에서 디폴트값은 CCSID ASCII입니다.

ASCII
문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031).

UNICODE
문자열 데이터를 유니코드로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, 문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있습니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, 문자 데이터는 UTF-8로 되어 있습니다.

데이터베이스가 유니코드 데이터베이스가 아닐 경우, CCSID UNICODE로 테이블을 작성할 수는 있지만 다음 규칙이 적용됩니다.

- 테이블을 작성하기 전에 데이터베이스 구성에 대해 대체 조합 시퀀스를 지정해야 합니다(SQLSTATE 56031). CCSID UNICODE 테이블은 데이터베이스 구성에 지정된 대체 조합 시퀀스와 조합됩니다.
- CCSID ASCII로 작성된 테이블 또는 테이블 함수와 CCSID UNICODE로 작성된 테이블 또는 테이블 함수 모두 단일 SQL문에서는 사용할 수 없습니다(SQLSTATE 53090). 이 사항은 명령문에서 직접 참조된 테이블 및 테

이블 함수는 물론 간접적으로 참조된(예: 참조 무결성 제한조건, 트리거, 구체화된 쿼리 테이블 및 뷰 본문에 있는 테이블을 통해서) 테이블 및 테이블 함수에도 적용됩니다.

- CCSID UNICODE로 작성된 테이블은 SQL 함수 또는 SQL 메소드에서 참조할 수 없습니다(SQLSTATE 560C0).
- CCSID UNICODE로 작성된 테이블을 참조하는 SQL문은 SQL 함수 또는 SQL 메소드를 호출할 수 없습니다(SQLSTATE 53090).
- 그래픽 유형, XML 유형 및 사용자 정의 유형을 CCSID UNICODE 테이블에서 사용할 수 없습니다(SQLSTATE 560C1).
- 앵커된 데이터 유형은 CCSID UNICODE (SQLSTATE 428HS)로 작성된 테이블의 컬럼에 고정할 수 없습니다.
- 테이블에 CCSID UNICODE절과 DATA CAPTURE CHANGES절 둘 다 지정할 수 없습니다(SQLSTATE 42613).
- Explain 테이블은 CCSID UNICODE로 작성할 수 없습니다(SQLSTATE 55002).
- 작성된 임시 테이블 및 선언된 임시 테이블은 CCSID UNICODE로 작성할 수 없습니다(SQLSTATE 56031).
- CCSID UNICODE 테이블은 CREATE SCHEMA문으로 작성할 수 없습니다(SQLSTATE 53090).
- 로드 조작의 예외 테이블은 조작에 대한 목표 테이블과 동일한 CCSID를 가져야 합니다(SQLSTATE 428A5).
- SET INTEGRITY문의 예외 테이블은 명령문에 대한 목표 테이블과 동일한 CCSID를 가져야 합니다(SQLSTATE 53090).
- 이벤트 모니터의 목표 테이블은 CCSID UNICODE로 정의하면 안됩니다(SQLSTATE 55049).
- CCSID UNICODE 테이블을 참조하는 명령문은 DB2 버전 8.1 또는 최신 클라이언트에서만 호출될 수 있습니다(SQLSTATE 42997).
- SQL문은 언제나 데이터베이스 코드 페이지에서 해석됩니다. 특히, 이것은 리터럴, 16진 리터럴 및 분리 ID의 모든 문자에 대한 표현이 데이터베이스 코드 페이지에 있어야 한다는 것을 의미합니다. 그렇지 않을 경우, 문자는 대체 문자로 교체됩니다.

응용프로그램에 있는 호스트 변수는 호출되는 SQL문에 있는 테이블의 CCSID와 상관없이 언제나 응용프로그램 코드 페이지에 있습니다. DB2는 코드 페이지 변환을 수행하여 필요에 따라 응용프로그램 코드 페이지와 섹션 코드 페이지 사이에 데이터를 변환합니다. 응용프로그램 코드 페이지를 변경하기 위해 레지스트리 변수 DB2CODEPAGE를 클라이언트에서 설정할 수 있습니다.

SECURITY POLICY

테이블과 연결될 보안 규정의 이름을 짓습니다.

policy-name

현재 서버에 이미 존재하는 보안 규정을 식별합니다(SQLSTATE 42704).

OPTIONS (ADD *table-option-name string-constant*, **...)**

테이블 옵션은 리모트 기본 테이블을 식별하는 데 사용됩니다. *table-option-name*은 옵션의 이름입니다. *string-constant*는 테이블 옵션에 대한 설정을 지정합니다. *string-constant*는 작은따옴표로 묶어야 합니다.

리모트 서버(CREATE SERVER문에 지정되어 있는 서버 이름)는 OPTIONS절로 지정해야 합니다. OPTIONS절은 작성되는 리모트 기본 테이블의 규정되지 않은 이름이나 스키마를 겹쳐쓰는 데에도 사용할 수 있습니다.

스키마 이름을 지정하는 것이 바람직합니다. 리모트 스키마 이름을 지정하지 않은 경우 테이블 이름의 규정자가 사용됩니다. 테이블 이름에 규정자가 없는 경우 명령문의 권한 부여 ID가 사용됩니다.

리모트 기본 테이블의 규정되지 않은 이름을 지정하지 않은 경우, *table-name*이 사용됩니다.

규칙

- 모든 구조화된 유형을 포함한 컬럼 또는 XML 유형 컬럼의 인라인 길이를 포함한 컬럼의 바이트 수 합은 테이블 스페이스의 페이지 크기에 기초하는 행 크기 한계보다 커서는 안됩니다(SQLSTATE 54010). 자세한 정보는 바이트 수를 참조하십시오. 유형이 지정된 테이블의 경우, 바이트 합계는 테이블 계층의 루트 테이블 컬럼과 테이블 계층에 있는 모든 서브테이블에 의해 도입된 모든 추가 컬럼에 적용됩니다. 널(NULL) 입력이 불가능한 것으로 정의된 경우에도, 바이트 계산상 추가 서브테이블 컬럼을 널(NULL) 입력이 가능한 것으로 간주해야 합니다. 각 행이 속해 있는 서브테이블을 찾기 위해 4바이트의 추가 오버헤드도 발생합니다.
- 테이블의 컬럼 수는 1 012를 초과할 수 없습니다(SQLSTATE 54011). 유형이 지정된 테이블의 경우, 테이블 계층에 있는 모든 서브테이블의 총 유형 속성 수는 1010을 초과할 수 없습니다.
- 유형이 지정된 테이블의 오브젝트 ID 컬럼은 갱신할 수 없습니다(SQLSTATE 42808).
- 테이블에 정의된 고유 키 또는 기본 키 제한조건은 분산 키의 수퍼 세트여야 합니다(SQLSTATE 42997).
- 다음 규칙은 다중 데이터베이스 파티션 데이터베이스에만 적용됩니다.
  - LOB, XML, 이러한 유형 중 하나를 기반으로 하는 구별 유형 또는 구조화된 유형 컬럼으로만 이루어진 테이블은 단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에서만 작성될 수 있습니다.

  - 다중 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 있는 테이블의 분산 키 정의는 변경될 수 없습니다.
  - 유형이 지정된 테이블의 분산 키 컬럼은 OID 컬럼이어야 합니다.
  - 파티션된 스테이징 테이블을 지원하지 않습니다.
- 다음 제한사항은 범위 클러스터된 테이블에 적용됩니다.
  - 범위 클러스터 테이블은 다중 데이터베이스 파티션이 있는 데이터베이스에 지정할 수 없습니다(SQLSTATE 42997).
  - 클러스터링 인덱스를 작성할 수 없습니다.
  - 테이블을 변경하여 컬럼을 추가하는 것은 지원되지 않습니다.
  - 테이블을 변경하여 컬럼의 데이터 유형을 변경하는 것은 지원되지 않습니다.
  - 테이블을 변경하여 PCTFREE를 변경하는 것은 지원되지 않습니다.
  - 테이블을 변경하여 APPEND ON을 설정하는 것은 지원되지 않습니다.
  - DETAILED 통계는 사용 가능하지 않습니다.
  - 테이블을 채우는 데 로드 유틸리티를 사용할 수 없습니다.
  - 컬럼은 XML 유형일 수 없습니다.
- 테이블과 연결된 보안 규정이 없고 DB2SECURITYLABEL 유형 컬럼 또는 SECURED WITH절과 함께 정의된 컬럼을 포함하지 않는 경우 테이블은 보호되지 않습니다. 전자는 **행 레벨 세분화도**를 가진 보호된 테이블임을 나타내고 후자는 **컬럼 레벨 단위**를 가진 보호된 테이블임을 나타냅니다.
- 테이블과 연결된 보안 규정이 없는 경우 DB2SECURITYLABEL 유형 컬럼을 선언을 할 수 없습니다(SQLSTATE 55064).
- 유형이 지정된 테이블(SQLSTATE 428DH), 구체화된 쿼리 테이블 또는 스테이징 테이블 (SQLSTATE 428FG)에 보안 규정을 추가할 수 없습니다.
- 오류가 허용되는 *nested-table-expression*은 *materialized-query-definition*의 fullselect에 지정될 수 없습니다(SQLSTATE 428GG).
- *isolation-clause*는 *materialized-query-table-definition*의 *full-select*에 지정될 수 없습니다(SQLSTATE 42601).
- *lock-request-clause*를 포함하는 Subselect 명령문은 MQT 경로지정에 적합하지 않습니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 테이블을 작성할 경우 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 갖고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 외부 키를 지정할 경우

  - 상위 테이블에서 삭제 사용을 갖고 있는 모든 패키지가 무효화됩니다.
  - 상위 키 내에서 적어도 한 컬럼에 대해 갱신 사용을 갖고 있는 모든 패키지가 무효화됩니다.
- 서브테이블을 작성하면 테이블 계층의 다른 테이블에 종속된 패키지가 모두 무효화됩니다.
- 각기 4 000과 2 000보다 큰 VARCHAR 및 VARGRAPHIC 컬럼은 SYSFUN 스키마에 있는 함수의 입력 매개변수로 사용되어서는 안됩니다. 이 길이를 초과하는 인수 값으로 함수가 호출되면 오류가 발생합니다(SQLSTATE 22001).
- 참조 제한조건에 대한 삭제 또는 갱신 규칙으로 NO ACTION 또는 RESTRICT를 사용하면 제한조건이 시행되는 기기를 판별할 수 있습니다. RESTRICT의 삭제 또는 갱신 규칙은 CASCADE 또는 SET NULL과 같은 수정 규칙과 함께 이러한 참조 제한조건을 포함하여 다른 모든 제한조건 *이전에* 실행됩니다. NO ACTION의 삭제 또는 갱신 규칙은 다른 참조 제한조건 *이후에* 실행됩니다. 서로 다른 동작이 명백한 영향을 주는 한 예로, 관련 테이블의 UNION ALL로 정의되어 있는 뷰에서의 행 삭제를 들 수 있습니다.

  ```
  Table T1 is a parent of table T3; delete rule as noted below.
     Table T2 is a parent of table T3; delete rule CASCADE.

     CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2

     DELETE FROM V1
  ```

  테이블 T1이 테이블 T3의 상위 테이블이고 삭제 규칙이 RESTRICT인 경우, T3에 T1의 상위 키에 대한 하위 행이 하나라도 있으면 RESTRICT 위반이 발생합니다(SQLSTATE 23001).

  테이블 T1이 테이블 T3의 상위 테이블이고 삭제 규칙은 NO ACTION인 경우, NO ACTION 삭제 규칙이 T1에서의 삭제를 위해 시행되기 전에 T2에서 행을 삭제하면 CASCADE 삭제 규칙이 하위 행을 삭제할 수 있습니다. T2에서의 삭제로 인해 T3에 있는 T1의 상위 키에 대한 모든 하위 행이 삭제되지 않는 경우, 제한조건 위반이 발생합니다(SQLSTATE 23504).

  리턴되는 SQLSTATE는 삭제 또는 갱신 규칙이 RESTRICT 또는 NO ACTION인지에 따라 다릅니다.
- 다중 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 있는 테이블의 경우, 분산 키 선택시 테이블 공동 배치를 고려해야 합니다. 다음은 고려해야 할 항목의 목록입니다.
  - 공동 배치하려면 테이블이 동일한 데이터베이스 파티션 그룹에 있어야 합니다. 테이블 스페이스는 다를 수 있지만, 동일한 데이터베이스 파티션 그룹에 정의되어야 합니다.

  - 테이블의 분산 키에는 동일한 수의 컬럼이 있어야 하며 해당 키 컬럼은 공동 배치에 대해 데이터베이스 파티션 호환이 가능해야 합니다.
  - 분산 키 선택은 조인의 성능에도 영향을 미칩니다. 테이블이 다른 테이블과 자주 조인되는 경우, 두 테이블에 대한 분산 키로 조인 컬럼을 간주해야 합니다.
- NOT LOGGED INITIALLY 옵션은 대체 소스(다른 테이블이나 파일)의 데이터를 통해 대량의 결과 세트를 작성해야 하는 상황에서 유용하며 테이블 복구는 필요하지 않습니다. 이 옵션을 사용하면 데이터 로깅에 따른 오버헤드가 감소합니다. 다음 고려사항은 이 옵션이 지정될 때 적용됩니다.
  - 작업 단위(UOW)가 커미트될 때, 해당 작업 단위 중에 테이블에 대해 수행된 모든 변경사항은 디스크로 이동됩니다.
  - 롤 포워드 유틸리티를 실행하는 중에 데이터베이스의 테이블이 로드 유틸리티를 통해 채워지거나 NOT LOGGED INITIALLY 옵션을 통해 작성되었음을 나타내는 로그 레코드가 표시되는 경우, 테이블은 사용 불가능으로 표시됩니다. 나중에 DROP TABLE 로그를 만나게 되면 롤 포워드 유틸리티는 해당 테이블을 삭제합니다. 그렇지 않으면, 데이터베이스가 복구된 후, 테이블에 액세스하려고 하면 오류가 표시됩니다(SQLSTATE 55019). 유일하게 허용되는 조작은 테이블 삭제입니다.
  - 이러한 테이블이 데이터베이스나 테이블 스페이스 백업의 일부로 백업되면, 테이블을 복구할 수 있습니다.
- ENABLE QUERY OPTIMIZATION으로 정의된 REFRESH DEFERRED 시스템 유지보수 구체화된 쿼리 테이블은 CURRENT REFRESH AGE가 ANY로 설정되고 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION이 시스템 유지보수 구체화된 쿼리 테이블을 포함하여 설정된 경우, 쿼리 처리를 최적화하는 데 사용할 수 있습니다. ENABLE QUERY OPTIMIZATION으로 정의된 REFRESH DEFERRED 사용자 유지보수 구체화된 쿼리 테이블은 CURRENT REFRESH AGE가 ANY로 설정되고 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION이 사용자 유지보수 구체화된 쿼리 테이블을 포함하여 설정된 경우, 쿼리 처리를 최적화하는 데 사용할 수 있습니다. ENABLE QUERY OPTIMIZATION으로 정의된 REFRESH IMMEDIATE 구체화된 쿼리 테이블은 항상 최적화를 위해 고려됩니다. 최적화를 위해 REFRESH DEFERRED 또는 REFRESH IMMEDIATE 구체화된 쿼리 테이블을 사용할 수 있도록 하려면, fullselect가 이미 설명된 규칙 외에도 특정 규칙을 따라야 합니다. 이 fullselect는 다음과 같아야 합니다.
  - GROUP BY절을 가진 subselect이거나 단일 테이블 참조를 가진 subselect여야 합니다.
  - 선택 목록 어디에도 DISTINCT가 포함되면 안됩니다.
  - 특수 레지스터의 값에 따라 다른 특수 레지스터 및 내장 함수를 포함하지 않습니다.

  - 전역 변수를 포함해서는 안됩니다.
  - 비결정적 함수를 포함해서는 안됩니다.

  구체화된 쿼리 테이블 작성시, 지정된 쿼리가 이러한 규칙을 지키지 않으면 경고가 리턴됩니다(SQLSTATE 01633).
- 구체화된 쿼리 테이블이 REFRESH IMMEDIATE로 정의되는 경우, 하위 테이블을 삽입, 갱신 또는 삭제 작업한 결과 생긴 변경을 적용하려고 시도할 때 오류가 발생할 수도 있습니다. 이 오류로 인해 하위 테이블의 삽입, 갱신 또는 삭제 조작이 실패하게 됩니다.
- 로드 조작 또는 SET INTEGRITY문의 실행과 같이 제한조건이 대량으로 점검될 경우 구체화된 쿼리 테이블 또는 스테이징 테이블을 예외 테이블로 사용할 수 없습니다.
- REFRESH IMMEDIATE로 정의되거나 연관된 스테이징 테이블과 같이 REFRESH DEFERRED로 정의된 구체화된 쿼리 테이블에 의해 참조되는 테이블에 대해 특정 조작을 수행할 수 없습니다.
  - IMPORT REPLACE를 사용할 수 없습니다.
  - ALTER TABLE NOT LOGGED INITIALLY WITH EMPTY TABLE을 수행할 수 없습니다.
- 페더레이티드 시스템에서 관계형 데이터 소스 또는 로컬 테이블에 대한 별칭은 구체화된 쿼리 테이블을 작성하기 위한 하위 테이블로 사용할 수 있습니다. 관계형이 아닌 데이터 소스에 대한 별칭은 지원되지 않습니다. 별칭이 하위 테이블 중의 하나일 때 REFRESH DEFERRED 옵션을 사용해야 합니다. 별칭을 참조하는 시스템 유지보수 구체화된 쿼리 테이블은 파티션된 데이터베이스 환경에서 지원되지 않습니다.
- ***투명한 DDL:*** 페더레이티드 시스템에서는 DB2 SQL을 사용하여 리모트 기본 테이블을 작성, 변경 또는 삭제할 수 있습니다. 이 기능을 *투명한 DDL*이라고 합니다. 리모트 기본 테이블을 데이터 소스에 작성하려면, 해당 데이터 소스에 액세스할 수 있도록 페더레이티드 서버를 구성해야 합니다. 이 구성에는 데이터 소스의 랩퍼 작성, 리모트 기본 테이블이 있는 서버에 대한 서버 정의 제공 및 페더레이티드 서버와 데이터 소스 간의 사용자 맵핑 작성이 포함됩니다.

  투명한 DDL은 CREATE TABLE문에 포함될 수 있는 내용에 대해 몇 가지 제한사항이 있습니다.
  - 컬럼과 기본 키만 리모트 기본 테이블에 작성될 수 있습니다.
  - 투명 DDL이 지원하는 특정 절은 다음을 포함합니다.
    - *element-list*절의 *column-definition* 및 *unique-constraint*
    - *column-options*절의 NOT NULL 및 PRIMARY KEY
    - OPTIONS
  - 리모트 데이터 소스는 다음을 지원해야 합니다.

- DB2 컬럼 데이터 유형이 맵핑된 리모트 컬럼 데이터 유형
- CREATE TABLE문의 기본 키 옵션

데이터 소스가 지원되지 않는 요청에 응답하는 방식에 따라, 오류가 리턴되거나 요청이 무시될 수 있습니다.

투명한 DDL을 사용하여 리모트 기본 테이블을 작성할 경우, 별칭이 자동적으로 해당 리모트 기본 테이블에 대해 작성됩니다.

- 참조 제한조건은 상위 테이블이나 종속 테이블이 상위 계층의 일부가 되는 방법으로 정의할 수 있습니다. 이러한 경우, 참조 제한조건의 영향은 다음과 같습니다.

  1. INSERT, UPDATE 및 DELETE문의 영향:
     - 참조 제한조건이 있는 경우 PT가 상위 테이블이고 DT가 종속 테이블이라면, 이 제한조건은 널(NULL)이 아닌 외부 키를 갖는 DT(또는 하위 테이블)의 각 행에 대해 일치하는 상위 키를 갖는 행이 PT에 있는지 확인합니다. 이 규칙은 조치 시작 방법에 관계없이 PT 또는 DT의 행에 영향을 주는 모든 조치에 대해 시행됩니다.
  2. DROP TABLE문의 영향:
     - 삭제된 테이블이 상위 테이블이거나 종속 테이블인 참조 제한조건의 경우에는 해당 제한조건이 삭제됩니다.
     - 삭제된 테이블의 슈퍼 테이블이 상위 테이블인 참조 제한조건의 경우에는 삭제된 테이블의 행을 슈퍼 테이블에서 삭제된 것으로 간주됩니다. 참조 제한조건이 검사되고 삭제된 행 각각에 대해 삭제 규칙이 호출됩니다.
     - 삭제된 테이블의 슈퍼 테이블이 종속 테이블인 참조 제한조건의 경우에는 해당 제한조건이 검사되지 않습니다. 종속 테이블에서 행을 삭제해도 참조 제한조건 위반이 발생할 수 없습니다.

- ***특권:*** 테이블이 작성될 때 테이블 정의자에게는 CONTROL 특권이 부여됩니다. 서브테이블이 작성될 때, 각 사용자나 그룹이 중간 슈퍼 테이블에 대해 가지고 있는 SELECT 특권이 권한을 부여한 사용자인 테이블 정의자의 해당 서브테이블에 자동으로 부여됩니다.
- ***행 크기 제한:*** 테이블의 행에 허용되는 최대 바이트 수는 테이블이 작성된 테이블 스페이스의 페이지 *크기*에 따라 다릅니다(*tablspace-name1*). 다음 목록은 각 테이블 스페이스 페이지 *크기*와 연관된 행 *크기* 한계 및 컬럼 한계 수를 나타냅니다.

표 20. 각 테이블 스페이스 페이지 크기에 대한 컬럼 수 및 행 크기 한계

페이지 *크기*	행 *크기* 한계	컬럼 수 한계
4K	4,005	500
8K	8,101	1,012
16K	16,293	1,012
32K	32,677	1,012

테이블에 대한 실제 컬럼 수는 다음 공식에 의해 더 제한할 수 있습니다.

```
총 컬럼 수 * 8 + LOB 컬럼 수 * 12 <=
    페이지 크기에 대한 행 크기 한계
```

- ***바이트 수:*** 다음 표에는 데이터 유형별 컬럼 바이트 수가 포함됩니다. 이것은 행 크기를 계산할 때 사용됩니다. 바이트 수는 VALUE COMPRESSION의 활성 여부에 따라 달라집니다. VALUE COMPRESSION이 활성화되지 않은 경우 바이트 수는 컬럼의 널(NULL) 입력 가능 여부에도 좌우됩니다.

  구조화된 유형에 기반한 테이블의 경우, 서브테이블이 정의되어 있는지에 관계없이 서브테이블 행을 식별하기 위해 4바이트의 추가 오버헤드가 예약되어 있습니다. 추가 서브테이블 컬럼은 널(NULL) 입력 불가능으로 정의되어 있는 경우라도, 바이트 계산상 널(NULL) 입력이 가능한 것으로 간주해야 합니다.

표 21. 데이터 유형별 컬럼의 바이트 수

	VALUE COMPRESSION 활성[1]	**VALUE COMPRESSION 비활성**	
데이터 유형		**컬럼 널(NULL) 입력 가능**	**컬럼 널(NULL) 입력 불가능**
SMALLINT	4	3	2
INTEGER	6	5	4
BIGINT	10	9	8
REAL	6	5	4
DOUBLE	10	9	8
DECIMAL	(*p*/2)+3의 정수 부분. 여기서 *p*는 정밀도입니다.	(*p*/2)+2의 정수 부분. 여기서 *p*는 정밀도입니다.	(*p*/2)+1의 정수 부분. 여기서 *p*는 정밀도입니다.
DECFLOAT(16)	10	9	8
DECFLOAT(34)	18	17	16
CHAR(*n*)	*n*+2	*n*+1	*n*
VARCHAR(*n*)	*n*+2	*n*+5 (테이블 내)	*n*+4 (테이블 내)
LONG VARCHAR[2]	22	25	24
GRAPHIC(*n*)	*n**2+2	*n**2+1	*n**2
VARGRAPHIC(*n*)	*n**2+2	*n**2+5 (테이블 내)	*n**2+4 (테이블 내)
LONG VARGRAPHIC[2]	22	25	24
DATE	6	5	4
TIME	5	4	3
TIMESTAMP(*p*)	(*p*+1)/2+9의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.	(*p*+1)/2+8의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.	(*p*+1)/2+7의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.
XML (INLINE LENGTH을 지정하지 않은 경우)	82	85	84
XML(INLINE LENGTH를 지정한 경우)	INLINE LENGTH +2	INLINE LENGTH +4	INLINE LENGTH +3
최대 LOB[3] 길이 1024 (INLINE LENGTH를 지정하지 않은 경우)	70	73	72
최대 LOB 길이 8192(INLINE LENGTH를 지정하지 않은 경우)	94	97	96

표 21. 데이터 유형별 컬럼의 바이트 수 (계속)

데이터 유형	VALUE COMPRESSION 활성[1]	VALUE COMPRESSION 비활성	
		컬럼 널(NULL) 입력 가능	컬럼 널(NULL) 입력 불가능
최대 LOB 길이 65 536 (INLINE LENGTH를 지정하지 않은 경우)	118	121	120
최대 LOB 길이 524 000 (INLINE LENGTH를 지정하지 않은 경우)	142	145	144
최대 LOB 길이 4 190 000 (INLINE LENGTH를 지정하지 않은 경우)	166	169	168
최대 LOB 길이 134 000 000 (INLINE LENGTH를 지정하지 않은 경우)	198	201	200
최대 LOB 길이 536 000 000 (INLINE LENGTH를 지정하지 않은 경우)	222	225	224
최대 LOB 길이 1 070 000 000(INLINE LENGTH를 지정하지 않은 경우)	254	257	256
최대 LOB 길이 1 470 000 000(INLINE LENGTH를 지정하지 않은 경우)	278	281	280
최대 LOB 길이 2 147 483 647(INLINE LENGTH를 지정하지 않은 경우)	314	317	316
INLINE LENGTH가 지정된 LOB	INLINE LENGTH + 2	INLINE LENGTH + 5	INLINE LENGTH + 4

[1] 해당 행에 대해 VALUE COMPRESSION이 활성화된 경우 각 행에 의해 사용되는 추가의 2바이트 스토리지가 있습니다.

[2]LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형이 지원되지만 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다.

[3] 각 LOB 값에는 실제 값의 위치를 지시하는 기본 레코드의 *LOB 디스크립터*가 있습니다. 디스크립터의 *크기*는 컬럼에 대해 정의된 최대 길이에 따라 다릅니다.

*구별 유형*의 경우, 바이트 수는 구별 유형에 대한 소스 유형의 길이와 동등합니다. *참조 유형*의 경우, 바이트 수는 참조 유형의 기반이 되는 내장 데이터 유형의 길이와 같습니다. *구조화된 유형*의 경우, 바이트 수는 INLINE LENGTH + 4와 같습니다. INLINE LENGTH는 *column-options*절의 컬럼에 대해 지정된(또는 내재적으로 계산된) 값입니다.

다음 샘플 테이블의 행 *크기*에서는 VALUE COMPRESSION이 지정되지 않았다고 간주합니다.

```
DEPARTMENT 63 (0 + 3 + 33 + 7 +  3 + 17)
ORG        57 (0 + 3 + 19 + 2 + 15 + 18)
```

VALUE COMPRESSION이 지정되었다면 행 *크기*는 다음과 같이 변경됩니다.

```
DEPARTMENT 69 (2 + 5 + 31 + 8 +  5 + 18)
ORG        53 (2 + 4 + 16 + 4 + 12 + 15)
```

- ***스토리지 바이트 수:*** 다음 표에는 데이터 값에 대한 데이터 유형별 컬럼의 스토리지 바이트 수가 포함됩니다. 바이트 수는 VALUE COMPRESSION의 활성 여부에 따라 달라집니다. VALUE COMPRESSION이 활성화되지 않은 경우 바이트 수는 컬럼의 널(NULL) 입력 가능 여부에도 좌우됩니다. 테이블의 값은 값을 저장하는 데 사용되는 스토리지 양(바이트)을 나타냅니다.

표 22. 행 형식, 데이터 유형 및 데이터 값에 기초한 스토리지 바이트 수

데이터 값 →	NULL	NULL	0길이	시스템 디폴트[2]	모든 기타 데이터 값	모든 기타 데이터 값	모든 기타 데이터 값
VALUE COMPRESSION →	비활성	활성[1]	활성[1]	활성[1]	비활성	비활성	활성[1]
컬럼 널(null) 가능성 →	널(NULL) 입력 가능	널(NULL) 입력 가능	n/a	n/a	널(NULL) 입력 가능	널(NULL) 입력 불가능	n/a
데이터 유형							
SMALLINT	3	3	-	3	3	2	4
INTEGER	5	3	-	3	5	4	6
BIGINT	9	3	-	3	9	8	10
REAL	5	3	-	3	5	4	6
DOUBLE	9	3	-	3	9	8	10
DECIMAL	(*p*/2)+2의 정수 부분. 여기서 *p*는 정밀도입니다.	3	-	3	(*p*/2)+2의 정수 부분. 여기서 *p*는 정밀도입니다.	(*p*/2)+1의 정수 부분. 여기서 *p*는 정밀도입니다.	(*p*/2)+3의 정수 부분. 여기서 *p*는 정밀도입니다.
DECFLOAT(16)	9	3	-	3	9	8	10
DECFLOAT(34)	17	3	-	3	17	16	18
CHAR(*n*)	*n*+1	3	-	3	*n*+1	*n*	*n*+2
VARCHAR(*n*)	5	3	2	2	*N*+5, 여기서 *N*은 데이터의 바이트 수입니다.	*N*+4, 여기서 *N*은 데이터의 바이트 수입니다.	*N*+2, 여기서 *N*은 데이터의 바이트 수입니다.
LONG VARCHAR[3]	5	3	2	2	25	24	22
GRAPHIC(*n*)	*n**2+1	3	-	3	*n**2+1	*n**2	*n**2+2
VARGRAPHIC(*n*)	5	3	2	2	*N**2+5, 여기서 *N*은 데이터의 바이트 수입니다.	*N**2+4, 여기서 *N*은 데이터의 바이트 수입니다.	*N**2+2, 여기서 *N*은 데이터의 바이트 수입니다.
LONG VARGRAPHIC[3]	5	3	2	2	25	24	22
DATE	5	3	-	-	5	4	6
TIME	4	3	-	-	4	3	5
TIMESTAMP(*p*)	(*p*+1)/2+8의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.	3	-	-	(*p*+1)/2+8의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.	(*p*+1)/2+7의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.	(*p*+1)/2+9의 정수 부분. 여기서 *p*는 소수 초의 정밀도입니다.
최대 LOB[2] 길이 1024	5	3	2	2	(60 - 68)+5	(60 - 68)+4	(60 - 68)+2
최대 LOB 길이 8192	5	3	2	2	(60 - 92)+5	(60 - 92)+4	(60 - 92)+2

표 22. 행 형식, 데이터 유형 및 데이터 값에 기초한 스토리지 바이트 수 (계속)

데이터 값 →	NULL	NULL	0길이	시스템 디폴트[2]	모든 기타 데이터 값	모든 기타 데이터 값	모든 기타 데이터 값
VALUE COMPRESSION →	비활성	활성[1]	활성[1]	활성[1]	비활성	비활성	활성[1]
컬럼 널(null) 가능성 →	널(NULL) 입력 가능	널(NULL) 입력 가능	n/a	n/a	널(NULL) 입력 가능	널(NULL) 입력 불가능	n/a
데이터 유형							
최대 LOB 길이 65 536	5	3	2	2	(60 - 116)+5	(60 - 116)+4	(60 - 116)+2
최대 LOB 길이 524 000	5	3	2	2	(60 - 140)+5	(60 - 140)+4	(60 - 140)+2
최대 LOB 길이 4 190 000	5	3	2	2	(60 - 164)+5	(60 - 164)+4	(60 - 164)+2
최대 LOB 길이 134 000 000	5	3	2	2	(60 - 196)+5	(60 - 196)+4	(60 - 196)+2
최대 LOB 길이 536 000 000	5	3	2	2	(60 - 220)+5	(60 - 220)+4	(60 - 220)+2
최대 LOB 길이 1 070 000 000	5	3	2	2	(60 - 252)+5	(60 - 252)+4	(60 - 252)+2
최대 LOB 길이 1 470 000 000	5	3	2	2	(60 - 276)+5	(60 - 276)+4	(60 - 276)+2
최대 LOB 길이 2 147 483 647	5	3	2	2	(60 - 312)+5	(60 - 312)+4	(60 - 312)+2
XML	5	3	-	-	85	84	82

[1] 해당 행에 대해 VALUE COMPRESSION이 활성화된 경우 각 행에 의해 사용되는 추가의 2바이트 스토리지가 있습니다.

[2] COMPRESS SYSTEM DEFAULT가 컬럼에 지정된 경우

[3]LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형이 지원되지만 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다.

- *차원 컬럼:* 차원 컬럼의 각 구별 값은 테이블의 서로 다른 블록에 지정되므로 ″INTEGER(ORDER_DATE)/100″과 같이 표현식으로 클러스터링하는 것이 바람직합니다. 이 경우 생성된 컬럼을 테이블에 대해 정의할 수 있으며 그러면 이 컬럼을 ORGANIZE BY DIMENSIONS절에 사용할 수 있습니다. 표현식이 테이블의 컬럼에 대해 단조적인 경우, DB2가 차원 인덱스를 사용하여 해당 컬럼에 대한 범위 술어를 충족시킬 수 있습니다. 예를 들어 *column-name + some-positive-constant*와 같은 단순 표현식은 단조적 증가 표현식입니다. 표현식에 사용자 정의 함수(UDF)나 특정 내장 함수 또는 여러 개의 컬럼을 사용하여 단조로움을 줄이거나 단조로움이 발견되지 않도록 할 수 있습니다.

  생성된 컬럼 중 컬럼의 표현식이 단조롭지 않거나 단조로움을 판별할 수 없는 컬럼과 관련된 차원을 작성할 수는 있지만 이런 차원의 슬라이스나 셀 바운더리에 대한 범위 쿼리는 지원되지 않습니다. 등호와 IN 술어는 슬라이스나 셀이 *처리할 수 있습니다.*

  생성 함수 fn과 관련하여 다음에 해당될 경우에는 생성된 컬럼이 단조롭습니다.

– 단조로운 증가

  가능한 모든 x1과 x2 값 쌍의 경우, x2>x1이면 fn(x2)>fn(x1). 예를 들면, 다음과 같습니다.

```
SALARY - 10000
```

– 단조로운 감소

  가능한 모든 x1과 x2 값 쌍의 경우, x2>x1이면 fn(x2)<fn(x1). 예를 들면, 다음과 같습니다.

```
-SALARY
```

– 단조로운 비감소

  가능한 x1과 x2 값 쌍의 경우, x2>x1이면 fn(x2)>=fn(x1). 예를 들면, 다음과 같습니다.

```
SALARY/1000
```

– 단조로운 비증가

  가능한 x1과 x2 값 쌍의 경우, x2>x1이면 fn(x2)<=fn(x1). 예를 들면, 다음과 같습니다.

```
-SALARY/1000
```

표현식 ″PRICE*DISCOUNT″에는 테이블의 여러 행이 관련되므로 단조롭지 않습니다.

- ***범위 클러스터 테이블:*** 키 시퀀스에 의한 테이블 구성은 특정 유형의 테이블에 효과적입니다. 테이블에는 가능한 값 범위를 초과하여 조밀하게 클러스터된 정수 키가 있어야 합니다. 이 정수 키의 컬럼에는 널(NULL) 입력이 불가능해야 하며 이 키는 논리적으로 테이블의 기본 키가 되어야 합니다. 범위 클러스터 테이블 구성을 사용하면 지정한 키 값에 해당하는 행 또는 지정한 키 값 범위에 해당하는 행 범위에 대한 직접 액세스를 제공하기 때문에 별도의 고유 인덱스 오브젝트가 필요하지 않습니다. 정의된 키 시퀀스 범위에 있는 전체 행에 필요한 스페이스가 테이블 작성시 모두 할당되며 범위 클러스터 테이블을 정의하는 경우에도 모두 할당되어야 합니다. 삭제되도록 초기에 표시된 행이라 하더라도 해당 스토리지 스페이스를 다른 용도로 사용할 수 없습니다. 전체 키 시퀀스 범위를 장시간에 걸쳐 데이터로 채우는 경우 이 테이블 구성은 올바른 선택이 아닐 수도 있습니다.
- 테이블에는 많아야 하나의 보안 규정이 있을 수 있습니다.
- DB2는 보호된 테이블에 정의된 참조 무결성 제한조건을 강제 실행합니다. 이 경우, 사용자가 적절한 보안 레이블 또는 면제 증명서를 가지고 있지 않으면 DB2는 어떤 행이 위반을 일으켰는지 알려주지 않으므로 제한조건 위반을 디버그하기 어려울 수 있습니다.

- 테이블의 컬럼 순서를 지정할 때, 자주 갱신되는 컬럼은 정의 끝에 배치되어야 갱신하기 위해 로그되는 데이터 양이 최소화됩니다. ROW CHANGE TIMESTAMP 컬럼이 포함됩니다. ROW CHANGE TIMESTAMP 컬럼은 각 행 갱신 시 갱신됩니다.
- ***보안 및 복제:*** 복제 조작은 데이터 행이 보호된 테이블로부터 데이터베이스 밖으로 복제되도록 할 수 있습니다. DB2는 데이터베이스 밖의 데이터를 보호할 수 없으므로 보호된 테이블의 복제를 설정할 때는 조심해야 합니다.
- ***호환성***: z/OS용 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 다음 구문이 디폴트 동작으로 허용됩니다.
    - IN database-name.tablespace-name
    - IN DATABASE database-name
    - FOR MIXED DATA
    - FOR SBCS DATA
  - P ARTITION 대신 PART 지정 가능
  - PARTITION *partition-name*을 대신해서 PARTITION *partition-number*를 지정할 수 있습니다. *partition-number*는 이전에 CREATE TABLE문에 지정된 파티션을 식별해서는 안됩니다. *partition-number*를 지정하지 않으면 데이터베이스 관리 프로그램이 고유한 파티션 번호를 생성합니다.
  - ENDING AT 대신 VALUES 지정 가능

  이전 버전의 DB2 데이터베이스와의 호환성을 위해 다음과 같이 수행할 수 있습니다.
  - references-clause를 정의하는 *column-definition*에서 CONSTRAINT 키워드를 생략할 수 있습니다.
  - *constraint-name*을 FOREIGN KEY 다음에 지정할 수 있습니다(CONSTRAINT 키워드없이).
  - CREATE 다음에 SUMMARY를 선택적으로 지정할 수 있습니다.
  - WITH NO DATA 대신 DEFINITION ONLY를 지정할 수 있습니다.
  - DISTRIBUTE BY절 대신 PARTITIONING KEY절 지정 가능
  - DISTRIBUTE BY REPLICATION 대신 REPLICATED를 지정할 수 있습니다.

  이전 버전의 DB2 데이터베이스의 호환성 및 일관성을 위해 다음과 같이 수행할 수 있습니다.
  - *identity-options*절에서 여러 옵션을 구분하는 데 쉼표를 사용할 수 있습니다.

  다음 구문도 지원됩니다.
  - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE 및 NOORDER

예:

예 1: DEPARTX 테이블 스페이스에 TDEPT 테이블을 작성하십시오. DEPTNO, DEPTNAME, MGRNO 및 ADMRDEPT는 컬럼 이름입니다. CHAR은 컬럼에 문자 데이터를 넣을 수 있음을 의미합니다. NOT NULL은 컬럼에 널(NULL) 값을 넣을 수 없음을 의미합니다. VARCHAR은 컬럼에 가변 길이 문자 데이터를 넣을 수 있음을 의미합니다. 기본 키는 DEPTNO 컬럼으로 구성됩니다.

```
CREATE TABLE TDEPT
       (DEPTNO   CHAR(3)     NOT NULL,
         DEPTNAME   VARCHAR(36)   NOT NULL,
         MGRNO      CHAR(6),
         ADMRDEPT CHAR(3)     NOT NULL,
   PRIMARY KEY(DEPTNO))
IN DEPARTX
```

예 2: SHED 테이블 스페이스에 PROJ 테이블을 작성하십시오. PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTAFF, PRSTDATE, PRENDATE 및 MAJPROJ는 컬럼 이름입니다. CHAR은 컬럼에 문자 데이터를 넣을 수 있음을 의미합니다. DECIMAL은 컬럼에 압축 10진수 데이터를 넣을 수 있음을 의미합니다. 5는 10진수 자릿수를 나타내고, 2는 소수점 오른쪽의 자릿수를 나타냅니다. NOT NULL은 컬럼에 널(NULL) 값을 넣을 수 없음을 의미합니다. VARCHAR은 컬럼에 가변 길이 문자 데이터를 넣을 수 있음을 의미합니다. DATE는 컬럼에 세 부분 형식(년, 월, 일)으로 되어 있는 날짜 정보를 넣을 수 있음을 의미합니다.

```
CREATE TABLE PROJ
       (PROJNO   CHAR(6)      NOT NULL,
         PROJNAME VARCHAR(24)  NOT NULL,
         DEPTNO   CHAR(3)      NOT NULL,
         RESPEMP  CHAR(6)      NOT NULL,
         PRSTAFF  DECIMAL(5,2)         ,
         PRSTDATE DATE                 ,
         PRENDATE DATE                 ,
         MAJPROJ  CHAR(6)      NOT NULL)
IN SCHED
```

예 3: 알 수 없는 급여는 0으로 간주하는 EMPLOYEE_SALARY 테이블을 작성하십시오. 테이블 스페이스를 지정하지 않았으므로, 테이블은 *IN tablespace-name*절에 대해 기술된 규칙에 따라 시스템이 선택한 테이블 스페이스에 작성됩니다.

```
CREATE TABLE EMPLOYEE_SALARY
       (DEPTNO   CHAR(3)     NOT NULL,
         DEPTNAME   VARCHAR(36)   NOT NULL,
         EMPNO    CHAR(6)      NOT NULL,
         SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT)
```

예 4: 총 급여와 마일에 대한 구별 유형을 작성하여 이를 디폴트 테이블 스페이스에 작성된 테이블 컬럼에 사용하십시오. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 JOHNDOE이고 CURRENT PATH는 디폴트값(″SYSIBM″, ″SYSFUN″,″JOHNDOE″)이라고 가정하십시오.

SALARY값이 지정되어 있지 않으면 0으로 설정해야 하며, LIVING_DIST 값이 지정되어 있지 않으면 1마일로 설정해야 합니다.

```
CREATE TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS

CREATE TYPE JOHNDOE.MILES  AS FLOAT WITH COMPARISONS

CREATE TABLE EMPLOYEE
       (ID          INTEGER NOT NULL,
         NAME        CHAR (30),
         SALARY      T_SALARY NOT NULL WITH DEFAULT,
   LIVING_DIST MILES    DEFAULT MILES(1) )
```

예 5: 이미지와 오디오에 대한 구별 유형을 작성하여 이를 테이블 컬럼에 사용하십시오. 테이블 스페이스를 지정하지 않았으므로 테이블은 IN *tablespace-name* 절에 대해 기술된 규칙에 따라 시스템이 선택한 테이블 스페이스에 작성됩니다. CURRENT PATH가 디폴트값이라고 가정합니다.

```
CREATE TYPE IMAGE AS BLOB (10M)

CREATE TYPE AUDIO AS BLOB (1G)

CREATE TABLE PERSON
       (SSN    INTEGER NOT NULL,
         NAME        CHAR (30),
         VOICE  AUDIO,
         PHOTO  IMAGE)
```

예 6: HUMRES 테이블 스페이스에 EMPLOYEE 테이블을 작성하십시오. 테이블에 정의된 제한조건은 다음과 같습니다.

- 부서 번호의 값은 10 - 100 범위 사이여야 합니다.
- 사원의 직위는 'Sales', 'Mgr' 또는 'Clerk' 중 하나일 수 있습니다.
- 1986년 이후에 입사한 모든 사원은 $40,500 이상을 받아야 합니다.

주: 점검 제한조건에 포함된 컬럼들이 널(NULL) 입력 가능할 경우, 이 컬럼들은 널(NULL)일 수 있습니다.

```
CREATE TABLE EMPLOYEE
       (ID          SMALLINT NOT NULL,
         NAME        VARCHAR(9),
         DEPT        SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
         JOB         CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
         HIREDATE    DATE,
         SALARY      DECIMAL(7,2),
         COMM        DECIMAL(7,2),
   PRIMARY KEY (ID),
   CONSTRAINT  YEARSAL CHECK (YEAR(HIREDATE) > 1986
     OR SALARY > 40500)
  )
  IN HUMRES
```

예 7: PAYROLL 테이블 스페이스에 모두 포함되는 테이블을 작성하십시오.

```
CREATE TABLE EMPLOYEE .....
  IN PAYROLL
```

예 8: 데이터 부분은 ACCOUNTING에 포함되고 인덱스 부분은 ACCOUNT_IDX에 포함되는 테이블을 작성하십시오.

```
CREATE TABLE SALARY.....
  IN ACCOUNTING INDEX IN ACCOUNT_IDX
```

예 9: 디폴트 형식으로 테이블을 작성하고 SQL 변경사항을 기록하십시오.

```
CREATE TABLE SALARY1 .....
```

또는

```
CREATE TABLE SALARY1 .....
  DATA CAPTURE NONE
```

예 10: 확장 형식으로 테이블을 작성하고 SQL 변경사항을 기록하십시오.

```
CREATE TABLE SALARY2 .....
  DATA CAPTURE CHANGES
```

예 11: SCHED 테이블 스페이스에 EMP_ACT 테이블을 작성하십시오. EMPNO, PROJNO, ACTNO, EMPTIME, EMSTDATE 및 EMENDATE는 컬럼 이름입니다. 테이블에 정의된 제한조건은 다음과 같습니다.

- 모든 행에 있는 컬럼 세트(EMPNO, PROJNO 및 ACTNO)에 대한 값은 고유해야 합니다.
- PROJNO 값은 PROJECT 테이블의 PROJNO 컬럼에 대한 기존 값과 일치해야 하며, 프로젝트가 삭제된 경우 EMP_ACT에서 그 프로젝트를 참조하는 모든 행도 삭제되어야 합니다.

```
CREATE TABLE EMP_ACT
       (EMPNO       CHAR(6) NOT NULL,
        PROJNO      CHAR(6) NOT NULL,
        ACTNO       SMALLINT NOT NULL,
        EMPTIME     DECIMAL(5,2),
        EMSTDATE    DATE,
        EMENDATE    DATE,
   CONSTRAINT EMP_ACT_UNIQ UNIQUE (EMPNO,PROJNO,ACTNO),
   CONSTRAINT FK_ACT_PROJ FOREIGN KEY (PROJNO)
      REFERENCES PROJECT (PROJNO) ON DELETE CASCADE
  )
  IN SCHED
```

고유 인덱스 EMP_ACT_UNIQ는 동일한 스키마에 자동으로 작성되어 고유 제한조건을 적용합니다.

예 12: 아이스하키 영예 전당의 유명한 골에 대한 정보를 보유하는 테이블을 작성하십시오. 이 테이블에는 골 득점한 선수, 득점 시 골키퍼, 날짜 및 설명에 대한 정보가 나열될 것입니다. 설명 컬럼은 널(NULL) 입력 가능합니다.

```
CREATE TABLE HOCKEY_GOALS
  ( BY_PLAYER      VARCHAR(30)   NOT NULL,
    BY_TEAM        VARCHAR(30)   NOT NULL,
          AGAINST_PLAYER VARCHAR(30)   NOT NULL,
          AGAINST_TEAM   VARCHAR(30)   NOT NULL,
    DATE_OF_GOAL   DATE          NOT NULL,
    DESCRIPTION    CLOB(5000) )
```

예 13: EMPLOYEE 테이블에 예외 테이블이 필요하다고 가정하십시오. 다음 명령문을 사용하여 예외 테이블을 작성하십시오.

```
CREATE TABLE EXCEPTION_EMPLOYEE AS
      (SELECT EMPLOYEE.*,
          CURRENT TIMESTAMP AS TIMESTAMP,
    CAST ('' AS CLOB(32K)) AS MSG
  FROM EMPLOYEE
  ) WITH NO DATA
```

예 14: 표시된 속성을 갖는 다음과 같은 테이블 스페이스가 제공됩니다.

```
TBSPACE            PAGESIZE    USER   USERAUTH
------------------ ----------- ------ --------
   DEPT4K                     4096 BOBBY  Y
   PUBLIC4K                   4096 PUBLIC Y
   DEPT8K                     8192 BOBBY  Y
   DEPT8K                     8192 RICK   Y
   PUBLIC8K                   8192 PUBLIC Y
```

- RICK이 다음 테이블을 작성하는 경우 이 테이블은 바이트 수가 4005 미만이므로 테이블 스페이스 PUBLIC4K에 위치하나, BOBBY가 같은 테이블을 작성하는 경우에는 명시적 권한 부여로 인해 BOBBY가 USE 특권을 가지므로 테이블이 테이블 스페이스 DEPT4K에 위치합니다.

  ```
  CREATE TABLE DOCUMENTS
         (SUMMARY    VARCHAR(1000),
           REPORT     VARCHAR(2000))
  ```

- BOBBY가 다음 테이블을 작성하는 경우 이 테이블은 바이트 수가 4005보다 크고 명시적 권한 부여로 인해 BOBBY가 USE 특권을 가지므로 테이블이 테이블 스페이스 DEPT8K에 위치합니다. 그러나 DUNCAN이 같은 테이블을 작성하는 경우에는 DUNCAN이 특정 특권을 가지지 않으므로 테이블이 PUBLIC8K에 위치합니다.

  ```
  CREATE TABLE CURRICULUM
         (SUMMARY    VARCHAR(1000),
           REPORT     VARCHAR(2000),
           EXERCISES  VARCHAR(1500))
  ```

예 15: 구조화된 유형 EMP로 정의된 LEAD 컬럼을 갖는 테이블을 작성하십시오. LEAD 컬럼에 대해 INLINE LENGTH로 300바이트를 지정하십시오. 이는 300바이트내에 들어갈 수 없는 LEAD의 인스턴스는 테이블 외부에 저장됨을 의미합니다(LOB 값의 처리 방법과 유사하게 기본 테이블 행과는 별도로).

```
CREATE TABLE PROJECTS (PID INTEGER,
  LEAD EMP INLINE LENGTH 300,
  STARTDATE DATE,
            ...)
```

예 16: 다섯 개의 컬럼 DEPTNO, DEPTNAME, MGRNO, ADMRDEPT 및 LOCATION을 갖는 DEPT 테이블을 작성하십시오. DEPT 컬럼은 DB2가 컬럼 값을 항상 생성하도록 IDENTITY 컬럼으로 정의되어야 합니다. DEPT 컬럼의 값은 500에서 시작하여 1씩 증가해야 합니다.

```
CREATE TABLE DEPT
        (DEPTNO     SMALLINT      NOT NULL
                                      GENERATED ALWAYS AS IDENTITY
                                          (START WITH 500, INCREMENT BY 1),
          DEPTNAME   VARCHAR(36)   NOT NULL,
          MGRNO      CHAR(6),
          ADMRDEPT   SMALLINT      NOT NULL,
          LOCATION   CHAR(30))
```

예 17: YEAR 컬럼에서 분산되고 REGION 및 YEAR 컬럼에 차원이 있는 SALES 테이블을 작성하십시오. 데이터는 YEAR 컬럼의 해시된 값에 따라 데이터베이스 파티션에 분산됩니다. 각 데이터베이스 파티션에서 데이터는 REGION 및 YEAR 컬럼 값의 고유 조합에 기반한 범위로 구성됩니다.

```
CREATE TABLE SALES
       (CUSTOMER   VARCHAR(80),
         REGION     CHAR(5),
         YEAR       INTEGER)
DISTRIBUTE BY HASH (YEAR)
ORGANIZE BY DIMENSIONS (REGION, YEAR)
```

예 18: PURCHASEDATE 컬럼에서 생성된 PURCHASEYEARMONTH 컬럼을 갖는 SALES 테이블을 작성하십시오. 표현식을 사용하여 원래 PURCHASEDATE 컬럼에 대해 단조로워서 차원으로 사용하기에 적합한 컬럼을 작성하십시오. 테이블은 REGION 컬럼에서 분산되고 각 데이터베이스 파티션 내에서
PURCHASEYEARMONTH 컬럼에 따라 범위로 구성되는데, 이는 다른 영역이 다른 데이터베이스 파티션에 존재하고 다른 구입 월은 해당 데이터베이스 파티션 내에서 다른 셀(또는 범위 세트)에 속하게 됨을 의미합니다.

```
CREATE TABLE SALES
        (CUSTOMER            VARCHAR(80),
          REGION              CHAR(5),
          PURCHASEDATE        DATE,
          PURCHASEYEARMONTH   INTEGER
                                GENERATED ALWAYS AS (INTEGER(PURCHASEDATE)/100))
DISTRIBUTE BY HASH (REGION)
ORGANIZE BY DIMENSIONS (PURCHASEYEARMONTH)
```

예 19: CUSTOMERNUM 컬럼에서 생성된 CUSTOMERNUMDIM 컬럼을 갖는 CUSTOMER 테이블을 작성하십시오. 표현식을 사용하여 원래 CUSTOMERNUM 컬럼에 대해 일정하므로 차원으로 사용하기에 적합한 컬럼을 작성하십시오. 테이블은 CUSTOMERNUMDIM 컬럼에 따라서 셀로 조직화되어 50명의 고객마다 테이블에 다

른 셀이 있습니다. 고유 인덱스가 CUSTOMERNUM에서 작성된 경우, 각각의 50개의 값 세트가 테이블에서 특정 범위 세트로 발견되는 방식으로 고객 번호가 클러스터됩니다.

```
CREATE TABLE CUSTOMER
       (CUSTOMERNUM      INTEGER,
         CUSTOMERNAME     VARCHAR(80),
         ADDRESS          VARCHAR(200),
         CITY             VARCHAR(50),
         COUNTRY          VARCHAR(50),
         CODE             VARCHAR(15),
         CUSTOMERNUMDIM   INTEGER
                       GENERATED ALWAYS AS (CUSTOMERNUM/50))
ORGANIZE BY DIMENSIONS (CUSTOMERNUMDIM)
```

예 20: Oracle 서버 ORASERVER에 리모트 기본 테이블 EMPLOYEE를 작성하십시오. 새로 작성된 이 리모트 기본 테이블을 참조하는 별칭 EMPLOYEE도 자동으로 작성됩니다.

```
CREATE TABLE EMPLOYEE
       (EMP_NO CHAR(6) NOT NULL,
         FIRST_NAME   VARCHAR(12)   NOT NULL,
         MID_INT      CHAR(1)       NOT NULL,
         LAST_NAME    VARCHAR(15)   NOT NULL,
         HIRE_DATE    DATE,
         JOB          CHAR(8),
         SALARY       DECIMAL(9,2),
   PRIMARY KEY (EMP_NO))
OPTIONS
       (REMOTE_SERVER 'ORASERVER',
   REMOTE_SCHEMA 'J15USER1',
   REMOTE_TABNAME 'EMPLOYEE')
```

다음 CREATE TABLE문은 원하는 대소문자를 가져오기 위해 테이블 이름 또는 테이블 이름과 명시적 리모트 기본 테이블 이름을 지정하는 방법을 표시합니다. 소문자 ID employee를 사용하여 내재된 ID 접기를 표시합니다.

Informix® 서버에 이름이 EMPLOYEE(대문자)인 리모트 기본 테이블을 작성하고 해당 테이블에 이름이 EMPLOYEE(대문자)인 별칭을 작성하십시오.

```
CREATE TABLE employee
       (EMP_NO CHAR(6) NOT NULL,
   ...)
OPTIONS
       (REMOTE_SERVER 'INFX_SERVER')
```

REMOTE_TABNAME 옵션을 지정하지 않고 *table-name*이 분리되지 않은 경우, 리모트 데이터 소스가 일반적으로 이름을 소문자로 저장할 경우에도 리모트 기본 테이블 이름은 대문자로 작성됩니다.

Informix 서버에 이름이 employee(소문자)인 리모트 기본 테이블을 작성하고 해당 테이블에 이름이 EMPLOYEE(대문자)인 별칭을 작성하십시오.

```
CREATE TABLE employee
       (EMP_NO CHAR(6) NOT NULL,
   ...)
OPTIONS
       (REMOTE_SERVER 'INFX_SERVER',
   REMOTE_TABNAME 'employee')
```

분리 ID를 지원하는 리모트 데이터 소스에 테이블을 작성할 경우, REMOTE_TABNAME 옵션과 원하는 대소문자로 테이블 이름을 지정하는 문자열 상수를 사용하십시오.

Informix 서버에 이름이 employee(소문자)인 리모트 기본 테이블을 작성하고 해당 테이블에 이름이 employee(소문자)인 별칭을 작성하십시오.

```
CREATE TABLE "employee"
       (EMP_NO CHAR(6) NOT NULL,
   ...)
OPTIONS
       (REMOTE_SERVER 'INFX_SERVER')
```

REMOTE_TABNAME 옵션을 지정하지 않고 *table-name*이 분리된 경우, 리모트 기본 테이블 이름은 *table-name*과 동일합니다.

예 21: 학생 ID를 사용하여 학생을 찾을 수 있는 범위 클러스터 테이블을 작성하십시오. 각 학생 레코드에는 학교 ID, 프로그램 ID, 학생 번호, 학생 ID, 학생 이름, 학생 성 및 학생 평점(GPA)이 있습니다.

```
CREATE TABLE STUDENTS
       (SCHOOL_ID     INTEGER   NOT NULL,
         PROGRAM_ID    INTEGER   NOT NULL,
         STUDENT_NUM   INTEGER   NOT NULL,
         STUDENT_ID    INTEGER   NOT NULL,
         FIRST_NAME    CHAR(30),
         LAST_NAME     CHAR(30),
         GPA           DOUBLE)
ORGANIZE BY KEY SEQUENCE
       (STUDENT_ID
    STARTING FROM 1
    ENDING AT 1000000)
  DISALLOW OVERFLOW
```

각 레코드의 *크기*는 모든 컬럼, 맞추기 및 범위 클러스터 테이블 행 헤더를 더한 값입니다. 이 경우에 행 *크기*는 98바이트입니다. 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3(널(NULL) 입력 가능한 컬럼의 경우) + 1(맞추기) + 10(헤더). 4KB 페이지 *크기*(또는 4096바이트) 경우 페이지 오버헤드를 어카운팅한 후 남는 4038바이트가 사용 가능하며 페이지 당 41개 레코드를 표시하기에 충분한 *크기*입니다. 백만 명의 학생 레코드가 허용되며 24 391페이지가 필요합니다(백만 명 나누기 페이지당 41개 레코드). 테이블 오버헤드에 필요한 두 페이지를 추가하면 테이블 작성 시 할당되는 4KB 페이지의 최종 개수는 24 393입니다.

예 22: 지정된 제한조건 이름이 없는 함수적 종속성을 갖는 DEPARTMENT 테이블을 작성하십시오.

```
CREATE TABLE DEPARTMENT
  (DEPTNO     SMALLINT      NOT NULL,
         DEPTNAME   VARCHAR(36)   NOT NULL,
         MGRNO      CHAR(6),
         ADMRDEPT   SMALLINT      NOT NULL,
         LOCATION   CHAR(30),
  CHECK (DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED)
```

예 23: 보호된 행이 있는 테이블을 작성하십시오.

```
CREATE TABLE TOASTMASTERS
       (PERFORMANCE DB2SECURITYLABEL,
         POINTS      INTEGER,
         NAME        VARCHAR(50))
  SECURITY POLICY CONTRIBUTIONS
```

예 24: 보호된 컬럼이 있는 테이블을 작성하십시오.

```
CREATE TABLE TOASTMASTERS
       (PERFORMANCE CHAR(8),
         POINTS      INTEGER COLUMN SECURED WITH CLUBPOSITION,
         NAME        VARCHAR(50))
  SECURITY POLICY CONTRIBUTIONS
```

예 25: 보호된 행과 컬럼이 있는 테이블을 작성하십시오.

```
CREATE TABLE TOASTMASTERS
       (PERFORMANCE DB2SECURITYLABEL,
         POINTS      INTEGER COLUMN SECURED WITH CLUBPOSITION,
         NAME        VARCHAR(50))
  SECURITY POLICY CONTRIBUTIONS
```

예 26: 파티션된 테이블에 대한 대형 오브젝트는 데이터와 동일한 테이블 스페이스에 있는 것이 디폴트값입니다. 이 디폴트 동작은 LONG IN절을 사용하여 무시된 후 대형 오브젝트에 대한 하나 이상의 테이블 스페이스를 지정할 수 있습니다. 각 데이터 파티션에 대해 라운드 로빈 방식으로 저장될 대형 오브젝트를 가진 DOCUMENTS라는 이름의 테이블을 TBSP1 및 TBSP2 테이블 스페이스에 작성하십시오.

```
CREATE TABLE DOCUMENTS
       (ID INTEGER,
         CONTENTS CLOB)
  LONG IN TBSP1, TBSP2
  PARTITION BY RANGE (ID)
    (STARTING 1 ENDING 1000
     EVERY 100)
```

혹은, 구문의 긴 형식을 사용하여 각 데이터 파티션에 대해 대형 테이블 스페이스를 명시적으로 식별하십시오. 이 예제에서 첫 번째 데이터 파티션에 대한 CLOB 데이터는 LARGE_TBSP3에 위치하고 나머지 데이터 파티션에 대한 CLOB 데이터는 LARGE_TBSP1 및 LARGE_TBSP2 전체에 라운드 로빈 방식으로 분산됩니다.

```
CREATE TABLE DOCUMENTS
       (ID INTEGER,
         CONTENTS CLOB)
  LONG IN LARGE_TBSP1, LARGE_TBSP2
  PARTITION BY RANGE (ID)
    (STARTING 1 ENDING 100
     IN TBSP1 LONG IN LARGE_TBSP3,
     STARTING 101 ENDING 1000
     EVERY 100)
```

예 27: 두 개의 데이터 파티션이 있는 ACCESSNUMBERS라는 파티션된 테이블을 작성합니다. 행(10, NULL)은 첫 번째 파티션에 위치하고 행(NULL, 100)은 두 번째(마지막) 데이터 파티션에 위치합니다.

```
CREATE TABLE ACCESSNUMBERS
       (AREA INTEGER,
         EXCHANGE INTEGER)
  PARTITION BY RANGE (AREA NULLS LAST, EXCHANGE NULLS FIRST)
          (STARTING (1,1) ENDING (10,100),
     STARTING (11,1) ENDING (MAXVALUE,MAXVALUE))
```

두 번째 컬럼에 있는 널(NULL) 값이 먼저 정렬되므로 행(11, NULL)은 마지막 데이터 파티션(11, 1)의 하위 바운더리에서 정렬되고, 이 행을 삽입하려고 하면 오류가 리턴됩니다. 행(12, NULL)은 마지막 데이터 파티션에서 분리될 수 있습니다.

예 28: 단일 데이터 파티션이 있고 PERCENT 컬럼을 파티션하는 RATIO라는 테이블을 작성합니다.

```
CREATE TABLE RATIO
       (PERCENT INTEGER)
  PARTITION BY RANGE (PERCENT)
          (STARTING (MINVALUE) ENDING (MAXVALUE))
```

이 테이블 정의를 사용하면 PERCENT 컬럼의 정수 값을 삽입할 수 있습니다. 다음과 같이 RATIO 테이블을 정의하여 1 - 100의 정수값(1과 100 포함)을 PERCENT 컬럼에 삽입할 수 있습니다.

```
CREATE TABLE RATIO
       (PERCENT INTEGER)
  PARTITION BY RANGE (PERCENT)
    (STARTING 0 EXCLUSIVE ENDING 100 INCLUSIVE)
```

예 29: ID인 컬럼 한 개와 XML 문서를 저장하는 컬럼 한 개를 가진 MYDOCS라는 이름의 테이블을 작성하십시오.

```
CREATE TABLE MYDOCS
       (ID INTEGER,
         DOC XML)
IN HLTBSPACE
```

예 30: XML 기반 주를 저장하는 컬럼을 포함하는 네 개의 컬럼을 가진 NOTES라는 이름의 테이블을 작성하십시오.

```
CREATE TABLE NOTES
       (ID          INTEGER,
         DESCRIPTION VARCHAR(255),
         CREATED     TIMESTAMP,
         NOTE        XML)
```

예 31: 각 직원의 전화번호와 주소를 포함하는 테이블, EMP_INFO를 작성합니다. ROW CHANGE TIMESTAMP 컬럼을 테이블에 포함시켜 직원 정보의 수정 내용을 추적합니다.

```
CREATE TABLE EMP_INFO
       (EMPNO       CHAR(6) NOT NULL,
   EMP_INFOCHANGE  NOT NULL GENERATED ALWAYS
    FOR EACH ROW ON UPDATE
    AS ROW CHANGE TIMESTAMP,
   EMP_ADDRESS     VARCHAR(300),
   EMP_PHONENO     CHAR(4),
   PRIMARY KEY (EMPNO) )
```

예 32: 두 개의 데이터 파티션이 있는 DOCUMENTS라는 파티션된 테이블을 작성합니다.

- 첫 번째 파티션에 있는 데이터 오브젝트가 테이블 스페이스 TBSP11에 상주합니다. 파티션에서 파티션된 인덱스 파티션은 테이블 스페이스 TBSP21에 상주합니다. XML 데이터 오브젝트는 테이블 스페이스 TBSP31에 상주합니다.
- 두 번째 파티션에 있는 데이터 오브젝트가 테이블 스페이스 TBSP12에 상주합니다. 파티션에서 파티션된 인덱스 파티션은 테이블 스페이스 TBSP22에 상주합니다. XML 데이터 오브젝트는 테이블 스페이스 TBSP32에 상주합니다.

테이블 레벨 INDEX IN절은 파티션된 인덱스에 대하여 테이블 스페이스 선택에 영향을 주지 않습니다.

```
CREATE TABLE DOCUMENTS
     (ID          INTEGER,
 CONTENTS  XML)  INDEX IN TBSPX
PARTITION BY (ID)
(STARTING 1 ENDING 100 IN TBSP11 INDEX IN TBSP21 LONG IN TBSP31,
 STARTING 101 ENDING 101 IN TBSP21 INDEX IN TBSP22 LONG IN TBSP32 );
```

예 33: 두 개의 데이터 파티션이 있는 SALES라는 파티션된 테이블을 작성합니다.

- 첫 번째 파티션에 있는 데이터 오브젝트가 테이블 스페이스 TBSP11에 상주합니다. 파티션에서 파티션된 인덱스 파티션은 테이블 스페이스 TBSP21에 상주합니다.
- 두 번째 파티션에 있는 데이터 오브젝트가 테이블 스페이스 TBSP12에 상주합니다. 파티션된 인덱스 오브젝트는 테이블 스페이스 TBSP22에 상주합니다.

테이블 레벨 INDEX IN절은 파티션된 인덱스에 대하여 테이블 스페이스 선택에 영향을 주지 않습니다.

```
CREATE TABLE SALES
(SID     INTEGER,
 AMOUNT  INTEGER)  INDEX IN TBSPX
```

```
PARTITION BY (SID)
(STARTING 1 ENDING 100 IN TBSP11 INDEX IN TBSP21,
 STARTING 101 ENDING 101 IN TBSP12 INDEX IN TBSP22);
```

CREATE TABLESPACE

CREATE TABLESPACE문은 데이터베이스 내에 새 테이블 스페이스를 정의하고 테이블 스페이스에 컨테이너를 지정하며, 테이블 스페이스 정의와 속성을 카탈로그에 기록합니다.

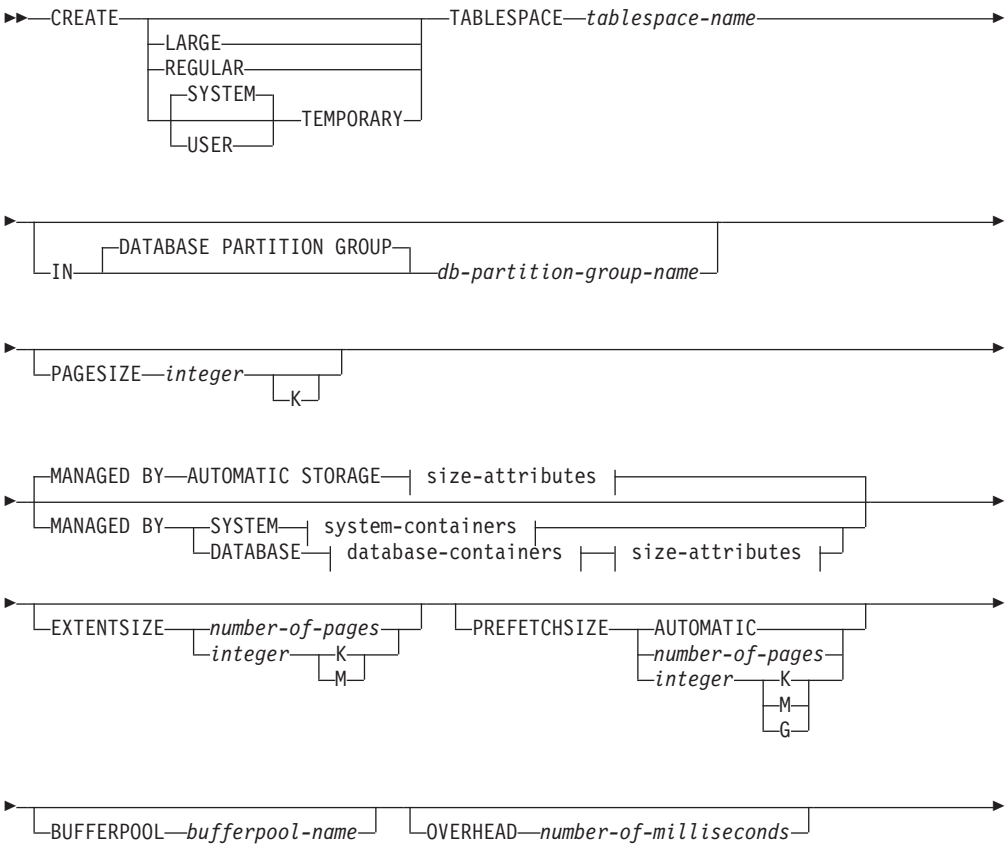
호출

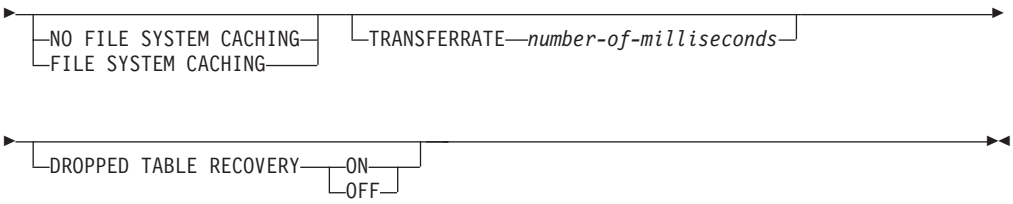
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

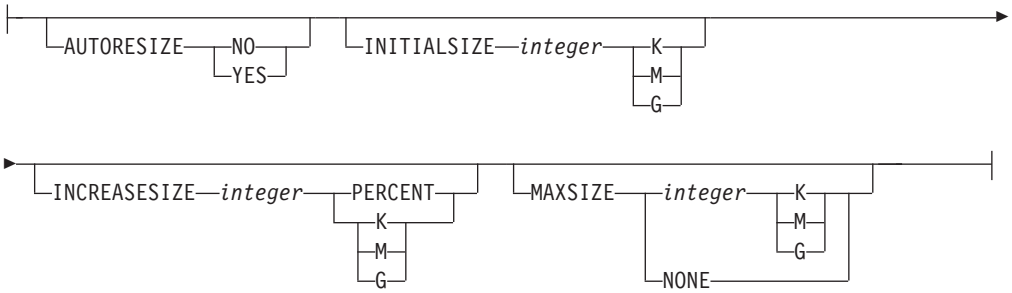
명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 SYSCTRL 권한을 포함해야 합니다.

구문

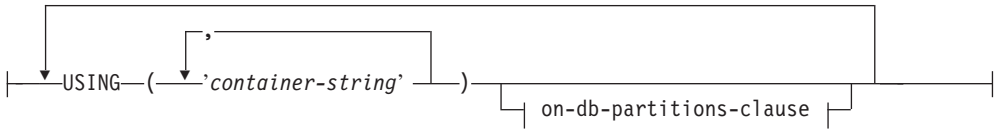




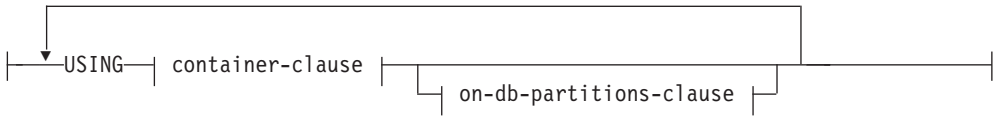
size-attributes:



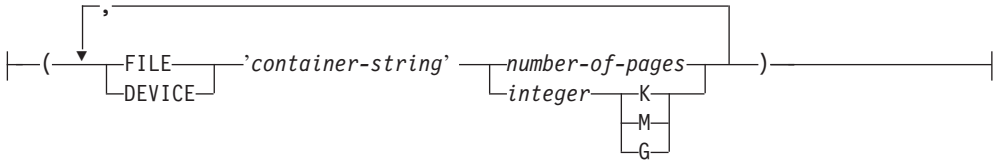
system-containers:



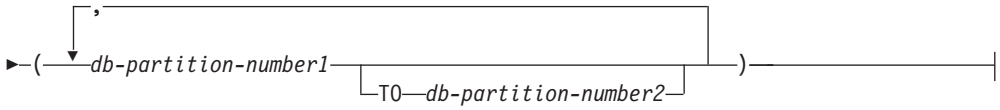
database-containers:



container-clause:



on-db-partitions-clause:



설명

LARGE, REGULAR, SYSTEM TEMPORARY 또는 USER TEMPORARY

작성될 테이블 스페이스의 유형을 지정합니다. 유형이 지정되지 않으면 MANAGED BY절이 디폴트값을 결정합니다.

LARGE

모든 영구 데이터를 저장합니다. 이 유형은 데이터베이스 관리 스페이스(DMS) 테이블 스페이스에서만 허용됩니다. 또한 이 유형은 유형이 지정되지 않는 경우 DMS 테이블 스페이스의 디폴트 유형이 됩니다. 테이블이 대형 테이블 스페이스에 위치한 경우는 다음과 같은 등록 정보를 나타냅니다.

- 일반 테이블 스페이스의 테이블보다 클 수 있습니다. 테이블 및 테이블 스페이스 한계에 대한 자세한 내용은『SQL 한계』를 참조하십시오.
- 테이블은 데이터 페이지당 255행 이상을 지원할 수 있으며 이는 데이터 페이지의 스페이스 사용도를 개선할 수 있습니다.
- 테이블에 정의된 인덱스는 일반 테이블 스페이스에 있는 테이블에 정의된 인덱스에 비해 행 항목당 2바이트가 더 필요합니다.

REGULAR

모든 영구 데이터를 저장합니다. 이 유형은 DMS 및 SMS 테이블 스페이스 모두에 적용됩니다. 이 유형은 SMS 테이블 스페이스에 대해 허용되는 유일한 유형이며, 유형이 지정되지 않는 경우 SMS 테이블 스페이스의 디폴트 유형이 됩니다.

SYSTEM TEMPORARY

데이터베이스 관리 프로그램이 정렬 또는 조인과 같은 조작을 수행하는 데 사용하는 작업 영역인 임시 테이블을 저장합니다. 임시 테이블은 SYSTEM TEMPORARY 테이블 스페이스에만 저장될 수 있으므로, 데이터베이스에는 항상 최소한 하나의 SYSTEM TEMPORARY 테이블 스페이스가 있어야 합니다. 임시 테이블 스페이스는 데이터베이스가 작성될 때 자동으로 작성됩니다.

USER TEMPORARY

작성된 임시 테이블과 선언된 임시 테이블을 저장합니다. 데이터베이스 작성시 사용자 임시 테이블 스페이스가 존재하지 않습니다. 작성된 임시 테이블 또는 선언된 임시 테이블의 정의가 허용되도록 적절한 USE 특권으로 최소한 하나의 사용자 임시 테이블 스페이스를 작성해야 합니다.

tablespace-name

테이블 스페이스 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이것은 SQL

ID(일반 또는 분리 ID)입니다. *tablespace-name*은 카탈로그에 이미 존재하는 테이블 스페이스를 식별하면 안됩니다(SQLSTATE 42710). *tablespace-name*은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

IN DATABASE PARTITION GROUP *db-partition-group-name*
테이블 스페이스에 대한 데이터베이스 파티션 그룹을 지정합니다. 데이터베이스 파티션 그룹이 존재해야 합니다. SYSTEM TEMPORARY 테이블 스페이스 작성시 지정할 수 있는 유일한 데이터베이스 파티션 그룹은 IBMTEMPGROUP뿐입니다. DATABASE PARTITION GROUP 키워드는 선택적입니다.

데이터베이스 파티션 그룹을 지정하지 않은 경우, 디폴트 데이터베이스 파티션 그룹(IBMDEFAULTGROUP)은 REGULAR, LARGE 및 USER TEMPORARY 테이블 스페이스에 사용됩니다. SYSTEM TEMPORARY 테이블 스페이스의 경우, 디폴트 데이터베이스 파티션 그룹 IBMTEMPGROUP이 사용됩니다.

PAGESIZE *integer* **[K]**
테이블 스페이스에 사용되는 페이지 *크기*를 정의합니다. 접미부 K가 없는 *integer*에 유효한 값은 4 096, 8 192, 16 384 또는 32 768입니다. 접미부 K가 있는 *integer*에 유효한 값은 4, 8, 16 또는 32입니다. *integer*와 K 사이에는 공백이 없을 수도 있고 임의의 수의 공백이 들어갈 수도 있습니다. 페이지 *크기*가 이러한 값 중의 하나가 아니거나(SQLSTATE 428DE) 테이블 스페이스가 연관된 버퍼 풀의 페이지 *크기*와 동일하지 않으면 오류가 발생합니다(SQLSTATE 428CB).

디폴트값은 데이터베이스 작성시 설정된 **pagesize** 데이터베이스 구성 매개변수에 의해 제공됩니다.

MANAGED BY AUTOMATIC STORAGE
테이블 스페이스가 자동 스토리지 테이블 스페이스임을 지정합니다. 데이터베이스에 대해 자동 스토리지가 정의되지 않은 경우 오류가 리턴됩니다(SQLSTATE 55060).

자동 스토리지 테이블 스페이스가 시스템 관리 스페이스(SMS) 테이블 스페이스 또는 데이터베이스 관리 스페이스(DMS) 테이블 스페이스로 작성됩니다. DMS가 선택되고 테이블 스페이스의 유형이 지정되지 않은 경우에는 대형 테이블 스페이스 작성이 디폴트 동작입니다. 자동 스토리지 테이블 스페이스에서 데이터베이스 관리 프로그램은 데이터베이스와 연관된 스토리지 경로에 기초하여 테이블 스페이스에 지정된 컨테이너를 결정합니다.

size-attributes
자동 스토리지 테이블 스페이스 또는 자동 스토리지 테이블 스페이스가 아닌 DMS 테이블 스페이스에 *크기* 속성을 지정합니다. SMS 테이블 스페이스는 자동 *크기* 조정이 불가능합니다.

AUTORESIZE
DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스의 자동 *크기* 조정 성능의 사용 여부를 지정합니다. 자동 *크기* 조정 가능 테이블 스페이스는 가득

차게 되면 자동으로 크기가 증가됩니다. 디폴트값은 DMS 테이블 스페이스의 경우 NO이고 자동 스토리지 테이블 스페이스의 경우 YES입니다.

NO

DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스의 자동 크기 조정 성능의 사용 안함 여부를 지정합니다.

YES

DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스의 자동 크기 조정 성능의 사용을 지정합니다.

INITIALSIZE *integer* **K | M | G**

자동 스토리지 테이블 스페이스의 데이터베이스 파티션별 초기 크기를 지정합니다. 이 옵션은 자동 스토리지 테이블 스페이스에 대해서만 유효합니다. 정수 값은 K(킬로바이트), M(메가바이트) 또는 G(기가바이트) 앞에 나와야 합니다. 데이터베이스 관리 프로그램이 테이블 스페이스의 컨테이너에서 크기를 일관적으로 유지하려고 하기 때문에 사용되는 실제 값은 지정한 값보다 약간 작을 수 있습니다. 또한 테이블 스페이스가 자동 크기 조정이 가능하고 최초 크기가 새 테이블 스페이스에 추가되어야 하는 메타데이터를 포함할 정도로 충분히 크지 않은 경우에는 데이터베이스 관리 프로그램이 충분한 스페이스를 확보할 때까지 **INCREASESIZE**의 값을 사용하여 테이블 스페이스 확장을 계속합니다. **INITIALSIZE**절이 지정되지 않은 경우, 데이터베이스 관리 프로그램이 적절한 값을 결정합니다. *integer*의 값이 적어도 48K여야 합니다.

INCREASESIZE *integer* **PERCENT 또는 INCREASESIZE** *integer* **K | M | G**

테이블 스페이스가 가득 차고 스페이스 요청이 있으면 자동 크기 조정이 사용 가능한 테이블 스페이스가 자동으로 증가되는 양을 데이터베이스 파티션별로 지정합니다. 정수 값은 다음 앞에 나와야 합니다.

- 스페이스 요청시 테이블 스페이스 크기 백분율로 양을 지정하는 PERCENT. PERCENT가 지정되면 정수 값은 0 - 100 사이에 있어야 합니다 (SQLSTATE 42615).
- 바이트 단위로 양을 지정하는 K(킬로바이트), M (메가바이트) 또는 G(기가바이트)

데이터베이스 관리 프로그램이 테이블 스페이스의 컨테이너에서 증가량을 일관적으로 유지하려고 하기 때문에 사용되는 실제 값은 지정한 값보다 약간 작거나 클 수 있습니다. 테이블 스페이스가 자동 크기 조정 가능하지만 INCREASESIZE절이 지정되지 않은 경우, 데이터베이스 관리 프로그램이 적절한 값을 결정합니다.

MAXSIZE *integer* **K | M | G 또는 MAXSIZE NONE**

자동 크기 조정이 사용 가능한 테이블 스페이스가 자동으로 증가될 수 있도록

최대 크기를 지정합니다. 테이블 스페이스가 자동 크기 조정 가능하지만 MAXSIZE절이 지정되지 않은 경우, 디폴트값은 NONE입니다.

integer

DMS 테이블 스페이스 또는 자동 스토리지 테이블 스페이스가 자동으로 증가될 수 있도록 데이터베이스 파티션별로 크기에 대한 하드 한계를 지정합니다. 정수 값은 K(킬로바이트), M(메가바이트) 또는 G(기가바이트) 앞에 나와야 합니다. 데이터베이스 관리 프로그램이 테이블 스페이스의 컨테이너에서 증가량을 일관적으로 유지하려고 하기 때문에 사용되는 실제 값은 지정한 값보다 약간 작을 수 있습니다.

NONE

테이블 스페이스가 파일 시스템 용량 또는 최대 테이블 스페이스 크기(『SQL 한계』에서 설명)까지 증가될 수 있도록 지정합니다.

MANAGED BY SYSTEM

테이블 스페이스가 SMS 테이블 스페이스임을 지정합니다. 테이블 스페이스의 유형이 지정되지 않은 경우에는 일반 테이블 스페이스 작성이 디폴트 동작입니다.

system-containers

SMS 테이블 스페이스에 대한 컨테이너를 지정합니다.

USING *('container-string',...)*

SMS 테이블 스페이스의 경우, 해당 테이블 스페이스에 속하면서 해당 테이블 스페이스의 데이터가 저장될 컨테이너를 식별합니다. *container-string*은 240바이트를 초과할 수 없습니다.

각 *container-string*은 절대 또는 상대 디렉토리 이름이 될 수 있습니다.

절대 디렉토리가 아닌 경우 디렉토리 이름은 데이터베이스 디렉토리에 대해 상대적이며, 데이터베이스 디렉토리와 실제적으로 연결되지 않은 스토리지로 가는 경로 이름 별명(UNIX 시스템의 기호 링크)이 될 수 있습니다. 예를 들어 `<dbdir>/work/c1`은 별도의 파일 시스템으로의 기호 링크가 될 수 있습니다.

디렉토리 이름의 구성요소가 존재하지 않으면, 데이터베이스 관리 프로그램이 이를 작성합니다. 테이블 스페이스가 삭제되면, 데이터베이스 관리 프로그램이 작성한 모든 구성요소도 삭제됩니다. *container-string*에 의해 식별된 디렉토리가 존재하는 경우 그 디렉토리에 파일 또는 서브디렉토리가 들어 있으면 안됩니다 (SQLSTATE 428B2).

*container-string*의 형식은 운영 체제에 따라 다릅니다. Windows 운영 체제에서는 절대 디렉토리 경로 이름이 드라이브 이름 및 콜론(:)으로 시작하고, UNIX 시스템에서는 슬래시(/)로 시작합니다. 모든 플랫폼에서 상대 경로 이름은 운영 체제와 상관없는 문자로 시작합니다.

리모트 자원(예: LAN 경로 재지정된 드라이브 또는 NFS 마운트 파일 시스템)은 현재 Network Appliance Filers, IBM iSCSI, IBM Network Attached

Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 또는 S4100 또는 Windows DB2 서버가 있는 NEC Storage NS 시리즈를 사용할 경우에만 지원됩니다. NEC Storage NS 시리즈는 UPS(Uninterrupted Power Supply)를 사용할 경우에만 지원됩니다. 대기(Standby)보다는 연속 UPS를 사용하도록 권장합니다. AIX의 NFS 마운트 파일 시스템은 -o nointr 옵션을 사용하여 무정전 모드에서 마운트되어 있어야 합니다.

on-db-partitions-clause

파티션된 데이터베이스에서 컨테이너가 작성되는 데이터베이스 파티션을 지정합니다. 이 절이 지정되지 않으면 다른 *on-db-partitions-clause*에 명시적으로 지정되지 않은 데이터베이스 파티션 그룹의 데이터베이스 파티션에 컨테이너가 작성됩니다. 데이터베이스 파티션 그룹 IBMTEMPGROUP에 정의된 SYSTEM TEMPORARY 테이블 스페이스의 경우, *on-db-partitions-clause*가 지정되지 않으면 데이터베이스에 추가된 모든 새 데이터베이스 파티션에도 컨테이너가 작성됩니다.

MANAGED BY DATABASE

테이블 스페이스가 DMS 테이블 스페이스임을 지정합니다. 테이블 스페이스의 유형이 지정되지 않은 경우에는 대형 테이블 스페이스 작성이 디폴트 동작입니다.

database-containers

DMS 테이블 스페이스에 대한 컨테이너를 지정합니다.

USING

container-clause를 사용합니다.

container-clause

DMS 테이블 스페이스에 대한 컨테이너를 지정합니다.

(FILE|DEVICE *'container-string' number-of-pages,...***)**

DMS 테이블 스페이스의 경우, 해당 테이블 스페이스에 속하면서 해당 테이블 스페이스의 데이터가 저장될 컨테이너를 식별합니다. 컨테이너의 유형(FILE 또는 DEVICE) 및 크기(PAGESIZE 페이지)가 지정됩니다. 크기는 정수로도 지정될 수 있으며 숫자 뒤에 K(킬로바이트인 경우), M(메가바이트인 경우) 또는 G(기가바이트인 경우)가 올 수 있습니다. 이런 방법으로 지정될 경우, 페이지 크기로 나눈 최소 바이트 수는 컨테이너에 대한 페이지 수를 결정하는 데 사용됩니다. FILE과 DEVICE 컨테이너를 혼합하여 지정할 수 있습니다. *container-string*은 254바이트를 초과할 수 없습니다.

FILE 컨테이너의 경우, *container-string*은 절대 또는 상대 파일 이름이어야 합니다. 절대 경로가 아니면 파일 이름은 데이터베이스 디렉토리에 상대적입니다. 디렉토리 이름의 구성요소가 존재하지 않으면, 데이터베이스 관리 프로그램이 이를 작성합니다. 파일이 존재하지 않으면, 파일이 작성되어

데이터베이스 관리 프로그램이 지정하는 *크기*로 초기화됩니다. 테이블 스페이스가 삭제되면, 데이터베이스 관리 프로그램이 작성한 모든 구성요소도 삭제됩니다.

주: 파일이 존재하면 겹쳐쓰여지고, 지정된 *크기*보다 작으면 확장됩니다. 지정된 *크기*보다 크더라도, 파일은 잘리지 않습니다.

DEVICE 컨테이너의 경우, *container-string*은 디바이스 이름이어야 합니다. 디바이스는 이미 존재하고 있어야 합니다.

모든 컨테이너는 모든 데이터베이스에서 고유해야 합니다. 하나의 컨테이너는 단 하나의 테이블 스페이스에만 속할 수 있습니다. 컨테이너의 *크기*는 다를 수 있으나 모든 컨테이너가 동일한 *크기*일 때 최적의 성능을 얻을 수 있습니다. *container-string*의 정확한 형식은 운영 체제에 따라 다릅니다.

리모트 자원(예: LAN 경로 재지정된 드라이브 또는 NFS 마운트 파일 시스템)은 현재 Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 또는 S4100 또는 Windows DB2 서버가 있는 NEC Storage NS 시리즈를 사용할 경우에만 지원됩니다. NEC Storage NS 시리즈는 UPS(Uninterrupted Power Supply)를 사용할 경우에만 지원됩니다. 대기(Standby)보다는 연속 UPS를 사용하도록 권장합니다.

on-db-partitions-clause

파티션된 데이터베이스에서 컨테이너가 작성되는 데이터베이스 파티션을 지정합니다. 이 절이 지정되지 않으면 다른 *on-db-partitions-clause*에 명시적으로 지정되지 않은 데이터베이스 파티션 그룹의 데이터베이스 파티션에 컨테이너가 작성됩니다. 데이터베이스 파티션 그룹 IBMTEMPGROUP에 정의된 SYSTEM TEMPORARY 테이블 스페이스의 경우, *on-db-partitions-clause*가 지정되지 않으면 데이터베이스에 추가된 모든 새 데이터베이스 파티션에도 컨테이너가 작성됩니다.

on-db-partitions-clause

파티션된 데이터베이스에서 컨테이너가 작성되는 데이터베이스 파티션을 지정합니다.

ON DBPARTITIONNUMS

지정된 개별 데이터베이스 파티션을 가리키는 키워드. DBPARTITIONNUM은 DBPARTITIONNUMS와 동의어입니다.

db-partition-number1

데이터베이스 파티션 번호를 지정합니다.

TO *db-partition-number2*

데이터베이스 파티션 번호의 범위를 지정하십시오. *db-partition-number2* 값은 *db-partition-number1* 값보다 크거나 같아야 합니다(SQLSTATE 428A9). 컨테이너는 지정된 값 사이 및 지정된 값을 포함하는 각 데이터

베이스 파티션에서 작성됩니다. 지정된 데이터베이스 파티션은 테이블 스페이스에 대한 데이터베이스 파티션 그룹에 있어야 합니다.

번호 순으로 지정된 데이터베이스 파티션 및 데이터베이스 파티션의 지정된 범위 내의 모든 데이터베이스 파티션은 테이블 스페이스에 대한 데이터베이스 파티션 그룹에 존재해야 합니다(SQLSTATE 42729). 데이터베이스 파티션 번호는 명시적으로만 보이거나 명령문에 대한 정확히 하나의 *on-db-partitions-clause*의 범위 내에서만 보일 수 있습니다(SQLSTATE 42613).

EXTENTSIZE *number-of-pages*

다음 컨테이너로 건너뛰기 전에 컨테이너에 기록될 PAGESIZE 페이지의 수를 지정합니다. Extent 크기 값은 정수로 지정될 수 있으며 숫자 뒤에 K(킬로바이트의 경우), M(메가바이트의 경우)이 붙습니다. 이런 방법으로 지정될 경우, 바이트 수를 페이지 크기로 나눈 최저값은 Extent 크기에 대한 값을 결정하는 데 사용됩니다. 데이터베이스 관리 프로그램은 데이터가 저장됨에 따라 컨테이너를 반복적으로 순환합니다.

디폴트값은 유효한 범위가 있는 **dft_extent_sz** 데이터베이스 구성 매개변수에 의해 제공되며, 유효한 범위는 2 - 256페이지입니다.

PREFETCHSIZE

쿼리에서 참조되기 전에 쿼리에서 필요한 데이터를 읽도록 지정하여, 입출력 수행시 쿼리가 지연되지 않도록 합니다.

디폴트값은 **dft_prefetch_sz** 데이터베이스 구성 매개변수에 의해 제공됩니다.

AUTOMATIC

테이블 스페이스의 프리페치 크기가 자동으로 갱신되도록 지정하며, 즉 프리페치 크기는 다음 공식을 사용하여 DB2가 관리합니다.

```
프리페치 크기 =
  (컨테이너 수) *
  (컨테이너당 실제 디스크 수) *
  (Extent 크기)
```

DB2_PARALLEL_IO 레지스트리 변수를 통해 값을 지정하지 않을 경우, 컨테이너당 실제 디스크 수의 디폴트값은 1입니다.

DB2는 하나 이상의 컨테이너를 추가 또는 삭제하는 ALTER TABLESPACE문의 성공적인 실행 후 테이블 스페이스의 컨테이너 수가 변경될 때마다 프리페치 크기를 자동으로 갱신합니다. 프리페치 크기는 데이터베이스가 시작될 때 갱신됩니다.

number-of-pages

데이터 프리페치가 수행되는 동안 테이블 스페이스로부터 읽어 들일 PAGESIZE 페이지의 수를 지정합니다. 프리페치 크기 값은 정수로도 지정할 수 있으며 숫자 뒤에 K(킬로바이트의 경우), M(메가바이트의 경우) 또

는 G(기가바이트의 경우)가 붙습니다. 이런 방식으로 지정될 경우, 바이트 수를 페이지 *크기*로 나눈 최저값은 프리페치 *크기*에 대한 페이지 수를 결정하는 데 사용됩니다.

BUFFERPOOL *bufferpool-name*

이 테이블 스페이스의 테이블에 사용된 버퍼 풀의 이름입니다. 버퍼 풀은 반드시 존재해야 합니다(SQLSTATE 42704). 지정하지 않으면, 디폴트 버퍼 풀(IBMDEFAULTBP)이 사용됩니다. 버퍼 풀의 페이지 *크기*는 테이블 스페이스에 대해 지정된 페이지 *크기*(또는 디폴트값)와 일치해야 합니다(SQLSTATE 428CB). 테이블 스페이스의 데이터베이스 파티션 그룹은 버퍼 풀에 대해 정의되어야 합니다(SQLSTATE 42735).

OVERHEAD *number-of-milliseconds*

입출력 제어기 오버헤드와 디스크 탐색 및 대기 시간을 지정합니다. 이 값은 쿼리 최적화 시 I/O의 비용을 계산할 때 사용됩니다. *number-of-milliseconds* 값은 모든 숫자 리터럴(정수, 10진수 또는 부동 소수점)입니다. 이 값이 모든 컨테이너에 대해 동일하지 않은 경우 이 숫자는 테이블 스페이스에 속하는 모든 컨테이너에 대한 평균이어야 합니다.

버전 9 이상에서 작성된 데이터베이스의 경우 디폴트 입출력 제어기 오버헤드와 디스크 검색 및 대기 시간은 7.5 밀리초입니다. DB2 이전 버전에서 버전 9 이상으로 업그레이드된 데이터베이스의 경우 디폴트값은 12.67밀리초입니다.

FILE SYSTEM CACHING 또는 NO FILE SYSTEM CACHING

I/O 조작이 파일 시스템 레벨에서 캐시되는지 여부를 지정합니다. 옵션이 지정되지 않으면 디폴트 값은 다음과 같습니다.

- AIX, Linux System z®에서의 JFS용 FILE SYSTEM CACHING, Solaris에서의 모든 비VxFS 파일 시스템, 모든 플랫폼 및 모든 LOB 및 대용량 데이터에서의 HPUX, SMS 임시 테이블 스페이스 파일
- 다른 모든 플랫폼과 파일 시스템 유형에서의 NO FILE SYSTEM CACHING

FILE SYSTEM CACHING

목표 테이블 스페이스의 모든 I/O 조작이 파일 시스템 레벨에서 캐시되도록 지정합니다.

NO FILE SYSTEM CACHING

모든 I/O 조작이 파일 시스템 레벨 캐시를 생략하도록 지정합니다.

TRANSFERRATE *number-of-milliseconds*

1페이지를 메모리로 읽는 시간을 지정합니다. 이 값은 쿼리 최적화 시 I/O의 비용을 계산할 때 사용됩니다. *number-of-milliseconds* 값은 모든 숫자 리터럴(정

수, 10진수 또는 부동 소수점)입니다. 이 값이 모든 컨테이너에 대해 동일하지 않은 경우 이 숫자는 테이블 스페이스에 속하는 모든 컨테이너에 대한 평균이어야 합니다.

버전 9 이상에서 작성된 데이터베이스의 경우 1페이지를 메모리로 읽는 디폴트 시간은 0.06 밀리초입니다. DB2 이전 버전에서 버전 9 이상으로 업그레이드된 데이터베이스의 경우 디폴트값은 0.18밀리초입니다.

DROPPED TABLE RECOVERY

지정된 테이블 스페이스의 삭제된 테이블이 ROLLFORWARD DATABASE 명령의 RECOVER TABLE ON 옵션을 사용하여 복구될 수 있는지의 여부를 표시합니다. 이 절은 일반 또는 대형 테이블 스페이스에 대해서만 지정할 수 있습니다(SQLSTATE 42613).

ON

삭제된 테이블을 복구할 수 있도록 지정합니다. 버전 8 이상의 디폴트값입니다.

OFF

삭제된 테이블을 복구할 수 없도록 지정합니다. 버전 7의 디폴트값입니다.

규칙

- 데이터베이스에 대해 자동 스토리지가 정의되지 않은 경우 오류가 리턴됩니다(SQLSTATE 55060).
- INITIALSIZE절은 MANAGED BY SYSTEM 또는 MANAGED BY DATABASE 절과 함께 지정할 수 없습니다(SQLSTATE 42601).
- AUTORESIZE, INCREASESIZE 또는 MAXSIZE절은 MANAGED BY SYSTEM 절과 함께 지정할 수 없습니다(SQLSTATE 42601).
- AUTORESIZE, INITIALSIZE, INCREASESIZE 또는 MAXSIZE절은 임시 자동 스토리지 테이블 스페이스 작성시 지정할 수 없습니다(SQLSTATE 42601).
- 테이블 스페이스의 자동 *크기* 조정이 불가능한 경우 INCREASESIZE 또는 MAXSIZE절을 지정할 수 없습니다(SQLSTATE 42601).
- 원시 디바이스 컨테이너를 사용하기 위해 정의된 DMS 테이블 스페이스에는 AUTORESIZE를 사용 가능하게 할 수 없습니다(SQLSTATE 42601).
- 테이블 스페이스는 처음에 5개의 Extent를 보유할 수 있을만큼 충분히 커야 합니다(SQLSTATE 57011).
- 테이블 스페이스의 최대 *크기*는 *초기 크기*보다 커야 합니다(SQLSTATE 560B0).
- 데이터베이스 관리 프로그램이 이러한 테이블 스페이스에 대한 스페이스 관리를 제어 하고 있기 때문에 자동 스토리지 테이블 스페이스에서 컨테이너 조작(ADD, EXTEND, RESIZE, DROP 또는 BEGIN STRIPE SET)을 수행할 수 없습니다(SQLSTATE 42858).

- 각 컨테이너 정의에는 컨테이너 이름을 저장하는 데 필요한 바이트 수 더하기 53바이트가 필요합니다. 테이블 스페이스의 모든 컨테이너 이름 길이의 합계가 20 480바이트를 초과할 수 없습니다(SQLSTATE 54034).
- 파티션된 데이터베이스의 경우 2개 이상의 데이터베이스 파티션이 동일한 실제 노드에 상주하면 2개 이상의 데이터베이스 파티션에 대해 동일한 디바이스 또는 경로를 지정할 수 없습니다(SQLSTATE 42730). 이러한 환경에서는 각 데이터베이스 파티션에 대해 고유한 *container-string*을 지정하거나 상대 경로 이름을 사용하십시오.

주

- 기본적으로 테이블 스페이스에 대해 데이터베이스 관리 스페이스를 선택할 것인지 아니면 시스템 관리 스페이스를 선택할 것인지는 장단점을 따져 봐야 합니다.
- 2개 이상의 TEMPORARY 테이블 스페이스가 데이터베이스에 존재하면, 사용의 균형을 이루도록 TEMPORARY 테이블 스페이스에 라운드 로빈 방식이 사용됩니다.
- 테이블 스페이스 소유자에게는 테이블 스페이스 작성 시 테이블 스페이스에 대한 WITH GRANT OPTION이 있는 USE 특권이 부여됩니다.
- SMS 또는 DMS 컨테이너 작성 시 컨테이너 문자열 구문에 대한 데이터베이스 파티션 표현식을 지정할 수 있습니다. 파티션된 데이터베이스 시스템에서 다중 논리 데이터베이스 파티션을 사용했을 경우, 일반적으로 데이터베이스 파티션 표현식을 지정하십시오. 이는 컨테이너 이름이 데이터베이스 파티션 서버 전체에서 고유하도록 보장합니다. 표현식을 지정할 때 데이터베이스 파티션 번호는 컨테이너 이름의 일부이거나, 추가 인수를 지정할 경우 인수의 결과는 컨테이너 이름의 일부입니다.

  인수『 $N』([blank]$N)을 사용하여 데이터베이스 파티션 표현식을 나타낼 수 있습니다. 데이터베이스 파티션 표현식은 컨테이너 이름의 어디에서든 사용할 수 있으며 데이터베이스 파티션 표현식을 여러 개 지정할 수도 있습니다. 데이터베이스 파티션 표현식은 공백 문자로 끝내십시오. 공백 다음에 오는 문자는 데이터베이스 파티션 표현식을 평가한 후 컨테이너 이름에 추가됩니다. 컨테이너 이름에서 데이터베이스 파티션 표현식 뒤에 공백 문자가 없으면 나머지 문자열은 표현식의 일부로 간주됩니다. 인수는 다음 형식 중 한 가지 형식으로만 사용할 수 있습니다.

표 23. 컨테이너 작성에 필요한 인수. 연산자는 왼쪽에서 오른쪽으로 계산됩니다. 예에서 데이터베이스 파티션 번호는 5로 가정됩니다.

구문	예 :	값
[blank]$N	″ $N″	5
[blank]$N+[number]	″ $N+1011″	1016
[blank]$N%[number]	″ $N%3″ [a]	2
[blank]$N+[number]%[number]	″ $N+12%13″	4
[blank]$N%[number]+[number]	″ $N%3+20″	22

[a] %는 모듈러스 연산자를 나타냅니다.

예를 들면, 다음과 같습니다.

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
  (device '/dev/rcont $N' 20000)

   두 개의 데이터베이스 파티션 시스템에서,
    다음 컨테이너가 작성되었습니다.
       /dev/rcont0  - on DATABASE PARTITION 0
       /dev/rcont1  - on DATABASE PARTITION 1


CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
  (file '/DB2/containers/TS2/container $N+100' 10000)

   네 개의 데이터베이스 파티션에서,
    다음 컨테이너가 작성되었습니다.
       /DB2/containers/TS2/container100  - on DATABASE PARTITION 0
       /DB2/containers/TS2/container101  - on DATABASE PARTITION 1
       /DB2/containers/TS2/container102  - on DATABASE PARTITION 2
       /DB2/containers/TS2/container103  - on DATABASE PARTITION 3


CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
  ('/TS3/cont $N%2','/TS3/cont $N%2+2')

   두 개의 데이터베이스 파티션 시스템에서,
    다음 컨테이너가 작성되었습니다.
       /TS3/cont0  - On DATABASE PARTITION 0
       /TS3/cont2  - On DATABASE PARTITION 0
       /TS3/cont1  - On DATABASE PARTITION 1
       /TS3/cont3  - On DATABASE PARTITION 1


   데이터베이스 파티션이 다섯 개이고 컨테이너가 다음과 같은 경우
       '/dbdir/node $N /cont1'
  '/ $N+1000 /file1'
       ' $N%10 /container'
  '/dir/ $N2000 /dmscont'

   다음과 같은 경우가 포함됩니다.
       '/dbdir/node5/cont1'
       '/1005/file1'
       '5/container'
       '/dir/2000/dmscont'
```

- 자동 스토리지 테이블 스페이스가 SMS 테이블 스페이스 또는 DMS 테이블 스페이스로 작성됩니다. DMS는 대형 및 일반 테이블 스페이스에 대해, SMS는 임시 테이블 스페이스에 대해 선택됩니다. 이 동작은 추후 릴리스에서 변경될 수 있습니다. DMS가 선택되고 테이블 스페이스의 유형이 지정되지 않은 경우에는 대형 테이블 스페이스 작성이 디폴트 동작입니다.
- 자동 스토리지 테이블 스페이스 작성에는 컨테이너 정의가 포함되지 않습니다. 데이터베이스 관리 프로그램은 데이터베이스와 연관된 스토리지 경로에 기초하여 컨테이너의 위치 및 크기를 자동으로 결정합니다. 최대 크기에 도달하지 않은 경우 데이터베이스 관리 프로그램은 필요에 따라 대형 및 일반 테이블 스페이스를 증가시키려고

합니다. 여기에는 기존 컨테이너 확장 또는 새 스트라이프 세트에 컨테이너 추가가 포함될 수 있습니다. 데이터베이스가 활성화될 때마다 데이터베이스 관리 프로그램은 비정상 상태에 있지 않은 임시 테이블 스페이스에 대한 컨테이너의 수 및 위치를 자동으로 재구성합니다.

- 대형 또는 일반 자동 스토리지 테이블 스페이스는 테이블 스페이스가 사용하는 기존 스토리지 경로 중 하나에 더 이상 스페이스가 없을 때까지 새 스토리지 경로를 사용하지 않습니다(ALTER DATABASE문에 대한 설명 참조). 임시 자동 스토리지 테이블 스페이스는 일단 데이터베이스가 비활성화된 후 재활성화되는 경우에만 새 스토리지 경로를 사용할 수 있습니다.
- *호환성*: 이전 버전의 DB2 데이터베이스와의 호환성을 위해,
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - DBPARTITIONNUMS 대신 NODES를 지정할 수 있습니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.
  - LARGE 대신 LONG을 지정할 수 있습니다.

예:

예 1: 각각 10,000 4K 페이지의 3개의 디바이스를 사용하여 UNIX 시스템에서 대형 DMS 테이블 스페이스를 작성하십시오. 입출력 등록 정보를 지정하십시오.

```
CREATE TABLESPACE PAYROLL
  MANAGED BY DATABASE
  USING (DEVICE'/dev/rhdisk6' 10000,
    DEVICE '/dev/rhdisk7' 10000,
    DEVICE '/dev/rhdisk8' 10000)
  OVERHEAD 12.67
  TRANSFERRATE 0.18
```

*예 2:*64페이지의 Extent *크기* 및 32페이지의 프리페치 *크기*로 3개의 별도 드라이브의 3개의 디렉토리를 사용하여 Windows에 일반 SMS 테이블 스페이스를 작성하십시오.

```
CREATE TABLESPACE ACCOUNTING
  MANAGED BY SYSTEM
  USING ('d:₩acc_tbsp', 'e:₩acc_tbsp', 'f:₩acc_tbsp')
  EXTENTSIZE 64
  PREFETCHSIZE 32
```

예 3: 256페이지 Extent *크기*로 각각 50 000페이지의 2개 파일을 사용하여 UNIX 시스템에 임시 DMS 테이블 스페이스를 작성합니다.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
  MANAGED BY DATABASE
  USING (FILE 'dbtmp/tempspace2.f1' 50000,
    FILE 'dbtmp/tempspace2.f2' 50000)
  EXTENTSIZE 256
```

예 4: UNIX 시스템에서, 데이터베이스 파티션 그룹 ODDNODEGROUP(데이터베이스 파티션 1, 3 및 5)에 DMS 테이블 스페이스를 작성하십시오. 각 데이터베이스 파티션의 10,000 4K 페이지용으로 디바이스 /dev/rhdisk0을 사용하십시오. 각 데이터베이스 파티션에 대한 40 000 4K 페이지의 데이터베이스 파티션 특정 디바이스를 지정하십시오.

```
CREATE TABLESPACE PLANS
  MANAGED BY DATABASE
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
  ON DBPARTITIONNUM (1)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
  ON DBPARTITIONNUM (3)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
  ON DBPARTITIONNUM (5)
```

예 5: 시스템이 테이블 스페이스 *크기* 및 *증가*에 대한 모든 결정을 내릴 수 있도록 DATATS라는 이름의 대형 자동 스토리지 테이블 스페이스를 작성하십시오.

```
CREATE TABLESPACE DATATS
```

또는

```
CREATE TABLESPACE DATATS
  MANAGED BY AUTOMATIC STORAGE
```

예 6: TEMPDATA라는 이름의 임시 자동 스토리지 테이블 스페이스를 작성합니다.

```
CREATE TEMPORARY TABLESPACE TEMPDATA
```

또는

```
CREATE TEMPORARY TABLESPACE TEMPDATA
  MANAGED BY AUTOMATIC STORAGE
```

예 7: *초기 크기* 100MB, 최대 *크기* 1GB를 가진 대형 자동 스토리지 테이블 스페이스 USERSPACE3을 작성합니다.

```
CREATE TABLESPACE USERSPACE3
  INITIALSIZE 100 M
  MAXSIZE 1 G
```

예 8: *증가* 비율 10%(즉, 자동 *크기* 조정 때마다 10%씩 전체 *크기* *증가*) 및 최대 *크기* 512MB를 가진 대형 자동 스토리지 테이블 스페이스 LARGEDATA를 작성합니다. INITIALSIZE절을 지정하는 대신 데이터베이스 관리 프로그램이 테이블 스페이스의 적절한 *초기 크기*를 결정하게 합니다.

```
CREATE LARGE TABLESPACE LARGEDATA
  INCREASESIZE 10 PERCENT
  MAXSIZE 512 M
```

예 9: 2개의 파일 컨테이너(각 컨테이너 *크기*는 1MB), *증가* 비율 2MB, 최대 *크기* 100MB를 가진 대형 DMS 테이블 스페이스 USERSPACE4를 작성합니다.

```
CREATE TABLESPACE USERSPACE4
  MANAGED BY DATABASE USING (FILE '/db2/file1' 1 M, FILE '/db2/file2' 1 M)
       AUTORESIZE YES
       INCREASESIZE 2 M
  MAXSIZE 100 M
```

예 10: Windows 운영 체제의 RAW 디바이스를 사용하여 대형 DMS 테이블 스페이스를 작성하십시오.

- 실제 드라이브 전체를 지정하려면 ₩₩.₩*physical-drive* 형식을 사용하십시오.

  ```
  CREATE TABLESPACE TS1
   MANAGED BY DATABASE USING (DEVICE '₩₩.₩PhysicalDrive5' 10000,
     DEVICE '₩₩.₩PhysicalDrive6' 10000)
  ```

- 논리적 파티션을 지정하려면 드라이브 이름을 사용하십시오.

  ```
  CREATE TABLESPACE TS2
    MANAGED BY DATABASE USING (DEVICE '₩₩.₩G:' 10000,
      DEVICE '₩₩.₩H:' 10000)
  ```

- 볼륨 전역 고유 ID(GUID)를 사용하여 논리적 파티션을 지정하려면, db2listvolumes 유틸리티를 사용하여 각 로컬 파티션에 대한 볼륨 GUID를 검색한 후 원하는 논리적 파티션에 대한 GUID를 테이블 스페이스 컨테이너 절로 복사하십시오.

  ```
  CREATE TABLESPACE TS3
    MANAGED BY DATABASE USING (
      DEVICE '₩₩?₩Volume{2ca6a0c1-8542-11d8-9734-00096b5322d2}₩' 20000M)
  ```

  머신에 사용 가능한 드라이브 이름보다 많은 파티션이 있는 경우, 드라이브 이름 형식보다 볼륨 GUID를 사용하는 것이 나을 수 있습니다.

- 교차 지점(또는 볼륨 마운트 위치)을 사용하여 논리적 파티션을 지정하려면, RAW 파티션을 다른 NTFS 형식의 볼륨에 교차 지점으로 마운트한 후 NTFS 볼륨의 교차 지점에 이르는 경로를 컨테이너 경로로 지정하십시오. 예를 들면, 다음과 같습니다.

  ```
  CREATE TABLESPACE TS4
    MANAGED BY DATABASE USING (DEVICE 'C:₩JUNCTION₩DISK_1' 10000,
      DEVICE 'C:₩JUNCTION₩DISK_2' 10000)
  ```

  DB2는 우선 파티션에 파일 시스템이 있는지 여부를 판별하기 위해 파티션을 쿼리한 후, 파일 시스템이 있으면 파티션을 RAW 디바이스로 취급하지 않고 DB2가 파티션에 있는 보통 파일 시스템 입출력 조작을 수행합니다.

CREATE THRESHOLD

CREATE THRESHOLD문은 임계값을 정의합니다.

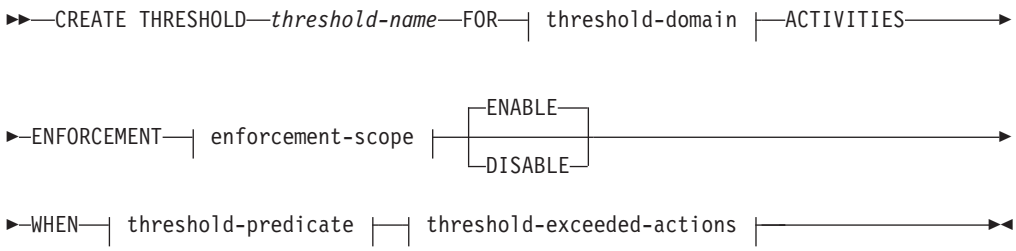
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

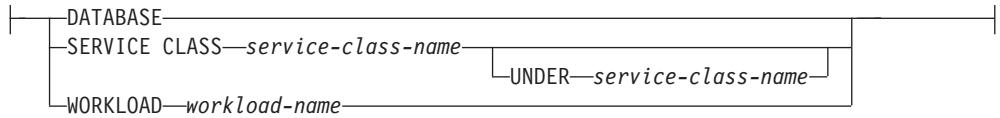
권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

구문

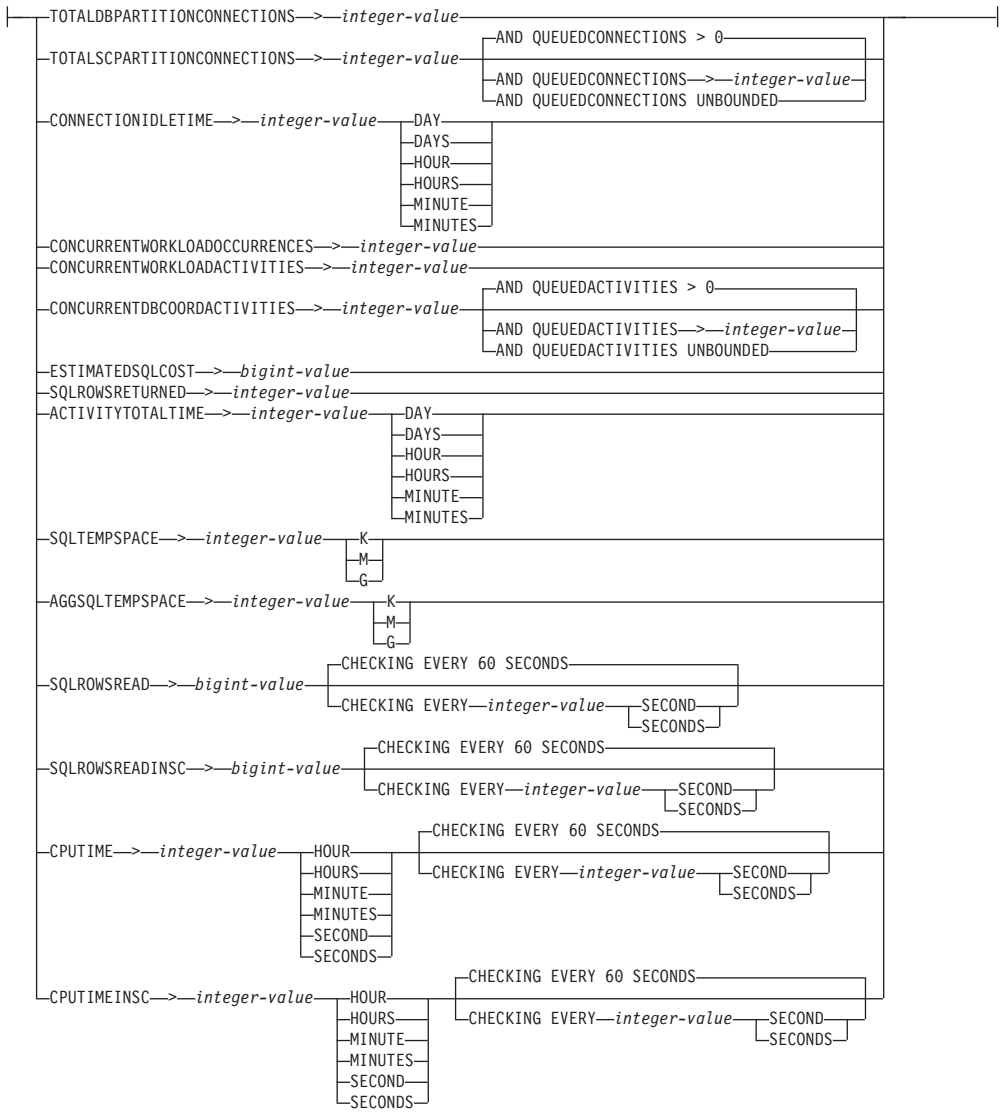


threshold-domain:

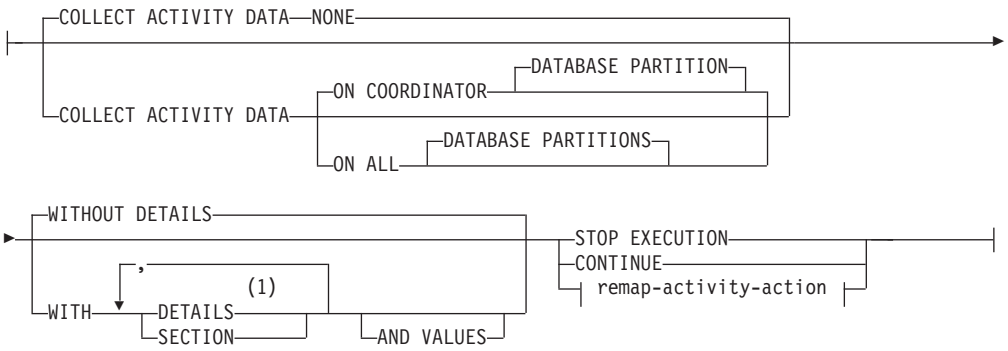


enforcement-scope:

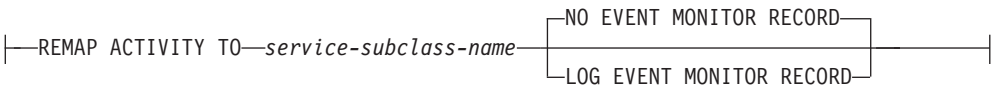
threshold-predicate:



threshold-exceeded-actions:



remap-activity-action:



주:

1 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

threshold-name

임계값의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL ID(일반 또는 분리 ID)입니다. *threshold-name*은 현재 서버에 이미 존재하는 임계값을 식별하면 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

FOR *threshold-domain* **ACTIVITIES**

임계값의 정의 도메인을 지정합니다.

DATABASE

임계값은 데이터베이스의 활동에 적용됩니다.

SERVICE CLASS *service-class-name*

임계값은 *service-class-name* 서비스 클래스에서 실행 중인 활동에 적용됩니다. UNDER가 지정되지 않은 경우, *service-class-name*은 기존 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704). UNDER가 지정된 경우, *service-class-name*은 UNDER 키워드 다음에 지정된 서비스 수퍼 클래스의 기존 서비스 서브클래스를 식별해야 합니다(SQLSTATE 42704).

UNDER *service-class-name*

서비스 수퍼 클래스를 지정합니다. *service-class-name*은 기존 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704).

WORKLOAD *workload-name*

임계값은 지정된 워크로드에 적용됩니다. *workload-name*은 기존 워크로드를 식별해야 합니다(SQLSTATE 42704).

ENFORCEMENT *enforcement-scope*

임계값의 시행 범위입니다.

DATABASE

임계값이 정의 도메인 내에서 모든 데이터베이스 파티션 사이에 시행됩니다(즉, 데이터베이스의 모든 데이터베이스 파티션과, 서비스 클래스의 모든 데이터베이스 파티션).

DATABASE PARTITION
임계값이 데이터베이스 파티션 기준으로 시행됩니다. 임계값을 시행하기 위한 모든 데이터베이스 파티션 사이의 조정은 없습니다.

WORKLOAD OCCURRENCE
임계값이 워크로드 어커런스 내에서만 시행됩니다. 동일한 데이터베이스 파티션에서 동시에 실행 중인 두 개의 워크로드 어커런스는 각각 해당 임계값에 대해 자체의 고유한 실행 중 계수를 갖습니다.

ENABLE 또는 DISABLE
데이터베이스 관리 프로그램에 의한 사용에 임계값이 사용 가능한지 여부를 지정합니다.

ENABLE
데이터베이스 활동 실행을 제한하기 위해 데이터베이스 관리 프로그램에서 임계값이 사용됩니다.

DISABLE
데이터베이스 활동 실행을 제한하기 위해 데이터베이스 관리 프로그램에서 임계값이 사용되지 않습니다.

WHEN *threshold-predicate*
임계값 조건을 지정합니다.

TOTALDBPARTITIONCONNECTIONS > *integer-value*
이 조건은 데이터베이스 파티션에서 동시에 실행할 수 있는 코디네이터 연결 수에 대한 상한을 정의합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 새 코디네이터 연결이 연결되지 않음을 의미합니다. 현재 실행 중이거나 큐에서 대기 상태인 모든 연결이 계속됩니다. 이 조건에 대한 정의 도메인은 DATABASE여야 하며, 시행 범위는 DATABASEPARTITION이어야 합니다(SQLSTATE 5U037).

TOTALSCPARTITIONCONNECTIONS > *integer-value*
이 조건은 특정 서비스 수퍼 클래스의 데이터베이스 파티션에서 동시에 실행할 수 있는 코디네이터 연결 수에 대한 상한을 정의합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 새 연결이 서버 클래스를 조인하지 않음을 의미합니다. 현재 실행 중이거나 큐에서 대기 상태인 모든 연결이 계속됩니다. 이 조건에 대한 정의 도메인은 SERVICE SUPERCLASS여야 하고, 시행 범위는 DATABASEPARTITION이어야 합니다(SQLSTATE 5U037).

AND QUEUEDCONNECTIONS > *integer-value* **또는 AND QUEUEDCONNECTIONS UNBOUNDED**
최대 코디네이터 연결 수를 초과할 때 큐 크기를 지정합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영

(0) 값은 코디네이터 연결이 큐에 대기되지 않음을 의미합니다. UNBOUNDED를 지정하면 지정된 최대 코디네이터 연결 수를 초과하는 모든 연결을 큐에 넣지만 *threshold-exceeded-actions*는 실행되지 않습니다. 디폴트값은 0입니다.

CONNECTIONIDLETIME > *integer-value* **DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES**

이 조건은 데이터베이스 관리 프로그램이 연결이 유휴 상태로 유지되도록 허용하는 시간의 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820). 유효한 지속기간 키워드를 사용하여 *integer-value*에 적절한 시간 단위를 지정하십시오. 이 조건에 대한 정의 도메인은 DATABASE 또는 SERVICE SUPERCLASS여야 하고, 시행 범위는 DATABASE여야 합니다(SQLSTATE 5U037). 이 조건은 코디네이터 데이터베이스 파티션에서 논리적으로 시행됩니다.

CONNECTIONIDLETIME 임계값을 사용하여 STOP EXECUTION 조치를 지정하는 경우 응용프로그램의 연결은 임계값이 초과될 때 삭제됩니다. 응용프로그램이 데이터 서버에 액세스하기 위한 연속 시도에 SQLSTATE 5U026이 수신되지 않습니다.

이 임계값의 최대값은 2,147,483,640초입니다. 2,147,483,640초보다 많은 초 값을 지정하면 이 초 수로 설정됩니다.

CONCURRENTWORKLOADOCCURRENCES > *integer-value*

이 조건은 각 데이터베이스 파티션에서 워크로드에 대한 동시 어커런스 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820). 이 조건에 대한 정의 도메인은 WORKLOAD여야 하고, 시행 범위는 DATABASE PARTITION이어야 합니다(SQLSTATE 5U037).

CONCURRENTWORKLOADACTIVITIES > *integer-value*

이 조건은 각 데이터베이스 파티션에서 워크로드에 대한 동시 코디네이터 활동 및 중첩 활동 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820). 이 조건에 대한 정의 도메인은 WORKLOAD여야 하고, 이 조건의 시행 범위는 WORKLOAD OCCURRENCE여야 합니다(SQLSTATE 5U037).

중첩된 각 활동은 다음 조건을 충족해야 합니다.

- 인식되는 코디네이터 활동이어야 합니다. 인식되는 활동 유형 내에 속하지 않는 중첩된 코디네이터 활동은 계산되지 않습니다. 마찬가지로, 리모트 노드 요청과 같은 중첩된 서브에이전트 활동은 계산되지 않습니다.
- SQL문을 발행하는 사용자 작성 프로시저와 같이, 사용자 논리를 통해 직접 호출해야 합니다.

결과적으로, SYSIBM, SYSFUN 또는 SYSPROC 스키마의 DB2 유틸리티 또는 루틴 호출 하에 자동으로 시작된 중첩된 코디네이터 활동은 이 임계값에 지정된 상한에 계산되지 않습니다.

제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값에서 계산되지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

CONCURRENTDBCOORDACTIVITIES > *integer-value*

이 조건은 지정된 도메인의 모든 데이터베이스 파티션에서 동시에 실행할 수 있는 인식된 데이터베이스 코디네이터 활동 수에 대한 상한을 정의합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 새 데이터베이스 코디네이터 활동이 실행되지 않음을 의미합니다. 현재 실행 중이거나 큐에서 대기 상태인 모든 데이터베이스 코디네이터 활동이 계속됩니다. 이 조건에 대한 정의는 DATABASE, 작업 조치(작업 조치 정의 도메인에 대한 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET문을 사용하여 작성됨), SERVICE SUPERCLASS 또는 SERVICE SUBCLASS여야 하며, 시행 범위는 DATABASE여야 합니다(SQLSTATE 5U037).

중요사항: CONCURRENTDBCOORDACTIVITIES 임계값의 결과 외부 개입이 필요한 큐 기반 경쟁이 발생할 수 있습니다. 자세한 정보는 "CONCURRENTDBCOORDACTIVITIES 임계값"을 참조하십시오.

AND QUEUEDACTIVITIES > *integer-value* **또는 AND QUEUEDACTIVITIES UNBOUNDED**

최대 데이터베이스 코디네이터 활동 수를 초과할 때 큐 크기를 지정합니다. 이 값은 영(0)을 포함한 어떤 양의 정수도 될 수 있습니다(SQLSTATE 42820). 영(0) 값은 데이터베이스 코디네이터 활동이 큐에 대기되지 않음을 의미합니다. UNBOUNDED를 지정하면 지정된 최대 데이터베이스 코디네이터 활동 수를 초과하는 모든 데이터베이스 코디네이터 활동을 큐에 넣지만 *threshold-exceeded-actions*는 실행되지 않습니다. 디폴트값은 0입니다.

ESTIMATEDSQLCOST > *bigint-value*

이 조건은 활동의 옵티마이저 지정 비용(timeron 단위)의 상한을 정의합니다. 이 값은 0이 아닌 양의 큰 정수가 될 수 있습니다(SQLSTATE 42820). 이 조건의 정의 도메인은 DATABASE여야 하며 작업 조치(작업 조치 정의 도메인의 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET문을 사용하여 작성), SERVICE SUPERCLASS, SERVICE SUBCLASS 또는 WORKLOAD 및 시행 범위는 DATABASE여야 합니다(SQLSTATE 5U037). 이 조건은 코디네이터 데이터베이스 파티션에서 시행됩니다. 이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 데이터 처리 언어(DML) 유형의 코디네이터 활동.
- 사용자 논리를 통해 호출되는 중첩된 DML 활동. 결과적으로, 데이터베이스 관리 프로그램에 의해 시작될 수 있는 DML 활동(예: 유틸리티, 프로시저 또는 내부 SQL)은 이 조건에 의해 추적되지 않습니다(간접적으로 추적되는 경우 상위의 추정에 비용이 포함되는 경우는 제외).

SQLROWSRETURNED > *integer-value*

이 조건은 응용프로그램 서버(AS)에서 클라이언트 응용프로그램으로 리턴되는 행 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 정수가 될 수 있습니다(SQLSTATE 42820). 이 조건의 정의 도메인은 DATABASE여야 하며 작업 조치(작업 조치 정의 도메인의 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET문을 사용하여 작성), SERVICE SUPERCLASS, SERVICE SUBCLASS 또는 WORKLOAD 및 시행 범위는 DATABASE여야 합니다(SQLSTATE 5U037). 이 조건은 코디네이터 데이터베이스 파티션에서 시행됩니다. 이 조건에 의해 추적되는 활동은 다음과 같습니다.

- DML 유형의 코디네이터 활동.
- 사용자 논리를 통해 파생되는 중첩된 DML 활동. 유틸리티, 프로시저 또는 내부 SQL을 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건의 영향을 받지 않습니다.

프로시저 내에서 리턴된 결과 세트는 개별적 활동으로 별도로 처리됩니다. 프로시저 자체에서 리턴되는 행의 집계는 없습니다.

ACTIVITYTOTALTIME > *integer-value* **DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES**

이 조건은 활동이 큐에서 대기한 시간을 포함하여, 데이터베이스 관리 프로그램이 연결 실행을 허용할 시간의 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820). 유효한 지속기간 키워드를 사용하여 *integer-value*에 적절한 시간 단위를 지정하십시오. 이 조건의 정의 도메인은 DATABASE여야 하며 작업 조치(작업 조치 정의 도메인의 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET문을 사용하여 작성), SERVICE SUPERCLASS, SERVICE SUBCLASS 또는 WORKLOAD 및 시행 범위는 DATABASE여야 합니다(SQLSTATE 5U037). 이 조건은 코디네이터 데이터베이스 파티션에서 논리적으로 시행됩니다.

이 임계값의 최대값은 2,147,483,640초입니다. 2,147,483,640초보다 많은 초 값을 지정하면 이 초 수로 설정됩니다.

SQLTEMPSPACE > *integer-value* **K | M | G**

이 조건은 데이터베이스 파티션에서 시스템 임시 테이블 스페이스 크기에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

integer-value K(대문자 또는 소문자)를 지정한 경우, 최대 *크기*는 1024 x *integer-value*입니다. *integer-value M*을 지정한 경우, 최대 *크기*는 1,048,576 x *integer-value*입니다. *integer-value G*를 지정한 경우, 최대 *크기*는 1 073 741 824 x *integer-value*입니다.

이 조건의 정의 도메인은 DATABASE이어야 하며 작업 조치(작업 조치 정의 도메인의 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET문을 사용하여 작성), SERVICE SUPERCLASS, SERVICE SUBCLASS 또는 WORKLOAD 및 시행 범위는 DATABASEPARTITION이어야 합니다(SQLSTATE 5U037). 이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당 서브에이전트 작업(서브섹션 실행).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행). 유틸리티, 프로시저 또는 내부 SQL을 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건의 영향을 받지 않습니다.

AGGSQLTEMPSPACE > *integer-value* **K | M | G**

이 조건은 데이터베이스 파티션의 정의 도메인에서 모든 활동 사이에 소비될 수 있는 시스템 임시 스페이스의 최대량을 정의합니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

integer-value K(대문자 또는 소문자)를 지정한 경우, 최대 *크기*는 1024 x *integer-value*입니다. *integer-value* M을 지정한 경우, 최대 *크기*는 1,048,576 x *integer-value*입니다. *integer-value* G를 지정한 경우, 최대 *크기*는 1 073 741 824 x *integer-value*입니다.

이 조건에 대한 정의 도메인은 SERVICE SUBCLASS여야 하고, 시행 범위는 DATABASE PARTITION이어야 합니다(SQLSTATE 5U037).

이 조건에 의해 추적되는 집계에 제공되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(예: 서브섹션 실행과 같은). 유틸리티, 프로시저 또는 내부 SQL문을 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건의 영향을 받지 않습니다.

SQLROWSREAD > *bigint-value*

이 조건은 특정 데이터베이스 파티션에서 수명 동안 활동에 의해 읽을 수 있는 행 수에 대한 상한을 정의합니다. 이 값은 0이 아닌 양의 큰 정수가 될 수 있습니다(SQLSTATE 42820). 읽혀진 행 수는 리턴된 행 수와 다릅니다. 리턴된 행 수는 SQLROWSRETURNED 조건의 제어를 받습니다.

이 조건에 대한 정의 도메인은 DATABASE, SERVICE SUPERCLASS, 서비스 서브클래스(UNDER 절을 지정하는 SERVICE CLASS), WORKLOAD 또는 작업 조치(작업 조치 정의 도메인의 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET 문을 사용하여 작성됨)여야 하고, 시행 범위는 DATABASE PARTITION이어야 합니다(SQLSTATE 5U037). 이 조건은 데이터베이스 파티션마다 독립적으로 시행됩니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.
- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. 임계값은 각 요청(페치 조작과 같은)의 끝에서, CHECKING 절에 정의된 간격으로 점검됩니다. CHECKING 절은 임계값 위반이 발견되지 않을 수 있는 시간의 상한을 정의합니다. 디폴트는 60초입니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

SQLROWSREADINSC > *bigint-value*

이 조건은 서비스 서브클래스에서 실행 중인 동안 특정 데이터베이스 파티션에서 활동에 의해 읽을 수 있는 행 수에 대한 상한을 정의합니다. 지정된 서비스 서브클래스에서 실행하기 전에 읽은 행 수는 계산되지 않습니다. 이 값은 0이 아닌 양의 큰 정수가 될 수 있습니다(SQLSTATE 42820). 읽혀진 행 수는 리턴된 행 수와 다릅니다. 리턴된 행 수는 SQLROWSRETURNED 조건의 제어를 받습니다.

이 조건에 대한 정의 도메인은 서비스 서브클래스(UNDER 절을 지정하는 SERVICE CLASS)여야 하고, 시행 범위는 DATABASE PARTITION이어야 합니다(SQLSTATE 5U037). 이 조건은 데이터베이스 파티션마다 독립적으로 시행됩니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.
- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. 임계값은 각 요청(페치 조작과 같은)의 끝에서, CHECKING 절에 정의된 간격으로 점검됩니다. CHECKING 절은 임계값 위반이 발견되지 않을 수 있는 시간의 상한을 정의합니다. 디폴트는 60초입니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

CPUTIME > *integer-value* **HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS**

이 조건은 특정 데이터베이스 파티션에서 수명 동안 활동에 소비될 수 있는 프로세서 시간에 대한 상한을 정의합니다. 이 임계값에 의해 추적되는 프로세서 시간은 활동이 실행을 시작하는 시간부터 측정됩니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

이 조건에 대한 정의 도메인은 DATABASE, 서비스 수퍼 클래스(SERVICE CLASS), 서비스 서브클래스(UNDER 절을 지정하는 SERVICE CLASS), WORKLOAD 또는 작업 조치(작업 조치 정의 도메인의 임계값은 CREATE WORK ACTION SET 또는 ALTER WORK ACTION SET 문을 사용하여 작성됨)여야 하고, 시행 범위는 DATABASE PARTITION이어야 합니다(SQLSTATE 5U037). 이 조건은 데이터베이스 파티션마다 독립적으로 시행됩니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.

- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.
- 유형 CALL의 활동. CALL 활동의 경우, 스토어드 프로시저에 대해 추적되는 프로세서 시간에는 하위 활동이나 분리 모드 프로세스에 사용되는 프로세서 시간이 포함되지 않습니다. 임계값 조건은 사용자 논리에서 데이터베이스 엔진으로 리턴될 때만 점검됩니다. 예를 들어, 트러스트된 루틴 실행 중, 임계값 조건은 루틴이 데이터베이스 엔진에 요청을 발행하는 경우에만 점검됩니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. CPUTIME 임계값의 세분화도는 대략 이 숫자에 활동의 병렬 처리 수준을 곱한 값입니다. 예를 들어, 임계값이 60초마다 점검되고 병렬 처리 수준이 2이면 활동은 임계값 위반이 발견되기 전에 1분 대신 여분의 프로세서 시간 2분을 사용할 수 있습니다. 디폴트는 60초입니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

CPUTIMEINSC > *integer-value* **HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS**

이 조건은 특정 서비스 서브클래스에서 실행 중인 동안 특정 데이터베이스 파티션에서 활동에 소비될 수 있는 프로세스 시간에 대한 상한을 정의합니다. 이 임계값에 의해 추적되는 프로세서 시간은 활동이 임계값 도메인에서 식별된 서비스 서브클래스에서 실행을 시작하는 시간부터 측정됩니다. 해당 시점 이전에 사용된 프로세서 시간은 이 임계값에 의해 부과되는 한계에 대해 계산되지 않습니다. 이 값은 0이 아닌 양의 정수가 될 수 있습니다(SQLSTATE 42820).

이 조건에 대한 정의 도메인은 서비스 서브클래스(UNDER 절을 지정하는 SERVICE CLASS)여야 하고, 시행 범위는 DATABASE PARTITION이어야 합니다(SQLSTATE 5U037). 이 조건은 데이터베이스 파티션마다 독립적으로 시행됩니다.

이 조건에 의해 추적되는 활동은 다음과 같습니다.

- 유형 DML의 코디네이터 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은).
- 사용자 논리를 통해 파생되는 중첩된 DML 활동 및 해당되는 서브에이전트 작업(서브섹션 실행과 같은). 유틸리티 또는 프로시저(ADMIN_CMD 프로시저 예외)를 통해 데이터베이스 관리 프로그램이 시작하는 활동은 이 조건에 계산되지 않습니다.

- 제한조건 설정 또는 구체화된 쿼리 테이블 새로 고침으로 시작된 활동과 같은 내부 SQL 활동도 이 임계값의 추적을 받지 않습니다. 데이터베이스 관리 프로그램에 의해 시작되고 사용자 논리에 의해 직접 호출되지 않기 때문입니다.
- 유형 CALL의 활동. CALL 활동의 경우, 스토어드 프로시저에 대해 추적되는 프로세서 시간에는 하위 활동이나 분리 모드 프로세스에 사용되는 프로세서 시간이 포함되지 않습니다. 임계값 조건은 사용자 논리에서 데이터베이스 엔진으로 리턴될 때만 점검됩니다. 예를 들어, 트러스트된 루틴 실행 중, 임계값 조건은 루틴이 데이터베이스 엔진에 요청을 발행하는 경우에만 점검됩니다.

CHECKING EVERY *integer-value* **SECOND | SECONDS**

활동에 대해 임계값 조건을 점검하는 횟수를 지정합니다. CPUTIMEINSC 임계값의 세분화도는 대략 이 숫자에 활동의 병렬 처리 수준을 곱한 값입니다. 예를 들어, 임계값이 60초마다 점검되고 병렬 처리 수준이 2이면 활동은 임계값 위반이 발견되기 전에 1분 대신 여분의 프로세서 시간 2분을 사용할 수 있습니다. 디폴트는 60초입니다. 이 값은 0이 아닌 양의 정수가 될 수 있으며 최대값은 86400초입니다(SQLSTATE 42820). 낮은 값을 설정하면 시스템 성능에 부정적인 영향을 줄 수 있습니다.

threshold-exceeded-actions

조건을 초과할 때 수행할 조치를 지정합니다. 조건을 초과할 때마다 이벤트가 임계값 위반 이벤트 모니터(활성 상태인 경우)에 기록됩니다.

COLLECT ACTIVITY DATA

임계값을 초과한 각 활동에 대한 데이터가 활동 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 디폴트는 COLLECT ACTIVITY DATA NONE입니다. COLLECT ACTIVITY DATA를 지정하는 경우 디폴트는 WITHOUT DETAILS입니다.

NONE

임계값을 초과하는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

ON COORDINATOR DATABASE PARTITION

활동 코디네이터의 데이터베이스 파티션에서만 수집될 활동 데이터를 지정합니다.

ON ALL DATABASE PARTITIONS

활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. 예측적 임계값의 경우, 초과된 임계값에 CONTINUE 조치도 지정하는 경우에만 모든 파티션에서 활동 정보가 수집됩니다. SECTION이 지정된 경우 활동 세부사항 및 섹션 환경도 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우 활동 세부

사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 반응적 임계값의 경우 ON ALL DATABASE PARTITIONS 절은 효과 및 활동, 활동 세부사항, 섹션 정보가 없으며 값은 항상 코디네이터 파티션에서만 수집됩니다.

WITHOUT DETAILS
작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

STOP EXECUTION
활동 실행이 중지되고 오류가 리턴됩니다(SQLSTATE 5U026).

CONTINUE
활동 실행이 중지되지 않습니다.

remap-activity-action

REMAP ACTIVITY TO *service-subclass-name*
활동은 *service-subclass-name*에 맵핑됩니다.활동 실행이 중지되지 않습니다. 이 조치는 CPUTIMEINSC 및 SQLROWSREADINSC 임계값과 같은 in-service-class 임계값에만 유효합니다(SQLSTATE 5U037). service-subclass-name은 임계값과 연관되는 동일한 수퍼 클래스 아래에 있는 기존 서비스 서브클래스를 식별해야 합니다(SQLSTATE 5U037). service-subclass-name은 임계값의 연관된 서비스 서브클래스와 동일할 수 없습니다(SQLSTATE 5U037).

NO EVENT MONITOR RECORD
임계값 위반 레코드가 기록되지 않도록 지정합니다.

LOG EVENT MONITOR RECORD

THRESHOLD VIOLATIONS 이벤트 모니터가 존재하고 활성 상태인 경우 임계값 위반 레코드가 이 이벤트 모니터에 기록되도록 지정합니다.

주

- ***CONTINUE 임계값 초과 조치 및 이벤트 모니터 데이터:*** 이벤트 모니터 데이터는 임계값 조건을 초과했을 때 파티션마다 한 번씩만 수집됩니다. 임계값 초과 조치가 CONTINUE인 경우, 활동은 계속 실행되고 해당 임계값에 대해 영향을 받는 파티션에서 추가 이벤트 모니터 데이터가 수집되지 않습니다. 예를 들어, 조치가 CONTINUE인 10분 시간 임계값을 고려해 보십시오. 활동이 10분 상한을 초과한 후, 임계값에 대해 이벤트 모니터 데이터가 영향을 받는 파티션에서 수집됩니다.
- ***서비스 클래스 Quiesce 진행 중:*** TOTALSCPARTITIONCONNECTIONS 임계값 조건을 사용하여 보통 Quiesce 상태가 될 수 없는 Quiesce 진행 중 서비스 클래스(예: 디폴트 사용자 클래스 또는 디폴트 시스템 클래스)를 시뮬레이트할 수 있습니다. 이는 SYSDEFAULTADMWORKLOAD에서 실행 중인 DBADM 권한을 가지고 있는 사용자에게 임계값이 적용되지 않는 반면 Quiesce 상태의 서비스 클래스는 누구도 사용할 수 없으므로 유용합니다. 결과적으로, 디폴트 서비스 클래스는 직접 Quiesce 상태가 될 수 없지만, SYSDEFAULTADMWORKLOAD를 사용하여 데이터베이스에 연결했을 때 DBADM 권한이 있는 사용자가 조인할 수 있는 임계값을 통해서만 Quiesce 상태가 될 수 있습니다.

예:

예 1: 데이터베이스에서의 활동에 최대 50M의 임시 테이블 스페이스 사용(데이터베이스 파티션마다)을 시행하는 임계값을 작성하십시오. 이 임계값을 위반하는 활동은 중지됩니다.

```
CREATE THRESHOLD DBMAX50MEGTEMPSPACE
  FOR DATABASE ACTIVITIES
  ENFORCEMENT DATABASE PARTITION
  WHEN SQLTEMPSPACE > 50 M
  STOP EXECUTION
```

예 2: 데이터베이스에서 활동의 디폴트 런타임을 최대 1시간으로 제한하기 위한 초 임계값을 작성하십시오. 이 임계값을 위반하는 활동은 중지됩니다.

```
CREATE THRESHOLD DBMAX1HOURRUNTIME
  FOR DATABASE ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN ACTIVITYTOTALTIME > 1 HOUR
  STOP EXECUTION
```

예 3: 평균보다 많은 임시 스페이스를 사용하고 1시간보다 더 오래 실행하는 쿼리를 호스트하기 위해 서비스 수퍼 클래스 BIGQUERIES가 작성되었다고 가정하십시오. 이 서비스 클래스 내에 정의된 임계값은 위에서 데이터베이스 레벨에서 설정된 값보다 우

선합니다. 이 수퍼 클래스 내에서 임계값을 위반하는 활동이 계속 실행될 수 있지만 추가 분석을 위해 자세한 정보가 수집되는 방법에 유의하십시오.

```
CREATE THRESHOLD BIGQUERIESMAX500MEGTEMPSPACE
  FOR SERVICE CLASS BIGQUERIES ACTIVITIES
  ENFORCEMENT DATABASE PARTITION
  WHEN SQLTEMPSPACE  > 500 M
  COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
  CONTINUE

CREATE THRESHOLD BIGQUERIESLONGRUNNINGTIME
  FOR SERVICE CLASS BIGQUERIES ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN ACTIVITYTOTALTIME   > 10 HOURS
  COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
  CONTINUE
```

예 4: 워크로드 PAYROLL이 있다고 가정하고, 워크로드 내에서 최대 활동 수를 10 이하로 하는 임계값을 작성하십시오.

```
CREATE THRESHOLD MAXACTIVITIESINPAYROLL
  FOR WORKLOAD PAYROLL ACTIVITIES
  ENFORCEMENT WORKLOAD OCCURRENCE
  WHEN CONCURRENTWORKLOADACTIVITIES > 10
  STOP EXECUTION
```

예 5: 서비스 클래스 BIGQUERIES에서 최대 두 개의 활동을 동시에 허용하는 임계값을 작성하십시오.

```
CREATE THRESHOLD MAXBIGQUERIESCONCURRENCY
  FOR SERVICE CLASS BIGQUERIES  ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN CONCURRENTDBCOORDACTIVITIES > 2
  STOP EXECUTION
```

CREATE TRANSFORM

CREATE TRANSFORM문은 구조화된 유형 값을 호스트 언어 프로그램, 외부 함수와 교환하는 데 사용되며, 그룹 이름으로 식별되는 변환 함수를 정의합니다.

호출

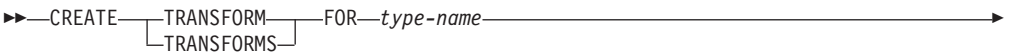
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

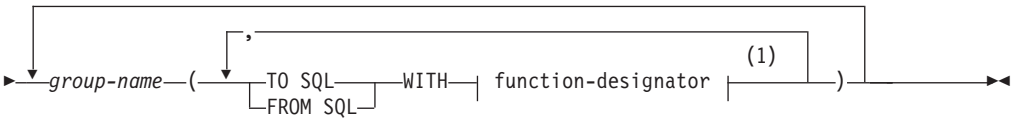
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- *type-name*으로 식별된 유형 소유자 및 지정된 모든 함수의 EXECUTE 특권
- DBADM 권한

구문





주:

1 같은 절은 두 번 이상 지정될 수 없습니다.

설명

TRANSFORM 또는 TRANSFORMS

하나 이상의 변환 그룹이 정의되고 있음을 나타냅니다. 키워드 버전 중 하나를 지정할 수 있습니다.

FOR *type-name*

변환 그룹이 정의될 사용자 정의 구조화된 유형의 이름을 지정합니다.

동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 *type-name*의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 *type-name*의 규정자를 내재적으로 지정합니다. *type-name*은 기존의 사용자 정의 유형의 이름이거나(SQLSTATE 42704), 구조화된 유형의 이름이어야

합니다(SQLSTATE 42809). 구조화된 유형이나 같은 자료형 계층에 있는 다른 구조화된 유형의 경우, 제공된 그룹 이름으로 이미 정의되어 있는 변환을 가질 수 없습니다(SQLSTATE 42739).

group-name

변환 그룹 이름. 이 이름은 한 부분의 이름입니다. 이것은 SQL ID(일반 또는 분리 ID)입니다. *group-name*은 지정된 *type-name*에 대한 카탈로그에 이미 존재하는 변환 그룹을 식별합니다(SQLSTATE 42739). *group-name*은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939). 대부분의 경우 FROM SQL 및 TO SQL 함수 각각을 제공된 그룹에 대해 지정할 수 있습니다(SQLSTATE 42628).

TO SQL

SQL 사용자 정의 구조화된 유형 형식으로 값을 변환하는 데 사용되는 특정 함수를 정의합니다. 이 함수는 모든 매개변수를 내장 데이터 유형으로 가져야 하며 리턴된 유형은 *type-name*입니다.

FROM SQL

SQL 사용자 정의 구조화된 유형을 나타내는 데이터 유형으로 값을 변환하는 데 사용되는 특정 함수를 정의합니다. 이 함수는 데이터 유형 *type-name*의 매개변수를 한 개 가져야 하며 내장 데이터 유형(또는 내장 데이터 유형 세트)을 리턴해야 합니다.

WITH *function-designator*

변환 함수를 고유하게 식별합니다.

FROM SQL이 지정된 경우, *function-designator*는 다음 조건에 따른 함수를 식별해야 합니다.

- *type-name* 유형에 대해 하나의 매개변수가 있습니다.
- 리턴 유형이 내장 유형이거나 행이 내장 유형을 모두 가진 컬럼입니다.
- 시그니처는 LANGUAGE SQL 또는 LANGUAGE SQL을 가지는 또 다른 FROM SQL 변환 함수의 사용을 지정합니다.

TO SQL이 지정된 경우, *function-designator*는 다음 조건에 따른 함수를 식별해야 합니다.

- 모든 매개변수는 내장 유형을 갖습니다.
- 리턴 유형이 *type-name*입니다.
- 시그니처는 LANGUAGE SQL 또는 LANGUAGE SQL을 가지는 또 다른 TO SQL 변환 함수의 사용을 지정합니다.

*function-designator*가 FROM SQL이나 TO SQL 변환 함수로 사용하는 것에 따른 요구사항을 충족하지 않는 함수를 식별하는 경우에는 오류가 발생합니다(SQLSTATE 428DC).

메소드(FUNCTION ACCESS로 지정되어도)는 *function-designator*를 통해 변환으로 지정할 수 없습니다. 대신 CREATE FUNCTION문에서 정의된 함수만이 변환으로 동작할 수 있습니다(SQLSTATE 42704 또는 42883).

자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』를 참조하십시오.

규칙

- FROM SQL 함수에서 리턴된 하나 이상의 내장 유형은 TO SQL 함수의 매개변수인 하나 이상의 내장 유형에 직접적으로 해당되어야 합니다. 이는 이러한 두 함수 간의 역관계에 대한 논리적 결과입니다.

주

- 변환 그룹이 응용프로그램에 지정되지 않은 경우(정적 SQL의 경우 TRANSFORM GROUP 프리컴파일 또는 바인드 옵션 사용, 동적 SQL의 경우 SET CURRENT DEFAULT TRANSFORM GROUP문 사용), 응용프로그램이 *type-name*으로 식별된 사용자 정의 구조화된 유형을 기반으로 하는 호스트 변수를 검색하거나 보내는 중이면 변환 그룹 'DB2_PROGRAM'의 변환 함수(정의된 경우)가 사용됩니다. 데이터 유형 *type-name*의 값을 검색할 경우, FROM SQL 변환을 호출하여 구조화된 유형을 변환 함수가 리턴하는 내장 데이터 유형으로 변환합니다. 마찬가지로 데이터 유형 *type-name*의 값이 지정되는 호스트 변수를 보낼 때, TO SQL 변환을 호출하여 내장 데이터 유형 값을 구조화된 유형 값으로 변환합니다. 사용자 정의된 변환 그룹이 지정되지 않거나 'DB2_PROGRAM' 그룹이 정의되지 않으면(주어진 구조화된 유형 주제에 대해), 오류가 발생합니다(SQLSTATE 42741).
- 구조화된 유형 호스트 변수에 대한 내장 데이터 유형 표현을 지정할 수 있어야 합니다.
  - 프리컴파일 명령(검색 지정 규칙 사용)의 지정된 TRANSFORM GROUP 옵션에 정의된 대로 구조화된 유형에 대한 FROM SQL 변환 함수의 결과로부터
  - 프리컴파일 명령(스토리지 지정 규칙 사용)의 지정된 TRANSFORM GROUP 옵션에 정의된 대로 구조화된 유형에 대한 TO SQL 변환 함수의 매개변수로

  호스트 변수가 해당 변환 함수가 요구하는 유형과 지정 호환 가능하지 않을 경우, 오류가 발생합니다(바인드 인의 경우 SQLSTATE 42821, 바인드 아웃의 경우 SQLSTATE 42806). 문자열 지정으로 인해 발생한 오류에 대해서는 『문자열 지정』을 참조하십시오.
- 디폴트 변환 그룹 'DB2_FUNCTION'에서 식별된 변환 함수는 SQL로 작성되지 않은 사용자 정의 함수가 매개변수 또는 리턴 유형으로 데이터 유형 *type-name*을 사용하여 호출될 때마다 사용됩니다. 이는 함수가 TRANSFORM GROUP절을 지정하지 않을 때 적용됩니다. 데이터 유형 *type-name*의 인수로 함수를 호출하는 경우, FROM SQL 변환을 실행하여 구조화된 유형을 변환 함수가 리턴하는 내장 데이터 유형으로 변환합니다. 마찬가지로, 함수의 리턴 데이터 유형이 데이터 유형 *type-name*

인 경우, TO SQL 변환을 호출하여 외부 함수 프로그램으로부터 리턴된 내장 데이터 유형을 구조화된 유형 값으로 변환합니다.

- 구조화된 유형에 구조화된 유형이기도 한 속성이 포함되는 경우, 연관되는 변환 함수가 재귀적으로 중첩된 모든 구조화된 유형을 확장해야 합니다. 이는 변환 함수의 결과나 매개변수가 주제 구조화된 유형의 모든 기본 속성을 나타내는 내장 유형 세트만으로 구성됨을 의미합니다(중첩된 모든 구조화된 유형 포함). 중첩된 구조화된 유형을 처리하기 위한 변환 함수의 "연쇄" 작용은 없습니다.
- 이 명령문에서 식별된 함수는 이 명령문의 실행 시 위에 대략적으로 설명된 규칙에 따라 해석됩니다. 이러한 함수가 후속 SQL문에서 (내재적으로) 사용되면 다른 해석 프로세스를 수행하지 않습니다. 이 명령문에 정의된 변환 함수는 이 명령문에서 해석된 대로 정확하게 기록됩니다.
- 제공된 유형의 속성이나 부속 유형이 작성되거나 삭제될 경우, 사용자 정의 구조화된 유형에 대한 변환 함수도 변경해야 합니다.
- 제공된 변환 그룹에 대해 FROM SQL 및 TO SQL 변환을 같은 *group-name*절, 별도의 *group-name*절 또는 별도의 CREATE TRANSFORM문에 지정할 수 있습니다. 먼저 기존의 그룹 정의를 삭제하지 않고는 제공된 FROM SQL 또는 TO SQL 변환 지정을 재정의할 수 없다는 것이 유일한 제한사항입니다. 이를 통해 나중에 먼저 제공된 그룹의 FROM SQL 변환 및 같은 그룹의 해당 TO SQL 변환을 정의할 수 있습니다.

예:

예 1: 사용자 정의된 구조화 유형 다각형을 C 및 Java에 맞게 사용자 정의된 변환 함수와 각각 연관시키는 두 개의 변환 그룹을 작성하십시오.

```
CREATE TRANSFORM FOR POLYGON
       mystruct1 (FROM SQL WITH FUNCTION myxform_sqlstruct,
             TO SQL WITH FUNCTION myxform_structsql)
       myjava1   (FROM SQL WITH FUNCTION myxform_sqljava,
             TO SQL WITH FUNCTION myxform_javasql)
```

CREATE TRIGGER

CREATE TRIGGER문은 데이터베이스에서 트리거를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- BEFORE 또는 AFTER 트리거가 정의된 테이블에 대한 ALTER 특권
- INSTEAD OF TRIGGER가 정의된 뷰에 대한 CONTROL 특권
- INSTEAD OF 트리거가 정의되어 있는 뷰에서의 소유자
- 테이블 스키마 또는 트리거가 정의되어 있는 뷰에 대한 ALTERIN 특권
- DBADM 권한

다음 중 하나

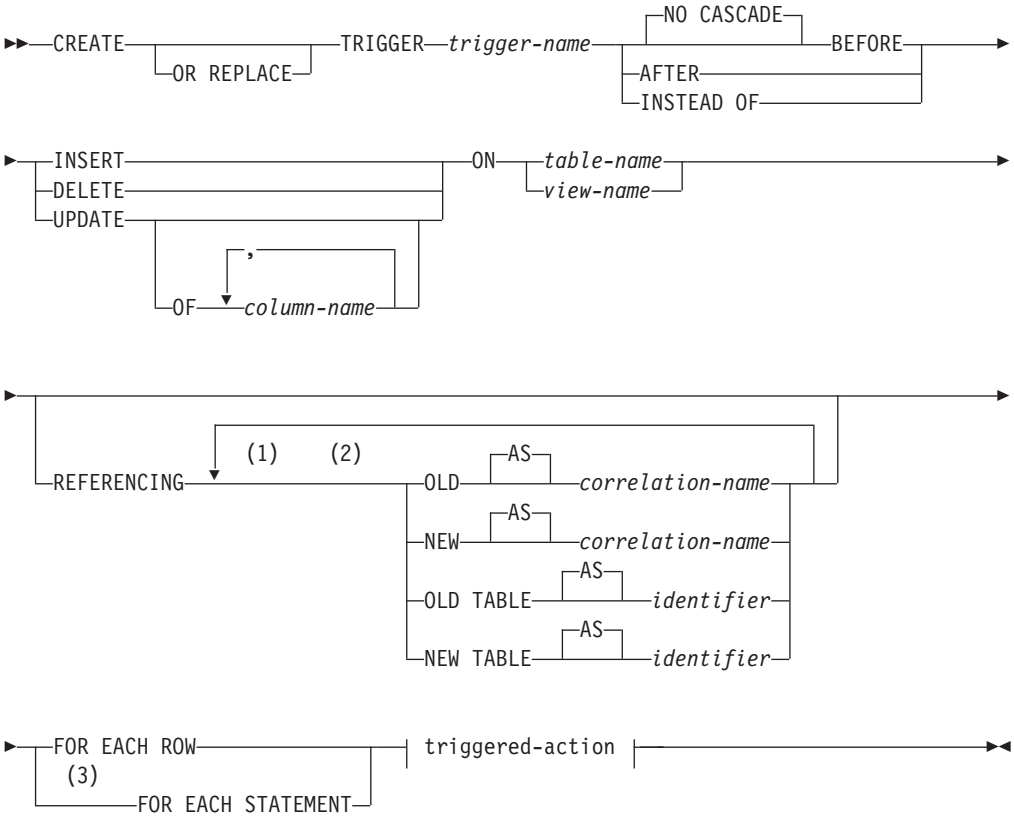
- 트리거의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 트리거의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

명령문의 권한 부여 ID에 DATAACCESS 권한이 없는 경우에는 트리거가 존재하는 한 명령문의 권한 부여 ID가 보유한 특권(그룹 특권 제외)에는 다음이 모두 포함되어야 합니다.

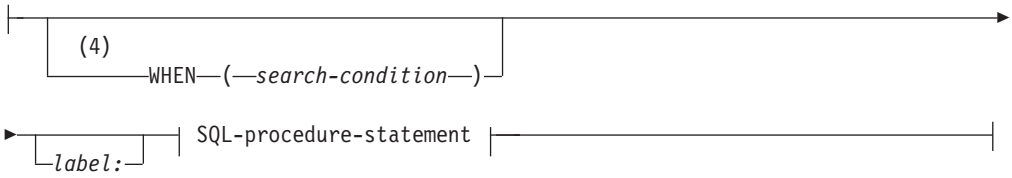
- 전이 변수나 테이블이 지정된 경우, 트리거가 정의되는 테이블에서,
  - 전이 변수나 테이블이 지정된 경우, 트리거가 정의되는 테이블에 대한 SELECT 특권
  - 전이 변수 또는 테이블이 지정된 경우 트리거가 정의된 테이블에서 CONTROL 특권
  - DATAACCESS 권한
- 트리거된 조치 조건에서 참조되는 모든 테이블 또는 뷰에서,
  - 트리거되는 조치 조건에서 참조되는 테이블이나 뷰에 대한 SELECT 특권
  - 트리거되는 조치 조건에서 참조되는 테이블이나 뷰에 대한 CONTROL 특권
  - DATAACCESS 권한
- 지정된 트리거 SQL문을 호출하는 필수 특권.

기존 트리거를 교체하려면 명령문의 권한 부여 ID가 기존 트리거의 소유자여야 합니다 (SQLSTATE 42501).

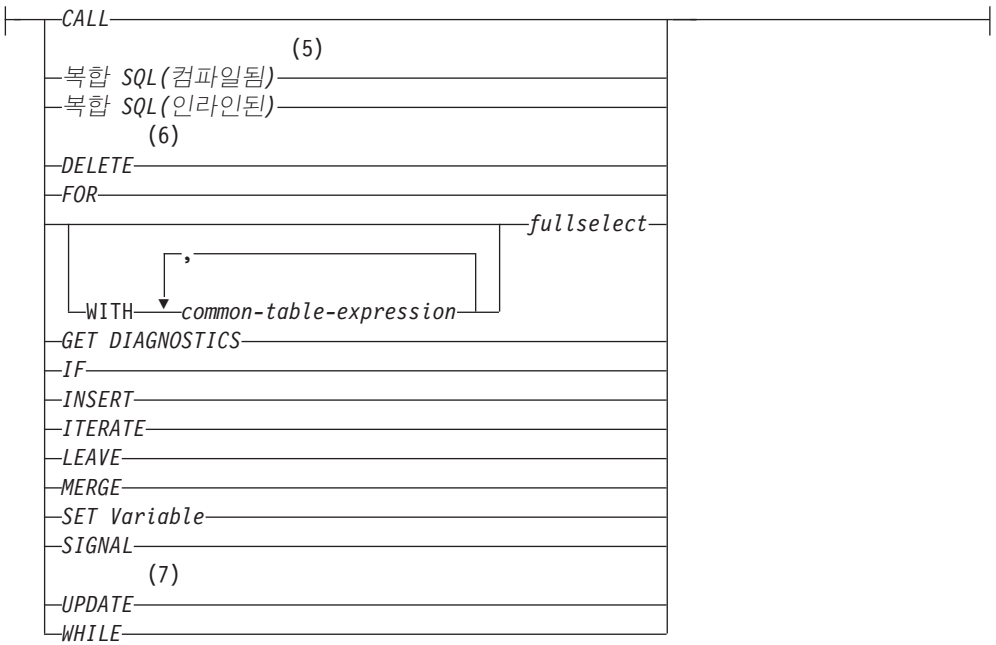
구문



triggered-action:



SQL-procedure-statement:



주:

1 OLD 및 NEW는 각각 한 번만 지정될 수 있습니다.

2 OLD TABLE 및 NEW TABLE은 AFTER 트리거 또는 INSTEAD OF 트리거에 대해서만 그리고 각각 한 번씩만 지정될 수 있습니다.

3 FOR EACH STATEMENT는 BEFORE 트리거 또는 INSTEAD OF 트리거에 대해 지정할 수 없습니다.

4 WHEN 조건은 INSTEAD OF 트리거에 대해 지정할 수 없습니다.

5 트리거 정의가 REFERENCING OLD TABLE 절, REFERENCING NEW TABLE 절 또는 FOR EACH STATEMENT 절을 포함하는 경우 복합 SQL(컴파일됨) 명령문은 지정될 수 없습니다. 복합 SQL(컴파일됨) 명령문도 파티션된 데이터베이스 환경에 있는 트리거 정의에 지정될 수 없습니다.

6 이 컨텍스트에서 *searched-delete*만 지원됩니다.

7 이 컨텍스트에서 *searched-update*만 지원됩니다.

설명

OR REPLACE

현재 서버에 트리거가 존재하면 트리거의 정의를 교체하도록 지정합니다. 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 삭제됩니다. 트리거의 정의가 현재 서버에 존재하지 않으면 이 옵션이 무시됩니다.

trigger-name

트리거의 이름을 지정합니다. 내재적 또는 명시적 스키마 이름을 포함하는 이름은

카탈로그에 기술된 테이블, 뷰 또는 별명을 식별해서는 안됩니다. 두 부분으로 구성된 이름이 지정된 경우, 스키마 이름은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

NO CASCADE BEFORE

주제 테이블의 실제 갱신으로 인한 변경사항이 데이터베이스에 적용되기 전에 트리거된 연관 조치가 적용되도록 지정합니다. 또한 트리거의 트리거된 조치로 인해 다른 트리거가 활성화되지 않도록 지정합니다.

AFTER

주제 테이블의 실제 갱신으로 변경사항이 데이터베이스에 적용된 후에 트리거된 연관 조치가 적용되도록 지정합니다.

INSTEAD OF

주제 뷰에 대한 조치를 연관된 트리거 조치로 바꾸도록 지정합니다. 해당 주제 뷰에 대한 각 조작의 종류에 대하여 하나의 INSTEAD OF 트리거만 허용됩니다.

INSERT

INSERT 조작이 주제 테이블이나 주제 뷰에 적용될 때마다 트리거와 연관된 트리거 조치가 실행되도록 지정합니다.

DELETE

DELETE 조작이 주제 테이블이나 주제 뷰에 적용될 때마다 트리거와 연관된 트리거 조치가 실행되도록 지정합니다.

UPDATE

UPDATE 조작이 지정되거나 내재된 컬럼에 따라 주제 테이블이나 주제 뷰에 적용될 때마다 트리거와 연관된 트리거 조치가 실행되도록 지정합니다.

선택적 *column-name* 목록을 지정하지 않으면, 테이블의 모든 컬럼이 내재됩니다. 그러므로 *column-name* 목록을 생략하면, 테이블의 컬럼에 의한 갱신에 의해 트리거가 활성화됩니다.

OF *column-name*,...

지정된 각 *column-name*은 기본 테이블의 컬럼이어야 합니다(SQLSTATE 42703). 트리거가 BEFORE 트리거인 경우 지정된 *column-name*은 식별 컬럼이 아닌 생성된 컬럼일 수 없습니다(SQLSTATE 42989). *column-name*은 *column-name* 목록에서 두 번 이상 나타날 수 없습니다(SQLSTATE 42711). 트리거는 *column-name* 목록에서 식별되는 컬럼의 갱신에 의해서만 활성화됩니다. 이 절은 INSTEAD OF 트리거에 대해서는 지정될 수 없습니다(SQLSTATE 42613).

ON

table-name

BEFORE 트리거 또는 AFTER 트리거 정의의 주제 테이블을 지정합니다. 이름은 기본 테이블을 해석하는 기본 테이블이나 별명을 지정해야 합니다

(SQLSTATE 42704 또는 42809). 이름은 카탈로그 테이블(SQLSTATE 42832), 구체화된 쿼리 테이블(SQLSTATE 42997), 작성된 임시 테이블, 선언된 임시 테이블(SQLSTATE 42995) 또는 별칭(SQLSTATE 42809)을 지정할 수 없습니다.

view-name

INSTEAD OF 트리거 정의의 주제 테이블을 지정합니다. 이 이름은 XML 유형의 컬럼 없이 유형이 지정되지 않은 뷰를 해석하는 별명을 지정하거나 유형이 지정되지 않은 뷰를 지정해야 합니다(SQLSTATE 42704 또는 42809). 이 이름은 카탈로그 테이블을 지정할 수 없습니다(SQLSTATE 42832). 이 이름은 WITH CHECK OPTION을 사용하여 정의된 뷰(대칭 뷰)나 직/간접적으로 대칭 뷰가 정의된 뷰를 지정해서는 안됩니다(SQLSTATE 428FQ).

REFERENCING

전이 변수 및 *전이 테이블*에 대한 테이블 이름에 대해 상관 이름을 지정합니다. 상관 이름은 트리거링 SQL 조작에 의해 영향을 받는 행 세트로 특정 행을 식별합니다. 테이블 이름은 영향을 받는 행의 완전한 세트를 식별합니다. 트리거링 SQL 조작에 의해 영향을 받는 각 행은 다음과 같이 지정된 *correlation-name*을 갖는 컬럼을 규정하여 트리거된 조치에 사용할 수 있습니다.

OLD AS *correlation-name*

트리거링 SQL 조작 이전의 행 상태를 식별하는 상관 이름을 지정합니다.

NEW AS *correlation-name*

트리거링 SQL 조작에 의해, 그리고 이미 실행된 BEFORE 트리거의 SET문에 의해 수정된 대로 행 상태를 식별하는 상관 이름을 지정합니다.

NEW AS 상관 이름은 XML 유형일 수 있습니다.

트리거링 SQL 조작에 의해 영향을 받는 행들의 완전한 세트는 다음과 같이 지정된 임시 테이블 이름을 사용하여 트리거된 조치에 사용할 수 있습니다.

OLD TABLE AS *identifier*

트리거링 SQL 조작 이전의 영향을 받는 행 세트를 식별하는 임시 테이블 이름을 지정합니다.

NEW TABLE AS *identifier*

트리거링 SQL 조작에 의해, 그리고 이미 실행된 BEFORE 트리거의 SET문에 의해 수정된 대로 영향을 받는 행을 식별하는 임시 테이블 이름을 지정합니다.

다음 규칙이 REFERENCING절에 적용됩니다.

- OLD 및 NEW 상관 이름과 OLD TABLE 및 NEW TABLE 이름 중 어느 것도 동일할 수 없습니다(SQLSTATE 42712).
- 트리거에 대해 단 하나의 OLD 및 NEW *correlation-name*을 지정할 수 있습니다(SQLSTATE 42613).

- 트리거에 대해 단 하나의 OLD TABLE 및 하나의 NEW TABLE *identifier*를 지정할 수 있습니다(SQLSTATE 42613).
- OLD *correlation-name* 및 OLD TABLE *identifier*는 트리거 이벤트가 DELETE 조작 또는 UPDATE 조작일 때만 사용할 수 있습니다(SQLSTATE 42898). 조작이 DELETE 조작이면, OLD *correlation-name*은 삭제된 행의 값을 캡처합니다. 이것이 UPDATE 조작이면, UPDATE 조작 전에 행의 값을 캡처합니다. OLD TABLE *identifier* 및 영향을 받는 행 세트에도 동일하게 적용됩니다.
- NEW *correlation-name* 및 NEW TABLE *identifier*는 트리거 이벤트가 INSERT 조작 또는 UPDATE 조작일 경우에만 사용할 수 있습니다(SQLSTATE 42898). 두 조작 모두에서 NEW 값은 원래 조작에서 제공된 대로, 이 시점까지 실행된 BEFORE 트리거에 의해 수정된 대로 행의 새로운 상태를 캡처합니다. NEW TABLE *identifier* 및 영향을 받는 행 세트에도 동일하게 적용됩니다.
- OLD TABLE 또는 NEW TABLE ID는 BEFORE 트리거에서 정의될 수 없습니다(SQLSTATE 42898).
- NEW 전이 변수는 AFTER 트리거에서 정의될 수 없습니다(SQLSTATE 42987).
- OLD 또는 NEW 상관 이름은 FOR EACH STATEMENT 트리거에서 정의될 수 없습니다(SQLSTATE 42899).
- 전이 테이블은 수정할 수 없습니다(SQLSTATE 42807).
- 전이 테이블 컬럼에 대한 총 참조 수와 트리거 조치 내의 전이 변수의 총 수는 테이블 내 컬럼 수에 대한 한계를 초과할 수 없고, 그 길이의 합이 테이블 내 행의 최대 길이를 초과할 수 없습니다(SQLSTATE 54040).
- 각 *correlation-name*과 각 *identifier*의 범위는 전체 트리거 정의입니다.

FOR EACH ROW

트리거된 조치가 트리거링 SQL 조작에 의해 영향을 받는 주제 테이블 또는 주제 뷰의 각 행에 대해 한 번 적용되도록 지정합니다.

FOR EACH STATEMENT

트리거된 조치가 전체 명령문에 대해 한 번만 적용되도록 지정합니다. 이 트리거 수준 유형은 BEFORE 트리거 또는 INSTEAD OF 트리거에 대해 지정할 수 없습니다(SQLSTATE 42613). 지정할 경우, UPDATE 또는 DELETE문을 트리거하여 영향을 받는 행이 없더라도, UPDATE 또는 DELETE 트리거가 활성화됩니다.

triggered-action

트리거가 활성화될 때 수행될 조치를 지정합니다. 트리거 조치는 *SQL-procedure-statement* 및 *SQL-procedure-statement*의 실행에 대한 선택적 조건으로 구성됩니다.

WHEN

(*search-condition***)**

참, 거짓 또는 알 수 없음 조건을 지정합니다. *search-condition*은 특정 트

리거 조치가 실행되어야 하는지 여부를 판별할 수 있는 능력을 제공합니다. 연관 조치는 지정된 검색 조건이 참으로 평가될 경우에만 수행됩니다. WHEN절을 생략하면, 연관된 *SQL-procedure-statement*가 항상 수행됩니다.

WHEN절은 INSTEAD OF 트리거에 대해 지정할 수 없습니다(SQLSTATE 42613).

XML 데이터 유형인 전이 변수에 대한 참조는 VALIDATED 술어에서만 사용할 수 있습니다.

label:

SQL 프로시저 명령문에 대해 레이블을 지정합니다. 레이블은 SQL 프로시저 명령문 목록 내에서 고유해야 합니다(이 목록에 중첩된 복합 명령문 포함). 중첩되지 않은 복합 명령문은 같은 레이블을 사용할 수도 있음에 유의하십시오. SQL 프로시저 목록은 수많은 SQL 제어 명령문에서 가능합니다.

FOR문, WHILE문 및 복합 SQL문만 레이블을 포함할 수 있습니다.

SQL-procedure-statement

트리거된 조치의 일부분인 SQL문을 지정합니다. 복합 SQL 내에서 별칭에 대한 검색 갱신, 검색 삭제, 삽입 및 병합은 지원되지 않습니다.

XML 유형의 컬럼에 있는 BEFORE 트리거의 트리거 조치는 SET문을 통해 XMLVALIDATE 함수를 호출할 수 있으며, XML 유형의 값을 변경하지 않고 그대로 두거나 SET문을 사용하여 널(NULL)로 지정합니다.

*SQL-procedure-statement*는 지원되지 않는 명령문을 포함하지 않아야 합니다(SQLSTATE 42987).

*SQL-procedure-statement*는 정의되지 않은 전이 변수(SQLSTATE 42703), 페더레이티드 오브젝트(SQLSTATE 42997) 또는 선언된 임시 테이블(SQLSTATE 42995)을 참조할 수 없습니다.

BEFORE 트리거의 *SQL-procedure-statement*는 다음을 수행할 수 없습니다.

- MODIFIES SQL DATA 또는 MERGE문으로 정의된 프로시저를 호출하는 CALL문일 수 없습니다(SQLSTATE 42987).
- REFRESH IMMEDIATE로 정의된 구체화된 쿼리 테이블을 참조할 수 없습니다(SQLSTATE 42997).
- NEW 전이 변수의 식별 컬럼이 아닌 생성된 컬럼을 참조할 수 없습니다(SQLSTATE 42989).

주

- 이미 행을 갖고 있는 테이블에 트리거를 추가하면 트리거된 조치가 활성화되지 않습니다. 그러므로 트리거가 테이블의 데이터에 대해 제한조건을 시행하도록 설계된 경우, 기존 행에서는 이 제한조건이 충족되지 않을 수도 있습니다.

- 두 트리거에 대한 이벤트가 동시에 발생하는 경우 즉, 두 트리거의 이벤트, 활동 시간 및 주제 테이블이 동일한 경우 먼저 작성된 트리거부터 실행됩니다. OR REPLACE 옵션이 사전에 작성된 트리거를 교체하는 데 사용되는 경우 작성 시간이 변경되므로 트리거 실행 순서에 영향을 줄 수 있습니다.
- 트리거가 정의된 이후에 주제 테이블에 컬럼이 추가되면, 다음 규칙이 적용됩니다.
  - 트리거가 명시적 컬럼 목록없이 지정된 UPDATE 트리거인 경우, 새 컬럼을 갱신하면 트리거가 활성화됩니다.
  - 이전에 정의된 트리거의 트리거 조치에 컬럼이 표시되지 않습니다.
  - OLD TABLE 및 NEW TABLE 전이 테이블에는 이 컬럼이 포함되지 않습니다. 그러므로 전이 테이블에서 ″SELECT *″의 수행 결과에는 추가된 컬럼이 포함되지 않습니다.
- 컬럼이 트리거 조치에서 참조된 테이블에 추가되면, 새 컬럼은 트리거 조치에서 볼 수 없게 됩니다.
- 트리거 내용에서 참조된 오브젝트가 존재하지 않거나 유효하지 않은 것으로 표시되거나 정의자에 임시로 오브젝트 액세스 특권이 없고 데이터베이스 구성 매개변수 **auto_reval**이 DISABLED로 설정되어 있지 않으면, 트리거가 여전히 성공적으로 작성됩니다. 트리거는 유효하지 않은 것으로 표시되며 다음에 호출될 때 유효성을 다시 확인합니다.
- *SQL-procedure-statement*에 지정된 fullselect 결과는 트리거 외부나 내부에서 사용할 수 없습니다.
- 트리거된 복합 명령문 내에서 호출된 프로시저는 COMMIT 또는 ROLLBACK문을 발행할 수 없습니다(SQLSTATE 42985).
- 검색 UPDATE문, 검색 DELETE문 또는 INSERT문에 있는 별칭에 대한 참조가 포함된 프로시저는 지원되지 않습니다(SQLSTATE 25000).
- ***테이블 액세스 제한사항***: 프로시저가 READS SQL DATA 또는 MODIFIES SQL DATA로 정의되어 있으면, 프로시저의 어떤 명령문도 프로시저를 호출한 복합 명령문에서 수정 중인 테이블에 액세스할 수 없습니다(SQLSTATE 57053). 프로시저가 MODIFIES SQL DATA로 정의되어 있으면, 프로시저의 어떤 명령문도 프로시저를 호출한 복합 명령문에서 읽거나 수정 중인 테이블을 수정할 수 없습니다(SQLSTATE 57053).
- 연쇄 참조 제한조건의 주기에 포함된 테이블에 정의된 BEFORE DELETE 트리거는 이것이 정의된 테이블이나 참조 무결성 제한조건의 주기 평가시 연쇄적으로 수정된 다른 테이블을 참조하면 안됩니다. 그러한 트리거의 결과는 데이터에 의존하므로, 일관된 데이터를 작성하지 못할 수도 있습니다.

  가장 간단한 양식에서, 이는 자체 참조 참조 제한사항이 있는 테이블의 BEFORE DELETE 트리거와 CASCADE의 삭제 규칙에서 *triggered-action*의 테이블을 참조하면 안 된다는 것을 의미합니다.

- 트리거를 작성하면 특정 패키지가 유효하지 않은 것으로 표시됩니다.
  - 명시적인 컬럼 목록없이 UPDATE 트리거를 작성하면, 목표 테이블 또는 뷰에서 갱신 사용을 갖는 패키지가 무효화됩니다.
  - 컬럼 목록과 함께 UPDATE 트리거를 작성하면, CREATE TRIGGER문의 *column-name* 목록에 있는 최소한 하나의 컬럼에서 패키지가 갱신을 수행할 수 있는 경우 목표 테이블에서 갱신을 수행할 수 없는 패키지만 무효화됩니다.
  - INSERT 트리거가 작성되면, 목표 테이블 또는 뷰에서 삽입을 수행할 수 있는 패키지가 무효화됩니다.
  - 삭제 트리거가 작성되면, 목표 테이블 또는 뷰에서 삭제를 수행할 수 있는 패키지가 무효화됩니다.
- 응용프로그램이 명시적으로 바인딩되거나 리바인딩될 때까지 또는 실행된 후 데이터베이스 관리 프로그램에 의해 자동으로 리바인딩될 때까지 패키지는 유효하지 않은 상태로 유지됩니다.
- ***작동 불능 트리거***: *작동 불능 트리거*는 더 이상 사용할 수 없고 따라서 활성화되지 않는 트리거입니다. 다음의 경우에 트리거는 작동 불능 상태가 됩니다.
  - 트리거 작성자가 트리거를 실행하는 데 필요한 특권이 취소될 경우
  - 테이블, 뷰 또는 별명과 같이 트리거 조치에 의존하는 오브젝트가 삭제되는 경우
  - 트리거 조치에 종속되는 뷰가 작동 불능이 되는 경우
  - 트리거의 주제 테이블 별명이 삭제된 경우

  실제 용어에서, 작동 불능 트리거는 DROP 또는 REVOKE문에 대한 연쇄 규칙의 결과로 트리거 정의가 삭제된 트리거입니다. 예를 들어, 뷰가 삭제되면, 그 뷰에 대한 참조를 포함하는 *SQL-procedure-statement*가 있는 트리거는 작동 불능 상태가 됩니다.

  트리거가 작동 불능 상태이면, 트리거를 활성화했던 조작을 수행하는 명령문을 갖는 모든 패키지가 유효하지 않은 것으로 표시됩니다. 패키지가 명시적 또는 내재적으로 리바인드되면, 작동 불능 트리거는 완전히 무시됩니다. 마찬가지로, 트리거를 활성화했던 조작을 수행하는 동적 SQL문을 갖는 응용프로그램도 작동 불능 트리거를 완전히 무시합니다.

  트리거 이름은 계속해서 DROP TRIGGER 및 COMMENT ON TRIGGER문에 지정할 수 있습니다.

  작동 불능 트리거는 작동 불능 트리거의 정의 텍스트를 사용하는 CREATE TRIGGER문을 발행하여 재작성할 수 있습니다. 이 트리거 정의 텍스트는 SYSCAT.TRIGGERS 카탈로그 뷰의 TEXT 컬럼에 저장됩니다. 작동 불능 트리거를 재작성하기 위해 이를 명시적으로 삭제할 필요는 없습니다. CREATE TRIGGER

문을 작동 불능 트리거와 동일한 *trigger-name*과 함께 발행하면, 작동 불능 트리거가 경고와 함께 대체됩니다(SQLSTATE 01595).

작동 불능 트리거는 SYSCAT.TRIGGERS 카탈로그 뷰의 VALID 컬럼에서 X로 표시됩니다.

- ***트리거 실행 중 오류***: 트리거 SQL문 실행 중에 발생하는 오류는 심각한 오류가 아닌 경우, SQLSTATE 09000을 사용하여 리턴됩니다. 오류가 심각하면, 심각한 오류 SQLSTATE가 리턴됩니다. 심각하지 않은 오류에 대한 SQLCA의 SQLERRMC 필드에는 트리거 이름, SQLCODE, SQLSTATE, 그리고 실패한 토큰으로부터 맞출 수 있는 만큼의 토큰이 포함됩니다.

  *SQL-procedure-statement*는 SIGNAL SQLSTATE문 또는 RAISE_ERROR 함수를 포함할 수 있습니다. 두 가지 경우 모두, 리턴된 SQLSTATE는 SIGNAL SQLSTATE문 또는 RAISE_ERROR 조건에서 지정된 것입니다.
- 아직 존재하지 않는 스키마 이름을 사용하여 트리거를 작성하면, 명령문의 권한 부여 ID에 IMPLICIT_SCHEMA 권한이 있을 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- BEFORE 트리거의 실행 전에 데이터베이스 관리 프로그램이 식별 컬럼에 대한 값을 생성합니다. 따라서 생성된 식별 값은 BEFORE 트리거에 가시적입니다.
- ROW CHANGE TIMESTAMP 컬럼에 대해 데이터베이스 관리 프로그램에서 생성된 값은 모든 BEFORE 트리거의 실행 후에 생성됩니다. 그러므로 ROW CHANGE TIMESTAMP 값은 BEFORE 트리거를 볼 수 없습니다.
- 모든 BEFORE 트리거의 실행 후에 데이터베이스 관리 프로그램이 표현식 컬럼에 대한 값을 생성합니다. 따라서 표현식으로 생성된 값은 BEFORE 트리거에 가시적이 아닙니다.
- ***읽기 전용 뷰***: 뷰에 대한 INSTEAD OF 트리거 추가는 뷰의 읽기 전용 특성에 영향을 줍니다. 읽기 전용 뷰에 INSTEAD OF 트리거를 포함한 종속성 관계가 있으면, INSTEAD OF 트리거에 대해 정의된 조작 유형은 뷰가 삭제, 삽입 또는 갱신 가능한지 여부를 정의합니다.
- ***전이 변수 및 INSTEAD OF 트리거***: INSTEAD OF INSERT 트리거에서 볼 수 있는 새 전이 테이블 컬럼 또는 새 전이 변수의 초기 값은 다음과 같이 설정됩니다.
  - 삽입 조작의 컬럼에 대해 값이 명시적으로 지정되면, 해당되는 새 전이 변수는 명시적으로 지정된 값이 됩니다.
  - 삽입 조작의 컬럼에 대해 값이 명시적으로 지정되지 않거나 DEFAULT절이 지정되면, 해당되는 새 전이 변수는 다음과 같습니다.
    - 뷰 컬럼이 갱신 가능한 경우(INSTEAD OF 트리거 없음) 기본 테이블 컬럼의 디폴트 값

- 그 외의 경우 널(NULL) 값

INSTEAD OF UPDATE 트리거에서 볼 수 있는 새 전이 변수에 대한 초기 값은 다음과 같이 설정됩니다.

– 갱신 조작의 컬럼에 대해 값이 명시적으로 지정되면, 해당되는 새 전이 변수는 명시적으로 지정된 값이 됩니다.
– 갱신 조작의 컬럼에 대해 DEFAULT절이 명시적으로 지정되면, 해당되는 새 전이 변수는 다음과 같습니다.
  - 뷰 컬럼이 갱신 가능한 경우(INSTEAD OF 트리거 없음) 기본 테이블 컬럼의 디폴트 값
  - 그 외의 경우 널(NULL) 값
– 그 외의 경우 해당되는 새 전이 변수는 행의 컬럼의 기존 값이 됩니다.

• *트리거 및 유형이 지정된 테이블*: 테이블 계층의 모든 레벨에서 BEFORE과 AFTER 트리거를 유형이 지정된 테이블에 첨부할 수 있습니다. SQL문이 여러 트리거를 활성화하는 경우, 트리거는 유형이 지정된 테이블 계층 내의 여러 다른 테이블에 첨부되더라도 작성 순서대로 실행됩니다.

  트리거가 활성화되는 경우, 트리거의 전이 변수(OLD, NEW, OLD TABLE 및 NEW TABLE)에 서브테이블 행이 포함될 수도 있습니다. 그러나 첨부된 테이블에 정의된 컬럼만 포함됩니다.

  INSERT, UPDATE 및 DELETE문의 영향:

  – 행 트리거: SQL문을 사용하여 테이블 행을 삽입, 갱신 또는 삭제하는 경우, 이 SQL문은 해당 행을 포함하는 특정 테이블과 이 테이블의 모든 슈퍼 테이블에 첨부된 행 트리거를 활성화합니다. 이 규칙은 SQL문의 테이블 액세스 방법에 관계없이 항상 적용됩니다. 예를 들어, UPDATE EMP 명령을 발행하는 경우 갱신된 행의 일부가 서브테이블 MGR에 있을 수 있습니다. EMP 행의 경우, EMP 및 그의 슈퍼 테이블에 첨부된 행 트리거가 활성화됩니다. MGR 행의 경우, MGR 및 그의 슈퍼 테이블에 첨부된 행 트리거가 활성화됩니다.
  – 명령문 트리거: INSERT, UPDATE 또는 DELETE문은 명령문의 영향을 받는 테이블(및 슈퍼 테이블)에 첨부된 명령문 트리거를 활성화합니다. 이 규칙은 이러한 테이블의 실제 행이 영향을 받는지 여부에 관계없이 항상 적용됩니다. 예를 들어, INSERT INTO EMP 명령에서 EMP 및 그의 슈퍼 테이블에 대한 명령문 트리거가 활성화됩니다. 다른 예로서, UPDATE EMP 또는 DELETE EMP 명령에서는 서브테이블 행이 갱신되거나 삭제되지 않지만 EMP 및 그의 슈퍼 테이블에 대한 명령문 트리거가 활성화됩니다. 마찬가지로, UPDATE ONLY(EMP) 또는 DELETE ONLY(EMP) 명령은 EMP 및 그의 슈퍼 테이블에 대한 명령문 트리거는 활성화하나 서브테이블에 대한 명령문 트리거는 활성화하지 않습니다.

DROP TABLE문의 효과: DROP TABLE문은 삭제하려는 테이블에 첨부된 트리거는 활성화하지 않습니다. 그러나 삭제된 테이블이 서브테이블인 경우에는 삭제 다음이 수행됩니다. 테이블의 모든 행이 슈퍼 테이블에서 삭제됩니다. 따라서 테이블 T의 경우:

– 행 트리거: DROP TABLE T는 T의 각 행에 대해 T의 모든 슈퍼 테이블에 첨부된 행 유형 삭제 트리거를 활성화합니다.

– 명령문 트리거: DROP TABLE T는 T에 행이 있는지 여부에 관계없이 T의 모든 슈퍼 테이블에 첨부된 명령문 유형 삭제 트리거를 활성화합니다.

뷰에 대한 조치: 뷰에 대한 조치로 활성화되는 트리거를 예측하려면, 뷰 정의를 사용하여 해당 조치를 기본 테이블에 대한 조치로 변환하십시오. 예를 들면, 다음과 같습니다.

1. SQL문은 UPDATE V1을 수행합니다. 여기서, V1은 서브뷰 V2를 갖는 유형이 지정된 뷰입니다. V1에는 하위 테이블 T1이 있고 V2에는 하위 테이블 T2가 있다고 가정합니다. 명령문은 잠재적으로 T1, T2 및 그의 서브테이블에 있는 행에 영향을 미치므로, T1, T2 및 그의 모든 서브테이블과 슈퍼 테이블에 대한 명령문 트리거가 활성화됩니다.
2. SQL문은 UPDATE V1을 수행합니다. 여기서, V1은 서브뷰 V2를 갖는 유형이 지정된 뷰입니다. V1은 SELECT ... FROM ONLY(T1)으로 정의되고 V2는 SELECT ... FROM ONLY(T2)로 정의된다고 가정합니다. 명령문은 T1과 T2의 서브테이블에 있는 행에 영향을 줄 수 없으므로, T1, T2 및 그의 슈퍼 테이블에 대한 명령문 트리거는 활성화되나 서브테이블에 대한 명령문 트리거는 활성화되지 않습니다.
3. SQL문은 UPDATE ONLY(V1)을 수행합니다. 여기서, V1은 SELECT ... FROM T1으로 정의된 유형이 지정된 뷰입니다. 명령문은 잠재적으로 T1과 그의 서브테이블에 영향을 미칠 수 있습니다. 따라서 T1 및 그의 모든 서브테이블과 슈퍼 테이블에 대한 명령문 트리거가 활성화됩니다.
4. SQL문은 UPDATE ONLY(V1)을 수행합니다. 여기서, V1은 SELECT ... FROM ONLY(T1)으로 정의된 유형이 지정된 뷰입니다. 이 경우, T1에 서브뷰가 있고 T1에 서브테이블이 있더라도 이 명령문이 영향을 미치는 유일한 테이블은 T1뿐입니다. 따라서 T1 및 그의 슈퍼 테이블에 대한 명령문 트리거만 활성화됩니다.

- ***MERGE문 및 트리거***: MERGE문은 갱신, 삭제 및 삽입 조작을 실행할 수 있습니다. 적용 가능한 UPDATE, DELETE 또는 INSERT 트리거는 갱신, 삭제 또는 삽입 조작이 실행될 때 MERGE문에 대해 활성화됩니다.
- *호환성*: 이전 버전의 DB2 데이터베이스와의 호환성을 위해,
  – OLD TABLE 대신 OLD_TABLE을, 그리고 NEW TABLE 대신 NEW_TABLE을 지정할 수 있습니다.

– MODE DB2SQL은 다음 FOR EACH ROW 또는 FOR EACH STATEMENT를 지정할 수 있습니다.

예:

예 1: 회사가 관리하는 사원의 수를 자동으로 추적하는 두 개의 트리거를 작성하십시오. 트리거는 다음 테이블과 상호 작용합니다.

- ID, NAME, ADDRESS, POSITION 컬럼을 갖고 있는 EMPLOYEE 테이블
- NBEMP, NBPRODUCT 및 REVENUE 컬럼이 있는 COMPANY_STATS 테이블

신입 사원이 고용될 때마다, 즉 새로운 행이 EMPLOYEE 테이블에 삽입될 때마다 첫 번째 트리거는 사원 수를 증가시키십시오.

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

두 번째 트리거는 사원이 회사를 그만 둘 때마다, 즉 행이 EMPLOYEE 테이블에서 삭제될 때마다 사원 수를 감소시키십시오.

```
CREATE TRIGGER FORMER_EMP
  AFTER DELETE ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1
```

예 2: 부품 레코드가 갱신될 때마다 다음이 점검되고(필요 시) 조치가 취해지도록 하는 트리거를 작성하십시오.

- 남은 양이 최대 재고량의 10% 이내이면, (최대 재고량 - 남은 양)과 같은 영향을 받는 부품에 대해 품목 수만큼 주문하는 선적 요구를 발행합니다.

트리거는 PARTNO, DESCRIPTION, ON_HAND, MAX_STOCKED 및 PRICE 컬럼이 있는 PARTS 테이블과 상호 작용합니다.

ISSUE_SHIP_REQUEST는 적합한 회사에 추가 부품 주문서 양식을 보내는 사용자 정의 함수입니다.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (N.ON_HAND < 0.10 * N.MAX_STOCKED)
  BEGIN ATOMIC
  VALUES(ISSUE_SHIP_REQUEST(N.MAX_STOCKED - N.ON_HAND, N.PARTNO));
  END
```

예 3: 현재 급여의 10%가 넘는 급여 인상을 가져오는 갱신이 발생하는 경우, 오류를 발생시키는 트리거를 작성하십시오.

```
CREATE TRIGGER RAISE_LIMIT
  AFTER UPDATE OF SALARY ON EMPLOYEE
  REFERENCING NEW AS N OLD AS O
  FOR EACH ROW
  WHEN (N.SALARY > 1.1 * O.SALARY)
        SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT='Salary increase>10%'
```

예 4: 주식 가격에 대한 변경을 기록하고 추적하는 응용프로그램을 고려합니다. 데이터베이스에는 CURRENTQUOTE와 QUOTEHISTORY 테이블이 포함됩니다.

```
Tables: CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
                QUOTEHISTORY (SYMBOL, QUOTE, QUOTE_TIMESTAMP)
```

CURRENTQUOTE의 QUOTE 컬럼이 갱신될 때, 새로운 할당량이 시각과 함께 QUOTEHISTORY 테이블에 복사되어야 합니다. 또한 CURRENTQUOTE의 STATUS 컬럼도 다음과 같은 재고 상태를 나타내기 위해 갱신되어야 합니다.

1. 값 상승 여부
2. 그 해에 대해 새 높은 가격에 있는지 여부
3. 값 삭제 여부
4. 그 해에 대해 새 낮은 가격에 있는지 여부
5. 값이 안정적인지 여부

이를 수행하는 CREATE TRIGGER문은 다음과 같습니다.

- 상태를 설정하는 트리거 정의

```
CREATE TRIGGER STOCK_STATUS
  NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
  FOR EACH ROW
  BEGIN ATOMIC
            SET NEWQUOTE.STATUS =
      CASE
                     WHEN NEWQUOTE.QUOTE >
                (SELECT MAX(QUOTE) FROM QUOTEHISTORY
                WHERE SYMBOL = NEWQUOTE.SYMBOL
                AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
                THEN 'High'
           WHEN NEWQUOTE.QUOTE <
(SELECT MIN(QUOTE) FROM QUOTEHISTORY
                WHERE SYMBOL = NEWQUOTE.SYMBOL
                AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
                THEN 'Low'
           WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
                THEN 'Rising'
           WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
                THEN 'Dropping'
           WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
                THEN 'Steady'
      END;
  END
```

- QUOTEHISTORY 테이블에 변경사항을 기록하는 트리거 정의

```
CREATE TRIGGER RECORD_HISTORY
  AFTER UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE
  FOR EACH ROW
  BEGIN ATOMIC
          INSERT INTO QUOTEHISTORY
      VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
  END
```

*예 5:*org 테이블의 직원 레코드의 위치 필드에 대한 변경사항을 겹쳐쓰는 트리거를 작성합니다. 작은 규모의 회사가 구입해서 획득한 새 직원 레코드가 처리되고 직원에 할당된 목표 위치는 'Toronto'고 새 목표 위치는 'Los Angeles'라면 이 트리거가 유용합니다. BEFORE 트리거는 이 필드에 대해 응용프로그램이 할당한 값에 상관없이 마지막 결과 값은 'Los Angeles'임을 확인합니다.

```
CREATE TRIGGER LOCATION_TRIGGER
  NO CASCADE
   BEFORE UPDATE ON ORG
   REFERENCING
     OLD AS PRE
     NEW AS POST
   FOR EACH ROW
   WHEN (POST.LOCATION = 'Toronto')
      SET POST.LOCATION = 'Los Angeles';
   END
```

예 6: SAMPLE 데이터베이스의 PRODUCT 테이블로 삽입하기 전에 새 제품 설명을 포함한 XML 문서를 자동으로 유효성 확인하는 BEFORE 트리거를 작성합니다.

```
CREATE TRIGGER NEWPROD NO CASCADE BEFORE INSERT ON PRODUCT
   REFERENCING NEW AS N
   FOR EACH ROW
   BEGIN ATOMIC
      SET (N.DESCRIPTION) = XMLVALIDATE(N.DESCRIPTION
         ACCORDING TO XMLSCHEMA ID product);
   END
```

CREATE TRUSTED CONTEXT

CREATE TRUSTED CONTEXT문은 현재 서버에서 트러스트된 컨텍스트를 정의합니다.

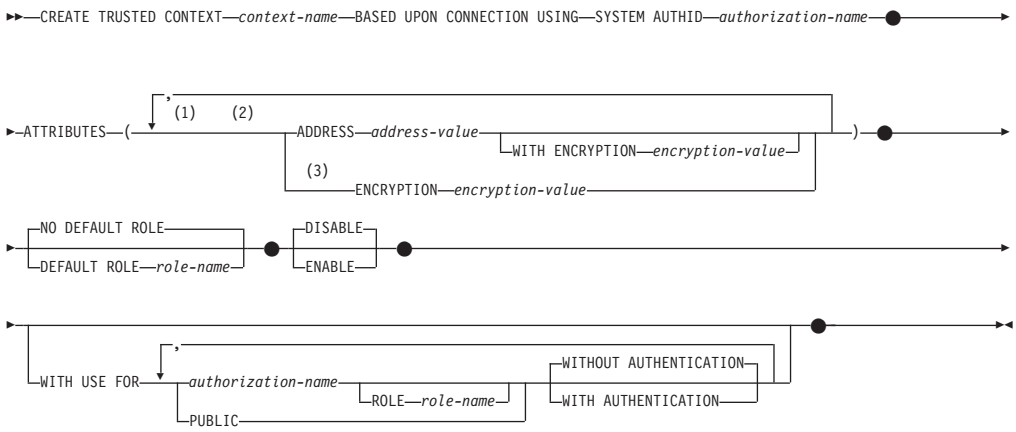
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



주:

1 ATTRIBUTES, DEFAULT ROLE, ENABLE 및 WITH USE 절 각각은 최대 한 번 지정할 수 있습니다(SQLSTATE 42614).

2 각각의 속성 이름과 해당 이름은 고유해야 합니다(SQLSTATE 4274D).

3 ENCRYPTION은 두 번 이상 지정할 수 없습니다(SQLSTATE 42614). 그러나 지정된 ADRESS마다 WITH ENCRYPTION을 지정할 수 있습니다.

설명

context-name

트러스트된 컨텍스트의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL ID(일반 또는 분리 ID)입니다. 이름은 현재 서버에 이미 존재하는 트러스트된 컨텍스트를 식별하면 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

BASED UPON CONNECTION USING SYSTEM AUTHID *authorization-name*
컨텍스트가 시스템 권한 부여 ID *authorization-name*에 의해 설정된 연결임을 지정합니다. 이 ID는 기존의 트러스트된 컨텍스트와 연관될 수 없습니다(SQLSTATE 428GL). 이는 명령문의 권한 부여 ID가 될 수 없습니다(SQLSTATE 42502).

ATTRIBUTES (...)
트러스트된 컨텍스트가 정의되는 하나 이상의 연결 트러스트 속성 목록을 지정합니다.

ADDRESS *address-value*
클라이언트가 데이터베이스 서버와 통신하기 위해 사용하는 실제 통신 주소를 지정합니다. 지원되는 유일한 프로토콜은 TCP/IP입니다. ADDRESS 속성을 여러 번 지정할 수 있지만, 각 *address-value* 쌍은 속성 세트에 대해 고유해야 합니다(SQLSTATE 4274D).

트러스트된 연결을 설정할 때 트러스트된 컨텍스트의 ADDRESS 속성에 복수의 값이 정의되면, 연결에 사용되는 주소가 트러스트된 컨텍스트의 ADDRESS 속성에 대해 정의된 값 중 하나와 일치하는 경우 후보 연결이 이 속성과 일치하는 것으로 간주됩니다.

address-value
ADDRESS 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다. *address-value*는 IPv4 주소, IPv6 주소 또는 보안 도메인 이름이어야 합니다.

- IPv4 주소는 앞 공백을 포함할 수 없으며 점 10진 주소로 나타냅니다. IPv4 주소의 예로 9.112.46.111이 있습니다. 값 'localhost' 또는 해당되는 표현 '127.0.0.1'은 결과적으로 일치하지 않습니다. 대신 호스트의 실제 IPv4 주소를 지정해야 합니다.
- IPv6 주소는 앞 공백을 포함할 수 없으며 콜론 16진 주소로 나타냅니다. IPv6 주소의 예로 2001:0DB8:0000:0000:0008:0800:200C:417A가 있습니다. IPv4 맵핑 IPv6 주소(예, ::ffff:192.0.2.128)는 결과적으로 일치하지 않습니다. 마찬가지로, 'localhost' 또는 해당되는 IPv6 축약 표현 '::1'은 결과적으로 일치하지 않습니다.
- 도메인 이름은 도메인 이름 서버에 의해 결과 IPv4 또는 IPv6 주소가 판별되는 IP 주소로 변환됩니다. 도메인 이름의 예로 corona.torolab.ibm.com이 있습니다. 도메인 이름이 IP 주소로 변환될 때 이 변환의 결과는 하나 이상의 IP 주소로 설정될 수 있습니다. 이 경우, 연결이 시작되는 IP 주소가 도메인 이름이 변환된 IP 주소 중 하나와 일치하면 트러스트된 컨텍스트 오브젝트의 ADDRESS 속성과 일치함을 의미합니다. 트러스트된 컨텍스트 오브젝트를 작성할 때 정적 IP 주소 대신 ADDRESS 속성의 도메인 이름 값을 제공하는 것이 좋습니다(특히

DHCP(Dynamic Host Configuration Protocol) 환경에서). DHCP를 사용하는 경우, 디바이스는 네트워크에 연결할 때마다 다른 IP 주소를 가질 수 있습니다. 따라서, 트러스트된 컨텍스트 오브젝트의 ADDRESS 속성에 대해 정적 IP 주소가 제공되는 경우 일부 디바이스는 의도하지 않게 트러스트된 연결을 획득할 수 있습니다. 트러스트된 컨텍스트 오브젝트의 ADDRESS 속성에 도메인 이름을 제공하면 DHCP 환경에서 이러한 문제점이 방지됩니다.

WITH ENCRYPTION *encryption-value*

특정 *address-value*에 대한 네트워크 암호화 또는 데이터 스트림 암호화의 최소 레벨을 지정합니다. 이 *encryption-value*는 특정 *address-value*에 대한 전역 ENCRYPTION 속성을 대체합니다.

encryption-value

특정 *address-value*에 대한 ENCRYPTION 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다.

*encryption-value*는 다음 중 하나여야 합니다(SQLSTATE 42615).

- NONE: 암호화의 특정 레벨이 필요하지 않습니다.
- LOW: 최소한의 가벼운 암호화가 필요합니다. 데이터베이스 관리 프로그램에 대한 인증 유형은 수신 연결이 이 특정 주소에 대한 암호화 설정과 일치할 경우 DATA_ENCRYPT여야 합니다.
- HIGH, 수신 연결이 이 특정 주소에 대한 암호화 설정과 일치하는 경우 DB2 클라이언트와 DB2 서버 사이의 데이터 통신에 SSL(Secure Sockets Layer) 암호화를 사용해야 합니다.

ENCRYPTION *encryption-value*

네트워크 암호화 또는 데이터 스트림 암호화의 최소 레벨을 지정합니다. 디폴트는 NONE입니다.

encryption-value

특정 *address-value*에 대한 ENCRYPTION 트러스트 속성과 연관될 값을 포함하는 문자열 상수를 지정합니다. *encryption-value*는 다음 중 하나여야 합니다(SQLSTATE 42615).

- NONE: 수신 연결이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치하도록 하는 데 암호화의 특정 레벨이 필요하지 않습니다.
- LOW: 최소한의 가벼운 암호화가 필요합니다. 데이터베이스 관리 프로그램에 대한 인증 유형은 수신 연결이 이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치할 경우 DATA_ENCRYPT여야 합니다.

- HIGH, 수신 연결이 이 트러스트된 컨텍스트 오브젝트의 ENCRYPTION 속성과 일치할 경우 DB2 클라이언트와 DB2 서버 사이의 데이터 통신에 SSL(Secure Sockets Layer) 암호화를 사용해야 합니다.

다음 테이블은 기존 연결에서 사용되는 암호화에 따라 트러스트된 컨텍스트를 사용하는 경우를 요약한 것입니다. 트러스트된 컨텍스트를 연결에 사용할 수 없는 경우 경고가 리턴되고(SQLSTATE 01679) SQLCA의 SQLWARN8 필드는 'Y'로 설정됩니다. 이는 연결이 일반(트러스트되지 않은) 연결임을 표시합니다.

표 24. 암호화 및 트러스트된 컨텍스트

기존 연결에서 사용되는 암호화	트러스트된 컨텍스트의 **ENCRYPTION** 값	연결에 트러스트된 컨텍스트를 사용할 수 있는지 여부
암호화 안함	'NONE'	예
암호화 안함	'LOW'	없음
암호화 안함	'HIGH'	없음
낮은 암호화 (DATA_ENCRYPT)	'NONE'	예
낮은 암호화 (DATA_ENCRYPT)	'LOW'	예
낮은 암호화 (DATA_ENCRYPT)	'HIGH'	없음
높은 암호화(SSL)	'NONE'	예
높은 암호화(SSL)	'LOW'	예
높은 암호화(SSL)	'HIGH'	예

NO DEFAULT ROLE 또는 DEFAULT ROLE *role-name*

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결과 디폴트 역할이 연관되는지 여부를 지정합니다. 디폴트는 NO DEFAULT ROLE입니다.

NO DEFAULT ROLE

트러스트된 컨텍스트가 디폴트 역할을 가지고 있지 않음을 지정합니다.

DEFAULT ROLE *role-name*

*role-name*이 트러스트된 컨텍스트의 디폴트 역할임을 지정합니다. *role-name*은 현재 서버에 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). 이 역할은 사용자가 트러스트된 컨텍스트 정의의 일부로 정의된 사용자 특정 역할을 가지고 있지 않은 경우 이 트러스트된 컨텍스트를 기초로, 트러스트된 연결에서 사용자에 대해 사용됩니다.

DISABLE 또는 ENABLE

트러스트된 컨텍스트가 사용 가능 또는 사용 불가능 상태에서 작성되는지 여부를 지정합니다. 디폴트는 DISABLE입니다.

DISABLE

트러스트된 컨텍스트가 사용 불가능 상태에서 작성됨을 지정합니다. 사용 불가능한 트러스트된 컨텍스트는 트러스트된 연결이 설정될 때 고려되지 않습니다.

ENABLE

트러스트된 컨텍스트가 사용 가능 상태에서 작성됨을 지정합니다.

WITH USE FOR

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결을 사용할 수 있는 사용자를 지정합니다.

authorization-name

지정된 *authorization-name*이 트러스트된 연결을 사용할 수 있음을 지정합니다. *authorization-name*은 WITH USE FOR 절에서 두 번 이상 지정할 수 없습니다(SQLSTATE 428GM). 이는 명령문의 권한 부여 ID여도 안됩니다(SQLSTATE 42502). 트러스트된 컨텍스트의 정의에서 PUBLIC 및 사용자 목록 둘 다의 액세스를 허용하는 경우, 사용자 스펙이 PUBLIC 스펙보다 우선합니다. 예를 들어, PUBLIC WITH AUTHENTICATION 및 JOE WITHOUT AUTHENTICATION 둘 다에 의한 액세스를 허용하는 트러스트된 컨텍스트가 정의되어 있다고 가정합니다. JOE가 트러스트된 컨텍스트를 사용하는 경우 인증이 필요하지 않습니다. 그러나 GEORGE가 트러스트된 컨텍스트를 사용하는 경우에는 인증이 필요합니다.

ROLE *role-name*

트러스트된 연결이 트러스트된 컨텍스트를 사용 중일 때 *role-name*이 사용자에 대해 사용할 역할임을 지정합니다. *role-name*은 현재 서버에 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). 사용자에 대해 명시적으로 지정된 역할이 트러스트된 컨텍스트와 연관되는 디폴트 역할보다 우선합니다.

PUBLIC

트러스트된 컨텍스트를 기초로 하는 트러스트된 연결을 모든 사용자가 사용할 수 있음을 지정합니다. PUBLIC은 두 번 이상 지정할 수 없습니다(SQLSTATE 428GM). 이와 같은 트러스트된 연결을 사용하는 모든 사용자는 연관된 트러스트된 컨텍스트의 디폴트 역할과 연관된 특권을 사용합니다. 트러스트된 컨텍스트에 대해 디폴트 역할이 정의되지 않은 경우, 이 트러스트된 컨텍스트를 기초로 하는 트러스트된 연결을 사용하는 사용자와 연관되는 역할이 없습니다.

WITHOUT AUTHENTICATION 또는 WITH AUTHENTICATION

트러스트된 연결에 대한 사용자 전환에 사용자 인증이 필요한지 여부를 지정합니다. 디폴트는 WITHOUT AUTHENTICATION입니다.

WITHOUT AUTHENTICATION

트러스트된 연결의 현재 사용자를 해당 사용자로 전환하는 데 인증이 필요하지 않음을 지정합니다.

WITH AUTHENTICATION
트러스트된 연결의 현재 사용자를 해당 사용자로 전환하는 데 인증이 필요함을 지정합니다.

규칙

- 트러스트된 컨텍스트 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). 트러스트된 컨텍스트 독점 SQL문은 다음과 같습니다.
  - CREATE TRUSTED CONTEXT, ALTER TRUSTED CONTEXT 또는 DROP(TRUSTED CONTEXT)
- 트러스트된 컨텍스트 독점 SQL문은 페더레이티드 트랜잭션에 대한 2단계 커미트의 일부로 시작되는 전역 트랜잭션이나 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 트러스트된 컨텍스트 정의의 일부로 IP 주소를 제공할 때 주소는 네트워크에 적용되는 형식이어야 합니다. 예를 들어, 네트워크가 IPv4일 때 IPv6 형식으로 주소를 제공하면 일치하지 않게 됩니다. 혼합 환경에서는, 주소의 IPv4 및 IPv6 표시를 둘 다 지정하거나, 더 낫게는 주소 형식 세부사항을 숨기는 보안 도메인 이름(예: corona.torolab.ibm.com)을 지정하는 것이 좋습니다.
- ***트러스트된 컨텍스트 정의에서 역할 지정:*** 트러스트된 컨텍스트의 정의는 특정 권한 부여 ID에 대한 역할과, 트러스트된 컨텍스트 정의에 특정 역할이 지정되지 않은 권한 부여 ID에 사용할 디폴트 역할을 지정할 수 있습니다. 이 역할은 트러스트된 컨텍스트를 기초로 하는 트러스트된 연결에 사용할 수 있지만, 트러스트된 컨텍스트를 기초로 하는 트러스트된 연결 외부에서는 역할을 사용할 수 없습니다.
- 트러스트된 연결을 사용하여 데이터 처리 언어(DML) SQL문을 실행할 때, 명령문의 권한 부여 ID에 의해 직접 보유하고 있거나 명령문의 권한 부여 ID에 의해 다른 역할이 간접적으로 보유하는 다른 특권 외에도, 연관되는 트러스트된 컨텍스트 정의 내에서 권한 부여 ID에 대해 적용 중인 컨텍스트 지정 역할에 의해 보유되는 특권이 고려됩니다.
- 연관되는 트러스트된 컨텍스트 정의 내에서 권한 부여 ID에 대해 적용 중인 컨텍스트 지정 역할에 의해 보유되는 특권은 데이터 정의 언어(DDL) SQL문에 대해 고려되지 않습니다. 예를 들어, 오브젝트를 작성하려면 컨텍스트 지정 역할이 보유하는 특권을 포함하지 않아도 명령문의 권한 부여 ID가 작성할 수 있어야 합니다.
- 동일한 머신에서 기존 응용프로그램과 동일한 증명서를 사용하여 DB2에 대해 인증하고 트러스트된 컨텍스트를 이용하는 새 응용프로그램을 설치할 때, 새 응용프로그램은 동일한 트러스트된 컨텍스트 오브젝트도 이용할 수 있습니다(예를 들어, 트러스

트된 오브젝트 역할을 상속함). 이는 보안 관리자의 의도가 아닐 수 있습니다. 보안 관리자는 트러스트된 컨텍스트 오브젝트를 이용하는 응용프로그램을 찾기 위해 DB2 감사 기능을 작동시킬 수 있습니다.

- 모든 데이터베이스 파티션 사이에서 한 번에 단 하나의 언커미트된 트러스트된 컨텍스트 독점 SQL문만 허용됩니다. 언커미트된 트러스트된 컨텍스트 독점 SQL문이 실행 중인 경우, 연속 트러스트된 컨텍스트 독점 SQL문은 현재 트러스트된 컨텍스트 독점 SQL문이 커미트되거나 롤백될 때까지 기다립니다.
- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예:

예 1: 트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자가 두 개의 다른 사용자 ID로 전환될 수 있는 트러스트된 컨텍스트를 작성하십시오. 연결의 현재 사용자가 사용자 ID JOE로 전환될 때, 인증이 필요하지 않습니다. 그러나 연결의 현재 사용자가 사용자 ID BOB로 전환될 때는 인증이 필요합니다. 트러스트된 컨텍스트가 디폴트 역할 *context-role*을 가지고 있습니다. 이는 트러스트된 컨텍스트 범위 안에서 작업 중인 사용자가 역할 *context-role*과 연관되는 특권을 상속한다는 것을 의미합니다.

```
CREATE TRUSTED CONTEXT APPSERVER
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  DEFAULT ROLE CONTEXT_ROLE
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204')
  WITH USE FOR JOE WITHOUT AUTHENTICATION
    BOB WITH AUTHENTICATION
```

예 2: 트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자가 인증없이 임의 사용자 ID로 전환될 수 있는 트러스트된 컨텍스트를 작성하십시오.

```
CREATE TRUSTED CONTEXT SECUREROLE
  BASED UPON CONNECTION USING SYSTEM AUTHID PBIRD
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204')
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

예 3: 트러스트된 컨텍스트를 기초로 하는 트러스트된 연결의 현재 사용자가 인증없이 임의 사용자 ID로 전환될 수 있는 트러스트된 컨텍스트를 작성하십시오. 이 트러스트된 컨텍스트와 예 2에서 작성된 트러스트된 컨텍스트 사이의 차이는, 이 트러스트된 컨텍스트가 ENCRYPTION이라고 하는 추가 속성을 가지고 있는 것입니다. 트러스트된 컨텍스트 SECUREROLEENCRYPT에 대한 ENCRYPTION 속성 설정은, 연결에서 사용되는 암호화 설정이 이 트러스트된 컨텍스트 속성과 일치하도록 최소한 ″낮은 암호화 ″여야 함을 의미합니다(794 페이지의 표 24 참조).

```
CREATE TRUSTED CONTEXT SECUREROLEENCRYPT
  BASED UPON CONNECTION USING SYSTEM AUTHID SHARPER
  ENABLE
```

```
  ATTRIBUTES (ADDRESS '9.26.113.204'
    ENCRYPTION 'LOW')
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

예 4: 주소 9.26.146.201 및 9.26.146.203에서 사용자 WRJAIBI가 작성한 연결이 암호화가 사용되지 않는 경우에도 트러스트되지만, 주소 9.26.146.202에서 사용자 WRJAIBI가 작성한 연결이 트러스트되려면 LOW 레벨의 암호화가 필요한 트러스트된 컨텍스트를 작성하십시오.

```
CREATE TRUSTED CONTEXT WALIDLOCSENSITIVE
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.146.201',
    ADDRESS '9.26.146.202' WITH ENCRYPTION 'LOW',
    ADDRESS '9.26.146.203'
    ENCRYPTION 'NONE')
```

CREATE TYPE(배열)

CREATE TYPE(배열)문은 배열 유형을 정의합니다. 배열 유형의 요소는 내장 데이터 유형 또는 사용자 정의 구별 유형 중 하나를 기반으로 합니다.

호출

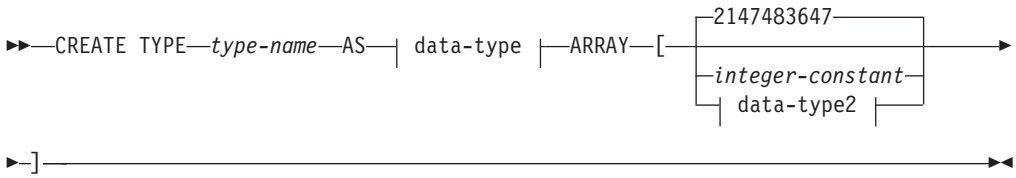
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509) 에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

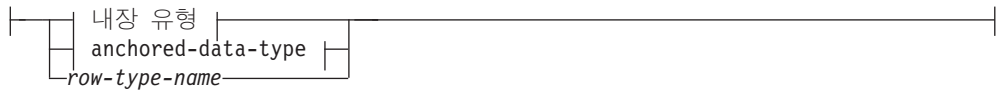
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 배열 유형의 스키마 이름이 기존 스키마를 참조하지 않는 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 배열 유형의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

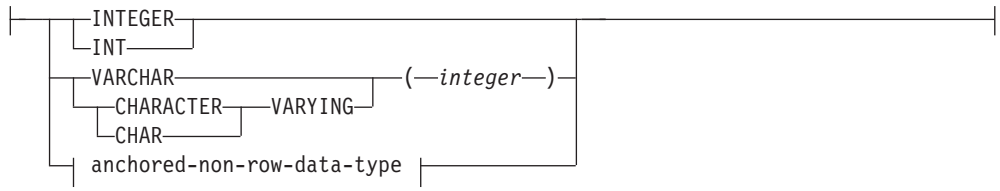
구문



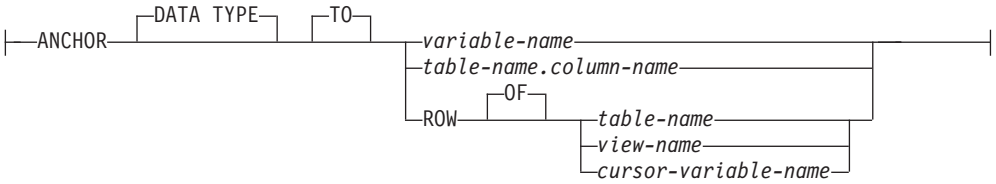
data-type:



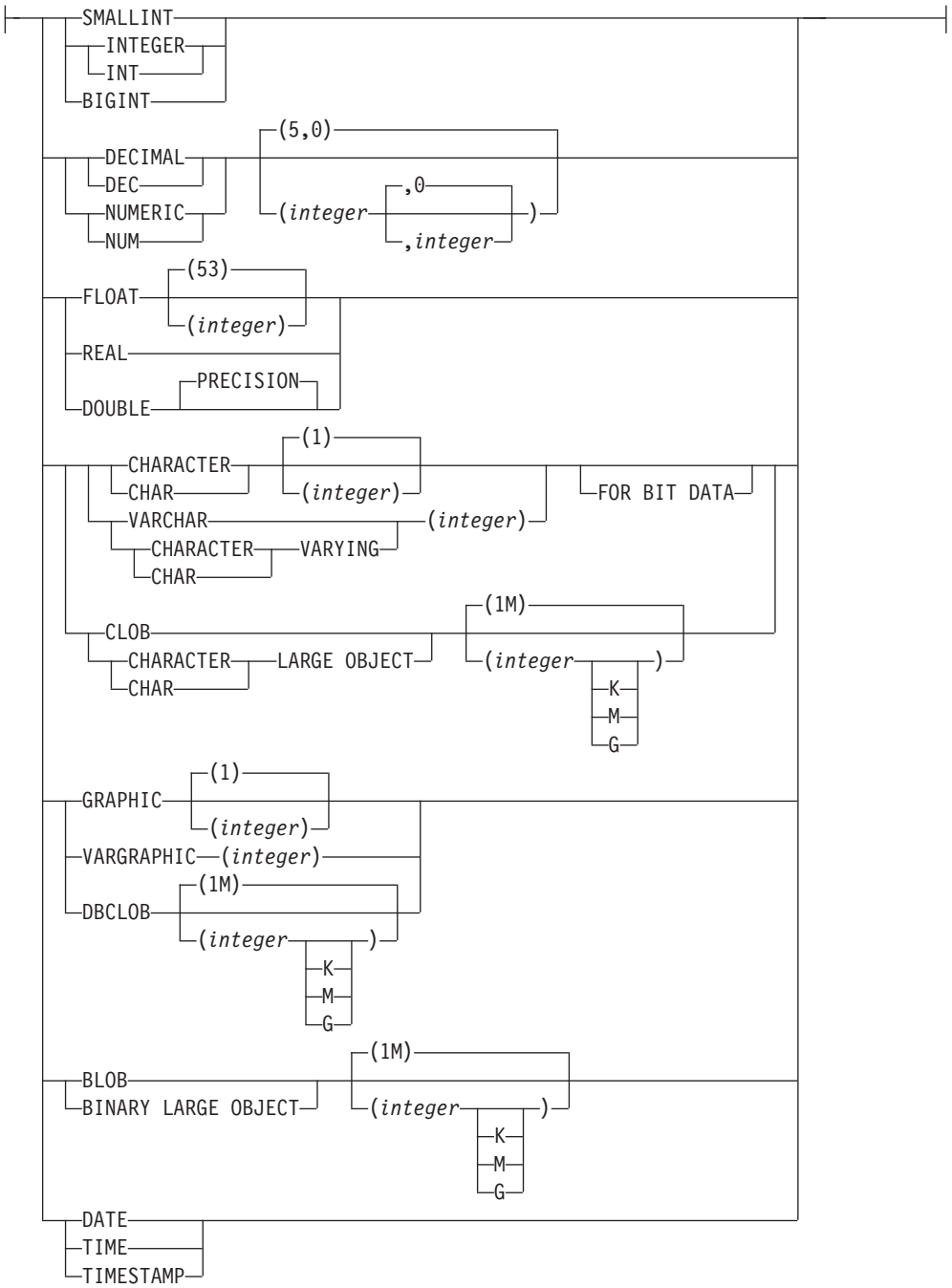
data-type2:



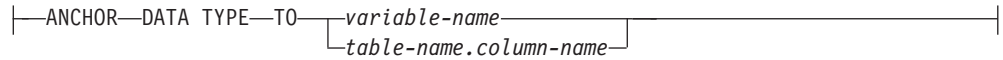
anchored-data-type:



built-in-type:



anchored-non-row-data-type:



설명

type-name

유형의 이름을 지정합니다. 이름(내재된 또는 명시적 규정자 포함)은 현재 서버에 이미 존재하는 다른 유형(내장 유형 또는 사용자 정의 유형)을 식별해서는 안됩니다. 규정되지 않은 이름은 내장 데이터 유형 이름이나 BOOLEAN, BINARY 또는 VARBINARY와 같을 수 없습니다(SQLSTATE 42918).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *type-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

두 부분으로 구성된 *type-name*이 지정된 경우, 스키마 이름은 문자 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

data-type

배열 요소의 데이터 유형을 지정합니다.

built-in-type

내장 데이터 유형을 지정합니다. 내장 데이터 유형에 대해서는 ″CREATE TABLE″을 참조하십시오. 내장 유형은 『CREATE TABLE』에 설명된 데이터 유형(참조, SYSPROC.DB2SECURITYLABEL, XML 또는 사용자 정의 유형 이외)을 포함합니다(SQLSTATE 429C2).

row-type-name

사용자 정의 행 유형 이름을 지정합니다. 각 배열 요소 필드는 행 유형의 필드입니다. *row-type-name*이 스키마 이름 없이 지정된 경우 SQL 경로에서 스키마를 검색하여 *row-type-name*이 해석됩니다.

anchored-data-type

데이터 유형을 판별하기 위해 사용되는 다른 오브젝트를 식별합니다. 고정 오브젝트의 데이터 유형은 데이터 유형을 직접 지정할 때, 또는 행의 경우 행 유형을 작성하기 위해 적용하는 것과 동일한 제한사항에 의해 바운드됩니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

전역 변수를 식별합니다. 전역 변수의 데이터 유형은 배열 요소의 데이터 유형으로 사용됩니다.

table-name.column-name

기존 테이블 또는 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 배열 요소의 데이터 유형으로 사용됩니다.

ROW OF *table-name* **또는** *view-name*

*table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뮤의 컬럼 이름 및 컬럼 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 포함된 필드 행을 지정합니다. 배열 요소의 데이터 유형은 unnamed row 자료형입니다.

ROW OF *cursor-variable-name*

*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 포함된 필드 행을 식별합니다. 지정된 커서 변수는 다음 중 하나여야 합니다(SQLSTATE 428HS).

- 강하게 유형 지정된 커서 데이터 유형이 있는 전역 변수
- 모든 결과 컬럼이 이름 지정된 *select-statement*를 지정하는 CONSTANT절로 작성되거나 선언되어 약하게 유형 지정된 커서 데이터 유형이 있는 전역 변수

커서 변수의 커서 유형이 named row 자료형을 사용하여 강하게 유형이 지정되지 않은 경우 배열 요소의 데이터 유형은 unnamed row 자료형입니다.

anchored-non-row-data-type

데이터 유형을 판별하기 위해 사용되는 다른 오브젝트를 식별합니다. 고정 오브젝트의 데이터 유형은 데이터 유형을 직접 지정할 때 적용되는 것과 동일한 제한사항에 의해 바운드됩니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

INTEGER 또는 VARCHAR 데이터 유형인 데이터 유형의 전역 변수를 식별합니다. 전역 변수의 데이터 유형은 배열 인덱스의 데이터 유형으로 사용됩니다.

table-name.column-name

INTEGER 또는 VARCHAR 데이터 유형인 데이터 유형의 기존 테이블이나 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 배열 인덱스의 데이터 유형으로 사용됩니다.

ARRAY [*integer-constant***]**

유형이 *integer-constant*의 최대 카디널리티(cardinality)를 가지고 있는 배열임을 지정합니다. 값은 0보다 크고 가장 큰 양의 정수 값보다 작아야 합니다(SQLSTATE

42820). 디폴트값은 가장 큰 양수 값입니다(2 147 483 647). 배열 값의 카디널리티(cardinality)는 배열 값에 지정된 가장 높은 요소 위치로 판별됩니다.

제공된 시스템에서 배열의 최대 카디널리티(cardinality)는 DB2 응용프로그램에 사용 가능한 총 메모리 양으로 제한됩니다. 큰 카디널리티 배열을 작성할 수는 있지만 모든 요소를 사용할 수 있는 것은 아닙니다.

ARRAY[*data-type2***]**

유형이 데이터 유형 *data-type2*의 값으로 인덱스된 연관 배열임을 지정합니다. 데이터 유형은 INTEGER 또는 VARCHAR 데이터 유형이어야 합니다(SQLSTATE 429C2). 배열 요소를 지정할 때 인덱스로 지정된 값은 *data-type2*의 값으로 지정 가능해야 합니다. 배열 값의 카디널리티(cardinality)는 배열 요소를 지정할 때 사용된 고유 인덱스 값의 수로 판별됩니다.

규칙

- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.

주

- ***배열 유형 사용:*** 배열 유형은 다음과 같은 데이터 유형으로만 사용할 수 있습니다.
  - 복합 SQL(컴파일된)문의 로컬 변수
  - SQL 루틴의 매개변수
  - Java 프로시저의 매개변수(일반 배열에만 해당)
  - SQL 함수의 리턴 유형
  - 전역 변수
- 배열 유형을 사용하여 정의된 변수 또는 매개변수는 복합 SQL(컴파일된) 명령문에서만 사용할 수 있습니다.

예:

예 1: 데이터 유형이 DECIMAL(10, 0)인 최대 50개의 요소가 있는 PHONENUMBERS 배열 유형을 작성합니다.

```
CREATE TYPE PHONENUMBERS AS DECIMAL(10,0)
  ARRAY[50]
```

예 2: 스키마 GENERIC에 디폴트 개수의 요소가 있는 NUMBERS 배열 유형을 작성하십시오.

```
CREATE TYPE GENERIC.NUMBERS AS DECFLOAT(34)
  ARRAY[]
```

예 3: 'Home', 'Work' 또는 'Mom'과 같은 문자열로 인덱스된 DECIMAL(16, 0) 요소가 포함된 연관 배열 PERSONAL_PHONENUMBERS를 작성합니다.

```
CREATE TYPE PERSONALPHONENUMBERS AS DECIMAL(16, 0) ARRAY[VARCHAR(8)]
```

예 4: 인덱스가 지방, 지역 또는 국가 이름이고 요소는 수도인 연관된 배열 유형을 작성합니다.

```
CREATE TYPE CAPITALSARRAY AS VARCHAR(30) ARRAY[VARCHAR(20)]
```

예 5: 길이가 최대 12자인 인덱스가 제품 번호가 되는 최대 40자 길이의 제품 설정에 대한 연관된 배열 유형을 작성합니다.

```
CREATE TYPE PRODUCTS AS VARCHAR(40) ARRAY[VARCHAR(12)]
```

CREATE TYPE(커서)

CREATE TYPE(커서)문은 사용자 정의 커서 유형을 정의합니다.

호출

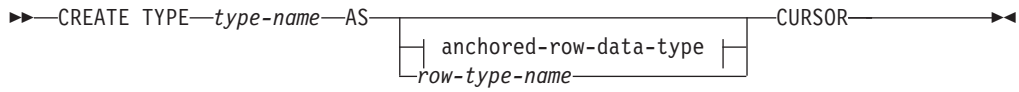
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

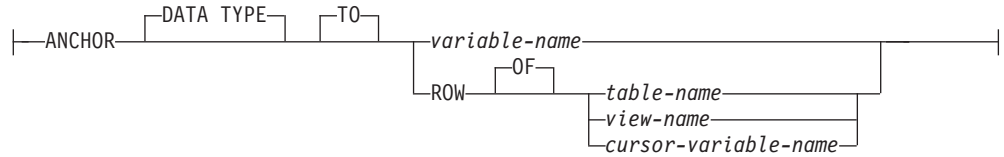
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 커서 유형의 스키마 이름이 기존 스키마를 참조하지 않는 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 커서 유형의 스키마 이름이 기존의 스키마를 참조하는 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

구문



anchored-row-data-type:



설명

type-name

유형의 이름을 지정합니다. 이름(내재된 또는 명시적 규정자 포함)은 현재 서버에 이미 존재하는 다른 유형(내장 유형 또는 사용자 정의 유형)을 식별해서는 안됩니다. 규정되지 않은 이름은 내장 데이터 유형 이름이나 BOOLEAN, BINARY 또는 VARBINARY와 같을 수 없습니다(SQLSTATE 42918).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 type-name으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름 및 비교 연산자가 이에 해당합니다. 두 부분으로 구성된 type-name이 지정된 경우, 스키마 이름은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

anchored-row-data-type

커서 유형과 연관된 행 유형을 판별하는 데 사용되는 다른 오브젝트에서 행 정보를 식별합니다. 앵커 오브젝트의 데이터 유형에는 행 유형을 작성할 때 적용되는 것과 동일한 제한사항이 있습니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

전역 변수를 식별합니다. 참조된 변수의 데이터 유형은 행 유형이어야 하며 커서 유형과 연관된 행 유형으로 사용됩니다.

ROW OF *table-name* **또는** *view-name*

*table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뷰의 컬럼 이름 및 컬럼 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 앵커 오브젝트 컬럼의 데이터 유형에는 필드 데이터 유형에 적용되는 것과 동일한 제한사항이 있습니다. 커서 유형과 연관된 행 유형은 unnamed row 자료형입니다.

ROW OF *cursor-variable-name*

*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 지정된 커서 변수는 다음 중 하나여야 합니다(SQLSTATE 428HS).

- 강하게 유형 지정된 커서 데이터 유형이 있는 전역 변수
- 모든 결과 컬럼이 이름 지정된 *select-statement*를 지정하는 CONSTANT 절로 작성되거나 선언되어 약하게 유형 지정된 커서 데이터 유형이 있는 전역 변수

커서 변수의 커서 유형이 named row 자료형을 사용하여 강하게 유형이 지정되지 않은 경우 커서 유형과 연관된 행 유형은 unnamed row 자료형입니다.

row-type-name

커서 유형의 변수에 지정된 *select-statement* 결과 테이블의 행 유형을 확인하는 데 사용할 행 유형을 지정합니다. 유형 점검이 실패하면 지정이 실패합니다(SQLSTATE 42821). 스키마 이름 없이 *row-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 행 유형이 분석됩니다.

규칙

- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.

주

- ***커서 유형 사용법:*** 커서 유형은 다음 데이터 유형으로만 사용할 수 있습니다.
  - 복합 SQL(컴파일된)문의 로컬 변수
  - SQL 루틴의 매개변수
  - SQL 함수의 리턴 유형
  - 전역 변수
- 커서 유형을 사용하여 정의된 변수 또는 매개변수는 복합 SQL(컴파일된) 명령문에서만 사용할 수 있습니다.
- 강하게 유형 지정된 커서 유형이 있는 변수 또는 매개변수를 사용하여 *select-statement* 대신에 *statement-name*에 기반한 커서 값을 지정해서는 안됩니다.
- 관련된 행 유형이 포함된 사용자 정의 커서 유형은 강하게 유형이 지정된 커서 유형이며 그 이외의 경우는 약하게 유형이 지정된 커서 유형입니다.

예:

예 1: 어떤 커서와도 함께 사용할 수 있는 커서 유형을 작성합니다.

```
CREATE TYPE EMPCURSOR AS CURSOR
```

예 2: 행 데이터 유형 DEPTROW에 기반한 강하게 유형 지정된 커서 유형을 다음과 같이 작성합니다.

```
CREATE TYPE DEPTCURSOR AS DEPTROW CURSOR
```

CREATE TYPE(구별)

CREATE TYPE(구별)문은 구별 유형을 정의합니다. 구별 유형은 내장 데이터 유형 중 하나를 기반으로 합니다. 명령문이 성공적으로 실행되면 구별 유형과 이 소스 유형 사이에서 캐스트되는 함수가 생성되며, 선택적으로 구별 유형에 사용하도록 비교 연산자 (=, <>, <, <=, > 및 >=)에 대한 지원이 생성됩니다.

호출

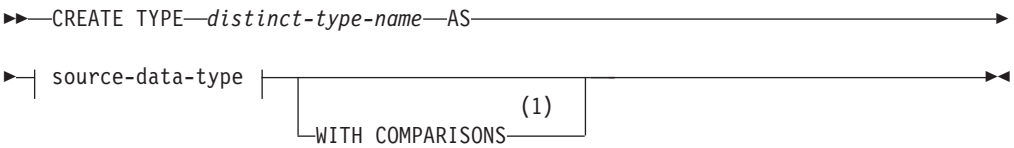
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

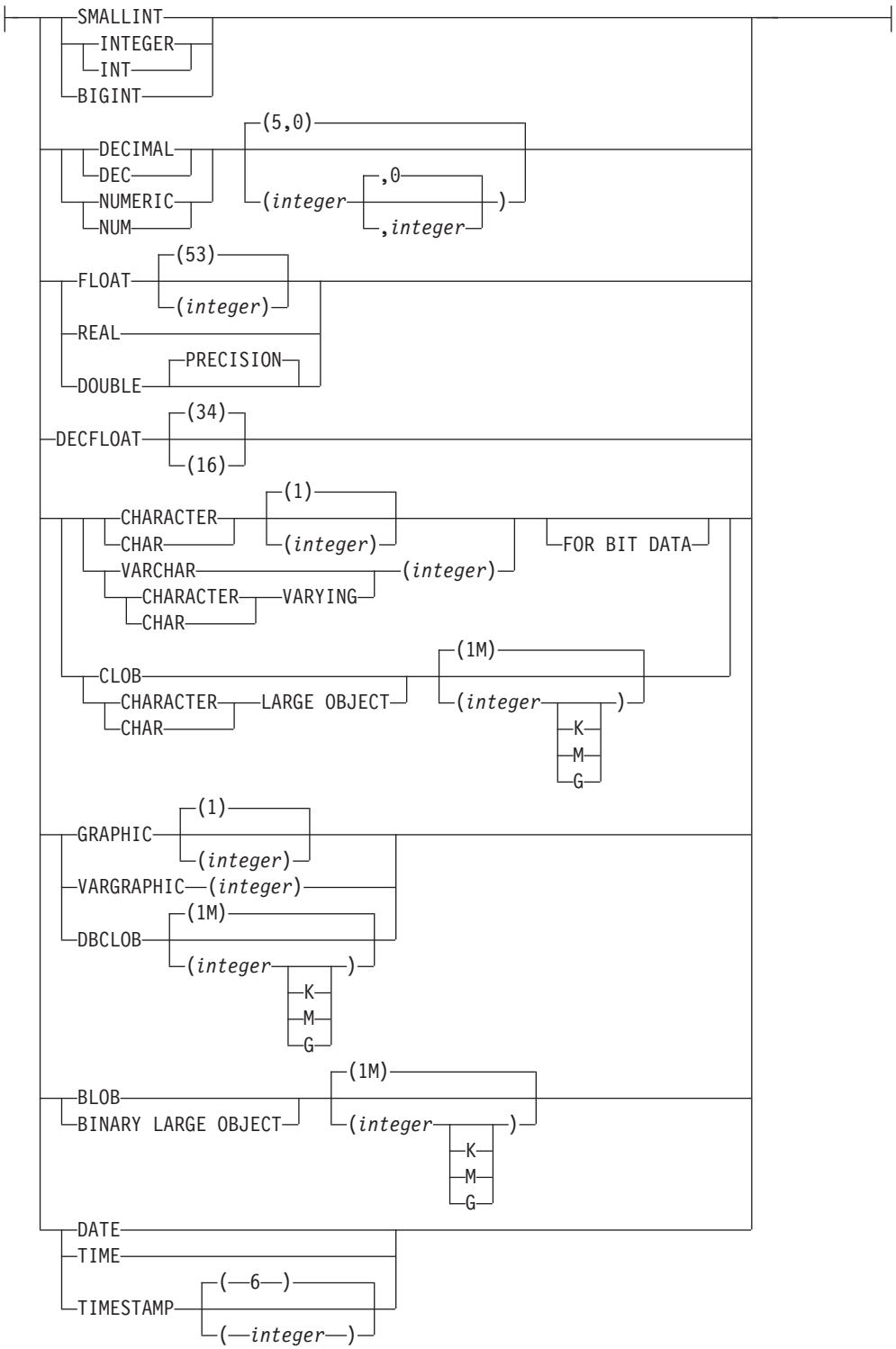
- 구별 유형의 스키마 이름이 기존 스키마를 참조하지 않는 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 구별 유형의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

구문

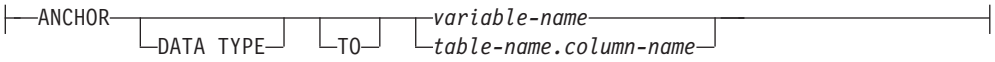


source-data-type:

내장 유형:



anchored-data-type:



주:

1 비교가 지원되지 않기 때문에 LOB를 제외한 모든 소스 데이터 유형에 대해서는 필수입니다.

설명

distinct-type-name

구별 유형의 이름을 지정합니다. 내재적 또는 명시적인 규정자를 포함한, 이름은 현재 서버에 이미 존재하는 다른 유형(내장 또는 사용자 정의 유형)을 식별해서는 안됩니다. 규정되지 않은 이름은 내장 데이터 유형, BOOLEAN, BINARY 또는 VARBINARY와 동일한 이름이어서는 안됩니다(SQLSTATE 42918). 또한 규정되지 않은 이름은 ARRAY, INTERVAL 또는 ROWID일 수 없습니다.

동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 규정된 형식은 마침표와 SQL ID 다음에 오는 *schema-name*입니다.

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *distinct-type-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

두 부분으로 구성된 *distinct-type-name*이 지정된 경우, 스키마 이름은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

source-data-type

구별 유형의 내부 표현에 대한 기준으로 사용되는 데이터 유형을 지정합니다. 데이터 유형은 내장 데이터 유형이어야 합니다. 내장 데이터 유형에 관한 자세한 정보는 『CREATE TABLE』을 참조하십시오. 소스 데이터 유형은 XML 또는 ARRAY 유형일 수 없습니다(SQLSTATE 42601). 플랫폼에서의 응용프로그램 이식성에 대해 다음 권장된 데이터 유형 이름을 사용합니다.

- FLOAT 대신 DOUBLE 또는 REAL
- NUMERIC 대신 DECIMAL
- LONG VARCHAR 대신 VARCHAR, BLOB 또는 CLOB
- LONG VARGRAPHIC 대신 VARGRAPHIC 또는 DBCLOB

anchored-data-type

데이터 유형을 판별하는 데 사용되는 다른 오브젝트를 식별합니다. 데이터 유형을 바로 지정할 때 적용되는 제한사항이 앵커 오브젝트의 데이터 유형에 동일하게 적용됩니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

ROW 또는 CURSOR가 아닌 내장 유형인 데이터 유형을 가진 전역 변수를 식별합니다. 전역 변수의 데이터 유형은 구별 유형의 소스 데이터 유형으로 사용됩니다.

table-name.column-name

내장 유형으로 식별되어야 하는 데이터 유형을 포함한 기존 테이블이나 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 구별 유형의 소스 데이터 유형으로 사용됩니다.

WITH COMPARISONS

구별 유형의 두 인스턴스를 비교하기 위해 시스템에서 생성한 비교 연산자가 작성되도록 지정합니다. 소스 데이터 유형이 BLOB, CLOB 또는 DBCLOB인 경우에는 이 키워드를 지정해서는 안됩니다. 지정하면 경고(SQLSTATE 01596)가 리턴되고 비교 연산자가 생성되지 않습니다. 이외의 모든 소스 데이터 유형의 경우, WITH COMPARISONS 키워드는 선택적이지만 WITH COMPARISONS절이 지정되지 않으면 시스템 작성 비교 연산자가 작성됩니다.

규칙

- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다 (SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.

주

- ***특권***: 사용자 정의 유형의 정의자는 구별 유형에 대하여 자동으로 생성된 모든 함수에 대해 EXECUTE 특권을 수신합니다.

  CREATE TYPE(구별)문을 실행하는 동안 자동으로 생성된 모든 함수에 대한 EXECUTE 특권은 PUBLIC에 부여됩니다.
- 아직 존재하지 않는 스키마 이름을 사용하여 구별 유형을 작성할 경우 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가지고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 소스 유형 간의 캐스트를 위해 다음과 같은 함수가 생성됩니다.
  - 구별 유형에서 소스 유형으로 변환되는 하나의 함수
  - 소스 유형에서 구별 유형으로 변환되는 하나의 함수
  - 소스 유형이 SMALLINT인 경우 INTEGER에서 구별 유형으로 변환되는 하나의 함수

- 소스 유형이 CHAR인 경우 VARCHAR에서 구별 유형으로 변환되는 하나의 함수
- 소스 유형이 GRAPHIC인 경우 VARGRAPHIC에서 구별 유형으로 변환되는 하나의 함수

일반적으로 이 함수의 형식은 다음과 같습니다.

```
CREATE FUNCTION source-type-name (distinct-type-name)
  RETURNS source-type-name ...

CREATE FUNCTION distinct-type-name (source-type-name)
  RETURNS distinct-type-name ...
```

소스 유형이 매개변수화된 유형인 경우, 구별 유형에서 소스 유형으로 변환되는 함수는 매개변수가 없는 소스 유형의 이름을 함수 이름으로 가지게 됩니다(자세한 내용은 표 25 참조). 이 함수의 리턴 값 유형에는 CREATE TYPE(구별)문에 제공되는 매개변수가 포함됩니다. 소스 유형에서 구별 유형으로 변환되는 함수는 소스 유형이면서 매개변수를 포함하는 입력 매개변수를 갖습니다. 예를 들면, 다음과 같습니다.

```
CREATE TYPE T_SHOESIZE AS CHAR(2)
  WITH COMPARISONS

CREATE TYPE T_MILES AS DOUBLE
  WITH COMPARISONS
```

다음 함수가 생성됩니다.

```
FUNCTION CHAR (T_SHOESIZE) RETURNS CHAR (2)


FUNCTION T_SHOESIZE (CHAR (2))
RETURNS T_SHOESIZE

FUNCTION DOUBLE (T_MILES) RETURNS DOUBLE

FUNCTION T_MILES (DOUBLE) RETURNS T_MILES
```

생성된 캐스트 함수의 스키마는 구별 유형의 스키마와 같습니다. 이 이름 및 동일한 시그니처를 갖고 있는 다른 함수가 데이터베이스에 존재하지 않을 수 있습니다(SQLSTATE 42710).

다음 표는 사전 정의된 모든 데이터 유형에 대해 구별 유형에서 소스 유형으로, 소스 유형에서 구별 유형으로 변환되는 함수들의 이름을 보여 줍니다.

표 25. 구별 유형에서의 CAST 함수

소스 유형 이름	함수 이름	매개변수	리턴 유형
CHAR	*distinct-type-name*	CHAR(*n*)	*distinct-type-name*
	CHAR	*distinct-type-name*	CHAR(*n*)
	distinct-type-name	VARCHAR(*n*)	*distinct-type-name*

표 25. 구별 유형에서의 CAST 함수 (계속)

소스 유형 이름	함수 이름	매개변수	리턴 유형
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*)
BLOB	*distinct-type-name*	BLOB(*n*)	*distinct-type-name*
	BLOB	*distinct-type-name*	BLOB(*n*)
GRAPHIC	*distinct-type-name*	GRAPHIC(*n*)	*distinct-type-name*
	GRAPHIC	*distinct-type-name*	GRAPHIC(*n*)
	distinct-type-name	VARGRAPHIC(*n*)	*distinct-type-name*
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*)
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*	DECIMAL(*p,s*)	*distinct-type-name*
	DECIMAL	*distinct-type-name*	DECIMAL(*p,s*)
REAL	*distinct-type-name*	REAL	*distinct-type-name*
	distinct-type-name	DOUBLE	*distinct-type-name*
	REAL	*distinct-type-name*	REAL
FLOAT(*n*), *n*<=24	*distinct-type-name*	REAL	*distinct-type-name*
	distinct-type-name	DOUBLE	*distinct-type-name*
	REAL	*distinct-type-name*	REAL
FLOAT(*n*), *n*>24	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
FLOAT	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
DOUBLE	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
DOUBLE PRECISION	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
DECFLOAT	*distinct-type-name*	DECFLOAT(*n*)	*distinct-type-name*
	DECFLOAT	*distinct-type-name*	DECFLOAT(*n*)

표 25. 구별 유형에서의 CAST 함수 (계속)

소스 유형 이름	함수 이름	매개변수	리턴 유형
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(*p*	*distinct-type-name*
	TIMESTAMP	*distinct-type-name*	TIMESTAMP(*p*)

주: 이식 가능한 응용프로그램에 대해 사용자 정의 유형을 작성할 때 NUMERIC과 FLOAT는 사용하지 않는 것이 좋습니다. 대신 DECIMAL 및 DOUBLE을 사용해야 합니다.

위 표에 설명된 함수는 구별 유형이 정의될 때 자동으로 생성되는 함수입니다. 따라서 CREATE FUNCTION문을 사용하여 구별 유형에 대한 사용자 정의 함수를 등록하고 이 사용자 정의 함수(UDF)가 적절한 내장 함수의 소스로 사용되어야 모든 내장 함수(AVG, MAX, LENGTH 등)가 구별 유형에 대해 지원됩니다. 특히, 내장 컬럼 함수를 소스로 사용하는 사용자 정의 함수를 등록할 수 있습니다.

WITH COMPARISONS절을 사용하여 구별 유형이 작성될 때 시스템 생성 비교 연산자가 작성됩니다. 이러한 비교 연산자를 작성하면 SYSCAT.ROUTINES 카탈로그 뷰에 새 함수에 대한 항목이 생성됩니다.

SQL문에서 이러한 연산자와 캐스트 함수를 성공적으로 사용하기 위해서는 구별 유형의 스키마 이름이 SQL 경로 또는 FUNCPATH BIND 옵션에 포함되어야 합니다.

- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - CREATE TYPE 대신 CREATE DISTINCT TYPE을 지정할 수 있습니다.
  - LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형 및 캐스트 함수가 지원되지만 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. WITH COMPARISONS절은 LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형을 계속 지원하지 않습니다.

예:

예 1: INTEGER 데이터 유형을 기반으로 하는 구별 유형 SHOESIZE를 작성하십시오.

```
CREATE TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

그러면, 비교 연산자(=, <>, <, <=, >, >=)와 두 개의 캐스트 함수, INTEGER를 리턴하는 INTEGER(SHOESIZE)와 SHOESIZE를 리턴하는 SHOESIZE(INTEGER)도 생성됩니다.

예 2: DOUBLE 데이터 유형을 기반으로 하는 구별 유형 MILES를 생성하십시오.

```
CREATE TYPE MILES AS DOUBLE WITH COMPARISONS
```

그러면, 비교 연산자(=, <>, <, =, >, >=)와 두 개의 캐스트 함수 DOUBLE을 리턴하는 DOUBLE(MILES)와 MILES를 리턴하는 MILES(DOUBLE)도 생성됩니다.

CREATE TYPE(행)

CREATE TYPE(행)문은 행 유형을 정의합니다. 행 유형에는 데이터 행을 구성하는 연관된 데이터 유형이 있는 하나 이상의 필드가 포함됩니다.

호출

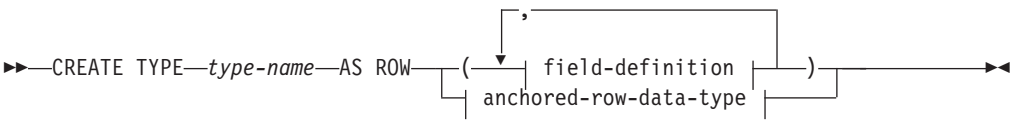
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 행 유형의 스키마 이름이 기존 스키마를 참조하지 않는 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 행 유형의 스키마 이름이 기존의 스키마를 참조하는 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

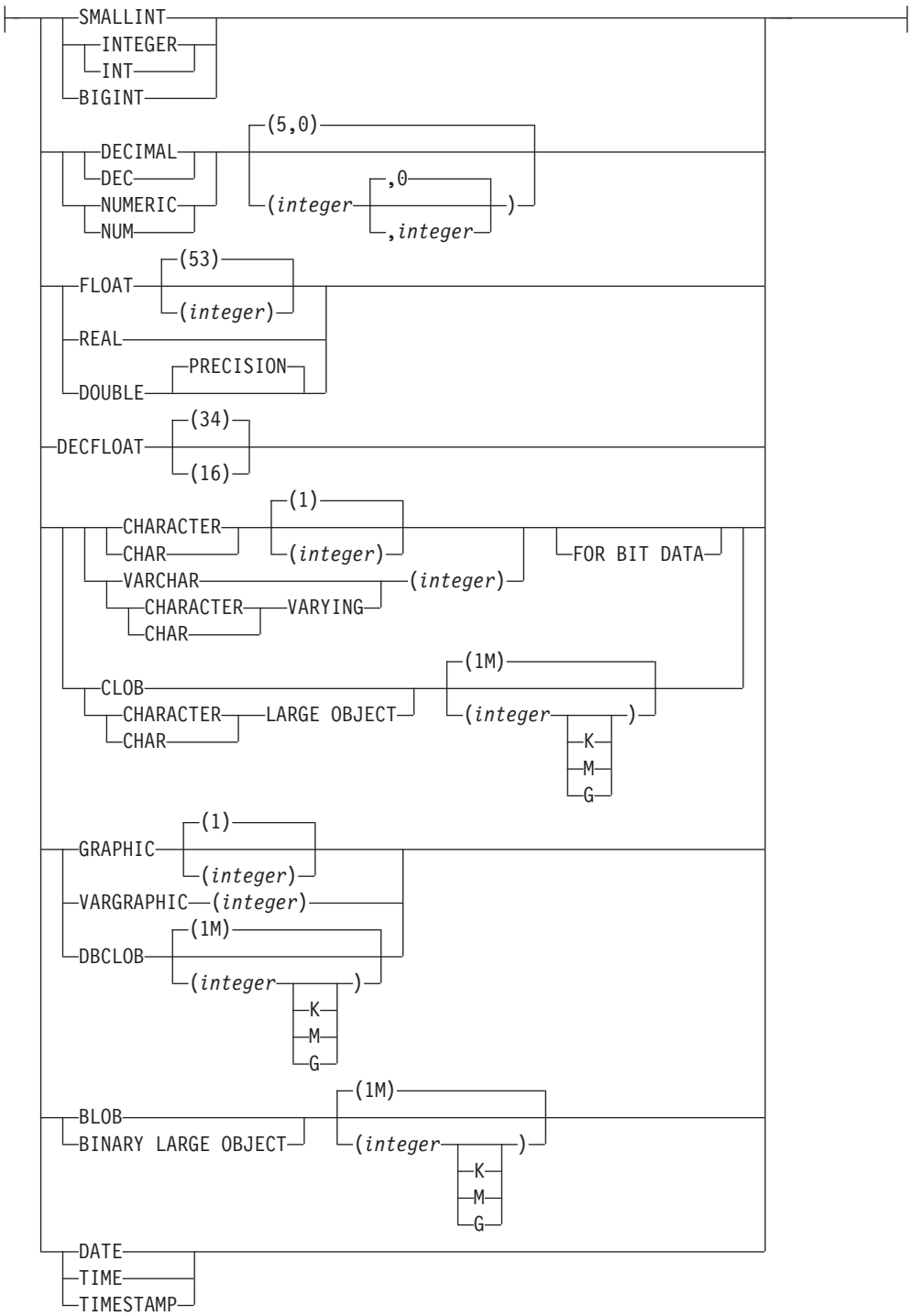
구문



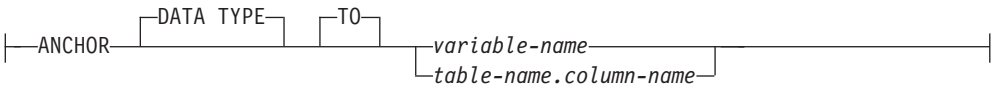
field-definition:

data-type:

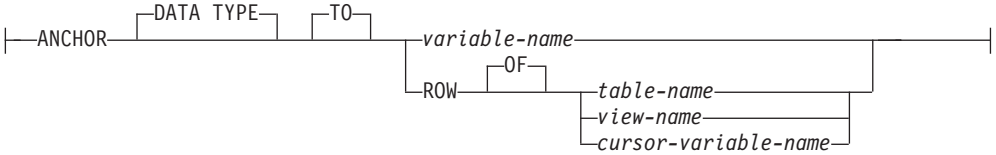
built-in-type:



anchored-non-row-data-type:



anchored-row-data-type:



설명

type-name

유형의 이름을 지정합니다. 이름(내재된 또는 명시적 규정자 포함)은 카탈로그에 이미 기술된 다른 유형(내장, 구조화, 배열, 행 또는 구별)을 식별해서는 안됩니다. 규정되지 않은 이름은 내장 데이터 유형 이름이나 BOOLEAN과 같을 수 없습니다(SQLSTATE 42918).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *type-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

두 부분으로 구성된 *type-name*을 지정하면, 스키마 이름은 'SYS'로 시작할 수 없습니다. 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 42939).

field-definition

행 유형의 필드를 정의합니다.

field-name

행 유형 내에서 필드의 이름을 지정합니다. 이름은 이 행 유형의 다른 필드와 같을 수 없습니다(SQLSTATE 42711).

data-type

필드의 데이터 유형을 지정합니다. 데이터 유형은 내장 유형 또는 구별 유형이 될 수 있습니다.

anchored-non-row-data-type

데이터 유형을 판별하기 위해 사용되는 다른 오브젝트를 식별합니다. 앵커 오브젝트의 데이터 유형에는 데이터 유형을 직접 지정할 때 적용되는 것과 동일한 제한사항이 있습니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

지원되는 행 필드 데이터 유형인 데이터 유형으로 전역 변수를 식별합니다. 전역 변수의 데이터 유형은 필드에 대한 데이터 유형으로 사용됩니다.

table-name.column-name

내장 유형 또는 구별 유형인 데이터 유형으로 기존 테이블 또는 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 필드의 데이터 유형으로 사용됩니다.

anchored-row-data-type

행 필드로 사용하기 위해 다른 오브젝트로부터 행 정보를 식별합니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name

전역 변수를 식별합니다. 참조된 변수의 데이터 유형은 행 유형이어야 하며 행 유형의 데이터 유형으로 사용됩니다.

ROW OF *table-name* **또는** *view-name*

*table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뷰의 컬럼 이름 및 컬럼 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 앵커 오브젝트 컬럼의 데이터 유형에는 필드 데이터 유형에 적용되는 것과 동일한 제한사항이 있습니다.

ROW OF *cursor-variable-name*

*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 있는 필드의 행을 지정합니다. 지정된 커서 변수는 다음 중 하나여야 합니다(SQLSTATE 428HS).

- 강하게 유형 지정된 커서 데이터 유형이 있는 전역 변수
- 모든 결과 컬럼이 이름 지정된 *select-statement*를 지정하는 CONSTANT 절로 작성되거나 선언되어 약하게 유형 지정된 커서 데이터 유형이 있는 전역 변수

규칙

- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.

주

- ***행 유형 사용법:*** 행 유형은 다음 데이터 유형으로만 사용할 수 있습니다.
  - 복합 SQL(컴파일된)문의 로컬 변수
  - SQL 루틴의 매개변수
  - SQL 함수의 리턴 유형
  - 배열 유형의 요소

  - 사용자 정의 커서 유형
  - 전역 변수
- 행 유형을 사용하여 정의된 변수 또는 매개변수는 복합 SQL(컴파일된) 명령문에서만 사용할 수 있습니다.

예 :

- DEPARTMENT 테이블 컬럼을 기초로 하는 행 유형을 작성하십시오.

```
CREATE TYPE DEPTROW AS ROW (DEPTNO   VARCHAR(3),
                            DEPTNAME VARCHAR(29),
                            MGRNO    CHAR(6),
                            ADMRDEPT CHAR(3),
                            LOCATION CHAR(16))
```

CREATE TYPE(구조화)

CREATE TYPE문은 사용자 정의 구조화된 유형을 정의합니다. 사용자 정의 구조화된 유형에는 제로 또는 그 이상의 속성이 포함될 수 있습니다. 구조화된 유형은 속성이 슈퍼 유형으로부터 상속받을 수 있도록 하는 부속 유형이 될 수 있습니다. 명령문을 성공적으로 실행하면 속성의 값을 검색하고 갱신하기 위한 메소드가 생성됩니다. 또한 한 컬럼에 사용되는 구조화된 유형의 인스턴스를 구성하고 참조 유형 및 표시 유형 간의 캐스팅 및 참조 유형의 비교 연산자(=, <>, <, <==, > 및 >=) 지원을 위한 함수도 생성됩니다.

CREATE TYPE문은 사용자 정의 구조화된 유형과 함께 사용될 사용자 정의 메소드에 대한 메소드 스펙을 정의하기도 합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

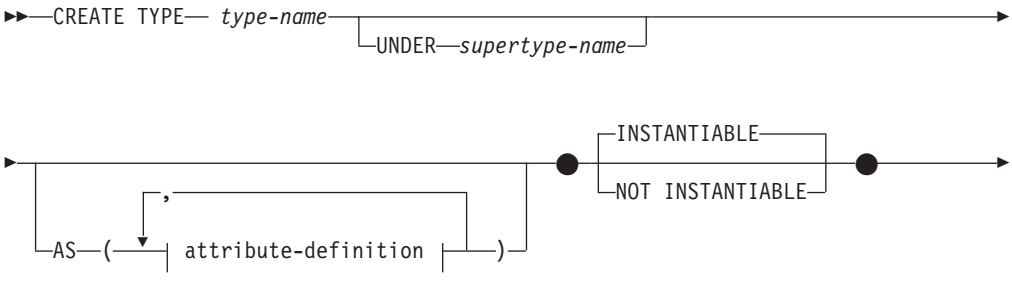
권한 부여

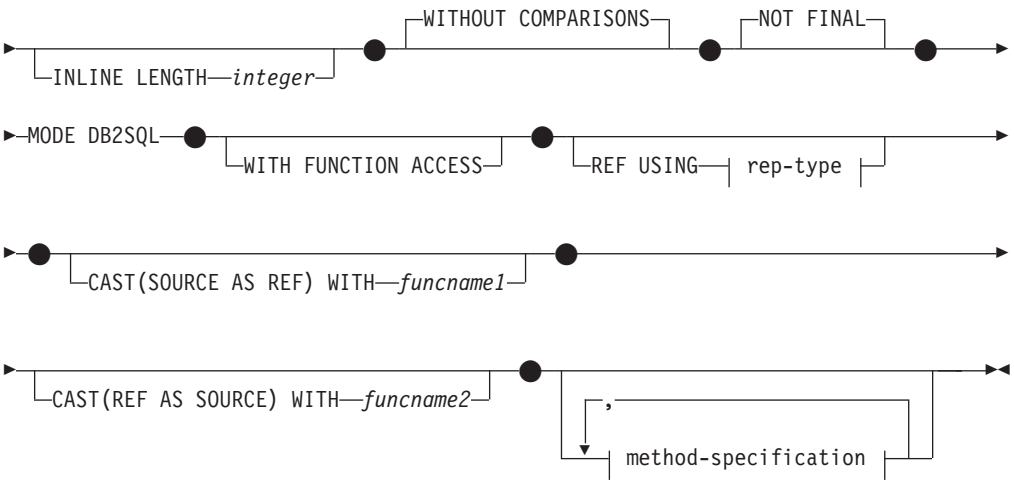
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 유형의 스키마 이름이 기존 스키마를 참조하지 않는 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 유형의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

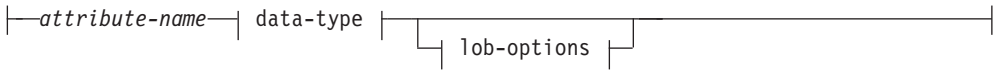
UNDER가 지정되고 명령문의 권한 부여 ID가 자료형 계층의 루트 유형 소유자와 동일하지 않을 경우, DBADM 권한이 필요합니다.

구문

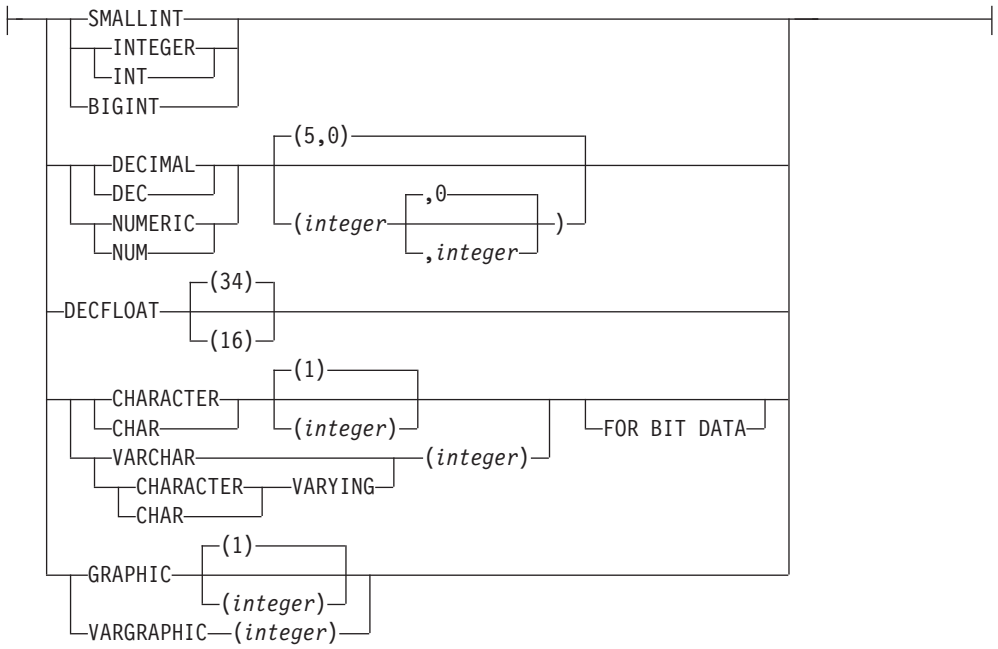




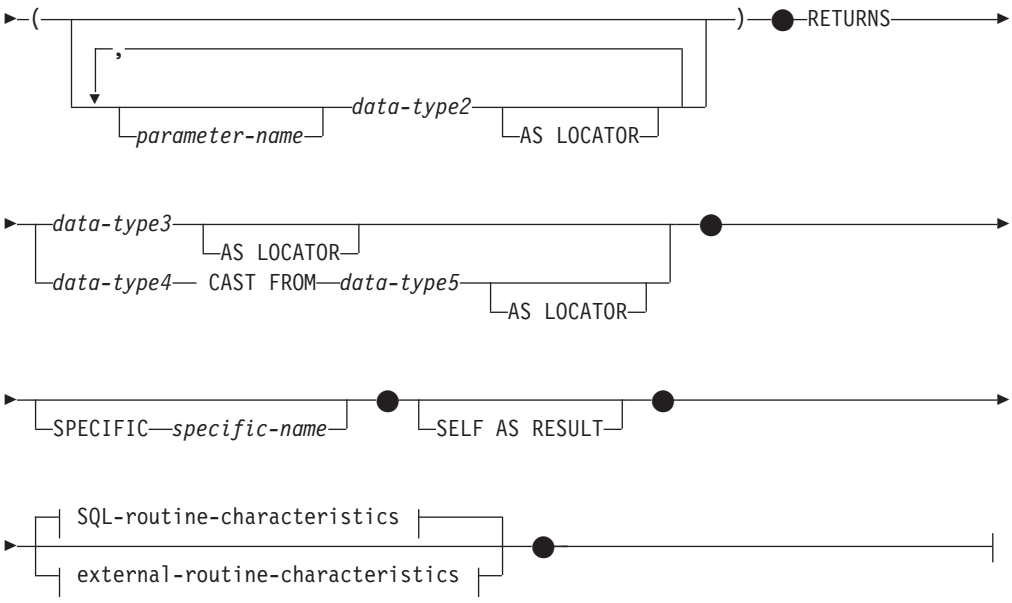
attribute-definition:



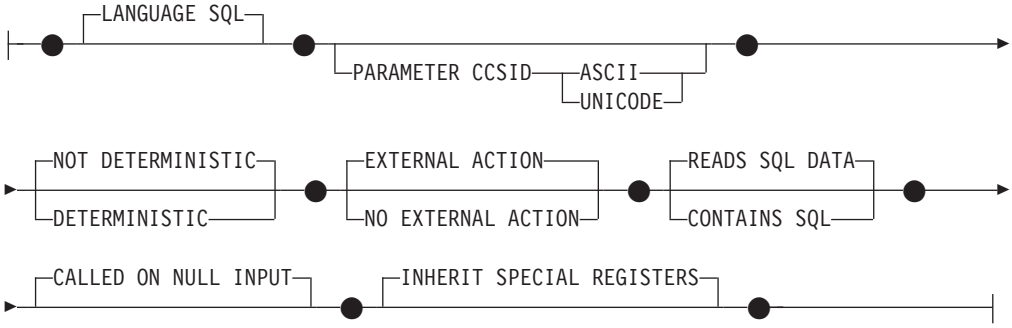
rep-type:



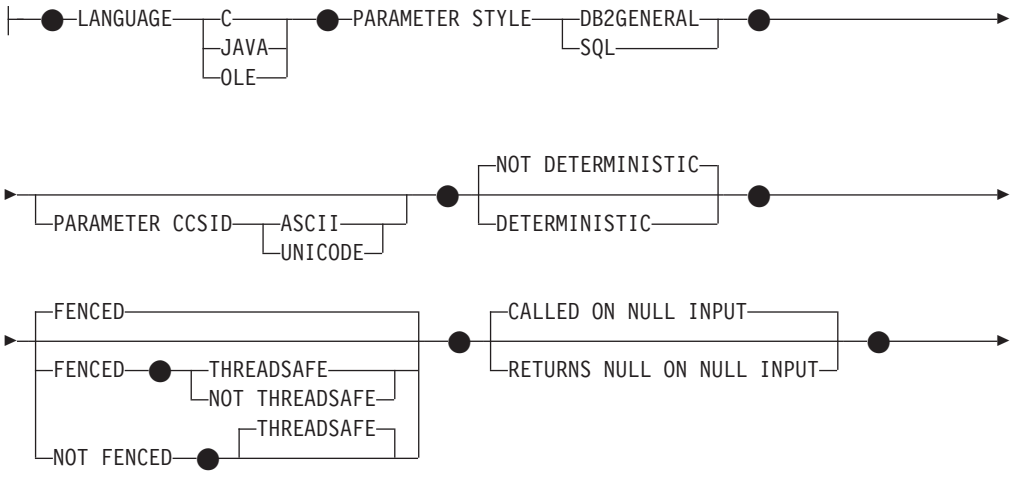
method-specification:

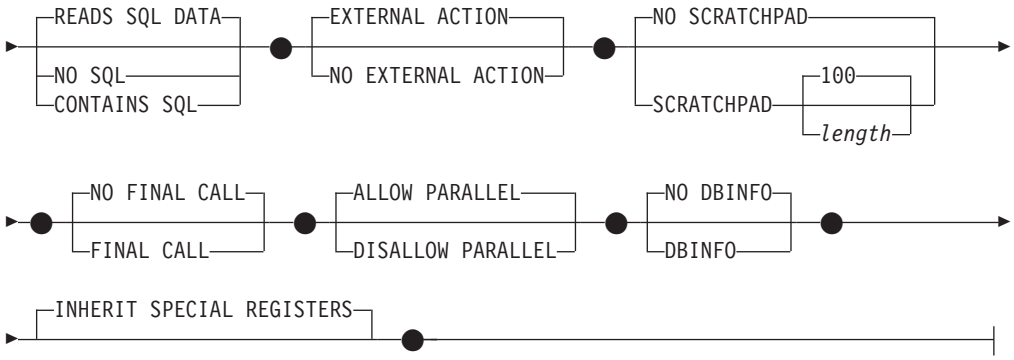


SQL-routine-characteristics:



external-routine-characteristics:





설명

type-name

유형의 이름을 지정합니다. 내재적 또는 명시적인 규정자를 포함한, 이름은 현재 서버에 이미 존재하는 다른 유형(내장, 구조화된 유형 또는 구별 유형)을 식별해서는 안됩니다. 규정되지 않은 이름은 내장 데이터 유형, BINARY, VARBINARY 또는 BOOLEAN과 동일한 이름이어서는 안됩니다(SQLSTATE 42918). 규정화되지 않은 이름은 ARRAY, INTERVAL 또는 ROWID일 수 없습니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다.

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *type-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

두 부분으로 구성된 *type-name*이 지정된 경우, 스키마 이름은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

UNDER *supertype-name*

이 구조화된 유형이 지정된 *supertype-name* 아래에 있는 부속 유형이 되도록 지정합니다. *supertype-name*은 기존의 구조화된 유형을 식별해야 합니다(SQLSTATE 42704). 스키마 이름없이 *supertype-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형이 해석됩니다. 구조화된 유형에는 슈퍼 유형의 모든 속성과 그 뒤에 *attribute-definition*에 있는 추가 속성이 포함됩니다.

attribute-definition

구조화된 유형의 속성을 정의합니다.

attribute-name

속성의 이름입니다. *attribute-name*은 이 구조화된 유형의 다른 속성 또는 슈퍼 유형과 같아서는 안됩니다(SQLSTATE 42711).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *attribute-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

data-type

속성의 데이터 유형입니다. XML이 아닌 『CREATE TABLE』에 나열된 데이터 유형 중 하나입니다(SQLSTATE 42601). 데이터 유형은 기존의 데이터 유형을 식별해야 합니다(SQLSTATE 42704). 스키마 이름없이 *data-type*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형이 해석됩니다. 다양한 데이터 유형에 대한 설명은 『CREATE TABLE』에 있습니다. 속성 데이터 유형이 참조 유형인 경우, 참조의 목표 유형은 기존의 구조화된 유형이어야 하며 그렇지 않으면 이 명령문에 의해 작성됩니다(SQLSTATE 42704).

런타임시 유형의 인스턴스가 직접 또는 간접적으로 같은 유형의 인스턴스 또는 그의 부속 유형 중 하나의 부속 유형을 포함하도록 허용하는 유형 정의를 막기 위해, 이 속성 유형 중 하나가 직접 또는 간접적으로 자신을 사용하도록 유형을 정의할 수 없습니다(SQLSTATE 428EP).

lob-options

LOB 유형과 연관된 옵션(또는 LOB 유형을 기반으로 하는 구별 유형)을 지정합니다. *lob-options*에 대한 자세한 내용은 『CREATE TABLE』을 참조하십시오.

INSTANTIABLE 또는 NOT INSTANTIABLE

구조화된 유형의 인스턴스를 작성할 수 있는지 여부를 판별합니다. 다음은 이러한 인스턴스화되는 구조화된 유형을 포함하지 않습니다.

- 인스턴스를 작성할 수 없는 유형에 대한 컨스트럭터 함수가 생성되지 않습니다.
- 인스턴스를 작성할 수 없는 유형은 테이블이나 뷰의 유형으로 사용할 수 없습니다(SQLSTATE 428DP).
- 인스턴스를 작성할 수 없는 유형은 컬럼의 유형으로 사용할 수 있습니다(인스턴스를 작성할 수 있는 부속 유형의 널(NULL) 값 또는 인스턴스만 컬럼에 삽입할 수 있습니다).

인스턴스를 작성할 수 없는 유형의 인스턴스를 작성하려면 인스턴스를 작성할 수 있는 부속 유형을 작성해야 합니다. NOT INSTANTIABLE을 지정하지 않으면 새 유형의 인스턴스를 작성할 수 없습니다.

INLINE LENGTH *integer*

이 옵션은 테이블 행에 값의 나머지 부분을 인라인으로 저장하기 위한 구조화된 유형 컬럼 인스턴스의 최대 크기(바이트)를 나타냅니다. 지정된 인라인 길이보다 긴 구조화된 유형이나 해당 부속 유형의 인스턴스는 LOB 값의 처리 방법과 유사하게 기본 테이블 행과는 별도로 저장됩니다.

지정된 INLINE LENGTH가 새로 작성된 유형(32바이트 + 속성당 10바이트)에 대한 컨스트럭터 함수의 결과 크기보다 작고 292바이트보다 작으면 오류가 발생합니다(SQLSTATE 429B2). 속성 수에는 해당 유형의 슈퍼 유형에서 상속된 모든 속성이 포함된다는 점에 유의하십시오.

유형에 대한 INLINE LENGTH는 지정되었거나 디폴트값이거나 관계없이 구조화된 유형을 사용하는 컬럼의 기본 인라인 길이입니다. CREATE TABLE 수행시 이 디폴트값을 겹쳐쓸 수 있습니다.

INLINE LENGTH는 구조화된 유형이 유형이 지정된 테이블의 유형으로 사용되는 경우에는 아무 의미가 없습니다.

구조화된 유형의 디폴트 INLINE LENGTH는 시스템에 의해 계산됩니다. 아래 제공된 공식에서는 다음 항목이 사용됩니다.

short attribute
: 다음 데이터 유형 중 하나를 갖는 속성을 참조합니다. SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, FLOAT, DATE 또는 TIME. 이러한 유형을 기반으로 하는 구별 유형이나 참조 유형도 포함됩니다.

non-short attribute
: 이러한 데이터 유형을 기반으로 하는 나머지 데이터 유형이나 구별 유형의 속성을 참조합니다.

시스템은 다음과 같이 디폴트 인라인 길이를 계산합니다.

1. 다음 공식을 사용하여 short가 아닌 속성에 대한 추가 스페이스 요구사항을 결정합니다.

   *space_for_non_short_attribute*s = SUM(*attributelength* + n)

   n은 다음과 같이 정의됩니다.
   - 중첩된 구조화된 유형 속성의 경우 0바이트
   - 비LOB 속성의 경우 2바이트
   - LOB 속성의 경우 9바이트

   *attributelength*는 827 페이지의 표 26에 나와 있는 대로 속성에 대해 지정된 데이터 유형을 기반으로 합니다.
2. 다음 공식을 사용하여 총 디폴트 인라인 길이를 계산합니다.

   default_length(*structured_type*) = (*number_of_attributes* * 10) + 32 + *space_for_non-short_attributes*

   *number_of_attributes*는 해당 슈퍼 유형에서 상속된 속성을 포함하여 구조화된 유형에 대한 총 속성 수입니다. 그러나 *number_of_attributes*에는 *structured_type*의 부속 유형에 대해 정의된 속성이 포함되지 않습니다.

표 26. 속성 데이터 유형에 대한 바이트 수

속성 데이터 유형	바이트 수
DECIMAL	(*p*/2)+1의 정수 부분. 여기서 *p*는 정밀도입니다.
DECFLOAT(*n*)	*n*이 16이면, 바이트 수는 8이며 *n*이 34이면 바이트 수는 16입니다.
CHAR(*n*)	*n*
VARCHAR(*n*)	*n*
GRAPHIC(*n*)	*n* * 2
VARGRAPHIC(*n*)	*n* * 2
TIMESTAMP	10
LOB 유형	각 LOB 속성은 실제 값의 위치를 가리키는 구조화된 유형 인스턴스에 LOB 디스크립터를 가집니다. 디스크립터 크기는 LOB 속성에 정의된 최대 길이에 따라 변합니다(표 27 페이지 참조).
구별 유형	구별 유형의 소스 유형 길이
참조 유형	참조 유형을 기반으로 하는 내장 데이터 유형 길이
구조화된 유형	inline_length(*attribute_type*)

표 27. 최대 LOB 길이 함수로써의 LOB 디스크립터 크기

최대 LOB 길이	LOB 디스크립터 크기
1024	68
8192	92
65 536	116
524 000	140
4 190 000	164
134 000 000	196
536 000 000	220
1 070 000 000	252
1 470 000 000	276
2 147 483 647	312

WITHOUT COMPARISONS

구조화된 유형 인스턴스에 대해 지원되는 비교 함수가 없음을 나타냅니다.

NOT FINAL

구조화된 유형이 슈퍼 유형으로 사용될 수 있음을 나타냅니다.

MODE DB2SQL

이 절은 필수이며 이 유형에 대한 컨스트럭터 함수의 직접 호출을 허용합니다.

WITH FUNCTION ACCESS

이 유형 및 부속 유형의 모든 메소드(이후에 작성되는 메소드 포함)는 함수 표기를 사용하여 액세스할 수 있음을 나타냅니다. 이 절은 구조화된 자료형 계층의 루트 유형에 대해서만 지정할 수 있으며, UNDER절은 지정하지 않습니다(SQLSTATE

42613). 메소드 호출 표기보다는 이 표기 양식을 선호하는 응용프로그램에 대해 함수 표기 사용이 가능하도록 이 절이 제공됩니다.

REF USING *rep-type*

이 구조 유형과 모든 부속 유형에 대한 표현(하위 데이터 유형)으로 사용된 내장 데이터 유형을 정의합니다. 이 절은 구조화된 자료형 계층의 루트 유형에 대해서만 지정할 수 있으며, UNDER절은 지정되지 않습니다(SQLSTATE 42613). *rep-type*은 REAL, FLOAT, DECFLOAT, BLOB, CLOB, DBCLOB, 배열 유형 또는 구조화된 유형일 수 없으며 32 672바이트보다 작거나 같아야 합니다(SQLSTATE 42613).

구조화된 자료형 계층의 루트 유형에 대해 이 절을 지정하지 않으면, REF USING VARCHAR(16) FOR BIT DATA로 간주됩니다.

CAST (SOURCE AS REF) WITH *funcname1*

*rep-type*의 데이터 유형을 가진 값을 이 구조화된 유형의 참조 유형으로 캐스트하는 시스템 생성 함수의 이름을 정의합니다. 스키마 이름은 *funcname1*의 일부로 지정할 수 없습니다(SQLSTATE 42601). 캐스트 함수는 구조화된 유형과 같은 스키마에서 생성됩니다. 이 절을 지정하지 않을 경우, *funcname1*에 대한 디폴트값은 *type-name*(구조화된 유형의 이름)입니다. *funcname1(rep-type)*과 일치하는 함수 시그니처가 같은 스키마에 있을 수 없습니다(SQLSTATE 42710).

CAST (REF AS SOURCE) WITH *funcname2*

이 구조화된 유형에 대한 참조 유형을 *rep-type*의 데이터 유형으로 캐스트하는 시스템 생성 함수의 이름을 정의합니다. 스키마 이름은 *funcname2*의 일부로 지정할 수 없습니다(SQLSTATE 42601). 캐스트 함수는 구조화된 유형과 같은 스키마에서 생성됩니다. 이 절을 지정하지 않을 경우, *funcname2*에 대한 디폴트값은 *rep-type*(표현 유형의 이름)입니다.

method-specification

이 유형에 대한 메소드를 정의합니다. 메소드는 실제로 CREATE METHOD문이 포함된 본문이 제공된 이후에 사용할 수 있습니다(SQLSTATE 42884).

OVERRIDING

정의되는 메소드가 정의되는 유형의 슈퍼 유형의 메소드를 겹쳐쓰도록 지정합니다. 쓰기를 사용하면 부속 유형의 메소드를 다시 구현할 수 있으므로 매우 구체적인 기능을 제공합니다. 겹쳐쓰기는 다음 메소드 유형에 대해서는 지원되지 않습니다.

- 테이블 및 행 메소드
- PARAMETER STYLE JAVA로 선언되는 외부 메소드
- 인덱스 확장에 술어로 사용될 수 있는 메소드
- 시스템 생성 mutator 메소드 또는 observer 메소드

이런 메소드를 겹쳐쓰려고 하면 오류가 발생합니다(SQLSTATE 42745).

메소드가 유효한 겹쳐쓰기 메소드가 되려면 정의 중인 유형의 적절한 슈퍼 유형 중 하나에 대한 원본 메소드가 있어야 하고, 중첩 메소드와 원본 메소드 간에 다음과 같은 관계가 있어야 합니다.

- 정의 중인 메소드의 메소드 이름과 원래 메소드는 동등합니다.
- 정의 중인 메소드와 원래 메소드에는 같은 수의 매개변수가 있습니다.
- 정의 중인 메소드에 있는 각 매개변수의 데이터 유형과 원본 메소드에 있는 일치하는 매개변수의 데이터 유형이 같아야 합니다. 이 요구사항에서 내재된 SELF 매개변수는 제외됩니다.

이와 같은 메소드가 존재하지 않으면 오류가 발생합니다(SQLSTATE 428FV).

겹쳐쓰기 메소드는 원래 메소드로부터 다음 속성을 상속받습니다.

- 언어
- 결정론 표시
- 외부 조치 표시
- 메소드의 인수 중 하나가 널(NULL) 값일 경우 이 메소드를 호출해야 할지 여부를 나타내는 표시
- 결과 캐스트(원본 메소드에 지정된 경우)
- SELF AS RESULT 표시
- SQL 데이터 액세스 또는 CONTAINS SQL 표시
- 외부 메소드인 경우
  - 매개변수 스타일
  - 매개변수 및 결과의 로케이터 표시(원본 메소드에 지정된 경우)
  - FENCED, SCRATCHPAD, FINAL CALL, ALLOW PARALLEL, DBINFO 표시
  - INHERIT SPECIAL REGISTER, THREADSAFE 표시

method-name

정의할 메소드의 이름입니다. 규정되지 않은 SQL ID여야 합니다(SQLSTATE 42601). 메소드 이름은 CREATE TYPE에 사용된 스키마를 사용하여 내재적으로 규정됩니다.

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *method-name*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 및 비교 연산자가 이에 해당합니다.

일반적으로 시그니처에 약간의 차이가 있다면 둘 이상의 메소드에 대해 같은 이름을 사용할 수 있습니다.

parameter-name

매개변수 이름을 식별하십시오. SELF(메소드의 내재적 주제 매개변수 이름)일 수 없습니다(SQLSTATE 42734). 메소드가 SQL 메소드인 경우, 모든 매개변수가 이름을 가져야 합니다(SQLSTATE 42629). 선언되는 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 메소드의 매개변수 이름이 겹쳐쓰여지는 메소드에 있는 일치하는 매개변수의 이름과 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 428FV).

data-type2

각 매개변수의 데이터 유형을 지정합니다. 메소드가 수신할 각 매개변수에 대해 목록의 한 항목을 지정해야 합니다. 내재된 SELF 매개변수를 포함하여 매개변수는 90개를 초과할 수 없습니다. 이 한계를 초과하면 오류가 발생합니다(SQLSTATE 54023).

CREATE TABLE문의 컬럼 유형으로 지정할 수 있고 메소드를 작성하는 데 사용 중인 언어에 equivalent를 가진 SQL 데이터 유형 및 약어를 지정할 수 있습니다. SQL 데이터 유형과 호스트 언어 데이터 유형 사이의 맵핑에 대한 자세한 정보는 아래의 관련 주제 목록에서 해당 언어와 관련된 주제를 참조하십시오.

주: 문제가 발생한 SQL 데이터 유형이 구조화된 유형이라면 호스트 언어 데이터 유형에 맵핑되는 디폴트값이 없습니다. 사용자 정의 변환 함수를 사용하여 구조화된 유형과 호스트 언어 데이터 유형 간의 맵핑을 작성해야 합니다.

DECIMAL(또는 NUMERIC) 및 10진수 부동 소수점은 LANGUAGE C 및 OLE에 대해 유효하지 않습니다(SQLSTATE 42815).

XML 데이터 유형은 사용될 수 없습니다(SQLSTATE 42815).

REF를 지정할 수 있으나 정의된 범위는 가지지 않습니다. 메소드 본문에서 먼저 범위를 가지도록 캐스팅해야만 참조 유형을 경로 표현식에 사용할 수 있습니다. 마찬가지로, 먼저 범위를 가지도록 캐스팅해야만 메소드가 리턴하는 참조를 경로 표현식에 사용할 수 있습니다.

AS LOCATOR

LOB 유형 또는 LOB 유형을 기반으로 하는 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값이 아닌 LOB 로케이터가 메소드로 전달됨을 나타냅니다. 그러면 메소드에 전달된 바이트 수가 크게 절약되고 특히 값의 몇몇 바이트만 실제 메소드와 관련되는 경우 성능도 향상될 수 있습니다.

LOB 또는 LOB을 기반으로 하는 구별 유형이 아닌 다른 유형에 대해 AS LOCATOR를 지정할 경우 오류가 발생합니다(SQLSTATE 42601).

메소드가 FENCED이거나 LANGUAGE가 SQL인 경우에는 AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

선언되는 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 메소드에 있는 매개변수의 AS LOCATOR 표시가 겹쳐쓰여지는 메소드에 있는 일치하는 매개변수의 AS LOCATOR 표시와 같아야 합니다(SQLSTATE 428FV).

선언되는 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 메소드에 있는 매개변수의 FOR BIT DATA 표시가 겹쳐쓰여지는 메소드에 있는 일치하는 매개변수의 FOR BIT DATA 표시와 같아야 합니다(SQLSTATE 428FV). (SQLSTATE 428FV).

RETURNS

이 필수 절은 메소드의 결과를 식별합니다.

data-type3

메소드 결과의 데이터 유형을 지정합니다. 이 경우 앞의 data-type2에서 설명한 메소드 매개변수의 경우와 정확하게 같은 고려사항이 적용됩니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR 절을 추가할 수 있습니다. 이는 실제 값이 아닌 LOB 로케이터가 메소드로부터 전달됨을 나타냅니다.

LOB 또는 LOB을 기반으로 하는 구별 유형이 아닌 다른 유형에 대해 AS LOCATOR를 지정할 경우 오류가 발생합니다(SQLSTATE 42601).

메소드가 FENCED이거나 LANGUAGE가 SQL인 경우에는 AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

정의할 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 절을 지정할 수 없습니다(SQLSTATE 428FV).

메소드가 다른 메소드를 겹쳐쓸 경우, 겹쳐쓰여지는 메소드의 결과 데이터 유형이 구조화된 유형이면 *data-type3*가 겹쳐쓰여지는 메소드의 결과 데이터 유형의 부속 유형이어야 합니다. 그렇지 않은 경우 두 데이터 유형은 같아야 합니다(SQLSTATE 428FV).

data-type4 **CAST FROM** *data-type5*

메소드 결과의 데이터 유형을 지정합니다.

이 절은 메소드 코드로 리턴된 데이터 유형에서 호출 명령문으로 다른 데이터 유형을 리턴하는 데 사용됩니다. *data-type5*는 *data-type4* 매개변수에 캐스트할 수 있어야 합니다. 캐스트할 수 없는 경우 오류가 리턴됩니다(SQLSTATE 42880).

*data-type4*에 대한 길이, 정밀도 또는 스케일은 *data-type5*로부터 추론될 수 있으므로, *data-type4*에 대해 지정된 매개변수 작성 유형의 길이, 정밀도 또는 스

케일을 지정할 필요가 없습니다(아직은 허용됨). 대신 예를 들어 VARCHAR() 와 같이 빈 괄호를 사용할 수 있습니다. 매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

구별 유형은 *data-type5*에 지정된 유형으로는 유효하지 않습니다(SQLSTATE 42815). XML은 *data-type4* 또는 *data-type5*에 지정된 유형으로는 유효하지 않습니다(SQLSTATE 42815).

캐스트 조작은 변환 오류가 리턴될 수 있는 런타임 검사에도 종속됩니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR 절을 추가할 수 있습니다. 이는 실제 값이 아닌 LOB 로케이터가 메소드로부터 전달됨을 나타냅니다.

LOB 또는 LOB을 기반으로 하는 구별 유형이 아닌 다른 유형에 대해 AS LOCATOR를 지정할 경우 오류가 발생합니다(SQLSTATE 42601).

메소드가 FENCED이거나 LANGUAGE가 SQL인 경우에는 AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

정의할 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 절을 지정할 수 없습니다(SQLSTATE 428FV).

정의할 메소드가 다른 메소드를 겹쳐쓸 경우에는 FOR BIT DATA절을 지정할 수 없습니다(SQLSTATE 428FV).

SPECIFIC *specific-name*

정의할 메소드의 인스턴스에 고유한 이름을 제공합니다. 이 특정 이름은 메소드 본문을 작성하거나 메소드를 삭제하는 경우에 사용할 수 있지만, 메소드를 호출하는 데는 사용할 수 없습니다. *specific-name*의 규정되지 않은 형식은 SQL ID(최대 길이: 18)입니다. 규정된 형식은 마침표와 SQL ID가 뒤에 오는 스키마 이름입니다. 내재된 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버(AS)에 있는 다른 특정 메소드 이름을 식별해서는 안됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *method-name*과 같을 수 있습니다.

규정자를 지정하지 않을 경우 *type-name*에 사용된 규정자가 사용됩니다. 규정자를 지정할 경우, *type-name*의 명시적 또는 내재된 규정자와 같아야 하며 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

SELF AS RESULT

이 메소드를 유형 보존 메소드로 식별합니다. 이는 다음을 의미합니다.

- 선언된 리턴 유형은 선언된 주제 유형과 같아야 합니다(SQLSTATE 428EQ).
- SQL문이 컴파일되고 유형 보존 메소드로 해석되는 경우, 메소드 결과의 정적 유형은 주제 인수의 정적 유형과 같습니다.
- 결과의 동적 유형이 주제 인수의 동적 유형과 같도록 메소드를 구현해야 하며(SQLSTATE 2200G), 결과는 널(NULL)일 수 없습니다(SQLSTATE 22004).

정의할 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 절을 지정할 수 없습니다(SQLSTATE 428FV).

SQL-routine-characteristics

CREATE METHOD를 사용하여 이 유형에 대해 정의될 메소드 본문의 등록 정보를 지정합니다.

LANGUAGE SQL

이 절은 메소드가 단일 RETURN문을 포함하는 SQL로 작성됨을 나타내는 데 사용됩니다. 메소드 본문은 CREATE METHOD문을 사용하여 지정됩니다.

PARAMETER CCSID

SQL 메소드에 전달되고 SQL 메소드로부터 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은 경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII

문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031).

UNICODE

문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있도록 지정합니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE를 지정할 수 없습니다(SQLSTATE 56031).

NOT DETERMINISTIC 또는 DETERMINISTIC

이 선택적 절은 메소드가 항상 주어진 인수 값에 대해 같은 결과를 리턴하는지 여부(DETERMINISTIC) 또는 메소드가 결과에 영향을 주는 몇몇 상태 값에 따라 달라지는지 여부(NOT DETERMINISTIC)를 지정합니다. 즉, DETERMINISTIC 메소드는 항상 동일한 입력으로 계속되는 호출로부터 같은 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC을 지정하여 방지할 수 있습니다.

메소드 본문이 특수 레지스터에 액세스하거나 다른 비결정 루틴을 호출하는 경우 NOT DETERMINISTIC을 명시적으로 또는 내재적으로 지정해야 합니다 (SQLSTATE 428C2).

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

이 선택적 절은 메소드가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지 여부를 지정합니다. 외부 영향이 없는 메소드를 가정하는 최적화는 EXTERNAL ACTION을 지정하여 방지합니다. 메시지 전송, 벨 울림 또는 파일에 레코드 기록 등의 작업이 여기에 해당합니다.

READS SQL DATA 또는 CONTAINS SQL

실행할 수 있는 SQL문의 유형을 나타냅니다. 지원되지 않는 SQL문은 RETURN문이므로, 표현식이 서브쿼리인지 여부에 따라 구별을 수행해야 합니다.

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문을 메소드로 실행할 수 있음을 나타냅니다(SQLSTATE 42985). 별명은 SQL문에서 참조할 수 없습니다 (SQLSTATE 42997).

CONTAINS SQL

SQL 데이터를 읽거나 수정하지 않는 SQL문을 메소드로 실행할 수 있음을 나타냅니다(SQLSTATE 42985).

CALLED ON NULL INPUT

이 선택적 절은 인수가 널(NULL)인지 여부에 관계없이 사용자 정의 메소드가 호출됨을 나타냅니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나 널(NULL) 인수 값의 테스트는 메소드에서 수행합니다.

정의할 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 절을 지정할 수 없습니다 (SQLSTATE 428FV).

NULL CALL은 CALLED ON NULL INPUT과 동의어로 사용할 수 있습니다.

INHERIT SPECIAL REGISTERS

이 선택적 절은 메소드에 있는 갱신 가능한 특수 레지스터가 레지스터의 초기값을 호출 명령문의 환경에서 상속하도록 지정합니다. 커서의 Select문에서 호출되는 메소드의 경우에는 초기값을 커서가 열려 있는 환경에서 상속합니다. 중첩 오브젝트(예: 트리거 또는 뷰)에서 호출되는 루틴의 경우에는 초기값을 오브젝트 정의에서 상속하지 않고 런타임 환경에서 상속합니다.

특수 레지스터에 대한 변경사항은 함수 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 갱신할 수 없는 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 디폴트값으로 설정됩니다.

external-routine-characteristics

LANGUAGE

이 필수 절은 사용자 정의 메소드 본문이 작성되는 언어 인터페이스 규칙을 지정하는 데 사용됩니다.

C 데이터베이스 관리 프로그램이 사용자 정의 메소드를 C 함수처럼 호출합니다. 사용자 정의 메소드는 표준 ANSI C 프로토타입으로 정의된 C 언어 호출 및 링크 규칙에 따라야 합니다.

JAVA

데이터베이스 관리 프로그램이 사용자 정의 메소드를 Java 클래스의 메소드로 호출함을 의미합니다.

OLE

데이터베이스 관리 프로그램이 사용자 정의 메소드를 OLE 자동 오브젝트에 의해 표시되는 메소드처럼 호출합니다. 메소드는 *OLE 자동 프로그래머 참조서*에 설명된 대로 OLE 자동 데이터 유형 및 호출 메커니즘을 따라야 합니다.

LANGUAGE OLE은 Windows 32비트 운영 체제에 저장된 사용자 정의 메소드에 대해서만 지원됩니다. LANGUAGE OLE로 정의된 메소드에 대해서는 THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

PARAMETER STYLE

이 절은 매개변수를 메소드에 전달하고 메소드로부터 값을 리턴하도록 규칙을 지정하는 데 사용됩니다.

DB2GENERAL

Java 클래스의 메소드로 정의된 외부 메소드에 매개변수를 전달하고 메소드로부터 값을 리턴하는 규칙을 지정하는 데 사용됩니다. 이는 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

값 DB2GENRL은 DB2GENERAL의 동의어로 사용할 수 있습니다.

SQL

OLE 자동 오브젝트에 의해 표시되는 메소드 또는 C 언어 호출 및 링크 규칙을 따르는 외부 메소드에 매개변수를 전달하고 메소드로부터 값을 리턴하는 규칙을 지정하는 데 사용됩니다. 이것은 LANGUAGE C 또는 LANGUAGE OLE를 사용할 때 지정해야 합니다.

PARAMETER CCSID

외부 메소드에 전달되고 외부 메소드로부터 전달되는 모든 문자열 데이터에 사용되는 코드화 체계를 지정합니다. PARAMETER CCSID절을 지정하지 않은

경우, 유니코드 데이터베이스에서 디폴트값은 PARAMETER CCSID UNICODE이고 그 외의 모든 데이터베이스에서 디폴트값은 PARAMETER CCSID ASCII입니다.

ASCII
문자열 데이터가 데이터베이스 코드 페이지로 인코드되도록 지정합니다. 데이터베이스가 유니코드 데이터베이스일 경우, PARAMETER CCSID ASCII를 지정할 수 없습니다(SQLSTATE 56031).

UNICODE
문자 데이터는 UTF-8로 되어 있고 그래픽 데이터는 UCS-2로 되어 있도록 지정합니다. 데이터베이스가 유니코드 데이터베이스가 아닐 경우, PARAMETER CCSID UNICODE를 지정할 수 없습니다(SQLSTATE 56031).

이 절은 LANGUAGE OLE로 지정될 수 없습니다(SQLSTATE 42613).

DETERMINISTIC 또는 NOT DETERMINISTIC
이 선택적 절은 메소드가 항상 주어진 인수 값에 대해 같은 결과를 리턴하는지 여부(DETERMINISTIC) 또는 메소드가 결과에 영향을 주는 몇몇 상태 값에 따라 달라지는지 여부(NOT DETERMINISTIC)를 지정합니다. 즉, DETERMINISTIC 메소드는 항상 동일한 입력으로 계속되는 호출로부터 같은 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC을 지정하여 방지할 수 있습니다. 결정적이지 않은 유형의 예는 결과 유형에 영향을 주는 특수 레지스터, 전역 변수 또는 결정적이지 않은 함수입니다.

FENCED 또는 NOT FENCED
이 절은 메소드가 데이터베이스 관리 프로그램 운영 환경의 프로세스 또는 어드레스 스페이스에서 실행되어도 안전한지(FENCED) 또는 안전하지 않은지(NOT FENCED)를 지정합니다.

메소드가 FENCED로 등록된 상태라면, 데이터베이스 관리 프로그램은 그 내부 자원(예: 데이터 버퍼)을 메소드에 의한 액세스로부터 보호합니다. 대부분의 메소드는 FENCED 또는 NOT FENCED로 실행됩니다. 일반적으로 NOT FENCED로 실행되는 메소드는 FENCED로 실행되는 메소드 만큼 잘 수행되지 않습니다.

주의:
제대로 검사되지 않은 메소드의 NOT FENCED 사용은 DB2 데이터베이스의 무결성을 손상시킬 수 있습니다. DB2 데이터베이스는 발생할 수 있는 많은 공통 유형의 의도하지 않은 실패에 대해 예방책을 사용하지만, NOT FENCED 사용자 정의 메소드가 사용될 경우에는 완전한 무결성을 보장할 수 없습니다.

LANGUAGE OLE 또는 NOT THREADSAFE의 메소드에 대해서는 FENCED만 지정할 수 있습니다(SQLSTATE 42613).

메소드가 FENCED이고 NO SQL 옵션을 가질 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

메소드를 NOT FENCED로 등록하기 위해서는 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED_ROUTINE)이 필요합니다.

THREADSAFE 또는 NOT THREADSAFE

메소드가 다른 루틴과 같은 프로세스에서 실행되어도 안전한지(THREADSAFE) 또는 안전하지 않은지(NOT THREADSAFE) 여부를 지정합니다.

메소드가 OLE가 아닌 다른 LANGUAGE로 정의된 경우에는 다음과 같습니다.

- 메소드가 THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램이 다른 루틴과 같은 프로세스에서 메소드를 호출할 수 있습니다. 일반적으로 Threadsafe 함수가 되려면 메소드가 전역 데이터 영역이나 정적 데이터 영역을 사용하지 않아야 합니다. 대부분의 프로그래밍 참조서에서는 스레드 안전 루틴을 작성하는 것에 대해 설명합니다. FENCED와 NOT FENCED 메소드는 둘 다 THREADSAFE될 수 있습니다.
- 메소드가 NOT THREADSAFE로 정의되어 있으면 데이터베이스 관리 프로그램은 다른 루틴과 같은 프로세스에서 메소드를 호출하지 않습니다.

FENCED 메소드에서 LANGUAGE가 JAVA인 경우에 THREADSAFE가 디폴트값입니다. 다른 모든 언어에서는 NOT THREADSAFE가 디폴트값입니다. 이 메소드가 LANGUAGE OLE로 정의된 경우, THREADSAFE를 지정할 수 없습니다(SQLSTATE 42613).

NOT FENCED 메소드에서는 THREADSAFE가 디폴트값입니다. NOT THREADSAFE는 지정될 수 없습니다(SQLSTATE 42613).

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 주제가 아닌 인수가 널(NULL)인 경우 외부 메소드 호출을 피하는 데 사용할 수 있습니다.

RETURNS NULL ON NULL INPUT이 지정된 경우와 실행시 메소드 인수 중 하나가 널(NULL)인 경우, 메소드는 호출되지 않고 결과는 널(NULL) 값입니다.

CALLED ON NULL INPUT이 지정되면, 널(NULL) 인수의 수에 관계없이 메소드가 호출됩니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나 널(NULL) 인수 값의 테스트는 메소드에서 수행합니다.

이전 버전과의 호환성 및 제품군 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로 NULL CALL을 사용할 수도 있습니다. 마찬가지로, NOT NULL CALL을 RETURNS NULL ON NULL INPUT에 대한 동의어로 사용할 수도 있습니다.

이 스펙이 무시되는 두 가지 경우가 있습니다.

- 주제 인수가 널(NULL)인 경우, 이 경우 메소드가 실행되지 않고 결과는 널(NULL)입니다.
- 메소드가 매개변수를 하나도 가지지 않도록 정의된 경우, 이 경우에는 이 널(NULL) 인수 조건이 발생할 수 없습니다.

NO SQL, CONTAINS SQL, READS SQL DATA

메소드가 SQL문을 발행할지 여부 및 발행할 경우 SQL문의 유형을 나타냅니다.

NO SQL

메소드가 SQL문을 실행할 수 없음을 나타냅니다(SQLSTATE 38001).

CONTAINS SQL

SQL 데이터를 읽지도 수정하지도 않는 SQL문이 메소드에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 38004 또는 42985). 메소드에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문이 이 메소드에 포함될 수 있음을 나타냅니다(SQLSTATE 38002 또는 42985). 메소드에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

이 선택적 절은 메소드가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지 여부를 지정합니다. 외부 영향이 없는 메소드를 가정하는 최적화는 EXTERNAL ACTION을 지정하여 방지합니다.

NO SCRATCHPAD 또는 SCRATCHPAD *length*

이 선택적 절은 외부 메소드에 대해 스크래치 패드가 제공되는지 여부를 지정하는 데 사용할 수 있습니다. 메소드를 다시 입력하도록 강력하게 권장되므로, 스크래치 패드는 하나의 호출에서 다음 호출로 "상태를 저장"하는 메소드 수단을 제공합니다.

SCRATCHPAD를 지정하면, 사용자 정의 메소드를 처음 호출할 때 외부 메소드가 사용할 스크래치 패드에 대해 메모리가 할당됩니다. 이 스크래치 패드는 다음 등록 정보를 갖습니다.

- *length*는 지정할 경우 스크래치 패드의 크기를 바이트 단위로 설정하며 이 크기는 1에서 32,767 사이여야 합니다(SQLSTATE 42820). 디폴트값은 100입니다.
- 모두 X'00'으로 초기화됩니다.
- 범위는 SQL문입니다. SQL문의 외부 메소드에 대한 참조당 하나의 스크래치 패드가 존재합니다.

따라서 다음 명령문의 메소드 X가 SCRATCHPAD 키워드로 지정되면 세 개의 스크래치 패드가 지정됩니다.

```
SELECT A, X..(A) FROM TABLEB
   WHERE X..(A) > 103 OR X..(A) < 19
```

ALLOW PARALLEL이 지정되거나 디폴트 설정되면, 범위가 위와 달라집니다. 메소드가 다중 데이터베이스 파티션에서 실행되는 경우, SQL문의 메소드의 각 참조에 대해 해당 메소드가 처리되는 각 데이터베이스 파티션에 하나의 스크래치 패드가 할당됩니다. 마찬가지로 파티션 내 병렬 처리를 사용하여 쿼리가 실행되면 네 개 이상의 스크래치 패드가 지정될 수 있습니다.

스크래치 패드는 지속적입니다. 내용이 하나의 외부 메소드 호출에서 다음 외부 메소드 호출로 보존됩니다. 하나의 호출에서 외부 메소드에 의한 스크래치 패드 변경은 다음 호출에서도 보존됩니다. 데이터베이스 관리 프로그램은 각 SQL문 실행이 시작될 때 스크래치 패드를 초기화하며, 각 서브쿼리 실행이 시작될 때 스크래치 패드를 재설정합니다. FINAL CALL 옵션을 지정하면, 시스템에서 스크래치 패드를 재설정하기 전에 마지막 호출을 발행합니다.

스크래치 패드는 외부 메소드가 획득할 수 있는 시스템 자원(예: 메모리)의 중앙 지점으로 사용될 수 있습니다. 메소드는 첫 번째 호출시 메모리를 획득하고 그 주소를 스크래치 패드에 보존하며 후속 호출시 이를 참조할 수 있습니다.

시스템 자원을 획득한 경우 FINAL CALL 키워드도 지정해야 합니다. 그러면 외부 메소드가 획득한 모든 시스템 자원을 해제할 수 있도록 명령문의 끝에서 특수 호출이 이뤄집니다.

SCRATCHPAD를 지정하면, 사용자 정의 메소드를 호출할 때마다 추가 인수가 스크래치 패드를 지정하는 외부 메소드에 전달됩니다.

NO SCRATCHPAD를 지정하면, 스크래치 패드가 외부 메소드에 할당되거나 전달되지 않습니다.

NO FINAL CALL 또는 FINAL CALL

이 선택적 절은 외부 메소드에 대한 마지막 호출이 이루어질지 여부를 지정합니다. 마지막 호출의 목적은 외부 메소드가 획득한 시스템 자원을 해제할 수 있게 하는 것입니다. 이는 외부 메소드가 메모리와 같은 시스템 자원을 획득하고 이를 스크래치 패드에 저장하는 경우에 SCRATCHPAD 키워드와 함께 사용하면 유용할 수 있습니다.

FINAL CALL을 지정하면, 실행시 추가 인수가 호출 유형을 지정하는 외부 메소드에 전달됩니다. 호출 유형은 다음과 같습니다.

- 정상 호출: SQL 인수가 전달되고 결과 리턴이 예상됩니다.
- 첫 번째 호출: 이러한 특정 SQL문에서 메소드의 특정 참조에 대한 외부 메소드의 첫 번째 호출. 첫 번째 호출은 정상적인 호출입니다.
- 마지막 호출: 메소드가 자원에 여유 공간을 확보할 수 있도록 해주는 외부 메소드에 대한 마지막 호출. 마지막 호출은 정상적인 호출이 아닙니다. 이 마지막 호출은 다음 상황에서 발생합니다.
  - 명령문의 끝: 이러한 경우는 커서 지향 명령문에 대한 커서가 닫히거나 명령문이 실행될 때 발생합니다.
  - 트랜잭션의 끝: 이 경우는 정상적인 명령문 끝이 발생하지 않을 때 발생됩니다. 예를 들어, 응용프로그램의 논리는 어떤 이유로 커서 닫기를 생략할 수 있습니다.

  WITH HOLD로 정의된 커서가 열려 있는 동안 커미트 조작이 발생하면, 그 다음의 커서를 닫을 때 또는 응용프로그램 종료시 마지막 호출이 수행됩니다.

NO FINAL CALL을 지정하면, "호출 유형" 인수가 외부 메소드에 전달되지 않고 마지막 호출도 이루어지지 않습니다.

ALLOW PARALLEL 또는 DISALLOW PARALLEL

이 선택적 절은 메소드에 대한 단일 참조에 대해 메소드 호출을 병렬화할 수 있는지 여부를 지정합니다. 일반적으로 대부분의 메소드 호출은 병렬화가 가능해야 하나, 병렬화할 수 없는 메소드(스크래치 패드의 단일 사본에 종속적인 메소드)가 있을 수 있습니다. 메소드에 대해 ALLOW PARALLEL 또는 DISALLOW PARALLEL을 지정할 경우 DB2는 이 스펙을 승인합니다.

메소드에 적절한 키워드를 판별할 때는 다음 질문을 고려해야 합니다.

- 모든 메소드 호출이 서로 완전히 독립적인가? 그럴 경우, ALLOW PARALLEL을 지정하십시오.
- 각 메소드 호출시 다음 호출과 관련된 값을 제공하여 스크래치 패드를 갱신하는가(예: 카운터 증가)? 외부 조치가 있을 경우 DISALLOW PARALLEL을 지정하거나 디폴트값을 사용하십시오.
- 하나의 데이터베이스 파티션에 대해서만 발생해야 하는 메소드로 수행되는 외부 조치가 있는가? 외부 조치가 있을 경우 DISALLOW PARALLEL을 지정하거나 디폴트값을 사용하십시오.
- 비용이 많이 드는 초기화 프로세스를 최소한의 횟수로 수행할 수 있도록 스크래치 패드가 사용되었는지 여부. 그럴 경우, ALLOW PARALLEL을 지정하십시오.

모든 경우에 모든 외부 메소드 본문이 모든 데이터베이스 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

구문 도표는 디폴트값이 ALLOW PARALLEL임을 나타냅니다. 단, 다음 옵션 중 하나 이상이 명령문에 지정되어 있을 경우, 디폴트값은 DISALLOW PARALLEL입니다.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO 또는 DBINFO

이 선택적 절은 DB2에서 제공하는 고유한 특정 정보가 추가 호출시간 인수로 메소드에 전달되는 지(DBINFO) 또는 전달되지 않는 지(NO DBINFO)를 지정합니다. NO DBINFO가 디폴트값입니다. DBINFO는 LANGUAGE OLE에 대해 지원하지 않습니다(SQLSTATE 42613). 정의할 메소드가 다른 메소드를 겹쳐쓸 경우에는 이 절을 지정할 수 없습니다(SQLSTATE 428FV).

DBINFO를 지정하면 다음 정보가 포함된 구조가 메소드에 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름입니다.
- 응용프로그램 ID - 각 데이터베이스 연결에 대해 설정되는 고유한 응용프로그램 ID입니다.
- 응용프로그램 권한 부여 ID - 이 메소드와 응용프로그램간의 중첩된 메소드에 관계없는 응용프로그램 권한 부여 ID입니다.
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 스키마 이름 - 테이블 이름과 완전히 같은 조건하에서 스키마의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 테이블 이름 - 메소드 참조가 UPDATE문의 SET절 오른쪽이거나 INSERT문의 VALUES 목록에 있는 한 항목인 경우, 갱신 또는 삽입 중인 테이블의 규정되지 않은 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 컬럼 이름 - 테이블 이름과 완전히 같은 조건하에서, 갱신하거나 삽입할 컬럼의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 데이터베이스 버전/릴리스 - 메소드를 호출하는 데이터베이스 서버의 버전, 릴리스 및 수정 레벨을 식별합니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.
- 테이블 메소드 결과 컬럼 수 - 메소드에는 해당되지 않습니다.

INHERIT SPECIAL REGISTERS

이 선택적 절은 메소드에 있는 특수 레지스터가 레지스터의 초기값을 호출 명령문에서 상속하도록 지정합니다. 커서의 경우에는 초기값이 커서를 여는 시점에서 상속됩니다.

특수 레지스터에 대한 변경사항은 메소드 호출자에게 다시 전달되지 않습니다.

날짜 시간 특수 레지스터와 같은 특수 레지스터는 현재 실행 중인 명령문의 등록 정보를 반영하므로 호출자로부터 상속하지 않습니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 구조화된 유형을 작성할 경우 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 갖는다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 속성없이 정의된 구조화된 부속 유형은 슈퍼 유형으로부터 모든 속성을 상속하는 부속 유형을 정의합니다. UNDER절이나 다른 속성이 지정되지 않았다면 이 유형은 속성없는 자료형 계층의 루트 유형입니다.
- 자료형 계층에 새 부속 유형을 추가하면 패키지가 무효화될 수 있습니다. 패키지가 새 유형의 슈퍼 유형에 종속적인 경우 이 패키지를 무효화할 수 있습니다. 그러한 종속성은 TYPE 술어 또는 TREAT 스펙의 사용 결과입니다.
- 구조화된 유형은 최대 4082개의 속성을 가질 수 있습니다(SQLSTATE 54050).
- 함수와 같은 시그니처를 가지는 메소드 스펙은 허용되지 않습니다(메소드의 주제 유형과 함수의 첫 번째 매개변수 유형 비교).
- 원래 메소드는 다른 메소드를 겹쳐쓸 수 없으며, 원래 메소드가 겹쳐쓰여질 수도 없습니다(SQLSTATE 42745). 더우기 함수와 메소드가 겹쳐쓰기 관계에 있을 수 없습니다. 이는 함수에서 첫 번째 매개변수를 포함하는 메소드를 주제 S로 간주할 경우, 이 함수가 S의 슈퍼 유형에 있는 다른 메소드를 겹쳐써서는 안되고, S의 부속 유형에 있는 다른 메소드에 의해 겹쳐쓰여져서도 안 됨을 의미합니다.
- 구조화된 유형을 작성하면 해당 유형과 함께 사용될 함수 및 메소드 세트가 자동으로 생성됩니다. 모든 함수와 메소드는 구조화된 유형과 같은 스키마에 생성됩니다. 생성된 함수 또는 메소드의 시그니처가 이 스키마에 있는 기존 함수의 시그니처와 상충되거나 이 시그니처를 대체하는 경우, 명령문은 실패합니다(SQLSTATE 42710). 생성된 함수 또는 메소드는 구조화된 유형을 삭제하지 않고는 삭제할 수 없습니다(SQLSTATE 42917). 다음과 같은 함수와 메소드가 생성됩니다.
  - 함수
    - 참조 비교

      이름이 =, <>, <, <=, >, >=인 여섯 가지 비교 함수가 참조 유형 REF(*type-name*)에 대해 생성됩니다. 이들 각 함수는 유형 REF(*type-name*)의

두 매개변수를 취하고, 참, 거짓 또는 알 수 없음을 리턴합니다. REF(*type-name*)에 대한 비교 연산자는 REF(*type-name*)의 하위 데이터 유형에 대한 비교 연산자와 같은 동작을 하도록 정의되어 있습니다. (자료형 계층의 모든 참조는 동일한 참조 표시 유형을 가집니다.) 이것은 S와 T가 공동의 슈퍼 유형을 가지면, REF(S)와 REF(T)가 비교 가능하도록 합니다. OID 컬럼의 고유성은 하나의 테이블 계층 내에서만 강요되므로, 한 테이블 계층의 REF(T) 값이 다른 테이블 계층의 REF(T) 값과 "동일"할 수 있으며, 이는 다른 행을 참조하더라도 마찬가지입니다.

비교 시 참조 유형의 범위는 고려되지 않습니다.

- 캐스트 함수

생성된 참조 유형 REF(*type-name*)와 이 참조 유형의 하위 데이터 유형 사이의 캐스트를 위해 두 개의 캐스트 함수가 생성됩니다.

- 하위 유형에서 참조 유형으로의 캐스트를 위한 함수 이름은 내재된 또는 명시적 *funcname1*입니다.

  이 함수의 형식은 다음과 같습니다.

  ```
  CREATE FUNCTION funcname1 (rep-type)
    RETURNS REF(type-name) ...
  ```

- 참조 유형에서 하위 유형으로의 캐스트를 위한 함수 이름은 내재된 또는 명시적 *funcname2*입니다.

  이 함수의 형식은 다음과 같습니다.

  ```
  CREATE FUNCTION funcname2 ( REF(type-name) )
    RETURNS rep-type ...
  ```

일부 rep-type의 경우, 상수로부터의 캐스팅을 처리하기 위해 *funcname1*로 생성된 캐스트 함수가 추가됩니다.

- *rep-type*이 SMALLINT인 경우, 추가로 생성된 캐스트 함수는 다음 형식을 가집니다.

  ```
  CREATE FUNCTION funcname1 (INTEGER)
    RETURNS REF(type-name)
  ```

- *rep-type*이 CHAR(n)인 경우, 추가로 생성된 캐스트 함수는 다음 형식을 가집니다.

  ```
  CREATE FUNCTION funcname1 ( VARCHAR(n))
    RETURNS REF(type-name)
  ```

- *rep-type*이 GRAPHIC(n)인 경우, 추가로 생성된 캐스트 함수는 다음 형식을 가집니다.

  ```
  CREATE FUNCTION funcname1 (VARGRAPHIC(n))
    RETURNS REF(type-name)
  ```

구조화된 유형의 스키마 이름은 SQL문에서 이러한 연산자와 캐스트 함수를 성공적으로 사용하기 위해 SQL 경로에 포함되어야 합니다.

- 컨스트럭터 함수

  컨스트럭터 함수는 구성될 유형의 새로운 인스턴스를 허용하기 위해 생성됩니다. 이러한 새 인스턴스는 슈퍼 유형으로부터 상속된 속성을 포함하여 해당 유형의 모든 속성에 대해 널(NULL)을 가지게 됩니다.

  다음은 생성된 컨스트럭터 함수의 형식입니다.

```
CREATE FUNCTION type-name ( )
  RETURNS type-name
 ...
```

  NOT INSTANTIABLE이 지정된 경우에는 컨스트럭터 함수가 생성되지 않습니다.

– 메소드

  - Observer 메소드

    구조화된 유형의 속성 각각에 대한 Observer 메소드가 정의됩니다. Observer 메소드는 각 속성에 대해 해당 속성의 유형을 리턴합니다. 주제가 널(NULL)인 경우, Observer 메소드는 해당 속성 유형의 널(NULL) 값을 리턴합니다.

    예를 들어, 구조화된 유형 ADDRESS의 인스턴스 속성은 `C1..STREET`, `C1..CITY`, `C1..COUNTRY` 및 `C1..CODE`를 사용하여 관찰할 수 있습니다.

    생성된 Observer 메소드의 메소드 시그니처는 다음 명령문이 실행된 경우와 같습니다.

```
CREATE TYPE  type-name
    ...
  METHOD attribute-name()
    RETURNS attribute-type
```

    여기서, *type-name*은 구조화된 유형의 이름입니다.

  - 변화(Mutator) 메소드

    구조화된 유형의 속성 각각에 대한 유형 보존 Mutator 메소드가 정의됩니다. Mutator 메소드를 사용하여 구조화된 유형의 인스턴스 내 속성을 변경하십시오. Mutator 메소드는 각 속성에 대해 사본의 이름 지정된 속성에 인수를 지정함으로써 수정된 주제의 사본을 리턴합니다.

    예를 들어, 구조화된 유형 ADDRESS의 인스턴스는 `C1..CODE('M3C1H7')`를 사용하여 변화시킬 수 있습니다. 주제가 널(NULL)인 경우, Mutator 메소드에 오류가 발생합니다(SQLSTATE 2202D).

생성된 Mutator 메소드의 메소드 시그니처는 다음 명령문이 실행된 경우와 같습니다.

```
CREATE TYPE  type-name
       ...
    METHOD attribute-name (attribute-type)
       RETURNS type-name
```

속성 데이터 유형이 SMALLINT, REAL, CHAR 또는 GRAPHIC인 경우, 상수를 사용한 변화를 지원하기 위해 추가 Mutator 메소드가 생성됩니다.

- *attribute-type*이 SMALLINT인 경우, 추가 Mutator 메소드는 INTEGER 유형의 인수를 지원합니다.
- *attribute-type*이 REAL인 경우, 추가 Mutator 메소드는 DOUBLE 유형의 인수를 지원합니다.
- *attribute-type*이 CHAR인 경우, 추가 Mutator 메소드는 VARCHAR 유형의 인수를 지원합니다.
- *attribute-type*이 GRAPHIC인 경우, 추가 Mutator 메소드는 VARGRAPHIC 유형의 인수를 지원합니다.

- 구조화된 유형이 컬럼 유형으로 사용되는 경우, 해당 유형의 인스턴스 길이는 런타임시 1GB를 넘을 수 없습니다(SQLSTATE 54049).

- 기존의 구조화된 유형에 대한 새 부속 유형을 작성하는 경우(컬럼 유형으로 사용하기 위해), 기존의 관련 구조화된 유형의 지원으로 이미 작성된 변환 함수를 필요한 경우 다시 검사하고 갱신해야 합니다. 새 유형이 주어진 유형과 같은 계층에 있는지 또는 중첩된 유형의 계층에 있는지에 따라, 이 유형과 연관된 기존의 변환 함수가 새 부속 유형에 의해 도입되는 새 속성 중 일부 또는 모두를 포함하도록 수정해야 합니다. 일반적으로, UDF 및 클라이언트 응용프로그램이 구조화된 유형에 액세스할 수 있게 해주는 주어진 유형(또는 자료형 계층)과 연관된 변환 함수 세트이므로, 주어진 복합 계층에서 모든 속성을 지원하는 변환 함수를 작성해야 합니다(즉, 모든 부속 유형 및 중첩된 구조화된 유형의 임시 닫기 포함).

  기존 유형의 새 부속 유형이 작성될 때, 작성되는 유형의 슈퍼 유형으로 정의된 메소드에 종속된 패키지 및 겹쳐쓸 수 있는 메소드에 종속된 모든 패키지는 무효화됩니다.

- ***테이블 액세스 제한사항***: 메소드가 READS SQL DATA로 정의되어 있으면 메소드의 명령문은 메소드를 호출한 명령문이 수정하고 있는 테이블에 액세스할 수 없습니다(SQLSTATE 57053). 예를 들어, BONUS() 메소드가 READS SQL DATA로 정의되어 있다고 가정해 보십시오. UPDATE DEPTINFO SET SALARY = SALARY + EMP..BONUS() 명령문을 호출하면 BONUS 메소드에 있는 어떤 SQL 문도 EMPLOYEE 테이블에서 데이터를 읽을 수 없습니다.
- ***특권***: 사용자 정의 유형의 정의자는 구조화된 유형에 대하여 자동으로 생성된 모든 메소드와 함수에 대해 EXECUTE 특권(WITH GRANT OPTION)을 수신합니다.

CREATE METHOD문을 사용하여 메소드 본문을 정의한 후에만 CREATE TYPE 문에 명시적으로 지정된 메소드에 대해 EXECUTE 특권을 부여할 수 있습니다. 사용자 정의 유형의 정의자는 ALTER TYPE문을 사용하여 메소드 스펙을 삭제할 수 있습니다. CREATE TYPE(구조화)문을 실행하는 동안 자동으로 생성된 모든 메소드 및 함수에 대한 EXECUTE 특권은 PUBLIC에 부여됩니다.

SQL문에 외부 메소드를 사용할 때는 메소드 정의자에게 메소드가 사용하는 모든 패키지에 대한 EXECUTE 특권이 있어야 합니다.

- 파티션된 데이터베이스 환경에서는 외부 사용자 정의 함수 또는 메소드에서 SQL을 사용할 수 없습니다(SQLSTATE 42997).
- NO SQL로 정의된 루틴만 인덱스 확장자를 정의하는 데 사용할 수 있습니다 (SQLSTATE 428F8).
- NOT FENCED로 정의된 Java 루틴은 FENCED THREADSAFE로 정의된 것처럼 호출됩니다.
- *호환성*: z/OS용 DB2와의 호환성을 위해 다음이 지원됩니다.
  - 다음 구문은 허용됩니다.
    - DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있습니다.
    - NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.
    - CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.
    - RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

  다음 구문이 외부 메소드에 대한 디폴트 동작으로 허용됩니다.
  - ASUTIME NO LIMIT
  - NO COLLID
  - PROGRAM TYPE SUB
  - STAY RESIDENT NO
  - 유니코드 데이터베이스의 경우 CCSID UNICODE
  - CCSID ASCII(PARAMETER CCSID UNICODE가 지정되지 않은 경우 비유니코드 데이터베이스)

  다음 구문이 SQL 메소드에 대한 디폴트 동작으로 허용됩니다.
  - 유니코드 데이터베이스의 경우 CCSID UNICODE
  - 비유니코드 데이터베이스의 경우 CCSID ASCII

  이전 버전의 DB2 데이터베이스와의 호환성을 위해,
  - PARAMETER STYLE SQL 대신 PARAMETER STYLE DB2SQL을 지정할 수 있습니다.

예:

예 1: 부서에 대한 유형을 작성하십시오.

```
CREATE TYPE DEPT AS
         (DEPT NAME     VARCHAR(20),
       MAX_EMPS INT)
                REF USING INT
    MODE DB2SQL
```

예 2: 관리 프로그램에 대한 부속 유형과 사원에 대한 유형으로 구성된 자료형 계층을 작성하십시오.

```
CREATE TYPE EMP AS
       (NAME      VARCHAR(32),
       SERIALNUM INT,
       DEPT      REF(DEPT),
       SALARY    DECIMAL(10,2))
  MODE DB2SQL

CREATE TYPE MGR UNDER EMP AS
       (BONUS     DECIMAL(10,2))
  MODE DB2SQL
```

예 3: 주소에 대한 자료형 계층을 작성하십시오. 주소는 컬럼의 유형으로 사용됩니다. 인라인 길이가 지정되지 않았으므로 DB2가 디폴트 길이를 계산합니다. 주소 유형 정의 내에서 이 주소가 주어진 입력 주소가 되도록 하는 방법을 계산하는 외부 메소드를 캡슐화합니다. CREATE METHOD문을 사용하여 메소드 본문을 작성하십시오.

```
CREATE TYPE address_t AS
       (STREET     VARCHAR(30),
       NUMBER     CHAR(15),
       CITY       VARCHAR(30),
       STATE      VARCHAR(10))
  NOT FINAL
  MODE DB2SQL
    METHOD SAMEZIP (addr address_t)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    NO EXTERNAL ACTION,

    METHOD DISTANCE (address_t)
    RETURNS FLOAT
    LANGUAGE C
    DETERMINISTIC
         PARAMETER STYLE SQL
          NO SQL
           NO EXTERNAL ACTION

CREATE TYPE germany_addr_t UNDER address_t AS
       (FAMILY_NAME VARCHAR(30))
  NOT FINAL
       MODE DB2SQL
```

```
CREATE TYPE us_addr_t UNDER address_t AS
       (ZIP VARCHAR(10))
  NOT FINAL
  MODE DB2SQL
```

예 4: 중첩된 구조화된 유형 속성을 갖는 유형을 작성하십시오.

```
CREATE TYPE PROJECT AS
  (PROJ_NAME  VARCHAR(20),
   PROJ_ID    INTEGER,
   PROJ_MGR   MGR,
   PROJ_LEAD  EMP,
   LOCATION   ADDR_T,
   AVAIL_DATE DATE)
   MODE DB2SQL
```

CREATE TYPE MAPPING

CREATE TYPE MAPPING문은 다음의 데이터 유형 간의 맵핑을 정의합니다.

- 페더레이티드 데이터베이스에 정의될 데이터 소스 테이블이나 뷰에 있는 컬럼의 데이터 유형
- 페더레이티드 데이터베이스에 이미 정의되어 있는 해당 데이터 유형

맵핑은 다음에서 페더레이티드 데이터베이스 데이터 유형과 데이터 유형을 연관지을 수 있습니다.

- 지정된 데이터 소스
- 데이터 소스의 범위(예를 들어, 특정 유형 및 버전의 모든 데이터 소스)

기존의 데이터 유형 맵핑이 적합하지 않은 경우에만 이를 작성해야 합니다.

별칭을 작성하거나 테이블을 작성할 때(투명한 DDL) 다중 유형 맵핑이 적용 가능할 경우, 가장 최근 맵핑이 적용됩니다.

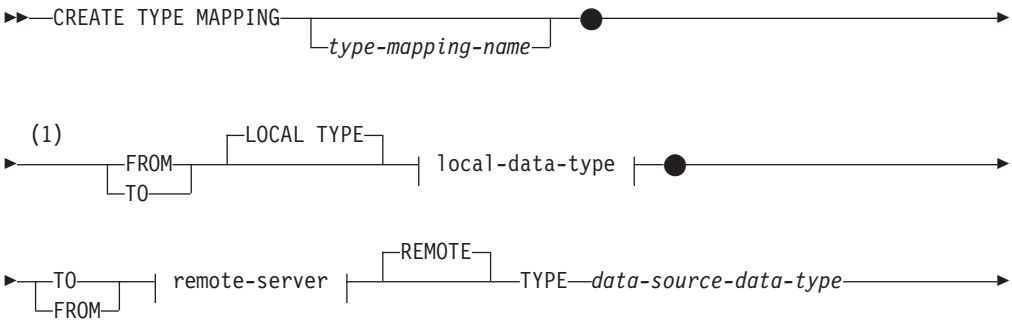
호출

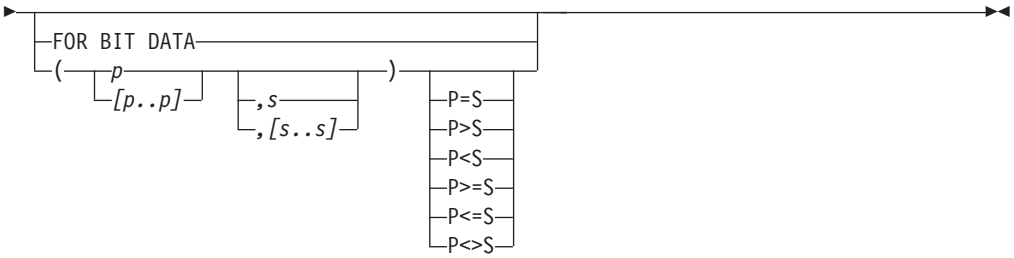
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

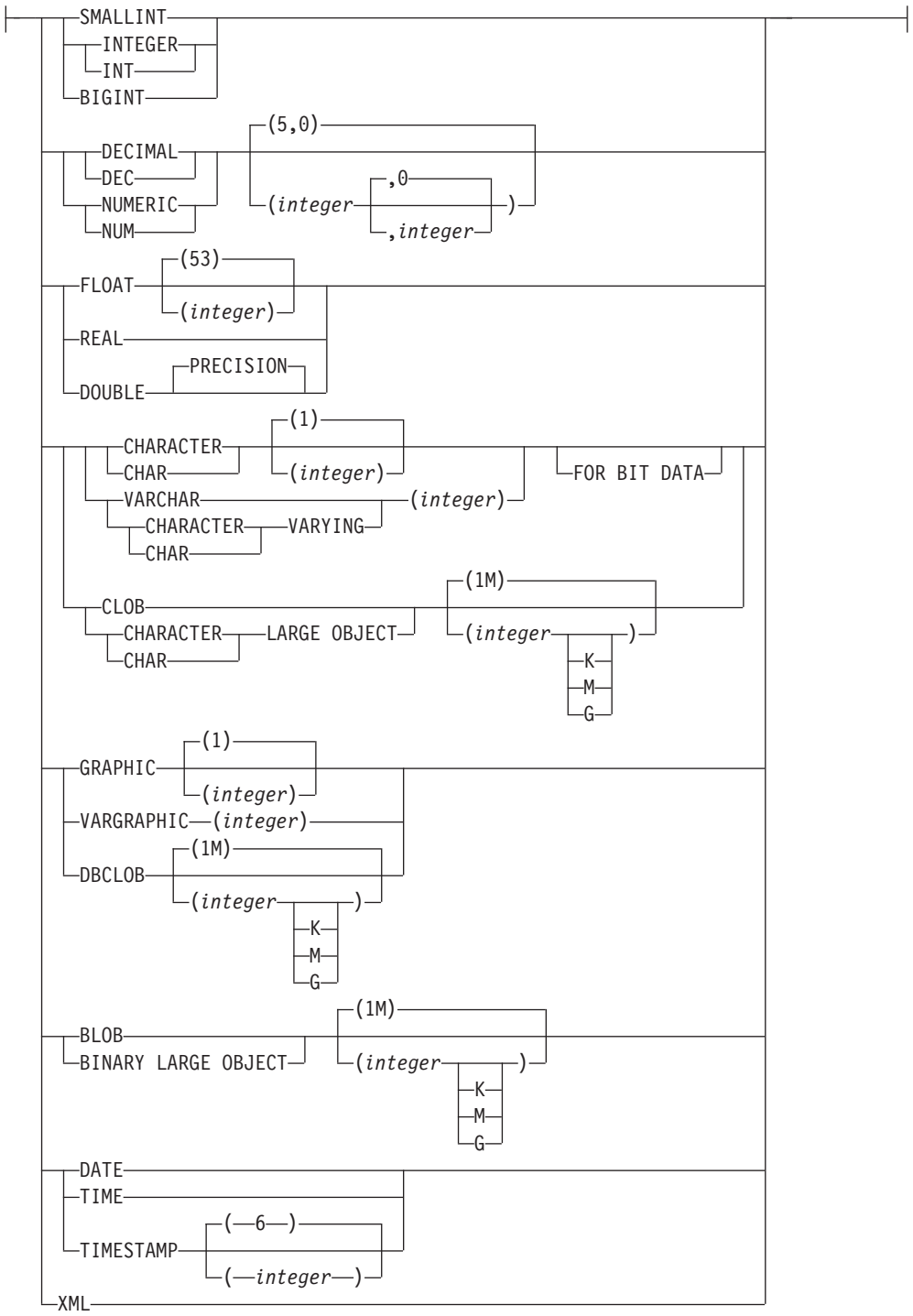
구문





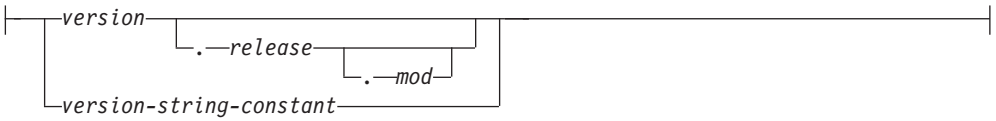
local-data-type:

내장 유형:



remote-server:

server-version:



주:

1 TO 키워드와 FROM 키워드 모두 CREATE TYPE MAPPING문에 있어야 합니다.

설명

type-mapping-name

데이터 유형 맵핑의 이름을 지정합니다. 이 이름은 카탈로그에 이미 기술되어 있는 데이터 유형 맵핑을 식별해서는 안됩니다. *type-mapping-name*을 지정하지 않으면 고유한 이름이 생성됩니다.

FROM 또는 TO

리버스 또는 포워드 유형 맵핑을 지정합니다.

FROM

뒤에 *local-data-type*이 오면 포워드 유형 맵핑을 지정하고 *remote-server*가 오면 리버스 유형 맵핑을 지정합니다.

TO

뒤에 *remote-server*가 오면 포워드 유형 맵핑을 지정하고 *local-data-type*이 오면 리버스 유형 맵핑을 지정합니다.

local-data-type

페더레이티드 데이터베이스에 정의되는 데이터 유형을 식별합니다. 스키마 이름없이 *local-data-type*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형 이름이 해석됩니다.

매개변수화된 데이터 유형에 대해 빈 괄호를 사용할 수 있습니다. 매개변수화된 데이터 유형은 특정 길이, 정밀도 또는 스케일을 사용하여 정의할 수 있는 데이터 유형 중 하나입니다. 포워드 유형 맵핑에 빈 괄호가 지정될 경우(예:CHAR()), 길이는 리모트 테이블의 컬럼 길이에 의해 판별됩니다. 리버스 유형 맵핑에 빈 괄호가 지정될 경우, 유형 맵핑은 모든 길이의 데이터 유형에 적용됩니다. 괄호 전체를 생략할 경우, 데이터 유형의 디폴트 길이가 사용됩니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601). 매개변수 값이 서로 다른 데이터 유형(DECFLOAT 또는 DECIMAL)을 나타내기 때문에 NUMBER()는 사용할 수 없습니다(SQLSTATE 42601).

DECFLOAT는 Oracle 랩퍼, Linux, UNIX, 및 Windows용 IBM DB2 버전 9.5 이상의 DB2 랩퍼를 사용하여 *local-data-type*으로만 허용 가능합니다.

*local-data-type*은 사용자 정의 유형일 수 없습니다(SQLSTATE 42611).

SERVER *server-name*

*data-source-data-type*이 정의되는 데이터 소스의 이름을 지정합니다.

SERVER TYPE *server-type*

*data-source-data-type*이 정의되는 데이터 소스의 유형을 식별합니다.

VERSION

*data-source-data-type*이 정의되는 데이터 소스의 버전을 식별합니다.

version

버전 번호를 지정합니다. 이 값은 정수여야 합니다.

release

*version*으로 표시된 버전의 릴리스 번호를 지정합니다. 이 값은 정수여야 합니다.

mod

*release*로 표시된 릴리스의 수정 번호를 지정합니다. 이 값은 정수여야 합니다.

version-string-constant

버전의 전체 명칭을 지정합니다. *version-string-constant*는 단일 값(예: '8i') 이거나 *version*, *release* 및 *mod*(적용 가능한 경우)의 조인된 값(예: '8.0.3') 일 수 있습니다.

WRAPPER *wrapper-name*

server-type 및 *server-version*에 의해 표시된 유형 및 버전의 데이터 소스와 상호 작용하기 위해 페더레이티드 서버가 사용하는 랩퍼의 이름을 지정합니다.

TYPE *data-source-data-type*

로컬 데이터 유형에서 맵핑되고 있는 데이터 소스 데이터 유형을 지정합니다.

매개변수화된 데이터 유형에 대해 빈 괄호를 사용할 수 있습니다. 포워드 유형 맵핑에 빈 괄호가 지정될 경우(예:CHAR()), 유형 맵핑은 모든 길이의 데이터 유형에 적용됩니다. 리버스 유형 맵핑에 빈 괄호가 지정될 경우, 길이는 투명한 DDL에 지정된 컬럼 길이에 의해 판별됩니다. 괄호 전체를 생략할 경우, 데이터 유형의 디폴트 길이가 사용됩니다.

*data-source-data-type*은 내장 데이터 유형이어야 합니다. 사용자 정의 유형은 허용되지 않습니다.

*server-name*을 유형 맵핑으로 지정하거나 기존 서버가 유형 맵핑의 영향을 받을 경우, 유형 맵핑을 작성할 때 *data-source-data-type*, *p* 및 *s*가 검증됩니다(SQLSTATE 42611).

p *p*를 지정한 경우, 길이 또는 정밀도가 *p*와 동일한 데이터 유형만 유형 맵핑의 영향을 받습니다.

[p1..p2]
포워드 유형 맵핑 전용. 10진수 데이터 유형의 경우, *p1* 및 *p2*는 값이 가질 수 있는 최대 자릿수를 지정합니다. 문자열 데이터 유형의 경우, *p1* 및 *p2*는 값이 가질 수 있는 최소 및 최대 문자 수를 지정합니다. 모든 경우에 있어, 최대값은 최소값과 같거나 커야 합니다. 그리고 두 숫자 모두 데이터 유형에 대해 유효해야 합니다.

s *s*를 지정한 경우, 스케일이 *s*와 동일한 데이터 유형만 유형 맵핑의 영향을 받습니다.

[s1..s2]
포워드 유형 맵핑 전용. 10진수 데이터 유형의 경우, *s1* 및 *s2*는 소수점 오른쪽으로 허용 가능한 최소 및 최대 자릿수를 지정합니다. 최대값은 최소값과 같거나 커야 합니다. 그리고 두 숫자 모두 데이터 유형에 대해 유효해야 합니다.

P [operand] S
10진수 데이터 유형의 경우, P *[operand]* S는 정밀도와 소수점 오른쪽으로 허용 가능한 자릿수와의 비교를 지정합니다. 예를 들어, 피연산자 =는 정밀도와 10진수 소수에 허용되는 자릿수가 동일하면 유형 맵핑이 적용된 것을 표시합니다.

FOR BIT DATA
*data-source-data-type*이 비트 데이터에 대한 것인지 여부를 나타냅니다. 데이터 소스 유형 컬럼이 2진 값을 포함하는 경우에 이들 키워드가 필요합니다. 문자 데이터 유형에 대해 지정되어 있지 않으면, 데이터베이스 관리 프로그램이 이 속성을 결정합니다.

주

- 다음 조건에서는 주어진 작업 단위(UOW) 내에서 CREATE TYPE MAPPING문을 처리할 수 없습니다(SQLSTATE 55007).
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 내에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문

  - 명령문은 데이터 소스의 범주(예: 특정 유형 및 버전의 모든 데이터 소스)를 참조하며, UOW는 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 중 하나에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문
- 다중 유형 맵핑이 적용 가능할 경우, 가장 최근 맵핑이 사용됩니다. SYSCAT.TYPEMAPPINGS 카탈로그 뷰의 CREATE_TIME 컬럼을 쿼리하여 유형 맵핑의 작성시간을 검색할 수 있습니다.

예:

예 1: Oracle 데이터 유형 DATE와 데이터 유형 SYSIBM.DATE 간의 포워드 유형 맵핑을 작성하십시오. 이 맵핑이 정의된 후에 작성된 모든 별칭에 대해 데이터 유형 DATE의 Oracle 컬럼은 데이터 유형 DATE의 DB2 컬럼에 맵핑됩니다.

```
CREATE TYPE MAPPING MY_ORACLE_DATE
  FROM LOCAL TYPE SYSIBM.DATE
  TO SERVER TYPE ORACLE
  REMOTE TYPE DATE
```

예 2: 데이터 유형 SYSIBM.DECIMAL(10,2)과 데이터 소스 ORACLE1에 있는 Oracle 데이터 유형 NUMBER([10..38],2) 간의 포워드 유형 맵핑을 작성하십시오. Oracle 테이블에 데이터 유형 NUMBER(11,2) 컬럼이 있으면 11이 10과 38 사이에 있기 때문에 데이터 유형 DECIMAL(10,2) 컬럼에 맵핑됩니다.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  FROM LOCAL TYPE SYSIBM.DECIMAL(10,2)
  TO SERVER ORACLE1
  REMOTE TYPE NUMBER([10..38],2)
```

예 3: 데이터 유형 SYSIBM.VARCHAR(*p*)과 데이터 소스 ORACLE1에 있는 Oracle 데이터 유형 CHAR(*p*) 간의 포워드 유형 맵핑을 작성하십시오(*p*는 임의의 길이임). Oracle 테이블에 데이터 유형 CHAR(10) 컬럼이 있으면 이 컬럼은 데이터 유형 VARCHAR(10) 컬럼에 맵핑됩니다.

```
CREATE TYPE MAPPING MY_ORACLE_CHAR
  FROM LOCAL TYPE SYSIBM.VARCHAR()
  TO SERVER ORACLE1
  REMOTE TYPE CHAR()
```

예 4: 데이터 유형 SYSIBM.DECIMAL(10,2)과 데이터 소스 ORACLE2에 있는 Oracle 데이터 유형 NUMBER(10,2) 간의 리버스 유형 맵핑을 작성하십시오. 투명한 DDL을 사용하여 Oracle 테이블을 작성하고 데이터 유형 DECIMAL(10,2) 컬럼을 지정할 경우, DB2는 데이터 유형 NUMBER(10,2) 컬럼이 있는 Oracle 테이블을 작성합니다.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  TO LOCAL TYPE SYSIBM.DECIMAL(10,2)
  FROM SERVER ORACLE2
  REMOTE TYPE NUMBER(10,2)
```

CREATE USER MAPPING

CREATE USER MAPPING문은 페더레이티드 데이터베이스를 사용하는 권한 부여 ID와 지정된 데이터 소스에서 사용할 권한 부여 ID 및 암호간의 맵핑을 정의합니다.

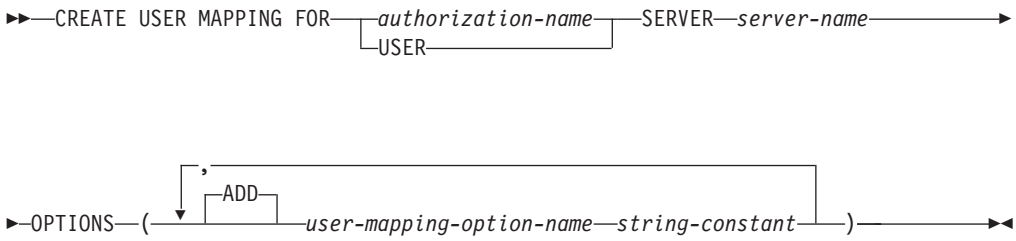
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 데이터 소스에 맵핑되는 권한 부여 이름과 다르면, 명령문의 권한 부여 ID가 보유한 특권은 DBADM 권한을 포함해야 합니다. 그렇지 않으면, 권한 부여 ID가 권한 부여 이름과 일치하는 경우 어떠한 권한이나 특권도 필요하지 않습니다.

구문



설명

authorization-name

사용자나 응용프로그램이 페더레이티드 데이터베이스에 연결할 때 사용하는 권한 부여 이름을 지정합니다. *authorization_name*은 REMOTE_AUTHID 사용자 맵핑 옵션에 맵핑됩니다.

USER

USER 특수 레지스터의 값입니다. USER를 지정한 경우, CREATE USER MAPPING문을 발행하는 권한 부여 ID는 REMOTE_AUTHID 사용자 맵핑 옵션에 맵핑됩니다.

SERVER *server-name*

*authorization-name*이 액세스할 수 있는 데이터 소스에 대한 서버 오브젝트의 이름을 표시합니다. *server-name*은 페더레이티드 데이터베이스에 등록된 리모트 서버의 로컬 이름입니다.

OPTIONS

사용자 맵핑이 작성될 때 사용 가능하게 되는 옵션을 표시합니다.

ADD

하나 이상의 사용자 맵핑 옵션을 활성화합니다.

user-mapping-option-name

옵션 이름을 지정합니다.

string-constant

*user-mapping-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

주

- 사용자 맵핑은 DB2 제품군, Documentum, Informix, Microsoft SQL Server, ODBC, Oracle, Sybase, 및 Teradata와 같은 데이터 소스의 경우에만 필요합니다.
- 사용자 맵핑에 REMOTE_PASSWORD 옵션은 항상 필요합니다.

예:

예 1: z/OS용 DB2 데이터 소스 서버 오브젝트 SERVER390에 사용자 맵핑을 등록합니다. 로컬 페더레이티드 데이터베이스의 권한 부여 이름을 SERVER390의 사용자 ID 및 암호로 맵핑하십시오. 권한 부여 이름은 RSPALTEN입니다. SERVER390의 사용자 ID는 SYSTEM입니다. SERVER390의 암호는 MANAGER입니다.

```
CREATE USER MAPPING FOR RSPALTEN
  SERVER SERVER390
  OPTIONS
  (REMOTE_AUTHID 'SYSTEM',
  REMOTE_PASSWORD 'MANAGER')
```

예 2: Oracle 데이터 소스 서버 오브젝트 ORACLE1에 사용자 맵핑을 등록하십시오. MARCR은 로컬 페더레이티드 데이터베이스의 권한 부여 이름이고 ORACLE1의 사용자 ID입니다. 권한 부여 이름과 사용자 ID가 동일하기 때문에 사용자 맵핑에 REMOTE_AUTHID 옵션을 지정할 필요가 없습니다. ORACLE1에서 MARCR의 암호는 NZXCZY입니다.

```
CREATE USER MAPPING FOR MARCR
  SERVER ORACLE1
  OPTIONS
  (REMOTE_PASSWORD 'NZXCZY')
```

CREATE VARIABLE

CREATE VARIABLE문은 전역 변수를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 변수의 내재되거나 명시된 스키마 이름이 존재하지 않을 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 변수의 스키마 이름이 기존의 스키마를 참조할 경우 스키마에 대한 CREATEIN 특권
- DBADM 권한

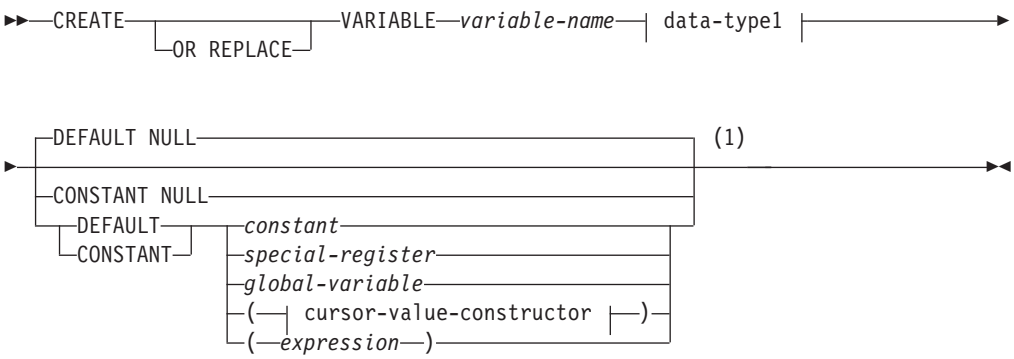
그리고 디폴트 표현식을 실행하는 데 필요한 모든 특권.

*select-statement*를 사용하는 *cursor-value-constructor*가 있는 이 명령문을 실행하려면, 명령문의 권한 부여 ID에 의해 보유된 특권에는 *select-statement*를 실행하는 데 필요한 특권을 포함해야 합니다. "SQL 쿼리"의 권한 부여 섹션을 참조하십시오.

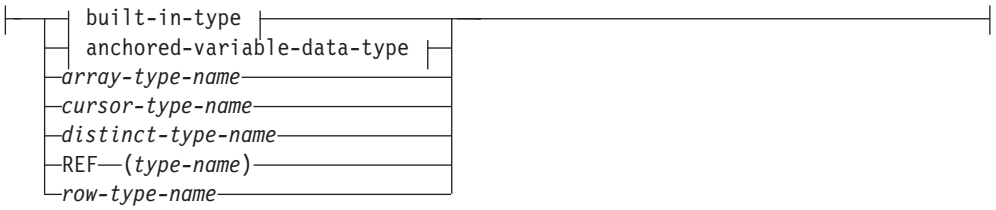
명령문에서 참조되는 오브젝트에 대한 권한 부여를 점검할 때 그룹 특권은 고려되지 않습니다.

기존 변수를 교체하려면 명령문의 권한 부여 ID가 기존 변수의 소유자여야 합니다 (SQLSTATE 42501).

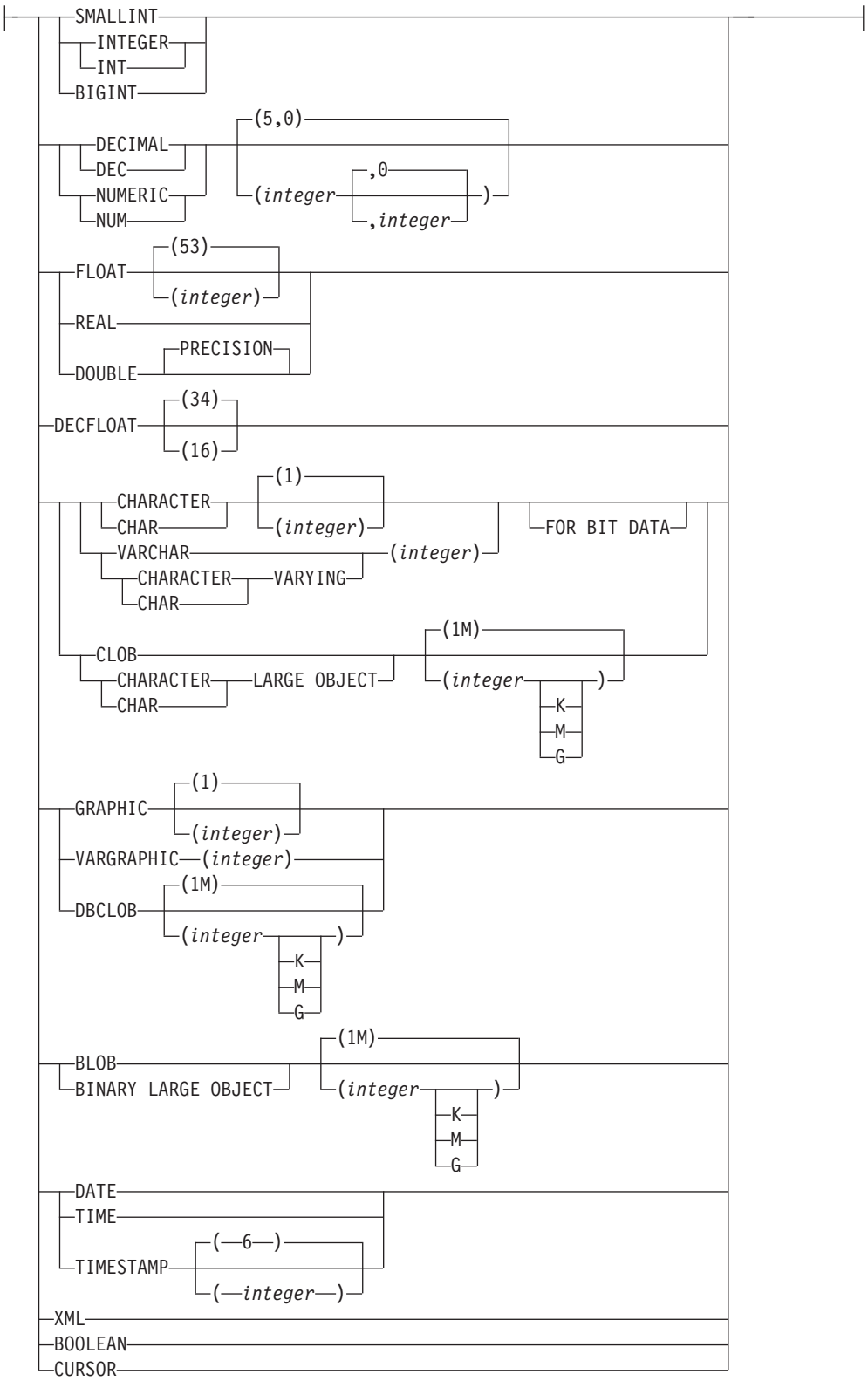
구문



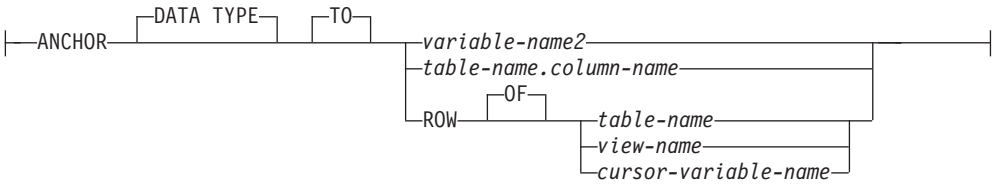
data-type1:



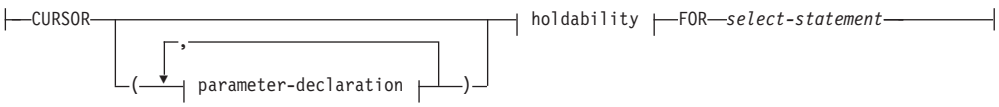
built-in type:



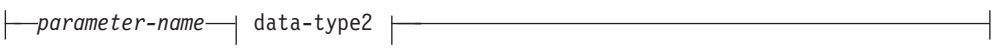
anchored-variable-data-type:



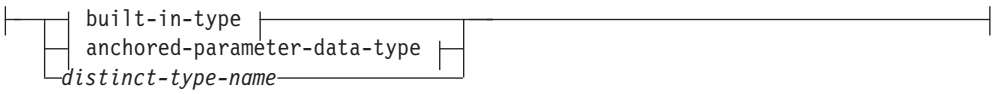
cursor-value-constructor:



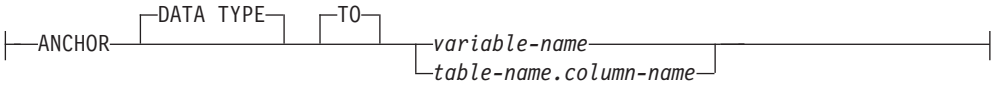
parameter-declaration:



data-type2:



anchored-parameter-data-type:



holdability:

주:

1 *data-type1*이 **CURSOR** 내장 유형 또는 *cursor-type-name*을 지정하는 경우, **NULL** 또는 *cursor-value-constructor*만 지정할 수 있습니다. *array-type-name* 또는 *row-type-name*에 대해서는 **DEFAULT NULL**만 명시적으로 지정할 수 있습니다.

설명

OR REPLACE

변수의 정의가 현재 서버에 존재하는 경우 정의를 교체할 것을 지정합니다. 기존 정의는 카탈로그에서 새 정의가 교체되기 전에 효율적으로 삭제됩니다. 예외로, 변수

에 대한 권한이 부여된 특권에는 영향이 미치지 않습니다. 이 옵션은 변수의 정의가 현재 서버에 존재하지 않는 경우 무시됩니다.

variable-name

전역 변수의 이름을 지정합니다. 내재되거나 명시된 규정자를 포함한 이름은 현재 서버에 이미 존재하는 전역 변수를 식별해서는 안됩니다(SQLSTATE 42710). 규정자가 지정되지 않은 경우에는 현재 스키마가 내재적으로 지정됩니다.

data-type1

전역 변수의 데이터 유형을 지정합니다. 구조화된 유형을 지정할 수 없습니다 (SQLSTATE 42611).

built-in-data-type

내장 데이터 유형을 지정합니다. 테이블에 대해 지정할 수 없는 각 내장 데이터 유형(BOOLEAN 및 CURSOR 제외)에 대한 자세한 설명은 "CREATE TABLE"을 참조하십시오. XML 데이터 유형을 지정할 수 없습니다(SQLSTATE 42611).

FOR BIT DATA는 문자열 데이터 유형의 일부로 지정할 수 있습니다.

BOOLEAN

부울

CURSOR

기반 커서에 대한 참조.

anchored-variable-data-type

전역 변수의 데이터 유형을 판별하는 데 사용되는 다른 오브젝트를 식별합니다. 고정 오브젝트의 데이터 유형은 데이터 유형을 직접 지정하기 위해, 또는 행의 경우 행 유형을 작성하기 위해 적용되는 동일한 제한사항을 수반합니다.

ANCHOR DATA TYPE TO

데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name2

전역 변수를 식별합니다. 참조된 변수의 데이터 유형은 전역 변수의 데이터 유형으로 사용됩니다.

table-name.column-name

기존 테이블 또는 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 전역 변수의 데이터 유형으로 사용됩니다.

ROW OF *table-name* **또는** *view-name*

전역 변수가 *table-name*으로 식별된 테이블 또는 *view-name*으로 식별된 뷰의 컬럼 이름과 컬럼 데이터 유형을 기반으로 하는 이름과 데이터 유형이 포함된 필드 행임을 지정합니다. 전역 변수의 데이터 유형은 unnamed row 자료형입니다.

ROW OF *cursor-variable-name*

*cursor-variable-name*으로 식별된 커서 변수의 필드 이름 및 필드 데이터 유형을 기반으로 하는 이름 및 데이터 유형이 포함된 필드 행을 지정합니다. 지정된 커서 변수는 다음 중 하나여야 합니다(SQLSTATE 428HS).

- 강하게 유형 지정된 커서 데이터 유형이 있는 전역 변수
- 모든 결과 컬럼의 이름이 지정된 select-statement를 지정하는 CONSTANT절로 작성되거나 선언된 약하게 유형 지정된 커서 데이터 유형이 포함된 전역 변수.

커서 변수의 커서 유형이 named row 자료형을 사용하여 강하게 유형이 지정되지 않은 경우 전역 변수의 데이터 유형은 unnamed row 자료형입니다.

array-type-name

사용자 정의 배열 유형의 이름을 지정합니다. 스키마 이름 없이 *array-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 배열 유형이 분석됩니다.

cursor-type-name

커서 유형의 이름을 지정합니다. 스키마 이름 없이 *cursor-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 커서 유형이 분석됩니다.

distinct-type-name

구별 유형의 이름을 지정합니다. 선언된 변수의 길이, 정밀도 및 스케일은 각각 구별 유형의 소스 유형에 대한 길이, 정밀도 및 스케일입니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

REF (*type-name*)

참조 유형을 지정합니다. 유형 이름이 스키마 이름 없이 지정된 경우, SQL 경로에서 스키마를 검색하여 *type-name*이 분석됩니다.

row-type-name

사용자 정의 행 유형의 이름을 지정합니다. 변수 필드는 행 유형의 필드입니다. 스키마 이름 없이 *row-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 행 유형이 분석됩니다.

DEFAULT 또는 CONSTANT

세션에서 전역 변수가 처음 참조될 때 그 전역 변수의 값을 지정합니다. DEFAULT 또는 CONSTANT 절 값은 이 첫 번째 참조에서 판별됩니다. 어느 것도 지정하지 않으면, 전역 변수의 디폴트값은 널(NULL) 값입니다. *array-type-name* 또는 *row-type-name*이 지정되는 경우, DEFAULT NULL만 명시적으로 지정할 수 있습니다.

DEFAULT

전역 변수의 디폴트값을 정의합니다. 디폴트값은 변수의 데이터 유형과 호환 가능한 지정이어야 합니다.

CONSTANT

전역 변수에 변경할 수 없는 고정 값이 있음을 지정합니다. CONSTANT를 사용하여 정의된 전역 변수는 지정 조작의 대상으로 사용할 수 없습니다. 고정 값은 변수의 데이터 유형과 호환 가능한 지정이어야 합니다.

NULL

전역 변수의 디폴트값으로 널(NULL)을 지정합니다. *row-type-name*이 지정된 경우, 전역 변수의 값은 각 필드에 널(NULL) 값이 있는 행입니다.

constant

전역 변수의 디폴트값으로 상수 값을 지정합니다. *data-type1*이 CURSOR 내장 유형 또는 *cursor-type-name*을 지정하는 경우, *constant*를 지정할 수 없습니다(SQLSTATE 42601).

special-register

전역 변수의 디폴트값으로 특수 레지스터 값을 지정합니다. *data-type1*이 CURSOR 내장 유형 또는 *cursor-type-name*을 지정하는 경우, *special-register*를 지정할 수 없습니다(SQLSTATE 42601).

global-variable

전역 변수의 디폴트값으로 전역 변수 값을 지정합니다. *data-type1*이 CURSOR 내장 유형 또는 *cursor-type-name*을 지정하는 경우, *global-variable*을 지정할 수 없습니다(SQLSTATE 42601).

cursor-value-constructor

*cursor-value-constructor*는 전역 변수와 연관된 *select-statement*를 지정합니다. *cursor-value-constructor*를 커서 변수에 지정하면 해당 커서 변수의 기본 커서를 정의합니다.

(parameter-declaration, ...)

각 매개변수의 이름 및 데이터 유형을 포함하는 커서의 입력 매개변수를 지정합니다.

parameter-name

select-statement 내에서 SQL 변수로 사용할 매개변수의 이름을 지정합니다. 이름은 커서의 다른 매개변수 이름과 동일할 수 없습니다. 또한 매개변수 이름 전에 컬럼 이름이 분석되므로 *select-statement*에서 사용할 수 있는 컬럼 이름을 피하도록 이름을 선택해야 합니다.

data-type2

*select-statement*내에서 사용되는 cursor 매개변수의 데이터 유형을 지정합니다.

built-in-type
내장 데이터 유형을 지정합니다. 각 내장 데이터 유형에 대한 자세한 설명은 “CREATE TABLE”을 참조하십시오. BOOLEAN 및 CURSOR 내장 유형을 지정할 수 없습니다(SQLSTATE 429BB).

anchored-parameter-data-type
cursor 매개변수의 데이터 유형을 판별하는 데 사용되는 다른 오브젝트를 식별합니다. 앵커 오브젝트의 데이터 유형은 데이터 유형을 직접 지정할 때 적용되는 것과 동일한 제한사항에 의해 바운드됩니다.

ANCHOR DATA TYPE TO
데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name
전역 변수를 식별합니다. 참조된 변수의 데이터 유형은 cursor 매개변수의 데이터 유형으로 사용됩니다.

table-name.column-name
기존 테이블 또는 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 cursor 매개변수의 데이터 유형으로 사용됩니다.

distinct-type-name
구별 유형의 이름을 지정합니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

holdability
커서가 커미트 조작의 결과로 닫히지 않도록 할지 여부를 지정합니다. 자세한 정보는 “DECLARE CURSOR”를 참조하십시오. 디폴트값은 WITHOUT HOLD입니다.

WITHOUT HOLD
커서가 커미트 조작의 결과로 닫히도록 합니다.

WITH HOLD
여러 작업 단위에서 자원을 유지보수합니다. 커서가 커미트 조작의 결과로 닫히지 않도록 합니다.

SELECT
커서의 SELECT문을 지정합니다. 자세한 정보는 “select-statement”를 참조하십시오.

statement-name
커서의 준비된 *select-statement*을 지정합니다. PREPARE문에 대한 설명은

"PREPARE"를 참조하십시오. 목표 커서 변수에 강하게 유형 지정된 사용자 정의 커서 유형인 데이터 유형이 있어서는 안 됩니다(SQLSTATE 428HU).

expression

전역 변수의 디폴트값으로 표현식 값을 지정합니다. 표현식은 "표현식"에 설명되어 있는 모든 유형의 표현식이 될 수 있습니다. 표현식은 변수 데이터 유형과 호환 가능한 지정이어야 합니다. 표현식의 최대 *크기*는 64K입니다. 디폴트 표현식은 SQL 데이터를 수정하거나(SQLSTATE 428FL) 외부 조치를 수행하면 안됩니다(SQLSTATE 42845). *data-type1*이 CURSOR 내장 유형 또는 *cursor-type-name*을 지정하는 경우, *expression*을 지정할 수 없습니다(SQLSTATE 42601).

규칙

- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.

주

- 전역 변수에는 세션 범위가 있습니다. 이는 데이터베이스에서 활성 상태인 모든 세션에 전역 변수를 사용할 수 있어도 해당 값은 세션마다 개별적임을 의미합니다.
- ***배열, 부울, 커서 및 행 전역 변수의 컨텍스트:*** 배열 변수, 부울 변수 또는 행 변수인 전역 변수는 복합 SQL(컴파일된)문 또는 SET 변수 명령문에서만 사용할 수 있습니다. 커서 변수인 전역 변수는 복합 SQL(컴파일된)문에서만 사용할 수 있습니다.
- ***오류와 함께 작성:*** 기본 표현식에서 참조된 오브젝트가 존재하지 않거나 유효하지 않은 것으로 표시되는 경우 또는 정의자에게 일시적으로 오브젝트에 액세스할 수 있는 특권이 없는 경우 그리고 데이터베이스 구성 매개변수 **auto_reval**이 DISABLED로 설정되지 않은 경우 변수는 계속 작성됩니다. 변수는 유효하지 않은 것으로 표시되고 다음에 호출될 때 유효성이 다시 확인됩니다.
- ***전역 변수 값의 범위:*** 전역 변수의 값은 현재 세션에서 갱신되거나, 전역 변수가 삭제 또는 변경되거나, 응용프로그램 세션을 종료할 때까지 지속적입니다. 값은 COMMIT 또는 ROLLBACK문에 영향을 받지 않습니다. 전역 변수의 디폴트값은 결정적이 아닐 수 있으며 전역 변수에 대해 디폴트값이 계산되는 때에 따라 달라집니다(예: 일의 시간에 대한 참조 또는 테이블에 저장된 일부 데이터에 대한 참조).

  흔히 사용되는 기술, 특히 성능은 연결 설정을 관리하고 임의 연결에 대한 트랜잭션을 라우트할 응용프로그램 또는 제품을 위한 것입니다. 이러한 상황에서, 전역 변수의 디폴트값이 아니거나 전역 변수의 결정적 초기 디폴트값이 아닌 값은 트랜잭션을 종료할 때까지만 의존해야 합니다. 이러한 상황 유형의 예에는 다음과 같은 응용프

로그램을 포함할 수 있습니다(XA 프로토콜, 연결 풀링, 연결 집중기(connection concentrator) 및 HADR을 사용하여 장애 복구 달성).

- ***전역 변수를 사용하기 위한 특권:*** 명령문으로 작성된 전역 변수에서 읽거나 전역 변수에 쓰려면 조치를 시도하는 권한 부여 ID가 전역 변수에 대한 적절한 특권을 가지고 있어야 합니다. 변수의 정의자에게는 변수에 대한 모든 특권이 내재적으로 부여됩니다.
- ***디폴트값 설정:*** 작성된 전역 변수는 제공된 범위 내에서 처음 참조될 때 디폴트값으로 인스턴스화됩니다. 명령문에 전역 변수가 참조되는 경우 해당 명령문에 대한 제어 순서와 독립적으로 인스턴스화됩니다.
- ***새로 작성된 세션 전역 변수 사용:*** 세션 내에 전역 변수가 작성된 경우 작업 단위가 커미트될 때까지 다른 세션에서 사용할 수 없습니다. 그러나 작업 단위가 커미트되기 전에 변수를 작성한 세션 내에서 새 전역 변수가 사용될 수 있습니다.

예:

예 1: 세션에 사용할 프린터를 표시하기 위한 전역 변수를 작성하십시오.

```
CREATE VARIABLE MYSCHEMA.MYJOB_PRINTER VARCHAR(30)
  DEFAULT 'Default printer'
```

예 2: 사원이 근무하는 부서를 표시하기 위한 전역 변수를 작성하십시오.

```
CREATE VARIABLE SCHEMA1.GV_DEPTNO INTEGER
  DEFAULT ((SELECT DEPTNO FROM HR.EMPLOYEES
    WHERE EMPUSER = SESSION_USER))
```

예 3: 현재 사용자의 보안 레벨을 표시하기 위한 전역 변수를 작성하십시오.

```
CREATE VARIABLE SCHEMA2.GV_SECURITY_LEVEL INTEGER
  DEFAULT (GET_SECURITY_LEVEL (SESSION_USER))
```

예 4: 지정된 작업 유형의 각 직원의 이름을 리턴하는 STAFF 테이블의 커서로서 전역 변수를 작성합니다. 부서 번호별로 결과를 정렬하십시오.

```
CREATE VARIABLE STAFFJOBS CURSOR
  CONSTANT (CURSOR (WHICHJOB CHAR(5))
  FOR SELECT NAME, DEPT FROM STAFF WHERE JOB = WHICHJOB
    ORDER BY DEPT)
```

CREATE VIEW

CREATE VIEW문은 하나 이상의 테이블, 뷰 또는 별칭에 대한 뷰를 정의합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 뷰의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 뷰의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

또한 fullselect에서 식별되는 각 테이블, 뷰 또는 별칭에 대해 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- 테이블, 뷰 또는 별칭에 대한 SELECT 특권
- DATAACCESS 권한

서브뷰를 작성한 경우:

- 명령문의 권한 부여 ID는 테이블 계층의 루트 테이블의 정의자와 동일해야 합니다. 또는
- 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

및

- 명령문의 권한 부여 ID에는 서브뷰의 하위 테이블에 대한 SELECT WITH GRANT 권한이 있어야 하며, 또는 수퍼 뷰에는 뷰 정의자가 아닌 다른 사용자에게 SELECT 특권을 부여해서는 안됩니다. 또는
- ACCESSCTRL 권한 및 다음 중 하나여야 합니다.
  - 서브뷰의 하위 테이블에 대한 SELECT 특권
  - DATAACCESS 권한

WITH ROW MOVEMENT를 지정하면 명령문의 권한 부여 ID에서 갖고 있는 특권에는 최소한 다음 중 하나가 포함되어야 합니다.

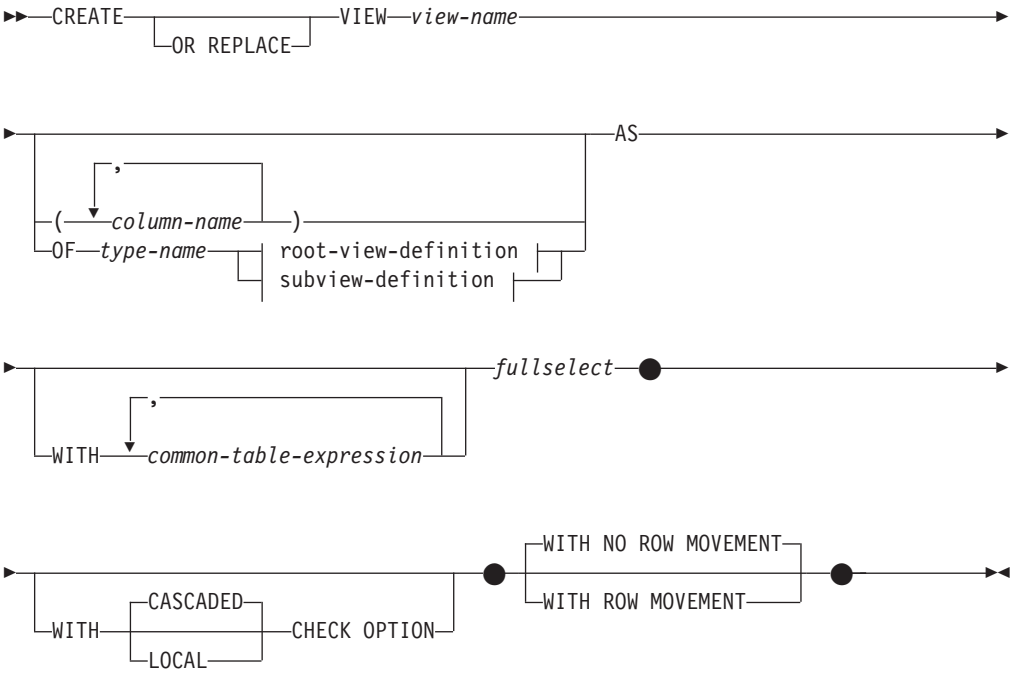
- 테이블 또는 뷰에 대한 UPDATE 특권
- DATAACCESS 권한

CREATE VIEW문에 지정된 테이블이나 뷰에 대해서는 그룹 특권이 고려되지 않습니다.

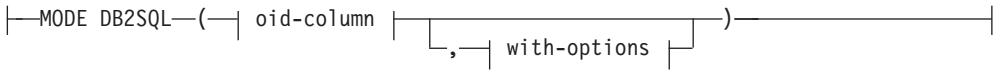
페더레이티드 데이터베이스 별칭에서 뷰를 정의할 때에는 특권이 고려되지 않습니다. 별칭으로 참조된 테이블이나 뷰에 대한 데이터 소스의 권한 부여 요건은 쿼리가 처리될 때 적용됩니다. 명령문의 권한 부여 ID는 서로 다른 리모트 권한 부여 ID로 맵핑될 수도 있습니다.

기존 뷰를 교체하려면 명령문의 권한 부여 ID가 기존 뷰의 소유자여야 합니다 (SQLSTATE 42501).

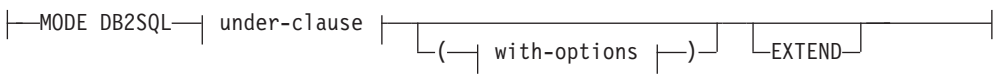
구문



root-view-definition:

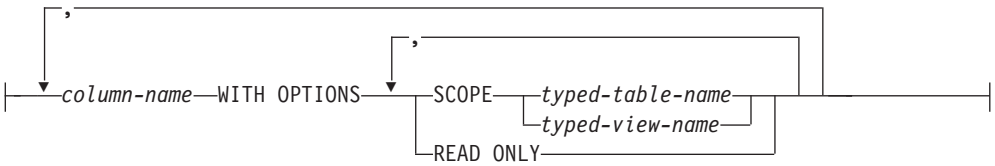


subview-definition:



oid-column:

with-options:



under-clause:

설명

OR REPLACE

현재 서버에 정의가 존재하면 정의를 교체하도록 지정합니다. 새 정의가 카탈로그에서 교체되기 전에 기존 정의가 삭제되며, 뷰에서 부여되었던 특권에는 영향을 주지 않습니다. 뷰에 대한 정의가 현재 서버에 존재하지 않으면 이 옵션이 무시됩니다.

view-name

뷰를 이름 지정합니다. 내재적 또는 명시적인 규정자를 포함하여 이름은 카탈로그에 설명된 테이블, 뷰, 별칭 또는 별명을 식별해서는 안됩니다. 규정자는 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT(SQLSTATE 42939)이 될 수 없습니다.

이름은 작동 불능 뷰의 이름과 같을 수 있습니다(작동 불능 뷰 참조). 이 경우, CREATE VIEW문에 지정된 새 뷰는 작동 불능 뷰를 대체합니다. 사용자는 작동 불능 뷰가 대체될 때 경고 (SQLSTATE 01595)를 받게 됩니다. 응용프로그램이 바인드 옵션 SQLWARN을 NO로 설정하여 바인드한 경우 경고가 리턴되지 않습니다.

column-name

뷰의 컬럼을 이름 지정합니다. 컬럼 이름의 목록이 지정되면, 이 목록에 포함된 이름 수는 fullselect의 결과 테이블에 있는 컬럼 수와 동일해야 합니다. 각 *column-name*은 고유해야 하고 규정화되어서는 안됩니다. 컬럼 이름의 목록이 지정되지 않으면, 뷰의 컬럼은 fullselect의 결과 컬럼에 대한 이름을 상속합니다.

fullselect의 결과 테이블에 중복된 컬럼 이름이나 알려져 있지 않은 컬럼이 있는 경우, 컬럼 이름 목록을 반드시 지정해야 합니다(SQLSTATE 42908). 이름 지정되

지 않은 컬럼은 컨테이너, 함수, 표현식 또는 세트 조작으로부터 유래된 컬럼으로서, 선택 목록의 AS절을 사용하여 이름 지정되지 않았습니다.

OF *type-name*

뷰의 컬럼들이 *type-name*에 의해 식별되는 구조화된 유형 속성에 기초하도록 지정합니다. 스키마 이름 없이 *type-name*이 지정되면, 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 사전 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의)에서 스키마를 검색하여 분석됩니다. 유형 이름은 기존 사용자 정의 유형의 이름이어야 하며(SQLSTATE 42704) 인스턴스 작성 가능한 구조화된 유형이어야 합니다(SQLSTATE 428DP).

MODE DB2SQL

이 절은 유형이 지정된 뷰의 모드를 지정하는 데 사용됩니다. 이것은 현재 지원되는 유일하게 유효한 모드입니다.

UNDER *superview-name*

이 뷰가 *superview-name*의 서브뷰임을 나타냅니다. 슈퍼 뷰는 기존의 뷰여야 하고(SQLSTATE 42704), 이 뷰는 *type-name*의 바로 위 슈퍼 유형인 구조화된 유형을 사용하여 정의되어야 합니다(SQLSTATE 428DB). *view-name* 및 *superview-name*의 스키마 이름은 같아야 합니다(SQLSTATE 428DQ). *superview-name*으로 식별되는 뷰에는 *type-name*을 사용하여 이미 정의된 기존의 서브뷰가 없어야 합니다(SQLSTATE 42742).

뷰의 컬럼에는 REF(*type-name*)로 유형이 수정될 슈퍼 뷰의 오브젝트 ID 컬럼이 포함됩니다. 이 뒤에는 *type-name* 속성에 기초한 컬럼이 옵니다. (유형에는 슈퍼 유형의 속성이 포함되어야 합니다.)

INHERIT SELECT PRIVILEGES

슈퍼 뷰에 대한 SELECT 특권을 보유하고 있는 사용자 또는 사용자 그룹에는 새로 작성된 서브뷰에도 그에 상응하는 특권이 권한 부여됩니다. 서브뷰 정의자가 이 특권의 권한을 준 사용자로 간주됩니다.

OID-column

유형이 지정된 뷰의 오브젝트 ID 컬럼을 정의합니다.

REF IS *OID-column-name* **USER GENERATED**

오브젝트 ID(OID) 컬럼이 첫 번째 컬럼으로서 뷰에 정의되도록 지정합니다. 뷰 계층의 루트 뷰에는 OID가 필수적입니다(SQLSTATE 428DX). 뷰는 서브뷰가 아닌 유형이 지정된 뷰(OF절이 존재해야 하는)여야 합니다(SQLSTATE 42613). 컬럼 이름은 *OID-column-name*으로서 정의되고, 구조화된 유형 *type-name*의 속성 이름과 같아서는 안됩니다(SQLSTATE 42711). *fullselect*에서 지정된 첫 번째 컬럼은 REF(*type-name*) 유형이어야 합니다(적절한 키를 가지도록 이를 캐스트해야 합니다). UNCHECKED가 지정되지 않으면, 인덱스(기본 키, 고유 제한조건, 고유 인덱스 또는 0ID 컬럼)를 통해 고유성이 강요되는 널(NULL) 입력 가능 컬럼이 기초가 되어야 합니다. 이 컬럼을 오브젝트

ID 컬럼 또는 *OID 컬럼*이라고 합니다. 키워드 USER GENERATED는 행을 삽입할 때 사용자가 OID 컬럼에 대한 초기 값을 제공해야 한다는 것을 나타냅니다. 일단 행이 삽입되면 OID 컬럼은 갱신할 수 없습니다(SQLSTATE 42808).

UNCHECKED

시스템이 이 고유성을 증명할 수 없을 지라도 고유성을 가정하도록 유형화 뷰 정의의 오브젝트 ID 컬럼을 정의합니다. 이는 사용자가 데이터가 이 고유성 규칙을 준수하나 시스템이 고유성을 증명하게 허용하는 규칙에는 따르지 않음을 알고 있는 유형화 뷰 계층으로 정의되고 있는 테이블이나 뷰와 사용하기 위한 것입니다. UNCHECKED 옵션은 여러 계층이나 레거시 테이블 또는 뷰에 걸치는 범위의 뷰 계층을 위한 필수입니다. UNCHECKED를 지정하여, 사용자는 뷰의 각 행이 반드시 고유 OID를 가지게 할 책임을 집니다. 사용자가 이러한 등록 정보를 보장하지 못하고, 뷰에 중복 OID 값이 있으며, 비고유 OID 값 중 하나를 수반하는 path-expression 또는 DEREF 연산자가 오류를 내보낼 수도 있습니다(SQLSTATE 21000).

with-options

유형이 지정된 뷰의 컬럼에 적용되는 추가 옵션을 정의합니다.

column-name **WITH OPTIONS**

추가 옵션이 지정될 컬럼의 이름을 지정합니다. *column-name*은 뷰의 *type-name*에 정의된(상속된 것이 아닌) 속성 이름과 일치해야 합니다. 컬럼은 참조 유형(SQLSTATE 42842)이어야 합니다. 슈퍼 뷰에 있는 컬럼일 수 없습니다(SQLSTATE 428DJ). 컬럼 이름은 명령문의 한 WITH OPTIONS SCOPE 절에만 나타날 수 있습니다(SQLSTATE 42613).

SCOPE

참조 유형 컬럼 영역을 식별합니다. 비참조 연산자의 왼쪽 피연산자 또는 DEREF 함수의 인수로서 사용될 예정인 컬럼에 대한 영역이 지정되어야 합니다.

참조 유형 컬럼 영역을 지정하는 것은 (영역을 상속받지 않은 경우) 후속 ALTER VIEW 명령으로 지연될 수 있습니다. 이는 상호 참조 뷰 및 테이블에서는 항상 목표 테이블이나 뷰가 정의될 수 있도록 하기 위한 것입니다. 뷰의 참조 유형 컬럼에 대해 영역이 지정되지 않고 기본 테이블 또는 뷰 컬럼 영역이 지정된 경우, 참조 유형 컬럼에 의해 기본 컬럼의 영역이 상속됩니다. 기본 테이블이나 뷰 컬럼에 영역이 없었던 경우 컬럼은 영역이 지정되지 않은 범위로 남아 있게 됩니다. 영역 및 참조 유형 컬럼에 대한 878 페이지의 『주』에서 좀더 자세한 정보를 참조하십시오.

typed-table-name

유형이 지정된 테이블의 이름입니다. 이미 테이블이 있어야 하고, 그렇지 않은 경우 작성되는 테이블 이름과 동일해야 합니다(SQLSTATE 42704).

*column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-table-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하는지 확인하기 위해 *column-name*에 있는 기존 값을 점검하지 않습니다.

typed-view-name

유형이 지정된 뷰의 이름입니다. 이미 뷰가 있어야 하고, 그렇지 않은 경우 작성되는 뷰 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서, *S*는 *typed-view-name* 유형입니다(SQLSTATE 428DM). 값이 *typed-view-name*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *column-name*에 있는 기존 값들에 대해 점검을 하지 않습니다.

READ ONLY

컬럼을 읽기 전용 컬럼으로 식별합니다. 이 옵션은 슈퍼 뷰 정의가 내재적으로 읽기 전용인 컬럼에 대해 표현식을 지정할 수 있도록 컬럼을 강제로 읽기 전용 컬럼으로 만드는 데 사용됩니다.

AS

뷰 정의를 식별합니다.

WITH *common-table-expression*

뒤에 나올 fullselect과 함께 사용할 공통 테이블 표현식을 정의합니다. 유형이 지정된 뷰를 정의할 경우에는 공통 테이블 표현식을 지정할 수 없습니다.

fullselect

뷰를 정의합니다. 언제든지, 뷰는 SELECT문이 실행된 경우 그 결과 행으로 구성됩니다. fullselect는 호스트 변수, 매개변수 표시문자 또는 선언된 임시 테이블을 참조해서는 안됩니다. 그러나 매개변수 작성 뷰는 SQL 테이블 함수로 작성할 수 있습니다.

fullselect는 FROM절에 SQL 데이터 변경문을 포함할 수 없습니다(SQLSTATE 428FL).

유형이 지정된 뷰 및 서브뷰의 경우: *fullselect*는 다음 규칙을 따라야 하며, 그렇지 않으면 오류가 리턴됩니다(지정되지 않을 경우 SQLSTATE 428EA).

- fullselect는 DBPARTITIONNUM 또는 HASHEDVALUE 함수, 결정할 수 없는 함수 또는 외부 조치를 갖도록 정의된 함수에 대한 참조를 포함해서는 안됩니다(SQLSTATE 428EA).
- 뷰의 본문은 단일 subselect 또는 두 개 이상의 subselect의 UNION ALL로 이루어져야 합니다. 뷰 본문에 직접 참여하고 있는 subselect의 각각을 뷰의 *분기*라고 부릅니다. 뷰에는 하나 이상의 가지가 있을 수 있습니다.
- 각 분기의 FROM절은 해당 분기의 *기초가 되는* 테이블 또는 뷰라고 하는 단일 테이블 또는 뷰(반드시 유형화될 필요는 없음)로 구성되어야 합니다.

- 각 분기의 기초가 되는 테이블이나 뷰는 별도의 계층에 있어야 합니다. (즉, 뷰에는 동일한 계층에 있는 하위 테이블이나 뷰를 가지고 있는 여러 개의 분기가 있을 수 있습니다.)
- 유형화 뷰 정의의 분기 중 어느 것도 GROUP BY 또는 HAVING을 지정하지 않을 것입니다.
- 뷰 본문이 UNION ALL을 포함하면, 계층의 루트 뷰는 OID 컬럼에 대한 UNCHECKED 옵션을 지정해야 합니다.

뷰 및 서브뷰의 계층에서, BR1 및 BR2를 계층의 뷰 정의에 나타나는 분기라고 가정을 합시다. T1을 BR1의 하위 테이블이나 뷰라 하고, T2를 BR2의 하위 테이블이나 뷰라고 가정한 후 그런 다음:

- T1과 T2가 동일한 계층에 있지 않으면, 뷰 계층의 루트 뷰가 그 OID 컬럼에 대한 UNCHECKED 옵션을 지정해야 합니다.
- T1과 T2가 동일한 계층에 있는 경우, BR1과 BR2는 그들의 행 세트들이 해체될 수 있는 충분한 술어나 ONLY절을 포함해야 합니다.

EXTEND AS를 사용하여 정의된 유형화 서브뷰의 경우, 서브뷰의 내용에 있는 모든 가지에 대해

- 각 분기의 하위 테이블은 바로 위 슈퍼뷰의 몇몇 하위 테이블이어야 합니다.
- SELECT 목록의 표현식은 서브뷰의 상속되지 않은 컬럼에 지정할 수 있어야 합니다(SQLSTATE 42854).

EXTEND없이 AS를 사용하여 정의된 유형화 서브뷰의 경우,

- 서브뷰의 본문에 있는 모든 가지의 경우, SELECT-목록의 표현식이 서브뷰의 상속 및 비 상속 컬럼의 선언된 유형으로 지정 가능해야 합니다(SQLSTATE 42854).
- 서브뷰에서 주어진 계층에 대한 각 분기의 OID-표현식은 루트 뷰에서 동일한 계층에 대한 분기의 OID-표현식에 상응해야 합니다(캐스팅의 경우는 제외).
- 슈퍼 뷰에서 READ ONLY로 정의되지 않은(내재적 또는 명시적으로) 컬럼에 대한 표현식은 그 서브뷰에 있는 동일한 하위 계층에 대한 모든 분기에서 동등해야 합니다.

WITH CHECK OPTION

뷰를 통해 삽입되거나 갱신되는 모든 행은 뷰의 정의를 따라야 한다는 제한 규정을 지정합니다. 뷰의 정의에 따르지 않는 행은 뷰의 검색 조건을 만족시키지 않는 행입니다.

다음 조건 중 하나라도 만족하는 경우에는 WITH CHECK OPTION을 지정하면 안됩니다.

- 뷰가 읽기 전용입니다(SQLSTATE 42813). WITH CHECK OPTION이 삽입을 허용하지 않는 갱신 가능한 뷰에 대해 지정된 경우, 제한조건은 갱신사항에만 적용됩니다.
- 뷰는 DBPARTITIONNUM 또는 HASHEDVALUE 함수, 비결정적 함수 또는 외부 조치를 가진 함수를 참조합니다(SQLSTATE 42997).
- 별칭이 뷰의 갱신 목표입니다.
- INSTEAD OF 트리거가 정의되어 있는 뷰가 뷰의 갱신 목표입니다(SQLSTATE 428FQ).

WITH CHECK OPTION이 생략되면, 뷰의 정의가 뷰를 사용하는 삽입 또는 갱신 조작을 점검할 때 사용되지 않습니다. 뷰가 직접적으로 또는 간접적으로 WITH CHECK OPTION을 포함하는 다른 뷰에 대해 종속적일 때, 삽입 또는 갱신 조작 시 일부 점검은 계속 발생할 수도 있습니다. 뷰의 정의가 사용되지 않으므로, 행은 뷰의 정의를 따르지 않는 뷰를 통해 삽입되거나 갱신됩니다.

CASCADED

뷰 *V*에 있는 WITH CASCADED CHECK OPTION 제한조건은 *V*가 종속되어 있는 갱신 가능 뷰로부터의 제한조건으로 검색 조건을 승계함을 의미합니다. 뿐만 아니라, *V*에 종속되어 있는 모든 갱신 가능 뷰는 이러한 제한조건의 주제가 됩니다. 그러므로 *V*의 검색 조건과 *V*가 종속되어 있는 각 뷰는 함께 추가되어 *V* 또는 *V*에 종속되어 있는 모든 뷰의 삽입 또는 갱신에 적용되는 제한조건을 이룹니다.

LOCAL

뷰 *V*에 있는 WITH CASCADED CHECK OPTION 제한조건은 *V* 또는 *V*에 종속되어 있는 모든 뷰의 삽입 또는 갱신에 대한 제한조건으로 *V*의 검색 조건이 적용됨을 의미합니다.

CASCADED와 LOCAL 사이의 차이는 다음 예에 표시됩니다. 다음의 갱신 가능 뷰를 고려하십시오(뒤에 오는 테이블의 컬럼 표제에서 Y 대체).

```
V1 defined on table T
   V2 defined on V1 WITH Y CHECK OPTION
   V3 defined on V2
   V4 defined on V3 WITH Y CHECK OPTION
   V5 defined on V4
```

다음 표는 삽입 또는 갱신된 행이 검사되는 검색 조건을 나타냅니다.

	Y = LOCAL	**Y = CASCADED**
V1 checked against:	뷰 없음	뷰 없음
V2 checked against:	V2	V2, V1
V3 checked against:	V2	V2, V1
V4 checked against:	V2, V4	V4, V3, V2, V1
V5 checked against:	V2, V4	V4, V3, V2, V1

디폴트 CASCADED 옵션을 사용한 WITH CHECK OPTION의 효과를 보여주는 다음과 같은 갱신 가능 뷰를 고려하십시오.

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

*V1*을 사용하는 다음의 INSERT문은 *V1*이 WITH CHECK OPTION을 갖고 있지 않고 *V1*이 WITH CHECK OPTION을 갖는 다른 뷰의 영향을 받지 않으므로 제대로 실행됩니다.

```
INSERT INTO V1 VALUES(5)
```

*V2*를 사용하는 다음의 INSERT문은 *V2*가 WITH CHECK OPTION을 갖고 있고 삽입이 *V2*의 정의를 따르지 않는 행을 생성하므로 오류가 발생합니다.

```
INSERT INTO V2 VALUES(5)
```

*V3*를 사용하는 다음의 INSERT문은 *V3*가 WITH CHECK OPTION이 있는 *V2*에 따라 달라지므로 WITH CHECK OPTION이 없더라도 오류가 발생합니다 (SQLSTATE 44000).

```
INSERT INTO V3 VALUES(5)
```

*V3*를 사용하는 다음의 INSERT문은 *V3*(*V3*는 WITH CHECK OPTION을 갖고 있지 않음)의 정의에 따르지 않을 경우에도 제대로 실행됩니다. WITH CHECK OPTION을 갖고 있지 않는 *V2*의 정의에 따릅니다.

```
INSERT INTO V3 VALUES(200)
```

WITH NO ROW MOVEMENT 또는 WITH ROW MOVEMENT

행 갱신 시 하위 테이블의 점검 제한조건에 위반되는 경우 갱신 가능한 UNION ALL 뷰에 대해 수행할 조치를 지정합니다. 디폴트값은 WITH NO ROW MOVEMENT입니다.

WITH NO ROW MOVEMENT

행 갱신 시 하위 테이블의 점검 제한조건에 위반되는 경우 오류가 리턴되도록 지정합니다(SQLSTATE 23513).

WITH ROW MOVEMENT

하위 테이블의 점검 제한조건에 위반되는 경우라도 갱신된 행이 올바른 기본 테이블로 이동하도록 지정합니다.

행 이동에는 점검 제한조건을 위반한 행 삭제 및 해당 행 뷰에 다시 삽입이 수반됩니다. 모든 컬럼이 갱신 가능한 UNION ALL 뷰에 대해서만 WITH ROW MOVEMENT절을 지정할 수 있습니다(SQLSTATE 429BJ). 행이 삭제됐던 동일한 하위 테이블에 행을 삽입하는 경우(AFTER 트리거 활성화) 오류가 발생합니다(SQLSTATE 23524). WITH ROW MOVEMENT를 사용하여 정의한

뷰에서는 UNION ALL 조작이 중첩될 수 없지만 가장 외부의 fullselect에서는 중첩될 수도 있습니다(SQLSTATE 429BJ).

주

- 아직 존재하지 않는 스키마 이름을 사용하여 뷰를 작성할 경우 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 갖고 있다면 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 뷰 컬럼은 컬럼이 표현식에서 파생될 경우를 제외하고 기본 테이블이나 뷰로부터 NOT NULL WITH DEFAULT 속성을 상속합니다. 행이 갱신 가능한 뷰에 삽입되거나 갱신될 때, 기본 테이블에 제한조건이 정의되어 있으면 제한조건(기본 키, 참조 무결성 및 점검)에 대한 점검이 수행됩니다.
- 정의에서 작동 불능 뷰를 사용할 경우 새로운 뷰를 작성할 수 없습니다 (SQLSTATE 51024).
- 뷰 본문에서 참조된 오브젝트가 존재하지 않거나 유효하지 않은 것으로 표시되거나 정의자에 임시로 오브젝트 액세스 특권이 없고 데이터베이스 구성 매개변수 **auto_reval**이 DISABLED로 설정되어 있지 않으면 뷰가 성공적으로 여전히 작성됩니다. 뷰는 유효하지 않은 것으로 표시되며 다음 참조 시 유효성이 다시 확인됩니다.
- 이 명령문은 선언된 임시 테이블을 지원하지 않습니다(SQLSTATE 42995).
- *삭제 가능 뷰:* 삭제 조작에 대한 INSTEAD OF 트리거가 뷰에 대해 정의되어 있거나 다음 중 모두에 해당되면 *삭제 가능* 뷰입니다.
  - 다른 fullselect의 각 FROM절이 하나의 기본 테이블(OUTER절 없음), 삭제 가능 뷰(OUTER절 없음), 삭제 가능 중첩 테이블 표현식 또는 삭제 가능 공통 테이블 표현식(별칭을 식별할 수 없음)만 식별하는 경우
  - 외부 fullselect에 VALUES절이 포함되지 않는 경우
  - 외부 fullselect에 GROUP BY절 또는 HAVING절이 포함되지 않는 경우
  - 외부 fullselect에 선택 목록 내의 컬럼 함수가 포함되지 않는 경우
  - 외부 fullselect에 UNION ALL 예외가 있는 SET 조작(UNION, EXCEPT 또는 INTERSECT)이 포함되지 않은 경우
  - UNION ALL의 피연산자에 있는 기본 테이블들이 동일한 테이블이 아니고 각 피연산자가 삭제 가능해야 하는 경우
  - 외부 fullselect의 선택 목록에 DISTINCT가 포함되지 않는 경우
- *갱신 가능 뷰:* 갱신 조작에 대해 INSTEAD OF 트리거가 뷰에 정의되어 있거나 다음 중 모두에 해당되면 *갱신 가능* 뷰입니다.
  - 삭제에 대한 INSTEAD OF 트리거가 정의되어 있는지 여부에 상관없이 뷰가 삭제 가능 뷰이고, 비참조 연산을 사용하지 않고 컬럼이 기본 테이블의 컬럼을 분석하며, READ ONLY 옵션이 지정되어 있지 않습니다.

  - 뷰의 fullselect에 UNION ALL이 들어 있는 경우, UNION ALL 피연산자의 모든 해당 컬럼에 정확하게 일치하는 데이터 유형(길이, 정밀도, 스케일 포함)과 일치하는 디폴트값이 들어 있습니다.

  뷰의 모든 컬럼이 갱신 가능하면, 뷰도 갱신 가능합니다.

- ***삽입 가능한 뷰:*** 삽입 조작에 대한 INSTEAD OF 트리거가 뷰에 대해 정의되어 있고, 갱신에 대한 INSTEAD OF 트리거가 정의되어 있는지 여부에 상관없이 뷰의 컬럼 중 적어도 한 컬럼이 갱신 가능하며, 뷰의 fullselect에 UNION ALL이 포함되어 있지 않으면 삽입 가능 뷰입니다.

  하위 기본 테이블 중 한 테이블의 점검 제한조건을 충족할 때만 지정한 행을 뷰에 삽입할 수 있습니다(UNION ALL 포함).

  갱신할 수 없는 컬럼이 있는 뷰에 삽입하려면 컬럼 목록에서 해당 컬럼을 생략해야 합니다.

- ***읽기 전용 뷰:*** 뷰를 삭제, 갱신 또는 삽입할 수 없으면 *읽기 전용* 뷰입니다.

  SYSCAT.VIEWS 카탈로그 뷰의 READONLY 컬럼은 INSTEAD OF 트리거를 고려하지 않고 뷰가 읽기 전용인지를 나타냅니다.

- 공통 테이블 표현식과 중첩 테이블 표현식은 동일한 규칙을 사용하여 삭제 가능, 갱신 가능, 삽입 가능 또는 읽기 전용 여부를 판별합니다.

- ***작동 불능 뷰:*** *작동 불능 뷰*는 SQL문에 대해 더 이상 사용할 수 없는 뷰입니다. 다음의 경우, 작동 불능 뷰가 됩니다.
  - 뷰 정의가 종속되어 있는 특권이 취소되는 경우
  - 뷰 정의가 종속되어 있는 테이블, 별칭, 별명 또는 함수와 같은 오브젝트가 삭제되는 경우
  - 뷰 정의가 종속되어 있는 뷰가 작동 불능 상태가 되는 경우
  - 뷰 정의(서브뷰)의 슈퍼 뷰인 뷰가 작동 불능 상태가 되는 경우

  실제 용어에서, 작동 불능 뷰는 뷰 정의가 실수로 삭제된 뷰입니다. 예를 들어, 별명이 삭제되면, 그 별명을 사용하여 정의된 뷰가 작동 불능 상태가 됩니다. 모든 종속 뷰도 작동 불능 상태가 되고, 뷰에 종속되는 패키지는 더 이상 유효하지 않게 됩니다.

  작동 불능 뷰가 명시적으로 재작성되거나 삭제될 때까지, 작동 불능 뷰를 사용하는 명령문은 컴파일될 수 없습니다(SQLSTATE 51024). 이 경우, CREATE ALIAS, CREATE VIEW, DROP VIEW 및 COMMENT ON TABLE문은 예외입니다. 작동 불능 뷰가 명시적으로 삭제될 때까지, 다른 테이블이나 별명을 작성하는 데 규정된 이름을 사용할 수 없습니다(SQLSTATE 42710).

작동 불능 뷰는 작동 불능 뷰의 정의 텍스트를 사용하는 CREATE VIEWS문을 발행하여 재작성됩니다. 이 뷰 정의 텍스트는 SYSCAT.VIEWS 카탈로그의 TEXT 컬럼에 저장됩니다. 작동 불능 뷰를 재작성할 때, 다른 사용자에게 필요한 특권을 명시적으로 부여하는 것이 필요한데, 이는 뷰가 작동 불능으로 표시되면 뷰에 있는 모든 권한 부여 레코드가 삭제되기 때문입니다. 작동 불능 뷰를 재작성하기 위해 이를 명시적으로 삭제할 필요가 없다는 점에 주의하십시오. 작동 불능 뷰와 동일한 *view-name*을 사용하여 CREATE VIEW문을 발행하면 작동 불능 뷰가 대체되고 CREATE VIEW문은 경고를 리턴합니다(SQLSTATE 01595).

작동 불능 뷰는 SYSCAT.VIEWS 카탈로그 뷰의 VALID 컬럼에 있는 X와 SYSCAT.TABLES 카탈로그 뷰의 STATUS 컬럼에 있는 X로 표시됩니다.

- *특권*: 뷰 정의자는 뷰 삭제 권한 뿐 아니라 뷰에 대한 SELECT 특권도 부여받습니다. 뷰 정의자는 fullselect에서 식별된 모든 기본 테이블, 뷰 또는 별칭에 대한 CONTROL 특권이 있거나 정의자가 다음 각 권한이 있는 경우에만 해당 뷰에 대한 CONTROL 특권을 확보하게 됩니다.
  - ACCESSCTRL 또는 SECADM
  - DATAACCESS
  - DBADM

  뷰가 읽기 전용이 아니고 뷰의 정의자에게 하위 오브젝트에 대해 해당하는 특권이 있는 경우, 뷰 정의자에게는 뷰에 대한 INSERT, UPDATE, 컬럼 레벨 UPDATE 또는 DELETE 특권이 부여됩니다.

  WITH ROW MOVEMENT를 사용하여 정의된 뷰의 경우, 정의자가 모든 기본 테이블 또는 뷰에 대한 INSERT 및 DELETE 특권뿐만 아니라 뷰의 모든 컬럼에 대한 UPDATE 특권을 가지고 있는 경우에만 정의자가 뷰에 대한 UPDATE 특권을 갖습니다.

  뷰 정의자는 뷰 작성시 파생되는 특권들이 존재할 경우 특권들을 획득할 수 있습니다. 뷰 정의자는 PUBLIC이 특권을 갖기 때문에 또는 직접 특권을 가져야 합니다. 페더레이티드 서버 별칭에서 뷰를 정의할 때에는 특권이 고려되지 않습니다. 그러나 별칭에 대한 뷰를 사용할 때에는 사용자의 권한 부여 ID가 별명이 데이터 소스에서 참조하는 테이블이나 뷰에 대해 유효한 선택 특권을 가지고 있어야 합니다. 그렇지 않으면 오류가 발생합니다. 뷰 정의자가 구성원인 그룹에서 보유하는 특권은 고려되지 않습니다.

  서브뷰가 작성될 때, 바로 위 슈퍼 뷰에 있는 SELECT 특권은 자동으로 서브뷰에 부여됩니다.
- ***Scope* 및 *REF* 컬럼**: 뷰 정의의 fullselect에서 참조 유형 컬럼을 선택할 때, 목표 유형과 필요한 범위를 고려하십시오.

  - 필요한 목표 유형과 범위가 하위 테이블이나 뷰와 동일한 경우, 컬럼을 선택하기만 해도 됩니다.
  - 범위를 변경해야 할 경우, WITH OPTIONS SCOPE절을 사용하여 필요한 범위 테이블이나 뷰를 정의하십시오.
  - 참조의 목표 유형을 변경해야 하는 경우, 컬럼을 우선 참조의 표시 유형으로 캐스트한 후 새로운 참조 유형으로 변환해야 합니다. 이런 경우의 범위는 WITH OPTIONS SCOPE절을 사용하거나, 캐스트 시 참조 유형으로 지정할 수 있습니다. 예를 들어, REF(TYP1) SCOPE TAB1으로 지정된 컬럼 Y를 선택한다고 가정하십시오. 이를 REF(VTYP1) SCOPE VIEW1으로 정의하고자 합니다. 선택 목록 항목은 다음과 같습니다.

    ```
    CAST(CAST(Y AS VARCHAR(16) FOR BIT DATA) AS REF(VTYP1) SCOPE VIEW1)
    ```

- ***식별 컬럼:*** 뷰의 컬럼은 뷰 정의의 fullselect에서 해당 컬럼의 요소가 테이블의 식별 컬럼 이름이거나 기본 테이블의 식별 컬럼 이름에 직접 또는 간접적으로 맵핑되는 뷰의 컬럼 이름인 경우에 식별 컬럼으로 간주됩니다.

  다른 모든 경우에는 뷰의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.
  - 뷰의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함되는 경우(즉, 두 번 이상 같은 컬럼 선택)
  - 뷰 정의에 하나의 조인이 포함되는 경우
  - 뷰 정의의 컬럼에 식별 컬럼을 참조하는 하나의 표현식이 포함되는 경우
  - 뷰 정의에 UNION이 포함되는 경우

  뷰 정의의 선택 목록에 직접 또는 간접적으로 기본 테이블의 식별 컬럼 이름이 포함되는 뷰에 삽입할 경우, INSERT문이 직접 기본 테이블의 식별 컬럼을 참조할 때와 같은 규칙이 적용됩니다.
- ***페더레이티드 뷰:*** 페더레이티드 뷰는 fullselect에 별칭에 대한 참조를 포함하는 뷰입니다. 그러한 별칭의 존재는 나중에 쿼리에서 뷰가 참조되는 경우 뷰에 사용되는 권한 부여 모델을 변경합니다.

  뷰가 작성될 때 뷰 정의자가 별칭 또는 하위 데이터 소스 테이블이나 뷰에 대한 액세스 권한을 가지는지 여부를 판별하기 위해 어떠한 특권 점검도 수행되지 않습니다. 뷰 정의자가 그러한 오브젝트에 대해 최소한 SELECT 특권을 가질 것을 요구하면서 페더레이티드 데이터베이스에 있는 테이블이나 뷰에 대한 참조 점검이 평상시대로 처리됩니다.

  페더레이티드 뷰가 다음에 쿼리에서 참조될 경우 별칭으로 인해 데이터 소스에 대한 쿼리가 작성되며, 쿼리를 발행한 권한 부여 ID(또는 이것이 맵핑되는 리모트 권한 부여 ID)는 데이터 소스 테이블이나 뷰 액세스에 필요한 특권을 가지고 있어야 합니

다. 페더레이티드 뷰를 참조하는 쿼리를 발행하는 권한 부여 ID는 페더레이티드 서버에 존재하는 테이블이나 뷰에 대해 어떠한 추가 특권도 필요로 하지 않습니다.

- ***ROW MOVEMENT, 트리거 및 제한조건:*** WITH ROW MOVEMENT절을 사용하여 정의된 뷰를 갱신하는 경우 트리거 및 제한조건 조작의 시퀀스는 다음과 같습니다.
  1. 이동하는 행을 포함하여 갱신되는 모든 행에 대해 BEFORE UPDATE 트리거가 활성화됩니다.
  2. 갱신 조작이 처리됩니다.
  3. 갱신된 모든 행에 대해 제한조건이 처리됩니다.
  4. 갱신 조작 이후 제한조건을 만족시키는 모든 행에 대해 AFTER UPDATE 트리거(행 레벨 및 명령문 레벨 모두)가 작성 순서대로 활성화됩니다. 이것은 UPDATE문이기 때문에 모든 UPDATE문 레벨의 트리거가 모든 기본 테이블에 대해 활성화됩니다.
  5. BEFORE DELETE 트리거는 갱신 조작 이후 제한조건을 만족시키지 못한 모든 행(이동될 행)에 대해 활성화됩니다.
  6. 삭제 조작이 처리됩니다.
  7. 삭제된 모든 행에 대해 제한조건이 처리됩니다.
  8. AFTER DELETE 트리거(행 레벨 및 명령문 레벨 모두)가 작성 순서대로 모든 삭제 행에 대해 활성화됩니다. 명령문 레벨 트리거는 삭제 조작 시 사용된 해당 테이블에 대해서만 활성화됩니다.
  9. BEFORE INSERT 트리거가 삽입되는 모든 행(이동할 행)에 대해 활성화됩니다. BEFORE INSERT 트리거에 사용되는 새 전이 테이블에는 사용자가 제공하는 입력 데이터가 있습니다.
  10. 삽입 조작이 처리됩니다.
  11. 삽입된 모든 행에 대해 제한조건이 처리됩니다.
  12. AFTER INSERT 트리거(행 레벨 및 명령문 레벨 모두)가 작성 순서대로 모든 삽입 행에 대해 활성화됩니다. 명령문 레벨 트리거는 삽입 조작 시 사용되는 해당 테이블에 대해서만 활성화됩니다.
- ***중첩된 UNION ALL 뷰:*** UNION ALL을 사용하여 정의되고 직간접적으로, UNION ALL을 사용하여 정의된 뷰를 기반으로 하는 뷰의 경우 WITH ROW MOVEMENT 절을 사용하여 정의된 뷰가 있으면 갱신될 수 없습니다(SQLSTATE 429BK).
- ***내재적으로 숨겨진 컬럼 고려사항:*** fullselect의 결과 테이블이 내재적으로 숨김으로 정의된 기본 테이블 컬럼을 포함할 수 있습니다. 내재적으로 숨겨진 컬럼이 뷰 정의의 fullselect를 명시적으로 참조하면 이런 상황이 발생할 수 있습니다. 그러나 해당되는 뷰의 컬럼은 내재적으로 숨겨진 속성을 상속하지 않습니다. 뷰의 컬럼은 숨김으로 정의될 수 없습니다.
- ***호환성:*** 이전 버전의 DB2 데이터베이스와의 호환성을 위해,

  – FEDERATED 키워드는 CREATE와 VIEW 키워드 사이에 지정될 수 있습니다. 그러나 페더레이티드 오브젝트가 뷰 정의에서 사용되었다면 경고가 더 이상 리턴되지 않기 때문에 FEDERATED 키워드는 무시됩니다.

- ***Subselect:*** *isolation-clause*가 *fullselect*에 지정될 수 없습니다(SQLSTATE 42601).

예:

예 1: 문자 'MA'로 시작되는 프로젝트 번호(PROJNO)가 있는 행만 포함하는 PROJECT 테이블에 뷰 MA_PROJ를 작성하십시오.

```
CREATE VIEW MA_PROJ  AS SELECT *
   FROM PROJECT
   WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 2: 예 1과 같은 뷰에서 프로젝트 번호(PROJNO), 프로젝트 이름(PROJNAME) 및 프로젝트 담당 사원(RESPEMP)에 대한 컬럼만 선택하십시오.

```
CREATE VIEW MA_PROJ
  AS SELECTPROJNO, PROJNAME, RESPEMP
   FROM PROJECT
  WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 3: 예 2와 같은 뷰에서 뷰에서 프로젝트 IN_CHARGE를 담당하는 직원에 대한 컬럼을 호출하십시오.

```
CREATE VIEW MA_PROJ
  (PROJNO, PROJNAME, IN_CHARGE)
  AS SELECTPROJNO, PROJNAME, RESPEMP
   FROM PROJECT
  WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

주: 컬럼 이름 중 하나만 변경되는 경우에도, 뷰에 있는 세 컬럼 모두의 이름이 괄호안에 나열되어야 하며 그 다음에는 MA_PROJ가 와야 합니다.

예 4: 프로젝트를 담당하는 직원(RESPEMP)의 성(LASTNAME)과 함께 PROJECT 테이블의 처음 네 컬럼(PROJNO, PROJNAME, DEPTNO, RESPEMP)을 포함하는 뷰 PRJ_LEADER를 작성하십시오. EMPLOYEE의 EMPNO를 PROJECT의 RESPEMP와 대조하여 EMPLOYEE 테이블로부터 이름을 확보하십시오.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
  FROM PROJECT, EMPLOYEE
  WHERE RESPEMP = EMPNO
```

예 5: 예 4와 같은 뷰에서 PROJNO, PROJNAME, DEPTNO, RESPEMP 및 LASTNAME 컬럼 외에 책임자의 총 급여(SALARY + BONUS + COMM)를 표시하십시오. 또한 중간 직원(PRSTAFF)이 둘 이상인 프로젝트만 선택하십시오.

```
 CREATE VIEW PRJ_LEADER
(PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY )
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
  FROM PROJECT, EMPLOYEE
  WHERE RESPEMP = EMPNO
  AND PRSTAFF > 1
```

fullselect에서 표현식 SALARY+BONUS+COMM의 이름을 TOTAL_PAYS로 지정하면 컬럼 이름 목록을 지정하지 않아도 됩니다.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP,
                LASTNAME, SALARY+BONUS+COMM AS TOTAL_PAY
    FROM PROJECT, EMPLOYEE
    WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

예 6: 제공된 테이블과 뷰는 다음과 같습니다.

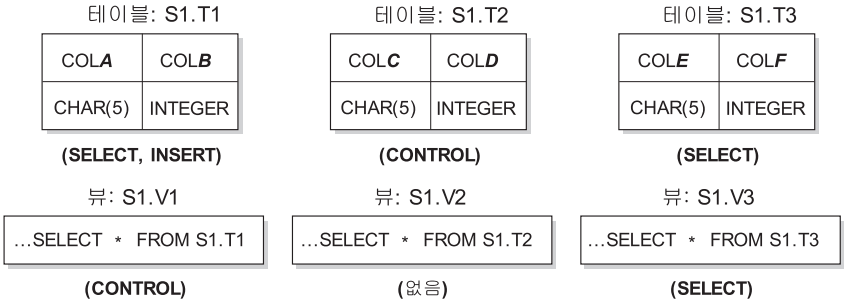


그림 1. 예 6에 대한 테이블과 뷰

사용자 ZORPIE(ACCESSCTRL, DATAACCESS 또는 DBADM 권한을 갖지 않음)에게는 각 오브젝트 아래에 괄호로 표시된 특권이 부여되었습니다.

1. ZORPIE는 다음을 사용하여 작성하는 뷰에 대해 CONTROL 특권을 갖게 됩니다.

   ```
   CREATE VIEW VA AS SELECT * FROM S1.V1
   ```

   이는 ZORPIE가 S1.V1에 대한 CONTROL을 가지고 있기 때문입니다. (S1.V1에 대한 CONTROL은 ACCESSCTRL 또는 SECADM 권한이 있는 사용자에 의해 ZORPIE에게 부여되어 있어야 합니다.) 하위 기본 테이블에 대해 ZORPIE가 갖고 있는 특권은 중요하지 않습니다.

2. ZORPIE는 뷰를 작성할 수 없습니다.

   ```
   CREATE VIEW VB AS SELECT * FROM S1.V2
   ```

   이는 S1.V2에 대해 CONTROL과 SELECT를 갖고 있지 않기 때문입니다. 하위 기본 테이블(S1.T2)에 대해 CONTROL을 갖고 있는 것은 중요하지 않습니다.

3. ZORPIE는 다음을 사용하여 작성하는 뷰에 대해 CONTROL 특권을 갖게 됩니다.

   ```
   CREATE VIEW VC (COLA, COLB, COLC, COLD)
     AS SELECT * FROM S1.V1, S1.T2
     WHERE COLA = COLC
   ```

이는 ZORPIE.VC의 fullselect는 뷰 S1.V1과 테이블 S1.T2를 참조하고 이 두 곳에 CONTROL을 갖고 있기 때문입니다. 뷰 VC는 읽기 전용이므로, ZORPIE는 INSERT, UPDATE 또는 DELETE 특권을 갖지 않습니다.

4. ZORPIE는 다음을 사용하여 작성하는 뷰에 대해 SELECT 특권을 갖게 됩니다.

```
CREATE VIEW VD (COLA,COLB, COLE, COLF)
  AS SELECT * FROM S1.V1, S1.V3
  WHERE COLA = COLE
```

이는 ZORPIE.VD의 fullselect가 SELECT 특권만 있는 뷰와 CONTROL 특권이 있는 뷰인 두 가지 뷰 S1.V1 및 S1.V3을 참조하기 때문입니다. ZORPIE.VD에 대한 두 특권 중 덜 중요한 SELECT가 부여됩니다.

5. ZORPIE는 GRANT 옵션과 더불어 INSERT, UPDATE 및 DELETE 특권과 아래의 뷰 정의에 있는 뷰 VE에 대해 SELECT 특권을 갖게 됩니다.

```
CREATE VIEW VE
   AS SELECT * FROM S1.V1
WHERE COLA > ANY
                      (SELECT COLE FROM S1.V3)
```

VE에서 ZORPIE의 특권은 S1.V1에서의 특권에 따라 1차적으로 결정됩니다. S1.V3는 서브쿼리에서만 참조되므로, VE 뷰를 작성하려면 S1.V3에 대해 SELECT 특권만 있으면 됩니다. 뷰 정의자는 뷰 정의시 참조되는 모든 오브젝트에 대해 CONTROL을 갖는 경우 뷰에 대해 CONTROL을 얻게 됩니다. ZORPIE는 S1.V3에 대해 CONTROL을 갖지 않으므로 VE에 대해 CONTROL을 갖지 않습니다.

CREATE WORK ACTION SET

CREATE WORK ACTION SET문은 작업 조치 세트와 작업 조치 세트 내의 작업 조치를 정의합니다.

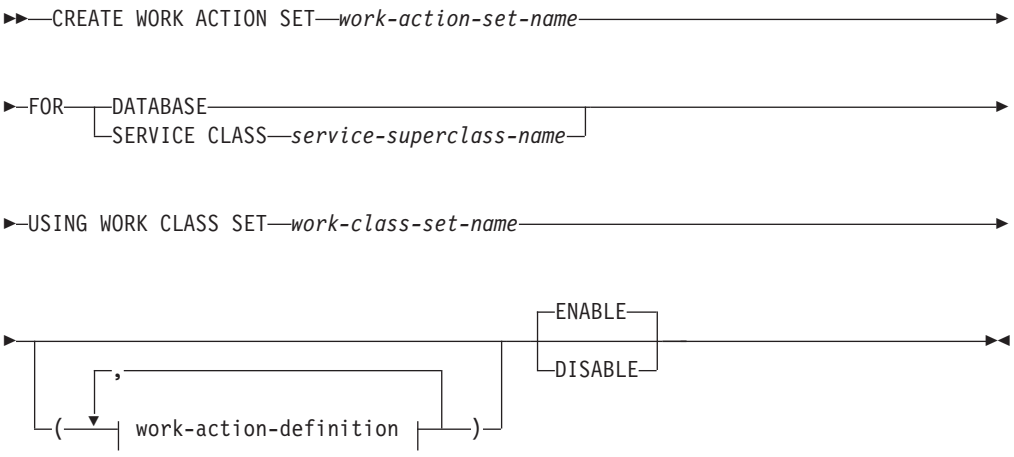
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

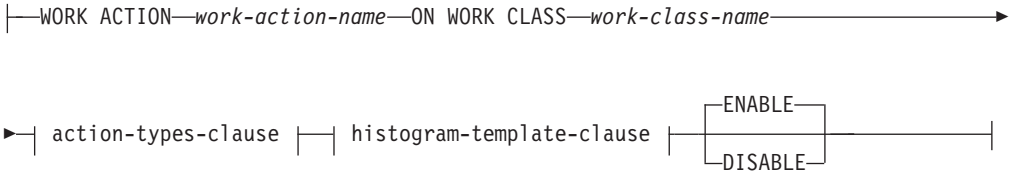
권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

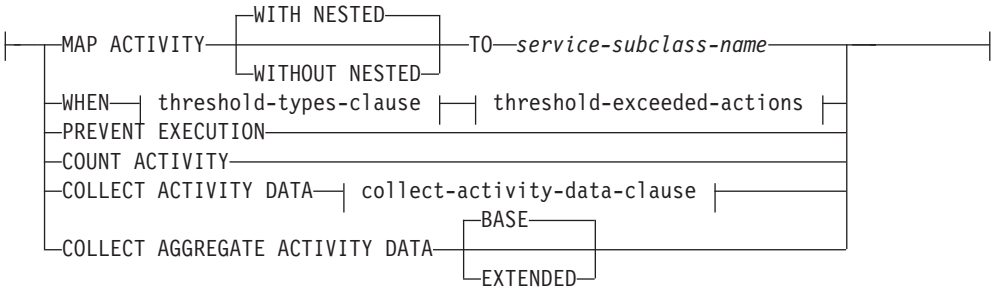
구문



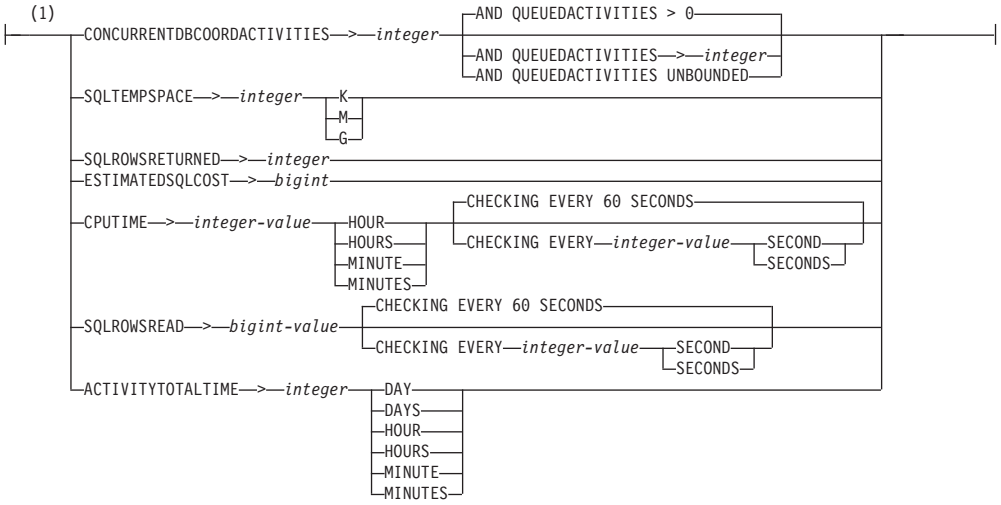
work-action-definition:



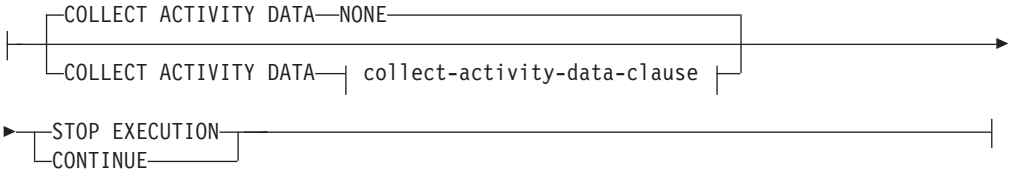
action-types-clause:



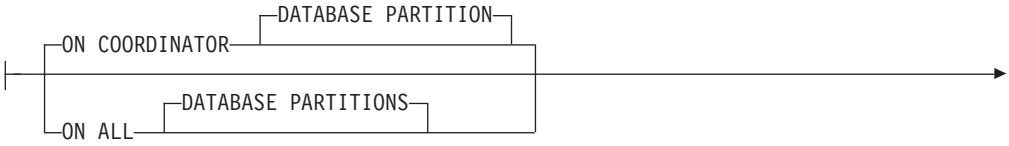
threshold-types-clause:

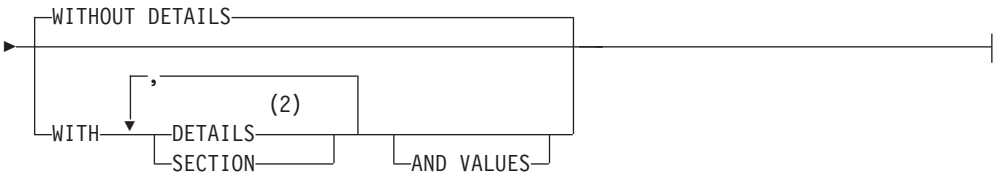


threshold-exceeded-actions:

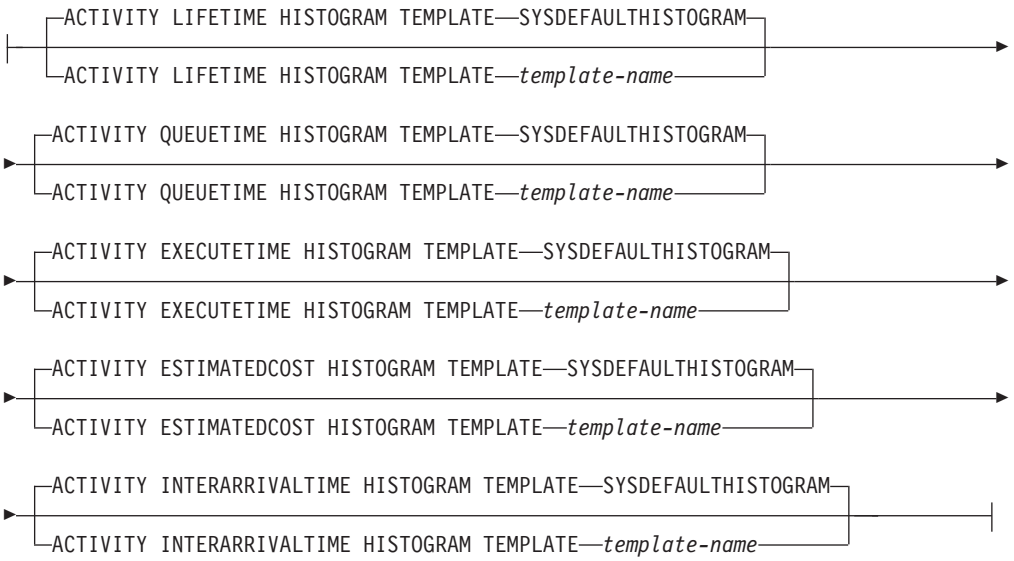


collect-activity-data-clause:





histogram-template-clause:



주:

1 한 번에 임계값 유형이 동일한 단 하나의 작업 조치만 단일 작업 클래스에 적용할 수 있습니다.

2 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

work-action-set-name

작업 조치 세트의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 분리)입니다. *work-action-set-name*은 현재 서버에 이미 존재하는 작업 클래스 세트를 식별하면 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

FOR

해당 작업 조치 세트의 조치를 적용할 데이터베이스 관리 프로그램 오브젝트를 지정합니다. 데이터베이스 관리 프로그램 오브젝트마다 단 하나의 작업 조치 세트를 정의할 수 있습니다(SQLSTATE 5U017).

DATABASE

해당 작업 조치 세트의 조치가 데이터베이스에 적용됩니다. DATABASE가 지정되면 MAP ACTIVITY 조치를 지정할 수 없습니다(SQLSTATE 5U034).

SERVICE CLASS *service-superclass-name*

해당 작업 조치 세트의 조치가 *service-superclass-name*에 적용됩니다. SERVICE CLASS가 지정되면 임계값 조치를 지정할 수 없습니다(SQLSTATE 5U034). *service-superclass-name*은 현재 서버에 존재해야 합니다(SQLSTATE 42704). *service-superclass-name*은 서비스 서브클래스이면 안되므로 다음 클래스 중 하나가 될 수 없습니다(SQLSTATE 5U032).

- 시스템 서비스 클래스(SYSDEFAULTSYSTEMCLASS)
- 유지보수 서비스 클래스(SYSDEFAULTMAINTENANCECLASS)
- 디폴트 사용자 서비스 클래스(SYSDEFAULTUSERCLASS)

USING WORK CLASS SET *work-class-set-name*

조치를 수행할 데이터베이스 활동을 분류할 작업 클래스를 포함하는 작업 클래스 세트를 지정합니다. *work-class-set-name*은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

work-action-definition

작업 조치의 정의를 지정합니다.

WORK ACTION *work-action-name*

작업 조치의 이름을 지정합니다. *work-action-name*은 현재 서버에서 해당 작업 조치 세트 아래에 이미 존재하는 작업 클래스 세트를 식별하면 안됩니다(SQLSTATE 42710). *work-action-name*은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

ON WORK CLASS *work-class-name*

작업 조치를 적용할 데이터베이스 활동을 식별하는 작업 클래스를 지정합니다. *work-class-name*은 현재 서버에서 *work-class-set-name*에 존재해야 합니다(SQLSTATE 42704).

MAP ACTIVITY

활동을 맵핑하는 작업 조치를 지정합니다. 이 조치는 작업 조치 세트가 정의된 오브젝트가 서비스 수퍼 클래스인 경우에만 지정할 수 있습니다(SQLSTATE 5U034).

WITH NESTED 또는 WITHOUT NESTED

해당 활동 아래에 중첩되는 활동이 서비스 서브클래스에 맵핑되는지 여부를 지정합니다. 디폴트는 WITH NESTED입니다.

WITH NESTED

작업 클래스 아래에 분류되어 있는 중첩 레벨 0의 모든 데이터베이스 활동과 이 활동 아래에 중첩된 모든 데이터베이스 활동은 서비스 서브

클래스에 맵핑됩니다. 즉, 중첩 레벨이 0보다 큰 활동은 중첩 레벨이 0인 활동과 같은 서비스 클래스 아래에서 실행됩니다.

WITHOUT NESTED
작업 클래스 아래에 분류되어 있는 중첩 레벨 0의 데이터베이스 활동만 서비스 서브클래스에 맵핑됩니다. 해당 활동 아래에 중첩되는 데이터베이스 활동은 해당 활동 유형에 따라 처리됩니다.

TO *service-subclass-name*
활동이 맵핑될 서비스 서브클래스를 지정합니다. *service-subclass-name*은 현재 서버에서 *service-superclass-name*에 이미 존재해야 합니다(SQLSTATE 42704). *service-subclass-name*은 디폴트 서비스 서브클래스 SYSDEFAULTSUBCLASS가 될 수 없습니다(SQLSTATE 5U018).

WHEN
작업 조치가 정의된 작업 클래스와 연관되는 데이터베이스 활동에 적용될 임계값을 지정합니다. 임계값은 작업 조치 세트가 정의된 데이터베이스 관리 프로그램 오브젝트가 데이터베이스인 경우에만 지정할 수 있습니다(SQLSTATE 5U034). 관리 SQL 루틴에서 생성된 데이터베이스 활동이나 데이터베이스 관리 프로그램에서 시작된 내부 데이터베이스 활동에는 이 임계값이 전혀 적용되지 않습니다.

threshold-types-clause
유효한 임계값 유형에 대한 설명은 『CREATE THRESHOLD』문을 참조하십시오.

threshold-exceeded-actions
유효한 임계값 초과 조치에 대한 설명은 『CREATE THRESHOLD』문을 참조하십시오.

PREVENT EXECUTION
작업 조치가 정의된 작업 클래스와 연관되는 데이터베이스 활동 중 어떤 것도 실행할 수 없음을 지정합니다(SQLSTATE 5U033).

COUNT ACTIVITY
작업 조치가 정의된 작업 클래스와 연관되는 모든 데이터베이스 활동이 실행되고 활동이 실행될 때마다 작업 클래스의 카운터가 증가함을 지정합니다.

COLLECT ACTIVITY DATA
작업 조치가 정의된 작업 클래스와 연관되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 디폴트는 COLLECT ACTIVITY DATA WITHOUT DETAILS입니다.

collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION
활동 코디네이터의 데이터베이스 파티션에서만 수집될 활동 데이터를 지정합니다.

ON ALL DATABASE PARTITIONS
활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지정합니다. 예측적 임계값의 경우, 초과된 임계값에 CONTINUE 조치도 지정하는 경우에만 모든 파티션에서 활동 정보가 수집됩니다. SECTION이 지정된 경우 활동 세부사항 및 섹션 환경도 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 반응적 임계값의 경우 ON ALL DATABASE PARTITIONS 절은 효과 및 활동, 활동 세부사항, 섹션 정보가 없으며 값은 항상 코디네이터 파티션에서만 수집됩니다.

WITHOUT DETAILS
서비스 클래스에서 실행되는 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내져야 함을 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA
작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. 디폴트는 COLLECT

AGGREGATE ACTIVITY DATA BASE입니다. 이 절은 데이터베이스에 적용되는 작업 조치 세트에 정의된 작업 조치에 지정할 수 없습니다.

BASE
작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 기본 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 실행 시간 막대 그래프

EXTENDED
작업 조치가 정의된 작업 클래스와 연관되는 활동에 대한 모든 집계 활동 데이터가 캡처되어 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 데이터 처리 언어(DML) 계산 비용 막대 그래프
- 활동 DML 도착 간 시간 막대 그래프

ENABLE 또는 DISABLE
데이터베이스 활동이 제출될 때 작업 조치가 고려되는지 여부를 지정합니다. 디폴트는 ENABLE입니다.

ENABLE
작업 조치가 사용 가능하고 데이터베이스 활동이 제출될 때 고려되도록 지정합니다.

DISABLE
작업 조치가 사용 불가능하고 데이터베이스 활동이 제출될 때 고려되지 않도록 지정합니다.

ENABLE 또는 DISABLE
데이터베이스 활동이 제출될 때 작업 조치 세트가 고려되는지 여부를 지정합니다. 디폴트는 ENABLE입니다.

ENABLE
작업 조치 세트가 사용 가능하고 데이터베이스 활동이 제출될 때 고려되도록 지정합니다.

DISABLE
작업 조치 세트가 사용 불가능하고 데이터베이스 활동이 제출될 때 고려되지 않도록 지정합니다.

histogram-template-clause
작업 조치가 지정된 작업 클래스와 연관되는 활동에 대한 집계 활동 데이터를 수집할 때 사용할 막대 그래프 템플리트를 지정합니다. 집계 활동 데이터는 작업 조치 유형이 COLLECT AGGREGATE ACTIVITY DATA인 경우에만 작업 클래스에 대해 수집됩니다.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*
특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동의 지속기간(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*
특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동이 큐에 대기된 시간 길이(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*
특정 간격 동안 작업 조치가 지정된 작업 클래스와 연관되는 DB2 활동이 실행 중인 시간 길이(마이크로초)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 활동이 실행되는 데이터베이스 파티션마다 이 막대 그래프에서 수집됩니다. 활동의 코디네이터 데이터베이스 파티션에서, 이 시간은 엔드-투-엔드 실행 시간입니다(즉, 수명이 큐에 대기하며 소비된 시간보다 적음). 코디네이터가 아닌 데이터베이스 파티션에서 이 시간은 파티션이 활동 대신 작업에 소비한 시간입니다. 제공된 활동 실행 중, DB2는 리모트 데이터베이스 파티션에 두 번 이상 작업을 제시할 수 있으며, 리모트 파티션은 매번 활동의 해당 어커런스에 대해 실행 시간을 수집합니다. 따라서, 실행 시간 막대 그래프의 계수는 데이터베이스 파티션에서 실행한 실제 고유 활동 수를 나타내지 못할 수도 있습니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*
작업 조치가 지정된 작업 클래스와 연관되는 DML 활동의 계산된 비용(timeron)에 대한 통계 정보를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는
SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*
작업 조치가 지정된 작업 클래스와 연관되는 활동에 대해, 하나의 DML 활동이 도착하고 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는
SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예:

예 1: 모든 데이터베이스 활동에 적용할 작업 조치 세트 DATABASE_ACTIONS를 작성하십시오. LARGE_QUERIES 작업 클래스 세트를 사용하고 다음 작업 조치를 정의하십시오. 작업 조치 ONE_CONCURRENT_QUERY에는 LARGE_ESTIMATED_COST 작업 클래스에 속하는 쿼리에 대해 시스템에서 한 번에 하나의 동시 쿼리가 실행될 수 있도록 하는 임계값 조치가 있습니다. 해당 임계값을 초과하면 데이터베이스 관리 프로그램은 활동을 큐에 넣으려고 하지만 한 번에 두 개 이상의 데이터베이스 활동을 큐에 넣을 수 없습니다. 큐 임계값을 초과하면 데이터베이스 활동이 실행되지 않습니다. 작업 조치 TWO_CONCURRENT_QUERY에는 LARGE_CARDINALITY 작업 클래스에 속하는 쿼리에 대해 두 개의 동시 쿼리가 동시에 실행되도록 허용하고 세 개 이상의 쿼리는 큐에 대기할 수 없는 임계값 조치가 있습니다. 세 개 이상의 쿼리가 큐에 대기되어 있으면 데이터베이스 활동은 계속 쿼리를 큐에 넣고 활동 이벤트 모니터(활성 상태인 경우)에서 데이터베이스 활동 데이터를 수집합니다.

```
CREATE WORK ACTION SET DATABASE_ACTIONS
     FOR DATABASE USING WORK CLASS SET LARGE_QUERIES
       (WORK ACTION ONE_CONCURRENT_QUERY ON WORK CLASS LARGE_ESTIMATED_COST
        WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 1
          STOP EXECUTION,
        WORK ACTION TWO_CONCURRENT_QUERIES ON WORK CLASS LARGE_CARDINALITY
        WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 2
          COLLECT ACTIVITY DATA CONTINUE)
```

예 2: 서비스 수퍼 클래스 ADMIN_APPS 아래에서 실행되는 데이터베이스 활동에 적용될 하나의 작업 조치 MAP_SELECTS가 있는 작업 조치 세트 ADMIN_APPS_ACTIONS를 작성하십시오. 작업 조치는 DML_SELECTS 작업 클래스 세트에 있는 서비스 서브클래스 SELECTS_SERVICE_CLASS에, SELECT_CLASS 작업 클래스 내에 속하는 모든 데이터베이스 활동을 맵핑하는 것입니다.

```
CREATE WORK ACTION SET ADMIN_APPS_ACTIONS
  FOR SERVICE CLASS ADMIN_APPS USING
    WORK CLASS SET DML_SELECTS
      (WORK ACTION MAP_SELECTS ON WORK CLASS SELECT_CLASS
       MAP ACTIVITY TO SELECTS_SERVICE_CLASS)
```

CREATE WORK CLASS SET

CREATE WORK CLASS SET문은 작업 클래스 세트를 정의합니다.

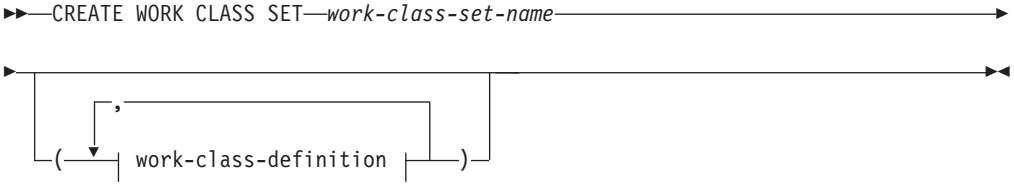
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

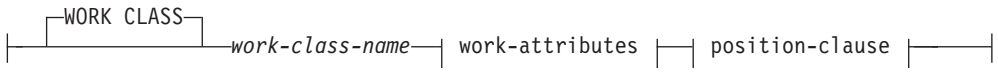
권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

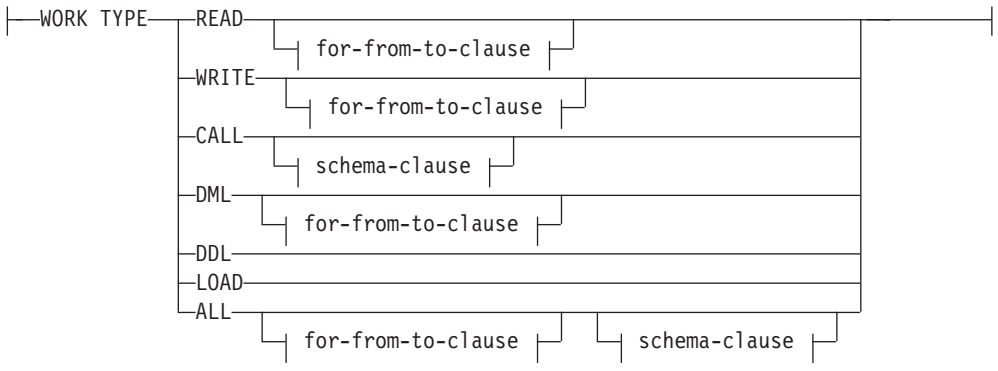
구문



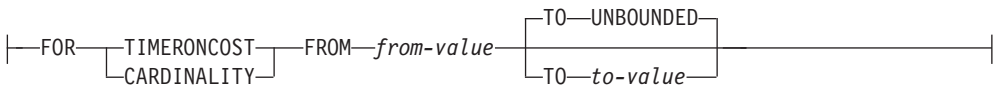
work-class-definition:



work-attributes:

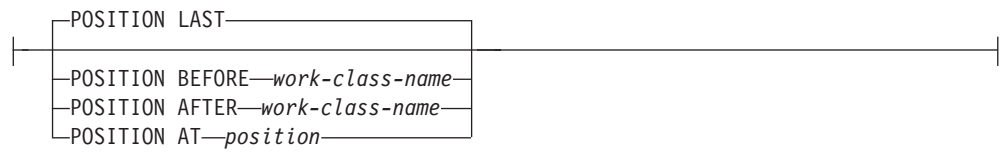


for-from-to-clause:



schema-clause:

position-clause:



설명

work-class-set-name

작업 클래스 세트의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *work-class-set-name*은 현재 서버에 이미 존재하는 작업 클래스 세트를 식별하면 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다(SQLSTATE 42939).

work-class-definition

작업 클래스의 정의를 지정합니다.

WORK CLASS *work-class-name*

작업 클래스의 이름을 지정합니다. *work-class-name*은 현재 서버에서 작업 클래스 세트 내에 이미 존재하는 작업 클래스 세트를 식별하면 안됩니다(SQLSTATE 42710). *work-class-name*은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

work-attributes

데이터베이스 활동의 속성은 해당 활동이 작업 클래스와 연관되는 경우 이 작업 클래스에 지정된 모든 속성과 일치해야 합니다.

WORK TYPE

데이터베이스 활동의 유형을 지정합니다.

READ

이 활동에는 다음 명령문이 포함됩니다.

- DELETE, INSERT, MERGE 또는 UPDATE 문을 포함하지 않는 모든 SELECT 또는 SELECT INTO문과 모든 VALUES INTO문
- 모든 XQuery문

WRITE

이 활동에는 다음 명령문이 포함됩니다.

- UPDATE
- DELETE

- INSERT
- MERGE
- DELETE, INSERT 또는 UPDATE 문을 포함하는 모든 SELECT 문과 모든 VALUES INTO문
- 모든 XQuery문

CALL

CALL문을 포함합니다. CALL문은 작업 유형이 CALL 또는 ALL인 작업 클래스에 대해 고려됩니다.

DML

READ 및 WRITE 아래에 나열되는 명령문을 포함합니다.

DDL

이 활동에는 다음 명령문이 포함됩니다.

- ALTER
- CREATE
- COMMENT
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- FLUSH PACKAGE CACHE
- GRANT
- REFRESH TABLE
- RENAME
- REVOKE
- SET INTEGRITY

LOAD

DB2 로드 조작.

ALL

위에 설명된 키워드 중 하나에 속하는 모든 인식되는 워크로드 관리 (WLM).

FOR

FROM *from-value* TO *to-value* 절에 지정되는 정보의 유형을 표시합니다. FOR 절은 다음 작업 유형에 대해서만 사용됩니다.

- READ
- WRITE
- DML
- ALL

TIMERONCOST

작업의 계산된 비용(timeron 단위)입니다. 이 값은 작업이 FROM *from-value* TO *to-value* 절에 지정된 범위 내에 속하는지 여부를 판별하기 위해 사용됩니다.

CARDINALITY

작업의 계산된 카디널리티(cardinality)입니다. 이 값은 작업이 FROM *from-value* TO *to-value* 절에 지정된 범위 내에 속하는지 여부를 판별하기 위해 사용됩니다.

FROM *from-value* **TO UNBOUNDED** 또는 **FROM** *from-value* **TO** *to-value*

데이터베이스 활동이 해당 작업 클래스의 일부가 되는 경우 이 활동이 속해야 하는 timeron 값(계산된 비용의 경우) 또는 카디널리티(cardinality)의 범위를 지정합니다. 범위는 *from-value* - *to-value*입니다. 작업 클래스에 대해 이 절을 지정하지 않으면 지정된 작업 유형에 속하는 모든 작업이 포함됩니다(즉, 디폴트는 FROM 0 TO UNBOUNDED임). 이 범위는 다음 작업 유형에 대해서만 사용됩니다.

- READ
- WRITE
- DML
- ALL

FROM *from-value* **TO UNBOUNDED**

*from-value*는 0 또는 양수 DOUBLE 값이어야 합니다(SQLSTATE 5U019). 범위에 상한은 없습니다.

FROM *from-value* **TO** *to-value*

*from-value*는 0 또는 양수 DOUBLE 값이어야 하고 *to-value*는 양수 DOUBLE 값이어야 합니다. *from-value*는 *to-value* 이하여야 합니다(SQLSTATE 5U019).

schema-clause

ROUTINES IN SCHEMA *schema-name*

CALL문이 호출될 프로시저의 스키마 이름을 지정합니다. 이 절은 작업 유형이 CALL 또는 ALL이고 데이터베이스 활동이 CALL문인 경우에만 사용됩니다. 값을 지정하지 않은 경우 모든 스키마가 포함됩니다.

position-clause

POSITION

작업 클래스 세트 내에서 해당 작업 클래스가 배치될 위치를 지정합니다. 이 위치에 따라 작업 클래스가 평가되는 순서가 결정됩니다. 런타임 시 작

업 클래스 지정을 수행하는 경우, 데이터베이스 관리 프로그램은 먼저 오브젝트(데이터베이스 또는 서비스 수퍼 클래스)와 연관되는 작업 클래스 세트를 판별합니다. 작업 클래스 세트에서 첫 번째 일치하는 작업 클래스가 선택됩니다. 이 키워드를 지정하지 않는 경우 작업 클래스는 마지막 위치에 배치됩니다.

LAST

작업 클래스 세트 내에서 순서화된 작업 클래스 목록에서 마지막에 작업 클래스를 배치함을 지정합니다. 이는 디폴트값입니다.

BEFORE *work-class-name*

목록에서 작업 클래스 *work-class-name* 앞에 작업 클래스를 배치할 것을 지정합니다. *work-class-name*은 현재 서버에 존재하는 작업 클래스 세트의 작업 클래스를 식별해야 합니다(SQLSTATE 42704).

AFTER *work-class-name*

목록에서 작업 클래스 *work-class-name* 다음에 작업 클래스를 배치할 것을 지정합니다. *work-class-name*은 현재 서버에 존재하는 작업 클래스 세트의 작업 클래스를 식별해야 합니다(SQLSTATE 42704).

AT *position*

작업 클래스 세트 내의 순서화된 작업 클래스 목록에서 작업 클래스가 배치될 절대 위치를 지정합니다. 이 값은 어떤 양수 값도 될 수 있습니다(SQLSTATE 42615). *position*이 기존 작업 클래스 수 + 1보다 클 경우 작업 클래스는 작업 클래스 세트 내의 마지막 위치에 배치됩니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)

- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예:

예 1: 계산된 비용이 9999보다 크고 계산된 카디낼리티가 1000보다 큰 모든 DML을 나타내는 작업 클래스 세트를 가지고 있는 작업 클래스 세트 LARGE_QUERIES를 작성하십시오.

```
CREATE WORK CLASS SET LARGE_QUERIES
  (WORK CLASS LARGE_ESTIMATED_COST WORK TYPE DML
   FOR TIMERONCOST FROM 9999 TO UNBOUNDED,
   WORK CLASS LARGE_CARDINALITY WORK TYPE DML
   FOR CARDINALITY FROM 1000 TO UNBOUNDED)
```

예 2: DELETE, INSERT, MERGE 또는 UPDATE 문을 포함하지 않는 모든 DML SELECT문을 나타내는 작업 클래스가 있는 작업 클래스 세트 DML_SELECTS를 작성하십시오.

```
CREATE WORK CLASS SET DML_SELECTS
  (WORK CLASS SELECT_CLASS WORK TYPE READ)
```

CREATE WORKLOAD

CREATE WORKLOAD문은 워크로드를 정의합니다.

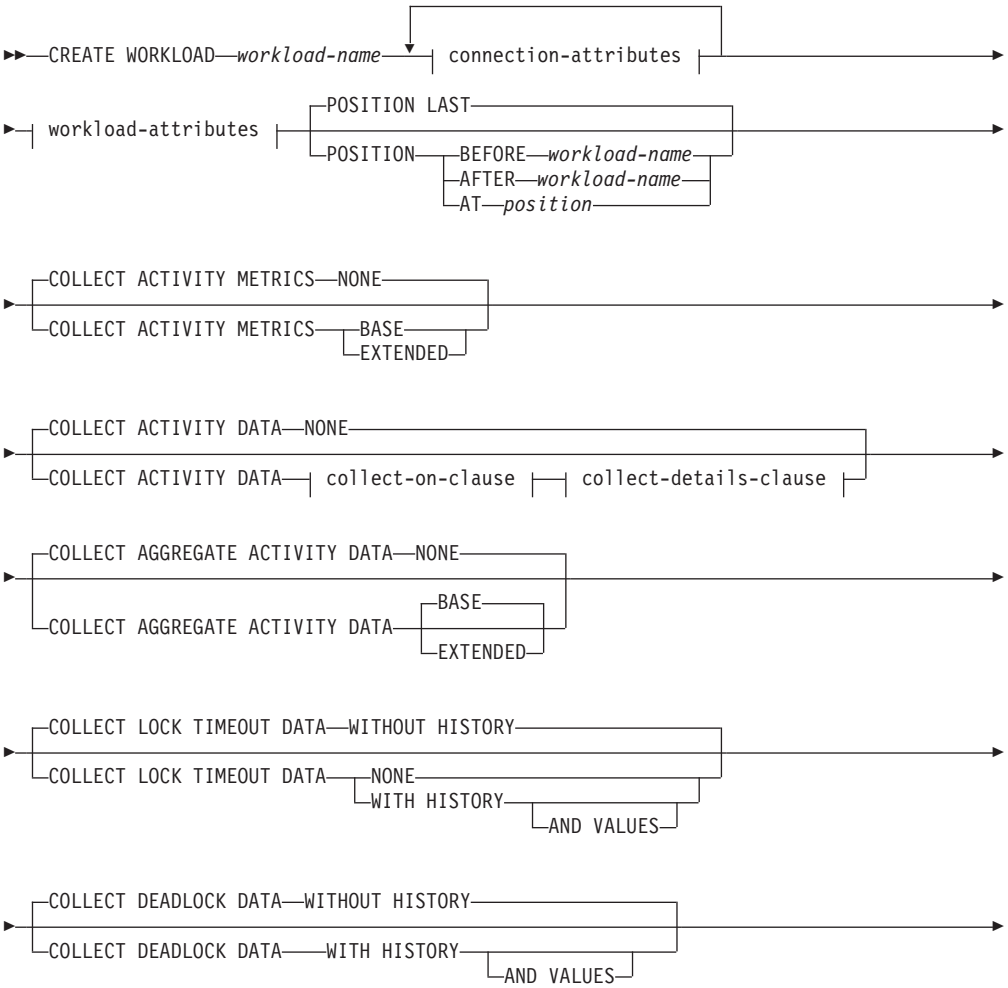
호출

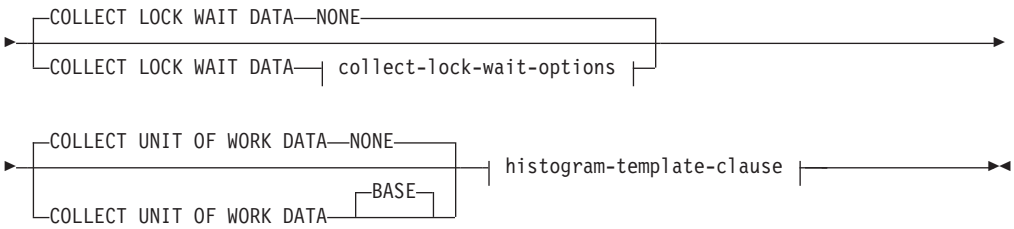
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

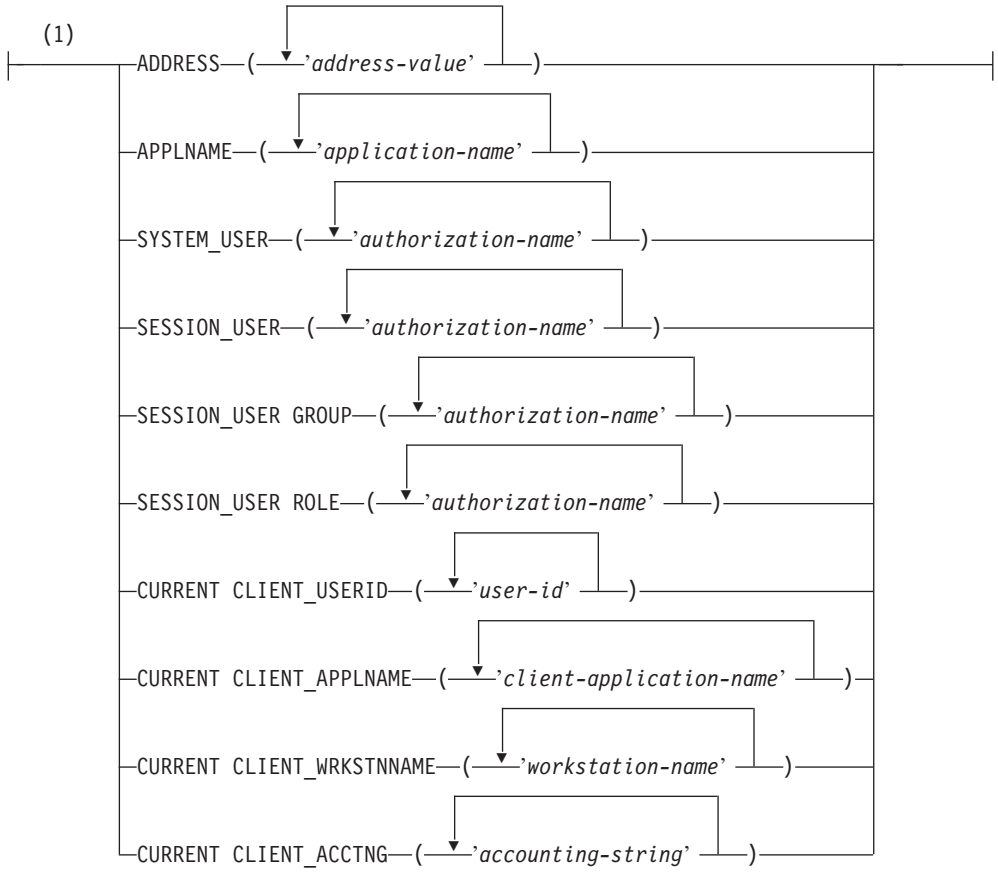
명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

구문

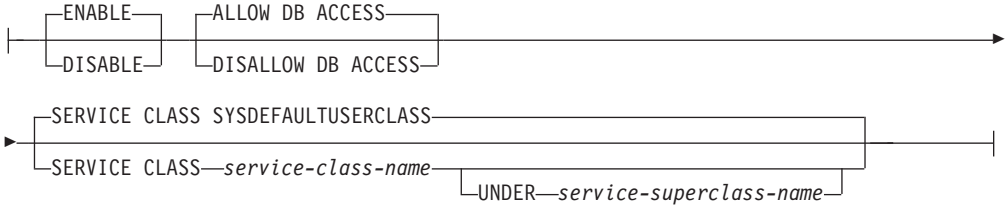




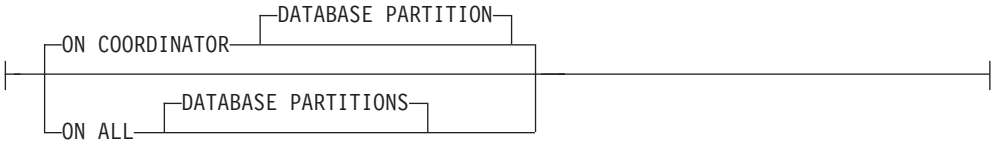
connection-attributes:



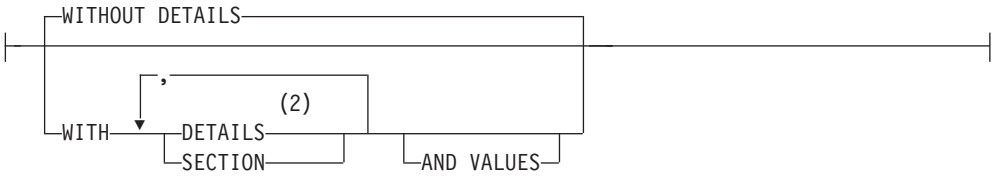
workload-attributes:



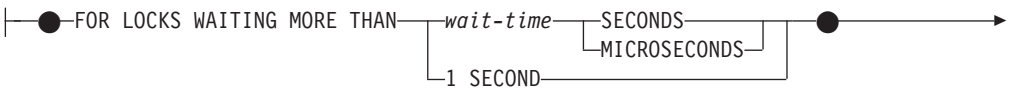
collect-on-clause:

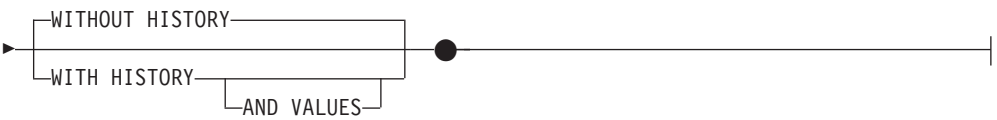


collect-details-clause:

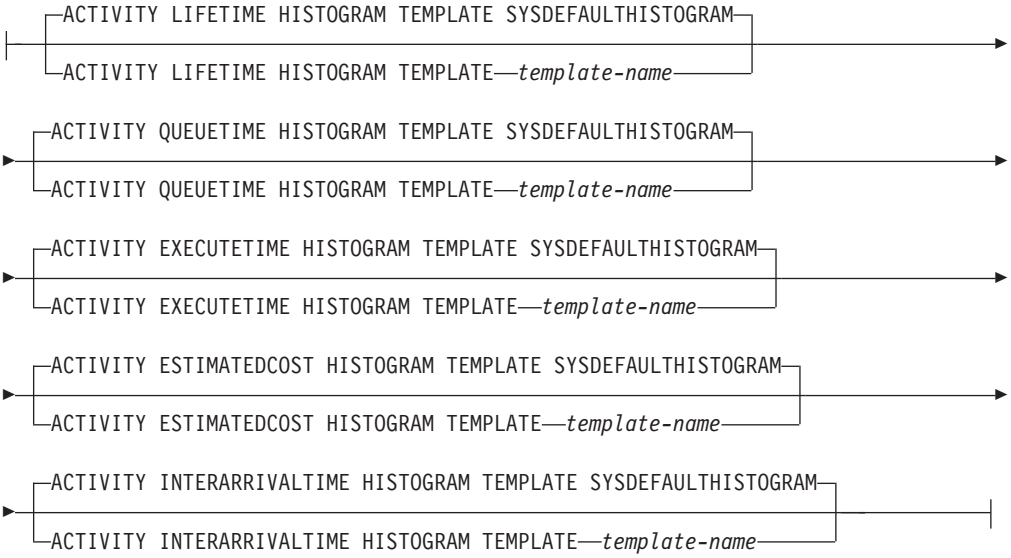


collect-lock-wait-options:





histogram-template-clause:



주:

1 각각의 연결 속성 절은 한 번만 지정할 수 있습니다.

2 DETAILS 키워드는 지정되는 최소값이며 그 뒤에는 쉼표로 구분되는 옵션이 있습니다.

설명

workload-name

워크로드 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. SQL ID(일반 또는 구분 ID)입니다. *workload-name*은 현재 서버에 이미 존재하는 워크로드를 식별하면 안됩니다(SQLSTATE 42710). 이름은 문자 'SYS'로 시작하면 안됩니다 (SQLSTATE 42939).

connection-attributes

연결의 속성은 연결 설정 시 연결이 이 워크로드와 연관되는 경우 이 워크로드 정의에 지정된 모든 속성과 일치해야 합니다. 워크로드 정의에서 연결 속성에 대해 값 목록이 지정되는 경우, 해당되는 연결 속성은 목록에 있는 값 중 하나 이상과 일치해야 합니다. 워크로드 정의에 연결 속성이 지정되지 않은 경우 연결은 해당되는 연결 속성에 대해 어떤 값도 가질 수 있습니다.

ADDRESS ('*address-value*', ...)

ADDRESS 연결 속성에 대해 하나 이상의 IPv4 주소, IPv6 주소 또는 보안 도메인 이름을 지정합니다. 주소 값은 목록에 두 번 이상 표시될 수 없습니다 (SQLSTATE 42713). 주소 값은 IPv4 주소, IPv6 주소 또는 보안 도메인 이름이어야 합니다.

IPv4 주소는 앞 공백을 포함할 수 없으며 점 10진 주소로 나타냅니다. IPv4 주소의 예로 `9.112.46.111`이 있습니다. 값 `localhost` 또는 해당되는 표현 `127.0.0.1`은 결과적으로 일치하지 않습니다. 대신 호스트의 실제 IPv4 주소를 지정해야 합니다. IPv6 주소는 앞 공백을 포함할 수 없으며 콜론 16진 주소로 나타냅니다. IPv6 주소의 예로 `2001:0DB8:0000:0000:0008:0800:200C:417A`가 있습니다. IPv4 맵핑 IPv6 주소(예: `::ffff:192.0.2.128`)는 결과적으로 일치하지 않습니다. 마찬가지로, `localhost` 또는 해당되는 IPv6 축약 표현 `::1`은 결과적으로 일치하지 않습니다. 도메인 이름은 도메인 이름 서버에 의해 결과 IPv4 또는 IPv6 주소가 판별되는 IP 주소로 변환됩니다. 도메인 이름의 예로 `corona.torolab.ibm.com`이 있습니다. 도메인 이름이 IP 주소로 변환될 때 이 변환의 결과는 하나 이상의 IP 주소로 설정될 수 있습니다. 이 경우, 연결이 시작되는 IP 주소가 도메인 이름이 변환된 IP 주소 중 하나와 일치하면 워크로드 오브젝트의 ADDRESS 속성과 일치함을 의미합니다.

워크로드 오브젝트를 작성할 때 정적 IP 주소 대신 ADDRESS 속성의 도메인 이름 값을 지정해야 합니다(특히 디바이스가 네트워크에 연결할 때마다 다른 IP 주소를 가질 수 있는 DHCP(Dynamic Host Configuration Protocol) 환경에서).

APPLNAME ('*application-name*', ...)

APPLNAME 연결 속성에 대한 하나 이상의 응용프로그램을 지정합니다. 응용프로그램 이름은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

*application-name*은 대소문자를 구분하며, 단일 별표 문자(*)를 포함하지 않는 경우 시스템 모니터 출력과 LIST APPLICATIONS 명령의 출력에서 『응용프로그램 이름』 필드에 표시되는 값과 같습니다. *application-name*이 단일 별표 문자(*)를 포함하는 경우, 값은 응용프로그램 이름 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 응용프로그램 이름에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

SYSTEM_USER ('*authorization-name***', ...)**
SYSTEM USER 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

SESSION_USER ('*authorization-name***', ...)**
SESSION USER 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

SESSION_USER GROUP ('*authorization-name***', ...)**
SESSION_USER GROUP 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

SESSION_USER ROLE ('*authorization-name***', ...)**
SESSION_USER ROLE 연결 속성에 대한 하나 이상의 권한 부여 ID를 지정합니다. 이 컨텍스트에서 세션 권한 부여 ID의 역할은 역할을 얻은 방법에 관계없이 세션 권한 부여 ID에 사용 가능한 모든 역할을 의미합니다. 권한 부여 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713).

CURRENT CLIENT_USERID ('*user-id***', ...)**
CURRENT CLIENT_USERID 연결 속성에 대한 하나 이상의 클라이언트 사용자 ID를 지정합니다. 클라이언트 사용자 ID는 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *user-id*에 단일 별표 문자(*)가 포함되는 경우, 값은 사용자 ID 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 사용자 ID에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

CURRENT CLIENT_APPLNAME ('*client-application-name***', ...)**
CURRENT CLIENT_APPLNAME 연결 속성에 대한 하나 이상의 응용프로그램을 지정합니다. 응용프로그램 이름은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *client-application-name*은 대소문자를 구분하며, 단일 별표 문자(*)를 포함하지 않는 경우 시스템 모니터 출력에서 『TP 모니터 클라이언트 응용프로그램 이름』 필드에 표시되는 값과 같습니다. *client-application-name*이 단일 별표 문자(*)를 포함하는 경우, 값은 응용프로그램 이름 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문

자로 된 문자열을 나타냅니다. 표현식에서 응용프로그램 이름에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

CURRENT CLIENT_WRKSTNNAME ('*workstation-name*', ...)

CURRENT CLIENT_WRKSTNNAME 연결 속성에 대한 하나 이상의 클라이언트 워크스테이션 이름을 지정합니다. 클라이언트 워크스테이션 이름은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *workstation-name*에 단일 별표 문자(*)가 포함되는 경우, 값은 워크스테이션 이름 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 워크스테이션 이름에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

CURRENT CLIENT_ACCTNG ('*accounting-string*', ...)

CURRENT CLIENT_ACCTNG 연결 속성에 대한 하나 이상의 클라이언트 어카운팅 문자열을 지정합니다. 클라이언트 어카운팅 문자열은 목록에 두 번 이상 표시될 수 없습니다(SQLSTATE 42713). *accounting-string*에 단일 별표 문자(*)가 포함되는 경우, 값은 어카운팅 문자열 세트를 나타내기 위한 표현식으로 사용됩니다. 여기서 별표(*)는 0개 이상의 문자로 된 문자열을 나타냅니다. 표현식에서 어카운팅 문자열에 별표 문자를 포함해야 하는 경우 두 개의 별표 문자 시퀀스(**)를 사용하십시오.

workload-attributes

워크로드의 속성을 지정합니다.

ENABLE 또는 DISABLE

워크로드가 선택될 때 워크로드가 고려되는지 여부를 지정합니다. 디폴트는 ENABLE입니다.

ENABLE

워크로드가 사용 가능하고 워크로드가 선택될 때 고려됨을 지정합니다.

DISABLE

워크로드가 사용 불가능하고 워크로드가 선택될 때 고려되지 않음을 지정합니다.

ALLOW DB ACCESS 또는 DISALLOW DB ACCESS

워크로드와 연관된 워크로드 어커런스가 데이터베이스에 액세스할 수 있는지 여부를 지정합니다. 디폴트는 ALLOW DB ACCESS입니다.

ALLOW DB ACCESS

워크로드와 연관된 워크로드 어커런스가 데이터베이스에 액세스할 수 있음을 지정합니다.

DISALLOW DB ACCESS

워크로드와 연관된 워크로드 어커런스가 데이터베이스에 액세스할 수 없음

을 지정합니다. 워크로드와 연관된 다음 작업 단위가 거부됨을 지정합니다 (SQLSTATE 5U020). 이미 실행 중인 워크로드 어커런스는 완료될 수 있습니다.

SERVICE CLASS *service-class-name*

워크로드와 연관된 요청이 *service-class-name* 서비스 클래스에서 실행됨을 지정합니다. *service-class-name*은 현재 서버에 존재하는 서비스 클래스를 식별해야 합니다(SQLSTATE 42704). *service-class-name*은 'SYSDEFAULTSUBCLASS', 'SYSDEFAULTSYSTEMCLASS' 또는 'SYSDEFAULTMAINTENANCECLASS'가 될 수 없습니다(SQLSTATE 5U032). 디폴트는 SYSDEFAULTUSERCLASS입니다.

UNDER *service-superclass-name*

이 절은 서비스 서브클래스를 지정할 때 사용됩니다.

*service-superclass-name*은 *service-class-name*의 서비스 수퍼 클래스를 식별합니다. *service-superclass-name*은 현재 서버에 존재하는 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704). *service-superclass-name*은 'SYSDEFAULTSYSTEMCLASS' 또는 'SYSDEFAULTMAINTENANCECLASS'가 될 수 없습니다(SQLSTATE 5U032).

POSITION

순서화된 워크로드 목록에서 해당 워크로드를 배치할 위치를 지정합니다. 런타임 시, 이 목록을 순서대로 검색하여 필수 연결 속성과 일치하는 첫 번째 워크로드를 찾습니다. 디폴트는 LAST입니다.

LAST

디폴트 워크로드 SYSDEFAULTUSERWORKLOAD 및 SYSDEFAULTADMWORKLOAD 이전에, 목록에서 워크로드가 마지막이 됨을 지정합니다.

BEFORE *relative-workload-name*

목록에서 워크로드 *relative-workload-name* 앞에 워크로드를 배치할 것을 지정합니다. *relative-workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). *relative-workload-name*이 'SYSDEFAULTUSERWORKLOAD' 또는 'SYSDEFAULTADMWORKLOAD'인 경우 BEFORE 옵션을 지정할 수 없습니다(SQLSTATE 42832).

AFTER *relative-workload-name*

목록에서 워크로드 *relative-workload-name* 다음에 워크로드를 배치할 것을 지정합니다. *relative-workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). *relative-workload-name*이

'SYSDEFAULTUSERWORKLOAD' 또는
'SYSDEFAULTADMWORKLOAD'인 경우 AFTER 옵션을 지정할 수 없습니다(SQLSTATE 42832).

AT *position*
목록에서 워크로드가 배치될 절대 위치를 지정합니다. 이 값은 어떤 양수 값도 될 수 있습니다(SQLSTATE 42615). *position*이 기존 워크로드 수 + 1보다 클 경우 워크로드는 마지막 위치, SYSDEFAULTUSERWORKLOAD 및 SYSDEFAULTADMWORKLOAD 직전에 위치됩니다.

COLLECT ACTIVITY METRICS
워크로드 어커런스에서 제출된 활동에 대해 모니터 메트릭을 수집해야 함을 지정합니다. 디폴트는 COLLECT ACTIVITY METRICS NONE입니다.

주: 유효한 활동 메트릭 콜렉션 설정은 활동을 제출하는 워크로드에서 COLLECT ACTIVITY METRICS절로 지정된 속성 및 **mon_act_metrics** 데이터베이스 구성 매개변수의 조합입니다. 워크로드 속성 또는 구성 매개변수에 NONE이 아닌 다른 값이 있는 경우 메트릭이 활동에 대해 수집됩니다.

NONE
워크로드 어커런스에서 제출된 활동에 대해 메트릭이 수집되지 않음을 지정합니다.

BASE
워크로드 어커런스에서 제출된 활동에 대해 기본 메트릭이 수집됨을 지정합니다.

EXTENDED
워크로드 어커런스에서 제출된 활동에 대해 확장 메트릭이 수집됨을 지정합니다.

COLLECT ACTIVITY DATA
해당 워크로드와 연관된 각 활동에 대한 데이터가 활동 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 디폴트는 COLLECT ACTIVITY DATA NONE입니다.

collect-on-clause
활동 데이터가 수집될 위치를 지정합니다. 디폴트는 ON COORDINATOR DATABASE PARTITION입니다.

ON COORDINATOR DATABASE PARTITION
활동 데이터가 활동 코디네이터의 데이터베이스 파티션에서만 수집되도록 지정합니다.

ON ALL DATABASE PARTITIONS
활동 데이터가 활동이 처리되는 모든 데이터베이스 파티션에서 수집됨을 지

정합니다. SECTION이 지정된 경우 활동 세부사항 및 섹션 환경은 활동이 처리되는 모든 데이터베이스 파티션에서 수집됩니다. SECTION이 지정되지 않은 경우 활동 세부사항은 코디네이터의 데이터베이스 파티션에서만 수집됩니다. 그렇지 않은 경우, 활동 값은 코디네이터의 데이터베이스 파티션에서만 수집됩니다.

NONE
해당 워크로드와 연관되는 활동마다 활동 데이터가 수집되지 않음을 지정합니다.

collect-details-clause
수집할 활동 데이터 유형을 지정합니다. 디폴트는 WITHOUT DETAILS입니다.

WITHOUT DETAILS
해당 워크로드와 연관된 각 활동에 대한 데이터가 활동 실행 완료 시 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 명령문, 컴파일 환경 및 섹션 환경 데이터에 대한 세부사항은 보내지 않습니다.

WITH

DETAILS
명령문과 컴파일 환경 데이터는 이 데이터를 가지고 있는 활동에 대한 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다. 섹션 환경 데이터는 보내지 않습니다.

SECTION
명령문, 컴파일 환경 및 섹션 환경 데이터는 이 데이터를 가지고 있는 해당 활동에 대한 활성 활동 이벤트 모니터로 전송되도록 지정합니다. SECTION이 지정된 경우 DETAILS도 지정해야 합니다.

AND VALUES
입력 데이터 값이 이 값을 가지고 있는 활동에 대해, 모든 활성 활동 이벤트 모니터로 보내지도록 지정합니다.

COLLECT AGGREGATE ACTIVITY DATA
해당 워크로드와 연관된 활동에 대한 집계 활동 데이터가 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 이 정보는 **wlm_collect_int** 데이터베이스 구성 매개변수에 의해 지정되는 간격으로 정기적으로 수집됩니다. COLLECT AGGREGATE ACTIVITY DATA를 지정하지 않은 경우 디폴트는 COLLECT AGGREGATE ACTIVITY DATA NONE입니다. COLLECT AGGREGATE ACTIVITY DATA를 지정한 경우 디폴트는 COLLECT AGGREGATE ACTIVITY DATA BASE입니다.

BASE

해당 워크로드와 연관된 활동에 대한 기본 집계 활동 데이터가 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 기본 집계 활동에는 다음이 포함됩니다.

- 활동 CPU 시간 상위 워터 마크(water mark)
- 활동 실행 시간 막대 그래프
- 활동 수명 막대 그래프
- 활동 큐 시간 막대 그래프
- 활동 행 읽기 상위 워터 마크(water mark)
- 추정된 활동 비용 상위 워터 마크(water mark)
- 리턴된 행 수 상위 워터 마크(water mark)
- 임시 테이블 스페이스 사용량 상위 워터 마크(water mark)

EXTENDED

해당 워크로드와 연관된 활동에 대한 모든 집계 활동 데이터가 통계 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. 여기에는 모든 기본 집계 활동 데이터와 다음 정보가 포함됩니다.

- 활동 데이터 처리 언어(DML) 계산 비용 막대 그래프
- 활동 DML 도착 간 시간 막대 그래프

NONE

어떤 집계 활동 데이터도 해당 워크로드에 대해 수집되지 않음을 지정합니다.

COLLECT LOCK TIMEOUT DATA

해당 워크로드 내에서 발생하는 잠금 시간종료 이벤트에 대한 데이터가 잠금 이벤트 발생 시 적용 가능한 이벤트 모니터로 보내지도록 지정합니다. 잠금 시간종료 데이터는 모든 파티션에서 수집됩니다. 디폴트는 COLLECT LOCK TIMEOUT DATA WITHOUT HISTORY입니다. 이 설정은 **mon_locktimeout** 데이터베이스 구성 매개변수 설정과 결합하여 작동합니다. 가장 상세한 출력을 생성하는 설정이 적용됩니다.

WITHOUT HISTORY

이 워크로드 내에서 발생하는 잠금 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 지나간 활동 실행기록 및 입력 값은 이벤트 모니터로 보내지 않습니다.

NONE

워크로드에 대한 잠금 시간종료 데이터가 어떤 파티션에서도 수집되지 않도록 지정합니다.

WITH HISTORY

해당 유형의 모든 잠금 이벤트에 대해 현재 작업 단위(UOW)에 있는 지나간 활동 실행기록을 수집할 것을 지정합니다. 활동 실행기록 버퍼는 최대 크기 한계가 사용된 후 래핑됩니다.

하나의 응용프로그램이 보존할 지나간 활동 수에 대한 디폴트 한계는 250입니다. 지나간 활동 수가 한계보다 큰 경우 최신 활동만 보고됩니다. 이 디폴트값은 다른 값을 지정하기 위한 레지스트리 변수 DB2_MAX_INACT_STMTS를 사용하여 대체할 수 있습니다. 지나간 활동 정보에 사용되는 시스템 모니터 힙 양을 늘리거나 줄이기 위해 한계로 다른 값을 선택할 수 있습니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대한 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 데이터 값에는 LOB 데이터, LONG VARCHAR 데이터, LONG VARGRAPHIC 데이터, 구조화된 유형 데이터 또는 XML 데이터가 포함되지 않습니다. REOPT ALWAYS 바인드 옵션을 사용하여 컴파일된 SQL문의 경우 이벤트 정보에는 REOPT 컴파일 또는 명령문 실행 데이터 값이 제공되지 않습니다.

COLLECT DEADLOCK DATA

이 워크로드 내에서 발생하는 교착 상태 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 교착 상태 데이터는 모든 파티션에서 수집됩니다. 디폴트는 COLLECT DEADLOCK DATA WITHOUT HISTORY입니다. 이 설정은 **mon_deadlock** 데이터베이스 구성 매개변수가 NONE으로 설정되지 않는 경우에만 적용됩니다.

WITHOUT HISTORY

이 워크로드 내에서 발생하는 잠금 이벤트에 대한 데이터가 잠금 이벤트 발생 시 모든 사용 중인 잠금 이벤트 모니터로 보내지도록 지정합니다. 지나간 활동 실행기록 및 입력 값은 이벤트 모니터로 보내지 않습니다.

WITH HISTORY

해당 유형의 모든 잠금 이벤트에 대해 현재 작업 단위(UOW)에 있는 지나간 활동 실행기록을 수집할 것을 지정합니다. 활동 실행기록 버퍼는 최대 크기 한계가 사용된 후 래핑됩니다.

하나의 응용프로그램이 보존할 지나간 활동 수에 대한 디폴트 한계는 250입니다. 지나간 활동 수가 한계보다 큰 경우 최신 활동만 보고됩니다. 이 디폴트값은 다른 값을 지정하기 위한 레지스트리 변수 DB2_MAX_INACT_STMTS를 사용하여 대체할 수 있습니다. 지나간 활동 정보에 사용되는 시스템 모니터 힙 양을 늘리거나 줄이기 위해 한계로 다른 값을 선택할 수 있습니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대한 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 데이터 값에는 LOB 데이터, LONG VARCHAR 데이터, LONG VARGRAPHIC 데이터, 구조화된 유형 데이터 또는 XML 데이터가 포함되지 않습니다. REOPT ALWAYS 바인드 옵션을 사용하여 컴파일된 SQL문의 경우 이벤트 정보에는 REOPT 컴파일 또는 명령문 실행 데이터 값이 제공되지 않습니다.

COLLECT LOCK WAIT DATA

해당 워크로드 내에서 발생하는 잠금 대기 이벤트에 대한 데이터가 *wait-time* 내에 잠금이 획득되지 않은 경우 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 디폴트는 디폴트 *wait-time* 값이 0 마이크로초인 COLLECT LOCK WAIT DATA NONE입니다. 이 설정은 **mon_lockwait** 및 **mon_lw_thresh** 데이터베이스 구성 매개변수와 결합하여 작동합니다. 가장 상세한 출력을 생성하는 설정이 적용됩니다.

NONE

워크로드에 대한 잠금 대기 이벤트가 어떤 파티션에서도 수집되지 않도록 지정합니다.

FOR LOCKS WAITING MORE THAN *wait-time*(**SECONDS | MICROSECONDS) | 1 SECOND**

해당 워크로드 내에서 발생하는 잠금 대기 이벤트에 대한 데이터가 *wait-time* 내에 잠금이 획득되지 않은 경우 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다.

이 값은 어떤 음수가 아닌 정수도 될 수 있습니다. 유효한 지속기간 키워드를 사용하여 *wait-time*으로 적절한 시간 단위를 지정하십시오. *wait-time* 매개변수의 유효한 최소값은 1000마이크로초입니다.

WITH HISTORY

해당 유형의 모든 잠금 이벤트에 대해 현재 작업 단위(UOW)에 있는 지나간 활동 실행기록을 수집할 것을 지정합니다. 활동 실행기록 버퍼는 최대 크기 한계가 사용된 후 래핑됩니다.

하나의 응용프로그램이 보존할 지나간 활동 수에 대한 디폴트 한계는 250입니다. 지나간 활동 수가 한계보다 큰 경우 최신 활동만 보고됩니다. 이 디폴트값은 다른 값을 지정하기 위한 레지스트리 변수
DB2_MAX_INACT_STMTS를 사용하여 대체할 수 있습니다. 지나간 활동 정보에 사용되는 시스템 모니터 힙 양을 늘리거나 줄이기 위해 한계로 다른 값을 선택할 수 있습니다.

AND VALUES

입력 데이터 값이 이 값을 가지고 있는 활동에 대한 모든 활성 잠금 이벤트 모니터로 보내지도록 지정합니다. 이 데이터 값에는 LOB 데이터, LONG

VARCHAR 데이터, LONG VARGRAPHIC 데이터, 구조화된 유형 데이터 또는 XML 데이터가 포함되지 않습니다. REOPT ALWAYS 바인드 옵션을 사용하여 컴파일된 SQL문의 경우 이벤트 정보에는 REOPT 컴파일 또는 명령문 실행 데이터 값이 제공되지 않습니다.

COLLECT UNIT OF WORK DATA

해당 워크로드와 연관된 각 트랜잭션에 대한 데이터가 작업 단위(UOW) 종료 시 작업 단위 이벤트 모니터(활성 상태인 경우)로 보내지도록 지정합니다. COLLECT UNIT OF WORK DATA가 지정되지 않을 때의 디폴트는 COLLECT UNIT OF WORK DATA NONE입니다. COLLECT UNIT OF WORK DATA가 지정될 때의 디폴트값은 COLLECT UNIT OF WORK DATA BASE입니다. **mon_uow_data** 데이터베이스 구성 매개변수가 BASE로 설정된 경우, COLLECT UNIT OF WORK DATA 매개변수보다 우선합니다. **mon_uow_data**의 NONE 값은 개별 워크로드의 COLLECT UNIT OF WORK DATA 매개변수를 사용함을 표시합니다.

BASE

이 워크로드와 연관된 트랜잭션에 대한 데이터의 기본 레벨이 작업 단위(UOW) 이벤트 모니터로 보내지도록 지정합니다.

작업 단위 이벤트로 보고되는 일부 정보는 시스템 레벨 요청 메트릭입니다. 이 메트릭의 콜렉션은 작업 단위 데이터 콜렉션과 독립적으로 제어됩니다. 요청 메트릭은 수퍼 클래스에서 COLLECT REQUEST METRICS절 또는 **mon_req_metrics** 데이터베이스 구성 매개변수를 사용하여 제어됩니다. 워크로드가 연관되는 서비스 수퍼 클래스나, 워크로드가 연관되는 서비스 수퍼 클래스의 서비스 수퍼 클래스는 작업 단위 이벤트에서 있을 요청 메트릭을 위해 요청 메트릭 콜렉션이 사용 가능해야 합니다. 요청 메트릭 콜렉션이 사용 가능하지 않으면 요청 메트릭 값은 영(0)이 됩니다.

NONE

해당 워크로드와 연관된 트랜잭션에 대한 어떤 작업 단위 데이터도 작업 단위 이벤트 모니터로 보내지지 않도록 지정합니다. 디폴트는 COLLECT UNIT OF WORK DATA NONE입니다.

histogram-template-clause

워크로드에서 실행 중인 활동에 대한 집계 활동 데이터를 수집할 때 사용할 막대 그래프 템플리트를 지정합니다.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

특정 간격 동안 워크로드에서 실행 중인 DB2 활동의 지속기간(마이크로초)에 대한 통계 데이터를 수집하는 데 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기된 시간과 실행 중 시간이 포함됩니다. 디폴

트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*
워크로드에서 실행 중인 DB2 활동이 특정 간격 동안 큐에서 대기하는 시간 길이(마이크로초)에 대한 통계 데이터를 수집하는 데 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*
워크로드에서 실행 중인 DB2 활동이 특정 간격 동안 실행되는 시간 길이(마이크로초)에 대한 통계 데이터를 수집하는 데 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 이 시간에는 큐에 대기하면서 소요된 시간은 포함되지 않습니다. 활동 실행 시간은 코디네이터 데이터베이스 파티션에서만 막대 그래프에서 수집됩니다. 시간에는 유휴 시간이 포함되지 않습니다. 유휴 시간은 어떤 작업도 완료되지 않은 경우에 동일한 활동에 속하는 요청의 실행 사이 시간입니다. 유휴 시간의 예로, 커서 열기가 종료되고 그 커서로부터 페치가 시작되는 시간 사이의 시간을 들 수 있습니다. 디폴트는
SYSDEFAULTHISTOGRAM입니다. 이 정보는 BASE 또는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 중첩 레벨 0의 활동만 막대 그래프에 포함되는 것으로 간주됩니다.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*
워크로드에서 실행 중인 DML 활동의 계산된 비용(timeron 단위)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다. 중첩 레벨 0의 활동만 막대 그래프에 포함되는 것으로 간주됩니다.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*
워크로드에 하나의 DML 활동이 도착하고 이 워크로드에 다음 DML 활동이 도착하는 시간 사이의 시간 길이(마이크로초)에 대한 통계 데이터를 수집하기 위해 사용되는 막대 그래프를 설명하는 템플리트를 지정합니다. 디폴트는 SYSDEFAULTHISTOGRAM입니다. 이 정보는 EXTENDED 옵션을 사용하여 COLLECT AGGREGATE ACTIVITY DATA 절을 지정한 경우에만 수집됩니다.

규칙

- 워크로드 관리(WLM) 독점 SQL문 다음에는 COMMIT 또는 ROLLBACK문이 있어야 합니다(SQLSTATE 5U021). WLM 독점 SQL문은 다음과 같습니다.
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET 또는 DROP(WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET 또는 DROP(WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- WLM 독점 SQL문은 XA 트랜잭션과 같은 전역 트랜잭션 내에서 실행할 수 없습니다(SQLSTATE 51041).

주

- 변경사항은 명령문을 실행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.
- 데이터베이스 연결이 설정되면, 데이터베이스 관리 프로그램은 POSITION 절에 지정된 연결 속성을 기초로 일치하는 워크로드를 찾습니다(스펙 순서대로). 일치하는 워크로드가 발견되면 데이터베이스 관리 프로그램은 현재 세션 사용자가 해당 워크로드에 대해 USAGE 특권을 가지고 있는지 확인합니다. 세션 사용자가 워크로드에 대해 USAGE 특권을 가지고 있지 않으면 데이터베이스 관리 프로그램은 다음으로 일치하는 워크로드를 찾습니다. 세션 사용자가 해당 워크로드에 대해 USAGE 특권을 가지고 있으면 연결은 워크로드와 연관됩니다. 일치하는 워크로드가 발견되지 않으면 연결은 디폴트 사용자 워크로드 SYSDEFAULTUSERWORKLOAD와 연관됩니다. 세션 사용자가 SYSDEFAULTUSERWORKLOAD에 대해 USAGE 특권을 가지고 있지 않으면 오류가 리턴됩니다(SQLSTATE 42501).
- 워크로드 연관은 데이터베이스 관리 프로그램이 다음 조건 중 하나를 발견하는 경우 새 작업 단위가 시작될 때마다 다시 평가됩니다.
  - 연결 속성이 변경되었습니다. 이는 다음 이벤트 중 하나가 발생한 경우에 발생할 수 있습니다.
    - 설정된 클라이언트 정보 API(sqleseti)가 호출되었고 이 API가 워크로드 정의에 포함된 연결 속성을 변경했습니다. 워크로드 재평가를 시작할 수 있도록 일반 사용자가 클라이언트 정보를 설정할 수 있어도, 세션 사용자가 워크로드에 대해 USAGE 특권을 가지고 있지 않으면 워크로드 자체 재맵핑이 발생할 수 없습니다.

  - SET SESSION AUTHORIZATION문이 호출되어 현재 세션 사용자를 변경했습니다.
  - 세션 사용자가 사용 가능한 역할이 변경되었습니다.
- 워크로드가 작성됩니다.
- 워크로드가 삭제됩니다.
- 워크로드가 변경됩니다.
- 워크로드에 대한 USAGE 특권이 사용자, 그룹 또는 역할에 부여되었습니다.
- 워크로드에 대한 USAGE 특권이 사용자, 그룹 또는 역할에서 취소되었습니다.

워크로드 재평가 결과 어떤 워크로드 재지정도 발생하지 않으면, 현재 워크로드 어커런스가 계속 실행됩니다. 즉, 새 워크로드 어커런스가 시작되지 않습니다.

- 활동이 계속 활성 상태이면 연결을 다른 워크로드에 재지정할 수 없습니다. 이와 같은 활동의 예로는, 로드 조작, 실행 프로시저 또는 여러 작업 단위 사이에 자원을 유지보수하는 명령문(예: WITH HOLD 커서 열기)이 있습니다. 현재 워크로드 어커런스는 실행 중인 모든 활동이 완료될 때까지 계속 실행됩니다. 워크로드 재지정은 다음 작업 단위가 시작될 때 발생합니다.
- 서비스 클래스가 워크로드에서 참조되면, 더 이상 어떤 워크로드에서도 참조되지 않을 때까지 삭제할 수 없습니다. 다음 조치 중 하나를 수행하여 워크로드에서 서비스 클래스 참조를 제거할 수 있습니다.
  - 워크로드를 변경하여 서비스 클래스 이름을 변경합니다.
  - 워크로드를 변경합니다.
- 역할이 워크로드에서 참조되면, 더 이상 어떤 워크로드에서도 참조되지 않을 때까지 삭제할 수 없습니다. 다음 조치 중 하나를 수행하여 워크로드에서 역할 참조를 제거할 수 있습니다.
  - 워크로드를 변경하여 역할을 변경합니다.
  - 워크로드를 변경합니다.

예:

예 1: 그룹 FINANCE에 속하는 세션 사용자가 제출하는 요청에 대해 CAMPAIGN 워크로드를 작성하십시오. 이 요청은 디폴트 사용자 서비스 클래스 SYSDEFAULTUSERCLASS에서 실행됩니다.

```
CREATE WORKLOAD CAMPAIGN
  SESSION_USER GROUP ('FINANCE')
```

예 2: HR 역할을 가지고 있는 세션 사용자에 대해 CURRENT CLIENT_APPLNAME 특수 레지스터가 SALARYSYS로 설정된 워크로드 PAYROLL을 작성하십시오. 이 워크로드와 연관된 작업 단위(UOW)는 서비스 수퍼 클래스 HRSC 아래에 있는 서비스

클래스 MEDIUMSC에서 실행됩니다. 런타임 시 워크로드가 선택된 경우 이 워크로드는 워크로드 CAMPAIGN이 평가되고 일치하지 않는 것으로 판별된 후에만 평가해야 합니다.

```
CREATE WORKLOAD PAYROLL
  SESSION_USER ROLE ('HR')
  CURRENT CLIENT_APPLNAME ('SALARYSYS') SERVICE CLASS MEDIUMSC
  UNDER HRSC POSITION AFTER CAMPAIGN
```

예 3: 워크로드 CAMPAIGN(예 1의) 어커런스가 현재 시스템에서 실행 중입니다. 그룹 FINANCE에 속하는 세션 사용자가 제출하는 요청에 대해 NEWCAMPAIGN 워크로드도 작성하십시오. 단, 요청은 DB2BP.EXE 응용프로그램을 통해서만 제출됩니다. 이 워크로드와 연관되는 요청은 서비스 클래스 MARKETINGSC에서 실행됩니다. NEWCAMPAIGN은 CAMPAIGN 이전에 평가해야 합니다.

```
CREATE WORKLOAD NEWCAMPAIGN
  SESSION_USER GROUP ('FINANCE')
  APPLNAME ('DB2BP.EXE') SERVICE CLASS MARKETINGSC
  POSITION BEFORE CAMPAIGN
```

CAMPAIGN의 실행 중인 워크로드 어커런스는 현재 작업 단위가 완료될 때까지 계속됩니다. 완료되면 워크로드 재평가가 발생하고 연결은 다시 워크로드 NEWCAMPAIGN에 맵핑될 수 있습니다.

예 4: 시스템 사용자 BOB 또는 MARY가 응용프로그램 appl1, appl2 또는 appl3을 통해 제출하는 요청에 대해 워크로드 REPORTS를 작성하십시오.

```
CREATE WORKLOAD REPORTS
  APPLNAME ('appl1', 'appl2', 'appl3')
  SYSTEM_USER ('BOB', 'MARY')
```

예 5: PAYROLL이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 워크로드 EMPLOYEES 내에서 발생하는 잠금 시간종료 이벤트의 명령문 실행기록으로 잠금 이벤트 레코드를 작성하십시오.

```
CREATE WORKLOAD EMPLOYEES
  APPLNAME ("app1", "app2")
  COLLECT LOCK TIMEOUT DATA WITH HISTORY
```

예 6: PAYROLL이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 모든 파티션에서 워크로드 FINANCE 내에 발생하는 교착 상태 및 잠금 시간종료 이벤트에 대한 잠금 이벤트 레코드를 작성하십시오.

```
CREATE WORKLOAD FINANCE
  APPLNAME ("app1", "app2")
  COLLECT DEADLOCK DATA
  COLLECT LOCK TIMEOUT DATA
```

예 7: PAYROLL이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, 워크로드 관리 프로그램 내에서 발생하는 교착 상태 이벤트의 명령문 실행기록 및 값으로 잠금 이벤트 레코드를 작성하십시오.

```
CREATE WORKLOAD MANAGERS
  APPLNAME ("app1", "app2")
  COLLECT DEADLOCK DATA WITH HISTORY AND VALUES
```

예 8: PAYROLL이라고 하는 잠금 이벤트 모니터가 존재하며 활성 상태인 것으로 가정하고, MANAGERS 워크로드 내에서 5000밀리초 동안 기다린 후 획득되는 잠금에 대한 명령문 실행기록으로 잠금 이벤트 레코드를 작성하십시오.

```
CREATE WORKLOAD MANAGERS
  APPLNAME ("app1", "app2")
  COLLECT LOCK WAIT DATA FOR LOCKS WAITING MORE THAN 5 SECONDS WITH HISTORY
```

예 9: 유사한 이름(*accrec01, accrec02 ... accrec15*)을 공유하고 서비스 클래스 ACCOUNTNGSC에 이를 지정하는 모든 어카운트 수신 가능 응용프로그램에 대해 워크로드 ACCRECS를 작성하십시오. 응용프로그램 이름은 와일드 카드(*)의 도움을 사용하는 APPLNAME 연결 속성을 통해 식별되며 개별적으로 지정하지 않아도 됩니다.

```
CREATE WORKLOAD ACCRECS
  SESSION_USER GROUP ('ACCOUNTING')
  APPLNAME ('accrec*')
  SERVICE CLASS ACCOUNTNGSC
```

예 10: 응용프로그램 appl1을 통해 제출되고 작업 단위 데이터가 수집되어 활성 상태의 작업 단위 이벤트 모니터로 보내도록 하는 요청에 대해 워크로드 CAMPAIGN을 작성하십시오.

```
CREATE WORKLOAD CAMPAIGN
  APPLNAME ('appl1')
  COLLECT UNIT OF WORK DATA BASE
```

예 11: 다음 명령문은 워크로드를 작성할 때 ADDRESS 연결 속성에 의해 지원되는 여러 주소 값 형식을 지정할 수 있는 방법을 나타냅니다.

- 보안 도메인 이름 지정:

```
CREATE WORKLOAD DOMAINWORKLOAD
  ADDRESS ('aviator.torolab.ibm.com')
```

- IPv4 주소 값 지정:

```
CREATE WORKLOAD IPWORKLOAD1
  ADDRESS ('9.26.53.111')
```

- IPv6 주소 값(long 형식) 지정:

```
CREATE WORKLOAD IPWORKLOAD2
  ADDRESS ('2002:91a:519:13:204:acff:fe57:6135')
```

- IPv6 주소 값(short 형식) 지정:

```
CREATE WORKLOAD IPWORKLOAD3
  ADDRESS ('fe80::202:55ff:fe9a:6eee')
```

CREATE WRAPPER

CREATE WRAPPER문은 페더레이티드 서버를 사용하여 랩퍼를 등록합니다. 랩퍼는 페더레이티드 서버가 특정 데이터 소스 유형과 상호 작용할 수 있는 메커니즘입니다.

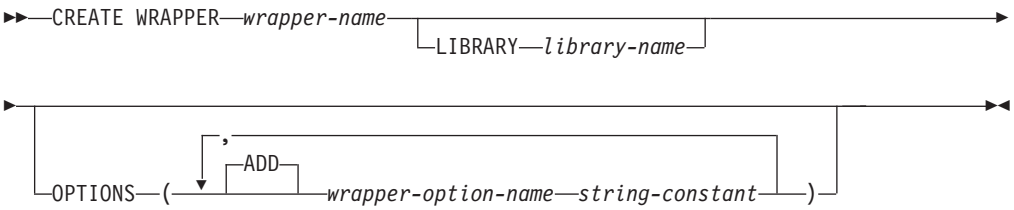
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 DBADM 권한이 포함되어야 합니다.

구문



설명

wrapper-name

랩퍼를 이름 지정하십시오. 다음과 같은 이름이 가능합니다.

- 사전 정의된 이름. 사전 정의된 이름이 지정되면 페더레이티드 서버가 자동으로 디폴트값을 *library-name*으로 지정합니다.
- 사용자 지정 이름. 사용자 지정 이름을 제공하면, 해당 랩퍼와 운영 체제에 사용되는 적절한 *library-name*을 지정해야 합니다.

LIBRARY *library-name*

랩퍼 라이브러리 모듈을 포함하는 파일의 이름을 지정합니다.

라이브러리 이름을 절대 경로 이름으로 지정하거나 경로없이 기본 이름만 지정할 수 있습니다. 기본 이름만 지정할 경우에는 라이브러리가 DB2 설치 경로의 `bin`(Windows) 서브디렉토리 또는 `lib`(UNIX)에 있어야 합니다. *library-name*은 작은따옴표로 묶어야 합니다.

LIBRARY 옵션은 사용자 지정 *wrapper-name*이 사용된 경우에만 필요합니다. 이 옵션은 사전 정의된 *wrapper-name*이 제공된 경우에는 사용할 수 없습니다.

OPTIONS (ADD *wrapper-option-name string-constant*, **...)**

랩퍼 옵션은 랩퍼를 구성하거나 DB2가 랩퍼를 사용하는 방법을 정의하기 위해 사

용됩니다. *wrapper-option-name*은 옵션 이름입니다. *string-constant*는 랩퍼 옵션에 대한 설정을 지정합니다. *string-constant*는 작은따옴표로 묶어야 합니다. 랩퍼 옵션 중 일부는 모든 랩퍼가, 일부 특정 랩퍼만 사용할 수 있습니다.

예:

예 1: 페더레이티드 서버에 NET8 랩퍼를 등록하여 Oracle 데이터 소스를 액세스하십시오. *NET8*은 Oracle 데이터 소스를 액세스하는 데 사용 할 수 있는 두 개의 랩퍼 중 하나의 사전 정의된 이름입니다.

```
CREATE WRAPPER NET8
```

예 2: Linux 운영 체제를 사용하는 DB2 페더레이티드 서버에 랩퍼를 등록하여 ODBC 데이터 소스를 액세스하십시오. 페더레이티드 데이터베이스에 등록되는 랩퍼에 *odbc* 이름을 지정하십시오. ODBC 드라이버 관리자를 포함하는 라이브러리의 전체 경로는 랩퍼 옵션 *MODULE '/usr/lib/odbc.so'*에 정의됩니다.

```
CREATE WRAPPER odbc OPTIONS (MODULE '/usr/lib/odbc.so')
```

예 3: Windows 운영 체제를 사용하는 DB2 페더레이티드 서버에 랩퍼를 등록하여 ODBC 데이터 소스에 액세스하십시오. ODBC 랩퍼의 라이브러리 이름인 *'db2rcodbc.dll'* 입니다.

```
CREATE WRAPPER odbc LIBRARY 'db2rcodbc.dll'
```

DECLARE CURSOR

DECLARE CURSOR문은 커서를 정의합니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베드될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다. 명령행 처리기를 사용하여 호출 시, 추가 옵션을 지정할 수 있습니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

권한 부여

이 용어는 『커서의 SELECT문』은 권한 부여 규칙을 지정하는 데 사용됩니다. 커서의 SELECT문은 다음 중 하나입니다.

- *statement-name*에 의해 식별된 준비된 select문
- 지정된 *select-statement*

명령문의 권한 부여 ID에 의해 보유된 특권은 *select-statement*를 실행하는 데 필요한 특권을 포함해야 합니다. "SQL 쿼리"의 권한 부여 섹션을 참조하십시오.

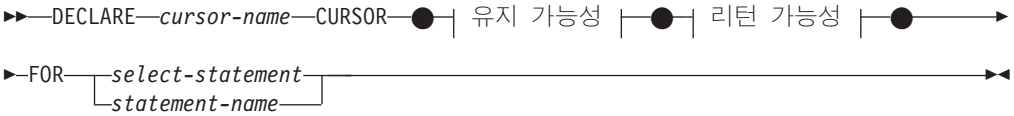
*statement-name*이 지정된 경우:

- 명령문의 권한 부여 ID는 런타임 권한 부여 ID입니다.
- Select문이 준비될 때 권한 부여를 점검합니다.
- 커서는 Select문이 올바로 준비되지 않으면 열리지 않습니다.

*select-statement*이 지정된 경우:

- GROUP 특권이 점검되지 않습니다.
- 명령문의 권한 부여 ID는 프로그램을 준비하는 동안 지정되는 권한 부여 ID입니다.

구문



유지 가능성:

리턴 가능성:

설명

cursor-name

소스 프로그램이 실행될 때 작성되는 커서 이름을 지정합니다. 이름은 소스 프로그램에서 선언된 다른 커서의 이름과 달라야 합니다. 커서는 사용하기 전에 열려 있어야 합니다.

WITHOUT HOLD 또는 WITH HOLD

커서가 커미트 조작의 결과로 닫히지 않도록 할지 여부를 지정합니다.

WITHOUT HOLD

커서가 커미트 조작의 결과로 닫히도록 합니다. 이는 디폴트값입니다.

WITH HOLD

여러 작업 단위에서 자원을 유지보수합니다. WITH HOLD 커서 속성의 영향은 다음과 같습니다.

- COMMIT로 끝나는 작업 단위의 경우
  - WITH HOLD로 정의된 열린 커서는 열린 채로 있습니다. 커서는 결과 테이블의 다음 논리 행 앞에 위치됩니다.

    DISCONNECT문이 WITH HOLD 커서와의 연결에 대해 COMMIT문 다음에 발행된 경우, 보유된 커서는 명시적으로 닫혀 있어야 합니다. 그렇지 않으면, 연결은 작업을 수행한 것(SQL문이 발행되지 않았어도 열린 WITH HELD 커서를 지님으로써)으로 간주되어 DISCONNECT문이 실패하게 됩니다.
  - 열려 있는 WITH HOLD 커서의 현재 커서 위치를 보호하는 잠금을 제외한 모든 잠금이 해제됩니다. 테이블에 대한 잠금과, 병렬 환경의 경우 커서가 현재 위치한 행에 대한 잠금은 그대로 유지됩니다. 패키지와 동적 SQL 섹션(있을 경우)에 대한 잠금은 그대로 유지됩니다.
  - 다음은 COMMIT 요청 바로 다음에 오는 WITH HOLD 정의되는 커서에 대한 유효한 조작입니다.
    - FETCH: 커서의 다음 행을 페치합니다.
    - CLOSE: 커서를 닫습니다.
  - UPDATE 및 DELETE CURRENT OF CURSOR는 같은 작업 단위 내에서 페치되는 행에 대해서만 유효합니다.

  - LOB 로케이터가 해제됩니다.
  - 행 설정은 다음 항목에 의해 수정됩니다.
    - 데이터 변경 명령문
    - 열린 WITH HOLD 커서에 임베드된 SQL 데이터를 수정하는 루틴

    이 커미트됩니다.
- ROLLBACK로 끝나는 작업 단위의 경우
  - 열린 커서는 모두 닫힙니다.
  - 작업 단위에서 획득된 모든 잠금은 해제됩니다.
  - LOB 로케이터가 해제됩니다.
- 특수 COMMIT의 경우
  - 패키지는 패키지를 바인딩하여 명시적 또는 내재적으로 재작성할 수 있는데 이는 패키지가 무효화된 이후 패키지를 처음으로 참조할 때 동적으로 재작성되었기 때문입니다. 패키지를 리바인드하는 동안 보유된 모든 커서가 닫힙니다. 이는 후속 실행에서 오류를 야기할 수도 있습니다.

WITHOUT RETURN 또는 WITH RETURN

커서의 결과 테이블이 프로시저에서 리턴되는 결과 세트로 사용되도록 할지 여부를 지정합니다.

WITHOUT RETURN

커서의 결과 테이블이 프로시저에서 리턴되는 결과 세트로 사용되지 않도록 지정합니다.

WITH RETURN

커서의 결과 테이블이 프로시저에서 리턴되는 결과 세트로 사용되도록 지정합니다. WITH RETURN은 DECLARE CURSOR문이 프로시저의 소스 코드와 함께 포함되는 경우에만 관련됩니다. 다른 경우에는 프리컴파일러가 이 절을 사용할 수 있으나 어떤 영향도 미치지 않습니다.

SQL 프로시저내에서 SQL 프로시저 종료시에도 여전히 열려 있는 WITH RETURN절을 사용하여 선언된 커서는 SQL 프로시저의 결과 세트를 정의합니다. SQL 프로시저 내의 다른 모든 열린 커서는 SQL 프로시저가 종료될 때 닫힙니다. 외부 프로시저(LANGUAGE SQL을 사용하여 정의되지 않은) 내에서는 모든 커서의 디폴트가 WITH RETURN TO CALLER입니다. 그러므로 프로시저를 종료할 때 열려 있는 모든 커서는 결과 세트로 간주됩니다. 프로시저에서 리턴된 커서는 화면 이동 커서로 선언될 수 없습니다.

TO CALLER

커서가 결과 세트를 호출자에게 리턴할 수 있도록 지정합니다. 예를 들어, 호출자가 다른 프로시저인 경우 결과 세트는 해당 프로시저로 리

턴됩니다. 호출자가 클라이언트 응용프로그램인 경우 결과 세트는 클라이언트 응용프로그램으로 리턴됩니다.

TO CLIENT

커서가 결과 세트를 클라이언트 응용프로그램으로 리턴할 수 있도록 지정합니다. 이 커서는 중간 중첩된 프로시저에게는 보이지 않습니다. 함수, 메소드나 트리거가 직간접적으로 프로시저를 호출한 경우에는 결과 세트가 클라이언트에게 리턴되지 않고 프로시저가 완료되면 커서가 닫힙니다.

SELECT

커서의 SELECT문을 식별합니다. *select-statement*은 매개변수 표시문자를 포함해서는 안 되지만, 호스트 변수에 대한 참조는 포함할 수 있습니다. 호스트 변수의 선언은 소스 프로그램에서 DECLARE CURSOR문 앞에 있어야 합니다.

statement-name

statement-name 커서의 SELECT문은 커서가 열릴 때 명령문 이름으로 식별되는 준비된 SELECT문입니다. *statement-name*은 소스 프로그램의 다른 DECLARE CURSOR문에 지정된 *statement-name*과 같으면 안됩니다.

prepared SELECT문에 대한 설명은 『PREPARE』를 참조하십시오.

주

- 다른 프로그램 또는 같은 프로그램 내의 다른 소스 파일로부터 호출되는 프로그램은 호출하는 프로그램이 열어 놓은 커서를 사용할 수 없습니다.
- SQL이 아닌 다른 LANGUAGE로 중첩되지 않은 프로시저는 WITH RETURN절 없이 DECLARE CURSOR가 지정되고 커서가 프로시저에서 열려 있는 경우 디폴트 동작으로 WITH RETURN TO CALLER를 가지게 됩니다. 이는 이전 버전의 프로시저로 하여금 리턴 결과 세트를 적용 가능한 클라이언트 응용프로그램으로 리턴할 수 있는 호환성을 제공합니다. 이러한 동작을 피하려면 해당 프로시저에서 열려 있는 모든 커서를 닫으십시오.
- 커서의 SELECT문에 CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP가 포함되어 있으면, 이 특수 레지스터에 대한 모든 참조사항은 각 FETCH에서 각각 같은 날짜 시간 값을 생성합니다. 이 값은 커서가 열릴 때 결정됩니다.
- 데이터의 효율적인 처리를 위해, 데이터베이스 관리 프로그램은 리모트 서버로부터 데이터를 검색할 때 읽기 전용 커서에 대해 데이터를 블록화할 수 있습니다. FOR UPDATE절을 사용하면 데이터베이스 관리 프로그램에서 커서가 갱신 가능 여부를 쉽게 결정할 수 있습니다. 갱신 가능성은 액세스 경로 선택을 결정할 경우에도 사용됩니다. 커서가 위치 지정된 UPDATE나 DELETE문에서 사용되지 않을 경우 FOR READ ONLY로 선언되어야 합니다.

- 열린 상태의 커서는 결과 테이블과 해당 테이블의 행에 상대적인 위치를 지시합니다. 테이블은 커서의 SELECT문에 의해 지정된 결과 테이블입니다.
- 커서는 다음 각 조건이 만족되는 경우 *삭제 가능*합니다.
  - 외부 fullselect의 각 FROM절이 OUTER절을 사용하지 않고 하나의 기본 테이블이나 삭제 가능 뷰를 식별하는 경우(중첩된 또는 공통 테이블 표현식이나 별칭은 식별할 수 없음)
  - 외부 fullselect에 VALUES절이 포함되지 않는 경우
  - 외부 fullselect에 GROUP BY절 또는 HAVING절이 포함되지 않는 경우
  - 외부 fullselect에 선택 목록 내의 컬럼 함수가 포함되지 않는 경우
  - 외부 fullselect에 UNION ALL 예외가 있는 SET 조작(UNION, EXCEPT 또는 INTERSECT)이 포함되지 않는 경우
  - 외부 fullselect의 선택 목록에 DISTINCT가 포함되지 않는 경우
  - 외부 fullselect에 ORDER BY절이 포함되어 있지 않고(ORDER BY절이 뷰에 중첩되어 있는 경우라도) FOR UPDATE절은 지정되어 있지 않은 경우
  - select-statement에 FOR READ ONLY절이 포함되지 않는 경우
  - 외부 fullselect의 FROM절에 *data-change-table-reference*가 포함되지 않은 경우
  - 다음 중 하나 이상이 만족되는 경우
    - FOR UPDATE절이 지정된 경우
    - STATICREADONLY 바인드 옵션이 YES가 아닌 경우 해당 커서는 정적으로 정의된 경우
    - LANGLEVEL 바인드 옵션이 MIA 또는 SQL92E인 경우

  커서와 연관된 외부 fullselect의 선택 목록에 있는 컬럼은 다음 각 조건이 만족되는 경우 *갱신 가능*합니다.
  - 커서가 삭제 가능한 경우
  - 컬럼이 기본 테이블의 컬럼을 해석하는 경우
  - LANGLEVEL 바인드 옵션이 MIA이고, SQL92E 또는 선택문에 FOR UPDATE 절이 포함되는 경우(해당 컬럼을 FOR UPDATE절에 명시적으로 또는 내재적으로 지정해야 함)

  커서가 삭제 불가능한 경우 *읽기 전용*입니다.

  커서는 다음 각 조건이 만족되는 경우 *앰비규어스*합니다.
  - 선택문이 동적으로 준비된 경우
  - 선택문에 FOR READ ONLY절 또는 FOR UPDATE절이 포함되지 않는 경우
  - LANGLEVEL 바인드 옵션이 SAA1인 경우

– 그렇지 않으면, 커서가 삭제 가능 커서의 조건을 만족하는 경우

앰비규어스 커서는 BLOCKING 바인드 옵션이 ALL인 경우 읽기 전용으로 간주되고, 그렇지 않으면 갱신 가능으로 간주됩니다.

- CLI를 사용하여 작성된 응용프로그램이 호출한 프로시저의 커서는 클라이언트 응용프로그램으로 직접 리턴된 결과 세트를 정의하는 데 사용될 수 있습니다. SQL 프로시저의 커서는 WITH RETURN절을 사용하여 정의된 경우에만 호출하는 SQL 프로시저로 리턴될 수 있습니다.
- WITH HOLD로 선언된 커서에서 직간접적으로 호출한 루틴에 선언된 커서는 WITH HOLD 옵션을 상속하지 않습니다. 따라서 루틴의 커서가 명시적으로 WITH HOLD로 정의되어 있지 않으면 응용프로그램의 COMMIT가 커서를 닫습니다.

다음 응용프로그램과 두 개의 UDF를 고려하십시오.

```
Application:

   DECLARE APPCUR CURSOR WITH HOLD FOR SELECT UDF1() ...
   OPEN APPCUR
   FETCH APPCUR ...
   COMMIT

UDF1:

   DECLARE UDF1CUR CURSOR FOR SELECT UDF2() ...
   OPEN UDF1CUR
   FETCH UDF1CUR ...

UDF2:

   DECLARE UDF2CUR CURSOR WITH HOLD FOR SELECT UDF2() ...
   OPEN UDF2CUR
   FETCH UDF2CUR ...
```

응용프로그램이 APPCUR 커서를 페치(fetch)한 후에는 세 커서가 모두 열려 있습니다. APPCUR 커서는 WITH HOLD로 선언되어 있기 때문에 응용프로그램이 COMMIT문을 발행할 때 APPCUR 커서는 계속 열려 있습니다. 그러나 UDF1에서는 UDF1CUR 커서가 WITH HOLD 옵션으로 정의되어 있지 않기 때문에 UDF1CUR 커서가 닫힙니다. UDF1CUR 커서가 닫히면 해당 Select문에 있는 모든 루틴 호출이 완료되고, 마지막 호출을 수신하도록 정의되어 있으면 마지막 호출을 수신합니다. UDF2가 완료되면 UDF2CUR가 닫힙니다.

예:

예1: DECLARE CURSOR문으로 커서 이름 C1을 SELECT의 결과와 연관시키십시오.

```
 EXEC SQL DECLARE C1 CURSOR FOR
SELECT DEPTNO, DEPTNAME, MGRNO
FROM DEPARTMENT
WHERE ADMRDEPT = 'A00';
```

예 2: EMPLOYEE 테이블을 변경하여 연봉을 기초로 주급을 계산한 WEEKLYPAY 컬럼을 추가한다고 가정하십시오. 커서를 선언하여 삽입되는 행에서 시스템 생성 컬럼 값을 검색하십시오.

```
 EXEC SQL DECLARE C2 CURSOR FOR
SELECT E.WEEKLYPAY
FROM NEW TABLE
  (INSERT INTO EMPLOYEE
   (EMPNO, FIRSTNME, MIDINIT, LASTNAME, EDLEVEL, SALARY)
   VALUES('000420', 'Peter', 'U', 'Bender', 16, 31842) AS E;
```

DECLARE GLOBAL TEMPORARY TABLE

DECLARE GLOBAL TEMPORARY TABLE문은 현재 세션의 임시 테이블을 정의합니다. 선언된 임시 테이블 설명은 시스템 카탈로그에 나타나지 않습니다. 임시 테이블은 영속적이 아니며 다른 세션과 공유할 수 없습니다. 같은 이름의 선언된 전역 임시 테이블을 정의하는 각 세션은 그 소유의 고유한 임시 테이블 설명을 가집니다. 세션이 종료되면 테이블의 행이 삭제되고 임시 테이블의 설명이 삭제됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

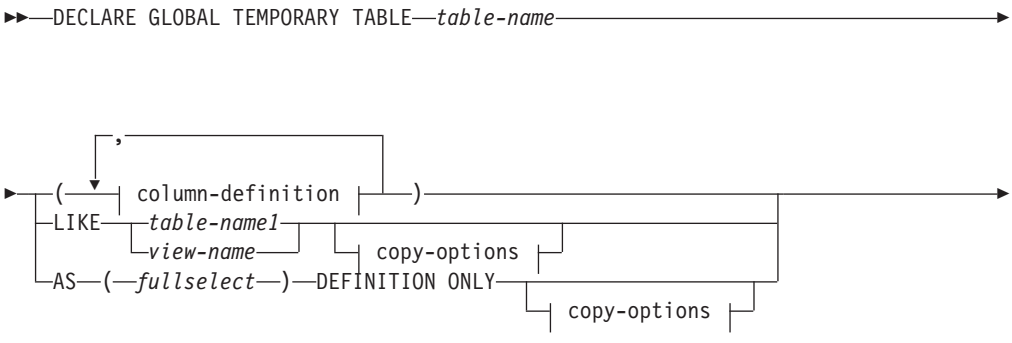
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

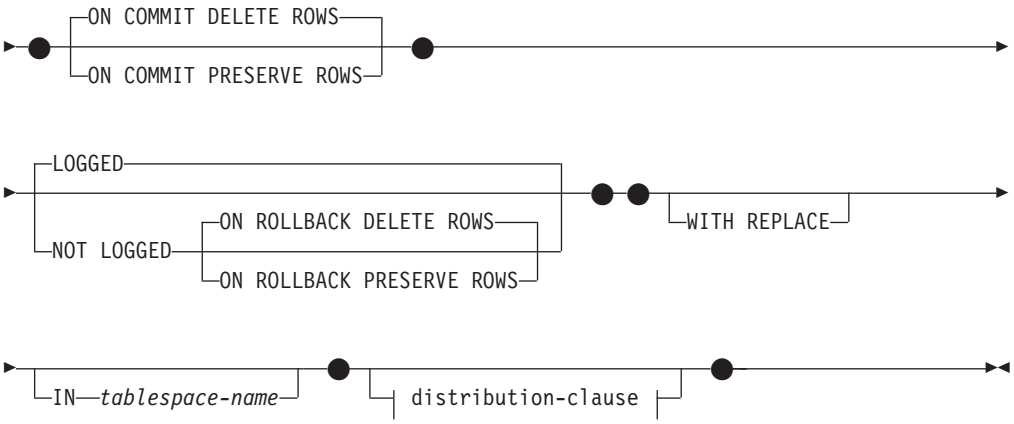
- USER TEMPORARY 테이블 스페이스에 대한 USE 특권
- DBADM 권한
- SYSADM 권한
- SYSCTRL 권한

LIKE 또는 fullselect를 사용하여 테이블을 정의하는 경우 명령문의 권한 부여 ID가 보유하는 특권은 각각의 식별된 테이블이나 뷰에 대해 최소한 다음 중 하나를 포함해야 합니다.

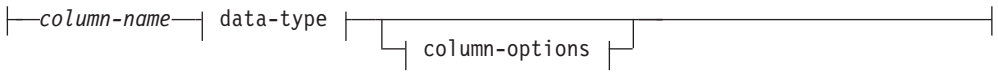
- 테이블 또는 뷰에 대한 SELECT 특권
- 테이블 또는 뷰에 대한 CONTROL 특권
- DATAACCESS 권한

구문



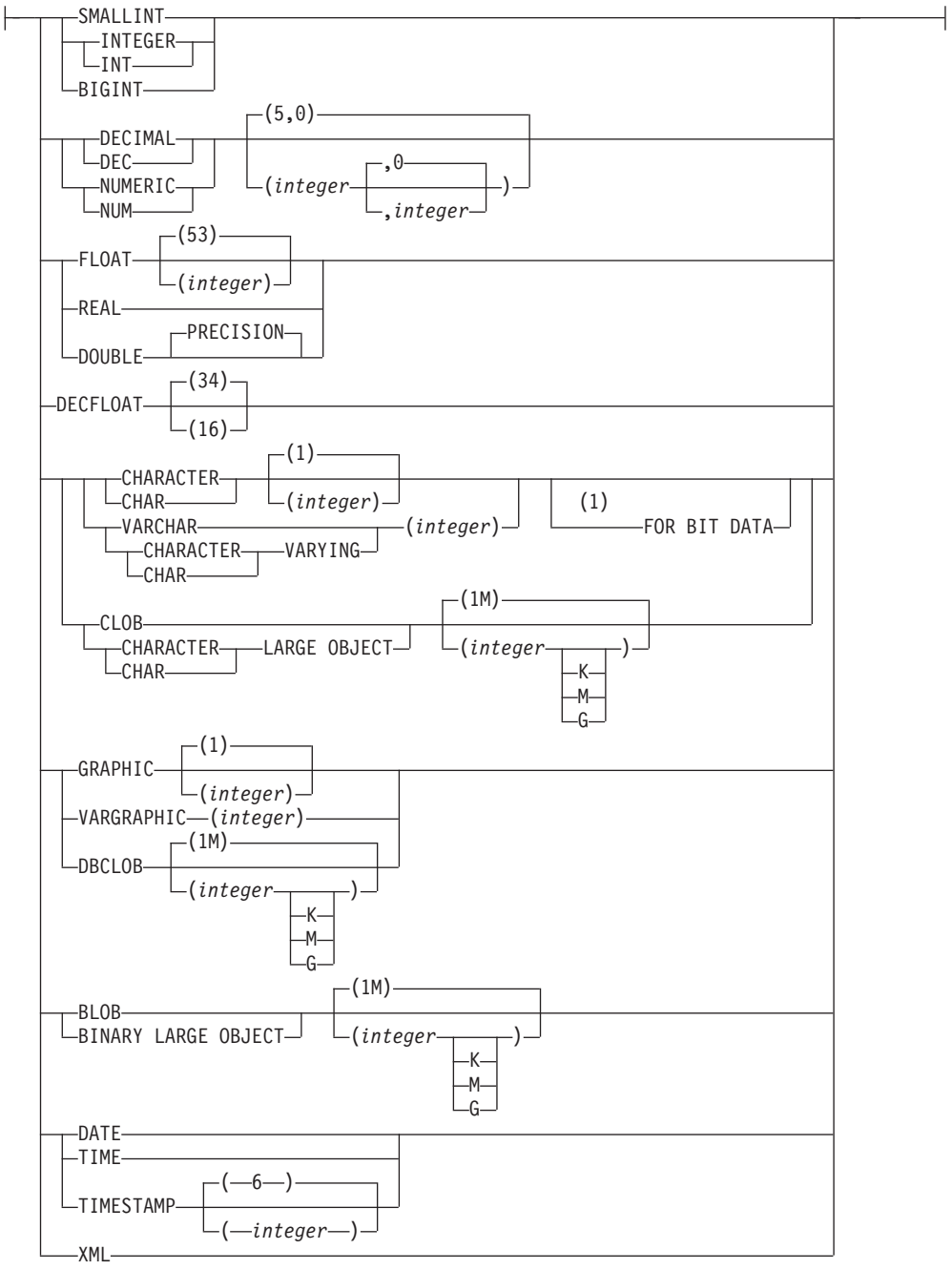


column-definition:

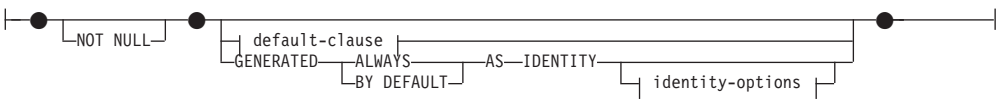


data-type:

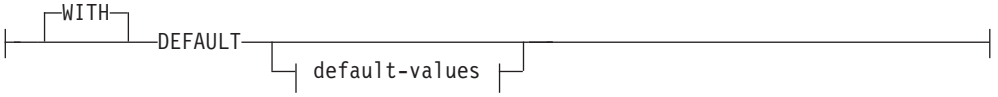
built-in-type:



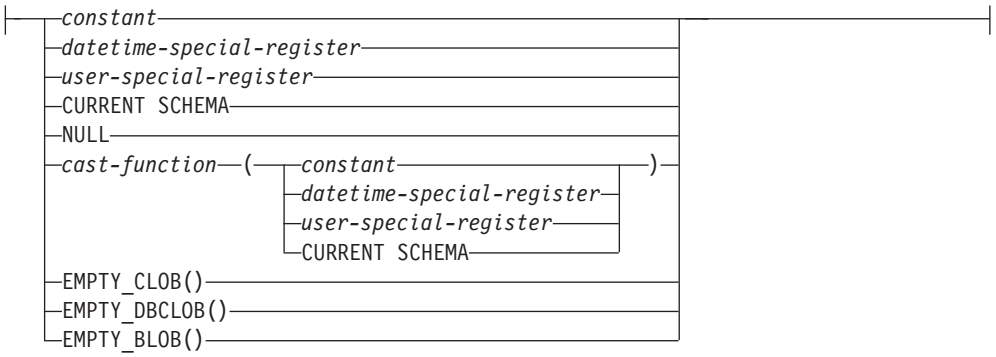
column-options:



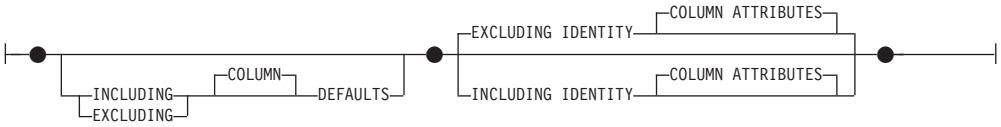
default-clause:



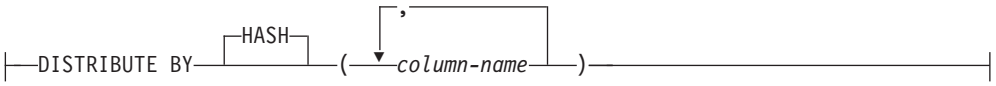
default-values:



copy-options:



distribution-clause:



주:

1 FOR BIT DATA절은 다른 컬럼 제한조건에 대해 임의의 순서로 지정될 수 있습니다.

설명

table-name

임시 테이블의 이름을 지정합니다. 명시적으로 지정된 경우 규정자는 SESSION이어야 하며, 그렇지 않으면 오류가 발생합니다(SQLSTATE 428EK). 규정자가 지정되지 않은 경우에는 SESSION이 내재적으로 지정됩니다.

같은 *table-name*의 선언된 임시 테이블을 정의하는 각 세션은 고유의 선언된 임시 테이블 설명을 가집니다. *table-name*이 이미 세션에 있는 선언된 임시 테이블을 식별하는 경우 WITH REPLACE절을 지정해야 합니다(SQLSTATE 42710).

같은 이름을 가지며 스키마 이름이 SESSION인 테이블, 뷰, 별명 또는 별칭이 이미 카탈로그에 있을 수 있습니다.

- 선언된 임시 테이블 *table-name*은 오류나 경고없이 여전히 정의되어 있을 수 있습니다.

- SESSION.*table-name*에 대한 참조는 카탈로그에 이미 정의되어 있는 SESSION.*table-name*이 아닌 선언된 임시 테이블로 분석됩니다.

column-definition

임시 테이블의 컬럼 속성을 정의합니다.

column-name

테이블의 컬럼을 이름 지정합니다. 이름을 규정할 수 없으며 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

테이블은 다음 사항을 포함합니다.

- 최대 500개 컬럼의 4K 페이지 크기. 여기서, 컬럼의 바이트 수는 4,005 이하여야 합니다.
- 최대 1,012개 컬럼의 8K 페이지 크기. 여기서, 컬럼의 바이트 수는 8,101 이하여야 합니다.
- 최대 1,012개 컬럼의 16K 페이지 크기. 여기서, 컬럼의 바이트 수는 16,293 이하여야 합니다.
- 최대 1,012개 컬럼의 32K 페이지 크기. 여기서, 컬럼의 바이트 수는 32,677 이하여야 합니다.

자세한 정보는 『CREATE TABLE』의 『행 크기』를 참조하십시오.

data-type

컬럼의 데이터 유형을 지정합니다.

built-in-type

내장 데이터 유형을 지정합니다. *built-in-type*의 설명은 『CREATE TABLE』을 참조하십시오.

SYSPROC.DB2SECURITYLABEL 데이터 유형은 선언된 임시 테이블에 지정할 수 없습니다.

distinct-type-name

구별 유형인 사용자 정의 유형의 경우. 구별 유형 이름을 스키마 이름없이 지정하면, 구별 유형 이름은 SQL 경로(정적 SQL의 경우 FUNCPATH 사전 처리 옵션에 의해 정의되고, 동적 SQL의 경우 CURRENT PATH 레지스터에 의해 정의됨)의 스키마를 검색하여 분석됩니다.

구별 유형을 사용하여 정의된, 컬럼의 데이터 유형은 구별 유형입니다. 컬럼의 길이와 스케일은 각각 구별 유형의 소스 유형의 길이와 스케일입니다.

column-options

테이블 컬럼에 관련된 추가 옵션을 정의합니다.

NOT NULL

컬럼에 널(NULL)값이 포함되지 못하도록 합니다. 널(NULL) 값 스펙은 『CREATE TABLE』의 NOT NULL을 참조하십시오.

default-clause

컬럼의 디폴트값을 지정합니다.

WITH

선택적 키워드입니다.

DEFAULT

값이 INSERT에 제공되지 않거나 INSERT 또는 UPDATE의 DEFAULT로 지정된 이벤트에서 디폴트값을 제공합니다. 디폴트값을 DEFAULT 키워드 다음에 지정하지 않은 경우, 디폴트값은 『ALTER TABLE』에 표시된 컬럼의 데이터 유형에 따라 달라집니다.

컬럼이 유형이 지정된 테이블 컬럼을 기반으로 하는 경우, 디폴트값 정의시 특정 디폴트값을 지정해야 합니다. 유형이 지정된 테이블의 오브젝트 ID 컬럼에 대한 디폴트값은 지정할 수 없습니다(SQLSTATE 42997).

컬럼이 구별 유형을 사용하여 정의되면, 컬럼의 디폴트값은 구별 유형으로 캐스트되는 소스 데이터 유형의 디폴트값입니다.

구조화된 유형을 사용하여 컬럼을 정의한 경우, *default-clause*를 지정할 수 없습니다(SQLSTATE 42842).

*column-definition*에서 DEFAULT를 생략하면, 컬럼의 디폴트값으로 널(NULL)값이 사용됩니다. 그러한 컬럼은 NOT NULL로 정의할 경우 유효 디폴트값을 갖지 않습니다.

default-values

지정될 수 있는 디폴트값의 특정 유형은 다음과 같습니다.

constant

컬럼의 디폴트값으로 상수를 지정합니다. 지정된 상수는 다음과 같아야 합니다.

- 지정된 규칙에 따라 컬럼에 지정할 수 있는 값을 나타내야 합니다.
- 컬럼이 부동 소수점 데이터 유형으로 정의되어 있지 않는 한, 부동 소수점 상수가 아니여야 합니다.
- 컬럼의 데이터 유형이 10진수 부동 소수점인 경우 숫자 상수 또는 10진수 부동 소수점 특수 값이어야 합니다. 목표 컬럼이 DECFLOAT이면 부동 소수점 상수는 먼저 DOUBLE로 해석된 다음 10진수 부동 소수점으로 변환됩니다. DECFLOAT(16) 컬럼의 경우, 16보다 정밀도가 큰 10진 상수는 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터에서 지정된 반올림 모드를 사용하여 반올림됩니다.

- 상수가 10진 상수인 경우 컬럼 데이터 유형의 전체 자릿수를 초과하는 0이 아닌 자릿수를 갖지 않아야 합니다(예를 들면, 1.234는 DECIMAL(5,2) 컬럼에 대한 디폴트값이 될 수 없음).
- 따옴표, 16진 상수를 나타내는 X와 같은 도입 문자 및 상수가 *cast-function*의 인수일 때 완전한 함수 이름의 문자 및 괄호를 포함하여 254바이트 이하로 표시됩니다.

datetime-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 날짜 시간 특수 레지스터의 값(CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP)을 지정합니다. 컬럼의 데이터 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다(예: CURRENT DATE 지정시 DATE여야 함).

user-special-register

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시에 사용자 특수 레지스터의 값(CURRENT USER, SESSION_USER, SYSTEM_USER)을 지정합니다. 컬럼의 데이터 유형은 사용자 특수 레지스터의 길이 속성보다 짧지 않은 길이의 문자열이어야 합니다. SESSION_USER 대신 USER를, CURRENT USER 대신 CURRENT_USER를 지정할 수 있습니다.

CURRENT SCHEMA

컬럼의 디폴트값으로 INSERT, UPDATE 또는 LOAD 사용시 CURRENT SCHEMA 특수 레지스터값을 지정합니다. CURRENT SCHEMA가 지정되면, 컬럼의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성과 같거나 그 이상인 문자열이어야 합니다.

NULL

컬럼의 디폴트값으로 널(NULL)을 지정합니다. NOT NULL이 지정되었으면, DEFAULT NULL은 동일한 컬럼 정의 내에 지정될 수 있으나, 이로 인해 디폴트값으로 컬럼을 설정하려고 시도할 때 오류가 발생합니다.

cast-function

이 형식의 디폴트값은 구별 유형, BLOB 또는 날짜 시간(DATE, TIME 또는 TIMESTAMP) 데이터 유형으로 정의된 컬럼과 함께 사용할 수 있습니다. 구별 유형의 경우, BLOB 또는 날짜 시간 유형을 따르는 구별 유형을 제외하고는 함수 이름이 컬럼의 구별 유형 이름과 일치해야 합니다. 스키마 이름으로 규정된 경우, 함수 이름이 구별 유형에 대한 스키마 이름과 같아야 합니다. 규정되지 않은 경우, 함수 결정의 스키마 이름은 구별 유형에 대한 스키마 이

름과 같아야 합니다. 디폴트값이 상수인 날짜 시간 유형을 따르는 구별 유형의 경우, 함수가 사용되어야 하며, 함수 이름은 내재적 또는 명시적 스키마 이름이 SYSIBM인 구별 유형의 소스 유형 이름과 일치해야 합니다. 다른 날짜 시간 컬럼의 경우, 해당하는 날짜 시간 함수가 사용될 수도 있습니다. BLOB 또는 BLOB을 기반으로 하는 구별 유형의 경우, 함수가 사용되어야 하며 함수 이름은 내재적 또는 명시적 스키마 이름으로 SYSIBM을 사용하는 BLOB이어야 합니다.

constant

상수를 인수로 지정합니다. 상수는 구별 유형의 소스 유형 또는 데이터 유형(구별 유형이 아닌 경우)에 대한 상수 규칙을 따라야 합니다. *cast-function*이 BLOB인 경우, 상수는 문자열 상수여야 합니다.

datetime-special-register

CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다.

user-special-register

CURRENT USER, SESSION_USER 또는 SYSTEM_USER를 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 최소한 8바이트 길이의 문자열 데이터 유형이어야 합니다. *cast-function*이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

CURRENT SCHEMA

CURRENT SCHEMA 특수 레지스터의 값을 지정합니다. 컬럼의 구별 유형에 대한 소스 유형의 데이터 유형은 CURRENT SCHEMA 특수 레지스터의 길이 속성보다 크거나 같은 문자열이어야 합니다. cast-function이 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

EMPTY_CLOB(), EMPTY_DBCLOB() 또는 EMPTY_BLOB()

컬럼의 디폴트값으로 영(0) 길이의 문자열을 지정합니다. 컬럼의 데이터 유형은 함수의 결과 데이터 유형을 기반으로 해야 합니다.

지정된 값이 유효하지 않으면, 오류가 발생합니다(SQLSTATE 42894).

IDENTITY 및 *identity-options*

ID 컬럼의 스펙은 『CREATE TABLE』의 IDENTITY 및 *identity-options*를 참조하십시오.

LIKE *table-name1* 또는 *view-name* 또는 *nickname*

테이블 컬럼이 식별된 테이블(*table-name1*)이나 뷰(*view-name*) 또는 별칭(*nickname*)의 컬럼과 정확하게 일치하는 이름 및 설명을 가지도록 지정합니다. LIKE 다음에 지정된 이름은 카탈로그 또는 선언된 임시 테이블에 있는 테이블, 뷰 또는 별칭을 식별해야 합니다. 유형이 지정된 테이블 또는 유형이 지정된 뷰는 지정할 수 없습니다(SQLSTATE 428EC). 보호된 테이블을 지정할 수 없습니다(SQLSTATE 42962).

LIKE의 사용은 *n* 컬럼의 내재된 정의입니다. 여기서, *n*은 식별된 테이블(내재적으로 숨겨진 컬럼 포함), 뷰, 또는 별칭의 컬럼 수입니다. 기존 테이블에서 내재적으로 숨겨진 컬럼에 해당되는 새 테이블의 컬럼은 내재된 숨김으로 정의됩니다. 내재된 정의는 LIKE 이후에 식별되는 내용에 따라 다릅니다.

- 식별된 테이블인 경우 내재된 정의에는 *table-name1*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다. EXCLUDING COLUMN DEFAULTS가 지정되지 않을 경우에는 컬럼 디폴트값도 포함됩니다.
- 식별된 뷰인 경우 내재된 정의에는 *view-name*에 정의된 fullselect의 각 결과 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다.
- 식별된 별칭인 경우 내재된 정의에는 *nickname*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 가능성 특성이 포함됩니다.

*copy-attributes*절에 따라, 컬럼 디폴트값 및 식별 컬럼 속성을 포함하거나 제외시킬 수 있습니다. 내재된 정의에는 식별된 테이블, 뷰 또는 별칭의 다른 속성도 포함되지 않습니다. 따라서 새 테이블은 고유한 제한조건, 외부 키 제한조건, 트리거, 인덱스, 테이블 파티션 키 또는 분산 키를 포함하지 않습니다. 테이블은 IN절에 의해 내재적 또는 명시적으로 지정된 테이블 스페이스에 작성되며, 선택적 절이 지정되는 경우에만 테이블이 다른 선택적 절을 가집니다.

테이블이 LIKE절에서 식별되고 테이블에 ROW CHANGE TIMESTAMP 컬럼이 포함되어 있으면, 새 테이블의 해당 컬럼이 ROW CHANGE TIMESTAMP 컬럼의 데이터 유형만을 상속합니다. 새 컬럼이 생성된 컬럼이 되지 않습니다.

AS (*fullselect*) **DEFINITION ONLY**

fullselect가 실행되는 경우 테이블 컬럼이 fullselect의 도출된 결과 테이블에 나타난 컬럼과 동일한 이름 및 설명을 갖도록 지정하십시오. AS(*fullselect*)의 사용은 선언된 임시 테이블에 대한 *n* 컬럼의 내재된 정의입니다. 여기서, *n*은 fullselect 결과 컬럼의 수입니다.

내재된 정의에는 *n* 컬럼에 대한 다음 속성이 포함됩니다(데이터 유형에 해당되는 경우).

- 컬럼 이름
- 데이터 유형, 길이, 정밀도 및 스케일
- 널(null) 가능성

다음 속성은 포함되지 않습니다(디폴트값 및 ID 속성은 *copy-options*을 사용하여 포함될 수 있습니다).

- 디폴트값
- ID 속성
- ROW CHANGE TIMESTAMP

내재된 정의에는 fullselect에서 참조되는 테이블 또는 뷰의 다른 선택적 속성은 포함되지 않습니다.

모든 선택 목록 요소는 고유한 이름이어야 합니다(SQLSTATE 42711). AS절은 고유한 이름을 제공하는 SELECT절에서 사용할 수 있습니다. fullselect는 호스트 변수나 포함 매개변수 표시문자를 참조해서는 안됩니다.

copy-options

이러한 옵션은 소스 결과 테이블 정의(테이블, 뷰 또는 fullselect)의 추가 속성을 복사할지 여부를 지정합니다.

INCLUDING COLUMN DEFAULTS

소스 결과 테이블 정의를 갱신할 수 있는 각 컬럼의 컬럼 디폴트값이 복사됩니다. 갱신 가능하지 않은 컬럼은 작성된 테이블의 해당 컬럼에 정의된 디폴트값을 가지지 않습니다.

LIKE *table-name1*이 지정되고 *table-name1*이 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별하면, INCLUDING COLUMN DEFAULTS가 디폴트값입니다.

EXCLUDING COLUMN DEFAULTS

컬럼 디폴트값은 소스 결과 테이블 정의로부터 복사되지 않습니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별하는 경우를 제외하고는 이 절이 디폴트값입니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES

가능한 경우 식별 컬럼 속성(START WITH, INCREMENT BY 및 CACHE 값)이 소스 결과 테이블 정의로부터 복사됩니다. 테이블, 뷰 또는 fullselect의 해당 컬럼 요소가 테이블 컬럼의 이름이거나 직접 또는 간접적으로 식별 등록 정보를 가지는 기본 테이블 또는 작성된 임시 테이블의 컬럼 이름에 해당되는 뷰 컬럼의 이름인 경우 식별 컬럼 속성을 복사할 수 있습니다. 다른 모든 경우에는 새 임시 테이블의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.

- fullselect의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함됩니다(즉, 두 번 이상 동일 컬럼 선택).

- fullselect의 선택 목록에 여러 식별 컬럼이 포함됩니다(즉, 하나의 조인 포함).
- 식별 컬럼이 선택 목록의 표현식에 포함됩니다.
- fullselect에 하나의 설정 연산(union, except 또는 intersect)이 포함됩니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES
식별 컬럼 속성이 소스 결과 테이블 정의로부터 복사되지 않습니다.

ON COMMIT
COMMIT 조작 수행시 전역 임시 테이블에 대해 취한 조치를 지정합니다. 디폴트 값은 DELETE ROWS입니다.

DELETE ROWS
테이블에 대해 열려 있는 WITH HOLD 커서가 없는 경우 테이블의 모든 행이 삭제됩니다.

PRESERVE ROWS
테이블 행은 유지됩니다.

LOGGED 또는 NOT LOGGED
테이블에 대한 조작이 로깅되는지 여부를 지정합니다. 디폴트값은 LOGGED입니다.

LOGGED
테이블에 대한 삽입, 갱신 또는 삭제 조작과 테이블의 작성 또는 삭제(drop)도 로그되도록 지정합니다.

NOT LOGGED
테이블에 대한 삽입, 갱신 또는 삭제 조작은 로그되지 않지만 테이블의 작성 또는 삭제(drop)는 로그되도록 지정합니다. ROLLBACK(또는 ROLLBACK TO SAVEPOINT) 조작 중에 다음을 수행하십시오.

- 테이블이 작업 단위(UOW)(또는 세이브포인트) 내에 작성된 경우 테이블은 삭제(drop)됩니다.
- 테이블이 작업 단위(UOW)(또는 세이브포인트) 내에 작성된 경우 테이블은 재작성되지만 데이터는 포함되지 않습니다.

ON ROLLBACK
ROLLBACK(또는 ROLLBACK TO SAVEPOINT) 조작이 수행될 때 로그되지 않은 전역 임시 테이블에 취할 조치를 지정합니다. 디폴트값은 DELETE ROWS입니다.

DELETE ROWS
테이블 데이터가 변경되었으면 모든 행은 삭제됩니다.

PRESERVE ROWS
테이블 행은 유지됩니다.

WITH REPLACE

지정된 이름의 선언된 임시 테이블이 이미 있는 경우 기존 테이블은 이 명령문으로 정의된 임시 테이블로 대체됩니다. 또한 기존 테이블의 모든 행은 삭제됩니다.

WITH REPLACE가 지정되지 않았으면 지정된 이름이 이미 현재 세션에 있는 선언된 임시 테이블을 식별해서는 안됩니다(SQLSTATE 42710).

IN *tablespace-name*

선언된 임시 테이블의 인스턴스화될 테이블 스페이스를 식별합니다. 테이블 스페이스가 있어야 하고 이 테이블 스페이스는 명령문의 권한 부여 ID가 USE 특권을 가지는 사용자 임시 테이블 스페이스이어야 합니다(SQLSTATE 42501). 이 절이 지정되지 않은 경우 해당 명령문의 권한 부여 ID가 USE 특권을 가지는 가장 작은 충분한 페이지 크기의 사용자 임시 테이블 스페이스를 선택함으로써 이 테이블의 테이블 스페이스를 결정합니다. 둘 이상의 테이블 스페이스가 규정되었다면 USE 특권이 부여된 사용자에 따라 환경설정이 제공됩니다.

1. 권한 부여 ID
2. 권한 부여 ID가 속한 그룹
3. PUBLIC

여전히 둘 이상의 테이블 스페이스가 규정되었다면 데이터베이스 관리 프로그램에서 최종 선택을 합니다. 사용자 임시 테이블 스페이스가 규정된 경우에는 오류가 발생합니다(SQLSTATE 42727).

테이블 스페이스 판별은 다음 경우에 변경될 수 있습니다.

- 테이블 스페이스가 삭제(drop) 또는 작성되었을 때
- USE 특권이 부여되었거나 취소되었을 때

테이블의 페이지 크기가 충분한지 여부는 행의 바이트 합계 또는 컬럼 수에 의해 결정됩니다. 자세한 정보는 『CREATE TABLE』의 『행 크기』를 참조하십시오.

distribution-clause

데이터베이스 파티션 또는 다중 데이터베이스 파티션에 데이터를 분배하는 방법을 지정합니다.

DISTRIBUTE BY HASH (*column-name***,...)**

분산 키라고 불리는 지정 컬럼의 디폴트 해싱 함수 사용을 데이터베이스 파티션에 분산 메소드로 지정합니다. *column-name*은 테이블의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다(SQLSTATE 42703). 동일한 컬럼은 두 번 이상 식별될 수 없습니다(SQLSTATE 42709). 데이터 유형이 BLOB, CLOB, DBCLOB, XML, 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형인 컬럼은 분산 키의 일부로 사용될 수 없습니다(SQLSTATE 42962).

해당 절이 지정되지 않고 테이블이 여러 데이터베이스 파티션이 있는 다중 파티션 데이터베이스 파티션 그룹에 상주할 경우, 분산 키는 해당 데이터 유형이 분산 키에 대해 유효한 첫 번째 컬럼으로 정의됩니다.

디폴트 분산 키 요건을 충족시키는 컬럼이 없을 경우, 분산 키 없이 테이블이 작성됩니다. 이러한 테이블은 단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에서만 사용할 수 있습니다.

단일 파티션 데이터베이스 파티션 그룹에 정의된 테이블 스페이스에 있는 테이블의 경우, 분산 키에 대해 유효한 데이터 유형을 가진 컬럼의 콜렉션은 분산 키를 정의하는 데 사용될 수 있습니다. 이 절을 지정하지 않으면 분산 키가 작성되지 않습니다.

주

- 사용자 임시 테이블 스페이스는 선언된 임시 테이블이 선언되기 전에 존재해야 합니다(SQLSTATE 42727).
- ***선언된 임시 테이블 참조:*** 선언된 임시 테이블의 설명은 DB2 카탈로그(SYSCAT.TABLES)에 나타나지 않으므로 지속적이 아니고 데이터베이스 연결을 통해 공유 가능하지 않습니다. 이는 선언된 임시 테이블 *table-name*을 정의하는 각 세션이 선언된 전역 임시 테이블의 고유한 설명을 가짐을 의미합니다.

  SQL문(DECLARE GLOBAL TEMPORARY TABLE 문이 아닌 다른)에서 선언된 임시 테이블을 참조하기 위해서는 스키마 이름 SESSION으로 이 테이블을 명시적 또는 내재적으로 규정해야 합니다. *table-name*이 SESSION으로 규정되지 않은 경우에는 참조 해석 시 선언된 임시 테이블이 고려되지 않습니다.

  이름으로 선언된 임시 테이블을 선언하지 않은 연결에서 SESSION.*table-name* 에 대한 참조는 카탈로그의 지속적 오브젝트로부터 분석하려고 시도합니다. 이와 같은 오브젝트가 없으면 오류가 발생합니다(SQLSTATE 42704).
- SESSION에 의해 내재적으로 또는 명시적으로 규정된 테이블을 참조하는 정적 SQL 문이 있는 패키지를 바인딩하는 경우, 이러한 명령문이 정적으로 바인드되지 않습니다. 이러한 명령문은 호출될 때 패키지 바인딩 시 선택된 VALIDATE 옵션에 관계없이 증분식으로 바인드됩니다. 런타임 시 각 테이블 참조는 선언된 임시 테이블이 있는 경우 선언된 임시 테이블, 또는 작성된 임시 테이블이나 영구 테이블로 해석됩니다. 모두 없으면 오류가 발생합니다(SQLSTATE 42704).
- ***특권:*** 선언된 임시 테이블이 정의된 경우 이 테이블의 정의자에게 테이블 삭제 권한을 비롯하여 테이블에 대한 모든 테이블 특권이 부여됩니다. 또한 이들 특권은 PUBLIC에 부여됩니다. GRANT 옵션으로 부여된 특권이 없으므로 카탈로그 테이블에 특권이 나타나지 않습니다. 이를 통해 세션의 모든 SQL문은 이미 해당 세션에 정의되어 있는 선언된 임시 테이블을 참조할 수 있습니다.

- *인스턴스화 및 종료:* 아래 설명에서 P는 세션이고 T는 세션 P에서 선언된 임시 테이블입니다.
  - 빈 인스턴스 T가 P에서 실행되는 DECLARE GLOBAL TEMPORARY TABLE 문의 결과로 작성됩니다.
  - P의 SQL문은 T에 대한 참조를 작성할 수 있습니다. P에서 T에 대한 모든 참조는 T의 동일한 인스턴스를 참조합니다.
  - DECLARE GLOBAL TEMPORARY TABLE 문이 SQL 프로시저 복합 명령문(BEGIN 및 END로 정의) 내에 지정되는 경우, 선언된 임시 테이블의 범위는 복합 명령문일 뿐만 아니라 연결 명령문이기도 하며 이 테이블은 복합 명령문의 외부에 알려져 있습니다. 테이블은 복합 명령문 종료시 내재적으로 삭제되지 않습니다. 선언된 임시 테이블은 테이블이 명시적으로 삭제되지 않는 한 해당 세션의 다른 복합 명령문에 같은 이름으로 여러 번 정의할 수 없습니다.
  - ON COMMIT DELETE ROWS절이 내재적으로 또는 명시적으로 지정되었다고 가정할 때, 커미트 조작이 P에서 작업 단위(UOW)를 종료하고 T에 종속적인 P에 열려 있는 WITH HOLD 커서가 없다면 커미트에 DELETE FROM SESSION.T 조작이 포함됩니다.
  - 롤백 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 해당 작업 단위 또는 세이브포인트에 SESSION.T의 수정사항이 포함되는 경우
    - NOT LOGGED가 지정되면 ON ROLLBACK PRESERVE ROWS도 지정되어 있지 않은 경우 롤백에는 SESSION.T의 DELETE 조작이 포함되며, 이런 경우에 DELETE 조작은 T의 모든 행을 삭제합니다.
    - NOT LOGGED가 지정되지 않으면 T는 변경되지 않습니다.

    롤백 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 SESSION.T의 선언이 포함되면 롤백에 DROP SESSION.T 조작이 포함됩니다.

    롤백 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 선언된 임시 테이블 SESSION.T의 삭제가 포함되면 롤백이 테이블 삭제 실행을 취소합니다. NOT LOGGED가 지정되면 테이블도 비워집니다.
  - T를 선언한 응용프로그램 프로세스가 종료되고 데이터베이스와의 연결이 끊기면 T가 삭제되고 인스턴스화된 행이 없어집니다.
  - T가 선언된 서버로의 연결이 종료되면, T가 삭제되고 인스턴스화된 행이 없어집니다.
- *선언된 임시 테이블 사용의 제한사항:* 선언된 임시 테이블에서는 다음을 수행할 수 없습니다.
  - ALTER, COMMENT, GRANT, LOCK, RENAME 또는 REVOKE문에서의 정의(SQLSTATE 42995)

  - AUDIT, CREATE ALIAS 또는 CREATE VIEW문에서의 참조(SQLSTATE 42995)
  - 참조 제한조건에 지정(SQLSTATE 42995)
- 선언된 임시 테이블에 대해 데이터 행 압축을 사용할 수 있습니다. 데이터베이스 관리 프로그램은 성능이 더 향상될 수 있다면 기본 테이블 오브젝트에 인라인으로 저장된 XML 문서가 포함된 테이블 행 데이터를 압축합니다. 그렇지만 선언된 임시 테이블의 XML 스토리지 오브젝트에 대한 데이터 압축은 지원되지 않습니다.
- 선언된 임시 테이블에서 작성된 인덱스에 대해서는 인덱스 압축을 수행할 수 있습니다.
- *호환성*:
  - 이전 버전 DB2와의 호환성을 위해,
    - DISTRIBUTE BY절 대신 PARTITIONING KEY절 지정 가능
  - z/OS용 DB2와의 호환성을 위해,
    - 다음 구문이 디폴트 동작으로 허용됩니다.
      - CCSID ASCII
      - CCSID UNICODE

예:

예 1: 직원 수, 임금, 보너스 및 커미션에 대해 컬럼 정의로 선언된 임시 테이블을 정의합니다.

```
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 테이블 USER1.EMPTAB가 존재하고 3개의 컬럼을 포함하며 이들 중 하나가 ID 컬럼이라고 가정합니다. 동일 컬럼 이름 및 속성을 갖는 임시 테이블(ID 속성 포함)을 기본 테이블로 선언합니다.

```
DECLARE GLOBAL TEMPORARY TABLE TEMPTAB1
   LIKE USER1.EMPTAB
   INCLUDING IDENTITY
   ON COMMIT PRESERVE ROWS
```

위의 예에서 SESSION은 TEMPTAB1의 내재된 규정자로 사용됩니다.

DELETE

DELETE문은 테이블, 별칭 또는 뷰에서 행을 삭제하거나 지정된 fullselect의 뷰, 별칭 또는 하위 테이블을 삭제합니다. 별칭에서 행을 삭제하면 별칭이 참조하는 데이터 소스 오브젝트의 행이 삭제됩니다. 뷰에 대한 삭제 조작에 대해 INSTEAD OF 트리거가 정의되어 있지 않은 경우 이 뷰에서 행을 삭제하면 뷰가 바탕을 두는 테이블의 행이 삭제됩니다. 이와 같은 트리거가 정의되었을 경우 트리거가 대신 실행됩니다.

이 명령문은 다음 두 가지 형식을 갖고 있습니다.

- *검색* DELETE 형식은 두 개 이상의 행을 삭제할 경우 사용됩니다(검색 조건에 의해 선택적으로 판별됨).
- *위치 지정* DELETE 형식은 정확히 한 행만 삭제할 경우 사용됩니다(커서의 현재 위치에 의해 판별되는 대로).

호출

DELETE문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

이 명령문의 두 형식 중 하나를 실행하려면, 명령문은 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 행이 삭제될 테이블, 뷰 또는 별칭에서의 DELETE 특권
- 행이 삭제될 테이블, 뷰 또는 별칭에서의 CONTROL 특권
- DATAACCESS 권한

검색된 DELETE문을 실행하려면 명령문의 권한 부여 ID는 서브쿼리에서 참조하는 각 테이블, 뷰 또는 별칭에 대해 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- SELECT 특권
- CONTROL 특권
- DATAACCESS 권한

명령문 처리에 사용되는 패키지가 SQL92 규칙으로 프리컴파일되어 있고(즉, LANGLEVEL 옵션 값이 SQL92E나 MIA), DELETE문의 검색 양식에 *search-condition*에 있는 테이블이나 뷰의 컬럼에 대한 참조가 포함되어 있으면 명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- SELECT 특권
- CONTROL 특권
- DATAACCESS 권한

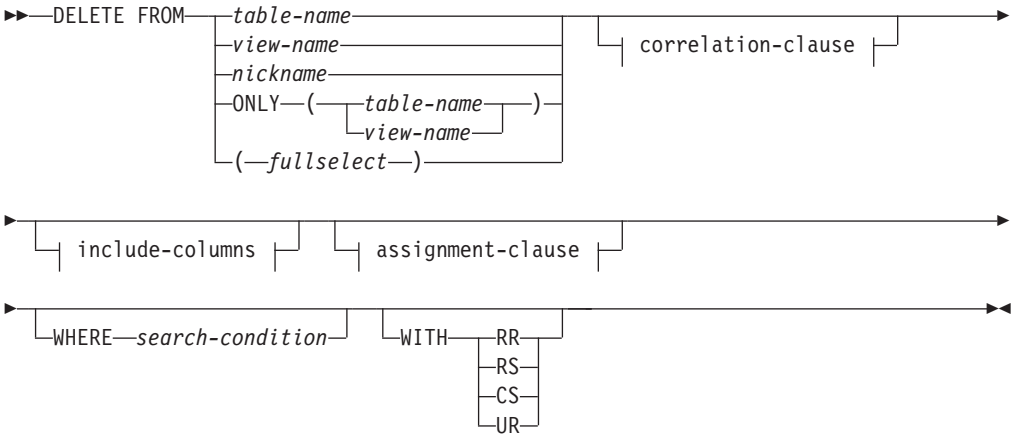
지정된 테이블이나 뷰 앞에 ONLY 키워드가 오는 경우 명령문의 권한 부여 ID가 가지고 있는 특권에는 지정된 테이블이나 뷰의 모든 서브테이블 또는 서브뷰에 대한 SELECT 특권도 포함되어야 합니다.

정적 DELETE문에 대해서는 Group 특권이 점검되지 않습니다.

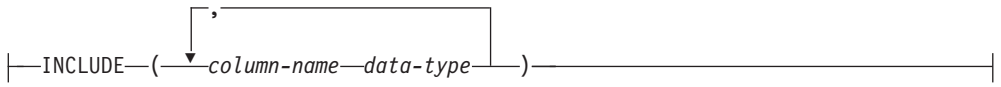
삭제 조작의 목표가 별칭일 경우에는 데이터 소스에 대해 명령문을 발행하기 전에는 데이터 소스의 오브젝트에 대한 특권이 무시됩니다. 이 때 데이터 소스에 연결하는 데 사용되는 권한 부여 ID에는 데이터 소스에서 해당 오브젝트에 조작을 수행하는 데 필요한 특권이 있어야 합니다. 명령문의 권한 부여 ID는 데이터 소스에서 서로 다른 권한 부여 ID로 맵핑될 수 있습니다.

구문

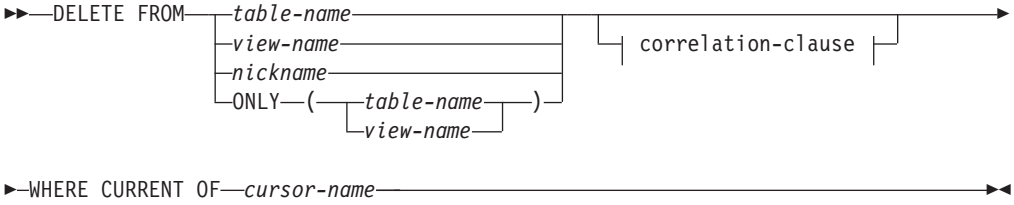
searched-delete:



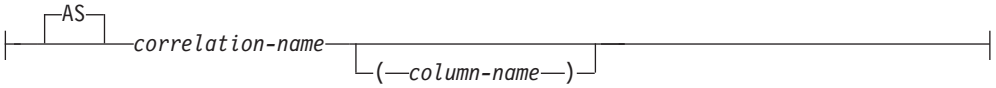
include-columns:



positioned-delete:



correlation-clause:



설명

FROM *table-name*, *view-name*, *nickname*, 또는 *(fullselect)*

삭제 조작의 오브젝트를 식별합니다. 이름은 카탈로그에 있는 테이블 또는 뷰를 식별해야 하지만 카탈로그 테이블, 카탈로그 뷰, 시스템 유지보수 구체화된 쿼리 테이블 또는 읽기 전용 뷰를 식별하면 안됩니다.

*table-name*이 유형이 지정된 테이블인 경우 테이블 행 또는 적절한 서브테이블이 명령문에 의해 삭제될 수 있습니다.

*view-name*이 유형이 지정된 뷰인 경우, 하위 테이블의 행 또는 뷰의 적절한 서브뷰의 하위 테이블에서 행이 명령문에 의해 삭제될 수도 있습니다. *view-name*이 유형이 지정된 테이블인 하위 테이블을 가진 유형이 지정된 뷰인 경우, 유형이 지정된 테이블이나 적절한 서브의 행이 명령문에 의해 삭제될 수도 있습니다.

삭제 조작의 오브젝트가 fullselect인 경우 CREATE VIEW문의 설명에 있는 『삭제 가능한 뷰』 주 항목에 정의되어 있는 대로 fullselect는 삭제 가능해야 합니다.

지정된 테이블의 컬럼만 WHERE절에서 참조할 수 있습니다. 위치 지정된 DELETE의 경우 연관된 커서가 ONLY를 사용하지 않고도 FROM절에서 테이블이나 뷰도 지정했어야 합니다.

FROM ONLY (*table-name***)**

유형이 지정된 테이블에 적용 가능한 ONLY 키워드는 명령문이 지정된 테이블 데이터에만 적용되고 해당 서브테이블 행은 삭제할 수 없도록 지정합니다. 위치 지정된 DELETE인 경우 연관된 커서가 ONLY를 사용하여 FROM절에 있는 테이블도 지정했을 것입니다. *table-name*이 유형이 지정된 테이블이 아니면 ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

FROM ONLY (*view-name***)**

유형이 지정된 뷰에 적용 가능한 ONLY 키워드는 명령문이 지정된 뷰 데이터에만 적용되고 해당 서브뷰의 행은 삭제할 수 없도록 지정합니다. 위치 지정된 DELETE인 경우 연관된 커서가 ONLY를 사용하여 FROM절에 있는 뷰도 지정합니다. *view-name*이 유형이 지정된 뷰가 아니면 ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

correlation-clause

테이블, 뷰, 별칭 또는 fullselect를 지시하기 위해 *search-condition* 내에서 사용될 수 있습니다. *correlation-clause*의 설명은 『Subselect』의 『table-reference』를 참조하십시오.

include-columns

table-name 또는 *view-name*의 컬럼과 함께 DELETE문의 중간 결과 테이블에 포함되는 일련의 컬럼을 지정합니다(컬럼이 fullselect의 FROM절에서 중첩되는 경우). *include-columns*는 *table-name* 또는 *view-name*으로 사용되는 컬럼의 목록 끝에 추가됩니다.

INCLUDE

컬럼 목록이 DELETE문의 중간 결과 테이블에 포함되도록 지정합니다.

column-name

DELETE문에 있는 중간 결과 테이블의 컬럼을 지정합니다. 이름은 *table-name* 또는 *view-name*에서 컬럼 또는 다른 Include 컬럼의 이름과 동일해서는 안됩니다(SQLSTATE 42711).

data-type

Include 컬럼의 데이터 유형을 지정합니다. 데이터 유형은 CREATE TABLE 문으로 지원된 하나여야 합니다.

assignment-clause

UPDATE문의 *assignment-clause* 설명을 참조하십시오. 동일한 규칙이 적용됩니다. *include-columns*는 *assignment-clause*를 사용하여 설정할 수 있는 유일한 컬럼입니다(SQLSTATE 42703).

WHERE

삭제될 행을 선택하는 조건을 지정합니다. 절을 생략하거나, 검색 절을 지정하거나, 커서 이름을 지정할 수 있습니다. 절이 생략되면 테이블이나 뷰의 모든 절이 삭제됩니다.

search-condition

서브쿼리가 아닌 검색 조건에서의 각 *column-name*은 테이블이나 뷰의 컬럼을 식별해야 합니다.

*search-condition*은 테이블, 뷰 또는 별칭의 각 행에 적용되고 삭제된 행은 *search-condition*을 만족하는 행입니다.

검색 조건에 서브쿼리가 있으면 *search-condition*이 행에 적용될 때마다 실행되어 *search-condition*을 적용할 때 사용되는 결과가 될 수 있습니다. 실제로 상관된 참조가 없는 서브쿼리는 한 번 실행되는 반면, 상관된 참조를 갖는 서브쿼리는 각 행에 대해 매번 실행되어야 할 수도 있습니다. 서브쿼리가 DELETE 문의 오브젝트 테이블이나 CASCADE 또는 SET NULL의 삭제 규칙을 갖는 종속 테이블을 참조할 경우 서브쿼리는 행이 삭제되기 전에 완전히 평가됩니다.

CURRENT OF *cursor-name*

프로그램의 DECLARE CURSOR문에 정의되는 커서를 식별합니다. DECLARE CURSOR문 앞에는 DELETE문이 있어야 합니다.

이름이 지정된 테이블, 뷰 또는 별칭은 커서에 대해 SELECT문의 FROM절에서도 이름이 지정되어야 하고 커서의 결과 테이블은 읽기 전용이어야 합니다. (읽기 전용 결과 테이블에 대한 설명은 『DECLARE CURSOR』에서 참조하십시오.)

DELETE문이 실행될 때 커서는 행에 있어야 합니다. 해당 행은 삭제됩니다. 삭제하고 나면 커서는 결과 테이블의 다음 행 앞에 놓입니다. 다음 행이 없으면 커서는 마지막 행 뒤에 놓입니다.

WITH

삭제될 행을 찾을 때 사용되는 분리 레벨을 지정합니다.

RR

반복 읽기

RS

읽기 안정성

CS

커서 안정성

UR

언커미트 읽기

명령문의 디폴트 분리 레벨은 명령문이 바운드되는 패키지의 분리 레벨입니다. WITH 절은 항상 명령문의 디폴트 분리 레벨을 사용하는 별칭에 영향을 주지 않습니다.

규칙

- *트리거:* DELETE문이 트리거를 실행시킵니다. 트리거로 인해 다른 명령문이 실행될 수도 있고, 삭제된 행으로 인해 오류가 발생할 수도 있습니다. 뷰의 DELETE문으로 인해 INSTEAD OF 트리거가 발생하면 트리거를 발생시킨 뷰의 하위 테이블에 대해서가 아니라 트리거에서 수행된 갱신에 대해 참조 무결성이 점검됩니다.
- *참조 무결성:* 식별된 테이블이나 식별된 뷰의 기본 테이블이 상위인 경우, 삭제하도록 선택된 행에는 RESTRICT의 삭제 규칙과 관련된 종속성이 없어야 하며 DELETE는 RESTRICT의 삭제 규칙과 관련하여 종속 항목이 포함된 하위 행에 연쇄될 수 없습니다.

  삭제 조작이 RESTRICT 삭제 규칙으로 금지되지 않는 경우 선택된 행이 삭제됩니다. 선택된 행에 종속되는 행도 영향을 받습니다.

  – 또한 SET NULL의 삭제 규칙과의 관계에서 종속인 행에 대한 외부 키의 널(NULL) 값이 될 수 있는 컬럼은 널(NULL) 값으로 설정됩니다.
  – CASCADE의 삭제 규칙과의 관계에서 종속인 행도 삭제되고 같은 규칙이 해당 행에 적용됩니다.

다른 참조 제한조건이 시행된 후 널(NULL) 값이 아닌 외부 키가 기존의 상위 행을 참조할 수 있도록 하기 위해 NO ACTION의 삭제 규칙이 점검됩니다.

- *보안 규정:* 식별된 테이블이나 식별된 뷰의 기본 테이블이 보안 규정으로 보호된 경우, 세션 권한 부여 ID에는 허용되는 레이블 기반 액세스 제어(LBAC) 증명서가 있어야 합니다.
  - 보호되는 모든 컬럼에 대한 쓰기 액세스(SQLSTATE 42512)
  - 삭제를 선택한 모든 행에 대한 읽기 및 쓰기 액세스(SQLSTATE 42519)

주

- 여러 행으로 된 DELETE 실행시 오류가 발생하면 데이터베이스가 변경되지 않습니다.
- 적절한 잠금이 이미 존재하지 않는 한 성공적인 DELETE문 실행시 하나 이상의 배타적 잠금이 확보됩니다. COMMIT 또는 ROLLBACK문을 발행하면 잠금사항이 해제됩니다. 커미트 및 롤백 조작으로 잠금이 해제될 때까지 삭제 조작의 효과는 다음에 의해서만 발견될 수 있습니다.
  - 삭제를 수행한 응용프로그램 프로세스
  - 분리 레벨 UR을 사용하는 다른 응용프로그램 프로세스

  잠금은 다른 응용프로그램 프로세스가 테이블에서 수행되지 못하도록 합니다.
- 응용프로그램 프로세스에서 커서가 가리키는 행을 삭제할 경우 해당 커서들은 결과 테이블의 다음 행 앞에 놓입니다. 행 R 앞에 놓인 커서를 C라고 가정합시다(OPEN의 결과로: DELETE - C, DELETE - 일부 다른 커서 또는 검색된 DELETE). R이 유래되는 기본 테이블에 영향을 미치는 INSERT, UPDATE 및 DELETE 조작 시 C를 참조하는 다음 FETCH 조작이 반드시 C를 R에 위치시켜야 하는 것은 아닙니다. 예를 들어, 이 조작은 C를 R에 위치시킬 수 있습니다. 여기서, R은 현재 결과 테이블의 다음 행인 새 행입니다.
- SQLCA의 SQLERRD(3)는 삭제 조작에 대해 규정된 행 수를 표시합니다. SQL 프로시저 명령문에서는 GET DIAGNOSTICS문의 ROW_COUNT 변수를 사용하여 이 값을 검색할 수 있습니다. SQLCA의 SQLERRD(5)는 참조 제한조건과 트리거된 명령문에 의해 영향을 받는 행 수를 나타냅니다. 여기에는 외부 키가 SET NULL 삭제 규칙의 결과로 NULL이 설정된 행과 CASCADE 삭제 규칙의 결과로 삭제된 행이 포함됩니다. 트리거된 명령문의 경우 여기에 삽입, 갱신 또는 삭제된 행 수도 포함됩니다.
- 검색 조건과 일치하는 모든 행과 기존의 참조 제한조건에 의해 요구되는 모든 조작을 삭제할 수 없는 오류가 발생하면 테이블에서 어떠한 변경사항도 수행되지 않고 오류가 발생합니다.
- 별칭의 경우에는 외부 서버 옵션 `iud_app_svpt_enforce`가 추가 제한을 설정합니다. 자세한 내용은 페더레이티드 문서를 참조하십시오.

- 일부 데이터 소스의 경우 별칭에 대한 삭제 조작을 수행하면 가능한 데이터 불일치로 인해 SQLCODE -20190이 리턴됩니다. 자세한 내용은 페더레이티드 문서를 참조하십시오.
- **호환성:** 다음 구문이 지원되지만 표준이 아니며 사용하면 안됩니다.
  - FROM 키워드를 생략할 수 있습니다.

예:

예 1: DEPARTMENT 테이블에서 부서(DEPTNO) 'D11'을 삭제하십시오.

```
DELETE FROM DEPARTMENT
  WHERE DEPTNO = 'D11'
```

예 2: DEPARTMENT 테이블에서 모든 부서를 삭제하십시오. (즉, 테이블을 비우십시오.)

```
DELETE FROM DEPARTMENT
```

예 3: EMPLOYEE 테이블에서 1995년에 영업하지 않은 모든 영업 대표 또는 현장 책임자를 삭제하십시오.

```
DELETE FROM EMPLOYEE
  WHERE LASTNAME NOT IN
    (SELECT SALES_PERSON
      FROM SALES
      WHERE YEAR(SALES_DATE)=1995)
      AND JOB IN ('SALESREP','FIELDREP')
```

예: EMPLOYEE 테이블에서 중복되는 피고용인 행을 모두 삭제하십시오. 피고용인의 성이 일치하면 중복되는 행으로 간주합니다. 피고용인 행을 이름의 사전적 순서대로 유지하십시오.

```
DELETE FROM
      (SELECT ROWNUMBER() OVER (PARTITION BY LASTNAME ORDER BY FIRSTNME)
          FROM EMPLOYEE) AS E(RN)
   WHERE RN = 1
```

DESCRIBE

DESCRIBE문은 오브젝트에 대한 정보를 얻습니다. 이 명령문을 사용하여 얻을 수 있는 두 가지 유형의 정보가 있습니다. 각 정보 유형은 별도로 설명됩니다.

- 준비된 명령문의 입력 매개변수 표시문자. 준비된 명령문에 있는 입력 매개변수 표시문자에 관한 정보를 얻습니다. 이 정보는 디스크립터에 들어갑니다.
- 준비된 명령문의 출력. 준비된 명령문에 대한 정보 또는 준비된 SELECT문의 선택 목록 컬럼에 대한 정보를 얻습니다. 이 정보는 디스크립터에 들어갑니다.

DESCRIBE INPUT

DESCRIBE INPUT문은 PREPARE문의 입력 매개변수 표시문자에 대한 정보를 얻습니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

해당사항 없음.

구문

설명

statement-name

PREPARE문을 식별합니다. DESCRIBE INPUT문이 실행되면 이름은 현재 서버의 응용프로그램 프로세스에서 준비된 명령문을 식별해야 합니다.

CALL문의 경우, 리턴된 정보는 프로시저의 IN 또는 INOUT로 정의된 입력 매개변수를 설명합니다. 입력 매개변수 표시문자는 사용과 상관없이 항상 널(NULL)을 입력할 수 있다고 간주됩니다.

INTO *descriptor-name*

SQL 디스크립터 영역(SQLDA)을 식별합니다. DESCRIBE INPUT문을 실행하기 전에 SQLDA에 있는 다음 변수가 설정되어 있어야 합니다.

SQLN SQLDA에 제공된 SQLVAR 어커런스 수를 지정합니다. SQLN이 0 이상의 값으로 설정되어야 DESCRIBE INPUT문이 실행됩니다.

DESCRIBE INPUT문이 실행될 때 데이터베이스 관리 프로그램이 다음과 같이 값을 SQLDA의 변수에 지정합니다.

SQLDAID

첫 번째 6바이트는 'SQLDA'로 설정되어 있습니다(즉, 5개 문자 뒤에 공백 문자로 구성).

SQLDOUBLED로 정의된 일곱 번째 바이트가 다음에서 설명된 매개변수 표시문자를 기본으로 설정됩니다.

- SQLDA에 각 입력 매개변수에 대해 두 개의 SQLVAR 항목을 포함하면, 7번째 바이트가 '2'로 설정됩니다. 이 기술은 LOB 또는 구조화된 유형 입력 매개변수를 수용하는 데 사용됩니다.

- 그 외의 경우 7번째 바이트는 공백 문자로 설정됩니다.

SQLDA에 모든 입력 매개변수 표시 문자의 설명을 포함할 만큼 공간이 없으면 7번째 바이트가 공백 문자로 설정됩니다.

8번째 바이트는 공백 문자로 설정됩니다.

SQLDABC

SQLDA의 길이입니다(바이트).

SQLD 프로시저의 IN 및 INOUT 매개변수.

SQLVAR

SQLD 값이 0이거나 SQLN 값보다 큰 경우, SQLVAR 어커런스에 아무 값도 지정되지 않습니다.

SQLD 값이 *n*이고, *n*이 0보다 크지만 SQLN보다 작거나 같으면, 값은 SQLVAR의 첫 번째 *n* 어커런스에 지정됩니다. 값은 프로시저의 입력 매개변수에 대한 매개변수 표시 문자를 설명합니다. SQLVAR의 첫 번째 어커런스는 첫 번째 입력 매개변수 표시 문자를 설명하며, SQLVAR의 두 번째 어커런스는 두 번째 입력 매개변수 표시문자를 설명하는 식입니다.

기본 SQLVAR

SQLTYPE

매개변수의 데이터 유형과 널(NULL) 값이 들어 있는지 여부를 보여주는 코드입니다.

SQLLEN

매개변수 데이터 유형에 따라 다른 길이 값. SQLLEN은 LOB 데이터 유형에 대해 0입니다.

SQLNAME

sqlname은 다음과 같이 파생됩니다.

- SQLVAR이 프로시저의 매개변수 목록에 있는 표현식의 일부가 아닌 매개변수 표시문자와 일치하는 경우, CREATE PROCEDURE 문에 매개변수 이름이 지정되어 있으면 sqlname에 매개변수 이름이 포함됩니다.
- SQLVAR이 이름이 지정된 매개변수 표시문자와 일치하는 경우, sqlname에 매개변수 표시문자의 이름이 포함됩니다.
- 그렇지 않으면 sqlname에 SQLDA 내에서 SQLVAR의 위치를 나타내는 ASCII 숫자 리터럴 값이 포함됩니다.

보조 SQLVAR

이러한 변수는 LOB, 구별 유형, 구조화된 유형 또는 참조 유형 매개변수를 수용하도록 SQLVAR 항목 수가 두 배가 되는 경우에만 사용됩니다.

SQLLONGLEN

BLOB, CLOB 또는 DBCLOB 매개변수의 길이 속성입니다.

SQLDATATYPE_NAME

사용자 정의 유형(구별 또는 구조화) 매개변수의 경우 데이터베이스 관리 프로그램이 이를 완전한 사용자 정의 유형 이름으로 설정합니다. 참조 유형 매개변수의 경우, 데이터베이스 관리 프로그램이 이를 참조 목표 유형의 완전한 사용자 정의 유형 이름으로 설정합니다. 그렇지 않을 경우, 스키마 이름은 SYSIBM이고 유형 이름은 SYSCAT.DATATYPES 카탈로그 뷰의 TYPENAME 컬럼 이름입니다.

주

- **SQLDA 준비:** DESCRIBE INPUT문이 실행되기 전에, SQLDA가 할당되어야 하며 SQLN 값이 0보다 크거나 같은 값으로 설정되어야 SQLVAR 어커런스가 얼마나 많이 제공되어 있는지를 표시할 수 있습니다. 충분한 스토리지가 할당되어 SQLN 어커런스를 포함시켜야 합니다. PREPARE문의 입력 매개변수 표시문자의 설명을 보려면 SQLVAR 어커런스 수가 입력 매개변수 표시문자의 수보다 적으면 안됩니다. 더우기 입력 매개변수 표시문자에 LOB나 구조화된 유형이 포함되면, SQLVAR 어커런스 수가 입력 매개변수 표시문자 수의 두 배가 되어야 합니다.
- 확장 UNIX 코드(EUC) 코드 페이지와 DBCS 코드 페이지 사이, 또는 유니코드와 유니코드 이외의 코드 페이지 사이의 코드 페이지 변환으로 문자 길이가 확장 또는 축소될 수 있습니다.
- 구조화된 유형이 선택되지만 FROM SQL 변환은 지정되지 않은 경우(CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터를 사용하여 TRANSFORM GROUP이 지정되지 않았기 때문이거나(SQLSTATE 428EM) 이름이 지정된 그룹에 정의된 FROM SQL 변환 함수가 없기 때문에(SQLSTATE 42744)), 오류가 리턴됩니다.
- **SQLDA 할당:** 다음은 SQLDA를 할당할 수 있는 세 가지 방법입니다.

  첫 번째 기술: 응용프로그램이 처리해야 할 선택 목록을 수용할 수 있을 만큼 충분한 SQLVAR 어커런스와 함께 SQLDA를 할당하십시오. 테이블에 LOB, 구별 유형, 구조화된 유형 또는 참조 유형 컬럼이 포함되는 경우, SQLVAR 수는 최대 컬럼 수의 두 배여야 합니다. 그렇지 않은 경우에는 SQLVAR 수가 최대 컬럼 수와 같아야 합니다. 할당을 마친 후 응용프로그램은 이 SQLDA를 반복적으로 사용할 수 있습니다.

  이 기술은 특정 선택 목록에 대해 대부분의 스토리지가 사용되지 않은 경우에도 할당 해제되지 않은 많은 양의 스토리지를 사용합니다.

  두 번째 기술: 처리되는 모든 선택 목록에 대해 다음 두 단계를 반복하십시오.

1. SQLVAR 어커런스가 없는 SQLDA(즉, SQLN이 영(0)인 SQLDA)를 사용하여 DESCRIBE INPUT문을 실행하십시오. SQLD에 대해 리턴된 값은 결과 테이블에 있는 컬럼 수입니다. 이것은 SQLVAR 어커런스에 필수적인 수이거나 필요한 수의 반값입니다. SQLVAR 항목이 없으므로 경고가 발행됩니다(SQLSTATE 01005). 이러한 경고를 수반하는 SQLCODE가 +237, +238 또는 +239 중 하나이면 SQLVAR 항목 수는 SQLD에 리턴되는 값의 두 배여야 합니다. (이러한 양수 SQLCODE의 리턴은 SQLWARN 바인드 옵션 설정이 YES(양수 SQLCODE 리턴)라고 가정합니다.) SQLWARN이 NO로 설정된 경우에는 아직도 +238이 리턴되며 이는 SQLVAR 항목 수가 SQLD에 리턴되는 값의 두 배여야 함을 나타냅니다.
2. SQLVAR의 충분한 어커런스를 가지고 SQLDA를 할당하십시오. 그런 다음 이 새로운 SQLDA를 사용하여 다시 DESCRIBE문을 실행하십시오.

이 기술은 첫 번째 기술보다 향상된 스토리지 관리이지만, DESCRIBE INPUT문의 수가 두 배가 됩니다.

세 번째 기술: 대부분 및 거의 모든 선택 목록을 처리하기에 충분하지만 합리적으로 *크기*가 작은 SQLDA를 할당하십시오. DESCRIBE INPUT을 실행한 후 SQLD 값을 검사하십시오. SQLVAR의 어커런스 수에 대한 SQLD값을 사용하여 필요한 경우 더 큰 SQLDA를 할당하십시오.

이 기술은 첫 번째 두 기술 사이의 보완 기술에 해당합니다. 그 효과는 원래의 SQLDA에 대한 *크기*를 적절히 선택하는 것에 달려 있습니다.

예 :

PREPARE문이 가지는 입력 매개변수의 수를 설명하기에 충분한 SQLVAR 어커런스를 가지고 있는 SQLDA로 DESCRIBE INPUT문을 실행하십시오. 많아야 5개의 매개변수 표시문자가 설명하는 데 필요하고 입력 데이터가 LOB를 포함하지 않는다고 가정하십시오.

```
    /* STMT1_STR은 VALUES절이 있는 INSERT문을 포함합니다. */
EXEC SQL PREPARE STMT1_NAME FROM :STMT1_STR;
... /* code to set SQLN to 5 and to allocate the SQLDA        */
EXEC SQL DESCRIBE INPUT STMT1_NAME INTO :SQLDA;
.
.
.
```

이 예는 『DESCRIBE OUTPUT』의 『SQLDA 할당』에서 설명한 첫 번째 기술을 사용합니다.

DESCRIBE OUTPUT

DESCRIBE OUTPUT문은 준비된 명령문에 대한 정보를 얻습니다.

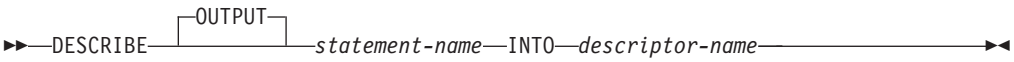
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

해당사항 없음.

구문



설명

statement-name

준비된 명령문을 식별합니다. DESCRIBE OUTPUT문이 실행될 때 이름은 현재 서버에서 응용프로그램 프로세스에 의해 준비된 명령문을 식별해야 합니다.

준비된 명령문이 SELECT 또는 VALUES INTO문인 경우 리턴된 정보는 결과 테이블의 컬럼을 설명합니다. 준비된 명령문이 CALL문인 경우 리턴된 정보는 프로시저의 출력 매개변수(OUT 또는 INOUT으로 정의된)를 설명합니다.

INTO *descriptor-name*

SQL 디스크립터 영역(SQLDA)을 식별합니다. DESCRIBE OUTPUT문을 실행하기 전에 SQLDA에 있는 다음 변수가 설정되어 있어야 합니다.

SQLN SQLDA에 제공되는 SQLVAR 어커런스 수를 지정합니다. SQLN이 0 이상의 값으로 설정되어야 DESCRIBE OUTPUT문이 실행됩니다.

DESCRIBE OUTPUT문이 실행될 때 데이터베이스 관리 프로그램이 다음과 같이 값을 SQLDA의 변수에 지정합니다.

SQLDAID

첫 번째 6바이트는 'SQLDA'로 설정되어 있습니다(즉, 5개 문자 뒤에 공백 문자로 구성).

SQLDOUBLED로 정의된 7번째 바이트는 설명된 결과 컬럼 또는 매개변수 표시문자를 기반으로 설정됩니다.

- SQLDA에 모든 컬럼 또는 출력 매개변수에 대해 두 개의 SQLVAR 항목을 포함하는 경우 7번째 바이트는 '2'로 설정됩니다. 이러한 기술은 LOB, 구별 유형, 구조화된 유형 또는 참조 유형 컬럼이나 출력 매개변수를 수용하는 데 사용됩니다.
- 그렇지 않은 경우 7번째 바이트는 공백 문자로 설정됩니다.

SQLDA에 모든 결과 컬럼 또는 출력 매개변수 표시문자의 설명을 포함하기에 충분한 공간이 없는 경우 7번째 바이트는 스페이스 문자로 설정됩니다.

8번째 바이트는 공백 문자로 설정됩니다.

SQLDABC

SQLDA의 길이(바이트)입니다.

SQLD 준비된 명령문이 SELECT인 경우 SQLD는 결과 테이블의 컬럼 수로 설정됩니다. 준비된 명령문이 CALL문인 경우 SQLD는 프로시저의 OUT 및 INOUT 매개변수 수로 설정됩니다. 그렇지 않은 경우 SQLD는 0으로 설정됩니다.

SQLVAR

SQLD 값이 0이거나 SQLN 값보다 큰 경우, SQLVAR 어커런스에 아무 값도 지정되지 않습니다.

SQLD 값이 *n*인 경우(여기서 *n*은 0보다 크지만 SQLN 값보다 작거나 같음) SQLVAR의 처음 *n* 어커런스에 대한 SQLTYPE, SQLLEN, SQLNAME, SQLLONGLEN 및 SQLDATATYPE_NAME에 값이 지정됩니다. 이 값은 결과 테이블의 값이나 프로시저의 출력 매개변수에 대한 매개변수 표시문자를 설명합니다. SQLVAR의 첫 번째 어커런스는 첫 번째 컬럼 또는 출력 매개변수 표시문자를 설명하고, SQLVAR의 두 번째 어커런스는 두 번째 컬럼 또는 출력 매개변수 표시문자를 설명합니다. 그 다음 어커런스도 마찬가지입니다.

기본 SQLVAR

SQLTYPE

컬럼 또는 매개변수의 데이터 유형과 널(NULL) 값을 포함할 수 있는지 여부를 보여 주는 코드입니다.

SQLLEN

컬럼 또는 매개변수의 데이터 유형에 따른 길이 값입니다. SQLLEN은 LOB 데이터 유형에 대해 0입니다.

SQLNAME

sqlname은 다음과 같이 파생됩니다.

- SQLVAR가 select문의 선택 목록에 있는 단순 컬럼 참조에 대해 파생된 컬럼과 일치하면 sqlname이 컬럼 이름입니다.

- SQLVAR이 프로시저의 매개변수 목록에 있고 표현식의 일부가 아닌 매개변수 표시문자에 해당되는 경우, CREATE PROCEDURE에 매개변수 이름이 지정되어 있으면 sqlname에 매개변수 이름이 포함됩니다.
- 그렇지 않으면 sqlname에 SQLDA 내에서 SQLVAR의 위치를 나타내는 ASCII 숫자 리터럴 값이 포함됩니다.

보조 SQLVAR

이러한 변수는 LOB, 구별 유형, 구조화된 유형 또는 참조 유형 컬럼이나 매개변수를 수용하도록 SQLVAR 항목 수가 두 배가 되는 경우에만 사용됩니다.

SQLLONGLEN

BLOB, CLOB 또는 DBCLOB 컬럼이나 매개변수의 길이 속성입니다.

SQLDATATYPE_NAME

사용자 정의 유형(구별 또는 구조화) 컬럼 또는 매개변수의 경우 데이터베이스 관리 프로그램이 이를 완전한 사용자 정의 유형 이름으로 설정합니다. 참조 유형 컬럼 또는 매개변수의 경우, 데이터베이스 관리 프로그램이 이를 참조 목표 유형의 완전한 사용자 정의 유형 이름으로 설정합니다. 그렇지 않을 경우, 스키마 이름은 SYSIBM이고 유형 이름은 SYSCAT.DATATYPES 카탈로그 뷰의 TYPENAME 컬럼 이름입니다.

주

- DESCRIBE OUTPUT문이 실행되기 전에, SQLN 값은 SQLDA에 SQLVAR 어커런스가 얼마나 많이 제공되어 있는지를 표시하도록 설정되어야 하고 SQLN 어커런스를 포함할 수 있는 충분한 스토리지가 할당되어야 합니다. 예를 들어, 준비된 SELECT문의 결과 테이블 컬럼에 대한 설명을 보려면 SQLVAR 어커런스 수가 컬럼 수보다 적으면 안됩니다.
- 큰 *크기*의 LOB이 예상되는 경우 이러한 대형 오브젝트(LOB)를 조작하면 응용프로그램 메모리에 영향을 줄 수 있다는 점을 기억하십시오. 이러한 조건을 고려할 때 로케이터나 파일 참조 변수를 고려하십시오. SQLLEN과 같은 다른 필드에 대한 해당 변경사항으로 SQL_TYP_xLOB의 SQLTYPE이 SQL_TYP_xLOB_LOCATOR 또는 SQL_TYP_xLOB_FILE로 변경되도록, DESCRIBE OUTPUT문이 실행된 후 스토리지를 할당하기 전에 SQLDA를 수정하십시오. 그런 다음 SQLTYPE에 기초한 스토리지를 할당한 후 계속 진행하십시오.
- 확장 UNIX 코드(EUC) 코드 페이지와 DBCS 코드 페이지 사이, 또는 유니코드 및 유니코드가 아닌 코드 페이지 사이의 코드 페이지 변환으로 문자 길이가 확장 및 축소될 수 있습니다.
- 구조화된 유형이 선택되지만 FROM SQL 변환은 지정되지 않은 경우(CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터를 사용하여 TRANSFORM

GROUP이 지정되지 않았거나(SQLSTATE 428EM) 이름이 지정된 그룹에 정의된 FROM SQL 변환 함수가 없어서(SQLSTATE 42744)), 오류가 리턴됩니다.

- **SQLDA 할당:** 다음은 SQLDA를 할당할 수 있는 세 가지 방법입니다.

  첫 번째 기술: 응용프로그램이 처리해야 할 선택 목록을 수용할 수 있을 만큼 충분한 SQLVAR 어커런스와 함께 SQLDA를 할당하십시오. 테이블에 LOB, 구별 유형, 구조화된 유형 또는 참조 유형 컬럼이 포함되는 경우, SQLVAR 수는 최대 컬럼 수의 두 배여야 합니다. 그렇지 않은 경우에는 SQLVAR 수가 최대 컬럼 수와 같아야 합니다. 할당을 마친 후 응용프로그램은 이 SQLDA를 반복적으로 사용할 수 있습니다.

  이 기술은 특정 선택 목록에 대해 대부분의 스토리지가 사용되지 않은 경우에도 할당 해제되지 않은 많은 양의 스토리지를 사용합니다.

  두 번째 기술: 처리되는 모든 선택 목록에 대해 다음 두 단계를 반복하십시오.

  1. SQLVAR 어커런스가 전혀 없는 SQLDA(즉, SQLN이 0인 SQLDA)로 DESCRIBE OUTPUT문을 실행하십시오. SQLD에 대해 리턴된 값은 결과 테이블에 있는 컬럼 수입니다. 이것은 SQLVAR 어커런스에 필수적인 수이거나 필요한 수의 반값입니다. SQLVAR 항목이 없으므로 경고가 발행됩니다(SQLSTATE 01005). 이러한 경고를 수반하는 SQLCODE가 +237, +238 또는 +239 중 하나이면 SQLVAR 항목 수는 SQLD에 리턴되는 값의 두 배여야 합니다. (이러한 양수 SQLCODE의 리턴은 SQLWARN 바인드 옵션 설정이 YES(양수 SQLCODE 리턴)라고 가정합니다.) SQLWARN이 NO로 설정된 경우에는 아직도 +238이 리턴되며 이는 SQLVAR 항목 수가 SQLD에 리턴되는 값의 두 배여야 함을 나타냅니다.
  2. SQLVAR의 충분한 어커런스를 가지고 SQLDA를 할당하십시오. 그런 다음 이 새로운 SQLDA를 사용하여 다시 DESCRIBE OUTPUT문을 실행하십시오.

  이 기술을 사용하면 첫 번째 기술보다 스토리지 관리를 더 잘할 수 있지만, DESCRIBE OUTPUT문의 수가 두 배가 됩니다.

  세 번째 기술: 대부분 및 거의 모든 선택 목록을 처리하기에 충분하지만 합리적으로 *크기*가 작은 SQLDA를 할당하십시오. DESCRIBE를 실행한 후 SQLD값을 검사하십시오. SQLVAR의 어커런스 수에 대한 SQLD값을 사용하여 필요한 경우 더 큰 SQLDA를 할당하십시오.

  이 기술은 첫 번째 두 기술 사이의 보완 기술에 해당합니다. 그 효과는 원래의 SQLDA에 대한 *크기*를 적절히 선택하는 것에 달려 있습니다.

- ***내재적으로 숨겨진 컬럼에 대한 고려사항:*** DESCRIBE OUTPUT문은 설명하는 쿼리의 최종 결과 테이블의 SELECT 목록 일부로 컬럼을 명시적으로 지정한 경우에만 내재적으로 숨겨진 컬럼에 대한 정보를 리턴합니다. 내재적으로 숨겨진 컬럼이 쿼리

의 결과 테이블 일부가 아니면 해당 쿼리에 대한 정보를 리턴하는 DESCRIBE OUTPUT문은 내재적으로 숨겨진 컬럼에 대한 정보를 포함하지 않습니다.

예 :

C 프로그램에서 SQLVAR 어커런스가 전혀 없는 SALDA를 사용하여 DESCRIBE OUTPUT문을 실행하십시오. SQLD이 0보다 큰 경우, 그 값을 사용하여 SQLVAR의 필요한 어커런스 수를 가진 SQLDA를 할당한 후, SQLDA를 사용하여 DESCRIBE문을 실행하십시오.

```
   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             */
    /* and allocate buffers to receive the data             */
   EXEC SQL  OPEN DYN_CURSOR;

... /* loop to fetch rows from result table                 */
   EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
.
.
.
```

DISCONNECT

DISCONNECT문은 사용 중인 작업 단위가 없을 때(즉, 커미트 또는 롤백 조작 후) 하나 이상의 연결을 삭제합니다. 단일 연결이 DISCONNECT문의 목표인 경우, 해당 연결은 사용 중인 작업 단위가 있더라도 데이터베이스가 기존의 작업 단위에 참여하는 경우에만 연결이 끊깁니다. 예를 들어, 몇 개의 다른 데이터베이스가 작업을 완료하였지만 문제의 대상이 완료되지 않은 경우, 연결을 삭제하지 않고 계속 연결이 끊어진 상태로 있을 수 있습니다.

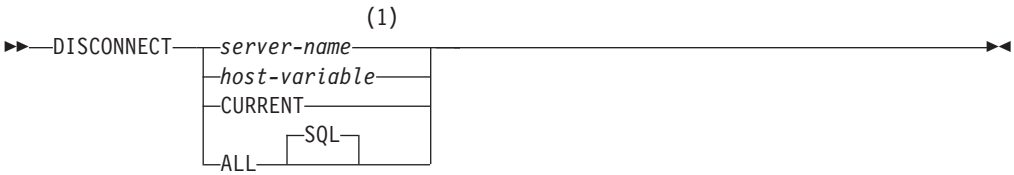
호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

필요한 권한 없음

구문



주:

1 응용프로그램 서버(AS) CURRENT 또는 ALL은 호스트 변수에 의해서만 식별됩니다.

설명

server-name 또는 *host-variable*

*server-name*이 들어 있는 *host-variable* 또는 지정된 *server-name*으로 응용프로그램 서버(AS)를 식별합니다.

*host-variable*을 지정할 경우 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안됩니다. *host-variable* 내에 들어 있는 *server-name*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안됩니다.

*server-name*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 로컬 디렉토리에 나열되어야 합니다.

지정된 데이터베이스 별명이나 호스트 변수에 포함된 데이터베이스 별명은 응용프로그램 프로세스의 기존 연결을 식별해야 합니다. 데이터베이스 별명이 기존 연결을 식별하지 않는 경우 오류(SQLSTATE 08003)가 발생합니다.

CURRENT
응용프로그램 프로세스의 현재 연결을 식별합니다. 응용프로그램 프로세스는 연결됨 상태여야 합니다. 그렇지 않으면, 오류(SQLSTATE 08003)가 발생합니다.

ALL
응용프로그램 프로세스의 모든 기존 연결이 삭제되었음을 표시합니다. 명령문이 실행될 때 어떤 연결도 없으면 오류 또는 경고가 발생하지 않습니다. 선택적 키워드 SQL은 RELEASE문 구문의 일관성을 위해 포함됩니다.

규칙

- 일반적으로, DISCONNECT문은 작업 단위(UOW) 내에 있는 동안 실행할 수 없습니다. 시도하면 오류(SQLSTATE 25000)가 발생합니다. 이 규칙의 예외는 단일 연결이 끊어지도록 지정되고 데이터베이스가 기존 작업 단위에 참여하지 않은 경우입니다. 이와 같은 경우, DISCONNECT문이 발행될 때 활성 작업 단위가 있어도 문제가 되지 않습니다.
- DISCONNECT문은 TP(Transaction Processing) 모니터 환경에서 실행될 수 없습니다(SQLSTATE 25000). SYNCPOINT 프리컴파일러 옵션이 TWOPHASE로 설정된 경우에 사용됩니다.

주

- DISCONNECT문이 성공하면 식별된 각 연결이 삭제됩니다.

  DISCONNECT문이 수행되지 않으면 응용프로그램 프로세스의 연결 상태 및 해당 연결 상태는 변경되지 않습니다.
- 현재 연결을 삭제하기 위해 DISCONNECT가 사용된 경우, 다음 실행되는 SQL문은 CONNECT 또는 SET CONNECTION이어야 합니다.
- 유형 1 CONNECT 시맨틱은 DISCONNECT 사용을 배제하지 않습니다. 그러나 DISCONNECT CURRENT 및 DISCONNECT ALL을 사용해도 CONNECT RESET 문이 수행하는 것과 같이 커미트 조작을 발생하지는 않습니다.

  server-name 또는 *host-variable*을 DISCONNECT문에 지정한 경우, 유형 1 CONNECT는 한 번에 하나의 연결만 지원하므로 현재 연결을 식별해야 합니다. 일반적으로, DISCONNECT는 "규칙"에 명시된 예외가 있는 작업 단위(UOW) 내에 있는 경우 실패합니다.
- 리모트 연결을 작성하고 유지보수하기 위해 자원이 필요합니다. 따라서 재사용되지 않을 리모트 연결은 가능한 한 바로 삭제해야 합니다.

- 연결 옵션이 적용되므로 커미트 조작 중에 연결도 삭제할 수 있습니다. 연결 옵션은 AUTOMATIC, CONDITIONAL 또는 EXPLICIT가 될 수 있으며, 이는 프리컴파일러 옵션으로 설정되거나 런타임 시 SET CLIENT API를 통해 설정될 수 있습니다. DISCONNECT 옵션 스펙에 대한 정보는 『분산 관계형 데이터베이스』를 참조하십시오.

예:

예 1: 응용프로그램에서 더 이상 IBMSTHDB와의 SQL 연결이 필요하지 않습니다. 다음 명령문은 커미트 또는 롤백 조작 후에 연결을 삭제하기 위해 실행해야 합니다.

```
EXEC SQL DISCONNECT IBMSTHDB;
```

예 2: 응용프로그램에서 더 이상 현재 연결이 필요하지 않습니다. 다음 명령문은 커미트 또는 롤백 조작 후에 연결을 삭제하기 위해 실행해야 합니다.

```
EXEC SQL DISCONNECT CURRENT;
```

예 3: 응용프로그램에서 더 이상 기존 연결이 필요하지 않습니다. 다음 명령문은 커미트 또는 롤백 조작 후에 모든 연결을 삭제하기 위해 실행해야 합니다.

```
EXEC SQL DISCONNECT ALL;
```

DROP

DROP문은 오브젝트를 삭제합니다. 그 오브젝트에 직접 또는 간접으로 의존하고 있는 모든 오브젝트는 삭제되거나 작동되지 않습니다. 오브젝트가 삭제될 때마다, 카탈로그에서 그 설명을 삭제하고 이 오브젝트를 참조하는 모든 패키지는 무효화됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

두 부분의 이름을 허용하는 오브젝트를 삭제할 때, 명령문의 권한 부여 ID에서 갖고 있는 특권에는 최소한 다음 중 하나가 포함되어야 합니다.

- 오브젝트의 스키마에 대한 DROPIN 특권
- 오브젝트에 대한 카탈로그 뷰의 소유자 컬럼에 기록된 오브젝트의 소유자
- 오브젝트에 대한 CONTROL 특권(인덱스, 인덱스 스펙, 별칭, 패키지, 테이블 및 뷰에만 해당)
- 카탈로그 뷰 SYSCAT.DATATYPES의 OWNER 컬럼에 기록된 사용자 정의 유형의 소유자(사용자 정의 유형과 연관된 메소드를 삭제(drop)하는 경우에만 해당)
- DBADM 권한

테이블이나 뷰 계층을 삭제(drop)할 때 명령문의 권한 부여 ID가 보유한 특권은 계층에 있는 테이블이나 뷰 각각에 대해 위의 특권 중 하나를 포함해야 합니다.

감사 규정을 삭제하면 명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

버퍼 풀, 데이터베이스 파티션 그룹 또는 테이블 스페이스를 삭제(drop)할 경우, 명령문의 권한 부여 ID에서 보유하는 특권에는 SYSADM 또는 SYSCTRL 권한이 포함되어야 합니다.

데이터 유형 맵핑, 함수 맵핑, 서버 정의 또는 랩퍼를 삭제하는 경우 명령문의 권한 부여 ID가 보유한 특권은 DBADM 권한을 포함해야 합니다.

이벤트 모니터를 삭제하는 경우 명령문의 권한 부여 ID가 보유하는 특권에는 SQLADM 또는 DBADM 권한이 포함되어야 합니다.

역할을 삭제할 때 명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

스키마를 삭제(drop)할 경우 명령문의 권한 부여 ID는 카탈로그 뷰 SYSCAT.SCHEMATA의 OWNER 컬럼에 기록된 스키마 소유자이거나 DBADM 권한을 가져야 합니다.

보안 레이블, 보안 레이블 구성요소 또는 보안 규정을 삭제할 때 명령문의 권한 부여 ID가 보유한 특권은 SECADM 권한을 포함해야 합니다.

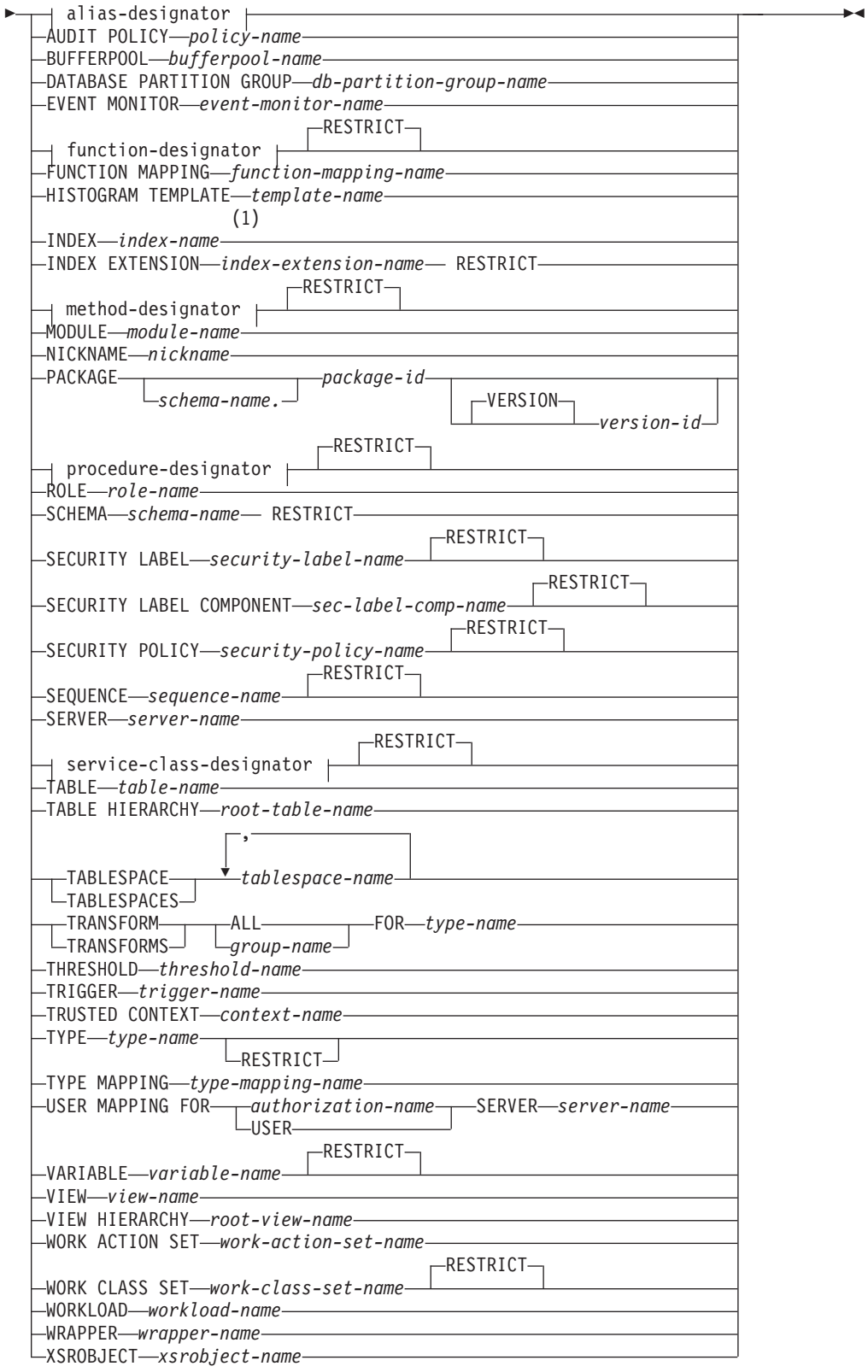
서비스 클래스, 작업 조치 세트, 작업 클래스 세트, 워크로드, 임계값 또는 막대 그래프 템플리트, 명령문의 권한 부여 ID에 의해 보유된 특권은 WLMADM 또는 DBADM 권한을 포함해야 합니다.

변환을 삭제(drop)할 때 명령문의 권한 부여 ID는 DBADM 권한을 포함하거나 *type-name*의 소유자여야 합니다.

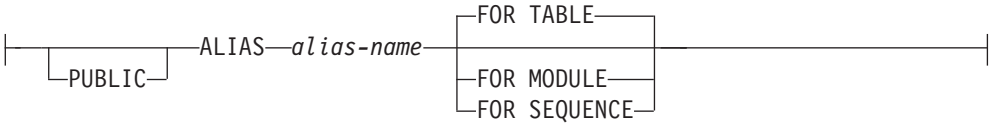
트러스트된 컨텍스트를 삭제하는 경우 명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

사용자 맵핑을 삭제(drop)할 때 이 권한 부여 ID가 맵핑 내의 페더레이티드 데이터베이스 권한 부여 이름과 다른 경우, 명령문의 권한 부여 ID는 DBADM 권한을 포함해야 합니다. 그렇지 않으면, 권한 부여 ID가 권한 부여 이름과 일치하는 경우 어떠한 권한이나 특권도 필요하지 않습니다.

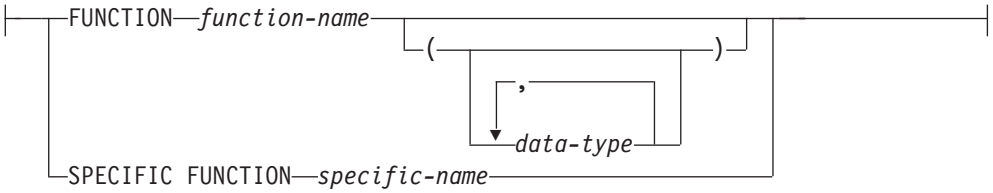
구문



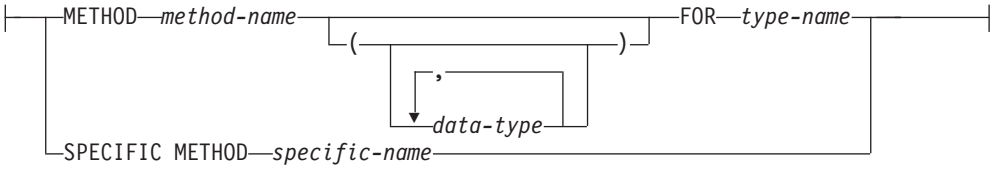
alias-designator:



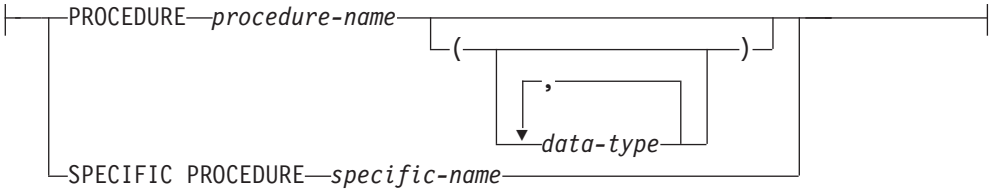
function-designator:



method-designator:



procedure-designator:



service-class-designator:

주:

1 *Index-name*은 인덱스 또는 인덱스 스펙의 이름일 수 있습니다.

설명

alias-designator

ALIAS *alias-name*

삭제될 별명을 식별합니다. *alias-name*은 카탈로그에서 설명되는 별명을 식별해야 합니다(SQLSTATE 42704). 지정된 별명은 삭제됩니다.

FOR TABLE, FOR MODULE 또는 FOR SEQUENCE

별명에 대한 오브젝트 유형을 지정합니다.

FOR TABLE
테이블, 뷰 또는 별칭에 대한 별명입니다.

FOR MODULE
모듈에 대한 별명입니다.

FOR SEQUENCE
시퀀스에 대한 별명입니다.

지정된 별명은 삭제됩니다. 별명을 참조하는 모든 테이블, 뷰 및 트리거는 작동 불능 상태가 됩니다. 여기에는 CREATE TRIGGER문의 ON절에서 참조된 테이블과 트리거 SQL문 내에서 참조된 모든 테이블이 포함됩니다.

PUBLIC이 지정되면 *alias-name*은 현재 서버에 존재하는 공용 별명을 식별해야 합니다(SQLSTATE 42704).

AUDIT POLICY *policy-name*
삭제될 감사 규정을 식별합니다. *policy-name*은 현재 서버에 존재하는 별명을 식별해야 합니다(SQLSTATE 42704). 감사 규정은 데이터베이스 오브젝트와 연관되어서는 안됩니다(SQLSTATE 42893). 지정된 감사 규정이 카탈로그에서 삭제됩니다.

BUFFERPOOL *bufferpool-name*
삭제될 버퍼 풀을 식별합니다. *bufferpool-name*은 카탈로그에 기술된 버퍼 풀을 식별해야 합니다(SQLSTATE 42704). 버퍼 풀에 지정된 테이블 스페이스가 있으면 안됩니다(SQLSTATE 42893). IBMDEFAULTBP 버퍼 풀은 삭제할 수 없습니다(SQLSTATE 42832). DB2에서 사용할 수 있도록 버퍼 풀 메모리가 즉시 해제됩니다. 디스크 스토리지는 다음에 데이터베이스에 연결할 때 해제됩니다.

DATABASE PARTITION GROUP *db-partition-group-name*
삭제될 데이터베이스 파티션 그룹을 식별합니다. *db-partition-group-name* 매개변수는 카탈로그에 설명된 데이터베이스 파티션 그룹을 식별해야 합니다(SQLSTATE 42704). 이 이름은 한 부분의 이름으로,

데이터베이스 파티션 그룹을 삭제하면 이 데이터베이스 파티션 그룹에 정의된 모든 테이블 스페이스가 삭제됩니다. 테이블 스페이스에 있는 테이블에 종속되는 기존의 모든 데이터베이스 오브젝트(예: 패키지, 참조 제한조건 등)가 삭제되거나 무효화되며(해당되는 경우) 종속 뷰 및 트리거를 사용할 수 없게 됩니다.

시스템 정의 데이터베이스 파티션 그룹은 삭제할 수 없습니다(SQLSTATE 42832).

DROP DATABASE PARTITION GROUP문이 현재 데이터 재분배에 영향을 받은 데이터베이스 파티션 그룹에 대해 발행될 경우 데이터베이스 파티션 그룹 삭제 연산은 실패하게 되고 오류를 리턴합니다(SQLSTATE 55038). 그러나 부분적으로 재분배된 데이터베이스 파티션 그룹은 삭제할 수 있습니다. REDISTRIBUTE DATABASE GROUP 명령이 완료를 실행하지 않으면 데이터베이스 파티션 그룹

은 부분적으로 재분배됩니다. 이는 오류나 FORCE APPLICATION ALL 명령으로 인터럽트될 경우에 발생할 수 있습니다. (부분적으로 재분배되는 데이터베이스 파티션 그룹의 경우 SYSCAT.DBPARTITIONGROUPS 카탈로그의 REBALANCE_PMAP_ID는 -1이 아닙니다.)

EVENT MONITOR *event-monitor-name*

삭제될 이벤트 모니터를 식별합니다. *event-monitor-name*은 카탈로그에 기술되어 있는 이벤트 모니터를 식별해야 합니다(SQLSTATE 42704).

식별된 이벤트 모니터가 활성화되면 오류가 리턴되고(SQLSTATE 55034), 활성화되지 않으면 이벤트 모니터가 삭제됩니다. 이벤트 모니터가 SET EVENT MONITOR STATE문을 사용하여 이전에 활성화되었고 데이터베이스가 비활성화된 후 다시 활성화되었으면, DROP문을 발행하기 전에 SET EVENT MONITOR STATE문을 사용하여 이벤트 모니터를 비활성화하십시오.

삭제 중인 WRITE TO FILE 이벤트 모니터의 목표 경로에 이벤트 파일이 있으면 이벤트 파일은 삭제되지 않습니다. 그러나 동일한 목표 경로를 지정하는 새 이벤트 모니터가 작성될 경우에는 이벤트 파일이 삭제됩니다.

WRITE TO TABLE 이벤트 모니터를 삭제할 때 테이블 정보는 SYSCAT.EVENTTABLES 카탈로그 뷰에서 제거되지만 테이블 자체는 삭제되지 않습니다.

function-designator

삭제될 사용자 정의 함수(완전한 함수 또는 함수 템플리트)의 인스턴스를 식별합니다. 지정된 함수 인스턴스는 카탈로그에 기술되어 있는 사용자 정의 함수여야 합니다. CREATE TYPE(구별)문에서 내재적으로 생성된 함수는 삭제할 수 없습니다.

다음과 같은 여러 가지 방법으로 함수 인스턴스를 식별할 수 있습니다.

FUNCTION *function-name*

특정 함수가 식별하여 *function-name*이 있는 함수 인스턴스가 하나인 경우에만 유효합니다. 그러므로 식별되는 함수는 이에 대해 정의된 임의 수의 매개변수를 가질 수 있습니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. 이름이 지정된 스키마 또는 내재적 스키마에 해당 이름의 함수가 없으면, 오류가 발생합니다(SQLSTATE 42704). 이름이 지정된 스키마 또는 내재적 스키마에 함수의 특정 인스턴스가 둘 이상 있을 경우, 오류가 발생합니다(SQLSTATE 42725).

FUNCTION *function-name* **(***data-type***,...)**

삭제될 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 선택 알고리즘은 사용하지 않습니다.

function-name

삭제될 함수의 함수 이름을 지정합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

(data-type,...)

해당 위치에 있는 CREATE FUNCTION문에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수 및 데이터 유형의 논리적 병합이 사용되어 삭제될 특정 함수 인스턴스를 식별합니다.

*data-type*이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해서는 정밀도나 스케일 및 길이를 지정하지 않아도 됩니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE FUNCTION 문에 지정된 값과 정확히 일치해야 합니다.

0<n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처의 함수가 이름이 지정된 스키마 또는 내재적 스키마에 존재하지 않는 경우 오류가 리턴됩니다(SQLSTATE42883).

SPECIFIC FUNCTION *specific-name*

함수 생성 시에 지정되거나 디폴트값으로 되는 특정 이름을 사용하여 삭제될 특정 사용자 정의 함수를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 함수 인스턴스를 식별해야 하며, 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

RESTRICT

RESTRICT 키워드는 다음 종속성 중 하나가 있으면 함수를 삭제하지 않는 규칙을 강제 실행합니다.

- 다른 루틴이 함수에서 전래된 경우

- 함수를 사용한 뷰
- 함수를 사용한 트리거
- 자체 정의에 있는 함수를 사용하는 구체화된 쿼리 테이블

RESTRICT가 디폴트 동작입니다.

SYSIBM, SYSFUN 또는 SYSPROC 스키마에 있는 함수를 삭제할 수 없습니다 (SQLSTATE 42832).

다른 오브젝트는 함수에 종속적일 수 있습니다. 그러한 모든 종속 오브젝트는 함수가 삭제되기 전에 작동 불능으로 표시된 패키지를 제외하고는 제거되어야 합니다. 그러한 종속 함수를 지닌 함수를 삭제하려 하면 오류(SQLSTATE 42893)가 발생합니다. 종속성 목록은 986 페이지의 『규칙』의 내용을 참조하십시오.

함수가 삭제 가능한 경우 삭제됩니다.

삭제 중인 특정 함수에 종속적인 패키지는 작동 불능으로 표시됩니다. 그러한 패키지는 내재적으로 리바인드되지 않습니다. BIND 또는 REBIND 명령을 사용하여 리바인드되거나 PREP 명령을 사용하여 다시 준비되어야 합니다.

FUNCTION MAPPING *function-mapping-name*

삭제될 함수 맵핑을 식별합니다. *function-mapping-name*은 카탈로그에서 설명되는 사용자 정의 함수(UDF) 맵핑을 식별해야 합니다(SQLSTATE 42704). 함수 맵핑이 데이터베이스에서 삭제됩니다.

디폴트 함수 맵핑은 삭제할 수 없지만 CREATE FUNCTION MAPPING문을 사용하여 비활성화할 수는 있습니다. 디폴트 함수 맵핑을 겹쳐쓰도록 작성된 사용자 정의 함수 맵핑을 삭제하면 디폴트 함수 맵핑을 다시 사용합니다.

삭제된 함수 맵핑에 종속성을 가지는 패키지는 무효화됩니다.

HISTOGRAM TEMPLATE *template-name*

삭제될 막대 그래프 템플리트를 식별합니다. *template-name*은 현재 서버에 존재하는 막대 그래프 템플리트를 식별해야 합니다(SQLSTATE 42704). *template-name*은 SYSDEFAULTHISTOGRAM일 수 없습니다(SQLSTATE 42832). 서비스 클래스나 작업 조치가 막대 그래프 템플리트에 종속되어 있으면 막대 그래프 템플리트를 삭제할 수 없습니다(SQLSTATE 42893). 지정된 막대 그래프 템플리트가 데이터베이스에서 삭제됩니다.

INDEX *index-name*

삭제될 인덱스나 인덱스 스펙을 식별합니다. *index-name*은 카탈로그에 기술되어 있는 인덱스 또는 인덱스 스펙을 식별해야 합니다(SQLSTATE 42704). 기본 키 또는 고유 제한조건이나 복제된 구체화된 쿼리 테이블 또는 XML 컬럼용으로 시스템이 요청하는 인덱스이여서는 안됩니다(SQLSTATE 42917). 지정된 인덱스나 인덱스 스펙이 삭제됩니다.

삭제된 인덱스나 인덱스 스펙에 종속성을 가지는 패키지는 무효화됩니다.

INDEX EXTENSION *index-extension-name* **RESTRICT**

삭제될 확장 인덱스를 식별합니다. *index-extension-name*은 카탈로그에 기술되어 있는 인덱스 확장을 식별해야 합니다(SQLSTATE 42704). RESTRICT 키워드는 이 인덱스 확장 정의에 따르는 인덱스를 정의할 수 없다는 규칙을 시행합니다(SQLSTATE 42893).

method-designator

삭제될 메소드 본문을 식별합니다. 지정된 메소드 본문은 카탈로그에 기술된 메소드여야 합니다(SQLSTATE 42704). 내재적으로 CREATE TYPE에서 생성한 메소드 본문은 삭제할 수 없습니다.

DROP METHOD는 메소드 본문을 삭제하지만 메소드 스펙(시그니처)은 주제 유형 정의의 한 부분으로 남습니다. 메소드 본문을 삭제한 후, ALTER TYPE DROP METHOD로 메소드 스펙을 주제 유형 정의에서 제거할 수 있습니다.

다음과 같은 여러 가지 방법으로 삭제될 메소드 본문을 식별할 수 있습니다.

METHOD *method-name*

삭제할 특정 메소드를 식별하며 이름은 *method-name*이고 주제 유형은 *type-name*인 메소드 인스턴스가 정확히 하나 있는 경우에만 유효합니다. 즉, 식별된 메소드는 임의 수의 매개변수를 가질 수 있습니다. *type-name* 유형에 대해 해당 이름의 메소드가 존재하지 않으면, 오류가 발생합니다(SQLSTATE 42704). 이름 지정된 데이터 유형에 메소드의 특정 인스턴스가 둘 이상 있을 경우, 오류가 발생합니다(SQLSTATE 42725).

METHOD *method-name* **(***data-type***,...)**

삭제될 메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 선택 알고리즘은 사용하지 않습니다.

method-name

지정된 유형에 대해 삭제될 메소드의 메소드 이름입니다. 이 이름은 규정되지 않은 ID이어야 합니다.

(*data-type***, ...)**

CREATE TYPE 또는 ALTER TYPE문의 메소드 스펙 중 해당 위치에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형 수와 데이터 유형의 논리적 병합을 사용하여 삭제될 특정 메소드 인스턴스를 식별합니다.

data-type이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다.

매개변수화된 데이터 유형에 대해 정밀도 또는 스케일, 길이를 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일이 코드화된 경우 그 값은 CREATE TYPE 문에 지정된 것과 정확히 일치해야 합니다.

0<n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 갖는 메소드가 이름이 지정된 데이터 유형에 대해 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

FOR *type-name*

지정된 메소드가 삭제될 유형을 이름 지정합니다. 이 이름은 카탈로그에 기술되어 있는 유형을 식별해야 합니다(SQLSTATE 42704). 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 유형 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 유형 이름의 규정자를 지정합니다.

SPECIFIC METHOD *specific-name*

CREATE TYPE 또는 ALTER TYPE 수행시 지정되었거나 디폴트값으로 설정된 이름을 사용하여 삭제될 특정 메소드를 식별합니다. 특정 이름이 규정되지 않은 경우 CURRENT SCHEMA 특수 레지스터는 동적 SQL에서 규정되지 않은 특정 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 특정 이름의 규정자를 지정합니다. specific-name은 메소드를 식별해야 하며, 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 42704).

RESTRICT

RESTRICT 키워드는 다음 종속성 중 하나가 있으면 메소드를 삭제하지 않는 규칙을 강제 실행합니다.

- 다른 루틴이 메소드에서 전래된 경우
- 메소드를 사용한 뷰
- 함수를 사용한 트리거

RESTRICT가 디폴트 동작입니다.

다른 오브젝트는 메소드에 종속적일 수 있습니다. 그러한 모든 종속 오브젝트는 메소드가 삭제되기 전에 작동 불능으로 표시된 패키지를 제외하고 제거해야 합니다. 그러한 종속 함수를 지닌 메소드를 삭제하려 하면 오류가 발생합니다(SQLSTATE 42893).

메소드를 삭제 가능한 경우 메소드는 삭제됩니다.

삭제 중인 특정 메소드에 종속적인 패키지는 작동 불능으로 표시됩니다. 그러한 패키지는 내재적으로 다시 바인드됩니다. 이 패키지는 BIND 또는 REBIND 명령을 사용하여 리바인드하거나 PREP 명령을 사용하여 다시 준비해야 합니다.

삭제되는 특정 메소드가 다른 메소드를 겹쳐쓸 경우, 겹쳐쓰여지는 메소드에 종속된 모든 패키지와 삭제되는 특정 메소드의 슈퍼 유형으로 이 메소드를 겹쳐쓸 메소드에 종속된 모든 패키지는 무효화됩니다.

MODULE *module-name*

삭제될 모듈을 식별합니다. *module-name*은 현재 서버에 존재하는 모듈을 식별해야 합니다 (SQLSTATE 42704). 지정된 모듈은 모든 모듈 오브젝트를 포함하여 스키마에서 삭제됩니다. 모듈에 대한 모든 특권도 삭제됩니다.

NICKNAME *nickname*

삭제될 별칭을 식별합니다. 별칭은 카탈로그에서 나열되어야 합니다(SQLSTATE 42704). 별칭이 데이터베이스에서 삭제됩니다.

별칭과 연관된 컬럼 및 인덱스에 대한 모든 정보가 카탈로그에서 삭제됩니다. 별칭에 종속되는 모든 구체화된 쿼리 테이블이 삭제됩니다. 별칭에 종속되는 모든 인덱스 스펙도 삭제됩니다. 별칭에 종속되는 모든 뷰는 작동하지 않는 것으로 표시됩니다. 삭제된 인덱스 스펙에 종속되는 모든 패키지 및 작동 불능 뷰가 무효화됩니다. 별칭이 참조하는 데이터 소스 테이블은 영향받지 않습니다.

SQL 함수 또는 메소드가 별칭에 종속된 경우 별칭을 삭제할 수 없습니다 (SQLSTATE 42893).

PACKAGE *schema-name.package-id*

삭제될 패키지를 식별합니다. 스키마 이름이 지정되지 않은 경우 디폴트 스키마가 패키지 ID를 내재적으로 규정합니다. 스키마 이름과 패키지 ID는 내재적으로 또는 명시적으로 지정된 버전 ID와 함께 카탈로그에 설명된 패키지를 식별해야 합니다 (SQLSTATE 42704). 지정된 패키지가 삭제됩니다. 삭제되는 패키지가 *schema-name.package-id*로 식별되는 유일한 패키지면(즉, 다른 버전이 없으면) 이 패키지에 대한 특권도 모두 삭제됩니다.

VERSION *version-id*

삭제될 패키지 버전을 식별합니다. 이 값을 지정하지 않은 경우, 버전은 빈 문자열로 디폴트됩니다. 패키지 이름은 같지만 버전이 다른 여러 개의 패키지가 있을 경우에는 DROP문을 한 번 호출할 때마다 한 개의 패키지 버전만 삭제할 수 있습니다. 큰따옴표를 사용하여 버전 ID를 구분하는 경우는 다음과 같습니다.

- VERSION(AUTO) 프리컴파일러 옵션을 사용하여 생성한 경우
- 숫자로 시작되는 경우
- 소문자 또는 대소문자 혼용 문자가 있는 경우

운영 체제 명령 프롬프트에서 명령문을 호출한 경우, 각 큰따옴표 분리문자 앞에 백슬래시 문자를 사용하여 운영 체제가 분리문자를 없애지 못하게 하십시오.

procedure-designator

삭제(drop)될 프로시저의 인스턴스를 식별합니다. 지정된 프로시저 인스턴스는 카탈로그에 기술되어 있는 프로시저야 합니다.

프로시저 인스턴스를 식별하는 데 사용할 수 있는 방법이 여러 가지 있습니다.

PROCEDURE *procedure-name*

삭제할 특정 프로시저를 식별하고 스키마 내에 *procedure-name*이 있는 프로시저 인스턴스가 하나인 경우에만 유효합니다. 그러므로 식별되는 프로시저에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 이름이 지정된 스키마나 내재적 스키마에 해당 이름의 프로시저가 존재하지 않으면 오류가 리턴됩니다(SQLSTATE 42704). 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. 이름 지정된 또는 내재적 스키마에 하나 이상의 특정 프로시저 인스턴스가 있는 경우 오류가 발생합니다(SQLSTATE 42725).

PROCEDURE *procedure-name* **(***data-type***,...)**

삭제될 프로시저를 고유하게 식별하는 프로시저 시그니처를 제공합니다. 프로시저 선택 알고리즘은 사용하지 않습니다. 페더레이티드 프로시저의 경우 서명 정보는 CREATE PROCEDURE문에 지정되지 않지만, 해당 정보는 시스템 카탈로그에서 사용할 수 있습니다.

procedure-name

삭제될 프로시저 이름을 제공합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다.

(data-type,...)

해당 매개변수에 대한 로컬 카탈로그에 저장된 데이터 유형과 일치해야 하는 페더레이티드 프로시저를 제외하고, 해당 위치에 있는 CREATE PROCEDURE문에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합을 사용하여 삭제될 특정 프로시저 인스턴스를 식별합니다.

*data-type*이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해서는 정밀도나 스케일 및 길이를 지정하지 않아도 됩니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일이 코딩된 경우 해당 값은 CREATE PROCEDURE문에 지정된 값과 정확히 일치해야 하고, 또는 페더레이티드 프로시저의 경우 해당 매개변수의 로컬 카탈로그에 저장된 값과 정확히 일치해야 합니다.

0<n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

이름이 지정되거나 내재적 스키마에 지정된 시그니처가 있는 프로시저가 존재하지 않는 경우 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC PROCEDURE *specific-name*

프로시저 생성시에 지정되거나 디폴트값으로 되는 특정 이름을 사용하여, 삭제(drop)될 특정 프로시저를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER precompile/bind 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 프로시저 인스턴스를 식별해야 하며, 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

RESTRICT

RESTRICT 키워드는 트리거 정의, SQL 함수 또는 SQL 메소드에 프로시저 이름과 함께 CALL문이 포함되어 있을 경우에는 프로시저가 삭제되지 않도록 합니다. RESTRICT가 디폴트 동작입니다.

SYSIBM, SYSFUN 또는 SYSPROC 스키마에 있는 프로시저는 삭제할 수 없습니다(SQLSTATE 42832).

ROLE *role-name*

삭제될 역할을 식별합니다. *role-name*은 현재 서버에 이미 존재하는 역할을 식별해야 합니다(SQLSTATE 42704). 역할이 루틴의 EXECUTE 특권 또는 시퀀스의 USAGE 특권을 포함하고 패키지가 아닌 SQL이 루틴이나 시퀀스에 종속되어 있으면 *role-name*은 역할, *role-name*을 포함하는 역할을 식별해서는 안됩니다(SQLSTATE 42893). SQL 오브젝트의 소유자는 *authorization-name* 또는 *authorization-name*의 구성원인 사용자이며, 여기서 *authorization-name*은 역할입니다.

삭제될 역할에 대해 다음 중 하나라도 만족할 경우 DROP ROLE문은 실패합니다(SQLSTATE 42893).

- 연결 속성 SESSION_USER ROLE에 대한 값 중 하나가 *role-name*이 되도록 워크로드가 존재합니다.
- *role-name*을 사용한 트러스트된 컨텍스트가 존재합니다.

지정된 역할이 카탈로그에서 삭제됩니다.

SCHEMA *schema-name* **RESTRICT**

삭제할 특정 스키마를 식별합니다. *schema-name*은 카탈로그에 기술되어 있는 스키마를 식별해야 합니다(SQLSTATE 42704). RESTRICT 키워드는 스키마가 데이터베이스에서 삭제되도록 지정된 스키마에 오브젝트를 정의할 수 없다는 규칙을 제시합니다(SQLSTATE 42893).

SECURITY LABEL *security-label-name*

삭제할 보안 레이블을 식별합니다. 이름은 보안 규정으로 규정해야 하며(SQLSTATE 42704) 현재 서버에 존재하는 보안 레이블을 식별해야 합니다(SQLSTATE 42704).

RESTRICT

디폴트값인 해당 옵션은 다음 종속성 중 하나 이상이 존재할 때 보안 레이블이 삭제되지 않도록 합니다(SQLSTATE 42893).

- 하나 이상의 권한부여 ID가 읽기 액세스에 대한 보안 레이블을 현재 보유합니다.
- 하나 이상의 권한부여 ID가 쓰기 액세스에 대한 보안 레이블을 현재 보유합니다.
- 보안 레이블이 하나 이상의 컬럼을 보호하기 위해 현재 사용되고 있습니다.

SECURITY LABEL COMPONENT *sec-label-comp-name*

삭제할 보안 레이블 구성요소를 식별합니다. *sec-label-comp-name*은 카탈로그에 기술된 보안 레이블 구성요소를 식별해야 합니다(SQLSTATE 42704).

RESTRICT

디폴트값인 해당 옵션은 다음 종속성 중 하나 이상이 존재할 때 보안 레이블 구성요소가 삭제되지 않도록 합니다(SQLSTATE 42893).

- 보안 레이블 구성요소를 포함하는 하나 이상의 보안 규정이 현재 정의되어 있음

SECURITY POLICY *security-policy-name*

삭제할 보안 규정을 식별합니다. *security-policy-name*은 현재 서버에 존재하는 보안 규정을 식별해야 합니다(SQLSTATE 42704).

RESTRICT

디폴트값인 해당 옵션은 다음 종속성 중 하나 이상이 존재할 때 보안 규정이 삭제되지 않도록 합니다(SQLSTATE 42893).

- 하나 이상의 테이블이 해당 보안 규정과 연결됨

- 하나 이상의 권한 부여 ID가 이 보안 규정의 규칙 중 하나에 대한 면제권을 보유합니다.
- 하나 이상의 보안 레이블이 이 보안 규정에 대해 정의됩니다.

SEQUENCE *sequence-name*

삭제할 특정 시퀀스를 식별합니다. *sequence-name*은 내재적 또는 명시적 스키마 이름과 함께 현재 서버의 기존 순서를 식별해야 합니다. 명시적 또는 내재적으로 지정된 스키마에 해당 이름을 가진 시퀀스가 없으면 오류가 리턴됩니다(SQLSTATE 42704).

RESTRICT

디폴트값인 해당 옵션은 다음 종속성 중 하나 이상이 존재할 때 시퀀스가 삭제되지 않도록 합니다.

- 트리거의 NEXT VALUE 또는 PREVIOUS VALUE 표현식이 시퀀스를 지정할 수 있도록 트리거가 존재합니다(SQLSTATE 42893).
- 루틴 본문의 NEXT VALUE 표현식이 시퀀스를 지정할 수 있도록 SQL 함수나 SQL 메소드가 존재합니다(SQLSTATE 42893).

SERVER *server-name*

그 정의가 카탈로그에서 삭제될 데이터 소스를 식별합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다(SQLSTATE 42704). 데이터 소스의 정의가 삭제됩니다.

데이터 소스에 있는 테이블 및 뷰에 대한 모든 별칭이 삭제됩니다. 이들 별칭에 종속된 모든 인덱스 스펙도 삭제됩니다. 삭제된 서버 정의에 종속되는 모든 사용자 정의 함수(UDF) 맵핑, 사용자 정의 유형 맵핑 및 사용자 맵핑 역시 삭제됩니다. 삭제된 서버 정의, 함수 맵핑, 별칭 및 인덱스 스펙에 종속되는 모든 패키지가 무효화됩니다. 서버 정의에 종속되는 모든 페더레이티드 프로시저도 삭제됩니다.

service-class-designator

SERVICE CLASS *service-class-name*

삭제될 서비스 클래스를 식별합니다. *service-class-name*은 카탈로그에서 설명되는 서비스 클래스를 식별해야 합니다(SQLSTATE 42704). 서비스 서브클래스를 삭제하려면, *service-superclass-name*이 UNDER절을 사용하여 지정되어야 합니다.

UNDER *service-superclass-name*

서비스 서브클래스를 삭제할 때 서비스 서브클래스의 서비스 수퍼 클래스를 지정합니다. *service-superclass-name*은 카탈로그에서 설명되는 서비스 수퍼 클래스를 식별해야 합니다(SQLSTATE 42704).

RESTRICT

이 키워드는 다음 종속성 중 하나가 있으면 서비스 클래스를 삭제하지 않는 규칙을 강제 실행합니다.

- 서비스 클래스는 서비스 수퍼 클래스이며 서비스 클래스 아래에 사용자 정의 서비스 서브 클래스가 있습니다(SQLSTATE 5U031). 먼저 서비스 서브클래스를 삭제해야 합니다.
- 서비스 클래스는 서비스 수퍼 클래스이며 서비스 클래스에 맵핑되는 작업 조치 세트가 있습니다(SQLSTATE 5U031). 먼저 작업 조치 세트를 삭제해야 합니다.
- 서비스 클래스는 서비스 서브클래스이며 서비스 클래스에 맵핑되는 작업 조치가 있습니다(SQLSTATE 5U031). 먼저 작업 조치를 삭제해야 합니다.
- 서비스 클래스는 워크로드 맵핑을 포함합니다(SQLSTATE 5U031). 먼저 워크로드 맵핑이 제거됩니다. 워크로드를 삭제하거나 서비스 클래스로 맵핑되지 않도록 워크로드를 변경하여 워크로드 맵핑을 제거합니다.
- 서비스 클래스는 연관된 임계값을 포함합니다(SQLSTATE 5U031). 먼저 임계값을 삭제해야 합니다.
- 서비스 클래스는 임계값에서의 REMAP ACTIVITY 조치의 목표입니다(SQLSTATE 5U031). REMAP ACTIVITY 조치의 목표로 다른 서비스 서브클래스로 임계값을 변경하거나 임계값을 삭제합니다.
- 서비스 클래스는 사용 불가능합니다(SQLSTATE 5U031). 먼저 서비스 클래스를 사용 불가능하게 해야 합니다.

RESTRICT가 디폴트 동작입니다.

TABLE *table-name*

삭제될 기본 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블을 식별합니다. *table-name*은 카탈로그에 기술된 테이블을 식별해야 하며 선언된 임시 테이블인 경우에는 *table-name*이 스키마 이름 SESSION으로 규정되고 응용프로그램에 존재해야 합니다(SQLSTATE 42704). 유형이 지정된 테이블의 서브테이블은 슈퍼 테이블에 종속됩니다. 모든 서브테이블을 삭제해야 슈퍼 테이블을 삭제할 수 있습니다(SQLSTATE 42893). 지정된 테이블이 데이터베이스에서 삭제됩니다.

테이블을 참조하는 모든 인덱스, 기본 키, 외부 키, 점검 제한조건, 구체화된 쿼리 테이블 및 스테이징 테이블이 삭제됩니다. 테이블을 참조하는 모든 뷰와 트리거가 작동 불능 상태가 됩니다. (여기에는 CREATE TRIGGER문의 ON절에서 참조된 테이블과 트리거 SQL문 내에서 참조된 모든 테이블이 포함됩니다.) 작동 불능으로 표시되거나 삭제된 오브젝트와 관련된 모든 패키지가 무효화됩니다. 여기에는 계층에서 서브테이블 위에 있는 슈퍼 테이블에 종속된 패키지가 포함됩니다. 삭제된 테이블이 참조 영역으로서 정의된 참조 컬럼의 영역이 해제됩니다.

패키지가 선언된 임시 테이블에 종속되지 않을 경우 테이블이 삭제될 때는 패키지가 무효화되지 않습니다. 이러한 테이블이 삭제될 때 패키지는 작성된 임시 테이블에 종속되며 무효화됩니다.

페더레이티드 시스템에서 투명한 DDL을 사용하여 작성된 리모트 테이블이 삭제될 수 있습니다. 리모트 테이블을 삭제하면 해당 테이블과 연관된 별칭도 삭제되고 해당 별칭에 종속된 모든 패키지도 무효화됩니다.

테이블 계층에서 서브테이블이 삭제되면 컬럼 수 및 행 크기 한계와 관련된 사항이라도 서브테이블과 연관된 컬럼에 더 이상 액세스할 수 없습니다. 서브테이블 삭제는 서브테이블의 모든 행을 슈퍼 테이블에서 삭제하는 데 영향을 미칩니다. 서브테이블 삭제 결과로 슈퍼 테이블에 정의된 트리거 또는 참조 무결성 제한조건이 활성화될 수 있습니다.

작성된 임시 테이블 또는 선언된 임시 테이블이 삭제되고 작업 단위 활성화 또는 세이브포인트에 앞서 작성되는 경우, 이 테이블은 기능적으로 삭제되고 응용프로그램이 테이블에 액세스할 수 없게 됩니다. 그러나 테이블은 아직 테이블 스페이스의 일부 스페이스를 차지하며 작업 단위가 커미트되거나 세이브포인트가 종료될 때까지 USER TEMPORARY 테이블 스페이스가 삭제되지 않도록 방지하고, USER TEMPORARY 테이블 스페이스의 데이터베이스 파티션 그룹이 재분배되지 않도록 방지합니다. 작성된 임시 테이블 또는 선언된 임시 테이블을 삭제하면 DROP의 커미트 또는 롤백 여부에 관계없이 테이블의 데이터가 파괴됩니다.

테이블에 RESTRICT ON DROP 속성이 있으면 테이블을 삭제할 수 없습니다.

새로 접속 해제된 테이블을 초기에 액세스할 수 없습니다. SET INTEGRITY문이 MQT를 증분 방식으로 새로 고치기 위해 또는 외부 키 제한조건에 대한 처리를 완료하기 위해 실행될 수 있을 때까지는 테이블을 읽거나 수정하거나 삭제하지 못하도록 합니다. SET INTEGRITY문이 모든 종속 테이블에 대해 실행된 후에는 테이블을 전체 액세스할 수 있고 접속 해제된 속성이 다시 설정되며 테이블을 삭제할 수 있습니다.

TABLE HIERARCHY *root-table-name*

삭제될 유형이 지정된 테이블 계층을 식별합니다. *root-table-name*은 유형이 지정된 테이블 계층에서 루트 테이블인 유형이 지정된 테이블을 식별해야 합니다 (SQLSTATE 428DR). *root-table-name*에 의해 식별된 유형이 지정된 테이블과 모든 서브테이블이 데이터베이스에서 삭제됩니다.

삭제된 테이블을 참조하는 모든 인덱스, 구체화된 쿼리 테이블, 스테이징 테이블, 기본 키, 외부 키 및 점검 제한조건이 삭제됩니다. 삭제된 테이블을 참조하는 모든 뷰 및 트리거가 작동 불능으로 됩니다. 작동 불능으로 표시되거나 삭제된 오브젝트와 관련된 모든 패키지가 무효화됩니다. 삭제된 테이블 중 하나에 대한 영역으로 정의된 모든 참조 컬럼의 영역이 해제됩니다.

하나의 테이블을 삭제하는 것과 달리, 테이블 계층을 삭제하면 계층에 있는 테이블의 삭제 트리거를 활성화시키지도 않고 삭제된 행을 로그하지도 않습니다.

TABLESPACE 또는 TABLESPACES *tablespace-name*

삭제될 테이블 스페이스를 식별하며, *tablespace-name*은 카탈로그에 기술된 테이블 스페이스를 식별해야 합니다(SQLSTATE 42704). 이 이름은 한 부분의 이름입니다.

삭제 중인 테이블 스페이스에 일부분을 저장했고 삭제되지 않는 다른 테이블 스페이스에 테이블의 일부분이 있거나 테이블 스페이스에 있는 테이블 중 RESTRICT ON DROP 속성이 있는 테이블이 있으면 테이블 스페이스는 삭제되지 않습니다(SQLSTATE 55024).

'SYS'가 접두어인 오브젝트는 시스템 정의 오브젝트인데 SYSTOOLSPACE 및 SYSTOOLSTMPSPACE 테이블 스페이스인 경우를 제외하고는 삭제될 수 없습니다(SQLSTATE 42832).

SYSTEM TEMPORARY 테이블 스페이스는 데이터베이스에 임시 테이블 스페이스만 있는 경우에는 삭제할 수 없습니다(SQLSTATE 55026). 사용자 임시 테이블 스페이스는 그 안에 작성된 임시 테이블이나 선언된 임시 테이블이 작성된 경우 삭제할 수 없습니다(SQLSTATE 55039). 작성된 임시 테이블이 삭제된 경우에도 USER TEMPORARY 테이블 스페이스는 작성된 임시 테이블의 모든 인스턴스가 삭제될 때까지 여전히 사용 중인 것으로 간주됩니다. 세션이 종료되거나 작성된 임시 테이블이 세션에서 참조되면 작성된 임시 테이블의 인스턴스가 삭제됩니다. 선언된 임시 테이블이 삭제되었을지라도 USER TEMPORARY 테이블 스페이스는 DROP TABLE문을 포함하는 작업 단위가 커미트될 때까지 여전히 사용 중인 것으로 간주됩니다.

테이블 스페이스를 삭제하면 테이블 스페이스에 정의된 모든 오브젝트가 삭제됩니다. 패키지, 참조 제한조건 등과 같이 테이블 스페이스에서 종속성이 있는 기존의 모든 데이터베이스 오브젝트가 삭제되거나 무효화되며(해당되는 경우), 종속 뷰 및 트리거는 작동되지 않게 됩니다.

사용자가 작성한 컨테이너는 삭제되지 않습니다. CREATE TABLESPACE 실행 중에 데이터베이스 관리 프로그램이 작성한 컨테이너 이름의 경로에 있는 디렉토리는 삭제됩니다. 데이터베이스 디렉토리 아래의 모든 컨테이너도 삭제됩니다. DROP TABLESPACE문이 커미트되는 경우, 가능하면 지정된 테이블 스페이스에 해당하는 DMS 파일 컨테이너 또는 SMS 컨테이너는 삭제됩니다. 컨테이너가 삭제될 수 없는 경우(예: 다른 에이전트의 사용으로 열려 있는 경우) 파일은 제로 길이로 절단됩니다. 모든 연결이 종료된 후 또는 DEACTIVATE DATABASE 명령어가 발행된 후에, 길이가 0인 파일이 삭제됩니다.

THRESHOLD *threshold-name*

삭제될 임계값을 식별합니다. *threshold-name*은 현재 서버에 존재하는 임계값을 식별해야 합니다(SQLSTATE 42704). 이 이름은 한 부분의 이름으로, 예를 들어 TOTALSCPARTITIONCONNECTIONS 및

CONCURRENTDBCOORDACTIVITIES와 같이 큐가 포함된 임계값은 삭제되기 전에 사용 불가능하게 해야 합니다(SQLSTATE 5U025). 지정된 임계값이 카탈로그에서 삭제됩니다.

TRIGGER *trigger-name*

삭제될 트리거를 식별합니다. *trigger-name*은 카탈로그에 기술되어 있는 트리거를 식별해야 합니다(SQLSTATE 42704). 지정된 트리거가 삭제됩니다.

트리거를 삭제하면 일부 패키지가 유효하지 않은 것으로 표시됩니다.

*trigger-name*이 뷰에 INSTEAD OF 트리거를 지정하면 다른 트리거가 뷰에 대한 갱신을 통해 이 트리거에 의존할 수 있습니다.

TRANSFORM ALL FOR *type-name*

사용자 정의 데이터 유형 *type-name*에 대해 정의된 모든 변환 그룹이 삭제될 것임을 나타냅니다. 이러한 그룹에서 참조되는 변환 함수는 삭제되지 않습니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. *type-name*은 카탈로그에서 설명되는 사용자 정의 유형을 식별해야 합니다(SQLSTATE 42704).

*type-name*에 대해 정의된 변환이 없는 경우 오류가 리턴됩니다(SQLSTATE 42740).

DROP TRANSFORM은 CREATE TRANSFORM의 반대입니다. 이 명령은 지정된 데이터 유형의 특정 그룹과 연관된 변환 함수를 정의되지 않은 상태로 변경합니다. 이러한 그룹과 연관된 함수는 아직 존재하며 여전히 명시적으로 호출할 수 있으나, 더 이상 변환 등록 정보를 가지지 않으므로 호스트 언어 환경과의 값 교환을 위해 내재적으로 호출되지 않습니다.

변환 그룹은 사용자 정의 유형 *type-name*에 대해 정의된 그룹의 변환 함수 중 하나에 종속되며 SQL이 아닌 다른 언어로 작성된 사용자 정의 함수(UDF)가 있는 경우 삭제되지 않습니다(SQLSTATE 42893). 이 함수는 유형 *type-name*에 대해 정의된 참조된 변환 그룹과 연관되는 변환 함수에 종속됩니다. 이름 지정된 변환 그룹과 연관된 변환 함수에 종속되는 패키지는 작동 불능으로 표시됩니다.

TRANSFORMS *group-name* **FOR** *type-name*

사용자 정의 데이터 유형 *type-name*에 대해 지정된 변환 그룹이 삭제됨을 나타냅니다. 이러한 그룹에서 참조되는 변환 함수는 삭제되지 않습니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서는 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. *type-name*은 카탈로그에 기술된 사용자 정의 유형을 식별해야 하고(SQLSTATE 42704), *group-name*은 *type-name*에 대한 기존의 변환 그룹을 식별해야 합니다.

TRIGGER *trigger-name*

삭제될 트리거를 식별합니다. *trigger-name*은 카탈로그에 기술되어 있는 트리거를 식별해야 합니다(SQLSTATE 42704). 지정된 트리거가 삭제됩니다.

트리거를 삭제하면 일부 패키지가 유효하지 않은 것으로 표시됩니다.

*trigger-name*이 뷰에 INSTEAD OF 트리거를 지정하면 다른 트리거가 뷰에 대한 갱신을 통해 이 트리거에 의존할 수 있습니다.

TRUSTED CONTEXT *context-name*

삭제될 트러스트된 컨텍스트를 식별합니다. *context-name*은 현재 서버에 존재하는 트러스트된 컨텍스트를 식별해야 합니다(SQLSTATE 42704). 이 컨텍스트에 대한 트러스트된 연결이 활성화되는 동안 트러스트된 컨텍스트가 삭제되면 연결이 종료되거나 다음 재사용이 시도될 때까지 연결은 트러스트된 채로 있습니다. 이들 트러스트된 연결에서 사용자를 전환하려고 시도하면 오류가 리턴됩니다(SQLSTATE 42517). 지정된 트러스트된 컨텍스트가 카탈로그에서 삭제됩니다.

TYPE *type-name*

삭제할 사용자 정의 유형을 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. 구조화된 유형의 경우, 관련 참조 유형도 삭제됩니다. *type-name*은 카탈로그에 기술된 사용자 정의 유형을 식별해야 합니다.

RESTRICT

다음 사항 중 하나라도 만족할 경우 유형은 삭제되지 않습니다(SQLSTATE 42893).

- 테이블 또는 뷰 컬럼 유형으로 사용되는 경우
- 부속 유형이 있는 경우
- 유형이 지정된 테이블 또는 유형이 지정된 뷰의 데이터 유형으로 사용된 구조화된 유형인 경우
- 유형이 또 다른 구조화된 유형의 속성인 경우
- 유형이 *type-name*의 인스턴스를 포함할 수 있는 테이블의 컬럼이 있는 경우. 이는 *type-name*이 컬럼의 유형이거나 컬럼의 관련 자료형 계층에서 사용되는 경우 발생할 수 있습니다. T 유형의 경우 T는 그 유형이 직접 또는 간접적으로 *type-name*을 사용하는 테이블의 컬럼이 있으면 삭제될 수 없습니다.
- 유형이 테이블 또는 뷰의 참조 유형 컬럼의 목표 유형이거나 또 다른 구조화된 유형의 참조 유형 속성인 경우
- 유형이나 해당 유형에 대한 참조가 함수 또는 메소드의 매개변수 유형이나 리턴 값 유형인 경우
- 유형이 매개변수 유형이거나 SQL 프로시저의 본문에서 사용된 경우

- 유형이나 해당 유형에 대한 참조가 SQL 함수 또는 메소드의 본문에 사용되지만 매개변수 유형이나 리턴 값 유형이 아닌 경우
- 유형이 점검 제한조건, 트리거, 뷰 정의 또는 확장 인덱스에 사용된 경우

RESTRICT를 지정하지 않은 경우 그 유형을 사용하는 함수와 메소드를 제외하고 동작은 RESTRICT와 같습니다.

해당 유형을 사용하는 함수: 사용자 정의 유형을 삭제할 수 있는 경우, 삭제 중인 유형 또는 삭제 중인 유형에 대한 참조의 매개변수나 리턴 값을 가지는 모든 함수 F(특정 이름 SF)에 대해 다음의 DROP FUNCTION문이 효과적으로 실행됩니다.

```
DROP SPECIFIC FUNCTION SF
```

이 명령문 또한 종속 함수를 삭제하기 위해 연쇄시키는 것이 가능합니다. 이들 모든 함수가 사용자 정의 유형의 종속성으로 인해 삭제될 목록에도 있는 경우, 사용자 정의 유형도 삭제됩니다. (그렇지 않은 경우 SQLSTATE 42893로 실패합니다.)

해당 유형을 사용하는 메소드: 사용자 정의 유형을 삭제할 수 있는 경우, 삭제 중인 유형 또는 삭제 중인 유형에 대한 참조의 매개변수나 리턴 값을 가지는 유형이 T1인 모든 메소드 M(특정 이름 SM)에 대해 다음의 명령문이 효과적으로 실행됩니다.

```
DROP SPECIFIC METHOD SM
ALTER TYPE T1 DROP SPECIFIC METHOD SM
```

이러한 메소드에 종속되는 오브젝트가 있으면 DROP TYPE 조작이 실패할 수 있습니다.

삭제될 유형은 슈퍼 유형으로 정의된 메소드에 의존하므로 겹쳐쓸 수 있는 모든 패키지가 무효화됩니다.

TYPE MAPPING *type-mapping-name*
삭제할 사용자 정의 데이터 유형 맵핑을 식별합니다. *type-mapping-name*은 카탈로그에 기술되어 있는 데이터 유형 맵핑을 식별해야 합니다(SQLSTATE 42704). 데이터 유형 맵핑이 데이터베이스에서 삭제됩니다.

추가 오브젝트는 삭제되지 않습니다.

USER MAPPING FOR *authorization-name* **| USER SERVER** *server-name*
삭제될 사용자 맵핑을 식별합니다. 이 맵핑은 페더레이티드 데이터베이스를 액세스하기 위해 사용되는 권한 부여 이름을 데이터 소스를 액세스하기 위해 사용된 권한 부여 이름과 연관시킵니다. 이들 두 권한 부여 이름의 첫 번째는 *authorization-name*으로 식별되거나 특수 레지스터 USER에 의해 참조됩니다. *server-name*은 액세스하기 위해 두 번째 권한 부여 이름이 사용되는 데이터 소스를 식별합니다.

*authorization-name*은 카탈로그에 나열되어야 합니다(SQLSTATE 42704). *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다(SQLSTATE 42704). 사용자 맵핑이 삭제됩니다.

추가 오브젝트는 삭제되지 않습니다.

VARIABLE *variable-name*

삭제될 전역 변수를 식별합니다. *variable-name*은 현재 서버에 존재하는 전역 변수를 식별해야 합니다(SQLSTATE 42704).

RESTRICT

이 키워드는 함수, 메소드, 트리거 또는 뷰에서 참조되면 전역 변수를 삭제할 수 없는 규칙을 강제 실행합니다(SQLSTATE 42893). RESTRICT가 디폴트 동작입니다.

VIEW *view-name*

삭제될 뷰를 식별합니다. *view-name*은 카탈로그에 기술되어 있는 뷰(SQLSTATE 42704)를 식별해야 합니다. 유형이 지정된 뷰의 서브뷰는 슈퍼 뷰에 종속됩니다. 모든 서브뷰를 삭제해야 슈퍼 뷰를 삭제할 수 있습니다(SQLSTATE 42893).

지정된 뷰가 삭제됩니다. 이 뷰에 직접 또는 간접으로 종속적인 뷰 또는 트리거의 정의는 작동되지 않는 것으로 표시됩니다. 작동 불능으로 표시된 뷰에 종속적인 구체화된 쿼리 테이블이 삭제됩니다. 작동 불능으로 표시되거나 삭제되는 뷰에 종속적인 패키지는 무효화됩니다. 여기에는 계층에서 서브뷰 위에 있는 슈퍼 뷰에 종속된 패키지가 포함됩니다. 삭제된 뷰가 참조 영역으로서 정의된 참조 컬럼의 영역이 해제됩니다.

VIEW HIERARCHY *root-view-name*

삭제될 유형이 지정된 뷰 계층을 식별합니다. *root-view-name*은 유형이 지정된 뷰 계층에서 루트 뷰인 유형이 지정된 뷰를 식별해야 합니다(SQLSTATE 428DR). *root-view-name*에 의해 식별된 유형이 지정된 뷰와 모든 서브뷰가 데이터베이스에서 삭제됩니다.

삭제된 뷰에 직접 또는 간접으로 종속적인 뷰 또는 트리거의 정의는 작동 불능으로 표시됩니다. 삭제되거나 작동 불능으로 표시되는 뷰나 트리거에 종속적인 패키지는 무효화됩니다. 삭제된 뷰나 작동 불능으로 표시된 뷰에 대한 참조 영역으로서 정의된 참조 컬럼의 영역이 해제됩니다.

WORK ACTION SET *work-action-set-name*

삭제될 트러스트된 작업 조치를 식별합니다. *work-action-set-name*은 현재 서버에 존재하는 작업 조치 세트를 식별해야 합니다(SQLSTATE 42704). *work-action-set-name*에서 포함된 모든 작업 조치도 삭제됩니다.

WORK CLASS SET *work-class-set-name*

삭제될 작업 클래스 세트를 식별합니다. *work-class-set-name*은 현재 서버에 존재하는 작업 클래스 세트를 식별해야 합니다(SQLSTATE 42704). *work-class-set-name*에서 포함된 모든 작업 클래스도 삭제됩니다.

RESTRICT

이 키워드는 작업 조치 세트와 연관되면 작업 클래스 세트를 삭제하지 않는 규칙을 강제 실행합니다. RESTRICT가 디폴트 동작입니다.

WORKLOAD *workload-name*

삭제될 워크로드를 식별합니다. 이 이름은 한 부분의 이름입니다. *workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). SYSDEFAULTUSERWORKLOAD 또는 SYSDEFAULTADMWORKLOAD를 삭제할 수 없습니다(SQLSTATE 42832). 워크로드를 삭제하기 전에 워크로드가 사용 불가능해야 하며 워크로드와 연관된 활성 워크로드 어커런스가 없어야 합니다(SQLSTATE 5U023). 지정된 워크로드는 데이터베이스에서 삭제됩니다.

WRAPPER *wrapper-name*

삭제될 랩퍼를 식별합니다. *wrapper-name*은 카탈로그에 기술된 랩퍼를 식별해야 합니다(SQLSTATE 42704). 랩퍼가 삭제됩니다.

랩퍼에 종속되는 모든 서버 정의, 사용자 정의 함수(UDF) 맵핑 및 사용자 정의 데이터 유형 맵핑이 삭제됩니다. 삭제된 서버 정의에 종속되는 모든 사용자 정의 함수(UDF) 맵핑, 별칭, 사용자 정의 데이터 유형 맵핑 및 사용자 맵핑 역시 삭제됩니다. 삭제된 별명에 종속된 모든 인덱스 스펙이 삭제되고, 이들 별칭에 종속된 모든 뷰가 작동 불능으로 표시됩니다. 삭제된 오브젝트에 종속된 모든 패키지 및 작동 불능 뷰가 무효화됩니다. 삭제된 서버 정의에 종속되는 모든 페더레이티드 프로시저도 삭제됩니다.

XSROBJECT *xsrobject-name*

삭제할 XSR 오브젝트를 식별합니다. *xsrobject-name*은 카탈로그에 기술된 XSR 오브젝트를 식별해야 합니다(SQLSTATE 42704).

XSR 오브젝트를 참조하는 점검 제한조건이 삭제됩니다. XSR 오브젝트를 참조하는 모든 트리거와 뷰는 작동 불능으로 표시됩니다. 삭제된 XSR 오브젝트에 종속된 패키지는 무효화됩니다.

파티션된 데이터베이스 환경에서 임의의 파티션에 연결하여 XSR 오브젝트에 대해 이 명령문을 실행할 수 있습니다.

규칙

종속성: 988 페이지의 표 28에서 오브젝트 간의 종속성을 나타냅니다. 모든 종속성이 카탈로그에 명시적으로 기록되지는 않습니다. 예를 들어, 패키지가 종속되는 제한조건은 기록되지 않습니다. 종속성에는 다음 네 가지가 있습니다.

R 제한 시맨틱. 하위 오브젝트는 이에 종속되는 오브젝트가 존재하는 한 삭제될 수 없습니다.

C 연쇄 시맨틱. 하위 오브젝트를 삭제하면 이에 종속되는 오브젝트(종속 오브젝트)

역시 삭제됩니다. 그러나 종속 오브젝트가 일부 다른 오브젝트상에서 제한 종속성으로 인해 삭제될 수 없는 경우, 하위 오브젝트 삭제는 실패하게 됩니다.

X 비작동 시맨틱. 하위 오브젝트를 삭제하면, 이 오브젝트에 종속된 오브젝트가 사용할 수 없게 됩니다. 사용자가 명시적인 특정 조치를 취할 때까지 작동 불능 상태가 됩니다.

A 자동 무효화 및 유효성 다시 확인 시맨틱. 하위 오브젝트를 삭제하면 이 오브젝트에 종속된 오브젝트가 유효하지 않게 됩니다. 데이터베이스 관리 프로그램은 유효하지 않은 오브젝트의 유효성을 다시 확인하려고 합니다.

함수나 메소드 또는 함수나 메소드에서 직간접적으로 호출한 프로시저가 사용하는 패키지는 루틴이 MODIFIES SQL DATA로 정의된 경우에만 자동으로 유효성이 다시 확인됩니다. 루틴이 MODIFIES SQL DATA가 아니면 오류가 발생합니다(SQLSTATE 56098).

데이터베이스 구성 매개변수 **auto_reval**이 IMMEDIATE 또는 DEFERRED로 설정될 때 988 페이지의 표 28에 표시된 일부 종속성이 『A』(자동 무효화/유효성 다시 확인 시맨틱)로 변경됩니다. 994 페이지의 표 29는 영향을 받는 종속 오브젝트를 요약합니다. 『명령문』 컬럼에 나열된 해당 명령문이 실행될 때 『영향을 받는 종속 오브젝트』 컬럼에 나열된 오브젝트를 무효화합니다.

일반적으로 다음에 오브젝트가 사용될 때 데이터베이스 관리 프로그램은 유효하지 않은 오브젝트의 유효성을 다시 확인하려고 합니다. 그러나 **auto_reval**이 IMMEDIATE로 설정되어 있는 상황에서는 영향을 받는 종속 오브젝트가 무효화된 다음 즉시 다시 유효성이 확인됩니다. 가능한 상황은 다음과 같습니다.

- ALTER TABLE ... ALTER COLUMN
- ALTER TABLE ... DROP COLUMN
- ALTER TABLE ... RENAME COLUMN
- ALTER TYPE ... ADD ATTRIBUTE
- ALTER TYPE ... DROP ATTRIBUTE
- 『OR REPLACE』를 지정하는 CREATE문

일부 DROP문 매개변수 및 오브젝트는 공백 행이나 컬럼의 결과를 낳을 수도 있기 때문에 988 페이지의 표 28에는 표기되지 않았습니다.

- EVENT MONITOR, PACKAGE, PROCEDURE, SCHEMA, TYPE MAPPING 및 USER MAPPING DROP문에는 오브젝트 종속성이 없습니다.
- 별명, 버퍼 풀, 분산 키, 특권 및 프로시저 오브젝트 유형에는 DROP문 종속성이 없습니다.
- A DROP SERVER, DROP FUNCTION MAPPING 또는 DROP TYPE MAPPING문에는 다음 조건에서 처리될 수 없는 작업 단위(UOW)가 부여됩니다.

– 명령문이 단일 데이터 소스를 참조하여 UOW가 이미 데이터 소스 내의 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문을 포함합니다(SQLSTATE 55006).
– 명령문이 데이터 소스의 범주를 참조하여(예: 특정 유형 및 버전의 모든 데이터 소스), UOW가 이미 이들 데이터 소스 중 하나 안에 있는 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문을 포함합니다(SQLSTATE 55006).

표 28. 종속성

	오브젝트 유형																								
명령문	CONSTRAINT	FUNCTION MAPPING	FUNCTION	GLOBAL VARIABLE	INDEX EXTENSION	INDEX	MATERIALIZED QUERY TABLE	NICKNAME	PACKAGE	PROCEDURE	ROLE	SEQUENCE	SERVER OPTION	TABLE	TRIGGER	TYPE MAPPING	USER MAPPING	VIEW	WORKLOAD	WRAPPER	XSROBJECT
ALTER FUNCTION	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER METHOD	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER NICKNAME, 로컬 이름 또는 로컬 유형 변경	R[33]	R	-	-	-	-	R	-	-	A	-	-	R	-	-	-	-	-	-	R	-	-	-	-	-
ALTER NICKNAME, 컬럼 옵션 또는 별칭 옵션 변경	-	-	-	-	-	-	-	-	-	A	-	-	R	-	-	-	-	-	-	-	-	-	-	-	-
ALTER NICKNAME, 제한조건 추가, 변경 또는 삭제	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER PROCEDURE	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER SERVER	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TABLE ALTER COLUMN	-	A	-	A	-	-	-	-	-	A	-	-	-	-	-	A	-	-	-	A	-	-	-	X[34]
ALTER TABLE DROP COLUMN	C	C	-	C	C	-	-	-	-	-	-	-	-	-	-	C	-	-	-	C	-	-	-	X[34]
ALTER TABLE DROP CONSTRAINT	C	-	-	-	-	-	-	-	-	A[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TABLE DROP PARTITIONING KEY	-	-	-	-	-	-	-	-	R[20]	A[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TYPE ADD ATTRIBUTE	-	-	-	-	-	R	-	-	-	A[23]	-	-	R[24]	-	-	-	-	-	-	R[14]	-	-	-	-
ALTER TYPE ALTER METHOD	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-

표 28. 종속성 (계속)

	오브젝트 유형																								
명령문	CONSTRAINT	CONSTRAINT FUNCTION MAPPING	DATABASE PARTITION GROUP	FUNCTION	GLOBAL VARIABLE	INDEX	INDEX EXTENSION	METHOD	NICKNAME	PROCEDURE	SEQUENCE	SERVER	SERVICE CLASS	TABLE	TABLESPACE	TRANSFORM	TRIGGER	TYPE	USER MAPPING	VARIABLE	WORKLOAD	WRAPPER	XSROBJECT
ALTER TYPE DROP ATTRIBUTE	-	-	-	-	-	R	-	-	-	A[23]	-	-	R[24]	-	-	-	-	-	-	R[14]	-	-	-	-
ALTER TYPE ADD METHOD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TYPE DROP METHOD	-	-	-	-	-	-	R[27]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CREATE METHOD	-	-	-	-	-	-	-	-	-	A[28]	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CREATE TYPE	-	-	-	-	-	-	-	-	-	A[29]	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP ALIAS	-	R	-	R	-	-	-	-	-	A[3]	-	-	R[3]	-	-	X[3]	-	-	-	X[3]	-	-	-	-
DROP BUFFERPOOL	-	-	-	-	-	-	-	-	-	-	-	-	-	R	-	-	-	-	-	-	-	-	-	-
DROP DATABASE PARTITION GROUP	-	-	-	-	-	-	-	-	-	-	-	-	-	C	-	-	-	-	-	-	-	-	-	-
DROP FUNCTION	R	R[7]	R	R	-	R	R[7]	-	-	X	-	-	R	-	-	R	-	-	-	R	-	-	-	-
DROP FUNCTION MAPPING	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP INDEX	R	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	R[17]	-	-	-	-
DROP INDEX EXTENSION	-	R	-	R	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP METHOD	R	R[7]	R	R	-	R	R	-	-	X/A[30]	-	-	R	-	-	R	-	-	-	R	-	-	-	-
DROP NICKNAME	-	R	-	R	C	-	R	-	-	A	-	-	C[11]	-	-	-	-	-	-	X[16]	-	-	-	-
DROP PROCEDURE	-	R[7]	-	R	-	-	R[7]	-	-	A	-	-	-	-	-	R	-	-	-	-	-	-	-	-
DROP SEQUENCE	-	R	-	-	-	-	R	-	-	A	-	-	-	-	-	R	-	-	-	-	-	-	-	-
DROP SERVER	-	C[21]	C[19]	-	-	-	-	C	-	A	-	-	-	-	-	-	-	C[19]	C	-	-	-	-	-
DROP SERVICE CLASS	-	-	-	-	-	-	-	-	-	-	-	R[35]	-	-	R[35]	-	-	-	-	-	R[35]	-	R[35]	-
DROP TABLE[32]	C	R	-	R	C	-	-	-	-	A[9]	-	-	RC[11]	-	-	X[16]	-	-	-	X[16]	-	-	-	X[34]
DROP TABLE HIERARCHY	C	R	-	R	C	-	-	-	-	A[9]	-	-	RC[11]	-	-	X[16]	-	-	-	X[16]	-	-	-	-
DROP TABLESPACE	-	-	-	-	C[6]	-	-	-	-	-	-	-	CR[6]	-	-	-	-	-	-	-	-	-	-	-
DROP TRANSFORM	-	R	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP TRIGGER	-	-	-	-	-	-	-	-	-	A[1]	-	-	-	-	-	X[26]	-	-	-	-	-	-	-	-
DROP TYPE	R[13]	R[5]	-	R	-	R	-	-	-	A[12]	-	-	R[18]	-	-	R[13]	R[4]	-	-	R[14]	-	-	-	-
DROP VARIABLE	-	-	R	R	-	-	R	-	-	A	-	-	-	-	-	R	-	-	-	R	-	-	-	-

표 28. 종속성 (계속)

	오브젝트 유형																								
명령문	CONSTRAINT	FUNCTION MAPPING	FUNCTION	GLOBAL VARIABLE	INDEX EXTENSION	INDEX	MASK	NICKNAME	DBPARTITIONGROUP	PROCEDURE	ALIAS	SEQUENCE	TRIGGER	TRANSFORM	TABLE	SUMMARY TABLE	VIEW	HIERARCHY	TYPE MAPPING	TYPE	USAGE LIST	WORK CLASS SET	WORKLOAD	WRAPPER	XSROBJECT
DROP VIEW	-	R	-	R	-	-	-	-	-	A^{2}	-	-	-	-	-	X^{16}	-	-	-	X^{15}	-	-	-	-	-
DROP VIEW HIERARCHY	-	R	-	R	-	-	-	-	-	A^{2}	-	-	-	-	-	X^{16}	-	-	-	X^{16}	-	-	-	-	-
DROP WORK CLASS SET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R^{36}	-	-	-
DROP WRAPPER	-	-	C	-	-	-	-	-	-	-	C	-	-	-	-	-	-	C	-	-	-	-	-	-	-
DROP XSROBJECT	C	-	-	-	-	-	-	-	-	A	-	-	-	-	-	X	-	-	-	X	-	-	-	-	-
REVOKE 특권[10]	-	CR^{25}	-	-	-	-	CR^{25}	-	-	A^{1}	-	-	CX^{8}	-	-	X	-	-	-	X^{8}	-	-	-	-	-

1 해당 종속성은 제한조건, 트리거 또는 분산 키를 가진 테이블에 내재적으로 종속됩니다.

2 패키지에 뷰에서 활동하는 INSERT, UPDATE, DELETE문이 있는 경우 해당 뷰의 하위 기본 테이블에서 삽입, 갱신 또는 삭제를 사용할 수 있습니다. UPDATE의 경우 이것으로 수정된 하위 기본 테이블의 각 컬럼에서 갱신을 사용할 수 있습니다.

패키지에 유형이 지정된 뷰에 대해 작동하는 명령문이 있는 경우 동일한 뷰 계층에 있는 뷰를 작성하거나 삭제하면 패키지가 무효화됩니다.

3 패키지, 구체화된 쿼리 테이블, 스테이징 테이블, 뷰 또는 트리거가 별명을 사용하는 경우 이 별명과 그 별명이 참조하는 오브젝트 모두에 종속됩니다. 별명이 연쇄되어 있는 경우 연쇄되어 있는 각 별명에 대해 종속성이 작성됩니다.

별명 자체는 독립적입니다. 존재하지 않는 오브젝트상에 별명을 정의할 수 있습니다.

4 사용자 정의 유형 T는 다음과 같은 경우 다른 사용자 정의 유형 B에 종속될 수 있습니다.

- B를 속성의 데이터 유형으로 이름 지정
- REF(B) 속성을 가짐
- 슈퍼 유형으로 B를 가짐

5 사용자 정의 유형을 삭제하면 연쇄적으로 해당 유형을 매개변수, 결과 유형 또는 함수나 메소드 내용에서 사용하는 함수와 메소드가 삭제됩니다. 사용자 정의 유형이 구조화된 유형이면 해당 유형과 연관된 모든 메소드도 삭제됩니다. 이러한 함수와 메소드의 삭제는 유형과 함수 또는 메소드가 서로 종속적이라는 사실만으로 예방되지는 않습니다.

6 테이블 스페이스나 테이블 스페이스의 목록을 삭제하면 주어진 테이블 스페이스 목록 내에 포함된 모든 테이블이 삭제됩니다. 그러나 한 테이블이 테이블 스페이스(다른 테이블 스페이스에 있는 인덱스, 긴 컬럼 또는 데이터 파티션)에 걸쳐 있고 해당 테이블 스페이스가 삭제 중인 목록에 없는 경우, 테이블이 존재하는 동안은 테이블 스페이스를 삭제할 수 없습니다.

7 종속 함수가 SOURCE절에 있는 기본 함수를 이름 지정하는 경우 함수가 다른 특정 함수에 종속될 수 있습니다. 함수나 메소드는 종속되는 루틴이 SQL로 작성되고 본문에서 기본 루틴을 사용하는 경우 다른 특정 함수나 메소드에 종속적일 수 있습니다. 구조화된 유형 매개변수나 리턴 유형을 갖는 외부 메소드나 외부 함수는 하나 이상의 변환 함수에 종속적이 됩니다.

8 SELECT 특권이 없어지면 삭제될 구체화된 쿼리 테이블이나 뷰가 작동 불능 상태로 됩니다. 작동 불능 상태가 된 뷰가 유형이 지정된 뷰 계층에 포함될 경우 그에 속한 모든 서브뷰도 작동 불능 상태가 됩니다.

9 패키지에 테이블 T에서 활동하는 INSERT, UPDATE, DELETE문이 있는 경우 패키지는 T에서 삽입, 갱신 또는 삭제를 사용할 수 있습니다. UPDATE의 경우 이것으로 수정된 기초가 되는 기본 테이블 T의 각 컬럼에서 갱신을 사용할 수 있습니다.

패키지에 유형이 지정된 테이블에 대해 작동하는 명령문이 있는 경우 동일한 테이블 계층에 있는 테이블을 작성하거나 삭제하면 패키지가 무효화됩니다.

10 컬럼의 특권은 개별적으로 취소될 수 없으므로 컬럼 레벨의 종속성은 존재하지 않습니다.

모든 서브테이블이나 서브뷰상의 SELECT 특권에 대한 종속성이 있습니다. 패키지, 트리거 또는 뷰가 FROM절에서 OUTER(*Z*)의 사용을 포함하면, *Z*의 모든 서브테이블이나 서브뷰의 SELECT 특권에 대해 종속성이 있습니다. 마찬가지로, 패키지, 트리거 또는 뷰가 DEREF(*Y*)의 사용을 포함하면(여기서, *Y*는 테이블이나 뷰 *Z*와의 참조 유형) *Z*의 모든 서브테이블이나 서브뷰상의 SELECT 특권에 대해 종속성이 있습니다.

11 구체화된 쿼리 테이블은 기본 테이블 또는 테이블 정의의 fullselect에 지정된 별칭에 종속됩니다.

연쇄 시맨틱이 종속되는 구체화된 쿼리 테이블에 적용됩니다.

서브테이블은 최대 루트 테이블에 이르기까지 그의 슈퍼 테이블들에 종속적입니다. 모든 서브테이블을 삭제할 때까지 슈퍼 테이블은 삭제할 수 없습니다.

12 패키지는 TYPE 술어나 부속 유형 처리 표현식(TREAT *expression* AS *data-type*)을 사용한 결과로서 구조화된 유형에 종속적일 수 있습니다. 패키지는 TYPE 술어 오른쪽 또는 TREAT 표현식의 오른쪽 부분에 지정된 각 구조화된 유형의 부속 유형에 종속적입니다. 패키지가 종속된 부속 유형을 교체하는 구조화된 유형을 삭제하거나 작성하면 무효화됩니다.

삭제될 유형은 슈퍼 유형으로 정의된 메소드에 의존하므로 겹쳐쓸 수 있는 모든 패키지가 무효화됩니다.

13 제한조건이나 트리거에서 유형이 사용되는 경우 점검 제한조건 또는 트리거는 유형에 종속적입니다. 점검 제한조건 또는 트리거 내부의 TYPE 술어에 사용되는 구조화된 유형의 부속 유형에 종속성이 없습니다.

14 뷰 정의(유형이 지정된 뷰의 유형 포함)에서 유형이 사용되는 경우 뷰는 유형에 종속적입니다. 뷰 정의 내부의 TYPE 술어에 사용되는 구조화된 유형의 부속 유형에 종속성이 없습니다.

15 서브뷰는 그 슈퍼 뷰(루트 뷰까지)에 종속됩니다. 모든 서브뷰를 삭제할 때까지 슈퍼 뷰는 삭제할 수 없습니다. 추가 뷰 종속성에 대해서는 주[16]을 참조하십시오.

16 트리거나 뷰는 비참조 연산 또는 DEREF 함수의 목표 테이블 또는 목표 뷰에도 종속적입니다. OUTER(*Z*)를 포함하는 FROM절이 있는 트리거 또는 뷰는, 트리거나 뷰가 작성될 때 있었던 *Z*의 모든 서브테이블 또는 서브뷰에 종속적입니다.

17 오브젝트 ID 컬럼의 고유성을 확인하기 위해 유형이 지정된 뷰는 고유 인덱스 존재에 종속될 수 있습니다.

18 테이블은 사용자 정의 데이터 유형이 다음과 같기 때문에 이 유형(구별 또는 구조화)에 종속될 수 있습니다.

- 컬럼의 유형으로 사용된 경우
- 테이블의 유형으로 사용된 경우
- 테이블 유형의 속성으로 사용된 경우
- 테이블의 컬럼 유형이나 테이블의 유형 속성인 참조 유형의 목표 유형으로 사용된 경우
- 직접 또는 간접적으로 테이블의 컬럼인 유형에 의해 사용되는 경우

19 서버를 삭제하면 그 이름 지정된 서버에 작성된 함수 맵핑 및 유형 맵핑도 연쇄적으로 삭제됩니다.

[20] 다중 파티션 데이터베이스 파티션 그룹에 있는 테이블에 분산 키가 정의되면, 분산 키가 필요합니다.

[21] 종속 OLE DB 테이블 함수가 ″R″ 종속 오브젝트를 가지는 경우(DROP FUNCTION 참조) 서버를 삭제할 수 없습니다.

[22] SQL 함수나 메소드는 그 내용에서 참조하는 오브젝트에 종속될 수 있습니다.

[23] *type-name* T인 유형 TA의 속성 A를 삭제했다면 다음 DROP문이 효과적으로 실행됩니다.

```
Mutator method: DROP METHOD A (TA) FOR T
Observer method: DROP METHOD A () FOR T
ALTER TYPE T
    DROP METHOD A(TA)
    DROP METHOD A()
```

[24] 테이블은 다음 경우에 사용자 정의 구조화된 데이터 유형의 속성에 종속될 수 있습니다.

1. 해당 테이블이 *type-name* 또는 그의 부속 유형에 기초하는 유형이 지정된 테이블인 경우
2. 테이블에 직접 또는 간접적으로 *type-name*을 참조하는 유형의 기존 컬럼이 있는 경우

[25] SQL 함수 또는 메소드 본문의 내용에 사용되는 테이블이나 뷰에 대한 SELECT 특권을 취소하는 경우 정의된 해당 함수 또는 메소드 본문에 더 이상 SELECT 특권이 없다면 해당 함수 또는 메소드가 삭제됩니다. 이러한 함수 또는 메소드 본문은 뷰, 트리거, 함수 또는 메소드 본문에 사용될 경우 삭제할 수 없으며 REVOKE는 결과로 제한됩니다. 그렇지 않으면 REVOKE가 연쇄적으로 이러한 함수를 삭제합니다.

[26] 트리거에 INSTEAD OF 트리거가 정의되어 있어 INSTEAD OF 트리거가 발생하는 뷰를 수정할 때 트리거는 INSTEAD OF 트리거에 의존합니다.

[27] 다른 메소드로 겹쳐쓴 원본 메소드의 메소드 선언은 삭제할 수 없습니다 (SQLSTATE 42893).

[28] 작성될 메소드 본문의 메소드가 다른 메소드를 겹쳐쓰도록 선언되어 있을 경우, 겹쳐쓰여지는 메소드에 종속된 모든 패키지와 작성될 메소드의 슈퍼 유형으로 이 메소드를 겹쳐쓸 메소드에 종속된 모든 패키지는 무효화됩니다.

[29] 기존 유형의 새 부속 유형이 작성될 때 작성되는 유형의 슈퍼 유형으로 정의된 메소드에 종속되고 겹쳐쓸 수 있는 메소드(예: no mutator 또는 observer)에 종속된 모든 패키지가 무효화됩니다.

[30] 작성될 메소드 본문의 특정 메소드가 다른 메소드를 겹쳐쓰도록 선언되어 있을 경우, 겹쳐쓰여지는 메소드에 종속된 모든 패키지와 작성될 메소드의 슈퍼 유형으로 이 메소드를 겹쳐쓸 메소드에 종속된 모든 패키지는 무효화됩니다.

31 캐시된 동적 SQL의 시맨틱이 패키지의 시맨틱과 같습니다.

32 DROP TABLE문을 사용하여 리모트 기본 테이블을 삭제할 경우 별칭 및 리모트 기본 테이블 모두가 삭제됩니다.

33 외부 키에 의해 참조되지 않는 기본 키 또는 고유 키는 별칭 로컬 이름 또는 로컬 유형의 변경을 제한하지 않습니다.

34 XML 스키마와 연결된 테이블에 분석을 위해 변경을 수행하면 XSROBJECT는 분석 작동 불능이 될 수 있습니다. 분석에 영향을 줄 수 있는 변경사항은 다음과 같습니다. 테이블 삭제, 테이블 컬럼 삭제 또는 테이블 컬럼 변경입니다. ALTER XSROBJECT문을 발행하여 XML 스키마의 분석 상태를 재설정하여 XML 스키마 분석을 사용 가능하게 하거나 불가능하게 할 수 있습니다.

35

- 임계값이 서비스 클래스에 맵핑되면 서비스 클래스를 삭제할 수 없습니다(SQLSTATE 5U031).
- 워크로드가 서비스 클래스에 맵핑되면 서비스 클래스를 삭제할 수 없습니다(SQLSTATE 5U031).
- 모든 사용자 정의 서비스 서브클래스를 삭제할 때까지 서비스 수퍼 클래스는 삭제할 수 없습니다(SQLSTATE 5U031).
- 작업 조치 세트가 서비스 수퍼 클래스에 맵핑되면 서비스 수퍼 클래스를 삭제할 수 없습니다(SQLSTATE 5U031).
- 작업 조치 세트가 서비스 서브클래스에 맵핑되면 서비스 서브클래스를 삭제할 수 없습니다(SQLSTATE 5U031).

36 작업 클래스 세트에서 정의된 작업 조치 세트가 삭제될 때까지 작업 클래스 세트를 삭제할 수 없습니다.

표 29. **auto_reval***에 영향을 받는 종속 오브젝트*

명령문	영향을 받는 종속 오브젝트
ALTER NICKNAME(로컬 이름 또는 로컬 유형 변경)	앵커 유형, 함수, 메소드, 프로시저, 사용자 정의 유형, 변수, 뷰
ALTER TABLE ALTER COLUMN	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰, XSROBJECT
ALTER TABLE DROP COLUMN[2]	앵커 유형, 함수, 메소드, 인덱스, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰, XSROBJECT
ALTER TABLE RENAME COLUMN[1, 3]	앵커 유형, 함수, 메소드, 인덱스, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰, XSROBJECT
ALTER TYPE ADD ATTRIBUTE	뷰
ALTER TYPE DROP ATTRIBUTE	뷰
DROP ALIAS	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰
DROP FUNCTION(ALTER MODULE DROP FUNCTION)	함수, 함수 맵핑, 인덱스 확장, 메소드, 프로시저, 트리거, 변수, 뷰
DROP METHOD	함수, 함수 맵핑, 인덱스 확장, 메소드, 프로시저, 트리거, 뷰, 변수

표 29. **auto_reval***에 영향을 받는 종속 오브젝트 (계속)*

명령문	영향을 받는 종속 오브젝트
DROP NICKNAME	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰
DROP PROCEDURE (ALTER MODULE DROP PROCEDURE)	함수, 메소드, 프로시저, 트리거
DROP SEQUENCE	함수, 메소드, 프로시저, 트리거, 변수, 뷰
DROP TABLE	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰, XSROBJECT
DROP TABLE HIERARCHY	함수, 메소드, 프로시저, 트리거, 변수, 뷰
DROP TRIGGER	트리거
DROP TYPE (ALTER MODULE DROP TYPE)	앵커 유형, 커서 유형, 함수, 메소드, 프로시저, 인덱스 확장, 트리거, 사용자 정의 유형, 변수, 뷰
DROP VARIABLE (ALTER MODULE DROP VARIABLE)	앵커 유형, 함수, 함수 맵핑, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰
DROP VIEW	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰
DROP VIEW HIERARCHY	함수, 프로시저, 트리거, 변수, 뷰
DROP XSROBJECT	트리거, 뷰
RENAME TABLE	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 뷰, 변수, XSROBJECT
REVOKE 특권	함수, 메소드, 프로시저, 트리거, 변수, 뷰
CREATE OR REPLACE ALIAS[1]	함수, 트리거, 프로시저, 변수, 뷰
CREATE OR REPLACE VIEW[1]	앵커 유형, 함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰
CREATE OR REPLACE FUNCTION[1]	함수, 함수 맵핑, 인덱스 확장, 메소드, 프로시저, 변수, 뷰
CREATE OR REPLACE PROCEDURE[1]	함수, 메소드, 프로시저, 트리거
CREATE OR REPLACE NICKNAME[1]	함수, 메소드, 프로시저, 변수, 뷰
CREATE OR REPLACE SEQUENCE[1]	함수, 메소드, 프로시저, 트리거, 변수, 뷰
CREATE OR REPLACE VARIABLE[1]	함수, 메소드, 프로시저, 트리거, 사용자 정의 유형, 변수, 뷰
CREATE OR REPLACE TRIGGER[1]	트리거

1 **auto_reval** 데이터베이스 구성 매개변수의 설정에 상관 없이 REVALIDATION IMMEDIATE 시맨틱은 명령문에 적용됩니다(CREATE문의 경우, OR REPLACE가 지정된 경우만).

2 나열된 종속 오브젝트는 다음에 오브젝트를 사용할 때 유효성이 다시 확인되며, 다음 오브젝트를 제외한 오브젝트는 명령문의 일부로 즉시 유효성이 다시 확인됩니다.

- ANCHOR TYPE
- CURSOR TYPE
- VIEW(선택 목록이 SELECT *로만 구성되어 있고 명시적으로 정의된 뷰 컬럼이 포함되지 않은 경우).

즉각적인 뷰 유효성 다시 확인의 경우, 선택 목록의 컬럼 이름 목록이 유효성 다시 확인 중에 다시 설정됩니다.

3 나열된 종속 오브젝트는 다음에 오브젝트가 다음을 제외하고 사용할 때 다시 유효성이 확인되며, 다음 오브젝트는 명령문의 일부로 즉시 다시 유효성이 확인됩니다.

- 사용자 정의 유형
- VIEW(선택 목록이 SELECT *로만 구성되어 있고 명시적으로 정의된 뷰 컬럼이 포함되지 않은 경우).

즉각적인 뷰 유효성 다시 확인의 경우, 선택 목록의 컬럼 이름 목록이 유효성 다시 확인 중에 다시 설정됩니다.

데이터베이스 파티션 서버 요청이 보류 또는 진행 중인 경우 DROP DATABASE PARTITION GROUP문이 실패할 수 있습니다(SQLSTATE 55071). 새 데이터베이스 파티션 서버가 온라인으로 인스턴스에 추가되고 모든 응용프로그램이 새 데이터베이스 파티션 서버를 인식하지 않는 경우, 이 명령문도 실패할 수 있습니다(SQLSTATE 55077).

주

- 사용자 정의 함수를 사용 중인 동안에는 삭제할 수 있습니다. 또한 커서가 사용자 정의 함수를 참조하는 명령문에서 열린 경우 이 커서가 열려 있는 동안 함수는 그 커서의 페치를 실패하지 않고도 삭제할 수 있습니다.
- 사용자 정의 함수에 종속되는 패키지를 실행하는 경우 패키지가 현재의 작업 단위(UOW)를 완료할 때까지는 또 다른 권한 부여 ID로 함수를 삭제할 수 없습니다. 그 시점에서 함수는 삭제되고 패키지는 작동 불능 상태가 됩니다. 다음에 이 패키지에 대한 요청을 하면 패키지가 명시적으로 리바인드되어야 함을 나타내는 오류가 발생합니다.
- 함수 본문의 제거(함수의 삭제와는 전혀 다름)가 이 함수 본문을 필요로 하는 응용프로그램의 실행 중에 발생할 수 있습니다. 이로 인해 명령문이 실패할 수도 있는데 이는 명령문을 대신해서 데이터베이스 관리 프로그램이 함수 본문을 스토리지에 로드해야 하는지에 따라 다릅니다.
- 변환이 내재적으로 필요한 경우에는 명시적으로 지정된 UDF에 대해 기록된 종속성에 추가로 다음 종속성도 기록됩니다.
  1. 함수나 메소드의 구조화된 유형 매개변수 또는 결과에 변환이 필요한 경우 필요한 TO SQL 또는 FROM SQL 변환 함수에 대해 해당 함수나 메소드의 종속성이 기록됩니다.
  2. 패키지에 포함된 SQL문이 변환 함수를 필요로 하는 경우에는 지정된 TO SQL이나 FROM SQL 변환 함수에 대해 해당 패키지의 종속성이 기록됩니다.

위에서는 변환의 내재적 호출로 인해 종속성이 기록되는 환경만을 기술하였으므로 함수, 메소드 또는 패키지가 아닌 다른 오브젝트는 내재적으로 호출된 변환 함수에 대한 종속성을 가질 수 없습니다. 다시 말해, 변환 함수에 대한 명시적 호출(예를 들어, 뷰 및 트리거에서)의 결과 이러한 유형의 다른 오브젝트가 변환 함수에 종속됩니다. 즉, DROP TRANSFORM문이 삭제 중인 변환 함수에 대한 오브젝트의 이러한 "명시적" 유형 종속성으로 인해 실패할 수도 있습니다(SQLSTATE 42893).

- 종속성 카탈로그는 변환 함수로서 함수에 종속되는지와 명시적 함수 호출에 의해 함수에 종속되는지를 구별하지 않으므로 변환 함수에 대한 명시적 호출은 작성되지 않는다고 가정합니다. 이 경우 함수에 대한 변환 등록 정보를 삭제할 수 없거나 패키지가 작동 불능으로 표시되는데 이는 단순히 SQL 표현식에 명시적 호출을 포함하기 때문입니다.
- IDENTITY 컬럼에 대해 시스템이 작성한 시퀀스는 DROP SEQUENCE문을 사용하여 삭제할 수 없습니다.
- 시퀀스가 삭제되면 시퀀스에 대한 모든 특권도 삭제되고 시퀀스를 참조하는 모든 패키지도 무효화됩니다.
- 관계형 별칭의 경우 다음 조건에서는 주어진 작업 단위(UOW) 내에서는 DROP NICKNAME문을 처리할 수 없습니다(SQLSTATE 55007).
  - 이 명령문에서 참조하는 별칭의 동일한 UOW에서 커서가 열려 있는 경우
  - 이 명령문에서 참조하는 별칭에 대해 동일한 UOW에서 이미 INSERT, DELETE 또는 UPDATE문이 발행된 경우
- 비관계형 별칭의 경우 다음 조건에서는 주어진 작업 단위(UOW) 내에서는 DROP NICKNAME문을 처리할 수 없습니다(SQLSTATE 55007).
  - 이 명령문에서 참조하는 별칭의 동일한 UOW에서 커서가 열려 있는 경우
  - 이 명령문에서 참조하는 별칭이 동일한 UOW에 있는 SELECT문에 의해 이미 참조된 경우
  - 이 명령문에서 참조하는 별칭에 대해 동일한 UOW에서 이미 INSERT, DELETE 또는 UPDATE문이 발행된 경우
- 다음 조건에서는 주어진 작업 단위(UOW) 내에서 DROP SERVER문(SQLSTATE 55006) 또는 DROP FUNCTION MAPPING이나 DROP TYPE MAPPING문(SQLSTATE 55007)을 처리할 수 없습니다.
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 내의 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 내에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문

  - 명령문은 데이터 소스의 범주(예: 특정 유형 및 버전의 모든 데이터 소스)를 참조하며, UOW는 이미 다음 중 하나를 포함합니다.
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭을 참조하는 SELECT문
    - 이 데이터 소스 중 하나에 있는 테이블이나 뷰에 대한 별칭의 열린 커서
    - 이 데이터 소스 중 하나에 있는 테이블 또는 뷰의 별칭에 대해 발행된 INSERT, DELETE 또는 UPDATE문
- 명령문을 발행하는 연결에 대해서도 명령문이 커미트될 때까지 DROP WORKLOAD 명령문에서 변경한 내용은 영향을 주지 않습니다.
- 한 번에 이들 명령문의 하나만 응용프로그램에서 발행될 수 있으며 이들 명령문 중 하나만 하나의 작업 단위 내에서 허용됩니다. 각 명령문 다음에는 COMMIT 또는 ROLLBACK 명령문이 와야 이들 명령문 중 다른 명령문을 발행할 수 있습니다 (SQLSTATE 5U021).
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE 또는 DROP(HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS 또는 DROP(SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD 또는 DROP(THRESHOLD)
  - CREATE WORK ACTION, ALTER WORK ACTION 또는 DROP(WORK ACTION)
  - CREATE WORK CLASS, ALTER WORK CLASS 또는 DROP(WORK CLASS)
  - CREATE WORKLOAD, ALTER WORKLOAD 또는 DROP(WORKLOAD)
  - GRANT(워크로드 특권) 또는 REVOKE(워크로드 특권)
- ***소프트 무효화:*** 다음 명령문으로 데이터베이스 오브젝트가 삭제되거나 변경된 후, 액세스가 완료될 때까지 삭제되거나 변경된 오브젝트에 대한 활성 액세스가 지속됩니다.
  - ALTER FUNCTION
  - ALTER MODULE ... DROP FUNCTION
  - ALTER MODULE ... DROP VARIABLE
  - ALTER TABLE ... DETACH PARTITION
  - ALTER VIEW
  - DROP ALIAS
  - DROP FUNCTION
  - DROP TRIGGER
  - DROP VARIABLE

  - DROP VIEW
  - CREATE OR REPLACE SEQUENCE를 제외한 모든 CREATE OR REPLACE문.

  데이터베이스 레지스트리 변수 *DB2_DLL_SOFT_INVALID*가 ON으로 설정된 경우입니다. OFF로 설정되면 삭제되거나 변경될 오브젝트에 대한 모든 활성 액세스가 완료된 다음에만 이들 오브젝트의 삭제나 변경이 완료됩니다.

- *호환성*:
  - 이전 버전의 DB2 데이터베이스와의 호환성을 위해,
    - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.
    - TYPE *type-name* 대신 DISTINCT TYPE *type-name*을 지정할 수 있습니다.
    - TYPE *type-name* 대신 DATA TYPE *type-name*을 지정할 수 있습니다.
  - ALIAS 대신 SYNONYM을 지정할 수 있습니다.
  - OS/390 및 z/OS용 DB2 UDB와의 호환성:
    - PACKAGE 대신 PROGRAM을 지정할 수 있습니다.

예:

예 1: TDEPT 테이블을 삭제하십시오.

```
DROP TABLE TDEPT
```

예 2: 뷰 VDEPT를 삭제하십시오.

```
DROP VIEW VDEPT
```

예 3: 권한 부여 ID, HEDGES로 별명을 삭제하십시오.

```
DROP ALIAS A1
```

별명 HEDGES.A1은 카탈로그에서 제거됩니다.

예 4: T1을 별명 이름으로 별명을 삭제하십시오. 여기서, T1은(별명이 아닌) 기존 테이블의 이름입니다.

```
DROP ALIAS T1
```

이 명령문은 실패합니다(SQLSTATE 42809).

예 5:

BUSINESS_OPS 데이터베이스 파티션 그룹을 삭제하십시오. 데이터베이스 파티션 그룹을 삭제하려면 데이터베이스 파티션 그룹의 두 테이블 스페이스(ACCOUNTING 및 PLANS)를 먼저 삭제해야 합니다.

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP DATABASE PARTITION GROUP BUSINESS_OPS
```

예 6: Pellow 사용자로 삭제할 함수 인스턴스를 식별하기 위한 시그니처를 사용하여 PELLOW 스키마에 작성한 CENTRE 함수를 삭제하십시오.

```
DROP FUNCTION CENTRE (INT,FLOAT)
```

예 7 Pellow 사용자로 삭제할 함수 인스턴스를 식별하기 위한 시그니처를 사용하여 PELLOW 스키마에 작성한 CENTRE 함수를 삭제하십시오.

```
DROP SPECIFIC FUNCTION PELLOW.FOCUS92
```

예 8: CHEM 스키마의 함수 ATOMIC_WEIGHT를 삭제하십시오. 이 스키마에는 이 이름을 갖는 함수가 하나 뿐입니다.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT
```

예 9: 사원들이 특정 조건하에서 급여에 대한 상여금을 받도록 한 트리거 SALARY_BONUS를 삭제하십시오.

```
DROP TRIGGER SALARY_BONUS
```

예 10: shoesize라는 구별 데이터 유형을 현재 사용하지 않는 경우 이를 삭제하십시오.

```
DROP TYPE SHOESIZE
```

예 11: SMITHPAY 이벤트 모니터를 삭제하십시오.

```
DROP EVENT MONITOR SMITHPAY
```

예 12: RESTRICT를 사용하는 CREATE SCHEMA에 따라 예 2에서 스키마를 삭제하십시오. PART 테이블을 우선 삭제해야 합니다.

```
  DROP TABLE PART
DROP SCHEMA INVENTRY RESTRICT
```

예 13: Macdonald로 삭제할 프로시저 인스턴스를 식별하기 위한 특정 이름을 사용하여 EIGLER 스키마에 작성한 DESTROY 프로시저를 삭제하십시오.

```
DROP SPECIFIC PROCEDURE  EIGLER.DESTROY
```

예 14: BIOLOGY 스키마에서 프로시저 OSMOSIS를 삭제하십시오. 이 스키마에는 이 이름을 갖는 프로시저가 하나뿐입니다.

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

예 15: 사용자 SHAWN이 하나의 권한 부여 ID를 페더레이티드 데이터베이스를 액세스하는 데 사용하고, 다른 것을 ORACLE1이라고 하는 Oracle 데이터 소스를 액세스하는 데 사용했습니다. 두 권한 부여 사이에 맵핑이 작성되었으나, SHAWN은 더 이상 데이터 소스를 액세스할 필요가 없습니다. 맵핑을 삭제하십시오.

```
DROP USER MAPPING FOR  SHAWN SERVER ORACLE1
```

예 16: 별칭이 참조하는 데이터 소스 테이블의 인덱스가 삭제되었습니다. 옵티마이저가 이 인덱스에 대해 알게 하도록 작성된 인덱스 스펙을 삭제하십시오.

```
DROP INDEX INDEXSPEC
```

예 17: MYSTRUCT1 변환 그룹을 삭제하십시오.

```
DROP TRANSFORM MYSTRUCT1 FOR POLYGON
```

예 18: PERSONNEL 스키마에서 EMP 데이터 유형의 메소드 BONUS를 삭제하십시오.

```
DROP METHOD BONUS (SALARY DECIMAL(10,2)) FOR PERSONNEL.EMP
```

예 19: 제한사항을 갖는 ORG_SEQ 시퀀스를 삭제하십시오.

```
DROP SEQUENCE ORG_SEQ
```

예 20: 리모트 테이블 EMPLOYEE가 투명한 DDL을 사용하는 페더레이티드 시스템에서 작성되었습니다. 테이블에 대한 액세스는 더 이상 필요하지 않습니다. 리모트 테이블 EMPLOYEE를 삭제하십시오.

```
DROP TABLE EMPLOYEE
```

예 21: 함수 맵핑 BONUS_CALC를 삭제하고 디폴트 함수 맵핑(존재할 경우)을 다시 사용하십시오.

```
DROP FUNCTION MAPPING BONUS_CALC
```

예 22: 보안 레이블 구성요소 LEVEL을 삭제하십시오.

```
DROP SECURITY LABEL COMPONENT LEVEL
```

예 23: 보안 규정 DATA_ACCESS의 보안 레이블 EMPLOYEESECLABEL을 삭제하십시오.

```
DROP SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
```

예 24: 보안 규정 DATA_ACCESS를 삭제하십시오.

```
DROP SECURITY POLICY DATA_ACCESS
```

예 25: 보안 레이블 구성요소 GROUPS를 삭제하십시오.

```
DROP SECURITY LABEL COMPONENT GROUPS
```

예 26: SQL 스키마 HR에 위치한 XML 스키마 EMPLOYEE를 삭제하십시오.

```
DROP XSROBJECT HR.EMPLOYEE
```

예 27: 서비스 수퍼 클래스 PETSALES에서 서비스 서브클래스 DOGSALES를 삭제합니다.

```
DROP SERVICE CLASS DOGSALES UNDER PETSALES
```

예 28: 사용자 정의 서비스 서브클래스가 없는 서비스 수퍼 클래스 PETSALES를 삭제합니다. 서비스 클래스 PETSALES에 대한 디폴트 서브클래스는 자동으로 삭제됩니다.

```
DROP SERVICE CLASS PETSALES
```

END DECLARE SECTION

END DECLARE SECTION 문은 호스트 변수 선언 섹션의 끝을 표시합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이는 실행문이 아닙니다. REXX에 지정해서는 안됩니다.

권한 부여

필요한 권한 없음

구문

설명

END DECLARE SECTION 문은 호스트 언어의 규칙에 따라 선언이 나타날 수 있는 응용프로그램에서 코딩할 수 있습니다. 호스트 변수 선언 섹션의 끝을 표시합니다. 호스트 변수 섹션은 BEGIN DECLARE SECTION 문으로 시작합니다.

BEGIN DECLARE SECTION 및 END DECLARE SECTION 문은 쌍이어야 하며 중첩될 수 없습니다.

호스트 변수 선언은 SQL INCLUDE 문을 사용하여 지정할 수 있습니다. 그렇지 않으면, 호스트 변수 선언 섹션에 호스트 변수 선언이 아닌 다른 명령문을 포함할 수 없습니다.

SQL문에 참조된 호스트 변수는 REXX가 아닌 다른 모든 호스트 언어로 호스트 변수 선언 섹션에서 선언해야 합니다. 또한 각 변수의 선언은 첫 번째 변수 참조 이전에 있어야 합니다.

선언 섹션 밖에서 선언된 변수는 선언 섹션 내에서 선언된 변수와 동일한 이름을 가질 수 없습니다.

EXECUTE

EXECUTE문은 Prepared SQL문을 실행합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

각 전역 변수를 USING절의 *expression*으로 사용하거나 *array-index*에 대한 표현식에서 사용하는 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 READ 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

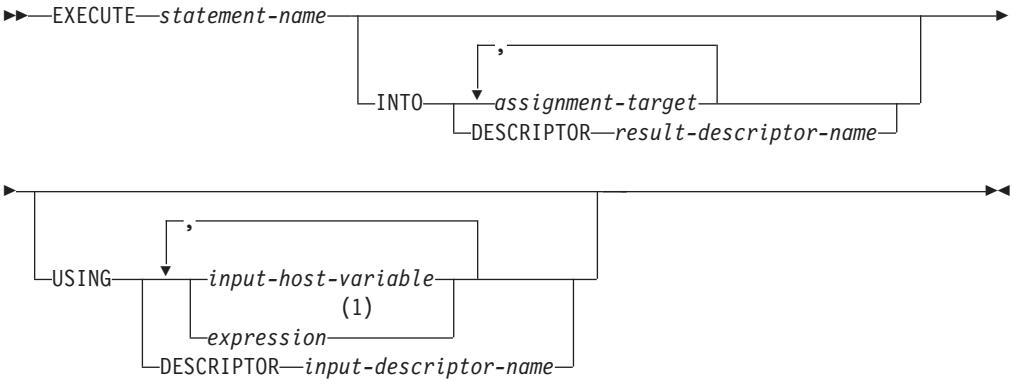
*assignment-target*으로 사용되는 각 전역 변수의 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 WRITE 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

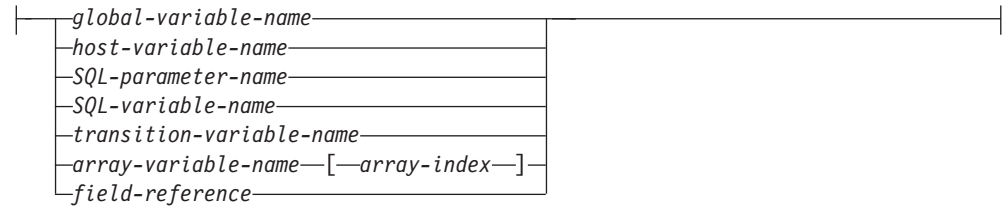
명령문(DDL, GRANT 및 REVOKE문) 실행시 권한 부여 점검이 수행될 경우 명령문의 권한 부여 ID가 부여하는 특권은 PREPARE문이 지정한 SQL문을 실행하는 데 필요한 명령문을 포함해야 합니다. 명령문의 권한 부여 ID는 바인드 옵션 DYNAMICRULES의 영향을 받을 수 있습니다.

명령문(DML) 준비 시 권한 부여 점검이 수행되는 명령문의 경우, PREPARE문에 의해 지정된 SQL문에서 권한 부여 검사가 더 이상 수행되지 않습니다.

구문



assignment-target:



주:

1 *host-variable* 이외의 표현식은 EXECUTE문이 복합 SQL(컴파일된)문 내에서 사용되는 경우에만 사용할 수 있습니다.

설명

statement-name

실행될 준비 명령문을 식별합니다. *statement-name*은 이전에 준비된 명령문을 식별해야 하며, 준비된 명령문은 SELECT문일 수 없습니다.

INTO

PREPARE문에 있는 출력 매개변수 표시문자에서 값을 수신하는 데 사용되는 목표 목록을 표시합니다. 목록에 나타난 순서대로 목표에 값이 지정됩니다. 지정 오류가 발생하는 경우, 목표에 값이 지정되지 않고 목표에 더 이상 값이 지정되지 않습니다. 목표에 이미 지정된 값은 지정된 상태로 남아 있습니다.

동적 CALL문의 경우에는 프로시저에 대한 OUT 및 INOUT 인수에 나타나는 매개변수 표시문자가 출력 매개변수 표시문자입니다. 명령문에 출력 매개변수 표시문자가 나타날 경우에는 INTO절을 지정해야 합니다(SQLSTATE 07007).

assignment-target

출력 값을 지정할 하나 이상의 목표를 식별합니다. 결과 행의 첫 번째 값은 목록의 첫 번째 목표에 지정되고, 두 번째 값은 두 번째 목표에, 이와 같은 식으로 지정됩니다.

*assignment-target*의 데이터 유형이 행 유형이면, 하나의 *assignment-target*가 명확히 지정되어 있어야 하며(SQLSTATE 428HR), 컬럼 수는 행 유형의 필드 수와 일치되어야 하며, 페치된 행 컬럼의 데이터 유형이 행 유형의 해당 필드에 지정 가능해야 합니다(SQLSTATE 42821).

*assignment-target*의 데이터 유형이 배열 요소이면, 정확히 지정된 하나의 *assignment-target*이 있어야 합니다.

global-variable-name

지정 목표인 전역 변수를 식별합니다.

host-variable-name

지정 목표인 호스트 변수를 식별합니다. LOB 출력 값의 경우, 목표는 일반 호스트 변수(충분히 큰 경우), LOB 로케이터 변수 또는 LOB 파일 참조 변수가 될 수 있습니다.

SQL-parameter-name

지정 목표인 루틴 매개변수를 식별합니다.

SQL-variable-name

지정 목표인 SQL 변수를 식별합니다. SQL 변수는 사용하기 전에 선언해야 합니다.

transition-variable-name

전이 행에서 갱신될 컬럼을 식별합니다. *transition-variable-name*은 트리거의 주제 테이블에 있는 컬럼을 식별해야 하고, 새 값을 식별하는 상관 이름에 의해 선택적으로 규정됩니다.

array-variable-name

배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다.

array-index

지정의 목표가 되는 배열에 있는 요소를 지정하는 표현식. 일반 배열의 경우, *array-index* 표현식은 INTEGER(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다. 배열에 정의된 1과 최대 카디널리티 사이의 값이어야 합니다(SQLSTATE 2202E). 연관된 배열의 경우, *array-index* 표현식은 연관된 배열의 인덱스 데이터 유형(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다.

field-reference

지정 목표인 행 유형 값 내에서 필드를 식별합니다. *field-reference*는 규정된 *field-name*으로 지정되어야 하며 이때 규정자는 필드가 정의된 행 값을 식별합니다.

DESCRIPTOR *result-descriptor-name*

호스트 변수의 유효한 설명이 포함되어 있는 출력 SQLDA를 식별합니다.

EXECUTE문이 처리되기 전에 다음 필드를 입력 SQLDA에 설정해야 합니다.

- SQLDA에 제공되는 SQLVAR 어커런스 수를 나타내기 위한 SQLN
- SQLDA에 대해 할당되는 스토리지의 바이트 수를 나타내기 위한 SQLDABC
- 명령문 처리시 SQLDA에서 사용되는 변수의 수를 나타내기 위한 SQLDA
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수

SQLDA는 모든 SQLVAR 어커런스가 들어가기에 충분한 스토리지가 있어야 합니다. 따라서 SQLDABC의 값은 16 + SQLN * (N)보다 크거나 같아야 합니다. 여기서, N은 SQLVAR 어커런스의 길이입니다.

LOB 또는 구조화된 데이터 유형 출력 데이터를 허용해야 할 경우에는 모든 출력 매개변수 표시문자에 대해 두 개의 SQLVAR 항목이 있어야 합니다.

SQLD는 0 이상이거나 SQLN 이하의 값으로 설정되어야 합니다.

USING

PREPARE문에 있는 입력 매개변수 표시문자에 대한 값이 교체되는 변수 또는 표현식 목록을 표시합니다.

동적 CALL문의 경우에는 프로시저에 대한 IN 및 INOUT 인수에 나타나는 매개변수 표시문자가 입력 매개변수 표시문자입니다. 다른 동적 명령문에서는 모든 매개변수 표시문자가 입력 매개변수 표시문자입니다. 명령문에 입력 매개변수 표시문자가 나타날 경우에는 USING절을 지정해야 합니다(SQLSTATE 07004).

input-host-variable, **...**

호스트 변수 선언 규칙에 따라 프로그램에 선언되어 있는 호스트 변수를 식별합니다. 변수의 수는 준비된 명령문의 입력 매개변수 표시문자수와 같아야 합니다. *n*번째 변수는 준비된 명령문의 *n*번째 매개변수에 일치합니다. 로케이터 변수와 파일 참조 변수를 매개변수 표시문자에 대한 값 소스로 제공할 수 있습니다.

expression

PREPARE문의 입력 매개변수 표시문자에 해당되는 입력으로 사용할 표현식을 식별합니다. *host-variable* 이외의 표현식은 EXECUTE문이 복합 SQL(컴파일된)문 내에서 발행되는 경우에만 지정할 수 있습니다.

DESCRIPTOR *input-descriptor-name*

유효한 호스트 변수 설명이 들어 있는 입력 SQLDA를 식별합니다.

EXECUTE문이 처리되기 전에 다음 필드를 입력 SQLDA에 설정해야 합니다.

- SQLDA에 제공되는 SQLVAR 어커런스 수를 나타내기 위한 SQLN
- SQLDA에 대해 할당되는 스토리지의 바이트 수를 나타내기 위한 SQLDABC
- 명령문 처리시 SQLDA에서 사용되는 변수의 수를 나타내기 위한 SQLDA
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수

SQLDA는 모든 SQLVAR 어커런스가 들어가기에 충분한 스토리지가 있어야 합니다. 따라서 SQLDABC의 값은 16 + SQLN * (N)보다 크거나 같아야 합니다. 여기서, N은 SQLVAR 어커런스의 길이입니다.

LOB나 구조화된 데이터 유형 입력 데이터를 허용해야 할 경우에는 모든 매개변수 표시문자에 대해 두 개의 SQLVAR 항목이 있어야 합니다.

SQLD는 0 이상이거나 SQLN 이하의 값으로 설정되어야 합니다.

주

- PREPARE문이 실행되기 전에, 각 입력 매개변수 표시문자는 해당 변수 또는 표현식의 값으로 교체됩니다. 입력된 매개변수 표시문자의 경우, 목표 변수 또는 표현식의 속성은 CAST 스펙으로 지정된 속성입니다. 유형이 지정되지 않은 매개변수 표시문자의 경우, 목표 변수 또는 표현식의 속성은 매개변수 표시문자의 컨텍스트에 따라 결정됩니다.

  매개변수 표시문자 P에 해당하는 입력 변수 또는 표현식으로 V를 지정하도록 하십시오. V 값은 컬럼에 값을 지정하는 규칙에 따라 P에 대한 목표 변수에 지정됩니다. 따라서 다음과 같습니다.

  – V는 목표와 호환성이 있어야 합니다.
  – V가 문자열인 경우 그 길이는 목표 속성 길이보다 작아야 합니다.
  – V가 숫자인 경우 정수 부분의 절대값은 목표의 정수 부분의 최대 절대값 길이보다 작아야 합니다.
  – V 속성이 목표 속성과 같지 않은 경우 그 값은 목표 속성에 일치하도록 변환됩니다.

  PREPARE문이 실행될 때 P 대신 사용된 값은 P에 대한 목표 변수 또는 P에 대한 목표 표현식의 값입니다. 예를 들어, V가 CHAR(6)이고 목표가 CHAR(8)인 경우 P 대신에 사용된 값은 두 개의 공백이 채워진 V의 값입니다.
- 동적 CALL문의 경우, PREPARE문이 실행된 후 각 OUT 및 INOUT 인수에 대해 리턴된 값이 인수에 사용된 출력 매개변수 표시문자에 해당하는 지정 목표에 지정됩니다. 입력된 매개변수 표시문자의 경우 목표 변수 속성은 CAST 스펙으로 지정된 속성입니다. 유형이 지정되지 않은 매개변수 표시문자의 경우 목표 변수의 속성은 프로시저의 매개변수 정의가 지정한 속성입니다.

  V가 매개변수 표시문자 P에 해당하는 출력 지정 목표를 나타낸다고 가정해 보십시오. 이때 매개변수 표시문자 P는 프로시저의 인수 A에 사용됩니다. 컬럼에서 값을 검색하는 규칙에 따라 값 A가 V에 지정됩니다. 따라서 다음과 같습니다.

  – V는 A와 호환성이 있어야 합니다.
  – V가 문자열이면 문자열 길이가 A의 길이보다 길어야 합니다. 그렇지 않으면 값 A가 절단됩니다.
  – V가 숫자면 V의 정수 부분에 해당하는 최대 절대값이 A의 정수 부분에 해당하는 절대값보다 크거나 같아야 합니다.
  – V의 속성이 A의 속성과 다르면 V의 속성에 맞게 A의 속성이 변환됩니다.
- ***동적 SQL문 캐시:*** 정적 SQL문이 첫 번째로 참조되거나 동적 SQL문이 첫 번째로 준비될 경우 동적 및 정적 SQL문을 실행하는 데 필요한 정보가 데이터베이스 패키

지 캐시에 배치됩니다. 이 정보는 데이터베이스를 종료할 때, 다른 명령문에서 캐시 스페이스를 필요로 할 때, 정보가 유효하지 않을 때까지 패키지 캐시에 남아 있습니다.

SQL문이 실행되거나 준비되면 응용프로그램 발행 요청과 관련된 패키지 정보는 시스템 카탈로그에서 패키지 캐시로 로드됩니다. 개개의 SQL문에 대해 실제로 실행되는 섹션은 캐시에 위치됩니다. 명령문이 첫 번째로 참조되거나 동적 SQL 섹션이 작성된 후에 직접 캐시에 위치되면, 시스템 카탈로그로에서 정적 SQL 섹션이 읽혀지고 패키지에 위치됩니다. 동적 SQL 섹션은 PREPARE 또는 EXECUTE IMMEDIATE와 같은 명시적 명령문으로 작성할 수 있습니다. 일단 작성되면 원래 섹션이 스페이스 관리 이유로 삭제되거나 환경 변경으로 인해 유효하지 않게 될 경우 동적 SQL문에 대해 섹션은 시스템에 의해 내재적으로 준비된 명령문으로 재작성할 수 있습니다.

각 SQL문은 데이터베이스 레벨에서 캐시되며 응용프로그램들 사이에서 공유할 수 있습니다. 정적 SQL문은 동일한 응용프로그램들 사이에서 공유할 수 있습니다. 동적 SQL문은 정확히 동일한 명령문 텍스트와 동일한 컴파일 환경으로 응용프로그램들 사이에서 공유할 수 있습니다. 응용프로그램이 발행한 각 SQL문의 텍스트는 내재적 준비가 필요한 경우에 사용하기 위해 응용프로그램에 로컬로 캐시됩니다. 응용프로그램의 PREPARE문은 각각 하나의 명령문을 캐시할 수 있습니다. 응용프로그램의 모든 EXECUTE IMMEDIATE문은 캐시 스페이스를 공유하며 모든 EXECUTE IMMEDIATE문에 대해 한 번에 하나의 캐시 명령문만 존재합니다. 동일한 PREPARE 또는 임의의 EXECUTE IMMEDIATE문이 항상 다른 SQL문으로 여러 번 발행되면 마지막 명령문만이 재사용을 위해 캐시에 보관됩니다. 캐시의 선택적 사용으로 응용프로그램이 한 번 시작되고 나면 필요에 따라 EXCUTE 또는 OPEN 문과 서로 다른 번호의 PREPARE문이 발행됩니다.

캐시된 동적 SQL문을 사용시 일단 명령문이 작성되면 명령문을 다시 준비할 필요없이 여러 작업 단위(UOW)에서 재사용할 수 있습니다. 시스템은 환경이 변경되는 경우 필요에 따라 명령문을 다시 컴파일합니다.

다음과 같은 환경 또는 데이터 오브젝트의 변경으로 인해 내재적으로 준비될 동적 명령문이 PREPARE, EXECUTE, EXECUTE IMMEDIATE 또는 OPEN 요청에서 캐시됩니다.

- ALTER FUNCTION
- ALTER METHOD
- ALTER NICKNAME
- ALTER PROCEDURE
- ALTER SERVER
- ALTER TABLE

- ALTER TABLESPACE
- ALTER TYPE
- CREATE FUNCTION
- CREATE FUNCTION MAPPING
- CREATE INDEX
- CREATE METHOD
- CREATE PROCEDURE
- CREATE TABLE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- DROP(모든 오브젝트)
- 테이블 또는 인덱스의 RUNSTATS
- 뷰가 작동 불능이 되도록 하는 모든 조치
- UPDATE(시스템 카탈로그 테이블내 통계)
- SET CURRENT DEGREE
- SET PATH
- SET QUERY OPTIMIZATION
- SET SCHEMA
- SET SERVER OPTION

다음 목록은 캐시된 동적 SQL문으로부터 예상할 수 있는 사항을 요약한 것입니다.

- *PREPARE 요청:* 이 후의 동일한 명령문 준비로 인해 섹션이 여전히 유효한 경우 명령문을 컴파일하는 데 비용이 발생하지 않게 됩니다. 현재의 캐시 섹션에 대해 산정한 비용과 카디낼리티가 리턴됩니다. 이 값은 동일한 SQL문에 대해 이전에 PREPARE로부터 리턴된 값과 다릅니다. COMMIT 또는 ROLLBACK문 이후에 PREPARE문을 발행할 필요가 없습니다.
- *EXECUTE 요청:* EXECUTE문이 이후에 원래의 PREPARE 이후에 유효하지 않게 되면, 명령문을 내재적으로 준비하는 데 비용이 발생합니다. 내재적으로 준비된 섹션은 현재 환경에서는 사용되지만 원래의 PREPARE문 환경에서는 사용되지 않습니다.
- *EXECUTE IMMEDIATE 요청:* 동일한 명령에 대해 이 후의 EXECUTE IMMEDIATE문은 섹션이 여전히 유효한 경우 명령문을 컴파일하는 데 비용이 발생하지 않습니다.

- *OPEN 요청:* 동적으로 정의된 커서에 대한 OPEN 요청은 원래의 PREPARE 이후 유효하지 않게 되었으면 명령문을 내재적으로 준비하는 데 비용이 발생합니다. 내재적으로 준비된 섹션은 현재 환경에서는 사용되지만 원래의 PREPARE문 환경에서는 사용되지 않습니다.
- *FETCH 요청:* 어떠한 작동의 변경도 없을 것입니다.
- *ROLLBACK:* 작업 단위가 롤백 조작에 영향을 미치는 동안 준비 또는 내재적으로 준비된 동적 SQL문만 무효화됩니다.
- *COMMIT:* 동적 SQL문은 무효화되지 않지만 확보된 잠금은 해제됩니다. WITH HOLD 커서로 정의되지 않은 커서는 닫히고, 그 잠금은 해제됩니다. 열려 있는 WITH HOLD 커서는 패키지와 섹션 잠금을 유지하여 커미트 프로세스 동안이나 후에 활성 섹션을 보호합니다.

내재적 준비 중 오류가 발생하면 내재적 준비를 일으킨 요청에 대해 오류를 리턴합니다.

예:

예 1: 아래 C의 예에서 매개변수 표시문자를 가진 INSERT문이 준비되고 실행됩니다. 호스트 변수 h1 - h4는 TDEPT 형식에 해당합니다.

```
     strcpy (s,"INSERT INTO TDEPT VALUES(?,?,?,?)");
EXEC SQL PREPARE DEPT_INSERT FROM :s;
     .
     .
  (Check for successful execution and put values into :h1, :h2, :h3, :h4)
     .
     .
     EXEC SQL EXECUTE DEPT_INSERT USING :h1, :h2,
  :h3, :h4;
```

예 2: 이 EXECUTE문이 SQLDA를 사용합니다.

```
EXECUTE S3 USING DESCRIPTOR :sqlda3
```

예 3: 직원에게 상여금을 지급하는 프로시저가 있다고 가정합니다.

```
CREATE PROCEDURE GIVE_BONUS (IN EMPNO INTEGER,
                             IN DEPTNO INTEGER,
                             OUT CHEQUE INTEGER,
                             INOUT BONUS DEC(6,0))
...
```

C 응용프로그램에서 동적으로 프로시저를 호출합니다. 프로시저는 입력으로 다음 호스트 변수를 사용합니다.

- *employee*: 직원 ID 번호
- *dept*: 부서 번호
- *bonus*: 직원의 희망 상여금

이 프로시저는 호스트 변수에 다음 값을 리턴합니다.

- *cheque_no*: 수표의 ID 번호
- *bonus*: 조정 후 실제 상여금 액수

```
   strcpy (s, "CALL GIVE_BONUS(?, ?, ?, ?)");
EXEC SQL PREPARE DO_BONUS FROM :s;
  .
  .
  /* Check for successful execution and put values into
        :employee, :dept, and :bonus */
  .
  .
  EXEC SQL EXECUTE DO_BONUS INTO :cheque_no, :bonus
                            USING :employee, :dept, :bonus;
  .
  .
  /* Check for successful execution and process the
        values returned in :cheque_no and :bonus */
```

EXECUTE IMMEDIATE

EXECUTE IMMEDIATE 명령문:

- 명령문의 문자열 양식에서 SQL 명령문의 실행 파일 양식을 준비합니다.
- SQL문을 실행합니다.

EXECUTE IMMEDIATE는 PREPARE와 EXECUTE문의 기본 함수를 조합합니다. 이는 호스트 변수와 매개변수 표시문자가 없는 SQL문을 준비 및 실행하는 데 사용됩니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

권한 부여 규칙은 지정된 SQL문에서 정의됩니다.

명령문의 권한 부여 ID는 바인드 옵션 DYNAMICRULES의 영향을 받을 수 있습니다.

구문

설명

expression

실행될 명령문 문자열을 리턴하는 표현식입니다. 표현식은 최대 명령문 *크기* 2,097,152바이트보다 작은 문자열 유형을 리턴해야 합니다. CLOB(2097152)는 최대 *크기* 명령문을 포함할 수 있으나 VARCHAR은 포함할 수 없음에 유의하십시오.

명령문 문자열은 다음 SQL문 중 하나여야 합니다.

- ALTER
- CALL
- COMMENT
- COMMIT
- 복합 SQL(컴파일된)
- 복합 SQL(인라인된)
- CREATE
- DECLARE GLOBAL TEMPORARY TABLE

- DELETE
- DROP
- EXPLAIN
- FLUSH EVENT MONITOR
- FLUSH PACKAGE CACHE
- GRANT
- INSERT
- LOCK TABLE
- MERGE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME
- REVOKE
- ROLLBACK
- SAVEPOINT
- SET COMPILATION ENVIRONMENT
- SET CURRENT DECFLOAT ROUNDING MODE
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT FEDERATED ASYNCHRONY
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT IMPLICIT XMLPARSE OPTION
- SET CURRENT ISOLATION
- SET CURRENT LOCALE LC_TIME
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT MDC ROLLOUT MODE
- SET CURRENT OPTIMIZATION PROFILE
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ROLE(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- SET ENCRYPTION PASSWORD

- SET EVENT MONITOR STATE(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- SET SESSION AUTHORIZATION
- SET 변수
- TRANSFER OWNERSHIP(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- TRUNCATE(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- UPDATE

명령문 문자열은 매개변수 표시문자나 호스트 변수에 대한 참조를 포함하지 않아야 하며, EXEC SQL로 시작하지 않아야 합니다. 또한 세미콜론(;)을 포함하여 복합 블록에서 명령문을 구분할 수 있는 복합 SQL문을 제외하고는 명령문 종료문자를 포함하지 않아야 합니다. 복합 SQL문은 일부 CREATE 및 ALTER문에서 사용되므로, 세미콜론도 포함할 수 있습니다.

EXECUTE IMMEDIATE문이 실행될 때, 지정된 명령문 문자열이 구문 분석되고 오류가 검사됩니다. SQL문이 유효하지 않은 경우, 이는 실행되지 않으며 실행을 예방하는 오류 조건이 SQLCA에 보고됩니다. SQL문이 유효하나 실행 중 오류가 발생한 경우, 해당 오류 조건이 SQLCA에 보고됩니다.

주

- 명령문 캐시는 EXECUTE IMMEDIATE문의 실행 방식에 영향을 미칩니다.

예 :

C 프로그램 명령문을 사용하여 SQL문을 호스트 변수 qstring(char[80])으로 이동하고, 호스트 변수 qstring에 있는 모든 SQL문을 준비 및 실행하십시오.

```
if ( strcmp(accounts,"BIG") == 0 )
  strcpy(qstring,"INSERT INTO WORK_TABLE SELECT *
    FROM EMP_ACT WHERE ACTNO < 100");
else
  strcpy(qstring,"INSERT INTO WORK_TABLE SELECT *
    FROM EMP_ACT WHERE ACTNO >= 100");
  .
  .
  .
EXEC SQL  EXECUTE IMMEDIATE :qstring;
```

EXPLAIN

EXPLAIN문은 제공된 설명 가능한 명령문에 선택된 액세스 플랜에 관한 정보를 보관하고 이 정보를 Explain 테이블에 둡니다.

*Explain 가능한 명령문*은 유효한 XQuery문 또는 다음 SQL문 중의 하나일 수 있습니다. CALL, Compound SQL(Dynamic), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES 또는 VALUES INTO.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

설명될 명령문은 실행되지 않습니다.

권한 부여

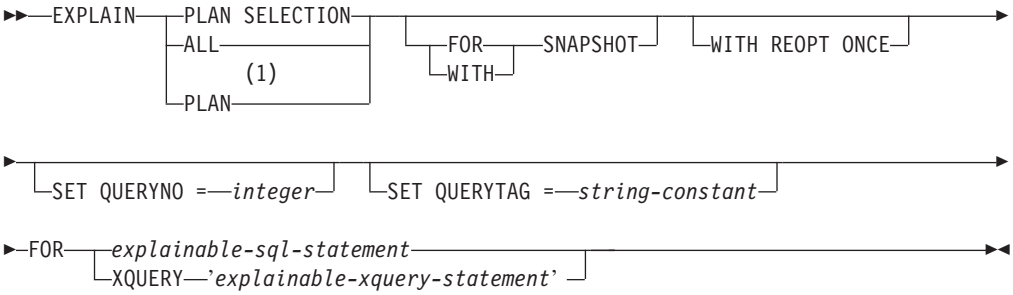
명령문의 권한 부여 ID가 보유하는 특권에는 다음이 포함되어야 합니다.

- 예외 테이블의 INSERT 특권
- DATAACCESS 권한

다음 중 최소한 하나는 포함해야 합니다.

- EXPLAIN문에 지정된 Explain 가능한 명령문을 실행하는 데 필요한 모든 특권(예를 들어 Explain 가능한 명령문으로써 DELETE문이 사용된 경우, DELETE문 설명 시 DELETE문에 대한 권한 부여 규칙이 적용됨)
- EXPLAIN 권한
- SQLADM 권한
- DBADM 권한

구문



주:

1 PLAN 옵션은 z/OS EXPLAIN문에 대한 기존 DB2의 구문 허용 한도에 대해서

만 지원됩니다. PLAN 테이블은 없습니다. PLAN을 지정하는 것은 PLAN SELECTION을 지정하는 것과 같습니다.

설명

PLAN SELECTION

쿼리 컴파일의 플랜 선택 단계 정보가 Explain 테이블로 삽입된다는 것을 나타냅니다.

ALL

ALL을 지정하는 것은 PLAN SELECTION을 지정하는 것과 같습니다.

PLAN

PLAN 옵션은 다른 시스템에서 기존의 데이터베이스 응용프로그램에 대해 구문 허용 한도를 제공합니다. PLAN을 지정하는 것은 PLAN SELECTION을 지정하는 것과 같습니다.

FOR SNAPSHOT

이 절은 Explain 스냅샷만이 취해져 EXPLAIN_STATEMENT 테이블의 SNAPSHOT 컬럼에 위치한다는 것을 나타냅니다. EXPLAIN_INSTANCE 및 EXPLAIN_STATEMENT 테이블에 존재하는 것 이외의 다른 Explain 정보가 보관되지 않습니다.

Explain 스냅샷 정보는 Visual Explain에 사용하기 위한 것입니다.

WITH SNAPSHOT

이 절은 일반적인 Explain 정보 외에도 Explain 스냅샷이 취해진다는 것을 나타냅니다.

EXPLAIN문의 디폴트 작동은 Explain 스냅샷이 아닌 일반적인 Explain 정보만 수집하는 것입니다.

Explain 스냅샷 정보는 Visual Explain에 사용하기 위한 것입니다.

디폴트값(FOR SNAPSHOT 또는 WITH SNAPSHOT이 지정되지 않았음)

Explain 테이블에 Explain 정보를 둡니다. Visual Explain에 사용할 스냅샷이 취해지지 않습니다.

WITH REOPT ONCE

이 절은 지정된 *Explain 가능한 명령문*을 이전에 REOPT ONCE로 이 명령문을 재최적화할 때 사용된 호스트 변수, 매개변수 표시문자 또는 특수 레지스터 또는 전역 변수의 값을 사용하여 재최적화시키도록 표시합니다. Explain 테이블은 새 액세스 플랜으로 채워집니다. 사용자가 DBADM 권한을 가지고 있거나 데이터베이스 레지스트리 변수 DB2_VIEW_REOPT_VALUES가 YES로 설정된 경우 EXPLAIN_PREDICATE 테이블도 명령문을 다시 최적화하는 데 사용된 값으로 채워집니다.

SET QUERYNO = *integer*

EXPLAIN_STATEMENT 테이블의 QUERYNO 컬럼을 통해 *integer*를 *explainable statement*에 연결시키십시오. 제공되는 정수 값은 양수여야 합니다.

이 절이 동적 EXPLAIN문에 지정되어 있지 않으면 (1)의 디폴트값이 지정됩니다. 정적 EXPLAIN문의 경우 지정되지 않은 디폴트값은 프리컴파일러가 지정한 명령문 번호입니다.

SET QUERYTAG = *string-constant*

EXPLAIN_STATEMENT 테이블의 QUERYNO 컬럼을 통해 *string-constant*를 *Explain 가능한 명령문*에 연결시키십시오. *string-constant*는 최대 20바이트의 문자열일 수 있습니다. 제공된 값의 길이가 20바이트보다 작은 경우 값은 필수 길이까지 오른쪽에 공백이 채워집니다.

이 절이 EXPLAIN문에 지정되어 있지 않으면, 디폴트값으로 공백이 사용됩니다.

FOR *explainable-sql-statement*

설명될 SQL문을 지정합니다. 이 명령문은 유효한 CALL, Compound SQL(Dynamic), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES 또는 VALUES INTO SQL 명령문일 수 있습니다. EXPLAIN문이 프로그램에 임베드되어 있으면 *explainable-sql-statement*에는 호스트 변수(프로그램에 정의되어 있어야 함)에 대한 참조가 포함될 수 있습니다. 마찬가지로 EXPLAIN이 동적으로 준비되고 있는 경우 *explainable-sql-statement*에는 매개변수 표시문자가 포함될 수 있습니다.

*explainable-sql-statement*는 EXPLAIN문과 독립적으로 준비되고 실행될 수 있는 유효한 SQL문이어야 합니다. 이것은 명령문 이름이나 호스트 변수여서는 안됩니다. CLP를 통해 정의된 커서를 참조하는 SQL문은 이 명령문에 사용하기에 적합하지 않습니다.

응용프로그램 내의 동적 SQL을 Explain하려면 전체 EXPLAIN문이 동적으로 준비되어야 합니다.

FOR XQUERY *'explainable-xquery-statement'*

설명될 XQUERY문을 지정합니다. 이 명령문은 모든 유효한 XQUERY문이 될 수 있습니다.

EXPLAIN문이 프로그램에 임베드되는 경우, 호스트 변수가 top 레벨 XQUERY 문에서 사용되지는 않지만 XMLEXISTS 술어 또는 XMLTABLE 함수가 XMLQUERY 함수를 통해 호스트 변수를 전달하면 *'explainable-xquery-statement'*는 호스트 변수 참조를 포함할 수 있습니다. 호스트 변수는 프로그램에서 정의되어야 합니다.

마찬가지로 EXPLAIN이 동적으로 준비되고 있는 경우, 호스트 변수 전달시 사용되는 것과 동일한 제한사항이 뒤에 수반되면 *'explainable-xquery-statement'*에는 매개변수 표시문자가 포함될 수 있습니다.

그외에 DB2 XQUERY 함수 db2-fn:sqlquery를 사용하여 SQL문을 호스트 변수 및 매개변수 표시문자 참조로 임베드할 수 있는 방법이 있습니다.

*'explainable-xquery-statement'*는 EXPLAIN문과 독립적으로 준비되고 실행될 수 있는 유효한 XQUERY문이어야 합니다. CLP를 통해 정의된 커서를 참조하는 쿼리 명령문은 이 명령문에 사용하기에 적합하지 않습니다.

주

Explain 기능은 데이터 채우기 중인 Explain 테이블을 규정할 때 다음의 ID를 스키마로 사용합니다.

- 동적 SQL의 경우 세션 권한 부여 ID
- 정적 SQL의 경우 명령문 권한 부여 ID

스키마는 Explain 테이블 세트 또는 다른 스키마의 Explain 테이블 세트를 가리키는 별명과 연결될 수 있습니다. 만일 해당 스키마에 Explain 테이블이 없으면, Explain 기능은 SYSTOOLS 스키마의 Explain 테이블을 확인하고 그 테이블을 사용하려고 시도합니다.

다음 표는 스냅샷 키워드와 Explain 정보와의 상호 관계를 나타냅니다.

지정된 키워드	설명 정보 캡쳐 여부	**Visual Explain**에 대한 스냅샷 여부
없음	YES	없음
FOR SNAPSHOT	아니오	예
WITH SNAPSHOT	YES	예

FOR SNAPSHOT 및 WITH SNAPSHOT절이 모두 지정되지 않은 경우 Explain 스냅샷을 취할 수 없습니다.

Explain 테이블은 EXPLAIN문을 호출하기 전에 사용자가 작성해야 합니다. 이 명령문에 의해 생성된 정보는 명령문이 컴파일될 때 지정된 스키마에 있는 Explain 테이블에 저장됩니다.

제공된 *explainable statement*의 컴파일 중 오류가 발생하면 Explain 테이블에 정보가 저장되지 않습니다.

*explainable statement*에 대해 생성된 액세스 플랜은 보관되지 않으므로 나중에 호출할 수 없습니다. EXPLAIN문 자체가 컴파일될 때 *explainable statement*에 대한 Explain 정보가 삽입됩니다.

정적 EXPLAIN 쿼리 명령문의 경우 정보는 바인드할 때와 명시적 리바인드 중에 Explain 테이블에 삽입됩니다. 프리컴파일되는 동안 정적 EXPLAIN문은 수정된 응용프로그램 소스 파일에 주석 처리됩니다. 바인드될 때 EXPLAIN문은

SYSCAT.STATEMENTS 카탈로그에 저장됩니다. 패키지가 실행될 때 EXPLAIN문은 실행되지 않습니다. 응용프로그램에 있는 모든 명령문에 대한 섹션 번호는 순차적이어야 하며 EXPLAIN문이 포함되어야 함에 유의하십시오. 정적 EXPLAIN문 사용의 대체 방법은 EXPLAIN 및 EXPLSNAP BIND 또는 PREP 옵션의 조합을 사용하는 것입니다. 정적 EXPLAIN문을 사용하여 여러 개 중 하나의 정적 쿼리 명령문에 Explain 테이블이 채워지게 할 수 있습니다. 적절한 EXPLAIN문 구문을 사용하여 목표 명령문에 접두어를 붙인 다음 Explain BIND 또는 PREP 옵션을 사용하지 않고 응용프로그램을 바인드하십시오. 또한 EXPLAIN문은 실제 Explain 호출시 QUERYNO 또는 QUERYTAG 필드를 설정하는 것이 유리한 경우에 사용될 수도 있습니다.

SQL 프로시저의 정적 EXPLAIN문은 프로시저가 컴파일될 때 평가됩니다.

증분식 바인드 EXPLAIN 쿼리 명령문의 경우 Explain 테이블은 EXPLAIN문이 컴파일을 위해 제출될 때 채워지게 됩니다. 패키지가 실행될 때 EXPLAIN문은 처리를 수행하지 않습니다(명령문이 성공될지라도). Explain 테이블의 데이터를 채우는 경우 데이터 채우기 중에 사용되는 Explain 테이블 규정자와 권한 부여 ID는 패키지 소유자의 규정자 및 권한 부여 ID입니다. 또한 EXPLAIN문은 실제 Explain 호출시 QUERYNO 또는 QUERYTAG 필드를 설정하는 것이 유리한 경우에 사용될 수도 있습니다.

동적 EXPLAIN문의 경우 Explain 테이블은 EXPLAIN문이 컴파일을 위해 제출될 때 채워지게 됩니다. EXPLAIN문은 PREPARE문으로 준비될 수 있으나 실행되는 경우 명령문이 성공적일지라도 처리되지 않습니다. 동적 EXPLAIN문을 실행하는 대신 CURRENT EXPLAIN MODE와 CURRENT EXPLAIN SNAPSHOT 특수 레지스터를 사용하여 동적 쿼리 명령문을 설명할 수 있습니다. EXPLAIN문은 실제 EXPLAIN 호출시 QUERYNO 또는 QUERYTAG 필드를 설정하는 것이 유리한 경우에 사용되어야 합니다.

REOPT 바인드 옵션을 ONCE로 설정하고 CURRENT EXPLAIN MODE 또는 CURRENT EXPLAIN SNAPSHOT 특수 레지스터를 REOPT로 설정한 경우 호스트 변수, 특수 레지스터 또는 매개변수 표시문자 또는 전역 변수가 포함된 정적 및 동적 쿼리 명령문을 실행하면 명령문이 재최적화될 때만 명령문에 대한 Explain 정보가 캡처됩니다. 그 외에 REOPT 바인드 옵션을 ALWAYS로 설정하면 이 명령문이 실행될 때마다 Explain 정보가 캡처되는 방법이 있습니다.

예:

예 1: 간단한 SELECT문을 Explain하고 QUERYNO = 13으로 태그를 붙이십시오.

```
EXPLAIN PLAN SET QUERYNO = 13
  FOR SELECT C1
  FROM T1
```

예 2: 간단한 SELECT문을 Explain하고 QUERYTAG = 'TEST13'을 붙이십시오.

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
  FOR SELECT C1
  FROM T1
```

예 3: 간단한 SELECT문을 Explain하고 QUERYNO = 13 및 QUERYTAG = 'TEST13'으로 태그를 붙이십시오.

```
EXPLAIN PLAN SELECTION SET QUERYNO = 13 SET QUERYTAG = 'TEST13'
  FOR SELECT C1
  FROM T1
```

예 4: Explain 테이블이 존재하지 않으면 Explain 정보를 확보하십시오.

```
EXPLAIN ALL FOR SELECT C1
  FROM T1
```

이 명령문은 Explain 테이블이 정의되어 있지 않으므로 실패합니다(SQLSTATE 42704).

예 5: 다음 명령문은 패키지 캐시에서 찾을 수 있고 이미 REOPT ONCE를 사용하여 컴파일된 경우 성공합니다.

```
EXPLAIN ALL WITH REOPT ONCE FOR SELECT C1
  FROM T1
  WHERE C1 = :<host variable>
```

예 6: 다음의 예는 XML 컬럼의 대소문자 구분 이름을 인수로 취하여 XML 컬럼 값의 병합인 XML 시퀀스를 리턴하는 db2-fn:xmlcolumn 함수를 사용합니다.

BUSINESS.CUSTOMER 테이블을 INFO라는 XML 컬럼으로 고려하십시오. 모든 문서를 INFO 컬럼으로부터 리턴하는 단순 XQuery는 다음과 같습니다.

```
EXPLAIN PLAN SELECTION
         FOR XQUERY 'db2-fn:xmlcolumn ("BUSINESS.CUSTOMER.INFO")'
```

컬럼 값이 널(NULL)이면 해당 행에 대해 돌아오는 리턴 시퀀스는 공백이 됩니다.

FETCH

FETCH문은 결과 테이블의 다음 행에 커서를 위치시키고, 목표 변수에 그 행의 값을 지정합니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 명령행 처리기를 사용하여 호출 시, *cursor-name*을 따르는 구문은 선택적이며 SQL 구문과 다릅니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

권한 부여

각 전역 변수를 *cursor-variable-name*으로 사용하거나 *array-index*에 대한 표현식에서 사용하는 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

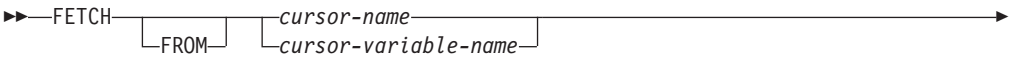
- 모듈에 정의되지 않은 전역 변수에 대한 READ 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

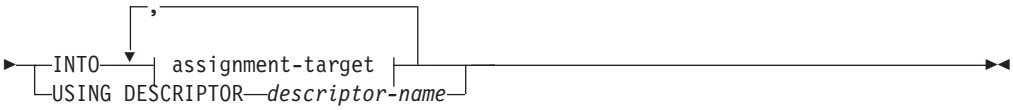
*assignment-target*으로 사용되는 각 전역 변수의 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 WRITE 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

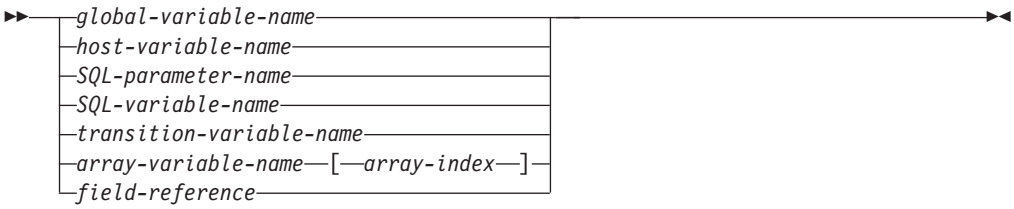
커서를 사용하기 위해 필요한 권한 부여에 대해서는 『DECLARE CURSOR』를 참조하십시오.

구문





assignment-target



설명

cursor-variable-name

페치(fetch)에 사용되는 커서를 식별합니다. cursor-variable-name은 범위 내에 있는 커서 변수를 식별해야 합니다. FETCH문이 실행될 때 *cursor-variable-name*의 기본 커서는 열린 상태에 있어야 합니다. *cursor-variable-name*을 사용한 FETCH 문은 복합 SQL(컴파일된)문 내에서만 사용될 수 있습니다.

INTO *assignment-target*

출력 값을 지정할 하나 이상의 목표를 식별합니다. 결과 행의 첫 번째 값은 목록의 첫 번째 목표에 지정되고, 두 번째 값은 두 번째 목표에, 이와 같은 식으로 지정됩니다. 목록에 나타난 순서대로 *assignment-target*에 값이 지정됩니다. 지정 오류가 발생하는 경우, 목표에 값이 지정되지 않고 목표에 더 이상 값이 지정되지 않습니다. 목표에 이미 지정된 값은 지정된 상태로 남아 있습니다.

모든 *assignment-target*의 데이터 유형이 행 유형이 아닌 경우, *assignment-targets* 수가 결과 컬럼 값보다 작으면 'W' 값이 SQLCA의 SQLWARN3 필드에 지정됩니다.

*assignment-target*의 데이터 유형이 행 유형이면, 하나의 *assignment-target*가 명확히 지정되어 있어야 하며(SQLSTATE 428HR), 컬럼 수는 행 유형의 필드 수와 일치되어야 하며, 페치된 행 컬럼의 데이터 유형이 행 유형의 해당 필드에 지정 가능해야 합니다(SQLSTATE 42821).

*assignment-target*의 데이터 유형이 배열 요소이면, 정확히 지정된 하나의 *assignment-target*이 있어야 합니다.

global-variable-name

지정 목표인 전역 변수를 식별합니다.

host-variable-name

지정 목표인 호스트 변수를 식별합니다. LOB 출력 값의 경우, 목표는 일반 호스트 변수(충분히 큰 경우), LOB 로케이터 변수 또는 LOB 파일 참조 변수가 될 수 있습니다.

SQL-parameter-name

지정 목표인 매개변수를 식별합니다.

SQL-variable-name

지정 목표인 SQL 변수를 식별합니다. SQL 변수는 사용하기 전에 선언해야 합니다.

transition-variable-name

전이 행에서 갱신될 컬럼을 식별합니다. *transition-variable-name*은 트리거의 주제 테이블에 있는 컬럼을 식별해야 하고, 새 값을 식별하는 상관 이름에 의해 선택적으로 규정됩니다.

array-variable-name

배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다.

[array-index]

지정의 목표가 되는 배열에 있는 요소를 지정하는 표현식. 일반 배열의 경우, *array-index* 표현식은 INTEGER(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다. 배열에 정의된 1과 최대 카디널리티 사이의 값이어야 합니다(SQLSTATE 2202E). 연관된 배열의 경우, *array-index* 표현식은 연관된 배열의 인덱스 데이터 유형(SQLSTATE 428H1)으로 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다.

field-reference

지정 목표인 행 유형 값 내에서 필드를 식별합니다. *field-reference*는 규정된 *field-name*으로 지정되어야 하며 이때 규정자는 필드가 정의된 행 값을 식별합니다.

USING DESCRIPTOR *descriptor-name*

0개 이상의 호스트 변수의 유효한 설명이 들어 있는 SQLDA를 식별합니다.

FETCH문이 처리되기 전에 사용자는 다음 필드를 SQLDA에 설정해야 합니다.

- SQLDA에 제공된 SQLVAR 어커런스 수를 표시하는 SQLN
- SQLDA에 할당된 스토리지의 바이트 수를 표시하는 SQLDABC
- 명령문 처리시 SQLDA에 사용된 변수의 수를 표시하는 SQLD
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수

SQLDA는 모든 SQLVAR 어커런스가 들어가기에 충분한 스토리지가 있어야 합니다. 따라서 SQLDABC의 값은 16 + SQLN * (N)보다 크거나 같아야 합니다. 여기서, N은 SQLVAR 어커런스의 길이입니다.

LOB 또는 구조화된 유형 결과 컬럼을 조절해야 하는 경우 모든 선택 목록 항목(또는 결과 테이블의 컬럼)에 대해 두 개의 SQLVAR 항목이 있어야 합니다.

SQLD는 0 이상이거나 SQLN 이하의 값으로 설정되어야 합니다.

SQLDA에 설명되어 있는 *n*번째 변수는 커서의 결과 테이블 *n*번째 컬럼에 해당합니다. 각 변수의 데이터 유형은 해당 컬럼과 일치해야 합니다.

특정 규칙에 따라 변수에 값이 지정됩니다. 변수의 수가 그 행의 값의 수보다 적은 경우, SQLDA의 SQLWARN3 필드는 'W'로 설정됩니다. 결과 컬럼의 수보다 많은 변수가 있는 경우 경고가 없음에 유의하십시오. 지정 오류가 발생하는 경우 변수에 값이 지정되지 않고 변수에 더 이상의 값이 지정되지 않습니다. 이미 변수에 지정된 값은 지정된 상태로 남아 있습니다.

주

- *커서 위치:* 열린 커서에는 다음의 세 가지 가능한 위치가 있습니다.
  - 행 앞에
  - 행에
  - 마지막 행 뒤에

  커서는 FETCH문의 결과로서 행에만 있을 수 있습니다. 커서가 현재 결과 테이블의 마지막 행 이후에 있는 경우 다음과 같습니다.
  - SQLCODE는 +100으로 설정되고, SQLSTATE는 '02000'으로 설정됩니다.
  - 커서는 마지막 행 후에 위치합니다.
  - 지정 목표에 값이 지정되지 않습니다.

  커서가 현재 행 앞에 위치한 경우, 커서가 그 행에 재위치되며, 값은 INTO 또는 USING 절에 의해 지정된 대로 목표에 지정됩니다.

  커서가 현재 마지막 행이 아닌 행에 위치하면, 다음 행에 커서가 재위치되고, 그 행의 값은 INTO 또는 USING절에 의해 지정된 대로 목표에 지정됩니다.

  커서가 행에 있는 경우, 해당 행을 그 커서의 현재 행이라고 합니다. UPDATE 또는 DELETE문에 참조되는 커서는 행에 위치해야 합니다.

  커서를 예측할 수 없는 상태로 만드는 오류가 발생할 수 있습니다.
- FETCH문을 통해 로케이터를 보유하는 것이 필요없는 상황에서 LOB 로케이터를 검색할 때 로케이터 자원이 제한되어 있으므로 다음 FETCH문을 발행하기 전에 FREE LOCATOR문을 발행하는 것이 좋습니다.
- FETCH에 대해 경고가 리턴되지 않을 수도 있습니다. 리턴된 경고가 이전에 페치된 행에 적용될 수도 있습니다. 이는 SYSTEM TEMPORARY 테이블이나 푸쉬다운 연산자 사용과 같은 최적화의 결과로서 발생합니다.
- 명령문 캐시는 EXECUTE IMMEDIATE문의 실행 방식에 영향을 미칩니다.
- DB2 CLI는 추가의 페치 능력을 지원합니다. 예를 들어, 커서의 결과 테이블이 읽기 전용일 경우 SQLFetchScroll() 함수를 사용하여 커서를 결과 테이블 내의 아무 지점에 위치시킬 수 있습니다.
- 갱신 가능한 커서의 경우 페치(fetch)될 때 행에서 잠금이 확보됩니다.

- 커서 정의가 SQL 데이터 변경문을 포함하거나 SQL 데이터를 수정하는 루틴을 호출하는 경우 페치(fetch) 조작 시의 오류로 인해 커서가 닫히는 경우 수정된 행이 롤백되지는 않습니다.

예:

예 1: 이 예에서 FETCH문은 SELECT문의 결과를 프로그램 변수 dnum, dname 및 mnum으로 페치합니다. 페치할 행이 더 이상 없으면 찾기 불가능 조건으로 돌아갑니다.

```
 EXEC SQL DECLARE C1 CURSOR FOR
SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
  WHERE ADMRDEPT = 'A00';

 EXEC SQL OPEN C1;

 while (SQLCODE==0) {
       EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
}

 EXEC SQL CLOSE C1;
```

예 2: 이 FETCH문은 SQLDA를 사용합니다.

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

FLUSH EVENT MONITOR

FLUSH EVENT MONITOR문은 이벤트 모니터 *event-monitor-name*과 연관된 모든 사용 중 모니터 유형에 대한 현재 데이터베이스 모니터 값을 이벤트 모니터 입출력 목표에 기록합니다. 그러므로 언제든 부분적 이벤트 레코드가 낮은 레코드 생성 빈도를 가진 이벤트 모니터(데이터베이스 이벤트 모니터와 같은)에 사용 가능합니다. 그러한 레코드는 *부분적 레코드* ID로 이벤트 모니터 로그에 기록됩니다.

이벤트 모니터가 지워질 때 사용 중인 내부 버퍼들이 이벤트 모니터 출력 오브젝트에 쓰여집니다.

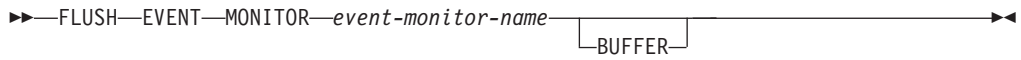
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 SQLADM 또는 DBADM 권한을 포함해야 합니다.

구문



설명

event-monitor-name

이벤트 모니터의 이름입니다. 이 이름은 한 부분의 이름입니다. 일반 ID입니다.

BUFFER

이벤트 모니터 버퍼가 쓰여짐을 나타냅니다. BUFFER를 지정하면 부분적 레코드가 생성되지 않습니다. 이벤트 모니터 버퍼에 이미 존재하는 데이터만이 쓰여집니다.

주

- 이벤트 모니터를 비우면 이벤트 모니터 값이 재설정되지 않을 것입니다. 이는 비우기가 수행되지 않은 경우 생성된 이벤트 모니터 레코드가 정상 모니터 이벤트가 트리거될 때에도 여전히 생성될 것임을 나타냅니다.
- FLUSH EVENT MONITOR문은 UNIT OF WORK 이벤트 모니터에 대해 이벤트가 생성되고 기록되도록 하지 않습니다.

FLUSH OPTIMIZATION PROFILE CACHE

동일한 최적화 프로파일을 사용하여 여러 개의 명령문을 컴파일할 수 있습니다. 최적화 프로파일이 한층 더 효율적으로 처리되도록, 최적화 프로파일은 명령문을 최적화하기 위해 처음 사용될 때 처리되며 출력은 최적화 프로파일 캐시에 저장됩니다. 최적화 프로파일에 대한 연속 참조에서는 최적화 프로파일 캐시에서 처리된 버전이 사용됩니다.

SYSTOOLS.OPT_PROFILE에 저장된 버전이 갱신된 경우 최적화 프로파일 캐시에서 최적화 프로파일을 제거해야 합니다. 캐시에서 이전 버전이 제거되면 최적화 프로파일을 사용하는 연속 명령문의 최적화에 새 버전이 사용됩니다.

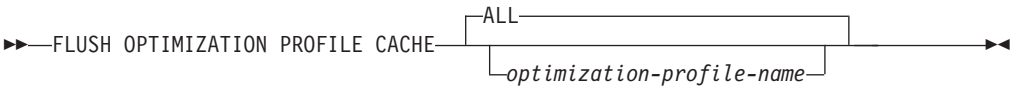
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 SQLADM 또는 DBADM 권한을 포함해야 합니다(SQLSTATE 42502).

구문



설명

optimization-profile-name

최적화 프로파일 캐시에서 플러시될 최적화 프로파일의 이름을 지정합니다. 지정된 이름이 규정되지 않은 경우 CURRENT DEFAULT SCHEMA 레지스터의 값은 내재된 규정자로 사용됩니다.

ALL

모든 활성 데이터베이스 파티션에 대한 모든 프로파일이 최적화 프로파일 캐시에서 플러시됨을 지정합니다.

주

- FLUSH OPTIMIZATION PROFILE CACHE문은 최적화 프로파일 캐시에서 모든 또는 단일 최적화 프로파일을 제거합니다. 또한 해당 최적화 프로파일에 대해 준비된, 캐시된 동적 SQL문이 논리적으로 무효화됩니다.
- 동일한 SQL문에 대한 다음 요청이 작성될 때 무효화된 동적 플랜에 대한 새 액세스 플랜이 다시 생성됩니다.

- 이 명령문에 의해 최적화 프로파일 캐시에서 제거된 최적화 프로파일을 참조하는 패키지는 새 액세스 플랜이 생성될 수 있도록 명시적으로 다시 바인드되어야 합니다.

예:

예 1: 최적화 프로파일 ″Rick″.″Foo″가 최적화 프로파일 캐시에서 플러시됩니다.

```
SET CURRENT SCHEMA = '"Rick"'
FLUSH OPTIMIZATION PROFILE CACHE "Foo"
```

예 2: 최적화 프로파일 JOHN.ALL이 최적화 프로파일 캐시에서 제거됩니다.

```
SET CURRENT SCHEMA = '"Rick"'
FLUSH OPTIMIZATION PROFILE CACHE JOHN.ALL
```

메시지

- 최적화 프로파일 캐시가 비어 있거나 지정된 최적화 프로파일(명시적으로 또는 내재적으로 지정된)이 최적화 프로파일 캐시에 존재하지 않는 경우 오류가 발생되지 않습니다.

FLUSH PACKAGE CACHE

FLUSH PACKAGE CACHE문은 현재 패키지 캐시에 있는 캐시된 동적 SQL문을 모두 제거합니다. 이 명령문은 캐시된 동적 SQL문이 논리적으로 무효화되도록 하고 같은 SQL문에 대한 다음 요청을 DB2가 내재적으로 컴파일하도록 합니다.

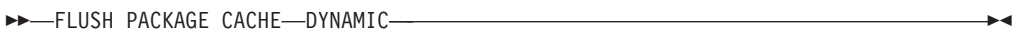
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유한 특권은 SQLADM 또는 DBADM 권한을 포함해야 합니다.

구문



주

- 이 명령문은 모든 활성 데이터베이스 파티션의 패키지 캐시에 있는 캐시된 모든 동적 SQL 항목에 영향을 줍니다.
- 캐시된 동적 SQL문은 무효화되기 때문에 FLUSH PACKAGE CACHE문을 실행할 때 항목이 사용되고 있지 않으면 캐시된 항목에 사용된 패키지 캐시 메모리가 해제됩니다.
- 현재 사용되고 있는 캐시된 동적 SQL문은 현재 사용자에게 더 이상 필요하지 않게 될 때까지 패키지 캐시에 남습니다. 같은 명령문을 다음에 사용하는 새 사용자는 DB2가 이 명령문을 내재적으로 준비하도록 하고 캐시된 동적 SQL문의 새 버전을 실행합니다.

FOR

FOR문은 테이블의 각 행에 대해 하나의 명령문이나 명령문 그룹을 실행합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

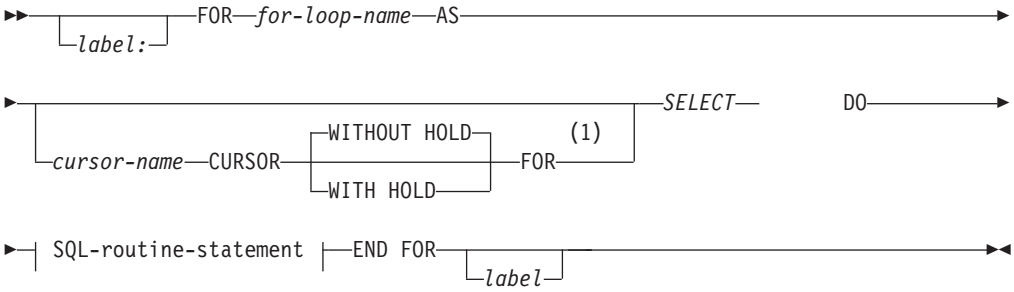
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

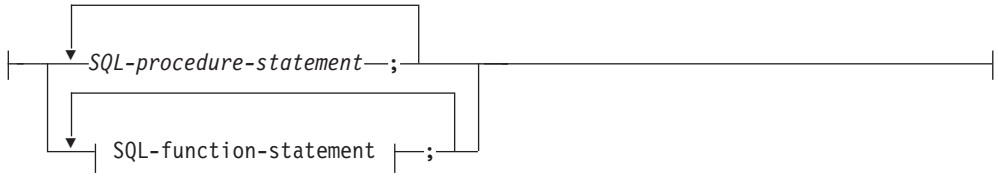
권한 부여

FOR문을 호출하기 위해 필요한 특권은 없습니다. 단, 명령문의 권한 부여 ID가 FOR 문에 임베디드(embedded)되는 SQL문을 호출하는 데 필요한 특권을 가지고 있어야 합니다. 커서를 사용하기 위해 필요한 권한 부여에 대해서는 『DECLARE CURSOR』를 참조하십시오.

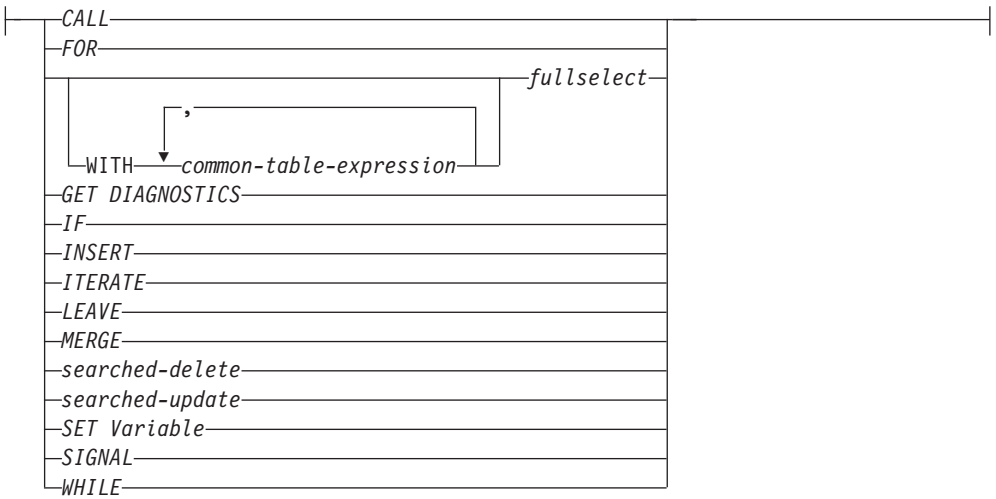
구문



SQL-routine-statement:



SQL-function-statement:



주:

1 이 옵션은 SQL 프로시저 또는 SQL 프로시저의 복합 SQL(컴파일된) 명령문의 컨텍스트에서만 사용할 수 있습니다.

설명

label

FOR문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우 해당 레이블을 LEAVE 및 ITERATE문에 사용할 수 있습니다. 끝 레이블이 지정된 경우 이 레이블은 시작 레이블과 같아야 합니다.

for-loop-name

FOR문을 구현하기 위해 생성된 내재된 복합 명령문의 레이블을 지정합니다. FOR문에서 ITERATE 또는 LEAVE문과 함께 사용할 수 없다는 점을 제외하고는 복합 명령문의 레이블 규칙을 따릅니다. *for-loop-name*은 지정된 *select-statement*에서 리턴하는 컬럼 이름을 규정하는 데 사용됩니다.

cursor-name

SELECT문의 결과 테이블에서 행을 선택하는 데 사용되는 커서를 이름 지정합니다. 지정하지 않으면 DB2는 고유 커서 이름을 생성합니다. WITHOUT HOLD 또는 WITH HOLD에 대한 설명은 『DECLARE CURSOR』를 참조하십시오.

SELECT

커서의 SELECT문을 지정합니다. 선택 목록의 모든 컬럼은 이름을 가져야 하며 같은 이름을 갖는 두 컬럼이 있을 수 없습니다.

트리거, 함수, 메소드 또는 복합 SQL(인라인된)문에서 *select-statement*는 선택적인 일반 테이블 표현식이 포함된 *fullselect*만으로 구성되어야 합니다.

SQL-procedure-statement

각 테이블 행에 대해 호출되는 하나 이상의 명령문을 지정합니다. *SQL-procedure-statement*는 SQL 프로시저의 컨텍스트에 있을 때 또는 복합 SQL(컴파일된) 명령

문 안에 있을 때만 적용할 수 있습니다. 『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

SQL-function-statement
각 테이블 행에 대해 호출되는 하나 이상의 명령문을 지정합니다. 별칭에 대한 검색 갱신, 검색 삭제 또는 INSERT 조작은 지원되지 않습니다. *SQL-function-statement*는 SQL 함수 또는 SQL 메소드의 컨텍스트에서만 적용할 수 있습니다.

규칙

- 선택 목록은 고유 컬럼 이름으로 구성되어야 하며 프로시저 작성 시 *select-statement*에 지정된 오브젝트가 존재하거나 오브젝트가 이전 SQL 프로시저 명령문으로 작성되어야 합니다.
- FOR문에 지정된 커서는 FOR문 외부에서 참조될 수 없으며 OPEN, FETCH 또는 CLOSE문에 지정될 수 없습니다.

예:

다음 예에서 FOR문은 employee 테이블 전체를 반복하는 데 사용됩니다. 테이블의 각 행에 대해 SQL 변수 fullname이 직원의 성, 쉼표, 이름, 공백 하나 및 중간 이니셜로 설정됩니다. fullname에 대한 각 값은 tnames 테이블에 삽입됩니다.

```
BEGIN ATOMIC
  DECLARE fullname CHAR(40);
  FOR v1 AS
    SELECT firstnme, midinit, lastname FROM employee
      DO
               SET fullname = lastname CONCAT ','
        CONCAT firstnme CONCAT ' ' CONCAT midinit;
      INSERT INTO tnames VALUES (fullname);
  END FOR;
END
```

FREE LOCATOR

FREE LOCATOR 문은 로케이터 변수와 로케이터 값 사이의 연관을 제거합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문

설명

LOCATOR *variable-name,* **...**

로케이터 변수 선언 규칙에 따라 선언해야 하는 하나 이상의 로케이터 변수를 식별합니다.

로케이터 변수는 현재 지정된 로케이터를 가지고 있어야 합니다. 즉, 로케이터는 해당 작업 단위 중에 지정되어 있어야 하고(CALL, FETCH, SELECT INTO 또는 VALUES INTO문으로) 그 뒤로 해제(FREE LOCATOR 문으로)되지 않아야 합니다. 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 0F001).

두 개 이상의 로케이터를 지정한 경우, 목록에 있는 다른 로케이터에서 발견된 오류에 관계없이 해제할 수 있는 모든 로케이터가 해제됩니다.

예 :

COBOL 프로그램에서 BLOB 로케이터 변수 TKN-VIDEO 및 TKN-BUF와 CLOB 로케이터 변수 LIFE-STORY-LOCATOR를 해제하십시오.

```
EXEC SQL
FREE LOCATOR :TKN-VIDEO, :TKN-BUF, :LIFE-STORY-LOCATOR
   END-EXEC.
```

GET DIAGNOSTICS

GET DIAGNOSTICS문은 이전에 실행된 SQL문에 대한 정보를 얻는 데 사용됩니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

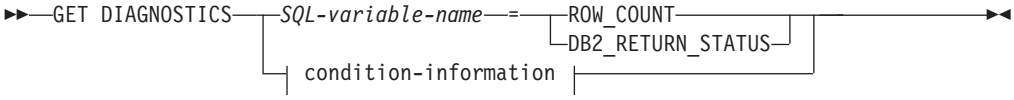
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

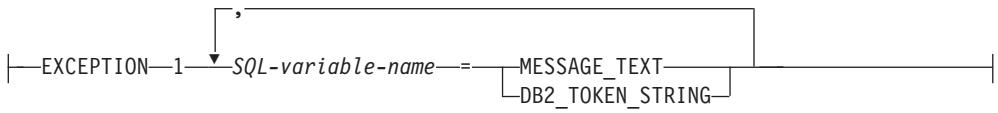
권한 부여

필요한 권한 없음

구문



condition-information:



설명

SQL-variable-name

지정 목표인 변수를 식별합니다. ROW_COUNT 또는 DB2_RETURN_STATUS가 지정된 경우, 변수는 정수 변수여야 하며 전역 변수가 아니어야 합니다. 그렇지 않으면, 변수는 CHAR 또는 VARCHAR여야 합니다. SQL 변수는 복합 명령문에 정의할 수 있습니다.

ROW_COUNT

이전 SQL문과 연관된 행의 수를 식별합니다. 이전 SQL문이 DELETE, INSERT, UPDATE문일 경우에는 ROW_COUNT가 해당 조작에 대해 규정된 행의 수를 식별합니다. 이전 명령문이 PREPARE문인 경우 ROW_COUNT는 준비된 명령문에서 *예상되는* 결과 행 수를 식별합니다.

DB2_RETURN_STATUS

해당 명령문이 상태를 리턴하는 프로시저를 호출하는 CALL문이었을 경우, 이전에

실행된 SQL문과 연관된 프로시저로부터 리턴된 상태 값을 식별하십시오. 이전 명령문이 그러한 명령문이 아닌 경우에는 리턴된 값이 아무 의미도 없으며 임의의 정수일 수 있습니다.

condition-information

이전에 실행된 SQL문에 대한 오류 정보나 경고 정보가 리턴되도록 지정합니다. 오류 정보가 필요할 경우에는 오류를 처리하는 핸들러에 GET DIAGNOSTICS문을 첫 번째 명령문으로 지정해야 합니다. 경고 정보가 필요하고 핸들러가 경고 조건을 제어하는 경우에는 핸들러에 GET DIAGNOSTICS문을 첫 번째 명령문으로 지정해야 합니다. 핸들러가 경고 조건을 제어하지 않을 경우에는 GET DIAGNOSTICS문이 실행되는 다음 명령문이어야 합니다. 이 옵션은 SQL 프로시저의 컨텍스트에서만 지정할 수 있습니다(SQLSTATE 42601).

MESSAGE_TEXT

이전에 실행된 SQL문에서 리턴된 오류 메시지 텍스트나 경고 메시지 텍스트를 식별합니다. 메시지 텍스트는 명령문이 처리된 데이터베이스 서버의 언어로 리턴됩니다. SQLCODE 값이 0인 상태로 명령문이 완료되면 VARCHAR 변수에 대해 빈 문자열이 리턴되거나 CHAR 변수에 대해 공백이 리턴됩니다.

DB2_TOKEN_STRING

이전에 실행된 SQL문에서 리턴된 오류 메시지 토큰이나 경고 메시지 토큰을 식별합니다. SQLCODE 값이 0인 상태로 명령문이 완료되거나 SQLCODE에 토큰이 없으면 VARCHAR 변수에 대해 빈 문자열이 리턴되거나 CHAR 변수에 대해 공백이 리턴됩니다.

주

- GET DIAGNOSTICS문이 진단 영역(SQLCA)의 내용을 변경하지 않습니다. SQLSTATE 또는 SQLCODE 특수 변수가 SQL 프로시저에서 선언된 경우, 이러한 변수들은 발행되는 GET DIAGNOSTICS문으로부터 리턴된 SQLSTATE 또는 SQLCODE로 설정됩니다.
- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  - DB2_RETURN_STATUS 대신 RETURN_STATUS를 지정할 수 있습니다.

예:

SQL 프로시저에서 GET DIAGNOSTICS문을 실행하여 갱신된 행의 수를 판별하십시오.

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
   LANGUAGE SQL
   BEGIN
      DECLARE SQLSTATE CHAR(5);
      DECLARE rcount INTEGER;
      UPDATE CORPDATA.PROJECT
          SET PRSTAFF = PRSTAFF + 1.5
        WHERE DEPTNO = deptnbr;
```

```
      GET DIAGNOSTICS rcount = ROW_COUNT;
-- At this point, rcount contains the number of rows that were updated.
...
   END
```

SQL 프로시저 내에서 사용자 실패를 나타내는 양수 값을 명시적으로 리턴하거나 음수 리턴 상태 값을 생성하는 SQL 오류를 발견할 수 있는 프로시저 TRYIT의 호출로부터 리턴된 상태 값을 처리하십시오. 프로시저가 정상 수행되는 경우에는 0 값을 리턴합니다.

```
CREATE PROCEDURE TESTIT ()
   LANGUAGE SQL
      A1:BEGIN
      DECLARE RETVAL INTEGER DEFAULT 0;
      ...
      CALL TRYIT;
      GET DIAGNOSTICS RETVAL = DB2_RETURN_STATUS;
      IF RETVAL <> 0 THEN
         ...
         LEAVE A1;
      ELSE
         ...
      END IF;
   END A1
```

GOTO

GOTO문은 SQL 프로시저 안에서 사용자 정의 레이블로 분기하는 데 사용됩니다.

호출

이 명령문은 SQL 프로시저에만 임베드될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

권한 부여

필요한 권한 없음

구문

설명

label

처리가 계속되는 레이블이 붙은 명령문을 지정합니다. 레이블이 붙은 명령문과 GOTO문은 동일한 범위에 있어야 합니다.

- GOTO문이 FOR문에서 정의되는 경우 *label*은 중첩된 FOR문 또는 중첩된 복합 명령문을 제외하고 동일한 FOR문 내부에서 정의되어야 합니다.
- GOTO문이 복합 명령문에서 정의되는 경우 *label*은 중첩된 FOR문 또는 중첩된 복합 명령문을 제외하고 동일한 복합 명령문 내부에서 정의되어야 합니다.
- GOTO문이 핸들러에서 정의되는 경우 *label*은 다른 범위 규칙을 따르면서 동일한 핸들러에서 정의되어야 합니다.
- GOTO문이 핸들러 밖에서 정의되는 경우 *label*은 핸들러 내부에서 정의되지 않아야 합니다.

*label*이 GOTO문이 도달할 수 있는 범위 안에서 정의되지 않으면 오류가 리턴됩니다(SQLSTATE 42736).

주

- GOTO문은 자제하여 사용할 것을 권장합니다. 이 명령문은 일반 처리 시퀀스와 간섭하므로 루틴을 읽고 유지보수하기에 더 어렵게 만듭니다. GOTO문을 사용하기 전에 IF 또는 LEAVE 같은 다른 명령문을 대신 사용하여 GOTO문이 필요없도록 할 수 있는지 확인하십시오.

예:

다음 복합 명령문에서 *rating* 및 *v_empno* 매개변수가 프로시저로 전달되며, 그 뒤에 출력 매개변수 *return_parm*을 날짜 기간으로 리턴합니다. 직원의 회사 고용 시간이 6 개월 미만인 경우 GOTO문은 제어를 프로시저 끝으로 전송하며 *new_salary*는 변경되지 않은 채로 남습니다.

```
CREATE PROCEDURE adjust_salary
  (IN v_empno CHAR(6),
  IN rating INTEGER)
  OUT return_parm DECIMAL (8,2))
  MODIFIES SQL DATA
  LANGUAGE SQL
  BEGIN
    DECLARE new_salary DECIMAL (9,2)
    DECLARE service DECIMAL (8,2)
      SELECT SALARY, CURRENT_DATE - HIREDATE
        INTO new_salary, service
        FROM EMPLOYEE
        WHERE EMPNO = v_empno
      IF service < 600
        THEN GOTO EXIT
      END IF
      IF rating = 1
        THEN SET new_salary = new_salary + (new_salary * .10)
      ELSE IF rating = 2
        THEN SET new_salary = new_salary + (new_salary * .05)
      END IF
      UPDATE EMPLOYEE
        SET SALARY = new_salary
        WHERE EMPNO = v_empno
      EXIT: SET return_parm = service
END
```

GRANT(데이터베이스 권한)

이러한 형태의 GRANT문은(데이터베이스 내의 특정 오브젝트에 적용되는 특권이 아닌) 전체 데이터베이스에 적용되는 권한을 부여합니다.

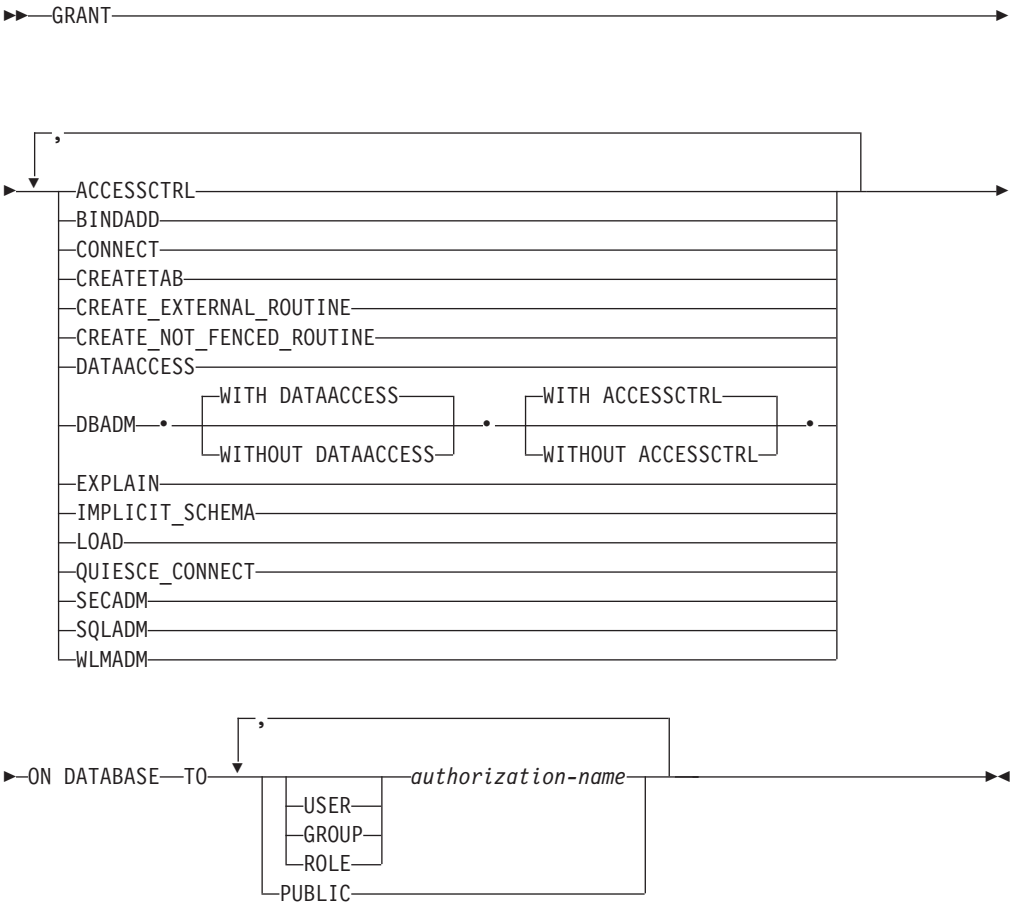
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

ACCESSCTRL, DATAACCESS, DBADM 또는 SEADM 권한을 부여하려면 SECADM 권한이 필요합니다. 다른 권한을 부여하려면, ACCESSCTRL 또는 SECADM 권한이 필요합니다.

구문



설명

ACCESSCTRL

액세스 제어 권한을 부여합니다. ACCESSCTRL 권한은 보유자에게 다음과 같은 권한을 허용합니다.

- 다음 데이터베이스 권한을 부여하고 권한을 취소합니다. BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, EXPLAIN, IMPLICIT_SCHEMA, LOAD, QUIESE_CONNECT, SQLADM, WLMADM
- 모든 오브젝트 레벨 특권 부여 및 취소

ACCESSCTRL 권한은 PUBLIC에 부여될 수 없습니다(SQLSTATE 42508).

BINDADD

패키지를 만들 권한을 부여합니다. 패키지 작성자는 자동으로 해당 패키지에 대한 CONTROL 특권을 가지며 나중에 BINDADD 권한이 취소되어도 이 특권을 유지합니다.

CONNECT

데이터베이스에 액세스할 권한을 부여합니다.

CREATETAB

기본 테이블을 작성할 권한을 부여합니다. 기본 테이블 작성자는 자동으로 그 테이블에 대한 CONTROL 특권을 가지게 됩니다. 작성자는 후에 CREATETAB 권한이 취소되어도 이 특권을 유지합니다.

뷰 작성에 필요한 특별한 권한은 없습니다. 뷰 작성에 사용되는 명령문의 권한 부여 ID가 그 뷰의 각 기본 테이블에 CONTROL 또는 SELECT 특권이 있는 경우 언제라도 뷰를 작성할 수 있습니다.

CREATE_EXTERNAL_ROUTINE

외부 루틴을 등록하기 위한 권한을 부여합니다. 이렇게 등록된 루틴은 악영향이 발생하지 않도록 주의해야 합니다. 자세한 내용은 CREATE 또는 ALTER 루틴 명령문의 THREADSAFE절을 참조하십시오.

외부 루틴을 등록하면 나중에 CREATE_EXTERNAL_ROUTINE을 취소하더라도 이 루틴은 남아 있습니다.

CREATE_NOT_FENCED_ROUTINE

데이터베이스 관리 프로그램 프로세스에서 실행되는 레지스터 루틴에 권한을 부여합니다. 이렇게 등록된 루틴은 악영향이 발생하지 않도록 주의해야 합니다. 자세한 내용은 CREATE 또는 ALTER 루틴 명령문의 FENCEDE절을 참조하십시오.

일단 루틴이 안정되지 않은 것으로 등록되면 나중에 CREATE_NOT_FENCED_ROUTINE이 권한 취소되어도 이러한 방식으로 계속 실행됩니다.

CREATE_EXTERNAL_ROUTINE은 CREATE_NOT_FENCED_ROUTINE 권한이 부여된 *authorization-name*에 자동으로 권한 부여됩니다.

DATAACCESS
데이터에 액세스할 권한을 부여합니다. DATAACCESS 권한은 보유자에게 다음과 같은 권한을 허용합니다.

- 데이터 선택, 삽입, 갱신, 삭제 및 로드
- 패키지 실행
- 루틴 실행(감사 루틴 제외)

DATAACCESS 권한은 PUBLIC에 부여될 수 없습니다(SQLSTATE 42508).

DBADM
데이터베이스 관리자 권한을 부여합니다. 데이터베이스 관리자는 데이터베이스의 거의 모든 오브젝트에 대해 거의 모든 특권을 갖고 있습니다. 유일한 예외는 액세스 제어, 데이터 액세스 및 보안 관리자 권한의 일부 특권입니다.

EXPLAIN
명령문을 설명할 권한을 부여합니다. EXPLAIN 권한을 사용하여 보유자는 데이터에 액세스하지 않아도 동적 및 정적 SQL문을 Explain, 준비 및 설명할 수 있습니다.

IMPLICIT_SCHEMA
스키마를 내재적으로 작성하기 위한 권한을 부여합니다.

LOAD
이 데이터베이스에서 로드할 권한을 부여합니다. 이 권한은 사용자에게 이 데이터베이스의 로드 유틸리티를 사용할 수 있는 권한을 제공합니다. DATAACCESS 및 DBADM도 디폴트로 이러한 권한을 가집니다. 그러나 사용자가 LOAD 권한(DATAACCESS이 아닌)만을 가지는 경우 사용자는 테이블 레벨 특권을 가져야 합니다. LOAD 특권에 추가로 사용자는 다음을 가져야 합니다.

- INSERT, TERMINATE(이전의 LOAD INSERT를 종료) 또는 RESTART(이전 LOAD INSERT를 재시작) 모드로 LOAD 테이블에 대한 INSERT 특권
- REPLACE, TERMINATE(이전의 LOAD REPLACE를 종료) 또는 RESTART(이전 LOAD INSERT를 재시작) 모드로 LOAD 테이블에 대한 INSERT 및 DELETE 특권
- 그러한 테이블이 LOAD 일부로 사용되는 경우 예외 테이블에 대한 INSERT 특권

QUIESCE_CONNECT
Quiesce 상태에서 데이터베이스에 액세스할 권한을 부여합니다.

SECADM

보안 관리자 권한을 부여합니다. SECADM 권한 소유자에게는 다음과 같은 권한이 주어집니다.

- 감사 규정, 역할, 보안 레이블, 보안 레이블 구성요소, 보안 규정 및 트러스트된 컨텍스트와 같은 보안 오브젝트 작성 및 삭제
- 권한 부여와 취소, 면제, 특권, 역할 및 보안 레이블
- SETSESSIONUSER 특권 부여 및 취소
- 다른 사용자가 소유한 오브젝트의 TRANSFER OWNERSHIP 실행

SECADM 권한은 PUBLIC에 부여될 수 없습니다(SQLSTATE 42508).

SQLADM

SQL문 실행을 관리할 권한을 부여합니다. SQLADM 권한은 보유자에게 다음과 같은 권한을 허용합니다.

- 이벤트 모니터 작성, 삭제, 플러시 및 설정
- 데이터에 액세스하지 않아도 동적 및 정적 SQL문을 Explain, 준비 및 설명할 수 있습니다.
- 프로파일 캐시 플러시 최적화
- 패키지 캐시 플러시
- runstats 유틸리티 실행

WLMADM

워크로드를 관리할 권한을 부여합니다. WLMADM 권한은 보유자에게 다음과 같은 권한을 허용합니다.

- 서비스 클래스, 작업 조치 세트, 작업 클래스 세트 또는 워크로드를 작성, 삭제 및 변경합니다.

TO

권한이 부여되는 대상을 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정하십시오. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC
사용자 세트에 권한을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오. DBADM은 PUBLIC으로 권한 부여될 수 없습니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.

주

- DBADM 권한은 특수 그룹 PUBLIC으로 부여될 수 없습니다. 그러므로 *role-name*이 PUBLIC에 직접 또는 간접적으로 부여된 경우 DBADM 권한을 역할 *role-name*에 부여할 수 없습니다(SQLSTATE 42508).
  - 다음 명령문이 발행되면 역할 *role-name*이 직접적으로 PUBLIC에 부여됩니다.

    ```
    GRANT ROLE role-name TO PUBLIC
    ```

  - 다음 명령문이 발행되면 역할 *role-name*이 간접적으로 PUBLIC에 부여됩니다.

    ```
    GRANT ROLE role-name TO ROLE role-name2
    GRANT ROLE role-name2 TO PUBLIC
    ```

- *호환성*:
  - 이전 버전 DB2와의 호환성을 위해,
    - CREATE_NOT_FENCED_ROUTINE 대신 CREATE_NOT_FENCED를 지정할 수 있습니다.
  - z/OS용 DB2와의 호환성을 위해,

- DATABASE 대신 SYSTEM을 지정할 수 있습니다.

예:

예 1: 사용자 WINKEN, BLINKEN 및 NOD에게 데이터베이스에 연결하기 위한 권한을 제공하십시오.

```
GRANT CONNECT ON DATABASE TO USER WINKEN, USER BLINKEN, USER NOD
```

예 2: 데이터베이스에 대한 BINDADD 권한을 그룹 D024에 부여합니다. 시스템에 D024란 사용자와 그룹이 모두 있습니다.

```
GRANT BINDADD ON DATABASE TO GROUP D024
```

GROUP 키워드가 지정되어야 함에 유의하십시오. 그렇지 않은 경우 D024라는 사용자와 그룹이 모두 존재하므로 오류가 발생합니다. D024 그룹 구성원은 데이터베이스의 패키지를 바인드할 수 있도록 허용되어 있으나 이 사용자 또한 D024 그룹의 구성원이고 이전에 BINDADD 권한을 부여받았거나, D024가 구성원이었던 다른 그룹에 BINDADD 권한이 부여된 경우 D024 사용자는 허용되지 않습니다.

예 3: 사용자 Walid에게 보안 관리자 권한을 부여하십시오.

```
GRANT SECADM ON DATABASE TO USER Walid
```

GRANT(면제)

이러한 양식의 GRANT문은 지정된 레이블 기반 액세스 제어(LBAC) 보안 규정의 액세스 규칙에 대한 면제를 사용자, 그룹 또는 역할에 부여합니다. 면제를 보유 중인 사용자가 해당 보안 규정으로 보호된 테이블의 데이터를 액세스할 경우 데이터 액세스 가능 여부를 판별할 때 표시된 규칙이 강제실행되지 않습니다.

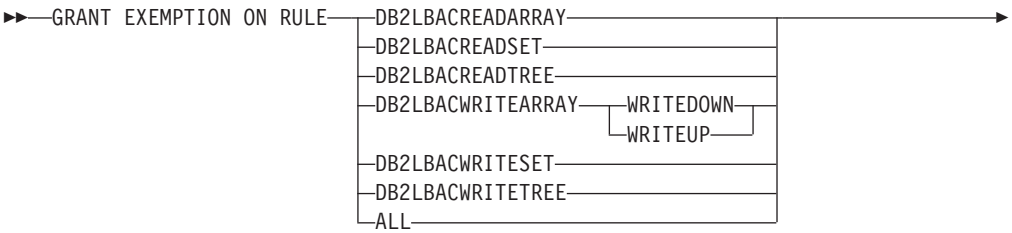
호출

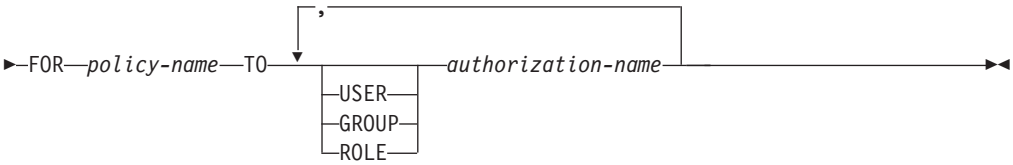
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문





설명

EXEMPTION ON RULE

액세스 규칙에 대한 면제를 부여합니다.

DB2LBACREADARRAY

사전 정의된 DB2LBACREADARRAY 규칙에 대한 면제를 부여합니다.

DB2LBACREADSET

사전 정의된 DB2LBACREADSET 규칙에 대한 면제를 부여합니다.

DB2LBACREADTREE

사전 정의된 DB2LBACREADTREE 규칙에 대한 면제를 부여합니다.

DB2LBACWRITEARRAY
사전 정의된 DB2LBACWRITEARRAY 규칙에 대한 면제를 부여합니다.

WRITEDOWN
면제가 아래로 쓰기에만 적용되도록 지정합니다.

WRITEUP
면제가 위로 쓰기에만 적용되도록 지정합니다.

DB2LBACWRITESET
사전 정의된 DB2LBACWRITESET 규칙에 대한 면제를 부여합니다.

DB2LBACWRITETREE
사전 정의된 DB2LBACWRITETREE 규칙에 대한 면제를 부여합니다.

ALL
사전 정의된 모든 규칙에 대한 면제를 부여합니다.

FOR *policy-name*
면제가 허용되는 보안 규정을 식별합니다. 면제는 해당 보안 규정으로 보호된 테이블에 대해서만 유효합니다. 이 이름은 카탈로그에 이미 기술된 유형을 식별해야 합니다(SQLSTATE 42704).

TO
면제를 부여할 사용자를 지정합니다.

USER
*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP
*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE
*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름이 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...
하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

규칙

- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 인스턴스에 적용되는 보안 플러그인이 *authorization-name* 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에 ROLE로 정의되고 적용되는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).

  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER와 GROUP 둘 다로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER로만 정의되거나 정의되지 않으면, USER로 가정됩니다.
  - *authorization-name*이 적용되는 보안 플러그인에 따라 GROUP으로만 정의되면, GROUP으로 가정됩니다.
  - *authorization-name*이 데이터베이스에 ROLE로만 정의되면, ROLE로 가정됩니다.
- 보안 규정이 그룹이나 역할을 통한 액세스를 고려하도록 정의되지 않은 경우, 액세스를 시도할 때 그룹이나 역할에 부여된 면제가 무시됩니다.

주

- 기본적으로 보안 규정이 작성될 때는 개별 사용자에게 부여된 면제만 고려됩니다. 보안 규정에 그룹이나 역할을 고려하려면, ALTER SECURITY POLICY문을 실행하고 USE GROUP AUTHORIZATION 또는 USE ROLE AUTHORIZATION을 지정해야 합니다.

예:

예 1: 사용자 WALID에게 보안 규정 DATA_ACCESS에 대한 액세스 규칙 DB2LBACREADSET의 면제를 허용합니다.

```
GRANT EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS TO USER WALID
```

예 2: 사용자 BOBBY에게 보안 규정 DATA_ACCESS에 대한 WRITEDOWN 옵션을 사용하여 액세스 규칙 DB2LBACWRITEARRAY에 대한 면제를 부여합니다.

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
  FOR DATA_ACCESS TO USER BOBBY
```

예 3: 사용자 BOBBY에게 보안 규정 DATA_ACCESS에 대한 WRITEUP 옵션을 사용하여 액세스 규칙 DB2LBACWRITEARRAY에 대한 면제를 부여합니다.

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
  FOR DATA_ACCESS TO USER BOBBY
```

GRANT(전역 변수 특권)

이 양식의 GRANT문은 작성된 전역 변수에 대해 하나 이상의 특권을 부여합니다.

호출

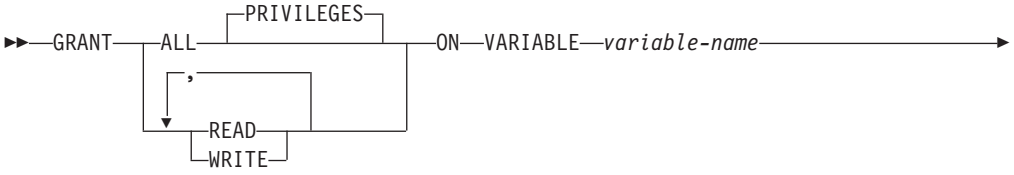
이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

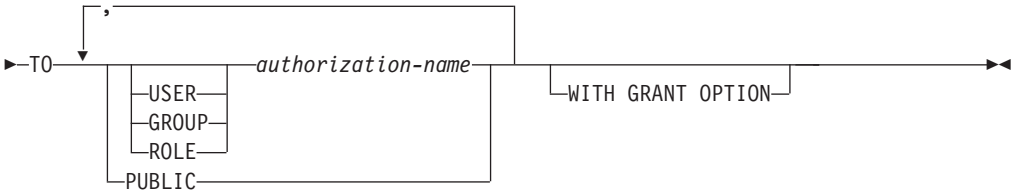
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 전역 변수에 대해 식별된 각 특권에 대한 WITH GRANT OPTION
- ACCESSCTRL 또는 SECADM 권한

구문





설명

ALL PRIVILEGES

지정된 전역 변수에 대해 모든 특권을 부여합니다.

READ

지정된 전역 변수의 값을 읽을 수 있는 특권을 부여합니다.

WRITE

지정된 전역 변수에 값을 지정할 수 있는 특권을 부여합니다.

ON VARIABLE *variable-name*

하나 이상의 특권이 부여될 전역 변수를 식별합니다. *variable-name*(내재된 또는 명시적 규정자 포함)은 현재 서버에 있으며 모듈 변수가 아닌(SQLSTATE 42704) 전역 변수를 식별해야 합니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

ROLE

*authorization-name*이 현재 서버에서 기존 역할을 식별함을 지정합니다 (SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다. 권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다 (SQLSTATE 42502).

PUBLIC

사용자(권한 부여 ID) 세트에 지정된 특권을 부여합니다. 자세한 정보는 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 부여할 수 있도록 합니다. WITH GRANT OPTION 절이 생략되면 지정된 *authorization-name*은 다른 소스로부터 해당 권한이 수신되지 않는 한 다른 사람에게 특권을 부여할 수 없습니다.

규칙

- 지정된 *authorization-name*마다, USER, GROUP 또는 ROLE 키워드 중 어느 것도 지정하지 않은 경우
  - 인스턴스에 대해 적용 중인 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없는 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 운영 체제에서 GROUP 또는 USER로 정의된 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 USER 및 GROUP 둘 다로 정의된 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 USER로만 정의된 경우 또는 정의되지 않은 경우 USER가 가정됩니다.
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 GROUP으로만 정의된 경우 GROUP이 가정됩니다.
  - *authorization-name*이 데이터베이스에서 ROLE로만 정의된 경우 ROLE이 가정됩니다.

예 :

사용자 ZUBIRI에서 전역 변수 MYSCHEMA.MYJOB_PRINTER에 대한 READ 및 WRITE 특권을 부여하십시오.

```
GRANT READ, WRITE ON VARIABLE MYSCHEMA.MYJOB_PRINTER TO ZUBIRI
```

GRANT(인덱스 권한)

이러한 형태의 GRANT문은 인덱스에 대한 CONTROL 특권을 부여합니다.

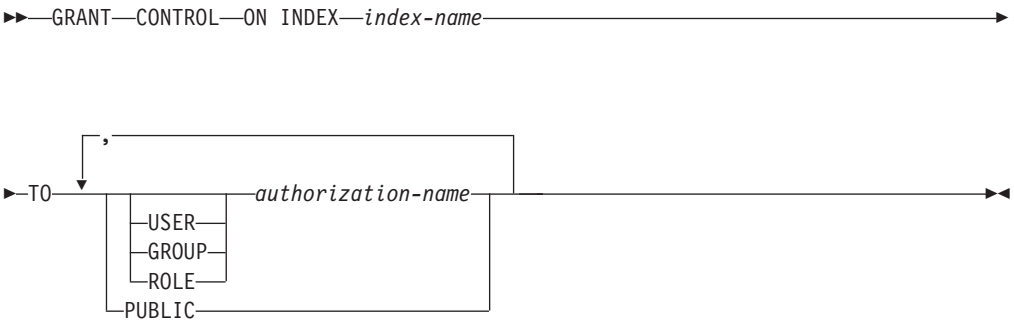
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문



설명

CONTROL

인덱스를 삭제(drop)할 특권을 부여합니다. 이는 인덱스에 대한 CONTROL 권한으로서, 인덱스 작성자에게 자동으로 권한 부여됩니다.

ON INDEX *index-name*

CONTROL 특권이 권한 부여될 인덱스를 식별합니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정하십시오. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

사용자 세트에 특권을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.

예 :

```
GRANT CONTROL ON INDEX DEPTIDX TO USER KIESLER
```

GRANT(모듈 특권)

이러한 형태의 GRANT문은 모듈에 대한 특권을 부여합니다.

호출

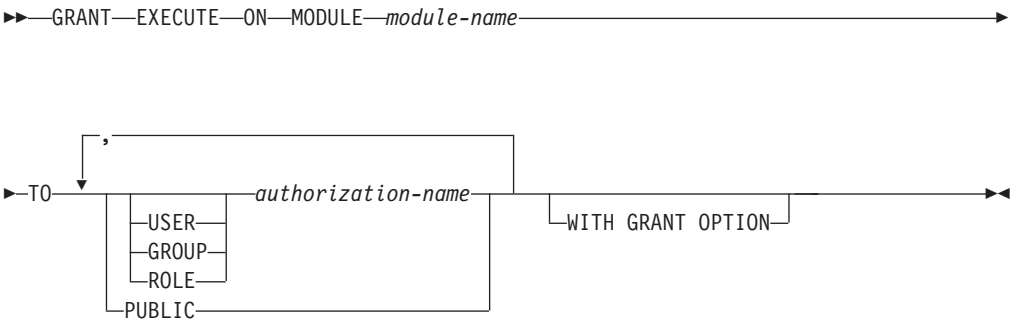
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 모듈에서 EXECUTE에 대한 WITH GRANT OPTION.
- ACCESSCTRL 또는 SECADM 권한.

구문



설명

EXECUTE

발행된 모듈 오브젝트를 참조하는 특권을 부여합니다. 여기에는 다음 특권을 포함합니다.

- 모듈에 정의된 발행 루틴을 실행하십시오.
- 모듈에서 정의된 발행 전역 변수를 읽고 여기에 작성하십시오.
- 모듈에 정의된 발행 사용자 정의 유형을 참조하십시오.
- 모듈에 정의된 발행 조건을 참조하십시오.

ON MODULE *module-name*

특권이 부여되는 모듈을 식별합니다. module-name은 현재 서버에 있는 모듈을 식별해야 합니다(SQLSTATE 42704).

TO

특권을 부여한 사용자로 표시합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정하십시오. 역할 이름이 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

하나 이상의 권한 부여 ID를 나열합니다.

PUBLIC

사용자 세트에 특권을 부여합니다(권한 부여 ID). 자세한 정보는 "권한 부여, 특권 및 오브젝트 소유권"을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사용자에게 EXECUTE 특권을 부여할 수 있도록 합니다. WITH GRANT OPTION이 생략되면, 지정된 *authorization-name*은 다른 소스로부터 해당 권한이 수신되지 않는 한 다른 사람에게 EXECUTE 특권을 부여할 수 없습니다.

예 :

다음은 MYMODA 모듈에서 사용자 JONES에게 EXECUTE 특권을 부여하는 방법의 예입니다.

```
GRANT EXECUTE
  ON MODULE MYMODA
    TO JONES
```

GRANT(패키지 특권)

이러한 형태의 GRANT문은 패키지에 대한 특권을 부여합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

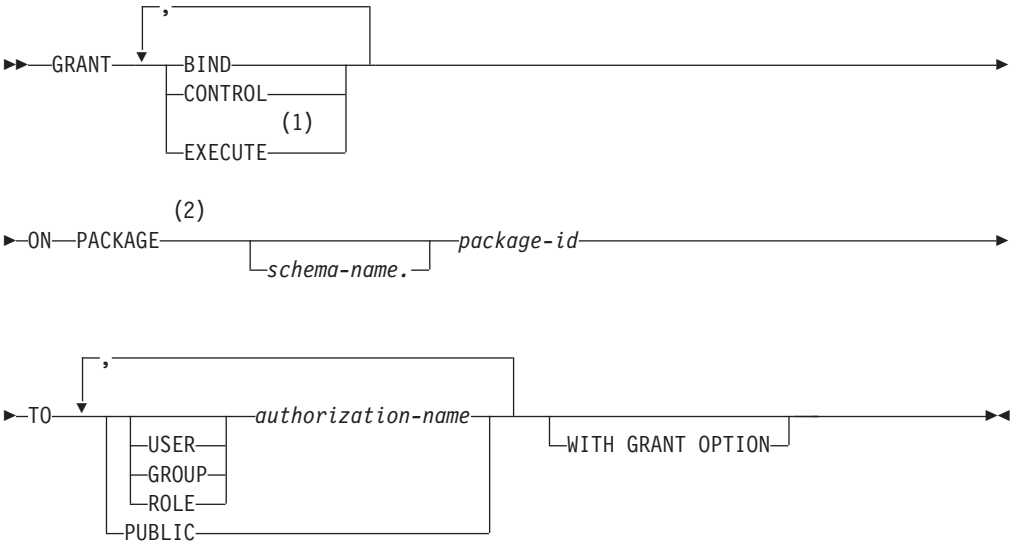
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 참조된 패키지에 대한 CONTROL 특권
- *package-name*에서 식별된 각 특권에 대한 WITH GRANT OPTION
- ACCESSCTRL 또는 SECADM 권한

ACCESSCTRL 또는 SECADM 권한은 CONTROL 특권을 부여하는 데 필요합니다.

구문



주:

1 RUN은 EXECUTE에 대한 동의어로 사용할 수 있습니다.

2 PACKAGE에 대한 동의어로 PROGRAM을 사용할 수 있습니다.

설명

BIND

패키지를 바인드할 특권을 부여합니다. BIND 특권을 사용하여 사용자가 해당 패

키지에 대하여 BIND 명령을 다시 발행하거나 REBIND 명령을 발행할 수 있습니다. 뿐만 아니라 사용자는 기존 패키지의 새 버전을 작성할 수도 있습니다.

사용자는 BIND 특권 이외에도 프로그램 내의 정적 DML문에서 참조하는 각 테이블에 대해 필요한 특권을 소유해야 합니다. 정적 DML문에서 권한 부여가 바인드될 때 점검되므로 이는 반드시 필요합니다.

CONTROL

패키지 리바인드, 삭제 또는 실행 특권을 부여하고 다른 사용자에게 패키지 특권을 확장시킵니다. 이는 인덱스에 대한 CONTROL 특권으로서 인덱스 작성자에게 자동으로 권한 부여됩니다. 패키지 소유자는 패키지 바인더 또는 바인드/프리컴파일 시 OWNER 옵션으로 지정된 ID입니다.

BIND 및 EXECUTE는 CONTROL 특권이 부여된 *authorization-name*에게 자동으로 권한 부여됩니다.

CONTROL은 위의 특권(CONTROL은 제외)을 다른 사람에게 권한 부여할 수 있는 능력을 권한 부여합니다.

EXECUTE

패키지를 실행할 특권을 부여합니다.

ON PACKAGE *schema-name.package-id*

특권이 부여될 패키지 이름을 지정합니다. 스키마 이름이 지정되지 않은 경우 디폴트 스키마가 패키지 ID를 내재적으로 규정합니다. 패키지 특권을 부여하면 패키지의 모든 버전 즉, 동일한 패키지 ID와 패키지 스키마를 공유하는 모든 패키지에 적용됩니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

사용자 세트에 특권을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 부여할 수 있도록 합니다.

지정된 특권에 CONTROL이 포함되는 경우 WITH GRANT OPTION은 CONTROL을 제외한 적용 가능한 모든 특권에 적용됩니다(SQLSTATE 01516).

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.

주

- 패키지 특권은 패키지의 모든 버전 즉, 동일한 패키지 ID와 패키지 스키마를 공유하는 모든 패키지에 적용됩니다. 한 버전만 액세스할 수 있도록 제한할 수는 없습니다. CONTROL 특권은 패키지를 바인드한 사용자에게 내재적으로 부여되므로 서로 다른 두 사용자가 패키지의 두 버전을 바인드하면 두 사용자 모두에게 상대방의 패키지에 대한 액세스 권한이 내재적으로 부여됩니다.

예:

예 1: 패키지 CORPDATA.PKGA에 대해 EXECUTE 특권을 PUBLIC에 부여하십시오.

```
GRANT EXECUTE
  ON PACKAGE CORPDATA.PKGA
  TO PUBLIC
```

예 2: 패키지 CORPDATA.PKGA에 대한 GRANT EXECUTE 특권을 EMPLOYEE라는 사용자에게 부여하십시오. EMPLOYEE 사용자 또는 그룹은 없습니다.

```
GRANT EXECUTE ON PACKAGE
       CORPDATA.PKGA TO EMPLOYEE
```

또는

```
GRANT EXECUTE ON PACKAGE
       CORPDATA.PKGA TO USER EMPLOYEE
```

GRANT(역할)

이 양식의 GRANT문은 사용자, 그룹 또는 다른 역할에 역할을 부여합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

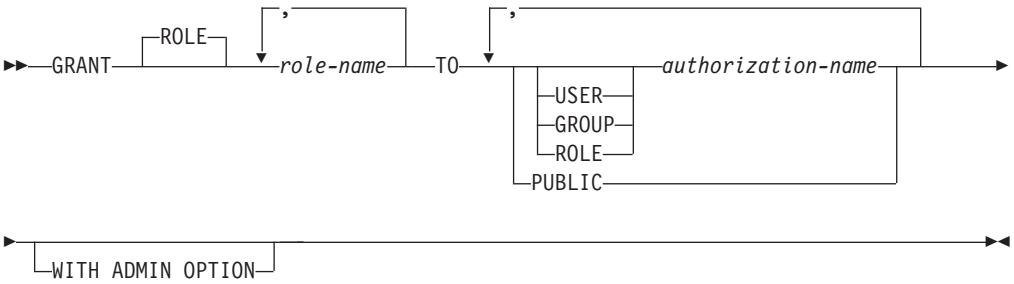
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 역할에 대한 WITH ADMIN OPTION
- SECADM 권한

SECADM 권한은 WITH ADMIN OPTION을 *authorization-name*에 부여하는 데 필요합니다.

구문



설명

ROLE *role-name*,...

권한을 부여할 하나 이상의 역할을 식별합니다. 각 *role-name*은 현재 서버에서 기존 역할을 식별해야 합니다(SQLSTATE 42704).

TO

역할이 부여될 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

ROLE

*authorization-name*이 현재 서버에서 기존 역할을 식별함을 지정합니다 (SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다. 권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다 (SQLSTATE 42502).

PUBLIC

사용자(권한 부여 ID) 세트에 지정된 역할을 부여합니다. 자세한 정보는 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH ADMIN OPTION

지정된 *authorization-name*이 다른 권한 부여 이름에 대해 *role-name*을 부여 또는 취소하거나, 주석을 역할과 연관시킬 수 있습니다. 지정된 *authorization-name*이 역할을 삭제할 수는 없습니다.

규칙

- 지정된 *authorization-name*마다, USER, GROUP 또는 ROLE 키워드 중 어느 것도 지정하지 않은 경우:
  - 인스턴스에 대해 적용 중인 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없는 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 운영 체제에서 GROUP 또는 USER로 정의된 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 USER 및 GROUP 둘 다로 정의된 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 USER로만 정의된 경우 또는 정의되지 않은 경우 USER가 가정됩니다.
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 GROUP으로만 정의된 경우 GROUP이 가정됩니다.
  - *authorization-name*이 데이터베이스에서 ROLE로만 정의된 경우 ROLE이 가정됩니다.
- 역할의 계층 구조는 다른 역할에 하나의 역할을 부여하여 빌드할 수 있습니다. 그러나 순환은 허용되지 않습니다(SQLSTATE 428GF). 예를 들어, 역할 R1이 다른 역할 R2에 부여되면, 역할 R2(또는 R2를 포함하는 일부 다른 역할 R*n*)는 다시 R1에 부여될 수 없습니다. 이와 같이 하면 순환이 생성되기 때문입니다.

주

- 역할 R1이 다른 역할 R2에 부여되면 R2는 R1을 포함합니다.
- DBADM 권한은 PUBLIC에 부여될 수 없습니다. 따라서 다음과 같습니다.

– 역할 R1이 직접 또는 간접으로 DBADM 권한을 보유하는 경우 역할 R1을 PUBLIC에 부여할 수 없습니다(SQLSTATE 42508).
  - 다음 명령문이 실행된 경우 역할 R1이 DBADM 권한을 직접 보유합니다.

    ```
    GRANT DBADM ON DATABASE TO ROLE R1
    ```

  - 다음 명령문이 실행된 경우 역할 R1은 간접적으로 DBADM 권한을 보유합니다.

    ```
    GRANT DBADM ON DATABASE TO ROLE R2

    GRANT ROLE R2 TO ROLE R1
    ```

– 역할 R2가 직접 또는 간접으로 PUBLIC에 부여된 경우 DBADM 권한을 보유하는 역할 R1을 역할 R2에 부여할 수 없습니다(SQLSTATE 42508).
  - 다음 명령문이 실행된 경우 역할 R2가 직접 PUBLIC에 부여됩니다.

    ```
    GRANT ROLE R2 TO PUBLIC
    ```

  - 다음 명령문이 실행된 경우 역할 R2가 간접적으로 PUBLIC에 부여됩니다.

    ```
    GRANT ROLE R2 TO ROLE R3

    GRANT ROLE R3 TO PUBLIC
    ```

예:

예 1: 역할 INTERN을 역할 DOCTOR에 부여하고 역할 DOCTOR를 역할 SPECIALIST에 부여하십시오.

```
GRANT ROLE INTERN TO ROLE DOCTOR

GRANT ROLE DOCTOR TO ROLE SPECIALIST
```

예 2: 역할 INTERN을 PUBLIC에 부여하십시오.

```
GRANT ROLE INTERN TO PUBLIC
```

예 3: 역할 SPECIALIST를 사용자 BOB 및 그룹 TORONTO에 부여하십시오.

```
GRANT ROLE SPECIALIST TO USER BOB, GROUP TORONTO
```

GRANT(루틴 특권)

이러한 양식의 GRANT문은 모듈에 정의되지 않은 루틴(함수, 메소드 또는 프로시저)에 대한 특권을 부여합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

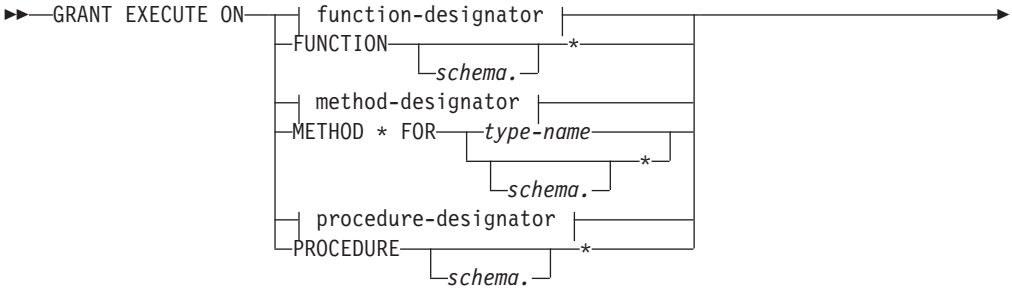
- 루틴에서 EXECUTE에 대한 WITH GRANT OPTION
- ACCESSCTRL 또는 SECADM 권한

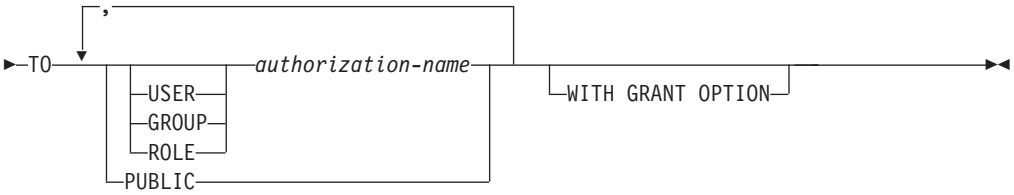
스키마나 유형에 있는 모든 루틴 EXECUTE 특권을 부여하려면 명령문의 권한 부여 ID가 갖는 특권에 다음 중 적어도 하나가 포함되어야 합니다.

- 지정한 스키마에 있는 지정한 유형의 모든 기존 루틴과 추후 루틴에 대한 EXECUTE에 대한 WITH GRANT OPTION
- ACCESSCTRL 또는 SECADM 권한

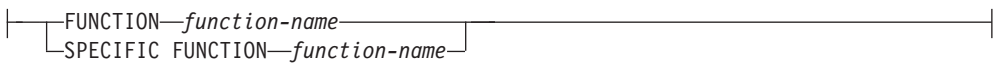
감사 프로시저 및 테이블 함수에 대한 EXECUTE 특권을 부여하려면 SECADM 권한이 필요합니다. 권한 부여 옵션 사용으로 EXECUTE 특권이 다음 루틴에 권한 부여될 수 없습니다(SQLSTATE 42501).

구문

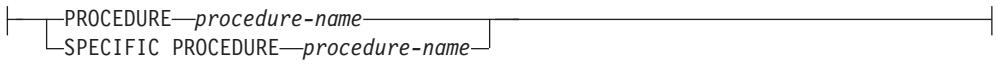




function-designator:



procedure-designator:



설명

EXECUTE

식별된 사용자 정의 함수(UDF), 메소드 또는 프로시저를 실행할 수 있는 특권을 부여하십시오.

function-designator

특권이 부여되는 함수에서 고유하게 식별합니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

FUNCTION *schema*.*

추후에 작성될 함수를 포함하여 스키마의 모든 함수를 식별합니다. 동적 SQL문에서 스키마를 지정하지 않으면 CURRENT SCHEMA 특수 레지스터의 스키마가 사용됩니다. 정적 SQL문에서 스키마를 지정하지 않으면 QUALIFIER 프리컴파일/바인드 옵션의 스키마가 사용됩니다.

method-designator

특권이 부여된 메소드를 고유하게 식별합니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

METHOD *

추후에 작성될 메소드를 포함하여 *type-name* 유형에 대한 모든 메소드를 식별합니다.

FOR *type-name*

지정한 메소드가 있는 유형의 이름을 지정합니다. 이 이름은 카탈로그에 기술되어 있는 유형을 식별해야 합니다(SQLSTATE 42704). 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터의 값은 규정되지 않은 유형 이름을 위한 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵

션은 내재적으로 규정되지 않은 유형 이름의 규정자를 지정합니다.*type-name* 대신 별표(*)를 사용하여 추후에 작성될 유형을 포함하여 스키마의 모든 유형을 식별할 수 있습니다.

procedure-designator

특권이 부여된 프로시저를 고유하게 식별합니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

PROCEDURE *schema*.*

추후에 작성될 프로시저를 포함하여 스키마의 모든 프로시저를 식별합니다. 동적 SQL문에서 스키마를 지정하지 않으면 CURRENT SCHEMA 특수 레지스터의 스키마가 사용됩니다. 정적 SQL문에서 스키마를 지정하지 않으면 QUALIFIER 프리컴파일/바인드 옵션의 스키마가 사용됩니다.

TO

EXECUTE 특권이 부여되는 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름이 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

PUBLIC

일련의 사용자에게 EXECUTE 특권을 부여합니다(권한 부여 ID). 자세한 정보는 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 EXECUTE 특권을 GRANT할 수 있도록 합니다.

WITH GRANT OPTION이 생략되면 지정된 *authorization-name*은 다음의 경우에만 EXECUTE 특권을 부여할 수 있습니다.

- SYSADM 또는 DBADM 권한이 있는 경우
- 다른 소스의 EXECUTE 특권을 부여하기 위한 능력을 제공받은 경우

규칙

- 'SYSIBM' 스키마나 'SYSFUN' 스키마로 정의된 함수나 메소드에 대해 EXECUTE 특권은 부여할 수 없습니다(SQLSTATE 42832).

- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 인스턴스에 적용되는 보안 플러그인이 *authorization-name* 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에 ROLE로 정의되고 적용되는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER와 GROUP 둘 다로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER로만 정의되거나 정의되지 않으면, USER로 가정됩니다.
  - *authorization-name*이 적용되는 보안 플러그인에 따라 GROUP으로만 정의되면, GROUP으로 가정됩니다.
  - *authorization-name*이 데이터베이스에 ROLE로만 정의되면, ROLE로 가정됩니다.
- 일반적으로 GRANT문은 명령문의 권한 부여 ID가 권한을 부여하도록 허용한 특권 부여를 수행하며 두 개 이상의 특권이 부여되지 않은 경우 경고가 리턴됩니다(SQLSTATE 01007). 명령문 처리에 사용된 패키지가 SQL92E 또는 MIA로 설정된 LANGLEVEL로 프리컴파일되었으며 특권이 부여되지 않은 경우 경고가 리턴됩니다(SQLSTATE 01007). 권한을 준 사용자에게 권한 부여 조작의 오브젝트에 대한 특권이 없으면, 오류가 표시됩니다(SQLSTATE 42501).

주

- 모듈에 정의된 루틴의 특권은 GRANT(모듈 특권)문을 사용하여 모듈 레벨에 부여됩니다. 모듈의 EXECUTE 특권으로 모듈의 모든 오브젝트에 액세스할 수 있습니다.

예:

예 1: 사용자 JONES에게 CALC_SALARY 함수에 대한 EXECUTE 특권을 부여하십시오. 스키마에 함수 이름이 CALC_SALARY인 함수가 한 개만 있다고 가정합니다.

```
GRANT EXECUTE ON FUNCTION CALC_SALARY TO JONES
```

예 2: 현재 서버에 있는 모든 사용자에게 VACATION_ACCR 프로시저에 대한 EXECUTE 특권을 부여하십시오.

```
GRANT EXECUTE ON PROCEDURE VACATION_ACCR TO PUBLIC
```

예 3: 관리 보조자에게 DEPT_TOTALS 함수에 대한 EXECUTE 특권을 부여하여 다른 사용자에게 이 함수에 대한 EXECUTE 특권을 부여할 수 있도록 하십시오. 함수는 특정 이름 DEPT85_TOT를 갖습니다. 스키마에 이름이 DEPT_TOTALS인 함수가 여러 개 있다고 가정합니다.

```
GRANT EXECUTE ON SPECIFIC FUNCTION DEPT85_TOT
  TO ADMIN_A WITH GRANT OPTION
```

예 4: HR(인사과)에 NEW_DEPT_HIRES 함수에 대한 EXECUTE 특권을 부여하십시오. 이 함수에는 유형이 각각 INTEGER와 CHAR(10)인 두 개의 입력 매개변수가 있습니다. 스키마에 이름이 NEW_DEPT_HIRES인 함수가 여러 개 있다고 가정합니다.

```
GRANT EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10)) TO HR
```

예 5: 사용자 JONES에게 EMPLOYEE 유형의 SET_SALARY 메소드에 대한 EXECUTE 특권을 부여하십시오.

```
GRANT EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE TO JONES
```

GRANT(스키마 특권)

이러한 형태의 GRANT문은 스키마에 대한 특권을 부여합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

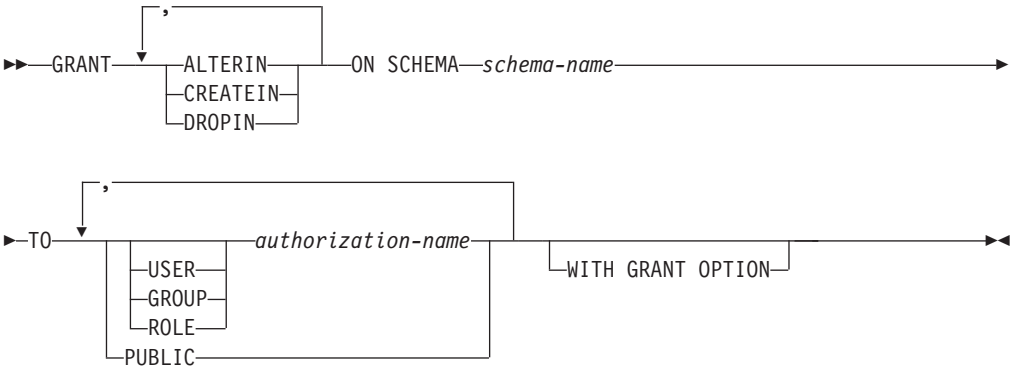
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- *sequence-name*의 식별된 각 특권에 대한 WITH GRANT OPTION
- ACCESSCTRL 또는 SECADM 권한

사용자는 SYSIBM, SYSIBMADM, SYSCAT, SYSFUN 또는 SYSSTAT에 대한 특권을 부여할 수 없습니다(SQLSTATE 42501).

구문



설명

ALTERIN

스키마에 있는 모든 오브젝트에 주석을 달거나 변경할 수 있는 특권을 부여합니다. 명시적으로 작성된 스키마의 소유자는 자동으로 ALTERIN 특권을 부여받습니다.

CREATEIN

스키마에 오브젝트를 작성할 특권을 부여합니다. 그래도 오브젝트를 작성하는 데 필요한 기타 권한이나 특권(예: CREATETAB)이 있어야 합니다. 명시적으로 작성된 스키마의 소유자는 자동으로 CREATEIN 특권을 부여받습니다. 내재적으로 작성된 스키마는 CREATEIN 특권을 자동으로 PUBLIC에 부여합니다.

DROPIN

스키마의 모든 오브젝트를 삭제하기 위한 특권을 부여합니다. 명시적으로 작성된 스키마의 소유자는 자동으로 DROPIN 특권을 부여받습니다.

ON SCHEMA *schema-name*

특권이 부여될 스키마를 식별합니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

사용자 세트에 특권을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 부여할 수 있도록 합니다.

WITH GRANT OPTION이 생략되면 지정된 *authorization-name*은 다음의 경우에만 다른 사람에게 특권을 부여할 수 있습니다.

- DBADM 권한을 갖는 경우
- 다른 소스의 특권을 부여할 수 있는 능력을 제공받은 경우

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).

  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.
- 일반적으로 GRANT문은 명령문의 권한 부여 ID가 권한을 부여하도록 허용한 특권 부여를 수행하며 두 개 이상의 특권이 부여되지 않은 경우 경고가 리턴됩니다(SQLSTATE 01007). 부여된 특권이 없을 경우 오류가 발생합니다(SQLSTATE 42501). 명령문 처리에 사용된 패키지가 MIA에 대해 SQL92E로 설정된 LANGLEVEL로 프리컴파일되었으면 권한 부여자에게 해당 권한 조작의 오브젝트에 대한 특권이 없는 한 경고가 리턴됩니다.

주

- **SYSPUBLIC에서 부여:** 특권을 예약된 스키마 SYSPUBLIC에서 부여할 수 있습니다. CREATEIN 특권을 부여하면 사용자가 공용 별명을 작성할 수 있으며 DROPIN 특권을 부여하면 사용자가 공용 별명을 삭제할 수 있습니다.

예:

예 1: 스키마 CORPDATA에 오브젝트를 작성하도록 사용자 JSINGLETON에게 권한 부여하십시오.

```
GRANT CREATEIN ON SCHEMA CORPDATA TO JSINGLETON
```

예 2: 스키마 CORPDATA에 오브젝트를 작성 및 삭제하도록 사용자 IHAKES에게 권한 부여하십시오.

```
GRANT CREATEIN, DROPIN ON SCHEMA CORPDATA TO IHAKES
```

GRANT(보안 레이블)

이러한 양식의 GRANT문은 읽기 액세스, 쓰기 액세스 또는 읽기 액세스와 쓰기 액세스 둘 다에 대한 레이블 기반 액세스 제어(LBAC) 보안 레이블을 사용자, 그룹 또는 역할에 부여합니다.

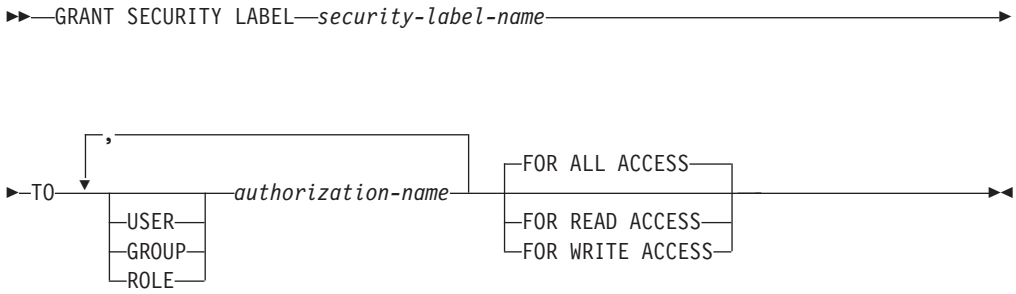
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



설명

SECURITY LABEL *security-label-name*

security-label-name 보안 레이블에 대한 권한을 부여합니다. 이름은 보안 규정을 사용하여 규정해야 하며(SQLSTATE 42704), 현재 서버에 존재하는 보안 레이블을 식별해야 합니다(SQLSTATE 42704).

TO

지정된 보안 레이블을 부여할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름이 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

FOR ALL ACCESS

읽기 액세스 및 쓰기 액세스 둘 다에 대한 권한을 보안 레이블에 부여하도록 표시합니다.

FOR READ ACCESS

읽기 액세스에 대한 권한만을 보안 레이블에 부여하도록 표시합니다.

FOR WRITE ACCESS

쓰기 액세스에 대한 권한만을 보안 레이블에 부여하도록 표시합니다.

규칙

- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 인스턴스에 적용되는 보안 플러그인이 *authorization-name* 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에 ROLE로 정의되고 적용되는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER와 GROUP 둘 다로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER로만 정의되거나 정의되지 않으면, USER로 가정됩니다.
  - *authorization-name*이 적용되는 보안 플러그인에 따라 GROUP으로만 정의되면, GROUP으로 가정됩니다.
  - *authorization-name*이 데이터베이스에 ROLE로만 정의되면, ROLE로 가정됩니다.
- 주어진 보안 규정에 대해 읽기 액세스에 대한 해당 규정의 최대 하나의 보안 레이블과 쓰기 액세스에 대한 하나의 보안 레이블을 *authorization-name*에 부여할 수 있습니다. 권한 받은 사용자가 이미 표시된 액세스 유형(읽기 또는 쓰기)에 대한 보안 레이블을 보유하고 *security-label-name*을 보유하는 보안 규정의 일부인 경우, 오류가 리턴됩니다(SQLSTATE 428GR).
- 보안 규정이 그룹이나 역할을 통한 액세스를 고려하도록 정의되지 않은 경우, 액세스를 시도할 때 그룹이나 역할에 부여된 보안 레이블이 무시됩니다.
- *authorization-name*이 읽기 액세스와 쓰기 액세스에 대해 서로 다른 보안 레이블을 보유하는 경우, 보안 레이블은 다음 기준을 충족시켜야 합니다(SQLSTATE 428GQ).
  - 보안 레이블의 구성요소가 ARRAY 유형일 경우 해당 구성요소의 값이 두 보안 레이블에서 동일해야 합니다.

- 보안 레이블의 구성요소가 SET 유형일 경우 쓰기 보안 레이블에서 해당 유형에 대한 값의 모든 요소가 또한 읽기 보안 레이블에서 해당 구성요소에 대한 값의 일부여야 합니다.
- 보안 레이블의 구성요소가 TREE 유형일 경우 쓰기 보안 레이블에서 해당 구성요소에 대한 값의 모든 요소가 읽기 보안 레이블에서 동일한 구성요소에 대한 값의 요소 중 하나와 같거나 또는 하위 요소여야 합니다.

주

- 기본적으로 보안 규정이 작성될 때는 개별 사용자에게 부여된 보안 레이블만 고려됩니다. 보안 규정에 그룹이나 역할을 고려하려면, ALTER SECURITY POLICY문을 실행하고 USE GROUP AUTHORIZATION 또는 USE ROLE AUTHORIZATION을 지정해야 합니다.

예:

예 1: 다음 명령문은 두 개의 보안 레이블을 GUYLAINE 사용자에게 부여합니다. EMPLOYEESECLABELREAD 보안 레이블이 읽기 액세스를 위해 부여되며 EMPLOYEESECLABELWRITE 보안 레이블이 쓰기 액세스를 위해 부여됩니다. 두 보안 레이블은 모두 DATA_ACCESS 보안 레이블의 일부입니다.

```
GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
  TO USER GUYLAINE FOR READ ACCESS

GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
  TO USER GUYLAINE FOR WRITE ACCESS
```

이제 동일한 사용자에게 읽기 및 쓰기 액세스 모두에 대한 BEGINNER 보안 레이블이 부여됩니다. BEGINNER가 CLASSPOLICY 보안 규정의 일부이고, 기존에 보유된 보안 레이블이 DATA_ACCESS 보안 규정의 일부이므로 오류가 발생하지 않습니다.

```
GRANT SECURITY LABEL CLASSPOLICY.BEGINNER
  TO USER GUYLAINE FOR ALL ACCESS
```

GRANT(시퀀스 특권)

이러한 형태의 GRANT문은 시퀀스에 대한 특권을 부여합니다.

호출

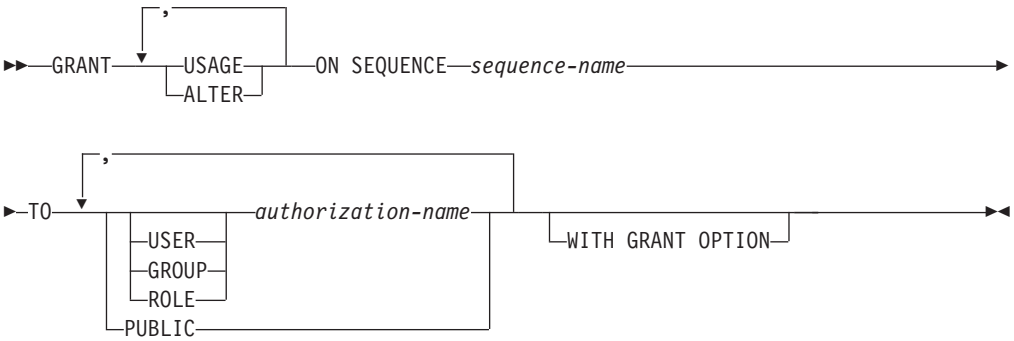
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- *sequence-name*의 식별된 각 특권에 대한 WITH GRANT OPTION
- ACCESSCTRL 또는 SECADM 권한

구문



설명

USAGE

nextval-expression 또는 *prevval-expression*을 사용하여 시퀀스를 참조할 수 있는 특권을 부여합니다.

ALTER

ALTER SEQUENCE문을 사용하여 시퀀스 등록 정보를 변경할 특권을 부여합니다.

ON SEQUENCE *sequence-name*

지정된 특권이 부여될 시퀀스를 식별합니다. 내재적 또는 명시적 스키마를 포함하는 시퀀스 이름은 현재 서버에 이미 존재하는 시퀀스를 고유하게 식별해야 합니다. 이 이름에 의한 시퀀스가 없을 경우 오류가 발생합니다(SQLSTATE 42704).

TO

지정된 특권이 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

PUBLIC

사용자 세트에 지정된 특권을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 지정된 특권을 부여할 수 있도록 합니다.

WITH GRANT OPTION이 생략되면 지정된 *authorization-name*은 다음의 경우에만 다른 사람에게 특권을 부여할 수 있습니다.

- SYSADM 또는 DBADM 권한이 있는 경우
- 다른 소스로부터 지정된 특권을 부여할 수 있는 능력을 제공받은 경우

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.

- 일반적으로 GRANT문은 명령문의 권한 부여 ID가 권한을 부여하도록 허용한 특권 부여를 수행하며 두 개 이상의 특권이 부여되지 않은 경우 경고가 리턴됩니다 (SQLSTATE 01007). 부여된 특권이 없을 경우 오류가 발생합니다(SQLSTATE 42501). 명령문 처리에 사용된 패키지가 MIA에 대해 SQL92E로 설정된 LANGLEVEL로 프리컴파일되었으면 권한 부여자에게 해당 권한 조작의 오브젝트에 대한 특권이 없는 한 경고가 리턴됩니다.

예 :

예 1: 모든 사용자에게 ORG_SEQ라는 시퀀스에 대한 USAGE 특권을 부여하십시오.

```
GRANT USAGE ON SEQUENCE ORG_SEQ TO PUBLIC
```

예 2: 사용자 BOBBY에게 테이블 스페이스 PLANS에 테이블을 작성하고 이 특권을 다른 사용자에게 부여할 수 있는 능력을 부여하십시오.

```
GRANT ALTER ON SEQUENCE GENERATE_ID TO BOBBY WITH GRANT OPTION
```

GRANT(서버 특권)

이러한 형태의 GRANT문은 통과 모드에서 지정된 데이터 소스를 액세스하고 사용할 특권을 부여합니다.

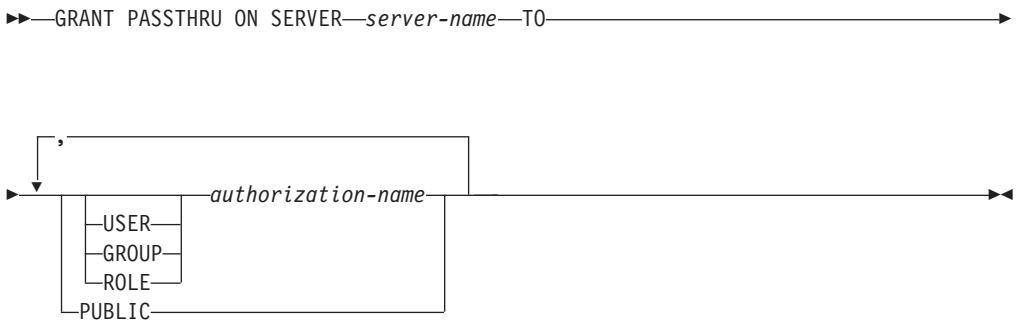
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문



설명

server-name

pass-through 모드에서 특권이 부여되고 있는 데이터 소스에 이름을 지정합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

TO

특권을 부여한 사용자로 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

사용자 세트(권한 부여 ID)에 *server-name*을 pass-through하는 특권을 부여합니다. 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.

예:

예 1: 데이터 소스 SERVALL을 통과하는 특권을 Give R. Smith 및 J. Jones에게 부여하십시오. 이들의 권한 부여 ID는 RSMITH와 JJONES입니다.

```
   GRANT PASSTHRU ON SERVER SERVALL
TO USER RSMITH,
USER JJONES
```

예 2: 권한 부여 ID가 D024인 그룹에 데이터 소스 EASTWING을 통과할 특권을 부여하십시오. 권한 부여 ID 또한 D024인 사용자가 있습니다.

```
   GRANT PASSTHRU ON SERVER EASTWING TO GROUP D024
```

GROUP 키워드가 지정되어야 합니다. 그렇지 않으면, D024가 지정된 그룹의 ID일 뿐만 아니라 사용자의 ID이기 때문에 오류가 발생합니다(SQLSTATE 56092). 그룹 D024의 어떠한 구성원이든 EASTWING에 통과하도록 허용될 것입니다. 그러므로 사용자

D024가 그룹에 속하면 이 사용자는 EASTWING로 통과할 수 없습니다.

GRANT(SETSESSIONUSER 특권)

이러한 형태의 GRANT문은 하나 이상의 권한 부여 ID에 SETSESSIONUSER 특권을 부여합니다. 특권을 사용하여 특권 보유자는 SET SESSION AUTHORIZATION 문으로 지정된 권한 부여 ID 세트 중 하나에 세션 권한 부여를 설정할 수 있습니다.

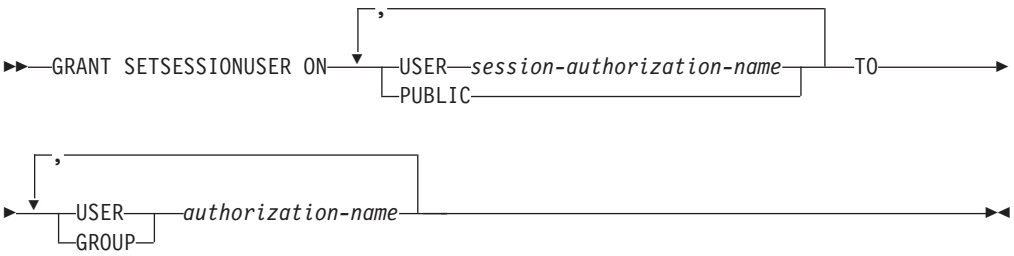
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



설명

SETSESSIONUSER ON

새 권한 부여 ID의 식별을 가정하는 특권을 부여합니다.

USER *session-authorization-name*

*authorization-name*이 SET SESSION AUTHORIZATION문을 사용하여 가정할 수 있는 권한 부여 ID를 지정합니다. *session-authorization-name*은 그룹이 아닌 사용자를 식별해야 합니다.

PUBLIC

권한을 받은 사용자가 SET SESSION AUTHORIZATION문을 사용하여 유효한 권한 부여 ID를 가정하도록 지정합니다.

TO

특권을 부여한 사용자로 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열하십시오.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

규칙

- 지정된 각 *authorization-name*에 대해 USER 및 GROUP이 지정되지 않은 경우
  - 인스턴스에 적용되는 보안 플러그인이 *authorization-name* 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER와 GROUP 둘 다로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용되는 보안 플러그인에 따라 USER로만 정의되거나 정의되지 않으면, USER로 가정됩니다.
  - *authorization-name*이 적용되는 보안 플러그인에 따라 GROUP으로만 정의되면, GROUP으로 가정됩니다.

예:

예 1: 다음 명령문은 사용자 PAUL이 사용자 WALID에게 세션 권한 부여를 설정하고 명령문을 WALID로 실행할 수 있는 권한을 부여합니다.

```
GRANT SETSESSIONUSER ON USER WALID
  TO USER PAUL
```

예 2: 다음 명령문은 사용자 GUYLAINE이 사용자 BOBBY에게 세션 권한 부여를 설정할 수 있는 권한을 부여합니다. 또한 사용자 RICK 및 KEVIN에게 세션 권한 부여를 설정할 수 있는 권한을 부여합니다.

```
GRANT SETSESSIONUSER ON USER BOBBY, USER RICK, USER KEVIN
  TO USER GUYLAINE
```

예 3: 다음 명령문은 사용자 WALID와 ADMINS 및 ACCTG 그룹의 모든 사용자가 임의의 사용자에게 세션 권한 부여를 설정할 수 있는 권한을 부여합니다.

```
GRANT SETSESSIONUSER ON PUBLIC TO USER WALID, GROUP ADMINS, ACCTG
```

GRANT(테이블 스페이스 특권)

이러한 형태의 GRANT문은 테이블 스페이스에 대한 특권을 부여합니다.

호출

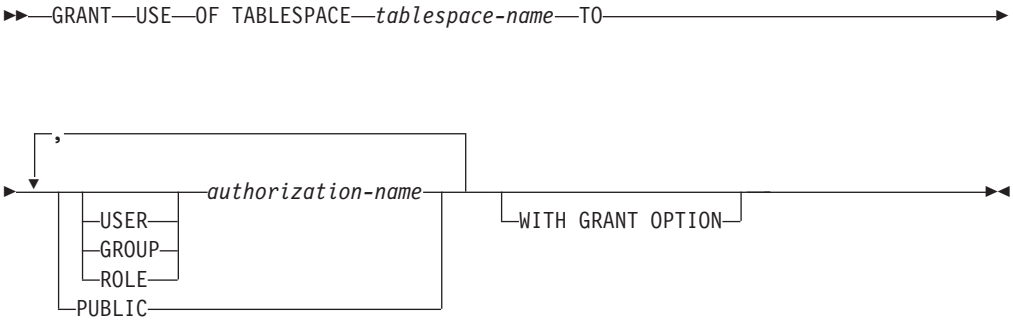
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블 스페이스 사용에 대한 WITH GRANT OPTION
- ACCESSCTRL, SECADM, SYSADM 또는 SYSCTRL 권한

구문



설명

USE

테이블 작성시 테이블 스페이스를 지정하거나 디폴트값으로 설정할 수 있는 특권을 부여합니다. 테이블 스페이스의 작성자는 권한 부여 옵션을 사용하여 USE 특권을 자동으로 받습니다.

OF TABLESPACE *tablespace-name*

USE 특권이 부여될 테이블 스페이스를 식별합니다. 테이블 스페이스는 SYSCATSPACE(SQLSTATE 42838) 또는 SYSTEM TEMPORARY 테이블 스페이스(SQLSTATE 42809)일 수 없습니다.

TO

USE 특권이 부여되는 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

사용자 세트에 USE 권한을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 부여할 수 있도록 합니다.

WITH GRANT OPTION이 생략되면, 지정된 *authorization-name*은 다음의 경우에만 USE 특권을 GRANT할 수 있습니다.

- SYSADM 또는 DBADM 권한이 있는 경우
- 다른 소스의 USE 특권을 GRANT하기 위한 능력을 제공받은 경우

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.

예:

예 1: 사용자 BOBBY에게 테이블 스페이스 PLANS에 테이블을 작성하고 이 특권을 다른 사용자에게 부여할 수 있는 능력을 부여하십시오.

```
GRANT USE OF TABLESPACE PLANS TO BOBBY WITH GRANT OPTION
```

GRANT(테이블, 뷰 또는 별칭 특권)

이러한 형태의 GRANT문은 테이블, 뷰 또는 별칭에 대한 특권을 부여합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

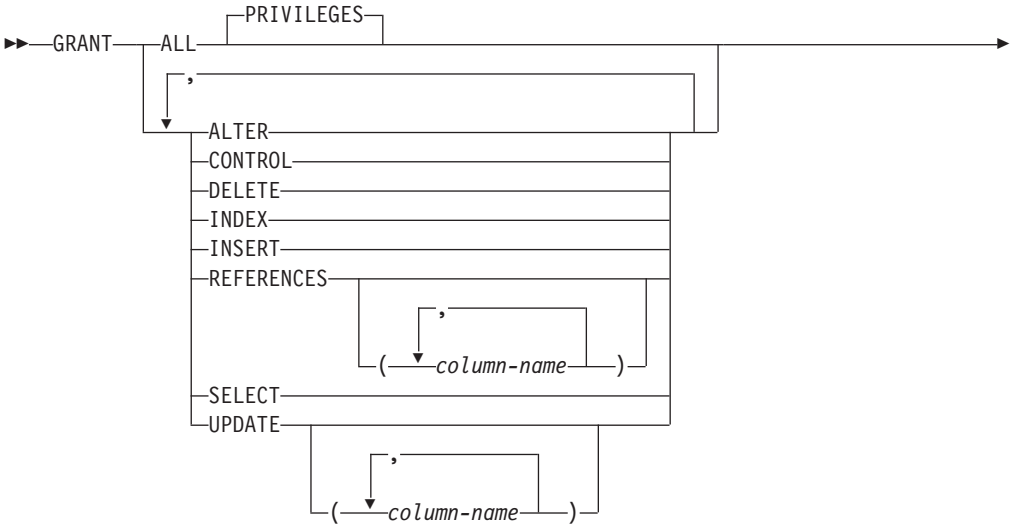
권한 부여

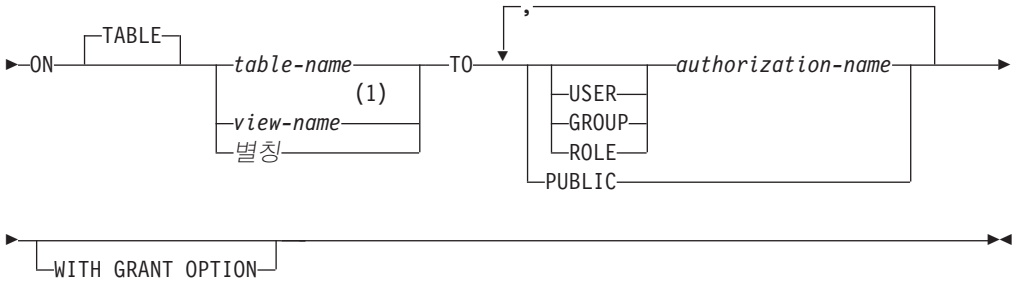
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 참조된 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- 식별된 각 특권에 대한 WITH GRANT OPTION. ALL이 지정된 경우 권한 부여 ID는 식별된 테이블, 뷰 또는 별칭에 대해 어느 정도의 특권이 부여 가능해야 합니다.
- ACCESSCTRL 또는 SECADM 권한

ACCESSCTRL 또는 SECADM 권한은 CONTROL 특권을 부여하거나 카탈로그 테이블 및 뷰에 대한 특권을 부여하는 데 필요합니다.

구문





주:

1 ALTER, INDEX 및 REFERENCES 특권은 뷰에 적용되지 않습니다.

설명

ALL 또는 ALL PRIVILEGES

ON절에 이름 지정된 기본 테이블, 뷰 또는 별칭에 CONTROL을 제외한 적절한 모든 특권을 부여합니다.

명령문의 권한 부여 ID가 테이블, 뷰 또는 별칭에 대한 CONTROL 특권이나 ACCESSCTRL 또는 SECADM 권한을 갖는 경우, 오브젝트에 적용할 수 있는 모든 특권(CONTROL 제외)이 부여됩니다. 그렇지 않으면, 권한 부여된 특권은 명령문의 권한 부여 ID가 식별된 테이블, 뷰 또는 별칭에 대해 갖는 권한 부여 가능한 모든 특권입니다.

ALL이 지정되지 않으면, 특권 목록에 있는 하나 이상의 키워드를 반드시 지정해야 합니다.

ALTER

다음 사항에 특권을 부여합니다.

- 기본 테이블 정의에 컬럼 추가
- 기본 테이블에 대한 기본 키나 고유 제한조건 작성 또는 삭제(drop)
- 기본 테이블에 외부 키 작성 또는 삭제(drop)

  상위 테이블의 각 컬럼에 대한 REFERENCES 특권도 필요합니다.
- 기본 테이블에 점검 제한조건 작성 또는 삭제(drop)
- 기본 테이블에 트리거 작성
- 별칭에 대한 컬럼 추가, 재설정 또는 삭제(drop) 옵션
- 별칭 컬럼 이름이나 데이터 유형 변경
- 기본 테이블 또는 별칭에 주석 추가 또는 변경

CONTROL

다음을 권한 부여합니다.

- 목록에 있는 적절한 모든 특권, 즉,
  - 기본 테이블에 ALTER, CONTROL, DELETE, INSERT, INDEX, REFERENCES, SELECT, UPDATE 등
  - 뷰에 CONTROL, DELETE, INSERT, SELECT, UPDATE 등
  - 별칭에 ALTER, CONTROL, INDEX 및 REFERENCES 등
- 위의 특권(CONTROL은 제외)을 다른 사람에게 권한 부여할 수 있는 기능
- 기본 테이블, 뷰 또는 별칭을 삭제(drop)할 수 있는 능력

  이 능력은 CONTROL 특권을 기초로 하여 다른 사용자에게 부여할 수 없습니다. 확장할 수 있는 유일한 방법을 CONTROL 특원 자체를 부여하는 것으로 이는 ACCESSCTRL 또는 SECADMT 권한이 있는 권한 부여 ID로만 수행할 수 있습니다.
- 테이블 및 인덱스에서 RUNSTATS 유틸리티를 실행하는 능력
- 테이블에서 REORG 유틸리티를 실행하는 능력
- 기본 테이블, 구체화된 쿼리 테이블, 스테이징 테이블에 대하여 SET INTEGRITY 문을 발행할 수 있는 능력

기본 테이블, 구체화된 쿼리 테이블, 스테이징 테이블 또는 별칭의 정의자는 CONTROL 특권을 자동으로 갖게 됩니다.

뷰의 정의자는 fullselect에서 식별되는 모든 테이블, 뷰 및 별칭의 CONTROL 특권을 보유하는 경우, 자동으로 CONTROL 특권을 받습니다.

DELETE

테이블이나 갱신 가능 뷰에서 행을 삭제하는 특권을 부여합니다.

INDEX

테이블에 인덱스를 작성하거나 별칭에 인덱스 스펙을 작성할 특권을 부여합니다. 이 특권은 뷰에서는 권한 부여할 수 없습니다. 인덱스나 인덱스 스펙 작성자는 자동으로 인덱스나 인덱스 스펙에 대한 CONTROL 특권을 가집니다(작성자에게 인덱스나 인덱스 스펙을 삭제(drop)할 권한을 줍니다). 또한, 작성자는 INDEX 특권이 권한 취소될지라도 CONTROL 특권을 보유합니다.

INSERT

테이블이나 갱신 가능한 뷰로 행을 삽입하고 IMPORT 유틸리티를 실행할 특권을 부여합니다.

REFERENCES

상위 테이블로 참조되는 외부 키를 작성 및 삭제(drop)하는 특권을 부여합니다.

명령문의 권한 부여 ID가 다음 중 하나인 경우, 다음과 같습니다.

- ACCESSCTRL 또는 SECADM 권한
- 테이블에 대한 CONTROL 특권

- 테이블에 대해 REFERENCES WITH GRANT OPTION

권한을 부여받은 사용자는 ALTER TABLE문을 통해 추후에 추가된 경우라도 테이블의 모든 컬럼을 상위 키로 사용하여 참조 제한조건을 작성할 수 있습니다. 그렇지 않으면, 권한 부여된 특권은 명령문의 권한 부여 ID가 식별된 테이블에 대해 갖는 권한 부여 가능한 컬럼 REFERENCES 특권입니다.

외부 키가 참조 별칭에 정의될 수 없더라도 별칭에 특권이 부여될 수 있습니다.

REFERENCES (*column-name*,...)
컬럼 목록에 상위 키로 지정된 컬럼만을 사용하여 외부 키를 작성 및 삭제(drop)하기 위한 특권을 부여합니다. 각 *column-name*은 ON절에 식별된 테이블의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다. 유형이 지정된 테이블, 유형이 지정된 뷰 또는 별칭에는 컬럼 레벨의 REFERENCES 특권을 부여할 수 없습니다 (SQLSTATE 42997).

SELECT
다음 사항에 특권을 부여합니다.

- 테이블이나 뷰로부터 행을 검색합니다.
- 테이블에 대한 뷰를 작성합니다.
- 테이블이나 뷰에 대해 EXPORT 유틸리티를 실행합니다.

UPDATE
ON절에서 식별된 테이블이나 갱신 가능 뷰에 UPDATE문을 사용할 수 있는 특권을 부여합니다.

명령문의 권한 부여 ID가 다음 중 하나인 경우, 다음과 같습니다.

- ACCESSCTRL 또는 SECADM 권한
- 테이블 또는 뷰에 대한 CONTROL 특권
- 테이블 또는 뷰에 대한 UPDATE WITH GRANT OPTION

권한을 부여받은 사용자는 ALTER TABLE문을 사용하여 추후에 추가된 컬럼뿐 아니라 권한을 준 사용자가 권한 부여 특권을 갖고 있는 테이블이나 뷰의 갱신 가능한 모든 컬럼을 갱신할 수 있습니다. 그렇지 않으면, 권한 부여된 특권은 명령문 권한 부여 ID가 식별된 테이블이나 뷰에 대해 갖는 권한 부여 가능한 컬럼 UPDATE 특권입니다.

UPDATE (*column-name*,...)
UPDATE문을 사용하여 컬럼 목록에 지정된 컬럼만을 갱신할 수 있는 특권을 부여합니다. 각 *column-name*은 ON절에 식별된 테이블이나 뷰의 컬럼을 식별하는 규정되지 않은 이름이어야 합니다. 유형이 지정된 테이블, 유형이 지정된 뷰 또는 별칭에는 컬럼 레벨의 UPDATE 특권을 부여할 수 없습니다(SQLSTATE 42997).

ON TABLE *table-name* **또는** *view-name* **또는** *nickname*
특권이 권한 부여될 해당 테이블, 뷰 또는 별칭을 지정합니다.

작동 불능 뷰 또는 작동 불능 구체화된 쿼리 테이블에는 특권이 권한 부여되지 않습니다(SQLSTATE 51024). 선언된 임시 테이블에 대한 특권을 부여할 수 없는 경우도 있습니다(SQLSTATE 42995).

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름은 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

다음과 같이 그룹에 부여된 특권은 권한 부여 점검에 사용되지 않습니다.

- 패키지에 있는 정적 DML문의 경우
- CREATE VIEW문을 처리하는 기본 테이블의 경우
- 구체화된 쿼리 테이블에 대한 CREATE TABLE문을 처리하는 기본 테이블의 경우

Linux, UNIX 및 Windows용 DB2 Database에서 그룹에 권한 부여된 테이블 특권은 동적으로 준비된 명령문에만 적용됩니다. 예를 들어, PROJECT 테이블에 대한 INSERT 특권이 그룹 D204에는 권한 부여되고 UBIQUITY(D204의 구성원)에는 부여되지 않았다면, UBIQUITY는 다음과 같은 명령문을 실행할 수 있습니다.

```
EXEC SQL EXECUTE IMMEDIATE :INSERT_STRING;
```

여기서, 문자열의 내용은 다음과 같습니다.

```
INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

그러나 다음과 같은 명령문으로 프로그램을 프리컴파일하거나 바인드할 수 없습니다.

```
   EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

PUBLIC

사용자 세트에 특권을 부여합니다(권한 부여 ID). 자세한 정보는, 『권한 부여,

특권 및 오브젝트 소유권』을 참조하십시오. (정적 SQL문과 CREATE VIEW 문에 대해 PUBLIC에 부여된 특권을 사용하는 이전의 제한사항이 제거되었습니다.)

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 부여할 수 있도록 합니다.

지정된 특권에 CONTROL이 포함되는 경우 WITH GRANT OPTION은 CONTROL을 제외한 적용 가능한 모든 특권에 적용됩니다(SQLSTATE 01516).

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 인스턴스에 대해 유효한 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없으면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 효과 있는 보안 플러그인에 따라 GROUP 또는 USER로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER 및 GROUP으로 정의되면, 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 USER만으로 정의되거나, 정의되지 않으면, USER로 가정합니다.
  - *authorization-name*이 효과 있는 보안 플러그인에 따라 GROUP만으로 정의되면, GROUP으로 가정합니다.
  - *authorization-name*이 데이터베이스에서 ROLE만으로 정의되면, ROLE로 가정합니다.
- 일반적으로 GRANT문은 명령문의 권한 부여 ID가 권한을 부여하도록 허용한 특권 부여를 수행하며 두 개 이상의 특권이 부여되지 않은 경우 경고가 리턴됩니다(SQLSTATE 01007). 부여된 특권이 없을 경우 오류가 발생합니다(SQLSTATE 42501). 명령문 처리에 사용된 패키지가 MIA에 대해 SQL92E로 설정된 LANGLEVEL로 프리컴파일되었으면 권한 부여자에게 해당 권한 조작의 오브젝트에 대한 특권이 없는 한 경고가 리턴됩니다. CONTROL 특권이 지정되면, 명령문의 권한 부여 ID에 ACCESSCTRL 또는 SECADM 권한이 있는 경우에만 특권이 부여됩니다(SQLSTATE 42501).

주

- 테이블 계층의 모든 레벨에서 특권이 독립적으로 부여될 수 있습니다. 슈퍼 테이블에 대한 특권을 가진 사용자는 서브테이블에 영향을 줄 수 있습니다. 예를 들어, 슈퍼 테이블 *T*를 지정하는 갱신은 *T*에 대해 UPDATE 특권을 가지고 있지만 *S*에 대해서는 UPDATE 특권이 없는 사용자에 의해 *T*의 서브테이블 *S*에 있는 행에 대한 변경

으로 나타날 수 있습니다. 사용자는 필요한 특권이 서브테이블에 있는 경우에만 해당 서브테이블을 직접 조작할 수 있습니다.

- 별칭 특권을 부여하면 데이터 소스 오브젝트(테이블이나 뷰) 특권에 아무 영향도 미치지 않습니다. 보통 데이터 소스 특권은 데이터 검색을 시도할 때 별칭이 참조하는 테이블이나 뷰를 위해 필요합니다.
- *호환성*: z/OS용 DB2와의 호환성:
  - 다음 구문은 허용되기도 하고 무시되기도 합니다.
    - PUBLIC AT ALL LOCATIONS

예:

예 1: WESTERN_CR 테이블에 대한 모든 특권을 PUBLIC으로 부여하십시오.

```
GRANT ALL ON WESTERN_CR
   TO PUBLIC
```

예 2: 사용자 PHIL과 CLAIRE가 CALENDAR 테이블을 읽고 새로운 항목을 삽입할 수 있도록 CALENDAR 테이블에 대해 적절한 특권을 부여하십시오. 단 기존의 항목을 변경하거나 제거하지 못하도록 하십시오.

```
GRANT SELECT, INSERT ON CALENDAR
   TO USER  PHIL, USER CLAIRE
```

예 3: COUNCIL 테이블의 모든 특권을 사용자 FRANK에게 권한 부여하고 모든 특권을 다른 사람에게로 확대할 수 있도록 부여하십시오.

```
GRANT ALL ON COUNCIL
   TO USER FRANK WITH GRANT OPTION
```

예 4: 테이블 CORPDATA.EMPLOYEE에 대해 SELECT 특권을 JOHN이라는 사용자에게 권한 부여하십시오. JOHN이라는 사용자는 있고 그룹은 없습니다.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

또는

```
GRANT SELECT
       ON CORPDATA.EMPLOYEE TO USER JOHN
```

예 5: 테이블 CORPDATA.EMPLOYEE에 대해 SELECT 특권을 JOHN이라는 그룹에 권한 부여하십시오. JOHN이라는 그룹은 있고 사용자는 없습니다.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

또는

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO GROUP JOHN
```

예 6: 테이블 T1에 대한 INSERT와 SELECT 특권을 그룹 D024와 사용자 D024에게 권한 부여합니다.

```
GRANT INSERT, SELECT ON TABLE T1
  TO GROUP D024, USER D024
```

이런 경우, D024 그룹 구성원과 D024 사용자 모두는 테이블 T1로 INSERT 및 SELECT할 수 있습니다. 또한, SYSCAT.TABAUTH 카탈로그 뷰에 추가되는 두 개의 행이 있습니다.

예 7: CALENDAR 테이블에 대한 INSERT, SELECT 및 CONTROL 특권을 사용자 FRANK에게 권한 부여합니다. FRANK는 특권을 다른 사용자에게 전달할 수 있어야 합니다.

```
GRANT CONTROL ON TABLE CALENDAR
  TO FRANK WITH GRANT OPTION
```

이 명령문의 실행 결과 CONTROL에 WITH GRANT OPTION이 제공되지 않았다는 경고가 리턴됩니다(SQLSTATE 01516). Frank는 필요한 대로 INSERT와 SELECT 등 CALENDAR에 관한 모든 특권을 부여할 수 있는 능력을 갖게 됩니다. FRANK는 ACCESSCTRL이나 SECADM 권한을 가지고 있는 경우가 아니면 CALENDAR에 대한 CONTROL을 다른 사용자에게 권한 부여할 수 없습니다.

예 8: 사용자 JON이 인덱스가 없는 Oracle 테이블에 대한 별칭을 작성했습니다. 별칭은 ORAREM1입니다. 나중에, Oracle DBA가 이 테이블에 대한 인덱스를 정의했습니다. 사용자 SHAWN은 이제 DB2가 인덱스의 존재를 알아서 옵티마이저가 테이블에 더 효율적으로 액세스할 수 있는 전략을 고안할 수 있기를 원합니다. SHAWN은 ORAREM1에 대한 인덱스 스펙을 작성함으로써 DB2에 인덱스를 알릴 수 있습니다. SHAWN에게 이 별칭에 대한 인덱스 특권을 부여하여, 인덱스 스펙을 작성할 수 있게 하십시오.

```
GRANT INDEX ON NICKNAME ORAREM1
  TO USER SHAWN
```

GRANT(워크로드 특권)

이 형태의 GRANT문은 워크로드에 대한 USAGE 특권을 부여합니다.

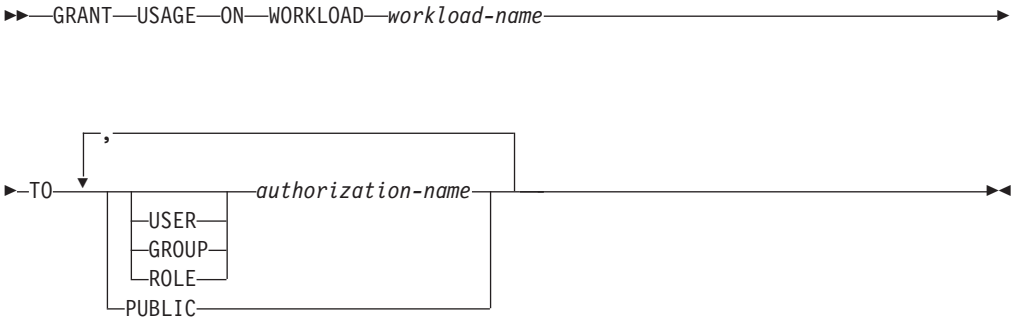
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 ACCESSCTRL, SECADM 또는 WLMADM 권한을 포함해야 합니다.

구문



설명

USAGE

워크로드 사용 특권을 부여합니다. 사용자가 제출하는 작업 단위는 사용자가 USAGE 특권을 가지고 있는 워크로드에만 맵핑됩니다. SYSADM 또는 DBADM 권한을 가지고 있는 현재 서버에 존재하는 워크로드에 대해 자동으로 USAGE 특권을 갖습니다.

ON WORKLOAD *workload-name*

USAGE 특권이 부여될 워크로드를 식별합니다. 이 이름은 한 부분의 이름입니다. *workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). 이름은 'SYSDEFAULTADMWORKLOAD'가 될 수 없습니다(SQLSTATE 42832).

TO

USAGE 특권이 부여되는 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

ROLE

*authorization-name*이 현재 서버에서 기존 역할을 식별함을 지정합니다 (SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다. 권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다 (SQLSTATE 42502).

PUBLIC

사용자(권한 부여 ID) 세트에 USAGE 특권을 부여합니다. 자세한 정보는 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

규칙

- 지정된 *authorization-name*마다, USER, GROUP 또는 ROLE 키워드 중 어느 것도 지정하지 않은 경우
  - 인스턴스에 대해 적용 중인 보안 플러그인이 *authorization-name*의 상태를 판별할 수 없는 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 데이터베이스에서 ROLE로 정의되고 운영 체제에서 GROUP 또는 USER로 정의된 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 USER 및 GROUP 둘 다로 정의된 경우 오류가 리턴됩니다(SQLSTATE 56092).
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 USER로만 정의된 경우 또는 정의되지 않은 경우 USER가 가정됩니다.
  - *authorization-name*이 적용 중인 보안 플러그인에 따라 GROUP으로만 정의된 경우 GROUP이 가정됩니다.
  - *authorization-name*이 데이터베이스에서 ROLE로만 정의된 경우 ROLE이 가정됩니다.

주

- GRANT문은 명령문을 발행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.
- 데이터베이스가 RESTRICT 옵션을 사용하여 작성되는 경우, 디폴트 사용자 워크로드 SYSDEFAULTUSERWORKLOAD의 USAGE 특권은 DBADM 권한을 가지고 있는 사용자가 명시적으로 부여해야 합니다. 데이터베이스가 RESTRICT 옵션을 사용하지 않고 작성되는 경우, SYSDEFAULTUSERWORKLOAD의 USAGE 특권은 데이터베이스 작성 시 PUBLIC에 부여됩니다.

예 :

사용자 LISA에 워크로드 CAMPAIGN을 사용할 수 있는 능력을 부여하십시오.

```
GRANT USAGE ON WORKLOAD CAMPAIGN TO USER LISA
```

GRANT(XSR 오브젝트 특권)

이 양식의 GRANT문은 XSR 오브젝트에 대한 USAGE 특권을 부여합니다.

호출

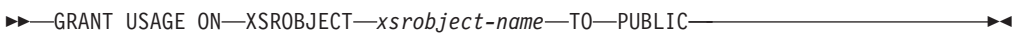
GRANT문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때만(SQLSTATE 42509) 동적으로 준비될 수 있는 실행문입니다.

권한 부여

다음 권한 중 하나가 필요합니다.

- ACCESSCTRL 또는 SECADM 권한
- SYSCAT.XSROBJECTS 카탈로그 뷰의 OWNER 컬럼에 기록된 XSR 오브젝트의 소유자

구문



설명

ON XSROBJECT *xsrobject-name*

이 이름은 USAGE 특권이 부여된 XSR 오브젝트를 식별합니다. 내재적 또는 명시적 스키마 규정자를 포함하는 *xsrobject-name*은 현재 서버에 있는 기존 XSR 오브젝트를 고유하게 식별해야 합니다. 이 이름의 XSR 오브젝트가 없는 경우 오류가 리턴됩니다(SQLSTATE 42704).

TO PUBLIC

사용자(권한 부여 ID) 세트에 USAGE 특권을 부여합니다. 자세한 정보는 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

예 :

모든 사용자에게 XML 스키마 MYSCHEMA에 대한 사용 특권을 부여하십시오.

```
GRANT USAGE ON XSROBJECT MYSCHEMA TO PUBLIC
```

IF

IF문은 조건의 평가를 기반으로 하는 실행 경로를 선택합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

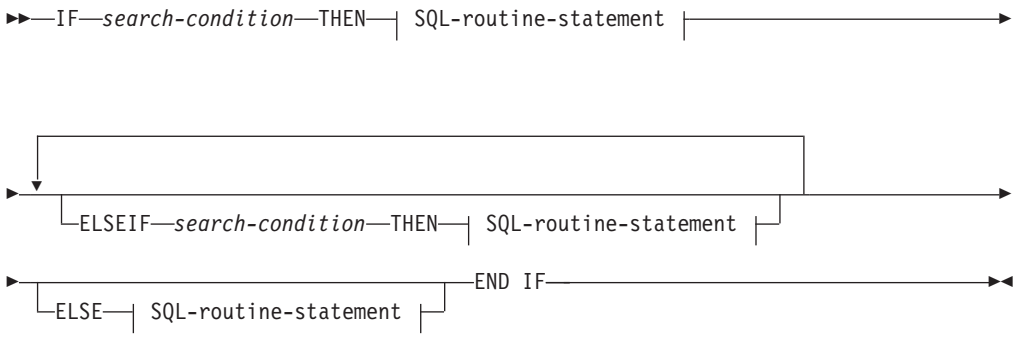
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

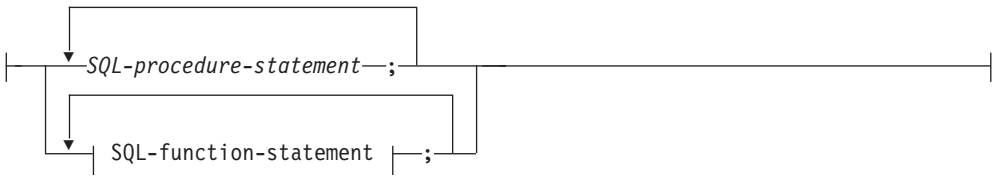
권한 부여

이 명령문은 동적으로 준비될 수 없으므로 그룹 특권을 고려하지 않습니다.

구문



SQL-routine-statement:



설명

search-condition

호출해야 할 SQL문의 조건을 지정합니다. 조건이 알 수 없음 또는 거짓인 경우, 조건이 참이거나 처리가 ELSE절에 도달할 때까지 다음 검색 조건으로 처리를 계속합니다.

SQL-procedure-statement

표시하는 *search-condition*이 참인 경우 호출할 명령문을 지정합니다. *SQL-procedure-statement*는 SQL 프로시저의 컨텍스트에 있을 때 또는 복합 SQL(컴파일된) 명령문 안에 있을 때만 적용할 수 있습니다. 『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

SQL-function-statement

바로 앞의 *search-condition*이 참인 경우 호출할 명령문을 지정합니다. *SQL-function-statement*는 복합 SQL(인라인된)문, SQL 트리거, SQL 함수 또는 SQL 메소드의 컨텍스트에서만 적용할 수 있습니다. 『FOR』의 *SQL-function-statement*를 참조하십시오.

예

다음 SQL 프로시저는 두 가지 IN 매개변수를 승인합니다(직원 수 *employee_number* 및 직원 비율 *rating*). *rating* 값에 따라, employee 테이블이 salary 및 bonus 컬럼의 새 값으로 갱신됩니다.

```
CREATE PROCEDURE UPDATE_SALARY_IF
  (IN employee_number CHAR(6), INOUT 비율 SMALLINT)
  LANGUAGE SQL
  BEGIN
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE EXIT HANDLER FOR not_found
      SET rating = -1;
    IF rating = 1
      THEN UPDATE employee
      SET salary = salary * 1.10, bonus = 1000
      WHERE empno = employee_number;
    ELSEIF rating = 2
      THEN UPDATE employee
      SET salary = salary * 1.05, bonus = 500
      WHERE empno = employee_number;
    ELSE UPDATE employee
      SET salary = salary * 1.03, bonus = 0
      WHERE empno = employee_number;
    END IF;
  END
```

INCLUDE

INCLUDE문은 선언을 소스 프로그램에 삽입합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이는 실행문이 아닙니다.

권한 부여

필요한 권한 없음

구문

설명

SQLCA

SQL 통신 영역(SQLCA)의 설명이 포함됨을 표시합니다.

SQLDA

SQL 디스크립터 영역(SQLCA)의 설명이 포함됨을 표시합니다.

name

프리컴파일되는 소스 프로그램에 포함될 텍스트를 포함하는 외부 파일을 식별합니다. 파일 이름 확장자가 없는 SQL ID나 작은따옴표(' ')로 묶은 리터럴이 될 수 있습니다. SQL ID는 프리컴파일되는 소스 파일의 파일 이름 확장자를 사용합니다. 따옴표로 묶은 리터럴에서 파일 이름 확장자가 제공되지 않으면 어떤 것도 사용되지 않습니다.

주

- 프로그램이 프리컴파일될 때 INCLUDE문은 소스 명령문에 의해 교체됩니다. 따라서 INCLUDE문은 결과 소스 명령문이 컴파일러에 승인될 수 있도록 프로그램 내의 지점에 지정해야 합니다.
- 외부 소스 파일은 *name*에 지정된 호스트 언어로 작성해야 합니다. 18바이트보다 크거나 SQL ID에서 허용되지 않는 문자를 포함하는 경우 작은따옴표로 묶어야 합니다. INCLUDE *name* 문은 순환되지 않게 중첩될 수 있습니다(예를 들어, A 및 B가 모듈이고 A에 INCLUDE *name* 문이 포함된 경우, A가 B를 호출한 후 B가 A를 호출하는 것은 유효하지 않음).
- SQL92E 값을 사용하여 LANGLEVEL 프리컴파일 옵션을 지정하는 경우 INCLUDE SQLCA를 지정해야 합니다. SQLSTATE 및 SQLCODE 변수는 호스트 변수 선언 섹션 내에 정의할 수 있습니다.

예 :

C 프로그램에 SQLCA를 포함하십시오.

```
EXEC SQL INCLUDE SQLCA;

   EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';

   EXEC SQL OPEN C1;

   while (SQLCODE==0) {
       EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;

(Print results)

}

   EXEC SQL CLOSE C1;
```

INSERT

INSERT문은 테이블, 별칭 또는 뷰에 행을 삽입하거나 지정된 fullselect의 뷰, 별칭 또는 하위 테이블을 삽입합니다. 별칭에 행을 삽입하는 것은 별칭을 참조하는 데이터 소스 오브젝트에 행을 삽입하는 것입니다. 이 뷰에서 삽입 조작을 위해 정의된 INSTEAD OF 트리거가 없을 경우 뷰에 행을 삽입하면 그 뷰의 기반이 되는 테이블에도 행이 삽입됩니다. 이와 같은 트리거가 정의되었을 경우 트리거가 대신 실행됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 목표 테이블, 뷰 또는 별칭에 대한 INSERT 특권
- 목표 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- DATAACCESS 권한

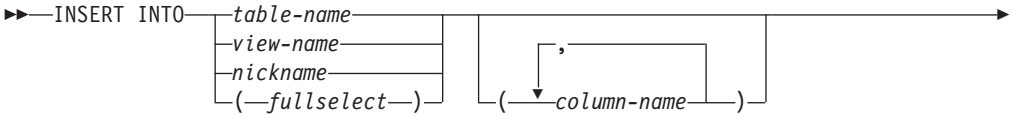
또한, INSERT문에서 사용되는 fullselect에 참조되는 각 테이블, 뷰 또는 별칭마다 명령문의 권한 부여 ID가 보유하는 특권에는 최소한 다음 중 하나가 포함되어야 합니다.

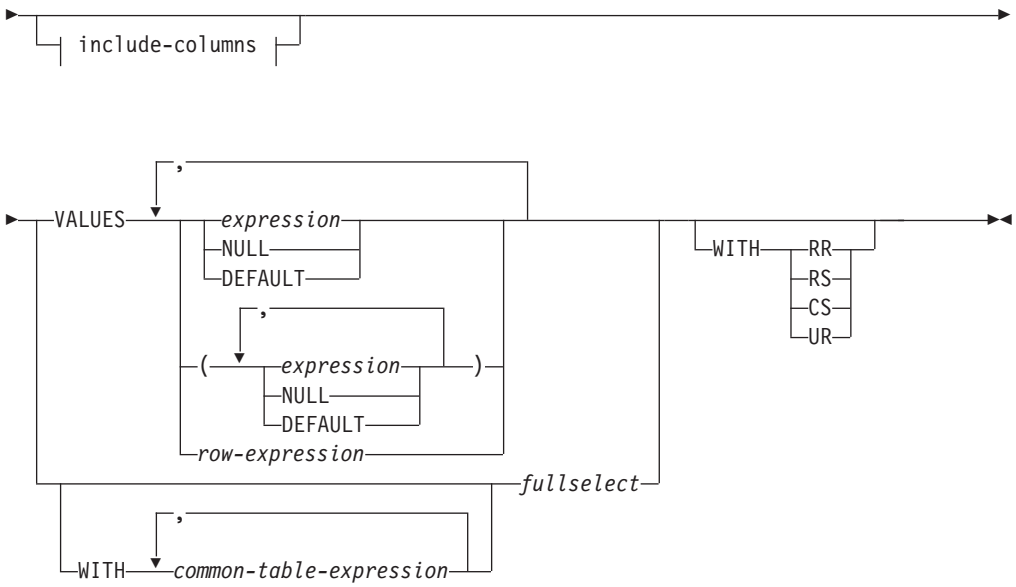
- SELECT 특권
- CONTROL 특권
- DATAACCESS 권한

정적 INSERT문에 대해서는 GROUP 특권이 점검되지 않습니다.

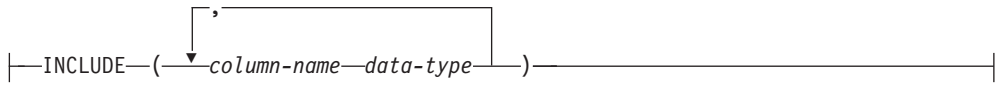
삽입 조작의 목표가 별칭일 경우에는 데이터 소스에서 명령문을 실행하기 전에는 데이터 소스의 오브젝트에 대한 특권이 무시됩니다. 이 때 데이터 소스에 연결하는 데 사용되는 권한 부여 ID에는 데이터 소스에서 해당 오브젝트에 조작을 수행하는 데 필요한 특권이 있어야 합니다. 명령문의 권한 부여 ID는 데이터 소스에서 서로 다른 권한 부여 ID로 맵핑될 수 있습니다.

구문





include-columns:



설명

INTO *table-name*, *view-name*, *nickname* 또는 *(fullselect)*

삽입 조작의 오브젝트를 식별합니다. 이름은 응용프로그램 서버(AS)에 있는 테이블, 뷰 또는 별칭을 식별해야 하지만 주제 뷰에 대한 삽입 조작에 대해 INSTEAD OF 트리거가 정의되어 있지 않았을 경우 카탈로그 테이블, 시스템 유지보수 구체화된 쿼리 테이블, 카탈로그 테이블 뷰 또는 읽기 전용 뷰를 식별해서는 안됩니다. 별칭에 삽입된 행은 별칭을 참조하는 데이터 소스 오브젝트에 위치하게 됩니다.

삽입 조작의 오브젝트가 fullselect인 경우 CREATE VIEW문의 설명에 있는 『삽입 가능한 뷰』의 주 항목에 정의되어 있듯이 fullselect는 삽입 가능해야 합니다.

이 뷰에서 삽입 조작을 위해 존재하는 INSTEAD OF 트리거가 없을 경우 다음에서 파생된 뷰 컬럼에는 값이 삽입될 수 없습니다.

- 상수, 표현식 또는 스칼라 함수
- 다른 뷰 컬럼과 같은 기본 테이블 컬럼

삽입 조작의 오브젝트가 그러한 컬럼을 가진 뷰인 경우 컬럼 이름 목록이 지정되어야 하고, 그 목록은 이들 컬럼을 식별해서는 안됩니다.

행이 하위 기본 테이블 중 한 테이블만의 점검 제한조건을 충족할 때만 UNION ALL을 사용하여 정의한 뷰 또는 fullselect에 행을 삽입할 수 있습니다. 행이 여러 테이블의 점검 제한조건을 충족하거나 어떤 테이블의 점검 제한조건도 충족하지 않으면 오류가 발생합니다(SQLSTATE 23513).

(*column-name*,...)

삽입 값이 제공될 컬럼을 지정합니다. 각 이름은 지정된 테이블이나 뷰의 컬럼 또는 fullselect에 있는 컬럼을 식별해야 합니다. 같은 컬럼은 한 번 이상 식별될 수 없습니다. 삽입 값을 허용할 수 없는 컬럼(예: 표현식 기반의 컬럼)은 식별해서는 안 됩니다.

컬럼 목록을 생략하면 테이블(내재적으로 숨겨지지 않은 테이블) 또는 뷰, 또는 fullselect의 선택 목록에서 모든 항목 의 모든 컬럼이 왼쪽에서 오른쪽으로 식별되도록 목록이 내재적으로 지정됩니다. 이 목록은 명령문이 준비된 이후 테이블에 추가된 컬럼을 포함하지 않을 때 만들어집니다.

include-columns

table-name 또는 *view-name*의 컬럼과 함께 INSERT문의 중간 결과 테이블에 포함되는 일련의 컬럼을 지정합니다(컬럼이 fullselect의 FROM절에서 중첩되는 경우). *include-columns*는 *table-name* 또는 *view-name*으로 사용되는 컬럼의 목록 끝에 추가됩니다.

INCLUDE

컬럼 목록이 INSERT문의 중간 결과 테이블에 포함되도록 지정합니다. INSERT문이 fullselect의 FROM절에서 중첩된 경우에만 이 절을 지정할 수 있습니다.

column-name

INSERT문에 있는 중간 결과 테이블의 컬럼을 지정합니다. 이름은 *table-name* 또는 *view-name*에서 컬럼 또는 다른 Include 컬럼의 이름과 동일해서는 안됩니다(SQLSTATE 42711).

data-type

Include 컬럼의 데이터 유형을 지정합니다. 데이터 유형은 CREATE TABLE문으로 지원된 하나여야 합니다.

VALUES

삽입될 하나 이상의 값 행을 사용합니다.

VALUES절에 지정된 각 행은 행 변수를 사용하는 경우를 제외하고 내재되거나 명시된 컬럼 목록 및 INCLUDE 절에서 식별된 컬럼에 지정 가능해야 합니다. 괄호 안의 행 값 목록이 지정된 경우, 첫 번째 값이 목록의 첫 번째 컬럼에 삽입되고 두 번째 값이 두 번째 컬럼으로 삽입되는 방식으로 작동합니다. 행표현식이 지정된 경우 행 유형의 필드 수는 내재된 컬럼 또는 명시적 컬럼 목록의 이름 수와 일치해야 합니다.

expression

*expression*에는 『표현식』에 정의된 모든 표현식을 사용할 수 있습니다. *expression*이 행 유형인 경우, 괄호로 표시되지 않아야 합니다.

NULL

널(NULL) 입력 가능 컬럼에 대해서만 지정되어야 하는 널(NULL) 값을 지정합니다.

DEFAULT

디폴트값이 사용되도록 지정합니다. DEFAULT 지정 결과는 다음과 같이 컬럼 정의 방법에 따라 달라집니다.

- 표현식을 기반으로 컬럼이 생성된 컬럼으로 정의된 경우 시스템이 표현식을 근거로 컬럼 값을 생성합니다.
- IDENTITY절이 사용되면 데이터베이스 관리 프로그램에서 값을 생성합니다.
- ROW CHANGE TIMESTAMP절이 사용되면 삽입된 각 행의 값은 데이터베이스 파티션 내의 테이블 파티션에 대해 고유한 시간소인으로 데이터베이스 관리 프로그램에서 생성됩니다.
- WITH DEFAULT절이 사용되면 삽입된 값이 컬럼에 대해 정의됩니다(『CREATE TABLE』의 *default-clause* 참조).
- NOT NULL절이 사용되고 GENERATED절이 사용되지 않거나 WITH DEFAULT절이 사용되지 않거나 DEFAULT NULL이 사용되면 해당 컬럼에 대해 DEFAULT 키워드를 지정할 수 없습니다(SQLSTATE 23502).
- 별칭에 삽입할 때는 데이터 소스가 쿼리 언어 구문에서 DEFAULT 키워드를 지원할 경우에만 DEFAULT 키워드가 INSERT문을 통해 데이터 소스에 패스됩니다.

row-expression

컬럼 이름을 포함하지 않는 "행 표현식"에 설명되어 있는 유형의 모든 행 표현식을 지정합니다. 행의 필드 수는 입력의 목표와 일치해야 하며 각 필드를 해당 컬럼에 지정할 수 있어야 합니다.

WITH *common-table-expression*

뒤에 나올 fullselect과 함께 사용할 공통 테이블 표현식을 정의합니다.

fullselect

fullselect의 결과 테이블 형태로 새로운 행 세트를 지정합니다. 하나 이상 또는 하나도 없을 수 있습니다. 결과 테이블이 비어 있는 경우 SQLCODE는 +100으로 설정되고 SQLSTATE는 '02000'으로 설정됩니다.

INSERT의 기본 오브젝트와 fullselect의 기본 오브젝트 또는 fullselect의 서브쿼리가 같은 테이블인 경우 행이 삽입되기 전에 fullselect가 완전히 평가됩니다.

결과 테이블의 컬럼 수는 컬럼 목록의 이름 수와 같아야 합니다. 결과의 첫 번째 컬럼 값은 목록의 첫 번째 컬럼에, 두 번째 값은 두 번째 컬럼에 삽입되는 방식으로 삽입됩니다.

WITH

fullselect가 실행되는 위치에 분리 레벨을 지정합니다.

RR

반복 읽기

RS

읽기 안정성

CS

커서 안정성

UR

언커미트 읽기

명령문의 디폴트 분리 레벨은 명령문이 바운드되는 패키지의 분리 레벨입니다. WITH 절은 항상 명령문의 디폴트 분리 레벨을 사용하는 별칭에 영향을 주지 않습니다.

규칙

- *트리거:* INSERT문이 트리거를 실행시킵니다. 트리거로 인해 다른 명령문이 실행될 수도 있고, 삽입 값으로 인해 오류가 발생할 수도 있습니다. 뷰에 삽입하는 조작으로 인해 INSTEAD OF 트리거가 발생하면 트리거를 발생시킨 뷰나 이 뷰의 하위 테이블에 대해서가 아니라 트리거에서 수행된 갱신에 대하여 유효성, 참조 무결성, 제한조건이 점검됩니다.
- *디폴트값:* 컬럼 목록에 없는 컬럼에 삽입된 값은 컬럼의 디폴트값이거나 널(NULL)입니다. 널(NULL) 값을 허용하지 않고 NOT NULL WITH DEFAULT로 정의되지 않은 컬럼은 컬럼 목록에 포함되어야 합니다. 마찬가지로, 뷰에 삽입하는 경우 뷰에 없는 기본 테이블의 컬럼에 삽입되는 값은 컬럼의 디폴트값 또는 널(NULL)입니다. 그러므로 뷰에 없는 기본 테이블의 모든 컬럼에는 디폴트값이 있거나 널(NULL) 값을 허용해야 합니다. GENERATED ALWAYS절로 정의된 생성된 컬럼에 삽입할 수 있는 유일한 값은 DEFAULT입니다(SQLSTATE 428C9).
- *길이:* 컬럼의 삽입 값이 숫자라면, 컬럼은 숫자의 정수 부분을 나타내는 숫자 컬럼이어야 합니다. 컬럼의 삽입 값이 문자열인 경우 컬럼은 적어도 그 문자열 길이의 길이 속성을 지닌 문자열 컬럼이 되거나, 문자열이 날짜, 시간 또는 시간 소인을 나타내는 경우 날짜 시간 컬럼이 됩니다.
- *지정:* 삽입 값은 특정 지정 규칙에 따라 컬럼에 지정됩니다.
- *유효성:* 이름 지정된 테이블 또는 이름 지정된 뷰의 기본 테이블에 하나 이상의 고유 인덱스가 있는 경우 테이블에 삽입된 각 행은 이러한 인덱스로 강요되는 제한조건에 따라야 합니다. 정의에 WITH CHECK OPTION이 포함된 뷰가 이름 지정되면 그 뷰에 삽입된 각 행은 뷰의 정의에 따라야 합니다. 이러한 상황에 적용되는 규칙에 대한 설명은 『CREATE VIEW』를 참조하십시오.

- *참조 무결성:* 테이블에 정의된 각 제한조건의 경우 외부 키의 널(NULL)이 아닌 각 삽입 값은 상위 테이블의 기본 키 값과 동일해야 합니다.
- *점검 제한조건:* 삽입 값은 테이블에 정의된 점검 제한조건의 점검 조건을 만족시켜야 합니다. 정의된 점검 제한조건을 가진 테이블에 INSERT를 수행하는 경우 삽입되는 각 행에 대해 의무 규정 조건이 한 번 평가됩니다.
- *XML 값:* XML 컬럼에 삽입된 값은 바르게 구성된 XML 문서여야 합니다 (SQLSTATE 2200M).
- *보안 규정:* 식별된 테이블이나 식별된 뷰의 기본 테이블이 보안 규정으로 보호된 경우, 세션 권한 부여 ID에는 허용되는 레이블 기반 액세스 제어(LBAC) 증명서가 있어야 합니다.
  - 명시적으로 제공되는 데이터 값에 대해 보호되는 모든 컬럼에 대한 쓰기 액세스 (SQLSTATE 42512)
  - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션으로 작성되었던 보안 규정에 대해 DB2SECURITYLABEL 컬럼에서 제공된 명시적인 값에 대한 쓰기 액세스(SQLSTATE 23523)

  내재적인 값이 DB2SECURITYLABEL 컬럼에 대해 사용되면(SQLSTATE 23523) 이 세션 권한 부여 ID는 보안 규정에 대한 쓰기 액세스에 대해 보안 레이블을 부여해야 하며, 다음과 같은 상황에서 발생할 수 있습니다.
  - DB2SECURITYLABEL 컬럼 값이 명시적으로 제공되지 않은 경우
  - DB2SECURITYLABEL 컬럼 값이 명시적으로 제공되지만 세션 권한 부여 ID에 해당 값에 대한 쓰기 액세스가 없으며, 보안 규정이 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션으로 작성된 경우

주

- INSERT문 실행 후 SQLCA의 SQLERRD(3) 부분의 세 번째 변수 값은 삽입 조작에 넘겨진 행의 수를 나타냅니다. SQL 프로시저 명령문에서는 GET DIAGNOSTICS문의 ROW_COUNT 변수를 사용하여 이 값을 검색할 수 있습니다. SQLERRD(5)에는 모든 트리거된 삽입, 갱신 및 삭제 조작 수의 합계가 들어 있습니다.
- 적절하게 잠겨 있지 않으면 INSERT문을 실행할 때 하나 이상의 배타적 잠금이 생깁니다. 잠금이 해제될 때까지 삽입된 행은 다음에 의해서만 액세스될 수 있습니다.
  - 삽입을 수행하는 응용프로그램 프로세스
  - 읽기 전용 커서를 통해 분리 레벨 UR을 사용하는 다른 응용프로그램 프로세스, SELECT INTO문 또는 서브쿼리에 사용되는 Subselect
- 잠금에 대한 자세한 내용은 COMMIT, ROLLBACK 및 LOCK TABLE문의 설명을 참조하십시오.

- 파티션된 데이터베이스에 대해 응용프로그램이 실행되고, INSERT BUF 옵션으로 바운드된 경우 EXECUTE IMMEDIATE를 사용하여 처리되지 않은 INSERT WITH VALUES문은 버퍼됩니다. DB2는 이러한 INSERT문이 응용프로그램 로직의 루프 내에서 처리되는 것으로 가정합니다. 명령문이 완료에 실행되기 보다는 하나 이상의 버퍼에 새 행 값을 버퍼하려고 합니다. 행을 테이블에 실제로 삽입하는 것의 결과는 응용프로그램 INSERT 로직이 있는 비동기로 나중에 수행됩니다. 이러한 비동기 삽입은 INSERT와 관련된 오류가 응용프로그램에서 INSERT에 후속하는 다른 SQL 문에 리턴되게 할 수 있습니다.

  이것은 INSERT의 성능을 상당히 향상시킬 잠재력을 가지고 있지만 오류 처리의 비동기성 때문에 깨끗한 데이터와 함께 사용하는 것이 가장 좋습니다.
- 식별 컬럼이 있는 테이블에 행이 삽입될 경우 DB2는 해당 식별 컬럼에 대한 값을 생성합니다.
  - DB2는 GENERATED ALWAYS 식별 컬럼에 대한 값을 항상 생성합니다.
  - GENERATED BY DEFAULT 컬럼의 경우, VALUES절이나 subselect를 사용하여 값이 명시적으로 지정되지 않았다면 DB2가 값을 생성합니다.

  DB2가 생성하는 첫 번째 값은 식별 컬럼에 대한 START WITH 스펙의 값입니다.
- 사용자 정의 구별 유형 식별 컬럼에 대한 값이 삽입되는 경우, 전체 계산이 소스 유형으로 수행되고 그 값이 실제로 컬럼에 지정되기 전에 결과가 구별 유형으로 캐스트됩니다. (이전 값은 계산에 앞서 소스 유형으로 캐스팅되지 않습니다.)
- GENERATED ALWAYS 식별 컬럼에 삽입될 때 DB2는 항상 해당 컬럼에 대한 값을 생성하고 사용자는 삽입시 값을 지정해서는 안됩니다. GENERATED ALWAYS 식별 컬럼이 VALUES절의 디폴트값이 아닌 다른 값으로 INSERT문의 컬럼 목록에 나열되면 오류가 발생합니다(SQLSTATE 428C9).

  예를 들어, EMPID가 GENERATED ALWAYS인 식별 컬럼으로 정의되었다고 가정할 때 다음 명령은 오류를 발생시킵니다.

  ```
  INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
    VALUES (:hv_valid_emp_id, :hv_name, :hv_addr)
  ```

  오류를 발생시킵니다.
- GENERATED ALWAYS ROW CHANGE TIMESTAMP 컬럼에 삽입될 때 DB2는 항상 해당 컬럼에 대한 값을 생성하고 사용자는 삽입 시 값을 지정해서는 안됩니다(SQLSTATE 428C9). DB2에서 생성된 값은 데이터베이스 파티션에 삽입된 각 행에 대해 고유합니다.
- GENERATED BY DEFAULT 컬럼에 삽입될 때 DB2는 컬럼의 실제 값이 VALUES절 내에서 또는 subselect로부터 해당 컬럼의 실제 값이 지정되게 합니다. 그러나 VALUES절에 값이 지정되면 DB2가 값의 검증을 수행하지 않습니다. IDENTITY 컬럼 값의 고유성을 보증하기 위해서는 식별 컬럼의 고유 인덱스를 작성해야 합니다.

컬럼 목록을 지정하지 않고 GENERATED BY DEFAULT 식별 컬럼이 있는 테이블에 삽입할 때, VALUES절은 DEFAULT 키워드를 지정하여 식별 컬럼의 값을 나타낼 수 있습니다. DB2는 해당 식별 컬럼의 값을 생성합니다.

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
  VALUES (DEFAULT, :hv_name, :hv_addr)
```

이 예에서 EMPID는 식별 컬럼으로 정의되므로 DB2가 이 컬럼에 삽입된 값을 생성합니다.

- subselect로 식별 컬럼에 삽입 규칙은 VALUES절로 삽입하는 규칙과 유사합니다. 식별 컬럼이 GENERATED BY DEFAULT로 정의된 경우에만 식별 컬럼의 값을 지정할 수 있습니다.

  예를 들어, T1 및 T2가 *intcol1*과 *identcol2* 컬럼 모두를 포함하는 동일한 정의를 갖는 테이블이라고 가정합시다. (두 컬럼 모두는 INTEGER 유형이고 두 번째 컬럼은 식별 속성을 가집니다.) 다음 삽입을 고려하십시오.

  ```
  INSERT INTO T2
    SELECT *
    FROM T1
  ```

  이 예는 논리적으로 다음과 같습니다.

  ```
  INSERT INTO T2 (intcol1,identcol2)
    SELECT intcol1, identcol2
    FROM T1
  ```

  두 경우 모두에서 INSERT문은 T2의 식별 컬럼에 대한 명시적 값을 제공합니다. 이 명시적 스펙에는 식별 컬럼의 값을 제공할 수 있으나 T2의 식별 컬럼을 GENERATED BY DEFAULT로 정의해야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 428C9).

  테이블에 GENERATED ALWAYS 식별로 정의된 컬럼이 있는 경우 여전히 같은 정의를 갖는 테이블로부터 다른 모든 컬럼을 전파할 수 있습니다. 예를 들어, 위에 기술된 예제 테이블 T1 및 T2가 제공된 경우 다음 SQL을 사용하여 intcol1 값을 T1에서 T2로 전파할 수 있습니다.

  ```
  INSERT INTO T2 (intcol1)
    SELECT intcol1
    FROM T1
  ```

  identcol2가 컬럼 목록에 지정되지 않았기 때문에 (생성된) 디폴트값으로 채워집니다.

- 컬럼이 GENERATED ALWAYS 식별 컬럼 또는 ROW CHANGE TIMESTAMP 컬럼으로 정의된 단일 컬럼 테이블에 행을 삽입하는 경우 DEFAULT 키워드로 VALUES절을 지정할 수 있습니다. 이 경우 응용프로그램은 테이블에 대한 값을 제공하지 않으며 DB2가 해당 식별 또는 ROW CHANGE TIMESTAMP 컬럼의 값을 생성합니다.

```
INSERT INTO IDTABLE
  VALUES(DEFAULT)
```

컬럼이 식별 속성을 갖는 동일한 단일 컬럼 테이블의 경우, 단일 INSERT문으로 여러 행을 삽입하려면 다음 INSERT문을 사용할 수 있습니다.

```
INSERT INTO IDTABLE
  VALUES (DEFAULT), (DEFAULT), (DEFAULT), (DEFAULT)
```

- DB2가 식별 컬럼의 값을 생성할 때 생성된 값이 소비되고, 값이 필요한 다음 번에는 DB2가 새 값을 생성합니다. 이는 식별 컬럼을 포함하는 INSERT문이 실패하거나 롤백되는 경우에도 적용됩니다.

  예를 들어, 식별 컬럼에 대한 고유 인덱스가 작성되었다고 가정합니다. 식별 컬럼의 값 생성 시 중복되는 키 위반이 발견되면 오류가 발생하고(SQLSTATE 23505) 해당 식별 컬럼에 대해 생성된 값이 소비된 것으로 간주됩니다. 이는 식별 컬럼이 GENERATED BY DEFAULT로 정의되었고 시스템이 새 값을 생성하려하나 사용자가 이전 INSERT문에 식별 컬럼 값을 명시적으로 지정한 경우 발생합니다. 이 경우 동일한 INSERT문을 다시 발행하면 성공할 수 있습니다. DB2가 식별 컬럼에 대한 다음 값을 생성하고, 해당 다음 값이 고유하게 되고 해당 INSERT문이 성공하는 것이 가능합니다.
- 식별 컬럼 값을 생성할 때 식별 컬럼의 최대값(또는 내림차순으로 최소값)을 초과하면 오류가 발생합니다(SQLSTATE 23522). 이러한 상황에서 사용자는 범위가 더 큰 식별 컬럼이 있는 새 테이블을 삭제하고 작성해야 합니다. 즉, 더 큰 값 범위가 가능하도록 데이터 유형을 변경하거나 컬럼 값을 증가시키십시오.

  예를 들어, 데이터 유형이 SMALLINT인 식별 컬럼을 정의했을 수 있으며 결과적으로 해당 컬럼은 지정 가능한 값을 모두 소비합니다. 식별 컬럼을 INTEGER로 다시 정의하려면 데이터를 언로드해야 하고 테이블을 삭제한 다음 해당 컬럼에 대한 새 정의로 다시 작성하고 데이터를 다시 로드해야 합니다. 테이블이 다시 정의되는 경우 DB2가 생성하는 다음 값이 원래 시퀀스의 다음 값이 되도록 식별 컬럼에 대해 START WITH 값을 지정해야 합니다. 끝 값을 결정하려면 데이터를 언로드하기 전에 식별 컬럼의 MAX(오름차순의 경우) 또는 식별 컬럼의 MIN(내림차순의 경우)을 사용하여 쿼리를 발행하십시오.

예:

예 1: 다음 스펙으로 DEPARTMENT 테이블에 새로운 부서를 삽입하십시오.

- 부서 번호(DEPTNO)는 E31'입니다.
- 부서 이름(DEPTNAME)은 'ARCHITECTURE'입니다.
- 번호가 '00390'인 사람에 의해 관리됩니다(MGRNO).
- 'E01' 부서로 보고합니다(ADMRDEPT).

```
INSERT INTO DEPARTMENT
  VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

예 2 예 1에서와 같이 하나의 명령문을 사용하여 두 개의 새로운 부서를 DEPARTMENT 테이블에 삽입할 때 새로운 부서에 관리자를 지정하지 마십시오.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT )
  VALUES ('E31', 'ARCHITECTURE', 'E01')
```

예 3: 예 2에서와 같이 하나의 명령문을 사용하여 두 개의 새로운 부서를 DEPARTMENT 테이블에 삽입할 때 새로운 부서에 관리자를 지정하지 마십시오.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('B11', 'PURCHASING', 'B01'),
         ('E41', 'DATABASE ADMINISTRATION', 'E01')
```

예 4: EMP_ACT 테이블과 같은 컬럼을 갖는 임시 테이블 MA_EMP_ACT를 작성하십시오. 문자 'MA'로 시작하는 프로젝트 번호(PROJNO)가 있는 EMP_ACT 테이블의 행으로 MA_EMP_ACT를 로드하십시오.

```
CREATE TABLE MA_EMP_ACT
         ( EMPNO CHAR(6)  NOT NULL,
                       PROJNO CHAR(6)  NOT NULL,
                       ACTNO SMALLINT  NOT NULL,
                       EMPTIME DEC(5,2),
                       EMSTDATE DATE,
                       EMENDATE  DATE )
INSERT INTO MA_EMP_ACT
  SELECT * FROM EMP_ACT
    WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 5: C 프로그램 명령문을 사용하여 스켈리톤 프로젝트를 PROJECT 테이블에 추가하십시오. 호스트 변수를 통해 프로젝트 번호(PROJNO), 프로젝트 이름(PROJNAME), 부서 번호(DEPTNO) 및 담당 직원(RESPEMP)을 알아내십시오. 현재 날짜를 프로젝트 시작 날짜(PRSTDATE)로 사용하십시오. NULL 값을 테이블에 남아 있는 컬럼에 지정하십시오.

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
     VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);
```

예 6: INSERT문을 SELECT문의 *data-change-table-reference*로 지정하십시오. VALUES절에 값을 지정하는 추가 Include 컬럼을 정의하여 삽입된 행의 정렬 컬럼으로 사용하십시오.

```
SELECT inorder.ordernum
  FROM (INSERT INTO orders(custno)INCLUDE (insertnum integer)
    VALUES(:cnum1, 1), (:cnum2, 2)) InsertedOrders
  ORDER BY insertnum;
```

예 7: C 프로그램 명령문을 사용하여 문서를 DOCUMENTS 테이블에 추가하십시오. SQL TYPE IS XML AS BLOB_FILE로 바인드하는 호스트 변수에서 문서 ID(DOCID) 컬럼 및 문서 데이터(XMLDOC) 컬럼값을 확보하십시오.

```
EXEC SQL INSERT INTO DOCUMENTS
       (DOCID, XMLDOC) VALUES (:docid, :xmldoc)
```

예 8: 다음 INSERT문의 경우, 테이블 SALARY_INFO가 3개의 컬럼으로 정의되며 마지막 컬럼은 내재적으로 숨겨진 ROW CHANGE TIMESTAMP 컬럼입니다. 다음 명령문에서 내재적으로 숨겨진 컬럼은 컬럼 목록에서 명시적으로 참조되며 값은 VALUES 절에서 제공됩니다.

```
INSERT INTO SALARY_INFO (LEVEL, SALARY, UPDATE_TIME)
  VALUES (2, 30000, CURRENT TIMESTAMP)
```

다음 INSERT문은 내재적 컬럼 목록을 사용합니다. 내재적 컬럼 목록은 내재적으로 숨겨진 컬럼을 포함하지 않으므로 VALUES절은 다른 두 개의 컬럼에 대한 값만을 포함합니다.

```
INSERT INTO SALARY_INFO VALUES (2, 30000)
```

이 경우 UPDATE_TIME 컬럼이 디폴트 값을 갖도록 정의되어야 하며 디폴트 값은 삽입된 행으로 사용됩니다.

ITERATE

ITERATE문은 제어 흐름이 레이블된 루프의 시작 부분으로 리턴하게 합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

권한 부여

필요한 권한 없음

구문

설명

label

DB2가 제어의 흐름을 전달할 FOR, LOOP, REPEAT 또는 WHILE문의 레이블을 지정합니다.

예

이 예는 커서를 사용하여 새 부서에 대한 정보를 리턴합니다. *not_found* 조건 핸들러가 호출된 경우, 제어의 흐름이 루프 밖으로 전달합니다. *v_dept* 값이 'D11'인 경우, ITERATE문은 제어 흐름을 다시 LOOP문의 맨 위로 전달합니다. 그렇지 않으면, 새 행이 DEPARTMENT 테이블에 삽입됩니다.

```
CREATE PROCEDURE ITERATOR()
  LANGUAGE SQL
  BEGIN
    DECLARE v_dept CHAR(3);
    DECLARE v_deptname VARCHAR(29);
    DECLARE v_admdept CHAR(3);
    DECLARE at_end INTEGER DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE c1 CURSOR FOR
      SELECT deptno, deptname, admrdept
      FROM department
      ORDER BY deptno;
    DECLARE CONTINUE HANDLER FOR not_found
      SET at_end = 1;
```

```
  OPEN c1;
  ins_loop:
  LOOP
    FETCH c1 INTO v_dept, v_deptname, v_admdept;
    IF at_end = 1 THEN
      LEAVE ins_loop;
    ELSEIF v_dept = 'D11' THEN
      ITERATE ins_loop;
    END IF;
    INSERT INTO department (deptno, deptname, admrdept)
    VALUES ('NEW', v_deptname, v_admdept);
  END LOOP;
  CLOSE c1;
END
```

LEAVE

LEAVE문은 프로그램 제어를 루프나 복합 명령문 밖으로 전송합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

권한 부여

필요한 권한 없음

구문

설명

label
종료할 복합, FOR, LOOP, REPEAT 또는 WHILE문의 레이블을 지정합니다.

주

- LEAVE문이 제어를 복합 명령문 밖으로 전송하면, 복합 명령문의 모든 열린 커서가 닫힙니다(결과 세트를 리턴하는 데 사용되는 커서는 제외).

예

다음 예는 커서 *c1*에 대한 데이터를 페치하는 루프를 포함합니다. SQL 변수 *at_end*의 값이 영(0)이 아닌 경우, LEAVE문은 제어를 루프 밖으로 전송합니다.

```
CREATE PROCEDURE LEAVE_LOOP(OUT counter INTEGER)
  LANGUAGE SQL
  BEGIN
    DECLARE v_counter INTEGER;
    DECLARE v_firstnme VARCHAR(12);
    DECLARE v_midinit CHAR(1);
    DECLARE v_lastname VARCHAR(15);
    DECLARE at_end SMALLINT DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE c1 CURSOR FOR
      SELECT firstnme, midinit, lastname
        FROM employee;
    DECLARE CONTINUE HANDLER for not_found
```

```
    SET at_end = 1;
  SET v_counter = 0;
  OPEN c1;
  fetch_loop:
  LOOP
    FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
    IF at_end <> 0 THEN LEAVE fetch_loop;
    END IF;
    SET v_counter = v_counter + 1;
  END LOOP fetch_loop;
  SET counter = v_counter;
  CLOSE c1;
END
```

LOCK TABLE

LOCK TABLE문은 동시적인 응용프로그램 프로세스가 테이블을 사용하거나 변경하지 못하도록 합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블의 SELECT 특권
- 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

구문

설명

table-name 또는 *nickname*

테이블이나 별칭을 식별합니다. *table-name*은 응용프로그램 서버(AS)에 있는 테이블을 식별해야 하며, 카탈로그 테이블, 작성된 임시 테이블, 또는 선언된 임시 테이블을 식별해서는 안됩니다(SQLSTATE 42995). *table-name*이 유형이 지정된 테이블인 경우 테이블 계층의 루트 테이블이어야 합니다(SQLSTATE 428DR). 별칭을 지정하면 DB2는 별칭이 참조하는 데이터 소스의 하위 오브젝트(테이블 또는 뷰)를 잠급니다.

IN SHARE MODE

동시에 사용 중인 응용프로그램 프로세스에서 테이블을 읽을 수는 있으나 다른 조작은 실행하지 못하도록 하십시오.

IN EXCLUSIVE MODE

동시에 사용 중인 응용프로그램 프로세스에서 테이블을 조작하지 못하도록 하십시오. EXCLUSIVE MODE는 분리 레벨 언커미트 읽기에서 동시에 실행 중인 응용프로그램 프로세스에서 테이블을 읽을 수 있도록 합니다.

주

- 동시에 조작하지 못하도록 잠금을 사용하면 동시에 조작할 수 없습니다. 적절한 잠금이 이미 존재하는 경우, LOCK TABLE문의 실행 중에 반드시 잠금을 얻을 수 있는 것은 아닙니다. 동시 조작을 방지하는 잠금은 최소한 작업 단위(UOW)가 종료될 때까지는 그대로 있습니다.
- 파티션된 데이터베이스에서 데이터베이스 파티션 그룹의 첫 번째 데이터베이스 파티션(가장 낮은 번호를 갖는 데이터베이스 파티션)에서 가장 먼저 테이블이 잠기면 그 다음에 또 다른 데이터베이스 파티션에서 테이블이 잠깁니다. LOCK TABLE문이 인터럽트될 경우, 테이블이 일부 데이터베이스 파티션에서는 잠기지만 다른 데이터베이스 파티션에서는 잠기지 않을 수 있습니다. 이 경우, 또 하나의 LOCK TABLE문을 실행하여 모든 데이터베이스 파티션에서 잠금을 완료하거나, COMMIT 또는 ROLLBACK문을 실행하여 현재의 잠금을 해제하십시오.
- 이 명령문은 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션에 영향을 줍니다.
- 파티션된 테이블에서는 테이블 레벨의 LOCK TABLE문 잠금만이 가능합니다. 데이터 파티션 잠금은 확보되지 않습니다.

예 :

테이블 EMP에서 잠금을 수행하십시오. 다른 프로그램이 테이블을 읽거나 갱신하지 못하도록 합니다.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

LOOP

LOOP문은 명령문이나 명령문 그룹의 실행을 반복합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

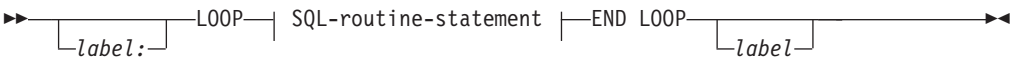
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

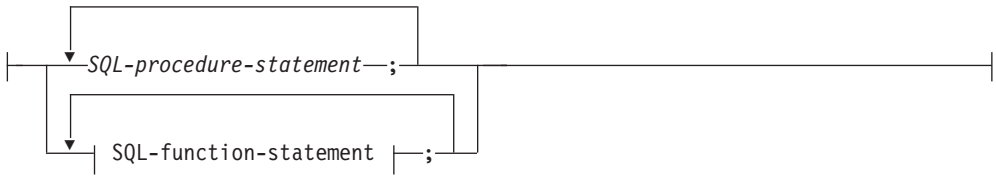
권한 부여

LOOP문을 호출하기 위해 필요한 특권은 없습니다. 단, 명령문의 권한 부여 ID가 LOOP문에 임베디드(embedded)되는 SQL문을 호출하는 데 필요한 특권을 가지고 있어야 합니다.

구문



SQL-routine-statement:



설명

label

LOOP문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 해당 레이블이 LEAVE 및 ITERATE문에 지정될 수 있습니다. 종료 레이블이 지정된 경우, 이와 일치하는 시작 레이블이 지정되어야 합니다.

SQL-procedure-statement

루프에서 호출될 SQL문을 지정합니다. *SQL-procedure-statement*는 SQL 프로시저의 컨텍스트 또는 복합 SQL(컴파일된)문에서만 적용할 수 있습니다. 『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

SQL-function-statement

루프에서 호출될 SQL문을 지정합니다. *SQL-function-statement*는 SQL 함수, SQL 메소드의 컨텍스트 또는 복합 SQL(인라인된)문에서만 적용할 수 있습니다. 『FOR』의 *SQL-function-statement*를 참조하십시오.

예

이 프로시저는 LOOP문을 사용하여 employee 테이블의 값을 페치합니다. 루프가 반복할 때마다, OUT 매개변수 *counter*가 증가하고 *v_midinit* 값이 점검되어 값이 1줄 간격(' ')이 아닌지 확인합니다. *v_midinit*가 1줄 간격인 경우, LEAVE문은 루프 밖 제어의 플로우를 전달합니다.

```
CREATE PROCEDURE LOOP_UNTIL_SPACE(OUT 카운터 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
```

MERGE

MERGE문은 소스의 데이터를 사용하여(테이블 참조 결과) 목표(테이블 또는 뷰, 또는 fullselect의 하위 테이블 또는 뷰)를 갱신합니다. 소스와 일치하는 목표의 행은 지정된 대로 삭제되거나 갱신될 수 있으며 목표에 존재하지 않는 행은 삽입될 수 있습니다. 뷰에서 행을 갱신, 삭제 또는 삽입하면 뷰 하위 테이블에서 행이 갱신, 삭제 또는 삽입됩니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 삽입 조작이 지정되면 테이블 또는 뷰에 대한 INSERT 특권, 삭제 조작이 지정되면 테이블 또는 뷰에 대한 DELETE 특권, 갱신 조작이 지정되면 다음 중 하나가 있어야 합니다.
  - 테이블 또는 뷰에 대한 UPDATE 특권
  - 갱신될 각 컬럼에 대한 UPDATE 특권
- 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

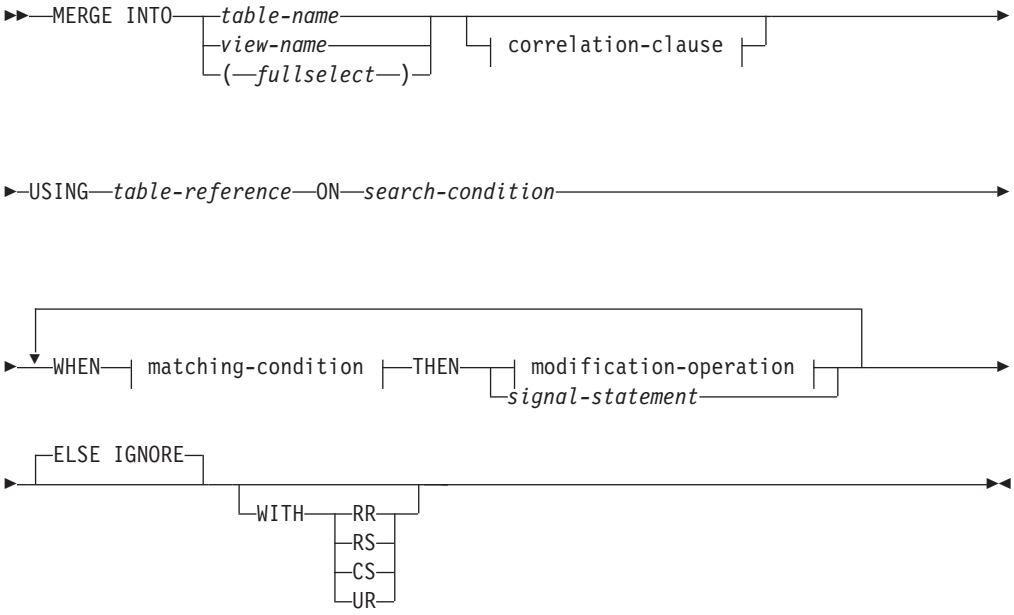
- *table-reference*에서 식별되는 모든 테이블 또는 뷰에 대한 SELECT 특권
- *table-reference*에서 식별되는 테이블 또는 뷰에 대한 CONTROL 특권
- DATAACCESS 권한

search-condition, *insert-operation* 또는 *assignment-clause*에 서브쿼리가 들어 있으면, 명령문의 권한 부여 ID에서 보유하는 특권은 또한 최소한 다음 중 하나를 포함해야 합니다.

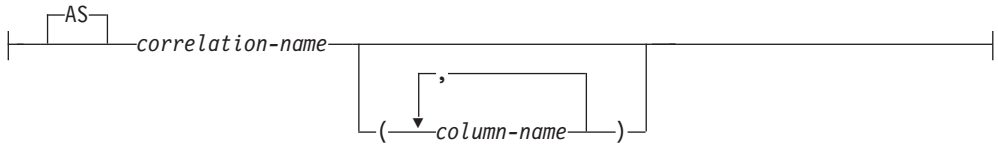
- 서브쿼리에서 식별되는 모든 테이블 또는 뷰에 대한 SELECT 특권
- 서브쿼리에서 식별되는 테이블 또는 뷰에 대한 CONTROL 특권
- DATAACCESS 권한

함수를 참조하는 표현식이 지정되면 특권 세트에는 함수를 실행하는 데 필요한 권한이 포함되어야 합니다.

구문

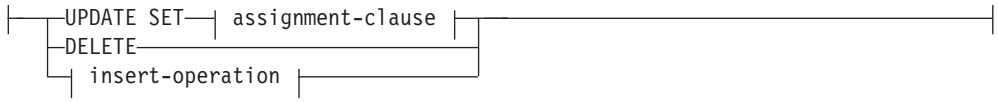


correlation-clause:

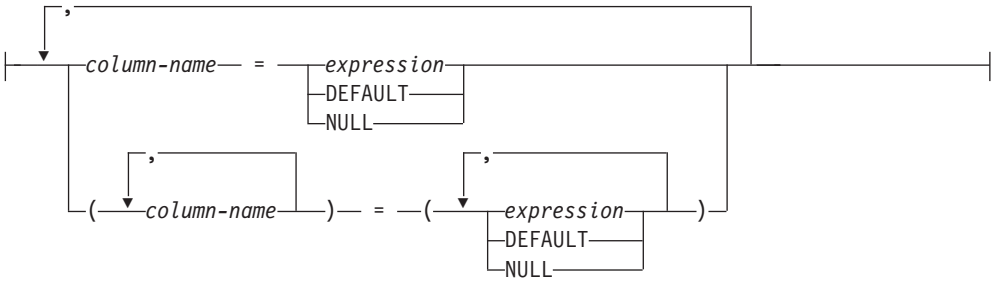


matching-condition:

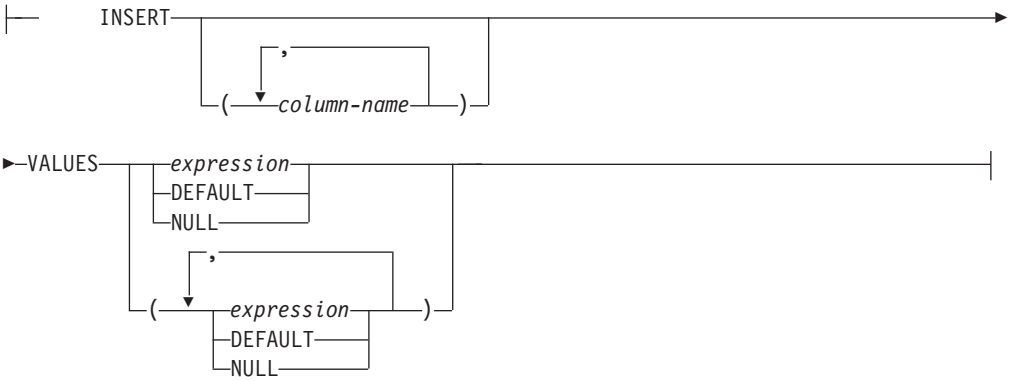
modification-operation:



assignment-clause:



insert-operation:



설명

table-name, *view-name* 또는 *(fullselect)*

병합의 갱신, 삭제 또는 삽입 조작 목표를 나타냅니다. 이름은 현재 서버에 존재하는 테이블 또는 뷰를 나타내어야 하지만 카탈로그 테이블, 시스템에서 유지보수하는 구체화된 쿼리 테이블, 읽기 전용 뷰 또는 NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의된 서브쿼리나 루틴을 참조하는 WHERE절이 직접 또는 간접적으로 들어 있는 뷰를 나타내서는 안됩니다(SQLSTATE 42807).

병합 조작의 목표가 fullselect인 경우, fullselect는 CREATE VIEW문 설명에 있는 『갱신 가능한 뷰』, 『삭제 가능한 뷰』 또는 『삽입 가능한 뷰』의 주 항목에 정의된 대로 갱신 가능, 삭제 가능 또는 삽입 가능해야 합니다.

correlation-clause

search-condition 내에서 또는 *assignment-clause*의 오른쪽에서 사용되어 테이블, 뷰 또는 fullselect를 지정할 수 있습니다. *correlation-clause*의 설명은 『Subselect』의 『table-reference』를 참조하십시오.

USING *table-reference*

목표로 병합될 결과 테이블로 일련의 행을 지정합니다. 결과 테이블이 비어 있으면 경고가 표시됩니다(SQLSTATE 02000).

ON *search-condition*

병합의 갱신 및 삭제 조작에서 사용될 *table-reference*의 행과 병합의 삽입 조작에

사용될 행을 지정합니다. *search-condition*은 *table-reference*의 결과 테이블과 목표 테이블의 각 행에 적용됩니다. *search-condition*의 결과가 참인 *table-reference*의 결과 테이블 행에서는 지정된 갱신 또는 삭제 조작이 수행됩니다. *search-condition*의 결과가 참이 아닌 *table-reference*의 결과 테이블 행에서는, 지정된 삽입 조작이 수행됩니다.

*search-condition*에는 다음과 같은 제한사항이 있습니다(SQLSTATE 42972).

- 서브쿼리, 스칼라 등을 포함할 수 없는 경우
- 참조 값이 오브젝트 ID 컬럼이 아닌 다른 값일 경우 비참조 조작 또는 DEREF 함수를 포함할 수 없음
- SQL 함수를 포함할 수 없는 경우
- XMLQUERY 또는 XMLEXISTS 표현식을 포함할 수 없는 경우
- *search-condition* 표현식에서 참조되는 컬럼이 목표 테이블, 뷰 또는 *table-reference*의 컬럼이어야 함
- 완전 외부 조인의 *join-condition* 표현식에서 참조된 함수가 결정적이어야 하고 외부 조치가 없어야 함

WHEN *matching-condition*

modification-operation 또는 *signal-statement*가 실행되는 조건을 지정합니다. 각 *matching-condition*은 스펙 순서로 평가됩니다. *matching-condition*이 참으로 평가하는 행은 후속의 일치하는 조건에서 고려되지 않습니다.

MATCHED

ON 검색 조건이 참인 행에서 실행할 조작을 나타냅니다. UPDATE, DELETE 또는 *signal-statement*가 THEN 다음에 지정될 수 있습니다.

AND *search-condition*

THEN 다음에 수행될 조작의 ON 검색 조건과 일치하는 행에 대해 적용될 추가 검색 조건을 지정합니다.

NOT MATCHED

ON 검색 조건이 거짓이거나 알 수 없는 행에서 실행할 조작을 나타냅니다. INSERT 또는 *signal-statement*만이 THEN 다음에 지정될 수 있습니다.

AND *search-condition*

THEN 다음에 수행될 조작의 ON 검색 조건과 일치하지 않는 행에 대해 적용될 추가 검색 조건을 지정합니다.

THEN *modification-operation*

*matching-condition*이 참으로 평가될 때 실행할 조작을 지정합니다.

UPDATE SET

*matching-condition*이 참으로 평가하는 행에 실행할 갱신 조작을 지정합니다.

assignment-clause

컬럼 갱신사항의 목록을 지정합니다.

column-name

갱신될 컬럼을 식별합니다. *column-name*은 지정된 테이블 또는 뷰의 컬럼을 식별해야 하지만 스칼라 함수, 상수 또는 표현식에서 얻은 뷰 컬럼은 아닙니다. 컬럼은 두 번 이상 지정되서는 안됩니다(SQLSTATE 42701).

뷰의 또 다른 컬럼과 동일한 컬럼에서 얻은 뷰 컬럼은 갱신될 수 있으나 두 컬럼 모두 동일한 MERGE문에서 갱신될 수 없습니다 (SQLSTATE 42701).

expression

컬럼의 새 값을 나타냅니다. *expression*에 집계 함수가 포함되어서는 안됩니다(SQLSTATE 42903).

*expression*은 *table-name* 또는 *view-name*의 컬럼을 참조할 수 있습니다. 갱신되는 각 행에 대해 표현식에 있는 컬럼 값은 갱신되기 전의 행에 있는 컬럼 값입니다.

DEFAULT

디폴트값은 컬럼에 지정됩니다. DEFAULT는 디폴트값이 있는 컬럼에만 지정될 수 있습니다. 데이터 유형의 디폴트값에 대한 자세한 내용은 『CREATE TABLE』문의 DEFAULT절 설명을 참조하십시오.

DEFAULT는 GENERATED ALWAYS로 정의된 컬럼에 지정되어야 합니다. 유효한 값은 GENERATED BY DEFAULT로 정의된 컬럼에 지정될 수 있습니다.

NULL

컬럼의 새 값으로 널(NULL) 값을 지정합니다. 널(NULL) 입력 가능 컬럼에 대해서만 널(NULL)을 지정합니다(SQLSTATE 23502).

DELETE

*matching-condition*이 참으로 평가하는 행에 실행할 삭제 조작을 지정합니다.

insert-operation

*matching-condition*이 참으로 평가하는 행에 실행할 삽입 조작을 지정합니다.

INSERT

삽입 조작에 사용할 컬럼 이름 및 행 값 표현식 목록을 사용합니다.

행 값 표현식의 행 값 수는 삽입 컬럼 목록의 이름 수와 같아야 합니다. 첫 번째 값은 목록의 첫 번째 컬럼에, 두 번째 값은 두 번째 컬럼에 삽입됩니다.

(*column-name,...*)

삽입 값이 제공될 컬럼을 지정합니다. 각 이름은 테이블 또는 뷰의 컬럼을 식별해야 합니다. 같은 컬럼은 한 번 이상 식별될 수 없습니다(SQLSTATE 42701). 삽입 값을 허용할 수 없는 뷰 컬럼은 식별해서는 안됩니다. 다음에서 파생된 뷰 컬럼에는 값이 삽입될 수 없습니다.

- 상수, 표현식 또는 스칼라 함수
- 다른 뷰 컬럼과 같은 기본 테이블 컬럼

조작의 오브젝트가 그러한 컬럼을 가진 뷰인 경우 컬럼 이름 목록이 지정되어야 하고 그 목록은 이들 컬럼을 식별해서는 안됩니다.

컬럼 목록을 생략하면 테이블(내재적으로 숨겨진 것으로 정의되지 않음) 또는 뷰의 모든 컬럼이 왼쪽에서 오른쪽으로 식별되도록 목록이 내재적으로 지정됩니다. 이 목록은 명령문이 준비된 이후 테이블에 추가된 컬럼을 포함하지 않을 때 만들어집니다.

VALUES

삽입될 하나 이상의 값 행을 사용합니다.

expression

모든 표현식은 컬럼 이름이 포함되어 있지 않습니다(SQLSTATE 42703).

DEFAULT

디폴트값은 컬럼에 지정됩니다. DEFAULT는 디폴트값이 있는 컬럼에만 지정될 수 있습니다. 데이터 유형의 디폴트값에 대한 자세한 내용은 『CREATE TABLE』문의 DEFAULT절 설명을 참조하십시오.

DEFAULT는 GENERATED ALWAYS로 정의된 컬럼에 지정되어야 합니다. 유효한 값은 GENERATED BY DEFAULT로 정의된 컬럼에 지정될 수 있습니다.

NULL

컬럼의 값으로 널(NULL) 값을 지정합니다. 널(NULL) 입력 가능 컬럼에 대해서만 널(NULL)을 지정합니다(SQLSTATE 23502).

signal-statement

*matching-condition*이 참으로 평가될 때 오류를 리턴하기 위해 실행할 SIGNAL문을 지정합니다.

ELSE IGNORE

*matching-condition*이 참으로 평가하지 않는 행에는 취할 조치가 없음을 지정합니다.

WITH

MERGE문이 실행되는 분리 레벨을 지정합니다.

RR
반복 읽기

RS
읽기 안정성

CS
커서 안정성

UR
언커미트 읽기

명령문의 디폴트 분리 레벨은 명령문이 바운드되는 패키지의 분리 레벨입니다.

규칙

- 둘 이상의 *modification-operation*(UPDATE SET, DELETE 또는 *insert-operation*) 또는 *signal-statement*가 하나의 MERGE문에 지정될 수 있습니다.
- 목표의 각 행은 한 번만 작동될 수 있습니다. 목표의 행은 *table-reference*의 결과 테이블에서 한 행만 일치하는 것으로 식별될 수 있습니다(SQLSTATE 21506). 중첩된 SQL 조작(INSTEAD OF 트리거를 제외한 RI 또는 트리거)은 UPDATE, DELETE, INSERT 또는 MERGE문의 목표로 목표 테이블(또는 동일한 테이블 계층 구조 내의 테이블)을 지정할 수 없습니다(SQLSTATE 27000).
- ***보안 규정:*** 식별된 목표 테이블 또는 식별된 목표 뷰의 기본 테이블이 보안 규정으로 보호되는 경우, 세션 권한 부여 ID에는 다음 액세스 유형을 허용하는 레이블 기반 액세스 제어(LBAC) 증명서가 있어야 합니다.
  - 갱신 조작의 경우:
    - 갱신되는 모든 보호 컬럼에 대한 쓰기 액세스(SQLSTATE 42512)
    - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 사용하여 작성된 보안 규정에 대해 DB2SECURITYLABEL 컬럼에 제공되는 명시적 값에 대한 쓰기 액세스(SQLSTATE 23523)
    - 갱신되는 모든 행에 대한 읽기 및 쓰기 액세스(SQLSTATE 42519)

    DB2SECURITYLABEL 컬럼에 내재된 값이 사용되는 경우 세션 권한 부여 ID에는 쓰기 액세스에 대한 보안 규정의 보안 레이블도 부여되어야 합니다(SQLSTATE 23523). 내재된 값이 사용되는 경우는 다음과 같습니다.
    - DB2SECURITYLABEL 컬럼이 갱신될 컬럼 목록에 포함되어 있지 않음(따라서 세션 권한 부여 ID의 쓰기 액세스에 대한 보안 레이블로 내재적으로 갱신됨)

- DB2SECURITYLABEL 컬럼 값이 명시적으로 제공되지만 세션 권한 부여 ID에 해당 값에 대한 쓰기 액세스 권한이 없으며, OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 사용하여 보안 규정이 작성됨

– 삭제 조작의 경우:
  - 모든 보호 컬럼에 대한 쓰기 액세스(SQLSTATE 42512)
  - 삭제 대상으로 선택된 모든 행에 대한 읽기 및 쓰기 액세스(SQLSTATE 42519)

– 삽입 조작의 경우:
  - 데이터 값이 명시적으로 제공되는 모든 보호 컬럼에 대한 쓰기 액세스 (SQLSTATE 42512)
  - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 사용하여 작성된 보안 규정에 대해 DB2SECURITYLABEL 컬럼에 제공되는 명시적 값에 대한 쓰기 액세스(SQLSTATE 23523)

  DB2SECURITYLABEL 컬럼에 내재된 값이 사용되는 경우 세션 권한 부여 ID에는 쓰기 액세스에 대한 보안 규정의 보안 레이블도 부여되어야 합니다 (SQLSTATE 23523). 내재된 값이 사용되는 경우는 다음과 같습니다.
  - DB2SECURITYLABEL 컬럼 값이 명시적으로 제공되지 않음
  - DB2SECURITYLABEL 컬럼 값이 명시적으로 제공되지만 세션 권한 부여 ID에 해당 값에 대한 쓰기 액세스 권한이 없으며, OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 사용하여 보안 규정이 작성됨

MERGE문의 갱신, 삽입 또는 삭제 조작 부분에 영향을 주는 다른 규칙에 대해서는 해당 명령문 설정의 "규칙" 절을 참조하십시오.

주

• *처리 순서*:

1. 소스 및 목표에서 처리할 일련의 행을 판별하십시오. CURRENT TIMESTAMP가 이 명령문에 사용되면 전체 명령문에 대해 하나의 클럭 읽기만 수행됩니다.
2. MATCHED 또는 NOT MATCHED로 이러한 행을 분류하려면, ON절을 사용하십시오.
3. WHEN절의 모든 *matching-condition*을 평가하십시오.
4. 임의의 *assignment-clause* 및 *insert-operation*에서 임의의 *expression*을 평가하십시오.
5. 각 *signal-statement*를 실행하십시오.
6. 스펙 순서로 적용 가능한 행에 각 *modification-operation*을 적용하십시오. 각 *modification-operation*으로 활성화된 제한조건 및 트리거는 *modification-*

*operation*에 실행됩니다. 명령문 레벨 트리거는 *modification-operation*을 만족시키는 행이 없는 경우에도 활성화됩니다. 각 *modification-operation*은 후속 *modification-operation*의 트리거 및 참조 제한조건에 영향을 줄 수 있습니다.

- ***명령문 레벨 원자 수***: MERGE문 실행 중에 오류가 발생하면 전체 명령문이 롤백됩니다.
- ***갱신된 행 수***: MERGE문이 실행을 완료하면 SQLCA에서 GET DIAGNOSTICS 및 SQLERRD(3)의 ROW_COUNT 항목 값은 ELSE IGNORE절로 식별되는 행을 제외하고 MERGE문에서 작동되는 행 수입니다. SQLERRD(3)의 값은 제한조건 또는 트리거 결과로 작동되는 행 수를 포함하지 않습니다. SQLERRD(5)의 값은 이러한 행 수를 포함합니다.
- ***삽입된 행이 갱신 안 됨***: MERGE문이 실행되기 전에 이미 존재하지 않는 목표의 행이 갱신되지 않았습니다. 즉, MERGE문으로 삽입된 행이 갱신되지 않았습니다.
- ***INSTEAD OF 트리거***: 뷰가 MERGE문의 목표로 지정되면 INSTEAD OF 트리거가 뷰에 대해 정의되지 않거나 INSTEAD OF 트리거가 각각의 갱신, 삭제 및 삽입 조작에 정의되어야 합니다(SQLSTATE 428FZ).

예:

예 1: 설명이 변경된 활동의 경우 아카이브 테이블에서 설명을 갱신하고, 새 활동의 경우 archive 테이블에 삽입하십시오. archive 및 activity 테이블 둘 모두에는 기본 키로서의 활동이 있습니다.

```
MERGE INTO archive ar
USING (SELECT activity, description FROM activities) ac
ON (ar.activity = ac.activity)
WHEN MATCHED THEN
        UPDATE SET
              description = ac.description
WHEN NOT MATCHED THEN
   INSERT
              (activity, description)
      VALUES (ac.activity, ac.description)
```

예 2: shipment 테이블을 사용하여 일치하는 행의 shipment 테이블에서 부품 계수별로 수량을 증가시켜 inventory 테이블로 행을 병합하십시오. 그렇지 않으면, inventory 테이블에 새 *partno*를 삽입하십시오.

```
MERGE INTO inventory AS in
USING (SELECT partno, description, count FROM shipment
       WHERE shipment.partno IS NOT NULL) AS sh
ON (in.partno = sh.partno)
WHEN MATCHED THEN
        UPDATE SET
              description = sh.description,
              quantity = in.quantity + sh.count
WHEN NOT MATCHED THEN
```

```
   INSERT
              (partno, description, quantity)
      VALUES (sh.partno, sh.description, sh.count)
```

예 3: transaction 테이블을 사용하여 어카운트 ID에 대해 일련의 트랜잭션의 밸런스를 갱신하고 이미 존재하지 않는 통합된 트랜잭션에서 새 어카운트를 삽입하며 어카운트 테이블에 행을 병합하십시오.

```
MERGE INTO account AS a
USING (SELECT id, sum(amount) sum_amount FROM transaction
       GROUP BY id) AS t
ON a.id = t.id
WHEN MATCHED THEN
        UPDATE SET
              balance = a.balance + t.sum_amount
WHEN NOT MATCHED THEN
   INSERT
              (id, balance)
      VALUES (t.id, t.sum_amount)
```

예 4: transaction_log 테이블을 사용하여 트랜잭션 시간에 따라 달라지는 최신 transaction_log 행으로 전화 및 사무실을 갱신하고 행이 이미 존재하지 않는 최신의 새 employee_file 행을 삽입하며 employee_file에 행을 병합하십시오.

```
MERGE INTO employee_file AS e
USING (SELECT empid, phone, office
       FROM (SELECT empid, phone, office,
             ROW_NUMBER() OVER (PARTITION BY empid
             ORDER BY transaction_time DESC) rn
             FROM transaction_log) AS nt
             WHERE rn = 1) AS t
ON e.empid = t.empid
WHEN MATCHED THEN
        UPDATE SET
              (phone, office) =
              (t.phone, t.office)
WHEN NOT MATCHED THEN
   INSERT
              (empid, phone, office)
      VALUES (t.empid, t.phone, t.office)
```

예 5: 사원 행의 동적으로 제공되는 값을 사용하여 데이터가 기존 사원에 해당하는 경우 마스터 사원 테이블을 갱신하거나 데이터가 신입 사원용인 경우 행을 삽입하십시오. 다음 예는 C 프로그램의 코드 조각입니다.

```
  hv1 =
  "MERGE INTO employee AS t
USING TABLE(VALUES(CAST (? AS CHAR(6)), CAST (? AS VARCHAR(12)),
                   CAST (? AS CHAR(1)), CAST (? AS VARCHAR(15)),
                   CAST (? AS SMALLINT), CAST (? AS INTEGER)))
                      s(empno, firstnme, midinit, lastname, edlevel, salary)
ON t.empno = s.empno
WHEN MATCHED THEN
        UPDATE SET
              salary = s.salary
WHEN NOT MATCHED THEN
   INSERT
```

```
                (empno, firstnme, midinit, lastname, edlevel, salary)
        VALUES (s.empno, s.firstnme, s.midinit, s.lastname, s.edlevel,
                s.salary)";
EXEC SQL PREPARE s1 FROM :hv1;
  EXEC SQL EXECUTE s1 USING '000420', 'SERGE', 'K', 'FIELDING', 18, 39580;
```

예 6: archive 테이블에서 그룹 A로 구성되는 활동 목록을 갱신하십시오. 이전의 모든 활동을 삭제하고 활동이 변경된 경우 archive 테이블에서 활동 정보(설명 및 날짜)를 갱신하십시오. 이번에 공개될 새 활동의 경우 아카이브에 삽입하십시오. 활동 날짜를 알 수 없는 경우 오류를 표시하십시오. archive 테이블의 활동 날짜가 지정되어야 합니다. 각 그룹에는 activity 테이블이 있습니다. 예를 들어, activities_groupA에는 구성된 모든 활동이 들어 있으며 archive 테이블에는 회사에서 다른 그룹으로 구성된 이번에 공개될 모든 활동이 들어 있습니다. archive 테이블에는 기본 키로 (그룹, 활동)이 있으며 날짜는 널(NULL)이 될 수 없습니다. 모든 activity 테이블에는 기본 키로 activity가 지정되어 있습니다. 아카이브의 last_modified 컬럼은 디폴트값으로 CURRENT TIMESTAMP를 사용하여 정의됩니다.

```
  MERGE INTO archive ar
  USING (SELECT activity, description, date, last_modified
         FROM activities_groupA) ac
  ON (ar.activity = ac.activity) AND ar.group = 'A'
  WHEN MATCHED AND ac.date IS NULL THEN
          SIGNAL SQLSTATE '70001'
        SET MESSAGE_TEXT =
                     ac.activity CONCAT ' cannot be modified. Reason: Date is not known'
WHEN MATCHED AND ac.date < CURRENT DATE THEN
     DELETE
  WHEN MATCHED AND ar.last_modified < ac.last_modified THEN
     UPDATE SET
                (description, date, last_modified) = (ac.description, ac.date, DEFAULT)
  WHEN NOT MATCHED AND ac.date IS NULL THEN
          SIGNAL SQLSTATE '70002'
        SET MESSAGE_TEXT =
                     ac.activity CONCAT ' cannot be inserted. Reason: Date is not known'
WHEN NOT MATCHED AND ac.date >= CURRENT DATE THEN
     INSERT
                (group, activity, description, date)
        VALUES ('A', ac.activity, ac.description, ac.date)
  ELSE IGNORE
```

OPEN

OPEN문은 커서를 열어 그 결과 테이블의 행을 페치하는 데 사용됩니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 명령행 처리기를 사용하여 호출 시, 일부 옵션을 지정할 수 없습니다. 자세한 정보는 『명령행 SQL문 및 XQuery문 사용』을 참조하십시오.

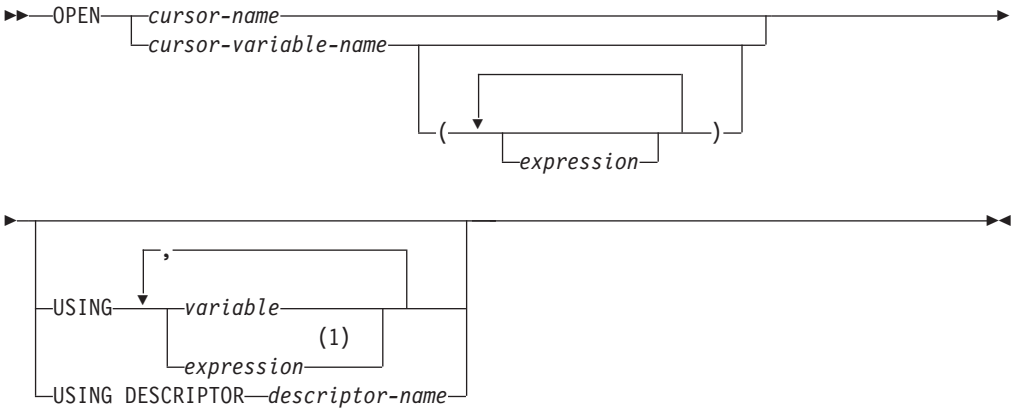
권한 부여

전역 변수가 참조되면, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 READ 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

이 명령문은 동적으로 준비될 수 없으므로 그룹 특권을 고려하지 않습니다.

구문



주:

1 변수가 아닌 표현식은 컴파일된 복합 명령문에서만 사용될 수 있습니다.

설명

cursor-name

프로그램에서 이미 명시된 DECLARE CURSOR문에 정의된 커서를 이름 지정합니다. OPEN문이 실행될 때 *cursor-name*로 식별되는 커서는 닫힌 상태여야 합니다.

DECLARE CURSOR문은 다음과 같은 방식으로 SELECT문을 식별해야 합니다.

- DECLARE CURSOR문에 SELECT문 포함
- 준비된 SELECT문을 이름 지정하는 *statement-name* 포함

커서의 결과 테이블은 SELECT문을 평가하여 파생됩니다. 평가에서는 SELECT문에 지정된 PREVIOUS VALUE 표현식이나 특수 레지스터, 전역 변수의 현재 값과, OPEN문의 USING절 또는 SELECT문에 지정된 호스트 변수의 현재 값을 사용합니다. 결과 테이블의 행은 OPEN문 실행 중에 도출되고, 임시 테이블은 그 행을 보유하기 위해 작성됩니다. 또는 후속 FETCH문 실행 중에 파생될 수도 있습니다. 어떤 경우이든, 커서는 열린 상태에 놓이며 결과 테이블의 첫 번째 행 앞에 위치합니다. 테이블이 빈 경우 커서 상태는 "마지막 행 뒤"에서 효력을 발생합니다.

cursor-variable-name

커서 변수의 이름을 지정합니다. 커서 변수의 값은 널(NULL)일 수 없습니다(SQLSTATE 34000). 커서 변수에 지정된 커서 값 컨스트럭터가 로컬 SQL 변수나 루틴 SQL 매개변수에 대한 참조를 포함하는 SELECT문을 지정하면, OPEN문은 SELECT문이 지정된 같은 범위에 있어야 합니다(SQLSTATE 51044). 커서 변수에 지정된 커서 값 컨스트럭터가 *statement-name*을 지정하면, OPEN문은 *statement-name*문이 명시적으로나 내재적으로 선언한 같은 범위에 있어야 합니다(SQLSTATE).

OPEN문이 실행될 때 커서 변수의 기본 커서는 닫힌 상태에 있어야 합니다. 기본 커서의 결과 테이블은 커서 변수와 연관된 SELECT문이나 동적 명령문을 평가하여 파생됩니다. 평가에서는 SELECT문에 지정된 PREVIOUS VALUE 표현식이나 특수 레지스터, 전역 변수의 현재 값과, OPEN문의 USING절 또는 SELECT문에 지정된 변수의 현재 값을 사용합니다. 결과 테이블의 행은 OPEN문 실행 중에 도출되고, 임시 테이블은 그 행을 보유하기 위해 작성됩니다. 또는 후속 FETCH문 실행 중에 파생될 수도 있습니다. 어떤 경우이든, 커서는 열린 상태에 놓이며 결과 테이블의 첫 번째 행 앞에 위치합니다. 테이블이 빈 경우 커서 상태는 "마지막 행 뒤"에서 효력을 발생합니다.

*cursor-variable-name*을 사용한 OPEN문은 복합 SQL(컴파일된) 명령문 내에서만 사용될 수 있습니다.

(expression, ...)

매개변수화된 커서 변수의 이름 지정된 매개변수와 연관된 인수를 지정합니다. 커서 변수에 지정된 *cursor-value-constructor*는 지정된 인수와 매개변수 수가 같은 매개변수 목록을 포함해야 합니다(SQLSTATE 07006 또는 07004). *n*번째 표현식의 데이터 유형과 값은 *n*번째 매개변수에 지정 가능해야 합니다(SQLSTATE 07006 또는 22018).

USING

매개변수 표시문자 대신에 사용되는 값 또는 커서의 명령문의 변수를 소개합니다. 매개변수 표시문자의 설명은 『PREPARE』를 참조하십시오.

*statement-name*이 DECLARE CURSOR문에 지정되거나 매개변수 표시문자를 포함한 커서 변수와 연관된 커서 값 컨스트럭터에서 지정되면, USING이 사용되어야 합니다. 준비된 명령문에 매개변수 표시문자가 포함되지 않는 경우 USING이 무시됩니다.

*select-statement*가 DECLARE CURSOR문에 지정되거나 커서 변수와 연관된 매개변수로 사용 불가능한 커서 값 컨스트럭터에 지정되면, USING을 사용하여 변수 값을 겹쳐쓸 수 있습니다.

variable

변수 및 호스트 변수 선언 규칙에 따라 프로그램에 선언되어 있는 변수나 호스트 변수를 식별합니다. 변수의 수는 준비된 명령문의 매개변수 표시문자 수와 같아야 합니다. *n*번째 변수는 준비된 명령문의 *n*번째 매개변수에 일치합니다. 로케이터 변수와 파일 참조 변수는 적절히 매개변수 표시문자에 대한 값의 소스로 제공될 수 있습니다.

expression

표현식을 사용하여 값이 매개변수 표시문자와 연관되도록 지정합니다. USING절에서 표현식을 지정하는 OPEN문은 복합 SQL(컴파일된)문 내에서만 사용될 수 있습니다(SQLSTATE 42601). 표현식의 수는 준비된 명령문의 매개변수 표시문자 수와 같아야 합니다(SQLSTATE 07001). *n*번째 표현식은 준비된 명령문의 *n*번째 매개변수 표시문자에 해당합니다. n번째 표현식의 데이터 유형과 값은 n번째 매개변수 표시문자와 연관된 유형에 지정 가능해야 합니다(SQLSTATE 07006).

규칙

- 커서의 SELECT문이 평가될 때 명령문의 각 매개변수 표시문자는 해당 호스트 변수로 대체됩니다. 입력된 매개변수 표시문자의 경우 목표 변수 속성은 CAST 스펙으로 지정된 속성입니다. 미입력 매개변수 표시문자의 경우 목표 변수 속성은 매개변수 표시문자의 컨텍스트에 따라 결정됩니다.
- 매개변수 표시문자 P에 해당하는 호스트 변수로 V를 지정하도록 하십시오. V값은 컬럼에 값을 지정하는 규칙에 따라 P에 대한 목표 변수에 할당됩니다. 따라서 다음과 같습니다.
  - V는 목표와 호환성이 있어야 합니다.
  - V가 문자열인 경우 그 길이(긴 문자열이 아닌 문자열의 뒤 공백을 제외한)는 목표 속성 길이보다 작아야 합니다.
  - V가 숫자인 경우 정수 부분의 절대값은 목표의 정수 부분의 최대 절대값 길이보다 작아야 합니다.
  - V 속성이 목표 속성과 같지 않은 경우 그 값은 목표 속성에 일치하도록 변환됩니다.

커서의 SELECT문이 평가될 때 P 대신 사용되는 값은 P에 대한 목표 변수의 값입니다. 예를 들어, V가 CHAR(6)이고 목표가 CHAR(8)인 경우 P 대신에 사용된 값은 두 개의 공백이 채워진 V의 값입니다.

- USING절은 매개변수 표시문자가 들어 있는 준비된 SELECT문을 위한 것입니다. 그러나 커서의 SELECT문이 DECLARE CURSOR문 또는 커서 변수와 연관된 매개변수로 사용 불가능한 커서 값 컨스트럭터의 일부일 때에도 사용됩니다. 이런 경우 OPEN문은 목표 변수의 속성이 SELECT문의 호스트 변수 속성과 같을 때를 제외하고, SELECT문의 각 호스트 변수가 매개변수 표시문자인 것처럼 실행됩니다. 그 결과 커서의 SELECT문에 있는 호스트 변수 값은 USING절에 지정된 호스트변수 값으로 대체됩니다.SELECT문이 다른 변수를 포함하지 않으므로 변수 값 겹쳐쓰기는 매개변수화된 커서 변수를 열 때 사용해서는 안됩니다.
- 커서 정의에서 임베드된 SQL 데이터를 수정하는 루틴 및 SQL 데이터 변경문에 대한 실행이 완료되면 커서 열기 시 결과 세트가 임시 테이블에 저장됩니다. 명령문 실행이 완료되면 SQLERRD(3) 필드에 삽입, 갱신 및 삭제 조작이 가능한 행 수의 합이 포함됩니다. 데이터 변경문을 fullselect에 포함하는 커서를 사용하는 OPEN문을 실행하는 동안 오류가 발생하면 데이터 변경문의 결과는 롤백됩니다.

  OPEN문의 명시적 롤백 또는 OPEN문 이전의 세이브포인트까지 ROLLBACK 시 커서는 닫힙니다. 커서 정의가 fullselect의 FROM절에 데이터 변경문을 포함하는 경우 데이터 변경문의 결과는 롤백됩니다.

  SELECT문 또는 SELECT INTO문에 중첩된 데이터 변경문이 목표로 하는 테이블의 행에 대한 변경은 커서 열기 후 프로세스되고 해당 커서에 대한 페치(fetch) 조작 시 오류가 발생하는 경우에도 완료됩니다.

주

- ***커서의 닫힌 상태:*** 프로그램이 시작될 때와 프로그램이 ROLLBACK문을 시작할 때 프로그램의 모든 커서는 닫힌 상태입니다.

  WITH HOLD로 선언된 열린 커서를 제외한 모든 커서는 프로그램이 COMMIT문을 발행할 때 닫힌 상태에 있습니다.

  CLOSE문이 실행되었거나 커서 위치를 예측 불가능하게 만드는 오류가 발견되어 커서도 닫힌 상태에 있을 수 있습니다.

  커서 변수가 범위 밖에 있고 기본 커서를 참조하는 다른 커서 변수가 없으면, 커서 변수의 기본 커서가 닫힙니다.
- 커서의 결과 테이블에서 행을 검색하려면 커서가 열릴 때 FETCH문을 실행하십시오. 닫힌 상태에서 열린 상태로 커서 상태를 변경하는 유일한 방법은 OPEN문을 실행하는 것입니다.

- *임시 테이블의 효과:* 일부 경우에는 커서의 결과 테이블이 FETCH문 실행 동안 파생됩니다. 임시 테이블 방법이 대신 사용되는 경우도 있습니다. 이러한 방법으로 OPEN문 실행 중에 전체 결과 테이블이 임시 테이블로 전송됩니다. 임시 테이블이 사용될 때 프로그램의 결과는 다음과 같은 방법에 따라 다를 수 있습니다.
  - FETCH문이 나타날 때까지 OPEN일 때 오류가 발생할 수 있습니다.
  - 커서가 열린 동안 같은 트랜잭션에서 실행되는 INSERT, UPDATE, DELETE 문은 결과 테이블에 영향을 줄 수 없습니다.
  - SELECT문의 NEXT VALUE 표현식은 OPEN 중에 결과 테이블의 모든 행에 대해 평가됩니다.

  반대로 임시 테이블이 사용되지 않으면 커서가 열려 있는 동안 실행된 INSERT, UPDATE 및 DELETE문은 동일한 작업 단위에서 발생한 경우 결과 테이블에 영향을 미칠 수 있으며, SELECT문의 NEXT VALUE 표현식은 각 행이 페치(fetch)되는 것으로 평가됩니다. 이 결과 테이블은 동일한 작업 단위(UOW)에 의해 영향받을 수도 있지만, 이러한 조작의 결과는 항상 예측 가능한 것은 아닙니다. 예를 들어, 커서 C가 SELECT * FROM T로 정의된 결과 테이블 행에 있고 새로운 행이 T에 삽입된 경우 행이 순서대로 지정되어 있지 않으므로 결과 테이블에 대한 삽입 효과는 예측할 수 없습니다. 따라서 후속 FETCH C는 새 T행을 검색할 수도 있고 하지 않을 수도 있습니다.
- 명령문 캐시는 OPEN문에 의해 열린 것으로 선언된 커서에 영향을 줍니다.

예:

예 1: COBOL 프로그램으로 다음과 같은 역할을 하는 임베드 명령문을 작성하십시오.

1. (ADMRDEPT) 부서 'A00'에 의해 관리되는 부서에 대한 DEPARTMENT 테이블로부터 모든 행을 검색하는 데 사용할 커서 C1을 정의하십시오.
2. 페치(fetch)될 첫 번째 행 앞에 커서 C1을 두십시오.

```
EXEC SQL  DECLARE C1 CURSOR FOR
                SELECT DEPTNO, DEPTNAME, MGRNO
                  FROM DEPARTMENT
                  WHERE ADMRDEPT = 'A00'
END-EXEC.


   EXEC SQL  OPEN C1
END-EXEC.
```

예 2: 커서 DYN_CURSOR를 C 프로그램에서 동적으로 정의된 선택문과 연관시키기 위해 OPEN문을 코드화하십시오. 두 개의 매개변수 표시문자가 Select문의 술어 부분에 사용되었다고 가정하면 OPEN문과 함께 제공되는 두 개의 호스트 변수가 정수와 varchar(64) 값을 응용프로그램과 데이터베이스 사이에 전달합니다. (관련 호스트 변수 정의, PREPARE문 및 DECLARE CURSOR문은 또한 아래의 예에 나타나 있습니다.)

```
   EXEC SQL  BEGIN DECLARE SECTION;
  static short    hv_int;
  char            hv_vchar64[65];
  char            stmt1_str[200];
   EXEC SQL  END DECLARE SECTION;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

예 3: 예 2에서와 같이 OPEN문을 코드화하되 WHERE절에 있는 매개변수 표시문자의 수와 데이터 유형이 확인되지 않은 경우입니다.

```
   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;
```

예 4: 다음을 수행하는 프로시저를 작성합니다.

1. 출력 커서 변수에 대한 커서 지정
2. 커서 열기

```
CREATE PROCEDURE PROC1 (OUT P1 CURSOR)LANGUAGE SQL
BEGIN
SET P1 = CURSOR FOR SELECT DEPTNO, DEPTNAME, MGRNO FROM DEPARTMENT WHERE ADMRDEPT = 'A00'; --
OPEN P1; --
END;
```

PREPARE

SQL문이 실행되도록 동적으로 준비하기 위해 응용프로그램에 의해 PREPARE문이 사용됩니다. PREPARE문은 *명령문 문자열*이라는 문자열 형태의 명령문에서 *준비된 명령문*이라는 실행 가능한 SQL문을 작성합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

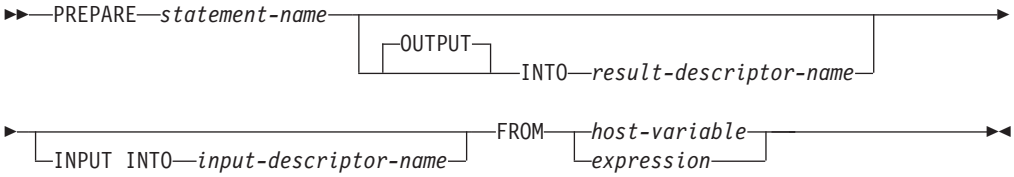
명령문 준비 시간에 권한 부여 점검이 수행되는 명령문(DML)의 경우, 명령문의 권한 부여 ID에서 가지는 특권에는 PREPARE문에 의해 지정된 SQL문을 실행하는 데 필요한 특권이 포함되어야 합니다. 명령문의 권한 부여 ID는 바인드 옵션 DYNAMICRULES의 영향을 받을 수 있습니다.

명령문 실행시 권한 부여 점검이 수행되는 명령문(DDL, GRANT 및 REVOKE문)의 경우, 명령문을 사용하는 데 필요한 권한 부여는 없으나 준비된 명령문이 실행될 때 권한 부여가 점검됩니다.

보안 규정으로 보호되는 테이블을 포함하는 명령문의 경우, 보안 규정과 연관된 규칙은 명령문 실행시 항상 평가됩니다.

명령문의 권한 부여 ID가 EXPLAIN, SQLADM 또는 DBADM 권한을 포함한 경우, 사용자는 명령문을 준비할 수 있습니다. 그러나 명령문을 실행할 수 있는 능력은 명령문 실행 시 다시 점검됩니다.

구문



설명

statement-name

준비된 명령문의 이름을 지정합니다. 이름이 기존의 준비된 명령문을 식별하는 경우 그 이전에 준비된 명령문은 무효화됩니다. 이 설명은 열린 커서의 SELECT문인 준비된 명령문을 식별해서는 안됩니다.

OUTPUT INTO

OUTPUT INTO가 사용되어 PREPARE문이 실행된 경우 준비된 명령문에 있는 출력 매개변수 표시 문자에 대한 정보가 *result-descriptor-name*에 의해 지정된 SQLDA에 놓입니다.

result-descriptor-name

SQLDA 이름을 지정합니다. (DESCRIBE문이 이 절에 대신 사용될 수도 있습니다.)

INPUT INTO

INPUT INTO가 사용되어 PREPARE문이 실행된 경우 준비된 명령문에 있는 입력 매개변수 표시 문자에 대한 정보가 *input-descriptor-name*에 의해 지정된 SQLDA에 놓입니다. 입력 매개변수 표시문자는 사용과 상관없이 항상 널(NULL)을 입력할 수 있다고 간주됩니다.

input-descriptor-name

SQLDA 이름을 지정합니다. (DESCRIBE문이 이 절에 대신 사용될 수도 있습니다.)

FROM

명령문 문자열을 사용합니다. 이 명령문 문자열은 지정된 호스트 변수의 값입니다.

host-variable

문자열 변수 선언 규칙에 따라 프로그램에 기술된 호스트 변수를 지정합니다. 최대 명령문 길이 2 097 152바이트보다 작은 고정 길이 또는 가변 길이 문자열 변수여야 합니다. CLOB(2097152)는 최대 크기 명령문을 포함할 수 있으나 VARCHAR은 포함할 수 없습니다.

expression

명령문 문자열을 지정하는 표현식입니다. 표현식은 최대 명령문 길이 2 097 152바이트보다 작은 고정 길이 또는 가변 길이 문자열 유형을 리턴해야 합니다.

규칙

- ***명령문 문자열에 대한 규칙:*** 명령문 문자열은 동적으로 준비할 수 있는 실행 명령문이어야 합니다. 명령문 문자열은 다음 SQL문 중 하나여야 합니다.
  - ALTER
  - CALL
  - COMMENT
  - COMMIT
  - 복합 SQL(인라인된)
  - 복합 SQL(컴파일된)
  - CREATE
  - DECLARE GLOBAL TEMPORARY TABLE

- DELETE
- DROP
- EXPLAIN
- FLUSH EVENT MONITOR
- FLUSH PACKAGE CACHE
- GRANT
- INSERT
- LOCK TABLE
- MERGE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME
- REVOKE
- ROLLBACK
- SAVEPOINT
- SELECT
- SET COMPILATION ENVIRONMENT
- SET CURRENT DECFLOAT ROUNDING MODE
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT FEDERATED ASYNCHRONY
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT IMPLICIT XMLPARSE OPTION
- SET CURRENT ISOLATION
- SET CURRENT LOCALE LC_TIME
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT MDC ROLLOUT MODE
- SET CURRENT OPTIMIZATION PROFILE
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ROLE(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- SET ENCRYPTION PASSWORD

- SET EVENT MONITOR STATE(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- SET SESSION AUTHORIZATION
- SET 변수
- TRANSFER OWNERSHIP(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- TRUNCATE(DYNAMICRULES 실행 동작이 패키지에 영향을 주는 경우에만)
- UPDATE

주

- ***매개변수 표시문자:*** 명령문 문자열에 호스트 변수 참조는 포함될 수 없으나 *매개변수 표시문자*는 포함될 수 있습니다. 명령문 문자열은 준비된 명령문 실행시 호스트 변수의 값으로 대체할 수 있습니다. CALL문에서는 프로시저에 대한 OUT 및 INOUT 인수에 대해서도 매개변수 표시문자를 사용할 수 있습니다. CALL문이 실행된 후 인수에 대해 리턴된 값이 매개변수 표시문자에 해당하는 호스트 변수에 지정됩니다.

  매개변수 표시문자는 물음표(?)로서, 명령문 문자열이 고정 SQL문인 경우 호스트 변수가 사용 될 수 있는 곳에서 사용되는 또는 콜론 다음의 이름(:name). 매개변수 표시문자가 값으로 교체되는 방법에 대한 설명은 『OPEN』 및 『EXECUTE』를 참조하십시오.

  매개변수 표시문자 이름이 지정된 경우, 이름은 문자, 숫자 및 기호 @, #, $ 및 _을 포함할 수 있습니다. 이름은 대문자로 규정되지 않습니다.

  이름 지정된 매개변수 표시문자는 호스트 변수와 같은 구문을 갖지만 두 개는 상호 교환할 수 없습니다. 호스트 변수에는 값이 있으며 정적 SQL문에서 직접적으로 사용됩니다. 이름 지정된 매개변수 표시문자는 동적 SQL문의 값에 대한 위치 지정자로 명령문이 실행될 때 제공됩니다.

  매개변수 표시문자에는 두 가지 유형이 있습니다.

  입력된 매개변수 표시문자

  목표 데이터 유형과 함께 지정된 매개변수 표시문자로 다음과 같은 일반 형태를 지닙니다.

  ```
  CAST(? AS data-type)
  ```

이 표기는 함수 호출이 아니라 런타임 매개변수의 유형이 지정된 데이터 유형이거나 지정된 데이터 유형으로 변환할 수 있는 데이터 유형이라는 "약속"입니다. 예를 들어, 다음 표기의 경우

```
  UPDATE EMPLOYEE
SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
WHERE EMPNO = ?
```

TRANSLATE 함수의 인수 값은 런타임시 제공됩니다. 이 값의 데이터 유형은 VARCHAR(12)이거나 VARCHAR(12)로 변환될 수 있는 유형이 됩니다.

미입력 매개변수 표시문자

목표 데이터 유형없이 지정되는 매개변수 표시문자입니다. 작은따옴표 형태를 지닙니다. 미입력 매개변수 표시문자의 데이터 유형은 컨텍스트로 제공됩니다. 예를 들어, 위 Update문의 술어에 있는 미입력 매개변수 표시문자는 EMPNO 컬럼의 데이터 유형과 같습니다.

입력된 매개변수 표시문자는 호스트 변수가 지원되는 동적 SQL문에서 사용될 수 있고, 그 데이터 유형은 CAST 함수에서의 약속에 기초합니다.

매개변수 표시문자의 데이터 유형이 SQL문의 컨텍스트를 기반으로 파생될 수 있는 한 유형이 지정되지 않은 매개변수 표시문자는 동적 SQL문에서 사용될 수 있습니다 (SQLSTATE 42610).

첫 번째 컨텍스트에서는 *c1*이 문자열 데이터 유형으로 분석되나 두 번째 컨텍스트에서는 *c1*이 숫자 데이터 유형으로 분석되므로 다음 예에서 오류가 발생합니다.

```
SELECT 'Hello' || c1, 5 + c1 FROM (VALUES(?)) AS T(c1)
```

그러나 파생된 컬럼과 연관된 매개변수 표시문자 *c1*이 두 컨텍스트 모두에서 숫자 데이터 유형으로 분석되므로 다음 명령문은 성공적입니다.

```
SELECT 7 + c1, 5 + c1 FROM (VALUES(?)) AS T(c1)
```

유형이 지정되지 않은 매개변수 표시문자 입력 규칙은 『유형이 지정되지 않은 매개변수 표시문자 입력 규칙』을 참조하십시오.

- PREPARE문이 실행될 때 명령문 문자열이 구문 분석되고 오류가 검사됩니다. 명령문 문자열이 유효하지 않다면 오류 조건이 SQLCA에 보고됩니다. 오류가 정정되지 않는 한, 시스템에서 수행된 내재적인 준비로 인해 이 명령문을 참조하는 모든 후속 EXECUTE 또는 OPEN문 또한 동일한 오류를 수신합니다.
- 준비된 명령문은 다음과 같은 제약점을 가지고 다음 종류의 명령문에서 참조될 수 있습니다.

In... **준비된 명령문**

DESCRIBE

모든 명령문이 가능함

DECLARE CURSOR

SELECT여야 함

EXECUTE

SELECT여서는 *안 됨*

- 준비된 명령문은 여러 번 실행될 수 있습니다. 준비된 명령문이 한 번 이상 실행되지 않고 매개변수 표시문자가 포함되지 않는 경우 PREPARE문과 EXECUTE문을 사용하지 않고 EXECUTE IMMEDIATE문을 사용하는 것이 더 효과적입니다.
- 명령문 캐싱은 반복되는 준비에 영향을 미칩니다.

예:

예 1: COBOL 프로그램에서 비Select문을 준비하여 실행하십시오. 명령문이 호스트 변수 HOLDER에 포함되어 있고, 사용자의 지시에 따라 프로그램이 명령문 문자열을 그 호스트 변수에 위치시킨다고 가정하십시오. 준비될 명령문은 매개변수 표시문자를 가지지 않습니다.

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER
   END-EXEC.
EXEC SQL  EXECUTE STMT_NAME
   END-EXEC.
```

예 2: 예 1에서와 같이 비Select문을 준비하여 실행하되 C 프로그램용으로 코드하십시오. 준비될 명령문에는 매개변수 표시문자가 얼마든지 들어갈 수 있다고 가정하십시오.

```
EXEC SQL  PREPARE STMT_NAME FROM :holder;
EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

다음 명령문이 준비된다고 가정하십시오.

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

DEPT 테이블의 컬럼이 다음과 같이 정의되어 있습니다.

```
DEPT_NO   CHAR(3) NOT NULL, -- department number
DEPTNAME  VARCHAR(29), -- department name
MGRNO     CHAR(6), -- manager number
ADMRDEPT  CHAR(3)  -- admin department number
```

관리자가 없고 부서 AOO에 보고하며 부서 번호가 G01인 COMPLAINTS를 삽입하기 위해서는 EXECUTE문을 발행하기 전에 구조 INSERT_DA가 표 30의 값을 가져야 합니다.

표 30. INSERT_DA 구조의 필요값

SQLDA 필드	값
SQLDAID	SQLDA
SQLDABC	192(주 1 참조)
SQLN	4
SQLD	4

표 30. INSERT_DA 구조의 필요값 (계속)

SQLDA 필드	값
SQLTYPE SQLLEN SQLDATA SQLIND SQLNAME	452 3 G01에 대한 *포인터* (주 2 참조)
SQLTYPE SQLLEN SQLDATA SQLIND SQLNAME	449 29 COMPLAINTS에 대한 *포인터* 0에 대한 *포인터*
SQLTYPE SQLLEN SQLDATA SQLIND SQLNAME	453 6 (주 3 참조) *-1에 대한 포인터* -
SQLTYPE SQLLEN SQLDATA SQLIND SQLNAME	453 3 A00에 대한 *포인터* 0에 대한 *포인터*
주: 1. 이 값은 32비트 응용프로그램에서 PREPARE에 대해 수행된 값입니다. PREPARE가 64비트 응용프로그램에서 수행되면 SQLDABC는 값 240을 갖게 됩니다. 2. SQLTYPE이 널(NULL)을 허용하지 않는 데이터 유형을 식별하기 때문에 SQLIND의 SQLVAR에 대한 값이 무시됩니다. 3. SQLIND 값이 이 값이 널(NULL) 값임을 나타내기 때문에 SQLDATA의 SQLVAR에 대한 값이 무시됩니다.	

REFRESH TABLE

REFRESH TABLE문은 구체화된 쿼리 테이블의 데이터를 새로 고칩니다.

호출

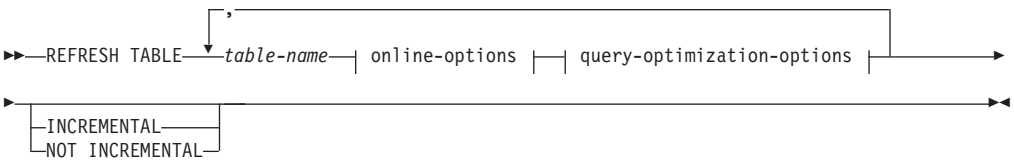
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

구문



online-options:

query-optimization-options:

설명

table-name

새로 고칠 테이블을 식별합니다.

내재적 또는 명시적 스키마를 포함한 이름은 현재 서버에 이미 존재하는 테이블을 식별해야 합니다. 테이블에서는 REFRESH TABLE문을 사용할 수 있어야 합니다 (SQLSTATE 42809). 여기에는 다음으로 정의된 구체화된 쿼리 테이블이 포함됩니다.

- REFRESH IMMEDIATE

- REFRESH DEFERRED

online-options

테이블 처리 중의 접근성을 지정합니다.

ALLOW NO ACCESS

언커미트된 분리 레벨이 사용되는 경우를 제외하고는 테이블 새로 고침 중에는 다른 사용자가 테이블에 액세스할 수 없도록 지정합니다.

ALLOW READ ACCESS

테이블 새로 고침 중에 다른 사용자가 테이블에 읽기 전용으로 액세스할 수 있도록 지정합니다.

ALLOW WRITE ACCESS

테이블 새로 고침 중에 다른 사용자가 테이블에 읽기 및 쓰기 액세스를 할 수 있도록 지정합니다.

ALLOW READ ACCESS 또는 ALLOW WRITE ACCESS 옵션을 사용할 때 잠금 시간종료로 인해 전체 명령문이 롤백되는 것을 방지하려면, REFRESH TABLE 명령문을 실행하기 전에 SET CURRENT LOCK TIMEOUT 명령문(WAIT 옵션 지정)을 발행한 후 특수 레지스터를 이전 값에 재설정할 것을 권장합니다. CURRENT LOCK TIMEOUT 레지스터는 모든 잠금 유형이 아니라 특정 잠금 유형 세트에만 영향을 준다는 점에 유의하십시오.

query-optimization-options

REFRESH DEFERRED 구체화된 쿼리 테이블의 새로 고침에 필요한 쿼리 최적화 옵션을 지정합니다.

ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY

CURRENT REFRESH AGE 특수 레지스터가 'ANY'로 설정되는 경우, *table-name*의 새로 고침은 REFRESH DEFERRED 구체화된 쿼리 테이블이 *table-name*을 새로 고치는 데 사용되는 쿼리를 최적화하는 데 쓰일 수 있도록 지정합니다. *table-name*이 REFRESH DEFERRED 구체화된 쿼리 테이블이 아니면 오류가 리턴됩니다(SQLSTATE 428FH). REFRESH IMMEDIATE 구체화된 쿼리 테이블은 쿼리 최적화시 항상 고려됩니다.

INCREMENTAL

하위 테이블에 델타 부분이 있으면 그 부분만 고려하거나 연관된 스테이징 테이블이 존재하고 테이블의 내용이 일치할 경우 연관된 스테이징 테이블의 내용만을 고려하여 테이블에 대한 증분 새로 고침을 지정합니다. 이같은 요청을 만족시키지 못할 경우(즉, 시스템이 구체화된 쿼리 테이블 정의를 전부 다시 계산해야 함을 발견한 경우) 오류가 발생합니다(SQLSTATE 55019).

NOT INCREMENTAL

구체화된 쿼리 테이블 정의를 다시 계산하여 테이블을 완전히 새로 고치도록 지정합니다.

INCREMENTAL과 NOT INCREMENTAL이 모두 지정되지 않은 경우 시스템은 증분 처리가 가능한지를 판별합니다. 가능하지 않으면 완전 새로 고침을 수행합니다. 새로 고칠 구체화된 쿼리 테이블에 대한 스테이징 테이블이 있는데 스테이징 테이블이 보류 상태에 있기 때문에 증분 처리를 할 수 없는 경우에는 오류가 발생합니다 (SQLSTATE 428A8). 스테이징 테이블이나 구체화된 쿼리 테이블이 불일치 상태에 있으면 완전 새로 고침이 수행되고 불일치 상태에 있지 않으면 스테이징 테이블의 내용이 증분 처리에 사용됩니다.

규칙

- 하나 이상의 별칭을 참조하는 구체화된 쿼리 테이블에 REFRESH TABLE을 발행할 경우, 명령문의 권한 부여 ID는 데이터 소스에 있는 테이블에서 선택할 수 있는 권한이 있어야 합니다(SQLSTATE 42501).

주

- 하위 테이블이 로드, 접속 또는 접속 해제된 REFRESH IMMEDIATE 구체화된 쿼리 테이블을 새로 고치기 위해 이 명령문을 사용하면, 시스템은 하위 테이블의 델타 부분을 사용하여 구체화된 쿼리 테이블에 대해 증분 새로 고침 수행을 선택할 수 있습니다. 지원 스테이징 테이블이 있는 REFRESH DEFERRED 구체화된 쿼리 테이블을 새로 고치기 위해 이 명령문을 사용하면, 시스템은 스테이징 테이블에 캡처된 하위 테이블의 델타 부분을 사용하여 구체화된 쿼리 테이블에 대해 증분 새로 고침 수행을 선택할 수 있습니다. 그러나 이 최적화가 가능하지 않으므로 데이터 무결성을 위해 완전 새로 고침(구체화된 쿼리 테이블 정의를 다시 계산)이 필요한 경우도 있습니다. 사용자는 INCREMENTAL 옵션을 지정하여 증분 유지보수를 명시적으로 요청할 수 있으며, 해당 최적화가 가능하지 않을 경우에는 시스템이 오류를 리턴합니다(SQLSTATE 55019).
- ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY 옵션이 사용되는 경우에는 REFRESH DEFERRED 구체화된 쿼리 테이블의 새로 고침 순서가 올바르도록 확실히 하십시오. 예를 들어 구체화된 쿼리가 동일한 하위 테이블을 공유하는 2개의 구체화된 쿼리 테이블 MQT1 및 MQT2가 있다고 생각하십시오. MQT2의 구체화된 쿼리는 하위 테이블 대신 MQT1을 사용해서 계산될 수 있습니다. 두 개의 구체화된 쿼리 테이블 MQT1 및 MQT2를 새로 고치기 위해 개별적인 명령문을 사용하고 MQT2를 먼저 새로 고친다면, 시스템은 MQT2를 새로 고치기 위해 아직 새로 고쳐지지 않은 MQT1의 내용을 사용하는 것을 선택할 수도 있습니다. 이 경우 두 개의 구체화된 쿼리 테이블이 거의 동시에 새로 고쳐졌지만 MQT1은 현재 데이터를 포함하는데 반해 MQT2

는 스테일 데이터를 포함할 수 있습니다. 한 개의 REFRESH문 대신 두 개가 쓰인 경우 올바른 새로 고침 순서는 MQT1을 먼저 새로 고치는 것입니다.

- 구체화된 쿼리 테이블에 연관된 스테이징 테이블이 있을 경우 새로 고침이 성공적으로 수행되면 스테이징 테이블이 프룬됩니다.

RELEASE(연결)

RELEASE(연결)문은 하나 이상의 연결을 연결 보류 상태로 둡니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

필요한 권한 없음

구문

주:

1 응용프로그램 서버(AS) CURRENT 또는 ALL은 호스트 변수나 분리 ID에 의해서만 식별됩니다.

설명

server-name 또는 *host-variable*

*server-name*이 들어 있는 *host-variable* 또는 지정된 *server-name*으로 응용프로그램 서버(AS)를 식별합니다.

*host-variable*을 지정할 경우 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안됩니다. *host-variable* 내에 들어 있는 *server-name*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안됩니다.

*server-name*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 로컬 디렉토리에 나열되어야 합니다.

지정된 데이터베이스 별명이나 호스트 변수에 포함된 데이터베이스 별명은 응용프로그램 프로세스의 기존 연결을 식별해야 합니다. 데이터베이스 별명이 기존 연결을 식별하지 않는 경우 오류(SQLSTATE 08003)가 발생합니다.

CURRENT

응용프로그램 프로세스의 현재 연결을 식별합니다. 응용프로그램 프로세스는 연결됨 상태여야 합니다. 그렇지 않으면, 오류(SQLSTATE 08003)가 발생합니다.

ALL 또는 ALL SQL

응용프로그램 프로세스의 모든 기존 연결을 식별합니다. 이 양식의 RELEASE문은 응용프로그램 프로세스의 모든 기존 연결을 릴리스 보류 상태에 둡니다. 따라서 모든 연결은 다음 커미트 조작 중에 파괴됩니다. 명령문이 실행될 때 어떤 연결도 없으면 오류 또는 경고가 발생하지 않습니다.

예:

예 1: 응용프로그램에서 더 이상 IBMSTHDB와의 SQL 연결이 필요하지 않습니다. 다음 명령문은 다음 커미트 조작 중에 연결이 파괴되도록 합니다.

```
EXEC SQL RELEASE IBMSTHDB;
```

예 2: 응용프로그램에서 더 이상 현재 연결이 필요하지 않습니다. 다음 명령문은 다음 커미트 조작 중에 연결이 파괴되도록 합니다.

```
EXEC SQL RELEASE CURRENT;
```

예 3: 응용프로그램이 커미트 후에 데이터베이스에 액세스하지 않아도 되지만 잠시 동안 계속 실행될 경우 해당 연결을 불필요하게 묶지 않는 것이 좋습니다. 다음 명령문은 커미트 시 모든 연결이 파괴되도록 커미트 이전에 실행할 수 있습니다.

```
EXEC SQL RELEASE ALL;
```

RELEASE SAVEPOINT

RELEASE SAVEPOINT문은 응용프로그램이 더 이상 이름 지정된 세이브포인트가 유지보수되기를 원하지 않음을 표시하는 데 사용됩니다. 이 명령문이 호출된 후 세이브포인트로의 롤백은 더 이상 가능하지 않습니다.

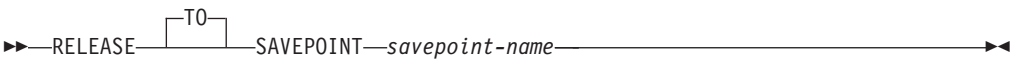
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

savepoint-name

해제될 세이브포인트를 지정합니다. 이름 지정된 세이브포인트에 중첩된 모든 세이브포인트도 해제됩니다. 해당 세이브포인트 또는 그 안에 중첩된 모든 세이브포인트로의 롤백은 이제 불가능합니다. 이름 지정된 세이브포인트가 현재 세이브포인트 레벨에 존재하지 않는 경우(SAVEPOINT문의 설명에 있는 『규칙』 절 참조) 오류가 리턴됩니다(SQLSTATE 3B001). 지정된 *savepoint-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939).

주

- 해제된 세이브포인트의 이름은 이제 UNIQUE 키워드가 동일한 세이브포인트 이름을 지정하는 이전 SAVEPOINT문에서 지정되었는지 여부와 상관없이 다른 SAVEPOINT문에서 다시 사용할 수 있습니다.

예 :

예 1: 세이브포인트 SAVEPOINT1을 해제하십시오.

```
RELEASE SAVEPOINT SAVEPOINT1
```

RENAME

RENAME문은 기존 테이블이나 인덱스의 이름을 바꿉니다.

호출

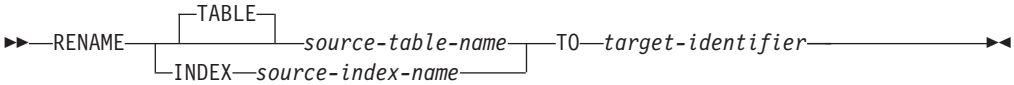
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블 또는 인덱스에서 CONTROL 특권
- 테이블에 대한 SYSCAT.TABLES 카탈로그 뷰와 인덱스에 대한 SYSCAT.INDEXES 카탈로그 뷰의 OWNER 컬럼에 기록된 테이블 또는 인덱스의 소유권
- 스키마에서 ALTERIN 특권
- DBADM 권한

구문



설명

TABLE *source-table-name*

이름이 변경될 기존 인덱스 이름을 지정하십시오. 스키마 이름을 포함한 이름은 데이터베이스에 이미 존재하는 테이블을 식별해야 합니다(SQLSTATE 42704). 이 이름은 카탈로그 테이블의 이름(SQLSTATE 42832), 구체화된 쿼리 테이블, 유형이 지정된 테이블(SQLSTATE 42997), 작성된 임시 테이블, 선언된 전역 임시 테이블(SQLSTATE 42995), 별칭 또는 테이블이나 별명이 아닌 오브젝트의 이름(SQLSTATE 42809)이어서는 안됩니다. TABLE 키워드는 선택적입니다.

INDEX *source-index-name*

이름이 변경될 기존 인덱스 이름을 지정하십시오. 스키마 이름을 포함한 이름은 데이터베이스에 이미 존재하는 인덱스를 식별해야 합니다(SQLSTATE 42704). 이 이름은 작성된 임시 테이블 또는 선언된 전역 임시 테이블에서의 인덱스 이름이어서는 안됩니다(SQLSTATE 42995). 스키마 이름은 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT여서는 안됩니다(SQLSTATE 42832).

target-identifier

스키마 이름이 없는 테이블 또는 인덱스에 대한 새 이름을 지정하십시오. 소스 오

브젝트의 스키마 이름은 오브젝트에 대한 새 이름을 규정하는 데 사용됩니다. 규정된 이름은 데이터베이스에 이미 존재하는 테이블, 뷰, 별칭 또는 인덱스를 식별하면 *안됩니다*(SQLSTATE 42710).

규칙

테이블 이름을 바꿀 경우, 소스 테이블은 다음과 같으면 안됩니다.

- 기존의 구체화된 쿼리 테이블 정의 내에서 참조될 수 없습니다.
- 기존 트리거의 주제 테이블일 수 없습니다.
- 참조 무결성 제한조건에서 상위 또는 종속 테이블일 수 없습니다.
- 기존의 참조된 컬럼 범위일 수 없습니다.
- 분석 가능한 XSR 오브젝트를 참조할 수 없습니다.

소스 테이블이 위 조건을 하나 이상 위반할 경우 오류가 발생합니다(SQLSTATE 42986).

인덱스 이름을 바꿀 경우 다음과 같은 제한사항이 있습니다.

- 유형이 지정된 테이블이 바탕을 두는 구현 테이블에 대한 소스 인덱스는 시스템에서 생성한 인덱스가 될 수 없습니다(SQLSTATE 42858).

주

- 새 테이블 또는 인덱스 이름을 반영하도록 카탈로그 항목이 갱신됩니다.
- 소스 테이블이나 인덱스의 이름과 연관되는 모든 권한 부여는 새 테이블이나 인덱스의 이름으로 *전송됩니다*. (권한 부여 카탈로그 테이블도 그에 맞게 갱신됩니다.)
- 소스 테이블에 대해 정의된 인덱스가 새로운 테이블로 *전송*됩니다. (인덱스 카탈로그 테이블도 그에 맞게 갱신됩니다.)
- RENAME TABLE은 소스 테이블에 종속된 모든 패키지를 무효화시킵니다. RENAME INDEX는 소스 인덱스에 종속된 모든 패키지를 무효화시킵니다.
- 별명이 *source-table-name*에 사용될 경우 이것은 테이블 이름으로 변형되어야 합니다. 테이블은 이 테이블 스키마 내에서 이름이 재지정됩니다. 별명은 RENAME문에 의해 변경되지 않으며 기존의 테이블 이름을 그대로 지칭합니다.
- 외부 키로 기본 키나 고유 제한조건이 참조되지 않을 경우 기본 키나 고유 제한조건을 갖는 테이블의 이름을 재지정할 수 있습니다.

예:

EMP 테이블의 이름을 EMPLOYEE로 변경하십시오.

```
RENAME TABLE EMP TO EMPLOYEE
RENAME TABLE ABC.EMP TO EMPLOYEE
```

인덱스 이름을 NEW-IND에서 IND로 변경하십시오.

```
RENAME INDEX NEW-IND TO IND
RENAME INDEX ABC.NEW-IND TO IND
```

RENAME TABLESPACE

RENAME TABLESPACE문은 기존 테이블 스페이스의 이름을 바꿉니다.

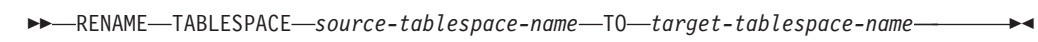
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 SYSCTRL 권한을 포함해야 합니다.

구문



설명

source-tablespace-name

이름이 바뀔 기존 테이블 스페이스를 한 부분으로 된 이름으로 지정합니다. 이것은 SQL ID(일반 또는 분리 ID)입니다. 테이블 스페이스 이름은 카탈로그에 이미 있는 테이블 스페이스를 식별해야 합니다(SQLSTATE 42704).

target-tablespace-name

테이블 스페이스에 대한 새 이름을 한 부분으로 된 이름으로 지정합니다. 이것은 SQL ID(일반 또는 분리 ID)입니다. 새 테이블 스페이스 이름은 카탈로그에 이미 있는 테이블 스페이스를 식별하지 *않아야* 합니다(SQLSTATE 42710). 이것은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

규칙

- SYSCATSPACE 테이블 스페이스는 이름을 바꿀 수 없습니다(SQLSTATE 42832).
- ″롤 포워드 보류″ 또는 ″롤 포워드 진행 중″ 상태를 갖는 모든 테이블 스페이스는 이름을 바꿀 수 없습니다(SQLSTATE 55039).

주

- 테이블 스페이스 이름을 바꾸면 테이블 스페이스의 최소 복구 시간이 이름 바꾸기가 발생하는 특정 시점으로 갱신됩니다. 이는 테이블 스페이스 레벨에서의 롤 포워드는 최소한 이 특정 시점에 있어야 함을 의미합니다.
- 백업 작성후 이름 바꾸기가 수행된 백업 이미지로부터 테이블 스페이스를 리스토어할 경우에는 새 테이블 스페이스 이름을 사용해야 합니다.

예 :

USERSPACE1 테이블 스페이스의 이름을 DATA2000으로 변경합니다.

```
RENAME TABLESPACE USERSPACE1 TO DATA2000
```

REPEAT

REPEAT문은 검색 조건이 참이 될 때까지 하나의 명령문이나 명령문 그룹을 실행합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

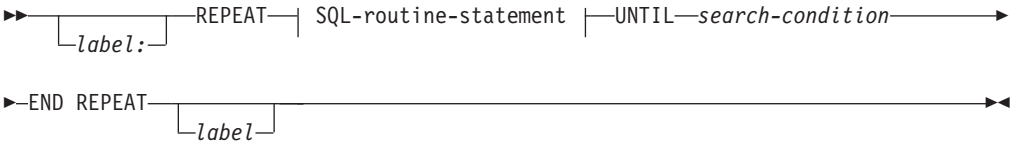
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

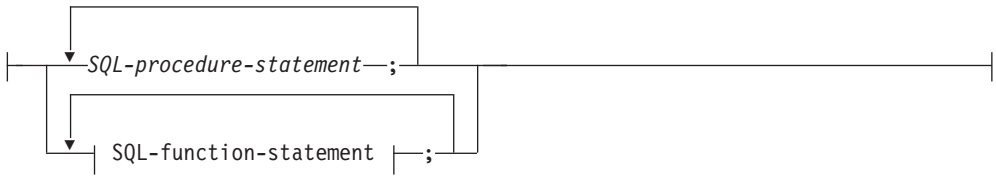
권한 부여

REPEAT문을 호출하기 위해 필요한 특권은 없습니다. 단, 명령문의 권한 부여 ID가 SQL문을 호출하고 REPEAT문에서 임베디드(embedded)된 조건을 검색하는 데 필요한 특권을 보유해야 합니다.

구문



SQL-routine-statement:



설명

label

REPEAT문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 해당 레이블이 LEAVE 및 ITERATE문에 지정될 수 있습니다. 종료 레이블이 지정된 경우, 이와 일치하는 시작 레이블도 지정되어야 합니다.

SQL-procedure-statement

루프에서 실행할 SQL문을 지정합니다.*SQL-procedure-statement*는 SQL 프로시저

의 컨텍스트에 있을 때또는 복합 SQL(컴파일된) 명령문 안에 있을 때만 적용할 수 있습니다. 『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

SQL-function-statement

루프에서 실행할 SQL문을 지정합니다.*SQL-function-statement*는 SQL 트리거, SQL 함수 또는 SQL 메소드의 컨텍스트에서만 적용할 수 있습니다. 『FOR』의 *SQL-function-statement*를 참조하십시오.

search-condition

*search-condition*은 각 REPEAT 루프를 실행한 후 평가됩니다. 조건이 참인 경우, 루프를 종료합니다. 조건이 알 수 없음 또는 거짓인 경우, 계속 루핑합니다.

예

REPEAT문은 *not_found* 조건 핸들러가 호출될 때까지 테이블에서 행을 페치합니다.

```
CREATE PROCEDURE REPEAT_STMT(OUT counter INTEGER)
  LANGUAGE SQL
  BEGIN
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE v_firstnme VARCHAR(12);
    DECLARE v_midinit CHAR(1);
    DECLARE v_lastname VARCHAR(15);
    DECLARE at_end SMALLINT DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE c1 CURSOR FOR
      SELECT firstnme, midinit, lastname
        FROM employee;
    DECLARE CONTINUE HANDLER FOR not_found
      SET at_end = 1;
    OPEN c1;
    fetch_loop:
    REPEAT
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      SET v_counter = v_counter + 1;
      UNTIL at_end > 0
    END REPEAT fetch_loop;
    SET counter = v_counter;
    CLOSE c1;
  END
```

RESIGNAL

RESIGNAL문은 핸들러를 활성화한 조건의 신호를 다시 보내거나 상위 레벨에서 처리될 수 있도록 대체 조건을 발생시키기 위해 조건 핸들러에서 사용됩니다. 예외, 경고 찾을 수 없음 조건이 선택적 메시지 텍스트와 함께 리턴되게 합니다.

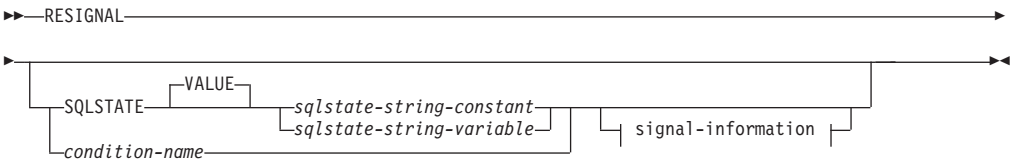
호출

이 명령문은 복합 SQL(컴파일된)문 내의 조건 핸들러에서만 임베디드될 수 있습니다. 복합 SQL(컴파일된)문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다.

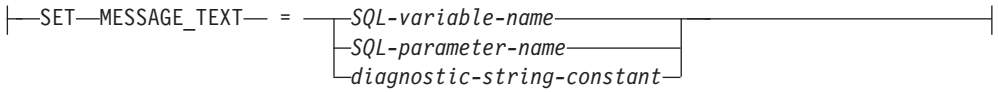
권한 부여

모듈 조건이 참조되는 경우, 명령문의 권한 부여 ID가 보유하는 특권은 모듈의 EXECUTE 특권을 포함해야 합니다.

구문



signal-information:



설명

SQLSTATE VALUE *sqlstate-string-constant*

지정된 문자열 상수는 SQLSTATE를 나타냅니다. 이는 SQLSTATE 규칙을 따르는 정확히 5자로 된 문자열 상수이어야 합니다.

- 각 문자는 숫자 세트('0' - '9')이거나 강조 표시가 없는 대문자('A' - 'Z')이어야 합니다.
- SQLSTATE 클래스(처음 두 문자)는 '00'(성공적인 완료)일 수 없습니다.

SQLSTATE가 이 규칙에 따르지 않으면 오류가 발생합니다(SQLSTATE 428B3).

SQLSTATE VALUE

리턴될 SQLSTATE를 지정합니다. 유효한 모든 SQLSTATE 값이 사용될 수 있습니다. 지정된 값은 SQLSTATE의 규칙을 따라야 합니다.

- 각 문자는 숫자 세트('0'- '9')이거나 발음 구별 부호가 없는 대문자('A'-'Z')여야 합니다.
- SQLSTATE 클래스(처음 두 문자)는 '00'(성공적인 완료)일 수 없습니다. 성공적인 완료를 표시하기 때문입니다.

SQLSTATE가 이 규칙에 따르지 않으면, 오류가 리턴됩니다.

sqlstate-string-constant

*sqlstate-string-constant*는 정확히 5자로 된 문자열 상수여야 합니다.

sqlstate-string-variable

지정된 SQL 변수 또는 SQL 매개변수는 데이터 유형 CHAR(5)이어야 합니다.

condition-name

리턴될 조건의 이름을 지정합니다. 복합 명령문에 *condition-name*을 선언해야 하며, 그렇지 않으면 현재 서버에 존재하는 조건을 식별하십시오.

SET MESSAGE_TEXT =

오류나 경고를 설명하는 문자열을 지정합니다. 이 문자열은 SQLCA의 `sqlerrmc` 필드에 리턴됩니다. 실제 문자열이 70바이트보다 큰 경우에는 경고없이 절단됩니다.

SQL-variable-name

메시지 텍스트가 있는 복합 명령문 내에서 선언되는 SQL 변수를 식별합니다.

SQL-parameter-name

메시지 텍스트가 있는 루틴으로 정의된 SQL 매개변수를 식별합니다. SQL 매개변수는 CHAR 또는 VARCHAR 데이터 유형으로 정의해야 합니다.

diagnostic-string-constant

메시지 텍스트를 포함한 문자열 상수를 지정합니다.

주

- RESIGNAL문이 SQLSTATE절 또는 *condition-name*을 지정하지 않고 발행된 경우, 핸들러가 호출된 동일한 조건으로 리턴됩니다. 조건과 연관된 SQLSTATE, SQLCODE 및 SQLCA는 변경되지 않습니다.
- RESIGNAL문이 연관된 SQLSTATE 값이 없는 *condition-name*을 사용하여 발행되고 조건이 처리되지 않으면, SQLSTATE 45000이 리턴되며 SQLCODE가 -438로 설정됩니다. 이러한 조건은 RESIGNAL문을 발행하는 루틴의 범위 내에 있는 SQLSTATE 45000에 대한 조건 핸들러에 의해 처리되지 않음에 유의하십시오.
- SQLSTATE 값 또는 연관된 SQLSTATE 값이 있는 *condition-name*을 사용하여 RESIGNAL문을 발행한 경우, 리턴된 SQLCODE는 다음과 같은 SQLSTATE 값을 기반으로 합니다.
  - 지정된 SQLSTATE 클래스가 '01'이나 '02' 중 하나가 아니면 경고나 찾을 수 없음 조건이 리턴되고 SQLCODE가 +438로 설정됩니다.

  - 그렇지 않으면 예외 조건이 리턴되고 SQLCODE가 -438로 설정됩니다.
- RESIGNAL문은 SQLCA의 표시된 필드를 다음과 같이 설정합니다.
  - sqlerrd 필드를 0으로 설정
  - sqlwarn 필드를 공백으로 설정
  - sqlerrmc를 MESSAGE_TEXT의 처음 70바이트로 설정
  - sqlerrml이 sqlerrmc의 길이로 설정되거나 SET MESSAGE_TEXT절이 지정되지 않은 경우에는 0으로 설정
  - sqlerrp를 ROUTINE으로 설정
- SQLSTATE 값에 대한 자세한 내용은 ″SIGNAL문″의 ″주″를 참조하십시오.

예 :

이 예에서는 0으로 나눔 오류를 발견합니다. IF문은 SIGNAL문을 사용하여 *overflow* 조건 핸들러를 호출합니다. 조건 핸들러는 RESIGNAL문을 사용하여 다른 SQLSTATE 값을 클라이언트 응용프로그램으로 리턴합니다.

```
CREATE PROCEDURE divide ( IN numerator INTEGER,
                          IN denominator INTEGER,
                          OUT result INTEGER)
LANGUAGE SQL
BEGIN
  DECLARE overflow CONDITION FOR SQLSTATE '22003';
  DECLARE CONTINUE HANDLER FOR overflow
    RESIGNAL SQLSTATE '22375';
  IF denominator = 0 THEN
    SIGNAL overflow;
  ELSE
    SET result = numerator / denominator;
  END IF;
END
```

RETURN

RETURN문은 루틴에서 리턴하는 데 사용됩니다. SQL 함수 또는 메소드의 경우 함수 또는 메소드 결과를 리턴합니다. SQL 프로시저의 경우 선택적으로 정수 상태 값을 리턴합니다.

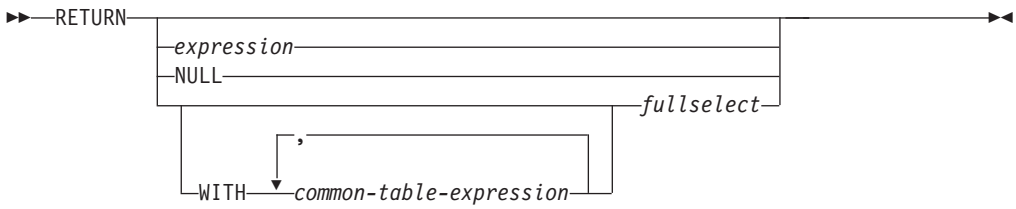
호출

이 명령문은 SQL 함수, SQL 메소드 또는 SQL 프로시저에 임베디드(embedded)할 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

권한 부여

RETURN문을 호출하기 위해 필요한 특권은 없습니다. 그러나 명령문의 권한 부여 ID가 RETURN문에 임베디드(embedded)되는 표현식 또는 fullselect를 호출하는 데 필요한 특권을 가지고 있어야 합니다.

구문



설명

expression

루틴에서 리턴되는 값을 지정합니다.

- 루틴이 함수이거나 메소드인 경우 *expression* 중 하나, NULL 또는 *fullselect*를 지정해야 하고(SQLSTATE 42631) 결과의 데이터 유형은 루틴의 RETURNS 유형에 지정할 수 있어야 합니다(SQLSTATE 42866).
- 루틴이 테이블 함수인 경우 스칼라 표현식(스칼라 fullselect가 아닌)을 지정할 수 없습니다(SQLSTATE 428F1).
- 루틴이 프로시저인 경우 *expression*의 데이터 유형은 INTEGER여야 합니다(SQLSTATE 428F2). 프로시저는 NULL 또는 *fullselect*를 리턴할 수 없습니다.

NULL

함수 또는 메소드가 RETURNS절에서 정의되는 데이터 유형의 널(NULL) 값을 리턴하도록 지정합니다. 프로시저로부터의 RETURN에 대해 NULL을 지정할 수 없습니다.

WITH *common-table-expression*

뒤에 오는 *fullselect*와 함께 사용할 공통 테이블 표현식을 정의합니다.

fullselect

함수에 대해 리턴될 행을 지정합니다. *fullselect*의 컬럼 수는 함수 결과의 컬럼 수와 일치해야 합니다(SQLSTATE 42811). 또한 *fullselect*의 정적 컬럼 유형은 컬럼에 대한 지정 규칙을 사용하여, 함수 결과의 선언된 컬럼 유형에 대해 지정 가능해야 합니다(SQLSTATE 42866).

*fullselect*는 프로시저로부터의 RETURN에 대해 지정할 수 없습니다.

루틴이 스칼라 함수 또는 메소드인 경우 *fullselect*는 컬럼 하나와(SQLSTATE 42823) 많아야 하나의 행을 리턴해야 합니다(SQLSTATE 21000).

루틴이 행 함수인 경우 많아야 하나의 행을 리턴해야 합니다(SQLSTATE 21505). 그러나 하나 이상의 컬럼이 리턴될 수 있습니다.

루틴이 테이블 함수인 경우 하나 이상의 컬럼을 갖는 0개 이상의 행을 리턴할 수 있습니다.

규칙

- SQL 함수 또는 메소드의 실행은 RETURN문으로 종료해야 합니다(SQLSTATE 42632).
- 복합 SQL(인라인된)문을 사용하는 SQL 테이블이나 행 함수에서 복합 명령문 마지막에서만 RETURN문을 사용할 수 있습니다. (SQLSTATE 429BD).
- SQL 프로시저에서 RETURN문은 조건 핸들러의 본문에서 허용되지 않습니다 (SQLSTATE 42601).

주

- 값이 프로시저에서 리턴될 때 호출자는 다음과 같이 값에 액세스할 수 있습니다.
  - SQL 프로시저가 다른 SQL 프로시저에서 호출될 때 GET DIAGNOSTICS문을 사용하여 DB2_RETURN_STATUS 검색
  - CLI 응용프로그램에서 escape절 CALL 구문(?=CALL...)에서 리턴 값 매개변수 마커에 대해 바운드된 매개변수 사용
  - SQL 프로시저의 CALL을 처리한 후 SQLCA의 `sqlerrd[0]` 필드에서 직접. 이 필드는 SQLCODE가 0 또는 양수인 경우에만 유효합니다(그 외에는 값 -1을 가정함).

예:

성공하는 경우 0, 실패하는 경우 -200의 상태 값으로 SQL 프로시저에서 리턴하려면 RETURN문을 사용하십시오.

```
BEGIN
...
  GOTO FAIL
...
  SUCCESS: RETURN 0
  FAIL: RETURN -200
END
```

REVOKE(데이터베이스 권한)

이러한 형태의 REVOKE문은 전체 데이터베이스에 적용되는 권한을 취소합니다.

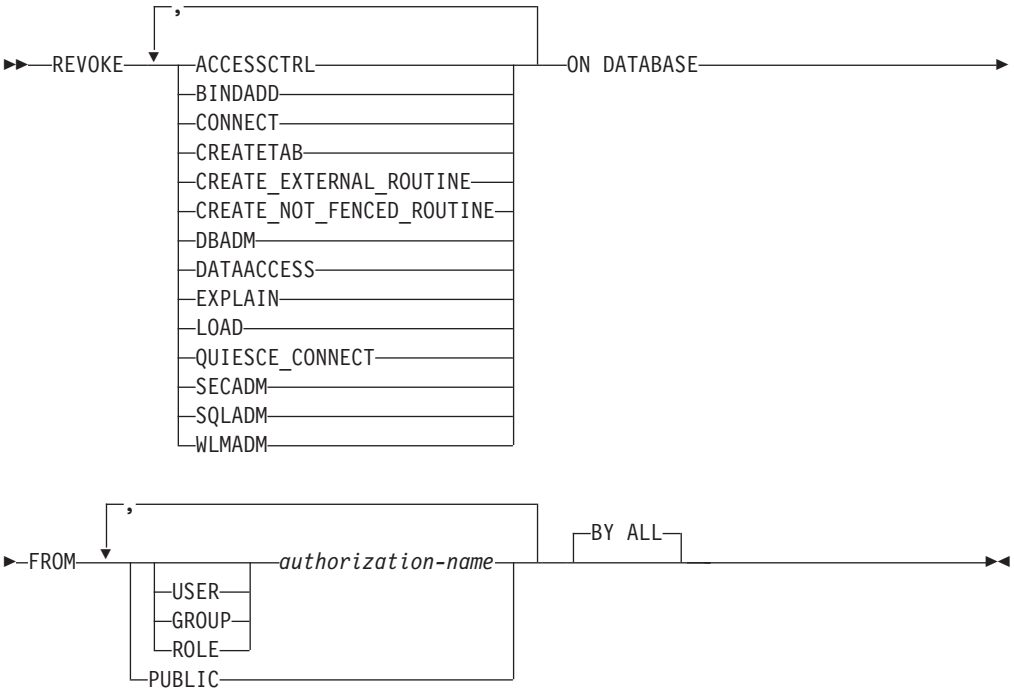
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

ACCESSCTRL, DATAACCESS, DBADM 또는 SECADM 권한을 취소하려면 SECADM이 필요합니다. 다른 권한을 취소하려면, ACCESSCTRL 또는 SECADM 권한이 필요합니다.

구문



설명

ACCESSCTRL

대부분의 데이터베이스 권한 및 오브젝트 특권을 부여하고 취소할 권한을 취소합니다.

BINDADD

패키지 작성 권한을 취소합니다. 패키지 작성자는 자동으로 해당 패키지에 대한 CONTROL 특권을 가지며 나중에 BINDADD 권한이 취소되어도 이 특권을 보유합니다.

BINDADD 권한은 DBADM 권한을 취소해야 DBADM 권한을 보유하는 *authorization-name*으로부터 취소할 수 있습니다.

CONNECT

데이터베이스에 액세스할 권한을 취소합니다.

사용자로부터 CONNECT 권한을 취소해도 데이터베이스의 오브젝트에 있는 사용자에게 권한 부여된 특권에는 영향을 미치지 않습니다. 사용자가 나중에 다시 CONNECT 권한을 받으면 이전에 보유했던 모든 특권이 여전히 유효하게 됩니다. (즉, 권한이 명시적으로 권한 취소되지 않은 것으로 간주됩니다.)

CONNECT 권한은 DBADM 권한을 취소해야 DBADM 권한을 보유하는 *authorization-name*으로부터 취소할 수 있습니다.

CREATETAB

테이블 작성 권한을 취소합니다. 테이블 작성자는 자동으로 그 테이블에 대한 CONTROL 특권을 갖게 되며, 이 특권은 나중에 작성자의 CREATETAB 권한이 취소되어도 그대로 남아 있습니다.

CREATETAB 권한은 DBADM 권한을 취소해야 DBADM 권한을 보유하는 *authorization-name*으로부터 취소할 수 있습니다(SQLSTATE 42504).

CREATE_EXTERNAL_ROUTINE

외부 루틴을 등록하기 위한 권한을 취소합니다. 일단 외부 루틴이 등록되면 CREATE_EXTERNAL_ROUTINE이 루틴을 등록한 권한 부여 ID로부터 권한 취소된 이후라도 종료될 때까지 계속됩니다.

CREATE_EXTERNAL_ROUTINE 권한은 DBADM 또는 CREATE_NOT_FENCED_ROUTINE 권한을 취소해야 DBADM 또는 CREATE_NOT_FENCED_ROUTINE 권한을 보유하는 *authorization-name*으로부터 권한 취소할 수 있습니다(SQLSTATE 42504).

CREATE_NOT_FENCED_ROUTINE

데이터베이스 관리 프로그램 프로세스에서 실행되는 레지스터 루틴에 권한을 취소합니다. 일단 루틴이 분리(fenced)되지 않은 것으로 등록되면 CREATE_NOT_FENCED_ROUTINE이 루틴을 등록한 권한 부여 ID로부터 권한이 취소된 이후라도 이러한 방식으로 계속 실행됩니다.

CREATE_NOT_FENCED_ROUTINE 권한은 DBADM 권한을 취소해야 DBADM 권한을 보유하는 *authorization-name*으로부터 취소할 수 있습니다.

DATAACCESS
데이터에 액세스할 권한을 취소합니다.

DBADM
DBADM 권한을 취소합니다.

DBADM 권한은 PUBLIC으로 부여할 수 없으므로 PUBLIC으로부터 취소될 수 없습니다.

주의:
DBADM 권한을 취소하면 데이터베이스의 오브젝트에서 *authorization-name*에서 보유한 특권이 자동으로 취소되지 않습니다.

EXPLAIN
데이터에 액세스하지 않아도 정적 및 동적 명령문을 Explain, 준비 및 설명할 권한을 취소합니다.

LOAD
이 데이터베이스에서 로드할 권한을 취소합니다.

QUIESCE_CONNECT
Quiesce 상태에서 데이터베이스에 액세스할 권한을 취소합니다.

SECADM
데이터베이스 보안을 관리하는 권한을 취소합니다.

SQLADM
SQL문을 모니터하고 조정하는 권한을 취소합니다.

WLMADM
워크로드 관리 프로그램 오브젝트를 관리하는 권한을 취소합니다.

FROM
권한을 취소할 사용자를 지정합니다.

USER
*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP
*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE
*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...
여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC
PUBLIC에서 권한을 취소합니다.

BY ALL
누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여받은 지정한 모든 사용자에게서 지정한 각 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

보안 관리자 필수: 데이터베이스에는 SECADM 권한이 있는 권한 부여 ID가 적어도 하나 있어야 합니다. SECADM 권한은 모든 사용자 권한 부여 ID에서 취소될 수 없습니다(SQLSTATE 42523).

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.DBAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다 (SQLSTATE 56092).

주

- 특정의 특권을 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. 기타 특권을 PUBLIC, 그룹 또는 역할에서 보유하거나 또는 사용자가 DBADM과 같은 상위 레벨의 권한을 보유하면 사용자는 태스크를 처리할 수 있습니다.
- *호환성*:
  - 이전 버전 DB2와의 호환성을 위해,
    - CREATE_NOT_FENCED_ROUTINE 대신 CREATE_NOT_FENCED를 지정할 수 있습니다.
  - z/OS용 DB2와의 호환성을 위해,
    - DATABASE 대신 SYSTEM을 지정할 수 있습니다.
    - NOT INCLUDING DEPENDANT PRIVILEGES가 대체 구문으로 지정될 수 있습니다.

예:

예 1: USER6이 유일한 사용자이고 그룹이 아닌 상태에서 사용자 USER6으로부터 테이블을 작성하는 특권을 취소하십시오.

```
REVOKE CREATETAB ON DATABASE FROM USER6
```

예 2: 데이터베이스에 대한 BINDADD 권한을 그룹 D024에 부여합니다. SYSCAT.DBAUTH 카탈로그 뷰에는 권한 받은 사용자에 대한 두 개의 행이 있습니다. 하나는 U의 GRANTEETYPE이고 다른 하나는 G의 GRANTEETYPE입니다.

```
REVOKE BINDADD ON DATABASE FROM GROUP D024
```

이 경우 GROUP 키워드를 지정해야 합니다. 그렇지 않으면 오류가 발생합니다 (SQLSTATE 56092).

예 3: 사용자 Walid의 보안 관리자 권한을 취소하십시오.

```
REVOKE SECADM ON DATABASE FROM USER Walid
```

REVOKE(면제)

이러한 형태의 REVOKE문은 레이블 기반 액세스 제어(LBAC) 액세스 규칙에 대한 면제를 취소합니다.

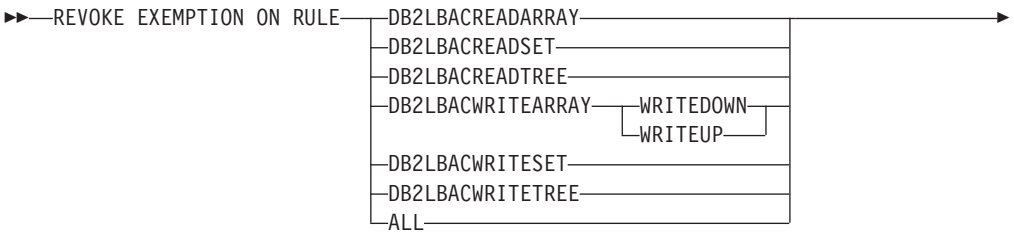
호출

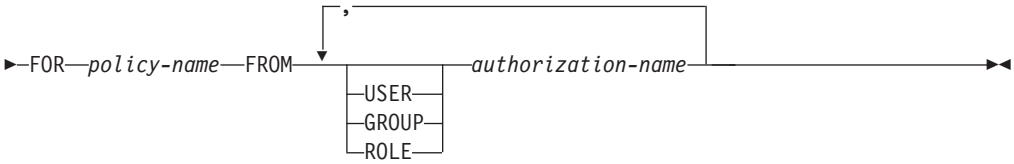
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문





설명

EXEMPTION ON RULE
액세스 규칙에 대한 면제를 취소합니다.

DB2LBACREADARRAY
사전 정의된 DB2LBACREADARRAY 규칙에 대한 면제를 취소합니다.

DB2LBACREADSET
사전 정의된 DB2LBACREADSET 규칙에 대한 면제를 취소합니다.

DB2LBACREADTREE
사전 정의된 DB2LBACREADTREE 규칙에 대한 면제를 취소합니다.

DB2LBACWRITEARRAY
사전 정의된 DB2LBACWRITEARRAY 규칙에 대한 면제를 취소합니다.

WRITEDOWN
면제가 아래로 쓰기에만 적용되도록 지정합니다.

WRITEUP
면제가 위로 쓰기에만 적용되도록 지정합니다.

DB2LBACWRITESET
사전 정의된 DB2LBACWRITESET 규칙에 대한 면제를 취소합니다.

DB2LBACWRITETREE
사전 정의된 DB2LBACWRITETREE 규칙에 대한 면제를 취소합니다.

ALL
사전 정의된 모든 규칙에 대한 면제를 취소합니다.

FOR *policy-name*
면제를 취소할 보안 규정의 이름을 지정합니다.

FROM
면제를 취소할 사용자를 지정합니다.

USER
*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP
*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE
*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...
하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

규칙

- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.SECURITYPOLICYEXEMPTIONS 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해 다음과 같습니다.
    - 모든 행이 'U' GRANTEETYPE을 가진 경우, USER로 가정됩니다.
    - 모든 행이 'G' GRANTEETYPE을 가진 경우, GROUP으로 가정됩니다.
    - 모든 행이 'R' GRANTEETYPE을 가진 경우, ROLE로 가정됩니다.
    - 모든 행이 GRANTEETYPE에 동일한 값을 갖지 않으면, 오류가 리턴됩니다 (SQLSTATE 56092).

예:

예 1: 사용자 WALID로부터 보안 규정 DATA_ACCESS에 대한 액세스 규칙 DB2LBACREADSET의 면제를 취소합니다.

```
REVOKE EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS
  FROM USER WALID
```

예 2: 사용자 BOBBY로부터 보안 규정 DATA_ACCESS에 대한 WRITEDOWN 옵션을 사용하여 액세스 규칙 DB2LBACWRITEARRAY에 대한 면제를 취소합니다.

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
  FOR DATA_ACCESS FROM USER BOBBY
```

예 3 사용자 BOBBY로부터 보안 규정 DATA_ACCESS에 대한 WRITEUP 옵션을 사용하여 액세스 규칙 DB2LBACWRITEARRAY에 대한 면제를 취소합니다.

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
  FOR DATA_ACCESS FROM USER BOBBY
```

REVOKE(전역 변수 특권)

이 양식의 REVOKE문은 작성된 전역 변수에 대한 하나 이상의 특권을 취소합니다.

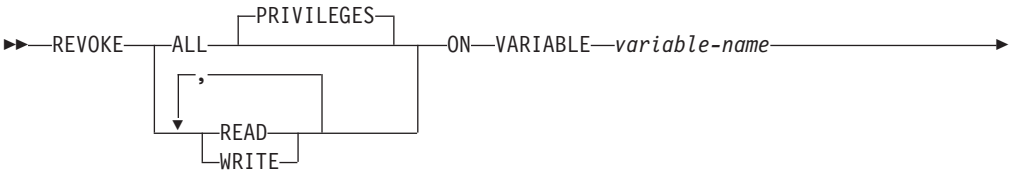
호출

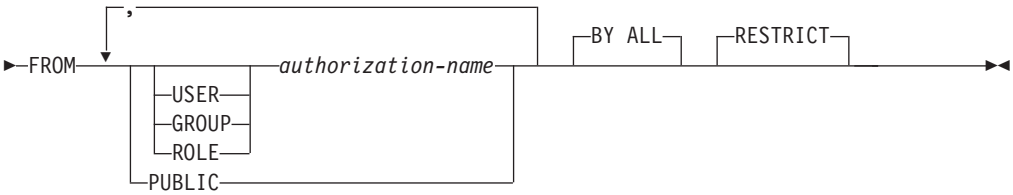
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문





설명

ALL PRIVILEGES

지정된 글로벌 변수에 대해 *authorization-name*이 보유하는 모든 특권을 취소합니다. ALL이 지정되지 않으면, READ 또는 WRITE를 지정해야 합니다. READ 또는 WRITE는 두 번 이상 지정할 수 없습니다.

READ

지정된 전역 변수의 값을 읽을 수 있는 특권을 취소합니다.

WRITE

지정된 전역 변수에 값을 지정할 수 있는 특권을 취소합니다.

ON VARIABLE *variable-name*

하나 이상의 특권이 취소될 전역 변수를 식별합니다. *variable-name*은 현재 서버에 있는 전역 변수를 식별해야 하며 모듈 변수가 아닙니다(SQLSTATE 42704).

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

ROLE

*authorization-name*이 현재 서버에서 기존 역할을 식별함을 지정합니다 (SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다. 권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다 (SQLSTATE 42502).

PUBLIC

PUBLIC에서 지정된 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 해당 특권을 명시적으로 부여 받은 이름 지정된 모든 사용자에게서 지정된 각 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

RESTRICT

오브젝트가 취소되는 특권의 영향을 받는 경우 명령문이 실패함을 지정합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 *authorization-name*마다, USER, GROUP 또는 ROLE 키워드 중 어느 것도 지정하지 않은 경우 권한 받은 사용자가 *authorization-name*인 SYSCAT.VARIABLEAUTH 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해,
  - GRANTEETYPE이 'U'이면 USER가 가정됩니다.
  - GRANTEETYPE이 'G'이면 GROUP이 가정됩니다.
  - GRANTEETYPE이 'R'이면 ROLE이 가정됩니다.
  - GRANTEETYPE이 동일한 값을 가지고 있지 않으면 오류가 리턴됩니다 (SQLSTATE 56092).
- SQL 함수, SQL 메소드, 프로시저, 뷰, 트리거 또는 다른 전역 변수가 전역 변수를 포함하고 취소되는 특권에 영향을 받는 경우 권한 취소 조작은 실패합니다 (SQLSTATE 42893).

주

- 전역 변수에 대한 READ 특권이 취소되면 전역 변수 값 쓰기에 대해 종속성을 가지고 있는(예를 들어, SET문에 의해) 패키지는 영향을 받지 않습니다. 전역 변수에 쓰는 것은 전역 변수에 대한 WRITE 특권으로 제어되기 때문입니다.
- 전역 변수에 대한 WRITE 특권이 취소되면 전역 변수 값 읽기에 대해 종속성을 가지고 있는 패키지는 영향을 받지 않습니다. 전역 변수에서 읽는 것은 전역 변수에 대한 READ 특권으로 제어되기 때문입니다.
- 특권을 취소한다고 하여 조치를 수행할 능력이 반드시 손상되는 것은 아닙니다. 사용자는 다른 그룹 또는 역할에서 멤버쉽을 통해 또는 PUBLIC으로 필수 특권이 보유되는 경우 계속 진행할 수 있습니다.

예 :

사용자 ZUBIRI에서 전역 변수 MYSCHEMA.MYJOB_PRINTER에 대한 WRITE 특권을 취소하십시오.

```
REVOKE WRITE ON VARIABLE MYSCHEMA.MYJOB_PRINTER FROM ZUBIRI
```

REVOKE(인덱스 권한)

이러한 형태의 REVOKE문은 인덱스에 대한 CONTROL 특권을 취소합니다.

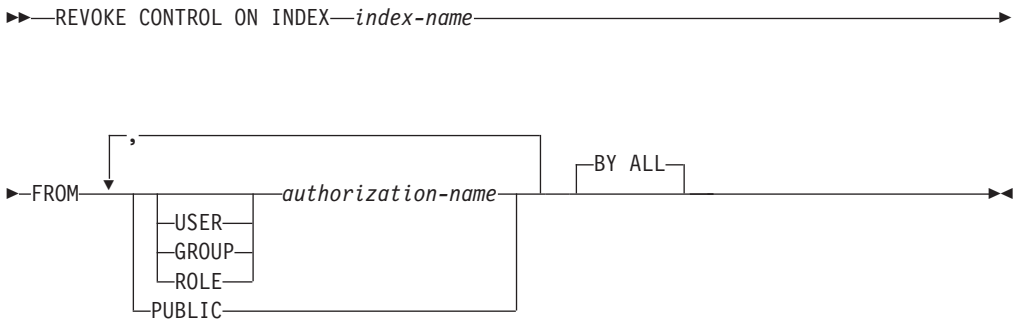
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문



설명

CONTROL

인덱스 삭제 특권을 취소합니다. 이는 인덱스에 대한 CONTROL 특권으로서 인덱스 작성자에게 자동으로 권한 부여됩니다.

ON INDEX *index-name*

CONTROL 특권이 권한 취소될 인덱스 이름을 지정합니다.

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여 받은 지정한 모든 사용자에게서 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.INDEXAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다 (SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. 기타 특권을 PUBLIC, 그룹 또는 역할에서 보유하거나 또는 인덱스의 스키마에서 사용자가 ALTERIN과 같은 권한을 보유하면 사용자는 태스크를 처리할 수 있습니다.

예:

예 1: USER4가 유일한 사용자이고 그룹이 아닌 상태에서 사용자 USER4로부터 인덱스 DEPTIDX를 삭제할 수 있는 특권을 취소하십시오.

```
REVOKE CONTROL ON INDEX DEPTIDX FROM KIESLER
```

예 2: 사용자 CHEF와 그룹 WAITERS로부터 인덱스 LUNCHITEMS를 삭제할 수 있는 특권을 취소하십시오.

```
REVOKE CONTROL ON INDEX LUNCHITEMS
  FROM USER CHEF, GROUP WAITERS
```

REVOKE(모듈 특권)

이 형태의 REVOKE문은 모듈에 대한 USAGE 특권을 취소합니다.

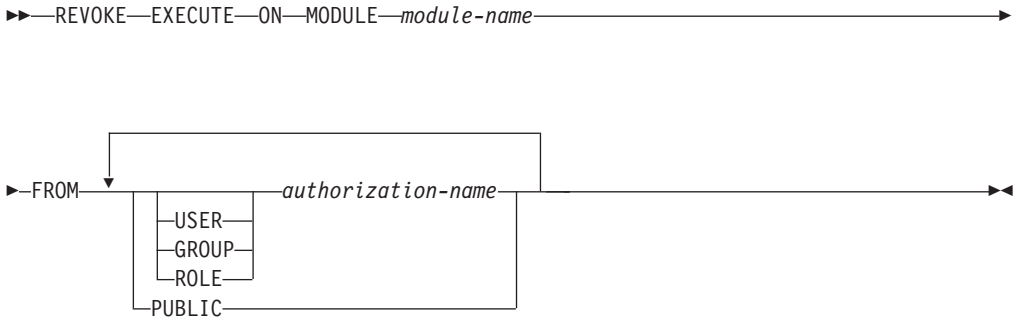
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문



설명

EXECUTE

발행된 모듈 오브젝트를 참조하는 특권을 취소합니다. 여기에는 다음 특권을 취소하는 것을 포함합니다.

- 모듈에 정의된 발행 루틴을 실행하십시오.
- 모듈에서 정의된 발행 전역 변수를 읽고 여기에 작성하십시오.
- 모듈에 정의된 발행 사용자 정의 유형을 참조하십시오.
- 모듈에 정의된 발행 조건을 참조하십시오.

ON MODULE *module-name*

특권이 취소되는 모듈을 식별합니다. *module-name*은 현재 서버에 존재하는 모듈을 식별해야 합니다 (SQLSTATE 42704).

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정하십시오. 역할 이름이 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name

하나 이상의 권한 부여 ID를 나열합니다. 동일한 *authorization-name*은 한 번 이상 지정될 수 없습니다.

PUBLIC

사용자 세트에 특권을 부여합니다(권한 부여 ID). 자세한 정보는 『권한 부여, 특권 및 오브젝트 소유권』을 참조하십시오.

예 :

다음 예는 사용자 *jones*의 *myModa*로 이름 지정된 모듈에서 EXECUTE 특권을 취소하는 방법을 설명합니다.

```
REVOKE EXECUTE ON MODULE MYMODA FROM JONES
```

REVOKE(패키지 특권)

이러한 형태의 REVOKE문은 패키지에 대한 CONTROL, BIND, EXECUTE 특권을 취소합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

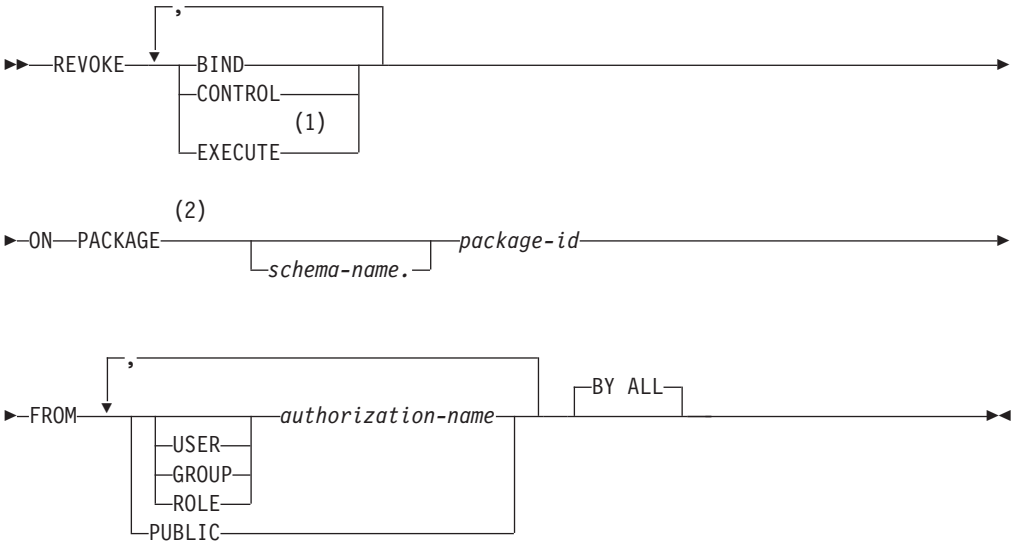
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 참조된 패키지에 대한 CONTROL 특권
- ACCESSCTRL 또는 SECADM 권한

ACCESSCTRL 또는 SECADM 권한은 CONTROL 특권을 취소하는 데 필요합니다.

구문



주:

1 RUN은 EXECUTE에 대한 동의어로 사용할 수 있습니다.

2 PACKAGE에 대한 동의어로 PROGRAM을 사용할 수 있습니다.

설명

BIND

참조된 패키지에서 BIND 또는 REBIND를 실행할 수 있는(또는, 참조된 패키지의 새 버전을 추가하는) 특권을 취소합니다.

BIND 특권은 CONTROL 특권을 취소하지 않고도 패키지에 CONTROL 특권을 보유하는 *authorization-name*으로부터 취소될 수 없습니다.

CONTROL

패키지 삭제 특권을 취소하고 패키지 특권을 다른 사용자에게 부여합니다.

CONTROL을 취소해도 다른 패키지 특권은 취소되지 않습니다.

EXECUTE

패키지 실행 특권을 취소합니다.

EXECUTE 특권은 CONTROL 특권을 취소해야 패키지상에 CONTROL 특권을 보유하는 *authorization-name*으로 취소할 수 있습니다.

ON PACKAGE *schema-name.package-id*

특권이 권한 취소될 패키지의 이름을 지정합니다. 스키마 이름이 지정되지 않은 경우 디폴트 스키마가 패키지 ID를 내재적으로 규정합니다. 패키지 특권의 권한 취소는 패키지의 모든 버전에 적용됩니다.

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여받은 지정한 모든 사용자에게서 지정한 각 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.PACKAGEAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다 (SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. 기타 특권을 PUBLIC, 그룹 또는 역할에서 보유하거나 또는 패키지의 스키마에서 사용자가 ALTERIN과 같은 권한을 보유하면 사용자는 태스크를 처리할 수 있습니다.

예:

예 1: 패키지 CORPDATA.PKGA에 대한 EXECUTE 특권을 PUBLIC으로부터 취소하십시오.

```
REVOKE EXECUTE
  ON PACKAGE CORPDATA.PKGA
  FROM PUBLIC
```

예 2: 사용자 FRANK와 PUBLIC에 대해 RRSP_PKG 패키지의 CONTROL 권한을 취소하십시오.

```
REVOKE CONTROL
  ON PACKAGE RRSP_PKG
  FROM USER FRANK, PUBLIC
```

REVOKE(역할)

이 양식의 REVOKE문은 사용자, 그룹 또는 다른 역할에서 역할을 취소합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

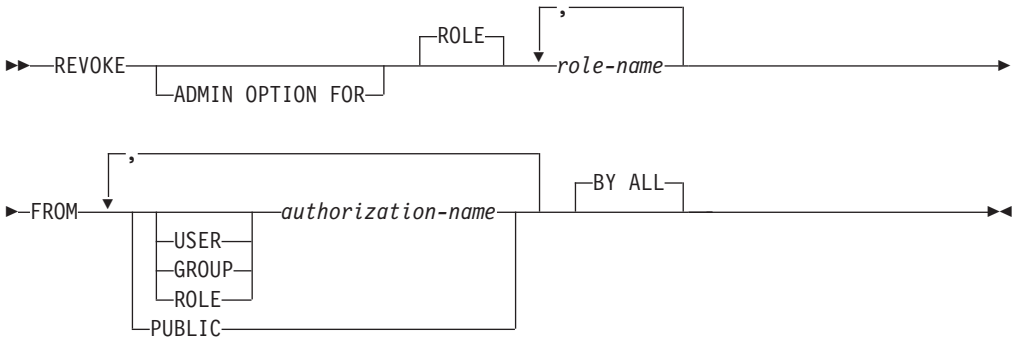
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 역할에 대한 WITH ADMIN OPTION
- SECADM 권한

*authorization-name*에서 ADMIN OPTION FOR *role-name*을 취소하거나 해당 역할에 대해 WITH ADMIN OPTION을 가지고 있는 *authorization-name*에서 *role-name*을 취소하려면 SECADM 권한이 필요합니다.

구문



설명

ADMIN OPTION FOR

*role-name*에 대한 WITH ADMIN OPTION을 취소합니다. *role-name*에 대한 WITH ADMIN OPTION은 *authorization-name*이나 PUBLIC(PUBLIC이 지정된 경우)에서 보유해야 합니다(SQLSTATE 42504). ADMIN OPTION FOR 절이 지정된 경우 역할 자체가 아니라 ROLE *role-name*에 대한 WITH ADMIN OPTION이 취소됩니다.

ROLE *role-name*

취소될 역할을 지정합니다. *role-name*은 현재 서버에서 *authorization-name* 또는 PUBLIC(PUBLIC이 지정된 경우)에 권한이 부여된 기존 역할을 식별해야 합니다(SQLSTATE 42504).

FROM

역할을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

ROLE

*authorization-name*이 현재 서버에서 기존 역할을 식별함을 지정합니다 (SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다. 권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다 (SQLSTATE 42502).

PUBLIC

PUBLIC에서 지정된 역할을 취소합니다.

BY ALL

권한을 부여한 사람에 관계없이 해당 역할이 명시적으로 부여된, 지정된 각 *authorization-name*에서 *role-name*을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 *authorization-name*마다, USER, GROUP 또는 ROLE 키워드 중 어느 것도 지정하지 않은 경우 권한 받은 사용자가 *authorization-name*인 SYSCAT.ROLEAUTH 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해,
  - GRANTEETYPE이 'U'이면 USER가 가정됩니다.
  - GRANTEETYPE이 'G'이면 GROUP이 가정됩니다.
  - GRANTEETYPE이 'R'이면 ROLE이 가정됩니다.
  - GRANTEETYPE이 동일한 값을 가지고 있지 않으면 오류가 리턴됩니다 (SQLSTATE 56092).
- 역할이 루틴에 대해 EXECUTE 특권을 가지고 있거나 시퀀스에 대해 USAGE 특권을 가지고 있고 패키지가 아닌 다른 SQL 오브젝트가 루틴 또는 시퀀스에 종속되는 경우 *role-name*은 역할 또는 *role-name*을 포함하는 역할을 식별하지 않아야 합니다(SQLSTATE 42893). SQL 오브젝트의 소유자는 *authorization-name*이거나 *authorization-name*의 구성원인 사용자입니다. 여기서 *authorization-name*은 역할입니다.

주

- *authorization-name* 또는 PUBLIC에서 역할이 취소되는 경우 역할이 보유하는 모든 특권은 더 이상 *authorization-name*이나 해당 역할을 통해 PUBLIC에 사용할 수 없습니다.
- 역할을 취소한다고 하여 해당 역할에 부여된 특권으로 특정 조치를 수행할 능력이 반드시 취소되는 것은 아닙니다. 사용자는 PUBLIC, 사용자가 속하는 그룹, 사용자에게 권한이 부여된 다른 역할이 다른 특권을 보유하거나 사용자에게 DBADM과 같은 상위 레벨의 권한이 있는 경우 계속 진행할 수 있습니다.

예:

예 1: 역할 DOCTOR에서 역할 INTERN을 취소하고 역할 SPECIALIS에서 역할 DOCTOR를 취소하십시오.

```
REVOKE ROLE INTERN FROM ROLE DOCTOR

REVOKE ROLE DOCTOR FROM ROLE SPECIALIST
```

예 2: PUBLIC에서 역할 INTERN을 취소하십시오.

```
REVOKE ROLE INTERN FROM PUBLIC
```

예 3: 사용자 BOB 및 그룹 TORONTO에서 역할 SPECIALIST를 취소하십시오.

```
REVOKE ROLE SPECIALIST FROM USER BOB, GROUP TORONTO BY ALL
```

REVOKE(루틴 특권)

이러한 양식의 REVOKE문은 모듈에 정의되지 않은 루틴(함수, 메소드 또는 프로시저)에 대한 특권을 취소합니다.

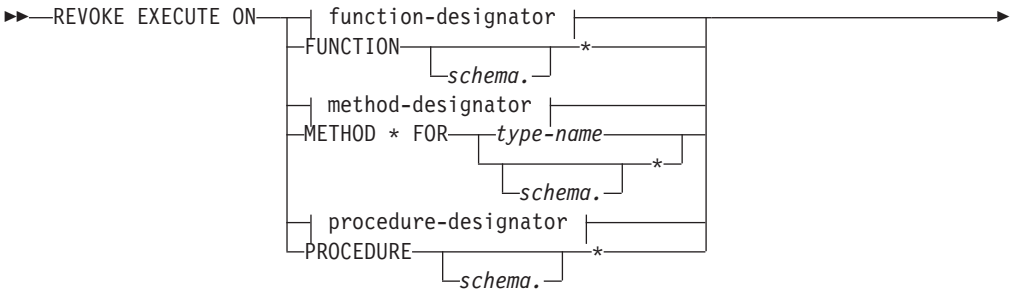
호출

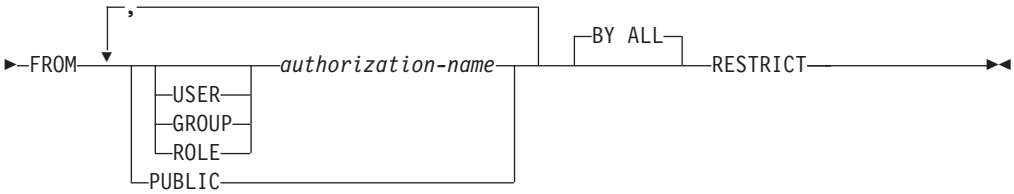
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

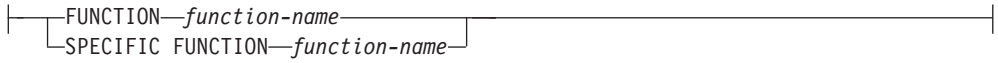
명령문의 권한 부여 ID가 보유한 특권은 ACCESSCTRL 또는 SECADM 권한을 포함해야 합니다.

구문



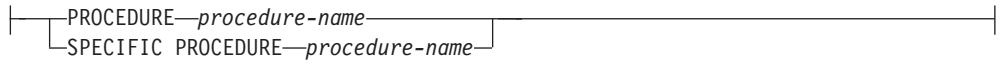


function-designator:



method-designator:

procedure-designator:



설명

EXECUTE

식별된 사용자 정의 함수(UDF), 메소드 또는 프로시저를 실행할 수 있는 특권을 취소합니다.

function-designator

특권의 권한이 취소된 함수를 고유하게 식별합니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

FUNCTION *schema.**

스키마의 모든 기존 함수와 앞으로 작성될 함수에 대한 명시적 권한 부여를 식별하십시오. *schema*.* 특권을 취소하더라도 특정 함수에 부여된 특권은 취소되지 않습니다. 동적 SQL문에서 스키마를 지정하지 않으면 CURRENT SCHEMA 특수 레지스터의 스키마가 사용됩니다. 정적 SQL문에서 스키마를 지정하지 않으면 QUALIFIER 프리컴파일/바인드 옵션의 스키마가 사용됩니다.

method-designator

특권의 권한이 취소된 메소드를 고유하게 식별합니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

METHOD *

type-name 유형에 대한 모든 기존의 메소드와 앞으로 작성될 메소드에 대한 명시적 권한 부여를 식별합니다. * 특권을 취소하더라도 특정 메소드에 부여된 특권은 취소되지 않습니다.

FOR *type-name*

지정한 메소드가 있는 유형의 이름을 지정합니다. 이 이름은 카탈로그에 기술되어 있는 유형을 식별해야 합니다(SQLSTATE 42704). 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터의 값은 규정되지 않은 유형 이름을 위한 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일/바인드 옵션은 내재적으로 규정되지 않은 유형 이름의 규정자를 지정합니다.*type-name* 대신 별표(*)를 사용하여 스키마에 있는 모든 기존 유형과 앞으로 작성될 유형에 대한 모든 기존 메소드 및 추후 메소드에 대하여 명시적 권한 부여를 식별할 수 있습니다. 메소드와 *type-name*에 대해 별표를 사용하여 특권을 취소하더라도 특정 메소드나 특정 유형의 모든 메소드에 부여된 특권은 취소되지 않습니다.

procedure-designator

특권의 권한이 취소된 프로시저를 고유하게 식별합니다. 자세한 정보는 18 페이지의 『함수, 메소드 및 프로시저 지정자』의 내용을 참조하십시오.

PROCEDURE *schema.**

스키마의 모든 기존 프로시저와 앞으로 작성될 프로시저에 대한 명시적 권한 부여를 식별하십시오. *schema.** 특권을 취소하더라도 특정 프로시저에 부여된 특권은 취소되지 않습니다. 동적 SQL문에서 스키마를 지정하지 않으면 CURRENT SCHEMA 특수 레지스터의 스키마가 사용됩니다. 정적 SQL문에서 스키마를 지정하지 않으면 QUALIFIER 프리컴파일/바인드 옵션의 스키마가 사용됩니다.

FROM

EXECUTE 특권을 권한 취소한 사용자로부터 지정하십시오.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 EXECUTE 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여 받은 지정한 모든 사용자에게서 EXECUTE 특권을 취소하십시오. 이 설정이 디폴트 동작입니다.

RESTRICT

다음 모두에 해당될 때는 EXECUTE 특권을 취소할 수 없도록 지정하십시오 (SQLSTATE 42893).

- 지정한 루틴이 뷰, 트리거, 제한조건, 인덱스 확장자, SQL 함수, SQL 메소드 변환 그룹에서 사용되거나 전래 함수의 SOURCE로 참조될 경우
- EXECUTE 특권을 상실하여 뷰, 트리거, 제한조건, 인덱스 확장, SQL 함수, SQL 메소드, 변환 그룹 또는 전래 함수의 소유자가 지정된 루틴을 더 이상 실행할 수 없는 경우.

규칙

- 'SYSIBM' 스키마 또는 'SYSFUN' 스키마로 정의된 함수나 메소드에 대한 EXECUTE 특권은 취소할 수 없습니다(SQLSTATE 42832).
- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.ROUTINEAUTH 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해 다음과 같습니다.
    - 모든 행이 'U' GRANTEETYPE을 가진 경우, USER로 가정됩니다.
    - 모든 행이 'G' GRANTEETYPE을 가진 경우, GROUP으로 가정됩니다.
    - 모든 행이 'R' GRANTEETYPE을 가진 경우, ROLE로 가정됩니다.
    - 모든 행이 GRANTEETYPE에 동일한 값을 갖지 않으면, 오류가 리턴됩니다(SQLSTATE 56092).

주

- 패키지가 루틴(함수, 메소드 또는 프로시저)에 종속되고 해당 루틴에 대한 EXECUTE 권한이 PUBLIC, 사용자 또는 역할에서 취소되는 경우, 패키지 소유자가 루틴에 대한 EXECUTE 특권을 계속 보유하지 않으면 루틴이 함수 또는 메소드인 경우에는 패키지가 작동 불능이 되고 루틴이 프로시저인 경우에는 패키지가 유효하지 않게 됩니다. 패키지 소유자는 다음 경우 EXECUTE 특권을 계속 보유할 수 있습니다.
  - 패키지 소유자에게 EXECUTE 특권이 명시적으로 부여됨
  - 패키지 소유자가 EXECUTE 특권을 보유한 역할의 구성원임
  - EXECUTE 특권이 PUBLIC에 부여되지 않음

  정적 패키지의 경우 그룹 특권이 고려되지 않기 때문에 패키지 소유자가 속한 그룹이 EXECUTE 특권을 보유한 경우에도 패키지가 작동 불능(함수 또는 메소드의 경우) 또는 유효하지 않게 됩니다(프로시저의 경우).

예:

예 1: 함수 CALC_SALARY에 대한 EXECUTE 특권을 사용자 JONES로부터 권한 취소합니다. 스키마에 함수 이름이 CALC_SALARY인 함수가 한 개만 있다고 가정합니다.

```
REVOKE EXECUTE ON FUNCTION CALC_SALARY FROM JONES RESTRICT
```

예 2: 현재 서버에 있는 모든 사용자로부터 VACATION_ACCR 프로시저에 대한 EXECUTE 특권을 취소하십시오.

```
REVOKE EXECUTE ON PROCEDURE VACATION_ACCR FROM PUBLIC RESTRICT
```

예 3: 함수 NEW_DEPT_HIRES에 대한 EXECUTE 특권을 HR(인적 자원)로부터 취소하십시오. 이 함수에는 유형이 각각 INTEGER와 CHAR(10)인 두 개의 입력 매개변수가 있습니다. 스키마에 이름이 NEW_DEPT_HIRES인 함수가 여러 개 있다고 가정합니다.

```
REVOKE EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10))
  FROM HR RESTRICT
```

예 4: 유형 EMPLOYEE의 메소드 SET_SALARY에 대한 EXECUTE 특권을 사용자 Jones로부터 취소하십시오.

```
REVOKE EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE FROM JONES RESTRICT
```

REVOKE(스키마 특권)

이러한 형태의 REVOKE문은 스키마에 대한 특권을 취소합니다.

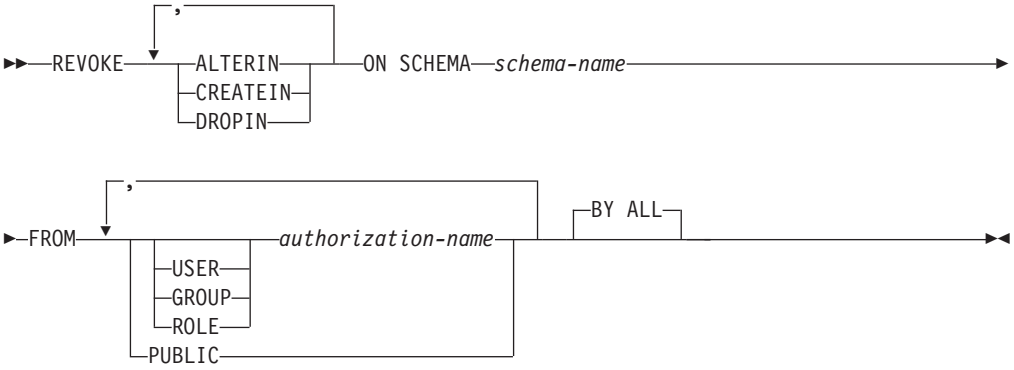
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문



설명

ALTERIN
스키마에 있는 오브젝트에 주석을 달거나 변경할 수 있는 특권을 취소합니다.

CREATEIN
스키마에 오브젝트를 작성할 수 있는 특권을 취소합니다.

DROPIN
스키마의 오브젝트를 삭제할 수 있는 특권을 취소합니다.

ON SCHEMA *schema-name*
특권이 취소될 스키마의 이름을 지정합니다.

FROM
특권을 취소할 사용자를 지정합니다.

USER
*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여받은 지정한 모든 사용자에게서 지정한 각 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.SCHEMAAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다(SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. 기타 특권을 PUBLIC, 그룹 또는 역할에서 보유하거나 또는 사용자가 DBADM과 같은 상위 레벨의 권한을 보유하면 사용자는 태스크를 처리할 수 있습니다.

예:

예 1: USER4가 유일한 사용자이고 그룹이 아닌 상태에서 사용자 USER4로부터 스키마 DEPTIDX를 작성할 수 있는 특권을 취소하십시오.

```
REVOKE CREATEIN ON SCHEMA DEPTIDX FROM USER4
```

예 2: 사용자 CHEF와 그룹 WAITERS에서 스키마 LUNCH의 오브젝트를 삭제할 수 있는 특권을 취소하십시오.

```
REVOKE DROPIN ON SCHEMA LUNCH
  FROM USER CHEF, GROUP WAITERS
```

REVOKE(보안 레이블)

이러한 형태의 REVOKE문은 레이블 기반 액세스 제어(LBAC) 보안 레이블에 대한 특권을 취소합니다.

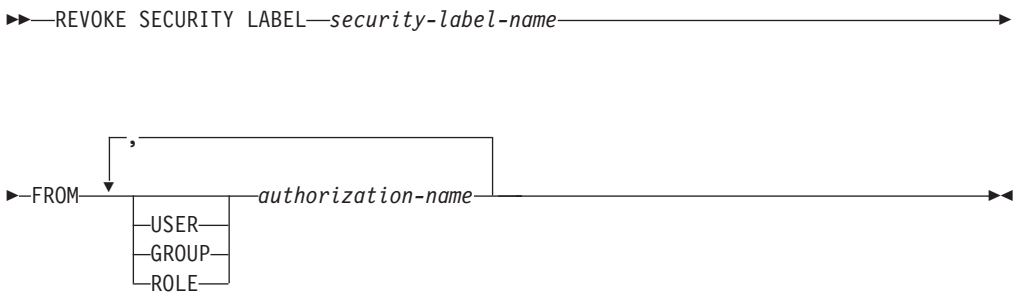
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



설명

SECURITY LABEL *security-label-name*

security-label-name 보안 레이블에 대한 특권을 취소합니다. 이름은 보안 규정을 사용하여 규정해야 하며(SQLSTATE 42704), 현재 서버에 존재하고(SQLSTATE 42704) *authorization-name*이 보유한 보안 레이블을 식별해야 합니다(SQLSTATE 42504).

FROM

지정된 보안 레이블을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다. 역할 이름이 현재 서버에 존재해야 합니다(SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

규칙

- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.SECURITYLABELACCESS 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해 다음과 같습니다.
    - 모든 행이 'U' GRANTEETYPE을 가진 경우, USER로 가정됩니다.
    - 모든 행이 'G' GRANTEETYPE을 가진 경우, GROUP으로 가정됩니다.
    - 모든 행이 'R' GRANTEETYPE을 가진 경우, ROLE로 가정됩니다.
    - 모든 행이 GRANTEETYPE에 동일한 값을 갖지 않으면, 오류가 리턴됩니다 (SQLSTATE 56092).

예:

예 1: 사용자 WALID로부터 보안 규정 DATA_ACCESS의 일부인 EMPLOYEESECLABEL 보안 레이블의 권한을 취소합니다.

```
REVOKE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
  FROM USER WALID
```

REVOKE(시퀀스 특권)

이러한 형태의 REVOKE문은 시퀀스에 대한 특권을 취소합니다.

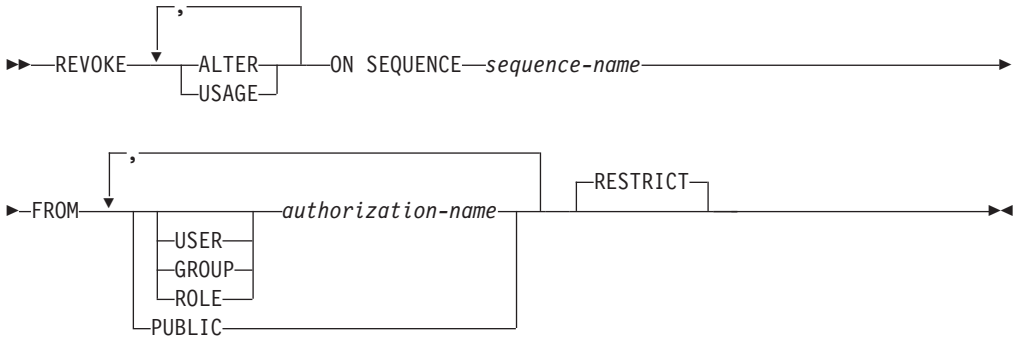
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다. 그러나 바인드 옵션인 DYNAMICRULES BIND가 적용되면 명령문을 동적으로 준비할 수 없습니다 (SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID가 보유한 특권은 ACCESSCTRL 또는 SECADM 권한을 포함해야 합니다.

구문



설명

ALTER

시퀀스 등록 정보를 변경하거나 ALTER SEQUENCE문을 사용하여 시퀀스 번호 생성을 재시작할 특권을 취소합니다.

USAGE

nextval-expression 또는 *prevval-expression*을 사용하여 시퀀스를 참조할 수 있는 특권을 취소합니다.

ON SEQUENCE *sequence-name*

지정된 특권이 취소될 시퀀스를 식별합니다. 내재적 또는 명시적 스키마를 포함하는 시퀀스 이름은 현재 서버에 이미 존재하는 시퀀스를 고유하게 식별해야 합니다. 이 이름에 의한 시퀀스가 없을 경우 오류가 발생합니다(SQLSTATE 42704).

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 지정된 특권을 취소합니다.

RESTRICT

이 선택적 키워드는 오브젝트가 취소되고 있는 특권에 따라 달라지는 경우 명령문이 실패한다는 것을 나타냅니다.

규칙

- 지정된 각 *authorization-name*에 대해 USER, GROUP 및 ROLE이 지정되지 않은 경우
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.SEQUENCEAUTH 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해 다음과 같습니다.
    - 모든 행이 'U' GRANTEETYPE을 가진 경우, USER로 가정됩니다.
    - 모든 행이 'G' GRANTEETYPE을 가진 경우, GROUP으로 가정됩니다.
    - 모든 행이 'R' GRANTEETYPE을 가진 경우, ROLE로 가정됩니다.
    - 모든 행이 GRANTEETYPE에 동일한 값을 갖지 않으면, 오류가 리턴됩니다 (SQLSTATE 56092).

주

- 패키지가 바인드된 권한 부여 ID에서 시퀀스에 대한 특권을 취소하면 권한 부여 ID가 다른 방법을 통해(예를 들어, 특권을 보유한 역할의 멤버쉽을 통해) 시퀀스에 대한 특권을 계속 보유하지 않으면 패키지가 유효하지 않게 됩니다.
- 특정의 특권을 취소한다고 해서 조치를 수행할 능력이 제거되는 것은 아닙니다. 사용자는 PUBLIC 또는 사용자가 속하는 그룹이 다른 특권을 보유하거나 사용자에게 DBADM과 같은 상위 레벨의 권한이 있는 경우 진행할 수 있습니다.

예:

예 1: 사용자 ENGLES로부터 GENERATE_ID라는 시퀀스에 대한 USAGE 특권을 취소하십시오. 이 시퀀스 및 권한 받은 사용자에 대해 SYSCAT.SEQUENCEAUTH 카탈로그 뷰에 한 행이 있으며, GRANTEETYPE 값은 U입니다.

```
REVOKE USAGE ON SEQUENCE GENERATE_ID FROM ENGLES
```

예 2: 이전에 모든 로컬 사용자에게 권한 부여된 시퀀스 GENERATE_ID에 대한 변경 특권을 취소하십시오. (특정 사용자에게 권한 부여하는 것은 영향받지 않습니다.)

```
REVOKE ALTER ON SEQUENCE GENERATE_ID FROM PUBLIC
```

예 3: 사용자 PELLOW와 MLI 및 그룹 PLANNERS에서 시퀀스 GENERATE_ID에 대한 모든 특권을 취소하십시오.

```
REVOKE ALTER, USAGE ON SEQUENCE GENERATE_ID
  FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

REVOKE(서버 특권)

이러한 형태의 REVOKE문은 pass-through 모드에서 지정된 데이터 소스를 액세스하고 사용할 특권을 취소합니다.

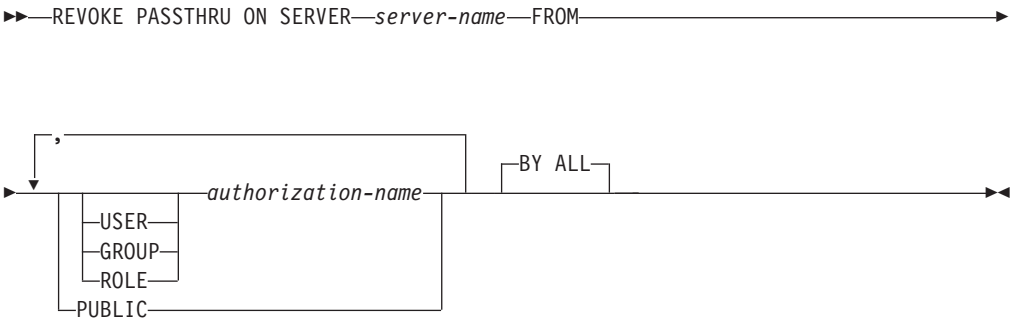
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 ACCESSCTRL 또는 SECADM 권한이 포함되어야 합니다.

구문



설명

SERVER *server-name*

pass-through 모드에서 사용할 특권이 취소되고 있는 데이터 소스에 이름을 지정합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

*server-name*을 pass-through하는 특권을 PUBLIC에서 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여 받은 지정한 모든 사용자에게서 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.PASSTHRUAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다 (SQLSTATE 56092).

예:

예 1: 데이터 소스를 pass-through하는 USER6 특권을 취소하십시오.

```
REVOKE PASSTHRU ON SERVER MOUNTAIN FROM USER USER6
```

예 2: 데이터 소스 EASTWING을 pass-through하는 그룹 D024 특권을 취소하십시오.

```
REVOKE PASSTHRU ON SERVER EASTWING FROM GROUP D024
```

그룹 D024의 구성원들은 더 이상 그룹 ID를 사용하여 EASTWING을 pass-through할 수 없습니다. 그러나 구성원에게 자신의 사용자 ID하에서 EASTWING으로 pass-through할 수 있는 권한이 있는 경우 구성원은 이러한 특권을 유지합니다.

REVOKE(SETSESSIONUSER 특권)

이러한 형태의 REVOKE문은 하나 이상의 권한 부여 ID로부터 하나 이상의 SETSESSIONUSER 특권을 취소합니다.

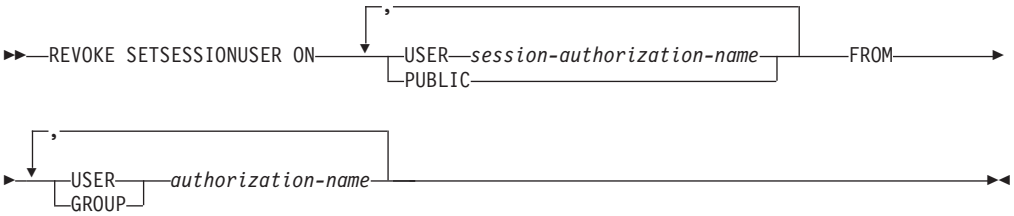
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권에는 SECADM 권한이 포함되어야 합니다.

구문



설명

SETSESSIONUSER ON
새 권한 부여 ID의 식별을 가정하는 특권을 권한 취소합니다.

USER *session-authorization-name*
SET SESSION AUTHORIZATION문을 사용하여 *authorization-name*이 가정할 수 있는 권한 부여 ID를 지정합니다. *session-authorization-name*은 *authorization-name*이 가정할 수 있는 그룹이 아닌 사용자를 식별해야 합니다 (SQLSTATE 42504).

PUBLIC
세션 권한 부여를 설정할 모든 특권이 권한 취소되도록 지정합니다.

FROM
특권을 취소할 사용자를 지정합니다.

USER
*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP
*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열하십시오.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

예:

예 1: 사용자 PAUL은 WALID에게 세션 권한 부여를 설정하고 SQL문을 사용자 WALID로 실행할 수 있는 특권을 갖습니다. 다음 명령문은 해당 특권을 취소합니다.

```
REVOKE SETSESSIONUSER ON USER WALID
  FROM USER PAUL
```

예 2: 사용자 GUYLAINE이 BOBBY, RICK 또는 KEVIN에 대한 세션 권한 부여를 설정하므로 BOBBY, RICK 또는 KEVIN으로서 SQL문을 실행할 특권을 보유합니다. 다음 명령문은 해당 권한 부여 ID 중 둘을 사용할 특권을 권한 취소합니다. 이 명령문이 실행된 후 GUYLAINE은 세션 권한 부여를 KEVIN으로만 설정할 수 있습니다.

```
REVOKE SETSESSIONUSER ON USER BOBBY, USER RICK
  FROM USER GUYLAINE
```

예 3: 그룹 ACCTG 및 사용자 WALID가 세션 권한 부여를 임의의 권한 부여 ID로 설정할 수 있습니다. 다음 명령문은 ACCTG와 WALID로부터 해당 특권을 권한 취소합니다.

```
REVOKE SETSESSIONUSER ON PUBLIC
  FROM USER WALID, GROUP ACCTG
```

REVOKE(테이블 스페이스 특권)

이러한 형태의 REVOKE문은 테이블에 대한 USE 특권을 취소합니다.

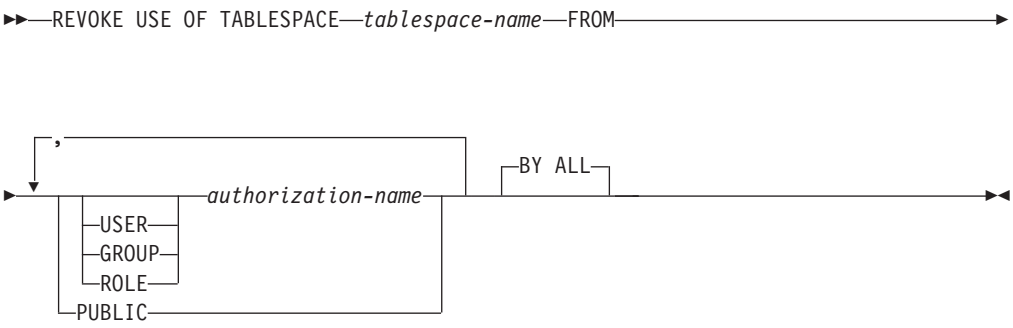
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 ACCESSCTRL, SECADM, SYSCTRL 또는 SYSADM 권한을 포함해야 합니다.

구문



설명

USE

테이블 작성시 테이블 스페이스를 지정하거나 디폴트값으로 설정할 수 있는 특권을 취소합니다.

OF TABLESPACE *tablespace-name*

USE 특권이 취소될 테이블 스페이스를 지정합니다. 테이블 스페이스는 SYSCATSPACE(SQLSTATE 42838) 또는 SYSTEM TEMPORARY 테이블 스페이스(SQLSTATE 42809)일 수 없습니다.

FROM

USE 특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 USE 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여 받은 지정한 모든 사용자에게서 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.TBSPACEAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다 (SQLSTATE 56092).

주

- USE 특권을 취소한다고 해서 해당 테이블 스페이스에 테이블을 작성할 수 있는 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 USE 특권을 보유하거나 DBADM과 같은 상위 레벨 권한을 가지는 경우 사용자는 여전히 테이블을 해당 테이블 스페이스에 작성할 수 있습니다.

예:

예 1: 테이블을 테이블 스페이스 PLANS에 작성할 수 있는 특권을 사용자 BOBBY로부터 취소하십시오.

```
REVOKE USE OF TABLESPACE PLANS FROM USER BOBBY
```

REVOKE(테이블, 뷰 또는 별칭 특권)

REVOKE문의 이 형태는 테이블, 뷰 또는 별칭에 대한 특권을 취소합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

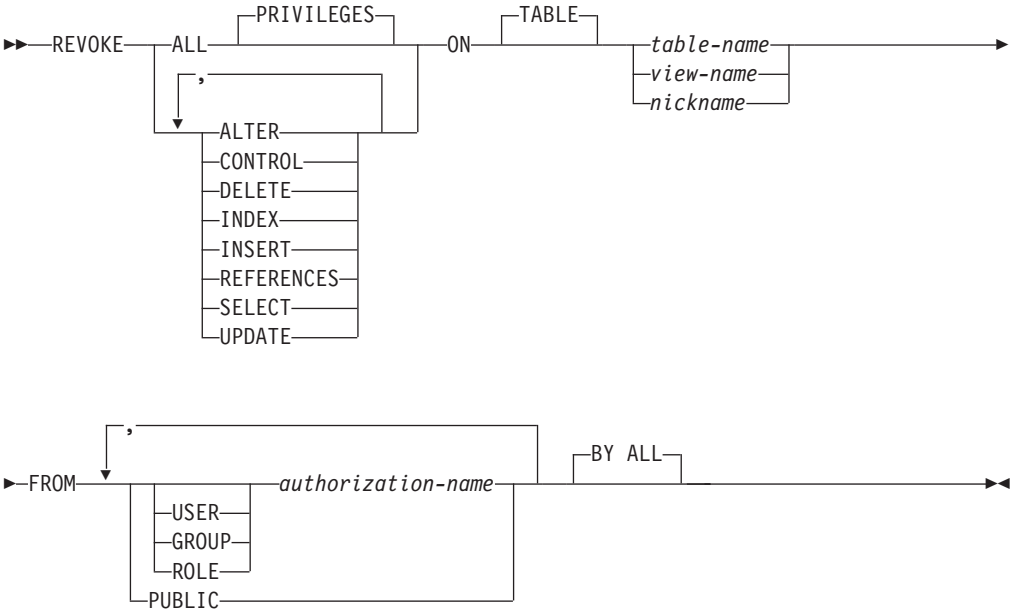
권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 참조된 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- ACCESSCTRL 또는 SECADM 권한

ACCESSCTRL 또는 SECADM 권한은 CONTROL 특권을 권한 취소하거나, 카탈로그 테이블 및 뷰에 대한 특권을 권한 취소하는 데 필요합니다.

구문



설명

ALL 또는 ALL PRIVILEGES

지정된 테이블, 뷰 또는 별칭에 대한 권한 부여 이름이 보유하는 (CONTROL을 제외한) 모든 특권을 권한 취소하십시오.

ALL을 사용하지 않는 경우, 아래 나열된 키워드가 하나 이상 사용되어야 합니다. 각 키워드는 기술된 특권을 취소하나, ON절에 이름 지정된 테이블, 뷰 또는 또는 별칭에 적용될 때에만 권한 취소됩니다. 같은 키워드는 한 번 이상 지정될 수 없습니다.

ALTER

기본 테이블 정의에 컬럼 추가, 테이블상의 기본 키나 고유 제한조건 작성 또는 삭제(drop), 테이블상의 외부 키 작성 또는 삭제(drop), 테이블과 뷰 또는 별칭에 주석 추가/변경, 점검 제한조건 작성 또는 삭제(drop), 트리거 작성, 별칭에 대한 컬럼 옵션 추가, 재설정 또는 삭제(drop), 별칭 컬럼 이름이나 데이터 유형을 변경할 특권을 권한 취소하십시오.

CONTROL

테이블, 뷰 또는 별칭을 삭제(drop)할 수 있는 능력과 테이블 및 인덱스에 대해 Runstats 유틸리티를 실행할 수 있는 능력을 권한 취소하십시오.

*authorization-name*으로부터 CONTROL 특권을 취소해도 그 오브젝트상의 사용자에게 권한 부여된 다른 특권은 권한 취소되지 않습니다.

DELETE

테이블, 갱신 가능 뷰 또는 별칭에서 행 삭제 권한을 취소합니다.

INDEX

테이블의 인덱스 또는 별칭의 인덱스 스펙 작성 특권을 권한 취소하십시오. 인덱스나 인덱스 스펙 작성자는 자동으로 인덱스나 인덱스 스펙에 대한 CONTROL 특권을 가집니다. (작성자에게 인덱스나 인덱스 스펙을 삭제(drop)할 권한을 줍니다.) 또한 작성자는 INDEX 특권이 권한 취소될 지라도 이 특권을 보유합니다.

INSERT

테이블, 갱신 가능 뷰 또는 별칭에 행을 삽입하고, IMPORT 유틸리티를 실행하는 특권을 권한 취소하십시오.

REFERENCES

테이블을 상위 테이블로 참조하는 외부 키를 작성 또는 삭제(drop)하는 특권을 권한 취소하십시오. 모든 컬럼 레벨의 REFERENCES 특권도 권한 취소됩니다.

SELECT

테이블이나 뷰에서 행을 검색하고 테이블에 뷰를 작성하며 테이블이나 뷰에 대해 EXPORT 유틸리티를 실행할 수 있는 특권을 취소하십시오.

SELECT 특권을 취소하면, 일부 뷰가 작동 불능으로 표시됩니다. (작동 불능 뷰에 대한 정보는『CREATE VIEW』를 참조하십시오.)

UPDATE

테이블, 갱신 가능 뷰 또는 별칭에서의 행 갱신 권한을 취소합니다. 모든 컬럼 레벨의 UPDATE 특권도 권한 취소하십시오.

ON TABLE *table-name* **또는** *view-name* **또는** *nickname*

특권이 권한 취소될 해당 테이블, 뷰 또는 별칭을 지정하십시오. *table-name*은 임시 테이블로 선언될 수 없습니다(SQLSTATE 42995).

FROM

특권을 취소할 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정하십시오.

ROLE

*authorization-name*이 역할 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹 또는 역할의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

PUBLIC에서 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이 특권을 명시적으로 부여받은 지정한 모든 사용자에게서 지정한 각 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 각 *authorization-name*에 대해, USER, GROUP 또는 ROLE이 지정되어 있지 않은 경우:
  - 권한 받은 사용자가 *authorization-name*인 SYSCAT.TABAUTH 및 SYSCAT.COLAUTH 카탈로그 뷰에서 지정된 오브젝트에 대한 모든 행의 경우:
    - 모든 행에 'U'의 GRANTEETYPE이 있는 경우, USER로 가정합니다.
    - 모든 행에 'G'의 GRANTEETYPE이 있는 경우, GROUP으로 가정합니다.
    - 모든 행에 'R'의 GRANTEETYPE이 있는 경우, ROLE로 가정합니다.
    - 모든 행에 GRANTEETYPE과 동일한 값이 없으면 오류가 리턴됩니다 (SQLSTATE 56092).

주

- 뷰의 소유자인 *authorization-name*에서 특권이 권한 취소되는 경우(SYSCAT.VIEWS의 OWNER 컬럼에 기록된), 그 특권은 모든 종속 뷰에서도 권한 취소됩니다.

- 뷰의 소유자가 뷰 정의가 종속된 일부 오브젝트에 대해 SELECT 특권을 상실한 경우(뷰 정의가 종속된 오브젝트가 삭제(drop)되었거나, 다른 뷰의 경우 작동 불능 상태가 된 경우), 뷰는 작동 불능 상태가 됩니다.

  그러나 ACCESSCTRL 또는 SECADM이 명시적으로 보유한 사용자가 소유자로부터 뷰에 대한 모든 특권을 권한 취소한 경우, OWNER의 레코드는 SYSCAT.TABAUTH에 나타나지 않으나 뷰에는 아무 것도 발생하지 않으며 조작은 가능합니다.
- 작동 불능 뷰에 대한 특권은 권한 취소될 수 없습니다.
- 패키지가 바운드된 권한 부여 ID가 패키지가 종속되는 오브젝트에 대한 권한을 잃으면 패키지가 무효화됩니다. 다음 중 하나의 방식으로 특권을 잃을 수 있습니다.
  - 특권이 권한 부여 ID에서 취소됨
  - 특권이 권한 부여 ID가 구성원인 역할에서 취소됨
  - 특권이 PUBLIC에서 취소됨

  응용프로그램의 바인드 또는 리바인드 조작이 정상적으로 실행되거나, 그 응용프로그램이 실행되고 데이터베이스 관리 프로그램이 카탈로그에 저장된 정보를 사용하여 응용프로그램을 정상적으로 리바인드할 때까지 패키지는 유효하지 않은 상태로 있게 됩니다. 권한 취소로 인해 유효하지 않은 것으로 표시된 패키지는 추가로 권한을 부여할 필요없이 성공적으로 리바인드됩니다.

  예를 들어, USER1이 소유한 패키지에 테이블 T1의 SELECT가 포함되고, 테이블 T1에 대한 SELECT 특권이 USER1에서 권한 취소되면 그 패키지는 유효하지 않은 것으로 표시됩니다. SELECT 권한이 재부여되거나 사용자가 DBADM 권한을 가지고 있는 경우, 그 패키지는 실행 시 성공적으로 리바인드됩니다.

  다른 예는 R1의 구성원인 USER1에서 소유한 패키지입니다. 패키지에 테이블 T1의 SELECT가 포함되고, 테이블 T1에 대한 SELECT 특권이 역할 R1에서 권한 취소됩니다. USER1이 다른 의미로 테이블 T1에서 SELECT 특권을 보유하지 않는다고 가정하며, 패키지가 유효하지 않은 것으로 표시됩니다.
- FROM절에서 OUTER(*Z*)의 사용을 포함하는 패키지, 트리거 또는 뷰는 *Z*의 모든 서브테이블이나 서브뷰에 SELECT 특권을 가지는 것에 종속됩니다. 마찬가지로, DEREF(*Y*)(여기서, *Y*는 목표 테이블 또는 뷰가 *Z*인 참조 유형)의 사용을 포함하는 패키지, 트리거 또는 뷰는 *Z*의 모든 서브테이블이나 서브뷰에 특권을 가지는 것에 종속됩니다. 패키지가 바운드된 권한 부여 ID 또는 트리거나 뷰의 소유자가 SELECT 특권을 유실하면 이들 패키지는 무효화되며 트리거나 뷰는 동작 불능으로 됩니다. 다음 중 하나의 방식으로 SELECT 특권을 잃을 수 있습니다.
  - SELECT 특권이 권한 부여 ID에서 취소됨
  - SELECT 특권이 권한 부여 ID가 구성원인 역할에서 취소됨
  - SELECT 특권이 PUBLIC에서 취소됨

- 테이블, 뷰 또는 별칭 특권은 CONTROL 특권도 권한 취소하지 않고 오브젝트에 대한 CONTROL을 보유하는 *권한 부여 이름*으로부터 취소할 수 없습니다(SQLSTATE 42504).
- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. 기타 특권을 PUBLIC, 그룹 또는 역할에서 보유하거나 또는 테이블이나 뷰의 스키마에서 사용자가 ALTERIN과 같은 권한을 보유하면 사용자는 태스크를 처리할 수 있습니다.
- 구체화된 쿼리 테이블의 소유자가 구체화된 쿼리 테이블 정의가 종속된 테이블에서 SELECT 특권을 상실한 경우(또는 구체화된 쿼리 테이블 정의가 종속된 테이블이 삭제된 경우), 구체화된 쿼리 테이블은 삭제됩니다.

  그러나 SECADM 또는 ACCESSCTRL 권한이 명시적으로 소유자부터 구체화된 쿼리 테이블에 있는 모든 특권을 취소하면 해당 소유자의 SYSTABAUTH에 있는 레코드는 삭제되지만 구체화된 쿼리 테이블에는 아무런 영향을 주지 않으며, 작동 가능합니다.
- 별칭 특권을 취소해도 데이터 소스 오브젝트(테이블이나 뷰) 특권에 아무 영향도 미치지 않습니다.
- 다른 몇몇 오브젝트가 SQL 함수나 메소드에 종속적이므로, SQL 함수나 메소드를 삭제(drop)할 수 없는 경우에는 SQL 함수나 메소드에서 직접 또는 간접적으로 참조되는 테이블 또는 뷰에 대한 SELECT 특권 취소에 실패할 수 있습니다(SQLSTATE 42893).
- 다음과 같은 경우에 SELECT 특권을 취소하면 SQL 함수나 메소드 내용이 삭제될 수 있습니다.
  - SQL 함수나 메소드 본문의 소유자가 SQL 함수나 메소드 본문 정의가 의존하는 일부 오브젝트에 대한 SELECT 특권을 잃게 됩니다. 소유자가 구성원인 역할이나 PUBLIC으로부터 취소되기 때문에 특권이 유실될 수 있습니다.
  - SQL 함수나 메소드 내용 정의가 의존하는 오브젝트가 삭제됩니다.

  그러나 다른 오브젝트가 함수나 메소드에 의존하면 취소할 수 없습니다(SQLSTATE 42893).

예:

예 1: 사용자 ENGLES에서 EMPLOYEE 테이블에 대한 SELECT 특권을 권한 취소합니다. 이 테이블 및 권한 받은 사용자에 대해 SYSCAT.TABAUTH 카탈로그 뷰에 한 행이 있고, GRANTEETYPE 값은 U입니다.

```
REVOKE SELECT
  ON TABLE EMPLOYEE
  FROM ENGLES
```

예 2: 이전에 모든 로컬 사용자에게 권한 부여된 테이블 EMPLOYEE에 대한 갱신 특권을 취소하십시오. 특정 사용자에게 권한 부여하는 것은 영향받지 않음에 유의하십시오.

```
REVOKE UPDATE
  ON EMPLOYEE
  FROM PUBLIC
```

예 3: 사용자 PELLOW와 MLI, 그룹 PLANNERS으로부터 테이블 EMPLOYEE에 대한 모든 특권을 취소하십시오.

```
REVOKE ALL
  ON EMPLOYEE
  FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

예 4: 사용자 JOHN으로부터 테이블 CORPDATA.EMPLOYEE에 대한 SELECT 특권을 취소하십시오. 이 테이블 및 권한 받은 사용자에 대해 SYSCAT.TABAUTH 카탈로그 뷰에 한 행이 있고, GRANTEETYPE 값은 U입니다.

```
REVOKE SELECT
       ON CORPDATA.EMPLOYEE FROM JOHN
```

또는

```
REVOKE SELECT
       ON CORPDATA.EMPLOYEE FROM USER JOHN
```

GROUP JOHN으로부터 특권을 취소하려고 하면 오류가 발생함에 유의하십시오. 이는 GROUP JOHN에게 특권이 주어져 있지 않기 때문입니다.

예 5: 그룹 JOHN으로부터 테이블 CORPDATA.EMPLOYEE에 대한 SELECT 특권을 취소하십시오. 이 테이블 및 권한 받은 사용자에 대해 SYSCAT.TABAUTH 카탈로그 뷰에 한 행이 있으며 GRANTEETYPE 값은 G입니다.

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM JOHN
```

또는

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM GROUP JOHN
```

예 6: 별칭 ORAREM1에 대한 인덱스 스펙을 작성하는 사용자 SHAWN으로부터 특권을 취소하십시오.

```
REVOKE INDEX
       ON ORAREM1 FROM USER SHAWN
```

REVOKE(워크로드 특권)

이 형태의 REVOKE문은 워크로드에 대한 USAGE 특권을 취소합니다.

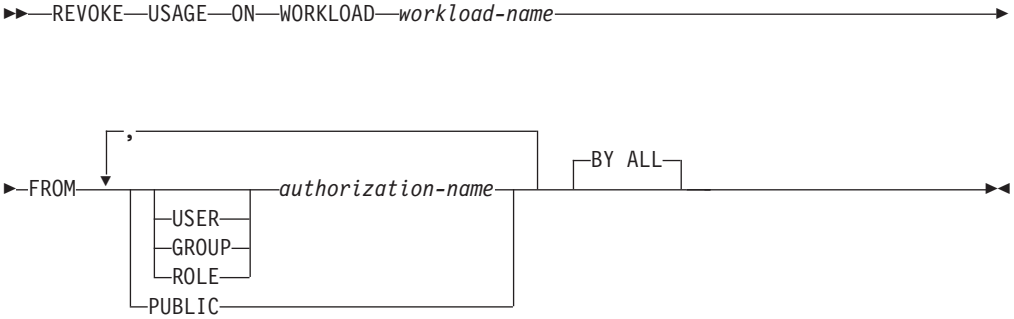
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 ACCESSCTRL, SECADM 또는 WLMADM 권한을 포함해야 합니다.

구문



설명

USAGE

워크로드 사용 특권을 취소합니다.

ON WORKLOAD *workload-name*

USAGE 특권이 취소될 워크로드를 식별합니다. 이 이름은 한 부분의 이름입니다. *workload-name*은 현재 서버에 존재하는 워크로드를 식별해야 합니다(SQLSTATE 42704). 이름은 'SYSDEFAULTADMWORKLOAD'가 될 수 없습니다(SQLSTATE 42832).

FROM

USAGE 특권이 취소된 출처를 지정하십시오.

USER

*authorization-name*이 사용자를 식별하도록 지정하십시오.

GROUP

*authorization-name*이 그룹을 식별하도록 지정합니다.

ROLE

*authorization-name*이 현재 서버에서 기존 역할을 식별함을 지정합니다 (SQLSTATE 42704).

authorization-name,...

하나 이상의 사용자, 그룹 또는 역할의 권한 부여 ID를 나열합니다. 권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다 (SQLSTATE 42502).

PUBLIC

PUBLIC에서 USAGE 특권을 취소합니다.

BY ALL

누가 권한을 부여했는지에 상관없이 해당 특권을 명시적으로 부여 받은 이름 지정된 모든 사용자에게서 USAGE 특권을 취소합니다. 이 설정이 디폴트 동작입니다.

규칙

- 지정된 *authorization-name*마다, USER, GROUP 또는 ROLE 키워드 중 어느 것도 지정하지 않은 경우 권한 받은 사용자가 *authorization-name*인 SYSCAT.WORKLOADAUTH 카탈로그 뷰에 지정된 오브젝트의 모든 행에 대해,
  - GRANTEETYPE이 'U'이면 USER가 가정됩니다.
  - GRANTEETYPE이 'G'이면 GROUP이 가정됩니다.
  - GRANTEETYPE이 'R'이면 ROLE이 가정됩니다.
  - GRANTEETYPE이 동일한 값을 가지고 있지 않으면 오류가 리턴됩니다 (SQLSTATE 56092).

주

- REVOKE문은 명령문을 발행하는 연결에 대해서도 커미트될 때까지 적용되지 않습니다.

예 :

사용자 LISA로부터 워크로드 CAMPAIGN을 사용하기 위한 특권을 취소하십시오.

```
REVOKE USAGE ON WORKLOAD CAMPAIGN FROM USER LISA
```

REVOKE(XSR 오브젝트 특권)

이 양식의 REVOKE문은 XSR 오브젝트에 대한 USAGE 특권을 권한 취소합니다.

호출

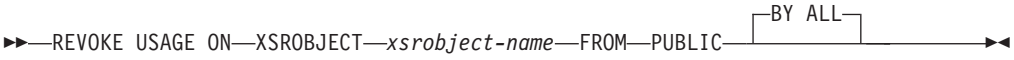
REVOKE문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때만(SQLSTATE 42509) 동적으로 준비될 수 있는 실행문입니다.

권한 부여

다음 권한 중 하나가 필요합니다.

- ACCESSCTRL 또는 SECADM 권한

구문



설명

ON XSROBJECT *xsrobject-name*

이 이름은 USAGE 특권이 권한 취소된 XSR 오브젝트를 식별합니다. 내재적 또는 명시적 스키마 규정자를 포함하는 *xsrobject-name*은 현재 서버에 있는 기존 XSR 오브젝트를 고유하게 식별해야 합니다. 이 이름의 XSR 오브젝트가 지정된 스키마에 없으면 오류가 발생합니다(SQLSTATE 42704).

FROM PUBLIC

PUBLIC에서 USAGE 특권을 취소합니다.

BY ALL

누가 특권을 부여했는지에 상관없이 이들 특권을 명시적으로 부여받은 모든 사용자에게서 각 지정된 특권을 권한 취소합니다. 이 설정이 디폴트 동작입니다.

예 :

PUBLIC에서 XML 스키마 MYSCHEMA에 대한 사용 특권을 권한 취소하십시오.

```
REVOKE USAGE ON XSROBJECT MYSCHEMA FROM PUBLIC
```

ROLLBACK

ROLLBACK문은 작업 단위(UOW) 또는 세이브포인트 내에서 작성된 데이터베이스 변경사항을 백아웃하는 데 사용됩니다.

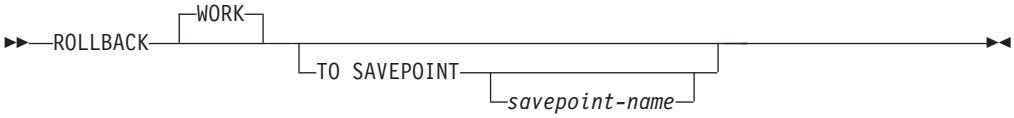
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

ROLLBACK문이 실행되는 작업 단위(UOW)는 종료되고 새로운 작업 단위가 시작됩니다. 작업 단위 중에 작성된 모든 데이터베이스 변경사항을 백아웃합니다.

그러나 다음 명령문은 트랜잭션의 제어를 받지 않으며, 트랜잭션에 의한 변경사항은 ROLLBACK문과 무관합니다.

- SET CONNECTION
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT PACKAGESET
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET PASSTHRU

주: SET PASSTHRU문이 트랜잭션의 제어를 받지 않아도 명령문이 트랜잭션 제어 하에 있으면 passthru 세션은 초기화됩니다.

- SET PATH
- SET SCHEMA
- SET SERVER OPTION

시퀀스 값과 식별 값 생성은 트랜잭션의 제어를 받지 않습니다. *nextval-expression*을 사용하거나 ID 컬럼이 있는 테이블에 행을 삽입하여 생성되거나 사용되는 값은 ROLLBACK문을 발행해도 영향을 받지 않습니다. 뿐만 아니라 ROLLBACK문을 발행해도 *prevval-expression* 또는 IDENTITY_VAL_LOCAL 함수가 리턴한 값은 영향을 받지 않습니다.

전역 변수의 값 수정은 트랜잭션의 제어를 받지 않습니다. ROLLBACK문은 전역 변수에서 지정한 값에 영향을 주지 않습니다.

TO SAVEPOINT

부분 롤백(ROLLBACK TO SAVEPOINT)이 수행됨을 지정합니다. 현재 세이브포인트 레벨에 사용 중인 세이브포인트가 없는 경우(SAVEPOINT문의 설명에 있는 『규칙』 절 참조) 오류가 발생합니다(SQLSTATE 3B502). 성공적인 롤백 이후에는 세이브포인트가 계속 존재하지만 중첩된 모든 세이브포인트는 해제되며 더 이상 존재하지 않습니다. 중첩된 세이브포인트(존재할 경우)는 롤백된 것으로 간주되며 현재 세이브포인트에 롤백의 일부분으로 해제됩니다. *savepoint-name*이 제공되지 않을 경우 현재 세이브포인트 레벨에서 가장 최근에 설정된 세이브포인트로 롤백이 발생합니다.

이 절이 생략되면 ROLLBACK문이 트랜잭션 전체를 롤백하고 트랜잭션 내의 세이브포인트가 해제됩니다.

savepoint-name

롤백 조작에 사용되는 세이브포인트를 지정합니다. 지정된 *savepoint-name*은 'SYS'로 시작될 수 없습니다(SQLSTATE 42939). 성공적인 롤백 조작 이후에는 이름이 지정된 세이브포인트가 계속 존재합니다. 세이브포인트 이름이 없는 경우에는 오류가 발생합니다(SQLSTATE 3B001). 세이브포인트 설정 이후에 작성된 데이터 및 스키마 변경사항은 실행 취소됩니다.

주

- 모든 보류된 잠금은 작업 단위의 ROLLBACK 시 해제됩니다. 열린 커서는 모두 닫히고 LOB 로케이터는 모두 없어집니다.
- ROLLBACK문을 실행해도 특수 레지스터값을 변경하는 SET문이나 RELEASE문은 영향을 받지 않습니다.
- 프로그램이 비정상적으로 종료되면 작업 단위가 내재적으로 롤백됩니다.
- 명령문 캐시는 롤백 조작의 영향을 받습니다.

- ROLLBACK TO SAVEPOINT가 커서에 미치는 영향은 세이브포인트 내의 명령문에 따라 달라집니다.
  - 세이브포인트에 커서가 종속되는 DDL이 포함되는 경우 커서는 유효하지 않음으로 표시됩니다. 이러한 커서를 사용하려 하면 오류가 발생합니다(SQLSTATE 57007).
  - 그렇지 않을 경우 다음과 같습니다.
    - 커서가 세이브포인트내에서 참조되는 경우, 커서는 여전히 열려 있고 결과 테이블의 다음 논리 행 앞에 위치합니다. 위치가 지정된 UPDATE 또는 DELETE 문을 발행하기 전에 FETCH를 수행해야 합니다.
    - 그렇지 않으면 커서가 ROLLBACK TO SAVEPOINT의 영향을 받지 않습니다(커서는 여전히 열려 있고 위치 지정됨).
- 동적으로 준비된 명령문 이름은 명령문을 내재적으로 다시 준비할 수 있는 경우에도 세이브포인트 내에서 롤백되는 DDL 조작의 결과로서 여전히 유효합니다.
- ROLLBACK TO SAVEPOINT 조작은 세이브포인트 내에서 작성된 임시 테이블을 삭제합니다. 작성된 임시 테이블이 세이브포인트 내에서 수정되고 해당 테이블이 로그되지 않음으로 정의되는 경우, 테이블의 모든 행이 삭제됩니다.
- ROLLBACK TO SAVEPOINT 조작은 세이브포인트 내에서 선언된 임시 테이블을 삭제합니다. 선언된 임시 테이블이 세이브포인트 내에서 수정되고 해당 테이블이 로그되지 않음으로 정의되는 경우, 테이블의 모든 행이 삭제됩니다.
- ROLLBACK TO SAVEPOINT문 다음의 모든 잠금은 보존됩니다.
- ROLLBACK TO SAVEPOINT 조작 다음의 모든 LOB 로케이터는 보존됩니다.

예 :

최종 커미트 지점 또는 롤백 이후 변경된 사항을 삭제합니다.

```
ROLLBACK WORK
```

SAVEPOINT

트랜잭션 내에 세이브포인트를 설정하려면 SAVEPOINT문을 사용하십시오.

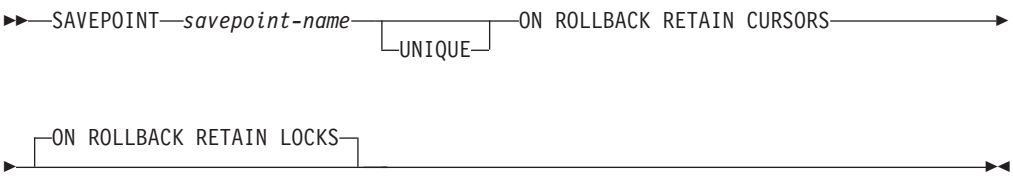
호출

이 명령문은 응용프로그램(스토어드 프로시저 포함)에 임베드하거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

savepoint-name

세이브포인트의 이름을 지정합니다. 지정된 *savepoint-name*은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939). 이 이름에 의한 세이브포인트가 이미 이 세이브포인트 레벨에서 UNIQUE로 정의된 경우 오류가 리턴됩니다(SQLSTATE 3B501).

UNIQUE

세이브포인트가 현재 세이브포인트 레벨 안에서 사용 중인 동안 응용프로그램이 이 세이브포인트 이름을 재사용할 의사가 없음을 지정합니다. *savepoint-name*이 이미 이 세이브포인트 레벨에 존재하는 경우 오류가 리턴됩니다(SQLSTATE 3B501).

ON ROLLBACK RETAIN CURSORS

SAVEPOINT문 후에 처리되는 열린 커서 명령문에 관하여 이 세이브포인트로의 롤백 시에 시스템 동작을 지정합니다. 이 절은 가능한 경우에는 항상 커서가 세이브포인트로의 롤백 조작의 영향을 받지 않음을 표시합니다. 커서가 세이브포인트로의 롤백의 영향을 받는 상황의 경우 『ROLLBACK』을 참조하십시오.

ON ROLLBACK RETAIN LOCKS

세이브포인트의 설정 후에 획득되는 잠금에 관하여 이 세이브포인트로의 롤백 시에 시스템 동작을 지정합니다. 세이브포인트 이후에 획득된 잠금은 추적되지 않으며, 세이브포인트로의 롤백 시에 롤백(해제)되지 않습니다.

규칙

- 세이브포인트 관련 명령문은 트리거 정의에서 사용해서는 안됩니다 (SQLSTATE 42987).
- 다음 중 하나가 발생할 때 새 세이브포인트 레벨이 시작됩니다.
  - 새 작업 단위(UOW)가 시작됩니다.
  - NEW SAVEPOINT LEVEL절로 정의되는 프로시저가 호출됩니다.
  - ATOMIC 복합 SQL문이 시작됩니다.
- 작성을 유발한 이벤트가 완료되거나 제거될 때 세이브포인트 레벨이 종료합니다. 세이브포인트 레벨이 종료할 때 그 안에 들어있는 모든 세이브포인트가 해제됩니다. 모든 열린 커서, DDL 조치 또는 데이터 수정이 상위 세이브포인트 레벨(즉, 방금 종료한 레벨이 작성된 세이브포인트 레벨)에 의해 상속되며, 상위 세이브포인트 레벨에 대해 발행되는 모든 세이브포인트 관련 명령문의 영향을 받습니다.
- 다음 규칙이 세이브포인트 레벨의 조치에 적용됩니다.
  - 세이브포인트는 세이브포인트가 설정된 세이브포인트 레벨 안에서만 참조될 수 있습니다. 현재 세이브포인트 레벨 밖에서 설정된 세이브포인트를 해제, 삭제 또는 롤백할 수 없습니다.
  - 현재 세이브포인트 레벨에서 설정된 모든 활성 세이브포인트는 세이브포인트 레벨이 종료할 때 자동으로 해제됩니다.
  - 세이브포인트 이름의 고유성은 현재 세이브포인트 레벨 안에서만 강제 실행됩니다. 다른 세이브포인트 레벨에서 활성인 세이브포인트의 이름은 다른 세이브포인트 레벨의 세이브포인트에 영향을 주지 않고 현재 세이브포인트 레벨에서 재사용할 수 있습니다.

주

- SAVEPOINT문이 발행된 후 별칭에 대한 삽입, 갱신 또는 삭제 조작은 허용되지 않습니다.
- UNIQUE절을 생략하면 *savepoint-name*이 다른 세이브포인트에 의해 세이브포인트 레벨 안에서 재사용될 수 있음을 지정합니다. 동일한 이름의 세이브포인트가 세이브포인트 레벨에 이미 존재하는 경우 기존 세이브포인트는 삭제되며 동일한 이름을 갖는 새 세이브포인트가 처리 중인 현재 위치에서 작성됩니다. 새 세이브포인트는 응용프로그램이 설정하는 마지막 세이브포인트인 것으로 간주됩니다. 다른 세이브포인트가 이름을 재사용함으로써 세이브포인트를 삭제하면 단순히 해당되는 하나의 세이브포인트를 삭제하며 삭제된 세이브포인트 후에 설정된 모든 세이브포인트는 해제하지 않습니다. 이들 후속 세이브포인트는 RELEASE SAVEPOINT문을 사용해서만 해제할 수 있으며, 이는 이름 지정된 세이브포인트 및 이름 지정된 세이브포인트 후에 설정된 모든 세이브포인트를 해제합니다.
- UNIQUE절이 지정되는 경우 *savepoint-name*은 동일한 이름을 갖는 기존 세이브포인트가 해제된 후에만 재사용할 수 있습니다.

- 세이브포인트 안에서 유틸리티, SQL문 또는 DB2 명령이 처리 중에 중간 커미트를 수행하는 경우 세이브포인트는 내제적으로 해제됩니다.
- SET INTEGRITY문이 세이브포인트 안에서 롤백되는 경우, 동적으로 준비된 명령문 이름은 명령문이 내재적으로 다시 준비될 수 있지만 여전히 유효합니다.
- 삽입이 버퍼되는 경우(즉, 응용프로그램이 INSERT BUF 옵션으로 사전 컴파일된 경우), SAVEPOINT, ROLLBACK 또는 RELEASE TO SAVEPOINT문이 발행될 때 버퍼가 비워집니다.

예 :

예 1: 중첩된 세이브포인트에 대한 롤백 조작을 수행하십시오. 먼저 DEPARTMENT 테이블을 작성하십시오. SAVEPOINT1을 시작하기 전에 행을 삽입하십시오. 다른 행을 삽입하고 SAVEPOINT2를 시작한 후 세 번째 행을 삽입하고 SAVEPOINT3을 시작하십시오.

```
CREATE TABLE DEPARTMENT (
  DEPTNO   CHAR(6),
  DEPTNAME VARCHAR(20),
  MGRNO    INTEGER)

  INSERT INTO DEPARTMENT VALUES ('A20', 'MARKETING', 301)

  SAVEPOINT SAVEPOINT1 ON ROLLBACK RETAIN CURSORS

  INSERT INTO DEPARTMENT VALUES ('B30', 'FINANCE', 520)

  SAVEPOINT SAVEPOINT2 ON ROLLBACK RETAIN CURSORS

  INSERT INTO DEPARTMENT VALUES ('C40', 'IT SUPPORT', 430)

  SAVEPOINT SAVEPOINT3 ON ROLLBACK RETAIN CURSORS

  INSERT INTO DEPARTMENT VALUES ('R50', 'RESEARCH', 150)
```

이 시점에서 DEPARTMENT 테이블이 A20, B30, C40 및 R50 행을 포함하고 존재합니다. 이제

```
ROLLBACK TO SAVEPOINT SAVEPOINT3
```

을 발행하는 경우 R50 행이 더 이상 DEPARTMENT 테이블에 없습니다. 그런 다음

```
ROLLBACK TO SAVEPOINT SAVEPOINT1
```

을 발행하면 DEPARTMENT 테이블은 존재하지만, SAVEPOINT1이 설정된 후에 삽입된 행(B30 및 C40)은 더 이상 테이블에 존재하지 않습니다.

SELECT

SELECT문은 쿼리 양식입니다. 응용프로그램에 임베디드(embedded)하거나 대화식으로 발행할 수 있습니다.

SELECT INTO

SELECT INTO문은 최대 한 행으로 구성되는 결과 테이블을 작성한 후, 해당 행의 값을 호스트 변수에 지정합니다. 테이블이 공백인 경우 명령문은 +100을 SQLCODE에 지정하고, '02000'을 SQLSTATE에 지정하며, 호스트 변수에는 값을 지정하지 않습니다. 하나 이상의 행이 검색 조건을 충족시킬 경우 명령문 처리는 종료되고 오류가 발생합니다(SQLSTATE 21000).

호출

이 명령문은 응용프로그램에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 테이블, 뷰 또는 별칭에 대한 SELECT 특권
- 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- DATAACCESS 권한

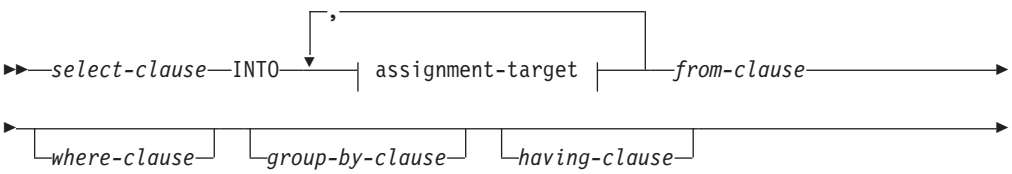
지정 목표로 사용되는 각 전역 변수의 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

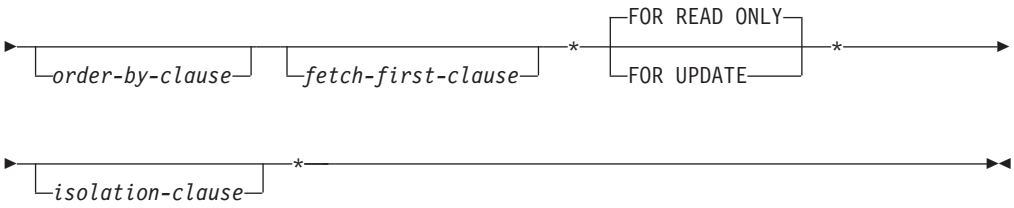
- 모듈에 정의되지 않은 전역 변수에 대한 WRITE 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

SELECT INTO문에 대해서는 GROUP 특권이 점검되지 않습니다.

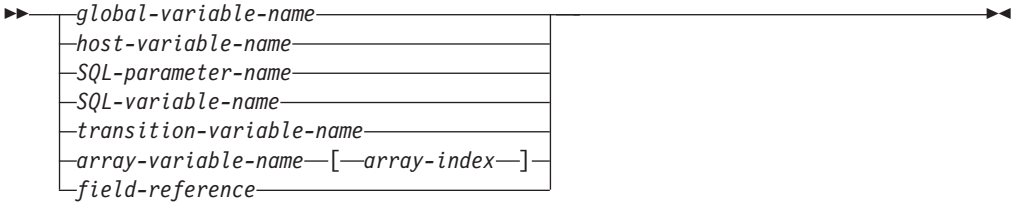
SELECT INTO문의 목표가 별칭일 경우에는 데이터 소스에서 명령문을 실행하기 전에는 데이터 소스에서 오브젝트에 대한 특권은 무시됩니다. 이 때 데이터 소스에 연결하는 데 사용되는 권한 부여 ID에는 데이터 소스에서 해당 오브젝트에 조작을 수행하는 데 필요한 특권이 있어야 합니다. 명령문의 권한 부여 ID는 데이터 소스에서 서로 다른 권한 부여 ID로 맵핑될 수 있습니다.

구문





assignment-target



설명

select-clause, *from-clause*, *where-clause*, *group-by-clause*, *having-clause*, *order-by-clause*, *fetch-first-clause* 및 *isolation-clause*에 대한 자세한 내용은 *SQL 참조서, 볼륨 1*의 『Queries』를 참조하십시오.

INTO *assignment-target*

출력 값을 지정할 하나 이상의 목표를 식별합니다.

결과 행의 첫 번째 값은 목록의 첫 번째 목표에 지정되고, 두 번째 값은 두 번째 목표에, 이와 같은 식으로 지정됩니다. 목록에 나타난 순서대로 *assignment-target*에 값이 지정됩니다. 지정 오류가 발생하면, 값이 *assignment-target*에 지정되지 않습니다.

모든 *assignment-target*의 데이터 유형이 행 유형이 아닌 경우, *assignment-targets* 수가 결과 컬럼 값보다 작으면 'W' 값이 SQLCA의 SQLWARN3 필드에 지정됩니다.

*assignment-target*의 데이터 유형이 행 유형이면, 하나의 *assignment-target*가 명확히 지정되어 있어야 하며(SQLSTATE 428HR), 컬럼 수는 행 유형의 필드 수와 일치되어야 하며, 페치된 행 컬럼의 데이터 유형이 행 유형의 해당 필드에 지정 가능해야 합니다(SQLSTATE 42821).

*assignment-target*의 데이터 유형이 배열 요소이면, 정확히 지정된 하나의 *assignment-target*이 있어야 합니다.

global-variable-name

지정 목표인 전역 변수를 식별합니다.

host-variable-name
지정 목표인 호스트 변수를 식별합니다. LOB 출력 값의 경우, 목표는 일반 호스트 변수(충분히 큰 경우), LOB 로케이터 변수 또는 LOB 파일 참조 변수가 될 수 있습니다.

SQL-parameter-name
지정 목표인 매개변수를 식별합니다.

SQL-variable-name
지정 목표인 SQL 변수를 식별합니다. SQL 변수는 사용하기 전에 선언해야 합니다.

transition-variable-name
전이 행에서 갱신될 컬럼을 식별합니다. *transition-variable-name*은 새로운 값을 식별하는 상관 이름에 의해 선택적으로 규정되는 트리거의 주제 테이블에 있는 컬럼을 식별해야 합니다.

array-variable-name
배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다.

[array-index]
지정 목표가 되는 배열에서의 요소를 지정하는 표현식. 일반 배열의 경우, *array-index* 표현식은 INTEGER(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다. 배열에 정의된 1과 최대 카디널리티 사이의 값이어야 합니다(SQLSTATE 2202E). 연관된 배열의 경우, *array-index* 표현식은 연관된 배열의 인덱스 데이터 유형(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다.

field-reference
지정 목표인 행 유형 값 내에서 필드를 식별합니다. *field-reference*는 규정된 *field-name*으로 지정되어야 하며 이때 규정자는 필드가 정의된 행 값을 식별합니다.

FOR READ ONLY 또는 FOR UPDATE
선택된 행의 의도된 사용을 표시합니다. 디폴트값은 FOR READ ONLY입니다.

FOR READ ONLY
선택된 행이 갱신을 위해 잠겨 있지 않도록 지정합니다.

FOR UPDATE
기본 테이블에서 선택된 행이 트랜잭션에서 나중에 행을 갱신할 수 있게 잠겨 있도록 지정하며, 마찬가지로 FOR UPDATE절을 포함한 커서의 선택문에 대해 잠금이 수행됩니다.

SELECT INTO문의 결과 테이블이 읽기 전용이면 FOR UPDATE는 지정되지 않아야 합니다(SQLSTATE 42829).

규칙

- 전역 변수는 트리거, 함수, 메소드 또는 복합 SQL(인라인된)문, 또는 이러한 오브젝트 중 하나에서 직접적 또는 간접적으로 호출되는 프로시저 내에서 지정할 수 없습니다(SQLSTATE 428GX).

주

- **호환성:** SQL 쿼리와의 일관성을 위해:
  - FOR FETCH ONLY는 FOR READ ONLY 대신에 지정될 수 있습니다.

예:

예 1: 이 C 예는 EMP 테이블의 최대 급여를 호스트 변수 MAXSALARY에 둡니다.

```
 EXEC SQL SELECT MAX(SALARY)
INTO :MAXSALARY
FROM EMP;
```

예 2: 이 C 예는 EMP 테이블에서 사원 528671에 대한 행을 호스트 변수에 둡니다.

```
 EXEC SQL SELECT * INTO :h1, :h2, :h3, :h4
FROM EMP
WHERE EMPNO = '528671';
```

예 3: 이 SQLJ 예는 EMP 테이블에서 사원 528671에 대한 행을 호스트 변수에 둡니다. 이 행은 검색 갱신을 사용하여 나중에 갱신되며 쿼리가 실행될 때 잠금되어야 합니다.

```
#sql { SELECT * INTO :FIRSTNAME, :LASTNAME, :EMPNO, :SALARY
  FROM EMP
  WHERE EMPNO = '528671'
  FOR UPDATE };
```

예 4: 이 C 예는 EMP 테이블의 최대 급여를 전역 변수 GV_MAXSALARY에 둡니다.

```
 EXEC SQL SELECT MAX(SALARY)
INTO GV_MAXSALARY
FROM EMP;
```

SET COMPILATION ENVIRONMENT

SET COMPILATION ENVIRONMENT문은 이벤트 모니터가 제공하는 컴파일 환경에 포함된 값과 일치하도록 연결에 있는 현재 컴파일 환경을 변경합니다. 이 명령문은 하나 이상의 특수 레지스터의 값을 변경합니다. 이러한 변경이 다시 모든 후속 동적 SQL 문의 컴파일에 영향을 미칩니다.

이 명령문은 트랜잭션의 제어를 받습니다.

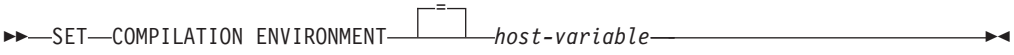
호출

이 명령문은 응용프로그램에 임베디드(embedded)될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

host-variable

이벤트 모니터가 제공하는 컴파일 환경이 들어있는 BLOB 유형의 변수입니다. 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815). 컴파일 환경의 형식이 올바르지 않은 경우 오류가 리턴되고 연결 설정은 수정되지 않습니다(SQLSTATE 51040).

주

- 컴파일 환경을 원래 디폴트값으로 재설정하려면 연결을 종료한 후 재시작하십시오. SQL 루틴 안에서 이 명령을 발행하여 해당 루틴에서의 리턴 시 특수 레지스터 변경 사항이 연결에 반영되지 않도록 함으로써 동일한 효과를 달성할 수 있습니다.
- 컴파일 환경에 들어있는 개별 요소를 보려면 COMPILATION_ENV 테이블 함수를 사용하십시오.

예:

예 1: 현재 세션의 컴파일 환경을 교착 상태 이벤트 모니터가 이전에 캡처한 컴파일 환경에 포함된 값으로 설정하십시오. WITH DETAILS HISTORY 옵션을 지정하여 작성되는 교착 상태 이벤트 모니터는 동적 SQL문에 대한 컴파일 환경을 캡처합니다. 이 캡처된 환경이 명령문에 대한 입력으로 승인되는 환경입니다.

```
SET COMPILATION ENVIRONMENT = :hv1
```

SET CONNECTION

SET CONNECTION문은 연결 상태를 유휴에서 현재로 변경하여 지정된 위치를 현재 서버로 지정합니다. 이는 트랜잭션의 제어를 받지 않습니다.

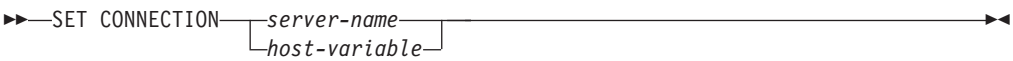
호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

server-name 또는 *host-variable*

*server-name*이 들어 있는 *host-variable* 또는 지정된 *server-name*으로 응용프로그램 서버(AS)를 식별합니다.

*host-variable*을 지정할 경우 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안됩니다. *host-variable* 내에 들어 있는 *server-name*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안됩니다.

*server-name*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 로컬 디렉토리에 나열되어야 합니다.

server-name 또는 *host-variable*은 응용프로그램 프로세스의 기존 연결을 식별해야 합니다. 기존 연결을 식별하지 않는 경우 오류(SQLSTATE 08003)가 발생합니다.

SET CONNECTION이 현재 연결인 경우 응용프로그램 프로세스의 모든 연결의 상태가 변경되지 않습니다.

성공적인 연결

SET CONNECTION문이 성공적으로 실행하는 경우:

- 연결이 작성되지 않습니다. CURRENT SERVER 특수 레지스터가 지정된 *server-name*으로 갱신됩니다.
- 이전의 현재 연결(있는 경우)이 유휴 상태로 바뀝니다(다른 *server-name*이 지정되는 경우).

- CURRENT SERVER 특수 레지스터 및 SQLCA가 『CONNECT(유형 1)』에서 문서화된 것과 같은 방법으로 갱신됩니다.

실패한 연결

SET CONNECTION문이 실패하는 경우:

- 실패한 이유가 무엇이든지 응용프로그램 프로세스의 연결 상태 및 연결 상태는 변경되지 않습니다.
- 실패한 유형 1 CONNECT에서와 마찬가지로, SQLCA의 SQLERRP 필드는 오류를 발견한 모듈의 이름으로 설정됩니다.

주

- 유휴 연결이 존재할 수 없기 때문에 SET CONNECTION문이 현재 연결을 지정하지 않는 경우, 유형 1 CONNECT문의 사용이 SET CONNECTION의 사용을 배제하지 않지만 명령문이 항상 실패합니다(SQLSTATE 08003).
- SQLRULES(DB2) 연결 옵션(『분산 작업 단위(DUOW) 시맨틱 제어 옵션』 참조)은 SET CONNECTION을 사용하지 못하게 하지는 않지만, 유형 2 CONNECT문을 대신 사용할 수 있으므로 SET CONNECTION은 필요하지 않습니다.
- 연결이 사용되고 유휴이 된 후 동일한 작업 단위(UOW)에서 현재 상태로 리스토어된 경우, 해당 연결은 잠금, 커서 및 준비된 명령문의 상태에 관하여 응용프로그램 프로세스의 최종 사용을 반영합니다.

예:

IBMSTHDB에서 SQL문을 실행하고, IBMTOKDB에서 SQL문을 실행한 후, IBMSTHDB에서 추가 SQL문을 실행하십시오.

```
EXEC SQL CONNECT TO IBMSTHDB;
/* Execute statements referencing objects at IBMSTHDB */

EXEC SQL CONNECT TO IBMTOKDB;
/* Execute statements referencing objects at IBMTOKDB */

EXEC SQL SET CONNECTION IBMSTHDB;
/* Execute statements referencing objects at IBMSTHDB */
```

첫 번째 CONNECT문은 IBMSTHDB 연결을 작성하고 두 번째 CONNECT문은 연결을 유휴 상태에 두고, SET CONNECTION문은 현재 상태로 바꿉니다.

SET CURRENT DECFLOAT ROUNDING MODE

SET CURRENT DECFLOAT ROUNDING MODE문은 지정된 근사값 모드가 현재 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터에 설정된 값인지 검증합니다.

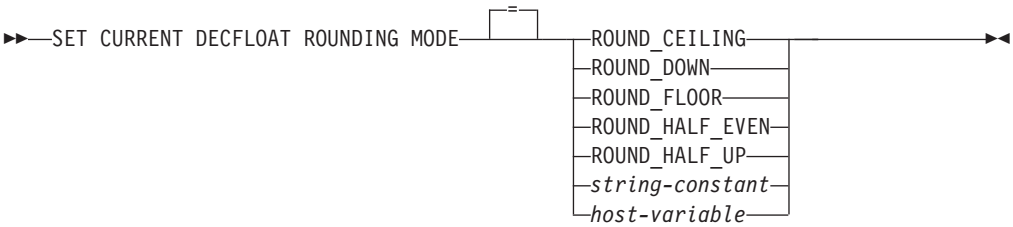
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

ROUND_CEILING

값을 양의 무한대 방향으로 라운드합니다. 버려지는 모든 숫자가 영(0)이거나 부호가 음수이면 결과는 변경되지 않습니다(버려지는 숫자 제거를 제외하고). 그렇지 않은 경우, 결과 계수는 1씩 증가합니다.

ROUND_DOWN

값을 0 방향으로 라운드합니다(절단). 버려지는 숫자는 무시됩니다.

ROUND_FLOOR

값을 음의 무한대 방향으로 라운드합니다. 버려지는 모든 숫자가 영(0)이거나 부호가 양수이면 결과는 변경되지 않습니다(버려지는 숫자 제거를 제외하고). 그렇지 않은 경우, 부호는 음수이고 결과 계수는 1씩 증가합니다.

ROUND_HALF_EVEN

값을 가장 가까운 값으로 라운드합니다. 값이 등거리이면 최종 숫자가 짝수가 되도록 값을 라운드합니다. 버려지는 숫자가 다음 왼쪽 위치에 있는 숫자 값의 반보다 큰 숫자를 나타내는 경우, 결과 계수는 1씩 증가합니다. 반보다 작은 숫자를 나타내면 결과 계수는 조정되지 않습니다(즉, 버려지는 숫자는 무시됨). 그렇지 않으면 결과 계수는 가장 오른쪽 숫자가 짝수인 경우 변경되지 않고 홀수이면 1씩 증가합니다(짝수 숫자를 만들기 위해).

ROUND_HALF_UP

값을 가장 가까운 값으로 라운드합니다. 값이 등거리이면 값을 위로 라운드합니다. 버려지는 숫자가 다음 왼쪽 위치에 있는 숫자 값의 반이거나 반보다 큰 숫자를 나타내는 경우, 결과 계수는 1씩 증가합니다. 그렇지 않으면 버려지는 숫자는 무시됩니다.

string-constant

뒤 공백을 제거한 후 최대 15바이트 길이의 문자열 상수입니다. 값은 5개의 근사값 모드 키워드 중 하나를 지정하는 왼쪽 맞춤 문자열이어야 합니다(대소문자 구분 안함).

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. 호스트 변수의 값은 5개의 근사값 모드 키워드 중 하나를 지정하는 왼쪽 맞춤 문자열이어야 합니다(대소문자 구분 안함). *host-variable* 컨텐츠의 실제 길이는 뒤 공백이 제거된 후 15바이트를 초과하면 안됩니다. 값은 고정 길이 문자 호스트 변수 사용 시 오른쪽을 공백으로 채워야 합니다. 호스트 변수는 널(NULL) 값으로 설정될 수 없습니다.

규칙

- 지정된 근사값 모드 값은 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터에 설정된 값과 같아야 합니다(SQLSTATE 42815).

주

- 이 명령문은 Linux, UNIX 및 Windows 서버용 DB2에서 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터 값을 변경하지 않습니다. 그러나 명령문이 z/OS 서버용 DB2 또는 System i 서버용 DB2에서 처리되는 경우, 이 명령문을 사용하여 해당 서버에서 CURRENT DECFLOAT ROUNDING MODE 특수 레지스터의 값을 변경할 수 있습니다.

예 :

예 1: 다음 명령문은 클라이언트에 대해 지정된 근사값 모드 값이 현재 서버에 설정된 근사값 모드 값과 일치하는지 검증합니다.

```
SET CURRENT DECFLOAT ROUNDING MODE = ROUND_CEILING
```

SET CURRENT DEFAULT TRANSFORM GROUP

SET CURRENT DEFAULT TRANSFORM GROUP문은 CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터값을 변경합니다. 이 명령문은 트랜잭션의 제어를 받습니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문

설명

group-name

모든 구조화된 유형에 대해 정의된 변환 그룹을 식별하는 한 부분 이름을 지정합니다. 이 이름은 후속 명령문에서 참조할 수 있습니다(또는 다른 SET CURRENT DEFAULT TRANSFORM GROUP문을 사용하여 다시 특수 레지스터값을 변경할 때까지).

이름은 최대 길이가 128바이트인 SQL ID이어야 합니다(SQLSTATE 42815). 특수 레지스터가 설정될 때까지는 모든 구조화된 유형의 *group-name*이 정의되었는지에 대한 유효성 검사가 수행되지 않습니다. 구조화된 유형이 특별히 참조되는 경우에만 유효성 검사를 위해 이름 지정된 변환 그룹 정의가 확인됩니다.

규칙

- 지정 값이 *group-name*에 대한 규칙을 지키지 않으면 오류가 발생합니다(SQLSTATE 42815).
- *group-name* 변환 그룹에 정의된 TO SQL 및 FROM SQL 함수는 사용자 정의 구조화된 유형을 호스트 프로그램과 교환하는 데 사용됩니다.

주

- CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터의 초기값은 빈 문자열입니다.

예:

예 1: 디폴트 변환 그룹을 MYSTRUCT1으로 설정하십시오. MYSTRUCT1 변환 그룹에 정의된 TO SQL 및 FROM SQL 함수는 사용자 정의 구조화된 유형 변수를 현재 호스트 프로그램과 교환하는 데 사용됩니다.

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

SET CURRENT DEGREE

SET CURRENT DEGREE문은 값을 CURRENT DEGREE 특수 레지스터에 지정합니다. 이 명령문은 트랜잭션의 제어를 받습니다.

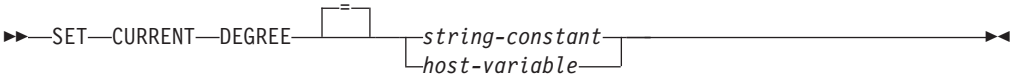
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

CURRENT DEGREE의 값이 문자열 상수 또는 호스트 변수의 값으로 교체됩니다. 값은 5바이트보다 길지 않은 문자열이어야 합니다. 값은 1부터 32,767까지의 정수의 문자열 표시 또는 'ANY'여야 합니다.

SQL문이 동적으로 준비될 때 정수로 표시되는 CURRENT DEGREE의 값이 1인 경우, 해당 명령문 실행 시 파티션 내 병렬 처리를 사용하지 않습니다.

SQL문이 동적으로 준비될 때 CURRENT DEGREE의 값이 숫자인 경우, 해당 명령문의 실행 시 지정된 등급을 갖는 파티션 내 병렬 처리가 포함될 수 있습니다.

SQL문이 동적으로 준비될 때 CURRENT DEGREE의 값이 'ANY'인 경우, 해당 명령문의 실행 시 데이터베이스 관리 프로그램이 판별하는 등급을 사용하는 파티션 내 병렬 처리가 포함될 수 있습니다.

host-variable

*host-variable*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하고 길이는 5자를 초과할 수 없습니다. 더 긴 필드가 제공되면 오류가 발생합니다(SQLSTATE 42815). 제공된 실제 값이 지정된 대체 값보다 크면 공백을 입력하여 오른쪽을 채워야 합니다. 앞 공백은 허용되지 않습니다(SQLSTATE 42815). 모든 입력값은 대소문자를 구별합니다. *host-variable*에 관련된 표시기 변수가 있는 경우 해당 표시기 변수의 값이 널(NULL) 값을 표시해서는 안됩니다(SQLSTATE 42815).

string-constant

string-constant 길이는 5를 초과하지 않아야 합니다.

주

정적 SQL문의 파티션 내 병렬 처리 수준은 PREP 또는 BIND 명령의 DEGREE 옵션을 사용하여 제어할 수 있습니다.

파티션 내 병렬 처리의 실제 런타임 등급은 다음 중 낮은 것입니다.

- 최대 쿼리 등급(**max_querydegree**) 구성 매개변수
- 응용프로그램 런타임 등급
- SQL문 컴파일 등급

파티션 내 병렬 처리를 사용하려면 **intra_parallel** 데이터베이스 관리 프로그램 구성 매개변수가 설정되어야 합니다. Off로 설정되는 경우 이 레지스터의 값은 무시되며 명령문은 최적화를 위해 파티션 내 병렬 처리를 사용하지 않습니다(SQLSTATE 01623).

일부 SQL문은 파티션 내 병렬 처리를 사용할 수 없습니다.

예 :

예 1: 다음 명령문은 파티션 내 병렬 처리를 금지하도록 CURRENT DEGREE를 설정합니다.

```
SET CURRENT DEGREE = '1'
```

예 2: 다음 명령문은 파티션 내 병렬 처리를 허용하도록 CURRENT DEGREE를 설정합니다.

```
SET CURRENT DEGREE = 'ANY'
```

SET CURRENT EXPLAIN MODE

SET CURRENT EXPLAIN MODE문은 CURRENT EXPLAIN MODE 특수 레지스터값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

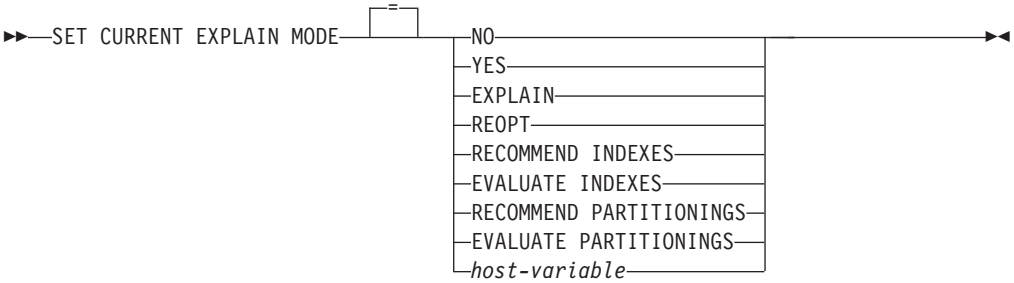
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

NO

'Explain' 기능을 사용 불가능하게 설정합니다. 캡처된 Explain 정보가 없습니다. NO는 특수 레지스터의 초기값입니다.

YES

Explain 기능을 사용 가능하게 설정하고 Explain 정보가 적합한 동적 SQL문에 대한 Explain 테이블에 삽입되도록 합니다. 모든 동적 SQL문이 컴파일되고 정상적으로 실행됩니다.

EXPLAIN

Explain 기능을 사용 가능하게 설정하고 Explain 정보가 준비된 동적 SQL문에 대해 캡처되도록 합니다. 그러나 동적 명령문은 실행되지 않습니다.

REOPT

명령문이 실행 시에 다시 최적화되는 동안 Explain 기능을 사용 가능하게 하고 Explain 정보가 정적 또는 동적 SQL문에 대해 캡처됩니다. 즉 호스트 변수, 특수 레지스터, 전역 변수 또는 매개변수 표시문자에 대한 실제 값이 사용 가능할 때입니다.

RECOMMEND INDEXES

SQL 컴파일러가 인덱스를 권장할 수 있게 합니다. 이 설명 모드에서 실행되는 모든 쿼리는 ADVISE_INDEX 테이블을 권장되는 인덱스로 채웁니다. 또한 권장되는 인덱스가 어떻게 사용되는지를 보이기 위해 설명 테이블에서 설명 정보가 캡처되지만, 명령문은 컴파일되지도 실행되지도 않습니다.

EVALUATE INDEXES

SQL 컴파일러가 인덱스를 평가할 수 있게 합니다. 평가될 인덱스를 ADVISE_INDEX 테이블에서 읽어 EVALUATE = Y로 표시해야 합니다. 옵티마이저는 카탈로그로의 값에 기초하여 가상 인덱스를 생성합니다. 이 설명 모드에서 실행되는 모든 쿼리는 가상 인덱스에 기초하여 예측된 통계를 사용하여 컴파일되고 최적화될 것입니다. 명령문은 실행되지 않습니다.

RECOMMEND PARTITIONINGS

컴파일러가 특정 쿼리에 의해 액세스되는 각 테이블에 대해 가장 적합한 데이터베이스 파티션을 권장하도록 지정합니다. 가장 적합한 데이터베이스 파티션이 ADVISE_PARTITION 테이블에 쓰여집니다. 쿼리가 실행되지 않습니다.

EVALUATE PARTITIONINGS

컴파일러가 ADVISE_PARTITION 테이블에 지정된 가상 데이터베이스 파티션을 사용하여 쿼리의 예상 성능을 획득하도록 지정합니다.

host-variable

*host-variable*은 데이터 유형이 CHAR 또는 VARCHAR여야 하고 길이는 254자를 초과할 수 없습니다. 이보다 긴 필드가 지정되면 오류가 발생합니다(SQLSTATE 42815). 지정된 값은 NO, YES, EXPLAIN, RECOMMEND INDEXES 또는 EVALUATE INDEXES이어야 합니다. 제공된 실제 값이 지정된 대체 값보다 크면 공백을 입력하여 오른쪽을 채워야 합니다. 앞 공백은 허용되지 않습니다(SQLSTATE 42815). 모든 입력값은 대소문자를 구별합니다. *host-variable*에 관련된 표시기 변수가 있는 경우 해당 표시기 변수의 값이 널(NULL) 값을 표시해서는 안됩니다(SQLSTATE 42815).

주

- Explain 기능은 데이터를 입력 중인 Explain 테이블을 규정할 때 다음과 같은 ID를 사용합니다.
  - 동적 SQL의 경우 세션 권한 부여 ID
  - 정적 SQL의 경우 명령문 권한 부여 ID

  서로 다른 스키마 아래 있는 Explain 테이블 세트 또는 Explain 테이블 세트를 가리키는 별명과 스키마를 연관시킬 수 있습니다. 만일 해당 스키마에 Explain 테이블이 없으면, Explain 기능은 SYSTOOLS 스키마의 Explain 테이블을 확인하고 그 테이블을 사용하려고 시도합니다.

- 정적 SQL문에 대한 Explain 정보는 PREP 또는 BIND 명령의 EXPLAIN 옵션을 사용하여 보관될 수 있습니다. EXPLAIN 옵션의 ALL 값이 지정되고 CURRENT EXPLAIN MODE 레지스터값이 NO이면 Explain 정보가 런타임 동적 SQL문에 의해 보관됩니다. CURRENT EXPLAIN MODE 레지스터의 값이 NO가 아니면 EXPLAIN 바인드 옵션 값이 무시됩니다.
- RECOMMEND INDEXES 및 EVALUATE INDEXES는 SET CURRENT EXPLAIN MODE문으로만 설정될 수 있는 특수 모드입니다. 이들 모드는 PREP나 BIND 옵션을 사용하여 설정할수 없으며 SET CURRENT EXPLAIN SNAPSHOT 문과 작동하지 않습니다.
- Explain 기능이 활성화되면 현재 권한 부여 ID에 Explain 테이블에 대한 INSERT 특권이 있어야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42501).
- SQL문이 루틴에서 설명되는 경우에는 SQL 데이터 액세스 표시기 MODIFIES SQL DATA로 루틴이 정의되어 있어야 합니다(SQLSTATE 42985).
- 특수 레지스터가 REOPT으로 설정되고 SQL문이 실행시간의 재최적화에 해당되지 않을 경우(명령문에 입력 변수가 없거나 REOPT 바인드 옵션이 NONE으로 설정된 경우) Explain 정보는 캡처되지 않습니다. REOPT 바인드 옵션을 ONCE로 설정하면 Explain 정보는 명령문이 초기에 재최적화될 때 한 번만 캡처됩니다. 명령문이 캐시된 후에는 다음 실행에서 이 명령문에 대한 Explain 정보는 더 필요하지 않습니다.
- Explain 기능을 사용할 수 있고 REOPT 바인드 옵션이 ONCE로 설정되어 있으며 이미 캐시된 SQL문을 실행하려는 시도를 할 경우 명령문은 입력 변수의 현재 값으로 컴파일되고 재최적화되며 Explain 테이블은 이에 따라 데이터로 채워집니다. 이 명령문에 대해 새로 생성된 액세스 플랜은 캐시되거나 실행되지 않습니다. 이 캐시된 명령문을 동시에 실행 중인 다른 응용프로그램은 계속 실행되며 이 명령문을 실행하는 새 요청은 이미 캐시된 액세스 플랜을 가져갑니다.
- CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터에 대한 REOPT 값은 정적 또는 동적 SQL문에 입력 변수가 있고 REOPT 바인드 옵션이 ONCE 또는 ALWAYS로 설정된 경우 바인드 시간에 EXPLAIN 및 EXPLSNAP 바인드 옵션 값을 겹쳐씁니다.

예 :

다음 명령문은 적절한 후속 동적 SQL문에 대한 Explain 정보가 캡처되고 명령문이 실행되지 않도록 CURRENT EXPLAIN MODE 특수 레지스터를 설정하십시오.

```
SET CURRENT EXPLAIN MODE = EXPLAIN
```

SET CURRENT EXPLAIN SNAPSHOT

SET CURRENT EXPLAIN SNAPSHOT문은 CURRENT EXPLAIN SNAPSHOT 특수 레지스터값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

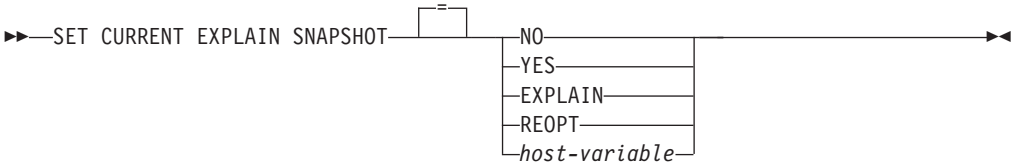
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

NO

Explain 스냅샷 기능을 사용 불가능하게 설정합니다. 스냅샷이 구해지지 않았습니다. NO는 특수 레지스터의 초기값입니다.

YES

Explain 스냅샷 기능을 사용 가능하게 설정하고 적합한 각 동적 SQL문에 대한 내부 표현의 스냅샷을 작성합니다. 이 정보는 EXPLAIN_STATEMENT 테이블의 SNAPSHOT 컬럼에 삽입됩니다.

EXPLAIN SNAPSHOT 기능은 Visual Explain에 사용하기 위한 것입니다.

EXPLAIN

Explain 스냅샷 기능을 사용 가능하게 설정하고 적합한 각 동적 SQL문에 대한 내부 표현의 스냅샷을 작성합니다. 그러나 동적 명령문은 실행되지 않습니다.

REOPT

명령문이 실행 시에 다시 최적화되는 동안 Explain 기능을 사용 가능하게 하고 Explain 정보가 정적 또는 동적 SQL문에 대해 캡처됩니다. 즉 호스트 변수, 특수 레지스터, 전역 변수 또는 매개변수 표시문자에 대한 실제 값이 사용 가능할 때입니다.

host-variable

*host-variable*은 데이터 유형이 CHAR 또는 VARCHAR여야 하고 컨텐츠 길이는

8자를 초과할 수 없습니다. 이보다 긴 필드가 지정되면 오류가 발생합니다 (SQLSTATE 42815). 이 레지스터에 들어 있는 값은 NO, YES 또는 EXPLAIN 이어야 합니다. 제공된 실제 값이 지정된 대체 값보다 크면 공백을 입력하여 오른쪽을 채워야 합니다. 앞 공백은 허용되지 않습니다(SQLSTATE 42815). 모든 입력값은 대소문자를 구별합니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

주

- Explain 기능은 데이터를 입력 중인 Explain 테이블을 규정할 때 다음과 같은 ID를 사용합니다.
  - 동적 SQL의 경우 세션 권한 부여 ID
  - 정적 SQL의 경우 명령문 권한 부여 ID

  서로 다른 스키마 아래 있는 Explain 테이블 세트 또는 Explain 테이블 세트를 가리키는 별명과 스키마를 연관시킬 수 있습니다. 만일 해당 스키마에 Explain 테이블이 없으면, Explain 기능은 SYSTOOLS 스키마의 Explain 테이블을 확인하고 그 테이블을 사용하려고 시도합니다.
- 정적 SQL문에 대한 Explain 스냅샷이 PREP 또는 BIND 명령의 EXPLSNAP 옵션을 사용하여 보관될 수 있습니다. EXPLSNAP 옵션의 ALL값이 지정되고 CURRENT EXPLAIN SNAPSHOT 레지스터값이 NO이면 Explain 스냅샷은 런타임 동적 SQL문에 대해 보관됩니다. CURRENT EXPLAIN SNAPSHOT 레지스터의 값이 NO가 아니면, EXPLSNAP 옵션이 무시됩니다.
- Explain 스냅샷 기능이 활성화되면, 현재 권한 부여 ID에 Explain 테이블에 대한 INSERT 특권이 있어야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42501).
- SQL문이 루틴에서 설명되는 경우에는 SQL 데이터 액세스 표시기 MODIFIES SQL DATA로 루틴이 정의되어 있어야 합니다(SQLSTATE 42985).
- 특수 레지스터가 REOPT으로 설정되고 SQL문이 실행시간의 재최적화에 해당되지 않을 경우(명령문에 입력 변수가 없거나 REOPT 바인드 옵션이 NONE으로 설정된 경우) Explain 정보는 캡처되지 않습니다. REOPT 바인드 옵션을 ONCE로 설정하면 Explain 스냅샷 정보는 명령문이 초기에 재최적화될 때 한 번만 캡처됩니다. 명령문이 캐시된 후에는 다음 실행에서 이 명령문에 대한 Explain 정보는 더 필요하지 않습니다.
- Explain 기능을 사용할 수 있고 REOPT 바인드 옵션이 ONCE로 설정되어 있으며 이미 캐시된 재최적화 가능 SQL문을 실행하려는 시도를 할 경우, 명령문은 입력 변수의 현재 값으로 컴파일되고 재최적화되며 Explain 스냅샷은 이에 따라 캡처됩니다. 이 명령문에 대해 새로 생성된 액세스 플랜은 캐시되거나 실행되지 않습니다. 이 캐시된 명령문을 동시에 실행 중인 다른 응용프로그램은 계속 실행되며 이 명령문을 실행하는 새 요청은 이미 캐시된 액세스 플랜을 가져갑니다.

- CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터에 대한 REOPT 값은 정적 또는 동적 SQL문에 입력 변수가 있고 REOPT 바인드 옵션이 ONCE 또는 ALWAYS로 설정된 경우 바인드 시간에 EXPLAIN 및 EXPLSNAP 바인드 옵션 값을 겹쳐씁니다.

예:

예 1: 다음 명령문은 CURRENT EXPLAIN SNAPSHOT 특수 레지스터를 설정하므로 후속하는 적절한 동적 SQL문에 대해 Explain 스냅샷이 캡처되고 명령문이 실행됩니다.

```
SET CURRENT EXPLAIN SNAPSHOT = YES
```

예 2: 다음 예는 CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 현재 값을 SNAP이라는 호스트 변수로 검색합니다.

```
EXEC SQL VALUES (CURRENT EXPLAIN SNAPSHOT) INTO :SNAP;
```

SET CURRENT FEDERATED ASYNCHRONY

SET CURRENT FEDERATED ASYNCHRONY문은 CURRENT FEDERATED ASYNCHRONY 특수 레지스터에 값을 지정합니다. 이는 트랜잭션의 제어를 받지 않습니다.

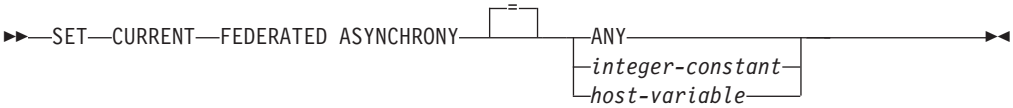
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

ANY

CURRENT FEDERATED ASYNCHRONY 값을 -1로 지정하십시오. 이는 명령문을 실행할 때 데이터베이스 관리 프로그램이 결정한 등급을 사용하여 비동시성을 포함할 수 있다는 것을 의미합니다.

integer-constant

0 - 32 767(0과 32,767 포함) 사이의 정수 값을 지정합니다. 명령문 실행시 지정된 등급을 사용하여 비동시성을 포함할 수 있습니다. SQL문이 동적으로 준비될 때 값이 0이면 이 명령문 실행시 비동시성을 사용하지 않습니다.

host-variable

유형 INTEGER의 변수입니다. 이 값은 0 - 32 767(0 및 32 767 포함) 사이이거나 -1(ANY를 나타냄)이어야 합니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

주

- 정적 SQL문에 대한 비동시성 등급은 PREP 또는 BIND 명령의 FEDERATED_ASYNCHRONY 옵션을 사용하여 제어할 수 있습니다.
- 명령행 처리기(CLP)를 통해 동적 명령문을 발행한 경우 CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 초기값은 **federated_async** 데이터베이스 관리 프

로그램 구성 매개변수에 의해 결정됩니다. 동적 명령문이 바인드 중인 응용프로그램에 포함된 경우 초기값은 FEDERATED_ASYNCHRONY 바인드 옵션에 의해 결정됩니다.

예:

예 1: 다음 명령문은 CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 값을 0으로 설정하여 비동시성을 사용할 수 없도록 합니다.

```
SET CURRENT FEDERATED ASYNCHRONY = 0
```

예 2: 다음 명령문은 비동시성 등급을 5로 설정합니다.

```
SET CURRENT FEDERATED ASYNCHRONY 5
```

예 3: 다음 명령문은 CURRENT FEDERATED ASYNCHRONY 특수 레지스터의 값을 -1로 설정하며, 이는 데이터베이스 관리 프로그램이 비동시성 등급을 결정하도록 지정합니다.

```
SET CURRENT FEDERATED ASYNCHRONY ANY
```

SET CURRENT IMPLICIT XMLPARSE OPTION

SET CURRENT IMPLICIT XMLPARSE OPTION문은 CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 값을 변경합니다. 이 명령문은 트랜잭션의 제어를 받습니다.

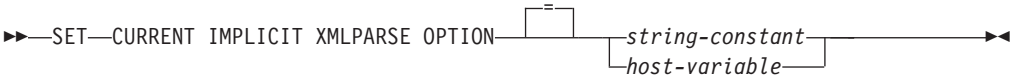
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

string-constant

문자열 상수입니다. 값은 키워드 사이에 추가 공백 문자가 없는 'PRESERVE WHITESPACE' 또는 'STRIP WHITESPACE'(대소문자 구분 안함)인 왼쪽으로 정렬된 문자열이어야 합니다.

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. 호스트 변수의 값은 키워드 사이에 추가 공백 문자가 없는 'PRESERVE WHITESPACE' 또는 'STRIP WHITESPACE'(대소문자 구분 안함)인 왼쪽으로 정렬된 문자열이어야 합니다. 고정 길이 문자 *host-variable*을 사용할 때 값의 오른쪽을 공백으로 채워야 합니다. 호스트 변수는 널(NULL)로 설정될 수 없습니다.

주

- CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 초기값은 'STRIP WHITESPACE'입니다.
- 동적 및 정적 SQL문이 둘 다 이 특수 레지스터의 영향을 받습니다.

예 :

CURRENT IMPLICIT XMLPARSE OPTION 특수 레지스터의 값을 'PRESERVE WHITESPACE'로 설정하십시오.

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

SET CURRENT ISOLATION

SET CURRENT ISOLATION문은 값을 CURRENT ISOLATION 특수 레지스터에 지정합니다. 이 명령문은 트랜잭션의 제어를 받습니다.

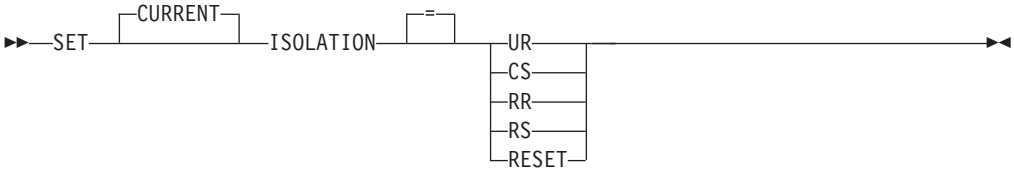
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

CURRENT ISOLATION 특수 레지스터의 값은 지정된 값으로 대체되거나 RESET이 지정된 경우 공백으로 설정됩니다.

주

- *호환성*: 다음 구문도 지원됩니다.
  - 같음 부호(=) 대신 TO를 지정할 수 있습니다.
  - UR 대신 DIRTY READ를 지정할 수 있습니다.
  - UR 대신 READ UNCOMMITTED를 지정할 수 있습니다.
  - READ COMMITTED를 인식하여 CS로 업그레이드합니다.
  - CS 대신 CURSOR STABILITY를 지정할 수 있습니다.
  - RR 대신 REPEATABLE READ를 지정할 수 있습니다.
  - RR 대신 SERIALIZABLE를 지정할 수 있습니다.

SET CURRENT LOCALE LC_TIME

SET CURRENT LOCALE LC_TIME문은 CURRENT LOCALE LC_TIME 특수 레지스터의 값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

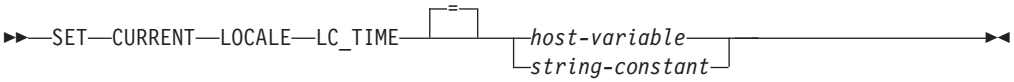
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

CURRENT LOCALE LC_TIME 특수 레지스터는 *locale-name* 인수가 명시적으로 지정되지 않은 경우 DAYNAME, MONTHNAME, NEXT_DAY, ROUND, ROUND_TIMESTAMP, TIMESTAMP_FORMAT, TRUNCATE, TRUNC_TIMESTAMP 및 VARCHAR_FORMAT 함수에서 사용됩니다.

host-variable
CHAR 또는 VARCHAR의 유형 변수입니다. 널(NULL)로 설정될 수 없습니다.

string-constant
문자열 상수입니다.

주

- CURRENT LOCALE LC_TIME 특수 레지스터의 초기값은 『en_US』입니다.

  주: 추후 릴리스에서는 CURRENT LOCALE LC_TIME 특수 레지스터의 값이 다른 스칼라 함수에서 사용되고 날짜 시간 값을 포함하는 데이터베이스 환경의 다른 영역에 사용될 수 있습니다.
- 유효한 로케일 및 이름 지정에 대해서는 *자국어 안내서*에서 『Locale names for SQL and XQuery』를 참조하십시오.

예:

- *예 1:* 다음 명령문은 CURRENT LOCALE LC_TIME 특수 레지스터를 DB2 데이터베이스 관리 프로그램에서 사용 가능한 CLDR(Common Locale Data Repository)의 최신 버전을 사용하는 영어(캐나다) 로케일로 설정합니다.

```
SET CURRENT LOCALE LC_TIME = 'en_CA'
```

- *예 2:* 다음 명령문은 CURRENT LOCALE LC_TIME 특수 레지스터를 CLDR(Common Locale Data Repository) 버전 1.5를 사용하는 프랑스어(프랑스) 로케일로 설정합니다. MONTHNAME 스칼라 함수는 '2008-11-10-00.00.00.000000'의 단일 인수로 호출됩니다.

```
SET CURRENT LOCALE LC_TIME = 'CLDR 1.5:fr_FR'
VALUES MONTHNAME( '2008-11-10-00.00.00.000000' )
```

다음을 리턴합니다.

```
'novembre'
```

SET CURRENT LOCK TIMEOUT

SET CURRENT LOCK TIMEOUT문은 CURRENT LOCK TIMEOUT 특수 레지스터의 값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

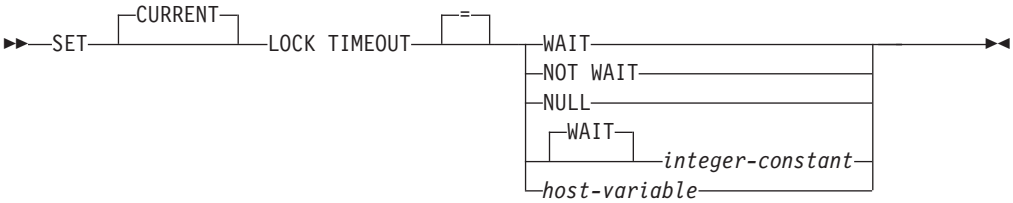
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

지정된 값은 -1과 32767 사이의 정수(SQLSTATE 428B7)이거나 널(NULL) 값이어야 합니다.

WAIT

CURRENT LOCK TIMEOUT 값을 -1로 지정합니다. 그러면, 데이터베이스 관리 프로그램이 잠금이 해제될 때까지 대기하거나 교착 상태가 발견됩니다(SQLSTATE 40001 또는 57033).

NOT WAIT

CURRENT LOCK TIMEOUT 값을 0으로 지정합니다. 그러면, 데이터베이스 관리 프로그램이 확보할 수 없는 잠금에 대해 대기하지 않으며 오류(SQLSTATE 40001 또는 57033)가 리턴됩니다.

NULL

CURRENT LOCK TIMEOUT 값을 지정하지 않으며 잠금에 대해 대기 중일 때 **locktimeout** 데이터베이스 구성 매개변수의 값을 사용할 것을 지정합니다. 특수 레지스터에 대해 리턴된 값은 **locktimeout** 값이 변경될 때 변경됩니다.

WAIT *integer-constant*

-1과 32767 사이의 정수 값을 지정합니다. -1 값은 정수 값없이 WAIT 키워드를 지정하는 것과 동일합니다. 0 값은 NOT WAIT절을 지정하는 것과 동일합니다. 값

이 1과 32767 사이에 있을 경우 데이터베이스 관리 프로그램은 오류(SQLSTATE 40001 또는47033)가 리턴되기 전에 그 값에 해당하는 시간(초) 동안 대기합니다 (잠금을 확보할 수 없는 경우).

host-variable

유형 INTEGER의 변수입니다. 이 값은 -1과 32767 사이여야 합니다. *host-variable* 에 연관된 표시기 변수가 있고 그 표시기 변수의 값이 널(NULL) 값을 지정할 경우 CURRENT LOCK TIMEOUT 값은 설정되지 않습니다. 이것은 널(NULL) 키워드를 지정하는 것과 같습니다.

주

- 특수 레지스터의 갱신된 값은 이 명령문의 성공적인 실행 후 즉시 적용됩니다. 명령문 실행 중에 사용될 특수 레지스터값이 명령문 실행의 시작 부분에서 고정되기 때문에 CURRENT LOCK TIMEOUT 특수 레지스터의 갱신된 값은 SET LOCK TIMEOUT문이 성공적으로 완료된 후에 실행을 시작한 명령문에 의해서만 리턴됩니다.
- *호환성*: Informix와의 호환성을 위해 다음이 지원됩니다.
  - TIMEOUT 대신 MODE를 지정할 수 있습니다.
  - 같음(=) 연산자 대신 TO를 지정할 수 있습니다.
  - SET CURRENT LOCK TIMEOUT WAIT 대신 SET LOCK WAIT를 지정할 수 있습니다.
  - SET CURRENT LOCK TIMEOUT NOT WAIT 대신 SET LOCK NO WAIT를 지정할 수 있습니다.

예:

예 1: 오류를 리턴하기 전에 30초를 대기하도록 잠금 시간종료 값을 설정하십시오.

```
SET CURRENT LOCK TIMEOUT 30
```

예 2: **locktimeout** 데이터베이스 구성 매개변수 값이 대신 사용되도록 잠금 시간종료 값을 설정하지 마십시오.

```
SET CURRENT LOCK TIMEOUT NULL
```

SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION문은 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터의 값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

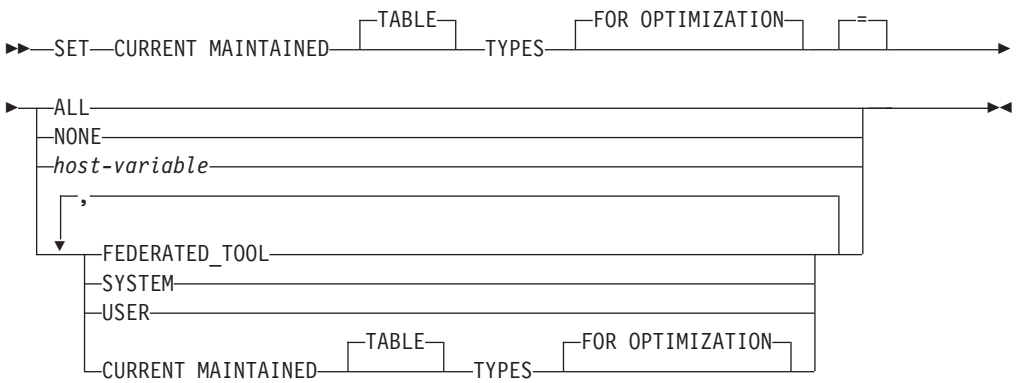
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

ALL

동적 SQL 쿼리 처리를 최적화할 때 이 특수 레지스터가 제어하는 유지보수된 테이블의 모든 가능한 유형을 현재와 미래에 고려하도록 지정합니다.

NONE

동적 SQL 쿼리 처리를 최적화할 때 이 특수 레지스터가 제어하는 어떤 유형의 오브젝트도 고려하지 않도록 지정합니다.

FEDERATED_TOOL

CURRENT QUERY OPTIMIZATION 특수 레지스터의 값이 2 또는 5보다 큰 경우 동적 SQL 쿼리의 처리를 최적화하기 위해 페더레이티드 도구가 유지보수하는 새로 고침 지연 구체화된 쿼리 테이블을 고려할 수 있도록 지정합니다.

SYSTEM
동적 SQL 쿼리의 처리를 최적화하기 위해 시스템이 유지보수하는 새로 고침 지연 구체화된 쿼리 테이블을 고려할 수 있도록 지정합니다. (즉시 구체화된 쿼리 테이블은 항상 사용 가능합니다.)

USER
동적 SQL 쿼리의 처리를 최적화하기 위해 사용자가 유지보수하는 새로 고침 지연 구체화된 쿼리 테이블을 고려할 수 있도록 지정합니다.

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
명령문을 실행하기 전 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터의 값입니다.

host-variable
CHAR 또는 VARCHAR의 유형 변수입니다. 호스트 변수의 컨텐츠 길이는 254 바이트를 초과할 수 없으며(SQLSTATE 42815) 널(NULL)로 설정될 수도 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

*host-variable*의 문자는 왼쪽 정렬되어야 합니다. *host-variable*의 컨텐츠는 특수 레지스터의 키워드로 지정할 수 있는 것과 일치하는 키워드의 쉼표로 구분된 목록인 문자열이어야 합니다. 대문자로 변환되지 않기 때문에 이 키워드는 원하는 대소문자로 정확하게 지정해야 합니다. 이 값의 길이가 호스트 변수보다 작을 때는 값 오른쪽에 공백을 채워야 합니다.

주

- CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터의 초기값은 SYSTEM입니다.
- CURRENT REFRESH AGE 특수 레지스터는 동적 SQL 쿼리의 처리를 최적화할 때 지정된 테이블 유형이 고려되도록 0이 아닌 값으로 설정되어야 합니다.

예:

예 1: CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터를 설정하십시오.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION SYSTEM = USER
```

예 2: CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터의 현재 값을 CURMAINTYPES라는 호스트 변수로 검색하십시오.

```
EXEC SQL VALUES (CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION)
  INTO :CURMAINTYPES
```

예 3: CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터를 값을 갖지 않도록 설정하십시오.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION = NONE
```

SET CURRENT MDC ROLLOUT MODE

SET CURRENT MDC ROLLOUT MODE문은 값을 CURRENT MDC ROLLOUT MODE 특수 레지스터에 지정합니다. 값은 다차원적으로 클러스터된(MDC) 테이블에 대해 규정하는 DELETE문에 수행될 롤아웃 정리 유형을 지정합니다.

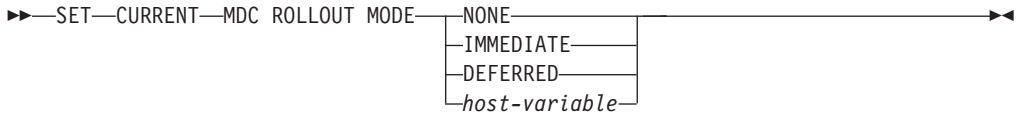
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

NONE

삭제 조작 중 MDC 롤아웃 최적화가 사용되지 않음을 지정합니다. DELETE문은 롤아웃에 대해 규정하지 않는 DELETE문과 같은 방식으로 처리됩니다.

IMMEDIATE

DELETE문이 규정하는 경우 MDC 롤아웃 최적화가 사용됨을 지정합니다. 테이블에 RID 인덱스가 있는 경우 인덱스는 삭제 처리 중에 즉시 갱신됩니다. 삭제된 블록은 트랜잭션 커미트 이후에 재사용할 수 있습니다.

DEFERRED

DELETE문이 규정하는 경우 MDC 롤아웃 최적화가 사용됨을 지정합니다. 테이블에 RID 인덱스가 있는 경우 인덱스 갱신사항은 트랜잭션이 커미트될 때까지 지연됩니다. 이 옵션을 사용하는 경우, 삭제 처리가 더 빨라지고 로그 공간도 적게 사용되지만 삭제된 블록은 인덱스 갱신사항이 완료될 때까지 다시 사용할 수 없습니다.

host-variable

VARCHAR 유형의 변수입니다. *host-variable*의 길이는 17바이트 이하여야 합니다(SQLSTATE 42815). 호스트 변수의 값은 'NONE', 'IMMEDIATE' 또는 'DEFERRED'(대소문자 구분 안함) 중 하나인 왼쪽에 맞춰진 문자열이어야 합니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

주

- 롤아웃 처리에 적합한 연속 DELETE문은 CURRENT MDC ROLLOUT MODE 특수 레지스터를 따릅니다. 현재 실행 중인 섹션은 이 특수 레지스터의 변경에 의해 영향을 받지 않습니다.
- 명령문이 실행되는 작업 단위(UOW)가 롤백된 경우 SET CURRENT MDC ROLLOUT MODE문의 실행 효과는 롤백되지 않습니다.

예 :

롤아웃 처리에 대해 규정하는 다음 DELETE문에 대한 지연된 정리 동작을 지정하십시오.

```
SET CURRENT MDC ROLLOUT MODE IMMEDIATE
```

SET CURRENT OPTIMIZATION PROFILE

SET CURRENT OPTIMIZATION PROFILE문은 CURRENT OPTIMIZATION PROFILE 특수 레지스터에 값을 지정합니다. 이 값은 동적 DML문을 준비할 때 옵티마이저가 사용해야 하는 최적화 프로파일을 지정합니다. 명령문은 트랜잭션의 제어를 받지 않습니다.

명령문이 평가될 때 최적화 프로파일 이름의 유효성이 점검되지만 프로파일은 옵티마이저에서 동적 DML문이 발생할 때까지 처리되지 않습니다.

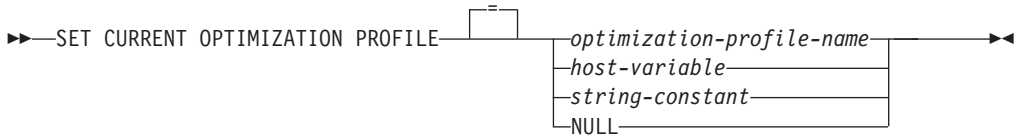
호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

optimization-profile-name

최적화 프로파일의 2부분으로 된 이름입니다. 이름은 리터럴, 호스트 변수 또는 특수 레지스터를 사용하여 지정할 수 있습니다. 지정되는 이름은 CURRENT OPTIMIZATION PROFILE 특수 레지스터에 입력된 이름입니다.

지정된 optimization-profile-name이 규정되지 않는 경우 CURRENT DEFAULT SCHEMA 레지스터의 값이 내재된 규정자로 사용됩니다. 특수 레지스터의 디폴트 값은 널(Null)입니다.

host-variable

최적화 프로파일의 이름을 포함하는 CHAR 또는 VARCHAR 유형의 변수입니다. 널(NULL) 표시기를 포함하는 호스트 변수는 OPTPROFILE 바인드 옵션의 값이 현재 패키지에 대해 지정되는 경우 해당 값이 사용될 것임을 표시합니다. 길이가 영(0)이거나 공백만으로 이루어진 호스트 변수는 최적화 프로파일이 사용되지 않음을 표시합니다.

호스트 변수는 다음 특성을 만족해야 합니다.

- 문자열의 컨텐츠는 앞 공백이 없는 단일 또는 2파트 ID(마침표로 구분됨)입니다.

- ID는 구분하거나 구분하지 않을 수 있습니다.
- 문자열 컨텐츠는 대문자로 변환되지 않습니다.
- 소문자 및 특수 문자를 컬럼 식별자 없는 문자열에서 사용할 수 없습니다.
- 첫 번째 문자가 큰따옴표인 경우, 닫는 큰따옴표는 마침표 앞에 오거나 문자열에서 마지막 비공백 문자여야 합니다.
- 마침표 뒤에 오는 첫 번째 문자가 큰따옴표인 경우 큰따옴표는 문자열에서 마지막 비공백 문자여야 합니다.
- ID가 구분되는 경우, ID에 큰따옴표를 포함시키려면 문자를 두 번 지정하십시오.
- 분리 ID 안에 있지 않은 모든 마침표는 구분자로 처리되며 문자열에는 마침표 구분자가 하나만 존재할 수 있습니다.

string-constant

상수를 최적화 프로파일의 이름인 문자열로 지정합니다. 문자열 상수의 컨텐츠는 호스트 변수와 동일한 특성을 만족해야 합니다.

NULL

CURRENT OPTIMIZATION PROFILE 레지스터를 널(NULL)로 설정합니다.

표 31는 최적화 프로파일 이름 지정 규칙에 따라서 레지스터를 지정하는 데 사용할 수 있는 문자열 리터럴 및 ID의 예를 제공합니다. SCHEMA 및 NAME 컬럼의 값은 OPT_PROFILE 테이블에 나타날 수 있는 최적화 프로파일을 나타냅니다. 유효한 문자열 리터럴 컬럼은 대응하는 SCHEMA 및 NAME 컬럼 값으로 이름 지정된 최적화 프로파일과 일치하는 문자열 리터럴을 표시합니다. 유효한 ID 컬럼은 동일한 최적화 프로파일을 식별하는 ID를 표시합니다.

표 31. 문자열 리터럴 및 ID의 예

SCHEMA	NAME	유효한 문자열 리터럴	유효한 ID
SIMMEN	BIG_PROF	'BIG_PROF' 'SIMMEN.BIG_PROF' '"BIG_PROF"' '"SIMMEN"."BIG_PROF"'	BIG_PROF SIMMEN.BIG_PROF "BIG_PROF" "SIMMEN"."BIG_PROF"
SIMMEN	low_profile	'"low_profile"' 'SIMMEN."low_profile"' '"SIMMEN"."low_profile"'	"low_profile" SIMMEN."low_profile" "SIMMEN"."low_profile"
eliaz	DBA3	'DBA3' '"DBA3"' '"eliaz".DBA3' '"eliaz"."DBA3"'	DBA3 "eliaz".DBA3 "eliaz"."DBA3"

표 31. 문자열 리터럴 및 ID의 예 (계속)

SCHEMA	NAME	유효한 문자열 리터럴	유효한 ID
SNOW	PROFILE1.0	'"PROFILE1.0"'	"PROFILE1.0"
		'SNOW."PROFILE1.0"'	SNOW."PROFILE1.0"
		'"SNOW"."PROFILE1.0"'	"SNOW"."PROFILE1.0"

주

- 레지스터 값이 기존 최적화 프로파일의 이름을 지정하는 경우, 연속 동적 DML문을 준비할 때 지정된 최적화 프로파일이 사용됩니다.
- 레지스터 값이 널(Null)인 경우, 연속 동적 DML문을 준비할 때 OPTPROFILE 바인드 옵션(있는 경우)으로 지정되는 최적화 프로파일이 사용됩니다.
- 레지스터 값이 널(Null)이고 OPTPROFILE 바인드 옵션이 설정되지 않는 경우, 연속 동적 DML문을 준비할 때 최적화 프로파일이 사용되지 않습니다.
- 레지스터 값이 비어 있는 문자열(『』)인 경우, OPTPROFILE 바인드 옵션 설정 여부와 상관없이 연속 동적 DML문을 준비할 때 최적화 프로파일이 사용되지 않습니다.
- CURRENT DEFAULT SCHEMA에 대한 연속 변경사항은 최적화 프로파일에 영향을 주지 않습니다. CURRENT OPTIMIZATION PROFILE 레지스터 값은 SET CURRENT OPTIMIZATION PROFILE문이 평가될 때 적용되는 2파트로 된 이름으로 설정됩니다. 다른 SET CURRENT OPTIMIZATION PROFILE문만이 사용되는 최적화 프로파일을 변경할 수 있습니다.

예:

예 1: 최적화 프로파일 RICK.FOO가 명령문 1, 2 및 3에 사용됩니다. TOM.FOO가 명령문 4에 사용됩니다.

```
SET CURRENT SCHEMA = 'RICK'
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
   statement 1
   statement 2
SET CURRENT SCHEMA = 'TOM'
   statement 3
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
   statement 4
```

예 2: 다음 명령문을 갖는 응용프로그램이 옵션 OPTPROFILE("Foo") 및 QUALIFIER("John")와 함께 바운드되었습니다. 최적화 프로파일 KAAREL.BAR이 명령문 1에 사용되고 최적화 프로파일 "John"."Foo"가 명령문 2에 사용됩니다.

```
SET CURRENT SCHEMA = 'KAAREL'
SET CURRENT OPTIMIZATION PROFILE = 'BAR'
   statement 1
SET CURRENT SCHEMA = "Tom"
SET CURRENT OPTIMIZATION PROFILE NULL
   statement 2
```

예 3: 비어 있는 문자열은 최적화 프로파일이 사용되지 않을 것임을 표시하는 특수 값입니다. 최적화 프로파일 ″Hamid″.″Foo″가 명령문 1에 사용되고 명령문 2에는 최적화 프로파일이 사용되지 않습니다.

```
SET CURRENT OPTIMIZATION PROFILE = '"Hamid"."Foo"'
   statement 1
SET CURRENT OPTIMIZATION PROFILE = ''
   statement 2
```

SET CURRENT PACKAGE PATH

SET CURRENT PACKAGE PATH문은 값을 CURRENT PACKAGE PATH 특수 레지스터에 지정합니다. 이는 트랜잭션의 제어를 받지 않습니다.

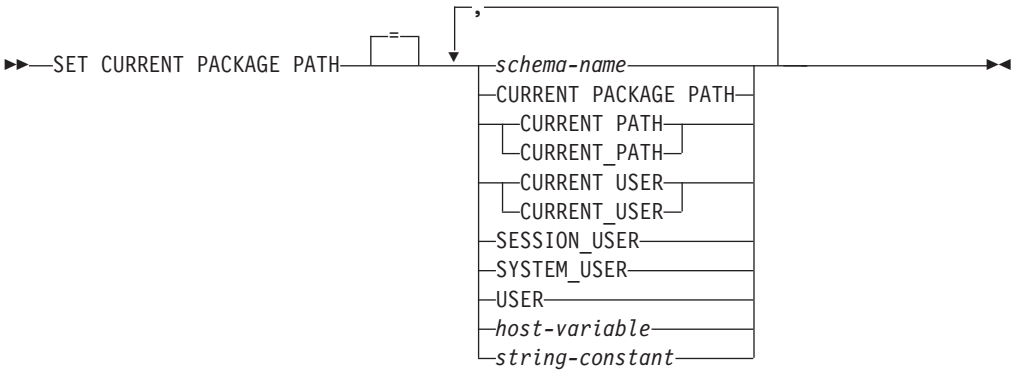
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

schema-name

스키마를 식별합니다. 이름은 비어 있거나 공백만 포함하는 분리 ID가 아니어야 합니다(SQLSTATE 42815).

CURRENT PACKAGE PATH

이 명령문이 실행되기 전의 CURRENT PACKAGE PATH 특수 레지스터 값입니다.

CURRENT PATH

CURRENT PATH 특수 레지스터의 값.

CURRENT USER

CURRENT USER 특수 레지스터의 값.

SESSION_USER

SESSION_USER 특수 레지스터의 값.

SYSTEM_USER

SYSTEM_USER 특수 레지스터의 값.

USER

USER 특수 레지스터의 값입니다.

host-variable

쉼표로 구분된 하나 이상의 스키마 이름을 포함합니다. 호스트 변수는 다음과 같아야 합니다.

- 문자열 변수(CHAR 또는 VARCHAR)입니다. 호스트 변수의 실제 컨텐츠 길이는 CURRENT PACKAGE PATH 특수 레지스터의 길이를 초과할 수 없습니다.
- 널(NULL) 값이 될 수 없습니다. 표시기 변수가 제공되는 경우 해당 값은 널(NULL) 값을 표시해서는 안됩니다.
- 비어 있거나 공백인 문자열 또는 쉼표로 구분된 하나 이상의 스키마 이름을 포함합니다.
- 호스트 변수의 실제 길이가 컨텐츠보다 클 때는 값 오른쪽에 공백을 채워야 합니다.
- CURRENT PACKAGE PATH, CURRENT PATH, CURRENT_PATH, CURRENT USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, PATH 또는 USER를 포함하지 않습니다.
- 비어 있거나 공백만 포함하는 분리 ID를 포함하지 않습니다.

string-constant

쉼표로 구분되는 0개 이상의 스키마 이름을 포함하는 문자열 상수를 지정합니다. 문자열 상수는 다음과 같아야 합니다.

- CURRENT PACKAGE PATH 특수 레지스터의 최대 길이를 초과하지 않는 길이를 갖습니다.
- CURRENT PACKAGE PATH, CURRENT PATH, CURRENT_PATH, CURRENT USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, PATH 또는 USER를 포함하지 않습니다.
- 비어 있거나 공백만 포함하는 분리 ID를 포함하지 않습니다.

규칙

- 동일한 스키마가 목록에서 두 번 이상 나타나는 경우 스키마의 첫 번째 어커런스가 사용됩니다(SQLSTATE 01625).
- 지정될 수 있는 스키마 수는 CURRENT PACKAGE PATH 특수 레지스터의 총 길이로 제한됩니다. 특수 레지스터 문자열은 지정된 각 스키마 이름을 취하고 뒤 공백을 제거하고 이름을 큰따옴표로 구분하고 스키마 이름을 쉼표로 분리하여 빌드됩니다. 결과 목록의 길이는 특수 레지스터의 최대 길이를 초과할 수 없습니다(SQLSTATE 0E000).

- 일반 ID에 대한 규칙을 준수하지 않는 스키마 이름(예를 들어, 소문자 또는 일반 ID에서 지정할 수 없는 문자를 포함하는 스키마 이름)은 구분 스키마 이름으로 지정되어야 하고 호스트 변수나 문자열 상수에서 지정해서는 안됩니다.
- 특수 레지스터(단일 키워드로 지정되는)의 현재 값이 패키지 경로에서 사용되도록 표시하려면 특수 레지스터의 이름을 키워드로 지정하십시오. 특수 레지스터의 이름이 대신 분리 ID(예: ″USER″)로 지정되는 경우 해당 값의 스키마 이름으로 해석됩니다('USER').
- 다음 규칙이 SET CURRENT PACKAGE PATH문에 지정되는 값이 변수 또는 스키마 이름인지 여부를 판별하는 데 사용됩니다.
  - *name*이 SQL 프로시저의 매개변수 또는 SQL 변수와 동일한 경우, *name*은 매개변수 또는 SQL 변수로 해석되며 *name*의 값이 패키지 경로에 지정됩니다.
  - *name*이 SQL 프로시저의 매개변수 또는 SQL 변수와 동일하지 않은 경우, *name*은 스키마 이름으로 해석되며 *name*의 값이 패키지 경로에 지정됩니다.

주

- ***트랜잭션 고려사항:*** SET CURRENT PACKAGE PATH문은 확약 가능한 조작이 아닙니다. ROLLBACK은 CURRENT PACKAGE PATH 특수 레지스터에 효과가 없습니다.
- ***스키마의 존재 검사:*** 지정된 스키마의 존재에 대한 유효성 확인이 CURRENT PACKAGE PATH 특수 레지스터 설정 시 수행되지 않았습니다. 예를 들어 철자가 틀린 스키마가 발견되지 않으며, 이는 후속 SQL이 동작하는 방법에 영향을 미칠 수 있습니다. 패키지 실행 시간에 일치하는 패키지에 대한 권한 부여가 검사되고, 이 권한 부여 검사에 실패하면 오류가 리턴됩니다(SQLSTATE 42501).
- ***호스트 변수 또는 문자열 상수의 컨텐츠:*** 호스트 변수 또는 문자열 상수의 컨텐츠는 스키마 이름의 목록으로 해석됩니다. 여러 개의 스키마 이름을 지정할 때는 쉼표로 구분되어야 합니다. 목록의 각 스키마 이름은 일반 ID 형성 규칙을 준수하거나 분리 ID로 지정되어야 합니다. 호스트 변수 또는 문자열 상수의 컨텐츠는 대문자로 바뀌지 않습니다.
- ***COBOL 응용프로그램에 대한 Embedded SQL에만 해당하는 제한사항:*** SET CURRENT PACKAGE PATH문의 오른쪽에는 최대 10개의 리터럴(호스트 변수가 아닌) 값이 나타날 수 있습니다. 그러한 값은 130(컬럼 식별자 없는) 또는 128(구분됨)의 최대 길이를 가질 수 있습니다.

예:

예 1: CURRENT PACKAGE PATH 특수 레지스터를 스키마 목록 MYPKGS, 'ABC E', SYSIBM으로 설정하십시오.

```
SET CURRENT PACKAGE PATH = MYPKGS, 'ABC E', SYSIBM
```

다음 명령문은 호스트 변수를 결과 목록의 값으로 설정합니다.

```
SET :hvpklist = CURRENT PACKAGE PATH
```

호스트 변수의 값은 ″MYPKGS″, ″ABC E″, ″SYSIBM″입니다.

예 2: CURRENT PACKAGE PATH 특수 레지스터를 스키마 목록 ″SCH4″,″SCH5″으로 설정하십시오. :hvar1은 'SCH4,SCH5'를 포함합니다.

```
SET CURRENT PACKAGE PATH :hvar1
```

이 명령문이 실행된 후 CURRENT PACKAGE PATH 특수 레지스터의 값은 ″SCH4″,″SCH5″입니다.

예 3: CURRENT PACKAGE PATH 특수 레지스터를 스키마 목록 ″SCH1″,″SCH#2″,″SCH3″,″SCH4″,″SCH5″로 설정하십시오. 여기서 :hvar1은 'SCH4,SCH5'를 포함합니다.

```
SET CURRENT PACKAGE PATH = SCH1,'SCH#2',"SCH3",:hvar1
```

이 명령문이 실행된 후 CURRENT PACKAGE PATH 특수 레지스터의 값은 ″SCH1″,″SCH#2″,″SCH3″,″SCH4″,″SCH5″입니다.

예 4: CURRENT PACKAGE PATH 특수 레지스터를 지우십시오.

```
SET CURRENT PACKAGE PATH = ''
```

예 5: SUMMARIZE 프로시저의 실행을 위해 임시로 ″SCH_PROD″ 스키마 (:prodschema 호스트 변수에 포함된) 및 ″SCH_PROD2″ 스키마(:prod2schema 호스트 변수에 포함된)를 CURRENT PACKAGE PATH 특수 레지스터의 끝에 추가하십시오. 그런 다음 CURRENT PACKAGE PATH 특수 레지스터를 이전 값으로 다시 전환하십시오.

```
 SET :oldCPP = CURRENT PACKAGE PATH

SET CURRENT PACKAGE PATH = CURRENT PACKAGE PATH,:prodschema,:prod2schema

CALL SUMMARIZE(:V1,:V2)

SET CURRENT PACKAGE PATH = :oldCPP
```

예 6: CURRENT PACKAGE PATH 특수 레지스터를 구분된 스키마 이름의 목록인 ″MY.SCHEMA″(임베드된 마침표), ″OLD SCHEMA″(임베드된 공백)로 설정하십시오. 분리 ID를 둘 다 포함하는 단일 호스트 변수를 사용하십시오.

```
hv = '"MY.SCHEMA", "OLD SCHEMA"'

SET CURRENT PACKAGE PATH = :hv
```

또는 분리 ID를 둘 다 포함하는 단일 문자열 상수를 사용하십시오.

```
SET CURRENT PACKAGE PATH = '"MY.SCHEMA", "OLD SCHEMA"'
```

또는 구분된 스키마 목록을 사용하십시오.

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

SET CURRENT PACKAGESET

SET CURRENT PACKAGESET문은 후속 SQL문에 사용할 패키지를 선택하는 데 사용할 스키마 이름(콜렉션 ID)을 설정합니다. 이 명령문은 트랜잭션의 제어를 받습니다.

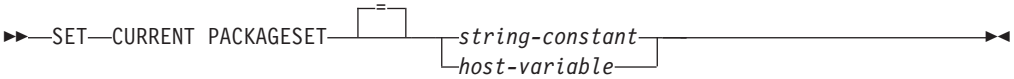
호출

이 명령문은 응용프로그램에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다. 이 명령문은 REXX에서 지원되지 않습니다.

권한 부여

필요한 권한 없음

구문



설명

string-constant

문자열 상수입니다. 값이 128바이트를 초과할 경우 처음 128바이트만 사용될 수 있습니다.

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. 이는 널(NULL)로 설정될 수 없습니다. 값이 128바이트를 초과할 경우 처음 128바이트만 사용될 수 있습니다.

주

- 이 명령문으로 응용프로그램은 실행 가능 SQL문에 패키지를 선택할 때 사용되는 스키마 이름을 지정할 수 있습니다. 명령문은 클라이언트에서 처리되고 응용프로그램 서버(AS)로 가지 않습니다.
- COLLECTION 바인드 옵션은 지정된 스키마 이름으로 패키지를 작성하는 데 사용할 수 있습니다.
- z/OS용 DB2와는 달리, SET CURRENT PACKAGESET문은 CURRENT PACKAGESET라고 하는 특수 레지스터의 지원 없이 구현됩니다.

예 :

TRYIT이라는 응용프로그램이 사용자 ID PRODUSA(이는 바인드 파일에 디폴트 스키마 이름 'PRODUSA'를 만듭니다)로 프리컴파일되었다고 가정합니다. 응용프로그램은 다른 바인드 옵션으로 두 번 바인드됩니다. 다음과 같은 명령행 처리기 명령이 사용되었습니다.

```
DB2 CONNECT TO SAMPLE USER PRODUSA
DB2 BIND TRYIT.BND DATETIME USA
DB2 CONNECT TO SAMPLE USER PRODEUR
DB2 BIND TRYIT.BND DATETIME EUR COLLECTION 'PRODEUR'
```

이는 TRYIT라는 두 개의 패키지 파일을 작성합니다. 첫 번째 바인드 명령은 'PRODUSA'라는 스키마에 패키지를 작성하였습니다. 두 번째 바인드 명령은 COLLECTION 옵션에 기초하여 'PRODEUR'라는 스키마에 패키지를 작성하였습니다.

응용프로그램 TRYIT에 다음의 명령문이 들어 있다고 가정합니다.

```
EXEC SQL CONNECT TO SAMPLE;
.
.
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 1
.
.
EXEC SQL SET CURRENT PACKAGESET 'PRODEUR';                            2
.
.
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 3
```

1 이 명령문은 PRODUSA.TRYIT 패키지를 사용하여 실행하게 되는데 이는 응용프로그램에 대한 디폴트 패키지이기 때문입니다. 그러므로 날짜는 USA 형식으로 리턴됩니다.

2 이 명령문은 패키지 선택에 대해 스키마 이름을 'PRODEUR'에 설정합니다.

3 이 명령문은 SET CURRENT PACKAGESET문의 결과로 PRODEUR.TRYIT 패키지를 사용하여 실행됩니다. 그러므로 날짜는 EUR 형식으로 리턴됩니다.

SET CURRENT QUERY OPTIMIZATION

SET CURRENT QUERY OPTIMIZATION문은 CURRENT QUERY OPTIMIZATION 특수 레지스터에 값을 지정합니다. 값은 동적 SQL문을 준비할 때 사용 가능한 최적화 기술의 현재 클래스를 지정합니다. 이는 트랜잭션의 제어를 받지 않습니다.

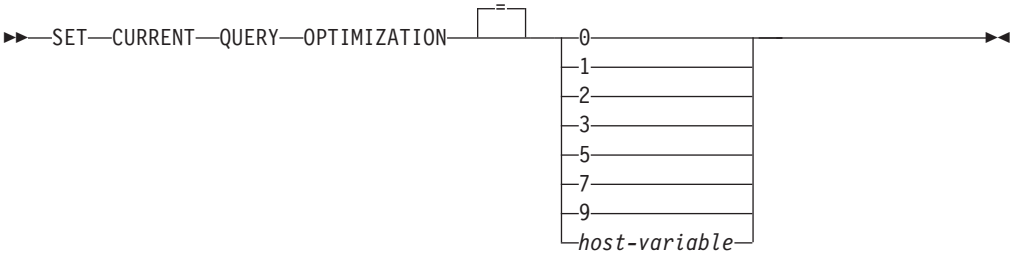
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

optimization-class

*optimization-class*는 정수 상수, 또는 런타임 시 해당값이 포함될 호스트 변수 이름으로서 지정될 수 있습니다. 클래스 개요는 다음과 같습니다.

0 액세스 플랜을 생성하기 위해 최소한의 최적화가 수행되도록 지정합니다. 이 클래스는 인덱스가 잘 작성된 테이블에 대한 단순 동적 SQL 액세스에 가장 적합합니다.

1 액세스 플랜을 생성하기 위해 DB2 버전 1에 대략적으로 비교 가능한 최적화가 수행되도록 지정합니다.

2 DB2 버전 1의 경우보다 높지만 특히 매우 복잡한 쿼리의 경우에 레벨 3 이상보다 상당히 낮은 최적화 비용에서 최적화 레벨을 지정합니다.

3 액세스 플랜을 생성하기 위해 중간 정도의 최적화가 수행되도록 지정합니다.

5 액세스 플랜을 생성하기 위해 상당한 양의 최적화가 수행되도록 지정합니다. 복합 동적 SQL 쿼리의 경우, 경험적 규칙이 액세스 플랜을 선택할 때

소비되는 시간을 제한하는 데 사용됩니다. 가능한 경우 쿼리는 기초적인 기본 테이블 대신 구체화된 쿼리 테이블을 사용합니다.

7 액세스 플랜을 생성하기 위해 상당한 양의 최적화가 수행되도록 지정합니다. 5와 비슷하지만 경험적 규칙이 없습니다.

9 액세스 플랜을 생성하기 위해 최대한의 최적화가 수행되도록 지정합니다. 평가되는 가능한 액세스 플랜의 수를 크게 늘릴 수 있습니다. 이 클래스는 큰 테이블을 사용하여 매우 복잡하고 매우 오랫동안 실행하는 쿼리에 대해 더 좋은 액세스 플랜을 생성할 수 있는지 판별하는 데 사용되어야 합니다. Explain 및 성능 측정을 사용하여 더 나은 플랜이 생성되었는지 검증할 수 있습니다.

host-variable

데이터 유형은 INTEGER입니다. 값은 0 - 9 범위에 있어야 하는데 (SQLSTATE 42815) 0, 1, 2, 3, 5, 7 또는 9여야 합니다. . *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

주

- CURRENT QUERY OPTIMIZATION 레지스터가 특정 값으로 설정되는 경우 쿼리 다시 쓰기 규칙 세트가 사용 가능하며 특정한 최적화 변수가 특정 값을 사용합니다. 그런 다음 이 최적화 기술 클래스가 동적 SQL문 준비 중에 사용됩니다.
- 일반적으로 최적화 클래스를 변경하면 응용프로그램의 실행 시간, 컴파일 시간 및 필요한 자원에 영향을 줍니다. 대부분의 명령문은 디폴트 쿼리 최적화 클래스를 사용하여 충분히 최적화됩니다. 더 낮은 쿼리 최적화 클래스(특히 클래스 1 및 2)가 동적 *PREPARE*에 의해 이용되는 자원이 쿼리를 실행하기 위해 필요한 자원의 중요한 부분인 동적 SQL문에 적합할 수 있습니다. 이용될 수 있는 추가 자원을 고려하고 더 나은 액세스 플랜이 생성되었는지 검증한 후에만 더 높은 최적화 클래스를 선택해야 합니다.
- 쿼리 최적화 클래스는 범위 0 - 9에 속해야 합니다. 이 범위를 벗어난 클래스는 오류를 리턴합니다(SQLSTATE 42815). 이 범위의 지원되지 않는 클래스는 경고를 리턴하며(SQLSTATE 01608) 다음으로 가장 낮은 쿼리 최적화 클래스로 교체됩니다. 예를 들어, 쿼리 최적화 클래스 6은 5로 바뀝니다.
- 동적으로 준비된 명령문은 가장 최근에 실행된 SET CURRENT QUERY OPTIMIZATION문에 의해 설정된 최적화 클래스를 사용합니다. SET CURRENT QUERY OPTIMIZATION문이 아직 실행되지 않은 경우, 쿼리 최적화 클래스는 **dft_queryopt** 데이터베이스 구성 매개변수의 값으로 판별됩니다.
- 정적으로 바운드된 명령문은 CURRENT QUERY OPTIMIZATION 특수 레지스터를 사용하지 않으므로, 이 명령문은 아무 영향이 없습니다. QUERYOPT 옵션이 선행 처리 또는 바인딩 중에 사용되어 정적으로 바운드된 명령문에 대한 원하는 최

적화 클래스를 지정합니다. QUERYOPT가 지정되지 않는 경우, **dft_queryopt** 데이터베이스 구성 매개변수가 지정하는 디폴트값이 사용됩니다.

- SET CURRENT QUERY OPTIMIZATION문이 실행되는 작업 단위(UOW)가 롤백된 경우 이 명령문의 실행 결과는 롤백되지 않습니다.

예:

예 1: 이 예는 최고 등급의 최적화를 선택할 수 있는 방법을 나타냅니다.

```
SET CURRENT QUERY OPTIMIZATION 9
```

예 2: 다음 예는 CURRENT QUERY OPTIMIZATION 특수 레지스터를 쿼리에서 사용할 수 있는 방법을 나타냅니다.

SYSCAT.PACKAGES 카탈로그 뷰를 사용하여 CURRENT QUERY OPTIMIZATION 특수 레지스터의 현재 값과 동일한 설정을 사용하여 바인드된 모든 플랜을 찾으십시오.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
  WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

SET CURRENT REFRESH AGE

SET CURRENT REFRESH AGE문은 CURRENT REFRESH AGE 특수 레지스터의 값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

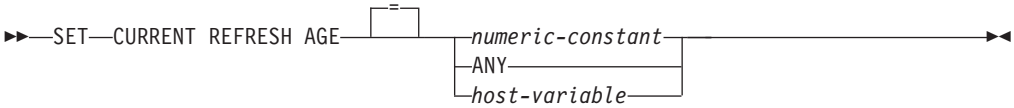
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

numeric-constant

시간소인 지속 기간을 표시하는 DECIMAL(20,6) 값입니다. 값은 0 또는 99 999 999 999 999(값의 마이크로초 부분은 무시되므로 어떤 값이든 가능합니다).

ANY

99 999 999 999 999의 속기입니다.

host-variable

DECIMAL(20,6) 유형 또는 DECIMAL(20,6)에 지정할 수 있는 기타 유형의 변수입니다. 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815). *host-variable*의 값은 0 또는 99 999 999 999 999여야 합니다.

주

- CURRENT REFRESH AGE 특수 레지스터의 초기값은 0입니다.
- CURRENT REFRESH AGE의 값은 지정된 값으로 대체됩니다. 값은 0 또는 99 999 999 999 999여야 합니다. 값 99 999 999 999 999는 9999년, 99월, 99일, 99시간, 99분 및 99초를 표시합니다.

  CURRENT REFRESH AGE 값이 0인 경우, 이 특수 레지스터의 영향을 받는 구체화된 쿼리 테이블은 쿼리 처리를 최적화하는 데 사용되지 않습니다. CURRENT

REFRESH AGE 값이 99 999 999 999 999일 경우, 이 특수 레지스터의 영향을 받는 구체화된 쿼리 테이블은 쿼리의 처리를 최적화하는 데 사용될 수 있지만 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터 값이 테이블을 포함하고 CURRENT QUERY OPTIMIZATION 특수 레지스터가 2로 설정되거나 값이 5 이상일 경우에만 가능합니다. 이 특수 레지스터의 영향을 받는 구체화된 쿼리 테이블은 REFRESH DEFERRED MAINTAINED BY USER 및 REFRESH DEFERRED MAINTAINED BY SYSTEM입니다.

CURRENT QUERY OPTIMIZATION 특수 레지스터가 2 또는 5보다 크거나 같은 값으로 설정되는 경우 항상 REFRESH IMMEDIATE MAINTAINED BY SYSTEM 구체화된 쿼리 테이블을 사용하여 쿼리 처리를 최적화할 수 있습니다.

CURRENT QUERY OPTIMIZATION 특수 레지스터가 2 또는 5보다 크거나 같은 값으로 설정되고 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터의 값이 ALL로 설정되거나 FEDERATED_TOOL을 포함하는 경우 REFRESH DEFERRED MAINTAINED BY FEDERATED_TOOL 구체화된 쿼리 테이블이 최적화에 사용됩니다.

- CURRENT REFRESH AGE 특수 레지스터를 0이 아닌 값으로 설정할 때 주의해야 합니다. CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터에 의해 지정되는 테이블 유형이 기초 기본 테이블의 값을 표시하지 않을 수 있습니다. 그러한 테이블이 쿼리 처리를 최적화하는 데 사용되는 경우 쿼리 결과가 기초 테이블의 데이터를 정확하게 표시하지 않을 수 있습니다. 기초 데이터가 변경되지 않았음을 아는 경우 또는 캐시된 데이터에 대한 지식을 바탕으로 결과에서 어느 정도의 오류를 승인할 의사가 있는 경우 이것이 타당할 수 있습니다.
- CURRENT REFRESH AGE 값 99 999 999 999 999는 시간소인 산술 연산에서 사용할 수 없는데, 결과가 유효한 날짜 범위를 벗어나기 때문입니다(SQLSTATE 22008).

예:

예 1: 다음 명령문은 CURRENT REFRESH AGE 특수 레지스터를 설정합니다.

```
SET CURRENT REFRESH AGE ANY
```

예 2: 다음 예는 CURRENT REFRESH AGE 특수 레지스터의 값을 CURMAXAGE라는 호스트 변수로 검색합니다. 이전 예에 의해 설정되는 값은 99999999999999.000000입니다.

```
EXEC SQL VALUES (CURRENT REFRESH AGE) INTO  :CURMAXAGE;
```

SET ENCRYPTION PASSWORD

SET ENCRYPTION PASSWORD문은 ENCRYPT, DECRYPT_BIN 및 DECRYPT_CHAR 함수가 사용할 암호를 설정합니다. 이 암호는 DB2 인증에 연결되지 않고 데이터 암호화 및 암호 해독만을 위해 사용됩니다.

이 명령문은 트랜잭션의 제어를 받습니다.

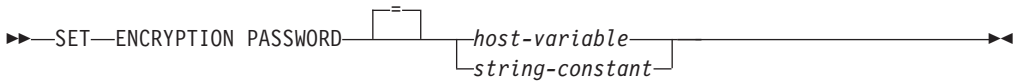
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

암호화 암호는 암호 기반 암호화를 위해 ENCRYPT, DECRYPT_BIN 및 DECRYPT_CHAR 내장 함수에서 사용할 수 있습니다. 암호의 길이는 6 - 127바이트여야 하며 대문자로의 자동 변환을 수행하지 않으므로 모든 문자는 대소문자를 정확히 구분하여 지정해야 합니다. 최상의 시스템 보안 레벨을 유지하려면, SET ENCRYPTION PASSWORD문의 리터럴 문자열을 사용하는 것보다 호스트 변수나 동적 매개변수 표시문자를 사용하여 암호를 지정하는 것이 좋습니다.

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. *host-variable*의 길이는 6 - 127바이트여야 하고(SQLSTATE 428FC) 이는 널(NULL)로 설정될 수 없습니다. 모든 문자는 대소문자로 변환이 되지 않으므로 대소문자를 정확히 구분하여 지정합니다.

string-constant

문자열 상수입니다. 길이는 6 - 127바이트여야 합니다(SQLSTATE 428FC).

주

- ENCRYPTION PASSWORD의 초기값은 빈 문자열('')입니다.
- *host-variable* 또는 *string-constant*가 표준 DB2 메커니즘을 사용하는 데이터베이스 서버에 전송됩니다.

예:

예 1 다음 예는 매개변수 표시문자를 사용하여 Embedded SQL 응용프로그램에 ENCRYPTION PASSWORD 특수 레지스터를 설정하는 방법에 대해 보여줍니다. 반드시 응용프로그램에서 매개변수 표시문자를 사용하여 항상 이 특수 레지스터를 설정하는 것이 좋습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
      char hostVarSetEncPassStmt[200];
      char hostVarPassword[128];
EXEC SQL END DECLARE SECTION;
/* prepare the statement with a parameter marker */
strcpy(hostVarSetEncPassStmt, "SET ENCRYPTION PASSWORD = ?");
EXEC SQL PREPARE hostVarSetEncPassStmt FROM :hostVarSetEncPassStmt;

/* execute the statement for hostVarPassword = 'Gre89Ea' */
strcpy(hostVarPassword, "Gre89Ea");
EXEC SQL EXECUTE hostVarSetEncPassStmt USING :hostVarPassword;
```

SET EVENT MONITOR STATE

SET EVENT MONITOR STATE문은 이벤트 모니터를 활성화 또는 비활성화합니다. 이벤트 모니터의 현재 상태(활성 또는 비활성)는 EVENT_MON_STATE 내장 함수를 이용하여 알 수 있습니다. SET EVENT MONITOR STATE문은 트랜잭션의 제어를 받지 않습니다.

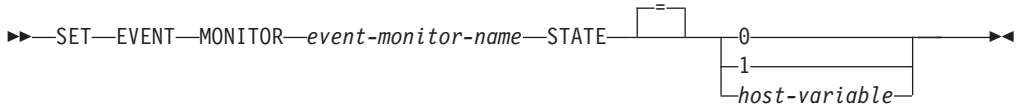
호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 DBADM 또는 SQLADM 권한을 포함해야 합니다.

구문



설명

event-monitor-name

활성화 또는 비활성화할 이벤트 모니터를 식별합니다. 이 이름은 카탈로그에 존재하는 이벤트 모니터를 식별해야 합니다(SQLSTATE 42704).

new-state

*new-state*는 런타임 해당 값이 들어갈 정수 상수 또는 호스트 변수 이름으로서 지정될 수 있습니다. 다음을 지정할 수 있습니다.

0 지정된 이벤트 모니터의 비활성화를 표시합니다.

1 지정된 이벤트 모니터의 활성화를 표시합니다. 이벤트 모니터가 이미 활성화되어 있어서는 안됩니다. 그렇지 않은 경우 경고(SQLSTATE 01598)가 발행됩니다.

host-variable

데이터 유형은 INTEGER입니다. 지정된 값은 0 또는 1이어야 합니다(SQLSTATE 42815). *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

규칙

- 이벤트 모니터를 무제한으로 정의할 수는 있어도 최대 128개의 이벤트 모니터가 동시에 작동할 수 있습니다. 다중 파티션 데이터베이스 환경에서 최대 32 GLOBAL 이벤트 모니터는 각 데이터베이스에 대해 동시에 활성화될 수 있습니다.
- 이벤트 모니터를 활성화하려면 이벤트 모니터가 작성된 트랜잭션이 커미트되어야 합니다(SQLSTATE 55033). 이 규칙은 하나의 작업 단위(UOW)에서 이벤트 모니터의 작성, 모니터 활성화 및 트랜잭션의 롤백을 금지합니다.
- 이벤트 모니터 파일의 크기 또는 수가 CREATE EVENT MONITOR문에 있는 MAXFILES 또는 MAXFILESIZE에 대해 지정된 값을 초과하는 경우 오류(SQLSTATE 54031)가 발생합니다.
- 이벤트 모니터의 목표 경로(CREATE EVENT MONITOR문에 지정되어 있음)를 이미 다른 이벤트 모니터에서 사용하고 있는 경우 오류가 발생합니다(SQLSTATE 51026).

주

- 비WLM 이벤트 모니터를 활성화하면 관련 카운터가 재설정됩니다. WLM, 잠금 및 작업 이벤트 모니터의 단위를 활성화할 때 카운터의 재설정이 발생하지 않습니다.
- WRITE TO TABLE 이벤트 모니터가 SET EVENT MONITOR STATE를 사용하여 시작되면 이 모니터는 SYSCAT.EVENTMONITORS 카탈로그 뷰의 EVMON_ACTIVATES 컬럼을 갱신합니다. 설정 조작이 수행된 작업 단위가 어떤 이유로 롤백되면, 해당 카탈로그 갱신사항이 유실됩니다. 이벤트 모니터를 다시 시작하면 롤백된 EVMON_ACTIVATES 값을 다시 사용합니다.
- 이벤트 모니터가 실행되는 데이터베이스 파티션이 활성화되지 않은 경우, 그 다음 데이터베이스 파티션이 활성화되면 이벤트 모니터가 활성화됩니다.
- 이벤트 모니터는 일단 활성화되면 이벤트 모니터가 명시적으로 비활성화되거나 인스턴스 처리가 되풀이될 때까지 자동 시작 이벤트 모니터처럼 작동합니다. 즉 데이터베이스 파티션이 비활성화되었을 때 이벤트 모니터가 활성화되고 그후에 그 데이터베이스 파티션이 다시 활성화되면, 이벤트 모니터도 명시적으로 재활성화됩니다.
- 데이터베이스가 비활성화될 때 활동 이벤트 모니터가 활성화되면, 큐에서 백로그된 활동 레코드가 지워집니다. 모든 활동 이벤트 모니터 레코드를 확보하고 지운 레코드가 없으려면, 데이터베이스를 비활성화하기 전에 먼저 활동 이벤트 모니터를 명시적으로 비활성화합니다. 활동 이벤트 모니터가 명시적으로 비활성화될 때, 큐에서 백로그된 모든 활동 레코드는 이벤트 모니터가 비활성화하기 전에 처리됩니다.

예:

예 1: 이벤트 모니터 SMITHPAY를 활성화하십시오.

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

예 2: MYSAMPLE가 0 및 2의 두 가지 데이터베이스 파티션을 가진 다중 데이터베이스 파티션이라고 가정하십시오. 파티션 2는 아직 활성화되지 않았습니다.

```
On database partition 0:

   CONNECT TO MYSAMPLE;
   CREATE EVENT MONITOR MYEVMON ON DBPARTITIONNUM 2;
   SET EVENT MONITOR MYEVMON STATE 1;
```

MYEVMON은 MYSAMPLE이 데이터베이스 파티션 2에서 활성화될 때마다 자동으로 활성화됩니다. 이 현상은 **SETEVENT MONITOR** MYEVMON **STATE** 0가 발행되거나 파티션 2가 중지될 때까지 나타납니다.

SET INTEGRITY

SET INTEGRITY문은 다음을 수행하는 데 사용됩니다.

- 테이블의 필수 무결성 처리를 실행하여 하나 이상의 테이블을 이전에는 ″점검 보류 상태″라고 알려졌던 무결성 설정 보류 상태에서 벗어나게 하십시오.
- 테이블의 필수 무결성 처리를 실행하지 않고 하나 이상의 테이블을 무결성 설정 보류 상태에서 벗어나게 하십시오.
- 무결성 설정 보류 상태에 하나 이상의 테이블을 위치시키십시오.
- 하나 이상의 테이블을 전체 액세스 상태로 두십시오.
- 하나 이상의 스테이징 테이블의 내용을 프룬(prune)합니다.

테이블이 로드 또는 접속된 후 무결성 처리를 실행하기 위해 이 명령문을 사용할 경우, 시스템은 제한조건 위반에 해당하는 첨부 부분만을 점검하여 테이블을 증분 방식으로 처리할 수 있습니다. 주제 테이블이 구체화된 쿼리 테이블 또는 스테이징 테이블이고 하위 테이블에 로드, 접속 또는 접속 해제 조작이 수행된 경우에는, 시스템이 구체화된 쿼리 테이블을 증분 방식으로 새로 고치거나 하위 테이블의 델타 부분만 사용하여 스테이징 테이블에 증분 방식으로 전파할 수 있습니다. 그러나 시스템에서 최적화를 수행하지 못하고 대신 데이터 무결성을 위해 전체 무결성 처리를 실행해야 하는 상황이 일부 있습니다. 제한조건을 위반하지 않았는지 전체 테이블을 점검하거나, 구체화된 쿼리 테이블의 정의를 다시 계산하거나 또는 스테이징 테이블을 불일치로 표시하는 방법으로 전체 무결성 처리를 수행합니다. 스테이징 테이블이 불일치로 표시되면 연관된 구체화된 쿼리 테이블에 대해 완전 새로 고침을 수행해야 합니다. INCREMENTAL 옵션을 지정하여 사용자가 증분 처리를 명시적으로 요청할 수 있는 상황도 있습니다.

SET INTEGRITY문은 트랜잭션의 제어를 받습니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

SET INTEGRITY문을 실행하는 데 필요한 특권은 아래에 설명된 대로 용도에 따라 다릅니다.

- 테이블을 무결성 설정 보류 상태에서 벗어나게 하여 필수 무결성 처리를 수행합니다.

  명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

  – CONTROL 특권

- 무결성 처리가 수행된 테이블, 이러한 테이블 중 하나 이상에 대해 예외 테이블이 제공된 경우 예외 테이블에 대한 INSERT 특권
- 명령문에 의해 내재적으로 무결성 설정 보류 상태에 놓일 모든 종속 외부 키 테이블, 종속 즉시 구체화된 쿼리 테이블 및 종속 즉시 스테이징 테이블.

– LOAD 권한(조건이 있음). LOAD 권한이 유효한 특권을 제공한다고 간주하려면 다음 조건을 모두 만족해야 합니다.

- 필수 무결성 처리에는 다음과 같은 조치가 포함되지 않습니다.
  - 구체화된 쿼리 테이블 새로 고침
  - 스테이징 테이블로 전파
  - 생성된 컬럼 또는 식별 컬럼 갱신
- 하나 이상의 테이블에 대해 예외 테이블이 제공된 경우, 무결성 처리가 수행 중인 테이블 또는 연관된 예외 테이블에 대한 무결성 처리 중에 필수 액세스가 부여됩니다. 즉,
  - 무결성 처리가 수행 중인 각 테이블에 대한 SELECT 및 DELETE 특권
  - 예외 테이블의 INSERT 특권

– DATAACCESS 권한

• 필수 무결성 처리 수행 없이 테이블을 무결성 설정 보류 상태에서 벗어나게 합니다.

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

– 처리 중인 테이블의 CONTROL 특권, 명령문에 의해 내재적으로 무결성 설정 보류 상태에 놓이게 되는 각각의 하위 외부 키 테이블, 하위 직접 구체화된 쿼리 테이블 및 하위 직접 스테이징 테이블의 CONTROL 특권

– LOAD 권한

– DATAACCESS 권한

– DBADM 권한

• 무결성 설정 보류 상태에 테이블을 위치시킵니다.

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

– CONTROL 특권

- 지정된 테이블
- 명령문에 의해 무결성 설정 보류 상태에 놓이게 될 하위 외부 키 테이블,
- 명령문에 의해 무결성 설정 보류 상태에 놓이게 될 하위 직접 구체화된 쿼리 테이블 및
- 명령문에 의해 무결성 설정 보류 상태에 놓이게 될 하위 직접 스테이징 테이블

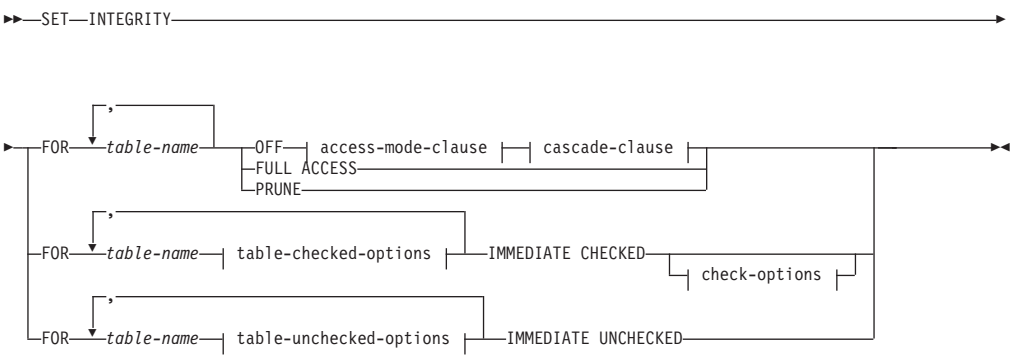
– LOAD 권한

  - DATAACCESS 권한
  - DBADM 권한
- 테이블을 전체 액세스 상태로 두십시오.

  명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.
  - 전체 액세스 상태에 있는 테이블에 대한 CONTROL 특권
  - LOAD 권한
  - DATAACCESS 권한
  - DBADM 권한
- 스테이징 테이블을 프룬(prune)합니다.

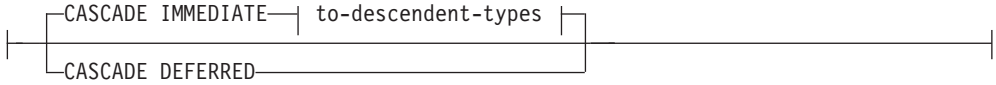
  명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.
  - 프룬(prune) 중인 테이블에 대한 CONTROL 특권
  - DATAACCESS 권한

구문



access-mode-clause:

cascade-clause:



to-descendent-types:

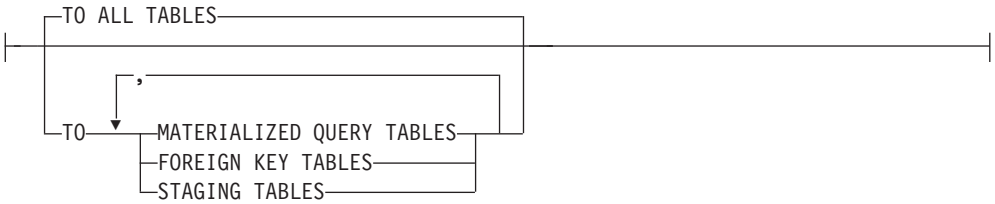
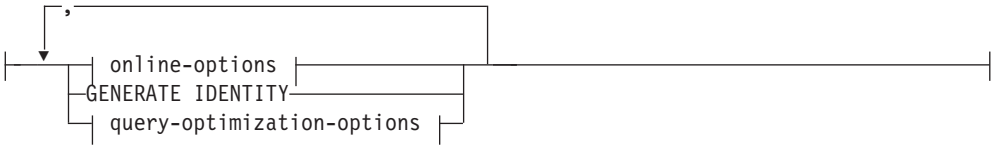


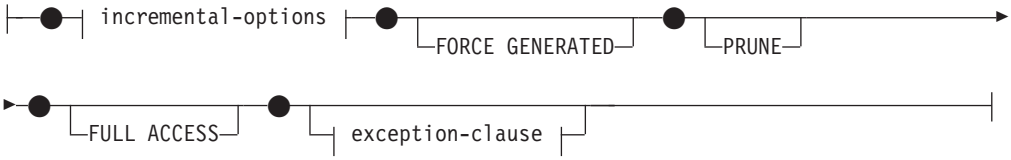
table-checked-options:



online-options:

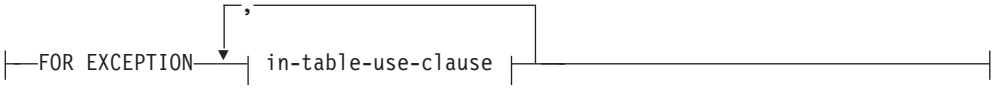
query-optimization-options:

점검 옵션:



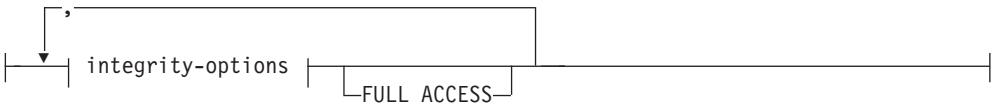
incremental-options:

exception-clause:

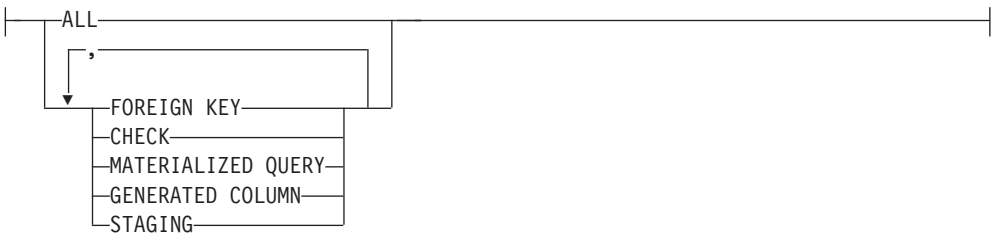


in-table-use-clause:

table-unchecked-options:



integrity-options:



설명

FOR *table-name*

무결성 처리를 위해 하나 이상의 테이블을 식별합니다. 카탈로그에 기술된 테이블이어야 하고 뷰, 카탈로그 테이블 또는 유형이 지정된 테이블이어서는 안됩니다.

OFF

테이블이 무결성 설정 보류 상태에 놓이도록 지정합니다. 무결성 설정 보류 상태 테이블에서는 극히 한정된 활동만 할 수 있습니다.

access-mode-clause

테이블이 무결성 설정 보류 상태일 때의 가독성을 지정합니다.

NO ACCESS

테이블이 무결성 설정 보류 권한 없음 상태에 놓이도록 지정하는데, 이 상태에서는 테이블에 대한 읽기 또는 쓰기 액세스를 할 수 없습니다.

READ ACCESS

테이블이 무결성 설정 보류 읽기 액세스 상태에 놓이도록 지정하는데, 이 상태에서는 테이블의 추가되지 않은 부분에 대해 읽기 액세스를 할 수 있습니다. 무결성 설정 보류 권한 없음 상태에 있는 테이블에는 이 옵션을 사용할 수 없습니다(SQLSTATE 428FH).

cascade-clause

SET INTEGRITY문에서 참조되는 테이블의 무결성 설정 보류 상태가 하위 테이블에 직접 연쇄되는지 지정합니다.

CASCADE IMMEDIATE

무결성 설정 보류 상태가 하위 테이블에 직접 확장되도록 지정합니다.

to-descendent-types

무결성 설정 보류 상태가 직접 연쇄되는 하위 테이블의 유형을 지정합니다.

TO ALL TABLES

무결성 설정 보류 상태가 호출 목록에 있는 테이블의 모든 하위 테이블로 직접 연쇄되도록 지정합니다. 하위 테이블에는 모든 종속 외부 키 테이블, 직접 스테이징 테이블, 호출 목록에서 테이블의 하위 테이블인 직접 구체화된 쿼리 테이블 또는 종속 외부 키 테이블의 하위 테이블이 포함됩니다.

TO ALL TABLES를 지정하면 TO FOREIGN KEY TABLES, TO MATERIALIZED QUERY TABLES 및 TO STAGING TABLES를 모두 동일한 명령문에 지정하는 것과 같습니다.

TO MATERIALIZED QUERY TABLES

TO MATERIALIZED QUERY TABLES만 지정하면 무결성 설정 보류 상태가 하위 직접 구체화된 쿼리 테이블로만 직접 연쇄됩니다. 테이블이 무결성 설정 보류 상태에서 벗어날 때 필요하면 다른 하위 테이블을 나중에 무결성 설정 보류 상태에 놓을 수 있습니다. TO FOREIGN KEY TABLES 및 TO MATERIALIZED QUERY TABLES를 지정하면, 무결성 설정 보류 상태가 하위 외부 키 테이블, 호출 목록에 있는 테이블의 하위 직접 구체화된 쿼리 테이블 및 하위 외부 키 테이블에 종속되는 직접 구체화된 쿼리 테이블로 모두 직접 연쇄됩니다.

TO FOREIGN KEY TABLES

무결성 설정 보류 상태가 하위 외부 키 테이블에 직접 연쇄되도록 지정합니다. 테이블이 무결성 설정 보류 상태에서 벗어날 때 필요하면 다른 하위 테이블을 나중에 무결성 설정 보류 상태에 놓을 수 있습니다.

TO STAGING TABLES

무결성 설정 보류 상태가 하위 스테이징 테이블에 직접 연쇄되도록 지정합니다. 테이블이 무결성 설정 보류 상태에서 벗어날 때 필요하면 다른 하위 테이블을 나중에 무결성 설정 보류 상태에 놓을 수 있습니다. TO FOREIGN KEY TABLES 및 TO STAGING TABLES를 지정하면, 무결성 설정 보류 상태가 하위 외부 키 테이블, 호출 목록에 있는 테이블의 하위 직접 스테이징 테이블 및 하위 외부 키 테이블에 종속되는 하위 직접 스테이징 테이블로 모두 직접 연쇄됩니다.

CASCADE DEFERRED

호출 목록에 있는 테이블만 무결성 설정 보류 상태에 놓이도록 지정합니다. 하위 테이블의 상태는 변경되지 않습니다. 상위 테이블에 제한조건 위반이 있는지 점검하는 경우 하위 외부 키 테이블은 나중에 내재적으로 무결성 설정 보류 상태에 놓일 수 있습니다. 하위 직접 구체화된 쿼리 테이블 및 하위 직접 스테

이징 테이블은 두 테이블의 하위 테이블 중 하나에 무결성 위반이 있는지 점검할 때 내재적으로 무결성 설정 보류 상태에 놓일 수 있습니다.

*cascade-clause*를 지정하지 않으면 무결성 설정 보류 상태가 모든 하위 테이블에 직접 연쇄됩니다.

IMMEDIATE CHECKED

테이블에 필수 무결성 처리를 수행하여 무결성 설정 보류 상태에서 벗어나도록 지정합니다. 해당 지정은 SYSCAT.TABLES 카탈로그 뷰의 STATUS 및 CONST_CHECKED 컬럼에 있는 정보 세트에 따라 수행됩니다. 즉,

- STATUS 컬럼의 값이 무결성 설정 보류 상태에 있는 테이블 'C'여야 하며, 그렇지 않을 경우 테이블이 하위 외부 키 테이블이 아니거나, 하위 구체화된 쿼리 테이블이 아니거나, 목록에 지정되고 무결성 설정 보류 상태에 있으며 중간 상위 구성원도 목록에 있는 테이블의 하위 스테이징 테이블이 아니면 오류가 리턴됩니다(SQLSTATE 51027).
- 점검 중인 테이블이 무결성 설정 보류 상태에 있으면 CONST_CHECKED의 값은 점검할 무결성 옵션이 어떤 것인지 나타냅니다.

테이블이 무결성 설정 보류 상태에서 벗어날 때 필요하면 하위 테이블은 무결성 설정 보류 상태에 놓이게 됩니다. 하위 테이블이 무결성 설정 보류 상태에 있음을 표시하는 경고가 리턴됩니다(SQLSTATE 01586).

시스템 유지보수 구체화된 쿼리 테이블인 경우 데이터는 쿼리에 대해 점검되며 필요에 따라 새로 고쳐집니다. (IMMEDIATE CHECKED는 사용자 유지보수 구체화된 쿼리 테이블에 사용될 수 없습니다.) 스테이징 테이블인 경우에는 데이터가 쿼리 정의에 대해 점검되고 필요하면 전파됩니다.

하위 테이블의 무결성을 점검할 때는 다음과 같습니다.

- 상위 테이블은 무결성 설정 보류 상태가 될 수 없으며 또는
- 각 상위 테이블에 대해 동일한 SET INTEGRITY문에서 제한조건 위반이 있는지 점검해야 합니다.

직접 구체화된 쿼리 테이블이 새로 고쳐지거나 델타가 스테이징 테이블에 전파될 때는 다음과 같습니다.

- 하위 테이블은 무결성 설정 보류 상태가 될 수 없으며 또는
- 각 하위 테이블은 동일한 SET INTEGRITY문에서 점검되어야 합니다.

그렇지 않으면 오류가 발생합니다(SQLSTATE 428A8).

table-checked-options

online-options

테이블 처리 중의 접근성을 지정합니다.

ALLOW NO ACCESS

언커미트된 분리 레벨이 사용되는 경우를 제외하고는 테이블 새로 고침 중에는 다른 사용자가 테이블에 액세스할 수 없도록 지정합니다.

ALLOW READ ACCESS

테이블 처리 중에 다른 사용자가 읽기 전용으로 액세스할 수 있도록 지정합니다.

ALLOW WRITE ACCESS

테이블 처리 중에 다른 사용자가 읽기 및 쓰기 액세스를 할 수 있도록 지정합니다.

GENERATE IDENTITY

테이블이 식별 컬럼을 포함하고 있으면 SET INTEGRITY문이 값을 생성하도록 지정합니다. GENERATE IDENTITY 옵션이 지정되면 접속된 행에만 SET INTEGRITY 문이 생성한 식별 컬럼 값이 존재하는 것이 디폴트값입니다. NOT INCREMENTAL 옵션이 GENERATE IDENTITY 옵션과 함께 지정되어 SET INTEGRITY문이 테이블의 모든 행(접속된 행, 로드된 행 및 기존의 행 포함)에 대해 식별 컬럼 값을 생성하도록 해야 합니다. GENERATE IDENTITY 옵션이 지정되지 않으면 테이블의 모든 행에 대한 현재의 식별 컬럼 값이 변경되지 않은 상태로 남습니다.

query-optimization-options

REFRESH DEFERRED 구체화된 쿼리 테이블의 유지보수에 필요한 쿼리 최적화 옵션을 지정합니다.

ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY

CURRENT REFRESH AGE 특수 레지스터가 'ANY'로 설정되는 경우, *table-name*의 유지보수는 REFRESH DEFERRED 구체화된 쿼리 테이블이 *table-name*을 유지보수하는 쿼리를 최적화하는 데 쓰이도록 지정합니다. *table-name*이 REFRESH DEFERRED 구체화된 쿼리 테이블이 아니면 오류가 리턴됩니다(SQLSTATE 428FH). REFRESH IMMEDIATE 구체화된 쿼리 테이블은 쿼리 최적화 동안 항상 고려됩니다.

check-options

incremental-options

INCREMENTAL

테이블의 첨부된 부분(있는 경우)의 무결성 처리 응용프로그램을 지정합니다. 이 요청이 만족될 수 없다면 즉 시스템이 데이터 무결성에 대한 전체 테이블 점검 필요성을 발견한 경우, 오류가 리턴됩니다(SQLSTATE 55019).

NOT INCREMENTAL

전체 테이블에 무결성 처리 응용프로그램을 지정합니다. 테이블이 구체화된 쿼리 테이블일 경우에는 구체화된 쿼리 테이블 정의가 다시 계산됩니다. 테이블에 적어도 하나의 제한조건이 정의되어 있는 경우, 이 옵션을 지정하면 종속 외부 키 테이블과 종속되는 즉시 구체화된 쿼리 테이블 전체가 처리됩니다. 스테이징 테이블인 경우에는 불일치 상태로 설정됩니다.

*incremental-options*절이 지정되지 않은 경우 시스템이 분증 처리가 가능한지의 여부를 판별하며, 불가능한 경우는 전체 테이블이 점검됩니다.

FORCE GENERATED

테이블에 표현식이 생성한 컬럼이 있는 경우, 값은 표현식에 기초하여 계산되고 해당 컬럼에 저장됩니다. 이 옵션이 지정되지 않을 경우, 마치 동일성 점검 제한조건이 유효한 것처럼 현재 값이 계산된 표현식 값과 비교됩니다. 증분 방식으로 테이블의 무결성을 처리할 경우 생성된 컬럼이 추가된 부분에 대해서만 계산됩니다.

PRUNE

이 옵션은 스테이징 테이블에만 지정할 수 있습니다. 스테이징 테이블의 내용이 프룬(prune)되고 스테이징 테이블이 불일치 상태로 설정되도록 지정합니다. *table-name* 목록에 있는 테이블 중 하나라도 스테이징 테이블이 아니면 오류가 발생합니다(SQLSTATE 428FH). INCREMENTAL 점검 옵션도 지정되는 경우 오류가 발생합니다(SQLSTATE 428FH).

FULL ACCESS

SET INTEGRITY문을 실행한 후 테이블을 완전히 액세스할 수 있도록 지정합니다.

호출 목록의 하위 테이블(종속되는 즉시 구체화된 쿼리 테이블 또는 종속 직접 스테이징 테이블이 있는)이 증분 처리되는 경우, SET INTEGRITY 문을 실행한 후 필요에 따라 하위 테이블이 데이터 이동 안함 모드에 놓입니다. 증분 새로 고침이 가능한 모든 종속되는 즉시 구체화된 쿼리 테이블 및 스테이징 테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우, 하위 테이블은 자동으로 데이터 이동 안함 상태에서 전체 액세스 상태로 전환됩니다. FULL ACCESS 옵션이 IMMEDIATE CHECKED 옵션으로 지정되는 경우, 하위 테이블이 데이터 이동 안함 모드를 생략하고 전체 액세스 상태에 직접 놓입니다. 새로 고치지 않은 종속되는 즉시 구체화된 쿼리 테이블의 경우에는 이후의 REFRESH TABLE문에서 테이블 전체가 다시 계산되고, 테이블의 추가된 부분이 전파되지 않은 종속 직접 스테이징 테이블은 일치하지 않는 것으로 플래그가 설정될 수 있습니다.

호출 목록의 하위 테이블에 전체 처리가 필요하거나 해당 테이블에 종속되는 즉시 구체화된 쿼리 테이블 또는 종속 직접 스테이징 테이블이 없으면,

FULL ACCESS 옵션을 지정했는지 여부에 상관없이 SET INTEGRITY 문을 실행한 후 하위 테이블이 전체 액세스 상태로 직접 전환됩니다.

exception-clause

FOR EXCEPTION

점검 중인 제한조건 위반 행이 예외 테이블로 이동하도록 지정합니다. 오류가 발견된 경우라도, 무결성 설정 보류 상태에서 테이블을 벗어나게 합니다. 하나 이상의 행이 예외 테이블로 이동했음을 나타내는 경고가 리턴됩니다(SQLSTATE 01603).

FOR EXCEPTION 옵션이 지정되지 않고 제한조건이 위반된 경우, 첫 번째로 발견된 위반사항만 리턴됩니다(SQLSTATE 23514). 테이블에 위반사항이 있는 경우, 모든 테이블이 무결성 설정 보류 상태로 남습니다.

위반사항이 발견되면 제한조건 위반 점검시 항상 FOR EXCEPTION 옵션을 사용하여 SET INTEGRITY문의 롤백을 방지할 것을 권장합니다.

IN *table-name*

제한조건을 위반한 행을 가져올 테이블을 지정합니다. 각 테이블에 대해 하나의 예외 테이블이 존재해야 합니다. 이 절은 구체화된 쿼리 테이블 또는 스테이징 테이블에 대해서 지정될 수 없습니다(SQLSTATE 428A7).

USE *table-name*

오류 행이 이동될 예외 테이블을 지정합니다.

FULL ACCESS

FULL ACCESS 옵션을 명령문의 유일한 조작으로 지정하면, 무결성 위반 재점검 없이 테이블이 전체 액세스 상태에 놓입니다. 그러나 새로 고치지 않은 종속되는 즉시 구체화된 쿼리 테이블의 경우에는 이후의 REFRESH TABLE문의 전체 재계산이 필요할 수 있고, 테이블의 델타 부분이 전파되지 않은 종속 직접 스테이징 테이블의 경우는 불완전한 상태로 변경될 수 있습니다. 이 옵션은 무결성 설정 보류 상태에 있는 테이블이 아니라 데이터 이동 안함 상태 또는 권한 없음 상태에 있는 테이블에만 지정할 수 있습니다(SQLSTATE 428FH).

PRUNE

이 옵션은 스테이징 테이블에만 지정할 수 있습니다. 스테이징 테이블의 내용이 프룬(prune)되고 스테이징 테이블이 불일치 상태로 설정되도록 지정합니다. *table-name* 목록에 있는 테이블 중 하나라도 스테이징 테이블이 아니면 오류가 발생합니다(SQLSTATE 428FH).

table-unchecked-options

integrity-options

테이블을 무결성 설정 보류 상태에서 벗어나게 한 경우 생략될 필수 무결성 처리의 유형을 정의하는 데 쓰입니다.

ALL

필수 무결성 처리를 수행하지 않고 테이블을 무결성 설정 보류 상태에서 즉시 벗어나게 합니다.

FOREIGN KEY

테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우 필수 외부 키 제한조건 점검이 실행되지 않습니다.

CHECK

테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우 필수 점검 제한조건 점검이 실행되지 않습니다.

MATERIALIZED QUERY

테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우 구체화된 쿼리 테이블의 필수 새로 고침이 실행되지 않습니다.

GENERATED COLUMN

테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우 필수 생성 컬럼 제한조건 점검이 실행되지 않습니다.

STAGING

테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우 스테이징 테이블로의 데이터 필수 전파가 실행되지 않습니다.

무결성 처리의 특정 유형이 생략된 것으로 표시된 후 테이블에 무결성 처리의 다른 유형이 요구되지 않으면 테이블을 직접 무결성 설정 보류 상태에서 벗어나게 합니다.

FULL ACCESS

SET INTEGRITY문을 실행한 후 테이블을 완전히 액세스할 수 있도록 지정합니다.

호출 목록의 하위 테이블이 증분 방식으로 처리되고 해당 테이블에 종속되는 즉시 구체화된 쿼리 테이블 또는 종속 직접 스테이징 테이블이 있으면, SET INTEGRITY문을 실행한 후 필요에 따라 하위 테이블이 데이터 이동 안 함 상태에 놓입니다. 증분 새로 고침이 가능한 모든 종속되는 즉시 구체화된 쿼리 테이블 및 스테이징 테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우, 하위 테이블은 자동으로 데이터 이동 안함 상태에서 전체 액세스 상태로 전환됩니다. FULL ACCESS 옵션이 IMMEDIATE UNCHECKED 옵션으로 지정되는 경우 하위 테이블이 데이터 이동 안함 모드를 생략하고 직접 전체 액세스 상태에 놓입니다. 새로 고치지 않은 종속되는 즉시 구체화된 쿼리 테이블의 경

우에는 다음 REFRESH TABLE문의 전체 재계산이 진행될 수 있고, 테이블의 추가된 부분이 전파되지 않은 종속 직접 스테이징 테이블은 일치하지 않는 것으로 플래그가 설정될 수 있습니다.

호출 목록의 하위 테이블에 전체 처리가 필요하거나 해당 테이블에 종속되는 즉시 구체화된 쿼리 테이블 또는 종속 직접 스테이징 테이블이 없으면, FULL ACCESS 옵션을 지정했는지 여부에 상관없이 SET INTEGRITY문을 실행한 후 하위 테이블이 전체 액세스 상태로 직접 전환됩니다.

FULL ACCESS 옵션이 IMMEDIATE UNCHECKED 옵션과 함께 지정되고 명령문이 무결성 설정 보류 상태에서 테이블을 벗어나게 하지 않은 경우 오류가 리턴됩니다(SQLSTATE 428FH).

IMMEDIATE UNCHECKED

다음 중 하나를 지정합니다.

- 필수 무결성 처리 없이 테이블을 무결성 설정 보류 상태에서 벗어나게 합니다.
- IMMEDIATE CHECKED 옵션을 사용하여 다음 SET INTEGRITY문이 테이블을 무결성 설정 보류 상태에서 벗어나게 하는 경우, 테이블에는 생략된 필수 무결성 처리의 유형이 하나 이상 있습니다.

해당 옵션을 사용하기 전에 옵션의 데이터 무결성 포함을 고려하십시오. 아래의 『주석』 섹션을 참조하십시오.

주

- 제한된 무결성 설정 관련 상태 중 하나의 상태에 있는 테이블에 미치는 영향입니다.
  - 읽기 액세스 상태 또는 권한 없음 상태에 있는 테이블에서는 INSERT, UPDATE 또는 DELETE를 사용할 수 없습니다. 그리고 읽기 액세스 또는 권한 없음 상태에 있는 테이블에 해당 유형의 수정을 요청하는 명령문은 모두 거부됩니다. 예를 들어 상위 테이블이 권한 없음 상태에 있는 종속 테이블로 연쇄될 경우에는 상위 테이블의 행을 삭제할 수 없습니다.
  - 권한 없음 상태에 있는 테이블에서는 SELECT를 사용할 수 없습니다. 그리고 권한 없음 상태에 있는 테이블에 읽기 액세스를 요구하는 명령문은 모두 거부됩니다.
  - 대부분의 경우 테이블에 추가된 새로운 제한조건은 즉시 실시됩니다. 그러나 테이블이 무결성 설정 보류 상태인 경우, 테이블을 무결성 설정 보류 상태에서 벗어나게 할 때까지 새 제한조건의 점검이 지연됩니다. 테이블이 무결성 설정 보류 상태에 있는 경우, 새 제한조건을 추가하면 데이터의 유효성이 손상될 수 있으므로 테이블이 무결성 설정 보류 권한 없음 상태에 놓입니다.
  - CREATE INDEX문은 읽기 액세스 또는 권한 없음 상태에 있는 테이블을 참조할 수 없습니다. 마찬가지로 기본 키 또는 고유 제한조건을 추가하는 ALTER TABLE문은 읽기 액세스 또는 권한 없음 상태에 있는 테이블을 참조할 수 없습니다.

  - 읽기 액세스 또는 권한 없음 상태에 있는 테이블에서는 임포트 유틸리티를 사용할 수 없습니다.
  - 권한 없음 상태에 있는 테이블에서는 익스포트 유틸리티를 사용할 수 없지만 읽기 액세스 상태에 있는 테이블에서는 사용 가능합니다. 테이블이 읽기 액세스 상태에 있을 때는 익스포트 유틸리티가 추가되지 않은 부분에 있는 데이터만 익스포트합니다.
  - 테이블에서 데이터의 이동을 가져올 수 있는 REORG, REDISTRIBUTE, 분산 키 갱신, 다차원 클러스터링(MDC) 키 갱신, 범위 클러스터링 키 갱신, 테이블 파티션 키 갱신 등의 조작은 읽기 액세스, 권한 없음 또는 데이터 이동 없음의 상태에 있는 테이블에서 사용할 수 없습니다.
  - 로드, 백업, 리스토어, 통계 갱신, runstats, reorgchk, 실행기록 목록 및 rollforward 유틸리티는 전체 액세스, 읽기 액세스, 권한 없음 또는 데이터 이동 없음 상태의 테이블에서 사용할 수 있습니다.
  - ALTER TABLE, COMMENT, DROP TABLE, CREATE ALIAS, CREATE TRIGGER, CREATE VIEW, GRANT, REVOKE 및 SET INTEGRITY문은 전체 액세스, 읽기 액세스, 권한 없음 또는 데이터 이동 없음 상태의 테이블을 참조할 수 있습니다. 그러나 해당 명령문은 테이블을 권한 없음 상태에 놓이도록 만들 수 있습니다.
  - 권한 없음 상태에 있는 테이블에 종속된 패키지, 뷰 및 오브젝트는 실행시 테이블에 액세스하는 경우 오류를 리턴합니다. 읽기 액세스 상태의 테이블에 종속된 패키지는 실행시 삽입, 갱신 또는 삭제 조작을 시도하는 경우 오류를 리턴합니다.

  SET INTEGRITY문으로 위반 행을 제거하는 것은 삭제 이벤트가 아닙니다. 그러므로 트리거는 SET INTEGRITY문으로 활성화되지 않습니다. 유사하게 FORCE GENERATED 옵션을 사용하여 생성된 컬럼을 갱신하면 트리거가 활성화되지 않습니다.

- 증분 처리가 매우 효율적이므로 가능한 경우에는 증분 처리가 사용됩니다. 대부분의 경우 INCREMENTAL 옵션은 필요하지 않습니다. 그러나 무결성 점검이 점증적으로 처리되게 하기 위해서는 필요합니다. 데이터 무결성을 위해 전체 처리가 필요하다고 시스템이 발견할 때는 오류가 리턴됩니다(SQLSTATE 55019).
- IMMEDIATE UNCHECKED절 사용에 관한 경고
  - 이 절은 유틸리티 프로그램에서 사용되어야 하며 응용프로그램에서 사용되는 것은 바람직하지 않습니다. 테이블에 대해 정의된 무결성 스펙을 충족하지 않는 테이블에 데이터가 존재하고 IMMEDIATE UNCHECKED 옵션이 사용되면, 올바르지 않은 쿼리 결과가 리턴될 수 있습니다.

    필수 무결성 처리를 수행하지 않고 테이블을 무결성 설정 보류 상태에서 벗어나게 했다는 사실이 카탈로그에 기록됩니다(SYSCAT.TABLES 뷰의 CONST_CHECKED 컬럼에 있는 각각의 바이트는 'U'로 설정됨). 이는 사용자

가 특정 의무 규정에 대해서 데이터 무결성에 대한 책임이 있다는 것을 나타냅니다. 이 값은 다음 상황 중 하나가 될 때까지는 변경되지 않습니다.

- CONST_CHECKED 컬럼의 'U' 값이 'W' 값으로 변경될 때 해당 테이블은 OFF 옵션이 지정된 SET INTEGRITY문의 테이블을 참조하여 다시 무결성 설정 보류 상태가 되며, 사용자에게 데이터 무결성에 대한 책임이 있었다고 가정되어 시스템이 데이터를 검증할 필요가 있음을 나타냅니다.
- 테이블에 대해 점검되지 않은 모든 의무 규정은 삭제됩니다.

'W' 상태는 이전에 사용자가 무결성을 점검했지만 시스템에서는 아직 점검하지 않았다는 것을 기록한다는 점에서 'N' 상태와 다릅니다. 사용자가 NOT INCREMENTAL 옵션을 지정하여 SET INTEGRITY ... IMMEDIATE CHECKED문을 발행하면 시스템은 전체 테이블의 데이터 무결성을 다시 점검하거나 구체화된 쿼리 테이블에 대해 완전 새로 고침을 수행한 다음 'W' 상태를 'Y' 상태로 변경합니다. IMMEDIATE UNCHECKED를 지정하거나 NOT INCREMENTAL을 지정하지 않으면 'W' 상태가 다시 'U' 상태로 변경되어 시스템이 아직 일부 데이터를 검증하지 않았다는 것을 기록합니다. 후자의 경우(NOT INCREMENTAL이 지정되지 않은 경우) 경고가 리턴됩니다(SQLSTATE 01636).

IMMEDIATE UNCHECKED절을 사용하여 하위 테이블의 무결성을 검증한 경우에는 하위 테이블의 CONST_CHECKED 컬럼에 있는 'U' 값이 다음 테이블에 있는 CONST_CHECKED 컬럼으로 전파됩니다.

- 종속되는 즉시 구체화된 쿼리 테이블
- 종속되는 지연 구체화된 쿼리 테이블
- 스테이징 종속 테이블

종속되는 즉시 구체화된 쿼리 테이블의 경우에는 무결성 설정 보류 상태에서 하위 테이블을 벗어나게 하고 구체화된 쿼리 테이블을 새로 고칠 때마다 위와 같이 값이 전파됩니다. 종속되는 지연 구체화된 쿼리 테이블의 경우에는 구체화된 쿼리 테이블이 새로 고쳐질 때마다 위와 같이 값이 전파됩니다. 종속 스테이징 테이블의 경우에는 무결성 설정 보류 상태에서 하위 테이블이 벗어날 때마다 위와 같이 값이 전파됩니다. 종속 구체화된 쿼리 테이블 및 스테이징 테이블의 CONST_CHECKED 컬럼에 있는 전파된 'U' 값은 IMMEDIATE UNCHECKED 옵션을 사용하여 필수 무결성 처리를 생략한 일부 하위 테이블에 해당 구체화된 쿼리 테이블 및 스테이징 테이블이 종속된다는 사실을 기록합니다.

구체화된 쿼리 테이블의 경우에는 CONST_CHECKED 컬럼에 있는 하위 테이블에서 전파된 'U' 값이 구체화된 쿼리 테이블이 완전히 새로 고쳐지고 이 테이블의 모든 하위 테이블에 있는 CONST_CHECKED 컬럼의 'U' 값이 없어질 때까지 남아 있습니다. 구체화된 쿼리 테이블이 완전히 새로 고쳐지면 구체화된 쿼리 테이블의 CONST_CHECKED 컬럼에 있는 'U' 값이 'Y'로 변경됩니다.

스테이징 테이블의 경우에는 CONST_CHECKED 컬럼의 하위 테이블에서 전파된 'U' 값은 스테이징 테이블의 지연 구체화된 쿼리 테이블이 새로 고쳐질 때까지 남아 있습니다. 지연 구체화된 쿼리 테이블이 새로 고쳐지면 스테이징 테이블의 CONST_CHECKED 컬럼에 있는 'U' 값이 'Y'로 변경됩니다.

– IMMEDIATE CHECKED 옵션이 지정된 동일한 SET INTEGRITY문에서 하위 테이블 및 상위 테이블을 점검하고 상위 테이블이 제한조건에 대한 전체 점검을 요청할 경우, 하위 테이블에 외부 키 제한조건에 대한 CONST_CHECKE 컬럼의 'U' 값이 있는지 여부에 상관없이 하위 테이블의 외부 키 제한조건이 점검됩니다.

- LOAD INSERT 또는 ALTER TABLE ATTACH를 사용하여 데이터를 추가한 후, IMMEDIATE CHECKED 옵션 지정한 SET INTEGRITY문이 테이블에 제한조건 위반이 있는지 점검합니다. 시스템은 테이블에 대한 점증 처리가 가능한지 여부를 결정합니다. 이 경우 첨부된 부분만이 무결성 위반에 대해 점검됩니다. 점증 처리가 가능하지 않을 경우에는 시스템이 전체 테이블에 대해 무결성 위반이 있는지 점검합니다.
- 다음 명령문을 고려하십시오.

```
SET INTEGRITY FOR T IMMEDIATE CHECKED
```

  전체 새로 고침이 필요하거나 INCREMENTAL 옵션을 지정할 수 없기 때문에 전체 테이블에 대해 무결성을 점검해야 하는 경우는 다음과 같습니다.

  – T가 무결성 설정 보류 상태일 때 새 제한조건이 추가된 경우
  – T, T의 상위 또는 T의 하위 테이블에 대해 LOAD REPLACE 조작이 일어난 경우
  – T, T의 상위, T의 하위 테이블에 대해 마지막으로 무결성 점검을 한 후 NOT LOGGED INITIALLY WITH EMPTY TABLE 옵션이 활성화된 경우
  – T의 상위(T가 구체화된 쿼리 테이블이나 스테이징 테이블인 경우에는 하위 테이블)에 대해 비증분 방식으로 무결성을 점검할 때 전체 처리의 연쇄 효과가 적용될 경우
  – 테이블이나 테이블의 상위(또는 구체화된 쿼리 테이블이나 스테이징 테이블의 하위 테이블)가 있는 테이블 스페이스를 특정 시점으로 롤 포워드할 때 테이블과 테이블의 상위(테이블이 구체화된 쿼리 테이블이나 스테이징 테이블일 경우에는 하위 테이블)가 서로 다른 테이블 스페이스에 있는 경우
  – T가 구체화된 쿼리 테이블이고, 마지막 새로 고침 이후 T에 대한 직접적인 LOAD REPLACE 또는 LOAD INSERT 조작이 발생한 경우

- 위 목록에서 설명한 전체 처리 조건이 충족되지 않을 경우 사용자가 SET INTEGRITY FOR T IMMEDIATE CHECKED문에 NOT INCREMENTAL 옵션을 지정하지 않으면 시스템은 추가된 부분에 대해서만 무결성을 점검하거나 구체화된 쿼리 테이블일 경우에는 증분 새로 고침을 수행합니다.

- 무결성 처리 중 오류가 발생하면, 원래의 테이블에서 삭제되고 예외 테이블로 삽입되는 것을 포함한 처리 효과가 모두 롤백됩니다.
- FORCE GENERATED 옵션을 지정하여 SET INTEGRITY문을 발행했는데 로그 스페이스가 부족해서 명령문이 실패할 경우에는 사용 가능한 로그 스페이스를 늘린 다음 SET INTEGRITY문을 다시 발행하십시오. 혹은 GENERATED COLUMN 및 IMMEDIATE UNCHECKED 옵션이 지정된 SET INTEGRITY문을 사용하여 테이블에 대한 생성된 컬럼 점검을 생략하십시오. 그리고 FORCE GENERATED 옵션은 지정되지 않고 IMMEDIATE CHECKED 옵션이 지정된 SET INTEGRITY 문을 발행하여, 적용 가능한 경우 테이블의 다른 무결성 위반을 점검하고 무결성 설정 보류 상태에서 테이블을 벗어나게 하십시오. 테이블이 무결성 설정 보류 상태에서 벗어난 후, 생성된 컬럼을 UPDATE문의 DEFAULT 키워드에 지정하여 해당 컬럼을 생성된 디폴트값으로 갱신할 수 있습니다. 각 커미트에 따른 범위에 따라 다중 검색 갱신 명령문을 사용하거나 간헐적인 커미트를 사용하는 커서 기반 방법을 사용하여 수행할 수 있습니다. 커서 기반 방법을 사용하여 간헐적으로 커미트한 후 잠금을 보유하려면『with hold』커서를 사용해야 합니다.
- SET INTEGRITY문 또는 LOAD 명령의 CASCADE DEFERRED 옵션을 사용하거나 ATTACH절을 가진 ALTER TABLE문을 통해 무결성 설정 보류 상태에 놓인 테이블 및 SET INTEGRITY문의 IMMEDIATE CHECKED 옵션을 사용하여 무결성 위반을 점검하는 테이블의 경우에는 하위 외부 키 테이블, 하위 직접 구체화된 쿼리 테이블 및 하위 직접 스테이징 테이블이 필요에 따라 무결성 설정 보류 상태에 놓입니다.
  - 전체 테이블의 무결성 위반을 점검한 테이블의 경우에는 하위 외부 키 테이블, 하위 직접 구체화된 쿼리 테이블 및 하위 직접 스테이징 테이블이 무결성 설정 보류 상태에 놓입니다.
  - 증분 방식으로 무결성 위반을 점검한 테이블의 경우에는 하위 직접 구체화된 쿼리 테이블 및 스테이징 테이블이 무결성 설정 보류 상태에 놓이고 하위 외부 키 테이블은 원래 상태로 남습니다.
  - 테이블에 점검이 필요하지 않을 경우에는 테이블의 종속되는 즉시 구체화된 쿼리 테이블, 스테이징 종속 테이블 및 종속 외부 키 테이블이 원래 상태로 남습니다.
- SET INTEGRITY문 또는 LOAD 명령의 CASCADE DEFERRED 옵션을 사용하여 무결성 설정 보류 상태에 놓인 테이블 및 SET INTEGRITY문의 IMMEDIATE UNCHECKED 옵션을 사용하여 무결성 설정 보류 상태에서 벗어난 테이블의 경우에는 하위 외부 키 테이블, 하위 직접 구체화된 쿼리 테이블 및 하위 직접 스테이징 테이블이 필요에 따라 무결성 설정 보류 상태에 놓입니다.
  - REPLACE 모드를 사용하여 테이블을 로드한 경우에는 하위 외부 키 테이블, 하위 직접 구체화된 쿼리 테이블 및 하위 직접 스테이징 테이블이 무결성 설정 보류 상태에 놓입니다.

  - INSERT 모드를 사용하여 테이블을 로드한 경우에는 하위 직접 구체화된 쿼리 테이블 및 스테이징 테이블이 무결성 설정 보류 상태에 놓이고 하위 외부 키 테이블은 원래 상태로 남습니다.
  - 테이블을 로드하지 않은 경우에는 테이블의 종속되는 즉시 구체화된 쿼리 테이블, 스테이징 종속 테이블 및 종속 외부 키 테이블이 원래 상태로 남습니다.

- SET INTEGRITY는 긴 실행 명령문입니다. 이로 미루어 볼 때 잠금 시간종료로 인해 전체 명령문이 롤백되는 위험을 줄이기 위해, SET INTEGRITY문을 실행하기 전에 WAIT 옵션이 지정된 SET CURRENT LOCK TIMEOUT문을 발행하고 해당 트랜잭션을 커미트한 후 특수 레지스터를 이전 값에 재설정할 수 있습니다. 그러나 CURRENT LOCK TIMEOUT 특수 레지스터는 특정 잠금 유형 세트에만 영향을 준다는 점에 유의하십시오.
- ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY 옵션을 사용하는 경우, REFRESH DEFERRED 구체화된 쿼리 테이블의 유지보수 순서가 올바르도록 확실히 하십시오. 예를 들어 구체화된 쿼리가 동일한 하위 테이블을 공유하는 2개의 구체화된 쿼리 테이블 MQT1 및 MQT2가 있다고 생각하십시오. MQT2의 구체화된 쿼리는 하위 테이블 대신 MQT1을 사용해서 계산될 수 있습니다. 2개의 구체화된 쿼리 테이블 MQT1 및 MQT2를 유지보수하기 위해 개별적인 명령문을 사용하는 경우 MQT2를 먼저 유지보수하면, 시스템은 MQT2를 유지보수하기 위해 아직 유지보수하지 않은 MQT1의 내용을 선택하여 사용할 수도 있습니다. 이 경우 2개의 구체화된 쿼리 테이블이 거의 동시에 유지보수되었지만 MQT1은 현재 데이터를 포함하는데 반해 MQT2는 스테일 데이터를 포함할 수 있습니다. 1개의 SET INTEGRITY문 대신 2개가 쓰인 경우 올바른 유지보수 순서는 MQT1을 먼저 유지보수하는 것입니다.
- 로드 또는 접속된 기본 테이블에 SET INTEGRITY문을 사용하여 무결성 처리를 수행하는 경우, 동일한 SET INTEGRITY문의 종속되는 REFRESH IMMEDIATE 구체화된 쿼리 테이블 및 PROPAGATE IMMEDIATE 스테이징 테이블을 처리하여 SET INTEGRITY 처리 마지막에 해당 종속 테이블이 무결성 설정 보류 권한 없음 상태에 놓이지 않도록 할 것을 권장합니다. 많은 종속되는 REFRESH IMMEDIATE 구체화된 쿼리 테이블 및 PROPAGATE IMMEDIATE 스테이징 테이블이 있는 기본 테이블의 경우, 메모리 제한조건으로 인해 기본 테이블과 동일한 명령문에서 모든 종속 테이블을 처리하는 것이 불가능할 수 있습니다.
- FORCE GENERATED 또는 GENERATE IDENTITY 옵션이 지정되고 고유 인덱스의 일부로 컬럼이 생성되는 경우 SET INTEGRITY문은 오류를 리턴하고 (SQLSTATE 23505), 고유 인덱스에서 중복 키를 발견하면 해당 명령문이 롤백됩니다. 이 오류는 처리 중인 테이블에 예외 테이블이 있는 경우라도 리턴됩니다.

  이 시나리오는 다음의 상황에서 발생할 수 있습니다.

  - SET INTEGRITY문은 테이블에 대한 LOAD 명령 이후 실행되고 GENERATEDOVERRIDE 또는 IDENTITYOVERRIDE 파일 유형 수정자는 로

드 조작 동안에 지정됩니다. 이 시나리오를 방지하려면 GENERATEDOVERRIDE 대신에 GENERATEDIGNORE 또는 GENERATEDMISSING 파일 유형 수정자를 사용하고 IDENTITYOVERRIDE 대신 IDENTITYIGNORE 또는 IDENTITYMISSING 수정자를 사용할 것을 권장합니다. 권장된 수정자를 사용하면 SET INTEGRITY문 실행 동안 표현식이 생성한 컬럼 또는 식별 컬럼 처리가 필요한 상황을 방지합니다.

– 표현식이 생성한 컬럼의 표현식을 변경하는 ALTER TABLE문 이후에 SET INTEGRITY문이 실행됩니다.

위와 같은 시나리오를 거친 후 무결성 설정 보류 상태에서 테이블을 벗어나게 하려면 다음을 실행하십시오.

– 컬럼 값을 다시 생성하기 위해 FORCE GENERATED 또는 GENERATE IDENTITY 옵션을 사용하지 마십시오. 대신 FOR EXCEPTION 옵션을 IMMEDIATE CHECKED 옵션과 함께 사용하여 생성된 컬럼 표현식을 위반하는 행을 예외 테이블로 이동시키십시오. 그리고 예외 테이블에서 행을 가져와 올바른 표현식을 생성하고 고유 키 점검을 수행하는 테이블에 다시 삽입하십시오. 이 과정을 거치면 생성된 컬럼 표현식을 위반한 행만을 다시 처리하면 되므로 전체 테이블을 다시 처리해야만 하는 상황을 방지해 줍니다.

– 처리 중인 테이블에 접속된 파티션이 있으면 이전 글머리표에 기술된 조치를 수행하기 전에 해당 파티션을 접속 해제하십시오. 그리고 해당 파티션을 다시 접속한 후 SET INTEGRITY문을 실행하여 접속된 파티션에서 무결성을 개별적으로 처리하십시오.

- 예외 테이블과 함께 보호된 테이블에 대해 SET INTEGRITY문을 지정할 경우, 다음과 같은 테이블 기준이 모두 만족해야 합니다. 그렇지 않으면 오류가 리턴됩니다 (SQLSTATE 428A5).
  – 동일한 보안 규정을 사용하여 테이블을 보호해야 합니다.
  – 보호된 테이블의 컬럼에 DB2SECURITYLABEL 데이터 유형이 있으면 예외 테이블의 해당 컬럼에도 DB2SECURITYLABEL 데이터 유형이 있어야 합니다.
  – 보안 레이블을 사용하여 보호된 테이블의 컬럼을 보호할 경우 예외 테이블의 해당 컬럼도 동일한 보안 레이블을 사용하여 보호되어야 합니다.
- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  – SET INTEGRITY 대신 SET CONSTRAINTS를 지정할 수 있습니다.
  – MATERIALIZED QUERY 대신 SUMMARY를 지정할 수 있습니다.

예:

예 1: 다음은 테이블의 무결성 설정 보류 상태 및 무결성 설정 관련 액세스 제한 상태에 대한 정보를 제공하는 쿼리의 예입니다. SUBSTR은 SYSCAT.TABLES의 CONST_CHECKED 컬럼에 있는 개별 바이트를 추출하는 데 사용됩니다. 첫 번째 바

이트는 외부 키 제한조건을 나타내고 두 번째 바이트는 점검 제한조건을 나타내며 다섯 번째 바이트는 구체화된 쿼리 테이블을, 여섯 번째 바이트는 생성된 컬럼 제한조건을, 일곱 번째 바이트는 스테이징 테이블 무결성을 그리고 여덟 번째 바이트는 데이터 파티션 제한조건을 나타냅니다. STATUS는 무결성 설정 보류 상태를 제공하고 ACCESS_MODE는 무결성 설정 관련 액세스 제한 상태를 제공합니다.

```
SELECT TABNAME, STATUS, ACCESS_MODE,
  SUBSTR(CONST_CHECKED,1,1) AS FK_CHECKED,
  SUBSTR(CONST_CHECKED,2,1) AS CC_CHECKED,
  SUBSTR(CONST_CHECKED,5,1) AS MQT_CHECKED,
  SUBSTR(CONST_CHECKED,6,1) AS GC_CHECKED,
  SUBSTR(CONST_CHECKED,7,1) AS STG_CHECKED,
  SUBSTR(CONST_CHECKED,8,1) AS DP_CHECKED
FROM SYSCAT.TABLES
```

예 2: PARENT 테이블을 무결성 설정 보류 권한 없음 상태에 놓고 무결성 설정 보류 상태를 종속 테이블에 직접 연쇄하십시오.

```
SET INTEGRITY FOR PARENT OFF
  NO ACCESS CASCADE IMMEDIATE
```

예 3: 종속 테이블에 무결성 설정 보류 상태를 직접 연쇄하지 않고 PARENT 테이블을 무결성 설정 보류 읽기 액세스 상태에 놓으십시오.

```
SET INTEGRITY FOR PARENT OFF
  READ ACCESS CASCADE DEFERRED
```

예 4: FACT_TABLE이라는 이름의 테이블에 대한 무결성을 점검하십시오. 무결성 위반이 발견되지 않으면 무결성 설정 보류 상태에서 테이블을 가져옵니다. 무결성 위반이 발견되면 전체 명령문이 롤백되고 테이블은 설정 보류 상태로 남습니다.

```
SET INTEGRITY FOR FACT_TABLE IMMEDIATE CHECKED
```

예 5: SALES 및 PRODUCTS 테이블에 대한 무결성을 점검하고 무결성을 위반한 행을 SALES_EXCEPTIONS 및 PRODUCTS_EXCEPTIONS이라는 이름의 예외 테이블로 이동하십시오. 무결성 위반이 있는지의 여부와 관계 없이 SALES 및 PRODUCTS 테이블을 무결성 설정 보류 상태에서 가져옵니다.

```
SET INTEGRITY FOR SALES, PRODUCTS IMMEDIATE CHECKED
  FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS,
  IN PRODUCTS USE PRODUCTS_EXCEPTIONS
```

예 6: MANAGER 테이블에 대해 FOREIGN KEY 제한조건 점검을 사용 가능하게 하고, EMPLOYEE 테이블에 대해 CHECK 제한조건 점검을 사용한 후 IMMEDIATE UNCHECKED 옵션을 사용하여 생략합니다.

```
SET INTEGRITY FOR MANAGER FOREIGN KEY,
  EMPLOYEE CHECK IMMEDIATE UNCHECKED
```

예 7: 두 개의 ALTER TABLE문을 사용하여 점검 제한조건 및 외부 키를 EMP_ACT 테이블에 추가하십시오. OFF 옵션을 지정한 SET INTEGRITY문을 사용하여 테이블

을 무결성 설정 보류 상태에 놓아 두 개의 ALTER TABLE문의 실행 즉시 제한조건을 직접 점검하지 않도록 합니다. IMMEDIATE CHECKED 옵션이 지정된 단일 SET INTEGRITY문은 테이블 전체의 단일 패스 동안 두 개의 추가된 제한조건을 점검하는 데 사용됩니다.

```
SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK
       (EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY
       (EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
       FOR EXCEPTION IN EMP_ACT USE EMP_ACT_EXCEPTIONS
```

예 8: 생성된 컬럼을 올바른 값으로 갱신하십시오.

```
SET INTEGRITY FOR SALES IMMEDIATE CHECKED
  FORCE GENERATED
```

예 9: LOAD INSERT를 사용하여 다른 소스로부터 REFRESH IMMEDIATE 구체화된 쿼리 테이블(SALES_SUMMARY)의 하위 테이블(SALES)로 추가합니다. SALES의 데이터 무결성에 대해 증분 방식으로 점검하고 SALES_SUMMARY를 증분 방식으로 새로 고치십시오. 이 시나리오의 경우 시스템이 증분 처리를 선택하므로 SALES에 대한 무결성 점검 및 SALES_SUMMARY의 새로 고침은 증분 방식으로 이루어집니다. ALLOW READ ACCESS 옵션은 SALES 테이블에 사용되어 처리 중인 테이블의 로드된 부분의 무결성 점검 동안 기존의 데이터를 동시 읽기할 수 있게 합니다.

```
LOAD FROM 2000_DATA.DEL OF DEL
       INSERT INTO SALES ALLOW READ ACCESS;
LOAD FROM 2001_DATA.DEL OF DEL
       INSERT INTO SALES ALLOW READ ACCESS;
SET INTEGRITY FOR SALES ALLOW READ ACCESS IMMEDIATE CHECKED
       FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS;
REFRESH TABLE SALES_SUMMARY;
```

예 10: 새 파티션을 SALES라는 이름의 데이터 파티션된 테이블에 접속하십시오. SALES 테이블의 접속된 데이터에 제한조건 위반이 있는지 증분 방식으로 점검하고 SALES_SUMMARY 종속 테이블을 증분 방식으로 새로 고치십시오. ALLOW WRITE ACCESS 옵션은 두 테이블 모두에 사용되어 무결성 점검이 일어나는 동안 동시 갱신이 가능하도록 합니다.

```
ALTER TABLE SALES
    ATTACH PARTITION STARTING (100) ENDING (200)
    FROM SOURCE;
  SET INTEGRITY FOR SALES ALLOW WRITE ACCESS, SALES_SUMMARY ALLOW WRITE ACCESS
    IMMEDIATE CHECKED FOR EXCEPTION IN SALES
    USE SALES_EXCEPTIONS;
```

예 11: SALES라는 이름의 데이터 파티션된 테이블에서 파티션을 접속 해제하십시오. SALES_SUMMARY 종속 테이블을 증분 방식으로 새로 고치십시오.

```
ALTER TABLE SALES
  DETACH PARTITION 2000_PART INTO ARCHIVE_TABLE;
SET INTEGRITY FOR SALES_SUMMARY
  IMMEDIATE CHECKED;
```

예 12: 사용자가 유지보수하는 구체화된 쿼리 테이블을 가져오면 무결성 설정 보류 상태에서 테이블을 벗어나게 합니다.

```
CREATE TABLE YEARLY_SALES
  AS (SELECT YEAR, SUM(SALES)AS SALES
  FROM FACT_TABLE GROUP BY YEAR)
  DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED BY USER

SET INTEGRITY FOR YEARLY_SALES
  ALL IMMEDIATE UNCHECKED
```

SET PASSTHRU

SET PASSTHRU문은 데이터 소스의 원시(native) SQL을 해당 데이터 소스에 직접 제출하기 위한 세션을 열고 닫습니다. 명령문은 트랜잭션의 제어를 받지 않습니다.

호출

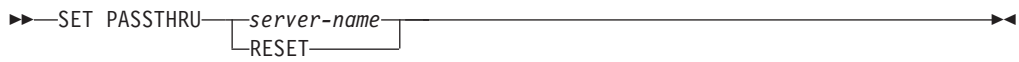
이 명령문은 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 다음을 위해 권한 부여를 제공해야 합니다.

- 데이터 소스로의 Pass through
- 데이터 소스에서 보안 수단 충족

구문



설명

server-name

Pass-through 세션이 열릴 데이터 소스를 이름 지정합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

RESET

Pass-through 세션을 닫습니다.

주

- 다음 제한사항이 Microsoft SQL Server, Sybase 및 Oracle 데이터 소스에 적용됩니다.
  - Microsoft SQL Server 및 Sybase는 사용자 정의 트랜잭션에서 지정할 수 있는 SQL문을 제한하기 때문에, pass-through 모드에서 Microsoft SQL Server 및 Sybase 소스에 사용자 정의 트랜잭션을 사용할 수 없습니다. Pass-through에서 처리되는 SQL문은 DB2에서 구문 분석되지 않기 때문에 사용자가 사용자 정의 트랜잭션에서 허용되는 SQL문을 지정했는지 여부를 발견할 수 없습니다.
  - COMPUTE절은 Microsoft SQL Server 및 Sybase 데이터 소스에서 지원되지 않습니다.

– DDL문은 Microsoft SQL Server, Oracle 및 Sybase 데이터 소스에서 트랜잭션 시맨틱에 해당되지 않습니다. 이 조작은 완료될 때 자동으로 Microsoft SQL Server, Oracle 또는 Sybase에 의해 커미트됩니다. 롤백이 발생하는 경우 DDL은 롤백되지 않습니다.

예:

예 1: 데이터 소스 BACKEND에 대한 pass-through 세션을 시작합니다.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
```

예 2: PREPARE문을 사용하여 pass-through 세션을 시작합니다.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL PREPARE STMT FROM :PASS_THRU;
EXEC SQL EXECUTE STMT;
```

예 3: pass-through 세션을 종료합니다.

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

예 4: PREPARE 및 EXECUTE문을 사용하여 pass-through 세션을 종료합니다.

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL PREPARE STMT FROM :PASS_THRU_RESET;
EXEC SQL EXECUTE STMT;
```

예 5: 데이터 소스에 대한 pass through 세션을 열고, 이 데이터 소스에 있는 테이블에 대한 클러스터된 인덱스를 작성하고, pass-through 세션을 닫습니다.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
EXEC SQL PREPARE STMT                 pass-through mode
  FROM "CREATE UNIQUE
        CLUSTERED INDEX TABLE_INDEX
        ON USER2.TABLE               table is not an
        WITH IGNORE DUP KEY";         alias
EXEC SQL EXECUTE STMT;
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

SET PATH

SET PATH문은 CURRENT PATH 특수 레지스터의 값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다.

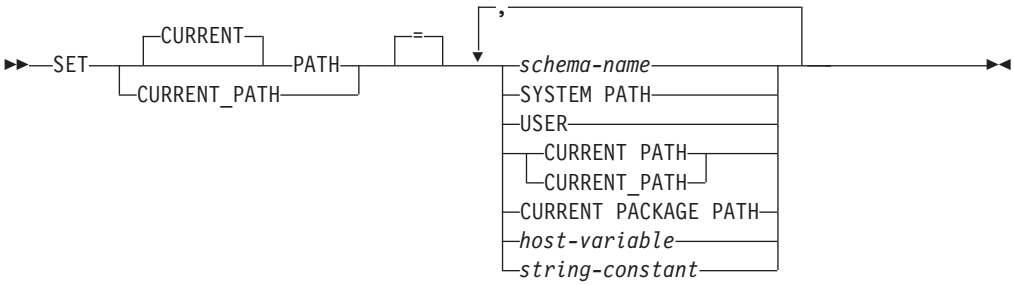
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

schema-name

이 한 부분 설명은 응용프로그램 서버(AS)에 존재하는 스키마를 식별합니다. 경로 설정 시에는 스키마가 존재하는지 확인되지 않습니다. 예를 들어, *schema-name* 철자가 잘못 입력된 경우 오류는 검출되지는 않고 후속 SQL이 작동되는 방식에 영향을 줄 수 있습니다.

SYSTEM PATH

이 값은 스키마 이름 *"SYSIBM","SYSFUN","SYSPROC","SYSIBMADM"*을 지정하는 것과 동일합니다.

USER

USER 특수 레지스터의 값입니다.

CURRENT PATH

명령문을 실행하기 전의 CURRENT PATH 특수 레지스터값입니다.

CURRENT PACKAGE PATH

CURRENT PACKAGE PATH 특수 레지스터값입니다.

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. *host-variable* 컨텐츠의 길이는 128

바이트를 초과할 수 없습니다(SQLSTATE 42815). 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

*host-variable*의 문자는 왼쪽 정렬되어야 합니다. *host-variable*로 *schema-name*을 지정할 때, 모든 문자는 대소문자를 정확히 구분하여 지정해야 합니다. (대문자로의 변환이 수행되지 않습니다.)

string-constant
최대 128바이트 길이의 문자열 상수입니다.

규칙

- 스키마 이름은 SQL 경로에 한 번을 초과하여 나타날 수 없습니다(SQLSTATE 42732).
- 스키마 이름 SYSPUBLIC은 SQL 경로에 지정할 수 없습니다(SQLSTATE 42815).
- 지정될 수 있는 스키마 수는 CURRENT FUNCTION PATH 특수 레지스터의 총 길이로 제한됩니다. 특수 레지스터 문자열은 지정된 각 스키마 이름을 취하고 뒤 공백을 제거하며 큰 따옴표로 한계를 정하고 필요한 스키마 이름 내에 큰 따옴표를 한 후 각 스키마 이름을 쉼표로 구분하여 만듭니다. 결과 문자열의 길이는 2048바이트를 초과할 수 없습니다(SQLSTATE 42907).

주

- CURRENT PATH 특수 레지스터의 초기 값은 ″SYSIBM″,″SYSFUN″,″SYSPROC″,″SYSIBMADM″,″X″입니다. 여기서 X는 USER 특수 레지스터의 값입니다.
- 스키마 SYSIBM은 지정될 필요가 없습니다. SQL 경로에 포함되지 않으면, 이는 내재적으로 첫 번째 스키마로 간주됩니다. (이 경우, CURRENT PATH 특수 레지스터에는 포함되지 않습니다.)
- CURRENT PATH 특수 레지스터는 동적 SQL문의 사용자 정의 데이터 유형, 프로시저 및 함수를 해석하는 데 사용되는 SQL 경로를 지정합니다. FUNCPATH 바인드 옵션은 정적 SQL문의 사용자 정의 데이터 유형 및 함수를 해석하는 데 사용되는 SQL 경로를 지정합니다.
- *호환성*: 이전 버전의 DB2와의 호환성을 위해 다음이 지원됩니다.
  - CURRENT PATH 대신 CURRENT FUNCTION PATH를 지정할 수 있습니다.

예:

예 1: 다음 명령문은 CURRENT PATH 특수 레지스터를 설정합니다.

```
SET PATH = FERMAT, "McDrw #8", SYSIBM
```

예 2: CURRENT PATH 특수 레지스터의 현재 값을 CURPATH라는 호스트 변수로 읽어오십시오.

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

이전 예의 값이 설정되는 경우 ″FERMAT″,″McDrw #8″,″SYSIBM″이 됩니다.

SET ROLE

SET ROLE문은 세션의 권한 부여 ID가 특정 역할의 구성원인지 검증합니다. 권한 부여 ID는 권한 부여 ID가 구성원인 그룹 또는 역할이나 권한 부여 ID에 역할이 부여될 때 역할에서 멤버쉽을 획득합니다.

호출

이 명령문은 응용프로그램에 임베디드하거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문

설명

role-name

세션의 권한 부여 ID가 검증될 멤버쉽의 해당 역할을 지정합니다. *role-name*은 현재 서버에서 기존 역할을 식별해야 합니다(SQLSTATE 42704). 세션의 권한 부여 ID가 *role-name*의 구성원이 아니면 오류가 리턴됩니다(SQLSTATE 42501).

주

- 권한 부여 ID에 부여된 모든 역할은 권한 검사에 사용됩니다. SET ROLE문은 이 권한 부여 검사에 사용되는 역할에 영향을 주지 않습니다. 권한 부여 ID가 멤버쉽을 가지고 있는 역할을 변경하려면 GRANT ROLE 및 REVOKE ROLE문을 사용하십시오.

예:

예 1: 사용자 WALID에게 역할 EDITOR가 부여되었지만 역할 AUTHOR는 부여되지 않았습니다. WALID가 EDITOR 역할의 구성원인지 검증하십시오.

```
SET ROLE EDITOR
```

예 2: WALID가 AUTHOR 역할의 구성원이 아닌지 검증하십시오. 다음 명령문은 오류를 리턴합니다(SQLSTATE 42501).

```
SET ROLE AUTHOR
```

SET SCHEM

SET SCHEMA문은 CURRENT SCHEMA 특수 레지스터의 값을 변경합니다. 이는 트랜잭션의 제어를 받지 않습니다. 패키지가 DYNAMICRULES BIND 옵션으로 바인드되는 경우 이 명령문은 규정되지 않은 데이터베이스 오브젝트 참조에 사용되는 규정자에 대해 아무런 영향을 주지 않습니다.

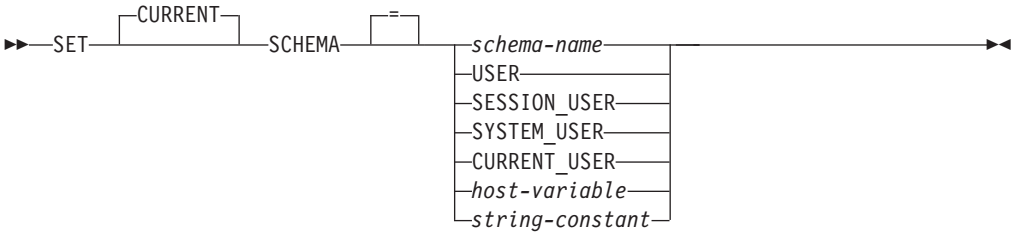
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

필요한 권한 없음

구문



설명

schema-name

이 한 부분 설명은 응용프로그램 서버(AS)에 존재하는 스키마를 식별합니다. 길이는 128바이트를 초과할 수 없습니다(SQLSTATE 42815). 해당 스키마가 설정되는 시점에 스키마 존재 여부는 확인되지 않습니다. *schema-name* 철자를 잘못 입력할 경우, 오류가 검출되지는 않으나 후속 SQL이 작동하는 방식에 영향을 미칠 수 있습니다.

USER

USER 특수 레지스터의 값

SESSION_USER

SESSION_USER 특수 레지스터의 값입니다.

SYSTEM_USER

SYSTEM_USER 특수 레지스터의 값

CURRENT_USER

CURRENT_USER 특수 레지스터의 값입니다.

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. *host-variable*의 컨텐츠의 길이는 128 바이트를 초과할 수 없습니다(SQLSTATE 42815). 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 42815).

*host-variable*의 문자는 왼쪽 정렬되어야 합니다. *host-variable*로 *schema-name*을 지정할 때, 모든 문자는 대소문자를 정확히 구분하여 지정해야 합니다. (대문자로의 변환이 수행되지 않습니다.)

string-constant

최대 128바이트 길이의 문자열 상수입니다.

규칙

- 지정된 값이 *schema-name*에 대한 규칙을 지키지 않으면 오류가 발생합니다 (SQLSTATE 3F000).
- CURRENT SCHEMA 특수 레지스터의 값은 모든 동적 SQL문에서 스키마로 사용되나 데이터베이스 오브젝트에 대한 규정되지 않은 참조가 존재하는 CREATE SCHEMA문은 예외입니다.
- UALIFIER 바인드 옵션은 정적 SQL문에서 규정되지 않은 데이터베이스 오브젝트 이름에 대한 규정자로 사용될 스키마 이름을 지정합니다.

주

- CURRENT SCHEMA 특수 레지스터의 초기값은 USER 특수 레지스터의 초기값과 동일합니다.
- CURRENT SCHEMA 특수 레지스터의 설정이 CURRENT PATH 특수 레지스터에 영향을 미치지는 않습니다. 따라서 CURRENT SCHEMA는 SQL 경로 및 함수에 포함되지 않으며 프로시저 및 사용자 정의 유형 분석 시 이러한 오브젝트를 찾지 못할 수 있습니다. SQL 경로에 현재 스키마 값을 포함시키려면, SET SCHEMA문을 발행할 때마다 SET SCHEMA문에 있는 스키마 이름을 포함하는 SET PATH 문도 발행하십시오.
- CURRENT SQLID는 CURRENT SCHEMA의 동의어로 채택되며 SET CURRENT SQLID문이 미치는 영향은 SET CURRENT SCHEMA문이 미치는 영향과 동일합니다. 명령문 권한 부여 변경과 같은 다른 영향은 발생하지 않습니다.

예:

예 1: 다음 명령문은 CURRENT SCHEMA 특수 레지스터를 설정합니다.

```
SET SCHEMA RICK
```

예 2: CURRENT SCHEMA 특수 레지스터의 현재 값을 CURSCHEMA라는 호스트 변수로 읽어오십시오.

```
EXEC SQL VALUES (CURRENT SCHEMA) INTO :CURSCHEMA;
```

값은 이전 예에서 설정된 99999999999999.000000입니다.

SET SERVER OPTION

SET SERVER OPTION문은 사용자 또는 응용프로그램이 페더레이티드 데이터베이스에 연결된 동안 효력이 적용될 서버 옵션 설정을 지정합니다. 연결이 종료될 때 이 서버 옵션의 이전 설정이 복원됩니다. 이 명령문은 트랜잭션의 제어를 받습니다.

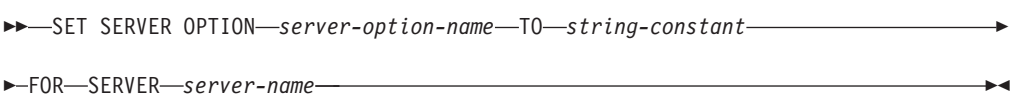
호출

이 명령문은 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

해당사항 없음.

구문



설명

server-option-name
설정될 서버 옵션을 이름 지정합니다.

TO *string-constant*
*server-option-name*에 대한 설정값을 문자열 상수로 지정합니다.

SERVER *server-name*
*server-option-name*이 적용되는 데이터 소스를 이름 지정합니다. 카탈로그에 기술된 서버여야 합니다.

주

- 서버 옵션 이름은 대문자 또는 소문자로 입력할 수 있습니다.
- 사용자 또는 응용프로그램이 페더레이티드 데이터베이스에 연결할 때 하나 이상의 SET SERVER OPTION문을 제출할 수 있습니다. 연결이 설정된 후 처리되는 첫 번째 작업 단위(UOW)의 시작에서 명령문을 지정해야 합니다.
- 변경사항이 현재 연결에만 영향을 주기 때문에 SET SERVER OPTION문을 기초로 SYSCAT.SERVEROPTIONS가 갱신되지 않습니다.
- 정적 SQL의 경우, SET SERVER OPTION문을 사용하면 정적 SQL문의 실행에만 영향을 줍니다. SET SERVER OPTION문 사용은 옵티마이저가 생성하는 플랜에 영향을 주지 않습니다.

예:

예 1: ORASERV라는 Oracle 데이터 소스가 페더레이티드 데이터베이스 DJDB에 정의됩니다. ORASERV는 플랜 힌트를 허용하지 않도록 구성됩니다. 그러나 DBA는 플랜 힌트가 새 응용프로그램의 테스트 실행에 대해 사용 가능하기 원합니다. 실행이 끝나면 플랜 힌트가 다시 승인 불가합니다.

```
CONNECT TO DJDB;
strcpy(stmt,"set server option plan_hints to 'Y' for server oraserv");
EXEC SQL EXECUTE IMMEDIATE :stmt;
strcpy(stmt,"select c1 from ora_t1 where c1 > 100"); /*Generate plan hints*/
EXEC SQL PREPARE s1 FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hv;
```

예 2: 모든 Oracle 8 데이터 소스에 대해 서버 옵션 PASSWORD를 'Y'로 설정했습니다(데이터 소스에서 암호의 유효성 확인). 그러나 응용프로그램이 특정 Oracle 8 데이터 소스(페더레이티드 데이터베이스 DJDB에 ORA8A로서 정의된 소스)에 액세스하기 위해 페더레이티드 데이터베이스에 연결되는 특정 세션의 경우 암호를 유효성 검증할 필요가 없습니다.

```
CONNECT TO DJDB;
strcpy(stmt,"set server option password to 'N' for server ora8a");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL EXECUTE STMT_NAME FROM :stmt;
strcpy(stmt,"select max(c1) from ora8a_t1");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR STMT_NAME;
EXEC SQL OPEN c1; /*Does not validate password at ora8a*/
EXEC SQL FETCH c1 INTO :hv;
```

SET SESSION AUTHORIZATION

SET SESSION AUTHORIZATION문은 SESSION_USER 특수 레지스터의 값을 변경합니다. 명령문은 트랜잭션의 제어를 받지 않습니다. SET SESSION AUTHORIZATION 명령문은 같은 연결에 대해 서로 다른 권한 부여 ID가 있다는 가정 아래 단일 사용자를 지원하므로 서로 다른 사용자가 같은 연결을 재사용(보통 연결 풀링(pooling)으로 지칭)하는 시나리오에서는 사용될 수 없습니다.

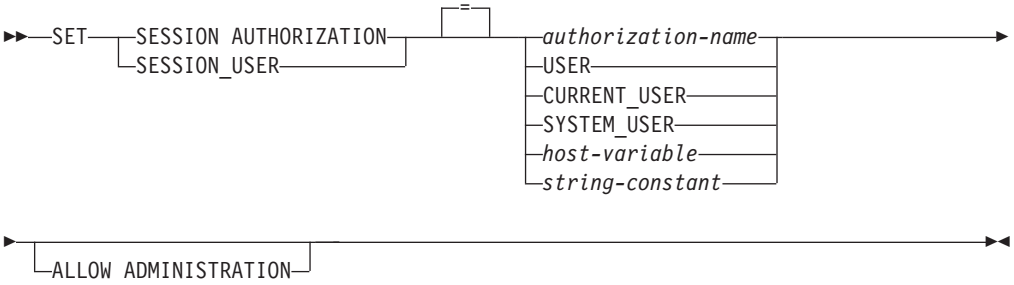
호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID에서 보유하는 특권은 특수 레지스터를 설정 중인 권한 부여 ID에 대한 SETSESSIONUSER를 포함해야 합니다.

구문



설명

authorization-name

SESSION_USER 특수 레지스터에 새 값으로 사용될 권한 부여 ID를 지정하십시오.

USER

USER 특수 레지스터의 값

CURRENT_USER

CURRENT USER 특수 레지스터의 값

SYSTEM_USER

SYSTEM_USER 특수 레지스터의 값

host-variable

CHAR 또는 VARCHAR의 유형 변수입니다. *host-variable* 컨텐츠의 길이는 128

바이트를 초과하지 않아야 합니다(SQLSTATE 28000). 널(NULL)로 설정될 수도 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 이 표시기 변수의 값은 널(Null) 값을 표시해서는 안됩니다(SQLSTATE 28000).

*host-variable*의 문자는 왼쪽 정렬되어야 합니다. 호스트 변수를 사용하여 *authorization-name*을 지정할 때, 모든 문자는 대문자로 지정해야 합니다. (대문자로의 변환이 수행되지 않음.)

string-constant

최대 길이가 128바이트인 문자열 상수.

ALLOW ADMINISTRATION

SQL 스키마 문을 동일한 작업 단위(UOW)에서 이 명령문 이전에 지정할 수 있음을 지정하십시오.

규칙

- SESSION_USER 특수 레지스터에 대해 지정하는 값은 USER 유형의 권한 부여 ID에 해당되는 규칙을 따라야 합니다(SQLSTATE 42602).
- OWNER 바인드 옵션에서 정적 SQL문에 사용될 권한 부여 ID를 지정합니다.
- 이 명령문은 WITH HOLD 커서를 열지 않은 상태의 새 작업 단위에서 첫 번째 명령문(SET 특수 레지스터 명령문을 제외한)으로서만 발행될 수 있습니다(SQLSTATE 25001). 이 제한조건에는 SET 특수 레지스터 명령문 이외의 명령문에 대한 PREPARE 요청이 포함됩니다.
- SESSION_USER 특수 레지스터의 값은 DYNAMICRULES(RUN) 바인드 옵션을 사용하여 바운드한 패키지의 모든 동적 SQL문에 사용되는 권한 부여 ID로서 사용됩니다. (패키지가 루틴에 사용되지 않는 경우 INVOKERUN 및 DEFINERUN이 포함됩니다.) 패키지가 소유자, 호출자 또는 DYNAMICRULES 옵션을 기본으로 한 정의자 권한 부여를 사용하는 경우 이 명령문은 패키지에서 발행한 동적 SQL문에 대해 효력이 없습니다.

주

- SET SESSION AUTHORIZATION 명령문은 세션 권한 부여 ID를 변경할 수 있도록 합니다. 세션 권한 부여 ID는 연결의 현재 사용자를 나타내며 DB2에서 DYNAMICRULES 실행 패키지에 있는 동적 SQL과 관련된 모든 권한 부여 검사에 대해 고려하는 권한 부여 ID입니다. SESSION_USER 특수 레지스터를 사용하여 이 세션 권한 부여 ID의 현재 값을 볼 수 있습니다.
- 새 연결에 사용되는 SESSION_USER 특수 레지스터의 초기값은 SYSTEM_USER 특수 레지스터의 값과 동일합니다.
- 이 명령문에서 지정한 세션 권한 부여 ID에 대한 그룹 정보는 명령문 실행 시 획득할 수 있습니다.

- SESSION_USER 특수 레지스터의 설정이 CURRENT SCHEMA 또는 CURRENT PATH 특수 레지스터에 영향을 미치지는 않습니다.
- SESSION_USER 특수 레지스터 설정 시 오류가 발생하는 경우 레지스터는 이전 값으로 되돌아갑니다.
- 이 명령문을 사용할 경우 각각의 사용자가 원래 연결 소유자의 SESSION_USER 특수 레지스터값을 변경하는 기능을 상속하기 때문에 여러 다른 사용자가 같은 연결을 재사용할 수 없습니다. 이 명령문은 특권 점검에 필요한 SYSTEM_USER 값에 좌우되며 SET SESSION AUTHORIZATION문이 초기 연결 권한 부여 ID를 변경할 수 없습니다. 또한 이 명령문은 연결 재사용에 영향을 미치는 다음 동작을 조정할 수 없습니다.
  - 새 권한 부여 ID에 대해 CONNECT 특권을 점검하지 않습니다.
  - 갱신 가능한 특수 레지스터의 컨텐츠를 재설정하지 않습니다. 특히 ENCRYPTION PASSWORD 특수 레지스터의 컨텐츠는 수정되지 않고도 암호화 또는 암호 해독에 필요한 새 권한 부여 ID에 사용될 수 있습니다.
  - 선언된 전역 임시 테이블의 컨텐츠는 영향을 받지 않고도 새 권한 부여 ID에 액세스할 수 있습니다.
  - 리모트 서버에 대한 기존의 링크를 재설정하지 않습니다.
- ALLOW ADMINISTRATION절을 지정한 경우 다음 명령문 또는 조작 유형이 SET SESSION AUTHORIZATION문 전에 수행될 수 있습니다.
  - 데이터 정의 언어(DDL)(세이브포인트 정의 및 전역 임시 테이블 선언은 포함되지만 SET INTEGRITY는 포함되지 않음)
  - GRANT 및 REVOKE문
  - LOCK TABLE문
  - COMMIT 및 ROLLBACK문
  - 특수 레비스터의 SET
  - 전역 변수의 SET

예:

예 1: 다음 명령문은 SESSION_USER 특수 레지스터를 설정합니다.

```
SET SESSION_USER = RAJIV
```

예 2: 세션 권한 부여 ID(SESSION_USER 특수 레지스터)가, 해당 연결(명령문이 발행된)을 설정했던 시스템 권한 부여 ID 값이 되도록 설정하십시오.

```
SET SESSION AUTHORIZATION SYSTEM_USER
```

SET variable

SET variable문은 변수에 값을 지정합니다. 이 명령문은 트랜잭션의 제어를 받습니다.

호출

이 명령문은 응용프로그램에 임베드되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

전이 변수를 참조하려면 트리거 작성자의 권한 부여 ID가 보유하는 특권이 최소한 다음 중 하나를 포함해야 합니다.

- 할당된 왼쪽에서 참조되는 컬럼에는 UPDATE 특권 그리고 오른쪽에서 참조되는 컬럼에는 SELECT 특권
- 테이블에 대한 CONTROL 특권(트리거의 주제 테이블)
- DATAACCESS 권한

전역 변수가 지정 명령문의 오른쪽에서 참조되면 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 READ 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

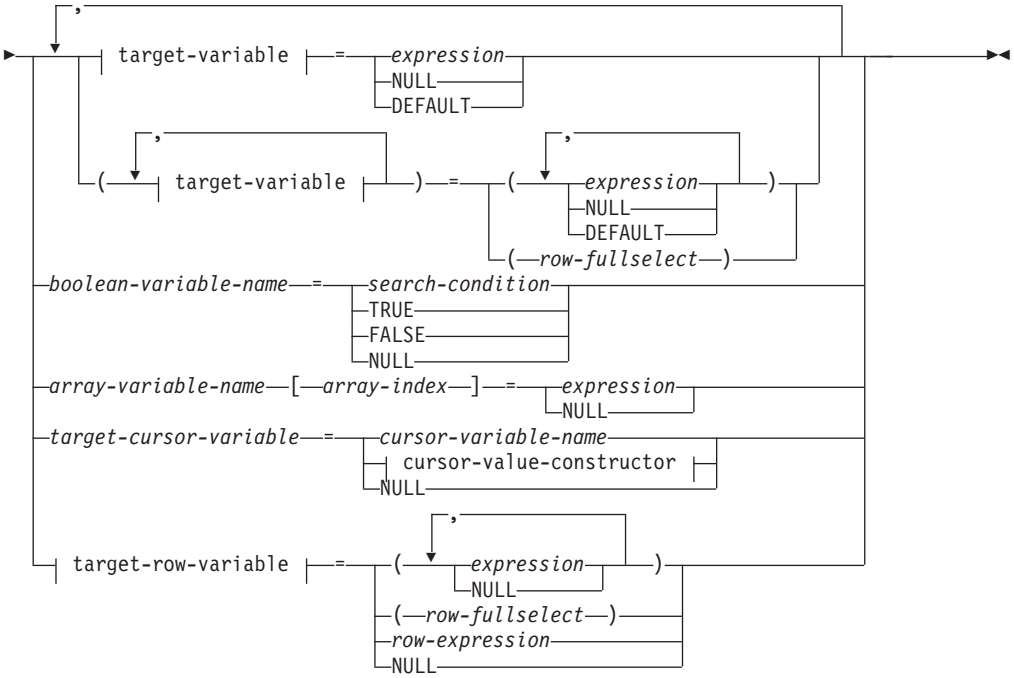
전역 변수가 지정 명령문의 왼쪽에서 참조되면 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 WRITE 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

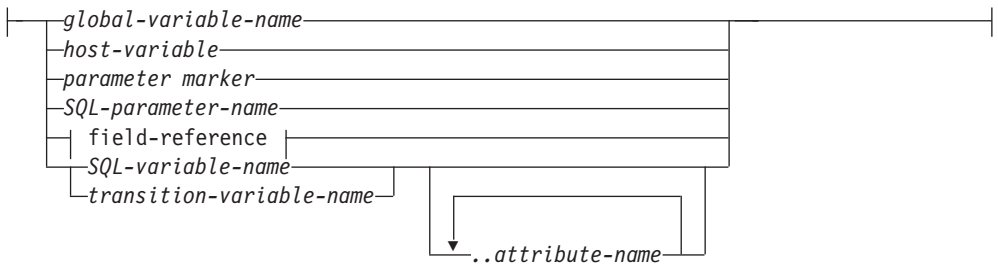
지정의 오른쪽으로 *row-fullselect*가 있는 이 명령문을 실행하려면, 명령문의 권한 부여 ID에 의해 보유된 특권에는 *row-fullselect*를 실행하는 데 필요한 특권을 포함해야 합니다. ″SQL 쿼리″의 권한 부여 섹션을 참조하십시오.

*select-statement*를 사용하는 *cursor-value-constructor*가 있는 이 명령문을 실행하려면, 명령문의 권한 부여 ID에 의해 보유된 특권에는 *select-statement*를 실행하는 데 필요한 특권을 포함해야 합니다. ″SQL 쿼리″의 권한 부여 섹션을 참조하십시오.

구문

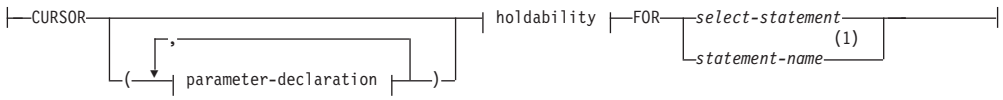


target-variable:

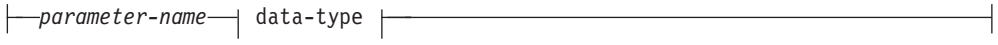


field-reference:

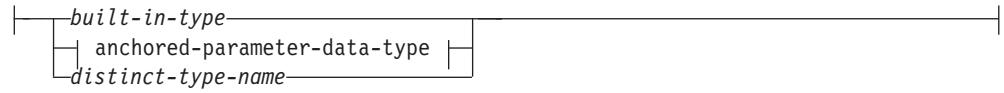
cursor-value-constructor:



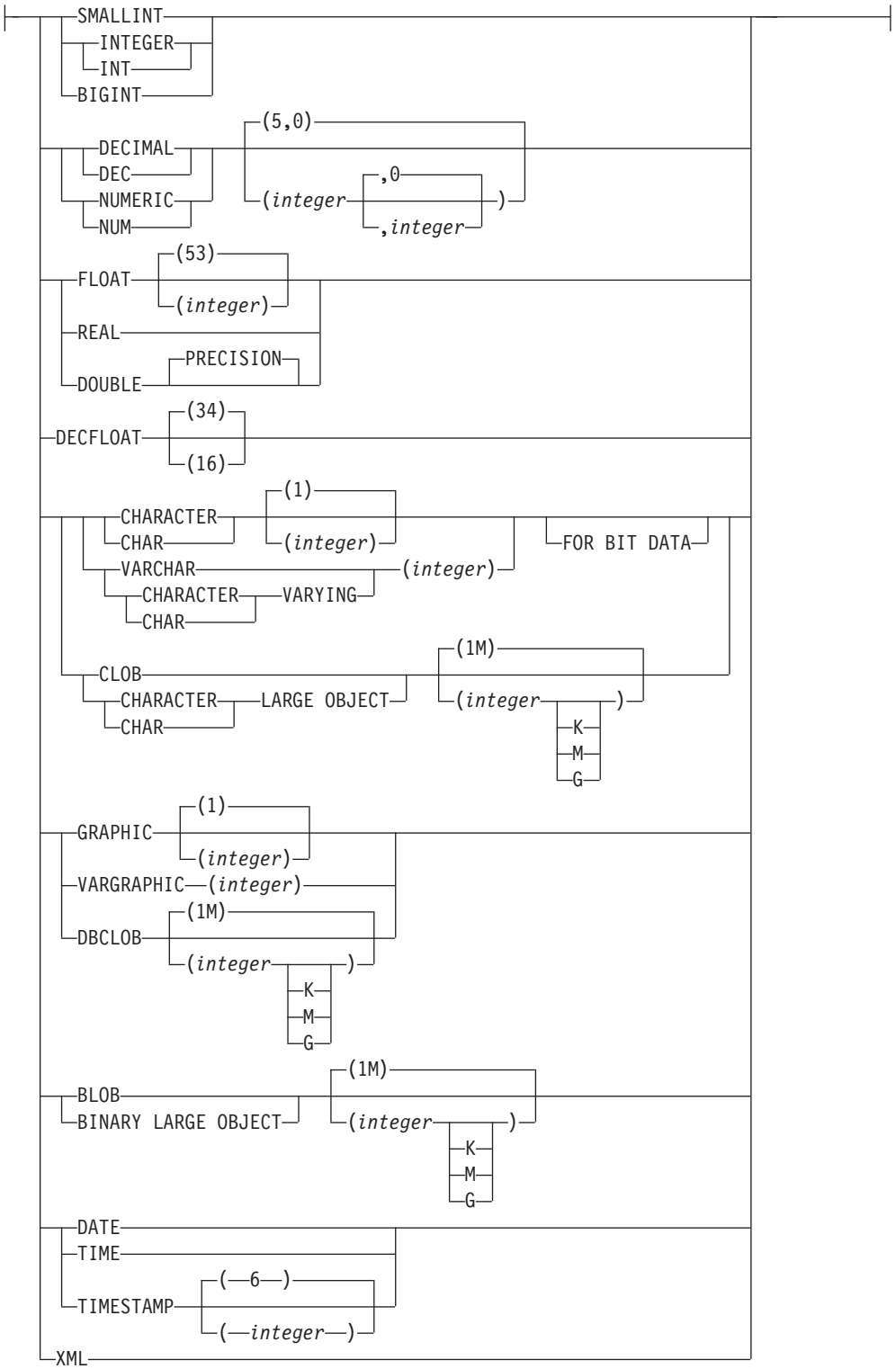
parameter-declaration:



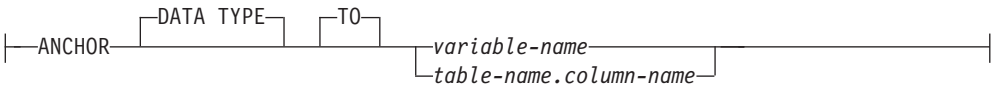
data-type:


built-in-type
anchored-parameter-data-type
distinct-type-name

built-in-type:



anchored-parameter-data-type:



유지 가능성:

target-row-variable:

주:

1 *parameter-declaration*이 지정된 경우, *statement-name*을 지정할 수 없습니다.

2 데이터 유형은 행 유형이어야 합니다.

설명

target-variable

할당 목표 변수를 식별합니다. 동일한 변수를 표현하는 *target-variable*이 두 번 이상 지정되면 안됩니다(SQLSTATE 42701).

global-variable-name

지정 목표인 전역 변수를 식별합니다. *global-variable-name*은 현재 서버에 존재하는 전역 변수를 식별해야 합니다(SQLSTATE 42704).

host-variable

지정 목표인 호스트 변수를 식별합니다.

parameter-marker

지정 목표인 매개변수 표시문자를 식별합니다.

SQL-parameter-name

지정 목표인 매개변수를 식별합니다. 매개변수를 CREATE PROCEDURE문의 *parameter-declaration*에 지정해야 하고 OUT 또는 INOUT 매개변수로 정의해야 합니다.

field-reference

지정 목표인 행 유형 값 내에서 필드를 식별합니다.

row-variable-name

행 유형인 데이터 유형을 가진 변수의 이름.

field-name
행 유형 내의 필드 이름.

SQL-variable-name
지정 목표인 SQL 변수를 식별합니다. SQL 변수는 사용하기 전에 선언해야 합니다.

transition-variable-name
전이 행에서 갱신될 컬럼을 식별합니다. *transition-variable-name*은 새로운 값을 식별하는 상관 이름에 의해 선택적으로 규정되는 트리거의 주제 테이블에 있는 컬럼을 식별해야 합니다(SQLSTATE 42703).

..attribute-name
설정된 구조화된 유형의 속성을 지정합니다(*attribute assignment*라고도 함). *SQL-variable-name* 또는 *transition-variable-name*을 사용자 정의 구조화된 유형으로 정의해야 합니다(SQLSTATE 428DP). *..attribute-name*은 구조화된 유형의 속성이어야 합니다(SQLSTATE 42703). *..attribute-name*절을 포함하지 않는 지정을 *기본 지정*이라고도 합니다.

expression
지정 목표의 새 값을 표시합니다. 표현식은 "표현식"에 설명되어 있는 모든 유형의 표현식입니다. 표현식에는 스칼라 fullselect 내에서 발생할 때를 제외하고 집계 함수를 포함시킬 수 없습니다(SQLSTATE 42903). CREATE TRIGGER문의 컨텍스트에서 *expression*은 OLD 및 NEW 전이 변수 참조를 포함할 수 있습니다. 트랜잭션 변수는 *correlation-name*으로 규정해야 합니다(SQLSTATE 42702).

NULL
널(NULL) 값을 지정합니다. 지정의 목표가 행 변수인 경우, 각 필드는 널(NULL) 값으로 지정됩니다. 널(NULL)은 특별히 속성의 데이터 유형으로 캐스트되지 않을 경우 속성 지정의 값이 널(NULL)이 될 수 없습니다(SQLSTATE 429B9).

DEFAULT
디폴트값이 사용되도록 지정합니다.

SQL 프로시저에서, DEFAULT절은 정적 SQL문에서만 지정될 수 있습니다. 예외로 *target-variable*이 동적 SQL문의 전역 변수일 때 DEFAULT절을 지정할 수 있습니다.

*target-variable*이 컬럼인 경우 이 값은 테이블에 컬럼이 정의되는 방식에 따라 삽입됩니다.

- WITH DEFAULT절을 사용하여 컬럼이 정의된 경우, 값은 컬럼에 정의된 디폴트값으로 설정됩니다("ALTER TABLE"의 *default-clause* 참조).
- IDENTITY절을 사용하여 컬럼을 정의하면 데이터베이스 관리 프로그램이 값을 생성합니다.

- WITH DEFAULT절, IDENTITY절 또는 NOT NULL절을 지정하지 않고 컬럼이 정의되었다면 값은 널(NULL)입니다.
- NOT NULL절을 사용하여 정의된 컬럼인 경우 다음과 같습니다.
  - IDENTITY절은 사용되지 않음
  - WITH DEFAULT절은 사용되지 않음
  - DEFAULT NULL이 사용됨

  DEFAULT 키워드는 해당 컬럼에 지정될 수 없습니다(SQLSTATE 23502).

*target-variable*이 SQL 변수인 경우 삽입되는 값은 변수 선언에 지정되거나 내재되는 디폴트값입니다.

*target-variable*이 전역 변수이면 삽입된 값은 변수 작성에서 지정한 대로 디폴트값입니다.

*target-variable*이 SQL 프로시저에서 SQL 변수 또는 SQL 매개변수이면, 호스트 변수 또는 매개변수 표시문자인 DEFAULT 키워드를 지정할 수 없습니다(SQLSTATE 42608).

row-fullselect

지정(assignment)에 지정된 목표 변수 또는 행 변수의 필드의 수와 일치하는 컬럼 수로 단일 행을 리턴하는 fullselect입니다. 각각의 해당 목표 변수 또는 필드에 값이 지정됩니다. fullselect 행의 결과가 행이 없는 경우 널(NULL) 값이 목록의 목표 변수에 지정되거나 또는 행 변수에 지정하는 경우 단일 널(NULL)이 지정됩니다. CREATE TRIGGER문의 컨텍스트에서 *row-fullselect*는 사용할 전이 변수를 지정하기 위해 *correlation-name*에 의해 규정되어야 하는 OLD 및 NEW 전이 변수에 대한 참조를 포함할 수 있습니다(SQLSTATE 42702). 결과에 한 행 이상이 있는 경우 오류가 발생합니다(SQLSTATE 21000).

boolean-variable-name

SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다. 변수 또는 매개변수는 부울 유형이어야 합니다(SQLSTATE 428H0). SET문은 복합 SQL(컴파일된)문 내에 발행해야 합니다(SQLSTATE 428H2).

search-condition

결과가 참, 거짓 또는 알 수 없음인 검색 조건입니다. 알 수 없는 결과가 부울 값 널(NULL)로 리턴됩니다.

TRUE

부울 값 TRUE를 지정합니다.

FALSE

부울 값 FALSE를 지정합니다.

NULL

부울 값 NULL을 지정합니다.

array-variable-name

배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수(SQLSTATE 428H0).

[*array-index***]**

지정의 목표가 되는 배열에 있는 요소를 지정하는 표현식. 일반 배열의 경우, array-index는 INTEGER(SQLSTATE 22018 또는 428H1)에 지정 가능해야 합니다. 배열에 정의된 1과 최대 카디널리티 사이의 값이어야 하며 널(NULL) 값이 될 수 없습니다(SQLSTATE 2202E).

연관된 배열의 경우, 배열 인덱스 표현식은 연관된 배열의 인덱스 데이터 유형(SQLSTATE 22018 또는 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다.

target-cursor-variable

커서 변수를 식별합니다. *target-cursor-variable*의 데이터 유형은 커서 유형이어야 합니다(SQLSTATE 42821).

cursor-variable-name

*target-cursor-variable*과 동일한 커서 유형의 커서 변수를 식별합니다.

cursor-value-constructor

*cursor-value-constructor*는 목표 변수와 연관된 *select-statement*를 지정합니다. *cursor-value-constructor*를 커서 변수에 지정하면 해당 커서 변수의 기본 커서를 정의합니다.

(parameter-declaration, ...)

각 매개변수의 이름 및 데이터 유형을 포함하는 커서의 입력 매개변수를 지정합니다. *cursor-value-constructor*에 *select-statement*도 지정되면 이름 지정된 입력 매개변수만 지정할 수 있습니다(SQLSTATE 428HU).

parameter-name

*select-statement*내에서 SQL 변수로 사용되는 cursor 매개변수의 이름을 지정합니다. 이름은 커서의 다른 매개변수 이름과 동일할 수 없습니다. 또한 매개변수 이름 전에 컬럼 이름이 분석되므로 *select-statement*에서 사용할 수 있는 컬럼 이름을 피하도록 이름을 선택해야 합니다.

data-type

*select-statement*내에서 사용되는 cursor 매개변수의 데이터 유형을 지정합니다. 구조화된 유형 및 참조 유형을 지정할 수 없습니다(SQLSTATE 429BB).

built-in-type

내장 데이터 유형을 지정합니다. 각 내장 데이터 유형에 대한 자세한 설명은 “CREATE TABLE”을 참조하십시오.

anchored-parameter-data-type

cursor 매개변수의 데이터 유형을 판별하는 데 사용되는 다른 오브젝

트를 식별합니다. 앵커 오브젝트의 데이터 유형은 데이터 유형을 직접 지정할 때 적용되는 것과 동일한 제한사항에 의해 바운드됩니다.

ANCHOR DATA TYPE TO
데이터 유형을 지정하기 위해 앵커된 데이터 유형이 사용됨을 표시합니다.

variable-name
로컬 SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다. 참조된 변수의 데이터 유형은 cursor 매개변수의 데이터 유형으로 사용됩니다.

table-name.column-name
기존 테이블 또는 뷰의 컬럼 이름을 식별합니다. 컬럼의 데이터 유형은 cursor 매개변수의 데이터 유형으로 사용됩니다.

distinct-type-name
구별 유형의 이름을 지정합니다. 스키마 이름 없이 *distinct-type-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 구별 유형이 분석됩니다.

holdability
커서가 커미트 조작의 결과로 닫히지 않도록 할지 여부를 지정합니다. 자세한 정보는 "DECLARE CURSOR"를 참조하십시오. 디폴트값은 WITHOUT HOLD입니다.

WITHOUT HOLD
커서가 커미트 조작의 결과로 닫히도록 합니다.

WITH HOLD
여러 작업 단위에서 자원을 유지보수합니다. 커서가 커미트 조작의 결과로 닫히지 않도록 합니다.

SELECT
커서의 SELECT문을 지정합니다. 자세한 정보는 "select-statement"를 참조하십시오. *parameter-declaration*이 *cursor-value-constructor*에 포함되면, *select-statement*가 로컬 SQL 변수 또는 루틴 SQL 매개변수를 포함해서는 안됩니다(SQLSTATE 42704).

statement-name
커서의 준비된 *select-statement*을 지정합니다. PREPARE문에 대한 설명은 "PREPARE"를 참조하십시오. 목표 커서 변수에 강하게 유형 지정된 사용자 정의 커서 유형인 데이터 유형이 있어서는 안됩니다(SQLSTATE 428HU). *statement-name*이 지정된 경우, 이름 지정된 입력 매개변수는 *cursor-value-constructor*에 지정되어서는 안 됩니다(SQLSTATE 428HU).

target-row-variable

지정의 목표 행 변수를 식별합니다. 데이터 유형은 행 유형이어야 합니다.

row-expression

지정 목표의 새 행 값을 지정합니다. "행 표현식"에 설명되어 있는 모든 유형의 표현식이 될 수 있습니다. 행의 필드 수는 지정 목표와 일치해야 하며 행의 각 필드를 지정 목표의 해당 필드에 지정할 수 있어야 합니다. 소스 및 목표 값이 사용자 정의 행 유형인 경우 유형 이름이 같아야 합니다(SQLSTATE 42821).

규칙

- 표현식으로부터 지정될 값, NULL, DEFAULT 또는 *row-fullselect*의 수는 할당 지정된 *target-variables*의 수에 일치해야 합니다(SQLSTATE 42802).
- SET variable 명령문은 한 명령문에 SQL 변수와 전이 변수를 지정할 수 없습니다(SQLSTATE 42997).
- 전역 변수는 트리거, 함수, 메소드 또는 복합 SQL(인라인된)문, 또는 이러한 오브젝트 중 하나에서 직접적 또는 간접적으로 호출되는 프로시저 내에서 지정할 수 없습니다(SQLSTATE 428GX).
- 지정된 값은 배열 컨스트럭터 또는 ARRAY_AGG의 배열이면, 배열 및 목표 변수의 기본 유형은 동일해야 합니다(SQLSTATE 42821).
- ***앵커된 데이터 유형 사용:*** 앵커된 데이터 유형은 다음을 참조할 수 없습니다(SQLSTATE 428HS). 예를 들어, 별칭, 유형이 지정된 테이블, 유형이 지정된 뷰, 선언된 임시 테이블, 약하게 유형이 지정된 커서와 연관된 행 정의, 데이터베이스 코드 페이지 또는 데이터베이스 조합과 다른 코드 페이지나 조합을 사용하는 오브젝트 등이 있습니다.
- ***커서 변수를 포함한 지정:*** 커서 변수 컨스트럭터의 값으로 설정된 커서 변수를 참조하는 지정은 복합 SQL(컴파일된)문에서만 사용할 수 있습니다. 커서 변수를 사용한 OPEN문은 지정과 동일한 범위 내에 있어야 합니다(SQLSTATE 51044).

주

- 값은 특정 지정 규칙에 따라 목표 변수에 지정됩니다.
- ***SQL 프로시저의 지정 명령문:*** SQL 프로시저의 지정 명령문은 SQL 지정 규칙에 따라야 합니다. 문자열 지정은 검색 지정 규칙을 사용합니다.
- ***배열 요소의 지정:*** 지정이 `SET A[idx] = rhs`의 양식인 경우, `A`는 배열 변수 이름이고 `idx`는 array-index로 사용되는 표현식이며 `rhs`는 배열 요소와 동일한 유형의 표현식입니다.

  1. 배열 `A`는 널(NULL) 값이면 `A`를 빈 배열로 설정합니다.
  2. `C`를 배열 `A`의 카디널리티가 되도록 합니다.
  3. `A`가 일반 배열인 경우:

- idx가 C와 같거나 작으면, idx로 식별되는 위치의 값은 rhs의 값으로 교체됩니다.
- idx가 C보다 크면, 다음과 같습니다.
  - *C*보다 크고 *idx*보다 작은 i의 경우, 위치 i에서의 값은 널(null) 값으로 설정됩니다.
  - 위치 idx의 값은 rhs의 값으로 설정됩니다.
  - A의 카디널리티는 idx로 설정됩니다.

4. A가 연관된 배열인 경우:
   - idx가 기존 배열 인덱스 값과 일치하는 경우, 배열 인덱스 idx가 있는 요소 값은 rhs 값으로 교체됩니다.
   - idx가 기존 배열 인덱스 값과 일치하지 않으면:
     - A의 카디널리티는 1씩 증가합니다.
     - 새 요소 값은 배열 인덱스 값 idx와 연관된 rhs로 설정됩니다.
5. idx가 C와 같거나 작으면, idx로 식별되는 위치의 값은 rhs의 값으로 교체됩니다.
6. idx가 C보다 크면, 다음과 같습니다.
   a. *C*보다 크고 *idx*보다 작은 i의 경우, 위치 i에서의 값은 널(null) 값으로 설정됩니다.
   b. 위치 idx의 값은 rhs의 값으로 설정됩니다.
   c. A의 카디널리티는 idx로 설정됩니다.

- 변수가 특수 레지스터의 이름(예:PATH)과 일치하는 ID로 선언된 경우에는 변수를 특수 레지스터로 우연히 지정되는 것을 방지하도록 분리해야 합니다(예를 들어, 정수로 선언된 PATH라는 변수에 대한 SET "PATH" = 1;).
- 하나 이상의 할당이 포함된 경우 각 *expression* 및 *row-fullselect*는 할당되기 전에 평가됩니다. 따라서 표현식이나 행 fullselect의 목표 변수에 대한 참조는 항상 단일 SET문에 있는 모든 지정 이전의 목표 변수 값입니다.
- 구별 유형으로 정의된 식별 컬럼이 갱신된 경우 전체 계산이 소스 유형으로 수행되고 그 값이 실제로 컬럼에 지정되기 전에 결과가 구별 유형으로 캐스트됩니다. (이전 값은 계산에 앞서 소스 유형으로 캐스팅되지 않습니다.)
- 데이터베이스 관리 프로그램이 ID 컬럼에 대한 SET문의 값을 생성하도록 하려면 DEFAULT 키워드를 사용하십시오.

```
SET NEW.EMPNO = DEFAULT
```

  이 예에서 NEW.EMPNO는 ID 컬럼으로 정의되고 이 컬럼을 갱신하는 데 사용되는 값은 데이터베이스 관리 프로그램이 생성합니다.
- ID 컬럼에 대해 생성된 시퀀스 값 소비에 대한 자세한 정보와 ID 컬럼의 최대값 초과에 대한 정보는 “INSERT”를 참조하십시오.

예:

예 1: 트리거 조치가 현재 실행 중인 행의 급여 컬럼을 50000으로 설정하십시오.

```
SET NEW_VAR.SALARY = 50000;
```

또는

```
SET (NEW_VAR.SALARY) = (50000);
```

예 2: 트리거 조치가 현재 실행 중인 행의 급여 및 상여금 컬럼을 각각 50000과 8000으로 설정하십시오.

```
SET NEW_VAR.SALARY = 50000, NEW_VAR.COMM = 8000;
```

또는

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (50000, 8000);
```

예 3: 트리거 조치가 현재 실행 중인 행의 급여 및 상여금 컬럼을 갱신된 행과 연관된 부서의 직원 급여 평균과 상여금 평균으로 설정하십시오.

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM)
   = (SELECT AVG(SALARY), AVG(COMM)
        FROM EMPLOYEE E
        WHERE E.WORKDEPT = NEW_VAR.WORKDEPT);
```

예 4: 트리거 조치가 현재 실행 중인 행의 급여 및 상여금 컬럼을 각각 10000과 급여의 원래 값(SET문 실행 전)으로 설정하십시오.

```
SET NEW_VAR.SALARY = 10000, NEW_VAR.COMM = NEW_VAR.SALARY;
```

또는

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (10000, NEW_VAR.SALARY);
```

예 5: SQL 변수 P_SALARY를 10% 증가시키십시오.

```
SET P_SALARY = P_SALARY + (P_SALARY * .10)
```

예 6: SQL 변수 P_SALARY를 널(NULL) 값으로 설정하십시오.

```
SET P_SALARY = NULL
```

예 7: 숫자 2.71828183 및 3.1415926을 배열 변수 SPECIALNUMBERS의 첫 번째 및 열 번째 요소에 지정하십시오. 첫 번째 지정 후에 P_PHONENUMBERS의 카디널리티(cardinality)는 1입니다. 두 번째 지정 후에 카디널리티는 10이며 요소 2 - 9에는 내재적으로 널(null) 값이 지정됩니다.

```
SET SPECIALNUMBERS[1] = 2.71828183;

SET SPECIALNUMBERS[10] = 3.14159265;
```

예 8: 데이터베이스에 연결할 수 있는 모든 사용자의 행이 있는 이름이 SECURITY.USERS인 테이블이 제공되면 현재 시간 및 권한 부여 레벨을 전역 변수 USERINFO.GV_CONNECT_TIME 및 USERINFO.GV_AUTH_LEVEL에 각각 지정하십시오.

```
SET USERINFO.GV_CONNECT_TIME = CURRENT TIMESTAMP,
  USERINFO.GV_AUTH_LEVEL = (
    SELECT AUTHLEVEL FROM SECURITY.USERS
      WHERE USERID = CURRENT USER)
```

예 9: 값을 배열 유형 CAPITALSARRAY로 선언된 연관된 배열 변수인 CAPITALS에 지정하십시오.

```
SET CAPITALS['British Columbia'] = 'Victoria';
SET CAPITALS['Alberta'] = 'Edmonton';
SET CAPITALS['Manitoba'] = 'Winnipeg';
SET CAPITALS['Canada'] = 'Ottawa';
```

CAPITALS 배열의 데이터를 채울 때, 배열 인덱스는 문자열로 지정된 지역, 지역 및 국가 이름이고 연관된 배열 요소는 수도이며, 이 또한 문자열별로 지정됩니다.

예 10: 기억하기 쉬운 이름을 배열 유형 PERSONAL_PHONENUMBERS의 배열 변수 PHONELIST에 저장된 개인용 전화번호 인덱스로 지정하십시오.

```
SET PHONELIST['Home'] = '4163053745';
SET PHONELIST['Work'] = '4163053746';
SET PHONELIST['Mom']  = '4164789683';
```

SIGNAL

SIGNAL문은 오류 또는 경고 조건의 신호를 보내는 데 사용됩니다. 이로 인해 선택적 메시지 텍스트, 지정된 SQLSTATE와 함께 오류나 경고가 리턴됩니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

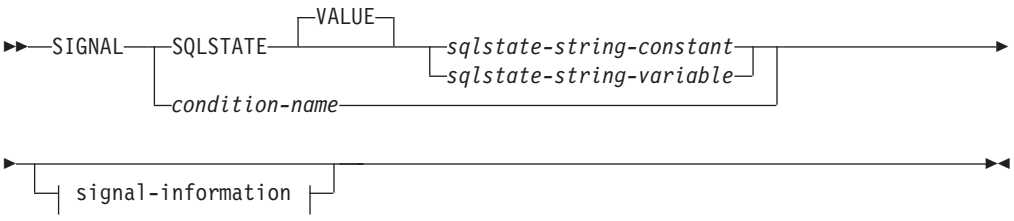
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

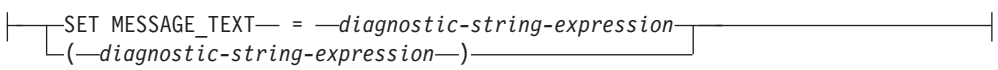
권한 부여

모듈 조건이 참조되는 경우, 명령문의 권한 부여 ID가 보유하는 특권은 모듈의 EXECUTE 특권을 포함해야 합니다.

구문



signal-information:



설명

SQLSTATE VALUE

리턴될 SQLSTATE를 지정합니다. 유효한 모든 SQLSTATE 값이 사용될 수 있습니다. 지정된 값은 SQLSTATE의 규칙을 따라야 합니다.

- 각 문자는 숫자 세트('0'- '9')이거나 발음 구별 부호가 없는 대문자(A- Z)여야 합니다.
- SQLSTATE 클래스(처음 두 문자)는 '00'(성공적인 완료)일 수 없습니다.

복합 SQL(인라인된)문 중 하나의 컨텍스트에서 다음 규칙도 적용되어야 합니다.

- SQLSTATE 클래스(처음 두 자)는 오류 클래스가 아니므로 '01' 또는 '02'가 될 수 없습니다.
- SQLSTATE 클래스가 '0'에서 '6'까지의 숫자 또는 'A'에서 'H'까지의 문자로 시작하는 경우, 서브클래스(마지막 3자)는 'I' - 'Z' 범위의 문자로 시작해야 합니다.
- SQLSTATE 클래스가 숫자 '7', '8', '9' 또는 'I'에서 'Z'까지의 문자로 시작하는 경우, 서브클래스는 '0' - '9' 또는 'A' - 'Z'일 수 있습니다.

SQLSTATE가 이 규칙에 따르지 않으면, 오류가 리턴됩니다.

sqlstate-string-constant

*sqlstate-string-constant*는 정확히 5자로 된 문자열 상수여야 합니다.

sqlstate-string-variable

지정된 SQL 변수 또는 SQL 매개변수는 데이터 유형 CHAR(5)이어야 합니다.

condition-name

리턴될 조건의 이름을 지정합니다. compound-statement내에 선언되거나 현재 서버에 있는 조건을 식별해야 합니다(SQLSTATE 42379).

SET MESSAGE_TEXT =

오류나 경고를 설명하는 문자열을 지정합니다. 이 문자열은 SQLCA의 SQLERRMC 필드에 리턴됩니다. 실제 문자열이 70바이트보다 큰 경우에는 경고없이 절단됩니다.

diagnostic-string-expression

오류 조건을 설명하는 리터럴 문자열 또는 로컬 변수나 매개변수. 문자열이 70바이트보다 크면 절단됩니다.

(*diagnostic-string-expression*)

오류 조건을 서술하기 위해 최대 70바이트의 문자열을 리턴하는 CHAR 또는 VARCHAR 유형을 가진 표현식입니다. 문자열이 70바이트보다 크면 절단됩니다. 이 옵션은 DB2의 이전 버전과의 호환성을 위해 CREATE TRIGGER문의 범위 내에서만 제공됩니다. 일반 사용은 권장되지 않습니다.

주

- SIGNAL문이 연관된 SQLSTATE 값이 없는 *condition-name*을 사용하여 발행되고 조건이 처리되지 않으면, SQLSTATE 45000이 리턴되며 SQLCODE가 -438로 설정됩니다. 이러한 조건은 SIGNAL문을 발행하는 루틴의 범위 내에 있는 SQLSTATE 45000에 대한 조건 핸들러에 의해 처리되지 않음에 유의하십시오.
- SQLSTATE 값 또는 연관된 SQLSTATE 값이 있는 *condition-name*을 사용하여 SIGNAL문을 발행한 경우, 리턴된 SQLCODE는 다음과 같은 SQLSTATE 값을 기반으로 합니다.

  - 지정된 SQLSTATE 클래스가 '01'이나 '02' 중 하나가 아니면 경고나 찾을 수 없음 조건이 리턴되고 SQLCODE가 +438로 설정됩니다.
  - 그렇지 않으면 예외 조건이 리턴되고 SQLCODE가 -438로 설정됩니다.
- SIGNAL문은 SQLCA의 표시된 필드를 다음과 같이 설정합니다.
  - sqlerrd 필드를 0으로 설정
  - sqlwarn 필드를 공백으로 설정
  - sqlerrmc를 MESSAGE_TEXT의 처음 70바이트로 설정
  - sqlerrml이 sqlerrmc의 길이로 설정되거나 SET MESSAGE_TEXT절이 지정되지 않은 경우에는 0으로 설정
  - sqlerrp를 ROUTINE으로 설정
- SQLSTATE 값은 두 문자 클래스 코드 값과 세 문자 서브클래스 코드 값으로 구성됩니다. 클래스 코드 값은 성공 및 실패 실행 조건의 클래스를 나타냅니다.

  유효한 모든 SQLSTATE 값은 SIGNAL문에서 사용될 수 있습니다. 그러나 프로그래머는 응용프로그램용으로 예약된 범위를 근거로 새 SQLSTATE를 정의하는 것이 바람직합니다. 이를 통해 추후 릴리스에서 데이터베이스 관리 프로그램이 잘못된 SQLSTATE 값을 정의하는 것을 막을 수 있습니다.
  - '7' - '9' 또는 'I' - 'Z' 문자로 시작하는 SQLSTATE 클래스를 정의할 수 있습니다. 이들 클래스 내에 있는 모든 서브클래스가 정의됩니다.
  - '0' - '6' 또는 'A' - 'H' 문자로 시작하는 SQLSTATE 클래스는 데이터베이스 관리 프로그램용으로 예약되어 있습니다. 이러한 클래스에서는 '0' - 'H' 문자로 시작하는 서브클래스가 데이터베이스 관리 프로그램용으로 예약되어 있습니다. 'I' - 'Z' 문자로 시작하는 서브클래스를 정의할 수 있습니다.

예:

응용프로그램에 고객 번호가 알려져 있지 않은 경우 응용프로그램 오류 신호를 보내는 주문 시스템용 SQL 프로시저입니다. ORDERS 테이블에는 CUSTOMER 테이블에 대한 외부 키가 포함되며 주문을 삽입하기 위해서는 CUSTNO가 필요합니다.

```
CREATE PROCEDURE SUBMIT_ORDER
       (IN ONUM INTEGER, IN CNUM INTEGER,
  IN PNUM INTEGER, IN QNUM INTEGER)
  SPECIFIC SUBMIT_ORDER
  MODIFIES SQL DATA
      LANGUAGE SQL
BEGIN
  DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
    SIGNAL SQLSTATE '75002'
      SET MESSAGE_TEXT = 'Customer number is not known';
  INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
    VALUES (ONUM, CNUM, PNUM, QNUM);
END
```

TRANSFER OWNERSHIP

TRANSFER OWNERSHIP문은 데이터베이스 오브젝트의 소유권을 전송합니다.

호출

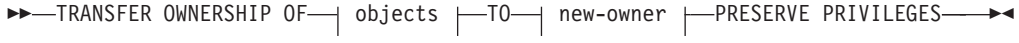
이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

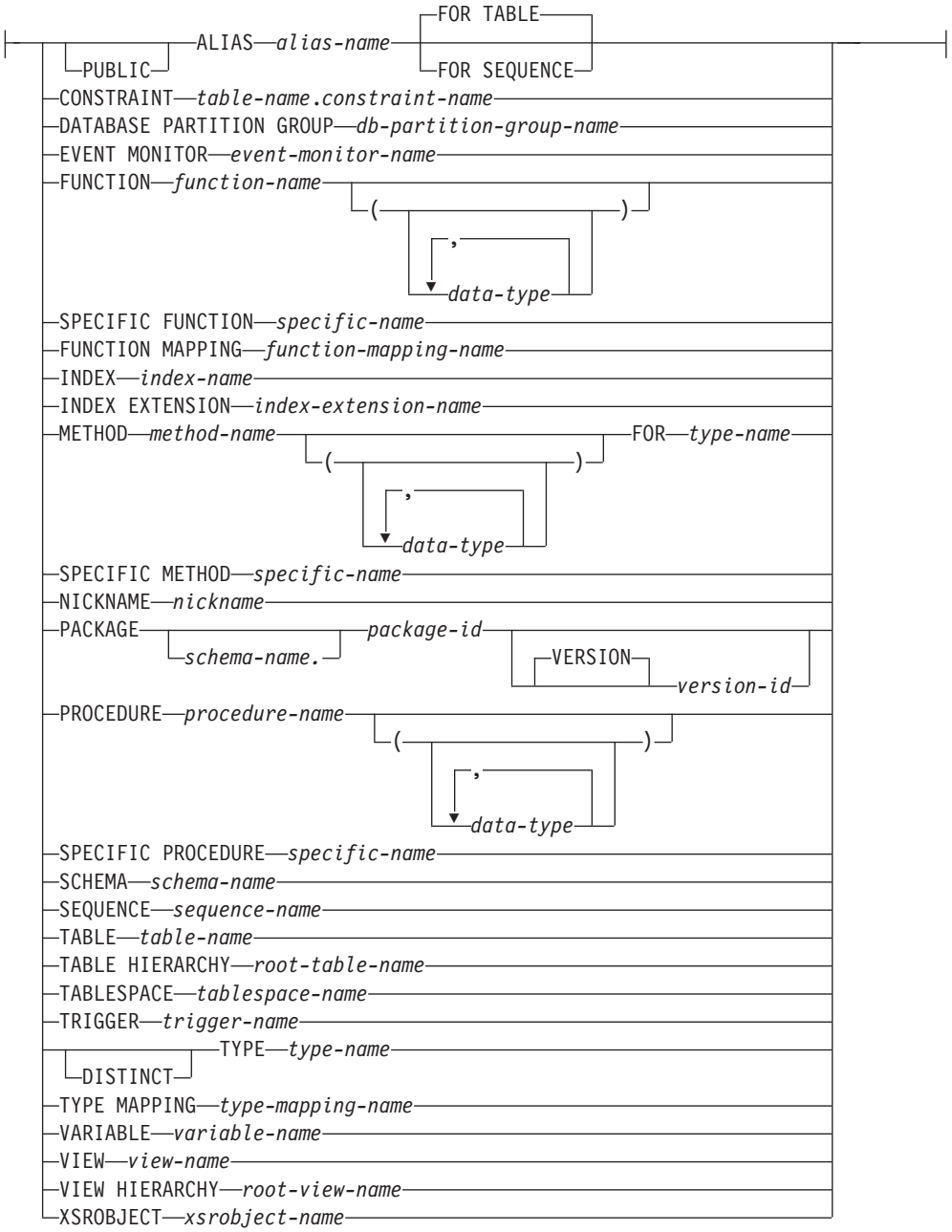
명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 오브젝트 소유권
- SECADM 권한

구문



오브젝트:



new-owner:

설명

ALIAS *alias-name*

해당 소유권을 전송할 별명을 식별합니다. *alias-name*은 카탈로그에서 설명되는 별명을 식별해야 합니다(SQLSTATE 42704). PUBLIC이 지정된 경우, *alias-name*은 현재 서버에 존재하는 공용 별명을 식별해야 합니다(SQLSTATE 42704).

FOR TABLE, 또는 FOR SEQUENCE
별명의 오브젝트 유형을 지정합니다.

FOR TABLE
별명은 테이블, 뷰 또는 별칭에 대한 것입니다. 별명의 소유권을 전송할 때, SYSCAT.TABLES 카탈로그 뷰에서 별명에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

FOR SEQUENCE
별명은 시퀀스에 대한 것입니다. 별명 소유권을 전송할 때, SYSCAT.SEQUENCES 카탈로그 뷰의 별명에 대한 OWNER 컬럼 값이 새 소유자의 권한 부여 ID로 교체됩니다.

CONSTRAINT *table-name.constraint-name*
해당 소유권을 전송할 제한조건을 식별합니다. *table-name.constraint-name* 조합은 제한조건과 해당 제한조건이 제한하는 테이블을 식별해야 합니다. *constraint-name* 은 카탈로그에 기술된 제한조건을 식별해야 합니다(SQLSTATE 42704).

제한조건의 소유권을 전송할 때, SYSCAT.TABCONST 카탈로그 뷰에서 제한조건에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

- 제한조건이 FOREIGN KEY 제한조건일 경우, SYSCAT.REFERENCES 카탈로그 뷰의 OWNER 컬럼이 새 소유자의 권한 부여 ID로 대체됩니다.
- 제한조건이 PRIMARY KEY 또는 UNIQUE 제한조건일 경우, 해당 제한조건에 대해 내재적으로 작성된 인덱스에 대한 SYSCAT.INDEXES 카탈로그 뷰의 OWNER 컬럼이 새 소유자의 권한 부여 ID로 대체됩니다. 인덱스가 존재하고 이러한 경우에서 재사용된 경우, 인덱스 소유자는 변경되지 않습니다.

DATABASE PARTITION GROUP *db-partition-group-name*
해당 소유권을 전송할 데이터베이스 파티션 그룹을 식별합니다. *db-partition-group-name*은 카탈로그에 기술된 데이터베이스 파티션 그룹을 식별해야 합니다(SQLSTATE 42704).

데이터베이스 파티션 그룹의 소유권을 전송할 때,
SYSCAT.DBPARTITIONGROUPS 카탈로그 뷰에서 데이터베이스 파티션 그룹에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

EVENT MONITOR *event-monitor-name*
해당 소유권을 전송할 이벤트 모니터를 식별합니다. *event-monitor-name*은 카탈로그에 기술되어 있는 이벤트 모니터를 식별해야 합니다(SQLSTATE 42704).

이벤트 모니터의 소유권을 전송할 때, SYSCAT.EVENTMONITORS 카탈로그 뷰에서 이벤트 모니터에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

식별된 이벤트 모니터가 활성 상태일 경우, 오류가 리턴됩니다(SQLSTATE 429BT).

해당 소유권을 전송할 WRITE TO FILE 이벤트 모니터의 목표 경로에 이벤트 파일이 있을 경우, 이벤트 파일은 삭제되지 않습니다.

WRITE TO TABLE 이벤트 모니터의 소유권이 전송될 때,
SYSCAT.EVENTTABLES 카탈로그 뷰의 테이블 정보가 보존됩니다.

FUNCTION

해당 소유권을 전송할 함수를 식별합니다. 지정된 함수 인스턴스는 카탈로그에 기술된 사용자 정의 함수 또는 함수 템플리트여야 합니다. CREATE TYPE(구별) 및 CREATE TYPE(구조화된)문에 의해 내재적으로 생성된 함수의 소유권은 전송할 수 없습니다(SQLSTATE 429BT).

함수 인스턴스를 식별하는 데 사용할 수 있는 방법에는 여러 가지가 있습니다.

FUNCTION *function-name*

해당 소유권을 전송할 특정 함수를 식별하며 해당 *function-name*을 갖는 함수가 정확히 하나 있는 경우에만 유효합니다. 그러므로 식별된 함수에 대해 임의의 수의 매개변수를 정의할 수 있습니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 이름이 지정된 스키마 또는 내재적 스키마에 해당 이름의 함수가 없으면, 오류가 발생합니다(SQLSTATE 42704). 이름이 지정된 스키마 또는 내재적 스키마에 함수의 특정 인스턴스가 둘 이상 있을 경우, 오류가 발생합니다(SQLSTATE 42725).

FUNCTION *function-name* **(***data-type***,...)**

해당 소유권을 삭제할 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 선택 알고리즘은 사용하지 않습니다.

function-name

해당 소유권을 전송할 함수의 이름을 지정합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다.

(data-type,...)

지정된 데이터 유형이 CREATE FUNCTION문에 지정된 유형 및 위치와 일치해야 합니다. 데이터 유형의 수와 데이터 유형의 논리적 병합을 사용하여 해당 소유권을 전송할 특정 함수를 식별합니다.

*data-type*이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다.

0 <*n*< 25는 REAL을 의미하고, 24 <*n*< 54는 DOUBLE을 의미하므로, FLOAT(*n*) 유형은 *n*에 정의된 값과 일치하지 않아도 됩니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

FOR BIT DATA 속성은 같은 목적으로 시그니처의 일부로 고려되지 않습니다. 그러므로 예를 들어, 시그니처에 지정된 CHAR FOR BIT DATA는 CHAR로만 정의된 함수와 일치하며, 그 반대도 또한 참입니다.

지정된 시그니처를 갖는 함수가 이름이 지정된 스키마 또는 내재적 스키마에 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

함수의 소유권을 전송할 때, SYSCAT.ROUTINES 카탈로그 뷰에서 함수에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

SPECIFIC FUNCTION *specific-name*

함수 작성시 지정되거나 디폴트값인 특정 이름을 사용하여 해당 소유권을 전송할 특정 사용자 정의 함수를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 함수 인스턴스를 식별해야 하며, 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

특정 함수의 소유권을 전송할 때, SYSCAT.ROUTINES 카탈로그 뷰에서 특정 함수에 대한 OWNER 컬럼의 값이 새 사용자의 권한 부여 ID로 대체됩니다.

FUNCTION MAPPING *function-mapping-name*

해당 소유권을 전송할 함수 맵핑을 식별합니다. *function-mapping-name*은 카탈로그에 기술되어 있는 함수 맵핑을 식별해야 합니다(SQLSTATE 42704).

함수 맵핑의 소유권을 전송할 때, SYSCAT.FUNCMAPPINGS 카탈로그 뷰에서 함수 맵핑에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

INDEX *index-name*

해당 소유권을 전송할 인덱스 또는 인덱스 스펙을 식별합니다. *index-name*은 카탈로그에 기술되어 있는 인덱스 또는 인덱스 스펙을 식별해야 합니다(SQLSTATE 42704).

인덱스의 소유권을 전송할 때, SYSCAT.INDEXES 카탈로그 뷰에서 인덱스에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

인덱스가 정의된 테이블이 전역 임시 테이블일 경우 인덱스의 소유권을 전송할 수 없습니다(SQLSTATE 429BT).

INDEX EXTENSION *index-extension-name*

해당 소유권을 전송할 인덱스 확장을 식별합니다. *index-extension-name*은 카탈로그에 기술되어 있는 인덱스 확장을 식별해야 합니다(SQLSTATE 42704).

인덱스 확장의 소유권을 전송할 때, SYSCAT.INDEXEXTENSIONS 카탈로그 뷰에서 인덱스 확장에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

METHOD

해당 소유권을 전송할 메소드를 식별합니다. 지정된 메소드 본문은 카탈로그에 기술된 메소드여야 합니다(SQLSTATE 42704). CREATE TYPE문에 의해 내재적으로 생성된 메소드 소유권은 전송할 수 없습니다(SQLSTATE 429BT).

메소드 본문을 식별하는 데 사용할 수 있는 방법에는 여러 가지가 있습니다.

METHOD *method-name*

해당 소유권을 전송할 특정 메소드를 식별하며 이름이 *method-name*이고 주제 유형이 *type-name*인 메소드 인스턴스가 정확히 하나 있는 경우에만 유효합니다. 따라서 식별된 메소드는 임의의 수의 매개변수를 가질 수 있습니다. *type-name* 유형에 대해 해당 이름의 메소드가 존재하지 않으면, 오류가 발생합니다(SQLSTATE 42704). 이름 지정된 데이터 유형에 메소드의 특정 인스턴스가 둘 이상 있을 경우, 오류가 발생합니다(SQLSTATE 42725).

METHOD *method-name* **(***data-type***,...)**

해당 소유권을 전송할 메소드를 고유하게 식별하는 메소드 서명을 제공합니다. 메소드 선택 알고리즘은 사용하지 않습니다.

method-name

해당 소유권을 전송할 메소드의 이름을 지정합니다. 이 이름은 규정되지 않은 ID이어야 합니다.

(*data-type***, ...)**

지정된 데이터 유형이 CREATE TYPE 또는 ALTER TYPE문에 지정된 유형 및 위치와 일치해야 합니다. 데이터 유형의 수와 데이터 유형의 논리적 병합을 사용하여 해당 소유권을 전송할 특정 메소드 인스턴스를 식별합니다.

*data-type*이 규정되지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일이 코드화된 경우 그 값은 CREATE TYPE 문에 지정된 것과 정확히 일치해야 합니다.

0<*n*<25는 REAL을 의미하고 24<*n*<54는 DOUBLE을 의미하므로 FLOAT(*n*) 유형은 *n*에 대해 정의된 값과 일치할 필요가 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 갖는 메소드가 이름이 지정된 데이터 유형에 대해 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

FOR *type-name*

지정된 메소드가 해당 소유권을 전송할 유형의 이름을 지정합니다. 이름은 카탈로그에 이미 기술된 유형을 식별해야 합니다(SQLSTATE 42704). 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 유형 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 유형 이름의 규정자를 내재적으로 지정합니다.

메소드의 소유권을 전송할 때, SYSCAT.ROUTINES 카탈로그 뷰에서 메소드에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

SPECIFIC METHOD *specific-name*

해당 소유권을 전송할 특정 메소드를 식별합니다. 특정 이름이 규정되지 않은 경우 CURRENT SCHEMA 특수 레지스터는 동적 SQL에서 규정되지 않은 특정 이름의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. *specific-name*은 메소드를 식별해야 하며, 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

특정 메소드의 소유권을 전송할 때, SYSCAT.ROUTINES 카탈로그 뷰에서 특정 메소드에 대한 OWNER 컬럼의 값이 새 사용자의 권한 부여 ID로 대체됩니다.

NICKNAME *nickname*

해당 소유권을 전송할 별칭을 식별합니다. *nickname*은 카탈로그에 기술되어 있는 별칭이어야 합니다(SQLSTATE 42704).

별칭의 소유권을 전송할 때, SYSCAT.TABLES 카탈로그 뷰에서 별칭에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

PACKAGE *schema-name.package-id*

해당 소유권을 전송할 패키지를 식별합니다. 스키마 이름이 지정되지 않은 경우 디폴트 스키마가 패키지 ID를 내재적으로 규정합니다. 스키마 이름과 패키지 ID는 내재적으로 또는 명시적으로 지정된 버전 ID와 함께 카탈로그에 설명된 패키지를 식별해야 합니다(SQLSTATE 42704).

VERSION *version-id*

해당 소유권을 전송할 패키지 버전을 식별합니다. 값을 지정하지 않을 경우, 버전은 디폴트로 빈 문자열이 되며 해당 패키지의 소유권이 전송됩니다. 이름이 같지만 버전이 다른 여러 개의 패키지가 있을 경우, *version-id*를 TRANSFER OWNERSHIP문에 지정한 패키지의 소유권만이 전송됩니다. 다음과 같은 경우 큰따옴표를 사용하여 버전 ID를 구분하십시오.

- VERSION(AUTO) 프리컴파일러 옵션을 사용하여 생성한 경우
- 숫자로 시작되는 경우
- 소문자 또는 대소문자 혼용 문자가 있는 경우

운영 체제 명령 프롬프트에서 명령문을 호출한 경우, 각 큰따옴표 분리문자 앞에 백슬래시 문자를 사용하여 운영 체제가 분리문자를 없애지 못하게 하십시오.

패키지의 소유권을 전송할 때, SYSCAT.PACKAGES 카탈로그 뷰에서 패키지에 대한 BOUNDBY 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

SQL 프로시저와 연관된 패키지 소유권은 전송할 수 없습니다(SQLSTATE 429BT).

PROCEDURE

해당 소유권을 전송할 프로시저를 식별합니다. 지정된 프로시저 인스턴스는 카탈로그에 기술된 프로시저여야 합니다.

프로시저 인스턴스를 식별하는 데 사용할 수 있는 방법에는 여러 가지가 있습니다.

PROCEDURE *procedure-name*

해당 소유권을 전송할 특정 프로시저를 식별하며 *procedure-name*을 갖는 프로시저가 스키마에 정확히 하나 있는 경우에만 유효합니다. 그러므로 식별되는 프로시저에 대해 수에 관계없이 매개변수를 정의할 수 있습니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. 이름이 지정된 스키마나 내재적 스키마에 해당 이름의 프로시저가 존재하지 않으면 오류가 리턴됩니다(SQLSTATE 42704). 이름이 지정된 스키마 또는 내재적 스키마에 프로시저의 특정 인스턴스가 둘 이상 있을 경우, 오류가 발생합니다(SQLSTATE 42725).

PROCEDURE *procedure-name* **(***data-type***,...)**

해당 소유권을 전송할 프로시저를 고유하게 식별하는 프로시저 시그니처를 제공합니다.

procedure-name

해당 소유권을 전송할 프로시저의 프로시저 이름을 지정합니다. 동적 SQL 문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다.

(data-type,...)

지정된 데이터 유형이 CREATE PROCEDURE문에 지정된 유형 및 위치와 일치해야 합니다. 데이터 유형의 수와 데이터 유형의 논리적 병합을 사용하여 해당 소유권을 전송할 특정 프로시저를 식별합니다.

*data-type*이 규정되어 있지 않으면 SQL 경로의 스키마를 검색하여 유형 이름을 분석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시된다는 것을 나타내기 위해 빈 괄호를 사용할 수 있습니다.

매개변수 값이 서로 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용할 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일을 코딩하는 경우, 이 값은 CREATE PROCEDURE문에 지정된 것과 정확히 일치해야 합니다.

0<*n*<25는 REAL을 의미하고 24<*n*<54는 DOUBLE을 의미하므로 FLOAT(*n*) 유형은 *n*에 대해 정의된 값과 일치할 필요가 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처를 갖는 프로시저가 이름이 지정된 스키마 또는 내재적 스키마에 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42883).

프로시저의 소유권을 전송할 때, SYSCAT.ROUTINES 카탈로그 뷰에서 프로시저에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

연관된 패키지가 있는 SQL 패키지의 소유권을 전송할 때 패키지의 소유권이 또한 새 소유자에게 내재적으로 전송됩니다.

SPECIFIC PROCEDURE *specific-name*

프로시저 작성시 지정되거나 디폴트값이 특정 이름을 사용하여 해당 소유권을 전송할 특정 프로시저를 식별합니다. 동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서

QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다. *specific-name*은 이름이 지정된 스키마 또는 내재적 스키마에서 특정 프로시저 인스턴스를 식별해야 하며, 그렇지 않은 경우 오류가 발생합니다(SQLSTATE 42704).

특정 프로시저의 소유권을 전송할 때, SYSCAT.ROUTINES 카탈로그 뷰에서 특정 프로시저에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

SCHEMA *schema-name*

해당 소유권을 전송할 스키마를 식별합니다. *schema-name*은 카탈로그에 기술되어 있는 스키마를 식별해야 합니다(SQLSTATE 42704).

스키마 소유권을 전송할 때, SYSCAT.SCHEMATA 카탈로그 뷰의 스키마에 대한 OWNER 컬럼 및 DEFINER 컬럼 값이 새 소유자의 권한 부여 ID로 교체됩니다.

시스템 정의 스키마(정의자가 SYSIBM임) 소유권은 전송할 수 없습니다(SQLSTATE 42832).

SEQUENCE *sequence-name*

해당 소유권을 전송할 시퀀스를 식별합니다. *sequence-name*은 카탈로그에 기술된 시퀀스를 식별해야 합니다(SQLSTATE 42704).

시퀀스의 소유권을 전송할 때, SYSCAT.SEQUENCES 카탈로그 뷰에서 스키마에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

TABLE *table-name*

해당 소유권을 전송할 테이블을 식별합니다. *table-name*은 데이터베이스에 존재하는 테이블을 식별해야 하며(SQLSTATE 42704), 선언된 임시 테이블을 식별하지 않아야 합니다(SQLSTATE 42995).

테이블 소유권을 전송할 때:

- SYSCAT.TABLES 카탈로그 뷰에서 테이블에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.
- SYSCAT.TABDEP 카탈로그 뷰에서 테이블의 모든 종속 오브젝트에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

테이블 계층 구조에서 하위 테이블의 소유권은 전송할 수 없습니다(SQLSTATE 429BT).

페더레이티드 시스템에서 투명한 DDL을 사용하여 작성된 리모트 테이블의 소유권을 전송할 수 있습니다. 리모트 테이블의 소유권을 전송할 때 테이블과 연관된 별칭의 소유권이 전송되지 않습니다. 해당 별칭의 소유권은 TRANSFER OWNERSHIP문을 사용하여 명시적으로 전송할 수 있습니다.

TABLE HIERARCHY *root-table-name*

해당 소유권을 전송할 유형이 지정된 테이블 계층 구조에서 루트 테이블인 유형이

지정된 테이블을 식별합니다. *root-table-name*은 유형이 지정된 테이블 계층 구조에서 루트 테이블인 유형이 지정된 테이블을 식별해야 하며(SQLSTATE 428DR), 데이터베이스에 존재하는 유형이 지정된 테이블을 참조해야 합니다(SQLSTATE 42704).

테이블 계층 구조의 소유권을 전송할 때:

- SYSCAT.TABLES 카탈로그 뷰에서 루트 테이블 및 모든 하위 테이블에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.
- SYSCAT.TABDEP 카탈로그 뷰에서 테이블 및 모든 하위 테이블의 모든 종속 오브젝트에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

TABLESPACE *tablespace-name*

해당 소유권을 전송할 테이블 스페이스를 식별합니다. *tablespace-name*은 카탈로그에 기술된 테이블 스페이스를 식별해야 합니다(SQLSTATE 42704).

테이블 스페이스의 소유권을 전송할 때, SYSCAT.TABLESPACES 카탈로그 뷰의 테이블 스페이스에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

TRIGGER *trigger-name*

해당 소유권을 전송할 트리거를 식별합니다. *trigger-name*은 카탈로그에 기술되어 있는 트리거를 식별해야 합니다(SQLSTATE 42704).

트리거의 소유권을 전송할 때, SYSCAT.TRIGGERS 카탈로그 뷰에서 트리거에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

TYPE *type-name*

해당 소유권을 전송할 사용자 정의 유형을 식별합니다. *type-name*은 카탈로그에 기술된 유형을 식별해야 합니다(SQLSTATE 42704). DISTINCT를 지정할 경우, *type-name*은 카탈로그에 기술되어 있는 구별 유형을 식별해야 합니다(SQLSTATE 42704).

동적 SQL문에서는 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 프리컴파일 또는 바인드 옵션은 규정되지 않은 오브젝트 이름의 규정자를 내재적으로 지정합니다.

유형의 소유권을 전송할 때, SYSCAT.DATATYPES 카탈로그 뷰에서 유형에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

TYPE MAPPING *type-mapping-name*

해당 소유권을 전송할 사용자 정의 데이터 유형 맵핑을 식별합니다. *type-mapping-name*은 카탈로그에 기술되어 있는 데이터 유형 맵핑을 식별해야 합니다(SQLSTATE 42704).

유형 맵핑의 소유권을 전송할 때, SYSCAT.TYPEMAPPINGS 카탈로그 뷰에서 유형 맵핑에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

VARIABLE *variable-name*

소유권을 전송할 오브젝트가 작성된 전역 변수임을 표시합니다. *variable-name*은 현재 서버에 존재하는 전역 변수를 식별해야 합니다(SQLSTATE 42704).

전역 변수를 전송할 때, SYSCAT.VARIABLES 카탈로그 뷰의 전역 변수에 대한 OWNER 컬럼 값이 새 소유자의 권한 부여 ID로 교체됩니다.

VIEW *view-name*

해당 소유권을 전송할 뷰를 식별합니다. *view-name*은 데이터베이스에 존재하는 뷰를 식별해야 합니다(SQLSTATE 42704).

뷰의 소유권을 전송할 때:

- SYSCAT.VIEWS 카탈로그 뷰에서 뷰에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.
- SYSCAT.TABDEP 카탈로그 뷰에서 뷰의 모든 종속 오브젝트에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

뷰 계층 규조에서 하위 뷰의 소유권은 전송할 수 없습니다(SQLSTATE 429BT).

VIEW HIERARCHY *root-view-name*

해당 소유권을 전송할 유형이 지정된 뷰 계층 구조에서 루트 뷰인 유형이 지정된 뷰를 식별합니다. *root-view-name*은 유형이 지정된 뷰 계층 구조에서 루트 뷰인 유형이 지정된 뷰를 식별해야 하며(SQLSTATE 428DR), 데이터베이스에 존재하는 유형이 지정된 뷰를 참조해야 합니다(SQLSTATE 42704).

뷰 계층 구조의 소유권을 전송할 때:

- SYSCAT.VIEWS 카탈로그 뷰에서 루트 뷰 및 모든 하위 뷰에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.
- SYSCAT.TABDEP 카탈로그 뷰에서 뷰 및 모든 하위 뷰의 모든 종속 오브젝트에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

XSROBJECT *xsrobject-name*

해당 소유권을 전송할 XSR 오브젝트를 식별합니다. *xsrobject-name*은 카탈로그에 기술된 XSR 오브젝트를 식별해야 합니다(SQLSTATE 42704).

XSR 오브젝트의 소유권을 전송할 때, SYSCAT.XSROBJECTS 카탈로그 뷰에서 XSR 오브젝트에 대한 OWNER 컬럼의 값이 새 소유자의 권한 부여 ID로 대체됩니다.

USER *authorization-name*

오브젝트 소유권을 전송 중인 권한 부여 ID를 지정합니다.

SESSION_USER

SESSION_USER 특수 레지스터의 값을 오브젝트 소유권을 전송 중인 권한 부여 ID로 사용하도록 지정합니다.

SYSTEM_USER

SYSTEM_USER 특수 레지스터의 값을 오브젝트 소유권을 전송 중인 권한 부여 ID로 사용하도록 지정합니다.

PRESERVE PRIVILEGES

해당 소유권을 전송할 오브젝트의 현재 소유자가 전송 후 계속해서 오브젝트에 대한 기존 특권을 보유하도록 지정합니다. 예를 들어, 뷰를 작성할 때 뷰의 작성자에게 뷰여한 특권이 소유권을 다른 사용자에게로 전송한 후에도 원래 소유자에 의해 계속해서 보유됩니다.

규칙

- 대부분의 시스템 정의 오브젝트(소유자가 SYSIBM임) 소유권은 전송할 수 없습니다(SQLSTATE 42832). 그러나 OWNER 컬럼에 SYSIBM이 있거나 DEFINER 컬럼에 SYSIBM이 없는 내재적으로 작성된 스키마 오브젝트의 소유권은 전송할 수 있습니다.
- 이름이 'SYS'로 시작하는 스키마의 소유권은 전송할 수 없습니다(SQLSTATE 42832).
- 다음 오브젝트의 소유권은 명시적으로 전송할 수 없습니다(SQLSTATE 429BT).
  - 테이블 계층 구조의 하위 테이블(루트 계층 구조 테이블과 함께 전송됨)
  - 뷰 계층 구조의 하위 뷰(루트 계층 구조 뷰와 함께 전송됨)
  - 전역 임시 테이블에 정의된 인덱스
  - 사용자 정의 유형 작성시 내재적으로 작성되는 메소드 또는 함수
  - SQL 프로시저에 종속되는 패키지(SQL 프로시저와 함께 전송됨)
  - 활성 상태의 이벤트 모니터(활성 상태가 아닐 경우 전송할 수 있음)
- SECADM 권한을 갖는 권한 부여 ID는 아직 오브젝트의 소유자가 아닐 경우 오브젝트의 소유권을 자신에게 전송할 수 없습니다(SQLSTATE 42502).

주

- 오브젝트 작성시 권한 부여되는 현재 소유자가 가지고 있는 모든 특권은 새 소유자에게 전송됩니다. 현재 소유자에게 오브젝트에 대한 특권이 취소되었다가 해당 특권이 나중에 다시 부여된 경우, 특권이 전송되지 않습니다.아직 전송되지 않은 내재적으로 작성된 스키마 오브젝트의 경우, 새 소유자에게 스키마에 대한 CREATEIN, DROPIN 및 ALTERIN 특권이 부여되며 이러한 특권을 다른 사용자에게도 부여할 수 있습니다.
- 데이터베이스 오브젝트의 소유권이 전송될 때 새 소유자는 오브젝트의 종속성에 따라 표시되는 기본 오브젝트의 특권 세트를 가지고 있어야 하며 이 특권 세트는 오브

젝트의 존재를 변경하지 않고 유지보수하는 데 필요합니다. 해당 특권이 오브젝트의 존재를 유지보수하는 데 필요하지 않은 경우 새 소유자는 오브젝트를 작성하는 데 필요한 특권이 필요하지 않습니다.

예를 들면, 다음과 같습니다.

- 기본 테이블에 대한 SELECT 및 INSERT 종속성을 갖는 뷰를 가정하십시오. 소유권을 전송하려면 뷰의 새 소유자가 보유하는 특권이 적어도 SELECT(GRANT OPTION 사용 또는 사용 안함) 및 INSERT(GRANT OPTION 사용 또는 사용 안함)를 포함해야 합니다. 종속성이 SELECT WITH GRANT OPTION 및 INSERT WITH GRANT OPTION인 경우, 뷰의 새 소유자가 보유하는 특권이 적어도 SELECT WITH GRANT OPTION 및 INSERT WITH GRANT OPTION을 포함해야 합니다.
- 루틴에 대한 종속성을 갖는 뷰를 가정하십시오. 뷰의 새 소유자가 보유하는 특권이 종속 루틴에 대하여 적어도 EXECUTE를 포함해야 합니다.
- 테이블에 대한 종속성을 갖는 트리거를 가정하십시오. 트리거의 새 소유자가 보유하는 특권이 트리거의 종속성에서 표시하는 테이블에 대하여 동일한 일련의 특권을 포함해야 합니다. 트리거가 정의된 테이블에 대하여 ALTER 특권은 필요하지 않습니다.

다음 표는 다른 데이터베이스 오브젝트가 종속된 오브젝트를 기술하는 시스템 카탈로그 뷰를 나열합니다.

표 32. 다른 오브젝트가 종속된 오브젝트를 기술하는 카탈로그 뷰

데이터베이스 오브젝트	시스템 카탈로그 뷰
CONSTRAINT	SYSCAT.CONSTDEP
FUNCTION	SYSCAT.ROUTINEDEP; SYSCAT.ROUTINES(전래 함수의 경우)
INDEX	SYSCAT.INDEXDEP
INDEX EXTENSION	SYSCAT.INDEXEXTENSIONDEP
METHOD	SYSCAT.ROUTINEDEP
PACKAGE	SYSCAT.PACKAGEDEP
PROCEDURE	SYSCAT.ROUTINEDEP
TABLE	SYSCAT.TABDEP
TRIGGER	SYSCAT.TRIGDEP
VIEW	SYSCAT.TABDEP
XSROBJECT	SYSCAT.XSROBJECTDEP

다른 오브젝트에 종속된 데이터베이스 오브젝트의 소유권을 성공적으로 전송하려면, 데이터베이스 오브젝트의 새 소유자가 해당 종속성의 종속 오브젝트에 대해 다음과 같은 특정 특권을 보유해야 합니다.

  - 종속 오브젝트가 시퀀스일 경우, 새 소유자에게 해당 시퀀스에 대한 USAGE 특권이 있어야 합니다.
  - 종속 오브젝트가 함수, 메소드 또는 프로시저일 경우 새 소유자에게 함수, 메소드 또는 프로시저에 대한 EXECUTE 특권이 있어야 합니다.
  - 종속 오브젝트가 패키지일 경우, 새 소유자에게 해당 패키지에 대한 EXECUTE 특권이 있어야 합니다.
  - 종속 오브젝트가 XSR 오브젝트일 경우, 새 소유자에게 해당 XSR 오브젝트에 대한 USAGE 특권이 있어야 합니다.

  종속성의 모든 기타 종속 오브젝트의 경우, 적절한 시스템 카탈로그 뷰에서 TABAUTH 컬럼을 사용하여 새 소유자가 보유해야 하는 특권을 판별하십시오.
- 오브젝트의 소유권을 해당 소유자에게 전송하려고 시도할 경우, 경고가 발생합니다 (SQLSTATE 01676).
- 감사 규정, 버퍼 풀, 역할, 보안 레이블, 보안 레이블 구성요소, 보안 규정, 서버, 변환 함수, 트러스트된 컨텍스트, 사용자 맵핑 및 랩퍼 데이터베이스 오브젝트의 소유권은 해당 오브젝트에 소유자가 없으므로 전송할 수 없습니다.
  SYSCAT.AUDITPOLICIES, SYSCAT.BUFFERPOOLS, SYSCAT.CONTEXTS, SYSCAT.ROLES, SYSCAT.SECURITYLABELS, SYSCAT.SECURITYLABELCOMPONENTS, SYSCAT.SECURITYPOLICIES, SYSCAT.SERVERS, SYSCAT.TRANSFORMS, SYSCAT.USEROPTIONS 및 SYSCAT.WRAPPERS 카탈로그 뷰에는 OWNER 컬럼이 없음에 유의하십시오.
- 소유권이 전송된 오브젝트의 스키마 이름은 자동으로 변경되지 않습니다.
- *호환성*: 기타 SQL문과의 일관성을 위해 다음이 수행됩니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.
  - ALIAS 대신 SYNONYM을 지정할 수 있습니다.

예:

예 1: 테이블 T1의 소유권을 PAUL에게 전송합니다.

```
TRANSFER OWNERSHIP OF TABLE WALID.T1
  TO USER PAUL PRESERVE PRIVILEGES
```

SYSCAT.TABLES 카탈로그 뷰에서 WALID.T1 테이블에 대한 OWNER 컬럼의 값이 'PAUL'로 대체됩니다. Paul에게는 테이블 WALID.T1에 대한 다음과 같은 특권이 내재적으로 부여됩니다(테이블의 이전 소유자가 특권을 손실하지 않았다고 가정). CONTROL 및 ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE, REFERENCE(WITH GRANT OPTION).

예 2: JOHN이 테이블 T1 및 T2를 작성하고, MIKE가 테이블 JOHN.T1 및 JOHN.T2에 대한 SELECT 특권을 보유한다고 가정합니다. MIKE는 테이블 JOHN.T1 및 JOHN.T2에 종속되는 뷰 V1을 작성합니다. 뷰 V1의 소유권을 DBADM 권한을 가진 HENRY에게 전송합니다.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER HENRY PRESERVE PRIVILEGES
```

SYSCAT.VIEWS 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'HENRY'로 대체됩니다. 다음과 같은 값을 갖는 새 행이 SYSCAT.TABAUTH에 추가됩니다. GRANTOR = 'SYSIBM', GRANTEE = 'HENRY' 및 TABNAME = 'V1'.

예 3: DBADM 권한을 보유하는 HENRY가 테이블 T1에 종속된 트리거 TR1을 작성한다고 가정합니다. 트리거 TR1의 소유권을 DBADM 권한이 없는 WALID로 전송합니다.

```
TRANSFER OWNERSHIP OF TRIGGER TR1
  TO USER WALID PRESERVE PRIVILEGES
```

Walid에게 DBADM 권한이 없는 경우에도 트리거의 소유권이 성공적으로 전송됩니다.

예 4: JOHN이 테이블 T1 및 T2를 작성하고 MIKE가 테이블 JOHN.T1에 대한 SELECT 특권 및 테이블 JOHN.T2에 대한 CONTROL 특권을 보유한다고 가정합니다. PAUL은 테이블 JOHN.T1 및 JOHN.T2에 대한 SELECT 특권을 보유합니다. MIKE는 테이블 JOHN.T1 및 JOHN.T2에 종속되는 뷰 V1을 작성합니다. 뷰에는 SYSCAT.TABAUTH에 SELECT 특권에 대한 하나의 항목과 테이블 JOHN.T1 및 JOHN.T2에 대하여 SYSCAT.TABDEP에 두 개의 SELECT 종속성이 있습니다. 뷰 V1의 소유권을 일반 사용자인 PAUL에게 전송합니다.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER PAUL PRESERVE PRIVILEGES
```

Paul에게 테이블 JOHN.T2에 대한 CONTROL 특권이 없는 경우에도 뷰의 소유권이 성공적으로 전송됩니다. Paul에게는 뷰의 존재를 유지보수하기 위한 테이블 JOHN.T1 및 JOHN.T2에 대한 SELECT 특권만을 테이블 특권만이 필요합니다. (뷰 작성시 Paul이 두 테이블 모두에 대해 CONTROL 특권이 없고 따라서 뷰에 대한 CONTROL이 부여되지 않으므로 뷰는 SELECT 특권만을 갖습니다.) SYSCAT.VIEWS 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'PAUL'로 대체됩니다. SYSCAT.TABDEP 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'PAUL'로 대체됩니다. 다음과 같은 값을 갖는 새 행이 SYSCAT.TABAUTH에 추가됩니다. GRANTOR = 'SYSIBM', GRANTEE = 'PAUL' 및 TABNAME = 'V1'.

예 5: JOHN이 테이블 T1을 작성하고, PUBLIC이 JOHN.T1에 대한 SELECT 특권을 보유한다고 가정합니다. PAUL은 JOHN.T1에 대한 SELECT 특권을 명시적으로 보

유하며, 테이블 JOHN.T1에 종속된 뷰 V1을 작성합니다. 뷰 V1의 소유권을 DBADM이 아니지만 특수 그룹 PUBLIC을 통해 뷰 소유권을 확보하는 데 필요한 특권을 보유한 MIKE에게 전송합니다.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER MIKE PRESERVE PRIVILEGES
```

Mike가 PUBLIC을 통해 테이블 JOHN.T1에 대한 SELECT 특권을 보유하므로, 뷰의 소유권이 성공적으로 전송됩니다. SYSCAT.VIEWS 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'MIKE'로 대체됩니다. SYSCAT.TABDEP 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'MIKE'로 대체됩니다. 다음과 같은 값을 갖는 새 행이 SYSCAT.TABAUTH에 추가됩니다. GRANTOR = 'SYSIBM', GRANTEE = 'MIKE' 및 TABNAME = 'V1'.

예 6: 예 5와 유사합니다. JOHN이 테이블 T1을 작성하고, 역할 R1이 JOHN.T1에 대한 SELECT 특권을 보유한다고 가정합니다. PAUL은 JOHN.T1에 대한 SELECT 특권을 명시적으로 보유하며, 테이블 JOHN.T1에 종속된 뷰 V1을 작성합니다. 뷰 V1의 소유권을 DBADM이 아니지만 역할 R1의 멤버쉽을 통해 뷰 소유권을 확보하는 데 필요한 특권을 보유한 MIKE에게 전송합니다.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER MIKE PRESERVE PRIVILEGES
```

Mike가 역할 R1의 멤버쉽을 통해 테이블 JOHN.T1에 대한 SELECT 특권을 보유하므로 뷰 소유권이 성공적으로 전송됩니다. SYSCAT.VIEWS 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'MIKE'로 대체됩니다. SYSCAT.TABDEP 카탈로그 뷰에서 뷰 V1에 대한 OWNER 컬럼의 값이 'MIKE'로 대체됩니다. 다음과 같은 값을 갖는 새 행이 SYSCAT.TABAUTH에 추가됩니다. GRANTOR = 'SYSIBM', GRANTEE = 'MIKE' 및 TABNAME = 'V1'.

TRUNCATE

TRUNCATE문은 테이블에서 모든 행을 삭제합니다.

호출

이 명령문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행할 수 있습니다. 이는 DYNAMICRULES 실행 동작이 패키지에 영향을 줄 때(SQLSTATE 42509)에만 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID가 보유하는 특권은 테이블과 테이블 계층의 모든 서브테이블에 대해 다음 중 최소한 하나를 포함해야 합니다.

- 절단할 테이블에 대한 DELETE 특권
- 절단할 테이블에 대한 CONTROL 특권
- DATAACCESS 권한

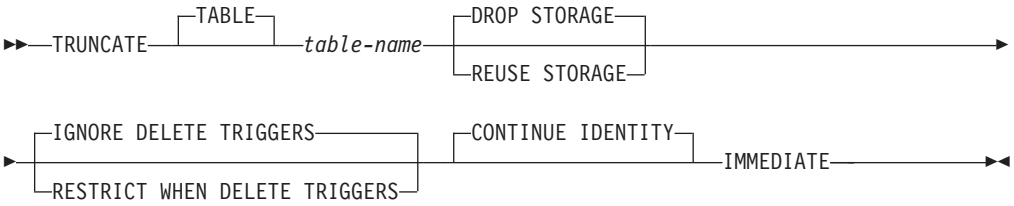
테이블에 정의된 DELETE 트리거를 무시하려면 명령문의 권한 부여 ID가 보유하는 특권에 테이블과 테이블 계층의 모든 서브테이블에 대해 다음 중 최소한 하나가 포함되어야 합니다.

- 테이블에 대한 ALTER 특권
- 테이블에 대한 CONTROL 특권
- DBADM 권한

보안 규정으로 보호되는 테이블을 절단하려면, 명령문의 권한 부여 ID가 보유하는 특권에 적어도 다음 중 하나가 포함되어야 합니다.

- 테이블에 대한 CONTROL 특권
- DBADM 권한

구문



설명

table-name

절단할 테이블을 식별합니다. 이름은 현재 서버에 있는 테이블을 식별해야 하지만 (SQLSTATE 42704), 카탈로그 테이블(SQLSTATE 42832), 별칭(SQLSTATE

42809), 뷰, 서브테이블, 스테이징 테이블, 시스템 유지보수 구체화된 쿼리 테이블 또는 범위 클러스터 테이블(SQLSTATE 42807)은 될 수 없습니다.

*table-name*이 테이블 계층의 루트 테이블인 경우 테이블 계층의 모든 테이블이 절단됩니다.

DROP STORAGE 또는 REUSE STORAGE

테이블에 대해 할당되는 기존 스토리지의 삭제(drop) 또는 재사용 여부를 지정합니다. 디폴트는 DROP STORAGE입니다.

DROP STORAGE

테이블에 할당된 모든 스토리지가 해제되어 사용 가능하게 됩니다. 이 옵션을 지정한 경우(내재적 또는 명시적으로) 온라인 백업이 차단됩니다.

REUSE STORAGE

테이블에 할당된 모든 스토리지는 테이블에 대해 계속 할당되지만 스토리지는 비어 있는 것으로 간주됩니다. 이 옵션은 DMS 테이블 스페이스의 테이블에만 적용 가능하며 다른 테이블에 대해서는 무시됩니다.

IGNORE DELETE TRIGGERS 또는 RESTRICT WHEN DELETE TRIGGERS

삭제 트리거가 테이블에 정의된 경우 수행할 사항을 지정합니다. 디폴트는 IGNORE DELETE TRIGGERS입니다.

IGNORE DELETE TRIGGERS

테이블에 대해 정의된 삭제 트리거가 절단 조작에 의해 활성화되지 않습니다.

RESTRICT WHEN DELETE TRIGGERS

삭제 트리거나 테이블에 정의된 경우 오류가 리턴됩니다(SQLSTATE 428GJ).

CONTINUE IDENTITY

테이블에 대해 ID 컬럼이 존재하는 경우 생성되는 다음 ID 컬럼 값은 TRUNCATE 문이 실행되지 않은 경우 생성되는 다음 값으로 계속됩니다.

IMMEDIATE

절단 조작이 즉시 처리되어 실행 취소할 수 없음을 지정합니다. 명령문은 트랜잭션에서 첫 번째 명령문이어야 합니다(SQLSTATE 25001).

절단된 테이블은 동일한 작업 단위에서 즉시 사용 가능합니다. TRUNCATE문 다음에 ROLLBACK문을 실행할 수 있지만, 전달 조작은 실행 취소되지 않아서 테이블은 절단된 상태로 유지됩니다. 예를 들어, TRUNCATE IMMEDIATE문 다음에 테이블에 대해 또 다른 데이터 변경 조작이 수행되고 ROLLBACK문이 실행되는 경우, 절단 조작은 실행 취소되지 않지만 다른 모든 데이터 변경 조작은 실행 취소됩니다.

규칙

- *참조 무결성:* 테이블과 테이블 계층의 모든 테이블은 강제로 실행되는 참조 제한조건에서 상위 테이블이 될 수 없습니다(SQLSTATE 428GJ). 자체 참조 RI 제한조건은 허용됩니다.
- *파티션된 테이블:* 테이블은 데이터 파티션에 연결하도록 변경되기 때문에 무결성 설정 보류 상태로 설정할 수 없습니다(SQLSTATE 55019).TRUNCATE문을 실행하기 전에 무결성에 대해 테이블을 점검해야 합니다.
- *독점 액세스:* 다른 세션은 테이블에서 커서를 열 수 없고 테이블에 잠금을 보유할 수 없습니다(SQLSTATE 25001).
- ***WITH HOLD 커서:*** 현재 세션은 테이블에 대해 WITH HOLD 커서를 열 수 없습니다(SQLSTATE 25001).

주

- *테이블 통계:* 테이블에 대한 통계는 TRUNCATE문으로 변경되지 않습니다.
- *삭제된 행 수:* SQLCA에서 SQLERRD(3)는 절단 조작에 대해 -1로 설정됩니다. 테이블에서 삭제된 행 수는 리턴되지 않습니다.

예 :

예 1: 기존 트리거에 관계없이 사용되지 않는 인벤토리 테이블을 비우고 해당되는 할당 스페이스를 리턴하십시오.

```
TRUNCATE TABLE INVENTORY
  IGNORE DELETE TRIGGERS
  DROP STORAGE
  IMMEDIATE
```

예 2: 기존의 삭제 트리거에 관계없이 사용되지 않는 인벤토리 테이블을 비우지만 나중에 재사용하기 위해 할당 스페이스를 보존하십시오.

```
TRUNCATE TABLE INVENTORY
   REUSE STORAGE
   IGNORE DELETE TRIGGERS
   IMMEDIATE
```

UPDATE

UPDATE문은 테이블, 별칭 또는 뷰의 행에 있는 지정된 컬럼의 값을 갱신하거나 지정된 fullselect의 뷰, 별칭 또는 하위 테이블을 갱신합니다. 뷰의 갱신 조작에 INSTEAD OF 트리거를 정의하지 않은 경우 뷰의 행을 갱신하면 뷰의 기본 테이블의 행이 갱신됩니다. 이와 같은 트리거가 정의되었을 경우 트리거가 대신 실행됩니다. 별칭을 사용하여 행을 갱신하면 별칭이 참조하는 데이터 소스 오브젝트에 있는 행이 갱신됩니다.

이 명령문의 양식은 다음과 같습니다.

- *검색* UPDATE 양식은 하나 이상의 행을 갱신하는 데 사용됩니다(검색 조건에 따라 선택적으로 결정).
- *위치 지정된* UPDATE 양식은 정확히 한 행을 갱신하는 데 사용됩니다(현재의 커서 위치에 따라 결정).

호출

UPDATE문은 응용프로그램에 임베드되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행문입니다.

권한 부여

명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.

- 목표 테이블, 뷰 또는 별칭에 대한 UPDATE 특권
- 갱신될 각 컬럼의 UPDATE 특권
- 목표 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
- DATAACCESS 권한

*row-fullselect*가 지정에 포함된 경우, 명령문의 권한 부여 ID가 보유한 특권에는 참조된 테이블, 뷰 또는 별칭 각각에 대해 적어도 다음 중 하나가 있어야 합니다.

- SELECT 특권
- CONTROL 특권
- DATAACCESS 권한

서브쿼리에서 참조하는 각 테이블, 뷰 또는 별칭의 경우, 명령문의 권한 부여 ID가 보유한 특권에는 다음 중 적어도 하나는 포함되어야 합니다.

- SELECT 특권
- CONTROL 특권
- DATAACCESS 권한

명령문 처리에 사용되는 패키지가 SQL92 규칙으로 프리컴파일되어 있고, SQL92E나 MIA 값이 포함된 LANGLEVEL 옵션의 UPDATE문의 검색 양식에 *assignment-clause*

의 오른쪽이나 *search-condition*에 있는 테이블의 컬럼, 뷰 또는 별칭에 대한 참조가 포함되어 있으면 명령문의 권한 부여 ID가 보유한 특권에 적어도 다음 중 하나가 있어야 합니다.

- SELECT 특권
- CONTROL 특권
- DATAACCESS 권한

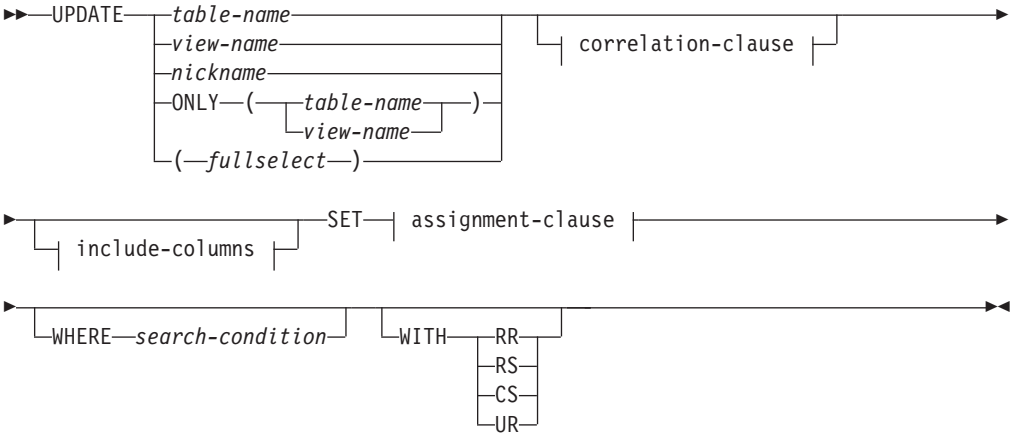
지정된 테이블이나 뷰 앞에 ONLY 키워드가 오는 경우 명령문의 권한 부여 ID가 가지고 있는 특권에는 지정된 테이블이나 뷰의 모든 서브테이블 또는 서브뷰에 대한 SELECT 특권도 포함되어야 합니다.

정적 UPDATE문에 대해서는 GROUP 특권이 점검되지 않습니다.

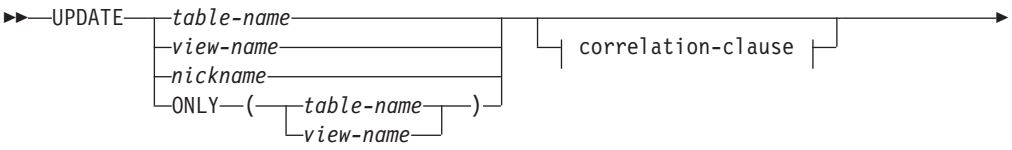
갱신 조작의 목표가 별칭일 경우에는 데이터 소스에서 명령문을 실행할 때까지 데이터 소스의 오브젝트에 대한 특권이 무시됩니다. 이 때 데이터 소스에 연결하는 데 사용되는 권한 부여 ID에는 데이터 소스에서 해당 오브젝트에 조작을 수행하는 데 필요한 특권이 있어야 합니다. 명령문의 권한 부여 ID는 데이터 소스에서 서로 다른 권한 부여 ID로 맵핑될 수 있습니다.

구문

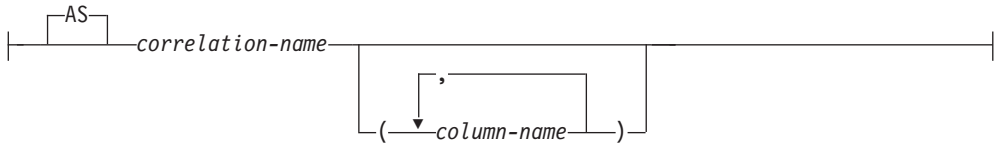
searched-update:



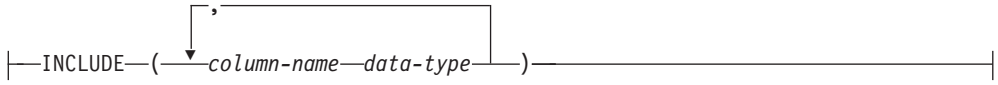
positioned-update:



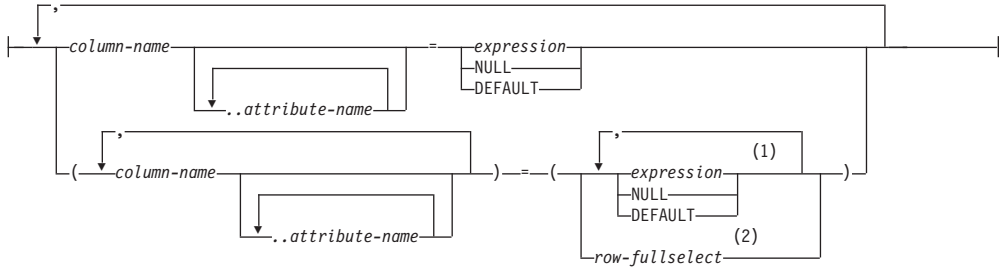
correlation-clause:



include-columns:



assignment-clause:



주:

1 표현식 NULL 및 DEFAULT의 수는 컬럼 이름 수와 일치해야 합니다.

2 선택 목록의 컬럼 수는 컬럼 이름 수와 일치해야 합니다.

설명

table-name, *view-name*, *nickname* 또는 *(fullselect)*

갱신 조작의 오브젝트를 식별합니다. 이름은 카탈로그에 설명되어 있는 테이블, 뷰 또는 별칭을 식별해야 하지만 갱신 조작에 대해 INSTEAD OF 트리거가 정의되어 있지 않으면 카탈로그 테이블, 카탈로그 테이블 뷰(갱신 가능한 SYSSTAT 뷰 중 하나이면 제외), 시스템 유지보수 구체화된 쿼리 테이블 또는 읽기 전용 뷰를 식별해서는 안됩니다.

*table-name*이 유형이 지정된 테이블인 경우, 테이블 행 또는 적절한 서브테이블이 명령문에 의해 갱신될 수 있습니다. 지정된 테이블의 컬럼들만이 WHERE절에서 설정되거나 참조될 수 있습니다. 위치 지정된 UPDATE의 경우, 연관된 커서가 ONLY를 사용하지 않고도 FROM절에서 동일한 테이블, 뷰 또는 별칭도 지정해야 합니다.

갱신 조작의 오브젝트가 fullselect인 경우 CREATE VIEW문의 설명에 있는 『갱신 가능한 뷰』 주 항목에 정의되었듯이 fullselect는 갱신 가능해야 합니다.

ONLY (*table-name***)**
유형이 지정된 뷰에 적용 가능한 ONLY 키워드는 명령문이, 지정된 테이블 데이터에만 적용되고 해당 서브테이블의 행은 갱신할 수 없도록 지정합니다. 위치 지정된 UPDATE인 경우 연관된 커서가 ONLY를 사용하여 FROM절에 있는 테이블도 지정했을 것입니다. *table-name*이 유형이 지정된 테이블이 아니면 ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

ONLY (*view-name***)**
유형이 지정된 뷰에 적용 가능한 ONLY 키워드는 명령문이, 지정된 뷰 데이터에만 적용되고 해당 서브뷰의 행은 갱신할 수 없도록 지정합니다. 위치 지정된 UPDATE인 경우 연관된 커서가 ONLY를 사용하여 FROM절에 있는 뷰도 지정했을 것입니다. *view-name*이 유형이 지정된 뷰가 아니면 ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

correlation-clause
테이블, 뷰, 별칭 또는 fullselect를 지시하기 위해 *search-condition* 또는 *assignment-clause* 내에서 사용될 수 있습니다. *correlation-clause*의 설명은 『Subselect』의 『table-reference』를 참조하십시오.

include-columns
table-name 또는 *view-name*의 컬럼과 함께 UPDATE문의 중간 결과 테이블에 포함되는 일련의 컬럼을 지정합니다(컬럼이 fullselect의 FROM절에서 중첩되는 경우). *include-columns*는 *table-name* 또는 *view-name*으로 사용되는 컬럼의 목록 끝에 추가됩니다.

INCLUDE
컬럼 목록이 UPDATE문의 중간 결과 테이블에 포함되도록 지정합니다.

column-name
UPDATE문에 있는 중간 결과 테이블의 컬럼을 지정합니다. 이름은 *table-name* 또는 *view-name*에서 컬럼 또는 다른 Include 컬럼의 이름과 동일해서는 안됩니다(SQLSTATE 42711).

data-type
Include 컬럼의 데이터 유형을 지정합니다. 데이터 유형은 CREATE TABLE 문으로 지원된 하나여야 합니다.

SET
컬럼 이름에 대한 값 지정을 사용합니다.

assignment-clause

column-name
갱신될 컬럼을 식별합니다. *column-name*은 지정된 테이블, 뷰 또는 별칭 또는

INCLUDE 컬럼 식별의 갱신 가능 컬럼을 식별해야 합니다. 유형이 지정된 테이블의 오브젝트 ID 컬럼은 갱신할 수 없습니다(SQLSTATE 428DZ). 컬럼은 *..attribute-name* 뒤에 오는 경우를 제외하고는 두 번 이상 지정해서는 안됩니다(SQLSTATE 42701).

INCLUDE 컬럼을 지정하는 경우 컬럼 이름은 규정될 수 없습니다.

Positioned UPDATE의 경우 다음과 같습니다.

- *update-clause*가 커서의 *select-statement*에 지정되어 있을 경우 *assignment-clause*의 각 컬럼 이름 또한 *update-clause*에 나타나야 합니다.
- *update-clause*가 커서의 *select-statement*에 지정되지 않고 응용프로그램 프리컴파일 시 LANGLEVEL MIA 또는 SQL92E가 지정된 경우 갱신 가능 컬럼의 이름을 지정할 수 있습니다.
- *update-clause*이 커서의 *select-statement*에 지정되지 않고 응용프로그램 프리컴파일 시 LANGLEVEL SAA1이 명시적으로 또는 디폴트값으로 지정된 경우 컬럼을 갱신할 수 없습니다.

..attribute-name

설정된 구조화된 유형의 속성을 지정합니다(*attribute assignment*라고도 함). 지정된 *column-name*을 사용자 정의 구조화된 유형으로 정의해야 합니다(SQLSTATE 428DP). attribute-name은 *column-name*의 구조화된 유형 속성이어야 합니다(SQLSTATE 42703). *..attribute-name*절을 포함하지 않는 지정을 *기본 지정*이라고도 합니다.

expression

컬럼의 새 값을 나타냅니다. 이 표현식은 『Expressions』에 설명되어 있는 모든 유형의 표현식을 말합니다. 표현식에는 스칼라 fullselect 내에서 발생할 때를 제외하고 집계 함수를 포함시킬 수 없습니다(SQLSTATE 42903).

*expression*에는 UPDATE문의 목표 테이블의 컬럼 참조가 포함될 수 있습니다. 갱신되는 각 행에 대해 표현식에 있는 컬럼 값은 갱신되기 전의 행에 있는 컬럼 값입니다.

표현식에는 INCLUDE 컬럼에 대한 참조가 포함될 수 없습니다.

NULL

널(NULL) 값을 지정하고 널(NULL) 입력 가능 컬럼에 대해서만 지정될 수 있습니다(SQLSTATE 23502). 널(NULL)은 특별히 속성의 데이터 유형으로 캐스트되지 않을 경우 속성 지정의 값이 될 수 없습니다(SQLSTATE 429B9).

DEFAULT

해당 컬럼이 테이블에 정의된 방식에 기초하여 디폴트값을 사용하도록 지정됩니다. 삽입되는 값은 컬럼 정의 방식에 따릅니다.

- 표현식을 기초로 해당 컬럼이 생성된 컬럼으로 정의된 경우, 시스템이 표현식을 근거로 컬럼 값을 생성합니다.

- IDENTITY절을 사용하여 컬럼을 정의했다면 데이터베이스 관리 프로그램이 값을 생성합니다.
- WITH DEFAULT절을 사용하여 컬럼이 정의된 경우, 값은 컬럼에 정의된 디폴트값으로 설정됩니다(『ALTER TABLE』의 *default-clause* 참조).
- NOT NULL절을 사용하여 컬럼을 정의했고 GENERATED절이 사용되지 않았거나 또는 WITH DEFAULT절이 사용되지 않았거나 DEFAULT NULL이 사용되었다면 해당 컬럼에 대해 DEFAULT 키워드를 지정할 수 없습니다(SQLSTATE 23502).
- ROW CHANGE TIMESTAMP절을 사용하여 컬럼을 정의하면 데이터베이스 관리 프로그램이 값을 생성합니다.

GENERATED ALWAYS절로 정의된 생성된 컬럼을 설정할 수 있는 유일한 값은 DEFAULT입니다(SQLSTATE 428C9).

DEFAULT 키워드는 속성 지정의 값으로 사용될 수 없습니다(SQLSTATE 429B9).

DEFAULT 키워드는 데이터 소스가 DEFAULT 구문을 지원하지 않는 별칭에 대한 갱신에 값을 지정할 때는 사용할 수 없습니다.

row-fullselect

할당을 위해 지정된 *column-name*의 수와 일치하는 컬럼 수로 단일 행을 리턴하는 fullselect입니다. 각각의 해당 *column-name*에 값이 할당됩니다. *row-fullselect* 결과에 행이 없는 경우 널(NULL) 값이 할당됩니다.

*row-fullselect*에는 UPDATE문의 목표 테이블 컬럼 참조가 포함됩니다. 갱신되는 각 행에 대해 표현식에 있는 컬럼 값은 갱신되기 전의 행에 있는 컬럼 값입니다. 결과에 한 행 이상이 있는 경우 오류가 발생합니다(SQLSTATE 21000).

WHERE

갱신되는 행을 표시하는 조건을 사용합니다. 절을 생략하거나 검색 조건을 제공하고 또는 커서를 이름 지정할 수 있습니다. 절이 생략되는 경우 테이블, 뷰 또는 별칭의 모든 행이 갱신됩니다.

search-condition

서브쿼리가 아닌 검색 조건의 각 *column-name*은 테이블, 뷰 또는 별칭의 컬럼을 이름 지정해야 합니다. 검색 조건에 UPDATE 및 서브쿼리 모두의 기본 오브젝트가 같은 테이블인 서브쿼리가 포함될 때 서브쿼리는 행이 갱신되기 전에 완전히 평가됩니다.

search-condition은 테이블, 뷰 또는 별칭의 각 행에 적용되고 갱신된 행은 search-condition을 만족시킵니다.

검색 조건에 서버쿼리가 포함되는 경우 서브쿼리는 검색 조건이 한 행에 적용될 때마다 실행되고 그 결과는 검색 조건 적용 시 사용된다고 생각할 수 있습

니다. 실제로 상관 참조가 없는 서브쿼리는 한 번만 실행되는 반면, 상관 참조가 있는 서브쿼리는 각 행에 대해 한 번씩 실행되어야 합니다.

CURRENT OF *cursor-name*

갱신 조작에 사용될 커서를 식별합니다. *cursor-name*은 선언된 커서를 『DECLARE CURSOR』에 설명된 대로 식별해야 합니다. DECLARE CURSOR문은 프로그램에서 UPDATE문 앞에 와야 합니다.

지정된 테이블, 뷰 또는 별칭은 커서에 대한 SELECT문의 FROM절에서도 이름이 지정되어야 하며, 커서의 결과 테이블은 읽기 전용이어서는 안됩니다. (읽기 전용 결과 테이블에 대한 설명은 『DECLARE CURSOR』에서 참조하십시오.)

UPDATE문이 실행될 때 커서는 행에 위치해야 그 행이 갱신됩니다.

이 UPDATE 형태는 커서가 다음을 참조하면 사용할 수 없습니다(SQLSTATE 42828).

- INSTEAD OF UPDATE 트리거가 정의된 뷰
- 뷰를 정의하는 fullselect의 선택된 목록에 OLAP 함수가 포함된 뷰
- WITH ROW MOVEMENT절을 사용하여 직접 또는 간접적으로 정의된 뷰

WITH

UPDATE문이 실행되는 위치에 분리 레벨을 지정합니다.

RR

반복 읽기

RS

읽기 안정성

CS

커서 안정성

UR

언커미트 읽기

명령문의 디폴트 분리 레벨은 명령문이 바운드되는 패키지의 분리 레벨입니다. WITH 절은 항상 명령문의 디폴트 분리 레벨을 사용하는 별칭에 영향을 주지 않습니다.

규칙

- *트리거:* UPDATE문이 트리거를 실행시킵니다. 트리거로 인해 다른 명령문이 실행될 수도 있고, 갱신 값에 기반한 오류가 발생할 수도 있습니다. 뷰를 갱신하는 조작으로 인해 INSTEAD OF 트리거가 발생하면 트리거를 발생시킨 뷰나 이 뷰의 하위 테이블에 대해서가 아니라 트리거에서 수행된 갱신에 대해 유효성, 참조 무결성, 제한조건이 점검됩니다.
- *지정*: 갱신 값은 특정 지정 규칙에 따라 컬럼에 지정됩니다.

- *유효성*: 갱신된 행은 갱신된 컬럼의 고유 인덱스에 의해 테이블(또는 뷰의 기본 테이블)에 강요되는 모든 제한조건을 따라야 합니다.

  WITH CHECK OPTION을 사용하여 정의되지 않은 뷰가 사용된 경우 뷰의 정의를 따르지 않도록 행이 변경될 수 있습니다. 그러한 행은 뷰의 기본 테이블에서 갱신되며, 뷰에는 더 이상 나타나지 않습니다.

  WITH CHECK OPTION을 사용하여 정의된 뷰를 사용하는 경우 갱신된 행은 그 뷰의 정의를 따라야 합니다. 이러한 상황에 적용되는 규칙에 대한 설명은 『CREATE VIEW』를 참조하십시오.
- *점검 제한조건*: 갱신 값은 테이블에 정의된 점검 제한조건의 점검 조건을 만족시켜야 합니다.

  정의된 점검 제한조건을 지닌 테이블로 UPDATE함으로써 갱신된 각 행에 대해 한 번 평가되고, 갱신된 각 컬럼에 대한 점검 제한조건을 가집니다. UPDATE문 처리시 갱신된 컬럼을 참조하는 점검 제한조건만이 점검됩니다.
- *참조 무결성*: 상위 고유 키의 값은 갱신 규칙이 제한되고 하나 이상의 종속 행이 있는 경우 변경될 수 없습니다. 그러나 갱신 규칙이 NO ACTION인 경우 Update문이 완료될 때까지 모든 하위 키가 상위 키를 갖게 되면, 상위 고유 키를 갱신할 수 있습니다. 외부 키의 널(NULL)이 아닌 갱신 값은 관계의 상위 테이블에서 기본 키 값과 같아야 합니다.
- ***XML 값:*** XML 컬럼 값이 갱신될 경우 새 값은 바르게 구성된 XML 문서여야 합니다(SQLSTATE 2200M).
- ***보안 규정:*** 식별된 테이블이나 식별된 뷰의 기본 테이블이 보안 규정으로 보호된 경우, 세션 권한 부여 ID에는 허용되는 레이블 기반 액세스 제어(LBAC) 증명서가 있어야 합니다.
  - 갱신되는 모든 보호 컬럼에 대한 쓰기 액세스(SQLSTATE 42512)
  - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션으로 작성되었던 보안 규정에 대해 DB2SECURITYLABEL 컬럼에서 제공된 명시적인 값에 대한 쓰기 액세스(SQLSTATE 23523)
  - 갱신되는 모든 행에 대한 읽기 및 쓰기 액세스(SQLSTATE 42519)

  내재된 값이 DB2SECURITYLABEL 컬럼에 대해 사용되면(SQLSTATE 23523) 이 세션 권한 부여 ID는 보안 규정에 대한 쓰기 액세스에 대해 보안 레이블을 부여해야 하며, 다음과 같은 상황에서 발생할 수 있습니다.
  - DB2SECURITYLABEL 컬럼은 갱신될 컬럼 목록에 포함되지 않습니다(또한 세션 권한 부여 ID의 쓰기 액세스에 대한 보안 레벨로 내재적으로 갱신됨).
  - DB2SECURITYLABEL 컬럼 값이 명시적으로 제공되지만 세션 권한 부여 ID에 해당 값에 대한 쓰기 액세스가 없으며, 보안 규정이 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션으로 작성된 경우

주

- 갱신 값이 의무 규정을 위반하거나 UPDATE문 실행 중에 오류가 발생하면 행이 갱신되지 않습니다. 복수의 행이 갱신되는 순서는 정의되어 있지 않습니다.
- WITH ROW MOVEMENT절을 사용하여 정의된 뷰 갱신 시 뷰의 기본 테이블에 대한 삭제 조작 및 삽입 조작이 발생할 수 있습니다. 자세한 내용은 CREATE VIEW 문의 설명을 참조하십시오.
- UPDATE문이 실행을 마칠 때 SQLCA의 SQLERRD(3) 값이 갱신 조작 시 규정된 행의 수입니다. SQL 프로시저 명령문에서는 GET DIAGNOSTICS문의 ROW_COUNT 변수를 사용하여 이 값을 검색할 수 있습니다. SQLERRD(5) 필드에는 모든 활성 트리거가 삽입, 삭제 또는 갱신한 행 수가 들어 있습니다.
- 적절하게 잠겨져 있지 않은 경우 UPDATE문을 성공적으로 실행하면 하나 이상의 배타적 잠금이 생깁니다. 잠금이 해제될 때까지 갱신된 행은 갱신을 수행한 응용프로그램 프로세스에 의해서만 액세스됩니다(언커미트 읽기 분리 레벨을 사용하는 응용프로그램인 경우는 제외). 잠금에 대한 자세한 내용은 COMMIT, ROLLBACK 및 LOCK TABLE문의 설명을 참조하십시오.
- 유형이 지정된 테이블에 대한 컬럼 분배 통계를 갱신할 때 처음 컬럼을 도입한 서브테이블을 지정해야 합니다.
- 같은 구조화된 유형에 대해 여러 속성 지정이 SET절에 지정된 순서대로 그리고 괄호로 묶인 SET절내에서 왼쪽으로부터 오른쪽순으로 발생합니다.
- 속성 지정은 사용자 정의 구조화된 유형의 속성에 대한 변환 메소드를 호출합니다. 예를 들어, 지정 `st..a1=x`는 지정 `st = st..a1(x)`에서 변환 메소드를 사용할 때와 같은 효과를 지닙니다.
- 주어진 컬럼이 단 하나의 전통적인 지정에 있는 목표 컬럼일 수 있는 반면, 컬럼은 여러 속성 지정의 목표 컬럼일 수 있습니다(단, 전통적인 지정의 목표 컬럼이 아닌 경우에 한함).
- 구별 유형으로 정의된 식별 컬럼이 갱신된 경우 전체 계산이 소스 유형으로 수행되고 그 값이 실제로 컬럼에 지정되기 전에 결과가 구별 유형으로 캐스트됩니다. (이전 값은 계산에 앞서 소스 유형으로 캐스팅되지 않습니다.)
- DB2가 식별 컬럼에 대한 SET문의 값을 생성하도록 하려면 DEFAULT 키워드를 사용하십시오.

  ```
  SET NEW.EMPNO = DEFAULT
  ```

  이 예에서 NEW.EMPNO는 식별 컬럼으로 정의되고 이 컬럼을 갱신하는 데 사용되는 값은 DB2가 생성합니다.
- 식별 컬럼에 대해 생성된 값 시퀀스를 사용하는 방법 또는 식별 컬럼의 최대값 초과에 대한 자세한 내용은 『INSERT』를 참조하십시오.
- 파티션된 테이블에서 UPDATE WHERE CURRENT OF *cursor-name* 조작은 한 데이터 파티션에서 다른 데이터 파티션으로 행을 이동시킬 수 있습니다. 이러한 이동

이 발생하면 커서가 더 이상 행에 위치하지 않으며, 해당 행에 대한 UPDATE WHERE CURRENT OF *cursor-name* 수정이 더 이상 가능하지 않습니다. 그러나 커서의 다음 행은 페치(fetch)될 수 있습니다.

- ROW CHANGE TIMESTAMP절을 사용하여 정의되는 컬럼의 경우, 값은 항상 행의 갱신에서 변경됩니다. 컬럼이 SET 목록에서 명시적으로 지정되지 않으면, 데이터베이스 관리 프로그램이 해당 행의 값을 생성합니다. 값은 데이터베이스 파티션 내의 각 테이블 파티션에 대해 고유하며 행 갱신에 해당되는 대략의 시간소인으로 설정됩니다.

예:

- *예 1:* EMPLOYEE 테이블에서 직원 번호(EMPNO) '000290'의 작업(JOB)을 'LABORER'로 변경합니다.

```
UPDATE EMPLOYEE
   SET JOB = 'LABORER'
   WHERE EMPNO = '000290'
```

- *예 2:* 부서(DEPTNO) 'D21'에서 담당하는 PROJECT 테이블의 모든 프로젝트에 대한 프로젝트 인력(PRSTAFF)을 1.5만큼 늘리십시오.

```
UPDATE PROJECT
   SET PRSTAFF = PRSTAFF + 1.5
   WHERE DEPTNO = 'D21'
```

- *예 3:* 부서(WORKDEPT) 'E21'의 관리자를 제외한 모든 직원이 임시로 재지정되었습니다. 이들의 직책(JOB)을 널(NULL)로 변경하고 EMPLOYEE 테이블에서 급여(SALARY, BONUS, COMM) 값을 제로로 변경하여 표시하십시오.

```
UPDATE EMPLOYEE
  SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

이 명령문은 다음과 같이 쓰여질 수도 있습니다.

```
UPDATE EMPLOYEE
  SET (JOB, SALARY, BONUS, COMM) = (NULL, 0, 0, 0)
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

- *예 4:* 직원 번호 000120인 직원의 급여 및 상여금 컬럼을, 각각 갱신된 행 부서의 직원 급여 평균과 상여금 평균으로 설정하십시오.

```
UPDATE (SELECT EMPNO, SALARY, COMM,
  AVG(SALARY) OVER (PARTITION BY WORKDEPT),
  AVG(COMM) OVER (PARTITION BY WORKDEPT)
  FROM EMPLOYEE E) AS E(EMPNO, SALARY, COMM, AVGSAL, AVGCOMM)
SET (SALARY, COMM) = (AVGSAL, AVGCOMM)
WHERE EMPNO = '000120'
```

이전 명령문은 시맨틱적으로 다음 명령문과 동일합니다. 단, 이전 명령문은 EMPLOYEE 테이블에 대한 하나의 액세스만을 필요로 하지만 다음 명령문에서는 EMPLOYEE 테이블을 두 번 지정합니다.

```
UPDATE EMPLOYEE EU
  SET (EU.SALARY, EU.COMM)
  =
   (SELECT AVG(ES.SALARY), AVG(ES.COMM)
  FROM EMPLOYEE ES
  WHERE ES.WORKDEPT = EU.WORKDEPT)
  WHERE EU.EMPNO = '000120'
```

- *예 5:* C 프로그램에서 EMPLOYEE 테이블의 행을 표시한 다음, 요청이 있을 경우 특정 직원의 작업(JOB)을 입력된 새 작업으로 변경하십시오.

```
EXEC SQL  DECLARE C1 CURSOR FOR
               SELECT *
                 FROM EMPLOYEE
                 FOR UPDATE OF JOB;

  EXEC SQL  OPEN C1;

  EXEC SQL  FETCH C1 INTO ...     ;
  if ( strcmp (change, "YES") == 0 )
      EXEC SQL  UPDATE EMPLOYEE
             SET JOB = :newjob
             WHERE CURRENT OF C1;

  EXEC SQL  CLOSE C1;
```

- *예 6:* 다음 예는 컬럼 오브젝트의 속성을 변환합니다.

  다음과 같은 유형과 테이블이 있다고 가정하십시오.

```
CREATE TYPE POINT AS (X INTEGER, Y INTEGER)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TYPE CIRCLE AS (RADIUS INTEGER, CENTER POINT)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TABLE CIRCLES (ID INTEGER, OWNER VARCHAR(50), C CIRCLE
```

  다음 예는 ID가 999인 CIRCLE 컬럼의 RADIUS 속성과 OWNER 컬럼을 변경하여 CIRCLES 테이블을 갱신합니다.

```
UPDATE CIRCLES
  SET OWNER = 'Bruce'
          C..RADIUS = 5
  WHERE ID = 999
```

  다음 예는 999로 식별되는 원의 중심에 대한 X 및 Y 좌표를 바꿉니다.

```
UPDATE CIRCLES
  SET C..CENTER..X = C..CENTER..Y,
          C..CENTER..Y = C..CENTER..X
  WHERE ID = 999
```

  다음 예는 위의 두 명령문을 작성하는 또 다른 방법입니다. 이 예는 상위 두 예의 효과를 조인합니다.

```
UPDATE CIRCLES
  SET (OWNER,C..RADIUS,C..CENTER..X,C..CENTER..Y) =
           ('Bruce',5,C..CENTER..Y,C..CENTER..X)
  WHERE ID = 999
```

- *예 7:* DOCID '001'을 가진 DOCUMENTS 테이블의 XMLDOC 컬럼을 XMLTEXT 테이블에서 선택 및 구문 분석된 문자열로 갱신하십시오.

```
UPDATE DOCUMENTS SET XMLDOC =
       (SELECT XMLPARSE(DOCUMENT C1 STRIP WHITESPACE)
    FROM XMLTEXT WHERE TEXTID = '001')
WHERE DOCID = '001'
```

VALUES

VALUES문은 쿼리 양식입니다. 응용프로그램에 임베디드(embedded)하거나 대화식으로 발행할 수 있습니다.

VALUES INTO

VALUES INTO문은 많아야 한 행으로 구성되는 결과 테이블을 작성한 후, 이 행의 값을 호스트 변수에 지정합니다.

호출

이 명령문은 응용프로그램에만 임베드될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행문입니다.

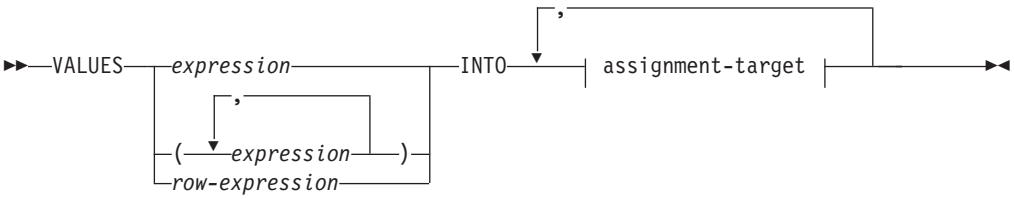
권한 부여

명령문의 권한 부여 ID가 보유한 특권은 각 *expression* 및 *row-expression*을 실행하는 데 필요한 권한을 포함해야 합니다.

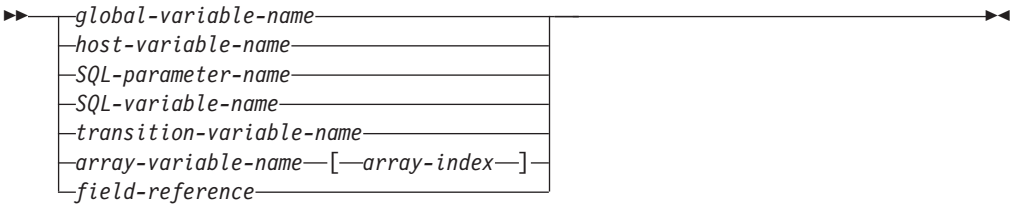
*assignment-target*으로 사용되는 각 전역 변수의 경우, 명령문의 권한 부여 ID에 의해 보유된 특권에는 다음 중 하나를 포함해야 합니다.

- 모듈에 정의되지 않은 전역 변수에 대한 WRITE 특권
- 모듈에 정의된 전역 변수의 모듈에 대한 EXECUTE 특권

구문



assignment-target



설명

VALUES

하나 이상의 컬럼으로 구성되는 단일 행을 도입합니다.

expression

단일 컬럼 결과 테이블의 단일 값을 정의하는 표현식

(expression,...)

결과 테이블의 한 개 이상의 컬럼에서 값을 정의하는 한 개 이상의 표현식

row-expression

값의 새 행을 지정합니다. *row-expression*은 『행 표현식』에서 설명된 유형의 모든 행 표현식입니다. *row-expression*은 컬럼 이름을 포함하지 않아야 합니다.

INTO *assignment-target*

출력 값을 지정할 하나 이상의 목표를 식별합니다.

결과 행의 첫 번째 값은 목록의 첫 번째 목표에 지정되고, 두 번째 값은 두 번째 목표에, 이와 같은 식으로 지정됩니다. 목록에 나타난 순서대로 *assignment-target*에 값이 지정됩니다. 지정 오류가 발생하면, 값이 *assignment-target*에 지정되지 않습니다.

모든 *assignment-target*의 데이터 유형이 행 유형이 아닌 경우, *assignment-targets* 수가 결과 컬럼 값보다 작으면 'W' 값이 SQLCA의 SQLWARN3 필드에 지정됩니다.

*assignment-target*의 데이터 유형이 행 유형이면, 하나의 *assignment-target*가 명확히 지정되어 있어야 하며(SQLSTATE 428HR), 컬럼 수는 행 유형의 필드 수와 일치되어야 하며, 페치된 행 컬럼의 데이터 유형이 행 유형의 해당 필드에 지정 가능해야 합니다(SQLSTATE 42821).

*assignment-target*의 데이터 유형이 배열 요소이면, 정확히 지정된 하나의 *assignment-target*이 있어야 합니다.

global-variable-name

지정 목표인 전역 변수를 식별합니다.

host-variable-name

지정 목표인 호스트 변수를 식별합니다. LOB 출력 값의 경우, 목표는 일반 호스트 변수(충분히 큰 경우), LOB 로케이터 변수 또는 LOB 파일 참조 변수가 될 수 있습니다.

SQL-parameter-name

지정 목표인 이름 매개변수를 식별합니다.

SQL-variable-name

지정 목표인 SQL 변수를 식별합니다. SQL 변수는 사용하기 전에 선언해야 합니다.

transition-variable-name

전이 행에서 갱신될 컬럼을 식별합니다. *transition-variable-name*은 새로운 값을 식별하는 상관 이름에 의해 선택적으로 규정되는 트리거의 주제 테이블에 있는 컬럼을 식별해야 합니다.

array-variable-name

배열 유형의 SQL 변수, SQL 매개변수 또는 전역 변수를 식별합니다.

[array-index]

지정의 목표가 되는 배열에 있는 요소를 지정하는 표현식. 일반 배열의 경우, *array-index* 표현식은 INTEGER(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다. 배열에 정의된 1과 최대 카디널리티 사이의 값이어야 합니다(SQLSTATE 2202E). 연관된 배열의 경우, *array-index* 표현식은 연관된 배열의 인덱스 데이터 유형(SQLSTATE 428H1)에 지정 가능해야 하고 널(NULL) 값이 될 수 없습니다.

field-reference

지정 목표인 행 유형 값 내에서 필드를 식별합니다. *field-reference*는 규정된 *field-name*으로 지정되어야 하며 이때 규정자는 필드가 정의된 행 값을 식별합니다.

규칙

- 전역 변수는 트리거, 함수, 메소드 또는 복합 SQL(인라인된)문, 또는 이러한 오브젝트 중 하나에서 직접적 또는 간접적으로 호출되는 프로시저 내에서 지정할 수 없습니다(SQLSTATE 428GX).

예:

*예 1:*이 C 예는 CURRENT PATH 특수 레지스터의 값을 호스트 변수로 읽어들입니다.

```
EXEC SQL VALUES(CURRENT PATH)
      INTO :hv1;
```

예 2: 이 C 예는 LOB 필드의 부분을 호스트 변수로 검색하여, 지연된 검색에 대한 LOB 로케이터를 실행합니다.

```
EXEC SQL VALUES (substr(:locator1,35))
      INTO :details;
```

예 3: 이 C 예는 SESSION_USER 특수 레지스터의 값을 전역 변수로 검색합니다.

```
EXEC SQL VALUES(SESSION_USER)
  INTO GV_SESS_USER;
```

WHENEVER

WHENEVER문은 지정된 예외 조건이 발생할 때 취할 조치를 지정합니다.

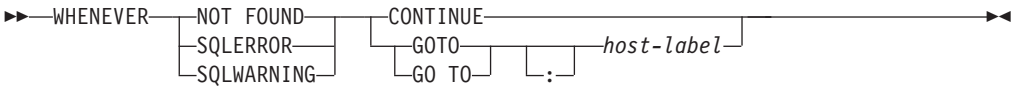
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 실행문이 아닙니다. 이 명령문은 REXX에서 지원되지 않습니다.

권한 부여

필요한 권한 없음

구문



설명

NOT FOUND, SQLERROR 또는 SQLWARNING절은 예외 조건의 유형을 식별하는 데 사용됩니다.

NOT FOUND
+100의 SQLCODE 또는 '02000'의 SQLSTATE를 생성하는 모든 조건을 식별합니다.

SQLERROR
음수 SQLCODE를 생성하는 모든 조건을 식별합니다.

SQLWARNING
경고 조건(SQLWARN0가 'W')을 생성하거나 +100 이외의 양수 SQL 리턴 코드를 생성하는 모든 조건을 식별합니다.

CONTINUE 또는 GO TO절은 식별된 예외 조건 유형이 존재할 때 발생하는 사항을 지정하는 데 사용됩니다.

CONTINUE
소스 프로그램의 다음 순차 지시사항이 실행되도록 합니다.

GOTO 또는 GO TO *host-label*
제어가 host-label로 식별되는 명령문으로 전달하도록 합니다. host-label에 선택적으로 콜론이 앞에 오는 단일 토큰을 대체하십시오. 토큰의 양식은 호스트 언어에 따라 다릅니다.

주

다음 세 가지 유형의 WHENEVER문이 있습니다.

- WHENEVER NOT FOUND
- WHENEVER SQLERROR
- WHENEVER SQLWARNING

프로그램의 모든 실행할 수 있는 SQL문은 각 유형의 하나의 내재적 또는 명시적 WHENEVER문의 범위에 있습니다. WHENEVER문의 범위는 실행 순서가 아니라 프로그램에서 명령문의 나열 시퀀스와 관련됩니다.

SQL문은 소스 프로그램에서 해당 SQL문 앞에 지정되는 각 유형의 마지막 WHENEVER문의 범위 안에 있습니다. 일부 유형의 WHENEVER문이 SQL문 앞에 지정되지 않는 경우, 해당 SQL문은 CONTINUE가 지정되는 해당 유형의 내재된 WHENEVER문의 범위에 있습니다.

예 :

다음 C 예에서, 오류가 발생하면 HANDLERR로 이동하십시오. 경고 코드가 생성되면 프로그램의 정상 순서를 계속하십시오. 데이터가 리턴되지 않으면 ENDDATA로 이동하십시오.

```
EXEC SQL WHENEVER SQLERROR GOTO HANDLERR;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND GO TO ENDDATA;
```

WHILE

WHILE 명령문은 지정된 조건이 참인 동안 하나의 명령문이나 명령문 그룹의 실행을 반복합니다.

호출

이 명령문은 다음에 임베디드(embedded)될 수 있습니다.

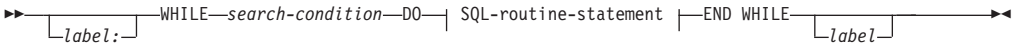
- SQL 프로시저 정의
- 복합 SQL(컴파일된) 명령문
- 복합 SQL(인라인된) 명령문

복합 명령문은 SQL 프로시저 정의, SQL 함수 정의 또는 SQL 트리거 정의에 임베디드(embedded)될 수 있습니다. 또한 실행문이 아니므로 동적으로 준비될 수 없습니다.

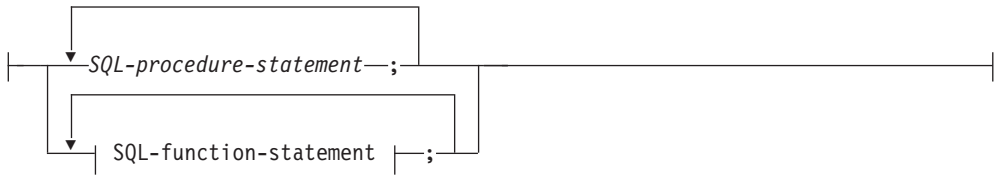
권한 부여

WHILE문을 호출하기 위해 필요한 특권은 없습니다. 단, 명령문의 권한 부여 ID가 SQL 문을 호출하고 WHILE문에서 임베디드(embedded)된 조건을 검색하는 데 필요한 특권을 보유해야 합니다.

구문



SQL-routine-statement:



설명

label

WHILE문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 이는 LEAVE 및 ITERATE문에 지정될 수 있습니다. 끝 레이블이 지정된 경우 이 레이블은 시작 레이블과 같아야 합니다.

search-condition

각 루프를 실행하기 전에 평가되는 조건을 지정합니다. 조건이 참인 경우, 루프의 SQL-procedure-statements가 처리됩니다.

SQL-procedure-statement

루프에서 실행할 SQL문을 지정합니다.*SQL-procedure-statement*는 SQL 프로시저의 컨텍스트 또는 복합 SQL(컴파일된) 명령문에서만 적용할 수 있습니다. 『복합 SQL(컴파일된)』 명령문의 *SQL-procedure-statement*를 참조하십시오.

SQL-function-statement

루프에서 실행할 SQL문을 지정합니다. *SQL-function-statement*는 SQL 트리거에서 임베디드(embedded)될 수 있는 SQL 함수 또는 복합 SQL(인라인된)문, SQL 함수 또는 SQL 메소드에서만 적용할 수 있습니다. 『FOR』의 *SQL-function-statement*를 참조하십시오.

예

이 예는 FETCH 및 SET문을 통해 반복하기 위한 WHILE문을 사용합니다. SQL 변수 값 *v_counter*는 IN 매개변수 *deptNumber*로 식별된 부서에 있는 직원의 절반 보다 작으나, WHILE문은 FETCH 및 SET문을 계속 수행합니다. 조건이 더 이상 참이 아니면, 제어의 플로우는 WHILE문으로 남고 커서를 닫습니다.

```
CREATE PROCEDURE DEPT_MEDIAN
  (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
  LANGUAGE SQL
  BEGIN
    DECLARE v_numRecords INTEGER DEFAULT 1;
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE c1 CURSOR FOR
      SELECT CAST(salary AS DOUBLE)
        FROM staff
        WHERE DEPT = deptNumber
        ORDER BY salary;
    DECLARE EXIT HANDLER FOR NOT FOUND
      SET medianSalary = 6666;
    SET medianSalary = 0;
    SELECT COUNT(*) INTO v_numRecords
      FROM staff
      WHERE DEPT = deptNumber;
    OPEN c1;
    WHILE v_counter < (v_numRecords / 2 + 1) DO
      FETCH c1 INTO medianSalary;
      SET v_counter = v_counter + 1;
    END WHILE;
    CLOSE c1;
  END
```

부록 A. DB2 기술 정보 개요

DB2 기술 정보는 다음 도구 및 메소드를 통해 사용할 수 있습니다.

- DB2 정보 센터
  - 주제 항목(태스크, 개념 및 참조 항목)
  - DB2 도구에 대한 도움말
  - 샘플 프로그램
  - 자습서
- DB2 서적
  - PDF 파일(다운로드)
  - PDF 파일(DB2 PDF DVD)
  - 인쇄된 서적
- 명령행 도움말
  - 명령 도움말
  - 메시지 도움말

주: DB2 정보 센터의 주제는 PDF 또는 하드카피 서적보다 더 자주 갱신됩니다. 최신 정보를 보려면 사용 가능한 문서 갱신사항을 설치하거나 ibm.com에서 DB2 정보 센터를 참조하십시오.

ibm.com에서 추가 DB2 기술 정보(예: 기술 노트, 백서 및 IBM Redbooks® 서적)를 온라인으로 액세스할 수 있습니다. 다음은 DB2 정보 관리 라이브러리 소프트웨어 사이트의 주소입니다. http://www.ibm.com/software/data/sw-library/

문서 피드백

DB2 문서에 대한 피드백을 환영합니다. DB2 문서를 향상시키는 방법에 대해서 제안사항이 있는 경우 db2docs@ca.ibm.com으로 전자 우편을 보내십시오. DB2 문서 팀에서는 고객의 모든 피드백을 읽지만 직접 응답할 수는 없습니다. 고객의 문제를 더 잘 이해할 수 있도록 가능한 한 구체적인 예를 제공해주십시오. 특정 주제 또는 도움말 파일에 대한 피드백을 보내실 경우, 제목 및 URL을 알려주십시오.

DB2 고객 지원에 문의할 때는 이 전자 우편 주소를 사용하지 마십시오. 문서에서 해결할 수 없는 DB2 기술 문제점이 있는 경우, 해당 지역의 IBM 서비스 센터에 도움을 요청하십시오.

DB2 기술 라이브러리(하드카피 또는 PDF 형식)

다음 표는 IBM Publications Center(www.ibm.com/shop/publications/order)에서 사용할 수 있는 DB2 라이브러리에 대한 설명입니다. PDF 형식의 영문 DB2 버전 9.7 매뉴얼 및 번역된 버전은 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947에서 다운로드할 수 있습니다.

표에 인쇄할 수 있는 책으로 설명된 경우라도, 사용 국가 또는 지역에 따라 해당 책을 사용할 수 없을 수도 있습니다.

매뉴얼이 갱신될 때마다 문서 번호가 증가합니다. 다음 사항을 참조하여 읽고 있는 매뉴얼이 최신 버전인지 확인하십시오.

주: *DB2 정보 센터*는 PDF 또는 하드카피 서적보다 자주 갱신됩니다.

표 33. DB2 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
관리 API 참조서	SA30-3958-00	예	2009년 8월
관리 루틴 및 뷰	SA30-3955-00	아니오	2009년 8월
Call Level Interface Guide and Reference, Volume 1	SC27-2437-00	예	2009년 8월
Call Level Interface Guide and Reference, Volume 2	SC27-2438-00	예	2009년 8월
명령어 참조서	SA30-3959-00	예	2009년 8월
데이터 이동 유틸리티 안내서 및 참조서	SA30-3969-00	예	2009년 8월
데이터 복구 및 고가용성 안내서 및 참조서	SA30-3970-00	예	2009년 8월
데이터베이스 관리 개념 및 구성 참조서	SA30-3951-00	예	2009년 8월
데이터베이스 모니터링 안내서 및 참조서	SA30-3953-00	예	2009년 8월
데이터베이스 보안 안내서	SA30-3971-00	예	2009년 8월
DB2 Text Search Guide	SC27-2459-00	예	2009년 8월
Developing ADO.NET and OLE DB Applications	SC27-2444-00	예	2009년 8월
Developing Embedded SQL Applications	SC27-2445-00	예	2009년 8월
Developing Java Applications	SC27-2446-00	예	2009년 8월

표 33. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-2447-00	아니오	2009년 8월
Developing User-defined Routines(SQL and External)	SC27-2448-00	예	2009년 8월
Getting Started with Database Application Development	GI11-9410-00	예	2009년 8월
Linux 및 Windows에서 DB2 설치 및 관리 시작하기	GA30-3960-00	예	2009년 8월
자국어 안내서	SA30-3972-00	예	2009년 8월
DB2 Server 설치	GA30-3962-00	예	2009년 8월
IBM Data Server Client 설치	GA30-3963-00	아니오	2009년 8월
Message Reference Volume 1	SC27-2450-00	아니오	2009년 8월
Message Reference Volume 2	SC27-2451-00	아니오	2009년 8월
Net Search Extender Administration and User's Guide	SC27-2469-00	아니오	2009년 8월
파티셔닝 및 클러스터링 안내서	SA30-3973-00	예	2009년 8월
pureXML Guide	SC27-2465-00	예	2009년 8월
Query Patroller 관리 및 사용자 안내서	SA30-3974-00	아니오	2009년 8월
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC27-2468-00	아니오	2009년 8월
SQL Procedural Languages: Application Enablement and Support	SC27-2470-00	예	2009년 8월
SQL 참조서, 볼륨 1	SA30-3956-00	예	2009년 8월
SQL 참조서, 볼륨 2	SA30-3957-00	예	2009년 8월
문제점 해결 및 데이터베이스 성능 조정	SA30-3952-00	예	2009년 8월
DB2 버전 9.7로 업그레이드	SA30-3961-00	예	2009년 8월
Visual Explain 자습서	SA30-3968-00	아니오	2009년 8월

표 33. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
DB2 버전 9.7의 새로운 내용	SA30-3967-00	예	2009년 8월
Workload Manager Guide and Reference	SC27-2464-00	예	2009년 8월
XQuery Reference	SC27-2466-00	아니오	2009년 8월

표 34. DB2 Connect 특정 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
DB2 Connect Personal Edition 설치 및 구성	SA30-3965-00	예	2009년 8월
DB2 Connect Server 설치 및 구성	SA30-3966-00	예	2009년 8월
DB2 Connect 사용자 안내서	SA30-3964-00	예	2009년 8월

표 35. Information Integration 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	예	2009년 8월
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	예	2009년 8월
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	아니오	2009년 8월
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	예	2009년 8월
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	예	2009년 8월

인쇄된 DB2 서적 주문

인쇄된 DB2 서적이 필요한 경우, 대부분 온라인으로 구매할 수 있으나 모든 국가 또는 지역에서 가능한 것은 아닙니다. 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. *DB2 PDF 문서* DVD의 일부 소프트카피 서적은 인쇄할 수 없다는 점에 유의하십시오. 예를 들어, *DB2 메시지 참조서*의 볼륨은 인쇄된 서적으로 사용할 수 없습니다.

DB2 PDF 문서 DVD에서 사용할 수 있는 다수의 DB2 서적의 인쇄된 버전은 IBM에서 유료로 주문할 수 있습니다. 주문하는 위치에 따라 IBM Publications Center에서 온라인으로 서적을 주문할 수도 있습니다. 해당 국가 또는 지역에서 온라인 주문이 불가능하면, 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. DB2 PDF 문서 DVD의 모든 서적을 인쇄할 수는 없다는 점에 유의하십시오.

주: 가장 최신의 완전한 DB2 문서는 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7의 DB2 정보 센터에서 유지보수됩니다.

인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.

- 해당 국가 또는 지역에서 인쇄된 DB2 서적을 온라인으로 주문할 수 있는지 여부를 확인하려면 http://www.ibm.com/shop/publications/order의 IBM Publications Center를 확인하십시오. 서적 주문 정보를 액세스하려면 국가/지역/언어를 선택한 다음 해당 위치에서 주문 지시사항을 따르십시오.
- 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.
  1. 다음 웹 사이트 중 하나에서 해당 지역 담당자에 대한 문의처 정보를 찾으십시오.
     - www.ibm.com/planetwide에 있는 IBM 전세계 문의처 디렉토리
     - http://www.ibm.com/shop/publications/order의 IBM Publications 웹 사이트. 사용 지역의 해당 서적 홈 페이지에 액세스하려면 해당 국가, 지역 또는 언어를 선택해야 합니다. 이 페이지에서 "이 제품의 정보" 링크를 수행하십시오.
  2. 전화로 주문할 경우, 주문할 DB2 서적을 지정하십시오.
  3. 담당자에게 주문하려는 서적의 제목 및 문서 번호를 제공하십시오. 서적의 제목 및 문서 번호는 1366 페이지의 『DB2 기술 라이브러리(하드카피 또는 PDF 형식)』를 참조하십시오.

명령행 처리기에서 SQL 상태 도움말 표시

DB2 제품은 SQL문의 결과로 나타나는 상태에 대한 SQLSTATE 값을 리턴합니다. SQLSTATE 도움말은 SQL 상태 및 SQL 상태 클래스 코드의 의미를 설명합니다.

SQL 상태 도움말을 시작하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? sqlstate or ? class code
```

여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자를 나타냅니다.
예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

DB2 정보 센터의 다른 버전에 액세스

DB2 버전 9.7 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/입니다.

DB2 버전 9.5 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/입니다.

DB2 버전 9 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9/입니다.

DB2 버전 8 주제에 대한 버전 8 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v8/입니다.

DB2 정보 센터에서 원하는 언어로 항목 표시

DB2 정보 센터는 브라우저 환경 설정에 지정된 언어로 주제 항목을 표시합니다. 주제가 원하는 언어로 변환되지 않은 경우, DB2 정보 센터는 해당 주제 항목을 영어로 표시합니다.

- Internet Explorer 브라우저에서 원하는 언어로 항목을 표시하려면 다음을 수행하십시오.
  1. Internet Explorer에서 **도구 —> 인터넷 옵션 —> 언어** 단추를 누르십시오. 언어 환경 설정 창이 열립니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 누르십시오.

       주: 언어를 추가하더라도 원하는 언어로 항목을 표시하는 데 필요한 글꼴이 컴퓨터에 설치되지는 않습니다.

   – 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 해당 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
3. 브라우저 캐시를 지운 후 페이지를 새로 고치면 원하는 언어로 DB2 정보 센터가 표시됩니다.

- Firefox 또는 Mozilla 브라우저에서 원하는 언어로 주제 항목을 표시하려면 다음을 수행하십시오.
  1. **도구 —> 설정 —> 내용** 대화 상자의 **언어** 섹션에서 단추를 선택하십시오. 환경 설정 창에 언어 패널이 표시됩니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     – 목록에 새 언어를 추가하려면 언어 선택 창에서 원하는 언어를 선택한 다음 **추가...** 단추를 누르십시오.
     – 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 해당 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
  3. 브라우저 캐시를 지운 후 페이지를 새로 고치면 원하는 언어로 DB2 정보 센터가 표시됩니다.

일부 브라우저 및 운영 체제 조합에서는 운영 체제의 국가별 설정을 선택한 로케일 및 언어로 변경해야 합니다.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신

로컬로 설치된 DB2 정보 센터는 주기적으로 갱신해야 합니다.

시작하기 전에

DB2 버전 9.7 정보 센터는 미리 설치된 상태여야 합니다. 자세한 내용은 *DB2 Server 설치*의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치』 주제를 참조하십시오. 정보 센터 설치에 적용되는 모든 전제조건 및 제한사항은 정보 센터 갱신에도 적용됩니다.

이 태스크에 대한 정보

기존의 DB2 정보 센터는 자동 또는 수동으로 갱신할 수 있습니다.

- 자동 갱신 - 기존 정보 센터 기능 및 언어를 갱신합니다. 자동 갱신의 또 다른 이점으로는 갱신 동안 정보 센터를 사용할 수 없는 시간이 매우 짧다는 점입니다. 또한 자동 갱신은 주기적으로 실행되는 기타 일괄처리 작업의 일부로 실행되도록 설정할 수도 있습니다.
- 수동 갱신 - 갱신 프로세스 중에 기능이나 언어를 추가하려는 경우 사용하십시오. 예를 들어, 로컬 정보 센터는 기본적으로 영어와 프랑스어로 설치되어 있으며, 수동 갱신을 통해 기존 정보 센터의 기능 및 언어 갱신뿐만 아니라 독일어도 설치할 수 있

습니다. 단, 수동 갱신을 수행하려면 정보 센터를 중지한 다음 갱신하고 재시작해야 합니다. 정보 센터는 갱신 프로세스 동안에는 사용할 수 없습니다.

프로시저

이 주제는 자동 갱신 프로세스에 대한 설명입니다. 수동 갱신에 대한 지시사항은 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신』 주제를 참조하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 자동으로 갱신하려면 다음을 수행하십시오.

1. Linux 운영 체제의 경우
   a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.
   b. 설치 디렉토리에서 `doc/bin` 디렉토리로 이동하십시오.
   c. 다음과 같이 `ic-update` 스크립트를 실행하십시오.

      ```
      ic-update
      ```
2. Windows 운영 체제의 경우
   a. 명령 창을 여십시오.
   b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `<Program Files>₩IBM₩DB2 Information Center₩Version 9.7` 디렉토리에 디폴트로 설치됩니다. 여기서 `<Program Files>`는 프로그램 파일 디렉토리의 위치를 나타냅니다.
   c. 설치 디렉토리에서 `doc₩bin` 디렉토리로 이동하십시오.
   d. 다음과 같이 `ic-update.bat` 파일을 실행하십시오.

      ```
      ic-update.bat
      ```

결과

DB2 정보 센터가 자동으로 재시작됩니다. 갱신사항이 사용 가능한 경우, 정보 센터에는 새로 갱신된 주제가 표시됩니다. 정보 센터 갱신을 사용할 수 없는 경우, 메시지가 로그에 추가됩니다. 로그 파일은 `doc₩eclipse₩configuration` 디렉토리에 있습니다. 이 로그 파일 이름은 임의로 생성된 번호입니다. 예: `1239053440785.log`

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신

DB2 정보 센터를 로컬로 설치한 경우, IBM으로부터 문서 갱신사항을 받아 설치할 수 있습니다.

로컬로 설치된 DB2 정보 센터를 수동으로 갱신하려면 다음을 수행하십시오.

1. 컴퓨터에서 DB2 정보 센터를 중지한 후 독립형 모드에서 다시 시작하십시오. 독립형 모드에서 정보 센터를 실행하면 사용자의 네트워크와 연결된 다른 사용자는 정보 센터에 액세스할 수 없으므로 갱신사항을 적용할 수 있습니다. DB2 정보 센터의 워크스테이션 버전은 항상 독립형 모드에서 실행됩니다.
2. 사용 가능한 갱신사항을 확인하려면 갱신 기능을 사용하십시오. 설치해야 할 갱신사항이 있는 경우, 갱신 기능을 사용하여 이를 가져온 후 설치할 수 있습니다.

   주: 인터넷에 연결되지 않은 머신에 DB2 정보 센터 갱신사항을 설치해야 할 경우, 인터넷에 연결되고 DB2 정보 센터가 설치된 머신을 사용하여 갱신 사이트를 로컬 파일 시스템으로 미러하십시오. 네트워크 상에 문서 갱신사항을 설치하려는 사용자가 많을 경우에는 갱신 사이트를 로컬로 미러링하거나 갱신 사이트의 프록시를 작성하여 갱신을 수행하면 각 개인에게 필요한 시간을 줄일 수 있습니다.
   갱신 패키지가 사용 가능하면 갱신 기능을 사용하여 패키지를 가져오십시오. 그러나 갱신 기능은 독립형 모드에서만 사용할 수 있습니다.
3. 독립형 정보 센터를 중지한 후 컴퓨터에서 DB2 정보 센터를 재시작하십시오.

주: Windows 2008, Windows Vista 이상의 경우 이 절 다음에 나오는 명령은 관리자로 실행해야 합니다. 전체 관리자 권한으로 명령 프롬프트 또는 그래픽 도구를 열려면 단축 아이콘을 마우스 오른쪽 단추로 누른 후 **관리자로 실행**을 선택하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 갱신하려면 다음을 수행하십시오.

1. DB2 정보 센터를 중지하십시오.
   - Windows의 경우, **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 Information Center** 서비스를 마우스 오른쪽 단추로 누른 후 **중지**를 선택하십시오.
   - Linux의 경우, 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 독립형 모드에서 정보 센터를 시작하십시오.
   - Windows의 경우:
     a. 명령 창을 여십시오.
     b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.
     c. 설치 디렉토리에서 doc₩bin 디렉토리로 이동하십시오.
     d. 다음과 같이 help_start.bat 파일을 실행하십시오.

        ```
        help_start.bat
        ```

- Linux의 경우:
  a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 /opt/ibm/db2ic/V9.7 디렉토리에 디폴트로 설치됩니다.
  b. 설치 디렉토리에서 doc/bin 디렉토리로 이동하십시오.
  c. 다음과 같이 help_start 스크립트를 실행하십시오.

     ```
     help_start
     ```

시스템의 기본 웹 브라우저가 열리고 독립형 정보 센터가 표시됩니다.

3. **갱신** 단추( )를 누르십시오. (JavaScript™가 브라우저에서 사용 가능해야 합니다.) 정보 센터의 오른쪽 패널에서 **갱신사항 찾기**를 누르십시오. 기존 문서의 갱신사항 목록이 표시됩니다.
4. 설치 프로세스를 시작하려면 설치할 선택란을 체크한 후 **갱신사항 설치**를 누르십시오.
5. 설치 프로세스가 완료되면 **완료**를 누르십시오.
6. 독립형 정보 센터를 중지하십시오.
   - Windows의 경우, 설치 디렉토리의 doc¥bin 디렉토리로 이동한 후 다음과 같이 help_end.bat 파일을 실행하십시오.

     ```
     help_end.bat
     ```

     주: help_end 일괄처리 파일에는 help_start 일괄처리 파일로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start.bat를 중지할 때 Ctrl+C 또는 다른 메소드를 사용하지 마십시오.
   - Linux의 경우, 설치 디렉토리의 doc/bin 디렉토리로 이동한 후 다음과 같이 help_end 스크립트를 실행하십시오.

     ```
     help_end
     ```

     주: help_end 스크립트에는 help_start 스크립트로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start 스크립트를 중지할 때 다른 메소드를 사용하지 마십시오.
7. DB2 정보 센터를 재시작하십시오.
   - Windows의 경우, **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 Information Center** 서비스를 마우스 오른쪽 단추로 누른 후 **시작**을 선택하십시오.
   - Linux의 경우, 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 start
     ```

갱신된 DB2 정보 센터에는 새로 갱신된 주제가 표시됩니다.

DB2 자습서

DB2 자습서는 DB2 제품의 여러가지 측면을 학습하는 데 유용합니다. 각 레슨은 단계별 지시사항을 제공합니다.

시작하기 전에

정보 센터(http://publib.boulder.ibm.com/infocenter/db2help/)에서 XHTML 버전의 자습서를 볼 수 있습니다.

일부 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크에 필요한 전제조건 설명은 자습서를 참조하십시오.

DB2 자습서

자습서를 보려면 제목을 누르십시오.

『**pureXML**』(*pureXML Guide*)
: DB2 데이터베이스를 설정하여 XML 데이터를 저장하고 원시 XML 데이터 스토어로 기본 조작을 수행할 수 있습니다.

*Visual Explain 자습서*의 『**Visual Explain**』
: 더 나은 성능을 위해 Visual Explain을 사용하여 SQL문을 분석, 최적화 및 조정할 수 있습니다.

DB2 문제점 해결 정보

DB2 데이터베이스 제품 사용 시 발생하는 광범위한 문제점을 판별하고 해결하는 데 도움이 되는 정보를 사용할 수 있습니다.

DB2 문서
: 문제점 해결 정보는 *DB2 문제점 해결 안내서* 또는 *DB2 정보 센터*의 데이터베이스 기본 절을 참조하십시오. DB2 진단 도구 및 유틸리티를 사용하여 문제점을 찾아내고 식별하는 방법, 가장 일반적인 문제점에 대한 솔루션 및 DB2 데이터베이스 제품에서 발생할 수 있는 문제점을 해결하는 방법 등에 관한 정보가 있습니다.

DB2 기술 지원 웹 사이트
: 문제점이 발생한 경우 해당 원인 및 솔루션을 찾으려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 서적, 기술 노트, APAR(Authorized Program Analysis Report 또는 버그 수정), FixPack 및 기타 자원에 대한 링크가 있습니다. 이러한 기술 자료를 검색하여 문제에 대해 사용 가능한 솔루션을 찾을 수 있습니다.

  다음은 DB2 기술 지원 웹 사이트의 주소입니다. http://www.ibm.com/software/data/db2/support/db2_9/

이용약관

다음 조건에 따라 이 책을 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 이 책을 개인적, 비상업적 용도로 복제할 수 있습니다. IBM의 명시적인 동의 없이는 이 책 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 이 책을 귀하 기업 집단 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 이 책의 2차적 저작물을 만들거나 이 책 또는 그 일부를 복제, 배포 또는 전시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 이 책이나 이 책에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대한 어떠한 허가나 라이센스 또는 권한도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 이 책의 사용이 IBM의 이익을 해친다고 판단되거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 이 사이트에서 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 경우에만 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 이 책의 내용에 대해 어떠한 보증도 제공하지 않습니다. 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 현 상태대로 제공합니다.

부록 B. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. 비IBM 제품에 대한 정보는 이 책을 처음 발행할 때의 정보에 기초하고 있으며 변경될 수 있습니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트 문자 세트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

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

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠. 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등) 하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 다른 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 되는 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 샘플 프로그램은 어떠한 보증없이 ″있는 그대로″ 제공됩니다. IBM은 샘플 프로그램의 사용으로 인해 발생하는 모든 손해에 대해 책임을 지지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. *_enter* 연도_. All rights reserved.

상표

IBM, IBM 로고 및 ibm.com®은 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 기타 회사의 상표입니다. 현재 IBM 상표 목록은 웹 “저작권 및 상표 정보”(www.ibm.com/legal/copytrade.shtml)에 있습니다.

다음 용어는 기타 회사의 상표 또는 등록상표입니다.

- Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.
- Java 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.
- UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.
- Intel®, Intel 로고, Intel Inside®, Intel Inside 로고, Intel® Centrino®, Intel Centrino 로고, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium® 및 Pentium®은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표 또는 등록상표입니다.
- Microsoft, Windows, Windows NT® 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W

X

SA30-3957-00

Spine information:

Linux, UNIX 및 Windows용 IBM DB2 9.7

SQL 참조서, 볼륨 2