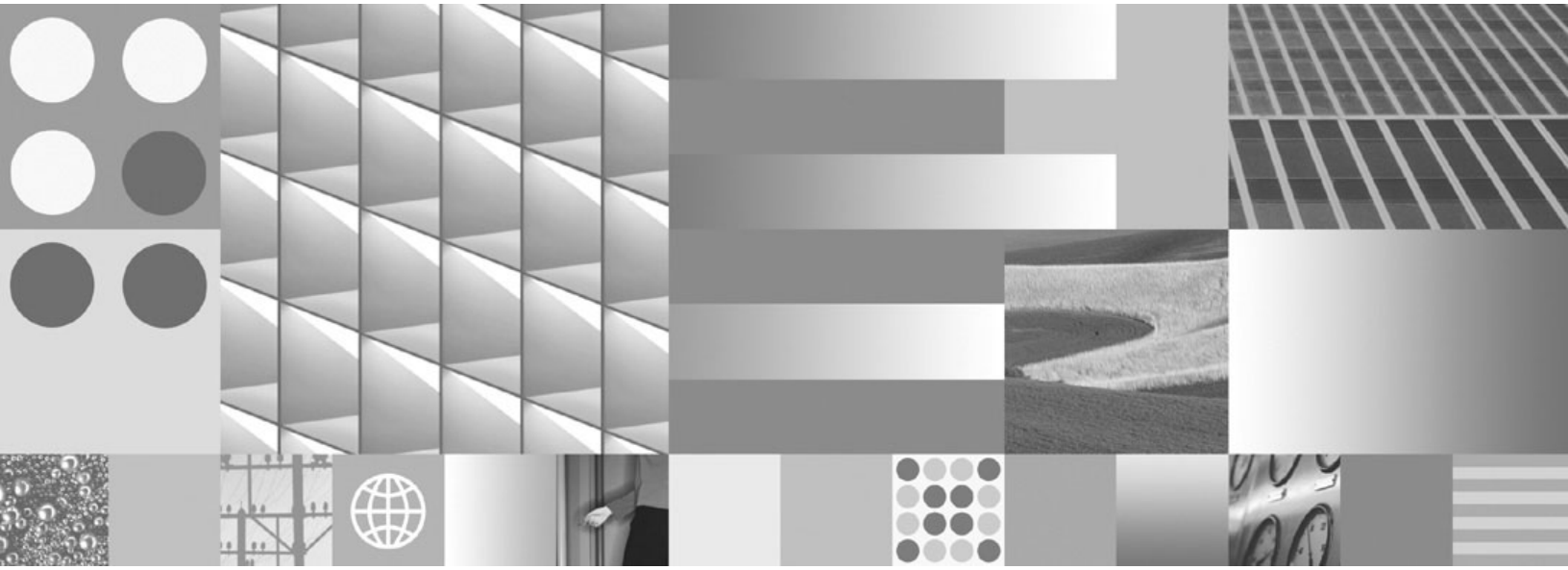
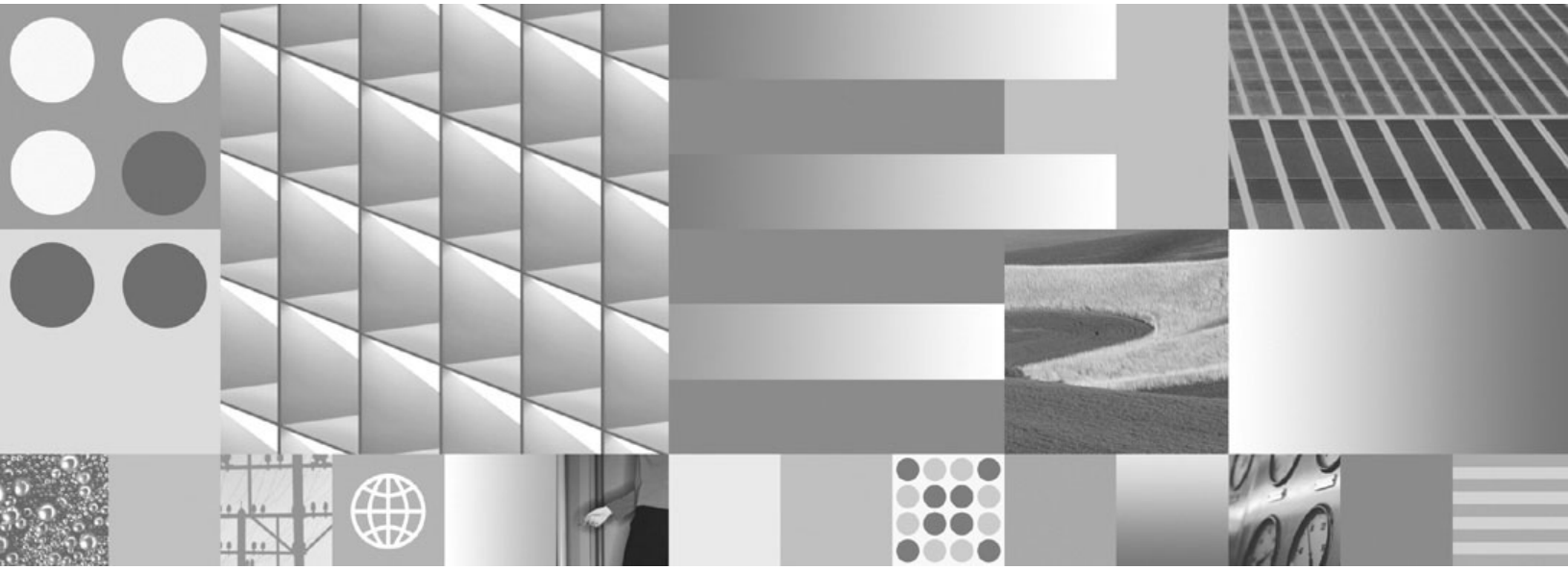
Linux, UNIX 및 Windows용
IBM DB2 9.7



데이터 이동 유틸리티 안내서 및 참조서

SA30-3969-00

Linux, UNIX 및 Windows용
IBM DB2 9.7



데이터 이동 유틸리티 안내서 및 참조서

SA30-3969-00

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 541 페이지의 부록 F 『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

이 책에서는 Linux®, UNIX® 및 Windows®용 DB2® 데이터베이스 데이터 이동 유틸리티를 사용하는 방법에 대한 정보와 사용법을 설명합니다.

- 익스포트 및 임포트

  임포트 및 익스포트 유틸리티는 테이블 또는 뷰와 다른 데이터베이스나 스프레드시트 프로그램 간, DB2 데이터베이스 간, DB2 데이터베이스와 DB2® Connect™를 사용하는 호스트 데이터베이스 간에 데이터를 이동시킵니다. 익스포트 유틸리티가 데이터베이스에서 운영 체제 파일로 데이터를 이동시키므로, 사용자가 이들 파일을 사용하여 다른 데이터베이스로 데이터를 임포트하거나 로드할 수 있습니다.

- 로드

  로드 유틸리티는 데이터를 테이블로 이동시켜 기존의 인덱스를 확장하고 통계를 생성합니다. 로드 유틸리티는 대량의 데이터가 관련될 경우 임포트 유틸리티보다 더 빠르게 데이터를 이동시킵니다. 익스포트 유틸리티를 사용하여 익스포트된 데이터는 로드 유틸리티를 사용하여 로드될 수 있습니다.

  로드 유틸리티를 파티션된 데이터베이스 환경에서 사용하는 경우, 대량의 데이터를 다른 데이터베이스 파티션으로 분산 및 로드할 수 있습니다.

완전한 데이터 이동 옵션 목록은 DB2 V9.5에서 사용 가능한 데이터 이동 옵션을 참조하십시오.

제 1 부 데이터 이동 유틸리티 및 참조

제 1 장 데이터 이동 옵션

다양한 데이터 이동 옵션을 사용할 수 있습니다. 다음 표는 사용 가능한 데이터 이동 도구 및 유틸리티에 대한 개요를 제공합니다. 이 표를 지침으로 삼아 요구에 가장 적합한 데이터 이동 옵션을 판별할 수 있습니다.

표 1. 데이터 이동 옵션

유틸리티 이름	로드 유틸리티
용도	새로 작성한 테이블 또는 이미 데이터를 포함하는 테이블로 많은 양의 데이터를 효과적으로 이동합니다.
교차 플랫폼 호환 가능 여부	예
우수 사례 사용법	이 유틸리티는 성능이 주된 관심사인 경우에 가장 적합합니다. 이 유틸리티는 임포트 유틸리티의 대안으로 사용될 수 있습니다. SQL INSERTS를 사용하지 않고 형식화된 페이지를 데이터베이스에 직접 작성하므로 임포트 유틸리티보다 빠릅니다. 또한 로드 유틸리티를 사용하면 옵션을 통해 데이터를 로그하지 않거나 COPY 옵션으로 로드된 데이터 사본을 저장할 수 있습니다. 로드 조작은 자원(예: SMP 및 MPP 환경에서의 CPU 및 메모리)을 완전히 사용할 수 있습니다.
참조	데이터 로드

유틸리티 이름	**db2move**
용도	COPY 옵션과 함께 db2move 유틸리티를 사용하면 소스 데이터베이스에서 목표 데이터베이스로 스키마 템플리트(데이터를 포함할 수도 있고 포함하지 않을 수도 있음)를 복사하거나 소스 데이터베이스에서 목표 데이터베이스로 전체 스키마를 이동할 수 있습니다. IMPORT 또는 EXPORT 옵션과 함께 db2move 유틸리티를 사용하면 DB2 데이터베이스 사이에서 많은 테이블을 쉽게 이동할 수 있습니다.
교차 플랫폼 호환 가능 여부	예
우수 사례 사용법	COPY 옵션을 사용하는 경우 소스 및 목표 데이터베이스가 서로 달라야 합니다. COPY 옵션은 스키마 템플리트를 작성하는 데 유용합니다. 여러 플랫폼에 걸친 백업 및 리스토어 작업이 지원되지 않는 클론 데이터베이스의 경우 IMPORT 또는 EXPORT 옵션을 사용합니다. IMPORT 및 EXPORT 옵션은 db2look 명령과 함께 사용합니다.
참조	• *데이터베이스 관리 개념 및 구성 참조서*의 『스키마 복사』 • 임포트된 테이블 재작성

유틸리티 이름	임포트 유틸리티
용도	외부 파일에서 테이블, 계층 구조, 뷰 또는 별칭으로 데이터를 삽입합니다.
교차 플랫폼 호환 가능 여부	예
우수 사례 사용법	임포트 유틸리티는 다음 경우에 로드 유틸리티의 바람직한 대안이 될 수 있습니다. • 목표 테이블이 뷰인 경우 • 목표 테이블에 제한조건이 있고 목표 테이블을 무결성 설정 보류 상태로 두지 않으려는 경우 • 목표 테이블에 트리거가 있고 이를 시작하려는 경우
참조	데이터 임포트

유틸리티 이름	익스포트 유틸리티
용도	데이터베이스에서 여러 외부 파일 형식 중 하나로 데이터를 익스포트합니다. 데이터는 나중에 임포트 또는 로드할 수 있습니다.
교차 플랫폼 호환 가능 여부	예
우수 사례 사용법	이 유틸리티는 외부 파일에 데이터를 저장하거나 처리를 계속하거나 다른 테이블로 데이터를 이동하는 경우 가장 적합합니다. HPU(High Performance Unload)를 사용할 수도 있지만 이는 별도로 구입해야 합니다. 익스포트는 XML 컬럼을 지원합니다.
참조	데이터 익스포트

유틸리티 이름	**ADMIN_COPY_SCHEMA** 프로시저
용도	단일 스키마에서 모든 오브젝트를 복사하고 이러한 오브젝트를 새 스키마에서 재작성할 수 있습니다. 이 복사 조작은 데이터베이스 내 데이터를 포함하거나 포함하지 않는 경우 모두 수행할 수 있습니다.
교차 플랫폼 호환 가능 여부	예

유틸리티 이름	**ADMIN_COPY_SCHEMA 프로시저**
우수 사례 사용법	이 유틸리티는 스키마 템플리트를 작성하는 데 유용합니다. 소스 스키마의 동작에 영향을 주지 않고 스키마를 검사하는 경우(새 인덱스 사용 시도)에도 유용합니다. ADMIN_COPY_SCHEMA 프로시저 및 db2move 유틸리티의 가장 큰 다른 점은 다음과 같습니다. • ADMIN_COPY_SCHEMA 프로시저는 단일 데이터베이스에서 사용되지만 db2move 유틸리티는 여러 데이터베이스 사이에서 사용됩니다. • 테이블 또는 인덱스와 같은 실제 오브젝트를 작성할 수 없는 경우 호출되면 db2move 유틸리티에 실패합니다. ADMIN_COPY_SCHEMA 프로시저는 오류를 로그하고 계속 진행됩니다. • ADMIN_COPY_SCHEMA 프로시저는 cursor의 로드를 사용하여 한 스키마에서 다른 스키마로 데이터를 이동합니다. db2move 유틸리티는 cursor의 로드와 비슷한 리모트 로드를 사용합니다. 이때 리모트 로드에서는 소스 데이터베이스에서 데이터를 검색합니다.
참조	*데이터베이스 관리 개념 및 구성 참조서*의 『스키마 복사』

유틸리티 이름	**REDIRECT 옵션 및 GENERATE SCRIPT 옵션을 사용하는 리스토어 유틸리티**
용도	기존 백업 이미지의 스크립트를 사용하여 한 시스템에서 다른 시스템으로 전체 데이터베이스를 복사합니다.
교차 플랫폼 호환 가능 여부	제한됨. 참조 확인
우수 사례 사용법	이 유틸리티는 백업 이미지가 존재하는 경우에 가장 적합합니다.
참조	• *데이터 복구 및 고가용성 안내서 및 참조서*의 『자동으로 생성된 스크립트를 사용하여 경로 재지정된 리스토어 수행』 • *데이터 복구 및 고가용성 안내서 및 참조서*의 『서로 다른 운영 체제 및 하드웨어 플랫폼 간 백업 및 리스토어 작업』

유틸리티 이름	**db2relocatedb - 데이터베이스 재배치 명령**
용도	데이터베이스 이름을 바꾸거나 데이터베이스 또는 데이터베이스 일부를 동일한 시스템 또는 다른 시스템으로 다시 배치합니다.
교차 플랫폼 호환 가능 여부	아니오

유틸리티 이름	**db2relocatedb - 데이터베이스 재배치 명령**
우수 사례 사용법	• 이 유틸리티는 백업 및 리스토어가 오래 걸리는 경우 사용할 수 있습니다. • 이 유틸리티는 백업 및 리스토어를 사용하여 데이터베이스 사본을 작성하거나 이동하는 방법 대신 사용할 수 있습니다. • 또한 테스트와 같은 대체 환경에서 데이터베이스를 복제하는 빠른 방법을 제공하기도 합니다. • 이를 사용하여 테이블 스페이스 컨테이너를 새 스토리지 디바이스 세트로 이동할 수 있습니다.
참조	*명령어 참조서*의 『db2relocatedb - 데이터베이스 재배치 명령』

유틸리티 이름	**분할 미러**
용도	클론, 대기 또는 데이터베이스 백업을 작성합니다.
교차 플랫폼 호환 가능 여부	아니오
우수 사례 사용법	• 1차 장애가 발생한 경우 중단 시간을 줄이기 위해 대기 시스템 작성 • 실시간 프로덕션 머신에서 분할 데이터베이스로 백업 작업 이동 • 테스트와 같은 대체 환경에서 데이터베이스를 복제하는 빠른 방법 제공
고려사항	• 데이터베이스의 분할 버전에서 DMS 테이블 스페이스만 백업할 수 있음 • 일반적으로 스토리지 시스템에서 제공하는 일부 flashcopy 기술과 함께 사용됩니다. • 데이터베이스가 일시중단되면 파일 사본을 실행하는 대안이 있습니다. 그러나 그러면 데이터베이스 스토리지 크기가 중복됩니다.
참조	*데이터 복구 및 고가용성 안내서 및 참조서*의 『온라인 분할 미러 및 일시중단된 입출력 지원을 통한 고가용성』

제 2 장 익스포트 유틸리티

익스포트 유틸리티 개요

익스포트 유틸리티에서는 SQL select 또는 XQuery문을 사용하여 데이터를 추출하고 해당 정보를 파일에 저장합니다. 출력 파일을 사용하여 향후 임포트 또는 로드 조작을 위해 데이터를 이동하거나 분석을 위해 데이터에 액세스할 수 있도록 할 수 있습니다.

익스포트 유틸리티는 비교적 단순하지만 유연한 데이터 이동 유틸리티입니다. 제어 센터를 사용하거나 CLP에서 EXPORT 명령을 실행하거나 ADMIN_CMD 스토어드 프로시저를 호출하거나 사용자 응용프로그램을 통해 db2Export API를 호출하여 활성화할 수 있습니다.

다음 항목은 기본 익스포트 조작에 필수 사항입니다.

- 익스포트한 데이터를 저장하려는 운영 체제 파일의 경로 및 이름
- 입력 파일에서 데이터 형식
  익스포트에서는 출력 파일로 IXF, WSF 및 DEL 데이터 형식을 지원합니다.
- 익스포트할 데이터의 스펙
  대부분의 익스포트 조작에서는 익스포트를 위해 검색할 데이터를 지정하는 SELECT 문을 제공해야 합니다. 유형이 지정된 테이블을 익스포트하는 경우 SELECT문을 명시적으로 발행하지 않아도 됩니다. 계층 구조에서 서브테이블 트래버스 순서를 지정하기만 하면 됩니다.

IXF 형식으로 데이터를 이동해야 하는 경우 DB2 Connect에서 익스포트 유틸리티를 사용할 수 있습니다.

추가 옵션

익스포트 조작을 사용자 정의할 수 있는 여러 매개변수가 있습니다. 파일 유형 수정자에서는 데이터 형식, 날짜 및 시간 소인 또는 코드 페이지를 변경하는 기능과 같은 여러 옵션을 제공하거나 별도의 파일에 특정 데이터 유형을 기록하기도 합니다. **METHOD** 매개변수를 사용하면 익스포트된 데이터에서 사용할 다른 컬럼 이름을 지정할 수 있습니다.

XML 데이터 유형이 있는 하나 이상의 컬럼을 포함하는 테이블에서 익스포트할 수 있습니다. **XMLFILE**, **XML TO** 및 **XMLSAVESCHEMA** 매개변수를 사용하여 익스포트한 문서를 저장하는 방법에 대한 세부사항을 지정하십시오.

익스포트 유틸리티의 성능을 향상시키는 몇 가지 방법이 있습니다. 익스포트 유틸리티가 Embedded SQL 응용프로그램이고 내부적으로 SQL 페치를 수행하는 경우, SQL 조작에 적용되는 최적화는 익스포트 유틸리티에도 적용됩니다. 용량이 큰 버퍼 풀, 인

덱싱, 및 정렬 힙을 활용하는 방법을 고려하십시오. 또한 출력 파일을 컨테이너 및 로그 디바이스 외부에 저장하여 출력 파일에서 디바이스 경쟁을 최소화하십시오.

메시지 파일
익스포트 유틸리티는 표준 ASCII 텍스트 메시지 파일로 오류, 경고 및 정보 메시지를 작성합니다. CLP를 제외한 모든 인터페이스에서 **MESSAGES** 매개변수를 사용하여 미리 이 파일의 이름을 지정해야 합니다. CLP를 사용하는 경우 메시지 파일을 지정하지 않으면 익스포트 유틸리티는 표준 출력으로 메시지를 작성합니다.

익스포트 유틸리티를 사용하는 데 필요한 특권 및 권한

특권이 있으면 데이터베이스 자원을 작성, 갱신, 삭제 또는 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작에 특권을 맵핑하는 방법을 제공합니다.

특권과 권한은 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다. 사용자는 적절한 권한(즉, 필요한 특권 또는 권한)을 가지고 있는 오브젝트에만 액세스할 수 있습니다.

익스포트 조작에 참여하는 각 테이블 또는 뷰에 대한 DATAACCESS 권한이나 CONTROL 또는 SELECT 특권이 있어야 합니다.

LBAC 보호 데이터를 익스포트하는 경우 세션 권한 부여 ID에서 익스포트하려는 행 또는 컬럼을 읽을 수 있어야 합니다. 세션 권한 부여 ID에 읽기 권한이 부여되지 않은 보호된 행은 익스포트되지 않습니다. 세션 권한 부여 ID에서 읽을 수 없는 보호된 컬럼이 SELECT문에 포함된 경우 익스포트 유틸리티는 실패하고 오류(SQLSTATE 42512)가 리턴됩니다.

데이터 익스포트

익스포트 유틸리티를 사용하여 데이터베이스에서 파일로 데이터를 익스포트합니다. 파일은 여러 외부 파일 형식 중 하나일 수 있습니다. SQL SELECT문을 제공하거나 유형이 지정된 테이블의 계층 정보를 제공하여 익스포트할 데이터를 지정할 수 있습니다.

데이터베이스에서 데이터를 익스포트하려면 참여하는 각 테이블 또는 뷰에 대한 DATAACCESS 권한, CONTROL 특권 또는 SELECT 특권이 필요합니다.

익스포트 유틸리티를 실행하기 전에 데이터를 익스포트할 데이터베이스에 연결하거나 내재적으로 연결할 수 있어야 합니다. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다. Linux, UNIX 또는 Windows 클라이언트에서 Linux, UNIX 또는 Windows 데이터베이스 서버로의 유틸리티 액세스는 DB2 Connect 게이트웨이 또는 루프백 환경이 아닌 엔진을 통해 직접 연결되어야 합니다.

유틸리티는 COMMIT문을 실행하므로 익스포트 유틸리티를 실행하기 전에 COMMIT 또는 ROLLBACK문을 실행하여 모든 트랜잭션을 완료하고 잠금을 해제해야 합니다. 테이블에 액세스하고 별도의 연결을 사용하는 응용프로그램에서 연결을 끊지 않아도 됩니다.

구조화된 유형이 지정된 컬럼으로 테이블을 익스포트할 수 없습니다.

제어 센터의 테이블 익스포트 노트북 또는 API db2Export를 사용하거나 명령행 처리기(CLP)에서 EXPORT 명령을 지정하여 익스포트 유틸리티를 실행할 수 있습니다.

테이블 익스포트 노트북 사용

테이블 익스포트 노트북을 사용하여 데이터를 익스포트하려면 다음을 수행하십시오.

1. 제어 센터에서 테이블 또는 뷰 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 작업할 폴더를 누르십시오. 기존 테이블 또는 뷰가 창 오른쪽의 분할 영역(컨텐츠 영역)에 표시됩니다.
3. 컨텐츠 영역에서 원하는 테이블 또는 뷰를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 익스포트를 선택하십시오. 테이블 익스포트 노트북이 열립니다.

테이블 익스포트 노트북에 대한 세부사항 정보는 제어 센터 온라인 도움말 기능에서 제공됩니다.

CLP를 사용하여 EXPORT 명령 실행

단순하게 익스포트 조작을 수행하려면 SELECT문에서 목표 파일, 파일 형식 및 소스 파일만 지정하면 됩니다.

CLP에서 데이터를 익스포트하려면 EXPORT 명령을 입력하십시오.

```
db2 export to filename of ixf select * from table
```

여기서 `filename`은 작성 및 익스포트할 출력 파일 이름이고 `ixf`는 파일 형식이며 `table`은 복사할 데이터를 포함하는 테이블 이름입니다.

그러나 경고 및 오류 메시지를 작성할 메시지 파일도 지정할 수 있습니다. 이를 수행하려면 **MESSAGES** 매개변수 및 메시지 파일 이름(이 경우, `msg.txt`)을 추가합니다. 명령은 다음과 같습니다.

```
db2 export to filename of ixf messages msgs.txt select * from table
```

전체 구문 및 사용법 정보는 ″EXPORT″ 명령을 참조하십시오.

XML 데이터 익스포트

XML 데이터 익스포트 시, 결과 XQuery 데이터 모델(QDM) 인스턴스는 익스포트된 관계형 데이터를 포함하는 주 데이터 파일과 다른 파일에 저장됩니다. XMLFILE 또는

XML TO 옵션이 둘 다 지정되지 않은 경우에도 그렇습니다. 디폴트로 익스포트된 QDM 인스턴스는 모두 동일한 XML 파일에 병합됩니다. XMLINSEPFILES 파일 유형 수정자를 사용하여 각 QDM 인스턴스를 개별 파일에 저장하도록 지정할 수 있습니다.

그러나 XML 데이터는 주 데이터 파일에서 XML 데이터 지정자(XDS)로 표시됩니다. XDS는 "XDS"라는 XML 태그로 표시되는 문자열입니다. 이 태그에는 컬럼의 실제 XML 데이터에 대한 정보를 설명하는 속성이 있습니다. 이러한 정보에는 실제 XML 데이터가 포함된 파일 이름 및 해당 파일에 있는 XML 데이터의 오프셋 및 길이가 포함됩니다.

XML TO 및 XMLFILE 옵션을 사용하여 익스포트된 XML 파일의 대상 경로와 기본 이름을 지정할 수 있습니다. XML TO 또는 XMLFILE 옵션을 지정하는 경우 XDS의 FIL 속성에 저장된 익스포트된 XML 파일 이름의 형식은 xmlfilespec.xxx.xml입니다. 여기서 xmlfilespec은 XMLFILE 옵션에 지정되는 값이고 xxx 는 익스포트 유틸리티에서 생성하는 xml 파일의 시퀀스 번호입니다. 옵션을 지정하지 않는 경우 익스포트된 XML 파일 이름의 형식은 exportfilename.xxx.xml입니다. 여기서 exportfilename은 EXPORT 명령에 지정되는 익스포트된 출력 파일의 이름이고 xxx는 익스포트 유틸리티에서 생성하는 xml 파일의 시퀀스 번호입니다.

디폴트로, 익스포트된 XML 파일은 익스포트된 데이터 파일의 경로에 기록됩니다. 익스포트된 XML 파일의 디폴트 기본 이름은 익스포트된 데이터 파일의 이름으로 3자리 시퀀스 번호가 추가되고 확장자는 .xml입니다.

예

다음 예에서는 4개의 컬럼과 2개의 행이 포함된 USER.T1 테이블을 가정합니다.

```
C1 INTEGER
C2 XML
C3 VARCHAR(10)
C4 XML
```

표 2. USER.T1

C1	C2	C3	C4
2	<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to><from> Me</from><heading>note1</heading> <body>Hello World!</body></note>	'char1'	<?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him</to><from> Her</from><heading>note2</heading><body>Hello World!</body></note>
4	NULL	'char2'	?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">to>Us</to><from> Them</from><heading>note3</heading><body>Hello World!</body></note>

예 1

다음 명령은 컬럼 식별자가 있는 ASCII(DEL) 형식의 USER.T1의 컨텐츠를 ″/mypath/t1export.del″ 파일에 익스포트합니다. XML TO 및 XMLFILE 옵션이 지정되지 않았기 때문에 컬럼 C2와 C4에 포함된 문서는 익스포트된 주 파일 ″/mypath″와 동일한 경로에 저장됩니다. 이러한 파일의 기본 이름은 ″t1export.del.xml″입니다. XMLSAVESCHEMA 옵션은 익스포트 프로시저 중에 XML 스키마 정보가 저장됨을 나타냅니다.

```
EXPORT TO /mypath/t1export.del OF DEL XMLSAVESCHEMA SELECT * FROM USER.T1
```

익스포트된 파일 ″/mypath/t1export.del″에는 다음이 포함됩니다.

```
2,"<XDS FIL='t1export.del.001.xml' OFF='0' LEN='144' />","char1",
   "<XDS FIL='t1export.del.001.xml' OFF='144' LEN='145' />"
   4,,"char2","<XDS FIL='t1export.del.001.xml' OFF='289'
   LEN='145' SCH='S1.SCHEMA_A' />"
```

익스포트된 XML 파일 ″/mypath/t1export.del.001.xml″에는 다음이 포함됩니다.

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
      <from>Me</from><heading>note1</heading><body>Hello World!</body>
      </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him
      </to><from>Her</from><heading>note2</heading><body>Hello World!
      </body></note><?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">
      <to>Us</to><from>Them</from>heading>note3</heading><body>
      Hello World!</body></note>
```

예 2

다음 명령은 DEL 형식의 USER.T1의 컨텐츠를 ″t1export.del″ 파일에 익스포트합니다. 컬럼 C2와 C4에 포함된 XML 문서는 경로 ″/home/user/xmlpath″에 저장됩니다. XML 파일 이름은 기본 이름 ″xmldocs″를 사용하여 지정되고 다중 익스포트된 XML 문서는 동일한 XML 파일에 저장됩니다. XMLSAVESCHEMA 옵션은 익스포트 프로시저 중에 XML 스키마 정보가 저장됨을 나타냅니다.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
   XMLFILE xmldocs XMLSAVESCHEMA SELECT * FROM USER.T1
```

익스포트된 DEL 파일 ″/home/user/t1export.del″에는 다음이 포함됩니다.

```
   2,"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='144' />","char1",
   "<XDS FIL='xmldocs.001.xml' OFF='144' LEN='145' />"
   4,,"char2","<XDS FIL='xmldocs.001.xml' OFF='289'
   LEN='145' SCH='S1.SCHEMA_A' />"
```

익스포트된 XML 파일 ″/home/user/xmlpath/xmldocs.001.xml″에는 다음이 포함됩니다.

```
   <?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
      <from>Me</from><heading>note1</heading><body>Hello World!</body>
      </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00">
      <to>Him</to><from>Her</from><heading>note2</heading><body>
```

```
Hello World!</body></note><?xml version="1.0" encoding="UTF-8" ?>
<note time="14:00:00"><to>Us</to><from>Them</from><heading>
note3</heading><body>Hello World!</body></note>
```

예 3

다음 명령은 예 2와 비슷하고 각각의 익스포트된 XML 문서가 개별 XML 파일에 저장되는 점만 다릅니다.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
  XMLFILE xmldocs MODIFIED BY XMLINSEPFILES XMLSAVESCHEMA
  SELECT * FROM USER.T1
```

익스포트된 파일 "/mypath/t1export.del"에는 다음이 포함됩니다.

```
2,"<XDS FIL='xmldocs.001.xml' />","char1","XDS FIL='xmldocs.002.xml' />"
4,,"char2","<XDS FIL='xmldocs.004.xml' SCH='S1.SCHEMA_A' />"
```

익스포트된 XML 파일 "/home/user/xmlpath/xmldocs.001.xml"에는 다음이 포함됩니다.

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
      <from>Me</from><heading>note1</heading><body>Hello World!</body>
      </note>
```

익스포트된 XML 파일 "/home/user/xmlpath/xmldocs.002.xml"에는 다음이 포함됩니다.

```
?xml version="1.0" encoding="UTF-8" ?>note time="13:00:00">to>Him/to>
   from>Her/from>heading>note2/heading>body>Hello World!/body>
   /note>
```

익스포트된 XML 파일 "/home/user/xmlpath/xmldocs.004.xml"에는 다음이 포함됩니다.

```
<?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00"><to>Us</to>
   <from>Them</from><heading>note3</heading><body>Hello World!</body>
   </note>
```

예 4

다음 명령은 XQuery의 결과를 XML 파일에 기록합니다.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
   XMLFILE xmldocs MODIFIED BY XMLNODECLARATION select
   xmlquery( '$m/note/from/text()' passing by ref c4 as "m" returning sequence)
      from USER.T1
```

익스포트된 DEL 파일 "/mypath/t1export.del"에는 다음이 포함됩니다.

```
"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='3' />"
"<XDS FIL='xmldocs.001.xml' OFF='3' LEN='4' />"
```

익스포트된 XML 파일 "/home/user/xmlpath/xmldocs.001.xml"에는 다음이 포함됩니다.

```
HerThem
```

주: 이러한 특정 XQuery의 결과는 잘 구성된 XML 문서를 생성하지 않습니다. 따라서 위의 익스포트된 파일을 직접 XML 컬럼에 임포트할 수 없습니다.

LBAC 보호 데이터 익스포트 고려사항

레이블 기반 액세스 제어(LBAC)로 보호되는 데이터를 익스포트하는 경우 익스포트되는 데이터는 LBAC 증명서에서 읽도록 허용한 데이터로 제한됩니다.

LBAC 증명서에서 행을 읽도록 허용하지 않는 경우 해당 행은 익스포트되지 않지만 오류는 리턴되지 않습니다. LBAC 증명서에서 컬럼을 읽도록 허용하지 않은 경우 익스포트 유틸리티에 실패하고 오류(SQLSTATE 42512)가 리턴됩니다.

데이터 유형이 DB2SECURITYLABEL인 컬럼의 값은 문자 분리문자로 묶인 원시 데이터로 익스포트됩니다. 원래 데이터에 문자 분리문자가 포함된 경우 분리문자를 이중으로 사용합니다. 익스포트된 값을 구성하는 바이트에 대해서 다른 사항은 변경되지 않습니다. 즉, DB2SECURITYLABEL 데이터를 포함하는 데이터 파일은 개행, 폼피드 또는 기타 인쇄 불가능한 ASCII 문자를 포함할 수 있습니다.

데이터 유형이 DB2SECURITYLABEL인 컬럼 값을 판독 가능한 양식으로 익스포트하려는 경우 SELECT문에서 SECLABEL_TO_CHAR 스칼라 함수를 사용하여 보안 레이블 문자열 형식으로 값을 변환할 수 있습니다.

예

다음 예에서 출력은 DEL 형식이며 myfile.del 파일에 작성됩니다. 다음 명령문으로 작성한 REPS 테이블에서 데이터가 익스포트됩니다.

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

이 예에서는 디폴트 형식으로 row_label 컬럼 값을 익스포트합니다.

```
db2 export to myfile.del of del select * from reps
```

대부분의 텍스트 편집기에서 데이터 파일은 판독이 어렵습니다. row_label 컬럼 값이 여러 ASCII 제어 문자를 포함할 수 있기 때문입니다.

다음 예에서는 보안 레이블 문자열 형식으로 row_label 컬럼 값을 익스포트합니다.

```
db2 export to myfile.del of del select SECLABEL_TO_CHAR
(row_label,'DATA_ACCESS_POLICY'), id, name from reps
```

다음은 이전 예에서 작성한 데이터 파일의 한 부분입니다. 보안 레이블 형식이 판독 가능하다는 점에 주의하십시오.

```
...
"Secret:():Epsilon 37", 2005, "Susan Liu"
"Secret:():(Epsilon 37,Megaphone,Cloverleaf)", 2006, "Johnny Cogent"
"Secret:():(Megaphone,Cloverleaf)", 2007, "Ron Imron"
...
```

테이블 익스포트 고려사항

일반 익스포트 조작은 기존 테이블에 삽입 또는 로드되는 선택된 데이터 출력과 관련이 있습니다. 그러나 임포트 유틸리티를 사용하여 후속 재작성을 위해 전체 테이블을 익스포트할 수도 있습니다.

테이블을 익스포트하려면 PC/IXF 파일 형식을 지정해야 합니다. CREATE 모드에서 임포트 유틸리티를 사용하여 저장한 테이블 및 해당 인덱스를 재작성할 수 있습니다. 그러나 다음 조건인 경우 일부 정보는 익스포트된 IXF 파일에 저장되지 않습니다.

- 인덱스 컬럼 이름에 16진수 값 0x2B 또는 0x2D가 있는 경우
- 테이블에 XML 컬럼이 있는 경우
- 테이블이 다차원 클러스터된(MDC) 테이블인 경우
- 테이블에 테이블 파티셔닝 키가 있는 경우
- 코드 페이지 변환 때문에 인덱스 이름이 128바이트보다 긴 경우
- 테이블이 보호된 경우
- EXPORT 명령에 SELECT * FROM *tablename* 이외의 조치 문자열이 있는 경우
- 익스포트 유틸리티에 대해 **METHOD N** 매개변수를 지정하는 경우

손실된 테이블 속성의 목록에 대해서는 "테이블 임포트 고려사항"을 참조하십시오. 저장되지 않은 정보가 있으면 정보가 테이블을 재작성할 때 경고 SQL27984W가 리턴됩니다.

주: 임포트의 CREATE 모드는 사용되지 않습니다. db2look 유틸리티를 사용하여 테이블을 캡처 및 재작성합니다.

인덱스 정보

인덱스에 지정된 컬럼 이름에 - 또는 + 문자가 있는 경우 인덱스 정보는 수집되지 않으며 경고 SQL27984W가 리턴됩니다. 익스포트 유틸리티에서 처리를 완료하며, 익스포트된 데이터는 영향을 받지 않습니다. 그러나 인덱스 정보는 IXF 파일에 저장되지 않습니다. 결과적으로 db2look 유틸리티를 사용하여 인덱스를 별도로 작성해야 합니다.

공간 제한사항

익스포트된 파일이 작성된 파일 시스템에서 사용 가능한 공간보다 익스포트한 데이터가 더 큰 경우 익스포트 조작에 실패합니다. 이 경우 익스포트된 파일이 목표 파일 시스템에 맞도록 WHERE절에서 조건을 지정하여 선택된 데이터 크기를 제한해야 합니다. 익스포트 유틸리티를 여러 번 실행하여 모든 데이터를 익스포트할 수 있습니다.

기타 파일 형식의 테이블

IXF 파일 형식을 사용하여 익스포트하지 않는 경우 출력 파일에는 목표 테이블에 대한 설명이 포함되지 않지만 레코드 데이터는 포함됩니다. 테이블 및 해당 데이터를 재작성하려면 목표 테이블을 작성하고 로드 또는 임포트 유틸리티를 사용하여 테이블을 채우십시오. db2look 유틸리티를 사용하여 원래 테이블 정의를 캡처하고 해당 데이터 정의 언어(DDL)를 생성할 수 있습니다.

유형이 지정된 테이블 익스포트 고려사항

DB2 익스포트 유틸리티를 사용하여 나중에 임포트하기 위해 유형이 지정된 테이블에서 데이터를 이동할 수 있습니다. 익스포트는 특정 순서에 따라 중간 플랫 파일을 작성하면서 유형이 지정된 테이블의 한 계층 구조에서 다른 계층 구조로 데이터를 이동합니다.

유형이 지정된 테이블에 대한 작업을 하는 경우 익스포트 유틸리티는 출력 파일에 배치되는 내용을 제어합니다. 이때 목표 테이블 이름 및 선택적으로 WHERE만 지정합니다. 목표 테이블 이름 및 WHERE절만 지정하여 subselect문을 표현할 수 있습니다. 계층 구조를 익스포트할 때 fullselect 또는 select문은 지정할 수 없습니다.

트래버스 순서를 사용하여 계층 구조 보존

유형이 지정된 테이블은 계층 구조일 수 있습니다. 여러 계층 구조에서 데이터를 이동하는 여러 가지 방법이 있습니다.

- 한 계층 구조에서 동일한 계층 구조로 이동
- 한 계층 구조에서 더 큰 계층 구조의 서브섹션으로 이동
- 큰 계층 구조의 서브섹션에서 별도의 계층 구조로 이동

계층 구조에서 유형 ID는 데이터베이스에 종속되어 있습니다. 즉, 다른 데이터베이스인 경우 동일한 유형이어도 ID가 다릅니다. 따라서 이러한 데이터베이스에서 데이터를 이동하는 경우 데이터를 올바르게 이동하려면 동일한 유형을 맵핑해야 합니다.

유형이 지정된 테이블에서 사용하는 맵핑은 *트래버스 순서*라고 합니다. 이는 계층 구조의 모든 슈퍼 테이블 및 서브테이블에 맨 위에서 맨 아래로, 왼쪽에서 오른쪽으로 진행되는 순서를 말합니다. 익스포트 조작 중 각 유형이 지정된 행이 기록되면 ID는 인덱스 값으로 변환됩니다. 이 인덱스 값은 1부터 계층 구조에서 관련 유형의 수에 이르기까지 모든 수가 가능합니다. 인덱스 값은 계층 구조에서 특정 순서(트래버스 순서)로 이동할 때 각 유형에 번호를 지정하여 생성됩니다. 그림 1에서는 네 가지 올바른 트래버스 순서로 계층 구조를 표시합니다.

- Person, Employee, Manager, Architect, Student
- Person, Student, Employee, Manager, Architect
- Person, Employee, Architect, Manager, Student
- Person, Student, Employee, Architect, Manager

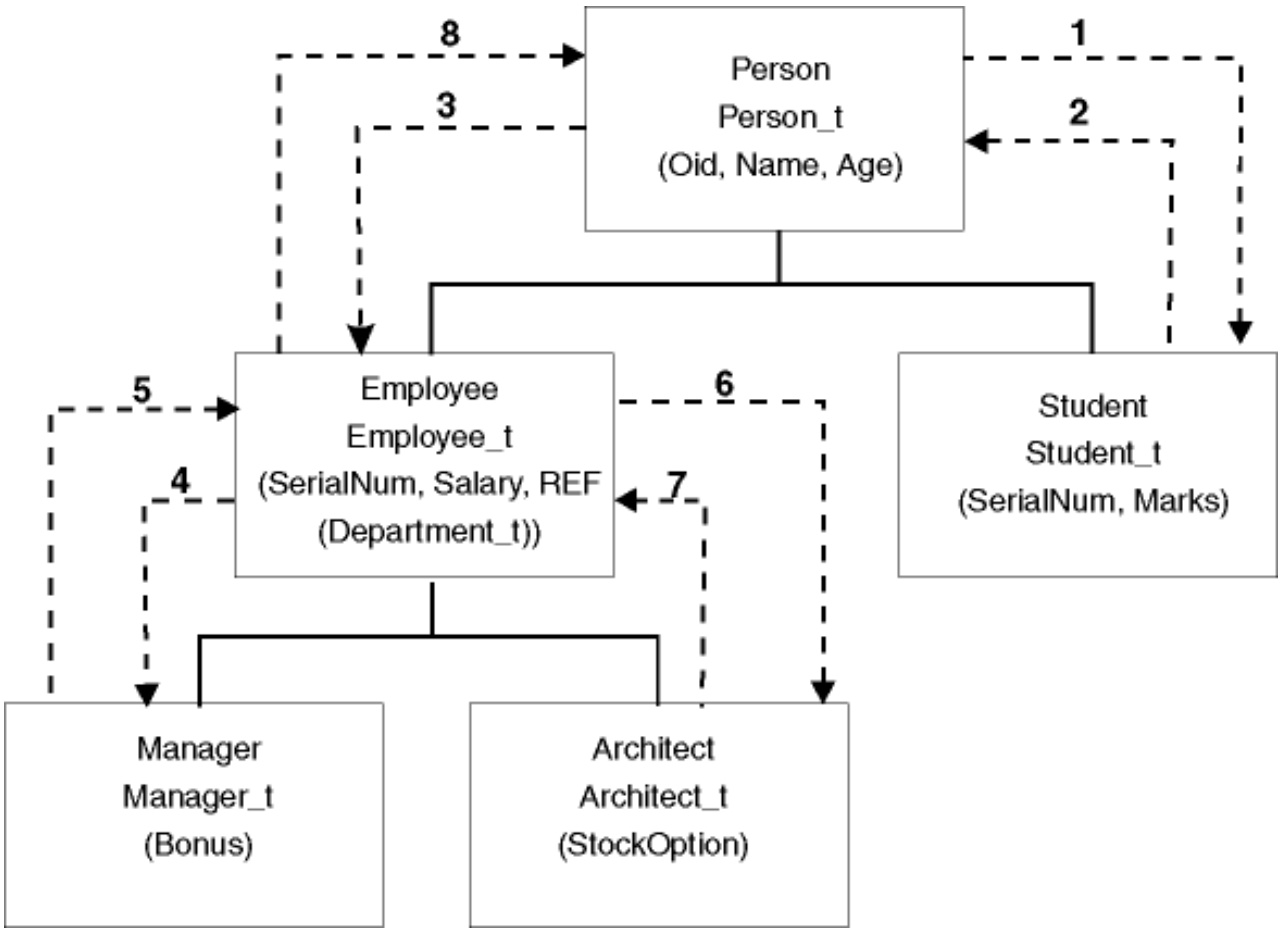


그림 1. 계층 구조 예

트래버스 순서는 테이블 계층에서 데이터를 이동할 때 매우 중요합니다. 데이터가 다른 데이터를 고려하여 이동되는 위치를 판별하기 때문입니다. 트래버스 순서에는 *디폴트* 및 *사용자 지정*과 같은 두 가지 유형이 있습니다.

디폴트 트래버스 순서

디폴트 트래버스 순서를 사용하면 모든 관련 유형이 계층 구조의 지정된 시작 지점부터 계층 구조의 도달 가능한 모든 유형을 참조합니다. 디폴트 순서에는 계층 구조의 모든 테이블이 포함되며 각 테이블은 OUTER 순서 술어에서 사용하는 스킴으로 정렬됩니다. 예를 들어 그림 1의 디폴트 트래버스 순서(점선으로 표시됨)는 Person, Student, Employee, Manager, Architect입니다.

다른 파일 형식에서 사용하면 디폴트 트래버스 순서가 조금 달라집니다. PC/IXF 파일 형식으로 데이터를 익스포트하면 모든 관련 유형의 레코드, 해당 정의 및 관련 테이블이 작성됩니다. 또한 익스포트 유틸리티는 인덱스 값을 각 테이블에 맵핑하는 작업도 완료합니다. PC/IXF 파일 형식에 대한 작업을 수행하는 경우 디폴트 트래버스 순서를 사용해야 합니다.

ASC, DEL 또는 WSF 파일 형식인 경우 소스 및 목표 계층 구조가 구조적으로 동일해도 유형이 지정된 행 및 유형이 지정된 테이블이 작성된 순서는 서로 다를 수 있습니다. 결과적으로 계층 구조를 진행할 때 디폴트 트래버스 순서가 식별하는 시간의 차이가 발생합니다. 각 유형 작성 시 디폴트 트래버스 순서를 사용할 때 소스 및 목표 모두에서 계층 구조 사이를 이동하는 데 사용된 순서를 판별합니다. 소스 및 목표 계층 구조에서 각 유형의 작성 순서가 동일하고 소스 및 목표 사이에 구조 ID가 있는지 확인합니다. 이 조건을 충족할 수 없으면 사용자 지정 트래버스 순서를 선택합니다.

사용자 지정 트래버스 순서
사용자 지정 트래버스 순서에서는 사용할 관련 유형을 트래버스 순서 목록에서 정의합니다. 이 순서에서는 계층 구조를 트래버스하는 방법과 익스포트할 서브테이블을 간략히 설명합니다. 반면, 디폴트 트래버스 순서에서는 계층 구조의 모든 테이블이 익스포트됩니다.

트래버스 순서를 정의할 때 계층 구조 아래 경로 및 시작 지점을 판별해도 서브테이블은 *이전 순서*대로 트래버스됩니다. 새 분기를 시작하려면 계층 구조의 각 분기를 맨 아래로 트래버스해야 합니다. 익스포트 유틸리티는 지정된 트래버스 순서로 이 조건 위반을 검색합니다. 조건을 만족하는 한 가지 방법은 계층 구조 맨 위(또는 루트 테이블)에서 계층 구조 아래로 내려가(서브테이블) 맨 아래 서브테이블까지 진행한 다음, 해당 슈퍼 테이블로 다시 올라갔다가 "맨 오른쪽" 서브테이블로 내려간 다음, 다음 상위에 해당하는 슈퍼 테이블로 다시 올라갔다가 해당 서브테이블로 내려오는 식으로 진행하면 됩니다.

계층 구조에서 트래버스 순서를 제어하려면 익스포트 및 임포트 유틸리티에서 동일한 트래버스 순서를 사용해야 합니다.

예 1

다음 예는 그림 1의 계층 구조에 기반합니다. 전체 계층 구조를 익스포트하려면 다음 명령을 입력하십시오.

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO entire_hierarchy.ixf OF IXF HIERARCHY STARTING Person
```

HIERARCHY STARTING 매개변수를 Person으로 설정하면 테이블 PERSON부터 디폴트 트래버스 순서가 시작됨을 의미합니다.

예 2

20세 이상인 사람의 데이터만 대상으로 전체 계층 구조를 익스포트하려면 다음 명령을 입력합니다.

```
DB2 CONNECT TO Source_db
   DB2 EXPORT TO entire_hierarchy.del OF DEL HIERARCHY (Person,
      Employee, Manager, Architect, Student) WHERE Age>=20
```

HIERARCHY를 Person, Employee, Manager, Architect, Student로 설정하면 사용자 지정 트래버스 순서를 나타냅니다.

ID 컬럼 익스포트 고려사항

익스포트 유틸리티를 사용하여 ID 컬럼을 포함하는 테이블에서 데이터를 익스포트할 수 있습니다. 그러나 ID 컬럼은 출력 파일 형식의 선택사항을 제한합니다.

익스포트 조작에 대해 지정된 SELECT문이 SELECT * FROM *tablename* 양식이고 METHOD 옵션을 사용하지 않은 경우 ID 컬럼 등록 정보를 IXF 파일로 익스포트하는 작업이 지원됩니다. 그러면 IMPORT 명령의 REPLACE_CREATE 및 CREATE 옵션을 사용하여 해당 ID 컬럼 등록 정보를 포함하는 테이블을 다시 작성할 수 있습니다. 유형이 GENERATED ALWAYS인 ID 컬럼을 포함하는 테이블에서 익스포트된 IXF 파일을 작성하는 경우 데이터 파일을 성공적으로 임포트하는 유일한 방법은 임포트 조작 중 `identityignore` 파일 유형 수정자를 지정하는 것입니다. 그렇지 않으면 모든 행이 거부됩니다(SQL3550W가 발행됨).

주: IMPORT 명령의 CREATE 및 REPLACE_CREATE 옵션은 사용되지 않으며 추후 릴리스에서 제거됩니다.

LOB 익스포트 고려사항

대형 오브젝트(LOB) 컬럼을 포함하는 테이블을 익스포트할 때 디폴트 조치는 LOB 값당 최대 32KB를 익스포트하고 나머지 컬럼 데이터와 동일한 파일에 배치하는 것입니다. 32KB를 초과하는 LOB 값을 익스포트하는 경우 절단되지 않도록 하려면 별도의 파일에 LOB 데이터를 작성해야 합니다.

고유한 파일에 LOB를 작성하도록 지정하려면 `lobsinfile` 파일 유형 수정자를 사용합니다. 이 수정자에서는 LOBS TO절에 지정된 디렉토리에 LOB 데이터를 배치하도록 익스포트 유틸리티에 명령합니다. LOBS TO 또는 LOBFILE을 사용하면 명시적으로 `lobsinfile` 파일 유형 수정자가 활성화됩니다. 디폴트로 LOB 값은 익스포트된 관계형 데이터가 작성된 동일한 경로에 작성됩니다. LOBS TO 옵션에 하나 이상의 경로가 지정된 경우 익스포트 유틸리티는 적절한 LOB 파일에 LOB 값을 성공적으로 작성하도록 여러 경로 사이를 순환합니다. LOBFILE 옵션을 사용하여 출력 LOB 파일의 이름을 지정할 수도 있습니다. LOBFILE 옵션이 지정되면 lobfilename 형식은 `lobfilespec.xxx.lob`입니다. 여기서 `lobfilespec`은 LOBFILE 옵션에 지정된 값이고 `xxx`은 익스포트 유틸리티에서 생성된 LOB 파일의 시퀀스 번호입니다. 그렇지 않으면 lobfilename 형식은 `exportfilename.xxx.lob`입니다. 여기서 `exportfilename`은 EXPORT 명령에 지정된 익스포트할 파일 이름이고 `xxx`는 익스포트 유틸리티에서 생성된 LOB 파일의 시퀀스 번호입니다.

디폴트로 LOB는 하나의 파일에 작성되지만 개별 LOB가 별도의 파일에 저장되도록 지정할 수도 있습니다. 익스포트 유틸리티는 한 파일에 여러 LOB를 저장할 수 있도록

LLS(LOB Location Specifier)를 생성합니다. 익스포트 출력 파일에 작성되는 LLS는 파일 내 LOB 데이터가 저장된 위치를 나타내는 문자열입니다. LLS의 형식은 다음과 같습니다. lobfilename.ext.nnn.mmm/. 여기서 lobfilename.ext는 LOB를 포함하는 파일 이름이고 nnn은 파일 내 LOB의 오프셋(바이트)이며 mmm은 LOB의 길이(바이트)입니다. 예를 들어 LLS가 db2exp.001.123.456/이면 LOB가 db2exp.001 파일에 있으면 파일의 오프셋 123바이트에서 시작하고 길이는 456바이트임을 나타냅니다. LLS에 표시된 크기가 0이면 LOB 길이는 0으로 간주됩니다. 길이가 -1이면 LOB는 널(null)로 간주되고 오프셋 및 파일 이름은 무시됩니다.

동일한 파일에 개별 LOB 데이터를 연결하지 않으려면 lobsinsepfiles 파일 유형 수정자를 사용하여 각 LOB를 별도의 파일에 작성합니다.

주: IXF 파일 형식은 LOB 컬럼 로깅 여부와 같은 컬럼의 LOB 옵션을 저장하지 않습니다. 즉, 임포트 유틸리티는 1GB 이상으로 정의된 LOB 컬럼을 포함하는 테이블을 재작성할 수 없습니다.

예 1

다음 예에서는 LOB(익스포트된 LOB 파일의 지정된 기본 이름은 lobs1임)를 DEL 파일로 익스포트하는 방법을 보여줍니다.

```
db2 export to myfile.del of del lobs to mylobs/
   lobfile lobs1 modified by lobsinfile
   select * from emp_photo
```

예 2

다음 예에서는 각 LOB 값이 별도의 파일에 작성되고 로그 파일이 두 개의 디렉토리에 작성되는 LOB를 DEL 파일로 익스포트하는 방법을 보여줍니다.

```
db2 export to myfile.del of del
lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
select * from emp_photo
```

참조 - 익스포트

EXPORT

데이터베이스의 데이터를 여러 외부 파일 형식 중 하나로 익스포트합니다. 사용자는 SQL SELECT문을 제공하거나 유형이 지정된 테이블에 대한 계층 정보를 제공하여 익스포트할 데이터를 지정합니다.

25 페이지의 『익스포트 유틸리티의 파일 유형 수정자』로의 빠른 링크를 참조하십시오.

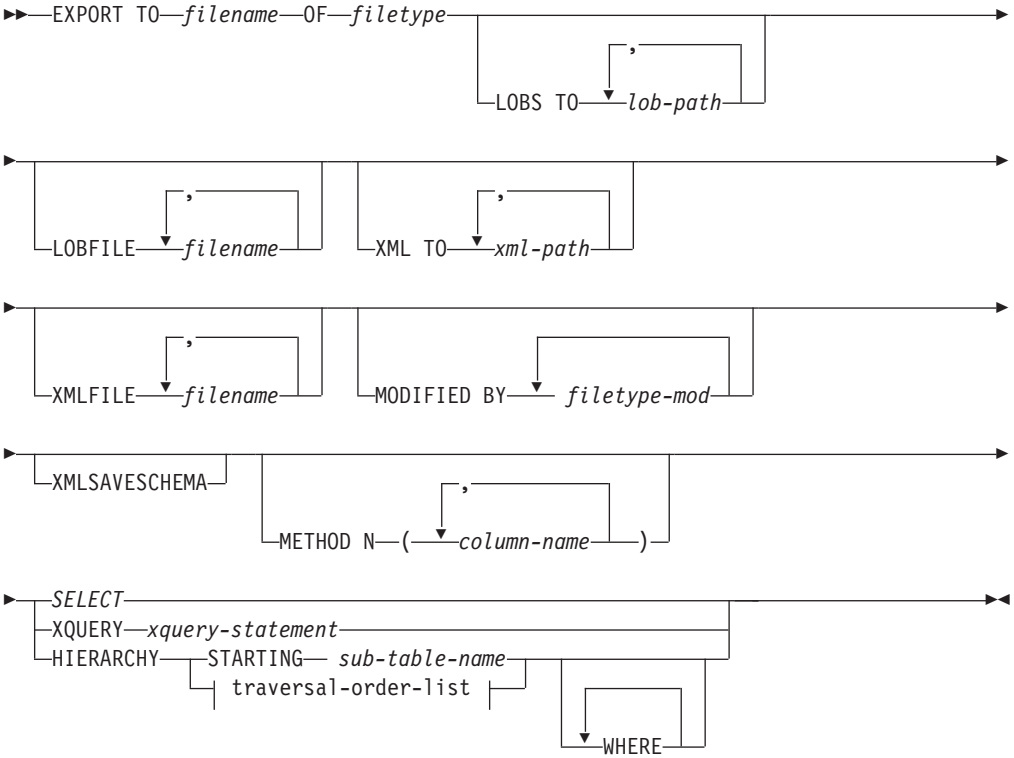
권한 부여

다음 중 하나가 필요합니다.

- *dataaccess* 권한
- 참여 중인 각 테이블 또는 뷰에 대한 CONTROL 또는 SELECT 특권

필수 연결

명령 구문



traversal-order-list:

명령 매개변수

HIERARCHY *traversal-order-list*

지정된 트래버스 순서를 사용하여 하위 계층 구조를 익스포트합니다. 모든 서브테이블은 PRE-ORDER 형식으로 나열되어야 합니다. 첫 번째 서브테이블 이름을 SELECT문의 목표 테이블 이름으로 사용합니다.

HIERARCHY STARTING *sub-table-name*

디폴트 트래버스 순서(ASC, DEL 또는 WSF 파일의 경우 OUTER 순서 또

는 PC/IXF 데이터 파일에 저장된 순서)를 사용하여 *sub-table-name*에서 시작하는 하위 계층 구조를 익스포트하십시오.

LOBFILE *filename*

LOB 파일에 대한 하나 이상의 기본 파일을 지정합니다. 첫 번째 이름에 대한 이름 공간이 모두 사용되었으면 두 번째 이름을 사용하고 계속 이와 같이 합니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

익스포트 조작 중에 LOB 파일을 작성할 경우, 파일 이름은 해당 목록의 현재 기본 이름을 현 재 경로(*lob-path*의)에 추가한 후 시작할 3자리 시퀀스 번호 및 3자의 ID(lob)를 추가하여 구성됩니다. 예를 들어, 현재 LOB 경로가 /u/foo/lob/path/ 디렉토리이고 현재 LOB 파일 이름이 bar이면 작성되는 LOB 파일은 /u/foo/lob/path/bar.001.lob, /u/foo/lob/path/bar.002.lob 등입니다. 999가 사용되고 나면 LOB 파일 이름의 3자리 시퀀스 번호는 4자리가 되며, 9999가 사용되고 나면 4자리는 5자리가 되는 식으로 진행됩니다.

LOBS TO *lob-path*

LOB 파일이 저장될 디렉토리에 대한 하나 이상의 경로를 지정하십시오. LOB 경로 당 최소한 하나의 파일이 있으며 각 파일에는 최소한 하나의 LOB가 있습니다. 지정할 수 있는 최대 경로 수는 999입니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

METHOD N *column-name*

출력 파일에서 사용될 하나 이상의 컬럼 이름을 지정합니다. 이 매개변수를 지정하지 않으면 테이블에 있는 컬럼 이름이 사용됩니다. 이 매개변수는 WSF 및 IXF 파일에 대해서만 유효하며 계층 데이터를 익스포트할 때는 유효하지 않습니다.

MODIFIED BY *filetype-mod*

파일 유형 수정자 옵션을 지정합니다. 25 페이지의 『익스포트 유틸리티의 파일 유형 수정자』를 참조하십시오.

OF *filetype*

출력 파일에 있는 데이터의 형식을 지정합니다.

- DEL(컬럼 식별자가 있는 ASCII 형식) - 여러 데이터베이스 관리 프로그램 및 파일 관리자 프로그램에서 사용됩니다.
- WSF(작업시트 형식) - 다음과 같은 프로그램에서 사용됩니다.
  - Lotus® 1-2-3®
  - Lotus Symphony

  BIGINT 또는 DECIMAL 데이터를 익스포트할 때 DOUBLE 유형의 범위에 해당하는 값만 정확히 익스포트할 수 있습니다. 이 범위에 해당하지 않

는 값은 익스포트되기는 하지만 이들 값을 다시 임포트하거나 로드할 때 운영 체제에 따라 올바르지 않은 데이터가 나타날 수 있습니다.

주: WSF 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

- IXF(Integration Exchange Format, PC 버전)는 독점 2진 형식입니다.

SELECT

익스포트될 데이터를 리턴하는 SELECT 또는 XQUERY문을 지정합니다. 명령문이 오류를 일으킬 경우 메시지 파일(또는 표준 출력)에 메시지가 쓰여집니다. 오류 코드가 SQL0012W, SQL0347W, SQL0360W, SQL0437W 또는 SQL1824W 중 하나이면 익스포트 조작을 계속 수행하고 그렇지 않으면 중지합니다.

TO *filename*

이미 존재하는 파일 이름을 지정할 경우 익스포트 유틸리티는 파일 내용을 겹쳐쓰고 정보를 추가하지 않습니다.

XMLFILE *filename*

XML 파일에 대한 하나 이상의 기본 파일 이름을 지정합니다. 첫 번째 이름에 대한 이름 공간이 모두 사용되었으면 두 번째 이름을 사용하고 계속 이와 같이 합니다.

익스포트 조작 중 XML 파일을 작성할 때 파일 이름은 이 목록의 현재 기본 이름을 현재 경로(*xml-path*로부터)에 추가하고 3자리 숫자 시퀀스 번호를 추가한 후 3자 ID `xml`을 추가합니다. 예를 들어, 현재 XML 경로가 `/u/foo/xml/path/` 디렉토리이고 현재 XML 파일 이름이 `bar`이면 작성되는 XML 파일은 `/u/foo/xml/path/bar.001.xml`, `/u/foo/xml/path/bar.002.xml` 등입니다.

XML TO *xml-path*

XML 파일이 저장될 디렉토리에 대한 하나 이상의 경로를 지정하십시오. XML 경로당 최소한 하나의 파일이 있으며 각 파일에는 최소한 하나의 XDM(XQuery Data Model) 인스턴스가 있습니다. 두 개 이상의 경로를 지정하면 XDM 인스턴스는 해당 경로에 균일하게 분산됩니다.

XMLSAVESCHEMA

모든 XML 컬럼에 대해 XML 스키마 정보가 저장되어야 함을 지정합니다. 각 익스포트된 XM 문서가 삽입될 때 XML 스키마에 대해 유효성이 확인되었으면 해당 스키마의 완전한 SQL ID는 해당 XDS(XML Data Specifier) 내에 (SCH) 속성으로 저장됩니다. 익스포트된 문서가 XML 스키마에 대해 유효성이 확인되지 않았거나 데이터베이스에 스키마 오브젝트가 더 이상 존재하지 않으면 해당 XDS에 SCH 속성이 포함되지 않습니다.

SQL ID의 스키마 및 이름 부분은 XML 스키마에 해당하는 SYSCAT.XSROBJECTS 카탈로그 테이블 행에 ″OBJECTSCHEMA″ 및 ″OBJECTNAME″ 값으로 저장됩니다.

XMLSAVESCHEMA 옵션은 올바른 형식의 XML 문서를 생성하지 않는 XQuery 시퀀스와는 호환 가능하지 않습니다.

사용 시 참고사항

- 익스포트 조작을 시작하기 전에 반드시 모든 테이블 조작을 완료하고 모든 잠금을 릴리스하십시오. 이는 WITH HOLD로 열려진 모든 커서를 닫고 COMMIT를 발행하거나 ROLLBACK을 발행하여 수행할 수 있습니다.
- SELECT문에서 테이블 별명을 사용할 수 있습니다.
- 메시지 파일에 넣어진 메시지는 메시지 검색 서비스에서 리턴된 정보를 포함합니다. 각 메시지는 새 행에서 시작합니다.
- 익스포트 유틸리티는 DEL 형식 파일로 익스포트하기 위해 길이가 254를 초과하는 문자 컬럼을 선택할 때마다 경고 메시지를 생성합니다.
- 데이터베이스 간에 데이터를 이동하려면 PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다.
- 동일한 클라이언트에서 소스 또는 목표 데이터베이스에 모두 액세스할 수 있는 경우 파일 복사 단계는 필요하지 않습니다.
- DB2 Connect는 OS/390®용 DB2, VM 및 VSE용 DB2, OS/400®용 DB2와 같은 DRDA® 서버에서 테이블을 익스포트하는 데 사용할 수 있습니다. PC/IXF 익스포트만 지원됩니다.
- IXF 형식으로 익스포트할 때, ID가 IXF 형식에서 지원하는 최대 *크기*를 초과하는 경우, 익스포트에 성공하지만 결과 데이터 파일은 CREATE 모드를 사용하는 추후 임포트 조작에서는 사용될 수 없습니다. SQL27984W가 리턴됩니다.
- Windows의 디스켓으로 익스포트하고 단일 디스켓의 용량보다 큰 데이터가 있는 테이블을 익스포트할 때, 시스템은 다른 디스켓을 위해 프롬프트를 표시하며, 다중 파트 PC/IXF 파일(다중 볼륨 PC/IXF 파일 또는 논리 분할 PC/IXF 파일이라고도 함)이 생성되어 별도의 디스켓에 저장됩니다. 마지막의 경우를 제외하고 각 파일에는, 파일이 논리적으로 분할되었음을 표시하고 다음 파일을 찾을 곳을 나타내기 위해 쓰여진 DB2 CONTINUATION RECORD(또는 짧게 ″AC″ 레코드)가 있습니다. AIX® 시스템으로 해당 파일을 전송하여, 임포트 및 로드 유틸리티에서 읽을 수 있습니다. 익스포트 유틸리티는 AIX 시스템에서 호출될 때 다중 파트 PC/IXF 파일을 작성하지 않습니다. 자세한 사용을 위해서는 IMPORT 명령 또는 LOAD 명령을 참조하십시오.

- 익스포트 유틸리티는 제공된 SELECT문이 SELECT * FROM tablename 형식인 경우 해당 테이블의 NOT NULL WITH DEFAULT 속성을 IXF 파일에 저장합니다.
- 유형이 지정된 테이블을 익스포트할 때 subselect문은 단지 목표 테이블 이름과 WHERE 절을 지정하여 표현할 수 있습니다. 계층 구조를 익스포트할 때는 Fullselect 및 *select-statement*를 지정할 수 없습니다.
- IXF 이외의 파일 형식에서는 트래버스 순서 목록을 지정할 것을 권장합니다. 이는 이 순서가 DB2에게 계층 구조를 트래버스하는 방법과 어떤 서브테이블을 익스포트할지를 알려주기 때문입니다. 이 목록을 지정하지 않으면 해당 계층 구조에 있는 모든 테이블이 익스포트되고 디폴트 순서는 OUTER 순서가 됩니다. 대안은 OUTER 기능이 제공하는 순서인 디폴트 순서를 사용하는 방법입니다.
- 임포트 조작 동안 동일한 트래버스 순서를 사용하십시오. 로드 유틸리티는 계층 구조 또는 하위 계층 구조를 지원하지 않습니다.
- 행이 보호 설정된 테이블에서 데이터를 익스포트할 때 세션 권한 부여 ID가 보유하는 LBAC 증명서가 익스포트되는 행을 제한할 수 있습니다. 세션 권한 부여 ID가 읽기 권한을 가지고 있지 않은 행은 익스포트되지 않습니다. 오류나 경고는 표시되지 않습니다.
- 세션 권한 부여 ID가 보유하는 LBAC 증명서가 익스포트에 포함된 하나 이상의 보호 설정된 컬럼에서 읽기를 허용하지 않을 경우 익스포트를 실패하고 오류(SQLSTATE 42512)가 리턴됩니다.
- 익스포트 패키지는 DATETIME ISO 형식을 사용하여 바운드되므로 문자열 표현식으로 캐스트할 때 모든 날짜/시간/시간소인 값은 ISO 형식으로 변환됩니다. CLP 패키지는 DATETIME LOC 형식(로케일 특정 형식)을 사용하여 바운드되므로 CLP DATETIME 형식이 ISO 형식과 다른 경우 CLP 및 익스포트 간에 동작이 불일치할 수 있습니다. 예를 들어, 다음 SELECT문에서는 예상대로 결과가 리턴됩니다.

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
   COL2
   ----------
   05/10/2005
   05/10/2005
   05/10/2005
   3 레코드가 선택되었습니다.
```

그러나 동일한 Select절을 사용한 Export 명령에서는 예상대로 결과가 리턴되지 않습니다.

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
   익스포트된 행 수: 0
```

이제 LOCALE 날짜 형식을 ISO 형식으로 바꾸면 예상대로 결과가 리턴됩니다.

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
  익스포트된 행 수: 3
```

익스포트 유틸리티의 파일 유형 수정자

표 3. 익스포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식

수정자	설명
lobsinfile	*lob-path*는 LOB 데이터를 포함하는 파일에 대한 경로를 지정합니다. 각 경로는 LLS(Lob Location Specifier)에 의해 포인트되는 최소한 하나의 LOB를 포함하는 최소한 하나의 파일을 데이터 파일에 포함합니다. LLS는 LOB 파일 경로에 저장된 파일에서 LOB 위치의 문자열 표시입니다. LLS의 형식은 *filename.ext.nnn.mmm/*이며, 여기서 *filename.ext*는 LOB를 포함하는 파일의 이름이며, *nnn*은 파일 내에서 LOB의 오프셋을 바이트로 나타낸 것이며, *mmm*은 LOB의 길이를 바이트로 나타낸 것입니다. 예를 들어, 문자열 db2exp.001.123.456/가 데이터 파일에 저장되는 경우, LOB는 db2exp.001 파일에서 오프셋 123에 위치하며 456바이트 길이입니다. EXPORT 사용 시 『lobsinfile』 수정자를 지정하는 경우, LOB 데이터는 LOBS TO 절에서 지정하는 위치에 있습니다. 그렇지 않으면 LOB 데이터는 데이터 파일 디렉토리로 보냅니다. LOBS TO 절은 LOB 파일이 저장될 디렉토리에 대한 하나 이상의 경로를 지정합니다. LOB 경로 당 최소한 하나의 파일이 있으며 각 파일에는 최소한 하나의 LOB가 있습니다. LOBS TO 또는 LOBFILE 옵션은 내재적으로 LOBSINFILE 동작을 활성화합니다. 널(NULL) LOB를 표시하려면, *크기*를 -1로 입력하십시오. *크기*가 0으로 지정되면, 길이가 0인 LOB로 처리됩니다. 길이가 -1인 널(NULL) LOBS의 경우, 오프셋 및 파일 이름은 무시됩니다. 예를 들어, 널(NULL) LOB의 LLS는 db2exp.001.7.-1/입니다.
xmlinsepfiles	각 XDM(XQuery Data Model) 인스턴스는 별도의 파일에 쓰여집니다. 디폴트로 복수의 값은 동일한 파일에서 함께 병합됩니다.
lobsinsepfiles	각 LOB 값은 별도의 파일에 쓰여집니다. 디폴트로 복수의 값은 동일한 파일에서 함께 병합됩니다.
xmlnodeclaration	XDM 인스턴스는 XML 선언 태그 없이 쓰여집니다. 디폴트로 XDM 인스턴스는 처음에 인코딩 속성을 포함하는 XML 선언 태그를 가지고 익스포트됩니다.
xmlchar	XDM 인스턴스는 문자 코드 페이지에 쓰여집니다. 문자 코드 페이지는 codepage 파일 유형 수정자에서 지정한 값이거나, 지정되지 않은 경우 응용프로그램 코드 페이지입니다. 디폴트로, XDM 인스턴스는 유니코드로 쓰여집니다.
xmlgraphic	xmlgraphic 수정자가 EXPORT 명령에서 지정된 경우, 익스포트된 XML 문서는 응용프로그램 코드 페이지나 codepage 파일 유형 수정자와 관계없이 UTF-16 코드 페이지로 인코드됩니다.

표 4. 익스포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식

수정자	설명

chardel*x*	*x*는 단일 문자열 분리문자입니다. 디폴트값은 큰따옴표(")입니다. 문자열을 묶기 위해 큰따옴표 대신 지정된 문자가 사용됩니다.[2] 명시적으로 큰따옴표를 문자열 분리문자로 지정하려는 경우 다음과 같이 지정해야 합니다.
	
	`modified by chardel""`
	
	다음과 같이 작은따옴표(')를 문자열 분리문자로 지정할 수도 있습니다.
	
	`modified by chardel''`

표 4. 익스포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식 (계속)

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 출력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 익스포트 조작 중 응용프로그램 코드 페이지로부터 이 코드 페이지로 문자 데이터를 변환합니다. 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. `codepage` 수정자는 `lobsinfile` 수정자와 함께 사용될 수 없습니다.

coldel*x*	*x*는 단일 문자 컬럼 분리문자입니다. 디폴트값은 쉼표(,)입니다. 지정된 문자가 쉼표 대신 사용되어 컬럼 끝을 나타냅니다.[2]
	
	다음 예에서, `coldel;`은 익스포트 유틸리티가 익스포트된 데이터의 컬럼 분리문자로 세미콜론 문자(;)를 사용하게 합니다.
	`db2 "export to temp of del modified by coldel;`
	`select * from staff where dept = 20"`

decplusblank	플러스 부호 문자. 플러스 부호(+) 대신 공백을 양의 10진수 값 앞에 붙입니다. 디폴트 조치는 플러스 부호를 양의 10진수 값 앞에 붙이는 것입니다.
decpt*x*	*x*는 소수점 문자로 마침표를 대신하는 단일 문자입니다. 디폴트값은 마침표(.)입니다. 마침표 대신 지정된 문자가 소수점 문자로 사용됩니다.[2]
nochardel	컬럼 데이터는 문자 분리문자 안에 두지 않습니다. DB2를 사용하여 데이터를 임포트하거나 로드하려는 경우 이 옵션을 지정하지 말아야 합니다. 이 옵션은 문자 분리문자가 없는 벤더 데이터 파일을 지원하기 위해 제공됩니다. 부적절한 사용은 데이터 유실이나 손상을 초래할 수 있습니다. `chardelx` 또는 `nodoubledel`에는 이 옵션을 지정할 수 없습니다. 이들은 상호 독점 옵션입니다.
nodoubledel	2바이트 분리문자를 인식하지 않습니다.[2]

striplzeros	익스포트된 모든 10진수 컬럼에서 선행 영(0)을 제거합니다.
	
	다음 예를 고려해 보십시오.
	`db2 create table decimalTable ( c1 decimal( 31, 2 ) )`
	`db2 insert into decimalTable values ( 1.1 )`
	
	`db2 export to data of del select * from decimalTable`
	
	`db2 export to data of del modified by STRIPLZEROS`
	`select * from decimalTable`
	
	첫 번째 익스포트 조작에서, 익스포트된 파일 데이터의 컨텐츠는 +00000000000000000000000000001.10입니다. `striplzeros` 수정자를 제외하고 첫 번째와 동일한 두 번째 조작에서, 익스포트된 파일 데이터의 컨텐츠는 +1.10입니다.

표 4. 익스포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식 (계속)

수정자	설명

timestampformat="*x*"	*x*는 소스 파일에서 시간소인의 형식입니다.[4] 유효한 시간소인 요소는 다음과 같습니다.
	
	YYYY - 연도(0000 - 9999 사이의 4자리 숫자)
	M - 월(1 - 12 사이의 1 또는 2자리 숫자)
	MM - 월(01 - 12 사이의 2자리 숫자, M 및 MMM은 상호 배타적)
	MMM - 월(월 이름으로 사용되는 3자리 문자로 대소문자를 구분하지 않으며, M 및 MM은 상호 배타적)
	D - 일(1 - 31 사이의 1 또는 2자리 숫자)
	DD - 일(1 - 31 사이의 2자리 숫자, D와는 상호 배타적)
	DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적)
	H - 시간(12시간 시스템의 경우 0 - 12 범위의 1 또는 2자리 숫자, 24시간 시스템의 경우 0 - 24)
	HH - 시간(12시간 시스템의 경우 0 - 12 범위의 2자리 숫자, 24시간 시스템의 경우 0 - 24, H와는 상호 배타적)
	M - 분(0 - 59 사이의 1 또는 2자리 숫자)
	MM - 분(0 - 59 사이의 2자리 숫자, M, 분과 상호 배타적)
	S - 초(0 - 59 사이의 1 또는 2자리 숫자)
	SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적)
	SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적)
	U(1 - 12회) - 초의 소수 부분(U 어커런스 수는 각 자리가 0 - 9 사이의 자리 수를 나타냄)
	TT - 오전/오후 지시(AM 또는 PM)
	
	다음은 시간소인 형식의 예입니다.
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 요소는 다음 값을 생성합니다. 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov' 및 'Dec'. 'Jan'은 1월이며, 'Dec'는 12월과 같습니다.
	
	다음 예는 'schedule'이라는 테이블에서 사용자 정의 시간소인을 포함하는 데이터를 익스포트하는 방법을 설명합니다.
	
	`db2 export to delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`select * from schedule`

표 5. 익스포트 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 출력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 익스포트 조작 중 이 코드 페이지로부터 응용프로그램 코드 페이지로 문자 데이터를 변환합니다. 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. codepage 수정자는 lobsinfile 수정자와 함께 사용될 수 없습니다.

표 6. 익스포트 유틸리티의 유효한 파일 유형 수정자: WSF 파일 형식[6]

수정자	설명
1	Lotus 1-2-3 릴리스 1 또는 Lotus 1-2-3 릴리스 1a와 호환 가능한 WSF 파일을 작성합니다.[5] 이것이 디폴트입니다.
2	Lotus Symphony 릴리스 1.0과 호환 가능한 WSF 파일을 작성합니다.[5]
3	Lotus 1-2-3 버전 2 또는 Lotus Symphony 릴리스 1.1과 호환 가능한 WSF 파일을 작성합니다.[5]
4	DBCS 문자를 포함하는 WSF 파일을 작성합니다.

주:

1. MODIFIED BY 옵션과 함께 지원되지 않는 파일 유형을 사용하려고 시도하는 경우 익스포트 유틸리티는 경고를 발행하지 않습니다. 이런 경우, 익스포트 조작에 실패하며 오류 코드가 리턴됩니다.
2. *데이터 이동을 위한 분리문자 고려사항*은 분리문자를 겹쳐쓰기할 때 사용할 수 있는 문자에 적용하는 제한사항을 나열합니다.
3. 정상적인 익스포트 유틸리티 쓰기
   - *YYYYMMDD* 형식의 날짜 데이터
   - *"YYYY-MM-DD"* 형식의 문자(날짜) 데이터
   - *"HH.MM.SS"* 형식의 시간 데이터
   - *"YYYY-MM-DD-HH .MM.SS.uuuuuu"* 형식의 시간 소인 데이터

   익스포트 조작의 SELECT문에서 지정된 날짜 및 시간 컬럼에 포함된 데이터도 다음 형식입니다.
4. 시간소인 형식의 경우, 문자 M을 사용하는 month 및 minute 디스크립터 간의 모호함을 피하려면 주의해야 합니다. month 필드는 다른 날짜 필드와 인접해야 합니다. minute 필드는 다른 시간 필드와 인접해야 합니다. 다음은 모호한 시간소인 형식입니다.

   ```
   "M"(month 또는 minute일 수 있음)
   "M:M"(month 및 minute 구분 가능?)
   "M:YYYY:M"(둘 다 month로 해석됨)
   "S:M:YYYY"(둘 다 시간 값 및 날짜 값에 인접)
   ```

   모호한 경우, 유틸리티는 오류 메시지를 발행하며 조작에 실패합니다.

   다음은 명확한 시간소인 형식입니다.

   ```
   "M:YYYY" (Month)
   "S:M" (Minute)
   "M:YYYY:S:M" (Month....Minute)
   "M:H:YYYY:M:D" (Minute....Month)
   ```
5. 또한 이러한 파일은 Lotus 1-2-3의 경우 L, Symphony의 경우 S를 *filetype-mod* 매개변수 문자열에 지정하여 특정 제품으로 방향지정할 수 있습니다. 하나의 값 또는 제품 지정자만 지정할 수 있습니다. WSF 파일 형식에 대한 지원은 사용되지

않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

6. WSF 파일 형식은 XML 컬럼에서 지원되지 않습니다. 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.
7. XMLFILE 또는 XML TO 절 중 어느 하나도 지정되지 않아도 모든 XDM 인스턴스는 기본 데이터 파일에서 분리된 XML 파일에 쓰여집니다. 디폴트로, XML 파일은 익스포트된 데이터 파일의 경로에 쓰여집니다. 디폴트로 XML 파일의 기본 이름은 익스포트된 데이터 파일의 이름으로 확장자가 ″.xml″입니다.
8. XMLNODECLARATION 파일 유형 수정자가 지정된 경우를 제외하고, 모든 XDM 인스턴스는 처음에 인코딩 속성을 포함하는 XML 선언을 이용하여 쓰여집니다.
9. XMLCHAR 또는 XMLGRAPHIC 파일 유형 수정자가 지정된 경우를 제외하고, 디폴트로 모든 XDM 인스턴스는 유니코드로 쓰여집니다.
10. XML 데이터 및 LOB 데이터의 디폴트 경로는 기본 데이터 파일 경로입니다. 디폴트 XML 파일 기본 이름이 주 데이터 파일입니다. 디폴트 LOB 파일 기본 이름이 주 데이터 파일입니다. 예를 들어, 주 데이터 파일이

    ```
    /mypath/myfile.del
    ```

    이고 XML 데이터 및 LOB 데이터의 디폴트 경로는

    ```
    /mypath"
    ```

    이며 디폴트 XML 파일 기본 이름은

    ```
    myfile.del
    ```

    이고 디폴트 LOB 파일 기본 이름은 다음과 같은 경우,

    ```
    myfile.del
    ```

    LOB 파일을 생성하려면 LOBSINFILE 파일 유형 수정자를 지정해야 합니다.
11. 익스포트 유틸리티는 각 LOB 파일이나 XML 파일에 숫자 ID를 추가합니다. ID는 0으로 패드된 시퀀스 값인 3자리 숫자로 시작하며

    ```
    .001
    ```

    에서 시작합니다. 999번째 LOB 파일 또는 XML 파일 이후 ID는 더 이상 0으로 패드되지 않습니다. 예를 들어, 1000번째 LOG 파일이나 XML 파일의 확장자는

    ```
    .1000
    ```

    입니다. 숫자 ID 다음에는 데이터 유형을 나타내는 세 자 유형 ID가 표시됩니다(

    ```
    .lob
    ```

    또는

.xml을 나타냄)

. 예를 들어, 생성된 LOB 파일 이름의 형식은

```
myfile.del.001.lob
```

이고 생성된 XML 파일 이름의 형식은 다음과 같습니다.

```
myfile.del.001.xml
```

12. XQuery를 지정하여 잘 양식화된 문서가 아닌 익스포트 유틸리티 익스포트 XDM 인스턴스를 갖는 것이 가능합니다. 그러나 XML 컬럼은 완전한 문서만을 포함하므로, 해당 익스포트된 문서를 직접 XML 컬럼에 임포트하거나 로드할 수 없습니다.

ADMIN_CMD 프로시저를 사용하는 EXPORT 명령

데이터베이스의 데이터를 여러 외부 파일 형식 중 하나로 익스포트합니다. 사용자는 SQL SELECT문을 제공하거나 유형이 지정된 테이블에 대한 계층 정보를 제공하여 익스포트할 데이터를 지정합니다.

36 페이지의 『익스포트 유틸리티의 파일 유형 수정자』로의 빠른 링크를 참조하십시오.

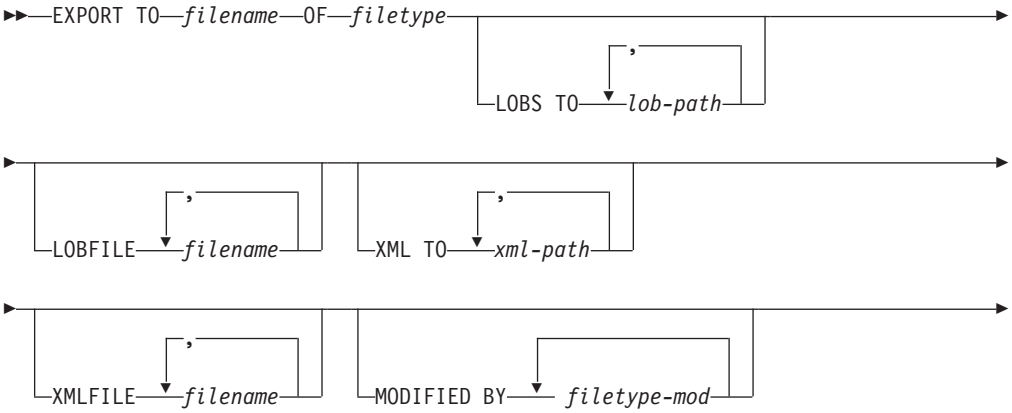
권한 부여

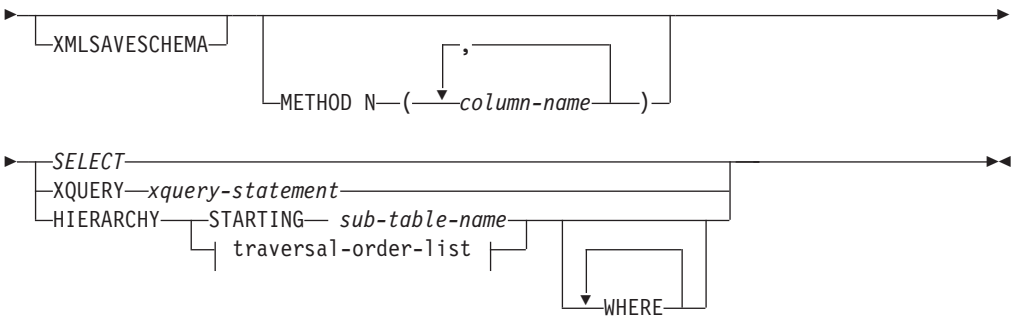
다음 중 하나가 필요합니다.

- *dataaccess* 권한
- 참여 중인 각 테이블 또는 뷰에 대한 CONTROL 또는 SELECT 특권

필수 연결

명령 구문





traversal-order-list:

명령 매개변수

HIERARCHY *traversal-order-list*

지정된 트래버스 순서를 사용하여 하위 계층 구조를 익스포트합니다. 모든 서브테이블은 PRE-ORDER 형식으로 나열되어야 합니다. 첫 번째 서브테이블 이름을 SELECT문의 목표 테이블 이름으로 사용합니다.

HIERARCHY STARTING *sub-table-name*

디폴트 트래버스 순서(ASC, DEL 또는 WSF 파일의 경우 OUTER 순서 또는 PC/IXF 데이터 파일에 저장된 순서)를 사용하여 *sub-table-name*에서 시작하는 하위 계층 구조를 익스포트하십시오.

LOBFILE *filename*

LOB 파일에 대한 하나 이상의 기본 파일을 지정합니다. 첫 번째 이름에 대한 이름 공간이 모두 사용되었으면 두 번째 이름을 사용하고 계속 이와 같이 합니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

익스포트 조작 중에 LOB 파일을 작성할 경우, 파일 이름은 해당 목록의 현재 기본 이름을 현 재 경로(*lob-path*의)에 추가한 후 시작할 3자리 시퀀스 번호 및 3자의 ID(lob)를 추가하여 구성됩니다. 예를 들어, 현재 LOB 경로가 /u/foo/lob/path/ 디렉토리이고 현재 LOB 파일 이름이 bar이면 작성되는 LOB 파일은 /u/foo/lob/path/bar.001.lob, /u/foo/lob/path/bar.002.lob 등입니다. 999가 사용되고 나면 LOB 파일 이름의 3자리 시퀀스 번호는 4자리가 되며, 9999가 사용되고 나면 4자리는 5자리가 되는 식으로 진행됩니다.

LOBS TO *lob-path*

LOB 파일이 저장될 디렉토리에 대한 하나 이상의 경로를 지정하십시오. LOB

경로 당 최소한 하나의 파일이 있으며 각 파일에는 최소한 하나의 LOB가 있습니다. 지정할 수 있는 최대 경로 수는 999입니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

METHOD N *column-name*

출력 파일에서 사용될 하나 이상의 컬럼 이름을 지정합니다. 이 매개변수를 지정하지 않으면 테이블에 있는 컬럼 이름이 사용됩니다. 이 매개변수는 WSF 및 IXF 파일에 대해서만 유효하며 계층 데이터를 익스포트할 때는 유효하지 않습니다.

MODIFIED BY *filetype-mod*

파일 유형 수정자 옵션을 지정합니다. 36 페이지의 『익스포트 유틸리티의 파일 유형 수정자』를 참조하십시오.

OF *filetype*

출력 파일에 있는 데이터의 형식을 지정합니다.

- DEL(컬럼 식별자가 있는 ASCII 형식) - 여러 데이터베이스 관리 프로그램 및 파일 관리자 프로그램에서 사용됩니다.
- WSF(작업시트 형식) - 다음과 같은 프로그램에서 사용됩니다.
  - Lotus 1-2-3
  - Lotus Symphony

  BIGINT 또는 DECIMAL 데이터를 익스포트할 때 DOUBLE 유형의 범위에 해당하는 값만 정확히 익스포트할 수 있습니다. 이 범위에 해당하지 않는 값은 익스포트되기는 하지만 이들 값을 다시 임포트하거나 로드할 때 운영 체제에 따라 올바르지 않은 데이터가 나타날 수 있습니다.

  주: WSF 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.
- IXF(Integration Exchange Format, PC 버전)는 독점 2진 형식입니다.

SELECT

익스포트될 데이터를 리턴하는 SELECT 또는 XQUERY문을 지정합니다. 명령문이 오류를 일으킬 경우 메시지 파일(또는 표준 출력)에 메시지가 쓰여집니다. 오류 코드가 SQL0012W, SQL0347W, SQL0360W, SQL0437W 또는 SQL1824W 중 하나이면 익스포트 조작을 계속 수행하고 그렇지 않으면 중지합니다.

TO *filename*

이미 존재하는 파일 이름을 지정할 경우 익스포트 유틸리티는 파일 내용을 겹쳐쓰고 정보를 추가하지 않습니다.

XMLFILE *filename*

XML 파일에 대한 하나 이상의 기본 파일 이름을 지정합니다. 첫 번째 이름에 대한 이름 공간이 모두 사용되었으면 두 번째 이름을 사용하고 계속 이와 같이 합니다.

익스포트 조작 중 XML 파일을 작성할 때 파일 이름은 이 목록의 현재 기본 이름을 현재 경로(*xml-path*로부터)에 추가하고 3자리 숫자 시퀀스 번호를 추가한 후 3자 ID `xml`을 추가합니다. 예를 들어, 현재 XML 경로가 `/u/foo/xml/path/` 디렉토리이고 현재 XML 파일 이름이 `bar`이면 작성되는 XML 파일은 `/u/foo/xml/path/bar.001.xml`, `/u/foo/xml/path/bar.002.xml` 등입니다.

XML TO *xml-path*

XML 파일이 저장될 디렉토리에 대한 하나 이상의 경로를 지정하십시오. XML 경로당 최소한 하나의 파일이 있으며 각 파일에는 최소한 하나의 XDM(XQuery Data Model) 인스턴스가 있습니다. 두 개 이상의 경로를 지정하면 XDM 인스턴스는 해당 경로에 균일하게 분산됩니다.

XMLSAVESCHEMA

모든 XML 컬럼에 대해 XML 스키마 정보가 저장되어야 함을 지정합니다. 각 익스포트된 XM 문서가 삽입될 때 XML 스키마에 대해 유효성이 확인되었으면 해당 스키마의 완전한 SQL ID는 해당 XDS(XML Data Specifier) 내에 (SCH) 속성으로 저장됩니다. 익스포트된 문서가 XML 스키마에 대해 유효성이 확인되지 않았거나 데이터베이스에 스키마 오브젝트가 더 이상 존재하지 않으면 해당 XDS에 SCH 속성이 포함되지 않습니다.

SQL ID의 스키마 및 이름 부분은 XML 스키마에 해당하는 SYSCAT.XSROBJECTS 카탈로그 테이블 행에 ″OBJECTSCHEMA″ 및 ″OBJECTNAME″ 값으로 저장됩니다.

XMLSAVESCHEMA 옵션은 올바른 형식의 XML 문서를 생성하지 않는 XQuery 시퀀스와는 호환 가능하지 않습니다.

사용 시 참고사항

- 익스포트 조작을 시작하기 전에 반드시 모든 테이블 조작을 완료하고 모든 잠금을 릴리스하십시오. 이는 WITH HOLD로 열려진 모든 커서를 닫고 COMMIT를 발행하거나 ROLLBACK을 발행하여 수행할 수 있습니다.
- SELECT문에서 테이블 별명을 사용할 수 있습니다.
- 메시지 파일에 넣어진 메시지는 메시지 검색 서비스에서 리턴된 정보를 포함합니다. 각 메시지는 새 행에서 시작합니다.
- 익스포트 유틸리티는 DEL 형식 파일로 익스포트하기 위해 길이가 254를 초과하는 문자 컬럼을 선택할 때마다 경고 메시지를 생성합니다.

- 데이터베이스 간에 데이터를 이동하려면 PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다.
- 동일한 클라이언트에서 소스 또는 목표 데이터베이스에 모두 액세스할 수 있는 경우 파일 복사 단계는 필요하지 않습니다.
- DB2 Connect는 OS/390용 DB2, VM 및 VSE용 DB2, OS/400용 DB2와 같은 DRDA 서버에서 테이블을 익스포트하는 데 사용할 수 있습니다. PC/IXF 익스포트만 지원됩니다.
- IXF 형식으로 익스포트할 때, ID가 IXF 형식에서 지원하는 최대 *크기*를 초과하는 경우, 익스포트에 성공하지만 결과 데이터 파일은 CREATE 모드를 사용하는 추후 임포트 조작에서는 사용될 수 없습니다. SQL27984W가 리턴됩니다.
- Windows의 디스켓으로 익스포트하고 단일 디스켓의 용량보다 큰 데이터가 있는 테이블을 익스포트할 때, 시스템은 다른 디스켓을 위해 프롬프트를 표시하며, 다중 파트 PC/IXF 파일(다중 볼륨 PC/IXF 파일 또는 논리 분할 PC/IXF 파일이라고도 함)이 생성되어 별도의 디스켓에 저장됩니다. 마지막의 경우를 제외하고 각 파일에는, 파일이 논리적으로 분할되었음을 표시하고 다음 파일을 찾을 곳을 나타내기 위해 쓰여진 DB2 CONTINUATION RECORD(또는 짧게 ″AC″ 레코드)가 있습니다. AIX 시스템으로 해당 파일을 전송하여, 임포트 및 로드 유틸리티에서 읽을 수 있습니다. 익스포트 유틸리티는 AIX 시스템에서 호출될 때 다중 파트 PC/IXF 파일을 작성하지 않습니다. 자세한 사용을 위해서는 IMPORT 명령 또는 LOAD 명령을 참조하십시오.
- 익스포트 유틸리티는 제공된 SELECT문이 `SELECT * FROM tablename` 형식인 경우 해당 테이블의 NOT NULL WITH DEFAULT 속성을 IXF 파일에 저장합니다.
- 유형이 지정된 테이블을 익스포트할 때 subselect문은 단지 목표 테이블 이름과 WHERE 절을 지정하여 표현할 수 있습니다. 계층 구조를 익스포트할 때는 Fullselect 및 *select-statement*를 지정할 수 없습니다.
- IXF 이외의 파일 형식에서는 트래버스 순서 목록을 지정할 것을 권장합니다. 이는 이 순서가 DB2에게 계층 구조를 트래버스하는 방법과 어떤 서브테이블을 익스포트할지를 알려주기 때문입니다. 이 목록을 지정하지 않으면 해당 계층 구조에 있는 모든 테이블이 익스포트되고 디폴트 순서는 OUTER 순서가 됩니다. 대안은 OUTER 기능이 제공하는 순서인 디폴트 순서를 사용하는 방법입니다.
- 임포트 조작 동안 동일한 트래버스 순서를 사용하십시오. 로드 유틸리티는 계층 구조 또는 하위 계층 구조를 지원하지 않습니다.

- 행이 보호 설정된 테이블에서 데이터를 익스포트할 때 세션 권한 부여 ID가 보유하는 LBAC 증명서가 익스포트되는 행을 제한할 수 있습니다. 세션 권한 부여 ID가 읽기 권한을 가지고 있지 않은 행은 익스포트되지 않습니다. 오류나 경고는 표시되지 않습니다.
- 세션 권한 부여 ID가 보유하는 LBAC 증명서가 익스포트에 포함된 하나 이상의 보호 설정된 컬럼에서 읽기를 허용하지 않을 경우 익스포트를 실패하고 오류(SQLSTATE 42512)가 리턴됩니다.
- 익스포트 패키지는 DATETIME ISO 형식을 사용하여 바운드되므로 문자열 표현식으로 캐스트할 때 모든 날짜/시간/시간소인 값은 ISO 형식으로 변환됩니다. CLP 패키지는 DATETIME LOC 형식(로케일 특정 형식)을 사용하여 바운드되므로 CLP DATETIME 형식이 ISO 형식과 다른 경우 CLP 및 익스포트 간에 동작이 불일치할 수 있습니다. 예를 들어, 다음 SELECT문에서는 예상대로 결과가 리턴됩니다.

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
   COL2
   ----------
   05/10/2005
   05/10/2005
   05/10/2005
   3 레코드가 선택되었습니다.
```

그러나 동일한 Select절을 사용한 Export 명령에서는 예상대로 결과가 리턴되지 않습니다.

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
   익스포트된 행 수: 0
```

이제 LOCALE 날짜 형식을 ISO 형식으로 바꾸면 예상대로 결과가 리턴됩니다.

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
   익스포트된 행 수: 3
```

익스포트 유틸리티의 파일 유형 수정자

표 7. 익스포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식

수정자	설명
lobsinfile	*lob-path*는 LOB 데이터를 포함하는 파일에 대한 경로를 지정합니다. 각 경로는 LLS(Lob Location Specifier)에 의해 포인트되는 최소한 하나의 LOB를 포함하는 최소한 하나의 파일을 데이터 파일에 포함합니다. LLS는 LOB 파일 경로에 저장된 파일에서 LOB 위치의 문자열 표시입니다. LLS의 형식은 *filename.ext.nnn.mmm/*이며, 여기서 *filename.ext*는 LOB를 포함하는 파일의 이름이며, *nnn*은 파일 내에서 LOB의 오프셋을 바이트로 나타낸 것이며, *mmm*은 LOB의 길이를 바이트로 나타낸 것입니다. 예를 들어, 문자열 db2exp.001.123.456/가 데이터 파일에 저장되는 경우, LOB는 db2exp.001 파일에서 오프셋 123에 위치하며 456바이트 길이입니다. EXPORT 사용 시 『lobsinfile』 수정자를 지정하는 경우, LOB 데이터는 LOBS TO 절에서 지정하는 위치에 있습니다. 그렇지 않으면 LOB 데이터는 데이터 파일 디렉토리로 보냅니다. LOBS TO 절은 LOB 파일이 저장될 디렉토리에 대한 하나 이상의 경로를 지정합니다. LOB 경로 당 최소한 하나의 파일이 있으며 각 파일에는 최소한 하나의 LOB가 있습니다. LOBS TO 또는 LOBFILE 옵션은 내재적으로 LOBSINFILE 동작을 활성화합니다. 널(NULL) LOB를 표시하려면, *크기*를 -1로 입력하십시오. *크기*가 0으로 지정되면, 길이가 0인 LOB로 처리됩니다. 길이가 -1인 널(NULL) LOBS의 경우, 오프셋 및 파일 이름은 무시됩니다. 예를 들어, 널(NULL) LOB의 LLS는 db2exp.001.7.-1/입니다.
xmlinsepfiles	각 XDM(XQuery Data Model) 인스턴스는 별도의 파일에 쓰여집니다. 디폴트로 복수의 값은 동일한 파일에서 함께 병합됩니다.
lobsinsepfiles	각 LOB 값은 별도의 파일에 쓰여집니다. 디폴트로 복수의 값은 동일한 파일에서 함께 병합됩니다.
xmlnodeclaration	XDM 인스턴스는 XML 선언 태그 없이 쓰여집니다. 디폴트로 XDM 인스턴스는 처음에 인코딩 속성을 포함하는 XML 선언 태그를 가지고 익스포트됩니다.
xmlchar	XDM 인스턴스는 문자 코드 페이지에 쓰여집니다. 문자 코드 페이지는 codepage 파일 유형 수정자에서 지정한 값이거나, 지정되지 않은 경우 응용프로그램 코드 페이지입니다. 디폴트로, XDM 인스턴스는 유니코드로 쓰여집니다.
xmlgraphic	xmlgraphic 수정자가 EXPORT 명령에서 지정된 경우, 익스포트된 XML 문서는 응용프로그램 코드 페이지나 codepage 파일 유형 수정자와 관계없이 UTF-16 코드 페이지로 인코드됩니다.

표 8. 익스포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식

수정자	설명

chardel*x*	*x*는 단일 문자열 분리문자입니다. 디폴트값은 큰따옴표(")입니다. 문자열을 묶기 위해 큰따옴표 대신 지정된 문자가 사용됩니다.[2] 명시적으로 큰따옴표를 문자열 분리문자로 지정하려는 경우 다음과 같이 지정해야 합니다.
	
	`modified by chardel""`
	
	다음과 같이 작은따옴표(')를 문자열 분리문자로 지정할 수도 있습니다.
	
	`modified by chardel''`

표 8. 익스포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식 (계속)

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 출력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 익스포트 조작 중 응용프로그램 코드 페이지로부터 이 코드 페이지로 문자 데이터를 변환합니다. 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. `codepage` 수정자는 `lobsinfile` 수정자와 함께 사용될 수 없습니다.

coldel*x*	*x*는 단일 문자 컬럼 분리문자입니다. 디폴트값은 쉼표(,)입니다. 지정된 문자가 쉼표 대신 사용되어 컬럼 끝을 나타냅니다.[2]
	
	다음 예에서, `coldel;`은 익스포트 유틸리티가 익스포트된 데이터의 컬럼 분리문자로 세미콜론 문자(;)를 사용하게 합니다.
	`db2 "export to temp of del modified by coldel;`
	`select * from staff where dept = 20"`

decplusblank	플러스 부호 문자. 플러스 부호(+) 대신 공백을 양의 10진수 값 앞에 붙입니다. 디폴트 조치는 플러스 부호를 양의 10진수 값 앞에 붙이는 것입니다.
decpt*x*	*x*는 소수점 문자로 마침표를 대신하는 단일 문자입니다. 디폴트값은 마침표(.)입니다. 마침표 대신 지정된 문자가 소수점 문자로 사용됩니다.[2]
nochardel	컬럼 데이터는 문자 분리문자 안에 두지 않습니다. DB2를 사용하여 데이터를 임포트하거나 로드하려는 경우 이 옵션을 지정하지 말아야 합니다. 이 옵션은 문자 분리문자가 없는 벤더 데이터 파일을 지원하기 위해 제공됩니다. 부적절한 사용은 데이터 유실이나 손상을 초래할 수 있습니다. `chardelx` 또는 `nodoubledel`에는 이 옵션을 지정할 수 없습니다. 이들은 상호 독점 옵션입니다.
nodoubledel	2바이트 분리문자를 인식하지 않습니다.[2]

striplzeros	익스포트된 모든 10진수 컬럼에서 선행 영(0)을 제거합니다.
	
	다음 예를 고려해 보십시오.
	`db2 create table decimalTable ( c1 decimal( 31, 2 ) )`
	`db2 insert into decimalTable values ( 1.1 )`
	
	`db2 export to data of del select * from decimalTable`
	
	`db2 export to data of del modified by STRIPLZEROS`
	`select * from decimalTable`
	
	첫 번째 익스포트 조작에서, 익스포트된 파일 데이터의 컨텐츠는 +0000000000000000000000000000001.10입니다. `striplzeros` 수정자를 제외하고 첫 번째와 동일한 두 번째 조작에서, 익스포트된 파일 데이터의 컨텐츠는 +1.10입니다.

표 8. 익스포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식 (계속)

수정자	설명

timestampformat="*x*"	*x*는 소스 파일에서 시간소인의 형식입니다.[4] 유효한 시간소인 요소는 다음과 같습니다.
	
	YYYY - 연도(0000 - 9999 사이의 4자리 숫자)
	M - 월(1 - 12 사이의 1 또는 2자리 숫자)
	MM - 월(01 - 12 사이의 2자리 숫자, M 및 MMM은 상호 배타적)
	MMM - 월(월 이름으로 사용되는 3자리 문자로 대소문자를 구분하지 않으며, M 및 MM은 상호 배타적)
	D - 일(1 - 31 사이의 1 또는 2자리 숫자)
	DD - 일(1 - 31 사이의 2자리 숫자, D와는 상호 배타적)
	DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적)
	H - 시간(12시간 시스템의 경우 0 - 12 범위의 1 또는 2자리 숫자, 24시간 시스템의 경우 0 - 24)
	HH - 시간(12시간 시스템의 경우 0 - 12 범위의 2자리 숫자, 24시간 시스템의 경우 0 - 24, H와는 상호 배타적)
	M - 분(0 - 59 사이의 1 또는 2자리 숫자)
	MM - 분(0 - 59 사이의 2자리 숫자, M, 분과 상호 배타적)
	S - 초(0 - 59 사이의 1 또는 2자리 숫자)
	SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적)
	SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적)
	U(1 - 12회) - 초의 소수 부분(U 어커런스 수는 각 자리가 0 - 9 사이의 자리 수를 나타냄)
	TT - 오전/오후 지시(AM 또는 PM)
	
	다음은 시간소인 형식의 예입니다.
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 요소는 다음 값을 생성합니다. 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov' 및 'Dec'. 'Jan'은 1월이며, 'Dec'는 12월과 같습니다.
	
	다음 예는 'schedule'이라는 테이블에서 사용자 정의 시간소인을 포함하는 데이터를 익스포트하는 방법을 설명합니다.
	
	`db2 export to delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`select * from schedule`

표 9. 익스포트 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 출력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 익스포트 조작 중 이 코드 페이지로부터 응용프로그램 코드 페이지로 문자 데이터를 변환합니다. 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. `codepage` 수정자는 `lobsinfile` 수정자와 함께 사용될 수 없습니다.

표 10. 익스포트 유틸리티의 유효한 파일 유형 수정자: WSF 파일 형식[6]

수정자	설명
1	Lotus 1-2-3 릴리스 1 또는 Lotus 1-2-3 릴리스 1a와 호환 가능한 WSF 파일을 작성합니다.[5] 이것이 디폴트입니다.
2	Lotus Symphony 릴리스 1.0과 호환 가능한 WSF 파일을 작성합니다.[5]
3	Lotus 1-2-3 버전 2 또는 Lotus Symphony 릴리스 1.1과 호환 가능한 WSF 파일을 작성합니다.[5]
4	DBCS 문자를 포함하는 WSF 파일을 작성합니다.

주:

1. MODIFIED BY 옵션과 함께 지원되지 않는 파일 유형을 사용하려고 시도하는 경우 익스포트 유틸리티는 경고를 발행하지 않습니다. 이런 경우, 익스포트 조작에 실패하며 오류 코드가 리턴됩니다.
2. *데이터 이동을 위한 분리문자 고려사항*은 분리문자를 겹쳐쓰기할 때 사용할 수 있는 문자에 적용하는 제한사항을 나열합니다.
3. 정상적인 익스포트 유틸리티 쓰기
   - *YYYYMMDD* 형식의 날짜 데이터
   - *"YYYY-MM-DD"* 형식의 문자(날짜) 데이터
   - *"HH.MM.SS"* 형식의 시간 데이터
   - *"YYYY-MM-DD-HH .MM.SS.uuuuuu"* 형식의 시간 소인 데이터

   익스포트 조작의 SELECT문에서 지정된 날짜 및 시간 컬럼에 포함된 데이터도 다음 형식입니다.
4. 시간소인 형식의 경우, 문자 M을 사용하는 month 및 minute 디스크립터 간의 모호함을 피하려면 주의해야 합니다. month 필드는 다른 날짜 필드와 인접해야 합니다. minute 필드는 다른 시간 필드와 인접해야 합니다. 다음은 모호한 시간소인 형식입니다.

   ```
   "M"(month 또는 minute일 수 있음)
   "M:M"(month 및 minute 구분 가능?)
   "M:YYYY:M"(둘 다 month로 해석됨)
   "S:M:YYYY"(둘 다 시간 값 및 날짜 값에 인접)
   ```

   모호한 경우, 유틸리티는 오류 메시지를 발행하며 조작에 실패합니다.

   다음은 명확한 시간소인 형식입니다.

   ```
   "M:YYYY" (Month)
   "S:M" (Minute)
   "M:YYYY:S:M" (Month....Minute)
   "M:H:YYYY:M:D" (Minute....Month)
   ```
5. 또한 이러한 파일은 Lotus 1-2-3의 경우 L, Symphony의 경우 S를 *filetype-mod* 매개변수 문자열에 지정하여 특정 제품으로 방향지정할 수 있습니다. 하나의 값 또는 제품 지정자만 지정할 수 있습니다. WSF 파일 형식에 대한 지원은 사용되지

않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

6. WSF 파일 형식은 XML 컬럼에서 지원되지 않습니다. 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.
7. XMLFILE 또는 XML TO 절 중 어느 하나도 지정되지 않아도 모든 XDM 인스턴스는 기본 데이터 파일에서 분리된 XML 파일에 쓰여집니다. 디폴트로, XML 파일은 익스포트된 데이터 파일의 경로에 쓰여집니다. 디폴트로 XML 파일의 기본 이름은 익스포트된 데이터 파일의 이름으로 확장자가 ″.xml″입니다.
8. XMLNODECLARATION 파일 유형 수정자가 지정된 경우를 제외하고, 모든 XDM 인스턴스는 처음에 인코딩 속성을 포함하는 XML 선언을 이용하여 쓰여집니다.
9. XMLCHAR 또는 XMLGRAPHIC 파일 유형 수정자가 지정된 경우를 제외하고, 디폴트로 모든 XDM 인스턴스는 유니코드로 쓰여집니다.
10. XML 데이터 및 LOB 데이터의 디폴트 경로는 기본 데이터 파일 경로입니다. 디폴트 XML 파일 기본 이름이 주 데이터 파일입니다. 디폴트 LOB 파일 기본 이름이 주 데이터 파일입니다. 예를 들어, 주 데이터 파일이

    ```
    /mypath/myfile.del
    ```

    이고 XML 데이터 및 LOB 데이터의 디폴트 경로는

    ```
    /mypath"
    ```

    이며 디폴트 XML 파일 기본 이름은

    ```
    myfile.del
    ```

    이고 디폴트 LOB 파일 기본 이름은 다음과 같은 경우,

    ```
    myfile.del
    ```

    LOB 파일을 생성하려면 LOBSINFILE 파일 유형 수정자를 지정해야 합니다.
11. 익스포트 유틸리티는 각 LOB 파일이나 XML 파일에 숫자 ID를 추가합니다. ID는 0으로 패드된 시퀀스 값인 3자리 숫자로 시작하며

    ```
    .001
    ```

    에서 시작합니다. 999번째 LOB 파일 또는 XML 파일 이후 ID는 더 이상 0으로 패드되지 않습니다. 예를 들어, 1000번째 LOG 파일이나 XML 파일의 확장자는

    ```
    .1000
    ```

    입니다. 숫자 ID 다음에는 데이터 유형을 나타내는 세 자 유형 ID가 표시됩니다(

    ```
    .lob
    ```

    또는

.xml을 나타냄)

. 예를 들어, 생성된 LOB 파일 이름의 형식은

```
myfile.del.001.lob
```

이고 생성된 XML 파일 이름의 형식은 다음과 같습니다.

```
myfile.del.001.xml
```

12. XQuery를 지정하여 잘 양식화된 문서가 아닌 익스포트 유틸리티 익스포트 XDM 인스턴스를 갖는 것이 가능합니다. 그러나 XML 컬럼은 완전한 문서만을 포함하므로, 해당 익스포트된 문서를 직접 XML 컬럼에 임포트하거나 로드할 수 없습니다.

db2Export - 데이터베이스에서 데이터 익스포트

데이터베이스의 데이터를 여러 외부 파일 형식 중 하나로 익스포트합니다. 사용자는 SQL SELECT문을 제공하거나 유형이 지정된 테이블에 대한 계층 정보를 제공하여 익스포트할 데이터를 지정합니다.

권한 부여

다음 중 하나가 필요합니다.

- *dataaccess* 권한
- 참여 중인 각 테이블 또는 뷰에 대한 CONTROL 또는 SELECT 특권

레이블 기반 액세스 제어(LBAC)가 이 함수에 강제로 사용됩니다. LBAC로 데이터가 보호되는 경우 익스포트되는 데이터는 호출자의 LBAC 증명서로 제한됩니다.

필수 연결

데이터베이스. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Export (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
```

```
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2ExportStruct;

typedef SQL_STRUCTURE db2ExportIn
{
   db2Uint16 *piXmlSaveSchema;
} db2ExportIn;

typedef SQL_STRUCTURE db2ExportOut
{
   db2Uint64 oRowsExported;
} db2ExportOut;

SQL_API_RC SQL_API_FN
  db2gExport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2gExportStruct;
```

db2Export API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ExportStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ExportStruct 데이터 구조 매개변수

piDataFileName

입력. 데이터가 익스포트되는 외부 파일의 경로 및 이름이 포함된 문자열.

piLobPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 LOB 파일이 저장되는 클라이언트 경로를 나열하는 sqlu_media_list 구조의 포인터. 익스포트되는 LOB는 sqlu_media_entry 구조에 나열되는 모든 경로에서 균등하게 분산됩니다.

piLobFileList

입력. 해당 media_type 필드가 SQLU_CLIENT_LOCATION으로 설정되고 해당 sqlu_location_entry 구조에는 기본 파일 이름이 포함되는 sqlu_media_list 구조의 포인터.

목록의 첫 번째 이름을 사용하여 이름 스페이스가 모두 사용된 경우 API는 두 번째 이름을 사용하며 계속 이와 같이 합니다. 익스포트 조작 중에 LOB 파일을 작성할 경우, 파일 이름은 이 목록에서 현재 기본 이름을 현재 경로(piLobPathList에서)에 추가하고 3자의 시퀀스 번호를 .lob 확장에 추가하여 작성됩니다. 예를 들어 현재 LOB 경로가 /u/foo/lob/path 디렉토리이고 현재 LOB 파일 이름은 bar이며 LOBSINSEPFILES 파일 유형 수정자가 설정된 경우 작성된 LOB 파일은 /u/foo/LOB/path/bar.001.lob, /u/foo/LOB/path/bar.002.lob 등이 됩니다. LOBSINSEPFILES 파일 유형 수정자가 설정되지 않은 경우 모든 LOB 문서는 병합되어 한 개의 /u/foo/lob/path/bar.001.lob 파일에 포함됩니다.

piDataDescriptor

입력. 출력 파일의 컬럼 이름을 지정하는 sqldcol 구조의 포인터. dcolmeth 필드 값으로 이 매개변수에서 제공되는 정보의 나머지를 익스포트 유틸리티가 해석하는 방법을 판별합니다. 이 매개변수의 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQL_METH_N

이름. 출력 파일에 사용할 컬럼 이름을 지정합니다.

SQL_METH_D

디폴트. 테이블의 기존 컬럼 이름이 출력 파일에 사용됩니다. 이 경우 컬럼 수 및 컬럼 스펙 배열 모두 무시됩니다. 컬럼 이름은 piActionString에 지정된 SELECT 문 출력에서 파생됩니다.

piActionString

입력. 유효한 동적 SQL SELECT 문을 포함하는 sqllob 구조의 포인터. 구조

에는 뒤에 SELECT 문을 구성하는 문자가 오는 4바이트 길이의 필드입니다. SELECT 문은 데이터베이스에서 추출되어 외부 파일로 쓰여지는 데이터를 지정합니다.

외부 파일의 컬럼(piDataDescriptor의) 및 SELECT 문의 데이터베이스 컬럼은 해당하는 각 목록/구조 위치에 따라 일치됩니다. 데이터베이스에서 선택한 데이터의 첫 번째 컬럼은 외부 파일의 첫 번째 컬럼이 되고 해당 컬럼 이름은 외부 컬럼 배열의 첫 번째 요소에서 사용됩니다.

piFileType

입력. 외부 파일 내의 데이터 형식을 나타내는 문자열. 지원되는 외부 파일 형식(sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQL_DEL

컬럼 식별자가 있는 ASCII, dBase, BASIC 및 IBM® Personal Decision Series 프로그램과 교환에 사용되는 다수의 데이터베이스 관리 프로그램 및 파일 관리자

SQL_WSF

Lotus Symphony 및 1-2-3 프로그램과의 교환을 위한 워크시트 형식(WSF). 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

SQL_IXF

통합 교환 형식의 PC 버전으로 테이블에서 데이터를 익스포트할 때 자주 사용되는 방법입니다. 이 파일 형식으로 익스포트된 데이터는 이후에 동일한 테이블이나 다른 데이터베이스 관리 프로그램 테이블로 임포트 또는 로드할 수 있습니다.

piFileTypeMod

입력. 하나 이상의 처리 옵션을 지정하는 문자 배열이 뒤에 오는 2바이트 길이의 필드를 포함하는 sqldcol 구조의 포인터. 이 포인터가 NULL이거나 지시된 구조에 영(0) 문자가 있는 경우 이 조치는 디폴트 스펙 선택으로 해석됩니다.

지원되는 모든 파일 유형에 모든 옵션을 사용할 수 있는 것은 아닙니다. 아래의 관련 링크인 "익스포트 유틸리티를 위한 파일 유형 수정자"를 참조하십시오.

piMsgFileName

입력. 유틸리티에서 리턴된 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우 정보가 추가됩니다. 없는 경우에는 파일이 작성됩니다.

iCallerAction

입력. 호출자가 요청한 조치. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_INITIAL

초기 호출. 이 값을 API의 첫 번째 호출에 사용해야 합니다. 초기 호출이나 이후에 임의의 후속 호출이 리턴되고 요청된 익스포트 조작을 완료하기 전에 일부 조치를 수행하는 응용프로그램을 호출해야 하는 경우 호출자 조치는 다음 중 하나로 설정해야 합니다.

SQLU_CONTINUE

처리를 계속합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 완료되고 유틸리티는 초기 요청 처리를 계속하도록 지정합니다.

SQLU_TERMINATE

처리를 종료합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 수행되지 않고 유틸리티가 초기 요청 처리를 종료하도록 지정합니다.

poExportInfoOut

db2ExportOut 구조의 포인터

piExportInfoIn

입력. db2ExportIn 구조의 포인터

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 XML 파일이 저장되는 클라이언트 경로를 나열하는 sqlu_media_list 구조의 포인터. 익스포트되는 XML은 sqlu_media_entry 구조에 나열되는 모든 경로에서 균등하게 분산됩니다.

piXmlFileList

입력. 해당 media_type 필드가 SQLU_CLIENT_LOCATION으로 설정되고 해당 sqlu_location_entry 구조에는 기본 파일 이름이 포함되는 sqlu_media_list 구조의 포인터.

목록의 첫 번째 이름을 사용하여 이름 스페이스가 모두 사용된 경우 API는 두 번째 이름을 사용하며 계속 이와 같이 합니다. 익스포트 조작 중에 XML 파일을 작성할 경우, 파일 이름은 이 목록에서 현재 기본 이름을 현재 경로(piXmlFileList에서)에 추가하고 3자의 시퀀스 번호를 .xml 확장에 추가하여 작성됩니다. 예를 들어 현재 XML 경로가 /u/foo/xml/path 디렉토리이고 현재 XML 파일 이름은 bar이며 XMLINSEPFILES 파일 유형 수정자가 설정된 경우 작성된 XML 파일은 /u/foo/xml/path/bar.001.xml, /u/foo/xml/path/bar.002.xml 등이 됩니다. XMLINSEPFILES 파일 유형 수정자가 설정되지 않은 경우 모든 XML 문서는 병합되어 한 개의 /u/foo/xml/path/bar.001.xml 파일에 포함됩니다.

db2ExportIn 데이터 구조 매개변수

piXmlSaveSchema

입력. 익스포트된 각 XML 문서가 익스포트된 데이터 파일에 저장되었는지 확인하는 데 사용되는 XML 스키마의 SQL ID를 표시합니다. 가능한 값은 참 및 거짓입니다.

db2ExportOut 데이터 구조 매개변수

oRowsExported

출력. 목표 파일에 익스포트된 레코드 수를 리턴합니다.

db2gExportStruct 데이터 구조 특정 매개변수

iDataFileNameLen

입력. 데이터 파일 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수.

iFileTypeLen

입력. 파일 유형의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수.

iMsgFileNameLen

입력. 메시지 파일 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수.

사용 시 참고사항

익스포트 조작을 시작하기 전에 다음 중 하나의 방법을 사용하여 모든 테이블 조작을 완료하고 모든 잠금을 해제해야 합니다.

- WITH HOLD 절로 정의된 모든 열려 있는 커서를 닫고 COMMIT 문을 실행하여 데이터 변경을 커미트하십시오.
- ROLLBACK 문을 실행하여 데이터 변경을 롤백하십시오.

SELECT문에서 테이블 별명을 사용할 수 있습니다.

메시지 파일에 넣어진 메시지는 메시지 검색 서비스에서 리턴된 정보를 포함합니다. 각 메시지는 새 행에서 시작합니다.

익스포트 유틸리티에서 경고가 작성되면 메시지가 메시지 파일에 기록되거나 해당 파일이 없는 경우 표준 출력으로 기록됩니다.

외부 컬럼 이름 배열 piDataDescriptor의 컬럼 수(sqldcol 구조의 dcolnum 필드)가 SELECT 문으로 생성된 컬럼 수와 동일하지 않는 경우 경고 메시지가 발행됩니다. 이 경우, 외부 파일에 기록되는 컬럼 수는 두 수 중 작은 수입니다. 초과 데이터베이스 컬럼이나 외부 컬럼 이름은 출력 파일 생성에 사용되지 않습니다.

db2uexpm.bnd 모듈이나 다른 모든 제공된 .bnd 파일은 수동으로 바인드되고 바인더의 형식 옵션은 사용하면 안됩니다.

DB2 Connect를 사용하여 z/OS® 및 OS/390용 DB2, VM 및 VSE용 DB2 및 System i®용 DB2와 같은 DRDA 서버에서 테이블을 익스포트하는 데 사용할 수 있습니다. PC/IXF 익스포트만 지원됩니다.

데이터베이스 간에 데이터를 이동하려면 PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다.

익스포트 유틸리티는 AIX 시스템에서 호출될 때 다중 파트 PC/IXF 파일을 작성하지 않습니다.

단일 데이터베이스 테이블의 컨텐츠가 SELECT * FROM tablename으로 시작되는 piActionString 매개변수 및 디폴트 이름을 지정하는 piDataDescriptor 매개변수가 포함된 PC/IXF 파일로 익스포트되는 경우 테이블의 인덱스 정의가 PC/IXF 파일에 포함됩니다. piActionString의 SELECT 절에 조인이 포함된 경우 인덱스가 뷰에 대해서는 저장되지 않습니다. piActionString 매개변수의 WHERE 절, GROUP BY 절 또는 HAVING 절을 사용하여 인덱스는 저장됩니다. 이 모든 경우에 유형이 지정된 테이블에서 익스포트할 때 전체 계층 구조를 익스포트해야 합니다.

제공된 SELECT 문이 SELECT * FROM tablename 양식인 경우 익스포트 유틸리티는 테이블의 NOT NULL WITH DEFAULT 속성을 IXF 파일에 저장합니다.

유형이 지정된 테이블을 익스포트할 때 subselect문은 단지 목표 테이블 이름과 WHERE 절을 지정하여 표현할 수 있습니다. 계층 구조를 익스포트할 때는 Fullselect 및 select-statement를 지정할 수 없습니다.

IXF 이외의 파일 형식에서는 트래버스 순서 목록을 지정할 것을 권장합니다. 이는 이 순서가 DB2에게 계층 구조를 트래버스하는 방법과 어떤 서브테이블을 익스포트할지를 알려주기 때문입니다. 이 목록을 지정하지 않으면 해당 계층 구조에 있는 모든 테이블이 익스포트되고 디폴트 순서는 OUTER 순서가 됩니다. 대안은 OUTER 기능이 제공하는 순서인 디폴트 순서를 사용하는 방법입니다.

주: 임포트 조작 동안 동일한 트래버스 순서를 사용하십시오. 로드 유틸리티는 계층 구조 또는 하위 계층 구조를 지원하지 않습니다.

REXX™ API 구문

```
EXPORT :stmt TO datafile OF filetype
[MODIFIED BY :filetmod] [USING :dcoldata]
MESSAGES msgfile [ROWS EXPORTED :number]
```

```
CONTINUE EXPORT

STOP EXPORT
```

REXX API 매개변수

stmt 유효한 동적 SQL SELECT 문이 포함된 REXX 호스트 변수. 명령문은 데이터베이스에서 추출하는 데이터를 지정합니다.

datafile

데이터가 익스포트되는 파일 이름.

filetype

익스포트 파일의 데이터 형식. 지원되는 파일 형식은 다음과 같습니다.

DEL 컬럼 식별자가 있는 ASCII

WSF 워크시트 형식. 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

IXF 통합 교환 형식의 PC 버전

filetmod

추가 처리 옵션이 포함된 호스트 변수

dcoldata

익스포트 파일에서 사용되는 컬럼 이름이 포함된 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름입니다.

XXX.0

컬럼 수(나머지 변수의 요소 수)

XXX.1

첫 번째 컬럼 이름

XXX.2

두 번째 컬럼 이름

XXX.3

기타.

이 매개변수가 NULL이거나 dcoldata의 값을 지정하지 않은 경우 유틸리티는 데이터베이스 테이블의 컬럼 이름을 사용합니다.

msgfile

오류 및 경고 메시지를 보낸 파일, 경로 또는 디바이스 이름

number

익스포트한 행 번호가 포함된 호스트 변수

익스포트 세션 - CLP 예

예 1

다음 예에서는 SAMPLE 데이터베이스의 STAFF 테이블(사용자가 연결되어 있어야 함)에서 myfile.ixf로 정보를 익스포트하는 방법(출력은 IXF 형식으로 표시함)을 보여줍니다. DB2 Connect를 통해 데이터베이스가 연결되지 않은 경우 인덱스 정의(있는 경우)는 출력 파일에 저장됩니다. 그렇지 않으면 데이터만 저장됩니다.

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

예 2

다음 예에서는 SAMPLE 데이터베이스의 STAFF 테이블(사용자가 연결되어 있어야 함)에서 awards.ixf로 부서 20의 직원 정보를 익스포트하는 방법(출력은 IXF 형식으로 표시함)을 보여줍니다.

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
   where dept = 20
```

예 3

다음 예에서는 LOB를 DEL 파일로 익스포트하는 방법을 보여줍니다.

```
db2 export to myfile.del of del lobs to mylobs/
   lobfile lobs1, lobs2 modified by lobsinfile
   select * from emp_photo
```

예 4

다음 예에서는 LOB를 DEL 파일로 익스포트하는 방법을 보여줍니다. 이때 첫 번째 디렉토리에 모두 포함되지 않을 수 있는 파일에 대해 두 번째 디렉토리를 지정합니다.

```
db2 export to myfile.del of del
   lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
   select * from emp_photo
```

예 5

다음 예에서는 문자열 분리문자로 작은따옴표를, 컬럼 분리문자로 세미콜론을, 소수점으로 쉼표를 사용하여 데이터를 DEL 파일로 익스포트하는 방법을 보여줍니다. 데이터를 다시 데이터베이스로 임포트할 때도 동일한 규칙을 사용해야 합니다.

```
db2 export to myfile.del of del
   modified by chardel'' coldel; decpt,
   select * from staff
```

제 3 장 임포트 유틸리티

임포트 개요

임포트 유틸리티는 INSERT문을 사용하여 테이블, 유형이 지정된 테이블 또는 뷰에 데이터를 채웁니다. 임포트된 데이터를 수신하는 테이블 또는 뷰에 이미 데이터가 있으면 입력 데이터가 교체되거나 기존 데이터에 추가될 수 있습니다.

익스포트와 마찬가지로 임포트도 비교적 단순한 데이터 이동 유틸리티입니다. 제어 센터를 사용하거나 CLP 명령을 실행하거나 ADMIN_CMD 스토어드 프로시저를 호출하거나 사용자 응용프로그램을 통해 해당 API db2Import를 호출하여 활성화할 수 있습니다.

임포트에서 지원하는 데이터 형식 및 임포트에서 사용할 수 있는 기능은 다양합니다.

- 임포트에서는 IXF, WSF, ASC 및 DEL 데이터 형식을 지원합니다.
- 임포트에서는 파일 유형 수정자를 사용하여 임포트 조작을 사용자 정의할 수 있습니다.
- 임포트는 계층 데이터 및 유형이 지정된 테이블을 이동하는 데 사용될 수 있습니다.
- 임포트는 모든 활동을 로그하고 인덱스를 갱신하고 제한조건을 검증하고 트리거를 시작합니다.
- 임포트를 통해 데이터를 삽입할 테이블 또는 뷰 내에서 컬럼 이름을 지정할 수 있습니다.
- 임포트는 DB2 Connect에서 사용될 수 있습니다.

중요사항: WSF 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

임포트 모드

임포트에는 데이터를 임포트하는 방법을 판별하는 다섯 가지 모드가 있습니다. 처음 세 가지 INSERT, INSERT_UPDATE 및 REPLACE는 목표 테이블이 이미 있는 경우 사용됩니다. 세 모드에서는 모두 IXF, WSF, ASC 및 DEL 데이터 형식을 지원합니다. 그러나 INSERT 및 INSERT_UPDATE는 별칭을 통해 사용할 수 있습니다.

표 11. INSERT, INSERT_UPDATE 및 REPLACE 임포트 모드의 개요

모드	우수 사례 사용법
INSERT	기존 데이터를 변경하지 않고 목표 테이블에 입력 데이터를 삽입합니다.

표 11. INSERT, INSERT_UPDATE 및 REPLACE 임포트 모드의 개요 (계속)

모드	우수 사례 사용법
INSERT_UPDATE	입력 행 값과 1차 키 값이 일치하도록 행을 갱신합니다. 일치하는 행이 없으면 테이블에 임포트된 행이 삽입됩니다.
REPLACE	테이블 및 인덱스 정의를 보존하면서 모든 기존 데이터를 삭제하고 임포트된 데이터를 삽입합니다.

다른 두 개 모드, REPLACE_CREATE 및 CREATE는 목표 테이블이 없는 경우 사용됩니다. 입력 파일이 PC/IXF 형식인 경우에만 사용할 수 있습니다. 이때 여기에는 작성할 테이블의 구조화된 설명이 있습니다. 오브젝트 테이블에 자체 항목 이외의 종속 항목이 있는 경우 이 모드에서 임포트를 수행할 수 없습니다.

주: 임포트의 CREATE 및 REPLACE_CREATE 모드는 사용되지 않습니다. 대신 db2look 유틸리티를 사용합니다.

표 12. REPLACE_CREATE 및 CREATE 임포트 모드의 개요

모드	우수 사례 사용법
REPLACE_CREATE	테이블 및 인덱스 정의를 보존하면서 모든 기존 데이터를 삭제하고 임포트된 데이터를 삽입합니다. 목표 테이블 및 인덱스가 없으면 작성합니다.
CREATE	목표 테이블 및 인덱스를 작성합니다. 새 테이블이 작성된 테이블 스페이스의 이름을 지정할 수 있습니다.

임포트 작동 방식

임포트에 필요한 시간 및 단계 수는 이동할 데이터 크기 및 지정하는 옵션에 따라 다릅니다. 임포트 조작에서는 다음 단계를 수행합니다.

1. 테이블 잠금
   임포트에서는 테이블에 대한 동시 액세스가 허용되는지에 따라 기존 목표 테이블에서 배타적(X) 잠금 또는 비배타적(IX) 잠금을 획득합니다.
2. 데이터 찾기 및 검색
   임포트에서는 FROM절을 사용하여 입력 데이터를 찾습니다. 명령에서 XML 또는 LOB 데이터가 있다고 표시되면 임포트에서 이 데이터를 찾습니다.
3. 데이터 삽입
   임포트에서는 기존 데이터를 교체하거나 테이블에 새 데이터 행을 추가합니다.
4. 제한조건 확인 및 트리거 시작
   데이터가 작성되면 임포트에서는 삽입된 각 행이 목표 테이블에 정의된 제한조건을 준수하는지 확인합니다. 거부된 행에 대한 정보는 메시지 파일에 작성됩니다. 또한 임포트에서는 기존 트리거를 시작합니다.

5. 조작 커미트
   임포트에서는 변경 사항을 저장하고 목표 테이블에서 잠금을 해제합니다. 또한 임포트 중 정기적 수행을 지정할 수도 있습니다.

다음 항목은 기본 임포트 조작에 필수 사항입니다.

- 입력 파일의 경로 및 이름
- 목표 테이블 또는 뷰의 이름 또는 별명
- 입력 파일에서 데이터 형식
- 데이터를 임포트할 방법
- 계층 데이터 임포트 시 트래버스 순서
- 유형이 지정된 테이블 임포트 시 서브테이블 목록

추가 옵션

임포트 조작을 사용자 정의할 수 있는 여러 옵션이 있습니다. MODIFIED BY절에서 파일 유형 수정자를 지정하여 데이터 형식을 변경하고 데이터에서 수행할 작업을 임포트 유틸리티에 지시하고 성능을 향상시킬 수 있습니다.

디폴트로 임포트 유틸리티는 일부 ALLOW WRITE ACCESS 임포트를 제외하고 임포트가 성공적으로 종료될 때까지 커미트를 수행하지 않습니다. 그러면 임포트 속도가 향상되지만 동시성, 재시작 가능성 및 사용 중인 로그 공간 고려사항을 위해 임포트 중 커미트를 수행하도록 지정하는 것이 선호될 수도 있습니다. 이를 수행하는 한 가지 방법으로 **COMMITCOUNT** 매개변수를 "automatic"으로 설정합니다. 그러면 커미트를 수행해야 하는 시점을 내부적으로 판별하도록 임포트에 지시합니다. 또는 **COMMITCOUNT**를 특정 숫자로 설정할 수 있습니다. 그러면 지정된 레코드 수를 임포트할 때마다 커미트를 수행하도록 임포트에 지시합니다.

임포트 성능을 향상시키는 몇 가지 방법이 있습니다. 임포트 유틸리티가 Embedded SQL 응용프로그램이고 내부적으로 SQL 페치를 수행하면 SQL 조작에 적용되는 최적화는 임포트에도 적용됩니다. `compound` 파일 유형 수정자를 사용하여 디폴트로 한 행씩 삽입하는 방법 대신 한 번에 지정된 행 수를 삽입하도록 수행할 수 있습니다. 임포트 중 많은 경고가 생성될 것으로 예상되면(이 경우 조작 속도가 느려짐) `norowwarnings` 파일 유형 수정자를 지정하여 거부된 행에 대한 경고를 제외할 수 있습니다.

메시지 파일

임포트 중 해당 조작에 관한 오류, 경고 및 정보 메시지를 포함하도록 표준 ASCII 텍스트 메시지 파일이 작성됩니다. API db2Import를 통해 유틸리티가 호출되면 **MESSAGES** 매개변수 앞에서 이러한 파일의 이름을 지정해야 합니다. 그렇지 않은 경우 선택적입니다. 메시지 파일은 임포트가 진행 중인 동안에 액세스할 수 있으므로 임포트 진행을 모니터링하는 편리한 방법입니다. 임포트 조작에 실패한 경우 성공적으로 임포트된 마지막 행을 표시하여 재시작 지점을 판별하도록 이 메시지 파일을 사용할 수 있습니다.

주: 리모트 데이터베이스에서 수행된 임포트 조작으로 생성된 출력 메시지 볼륨이 60KB를 초과하는 경우 유틸리티에서는 처음 30KB와 마지막 30KB를 보존합니다.

임포트를 사용하는 데 필요한 특권 및 권한

사용자는 특권을 사용하여 데이터베이스 자원을 작성하거나 자원에 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작과 특권을 그룹화하는 방법을 제공합니다. 이를 통해 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다.

적절히 권한이 부여된(즉, 필수 특권 또는 권한이 있음) 오브젝트에만 액세스할 수 있습니다.

DATAACCESS 권한이 있으면 모든 유형의 임포트 조작을 수행할 수 있습니다. 아래 테이블에서는 해당 유형의 임포트를 수행할 수 있도록 하는 각 참여 테이블, 뷰 또는 별칭에 대한 기타 권한을 보여줍니다.

표 13. 임포트 연산을 수행하기 위해 필요한 권한

모드	필수 권한
INSERT	CONTROL 또는 INSERT 및 SELECT
INSERT_UPDATE	CONTROL 또는 INSERT, SELECT, UPDATE 및 DELETE
REPLACE	CONTROL 또는 INSERT, SELECT 및 DELETE
REPLACE_CREATE	목표 테이블이 있는 경우: CONTROL 또는 INSERT, SELECT 및 DELETE 목표 테이블이 없는 경우: CREATETAB(데이터베이스), USE(테이블 스페이스) 및 스키마가 없는 경우: IMPLICIT_SCHEMA(데이터베이스) 또는 스키마가 있는 경우: CREATEIN(스키마)
CREATE	CREATETAB(데이터베이스), USE(테이블 스페이스) 및 스키마가 없는 경우: IMPLICIT_SCHEMA(데이터베이스) 또는 스키마가 있는 경우: CREATEIN(스키마)

주: IMPORT 명령의 **CREATE** 및 **REPLACE_CREATE** 옵션은 사용되지 않으며 추후 릴리스에서 제거됩니다.
마찬가지로 **REPLACE** 또는 **REPLACE_CREATE** 옵션을 테이블에서 사용하려면 테이블을 삭제할 권한이 세션 권한 부여 ID에 있어야 합니다.

계층 구조를 임포트하려는 경우 필수 권한은 모드에 따라 다릅니다. 기존 계층 구조인 경우 **REPLACE** 조작에 대해서는 계층 구조의 모든 서브테이블에 대한 CONTROL

특권이 있으면 됩니다. 없는 계층 구조인 경우 **REPLACE_CREATE** 조작에 대해서는 CREATETAB 및 USE와 함께 계층 구조의 모든 서브테이블에 대한 CONTROL 특권이 있으면 됩니다.

또한 테이블에 정의된 레이블 기반 액세스 제어(LBAC) 보안 레이블을 사용하여 테이블로 임포트하는 경우 몇 가지 고려사항이 있습니다. 보호 컬럼이 있는 테이블로 데이터를 임포트하려면 테이블의 모든 보호 컬럼에 대한 쓰기 액세스를 허용하는 LBAC 증명서가 세션 권한 부여 ID에 있어야 합니다. 보호 행이 있는 테이블로 데이터를 임포트하려면 테이블을 보호하는 보안 규정에 포함된 쓰기 액세스에 대한 보안 레이블이 세션 권한 부여 ID에 부여되어야 합니다.

데이터 임포트

임포트 유틸리티는 지원되는 파일 형식의 외부 파일에서 테이블, 계층 구조, 뷰 또는 별칭으로 데이터를 삽입합니다. 로드 유틸리티는 더 빠른 대안이지만 로드 유틸리티는 계층 구조 레벨에서의 데이터 로드를 지원하지 않습니다.

임포트 유틸리티를 호출하기 전에 데이터를 임포트할 데이터베이스에 연결하거나 내재적으로 연결할 수 있어야 합니다. 내재된 연결이 가능한 경우 디폴트 데이터베이스에 대한 연결이 설정됩니다. Linux, UNIX 또는 Windows 클라이언트용 DB2에서 Linux, UNIX 또는 Windows 데이터베이스 서버용 DB2로의 유틸리티 액세스는 DB2 Connect 게이트웨이 또는 루프백 환경이 아닌 엔진을 통해 직접 연결되어야 합니다. 유틸리티는 COMMIT 또는 ROLLBACK문을 실행하므로 임포트를 호출하기 전에 COMMIT문 또는 ROLLBACK 조작을 실행하여 모든 트랜잭션을 완료하고 모든 잠금을 해제해야 합니다.

주: IMPORT 명령의 CREATE 및 REPLACE_CREATE 옵션은 사용되지 않으며 추후 릴리스에서 제거됩니다.

임포트 유틸리티에 다음 제한사항이 적용됩니다.

- 기존 테이블이 종속 테이블의 외부 키에서 참조하는 1차 키를 포함하는 상위 테이블인 경우 해당 데이터는 교체할 수 없으며 오직 추가만 가능합니다.
- 즉시 새로 고침 모드에 정의된 구체화된 쿼리 테이블의 기본 테이블로 임포트 교체 조작을 수행할 수 없습니다.
- 구조화된 유형 컬럼을 포함하는 테이블, 요약 테이블 또는 시스템 테이블로 데이터를 임포트할 수 없습니다.
- 선언된 임시 테이블로 데이터를 임포트할 수 없습니다.
- 임포트 유틸리티를 통해 뷰를 작성할 수 없습니다.
- PC/IXF 파일에서 테이블을 작성할 때 참조 제한조건 및 외부 키 정의는 보존되지 않습니다. 이전에 SELECT *를 사용하여 데이터를 익스포트한 경우 1차 키 정의는 보존됩니다.

- 임포트 유틸리티는 고유한 SQL문을 생성하므로 일부 경우 최대 명령문 크기(2MB)가 초과될 수 있습니다.
- CREATE 또는 REPLACE_CREATE 임포트 옵션을 사용하여 다차원적으로 클러스터된(MDC) 테이블 또는 파티션된 테이블을 재작성할 수 없습니다.
- XML 컬럼을 포함하는 테이블을 재작성할 수 없습니다.
- 암호화된 데이터를 임포트할 수 없습니다.
- 임포트 교체 조작에서 Not Logged Initially절을 고려하지 않습니다. IMPORT 명령의 REPLACE 옵션은 CREATE TABLE문의 NLI(NOT LOGGED INITIALLY) 절 또는 ALTER TABLE문의 ACTIVATE NOT LOGGED INITIALLY절을 인식하지 않습니다. NLI절이 호출된 CREATE TABLE 또는 ALTER TABLE문과 동일한 트랜잭션에서 REPLACE 조치를 사용하는 임포트를 수행하면 임포트에서 NLI 절을 인식하지 못합니다. 모든 삽입은 로그됩니다.

  일시적인 해결책 1: DELETE문을 사용하여 테이블의 컨텐츠를 삭제하고 INSERT 문으로 임포트를 호출합니다.

  일시적인 해결책 2: 테이블을 삭제하고 재작성한 다음, INSERT문으로 임포트를 호출합니다.

다음 제한사항은 임포트 유틸리티에 적용됩니다. 리모트 데이터베이스에서 수행된 임포트 조작으로 생성된 출력 메시지 볼륨이 60KB를 초과하는 경우 유틸리티에서는 처음 30KB와 마지막 30KB를 보존합니다.

임포트 유틸리티는 명령행 처리기(CLP), 제어 센터의 테이블 임포트 노트북을 통해 또는 클라이언트 응용프로그램에서 API db2Import를 호출하여 호출할 수 있습니다.

테이블 임포트 노트북 사용

1. 제어 센터에서 테이블 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 테이블 폴더를 누르십시오. 기존 테이블은 창 오른쪽의 분할 영역(컨텐츠 영역)에 표시됩니다.
3. 컨텐츠 영역에서 원하는 테이블을 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 임포트를 선택하십시오. 테이블 임포트 노트북이 열립니다.

테이블 임포트 노트북에 대한 세부사항 정보는 제어 센터 온라인 도움말 기능에서 제공됩니다.

CLP를 통해 IMPORT 명령 실행

단순하게 임포트 조작을 수행하려면 입력 파일, 파일 형식, 임포트 모드 및 목표 테이블(또는 작성할 테이블 이름)만 지정하면 됩니다.

예를 들어 CLP에서 데이터를 임포트하려면 IMPORT 명령을 입력하십시오.

```
db2 import from filename of fileformat
import_mode into table
```

여기서 filename은 임포트할 데이터를 포함하는 입력 파일 이름이고 ixf는 파일 형식이며 insert는 모드이고 table은 데이터를 삽입할 테이블 이름입니다.

그러나 경고 및 오류 메시지를 작성할 메시지 파일도 지정할 수 있습니다. 이를 수행하려면 **MESSAGES** 매개변수 및 메시지 파일 이름을 추가합니다. 명령은 다음과 같습니다.

```
db2 import from filename of fileformat messages messagefile
import_mode into table
```

전체 구문 및 사용법 정보는 ″IMPORT″ 명령을 참조하십시오.

XML 데이터 임포트

Linux, UNIX 및 Windows용 DB2 데이터베이스 소스 데이터 오브젝트의 별칭 또는 테이블 이름을 사용하여 XML 테이블 컬럼에 XML 데이터를 임포트하는 데 임포트 유틸리티를 사용할 수 있습니다.

XML 테이블 컬럼에 데이터를 임포트하는 경우 XML FROM 옵션을 사용하여 입력 XML 데이터 파일의 경로를 지정할 수 있습니다. 예를 들어, 이전에 익스포트된 XML 파일 ″/home/user/xmlpath/xmldocs.001.xml″의 경우 다음 명령을 사용하여 데이터를 테이블에 다시 임포트할 수 있습니다.

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

스키마에 대해 삽입된 문서 유효성 확인

XMLVALIDATE 옵션을 사용하면 XML 문서 임포트 시 XML 스키마에 대해 문서의 유효성을 확인할 수 있습니다. 다음 예에서는 XML 문서 익스포트 시 저장된 스키마 정보와 대조하여 수신 XML 문서의 유효성을 확인합니다.

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
        USING XDS INSERT INTO USER.T1
```

구문 분석 옵션 지정

XMLPARSE 옵션을 사용하여 임포트된 XML 문서의 공백을 유지할지 또는 제거할지 여부를 지정할 수 있습니다. 다음 예에서는 XML 문서 익스포트 시 저장된 XML 스키마 정보와 대조하여 임포트된 모든 XML 문서의 유효성을 확인하며, 공백을 유지하고 해당 문서를 구문 분석합니다.

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
   WHITESPACE XMLVALIDATE USING XDS INSERT INTO USER.T1
```

임포트된 테이블 재작성

임포트 유틸리티의 CREATE 모드를 사용하여 익스포트 유틸리티로 저장한 테이블을 재작성할 수 있습니다. 그러나 입력 테이블의 속성이 보유되지 않으므로 프로세스에 여러 제한사항이 있습니다.

테이블을 재작성할 수 있도록 임포트를 수행하려면 익스포트 조작에서 몇 가지 요구사항을 만족해야 합니다. 원래 테이블은 IXF 파일로 익스포트되어야 합니다. DEL 또는 ASC 파일 형식으로 파일을 익스포트하는 경우 출력 파일에는 목표 테이블에 대한 설명이 포함되지 않지만 레코드 데이터는 포함됩니다. 이러한 파일 형식으로 데이터가 저장된 테이블을 재작성하려면 목표 테이블을 작성하고 로드 또는 임포트 유틸리티를 사용하여 이러한 파일의 테이블을 채우십시오. db2look 유틸리티를 사용하여 원래 테이블 정의를 캡처하고 해당 데이터 정의 언어(DDL)를 생성할 수 있습니다. 마찬가지로, 익스포트 중 사용된 SELECT문만 특정 조치 문자열을 포함할 수 있습니다. 예를 들어 SELECT절에서는 컬럼 이름을 사용할 수 없으며 SELECT *만 허용됩니다.

주: 임포트의 CREATE 모드는 사용되지 않습니다. db2look 유틸리티를 사용하여 테이블을 캡처 및 재작성합니다.

보유 속성

재작성된 테이블은 원래 테이블의 다음 속성을 보유합니다.

- 1차 키 이름 및 정의
- 컬럼 정보. 다음을 포함합니다.
  - 컬럼 이름
  - 열 데이터 유형(기본 유형으로 보존된 사용자 정의 구별 유형 포함)
  - ID 등록 정보
  - 길이(lob_file 유형 제외)
  - 코드 페이지(해당되는 경우)
  - ID 옵션
  - 컬럼이 널(NULL) 입력 가능 또는 널(NULL) 입력 불가능으로 정의되었는지 여부
  - 상수에 대한 디폴트값(있는 경우). 단, 디폴트값의 기타 유형은 해당되지 않습니다.
- 인덱스 정보. 다음을 포함합니다.
  - 인덱스 이름
  - 인덱스 작성자 이름
  - 컬럼 이름 및 각 컬럼의 정렬 기준(오름차순 또는 내림차순)
  - 인덱스가 고유하게 정의되었는지 여부
  - 인덱스가 클러스터되었는지 여부

- 인덱스에서 역 스캔이 허용되는지 여부
- PCTFREE 값
- MINPCTUSED 값

주: 인덱스의 컬럼 이름에 - 또는 + 문자가 있는 경우 인덱스 정보는 보유되지 않으며, 이 경우 SQL27984W가 리턴됩니다.

유실된 속성

재작성된 테이블은 다음을 포함하여 원래 테이블의 여러 속성을 보유하지 않습니다.

- 소스 유형(일반 테이블, 구체화된 쿼리 테이블(MQT), 뷰 또는 이 모든 소스이거나 이중 하나의 소스에 있는 컬럼 세트)
- 고유 제한조건 및 기타 유형의 제한조건 또는 트리거(1차 키 제한조건은 포함되지 않음)
- 테이블 정보. 다음을 포함합니다.
  - MQT 정의(해당되는 경우)
  - MQT 옵션(해당되는 경우)
  - 테이블 스페이스 옵션. 그러나 이 정보는 IMPORT 명령을 통해 지정될 수 있습니다.
  - 다차원 클러스터링(MDC) 차원
  - 파티션된 테이블 차원
  - 테이블 파티셔닝 키
  - NOT LOGGED INITIALLY 등록 정보
  - 점검 제한조건
  - 테이블 코드 페이지
  - 보호 테이블 등록 정보
  - 테이블 또는 값 압축 옵션
- 컬럼 정보. 다음을 포함합니다.
  - 상수 값을 제외한 디폴트값
  - LOB 옵션(있는 경우)
  - XML 등록 정보
  - CREATE TABLE문의 참조 절(있는 경우)
  - 참조 제한조건(있는 경우)
  - 점검 제한조건(있는 경우)
  - 생성된 컬럼 옵션(있는 경우)
  - 데이터베이스 범위 시퀀스에 종속된 컬럼
- 인덱스 정보. 다음을 포함합니다.

- INCLUDE 컬럼(있는 경우)
- 인덱스 이름(인덱스가 1차 키 인덱스인 경우)
- 키의 내림차순(인덱스가 1차 키 인덱스인 경우). 디폴트는 오름차순입니다.
- 16진수 값 0x2B 또는 0x2D가 포함된 인덱스 컬럼 이름
- 코드 페이지 변환 후 길이가 128바이트를 초과하는 인덱스 이름
- PCTFREE2 값
- 고유 제한조건

주: 이 목록은 완전하지 않으므로 주의하여 사용합니다.

임포트에 실패하고 SQL3311N이 리턴되면 파일 유형 수정자 forcecreate를 사용하여 계속 테이블을 재작성할 수 있습니다. 이 수정자를 사용하면 정보가 누락되거나 제한된 테이블을 작성할 수 있습니다.

유형이 지정된 테이블 임포트 고려사항

임포트 유틸리티는 데이터의 사전 존재하는 계층 구조를 보존하면서 유형이 지정된 테이블에서 데이터를 이동하는 데 사용될 수 있습니다. 원하는 경우 임포트를 사용하여 테이블 계층 및 자료형 계층을 작성할 수도 있습니다.

유형이 지정된 테이블의 한 계층 구조에서 다른 계층 구조로 데이터를 이동하는 작업은 특정 트래버스 순서로 수행되며 익스포트 조작 중 중간 플랫 파일이 작성됩니다. 그 다음에, 임포트 유틸리티는 CREATE, INTO *table-name*, UNDER 및 AS ROOT TABLE 매개변수를 사용하여 이동할 계층 구조의 크기 및 배치를 제어합니다. 또한 임포트에서는 목표 데이터베이스에 있는 내용도 판별합니다. 예를 들어 각 서브테이블 이름 끝에 속성 목록을 지정하여 목표 데이터베이스로 이동된 속성을 제한할 수 있습니다. 속성 목록이 사용되지 않으면 각 서브테이블의 모든 컬럼이 이동됩니다.

테이블 재작성

수행할 수 있는 임포트 유형은 입력 파일의 파일 형식에 따라 다릅니다. ASC, DEL 또는 WSF 데이터에 대한 작업을 수행하는 경우 데이터를 임포트하려면 목표 테이블 또는 계층 구조가 존재해야 합니다. 그러나 PC/IXF 파일의 데이터는 테이블 또는 계층 구조가 아직 없어도 임포트 CREATE 조작을 지정하면 임포트할 수 있습니다. CREATE 옵션을 지정하면 임포트할 때 서브테이블 정의를 변경할 수 없다는 점에 유의하십시오.

트래버스 순서

입력 파일에 포함된 트래버스 순서로 데이터의 계층 구조를 유지할 수 있습니다. 따라서 익스포트 유틸리티 및 임포트 유틸리티를 호출할 때 동일한 트래버스 순서를 사용해야 합니다.

PC/IXF 파일 형식의 경우 목표 서브테이블 이름을 지정하기만 하고 파일에 저장된 디폴트 트래버스 순서를 사용해야 합니다.

유형이 지정된 테이블에서 CREATE 이외의 옵션을 사용하면 트래버스 순서 목록을 통해 트래버스 순서를 지정할 수 있습니다. 이 사용자 지정 트래버스 순서는 익스포트 조작 중에 사용한 순서와 일치해야 합니다. 임포트 유틸리티는 다음 경우에 목표 데이터베이스로 데이터를 정확히 이동하도록 보장합니다.

- 소스 및 목표 데이터베이스에서 서브테이블의 정의가 동일함
- 소스 및 목표 데이터베이스에서 서브테이블 사이의 계층 관계가 동일함
- 트래버스 순서가 동일함

트래버스 순서를 정의할 때 계층 구조 아래 경로 및 시작 지점을 판별해도 경로의 다음 분기를 시작하려면 각 분기를 끝으로 트래버스해야 합니다. 임포트 유틸리티는 지정된 트래버스 순서로 이 조건 위반을 검색합니다.

예

이 절의 예에서는 유효한 트래버스 순서 4개를 사용하는 다음과 같은 계층 구조에 기반합니다.

- Person, Employee, Manager, Architect, Student
- Person, Student, Employee, Manager, Architect
- Person, Employee, Architect, Manager, Student
- Person, Student, Employee, Architect, Manager

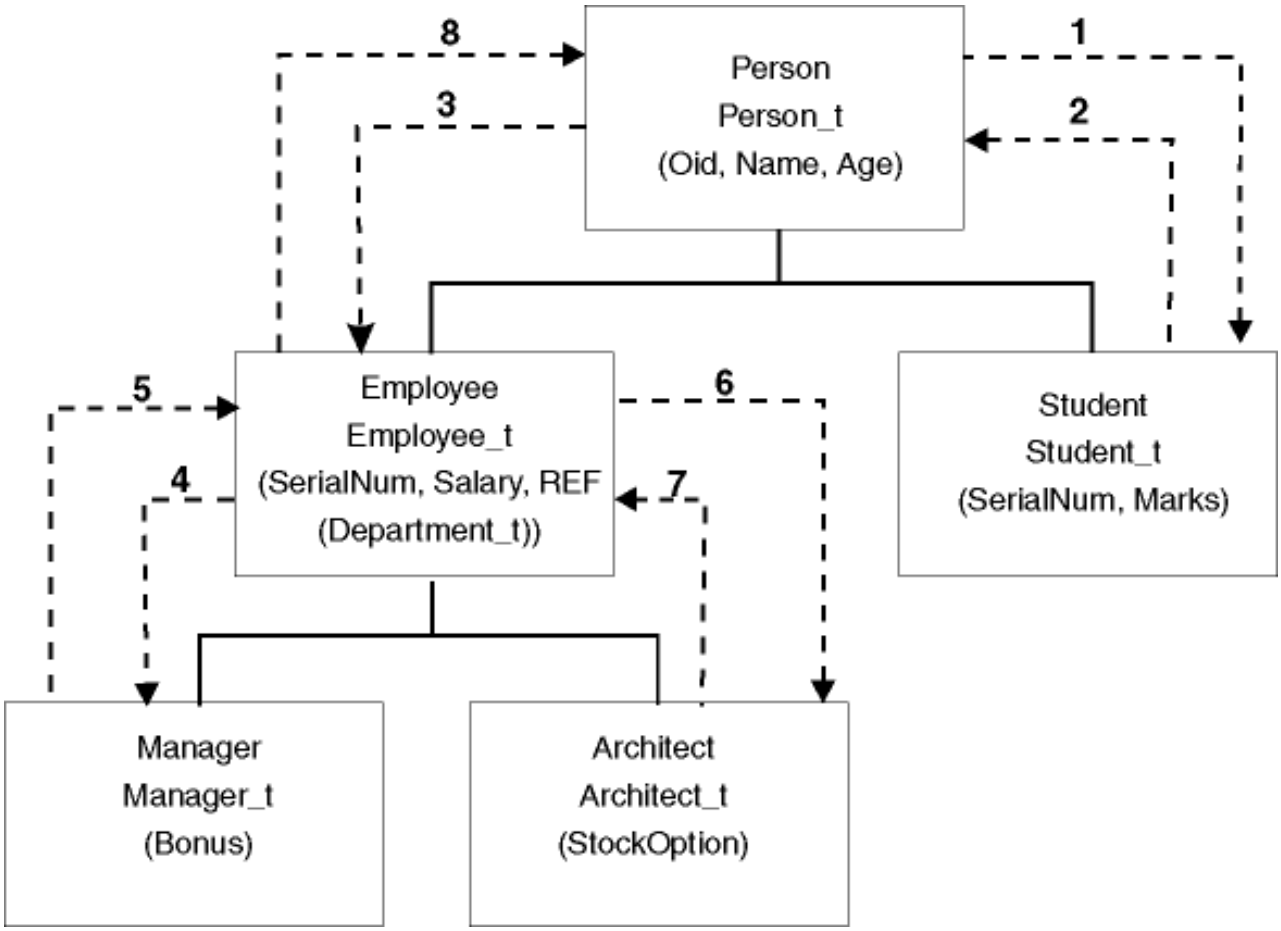


그림 2. 계층 구조 예

예 1

전체 계층 구조(이전 익스포트 조작으로 작성된 데이터 파일 `entire_hierarchy.ixf`에 포함됨)를 재작성하려면 다음 명령을 입력합니다.

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.ixf OF IXF CREATE INTO
   HIERARCHY STARTING Person AS ROOT TABLE
```

유형이 없으면 계층 구조의 각 유형이 작성됩니다. 이러한 유형이 이미 있으면 소스 데이터베이스와 동일한 정의가 목표 데이터베이스에 있어야 합니다. 동일하지 않으면 SQL 오류(SQL20013N)가 리턴됩니다. 새 계층 구조를 작성한 후 목표 데이터베이스(Target_db)로 이동할 데이터 파일에 정의된 서브테이블은 존재하지 않습니다. 소스 데이터베이스 계층 구조의 각 테이블이 작성됩니다. 소스 데이터베이스의 데이터가 목표 데이터베이스의 올바른 서브테이블로 임포트됩니다.

예 2

소스 데이터베이스의 전체 계층 구조를 재작성하고 목표 데이터베이스로 임포트하는 동시에, 선택한 데이터만 보존하려면 다음 명령을 입력합니다.

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.del OF DEL INSERT INTO (Person,
   Employee(Salary), Architect) IN HIERARCHY (Person, Employee,
   Manager, Architect, Student)
```

목표 테이블 PERSON, EMPLOYEE 및 ARCHITECT가 모두 있어야 합니다. 데이터는 PERSON, EMPLOYEE 및 ARCHITECT 서브테이블로 임포트됩니다. 즉, 다음이 임포트됩니다.

- PERSON에서 모든 컬럼이 PERSON으로
- PERSON 및 EMPLOYEE의 SALARY에서 모든 컬럼이 EMPLOYEE로
- PERSON, EMPLOYEE의 SALARY 및 ARCHITECT의 모든 컬럼이 ARCHITECT로

컬럼 SerialNum 및 REF(Employee_t)는 EMPLOYEE 또한 서브테이블로 임포트되지 않습니다. 즉, ARCHITECT가 임포트된 데이터를 포함하는 유일한 서브테이블입니다.

주: ARCHITECT는 EMPLOYEE의 서브테이블이며 EMPLOYEE에 지정된 유일한 임포트 컬럼은 SALARY이므로 SALARY도 ARCHITECT로 임포트되는 유일한 직원 특정 컬럼입니다. 즉, SerialNum 또는 REF(Employee_t) 컬럼은 EMPLOYEE 또는 ARCHITECT 행으로 임포트되지 않습니다.
MANAGER 및 STUDENT 테이블의 데이터는 임포트되지 않습니다.

예 3

이 예에서는 일반 테이블에서 익스포트하고 계층 구조의 단일 서브테이블로 임포트하는 방법을 보여줍니다. EXPORT 명령은 일반(유형이 없음) 테이블에서 작동하므로 데이터 파일에 Type_id 컬럼은 없습니다. no_type_id 파일 유형 수정자는 이를 표시하는 데 사용됩니다. 그러면 임포트 유틸리티에서 첫 번째 컬럼을 Type_id 컬럼으로 예상하지 않습니다.

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO Student_sub_table.del OF DEL SELECT * FROM
   Regular_Student
DB2 CONNECT TO Target_db
DB2 IMPORT FROM Student_sub_table.del OF DEL METHOD P(1,2,3,5,4)
   MODIFIED BY NO_TYPE_ID INSERT INTO HIERARCHY (Student)
```

이 예에서 목표 테이블 STUDENT가 존재해야 합니다. STUDENT가 서브테이블이므로 no_type_id 수정자를 사용하여 첫 번째 컬럼에 Type_id가 없음을 표시합니다. 그러나 STUDENT 테이블에 있는 다른 모든 속성 이외에도 기존 Object_id 컬럼이 있는지 확인해야 합니다. Object-id는 STUDENT 테이블로 임포트할 각 행의 첫 번째 컬럼으로 예상됩니다. METHOD절은 마지막 두 속성의 순서를 역으로 바꿉니다.

LBAC 보호 데이터 임포트 고려사항

보호된 행을 포함하는 테이블로 임포트하려면 레이블 기반 액세스 제어(LBAC) 증명서가 필요합니다. 또한 유효한 보안 레이블 또는 목표 테이블과 연관된 현재 보안 규정에 대해 유효한 레이블로 변환할 수 있는 보안 레이블을 제공해야 합니다.

유효한 LBAC 증명서가 없으면 임포트에 실패하고 오류(SQLSTATE 42512)가 리턴됩니다. 입력 데이터에 보안 레이블이 없거나 해당 보안 레이블이 내부 2진 형식이 아니면 여러 파일 유형 수정자를 사용하여 임포트를 진행할 수 있습니다.

보호된 행을 포함하는 테이블로 데이터를 임포트하는 경우 목표 테이블에 데이터 유형이 DB2SECURITYLABEL인 컬럼이 하나 있습니다. 데이터의 입력 행에 해당 컬럼 값이 없으면 import 명령에 usedefaults 파일 유형 수정자가 지정되지 않는 한, 해당 행은 거부됩니다. 이 경우 테이블을 보호하는 보안 규정의 쓰기 액세스에 대해 보유한 보안 레이블이 사용됩니다. 쓰기 액세스에 대한 보안 레이블을 보유하지 않은 경우 행이 거부되고 다음 행으로 처리가 계속됩니다.

보호된 행을 포함하는 테이블로 데이터를 임포트하고 입력 데이터에 데이터 유형이 DB2SECURITYLABEL인 컬럼의 값이 포함된 경우 해당 테이블로 데이터를 삽입할 때와 동일한 규칙이 적용됩니다. 임포트할 행을 보호하는 보안 레이블(데이터 파일의 해당 행에 있는 항목)이 쓰기 권한이 있는 항목인 경우 해당 보안 레이블을 사용하여 행을 보호합니다. 즉, 데이터 유형이 DB2SECURITYLABEL인 컬럼에 작성됩니다. 해당 보안 레이블로 보호되는 행에 대한 쓰기 권한이 없는 경우 소스 테이블을 보호하는 보안 규정이 작성된 방식에 따라 상황이 달라집니다.

- 규정을 작성한 CREATE SECURITY POLICY문에 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션이 포함된 경우 삽입에 실패하고 오류가 리턴됩니다.
- CREATE SECURITY POLICY문이 옵션을 포함하지 않거나 대신 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 포함하는 경우 해당 행에 대한 데이터 파일의 보안 레이블은 무시되고 쓰기 액세스에 대해 보유한 보안 레이블을 사용하여 해당 행을 보호합니다. 이 경우 오류나 경고는 발행되지 않습니다. 쓰기 액세스에 대한 보안 레이블을 보유하지 않은 경우 행이 거부되고 다음 행으로 처리가 계속됩니다.

분리문자 고려사항

데이터 유형이 DB2SECURITYLABEL인 컬럼으로 데이터를 임포트하는 경우 데이터 파일의 값은 디폴트로 해당 보안 레이블의 내부 표현을 구성하는 실제 바이트로 가정합니다. 그러나 일부 행 데이터는 IMPORT 명령에서 행 분리로 잘못 해석될 수 있는 개행 문자를 포함할 수 있습니다. 이 문제점이 발생한 경우 delprioritychar 파일 유형 수정자를 사용하여 문자 분리문자가 행 분리문자보다 우선되도록 해야 합니다. delprioritychar을 사용하는 경우 문자 분리문자에 포함된 레코드 또는 컬럼 분리문

자는 분리문자로 인식되지 않습니다. delprioritychar 파일 유형 수정자는 개행 문자를 포함하는 값이 없는 경우에도 안전하게 사용할 수 있습니다. 그러나 임포트 속도가 약간 느려질 수 있습니다.

임포트할 데이터가 ASC 형식인 경우 임포트된 보안 레이블 및 보안 레이블 이름에 뒤 공백이 포함되지 않도록 하기 위해 추가 단계를 수행할 수도 있습니다. ASCII 형식에서는 분리문자로 컬럼 위치를 사용하므로 가변 길이 필드로 임포트할 때 이 문제가 나타날 수 있습니다. striptblanks 파일 유형 수정자를 사용하여 뒤 공백을 절단합니다.

비표준 보안 레이블 값

또한 보안 레이블 값이 보안 레이블에서 구성요소 값을 포함하는 문자열(예: *S:(ALPHA,BETA)*)인 데이터 파일을 임포트할 수도 있습니다. 이를 수행하려면 seclabelchar 파일 유형 수정자를 사용해야 합니다. seclabelchar을 사용하면 데이터 유형이 DB2SECURITYLABEL인 컬럼의 값은 보안 레이블에 대한 문자열 형식으로 보안 레이블을 포함하는 문자열 상수로 가정합니다. 문자열이 적절한 형식이 아니면 행은 삽입되지 않고 경고(SQLSTATE 01H53)가 리턴됩니다. 문자열이 테이블을 보호하는 보안 규정에 포함된 유효한 보안 레이블을 표시하지 않으면 행은 삽입되지 않고 경고(SQLSTATE 01H53)가 리턴됩니다.

보안 레이블 컬럼 값이 보안 레이블 이름인 데이터 파일을 임포트할 수도 있습니다. 이러한 유형의 파일을 임포트하려면 seclabelname 파일 유형 수정자를 사용해야 합니다. seclabelname을 사용하는 경우 데이터 유형이 DB2SECURITYLABEL인 컬럼의 모든 값은 기존 보안 레이블의 이름을 포함하는 문자열 상수로 가정합니다. 테이블을 보호하는 보안 규정의 표시 이름에 보안 레이블이 없으면 행은 삽입되지 않고 경고(SQLSTATE 01H53)가 리턴됩니다.

예

모든 예에서 입력 데이터 파일 myfile.del은 DEL 형식입니다. 모두 REPS 테이블(다음 명령문으로 작성됨)로 데이터를 임포트합니다.

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

이 예에서 입력 파일은 디폴트 형식의 보안 레이블을 포함한다고 가정합니다.

```
db2 import from myfile.del of del modified by delprioritychar insert into reps
```

이 예에서 입력 파일은 보안 레이블 문자열 형식의 보안 레이블을 포함한다고 가정합니다.

```
db2 import from myfile.del of del modified by seclabelchar insert into reps
```

이 예에서 입력 파일은 보안 레이블 컬럼의 보안 레이블 이름을 포함한다고 가정합니다.

```
db2 import from myfile.del of del modified by seclabelname insert into reps
```

버퍼 지정 삽입 임포트

파티션된 데이터베이스 환경에서 임포트 유틸리티는 버퍼 지정 삽입을 사용하여 사용 가능하게 할 수 있습니다. 그러면 데이터 임포트 시 발생하는 메시징 양이 감소하므로 성능이 향상됩니다.

버퍼 지정 삽입 옵션은 오류가 보고되지 않는 확실한 경우에만 사용할 수 있습니다. 실패한 버퍼 지정 삽입에 대한 세부사항이 리턴되지 않기 때문입니다.

버퍼 지정 삽입을 사용하는 경우 임포트에서는 **WARNINGCOUNT** 디폴트값을 1로 설정합니다. 따라서 행이 거부되면 조작에 실패합니다. 레코드가 거부되면 유틸리티는 현재 트랜잭션을 롤백합니다. 커미트된 레코드 수를 사용하여 데이터베이스에 성공적으로 삽입된 레코드를 판별할 수 있습니다. 커미트된 레코드 수는 COMMITCOUNT 옵션이 지정된 경우에만 0이 될 수 없습니다.

다른 **WARNINGCOUNT** 값이 import 명령에 명시적으로 지정된 경우 일부 행이 거부되면 유틸리티의 행 요약 출력이 올바르지 않을 수 있습니다. 버퍼 지정 삽입에 사용된 비동기 오류 보고 및 행 그룹 삽입 중 발견된 오류로 해당 그룹의 모든 행이 백아웃되었다는 점이 오류의 원인입니다. 유틸리티는 거부된 입력 레코드에 대해 신뢰할 수 있는 정보를 보고하지 않으므로 커미트된 레코드와 데이터베이스로 다시 삽입해야 하는 레코드를 판별하기 어렵습니다.

DB2 바인드 유틸리티를 사용하여 버퍼 지정 삽입을 요청합니다. 임포트 패키지, db2uimpm.bnd는 INSERT BUF 옵션을 사용하여 데이터베이스에 리바인드되어야 합니다. 예를 들면, 다음과 같습니다.

```
db2 connect to your_database
db2 bind db2uimpm.bnd insert buf
```

버퍼 지정 삽입 기능은 INSERT_UPDATE 모드에서 임포트 조작과 함께 사용할 수 없습니다. 바인드 파일 db2uImpInsUpdate.bnd는 이 제한을 적용합니다. 이 파일은 INSERT BUF 옵션과 함께 바인드될 수 없습니다. 따라서 INSERT_UPDATE 모드에서 임포트 조작은 실패합니다. INSERT, REPLACE 또는 REPLACE_CREATE 모드에서 임포트 조작은 새 파일 바인드의 영향을 받지 않습니다.

ID 컬럼 임포트 고려사항

임포트 유틸리티는 입력 데이터에 ID 컬럼 값이 있는지에 상관없이 ID 컬럼을 포함하는 테이블로 데이터를 임포트하는 데 사용할 수 있습니다.

ID 관련 파일 유형 수정자가 사용되지 않으면 유틸리티는 다음 규칙에 따라 작동합니다.

- ID 컬럼이 GENERATED ALWAYS인 경우 입력 파일에서 ID 컬럼에 해당하는 행의 값이 누락되거나 명시적으로 널(NULL) 값이 지정되면 항상 테이블 행에 대한 ID 값이 생성됩니다. ID 컬럼에 대해 널(NULL)이 아닌 값이 지정되면 행은 거부됩니다(SQL3550W).
- ID 컬럼이 GENERATED BY DEFAULT인 경우 사용자 제공 값이 제공되면 임포트 유틸리티는 이 값을 사용합니다. 데이터가 누락되거나 명시적으로 널(NULL)이 지정되면 값이 생성됩니다.

임포트 유틸리티는 ID 컬럼의 데이터 유형(즉, SMALLINT, INT, BIGINT 또는 DECIMAL) 값에 대해 정상적으로 수행되는 작업 이외에도 사용자 제공 ID 컬럼 값에 대해 추가 유효성 확인 작업을 수행하지 않습니다. 중복 값은 보고되지 않습니다. 또한 compound=*x* 수정자는 ID 컬럼을 포함하는 테이블로 데이터를 임포트할 때 사용할 수 없습니다.

ID 컬럼을 포함하는 테이블로 데이터를 간단히 임포트할 수 있는 두 가지 방법이 있습니다. identitymissing 및 identityignore 파일 유형 수정자가 그 방법입니다.

ID 컬럼을 포함하지 않고 데이터 임포트

identitymissing 수정자를 사용하면 입력 데이터 파일에 ID 컬럼 값(널(NULL) 값도 아님)이 없는 경우 ID 컬럼을 포함한 테이블을 보다 편리하게 임포트할 수 있습니다. 예를 들어 다음 SQL문으로 정의된 테이블을 고려하십시오.

```
create table table1 (c1 char(30),
                     c2 int generated by default as identity,
                     c3 real,
                     c4 char(1))
```

파일(import.del)의 데이터를 TABLE1로 임포트하려고 합니다. 이 데이터는 ID 컬럼이 없는 테이블에서 익스포트되었을 수 있습니다. 다음은 이러한 파일에 대한 예입니다.

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

이 파일을 임포트하는 한 가지 방법은 다음과 같이 IMPORT 명령을 통해 임포트할 컬럼을 명시적으로 나열하는 것입니다.

```
db2 import from import.del of del replace into table1 (c1, c3, c4)
```

그러나 컬럼이 많은 테이블의 경우 구문이 복잡해지고 오류가 발생할 수 있습니다. 파일을 임포트하는 또 다른 방법은 다음과 같이 identitymissing 파일 유형 수정자를 사용하는 것입니다.

```
db2 import from import.del of del modified by identitymissing
      replace into table1
```

ID 컬럼을 포함하여 데이터 임포트

identityignore 수정자는 일부 측면에서 identitymissing 수정자와는 반대됩니다. 이 수정자는 입력 데이터 파일에 ID 컬럼 데이터가 있어도 데이터를 무시하고 각 행에 대한 ID 값을 생성하도록 임포트 유틸리티에 지시합니다. 예를 들어 위에서 정의한 대로 파일(import.del)에서 TABLE1로 다음 데이터를 임포트하려고 합니다.

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

사용자 제공 값 1, 2 및 3이 ID 컬럼에서 사용되지 않으면 다음 IMPORT 명령을 실행할 수 있습니다.

```
db2 import from import.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

다시 한 번 강조하지만 이 방법은 테이블에 컬럼이 많은 경우 복잡해지며 오류가 발생할 수 있습니다. identityignore 수정자는 다음과 같이 구문을 단순화합니다.

```
db2 import from import.del of del modified by identityignore
   replace into table1
```

ID 컬럼을 포함하는 테이블이 IXF 파일로 익스포트되면 IMPORT 명령의 REPLACE_CREATE 및 CREATE 옵션을 통해 ID 컬럼 등록 정보를 포함하여 테이블을 재작성할 수 있습니다. 유형이 GENERATED ALWAYS인 ID 컬럼을 포함하는 테이블에서 이러한 IXF 파일이 작성된 경우 데이터 파일을 성공적으로 임포트하는 유일한 방법은 identityignore 수정자를 지정하는 것입니다. 그렇지 않으면 모든 행이 거부됩니다(SQL3550W).

주: IMPORT 명령의 CREATE 및 REPLACE_CREATE 옵션은 사용되지 않으며 추후 릴리스에서 제거됩니다.

생성된 컬럼 임포트 고려사항

임포트 유틸리티는 입력 데이터에 생성된 컬럼 값이 있는지에 상관없이 생성된 컬럼(비 ID)을 포함하는 테이블로 데이터를 임포트하는 데 사용할 수 있습니다.

생성된 컬럼 관련 파일 유형 수정자가 사용되지 않으면 임포트 유틸리티는 다음 규칙에 따라 작동합니다.

- 입력 파일에서 컬럼에 해당하는 행의 값이 누락되거나 명시적으로 널(NULL) 값이 지정되면 항상 생성된 컬럼에 대한 값이 생성됩니다. 생성된 컬럼에 대해 널(NULL)이 아닌 값이 제공되면 행은 거부됩니다(SQL3550W).

- 서버에서 널(NULL) 값을 허용하지 않는 생성된 컬럼에 널(NULL) 값이 생성되면 이 필드가 속한 데이터의 행이 거부됩니다(SQL0407N). 예를 들어 널(NULL) 값을 허용하지 않는 생성된 컬럼이 입력 파일에서 널(NULL) 값을 제공하는 두 개의 테이블 컬럼 합으로 정의된 경우 이러한 상황이 나타날 수 있습니다.

생성된 컬럼을 포함하는 테이블로 데이터를 간단히 임포트할 수 있는 두 가지 방법이 있습니다. generatedmissing 및 generatedignore 파일 유형 수정자가 그 방법입니다.

생성된 컬럼을 포함하지 않고 데이터 임포트

generatedmissing 수정자를 사용하면 입력 데이터 파일에 테이블의 모든 생성된 컬럼 값(널(NULL) 값도 아님)이 없는 경우 생성된 컬럼을 포함하는 테이블로 데이터를 편리하게 임포트할 수 있습니다. 예를 들어 다음 SQL문으로 정의된 테이블을 고려하십시오.

```
create table table1 (c1 int,
                     c2 int,
                     g1 int generated always as (c1 + c2),
                     g2 int generated always as (2 * c1),
                     c3 char(1))
```

파일(load.del)의 데이터를 TABLE1로 임포트하려고 합니다. 이 데이터는 생성된 컬럼이 없는 테이블에서 익스포트되었을 수 있습니다. 다음은 이러한 파일에 대한 예입니다.

```
1, 5, J
2, 6, K
3, 7, I
```

이 파일을 임포트하는 한 가지 방법은 다음과 같이 IMPORT 명령을 통해 임포트할 컬럼을 명시적으로 나열하는 것입니다.

```
db2 import from import.del of del replace into table1 (c1, c2, c3)
```

그러나 컬럼이 많은 테이블의 경우 구문이 복잡해지고 오류가 발생할 수 있습니다. 파일을 임포트하는 또 다른 방법은 다음과 같이 generatedmissing 파일 유형 수정자를 사용하는 것입니다.

```
db2 import from import.del of del modified by generatedmissing
   replace into table1
```

생성된 컬럼을 포함하여 데이터 임포트

generatedignore 수정자는 일부 측면에서 generatedmissing 수정자와는 반대됩니다. 이 수정자는 입력 데이터 파일에 생성된 컬럼 데이터가 있어도 데이터를 무시하고 각 행에 대한 값을 생성하도록 임포트 유틸리티에 지시합니다. 예를 들어 위에서 정의한 대로 파일(import.del)에서 TABLE1로 다음 데이터를 임포트하려고 합니다.

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

사용자 제공, 널(NULL)이 아닌 값 10, 11 및 12(g1의 경우), 15, 16 및 17(g2의 경우)로 행이 거부됩니다(SQL3550W). 이를 방지하기 위해 다음 IMPORT 명령을 실행할 수 있습니다.

```
db2 import from import.del of del method P(1, 2, 5)
   replace into table1 (c1, c2, c3)
```

다시 한 번 강조하지만 이 방법은 테이블에 컬럼이 많은 경우 복잡해지며 오류가 발생할 수 있습니다. generatedignore 수정자는 다음과 같이 구문을 단순화합니다.

```
db2 import from import.del of del modified by generatedignore
   replace into table1
```

INSERT_UPDATE의 경우 생성된 컬럼이 1차 키이고 generatedignore 수정자가 지정된 경우 IMPORT 명령은 generatedignore 수정자를 고려합니다. IMPORT 명령은 UPDATE문의 WHERE절에서 이 컬럼에 대한 사용자 제공 값을 대체하지 않습니다.

LOB 임포트 고려사항

임포트 유틸리티에서는 단일 컬럼 값 크기를 32KB로 제한하므로 LOB를 임포트하는 경우 추가 고려사항을 고려해야 합니다.

임포트 유틸리티는 디폴트로 입력 파일의 데이터를 컬럼에 로드할 데이터로 처리합니다. 그러나 대형 오브젝트(LOB) 데이터는 기본 입력 데이터 파일에 저장되므로 데이터 크기는 32KB로 제한됩니다. 따라서 데이터 손실을 방지하려면 LOB 데이터를 기본 데이터 파일과는 별도로 저장하고 LOB를 임포트할 때 lobsinfile 파일 유형 수정자를 지정해야 합니다.

LOBS FROM절은 내재적으로 lobsinfile을 활성화합니다. LOBS FROM절은 데이터를 임포트할 때 LOB 파일에서 검색할 경로 목록을 임포트 유틸리티로 전달합니다. LOBS FROM 옵션을 지정하지 않으면 임포트할 LOB 파일은 입력 관계형 데이터 파일과 동일한 경로에 있다고 가정합니다.

LOB 데이터를 저장하는 위치 표시

LLS(LOB Location Specifier)는 LOB 정보를 임포트할 때 단일 파일에 여러 LOB를 저장하는 데 사용될 수 있습니다. lobsinfile이 지정되면 익스포트 유틸리티에서 이를 생성하고 익스포트 출력 파일에 저장합니다. 여기서 LOB 데이터를 저장할 수 있는 위치를 표시합니다. 지정된 lobsinfile 옵션으로 수정된 데이터를 임포트하는 경우 데이터베이스에서는 해당되는 각 LOB 컬럼에 대해 LLS가 있다고 예상합니다. LLS 이외의 항목이 LOB 컬럼에 있으면 데이터베이스는 이를 LOB 파일로 처리하고 전체 파일을 LOB로 로드합니다.

CREATE 모드에서 임포트하는 경우 LONG IN절을 사용하여 LOB 데이터를 별도의 테이블 스페이스에 작성 및 저장하도록 지정할 수 있습니다.

다음 예에서는 별도의 파일에 LOB가 저장된 DEL 파일을 임포트하는 방법을 보여줍니다.

```
IMPORT FROM inputfile.del OF DEL
 LOBS FROM /tmp/data
 MODIFIED BY lobsinfile
 INSERT INTO newtable
```

사용자 정의 구별 유형 임포트 고려사항

임포트 유틸리티는 사용자 정의 구별 유형(UDT)을 유사한 기본 데이터 유형에 자동으로 캐스트합니다. 그러므로 사용자가 UDT를 명시적으로 기본 데이터 유형에 캐스트하지 않아도 됩니다. 캐스팅을 통해 SQL에서 기본 데이터 유형 및 UDT 사이를 비교할 수 있습니다.

임포트 시 추가 주의 사항

클라이언트/서버 환경 및 임포트

리모트 데이터베이스로 파일을 임포트하는 경우 스토어드 프로시저를 호출하여 서버에서 임포트를 수행할 수 있습니다.

다음 경우에는 스토어드 프로시저를 호출할 수 없습니다.

- 응용프로그램 및 데이터베이스 코드 페이지가 다릅니다.
- 임포트할 파일이 다중 파트로 구성된 PC/IXF 파일입니다.
- 데이터 임포트에 컬럼 이름 또는 상대적 컬럼 위치가 사용됩니다.
- 제공된 목표 컬럼 목록이 4KB보다 큽니다.
- LOBS FROM절 또는 `lobsinfile` 수정자가 지정됩니다.
- NULL INDICATORS절이 ASC 파일에 지정됩니다.

임포트에서 스토어드 프로시저를 사용하는 경우 서버에 설치된 기본 언어를 사용하여 메시지 파일에 메시지가 작성됩니다. 클라이언트 및 서버의 언어가 동일하면 응용프로그램의 언어로 메시지가 작성됩니다.

임포트 유틸리티에서는 `sqllib` 디렉토리의 `tmp` 서브디렉토리 또는 **DB2INSTPROF** 레지스트리 변수를 지정한 경우 여기서 표시하는 디렉토리에 두 개의 임시 파일을 작성합니다. 한 파일은 데이터용이고 다른 파일은 임포트 유틸리티에서 생성된 메시지용입니다.

서버에서 데이터를 열거나 쓰는 중 오류가 수신된 경우 다음을 확인하십시오.

- 디렉토리가 존재합니다.

- 파일을 저장할 디스크 스페이스가 충분합니다.
- 인스턴스 소유자에게 디렉토리에 쓰기 권한이 있습니다.

테이블 잠금 모드는 임포트 유틸리티에서 지원됨

임포트 유틸리티에서는 오프라인 또는 ALLOW NO ACCESS 모드와, 온라인 또는 ALLOW WRITE ACCESS 모드와 같은 두 가지 테이블 잠금 모드를 지원합니다.

ALLOW NO ACCESS 모드에서는 동시 응용프로그램이 테이블 데이터에 액세스할 수 없습니다. ALLOW WRITE ACCESS 모드에서는 동시 응용프로그램에 임포트 목표 테이블에 대한 읽기 및 쓰기 액세스 권한이 부여됩니다. 모드가 명시적으로 지정되지 않은 경우 임포트는 디폴트 모드(ALLOW NO ACCESS)로 실행됩니다. 또한 임포트 유틸리티는 디폴트로 분리 수준 읽기 안정성(RS)에 기반하여 데이터베이스에 바인드됩니다.

오프라인 임포트(ALLOW NO ACCESS)

ALLOW NO ACCESS 모드에서 임포트는 행을 삽입하기 전에 목표 테이블에서 배타적(X) 잠금을 획득합니다. 테이블에서 잠금을 보유하는 것은 다음과 같은 두 가지 의미를 내포합니다.

- 먼저 임포트 목표 테이블에서 테이블 잠금 또는 행 잠금을 보유하는 다른 응용프로그램이 있는 경우 임포트 유틸리티는 해당 응용프로그램이 변경 사항을 롤백하거나 커미트하기를 기다립니다.
- 다음으로 임포트를 실행하는 중 잠금을 요청하는 다른 응용프로그램은 임포트 조작이 완료되길 기다립니다.

주: 임포트 유틸리티를 포함하여 응용프로그램이 잠금에 대해 무한정으로 대기하지 않도록 잠금 시간종료 값을 지정할 수 있습니다.
조작 시작 시 배타적 잠금을 요청하면 임포트할 때 다른 응용프로그램이 동일한 목표 테이블에서 행 잠금을 사용 및 보유하는 결과로서 발생하는 교착 상태를 방지할 수 있습니다.

온라인 임포트(ALLOW WRITE ACCESS)

ALLOW WRITE ACCESS 모드에서 임포트 유틸리티는 목표 테이블에 대한 비배타적(IX) 잠금을 획득합니다. 테이블에서 이 잠금을 보유하는 것은 다음과 같은 의미를 내포합니다.

- 호환되지 않는 테이블 잠금을 보유하는 다른 응용프로그램이 있는 경우 임포트 유틸리티는 해당 응용프로그램이 변경 사항을 롤백하거나 커미트할 때까지 데이터 삽입을 시작하지 않습니다.
- 임포트를 실행하는 중 호환되지 않는 테이블 잠금을 요청하는 다른 응용 프로그램은 임포트에서 현재 트랜잭션을 커미트하거나 롤백할 때까지 기다립니다. 임포트의 테이

블 잠금은 트랜잭션 경계를 넘어 지속되지 않습니다. 따라서 온라인 임포트에서는 모든 커미트 이후에 테이블 잠금을 요청하고 이를 기다려야 합니다.

- 호환되지 않는 행 잠금을 보유하는 다른 응용프로그램이 있는 경우 임포트 유틸리티는 해당 응용프로그램이 변경 사항을 롤백하거나 커미트할 때까지 데이터 삽입을 중지합니다.
- 임포트를 실행하는 중 호환되지 않는 행 잠금을 요청하는 다른 응용 프로그램은 임포트 조작에서 현재 트랜잭션을 커미트하거나 롤백할 때까지 기다립니다.

온라인 등록 정보를 보존하고 교착 상태 발생 가능성을 낮추기 위해 ALLOW WRITE ACCESS 임포트에서는 정기적으로 현재 트랜잭션을 커미트하고 배타적 테이블 잠금으로 에스컬레이션하기 전에 모든 행 잠금을 해제합니다. 명시적으로 커미트 빈도를 설정하지 않은 경우 임포트에서는 COMMITCOUNT AUTOMATIC이 지정된 경우와 같이 커미트를 수행합니다. COMMITCOUNT가 0으로 설정되면 커미트는 수행되지 않습니다.

ALLOW WRITE ACCESS 모드는 다음 경우에 호환 가능하지 않습니다.

- REPLACE, CREATE 또는 REPLACE_CREATE 모드에서 임포트하는 경우
- 버퍼 지정 삽입으로 임포트하는 경우
- 목표 뷰로 임포트하는 경우
- 계층 구조 테이블로 임포트하는 경우
- 해당 잠금 단위가 테이블 레벨로 설정된 테이블로 임포트하는 경우(ALTER TABLE 문의 LOCKSIZE 매개변수를 사용하여 설정됨)

참조 - 임포트

IMPORT

지원되는 파일 형식을 갖는 외부 파일의 데이터를 테이블, 계층 구조, 뷰 또는 별칭으로 데이터를 삽입합니다. LOAD가 더 빠른 방법이기는 하지만 로드 유틸리티는 계층 구조 레벨에서 데이터 로드를 지원하지 않습니다.

89 페이지의 『임포트 유틸리티의 파일 유형 수정자』로의 빠른 링크를 참조하십시오.

권한 부여

- **INSERT** 옵션을 사용하여 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 각 참여 중인 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT 및 SELECT 특권

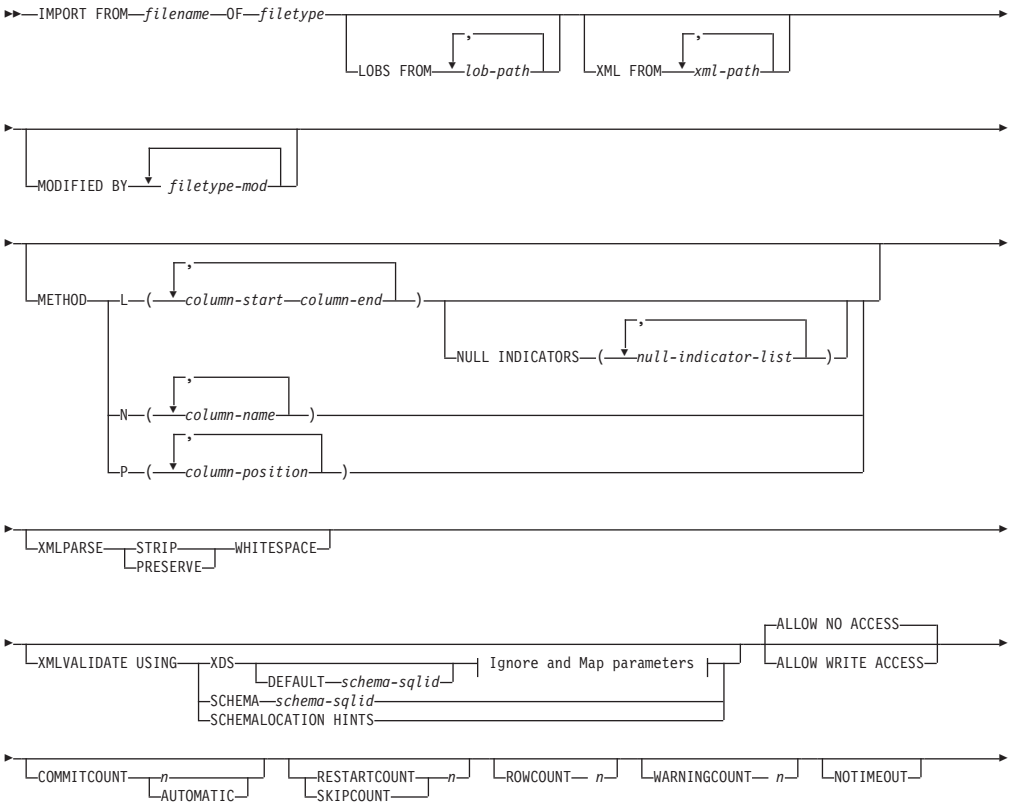
- **INSERT_UPDATE** 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 각 참여 중인 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT, SELECT, UPDATE 및 DELETE 특권
- **REPLACE** 또는 **REPLACE_CREATE** 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 테이블 또는 뷰에 대한 CONTROL 특권
  - 테이블 또는 뷰에 대한 INSERT, SELECT 및 DELETE 특권
- **CREATE** 또는 **REPLACE_CREATE** 옵션을 사용하여 새 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dbadm* 권한
  - 데이터베이스에 대한 CREATETAB 권한 및 테이블 스페이스에 대한 USE 특권은 다음 경우 중 하나입니다.
    - 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
- **CREATE** 또는 **REPLACE_CREATE** 옵션을 사용하여 존재하지 않는 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dbadm* 권한
  - 데이터베이스에 대한 CREATETAB 권한과 테이블 스페이스에 대한 USE 특권 및 다음 중 하나가 필요합니다.
    - 테이블의 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마가 존재할 경우, 스키마에 대한 CREATEIN 특권
    - 전체 계층 구조에 대한 **REPLACE_CREATE** 옵션이 사용될 경우, 계층 구조의 모든 서브테이블에 대한 CONTROL 특권
- **REPLACE** 옵션을 사용하여 기존 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 계층 구조에 있는 모든 서브테이블에 대한 CONTROL 특권

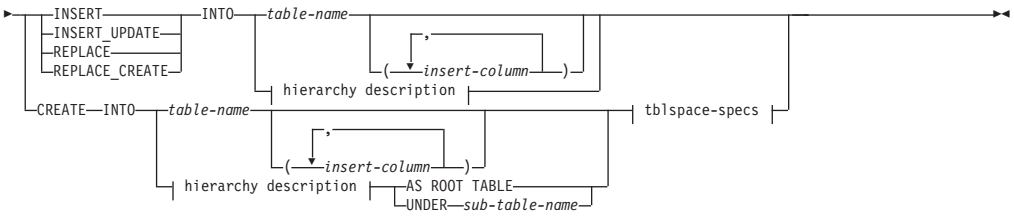
- 보호 컬럼이 있는 테이블로 데이터를 임포트하려면 테이블의 모든 보호 컬럼에 대한 쓰기 액세스를 허용하는 LBAC 증명서가 세션 권한 부여 ID에 있어야 합니다. 그렇지 않으면 임포트를 실패하고 오류(SQLSTATE 42512)가 리턴됩니다.
- 행이 보호 설정된 테이블로 데이터를 임포트하려면 세션 권한 부여 ID가 다음 기준을 충족하는 LBAC 증명서를 보유해야 합니다.
  - 테이블을 보호하는 보안 규정에 포함됨
  - 세션 권한 부여 ID에 쓰기 액세스에 대한 권한이 부여됨

  삽입할 행에 대한 레이블, 사용자의 LBAC 신임, 보안 규정 정의 및 LBAC 규칙이 행에 대한 레이블을 판별합니다.
- **REPLACE** 또는 **REPLACE_CREATE** 옵션을 지정할 경우 세션 권한 부여 ID는 테이블을 삭제할 수 있는 권한이 있어야 합니다.
- 데이터를 별칭으로 임포트하려면 세션 권한 부여 ID에 pass-through 모드에서 지정된 데이터 소스를 액세스하고 사용할 특권이 있어야 합니다.

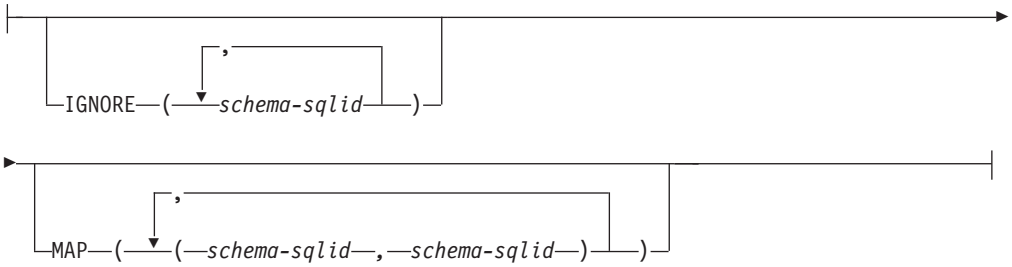
필수 연결

명령 구문



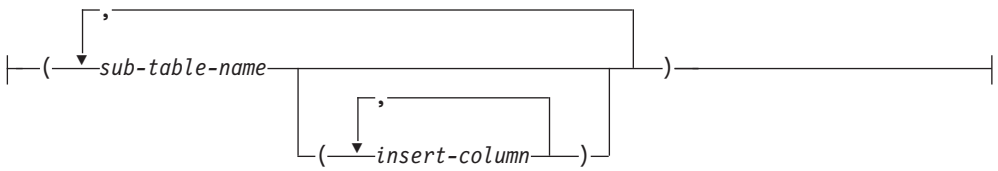


Ignore and Map parameters:



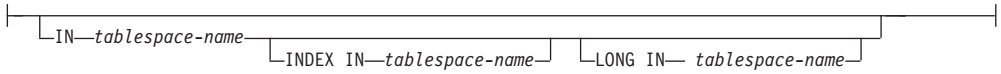
hierarchy description:

sub-table-list:



traversal-order-list:

tblspace-specs:



명령 매개변수

ALL TABLES

계층 구조 전용 내재적 키워드. 계층 구조를 임포트할 때 디폴트로 트래버스 순서로 지정된 모든 테이블을 임포트합니다.

ALLOW NO ACCESS

오프라인 모드로 임포트를 실행합니다. 행이 삽입되기 전에 목표 테이블에 대해 독점(X) 잠금을 획득합니다. 이렇게 하면 동시 응용프로그램이 테이블 데이터에 액세스하지 못하게 합니다. 이는 디폴트 임포트 동작입니다.

ALLOW WRITE ACCESS

온라인 모드로 임포트를 실행합니다. 첫 번째 행이 삽입될 때 목표 테이블에 대해 의도를 가진 독점(IX) 잠금을 획득합니다. 이렇게 하면 동시 판독기 및 기록기가 테이블 데이터에 액세스할 수 있습니다. 온라인 모드는 **REPLACE**, **CREATE** 또는 **REPLACE_CREATE** 임포트 옵션과 호환 가능하지 않습니다. 온라인 모드는 버퍼링된 삽입과 함께 지원되지 않습니다. 임포트 조작은 테이블 잠금에 대한 잠금 에스컬레이션을 방지하고 사용 중인 로그 스페이스 외부에서 실행되지 않도록 하기 위해 삽입된 데이터를 주기적으로 커미트합니다. 이러한 커미트는 **COMMITCOUNT** 옵션을 사용하지 않는 경우에도 수행됩니다. 각 커미트를 수행하는 동안 임포트는 해당 IX 테이블 잠금을 유실하므로 커미트 후 이를 다시 획득하려고 시도합니다. 별칭으로 임포트할 때 이 매개변수가 필요하며 유효한 숫자를 사용하여 **COMMITCOUNT**를 지정해 야 합니다(AUTOMATIC은 유효한 옵션으로 간주하지 않음).

AS ROOT TABLE

하나 이상의 서브테이블을 독립형 테이블 계층 구조로 작성합니다.

COMMITCOUNT *n* **| AUTOMATIC**

*n*개의 레코드가 모두 임포트된 후 COMMIT를 수행합니다. 숫자 *n*을 지정하면 임포트는 *n*개의 레코드가 모두 임포트된 후 COMMIT를 수행합니다. 복합 삽입을 사용할 때, 사용자 지정 커미트 빈도 *n*은 복합 계수 값의 첫 번째 정수 배수로 반올림됩니다. AUTOMATIC을 지정하면 임포트는 내부적으로 커미트를 수행해야 하는 때를 판별합니다. 이 유틸리티는 다음 두 가지 이유 중 하나에 대해 커미트합니다.

- 사용 중인 로그 스페이스 외부에서 실행하지 않도록 하기 위해
- 행 레벨에서 테이블 레벨로 잠금 에스컬레이션되지 않도록 하기 위해

ALLOW WRITE ACCESS 옵션을 지정하고 **COMMITCOUNT** 옵션을 지정하지 않으면, 임포트 유틸리티는 **COMMITCOUNT** AUTOMATIC이 지정된 것과 같이 커미트를 수행합니다.

사용 중인 로그 스페이스를 모두 사용해버리지 않도록 하기 위한 임포트 조작의 기능은 DB2 레지스트리 변수 **DB2_FORCE_APP_ON_MAX_LOG**에 의해 영향을 받습니다.

- **DB2_FORCE_APP_ON_MAX_LOG**가 FALSE로 설정되고 **COMMITCOUNT** AUTOMATIC 명령 옵션이 지정된 경우, 임포트 유틸리티는 자동으로 사용 중인 로그 스페이스를 모두 사용해버리지 않도록 할 수 있습니다.
- **DB2_FORCE_APP_ON_MAX_LOG**가 FALSE로 설정되고 **COMMITCOUNT** *n* 명령 옵션이 지정된 경우, 레코드를 삽입하거나 갱신하는 중 SQL0964C(트랜잭션 로그가 가득참) 발생 시 임포트 유틸리티는 로그 가득참 조건을 해결하려고 합니다. 무조건 커미트를 수행한 후 레코드를 삽입하거나 갱신하려고 재시도합니다. 이것이 로그 가득참 조건(로그 가득참이 데이터베이스의 다른 활동에 영향을 주는 경우)을 해결하는 데 유용하지 않을 경우, IMPORT 명령은 예상대로 실패합니다. 그러나 커미트되는 행 수는 **COMMITCOUNT** *n* 값 의 배수가 되지 않을 수 있습니다. 임포트 조작 재시도 시 이미 커미트된 행 처리를 피하려면, **RESTARTCOUNT** 또는 **SKIPCOUNT** 명령 매개변수를 사용하십시오.
- **DB2_FORCE_APP_ON_MAX_LOG**가 TRUE로 설정된 경우(디폴트), 레코드를 삽입하거나 갱신하는 중 SQL0964C 발생 시 임포트 조작이 실패합니다. **COMMITCOUNT** AUTOMATIC 또는 **COMMITCOUNT** *n* 지정 여부와 관계 없이 이런 현상이 일어날 수 있습니다.

  응용프로그램에서 데이터베이스가 강제 해제되고 현재 작업 단위(UOW)가 롤백됩니다. 임포트 조작 재시도 시 이미 커미트된 행 처리를 피하려면, **RESTARTCOUNT** 또는 **SKIPCOUNT** 명령 매개변수를 사용하십시오.

CREATE

주: CREATE 매개변수는 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항은 『IMPORT 명령 옵션 **CREATE** 및 **REPLACE_CREATE**가 사용되지 않음』을 참조하십시오.

데이터베이스의 코드 페이지에 테이블 정의 및 행 내용을 작성합니다. DB2 테이블, 서브테이블 또는 계층 구조에서 데이터를 익스포트한 경우, 인덱스가 작성됩니다. 이 옵션이 계층 구조에서 작동하고 DB2에서 데이터를 익스포트한 경우, 자료형 계층 구조도 작성됩니다. 이 옵션은 IXF 파일에 대해서만 사용할 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

주: MVS™ 호스트 데이터베이스에서 데이터를 익스포트하고 호스트 페이지 크기에서 계산된 길이가 254를 초과하는 LONGVAR 필드를 포함할 경우, 행이

너무 길기 때문에 **CREATE**에 실패할 수 있습니다. 제한사항 목록에 대해서는 『임포트된 테이블 재작성』을 참조하십시오. 이 경우, 테이블을 직접 작성한 후 **INSERT**를 사용하여 IMPORT를 호출하거나 LOAD 명령을 사용해야 합니다.

DEFAULT *schema-sqlid*

이 옵션은 **USING XDS** 매개변수가 지정된 경우에만 사용할 수 있습니다. **DEFAULT** 절을 통해 지정된 스키마는 임포트된 XML 문서의 XDS(XML Data Specifier)가 XML 스키마를 식별하는 SCH 속성을 포함하지 않을 때 유효성 확인에 사용할 스키마를 식별합니다.

DEFAULT 절은 **IGNORE** 및 **MAP** 절보다 우선순위를 갖습니다. XDS가 **DEFAULT** 절을 충족시키면 **IGNORE** 및 **MAP** 스펙은 무시됩니다.

FROM *filename*

HIERARCHY

계층 구조 데이터가 임포트되도록 지정합니다.

IGNORE *schema-sqlid*

이 옵션은 **USING XDS** 매개변수가 지정된 경우에만 사용할 수 있습니다. SCH 속성에 의해 식별될 경우 **IGNORE** 절은 무시할 하나 이상의 스키마 목록을 지정합니다. 임포트된 XML 문서에 대한 XDS(XML Data Specifier)에 SCH 속성이 존재하고 SCH 속성에 의해 식별된 스키마가 무시할 스키마 목록에 포함된 경우, 이 임포트된 XML 문서에 대해서는 스키마 유효성 확인이 발생하지 않습니다.

IGNORE 절에 스키마가 지정되어 있으면 이 스키마는 또한 **MAP** 절에 있는 스키마 쌍의 왼쪽에 존재할 수 없습니다.

IGNORE 절은 XDS에만 적용됩니다. **MAP** 절에 의해 맵핑된 스키마는 **IGNORE** 절에 의해 지정된 경우 계속 무시되지 않습니다.

IN *tablespace-name*

테이블이 작성될 테이블 스페이스를 식별합니다. 테이블 스페이스가 있어야 하며 이 테이블 스페이스는 REGULAR 테이블 스페이스이어야 합니다. 다른 테이블 스페이스를 지정하지 않으면 모든 테이블 부분이 이 테이블 스페이스에 저장됩니다. 이 절을 지정하지 않으면 테이블은 권한 부여 ID가 작성한 테이블 스페이스에 작성됩니다. 테이블 스페이스를 찾을 수 없으면 테이블은 디폴트 테이블 스페이스 USERSPACE1에 넣어집니다. USERSPACE1가 삭제된 경우 테이블 작성에 실패합니다.

INDEX IN *tablespace-name*

테이블의 인덱스가 작성될 테이블 스페이스를 식별합니다. 이 옵션은 **IN** 절에 지정된 1차 테이블 스페이스가 DMS 테이블 스페이스일 경우에만 허용됩니다.

지정된 테이블 스페이스가 있어야 하며 이 테이블 스페이스는 REGULAR 또는 LARGE DMS 테이블 스페이스이어야 합니다.

주: 인덱스를 포함할 테이블 스페이스는 테이블이 작성되었을 때만 지정할 수 있습니다.

insert-column
데이터가 삽입될 테이블 또는 뷰에 있는 컬럼 이름을 지정합니다.

INSERT
기존 테이블 데이터를 변경하지 않고 임포트된 데이터를 테이블에 추가합니다.

INSERT_UPDATE
임포트된 데이터 행을 목표 테이블에 추가하거나 기본 키가 일치하는 기존 행(목표 테이블의)을 갱신합니다.

INTO *table-name*
데이터가 임포트될 데이터베이스 테이블을 지정합니다. 이 테이블은 시스템 테이블, 작성된 임시 테이블, 선언된 임시 테이블 또는 요약 테이블이 될 수 없습니다.

완전한 테이블 이름 또는 규정되지 않은 테이블 이름을 사용해야 하는 이전 서버의 경우를 제외하고는 **INSERT**, **INSERT_UPDATE** 또는 **REPLACE**에 별명을 사용할 수 있습니다. 규정된 테이블 이름의 형식은 *schema.tablename* 입니다. *schema*는 테이블이 작성된 사용자 이름입니다.

LOBS FROM *lob-path*
LOB 데이터 파일의 이름은 기본 데이터 파일(ASC, DEL 또는 IXF)에 저장되는데 LOB 컬럼으로 로드될 컬럼에 저장됩니다. 지정할 수 있는 최대 경로 수는 999입니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

LONG IN *tablespace-name*
긴 컬럼(LONG VARCHAR, LONG VARGRAPHIC, LOB 데이터 유형 또는 이들 유형 중 하나를 소스 유형으로 갖는 구별 유형)의 값이 저장될 테이블 스페이스를 식별합니다. 이 옵션은 **IN** 절에 지정된 1차 테이블 스페이스가 DMS 테이블 스페이스일 경우에만 허용됩니다. 테이블 스페이스가 있어야 하며 LARGE DMS 테이블 스페이스이어야 합니다.

MAP *schema-sqlid*
이 옵션은 **USING XDS** 매개변수가 지정된 경우에만 사용할 수 있습니다. 임포트된 각 XML 문서에 대한 XDS(XML Data Specifier)의 SCH 속성이 지정하는 스키마 대신 사용할 대체 스키마를 지정하려면 이 **MAP** 절을 사용하십시오. **MAP** 절은 하나 이상의 스키마 쌍의 목록을 지정하며, 여기서 각 쌍은 한 스키마 대 다른 스키마의 맵핑을 나타냅니다. 쌍에서 첫 번째 스키마는

XDS에 있는 SCH 속성에 의해 참조되는 스키마를 나타냅니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

MAP 절에 있는 스키마 쌍의 왼쪽에 있는 스키마는 **IGNORE** 절에 지정될 수 없습니다.

스키마 쌍 맵핑이 적용된 후, 최종 결과가 됩니다. 맵핑 조작은 전이되지 않으므로 선택된 스키마는 다른 스키마 상 맵핑에 계속 적용되지 않습니다.

스키마가 두 번 이상 맵핑될 수 없다는 것은 쌍의 왼쪽에 두 번 이상 나타날 수 없다는 것을 의미합니다.

METHOD

L 데이터를 임포트할 시작 및 끝 컬럼 번호를 지정합니다. 컬럼 번호는 데이터 행이 시작되는 바이트 오프셋입니다. 1부터 번호가 매겨집니다.

주: 이 방법은 ASC 파일에만 사용할 수 있으며 이 파일 유형에 대해 유일한 유효 옵션입니다.

N 임포트할 데이터 파일의 컬럼 이름을 지정합니다. 이 컬럼 이름의 대소문자는 시스템 카탈로그의 해당 이름의 대소문자와 일치해야 합니다. 널(NULL) 입력이 가능하지 않은 각 테이블 컬럼에는 **METHOD N** 목록에 해당 항목이 있어야 합니다. 예를 들면, 주어진 데이터 필드 F1, F2, F3, F4, F5 및 F6과 테이블 컬럼 C1 INT, C2 INT NOT NULL, C3 INT NOT NULL 및 C4 INT, method N (F2, F1, F4, F3)은 유효한 요청인 반면, method N (F2, F1)은 유효하지 않습니다.

주: 이 방법은 IXF 파일에 대해서만 사용할 수 있습니다.

P 임포트될 입력 데이터 필드의 필드 번호를 지정합니다.

주: 이 방법은 IXF 또는 DEL 파일에만 사용할 수 있으며 DEL 파일 유형에 대해 유일한 유효 옵션입니다.

MODIFIED BY *filetype-mod*

파일 유형 수정자 옵션을 지정합니다. 89 페이지의 『임포트 유틸리티의 파일 유형 수정자』를 참조하십시오.

NOTIMEOUT

임포트 유틸리티가 잠금 대기 중 시간종료되지 않도록 지정합니다. 이 옵션은 **locktimeout** 데이터베이스 구성 매개변수를 대체합니다. 다른 응용프로그램에는 영향을 주지 않습니다.

NULL INDICATORS *null-indicator-list*

이 옵션은 **METHOD L** 매개변수가 지정된 경우에만 사용할 수 있습니다. 즉, 입력 파일은 ASC 파일입니다. 널(NULL) 표시기 목록은 각 널(NULL) 표시

기 필드의 컬럼 번호를 지정하는 쉼표로 구분된 양의 정수 목록입니다. 컬럼 번호는 데이터 행이 시작되는 널(NULL) 표시기 필드의 바이트 오프셋입니다. **METHOD L** 매개변수에 정의된 각 데이터 필드에 대해 한 개의 항목이 널(NULL) 표시기 목록에 있어야 합니다. 컬럼 번호 0은 해당 데이터 필드에 데이터가 항상 들어 있음을 나타냅니다.

널(NULL) 표시기의 `Y` 값은 컬럼 데이터를 널(NULL)로 지정합니다. 널(NULL) 표시기 컬럼에서 `Y` *이외의* 모든 문자는 컬럼 데이터가 널(NULL)이 아니며 **METHOD L** 옵션이 지정하는 컬럼 데이터가 임포트됨을 지정합니다.

`nullindchar` 파일 유형 수정자와 함께 **MODIFIED BY** 옵션을 사용하여 널(NULL) 표시기 문자를 변경할 수 있습니다.

OF *filetype*

입력 파일에 있는 데이터의 형식을 지정합니다.

- ASC(컬럼 식별자가 없는 ASCII 형식)
- DEL(컬럼 식별자가 있는 ASCII 형식) - 여러 데이터베이스 관리 프로그램 및 파일 관리자 프로그램에서 사용됩니다.
- WSF(작업시트 형식) - 다음과 같은 프로그램에서 사용됩니다.
  - Lotus 1-2-3
  - Lotus Symphony
- IXF(Integration Exchange Format, PC 버전)는 DB2에 의해 독점 사용되는 2진 형식입니다.

중요사항: WSF 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

별칭으로 임포트할 때 WSF 파일 유형은 지원되지 않습니다.

REPLACE

데이터 오브젝트를 잘라내어 테이블에서 모든 기존 데이터를 삭제한 후 임포트된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다. 이 옵션은 테이블이 존재할 경우에만 사용할 수 있습니다. 계층 구조 간에 데이터를 이동할 때 이 옵션을 사용하면 개별적 서브테이블이 아닌 전체 계층 구조에 대한 데이터만 바꿀 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

이 옵션은 CREATE TABLE문의 NOT LOGGED INITIALLY(NLI)절이나 ALTER TABLE문의 ACTIVE NOT LOGGED INITIALLY절을 인정하지 않습니다.

NLI절이 호출된 CREATE TABLE 또는 ALTER TABLE문과 동일한 트랜잭션에서 **REPLACE** 옵션을 사용한 임포트를 수행할 경우, 임포트는 NLI절을 인정하지 않습니다. 모든 삽입은 로그됩니다.

일시적인 해결책 1

DELETE문을 사용하여 테이블의 내용을 삭제한 후 INSERT문을 사용하여 임포트를 호출하십시오.

일시적인 해결책 2

테이블을 삭제하고 재작성한 후 INSERT문을 사용하여 임포트를 호출하십시오.

이 제한사항은 DB2® Universal Database™ 버전 7 및 DB2 UDB 버전 8에 적용됩니다.

REPLACE_CREATE

주: REPLACE_CREATE 매개변수는 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항은 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE가 사용되지 않음』을 참조하십시오.

테이블이 존재할 경우, 데이터 오브젝트를 잘라내어 테이블에서 모든 기존 데이터를 삭제한 후 테이블 정의 또는 인덱스 정의를 변경하지 않고 임포트된 데이터를 삽입합니다.

테이블이 존재하지 않을 경우, 데이터베이스의 코드 페이지에 행 내용과 함께 테이블 및 인덱스 정의를 작성합니다. 제한사항 목록에 대해서는 *임포트된 테이블 재작성*을 참조하십시오.

이 옵션은 IXF 파일에 대해서만 사용할 수 있습니다. 계층 구조 간에 데이터를 이동하는 경우에 이 옵션을 사용하면 개별 부속 테이블이 아닌 전체 계층 구조에 대한 데이터만 바꿀 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

RESTARTCOUNT *n*

임포트 조작이 레코드 *n* + 1에서 시작되도록 지정합니다. 첫 번째 *n* 레코드는 건너뜁니다. 이 솔루션은 기능적으로 **SKIPCOUNT**와 동일합니다. **RESTARTCOUNT** 및 **SKIPCOUNT**는 상호 배타적입니다.

ROWCOUNT *n*

파일에 있는 *n*개의 실제 레코드가 임포트(삽입되거나 갱신됨)되도록 지정합니다. 사용자가 **SKIPCOUNT** 또는 **RESTARTCOUNT** 옵션에 의해 판별된 레코드로부터 *n*개의 행만을 파일에서 임포트할 수 있게 합니다. **SKIPCOUNT** 또는 **RESTARTCOUNT** 옵션이 지정되지 않은 경우 첫 번째 *n* 행만 임포트합니다. **SKIPCOUNT** *m* 또는 **RESTARTCOUNT** *m*이 지정된 경우 *m*+1에

서 *m+n*까지의 행을 임포트합니다. 복합 삽입을 사용할 때, **ROWCOUNT** *n*은 복합 계수 값의 첫 번째 정수 배수로 반올림됩니다.

SKIPCOUNT *n*

임포트 조작이 레코드 *n* + 1에서 시작되도록 지정합니다. 첫 번째 *n* 레코드는 건너뜁니다. 이 옵션은 기능적으로 **RESTARTCOUNT**와 동일합니다. **SKIPCOUNT** 및 **RESTARTCOUNT**는 상호 배타적입니다.

STARTING *sub-table-name*

*sub-table-name*부터 시작하고 디폴트 순서를 요청하며 계층 구조 전용 키워드입니다. PC/IXF 파일의 경우 디폴트 순서는 입력 파일에 저장된 순서입니다. PC/IXF 파일 형식에는 디폴트 순서만 사용할 수 있습니다.

sub-table-list

INSERT 또는 **INSERT_UPDATE** 옵션을 사용하는 유형이 지정된 테이블의 경우, 데이터가 임포트될 서브테이블을 표시하기 위해 서브테이블 이름 목록을 사용합니다.

traversal-order-list

INSERT, **INSERT_UPDATE** 또는 **REPLACE** 옵션을 사용하는 유형이 지정된 테이블의 경우, 계층 구조에서 임포트 중인 서브테이블의 트래버스 순서를 표시하기 위해 서브테이블 이름 목록을 사용합니다.

UNDER *sub-table-name*

하나 이상의 서브테이블을 작성하기 위한 상위 테이블을 지정합니다.

WARNINGCOUNT *n*

*n*번의 경고 후 임포트 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 임포트 파일 또는 목표 테이블이 올바르지 않게 지정될 경우 임포트 유틸리티는 임포트하려는 각 행에 대해 경고를 생성하므로 임포트를 실패하게 됩니다. *n*이 0이거나 이 옵션이 지정되지 않은 경우 임포트 조작은 발행된 경고 수에 관계없이 계속 수행됩니다.

XML FROM *xml-path*

XML 파일이 들어 있는 하나 이상의 경로를 지정합니다.

XMLPARSE

XML 문서가 구문 분석되는 방법을 지정합니다. 이 옵션을 지정하지 않을 경우, XML 문서에 대한 구문 분석 동작은 CURRENT XMLPARSE OPTION 특수 레지스터의 값으로 판별됩니다.

STRIP WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하도록 지정합니다.

PRESERVE WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하지 않도록 지정합니다.

XMLVALIDATE

XML 문서가 스키마에 대해 유효성이 확인되도록 지정합니다(해당되는 경우).

USING XDS

기본 데이터 파일의 XDS(XML Data Specifier)에 의해 식별된 XML 스키마에 대해 XML 문서의 유효성을 확인합니다. **USING XDS**와 함께 **XMLVALIDATE** 옵션을 호출한 경우, 유효성 확인을 수행하는 데 사용된 스키마는 디폴트로 XDS의 SCH 속성에 의해 판별됩니다. SCH 속성이 XDS에 존재하지 않을 경우, **DEFAULT** 절로 디폴트 스키마를 지정하지 않으면 스키마 유효성 확인이 발생하지 않습니다.

DEFAULT, **IGNORE** 및 **MAP** 절은 스키마 판별 동작을 수정하는 데 사용될 수 있습니다. 이들 세 개의 선택적 절은 XDS의 권장 스펙에 직접적으로 적용되며 서로에게는 적용되지 않습니다. 예를 들어, 한 스키마가 **DEFAULT** 절에서 지정되어 선택되었으면 이 스키마는 **IGNORE** 절에 지정되어도 무시되지 않습니다. 마찬가지로 한 스키마가 MAP 절에서 첫 번째 파트 쌍으로 지정되어 선택되면 다른 **MAP** 절 쌍의 두 번째 파트에 지정되어도 다시 맵핑되지 않습니다.

USING SCHEMA *schema-sqlid*

XML 문서가 지정된 SQL ID가 있는 XML 스키마에 대해 유효성이 확인됩니다. 이 경우 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

USING SCHEMALOCATION HINTS

XML 문서가 소스 XML 문서의 XML 스키마 위치 힌트에 의해 식별된 스키마에 대해 유효성이 확인됩니다. XML 문서에서 schemaLocation 속성을 찾을 수 없으면 유효성 확인이 발생하지 않습니다. **USING SCHEMALOCATION HINTS** 절을 지정하면 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

아래의 **XMLVALIDATE** 옵션 예를 참조하십시오.

사용 시 참고사항

임포트 조작을 시작하기 전에 반드시 모든 테이블 조작을 완료하고 모든 잠금을 릴리스하십시오. 이는 WITH HOLD로 열려진 모든 커서를 닫고 COMMIT를 발행하거나 ROLLBACK을 발행하여 수행할 수 있습니다.

임포트 유틸리티는 SQL INSERT문을 사용하여 목표 테이블에 행을 추가합니다. 이 유틸리티는 입력 파일에 있는 각 데이터 행마다 하나의 INSERT문을 발행합니다. INSERT 문을 실패하면 다음 두 가지 조치 중 하나가 발생합니다.

- 후속 INSERT문이 성공할 수 있을 것 같으면 메시지 파일에 경고 메시지를 작성하고 처리를 계속합니다.
- 후속 INSERT문이 실패할 것 같으며 데이터베이스 손상이 생길 가능성이 있으면 메시지 파일에 오류 메시지를 작성하고 처리를 정지합니다.

이 유틸리티는 **REPLACE** 또는 **REPLACE_CREATE** 조작 중 이전 행이 삭제된 후 자동 COMMIT를 수행합니다. 따라서 테이블 오브젝트가 잘려진 후 응용프로그램이 데이터베이스 관리 프로그램를 인터럽트하거나 시스템이 실패할 경우 이전 데이터가 모두 유실됩니다. 이들 옵션을 사용하기 전에 이전 데이터가 더 이상 필요하지 않은지 확인하십시오.

CREATE, **REPLACE** 또는 **REPLACE_CREATE** 조작 중 로그가 가득차게 되면 이 유틸리티는 삽입된 레코드에 대해 자동 COMMIT를 수행합니다. 자동 COMMIT 후 응용프로그램이 데이터베이스 관리 프로그램를 인터럽트하거나 시스템이 실패할 경우 부분 데이터가 있는 테이블이 데이터베이스에 남아 있습니다. **REPLACE** 또는 **REPLACE_CREATE** 옵션을 사용하여 전체 임포트 조작을 재실행하거나 성공적으로 임포트된 행 수로 설정된 **RESTARTCOUNT** 매개변수를 사용하여 **INSERT**를 사용하십시오.

디폴트로 **INSERT** 또는 **INSERT_UPDATE** 옵션에 대해서는 자동 COMMIT가 수행되지 않습니다. 단, **COMMITCOUNT** 매개변수가 0이 아닌 경우에는 수행됩니다. 자동 COMMIT가 수행되지 않으면 가득찬 로그는 ROLLBACK됩니다.

다음 조건 중 하나라도 해당될 경우 오프라인 임포트는 자동 COMMIT를 수행하지 않습니다.

- 목표가 테이블이 아니고 뷰인 경우
- 복합 텍스트 삽입이 사용된 경우
- 버퍼링된 삽입이 사용된 경우

디폴트로 온라인 임포트는 자동 COMMIT를 수행하여 사용 중인 로그 스페이스와 잠금 목록을 모두 해제합니다. 자동 COMMIT는 **COMMITCOUNT** 값이 0으로 지정된 경우에만 수행되지 않습니다.

임포트 유틸리티가 COMMIT를 수행할 때마다 메시지 파일에 두 개의 메시지가 작성되는데, 하나는 커미트되는 레코드 수를 나타내고 다른 하나는 COMMIT 완료 후 작성됩니다. 실패 후 임포트 조작을 재시작할 때 마지막 완료된 COMMIT에서 판별된 대로 생략할 레코드 수를 지정하십시오.

임포트 유틸리티는 사소한 비호환성 문제점이 있는 입력 데이터는 승인합니다(예를 들어, 문자 데이터가 채우기 또는 절단을 사용하여 임포트되고 숫자 데이터가 다른 숫자 데이터 유형으로 임포트될 수 있습니다). 그러나 주요한 비호환성 문제점이 있는 데이터는 승인되지 않습니다.

오브젝트 테이블이 자체 테이블 외에 하위 테이블이 있는 경우 이 오브젝트 테이블을 **REPLACE** 또는 **REPLACE_CREATE** 할 수 없으며, 기본 테이블에 하위 테이블(자체 테이블을 포함하여)이 있는 경우 해당 오브젝트 뷰를 REPLACE 또는 REPLACE_CREATE 할 수 없습니다. 이러한 테이블 또는 뷰를 바꾸려면 다음을 수행하십시오.

1. 해당 테이블이 상위 테이블인 모든 외부 키를 삭제하십시오.
2. 임포트 유틸리티를 실행하십시오.
3. 테이블을 변경하여 외부 키를 재작성하십시오.

외부 키를 재작성하는 동안 오류가 발생하면 데이터를 수정하여 참조 무결성을 유지보수하십시오.

PC/IXF 파일에서 테이블을 재작성할 때 참조 제한조건 및 외부 키 정의는 보존되지 않습니다. (기본 키 정의는 데이터가 이전에 SELECT *를 사용하여 익스포트된 경우에 *보존됩니다*.)

리모트 데이터베이스로 임포트하려면 입력 데이터 파일의 사본, 출력 메시지 파일 및 잠재적인 데이터베이스 *크기* 확장에 필요한 디스크 스페이스가 충분해야 합니다.

리모트 데이터베이스에 대해 임포트 조작이 수행되고 출력 메시지 파일이 매우 긴 경우(60KB 초과), 클라이언트의 사용자에게 리턴되는 메시지 파일은 임포트 조작 중간부터 메시지가 누락될 수 있습니다. 메시지 정보의 처음 30KB와 메시지 정보의 마지막 30KB는 항상 보유됩니다.

PC/IXF 파일이 디스켓 대신 하드 드라이브에 있을 경우 PC/IXF 파일을 리모트 데이터베이스로 임포트하면 훨씬 더 빨리 수행됩니다.

ASC, **DEL** 또는 **WSF** 파일 형식의 데이터를 임포트하려면 먼저 데이터베이스 테이블 또는 계층 구조가 존재해야 합니다. 그러나 테이블이 이미 존재하지 않을 경우 IMPORT **CREATE** 또는 IMPORT **REPLACE_CREATE**가 PC/IXF 파일에서 데이터를 임포트할 때 테이블을 작성합니다. 유형이 지정된 테이블에서 IMPORT **CREATE**는 자료형 계층 구조와 테이블 계층 구조를 작성할 수 있습니다.

데이터베이스 간에 데이터를 이동하려면(계층 데이터를 포함하여) PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다. 동일한 클라이언트에서 소스 또는 목표 데이터베이스에 모두 액세스할 수 있는 경우 파일 복사 단계는 필요하지 않습니다.

ASC 및 DEL 파일의 데이터는 임포트를 수행하는 클라이언트 응용프로그램의 코드 페이지에 있는 것으로 간주됩니다. 서로 다른 코드 페이지의 데이터를 임포트할 때는 서로 다른 코드 페이지에 사용할 수 있는 PC/IXF 파일을 사용할 것을 권장합니다. PC/IXF

파일과 임포트 유틸리티가 동일한 코드 페이지에 있을 경우 일반 응용프로그램에 대해서와 같은 처리가 발생합니다. 두 코드 페이지가 서로 다르고 **FORCEIN** 옵션이 지정된 경우, 임포트 유틸리티는 PC/IXF 파일의 데이터가 임포트를 수행 중인 응용프로그램과 동일한 코드 페이지를 가지고 있다고 간주합니다. 이는 두 코드 페이지에 대한 변환 테이블이 있는 경우에도 마찬가지입니다. 두 코드 페이지가 서로 다르고, **FORCEIN** 옵션이 지정되지 않았으며 변환 테이블이 있는 경우, PC/IXF 파일의 모든 데이터는 파일 코드 페이지에서 응용프로그램 코드 페이지로 변환됩니다. 두 코드 페이지가 서로 다르고 **FORCEIN** 옵션이 지정되지 않았으며 변환 테이블이 없는 경우 임포트 조작은 실패합니다. 이는 AIX 운영 체제의 DB2 클라이언트에 있는 PC/IXF 파일에만 적용됩니다.

1012 컬럼의 한계에 가까운 8KB 페이지에 있는 테이블 오브젝트의 경우, PC/IXF 데이터 파일을 임포트하면 DB2가 오류를 리턴할 수 있는데, 이는 SQL문의 최대 크기를 초과했기 때문입니다. 이러한 상황은 컬럼 유형이 CHAR, VARCHAR 또는 CLOB인 경우에만 발생할 수 있습니다. **DEL** 또는 **ASC** 파일을 임포트하는 데는 이러한 제한사항이 적용되지 않습니다. PC/IXF 파일을 사용하여 새 테이블을 작성할 경우 다른 방법은 db2look을 사용하여 테이블을 작성한 DDL문을 덤프한 후 CLP를 통해 이 명령문을 발행하는 것입니다.

DB2 Connect는 OS/390용 DB2, VM 및 VSE용 DB2, OS/400용 DB2와 같은 DRDA 서버로 데이터를 임포트하는 데 사용할 수 있습니다. PC/IXF 임포트(**INSERT** 옵션)만 지원됩니다. **RESTARTCOUNT** 매개변수도 지원되지만 **COMMITCOUNT** 매개변수는 지원되지 않습니다.

유형이 지정된 테이블에서 **CREATE** 옵션을 사용할 때 PC/IXF 파일에 정의된 모든 서브테이블을 작성하십시오. 서브테이블 정의는 변경할 수 없습니다. 유형이 지정된 테이블에서 **CREATE** 이외의 옵션을 사용할 때, 트래버스 순서 목록을 통해 해당 옵션이 트래버스 순서를 지정하므로 트래버스 순서 목록은 익스포트 조작 시 사용된 옵션과 일치해야 합니다. PC/IXF 파일 형식의 경우, 옵션은 목표 서브테이블 이름만 지정해야 하며 파일에 저장된 트래버스 순서를 사용해야 합니다.

임포트 유틸리티를 사용하여 이전에 PC/IXF 파일로 익스포트된 테이블을 복구할 수 있습니다. 이 테이블은 익스포트될 때의 상태로 리턴합니다.

데이터는 시스템 테이블, 작성된 임시 테이블, 선언된 임시 테이블 또는 요약 테이블로 임포트될 수 없습니다.

임포트 유틸리티를 통해 뷰를 작성할 수 없습니다.

개별적 파트가 Windows 시스템에서 AIX 시스템으로 복사되는 다중 파트 PC/IXF 파일 임포트가 지원됩니다. 첫 번째 파일의 이름만 IMPORT 명령에 지정되어야 합니다.

예를 들어, IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1. 파일 data.002, etc가 data.ixf와 동일한 디렉토리에서 사용 가능해야 합니다.

Windows 운영 체제에서:

- 논리적으로 분할된 PC/IXF 파일 임포트는 지원되지 않습니다.
- 잘못된 형식의 PC/IXF 또는 WSF 파일 임포트는 지원되지 않습니다.

내부 형식으로 된 보안 레이블에는 줄 바꾸기 문자가 포함될 수 있습니다. DEL 파일 형식을 사용하여 파일을 임포트할 경우 이들 줄 바꾸기 문자는 분리문자로 오인될 수 있습니다. 이러한 문제점이 발생하면 IMPORT 명령에 delprioritychar 파일 유형 수정자를 지정하여 분리문자에 대해 이전 디폴트 우선순위를 사용하십시오.

페더레이티드 고려사항

IMPORT 명령 및 **INSERT**, **UPDATE** 또는 **INSERT_UPDATE** 명령 매개변수를 사용할 때 참여 중인 별칭에 대해 CONTROL 특권을 가지고 있는지 확인해야 합니다. 임포트 조작을 수행할 때 사용할 별칭이 이미 존재하는지 확인해야 합니다. 또한 IMPORT 명령 매개변수 섹션에서 표시된 바와 같이 여러 가지 제한사항을 유념해야 합니다.

ODBC와 같은 일부 데이터 소스는 별칭으로의 임포트를 지원하지 않습니다.

임포트 유틸리티의 파일 유형 수정자

표 14. 임포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식

수정자	설명
compound=*x*	*x*는 1 - 100의 숫자입니다. nonatomic 복합 SQL을 사용하여 데이터를 삽입하며, *x* 명령문이 매번 시도됩니다. 수정자가 지정되고 트랜잭션 로그가 충분히 크지 않으면, 임포트 조작이 실패합니다. **COMMITCOUNT**에서 지정한 행 수나, **COMMITCOUNT**가 지정되지 않은 경우 데이터 파일의 행 수를 수용할 정도로 트랜잭션 로그가 커야 합니다. 그러므로 트랜잭션 로그 오버플로우를 피하기 위해 **COMMITCOUNT** 옵션을 지정하도록 권장합니다. 이 수정자는 **INSERT_UPDATE** 모드, 계층 테이블 및 다음 수정자와 호환되지 않습니다. usedefaults, identitymissing, identityignore, generatedmissing 및 generatedignore.
generatedignore	이 수정자는 생성된 모든 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 임포트 유틸리티에 알립니다. 이로 인해 생성된 컬럼의 모든 값이 유틸리티에 의해 생성됩니다. 이 수정자는 generatedmissing 수정자와 함께 사용될 수 없습니다.
generatedmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 생성된 컬럼의 데이터가 포함되지 않는 것으로(NULL 값도 비포함) 가정하므로 각 행의 값을 생성합니다. 이 수정자는 generatedignore 수정자와 함께 사용될 수 없습니다.

표 14. 임포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
identityignore	이 수정자는 ID 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 임포트 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ID 값이 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 identitymissing 수정자와 함께 사용될 수 없습니다.
identitymissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 ID 컬럼의 데이터가 포함되지 않는 것으로(NULL 값도 비포함) 가정하므로 각 행의 값을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 identityignore 수정자와 함께 사용될 수 없습니다.
lobsinfile	*lob-path*는 LOB 데이터를 포함하는 파일에 대한 경로를 지정합니다. 각 경로는 LLS(Lob Location Specifier)에 의해 포인트되는 최소한 하나의 LOB를 포함하는 최소한 하나의 파일을 데이터 파일에 포함합니다. LLS는 LOB 파일 경로에 저장된 파일에서 LOB 위치의 문자열 표시입니다. LLS 형식은 *filename.ext.nnn.mmm/*이며, 여기서 *filename.ext*는 LOB를 포함하는 파일의 이름이며, *nnn*은 파일 내에서 LOB의 오프셋을 바이트로 나타낸 것이며, mmm은 LOB의 길이를 바이트로 나타낸 것입니다. 예를 들어, 문자열 db2exp.001.123.456/가 데이터 파일에 저장되는 경우, LOB는 db2exp.001 파일에서 오프셋 123에 위치하며 456바이트 길이입니다. **LOBS FROM** 절은 『lobsinfile』 수정자가 사용될 때 LOB 파일이 위치하는 곳을 지정합니다. **LOBS FROM** 절은 내재적으로 LOBSINFILE 동작을 활성화합니다. **LOBS FROM** 절은 데이터 임포트 중 LOB 파일을 검색하기 위해 경로 목록을 IMPORT 유틸리티로 전달합니다. 널(NULL) LOB를 표시하려면, *크기*를 -1로 입력하십시오. *크기*가 0으로 지정되면, 길이가 0인 LOB로 처리됩니다. 길이가 -1인 널(NULL) LOBS의 경우, 오프셋 및 파일 이름은 무시됩니다. 예를 들어, 널(NULL) LOB의 LLS는 db2exp.001.7.-1/입니다.
no_type_id	단일 서브테이블로 임포트할 때에만 유효합니다. 일반 설치는 일반 테이블에서 데이터를 익스포트한 후 임포트 조작을 호출하여(이 수정자 사용) 데이터를 단일 서브테이블로 변환합니다.
nodefaults	목표 테이블 컬럼의 소스 컬럼이 명시적으로 지정되지 않고, 테이블 컬럼이 널(NULL) 입력 가능하지 않으면, 디폴트값이 로드되지 않습니다. 목표 테이블 컬럼 중 하나의 소스 컬럼이 명시적으로 지정되지 않은 경우, 이 옵션이 없으면 다음 중 하나가 발생합니다. • 컬럼의 디폴트값이 지정될 수 있는 경우, 디폴트값이 로드됩니다. • 컬럼이 널(NULL) 입력 가능하고 해당 컬럼의 디폴트값을 지정할 수 없는 경우, NULL이 로드됩니다. • 컬럼이 널(NULL) 입력 가능하고 디폴트값을 지정할 수 없는 경우, 오류가 리턴되며 유틸리티가 처리를 중지합니다.
norowwarnings	거부된 행에 대한 모든 경고를 제외시킵니다.
rowchangetimestampignore	이 수정자는 행 변경 시간소인 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 임포트 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ROW CHANGE TIMESTAMP가 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 rowchangetimestampmissing 수정자와 함께 사용될 수 없습니다.
rowchangetimestampmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 행 변경 시간소인 컬럼의 데이터를 포함하지 않음(NULL 값도 비포함)을 가정하므로 각 행의 값을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 rowchangetimestampignore 수정자와 함께 사용될 수 없습니다.

표 14. 임포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
seclabelchar	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 보안 레이블 값의 문자열 형식임을 표시합니다. IMPORT는 각 보안 레이블을 로드된 대로의 내부 형식으로 변환합니다. 문자열이 적절한 형식으로 되어 있지 않은 경우 행은 로드되지 않으며 경고(SQLSTATE 01H53)가 리턴됩니다. 문자열이 테이블을 보호하는 보안 규정 파트인 유효한 보안 레이블을 나타내지 않는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3243W)가 리턴됩니다. seclabelname 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 임포트에 실패하며 오류(SQLCODE SQL3525N)가 리턴됩니다.
seclabelname	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 이름으로 표시됨을 나타냅니다. IMPORT는 이름이 있는 경우 적절한 보안 레이블로 변환합니다. 테이블을 보호하는 보안 규정에 대해 표시된 이름이 있는 보안 레이블이 없는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3244W)가 리턴됩니다. seclabelchar 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 임포트에 실패하며 오류(SQLCODE SQL3525N) 가 리턴됩니다. **주:** 파일 유형이 ASC인 경우, 보안 레이블의 이름 다음에 오는 모든 스페이스는 이름의 일부로 해석됩니다. 이를 피하려면 striptblanks 파일 유형 수정자를 사용하여 스페이스가 제거되었는지 확인하십시오.
usedefaults	목표 테이블 컬럼의 소스 컬럼이 지정되었지만 하나 이상의 행 인스턴스에 대한 데이터를 포함하지 않은 경우, 디폴트값이 로드됩니다. 누락된 데이터의 예: • DEL 파일: 임의 수의 스페이스(″, ,″)로 구분되는 2개의 인접 컬럼 분리문자나 2개의 인접 컬럼 분리문자(″,,″)가 컬럼 값으로 지정됩니다. • DEL/ASC/WSF 파일: 행의 컬럼이 충분하지 않거나 행의 길이가 원래 스펙만큼 충분하지 않습니다. **주:** ASC 파일의 경우, NULL 컬럼 값은 명시적으로 누락된 것으로 간주되지 않으며 디폴트가 NULL 컬럼 값을 대신하지 않습니다. 숫자, 날짜, 시간 및 /시간소인 컬럼의 모든 공백 문자로 NULL 컬럼 값을 표시하거나, 컬럼이 NULL임을 표시하기 위해 모든 유형의 컬럼에 널(NULL) 표시기를 사용함으로써 NULL 컬럼 값을 표시합니다. 이 옵션이 없는 경우, 소스 컬럼이 행 인스턴스의 데이터를 포함하지 않으면 다음 중 하나가 발생합니다. • DEL/ASC/WSF 파일: 컬럼에 널(NULL) 입력 가능한 경우 NULL이 로드됩니다. 컬럼이 널(NULL) 입력 가능하지 않으면 유틸리티가 행을 거부합니다.

표 15. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL)

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 입력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 임포트 조작 중 이 코드 페이지로부터 응용프로그램 코드 페이지로 문자 데이터를 변환합니다. 다음 규칙이 적용됩니다. • 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. • nullindchar은 x20 - x7F 코드 포인트의 표준 ASCII 세트에 포함된 기호를 지정해야 합니다. 이것은 ASCII 기호 및 코드 포인트를 나타냅니다. **주:** 1. codepage 수정자는 lobsinfile 수정자와 함께 사용될 수 없습니다. 2. 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 코드 페이지를 변환할 때 데이터 확장이 발생하면, 데이터가 절단되고 데이터 유실이 발생할 수 있습니다.
dateformat="*x*"	*x*는 소스 파일에서 날짜의 형식입니다.[2] 유효한 날짜 요소는 다음과 같습니다. YYYY - 연도(0000 - 9999 사이의 4자리 숫자) M - 월(1 - 12 사이의 1 또는 2자리 숫자) MM - 월(1 - 12 사이의 2자리 숫자, M과 상호 배타적) D - 일(1 - 31 사이의 1 또는 2자리 숫자) DD - 일(1 - 31 범위의 2자리 숫자, D와 상호 배타적) DDD - 년의 일(001 - 366 범위의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적) 지정되지 않은 각 요소에 대해 디폴트값 1이 지정됩니다. 날짜 형식의 예: "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
implieddecimal	내포된 소수점의 위치는 컬럼 정의로 판별되며, 값의 끝으로 가정되지 않습니다. 예를 들어, 12345 값은 12345.00이 *아닌* 123.45로 DECIMAL(8,2) 컬럼에 로드됩니다.

표 15. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
timeformat="*x*"	*x*는 소스 파일에서 시간의 형식입니다.[2] 유효한 시간 요소는 다음과 같습니다. H - 시간(12시간 시스템의 경우 0 - 12 범위의, 1 또는 2자리 숫자 24시간 시스템의 경우 0 - 24) HH - 시간(12시간 시스템의 경우 0 - 12 범위의 2자리 숫자, 24시간 시스템의 경우 0 - 24. H와 상호 배타적) M - 분(0 - 59 사이의 1 또는 2자리 숫자) MM - 분(0 - 59 사이의 2자리 숫자, M과 상호 배타적) S - 초(0 - 59 사이의 1 또는 2자리 숫자) SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적) SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적) TT - 오전/오후 지시(AM 또는 PM) 지정되지 않은 각 요소에 대해 디폴트값 0이 지정됩니다. 시간 형식의 예: "HH:MM:SS" "HH.MM TT" "SSSSS"

표 15. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명

timestampformat="*x*"	*x*는 소스 파일에서 시간소인의 형식입니다.[2] 유효한 시간소인 요소는 다음과 같습니다.
	
	YYYY - 연도(0000 - 9999 사이의 4자리 숫자)
	M - 월(1 - 12 사이의 1 또는 2자리 숫자)
	MM - 월(01 - 12 사이의 2자리 숫자, M 및 MMM은 상호 배타적)
	MMM - 월(월 이름으로 사용되는 3자리 문자로 대소문자를 구분하지 않으며, M 및 MM은 상호 배타적)
	D - 일(1 - 31 사이의 1 또는 2자리 숫자)
	DD - 일(1 - 31 사이의 2자리 숫자, D와는 상호 배타적)
	DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적)
	H - 시간(12시간 시스템의 경우 0 - 12 범위의 1 또는 2자리 숫자, 24시간 시스템의 경우 0 - 24)
	HH - 시간(12시간 시스템의 경우 0 - 12 범위의 2자리 숫자, 24시간 시스템의 경우 0 - 24, H와는 상호 배타적)
	M - 분(0 - 59 사이의 1 또는 2자리 숫자)
	MM - 분(0 - 59 사이의 2자리 숫자, M, 분과 상호 배타적)
	S - 초(0 - 59 사이의 1 또는 2자리 숫자)
	SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적)
	SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적)
	U(1 - 12회) - 초의 소수 부분(U 어커런스 수는 각 자리가 0 - 9 사이의 자리 수를 나타냄)
	TT - 오전/오후 지시(AM 또는 PM)
	
	디폴트값 1이 미지정된 YYYY, M, MM, D, DD 또는 DDD 요소에 지정됩니다. 디폴트값 'Jan'이 미지정된 MMM 요소에 지정됩니다. 미지정된 다른 모든 요소에 디폴트값 0이 지정됩니다. 다음은 시간소인 형식의 예입니다.
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 요소의 올바른 값은 다음을 포함합니다. 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' 및 'dec'. 이들 값은 대소문자가 구분되지 않습니다.
	
	다음 예는 사용자 정의 날짜 및 시간 형식을 포함하는 데이터를 schedule이라는 테이블로 임포트하는 방법을 설명합니다.
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

표 15. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
usegraphiccodepage	usegraphiccodepage가 제공되면, 그래픽 또는 2바이트 문자 대형 오브젝트(DBCLOB) 데이터 필드로 임포트되는 데이터는 그래픽 코드 페이지에 있음이 가정됩니다. 나머지 데이터는 문자 코드 페이지에 있다고 가정합니다. 그래픽 코드 페이지는 문자 코드 페이지와 연관됩니다. IMPORT는 codepage 수정자가 지정된 경우 이를 통해 문자 코드 페이지를 판별하고 codepage 수정자가 지정되지 않은 경우 응용프로그램의 코드 페이지를 통해 문자 코드 페이지를 판별합니다. 복구 중인 테이블에 그래픽 데이터가 있는 경우에만 삭제(drop) 테이블 복구로 생성되는 구분된 데이터 파일과 결합하여 이 수정자를 사용해야 합니다. **제한사항** 이들 파일이 오직 하나의 코드 페이지에 인코드된 데이터를 포함하면, usegraphiccodepage 수정자는 EXPORT 유틸리티로 작성된 DEL 파일과 함께 지정되지 않아야 합니다. usegraphiccodepage 수정자는 파일의 2바이트 문자 대형 오브젝트(DBCLOB)에서 무시됩니다.
xmlchar	XML 문서가 문자 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 지정된 문자 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 문자 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 문자 코드 페이지는 codepage 파일 유형 수정자에서 지정한 값이거나, 지정되지 않은 경우 응용프로그램 코드 페이지입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다.
xmlgraphic	XML 문서가 지정된 그래픽 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 특정 그래픽 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 그래픽 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 그래픽 코드 페이지는 codepage 파일 유형 수정자에서 지정한 값의 그래픽 구성요소이거나, 지정되지 않은 경우 응용프로그램 코드 페이지의 그래픽 구성요소입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다. **주:** xmlgraphic 수정자가 IMPORT 명령으로 지정된 경우, 임포트되는 XML 문서는 UTF-16 코드 페이지로 인코드되어야 합니다. 그렇지 않으면, XML 문서는 구문 분석 오류로 거부되거나 데이터 손상이 있는 테이블로 임포트될 수 있습니다.

표 16. 임포트 유틸리티의 유효한 파일 유형 수정자: ASC(컬럼 식별자가 없는 ASCII) 파일 형식

수정자	설명
nochecklengths	nochecklengths가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 임포트하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 임포트될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.

표 16. 임포트 유틸리티의 유효한 파일 유형 수정자: ASC(컬럼 식별자가 없는 ASCII) 파일 형식 (계속)

수정자	설명
nullindchar=*x*	*x*는 단일 문자입니다. 널(NULL) 값을 나타내는 문자를 *x*로 변경합니다. *x*의 디폴트값은 Y입니다.[3] 문자가 영문자인 경우를 제외하고 EBCDIC 데이터 파일의 경우 이 수정자의 대소문자를 구분합니다. 예를 들어, 널(NULL) 표시기 문자가 N 문자가 되도록 지정되는 경우, n은 널(NULL) 표시기로 인식됩니다.
reclen=*x*	*x*는 최대값이 32,767인 정수입니다. 각 행에 대해 *x* 문자가 읽혀지며 행의 끝을 표시하기 위한 줄 바꾸기 문자는 사용되지 않습니다.

striptblanks	데이터를 변수 길이 필드로 로드할 때 뒤 공백을 절단합니다. 이 옵션이 지정되지 않으면, 공백이 보존됩니다.
	
	다음 예에서, striptblanks를 사용하면 임포트 유틸리티가 뒤 공백을 절단합니다.
	`db2 import from myfile.asc of asc`
	`modified by striptblanks`
	`method l (1 10, 12 15) messages msgs.txt`
	`insert into staff`
	
	이 옵션은 striptnulls와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다.이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 t 옵션을 교체합니다.

striptnulls	데이터를 변수 길이 필드로 로드할 때 뒤 NULL 값(0x00 문자)을 절단합니다. 이 옵션이 지정되지 않으면, NULL 값이 보존됩니다. 이 옵션은 striptblanks와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다. 이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 padwithzero 옵션을 교체합니다.

표 17. 임포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식

수정자	설명

chardel*x*	*x*는 단일 문자열 분리문자입니다. 디폴트값은 큰따옴표(")입니다. 큰따옴표 대신 지정된 문자를 사용하여 문자열을 묶습니다.[3,4] 명시적으로 큰따옴표를 문자열 분리문자로 지정하려는 경우 다음과 같이 지정해야 합니다.
	`modified by chardel""`
	
	작은따옴표(')를 문자열 분리문자로 지정할 수도 있습니다. 다음 예에서, chardel''는 임포트 유틸리티가 작은따옴표를 문자열 분리문자로 해석하게 합니다.
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x*는 단일 문자 컬럼 분리문자입니다. 디폴트값은 쉼표(,)입니다. 지정된 문자가 쉼표 대신 사용되어 컬럼 끝을 나타냅니다.[3,4]
	
	다음 예에서, coldel;은 임포트 유틸리티가 세미콜론을 컬럼 분리문자로 해석하게 합니다.
	`db2 import from myfile.del of del`
	`modified by coldel;`
	`messages msgs.txt insert into staff`

decplusblank	플러스 부호 문자. 플러스 부호(+) 대신 공백을 양의 10진수 값 앞에 붙입니다. 디폴트 조치는 플러스 부호를 양의 10진수 값 앞에 붙이는 것입니다.

표 17. 임포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식 (계속)

수정자	설명

decpt*x*	*x*는 소수점 문자로 마침표를 대신하는 단일 문자입니다. 디폴트값은 마침표(.)입니다. 마침표 대신 지정된 문자가 소수점 문자로 사용됩니다.[34]
	
	다음 예에서, `decpt;`는 임포트 유틸리티가 세미콜론(;)을 소수점으로 해석하게 합니다.
	
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`decpt; messages msgs.txt insert into staff"`

delprioritychar	분리문자의 현재 디폴트 우선순위: 레코드 구분 문자, 문자 분리문자, 컬럼 분리문자. 이 수정자는 분리문자 우선순위를 문자 분리문자, 레코드 구분 문자, 컬럼 분리문자로 되돌림으로써 이전 우선순위에 따른 기존 응용프로그램을 보호합니다. 구문:
	
	`db2 import ... modified by delprioritychar ...`
	
	예를 들면, 다음과 같은 DEL 데이터 파일이 있습니다.
	
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	
	`delprioritychar` 수정자를 지정하고, 이 데이터 파일에는 두 개의 행만이 있습니다. 두 번째 <행 분리문자>는 두 번째 행의 첫 번째 데이터 컬럼의 파트로 해석되지만, 첫 번째 및 세 번째 <행 분리문자>는 실제 레코드 구분 문자로 해석됩니다. 이 수정자가 지정되지 않은 경우, 이 데이터 파일에는 세 개의 행이 있으며, 각 행은 <행 분리문자>로 구분됩니다.

keepblanks	유형 CHAR, VARCHAR, LONG VARCHAR 또는 CLOB의 각 필드에 앞뒤 공백을 둡니다. 이 옵션이 없으면, 문자 분리문자 내에 없는 앞 공백과 뒤 공백은 모두 제거되며 공백 필드의 테이블로 NULL이 삽입됩니다.
nochardel	임포트 유틸리티는 컬럼 분리문자 간의 모든 바이트를 컬럼 데이터의 파트로 가정합니다. 문자 분리문자는 컬럼 데이터의 파트로 구문 분석됩니다. DB2를 사용하여 데이터를 익스포트한 경우(익스포트 시 nochardel을 지정한 경우를 제외하고) 이 옵션을 지정하지 말아야 합니다. 이 옵션은 문자 분리문자가 없는 벤더 데이터 파일을 지원하기 위해 제공됩니다. 부적절한 사용은 데이터 유실이나 손상을 초래할 수 있습니다. 이 옵션은 `chardelx`, `delprioritychar` 또는 `nodoubledel`과 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다.
nodoubledel	2바이트 분리문자를 인식하지 않습니다.

표 18. 임포트 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식

수정자	설명
forcein	코드 페이지 불일치와 관계없이 데이터를 승인하고 코드 페이지 간에 변환하지 않도록 유틸리티에 지시합니다. 데이터의 고정 길이 대상 필드가 충분히 큰지 검증하도록 고정 길이 대상 필드를 점검합니다. `nochecklengths`가 지정된 경우, 검사가 수행되지 않으며 각 행을 임포트하려고 시도합니다.
indexixf	기존 테이블에 현재 정의된 모든 인덱스를 삭제(drop)하고 PC/IXF 파일의 인덱스 정의에서 새로운 인덱스를 작성하도록 유틸리티에 지시합니다. 이 옵션은 목차가 교체될 때에만 사용될 수 있습니다. 뷰 또는 *insert-column*와 함께 사용될 수 없습니다.
indexschema=*schema*	색인 작성 중 인덱스 이름에 지정된 *schema*를 사용합니다. *schema*가 지정되지 않은 경우(그러나 키워드 `indexschema`가 지정됨), 연결 사용자 ID를 사용합니다. 해당 키워드가 지정되지 않은 경우, IXF 파일에서 스키마를 사용합니다.

표 18. 임포트 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식 (계속)

수정자	설명
nochecklengths	nochecklengths가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 임포트하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 임포트될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.
forcecreate	임포트 조작 중 SQL3311N 리턴 후에 제한된 정보나 가능한 누락 정보로 테이블이 작성되어야 함을 지정합니다.

표 19. codepage 및 usegraphiccodepage 사용 시 IMPORT 동작

codepage=N	usegraphiccodepage	IMPORT 동작
Absent	Absent	파일의 모든 데이터는 응용프로그램 코드 페이지에 있는 것으로 가정됩니다.
Present	Absent	파일의 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 임포트될 때 그래픽 데이터가 손상됩니다.
Absent	Present	파일의 문자 데이터는 응용프로그램 코드 페이지에 있는 것으로 가정됩니다. 그래픽 데이터는 응용프로그램 그래픽 데이터의 코드 페이지에 있는 것으로 가정됩니다. 응용프로그램 코드 페이지가 1바이트이면, 모든 데이터는 응용프로그램 코드 페이지에 있는 것으로 가정됩니다. **경고:** 응용프로그램 코드 페이지가 1바이트이면, 데이터베이스가 그래픽 컬럼을 포함해도 데이터베이스로 임포트될 때 그래픽 데이터가 손상됩니다.
Present	Present	문자 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. 그래픽 데이터는 N의 그래픽 코드 페이지에 있는 것으로 가정됩니다. N이 1바이트 또는 2바이트 코드 페이지인 경우, 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 임포트될 때 그래픽 데이터가 손상됩니다.

주:

1. **MODIFIED BY** 옵션과 함께 지원되지 않는 파일 유형을 사용하려고 시도하는 경우 임포트 유틸리티는 경고를 발행하지 않습니다. 이런 경우, 임포트 조작에 실패하며 오류 코드가 리턴됩니다.
2. 날짜 출력 문자열을 둘러싼 큰따옴표는 필수입니다. 필드 구분자는 a - z, A - Z 및 0 - 9를 포함할 수 없습니다. 필드 구분자는 DEL 파일 형식의 필드 분리문자나 문자 분리문자와 같지 않아야 합니다. 요소의 시작 및 종료 위치가 명확한 경우 필드 구분자가 선택적입니다. D, H, M 또는 S와 같은 요소가 사용되는 경우 (수정자에 따라) 항목의 변수 길이 때문에 모호함이 있을 수 있습니다.

시간소인 형식의 경우, 문자 M을 사용하는 month 및 minute 디스크립터 간의 모호함을 피하려면 주의해야 합니다. month 필드는 다른 날짜 필드와 인접해야 합니다. minute 필드는 다른 시간 필드와 인접해야 합니다. 다음은 모호한 시간소인 형식입니다.

```
"M"(month 또는 minute일 수 있음)
"M:M"(month 및 minute 구분 가능?)
"M:YYYY:M"(둘 다 month로 해석됨)
"S:M:YYYY"(둘 다 시간 값 및 날짜 값에 인접)
```

모호한 경우, 유틸리티는 오류 메시지를 발행하며, 조작에 실패합니다.

다음은 명확한 시간소인 형식입니다.

```
"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month....Minute)
"M:H:YYYY:M:D" (Minute....Month)
```

큰따옴표 및 백슬래시와 같은 일부 문자는 Escape 문자(예: ₩)가 앞에 와야 합니다.

3. chardel, coldel 또는 decpt 파일 유형에 제공되는 문자 값은 소스 데이터의 코드 페이지에 지정되어야 합니다.

   문자 코드 포인트(문자 기호 대신)는 구문 xJJ 또는 0xJJ를 사용하여 지정될 수 있으며, 여기서 JJ는 코드 포인트의 16진수를 나타냅니다. 예를 들어, 컬럼 분리문자로 # 문자를 지정하려면, 다음 중 하나를 사용하십시오.

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

4. *데이터 이동을 위한 분리문자 고려사항*은 분리문자를 겹쳐쓰기할 때 사용할 수 있는 문자에 적용하는 제한사항을 나열합니다.
5. 별칭으로 임포트할 때 다음 파일 유형 수정자는 허용되지 않습니다.
   - indexixf
   - indexschema
   - dldelfiletype
   - nodefaults
   - usedefaults
   - no_type_idfiletype
   - generatedignore
   - generatedmissing
   - identityignore
   - identitymissing

- lobsinfile

6. **WSF** 파일 형식은 XML 컬럼에서 지원되지 않습니다. 또한 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.
7. **CREATE** 모드는 XML 컬럼에서 지원되지 않습니다.
8. 모든 XML 데이터는 주 데이터 파일에서 분리된 XML 파일에 있어야 합니다. 주 데이터 파일의 각 XML 컬럼에 대해 XDS(XML Data Specifier)(또는 NULL 값)가 있어야 합니다.
9. XMLCHAR 또는 XMLGRAPHIC 파일 유형 수정자가 지정된 경우를 제외하고, XML 문서는 유니코드 형식으로 되어 있거나 인코딩 속성을 포함하는 선언 태그를 포함한다고 가정합니다.
10. 잘 양식화되지 않은 문서를 포함하는 행은 거부됩니다.
11. **XMLVALIDATE** 옵션이 지정된 경우, 일치하는 스키마에 대해 정상적으로 유효성을 확인하는 문서는 삽입될 때 스키마 정보로 주석을 표시합니다. 일치하는 스키마에 대해 유효성을 확인하는 데 실패하는 문서를 포함하는 행은 거부됩니다. 정상적으로 유효성을 확인하려면, 임포트를 호출하는 사용자가 보유한 특권이 최소한 다음 중 하나를 포함해야 합니다.
    - DBADM 권한
    - 유효성 확인에 사용되는 XML 스키마의 USAGE 특권
12. 내재적으로 숨겨진 행 변경 시간소인 컬럼을 포함하는 테이블로 임포트할 때, 내재적으로 숨겨진 컬럼의 등록 정보는 무시됩니다. 그러므로, 컬럼의 데이터가 임포트되는 데이터에 표시되지 않으며 명시적 컬럼 목록이 표시되지 않으면 rowchangetimestampmissing 파일 유형 수정자가 임포트 명령에 지정*되어야* 합니다.

ADMIN_CMD 프로시저를 사용하는 IMPORT 명령

지원되는 파일 형식을 갖는 외부 파일의 데이터를 테이블, 계층 구조, 뷰 또는 별칭으로 데이터를 삽입합니다. LOAD가 더 빠른 방법이기는 하지만 로드 유틸리티는 계층 구조 레벨에서 데이터 로드를 지원하지 않습니다.

116 페이지의 『임포트 유틸리티의 파일 유형 수정자』로의 빠른 링크를 참조하십시오.

권한 부여

- **INSERT** 옵션을 사용하여 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 각 참여 중인 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT 및 SELECT 특권

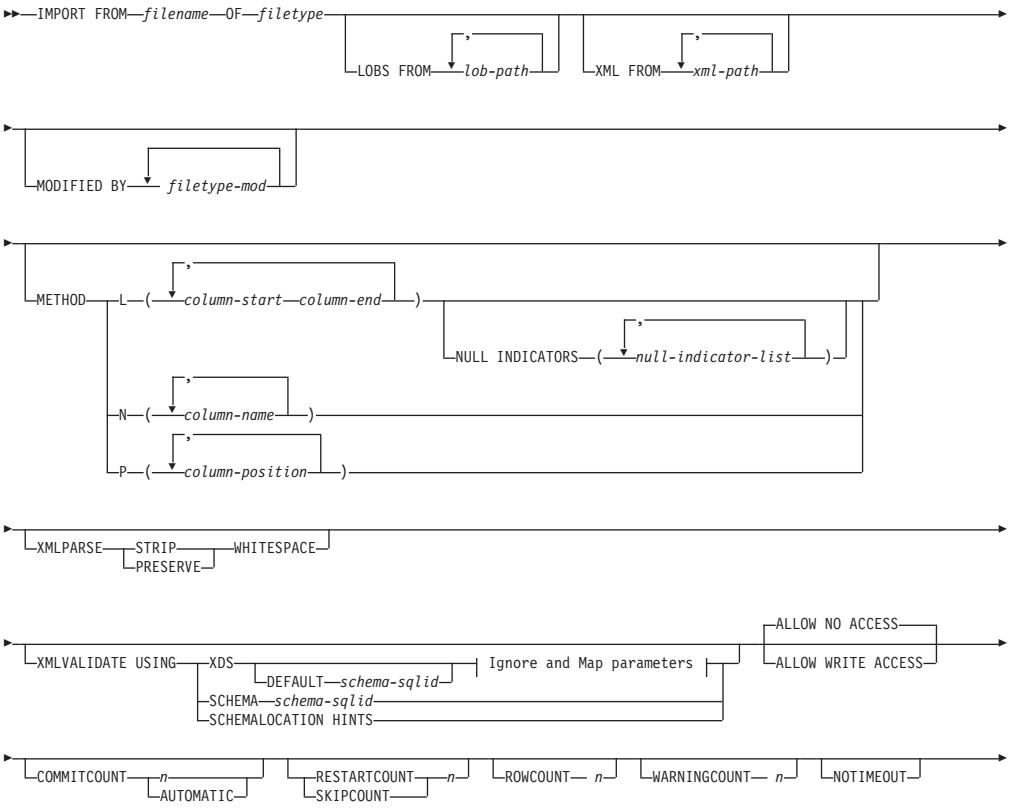
- **INSERT_UPDATE** 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 각 참여 중인 테이블, 뷰 또는 별칭에 대한 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT, SELECT, UPDATE 및 DELETE 특권
- **REPLACE** 또는 **REPLACE_CREATE** 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 테이블 또는 뷰에 대한 CONTROL 특권
  - 테이블 또는 뷰에 대한 INSERT, SELECT 및 DELETE 특권
- **CREATE** 또는 **REPLACE_CREATE** 옵션을 사용하여 새 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dbadm* 권한
  - 데이터베이스에 대한 CREATETAB 권한 및 테이블 스페이스에 대한 USE 특권은 다음 경우 중 하나입니다.
    - 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
- **CREATE** 또는 **REPLACE_CREATE** 옵션을 사용하여 존재하지 않는 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dbadm* 권한
  - 데이터베이스에 대한 CREATETAB 권한과 테이블 스페이스에 대한 USE 특권 및 다음 중 하나가 필요합니다.
    - 테이블의 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마가 존재할 경우, 스키마에 대한 CREATEIN 특권
    - 전체 계층 구조에 대한 **REPLACE_CREATE** 옵션이 사용될 경우, 계층 구조의 모든 서브테이블에 대한 CONTROL 특권
- **REPLACE** 옵션을 사용하여 기존 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - *dataaccess* 권한
  - 계층 구조에 있는 모든 서브테이블에 대한 CONTROL 특권

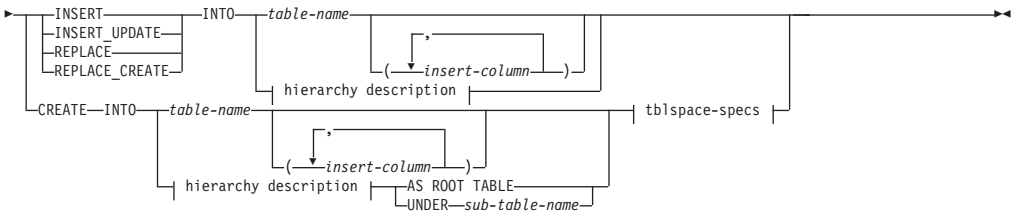
- 보호 컬럼이 있는 테이블로 데이터를 임포트하려면 테이블의 모든 보호 컬럼에 대한 쓰기 액세스를 허용하는 LBAC 증명서가 세션 권한 부여 ID에 있어야 합니다. 그렇지 않으면 임포트를 실패하고 오류(SQLSTATE 42512)가 리턴됩니다.
- 행이 보호 설정된 테이블로 데이터를 임포트하려면 세션 권한 부여 ID가 다음 기준을 충족하는 LBAC 증명서를 보유해야 합니다.
  - 테이블을 보호하는 보안 규정에 포함됨
  - 세션 권한 부여 ID에 쓰기 액세스에 대한 권한이 부여됨

  삽입할 행에 대한 레이블, 사용자의 LBAC 신임, 보안 규정 정의 및 LBAC 규칙이 행에 대한 레이블을 판별합니다.
- **REPLACE** 또는 **REPLACE_CREATE** 옵션을 지정할 경우 세션 권한 부여 ID는 테이블을 삭제할 수 있는 권한이 있어야 합니다.
- 데이터를 별칭으로 임포트하려면 세션 권한 부여 ID에 pass-through 모드에서 지정된 데이터 소스를 액세스하고 사용할 특권이 있어야 합니다.

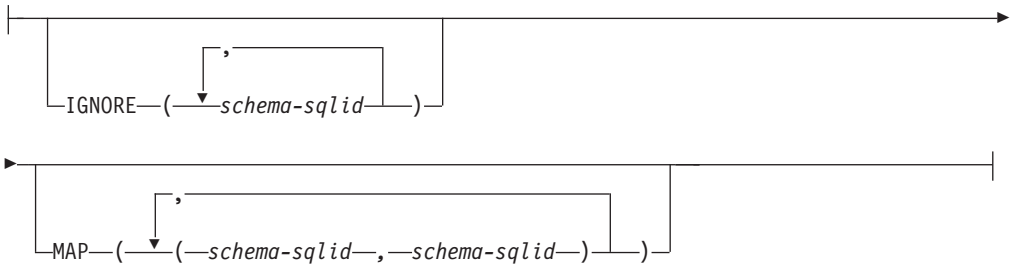
필수 연결

명령 구문



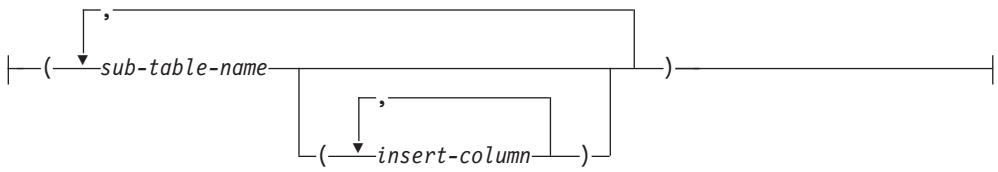


Ignore and Map parameters:



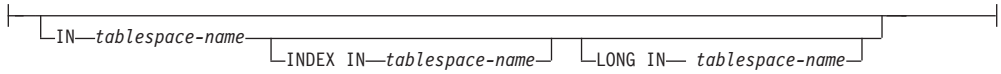
hierarchy description:

sub-table-list:



traversal-order-list:

tblspace-specs:



명령 매개변수

ALL TABLES

계층 구조 전용 내재적 키워드. 계층 구조를 임포트할 때 디폴트로 트래버스 순서로 지정된 모든 테이블을 임포트합니다.

ALLOW NO ACCESS

오프라인 모드로 임포트를 실행합니다. 행이 삽입되기 전에 목표 테이블에 대해 독점(X) 잠금을 획득합니다. 이렇게 하면 동시 응용프로그램이 테이블 데이터에 액세스하지 못하게 합니다. 이는 디폴트 임포트 동작입니다.

ALLOW WRITE ACCESS

온라인 모드로 임포트를 실행합니다. 첫 번째 행이 삽입될 때 목표 테이블에 대해 의도를 가진 독점(IX) 잠금을 획득합니다. 이렇게 하면 동시 판독기 및 기록기가 테이블 데이터에 액세스할 수 있습니다. 온라인 모드는 **REPLACE**, **CREATE** 또는 **REPLACE_CREATE** 임포트 옵션과 호환 가능하지 않습니다. 온라인 모드는 버퍼링된 삽입과 함께 지원되지 않습니다. 임포트 조작은 테이블 잠금에 대한 잠금 에스컬레이션을 방지하고 사용 중인 로그 스페이스 외부에서 실행되지 않도록 하기 위해 삽입된 데이터를 주기적으로 커미트합니다. 이러한 커미트는 **COMMITCOUNT** 옵션을 사용하지 않는 경우에도 수행됩니다. 각 커미트를 수행하는 동안 임포트는 해당 IX 테이블 잠금을 유실하므로 커미트 후 이를 다시 획득하려고 시도합니다. 별칭으로 임포트할 때 이 매개변수가 필요하며 유효한 숫자를 사용하여 **COMMITCOUNT**를 지정해 야 합니다(AUTOMATIC은 유효한 옵션으로 간주하지 않음).

AS ROOT TABLE

하나 이상의 서브테이블을 독립형 테이블 계층 구조로 작성합니다.

COMMITCOUNT *n* **| AUTOMATIC**

*n*개의 레코드가 모두 임포트된 후 COMMIT를 수행합니다. 숫자 *n*을 지정하면 임포트는 *n*개의 레코드가 모두 임포트된 후 COMMIT를 수행합니다. 복합 삽입을 사용할 때, 사용자 지정 커미트 빈도 *n*은 복합 계수 값의 첫 번째 정수 배수로 반올림됩니다. AUTOMATIC을 지정하면 임포트는 내부적으로 커미트를 수행해야 하는 때를 판별합니다. 이 유틸리티는 다음 두 가지 이유 중 하나에 대해 커미트합니다.

- 사용 중인 로그 스페이스 외부에서 실행하지 않도록 하기 위해
- 행 레벨에서 테이블 레벨로 잠금 에스컬레이션되지 않도록 하기 위해

ALLOW WRITE ACCESS 옵션을 지정하고 **COMMITCOUNT** 옵션을 지정하지 않으면, 임포트 유틸리티는 **COMMITCOUNT** AUTOMATIC이 지정된 것과 같이 커미트를 수행합니다.

사용 중인 로그 스페이스를 모두 사용해버리지 않도록 하기 위한 임포트 조작의 기능은 DB2 레지스트리 변수 **DB2_FORCE_APP_ON_MAX_LOG**에 의해 영향을 받습니다.

- **DB2_FORCE_APP_ON_MAX_LOG**가 FALSE로 설정되고 **COMMITCOUNT** AUTOMATIC 명령 옵션이 지정된 경우, 임포트 유틸리티는 자동으로 사용 중인 로그 스페이스를 모두 사용해버리지 않도록 할 수 있습니다.
- **DB2_FORCE_APP_ON_MAX_LOG**가 FALSE로 설정되고 **COMMITCOUNT** *n* 명령 옵션이 지정된 경우, 레코드를 삽입하거나 갱신하는 중 SQL0964C(트랜잭션 로그가 가득참) 발생 시 임포트 유틸리티는 로그 가득참 조건을 해결하려고 합니다. 무조건 커미트를 수행한 후 레코드를 삽입하거나 갱신하려고 재시도합니다. 이것이 로그 가득참 조건(로그 가득참이 데이터베이스의 다른 활동에 영향을 주는 경우)을 해결하는 데 유용하지 않을 경우, IMPORT 명령은 예상대로 실패합니다. 그러나 커미트되는 행 수는 **COMMITCOUNT** *n* 값 의 배수가 되지 않을 수 있습니다. 임포트 조작 재시도 시 이미 커미트된 행 처리를 피하려면, **RESTARTCOUNT** 또는 **SKIPCOUNT** 명령 매개변수를 사용하십시오.
- **DB2_FORCE_APP_ON_MAX_LOG**가 TRUE로 설정된 경우(디폴트), 레코드를 삽입하거나 갱신하는 중 SQL0964C 발생 시 임포트 조작이 실패합니다. **COMMITCOUNT** AUTOMATIC 또는 **COMMITCOUNT** *n* 지정 여부와 관계 없이 이런 현상이 일어날 수 있습니다.

  응용프로그램에서 데이터베이스가 강제 해제되고 현재 작업 단위(UOW)가 롤백됩니다. 임포트 조작 재시도 시 이미 커미트된 행 처리를 피하려면, **RESTARTCOUNT** 또는 **SKIPCOUNT** 명령 매개변수를 사용하십시오.

CREATE

주: CREATE 매개변수는 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항은 『IMPORT 명령 옵션 **CREATE** 및 **REPLACE_CREATE**가 사용되지 않음』을 참조하십시오.

데이터베이스의 코드 페이지에 테이블 정의 및 행 내용을 작성합니다. DB2 테이블, 서브테이블 또는 계층 구조에서 데이터를 익스포트한 경우, 인덱스가 작성됩니다. 이 옵션이 계층 구조에서 작동하고 DB2에서 데이터를 익스포트한 경우, 자료형 계층 구조도 작성됩니다. 이 옵션은 IXF 파일에 대해서만 사용할 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

주: MVS 호스트 데이터베이스에서 데이터를 익스포트하고 호스트 페이지 크기에서 계산된 길이가 254를 초과하는 LONGVAR 필드를 포함할 경우, 행이

너무 길기 때문에 **CREATE**에 실패할 수 있습니다. 제한사항 목록에 대해서는 『임포트된 테이블 재작성』을 참조하십시오. 이 경우, 테이블을 직접 작성한 후 **INSERT**를 사용하여 IMPORT를 호출하거나 LOAD 명령을 사용해야 합니다.

DEFAULT *schema-sqlid*

이 옵션은 **USING XDS** 매개변수가 지정된 경우에만 사용할 수 있습니다. **DEFAULT** 절을 통해 지정된 스키마는 임포트된 XML 문서의 XDS(XML Data Specifier)가 XML 스키마를 식별하는 SCH 속성을 포함하지 않을 때 유효성 확인에 사용할 스키마를 식별합니다.

DEFAULT 절은 **IGNORE** 및 **MAP** 절보다 우선순위를 갖습니다. XDS가 **DEFAULT** 절을 충족시키면 **IGNORE** 및 **MAP** 스펙은 무시됩니다.

FROM *filename*

HIERARCHY

계층 구조 데이터가 임포트되도록 지정합니다.

IGNORE *schema-sqlid*

이 옵션은 **USING XDS** 매개변수가 지정된 경우에만 사용할 수 있습니다. SCH 속성에 의해 식별될 경우 **IGNORE** 절은 무시할 하나 이상의 스키마 목록을 지정합니다. 임포트된 XML 문서에 대한 XDS(XML Data Specifier)에 SCH 속성이 존재하고 SCH 속성에 의해 식별된 스키마가 무시할 스키마 목록에 포함된 경우, 이 임포트된 XML 문서에 대해서는 스키마 유효성 확인이 발생하지 않습니다.

IGNORE 절에 스키마가 지정되어 있으면 이 스키마는 또한 **MAP** 절에 있는 스키마 쌍의 왼쪽에 존재할 수 없습니다.

IGNORE 절은 XDS에만 적용됩니다. **MAP** 절에 의해 맵핑된 스키마는 **IGNORE** 절에 의해 지정된 경우 계속 무시되지 않습니다.

IN *tablespace-name*

테이블이 작성될 테이블 스페이스를 식별합니다. 테이블 스페이스가 있어야 하며 이 테이블 스페이스는 REGULAR 테이블 스페이스이어야 합니다. 다른 테이블 스페이스를 지정하지 않으면 모든 테이블 부분이 이 테이블 스페이스에 저장됩니다. 이 절을 지정하지 않으면 테이블은 권한 부여 ID가 작성한 테이블 스페이스에 작성됩니다. 테이블 스페이스를 찾을 수 없으면 테이블은 디폴트 테이블 스페이스 USERSPACE1에 넣어집니다. USERSPACE1가 삭제된 경우 테이블 작성에 실패합니다.

INDEX IN *tablespace-name*

테이블의 인덱스가 작성될 테이블 스페이스를 식별합니다. 이 옵션은 **IN** 절에 지정된 1차 테이블 스페이스가 DMS 테이블 스페이스일 경우에만 허용됩니다.

지정된 테이블 스페이스가 있어야 하며 이 테이블 스페이스는 REGULAR 또는 LARGE DMS 테이블 스페이스이어야 합니다.

주: 인덱스를 포함할 테이블 스페이스는 테이블이 작성되었을 때만 지정할 수 있습니다.

insert-column
데이터가 삽입될 테이블 또는 뷰에 있는 컬럼 이름을 지정합니다.

INSERT
기존 테이블 데이터를 변경하지 않고 임포트된 데이터를 테이블에 추가합니다.

INSERT_UPDATE
임포트된 데이터 행을 목표 테이블에 추가하거나 기본 키가 일치하는 기존 행(목표 테이블의)을 갱신합니다.

INTO *table-name*
데이터가 임포트될 데이터베이스 테이블을 지정합니다. 이 테이블은 시스템 테이블, 작성된 임시 테이블, 선언된 임시 테이블 또는 요약 테이블이 될 수 없습니다.

완전한 테이블 이름 또는 규정되지 않은 테이블 이름을 사용해야 하는 이전 서버의 경우를 제외하고는 **INSERT**, **INSERT_UPDATE** 또는 **REPLACE**에 별명을 사용할 수 있습니다. 규정된 테이블 이름의 형식은 *schema.tablename* 입니다. *schema*는 테이블이 작성된 사용자 이름입니다.

LOBS FROM *lob-path*
LOB 데이터 파일의 이름은 기본 데이터 파일(ASC, DEL 또는 IXF)에 저장되는데 LOB 컬럼으로 로드될 컬럼에 저장됩니다. 지정할 수 있는 최대 경로 수는 999입니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

LONG IN *tablespace-name*
긴 컬럼(LONG VARCHAR, LONG VARGRAPHIC, LOB 데이터 유형 또는 이들 유형 중 하나를 소스 유형으로 갖는 구별 유형)의 값이 저장될 테이블 스페이스를 식별합니다. 이 옵션은 **IN** 절에 지정된 1차 테이블 스페이스가 DMS 테이블 스페이스일 경우에만 허용됩니다. 테이블 스페이스가 있어야 하며 LARGE DMS 테이블 스페이스이어야 합니다.

MAP *schema-sqlid*
이 옵션은 **USING XDS** 매개변수가 지정된 경우에만 사용할 수 있습니다. 임포트된 각 XML 문서에 대한 XDS(XML Data Specifier)의 SCH 속성이 지정하는 스키마 대신 사용할 대체 스키마를 지정하려면 이 **MAP** 절을 사용하십시오. **MAP** 절은 하나 이상의 스키마 쌍의 목록을 지정하며, 여기서 각 쌍은 한 스키마 대 다른 스키마의 맵핑을 나타냅니다. 쌍에서 첫 번째 스키마는

XDS에 있는 SCH 속성에 의해 참조되는 스키마를 나타냅니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

MAP 절에 있는 스키마 쌍의 왼쪽에 있는 스키마는 **IGNORE** 절에 지정될 수 없습니다.

스키마 쌍 맵핑이 적용된 후, 최종 결과가 됩니다. 맵핑 조작은 전이되지 않으므로 선택된 스키마는 다른 스키마 상 맵핑에 계속 적용되지 않습니다.

스키마가 두 번 이상 맵핑될 수 없다는 것은 쌍의 왼쪽에 두 번 이상 나타날 수 없다는 것을 의미합니다.

METHOD

L 데이터를 임포트할 시작 및 끝 컬럼 번호를 지정합니다. 컬럼 번호는 데이터 행이 시작되는 바이트 오프셋입니다. 1부터 번호가 매겨집니다.

주: 이 방법은 ASC 파일에만 사용할 수 있으며 이 파일 유형에 대해 유일한 유효 옵션입니다.

N 임포트할 데이터 파일의 컬럼 이름을 지정합니다. 이 컬럼 이름의 대소문자는 시스템 카탈로그의 해당 이름의 대소문자와 일치해야 합니다. 널(NULL) 입력이 가능하지 않은 각 테이블 컬럼에는 **METHOD N** 목록에 해당 항목이 있어야 합니다. 예를 들면, 주어진 데이터 필드 F1, F2, F3, F4, F5 및 F6과 테이블 컬럼 C1 INT, C2 INT NOT NULL, C3 INT NOT NULL 및 C4 INT, method N (F2, F1, F4, F3)은 유효한 요청인 반면, method N (F2, F1)은 유효하지 않습니다.

주: 이 방법은 IXF 파일에 대해서만 사용할 수 있습니다.

P 임포트될 입력 데이터 필드의 필드 번호를 지정합니다.

주: 이 방법은 IXF 또는 DEL 파일에만 사용할 수 있으며 DEL 파일 유형에 대해 유일한 유효 옵션입니다.

MODIFIED BY *filetype-mod*

파일 유형 수정자 옵션을 지정합니다. 116 페이지의 『임포트 유틸리티의 파일 유형 수정자』를 참조하십시오.

NOTIMEOUT

임포트 유틸리티가 잠금 대기 중 시간종료되지 않도록 지정합니다. 이 옵션은 **locktimeout** 데이터베이스 구성 매개변수를 대체합니다. 다른 응용프로그램에는 영향을 주지 않습니다.

NULL INDICATORS *null-indicator-list*

이 옵션은 **METHOD L** 매개변수가 지정된 경우에만 사용할 수 있습니다. 즉, 입력 파일은 ASC 파일입니다. 널(NULL) 표시기 목록은 각 널(NULL) 표시

기 필드의 컬럼 번호를 지정하는 쉼표로 구분된 양의 정수 목록입니다. 컬럼 번호는 데이터 행이 시작되는 널(NULL) 표시기 필드의 바이트 오프셋입니다. **METHOD L** 매개변수에 정의된 각 데이터 필드에 대해 한 개의 항목이 널(NULL) 표시기 목록에 있어야 합니다. 컬럼 번호 0은 해당 데이터 필드에 데이터가 항상 들어 있음을 나타냅니다.

널(NULL) 표시기의 `Y` 값은 컬럼 데이터를 널(NULL)로 지정합니다. 널(NULL) 표시기 컬럼에서 `Y` *이외의* 모든 문자는 컬럼 데이터가 널(NULL)이 아니며 **METHOD L** 옵션이 지정하는 컬럼 데이터가 임포트됨을 지정합니다.

`nullindchar` 파일 유형 수정자와 함께 **MODIFIED BY** 옵션을 사용하여 널(NULL) 표시기 문자를 변경할 수 있습니다.

OF *filetype*

입력 파일에 있는 데이터의 형식을 지정합니다.

- ASC(컬럼 식별자가 없는 ASCII 형식)
- DEL(컬럼 식별자가 있는 ASCII 형식) - 여러 데이터베이스 관리 프로그램 및 파일 관리자 프로그램에서 사용됩니다.
- WSF(작업시트 형식) - 다음과 같은 프로그램에서 사용됩니다.
  - Lotus 1-2-3
  - Lotus Symphony
- IXF(Integration Exchange Format, PC 버전)는 DB2에 의해 독점 사용되는 2진 형식입니다.

중요사항: WSF 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

별칭으로 임포트할 때 WSF 파일 유형은 지원되지 않습니다.

REPLACE

데이터 오브젝트를 잘라내어 테이블에서 모든 기존 데이터를 삭제한 후 임포트된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다. 이 옵션은 테이블이 존재할 경우에만 사용할 수 있습니다. 계층 구조 간에 데이터를 이동할 때 이 옵션을 사용하면 개별적 서브테이블이 아닌 전체 계층 구조에 대한 데이터만 바꿀 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

이 옵션은 CREATE TABLE문의 NOT LOGGED INITIALLY(NLI)절이나 ALTER TABLE문의 ACTIVE NOT LOGGED INITIALLY절을 인정하지 않습니다.

NLI절이 호출된 CREATE TABLE 또는 ALTER TABLE문과 동일한 트랜잭션에서 **REPLACE** 옵션을 사용한 임포트를 수행할 경우, 임포트는 NLI절을 인정하지 않습니다. 모든 삽입은 로그됩니다.

일시적인 해결책 1

DELETE문을 사용하여 테이블의 내용을 삭제한 후 INSERT문을 사용하여 임포트를 호출하십시오.

일시적인 해결책 2

테이블을 삭제하고 재작성한 후 INSERT문을 사용하여 임포트를 호출하십시오.

이 제한사항은 DB2 Universal Database 버전 7 및 DB2 UDB 버전 8에 적용됩니다.

REPLACE_CREATE

주: REPLACE_CREATE 매개변수는 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항은 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE가 사용되지 않음』을 참조하십시오.

테이블이 존재할 경우, 데이터 오브젝트를 잘라내어 테이블에서 모든 기존 데이터를 삭제한 후 테이블 정의 또는 인덱스 정의를 변경하지 않고 임포트된 데이터를 삽입합니다.

테이블이 존재하지 않을 경우, 데이터베이스의 코드 페이지에 행 내용과 함께 테이블 및 인덱스 정의를 작성합니다. 제한사항 목록에 대해서는 *임포트된 테이블 재작성*을 참조하십시오.

이 옵션은 IXF 파일에 대해서만 사용할 수 있습니다. 계층 구조 간에 데이터를 이동하는 경우에 이 옵션을 사용하면 개별 부속 테이블이 아닌 전체 계층 구조에 대한 데이터만 바꿀 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

RESTARTCOUNT *n*

임포트 조작이 레코드 *n* + 1에서 시작되도록 지정합니다. 첫 번째 *n* 레코드는 건너뜁니다. 이 솔루션은 기능적으로 **SKIPCOUNT**와 동일합니다. **RESTARTCOUNT** 및 **SKIPCOUNT**는 상호 배타적입니다.

ROWCOUNT *n*

파일에 있는 *n*개의 실제 레코드가 임포트(삽입되거나 갱신됨)되도록 지정합니다. 사용자가 **SKIPCOUNT** 또는 **RESTARTCOUNT** 옵션에 의해 판별된 레코드로부터 *n*개의 행만을 파일에서 임포트할 수 있게 합니다. **SKIPCOUNT** 또는 **RESTARTCOUNT** 옵션이 지정되지 않은 경우 첫 번째 *n* 행만 임포트합니다. **SKIPCOUNT** *m* 또는 **RESTARTCOUNT** *m*이 지정된 경우 *m*+1에

서 *m*+*n*까지의 행을 임포트합니다. 복합 삽입을 사용할 때, **ROWCOUNT** *n*은 복합 계수 값의 첫 번째 정수 배수로 반올림됩니다.

SKIPCOUNT *n*

임포트 조작이 레코드 *n* + 1에서 시작되도록 지정합니다. 첫 번째 *n* 레코드는 건너뜁니다. 이 옵션은 기능적으로 **RESTARTCOUNT**와 동일합니다. **SKIPCOUNT** 및 **RESTARTCOUNT**는 상호 배타적입니다.

STARTING *sub-table-name*

*sub-table-name*부터 시작하고 디폴트 순서를 요청하며 계층 구조 전용 키워드입니다. PC/IXF 파일의 경우 디폴트 순서는 입력 파일에 저장된 순서입니다. PC/IXF 파일 형식에는 디폴트 순서만 사용할 수 있습니다.

sub-table-list

INSERT 또는 **INSERT_UPDATE** 옵션을 사용하는 유형이 지정된 테이블의 경우, 데이터가 임포트될 서브테이블을 표시하기 위해 서브테이블 이름 목록을 사용합니다.

traversal-order-list

INSERT, **INSERT_UPDATE** 또는 **REPLACE** 옵션을 사용하는 유형이 지정된 테이블의 경우, 계층 구조에서 임포트 중인 서브테이블의 트래버스 순서를 표시하기 위해 서브테이블 이름 목록을 사용합니다.

UNDER *sub-table-name*

하나 이상의 서브테이블을 작성하기 위한 상위 테이블을 지정합니다.

WARNINGCOUNT *n*

*n*번의 경고 후 임포트 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 임포트 파일 또는 목표 테이블이 올바르지 않게 지정될 경우 임포트 유틸리티는 임포트하려는 각 행에 대해 경고를 생성하므로 임포트를 실패하게 됩니다. *n*이 0이거나 이 옵션이 지정되지 않은 경우 임포트 조작은 발행된 경고 수에 관계없이 계속 수행됩니다.

XML FROM *xml-path*

XML 파일이 들어 있는 하나 이상의 경로를 지정합니다.

XMLPARSE

XML 문서가 구문 분석되는 방법을 지정합니다. 이 옵션을 지정하지 않을 경우, XML 문서에 대한 구문 분석 동작은 CURRENT XMLPARSE OPTION 특수 레지스터의 값으로 판별됩니다.

STRIP WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하도록 지정합니다.

PRESERVE WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하지 않도록 지정합니다.

XMLVALIDATE

XML 문서가 스키마에 대해 유효성이 확인되도록 지정합니다(해당되는 경우).

USING XDS

기본 데이터 파일의 XDS(XML Data Specifier)에 의해 식별된 XML 스키마에 대해 XML 문서의 유효성을 확인합니다. **USING XDS**와 함께 **XMLVALIDATE** 옵션을 호출한 경우, 유효성 확인을 수행하는 데 사용된 스키마는 디폴트로 XDS의 SCH 속성에 의해 판별됩니다. SCH 속성이 XDS에 존재하지 않을 경우, **DEFAULT** 절로 디폴트 스키마를 지정하지 않으면 스키마 유효성 확인이 발생하지 않습니다.

DEFAULT, **IGNORE** 및 **MAP** 절은 스키마 판별 동작을 수정하는 데 사용될 수 있습니다. 이들 세 개의 선택적 절은 XDS의 권장 스펙에 직접적으로 적용되며 서로에게는 적용되지 않습니다. 예를 들어, 한 스키마가 **DEFAULT** 절에서 지정되어 선택되었으면 이 스키마는 **IGNORE** 절에 지정되어도 무시되지 않습니다. 마찬가지로 한 스키마가 MAP 절에서 첫 번째 파트 쌍으로 지정되어 선택되면 다른 **MAP** 절 쌍의 두 번째 파트에 지정되어도 다시 맵핑되지 않습니다.

USING SCHEMA *schema-sqlid*

XML 문서가 지정된 SQL ID가 있는 XML 스키마에 대해 유효성이 확인됩니다. 이 경우 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

USING SCHEMALOCATION HINTS

XML 문서가 소스 XML 문서의 XML 스키마 위치 힌트에 의해 식별된 스키마에 대해 유효성이 확인됩니다. XML 문서에서 schemaLocation 속성을 찾을 수 없으면 유효성 확인이 발생하지 않습니다. **USING SCHEMALOCATION HINTS** 절을 지정하면 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

아래의 **XMLVALIDATE** 옵션 예를 참조하십시오.

사용 시 참고사항

임포트 조작을 시작하기 전에 반드시 모든 테이블 조작을 완료하고 모든 잠금을 릴리스하십시오. 이는 WITH HOLD로 열려진 모든 커서를 닫고 COMMIT를 발행하거나 ROLLBACK을 발행하여 수행할 수 있습니다.

임포트 유틸리티는 SQL INSERT문을 사용하여 목표 테이블에 행을 추가합니다. 이 유틸리티는 입력 파일에 있는 각 데이터 행마다 하나의 INSERT문을 발행합니다. INSERT 문을 실패하면 다음 두 가지 조치 중 하나가 발생합니다.

- 후속 INSERT문이 성공할 수 있을 것 같으면 메시지 파일에 경고 메시지를 작성하고 처리를 계속합니다.
- 후속 INSERT문이 실패할 것 같으며 데이터베이스 손상이 생길 가능성이 있으면 메시지 파일에 오류 메시지를 작성하고 처리를 정지합니다.

이 유틸리티는 **REPLACE** 또는 **REPLACE_CREATE** 조작 중 이전 행이 삭제된 후 자동 COMMIT를 수행합니다. 따라서 테이블 오브젝트가 잘려진 후 응용프로그램이 데이터베이스 관리 프로그램를 인터럽트하거나 시스템이 실패할 경우 이전 데이터가 모두 유실됩니다. 이들 옵션을 사용하기 전에 이전 데이터가 더 이상 필요하지 않은지 확인하십시오.

CREATE, **REPLACE** 또는 **REPLACE_CREATE** 조작 중 로그가 가득차게 되면 이 유틸리티는 삽입된 레코드에 대해 자동 COMMIT를 수행합니다. 자동 COMMIT 후 응용프로그램이 데이터베이스 관리 프로그램를 인터럽트하거나 시스템이 실패할 경우 부분 데이터가 있는 테이블이 데이터베이스에 남아 있습니다. **REPLACE** 또는 **REPLACE_CREATE** 옵션을 사용하여 전체 임포트 조작을 재실행하거나 성공적으로 임포트된 행 수로 설정된 **RESTARTCOUNT** 매개변수를 사용하여 **INSERT**를 사용하십시오.

디폴트로 **INSERT** 또는 **INSERT_UPDATE** 옵션에 대해서는 자동 COMMIT가 수행되지 않습니다. 단, **COMMITCOUNT** 매개변수가 0이 아닌 경우에는 수행됩니다. 자동 COMMIT가 수행되지 않으면 가득찬 로그는 ROLLBACK됩니다.

다음 조건 중 하나라도 해당될 경우 오프라인 임포트는 자동 COMMIT를 수행하지 않습니다.

- 목표가 테이블이 아니고 뷰인 경우
- 복합 텍스트 삽입이 사용된 경우
- 버퍼링된 삽입이 사용된 경우

디폴트로 온라인 임포트는 자동 COMMIT를 수행하여 사용 중인 로그 스페이스와 잠금 목록을 모두 해제합니다. 자동 COMMIT는 **COMMITCOUNT** 값이 0으로 지정된 경우에만 수행되지 않습니다.

임포트 유틸리티가 COMMIT를 수행할 때마다 메시지 파일에 두 개의 메시지가 작성되는데, 하나는 커미트되는 레코드 수를 나타내고 다른 하나는 COMMIT 완료 후 작성됩니다. 실패 후 임포트 조작을 재시작할 때 마지막 완료된 COMMIT에서 판별된 대로 생략할 레코드 수를 지정하십시오.

임포트 유틸리티는 사소한 비호환성 문제점이 있는 입력 데이터는 승인합니다(예를 들어, 문자 데이터가 채우기 또는 절단을 사용하여 임포트되고 숫자 데이터가 다른 숫자 데이터 유형으로 임포트될 수 있습니다). 그러나 주요한 비호환성 문제점이 있는 데이터는 승인되지 않습니다.

오브젝트 테이블이 자체 테이블 외에 하위 테이블이 있는 경우 이 오브젝트 테이블을 **REPLACE** 또는 **REPLACE_CREATE** 할 수 없으며, 기본 테이블에 하위 테이블(자체 테이블을 포함하여)이 있는 경우 해당 오브젝트 뷰를 REPLACE 또는 REPLACE_CREATE 할 수 없습니다. 이러한 테이블 또는 뷰를 바꾸려면 다음을 수행하십시오.

1. 해당 테이블이 상위 테이블인 모든 외부 키를 삭제하십시오.
2. 임포트 유틸리티를 실행하십시오.
3. 테이블을 변경하여 외부 키를 재작성하십시오.

외부 키를 재작성하는 동안 오류가 발생하면 데이터를 수정하여 참조 무결성을 유지보수하십시오.

PC/IXF 파일에서 테이블을 재작성할 때 참조 제한조건 및 외부 키 정의는 보존되지 않습니다. (기본 키 정의는 데이터가 이전에 SELECT *를 사용하여 익스포트된 경우에 *보존됩니다*.)

리모트 데이터베이스로 임포트하려면 입력 데이터 파일의 사본, 출력 메시지 파일 및 잠재적인 데이터베이스 크기 확장에 필요한 디스크 스페이스가 충분해야 합니다.

리모트 데이터베이스에 대해 임포트 조작이 수행되고 출력 메시지 파일이 매우 긴 경우(60KB 초과), 클라이언트의 사용자에게 리턴되는 메시지 파일은 임포트 조작 중간부터 메시지가 누락될 수 있습니다. 메시지 정보의 처음 30KB와 메시지 정보의 마지막 30KB는 항상 보유됩니다.

PC/IXF 파일이 디스켓 대신 하드 드라이브에 있을 경우 PC/IXF 파일을 리모트 데이터베이스로 임포트하면 훨씬 더 빨리 수행됩니다.

ASC, **DEL** 또는 **WSF** 파일 형식의 데이터를 임포트하려면 먼저 데이터베이스 테이블 또는 계층 구조가 존재해야 합니다. 그러나 테이블이 이미 존재하지 않을 경우 IMPORT **CREATE** 또는 IMPORT **REPLACE_CREATE**가 PC/IXF 파일에서 데이터를 임포트할 때 테이블을 작성합니다. 유형이 지정된 테이블에서 IMPORT **CREATE**는 자료형 계층 구조와 테이블 계층 구조를 작성할 수 있습니다.

데이터베이스 간에 데이터를 이동하려면(계층 데이터를 포함하여) PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다. 동일한 클라이언트에서 소스 또는 목표 데이터베이스에 모두 액세스할 수 있는 경우 파일 복사 단계는 필요하지 않습니다.

ASC 및 DEL 파일의 데이터는 임포트를 수행하는 클라이언트 응용프로그램의 코드 페이지에 있는 것으로 간주됩니다. 서로 다른 코드 페이지의 데이터를 임포트할 때는 서로 다른 코드 페이지에 사용할 수 있는 PC/IXF 파일을 사용할 것을 권장합니다. PC/IXF

파일과 임포트 유틸리티가 동일한 코드 페이지에 있을 경우 일반 응용프로그램에 대해서와 같은 처리가 발생합니다. 두 코드 페이지가 서로 다르고 **FORCEIN** 옵션이 지정된 경우, 임포트 유틸리티는 PC/IXF 파일의 데이터가 임포트를 수행 중인 응용프로그램과 동일한 코드 페이지를 가지고 있다고 간주합니다. 이는 두 코드 페이지에 대한 변환 테이블이 있는 경우에도 마찬가지입니다. 두 코드 페이지가 서로 다르고, **FORCEIN** 옵션이 지정되지 않았으며 변환 테이블이 있는 경우, PC/IXF 파일의 모든 데이터는 파일 코드 페이지에서 응용프로그램 코드 페이지로 변환됩니다. 두 코드 페이지가 서로 다르고 **FORCEIN** 옵션이 지정되지 않았으며 변환 테이블이 없는 경우 임포트 조작은 실패합니다. 이는 AIX 운영 체제의 DB2 클라이언트에 있는 PC/IXF 파일에만 적용됩니다.

1012 컬럼의 한계에 가까운 8KB 페이지에 있는 테이블 오브젝트의 경우, PC/IXF 데이터 파일을 임포트하면 DB2가 오류를 리턴할 수 있는데, 이는 SQL문의 최대 크기를 초과했기 때문입니다. 이러한 상황은 컬럼 유형이 CHAR, VARCHAR 또는 CLOB인 경우에만 발생할 수 있습니다. **DEL** 또는 **ASC** 파일을 임포트하는 데는 이러한 제한사항이 적용되지 않습니다. PC/IXF 파일을 사용하여 새 테이블을 작성할 경우 다른 방법은 db2look을 사용하여 테이블을 작성한 DDL문을 덤프한 후 CLP를 통해 이 명령문을 발행하는 것입니다.

DB2 Connect는 OS/390용 DB2, VM 및 VSE용 DB2, OS/400용 DB2와 같은 DRDA 서버로 데이터를 임포트하는 데 사용할 수 있습니다. PC/IXF 임포트(**INSERT** 옵션)만 지원됩니다. **RESTARTCOUNT** 매개변수도 지원되지만 **COMMITCOUNT** 매개변수는 지원되지 않습니다.

유형이 지정된 테이블에서 **CREATE** 옵션을 사용할 때 PC/IXF 파일에 정의된 모든 서브테이블을 작성하십시오. 서브테이블 정의는 변경할 수 없습니다. 유형이 지정된 테이블에서 **CREATE** 이외의 옵션을 사용할 때, 트래버스 순서 목록을 통해 해당 옵션이 트래버스 순서를 지정하므로 트래버스 순서 목록은 익스포트 조작 시 사용된 옵션과 일치해야 합니다. PC/IXF 파일 형식의 경우, 옵션은 목표 서브테이블 이름만 지정해야 하며 파일에 저장된 트래버스 순서를 사용해야 합니다.

임포트 유틸리티를 사용하여 이전에 PC/IXF 파일로 익스포트된 테이블을 복구할 수 있습니다. 이 테이블은 익스포트될 때의 상태로 리턴합니다.

데이터는 시스템 테이블, 작성된 임시 테이블, 선언된 임시 테이블 또는 요약 테이블로 임포트될 수 없습니다.

임포트 유틸리티를 통해 뷰를 작성할 수 없습니다.

개별적 파트가 Windows 시스템에서 AIX 시스템으로 복사되는 다중 파트 PC/IXF 파일 임포트가 지원됩니다. 첫 번째 파일의 이름만 IMPORT 명령에 지정되어야 합니다.

예를 들어, IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1. 파일 data.002, etc가 data.ixf와 동일한 디렉토리에서 사용 가능해야 합니다.

Windows 운영 체제에서:

- 논리적으로 분할된 PC/IXF 파일 임포트는 지원되지 않습니다.
- 잘못된 형식의 PC/IXF 또는 WSF 파일 임포트는 지원되지 않습니다.

내부 형식으로 된 보안 레이블에는 줄 바꾸기 문자가 포함될 수 있습니다. DEL 파일 형식을 사용하여 파일을 임포트할 경우 이들 줄 바꾸기 문자는 분리문자로 오인될 수 있습니다. 이러한 문제점이 발생하면 IMPORT 명령에 delprioritychar 파일 유형 수정자를 지정하여 분리문자에 대해 이전 디폴트 우선순위를 사용하십시오.

페더레이티드 고려사항

IMPORT 명령 및 **INSERT**, **UPDATE** 또는 **INSERT_UPDATE** 명령 매개변수를 사용할 때 참여 중인 별칭에 대해 CONTROL 특권을 가지고 있는지 확인해야 합니다. 임포트 조작을 수행할 때 사용할 별칭이 이미 존재하는지 확인해야 합니다. 또한 IMPORT 명령 매개변수 섹션에서 표시된 바와 같이 여러 가지 제한사항을 유념해야 합니다.

ODBC와 같은 일부 데이터 소스는 별칭으로의 임포트를 지원하지 않습니다.

임포트 유틸리티의 파일 유형 수정자

표 20. 임포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식

수정자	설명
compound=*x*	*x*는 1 - 100의 숫자입니다. nonatomic 복합 SQL을 사용하여 데이터를 삽입하며, *x* 명령문이 매번 시도됩니다. 수정자가 지정되고 트랜잭션 로그가 충분히 크지 않으면, 임포트 조작이 실패합니다. **COMMITCOUNT**에서 지정한 행 수나, **COMMITCOUNT**가 지정되지 않은 경우 데이터 파일의 행 수를 수용할 정도로 트랜잭션 로그가 커야 합니다. 그러므로 트랜잭션 로그 오버플로우를 피하기 위해 **COMMITCOUNT** 옵션을 지정하도록 권장합니다. 이 수정자는 **INSERT_UPDATE** 모드, 계층 테이블 및 다음 수정자와 호환되지 않습니다. usedefaults, identitymissing, identityignore, generatedmissing 및 generatedignore.
generatedignore	이 수정자는 생성된 모든 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 임포트 유틸리티에 알립니다. 이로 인해 생성된 컬럼의 모든 값이 유틸리티에 의해 생성됩니다. 이 수정자는 generatedmissing 수정자와 함께 사용될 수 없습니다.
generatedmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 생성된 컬럼의 데이터가 포함되지 않는 것으로(NULL 값도 비포함) 가정하므로 각 행의 값을 생성합니다. 이 수정자는 generatedignore 수정자와 함께 사용될 수 없습니다.

표 20. 임포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
identityignore	이 수정자는 ID 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 임포트 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ID 값이 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 `identitymissing` 수정자와 함께 사용될 수 없습니다.
identitymissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 ID 컬럼의 데이터가 포함되지 않는 것으로(NULL 값도 비포함) 가정하므로 각 행의 값을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 `identityignore` 수정자와 함께 사용될 수 없습니다.
lobsinfile	*lob-path*는 LOB 데이터를 포함하는 파일에 대한 경로를 지정합니다. 각 경로는 LLS(Lob Location Specifier)에 의해 포인트되는 최소한 하나의 LOB를 포함하는 최소한 하나의 파일을 데이터 파일에 포함합니다. LLS는 LOB 파일 경로에 저장된 파일에서 LOB 위치의 문자열 표시입니다. LLS 형식은 *filename.ext.nnn.mmm/*이며, 여기서 *filename.ext*는 LOB를 포함하는 파일의 이름이며, *nnn*은 파일 내에서 LOB의 오프셋을 바이트로 나타낸 것이며, `mmm`은 LOB의 길이를 바이트로 나타낸 것입니다. 예를 들어, 문자열 `db2exp.001.123.456/`가 데이터 파일에 저장되는 경우, LOB는 `db2exp.001` 파일에서 오프셋 123에 위치하며 456바이트 길이입니다. **LOBS FROM** 절은『lobsinfile』수정자가 사용될 때 LOB 파일이 위치하는 곳을 지정합니다. **LOBS FROM** 절은 내재적으로 LOBSINFILE 동작을 활성화합니다. **LOBS FROM** 절은 데이터 임포트 중 LOB 파일을 검색하기 위해 경로 목록을 IMPORT 유틸리티로 전달합니다. 널(NULL) LOB를 표시하려면, *크기*를 -1로 입력하십시오. *크기*가 0으로 지정되면, 길이가 0인 LOB로 처리됩니다. 길이가 -1인 널(NULL) LOBS의 경우, 오프셋 및 파일 이름은 무시됩니다. 예를 들어, 널(NULL) LOB의 LLS는 db2exp.001.7.-1/입니다.
no_type_id	단일 서브테이블로 임포트할 때에만 유효합니다. 일반 설치는 일반 테이블에서 데이터를 익스포트한 후 임포트 조작을 호출하여(이 수정자 사용) 데이터를 단일 서브테이블로 변환합니다.
nodefaults	목표 테이블 컬럼의 소스 컬럼이 명시적으로 지정되지 않고, 테이블 컬럼이 널(NULL) 입력 가능하지 않으면, 디폴트값이 로드되지 않습니다. 목표 테이블 컬럼 중 하나의 소스 컬럼이 명시적으로 지정되지 않은 경우, 이 옵션이 없으면 다음 중 하나가 발생합니다. • 컬럼의 디폴트값이 지정될 수 있는 경우, 디폴트값이 로드됩니다. • 컬럼이 널(NULL) 입력 가능하고 해당 컬럼의 디폴트값을 지정할 수 없는 경우, NULL이 로드됩니다. • 컬럼이 널(NULL) 입력 가능하고 디폴트값을 지정할 수 없는 경우, 오류가 리턴되며 유틸리티가 처리를 중지합니다.
norowwarnings	거부된 행에 대한 모든 경고를 제외시킵니다.
rowchangetimestampignore	이 수정자는 행 변경 시간소인 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 임포트 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ROW CHANGE TIMESTAMP가 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 `rowchangetimestampmissing` 수정자와 함께 사용될 수 없습니다.
rowchangetimestampmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 행 변경 시간소인 컬럼의 데이터를 포함하지 않음(NULL 값도 비포함)을 가정하므로 각 행의 값을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 `rowchangetimestampignore` 수정자와 함께 사용될 수 없습니다.

표 20. 임포트 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
seclabelchar	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 보안 레이블 값의 문자열 형식임을 표시합니다. IMPORT는 각 보안 레이블을 로드된 대로의 내부 형식으로 변환합니다. 문자열이 적절한 형식으로 되어 있지 않은 경우 행은 로드되지 않으며 경고(SQLSTATE 01H53)가 리턴됩니다. 문자열이 테이블을 보호하는 보안 규정 파트인 유효한 보안 레이블을 나타내지 않는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3243W)가 리턴됩니다. seclabelname 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 임포트에 실패하며 오류(SQLCODE SQL3525N)가 리턴됩니다.
seclabelname	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 이름으로 표시됨을 나타냅니다. IMPORT는 이름이 있는 경우 적절한 보안 레이블로 변환합니다. 테이블을 보호하는 보안 규정에 대해 표시된 이름이 있는 보안 레이블이 없는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3244W)가 리턴됩니다. seclabelchar 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 임포트에 실패하며 오류(SQLCODE SQL3525N) 가 리턴됩니다. **주:** 파일 유형이 ASC인 경우, 보안 레이블의 이름 다음에 오는 모든 스페이스는 이름의 일부로 해석됩니다. 이를 피하려면 striptblanks 파일 유형 수정자를 사용하여 스페이스가 제거되었는지 확인하십시오.
usedefaults	목표 테이블 컬럼의 소스 컬럼이 지정되었지만 하나 이상의 행 인스턴스에 대한 데이터를 포함하지 않은 경우, 디폴트값이 로드됩니다. 누락된 데이터의 예: • DEL 파일: 임의 수의 스페이스(″, ,″)로 구분되는 2개의 인접 컬럼 분리문자나 2개의 인접 컬럼 분리문자(″,,″)가 컬럼 값으로 지정됩니다. • DEL/ASC/WSF 파일: 행의 컬럼이 충분하지 않거나 행의 길이가 원래 스펙만큼 충분하지 않습니다. **주:** ASC 파일의 경우, NULL 컬럼 값은 명시적으로 누락된 것으로 간주되지 않으며 디폴트가 NULL 컬럼 값을 대신하지 않습니다. 숫자, 날짜, 시간 및 /시간소인 컬럼의 모든 공백 문자로 NULL 컬럼 값을 표시하거나, 컬럼이 NULL임을 표시하기 위해 모든 유형의 컬럼에 널(NULL) 표시기를 사용함으로써 NULL 컬럼 값을 표시합니다. 이 옵션이 없는 경우, 소스 컬럼이 행 인스턴스의 데이터를 포함하지 않으면 다음 중 하나가 발생합니다. • DEL/ASC/WSF 파일: 컬럼에 널(NULL) 입력 가능한 경우 NULL이 로드됩니다. 컬럼이 널(NULL) 입력 가능하지 않으면 유틸리티가 행을 거부합니다.

표 21. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL)

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 입력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 임포트 조작 중 이 코드 페이지로부터 응용프로그램 코드 페이지로 문자 데이터를 변환합니다. 다음 규칙이 적용됩니다. • 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. • nullindchar은 x20 - x7F 코드 포인트의 표준 ASCII 세트에 포함된 기호를 지정해야 합니다. 이것은 ASCII 기호 및 코드 포인트를 나타냅니다. **주:** 1. codepage 수정자는 lobsinfile 수정자와 함께 사용될 수 없습니다. 2. 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 코드 페이지를 변환할 때 데이터 확장이 발생하면, 데이터가 절단되고 데이터 유실이 발생할 수 있습니다.
dateformat="*x*"	*x*는 소스 파일에서 날짜의 형식입니다.[2] 유효한 날짜 요소는 다음과 같습니다. YYYY - 연도(0000 - 9999 사이의 4자리 숫자) M - 월(1 - 12 사이의 1 또는 2자리 숫자) MM - 월(1 - 12 사이의 2자리 숫자, M과 상호 배타적) D - 일(1 - 31 사이의 1 또는 2자리 숫자) DD - 일(1 - 31 범위의 2자리 숫자, D와 상호 배타적) DDD - 년의 일(001 - 366 범위의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적) 지정되지 않은 각 요소에 대해 디폴트값 1이 지정됩니다. 날짜 형식의 예: "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
implieddecimal	내포된 소수점의 위치는 컬럼 정의로 판별되며, 값의 끝으로 가정되지 않습니다. 예를 들어, 12345 값은 12345.00이 *아닌* 123.45로 DECIMAL(8,2) 컬럼에 로드됩니다.

표 21. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
timeformat="*x*"	*x*는 소스 파일에서 시간의 형식입니다.[2] 유효한 시간 요소는 다음과 같습니다. H - 시간(12시간 시스템의 경우 0 - 12 범위의, 1 또는 2자리 숫자 24시간 시스템의 경우 0 - 24) HH - 시간(12시간 시스템의 경우 0 - 12 범위의 2자리 숫자, 24시간 시스템의 경우 0 - 24. H와 상호 배타적) M - 분(0 - 59 사이의 1 또는 2자리 숫자) MM - 분(0 - 59 사이의 2자리 숫자, M과 상호 배타적) S - 초(0 - 59 사이의 1 또는 2자리 숫자) SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적) SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적) TT - 오전/오후 지시(AM 또는 PM) 지정되지 않은 각 요소에 대해 디폴트값 0이 지정됩니다. 시간 형식의 예: "HH:MM:SS" "HH.MM TT" "SSSSS"

표 21. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명

timestampformat="*x*"	*x*는 소스 파일에서 시간소인의 형식입니다.[2] 유효한 시간소인 요소는 다음과 같습니다.
	
	YYYY - 연도(0000 - 9999 사이의 4자리 숫자)
	M - 월(1 - 12 사이의 1 또는 2자리 숫자)
	MM - 월(01 - 12 사이의 2자리 숫자, M 및 MMM은 상호 배타적)
	MMM - 월(월 이름으로 사용되는 3자리 문자로 대소문자를 구분하지 않으며, M 및 MM은 상호 배타적)
	D - 일(1 - 31 사이의 1 또는 2자리 숫자)
	DD - 일(1 - 31 사이의 2자리 숫자, D와는 상호 배타적)
	DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적)
	H - 시간(12시간 시스템의 경우 0 - 12 범위의 1 또는 2자리 숫자, 24시간 시스템의 경우 0 - 24)
	HH - 시간(12시간 시스템의 경우 0 - 12 범위의 2자리 숫자, 24시간 시스템의 경우 0 - 24, H와는 상호 배타적)
	M - 분(0 - 59 사이의 1 또는 2자리 숫자)
	MM - 분(0 - 59 사이의 2자리 숫자, M, 분과 상호 배타적)
	S - 초(0 - 59 사이의 1 또는 2자리 숫자)
	SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적)
	SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적)
	U(1 - 12회) - 초의 소수 부분(U 어커런스 수는 각 자리가 0 - 9 사이의 자리 수를 나타냄)
	TT - 오전/오후 지시(AM 또는 PM)
	
	디폴트값 1이 미지정된 YYYY, M, MM, D, DD 또는 DDD 요소에 지정됩니다. 디폴트값 'Jan'이 미지정된 MMM 요소에 지정됩니다. 미지정된 다른 모든 요소에 디폴트값 0이 지정됩니다. 다음은 시간소인 형식의 예입니다.
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 요소의 올바른 값은 다음을 포함합니다. 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' 및 'dec'. 이들 값은 대소문자가 구분되지 않습니다.
	
	다음 예는 사용자 정의 날짜 및 시간 형식을 포함하는 데이터를 schedule이라는 테이블로 임포트하는 방법을 설명합니다.
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

표 21. 임포트 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
usegraphiccodepage	usegraphiccodepage가 제공되면, 그래픽 또는 2바이트 문자 대형 오브젝트(DBCLOB) 데이터 필드로 임포트되는 데이터는 그래픽 코드 페이지에 있음이 가정됩니다. 나머지 데이터는 문자 코드 페이지에 있다고 가정합니다. 그래픽 코드 페이지는 문자 코드 페이지와 연관됩니다. IMPORT는 codepage 수정자가 지정된 경우 이를 통해 문자 코드 페이지를 판별하고 codepage 수정자가 지정되지 않은 경우 응용프로그램의 코드 페이지를 통해 문자 코드 페이지를 판별합니다. 복구 중인 테이블에 그래픽 데이터가 있는 경우에만 삭제(drop) 테이블 복구로 생성되는 구분된 데이터 파일과 결합하여 이 수정자를 사용해야 합니다. **제한사항** 이들 파일이 오직 하나의 코드 페이지에 인코드된 데이터를 포함하면, usegraphiccodepage 수정자는 EXPORT 유틸리티로 작성된 DEL 파일과 함께 지정되지 않아야 합니다. usegraphiccodepage 수정자는 파일의 2바이트 문자 대형 오브젝트(DBCLOB)에서 무시됩니다.
xmlchar	XML 문서가 문자 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 지정된 문자 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 문자 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 문자 코드 페이지는 codepage 파일 유형 수정자에서 지정한 값이거나, 지정되지 않은 경우 응용프로그램 코드 페이지입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다.
xmlgraphic	XML 문서가 지정된 그래픽 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 특정 그래픽 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 그래픽 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 그래픽 코드 페이지는 codepage 파일 유형 수정자에서 지정한 값의 그래픽 구성요소이거나, 지정되지 않은 경우 응용프로그램 코드 페이지의 그래픽 구성요소입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다. **주:** xmlgraphic 수정자가 IMPORT 명령으로 지정된 경우, 임포트되는 XML 문서는 UTF-16 코드 페이지로 인코드되어야 합니다. 그렇지 않으면, XML 문서는 구문 분석 오류로 거부되거나 데이터 손상이 있는 테이블로 임포트될 수 있습니다.

표 22. 임포트 유틸리티의 유효한 파일 유형 수정자: ASC(컬럼 식별자가 없는 ASCII) 파일 형식

수정자	설명
nochecklengths	nochecklengths가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 임포트하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 임포트될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.

표 22. 임포트 유틸리티의 유효한 파일 유형 수정자: ASC(컬럼 식별자가 없는 ASCII) 파일 형식 (계속)

수정자	설명
nullindchar=*x*	*x*는 단일 문자입니다. 널(NULL) 값을 나타내는 문자를 *x*로 변경합니다. *x*의 디폴트값은 Y입니다.[3] 문자가 영문자인 경우를 제외하고 EBCDIC 데이터 파일의 경우 이 수정자의 대소문자를 구분합니다. 예를 들어, 널(NULL) 표시기 문자가 N 문자가 되도록 지정되는 경우, n은 널(NULL) 표시기로 인식됩니다.
reclen=*x*	*x*는 최대값이 32,767인 정수입니다. 각 행에 대해 *x* 문자가 읽혀지며 행의 끝을 표시하기 위한 줄 바꾸기 문자는 사용되지 않습니다.

striptblanks	데이터를 변수 길이 필드로 로드할 때 뒤 공백을 절단합니다. 이 옵션이 지정되지 않으면, 공백이 보존됩니다.
	
	다음 예에서, striptblanks를 사용하면 임포트 유틸리티가 뒤 공백을 절단합니다.
	`db2 import from myfile.asc of asc`
	`modified by striptblanks`
	`method l (1 10, 12 15) messages msgs.txt`
	`insert into staff`
	
	이 옵션은 striptnulls와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다.이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 t 옵션을 교체합니다.

striptnulls	데이터를 변수 길이 필드로 로드할 때 뒤 NULL 값(0x00 문자)을 절단합니다. 이 옵션이 지정되지 않으면, NULL 값이 보존됩니다. 이 옵션은 striptblanks와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다. 이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 padwithzero 옵션을 교체합니다.

표 23. 임포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식

수정자	설명

chardel*x*	*x*는 단일 문자열 분리문자입니다. 디폴트값은 큰따옴표(")입니다. 큰따옴표 대신 지정된 문자를 사용하여 문자열을 묶습니다.[3,4] 명시적으로 큰따옴표를 문자열 분리문자로 지정하려는 경우 다음과 같이 지정해야 합니다.
	`modified by chardel""`
	
	작은따옴표(')를 문자열 분리문자로 지정할 수도 있습니다. 다음 예에서, chardel''는 임포트 유틸리티가 작은따옴표를 문자열 분리문자로 해석하게 합니다.
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x*는 단일 문자 컬럼 분리문자입니다. 디폴트값은 쉼표(,)입니다. 지정된 문자가 쉼표 대신 사용되어 컬럼 끝을 나타냅니다.[3,4]
	
	다음 예에서, coldel;은 임포트 유틸리티가 세미콜론을 컬럼 분리문자로 해석하게 합니다.
	`db2 import from myfile.del of del`
	`modified by coldel;`
	`messages msgs.txt insert into staff`

decplusblank	플러스 부호 문자. 플러스 부호(+) 대신 공백을 양의 10진수 값 앞에 붙입니다. 디폴트 조치는 플러스 부호를 양의 10진수 값 앞에 붙이는 것입니다.

표 23. 임포트 유틸리티의 유효한 파일 유형 수정자: DEL(컬럼 식별자가 있는 ASCII) 파일 형식 (계속)

수정자	설명

decpt*x*	*x*는 소수점 문자로 마침표를 대신하는 단일 문자입니다. 디폴트값은 마침표(.)입니다. 마침표 대신 지정된 문자가 소수점 문자로 사용됩니다.[34]
	
	다음 예에서, `decpt;`는 임포트 유틸리티가 세미콜론(;)을 소수점으로 해석하게 합니다.
	
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`decpt; messages msgs.txt insert into staff"`

delprioritychar	분리문자의 현재 디폴트 우선순위: 레코드 구분 문자, 문자 분리문자, 컬럼 분리문자. 이 수정자는 분리문자 우선순위를 문자 분리문자, 레코드 구분 문자, 컬럼 분리문자로 되돌림으로써 이전 우선순위에 따른 기존 응용프로그램을 보호합니다. 구문:
	
	`db2 import ... modified by delprioritychar ...`
	
	예를 들면, 다음과 같은 DEL 데이터 파일이 있습니다.
	
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	
	`delprioritychar` 수정자를 지정하고, 이 데이터 파일에는 두 개의 행만이 있습니다. 두 번째 <행 분리문자>는 두 번째 행의 첫 번째 데이터 컬럼의 파트로 해석되지만, 첫 번째 및 세 번째 <행 분리문자>는 실제 레코드 구분 문자로 해석됩니다. 이 수정자가 지정되지 않은 경우, 이 데이터 파일에는 세 개의 행이 있으며, 각 행은 <행 분리문자>로 구분됩니다.

keepblanks	유형 CHAR, VARCHAR, LONG VARCHAR 또는 CLOB의 각 필드에 앞뒤 공백을 둡니다. 이 옵션이 없으면, 문자 분리문자 내에 없는 앞 공백과 뒤 공백은 모두 제거되며 공백 필드의 테이블로 NULL이 삽입됩니다.
nochardel	임포트 유틸리티는 컬럼 분리문자 간의 모든 바이트를 컬럼 데이터의 파트로 가정합니다. 문자 분리문자는 컬럼 데이터의 파트로 구문 분석됩니다. DB2를 사용하여 데이터를 익스포트한 경우(익스포트 시 nochardel을 지정한 경우를 제외하고) 이 옵션을 지정하지 말아야 합니다. 이 옵션은 문자 분리문자가 없는 벤더 데이터 파일을 지원하기 위해 제공됩니다. 부적절한 사용은 데이터 유실이나 손상을 초래할 수 있습니다. 이 옵션은 `chardelx`, `delprioritychar` 또는 `nodoubledel`과 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다.
nodoubledel	2바이트 분리문자를 인식하지 않습니다.

표 24. 임포트 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식

수정자	설명
forcein	코드 페이지 불일치와 관계없이 데이터를 승인하고 코드 페이지 간에 변환하지 않도록 유틸리티에 지시합니다. 데이터의 고정 길이 대상 필드가 충분히 큰지 검증하도록 고정 길이 대상 필드를 점검합니다. `nochecklengths`가 지정된 경우, 검사가 수행되지 않으며 각 행을 임포트하려고 시도합니다.
indexixf	기존 테이블에 현재 정의된 모든 인덱스를 삭제(drop)하고 PC/IXF 파일의 인덱스 정의에서 새로운 인덱스를 작성하도록 유틸리티에 지시합니다. 이 옵션은 목차가 교체될 때에만 사용될 수 있습니다. 뷰 또는 *insert-column*와 함께 사용될 수 없습니다.
indexschema=*schema*	색인 작성 중 인덱스 이름에 지정된 *schema*를 사용합니다. *schema*가 지정되지 않은 경우(그러나 키워드 `indexschema`가 지정됨), 연결 사용자 ID를 사용합니다. 해당 키워드가 지정되지 않은 경우, IXF 파일에서 스키마를 사용합니다.

표 24. 임포트 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식 (계속)

수정자	설명
nochecklengths	nochecklengths가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 임포트하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 임포트될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.
forcecreate	임포트 조작 중 SQL3311N 리턴 후에 제한된 정보나 가능한 누락 정보로 테이블이 작성되어야 함을 지정합니다.

표 25. codepage 및 usegraphiccodepage 사용 시 IMPORT 동작

codepage=N	usegraphiccodepage	IMPORT 동작
Absent	Absent	파일의 모든 데이터는 응용프로그램 코드 페이지에 있는 것으로 가정됩니다.
Present	Absent	파일의 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 임포트될 때 그래픽 데이터가 손상됩니다.
Absent	Present	파일의 문자 데이터는 응용프로그램 코드 페이지에 있는 것으로 가정됩니다. 그래픽 데이터는 응용프로그램 그래픽 데이터의 코드 페이지에 있는 것으로 가정됩니다. 응용프로그램 코드 페이지가 1바이트이면, 모든 데이터는 응용프로그램 코드 페이지에 있는 것으로 가정됩니다. **경고:** 응용프로그램 코드 페이지가 1바이트이면, 데이터베이스가 그래픽 컬럼을 포함해도 데이터베이스로 임포트될 때 그래픽 데이터가 손상됩니다.
Present	Present	문자 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. 그래픽 데이터는 N의 그래픽 코드 페이지에 있는 것으로 가정됩니다. N이 1바이트 또는 2바이트 코드 페이지인 경우, 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 임포트될 때 그래픽 데이터가 손상됩니다.

주:

1. **MODIFIED BY** 옵션과 함께 지원되지 않는 파일 유형을 사용하려고 시도하는 경우 임포트 유틸리티는 경고를 발행하지 않습니다. 이런 경우, 임포트 조작에 실패하며 오류 코드가 리턴됩니다.
2. 날짜 출력 문자열을 둘러싼 큰따옴표는 필수입니다. 필드 구분자는 a - z, A - Z 및 0 - 9를 포함할 수 없습니다. 필드 구분자는 DEL 파일 형식의 필드 분리문자나 문자 분리문자와 같지 않아야 합니다. 요소의 시작 및 종료 위치가 명확한 경우 필드 구분자가 선택적입니다. D, H, M 또는 S와 같은 요소가 사용되는 경우 (수정자에 따라) 항목의 변수 길이 때문에 모호함이 있을 수 있습니다.

시간소인 형식의 경우, 문자 M을 사용하는 month 및 minute 디스크립터 간의 모호함을 피하려면 주의해야 합니다. month 필드는 다른 날짜 필드와 인접해야 합니다. minute 필드는 다른 시간 필드와 인접해야 합니다. 다음은 모호한 시간소인 형식입니다.

```
"M"(month 또는 minute일 수 있음)
"M:M"(month 및 minute 구분 가능?)
"M:YYYY:M"(둘 다 month로 해석됨)
"S:M:YYYY"(둘 다 시간 값 및 날짜 값에 인접)
```

모호한 경우, 유틸리티는 오류 메시지를 발행하며, 조작에 실패합니다.

다음은 명확한 시간소인 형식입니다.

```
"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month....Minute)
"M:H:YYYY:M:D" (Minute....Month)
```

큰따옴표 및 백슬래시와 같은 일부 문자는 Escape 문자(예: ₩)가 앞에 와야 합니다.

3. chardel, coldel 또는 decpt 파일 유형에 제공되는 문자 값은 소스 데이터의 코드 페이지에 지정되어야 합니다.

   문자 코드 포인트(문자 기호 대신)는 구문 xJJ 또는 0xJJ를 사용하여 지정될 수 있으며, 여기서 JJ는 코드 포인트의 16진수를 나타냅니다. 예를 들어, 컬럼 분리문자로 # 문자를 지정하려면, 다음 중 하나를 사용하십시오.

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

4. *데이터 이동을 위한 분리문자 고려사항*은 분리문자를 겹쳐쓰기할 때 사용할 수 있는 문자에 적용하는 제한사항을 나열합니다.
5. 별칭으로 임포트할 때 다음 파일 유형 수정자는 허용되지 않습니다.
   - indexixf
   - indexschema
   - dldelfiletype
   - nodefaults
   - usedefaults
   - no_type_idfiletype
   - generatedignore
   - generatedmissing
   - identityignore
   - identitymissing

- lobsinfile

6. **WSF** 파일 형식은 XML 컬럼에서 지원되지 않습니다. 또한 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.
7. **CREATE** 모드는 XML 컬럼에서 지원되지 않습니다.
8. 모든 XML 데이터는 주 데이터 파일에서 분리된 XML 파일에 있어야 합니다. 주 데이터 파일의 각 XML 컬럼에 대해 XDS(XML Data Specifier)(또는 NULL 값)가 있어야 합니다.
9. XMLCHAR 또는 XMLGRAPHIC 파일 유형 수정자가 지정된 경우를 제외하고, XML 문서는 유니코드 형식으로 되어 있거나 인코딩 속성을 포함하는 선언 태그를 포함한다고 가정합니다.
10. 잘 양식화되지 않은 문서를 포함하는 행은 거부됩니다.
11. **XMLVALIDATE** 옵션이 지정된 경우, 일치하는 스키마에 대해 정상적으로 유효성을 확인하는 문서는 삽입될 때 스키마 정보로 주석을 표시합니다. 일치하는 스키마에 대해 유효성을 확인하는 데 실패하는 문서를 포함하는 행은 거부됩니다. 정상적으로 유효성을 확인하려면, 임포트를 호출하는 사용자가 보유한 특권이 최소한 다음 중 하나를 포함해야 합니다.
    - DBADM 권한
    - 유효성 확인에 사용되는 XML 스키마의 USAGE 특권
12. 내재적으로 숨겨진 행 변경 시간소인 컬럼을 포함하는 테이블로 임포트할 때, 내재적으로 숨겨진 컬럼의 등록 정보는 무시됩니다. 그러므로, 컬럼의 데이터가 임포트되는 데이터에 표시되지 않으며 명시적 컬럼 목록이 표시되지 않으면 rowchangetimestampmissing 파일 유형 수정자가 임포트 명령에 *지정되어야* 합니다.

db2Import - 테이블, 계층 구조, 별칭 또는 뷰로 데이터 임포트

지원되는 파일 형식의 외부 파일 데이터를 테이블, 계층 구조, 뷰 또는 별칭에 삽입합니다. 로드 유틸리티가 이 기능보다 빠릅니다. 그러나 로드 유틸리티는 계층 구조 레벨에서 데이터 로드 또는 별칭으로 로드는 지원하지 않습니다.

권한 부여

- INSERT 옵션을 사용하여 IMPORT하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 각 파티셔닝 테이블, 뷰 또는 별칭의 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT 및 SELECT 특권
- INSERT_UPDATE 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.

  - dataaccess
  - 테이블, 뷰 또는 별칭의 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT, SELECT, UPDATE 및 DELETE 특권
- REPLACE 또는 REPLACE_CREATE 옵션을 사용하여 기존 테이블로 IMPORT 하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 테이블 또는 뷰에 대한 CONTROL 특권
  - 테이블 또는 뷰에 대한 INSERT, SELECT 및 DELETE 특권
- CREATE 또는 REPLACE_CREATE 옵션을 사용하여 새 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dbadm
  - 데이터베이스에 대한 CREATETAB 권한 및 테이블 스페이스에 대한 USE 특권은 다음 경우 중 하나입니다.
    - 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
- CREATE 또는 REPLACE_CREATE 옵션을 사용하여 존재하지 않는 테이블이나 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dbadm
  - 데이터베이스에 대한 CREATETAB 권한 및 다음 중 하나가 있어야 합니다.
    - 테이블의 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마가 존재할 경우, 스키마에 대한 CREATEIN 특권
    - 전체 계층 구조에 대한 REPLACE_CREATE 옵션이 사용될 경우, 계층 구조의 모든 서브테이블에 대한 CONTROL 특권
- REPLACE 옵션을 사용하여 기존 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 계층 구조에 있는 모든 서브테이블에 대한 CONTROL 특권

필수 연결

데이터베이스. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다.

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Import (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ImportIn *piImportInfoIn;
   struct db2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   struct sqllob *piLongActionString;
} db2ImportStruct;

typedef SQL_STRUCTURE db2ImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2ImportIn;

typedef SQL_STRUCTURE db2ImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2ImportOut;

typedef SQL_STRUCTURE db2DMUXmlMapSchema
{
   struct db2Char                       iMapFromSchema;
   struct db2Char                       iMapToSchema;
} db2DMUXmlMapSchema;

typedef SQL_STRUCTURE db2DMUXmlValidateXds
{
```

```
   struct db2Char *piDefaultSchema;
   db2Uint32 iNumIgnoreSchemas;
   struct db2Char *piIgnoreSchemas;
   db2Uint32 iNumMapSchemas;
   struct db2DMUXmlMapSchema *piMapSchemas;
} db2DMUXmlValidateXds;

typedef SQL_STRUCTURE db2DMUXmlValidateSchema
{
   struct db2Char *piSchema;
} db2DMUXmlValidateSchema;

typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16 iUsing;
   struct db2DMUXmlValidateXds *piXdsArgs;
   struct db2DMUXmlValidateSchema *piSchemaArgs;
} db2DMUXmlValidate;

SQL_API_RC SQL_API_FN
  db2gImport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2gImportIn *piImportInfoIn;
   struct dbg2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct sqllob *piLongActionString;
} db2gImportStruct;

typedef SQL_STRUCTURE db2gImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2gImportIn;

typedef SQL_STRUCTURE db2gImportOut
```

```
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2gImportOut;
```

db2Import API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입/출력(I/O). db2ImportStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ImportStruct 데이터 구조 매개변수

piDataFileName

입력. 데이터가 임포트되는 외부 입력 파일의 경로 및 이름이 포함된 문자열.

piLobPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 LOB 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터. 별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

piDataDescriptor

입력. 외부 파일에서 임포트하도록 선택한 컬럼에 대한 정보가 포함된 sqldcol 구조의 포인터. dcolmeth 필드 값으로 이 매개변수에서 제공되는 나머지 정보가 임포트 유틸리티로 해석되는 방법을 판별합니다. 이 매개변수의 유효한 값은 다음과 같습니다.

SQL_METH_N

이름. 외부 입력 파일 컬럼 선택은 컬럼 이름으로 수행됩니다.

SQL_METH_P

위치. 외부 입력 파일 컬럼 선택은 컬럼 위치로 수행됩니다.

SQL_METH_L

위치. 외부 입력 파일 컬럼 선택은 컬럼 위치로 수행됩니다. 데이터베이스 관리 프로그램은 다음 조건 중 하나로 인해 유효하지 않게 되는 위치 쌍이 있는 임포트 호출을 거부합니다.

- 시작 또는 종료 위치가 1에서 가장 큰 부호있는 2바이트 정수 사이가 아닙니다.
- 종료 위치가 시작 위치보다 작습니다.
- 입력 쌍으로 정의된 입력 컬럼 너비가 목표 컬럼의 유형 및 길이와 호환되지 않습니다.

두 위치의 위치 쌍이 0인 경우 NULL로 채워지는 널(NULL) 입력 가능 컬럼임을 나타냅니다.

SQL_METH_D

디폴트. piDataDescriptor가 NULL이거나 SQL_METH_D로 설정된 경우 외부 입력 파일의 디폴트 컬럼 선택이 수행됩니다. 이 경우 컬럼 수 및 컬럼 스펙 배열 모두 무시됩니다. DEL, IXF 또는 WSF 파일의 경우 외부 입력 파일 데이터의 첫 번째 n 컬럼은 기본 순서대로 사용되며 여기서 n은 데이터가 임포트되는 데이터베이스 컬럼 수입니다.

piActionString

사용되지 않습니다. piLongActionString으로 교체됩니다.

piLongActionString

입력. 테이블에 영향을 주는 조치를 지정하는 문자 배열이 뒤에 오는 4바이트 길이의 필드를 포함하는 *sqllob* 구조의 포인터.

문자 배열은 다음과 같은 양식입니다.

```
{INSERT | INSERT_UPDATE | REPLACE | CREATE | REPLACE_CREATE}
INTO {tname[(tcolumn-list)] |
[{ALL TABLES | (tname[(tcolumn-list)][, tname[(tcolumn-list)]])}]
[IN] HIERARCHY {STARTING tname | (tname[, tname])}
[UNDER sub-table-name | AS ROOT TABLE]}
```

INSERT

기존 테이블 데이터를 변경하지 않고 임포트된 데이터를 테이블에 추가합니다.

INSERT_UPDATE

임포트된 행의 기본 키가 테이블에 없는 경우 해당 행을 추가하고 해당 기본 키가 있는 경우 이를 갱신에 사용합니다. 이 옵션은 목표 테이블에 기본 키가 있고 임포트 중인 목표 컬럼의 지정된(또는 내재된) 목록에 기본 키의 모든 컬럼이 포함된 경우에만 유효합니다. 이 옵션은 뷰에는 적용할 수 없습니다.

REPLACE

테이블 오브젝트를 잘라내어 테이블에서 모든 기존 데이터를 삭제한 후 임포트된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되

지 않습니다. (indexixf가 FileTypeMod에 있고 FileType이 SQL_IXF인 경우 인덱스가 삭제되어 교체됩니다.) 테이블이 이미 정의되어 있지 않은 경우에는 오류가 리턴됩니다.

주: 기존 데이터가 삭제된 후에 오류가 발생하면 해당 데이터는 손실됩니다.
별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

CREATE

주: CREATE 매개변수는 더 이상 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항은 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE가 사용되지 않음』을 참조하십시오.

지정한 테이블이 정의되지 않은 경우에는 테이블 정의 및 지정된 PC/IXF 파일의 정보를 사용하는 행 컨텐츠를 작성합니다. DB2에서 이전에 파일을 익스포트하지 않은 경우에는 인덱스도 작성됩니다. 지정한 테이블이 이미 정의되어 있는 경우에는 오류가 리턴됩니다. 이 옵션은 PC/IXF 파일 형식에서만 유효합니다. 별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

REPLACE_CREATE

주: REPLACE_CREATE 매개변수는 더 이상 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항은 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE가 사용되지 않음』을 참조하십시오.

지정한 테이블이 정의된 경우 PC/IXF 파일의 PC/IXF 행 정보를 사용하여 테이블 컨텐츠를 교체합니다. 테이블이 아직 정의되어 있지 않은 경우에는 테이블 정의 및 행 컨텐츠가 지정한 PC/IXF 파일의 정보를 사용하여 작성됩니다. DB2에서 이미 PC/IXF 파일을 익스포트한 경우 인덱스도 작성됩니다. 이 옵션은 PC/IXF 파일 형식에서만 유효합니다.

주: 기존 데이터가 삭제된 후에 오류가 발생하면 해당 데이터는 손실됩니다.
별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

tname 데이터가 삽입되는 테이블, 유형이 지정된 테이블, 뷰 또는 오브젝트 뷰의 이름. 규정되거나 규정되지 않은 이름을 지정해야 하는 경우 REPLACE, INSERT_UPDATE 또는 INSERT의 별명은 지정할 수 있습니다(DB2 제품의 이전 버전이 설치된 서버는 제외). 뷰인 경우 읽기 전용 뷰는 불가능합니다.

tcolumn-list

데이터가 삽입되는 테이블 또는 뷰 컬럼 이름 목록. 컬럼 이름은 쉼표로 구분해야 합니다. 컬럼 이름을 지정하지 않으면 CREATE TABLE 또는 ALTER TABLE 문의 컬럼 이름이 사용됩니다. 유형이 지정된 테이블에 대해 컬럼 목록을 지정하지 않은 경우 데이터가 각 서브테이블의 모든 컬럼에 삽입됩니다.

sub-table-name

CREATE 옵션에 하나 이상의 서브테이블을 작성하는 경우 상위 테이블을 지정합니다.

ALL TABLES

계층 구조 전용 내재적 키워드. 계층 구조를 임포트할 때 traversal-order-list로 지정된 모든 테이블을 임포트하는 것이 디폴트입니다.

HIERARCHY

계층 구조 데이터가 임포트되도록 지정합니다.

STARTING

계층 구조의 키워드 전용. 지정한 서브테이블 이름을 시작으로 디폴트 순서가 사용되도록 지정합니다.

UNDER

계층 구조 및 CREATE 키워드 전용. 지정한 서브테이블에 새 계층 구조, 하위 계층 구조 또는 서브테이블이 작성되도록 지정합니다.

AS ROOT TABLE

계층 구조 및 CREATE 키워드 전용. 새 계층 구조, 하위 계층 구조 또는 서브테이블이 독립형 계층 구조로 작성되도록 지정합니다.

tname 및 tcolumn-list 매개변수는 SQL INSERT 문의 tablename 및 colname에 대응하며 동일한 제한사항이 적용됩니다.

tcolumn-list의 컬럼 및 외부 컬럼(지정 또는 내재)은 목록이나 구조의 위치에 따라 일치됩니다(sqldcol 구조에 지정된 첫 번째 데이터가 tcolumn-list의 첫 번째 요소에 따라 테이블이나 뷰 필드에 삽입됨).

동일하지 않은 컬럼 수가 지정된 경우 실제로 처리된 컬럼 수는 두 수 중 작은 수입니다. 이 경우 오류가 발생하거나(일부 널(null) 값을 허용하지 않는 테이블 필드에 값이 없기 때문에) 또는 정보 메시지(일부 외부 파일 컬럼이 무시되기 때문에)가 표시될 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

piFileType

입력. 외부 파일 내의 데이터 형식을 나타내는 문자열. 지원되는 외부 파일 형식은 다음과 같습니다.

SQL_ASC

컬럼 식별자가 없는 ASCII.

SQL_DEL

컬럼 식별자가 있는 ASCII, dBase, BASIC 및 IBM Personal Decision Series 프로그램과 교환에 사용되는 다수의 데이터베이스 관리 프로그램 및 파일 관리자

SQL_IXF

통합 교환 형식의 PC 버전으로 테이블에서 데이터를 익스포트할 때 이후에 동일한 테이블이나 다른 데이터베이스 관리 프로그램 테이블에 임포트할 수 있기 때문에 자주 사용되는 방법입니다.

SQL_WSF

Lotus Symphony 및 1-2-3 프로그램과의 교환을 위한 워크시트 형식. 별칭으로 임포트할 때 WSF 파일 유형은 지원되지 않습니다.

piFileTypeMod

입력. 하나 이상의 처리 옵션을 지정하는 문자 배열이 뒤에 오는 2바이트 길이의 필드를 포함하는 구조의 포인터. 이 포인터가 NULL이거나 지시된 구조에 영(0) 문자가 있는 경우 이 조치는 디폴트 스펙 선택으로 해석됩니다.

지원되는 모든 파일 유형에 모든 옵션을 사용할 수 있는 것은 아닙니다. 관련 링크인 "임포트 유틸리티의 파일 유형 수정자"를 참조하십시오.

piMsgFileName

입력. 유틸리티에서 리턴된 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우 추가됩니다. 없는 경우에는 파일이 작성됩니다.

iCallerAction

입력. 호출자가 요청한 조치. 가능한 값은 다음과 같습니다.

SQLU_INITIAL

초기 호출. 이 값을 API의 첫 번째 호출에 사용해야 합니다. 초기 호출이나 이후에 임의의 후속 호출이 리턴되고 요청된 임포트 조작을 완료하기 전에 일부 조치를 수행하는 응용프로그램을 호출해야 하는 경우 호출자 조치는 다음 중 하나로 설정해야 합니다.

SQLU_CONTINUE

처리를 계속합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 완료되고 유틸리티는 초기 요청 처리를 계속하도록 지정합니다.

SQLU_TERMINATE

처리를 종료합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테

이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 수행되지 않고 유틸리티가 초기 요청 처리를 종료하도록 지정합니다.

piImportInfoIn

입력. db2ImportIn 구조의 포인터

poImportInfoOut

출력. db2ImportOut 구조의 포인터

piNullIndicators

입력. ASC 파일에만 해당. 컬럼 데이터가 널(NULL) 입력 가능한지 여부를 표시하는 정수 배열. 이 배열의 요소 수는 입력 파일의 컬럼 수와 일치해야 합니다. 이 배열 요소와 데이터 파일에서 임포트 중인 컬럼은 1대1 대응 방식입니다. 따라서 요소 수는 piDataDescriptor 매개변수의 dcolnum 필드와 동일해야 합니다. 배열의 각 요소에는 널(NULL) 표시기 필드로 사용되는 데이터 파일의 컬럼을 식별하는 번호 또는 테이블 컬럼이 널(NULL) 입력 가능한 것을 표시하는 영(0)이 포함됩니다. 요소가 영(0)이 아닌 경우 데이터 파일의 식별된 컬럼에는 Y 또는 N이 포함됩니다. Y는 테이블 컬럼 데이터가 NULL임을 표시하고 N은 테이블 컬럼 데이터가 NULL이 아닌 것을 표시합니다.

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 XML 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터.

db2ImportIn 데이터 구조 매개변수

iRowcount

입력. 로드할 실제 레코드 수. 사용자가 파일의 첫 번째 iRowcount 행만 로드하도록 허용합니다. iRowcount가 0인 경우 임포트는 파일에서 모든 행을 처리하려고 합니다.

iRestartcount

입력. 레코드의 삽입 또는 갱신을 시작하기 전에 건너뛰어야 하는 레코드 수. 기능상으로는 iSkipcount 매개변수와 동등합니다. iRestartcount 및 iSkipcount 매개변수는 상호 배타적입니다.

iSkipcount

입력. 레코드의 삽입 또는 갱신을 시작하기 전에 건너뛰어야 하는 레코드 수. 기능상으로는 iRestartcount와 동등합니다.

piCommitcount

입력. 데이터베이스에 레코드를 커미트하기 전에 임포트하려는 레코드 수. piCommitcount 레코드가 임포트될 때마다 커미트가 수행됩니다. NULL 값은 디폴트 커미트 계수 값을 지정하고 오프라인 임포트의 경우 영(0)이며 온라인

임포트의 경우 AUTOMATIC입니다. Commitcount AUTOMATIC은 DB2IMPORT_COMMIT_AUTO 값을 전달하여 지정합니다.

iWarningcount

입력. iWarningcount 경고 후에 임포트 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 임포트 파일 또는 목표 테이블이 올바르지 않게 지정될 경우 임포트 유틸리티는 임포트하려는 각 행에 대해 경고를 생성하므로 임포트를 실패하게 됩니다.

iWarningcount가 0이거나 이 옵션을 지정하지 않은 경우 임포트 조작은 발행된 경고 수에 상관 없이 계속됩니다.

iNoTimeout

입력. 임포트 유틸리티가 잠금 대기 중 시간종료되지 않도록 지정합니다. 이 옵션은 locktimeout 데이터베이스 구성 매개변수를 대체합니다. 다른 응용프로그램에는 영향을 주지 않습니다.가능한 값은 다음과 같습니다.

DB2IMPORT_LOCKTIMEOUT

locktimeout 구성 매개변수 값이 사용되도록 표시합니다.

DB2IMPORT_NO_LOCKTIMEOUT

시간종료가 없음을 표시합니다.

iAccessLevel

입력. 액세스 등급을 지정합니다. 가능한 값은 다음과 같습니다.

- SQLU_ALLOW_NO_ACCESS

임포트 유틸리티가 테이블을 배타적으로 잠그도록 지정합니다.

- SQLU_ALLOW_WRITE_ACCESS

임포트가 진행 중인 동안에도 판독기 및 기록기가 테이블의 데이터에 액세스할 수 있도록 지정합니다.

첫 번째 행이 삽입될 때 목표 테이블에 대해 의도를 가진 독점(IX) 잠금을 획득합니다. 이렇게 하면 동시 판독기 및 기록기가 테이블 데이터에 액세스할 수 있습니다. 온라인 모드는 REPLACE, CREATE 또는 REPLACE_CREATE 임포트 옵션과 호환 가능하지 않습니다. 온라인 모드는 버퍼링된 삽입과 함께 지원되지 않습니다. 임포트 조작은 테이블 잠금에 대한 잠금 에스컬레이션을 방지하고 사용 중인 로그 스페이스 외부에서 실행되지 않도록 하기 위해 삽입된 데이터를 주기적으로 커미트합니다. piCommitCount가 사용되지 않는 경우에도 이 커미트가 수행됩니다. 각 커미트를 수행하는 동안 임포트는 해당 IX 테이블 잠금을 유실하므로 커미트 후 이를 다시 획득하려고 시도합니다. 별칭으로 임포트할 때 이 매개변수가 필요하며 유효한 숫자를 사용하여 piCommitCount 매개변수를 지정해야 합니다(AUTOMATIC은 유효한 옵션으로 간주하지 않음).

piXmlParse

입력. XML 문서에 대해 수행해야 하는 구문 분석 유형. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

DB2DMU_XMLPARSE_PRESERVE_WS

공백은 유지해야 합니다.

DB2DMU_XMLPARSE_STRIP_WS

공백은 스트립되어야 합니다.

piXmlValidate

입력. db2DMUXmlValidate 구조의 포인터. XML 스키마 유효성 확인이 XML 문서에 대해 수행되어야 하는 것을 나타냅니다.

db2ImportOut 데이터 구조 매개변수

oRowsRead

출력. 임포트 중 파일에서 읽은 레코드 수

oRowsSkipped

출력. 삽입 또는 갱신이 시작되기 전에 건너뛴 레코드 수

oRowsInserted

출력. 목표 테이블로 삽입된 행 수

oRowsUpdated

출력. 임포트된 레코드(테이블에 기본 키 값이 이미 존재하는 레코드)의 정보로 갱신된 목표 테이블의 행 수

oRowsRejected

출력. 임포트할 수 없는 레코드 수

oRowsCommitted

출력. 데이터베이스에 임포트되고 커미트된 레코드 수

db2DMUXmlMapSchema 데이터 구조 매개변수

iMapFromSchema

입력. 맵핑 원본의 XML 스키마에 대한 SQL ID.

iMapToSchema

입력. 맵핑 대상의 XML 스키마에 대한 SQL ID.

db2DMUXmlValidateXds 데이터 구조 매개변수

piDefaultSchema

입력. XDS에 SCH 속성이 포함되지 않은 경우 유효성 확인에 사용해야 하는 XML 스키마의 SQL ID.

iNumIgnoreSchemas

입력. XDS에서 SCH 속성으로 참조되는 경우 XML 스키마 유효성 확인에서 무시되는 XML 스키마 수.

piIgnoreSchemas

입력. XDS에서 SCH 속성으로 참조되는 경우 XML 스키마 유효성 확인에서 무시되는 XML 스키마 목록.

iNumMapSchemas

입력. XML 스키마 유효성 확인 중에 맵핑되는 XML 스키마 수. 스키마 맵 쌍의 첫 번째 스키마는 XDS의 SCH 속성으로 참조됩니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

piMapSchemas

입력. XML 스키마 쌍 목록으로 여기서 각 쌍은 임의의 스키마에서 다른 스키마로의 맵핑을 나타냅니다. 쌍에서 첫 번째 스키마는 XDS에 있는 SCH 속성에 의해 참조되는 스키마를 나타냅니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

db2DMUXmlValidateSchema 데이터 구조 매개변수

piSchema

입력. 사용하려는 XML 스키마의 SQL ID.

db2DMUXmlValidate 데이터 구조 매개변수

iUsing 입력. XML 스키마 유효성 확인 수행에 사용하려는 스펙. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

- **DB2DMU_XMLVAL_XDS**

  XDS에 따라 유효성 확인이 수행되어야 합니다. 이는 CLP ″XMLVALIDATE USING XDS″ 절에 대응합니다.

- **DB2DMU_XMLVAL_SCHEMA**

  지정한 스키마에 따라 유효성 확인이 수행되어야 합니다. 이는 CLP ″XMLVALIDATE USING SCHEMA″ 절에 대응합니다.

- **DB2DMU_XMLVAL_SCHEMALOC_HINTS**

  XML 문서에 있는 schemaLocation 힌트에 따라 유효성 확인이 수행되어야 합니다. 이는 ″XMLVALIDATE USING SCHEMALOCATION HINTS″ 절에 대응합니다.

piXdsArgs

입력. CLP ″XMLVALIDATE USING XDS″ 절에 대응하는 인수를 나타내는 db2DMUXmlValidateXds 구조의 포인터.

이 매개변수는 동일한 구조의 iUsing 매개변수가 DB2DMU_XMLVAL_XDS로 설정된 경우에만 적용됩니다.

piSchemaArgs

입력. CLP ″XMLVALIDATE USING SCHEMA″ 절에 대응하는 인수를 나타내는 db2DMUXmlValidateSchema 구조의 포인터.

이 매개변수는 동일한 구조의 iUsing 매개변수가 DB2DMU_XMLVAL_SCHEMA로 설정된 경우에만 적용됩니다.

db2gImportStruct 데이터 구조 특정 매개변수

iDataFileNameLen

입력. piDataFileName 매개변수 길이를 바이트 단위로 지정합니다.

iFileTypeLen

입력. piFileType 매개변수 길이를 바이트 단위로 지정합니다.

iMsgFileNameLen

입력. piMsgFileName 매개변수 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

임포트 조작을 시작하기 전에 다음 중 하나의 방법을 사용하여 모든 테이블 조작을 완료하고 모든 잠금을 해제해야 합니다.

- WITH HOLD 절로 정의된 모든 열려 있는 커서를 닫고 COMMIT 문을 실행하여 데이터 변경을 커미트하십시오.
- ROLLBACK 문을 실행하여 데이터 변경을 롤백하십시오.

임포트 유틸리티는 SQL INSERT문을 사용하여 목표 테이블에 행을 추가합니다.

이 유틸리티는 입력 파일에 있는 각 데이터 행마다 하나의 INSERT문을 발행합니다. INSERT문을 실패하면 다음 두 가지 조치 중 하나가 발생합니다.

- 후속 INSERT문이 성공할 수 있을 것 같으면 메시지 파일에 경고 메시지를 작성하고 처리를 계속합니다.
- 후속 INSERT문이 실패할 것 같으며 데이터베이스 손상이 생길 가능성이 있으면 메시지 파일에 오류 메시지를 작성하고 처리를 정지합니다.

이 유틸리티는 REPLACE 또는 REPLACE_CREATE 조작 중 이전 행이 삭제된 후 자동 COMMIT를 수행합니다. 따라서 테이블 오브젝트가 절단된 후에 시스템이 실패하거나 응용프로그램이 데이터베이스 관리 프로그램을 인터럽트하면 이전 데이터 모두가 유실됩니다. 이들 옵션을 사용하기 전에 이전 데이터가 더 이상 필요하지 않은지 확인하십시오.

CREATE, REPLACE 또는 REPLACE_CREATE 조작 중 로그가 가득차게 되면 이 유틸리티는 삽입된 레코드에 대해 자동 COMMIT를 수행합니다. 자동 COMMIT 후에 시스템이 실패하거나 응용프로그램이 데이터베이스 관리 프로그램을 인터럽트하면 부분적인 데이터가 포함된 테이블이 데이터베이스에 남게 됩니다. REPLACE 또는 REPLACE_CREATE 옵션을 사용하여 전체 임포트 조작을 재실행하거나 성공적으로 임포트된 행 수로 설정된 iRestartcount 매개변수를 사용하여 INSERT를 사용하십시오.

디폴트로 INSERT 또는 INSERT_UPDATE 옵션에 대해서는 자동 COMMIT가 수행되지 않습니다. 그러나 *piCommitcount 매개변수가 영(0)이 아닌 경우에는 수행됩니다. 전체 로그가 ROLLBACK됩니다.

임포트 유틸리티가 COMMIT를 수행할 때마다 메시지 파일에 두 개의 메시지가 작성되는데, 하나는 커미트되는 레코드 수를 나타내고 다른 하나는 COMMIT 완료 후 작성됩니다. 실패 후 임포트 조작을 재시작할 때 마지막 완료된 COMMIT에서 판별된 대로 생략할 레코드 수를 지정하십시오.

임포트 유틸리티는 사소한 비호환성 문제점이 있는 입력 데이터는 승인합니다(예를 들어, 문자 데이터가 채우기 또는 절단을 사용하여 임포트되고 숫자 데이터가 다른 숫자 데이터 유형으로 임포트될 수 있습니다). 그러나 주요한 비호환성 문제점이 있는 데이터는 승인되지 않습니다.

오브젝트 테이블이 자체 테이블 외에 하위 테이블이 있는 경우 이 오브젝트 테이블을 REPLACE 또는 REPLACE_CREATE 할 수 없으며, 기본 테이블에 하위 테이블(자체 테이블을 포함하여)이 있는 경우 해당 오브젝트 뷰를 REPLACE 또는 REPLACE_CREATE 할 수 없습니다. 이러한 테이블 또는 뷰를 바꾸려면 다음을 수행하십시오.

1. 해당 테이블이 상위 테이블인 모든 외부 키를 삭제하십시오.
2. 임포트 유틸리티를 실행하십시오.
3. 테이블을 변경하여 외부 키를 재작성하십시오.

외부 키를 재작성하는 동안 오류가 발생하면 데이터를 수정하여 참조 무결성을 유지보수하십시오.

PC/IXF 파일에서 테이블을 작성할 때 참조 제한조건 및 외부 키 정의는 보존되지 않습니다. (기본 키 정의는 데이터가 이전에 SELECT *를 사용하여 익스포트된 경우에 보존됩니다.)

리모트 데이터베이스로 임포트하려면 입력 데이터 파일의 사본, 출력 메시지 파일 및 잠재적인 데이터베이스 크기 확장에 필요한 디스크 스페이스가 충분해야 합니다.

리모트 데이터베이스에 대해 임포트 조작이 수행되고 출력 메시지 파일이 매우 긴 경우 (60KB 초과), 클라이언트의 사용자에게 리턴되는 메시지 파일은 임포트 조작 중간부터 메시지가 누락될 수 있습니다. 메시지 정보의 처음 30KB와 메시지 정보의 마지막 30KB는 항상 보유됩니다.

piDataDescriptor에 대해 디폴트가 아닌 값 또는 piLongActionString의 테이블 컬럼에 대한 내재적인 목록을 지정하면 리모트 데이터베이스로의 임포트가 느려집니다.

ASC, DEL 또는 WSF 파일 형식의 데이터를 임포트하려면 먼저 데이터베이스 테이블 또는 계층 구조가 존재해야 합니다. 그러나 테이블이 이미 존재하지 않을 경우 IMPORT CREATE 또는 IMPORT REPLACE_CREATE가 PC/IXF 파일에서 데이터를 임포트할 때 테이블을 작성합니다. 유형이 지정된 테이블에서 IMPORT CREATE는 자료형 계층 구조와 테이블 계층 구조를 작성할 수 있습니다.

데이터베이스 간에 데이터를 이동하려면(계층 데이터를 포함하여) PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다.

ASC 및 DEL 파일의 데이터는 임포트를 수행하는 클라이언트 응용프로그램의 코드 페이지에 있는 것으로 간주됩니다. 서로 다른 코드 페이지의 데이터를 임포트할 때는 서로 다른 코드 페이지에 사용할 수 있는 PC/IXF 파일을 사용할 것을 권장합니다. PC/IXF 파일과 임포트 유틸리티가 동일한 코드 페이지에 있을 경우 일반 응용프로그램에 대해서와 같은 처리가 발생합니다. 두 코드 페이지가 서로 다르고 FORCEIN 옵션이 지정된 경우, 임포트 유틸리티는 PC/IXF 파일의 데이터가 임포트를 수행 중인 응용프로그램과 동일한 코드 페이지를 가지고 있다고 간주합니다. 이는 두 코드 페이지에 대한 변환 테이블이 있는 경우에도 마찬가지입니다. 두 코드 페이지가 서로 다르고, FORCEIN 옵션이 지정되지 않았으며 변환 테이블이 있는 경우, PC/IXF 파일의 모든 데이터는 파일 코드 페이지에서 응용프로그램 코드 페이지로 변환됩니다. 두 코드 페이지가 서로 다르고 FORCEIN 옵션이 지정되지 않았으며 변환 테이블이 없는 경우 임포트 조작은 실패합니다. 이는 AIX용 DB2 클라이언트에서 PC/IXF 파일에만 적용됩니다.

1012 컬럼의 제한에 근접한 8KB 페이지에 있는 테이블 오브젝트의 경우, PC/IXF 데이터 파일을 임포트하면 SQL 문의 최대 크기를 초과했기 때문에 DB2가 오류를 리턴할 수 있습니다. 이러한 상황은 컬럼 유형이 CHAR, VARCHAR 또는 CLOB인 경우에만 발생할 수 있습니다. DEL 또는 ASC 파일을 임포트하는 데는 이러한 제한사항이 적용되지 않습니다.

DB2 Connect를 사용하여 데이터를 OS/390용 DB2, VM 및 VSE용 DB2 및 OS/400용 DB2와 같은 DRDA 서버로 임포트할 수 있습니다. PC/IXF 임포트(INSERT 옵션)만 지원됩니다. commitcnt 매개변수가 아닌 restartcnt 매개변수도 지원됩니다.

유형이 지정된 테이블에서 CREATE 옵션을 사용할 때 PC/IXF 파일에 정의된 모든 서브테이블을 작성하십시오. 서브테이블 정의는 변경할 수 없습니다. 유형이 지정된 테이블에서 CREATE 이외의 옵션을 사용할 때, 트래버스 순서 목록을 통해 해당 옵션이 트래버스 순서를 지정하므로 트래버스 순서 목록은 익스포트 조작시 사용된 옵션과 일치해야 합니다. PC/IXF 파일 형식의 경우, 옵션은 목표 서브테이블 이름만 지정해야 하며 파일에 저장된 트래버스 순서를 사용해야 합니다. 임포트 유틸리티를 사용하여 이전에 PC/IXF 파일로 익스포트된 테이블을 복구할 수 있습니다. 이 테이블은 익스포트될 때의 상태로 리턴합니다.

데이터는 시스템 테이블, 선언된 임시 테이블, 작성된 임시 테이블 또는 요약 테이블로 임포트할 수 없습니다.

임포트 유틸리티를 통해 뷰를 작성할 수 없습니다.

Windows 운영 체제에서,

- 논리적으로 분할된 PC/IXF 파일 임포트는 지원되지 않습니다.
- 잘못된 형식의 PC/IXF 또는 WSF 파일 임포트는 지원되지 않습니다.

페더레이티드 고려사항

db2Import API 및 INSERT, UPDATE 또는 INSERT_UPDATE 매개변수를 사용하는 경우 참여 중인 별칭에 CONTROL 특권이 있는지 확인해야 합니다. 임포트 조작을 수행할 때 사용할 별칭이 이미 존재하는지 확인해야 합니다.

임포트 세션 - CLP 예

예 1

다음 예에서는 myfile.ixf에서 STAFF 테이블로 정보를 임포트하는 방법을 보여줍니다.

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff

SQL3150N  PC/IXF 파일의 H 레코드는 제품이 "DB2    01.00", 날짜가
"19970220", 시간이 "140848"입니다.

SQL3153N  PC/IXF 파일에서 T 레코드의 이름은 "myfile",
규정자는 "        "이고, 소스는 "            "입니다.

SQL3109N  유틸리티가 파일 "myfile"에서 데이터를 로드하기 시작합니다.

SQL3110N  유틸리티가 처리를 완료했습니다.
입력 파일에서 "58"개의 행을 읽었습니다.

SQL3221W  ...COMMIT WORK 시작. 입력 레코드 계수 = "58".

SQL3222W  데이터베이스 변경사항의 ...COMMIT가 성공적이었습니다.

SQL3149N  입력 파일에서 "58"개의 행이 처리되었습니다.
"58"개의 행이
테이블에 삽입되었습니다. "0"개의 행이 거부되었습니다.
```

예 2

다음 예에서는 ID 컬럼이 있는 테이블로 임포트하는 방법을 보여줍니다.

TABLE1에는 다음과 같이 4개의 컬럼이 있습니다.

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2는 C2가 GENERATED ALWAYS ID 컬럼이라는 점만 제외하고 TABLE1과 동일합니다.

DATAFILE1의 데이터 레코드(DEL 형식):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2의 데이터 레코드(DEL 형식):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

다음 명령은 DATAFILE1에 행 1과 행 2에 대한 식별 값이 제공되지 않았기 때문에 이들 행에 대한 식별 값을 생성합니다. 그러나 행 3 및 4에는 각각 사용자 제공 ID 값 100 및 101이 지정됩니다.

```
db2 import from datafile1.del of del replace into table1
```

모든 행에서 ID 값이 생성되도록 DATAFILE1을 TABLE1로 임포트하려면 다음 명령 중 하나를 실행하십시오.

```
db2 import from datafile1.del of del method P(1, 3, 4)
      replace into table1 (c1, c3, c4)   db2 import from
      datafile1.del of del modified by identityignore
      replace into table1
```

각 행에서 ID 값이 생성되도록 DATAFILE2를 TABLE1로 임포트하려면 다음 명령 중 하나를 실행하십시오.

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
   db2 import from datafile2.del of del modified by identitymissing
      replace into table1
```

ID 관련 파일 유형 수정자를 사용하지 않고 DATAFILE1이 TABLE2로 임포트된 경우 행 1 및 2는 삽입되지만 행 3 및 4는 거부됩니다. 고유한 널(NULL)이 아닌 값을 제공하지 않았으며 ID 컬럼이 GENERATED ALWAYS이기 때문입니다.

예 3

다음 예에서는 널(NULL) 표시기가 있는 테이블로 임포트하는 방법을 보여줍니다.

TABLE1에는 다음과 같이 5개의 컬럼이 있습니다.

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1에는 다음 6개 요소가 있습니다.

- ELE1 위치 01 - 20
- ELE2 위치 21 - 22
- ELE5 위치 23 - 23
- ELE3 위치 24 - 27
- ELE4 위치 28 - 31
- ELE6 위치 32 - 32
- ELE6 위치 33 - 40

데이터 레코드:

```
1...5....10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    wxyzN
Test data 4,5 and 6 WWN6789    Y
```

다음 명령은 ASCFILE1에서 TABLE1로 레코드를 임포트합니다.

```
db2 import from ascfile1 of asc
method L (1 20, 21 22, 24 27, 28 31)
null indicators (0, 0, 23, 32)
insert into table1 (col1, col5, col2, col3)
```

주:

1. 입력 파일에 COL4가 제공되지 않았으므로 디폴트값(NOT NULL WITH DEFAULT 정의됨)을 사용하여 TABLE1로 삽입됩니다.
2. 위치 23 및 32를 사용하여 TABLE1의 COL2 및 COL3이 지정된 행에 널(NULL)로 로드되었는지 여부를 표시합니다. 지정된 레코드에서 컬럼의 널(NULL) 표시기 위치에 Y가 있으면 컬럼은 널(NULL)입니다. N이 있으면 입력 레코드에서 컬럼의 데

이터 위치에 지정된 데이터 값(L(.....)로 정의됨)이 행의 컬럼 데이터 소스로 사용됩니다. 이 예에서 행 1의 컬럼은 널(NULL)이 아니며 행 2의 COL2는 널(NULL)이고 행 3의 COL3도 널(NULL)입니다.

3. 이 예에서 COL1 및 COL5의 NULL INDICATORS는 영(0)으로 지정되며 데이터가 널(NULL) 입력 가능하지 않음을 나타냅니다.
4. 지정된 컬럼의 NULL INDICATOR는 입력 레코드 어느 위치에도 나타날 수 있지만 위치를 지정해야 하며 `Y` 또는 `N` 값을 제공해야 합니다.

제 4 장 로드 유틸리티

로드 개요

로드 유틸리티에서는 새로 작성한 테이블 또는 이미 데이터를 포함하는 테이블로 많은 양의 데이터를 효과적으로 이동할 수 있습니다. 이 유틸리티에서는 XML, 대형 오브젝트(LOB) 및 사용자 정의 유형(UDT)을 포함하여 대부분의 데이터 유형을 처리할 수 있습니다. 로드 유틸리티는 데이터베이스에 직접 형식화된 페이지를 작성하는 반면, 임포트 유틸리티는 SQL INSERT를 수행하므로 로드 유틸리티가 임포트 유틸리티보다 더 빠릅니다. 로드 유틸리티는 트리거를 시작하지 않으며 참조 또는 테이블 제한조건 검사(인덱스 고유성 확인 작업 제외)를 수행하지 않습니다.

로드 프로세스는 다음과 같이 구별된 4단계로 구성됩니다(그림 3 참조).

1. **로드**
   로드 단계에서 데이터는 테이블로 로드되고 필요하면 인덱스 키 및 테이블 통계가 수집됩니다. *저장 시점* 또는 일관성 지점이 LOAD 명령의 **SAVECOUNT** 매개변수를 통해 지정된 간격으로 설정됩니다. 저장 시점에 성공적으로 로드된 입력 행 수를 표시하는 메시지가 생성됩니다.
2. **빌드**
   빌드 단계에서는 로드 단계 중 수집된 인덱스 키에 따라 인덱스가 생성됩니다. 인덱스 키는 로드 단계 중 정렬되고 인덱스 통계가 수집됩니다(STATISTICS USE PROFILE 옵션이 지정되고, 프로파일에서 수집한 인덱스 통계를 표시하는 경우). 이 통계는 RUNSTATS 명령을 통해 수집한 통계와 비슷합니다.
3. **삭제**
   삭제 단계에서는 고유 또는 1차 키를 위반한 행이 테이블에서 삭제됩니다. 이와 같이 삭제된 행은 로드 예외 테이블(지정된 경우)에 저장됩니다.
4. **인덱스 복사**
   인덱스 복사 단계에서는 인덱스 데이터가 시스템 임시 테이블 스페이스에서 원래 테이블 스페이스로 복사됩니다. READ ACCESS 옵션이 지정된 로드 조작 중 인덱스 작성에 대해 시스템 임시 테이블 스페이스가 지정된 경우에만 수행됩니다.

그림 3. 로드 프로세스의 4단계: 로드, 빌드, 삭제 및 인덱스 복사

주: 로드 유틸리티를 호출한 후에는 LIST UTILITIES 명령을 사용하여 로드 조작의 진행을 모니터할 수 있습니다.

데이터를 로드하는 경우 다음 정보가 필요합니다.

- 입력 파일, Named Pipe 또는 디바이스의 경로 및 이름
- 목표 테이블의 이름 또는 별명
- 입력 소스의 형식. 가능한 형식은 DEL, ASC, PC/IXF 및 CURSOR입니다.
- 입력 데이터가 테이블에 추가되는지 또는 테이블의 기존 데이터를 교체하는지 여부
- 메시지 파일 이름(유틸리티가 API db2Load를 통해 호출되는 경우)

로드 모드

- **INSERT**
  이 모드에서는 로드 시 기존 데이터를 변경하지 않고 테이블에 입력 데이터를 추가합니다.
- **REPLACE**
  이 모드에서는 로드 시 테이블의 기존 데이터를 삭제하고 테이블에 입력 데이터를 채웁니다.
- **RESTART**
  이 모드에서는 인터럽트된 로드가 다시 시작됩니다. 대부분의 경우 로드는 실패한 단계부터 다시 시작됩니다. 해당 단계가 로드 단계인 경우 로드는 마지막으로 성공한 일관성 지점부터 다시 시작됩니다.
- **TERMINATE**
  이 모드에서는 실패한 로드 조작이 롤백됩니다.

다음은 사용자가 지정할 수 있는 옵션입니다.

- 로드 유틸리티가 리모트로 연결된 클라이언트에서 호출된 경우 로드할 데이터는 클라이언트에 있습니다. CLIENT 옵션을 지정해도 XML 및 LOB 데이터는 항상 서버에서 읽습니다.
- 데이터를 로드하는 데 사용할 방법: 컬럼 위치, 컬럼 이름 또는 상대적 컬럼 위치.
- 유틸리티에서 일관성 지점을 설정하는 빈도.
- 데이터를 삽입할 테이블 컬럼의 이름.
- 로드 조작을 진행하면서 사전에 존재하는 데이터를 조회할 수 있는지 여부.
- 로드 조작에서 다른 유틸리티 또는 응용프로그램이 테이블 사용을 완료할 때까지 기다리는지 또는 진행하기 전에 다른 응용프로그램을 강제로 해제하는지 여부.
- 인덱스를 빌드할 대체 시스템 임시 테이블 스페이스.
- LOB가 저장된 입력 파일의 경로 및 이름.

  주: 로드 유틸리티는 COMPACT LOB 옵션을 고려하지 않습니다.

- 메시지 파일 이름. 로드 조작 중 해당 조작에 관한 오류, 경고 및 정보 메시지를 포함하도록 해당 메시지 파일 작성을 지정할 수 있습니다. MESSAGES 매개변수를 사용하여 이러한 파일의 이름을 지정합니다.

  주:

  1. 조작을 완료할 때까지 메시지 파일의 컨텐츠는 볼 수만 있습니다. 로드 조작을 실행하는 중 로드 메시지를 보려는 경우 LOAD QUERY 명령을 사용할 수 있습니다.
  2. 메시지 파일의 각 메시지는 새 라인에서 시작되며 DB2 메시지 검색 기능에서 제공하는 정보를 포함합니다.
- 로드할 컬럼 값에서 소수점을 내포하는지 여부.
- 테이블을 로드한 후 유틸리티에서 사용 가능한 여유 공간 *크기*를 수정해야 하는지 여부.
- 로드 프로세스 중 통계를 수집할 것인지 여부. 이 옵션은 REPLACE 모드에서 로드 조작을 실행 중인 경우에만 지원됩니다. 테이블에 정의된 프로파일에 따라 통계가 수집됩니다. 프로파일은 LOAD 명령을 실행하기 전에 RUNSTATS 명령에 의해 작성되어야 합니다. 프로파일이 없고 로드 조작에서 프로파일에 따라 통계를 수집하도록 지시한 경우 로드에 실패하고 오류 메시지가 리턴됩니다.

  데이터가 테이블에 추가되면 통계는 수집되지 않습니다. 추가된 테이블에서 현재 통계를 수집하려면 로드 프로세스를 완료한 후 RUNSTATS 유틸리티를 호출합니다. 고유 인덱스를 포함한 테이블에서 통계를 수집하고 삭제 단계에서 중복 키가 삭제되면 삭제된 레코드를 고려하도록 통계가 갱신되지 않습니다. 중복 레코드의 상당한 수가 예상되는 경우 로드 조작 중 통계를 수집하지 않습니다. 대신, 로드 프로세스를 완료한 후 RUNSTATS 유틸리티를 호출합니다.
- 변경 사항 사본을 보존하는지 여부. 데이터베이스의 롤 포워드 복구를 실행할 수 있도록 수행합니다. 이 옵션은 데이터베이스에 대한 롤 포워드 복구가 불가능한 경우 (즉, 데이터베이스 구성 매개변수 *logarchmeth1* 및 *logarchmeth2*가 OFF로 설정된 경우) 지원되지 않습니다. 복사하지 않고 롤 포워드 복구가 사용 가능한 경우 로드 조작이 완료되면 테이블 스페이스는 백업 보류 상태로 남습니다.

  완전히 복구 가능한 데이터베이스에서는 로깅이 필요합니다. 로드 유틸리티는 데이터 로드와 관련된 로깅을 거의 대부분 제거합니다. 로깅 대신 테이블의 로드된 부분을 복사하는 옵션이 있습니다. 실패 후 데이터베이스 복구를 허용하는 데이터베이스 환경인 경우 다음 중 하나를 수행할 수 있습니다.

  – 테이블의 로드된 부분을 복사하도록 명시적으로 요청합니다.

  – 로드 조작 완료 직후 테이블이 있는 테이블 스페이스를 백업합니다.

  데이터베이스 구성 매개변수 *logindexbuild*가 설정된 경우 및 COPY YES 복구 가능 옵션 및 INCREMENTAL 인덱스 옵션을 사용하여 로드 조작이 호출된 경우 로

드 시 모든 인덱스 수정 사항이 로그됩니다. 이 옵션을 사용하면 이 로드에 대한 로그 레코드를 통해 롤 포워드할 때 인덱스도 복구한다는 장점이 있습니다. 로드에서 REBUILD 인덱스 모드를 사용하지 않는 한, 정상적으로 인덱스는 복구되지 않습니다.

이미 데이터가 있는 테이블을 로드하고 데이터베이스가 복구 불가능한 경우 오류로부터 복구할 수 있도록 로드 유틸리티를 호출하기 전에 로드할 테이블의 테이블 스페이스 또는 데이터베이스의 백업 사본이 있는지 확인합니다.

복구 가능한 데이터베이스에서 여러 로드 조작 시퀀스를 수행하려는 경우 각 로드 조작을 복구 불가능하게 지정하고 로드 시퀀스 마지막에 백업을 수행하면 COPY YES 옵션을 사용하여 각 로드 조작을 호출할 때보다 조작 시퀀스가 더 빨라집니다. NONRECOVERABLE 옵션을 사용하여 로드 트랜잭션을 복구 불가능한 것으로 표시하고 후속 롤 포워드 조작으로 복구할 수 없도록 지정할 수 있습니다. 롤 포워드 유틸리티는 트랜잭션을 건너뛰고 데이터를 로드할 테이블을 "유효하지 않은" 항목으로 표시합니다. 또한 이 유틸리티는 해당 테이블에 대한 후속 트랜잭션을 무시합니다. 롤 포워드 조작을 완료한 후 이러한 테이블은 삭제만 가능합니다(그림 4 참조). 이 옵션을 사용하면 로드 조작 후에 테이블 스페이스가 백업 보류 상태가 되지 않으며 로드 조작 중에 로드된 데이터의 사본을 작성할 필요가 없습니다.

그림 4. 롤 포워드 조작 중 복구 불가능한 처리

- 로드 조작 중 임시 파일을 작성할 때 사용할 완전한 경로. LOAD 명령의 TEMPFILES PATH 매개변수로 이름이 지정됩니다. 디폴트값은 데이터베이스 경로입니다. 경로는 서버 머신에 있으며 DB2 인스턴스에서 독점적으로 액세스됩니다. 따라서 이 매개변수에 지정된 경로 이름 자격은 클라이언트가 아닌 서버의 디렉토리 구조를 반영해야 하며 DB2 인스턴스 소유자는 경로에 대한 읽기 및 쓰기 권한이 있어야 합니다.

로드를 사용하는 데 필요한 특권 및 권한

사용자는 특권을 사용하여 데이터베이스 자원을 작성하거나 자원에 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작과 특권을 그룹화하는 방법을 제공합니다. 이를 통해 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다. 적절히 권한이 부여된 (즉, 필수 특권 또는 권한이 있음) 오브젝트에만 액세스할 수 있습니다.

테이블로 데이터를 로드하려면 다음 중 하나가 있어야 합니다.

- DATAACCESS 권한
- 데이터베이스에 대한 LOAD 또는 DBADM 권한.
  - 테이블에 대한 INSERT 특권(로드 유틸리티가 INSERT 모드, TERMINATE 모드(이전 로드 삽입 조작을 종료하기 위함) 또는 RESTART 모드(이전 로드 삽입 조작을 재시작하기 위함)로 호출되는 경우).
  - 테이블에 대한 INSERT 및 DELETE 특권(로드 유틸리티가 REPLACE 모드, TERMINATE 모드(이전 로드 교체 조작을 종료하기 위함) 또는 RESTART 모드(이전 로드 교체 조작을 재시작하기 위함)로 호출되는 경우).
  - 예외 테이블에 대한 INSERT 특권(이러한 테이블이 로드 조작 중에 사용되는 경우).
  - LOAD에서 카탈로그 테이블을 쿼리하는 경우 SYSCAT.TABLES에 대한 SELECT 특권이 필요합니다.

모든 로드 프로세스(및 일반적으로 모든 DB2 서버 프로세스)는 인스턴스 소유자가 소유하고 이러한 모든 프로세스에서는 필수 파일에 액세스하기 위해 인스턴스 소유자의 ID를 사용하며 인스턴스 소유자는 입력 데이터 파일에 대한 읽기 액세스 권한이 있어야 합니다. 이러한 입력 데이터 파일은 명령 호출자에 상관없이 인스턴스 소유자가 읽을 수 있어야 합니다.

REPLACE 옵션이 지정된 경우 세션 권한 부여 ID에 테이블을 삭제할 권한이 있어야 합니다.

DB2가 Windows 서비스로 실행되는 Windows 및 Windows.NET 운영 체제에서 네트워크 드라이브에 있는 파일에서 데이터를 로드하는 경우 이러한 파일에 대한 읽기 액세스 권한이 있는 사용자 어카운트로 DB2 서비스를 실행하도록 구성해야 합니다.

주:

- 보호 컬럼이 있는 테이블로 데이터를 로드하려면 테이블의 모든 보호 컬럼에 대한 쓰기 액세스를 허용하는 LBAC 증명서가 세션 권한 부여 ID에 있어야 합니다.
- 보호 행이 있는 테이블로 데이터를 로드하려면 테이블을 보호하는 보안 규정에 포함된 쓰기 액세스에 대한 보안 레이블이 세션 권한 부여 ID에 부여되어야 합니다.

LOAD 권한

테이블에 대한 INSERT 특권 및 데이터베이스 레벨의 LOAD 권한이 있는 사용자는 LOAD 명령을 사용하여 테이블에 데이터를 로드할 수 있습니다.

주: DATAACCESS 권한을 보유하면 LOAD 명령에 대한 전체 액세스 권한이 사용자에게 부여됩니다.

테이블에 대한 INSERT 특권 및 데이터베이스 레벨의 LOAD 권한이 있는 사용자는 이전 로드 조작이 데이터를 삽입하는 로드인 경우 LOAD RESTART 또는 LOAD TERMINATE를 수행할 수 있습니다.

테이블에 관한 INSERT 및 DELETE 특권과 데이터베이스 레벨의 LOAD 권한이 있는 사용자는 LOAD REPLACE 명령을 사용할 수 있습니다.

이전 로드 조작이 로드 바꾸기였으면 해당 사용자에게 DELETE 특권도 부여해야만 사용자가 LOAD RESTART 또는 LOAD TERMINATE를 수행할 수 있습니다.

로드 조작의 일부로서 예외 테이블이 사용되면 사용자는 예외 테이블에 대한 INSERT 특권이 있어야 합니다.

이 권한이 있는 사용자는 QUIESCE TABLESPACES FOR TABLE, RUNSTATS 및 LIST TABLESPACES 명령을 수행할 수 있습니다.

데이터 로드

로드 유틸리티에서는 새로 작성한 테이블 또는 이미 데이터를 포함하는 테이블로 많은 양의 데이터를 효과적으로 이동할 수 있습니다.

로드 유틸리티를 호출하기 전에 데이터를 로드할 데이터베이스에 연결하거나 내재적으로 연결할 수 있어야 합니다. 유틸리티는 COMMIT문을 실행하므로 로드 유틸리티를 호출하기 전에 COMMIT 또는 ROLLBACK문을 실행하여 모든 트랜잭션을 완료하고 잠금을 해제해야 합니다. 데이터는 다차원적으로 클러스터된(MDC) 테이블, 파티션된 테이블 또는 anyorder 파일 유형 수정자를 사용하는 경우를 제외하고 입력 파일에 나타나는 시퀀스로 로드됩니다. 특정 시퀀스를 원하는 경우 로드 조작을 시도하기 전에 데이터를 정렬합니다. 클러스터링이 필요한 경우 로드 전에 클러스터링 인덱스에서 데이터를 정렬해야 합니다. 다차원적으로 클러스터된(MDC) 테이블로 데이터를 로드하는 경우 로드 조작 전에 정렬하지 않아도 되며 MDC 테이블 정의에 따라 데이터가 클러스터됩니다. 파티션된 테이블로 데이터를 로드하는 경우 로드 조작 전에 정렬하지 않아도 되며 테이블 정의에 따라 데이터가 파티션됩니다.

다음은 로드 유틸리티에 적용되는 몇 가지 제한사항입니다. 즉, 이 목록은 모든 제한사항을 포함하지는 않습니다.

- 별칭으로의 데이터 로드는 지원되지 않습니다.

- 유형이 지정된 테이블 또는 구조화된 유형 컬럼이 있는 테이블로 데이터를 로드하는 작업은 지원되지 않습니다.
- 선언된 임시 테이블 및 작성된 임시 테이블로 데이터를 로드하는 작업은 지원되지 않습니다.
- 서버 측에서는 XML 데이터만 읽을 수 있습니다. 클라이언트에서 XML 파일을 읽으려는 경우 임포트 유틸리티를 사용합니다.
- 백업 보류 상태인 테이블 스페이스에서는 테이블을 작성하거나 삭제할 수 없습니다.
- DB2 Connect 또는 DB2 버전 2 이전의 서버 레벨을 통해 액세스된 데이터베이스로 데이터를 로드할 수 없습니다. 현재 버전에서만 사용 가능한 옵션은 이전 릴리스의 서버에서 사용할 수 없습니다.
- LOAD REPLACE 조작 중 오류가 발생하면 테이블의 원래 데이터가 유실됩니다. 로드 조작을 재시작할 수 있도록 입력 데이터의 사본을 보유합니다.
- 새로 로드된 행에서 트리거가 활성화되지 않습니다. 로드 유틸리티는 트리거와 연관된 비즈니스 규칙을 시행하지 않습니다.
- 암호화된 데이터 로드는 지원되지 않습니다.

다음은 파티션된 테이블로 로드하는 경우 로드 유틸리티에 적용되는 몇 가지 제한사항입니다. 즉, 이 목록은 모든 제한사항을 포함하지는 않습니다.

- 파티셔닝 에이전트 수가 1보다 큰 경우 일관성 지점은 지원되지 않습니다.
- 데이터 파티션의 서브세트로 데이터를 로드하면서 나머지 데이터 파티션은 완전히 온라인 상태로 남는 조건은 지원되지 않습니다.
- 로드 조작 또는 무결성 설정 보류 조작에서 사용하는 예외 테이블은 파티션할 수 없습니다.
- 로드 유틸리티를 삽입 모드 또는 재시작 모드로 실행하고 로드 목표 테이블에 접속 해제된 종속이 있는 경우 고유 인덱스는 재빌드할 수 없습니다.

로드 유틸리티는 명령행 처리기(CLP), 제어 센터의 로드 마법사 또는 API db2Load를 통해 호출될 수 있습니다.

로드 마법사 사용

1. 제어 센터에서 테이블 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 테이블 폴더를 누르십시오. 기존 테이블은 창 오른쪽의 분할 영역(컨텐츠 영역)에 표시됩니다.
3. 컨텐츠 영역에서 원하는 테이블을 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 로드를 선택하십시오. 로드 마법사가 열립니다.
4. 마법사의 각 페이지에서 데이터를 성공적으로 로드하는 데 필요한 정보를 지정하십시오.

로드 마법사에 대한 세부사항 정보는 온라인 도움말 기능을 통해 제공됩니다.

CLP를 사용하여 LOAD 명령 실행

다음은 CLP를 통해 LOAD 명령을 실행하는 예입니다.

```
db2 load from stafftab.ixf of ixf messages staff.msgs
   insert into userid.staff copy yes use tsm data buffer 4000
```

이 예에서,

- 경고 또는 오류 메시지는 staff.msgs 파일에 배치됩니다.
- 변경 사항 사본은 TSM(Tivoli® Storage Manager)에 저장됩니다.
- 로드 조작 중 버퍼 스페이스 4,000개 페이지가 사용됩니다.

다음은 CLP를 통해 LOAD 명령을 실행하는 또 다른 예입니다.

```
db2 load from stafftab.ixf of ixf messages staff.msgs
   tempfiles path /u/myuser replace into staff
```

이 예에서,

- 테이블 데이터가 교체됩니다.
- TEMPFILES PATH 매개변수를 사용하여 임시 파일이 작성되는 서버 경로로 /u/myuser를 지정합니다.

주: 이 예에서는 로드 입력 파일에 대해 상대 경로 이름을 사용합니다. 상대 경로 이름은 데이터베이스와 동일한 데이터베이스 파티션에 있는 클라이언트의 호출에만 사용할 수 있습니다. 완전한 경로 이름을 사용하는 것이 좋습니다.

로드 유틸리티를 호출한 후에는 LIST UTILITIES 명령을 사용하여 로드 조작의 진행을 모니터할 수 있습니다. INSERT 모드, REPLACE 모드 또는 RESTART 모드에서 로드 조작을 수행한 경우 자세한 진행 모니터링 지원이 사용 가능합니다. SHOW DETAILS 옵션과 함께 LIST UTILITIES 명령을 실행하여 현재 로드 단계에 대한 자세한 정보를 확인합니다. TERMINATE 모드로 로드 조작을 수행한 경우 세부사항은 사용 불가능합니다. LIST UTILITIES 명령은 현재 로드 종료 유틸리티가 실행 중이라는 점만 표시합니다.

로드 조작은 고유 제한조건, 파티션된 테이블에 대한 범위 제한조건, 생성된 컬럼 및 LBAC 보안 규칙을 유지합니다. 기타 모든 제한조건의 경우 로드 조작을 시작하면 테이블이 무결성 설정 보류 상태가 됩니다. 로드 조작을 완료한 후 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY문을 사용해야 합니다.

XML 데이터 로드

볼륨이 큰 XML 데이터를 효율적으로 테이블로 이동시키기 위해 로드 유틸리티를 사용할 수 있습니다.

XML 테이블 컬럼에 데이터를 로드하는 경우 XML FROM 옵션을 사용하여 입력 XML 데이터 파일의 경로를 지정할 수 있습니다. 예를 들어, XML 파일 /home/user/xmlpath/xmlfile1.xml에서 데이터를 로드하기 위해 다음 명령을 사용할 수 있습니다.

```
LOAD FROM data1.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

컬럼 식별자가 있는 ASCII 입력 파일 data1.del에는 로드할 XML 데이터의 위치를 설명하는 XML 데이터 지정자(XDS)가 포함되어 있습니다. 예를 들어, 다음 XDS는 길이가 456바이트인 xmldata.ext 파일에서 오프셋이 124바이트인 XML 문서를 나타냅니다.

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' />
```

선언된 커서를 사용하여 XML 데이터를 로드하는 것이 지원됩니다. 다음 예에서는 커서를 선언하고 해당 커서 및 LOAD 명령을 사용하여 CUSTOMERS 테이블에서 LEVEL1_CUSTOMERS 테이블로 데이터를 추가합니다.

```
DECLARE cursor_income_level1 CURSOR FOR
   SELECT * FROM customers
   WHERE XMLEXISTS('$DOC/customer[income_level=1]');

LOAD FROM cursor_income_level1 OF CURSOR INSERT INTO level1_customers;
```

XML 데이터를 XML 컬럼으로 로드할 수 있도록 LOAD 명령의 ANYORDER 파일 유형 수정자가 지원됩니다.

파티션된 데이터베이스 환경에서 XML 데이터 로드

데이터베이스 파티션 간에 분배된 테이블의 경우, XML 데이터를 XML 데이터 파일에서 테이블로 동시에 로드할 수 있습니다. 파일에서 테이블로 XML 데이터를 로드하는 경우 로드를 수행 중인 모든 데이터베이스 파티션에서 XML 데이터 파일에 대한 읽기 액세스가 가능해야 합니다.

스키마에 대해 삽입된 문서 유효성 확인

XMLVALIDATE 옵션을 사용하면 XML 문서 로드 시 XML 스키마와 대조하여 문서의 유효성을 확인할 수 있습니다. 다음 예에서는 컬럼 식별자가 있는 ASCII 입력 파일 data2.del의 XDS를 통해 식별된 스키마에 대해 수신 XML 문서의 유효성을 확인합니다.

```
   LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
        USING XDS INSERT INTO USER.T2
```

이 경우 XDS에는 XML 스키마의 완전한 SQL ID가 포함된 SCH 속성 "S1.SCHEMA_A"가 들어 있어 유효성 확인에 사용할 수 있습니다.

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' SCH='S1.SCHEMA_A' />
```

구문 분석 옵션 지정

XMLPARSE 옵션을 사용하여 로드된 XML 문서의 공백을 유지할지 또는 제거할지 여부를 지정할 수 있습니다. 다음 예에서는 SQL ID ″S2.SCHEMA_A″를 사용하여 스키마에 대해 로드된 모든 XML 문서의 유효성을 검증하고 해당 문서를 구문 분석하며 공백은 유지됩니다.

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
   WHITESPACE XMLVALIDATE USING SCHEMA S2.SCHEMA_A INSERT INTO USER.T1
```

파티션된 테이블에 대한 로드 고려사항

다음 일반 제한사항을 제외하고 목표 테이블이 파티션된 경우 모든 기존 로드 기능이 지원됩니다.

- 파티셔닝 에이전트 수가 1보다 큰 경우 일관성 지점은 지원되지 않습니다.
- 데이터 파티션의 서브세트로 데이터를 로드하면서 나머지 데이터 파티션은 완전히 온라인 상태로 남는 조건은 지원되지 않습니다.
- 로드 조작에서 사용하는 예외 테이블은 파티션할 수 없습니다.
- 목표 테이블에 XML 컬럼이 포함된 경우 예외 테이블을 지정할 수 없습니다.
- 로드 유틸리티를 삽입 모드 또는 재시작 모드로 실행하고 로드 목표 테이블에 접속 해제된 종속이 있는 경우 고유 인덱스는 재빌드할 수 없습니다.
- MDC 테이블 로드와 비슷하게 파티션된 테이블을 로드할 때 입력 데이터 레코드의 정확한 순서는 보존되지 않습니다. 순서는 셀 또는 데이터 파티션에서만 유지됩니다.
- 각 데이터베이스 파티션에서 다중 포맷터를 활용하는 로드 조작은 입력 레코드의 대략적인 순서만 보존합니다. 각 데이터베이스 파티션에서 단일 포맷터를 실행하면 셀 또는 테이블 파티셔닝 키로 입력 레코드를 그룹화합니다. 각 데이터베이스 파티션에서 단일 포맷터를 실행하려면 명시적으로 1의 CPU_PARALLELISM을 요청합니다.

일반 로드 동작

로드 유틸리티는 올바른 데이터 파티션에 데이터 레코드를 삽입합니다. 스플리터(splitter)와 같은 외부 유틸리티를 사용하여 로드 전에 입력 데이터를 파티션하는 경우 관련 요구사항은 없습니다.

로드 유틸리티는 접속 해제 또는 접속된 데이터 파티션에 액세스하지 않습니다. 데이터는 표시되는 데이터 파티션에만 삽입됩니다. 표시되는 데이터 파티션은 접속 또는 접속 해제된 상태가 아닙니다. 또한 로드 교체 조작은 접속 해제 또는 접속된 데이터 파티션을 절단하지 않습니다. 로드 유틸리티는 카탈로그 시스템 테이블에서 잠금을 획득하므로 로드 유틸리티는 커미트되지 않은 ALTER TABLE 트랜잭션을 기다립니다. 이러한 트랜잭션은 카탈로그 테이블의 관련 행에서 배타적 잠금을 획득합니다. 로드 조작을 계속 진행하려면 배타적 잠금을 종료해야 합니다. 즉, 로드 조작을 실행하는 동안 커미트되지 않은 ALTER TABLE ...ATTACH, DETACH 또는 ADD PARTITION 트랜잭션이 있을 수 있습니다. 접속 또는 접속 해제된 데이터 파티션이 목표로 지정된 입력

소스 레코드가 거부되고 하나가 지정되면 예외 테이블에서 검색될 수 있습니다. 목표 테이블 데이터 파티션의 일부가 접속 또는 접속 해제된 상태임을 나타내는 정보 메시지가 메시지 파일에 작성됩니다. 목표 테이블에 대응하는 관련 카탈로그 테이블 행을 잠그면 로드 유틸리티를 실행하는 동안 ALTER TABLE ...ATTACH, DETACH 또는 ADD PARTITION 조작을 실행하여 목표 테이블의 파티셔닝을 변경하지 못합니다.

유효하지 않은 행 처리

로드 유틸리티가 가시적 데이터 파티션에 속하지 않는 레코드를 발견하면 해당 레코드는 거부되고 로드 유틸리티는 처리를 계속합니다. 제한조건 위반 범위 때문에 거부된 레코드 수는 명시적으로 표시되지 않지만 거부된 레코드의 전체 수에 포함됩니다. 범위 위반으로 레코드가 거부된 경우 행 경고 수는 늘어나지 않습니다. 범위 위반을 발견했음을 나타내는 단일 메시지(SQL0327N)가 로드 유틸리티 메시지 파일에 작성되지만 레코드당 메시지는 로그되지 않습니다. 또한 목표 테이블의 모든 컬럼 이외에도 예외 테이블은 특별 행에서 나타나는 위반 유형을 설명하는 컬럼을 포함합니다. 파티션할 수 없는 데이터를 포함하여 유효하지 않은 데이터를 포함하는 행은 덤프 파일에 작성됩니다.

예외 테이블 삽입은 자원 소모가 많으므로 예외 테이블에 삽입할 제한조건 위반을 제어할 수 있습니다. 예를 들어 로드 유틸리티의 디폴트 동작은 범위 제한조건 또는 고유 제한조건 위반으로 거부된 행을 삽입하는 것입니다. 그렇지 않고 유효한 경우 예외 테이블로 삽입됩니다. 각각 FOR EXCEPTION절에서 NORANGEEXC 또는 NOUNIQUEEXC를 지정하여 이 동작을 끌 수 있습니다. 이러한 제한조건 위반을 예외 테이블에 삽입하지 않도록 하거나 예외 테이블을 지정하지 않는 경우 범위 제한조건 또는 고유 제한조건을 위반하는 행에 대한 정보는 유실됩니다.

실행기록 파일

목표 테이블이 파티션된 경우 해당하는 실행기록 파일 항목은 목표 테이블에 포함된 테이블 스페이스 목록을 포함하지 않습니다. 다른 조작 단계 ID('T' 대신 'R')는 파티션된 테이블에서 로드 조작을 실행함을 나타냅니다.

로드 조작 종료

로드 교체를 완전히 종료하면 모든 가시적 데이터 파티션이 절단되고 로드 삽입을 종료하면 모든 가시적 데이터 파티션을 로드 전의 길이로 절단합니다. 인덱스는 로드 복사 단계에서 실패한 ALLOW READ ACCESS 로드 조작 종료 중 무효화됩니다. 또한 인덱스를 처리하는 ALLOW NO ACCESS 로드 조작을 종료하는 경우에도 인덱스가 무효화됩니다. 인덱스 모드가 재빌드이거나 인덱스가 불일치 상태로 남게 되는 증분 유지보수 중에 키가 삽입되었기 때문입니다. 다중 목표로 데이터를 로드하는 경우 로드 복구 조작에 영향을 주지 않습니다. 단, 로드 단계 중 수행된 일관성 지점부터 로드 조작을 재시작하는 기능은 예외입니다. 이 경우 목표 테이블이 파티션되었으면 SAVECOUNT 로드 옵션이 무시됩니다. 이 동작은 MDC 목표 테이블로 데이터를 로드할 때와 일관됩니다.

생성된 컬럼

생성된 컬럼이 파티셔닝, 차원 또는 분산 키에 있는 경우 generatedoverride 파일 유형 수정자는 무시되고 generatedignore 파일 유형 수정자가 지정된 경우와 같이 로드 유틸리티에서 값을 생성합니다. 이 경우 올바르지 않은 생성된 컬럼 값을 로드하면 잘못된 물리적 위치(예: 잘못된 데이터 파티션, MDC 블록 또는 데이터베이스 파티션)에 레코드를 배치할 수 있습니다. 예를 들어 레코드가 잘못된 데이터 파티션에 배치되면 세트 무결성은 이 레코드를 다른 물리적 위치로 이동해야 합니다. 온라인 세트 무결성 조작 중 수행할 수 없습니다.

데이터 사용 가능성

현재 ALLOW READ ACCESS 로드 알고리즘은 파티션된 테이블로 확장됩니다. ALLOW READ ACCESS 로드 조작에서는 동시 판독기에서 로딩 및 비로딩 데이터 파티션 모두를 포함하여 전체 테이블에 액세스할 수 있습니다.

데이터 파티션 상태

로드에 성공한 후 가시적 데이터 파티션은 특정 조건에서 무결성 설정 보류 또는 읽기 액세스 전용 테이블 상태로 변경될 수 있습니다. 로드 조작을 유지보수할 수 없는 테이블에 대한 제한조건이 있으면 데이터 파티션은 이 상태로 배치될 수 있습니다. 이러한 제한조건으로는 점검 제한조건 및 접속 해제된 구체화된 쿼리 테이블을 포함할 수 있습니다. 실패한 로드 조작은 보이는 모든 데이터 파티션을 로드 보류 테이블 상태로 설정합니다.

오류 분리

데이터 파티션 레벨의 오류 분리는 지원되지 않습니다. 오류 분리는 오류가 발생하지 않는 데이터 파티션에서 로드를 계속하고 오류로 발생한 데이터 파티션을 중지하는 작업을 의미합니다. 오류는 다른 데이터베이스 파티션 사이에서 분리될 수 있지만 로드 유틸리티는 가시적 데이터 파티션의 서브세트에서 트랜잭션을 커미트하고 나머지 가시적 데이터 파티션을 롤백할 수 없습니다.

기타 고려사항

- 인덱스가 유효하지 않은 항목으로 표시되면 증분 인덱싱은 지원되지 않습니다. 재빌드해야 하거나 접속 해제된 종속 항목에서 SET INTEGRITY문으로 유효성을 확인해야 하는 경우 인덱스는 유효하지 않은 항목으로 간주됩니다.
- 범위로 파티션되거나 해시로 분산되거나 차원으로 구성되는 알고리즘 조합을 사용하여 파티션된 테이블로 로드하는 작업이 지원됩니다.
- 로드로 영향을 받는 오브젝트 및 테이블 스페이스 ID의 목록을 포함하는 로그 레코드의 경우 이러한 레코드(LOAD START 및 COMMIT(PENDING LIST))의 *크기*는 상당히 커질 수 있으므로 다른 응용프로그램에서 사용 가능한 사용 중인 로그 스페이스 *크기*가 줄어듭니다.

- 테이블이 파티션 및 분산된 경우 파티션된 데이터베이스 로드가 모든 데이터베이스 파티션에 영향을 주지 않을 수도 있습니다. 출력 데이터베이스 파티션의 오브젝트만 변경됩니다.
- 로드 조작 중 파티션된 테이블의 메모리 소모는 테이블 수만큼 증가합니다. 단, 전체 메모리 요구사항의 작은 비율만 데이터 파티션 수에 비례하므로 전체 증가분이 선형으로 나타나지는 않습니다.

LBAC 보호 데이터 로드 고려사항

보호된 행을 포함하는 테이블로 로드하려면 레이블 기반 액세스 제어(LBAC) 증명서가 필요합니다. 또한 유효한 보안 레이블 또는 목표 테이블과 연관된 현재 보안 규정에 대해 유효한 레이블로 변환할 수 있는 보안 레이블을 제공해야 합니다.

유효한 LBAC 증명서가 없으면 로드에 실패하고 오류(SQLSTATE 42512)가 리턴됩니다. 입력 데이터에 보안 레이블이 없거나 해당 보안 레이블이 내부 2진 형식이 아니면 여러 파일 유형 수정자를 사용하여 로드를 진행할 수 있습니다.

보호된 행을 포함하는 테이블로 데이터를 로드하는 경우 목표 테이블에 데이터 유형이 DB2SECURITYLABEL인 컬럼이 하나 있습니다. 데이터의 입력 행에 해당 컬럼 값이 없으면 load 명령에 usedefaults 파일 유형 수정자가 지정되지 않는 한, 해당 행은 거부됩니다. 이 경우 테이블을 보호하는 보안 규정의 쓰기 액세스에 대해 보유한 보안 레이블이 사용됩니다. 쓰기 액세스에 대한 보안 레이블을 보유하지 않은 경우 행이 거부되고 다음 행으로 처리가 계속됩니다.

보호된 행을 포함하는 테이블로 데이터를 로드하고 입력 데이터에 데이터 유형이 DB2SECURITYLABEL인 컬럼의 값이 포함된 경우 해당 테이블로 데이터를 삽입할 때와 동일한 규칙이 적용됩니다. 로드할 행을 보호하는 보안 레이블(데이터 파일의 해당 행에 있는 항목)이 쓰기 권한이 있는 항목인 경우 해당 보안 레이블을 사용하여 행을 보호합니다. 즉, 데이터 유형이 DB2SECURITYLABEL인 컬럼에 작성됩니다. 해당 보안 레이블로 보호되는 행에 대한 쓰기 권한이 없는 경우 소스 테이블을 보호하는 보안 규정이 작성된 방식에 따라 상황이 달라집니다.

- 규정을 작성한 CREATE SECURITY POLICY문에 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션이 포함된 경우 행이 거부됩니다.
- CREATE SECURITY POLICY문이 옵션을 포함하지 않거나 대신 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 포함하는 경우 해당 행에 대한 데이터 파일의 보안 레이블은 무시되고 쓰기 액세스에 대해 보유한 보안 레이블을 사용하여 해당 행을 보호합니다. 이 경우 오류나 경고는 발행되지 않습니다. 쓰기 액세스에 대한 보안 레이블을 보유하지 않은 경우 행이 거부되고 다음 행으로 처리가 계속됩니다.

분리문자 고려사항

데이터 유형이 DB2SECURITYLABEL인 컬럼으로 데이터를 로드하는 경우 데이터 파일의 값은 디폴트로 해당 보안 레이블의 내부 표현을 구성하는 실제 바이트로 가정합니다. 그러나 일부 행 데이터는 LOAD 명령에서 행 분리로 잘못 해석될 수 있는 개행 문자를 포함할 수 있습니다. 이 문제점이 발생한 경우 delprioritychar 파일 유형 수정자를 사용하여 문자 분리문자가 행 분리문자보다 우선되도록 해야 합니다. delprioritychar을 사용하는 경우 문자 분리문자에 포함된 레코드 또는 컬럼 분리문자는 분리문자로 인식되지 않습니다. delprioritychar 파일 유형 수정자는 개행 문자를 포함하는 값이 없는 경우에도 안전하게 사용할 수 있습니다. 그러나 로드 속도가 약간 느려질 수 있습니다.

로드할 데이터가 ASC 형식인 경우 로드된 보안 레이블 및 보안 레이블 이름에 뒤 공백이 포함되지 않도록 하기 위해 추가 단계를 수행해야 할 수도 있습니다. ASCII 형식에서는 분리문자로 컬럼 위치를 사용하므로 가변 길이 필드로 로드할 때 이 문제가 나타날 수 있습니다. striptblanks 파일 유형 수정자를 사용하여 뒤 공백을 절단합니다.

비표준 보안 레이블 값

또한 보안 레이블 값이 보안 레이블에서 구성요소 값을 포함하는 문자열(예: *S:(ALPHA,BETA)*)인 데이터 파일을 로드할 수도 있습니다. 이를 수행하려면 seclabelchar 파일 유형 수정자를 사용해야 합니다. seclabelchar을 사용하면 데이터 유형이 DB2SECURITYLABEL인 컬럼의 값은 보안 레이블에 대한 문자열 형식으로 보안 레이블을 포함하는 문자열 상수로 가정합니다. 문자열이 적절한 형식이 아니면 행은 삽입되지 않고 경고(SQLSTATE 01H53)가 리턴됩니다. 문자열이 테이블을 보호하는 보안 규정에 포함된 유효한 보안 레이블을 표시하지 않으면 행은 삽입되지 않고 경고(SQLSTATE 01H53)가 리턴됩니다.

보안 레이블 컬럼 값이 보안 레이블 이름인 데이터 파일을 로드할 수도 있습니다. 이러한 유형의 파일을 로드하려면 seclabelname 파일 유형 수정자를 사용해야 합니다. seclabelname을 사용하는 경우 데이터 유형이 DB2SECURITYLABEL인 컬럼의 모든 값은 기존 보안 레이블의 이름을 포함하는 문자열 상수로 가정합니다. 테이블을 보호하는 보안 규정의 표시 이름에 보안 레이블이 없으면 행은 로드되지 않고 경고(SQLSTATE 01H53)가 리턴됩니다.

거부된 행

로드 중 거부된 행은 행이 거부된 이유에 따라 덤프 파일 또는 예외 테이블(LOAD 명령에 지정된 경우)로 보내집니다. 구문 분석 오류로 거부된 행은 덤프 파일에 보내집니다. 보안 규정을 위반한 행은 예외 테이블로 보내집니다.

주: 목표 테이블에 XML 컬럼이 포함된 경우 예외 테이블을 지정할 수 없습니다.

예

모든 예에서 입력 데이터 파일 myfile.del은 DEL 형식입니다. 모두 REPS 테이블(다음 명령문으로 작성됨)로 데이터를 로드합니다.

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

이 예에서 입력 파일은 디폴트 형식의 보안 레이블을 포함한다고 가정합니다.

```
db2 load from myfile.del of del modified by delprioritychar insert into reps
```

이 예에서 입력 파일은 보안 레이블 문자열 형식의 보안 레이블을 포함한다고 가정합니다.

```
db2 load from myfile.del of del modified by seclabelchar insert into reps
```

이 예에서 입력 파일은 보안 레이블 컬럼의 보안 레이블 이름을 포함한다고 가정합니다.

```
db2 load from myfile.del of del modified by seclabelname insert into reps
```

ID 컬럼 로드 고려사항

로드 유틸리티는 입력 데이터에 ID 컬럼 값이 있는지에 상관없이 ID 컬럼을 포함하는 테이블로 데이터를 로드하는 데 사용할 수 있습니다.

ID 관련 파일 유형 수정자가 사용되지 않으면 유틸리티는 다음 규칙에 따라 작동합니다.

- ID 컬럼이 GENERATED ALWAYS인 경우 입력 파일에서 ID 컬럼에 해당하는 행의 값이 누락되거나 명시적으로 널(NULL) 값이 지정되면 항상 테이블 행에 대한 ID 값이 생성됩니다. ID 컬럼에 대해 널(NULL)이 아닌 값이 지정되면 행은 거부됩니다(SQL3550W).
- ID 컬럼이 GENERATED BY DEFAULT인 경우 사용자 제공 값이 제공되면 로드 유틸리티는 이 값을 사용합니다. 데이터가 누락되거나 명시적으로 널(NULL)이 지정되면 값이 생성됩니다.

로드 유틸리티는 ID 컬럼의 데이터 유형(즉, SMALLINT, INT, BIGINT 또는 DECIMAL) 값에 대해 정상적으로 수행되는 작업 이외에도 사용자 제공 ID 컬럼 값에 대해 추가 유효성 확인 작업을 수행하지 않습니다. 중복 값은 보고되지 않습니다.

대부분의 경우 로드 유틸리티는 이 행이 데이터 파일에 나타나는 동일한 순서로 ID 컬럼 값이 행에 지정된다고 보장할 수 없습니다. ID 컬럼 값 지정은 로드 유틸리티에서 병렬로 관리되므로 이 값은 임의의 순서로 지정됩니다. 다음은 예외 조건입니다.

- 단일 파티션 데이터베이스에서 CPU_PARALLELISM이 1로 설정되면 병렬로 처리되지 않습니다. 이 경우 ID 컬럼 값은 데이터 파일 매개변수에 행이 나타나는 동일한 순서로 내재적으로 지정됩니다.
- 다중 파티션 데이터베이스에서 ID 컬럼이 분산 키에 있고 단일 파티셔닝 에이전트가 있는 경우(즉, 다중 파티셔닝 에이전트 또는 anyorder 파일 유형 수정자를 지정하지 않는 경우) 데이터 파일에 행이 나타나는 동일한 순서로 ID 컬럼 값이 지정됩니다.

테이블의 ID 컬럼이 파티셔닝 키에 있고 identityoverride 수정자가 지정되지 않은 경우 파티션된 데이터베이스로 테이블을 로드하면 SAVECOUNT 옵션을 지정할 수 없습니다. 파티셔닝 키에 ID 컬럼이 있고 ID 값이 생성되는 경우 하나 이상의 데이터베이스 파티션에서 로드 단계부터 로드를 재시작하면 로드 단계의 시작부터 전체 로드를 재시작해야 합니다. 즉, 일관성 지점은 나타날 수 없습니다.

주: RESTART 로드 조작은 다음의 모든 기준이 만족된 경우 허용되지 않습니다.

- 로드할 테이블은 파티션된 데이터베이스 환경에 있고 분산 키에 있거나 분산 키에 포함된 생성된 컬럼에서 참조되는 하나 이상의 ID 컬럼을 포함합니다.
- identityoverride 수정자가 지정되지 않았습니다.
- 실패한 이전 로드 조작에 로드 단계 이후 실패한 데이터베이스 파티션 로드가 포함됩니다.

TERMINATE 또는 REPLACE 로드 조작을 대신 실행해야 합니다.

ID 컬럼을 포함하는 테이블로 데이터를 간단히 로드할 수 있는 서로 배타적인 세 가지 방법이 있습니다. identitymissing, identityignore 및 identityoverride 파일 유형 수정자가 그 방법입니다.

ID 컬럼을 포함하지 않고 데이터 로드

identitymissing 수정자를 사용하면 입력 데이터 파일에 ID 컬럼 값(널(NULL) 값도 아님)이 없는 경우 ID 컬럼을 포함한 테이블을 보다 편리하게 로드할 수 있습니다. 예를 들어 다음 SQL문으로 정의된 테이블을 고려하십시오.

```
create table table1 (c1 varchar(30),
                     c2 int generated by default as identity,
                     c3 decimal(7,2),
                     c4 char(1))
```

ID 컬럼이 없는 테이블에서 익스포트된 데이터를 파일(load.del)에서 TABLE1로 로드하려는 경우 다음 예를 참조하십시오.

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

이 파일을 로드하는 한 가지 방법은 다음과 같이 LOAD 명령을 통해 로드할 컬럼을 명시적으로 나열하는 것입니다.

```
db2 load from load.del of del replace into table1 (c1, c3, c4)
```

그러나 컬럼이 많은 테이블의 경우 구문이 복잡해지고 오류가 발생할 수 있습니다. 파일을 로드하는 또 다른 방법은 다음과 같이 identitymissing 파일 유형 수정자를 사용하는 것입니다.

```
db2 load from load.del of del modified by identitymissing
   replace into table1
```

이 명령으로 데이터 파일의 세 컬럼이 TABLE1의 c1, c3 및 c4로 로드됩니다. c2에서는 각 행에 대한 값이 생성됩니다.

ID 컬럼을 포함하여 데이터 로드

identityignore 수정자는 입력 데이터 파일에 ID 컬럼 데이터가 있어도 데이터를 무시하고 각 행에 대한 ID 값을 생성하도록 로드 유틸리티에 표시합니다. 예를 들어 위에서 정의한 대로 다음 데이터를 포함하는 데이터 파일(load.del)에서 TABLE1을 로드하려고 합니다.

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

사용자 제공 값 1, 2 및 3이 ID 컬럼에서 사용되지 않으면 다음 LOAD 명령을 실행할 수 있습니다.

```
db2 load from load.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

다시 한 번 강조하지만 이 방법은 테이블에 컬럼이 많은 경우 복잡해지며 오류가 발생할 수 있습니다. identityignore 수정자는 다음과 같이 구문을 단순화합니다.

```
db2 load from load.del of del modified by identityignore
   replace into table1
```

사용자가 제공한 값을 포함하여 데이터 로드

identityoverride 수정자는 GENERATED ALWAYS ID 컬럼을 포함하는 테이블로 사용자 제공 값을 로드하는 데 사용됩니다. 다른 데이터베이스 시스템에서 데이터를 이주하고 테이블을 GENERATED ALWAYS로 정의해야 하는 경우 또는 ROLLFORWARD DATABASE 명령에서 DROPPED TABLE RECOVERY 옵션을 사용하여 복구된 데이터에서 테이블을 로드하는 경우 매우 유용합니다. 이 수정자를 사용하면 ID 컬럼에 데이터가 없거나 널(NULL) 데이터인 행은 거부됩니다(SQL3116W). 또한 이 수정자를 사용하는 경우 GENERATED ALWAYS 컬럼의 고유성 등록 정보를 위반할 수 있다는 점에 주의해야 합니다. 이 경우 후속 INSERT 또는 REPLACE 로드 조작 이전에 TERMINATE 로드 조작을 수행합니다.

생성된 컬럼 로드 고려사항

입력 데이터에 생성된 컬럼 값이 있는지에 상관없이 생성된 컬럼(비ID)을 포함하는 테이블로 데이터를 로드할 수 있습니다. 로드 유틸리티는 컬럼 값을 생성합니다.

생성된 컬럼 관련 파일 유형 수정자가 사용되지 않으면 로드 유틸리티는 다음 규칙에 따라 작동합니다.

- 데이터 파일에서 컬럼에 해당하는 행의 값이 누락되거나 널(NULL) 값이 지정되면 생성된 컬럼에 대한 값이 생성됩니다. 생성된 컬럼에 대해 널(NULL)이 아닌 값이 제공되면 행은 거부됩니다(SQL3550W).
- 널(NULL) 입력이 가능하지 않은 생성된 컬럼에 널(NULL) 값이 작성되면 데이터의 전체 행이 거부됩니다(SQL0407N). 예를 들어 널(NULL) 값을 허용하지 않는 생성된 컬럼이 데이터 파일에서 널(NULL) 값을 포함하는 두 개의 테이블 컬럼 합으로 정의된 경우 이러한 상황이 나타날 수 있습니다.

생성된 컬럼을 포함하는 테이블로 데이터를 간단히 로드할 수 있는 서로 배타적인 세 가지 방법이 있습니다. generatedmissing, generatedignore 및 generatedoverride 파일 유형 수정자가 그 방법입니다.

생성된 컬럼을 포함하지 않고 데이터 로드

generatedmissing 수정자를 사용하면 입력 데이터 파일에 테이블의 모든 생성된 컬럼 값(널(NULL) 값도 아님)이 없는 경우 생성된 컬럼을 포함하는 테이블을 편리하게 로드할 수 있습니다. 예를 들어 다음 SQL문으로 정의된 테이블을 고려하십시오.

```
CREATE TABLE table1 (c1 INT,
                     c2 INT,
                     g1 INT GENERATED ALWAYS AS (c1 + c2),
                     g2 INT GENERATED ALWAYS AS (2 * c1),
                     c3 CHAR(1))
```

생성된 컬럼이 없는 테이블에서 익스포트된 데이터를 파일(load.del)에서 TABLE1로 로드하려는 경우 다음 예를 참조하십시오.

```
1, 5, J
2, 6, K
3, 7, I
```

이 파일을 로드하는 한 가지 방법은 다음과 같이 LOAD 명령을 통해 로드할 컬럼을 명시적으로 나열하는 것입니다.

```
DB2 LOAD FROM load.del of del REPLACE INTO table1 (c1, c2, c3)
```

그러나 컬럼이 많은 테이블의 경우 구문이 복잡해지고 오류가 발생할 수 있습니다. 파일을 로드하는 또 다른 방법은 다음과 같이 generatedmissing 파일 유형 수정자를 사용하는 것입니다.

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedmissing
   REPLACE INTO table1
```

이 명령으로 데이터 파일의 세 컬럼이 TABLE1의 c1, c2 및 c3으로 로드됩니다. generatedmissing 수정자 때문에 TABLE1의 컬럼 g1 및 g2 값이 자동으로 생성되며 데이터 파일 컬럼에는 맵핑되지 않습니다.

생성된 컬럼을 포함하여 데이터 로드

generatedignore 수정자는 입력 데이터 파일에 목표 테이블의 모든 생성된 컬럼 데이터가 있어도 데이터를 무시하고 각 생성된 컬럼으로 계산된 값을 로드하도록 로드 유틸리티에 표시합니다. 예를 들어 위에서 정의한 대로 다음 데이터를 포함하는 데이터 파일(load.del)에서 TABLE1을 로드하려고 합니다.

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

생성된 컬럼 관련 파일 유형 수정자가 사용되지 않는 경우 사용자 제공, 널(NULL)이 아닌 값 10, 11 및 12(g1의 경우), 15, 16 및 17(g2의 경우)로 행이 거부됩니다(SQL3550W). 이를 방지하기 위해 다음 LOAD 명령을 실행할 수 있습니다.

```
DB2 LOAD FROM load.del of del method P(1, 2, 5)
   REPLACE INTO table1 (c1, c2, c3)
```

다시 한 번 강조하지만 이 방법은 테이블에 컬럼이 많은 경우 복잡해지며 오류가 발생할 수 있습니다. generatedignore 수정자는 다음과 같이 구문을 단순화합니다.

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedignore
   REPLACE INTO table1
```

이 명령으로 데이터 파일의 컬럼이 TABLE1의 c1(데이터 1, 2, 3 포함), c2(데이터 5, 6, 7 포함) 및 c3(데이터 J, K, I 포함)으로 로드됩니다. generatedignore 수정자 때문에 TABLE1의 컬럼 g1 및 g2에 대한 값이 자동으로 생성되며 데이터 파일 컬럼(10, 11, 12 및 15, 16, 17)은 무시됩니다.

사용자가 제공한 값을 포함하여 데이터 로드

generatedoverride 수정자는 생성된 컬럼을 포함하는 테이블로 사용자 제공 값을 로드하는 데 사용됩니다. 다른 데이터베이스 시스템에서 데이터를 이주하거나 ROLLFORWARD DATABASE 명령의 RECOVER DROPPED TABLE 옵션을 사용하여 복구된 데이터에서 테이블을 로드하는 경우 매우 유용합니다. 이 수정자를 사용하면 널(NULL) 값을 허용하지 않는 생성된 컬럼에 데이터가 없거나 널(NULL) 데이터인 행은 거부됩니다(SQL3116W).

이 수정자를 사용하면 로드 조작 이후 테이블은 무결성 설정 보류 상태가 됩니다. 사용자 제공 값을 검증하지 않고 무결성 설정 보류 상태에서 테이블을 해제시키려면 다음 명령을 실행하십시오.

```
SET INTEGRITY FOR table-name GENERATED COLUMN IMMEDIATE
  UNCHECKED
```

사용자 제공 값을 강제로 검증하고 무결성 설정 보류 상태에서 테이블을 해제시키려면 다음 명령을 실행하십시오.

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED
```

생성된 컬럼이 파티셔닝, 차원 또는 분산 키에 있는 경우 generatedoverride 수정자는 무시되고 generatedignore 수정자가 지정된 경우와 같이 로드 유틸리티에서 값을 생성합니다. 사용자 제공 생성된 컬럼 값이 생성된 컬럼 정의와 충돌하는 경우(잘못된 데이터 파티션, MDC 블록 또는 데이터베이스 파티션과 같이 잘못된 물리적 위치에 결과로 생성된 레코드를 배치하는 경우)를 방지하기 위해 이를 수행합니다.

주: 로드에서 생성된 컬럼 값을 지원하지 않는 한 가지 경우가 있습니다. 생성된 컬럼 표현식 중 하나에 FENCED인 사용자 정의 함수(UDF)가 포함된 경우가 그렇습니다. 이러한 테이블로 로드하려고 하면 로드 조작에 실패합니다. 그러나 generatedoverride 파일 유형 수정자를 사용하여 이러한 유형의 생성된 컬럼에 대한 고유한 값을 제공할 수 있습니다.

버전 8 이상 서버에서 버전 7 이하 클라이언트를 사용하는 경우 고려사항
버전 7 이하 클라이언트와 버전 8 이상 서버 사이에서 로드 조작을 시작하는 경우 로드 유틸리티는 생성된 컬럼을 포함하는 테이블을 무결성 설정 보류 상태로 설정합니다. 버전 7 이하 클라이언트를 사용하여 생성된 컬럼을 포함하는 테이블로 데이터를 로드하므로 테이블이 무결성 설정 보류 상태로 설정되면 다음 명령문을 실행하여 해당 상태를 제거하고 강제로 값을 생성합니다.

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED FORCE GENERATED;
```

CURSOR 파일 유형을 사용하여 데이터 이동

LOAD 명령을 사용하는 경우 CURSOR 파일 유형을 지정하면 중간 익스포트 파일을 작성하지 않고도 목표 테이블로 SQL 쿼리 결과를 직접 로드할 수 있습니다.

또한 SQL 쿼리에서 별칭을 참조하거나 DECLARE CURSOR문에서 DATABASE 옵션을 사용하거나 API 인터페이스 사용 시 sqlu_remotefetch_entry 미디어 항목을 사용하여 다른 데이터베이스에서 데이터를 로드할 수 있습니다.

CURSOR 파일 유형을 사용하여 데이터를 이동하는 세 가지 접근 방법이 있습니다. 첫 번째 접근 방법은 명령행 처리기(CLP)를, 두 번째 방법은 API를, 세 번째 방법은 ADMIN_CMD 프로시저를 사용하는 것입니다. CLP 및 ADMIN_CMD 프로시저의 가장 큰 다른 점은 다음 표에서 설명합니다.

표 26. CLP 및 ADMIN_CMD 프로시저 간 다른 점입니다.

다른 점	CLP	ADMIN_CMD 프로시저
구문	쿼리 명령문 및 cursor에서 사용하는 소스 데이터베이스는 DECLARE CURSOR문을 사용하는 LOAD 명령 외부에 정의됩니다.	쿼리 명령문 및 cursor에서 사용하는 소스 데이터베이스는 Load from(DATABASE database-alias query-statement)을 사용하는 LOAD 명령 내부에 정의됩니다.
다른 데이터베이스에 액세스하는 경우 사용자 권한 부여	데이터가 현재 연결된 데이터베이스와는 다른 데이터베이스에 있는 경우 DATABASE 키워드를 DECLARE CURSOR문에서 사용해야 합니다. 동일한 명령문에서 사용자 ID 및 암호도 지정할 수 있습니다. 사용자 ID 및 암호가 DECLARE CURSOR문에 지정되지 않으면 소스 데이터베이스 연결에 명시적으로 지정된 사용자 ID 및 암호를 사용하여 목표 데이터베이스에 액세스합니다.	데이터가 현재 연결된 데이터베이스와는 다른 데이터베이스에 있는 경우 쿼리 명령문 이전에 DATABASE 키워드를 LOAD 명령에서 사용해야 합니다. 목표 데이터베이스에 액세스하려면 소스 데이터베이스 연결에 명시적으로 지정된 사용자 ID 및 암호가 필요합니다. 소스 데이터베이스에 대한 사용자 ID 또는 암호를 지정할 수 없습니다. 따라서 목표 데이터베이스에 연결할 때 사용자 ID 및 암호가 지정되지 않은 경우 또는 지정된 사용자 ID 및 암호를 소스 데이터베이스에서 인증하는 데 사용할 수 없는 경우 ADMIN_CMD 프로시저를 사용하여 로드를 수행할 수 없습니다.

CLP에서 LOAD FROM CURSOR 조작을 실행하려면 먼저 SQL 쿼리에서 cursor를 선언해야 합니다. 이를 선언하면 선언된 cursor 이름을 *cursorname*으로, CURSOR를 파일 유형으로 사용하는 LOAD 명령을 실행할 수 있습니다.

예를 들면, 다음과 같습니다.

1. 다음 정의를 통해 소스 및 목표 테이블이 동일한 데이터베이스에 있다고 가정합니다.

   테이블 ABC.TABLE1에는 다음 3개 컬럼이 있습니다.

   - ONE INT
   - TWO CHAR(10)
   - THREE DATE

   테이블 ABC.TABLE2에는 다음 3개 컬럼이 있습니다.

   - ONE VARCHAR
   - TWO INT
   - THREE DATE

다음 CLP 명령을 실행하면 ABC.TABLE1에서 ABC.TABLE2로 모든 데이터가 로드됩니다.

```
DECLARE mycurs CURSOR FOR SELECT TWO, ONE, THREE FROM abc.table1
   LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

주: 위 예에서는 CLP를 통해 SQL 쿼리를 로드하는 방법을 보여줍니다. 그러나 SQL 쿼리에서 로드하는 작업은 db2Load API를 통해서도 수행할 수 있습니다. sqlu_media_list 구조의 *piSourceList*를 정의하여 *sqlu_statement_entry* 구조 및 SQLU_SQL_STMT 미디어 유형을 사용하고 *piFileType* 값을 SQL_CURSOR로 정의합니다.

2. 다음 정의를 통해 소스 및 목표 테이블이 다른 데이터베이스에 있다고 가정합니다.

데이터베이스 'dbsource'의 테이블 ABC.TABLE1에는 다음 3개 컬럼이 있습니다.

- ONE INT
- TWO CHAR(10)
- THREE DATE

데이터베이스 'dbsource'의 테이블 ABC.TABLE2에는 다음 3개 컬럼이 있습니다.

- ONE VARCHAR
- TWO INT
- THREE DATE

카탈로그된 데이터 소스('dsdbsource') 및 페더레이션이 사용 가능한 경우 다음 예에서 설명한 대로, 소스 데이터베이스에서 별칭을 선언하고 이 별칭으로 cursor를 선언하고 FROM CURSOR 옵션을 사용하여 LOAD 명령을 호출할 수 있습니다.

```
CREATE NICKNAME myschema1.table1 FOR dsdbsource.abc.table1
DECLARE mycurs CURSOR FOR SELECT TWO,ONE,THREE FROM myschema1.table1
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

또는 다음 예에서 보여주는 대로, DECLARE CURSOR문의 DATABASE 옵션을 사용할 수 있습니다.

```
DECLARE mycurs CURSOR DATABASE dbsource USER dsciaraf USING mypasswd
FOR SELECT TWO,ONE,THREE FROM abc.table1
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

DECLARE CURSOR문의 DATABASE 옵션(로드 API 사용 시 remotefetch 미디어 유형이라고도 함)을 사용하면 별칭을 사용한 접근 방법에서 몇 가지 이점이 있습니다.

성능

remotefetch 미디어 유형을 사용하는 데이터 페치는 로드 조작에 긴밀히 통합되어 있습니다. 별칭 접근 방법과 비교하여 레코드를 페치하는 소수의 전의 계층이 있습니다.

또한 소스 및 목표 테이블이 다중 파티션 데이터베이스에서 균등하게 분산된 경우 로드 유틸리티는 데이터 페치를 병렬 처리하여 성능을 더욱 향상시킬 수 있습니다.

용이성

페더레이션을 사용 가능하게 하거나 리모트 데이터 소스를 정의하거나 별칭을 선언할 필요가 없습니다. DATABASE 옵션(필요한 경우 USER 및 USING 옵션 포함)을 지정하기만 하면 됩니다.

이 방법은 카탈로그된 데이터베이스에서 사용될 수 있는 반면, 별칭 사용은 단순히 카탈로그될 수 없는 다양한 데이터 소스를 페치하는 강력한 기능을 제공합니다.

이 remotefetch 기능을 지원하기 위해 로드 유틸리티는 SOURCEUSEREXIT 기능을 지원하는 인프라스트럭처를 사용합니다. 로드 유틸리티는 응용프로그램으로 실행하는 프로세스를 생성하여 소스 데이터베이스에 대한 연결을 관리하고 페치를 수행합니다. 이 응용프로그램은 고유한 트랜잭션과 연관되며 로드 유틸리티를 실행하는 트랜잭션과는 연관되지 않습니다.

주:

1. 이전 예에서는 DECLARE CURSOR문의 DATABASE 옵션을 사용하여 카탈로그된 데이터베이스에서 CLP를 통해 SQL 쿼리로부터 로드하는 방법을 보여줍니다. 그러나 카탈로그된 데이터베이스에서 SQL 쿼리로부터 로드하는 작업은 db2Load API를 통해, *db2LoadStruct* 구조의 *piSourceList* 및 *piFileTypevalues*를 정의하여 각각 sqlu_remotefetch_entry 미디어 항목 및 SQLU_REMOTEFETCH 미디어 유형을 사용함으로써 수행할 수도 있습니다.
2. 이전 예에서 설명한 대로 SQL 쿼리의 소스 컬럼 유형은 호환 가능해야 하는 경우에도 목표 컬럼 유형과 동일할 필요는 없습니다.

제한사항

DATABASE 옵션을 사용하여 정의된 cursor에서 로드하는 경우(또는 이와 동등하게 db2Load API에서 sqlu_remotefetch_entry 미디어 항목을 사용하는 경우) 다음 제한사항이 적용됩니다.

1. SOURCEUSEREXIT 옵션은 동시에 지정될 수 없습니다.
2. METHOD N 옵션은 지원되지 않습니다.
3. `usedefaults` 파일 유형 수정자는 지원되지 않습니다.

인접한 종속 스테이징 테이블 전파

로드할 테이블이 인접한 전파 속성을 포함하는 스테이징 테이블의 기본 테이블인 경우 및 로드 조작이 삽입 모드로 수행된 경우 인접한 종속 스테이징 테이블로의 후속 전파는 증분 방식으로 수행됩니다.

증분 전파를 수행하는 중 기본 테이블에서 추가된 행에 해당하는 행이 스테이징 테이블로 추가됩니다. 증분 전파는 대형 기본 테이블에서 추가된 데이터가 적은 경우에 더 빠릅니다. 또한 스테이징 테이블을 사용하여 지연된 종속 구체화된 쿼리 테이블을 새로 고치면 성능이 더 향상됩니다. 증분 전파가 허용되지 않는 경우도 있으며 이때 스테이징 테이블은 불완전한 항목으로 표시됩니다. 즉, CONST_CHECKED 컬럼의 스테이징 바이트 값이 F인 경우가 이에 해당합니다. 이 상태에서는 스테이징 테이블을 사용하여 지연된 종속 구체화된 쿼리 테이블을 새로 고칠 수 없으며 구체화된 쿼리 테이블 유지보수 프로세스에서 완전 새로 고침이 요구됩니다.

테이블이 불완전 상태이고 INCREMENTAL 옵션이 지정되지만 테이블의 증분 전파가 가능하지 않으면 오류가 리턴됩니다. 다음 중 하나를 수행한 경우 시스템은 즉시 데이터 전파를 끄고 테이블 상태를 불완전으로 설정합니다.

- 스테이징 테이블의 기본 테이블로 로드 교체 조작이 수행되거나 기본 테이블에서 마지막 무결성 검사를 수행한 후 NOT LOGGED INITIALLY WITH EMPTY TABLE 옵션이 활성화됩니다.
- 스테이징 테이블의 종속 구체화된 쿼리 테이블 또는 스테이징 테이블이 REPLACE 또는 INSERT 모드로 로드됩니다.
- 무결성 검사 중 FULL ACCESS 옵션을 사용하여 스테이징 테이블을 전파하기 전에 기본 테이블이 무결성 설정 보류 상태에서 해제됩니다.
- 스테이징 테이블의 기본 테이블에서 비증분 방식으로 무결성을 검사합니다.
- 스테이징 테이블 또는 해당 기본 테이블을 포함하는 테이블 스페이스는 특정 시점으로 롤 포워드되고 스테이징 테이블 및 해당 기본 테이블은 다른 테이블 스페이스에 있습니다.

스테이징 테이블에서 SYSCAT.TABLES 카탈로그의 CONST_CHECKED 컬럼에 W 값이 있고 SET INTEGRITY문에 NOT INCREMENTAL 옵션이 지정되지 않은 경우 스테이징 테이블로의 증분 전파가 수행되고 SYSCAT.TABLES의 CONST_CHECKED 컬럼은 U로 표시되어 시스템에서 일부 데이터는 검증하지 않음을 나타냅니다.

다음 예에서는 스테이징 테이블 G1 및 구체화된 쿼리 테이블 AST1의 기본 테이블 UT1로 로드 삽입 조작을 수행하는 경우를 보여줍니다. 이 시나리오에서 UT1의 무결성 검사 및 AST1의 새로 고침은 모두 증분 방식으로 처리됩니다.

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1,G1 IMMEDIATE CHECKED;

REFRESH TABLE AST1 INCREMENTAL;
```

인접한 종속 구체화된 쿼리 테이블 새로 고침

INSERT 옵션을 사용하여 새로 고친 인접한 구체화된 쿼리 테이블의 기본 테이블을 로드하는 경우 REFRESH IMMEDIATE로 정의된 종속 구체화된 쿼리 테이블에서 SET INTEGRITY문을 실행하면 구체화된 쿼리 테이블의 증분 새로 고침이 수행됩니다.

증분 새로 고침을 수행하는 중 기본 테이블에서 추가된 행에 해당하는 행이 갱신되어 구체화된 쿼리 테이블로 삽입됩니다. 증분 새로 고침은 대형 기본 테이블에서 추가된 데이터가 적은 경우에 더 빠릅니다. 증분 새로 고침이 허용되지 않는 경우 완전 새로 고침(즉, 구체화된 쿼리 테이블 정의 쿼리의 재적용)이 사용됩니다.

INCREMENTAL 옵션이 지정되지만 구체화된 쿼리 테이블의 증분 처리가 가능하지 않으면 다음 경우에 오류가 리턴됩니다.

- 구체화된 쿼리 테이블의 기본 테이블로 로드 교체 조작이 수행되거나 기본 테이블에서 마지막 무결성 검사를 수행한 후 NOT LOGGED INITIALLY WITH EMPTY TABLE 옵션이 활성화됩니다.
- 구체화된 쿼리 테이블이 REPLACE 또는 INSERT 모드로 로드됩니다.
- 무결성 검사 중 FULL ACCESS 옵션을 사용하여 구체화된 쿼리 테이블을 새로 고치기 전에 기본 테이블이 무결성 설정 보류 상태에서 해제됩니다.
- 구체화된 쿼리 테이블의 기본 테이블에서 비증분 방식으로 무결성을 검사합니다.
- 업그레이드 전에 구체화된 쿼리 테이블이 무결성 설정 보류 상태입니다.
- 구체화된 쿼리 테이블 또는 해당 기본 테이블을 포함하는 테이블 스페이스는 특정 시점으로 롤 포워드되고 구체화된 쿼리 테이블 및 해당 기본 테이블은 다른 테이블 스페이스에 있습니다.

구체화된 쿼리 테이블에서 SYSCAT.TABLES 카탈로그의 CONST_CHECKED 컬럼에 하나 이상의 W 값이 있고 SET INTEGRITY문에 NOT INCREMENTAL 옵션이 지정되지 않은 경우 테이블은 점차적으로 새로 고쳐지고 SYSCAT.TABLES의 CONST_CHECKED 컬럼은 U로 표시되어 시스템에서 일부 데이터는 검증하지 않음을 나타냅니다.

다음 예에서는 구체화된 쿼리 테이블 AST1의 기본 테이블 UT1로 로드 삽입 조작을 수행하는 경우를 보여줍니다. UT1에서 데이터 무결성을 검사하고 UT1이 데이터 이동 없음 모드로 배치됩니다. AST1의 증분 새로 고침이 완료되면 UT1은 다시 전체 액세스 상태가 됩니다. 이 시나리오에서 UT1의 무결성 검사 및 AST1의 새로 고침은 모두 증분 방식으로 처리됩니다.

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1 IMMEDIATE CHECKED;
REFRESH TABLE AST1;
```

다차원 클러스터링 고려사항

다음 제한사항은 다차원적으로 클러스터된(MDC) 테이블로 데이터를 로드하는 경우 적용됩니다.

- LOAD 명령의 SAVECOUNT 옵션은 지원되지 않습니다.
- 이러한 테이블은 고유한 여유 공간을 관리하므로 `totalfreespace` 파일 유형 수정자는 지원되지 않습니다.
- MDC 테이블에 대해 `anyorder` 파일 유형 수정자가 필요합니다. `anyorder` 수정자 없이 MDC 테이블로 로드가 실행되면 유틸리티에서 명시적으로 사용 가능해집니다.

MDC 테이블에서 LOAD 명령을 사용하는 경우 고유 제한조건 위반은 다음과 같이 처리됩니다.

- 로드 조작 전에 테이블에 고유 키가 포함되고 중복 레코드를 테이블로 로드한 경우 원래 레코드는 남아 있으며 새 레코드는 삭제 단계 중에 삭제됩니다.
- 로드 조작 전에 테이블에 고유 키가 없고 고유 키 및 중복 레코드를 테이블로 로드한 경우 고유 키를 포함하는 레코드 중 하나만 로드되고 나머지는 삭제 단계 중에 삭제됩니다.

  주: 로드되는 레코드와 삭제되는 레코드를 판별하는 명시적인 기술은 없습니다.

성능 고려사항

MDC 테이블을 로드할 때 로드 유틸리티의 성능을 향상시키려면 *util_heap_sz* 데이터베이스 구성 매개변수 값을 늘려야 합니다. mdc-load 알고리즘의 경우 유틸리티에서 사용 가능한 추가 메모리가 있으면 성능이 더 향상됩니다. 그러면 로드 단계 중 수행된 데이터 클러스터링 동안 디스크 입출력이 줄어듭니다. LOAD 명령의 DATA BUFFER 옵션이 지정되면 값도 증가해야 합니다. LOAD 명령을 사용하여 여러 MDC 테이블을 동시에 로드하는 경우 적절히 *util_heap_sz*를 늘려야 합니다.

모든 MDC 테이블에는 블록 인덱스가 있으므로 MDC 로드 조작은 항상 빌드 단계를 포함합니다.

로드 단계 중 블록 맵 유지보수를 위한 추가 로깅이 수행됩니다. 할당된 Extent당 약 2개의 추가 로그 레코드가 있습니다. 양호한 성능을 유지하려면 *logbufsz* 데이터베이스 구성 매개변수를 이를 고려한 값으로 설정해야 합니다.

인덱스를 포함하는 시스템 임시 테이블은 MDC 테이블로 데이터를 로드하는 데 사용됩니다. 테이블 *크기*는 로드된 구별 셀의 수에 비례합니다. 테이블에서 각 행의 *크기*는 MDC 차원 키의 *크기*에 비례합니다. 로드 조작 중 이 테이블의 처리로 인한 디스크 입출력을 최소화하려면 임시 테이블 스페이스의 버퍼 풀이 충분히 커야 합니다.

사용자 정의된 응용프로그램(User Exit)을 사용하여 데이터 이동

로드 SOURCEUSEREXIT 옵션에서는 로드 유틸리티가 사용자 정의 스크립트 또는 실행 파일(여기서 *User Exit*로 참조됨)을 실행할 수 있도록 하는 기능을 제공합니다.

User Exit의 목적은 로드 유틸리티에서 동시에 읽은 데이터로 하나 이상의 Named Pipes를 채우는 것입니다. 다중 파티션 데이터베이스에서 User Exit의 다중 인스턴스를 동시에 호출하여 입력 데이터를 병렬로 처리할 수 있습니다.

그림 5에서와 같이 로드 유틸리티는 하나 이상의 Named Pipes를 작성하고 사용자 정의된 실행 파일을 실행하는 프로세스를 생성합니다. 로드 유틸리티에서 동시에 데이터를 읽는 동안, User Exit에서는 Named Pipe로 데이터를 제공합니다.

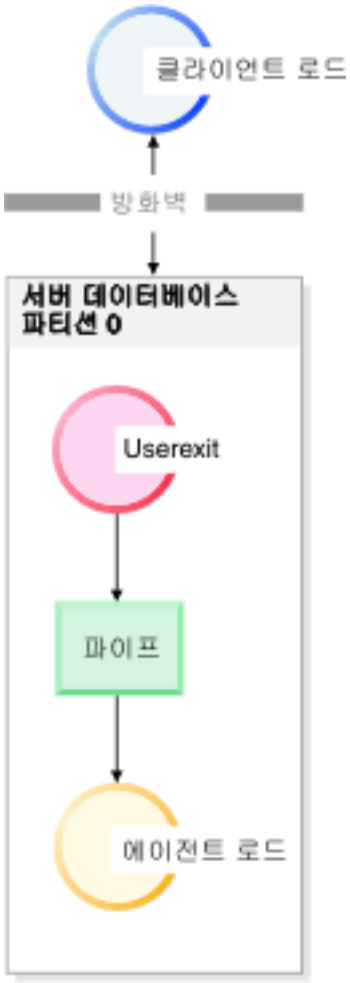


그림 5. 로드 유틸리티에서는 파이프를 통해 읽고 수신 데이터를 처리합니다.

파이프로 공급된 데이터는 파일 유형 및 파일 유형 수정자를 포함하여 지정된 로드 옵션을 반영해야 합니다. 로드 유틸리티는 지정된 데이터 파일을 직접 읽지 않습니다. 대신, User Exit를 실행할 때 지정된 데이터 파일을 User Exit에 인수로 전달합니다.

User Exit 호출

User Exit는 DB2 설치 디렉토리의 bin 서브디렉토리(종종 sqllib라고도 함)에 있어야 합니다. 로드 유틸리티는 다음 명령행 인수를 사용하여 User Exit 실행 파일을 호출합니다.

```
<base pipename> <number of source media>
<source media 1> <source media 2> ... <user exit ID>
<number of user exits> <database partition number>
```

여기서,

<base pipename >

로드 유틸리티가 작성하고 데이터를 읽는 Named Pipes의 기본 이름입니다. 유틸리티에서는 LOAD 명령에 제공된 각 소스 파일에 대해 하나의 파이프를 작성하고 이러한 각 파이프는 .xxx에 추가됩니다. 여기서 xxx는 제공된 소스 파일의 인덱스입니다. 예를 들어 LOAD 명령에 2개의 소스 파일이 제공되었고 User Exit에 전달된 <base pipename> 인수가 pipe123이면 User Exit에서 데이터를 제공해야 하는 2개의 Named Pipes는 pipe123.000 및 pipe123.001입니다. 파티션된 데이터베이스 환경에서 로드 유틸리티는 기본 파이프 이름에 데이터베이스 파티션(DBPARTITION) 번호 .yyy를 추가합니다. 이로 인해 파이프 이름은 pipe123.xxx.yyy가 됩니다.

<number of source media>

뒤에 나오는 미디어 인수 수입니다.

<source media 1> <source media 2> ...

LOAD 명령에 지정된 하나 이상의 소스 파일 목록입니다. 각 소스 파일은 큰 따옴표로 묶어 표시합니다.

<user exit ID>

PARALLELIZE 옵션이 사용 가능한 경우 유용한 특수 값입니다. 이 정수값(1에서 N, 여기서 N은 생성된 총 User Exit 수임)에서는 실행 중인 User Exit의 특정 인스턴스를 식별합니다. PARALLELIZE 옵션이 사용 가능하지 않으면 디폴트값은 1입니다.

<number of user exits>

PARALLELIZE 옵션이 사용 가능한 경우 유용한 특수 값입니다. 이 값은 동시에 실행 중인 총 User Exit 수를 표시합니다. PARALLELIZE 옵션이 사용 가능하지 않으면 디폴트값은 1입니다.

<database partition number>

PARALLELIZE 옵션이 사용 가능한 경우 유용한 특수 값입니다. 이는 User Exit가 실행되는 데이터베이스 파티션(DBPARTITION) 번호입니다. PARALLELIZE 옵션이 사용 가능하지 않으면 디폴트값은 0입니다.

추가 옵션 및 기능

다음 절에서는 추가 SOURCEUSEREXIT 기능 옵션을 설명합니다.

REDIRECT

이 옵션을 사용하면 User Exit 프로세스의 STDOUT 및 STDERR 핸들에서 데이터를 캡처하거나 STDIN 핸들로 데이터를 전달할 수 있습니다.

INPUT FROM BUFFER <buffer>

이 옵션을 사용하면 User Exit의 STDIN 입력 스트림으로 직접 정보를 전달할 수 있습니다. User Exit를 실행하는 프로세스를 생성한 후에 로드 유틸리티는 이 새 프로세스의 STDIN에 대한 파일 설명자를 획득하고 제공된 버퍼에 전달합니다. User Exit는 STDIN에서 데이터를 읽어 정보를 획득합니다. 로드 유틸리티는 단순히 STDIN을 사용하여 User Exit로 <buffer>의 컨텐츠를 보내고 컨텐츠를 해석하거나 수정하지는 않습니다. 예를 들어 User Exit가 STDIN에서 2개 값(8바이트 사용자 ID 및 8바이트 암호)을 읽도록 디자인된 경우 C로 작성된 User Exit 실행 파일은 다음 라인을 포함할 수 있습니다.

```
rc = read (stdin, pUserID, 8);
rc = read (stdin, pPasswd, 8);
```

다음 LOAD 명령에 표시된 대로 INPUT FROM BUFFER 옵션을 사용하여 이 정보를 전달할 수 있습니다.

```
LOAD FROM myfile1 OF DEL INSERT INTO table1
SOURCEUSEREXIT myuserexit1 REDIRECT INPUT FROM BUFFER myuseridmypasswd
```

주: 로드 유틸리티는 LOB 값의 최대 *크기*로 <buffer> *크기*를 제한합니다. 그러나 명령행 처리기(CLP)에서 <buffer> *크기*는 CLP 명령문의 최대 *크기*로 제한됩니다. CLP에서는 <buffer>가 기존 ASCII 문자만 포함하는 것이 좋습니다. 로드 유틸리티가 db2Load API를 사용하여 호출되거나 대신 INPUT FROM FILE 옵션이 사용되는 경우 이러한 문제는 방지할 수 있습니다.

INPUT FROM FILE <filename>

이 옵션을 사용하면 User Exit의 STDIN 입력 스트림으로 직접 클라이언트 측 파일의 컨텐츠를 전달할 수 있습니다. 이 옵션은 INPUT FROM BUFFER 옵션과 거의 비슷하지만 이 옵션을 사용하면 잠재적 CLP 제한을 회피할 수 있습니다. 파일 이름은 완전한 클라이언트 측 파일이어야 하며 LOB 값의 최대 크기를 초과할 수 없습니다.

OUTPUT TO FILE <filename>

User Exit 프로세스에서 서버 측 파일로 STDOUT 및 STDERR 스트림을 캡처할 수 있습니다. User Exit를 실행하는 프로세스를 생성한 후에 로드 유틸리티는 이 새 프로세스에서 지정된 파일 이름으로 STDOUT 및 STDERR 핸들 경로를 재지정합니다. 이 옵션은 User Exit 내 오류 및 활동을 디버그 및 로그하는 데 유용합니다. 파일 이름은 완전한 서버 측 파일이어야 합니다.

PARALLELIZE 옵션을 설정하면 User Exit당 하나의 파일이 존재하며 각 파일에 3자리 숫자 ID가 추가됩니다(예: *filename.000*).

PARALLELIZE

이 옵션은 동시에 여러 User Exit 프로세스를 호출하여 로드 유틸리티로 들어오는 데이터 처리량을 늘립니다. 이 옵션은 다중 파티션 데이터베이스에만 적용됩니다. 로드 조작 중 다중 데이터베이스 파티션에서 데이터가 분산된 경우 호출된 User Exit 인스턴스 수는 파티셔닝 에이전트 수와 동일합니다. 그렇지 않으면 로드 에이전트 수와 동일합니다.

각 User Exit에 전달된 <user exit ID>, <number of user exits> 및 <database partition number> 인수는 각각 고유 ID(1에서 N), 총 User Exit 수(N) 및 User Exit 인스턴스를 실행하는 데이터베이스 파티션(DBPARTITION) 번호를 반영합니다. 각 User Exit 프로세스에서 Named Pipe에 작성한 데이터는 다른 동시 프로세스에 의해 중복되지 않아야 합니다. User Exit 응용프로그램에서 이를 수행하는 많은 방법이 있지만 데이터가 중복되지 않도록 이 값을 유용하게 사용할 수 있습니다. 예를 들어 데이터의 각 레코드가 고유한 정수 컬럼 값을 포함하는 경우 User Exit 응용프로그램은 <user exit ID> 및 <number of user exits> 값을 사용하여 각 User Exit 인스턴스가 Named Pipe로 고유한 결과 세트를 리턴하도록 할 수 있습니다. User Exit 응용프로그램은 다음 방법으로 **MODULUS** 등록 정보를 사용할 수 있습니다.

```
i = <user exit ID>
N = <number of user exits>

foreach record
{
   if ((unique-integer MOD N) == i)
   {
     write this record to my named-pipe
   }
}
```

생성된 User Exit 프로세스 수는 데이터베이스 파티셔닝에 지정된 분산 모드에 따라 달라집니다.

1. 177 페이지의 그림 6에서와 같이 PARALLEL 없이 PARTITION_AND_LOAD(디폴트) 또는 PARTITION_ONLY가 지정된 경우 모든 사전 파티셔닝 에이전트에 대해 하나의 User Exit 프로세스가 생성됩니다.

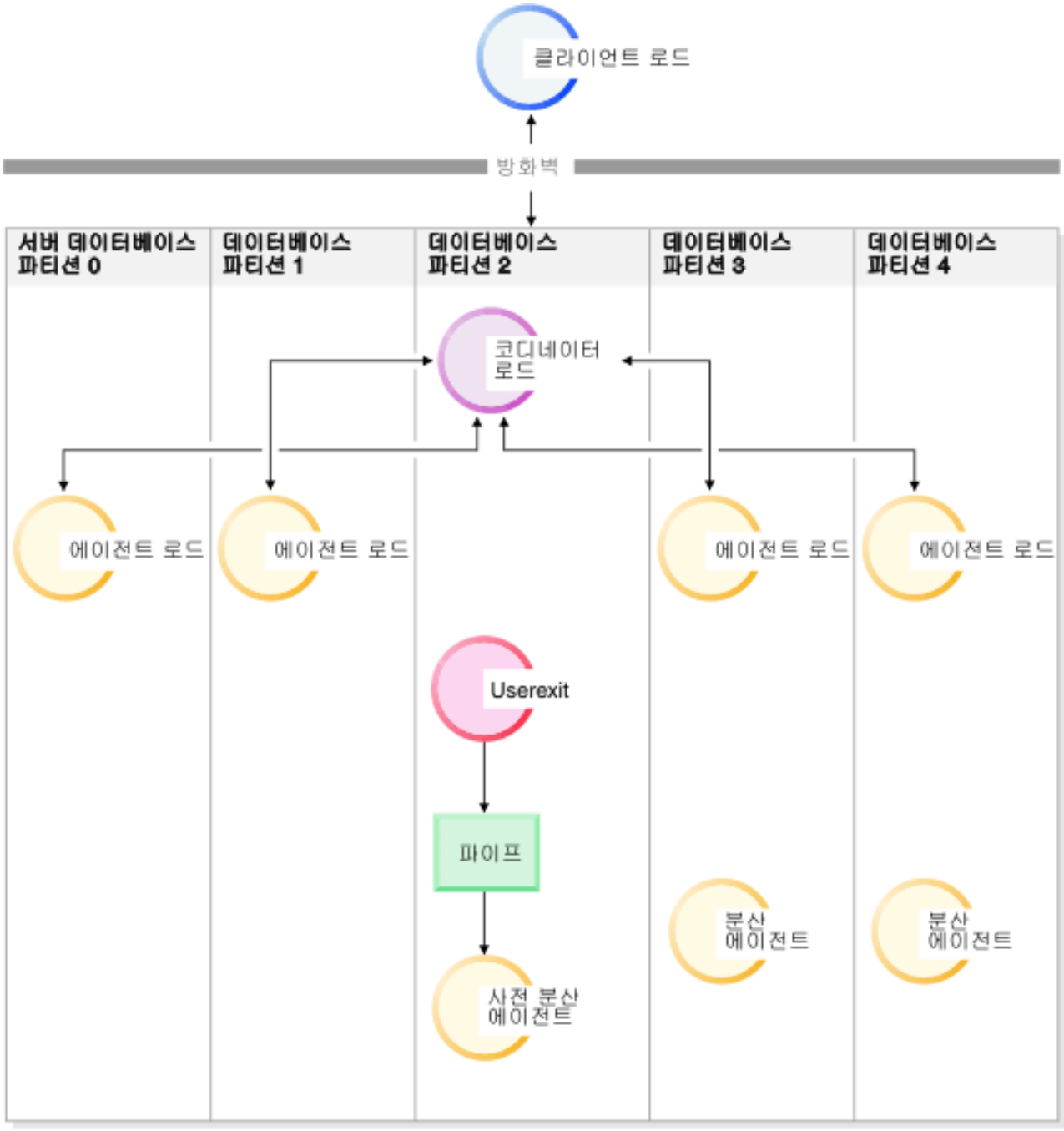


그림 6. PARALLEL 없이 PARTITION_AND_LOAD(디폴트) 또는 PARTITION_ONLY를 지정하면 다양한 태스크가 수행됩니다.

2. 178 페이지의 그림 7에서와 같이 PARALLEL과 함께 PARTITION_AND_LOAD(디폴트) 또는 PARTITION_ONLY가 지정된 경우 모든 파티셔닝 에이전트에 대해 하나의 User Exit 프로세스가 생성됩니다.

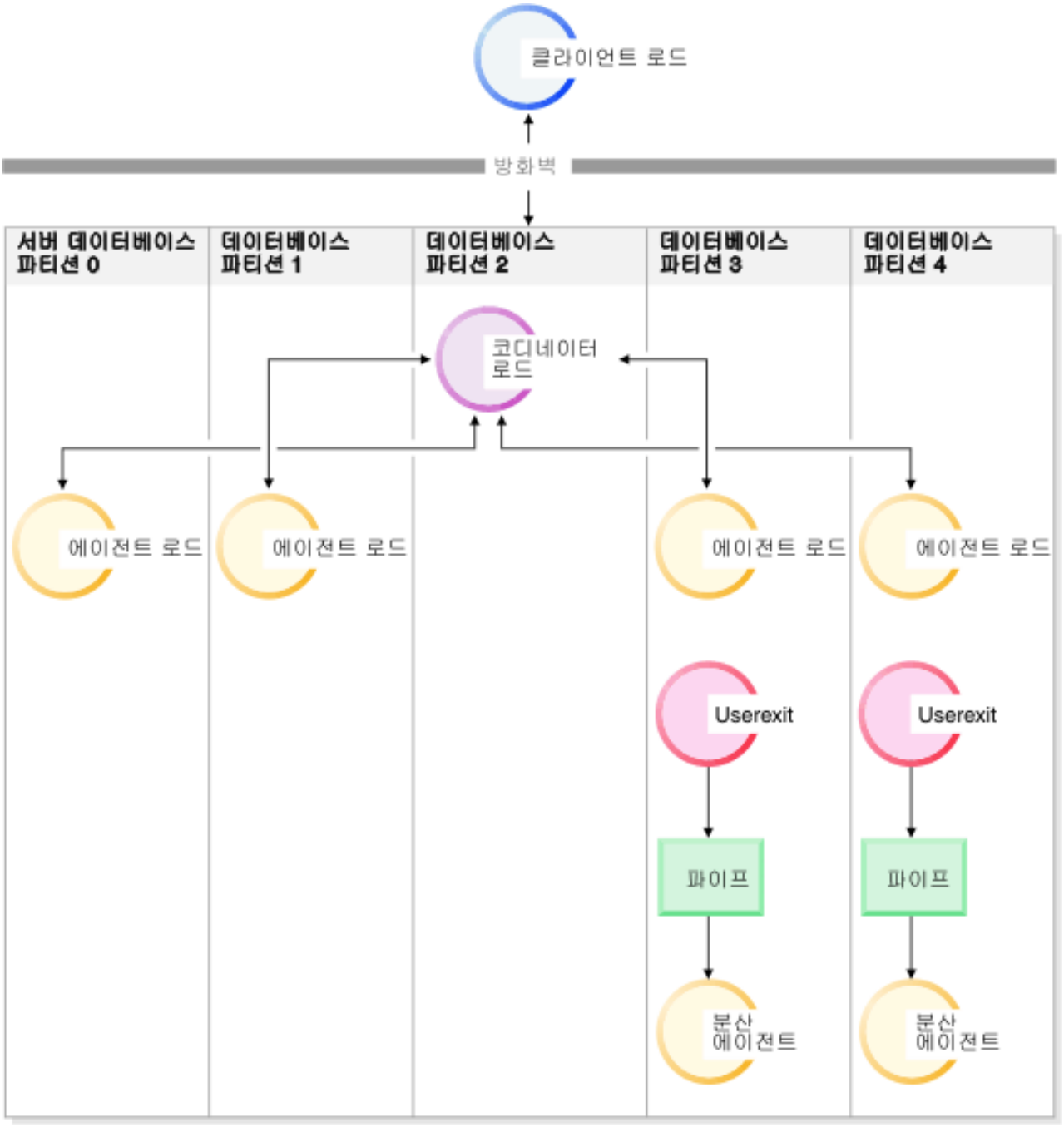


그림 7. PARALLEL과 함께 PARTITION_AND_LOAD(디폴트) 또는 PARTITION_ONLY를 지정하면 다양한 태스크가 수행됩니다.

3. 179 페이지의 그림 8에서와 같이 LOAD_ONLY 또는 LOAD_ONLY_VERIFY_PART가 지정되면 모든 로드 에이전트에 대해 하나의 User Exit 프로세스가 생성됩니다.

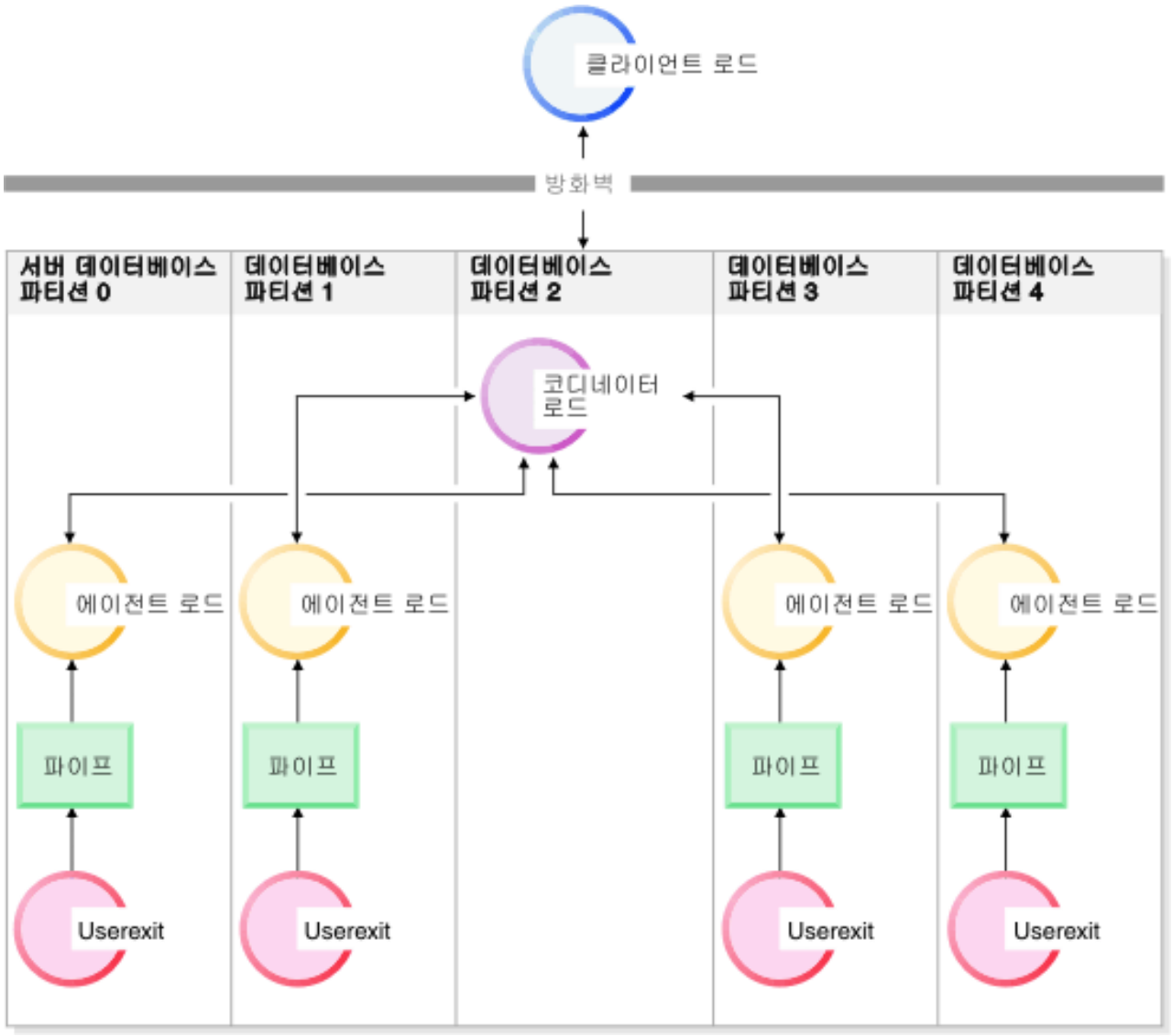


그림 8. LOAD_ONLY 또는 LOAD_ONLY_VERIFY_PART가 지정되면 다양한 태스크가 수행됩니다.

제한사항

- SOURCEUSEREXIT PARALLELIZE 옵션이 지정되지 않으면 LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART partitioned-db-cfg 모드 옵션은 지원되지 않습니다.

로드 시 추가 주의 사항

병렬 처리 및 로딩

로드 유틸리티는 SMP(Symmetric Multiprocessor) 환경과 같이 다중 프로세서 또는 다중 스토리지 디바이스를 사용하는 하드웨어 구성의 이점을 활용합니다.

로드 유틸리티를 사용하여 많은 데이터를 병렬 처리하는 여러 가지 방법이 있습니다. 한 가지 방법은 다중 스토리지 디바이스를 사용하는 것입니다. 그러면 로드 조작 중 입출력 병렬 처리가 가능합니다(그림 9 참조). 또 다른 방법은 SMP 환경에서 다중 프로세서를 사용하는 것입니다. 그러면 파티션 내 병렬 처리가 가능합니다(그림 10 참조). 두 방법을 함께 사용하여 보다 빨리 데이터를 로드할 수도 있습니다.

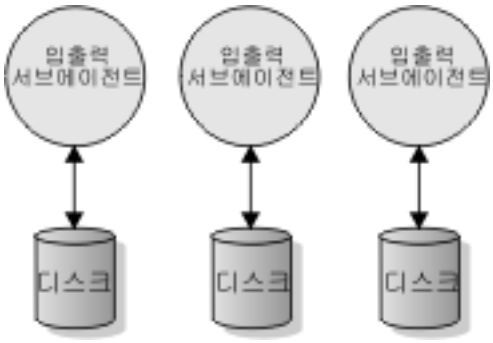


그림 9. 데이터 로드 시 입출력 병렬 처리 활용

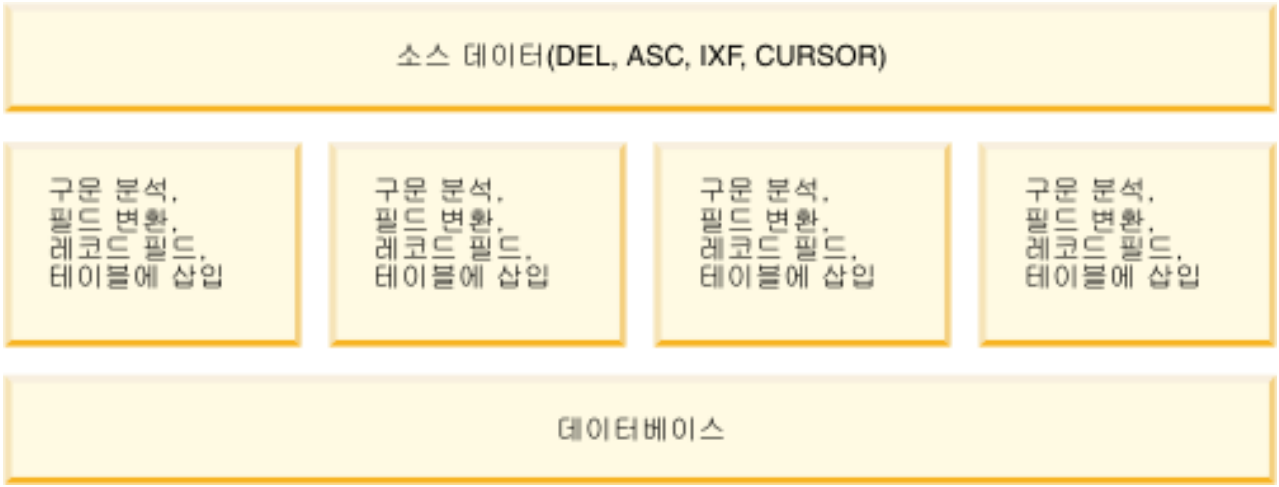


그림 10. 데이터 로드 시 파티션 내 병렬 처리 활용

로드 조작 중 인덱스 작성

로드 조작의 빌드 단계에서 인덱스가 빌드됩니다. LOAD 명령에서 지정할 수 있는 네 가지 인덱싱 모드가 있습니다.

1. REBUILD. 모든 인덱스가 재빌드됩니다.
2. INCREMENTAL. 인덱스가 새 데이터로 확장됩니다.
3. AUTOSELECT. 로드 유틸리티는 REBUILD 또는 INCREMENTAL 모드 중에서 자동으로 사용할 모드를 결정합니다. AUTOSELECT가 디폴트입니다. 로드

REPLACE 조작이 수행되면 REBUILD 인덱스 모드가 사용됩니다. 그렇지 않으면 선택된 인덱스 모드는 (새로 로드된 데이터 *크기*/*기존 데이터 크기*) 비율에 기반합니다. 비율이 상당히 크면 INCREMENTAL 인덱스 모드가 선택됩니다. 그렇지 않으면 REBUILD 인덱스 모드가 선택됩니다.

4. DEFERRED. 이 모드가 지정되면 로드 유틸리티는 인덱스 작성을 시도하지 않습니다. 인덱스가 새로 고침이 필요한 항목으로 표시되며 처음 액세스할 때 강제로 재빌드될 수 있습니다. 다음 상황에서는 DEFERRED 옵션이 허용되지 않습니다.
   - ALLOW READ ACCESS 옵션이 지정된 경우(인덱스를 유지보수하지 않으며 인덱스 스캐너에서 유효한 인덱스를 요구함)
   - 테이블에 대해 고유 인덱스가 정의된 경우
   - XML 데이터가 로드되는 경우(XML 경로 인덱스가 고유하고 XML 컬럼이 테이블에 추가될 때마다 디폴트로 작성됨)

ALLOW READ ACCESS 옵션을 지정하는 로드 조작에서는 선택한 인덱스 모드 유형에 따라 스페이스 사용 및 로깅 측면과 관련된 특별 고려사항을 고려해야 합니다. ALLOW READ ACCESS 옵션이 지정되면 재빌드하는 동안에도 로드 유틸리티가 인덱스를 쿼리에 사용할 수 있도록 유지합니다.

ALLOW READ ACCESS 모드의 로드 조작에서 INDEXING MODE INCREMENTAL 옵션을 지정하는 경우 로드 유틸리티는 인덱스 트리의 무결성을 보호하는 일부 로그 레코드를 작성합니다. 작성된 로그 레코드 수는 삽입된 키 수의 일부로, 유사한 SQL 조작에 필요한 수보다 훨씬 적습니다. INDEXING MODE INCREMENTAL 옵션이 지정된 ALLOW NO ACCESS 모드의 로드 조작에서는 정상 스페이스 할당 로그 외에 작은 로그 레코드만 작성합니다.

주: COPY YES를 지정하지 않고 *logindexrebuild* 구성 매개변수를 ON으로 설정한 경우에만 해당됩니다.

ALLOW READ ACCESS 모드의 로드 조작에서 INDEXING MODE REBUILD 옵션을 지정한 경우 원래 인덱스와 동일한 테이블 스페이스 또는 시스템 임시 테이블 스페이스에서 새 인덱스는 *쉐도우*로 빌드됩니다. 원래 인덱스는 변경되지 않은 상태로 남아 있으며 로드 조작 중 사용 가능하고 테이블이 배타적으로 잠겨 있는 동안 오직 로드 조작 마지막에 새 인덱스로 교체됩니다. 로드 조작에 실패하고 트랜잭션이 롤백되면 원래 인덱스는 변경되지 않은 상태로 남아 있습니다.

디폴트로 쉐도우 인덱스는 원래 인덱스와 동일한 테이블 스페이스에 빌드됩니다. 원래 인덱스 및 새 인덱스가 동시에 유지보수되므로 동시에 두 인덱스를 보유하도록 테이블 스페이스 *크기*가 충분해야 합니다. 로드 조작이 중단되면 새 인덱스를 빌드하는 데 사용된 추가 스페이스가 해제됩니다. 로드 조작이 커미트되면 원래 인덱스에 사용된 스페이스가 해제되고 새 인덱스가 현재 인덱스가 됩니다. 새 인덱스가 원래 인덱스와 동일한 테이블 스페이스에 빌드되면 거의 동시에 원래 인덱스가 교체됩니다.

인덱스가 SMS 테이블 스페이스 내에 빌드되면 접미부가 .IN1 및 .INX인 테이블 스페이스 디렉토리에서 인덱스 파일을 볼 수 있습니다. 이러한 접미부는 원래 인덱스 및 쉐도우 인덱스를 구별하여 표시하지 않습니다. 그러나 인덱스가 DMS 테이블 스페이스에 빌드된 경우 새 쉐도우 인덱스를 볼 수 없습니다.

인덱스 작성 성능 향상

시스템 임시 테이블 스페이스에 새 인덱스 빌드
새 인덱스는 원래 테이블 스페이스의 공간이 부족하지 않도록 시스템 임시 테이블 스페이스에 빌드될 수 있습니다. USE <tablespace-name> 옵션을 사용하면 INDEXING MODE REBUILD 및 ALLOW READ ACCESS 옵션을 사용할 때 인덱스를 시스템 임시 테이블 스페이스에서 재빌드할 수 있습니다. 시스템 임시 테이블은 SMS 또는 DMS 테이블 스페이스일 수 있습니다. 그러나 시스템 임시 테이블 스페이스의 페이지 크기는 원래 인덱스 테이블 스페이스의 페이지 크기와 일치해야 합니다.

USE <tablespace-name> 옵션은 로드 조작이 ALLOW READ ACCESS 모드가 아니거나 인덱스 모드가 호환되지 않는 경우 무시됩니다. USE <tablespace-name> 옵션은 INDEXING MODE REBUILD 또는 INDEXING MODE AUTOSELECT 옵션에서만 지원됩니다. INDEXING MODE AUTOSELECT 옵션이 지정되고 로드 유틸리티가 인덱스의 증분 유지보수를 선택하면 USE <tablespace-name>은 무시됩니다.

로드 재시작 조작은 원래 로드 조작이 대체 테이블 스페이스를 사용하지 않아도 인덱스를 재빌드하는 데 대체 테이블 스페이스를 사용할 수 있습니다. 로드 재시작 조작은 원래 로드 조작이 ALLOW READ ACCESS 모드로 실행되지 않은 경우 ALLOW READ ACCESS 모드로 실행될 수 없습니다. 로드 종료 조작은 인덱스를 재빌드하지 않으므로 USE <tablespace-name>은 무시됩니다.

로드 조작의 빌드 단계 중 인덱스는 시스템 임시 테이블 스페이스에 빌드됩니다. 따라서 인덱스 복사 단계 중 인덱스는 시스템 임시 테이블 스페이스에서 원래 인덱스 테이블 스페이스로 복사됩니다. 원래 인덱스 테이블 스페이스에 새 인덱스를 작성할 스페이스가 충분하도록 빌드 단계 중 원래 테이블 스페이스에서 스페이스가 할당됩니다. 따라서 로드 조작으로 인덱스 스페이스가 부족해지면 빌드 단계 중 이 작업이 수행됩니다. 이 경우 원래 인덱스는 유실되지 않습니다.

인덱스 복사 단계는 빌드 및 삭제 단계 이후에 발생됩니다. 인덱스 복사 단계가 시작되기 전에 테이블은 배타적으로 잠깁니다. 즉, 인덱스 복사 단계 내내 읽기 액세스가 불가능합니다. 인덱스 복사 단계는 실제 복사 작업이므로 상당한 시간 동안 테이블을 사용하지 못할 수도 있습니다.

주: 시스템 임시 테이블 스페이스 또는 인덱스 테이블 스페이스가 DMS 테이블 스페이스인 경우 시스템 임시 테이블 스페이스에서 데이터를 읽으면 시스템 임시 테이블 스

페이스에서 무작위 입출력이 수행되어 대기 시간이 발생할 수 있습니다. 인덱스 테이블 스페이스에서의 쓰기 작업은 계속 최적화되며 DISK_PARALLELISM 값이 사용됩니다.

대형 인덱스에 대한 고려사항

로드 중 대형 인덱스를 빌드할 때 성능을 향상시키기 위해 *sortheap* 데이터베이스 구성 매개변수를 조정하는 것이 유용할 수 있습니다. *sortheap*은 로드 조작 중 인덱스 키 정렬 전용 메모리를 할당합니다. 예를 들어 키 정렬을 위해 인덱스당 주기억장치의 4000 페이지를 사용하도록 로드 유틸리티에 지시하려면 *sortheap*을 4000페이지로 설정하고 모든 응용 프로그램에서 데이터베이스와의 연결을 끊고 LOAD 명령을 실행합니다.

인덱스가 너무 커서 메모리에서 정렬할 수 없는 경우 *초과 정렬(sort spill)* 또는 오버플로우가 발생합니다. 즉, 데이터가 여러 ″정렬 실행 단위(sort run)″로 나누어지고 나중에 병합되는 임시 테이블 스페이스에 저장됩니다. sort_overflows 모니터 요소를 사용하여 초과 정렬(sort spill)의 수행 여부를 판별합니다. *sortheap* 매개변수 *크기*를 늘려 초과 정렬(sort spill)을 방지할 수 없는 경우 임시 테이블 스페이스의 버퍼 풀 공간을 충분히 확보하여 초과(spill)가 발생하는 디스크 입출력 *크기*를 줄일 수 있어야 합니다. 또한 정렬 실행 단위(sort run) 병합 중 입출력 병렬 처리를 수행하려면 각각 다른 디스크 디바이스에 있는 다중 컨테이너로 임시 테이블 스페이스를 선언하는 것이 좋습니다. 테이블에 둘 이상의 인덱스가 정의된 경우 로드 조작 시 메모리에 모든 키를 보존하므로 메모리 소비는 비례하여 늘어납니다.

인덱스 작성 지연

일반적으로 인덱스 작성을 지연하는 것보다 REBUILD 또는 INCREMENTAL 모드를 지정하여 로드 조작 중 인덱스를 작성하는 것이 보다 효율적입니다. 184 페이지의 그림 11에서와 같이 테이블은 보통 데이터 로드, 인덱스 빌드 및 통계 수집과 같은 3단계로 빌드됩니다. 그러면 로드 조작, 인덱스 작성(각 테이블에 여러 인덱스가 있을 수 있음) 및 통계 수집(모든 인덱스 및 테이블 데이터에서 입출력 발생) 중 다중 데이터 입출력이 나타날 수 있습니다. 더 빠른 대안은 로드 유틸리티에서 이 모든 태스크를 하나의 데이터 전달 처리로 완료하게 하는 것입니다. 그러나 중복 항목이 발견되면 고유 인덱스로 인해 로드 성능이 저하될 수 있다는 점에 주의해야 합니다.

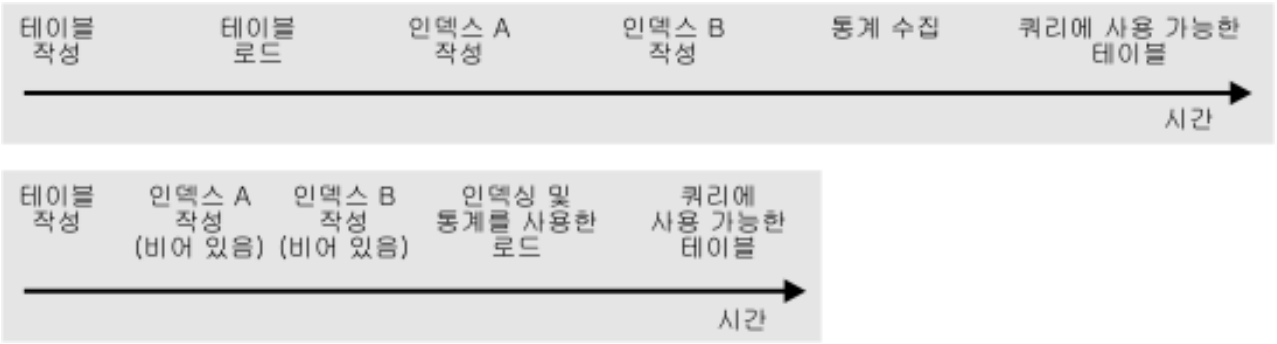


그림 11. 동시 인덱싱 및 통계 수집을 통해 로드 성능 향상. 테이블은 보통 데이터 로드, 인덱스 빌드 및 통계 수집과 같은 3단계로 빌드됩니다. 그러면 로드 조작, 인덱스 작성(각 테이블에 여러 인덱스가 있을 수 있음) 및 통계 수집(모든 인덱스 및 테이블 데이터에서 입출력 발생) 중 다중 데이터 입출력이 나타날 수 있습니다. 더 빠른 대안은 로드 유틸리티에서 이 모든 태스크를 하나의 데이터 전달 처리로 완료하게 하는 것입니다.

인덱스 작성을 지연하고 CREATE INDEX문을 호출하는 방법이 성능을 향상시키는 경우도 있습니다. 인덱스 재빌드 중 정렬 작업에서는 최대 *sortheap*개 페이지를 사용합니다. 추가 스페이스가 필요하면 TEMP 버퍼 풀이 사용되고 실제로 디스크로 초과(spill)됩니다. 로드 작업이 초과(spill)되어 성능이 저하되는 경우 INDEXING MODE DEFERRED와 함께 LOAD를 실행하고 나중에 인덱스를 재작성하는 것이 좋습니다. CREATE INDEX는 한 번에 하나의 인덱스를 작성하므로 테이블을 여러 번 스캔하여 키를 수집하는 동안 메모리 사용량이 줄어듭니다.

동시에 로드 조작을 수행하는 대신, CREATE INDEX문으로 인덱스를 빌드하는 경우 INTRA PARALLEL이 설정되면 CREATE INDEX문에서 다중 프로세스 또는 스레드를 사용하여 키를 정렬할 수 있다는 장점이 있습니다. 인덱스의 실제 빌드는 병렬로 실행되지 않습니다.

XML 데이터 로드 시 인덱싱 오류 해결

db2diag 로그 파일과 임포트 유틸리티를 함께 사용하여 XML 데이터의 문제점 값을 식별하여 정정함으로써 인덱싱 오류로 인해 실패하는 로드 조작을 해결할 수 있습니다.

로드 조작이 오류 메시지 SQL20305N(sqlcode -20305)을 리턴하는 경우는 하나 이상의 XML 노드 값을 인덱스화할 수 없음을 나타냅니다. 오류 메시지는 오류의 이유 코드를 출력합니다. 명령행 처리기에 `? SQL20305N`을 입력하여 해당 이유 코드에 대한 설명 및 사용자 조치를 검색하십시오.

삽입 조작 중에 발생하는 인덱싱 문제점의 경우 생성된 XQuery 명령문이 db2diag 로그 파일에 출력되어 문서 내에서 실패하는 XML 노드 값을 찾는 데 유용합니다. XQuery 명령문을 사용하여 실패하는 XML 노드 값을 찾는 방법에 대한 자세한 내용은 "일반 XML 인덱싱 문제"를 참조하십시오.

그러나 로드 조작 중에 발생하는 인덱싱 문제점의 경우, 생성된 XQuery 명령문이 db2diag 로그 파일에 출력되지 않습니다. 이러한 XQuery 명령문을 생성하려면 로드되

지 않은 실패하는 행에서 임포트 유틸리티를 실행해야 합니다. 거부된 행은 테이블에 존재하지 않기 때문에 XQuery 명령문을 실패하는 문서에서 실행할 수 없습니다. 이 문제점을 해결하려면 동일한 정의가 있는 새 테이블을 인덱스 없이 작성해야 합니다. 그러면 실패하는 행을 새 테이블에 로드한 후 XQuery 명령문을 새 테이블에서 실행하여 문서 내에서 실패하는 XML 노드 값을 찾을 수 있습니다.

인덱싱 오류를 해결하려면 다음 단계를 수행하십시오.

1. 출력 정보의 레코드 번호를 사용하여 로드 조작 중에 거부된 행을 판별하십시오.
2. 거부된 행만 포함하는 .del 파일을 작성하십시오.
3. 원래 테이블(T1)과 동일한 컬럼을 가진 새 테이블(예: T2)을 작성하십시오. 새 테이블에 대한 인덱스를 작성하지 마십시오.
4. 거부된 행을 새 테이블 T2에 로드하십시오.
5. 원래 테이블 T1의 거부된 각 행마다 다음과 같이 수행하십시오.
   a. 거부된 행을 T1에 임포트하여 SQL20305N 메시지를 가져오십시오. 임포트는 첫 번째 오류에 직면하면 중지됩니다.
   b. db2diag 로그 파일을 검토하고 생성된 XQuery 명령문을 가져오십시오. 입력 문서에서 실패하는 노드 값을 찾으려면 db2diag 로그 파일에서 'SQL20305N' 문자열을 검색하여 이유 코드 번호와 비교하십시오. 이유 코드 다음에 일련의 지시사항 및 사용 가능한 생성된 XQuery 명령문이 표시되어 오류 발생 원인이 된 문서의 문제점 값을 찾을 수 있습니다.
   c. 새 테이블 T2를 사용할 수 있도록 XQuery 명령문을 수정하십시오.
   d. T2에서 XQuery 명령문을 실행하여 문서에서 문제점 값을 찾으십시오.
   e. 문서를 포함하는 .xml 파일에서 문제점 값을 수정하십시오.
   f. 단계 a로 돌아가서 거부된 행을 다시 T1으로 임포트하십시오. 임포트 중지의 원인이 된 행이 이제 정상적으로 삽입되어야 합니다. .del 파일에 추가로 거부된 행이 있는 경우 다음 오류에서 임포트 유틸리티가 중지되고 또 다른 SQL20305N 메시지가 출력됩니다. 임포트가 정상적으로 실행될 때까지 이 단계를 계속 수행하십시오.

예

다음 예에서는 BirthdateIndex 인덱스가 *date* 데이터 유형에 작성되었습니다. REJECT INVALID VALUES 옵션이 지정되어 */Person/Confidential/Birthdate*의 XML 패턴 값이 모두 *date* 데이터 유형에 유효해야 합니다. 이 데이터 유형에 캐스트할 수 없는 XML 패턴 값이 있는 경우에는 오류가 리턴됩니다.

아래 XML 문서를 사용하여 5개의 행이 로드되어야 하지만 Birthdat 값을 인덱스화할 수 없기 때문에 첫 번째 및 네 번째 행이 거부됩니다. `person1.xml` 파일에서 `March 16, 2002` 값은 올바른 날짜 형식이 아닙니다. `person4.xml` 파일에서 `20000-12-09`

값의 연도에 영(0)이 하나 더 있으므로, 값은 유효한 XML 날짜 값이지만 DB2에서 연도에 허용하는 범위(0001 - 9999)에 해당되지 않습니다. 일부 샘플 출력을 편집하여 예를 더 간결하게 작성했습니다.

로드할 5개의 XML 파일은 다음과 같습니다.

person1.xml(Birthdate 값이 유효하지 않음)

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>March 16, 2002</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person>
```

person2.xml(Birthdate 값이 유효함)

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>2002-03-16</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person>
```

person3.xml(Birthdate 값이 유효함)

```
<?xml version="1.0"?>
<Person gender="Female">
  <Name>
    <Last>McCarthy</Last>
    <First>Laura</First>
  </Name>
  <Confidential>
    <Age unit="years">6</Age>
    <Birthdate>2001-03-12</Birthdate>
    <SS>444-55-6666</SS>
  </Confidential>
  <Address>5960 Daffodil Lane, San Jose, CA 95120</Address>
</Person>
```

person4.xml(Birthdate 값이 유효하지 않음)

```
<?xml version="1.0"?>
<Person gender="Female">
  <Name>
    <Last>Wong</Last>
    <First>Teresa</First>
  </Name>
  <Confidential>
    <Age unit="years">7</Age>
    <Birthdate>20000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential>
  <Address>5960 Tulip Court, San Jose, CA 95120</Address>
</Person>
```

person5.xml(Birthdate 값이 유효함)

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Smith</Last>
    <First>Chris</First>
  </Name>
  <Confidential>
    <Age unit="years">10</Age>
    <Birthdate>1997-04-23</Birthdate>
    <SS>666-77-8888</SS>
  </Confidential>
  <Address>5960 Dahlia Street, San Jose, CA 95120</Address>
</Person>
```

입력 파일 person.del에는 다음이 포함됩니다.

```
1, <XDS FIL='person1.xml'/>
2, <XDS FIL='person2.xml'/>
3, <XDS FIL='person3.xml'/>
4, <XDS FIL='person4.xml'/>
5, <XDS FIL='person5.xml'/>
```

DDL 및 LOAD 명령문은 다음과 같습니다.

```
CREATE TABLE T1 (docID INT, XMLDoc XML);

CREATE INDEX BirthdateIndex ON T1(xmlDoc)
   GENERATE KEY USING XMLPATTERN '/Person/Confidential/Birthdate' AS SQL DATE
   REJECT INVALID VALUES;

LOAD FROM person.del OF DEL INSERT INTO T1
```

위의 XML 파일 세트를 로드하려 할 때 발생할 수 있는 인덱싱 오류를 해결하려면 다음 단계를 수행하십시오.

1. 출력 정보의 레코드 번호를 사용하여 로드 조작 중에 거부된 행을 판별하십시오. 다음 출력에서 레코드 번호 1 및 레코드 번호 4가 거부되었습니다.

```
SQL20305N  XML 값이 테이블 "LEECM.T1"의 "IID = 3"에서
식별된 인덱스를 삽입 또는 갱신할 때 오류가 발견되었으므로 삽입하거나 갱신할 수 없습니다.
이유 코드 = "5".
XML 스키마에 관련된 이유 코드의 경우,
XML 스키마 ID = "*N" 및 XML 스키마 데이터 유형 = "*N". SQLSTATE=23525

SQL3185W  입력 파일의 "F0-1" 행에서 데이터를 처리하는 동안
```

```
이전의 오류가 발생했습니다.

SQL20305N  XML 값이 테이블 "LEECM.T1"의 "IID = 3"에서
식별된 인덱스를 삽입 또는 갱신할 때 오류가 발견되었으므로 삽입하거나 갱신할 수 없습니다.
이유 코드 = "4". XML 스키마에 관련된 이유 코드의 경우,
XML 스키마 ID = "*N" 및 XML 스키마 데이터 유형 = "*N". SQLSTATE=23525

SQL3185W  입력 파일의 "F0-4" 행에서 데이터를 처리하는 동안
이전의 오류가 발생했습니다.

SQL3227W  레코드 토큰 "F0-1"은 사용자 레코드 번호 "1"을 참조합니다.

SQL3227W  레코드 토큰 "F0-4"는 사용자 레코드 번호 "4"를 참조합니다.

SQL3107W  메시지 파일에 적어도 하나의 경고 메시지가 있습니다.

읽은 행 수               = 5
건너뛴 행 수             = 0
로드된 행 수             = 3
거부된 행 수             = 2
삭제된 행 수             = 0
커미트된 행 수           = 5
```

2. 거부된 행을 포함하는 새 파일 reject.del을 작성하십시오.

```
1, <XDS FIL='person1.xml'/>
4, <XDS FIL='person4.xml'/>
```

3. 원래 테이블 T1과 동일한 컬럼을 가진 새 테이블 T2를 작성하십시오. 새 테이블에 대한 인덱스를 작성하지 마십시오.

```
CREATE TABLE T2 LIKE T1
```

4. 거부된 행을 새 테이블 T2에 로드하십시오.

```
LOAD FROM reject.del OF DEL INSERT INTO T2;
```

5. 원래 테이블 T1에서 거부된 행 1과 관련하여 다음을 수행하십시오.

   a. 거부된 행을 T1에 임포트하여 -20305 메시지를 가져오십시오.

```
IMPORT FROM reject.del OF DEL INSERT INTO T1
SQL3109N  유틸리티가 파일 "reject.del"에서 데이터를 로드하기 시작합니다.

SQL3306N  테이블에 행을 삽입하는 동안 SQL 오류 "-20305"가 발생했습니다.

SQL20305N  XML 값이 테이블 "LEECM.T1"의 "IID = 3"에서
식별된 인덱스를 삽입 또는 갱신할 때 오류가 발견되었으므로 삽입하거나 갱신할 수 없습니다.
이유 코드 = "5". XML 스키마에 관련된 이유 코드의 경우,
XML 스키마 ID = "*N" 및 XML 스키마 데이터 유형 = "*N". SQLSTATE=23525

SQL3110N  유틸리티가 처리를 완료했습니다. 입력 파일에서 "1"개의 행을 읽었습니다.
```

   b. db2diag 로그 파일을 검토하고 생성된 XQuery 명령문을 가져오십시오.

```
FUNCTION: DB2 UDB, Xml Storage and Index Manager, xmlsDumpXQuery, probe:608
DATA #1 : String, 36 bytes
SQL Code: SQL20305N ; Reason Code: 5
DATA #2 : String, 265 bytes
To locate the value in the document that caused the error, create a
table with one XML column and insert the failing document in the table.
Replace the table and column name in the query below with the created
table and column name and execute the following XQuery.
DATA #3 : String, 247 bytes
xquery for $i in db2-fn:xmlcolumn(
  "LEECM.T1.XMLDOC")[/*:Person/*:Confidential/*:Birthdate="March 16, 2002"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

   c. 새 테이블 T2를 사용할 수 있도록 XQuery 명령문을 수정하십시오.

```
xquery for $i in db2-fn:xmlcolumn(
  "LEECM.T2.XMLDOC")[/*:Person/*:Confidential/*:Birthdate="March 16, 2002"]
return
```

```
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

d. 테이블 T2에서 XQuery 명령문을 실행하여 문서의 문제점 값을 찾으십시오.

```
<Result><ProblemDocument><Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>March 16, 2002</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person></ProblemDocument><ProblemValue><Confidential>
    <Age unit="years">5</Age>
    <Birthdate>March 16, 2002</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential></ProblemValue></Result>
```

e. 문서를 포함하는 person1.xml 파일에서 문제점 값을 수정하십시오. March 16, 2002 is not in the correct date format so it is changed to 2002-03-16.

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>2002-03-16</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person>
```

f. 단계 a.로 돌아가 거부된 행을 테이블 T1에 다시 임포트하십시오.

6. (단계 5 첫 번째 반복)

a. 거부된 행을 테이블 T1에 임포트하십시오. 임포트 파일에서 두 개의 행을 읽었기 때문에 첫 번째 행이 이제 정상적으로 임포트됩니다. 두 번째 행에서 새 오류가 발생합니다.

```
IMPORT FROM reject.del OF DEL INSERT INTO T1
SQL3109N  유틸리티가 파일 "reject.del"에서 데이터를
로드하기 시작합니다.

SQL3306N  테이블에 행을 삽입하는 동안 SQL 오류
"-20305"가 발생했습니다.

SQL20305N  XML 값이 테이블 "LEECM.T1"의 "IID = 3"에서
식별된 인덱스를 삽입 또는 갱신할 때 오류가 발견되었으므로
삽입하거나 갱신할 수 없습니다.
이유 코드 = "4". XML 스키마에 관련된 이유 코드의 경우,
XML 스키마 ID = "*N" 및 XML 스키마 데이터 유형 =
"*N". SQLSTATE=23525
```

```
SQL3110N  유틸리티가 처리를 완료했습니다.
입력 파일에서 "2"개의 행을 읽었습니다.
```

b. db2diag 로그 파일을 검토하고 생성된 XQuery 명령문을 가져오십시오.

```
FUNCTION: DB2 UDB, Xml Storage and Index Manager, xmlsDumpXQuery, probe:608
DATA #1 : String, 36 bytes
SQL Code: SQL20305N ; Reason Code: 4
DATA #2 : String, 265 bytes
To locate the value in the document that caused the error, create a
table with one XML column and insert the failing document in the table.
Replace the table and column name in the query below with the created
table and column name and execute the following XQuery.
DATA #3 : String, 244 bytes
xquery for $i in db2-fn:xmlcolumn("LEECM.T1.XMLDOC")
   [/*:Person/*:Confidential/*:Birthdate="20000-12-09"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

c. 테이블 T2를 사용하도록 XQuery 명령문을 수정하십시오.

```
xquery for $i in db2-fn:xmlcolumn("LEECM.T2.XMLDOC")
   [/*:Person/*:Confidential/*:Birthdate="20000-12-09"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

d. XQuery 명령문을 실행하여 문서에서 문제점 값을 찾으십시오.

```
<Result><ProblemDocument><Person gender="Female">
  <Name>
    <Last>Wong</Last>
    <First>Teresa</First>
  </Name>
  <Confidential>
    <Age unit="years">7</Age>
    <Birthdate>20000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential>
  <Address>5960 Tulip Court, San Jose, CA 95120</Address>
</Person></ProblemDocument><ProblemValue><Confidential>
    <Age unit="years">7</Age>
    <Birthdate>20000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential></ProblemValue></Result>
```

e. 문서를 포함하는 person4.xml 파일에서 문제점 값을 수정하십시오. 20000-12-09 값의 연도에 영(0)이 하나 더 있으므로 이 값은 DB2에서 연도에 허용하는 범위(0001 - 9999)에 해당되지 않습니다. 값이 2000-12-09로 변경됩니다.

```
<?xml version="1.0"?>
<Person gender="Female">
  <Name>
    <Last>Wong</Last>
    <First>Teresa</First>
  </Name>
  <Confidential>
    <Age unit="years">7</Age>
    <Birthdate>2000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential>
  <Address>5960 Tulip Court, San Jose, CA 95120</Address>
</Person>
```

f. 단계 a로 돌아가서 거부된 행을 다시 T1에 임포트하십시오.

7. (단계 5 두 번째 반복)

   a. 거부된 행을 T1에 임포트하십시오.

```
IMPORT FROM reject.del OF DEL INSERT INTO T1
SQL3109N  유틸리티가 파일 "reject.del"에서 데이터를 로드하기 시작합니다.

SQL3110N  유틸리티가 처리를 완료했습니다. 입력 파일에서 "2"개의 행을 읽었습니다.

SQL3221W  ...COMMIT WORK 시작. 입력 레코드 계수 = "2".

SQL3222W  데이터베이스 변경사항의 ...COMMIT가 성공적이었습니다.

SQL3149N  입력 파일에서 "2"개의 행이 처리되었습니다. 테이블에
"2"개의 행을 삽입했습니다. "0"개의 행이 거부되었습니다.
읽은 행 수                = 2
건너뛴 행 수              = 0
삽입된 행 수              = 2
갱신된 행 수              = 0
거부된 행 수              = 0
커미트된 행 수            = 2
```

   문제점이 이제 해결되었습니다. person.del의 모든 행이 정상적으로 테이블 T1에 삽입되었습니다.

로드 조작 중 압축 사전 작성

특정 기준을 만족하는 로드 INSERT 및 로드 REPLACE 조작은 자동 사전 작성(ADC)을 트리거합니다. 충분한 데이터가 처리된 후 COMPRESS 속성이 사용 가능하고 압축 사전이 없는 테이블에서 로드가 수행되면 ADC가 나타납니다.

데이터 행 압축에서는 데이터를 압축하는 데 정적 사전 기반 압축 알고리즘을 사용합니다. 최대 2개의 사전이 사용됩니다. 하나는 테이블 행 압축을 위한 사전, 다른 하나는 테이블에 하나 이상의 XML 컬럼이 포함된 경우 디폴트 XML 스토리지 오브젝트에 지정된 XML 문서 압축을 위한 사전입니다. 압축하려면 먼저 테이블에 사전이 있어야 합니다. 로드 조작 중 디폴트 동작(KEEPDICTIONARY 옵션으로 표시됨)은 기존 사전에 남거나 사전이 없는 경우 데이터의 특정 임계값을 스캔한 후 사전을 생성하는 것입니다. 개별 임계값을 초과하면 사전은 독립적으로 작성됩니다. 일반적으로 동시에 나타나지는 않습니다.

비XML 데이터의 경우 로드 유틸리티는 목표 테이블에 있는 데이터를 사용하여 사전을 빌드합니다. 이때 이 기존 데이터는 해당 테이블에 저장된 데이터 유형을 대표한다고 가정합니다. 목표 테이블에 기존 데이터가 충분하지 않은 경우 충분한 입력 데이터를 샘플링한 후에 로드 유틸리티가 입력 데이터 및 기존 데이터 모두를 사용하여 사전을 빌드합니다. XML 데이터의 경우 로드 유틸리티는 입력 데이터만 샘플링합니다.

범위 파티션된 테이블에 ADC가 나타나면 각 파티션은 개별 테이블로 처리됩니다. 교차 파티션 사전이 없으며 이미 사전을 포함한 파티션에 ADC는 나타나지 않습니다. 테이블 데이터의 경우 각 파티션에서 생성된 사전은 해당 파티션에만 있는 기존 테이블 데이터(및 필요한 경우 로드된 데이터)에 기반합니다. XML 데이터의 경우 각 파티션에서 생성된 사전은 해당 파티션으로 로드된 데이터에 기반합니다.

INSERT 모드로 수행된 조작은 내재적으로 KEEPDICTIONARY 동작을 따릅니다. 로드 REPLACE 조작의 경우 디폴트 옵션이지만 RESETDICTIONARY 옵션과 같은 추가 옵션이 제공됩니다.

KEEPDICTIONARY 옵션을 사용하는 로드 REPLACE

KEEPDICTIONARY 옵션을 사용하는 로드 REPLACE는 기존 사전을 보존하고 목표 테이블에서 COMPRESS 속성이 사용 가능한 한, 이를 사용하여 로드된 데이터를 압축합니다. 사전이 없는 경우 테이블로 로드된 데이터가 디폴트 XML 스토리지 오브젝트에 저장된 XML 문서 또는 테이블 행에 대한 사전 판별된 임계값을 억제하면 로드 유틸리티는 COMPRESS 속성이 사용 가능한 테이블에서 새 사전을 생성합니다. 목표 테이블의 데이터가 교체되므로 로드 유틸리티는 입력 데이터만 사용하여 사전을 빌드합니다. 사전을 작성한 후에는 테이블로 삽입되고 로드 조작이 계속됩니다.

RESETDICTIONARY 옵션을 사용하는 로드 REPLACE

COMPRESS 속성이 설정된 테이블로 로드할 때 RESETDICTIONARY 옵션 사용은 두 가지 중요한 내용을 함축합니다. 먼저 로드 REPLACE를 완료한 후 목표 테이블에 일정 데이터가 있으면 사전이 작성됩니다. 즉, 새 압축 사전은 단일 XML 문서 또는 데이터의 단일 행에 기반할 수 있습니다. 또 다른 함축된 내용은 다음 조건이 만족되는 경우 기존 사전이 삭제되지만 교체되지 않습니다. 즉, 목표 테이블에 더 이상 압축 사전이 없습니다.

- COMPRESS 속성이 해제된 테이블에서 조작이 수행된 경우
- 아무 항목도 로드되지 않는 경우(0개 행). 이때 통지 로그에 ADM5591W가 인쇄됩니다.

주: RESETDICTIONARY 옵션과 함께 로드 REPLACE를 실행한 후 로드 TERMINATE 조작을 실행하면 기존 압축 사전이 삭제되고 교체되지 않습니다.

성능 영향

사전 작성은 다음과 같은 두 가지 방식으로 로드 조작 성능에 영향을 줍니다.

- 로드 INSERT의 경우 ADC의 최소 임계값만 아닌 모든 기존 테이블 데이터가 압축 사전을 빌드하기 전에 스캔됩니다. 따라서 이 스캔에 소요되는 시간은 테이블 크기에 따라 늘어납니다. 이 영향은 XML 사전에는 적용되지 않습니다.
- 압축 사전을 빌드하는 추가 처리. 실제로 사전을 빌드하는 데 사용되는 시간은 최소의 시간입니다. 또한 일단 사전을 빌드하면 디폴트로 ADC가 꺼집니다.

로드 성능 향상을 위한 옵션

로드 성능을 최적화하는 데 사용할 수 있는 다양한 명령 매개변수가 있습니다. 일부 경우 유틸리티 성능을 크게 향상시킬 수 있는 로드에 고유한 여러 파일 유형 수정자도 있습니다.

명령 매개변수

로드 유틸리티는 DISK_PARALLELISM, CPU_PARALLELISM 및 DATA BUFFER 매개변수 값이 사용자에 의해 지정되지 않은 경우 최적의 값을 판별하여 최상의 성능을 전달하려고 합니다. 최적화는 유틸리티 힙에서 사용 가능한 여유 공간 및 크기에 따라 수행됩니다. 특정 요구에 따라 이 매개변수를 조정하기 전에 DISK_PARALLELISM 및 CPU_PARALLELISM 자동 설정 사용을 고려합니다.

다음은 로드 유틸리티를 통해 사용 가능한 다양한 옵션에 함축된 성능 정보입니다.

ALLOW READ ACCESS

이 옵션을 사용하면 로드 조작을 진행하면서 테이블을 쿼리할 수 있습니다. 로드 조작 이전에 테이블에 있는 데이터를 볼 수만 있습니다. INDEXING MODE INCREMENTAL 옵션도 지정된 경우 로드 조작에 실패하면 후속 로드 종료 조작은 인덱스에서 불일치를 정정해야 할 수도 있습니다. 이를 수행하려는 경우 상당한 입출력이 예상되는 인덱스 스캔이 요구됩니다. 로드 종료 조작에서 ALLOW READ ACCESS 옵션도 지정되면 입출력에 대해 버퍼 풀이 사용됩니다.

COPY YES 또는 NO

이 매개변수를 사용하여 로드 조작 중 입력 데이터를 복사할 것인지 여부를 지정합니다. COPY YES(포워드 복구가 가능한 경우에만 적용 가능함)는 로드 조작 중 로드하는 모든 데이터가 복사되므로 로드 성능이 떨어집니다. 증가된 입출력 활동으로 입출력에 바인드된 시스템에서 로드 시간이 늘어납니다. 다른 디스크에서 다중 디바이스 또는 디렉토리를 지정하면 이 조작으로 인한 성능 페널티를 일부 상쇄할 수 있습니다. COPY NO(포워드 복구가 가능한 경우에만 적용 가능함)는 로드 성능에 영향을 주지 않습니다. 그러나 로드된 테이블과 관련된 모든 테이블 스페이스가 백업 보류 상태가 되므로 테이블에 액세스하려면 먼저 해당 테이블 스페이스를 백업해야 합니다.

CPU_PARALLELISM

이 매개변수를 사용하여 데이터베이스 파티션당 실행하는 프로세스 수를 사용하고(시스템 성능에 포함된 경우) 로드 성능을 크게 향상시킵니다. 이 매개변수에서는 로드 유틸리티가 데이터 레코드를 구문 분석, 변환 및 형식화하는 데 사용하는 프로세스 또는 스레드 수를 지정합니다. 허용되는 최대 수는 30개입니다. 지정된 값을 지원하는 데 메모리가 부족한 경우 유틸리티에서 값을 조정합니다. 이 매개변수를 지정하지 않으면 로드 유틸리티는 시스템의 CPU 수에 기반한 디폴트값을 선택합니다.

소스 데이터의 레코드 순서는 다음 경우에 이 매개변수 값에 상관없이 보존됩니다(194 페이지의 그림 12 참조).

- `anyorder` 파일 유형 수정자가 지정되지 않음

- PARTITIONING_DBPARTNUMS 옵션(및 파티셔닝에 사용할 둘 이상의 파티션)이 지정되지 않음

테이블에 LOB 또는 LONG VARCHAR 데이터가 포함되면 CPU_PARALLELISM이 1로 설정됩니다. 이 경우 병렬 처리는 지원되지 않습니다.

이 매개변수 사용이 SMP(Symmetric Multi-Processor) 하드웨어로 제한되지 않은 경우 비SMP 환경에서 이를 사용해도 인식할 수 있는 성능 이점을 얻을 수 없습니다.

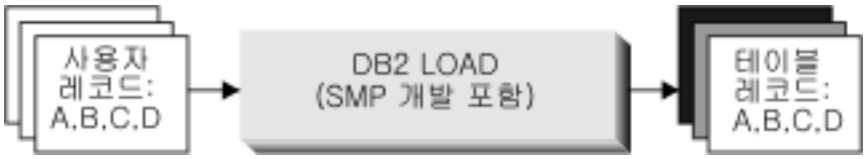


그림 12. 로드 조작 중 데이터베이스 파티션당 실행하는 프로세스 수를 사용하는 경우 소스 데이터의 레코드 순서 보존

DATA BUFFER

DATA BUFFER 매개변수에서는 버퍼로 로드 유틸리티에 할당된 전체 메모리 크기(4KB 단위)를 지정합니다. 이 버퍼는 크기 면에서 여러 *Extent*에 있는 것이 좋습니다. 데이터 버퍼는 유틸리티 힙에서 할당됩니다. 시스템에서 사용 가능한 스토리지 크기에 따라 DB2 유틸리티에서 사용할 추가 메모리를 할당하는 방법을 고려해야 합니다. 데이터베이스 구성 매개변수 *util_heap_sz*(유틸리티 힙 크기)는 적절히 수정할 수 있습니다. *util_heap_sz*의 디폴트값은 5 000 4KB 페이지입니다. 로드는 유틸리티 힙에서 메모리를 사용하는 여러 유틸리티 중 하나일 뿐이므로 이 매개변수에서 정의한 페이지의 최대 50%만 로드 유틸리티에서 사용 가능하게 하고 유틸리티 힙은 충분히 크게 정의하는 것이 좋습니다.

DISK_PARALLELISM

DISK_PARALLELISM 매개변수에서는 로드 유틸리티가 디스크에 데이터 레코드를 작성하는 데 사용하는 프로세스 또는 스레드 수를 지정합니다. 이 매개변수를 사용하여 데이터를 로드할 때 사용 가능한 컨테이너를 사용하고 로드 성능을 크게 향상시킵니다. 허용되는 최대 수는 CPU_PARALLELISM 값(실제로 로드 유틸리티에서 사용되는 값)의 4배 또는 50입니다. 디폴트로 DISK_PARALLELISM은 이 값이 허용되는 최대 수를 초과하는 경우를 제외하고 로드할 테이블의 오브젝트를 포함하는 모든 테이블 스페이스에 있는 테이블 스페이스 컨테이너의 합과 동일합니다.

NONRECOVERABLE

포워드 복구가 사용 가능한 경우 롤 포워드 시 테이블에서 로드 트랜잭션을 복구할 필요가 없으면 이 매개변수를 사용합니다. NONRECOVERABLE 로드 및 COPY NO 로드는 성능이 동일합니다. 그러나 잠재적 데이터 유실 측면에

서 큰 차이가 있습니다. NONRECOVERABLE 로드는 테이블이 완전히 액세스 가능한 동안 롤 포워드 복구 불가능한 항목으로 테이블을 표시합니다. 로드 조작을 통해 롤 포워드해야 하는 경우 로드된 데이터 및 테이블에 대한 모든 후속 갱신 사항이 유실되는 문제 상황으로 이어질 수 있습니다. COPY NO 로드는 백업 보류 상태(백업을 수행할 때까지 액세스 불가능한 테이블을 렌더링함)의 모든 종속 테이블 스페이스를 로드합니다. 이러한 유형의 로드 이후에 강제로 백업을 수행하므로 로드된 데이터 또는 테이블에 대한 후속 갱신 사항을 유실할 위험이 없습니다. 즉, COPY NO 로드는 완전히 복구 가능합니다.

주: 후속 리스토어 및 롤 포워드 복구 조작 중 이러한 로드 트랜잭션이 발생하면 테이블은 갱신되지 않으며 invalid로 표시됩니다. 이 테이블에서의 추가 조치는 무시됩니다. 롤 포워드 조작이 완료되면 테이블은 삭제만 가능합니다.

SAVECOUNT

이 매개변수를 사용하여 로드 조작의 **로드 단계 중** 일관성 지점을 설정하는 간격을 설정합니다. 일관성 지점을 설정하기 위해 수행되는 활동의 동기화는 다소 시간이 걸립니다. 너무 빈번하게 수행하면 로드 성능이 현저하게 떨어집니다. 많은 행 수를 로드할 경우 SAVECOUNT 값을 크게 지정하는 것이 좋습니다. 예를 들어 1억 개의 레코드를 포함하는 로드 조작의 경우 값을 1천 만으로 지정합니다.

로드 재시작 조작이 로드 단계부터 다시 시작되는 경우 로드 재시작 조작은 마지막 일관성 지점부터 자동으로 계속됩니다.

STATISTICS USE PROFILE

테이블 통계 프로파일에 지정된 통계를 수집합니다. 이 매개변수를 사용하여 로드 조작 자체의 성능이 감소한 경우(특히 DETAILED INDEXES ALL이 지정된 경우)에도 로드 조작 완료 이후에 RUNSTATS 유틸리티를 호출하는 것보다 더 효율적으로 데이터 분산 및 인덱스 통계를 수집합니다.

최적의 성능을 위해 응용프로그램에는 최상의 데이터 분산 및 인덱스 통계가 요구됩니다. 통계가 갱신되면 응용프로그램은 최신 통계에 기반한 테이블 데이터에 대한 새 액세스 경로를 사용할 수 있습니다. 테이블에 대한 새 액세스 경로는 BIND 명령을 사용하여 응용프로그램 패키지를 리바인드함으로써 작성할 수 있습니다. 테이블 통계 프로파일은 RUNSTATS 명령을 SET PROFILE 옵션과 함께 실행하여 작성됩니다.

데이터를 대형 테이블에 로드하는 경우 *stat_heap_sz*(통계 힙 크기) 데이터베이스 구성 매개변수에 더 큰 값을 지정하는 것이 좋습니다.

USE <tablespace-name>

ALLOW READ ACCESS 로드가 수행되고 인덱스 모드가 REBUILD인 경

우 이 매개변수를 사용하면 인덱스를 시스템 임시 테이블 스페이스에 재빌드하고 로드 조작의 인덱스 복사 단계 중 인덱스 테이블 스페이스로 다시 복사할 수 있습니다.

디폴트로 완전히 재빌드된 인덱스(*쉐도우 인덱스*라고도 함)는 원래 인덱스와 동일한 테이블 스페이스에 빌드됩니다. 원래 인덱스 및 쉐도우 인덱스가 동일한 테이블 스페이스에 동시에 있으므로 자원 문제점이 발생할 수 있습니다. 쉐도우 인덱스가 원래 인덱스와 동일한 테이블 스페이스에 빌드되면 원래 인덱스는 쉐도우로 즉시 교체됩니다. 그러나 쉐도우 인덱스가 시스템 임시 테이블 스페이스에 빌드되면 로드 조작에서는 시스템 임시 테이블 스페이스에서 인덱스 테이블 스페이스로 인덱스를 복사하는 인덱스 복사 단계를 수행해야 합니다. 복사 시 상당한 입출력이 예상됩니다. 두 테이블 스페이스가 DMS 테이블 스페이스인 경우 시스템 임시 테이블 스페이스의 입출력은 순차적이 아닐 수 있습니다. DISK_PARALLELISM 옵션에서 지정된 값은 인덱스 복사 단계 중 고려됩니다.

WARNINGCOUNT

이 매개변수를 사용하여 로드 조작을 강제로 종료하기 전에 유틸리티에서 리턴할 수 있는 경고 수를 지정합니다. 적은 수의 경고 또는 전혀 경고가 예상되지 않으면 WARNINGCOUNT 매개변수를 비교적 낮은 값으로 설정합니다. 로드 조작은 WARNINGCOUNT 수에 도달한 후에 중지됩니다. 이 경우 로드 조작을 완료하기 전에 문제점을 정정할 기회가 주어집니다.

파일 유형 수정자

ANYORDER

디폴트로 로드 유틸리티는 소스 데이터의 레코드 순서를 보존합니다. 로드가 SMP 환경에서 작동하는 경우 순서를 보존하려면 병렬 처리 간 동기화가 요구됩니다.

SMP 환경에서 anyorder 파일 유형 수정자를 지정하면 로드 유틸리티에 순서를 보존하도록 지시하지 않습니다. 그러면 해당 순서를 보존하는 데 필요한 동기화를 수행하지 않아도 되므로 성능이 향상됩니다. 그러나 로드할 데이터를 사전 정렬하면 anyorder로 인해 사전 정렬된 순서가 손상되고 후속 쿼리 시 사전 정렬의 이점이 사라질 수 있습니다.

주: anyorder 파일 유형 수정자는 CPU_PARALLELISM이 1인 경우 적용되지 않으며 SAVECOUNT 옵션과 호환 가능하지 않습니다.

BINARYNUMERICS, **ZONEDDECIMAL** 및 **PACKEDDECIMAL**

고정 길이 컬럼 식별자 없는 ASCII(ASC) 소스 데이터의 경우 2진으로 숫자 데이터를 표시하면 로드 시 성능이 향상될 수 있습니다. packeddecimal 파일 유형 수정자를 지정하면 10진수 데이터는 로드 유틸리티에서 압축 10진수 형식(바이트당 2자리)으로 해석됩니다. zoneddecimal 파일 유형 수정자를 지정하면 10진수 데이터는 로드 유틸리

티에서 존 10진수 형식(zoned decimal format)(바이트당 1자리)으로 해석됩니다. 기타 모든 숫자 유형에서 binarynumerics 파일 유형 수정자가 지정되면 데이터는 로드 유틸리티에서 2진 형식으로 해석됩니다.

주:

- binarynumerics, packeddecimal 또는 zoneddecimal 파일 유형 수정자를 지정하면 플랫폼에 상관없이 숫자 데이터는 빅 엔디안(big-endian) 형식(높은 바이트가 먼저 옴)으로 해석됩니다.
- packeddecimal 및 zoneddecimal 파일 유형 수정자는 상호 배타적입니다.
- packeddecimal 및 zoneddecimal 파일 유형 수정자는 10진수 목표 컬럼에만 적용되며 2진 데이터는 목표 컬럼 정의와 일치해야 합니다.
- binarynumerics, packeddecimal 또는 zoneddecimal 파일 유형 수정자가 지정된 경우 reclen 파일 유형 수정자를 지정해야 합니다.

FASTPARSE
주의하여 사용해야 합니다. 로드할 데이터가 유효한 경우 로드 시 의심되는 데이터와 동일한 크기의 구문 검사를 수행하지 않아도 됩니다. 실제로 이 단계의 범위를 줄이면 로드 성능이 10에서 20% 정도 향상될 수 있습니다. 이는 fastparse 파일 유형 수정자를 사용하면 됩니다. 그러면 ASC 및 DEL 파일의 사용자 제공 컬럼 값에서 수행되는 데이터 검사가 줄어듭니다.

NOROWWARNINGS
로드 조작 중 거부된 행에 대한 경고 메시지는 지정된 파일에 작성됩니다. 그러나 로드 유틸리티가 거부되거나 유효하지 않거나 절단된 많은 행을 처리해야 하는 경우 로드 성능이 저하될 수 있습니다. 많은 경고가 예상되는 경우 이러한 경고 기록을 억제하도록 norowwarnings 파일 유형 수정자를 사용하는 것이 좋습니다.

PAGEFREESPACE, INDEXFREESPACE 및 TOTALFREESPACE
테이블에서 데이터가 여러 번 삽입 및 갱신되어 테이블 및 인덱스를 재구성할 필요성이 증가합니다. 한 가지 해결 방법은 pagefreespace, indexfreespace 및 totalfreespace를 사용하여 테이블 및 인덱스의 여유 공간 크기를 늘리는 것입니다. 처음 두 수정자(PCTFREE 값보다 우선됨)는 여유 공간으로 남겨둘 데이터 및 인덱스 페이지의 백분율을 지정하는 반면, totalfreespace는 테이블에 여유 공간으로 추가할 전체 페이지 수의 백분율을 지정합니다.

참조 무결성 유지보수를 위한 기능 로드

일반적으로 로드 유틸리티가 임포트 유틸리티보다 효율적이지만 로드할 정보의 참조 무결성을 보장하려면 여러 기능이 필요합니다.

- **테이블 잠금**: 이 기능은 로드 조작 중 동시처리 제어를 제공하고 제어되지 않은 데이터가 없도록 예방합니다.

- **테이블 상태** 및 **테이블 스페이스 상태**: 이 기능은 데이터에 대한 액세스를 제어하거나 특정 사용자 조치를 유추할 수 있습니다.
- **예외 테이블 로드**: 이 기능을 사용하면 사용자 알림 없이 유효하지 않은 데이터 행이 삭제되지 않습니다.

로드 조작 이후 무결성 위반 검사

로드 조작 이후에 다음 조건에 해당되면 READ 또는 NO ACCESS 모드에서 로드된 테이블은 무결성 설정 보류 상태가 될 수 있습니다.

- 테이블에 테이블 점검 제한조건 또는 참조 무결성 제한조건이 정의되어 있습니다.
- 테이블에 생성된 컬럼이 있고 V7 이하 클라이언트가 로드 조작을 시작하는 데 사용됩니다.
- 테이블에 인접한 하위 구체화된 쿼리 테이블 또는 이를 참조하는 인접한 하위 스테이징 테이블이 있습니다.
- 테이블이 스테이징 테이블 또는 구체화된 쿼리 테이블입니다.

로드된 테이블에 대응하는 SYSCAT.TABLES 항목의 STATUS 플래그는 테이블이 무결성 설정 보류 상태임을 표시합니다. 로드된 테이블을 완전히 사용 가능하게 하려면 STATUS 값이 N이고 ACCESS MODE 값이 F여야 합니다. 이는 테이블이 완전히 액세스 가능하며 정상 상태임을 나타냅니다.

로드된 테이블에 하위 테이블이 있으면 SET INTEGRITY PENDING CASCADE 매개변수를 지정하여 로드된 테이블의 무결성 설정 보류 상태를 하위 테이블에 바로 전달해야 하는지 여부를 표시할 수 있습니다.

로드된 테이블에 제한조건이 있고 하위 외부 키 테이블, 종속 구체화된 쿼리 테이블 및 종속 스테이징 테이블이 있으며 모든 테이블이 로드 조작 이전에 정상 상태이면 지정된 로드 매개변수에 따라 다음 결과가 예상됩니다.

INSERT, ALLOW READ ACCESS 및 SET INTEGRITY PENDING CASCADE IMMEDIATE

로드된 테이블, 종속 구체화된 쿼리 테이블 및 종속 스테이징 테이블은 읽기 액세스 권한이 있는 무결성 설정 보류 상태가 됩니다.

INSERT, ALLOW READ ACCESS 및 SET INTEGRITY PENDING CASCADE DEFERRED

로드된 테이블만 읽기 액세스 권한이 있는 무결성 설정 보류 상태가 됩니다. 하위 외부 키 테이블, 하위 구체화된 쿼리 테이블 및 하위 스테이징 테이블은 원래 상태로 남아 있습니다.

INSERT, ALLOW NO ACCESS 및 SET INTEGRITY PENDING CASCADE IMMEDIATE

로드된 테이블, 종속 구체화된 쿼리 테이블 및 종속 스테이징 테이블은 액세스 권한이 없는 무결성 설정 보류 상태가 됩니다.

INSERT 또는 REPLACE, ALLOW NO ACCESS 및 SET INTEGRITY PENDING CASCADE DEFERRED

로드된 테이블만 액세스 권한이 없는 무결성 설정 보류 상태가 됩니다. 하위 외부 키 테이블, 인접한 하위 구체화된 쿼리 테이블 및 인접한 하위 스테이징 테이블은 원래 상태로 남아 있습니다.

REPLACE, ALLOW NO ACCESS 및 SET INTEGRITY PENDING CASCADE IMMEDIATE

테이블 및 모든 해당 하위 외부 키 테이블, 인접한 하위 구체화된 쿼리 테이블 및 인접한 하위 스테이징 테이블은 액세스 권한이 없는 무결성 설정 보류 상태가 됩니다.

주: 로드 교체 조작에서 ALLOW READ ACCESS 옵션을 지정하면 오류가 발생합니다.

무결성 설정 보류 상태를 제거하려면 SET INTEGRITY문을 사용합니다. SET INTEGRITY문에서는 테이블에서 제한조건 위반을 검사하고 테이블을 무결성 설정 보류 상태에서 해제합니다. INSERT 모드로 모든 로드 조작을 수행하면 SET INTEGRITY 문을 사용하여 점진적으로 제한조건을 처리할 수 있습니다. 즉, 테이블의 추가된 부분에서만 제한조건 위반을 검사합니다. 예를 들면, 다음과 같습니다.

```
db2 load from infile1.ixf of ixf insert into table1
db2 set integrity for table1 immediate checked
```

TABLE1의 추가된 부분에서만 제한조건 위반을 검사합니다. 특히 대형 테이블에서 추가된 데이터가 적은 경우 추가된 부분에서만 제한조건 위반을 검사하면 전체 테이블을 검사할 때보다 속도가 더 빠릅니다.

SET INTEGRITY PENDING CASCADE DEFERRED 옵션을 지정하여 테이블을 로드하고 SET INTEGRITY문을 사용하여 무결성 위반을 검사하는 경우 하위 테이블은 액세스 권한이 없이 무결성 설정 보류 상태가 됩니다. 테이블을 이 상태에서 해제하려면 명시적 요청을 실행해야 합니다.

INSERT 옵션을 사용하여 종속 구체화된 쿼리 테이블 또는 종속 스테이징 테이블을 포함하는 테이블을 로드하고 SET INTEGRITY문을 사용하여 무결성 위반을 검사하는 경우 테이블은 무결성 설정 보류 상태에서 해제되고 데이터 이동 없음 상태가 됩니다. 이는 종속 구체화된 쿼리 테이블의 증분 새로 고침 및 종속 스테이징 테이블의 증분 전파를 활용하기 위해 수행됩니다. 데이터 이동 없음 상태에서 테이블 내 행을 이동시키는 조작은 허용되지 않습니다.

SET INTEGRITY문을 실행할 때 FULL ACCESS 옵션을 지정하면 데이터 이동 없음 상태를 무시할 수 있습니다. 테이블은 완전히 액세스 가능하지만 후속 REFRESH TABLE문에서 종속 구체화된 쿼리 테이블을 재적용하게 되고 종속 스테이징 테이블이 강제로 불완전 상태가 됩니다.

로드 조작에서 ALLOW READ ACCESS 옵션이 지정되면 SET INTEGRITY문을 사용하여 제한조건 위반을 검사할 때까지 테이블은 읽기 액세스 상태로 남아 있습니다. 응용프로그램은 커미트된 후 로드 조작 이전에 있던 데이터를 테이블에서 쿼리할 수 있지만 SET INTEGRITY문을 실행할 때까지 새로 로드한 데이터를 볼 수 없습니다.

제한조건 위반을 검사하기 전에 테이블에서 여러 로드 조작을 수행할 수 있습니다. ALLOW READ ACCESS 모드에서 모든 로드 조작을 완료하면 첫 번째 로드 조작 이전에 테이블에 있던 데이터만 쿼리에 사용할 수 있습니다.

이 명령문의 단일 호출에서 하나 이상의 테이블을 검사할 수 있습니다. 종속 테이블을 고유하게 검사하려는 경우 상위 테이블은 무결성 설정 보류 상태일 수 없습니다. 그렇지 않으면 상위 테이블 및 종속 테이블 모두 동시에 검사해야 합니다. 참조 무결성 순환의 경우 순환과 관련된 모든 테이블은 SET INTEGRITY문의 단일 호출에 포함되어야 합니다. 그러면 종속 테이블을 로드하는 동안 상위 테이블에서 제한조건 위반을 편리하게 검사할 수 있습니다. 이는 두 테이블이 동일한 테이블 스페이스에 없는 경우에만 발생할 수 있습니다.

SET INTEGRITY문을 실행하는 경우 INCREMENTAL 옵션을 지정하여 증분 처리를 명시적으로 요청할 수 있습니다. 대부분의 경우 DB2 데이터베이스에서 증분 처리를 선택하므로 이 옵션은 필요하지 않습니다. 증분 처리가 가능하지 않으면 전체 처리를 자동으로 사용합니다. INCREMENTAL 옵션이 지정되지만 증분 처리가 가능하지 않으면 다음 경우에 오류가 리턴됩니다.

- 무결성 설정 보류 상태인 동안 새 제한조건이 테이블에 추가됩니다.
- 로드 교체 조작이 수행되거나 테이블에서 마지막 무결성 검사를 수행한 후 NOT LOGGED INITIALLY WITH EMPTY TABLE 옵션이 활성화됩니다.
- 비점진적으로 상위 테이블이 로드 교체되거나 상위 테이블에서 무결성을 검사합니다.
- 업그레이드 전에 테이블이 무결성 설정 보류 상태입니다. 업그레이드한 후 테이블에서 무결성을 처음 검사할 때 전체를 처리해야 합니다.
- 테이블 또는 해당 상위를 포함하는 테이블 스페이스는 특정 시점으로 롤 포워드되고 테이블 및 해당 상위는 다른 테이블 스페이스에 있습니다.

테이블에서 SYSCAT.TABLES 카탈로그의 CONST_CHECKED 컬럼에 하나 이상의 W 값이 있고 SET INTEGRITY문에 NOT INCREMENTAL 옵션이 지정되지 않은 경우 테이블은 점진적으로 처리되고 SYSCAT.TABLES의 CONST_CHECKED 컬럼은 U로 표시되어 시스템에서 일부 데이터는 검증하지 않음을 나타냅니다.

SET INTEGRITY문은 제한조건을 위반하는 행을 삭제하는 결과로 DELETE 트리거를 활성화하지 않지만 테이블이 무결성 설정 보류 상태에서 제거되면 트리거가 활성화됩니다. 따라서 데이터를 정정하고 예외 테이블에서 로드된 테이블로 행을 삽입하면 테이블에 정의된 INSERT 트리거가 활성화됩니다. 이 함축 내용을 고려해야 합니다. 한 가지 옵션은 INSERT 트리거를 삭제하고 예외 테이블에서 행을 삽입한 후 INSERT 트리거를 재작성하는 것입니다.

SET INTEGRITY를 사용한 제한조건 위반 검사

일반적으로, 테이블에 데이터를 로드한 이후, 테이블에 제한조건을 추가함으로써 테이블을 변경하는 경우 및 생성된 컬럼을 추가하기 위해 테이블을 변경하는 경우의 세 가지 상황에서 테이블의 무결성 처리를 수동으로 수행해야 합니다.

- 테이블에 대한 제한조건 검사를 설정하고 테이블에서 무결성 처리를 수행하려면 다음 중 하나가 필요합니다.
  - 검사 중인 테이블에 대한 CONTROL 특권 및 하나 이상의 테이블에 예외가 포스트되는 경우 예외 테이블에 대한 INSERT 특권
  - 명령문을 통해 내재적으로 무결성 설정 보류 상태에 놓이게 되는 모든 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블에 대한 CONTROL 특권
  - LOAD 권한 및 하나 이상의 테이블에 예외가 포스트되는 경우 다음과 같은 권한
    - 검사 중인 각 테이블에 대한 SELECT 및 DELETE 특권
    - 예외 테이블의 INSERT 특권
- 테이블에서 무결성 처리를 수행하지 않고 테이블에 대한 제한조건 검사를 설정하려면 다음 중 하나가 필요합니다.
  - 검사 중인 테이블에 대한 CONTROL 특권
  - 명령문을 통해 내재적으로 무결성 설정 보류 상태에 놓이게 되는 모든 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블에 대한 CONTROL 특권
  - LOAD 권한
  - DATAACCESS 권한
  - DBADM 권한
- 제한조건 검사, 즉시 새로 고침 또는 테이블에 즉시 전파를 설정 해제하려면 다음 중 하나가 필요합니다.
  - 명령문을 통해 무결성 검사가 설정 해제되는 모든 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블에 대한 CONTROL 특권
  - LOAD 권한

로드 조작을 수행하면 테이블에 제한조건이 정의되어 있거나 종속 외부 키 테이블, 종속 구체화된 쿼리 테이블 또는 종속 스테이징 테이블이 있는 경우 테이블이 자동으로 무결성 설정 보류 상태가 됩니다. 로드 조작이 완료되면 로드된 데이터의 무결성을 검증할 수 있으며 테이블에 대한 제한조건 검사를 설정할 수 있습니다. 테이블에 종속 외부 키 테이블, 종속 구체화된 쿼리 테이블 또는 종속 스테이징 테이블이 있는 경우 이들 종속 테이블은 자동으로 무결성 설정 보류 상태가 됩니다. 각각의 종속 테이블에서 개별 무결성 처리를 수행하려면 무결성 설정 창을 사용해야 합니다.

외부 키, 점검 제한조건 또는 생성된 컬럼을 추가하여 테이블을 변경 중인 경우 테이블을 변경하기 전에 먼저 제한조건 검사를 설정 해제해야 합니다. 제한조건을 추가한 후에는 새로 추가된 제한조건을 위반하지 않는지 기존 데이터를 점검하고 제한조건 검사를 다시 설정해야 합니다. 또한 테이블에 데이터를 로드 중인 경우 데이터 로드를 완료하기 전에는 테이블에 대한 제한조건 검사를 활성화할 수 없습니다. 테이블에 데이터를 임포트 중인 경우에는 데이터를 임포트하기 전에 테이블에 대한 제한조건 검사를 활성화해야 합니다.

제한조건 검사는 제한조건 위반, 외부 키 위반 및 생성된 컬럼 위반을 검사하는 것을 가리킵니다. 무결성 처리는 제한조건 검사 수행 이외에 ID 및 생성된 컬럼 채우기, 구체화된 쿼리 테이블 새로 고침 및 스테이징 테이블에 전파하는 것을 가리킵니다.

일반적으로 테이블에 대한 참조 무결성 및 점검 제한조건은 자동으로 강제 실행되고 구체화된 쿼리 테이블은 자동으로 즉시 새로 고쳐지며 스테이징 테이블은 자동으로 전파됩니다. 일부 상황에서는 이 동작을 수동으로 변경해야 할 수도 있습니다.

제어 센터를 사용하여 제한조건 위반을 점검하려면 다음과 같이 수행하십시오.

1. 무결성 설정 창 열기: 제어 센터에서 **테이블** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오. **테이블** 폴더를 누르십시오. 기존 테이블이 모두 창의 오른쪽 분할창에 표시됩니다. 원하는 테이블을 마우스 오른쪽 단추로 누른 후 팝업 메뉴에서 **무결성 설정**을 선택하십시오. 무결성 설정 창이 열립니다.
2. 작업 중인 테이블의 현재 무결성 상태를 검토하십시오.
3. 테이블에 대한 제한조건 검사를 설정하고 테이블 데이터를 검사하지 않으려면 다음과 같이 수행하십시오.
   a. **즉시 및 검사하지 않음** 라디오 단추를 선택하십시오.
   b. 성능 조정 중인 무결성 처리 유형을 지정하십시오.
   c. **전체 액세스** 라디오 단추를 선택하여 테이블에 대한 데이터 이동 조작(예: 재구성 또는 재분배)을 즉시 수행하십시오. 그러나 종속 구체화된 쿼리 테이블의 연속 새로 고침은 시간이 오래 걸리는 점에 유의하십시오. 테이블에 연관 구체화된 쿼리 테이블이 있는 경우, 구체화된 쿼리 테이블을 새로 고치는 데 필요한 시간을 단축시키기 위해 이 라디오 단추를 선택하지 않는 것이 좋습니다.

4. 테이블에 대한 제한조건 검사를 설정하고 기존 테이블 데이터를 점검하려면 다음과 같이 수행하십시오.
   a. **즉시 및 검사함** 라디오 단추를 선택하십시오.
   b. 수행할 무결성 처리 유형을 선택하십시오. **현재 무결성 상태**가 구체화된 쿼리 테이블의 제한조건 점검 값이 불완전한 것으로 표시되는 경우 구체화된 쿼리 테이블을 점진적으로 새로 고칠 수 없습니다.
   c. 선택사항: 무결성 처리 중에 ID 또는 생성된 컬럼을 채우려면 **강제 생성** 체크 박스를 선택하십시오.
   d. 테이블이 스테이징 테이블이 아닌 경우 **프룬(prune)** 체크 박스가 선택 취소되었는지 확인하십시오.
   e. **전체 액세스** 라디오 단추를 선택하여 테이블에 대한 데이터 이동 조작을 즉시 수행하십시오.
   f. 선택사항: 예외 테이블을 지정하십시오. 참조 또는 점검 제한조건을 위반하는 모든 행은 테이블에서 삭제되고 예외 테이블로 복사됩니다. 예외 테이블을 지정하지 않으면 제한조건이 위반되는 경우 첫 번째로 발견된 위반사항만 사용자에게 리턴되고 테이블은 무결성 설정 보류 상태로 남습니다.
5. 제한조건 검사, 즉시 새로 고침 또는 테이블에 즉시 전파를 설정 해제하려면 다음과 같이 수행하십시오.
   a. **해제** 라디오 단추를 선택하십시오. 테이블이 무결성 설정 보류 상태가 됩니다.
   b. **연쇄** 옵션을 사용하여 즉시 연쇄시킬지 또는 연쇄를 지연시킬지 여부를 지정하십시오. 즉시 연쇄시킬 경우 **구체화된 쿼리 테이블**, **외부 키 테이블** 및 **스테이징 테이블** 체크 박스를 사용하여 연쇄시킬 테이블을 표시하십시오.

      주: 상위 테이블의 제한조건 검사를 설정 해제하고 외부 키 테이블에 변경사항을 연쇄시키도록 지정하면 모든 종속 외부 키 테이블의 외부 키 제한 조건도 설정 해제됩니다. 기본 테이블의 제한조건 검사를 설정 해제하고 구체화된 쿼리 테이블에 점검 보류 상태를 연쇄시키도록 지정하면 모든 종속 구체화된 쿼리 테이블의 즉시 새로 고침 등록 정보도 설정 해제됩니다. 기본 테이블의 제한조건 검사를 설정 해제하고 스테이징 테이블에 무결성 설정 보류 상태를 연쇄시키도록 지정하면 모든 종속 스테이징 테이블의 즉시 전파 등록 정보도 설정 해제됩니다.

명령행을 사용하여 제한조건 위반을 점검하려면 SET INTEGRITY문을 사용하십시오.

문제점 해결 추가 정보

증상 제한조건 검사, 즉시 새로 고침 또는 테이블에 즉시 전파를 설정하려고 하면 다음 오류 메시지가 수신됩니다.

DB2 메시지

상위 테이블 또는 기본 테이블 TABLE2가 무결성 설정 보류 상태에

있거나 SET INTEGRITY문을 통해 무결성 설정 보류 상태가 될 경우 SET INTEGRITY문을 사용하여 종속 테이블 TABLE1을 검사할 수 없습니다.

여기서 TABLE1은 제한조건 검사, 즉시 새로 고침 또는 즉시 전파를 설정하려는 테이블이며 TABLE2에 종속됩니다.

가능한 원인

상위 또는 기본 테이블이 무결성 설정 보류 상태인 테이블에는 제한조건 검사, 즉시 새로 고침이나 즉시 전파를 설정할 수 없습니다.

조치 상위 또는 기본 테이블에 제한조건 검사를 설정하여 무결성 설정 보류 상태가 해제되도록 하십시오. DB2 메시지에서 상위 또는 기본 테이블로 식별된 테이블부터 시작하십시오. 해당 테이블이 다른 테이블에 종속되는 경우 종속성 체인의 맨 위에 있는 테이블에서 시작하여 맨 위에서 아래쪽으로 제한조건 검사를 설정해야 합니다.

주의: 선택된 테이블이 하나 이상의 테이블과 순환 참조 제한조건 관계에 있는 경우 무결성 설정 창을 사용하여 제한조건 검사를 설정할 수 없습니다. 이런 경우 명령 편집기를 사용하여 SQL SET INTEGRITY 명령을 발행해야 합니다.

로드 조작 중 테이블 잠금

대부분의 경우 로드 유틸리티에서는 테이블 레벨 잠금을 사용하여 테이블에 대한 액세스를 제한합니다. 잠금 레벨은 로드 조작 스테이지 및 읽기 액세스를 허용하도록 지정되었는지 여부에 따라 달라집니다.

ALLOW NO ACCESS 모드의 로드 조작에서는 로드 지속 기간에 테이블에 대한 강한 독점 잠금(Z-잠금)을 사용합니다.

ALLOW READ ACCESS 모드의 로드 조작을 시작하기 전에, 로드 유틸리티는 로드 조작보다 먼저 시작되는 모든 응용프로그램이 목표 테이블에서 잠금을 해제하길 기다립니다. 로드 조작 시작 시 로드 유틸리티는 테이블에 대한 갱신 잠금(U-잠금)을 획득합니다. 데이터를 커미트할 때까지 이 잠금을 보유합니다. 로드 유틸리티가 테이블에 대한 U-잠금을 획득하면 호환 가능한 잠금인 경우에도 로드 조작 시작 전에 테이블의 잠금을 보유하는 모든 응용프로그램이 잠금을 해제하길 기다립니다. 요청된 잠금이 로드 조작의 U-잠금과 호환 가능한 한, 목표 테이블에 대한 새 테이블 잠금 요청과 충돌하지 않는 Z-잠금으로 U-잠금을 일시적으로 업그레이드하여 수행할 수 있습니다. 데이터를 커미트하면 로드 유틸리티는 잠금을 Z-잠금으로 업그레이드합니다. 로드 유틸리티가 잠금이 충돌하는 응용프로그램이 완료되길 기다리는 동안 커미트 시간에 약간의 대기 시간이 발생할 수 있습니다.

주: 로드 전에 응용프로그램이 잠금을 해제하길 기다리는 동안 로드 조작이 제한 시간을 초과할 수 있습니다. 그러나 데이터를 커미트하는 데 필요한 Z-잠금을 기다리는 동안 로드 조작은 시간 초과되지 않습니다.

잠금이 충돌하는 응용프로그램

LOAD 명령의 LOCK WITH FORCE 옵션을 사용하여 목표 테이블에서 충돌하는 잠금을 보유하는 응용프로그램을 해제합니다. 그러면 로드 조작을 계속 진행할 수 있습니다. ALLOW READ ACCESS 모드의 로드 조작을 진행하기 전에 다음 잠금을 보유하는 응용프로그램이 강제로 해제됩니다.

- 테이블 갱신 잠금과 충돌하는 테이블 잠금(예: 임포트 또는 삽입).
- 로드 조작의 커미트 단계에 존재하는 모든 테이블 잠금.

시스템 카탈로그 테이블에서 충돌하는 잠금을 보유하는 응용프로그램은 로드 유틸리티에 의해 강제로 해제되지 않습니다. 시스템 로드 유틸리티에 의해 응용프로그램이 강제로 해제되면 응용프로그램은 데이터베이스 연결을 잃고 오류가 리턴됩니다(SQL1224N).

복구 가능한 데이터베이스에서 로드 조작에 대해 COPY NO 옵션을 지정하면 테이블스페이스가 백업 보류 상태가 되기 전에 목표 테이블 스페이스의 모든 오브젝트가 공유 모드로 잠깁니다. 액세스 모드와는 상관없이 발생합니다. LOCK WITH FORCE 옵션을 지정하면 공유 잠금과 충돌하는 테이블 스페이스에 있는 오브젝트에 대한 잠금을 보유하는 모든 응용프로그램이 강제로 해제됩니다.

읽기 액세스 로드 조작

로드 유틸리티에서는 다른 응용프로그램이 로드할 테이블에 대해 보유해야 하는 액세스 권한을 제어하는 두 가지 옵션을 제공합니다. ALLOW NO ACCESS 옵션은 테이블을 배타적으로 잠그고 테이블을 로드하는 동안 테이블 데이터에 대한 액세스를 허용하지 않습니다.

ALLOW NO ACCESS 옵션은 디폴트 동작입니다. ALLOW READ ACCESS 옵션을 사용하면 다른 응용프로그램에 테이블에 대한 모든 쓰기 액세스 권한을 부여하지 않지만 기존 데이터에 대한 읽기 액세스 권한은 허용합니다. 이 절에서는 ALLOW READ ACCESS 옵션을 다룹니다.

로드 조작을 시작하기 전부터 존재하는 테이블 데이터 및 인덱스 데이터는 로드 조작을 진행하는 동안에도 쿼리에 표시됩니다. 다음 예를 고려해 보십시오.

1. 하나의 정수 컬럼을 포함하는 테이블을 작성하십시오.

```
create table ED (ed int)
```

2. 3개 행을 로드하십시오.

```
load from File1 of del insert into ED
...
읽은 행 수                = 3
건너뛴 행 수              = 0
```

```
로드된 행 수               = 3
거부된 행 수               = 0
삭제된 행 수               = 3
커미트된 행 수             = 3
```

3. 테이블을 쿼리하십시오.

```
select * from ED

ED
-----------
          1

          2
          3

  3 레코드가 선택되었습니다.
```

4. ALLOW READ ACCESS 옵션을 지정하여 로드 조작을 수행하고 데이터의 두 행 이상을 로드하십시오.

```
load from File2 of del insert into ED allow read access
```

5. 로드 조작을 진행 중인 동안 동시에 다른 연결에서 테이블을 쿼리하십시오.

```
select * from ED

ED
-----------
          1

          2
          3

  3 레코드가 선택되었습니다.
```

6. 로드 조작 완료를 기다린 후 테이블을 쿼리하십시오.

```
select * from ED

ED
-----------
          1

          2
          3
          4              5
  5 레코드가 선택되었습니다.
```

ALLOW READ ACCESS 옵션은 대형 데이터를 로드하는 경우 매우 유용합니다. 이는 로드 조작이 진행 중이거나 로드 조작에 실패한 후에도 항상 사용자가 테이블 데이터에 액세스할 수 있기 때문입니다. ALLOW READ ACCESS 모드에서 로드 조작의 동작은 응용프로그램의 분리 수준과 독립되어 있습니다. 즉, 어떤 분리 수준이든 판독기는 항상 기존 데이터를 읽을 수 있습니다. 그러나 로드 조작을 완료할 때까지 새로 로드한 데이터는 읽을 수 없습니다.

읽기 액세스는 조작 시작 및 조작 종료와 같은 두 경우를 제외하고 로드 조작 내내 제공됩니다.

먼저 로드 조작은 설정 단계가 거의 종료되는 짧은 지속 기간에 특수 Z-잠금을 획득합니다. 응용프로그램이 특수 Z-잠금을 요청하는 로드 조작 전에 테이블에서 호환되지 않는 잠금을 보유하는 경우 로드 조작은 제한 시간이 초과되어 실패하기 전에 한정된 시간 동안 이 호환되지 않는 잠금을 해제하길 기다립니다. 대기 시간은 *locktimeout* 데이터베이스 구성 매개변수로 판별됩니다. LOCK WITH FORCE 옵션이 지정되면 로드 조작에서 제한 시간이 초과되지 않도록 다른 응용프로그램을 강제로 해제합니다. 그 다음 로드 조작은 특수 Z-잠금을 확보하고 단계를 커미트하고 잠금을 해제한 후 로드 단계로 계속 진행합니다. ALLOW READ ACCESS 모드로 로드 조작을 시작한 후 읽기 위해 테이블에 대한 잠금을 요청하는 응용프로그램에 잠금 권한이 부여되며 이 특수 Z-잠금과는 충돌하지 않습니다. 목표 테이블에서 기존 데이터를 읽으려는 새 응용프로그램도 이를 수행할 수 있습니다.

그 다음으로 로드 조작 종료 시 데이터를 커미트하기 전에 로드 유틸리티는 테이블에 대한 배타적 잠금(Z-잠금)을 확보합니다. 로드 유틸리티는 테이블에 대한 잠금을 보유하는 모든 응용프로그램이 이를 해제하기를 기다립니다. 그러면 데이터를 커미트하기 전에 대기 시간이 발생할 수 있습니다. LOCK WITH FORCE 옵션을 사용하여 충돌하는 응용프로그램을 강제로 해제하고 대기 시간 없이 로드 조작을 계속 진행할 수 있습니다. 일반적으로 ALLOW READ ACCESS 모드의 로드 조작은 짧은 기간 동안 배타적 잠금을 확보합니다. 그러나 USE <tablespace-name> 옵션이 지정되면 전체 인덱스 복사 단계 내내 배타적 잠금이 지속됩니다.

다중 데이터베이스 파티션에 정의된 테이블에서 로드 유틸리티를 실행하는 경우 각 개별 데이터베이스 파티션에서 로드 프로세스 모델이 실행됩니다. 즉, 다른 db-partitions 과는 독립적으로 잠금이 획득 및 해제됩니다. 따라서 쿼리 또는 기타 조작이 동시에 실행되고 동일한 잠금에 대해 경합하는 경우 교착 상태가 나타날 수 있습니다. 예를 들어 db-partition 0에서 조작 A에 테이블 잠금 권한이 부여되고 db-partition 1에서 로드 조작에 테이블 잠금 권한이 부여되는 경우를 고려합니다. 로드 조작이 db-partition 0에서 테이블 잠금을 기다리는 동안 db-partition 1에서 조작 A가 테이블 잠금 권한 부여를 대기하므로 교착 상태가 발생할 수 있습니다. 이 경우 교착 상태 검출기는 임의로 조작 중 하나를 롤백합니다.

주:

1. 로드 조작이 인터럽트되거나 실패한 경우 로드 조작을 실행할 때 지정한 동일한 액세스 등급으로 남아 있습니다. 즉, ALLOW NO ACCESS 모드의 로드 조작에 실패하면 로드가 종료되거나 로드 재시작이 실행될 때까지 테이블 데이터에 액세스할 수 없습니다. ALLOW READ ACCESS 모드의 로드 조작이 중단되면 기존 테이블 데이터는 읽기 액세스에 대해 여전히 액세스 가능합니다.

2. 인터럽트되거나 실패한 로드 조작에 대해 ALLOW READ ACCESS 옵션이 지정된 경우 로드 재시작 또는 로드 종료 조작에 대해서도 이를 지정할 수 있습니다. 그러나 인터럽트되거나 실패한 로드 조작에서 ALLOW NO ACCESS 옵션이 지정된 경우 로드 재시작 또는 로드 종료 조작에 대해 ALLOW READ ACCESS 옵션을 지정할 수 없습니다.

ALLOW READ ACCESS 옵션은 다음 경우에 지원되지 않습니다.

- REPLACE 옵션이 지정됩니다. 새 데이터를 로드하기 전에 로드 교체 조작이 기존 테이블 데이터를 절단하는 경우 로드 조작을 완료할 때까지 기존 데이터를 쿼리하지 못합니다.
- 인덱스가 유효하지 않은 항목으로 표시되며 재빌드를 기다립니다. 일부 롤 포워드 시나리오에서 또는 db2dart 명령을 사용하여 인덱스가 유효하지 않은 항목으로 표시될 수 있습니다.
- INDEXING MODE DEFERRED 옵션이 지정됩니다. 이 모드에서는 인덱스를 재빌드가 필요하다고 표시합니다.
- ALLOW NO ACCESS 로드 조작이 재시작되거나 종료됩니다. 완전히 온라인이 될 때까지 테이블에서 ALLOW READ ACCESS 모드의 로드 조작을 수행할 수 없습니다.
- 무결성 설정 보류 권한 없음 상태인 테이블로 로드 조작이 수행됩니다. 또한 제한조건이 있는 테이블에서 다중 로드 조작을 하는 경우에도 이에 해당합니다. SET INTEGRITY문을 실행할 때까지 테이블은 온라인 상태가 되지 않습니다.

일반적으로 테이블 데이터가 오프라인 상태가 되면 테이블 데이터가 다시 온라인 상태가 되기 전까지 로드 조작 중 읽기 액세스가 불가능합니다.

로드 조작 도중/이후 테이블 스페이스 상태

로드 유틸리티에서는 테이블 스페이스 상태를 사용하여 로드 조작 중 데이터베이스 일관성을 유지합니다. 이 상태는 데이터에 대한 액세스를 제어하거나 사용자 조치를 유추하여 작동합니다.

로드 유틸리티는 로드 조작과 관련된 테이블 스페이스에서 지속적 잠금 획득(Quiesce)을 수행하지 않습니다. COPY NO 매개변수를 지정한 경우 로드 조작에 대해서만 테이블 스페이스 상태를 사용합니다.

LIST TABLESPACES 명령을 사용하여 테이블 스페이스 상태를 점검할 수 있습니다. 테이블 스페이스는 동시에 여러 상태일 수 있습니다. LIST TABLESPACES에서 리턴하는 상태는 다음과 같습니다.

일반
정상 상태는 테이블을 작성한 이후 테이블 스페이스의 초기 상태로, 현재 테이블에 영향을 주는 상태(비정상 상태)가 없음을 나타냅니다.

로드 진행 중

로드 진행 중 상태는 테이블 스페이스에서 로드가 진행 중임을 나타냅니다. 이 상태이면 로드 중 종속 테이블을 백업하지 못합니다. 테이블 스페이스 상태는 로드 진행 중 테이블 상태(모든 로드 조작에 사용됨)와는 구별됩니다. 로드 유틸리티는 복구 가능한 데이터베이스에서 **COPY NO** 매개변수를 지정한 경우에만 로드 진행 중 상태로 테이블 스페이스를 배치합니다. 테이블 스페이스는 로드 조작 지속 기간에 이 상태로 남아 있습니다.

백업 보류

복구 가능한 데이터베이스에서 로드 조작을 수행하고 **COPY NO** 매개변수를 지정한 경우 첫 번째 커미트 이후 테이블 스페이스는 백업 보류 테이블 스페이스 상태로 배치됩니다. 백업 보류 상태의 테이블 스페이스는 갱신할 수 없습니다. 테이블 스페이스 백업만을 통해 테이블 스페이스를 백업 보류 상태에서 해제할 수 있습니다. 로드 조작을 취소해도 테이블 스페이스는 백업 보류 상태로 남아 있습니다. 로드 조작 시작 시 테이블 스페이스 상태가 변경되어 롤백될 수 없기 때문입니다.

리스토어 보류

COPY NO 옵션을 사용하여 로드 조작을 성공적으로 수행하고 데이터베이스를 리스토어한 후 해당 조작을 통해 롤 포워드한 경우 연관된 테이블 스페이스는 리스토어 보류 상태가 됩니다. 테이블 스페이스를 리스토어 보류 상태에서 해제하려면 리스토어 작업을 수행해야 합니다.

테이블 스페이스 상태에 대한 예

다음과 같이 입력 파일(staffdata.del)을 테이블 NEWSTAFF로 로드하는 경우

```
update db cfg for sample using logretain recovery;
backup db sample;
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff copy no;
connect reset;
```

다른 세션을 열고 다음 명령을 발행합니다.

```
connect to sample;
list tablespaces;
connect reset;
```

USERSPACE1(샘플 데이터베이스의 디폴트 테이블 스페이스)이 로드 진행 중 상태이고 첫 번째 커미트 이후 백업 보류 상태가 됩니다. 로드 조작을 완료하면 LIST TABLESPACES 명령에서 USERSPACE1의 현재 상태를 백업 보류 상태로 표시합니다.

```
테이블 스페이스 ID                    = 2
이름                                  = USERSPACE1
유형                                  = 데이터베이스 관리 스페이스
```

```
컨텐츠                                  = 모든 영구 데이터. 대형 테이블 스페이스
상태                                    = 0x0020
  세부사항 설명:
  백업 보류
```

로드 조작 도중/이후 테이블 상태

로드 유틸리티에서는 테이블 상태를 사용하여 로드 조작 중 데이터베이스 일관성을 유지합니다. 이 상태는 데이터에 대한 액세스를 제어하거나 사용자 조치를 유추하여 작동합니다.

테이블 상태를 판별하려면 LOAD QUERY 명령을 실행합니다. 이 명령에서는 로드 조작 상태도 확인합니다. 테이블은 동시에 여러 상태일 수 있습니다. LOAD QUERY에서 리턴하는 상태는 다음과 같습니다.

일반 상태
정상 상태는 테이블을 작성한 이후 테이블의 초기 상태로, 현재 테이블에 영향을 주는 상태(비정상 상태)가 없음을 나타냅니다.

읽기 액세스 전용
ALLOW READ ACCESS 옵션을 지정하면 테이블은 읽기 액세스 전용 상태가 됩니다. 로드 명령 호출 전에 존재하는 테이블의 데이터는 로드 조작 중 읽기 전용 모드로 사용 가능합니다. ALLOW READ ACCESS 옵션을 지정하고 로드 조작에 실패한 경우 로드 조작 전 테이블에 존재하는 데이터는 실패 후에도 읽기 전용 모드로 계속 사용 가능합니다.

로드 진행 중
로드 진행 중 테이블 상태는 테이블에서 로드가 진행 중임을 나타냅니다. 로드 유틸리티는 로드를 성공적으로 완료하면 이 임시 상태를 제거합니다. 그러나 로드 조작에 실패하거나 인터럽트되면 테이블 상태가 로드 보류로 변경됩니다.

재분배 진행 중
재분배 진행 중 테이블 상태는 테이블에서 재분배가 진행 중임을 나타냅니다. 테이블 처리가 성공적으로 완료되면 재분배 유틸리티가 이 임시 상태를 제거합니다. 그러나 재분배 조작에 실패하거나 인터럽트되면 테이블 상태가 재분배 보류로 변경됩니다.

로드 보류
로드 보류 테이블 상태는 로드 조작이 실패했거나 인터럽트되었음을 나타냅니다. 다음 단계 중 하나를 수행하여 로드 보류 상태를 제거할 수 있습니다.

- 실패 원인을 설명합니다. 예를 들어 로드 유틸리티에서 디스크 공간이 부족하면 테이블 스페이스에 컨테이너를 추가합니다. 그리고 로드 조작을 재시작합니다.
- 로드 조작을 종료합니다.
- 로드 조작에 실패했던 동일한 테이블에서 REPLACE 로드 조작을 실행합니다.

- 최신 테이블 스페이스 또는 데이터베이스 백업에서 RESTORE DATABASE 명령을 사용하여 로드하는 테이블의 테이블 스페이스를 복구한 후 추가 복구 조치를 수행합니다.

재분배 보류
재분배 보류 테이블 상태는 재분배 조작이 실패했거나 인터럽트되었음을 나타냅니다. REDISTRIBUTE CONTINUE 또는 REDISTRIBUTE ABORT 조작을 수행하여 재분배 보류 상태를 제거할 수 있습니다.

로드 재시작 불가능
로드 재시작 불가능 상태에서 테이블은 부분적으로 로드되며 로드 재시작 조작은 허용되지 않습니다. 다음 두 가지 경우에 테이블이 로드 재시작 불가능 상태가 됩니다.

- 재시작 또는 종료할 수 없는 실패한 로드 조작 이후 롤 포워드 조작을 수행하는 경우
- 테이블이 로드 진행 중 또는 로드 보류 상태인 동안 수행된 온라인 백업에서 리스토어 작업을 수행하는 경우

또한 테이블은 로드 보류 상태가 됩니다. 테이블을 로드 재시작 불가능 상태에서 해제하려면 LOAD TERMINATE 또는 LOAD REPLACE 명령을 실행합니다.

무결성 설정 보류
무결성 설정 보류 상태는 로드된 테이블에 아직 검증되지 않은 제한조건이 있음을 나타냅니다. 제한조건이 있는 테이블에서 로드 조작을 시작하면 로드 유틸리티는 테이블을 이 상태로 배치합니다. SET INTEGRITY문을 사용하여 테이블을 무결성 설정 보류 상태에서 해제합니다.

Type-1 인덱스
Type-1 인덱스 상태는 테이블이 현재 type-1 인덱스를 사용함을 나타냅니다. Type-1 인덱스는 더 이상 지원되지 않으며 type-2 인덱스로 변환되어야 합니다. REORG INDEXES/TABLE 명령의 CONVERT 옵션 또는 db2IdentifyType1 명령의 출력을 사용하여 인덱스를 type-2 인덱스로 변환할 수 있습니다. db2IdentifyType1 명령은 지정된 데이터베이스의 테이블 또는 스키마에서 찾은 type-1 인덱스의 변환에 적절한 명령을 생성합니다. 자세한 정보는 『type-1 인덱스를 type-2 인덱스로 변환』 주제를 참조하십시오.

사용 불가능
복구 불가능한 로드 조작을 통해 롤 포워드하면 테이블이 사용 불가능 상태가 됩니다. 이 상태에서 테이블은 사용 불가능하며 백업에서 리스토어하거나 삭제해야 합니다.

다중 상태의 테이블에 대한 예

다음과 같이 많은 데이터를 포함하는 입력 파일(`staffdata.del`)을 테이블 NEWSTAFF로 로드하는 경우:

```
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff allow read access;
connect reset;
```

다른 세션을 열고 다음 명령을 발행합니다.

```
connect to sample;
load query table newstaff;
connect reset;
```

LOAD QUERY 명령은 NEWSTAFF 테이블이 읽기 액세스 전용 및 로드 진행 중 상태임을 표시합니다.

```
테이블 상태:
 로드 진행 중
 읽기 액세스 전용
```

로드 예외 테이블

로드 예외 테이블은 로드 조작 중 고유 인덱스 규칙, 범위 제한조건 및 보안 규정을 위반한 모든 행을 통합하여 보고하는 항목입니다. LOAD 명령에서 FOR EXCEPTION 절을 사용하여 로드 예외 테이블을 지정합니다.

제한사항: 예외 테이블은 ID 컬럼 또는 생성된 컬럼의 기타 유형을 포함하지 않습니다. ID 컬럼이 기본 테이블에 있으면 예외 테이블의 해당 컬럼은 컬럼의 유형, 길이 및 널(null) 허용 가능 속성만 포함해야 합니다. 또한 예외 테이블은 파티션될 수 없습니다. 또한 고유 인덱스를 포함할 수 없습니다. 또한 다음 경우에 예외 테이블은 지정할 수 없습니다.

- 목표 테이블이 LBAC 보안을 사용하며 하나 이상의 XML 컬럼을 포함합니다.
- 목표 테이블이 범위가 파티션되었으며 하나 이상의 XML 컬럼을 포함합니다.

로드 유틸리티에서 사용하는 예외 테이블은 SET INTEGRITY문에서 사용하는 예외 테이블과 동일합니다. 사용자 작성된 테이블로, 로드할 테이블의 정의를 반영하며 일부 추가 컬럼을 포함합니다.

로드할 테이블이 있는 테이블 스페이스 또는 다른 테이블 스페이스에 로드 예외 테이블을 지정할 수 있습니다. 두 경우 모두 로드 예외 테이블 및 로드할 테이블을 동일한 데이터베이스 파티션 그룹에 지정하고 두 테이블이 동일한 분산 키를 사용하도록 합니다.

예외 테이블을 사용하는 경우

예외 테이블은 고유 인덱스를 포함하고 레코드가 중복될 수 있는 데이터를 로드할 때 사용합니다. 예외 테이블을 지정하지 않은 경우 중복 레코드가 발견되면 로드 조작이 계속되고 삭제된 중복 레코드에 대해 경고 메시지만 발행됩니다. 중복 레코드는 로그되지 않습니다.

로드 조작을 완료한 후 예외 테이블의 정보를 사용하여 오류가 발생한 데이터를 정정할 수 있습니다. 그러면 정정한 데이터를 테이블에 삽입할 수 있습니다.

행은 예외 테이블의 기존 정보에 추가됩니다. 테이블이 비어 있음을 확인하는 검사가 수행되지 않으므로 새 정보는 단순히 이전 로드 조작의 유효하지 않은 행에 추가됩니다. 현재 로드 조작의 유효하지 않은 행만 확인하려는 경우 유틸리티를 호출하기 전에 기존 행을 제거할 수 있습니다. 또는 로드 조작을 정의하는 경우 예외 테이블에서 위반을 발견한 시간 및 위반된 제한조건 이름을 기록하도록 지정할 수 있습니다.

각 삭제 이벤트가 로그되므로 고유성 조건을 위반한 레코드가 많은 경우 로드의 삭제 단계에서 로그가 가득 찰 수 있습니다.

인덱스 빌드 전에 유효하지 않은 데이터로 거부된 행은 예외 테이블에 삽입되지 않습니다.

실패 또는 불완전한 로드

인터럽트된 로드 조작 재시작

로드 조작 중 실패 또는 인터럽트가 발생한 경우 로드 유틸리티를 사용하여 조작을 종료하거나 테이블을 다시 로드하거나 로드 조작을 재시작할 수 있습니다.

존재하지 않는 데이터 파일 또는 유효하지 않은 컬럼 이름과 같은 사용자 오류로 로드 유틸리티가 시작되지 않으면 조작은 종료되고 목표 테이블은 정상 상태로 남습니다.

로드 조작이 시작되면 목표 테이블은 로드 진행 중 테이블 상태가 됩니다. 실패한 경우 테이블 상태는 로드 보류로 변경됩니다. 테이블을 이 상태에서 제거하기 위해 LOAD TERMINATE를 실행하여 조작을 롤백하거나 LOAD REPLACE를 실행하여 전체 테이블을 다시 로드하거나 LOAD RESTART를 실행할 수 있습니다.

일반적으로 이 상황에서는 로드 조작을 재시작하는 것이 최선의 방법입니다. 로드 유틸리티가 진행 중 마지막으로 성공했던 지점부터 로드 조작을 재시작하므로 조작 처음부터 시작할 때보다 시간을 절약할 수 있습니다. 조작이 재시작되는 정확한 위치는 원래 명령에 지정된 매개변수에 따라 달라집니다. SAVECOUNT 옵션을 지정하고 이전 로드 조작이 로드 단계에서 실패한 경우 로드 조작은 마지막으로 도달한 일관성 지점에서 재시작됩니다. 그렇지 않으면 로드 조작은 마지막으로 도달한 성공 지점(로드, 빌드 또는 삭제 단계)의 시작부터 재시작됩니다.

XML 문서를 로드하는 경우 동작이 약간 달라집니다. SAVECOUNT 옵션은 XML 데이터를 로드하는 경우에 지원되지 않으므로 로드 단계 중 실패한 로드 조작은 조작 시작부터 재시작됩니다. 다른 데이터 유형과 마찬가지로 빌드 단계 중 로드에 실패하면 REBUILD 모드에서 인덱스가 빌드되므로 각 행의 모든 인덱스 키를 선택하도록 테이블이 스캔됩니다. 그러나 인덱스 키를 선택하도록 각 XML 문서도 스캔되어야 합니다.

키에 대해 XML 문서를 스캔하는 이 프로세스는 해당 문서를 다시 구문 분석해야 하므로 자원 소비가 많은 조작입니다. 또한 내부 XML 인덱스(예: 영역 및 경로 인덱스)를 먼저 재빌드해야 하므로 XDA 오브젝트 스캔도 요구됩니다.

로드 조작이 실패하는 상황을 수정했으면 load 명령을 다시 실행하십시오. 원래 명령과 동일한 매개변수를 정확히 지정하여 로드 유틸리티가 필수 임시 파일을 찾을 수 있도록 해야 합니다. 읽기 액세스를 승인하지 않으려는 경우는 제외됩니다. ALLOW READ ACCESS 옵션을 지정한 로드 조작은 ALLOW NO ACCESS 옵션으로 재시작할 수도 있습니다.

주: 로드 유틸리티에서 작성한 임시 파일은 삭제하거나 수정하지 마십시오.

다음 명령으로 인한 로드 조작이 실패하는 경우,

```
LOAD FROM file_name OF file_type
SAVECOUNT n
MESSAGES message_file
load_method
INTO target_tablename
```

지정된 로드 방법(load_method)을 RESTART 방법으로 교체하여 재시작할 수 있습니다.

```
LOAD FROM file_name OF file_type
SAVECOUNT n
MESSAGES message_file
RESTART INTO target_tablename
```

재시작할 수 없는 실패한 로드

조작에 관련된 테이블이 로드 재시작 불가능 테이블 상태인 경우 실패 또는 인터럽트된 조작은 재시작할 수 없습니다. 테이블은 다음 이유로 해당 상태가 됩니다.

- 성공적으로 재시작 또는 종료되지 않은 실패한 로드 조작 이후에 롤 포워드 조작이 수행됩니다.
- 테이블이 로드 진행 중 또는 로드 보류 테이블 상태인 동안 수행된 온라인 백업에서 리스토어 작업이 수행됩니다.

LOAD TERMINATE 또는 LOAD REPLACE 명령을 실행해야 합니다.

ALLOW READ ACCESS 로드 조작 재시작 또는 종료

ALLOW READ ACCESS 옵션을 지정하는 인터럽트되거나 취소된 로드 조작은 ALLOW READ ACCESS 옵션을 사용하여 재시작하거나 종료할 수 있습니다. ALLOW READ ACCESS 옵션을 사용하면 종료 또는 재시작을 진행하면서 다른 응용프로그램이 테이블 데이터를 쿼리할 수 있습니다. ALLOW READ ACCESS 모드에서 로드 조작을 수행할 때와 마찬가지로 데이터를 커미트하기 전에 테이블이 배타적으로 잠깁니다.

인덱스 오브젝트가 사용 불가능하거나 유효하지 않은 항목으로 표시되면 ALLOW READ ACCESS 모드에서 로드 재시작 또는 종료 조작은 허용되지 않습니다.

원래 로드 조작이 인덱스 복사 단계에서 인터럽트되거나 취소되면 인덱스가 손상되었을 수 있으므로 ALLOW READ ACCESS 모드에서 재시작 조작은 허용되지 않습니다.

ALLOW READ ACCESS 모드의 로드 조작이 로드 단계에서 인터럽트되거나 취소되면 로드 단계에서 재시작됩니다. 로드 단계 이외의 다른 모든 단계에서 인터럽트되거나 취소되면 빌드 단계에서 재시작됩니다. 원래 로드 조작이 ALLOW NO ACCESS 모드인 경우 원래 로드 조작이 해당 지점에 도달하고 인덱스가 유효하면 삭제 단계에서 재시작 조작이 수행됩니다. 인덱스가 유효하지 않은 항목으로 표시되는 경우 로드 유틸리티는 빌드 단계에서 로드 조작을 재시작합니다.

주: 모든 로드 재시작 조작은 INDEXING MODE INCREMENTAL 옵션이 지정된 경우에도 REBUILD 인덱스 모드를 선택합니다.

LOAD TERMINATE 명령을 실행하면 일반적으로 인터럽트되거나 취소된 로드 조작이 최소 대기 시간으로 롤백됩니다. 그러나 ALLOW READ ACCESS 및 INDEXING MODE INCREMENTAL이 지정된 로드 조작에서 LOAD TERMINATE 명령을 실행하면 로드 유틸리티에서 인덱스를 스캔하고 불일치를 정정하는 동안 대기 시간이 발생할 수 있습니다. 이 대기 시간 길이는 인덱스 크기에 따라 달라지며 로드 종료 조작에 대해 ALLOW READ ACCESS 옵션이 지정되었는지 여부에 상관없이 대기 시간이 발생합니다. 빌드 단계 이전에 원래 로드 조작에 실패한 경우 대기 시간이 발생하지 않습니다.

주: 인덱스에서 불일치를 정정할 때 발생하는 대기 시간은 인덱스를 유효하지 않은 항목으로 표시하고 재빌드하는 경우 발생하는 대기 시간보다 훨씬 더 짧습니다.

로드 재시작 조작은 테이블 상태가 로드 재시작 불가능인 테이블에서는 수행할 수 없습니다. 롤 포워드 조작 중 테이블은 로드 재시작 불가능 상태로 설정할 수 있습니다. 이는 로드 조작 종료 전의 특정 시점으로 롤 포워드하는 경우 또는 인터럽트되거나 취소된 로드 조작을 통해 롤 포워드하지만 로드 종료 또는 로드 재시작 조작 종료로 롤 포워드하지 않는 경우 발생할 수 있습니다.

로드 사본 위치 파일로 데이터 복구

DB2LOADREC 레지스트리 변수는 로드 사본 위치 정보로 파일을 식별하는 데 사용됩니다. 이 파일은 롤 포워드 복구 중 로드 사본을 찾는 데 사용됩니다.

DB2LOADREC에는 다음에 대한 정보가 있습니다.

- 미디어 유형
- 사용할 미디어 디바이스 수
- 테이블 로드 조작 중 생성된 로드 사본 위치

- 로드 사본의 파일 이름(해당하는 경우)

위치 파일이 없거나 파일에서 일치하는 항목을 찾지 못하면 로그 레코드의 정보가 사용됩니다.

파일의 정보는 롤 포워드 복구를 수행하기 전에 겹쳐쓰기될 수 있습니다.

주:

1. 다중 파티션 데이터베이스에서 db2set 명령을 사용하여 모든 데이터베이스 파티션 서버에 대해 **DB2LOADREC** 레지스트리 변수를 설정해야 합니다.
2. 다중 파티션 데이터베이스에서 로드 사본 파일이 각 데이터베이스 파티션 서버에 있어야 하며 파일 이름(경로 포함)이 동일해야 합니다.
3. **DB2LOADREC** 레지스트리 변수로 식별된 파일의 항목이 유효하지 않으면 이전 로드 사본 위치 파일을 사용하여 유효하지 않은 항목을 교체하는 데 필요한 정보를 제공합니다.

위치 파일에서 다음 정보가 제공됩니다. 처음 다섯 개 매개변수에 유효한 값이 있고 이를 사용하여 로드 사본을 식별합니다. 전체 구조는 기록된 각 로드 사본에서 반복됩니다. 예를 들면, 다음과 같습니다.

```
TIMestamp      19950725182542         * Time stamp generated at load time
DBPartition    0                      * DB Partition number (OPTIONAL)
SCHema         PAYROLL                * Schema of table loaded
TABlename      EMPLOYEES              * Table name
DATabasename   DBT                    * Database name
DB2instance    toronto                * DB2INSTANCE
BUFfernumber   NULL                   * Number of buffers to be used for
                                        recovery
SESsionnumber  NULL                   * Number of sessions to be used for
                                        recovery
TYPeofmedia    L                      * Type of media - L for local device
                                                       A for TSM
                                                       O for other vendors
LOCationnumber 3                      * Number of locations
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
DBP            18
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

주:

1. 각 키워드에서 처음 3자는 매우 중요합니다. 모든 키워드는 지정된 순서로 있어야 합니다. 공백 라인은 허용되지 않습니다.
2. 시간소인은 *yyyymmddhhmmss* 양식입니다.
3. 모든 필드는 필수 필드입니다(BUF 및 SES(널(NULL)일 수 있음) 및 DBP(목록에서 누락될 수 있음)는 예외). SES가 널(NULL)이면 *dft_loadrec_ses* 구성 매개변수에 지정된 값이 사용됩니다. BUF가 널(NULL)이면 디폴트값은 SES+2입니다.
4. 위치 파일의 항목 중 하나라도 유효하지 않으면 이전 로드 사본 위치 파일을 사용하여 해당 값을 제공합니다.
5. 미디어 유형은 로컬 디바이스(테이프, 디스크 또는 디스켓의 경우 L), TSM(A) 또는 기타 벤더(O)일 수 있습니다. 유형이 L이면 위치 항목 앞에 위치 수가 나와야 합니다. 유형이 A이면 추가 입력은 필요하지 않습니다. 유형이 O이면 공유 라이브러리 이름이 필요합니다.
6. SHRlib 매개변수는 로드 사본 데이터를 저장하는 함수가 있는 라이브러리를 가리킵니다.
7. COPY NO 또는 NONRECOVERABLE 옵션을 지정하여 로드 조작을 호출하고 조작을 완료한 후 데이터베이스 또는 관련된 테이블 스페이스의 백업 사본을 가져오지 않으면 로드 조작 이후 특정 시점으로 데이터베이스 또는 테이블 스페이스를 리스토어할 수 없습니다. 즉, 로드 조작 이후 상태로 데이터베이스 또는 테이블 스페이스를 재작성하기 위해 롤 포워드 복구를 사용할 수 없습니다. 로드 조작 이전의 특정 시점으로만 데이터베이스 또는 테이블 스페이스를 리스토어할 수 있습니다.

특정 로드 사본을 사용하려는 경우 데이터베이스의 복구 실행기록 파일을 사용하여 특정 로드 조작의 시간소인을 판별할 수 있습니다. 다중 파티션 데이터베이스에서 복구 실행기록 파일은 각 데이터베이스 파티션에 로컬로 존재합니다.

덤프 파일 로드

dumpfile 파일 유형 수정자를 지정하면 거부된 행이 작성되는 예외 파일의 이름 및 위치를 로드 유틸리티에 알립니다.

파티션된 데이터베이스 환경에서 실행하는 경우 파티셔닝 서브에이전트 또는 로딩 서브에이전트에서 행을 거부할 수 있습니다. 이 때문에 덤프 파일 이름에 서브에이전트 유형을 식별하는 확장자 및 예외가 생성되는 데이터베이스 파티션 번호가 지정됩니다. 예를 들어 다음 덤프 파일 값을 지정하는 경우

```
dumpfile = "/u/usrname/dumpit"
```

데이터베이스 파티션 5의 로드 서브에이전트에서 거부된 행은 /u/usrname/dumpit.load.005 파일에 저장되고 데이터베이스 파티션 2의 로드 서브에이전트에서

거부된 행은 /u/usrname/dumpit.load.002 파일에 저장되며 데이터베이스 파티션 2의 파티셔닝 서브에이전트에서 거부된 행은 /u/usrname/dumpit.part.002 파일에 저장되는 식입니다.

로드 서브에이전트에서 거부된 행의 경우 행 길이가 32 768바이트 미만이면 레코드는 완전히 덤프 파일에 복사되고 길이가 더 길면 행 조각(레코드의 최종 바이트 포함)이 파일에 작성됩니다.

파티셔닝 서브에이전트에서 거부된 행의 경우 레코드 크기에 상관없이 전체 행이 덤프 파일에 복사됩니다.

임시 파일 로드

DB2에서는 로드 처리 중 임시 2진 파일을 작성합니다. 이 파일은 로드 응급 복구, 로드 종료 조작, 경고와 오류 메시지 및 런타임 제어 데이터에 사용됩니다.

로드 임시 파일은 로드 조작이 오류 없이 완료되면 제거됩니다. 임시 파일은 LOAD 명령의 *temp-pathname* 매개변수 또는 db2Load API의 *piTempFilesPath* 매개변수를 통해 지정될 수 있는 경로에 작성됩니다. 디폴트 경로는 데이터베이스 디렉토리의 서브디렉토리입니다.

임시 파일 경로는 서버 머신에 있으며 DB2 인스턴스에서 독점적으로 액세스됩니다. 따라서 *temp-pathname* 매개변수에 지정된 경로 이름 자격은 클라이언트가 아닌 서버의 디렉토리 구조를 반영해야 하며 DB2 인스턴스 소유자는 경로에 대한 읽기 및 쓰기 권한이 있어야 합니다.

주: MPP 시스템의 경우 임시 파일 경로는 NFS 마운트가 아닌 로컬 디스크에 있어야 합니다. 경로가 NFS 마운트에 있으면 로드 조작 중 성능이 크게 떨어집니다.
경고: 이 경로에 작성된 임시 파일은 어떤 경우에도 변경해서는 안됩니다. 변경하면 로드 조작이 오작동하고 데이터베이스가 손상될 수 있습니다.

로드 유틸리티 로그 레코드

유틸리티 관리 프로그램에서는 로드 유틸리티를 포함하여 여러 DB2 유틸리티와 관련된 로그 레코드를 생성합니다.

다음 로그 레코드는 로드 조작 중 특정 활동의 시작 또는 종료를 표시합니다.

- **설정 단계**
  - 로드 시작. 이 로그 레코드는 로드 조작의 설정 단계 시작을 나타냅니다.
  - 로그 레코드 커미트. 이 로그 레코드는 설정 단계의 완료를 나타냅니다.
  - 로그 레코드 중단. 이 로그 레코드는 설정 단계의 실패를 나타냅니다. 또는 단일 파티션 데이터베이스에서 로드 설정 단계가 실제로 테이블을 수정하기 전에 실패한 경우 로컬 보류 커미트 로그 레코드를 생성합니다.

- **로드 단계**
  - 로드 시작. 이 로그 레코드는 로드 조작의 로드 단계 시작을 나타냅니다.
  - 로컬 보류 커미트 로그 레코드. 이 로그 레코드는 로드 단계의 완료를 나타냅니다.
  - 로그 레코드 중단. 이 로그 레코드는 로드 단계의 실패를 나타냅니다.
- **삭제 단계**
  - 로드 삭제 시작. 이 로그 레코드는 로드 조작의 삭제 단계 시작과 연관됩니다. 삭제 단계는 중복 1차 키 값이 있는 경우에만 시작됩니다. 삭제 단계 중 테이블 레코드 또는 인덱스 키의 각 삭제 조작이 로그됩니다.
  - 로드 삭제 종료. 이 로그 레코드는 로드 조작의 삭제 단계 종료와 연관됩니다. 성공적인 로드 조작의 롤 포워드 복구 중에 이 삭제 단계가 반복됩니다.

다음 목록에서는 입력 데이터 *크기*에 따라 로드 유틸리티에서 작성하는 로그 레코드를 간략히 설명합니다.

- DMS 테이블 스페이스의 유틸리티에서 삭제 또는 할당된 모든 테이블 스페이스 범위에 대해 두 개의 로그 레코드가 작성됩니다.
- 사용된 ID 값의 모든 청크에 대해 하나의 로그 레코드가 작성됩니다.
- 로드 조작의 삭제 단계 중 삭제된 모든 데이터 행 또는 인덱스 키에 대해 로그 레코드가 작성됩니다.
- ALLOW READ ACCESS 및 INDEXING MODE INCREMENTAL 옵션을 지정하여 로드 조작을 수행하는 중 인덱스 트리의 무결성을 유지보수하는 로그 레코드가 작성됩니다. 로그된 레코드 수는 인덱스로 완전히 로그된 삽입 수보다 훨씬 적습니다.

로드 개요 - 파티션된 데이터베이스 환경

다중 파티션 데이터베이스에서 많은 데이터가 많은 데이터베이스 파티션에 교차하여 있습니다. 분산 키는 데이터의 각 부분이 있는 데이터베이스 파티션을 판별하는 데 사용됩니다. 데이터를 올바른 데이터베이스 파티션으로 로드하려면 데이터를 *분산*해야 합니다.

다중 파티션 데이터베이스에서 데이터를 로드하는 경우 로드 유틸리티에서는 다음을 수행할 수 있습니다.

- 입력 데이터를 병렬로 분산
- 해당 데이터베이스 파티션에서 동시에 데이터 로드
- 한 시스템에서 다른 시스템으로 데이터 전송

데이터를 다중 파티션 데이터베이스로 로드하는 작업은 테이블 잠금과 같은 데이터베이스 파티션 자원을 획득하는 *설정 단계*와 데이터베이스 파티션으로 데이터를 로드하는 로

드 *단계*의 두 단계로 수행됩니다. LOAD 명령의 ISOLATE_PART_ERRS 옵션을 사용하여 이 단계 중 하나를 수행하는 동안 오류를 처리하는 방법 및 하나 이상의 데이터베이스 파티션에서 발생한 오류가 오류가 발생하지 않은 데이터베이스 파티션의 로드 조작에 영향을 미치는 방법을 선택할 수 있습니다.

다중 파티션 데이터베이스로 데이터를 로드하는 경우 다음 방법 중 하나를 사용할 수 있습니다.

PARTITION_AND_LOAD
데이터가 해당 데이터베이스 파티션에서 동시에 분산(병렬 방식일 수 있음) 및 로드됩니다.

PARTITION_ONLY
데이터가 분산되고(병렬 방식일 수 있음) 로드하는 각 데이터베이스 파티션에서 지정된 위치의 파일에 작성됩니다. 각 파일은 여러 데이터베이스 파티션에서 데이터를 분산하는 방법을 지정하고 LOAD_ONLY 모드를 사용하여 데이터베이스로 파일을 로드할 수 있는 파티션 헤더를 포함합니다.

LOAD_ONLY
데이터는 여러 데이터베이스 파티션에서 이미 분산되었다고 가정합니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다.

LOAD_ONLY_VERIFY_PART
데이터는 여러 데이터베이스 파티션에서 이미 분산되었다고 가정합니다. 그러나 데이터 파일은 파티션 헤더를 포함하지 않습니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. 로드 조작 중에 각 행은 올바른 데이터베이스 파티션에 있는지 확인하기 위해 점검됩니다. 데이터베이스 파티션 위반을 포함하는 행은 `dumpfile` 파일 유형 수정자가 지정된 경우 덤프 파일에 배치됩니다. 그렇지 않으면 행을 버립니다. 데이터베이스 파티션 위반이 로드하는 특정 데이터베이스 파티션에 있으면 해당 데이터베이스 파티션에 대한 단일 경고가 로드 메시지 파일에 작성됩니다.

ANALYZE
모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다.

개념 및 용어

다음 용어는 다중 데이터베이스 파티션을 포함하는 파티션된 데이터베이스 환경에서 로드 유틸리티의 동작 및 조작을 논의할 때 사용됩니다.

- *코디네이터 파티션*은 로드 조작을 수행하기 위해 연결하는 데이터베이스 파티션입니다. PARTITION_AND_LOAD, PARTITION_ONLY 및 ANALYZE 모드의 경

우 LOAD 명령의 CLIENT 옵션을 지정하지 않는 한, 데이터 파일은 이 데이터베이스 파티션에 있다고 가정합니다. CLIENT를 지정하면 로드할 데이터가 리모트로 연결된 클라이언트에 있음을 표시합니다.

- PARTITION_AND_LOAD, PARTITION_ONLY 및 ANALYZE 모드의 경우 *사전 파티셔닝 에이전트*는 사용자 데이터를 읽고 데이터를 분산하는 *파티셔닝 에이전트*에 라운드 로빈 방식으로 데이터를 분산합니다. 이 프로세스는 항상 코디네이터 파티션에서 수행됩니다. 로드 조작에 대해 데이터베이스 파티션당 최대 하나의 파티셔닝 에이전트가 허용됩니다.
- PARTITION_AND_LOAD, LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART 모드의 경우 로드 *에이전트*는 각 출력 데이터베이스 파티션에서 실행되며 해당 데이터베이스 파티션으로 데이터를 로드하는 작업을 조정합니다.
- *파일 에이전트*로 로드는 PARTITION_ONLY 로드 조작 중에 각 출력 데이터베이스 파티션에서 실행됩니다. 파티셔닝 에이전트에서 데이터를 받아 해당 데이터베이스 파티션의 파일에 작성합니다.
- SOURCEUSEREXIT 옵션에서는 로드 유틸리티가 사용자 정의 스크립트 또는 실행 파일(여기서 *User Exit*로 참조됨)을 실행할 수 있도록 하는 기능을 제공합니다.

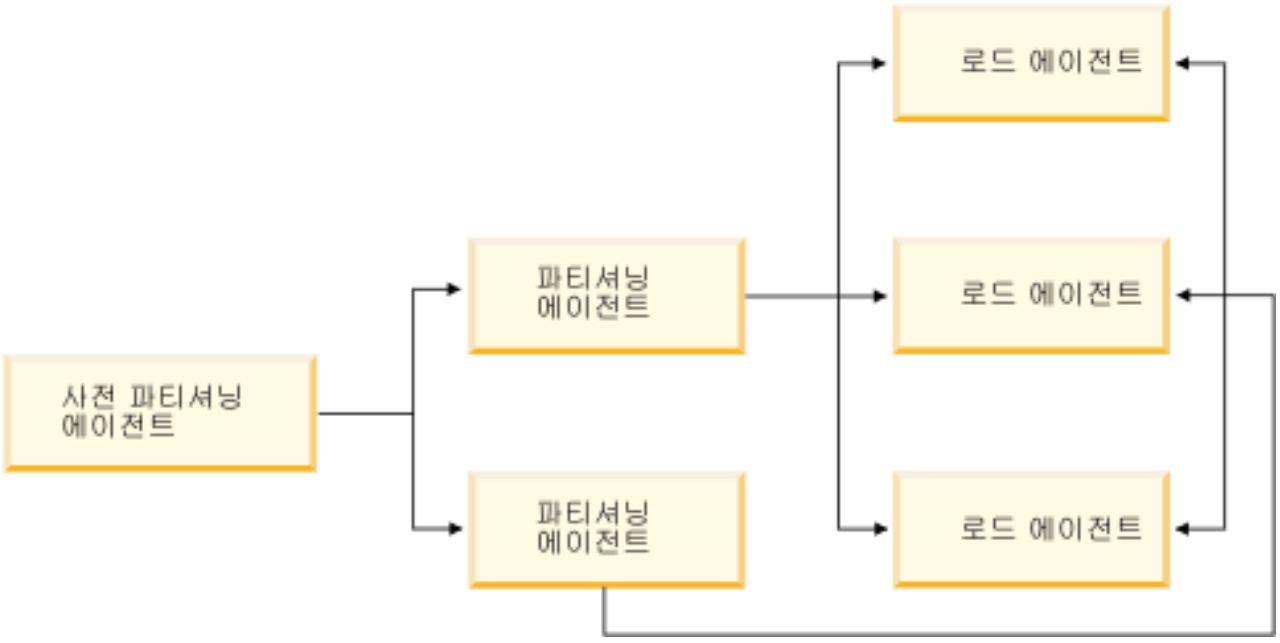


그림 13. 파티션된 데이터베이스 로드 개요. 사전 파티셔닝 에이전트가 소스 데이터를 읽으며 약 절반의 데이터가 두 파티셔닝 에이전트 중 각각에 전송되어 데이터를 분산하고 세 개의 데이터베이스 파티션 중 하나로 전송합니다. 각 데이터베이스 파티션의 로드 에이전트에서 데이터를 로드합니다.

파티션된 데이터베이스 환경에서 데이터 로드

로드 유틸리티를 사용하여 파티션된 데이터베이스 환경으로 데이터를 로드합니다.

다중 파티션 데이터베이스로 테이블을 로드하기 전에:

1. *svcename* 데이터베이스 관리 프로그램 구성 매개변수 및 **DB2COMM** 프로파일 레지스트리 변수를 올바르게 설정해야 합니다. 이는 로드 유틸리티가 사전 파티셔닝 에이전트에서 파티셔닝 에이전트로, 파티셔닝 에이전트에서 로드하는 데이터베이스 파티션으로 데이터를 전송하기 위해 TCP/IP를 사용하므로 중요합니다.
2. 로드 유틸리티를 호출하기 전에 데이터를 로드할 데이터베이스에 연결하거나 내재적으로 연결할 수 있어야 합니다. 로드 유틸리티는 COMMIT문을 실행하므로 로드 조작을 시작하기 전에 COMMIT 또는 ROLLBACK문을 실행하여 모든 트랜잭션을 완료하고 잠금을 해제해야 합니다. PARTITION_AND_LOAD, PARTITION_ONLY 또는 ANALYZE 모드를 사용하는 경우 로드할 데이터 파일은 이 데이터베이스 파티션에 있어야 합니다. 이때 다음은 예외 조건입니다.
   a. CLIENT 옵션이 지정된 경우. 이때 데이터는 클라이언트 머신에 있어야 합니다.
   b. 입력 소스 유형이 CURSOR인 경우. 이때 입력 파일은 없습니다.
3. 각 테이블에 대해 최상의 데이터베이스 파티션을 판별하도록 디자인 어드바이저를 실행합니다. 자세한 정보는 *문제점 해결 및 데이터베이스 성능 조정*의 『디자인 어드바이저』를 참조하십시오.

다음 제한사항은 다중 파티션 데이터베이스에서 로드 유틸리티를 사용하여 데이터를 로드하는 경우 적용됩니다.

- 로드 조작에 대한 입력 파일 위치는 테이프 디바이스일 수 없습니다.
- ANALYZE 모드를 사용하지 않는 한, ROWCOUNT 옵션은 지원되지 않습니다.
- 분산에 필요한 ID 컬럼이 목표 테이블에 있고 `identityoverride` 파일 유형 수정자가 지정되지 않은 경우 또는 다중 데이터베이스 파티션을 사용하여 데이터를 분산하고 로드하는 경우 LOAD 명령에서 0보다 큰 SAVECOUNT를 사용하는 작업은 지원되지 않습니다.
- ID 컬럼이 분산 키의 일부를 구성하는 경우 PARTITION_AND_LOAD 모드만 지원됩니다.
- LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART 모드는 LOAD 명령의 CLIENT 옵션과 함께 사용할 수 없습니다.
- LOAD_ONLY_VERIFY_PART 모드는 CURSOR 입력 소스 유형에서 사용할 수 없습니다.
- 분산 오류 분리 모드 LOAD_ERRS_ONLY 및 SETUP_AND_LOAD_ERRS는 LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션과 함께 사용할 수 없습니다.

- OUTPUT_DBPARTNUMS 및 PARTITIONING_DBPARTNUMS 옵션으로 지정된 데이터베이스 파티션이 오버랩되지 않는 경우 다중 로드 조작을 동시에 실행하여 동일한 테이블로 데이터를 로드할 수 있습니다. 예를 들어 데이터베이스 파티션 0 - 3까지 테이블이 정의된 경우 하나의 로드 조작이 데이터베이스 파티션 0 및 1로 데이터를 로드하는 동안, 두 번째 로드 조작은 데이터베이스 파티션 2 및 3으로 데이터를 로드할 수 있습니다.
- 컬럼 식별자 없는 ASCII(ASC) 및 컬럼 식별자가 있는 ASCII(DEL) 파일만 다중 데이터베이스 파티션에서 분산될 수 있습니다. PC/IXF 파일은 분산될 수 없지만 LOAD_ONLY_VERIFY_PART 모드에서 로드 조작을 사용하여 다중 데이터베이스 파티션에 분산된 테이블로 PC/IXF 파일을 로드할 수 있습니다.

다음 예에서는 LOAD 명령을 사용하여 다양한 유형의 로드 조작을 시작하는 방법을 보여줍니다. 다음 예에서 사용하는 데이터베이스에는 5개의 데이터베이스 파티션(0, 1, 2, 3, 4)이 있습니다. 각 데이터베이스 파티션에는 로컬 디렉토리 /db2/data/가 있습니다. 2개 테이블, TABLE1 및 TABLE2가 데이터베이스 파티션 0, 1, 3 및 4에 정의되어 있습니다. 클라이언트에서 로드하는 경우 데이터베이스 파티션에 포함되지 않은 리모트 클라이언트에 대한 액세스 권한이 사용자에게 부여됩니다.

서버 파티션에서 로드

분산 및 로드 예

이 시나리오에서는 데이터베이스 파티션에 연결됩니다. 이 데이터베이스 파티션이 TABLE1이 정의된 데이터베이스 파티션인지 여부와는 상관이 없습니다. 데이터 파일 load.del은 이 데이터베이스 파티션의 현재 작업 디렉토리에 있습니다. load.del에서 TABLE1이 정의된 모든 데이터베이스 파티션으로 데이터를 로드하려면 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

주: 이 예에서 파티션된 데이터베이스 환경의 모든 구성 매개변수에 대해 디폴트값이 사용됩니다. MODE 매개변수는 디폴트로 PARTITION_AND_LOAD로 설정되며 OUTPUT_DBPARTNUMS 옵션은 디폴트로 TABLE1이 정의된 모든 데이터베이스 파티션으로 설정됩니다. PARTITIONING_DBPARTNUMS는 디폴트로 아무것도 지정하지 않았을 때 데이터베이스 파티션 선택 시 LOAD 명령 규칙에 따라 선택된 데이터베이스 파티션으로 설정됩니다.

데이터베이스 파티션 3 및 4에서 데이터가 분산되는 로드 조작을 수행하려면 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

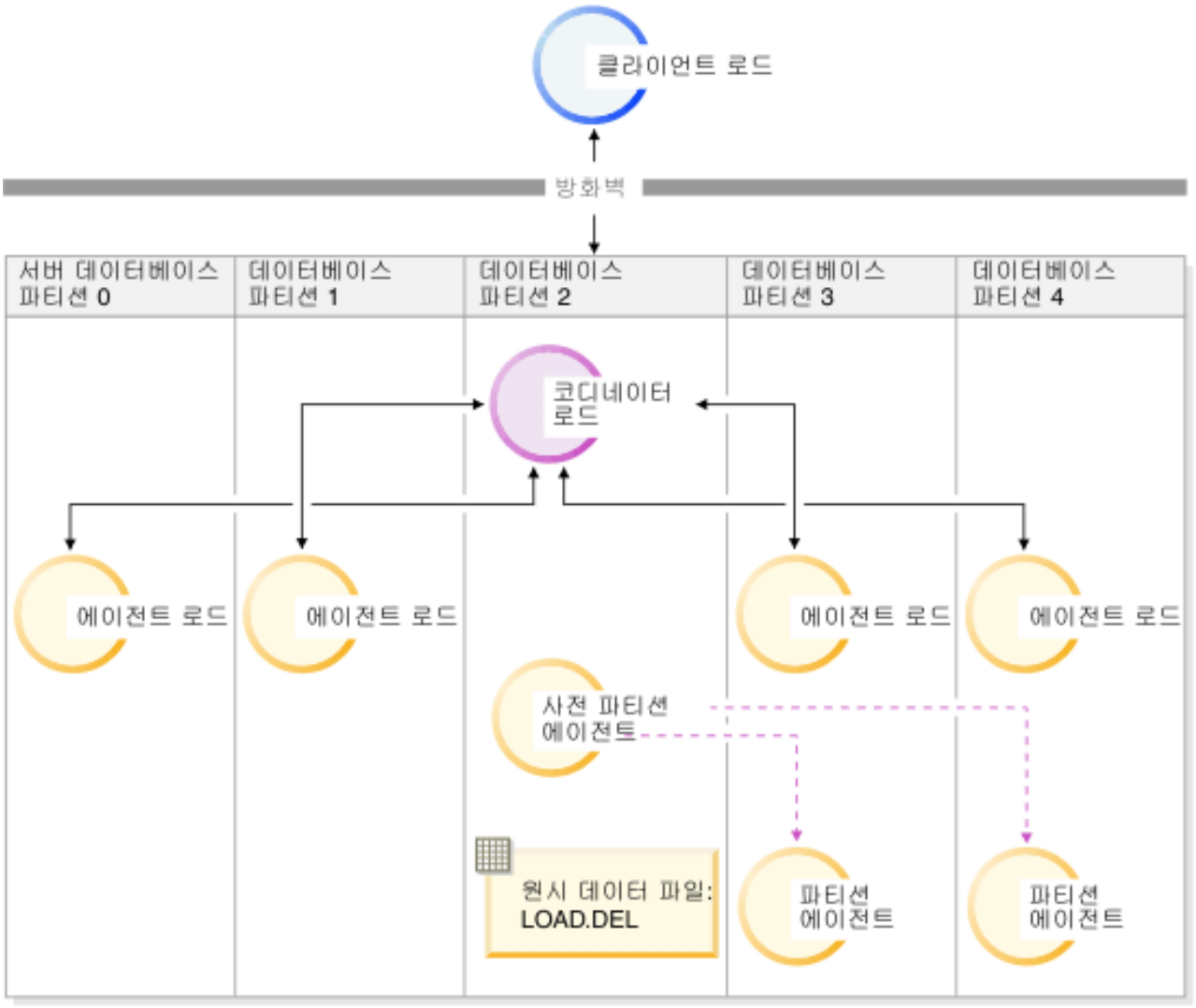


그림 14. 데이터베이스 파티션 3 및 4로 데이터 로드. 이 그림에서는 이전 명령을 실행할 때 나타나는 동작을 보여줍니다. 데이터는 데이터베이스 파티션 3 및 4로 로드됩니다.

분산만 수행하는 예

이 시나리오에서는 데이터베이스 파티션에 연결됩니다. 이 데이터베이스 파티션이 TABLE1이 정의된 데이터베이스 파티션인지 여부와는 상관이 없습니다. 데이터 파일 load.del은 이 데이터베이스 파티션의 현재 작업 디렉토리에 있습니다. 데이터베이스 파티션 3 및 4를 사용하여 load.del을 TABLE1이 정의된 모든 데이터베이스 파티션에 분산(로드가 아님)하려면 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                          PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                                PART_FILE_LOCATION /db2/data
                                                PARTITIONING_DBPARTNUMS (3,4)
```

그러면 load.del.xxx 파일이 각 데이터베이스 파티션의 /db2/data 디렉토리에 저장됩니다. 여기서 xxx는 데이터베이스 파티션 번호의 3자리 표시입니다.

데이터베이스 파티션 0에서 실행하는 파티셔닝 에이전트 하나만 사용하여 load.del 파일을 데이터베이스 파티션 1 및 3에 분산하려면(PARTITIONING_DBPARTNUMS의 디폴트값임) 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                        PART_FILE_LOCATION /db2/data
                                        OUTPUT_DBPARTNUMS (1,3)
```

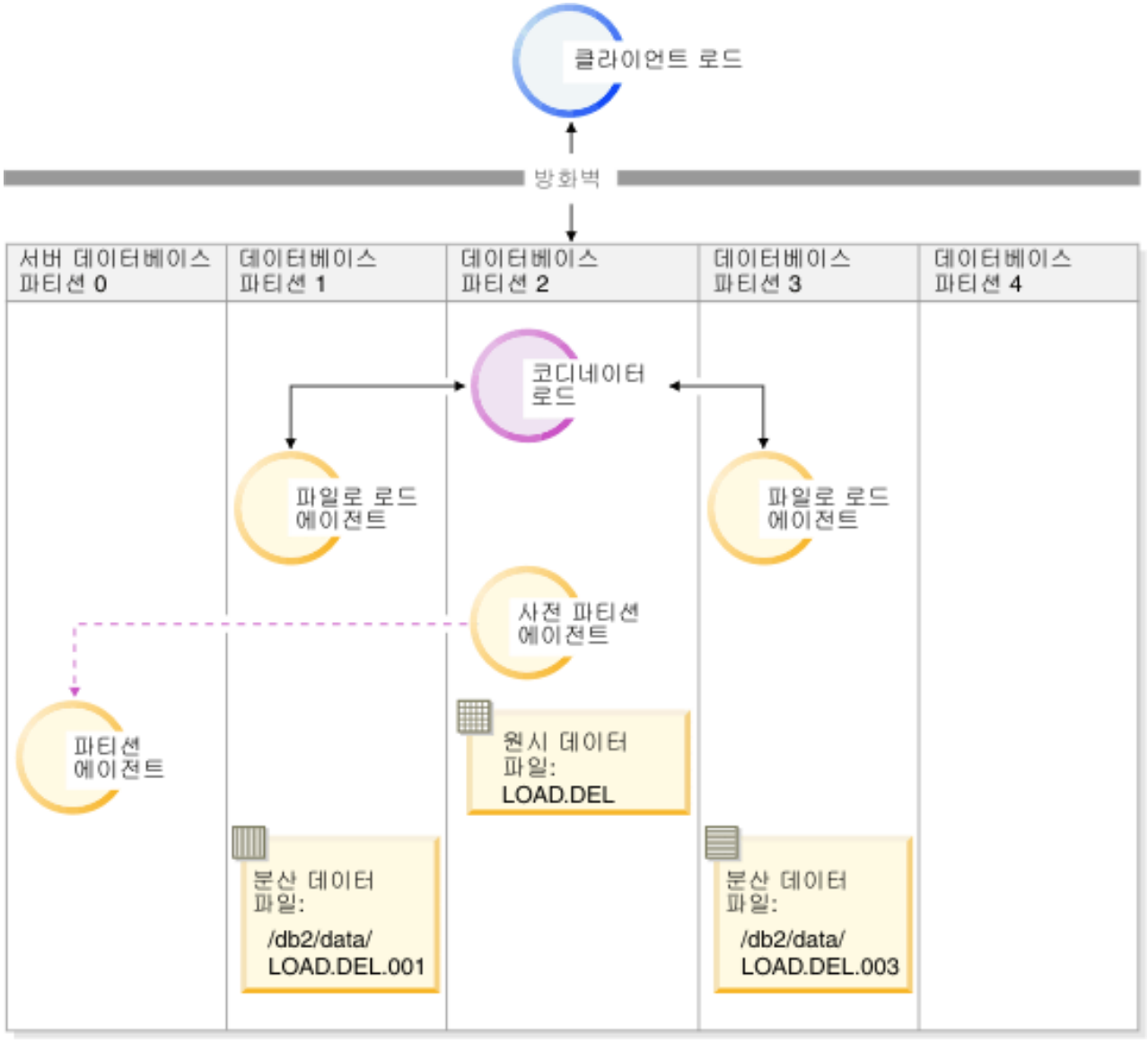


그림 15. 하나의 파티셔닝 에이전트를 사용하여 데이터베이스 파티션 1 및 3으로 데이터 로드. 이 그림에서는 이전 명령을 실행할 때 결과로 나타나는 동작을 보여줍니다. 데이터는 데이터베이스 파티션 0에서 실행하는 하나의 파티셔닝 에이전트를 사용하여 데이터베이스 파티션 1 및 3으로 로드됩니다.

로드만 수행하는 예

이미 PARTITION_ONLY 모드로 로드 조작을 수행했으며 로드하는 각 데이터베이스 파티션의 /db2/data 디렉토리에 있는 파티션된 파일을 TABLE1이 정의된 모든 데이터베이스 파티션으로 로드하려는 경우 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
                                    PART_FILE_LOCATION /db2/data
```

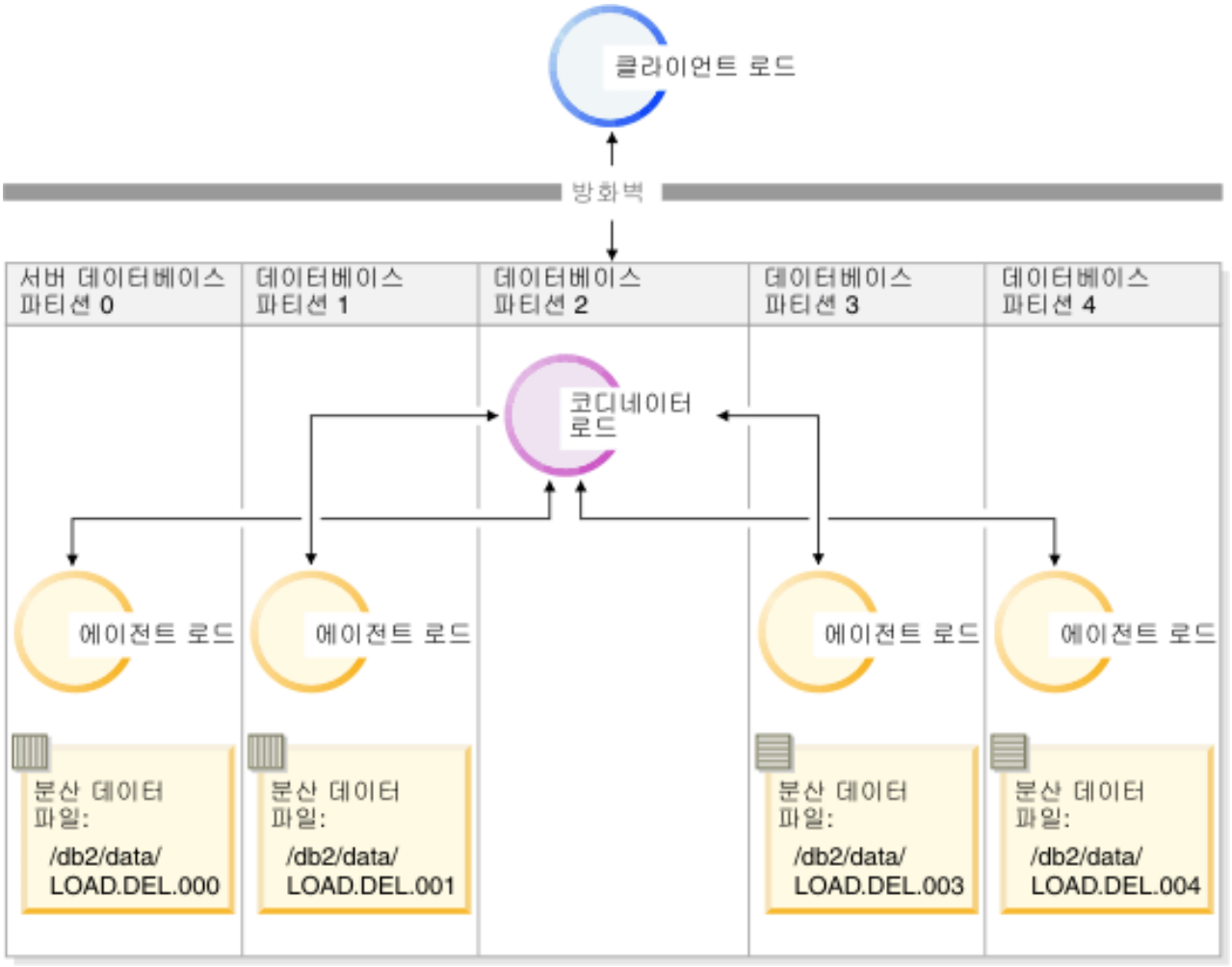


그림 16. 특정 테이블이 정의된 모든 데이터베이스 파티션으로 데이터 로드. 이 그림에서는 이전 명령을 실행할 때 나타나는 동작을 보여줍니다. 분산된 데이터는 TABLE1이 정의된 모든 데이터베이스 파티션으로 로드됩니다.

데이터베이스 파티션 4로만 로드하려면 다음 명령을 실행하십시오.

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
                                    PART_FILE_LOCATION /db2/data
                                    OUTPUT_DBPARTNUMS (4)
```

분산 맵 헤더 없이 사전 분산된 파일 로드

LOAD 명령을 사용하여 분산 헤더 없이 여러 데이터베이스 파티션으로 직접 데이터 파일을 로드할 수 있습니다. 데이터 파일이 TABLE1이 정의된 각 데이터베이스 파티션의 /db2/data 디렉토리에 있고 이름이 load.del.xxx(여기서 xxx는 데이터베이스 파티션 번호)이면 다음 명령을 실행하여 파일을 로드할 수 있습니다.

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                          REPLACE INTO TABLE1
                          PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                PART_FILE_LOCATION /db2/data
```

데이터베이스 파티션 1로만 데이터를 로드하려면 다음 명령을 실행하십시오.

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                          REPLACE INTO TABLE1
                          PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                PART_FILE_LOCATION /db2/data
                                                OUTPUT_DBPARTNUMS (1)
```

주: 로드된 데이터베이스 파티션에 속하지 않은 행은 거부되고 덤프 파일이 지정된 경우 이 파일에 배치됩니다.

리모트 클라이언트에서 다중 파티션 데이터베이스로 로드

리모트 클라이언트에 있는 파일에서 다중 파티션 데이터베이스로 데이터를 로드하려면 데이터 파일이 서버 파티션에 없음을 표시하도록 LOAD 명령의 CLIENT 옵션을 지정해야 합니다. 예를 들면, 다음과 같습니다.

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

주: LOAD_ONLY 또는 LOAD_ONLY_VERIFY_PART 모드는 CLIENT 옵션과 함께 사용할 수 없습니다.

cursor에서 로드

단일 파티션 데이터베이스에서는 cursor에서 다중 파티션 데이터베이스로 로드할 수 있습니다. 이 예에서 PARTITION_ONLY 및 LOAD_ONLY 모드의 경우 PART_FILE_LOCATION 옵션에서는 완전한 파일 이름을 지정해야 합니다. 이 이름은 각 출력 데이터베이스 파티션에서 로드 또는 작성된 분산 파일의 완전한 기본 파일 이름입니다. 목표 테이블에 LOB 컬럼이 있는 경우 지정된 기본 이름으로 다중 파일을 작성할 수 있습니다.

다음에 TABLE2로 로드하는 경우 SELECT * FROM TABLE1문의 응답 세트에 있는 모든 행을 이름이 /db2/data/select.out.xxx(여기서 xxx는 데이터베이스 파티션 번호임)인 각 데이터베이스 파티션으로 분산하려면 다음 명령을 실행합니다.

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1
```

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                        PART_FILE_LOCATION /db2/data/select.out
```

위 조작으로 생성된 데이터 파일은 다음 LOAD 명령을 실행하여 로드할 수 있습니다.

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                        PARTITIONED CB CONFIG MODE LOAD_ONLY
                        PART_FILE_LOCATION /db2/data/select.out
```

파티션된 데이터베이스 환경에서 데이터 로드 - 힌트 및 추가 정보

다음은 다중 파티션 데이터베이스에서 테이블을 로드하기 전에 고려할 몇 가지 정보입니다.

- 적은 양의 데이터를 기반으로 유틸리티를 사용하여 로드 구성 옵션에 익숙해져야 합니다.
- 입력 데이터가 이미 정렬되었거나 일부 선택된 순서로 배치된 경우 로드 프로세스 중 해당 순서를 유지하려면 분산 시 하나의 데이터베이스 파티션만 사용해야 합니다. 병렬 분산으로는 데이터를 수신과 동일한 순서로 로드하도록 보장할 수 없습니다. 로드 유틸리티는 anyorder 수정자가 LOAD 명령에 지정되지 않은 경우 디폴트로 단일 파티셔닝 에이전트를 선택합니다.
- 대형 오브젝트(LOB)가 별도의 파일에서 로드되는 경우(즉, 로드 유틸리티를 통해 lobsinfile 수정자를 사용하는 경우) LOB 파일을 포함하는 모든 디렉토리는 로드 수행 시 모든 데이터베이스 파티션에 대한 읽기 액세스 권한이 있어야 합니다. LOB에 대한 작업을 수행할 때 LOAD *lob-path* 매개변수는 완전한 형식이어야 합니다.
- 로드 조작 시작 시 로드하는 일부 데이터베이스 파티션 또는 연관된 테이블 스페이스나 테이블이 오프라인임을 발견한 경우에도 ISOLATE_PART_ERRS 옵션을 SETUP_ERRS_ONLY 또는 SETUP_AND_LOAD_ERRS로 설정하여 다중 파티션 데이터베이스에서 작업을 강제로 계속 실행할 수 있습니다.
- STATUS_INTERVAL 로드 구성 옵션을 사용하여 다중 파티션 데이터베이스에서 실행하는 작업의 진행을 모니터합니다. 로드 조작 시 지정된 간격으로 사전 파티셔닝 에이전트에서 읽은 데이터 *크기*(MB)를 표시하는 메시지가 생성됩니다. 이 메시지는 사전 파티셔닝 에이전트 메시지 파일로 덤프됩니다. 로드 조작 중 이 파일의 컨텐츠를 보려면 코디네이터 파티션에 연결하고 목표 테이블에서 LOAD QUERY 명령을 실행합니다.
- 분산 프로세스에 참여하는 데이터베이스 파티션(PARTITIONING_DBPARTNUMS 옵션에서 정의함)이 로드하는 데이터베이스 파티션(OUTPUT_DBPARTNUMS 옵션에서 정의함)과 다른 경우 CPU 주기에 대한 경합이 적으므로 더 나은 성능이 예상됩니다. 다중 파티션 데이터베이스로 데이터를 로드하는 경우 분산 또는 로드 조작에 참여하지 않는 데이터베이스 파티션에서 로드 유틸리티를 호출합니다.

- LOAD 명령에서 MESSAGES 매개변수를 지정하면 로드 조작 종료 시 참조할 메시지 파일을 사전 파티셔닝, 파티셔닝 및 로드 에이전트에서 저장합니다. 로드 조작 중 이 파일의 컨텐츠를 보려면 원하는 데이터베이스 파티션에 연결하고 목표 테이블에서 LOAD QUERY 명령을 실행합니다.
- 로드 유틸리티는 통계를 수집할 하나의 출력 데이터베이스 파티션만 선택합니다. RUN_STAT_DBPARTNUM 데이터베이스 구성 옵션은 데이터베이스 파티션을 지정하는 데 사용할 수 있습니다.
- 다중 파티션 데이터베이스로 데이터를 로드하기 전에 디자인 어드바이저를 실행하여 각 테이블에 대한 최상의 파티션을 판별합니다. 자세한 정보는 *문제점 해결 및 데이터베이스 성능 조정*의 『디자인 어드바이저』를 참조하십시오.

문제점 해결

로드 유틸리티가 정지되면 다음을 수행할 수 있습니다.

- STATUS_INTERVAL 매개변수를 사용하여 다중 파티션 데이터베이스 로드 조작의 진행을 모니터합니다. 상태 간격 정보는 코디네이터 파티션의 사전 파티셔닝 에이전트 메시지 파일로 덤프됩니다.
- 파티셔닝 에이전트 메시지 파일에서 각 데이터베이스 파티션의 파티셔닝 에이전트 프로세스 상태를 확인합니다. 오류 없이 로드가 진행되는 경우 TRACE 옵션이 설정되면 이 메시지 파일에서 여러 레코드에 대한 추적 메시지가 있어야 합니다.
- 로드 메시지 파일에서 로드 오류 메시지가 있는지 확인합니다.

  주: 이 파일이 존재하려면 LOAD 명령의 MESSAGES 옵션을 지정해야 합니다.
- 로드 프로세스 중 하나에서 오류가 발생했음을 알리는 오류가 발견되면 현재 로드 조작을 인터럽트합니다.

LOAD QUERY 명령을 사용하여 파티션된 데이터베이스 환경에서 로드 조작 모니터링

파티션된 데이터베이스 환경에서 로드 조작 중 로드 프로세스를 실행하는 데이터베이스 파티션에서 일부 로드 프로세스로 메시지 파일이 작성됩니다.

로드 조작 실행 중 생성되는 모든 정보, 경고 및 오류 메시지는 메시지 파일에 저장됩니다. 사용자가 볼 수 있는 메시지 파일을 생성하는 로드 프로세스는 로드 에이전트, 사전 파티셔닝 에이전트 및 파티셔닝 에이전트입니다. 메시지 파일의 컨텐츠는 로드 조작을 완료한 후에만 볼 수 있습니다.

로드 조작 중 개별 데이터베이스 파티션에 연결하고 목표 테이블에서 LOAD QUERY 명령을 실행할 수 있습니다. CLP에서 명령을 실행한 경우 이 명령은 LOAD QUERY 명령에 지정된 테이블의 해당 데이터베이스 파티션에 현재 있는 모든 메시지 파일의 컨텐츠를 표시합니다.

예를 들어 테이블 TABLE1은 데이터베이스 WSDB의 데이터베이스 파티션 0 - 3에 정의되어 있습니다. 데이터베이스 파티션 0에 연결한 상태에서 다음 LOAD 명령을 실행하십시오.

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

이 명령은 데이터베이스 파티션 0, 1, 2, 3에서 실행하는 로드 에이전트, 데이터베이스 파티션 1에서 실행하는 파티셔닝 에이전트 및 데이터베이스 파티션 0에서 실행하는 사전 파티셔닝 에이전트를 포함하는 로드 조작을 시작합니다.

데이터베이스 파티션 0에는 사전 파티셔닝 에이전트에 대한 하나의 메시지 파일과 해당 데이터베이스 파티션의 로드 에이전트에 대한 메시지 파일 하나가 포함되어 있습니다. 이 파일의 컨텐츠를 동시에 보려면 새 세션을 시작하고 CLP에서 다음 명령을 실행하십시오.

```
set client connect_node 0
connect to wsdb
load query table table1
```

데이터베이스 파티션 1에는 로드 에이전트에 대한 하나의 파일과 파티셔닝 에이전트에 대한 하나의 파일이 포함되어 있습니다. 이 파일의 컨텐츠를 보려면 새 세션을 시작하고 CLP에서 다음 명령을 실행하십시오.

```
set client connect_node 1
connect to wsdb
load query table table1
```

주: STATUS_INTERVAL 로드 구성 옵션으로 생성된 메시지는 사전 파티셔닝 에이전트 메시지 파일에 표시됩니다. 로드 조작 중 이 메시지를 보려면 코디네이터 파티션에 연결하고 LOAD QUERY 명령을 실행해야 합니다.

메시지 파일의 컨텐츠 저장

db2Load API를 통해 로드 조작이 시작되면 메시지 옵션(piLocalMsgFileName)을 지정해야 합니다. 메시지 파일은 서버에서 클라이언트로 가져와 사용자가 볼 수 있도록 저장됩니다.

CLP로 시작된 다중 파티션 데이터베이스 로드 조작의 경우 메시지 파일은 콘솔에 표시되지 않으며 보유되지 않습니다. 다중 파티션 데이터베이스 로드가 완료된 후 이러한 파일의 컨텐츠를 보거나 저장하려면 LOAD 명령의 MESSAGES 옵션을 지정해야 합니다. 이 옵션을 사용하면 로드 조작을 완료한 후 각 데이터베이스 파티션의 메시지 파일이 클라이언트 머신으로 전송되고 MESSAGES 옵션에 표시된 기본 이름을 사용하여 파일에 저장됩니다. 다중 파티션 데이터베이스 로드 조작의 경우 로드 프로세스에 대응하여 생성된 파일 이름이 아래 나열되어 있습니다.

프로세스 유형	파일 이름
로드 에이전트	<message-file-name>.LOAD.<dbpartition-number>
파티셔닝 에이전트	<message-file-name>.PART.<dbpartition-number>
사전 파티셔닝 에이전트	<message-file-name>.PREP.<dbpartition-number>

예를 들어 MESSAGES 옵션에서 /wsdb/messages/load를 지정하면 데이터베이스 파티션 2의 로드 에이전트 메시지 파일은 /wsdb/messages/load.LOAD.002입니다.

주: CLP에서 시작된 다중 파티션 데이터베이스 로드 조작에서는 MESSAGES 옵션을 사용하는 것이 좋습니다.

파티션된 데이터베이스 환경에서 로드 조작 다시 시작, 재시작 또는 종료

파티션된 데이터베이스 환경에서 실패한 로드 조작 이후에 수행해야 하는 단계는 실패한 시점에 따라 달라집니다.

다중 파티션 데이터베이스에서 로드 프로세스는 다음 두 단계로 구성됩니다.

1. *설정 단계*에서는 출력 데이터베이스 파티션의 테이블 잠금과 같은 데이터베이스 파티션 레벨의 자원이 확보됩니다.

   일반적으로 설정 단계에서 실패하면 재시작 및 종료 조작은 필요하지 않습니다. 실패한 로드 조작에 지정된 오류 분리 모드에 따라 수행해야 하는 작업이 달라집니다.

   로드 조작에서 설정 단계 오류를 분리하지 않도록 지정한 경우 전체 로드 조작이 취소되고 각 데이터베이스 파티션의 테이블은 로드 조작 전의 상태로 롤백됩니다.

   로드 조작에서 설정 단계 오류를 분리하도록 지정한 경우 설정 단계가 성공하면 데이터베이스 파티션에서 로드 조작이 계속되지만 실패한 각 데이터베이스 파티션의 테이블은 로드 조작 전의 상태로 롤백됩니다. 즉, 일부 파티션이 설정 단계에서 실패하고 또 다른 파티션이 로드 단계에서 실패한 경우 단일 로드 조작이 서로 다른 단계에서 실패할 수 있음을 의미합니다.
2. 로드 *단계*에서는 데이터를 형식화하여 데이터베이스 파티션의 테이블로 로드합니다.

   다중 파티션 데이터베이스 로드 조작의 로드 단계 중 하나 이상의 데이터베이스 파티션에서 로드 조작에 실패하면 로드 RESTART 또는 TERMINATE 명령을 실행해야 합니다. 다중 파티션 데이터베이스에서의 데이터 로드는 단일 트랜잭션으로 수행되므로 이 작업이 필요합니다.

로드 실패 원인이 되는 문제점을 수정할 수 있으면 로드 RESTART를 선택해야 합니다. 로드 재시작 조작을 시작하면 모든 데이터베이스 파티션에서 중단점 지점부터 로드 조작이 계속되므로 시간을 절약할 수 있습니다.

초기 로드 조작 전의 상태로 테이블을 되돌리려면 로드 TERMINATE를 선택해야 합니다.

프로시저:

로드 실패 시점 판별

파티션된 환경에서 로드 조작에 실패한 경우 처음 수행해야 하는 작업은 실패한 파티션 및 각 파티션에서 실패한 단계를 판별하는 것입니다. 파티션 요약을 보면 알 수 있습니다. CLP에서 load 명령을 실행한 경우 로드 끝에 파티션 요약이 표시됩니다(아래 예 참조). db2Load API에서 load 명령을 실행한 경우 db2PartLoadOut 구조의 poAgentInfoList 필드에 파티션 요약이 포함됩니다.

지정된 파티션에서 "에이전트 유형"이 "LOAD"인 항목이 있으면 해당 파티션은 로드 단계에 도달한 것입니다. 그렇지 않으면 설정 단계에서 실패합니다. 음수 SQL 코드는 실패했음을 의미합니다. 다음 예에서 로드는 로드 단계 중 파티션 1에서 실패했습니다.

```
에이전트 유형  노드     SQL 코드     결과
_____________________________________________________
LOAD          000      +00000000    성공
_____________________________________________________
LOAD          001      -00000289    오류. 재시작 필요
_____________________________________________________
LOAD          002      +00000000    성공
_____________________________________________________
LOAD          003      +00000000    성공
.
.
.
```

실패한 로드 다시 시작, 재시작 또는 종료

SETUP_ERRS_ONLY 또는 SETUP_AND_LOAD_ERRS를 지정하여 ISOLATE_PART_ERRS 옵션으로 수행하는 로드만 설정 단계 중에 실패합니다. 이 단계 중에 하나 이상의 출력 데이터베이스 파티션에서 로드에 실패하면 LOAD REPLACE 또는 LOAD INSERT 명령을 실행할 수 있습니다. OUTPUT_DBPARTNUMS 옵션을 사용하여 실패한 해당 데이터베이스 파티션만 지정합니다.

로드 단계 중 하나 이상의 출력 데이터베이스 파티션에서 로드에 실패하면 로드 RESTART 또는 로드 TERMINATE 명령을 실행합니다.

설정 단계 중 하나 이상의 출력 데이터베이스 파티션에서, 그리고 로드 단계 중 하나 이상의 출력 데이터베이스 파티션에서 로드에 실패하면 이전에 설명한 대로 로드 단계

중 실패한 로드 하나 및 설정 단계 중 실패한 로드 하나와 같은 두 개의 로드 조작을 수행하여 실패한 로드를 다시 시작해야 합니다. 이러한 유형의 실패한 로드 조작을 효과적으로 실행 취소하려면 로드 TERMINATE 명령을 실행합니다. 그러나 명령을 실행한 후에는 모든 파티션을 고려해야 합니다. 설정 단계 중 실패한 파티션에서 테이블은 변경되지 않고 로드 단계 중 실패한 파티션에서 모든 변경 사항은 실행 취소되기 때문입니다.

예를 들어 TABLE1은 데이터베이스 WSDB의 데이터베이스 파티션 0 - 3에 정의되어 있습니다. 다음 명령이 실행됩니다.

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

설정 단계 중 출력 데이터베이스 파티션 1에서 실패했습니다. 설정 단계 오류는 분리되므로 로드 조작은 계속되지만 로드 단계 중 파티션 3에서 실패했습니다. 로드 조작을 다시 시작하려면 다음 명령을 실행합니다.

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

주: 로드 재시작 조작의 경우 LOAD RESTART 명령에 지정된 옵션은 처리되므로 원래 LOAD 명령에 지정된 옵션과 동일해야 합니다.

이주 및 버전 호환성

DB2_PARTITIONEDLOAD_DEFAULT 레지스트리 변수는 다중 파티션 데이터베이스에서 이전 DB2 Universal Database 버전 8 로드 동작으로 되돌리는 데 사용될 수 있습니다.

주: DB2_PARTITIONEDLOAD_DEFAULT 레지스트리 변수는 사용되지 않으며 추후 릴리스에서 제거됩니다.

다중 파티션 데이터베이스에서 LOAD 명령의 이전 DB2 UDB 버전 8 동작으로 되돌리면 추가 파티션된 데이터베이스 구성 옵션을 지정하지 않고도 단일 데이터베이스 파티션에 올바른 분산 헤더를 포함하는 파일을 로드할 수 있습니다. **DB2_PARTITIONEDLOAD_DEFAULT** 값을 NO로 설정하면 됩니다. 단일 데이터베이스 파티션에서 LOAD 명령을 실행하는 기존 스크립트를 수정하지 않으려면 이 옵션을 사용하도록 선택할 수 있습니다. 예를 들어 4개의 데이터베이스 파티션 그룹으로 구성된 데이터베이스 파티션 그룹에 있는 테이블의 데이터베이스 파티션 3으로 분산 파일을 로드하려면 다음 명령을 실행합니다.

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

그 다음 DB2 명령행 처리기에서 다음 명령을 실행하십시오.

```
CONNECT RESET

SET CLIENT CONNECT_NODE 3

CONNECT TO DB MYDB

LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

다중 파티션 데이터베이스의 경우 다중 파티션 데이터베이스 로드 구성 옵션이 지정되지 않으면 테이블이 정의된 모든 데이터베이스 파티션에서 로드 조작을 수행합니다. 입력 파일에 분산 헤더가 필요하지 않으며 MODE 옵션은 디폴트로 PARTITION_AND_LOAD로 지정됩니다. 단일 데이터베이스 파티션을 로드하려면 OUTPUT_DBPARTNUMS 옵션을 지정해야 합니다.

참조 - 파티션된 환경에서 로드

파티션된 데이터베이스 환경의 로드 세션 - CLP 예

다음 예에서는 다중 파티션 데이터베이스로 데이터를 로드하는 방법에 대해 설명합니다.

데이터베이스에는 0부터 3까지 번호가 지정된 4개의 데이터베이스 파티션이 있습니다. 데이터베이스 WSDB는 모든 데이터베이스 파티션에 정의되어 있으며 테이블 TABLE1은 마찬가지로 모든 데이터베이스 파티션에 정의된 디폴트 데이터베이스 파티션 그룹에 있습니다.

예 1

데이터베이스 파티션 0에 있는 사용자 데이터 파일 load.del에서 TABLE1로 데이터를 로드하려면 데이터베이스 파티션 0에 연결하고 다음 명령을 실행하십시오.

```
load from load.del of del replace into table1
```

로드 조작에 성공하면 출력은 다음과 같습니다.

```
에이전트 유형  노드     SQL 코드     결과
_____________________________________________
LOAD           000      +00000000    성공
_____________________________________________
LOAD           001      +00000000    성공.
_____________________________________________
LOAD           002      +00000000    성공
_____________________________________________
LOAD           003      +00000000    성공
_____________________________________________
PARTITION      001      +00000000    성공.
_____________________________________________
PRE_PARTITION  000      +00000000    성공.
_____________________________________________
결과:          4 / 4 LOAD가 완료되었습니다.
_____________________________________________
```

```
파티션 에이전트 요약:
읽은 행 수                  = 100000
거부된 행 수                = 0
파티션된 행 수              = 100000

LOAD 에이전트 요약:
읽은 행 수                  = 100000
건너뛴 행 수                = 0
로드된 행 수                = 100000
거부된 행 수                = 0
삭제된 행 수                = 0
커미트된 행 수              = 100000
```

출력에서는 각 데이터베이스 파티션에 하나의 로드 에이전트가 있고 각각 성공적으로 실행되었음을 표시합니다. 또한 데이터베이스 파티션 1에서 실행하는 하나의 파티셔닝 에이전트와 코디네이터 파티션에서 실행하는 하나의 사전 파티셔닝 에이전트가 있음을 표시합니다. 이 프로세스는 일반 SQL 리턴 코드 0을 표시하며 성공적으로 완료됩니다. 통계 요약에서는 사전 파티셔닝 에이전트가 100,000개 행을 읽고 파티셔닝 에이전트가 100,000개 행을 분산시켰고 로드 에이전트에서 로드한 모든 행의 합계가 100,000개임을 표시합니다.

예 2

다음 예에서 PARTITION_ONLY 모드로 데이터가 TABLE1에 로드됩니다. 분산된 출력 파일은 /db/data 디렉토리의 각 출력 데이터베이스 파티션에 저장됩니다.

```
load from load.del of del replace into table1 partitioned db config mode
   partition_only part_file_location /db/data
```

load 명령의 출력은 다음과 같습니다.

```
에이전트 유형  노드     SQL 코드     결과
_____________________________________________________
LOAD_TO_FILE   000      +00000000    성공.
_____________________________________________________
LOAD_TO_FILE   001      +00000000    성공.
_____________________________________________________
LOAD_TO_FILE   002      +00000000    성공.
_____________________________________________________
LOAD_TO_FILE   003      +00000000    성공.
_____________________________________________________
PARTITION      001      +00000000    성공.
_____________________________________________________
PRE_PARTITION  000      +00000000    성공.
_____________________________________________________

파티션 에이전트 요약:
읽은 행 수                  = 100000
거부된 행 수                = 0
파티션된 행 수              = 100000
```

출력에서는 각 출력 데이터베이스 파티션에서 실행하는 파일로 로드하는 에이전트가 있으며 이 에이전트가 성공적으로 실행되었음을 표시합니다. 데이터베이스 파티션 1에서

실행하는 파티셔닝 에이전트와 코디네이터 파티션에서 실행하는 사전 파티셔닝 에이전트가 있습니다. 통계 요약에서는 사전 파티셔닝 에이전트에서 100,000개 행을 성공적으로 읽고 파티셔닝 에이전트에서 100,000개 행이 성공적으로 분산되었음을 나타냅니다. 테이블로 로드된 행이 없으므로 로드된 행 수에 대한 요약은 나타나지 않습니다.

예 3

위의 PARTITION_ONLY 로드 조작 중 생성된 파일을 로드하려면 다음 명령을 실행하십시오.

```
load from load.del of del replace into table1 partitioned db config mode
       load_only part_file_location /db/data
```

load 명령의 출력은 다음과 같습니다.

```
에이전트 유형  노드      SQL 코드      결과
------------------------------------------------------------------
LOAD           000       +00000000     성공
------------------------------------------------------------------
LOAD           001       +00000000     성공.
------------------------------------------------------------------
LOAD           002       +00000000     성공
------------------------------------------------------------------
LOAD           003       +00000000     성공
------------------------------------------------------------------
결과:          4 / 4 LOAD가 완료되었습니다.
------------------------------------------------------------------

LOAD 에이전트 요약:
읽은 행 수                  = 100000
건너뛴 행 수                = 0
로드된 행 수                = 100000
거부된 행 수                = 0
삭제된 행 수                = 0
커미트된 행 수              = 100000
```

출력에서는 각 출력 데이터베이스 파티션의 로드 에이전트가 성공적으로 실행되었으며 모든 로드 에이전트에서 로드된 행 수가 100,000개임을 표시합니다. 분산이 수행되지 않았으므로 분산된 행 수는 표시되지 않습니다.

예 4 - 실패한 로드 조작

다음 LOAD 명령을 발행한 경우

```
load from load.del of del replace into table1
```

로드 조작 중 로드하는 데이터베이스 파티션에서 테이블 스페이스의 공간이 부족한 경우 다음 출력이 리턴됩니다.

```
SQL0289N  테이블 스페이스 "DMS4KT"에 새 페이지를 할당할 수 없습니다.
SQLSTATE=57011

에이전트 유형  노드      SQL 코드      결과
------------------------------------------------------------------
```

```
LOAD           000      +00000000    성공
_____________________________________________________________
LOAD           001      -00000289    오류. 재시작 필요
_____________________________________________________________
LOAD           002      +00000000    성공
_____________________________________________________________
LOAD           003      +00000000    성공
_____________________________________________________________
PARTITION      001      +00000000    성공.
_____________________________________________________________
PRE_PARTITION  000      +00000000    성공.
_____________________________________________________________
결과:          3 / 4 LOAD가 완료되었습니다.
_____________________________________________________________

파티션 에이전트 요약:
읽은 행 수                = 0
거부된 행 수              = 0
파티션된 행 수            = 0

LOAD 에이전트 요약:
읽은 행 수                = 0
건너뛴 행 수              = 0
로드된 행 수              = 0
거부된 행 수              = 0
삭제된 행 수              = 0
커미트된 행 수            = 0
```

출력에서는 로드 조작에서 오류 SQL0289를 리턴함을 표시합니다. 데이터베이스 파티션 요약에서는 데이터베이스 파티션 1에서 스페이스가 부족함을 표시합니다. 데이터베이스 파티션 1에서 테이블 스페이스의 컨테이너에 추가 스페이스를 추가하면 다음과 같이 로드 조작을 재시작할 수 있습니다.

```
load from load.del of del restart into table1
```

파티션된 데이터베이스 환경에 대한 로드 구성

MODE X

다중 파티션 데이터베이스 로드 중 로드 조작에서 사용하는 모드를 지정합니다. PARTITION_AND_LOAD가 디폴트입니다. 가능한 값은 다음과 같습니다.

- PARTITION_AND_LOAD. 데이터가 해당 데이터베이스 파티션에서 동시에 분산(병렬 방식일 수 있음) 및 로드됩니다.
- PARTITION_ONLY. 데이터가 분산되고(병렬 방식일 수 있음) 로드하는 각 데이터베이스 파티션에서 지정된 위치의 파일에 작성됩니다. CURSOR 이외의 파일 유형에서 각 데이터베이스 파티션의 출력 파일 이름 형식은 filename.xxx입니다. 여기서 filename은 LOAD 명령에 지정된 입력 파일 이름이고 xxx는 3자리 데이터베이스 파티션 번호입니다. CURSOR 파일 유형의 경우 각 데이터베이스 파티션에서 출력 파일 이름은

PART_FILE_LOCATION 옵션으로 판별됩니다. 각 데이터베이스 파티션에서 분산 위치를 지정하는 방법에 대한 세부사항은 PART_FILE_LOCATION 옵션을 참조하십시오.

주:

1. 이 모드는 CLI 로드 조작에서 사용할 수 없습니다.
2. 분산에 필요한 ID 컬럼이 테이블에 포함된 경우 identityoverride 파일 유형 수정자를 지정하지 않는 한 이 모드는 지원되지 않습니다.
3. CURSOR 파일 유형에 대해 생성된 분산 파일은 DB2 릴리스 사이에서 호환 가능하지 않습니다. 즉, 이전 릴리스에서 생성된 파일 유형 CURSOR의 분산 파일은 LOAD_ONLY 모드를 사용하여 로드할 수 없습니다. 마찬가지로 현재 릴리스에서 생성된 파일 유형 CURSOR의 분산 파일은 LOAD_ONLY 모드를 사용하여 추후 릴리스에서 로드할 수 없습니다.

- LOAD_ONLY. 데이터는 이미 분산되었다고 가정합니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. CURSOR 이외의 파일 유형에서 각 데이터베이스 파티션의 입력 파일 이름 형식은 filename.xxx입니다. 여기서 filename은 LOAD 명령에 지정된 파일 이름이고 xxx는 3자리 데이터베이스 파티션 번호입니다. CURSOR 파일 유형의 경우 각 데이터베이스 파티션에서 입력 파일 이름은 PART_FILE_LOCATION 옵션으로 판별됩니다. 각 데이터베이스 파티션에서 분산 위치를 지정하는 방법에 대한 세부사항은 PART_FILE_LOCATION 옵션을 참조하십시오.

  주:

  1. 이 모드는 CLI 로드 조작 또는 LOAD 명령의 CLIENT 옵션이 지정된 경우에 사용할 수 없습니다.
  2. 분산에 필요한 ID 컬럼이 테이블에 포함된 경우 identityoverride 파일 유형 수정자를 지정하지 않는 한 이 모드는 지원되지 않습니다.
- LOAD_ONLY_VERIFY_PART. 데이터는 이미 분산되었다고 가정합니다. 그러나 데이터 파일은 파티션 헤더를 포함하지 않습니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. 로드 조작 중에 각 행은 올바른 데이터베이스 파티션에 있는지 확인하기 위해 점검됩니다. 데이터베이스 파티션 위반을 포함하는 행은 dumpfile 파일 유형 수정자가 지정된 경우 덤프 파일에 배치됩니다. 그렇지 않으면 행을 버립니다. 데이터베이스 파티션 위반이 로드하는 특정 데이터베이스 파티션에 있으면 해당 데이터베이스 파티션에 대한 단일 경고가 로드 메시지 파일에 작성됩니다. 각 데이터베이스 파티션의 입력 파일 이름 형식은 filename.xxx입니다. 여기서 filename은 LOAD 명령에 지정된 파일 이름이고 xxx는 3자리 데

이터베이스 파티션 번호입니다. 각 데이터베이스 파티션에서 분산 위치를 지정하는 방법에 대한 세부사항은 PART_FILE_LOCATION 옵션을 참조하십시오.

주:

1. 이 모드는 CLI 로드 조작 또는 LOAD 명령의 CLIENT 옵션이 지정된 경우에 사용할 수 없습니다.
2. 분산에 필요한 ID 컬럼이 테이블에 포함된 경우 `identityoverride` 파일 유형 수정자를 지정하지 않는 한 이 모드는 지원되지 않습니다.

- ANALYZE. 모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다.

PART_FILE_LOCATION X

PARTITION_ONLY, LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART 모드에서 이 매개변수를 사용하여 분산된 파일의 위치를 지정할 수 있습니다. 이 위치는 OUTPUT_DBPARTNUMS 옵션에서 지정한 각 데이터베이스 파티션에 있어야 합니다. 지정한 위치가 상대 경로 이름인 경우 경로는 현재 디렉토리에 추가되어 분산된 파일의 위치를 작성합니다.

CURSOR 파일 유형의 경우 이 옵션을 지정하고 위치는 완전한 파일 이름을 참조해야 합니다. 이 이름은 PARTITION_ONLY 모드에서 각 출력 데이터베이스 파티션에 작성된 분산 파일의 완전한 기본 파일 이름 또는 LOAD_ONLY 모드에서 각 데이터베이스 파티션에서 읽을 파일 위치입니다.
PARTITION_ONLY 모드를 사용하면 목표 테이블에 LOB 컬럼이 있는 경우 지정된 기본 이름으로 다중 파일을 작성할 수 있습니다.

CURSOR 이외의 파일 유형인 경우 이 옵션을 지정하지 않으면 분산 파일에 대해 현재 디렉토리가 사용됩니다.

OUTPUT_DBPARTNUMS X

X는 데이터베이스 파티션 번호 목록입니다. 데이터베이스 파티션 번호는 로드 조작을 수행할 데이터베이스 파티션을 나타냅니다. 데이터베이스 파티션 번호는 테이블이 정의된 데이터베이스 파티션의 서브세트여야 합니다. 모든 데이터베이스 파티션은 디폴트로 선택됩니다. 목록은 괄호로 묶어야 하며 목록의 항목은 쉼표로 구분되어야 합니다. 범위는 허용됩니다(예: 0, 2에서 10, 15).

PARTITIONING_DBPARTNUMS X

X는 분산 프로세스에서 사용된 데이터베이스 파티션 번호 목록입니다. 목록은 괄호로 묶어야 하며 목록의 항목은 쉼표로 구분되어야 합니다. 범위는 허용됩니다(예: 0, 2에서 10, 15). 분산 프로세스에 지정된 데이터베이스 파티션은 로드할 데이터베이스 파티션과 다를 수 있습니다.

PARTITIONING_DBPARTNUMS가 지정되지 않은 경우 로드 유틸리티는 최적의 성능을 얻기 위해 사용할 데이터베이스 파티션 및 필요한 데이터베이스 파티션 수를 판별합니다.

anyorder 파일 유형 수정자가 LOAD 명령에 지정되지 않은 경우 로드 세션에서는 하나의 파티셔닝 에이전트만 사용됩니다. 더 나아가 OUTPUT_DBPARTNUMS 옵션에 하나의 데이터베이스 파티션만 지정된 경우 또는 로드 조작의 코디네이터 파티션이 OUTPUT_DBPARTNUMS의 요소가 아닌 경우 분산 프로세스에서 로드 조작의 코디네이터 파티션이 사용됩니다. 그렇지 않으면 OUTPUT_DBPARTNUMS의 첫 번째 데이터베이스 파티션(코디네이터 파티션이 아님)이 분산 프로세스에 사용됩니다.

anyorder 파일 유형 수정자가 지정되면 분산 프로세스에 사용되는 데이터베이스 파티션 수는 다음과 같이 판별됩니다. (OUTPUT_DBPARTNUMS의 파티션 수/4 + 1).

MAX_NUM_PART_AGENTS X

로드 세션에서 사용할 최대 파티셔닝 에이전트 수를 지정합니다. 디폴트값은 25입니다.

ISOLATE_PART_ERRS X

개별 데이터베이스 파티션에서 발생하는 오류에 로드 조작이 대응하는 방식을 표시합니다. LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션이 지정되지 않는 한(이 경우 디폴트는 NO_ISOLATION임) 디폴트는 LOAD_ERRS_ONLY입니다. 가능한 값은 다음과 같습니다.

- SETUP_ERRS_ONLY. 설정 중 데이터베이스 파티션에서 발생하는 오류(예: 데이터베이스 파티션에 액세스하는 중 발생하는 문제점 또는 데이터베이스 파티션의 테이블 또는 테이블 스페이스에 액세스하는 중 발생하는 문제점)로 실패한 데이터베이스 파티션에서 로드 조작이 중지되지만 나머지 데이터베이스 파티션에서는 계속 진행됩니다. 데이터를 로드하는 중 데이터베이스 파티션에서 발생하는 오류로 전체 조작에 실패합니다.
- LOAD_ERRS_ONLY. 설정 중 데이터베이스 파티션에서 발생한 오류로 전체 로드 조작에 실패할 수 있습니다. 데이터를 로드하는 중 오류가 발생하면 오류가 발생한 데이터베이스 파티션에서 로드 조작이 중지됩니다. 실패가 발생하거나 모든 데이터를 로드할 때까지 나머지 데이터베이스 파티션에서 로드 조작이 계속됩니다. 새로 로드된 데이터는 로드 재시작 조작을 수행하여 성공적으로 완료할 때까지 보이지 않습니다.

  주: 이 모드는 LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션이 지정된 경우 사용할 수 없습니다.
- SETUP_AND_LOAD_ERRS. 이 모드에서는 설정이나 데이터 로드 중에 발생한 데이터베이스 파티션 오류로 인해 영향 받는 데이터베이스 파티션에서

만 처리가 중지됩니다. LOAD_ERRS_ONLY 모드와 마찬가지로 데이터를 로드하는 중 파티션 오류가 발생하면 새로 로드된 데이터는 로드 재시작 조작을 수행하여 성공적으로 완료할 때까지 보이지 않습니다.

주: 이 모드는 LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션이 지정된 경우 사용할 수 없습니다.

- NO_ISOLATION. 로드 조작 중 오류가 발생하면 로드 조작에 실패합니다.

STATUS_INTERVAL X

X는 읽은 데이터의 볼륨을 알리는 간격을 표시합니다. 측정 단위는 메가바이트(MB)입니다. 디폴트값은 100MB입니다. 올바른 값은 1에서 4000 사이의 정수입니다.

PORT_RANGE X

X는 내부 통신에서 소켓을 작성하는 데 사용하는 TCP 포트 범위를 표시합니다. 디폴트 범위는 6000에서 6063 사이입니다. 호출 시 정의되는 경우 **DB2ATLD_PORTS** 레지스트리 변수의 값이 PORT_RANGE 로드 구성 옵션의 값을 교체합니다. **DB2ATLD_PORTS** 레지스트리 변수의 경우 범위는 다음 형식으로 제공해야 합니다.

```
<lower-port-number:higher-port-number>
```

CLP에서 형식은 다음과 같습니다.

```
( lower-port-number, higher-port-number )
```

CHECK_TRUNCATION

입출력 시 프로그램에서 데이터 레코드가 절단되었는지 확인하도록 지정합니다. 디폴트 동작은 입출력 시 데이터가 절단되었는지 확인하지 않는 것입니다.

MAP_FILE_INPUT X

X는 분산 맵의 입력 파일 이름을 지정합니다. 이 매개변수는 분산 맵이 사용자 정의된 경우 사용자 정의된 분산 맵을 포함하는 파일을 가리킬 때 지정해야 합니다. 사용자 정의된 분산 맵은 db2gpmap 프로그램을 사용하여 데이터베이스 시스템 카탈로그 테이블에서 맵을 추출하거나 LOAD 명령의 ANALYZE 모드를 통해 최적의 맵을 생성하여 작성할 수 있습니다. ANALYZE 모드를 사용하여 생성된 맵은 로드 조작을 계속 진행하려면 데이터베이스의 각 데이터베이스 파티션으로 이동되어야 합니다.

MAP_FILE_OUTPUT X

X는 분산 맵의 출력 파일 이름을 나타냅니다. 출력 파일은 파티셔닝을 수행할 때 데이터베이스 파티션이 데이터베이스 파티션 그룹에 참여한다고 가정하고 LOAD 명령을 실행하여 데이터베이스 파티션에서 작성됩니다. PARTITIONING_DBPARTNUMS에서 정의한 대로 참여하지 않는 데이터베이스 파티션에서 LOAD 명령을 호출하면 출력 파일은

PARTITIONING_DBPARTNUMS 매개변수로 정의된 첫 번째 데이터베이스 파티션에서 작성됩니다. 다음의 파티션된 데이터베이스 환경 설정을 고려합니다.

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

serv3에서 다음 LOAD 명령을 실행하면 serv1에 분산 맵이 작성됩니다.

```
LOAD FROM file OF ASC METHOD L ( ...) INSERT INTO table CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

이 매개변수는 ANALYZE 모드가 지정된 경우 사용해야 합니다. 모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다. 이 매개변수를 지정하지 않고 ANALYZE 모드가 지정되면 프로그램은 오류로 종료됩니다.

TRACE X

해시 값의 출력 및 데이터 변환 프로세스의 덤프를 검토해야 하는 경우 추적할 레코드 수를 지정합니다. 디폴트값은 0입니다.

NEWLINE

입력 데이터 파일이 각 레코드가 개행 문자로 구분된 ASC 파일이고 reclen 파일 유형 수정자가 LOAD 명령에 지정된 경우 사용합니다. 이 옵션을 지정하면 각 레코드에 개행 문자가 있는지 확인합니다. reclen 파일 유형 수정자에 지정된 대로 레코드 길이도 확인합니다.

DISTFILE X

이 옵션을 지정하면 로드 유틸리티가 지정된 이름으로 데이터베이스 파티션 분산 파일을 생성합니다. 데이터베이스 파티션 분산 파일은 32 768개 정수를 포함하며 목표 테이블의 분산 맵에서 각 항목에 하나씩 존재합니다. 파일의 각 정수는 대응하는 분산 맵 항목으로 해시되는 로드할 입력 파일의 행 수를 나타냅니다. 이 정보는 데이터에서 비대칭 항목을 식별하고 유틸리티의 ANALYZE 모드를 사용하여 테이블에서 새 분산 맵을 생성해야 하는지 여부를 결정하는 데에도 도움이 됩니다. 이 옵션을 지정하지 않으면 로드 유틸리티의 디폴트 동작은 분산 파일을 생성하지 않는 것입니다.

주: 이 옵션을 지정하면 로드 조작에 대해 최대 하나의 파티셔닝 에이전트를 사용합니다. 명시적으로 다중 파티셔닝 에이전트를 요청해도 하나만 사용됩니다.

OMIT_HEADER

분산 맵 헤더를 분산 파일에 포함하지 않도록 지정합니다. 지정하지 않으면 헤더가 생성됩니다.

RUN_STAT_DBPARTNUM X

LOAD 명령에 STATISTICS YES 매개변수가 지정되면 하나의 데이터베이스 파티션에서만 통계가 수집됩니다. 이 매개변수에서는 통계를 수집할 데이터베이스 파티션을 지정합니다. 값이 -1이거나 전혀 지정되지 않으면 출력 데이터베이스 파티션 목록의 첫 번째 데이터베이스 파티션에서 통계가 수집됩니다.

참조 - 로드

LOAD

DB2 테이블에 데이터를 로드합니다. 서버에 있는 데이터는 파일, 테이프 또는 Named Pipe 양식이 될 수 있습니다. 테이블의 COMPRESS 속성이 YES로 설정된 경우, 테이블의 XML 스토리지 오브젝트의 데이터를 포함하여 이미 사전이 테이블에 존재하는 모든 데이터 및 데이터베이스 파티션에서 로드된 데이터가 압축됩니다.

267 페이지의 『로드 유틸리티의 파일 유형 수정자』로의 빠른 링크를 참조하십시오.

제한사항

로드 유틸리티는 계층 구조 레벨에서 데이터 로드를 지원하지 않습니다. 로드 유틸리티는 범위로 클러스터된 테이블과 호환 가능하지 않습니다.

범위

이 명령은 단일 요청으로 다중 데이터베이스 파티션에 대해 실행할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- *dataaccess*
- 데이터베이스의 LOAD 권한 및
  - 로드 유틸리티가 INSERT 모드, TERMINATE 모드 (이전 로드 삽입 조작을 종료하기 위해) 또는 RESTART 모드(이전 로드 삽입 조작을 재시작하기 위해)에서 호출될 때 테이블에 대한 INSERT 특권
  - 로드 유틸리티가 REPLACE 모드, TERMINATE 모드(이전 로드 삽입 조작을 종료하기 위해) 또는 RESTART 모드(이전 로드 삽입 조작을 재시작하기 위해)에서 호출될 때 테이블에 대한 INSERT 및 DELETE 특권
  - 예외 테이블에 대한 INSERT 특권(이러한 테이블이 로드 조작 중에 사용되는 경우).

- 보호 컬럼이 있는 테이블로 데이터를 로드하려면 테이블의 모든 보호 컬럼에 대한 쓰기 액세스를 허용하는 LBAC 증명서가 세션 권한 부여 ID에 있어야 합니다. 그렇지 않으면 로드에 실패하고 오류(SQLSTATE 5U014)가 리턴됩니다.
- 보호 설정된 행이 있는 테이블에 데이터를 로드하려면 세션 권한 부여 ID는 다음 기준에 부합되는 보안 레이블을 보유해야 합니다.
  - 테이블을 보호하는 보안 규정에 포함됨
  - 세션 권한 부여 ID에 쓰기 액세스 또는 모든 액세스 권한이 부여되었습니다.

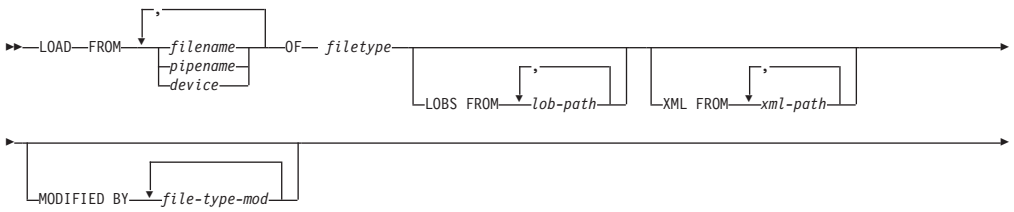
  세션 권한 부여 ID에 이러한 보안 레이블이 없으면 로드에 실패하고 오류(SQLSTATE 5U014)가 리턴됩니다. 이 보안 레벨은 세션 권한 부여 ID'의 LBAC 증명서가 데이터의 해당 행을 보호하는 보안 레벨에 쓰기를 허용하지 않는 경우 로드된 행을 보호하는 데 사용됩니다. 그러나 테이블을 보호하는 보안 규정이 CREATE SECURITY POLICY 명령문의 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 사용하여 작성되는 경우에는 발생하지 않습니다. 이러한 경우, 로드에 실패하고 오류(SQLSTATE 42519)가 리턴됩니다.
- REPLACE 옵션이 지정되어 있으면 세션 권한 부여 ID에 테이블을 삭제할 수 있는 권한이 있어야 합니다.
- LOCK WITH FORCE 옵션이 지정된 경우, SYSADM 권한이 필요합니다.

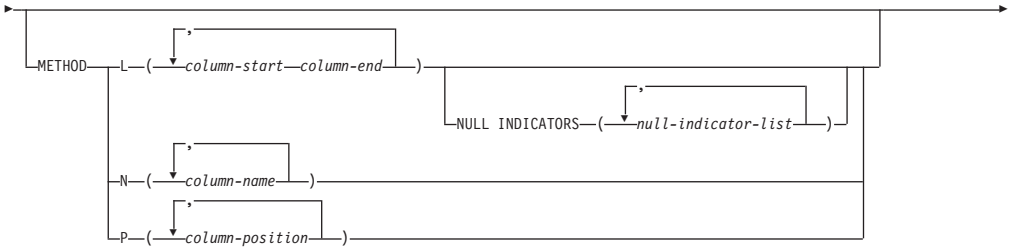
모든 로드 프로세스(및 일반적으로 모든 DB2 서버 프로세스)는 인스턴스 소유자가 소유하고 이들 모든 프로세스는 필요한 파일에 액세스하기 위해 인스턴스 소유자의 식별을 사용하므로 인스턴스 소유자는 입력 데이터 파일에 대한 읽기 액세스가 있어야 합니다. 이러한 입력 데이터 파일은 명령 호출자에 상관없이 인스턴스 소유자가 읽을 수 있어야 합니다.

필수 연결

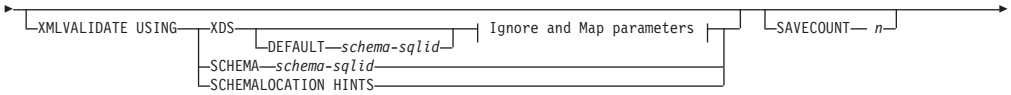
인스턴스. 명시적 접속은 필요하지 않습니다. 데이터베이스에 대한 연결이 설정되면 로컬 인스턴스에 대한 내재적 접속이 시도됩니다.

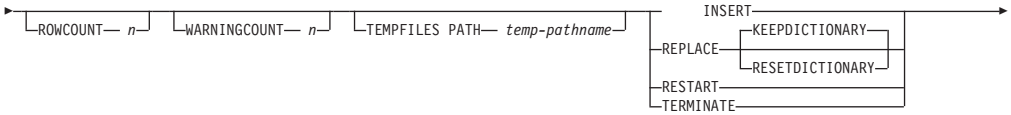
명령 구문

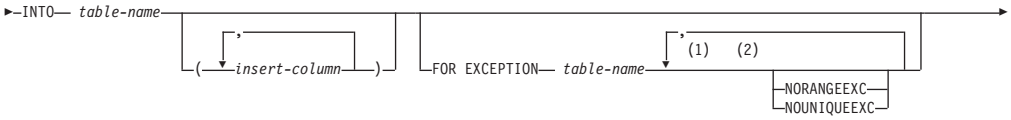


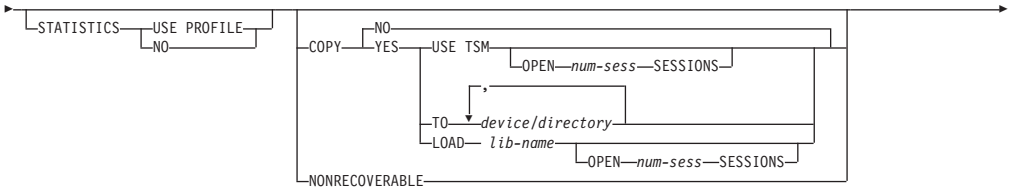

METHOD
L
(
,
column-start
column-end
)
NULL INDICATORS
(
,
null-indicator-list
)
N
(
,
column-name
)
P
(
,
column-position
)

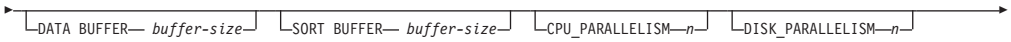

XMLPARSE
STRIP
PRESERVE
WHITESPACE

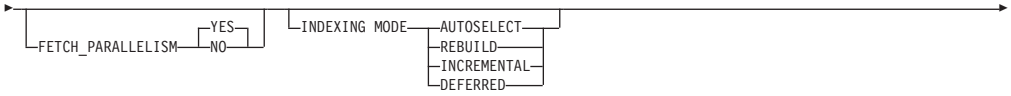

XMLVALIDATE USING
XDS
DEFAULT
schema-sqlid
Ignore and Map parameters
SCHEMA
schema-sqlid
SCHEMALOCATION HINTS
SAVECOUNT
n


ROWCOUNT
n
WARNINGCOUNT
n
TEMPFILES PATH
temp-pathname
INSERT
REPLACE
KEEPDICTIONARY
RESETDICTIONARY
RESTART
TERMINATE

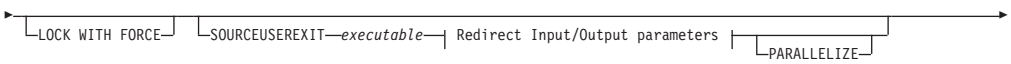

INTO
table-name
(
,
insert-column
)
FOR EXCEPTION
table-name
,
(1) (2)
NORANGEEXC
NOUNIQUEEXC

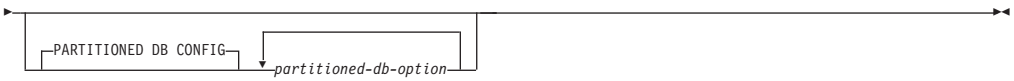

STATISTICS
USE PROFILE
NO
COPY
NO
YES
USE TSM
OPEN
num-sess
SESSIONS
TO
,
device/directory
LOAD
lib-name
OPEN
num-sess
SESSIONS
NONRECOVERABLE


DATA BUFFER
buffer-size
SORT BUFFER
buffer-size
CPU_PARALLELISM
n
DISK_PARALLELISM
n

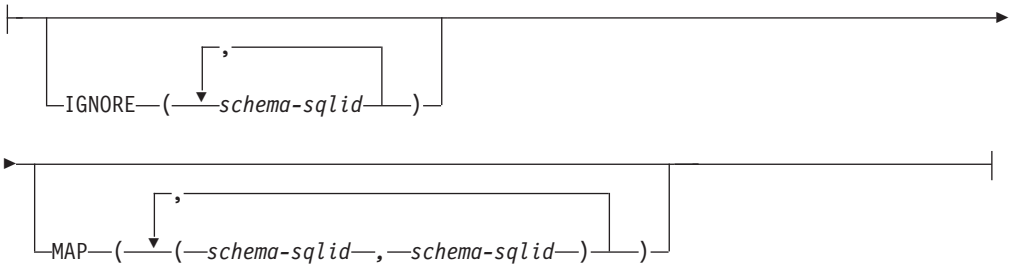

FETCH_PARALLELISM
YES
NO
INDEXING MODE
AUTOSELECT
REBUILD
INCREMENTAL
DEFERRED


ALLOW NO ACCESS
ALLOW READ ACCESS
USE
tablespace-name
SET INTEGRITY PENDING CASCADE
IMMEDIATE
DEFERRED

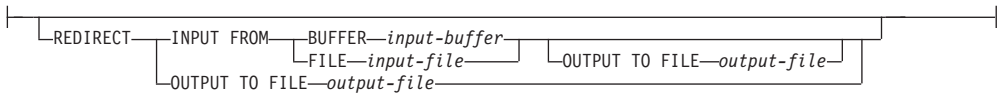

LOCK WITH FORCE
SOURCEUSEREXIT
executable
Redirect Input/Output parameters
PARALLELIZE



Ignore and Map parameters:



Redirect Input/Output parameters:



주:

1 이 키워드는 임의의 순서로 표시될 수 있습니다.

2 이들 각 키워드는 한 번만 표시될 수 있습니다.

명령 매개변수

FROM *filename* | *pipename* | *device*

주:

1. *ADMIN_CMD 프로시저를 사용하는 EXPORT 명령을* 사용하여 파일로 데이터를 익스포트하는 경우, 데이터 파일은 분리 사용자 ID 소유가 됩니다. 일반적으로 이 파일은 인스턴스 소유자가 액세스할 수 없습니다. CLP 또는 ADMIN_CMD 프로시저에서 LOAD를 실행시키려면 데이터 파일이 인스턴스 소유자 ID에 의해 액세스 가능해야 하며 따라서 데이터 파일에 대한 읽기 액세스 권한이 인스턴스 소유자에게 부여되어야 합니다.
2. 파일이 실제로는 분리되어 있지만 논리적으로는 한 개인 경우, 다중 IXF 파일로부터의 데이터 로드가 지원됩니다. 파일이 논리적 및 물리적으로 분리된 경우에는 지원되지 *않습니다*. (다중의 실제 파일도 EXPORT 명령을 한 번 호출하여 모두 작성된 경우 논리적으로는 한 개로 간주됩니다.)
3. 파티션된 데이터베이스 환경에서, XML 데이터를 파일에서 테이블로 로딩하는 경우에는 로드를 실행 중인 모든 데이터베이스 파티션에서 XML 데이터 파일에 읽기 액세스가 가능해야 합니다.

OF *filetype*

다음과 같은 데이터 형식을 지정합니다.

- ASC(컬럼 식별자가 없는 ASCII 형식)
- DEL(컬럼 식별자가 있는 ASCII 형식)
- IXF(Integration Exchange Format, PC 버전)는 DB2에 의해 독점 사용되는 2진 형식입니다.
- CURSOR(SELECT 또는 VALUES문에 대해 선언된 커서).

주: 분산 데이터베이스 환경에서, CURSOR 파일 유형을 사용하여 XML 데이터를 테이블로 로드하는 경우에는 PARTITION_ONLY 및 LOAD_ONLY 모드가 지원되지 않습니다.

LOBS FROM *lob-path*

로드할 LOB 값이 포함된 데이터 파일의 경로. 경로는 슬래시(/)로 끝나야 합니다. LOB 데이터 파일의 이름은 기본 데이터 파일(ASC, DEL 또는 IXF)에 저장되는데 LOB 컬럼으로 로드될 컬럼에 저장됩니다. 지정할 수 있는 최대 경로 수는 999입니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

이 옵션은 CURSOR 파일 유형과 함께 지정되면 무시됩니다.

MODIFIED BY *file-type-mod*

파일 유형 수정자 옵션을 지정합니다. 267 페이지의 『로드 유틸리티의 파일 유형 수정자』를 참조하십시오.

METHOD

L 데이터를 로드할 시작 및 끝 컬럼을 지정합니다. 컬럼 번호는 데이터 행이 시작되는 바이트 오프셋입니다. 컬럼 번호는 1부터 시작됩니다. 이 메소드는 ASC 파일에서만 사용될 수 있으며 해당 파일 유형에 유일한 유효 메소드입니다.

NULL INDICATORS *null-indicator-list*

이 옵션은 METHOD L 매개변수가 지정된 경우(입력 파일이 ASC 파일)에만 사용할 수 있습니다. 널(NULL) 표시기 목록은 각 널(NULL) 표시기 필드의 컬럼 번호를 지정하는 쉼표로 구분된 양의 정수 목록입니다. 컬럼 번호는 데이터 행이 시작되는 널(NULL) 표시기 필드의 바이트 오프셋입니다. METHOD L 매개변수에 정의된 각 데이터 필드에 대해 한 개의 항목이 널(NULL) 표시기 목록에 있어야 합니다. 컬럼 번호 0은 해당 데이터 필드에 데이터가 항상 들어 있음을 나타냅니다.

널(NULL) 표시기의 Y 값은 컬럼 데이터를 널(NULL)로 지정합니다. 널(NULL) 표시기 컬럼에서 Y *이외의* 모든 문자는 컬

럼 데이터가 널(NULL)이 아니며 METHOD L 옵션이 지정하는 컬럼 데이터가 로드됨을 지정합니다.

널(NULL) 표시기 문자는 MODIFIED BY 옵션을 사용하여 변경할 수 있습니다.

N 로드할 데이터 파일의 컬럼 이름을 지정합니다. 이 컬럼 이름의 대소문자는 시스템 카탈로그의 해당 이름의 대소문자와 일치해야 합니다. 널(NULL) 입력이 가능하지 않은 각 테이블 컬럼에는 METHOD N 목록에 해당 항목이 있어야 합니다. 예를 들면, 주어진 데이터 필드 F1, F2, F3, F4, F5 및 F6과 테이블 컬럼 C1 INT, C2 INT NOT NULL, C3 INT NOT NULL 및 C4 INT, `method N (F2, F1, F4, F3)`은 유효한 요청인 반면, `method N (F2, F1)`은 유효하지 않습니다. 이 메소드는 파일 유형이 IXF 또는 CURSOR인 경우에만 사용될 수 있습니다.

P 로드할 입력 데이터 필드의 필드 번호(1부터 번호를 지정)를 지정합니다. 널(NULL) 입력이 가능하지 않은 각 테이블 컬럼에는 METHOD P 목록에 해당 항목이 있어야 합니다. 예를 들면, 주어진 데이터 필드 F1, F2, F3, F4, F5 및 F6 그리고 테이블 컬럼 C1 INT, C2 INT NOT NULL, C3 INT NOT NULL 및 C4 INT, `method P (2, 1, 4, 3)`은 유효한 요청인 반면, `method P (2, 1)`은 유효하지 않습니다. 이 메소드는 파일 유형이 IXF, DEL 또는 CURSOR인 경우에만 사용할 수 있으며 DEL 파일 유형에 유일한 유효 메소드입니다.

XML FROM *xml-path*

XML 파일이 들어 있는 하나 이상의 경로를 지정합니다. XDS는 주 데이터 파일(ASC, DEL 또는 IXF)에 포함되며 XML 컬럼으로 로드될 컬럼에 있습니다.

XMLPARSE

XML 문서가 구문 분석되는 방법을 지정합니다. 이 옵션을 지정하지 않을 경우, XML 문서에 대한 구문 분석 동작은 CURRENT XMLPARSE OPTION 특수 레지스터의 값으로 판별됩니다.

STRIP WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하도록 지정합니다.

PRESERVE WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하지 않도록 지정합니다.

XMLVALIDATE

XML 문서가 스키마에 대해 유효성이 확인되도록 지정합니다(해당되는 경우).

USING XDS

기본 데이터 파일의 XDS(XML Data Specifier)에 의해 식별된 XML

스키마에 대해 XML 문서의 유효성을 확인합니다. USING XDS와 함께 XMLVALIDATE 옵션을 호출한 경우, 유효성 확인을 수행하는 데 사용 된 스키마는 디폴트로 XDS의 SCH 속성에 의해 판별됩니다. SCH 속성이 XDS에 존재하지 않을 경우, DEFAULT 절로 디폴트 스키마를 지정하지 않으면 스 키마 유효성 확인이 발생하지 않습니다.

DEFAULT, IGNORE 및 MAP 절은 스키마 판별 동작을 수정하는 데 사용될 수 있습니다. 이들 세 개의 선택적 절은 XDS의 권장 스펙에 직접적으로 적용되며 서로에게는 적용되지 않습니다. 예를 들어, 한 스키마가 DEFAULT 절에서 지정되어 선택되었으면 이 스키마는 IGNORE 절에 지정되어도 무시되지 않습니다. 마찬가지로 한 스키마가 MAP 절에서 첫 번째 파트 쌍으로 지정되어 선택되면 다른 MAP 절 쌍의 두 번째 파트에 지정되어도 다시 맵핑되지 않습니다.

USING SCHEMA *schema-sqlid*

XML 문서가 지정된 SQL ID가 있는 XML 스키마에 대해 유효성이 확인됩니다. 이 경우 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

USING SCHEMALOCATION HINTS

XML 문서가 소스 XML 문서의 XML 스키마 위치 힌트에 의해 식별된 스키마에 대해 유효성이 확인됩니다. XML 문서에서 schemaLocation 속성을 찾을 수 없으면 유효성 확인이 발생하지 않습니다. USING SCHEMALOCATION HINTS 절을 지정하면 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

아래의 XMLVALIDATE 옵션 예를 참조하십시오.

IGNORE *schema-sqlid*

이 옵션은 USING XDS 매개변수가 지정된 경우에만 사용할 수 있습니다. SCH 속성에 의해 식별될 경우 IGNORE 절은 무시할 하나 이상의 스키마 목록을 지정합니다. 로드된 XML 문서에 대한 XDS(XML Data Specifier)에 SCH 속성이 존재하고 SCH 속성에 의해 식별된 스키마가 IGNORE할 스키마 목록에 포함된 경우, 이 로드된 XML 문서에 대해서는 스키마 유효성 확인이 발생하지 않습니다.

주:

IGNORE 절에 스키마가 지정되어 있으면 이 스키마는 또한 MAP 절에 있는 스키마 쌍의 왼쪽에 존재할 수 없습니다.

IGNORE 절은 XDS에만 적용됩니다. MAP 절에 의해 맵핑된 스키마는 IGNORE 절에 의해 지정된 경우 계속 무시되지 않습니다.

DEFAULT *schema-sqlid*

이 옵션은 USING XDS 매개변수가 지정된 경우에만 사용할 수 있습니다. DEFAULT 절을 통해 지정된 스키마는 로드된 XML 문서의 XDS(XML Data Specifier)가 XML 스키마를 식별하는 SCH 속성을 포함하지 않을 때 유효성 확인에 사용할 스키마를 식별합니다.

DEFAULT 절은 IGNORE 및 MAP 절보다 우선순위를 갖습니다. XDS가 DEFAULT 절을 충족시키면 IGNORE 및 MAP 스펙은 무시됩니다.

MAP *schema-sqlid*

이 옵션은 USING XDS 매개변수가 지정된 경우에만 사용할 수 있습니다. 로드된 각 XML 문서에 대한 XDS(XML Data Specifier)의 SCH 속성이 지정하는 스키마 대신 사용할 대체 스키마를 지정하려면 이 MAP 절을 사용하십시오. MAP 절은 하나 이상의 스키마 쌍 목록을 지정하며, 여기서 각 쌍은 한 스키마 대 다른 스키마의 맵핑을 나타냅니다. 쌍에서 첫 번째 스키마는 XDS에 있는 SCH 속성에 의해 참조되는 스키마를 나타냅니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

MAP 절에 있는 스키마 쌍의 왼쪽에 있는 스키마는 IGNORE 절에 지정될 수 없습니다.

스키마 쌍 맵핑이 적용된 후, 최종 결과가 됩니다. 맵핑 조작은 전이되지 않으므로 선택된 스키마는 다른 스키마 상 맵핑에 계속 적용되지 않습니다.

스키마가 두 번 이상 맵핑될 수 없다는 것은 쌍의 왼쪽에 두 번 이상 나타날 수 없다는 것을 의미합니다.

SAVECOUNT *n*

로드 유틸리티가 *N*개의 행 뒤에 매번 일관성 지점을 설정하도록 지정합니다. 이 값은 쪽 수로 변환되며 Extent 크기의 간격으로 반올림됩니다. 메시지는 각각의 일관성 지점에서 발행되므로 LOAD QUERY를 사용하여 로드 조작을 모니터하는 경우 이 옵션을 선택해야 합니다. *N* 값이 충분히 높지 않으면 각 일관성 지점에서 수행된 활동의 동기화가 성능에 영향을 줍니다.

디폴트값은 0으로, 필요하지 않으면 일관성 지점을 설정하지 않음을 의미합니다.

CURSOR 파일 유형과 함께 지정되거나 XML 컬럼을 포함하는 테이블 로드 시 이 옵션은 무시됩니다.

ROWCOUNT *n*

로드될 파일에 있는 *N*개의 실제 레코드 수를 지정합니다. 사용자가 첫 번째 *N*개 행만 로드할 수 있도록 합니다.

WARNINGCOUNT *n*

*N*번의 경고 후에 로드 조작을 중지합니다. 경고가 예상되지는 않지만 올바른

파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 로드 파일이나 목표 테이블이 잘못 지정된 경우, 로드 유틸리티가 로드를 시도하는 각 행에 대해 경고를 생성하므로 로드에 실패하게 됩니다. *N*이 0이거나 이 옵션을 지정하지 않으면 발행된 경고 수에 관계없이 로드 조작을 계속합니다. 경고의 임계값에 도달하여 로드 조작이 중지된 경우, 다른 로드 조작이 RESTART 모드로 시작될 수 있습니다. 로드 조작은 마지막 일관성 지점에서 자동으로 계속됩니다. 또는 다른 로드 조작을 REPLACE 모드로 초기화하여 입력 파일의 처음부터 시작하게 할 수 있습니다.

TEMPFILES PATH *temp-pathname*

로드 조작 중에 임시 파일을 작성할 때 사용되는 경로의 이름을 지정하며 서버 데이터베이스 파티션에 따라 완전한 이름이어야 합니다.

임시 파일은 파일 시스템 스페이스를 차지합니다. 때때로 이 스페이스는 대량으로 요구됩니다. 다음은 모든 임시 파일에 할당되어야 하는 파일 시스템 스페이스의 예상 크기입니다.

- 로드 유틸리티가 생성하는 각각의 메시지 당 136바이트
- 데이터 파일에 Long 필드나 LOB가 포함된 경우 15KB의 오버헤드. 이 수량은 INSERT 옵션이 지정되어 있고 테이블에 대량의 Long 필드나 LOB 데이터가 있는 경우 급격히 증가할 수 있습니다.

INSERT

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 기존 테이블 데이터를 변경하지 않고 로드된 데이터를 테이블에 추가합니다.

REPLACE

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 테이블에서 기존 데이터를 모두 삭제하고 로드된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다. 계층 구조 간에 데이터를 이동하는 경우에 이 옵션을 사용하면 개별 부속 테이블이 아닌 전체 계층 구조에 대한 데이터만 바뀔 수 있습니다.

KEEPDICTIONARY

기존 압축 사전은 LOAD REPLACE 조작에서 보존됩니다. 테이블 COMPRESS 속성이 YES로 제공되면, 새로 교체된 데이터는 로드의 호출 이전에 존재하는 사전을 사용하여 압축됩니다. 테이블에 이미 존재하는 사전이 없는 경우, 테이블 COMPRESS 속성이 YES이면 테이블로 교체 중인 데이터를 사용하여 새 사전이 빌드됩니다. 이 경우 압축 사전을 빌드하는 데 필요한 데이터의 양은 ADC의 규정에 따릅니다. 이 데이터는 압축되지 않은 채로 테이블에 채워집니다. 사전이 테이블에 삽입되면, 로드되는 남아 있는 데이터는 이 사전과 함께 압축됩니다. 이는 디폴트 매개변수입니다. 요약 정보는 아래 표를 참조하십시오.

표 27. LOAD REPLACE KEEPDICTIONARY

압축	테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전 존재 여부[1]	압축 사전	데이터 압축
YES	YES	YES	테이블 행 데이터 및 XML 사전을 보존합니다.	로드되는 데이터는 압축됩니다.
YES	YES	NO	테이블 행 데이터 사전을 보존하고 새 XML 사전을 빌드합니다.	로드되는 테이블 행 데이터는 압축됩니다. XML 사전이 빌드된 후에, 로드되는 남아 있는 XML 데이터는 압축됩니다.
YES	NO	YES	테이블 행 데이터 사전을 빌드하고 새 XML 사전을 보존합니다.	테이블 행 데이터 사전이 빌드된 후에, 로드되는 남아 있는 테이블 행 데이터는 압축됩니다. 로드되는 XML 데이터는 압축됩니다.
YES	NO	NO	새 테이블 행 데이터 및 XML 사전을 빌드합니다.	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
NO	YES	YES	테이블 행 데이터 및 XML 사전을 보존합니다.	로드되는 데이터는 압축되지 않습니다.
NO	YES	NO	테이블 행 데이터 사전을 보존합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	YES	테이블 행 사전에 영향을 미치지 않습니다. XML 사전을 보존합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	NO	영향을 미치지 않습니다.	로드되는 데이터는 압축되지 않습니다.

주:

1. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.

RESETDICTIONARY

이 지시문은 LOAD REPLACE 처리 시 테이블 COMPRESS 속성이 YES인 경우 테이블 데이터 오브젝트의 새 사전을 빌드하도록 합니다. COMPRESS 속성이 NO이고 사전이 이미 테이블에 있으면 제거되며 새 사전은 테이블로 삽입되지 않습니다. 압축 사전은 오직 하나의 사용자 레코드로 빌드될 수 있습니다. 로드된 데이터 세트 크기가 0이고 기존에 존재하는 사전이 있는 경우, 이 사전은 보존되지 않습니다. 이 지시문으로 사전을 빌드하도록 요구되는 데이터의 양은 ADC의 규정에 따르지 않습니다. 요약 정보는 아래 테이블 2를 참조하십시오.

표 28. LOAD REPLACE RESETDICTIONARY

압축	테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전 존재 여부[1]	압축 사전	데이터 압축
YES	YES	YES	새 사전을 빌드합니다[2]. DATA CAPTURE CHANGES 옵션이 CREATE TABLE 또는 ALTER TABLE문에서 사용 가능하면, 현재 테이블 행 데이터 사전이 보존됩니다(*실행기록 압축 사전*이라고도 함).	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
YES	YES	NO	새 사전을 빌드합니다[2]. DATA CAPTURE CHANGES 옵션이 CREATE TABLE 또는 ALTER TABLE문에서 사용 가능하면, 현재 테이블 행 데이터 사전이 보존됩니다(*실행기록 압축 사전*이라고도 함).	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
YES	NO	YES	새 사전을 빌드합니다.	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
YES	NO	NO	새 사전을 빌드합니다.	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
NO	YES	YES	사전을 제거합니다.	로드되는 데이터는 압축되지 않습니다.
NO	YES	NO	테이블 행 데이터 사전을 제거합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	YES	XML 스토리지 오브젝트 사전을 제거합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	NO	영향을 미치지 않습니다.	모든 테이블 데이터는 압축되지 않습니다.

주:

1. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.
2. 사전이 존재하고 압축 속성이 사용 가능하지만 테이블 파티션에 로드할 레코드가 없는 경우, 새 사전은 빌드될 수 없으며 RESETDICTIONARY 조작은 기존의 사전을 보존하지 않습니다.

TERMINATE

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 일관성 지점이 패스된 경우라도 이전에 인터럽트된 로드 조작을 종료하고 조작이 시작된 특정 시점으로 롤백합니다. 조작에 관계된 테이블 스페이스의 상태가 정상으로 리턴되고 모

든 테이블 오브젝트는 일관성을 갖게 됩니다(인덱스 오브젝트가 유효하지 않은 것으로 표시되는 경우, 인덱스 재빌드가 다음 액세스에서 자동으로 발생). 종료되는 로드 조작이 LOAD REPLACE이면, 테이블은 LOAD TERMINATE 조작 이후에 빈 테이블로 절단됩니다. 종료되는 로드 조작이 LOAD INSERT이면, LOAD TERMINATE 조작 이후에 테이블은 원래의 모든 레코드를 보유합니다. 사전 관리 요약 정보는 아래 테이블 3을 참조하십시오.

LOAD TERMINATE 옵션은 테이블 스페이스에서 백업 보류 상태를 제거하지 않습니다.

RESTART

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 이전에 인터럽트된 로드 조작을 재시작합니다. 로드 조작은 로드, 빌드 또는 삭제 단계의 마지막 일관성 지점에서 자동으로 계속됩니다. 사전 관리 요약 정보는 아래 테이블 4를 참조하십시오.

INTO *table-name*

데이터가 로드되어야 할 데이터베이스 테이블을 지정합니다. 이 테이블은 시스템 테이블, 선언된 임시 테이블 또는 작성된 임시 테이블일 수 없습니다. 별명 또는 완전하거나 규정에 맞지 않는 테이블 이름을 지정할 수 있습니다. 규정된 테이블 이름의 양식은 `schema.tablename`입니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

insert-column

데이터가 삽입되어야 할 테이블 컬럼을 지정합니다.

로드 유틸리티는 하나 이상의 스페이스를 포함하는 이름를 가진 컬럼의 구문을 분석할 수 없습니다. 예를 들면, 다음과 같습니다.

`Int 4` 컬럼 때문에 실패합니다. 솔루션은 이 컬럼 이름을 큰따옴표 안에 두는 것입니다.

FOR EXCEPTION *table-name*

오류 행이 복사될 예외 테이블을 지정합니다. 고유 인덱스 또는 기본 키를 위반하는 모든 행이 복사됩니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

예외 테이블에 작성된 정보는 덤프 파일에는 작성되지 *않습니다*. 파티션된 데이터베이스 환경에서는 예외 테이블이 로드 중인 테이블이 정의된 데이터베이스 파티션에 대해 정의되어야 합니다. 그렇지 않으면, 덤프 파일에는 유효하지 않거나 구문 오류 때문에 로드될 수 없는 행이 포함됩니다.

XML 데이터 로드 시에는 다음과 같은 경우 FOR EXCEPTION절을 사용하여 로드 예외 테이블을 지정할 수 없습니다.

- 레이블 기반 액세스 제어(LBAC)를 사용하는 경우.
- 데이터 파티션 테이블로 데이터를 로드하는 경우.

NORANGEEXC

행이 범위 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

NOUNIQUEEXC

행이 고유 제한조건 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

STATISTICS USE PROFILE

이 테이블에 대해 정의된 프로파일에 따라 로드 중 통계를 수집하도록 로드를 지시합니다. 이 프로파일은 로드가 실행되기 전에 작성되어야 합니다. 이 프로파일은 RUNSTATS 명령으로 작성됩니다. 프로파일이 존재하지 않고 프로파일에 따라 통계를 수집하도록 로드에 지시된 경우 경고가 리턴되며 통계가 수집되지 않습니다.

STATISTICS NO

통계가 수집되지 않고 카탈로그의 통계가 변경되지 않도록 지정합니다. 이는 디폴트값입니다.

COPY NO

포워드 복구가 사용 가능한 경우(즉, *logretain* 또는 *userexit*이 설정된 경우), 테이블이 있는 테이블 스페이스가 백업 보류 상태에 있도록 지정합니다. COPY NO 옵션을 사용하면 테이블 스페이스 상태를 로드 진행 중 테이블 스페이스 상태로 둡니다. 이 상태는 로드가 완료되거나 중단되면 소멸되는 일시적인 상태입니다. 테이블 스페이스의 모든 테이블의 데이터는 테이블 스페이스 백업 또는 전체 데이터베이스 백업이 완료되어야 갱신하거나 삭제할 수 있습니다. 그러나 SELECT문을 사용하여 테이블의 데이터에 액세스할 수 있습니다.

복구 가능한 데이터베이스에 대해 COPY NO 옵션이 지정된 LOAD는 테이블 스페이스를 백업 보류 상태로 둡니다. 예를 들어, COPY NO 및 INDEXING MODE DEFERRED 옵션과 함께 LOAD를 수행하면 인덱스를 새로 고쳐야 합니다. 테이블에 대한 특정 쿼리는 인덱스 스캔을 해야 하며 인덱스를 새로 고쳐야 성공할 수 있습니다. 백업 보류 상태에 있는 테이블 스페이스에 위치한 인덱스는 새로 고칠 수 없습니다. 이러한 경우, 백업을 수행해야 테이블에 대한 액세스가 허용됩니다. 인덱스 새로 고침은 쿼리로 인덱스를 액세스할 때 데이터베이스에 의해 자동으로 완료됩니다. COPY NO, COPY YES 또는 NONRECOVERABLE 중 하나가 지정되지 않고 데이터베이스가 복구 가능한 경우(**logretain** 또는 **logarchmeth1**이 사용 가능함), COPY NO가 디폴트입니다.

COPY YES

로드된 데이터의 사본이 저장되도록 지정합니다. 포워드 복구가 사용 불가능한 경우 이 옵션은 유효하지 않습니다.

USE TSM

TSM(Tivoli Storage Manager)을 사용하여 사본이 저장되도록 지정합니다.

OPEN *num-sess* **SESSIONS**

TSM 또는 벤더 제품과 함께 사용되는 입출력 세션 수. 디폴트값은 1입니다.

TO *device/directory*

이미지 복사가 작성될 디바이스 또는 디렉토리를 지정합니다.

LOAD *lib-name*

사용될 벤더 백업 및 리스토어 I/O 기능이 들어 있는 공유 라이브러리 이름(Windows 운영 체제의 DLL). 여기에는 전체 경로가 포함될 수 있습니다. 전체 경로가 지정되지 않으면 디폴트값은 User Exit 프로그램이 있는 경로입니다.

NONRECOVERABLE

로드 트랜잭션이 복구 불가능하도록 표시하여 후속적인 롤 포워드 조치로 복구할 수 없도록 지정합니다. 롤 포워드 유틸리티가 트랜잭션을 건너뛰고 데이터가 로드 중인 테이블을 "유효하지 않음"으로 표시합니다. 또한 이 유틸리티는 해당 테이블에 대한 후속 트랜잭션을 무시합니다. 롤 포워드 조작이 완료된 후 이러한 테이블은 복구 불가능한 로드 조작 완료 후 커미트 지점 다음에 수행되는 백업(전체 또는 테이블 스페이스)에서 삭제되거나 리스토어될 수 있습니다.

이 옵션을 사용하면 로드 조작 후에 테이블 스페이스가 백업 보류 상태가 되지 않으며 로드 조작 중에 로드된 데이터의 사본을 작성할 필요가 없습니다. COPY NO, COPY YES 또는 NONRECOVERABLE 중 하나가 지정되지 않고 데이터베이스가 복구 가능하지 않은 경우(**logretain** 또는 **logarchmeth1**이 사용 가능하지 않음), NONRECOVERABLE이 디폴트입니다.

WITHOUT PROMPTING

데이터 파일의 목록에 로드될 모든 파일이 포함되고 나열된 디바이스 또는 디렉토리가 전체 로드 조작에 충분하도록 지정합니다. 연속 입력 파일을 찾을 수 없거나 목표 복사가 로드 조작이 완료되기 전에 채워지면 로드 조작에 실패하고 테이블이 로드 보류 상태가 됩니다.

DATA BUFFER *buffer-size*

유틸리티에서 데이터 전송에 필요한 버퍼 스페이스로 사용할 4KB 페이지의 수를 지정합니다(병렬 처리 수준과 무관함). 지정된 값이 알고리즘의 최소보다 작으면, 최소 필요 자원이 사용되고 경고는 리턴되지 않습니다.

이 메모리는 유틸리티 힙에서 바로 할당되며 크기는 *util_heap_sz* 데이터베이스 구성 매개변수로 수정할 수 있습니다.

값이 지정되지 않으면 런타임시 유틸리티에 의해 적절한 디폴트값이 계산됩니다. 디폴트값은 테이블의 일부 등록 정보 뿐 아니라 로더의 인스턴스화 시간에 유틸리티 힙에서 사용 가능한 여유 공간의 백분율을 기초로 합니다.

SORT BUFFER *buffer-size*

이 옵션은 로드 조작 중 SORTHEAP 데이터베이스 구성 매개변수를 겹쳐쓰는 값을 지정합니다. 이 옵션은 인덱스와 함께 테이블을 로드하는 경우와 INDEXING MODE 매개변수가 DEFERRED로 지정되지 않은 경우에만 의미가 있습니다. 지정된 값은 SORTHEAP의 값을 초과할 수 없습니다. 이 매개변수는 SORTHEAP의 값을 변경하지 않고 많은 인덱스가 있는 테이블을 로드할 때 사용되는 정렬 메모리를 조절하는 데 유용하며 일반 쿼리 처리에도 영향을 줍니다.

CPU_PARALLELISM *n*

테이블 오브젝트를 빌드할 때 레코드를 구문 분석, 변환 및 포맷팅하기 위해 로드 유틸리티가 작성하는 프로세스 또는 스레드의 수를 지정합니다. 이 매개변수는 데이터베이스 파티션당 실행하는 프로세스 수를 이용하도록 설계되었습니다. 소스 데이터에 있는 기록 순서가 보존되므로 사전에 정렬된 데이터를 로드할 때 특히 유용합니다. 이 매개변수의 값이 0이거나 지정되지 않으면 로드 유틸리티는 적절한 디폴트값(일반적으로 사용 가능한 CPU 수에 기초함)을 사용합니다.

주:

1. 이 매개변수는 LOB 또는 LONG VARCHAR 필드 중 하나가 있는 테이블과 함께 사용되면 시스템 CPU 수나 사용자가 지정한 값과 관계없이 값은 1이 됩니다.
2. SAVECOUNT 매개변수에 값을 작게 지정하면 로더는 데이터 및 테이블 메타데이터를 둘 다 비우기 위해 훨씬 더 많은 입출력 조작을 수행하게 됩니다. CPU_PARALLELISM이 1보다 크면 플러시 조작은 비동기로 수행되며 로더가 CPU를 이용할 수 있도록 합니다. CPU_PARALLELISM이 1로 설정되면 로더는 일관성 지점 동안 입출력을 대기합니다. CPU_PARALLELISM이 2로 설정되고 SAVECOUNT가 10 000으로 설정된 로드 조작은 CPU가 하나만 있어도 CPU_PARALLELISM이 1로 설정된 동일한 조작보다 빠르게 완료됩니다.

DISK_PARALLELISM *n*

로드 유틸리티가 데이터를 테이블 스페이스 컨테이너에 작성하기 위해 작성하는 프로세스나 스레드의 수를 지정합니다. 값이 지정되지 않으면 유틸리티가 테이블 스페이스 컨테이너의 수 및 테이블의 등록 정보를 기반으로 하여 적절한 디폴트값을 선택합니다.

FETCH_PARALLELISM YES | NO

DATABASE 키워드로 선언된 커서에서 로드를 수행하거나 API `sqlu_remotefetch_entry` 미디어 항목을 사용하고 이 옵션이 YES로 설정되어 있는 경우, 로드 유틸리티는 가능한 리모트 데이터 소스로부터 페치(fetch)를 병렬 처리하려고 합니다. NO로 설정되면 병렬 페치(fetch)는 수행되지 않습니다. 디폴트값은 YES입니다. 자세한 정보는 *CURSOR 파일 유형을 사용하여 데이터 이동*을 참조하십시오.

INDEXING MODE

로드 유틸리티가 인덱스를 재빌드할 것인지 또는 인덱스를 점차적으로 확장할 것인지를 지정합니다. 가능한 값은 다음과 같습니다.

AUTOSELECT

로드 유틸리티는 자동으로 REBUILD나 INCREMENTAL 모드를 결정합니다. 이 결정은 로드 중인 데이터 양과 인덱스 트리의 용량을 기반으로 합니다. 인덱스 트리의 용량에 관한 정보는 인덱스 오브젝트에 저장되어 있습니다. 이 정보를 채우기 위해 RUNSTATS는 필요하지 않습니다. 디폴트 인덱스 모드는 AUTOSELECT입니다.

REBUILD

모든 인덱스가 재빌드됩니다. 이전 및 추가된 테이블 데이터에 대해 모든 인덱스 키 파트를 정렬하려면 유틸리티에 충분한 자원이 있어야 합니다.

INCREMENTAL

인덱스는 새로운 데이터와 함께 확장됩니다. 이 방법은 인덱스 여유 공간을 사용합니다. 삽입된 레코드에 대한 인덱스 키를 추가하려면 충분한 정렬 스페이스만 있으면 됩니다. 이 메소드는 인덱스 오브젝트가 유효하고 로드 조작이 시작될 때 액세스할 수 있는 경우에만 지원 됩니다(예를 들어, DEFERRED 모드가 지정된 로드 조작 직후에는 유효하지 않음). 이 모드가 지정되었지만 인덱스 상태 때문에 지원되지 않는 경우, 경고가 리턴되며 로드 조작은 계속 REBUILD 모드입니다. 마찬가지로 로드 빌드 단계에서 로드 재시작 조작이 시작된 경우, INCREMENTAL 모드는 지원되지 않습니다.

다음 조건에 모두 해당될 때 증분 인덱싱은 지원되지 않습니다.

- LOAD COPY 옵션이 지정되었습니다(`USEREXIT` 또는 `LOGRETAIN` 옵션을 사용하여 *logarchmeth1*).
- 테이블이 DMS 테이블 스페이스에 있습니다.
- 인덱스 오브젝트가 로드 중인 테이블에 속한 다른 테이블 오브젝트가 공유하는 테이블에 있습니다.

이 제한사항을 통과하려면 인덱스를 별도의 테이블 스페이스에 넣는 것이 바람직합니다.

DEFERRED

이 모드를 지정하면 로드 유틸리티가 인덱스 작성을 시도하지 않습니다. 인덱스는 새로 고칠 필요가 있는 것으로 표시됩니다. 로드 조작과 관련이 없는 이러한 인덱스에 처음 액세스할 때 강제로 재빌드될 수 있습니다. 그렇지 않으면 인덱스는 데이터베이스가 재시작될 때 재빌드될 수 있습니다. 이러한 방법을 수행하려면 가장 큰 인덱스의 모든 키 부분에 대해 충분한 정렬 스페이스가 있어야 합니다. 인덱스 구성에 연속적으로 사용되는 총 시간은 REBUILD 모드에 필요한 시간보다 깁니다. 따라서 지연된 인덱싱으로 다중 로드 조작을 수행할 때 로드하지 않는 액세스에서 먼저 인덱스를 재빌드하게 하는 것보다는 시퀀스에서 마지막 로드 조작에서 인덱스 재빌드를 수행하게 하는 것이 바람직합니다(성능 관점에서).

지연된 인덱싱은 비고유 인덱스가 있는 테이블에만 지원되므로 로드 단계 중 삽입된 중복 키는 로드 조작 후 지속되지 않습니다.

ALLOW NO ACCESS

로드가 로드 중 독점 액세스를 위해 목표 테이블을 잠급니다. 로드 중 테이블 상태는 로드 진행 중으로 설정됩니다. ALLOW NO ACCESS는 디폴트 동작입니다. LOAD REPLACE에는 이 옵션만 유효합니다.

테이블에 제한조건이 있을 때 테이블 상태는 로드 진행 중과 함께 무결성 설정 보류로 설정됩니다. 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY문을 사용해야 합니다.

ALLOW READ ACCESS

로드가 목표 테이블을 공유 모드로 잠급니다. 테이블 상태는 로드 진행 중 및 읽기 액세스 모두로 설정됩니다. 판독기는 테이블이 로드되는 동안 데이터의 비델타 부분에 액세스할 수 있습니다. 다시 말해, 테이블 판독기는 로드가 시작되기 전에 있던 데이터에 액세스할 수 있지만 로드되고 있는 데이터는 로드가 완료될 때까지 사용할 수 없습니다. ALLOW READ ACCESS 로드의 LOAD TERMINATE 또는 LOAD RESTART는 이 옵션을 사용할 수 있지만 ALLOW NO ACCESS 로드의 LOAD TERMINATE 또는 LOAD RESTART는 이 옵션을 사용할 수 없습니다. 게다가 목표 테이블의 인덱스가 재빌드가 필요한 것으로 표시된 경우 이 옵션은 유효하지 않습니다.

테이블에 제한조건이 있으면 테이블 상태는 로드 진행 중, 읽기 액세스뿐만 아니라 무결성 설정 보류로 설정됩니다. 로드가 종료되면 테이블 상태 중에서 로드 진행 중 상태는 제거되지만 무결성 설정 보류 및 읽기 액세스 상태는 남아 있습니다. 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY 문을 사용해야 합니다. 테이블의 상태가 무결성 설정 보류 및 읽기 액세스인 경

우, 판독기는 데이터의 비델타 부분에는 여전히 액세스할 수 있으나 SET INTEGRITY문이 완료될 때까지 데이터의 새 (델타) 부분에는 액세스할 수 없습니다. 사용자는 SET INTEGRITY문을 발행하지 않고 동일한 테이블에서 다중 로드를 수행할 수 있습니다. 그러나 SET INTEGRITY문이 발행될 때까지 원본(확인된) 데이터만 볼 수 있습니다.

ALLOW READ ACCESS는 다음 수정자도 지원합니다.

USE *tablespace-name*

인덱스가 재빌드되는 경우, 인덱스의 쉐도우 사본이 테이블 스페이스 *tablespace-name*에 빌드되고 INDEX COPY PHASE 중 로드 맨 끝에서 원본 테이블 스페이스로 복사됩니다. 이 옵션에는 시스템 임시 테이블 스페이스만 사용할 수 있습니다. 테이블 스페이스를 지정하지 않을 경우 음영 인덱스는 인덱스 오브젝트와 동일한 테이블 스페이스에 작성됩니다. 음영 사본이 인덱스 오브젝트와 동일한 테이블 스페이스에 작성된 경우, 이전 인덱스 오브젝트에 대한 음영 인덱스 오브젝트의 복사는 순간적입니다. 음영 사본이 인덱스 오브젝트와 다른 테이블 스페이스에 있을 경우 실제 복사가 수행됩니다. 상당한 I/O 및 시간이 포함될 수 있습니다. INDEX COPY PHASE 중 로드 맨 끝에서 테이블이 오프라인 상태에 있는 동안 복사가 발생합니다.

이 옵션을 사용하지 않으면 음영 인덱스는 원본과 동일한 테이블 스페이스에서 빌드됩니다. 원본 인덱스와 음영 인덱스가 둘 다 디폴트로 동시에 동일한 테이블 스페이스에 있으므로 한 테이블 스페이스에 이들 인덱스를 모두 보유하기 위한 스페이스가 충분하지 않을 수 있습니다. 이 옵션을 사용하면 인덱스에 충분한 테이블 스페이스를 보유할 수 있습니다.

사용자가 INDEXING MODE REBUILD 또는 INDEXING MODE AUTOSELECT를 지정하지 않을 경우 이 옵션은 무시됩니다. INDEXING MODE AUTOSELECT가 선택되고 로드가 인덱스를 점차적으로 갱신하도록 선택할 경우에도 이 옵션이 무시됩니다.

SET INTEGRITY PENDING CASCADE

LOAD로 인해 테이블이 무결성 설정 보류 상태에 놓일 경우, 사용자는 SET INTEGRITY PENDING CASCADE 옵션을 사용하여 로드된 테이블의 무결성 설정 보류 상태가 모든 하위(하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블 포함)에 즉시 연쇄되는지 여부를 지정할 수 있습니다.

IMMEDIATE

무결성 설정 보류 상태가 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블로 즉시 확장됨을 표시합니다.

LOAD INSERT 조작의 경우, IMMEDIATE 옵션을 지정해도 무결성 설정 보류 상태는 하위 외부 키 테이블로 확장되지 않습니다.

나중에 로드된 테이블에 제한조건 위반이 있는지 여부를 점검할 때(SET INTEGRITY문의 IMMEDIATE CHECKED 옵션을 사용하여) 무결성 설정 보류 읽기 액세스에 있던 하위 외부 키 테이블은 무결성 설정 보류 권한 없음 상태가 됩니다.

DEFERRED

로드된 테이블만이 무결성 설정 보류 상태에 놓임을 표시합니다. 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블의 상태는 변경되지 않습니다.

하위 외부 키 테이블에 제한조건 위반이 있는지 여부를 점검할 때(SET INTEGRITY문의 IMMEDIATE CHECKED 옵션을 사용하여) 하위 외부 키 테이블 은 나중에 내재적으로 무결성 설정 보류 상태에 놓일 수 있습니다. 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블은 해당 기본 테이블 중 하나에 무결성 위반이 있는지 여부를 점검할 때 내재적으로 무결성 설정 보류 상태가 됩니다. 하위 테이블이 무결성 설정 보류 상태가 되었음을 나타내는 경고(SQLSTATE 01586)가 발행됩니다. 이들 하위 테이블이 언제 무결성 설정 보류 상태가 되는지에 대해서는 SQL 참조서에서 SET INTEGRITY문의 참고 절을 참조하십시오.

SET INTEGRITY PENDING CASCADE 옵션을 지정하지 않은 경우,

- 로드된 테이블만이 무결성 설정 보류 상태에 놓입니다. 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블의 상태는 계속 변경되지 않으며 나중에 로드된 테이블에 제한조건 위반이 있는지 점검할 때 내재적으로 무결성 설정 보류 상태가 될 수 있습니다.

LOAD 결과 목표 테이블이 무결성 설정 보류 상태가 되지 않을 경우 SET INTEGRITY PENDING CASCADE 옵션은 무시됩니다.

LOCK WITH FORCE

이 유틸리티는 로드 처리에서 테이블 잠금을 포함하는 다양한 잠금을 획득합니다. 대기라기 보다는 시간종료라고 할 수 있으며, 잠금을 획득할 때 로드는 이 옵션을 통해 목표 테이블에 충돌 잠금을 보유하고 있는 기타 응용프로그램을 강제로 해제할 수 있습니다. 시스템 카탈로그 테이블에서 충돌 잠금을 보유하고 있는 응용프로그램은 로드 유틸리티로 강제로 해제할 수 없습니다. 강제된 응용프로그램은 롤백되고 로드 유틸리티가 필요로 하는 잠금을 릴리스합니다. 그런 다음 로드 유틸리티는 계속 진행될 수 있습니다. 이 옵션을 사용하려면 FORCE APPLICATIONS 명령(SYSADM 또는 SYSCTRL)과 동일한 권한이 필요합니다.

ALLOW NO ACCESS 로드는 로드 조작이 시작될 때 충돌 잠금을 보유하고 있는 응용프로그램을 강제 실행할 수 있습니다. 로드가 시작될 때 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

ALLOW READ ACCESS 로드는 로드 조작이 시작되거나 종료될 때 충돌 잠금을 보유하고 있는 응용프로그램을 강제 실행할 수 있습니다. 로드가 시작될 때 유틸리티는 테이블 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다. 로드 조작이 종료될 때 로드 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

SOURCEUSEREXIT *executable*

유틸리티에 데이터를 입력하기 위해 호출될 실행 파일 이름을 지정합니다.

REDIRECT

INPUT FROM

BUFFER *input-buffer*

*input-buffer*에 지정된 바이트 스트림을 지정된 실행 파일을 실행하는 프로세스의 STDIN 파일 디스크립터로 패스합니다.

FILE *input-file*

이 클라이언트측 파일의 내용을 지정된 실행 파일을 실행하는 프로세스의 STDIN 파일 디스크립터로 패스합니다.

OUTPUT TO

FILE *output-file*

STDOUT 및 STDERR 파일 디스크립터를 지정된 서버측 파일로 캡처합니다.

PARALLELIZE

여러 User Exit 프로세스를 동시 호출하여 로드 유틸리티로 들어가는 데이터의 처리량을 증가시킵니다. 이 옵션은 다중 파티션 데이터베이스 환경에서만 적용할 수 있으므로 단일 파티션 데이터베이스 환경에서는 무시됩니다.

자세한 정보는 *사용자 정의된 응용프로그램을 사용하여 데이터 이동*을 참조하십시오.

PARTITIONED DB CONFIG *partitioned-db-option*

다중 데이터베이스 파티션에 분산된 테이블 로드를 실행할 수 있도록 합니다.

PARTITIONED DB CONFIG 매개변수를 사용하면 파티션된 데이터베이스에만 해당되는 구성 옵션을 지 정할 수 있습니다. *partitioned-db-option* 값은 다음과 같이 지정할 수 있습니다.

```
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

이들 옵션의 자세한 설명은 *파티션된 데이터베이스 환경을 위한 로드 구성 옵션*을 참조하십시오.

RESTARTCOUNT

예약됨.

USING *directory*

예약됨.

XML 문서에서 데이터 로드 예

XML 데이터 로딩

예 1

테이블에 삽입되는 문서를 설명하기 위해 사용자가 XDS 필드로 데이터 파일을 구성했습니다. 다음과 같이 표시됩니다.

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

첫 번째 행의 경우, XML 문서는 `file1.xml` 파일로 식별됩니다. 문자 분리문자는 큰따옴표이며 큰따옴표는 XDS 안에 존재하므로, XDS 안에 포함된 큰따옴표는 이중이 됩니다. 두 번째 행의 경우, XML 문서는 `file2.xml` 파일로 식별되며, 바이트 오프셋 23에서 시작하며 45바이트 길이입니다.

예 2:

사용자는 XML 컬럼의 구문 분석 또는 유효성 확인 옵션 없이 로드 명령을 발행하며, 데이터는 정상적으로 로드됩니다.

```
LOAD FROM data.del of DEL INSERT INTO mytable
```

커서에서 XML 데이터 로딩

커서에서 데이터 로딩은 일반 관계형 컬럼 유형과 동일합니다. 두 개의 테이블 T1 및 T2가 있으며, 각각 C1이라는 단일 XML 컬럼으로 구성됩니다. T1에서 T2로 로드하기 위해 사용자는 우선 커서를 선언합니다.

```
DECLARE X1 CURSOR FOR SELECT C1 FROM T1;
```

다음으로 사용자는 커서 유형을 사용하여 LOAD를 발행할 수 있습니다.

```
LOAD FROM X1 of CURSOR INSERT INTO T2
```

XML 특정 LOAD 옵션을 커서 유형에 적용하는 것은 파일에서 로드하는 것과 같습니다.

사용 시 참고사항

- 데이터는 입력 파일에 나타나는 시퀀스로 로드됩니다. 특정 시퀀스를 원할 경우 로드를 시도하기 전에 데이터를 정렬해야 합니다. 소스 데이터 순서의 보존이 필요하지 않은 경우, *로드 유틸리티의 파일 유형 수정자* 섹션에서 아래 설명된 ANYORDER 파일 유형 수정자 사용을 고려하십시오.
- 로드 유틸리티는 기존 정의에 따라 인덱스를 빌드합니다. 고유 키에 대한 중복을 처리하기 위해 예외 테이블을 사용합니다. 이 유틸리티는 참조 무결성을 강제하거나 제한조건 점검을 수행하거나 로드되는 테이블에 종속된 구체화된 쿼리 테이블을 갱신하지 않습니다. 참조 또는 점검 제한조건을 포함하는 테이블은 무결성 설정 보류 상태에 놓입니다. 로드되고 있는 테이블에 종속되고 REFRESH IMMEDIATE를 사용하여 정의된 요약 테이블도 무결성 설정 보류 상태에 놓입니다. 이들 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY문을 발행하십시오. 복제된 구체화된 쿼리 테이블에서는 로드 조작을 수행할 수 없습니다.
- 테이블에 클러스터링 인덱스가 존재할 경우, 로드하기 전에 클러스터링 인덱스에 대해 데이터를 정렬해야 합니다. 그러나 다차원적으로 클러스터된(MDC) 테이블에 로드할 경우에는 미리 데이터를 정렬할 필요가 없습니다.
- 보호 설정된 테이블에 로드할 때 예외 테이블을 지정하면, 유효하지 않은 보안 레이블로 보호 설정된 행은 이 테이블로 보내집니다. 따라서 예외 테이블에 액세스할 수 있는 사용자는 일반적으로 액세스 권한이 없는 데이터에 액세스할 수 있습니다. 보안 강화를 위해 누구에게 예외 테이블 액세스 권한을 부여할 것인지 유의하고, 행을 수리한 후 로드되고 있는 테이블에 복사한 후에는 곧바로 각 행을 삭제하고 예외 테이블 사용을 완료하면 곧바로 예외 테이블을 삭제하십시오.
- 내부 형식으로 된 보안 레이블에는 줄 바꾸기 문자가 포함될 수 있습니다. DEL 파일 형식을 사용하여 파일을 로드할 경우, 이들 줄 바꾸기 문자는 분리문자로 오인될

수 있습니다. 이러한 문제점이 발생하면 LOAD 명령에 delprioritychar 파일 유형 수정자를 지정하여 분리문자에 대해 이전 디폴트 우선순위를 사용하십시오.

- DECLARE CURSOR 명령 중에 DATABASE 키워드가 지정된 CURSOR 파일 유형을 사용하여 로드 조작을 수행할 경우, 현재(로드 중에) 연결된 데이터베이스에 대해 인증할 때 사용한 사용자 ID 및 암호를 사용하여 소스 데이터베이스 (DECLARE CURSOR 명령의 DATABASE 옵션에 의해 지정됨)에 대해서도 인증합니다. 로드 중인 데이터베이스에 대한 연결용으로 지정된 사용자 ID 및 암호가 없을 경우 DECLARE CURSOR 명령 중에 소스 데이터베이스에 대한 사용자 ID 및 암호를 지정해야 합니다.
- 개별적 파트가 Windows 시스템에서 AIX 시스템으로 복사되는 다중 파트 PC/IXF 파일 로딩이 지원 됩니다. 모든 파일의 이름이 LOAD 명령에서 지정되어야 합니다. 예를 들어, LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1. 논리적으로 분할된 PC/IXF 파일에서 Windows 운영 체제로의 로딩은 지원되지 않습니다.
- 실패한 LOAD 재시작 시, 인덱스의 REBUILD 모드를 사용하도록 BUILD 단계를 강제하는 기존 동작을 따릅니다.
- 데이터베이스 중간에서는 XML 문서를 로드할 수 없으며 이 경우 오류 메시지 SQL1407N이 리턴됩니다.

LOAD TERMINATE 및 LOAD RESTART 사전 관리의 요약

다음 도표는 TERMINATE 지시문 아래에서 LOAD 처리의 압축 사전 관리 동작을 요약합니다.

표 29. LOAD TERMINATE 사전 관리

테이블 COMPRESS 속성	LOAD 이전에 테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전이 LOAD 이전에 존재 여부[1]	TERMINATE: LOAD REPLACE KEEPDICTIONARY 또는 LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	기존 사전을 보존합니다.	어느 쪽 사전도 보존하지 않습니다.[2]
YES	YES	NO	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.[2]
YES	NO	YES	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
YES	NO	NO	어떤 것도 보존하지 않습니다.	어떤 것도 보존하지 않습니다.
NO	YES	YES	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
NO	YES	NO	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
NO	NO	YES	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
NO	NO	NO	아무 것도 수행하지 않습니다.	아무 것도 수행하지 않습니다.

주:

1. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.
2. 테이블에서 데이터 캡처가 사용 가능한 특수한 경우, 테이블 행 데이터 사전이 보존됩니다.

LOAD RESTART는 최종 일관성 지점까지 테이블을 절단합니다. 압축 사전은 LOAD RESTART 처리의 파트로서, 최종 LOAD 일관성 지점이 지정된 시간에 테이블에 존재합니다. 이 경우, LOAD RESTART는 새 사전을 작성하지 않습니다. 가능한 조건에 대한 요약은 아래 테이블 4를 참조하십시오.

표 30. LOAD RESTART 사전 관리

테이블 COMPRESS 속성	**LOAD 일관성 지점 이전 테이블 행 데이터 사전 존재 여부[1]**	**최종 로드 이전에 XML 스토리지 오브젝트 사전 존재 여부[2]**	**RESTART: LOAD REPLACE KEEPDICTIONARY 또는 LOAD INSERT**	**RESTART: LOAD REPLACE RESETDICTIONARY**
YES	YES	YES	기존 사전을 보존합니다.	기존 사전을 보존합니다.
YES	YES	NO	기존의 테이블 행 데이터 사전을 빌드하고 ADC에 따라 XML 사전을 빌드합니다.	기존의 테이블 행 데이터 사전을 빌드하고 XML 사전을 빌드합니다.
YES	NO	YES	ADC에 따른 테이블 행 사전을 빌드합니다. 기존 XML 사전을 보존합니다.	테이블 행 데이터 사전을 빌드합니다. 기존 XML 사전을 보존합니다.
YES	NO	NO	ADC에 따른 XML 사전과 테이블 행 데이터를 빌드합니다.	테이블 행 데이터 및 XML 사전을 빌드합니다.
NO	YES	YES	기존 사전을 보존합니다.	기존 사전을 제거합니다.
NO	YES	NO	기존 테이블 행 데이터 사전을 보존합니다.	기존 테이블 행 데이터 사전을 제거합니다.
NO	NO	YES	기존 XML 사전을 보존합니다.	기존 XML 사전을 제거합니다.
NO	NO	NO	아무 것도 수행하지 않습니다.	아무 것도 수행하지 않습니다.

주:

1. XML 데이터 로드 시, 조작 시작부터 로드 단계 재시작 중 실패하는 로드 조작의 경우 SAVECOUNT 옵션이 무시됩니다.
2. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.

로드 유틸리티의 파일 유형 수정자

표 31. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식

수정자	설명
anyorder	이 수정자는 *cpu_parallelism* 매개변수와 결합하여 사용됩니다. SMP 시스템에서 중요한 추가 성능 이점이 발생하며, 소스 데이터 순서의 보존이 필요하지 않음을 지정합니다. *cpu_parallelism* 의 값이 1이면, 이 옵션은 무시됩니다. 일관성 지점 이후의 응급 복구는 데이터가 시퀀스로 로드되도록 요구하므로 SAVECOUNT > 0인 경우 이 옵션은 지원되지 않습니다.
generatedignore	이 수정자는 생성된 모든 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 로드 유틸리티에 알립니다. 이로 인해 생성된 모든 컬럼의 값이 유틸리티에 의해 생성됩니다. 이 수정자는 `generatedmissing` 또는 `generatedoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
generatedmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 ID 컬럼의 데이터가 포함되지 않는 것으로(NULL 값도 비포함) 가정하므로 각 행의 값을 생성합니다. 이로 인해 생성된 모든 컬럼의 값이 유틸리티에 의해 생성됩니다. 이 수정자는 `generatedignore` 또는 `generatedoverride` 수정자 중 하나와 함께 사용될 수 없습니다.

generatedoverride	이 수정자는 테이블에서 생성되는 모든 컬럼의 사용자 제공 데이터를 승인하도록 로드 유틸리티에 지시합니다(이러한 유형의 컬럼에 대한 일반 규칙과 반대). 다른 데이터베이스 시스템에서 데이터를 이주하거나, ROLLFORWARD DATABASE 명령에서 RECOVER DROPPED TABLE 옵션을 사용하여 복구된 데이터에서 테이블을 로드할 때 유용합니다. 이 수정자가 사용될 때, 데이터가 없는 행이나 널(null) 값을 허용하지 않는 생성된 컬럼의 NULL 데이터는 거부됩니다(SQL3116W). 해당 수정자 사용 시, 테이블이 무결성 설정 보류 상태에 놓입니다. 사용자 제공 값을 검증하지 않고 무결성 설정 보류 상태로부터 테이블을 얻으려면, 로드 조작 이후에 다음 명령을 발행하십시오.
	`SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED`
	무결성 설정 보류 상태로부터 테이블을 얻고 사용자 제공 값의 검증을 강제하려면, 로드 조작 이후에 다음 명령을 발행하십시오.
	`SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.`
	이 수정자가 지정되고 임의의 파티셔닝 키, 차원 키 또는 분산 키에 생성된 컬럼이 있으면, LOAD 명령은 자동으로 수정자를 `generatedignore`로 변환하고 로드를 진행합니다. 이것은 생성된 모든 컬럼 값을 재생성하는 효과를 갖습니다.
	이 수정자는 `generatedmissing` 또는 `generatedignore` 수정자 중 하나와 함께 사용될 수 없습니다.

identityignore	이 수정자는 ID 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 로드 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ID 값이 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 `identitymissing` 또는 `identityoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
identitymissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 ID 컬럼의 데이터를 포함하지 않음(NULL 값도 비포함)을 가정하므로 각 행의 값 을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 `identityignore` 또는 `identityoverride` 수정자 중 하나와 함께 사용될 수 없습니다.

표 31. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
identityoverride	GENERATED ALWAYS로 정의된 ID 컬럼이 로드되는 테이블에 있을 때에만 이 수정자를 사용해야 합니다. 해당 컬럼에 대해 명시적, 널(NULL)이 아닌 데이터를 승인하도록 유틸리티에 지시합니다(이러한 유형의 ID 컬럼에 대한 일반 규칙과 반대). 테이블이 GENERATED ALWAYS로 정의되어야 할 때 다른 데이터베이스 시스템에서 데이터를 이주하거나, ROLLFORWARD DATABASE 명령에서 DROPPED TABLE RECOVERY 옵션을 사용하여 복구된 데이터에서 테이블을 로드할 때 유용합니다. 이 수정자가 사용될 때, 데이터가 없는 행이나 ID 컬럼의 NULL 데이터는 거부됩니다(SQL3116W). 이 수정자는 `identitymissing` 또는 `identityignore` 수정자 중 하나와 함께 사용될 수 없습니다. 이 옵션 사용 시 로드 유틸리티는 테이블의 ID 컬럼에서 값의 고유성을 검증하거나 유지하려고 시도하지 않습니다.
indexfreespace=*x*	*x*는 0 - 99의 정수입니다. 로드로 인덱스를 재빌드할 때 여유 공간으로 남게 되는 각 인덱스 페이지의 백분율입니다. INDEXING MODE INCREMENTAL을 사용하여 로드할 때 이 옵션은 무시됩니다. 페이지의 첫 번째 항목은 제한없이 추가되며, 여유 공간 비율 임계값을 유지하도록 추후 항목이 추가됩니다. 디폴트값은 CREATE INDEX 시간에 사용됩니다. 이 값은 CREATE INDEX문에 지정된 PCTFREE 값보다 우선합니다. indexfreespace 옵션은 인덱스 단말 페이지에만 영향을 미칩니다.
lobsinfile	*lob-path*는 LOB 데이터를 포함하는 파일에 대한 경로를 지정합니다. ASC, DEL 또는 IXF 로드 입력 파일은 LOB 컬럼에 LOB 데이터가 있는 파일의 이름을 포함합니다. 이 옵션은 CURSOR 파일 유형과 함께 지원되지 않습니다. LOBS FROM 절은 『lobsinfile』 수정자가 사용될 때 LOB 파일이 위치하는 곳을 지정합니다. LOBS FROM 절은 내재적으로 LOBSINFILE 동작을 활성화합니다. LOBS FROM 절은 데이터 임포트 중 LOB 파일을 검색하기 위해 경로 목록을 LOAD 유틸리티로 전달합니다. 각 경로는 LLS(Lob Location Specifier)에 의해 포인트되는 최소한 하나의 LOB를 포함하는 최소한 하나의 파일을 데이터 파일에 포함합니다. LLS는 LOB 파일 경로에 저장된 파일에서 LOB 위치의 문자열 표시입니다. LLS의 형식은 *filename.ext.nnn.mmm/*이며, 여기서 *filename.ext*는 LOB를 포함하는 파일의 이름이며, *nnn*은 파일 내에서 LOB의 오프셋을 바이트로 나타낸 것이며, *mmm*은 LOB의 길이를 바이트로 나타낸 것입니다. 예를 들어, 문자열 `db2exp.001.123.456/`가 데이터 파일에 저장되는 경우, LOB는 db2exp.001 파일에서 오프셋 123에 위치하며 456바이트 길이입니다. 널(NULL) LOB를 표시하려면, *크기*를 -1로 입력하십시오. *크기*가 0으로 지정되면, 길이가 0인 LOB로 처리됩니다. 길이가 -1인 널(NULL) LOBS의 경우, 오프셋 및 파일 이름은 무시됩니다. 예를 들어, 널(NULL) LOB의 LLS는 db2exp.001.7.-1/입니다.
noheader	헤더 검증 코드를 건너뜁니다(단일 파티션 데이터베이스 파티션 그룹에 있는 테이블로 조작을 로드하는 경우만 적용 가능). 디폴트 MPP 로드(모드 PARTITION_AND_LOAD)가 단일 파티션 데이터베이스 파티션 그룹에 있는 테이블에 대해 사용되는 경우, 파일에 헤더가 없습니다. 그러므로 헤더 수정자는 필요하지 않습니다. LOAD_ONLY 모드가 사용되는 경우, 파일에 헤더가 있습니다. 헤더가 없는 수정자를 사용할 필요가 있는 유일한 환경은 헤더가 없는 파일을 사용하여 LOAD_ONLY를 수행하려고 하는 경우입니다.
norowwarnings	거부된 행에 대한 모든 경고를 제외시킵니다.

표 31. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
pagefreespace=*x*	*x*는 0 - 100의 정수입니다. 여유 공간으로 남게 되는 각 데이터 페이지의 백분율입니다. 최소 행 크기 때문에 지정된 값이 유효하지 않은 경우, (예를 들어, 길이가 최소한 3000바이트이고, *x* 값이 50인 행), 새 페이지에 행이 위치합니다. 100 값이 지정된 경우, 각 행은 새 페이지에 있습니다. 테이블의 PCTFREE 값은 페이지 당 지정된 여유 공간의 크기를 결정합니다. 로드 조작의 pagefreespace 값이나 테이블의 PCTFREE 값이 설정되지 않은 경우, 유틸리티는 각 페이지에 가능한 많은 스페이스를 채웁니다. pagefreespace로 설정된 값은 테이블에 지정된 PCTFREE 값을 겹쳐씁니다.
rowchangetimestampignore	이 수정자는 행 변경 시간소인 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 로드 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ROW CHANGE TIMESTAMP가 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 `rowchangetimestampmissing` 또는 `rowchangetimestampoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
rowchangetimestampmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 행 변경 시간소인 컬럼의 데이터를 포함하지 않음(NULL 값도 비포함)을 가정하므로 각 행의 값을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 `rowchangetimestampignore` 또는 `rowchangetimestampoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
rowchangetimestampoverride	GENERATED ALWAYS로 정의된 행 변경 시간소인 컬럼이 로드되는 테이블에 있을 때에만 이 수정자를 사용해야 합니다. 해당 컬럼에 대해 명시적, 널(NULL)이 아닌 데이터를 승인하도록 유틸리티에 지시합니다(이러한 유형의 시간소인 컬럼에 대한 일반 규칙 과 반대). 테이블이 GENERATED ALWAYS로 정의되어야 할 때 다른 데이터베이스 시스템에서 데이터를 이주하거나, ROLLFORWARD DATABASE 명령에서 DROPPED TABLE RECOVERY 옵션을 사용하여 복구된 데이터에서 테이블을 로드할 때 유용합니다. 이 수정자가 사용될 때, 데이터가 없는 행이나 ROW CHANGE TIMESTAMP 컬럼의 NULL 데이터는 거부됩니다 (SQL3116W). 이 수정자는 `rowchangetimestampmissing` 또는 `rowchangetimestampignore` 수정자 중 하나와 함께 사용될 수 없습니다. 이 옵션 사용 시 로드 유틸리티는 테이블의 행 시간소인 컬럼에서 값의 고유성을 검증하거나 유지하려고 시도하지 않습니다.

seclabelchar	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 보안 레이블 값의 문자열 형식임을 표시합니다. LOAD는 각 보안 레이블을 로드된 대로의 내부 형식으로 변환합니다. 문자열이 적절한 형식으로 되어 있지 않은 경우 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3242W)가 리턴됩니다. 문자열이 테이블을 보호하는 보안 규정 파트인 유효한 보안 레이블을 나타내지 않는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3243W)가 리턴됩니다.
	
	seclabelname 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 로드에 실패하며 오류(SQLCODE SQL3525N)가 리턴됩니다.
	
	테이블이 단일 `DB2SECURITYLABEL` 컬럼으로 구성된 경우, 데이터 파일은 다음과 같습니다.
	`"CONFIDENTIAL:ALPHA:G2"`
	`"CONFIDENTIAL;SIGMA:G2"`
	`"TOP SECRET:ALPHA:G2"`
	
	이 데이터를 로드하거나 임포트하려면, SECLABELCHAR 파일 유형 수정자가 사용되어야 합니다.
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1`

표 31. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명

seclabelname	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 이름으로 표시됨을 나타냅니다. LOAD는 이름이 있는 경우 적절한 보안 레이블로 변환합니다. 테이블을 보호하는 보안 규정에 대해 표시된 이름이 있는 보안 레이블이 없는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3244W)가 리턴됩니다.
	
	seclabelchar 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 로드에 실패하며 오류(SQLCODE SQL3525N)가 리턴됩니다.
	
	테이블이 단일 DB2SECURITYLABEL 컬럼으로 구성된 경우, 데이터 파일은 다음과 유사한 보안 레이블 이름으로 구성됩니다.
	
	`"LABEL1"`
	`"LABEL1"`
	`"LABEL2"`
	
	이 데이터를 로드하거나 임포트하려면, SECLABELNAME 파일 유형 수정자가 사용되어야 합니다.
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1`
	
	주: 파일 유형이 ASC인 경우, 보안 레이블의 이름 다음에 오는 모든 스페이스는 이름의 일부로 해석됩니다. 이를 피하려면 striptblanks 파일 유형 수정자를 사용하여 스페이스가 제거되었는지 확인하십시오.

totalfreespace=*x*	*x*는 0 이상의 정수입니다. 이 값은 여유 공간으로 테이블의 끝에 추가되는 테이블에서 총 페이지의 백분율로 해석됩니다. 예를 들어, *x*가 20이며 데이터가 로드된 이후에 테이블에 100개의 데이터 페이지가 있는 경우, 20개의 추가적인 빈 페이지가 추가됩니다. 테이블의 총 데이터 페이지 수는 120입니다. 데이터 페이지 총 수는 테이블에서 인덱스 페이지의 수를 계산에 넣지 않습니다. 이 옵션은 인덱스 오브젝트에 영향을 미치지 않습니다. 이 옵션이 지정된 두 개의 로드가 완료되면, 두 번째 로드는 첫 번째 로드에서 끝에 추가한 여유 스페이스를 재사용하지 않습니다.
usedefaults	목표 테이블 컬럼의 소스 컬럼이 지정되었지만 하나 이상의 행 인스턴스에 대한 데이터를 포함하지 않은 경우, 디폴트값이 로드됩니다. 누락된 데이터의 예: • DEL 파일: 임의 수의 스페이스(″, ,″)로 구분되는 2개의 인접 컬럼 분리문자나 2개의 인접 컬럼 분리문자(″,,″)가 컬럼 값으로 지정됩니다. • DEL/ASC/WSF 파일: 행의 컬럼이 충분하지 않거나 행의 길이가 원래 스펙만큼 충분하지 않습니다. ASC 파일의 경우, NULL 컬럼 값은 명시적으로 누락된 것으로 간주되지 않으며 디폴트가 NULL 컬럼 값을 대신하지 않습니다. 숫자, 날짜, 시간 및 /시간소인 컬럼의 모든 공백 문자로 NULL 컬럼 값을 표시하거나, 컬럼이 NULL임을 표시하기 위해 모든 유형의 컬럼에 널(NULL) 표시기를 사용함으로써 NULL 컬럼 값을 표시합니다. 이 옵션이 없는 경우, 소스 컬럼이 행 인스턴스의 데이터를 포함하지 않으면 다음 중 하나가 발생합니다. • DEL/ASC/WSF 파일: 컬럼에 널(NULL) 입력 가능한 경우 NULL이 로드됩니다. 컬럼이 널(NULL) 입력 가능하지 않으면 유틸리티가 행을 거부합니다.

표 32. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL)

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 입력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 로드 조작 중 이 코드 페이지에서 데이터베이스 코드 페이지로 문자 데이터(및 문자로 지정된 숫자 데이터)를 변환합니다. 다음 규칙이 적용됩니다. • 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. • EBCDIC 코드 페이지에 지정된 DEL 데이터의 경우 분리문자는 시프트 인(Shift-In) 및 시프트 아웃(Shift-Out) DBCS 문자와 동시에 사용할 수 없습니다. • nullindchar은 x20 - x7F 코드 포인트의 표준 ASCII 세트에 포함된 기호를 지정해야 합니다. 이것은 ASCII 기호 및 코드 포인트를 나타냅니다. 코드 포인트가 달라도 EBCDIC 데이터는 해당 기호를 사용할 수 있습니다. 이 옵션은 CURSOR 파일 유형과 함께 지원되지 않습니다.

dateformat="*x*"	*x*는 소스 파일에서 날짜의 형식입니다.[1] 유효한 날짜 요소는 다음과 같습니다.
	
	`YYYY - 연도(0000 - 9999 사이의 4자리 숫자)`
	`M - 월(1 - 12 사이의 1 또는 2자리 숫자)`
	`MM - 월(1 - 12 사이의 2자리 숫자, M과 상호 배타적)`
	`D - 일(1 - 31 사이의 1 또는 2자리 숫자)`
	`DD - 일(1 - 31 사이의 2자리 숫자, D와 상호 배타적)`
	`DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적)`
	
	지정되지 않은 각 요소에 대해 디폴트값 1이 지정됩니다. 날짜 형식의 예:
	
	`"D-M-YYYY"`
	`"MM.DD.YYYY"`
	`"YYYYDDD"`

dumpfile = *x*	*x*는 거부된 행이 쓰여진 예외 파일의 완전한 이름(서버 데이터베이스 파티션에 따른)입니다. 레코드당 최대 32KB의 데이터가 쓰여집니다. 다음은 덤프 파일을 지정하는 방법을 표시하는 예입니다.
	
	`db2 load from data of del`
	`modified by dumpfile = /u/user/filename`
	`insert into table_name`
	
	인스턴스 소유자가 파일을 작성하고 소유합니다. 디폴트 파일 권한을 겹쳐쓰려면, dumpfileaccessall 파일 유형 수정자를 사용하십시오.
	주:
	1. 파티션된 데이터베이스 환경에서, 경로는 로딩 데이터베이스 파티션에 로컬이어야 하므로 현재 실행 중인 로드 조작은 동일한 파일에 쓰지 않습니다.
	2. 파일의 컨텐츠는 비동기 버퍼 지정 모드로 디스크에 쓰여집니다. 실패하거나 인터럽트된 로드 조작의 경우, 디스크에 커미트된 레코드 수는 확실히 알 수 없으며 LOAD RESTART 이후에 일관성을 보장할 수 없습니다. 파일에서는 단일 패스로 시작하고 완료하는 로드 조작에 대해서만 완료될 것으로 가정합니다.
	3. 지정된 파일이 이미 존재하는 경우, 재작성되지 않지만 추가됩니다.

표 32. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
dumpfileaccessall	덤프 파일 작성 시 'OTHERS'에 대한 읽기 액세스를 권한 부여합니다. 이 파일 유형 수정자는 다음 경우에만 유효합니다. 1. dumpfile 파일 유형 수정자와 결합하여 사용됩니다. 2. 사용자는 로드 목표 테이블에 대한 SELECT 특권을 갖습니다. 3. UNIX 운영 체제에 있는 DB2 서버 데이터베이스 파티션에서 발행됩니다. 지정된 파일이 이미 존재하는 경우, 권한을 변경하지 못합니다.
fastparse	주의하여 사용해야 합니다. 사용자 제공 컬럼 값에 대한 구문 검사를 줄이고, 성능을 향상시킵니다. 테이블은 구조적으로 옳은 것으로 보장되지만(유틸리티는 충분한 데이터 검사를 수행하여 분할 위반 또는 트랩을 방지함), 데이터 일관성의 유효성은 확인되지 않습니다. 데이터가 일관되고 올바름을 확신하는 경우에만 이 옵션을 사용하십시오. 예를 들어, 사용자 제공 데이터가 유효하지 않은 시간소인 컬럼 값 :1>0-00-20-07.11.12.000000을 포함하는 경우, FASTPARSE가 지정되면 이 값이 테이블에 삽입되며 FASTPARSE가 지정되지 않으면 거부됩니다.
implieddecimal	내포된 소수점의 위치는 컬럼 정의로 판별되며, 값의 끝으로 가정되지 않습니다. 예를 들어, 12345 값은 12345.00이 *아닌* 123.45로 DECIMAL(8,2) 컬럼에 로드됩니다. 이 수정자는 packeddecimal 수정자와 함께 사용될 수 없습니다.
timeformat="*x*"	*x*는 소스 파일에서 시간의 형식입니다.[1] 유효한 시간 요소는 다음과 같습니다. H - 시간(12시간 시스템의 경우 0 - 12 사이의 1 또는 2자리 숫자, 0 - 12 사이의 2자리 숫자, 24시간 시스템의 경우 0 - 24) HH - 시간(12시간 시스템의 경우 0 - 12 사이의 2자리 숫자, 24시간 시스템의 경우 0 - 24. H와 상호 배타적) M - 분(0 - 59 사이의 1 또는 2자리 숫자) MM - 분(0 - 59 사이의 2자리 숫자, M과 상호 배타적) S - 초(0 - 59 사이의 1 또는 2자리 숫자) SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적) SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적) TT - 오전/오후 지시(AM 또는 PM) 지정되지 않은 각 요소에 대해 디폴트값 0이 지정됩니다. 시간 형식의 예: "HH:MM:SS" "HH.MM TT" "SSSSS"

표 32. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
timestampformat="*x*"	*x*는 소스 파일에서 시간소인의 형식입니다.[1] 유효한 시간소인 요소는 다음과 같습니다. YYYY - 연도(0000 - 9999 사이의 4자리 숫자) M - 월(1 - 12 사이의 1 또는 2자리 숫자) MM - 월(01 - 12 사이의 2자리 숫자, M 및 MMM은 상호 배타적) MMM - 월(월 이름으로 사용되는 3자리 문자로 대소문자를 구분하지 않으며, M 및 MM은 상호 배타적) D - 일(1 - 31 사이의 1 또는 2자리 숫자) DD - 일(1 - 31 사이의 2자리 숫자, D와는 상호 배타적) DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적) H - 시간(12시간 시스템의 경우 0 - 12 사이의 1 또는 2자리 숫자, 24시간 시스템의 경우 0 - 24) HH - 시간(12시간 시스템의 경우 0 - 12 사이의 2자리 숫자, 24시간 시스템의 경우 0 - 24, H와는 상호 배타적) M - 분(0 - 59 사이의 1 또는 2자리 숫자) MM - 분(0 - 59 사이의 2자리 숫자, M, 분과 상호 배타적) S - 초(0 - 59 사이의 1 또는 2자리 숫자) SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적) SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적) U(1 - 12회) - 초의 소수 부분(U 어커런스 수는 각 자리가 0 - 9 사이의 자리 수를 나타냄) TT - 오전/오후 지시(AM 또는 PM)

표 32. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명

timestampformat="*x*"(계속)	디폴트값 1이 미지정된 YYYY, M, MM, D, DD 또는 DDD 요소에 지정됩니다. 디폴트값 'Jan'이 미지정된 MMM 요소에 지정됩니다. 미지정된 다른 모든 요소에 디폴트값 0이 지정됩니다. 다음은 시간소인 형식의 예입니다.
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	MMM 요소의 올바른 값은 다음을 포함합니다. 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' 및 'dec'. 이들 값은 대소문자가 구분되지 않습니다.
	TIMESTAMPFORMAT 수정자가 지정되지 않으면, 로드 유틸리티는 두 개의 가능한 형식 중 하나를 사용하여 시간소인 필드를 형식화합니다.
	`YYYY-MM-DD-HH.MM.SS`
	`YYYY-MM-DD HH:MM:SS`
	로드 유틸리티는 DD와 HH 간의 구분자를 확인하여 형식을 선택합니다. 대시 '-'인 경우, 로드 유틸리티는 일반 대시와 점 형식(YYYY-MM-DD-HH.MM.SS)을 사용합니다. 공백인 경우, 로드 유틸리티는 HH, MM 및 SS를 구분하기 위해 콜론 ':'을 예상합니다.
	어느 쪽 형식이든, 마이크로초 필드(UUUUUU)를 포함하는 경우, 로드 유틸리티는 점 '.'을 구분자로 예상합니다. YYYY-MM-DD-HH.MM.SS.UUUUUU 또는 YYYY-MM-DD HH:MM:SS.UUUUUU를 승인할 수 있습니다.
	다음 예는 사용자 정의 날짜 및 시간 형식을 포함하는 데이터를 schedule이라는 테이블로 로드하는 방법을 설명합니다.
	`db2 load from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

usegraphiccodepage	usegraphiccodepage가 제공되면, 그래픽 또는 2바이트 문자 대형 오브젝트(DBCLOB) 데이터 필드로 로드되는 데이터는 그래픽 코드 페이지에 있는 것으로 가정합니다. 나머지 데이터는 문자 코드 페이지에 있다고 가정합니다. 그래픽 코드 페이지는 문자 코드 페이지와 연관됩니다. LOAD는 codepage 수정자가 지정된 경우 이를 통해 문자 코드 페이지를 판별하고 codepage 수정자가 지정되지 않은 경우 데이터베이스의 코드 페이지를 통해 문자 코드 페이지를 판별합니다. 복구 중인 테이블에 그래픽 데이터가 있는 경우에만 삭제(drop) 테이블 복구로 생성되는 구분된 데이터 파일과 결합하여 이 수정자를 사용해야 합니다. **제한사항** 이들 파일이 오직 하나의 코드 페이지에 인코드된 데이터를 포함하면, usegraphiccodepage 수정자는 EXPORT 유틸리티로 작성된 DEL 파일과 함께 지정되지 않아야 합니다. usegraphiccodepage 수정자는 파일의 2바이트 문자 대형 오브젝트(DBCLOB)에서 무시됩니다.

표 32. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
xmlchar	XML 문서가 문자 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 지정된 문자 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 문자 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 문자 코드 페이지는 `codepage` 파일 유형 수정자에서 지정한 값이거나, 지정되지 않은 경우 응용프로그램 코드 페이지입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다.
xmlgraphic	XML 문서가 지정된 그래픽 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 특정 그래픽 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 그래픽 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 그래픽 코드 페이지는 `codepage` 파일 유형 수정자에서 지정한 값의 그래픽 구성요소이거나, 지정되지 않은 경우 응용프로그램 코드 페이지의 그래픽 구성요소입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다.

표 33. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(구분되지 않은 ASCII)

수정자	설명
binarynumerics	숫자(10진수는 아님) 데이터는 문자 표시가 아닌 2진 양식이어야 합니다. 손실이 큰 변환은 피합니다. 이 옵션은 `reclen` 옵션으로 지정된 고정 길이 레코드를 사용하며 위치상 ASC에서만 지원됩니다. 다음 규칙이 적용됩니다. • 데이터 유형 간의 변환은 수행되지 않으며, BIGINT, INTEGER 및 SMALLINT 예외가 있습니다. • 데이터 길이는 목표 컬럼 정의와 일치해야 합니다. • FLOAT는 IEEE 부동 소수점 형식이어야 합니다. • 로드 소스 파일의 2진 데이터는 로드 조작이 실행되는 플랫폼에 관계 없이 빅 엔디안(big-endian)으로 가정됩니다. 이 수정자에 의해 영향을 받는 컬럼의 데이터에 NULL 값을 표시할 수 없습니다. 이 수정자가 사용될 때 공백(보통 NULL로 해석)은 2진 값으로 해석됩니다.
nochecklengths	`nochecklengths`가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 로드하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 로드될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.

표 33. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(구분되지 않은 ASCII) (계속)

수정자	설명
nullindchar=*x*	*x*는 단일 문자입니다. 널(NULL) 값을 나타내는 문자를 *x*로 변경합니다. *x*의 디폴트값은 Y입니다. [2] 문자가 영문자인 경우를 제외하고 EBCDIC 데이터 파일의 경우 이 수정자의 대소문자를 구분합니다. 예를 들어, 널(NULL) 표시기 문자가 N 문자가 되도록 지정되는 경우, n은 널(NULL) 표시기로 인식됩니다.

packeddecimal	binarynumerics 수정자가 DECIMAL 필드 유형을 포함하지 않으므로 압축 10진수 데이터를 직접 로드합니다.
	
	이 옵션은 reclen 옵션으로 지정된 고정 길이 레코드를 사용하며 위치상 ASC에서만 지원됩니다.
	
	지원되는 부호 니블의 값은 다음과 같습니다.
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	이 수정자에 의해 영향을 받는 컬럼의 데이터에 NULL 값을 표시할 수 없습니다. 이 수정자가 사용될 때 공백(보통 NULL로 해석)은 2진 값으로 해석됩니다.
	
	서버 플랫폼에 관계 없이, 로드 소스 파일의 2진 데이터의 바이트 순서는 빅 엔디안(big-endian)으로 가정되므로, Windows 운영 체제에서 이 수정자를 사용하면 바이트 순서는 바뀌지 않아야 합니다.
	
	이 수정자는 implieddecimal 수정자와 함께 사용될 수 없습니다.

reclen=*x*	*x*는 최대값이 32 767인 정수입니다. 각 행에 대해 *x* 문자가 읽혀지며 행의 끝을 표시하기 위한 줄 바꾸기 문자는 사용되지 않습니다.
striptblanks	데이터를 변수 길이 필드로 로드할 때 뒤 공백을 절단합니다. 이 옵션이 지정되지 않으면, 공백이 보존됩니다. 이 옵션은 striptnulls와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다. 이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 t 옵션을 교체합니다.
striptnulls	데이터를 변수 길이 필드로 로드할 때 뒤 NULL 값(0x00 문자)을 절단합니다. 이 옵션이 지정되지 않으면, NULL 값이 보존됩니다. 이 옵션은 striptblanks와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다. 이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 padwithzero 옵션을 교체합니다.

zoneddecimal	BINARYNUMERICS 수정자는 DECIMAL 필드 유형을 포함하지 않으므로 존 10진수(zoned DECIMAL) 데이터를 로드합니다. 이 옵션은 RECLEN 옵션으로 지정된 고정 길이 레코드를 사용하며 위치상 ASC에서만 지원됩니다.
	
	반 바이트 기호 값은 다음 중 하나일 수 있습니다.
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	지원되는 숫자 값은 0x0 - 0x9입니다.
	
	지원되는 존 값은 0x3 및 0xF입니다.

표 34. 로드 유틸리티의 유효한 파일 유형 수정자: DEL 파일 형식(컬럼 식별자가 있는 ASCII)

수정자	설명

chardel*x*	*x*는 단일 문자열 분리문자입니다. 디폴트값은 큰따옴표(")입니다. 큰따옴표 대신 지정된 문자를 사용하여 문자열을 묶습니다. [23] 명시적으로 큰따옴표(")를 문자열 분리문자로 지정하려는 경우 다음과 같이 지정해야 합니다.
	`modified by chardel""`
	다음과 같이 작은따옴표(')를 문자열 분리문자로 지정할 수도 있습니다.
	`modified by chardel''`

coldel*x*	*x*는 단일 문자 컬럼 분리문자입니다. 디폴트값은 쉼표(,)입니다. 지정된 문자가 쉼표 대신 사용되어 컬럼 끝을 나타냅니다.[23]
decplusblank	플러스 부호 문자. 플러스 부호(+) 대신 공백을 양의 10진수 값 앞에 붙입니다. 디폴트 조치는 플러스 부호를 양의 10진수 값 앞에 붙이는 것입니다.
decpt*x*	*x*는 소수점 문자로 마침표를 대신하는 단일 문자입니다. 디폴트값은 마침표(.)입니다. 마침표 대신 지정된 문자가 소수점 문자로 사용됩니다.[23]

delprioritychar	분리문자의 현재 디폴트 우선순위: 레코드 구분 문자, 문자 분리문자, 컬럼 분리문자. 이 수정자는 분리문자 우선순위를 문자 분리문자, 레코드 구분 문자, 컬럼 분리문자로 되돌림으로써 이전 우선순위에 따른 기존 응용프로그램을 보호합니다. 구문:
	`db2 load ... modified by delprioritychar ...`
	예를 들면, 다음과 같은 DEL 데이터 파일이 있습니다.
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	`delprioritychar` 수정자를 지정하고, 이 데이터 파일에는 두 개의 행만이 있습니다. 두 번째 <행 분리문자>는 두 번째 행의 첫 번째 데이터 컬럼의 파트로 해석되지만, 첫 번째 및 세 번째 <행 분리문자>는 실제 레코드 구분 문자로 해석됩니다. 이 수정자가 지정되지 않은 경우, 이 데이터 파일에는 세 개의 행이 있으며, 각 행은 <행 분리문자>로 구분됩니다.

keepblanks	유형 CHAR, VARCHAR, LONG VARCHAR 또는 CLOB의 각 필드에 앞뒤 공백을 둡니다. 이 옵션이 없으면, 문자 분리문자 내에 없는 앞 공백과 뒤 공백은 모두 제거되며 공백 필드의 테이블로 NULL이 삽입됩니다.
	다음 예는 데이터 파일에서 앞과 뒤의 모든 공백을 보존하면서 TABLE1이라는 테이블로 데이터를 로드하는 방법을 설명합니다.
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nochardel	로드 유틸리티는 컬럼 분리문자 간의 모든 바이트를 컬럼 데이터의 파트로 가정합니다. 문자 분리문자는 컬럼 데이터의 파트로 구문 분석됩니다. DB2를 사용하여 데이터를 익스포트한 경우(익스포트 시 nochardel을 지정한 경우를 제외하고) 이 옵션을 지정하지 말아야 합니다. 이 옵션은 문자 분리문자가 없는 벤더 데이터 파일을 지원하기 위해 제공됩니다. 부적절한 사용은 데이터 유실이나 손상을 초래할 수 있습니다. 이 옵션은 `chardelx`, `delprioritychar` 또는 `nodoubledel`과 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다.
nodoubledel	2바이트 분리문자를 인식하지 않습니다.

표 35. 로드 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식

수정자	설명
forcein	코드 페이지 불일치와 관계없이 데이터를 승인하고 코드 페이지 간에 변환하지 않도록 유틸리티에 지시합니다. 데이터의 고정 길이 대상 필드가 충분히 큰지 검증하도록 고정 길이 대상 필드를 점검합니다. nochecklengths가 지정된 경우, 검사가 수행되지 않으며 각 행을 로드하려고 시도합니다.
nochecklengths	nochecklengths가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 로드하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 로드될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.

주:

1. 날짜 출력 문자열을 둘러싼 큰따옴표는 필수입니다. 필드 구분자는 a - z, A - Z 및 0 - 9를 포함할 수 없습니다. 필드 구분자는 DEL 파일 형식의 필드 분리문자나 문자 분리문자와 같지 않아야 합니다. 요소의 시작 및 종료 위치가 명확한 경우 필드 구분자가 선택적입니다. D, H, M 또는 S와 같은 요소가 사용되는 경우(수정자에 따라) 항목의 변수 길이 때문에 모호함이 있을 수 있습니다.

   시간소인 형식의 경우, 문자 M을 사용하는 month 및 minute 디스크립터 간의 모호함을 피하려면 주의해야 합니다. month 필드는 다른 날짜 필드와 인접해야 합니다. minute 필드는 다른 시간 필드와 인접해야 합니다. 다음은 모호한 시간소인 형식입니다.

   ```
   "M"(month 또는 minute일 수 있음)
   "M:M"(month 및 minute 구분 가능?)
   "M:YYYY:M"(둘 다 month로 해석됨.)
   "S:M:YYYY"(둘 다 시간 값 및 날짜 값에 인접)
   ```

   모호한 경우, 유틸리티는 오류 메시지를 발행하며, 조작에 실패합니다.

   다음은 명확한 시간소인 형식입니다.

   ```
   "M:YYYY" (Month)
   "S:M" (Minute)
   "M:YYYY:S:M" (Month....Minute)
   "M:H:YYYY:M:D" (Minute....Month)
   ```

   큰따옴표 및 백슬래시와 같은 일부 문자는 Escape 문자(예: ₩)가 앞에 와야 합니다.

2. chardel, coldel 또는 decpt 파일 유형에 제공되는 문자 값은 소스 데이터의 코드 페이지에 지정되어야 합니다.

문자 코드 포인트(문자 기호 대신)는 구문 xJJ 또는 0xJJ를 사용하여 지정될 수 있으며, 여기서 JJ는 코드 포인트의 16진수를 나타냅니다. 예를 들어, 컬럼 분리문자로 # 문자를 지정하려면, 다음 중 하나를 사용하십시오.

```
... modified by coldel# ...
... modified by coldel0x23 ...
... modified by coldelX23 ...
```

3. *데이터 이동을 위한 분리문자 고려사항*은 분리문자를 겹쳐쓰기할 때 사용할 수 있는 문자에 적용하는 제한사항을 나열합니다.
4. MODIFIED BY 옵션과 함께 지원되지 않는 파일 유형을 사용하려고 시도하는 경우 로드 유틸리티는 경고를 발행하지 않습니다. 이런 경우, 로드 조작에 실패하며 오류 코드가 리턴됩니다.
5. 내재적으로 숨겨진 행 변경 시간소인 컬럼을 포함하는 테이블로 임포트할 때, 내재적으로 숨겨진 컬럼의 등록 정보는 무시됩니다. 그러므로, 컬럼의 데이터가 임포트되는 데이터에 표시되지 않으며 명시적 컬럼 목록이 표시되지 않으면 rowchangetimestampmissing 파일 유형 수정자가 임포트 명령에 지정*되어야* 합니다.

표 36. codepage 및 usegraphiccodepage 사용 시 LOAD 동작

codepage=N	**usegraphiccodepage**	**LOAD 동작**
Absent	Absent	CLIENT 옵션이 지정된 경우에도 파일의 모든 데이터는 응용프로그램 코드 페이지가 아닌 데이터베이스 코드 페이지에 있는 것으로 가정됩니다.
Present	Absent	파일의 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 로드될 때 그래픽 데이터가 손상됩니다.
Absent	Present	CLIENT 옵션이 지정된 경우에도 파일의 문자 데이터는 데이터베이스 코드 페이지에 있는 것으로 가정됩니다. CLIENT 옵션이 지정된 경우에도 그래픽 데이터는 데이터베이스 그래픽 데이터의 코드 페이지에 있는 것으로 가정됩니다. 데이터베이스 코드 페이지가 1바이트이면, 모든 데이터는 데이터베이스 코드 페이지에 있는 것으로 가정됩니다. **경고:** 1바이트 데이터베이스로 로드될 때 그래픽 데이터가 손상됩니다.
Present	Present	문자 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. 그래픽 데이터는 N의 그래픽 코드 페이지에 있는 것으로 가정됩니다. N이 1바이트 또는 2바이트 코드 페이지인 경우, 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 로드될 때 그래픽 데이터가 손상됩니다.

ADMIN_CMD 프로시저를 사용하는 LOAD 명령

DB2 테이블에 데이터를 로드합니다. 서버에 있는 데이터는 파일, 테이프 또는 Named Pipe 양식이 될 수 있습니다. 테이블의 COMPRESS 속성이 YES로 설정된 경우, 테이블의 XML 스토리지 오브젝트의 데이터를 포함하여 이미 사전이 테이블에 존재하는 모든 데이터 및 데이터베이스 파티션에서 로드된 데이터가 압축됩니다.

304 페이지의 『로드 유틸리티의 파일 유형 수정자』로의 빠른 링크를 참조하십시오.

제한사항

로드 유틸리티는 계층 구조 레벨에서 데이터 로드를 지원하지 않습니다. 로드 유틸리티는 범위로 클러스터된 테이블과 호환 가능하지 않습니다.

범위

이 명령은 단일 요청으로 다중 데이터베이스 파티션에 대해 실행할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- *dataaccess*
- 데이터베이스의 LOAD 권한 및
  - 로드 유틸리티가 INSERT 모드, TERMINATE 모드 (이전 로드 삽입 조작을 종료하기 위해) 또는 RESTART 모드(이전 로드 삽입 조작을 재시작하기 위해)에서 호출될 때 테이블에 대한 INSERT 특권
  - 로드 유틸리티가 REPLACE 모드, TERMINATE 모드(이전 로드 삽입 조작을 종료하기 위해) 또는 RESTART 모드(이전 로드 삽입 조작을 재시작하기 위해)에서 호출될 때 테이블에 대한 INSERT 및 DELETE 특권
  - 예외 테이블에 대한 INSERT 특권(이러한 테이블이 로드 조작 중에 사용되는 경우).
- 보호 컬럼이 있는 테이블로 데이터를 로드하려면 테이블의 모든 보호 컬럼에 대한 쓰기 액세스를 허용하는 LBAC 증명서가 세션 권한 부여 ID에 있어야 합니다. 그렇지 않으면 로드에 실패하고 오류(SQLSTATE 5U014)가 리턴됩니다.
- 보호 설정된 행이 있는 테이블에 데이터를 로드하려면 세션 권한 부여 ID는 다음 기준에 부합되는 보안 레이블을 보유해야 합니다.
  - 테이블을 보호하는 보안 규정에 포함됨
  - 세션 권한 부여 ID에 쓰기 액세스 또는 모든 액세스 권한이 부여되었습니다.

  세션 권한 부여 ID에 이러한 보안 레이블이 없으면 로드에 실패하고 오류(SQLSTATE 5U014)가 리턴됩니다. 이 보안 레벨은 세션 권한 부여 ID'의 LBAC 증명서가 데이터의 해당 행을 보호하는 보안 레벨에 쓰기를 허용하지 않는 경우 로드된 행을 보호

하는 데 사용됩니다. 그러나 테이블을 보호하는 보안 규정이 CREATE SECURITY POLICY 명령문의 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 옵션을 사용하여 작성되는 경우에는 발생하지 않습니다. 이러한 경우, 로드에 실패하고 오류(SQLSTATE 42519)가 리턴됩니다.

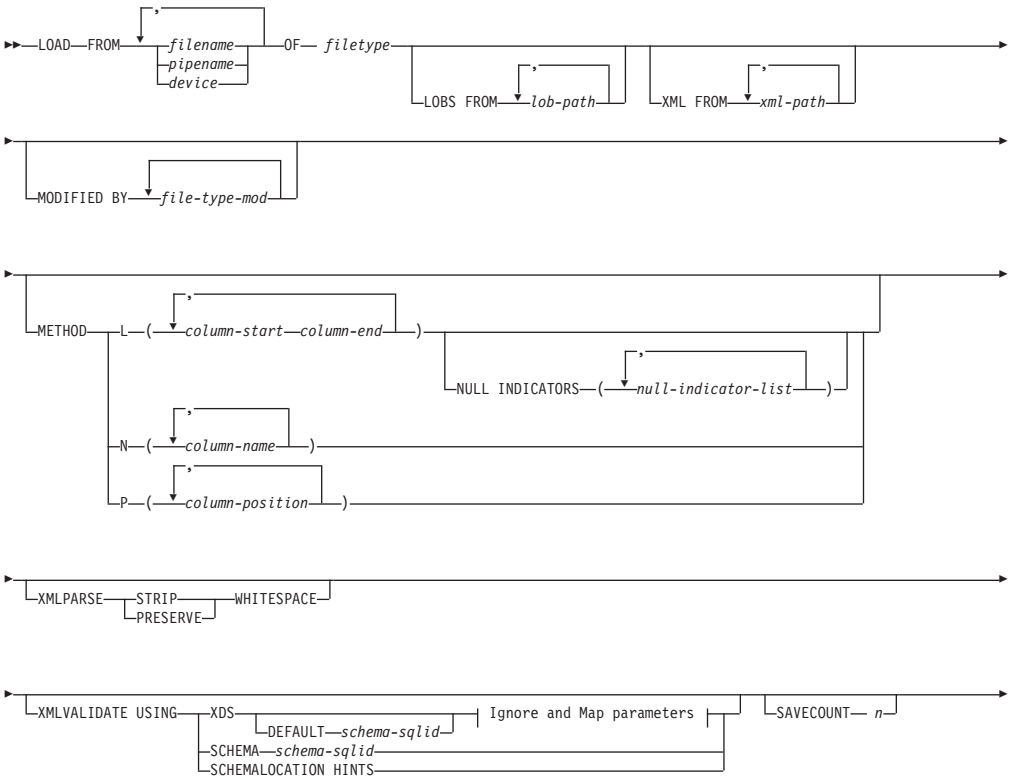
- REPLACE 옵션이 지정되어 있으면 세션 권한 부여 ID에 테이블을 삭제할 수 있는 권한이 있어야 합니다.
- LOCK WITH FORCE 옵션이 지정된 경우, SYSADM 권한이 필요합니다.

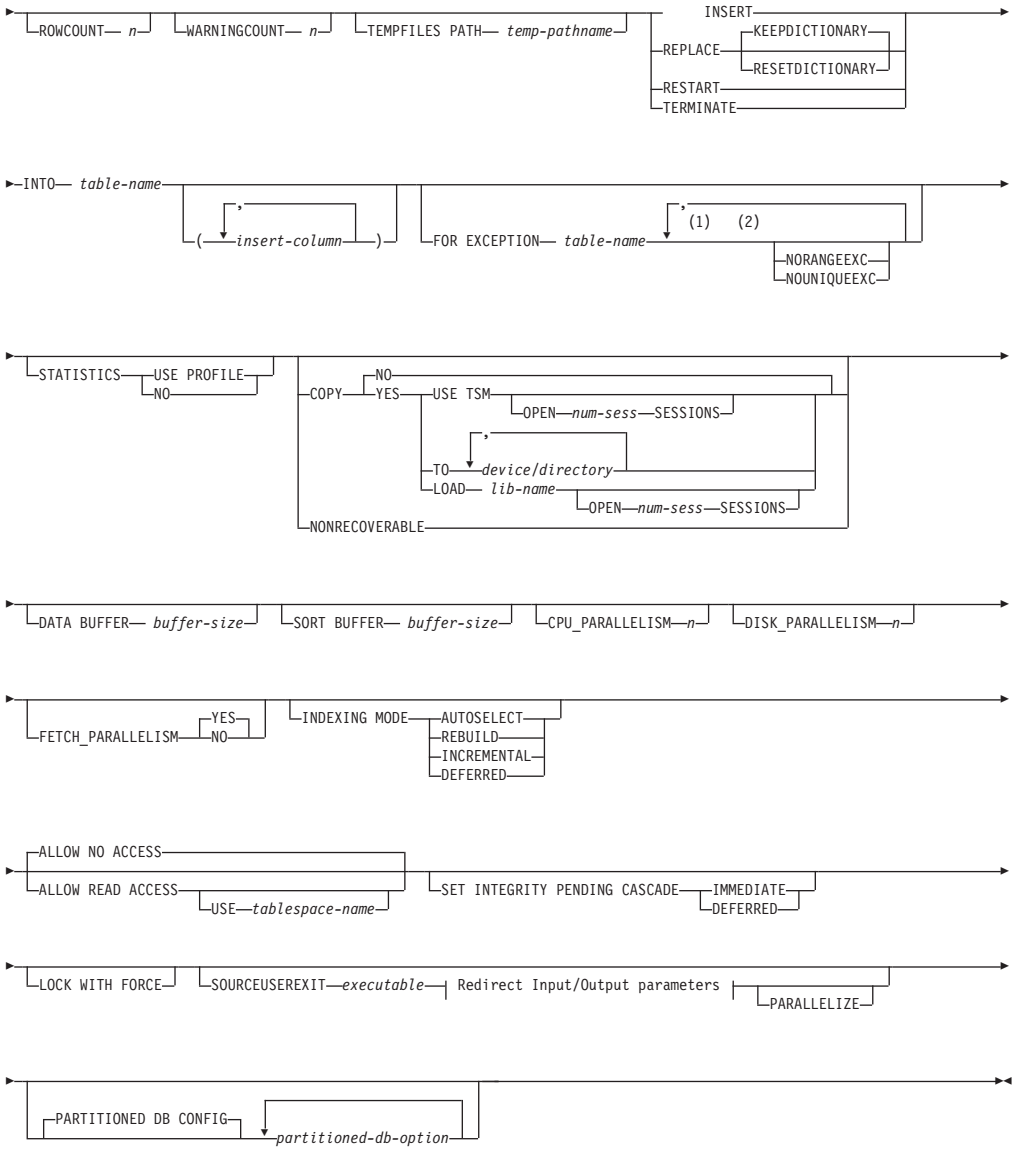
모든 로드 프로세스(및 일반적으로 모든 DB2 서버 프로세스)는 인스턴스 소유자가 소유하고 이들 모든 프로세스는 필요한 파일에 액세스하기 위해 인스턴스 소유자의 식별을 사용하므로 인스턴스 소유자는 입력 데이터 파일에 대한 읽기 액세스가 있어야 합니다. 이러한 입력 데이터 파일은 명령 호출자에 상관없이 인스턴스 소유자가 읽을 수 있어야 합니다.

필수 연결

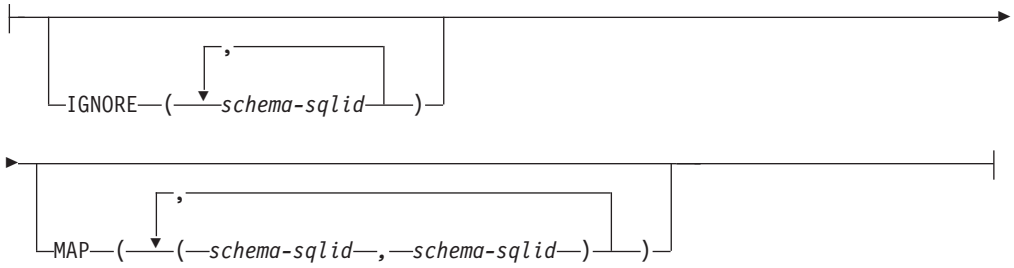
인스턴스. 명시적 접속은 필요하지 않습니다. 데이터베이스에 대한 연결이 설정되면 로컬 인스턴스에 대한 내재적 접속이 시도됩니다.

명령 구문

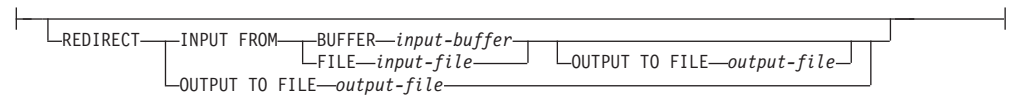




Ignore and Map parameters:



Redirect Input/Output parameters:



주:

1 이 키워드는 임의의 순서로 표시될 수 있습니다.

2 이들 각 키워드는 한 번만 표시될 수 있습니다.

명령 매개변수

FROM *filename* | *pipename* | *device*

주:

1. *ADMIN_CMD 프로시저를 사용하는 EXPORT 명령*을 사용하여 파일로 데이터를 익스포트하는 경우, 데이터 파일은 분리 사용자 ID 소유가 됩니다. 일반적으로 이 파일은 인스턴스 소유자가 액세스할 수 없습니다. CLP 또는 ADMIN_CMD 프로시저에서 LOAD를 실행시키려면 데이터 파일이 인스턴스 소유자 ID에 의해 액세스 가능해야 하며 따라서 데이터 파일에 대한 읽기 액세스 권한이 인스턴스 소유자에게 부여되어야 합니다.
2. 파일이 실제로는 분리되어 있지만 논리적으로는 한 개인 경우, 다중 IXF 파일로부터의 데이터 로드가 지원됩니다. 파일이 논리적 및 물리적으로 분리된 경우에는 지원되지 *않습니다*. (다중의 실제 파일도 EXPORT 명령을 한 번 호출하여 모두 작성된 경우 논리적으로는 한 개로 간주됩니다.)
3. 파티션된 데이터베이스 환경에서, XML 데이터를 파일에서 테이블로 로딩하는 경우에는 로드를 실행 중인 모든 데이터베이스 파티션에서 XML 데이터 파일에 읽기 액세스가 가능해야 합니다.

OF *filetype*

다음과 같은 데이터 형식을 지정합니다.

- ASC(컬럼 식별자가 없는 ASCII 형식)
- DEL(컬럼 식별자가 있는 ASCII 형식)
- IXF(Integration Exchange Format, PC 버전)는 DB2에 의해 독점 사용되는 2진 형식입니다.
- CURSOR(SELECT 또는 VALUES문에 대해 선언된 커서).

주: 분산 데이터베이스 환경에서, CURSOR 파일 유형을 사용하여 XML 데이터를 테이블로 로드하는 경우에는 PARTITION_ONLY 및 LOAD_ONLY 모드가 지원되지 않습니다.

LOBS FROM *lob-path*

로드할 LOB 값이 포함된 데이터 파일의 경로. 경로는 슬래시(/)로 끝나야 합니다. LOB 데이터 파일의 이름은 기본 데이터 파일(ASC, DEL 또는 IXF)에 저장되는데 LOB 컬럼으로 로드될 컬럼에 저장됩니다. 지정할 수 있는 최대 경로 수는 999입니다. 이는 내재적으로 LOBSINFILE 동작을 활성화합니다.

이 옵션은 CURSOR 파일 유형과 함께 지정되면 무시됩니다.

MODIFIED BY *file-type-mod*

파일 유형 수정자 옵션을 지정합니다. 304 페이지의 『로드 유틸리티의 파일 유형 수정자』를 참조하십시오.

METHOD

L 데이터를 로드할 시작 및 끝 컬럼을 지정합니다. 컬럼 번호는 데이터 행이 시작되는 바이트 오프셋입니다. 컬럼 번호는 1부터 시작됩니다. 이 메소드는 ASC 파일에서만 사용될 수 있으며 해당 파일 유형에 유일한 유효 메소드입니다.

NULL INDICATORS *null-indicator-list*

이 옵션은 METHOD L 매개변수가 지정된 경우(입력 파일이 ASC 파일)에만 사용할 수 있습니다. 널(NULL) 표시기 목록은 각 널(NULL) 표시기 필드의 컬럼 번호를 지정하는 쉼표로 구분된 양의 정수 목록입니다. 컬럼 번호는 데이터 행이 시작되는 널(NULL) 표시기 필드의 바이트 오프셋입니다. METHOD L 매개변수에 정의된 각 데이터 필드에 대해 한 개의 항목이 널(NULL) 표시기 목록에 있어야 합니다. 컬럼 번호 0은 해당 데이터 필드에 데이터가 항상 들어 있음을 나타냅니다.

널(NULL) 표시기의 Y 값은 컬럼 데이터를 널(NULL)로 지정합니다. 널(NULL) 표시기 컬럼에서 Y *이외의* 모든 문자는 컬럼 데이터가 널(NULL)이 아니며 METHOD L 옵션이 지정하는 컬럼 데이터가 로드됨을 지정합니다.

널(NULL) 표시기 문자는 MODIFIED BY 옵션을 사용하여 변경할 수 있습니다.

N 로드할 데이터 파일의 컬럼 이름을 지정합니다. 이 컬럼 이름의 대소문자는 시스템 카탈로그의 해당 이름의 대소문자와 일치해야 합니다. 널(NULL) 입력이 가능하지 않은 각 테이블 컬럼에는 METHOD N 목록에 해당 항목이 있어야 합니다. 예를 들면, 주어진 데이터 필드 F1, F2, F3, F4, F5 및 F6과 테이블 컬럼 C1 INT, C2 INT NOT NULL, C3 INT NOT NULL 및 C4 INT, method N (F2, F1, F4, F3)은

유효한 요청인 반면, `method N (F2, F1)`은 유효하지 않습니다. 이 메소드는 파일 유형이 IXF 또는 CURSOR인 경우에만 사용될 수 있습니다.

P 로드할 입력 데이터 필드의 필드 번호(1부터 번호를 지정)를 지정합니다. 널(NULL) 입력이 가능하지 않은 각 테이블 컬럼에는 METHOD P 목록에 해당 항목이 있어야 합니다. 예를 들면, 주어진 데이터 필드 F1, F2, F3, F4, F5 및 F6 그리고 테이블 컬럼 C1 INT, C2 INT NOT NULL, C3 INT NOT NULL 및 C4 INT, `method P (2, 1, 4, 3)`은 유효한 요청인 반면, `method P (2, 1)`은 유효하지 않습니다. 이 메소드는 파일 유형이 IXF, DEL 또는 CURSOR인 경우에만 사용할 수 있으며 DEL 파일 유형에 유일한 유효 메소드입니다.

XML FROM *xml-path*

XML 파일이 들어 있는 하나 이상의 경로를 지정합니다. XDS는 주 데이터 파일(ASC, DEL 또는 IXF)에 포함되며 XML 컬럼으로 로드될 컬럼에 있습니다.

XMLPARSE

XML 문서가 구문 분석되는 방법을 지정합니다. 이 옵션을 지정하지 않을 경우, XML 문서에 대한 구문 분석 동작은 CURRENT XMLPARSE OPTION 특수 레지스터의 값으로 판별됩니다.

STRIP WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하도록 지정합니다.

PRESERVE WHITESPACE

XML 문서가 구문 분석될 때 공백을 제거하지 않도록 지정합니다.

XMLVALIDATE

XML 문서가 스키마에 대해 유효성이 확인되도록 지정합니다(해당되는 경우).

USING XDS

기본 데이터 파일의 XDS(XML Data Specifier)에 의해 식별된 XML 스키마에 대해 XML 문서의 유효성을 확인합니다. USING XDS와 함께 XMLVALIDATE 옵션을 호출한 경우, 유효성 확인을 수행하는 데 사용 된 스키마는 디폴트로 XDS의 SCH 속성에 의해 판별됩니다. SCH 속성이 XDS에 존재하지 않을 경우, DEFAULT 절로 디폴트 스키마를 지정하지 않으면 스 키마 유효성 확인이 발생하지 않습니다.

DEFAULT, IGNORE 및 MAP 절은 스키마 판별 동작을 수정하는 데 사용될 수 있습니다. 이들 세 개의 선택적 절은 XDS의 권장 스펙에 직접적으로 적용되며 서로에게는 적용되지 않습니다. 예를 들어, 한 스키마가 DEFAULT 절에서 지정되어 선택되었으면 이 스키마는 IGNORE 절에 지정되어도 무시되지 않습니다. 마찬가지로 한 스키마

가 MAP 절에서 첫 번째 파트 쌍으로 지정되어 선택되면 다른 MAP 절 쌍의 두 번째 파트에 지정되어도 다시 맵핑되지 않습니다.

USING SCHEMA *schema-sqlid*
XML 문서가 지정된 SQL ID가 있는 XML 스키마에 대해 유효성이 확인됩니다. 이 경우 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

USING SCHEMALOCATION HINTS
XML 문서가 소스 XML 문서의 XML 스키마 위치 힌트에 의해 식별된 스키마에 대해 유효성이 확인됩니다. XML 문서에서 schemaLocation 속성을 찾을 수 없으면 유효성 확인이 발생하지 않습니다. USING SCHEMALOCATION HINTS 절을 지정하면 모든 XML 컬럼에 대해 XDS(XML Data Specifier)의 SCH 속성이 무시됩니다.

아래의 XMLVALIDATE 옵션 예를 참조하십시오.

IGNORE *schema-sqlid*
이 옵션은 USING XDS 매개변수가 지정된 경우에만 사용할 수 있습니다. SCH 속성에 의해 식별될 경우 IGNORE 절은 무시할 하나 이상의 스키마 목록을 지정합니다. 로드된 XML 문서에 대한 XDS(XML Data Specifier)에 SCH 속성이 존재하고 SCH 속성에 의해 식별된 스키마가 IGNORE할 스키마 목록에 포함된 경우, 이 로드된 XML 문서에 대해서는 스키마 유효성 확인이 발생하지 않습니다.

주:

IGNORE 절에 스키마가 지정되어 있으면 이 스키마는 또한 MAP 절에 있는 스키마 쌍의 왼쪽에 존재할 수 없습니다.

IGNORE 절은 XDS에만 적용됩니다. MAP 절에 의해 맵핑된 스키마는 IGNORE 절에 의해 지정된 경우 계속 무시되지 않습니다.

DEFAULT *schema-sqlid*
이 옵션은 USING XDS 매개변수가 지정된 경우에만 사용할 수 있습니다. DEFAULT 절을 통해 지정된 스키마는 로드된 XML 문서의 XDS(XML Data Specifier)가 XML 스키마를 식별하는 SCH 속성을 포함하지 않을 때 유효성 확인에 사용할 스키마를 식별합니다.

DEFAULT 절은 IGNORE 및 MAP 절보다 우선순위를 갖습니다. XDS가 DEFAULT 절을 충족시키면 IGNORE 및 MAP 스펙은 무시됩니다.

MAP *schema-sqlid*
이 옵션은 USING XDS 매개변수가 지정된 경우에만 사용할 수 있습니다. 로

드된 각 XML 문서에 대한 XDS(XML Data Specifier)의 SCH 속성이 지정하는 스키마 대신 사용할 대체 스키마를 지정하려면 이 MAP 절을 사용하십시오. MAP 절은 하나 이상의 스키마 쌍 목록을 지정하며, 여기서 각 쌍은 한 스키마 대 다른 스키마의 맵핑을 나타냅니다. 쌍에서 첫 번째 스키마는 XDS에 있는 SCH 속성에 의해 참조되는 스키마를 나타냅니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

MAP 절에 있는 스키마 쌍의 왼쪽에 있는 스키마는 IGNORE 절에 지정될 수 없습니다.

스키마 쌍 맵핑이 적용된 후, 최종 결과가 됩니다. 맵핑 조작은 전이되지 않으므로 선택된 스키마는 다른 스키마 상 맵핑에 계속 적용되지 않습니다.

스키마가 두 번 이상 맵핑될 수 없다는 것은 쌍의 왼쪽에 두 번 이상 나타날 수 없다는 것을 의미합니다.

SAVECOUNT *n*

로드 유틸리티가 *N*개의 행 뒤에 매번 일관성 지점을 설정하도록 지정합니다. 이 값은 쪽 수로 변환되며 Extent 크기의 간격으로 반올림됩니다. 메시지는 각각의 일관성 지점에서 발행되므로 LOAD QUERY를 사용하여 로드 조작을 모니터하는 경우 이 옵션을 선택해야 합니다. *N* 값이 충분히 높지 않으면 각 일관성 지점에서 수행된 활동의 동기화가 성능에 영향을 줍니다.

디폴트값은 0으로, 필요하지 않으면 일관성 지점을 설정하지 않음을 의미합니다.

CURSOR 파일 유형과 함께 지정되거나 XML 컬럼을 포함하는 테이블 로드 시 이 옵션은 무시됩니다.

ROWCOUNT *n*

로드될 파일에 있는 *N*개의 실제 레코드 수를 지정합니다. 사용자가 첫 번째 *N*개 행만 로드할 수 있도록 합니다.

WARNINGCOUNT *n*

*N*번의 경고 후에 로드 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 로드 파일이나 목표 테이블이 잘못 지정된 경우, 로드 유틸리티가 로드를 시도하는 각 행에 대해 경고를 생성하므로 로드에 실패하게 됩니다. *N*이 0이거나 이 옵션을 지정하지 않으면 발행된 경고 수에 관계없이 로드 조작을 계속합니다. 경고의 임계값에 도달하여 로드 조작이 중지된 경우, 다른 로드 조작이 RESTART 모드로 시작될 수 있습니다. 로드 조작은 마지막 일관성 지점에서 자동으로 계속됩니다. 또는 다른 로드 조작을 REPLACE 모드로 초기화하여 입력 파일의 처음부터 시작하게 할 수 있습니다.

TEMPFILES PATH *temp-pathname*

로드 조작 중에 임시 파일을 작성할 때 사용되는 경로의 이름을 지정하며 서버 데이터베이스 파티션에 따라 완전한 이름이어야 합니다.

임시 파일은 파일 시스템 스페이스를 차지합니다. 때때로 이 스페이스는 대량으로 요구됩니다. 다음은 모든 임시 파일에 할당되어야 하는 파일 시스템 스페이스의 예상 크기입니다.

- 로드 유틸리티가 생성하는 각각의 메시지 당 136바이트
- 데이터 파일에 Long 필드나 LOB가 포함된 경우 15KB의 오버헤드. 이 수량은 INSERT 옵션이 지정되어 있고 테이블에 대량의 Long 필드나 LOB 데이터가 있는 경우 급격히 증가할 수 있습니다.

INSERT

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 기존 테이블 데이터를 변경하지 않고 로드된 데이터를 테이블에 추가합니다.

REPLACE

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 테이블에서 기존 데이터를 모두 삭제하고 로드된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다. 계층 구조 간에 데이터를 이동하는 경우에 이 옵션을 사용하면 개별 부속 테이블이 아닌 전체 계층 구조에 대한 데이터만 바꿀 수 있습니다.

KEEPDICTIONARY

기존 압축 사전은 LOAD REPLACE 조작에서 보존됩니다. 테이블 COMPRESS 속성이 YES로 제공되면, 새로 교체된 데이터는 로드의 호출 이전에 존재하는 사전을 사용하여 압축됩니다. 테이블에 이미 존재하는 사전이 없는 경우, 테이블 COMPRESS 속성이 YES이면 테이블로 교체 중인 데이터를 사용하여 새 사전이 빌드됩니다. 이 경우 압축 사전을 빌드하는 데 필요한 데이터의 양은 ADC의 규정에 따릅니다. 이 데이터는 압축되지 않은 채로 테이블에 채워집니다. 사전이 테이블에 삽입되면, 로드되는 남아 있는 데이터는 이 사전과 함께 압축됩니다. 이는 디폴트 매개변수입니다. 요약 정보는 아래 표를 참조하십시오.

표 37. LOAD REPLACE KEEPDICTIONARY

압축	테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전 존재 여부[1]	압축 사전	데이터 압축
YES	YES	YES	테이블 행 데이터 및 XML 사전을 보존합니다.	로드되는 데이터는 압축됩니다.
YES	YES	NO	테이블 행 데이터 사전을 보존하고 새 XML 사전을 빌드합니다.	로드되는 테이블 행 데이터는 압축됩니다. XML 사전이 빌드된 후에, 로드되는 남아 있는 XML 데이터는 압축됩니다.

표 37. LOAD REPLACE KEEPDICTIONARY (계속)

압축	테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전 존재 여부[1]	압축 사전	데이터 압축
YES	NO	YES	테이블 행 데이터 사전을 빌드하고 새 XML 사전을 보존합니다.	테이블 행 데이터 사전이 빌드된 후에, 로드되는 남아 있는 테이블 행 데이터는 압축됩니다. 로드되는 XML 데이터는 압축됩니다.
YES	NO	NO	새 테이블 행 데이터 및 XML 사전을 빌드합니다.	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
NO	YES	YES	테이블 행 데이터 및 XML 사전을 보존합니다.	로드되는 데이터는 압축되지 않습니다.
NO	YES	NO	테이블 행 데이터 사전을 보존합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	YES	테이블 행 사전에 영향을 미치지 않습니다. XML 사전을 보존합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	NO	영향을 미치지 않습니다.	로드되는 데이터는 압축되지 않습니다.

주:

1. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.

RESETDICTIONARY

이 지시문은 LOAD REPLACE 처리 시 테이블 COMPRESS 속성이 YES인 경우 테이블 데이터 오브젝트의 새 사전을 빌드하도록 합니다. COMPRESS 속성이 NO이고 사전이 이미 테이블에 있으면 제거되며 새 사전은 테이블로 삽입되지 않습니다. 압축 사전은 오직 하나의 사용자 레코드로 빌드될 수 있습니다. 로드된 데이터 세트 크기가 0이고 기존에 존재하는 사전이 있는 경우, 이 사전은 보존되지 않습니다. 이 지시문으로 사전을 빌드하도록 요구되는 데이터의 양은 ADC의 규정에 따르지 않습니다. 요약 정보는 아래 테이블 2를 참조하십시오.

표 38. LOAD REPLACE RESETDICTIONARY

압축	테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전 존재 여부[1]	압축 사전	데이터 압축
YES	YES	YES	새 사전을 빌드합니다[2]. DATA CAPTURE CHANGES 옵션이 CREATE TABLE 또는 ALTER TABLE문에서 사용 가능하면, 현재 테이블 행 데이터 사전이 보존됩니다(*실행기록 압축 사전*이라고도 함).	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
YES	YES	NO	새 사전을 빌드합니다[2]. DATA CAPTURE CHANGES 옵션이 CREATE TABLE 또는 ALTER TABLE문에서 사용 가능하면, 현재 테이블 행 데이터 사전이 보존됩니다(*실행기록 압축 사전*이라고도 함).	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
YES	NO	YES	새 사전을 빌드합니다.	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
YES	NO	NO	새 사전을 빌드합니다.	사전이 빌드된 후에, 로드되는 남아 있는 데이터는 압축됩니다.
NO	YES	YES	사전을 제거합니다.	로드되는 데이터는 압축되지 않습니다.
NO	YES	NO	테이블 행 데이터 사전을 제거합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	YES	XML 스토리지 오브젝트 사전을 제거합니다.	로드되는 데이터는 압축되지 않습니다.
NO	NO	NO	영향을 미치지 않습니다.	모든 테이블 데이터는 압축되지 않습니다.

주:

1. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.
2. 사전이 존재하고 압축 속성이 사용 가능하지만 테이블 파티션에 로드할 레코드가 없는 경우, 새 사전은 빌드될 수 없으며 RESETDICTIONARY 조작은 기존의 사전을 보존하지 않습니다.

TERMINATE

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 일관성 지점이 패스된 경우라도 이전에 인터럽트된 로드 조작을 종료하고 조작이 시작된 특정 시점으로 롤백합니다. 조작에 관계된 테이블 스페이스의 상태가 정상으로 리턴되고 모

든 테이블 오브젝트는 일관성을 갖게 됩니다(인덱스 오브젝트가 유효하지 않은 것으로 표시되는 경우, 인덱스 재빌드가 다음 액세스에서 자동으로 발생). 종료되는 로드 조작이 LOAD REPLACE이면, 테이블은 LOAD TERMINATE 조작 이후에 빈 테이블로 절단됩니다. 종료되는 로드 조작이 LOAD INSERT이면, LOAD TERMINATE 조작 이후에 테이블은 원래의 모든 레코드를 보유합니다. 사전 관리 요약 정보는 아래 테이블 3을 참조하십시오.

LOAD TERMINATE 옵션은 테이블 스페이스에서 백업 보류 상태를 제거하지 않습니다.

RESTART

로드 유틸리티가 실행될 수 있는 네 가지 모드 중 하나. 이전에 인터럽트된 로드 조작을 재시작합니다. 로드 조작은 로드, 빌드 또는 삭제 단계의 마지막 일관성 지점에서 자동으로 계속됩니다. 사전 관리 요약 정보는 아래 테이블 4를 참조하십시오.

INTO *table-name*

데이터가 로드되어야 할 데이터베이스 테이블을 지정합니다. 이 테이블은 시스템 테이블, 선언된 임시 테이블 또는 작성된 임시 테이블일 수 없습니다. 별명 또는 완전하거나 규정에 맞지 않는 테이블 이름을 지정할 수 있습니다. 규정된 테이블 이름의 양식은 `schema.tablename`입니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

insert-column

데이터가 삽입되어야 할 테이블 컬럼을 지정합니다.

로드 유틸리티는 하나 이상의 스페이스를 포함하는 이름를 가진 컬럼의 구문을 분석할 수 없습니다. 예를 들면, 다음과 같습니다.

`Int 4` 컬럼 때문에 실패합니다. 솔루션은 이 컬럼 이름을 큰따옴표 안에 두는 것입니다.

FOR EXCEPTION *table-name*

오류 행이 복사될 예외 테이블을 지정합니다. 고유 인덱스 또는 기본 키를 위반하는 모든 행이 복사됩니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

예외 테이블에 작성된 정보는 덤프 파일에는 작성되지 *않습니다*. 파티션된 데이터베이스 환경에서는 예외 테이블이 로드 중인 테이블이 정의된 데이터베이스 파티션에 대해 정의되어야 합니다. 그렇지 않으면, 덤프 파일에는 유효하지 않거나 구문 오류 때문에 로드될 수 없는 행이 포함됩니다.

XML 데이터 로드 시에는 다음과 같은 경우 FOR EXCEPTION절을 사용하여 로드 예외 테이블을 지정할 수 없습니다.

- 레이블 기반 액세스 제어(LBAC)를 사용하는 경우.
- 데이터 파티션 테이블로 데이터를 로드하는 경우.

NORANGEEXC

행이 범위 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

NOUNIQUEEXC

행이 고유 제한조건 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

STATISTICS USE PROFILE

이 테이블에 대해 정의된 프로파일에 따라 로드 중 통계를 수집하도록 로드를 지시합니다. 이 프로파일은 로드가 실행되기 전에 작성되어야 합니다. 이 프로파일은 RUNSTATS 명령으로 작성됩니다. 프로파일이 존재하지 않고 프로파일에 따라 통계를 수집하도록 로드에 지시된 경우 경고가 리턴되며 통계가 수집되지 않습니다.

STATISTICS NO

통계가 수집되지 않고 카탈로그의 통계가 변경되지 않도록 지정합니다. 이는 디폴트값입니다.

COPY NO

포워드 복구가 사용 가능한 경우(즉, *logretain* 또는 *userexit*이 설정된 경우), 테이블이 있는 테이블 스페이스가 백업 보류 상태에 있도록 지정합니다. COPY NO 옵션을 사용하면 테이블 스페이스 상태를 로드 진행 중 테이블 스페이스 상태로 둡니다. 이 상태는 로드가 완료되거나 중단되면 소멸되는 일시적인 상태입니다. 테이블 스페이스의 모든 테이블의 데이터는 테이블 스페이스 백업 또는 전체 데이터베이스 백업이 완료되어야 갱신하거나 삭제할 수 있습니다. 그러나 SELECT문을 사용하여 테이블의 데이터에 액세스할 수 있습니다.

복구 가능한 데이터베이스에 대해 COPY NO 옵션이 지정된 LOAD는 테이블 스페이스를 백업 보류 상태로 둡니다. 예를 들어, COPY NO 및 INDEXING MODE DEFERRED 옵션과 함께 LOAD를 수행하면 인덱스를 새로 고쳐야 합니다. 테이블에 대한 특정 쿼리는 인덱스 스캔을 해야 하며 인덱스를 새로 고쳐야 성공할 수 있습니다. 백업 보류 상태에 있는 테이블 스페이스에 위치한 인덱스는 새로 고칠 수 없습니다. 이러한 경우, 백업을 수행해야 테이블에 대한 액세스가 허용됩니다. 인덱스 새로 고침은 쿼리로 인덱스를 액세스할 때 데이터베이스에 의해 자동으로 완료됩니다. COPY NO, COPY YES 또는 NONRECOVERABLE 중 하나가 지정되지 않고 데이터베이스가 복구 가능한 경우(**logretain** 또는 **logarchmeth1**이 사용 가능함), COPY NO가 디폴트입니다.

COPY YES

로드된 데이터의 사본이 저장되도록 지정합니다. 포워드 복구가 사용 불가능한 경우 이 옵션은 유효하지 않습니다.

USE TSM

TSM(Tivoli Storage Manager)을 사용하여 사본이 저장되도록 지정합니다.

OPEN *num-sess* **SESSIONS**

TSM 또는 벤더 제품과 함께 사용되는 입출력 세션 수. 디폴트값은 1입니다.

TO *device/directory*

이미지 복사가 작성될 디바이스 또는 디렉토리를 지정합니다.

LOAD *lib-name*

사용될 벤더 백업 및 리스토어 I/O 기능이 들어 있는 공유 라이브러리 이름(Windows 운영 체제의 DLL). 여기에는 전체 경로가 포함될 수 있습니다. 전체 경로가 지정되지 않으면 디폴트값은 User Exit 프로그램이 있는 경로입니다.

NONRECOVERABLE

로드 트랜잭션이 복구 불가능하도록 표시하여 후속적인 롤 포워드 조치로 복구할 수 없도록 지정합니다. 롤 포워드 유틸리티가 트랜잭션을 건너뛰고 데이터가 로드 중인 테이블을 "유효하지 않음"으로 표시합니다. 또한 이 유틸리티는 해당 테이블에 대한 후속 트랜잭션을 무시합니다. 롤 포워드 조작이 완료된 후 이러한 테이블은 복구 불가능한 로드 조작 완료 후 커미트 지점 다음에 수행되는 백업(전체 또는 테이블 스페이스)에서 삭제되거나 리스토어될 수 있습니다.

이 옵션을 사용하면 로드 조작 후에 테이블 스페이스가 백업 보류 상태가 되지 않으며 로드 조작 중에 로드된 데이터의 사본을 작성할 필요가 없습니다. COPY NO, COPY YES 또는 NONRECOVERABLE 중 하나가 지정되지 않고 데이터베이스가 복구 가능하지 않은 경우(**logretain** 또는 **logarchmeth1**이 사용 가능하지 않음), NONRECOVERABLE이 디폴트입니다.

WITHOUT PROMPTING

데이터 파일의 목록에 로드될 모든 파일이 포함되고 나열된 디바이스 또는 디렉토리가 전체 로드 조작에 충분하도록 지정합니다. 연속 입력 파일을 찾을 수 없거나 목표 복사가 로드 조작이 완료되기 전에 채워지면 로드 조작에 실패하고 테이블이 로드 보류 상태가 됩니다.

DATA BUFFER *buffer-size*

유틸리티에서 데이터 전송에 필요한 버퍼 스페이스로 사용할 4KB 페이지의 수를 지정합니다(병렬 처리 수준과 무관함). 지정된 값이 알고리즘의 최소보다 작으면, 최소 필요 자원이 사용되고 경고는 리턴되지 않습니다.

이 메모리는 유틸리티 힙에서 바로 할당되며 크기는 *util_heap_sz* 데이터베이스 구성 매개변수로 수정할 수 있습니다.

값이 지정되지 않으면 런타임시 유틸리티에 의해 적절한 디폴트값이 계산됩니다. 디폴트값은 테이블의 일부 등록 정보 뿐 아니라 로더의 인스턴스화 시간에 유틸리티 힙에서 사용 가능한 여유 공간의 백분율을 기초로 합니다.

SORT BUFFER *buffer-size*

이 옵션은 로드 조작 중 SORTHEAP 데이터베이스 구성 매개변수를 겹쳐쓰는 값을 지정합니다. 이 옵션은 인덱스와 함께 테이블을 로드하는 경우와 INDEXING MODE 매개변수가 DEFERRED로 지정되지 않은 경우에만 의미가 있습니다. 지정된 값은 SORTHEAP의 값을 초과할 수 없습니다. 이 매개변수는 SORTHEAP의 값을 변경하지 않고 많은 인덱스가 있는 테이블을 로드할 때 사용되는 정렬 메모리를 조절하는 데 유용하며 일반 쿼리 처리에도 영향을 줍니다.

CPU_PARALLELISM *n*

테이블 오브젝트를 빌드할 때 레코드를 구문 분석, 변환 및 포맷팅하기 위해 로드 유틸리티가 작성하는 프로세스 또는 스레드의 수를 지정합니다. 이 매개변수는 데이터베이스 파티션당 실행하는 프로세스 수를 이용하도록 설계되었습니다. 소스 데이터에 있는 기록 순서가 보존되므로 사전에 정렬된 데이터를 로드할 때 특히 유용합니다. 이 매개변수의 값이 0이거나 지정되지 않으면 로드 유틸리티는 적절한 디폴트값(일반적으로 사용 가능한 CPU 수에 기초함)을 사용합니다.

주:

1. 이 매개변수는 LOB 또는 LONG VARCHAR 필드 중 하나가 있는 테이블과 함께 사용되면 시스템 CPU 수나 사용자가 지정한 값과 관계없이 값은 1이 됩니다.
2. SAVECOUNT 매개변수에 값을 작게 지정하면 로더는 데이터 및 테이블 메타데이터를 둘 다 비우기 위해 훨씬 더 많은 입출력 조작을 수행하게 됩니다. CPU_PARALLELISM이 1보다 크면 플러시 조작은 비동기로 수행되며 로더가 CPU를 이용할 수 있도록 합니다. CPU_PARALLELISM이 1로 설정되면 로더는 일관성 지점 동안 입출력을 대기합니다. CPU_PARALLELISM이 2로 설정되고 SAVECOUNT가 10 000으로 설정된 로드 조작은 CPU가 하나만 있어도 CPU_PARALLELISM이 1로 설정된 동일한 조작보다 빠르게 완료됩니다.

DISK_PARALLELISM *n*

로드 유틸리티가 데이터를 테이블 스페이스 컨테이너에 작성하기 위해 작성하는 프로세스나 스레드의 수를 지정합니다. 값이 지정되지 않으면 유틸리티가 테이블 스페이스 컨테이너의 수 및 테이블의 등록 정보를 기반으로 하여 적절한 디폴트값을 선택합니다.

FETCH_PARALLELISM YES | NO

DATABASE 키워드로 선언된 커서에서 로드를 수행하거나 API sqlu_remotefetch_entry 미디어 항목을 사용하고 이 옵션이 YES로 설정되어 있는 경우, 로드 유틸리티는 가능한 리모트 데이터 소스로부터 페치(fetch)를 병렬 처리하려고 합니다. NO로 설정되면 병렬 페치(fetch)는 수행되지 않습니다. 디폴트값은 YES입니다. 자세한 정보는 *CURSOR 파일 유형을 사용하여 데이터 이동*을 참조하십시오.

INDEXING MODE

로드 유틸리티가 인덱스를 재빌드할 것인지 또는 인덱스를 점차적으로 확장할 것인지를 지정합니다. 가능한 값은 다음과 같습니다.

AUTOSELECT

로드 유틸리티는 자동으로 REBUILD나 INCREMENTAL 모드를 결정합니다. 이 결정은 로드 중인 데이터 양과 인덱스 트리의 용량을 기반으로 합니다. 인덱스 트리의 용량에 관한 정보는 인덱스 오브젝트에 저장되어 있습니다. 이 정보를 채우기 위해 RUNSTATS는 필요하지 않습니다. 디폴트 인덱스 모드는 AUTOSELECT입니다.

REBUILD

모든 인덱스가 재빌드됩니다. 이전 및 추가된 테이블 데이터에 대해 모든 인덱스 키 파트를 정렬하려면 유틸리티에 충분한 자원이 있어야 합니다.

INCREMENTAL

인덱스는 새로운 데이터와 함께 확장됩니다. 이 방법은 인덱스 여유 공간을 사용합니다. 삽입된 레코드에 대한 인덱스 키를 추가하려면 충분한 정렬 스페이스만 있으면 됩니다. 이 메소드는 인덱스 오브젝트가 유효하고 로드 조작이 시작될 때 액세스할 수 있는 경우에만 지원 됩니다(예를 들어, DEFERRED 모드가 지정된 로드 조작 직후에는 유효하지 않음). 이 모드가 지정되었지만 인덱스 상태 때문에 지원되지 않는 경우, 경고가 리턴되며 로드 조작은 계속 REBUILD 모드입니다. 마찬가지로 로드 빌드 단계에서 로드 재시작 조작이 시작된 경우, INCREMENTAL 모드는 지원되지 않습니다.

다음 조건에 모두 해당될 때 증분 인덱싱은 지원되지 않습니다.

- LOAD COPY 옵션이 지정되었습니다(USEREXIT 또는 LOGRETAIN 옵션을 사용하여 *logarchmeth1*).
- 테이블이 DMS 테이블 스페이스에 있습니다.
- 인덱스 오브젝트가 로드 중인 테이블에 속한 다른 테이블 오브젝트가 공유하는 테이블에 있습니다.

이 제한사항을 통과하려면 인덱스를 별도의 테이블 스페이스에 넣는 것이 바람직합니다.

DEFERRED

이 모드를 지정하면 로드 유틸리티가 인덱스 작성을 시도하지 않습니다. 인덱스는 새로 고칠 필요가 있는 것으로 표시됩니다. 로드 조작과 관련이 없는 이러한 인덱스에 처음 액세스할 때 강제로 재빌드될 수 있습니다. 그렇지 않으면 인덱스는 데이터베이스가 재시작될 때 재빌드될 수 있습니다. 이러한 방법을 수행하려면 가장 큰 인덱스의 모든 키 부분에 대해 충분한 정렬 스페이스가 있어야 합니다. 인덱스 구성에 연속적으로 사용되는 총 시간은 REBUILD 모드에 필요한 시간보다 깁니다. 따라서 지연된 인덱싱으로 다중 로드 조작을 수행할 때 로드하지 않는 액세스에서 먼저 인덱스를 재빌드하게 하는 것보다는 시퀀스에서 마지막 로드 조작에서 인덱스 재빌드를 수행하게 하는 것이 바람직합니다(성능 관점에서).

지연된 인덱싱은 비고유 인덱스가 있는 테이블에만 지원되므로 로드 단계 중 삽입된 중복 키는 로드 조작 후 지속되지 않습니다.

ALLOW NO ACCESS

로드가 로드 중 독점 액세스를 위해 목표 테이블을 잠급니다. 로드 중 테이블 상태는 로드 진행 중으로 설정됩니다. ALLOW NO ACCESS는 디폴트 동작입니다. LOAD REPLACE에는 이 옵션만 유효합니다.

테이블에 제한조건이 있을 때 테이블 상태는 로드 진행 중과 함께 무결성 설정 보류로 설정됩니다. 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY문을 사용해야 합니다.

ALLOW READ ACCESS

로드가 목표 테이블을 공유 모드로 잠급니다. 테이블 상태는 로드 진행 중 및 읽기 액세스 모두로 설정됩니다. 판독기는 테이블이 로드되는 동안 데이터의 비델타 부분에 액세스할 수 있습니다. 다시 말해, 테이블 판독기는 로드가 시작되기 전에 있던 데이터에 액세스할 수 있지만 로드되고 있는 데이터는 로드가 완료될 때까지 사용할 수 없습니다. ALLOW READ ACCESS 로드의 LOAD TERMINATE 또는 LOAD RESTART는 이 옵션을 사용할 수 있지만 ALLOW NO ACCESS 로드의 LOAD TERMINATE 또는 LOAD RESTART는 이 옵션을 사용할 수 없습니다. 게다가 목표 테이블의 인덱스가 재빌드가 필요한 것으로 표시된 경우 이 옵션은 유효하지 않습니다.

테이블에 제한조건이 있으면 테이블 상태는 로드 진행 중, 읽기 액세스뿐만 아니라 무결성 설정 보류로 설정됩니다. 로드가 종료되면 테이블 상태 중에서 로드 진행 중 상태는 제거되지만 무결성 설정 보류 및 읽기 액세스 상태는 남아 있습니다. 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY 문을 사용해야 합니다. 테이블의 상태가 무결성 설정 보류 및 읽기 액세스인 경

우, 판독기는 데이터의 비델타 부분에는 여전히 액세스할 수 있으나 SET INTEGRITY문이 완료될 때까지 데이터의 새 (델타) 부분에는 액세스할 수 없습니다. 사용자는 SET INTEGRITY문을 발행하지 않고 동일한 테이블에서 다중 로드를 수행할 수 있습니다. 그러나 SET INTEGRITY문이 발행될 때까지 원본(확인된) 데이터만 볼 수 있습니다.

ALLOW READ ACCESS는 다음 수정자도 지원합니다.

USE *tablespace-name*

인덱스가 재빌드되는 경우, 인덱스의 쉐도우 사본이 테이블 스페이스 *tablespace-name*에 빌드되고 INDEX COPY PHASE 중 로드 맨 끝에서 원본 테이블 스페이스로 복사됩니다. 이 옵션에는 시스템 임시 테이블 스페이스만 사용할 수 있습니다. 테이블 스페이스를 지정하지 않을 경우 음영 인덱스는 인덱스 오브젝트와 동일한 테이블 스페이스에 작성됩니다. 음영 사본이 인덱스 오브젝트와 동일한 테이블 스페이스에 작성된 경우, 이전 인덱스 오브젝트에 대한 음영 인덱스 오브젝트의 복사는 순간적입니다. 음영 사본이 인덱스 오브젝트와 다른 테이블 스페이스에 있을 경우 실제 복사가 수행됩니다. 상당한 I/O 및 시간이 포함될 수 있습니다. INDEX COPY PHASE 중 로드 맨 끝에서 테이블이 오프라인 상태에 있는 동안 복사가 발생합니다.

이 옵션을 사용하지 않으면 음영 인덱스는 원본과 동일한 테이블 스페이스에서 빌드됩니다. 원본 인덱스와 음영 인덱스가 둘 다 디폴트로 동시에 동일한 테이블 스페이스에 있으므로 한 테이블 스페이스에 이들 인덱스를 모두 보유하기 위한 스페이스가 충분하지 않을 수 있습니다. 이 옵션을 사용하면 인덱스에 충분한 테이블 스페이스를 보유할 수 있습니다.

사용자가 INDEXING MODE REBUILD 또는 INDEXING MODE AUTOSELECT를 지정하지 않을 경우 이 옵션은 무시됩니다. INDEXING MODE AUTOSELECT가 선택되고 로드가 인덱스를 점차적으로 갱신하도록 선택할 경우에도 이 옵션이 무시됩니다.

SET INTEGRITY PENDING CASCADE

LOAD로 인해 테이블이 무결성 설정 보류 상태에 놓일 경우, 사용자는 SET INTEGRITY PENDING CASCADE 옵션을 사용하여 로드된 테이블의 무결성 설정 보류 상태가 모든 하위(하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블 포함)에 즉시 연쇄되는지 여부를 지정할 수 있습니다.

IMMEDIATE

무결성 설정 보류 상태가 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블로 즉시 확장됨을 표시합니다.

LOAD INSERT 조작의 경우, IMMEDIATE 옵션을 지정해도 무결성 설정 보류 상태는 하위 외부 키 테이블로 확장되지 않습니다.

나중에 로드된 테이블에 제한조건 위반이 있는지 여부를 점검할 때(SET INTEGRITY문의 IMMEDIATE CHECKED 옵션을 사용하여) 무결성 설정 보류 읽기 액세스에 있던 하위 외부 키 테이블은 무결성 설정 보류 권한 없음 상태가 됩니다.

DEFERRED

로드된 테이블만이 무결성 설정 보류 상태에 놓임을 표시합니다. 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블의 상태는 변경되지 않습니다.

하위 외부 키 테이블에 제한조건 위반이 있는지 여부를 점검할 때(SET INTEGRITY문의 IMMEDIATE CHECKED 옵션을 사용하여) 하위 외부 키 테이블 은 나중에 내재적으로 무결성 설정 보류 상태에 놓일 수 있습니다. 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블은 해당 기본 테이블 중 하나에 무결성 위반이 있는지 여부를 점검할 때 내재적으로 무결성 설정 보류 상태가 됩니다. 하위 테이블이 무결성 설정 보류 상태가 되었음을 나타내는 경고(SQLSTATE 01586)가 발행됩니다. 이들 하위 테이블이 언제 무결성 설정 보류 상태가 되는지에 대해서는 SQL 참조서에서 SET INTEGRITY문의 참고 절을 참조하십시오.

SET INTEGRITY PENDING CASCADE 옵션을 지정하지 않은 경우,

- 로드된 테이블만이 무결성 설정 보류 상태에 놓입니다. 하위 외부 키 테이블, 하위 즉시 구체화된 쿼리 테이블 및 하위 즉시 스테이징 테이블의 상태는 계속 변경되지 않으며 나중에 로드된 테이블에 제한조건 위반이 있는지 점검할 때 내재적으로 무결성 설정 보류 상태가 될 수 있습니다.

LOAD 결과 목표 테이블이 무결성 설정 보류 상태가 되지 않을 경우 SET INTEGRITY PENDING CASCADE 옵션은 무시됩니다.

LOCK WITH FORCE

이 유틸리티는 로드 처리에서 테이블 잠금을 포함하는 다양한 잠금을 획득합니다. 대기라기 보다는 시간종료라고 할 수 있으며, 잠금을 획득할 때 로드는 이 옵션을 통해 목표 테이블에 충돌 잠금을 보유하고 있는 기타 응용프로그램을 강제로 해제할 수 있습니다. 시스템 카탈로그 테이블에서 충돌 잠금을 보유하고 있는 응용프로그램은 로드 유틸리티로 강제로 해제할 수 없습니다. 강제된 응용프로그램은 롤백되고 로드 유틸리티가 필요로 하는 잠금을 릴리스합니다. 그런 다음 로드 유틸리티는 계속 진행될 수 있습니다. 이 옵션을 사용하려면 FORCE APPLICATIONS 명령(SYSADM 또는 SYSCTRL)과 동일한 권한이 필요합니다.

ALLOW NO ACCESS 로드는 로드 조작이 시작될 때 충돌 잠금을 보유하고 있는 응용프로그램을 강제 실행할 수 있습니다. 로드가 시작될 때 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

ALLOW READ ACCESS 로드는 로드 조작이 시작되거나 종료될 때 충돌 잠금을 보유하고 있는 응용프로그램을 강제 실행할 수 있습니다. 로드가 시작될 때 유틸리티는 테이블 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다. 로드 조작이 종료될 때 로드 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

SOURCEUSEREXIT *executable*

유틸리티에 데이터를 입력하기 위해 호출될 실행 파일 이름을 지정합니다.

REDIRECT

INPUT FROM

BUFFER *input-buffer*

*input-buffer*에 지정된 바이트 스트림을 지정된 실행 파일을 실행하는 프로세스의 STDIN 파일 디스크립터로 패스합니다.

FILE *input-file*

이 클라이언트측 파일의 내용을 지정된 실행 파일을 실행하는 프로세스의 STDIN 파일 디스크립터로 패스합니다.

OUTPUT TO

FILE *output-file*

STDOUT 및 STDERR 파일 디스크립터를 지정된 서버측 파일로 캡처합니다.

PARALLELIZE

여러 User Exit 프로세스를 동시 호출하여 로드 유틸리티로 들어가는 데이터의 처리량을 증가시킵니다. 이 옵션은 다중 파티션 데이터베이스 환경에서만 적용할 수 있으므로 단일 파티션 데이터베이스 환경에서는 무시됩니다.

자세한 정보는 *사용자 정의된 응용프로그램을 사용하여 데이터 이동*을 참조하십시오.

PARTITIONED DB CONFIG *partitioned-db-option*

다중 데이터베이스 파티션에 분산된 테이블 로드를 실행할 수 있도록 합니다.

PARTITIONED DB CONFIG 매개변수를 사용하면 파티션된 데이터베이스에만 해당되는 구성 옵션을 지 정할 수 있습니다. *partitioned-db-option* 값은 다음과 같이 지정할 수 있습니다.

```
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

이들 옵션의 자세한 설명은 *파티션된 데이터베이스 환경을 위한 로드 구성 옵션*을 참조하십시오.

RESTARTCOUNT

예약됨.

USING *directory*

예약됨.

XML 문서에서 데이터 로드 예

XML 데이터 로딩

예 1

테이블에 삽입되는 문서를 설명하기 위해 사용자가 XDS 필드로 데이터 파일을 구성했습니다. 다음과 같이 표시됩니다.

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

첫 번째 행의 경우, XML 문서는 `file1.xml` 파일로 식별됩니다. 문자 분리문자는 큰따옴표이며 큰따옴표는 XDS 안에 존재하므로, XDS 안에 포함된 큰따옴표는 이중이 됩니다. 두 번째 행의 경우, XML 문서는 `file2.xml` 파일로 식별되며, 바이트 오프셋 23에서 시작하며 45바이트 길이입니다.

예 2:

사용자는 XML 컬럼의 구문 분석 또는 유효성 확인 옵션 없이 로드 명령을 발행하며, 데이터는 정상적으로 로드됩니다.

```
LOAD FROM data.del of DEL INSERT INTO mytable
```

커서에서 XML 데이터 로딩

커서에서 데이터 로딩은 일반 관계형 컬럼 유형과 동일합니다. 두 개의 테이블 T1 및 T2가 있으며, 각각 C1이라는 단일 XML 컬럼으로 구성됩니다. T1에서 T2로 로드하기 위해 사용자는 우선 커서를 선언합니다.

```
DECLARE X1 CURSOR FOR SELECT C1 FROM T1;
```

다음으로 사용자는 커서 유형을 사용하여 LOAD를 발행할 수 있습니다.

```
LOAD FROM X1 of CURSOR INSERT INTO T2
```

XML 특정 LOAD 옵션을 커서 유형에 적용하는 것은 파일에서 로드하는 것과 같습니다.

사용 시 참고사항

- 데이터는 입력 파일에 나타나는 시퀀스로 로드됩니다. 특정 시퀀스를 원할 경우 로드를 시도하기 전에 데이터를 정렬해야 합니다. 소스 데이터 순서의 보존이 필요하지 않은 경우, *로드 유틸리티의 파일 유형 수정자* 섹션에서 아래 설명된 ANYORDER 파일 유형 수정자 사용을 고려하십시오.
- 로드 유틸리티는 기존 정의에 따라 인덱스를 빌드합니다. 고유 키에 대한 중복을 처리하기 위해 예외 테이블을 사용합니다. 이 유틸리티는 참조 무결성을 강제하거나 제한조건 점검을 수행하거나 로드되는 테이블에 종속된 구체화된 쿼리 테이블을 갱신하지 않습니다. 참조 또는 점검 제한조건을 포함하는 테이블은 무결성 설정 보류 상태에 놓입니다. 로드되고 있는 테이블에 종속되고 REFRESH IMMEDIATE를 사용하여 정의된 요약 테이블도 무결성 설정 보류 상태에 놓입니다. 이들 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY문을 발행하십시오. 복제된 구체화된 쿼리 테이블에서는 로드 조작을 수행할 수 없습니다.
- 테이블에 클러스터링 인덱스가 존재할 경우, 로드하기 전에 클러스터링 인덱스에 대해 데이터를 정렬해야 합니다. 그러나 다차원적으로 클러스터된(MDC) 테이블에 로드할 경우에는 미리 데이터를 정렬할 필요가 없습니다.
- 보호 설정된 테이블에 로드할 때 예외 테이블을 지정하면, 유효하지 않은 보안 레이블로 보호 설정된 행은 이 테이블로 보내집니다. 따라서 예외 테이블에 액세스할 수 있는 사용자는 일반적으로 액세스 권한이 없는 데이터에 액세스할 수 있습니다. 보안 강화를 위해 누구에게 예외 테이블 액세스 권한을 부여할 것인지 유의하고, 행을 수리한 후 로드되고 있는 테이블에 복사한 후에는 곧바로 각 행을 삭제하고 예외 테이블 사용을 완료하면 곧바로 예외 테이블을 삭제하십시오.
- 내부 형식으로 된 보안 레이블에는 줄 바꾸기 문자가 포함될 수 있습니다. DEL 파일 형식을 사용하여 파일을 로드할 경우, 이들 줄 바꾸기 문자는 분리문자로 오인될

수 있습니다. 이러한 문제점이 발생하면 LOAD 명령에 delprioritychar 파일 유형 수정자를 지정하여 분리문자에 대해 이전 디폴트 우선순위를 사용하십시오.

- DECLARE CURSOR 명령 중에 DATABASE 키워드가 지정된 CURSOR 파일 유형을 사용하여 로드 조작을 수행할 경우, 현재(로드 중에) 연결된 데이터베이스에 대해 인증할 때 사용한 사용자 ID 및 암호를 사용하여 소스 데이터베이스 (DECLARE CURSOR 명령의 DATABASE 옵션에 의해 지정됨)에 대해서도 인증합니다. 로드 중인 데이터베이스에 대한 연결용으로 지정된 사용자 ID 및 암호가 없을 경우 DECLARE CURSOR 명령 중에 소스 데이터베이스에 대한 사용자 ID 및 암호를 지정해야 합니다.
- 개별적 파트가 Windows 시스템에서 AIX 시스템으로 복사되는 다중 파트 PC/IXF 파일 로딩이 지원 됩니다. 모든 파일의 이름이 LOAD 명령에서 지정되어야 합니다. 예를 들어, LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1. 논리적으로 분할된 PC/IXF 파일에서 Windows 운영 체제로의 로딩은 지원되지 않습니다.
- 실패한 LOAD 재시작 시, 인덱스의 REBUILD 모드를 사용하도록 BUILD 단계를 강제하는 기존 동작을 따릅니다.
- 데이터베이스 중간에서는 XML 문서를 로드할 수 없으며 이 경우 오류 메시지 SQL1407N이 리턴됩니다.

LOAD TERMINATE 및 LOAD RESTART 사전 관리의 요약

다음 도표는 TERMINATE 지시문 아래에서 LOAD 처리의 압축 사전 관리 동작을 요약합니다.

표 39. LOAD TERMINATE 사전 관리

테이블 COMPRESS 속성	LOAD 이전에 테이블 행 데이터 사전 존재 여부	XML 스토리지 오브젝트 사전이 LOAD 이전에 존재 여부[1]	TERMINATE: LOAD REPLACE KEEPDICTIONARY 또는 LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	기존 사전을 보존합니다.	어느 쪽 사전도 보존하지 않습니다.[2]
YES	YES	NO	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.[2]
YES	NO	YES	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
YES	NO	NO	어떤 것도 보존하지 않습니다.	어떤 것도 보존하지 않습니다.
NO	YES	YES	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
NO	YES	NO	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
NO	NO	YES	기존 사전을 보존합니다.	어떤 것도 보존하지 않습니다.
NO	NO	NO	아무 것도 수행하지 않습니다.	아무 것도 수행하지 않습니다.

주:

1. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.
2. 테이블에서 데이터 캡처가 사용 가능한 특수한 경우, 테이블 행 데이터 사전이 보존됩니다.

LOAD RESTART는 최종 일관성 지점까지 테이블을 절단합니다. 압축 사전은 LOAD RESTART 처리의 파트로서, 최종 LOAD 일관성 지점이 지정된 시간에 테이블에 존재합니다. 이 경우, LOAD RESTART는 새 사전을 작성하지 않습니다. 가능한 조건에 대한 요약은 아래 테이블 4를 참조하십시오.

표 40. LOAD RESTART 사전 관리

테이블 COMPRESS 속성	LOAD 일관성 지점 이전 테이블 행 데이터 사전 존재 여부[1]	최종 로드 이전에 XML 스토리지 오브젝트 사전 존재 여부[2]	RESTART: LOAD REPLACE KEEPDICTIONARY 또는 LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	기존 사전을 보존합니다.	기존 사전을 보존합니다.
YES	YES	NO	기존의 테이블 행 데이터 사전을 빌드하고 ADC에 따라 XML 사전을 빌드합니다.	기존의 테이블 행 데이터 사전을 빌드하고 XML 사전을 빌드합니다.
YES	NO	YES	ADC에 따른 테이블 행 사전을 빌드합니다. 기존 XML 사전을 보존합니다.	테이블 행 데이터 사전을 빌드합니다. 기존 XML 사전을 보존합니다.
YES	NO	NO	ADC에 따른 XML 사전과 테이블 행 데이터를 빌드합니다.	테이블 행 데이터 및 XML 사전을 빌드합니다.
NO	YES	YES	기존 사전을 보존합니다.	기존 사전을 제거합니다.
NO	YES	NO	기존 테이블 행 데이터 사전을 보존합니다.	기존 테이블 행 데이터 사전을 제거합니다.
NO	NO	YES	기존 XML 사전을 보존합니다.	기존 XML 사전을 제거합니다.
NO	NO	NO	아무 것도 수행하지 않습니다.	아무 것도 수행하지 않습니다.

주:

1. XML 데이터 로드 시, 조작 시작부터 로드 단계 재시작 중 실패하는 로드 조작의 경우 SAVECOUNT 옵션이 무시됩니다.
2. XML 컬럼이 DB2 버전 9.7 이상에서 테이블에 추가된 경우 또는 테이블이 온라인 테이블 이동을 사용하여 이주된 경우에만 테이블의 XML 스토리지 오브젝트의 압축 사전이 작성될 수 있습니다.

로드 유틸리티의 파일 유형 수정자

표 41. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식

수정자	설명
anyorder	이 수정자는 *cpu_parallelism* 매개변수와 결합하여 사용됩니다. SMP 시스템에서 중요한 추가 성능 이점이 발생하며, 소스 데이터 순서의 보존이 필요하지 않음을 지정합니다. *cpu_parallelism* 의 값이 1이면, 이 옵션은 무시됩니다. 일관성 지점 이후의 응급 복구는 데이터가 시퀀스로 로드되도록 요구하므로 SAVECOUNT > 0인 경우 이 옵션은 지원되지 않습니다.
generatedignore	이 수정자는 생성된 모든 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 로드 유틸리티에 알립니다. 이로 인해 생성된 모든 컬럼의 값이 유틸리티에 의해 생성됩니다. 이 수정자는 `generatedmissing` 또는 `generatedoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
generatedmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 ID 컬럼의 데이터가 포함되지 않는 것으로(NULL 값도 비포함) 가정하므로 각 행의 값을 생성합니다. 이로 인해 생성된 모든 컬럼의 값이 유틸리티에 의해 생성됩니다. 이 수정자는 `generatedignore` 또는 `generatedoverride` 수정자 중 하나와 함께 사용될 수 없습니다.

generatedoverride	이 수정자는 테이블에서 생성되는 모든 컬럼의 사용자 제공 데이터를 승인하도록 로드 유틸리티에 지시합니다(이러한 유형의 컬럼에 대한 일반 규칙과 반대). 다른 데이터베이스 시스템에서 데이터를 이주하거나, ROLLFORWARD DATABASE 명령에서 RECOVER DROPPED TABLE 옵션을 사용하여 복구된 데이터에서 테이블을 로드할 때 유용합니다. 이 수정자가 사용될 때, 데이터가 없는 행이나 널(null) 값을 허용하지 않는 생성된 컬럼의 NULL 데이터는 거부됩니다(SQL3116W). 해당 수정자 사용 시, 테이블이 무결성 설정 보류 상태에 놓입니다. 사용자 제공 값을 검증하지 않고 무결성 설정 보류 상태로부터 테이블을 얻으려면, 로드 조작 이후에 다음 명령을 발행하십시오.
	`SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED`
	무결성 설정 보류 상태로부터 테이블을 얻고 사용자 제공 값의 검증을 강제하려면, 로드 조작 이후에 다음 명령을 발행하십시오.
	`SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.`
	이 수정자가 지정되고 임의의 파티셔닝 키, 차원 키 또는 분산 키에 생성된 컬럼이 있으면, LOAD 명령은 자동으로 수정자를 `generatedignore`로 변환하고 로드를 진행합니다. 이것은 생성된 모든 컬럼 값을 재생성하는 효과를 갖습니다.
	이 수정자는 `generatedmissing` 또는 `generatedignore` 수정자 중 하나와 함께 사용될 수 없습니다.

identityignore	이 수정자는 ID 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 로드 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ID 값이 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 `identitymissing` 또는 `identityoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
identitymissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 ID 컬럼의 데이터를 포함하지 않음(NULL 값도 비포함)을 가정하므로 각 행의 값 을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT ID 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 `identityignore` 또는 `identityoverride` 수정자 중 하나와 함께 사용될 수 없습니다.

표 41. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
identityoverride	GENERATED ALWAYS로 정의된 ID 컬럼이 로드되는 테이블에 있을 때에만 이 수정자를 사용해야 합니다. 해당 컬럼에 대해 명시적, 널(NULL)이 아닌 데이터를 승인하도록 유틸리티에 지시합니다(이러한 유형의 ID 컬럼에 대한 일반 규칙과 반대). 테이블이 GENERATED ALWAYS로 정의되어야 할 때 다른 데이터베이스 시스템에서 데이터를 이주하거나, ROLLFORWARD DATABASE 명령에서 DROPPED TABLE RECOVERY 옵션을 사용하여 복구된 데이터에서 테이블을 로드할 때 유용합니다. 이 수정자가 사용될 때, 데이터가 없는 행이나 ID 컬럼의 NULL 데이터는 거부됩니다(SQL3116W). 이 수정자는 `identitymissing` 또는 `identityignore` 수정자 중 하나와 함께 사용될 수 없습니다. 이 옵션 사용 시 로드 유틸리티는 테이블의 ID 컬럼에서 값의 고유성을 검증하거나 유지하려고 시도하지 않습니다.
indexfreespace=*x*	*x*는 0 - 99의 정수입니다. 로드로 인덱스를 재빌드할 때 여유 공간으로 남게 되는 각 인덱스 페이지의 백분율입니다. INDEXING MODE INCREMENTAL을 사용하여 로드할 때 이 옵션은 무시됩니다. 페이지의 첫 번째 항목은 제한없이 추가되며, 여유 공간 비율 임계값을 유지하도록 추후 항목이 추가됩니다. 디폴트값은 CREATE INDEX 시간에 사용됩니다. 이 값은 CREATE INDEX문에 지정된 PCTFREE 값보다 우선합니다. indexfreespace 옵션은 인덱스 단말 페이지에만 영향을 미칩니다.
lobsinfile	*lob-path*는 LOB 데이터를 포함하는 파일에 대한 경로를 지정합니다. ASC, DEL 또는 IXF 로드 입력 파일은 LOB 컬럼에 LOB 데이터가 있는 파일의 이름을 포함합니다. 이 옵션은 CURSOR 파일 유형과 함께 지원되지 않습니다. LOBS FROM 절은 『lobsinfile』 수정자가 사용될 때 LOB 파일이 위치하는 곳을 지정합니다. LOBS FROM 절은 내재적으로 LOBSINFILE 동작을 활성화합니다. LOBS FROM 절은 데이터 임포트 중 LOB 파일을 검색하기 위해 경로 목록을 LOAD 유틸리티로 전달합니다. 각 경로는 LLS(Lob Location Specifier)에 의해 포인트되는 최소한 하나의 LOB를 포함하는 최소한 하나의 파일을 데이터 파일에 포함합니다. LLS는 LOB 파일 경로에 저장된 파일에서 LOB 위치의 문자열 표시입니다. LLS의 형식은 *filename.ext.nnn.mmm/*이며, 여기서 *filename.ext*는 LOB를 포함하는 파일의 이름이며, *nnn*은 파일 내에서 LOB의 오프셋을 바이트로 나타낸 것이며, *mmm*은 LOB의 길이를 바이트로 나타낸 것입니다. 예를 들어, 문자열 `db2exp.001.123.456/`가 데이터 파일에 저장되는 경우, LOB는 db2exp.001 파일에서 오프셋 123에 위치하며 456바이트 길이입니다. 널(NULL) LOB를 표시하려면, 크기를 -1로 입력하십시오. 크기가 0으로 지정되면, 길이가 0인 LOB로 처리됩니다. 길이가 -1인 널(NULL) LOBS의 경우, 오프셋 및 파일 이름은 무시됩니다. 예를 들어, 널(NULL) LOB의 LLS는 db2exp.001.7.-1/입니다.
noheader	헤더 검증 코드를 건너뜁니다(단일 파티션 데이터베이스 파티션 그룹에 있는 테이블로 조작을 로드하는 경우만 적용 가능). 디폴트 MPP 로드(모드 PARTITION_AND_LOAD)가 단일 파티션 데이터베이스 파티션 그룹에 있는 테이블에 대해 사용되는 경우, 파일에 헤더가 없습니다. 그러므로 헤더 수정자는 필요하지 않습니다. LOAD_ONLY 모드가 사용되는 경우, 파일에 헤더가 있습니다. 헤더가 없는 수정자를 사용할 필요가 있는 유일한 환경은 헤더가 없는 파일을 사용하여 LOAD_ONLY를 수행하려고 하는 경우입니다.
norowwarnings	거부된 행에 대한 모든 경고를 제외시킵니다.

표 41. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명
pagefreespace=*x*	*x*는 0 - 100의 정수입니다. 여유 공간으로 남게 되는 각 데이터 페이지의 백분율입니다. 최소 행 크기 때문에 지정된 값이 유효하지 않은 경우, (예를 들어, 길이가 최소한 3000바이트이고, *x* 값이 50인 행), 새 페이지에 행이 위치합니다. 100 값이 지정된 경우, 각 행은 새 페이지에 있습니다. 테이블의 PCTFREE 값은 페이지 당 지정된 여유 공간의 크기를 결정합니다. 로드 조작의 pagefreespace 값이나 테이블의 PCTFREE 값이 설정되지 않은 경우, 유틸리티는 각 페이지에 가능한 많은 스페이스를 채웁니다. pagefreespace로 설정된 값은 테이블에 지정된 PCTFREE 값을 겹쳐씁니다.
rowchangetimestampignore	이 수정자는 행 변경 시간소인 컬럼의 데이터가 데이터 파일에 표시되지만 무시해야 함을 로드 유틸리티에 알립니다. 이로 인해 유틸리티에 의해 모든 ROW CHANGE TIMESTAMP가 생성됩니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. GENERATED ALWAYS 컬럼의 경우, 어떤 행도 거부되지 않음을 의미합니다. 이 수정자는 `rowchangetimestampmissing` 또는 `rowchangetimestampoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
rowchangetimestampmissing	이 수정자가 지정된 경우, 유틸리티는 입력 데이터 파일에는 행 변경 시간소인 컬럼의 데이터를 포함하지 않음(NULL 값도 비포함)을 가정하므로 각 행의 값을 생성합니다. GENERATED ALWAYS 및 GENERATED BY DEFAULT 컬럼 둘 다의 동작이 동일합니다. 이 수정자는 `rowchangetimestampignore` 또는 `rowchangetimestampoverride` 수정자 중 하나와 함께 사용될 수 없습니다.
rowchangetimestampoverride	GENERATED ALWAYS로 정의된 행 변경 시간소인 컬럼이 로드되는 테이블에 있을 때에만 이 수정자를 사용해야 합니다. 해당 컬럼에 대해 명시적, 널(NULL)이 아닌 데이터를 승인하도록 유틸리티에 지시합니다(이러한 유형의 시간소인 컬럼에 대한 일반 규칙 과 반대). 테이블이 GENERATED ALWAYS로 정의되어야 할 때 다른 데이터베이스 시스템에서 데이터를 이주하거나, ROLLFORWARD DATABASE 명령에서 DROPPED TABLE RECOVERY 옵션을 사용하여 복구된 데이터에서 테이블을 로드할 때 유용합니다. 이 수정자가 사용될 때, 데이터가 없는 행이나 ROW CHANGE TIMESTAMP 컬럼의 NULL 데이터는 거부됩니다(SQL3116W). 이 수정자는 `rowchangetimestampmissing` 또는 `rowchangetimestampignore` 수정자 중 하나와 함께 사용될 수 없습니다. 이 옵션 사용 시 로드 유틸리티는 테이블의 행 시간소인 컬럼에서 값의 고유성을 검증하거나 유지하려고 시도하지 않습니다.

seclabelchar	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 보안 레이블 값의 문자열 형식임을 표시합니다. LOAD는 각 보안 레이블을 로드된 대로의 내부 형식으로 변환합니다. 문자열이 적절한 형식으로 되어 있지 않은 경우 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3242W)가 리턴됩니다. 문자열이 테이블을 보호하는 보안 규정 파트인 유효한 보안 레이블을 나타내지 않는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3243W)가 리턴됩니다.
	
	seclabelname 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 로드에 실패하며 오류(SQLCODE SQL3525N)가 리턴됩니다.
	
	테이블이 단일 `DB2SECURITYLABEL` 컬럼으로 구성된 경우, 데이터 파일은 다음과 같습니다.
	`"CONFIDENTIAL:ALPHA:G2"`
	`"CONFIDENTIAL;SIGMA:G2"`
	`"TOP SECRET:ALPHA:G2"`
	
	이 데이터를 로드하거나 임포트하려면, SECLABELCHAR 파일 유형 수정자가 사용되어야 합니다.
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1`

표 41. 로드 유틸리티의 유효한 파일 유형 수정자: 모든 파일 형식 (계속)

수정자	설명

seclabelname	입력 소스 파일의 보안 레이블이 디폴트로 인코드된 숫자 형식이 아니라 이름으로 표시됨을 나타냅니다. LOAD는 이름이 있는 경우 적절한 보안 레이블로 변환합니다. 테이블을 보호하는 보안 규정에 대해 표시된 이름이 있는 보안 레이블이 없는 경우, 행은 로드되지 않으며 경고(SQLSTATE 01H53, SQLCODE SQL3244W)가 리턴됩니다.
	
	seclabelchar 수정자가 지정된 경우 이 수정자를 지정할 수 없으며, 그렇지 않으면 로드에 실패하며 오류(SQLCODE SQL3525N)가 리턴됩니다.
	
	테이블이 단일 DB2SECURITYLABEL 컬럼으로 구성된 경우, 데이터 파일은 다음과 유사한 보안 레이블 이름으로 구성됩니다.
	
	`"LABEL1"`
	`"LABEL1"`
	`"LABEL2"`
	
	이 데이터를 로드하거나 임포트하려면, SECLABELNAME 파일 유형 수정자가 사용되어야 합니다.
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1`
	
	주: 파일 유형이 ASC인 경우, 보안 레이블의 이름 다음에 오는 모든 스페이스는 이름의 일부로 해석됩니다. 이를 피하려면 striptblanks 파일 유형 수정자를 사용하여 스페이스가 제거되었는지 확인하십시오.

totalfreespace=*x*	*x*는 0 이상의 정수입니다. 이 값은 여유 공간으로 테이블의 끝에 추가되는 테이블에서 총 페이지의 백분율로 해석됩니다. 예를 들어, *x*가 20이며 데이터가 로드된 이후에 테이블에 100개의 데이터 페이지가 있는 경우, 20개의 추가적인 빈 페이지가 추가됩니다. 테이블의 총 데이터 페이지 수는 120입니다. 데이터 페이지 총 수는 테이블에서 인덱스 페이지의 수를 계산에 넣지 않습니다. 이 옵션은 인덱스 오브젝트에 영향을 미치지 않습니다. 이 옵션이 지정된 두 개의 로드가 완료되면, 두 번째 로드는 첫 번째 로드에서 끝에 추가한 여유 스페이스를 재사용하지 않습니다.
usedefaults	목표 테이블 컬럼의 소스 컬럼이 지정되었지만 하나 이상의 행 인스턴스에 대한 데이터를 포함하지 않은 경우, 디폴트값이 로드됩니다. 누락된 데이터의 예: • DEL 파일: 임의 수의 스페이스(″, ,″)로 구분되는 2개의 인접 컬럼 분리문자나 2개의 인접 컬럼 분리문자(″,,″)가 컬럼 값으로 지정됩니다. • DEL/ASC/WSF 파일: 행의 컬럼이 충분하지 않거나 행의 길이가 원래 스펙만큼 충분하지 않습니다. ASC 파일의 경우, NULL 컬럼 값은 명시적으로 누락된 것으로 간주되지 않으며 디폴트가 NULL 컬럼 값을 대신하지 않습니다. 숫자, 날짜, 시간 및 /시간소인 컬럼의 모든 공백 문자로 NULL 컬럼 값을 표시하거나, 컬럼이 NULL임을 표시하기 위해 모든 유형의 컬럼에 널(NULL) 표시기를 사용함으로써 NULL 컬럼 값을 표시합니다. 이 옵션이 없는 경우, 소스 컬럼이 행 인스턴스의 데이터를 포함하지 않으면 다음 중 하나가 발생합니다. • DEL/ASC/WSF 파일: 컬럼에 널(NULL) 입력 가능한 경우 NULL이 로드됩니다. 컬럼이 널(NULL) 입력 가능하지 않으면 유틸리티가 행을 거부합니다.

표 42. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL)

수정자	설명
codepage=*x*	*x*는 ASCII 문자열입니다. 해당 값은 입력 데이터 세트에서 데이터의 코드 페이지로서 해석됩니다. 로드 조작 중 이 코드 페이지에서 데이터베이스 코드 페이지로 문자 데이터(및 문자로 지정된 숫자 데이터)를 변환합니다. 다음 규칙이 적용됩니다. • 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다. • EBCDIC 코드 페이지에 지정된 DEL 데이터의 경우 분리문자는 시프트 인(Shift-In) 및 시프트 아웃(Shift-Out) DBCS 문자와 동시에 사용할 수 없습니다. • nullindchar은 x20 - x7F 코드 포인트의 표준 ASCII 세트에 포함된 기호를 지정해야 합니다. 이것은 ASCII 기호 및 코드 포인트를 나타냅니다. 코드 포인트가 달라도 EBCDIC 데이터는 해당 기호를 사용할 수 있습니다. 이 옵션은 CURSOR 파일 유형과 함께 지원되지 않습니다.

dateformat="*x*"	*x*는 소스 파일에서 날짜의 형식입니다.[1] 유효한 날짜 요소는 다음과 같습니다.
	
	`YYYY - 연도(0000 - 9999 사이의 4자리 숫자)`
	`M - 월(1 - 12 사이의 1 또는 2자리 숫자)`
	`MM - 월(1 - 12 사이의 2자리 숫자, M과 상호 배타적)`
	`D - 일(1 - 31 사이의 1 또는 2자리 숫자)`
	`DD - 일(1 - 31 사이의 2자리 숫자, D와 상호 배타적)`
	`DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적)`
	
	지정되지 않은 각 요소에 대해 디폴트값 1이 지정됩니다. 날짜 형식의 예:
	
	`"D-M-YYYY"`
	`"MM.DD.YYYY"`
	`"YYYYDDD"`

dumpfile = *x*	*x*는 거부된 행이 쓰여진 예외 파일의 완전한 이름(서버 데이터베이스 파티션에 따른)입니다. 레코드당 최대 32KB의 데이터가 쓰여집니다. 다음은 덤프 파일을 지정하는 방법을 표시하는 예입니다.
	
	`db2 load from data of del`
	`modified by dumpfile = /u/user/filename`
	`insert into table_name`
	
	인스턴스 소유자가 파일을 작성하고 소유합니다. 디폴트 파일 권한을 겹쳐쓰려면, dumpfileaccessall 파일 유형 수정자를 사용하십시오.
	주:
	1. 파티션된 데이터베이스 환경에서, 경로는 로딩 데이터베이스 파티션에 로컬이어야 하므로 현재 실행 중인 로드 조작은 동일한 파일에 쓰지 않습니다.
	2. 파일의 컨텐츠는 비동기 버퍼 지정 모드로 디스크에 쓰여집니다. 실패하거나 인터럽트된 로드 조작의 경우, 디스크에 커미트된 레코드 수는 확실히 알 수 없으며 LOAD RESTART 이후에 일관성을 보장할 수 없습니다. 파일에서는 단일 패스로 시작하고 완료하는 로드 조작에 대해서만 완료될 것으로 가정합니다.
	3. 지정된 파일이 이미 존재하는 경우, 재작성되지 않지만 추가됩니다.

표 42. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
dumpfileaccessall	덤프 파일 작성 시 'OTHERS'에 대한 읽기 액세스를 권한 부여합니다. 이 파일 유형 수정자는 다음 경우에만 유효합니다. 1. dumpfile 파일 유형 수정자와 결합하여 사용됩니다. 2. 사용자는 로드 목표 테이블에 대한 SELECT 특권을 갖습니다. 3. UNIX 운영 체제에 있는 DB2 서버 데이터베이스 파티션에서 발행됩니다. 지정된 파일이 이미 존재하는 경우, 권한을 변경하지 못합니다.
fastparse	주의하여 사용해야 합니다. 사용자 제공 컬럼 값에 대한 구문 검사를 줄이고, 성능을 향상시킵니다. 테이블은 구조적으로 옳은 것으로 보장되지만(유틸리티는 충분한 데이터 검사를 수행하여 분할 위반 또는 트랩을 방지함), 데이터 일관성의 유효성은 확인되지 않습니다. 데이터가 일관되고 올바름을 확신하는 경우에만 이 옵션을 사용하십시오. 예를 들어, 사용자 제공 데이터가 유효하지 않은 시간소인 컬럼 값 :1>0-00-20-07.11.12.000000을 포함하는 경우, FASTPARSE가 지정되면 이 값이 테이블에 삽입되며 FASTPARSE가 지정되지 않으면 거부됩니다.
implieddecimal	내포된 소수점의 위치는 컬럼 정의로 판별되며, 값의 끝으로 가정되지 않습니다. 예를 들어, 12345 값은 12345.00이 아닌 123.45로 DECIMAL(8,2) 컬럼에 로드됩니다. 이 수정자는 packeddecimal 수정자와 함께 사용될 수 없습니다.
timeformat="*x*"	*x*는 소스 파일에서 시간의 형식입니다.[1] 유효한 시간 요소는 다음과 같습니다. H - 시간(12시간 시스템의 경우 0 - 12 사이의 1 또는 2자리 숫자, 0 - 12 사이의 2자리 숫자, 24시간 시스템의 경우 0 - 24) HH - 시간(12시간 시스템의 경우 0 - 12 사이의 2자리 숫자, 24시간 시스템의 경우 0 - 24. H와 상호 배타적) M - 분(0 - 59 사이의 1 또는 2자리 숫자) MM - 분(0 - 59 사이의 2자리 숫자, M과 상호 배타적) S - 초(0 - 59 사이의 1 또는 2자리 숫자) SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적) SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적) TT - 오전/오후 지시(AM 또는 PM) 지정되지 않은 각 요소에 대해 디폴트값 0이 지정됩니다. 시간 형식의 예: "HH:MM:SS" "HH.MM TT" "SSSSS"

표 42. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
timestampformat="*x*"	*x*는 소스 파일에서 시간소인의 형식입니다.[1] 유효한 시간소인 요소는 다음과 같습니다. YYYY - 연도(0000 - 9999 사이의 4자리 숫자) M - 월(1 - 12 사이의 1 또는 2자리 숫자) MM - 월(01 - 12 사이의 2자리 숫자, M 및 MMM은 상호 배타적) MMM - 월(월 이름으로 사용되는 3자리 문자로 대소문자를 구분하지 않으며, M 및 MM은 상호 배타적) D - 일(1 - 31 사이의 1 또는 2자리 숫자) DD - 일(1 - 31 사이의 2자리 숫자, D와는 상호 배타적) DDD - 년의 일(001 - 366 사이의 3자리 숫자, 다른 일 또는 월 요소와 상호 배타적) H - 시간(12시간 시스템의 경우 0 - 12 사이의 1 또는 2자리 숫자, 24시간 시스템의 경우 0 - 24) HH - 시간(12시간 시스템의 경우 0 - 12 사이의 2자리 숫자, 24시간 시스템의 경우 0 - 24, H와는 상호 배타적) M - 분(0 - 59 사이의 1 또는 2자리 숫자) MM - 분(0 - 59 사이의 2자리 숫자, M, 분과 상호 배타적) S - 초(0 - 59 사이의 1 또는 2자리 숫자) SS - 초(0 - 59 사이의 2자리 숫자, S와 상호 배타적) SSSSS - 자정 후 초(00000 - 86399 사이의 5자리 숫자, 다른 시간 요소와 상호 배타적) U(1 - 12회) - 초의 소수 부분(U 어커런스 수는 각 자리가 0 - 9 사이의 자리 수를 나타냄) TT - 오전/오후 지시(AM 또는 PM)

표 42. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명

timestampformat="x"(계속)	디폴트값 1이 미지정된 YYYY, M, MM, D, DD 또는 DDD 요소에 지정됩니다. 디폴트값 'Jan'이 미지정된 MMM 요소에 지정됩니다. 미지정된 다른 모든 요소에 디폴트값 0이 지정됩니다. 다음은 시간소인 형식의 예입니다.
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	MMM 요소의 올바른 값은 다음을 포함합니다. 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' 및 'dec'. 이들 값은 대소문자가 구분되지 않습니다.
	TIMESTAMPFORMAT 수정자가 지정되지 않으면, 로드 유틸리티는 두 개의 가능한 형식 중 하나를 사용하여 시간소인 필드를 형식화합니다.
	`YYYY-MM-DD-HH.MM.SS`
	`YYYY-MM-DD HH:MM:SS`
	로드 유틸리티는 DD와 HH 간의 구분자를 확인하여 형식을 선택합니다. 대시 '-'인 경우, 로드 유틸리티는 일반 대시와 점 형식(YYYY-MM-DD-HH.MM.SS)을 사용합니다. 공백인 경우, 로드 유틸리티는 HH, MM 및 SS를 구분하기 위해 콜론 ':'을 예상합니다.
	어느 쪽 형식이든, 마이크로초 필드(UUUUUU)를 포함하는 경우, 로드 유틸리티는 점 '.'을 구분자로 예상합니다. YYYY-MM-DD-HH.MM.SS.UUUUUU 또는 YYYY-MM-DD HH:MM:SS.UUUUUU를 승인할 수 있습니다.
	다음 예는 사용자 정의 날짜 및 시간 형식을 포함하는 데이터를 schedule이라는 테이블로 로드하는 방법을 설명합니다.
	`db2 load from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

usegraphiccodepage	usegraphiccodepage가 제공되면, 그래픽 또는 2바이트 문자 대형 오브젝트(DBCLOB) 데이터 필드로 로드되는 데이터는 그래픽 코드 페이지에 있는 것으로 가정합니다. 나머지 데이터는 문자 코드 페이지에 있다고 가정합니다. 그래픽 코드 페이지는 문자 코드 페이지와 연관됩니다. LOAD는 codepage 수정자가 지정된 경우 이를 통해 문자 코드 페이지를 판별하고 codepage 수정자가 지정되지 않은 경우 데이터베이스의 코드 페이지를 통해 문자 코드 페이지를 판별합니다. 복구 중인 테이블에 그래픽 데이터가 있는 경우에만 삭제(drop) 테이블 복구로 생성되는 구분된 데이터 파일과 결합하여 이 수정자를 사용해야 합니다. **제한사항** 이들 파일이 오직 하나의 코드 페이지에 인코드된 데이터를 포함하면, usegraphiccodepage 수정자는 EXPORT 유틸리티로 작성된 DEL 파일과 함께 지정되지 않아야 합니다. usegraphiccodepage 수정자는 파일의 2바이트 문자 대형 오브젝트(DBCLOB)에서 무시됩니다.

표 42. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(ASC/DEL) (계속)

수정자	설명
xmlchar	XML 문서가 문자 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 지정된 문자 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 문자 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 문자 코드 페이지는 `codepage` 파일 유형 수정자에서 지정한 값이거나, 지정되지 않은 경우 응용프로그램 코드 페이지입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다.
xmlgraphic	XML 문서가 지정된 그래픽 코드 페이지에서 인코드됨을 지정합니다. 이 옵션은 특정 그래픽 코드 페이지에서 인코드되었지만 인코딩 선언을 포함하지 않는 XML 문서 처리에 유용합니다. 각 문서에서 선언 태그가 존재하고 인코딩 속성을 포함하는 경우 인코딩은 그래픽 코드 페이지와 일치해야 하며, 그렇지 않으면 문서를 포함하는 행이 거부됩니다. 그래픽 코드 페이지는 `codepage` 파일 유형 수정자에서 지정한 값의 그래픽 구성요소이거나, 지정되지 않은 경우 응용프로그램 코드 페이지의 그래픽 구성요소입니다. 디폴트로 문서가 유니코드로 인코드되거나 인코딩 속성을 가진 선언 태그를 포함합니다.

표 43. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(구분되지 않은 ASCII)

수정자	설명
binarynumerics	숫자(10진수는 아님) 데이터는 문자 표시가 아닌 2진 양식이어야 합니다. 손실이 큰 변환은 피합니다. 이 옵션은 `reclen` 옵션으로 지정된 고정 길이 레코드를 사용하며 위치상 ASC에서만 지원됩니다. 다음 규칙이 적용됩니다. • 데이터 유형 간의 변환은 수행되지 않으며, BIGINT, INTEGER 및 SMALLINT 예외가 있습니다. • 데이터 길이는 목표 컬럼 정의와 일치해야 합니다. • FLOAT는 IEEE 부동 소수점 형식이어야 합니다. • 로드 소스 파일의 2진 데이터는 로드 조작이 실행되는 플랫폼에 관계 없이 빅 엔디안(big-endian)으로 가정됩니다. 이 수정자에 의해 영향을 받는 컬럼의 데이터에 NULL 값을 표시할 수 없습니다. 이 수정자가 사용될 때 공백(보통 NULL로 해석)은 2진 값으로 해석됩니다.
nochecklengths	`nochecklengths`가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 로드하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 로드될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.

표 43. 로드 유틸리티의 유효한 파일 유형 수정자: ASCII 파일 형식(구분되지 않은 ASCII) (계속)

수정자	설명
nullindchar=*x*	*x*는 단일 문자입니다. 널(NULL) 값을 나타내는 문자를 *x*로 변경합니다. *x*의 디폴트값은 Y입니다. [2]
	문자가 영문자인 경우를 제외하고 EBCDIC 데이터 파일의 경우 이 수정자의 대소문자를 구분합니다. 예를 들어, 널(NULL) 표시기 문자가 N 문자가 되도록 지정되는 경우, n은 널(NULL) 표시기로 인식됩니다.
packeddecimal	binarynumerics 수정자가 DECIMAL 필드 유형을 포함하지 않으므로 압축 10진수 데이터를 직접 로드합니다.
	이 옵션은 reclen 옵션으로 지정된 고정 길이 레코드를 사용하며 위치상 ASC에서만 지원됩니다.

	지원되는 부호 니블의 값은 다음과 같습니다.
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`

	이 수정자에 의해 영향을 받는 컬럼의 데이터에 NULL 값을 표시할 수 없습니다. 이 수정자가 사용될 때 공백(보통 NULL로 해석)은 2진 값으로 해석됩니다.
	서버 플랫폼에 관계 없이, 로드 소스 파일의 2진 데이터의 바이트 순서는 빅 엔디안(big-endian)으로 가정되므로, Windows 운영 체제에서 이 수정자를 사용하면 바이트 순서는 바뀌지 않아야 합니다.
	이 수정자는 implieddecimal 수정자와 함께 사용될 수 없습니다.
reclen=*x*	*x*는 최대값이 32 767인 정수입니다. 각 행에 대해 *x* 문자가 읽혀지며 행의 끝을 표시하기 위한 줄 바꾸기 문자는 사용되지 않습니다.
striptblanks	데이터를 변수 길이 필드로 로드할 때 뒤 공백을 절단합니다. 이 옵션이 지정되지 않으면, 공백이 보존됩니다.
	이 옵션은 striptnulls와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다. 이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 t 옵션을 교체합니다.
striptnulls	데이터를 변수 길이 필드로 로드할 때 뒤 NULL 값(0x00 문자)을 절단합니다. 이 옵션이 지정되지 않으면, NULL 값이 보존됩니다.
	이 옵션은 striptblanks와 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다. 이 옵션은 이전 호환성을 위해서만 지원되는 사용 안하는 padwithzero 옵션을 교체합니다.
zoneddecimal	BINARYNUMERICS 수정자는 DECIMAL 필드 유형을 포함하지 않으므로 존 10진수(zoned DECIMAL) 데이터를 로드합니다. 이 옵션은 RECLEN 옵션으로 지정된 고정 길이 레코드를 사용하며 위치상 ASC에서만 지원됩니다.

	반 바이트 기호 값은 다음 중 하나일 수 있습니다.
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`

	지원되는 숫자 값은 0x0 - 0x9입니다.
	지원되는 존 값은 0x3 및 0xF입니다.

표 44. 로드 유틸리티의 유효한 파일 유형 수정자: DEL 파일 형식(컬럼 식별자가 있는 ASCII)

수정자	설명

chardel*x*	*x*는 단일 문자열 분리문자입니다. 디폴트값은 큰따옴표(")입니다. 큰따옴표 대신 지정된 문자를 사용하여 문자열을 묶습니다. [23] 명시적으로 큰따옴표(")를 문자열 분리문자로 지정하려는 경우 다음과 같이 지정해야 합니다.
	`modified by chardel""`
	다음과 같이 작은따옴표(')를 문자열 분리문자로 지정할 수도 있습니다.
	`modified by chardel''`

coldel*x*	*x*는 단일 문자 컬럼 분리문자입니다. 디폴트값은 쉼표(,)입니다. 지정된 문자가 쉼표 대신 사용되어 컬럼 끝을 나타냅니다.[23]
decplusblank	플러스 부호 문자. 플러스 부호(+) 대신 공백을 양의 10진수 값 앞에 붙입니다. 디폴트 조치는 플러스 부호를 양의 10진수 값 앞에 붙이는 것입니다.
decpt*x*	*x*는 소수점 문자로 마침표를 대신하는 단일 문자입니다. 디폴트값은 마침표(.)입니다. 마침표 대신 지정된 문자가 소수점 문자로 사용됩니다.[23]

delprioritychar	분리문자의 현재 디폴트 우선순위: 레코드 구분 문자, 문자 분리문자, 컬럼 분리문자. 이 수정자는 분리문자 우선순위를 문자 분리문자, 레코드 구분 문자, 컬럼 분리문자로 되돌림으로써 이전 우선순위에 따른 기존 응용프로그램을 보호합니다. 구문:
	`db2 load ... modified by delprioritychar ...`
	예를 들면, 다음과 같은 DEL 데이터 파일이 있습니다.
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	`delprioritychar` 수정자를 지정하고, 이 데이터 파일에는 두 개의 행만이 있습니다. 두 번째 <행 분리문자>는 두 번째 행의 첫 번째 데이터 컬럼의 파트로 해석되지만, 첫 번째 및 세 번째 <행 분리문자>는 실제 레코드 구분 문자로 해석됩니다. 이 수정자가 지정되지 않은 경우, 이 데이터 파일에는 세 개의 행이 있으며, 각 행은 <행 분리문자>로 구분됩니다.

keepblanks	유형 CHAR, VARCHAR, LONG VARCHAR 또는 CLOB의 각 필드에 앞뒤 공백을 둡니다. 이 옵션이 없으면, 문자 분리문자 내에 없는 앞 공백과 뒤 공백은 모두 제거되며 공백 필드의 테이블로 NULL이 삽입됩니다.
	다음 예는 데이터 파일에서 앞과 뒤의 모든 공백을 보존하면서 TABLE1이라는 테이블로 데이터를 로드하는 방법을 설명합니다.
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nochardel	로드 유틸리티는 컬럼 분리문자 간의 모든 바이트를 컬럼 데이터의 파트로 가정합니다. 문자 분리문자는 컬럼 데이터의 파트로 구문 분석됩니다. DB2를 사용하여 데이터를 익스포트한 경우(익스포트 시 nochardel을 지정한 경우를 제외하고) 이 옵션을 지정하지 말아야 합니다. 이 옵션은 문자 분리문자가 없는 벤더 데이터 파일을 지원하기 위해 제공됩니다. 부적절한 사용은 데이터 유실이나 손상을 초래할 수 있습니다. 이 옵션은 `chardelx`, `delprioritychar` 또는 `nodoubledel`과 함께 지정될 수 없습니다. 이들은 상호 독점 옵션입니다.
nodoubledel	2바이트 분리문자를 인식하지 않습니다.

표 45. 로드 유틸리티의 유효한 파일 유형 수정자: IXF 파일 형식

수정자	설명
forcein	코드 페이지 불일치와 관계없이 데이터를 승인하고 코드 페이지 간에 변환하지 않도록 유틸리티에 지시합니다. 데이터의 고정 길이 대상 필드가 충분히 큰지 검증하도록 고정 길이 대상 필드를 점검합니다. nochecklengths가 지정된 경우, 검사가 수행되지 않으며 각 행을 로드하려고 시도합니다.
nochecklengths	nochecklengths가 지정되면, 목표 테이블 컬럼의 크기를 초과하는 컬럼 정의가 소스 데이터에 있는 경우에도 각 행을 로드하려고 시도합니다. 코드 페이지 변환으로 소스 데이터가 축소되는 경우 그러한 행은 정상적으로 로드될 수 있습니다. 예를 들어, 소스에서 4바이트 EUC 데이터는 목표에서 2바이트 DBCS 데이터로 축소되어 반 정도의 스페이스가 필요합니다. 불일치 컬럼 정의에도 불구하고 소스 데이터가 모든 경우에 맞는 경우 특히 이 옵션이 유용합니다.

주:

1. 날짜 출력 문자열을 둘러싼 큰따옴표는 필수입니다. 필드 구분자는 a - z, A - Z 및 0 - 9를 포함할 수 없습니다. 필드 구분자는 DEL 파일 형식의 필드 분리문자나 문자 분리문자와 같지 않아야 합니다. 요소의 시작 및 종료 위치가 명확한 경우 필드 구분자가 선택적입니다. D, H, M 또는 S와 같은 요소가 사용되는 경우(수정자에 따라) 항목의 변수 길이 때문에 모호함이 있을 수 있습니다.

   시간소인 형식의 경우, 문자 M을 사용하는 month 및 minute 디스크립터 간의 모호함을 피하려면 주의해야 합니다. month 필드는 다른 날짜 필드와 인접해야 합니다. minute 필드는 다른 시간 필드와 인접해야 합니다. 다음은 모호한 시간소인 형식입니다.

   ```
   "M"(month 또는 minute일 수 있음)
   "M:M"(month 및 minute 구분 가능?)
   "M:YYYY:M"(둘 다 month로 해석됨.)
   "S:M:YYYY"(둘 다 시간 값 및 날짜 값에 인접)
   ```

   모호한 경우, 유틸리티는 오류 메시지를 발행하며, 조작에 실패합니다.

   다음은 명확한 시간소인 형식입니다.

   ```
   "M:YYYY" (Month)
   "S:M" (Minute)
   "M:YYYY:S:M" (Month....Minute)
   "M:H:YYYY:M:D" (Minute....Month)
   ```

   큰따옴표 및 백슬래시와 같은 일부 문자는 Escape 문자(예: ₩)가 앞에 와야 합니다.

2. chardel, coldel 또는 decpt 파일 유형에 제공되는 문자 값은 소스 데이터의 코드 페이지에 지정되어야 합니다.

문자 코드 포인트(문자 기호 대신)는 구문 xJJ 또는 0xJJ를 사용하여 지정될 수 있으며, 여기서 JJ는 코드 포인트의 16진수를 나타냅니다. 예를 들어, 컬럼 분리문자로 # 문자를 지정하려면, 다음 중 하나를 사용하십시오.

```
... modified by coldel# ...
... modified by coldel0x23 ...
... modified by coldelX23 ...
```

3. *데이터 이동을 위한 분리문자 고려사항*은 분리문자를 겹쳐쓰기할 때 사용할 수 있는 문자에 적용하는 제한사항을 나열합니다.
4. MODIFIED BY 옵션과 함께 지원되지 않는 파일 유형을 사용하려고 시도하는 경우 로드 유틸리티는 경고를 발행하지 않습니다. 이런 경우, 로드 조작에 실패하며 오류 코드가 리턴됩니다.
5. 내재적으로 숨겨진 행 변경 시간소인 컬럼을 포함하는 테이블로 임포트할 때, 내재적으로 숨겨진 컬럼의 등록 정보는 무시됩니다. 그러므로, 컬럼의 데이터가 임포트되는 데이터에 표시되지 않으며 명시적 컬럼 목록이 표시되지 않으면 rowchangetimestampmissing 파일 유형 수정자가 임포트 명령에 지정*되어야* 합니다.

표 46. codepage 및 usegraphiccodepage 사용 시 LOAD 동작

codepage=N	**usegraphiccodepage**	**LOAD 동작**
Absent	Absent	CLIENT 옵션이 지정된 경우에도 파일의 모든 데이터는 응용프로그램 코드 페이지가 아닌 데이터베이스 코드 페이지에 있는 것으로 가정됩니다.
Present	Absent	파일의 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 로드될 때 그래픽 데이터가 손상됩니다.
Absent	Present	CLIENT 옵션이 지정된 경우에도 파일의 문자 데이터는 데이터베이스 코드 페이지에 있는 것으로 가정됩니다. CLIENT 옵션이 지정된 경우에도 그래픽 데이터는 데이터베이스 그래픽 데이터의 코드 페이지에 있는 것으로 가정됩니다. 데이터베이스 코드 페이지가 1바이트이면, 모든 데이터는 데이터베이스 코드 페이지에 있는 것으로 가정됩니다. **경고:** 1바이트 데이터베이스로 로드될 때 그래픽 데이터가 손상됩니다.
Present	Present	문자 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. 그래픽 데이터는 N의 그래픽 코드 페이지에 있는 것으로 가정됩니다. N이 1바이트 또는 2바이트 코드 페이지인 경우, 모든 데이터는 코드 페이지 N에 있는 것으로 가정됩니다. **경고:** N이 1바이트 코드 페이지이면 데이터베이스로 로드될 때 그래픽 데이터가 손상됩니다.

db2Load - 테이블에 데이터 로드

DB2 테이블에 데이터를 로드합니다. 서버에 있는 데이터는 파일, 커서, 테이프 또는 Named Pipe 양식이 될 수 있습니다. 리모트로 연결된 클라이언트에 있는 데이터는 완전한 파일 커서 또는 Named Pipe 양식이 될 수 있습니다. 임포트 유틸리티보다 더 빠른 방법이기는 하지만 로드 유틸리티는 계층 구조 레벨의 데이터 로드와 별칭으로 로드를 지원하지 않습니다.

권한 부여

다음 중 하나가 필요합니다.

- *dataaccess*
- 데이터베이스에서 로드 권한 및:
  - 로드 유틸리티가 INSERT 모드, TERMINATE 모드(이전의 로드 삽입 조작을 종료) 또는 RESTART 모드(이전의 로드 삽입 조작을 재시작)로 호출되는 경우 테이블에 대한 INSERT 특권
  - 로드 유틸리티를 REPLACE 모드, TERMINATE 모드(이전의 로드 바꾸기 조작을 종료) 또는 RESTART 모드(이전의 로드 바꾸기 조작을 재시작)로 호출하는 경우 테이블에 대한 INSERT 및 DELETE 특권
  - 예외 테이블에 대한 INSERT 특권(이러한 테이블이 로드 조작 중에 사용되는 경우).

FORCE 옵션을 지정한 경우에는 SYSADM 권한이 필요합니다.

주: 일반적으로 모든 로드 프로세스와 모든 DB2 서버 프로세스는 인스턴스 소유자가 소유합니다. 이 프로세스 전체에서는 인스턴스 소유자의 ID를 사용하여 필요한 파일에 액세스합니다. 따라서 인스턴스 소유자는 명령을 실행한 사용자에 상관 없이 입력 파일에 대해 읽기 액세스가 있어야 합니다.

필수 연결

데이터베이스. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다. Linux, UNIX 또는 Windows 클라이언트에서 Linux, UNIX 또는 Windows 데이터베이스 서버에 대한 유틸리티 액세스는 DB2 Connect 게이트웨이 또는 루프백 환경을 통해서가 아니라 엔진을 통해 직접 연결되어야 합니다.

인스턴스. 명시적 접속은 필요하지 않습니다. 데이터베이스에 대한 연결이 설정되면 로컬 인스턴스에 대한 내재적 접속이 시도됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Load (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2LoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2PartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2LoadStruct;

typedef SQL_STRUCTURE db2LoadUserExit
{
   db2Char iSourceUserExitCmd;
   struct db2Char *piInputStream;
   struct db2Char *piInputFileName;
   struct db2Char *piOutputFileName;
   db2Uint16 *piEnableParallelism;
} db2LoadUserExit;

typedef SQL_STRUCTURE db2LoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   db2Uint16 iSetIntegrityPending;
   struct db2LoadUserExit *piSourceUserExit;
} db2LoadIn;

typedef SQL_STRUCTURE db2LoadOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsLoaded;
```

```
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsDeleted;
   db2Uint64 oRowsCommitted;
} db2LoadOut;

typedef SQL_STRUCTURE db2PartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   SQL_PDB_NODE_TYPE *piRunStatDBPartNum;
} db2PartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint16 iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
   db2Uint16 iPortMin;
   db2Uint16 iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
   db2Uint64 oRowsRdPartAgents;
   db2Uint64 oRowsRejPartAgents;
   db2Uint64 oRowsPartitioned;
   struct db2LoadAgentInfo *poAgentInfoList;
   db2Uint32 iMaxAgentInfoEntries;
   db2Uint32 oNumAgentInfoEntries;
} db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
{
   db2int32 oSqlcode;
   db2Uint32 oTableState;
   SQL_PDB_NODE_TYPE oNodeNum;
   db2Uint16 oAgentType;
} db2LoadAgentInfo;

SQL_API_RC SQL_API_FN
  db2gLoad (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
```

```
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2gLoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2gPartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   db2Uint16 iFileTypeLen;
   db2Uint16 iLocalMsgFileLen;
   db2Uint16 iTempFilesPathLen;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2gLoadStruct;

typedef SQL_STRUCTURE db2gLoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 iUseTablespaceLen;
   db2Uint16 iSetIntegrityPending;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   struct db2LoadUserExit *piSourceUserExit;
} db2gLoadIn;

typedef SQL_STRUCTURE db2gPartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   void *piReserved1;
   db2Uint16 iHostnameLen;
   db2Uint16 iFileTransferLen;
   db2Uint16 iPartFileLocLen;
   db2Uint16 iMapFileInputLen;
```

```
   db2Uint16 iMapFileOutputLen;
   db2Uint16 iDistfileLen;
} db2gPartLoadIn;


/* Definitions for iUsing value of db2DMUXmlValidate structure               */
#define DB2DMU_XMLVAL_XDS              1         /* Use XDS                  */
#define DB2DMU_XMLVAL_SCHEMA           2         /* Use a specified schema   */
#define DB2DMU_XMLVAL_SCHEMALOC_HINTS  3         /* Use schemaLocation hints */
#define DB2DMU_XMLVAL_ORIGSCHEMA       4         /* Use schema that document was
                                                    originally validated against
                                                    (load from cursor only) */
```

db2Load API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2LoadStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LoadStruct 데이터 구조 매개변수

piSourceList

입력. 소스 파일, 디바이스, 벤더, 파이프 또는 SQL문 목록을 제공하는 데 사용된 sqlu_media_list 구조의 포인터.

이 구조에서 제공되는 정보는 media_type 필드 값에 따라 달라집니다. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_SQL_STMT

media_type 필드가 이 값으로 설정된 경우에 호출자는 대상 필드의 pStatement 필드를 사용하여 SQL 쿼리를 제공합니다. pStatement 필드는 sqlu_statement_entry 유형입니다. 로드 유틸리티가 로드당 한 개의 SQL 쿼리만 승인하기 때문에 세션 필드의 값은 1로 설정되어야 합니다.

SQLU_SERVER_LOCATION

media_type 필드가 이 값으로 설정된 경우에 호출자는 sqlu_location_entry 구조를 통해 정보를 제공합니다. 세션 필드는 제공된 sqlu_location_entry 구조 수를 나타냅니다. 이는 파일, 디바이스 및 Named Pipes에 대해 사용됩니다.

SQLU_CLIENT_LOCATION

media_type 필드가 이 값으로 설정된 경우에 호출자는 sqlu_location_entry 구조를 통해 정보를 제공합니다. 세션 필드는 제공된 sqlu_location_entry 구조 수를 나타냅니다. 이는 완전한 파일 및

Named Pipes에 대해 사용됩니다. 이 media_type은 리모트로 연결된 클라이언트를 통해 API가 호출 중인 경우에만 유효합니다.

SQLU_TSM_MEDIA

media_type 필드를 이 값으로 설정하면 파일 이름이 로드되는 데이터의 고유 ID인 경우에 sqlu_vendor 구조가 사용됩니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 TSM 세션 수를 나타냅니다. 로드 유틸리티는 다른 시퀀스 수로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

SQLU_OTHER_MEDIA

media_type 필드가 이 값으로 설정되면 sqlu_vendor 구조가 사용되며 이 경우 shr_lib는 공유 라이브러리 이름이고 filename은 로드되는 데이터의 고유 ID입니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 기타 벤더 세션 수입니다. 로드 유틸리티는 다른 시퀀스 수로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

SQLU_REMOTEFETCH

media_type 필드가 이 값으로 설정된 경우에 호출자는 sqlu_remotefetch_entry 구조를 통해 정보를 제공합니다. 세션 필드의 값은 1로 설정되어야 합니다.

piLobPathList

입력. sqlu_media_list 구조의 포인터. IXF, ASC 및 DEL 파일 유형의 경우 로드하려는 각 LOB 파일의 위치를 식별하는 완전한 경로 또는 디바이스 목록. 파일 이름은 IXF, ASC 또는 DEL 파일에 있으며 제공된 경로에 추가됩니다.

이 구조에서 제공되는 정보는 media_type 필드 값에 따라 달라집니다. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA

이 값으로 설정하면 호출자는 sqlu_media_entry 구조를 통해 정보를 제공합니다. 세션 필드는 제공된 sqlu_media_entry 구조 수를 나타냅니다.

SQLU_TSM_MEDIA

이 값으로 설정되면 sqlu_vendor 구조가 사용되며 이 경우 filename은 로드되는 데이터의 고유 ID입니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 TSM 세션 수를 나타냅니다. 로드 유틸리티는 다른 시퀀스 수로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

SQLU_OTHER_MEDIA

이 값으로 설정되면 sqlu_vendor 구조가 사용되며 이 경우 shr_lib는 공유 라이브러리 이름이고 filename은 로드되는 데이터의 고유 ID입니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 기타 벤더 세션 수입니다. 로드 유틸리티는 다른 시퀀스 수로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

piDataDescriptor

입력. 외부 파일에서 로드하기 위해 선택한 컬럼에 대한 정보를 포함하는 sqldcol 구조의 포인터.

piFileType 매개변수를 SQL_ASC로 설정한 경우 이 구조의 dcolmeth 필드를 SQL_METH_L로 설정해야 합니다. 사용자는 로드하려는 각 컬럼의 시작 및 종료 위치를 지정합니다.

파일 유형이 SQL_DEL인 경우 dcolmeth는 SQL_METH_P 또는 SQL_METH_D일 수 있습니다. SQL_METH_P인 경우 사용자는 소스 컬럼 위치를 제공해야 합니다. SQL_METH_D인 경우 파일의 첫 번째 컬럼은 테이블의 첫 번째 컬럼에 로드되는 방식입니다.

파일 유형이 SQL_IXF인 경우 dcolmeth는 SQL_METH_P, SQL_METH_D 또는 SQL_METH_N 중 하나일 수 있습니다. DEL 파일의 규칙이 여기에 적용되며 SQL_METH_N이 파일 컬럼 이름이 sqldcol 구조에서 제공되도록 표시된 경우는 제외입니다.

piActionString

더 이상 사용되지 않으며 piLongActionString으로 교체됩니다.

piLongActionString

입력. 테이블에 영향을 주는 조치를 지정하는 문자 배열이 뒤에 오는 4바이트 길이의 필드를 포함하는 *sqllob* 구조의 포인터.

문자 배열은 다음과 같은 양식입니다.

```
"INSERT|REPLACE KEEPDICTIONARY|REPLACE RESETDICTIONARY|RESTART|TERMINATE
INTO tbname [(column_list)]
[FOR EXCEPTION e_tbname]"
```

INSERT

기존 테이블 데이터를 변경하지 않고 로드된 데이터를 테이블에 추가합니다.

REPLACE

테이블에서 기존 데이터를 모두 삭제하고 로드된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다.

RESTART

이전에 인터럽트된 로드 조작을 재시작합니다. 로드 조작은 로드, 빌드 또는 삭제 단계의 마지막 일관성 지점에서 자동으로 계속됩니다.

TERMINATE

일관성 지점이 패스된 경우라도 이전에 인터럽트된 로드 조작을 종료하고 조작이 시작된 특정 시점으로 롤백합니다. 조작에 관계된 테이블 스페이스의 상태가 정상으로 리턴되고 모든 테이블 오브젝트는 일관성을 갖게 됩니다(인덱스 오브젝트가 유효하지 않은 것으로 표시되는 경우, 인덱스 재빌드가 다음 액세스에서 자동으로 발생). 테이블이 있는 테이블 스페이스가 로드 보류 상태가 아닌 경우 이 옵션은 테이블 스페이스 상태에 영향을 주지 않습니다.

로드 종료 옵션은 테이블 스페이스에서 백업 보류 상태를 제거하지 않습니다.

tbname

데이터가 로드되는 테이블 이름. 테이블은 시스템 테이블, 선언된 임시 테이블 또는 작성된 임시 테이블일 수 없습니다. 별명 또는 완전하거나 규정에 맞지 않는 테이블 이름을 지정할 수 있습니다. 규정된 테이블 이름의 양식은 schema.tablename입니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

(column_list)

데이터가 삽입되는 테이블 컬럼 이름 목록. 컬럼 이름은 쉼표로 구분해야 합니다. 이름에 공백이나 소문자가 포함된 경우에는 인용 부호로 묶어야 합니다.

FOR EXCEPTION e_tbname

오류 행이 복사될 예외 테이블을 지정합니다. 예외 테이블을 사용하여 고유 인덱스 규칙, 범위 제한조건 및 보안 규정을 위반하는 행 사본을 저장합니다.

NORANGEEXC

행이 범위 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

NOUNIQUEEXC

행이 고유 제한조건 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

piFileType

입력. 입력 데이터 소스 형식을 나타내는 문자열. 지원되는 외부 형식(sqlutil에 정의)은 다음과 같습니다.

SQL_ASC

컬럼 식별자가 없는 ASCII.

SQL_DEL

컬럼 식별자가 있는 ASCII, dBase, BASIC 및 IBM Personal Decision Series 프로그램과 교환에 사용되는 다수의 데이터베이스 관리 프로그램 및 파일 관리자.

SQL_IXF

통합 교환 형식의 PC 버전으로 테이블에서 데이터를 익스포트할 때 이후에 동일한 테이블이나 다른 데이터베이스 관리 프로그램 테이블에 로드할 수 있기 때문에 자주 사용되는 방법입니다.

SQL_CURSOR

SQL 쿼리. piSourceList 매개변수로 전달되는 sqlu_media_list 구조는 SQLU_SQL_STMT 또는 SQLU_REMOTEFETCH 유형으로 SQL 쿼리나 테이블 이름입니다.

piFileTypeMod

입력. sqlchar 구조의 포인터로 뒤에 하나 이상의 처리 옵션을 지정하는 문자 배열이 옵니다. 이 포인터가 NULL이거나 지시된 구조에 영(0) 문자가 있는 경우 이 조치는 디폴트 스펙 선택으로 해석됩니다.

지원되는 모든 파일 유형에 모든 옵션을 사용할 수 있는 것은 아닙니다. 관련 링크인 ″로드 유틸리티의 파일 유형 수정자″를 참조하십시오.

piLocalMsgFileName

입력. 출력 메시지가 기록되는 로컬 파일 이름이 포함된 문자열.

piTempFilesPath

입력. 임시 파일을 위해 서버에서 사용되는 경로 이름이 포함된 문자열. 임시 파일은 메시지, 일관성 지점을 저장하고 단계 정보를 삭제하기 위해 작성됩니다.

piVendorSortWorkPaths

입력. 벤더 정렬 작업 디렉토리를 지정하는 sqlu_media_list 구조의 포인터

piCopyTargetList

입력. 사본 이미지가 작성되는 경우 이를 위한 목표 경로, 디바이스 또는 공유 라이브러리 목록을 제공하는 데 사용되는 sqlu_media_list 구조의 포인터.

이 구조에서 제공되는 값은 media_type 필드 값에 따라 다릅니다. 이 매개변수의 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA

사본이 로컬 미디어에 작성되는 경우 media_type을 이 값으로 설정하

고 sqlu_media_entry 구조에 목표에 대한 정보를 제공하십시오. 세션 필드는 제공된 sqlu_media_entry 구조 수를 지정합니다.

SQLU_TSM_MEDIA
사본이 TSM에 작성되는 경우 이 값을 사용하십시오. 추가 정보는 필요하지 않습니다.

SQLU_OTHER_MEDIA
벤더 제품을 사용하려는 경우 이 값을 사용하여 sqlu_vendor 구조를 통해 추가 정보를 제공하십시오. 이 구조의 shr_lib 필드를 벤더 제품의 공유 라이브러리로 설정하십시오. 세션 값에 상관 없이 한 개의 sqlu_vendor 항목만 제공하십시오. 세션 필드는 제공된 sqlu_media_entry 구조 수를 지정합니다. 로드 유틸리티는 다른 시퀀스 수로 세션을 시작하지만 하나의 sqlu_vendor 항목에서 제공된 동일한 데이터를 사용합니다.

piNullIndicators
입력. ASC 파일에만 해당. 컬럼 데이터가 널(NULL) 입력 가능한지 여부를 표시하는 정수 배열. 이 배열 요소와 데이터 파일에서 로드 중인 컬럼은 1대1 대응 방식입니다. 즉, 요소 수는 piDataDescriptor 매개변수의 dcolnum 필드와 동일해야 합니다. 배열의 각 요소에는 널(NULL) 표시기 필드로 사용되는 데이터 파일의 위치를 식별하는 번호 또는 테이블 컬럼이 널(NULL) 입력 가능한 것을 표시하는 영(0)이 포함됩니다. 요소가 영(0)이 아닌 경우 데이터 파일의 식별된 위치에는 Y 또는 N이 포함됩니다. Y는 테이블 컬럼 데이터가 NULL임을 표시하고 N은 테이블 컬럼 데이터가 NULL이 아닌 것을 표시합니다.

piLoadInfoIn
입력. db2LoadIn 구조의 포인터

poLoadInfoOut
출력. db2LoadOut 구조의 포인터

piPartLoadInfoIn
입력. db2PartLoadIn 구조의 포인터

poPartLoadInfoOut
출력. db2PartLoadOut 구조의 포인터

iCallerAction
입력. 호출자가 요청한 조치. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_INITIAL
초기 호출. 이 값(또는 SQLU_NOINTERRUPT)은 API의 첫 번째 호출에 사용되어야 합니다.

SQLU_NOINTERRUPT

초기 호출. 처리를 일시중단하지 않습니다. 이 값(또는 SQLU_INITIAL)은 API의 첫 번째 호출에 사용되어야 합니다.

초기 호출이나 이후에 임의의 후속 호출이 리턴되고 요청된 로드 조작을 완료하기 전에 일부 조치를 수행하는 응용프로그램을 호출해야 하는 경우 호출자 조치는 다음 중 하나로 설정해야 합니다.

SQLU_CONTINUE

처리를 계속합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 완료되고 유틸리티는 초기 요청 처리를 계속하도록 지정합니다.

SQLU_TERMINATE

처리를 종료합니다. 로드 유틸리티가 너무 빨리 종료되어 로드 중인 테이블 스페이스는 LOAD_PENDING 상태가 됩니다. 이 옵션은 데이터의 추가 처리가 완료되지 않아야 하는 경우에 지정해야 합니다.

SQLU_ABORT

처리를 종료합니다. 로드 유틸리티가 너무 빨리 종료되어 로드 중인 테이블 스페이스는 LOAD_PENDING 상태가 됩니다. 이 옵션은 데이터의 추가 처리가 완료되지 않아야 하는 경우에 지정해야 합니다.

SQLU_RESTART

처리를 재시작합니다.

SQLU_DEVICE_TERMINATE

디바이스 하나를 종료합니다. 유틸리티가 디바이스에서 데이터 읽기를 중단하지만 데이터의 추가 처리는 완료해야 하는 경우에 이 옵션을 지정해야 합니다.

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 xml 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터.

db2LoadUserExit 데이터 구조 매개변수

iSourceUserExitCmd

입력. 데이터를 유틸리티에 공급하는 데 사용되는 실행 파일의 완전한 이름. 보안상의 이유로 실행 파일은 서버의 `sqllib/bin` 디렉토리에 있어야 합니다. 이 매개변수는 piSourceUserExit 구조가 NULL이 아닌 경우 필수입니다.

piInputStream, piInputFileName, piOutputFileName 및 piEnableParallelism 필드는 선택적입니다.

piInputStream

입력. STDIN을 통해 userexit 응용프로그램에 직접 전달되는 일반 바이트스트림. 이 바이트스트림에 포함되는 데이터와 해당 형식을 완벽하게 제어해야 합니다. 로드 유틸리티는 서버에서 이 바이트스트림을 단순하게 처리하고 프로세스의 STDIN을 공급하여 userexit 응용프로그램에 전달합니다(코드 페이지를 변환하거나 바이트스트림을 수정하지는 않음). userexit 응용프로그램은 STDIN에서 인수를 읽고 데이터를 의도한 대로 사용합니다.

이 기능의 중요한 한 가지 속성은 민감한 정보(사용자 ID/암호)를 숨기는 기능입니다.

piInputFileName

입력. 프로세스의 STDIN을 공급하여 컨텐츠가 userexit 응용프로그램에 전달되는 완전한 클라이언트 측 파일의 이름이 포함됩니다.

piOutputFileName

입력. 서버 측 파일의 완전한 이름. userexit 응용프로그램을 실행하는 프로세스의 STDOUT 및 STDERR 스트림이 이 파일로 스트림됩니다. piEnableParallelism이 참인 경우 다중 파일이 작성되고(userexit 인스턴스당 1개) 각 파일 이름에는 3자리의 숫자 노드 번호 값이 추가됩니다(예: `<filename>.000`).

piEnableParallelism

입력. 유틸리티가 userexit 응용프로그램 호출을 병렬 처리하도록 표시하는 플래그.

db2LoadIn 데이터 구조 매개변수

iRowcount

입력. 로드할 실제 레코드 수. 사용자가 파일의 첫 번째 rowcnt 행만 로드하도록 허용합니다.

iRestartcount

입력. 나중에 사용하도록 예약됩니다.

piUseTablespace

입력. 인덱스가 재빌드되는 경우, 인덱스의 쉐도우 사본이 테이블 스페이스 iUseTablespaceName에 빌드되고 로드 종료 시에 원래 테이블 스페이스에 복사됩니다. 이 옵션에는 시스템 임시 테이블 스페이스만 사용할 수 있습니다. 테이블 스페이스를 지정하지 않을 경우 음영 인덱스는 인덱스 오브젝트와 동일한 테이블 스페이스에 작성됩니다.

음영 사본이 인덱스 오브젝트와 동일한 테이블 스페이스에 작성된 경우, 이전 인덱스 오브젝트에 대한 음영 인덱스 오브젝트의 복사는 순간적입니다. 음영 사본이 인덱스 오브젝트와 다른 테이블 스페이스에 있을 경우 실제 복사가 수행

됩니다. 상당한 I/O 및 시간이 포함될 수 있습니다. 복사는 테이블이 로드 종료 시에 오프라인 상태에서 수행됩니다.

이 필드는 iAccessLevel이 SQLU_ALLOW_NO_ACCESS인 경우 무시됩니다.

사용자가 INDEXING MODE REBUILD 또는 INDEXING MODE AUTOSELECT를 지정하지 않을 경우 이 옵션은 무시됩니다. IDEXING MODE AUTOSELECT가 선택되고 로드가 인덱스를 점차적으로 갱신하도록 선택할 경우에도 이 옵션이 무시됩니다.

iSavecount

일관성 지점을 작성하기 전에 로드되는 레코드 수. 이 값은 쪽 수로 변환되며 Extent *크기*의 간격으로 반올림됩니다. 메시지는 각각의 일관성 지점에서 발행되므로 db2LoadQuery - 로드 쿼리를 사용하여 로드 조작을 모니터하는 경우 이 옵션을 선택해야 합니다. savecount 값이 충분히 높지 않으면 각 일관성 지점에서 수행된 활동의 동기화가 성능에 영향을 줍니다.

디폴트값은 0으로, 필요하지 않으면 일관성 지점을 설정하지 않음을 의미합니다.

iDataBufferSize

유틸리티에서 데이터 전송에 필요한 버퍼 스페이스로 사용할 4KB 페이지의 수 (병렬 처리 수준과 무관함). 지정된 값이 알고리즘의 최소 값보다 작으면, 필요한 최소값이 사용되고 경고는 리턴되지 않습니다.

이 메모리는 유틸리티 힙에서 바로 할당되며 *크기*는 util_heap_sz 데이터베이스 구성 매개변수로 수정할 수 있습니다.

값이 지정되지 않으면 런타임시 유틸리티에 의해 적절한 디폴트값이 계산됩니다. 디폴트값은 테이블의 일부 등록 정보 뿐 아니라 로더의 인스턴스화 시간에 유틸리티 힙에서 사용 가능한 여유 공간의 백분율을 기초로 합니다.

iSortBufferSize

입력. 이 옵션은 로드 조작 중 SORTHEAP 데이터베이스 구성 매개변수를 겹쳐쓰는 값을 지정합니다. 이 옵션은 인덱스와 함께 테이블을 로드하는 경우와 iIndexingMode 매개변수가 SQLU_INX_DEFERRED로 지정되지 않은 경우에만 의미가 있습니다. 지정된 값은 SORTHEAP의 값을 초과할 수 없습니다. 이 매개변수는 SORTHEAP의 값을 변경하지 않은 채로 LOAD에서 사용되는 정렬 메모리를 조절하는 데 유용하며 일반 쿼리 처리에도 영향을 줍니다.

iWarningcount

입력. warningcnt 경고 후에 로드 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 로드 파일이나 목표 테이블이 잘못 지정된 경우, 로드

유틸리티가 로드를 시도하는 각 행에 대해 경고를 생성하므로 로드에 실패하게 됩니다. warningcnt가 0이거나 이 옵션을 지정하지 않으면 로드 조작이 발생된 경고 수에 상관 없이 계속됩니다.

경고 임계값이 초과되어 로드 조작이 중지된 경우 다른 로드 조작은 RESTART 모드로 시작할 수 있습니다. 로드 조작은 마지막 일관성 지점에서 자동으로 계속됩니다. 또는 다른 로드 조작을 REPLACE 모드로 초기화하여 입력 파일의 처음부터 시작하게 할 수 있습니다.

iHoldQuiesce

입력. 유틸리티가 로드 후에 테이블을 Quiesce 독점 상태로 유지하는 경우 값이 참으로 설정되고 이 외의 경우에는 값이 거짓으로 설정되는 플래그.

iCpuParallelism

입력. 테이블 오브젝트를 빌드할 때 레코드를 구문 분석, 변환 및 포맷팅하기 위해 로드 유틸리티가 작성하는 프로세스 또는 스레드의 수. 이 매개변수는 파티션 내 병렬 처리를 이용하도록 설계되었습니다. 소스 데이터에 있는 기록 순서가 보존되므로 사전에 정렬된 데이터를 로드할 때 특히 유용합니다. 이 매개변수의 값이 0인 경우 로드 유틸리티는 런타임에서 인텔리전트한 디폴트값을 사용합니다. 참고: 이 매개변수는 LOB 또는 LONG VARCHAR 필드 중 하나가 있는 테이블과 함께 사용되면 시스템 CPU 수나 사용자가 지정한 값과 관계없이 값은 1이 됩니다.

iDiskParallelism

입력. 로드 유틸리티가 데이터를 테이블 스페이스 컨테이너에 기록하기 위해 작성하는 프로세스나 스레드 수. 값이 지정되지 않으면 유틸리티가 테이블 스페이스 컨테이너의 수 및 테이블의 등록 정보를 기반으로 하여 적절한 디폴트값을 선택합니다.

iNonrecoverable

입력. 로드 트랜잭션이 복구 불가능으로 표시되어 후속 롤 포워드 조치로 복구할 수 없는 경우에는 SQLU_NON_RECOVERABLE_LOAD로 설정하십시오. 롤 포워드 유틸리티는 트랜잭션을 건너뛰고 데이터를 로드할 테이블을 ″유효하지 않은″ 항목으로 표시합니다. 또한 이 유틸리티는 해당 테이블에 대한 후속 트랜잭션을 무시합니다. 롤 포워드가 완료되면 이런 테이블은 삭제할 수만 있습니다. 이 옵션을 사용하면 로드 조작 후에 테이블 스페이스가 백업 보류 상태가 되지 않으며 로드 조작 중에 로드된 데이터의 사본을 작성할 필요가 없습니다. 로드 트랜잭션이 복구 가능으로 표시된 경우 SQLU_RECOVERABLE_LOAD로 설정하십시오.

iIndexingMode

입력. 인덱스 모드를 지정합니다. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_INX_AUTOSELECT
LOAD는 REBUILD 및 INCREMENTAL 인덱스 모드에서 선택합니다.

SQLU_INX_REBUILD
테이블 인덱스를 재빌드합니다.

SQLU_INX_INCREMENTAL
기존 인덱스를 확장합니다.

SQLU_INX_DEFERRED
테이블 인덱스를 갱신하지 않습니다.

iAccessLevel
입력. 액세스 등급을 지정합니다. 가능한 값은 다음과 같습니다.

SQLU_ALLOW_NO_ACCESS
로드가 테이블을 배타적으로 잠그도록 지정합니다.

SQLU_ALLOW_READ_ACCESS
테이블의 원래 데이터(델타가 아닌 부분)가 로드 진행 중에도 판독기에 표시되도록 지정합니다. 이 옵션은 로드 삽입과 같은 로드 추가에만 유효하며 로드 교체에 대해서는 무시됩니다.

iLockWithForce
입력. 부울 플래그. 참으로 설정되면 로드는 테이블 잠금을 즉시 확보하도록 다른 응용프로그램을 강제 실행합니다. 이 옵션을 사용하려면 FORCE APPLICATIONS 명령(SYSADM 또는 SYSCTRL)과 동일한 권한이 필요합니다.

SQLU_ALLOW_NO_ACCESS 로드는 로드 조작 시작 시에 충돌하는 응용프로그램을 강제 실행할 수도 있습니다. 로드 시작 시에 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

SQLU_ALLOW_READ_ACCESS 로드는 로드 조작 시작 또는 종료 시에 충돌하는 응용프로그램을 강제 실행할 수도 있습니다. 로드 시작 시에 로드 유틸리티는 테이블을 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다. 로드 종료 시에 로드 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

iCheckPending
이 매개변수는 버전 9.1부터 더 이상 사용되지 않습니다. iSetIntegrityPending 매개변수를 대신 사용하십시오.

iRestartphase
입력. 예약됨. 유효한 값은 단일 공백 문자 ' '입니다.

iStatsOpt

입력. 수집할 통계의 세분화도. 가능한 값은 다음과 같습니다.

SQLU_STATS_NONE

수집할 통계가 없습니다.

SQLU_STATS_USE_PROFILE

현재 테이블에 대해 정의된 프로파일을 사용하여 통계가 수집됩니다. 이 프로파일은 RUNSTATS 명령을 사용하여 작성해야 합니다. 현재 테이블에 대한 프로파일이 없는 경우 경고가 리턴되고 통계가 수집되지 않습니다.

iSetIntegrityPending

입력. 테이블을 무결성 설정 보류 상태로 지정합니다.

SQLU_SI_PENDING_CASCADE_IMMEDIATE 값이 지정된 경우 무결성 설정 보류 상태가 즉시 모든 종속 및 하위 테이블에 중첩됩니다.

SQLU_SI_PENDING_CASCADE_DEFERRED 값이 지정된 경우 종속 테이블의 무결성 설정 보류 상태 중첩이 목표 테이블의 무결성 위반이 점검될 때까지 지연됩니다. SQLU_SI_PENDING_CASCADE_DEFERRED는 옵션을 지정하지 않는 경우 디폴트값입니다.

piSourceUserExit

입력. db2LoadUserExit 구조의 포인터

piXmlParse

입력. XML 문서에 대해 수행해야 하는 구문 분석 유형. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

DB2DMU_XMLPARSE_PRESERVE_WS

공백은 유지해야 합니다.

DB2DMU_XMLPARSE_STRIP_WS

공백은 스트립되어야 합니다.

piXmlValidate

입력. db2DMUXmlValidate 구조의 포인터. XML 스키마 유효성 확인이 XML 문서에 대해 수행되어야 하는 것을 나타냅니다.

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                        iUsing;      /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2LoadOut 데이터 구조 매개변수

oRowsRead

출력. 로드 조작에서 읽은 레코드 수

oRowsSkipped

출력. 로드 조작 시작 전에 건너뛴 레코드 수

oRowsLoaded

출력. 목표 테이블로 로드된 행 수.

oRowsRejected

출력. 로드할 수 없던 레코드 수

oRowsDeleted

출력. 삭제된 중복 행 수.

oRowsCommitted

출력. 처리된 총 레코드 수: 데이터베이스에 성공적으로 로드되어 커미트된 레코드 수 더하기 건너뛰고 거부된 레코드 수

db2PartLoadIn 데이터 구조 매개변수

piHostname

입력. iFileTransferCmd 매개변수의 호스트 이름. NULL인 경우 호스트 이름의 디폴트는 ″nohost″입니다. 이 매개변수는 더 이상 사용되지 않습니다.

piFileTransferCmd

입력. 파일 전송 명령 매개변수. 필요하지 않는 경우 NULL로 설정해야 합니다. 이 매개변수는 더 이상 사용되지 않습니다. piSourceUserExit 매개변수를 대신 사용하십시오.

piPartFileLocation

입력. PARTITION_ONLY, LOAD_ONLY, LOAD_ONLY_VERIFY_PART 모드에서 이 매개변수를 사용하여 파티션된 파일의 위치를 지정할 수 있습니다. 이 위치는 piOutputNodes 옵션으로 지정한 각 데이터베이스 파티션에 있어야 합니다.

SQL_CURSOR 파일 유형의 경우 이 매개변수는 NULL일 수 없으며 위치는 경로가 아니라 완전한 파일 이름입니다. PARTITION_ONLY 모드에서 각 출력 데이터베이스 파티션에서 작성된 파티션된 파일의 완전한 기본 파일 이름이거나 LOAD_ONLY 모드에서 각 데이터베이스 파티션에서 읽은 파일의 위치입니다. PARTITION_ONLY 모드의 경우 목표 테이블에 LOB 컬럼이 있는 경우 다중 파일이 지정한 기본 이름으로 작성될 수도 있습니다. SQL_CURSOR 이외의 파일 유형의 경우 이 매개변수 값이 NULL인 경우 현재 디렉토리가 디폴트입니다.

piOutputNodes

입력. 로드 출력 데이터베이스 파티션 목록. NULL은 목표 테이블의 모든 노드가 정의되는 것을 나타냅니다.

piPartitioningNodes

입력. 파티셔닝 노드의 목록. NULL은 디폴트를 표시합니다.

piMode

입력. 파티션된 데이터베이스의 로드 모드를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2LOAD_PARTITION_AND_LOAD

데이터가 해당 데이터베이스 파티션에서 동시에 분산(병렬 방식일 수 있음) 및 로드됩니다.

- DB2LOAD_PARTITION_ONLY

데이터가 분산되고(병렬 방식일 수 있음) 로드하는 각 데이터베이스 파티션에서 지정된 위치의 파일에 작성됩니다. SQL_CURSOR 이외의 파일 유형의 경우 각 데이터베이스 파티션의 출력 파일 이름은 filename.xxx이며 여기서 filename은 piSourceList로 지정한 첫 번째 입력 파일 이름이며 xxx는 데이터베이스 파티션 번호입니다. SQL_CURSOR 파일 유형의 경우 각 데이터베이스 파티션에 있는 출력 파일은 piPartFileLocation 매개변수로 판별합니다. 각 데이터베이스 파티션에서 데이터베이스 파티션 위치를 지정하는 방법에 대한 정보는 piPartFileLocation 매개변수를 참조하십시오.

주: CLI LOAD에는 이 모드를 사용할 수 없습니다.

DB2LOAD_LOAD_ONLY

데이터는 이미 분산되었다고 가정합니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. SQL_CURSOR 이외의 파일 유형의 경우 각 데이터베이스 파티션의 입력 파일 이름은 filename.xxx 양식이어야 하며 여기서 filename은 piSourceList로 지정한 첫 번째 파일 이름이고 xxx는 13자리의 데이터베이스 파티션 번호입니다. SQL_CURSOR 파일 유형의 경우 각 데이터베이스 파티션에 있는 입력 파일은 piPartFileLocation 매개변수로 판별합니다. 각 데이터베이스 파티션에서 데이터베이스 파티션 위치를 지정하는 방법에 대한 정보는 piPartFileLocation 매개변수를 참조하십시오.

주: 이 모드는 리모트 클라이언트에 있는 데이터 파일 로드 시 또는 CLI LOAD에는 사용할 수 없습니다.

DB2LOAD_LOAD_ONLY_VERIFY_PART

데이터가 이미 분산된 것으로 간주되지만 데이터 파일에는 데이터베이스 파티션 헤더가 포함되지 않습니다. 분산 프로세스는 건너뛰고 데이

터는 해당 데이터베이스 파티션에 동시에 로드됩니다. 로드 조작 중에 각 행은 올바른 데이터베이스 파티션에 있는지 확인하기 위해 점검됩니다. 데이터베이스 파티션 위반을 포함하는 행은 덤프 파일 파일 유형 수정자가 지정된 경우 덤프 파일에 배치됩니다. 그렇지 않으면 행을 버립니다. 데이터베이스 파티션 위반이 특정 로드 데이터베이스 파티션에 있는 경우 해당 데이터베이스 파티션에 대한 로드 메시지에 한 개의 경고가 기록됩니다. 각 데이터베이스 파티션의 입력 파일 이름은 filename.xxx 양식이어야 하며 여기서 filename은 piSourceList로 지정한 첫 번째 파일 이름이고 xxx는 13자리의 데이터베이스 파티션 번호입니다.

주: 이 모드는 리모트 클라이언트에 있는 데이터 파일 로드 시 또는 CLI LOAD에는 사용할 수 없습니다.

DB2LOAD_ANALYZE
모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다.

piMaxNumPartAgents
입력. 초대 파티셔닝 에이전트 수. NULL 값은 25인 디폴트입니다.

piIsolatePartErrs
입력. 로드 조작이 각 데이터베이스 파티션에서 발생한 오류에 대응하는 방법을 나타냅니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOAD_SETUP_ERRS_ONLY
이 모드에서 데이터베이스 파티션 액세스 문제 또는 테이블 스페이스 또는 데이터베이스 파티션의 테이블 액세스 문제와 같이 설정 중에 데이터베이스 파티션에서 발생하는 오류로 인해 로드 조작이 실패한 데이터베이스 파티션에서 중지되지만 나머지 데이터베이스 파티션에서는 계속됩니다. 데이터가 로드되는 중에 데이터베이스 파티션에서 발생한 오류로 인해 전체 조작이 실패하고 각 데이터베이스 파티션의 마지막 일관성 시점으로 롤백됩니다.

DB2LOAD_LOAD_ERRS_ONLY
이 모드에서는 설정 중에 데이터베이스 파티션에서 발생한 오류로 인해 전체 로드 조작이 실패합니다. 데이터 로드 중에 오류가 발생하면 오류가 발생한 데이터베이스 파티션이 최종 일관성 지점으로 롤백됩니다. 실패가 발생하거나 모든 데이터가 로드될 때까지 나머지 데이터베이스 파티션에서 로드 조작이 계속됩니다. 모든 데이터가 로드된 데이터베이스 파티션에서 로드 조작 후에 데이터가 표시되지 않습니다. 기타 데이터베이스 파티션의 오류로 인해 트랜잭션이 중단됩니다. 모든 데이터베이스 파티션의 데이터는 로드 시작 조작이 수행될 때까지 표시되

지 않습니다. 이로 인해 새로 로드된 데이터는 로드 조작이 완료되는 데이터베이스 파티션에 표시되고 오류가 발생한 데이터베이스 파티션에서 로드 조작이 다시 시작됩니다.

주: 이 모드는 iAccessLevel이 SQLU_ALLOW_READ_ACCESS로 설정되고 사본 대상도 지정된 경우에는 사용할 수 없습니다.

DB2LOAD_SETUP_AND_LOAD_ERRS

이 모드에서는 설정이나 데이터 로드 중에 발생한 데이터베이스 파티션 오류로 인해 영향 받는 데이터베이스 파티션에서만 처리가 중지됩니다. DB2LOAD_LOAD_ERRS_ONLY 모드를 사용하면 데이터 로드 중에 데이터베이스 파티션 오류가 발생하지 않은 경우 모든 데이터베이스 파티션의 데이터는 로드 시작 조작이 수행될 때까지 표시되지 않습니다.

주: 이 모드는 iAccessLevel이 SQLU_ALLOW_READ_ACCESS로 설정되고 사본 대상도 지정된 경우에는 사용할 수 없습니다.

DB2LOAD_NO_ISOLATION

로드 조작 중의 오류로 인해 트랜잭션이 중단됩니다. 이 매개변수가 NULL인 경우 iAccessLevel이 SQLU_ALLOW_READ_ACCESS로 설정되고 사본 대상도 지정된 경우에는 디폴트가 DB2LOAD_NO_ISOLATION인 경우 DB2LOAD_LOAD_ERRS_ONLY를 디폴트로 사용합니다.

piStatusInterval

입력. 진행 상황 메시지를 생성하기 전에 로드할 데이터 수(메가바이트(MB))를 지정합니다. 유효한 값은 1 - 4000 사의의 정수입니다. NULL을 지정하면 디폴트값인 100이 사용됩니다.

piPortRange

입력. 내부 통신을 위한 TCP 포트. NULL인 경우 사용되는 포트 범위는 6000 - 6063입니다.

piCheckTruncation

입력. 로드가 입/출력(I/O) 시에 레코드 절단을 점검합니다. 유효한 값은 참 및 거짓입니다. NULL인 경우 디폴트는 거짓입니다.

piMapFileInput

입력. 분산 맵 입력 파일 이름. 모드가 ANALYZE가 아닌 경우 이 매개변수는 NULL로 설정해야 합니다. 모드가 ANALYZE인 경우 이 매개변수를 지정해야 합니다.

piMapFileOutput

입력. 분산 맵 출력 파일 이름. piMapFileInput의 규칙이 여기에도 적용됩니다.

piTrace

입력. 모든 데이터 변환 프로세스 및 해시 값 출력의 덤프를 검토해야 하는 경우 추적할 레코드 수를 지정합니다. NULL인 경우 레코드 수의 디폴트값은 0입니다.

piNewline

입력. RECLEN 파일 유형 수정자도 지정된 경우 ASC 데이터 레코드 끝에서 로드가 개행 문자를 강제로 점검하도록 합니다. 가능한 값은 참 및 거짓입니다. NULL인 경우 디폴트는 거짓입니다.

piDistfile

입력. 데이터베이스 파티션 분산 파일 이름. NULL을 지정하면 디폴트 값은 ″DISTFILE″입니다.

piOmitHeader

입력. 분산 맵 헤더가 DB2LOAD_PARTITION_ONLY 모드 사용 시에 데이터베이스 파티션 파일에 포함되지 않는 것을 나타냅니다. 가능한 값은 참 및 거짓입니다. NULL인 경우 디폴트는 거짓입니다.

piRunStatDBPartNum

통계를 수집하는 데이터베이스 파티션을 지정합니다. 디폴트값은 출력 데이터베이스 파티션 목록의 첫 번째 데이터베이스 파티션입니다.

db2LoadNodeList 데이터 구조 매개변수

piNodeList

입력. 노드 번호 배열.

iNumNodes

입력. piNodeList 배열의 노드 수. 0이 디폴트이며 목표 테이블이 정의된 모든 노드입니다.

db2LoadPortRange 데이터 구조 매개변수

iPortMin

입력. 낮은 포트 번호.

iPortMax

입력. 높은 포트 번호.

db2PartLoadOut 데이터 구조 매개변수

oRowsRdPartAgents

출력. 모든 파티셔닝 에이전트가 읽는 총 행 수.

oRowsRejPartAgents

출력. 모든 파티셔닝 에이전트가 거부하는 총 행 수.

oRowsPartitioned

출력. 모든 파티셔닝 에이전트로 파티션된 총 행 수.

poAgentInfoList

출력. 파티션된 데이터베이스의 로드 조작 중에 다음 로드 처리 엔티티가 사용될 수 있습니다. 로드 에이전트, 파티셔닝 에이전트, 프리파티셔닝 에이전트, 파일 전송 명령 에이전트 및 파일에 로드 에이전트(이들에 대해서는 데이터 이동 안내서에서 설명됨). poAgentInfoList 출력 매개변수는 로드 조작에 사용되는 각 로드 에이전트에 대한 호출자 정보를 리턴하는 데 사용됩니다. 목록의 각 항목에는 다음 정보가 포함됩니다.

oAgentType

항목이 설명하는 로드 에이전트 종류를 나타내는 태그.

oNodeNum

에이전트가 실행된 데이터베이스 파티션 번호.

oSqlcode

에이전트 처리 결과로 작성된 최종 sqlcode.

oTableState

에이전트가 실행된 데이터베이스 파티션에서 테이블의 최종 상태(로드 에이전트에만 관련).

API 호출 전에 이 목록에 대해 API 호출자가 메모리를 할당해야 합니다. 호출자는 iMaxAgentInfoEntries 매개변수에 메모리를 할당한 항목 수도 표시해야 합니다. 호출자가 poAgentInfoList를 NULL로 설정하거나 iMaxAgentInfoEntries를 0으로 설정한 경우 로드 에이전트에 대한 정보가 리턴되지 않습니다.

iMaxAgentInfoEntries

입력. 사용자가 poAgentInfoList에 대해 할당한 에이전트 정보 항목의 최대 수. 일반적으로 이 매개변수는 로드 조작에 사용된 데이터베이스 파티션 수의 3배로 설정하면 충분합니다.

oNumAgentInfoEntries

출력. 로드 조작으로 작성된 에이전트 정보 항목의 실제 수. 이 항목 수는 iMaxAgentInfoEntries가 oNumAgentInfoEntries 이상인 경우 poAgentInfoList 매개변수로 사용자에게 리턴됩니다. iMaxAgentInfoEntries가 oNumAgentInfoEntries보다 작은 경우 poAgentInfoList로 리턴된 항목 수는 iMaxAgentInfoEntries와 같습니다.

db2LoadAgentInfo 데이터 구조 매개변수

oSqlcode

출력. 에이전트 처리 결과로 작성된 최종 sqlcode.

oTableState

출력. 이 출력 매개변수는 로드 조작 후에 테이블의 가능한 모든 상태를 보고하는 데 사용되지는 않습니다. 오히려 호출자에게 로드 처리 중에 발생하는 일반적인 정보를 제공하기 위해 가능한 테이블 상태의 일부 서브세트만을 보고합니다. 이 값은 로드 에이전트에만 연관됩니다. 가능한 값은 다음과 같습니다.

DB2LOADQUERY_NORMAL

데이터베이스 파티션에서 로드가 완료되었으며 테이블이 LOAD IN PROGRESS(또는 LOAD PENDING) 상태를 벗어났음을 나타냅니다. 이 경우 테이블은 이후 제한조건 처리를 위해 SET INTEGRITY PENDING 상태일 수는 있지만 이는 정상적이기 때문에 보고되지는 않습니다.

DB2LOADQUERY_UNCHANGED

오류로 인해 로드 작업이 처리를 중단했지만 db2Load 호출 이전의 상태에 관계 없이 데이터베이스 파티션의 테이블 상태가 아직 변경되지 않았음을 나타냅니다. 이런 데이터베이스 파티션에서 로드 재시작을 수행하거나 조작을 종료할 필요는 없습니다.

DB2LOADQUERY_LOADPENDING

처리 중에 로드 작업이 중단되었지만 데이터베이스 파티션의 테이블이 LOAD PENDING 상태로 해당 데이터베이스 파티션의 로드 작업은 종료 또는 재시작해야 함을 나타냅니다.

oNodeNum

출력. 에이전트가 실행된 데이터베이스 파티션 번호.

oAgentType

출력. 에이전트 유형. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2LOAD_LOAD_AGENT
- DB2LOAD_PARTITIONING_AGENT
- DB2LOAD_PRE_PARTITIONING_AGENT
- DB2LOAD_FILE_TRANSFER_AGENT
- DB2LOAD_LOAD_TO_FILE_AGENT

db2gLoadStruct 데이터 구조 특정 매개변수

iFileTypeLen

입력. iFileType 매개변수 길이를 바이트 단위로 지정합니다.

iLocalMsgFileLen

입력. iLocalMsgFileName 매개변수 길이를 바이트 단위로 지정합니다.

iTempFilesPathLen

입력. iTempFilesPath 매개변수 길이를 바이트 단위로 지정합니다.

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 xml 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터.

db2gLoadIn 데이터 구조 특정 매개변수

iUseTablespaceLen

입력. piUseTablespace 매개변수의 길이(바이트).

piXmlParse

입력. XML 문서에 대해 수행해야 하는 구문 분석 유형. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

DB2DMU_XMLPARSE_PRESERVE_WS

공백은 유지해야 합니다.

DB2DMU_XMLPARSE_STRIP_WS

공백은 스트립되어야 합니다.

piXmlValidate

입력. db2DMUXmlValidate 구조의 포인터. XML 스키마 유효성 확인이 XML 문서에 대해 수행되어야 하는 것을 나타냅니다.

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                       iUsing;       /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2gPartLoadIn 데이터 구조 특정 매개변수

piReserved1

나중에 사용하도록 예약됩니다.

iHostnameLen

입력. piHostname 매개변수의 길이(바이트).

iFileTransferLen

입력. piFileTransferCmd 매개변수의 길이(바이트).

iPartFileLocLen

입력. piPartFileLocation 매개변수의 길이(바이트).

iMapFileInputLen

입력. piMapFileInput 매개변수의 길이(바이트).

iMapFileOutputLen

입력. piMapFileOutput 매개변수의 길이(바이트).

iDistfileLen

입력. piDistfile 매개변수의 길이(바이트).

사용 시 참고사항

데이터는 입력 파일에 나타나는 시퀀스로 로드됩니다. 특정 시퀀스를 원할 경우 로드를 시도하기 전에 데이터를 정렬해야 합니다.

로드 유틸리티는 기존 정의에 따라 인덱스를 빌드합니다. 고유 키에 대한 중복을 처리하기 위해 예외 테이블을 사용합니다. 이 유틸리티는 참조 무결성을 강제하거나 제한조건 점검을 수행하거나 로드되는 테이블에 종속된 요약 테이블을 갱신하지 않습니다. 참조 또는 점검 제한조건을 포함하는 테이블은 무결성 설정 보류 상태에 놓입니다. 로드되고 있는 테이블에 종속되고 REFRESH IMMEDIATE를 사용하여 정의된 요약 테이블도 무결성 설정 보류 상태에 놓입니다. 이들 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY문을 발행하십시오. 복제된 요약 테이블에서는 로드 조작을 수행할 수 없습니다.

클러스터된 인덱스의 경우 로드하기 전에 클러스터링 인덱스에 대해 데이터를 정렬해야 합니다. 다차원적으로 클러스터된(MDC) 테이블에 로드하는 경우에는 데이터를 정렬하지 않아도 됩니다.

로드 세션 - CLP 예

예 1

TABLE1에는 다음과 같이 5개의 컬럼이 있습니다.

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1에는 다음 6개 요소가 있습니다.

- ELE1 위치 01 - 20
- ELE2 위치 21 - 22
- ELE3 위치 23 - 23
- ELE4 위치 24 - 27
- ELE5 위치 28 - 31

- ELE6 위치 32 - 32
- ELE7 위치 33 - 40

데이터 레코드:

```
1...5...10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    XXN
Test data 4,5 and 6 WWN6789    Y
```

다음 명령은 파일로부터 테이블을 로드합니다.

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
   method L (1 20, 21 22, 24 27, 28 31)
   null indicators (0,0,23,32)
   insert into table1 (col1, col5, col2, col3)
```

주:

1. MODIFIED BY 매개변수에서 striptblanks를 지정하면 VARCHAR 컬럼에 공백이 강제로 절단됩니다(예를 들어 COL1의 경우 행 1, 2, 3에서 각각 길이가 11, 17, 19바이트임).
2. MODIFIED BY 매개변수에서 reclen=40을 지정하면 각 입력 레코드 끝에 개행 문자가 없으며 각 레코드 길이는 40바이트임을 나타냅니다. 마지막 8바이트는 테이블을 로드하는 데 사용되지 않습니다.
3. 입력 파일에 COL4가 제공되지 않았으므로 디폴트값(NOT NULL WITH DEFAULT 정의됨)을 사용하여 TABLE1로 삽입됩니다.
4. 위치 23 및 32를 사용하여 TABLE1의 COL2 및 COL3이 지정된 행에 널(NULL)로 로드되었는지 여부를 표시합니다. 지정된 레코드에서 컬럼의 널(NULL) 표시기 위치에 Y가 있으면 컬럼은 널(NULL)입니다. N이 있으면 입력 레코드에서 컬럼의 데이터 위치에 지정된 데이터 값(L(.....)로 정의됨)이 행의 컬럼 데이터 소스로 사용됩니다. 이 예에서 행 1의 컬럼은 널(NULL)이 아니며 행 2의 COL2는 널(NULL)이고 행 3의 COL3도 널(NULL)입니다.
5. 이 예에서 COL1 및 COL5의 NULL INDICATORS는 영(0)으로 지정되며 데이터가 널(NULL) 입력 가능하지 않음을 나타냅니다.
6. 지정된 컬럼의 NULL INDICATOR는 입력 레코드 어느 위치에도 나타날 수 있지만 위치를 지정해야 하며 Y 또는 N 값을 제공해야 합니다.

예 2(덤프 파일 사용)

FRIENDS 테이블은 다음과 같이 정의됩니다.

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

다음 데이터 레코드를 이 테이블로 로드하려는 경우,

```
23, 24, bobby
, 45, john
4,, mary
```

첫 번째 INT가 널(NULL)이고 컬럼 정의가 NOT NULL을 지정하므로 두 번째 행은 거부됩니다. DEL 형식과 일치하지 않는 초기 문자가 들어 있는 컬럼은 오류를 생성하므로 해당 레코드는 거부됩니다. 이러한 레코드는 덤프 파일에 작성될 수 있습니다.

문자 분리문자 외부의 컬럼에 나타나는 DEL 데이터가 무시되지만 경고가 생성됩니다. 예를 들면, 다음과 같습니다.

```
22,34,"bob"
24,55,"sam" sdf
```

유틸리티는 테이블의 세 번째 컬럼에서 ″sam″을 로드하고 경고에 ″sdf″ 문자가 플래그됩니다. 이 레코드는 거부되지 않습니다. 다음은 또 다른 예입니다.

```
22 3, 34,"bob"
```

유틸리티는 22,34,"bob"를 로드하고 22 뒤의 컬럼 1에 일부 데이터가 무시된다는 경고가 생성됩니다. 이 레코드는 거부되지 않습니다.

예 3(ID 컬럼을 포함하는 테이블 로드)

TABLE1에는 다음과 같이 4개의 컬럼이 있습니다.

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2는 C2가 GENERATED ALWAYS ID 컬럼이라는 점만 제외하고 TABLE1과 동일합니다.

DATAFILE1의 데이터 레코드(DEL 형식):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2의 데이터 레코드(DEL 형식):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

주:

1. 다음 명령은 DATAFILE1에 행 1과 행 2에 대한 식별 값이 제공되지 않았기 때문에 이들 행에 대한 식별 값을 생성합니다. 그러나 행 3 및 4에는 각각 사용자 제공 ID 값 100 및 101이 지정됩니다.

   ```
   db2 load from datafile1.del of del replace into table1
   ```

2. 모든 행에서 ID 값이 생성되도록 DATAFILE1을 TABLE1로 로드하려면 다음 명령 중 하나를 실행하십시오.

   ```
   db2 load from datafile1.del of del method P(1, 3, 4)
         replace into table1 (c1, c3, c4)   db2load from datafile1.del of
         del modified by identityignore
         replace into table1
   ```

3. 각 행에서 ID 값이 생성되도록 DATAFILE2를 TABLE1로 로드하려면 다음 명령 중 하나를 실행하십시오.

   ```
   db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
      db2 load from datafile2.del of del modified by identitymissing
         replace into table1
   ```

4. ID 값 100 및 101이 행 3 및 4에 지정되도록 DATAFILE1을 TABLE2로 로드하려면 다음 명령을 실행하십시오.

   ```
   db2 load from datafile1.del of del modified by identityoverride
         replace into table2
   ```

   이 경우 행 1 및 2는 거부됩니다. 사용자 제공 값을 위해 시스템 생성 ID 값을 겹쳐쓰도록 유틸리티에 지시했기 때문입니다. 그러나 사용자 제공 값이 없으면 행이 거부됩니다. ID 컬럼은 내재적으로 널(NULL)일 수 없기 때문입니다.

5. ID 관련 파일 유형 수정자를 사용하지 않고 DATAFILE1이 TABLE2로 로드된 경우 행 1 및 2는 로드되지만 행 3 및 4는 거부됩니다. 고유한 널(NULL)이 아닌 값을 제공하지 않았으며 ID 컬럼이 GENERATED ALWAYS이기 때문입니다.

예 4(CURSOR에서 로드)

MY.TABLE1에는 다음 3개 컬럼이 있습니다.

- ONE INT
- TWO CHAR(10)
- THREE DATE

MY.TABLE2에는 다음 3개 컬럼이 있습니다.

- ONE INT
- TWO CHAR(10)
- THREE DATE

Cursor MYCURSOR는 다음과 같이 정의됩니다.

```
declare mycursor cursor for select * from my.table1
```

다음 명령은 MY.TABLE1의 모든 데이터를 MY.TABLE2로 로드합니다.

```
load from mycursor of cursor method P(1,2,3) insert into
     my.table2(one,two,three)
```

주:

1. 단일 LOAD 명령에서는 하나의 cursor 이름만 지정할 수 있습니다. 즉, `load from mycurs1, mycurs2 of cursor...`는 허용되지 않습니다.
2. `P` 및 `N`은 cursor에서 로드하는 유일한 유효한 METHOD 값입니다.
3. 이 예에서 METHOD P 및 삽입 컬럼 목록 `(one,two,three)`는 디폴트값을 표현하므로 생략될 수 있습니다.
4. MY.TABLE1은 테이블, 뷰, 별명 또는 별칭일 수 있습니다.

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

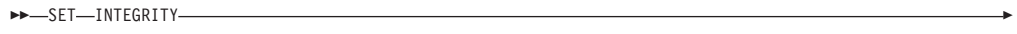
  - DATAACCESS 권한
  - DBADM 권한
- 무결성 설정 보류 상태에 테이블을 위치시킵니다.

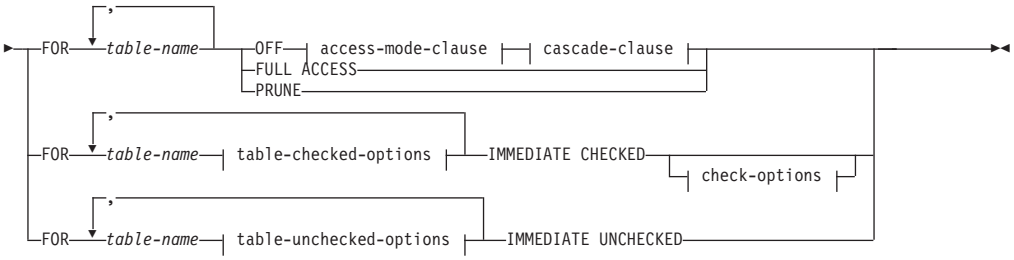
  명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.
  - CONTROL 특권
    - 지정된 테이블
    - 명령문에 의해 무결성 설정 보류 상태에 놓이게 될 하위 외부 키 테이블,
    - 명령문에 의해 무결성 설정 보류 상태에 놓이게 될 하위 직접 구체화된 쿼리 테이블 및
    - 명령문에 의해 무결성 설정 보류 상태에 놓이게 될 하위 직접 스테이징 테이블
  - LOAD 권한
  - DATAACCESS 권한
  - DBADM 권한
- 테이블을 전체 액세스 상태로 두십시오.

  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.

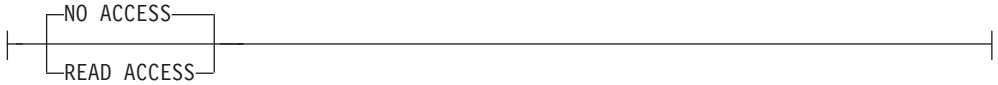
  명령문의 권한 부여 ID는 적어도 다음과 같은 특권 중 하나를 가지고 있어야 합니다.
  - 프룬(prune) 중인 테이블에 대한 CONTROL 특권
  - DATAACCESS 권한

구문

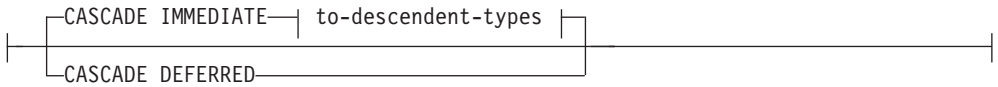




access-mode-clause:



cascade-clause:



to-descendent-types:

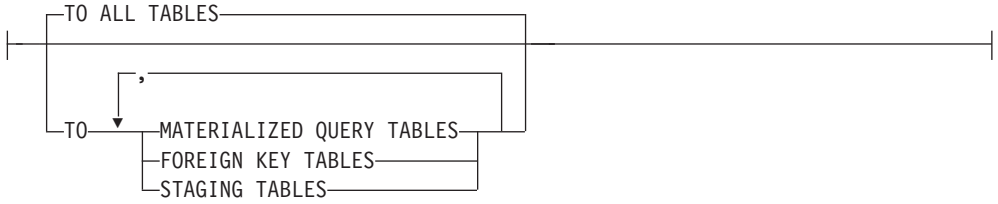
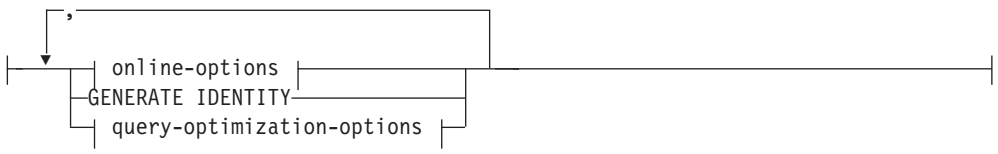


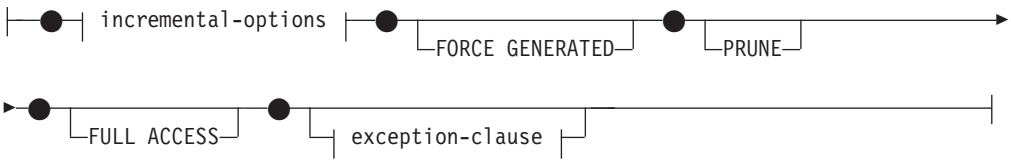
table-checked-options:



online-options:

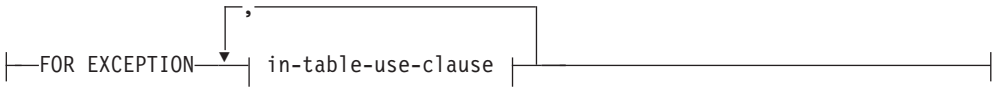
query-optimization-options:

점검 옵션:



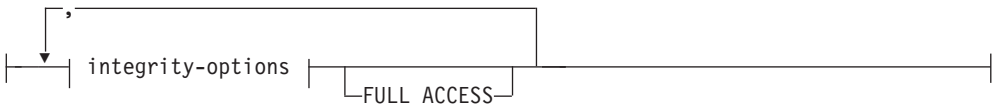
incremental-options:

exception-clause:

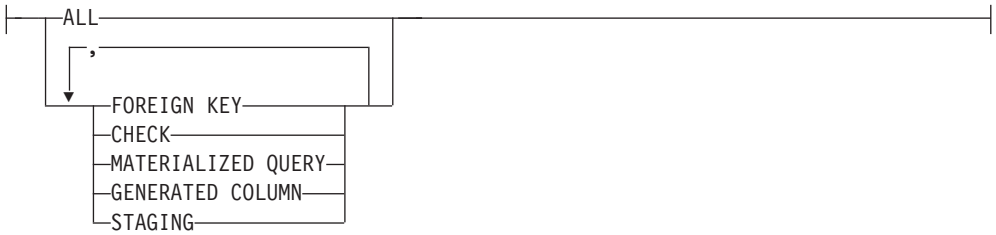


in-table-use-clause:

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 스테이징 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

커미트되지 않은 분리 수준이 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 합

니다. 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 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 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 『주』 절을 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 사용자가 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 시

스템이 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 결

정합니다. 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 구

체화된 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 로드 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 위반하

는 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 바이트는 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 테이블을 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
```

LOAD QUERY

처리 중에 로드 조작의 상태를 점검하고 테이블 상태를 리턴합니다. 로드가 처리 중이 아닌 경우 테이블 상태만 리턴됩니다. 이 명령을 성공적으로 호출하려면 동일한 데이터베이스에 대한 연결 및 개별 CLP 세션도 필요합니다. 로컬 또는 리모트 사용자가 사용할 수 있습니다.

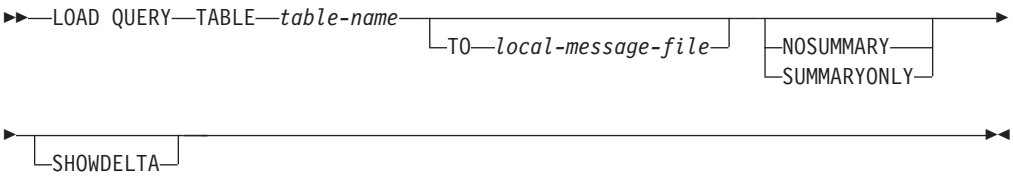
권한 부여

없음

필수 연결

데이터베이스

명령 구문



명령 매개변수

NOSUMMARY

로드 요약 정보(읽은 행 수, 생략한 행 수, 로드한 행, 거부된 행, 삭제된 행, 커미트된 행 및 경고 수)가 보고되지 않도록 지정합니다.

SHOWDELTA

새 정보만(LOAD QUERY 명령의 마지막 호출 이후 발생한 로드 이벤트에 관한) 보고되도록 지정합니다.

SUMMARYONLY

로드 요약 정보만 보고되도록 지정합니다.

TABLE *table-name*

데이터가 현재 로드되고 있는 테이블의 이름을 지정합니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

TO *local-message-file*

로드 조작 중 발생하는 경고 및 오류 메시지의 목적지를 지정합니다. 이 파일은 LOAD 명령에 대해 지정된 *message-file*일 수 없습니다. 파일이 이미 존재하는 경우 로드 유틸리티가 생성한 모든 메시지가 파일에 추가됩니다.

예:

많은 양의 데이터를 BILLYBOB 데이터베이스의 STAFF 테이블에 로드하는 사용자가 로드 조작의 상태를 점검하려고 합니다. 사용자는 다음을 지정할 수 있습니다.

```
db2 connect to billybob
db2 load query table staff to /u/mydir/staff.tempmsg
```

출력 파일 /u/mydir/staff.tempmsg는 다음과 비슷할 수 있습니다.

```
SQL3501W  데이터베이스에 대한 포워드 복구가 사용 불가능하므로
테이블이 있는 테이블 스페이스가 백업 보류 상태에 놓이지 않습니다.

SQL3109N  유틸리티가 파일 "/u/mydir/data/staffbig.del"에서
데이터를 로드하기 시작합니다.

SQL3500W  유틸리티가 "03-21-2002 11:31:16.597045"에 "LOAD"
단계를 시작 중입니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "0".

SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "104416".

SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "205757".

SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "307098".

SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "408439".

SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3532I  로드 유틸리티가 현재 "LOAD" 단계에 있습니다.


읽은 행 수                = 453376
건너뛴 행 수              = 0
로드된 행 수              = 453376
거부된 행 수              = 0
삭제된 행 수              = 0
커미트된 행 수            = 408439
경고 수                   = 0

테이블 상태:
  로드 진행 중
```

사용 시 참고사항

잠금 외에 로드 유틸리티는 테이블 상태를 사용하여 테이블에 대한 액세스를 제어합니다. LOAD QUERY 명령을 사용하여 테이블 상태를 판별할 수 있습니다. LOAD

QUERY는 현재 로드되지 않고 있는 테이블에서 사용할 수 있습니다. 파티션된 테이블의 경우 보고되는 상태는 대응하는 가시적 데이터 파티션 상태 중에서 가장 제한적입니다. 예를 들어 단일 데이터 파티션은 읽기 액세스 전용 상태이고 다른 모든 데이터 파티션은 일반 상태에 있는 경우 로드 쿼리 조작은 읽기 액세스 전용 상태를 리턴합니다. 로드 조작은 데이터 파티션의 서브세트를 테이블의 나머지와 다른 상태에 두지 않습니다. LOAD QUERY에 의해 설명되는 테이블 상태는 다음과 같습니다.

일반 다른(비정상) 테이블 상태 중 하나에 있지 않은 경우 테이블은 일반 상태에 있습니다. 일반 상태는 테이블이 작성된 후 테이블의 초기 상태입니다.

무결성 설정 보류
테이블은 아직 검증되지 않은 제한조건을 갖습니다. 테이블을 무결성 설정 보류 상태에서 벗어나도록 하려면 SET INTEGRITY문을 사용하십시오. 로드 유틸리티는 제한조건을 갖는 테이블에서 로드 조작을 시작할 때 테이블을 무결성 설정 보류 상태에 둡니다.

로드 진행 중
이것은 로드 조작 중에만 적용되는 임시 상태입니다. 로드 조작이 실패했거나 인터럽트된 로드 진행 중 상태에서 테이블을 벗어나도록 하는 데 대한 정보는 *관련 링크* 섹션에서 로드 조작 후 보류 상태에 대한 섹션을 참조하십시오. 로드 진행 중 테이블 스페이스 상태도 참조하십시오.

로드 보류
로드 조작이 이 테이블에서 사용 중이었지만 데이터를 커미트할 수 있기 전에 중단되었습니다. LOAD TERMINATE, LOAD RESTART 또는 LOAD REPLACE 명령을 발행하여 테이블을 이 상태에서 벗어나게 하십시오.

읽기 액세스 전용
테이블은 ALLOW READ ACCESS 옵션이 지정된 경우 로드 조작 중에 이 상태에 있습니다. 읽기 액세스 전용은 다른 응용프로그램 및 유틸리티가 로드 조작 전에 존재한 데이터에 대해 읽기 액세스할 수 있는 임시 상태입니다.

Reorg 보류
REORG 명령은 ALTER TABLE문이 테이블에서 실행되기를 권장합니다. 테이블에 다시 액세스할 수 있으려면 클래식 REORG를 수행해야 합니다.

사용 불가능
테이블이 사용 불가능합니다. 테이블은 백업에서만 삭제 또는 리스토어할 수 있습니다. 복구 불가능한 로드 조작을 통한 롤 포워드는 테이블을 사용 불가능 상태에 둡니다.

로드 재시작 불가능
테이블은 로드 재시작 조작을 허용하지 않는 부분적으로 로드된 상태에 있습니다. 이 테이블은 로드 보류 상태에도 있습니다. LOAD TERMINATE 또는 LOAD REPLACE 명령을 발행하여 테이블을 로드 재시작 불가능 상태에서 벗

어나게 하십시오. 테이블은 성공적으로 재시작 또는 종료되지 않은 실패한 로드 조작 후 롤 포워드 조작이 수행될 때 또는 테이블이 로드 진행 중 또는 로드 보류 상태에 있는 중에 작성된 온라인 백업으로부터 리스토어 조작이 수행될 때 로드 재시작 불가능 상태에 들어갑니다. 어느 경우에나 로드 재시작 조작에 필요한 정보를 믿을 수 없으며 로드 재시작 불가능 상태는 로드 재시작 조작이 발생하지 못하게 합니다.

Unknown

LOAD QUERY 명령이 테이블 상태를 판별할 수 없습니다.

현재 IBM DB2 데이터베이스 제품이 지원하는 최소 25개 테이블 또는 테이블 스페이스가 있습니다. 이들 상태는 특정 상황에서 데이터에 대한 액세스를 제어하거나 특정 사용자 조치를 유추하거나 필요한 경우 데이터베이스의 무결성을 보호하는 데 사용됩니다. 대부분은 로드 유틸리티 또는 백업 및 리스토어 유틸리티 같은 DB2 유틸리티 중 하나의 조작과 관련된 이벤트의 결과입니다.

종속 테이블 스페이스가 로드 조작 전에 더 이상 Quiesce되지 않는 경우(Quiesce는 지속적 잠금임) 로드 진행 중 테이블 스페이스 상태는 로드 조작 중에 종속 테이블의 백업을 금지합니다. 로드 진행 중 테이블 스페이스 상태는 로드 진행 중 테이블 상태와 다릅니다. 모든 로드 조작은 로드 진행 중 테이블 상태를 사용하지만 COPY NO 옵션이 지정된 로드 조작(복구 가능한 데이터베이스에 대한)도 로드 진행 중 테이블 스페이스 상태를 사용합니다.

다음 표는 지원되는 각 테이블 상태를 설명합니다. 또한 데이터베이스를 관리하는 중에 발생할 수 있는 상태를 해석하고 응답하는 방법을 정확하게 보여주는 작업 예를 제공합니다. 예는 AIX에서 실행된 명령 스크립트에서 제공됩니다. 사용자가 직접 복사, 붙여넣기 및 실행할 수 있습니다. UNIX가 아닌 시스템에서 DB2 데이터베이스 제품을 실행 중인 경우 모든 경로가 시스템에 맞는 형식인지 확인하십시오. 대부분의 예는 DB2 데이터베이스 제품과 함께 제공되는 SAMPLE 데이터베이스의 테이블을 기초로 합니다. 몇 가지 예는 SAMPLE 데이터베이스의 파트가 아닌 시나리오가 필요하지만, SAMPLE 데이터베이스에 대한 연결을 시작점으로 사용할 수 있습니다.

표 47. 지원되는 테이블 상태

상태	예:

로드 보류	상당한 양의 데이터(예를 들어, 20000개 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 제공될 때, NEWSTAFF라는 새 테이블인 로드 조작의 대상 테이블이 들어있는 작은 테이블 스페이스를 작성하십시오.
	`connect to sample;`
	`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256);`
	`create table newstaff like staff in ts1;`
	`load from staffdata.del of del insert into newstaff;`
	`load query table newstaff;`
	`load from staffdata.del of del terminate into newstaff;`
	`load query table newstaff;`
	`connect reset;`
	LOAD QUERY 명령이 리턴하는 정보는 NEWSTAFF 테이블이 로드 보류 상태에 있음을 표시합니다. 로드가 작업을 종료한 후 테이블은 일반 상태에 있습니다.

로드 진행 중	상당한 양의 데이터(예를 들어, 20000 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 있는 경우,
	`connect to sample;`
	`create table newstaff like staff;`
	`load from staffdata.del of del insert into newstaff;`
	로드 조작이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
	`connect to sample;`
	`load query table newstaff;`
	`connect reset;`
	LOAD QUERY 명령이 리턴하는 정보는 NEWSTAFF 테이블이 로드 진행 중 상태에 있음을 나타냅니다.

일반	`connect to sample;`
	`create table newstaff like staff;`
	`load query table newstaff;`
	LOAD QUERY 명령이 리턴하는 정보는 NEWSTAFF 테이블이 일반 상태에 있음을 표시합니다.

표 47. 지원되는 테이블 상태 (계속)

상태	예:

로드 재시작 불가능	상당한 양의 데이터(예를 들어, 20000 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 있는 경우,
	`update db cfg for sample using logretain recovery;`
	`backup db sample;`
	`connect to sample;`
	`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256);`
	`create table newstaff like staff in ts1;`
	`connect reset;`
	`backup db sample;`
	이 백업 이미지의 시간소인은 20040629205935입니다.
	`connect to sample;`
	`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
	`connect reset;`
	`restore db sample taken at 20040629205935;`
	`rollforward db sample to end of logs and stop;`
	`connect to sample;`
	`load query table newstaff;`
	`connect reset;`
	LOAD QUERY 명령이 리턴하는 정보는 NEWSTAFF 테이블이 로드 재시작 불가능 및 로드 보류 상태에 있음을 표시합니다.
	`connect to sample;`
	`load from staffdata.del of del terminate into newstaff copy yes to /home/melnyk/backups;`
	`load query table newstaff;`
	`connect reset;`
	LOAD QUERY 명령이 리턴하는 정보는 NEWSTAFF 테이블이 이제 일반 상태에 있음을 표시합니다.

읽기 액세스 전용	상당한 양의 데이터(예를 들어, 20000 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 있는 경우,
	`connect to sample;`
	`export to st_data.del of del select * from staff;`
	`create table newstaff like staff;`
	`import from st_data.del of del insert into newstaff;`
	`load from staffdata.del of del insert into newstaff allow read access;`
	로드 조작이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
	`connect to sample;`
	`load query table newstaff;`
	`select * from newstaff;`
	`connect reset;`
	LOAD QUERY 명령이 리턴하는 정보는 NEWSTAFF 테이블이 읽기 액세스 전용 및 로드 진행 중 상태에 있음을 표시합니다. 쿼리는 STAFF 테이블의 익스포트된 컨텐츠인 로드 조작 전에 NEWSTAFF 테이블에 존재했던 데이터만 리턴합니다.

무결성 설정 보류	다음 컨텐츠를 갖는 로드 입력 파일 staff_data.del을 가정합니다.
	11,"Melnyk",20,"Sales",10,70000,15000:
	`connect to sample;`
	`alter table staff add constraint max_salary check (100000 - salary > 0);`
	`load from staff_data.del of del insert into staff;`
	`load query table staff;`
	LOAD QUERY 명령이 리턴하는 정보는 STAFF 테이블이 무결성 설정 보류 상태에 있음을 표시합니다.

표 47. 지원되는 테이블 상태 (계속)

상태	예:

사용 불가능	다음 컨텐츠를 갖는 로드 입력 파일 staff_data.del을 가정합니다.
	
	11,″Melnyk″,20,″Sales″,10,70000,15000:
	
	`update db cfg for sample using logretain recovery;`
	`backup db sample;`
	
	이 백업 이미지의 시간소인은 20040629182012입니다.
	
	`connect to sample;`
	`load from staff_data.del of del insert into staff nonrecoverable;`
	`connect reset;`
	`restore db sample taken at 20040629182012;`
	`rollforward db sample to end of logs and stop;`
	`connect to sample;`
	`load query table staff;`
	`connect reset;`
	
	LOAD QUERY 명령이 리턴하는 정보는 STAFF 테이블이 사용 불가능 상태에 있음을 표시합니다.

테이블 상태에 대한 추가 정보는 *관련 링크* 섹션을 참조하십시오.

LIST UTILITIES 명령을 사용하여 로드 조작의 진행을 모니터할 수도 있습니다.

LIST TABLESPACES

테이블 스페이스와 현재 데이터베이스의 테이블 스페이스에 대한 정보를 나열합니다.

이 명령으로 표시되는 정보는 테이블 스페이스 스냅샷에서도 사용 가능합니다.

범위

이 명령은 명령이 실행되는 데이터베이스 파티션의 정보만 리턴합니다.

권한 부여

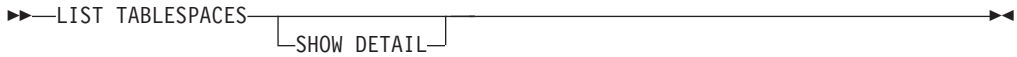
다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *sysmon*
- *dbadm*
- LOAD 권한

필수 연결

데이터베이스

명령 구문



명령 매개변수

SHOW DETAIL

이 옵션을 지정하지 않은 경우 각 테이블 스페이스에 대한 다음의 기본 정보가 제공됩니다.

- 테이블 스페이스 ID
- 이름
- 유형(시스템 관리 스페이스 또는 데이터베이스 관리 스페이스)
- 컨텐츠(모든 데이터, long 또는 인덱스 데이터, 또는 임시 데이터)
- 상태(현재 테이블 스페이스 상태를 표시하는 16진수 값). 외부에서 볼 수 있는 테이블 스페이스 상태는 특정 상태 값의 16진수 합으로 구성됩니다. 예를 들어, 상태가 ″Quiesce 상태: EXCLUSIVE″ 및 ″로드 보류″이면, 값은 0x0004 + 0x0008이고 0x000c가 됩니다. db2tbst(테이블 스페이스 상태 확보) 명령을 사용하여 제공된 16진수 값과 연관되는 테이블 스페이스 상태를 확보할 수 있습니다. 다음은 sqlutil.h에 나열된 비트 정의입니다.

```
0x0          일반
0x1          Quiesce됨: SHARE
0x2          Quiesce됨: UPDATE
0x4          Quiesce됨: EXCLUSIVE
0x8          로드 보류
0x10         삭제 보류
0x20         백업 보류
0x40         롤 포워드 진행 중
0x80         롤 포워드 보류
0x100        리스토어 보류
0x100        복구 보류(사용되지 않음)
0x200        작동 불가능화 보류
0x400        Reorg 진행 중
0x800        백업 진행 중
0x1000       스토리지가 정의되어야 함
0x2000       리스토어 진행 중
0x4000       오프라인 및 액세스 불가능
0x8000       삭제 보류
0x20000      로드 진행 중
0x2000000    스토리지가 정의되어야 함
0x4000000    StorDef가 '최종' 상태에 있음
0x8000000    StorDef가 롤 포워드 이전에 변경되었음
0x10000000   DMS 재조정 진행 중
0x20000000   테이블 스페이스 삭제 진행 중
0x40000000   테이블 스페이스 작성 진행 중
```

이 옵션을 지정한 경우 각 테이블 스페이스에 대한 다음 추가 정보가 제공됩니다.

- 총 페이지 수
- 사용할 수 있는 페이지 수
- 사용된 페이지 수
- 여유 페이지 수
- 상위 워터 마크(water mark)(페이지 수)
- 페이지 크기(바이트)
- Extent 크기(페이지 수)
- 프리페치 크기(페이지 수)
- 컨테이너 수
- 최소 복구 시간(0이 아닌 경우에만 표시됨)
- 상태 변경 테이블 스페이스 ID(테이블 스페이스 상태가 ″로드 보류″ 또는 ″삭제 보류″인 경우에만 표시됨)
- 상태 변경 오브젝트 ID(테이블 스페이스 상태가 ″로드 보류″ 또는 ″삭제 보류″인 경우에만 표시됨)
- Quiescer 수(테이블 스페이스 상태가 ″Quiesce 상태: SHARE″, ″Quiesce 상태: UPDATE″ 또는 ″Quiesce 상태: EXCLUSIVE″인 경우에만 표시됨)
- 각 Quiescer의 테이블 스페이스 ID 및 오브젝트 ID(Quiescer 수가 0보다 큰 경우에만 표시됨)

예:

다음은 LIST TABLESPACES SHOW DETAIL의 두 가지 샘플 출력입니다.

```
          현재 데이터베이스에 대한 테이블 스페이스
테이블 스페이스 ID                    = 0
이름                                  = SYSCATSPACE
유형                                  = 데이터베이스 관리 스페이스
내용                                  = 모든 데이터
상태                                  = 0x0000
  세부사항 설명:
    정상
전체 페이지 수                        = 895
사용 가능한 페이지 수                 = 895
사용된 페이지 수                      = 895
사용 가능한 페이지 수                 = 적용되지 않음
상위 워터 마크(water mark)(페이지)              = 적용되지 않음
페이지 크기(바이트)                   = 4096
Extent 크기(페이지)                   = 32
프리페치 크기(페이지)                 = 32
컨테이너 수                           = 1

테이블 스페이스 ID                    = 1
이름                                  = TEMPSPACE1
유형                                  = 시스템 관리 스페이스
내용                                  = 임시 데이터
```

```
상태                                 = 0x0000
  세부사항 설명:
    정상
전체 페이지 수                       = 1
사용 가능한 페이지 수                = 1
사용된 페이지 수                     = 1
사용 가능한 페이지 수                = 적용되지 않음
상위 워터 마크(water mark)(페이지)              = 적용되지 않음
페이지 크기(바이트)                  = 4096
Extent 크기(페이지)                  = 32
프리페치 크기(페이지)                = 32
컨테이너 수                          = 1

테이블 스페이스 ID                   = 2
이름                                 = USERSPACE1
유형                                 = 데이터베이스 관리 스페이스
내용                                 = 모든 데이터
상태                                 = 0x000c
  세부사항 설명:
    Quiesce됨: EXCLUSIVE
    로드 보류
전체 페이지 수                       = 337
사용 가능한 페이지 수                = 337
사용된 페이지 수                     = 337
사용 가능한 페이지 수                = 적용되지 않음
상위 워터 마크(water mark)(페이지)              = 적용되지 않음
페이지 크기(바이트)                  = 4096
Extent 크기(페이지)                  = 32
프리페치 크기(페이지)                = 32
컨테이너 수                          = 1
상태 변경 테이블 스페이스 ID         = 2
상태 변경 오브젝트 ID                = 3
Quiescer 수                          = 1
  Quiescer 1:
    테이블 스페이스 ID               = 2
    오브젝트 ID                      = 3
파티션된 데이터베이스 서버 환경에서는 현재 노드의
테이블 스페이스만 나열됩니다.


         현재 데이터베이스에 대한 테이블 스페이스
테이블 스페이스 ID                   = 0
이름                                 = SYSCATSPACE
유형                                 = 시스템 관리 스페이스
내용                                 = 모든 데이터
상태                                 = 0x0000
  세부사항 설명:
    정상
전체 페이지 수                       = 1200
사용 가능한 페이지 수                = 1200
사용된 페이지 수                     = 1200
사용 가능한 페이지 수                = 적용되지 않음
상위 워터 마크(water mark)(페이지)              = 적용되지 않음
페이지 크기(바이트)                  = 4096
Extent 크기(페이지)                  = 32
프리페치 크기(페이지)                = 32
컨테이너 수                          = 1

테이블 스페이스 ID                   = 1
```

```
 이름                                    = TEMPSPACE1
 유형                                    = 시스템 관리 스페이스
 내용                                    = 임시 데이터
 상태                                    = 0x0000
   세부사항 설명:
     정상
 전체 페이지 수                          = 1
 사용 가능한 페이지 수                   = 1
 사용된 페이지 수                        = 1
 사용 가능한 페이지 수                   = 적용되지 않음
 상위 워터 마크(water mark)(페이지)              = 적용되지 않음
 페이지 크기(바이트)                     = 4096
 Extent 크기(페이지)                     = 32
 프리페치 크기(페이지)                   = 32
 컨테이너 수                             = 1

테이블 스페이스 ID                       = 2
 이름                                    = USERSPACE1
 유형                                    = 시스템 관리 스페이스
 내용                                    = 모든 데이터
 상태                                    = 0x0000
   세부사항 설명:
     정상
 전체 페이지 수                          = 1
 사용 가능한 페이지 수                   = 1
 사용된 페이지 수                        = 1
 사용 가능한 페이지 수                   = 적용되지 않음
 상위 워터 마크(water mark)(페이지)              = 적용되지 않음
 페이지 크기(바이트)                     = 4096
 Extent 크기(페이지)                     = 32
 프리페치 크기(페이지)                   = 32
 컨테이너 수                             = 1

테이블 스페이스 ID                       = 3
 이름                                    = DMS8K
 유형                                    = 데이터베이스 관리 스페이스
 내용                                    = 모든 데이터
 상태                                    = 0x0000
   세부사항 설명:
     정상
 전체 페이지 수                          = 2000
 사용 가능한 페이지 수                   = 1952
 사용된 페이지 수                        = 96
 사용 가능한 페이지 수                   = 1856
 상위 워터 마크(water mark)(페이지)              = 96
 페이지 크기(바이트)                     = 8192
 Extent 크기(페이지)                     = 32
 프리페치 크기(페이지)                   = 32
 컨테이너 수                             = 2

테이블 스페이스 ID                       = 4
 이름                                    = TEMP8K
 유형                                    = 시스템 관리 스페이스
 내용                                    = 임시 데이터
 상태                                    = 0x0000
   세부사항 설명:
     정상
 전체 페이지 수                          = 1
```

```
 사용 가능한 페이지 수                = 1
 사용된 페이지 수                     = 1
 사용 가능한 페이지 수                = 적용되지 않음
 상위 워터 마크(water mark)(페이지)              = 적용되지 않음
 페이지 크기(바이트)                  = 8192
 Extent 크기(페이지)                  = 32
 프리페치 크기(페이지)                = 32
 컨테이너 수                          = 1
파티션된 데이터베이스 서버 환경에서는 현재 노드의
테이블 스페이스만 나열됩니다.
```

사용 시 참고사항

파티션된 데이터베이스 환경에서는 이 명령이 데이터베이스의 모든 테이블 스페이스를 리턴하지는 않습니다. 모든 테이블 스페이스 목록을 확보하려면 SYSCAT.TABLESPACES를 쿼리하십시오.

테이블 스페이스 재조정 중에 사용 가능한 페이지 수에는 새로 추가된 컨테이너 페이지가 포함되지만 이 새 페이지는 재조정이 완료될 때까지 사용 가능한 페이지 수에 반영되지 않습니다. 테이블 스페이스 재조정이 진행 중 상태가 아니면 사용된 페이지 수 + 여유 페이지 수가 사용할 수 있는 페이지 수와 동일하게 됩니다.

현재 IBM DB2 데이터베이스 제품이 지원하는 최소 25개 테이블 또는 테이블 스페이스가 있습니다. 이들 상태는 특정 상황에서 데이터에 대한 액세스를 제어하거나 특정 사용자 조치를 유추하거나 필요한 경우 데이터베이스의 무결성을 보호하는 데 사용됩니다. 대부분은 로드 유틸리티 또는 백업 및 리스토어 유틸리티 같은 DB2 유틸리티 중 하나의 조작과 관련된 이벤트의 결과입니다.

다음 표는 지원되는 각 테이블 스페이스 상태를 설명합니다. 또한 데이터베이스를 관리하는 중에 발생할 수 있는 상태를 해석하고 응답하는 방법을 정확하게 보여주는 작업 예를 제공합니다. 예는 AIX에서 실행된 명령 스크립트에서 제공됩니다. 사용자가 직접 복사, 붙여넣기 및 실행할 수 있습니다. UNIX가 아닌 시스템에서 DB2 데이터베이스 제품을 실행 중인 경우 모든 경로가 시스템에 맞는 형식인지 확인하십시오. 대부분의 예는 DB2 데이터베이스 제품과 함께 제공되는 SAMPLE 데이터베이스의 테이블을 기초로 합니다. 몇 가지 예는 SAMPLE 데이터베이스의 파트가 아닌 시나리오가 필요하지만, SAMPLE 데이터베이스에 대한 연결을 시작점으로 사용할 수 있습니다.

표 48. 지원되는 테이블 스페이스 상태

상태	16진수 상태 값	설명	예:

백업 보류	0x20	테이블 스페이스는 특정 시점 테이블 스페이스 롤 포워드 작업 후, 또는 COPY NO 옵션을 지정하는 로드 작업(복구 가능한 데이터베이스에 대한) 후 이 상태가 됩니다. 테이블 스페이스(또는 전체 데이터베이스)는 테이블 스페이스를 사용하기 전에 백업해야 합니다. 테이블 스페이스를 백업하지 않은 경우, 테이블 스페이스 내의 테이블은 쿼리할 수 있지만 갱신할 수는 없습니다.
		주: 데이터베이스는 또한 롤 포워드 복구에 사용 가능하도록 설정된 후 즉시 백업해야 합니다. 데이터베이스는 **logretain** 데이터베이스 구성 매개변수를 RECOVERY로 설정하거나, **userexit** 데이터베이스 구성 매개변수를 YES로 설정한 경우에 복구할 수 있습니다. 데이터베이스가 백업될 때까지 (**backup_pending** 정보용 데이터베이스 구성 매개변수 값이 NO로 설정될 때) 데이터베이스를 활성화하거나 연결할 수 없습니다.
			1. 다음 컨텐츠가 있는 로드 입력 파일 staff_data.del을 가정합니다.
			11,″Melnyk″,20,″Sales″,10,70000,15000:
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`load from staff_data.del of del messages load.msg insert into staff copy no;`
			`update staff set salary = 69000 where id = 11;`
			2.
			`update db cfg for sample using logretain recovery;`
			`connect to sample;`

백업 진행 중	0x800	백업 작업 중에만 적용되는 임시 상태입니다.	온라인 BACKUP DATABASE 명령을 발행하십시오.
			`backup db sample online;`
			백업 작업이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			또는
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 백업 진행 중 상태임을 표시합니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

DMS 재조정 진행 중	0x10000000	데이터 재조정 작업 중에만 적용되는 임시 상태입니다. 데이터베이스 관리 스페이스(DMS)로 정의된 테이블 스페이스에 새 컨테이너가 추가되거나 기존 컨테이너가 확장되는 경우, 테이블 스페이스 데이터의 재조정이 발생할 수 있습니다. *재조정*은 데이터를 스트라이프된 상태로 보존하려고 할 때 하나의 위치에서 다른 위치로 테이블 스페이스 Extent를 이동하는 프로세스입니다. *Extent*는 컨테이너 스페이스 단위(페이지 수로 측정)이며, 스트라이프는 테이블 스페이스에 대한 *컨테이너 세트 사이의* Extent 계층입니다.	상당한 양의 데이터(예를 들어, 20000 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 있는 경우,
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`load from staffdata.del of del insert into newstaff nonrecoverable;`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk /NODE0000/SQL00001/ts1c2' 1024);`
			`list tablespaces;`
			`connect reset;`
			
			TS1에 대해 리턴되는 정보는 테이블 스페이스가 DMS 재조정 진행 중 상태임을 표시합니다.

작동 불가능화 보류	0x200	테이블 스페이스는 데이터베이스 롤 포워드 작업 중에 이 상태가 될 수 있으며, 롤 포워드 작업을 종료하여 더 이상 이 상태에 있으면 안됩니다. 상태는 테이블 스페이스가 오프라인이 되어 트랜잭션에 대한 보상 로그 레코드가 기록되지 않도록 하는 조건으로 트리거됩니다. 이 테이블 스페이스 상태의 출현 및 후속 소멸은 사용자에게 표시되지 않습니다.	이 테이블 스페이스 상태를 보여주는 예는 이 문서의 범위를 벗어납니다.
삭제 보류	0x8000	해당 컨테이너 중 하나 이상이 데이터베이스 재시작 작업 중에 문제점이 있는 것으로 발견되는 경우 테이블 스페이스는 이 상태에 있습니다. (이 데이터베이스가 있는 이전 세션이 비정상적으로 종료된 경우(예: 전원 고장 시) 데이터베이스를 재시작해야 합니다.) 테이블 스페이스가 삭제 보류 상태에 있으면 사용할 수 없게 되고 삭제만 가능합니다.	이 테이블 스페이스 상태를 보여주는 예는 이 문서의 범위를 벗어납니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

로드 진행 중	0x20000	COPY NO 옵션을 지정하는 로드 작업(복구 가능한 데이터베이스에 대한) 중에만 적용되는 임시 상태입니다. 로드 진행 중 테이블 상태도 참조하십시오.	상당한 양의 데이터(예를 들어, 20000 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 있는 경우,
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create table newstaff like staff;`
			`load from staffdata.del of del insert into newstaff copy no;`
			`connect reset;`
			
			로드 조작이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			`list tablespaces;`
			`connect reset;`
			
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 로드 진행 중(및 백업 보류) 상태임을 표시합니다.

일반	0x0	테이블 스페이스 상태가 다른 (비정상) 테이블 스페이스 상태가 아닌 경우, 테이블 스페이스는 일반 상태입니다. 일반 상태는 테이블 스페이스가 작성된 후 초기 상태입니다.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
			`list tablespaces show detail;`

오프라인 및 액세스 불가능	0x4000	해당 컨테이너 중 하나 이상에 문제점이 있는 경우 테이블 스페이스는 이 상태에 있습니다. 컨테이너가 의도하지 않게 이름이 바뀌거나 이동 또는 손상되었을 수 있습니다. 문제점이 수정되고, 테이블 스페이스와 연관되는 컨테이너에 다시 액세스할 수 있으면, 데이터베이스에서 모든 응용프로그램의 연결을 끊은 후 다시 데이터베이스에 연결하여 이 비정상 상태를 제거할 수 있습니다. 또는 SWITCH ONLINE 절을 지정하는 ALTER TABLESPACE 문을 발행하여 데이터베이스에서 다른 응용프로그램의 연결을 끊지 않고 테이블 스페이스에서 오프라인 및 액세스 불가능 상태를 제거할 수도 있습니다.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc2' 1024);`
			`export to st_data.del of del select * from staff;`
			`create table stafftemp like staff in ts1;`
			`import from st_data.del of del insert into stafftemp;`
			`connect reset;`
			
			테이블 스페이스 컨테이너 이름을 tsc1에서 tsc3으로 바꾼 후 STAFFTEMP 테이블을 쿼리하십시오.
			`connect to sample;`
			`select * from stafftemp;`
			
			쿼리는 SQL0290N(테이블 스페이스 액세스가 허용되지 않음)을 리턴하고, LIST TABLESPACES 명령은 TS1에 대해 0x4000(오프라인 및 액세스 불가능) 상태 값을 리턴합니다. 테이블 스페이스 컨테이너 이름을 tsc3에서 다시 tsc1로 바꾸십시오. 이제 쿼리가 제대로 실행됩니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

Quiesce 상태 독점	0x4	테이블 스페이스 Quiesce 함수를 호출하는 응용프로그램이 테이블 스페이스에 대해 독점 (읽기 또는 쓰기) 액세스를 가지고 있는 경우 테이블 스페이스는 이 상태에 있습니다. QUIESCE TABLESPACES FOR TABLE 명령을 발행하여 테이블 스페이스가 명시적으로 Quiesce 상태 독점 상태가 되도록 할 수 있습니다.	Quiesce 상태 독점으로 설정하기 전에 테이블 스페이스 상태가 일반 상태인지 확인하십시오.
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff exclusive;`
			`connect reset;`
			
			다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			`select * from staff where id=60;`
			`update staff set salary=50000 where id=60;`
			`list tablespaces;`
			`connect reset;`
			
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 Quiesce 상태 독점 상태임을 표시합니다.

Quiesce 상태 공유	0x1	테이블 스페이스 Quiesce 함수를 호출하는 응용프로그램과 동시 응용프로그램이 테이블 스페이스에 대해 읽기(쓰기는 아님) 액세스를 가지고 있는 경우 테이블 스페이스는 이 상태에 있습니다. QUIESCE TABLESPACES FOR TABLE 명령을 발행하여 테이블 스페이스가 명시적으로 Quiesce 상태 공유 상태가 되도록 할 수 있습니다.	Quiesce 상태 공유로 설정하기 전에 테이블 스페이스 상태가 일반 상태인지 확인하십시오.
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff share;`
			`connect reset;`
			
			다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			`select * from staff where id=40;`
			`update staff set salary=50000 where id=40;`
			`list tablespaces;`
			`connect reset;`
			
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 Quiesce 상태 공유 상태임을 표시합니다.

Quiesce 상태 갱신	0x2	테이블 스페이스 Quiesce 함수를 호출하는 응용프로그램이 테이블 스페이스에 대해 독점 쓰기 액세스를 가지고 있는 경우 테이블 스페이스는 이 상태에 있습니다. QUIESCE TABLESPACES FOR TABLE 명령을 발행하여 테이블 스페이스가 명시적으로 Quiesce 상태 갱신 상태가 되도록 할 수 있습니다.	Quiesce 상태 갱신으로 설정하기 전에 테이블 스페이스 상태가 일반 상태인지 확인하십시오.
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff intent to update;`
			`connect reset;`
			
			다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			`select * from staff where id=50;`
			`update staff set salary=50000 where id=50;`
			`list tablespaces;`
			`connect reset;`
			
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 Quiesce 상태 갱신 상태임을 표시합니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

Reorg 진행 중	0x400	Reorg 작업 중에만 적용되는 임시 상태입니다.	REORG TABLE 명령을 발행합니다.
			`connect to sample;`
			`reorg table staff;`
			`connect reset;`
			Reorg 작업이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			또는
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 Reorg 진행 중 상태임을 표시합니다.
			주: SAMPLE 데이터베이스와 관련된 테이블 재구성 작업은 짧은 시간 안에 완료되므로, 결국 이 접근 방식으로 Reorg 진행 중 상태를 조사하는 것은 어려울 수 있습니다.

리스토어 보류	0x100	경로 재지정된 리스토어 조작의 첫 번째 파트 후(즉, SET TABLESPACE CONTAINERS 명령이 발행되기 전) 데이터베이스의 테이블 스페이스는 이 상태가 됩니다. 테이블 스페이스(또는 전체 데이터베이스)는 테이블 스페이스를 사용하기 전에 리스토어해야 합니다. 리스토어 조작이 완료될 때까지(**restore_pending** 정보용 데이터베이스 구성 매개변수 값이 NO로 설정될 때) 데이터베이스에 연결할 수 없습니다.	'스토리지를 정의할 수 있음' 상태에 있는, 경로 재지정된 리스토어 조작의 첫 번째 파트가 완료될 때, 모든 테이블 스페이스는 리스토어 보류 상태가 됩니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

리스토어 진행 중	0x2000	리스토어 조작 중에만 적용되는 임시 상태입니다.	`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`backup db sample tablespace (userspace1);`
			
			이 백업 이미지의 시간소인은 다음과 같습니다.
			
			20040611174124
			
			`restore db sample tablespace (userspace1) online`
			`taken at 20040611174124;`
			
			리스토어 조작이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			
			`connect to sample;`
			
			1.
			
			`list tablespaces show detail;`
			
			또는
			
			2.
			
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			USERSPACE1에 대해 리턴되는 정보는 테이블 스페이스가 리스토어 진행 중 상태임을 표시합니다.

롤 포워드 보류	0x80	테이블 스페이스는 복구 가능한 데이터베이스에 대한 리스토어 조작 후에 이 상태가 됩니다. 테이블 스페이스(또는 전체 데이터베이스)는 테이블 스페이스를 사용하기 전에 롤 포워드해야 합니다. 데이터베이스는 **logretain** 데이터베이스 구성 매개변수를 RECOVERY로 설정하거나, **userexit** 데이터베이스 구성 매개변수를 YES로 설정한 경우에 복구할 수 있습니다. 롤 포워드 작업이 완료될 때까지(**rollfwd_pending** 정보용 데이터베이스 구성 매개변수 값이 NO로 설정될 때) 데이터베이스를 활성화거나 데이터베이스에 연결할 수 없습니다.	리스토어 진행 중 상태의 온라인 테이블 스페이스 리스토어 조작이 완료되면, 테이블 스페이스 USERSPACE1은 롤 포워드 보류 상태가 됩니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

롤 포워드 진행 중	0x40	롤 포워드 작업 중에만 적용되는 임시 상태입니다.	상당한 양의 데이터(예를 들어, 20000 이상의 레코드)를 갖는 로드 입력 파일 staffdata.del이 있는 경우,
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`connect reset;`
			`backup db sample tablespace (ts1) online;`
			이 백업 이미지의 시간소인은 다음과 같습니다.
			20040630000715
			`connect to sample;`
			`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
			`connect reset;`
			`restore db sample tablespace (ts1) online taken at 20040630000715;`
			`rollforward db sample to end of logs and stop tablespace (ts1) online;`
			롤 포워드 작업이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			또는
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			TS1에 대해 리턴되는 정보는 테이블 스페이스가 롤 포워드 진행 중 상태임을 표시합니다.

스토리지를 정의할 수 있음	0x2000000	경로 재지정된 리스토어 조작의 첫 번째 파트 후(즉, SET TABLESPACE CONTAINERS 명령이 발행되기 전) 데이터베이스의 테이블 스페이스는 이 상태가 됩니다. 원하면 컨테이너를 재정의할 수 있습니다.	`backup db sample;`
			이 백업 이미지의 시간소인이 20040613204955인 것을 가정합니다.
			`restore db sample taken at 20040613204955 redirect;`
			`list tablespaces;`
			LIST TABLESPACES 명령에 의해 리턴되는 정보는 모든 테이블 스페이스가 '스토리지를 정의할 수 있음' 및 '리스토어 보류' 상태에 있음을 표시합니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

스토리지를 정의해야 함	0x1000	테이블 스페이스 컨테이너 설정 단계가 생략되거나 테이블 스페이스 컨테이너 설정 단계 중에 지정된 컨테이너를 획득할 수 없는 경우 새 데이터베이스로 경로 재지정된 리스토어 조작 동안 데이터베이스의 테이블 스페이스는 이 상태가 됩니다. 예를 들어 유효하지 않은 경로 이름을 지정한 경우 나중의 경우가 발생할 수 있으며, 그렇지 않으면 디스크 스페이스가 충분하지 않습니다.	`backup db sample;`
			
			이 백업 이미지의 시간소인이 20040613204955인 것을 가정합니다.
			
			`restore db sample taken at 20040613204955 into mydb redirect;`
			`set tablespace containers for 2 using (path 'ts2c1');`
			`list tablespaces;`
			
			LIST TABLESPACES 명령에 의해 리턴되는 정보는 테이블 스페이스 SYSCATSPACE 및 테이블 스페이스 TEMPSPACE1이 스토리지를 정의해야 함, 스토리지를 정의할 수 있음 및 리스토어 보류 상태에 있음을 표시합니다. 스토리지를 정의해야 함 상태가 스토리지를 정의할 수 있음 상태보다 우선합니다.

테이블 스페이스 작성 진행 중	0x40000000	테이블 스페이스 작성 작업 중에만 적용되는 임시 상태입니다.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc3' 1024);`
			
			테이블 스페이스 작성 작업이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			
			`connect to sample;`
			
			1.
			
			`list tablespaces show detail;`
			
			또는
			
			2.
			
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			TS1, TS2 및 TS3에서 리턴되는 정보는 테이블 스페이스가 테이블 스페이스 작성 진행 중 상태에 있음을 나타냅니다.

표 48. 지원되는 테이블 스페이스 상태 (계속)

상태	16진수 상태 값	설명	예:

테이블 스페이스 삭제 진행 중	0x20000000	테이블 스페이스 삭제 작업 중에만 적용되는 임시 상태입니다.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc3' 1024);`
			`drop tablespace ts1;`
			`drop tablespace ts2;`
			`drop tablespace ts3;`
			
			테이블 스페이스 삭제 작업이 실행 중인 동안 다른 세션에서 다음 스크립트를 실행하십시오.
			`connect to sample;`
			
			1.
			`list tablespaces show detail;`
			
			또는
			
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			TS1, TS2 및 TS3에서 리턴되는 정보는 테이블 스페이스가 테이블 스페이스 삭제 진행 중 상태에 있음을 나타냅니다.

테이블 스페이스 상태에 대한 추가 정보는 *관련 링크* 섹션을 참조하십시오.

제 5 장 기타 데이터 이동 옵션

ADMIN_MOVE_TABLE 프로시저를 사용하여 테이블 온라인 이동

ADMIN_MOVE_TABLE 프로시저를 사용하여 온라인 또는 오프라인 테이블 이동을 수행할 수 있습니다. 비용, 스페이스, 이동 성능 및 트랜잭션 오버헤드보다 가용성이 중요한 경우 오프라인 테이블 이동 대신 온라인 테이블 이동을 사용하십시오.

테이블 및 인덱스, 스테이징 테이블 및 추가 로그 항목의 사본을 수용할 충분한 디스크 스페이스가 있는지 확인하십시오.

프로시저가 수행하는 각 연산마다 한 번씩 호출하여 스토어드 프로시저를 한 번 또는 여러 번 호출하면 테이블을 온라인으로 이동할 수 있습니다. 여러 호출을 사용할 때 이동 취소 또는 목표 테이블을 갱신하기 위해 오프라인 상태로 전환하는 시기 제어와 같은 추가 옵션이 제공됩니다.

SYSPROC.ADMIN_MOVE_TABLE 프로시저를 호출하면 소스 테이블의 쉐도우 사본이 작성됩니다. 복사 단계 중 소스 테이블의 변경사항(갱신, 삽입 또는 삭제)이 트리거를 사용하여 캡처되며 스테이징 테이블에 배치됩니다. 복사 단계가 완료된 후 스테이징 테이블에 캡처된 변경사항을 쉐도우 사본으로 재생합니다. 그런 다음 스토어드 프로시저는 일시적으로 소스 테이블을 오프라인으로 전환하고 쉐도우 사본 및 인덱스에 소스 테이블 이름 및 인덱스 이름을 지정합니다. 쉐도우 테이블이 온라인으로 전환되어 소스 테이블을 교체합니다. 디폴트로 소스 테이블이 삭제되지만 KEEP 옵션을 사용하여 다른 이름으로 보유할 수 있습니다.

인덱스(특히 고유 인덱스의 경우)가 없으면 테이블을 온라인으로 이동하지 마십시오. 고유 인덱스 없이 테이블을 온라인으로 이동하면 교착 상태가 발생하고 재생이 복잡하거나 비용이 많이 듭니다.

프로시저

테이블을 온라인으로 이동하려면 다음을 수행하십시오.

1. 다음 중 한 방법으로 ADMIN_MOVE_TABLE 프로시저를 호출하십시오.
   - 최소한 소스 테이블의 스키마 이름, 소스 테이블 이름 및 연산 유형 MOVE를 지정하여 ADMIN_MOVE_TABLE 프로시저를 한 번 호출하십시오. 예를 들어, 다음 구문을 사용하여 동일한 테이블 스페이스의 기존 테이블로 데이터를 이동하십시오.

     ```
     CALL SYSPROC.ADMIN_MOVE_TABLE (
     'schema name',
     'source table',
     ```

```
'',
'',
'',
'',
'',
'',
'',
'',
'MOVE')
```

- 최소한 소스 테이블의 스키마 이름, 소스 테이블 이름 및 연산 이름을 지정하여 각 연산마다 한 번씩 ADMIN_MOVE_TABLE 프로시저를 여러 번 호출하십시오. 예를 들어, 다음 구문을 사용하여 동일한 테이블 스페이스의 새 테이블로 데이터를 이동하십시오.

```
CALL SYSPROC.ADMIN_MOVE_TABLE (
'schema name',
'source table',
'',
'',
'',
'',
'',
'',
'',
'',
'operation name')
```

  여기서 *operation name*은 INIT, COPY, REPLAY, VERIFY 또는 SWAP 값 중 하나입니다. 이 연산 순서에 따라 프로시저를 호출해야 합니다. 예를 들어, 첫 번째 호출에서 연산 이름으로 INIT를 지정해야 합니다.

  주: VERIFY 연산은 비용이 많이 듭니다. 테이블 이동에서 필요한 경우에만 이 연산을 수행하십시오.

2. 온라인 이동이 실패한 경우 다시 실행하십시오.
   a. 테이블 이동 실패의 원인이 된 문제점을 수정하십시오.
   b. SYSTOOLS.ADMIN_MOVE_TABLE 프로토콜 테이블에서 상태를 쿼리하여 테이블 이동이 실패한 경우 진행한 단계를 판별하십시오.
   c. 해당 옵션을 지정하여 스토어드 프로시저를 다시 호출하십시오.
      - 프로시저의 상태가 INIT인 경우 INIT 옵션을 사용하십시오.
      - 프로시저의 상태가 COPY인 경우 COPY 옵션을 사용하십시오.
      - 프로시저의 상태가 REPLAY인 경우 REPLAY 또는 SWAP 옵션을 사용하십시오.
      - 프로시저의 상태가 CLEANUP인 경우 CLEANUP 옵션을 사용하십시오.

   온라인 테이블 이동의 상태가 COMPLETED 또는 CLEANUP이 아닌 경우 스토어드 프로시저의 CANCEL 옵션을 지정하여 이동을 취소할 수 있습니다.

이 예에서 T1을 오프라인으로 전환하지 않고 ACCOUNTING 테이블 스페이스로 이동할 T1 테이블이 SVALENTI 스키마에 있다고 가정하십시오. 테이블을 이동하려면 다음과 같이 ADMIN_MOVE_TABLE을 호출하십시오. 새 테이블 스페이스로 테이블을 이동하므로 DATA, INDEX 및 LONG 테이블 스페이스를 지정해야 합니다.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'ACCOUNTING',
'ACCOUNTING',
'ACCOUNTING',
'',
'',
'',
'',
'',
'',
'MOVE')
```

이 예에서, 동일한 테이블 스페이스에서 T1 테이블을 이동한다고 가정하십시오. T1 컬럼 중 하나인 C1은 사용하지 않는 데이터 유형 LONG VARCHAR을 사용하므로 호환 가능한 데이터 유형을 사용하도록 C1을 변경할 수 있습니다. 테이블을 이동하고 C1의 새 데이터 유형을 사용하려면 다음과 같이 ADMIN_MOVE_TABLE 프로시저를 호출하십시오.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'',
'',
'',
'',
'',
'',
'',
'C1 VARCHAR(1000), C2 INT(5), C3 CHAR(5), C4 CLOB',
'',
'MOVE')
```

주: 이 연산 중 컬럼 이름을 변경할 수 없습니다.

이 예에서 다음 명령문을 발행하여 T2 테이블을 작성했다고 가정하십시오.

```
CREATE TABLE T1(C1 BIGINT,C2 BIGINT,C3 CHAR(20),C4 DEC(10,2),C5
TIMESTAMP,C6 BIGINT GENERATED ALWAYS AS (C1+c2),C7 GRAPHIC(10),C8
VARGRAPHIC(20),C9 XML
```

동일한 테이블 스페이스에서 테이블을 이동하려면 다음과 같이 C5 및 C6 컬럼을 제거하여 ADMIN_MOVE_TABLE 프로시저를 호출하십시오.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'',
'',
'',
```

```
'',
'',
'',
'c1 BIGINT,c2 BIGINT ,c3 CHAR(20),c4 DEC(10,2),c7 GRAPHIC(10),c8
VARGRAPHIC(20),c9 XML',
'',
'MOVE')
```

DB2 Connect에서 데이터 이동

호스트 데이터베이스 시스템 및 워크스테이션 사이에서 데이터를 이동해야 하는 복합 환경에서 작업하는 경우 호스트 및 워크스테이션 사이에서 데이터 전송을 담당하는 게이트웨이로 DB2 Connect를 사용할 수 있습니다(그림 17 참조).

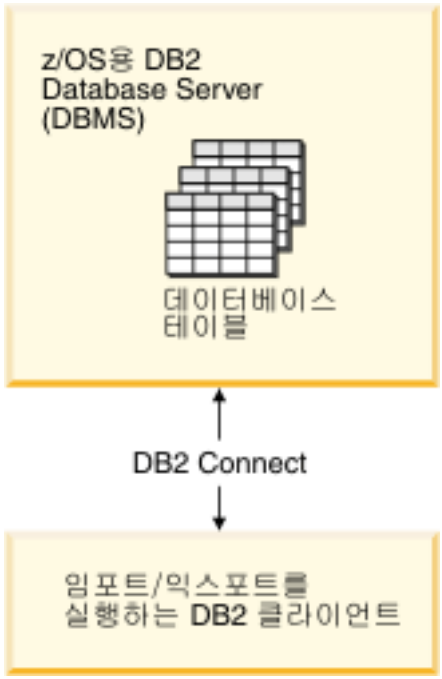


그림 17. DB2 Connect를 통해 임포트/익스포트

DB2 익스포트 및 임포트 유틸리티를 사용하면 IBM 메인프레임 서버 데이터베이스에서 DB2 Connect 워크스테이션의 파일로(또는 이 반대 방향) 데이터를 이동할 수 있습니다. 그러면 이 익스포트 또는 임포트 형식을 지원하는 다른 응용프로그램 또는 관계형 데이터베이스 관리 시스템에서 데이터를 사용할 수 있습니다. 예를 들어, IBM 메인프레임 서버 데이터베이스에서 PC/IXF 파일로 데이터를 익스포트한 후 Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터베이스로 임포트할 수 있습니다.

데이터베이스 클라이언트 또는 DB2 Connect 워크스테이션에서 익스포트 및 임포트 조작을 수행할 수 있습니다.

주:

1. 익스포트 또는 임포트할 데이터는 두 데이터베이스에서 적용 가능한 *크기* 및 데이터 유형 제한사항을 준수해야 합니다.
2. 임포트 성능을 향상시키기 위해 복합 쿼리를 사용할 수 있습니다. 임포트 유틸리티에서 `compound` 파일 유형 수정자를 지정하여 지정된 수의 쿼리 명령문을 하나의 블록으로 그룹화합니다. 그러면 네트워크 오버헤드가 줄어들고 응답 시간이 향상될 수 있습니다.

DB2 Connect에서 익스포트 및 임포트 조작은 다음 조건을 만족해야 합니다.

- 파일 유형은 PC/IXF여야 합니다.
- 데이터와 호환 가능한 속성을 포함하는 목표 테이블은 이를 임포트하기 전에 목표 서버에서 작성해야 합니다. db2look 유틸리티를 사용하여 소스 테이블의 속성을 가져올 수 있습니다. INSERT만 지원되는 옵션이므로 DB2 Connect를 통한 임포트로는 테이블을 작성할 수 없습니다.

이 조건을 만족하지 못하면 조작에 실패하고 오류 메시지가 리턴됩니다.

주: 인덱스 정의는 익스포트에서 저장되거나 임포트에서 사용되지 못합니다.

혼합된 데이터(1바이트 및 2바이트 데이터를 모두 포함하는 컬럼)를 익스포트 또는 임포트하는 경우 다음을 고려합니다.

- EBCDIC로 데이터를 저장하는 시스템(MVS, System z®, IBM Power Systems®, VM 및 VSE)의 경우, 시프트 아웃(Shift-Out) 및 시프트 인(Shift-In) 문자는 2바이트 데이터의 시작 및 끝을 표시합니다. 데이터베이스 테이블의 컬럼 길이를 정의하는 경우 이 문자를 표시할 공간이 충분히 있어야 합니다.
- 컬럼 데이터의 패턴이 일관되지 않으면 가변 길이 문자 컬럼이 권장됩니다.

워크스테이션에서 호스트 서버로 데이터 이동

호스트 또는 System i 서버 데이터베이스에서 데이터를 이동하려면 다음을 수행하십시오.

1. DB2 테이블에서 PC/IXF 파일로 데이터를 익스포트하십시오.
2. INSERT 옵션을 사용하여 호스트 서버 데이터베이스에서 호환 가능한 테이블로 PC/IXF 파일을 임포트하십시오.

호스트 서버 데이터베이스에서 워크스테이션으로 데이터를 이동하려면 다음을 수행하십시오.

1. 호스트 서버 데이터베이스 테이블에서 PC/IXF 파일로 데이터를 익스포트하십시오.
2. DB2 테이블로 PC/IXF 파일을 임포트하십시오.

예

이 예에서는 워크스테이션에서 호스트 또는 System i 서버 데이터베이스로 데이터를 이동하는 방법을 보여줍니다.

다음 명령을 실행하여 외부 IXF 형식으로 데이터를 익스포트합니다.

```
db2 export to staff.ixf of ixf select * from userid.staff
```

다음 명령을 실행하여 목표 DB2 데이터베이스에 대한 DRDA 연결을 설정합니다.

```
db2 connect to cbc664 user admin using xxx
```

아직 없는 경우 목표 DB2 데이터베이스 인스턴스에서 목표 테이블을 작성합니다.

```
CREATE TABLE mydb.staff (ID SMALLINT NOT NULL, NAME VARCHAR(9),
       DEPT SMALLINT, JOB CHAR(5), YEARS SMALLINT, SALARY DECIMAL(7,2),
       COMM DECIMAL(7,2))
```

데이터를 임포트하려면 다음 명령을 실행하십시오.

```
db2 import from staff.ixf of ixf insert into mydb.staff
```

데이터의 각 행은 IXF 형식으로 파일에서 읽혀지며 SQL INSERT문이 실행되어 행을 테이블 mydb.staff로 삽입합니다. 모든 데이터를 목표 테이블로 이동할 때까지 단일 행이 계속 삽입됩니다.

세부사항 정보는 IBM Redbooks® 서적의 ″DB2 계열에서 데이터 이동″을 참조하십시오. 이 Redbooks 서적은 다음 URL에서 찾을 수 있습니다.
http://www.redbooks.ibm.com/redbooks/SG246905.

구성요소별 IBM 복제 도구

IBM에서는 두 가지 기본 복제 솔루션, Q 복제 및 SQL 복제를 제공합니다.

Q 복제의 기본 구성요소는 Q Capture 프로그램 및 Q Apply 프로그램입니다. SQL 복제의 기본 구성요소는 Capture 프로그램 및 Apply 프로그램입니다. 두 복제 유형 모두 복제 경보 모니터 도구를 공유합니다. 복제 센터 및 ASNCLP 명령행 프로그램을 사용하여 이 복제 구성요소를 설정 및 관리할 수 있습니다.

다음 목록에서는 이 복제 구성요소를 간략히 요약합니다.

Q Capture 프로그램

DB2 소스 테이블의 변경 사항을 찾는 DB2 복구 로그를 읽고 커미트된 소스 데이터를 WebSphere® MQ 메시지로 변환합니다. 이 메시지는 XML 형식으로 서브스크라이빙 응용프로그램에 게시하거나 압축된 형식으로 Q Apply 프로그램에 복제될 수 있습니다.

Q Apply 프로그램

큐에서 WebSphere MQ 메시지를 가져와 SQL문으로 변환하고 목표 테이블 또는 스토어드 프로시저를 갱신합니다. 지원되는 목표에는 DB2 데이터베이스 또는 서브시스템 및 Oracle, Sybase, Informix® 및 Microsoft® SQL Server 데이터베이스(페더레이티드 서버 별칭을 통해 액세스됨)가 포함됩니다.

Capture 프로그램

등록된 소스 테이블 또는 뷰의 변경 사항이 있는지 확인하기 위해 DB2 복구 로그를 읽고 커미트된 트랜잭션 데이터를 데이터 변경(CD) 테이블이라고 하는 관계형 테이블에 스테이징합니다. 이 테이블은 목표 시스템이 복사할 준비가 될 때까지 데이터를 저장하는 위치입니다. SQL 복제에서는 데이터 일관 변경(CCD) 테이블이라고 하는 스테이징 테이블을 비DB2 소스 테이블에 대한 변경 사항 레코드로 채우는 Capture 트리거를 제공합니다.

Apply 프로그램

스테이징 테이블에서 데이터를 읽고 변경 사항을 목표에 적절히 적용합니다. 비DB2 데이터 소스의 경우 Apply 프로그램에서는 페더레이티드 데이터베이스에서 해당 테이블의 별칭을 통해 CCD 테이블을 읽고 변경 사항을 목표 테이블에 적절히 적용합니다.

복제 경보 모니터

Q Capture, Q Apply, Capture 및 Apply 프로그램의 상태를 점검하는 유틸리티입니다. 여기서는 프로그램 종료, 경고 또는 오류 메시지 발행, 지정된 값의 임계값 도달 또는 특정 조치 수행과 같은 상황을 확인하고 전자 우편 서버, 호출기 또는 z/OS 콘솔로 통지를 발행합니다.

복제 센터를 사용하여 다음을 수행할 수 있습니다.

- 등록, 서브스크립션, 게시, 큐 맵, 경보 조건 및 기타 오브젝트를 정의합니다.
- 복제 프로그램을 시작, 중지, 일시중단, 재시작 및 재초기화합니다.
- 자동화된 복사 시간 제어를 지정합니다.
- 데이터에 대한 SQL의 개선된 기능을 지정합니다.
- 소스 및 목표 테이블 간 관계를 정의합니다.

스키마 복사

db2move 유틸리티 및 ADMIN_COPY_SCHEMA 프로시저를 통해 데이터베이스 스키마의 사본을 빠르게 만들 수 있습니다. 모델 스키마가 설정되면 이 스키마를 새 버전 작성을 위한 템플리트로 사용할 수 있습니다.

ADMIN_COPY_SCHEMA 프로시저를 사용하여 동일한 데이터베이스 내에서 단일 스키마를 복사합니다. 또는 -co COPY 조치와 함께 db2move 유틸리티를 사용하여 소스 데이터베이스에서 목표 데이터베이스로 단일 스키마 또는 여러 스키마를 복사합니다. 새 스키마 아래에서 소스 스키마의 데이터베이스 오브젝트 대부분이 목표 데이터베이스로 복사됩니다.

문제점 해결 추가 정보

ADMIN_COPY_SCHEMA 프로시저 및 db2move 유틸리티는 모두 LOAD 명령을 호출합니다. 로드가 처리되는 중 데이터베이스 목표 오브젝트가 상주하고 있는 테이블 스페이스는 백업 보류 상태가 됩니다.

ADMIN_COPY_SCHEMA 프로시저

COPYNO 옵션과 함께 이 프로시저를 사용하면 위의 주에서 설명한 것처럼 목표 오브젝트가 상주하고 있는 테이블 스페이스가 백업 보류 상태가 됩니다. 테이블 스페이스를 무결성 설정 보류 상태에서 벗어나도록 하기 위해 이 프로시저는 SET INTEGRITY문을 발행합니다. 목표 테이블 오브젝트에 정의된 참조 제한조건이 있는 경우 목표 테이블도 무결성 설정 보류 상태가 됩니다. 테이블 스페이스가 이미 백업 보류 상태이므로 ADMIN_COPY_SCHEMA 프로시저는 SET INTEGRITY문을 발행하지 못합니다.

이러한 상황을 해결하려면 영향을 받는 테이블 스페이스를 백업 보류 상태에서 벗어나도록 BACKUP DATABASE 명령을 발행하십시오. 다음으로 이 프로시저에서 생성한 오류 테이블의 **Statement_text** 컬럼을 검색하여 무결성 설정 보류 상태인 테이블 목록을 찾으십시오. 그런 다음 나열된 각 테이블에 대해 SET INTEGRITY문을 발행하여 각 테이블을 무결성 설정 보류 상태에서 벗어나도록 하십시오.

db2move 유틸리티

이 유틸리티는 허용되는 모든 스키마 오브젝트의 복사를 시도하지만 다음과 같은 유형의 예외가 적용됩니다.

- 테이블 계층
- 스테이징 테이블(다중 파티션 데이터베이스 환경에서는 로드 유틸리티에 의해 지원되지 않음)
- jars(Java™ 루틴 아카이브)
- 별칭
- 패키지
- 뷰 계층 구조
- 오브젝트 특권(모든 새 오브젝트는 디폴트 권한 부여를 사용하여 작성됨)
- 통계(새 오브젝트에는 통계 정보가 없음)
- 인덱스 확장자(사용자 정의 구조화된 유형 관련)

- 사용자 정의 구조화된 유형 및 해당 변환 함수

지원되지 않는 유형 오류

소스 스키마에서 한 가지 지원되지 않는 유형의 오브젝트가 발견된 경우 항목이 오류 파일에 로그되어 지원되지 않는 오브젝트 유형이 발견되었음을 나타냅니다. COPY 조작은 여전히 완료되며 이 조작을 통해 오브젝트가 복사되지 않았음을 사용자에게 알리기 위해 로그된 엔트리가 작성됩니다.

스키마로 결합되지 않은 오브젝트

스키마로 결합되지 않은 오브젝트(예: 테이블 스페이스 및 이벤트 모니터)는 스키마 복사 조작 중 조작되지 않습니다. 스키마 복사 조작이 호출되기 전에 목표 데이터베이스에서 해당 오브젝트를 작성해야 합니다.

복제된 테이블

복제된 테이블을 복사하는 경우 테이블의 새 사본을 복제할 수 없습니다. 해당 테이블은 일반 테이블로 다시 작성됩니다.

다른 인스턴스

소스 데이터베이스는 목표 데이터베이스와 동일한 인스턴스에 상주하지 않은 경우 카탈로그되어야 합니다.

SCHEMA_MAP 옵션

SCHEMA_MAP 옵션을 사용하여 목표 데이터베이스에서 다른 스키마 이름을 지정하면 스키마 복사 조작은 오브젝트 정의 명령문에 대한 최소한의 구문 분석만 수행하여 새 스키마 이름으로 원래 스키마 이름을 교체합니다. 예를 들어 SQL 프로시저의 컨텐츠 내에 나타나는 원래 스키마의 모든 인스턴스는 새 스키마 이름으로 교체되지 않습니다. 따라서 스키마 복사 조작에서 이러한 오브젝트를 재작성하는 데 실패합니다. 오류 파일에서 DDL을 사용하여 복사 조작 완료 후 이러한 실패한 오브젝트를 수동으로 다시 작성할 수 있습니다.

오브젝트 간에 상호 종속성

스키마 복사 조작은 이러한 오브젝트 간 상호 종속성을 충족하는 순서대로 오브젝트를 다시 작성합니다. 예를 들어 테이블 T1에 사용자 정의 함수(UDF) U1을 참조하는 컬럼이 있는 경우 T1을 다시 작성하기 전에 먼저 U1을 다시 작성합니다. 그러나 프로시저에 대한 종속성 정보는 카탈로그에서 쉽게 사용할 수 없습니다. 따라서 프로시저를 다시 작성할 때 스키마 복사 조작은 먼저 모든 프로시저를 다시 작성한 다음 실패한 프로시저를 다시 작성하도록 시도합니다. 실패한 프로시저가 이전 시도에서 성공적으로 작성된 프로시저에 종속되어 있다고 가정하면 이후 시도에서 성공적으로 다시 작성됩니다. 후속 시도에서 하나 이상을 성공적으로 다시 작성할 수 있을 때까지 스키마 복사 조작은 실패한 프로시저를 다시 작성하도록 계속해서 시도합니다. 프로시저를 다시 작성하는 모든 시도 중 오류(및 DDL)는 오류 파일에 로그됩니다. 오류 파일에서 동일한 프로시저에 대해 여러 항목을 확인할 수 있더라도 다음 시도에서 해당 프로시저가 성공적으로 다시 작성되었을 수 있습니다. 스키마 복사 조작 완료 시

SYSCAT.PROCEDURES 테이블을 쿼리하여 오류 파일에 나열된 이러한 프로시저가 성공적으로 다시 작성되었는지 판별해야 합니다.

자세한 정보는 ADMIN_COPY_SCHEMA 프로시저 및 db2move 유틸리티를 참조하십시오.

db2move 유틸리티를 사용하는 스키마 사본의 예

-co COPY 조치와 함께 db2move 유틸리티를 사용하여 소스 데이터베이스에서 목표 데이터베이스로 하나 이상의 스키마를 복사합니다. 모델 스키마가 설정되면 이 스키마를 새 버전 작성을 위한 템플리트로 사용할 수 있습니다.

예 1: -c COPY 옵션 사용

db2move -co COPY 옵션의 다음 예에서는 스키마 BAR를 복사하고 샘플 데이터베이스에서 목표 데이터베이스로 해당 스키마의 이름을 FOO로 바꿉니다.

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u userid -p password
```

새(목표) 스키마 오브젝트는 소스 스키마의 오브젝트와 동일한 오브젝트 이름을 사용하여 작성되지만 목표 스키마 규정자가 포함되어 있습니다. 소스 테이블의 데이터가 포함되는지 여부와 관계없이 테이블 사본을 작성할 수 있습니다. 소스 및 목표 데이터베이스는 다른 시스템에 있을 수 있습니다.

예 2: COPY 조작 중 테이블 스페이스 이름 맵핑 지정

다음 예에서는 db2move COPY 조작 중 소스 시스템의 테이블 스페이스 대신 사용될 특정 테이블 스페이스 이름 맵핑을 지정하는 방법을 보여줍니다. 목표 테이블 스페이스가 디폴트 테이블 스페이스 선택 알고리즘을 사용하여 선택되어야 함을 표시하는 SYS_ANY 키워드를 지정할 수 있습니다. 이 경우 db2move 유틸리티는 다음과 같이 목표로 사용할 수 있는 모든 사용 가능한 테이블 스페이스를 선택합니다.

```
db2move sample COPY -sn BAR -co target_db  target schema_map
      "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u userid -p password
```

모든 테이블 스페이스에 SYS_ANY 키워드를 사용할 수 있습니다. 또는 사용자는 다음과 같이 일부 테이블 스페이스에 대해 특정 맵핑을 지정하고 나머지에 대해서는 디폴트 테이블 스페이스 선택 알고리즘을 지정할 수 있습니다.

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
 ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
 -u userid -p password
```

이것은 테이블 스페이스 TS1이 TS2로 맵핑되고, TS3은 TS4에 맵핑되지만 나머지 테이블 스페이스는 디폴트 테이블 스페이스 선택 알고리즘을 사용함을 나타냅니다.

예 3: COPY 조작 후 오브젝트 소유자 변경

사용자가 성공적인 COPY 후에 목표 스키마에 작성된 각 새 오브젝트의 소유자를 변경할 수 있습니다. 목표 오브젝트의 기본 소유자는 연결 사용자입니다. 이 옵션을 지정하면 아래와 같이 소유권이 새 소유자에게 이전됩니다.

```
db2move sample COPY -sn BAR -co target_db  target schema_map
      "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
      -u userid -p password
```

목표 오브젝트의 새 소유자는 jrichards입니다.

소스 및 목표 스키마가 다른 시스템 상주하고 있는 경우 목표 시스템에서 db2move 유틸리티가 호출되어야 합니다. 하나의 데이터베이스에서 다른 데이터베이스로 스키마를 복사하려면 이 조치에는 소스 데이터베이스에서 복사되는 스키마 이름 목록(쉼표로 구분됨)과 목표 데이터베이스 이름이 필요합니다.

스키마를 복사하려면 OS 명령 프롬프트에서 다음과 같이 db2move를 발생하십시오.

```
db2move <dbname> COPY -co <COPY- options>
 -u <userid> -p <password>
```

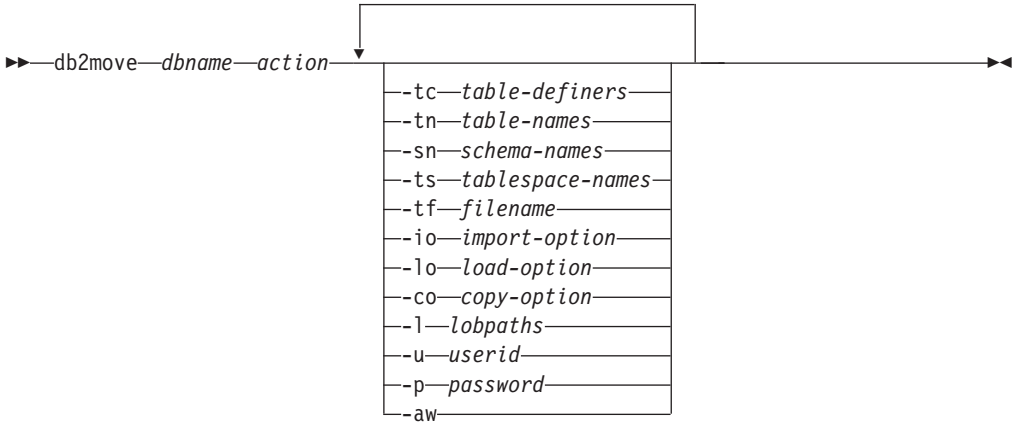
db2move - 데이터베이스 이동 도구

이 도구는 EXPORT/IMPORT/LOAD 모드에서 사용할 때 워크스테이션에 있는 DB2 데이터베이스 사이에 많은 수의 테이블 이동을 용이하게 합니다. 도구는 시스템 카탈로그 테이블에서 특정 데이터베이스를 쿼리하고 모든 사용자 테이블의 목록을 컴파일합니다. 그런 다음 이들 테이블을 PC/IXF 형식으로 익스포트합니다. PC/IXF 파일은 동일한 시스템의 다른 로컬 DB2 데이터베이스에 임포트 또는 로드할 수 있거나, 다른 워크스테이션 플랫폼으로 변환하고 해당 플랫폼의 DB2 데이터베이스에 임포트 또는 로드할 수 있습니다. 이 도구를 사용할 때 구조화된 유형을 갖는 테이블은 이동되지 않습니다. COPY 모드에서 사용할 때 이 도구는 스키마 중복을 용이하게 합니다.

권한 부여

이 도구는 사용자가 요청한 조치에 따라서 DB2 export, import 및 load API를 호출합니다. 그러므로 요청하는 사용자 ID가 해당 API에서 필요한 올바른 권한 부여를 가져야 하며, 그렇지 않으면 요청은 실패합니다.

명령 구문



명령 매개변수

dbname

데이터베이스 이름

action 다음 중 하나여야 합니다.

EXPORT

options의 필터링 기준을 만족하는 모든 테이블을 익스포트합니다. options가 지정되지 않으면 모든 테이블을 익스포트합니다. 내부 스테이징 정보는 db2move.lst 파일에 저장됩니다.

IMPORT

내부 스테이징 파일 db2move.lst에 나열되는 모든 테이블을 임포트합니다. IMPORT 특정 조치에 대해서는 -io 옵션을 사용하십시오.

LOAD

내부 스테이징 파일 db2move.lst에 나열되는 모든 테이블을 로드합니다. LOAD 특정 조치에 대해서는 -lo 명령을 사용하십시오.

COPY

스키마를 목표 데이터베이스에 중복합니다. -sn 옵션을 사용하여 하나 이상의 스키마를 지정하십시오. COPY 특정 옵션에 대해서는 -co 옵션을 참조하십시오. LOAD_ONLY 모드에서 테이블을 필터링하려면 -tn 또는 -tf 옵션을 사용하십시오.

각 조치 중에 생성되는 파일 목록은 아래를 참조하십시오.

-tc *table-definers*

디폴트는 모든 정의자입니다.

이것은 EXPORT 조치 전용입니다. 지정되는 경우 이 옵션과 함께 나열되는 정의자가 작성한 테이블만 익스포트됩니다. 지정되지 않는 경우 디폴트는 모든 정의자를 사용하는 것입니다. 다중 정의자를 지정할 때 정의자를 쉼표로 분리해

야 하며 정의자 ID 사이에 공백은 허용되지 않습니다. 이 옵션을 -tn *table-names* 옵션과 함께 사용하여 익스포트할 테이블을 선택할 수 있습니다.

별표(*)를 문자열의 어디에나 배치될 수 있는 와일드 카드로 사용할 수 있습니다.

-tn *table-names*

디폴트는 모든 사용자 테이블입니다.

이것은 EXPORT 또는 COPY 조치 전용입니다.

EXPORT 옵션과 함께 지정하면 지정된 문자열에 있는 이름과 이름이 일치하는 테이블만 익스포트됩니다. 지정되지 않는 경우 디폴트는 모든 사용자 테이블을 사용하는 것입니다. 다중 테이블 이름을 지정할 때 이름을 쉼표로 분리해야 하며 테이블 이름 사이에 공백은 허용되지 않습니다. 테이블 이름은 규정되지 않은 채로 나열되어야 하며 -sn 옵션을 사용하여 스키마를 필터링해야 합니다.

익스포트의 경우 별표(*)를 문자열의 어디에나 배치될 수 있는 와일드 카드로 사용할 수 있습니다.

COPY 조치와 함께 지정되는 경우 -co ″MODE″ LOAD_ONLY *copy-option* 도 지정해야 하며, 지정된 테이블만 목표 데이터베이스에서 다시 채워집니다. 테이블 이름은 ″schema″.″table″ 형식으로 스키마 규정자와 함께 나열되어야 합니다.

-sn *schema-names*

EXPORT의 디폴트는 모든 스키마입니다(COPY의 경우는 아님).

지정되는 경우 스키마 이름이 일치하는 테이블만 익스포트 또는 복사됩니다. 다중 스키마 이름이 지정되는 경우 쉼표로 구분해야 하며 스키마 이름 사이에 공백은 허용되지 않습니다. 8문자 미만의 스키마 이름은 8문자까지 채워집니다.

익스포트의 경우, 별표 와일드 카드 문자(*)가 스키마 이름에서 사용되는 경우 퍼센트 기호(%)로 변경되며 테이블 이름(퍼센트 기호를 갖는)이 WHERE절의 LIKE 술어에서 사용됩니다. 지정되지 않는 경우 디폴트는 모든 스키마를 사용하는 것입니다. -tn 또는 -tc 옵션과 함께 사용되는 경우 db2move는 스키마가 지정된 스키마 이름과 일치하고 정의자가 지정된 정의자와 일치하는 테이블에만 작용합니다. 별표를 사용할 때 스키마 이름 `fred`에 `-sn fr*d` 대신 `-sn fr*d*`가 지정되어야 합니다.

-ts *tablespace-names*

디폴트는 모든 테이블 스페이스입니다.

이것은 EXPORT 조치 전용입니다. 이 옵션이 지정되면 지정된 테이블 스페이스에 있는 테이블만 익스포트됩니다. 별표 와일드 카드 문자(*)가 테이블 스페이스 이름에서 사용되는 경우 퍼센트 기호(%)로 변경되며 테이블 이름(퍼센트

기호를 갖는)이 WHERE절의 LIKE 술어에서 사용됩니다. -ts 옵션이 지정되지 않는 경우 디폴트는 모든 테이블 스페이스를 사용하는 것입니다. 다중 테이블 스페이스 이름이 지정되는 경우 쉼표로 구분해야 하며 테이블 스페이스 이름 사이에 공백은 허용되지 않습니다. 8문자 미만의 테이블 스페이스 이름은 8문자까지 채워집니다. 예를 들어 테이블 스페이스 이름 mytb는 별표를 사용할 때 -sn my*b 대신 -ts my*b*로 지정되어야 합니다.

-tf *filename*

EXPORT 옵션과 함께 지정하면 이름이 지정된 파일에 있는 이름과 일치하는 테이블만 익스포트됩니다. 지정되지 않는 경우 디폴트는 모든 사용자 테이블을 사용하는 것입니다. 테이블은 라인당 하나씩 나열되어야 하며 각 테이블은 완전해야 합니다. 문자열에서 와일드 카드 문자는 허용되지 않습니다. 다음은 파일 컨텐츠의 예입니다.

```
"SCHEMA1"."TABLE NAME1"
"SCHEMA NAME77"."TABLE155"
```

COPY 조치와 함께 지정되는 경우 -co ″MODE″ LOAD_ONLY *copy-option* 도 지정해야 하며, 파일에서 지정된 테이블만 목표 데이터베이스에서 다시 채워집니다. 테이블 이름은 ″schema″.″table″ 형식으로 스키마 규정자와 함께 나열되어야 합니다.

-io *import-option*

디폴트는 REPLACE_CREATE입니다. 임포트 작성 기능의 제한사항은 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE는 사용되지 않음』을 참조하십시오.

유효한 옵션은 INSERT, INSERT_UPDATE, REPLACE, CREATE 및 REPLACE_CREATE입니다.

-lo *load-option*

디폴트는 INSERT입니다.

유효한 옵션은 INSERT와 REPLACE입니다.

-co db2move 조치가 COPY일 때, 다음 -co 후속 옵션을 사용할 수 있습니다.

“TARGET_DB *db name* **[USER** *userid* **USING** *password***]”**

사용자가 목표 데이터베이스의 이름 및 사용자/암호를 지정할 수 있습니다. (소스 데이터베이스 사용자/암호는 기존 -p 및 -u 옵션을 사용하여 지정할 수 있습니다.) USER/USING절은 선택적입니다. USER가 사용자 ID를 지정하는 경우 USING절 뒤에서 암호가 제공되거나, 지정되지 않는 경우 db2move가 암호 정보를 위해 프롬프트합니다. 프롬프트 이유는 아래에서 설명하는 보안 이유 때문입니다. TARGET_DB는 COPY 조치에 대한 필수 옵션입니다. TARGET_DB는 소스 데이터베이스와 같을 수 없습니다. ADMIN_COPY_SCHEMA 프로시저를 동일한 데

이터베이스에서 스키마 복사에 사용할 수 있습니다. COPY 조치는 최소한 하나의 스키마(-sn) 또는 한 테이블(-tn 또는 -tf) 입력이 필요합니다.

다중 db2move 명령을 실행하여 한 데이터베이스에서 다른 데이터베이스로 스키마를 복사하면 교착 상태가 발생합니다. 한 번에 하나의 db2move 명령만 발행해야 합니다. 복사 처리 중에 소스 스키마의 테이블 변경은 목표 스키마의 데이터가 복사 후에 동일하지 않음을 의미할 수 있습니다.

"MODE"

DDL_AND_LOAD
소스 스키마로부터 지원되는 모든 오브젝트를 작성하고 테이블을 소스 테이블 데이터로 채웁니다. 디폴트 옵션입니다.

DDL_ONLY
소스 스키마로부터 지원되는 모든 오브젝트를 작성하지만 테이블을 다시 채우지 않습니다.

LOAD_ONLY
소스 데이터베이스에서 목표 데이터베이스로 모든 지정된 테이블을 로드합니다. 테이블은 이미 목표에 존재해야 합니다. LOAD_ONLY 모드는 -tn 또는 -tf 옵션을 사용하여 최소 하나의 테이블을 입력해야 합니다.

이것은 COPY 조치에서만 사용되는 선택적 옵션입니다.

"SCHEMA_MAP"

사용자가 목표에 복사할 때 스키마 이름을 바꿀 수 있습니다. 소스-목표 스키마 맵핑 목록을 쉼표로 구분하고 대괄호로 묶어서 제공합니다. 예: schema_map ((s1, t1), (s2, t2)). 이것은 스키마 s1의 오브젝트가 목표의 스키마 t1에 복사되며 스키마 s2의 오브젝트는 목표의 스키마 t2에 복사됨을 의미합니다. 디폴트이며 권장되는 목표 스키마 이름은 소스 스키마 이름입니다. 이유는 db2move는 오브젝트 본문 내의 모든 자격이 부여된 오브젝트에 대한 스키마를 수정하려고 하지 않기 때문입니다. 그러므로 다른 목표 스키마 이름을 사용하면 오브젝트 본문 안에 자격이 부여된 오브젝트가 있는 경우 문제가 발생할 수 있습니다.

예를 들면, 다음과 같습니다.

```
create view FOO.v1 as 'select c1 from FOO.t1'
```

이 경우 BAR로 스키마 FOO를 복사하면 다음과 같이 v1이 다시 생성됩니다.

```
create view BAR.v1 as 'select c1 from FOO.t1'
```

이것은 스키마 FOO가 목표 데이터베이스에 존재하지 않으므로 실패하거나 FOO가 BAR과 다름으로 인해 예기치 않은 결과를 생성할 수 있습니다. 소스와 동일한 스키마 이름을 유지하면 이 문제를 피할 수 있습니다. 스키마 사이에 교차 종속성이 있는 경우 모든 상호 종속된 스키마가 복사되어야 하며, 그렇지 않으면 교차 종속성을 갖는 오브젝트를 복사할 때 오류가 발생할 수 있습니다.

예를 들면, 다음과 같습니다.

```
create view FOO.v1 as 'select c1 from BAR.t1'
```

이 경우 v1의 복사는 BAR도 복사되지 않는 경우 실패하거나, 목표의 BAR이 소스의 BAR과 다른 경우 예기치 않은 결과를 생성합니다. db2move는 교차 스키마 종속성을 발견하려고 시도하지 않습니다.

이것은 COPY 조치에서만 사용되는 선택적 옵션입니다.

"NONRECOVERABLE"

이 옵션을 사용하면 사용자가 COPY-NO로 수행되는 로드의 디폴트 동작을 겹쳐쓸 수 있습니다. 디폴트 동작을 사용하면 사용자는 로드된 각 테이블 스페이스의 백업을 작성해야 합니다. 이 NONRECOVERABLE 키워드를 지정하면 사용자는 테이블 스페이스의 백업을 즉시 작성하지 않아도 됩니다. 그러나 새로 작성되는 테이블이 제대로 복구 가능하도록 가능한 빨리 백업을 작성하는 것이 매우 바람직합니다. 이것은 COPY 조치에 사용 가능한 선택적 옵션입니다.

"OWNER"

사용자가 성공적인 COPY 후에 목표 스키마에 작성된 각 새 오브젝트의 소유자를 변경할 수 있습니다. 목표 오브젝트의 디폴트 소유자는 연결된 사용자입니다. 이 옵션이 지정되면 소유권이 새 소유자에게 이전됩니다. 이것은 COPY 조치에 사용 가능한 선택적 옵션입니다.

"TABLESPACE_MAP"

사용자는 복사 중에 소스 시스템의 테이블 스페이스 대신 사용될 테이블 스페이스 이름 맵핑을 지정할 수 있습니다. 이것은 대괄호로 묶은 테이블 스페이스 맵핑의 배열입니다. 예: `tablespace_map ((TS1, TS2),(TS3, TS4))`. 이것은 테이블 스페이스 TS1의 모든 오브젝트가 목표 데이터베이스의 테이블 스페이스 TS2에 복사되며 테이블 스페이스 TS3의 오브젝트는 목표의 테이블 스페이스 TS4에 복사됨을 의미합니다. `((T1, T2),(T2, T3))`의 경우, 소스 데이터베이스의 T1에 있는 모든 오브젝트는 목표 데이터베이스의 T2에서 재작성되며 소스 데이터베이스의 T2에서 발견되는 모든 오브젝트는 목표 데이터베이스의 T3에 재작성됩니다. 디폴트는 소스에서와 동일한 테이블 스페이스 이름을 사용하는 것이며, 이 경우 이 테이블 스페이스에 대한 입력 맵핑

은 필요하지 않습니다. 지정된 테이블 스페이스가 존재하지 않는 경우 해당 테이블 스페이스를 사용하는 오브젝트의 복사는 실패하며 오류 파일에 로그됩니다.

사용자는 또한 SYS_ANY 키워드를 사용하여 목표 테이블 스페이스가 디폴트 테이블 스페이스 선택 알고리즘을 사용하여 선택되어야 함을 표시하는 옵션도 있습니다. 이 경우 db2move는 목표로 사용할 수 있는 모든 사용 가능한 테이블 스페이스를 선택할 수 있습니다. SYS_ANY 키워드를 모든 테이블 스페이스에 사용할 수 있습니다(예: tablespace_map SYS_ANY). 또한 사용자는 일부 테이블 스페이스에 대해 특정 맵핑을 지정하고 나머지에 대해서는 디폴트 테이블 스페이스 선택 알고리즘을 지정할 수 있습니다. 예: tablespace_map ((TS1, TS2),(TS3, TS4), SYS_ANY). 이것은 테이블 스페이스 TS1이 TS2로 맵핑되고, TS3은 TS4에 맵핑되지만 나머지 테이블 스페이스는 디폴트 테이블 스페이스 목표를 사용함을 표시합니다. "SYS"로 시작하는 테이블 스페이스를 가질 수 없으므로 SYS_ANY 키워드가 사용되고 있습니다.

이것은 COPY 조치에 사용 가능한 선택적 옵션입니다.

-l *lobpaths*

IMPORT 및 EXPORT의 경우 이 옵션이 지정되면 XML 경로에도 사용됩니다. 디폴트는 현재 디렉토리입니다.

이 옵션은 LOB 또는 XML 파일이 작성되거나(EXPORT의 파트로서) 검색되는(IMPORT 또는 LOAD의 파트로서) 절대 경로 이름을 지정합니다. 다중 경로를 지정할 때 각 경로는 쉼표로 분리되어야 하며 경로 사이에 공백은 허용되지 않습니다. 다중 경로가 지정되는 경우 EXPORT는 경로를 라운드 로빈 방식으로 사용합니다. 하나의 LOB 문서를 첫 번째 경로에 쓰고, 하나를 두 번째 경로에 쓰고, 이런 식으로 마지막까지 쓴 후 다시 첫 번째 경로로 돌아옵니다. XML 문서의 경우도 동일합니다. 파일이 첫 번째 경로에 없는 경우(IMPORT 또는 LOAD 중에) 두 번째 경로가 사용되는 방식입니다.

-u *userid*

디폴트는 로그온한 사용자 ID입니다.

사용자 ID 및 암호가 둘 다 선택적입니다. 그러나 하나가 지정되면 다른 것도 지정해야 합니다. 명령이 리모트 서버에 연결된 클라이언트에서 실행되는 경우 사용자 ID와 암호를 지정해야 합니다.

-p *password*

디폴트는 로그온된 암호입니다. 사용자 ID 및 암호가 둘 다 선택적입니다. 그러나 하나가 지정되면 다른 것도 지정해야 합니다. -p 옵션이 지정되지만 암호가 제공되지 않는 경우 db2move는 암호를 위해 프롬프트합니다. 이것은 보안상의 이유 때문입니다. 명령행을 통해 암호를 입력하면 보안 문제가 발생합니

다. 예를 들어 ps -ef 명령은 암호를 표시합니다. 그러나 db2move가 스크립트를 통해 호출되면 암호를 제공해야 합니다. 명령이 리모트 서버에 연결된 클라이언트에서 실행되는 경우 사용자 ID와 암호를 지정해야 합니다.

-aw 경고를 허용합니다. -aw가 지정되지 않으면 익스포트 중에 경고가 발생하는 테이블은(테이블의 .ixf 파일 및 .msg 파일은 여전히 생성되지만) db2move.lst 파일에 포함되지 않습니다. 일부 시나리오(예: 데이터 절단)에서는 사용자가 그런 테이블이 db2move.lst 파일에 포함되기 원할 수 있습니다. 이 옵션을 지정하면 익스포트 중에 경고를 수신하는 테이블이 .lst 파일에 포함될 수 있습니다.

예:

- SAMPLE 데이터베이스의 모든 테이블을(모든 옵션에 대해 디폴트값을 사용하여) 익스포트하려면 다음 명령을 발행하십시오.

```
db2move sample export
```

- userid1 또는 사용자 ID LIKE us%rid2가 작성하고 이름이 tbname1이거나 테이블 이름 LIKE %tbname2를 갖는 모든 테이블을 익스포트하려면 다음 명령을 발행하십시오.

```
db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2
```

- SAMPLE 데이터베이스의 모든 테이블을 임포트하려면(LOB 경로 D:₩LOBPATH1 및 C:₩LOBPATH2가 LOB 파일에 대해 검색되며, 이 예는 Windows 운영 체제에만 적용할 수 있음), 다음 명령을 발행하십시오.

```
db2move sample import -l D:₩LOBPATH1,C:₩LOBPATH2
```

- SAMPLE 데이터베이스의 모든 테이블을 로드하려면(/home/userid/lobpath 서브디렉토리 및 tmp 서브디렉토리에서 LOB 파일을 검색하며, 이 예는 Linux 및 UNIX 시스템에만 적용할 수 있음) 다음 명령을 발행하십시오.

```
db2move sample load -l /home/userid/lobpath,/tmp
```

- REPLACE 모드에서 지정된 사용자 ID와 암호를 사용하여 SAMPLE 데이터베이스의 모든 테이블을 임포트하려면 다음 명령을 발행하십시오.

```
db2move sample import -io replace -u userid -p password
```

- 소스 데이터베이스 dbsrc에서 목표 데이터베이스 dbtgt로 schema1 스키마를 중복하려면 다음 명령을 발행하십시오.

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
```

- schema1 스키마를 소스 데이터베이스 dbsrc에서 목표 데이터베이스 dbtgt로 중복하고, 목표에서 스키마 이름을 newschema1로 변경하고 소스 테이블 스페이스 ts1을 목표의 ts2에 맵핑하려면 다음 명령을 발행하십시오.

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
       SCHEMA_MAP ((schema1,newschema1)) TABLESPACE_MAP ((ts1,ts2), SYS_ANY))
```

사용 시 참고사항

- db2move IMPORT/LOAD가 뒤따르는 db2move EXPORT는 테이블 데이터의 이동을 용이하게 합니다. 이들 테이블이 종속될 수 있는 오브젝트(예: 사용자 정의 유형 또는 사용자 정의 함수(UDF))뿐 아니라 테이블과 연관된 다른 모든 데이터베이스 오브젝트(예: 별명, 뷰 또는 트리거)를 수동으로 이동해야 합니다.
- CREATE 또는 REPLACE_CREATE 옵션을 갖는 IMPORT 조치를 사용하여 목표 데이터베이스에 테이블을 작성하는 경우(두 옵션은 모두 사용되지 않으며 추후 릴리스에서 제거될 수 있음), 『임포트한 테이블 재작성』에서 설명하는 제한사항이 적용됩니다. REPLACE_CREATE 옵션이 사용될 때 db2move 임포트 단계 중에 예기치 않은 오류가 발생하는 경우, 적합한 `tabnnn.msg` 메시지 파일을 조사하고 오류가 테이블 작성에 대한 제한사항의 결과일 수 있는지 고려하십시오.
- GENERATED ALWAYS ID 컬럼이 들어있는 테이블은 db2move를 사용하여 임포트하거나 로드할 수 없습니다. 그러나 이들 테이블을 수동으로 임포트 또는 로드할 수 있습니다. 자세한 정보는 『ID 컬럼 로드 고려사항』 또는 『ID 컬럼 임포트 고려사항』을 참조하십시오.
- db2move가 export, import 또는 load API를 호출할 때 **FileTypeMod** 매개변수는 lobsinfile로 설정됩니다. 즉, LOB 데이터는 모든 테이블에 대해 PC/IXF 파일과는 별개인 파일에 보존됩니다.
- LOAD 명령은 데이터베이스 및 데이터 파일이 있는 머신에서 로컬로 실행되어야 합니다.
- db2move LOAD를 사용하고 데이터베이스에 대해 logretain이 사용될 때(데이터베이스가 복구 가능함),
  - NONRECOVERABLE 옵션이 지정되지 않으면 db2move는 디폴트 COPY NO 옵션을 사용하여 db2Load API를 호출하는 경우, 로드된 테이블이 있는 테이블 스페이스는 유틸리티 완료 시에 백업 보류 상태에 있습니다(테이블 스페이스를 백업 보류 상태에서 벗어나게 하려면 전체 데이터베이스 또는 테이블 스페이스 백업이 필요함).
  - NONRECOVERABLE 옵션이 지정되면 테이블 스페이스가 백업 보류 상태에 들어가지 않지만, 나중에 롤 포워드 복구가 수행되는 경우 테이블은 액세스 불가능으로 표시되며 삭제해야 합니다. 로드 복구 가능성 옵션에 대한 자세한 정보는 『로드 성능 개선을 위한 옵션』을 참조하십시오.
- IMPORT 또는 LOAD 옵션을 갖는 db2move 명령의 성능은 디폴트 버퍼 풀 IBMDEFAULTBP를 변경하고 구성 매개변수 **sortheap**, **util_heap_sz**, **logfilsiz** 및 **logprimary**를 갱신하여 개선될 수 있습니다.

EXPORT를 사용할 때 필요한/생성되는 파일:

- 입력: 없음
- 출력:

EXPORT.out

EXPORT 조치의 요약 결과.

db2move.lst

원래 테이블 이름, 대응하는 PC/IXF 파일 이름(tabnnn.ixf) 및 메시지 파일 이름(tabnnn.msg)의 목록입니다. 이 목록, 익스포트된 PC/IXF 파일 및 LOB 파일(tabnnnc.yyy)이 db2move IMPORT 또는 LOAD 조치의 입력으로 사용됩니다.

tabnnn.ixf

특정 테이블의 익스포트된 PC/IXF 파일.

tabnnn.msg

대응하는 테이블의 익스포트 메시지 파일.

tabnnnc.yyy

특정 테이블의 익스포트된 LOB 파일.

『nnn』은 테이블 번호입니다. 『c』는 영문자의 한 문자입니다. 『yyy』는 001 - 999 사이의 숫자입니다.

이들 파일은 익스포트되는 테이블에 LOB 데이터가 있는 경우에만 작성됩니다. 작성되는 경우 이들 LOB 파일은 『lobpath』 디렉토리에 위치합니다. LOB 파일에 대해 총 26,000개의 가능한 이름이 있습니다.

system.msg

파일 또는 디렉토리 작성 또는 삭제 명령에 대한 시스템 메시지가 들어있는 메시지 파일입니다. 이것은 조치가 EXPORT이고 LOB 경로가 지정되는 경우에만 사용됩니다.

IMPORT를 사용할 때 필요한/생성되는 파일:

- 입력:

  db2move.lst

  EXPORT 조치의 출력 파일.

  tabnnn.ixf

  EXPORT 조치의 출력 파일.

  tabnnnc.yyy

  EXPORT 조치의 출력 파일.

- 출력:

  IMPORT.out

  IMPORT 조치의 요약 결과.

  tabnnn.msg

  대응하는 테이블의 임포트 메시지 파일입니다.

LOAD를 사용할 때 필요한/생성되는 파일:

- 입력:

  db2move.lst
  EXPORT 조치의 출력 파일.

  tabnnn.ixf
  EXPORT 조치의 출력 파일.

  tabnnnc.yyy
  EXPORT 조치의 출력 파일.

- 출력:

  LOAD.out
  LOAD 조치의 요약 결과.

  tabnnn.msg
  대응하는 테이블의 LOAD 메시지 파일.

COPY를 사용할 때 필요한/생성되는 파일:

- 입력: 없음
- 출력:

  COPYSCHEMA.msg
  COPY 작업 중에 생성되는 메시지가 들어있는 출력 파일.

  COPYSCHEMA.err
  목표 데이터베이스에서 재작성될 수 없는 각 오브젝트에 대한 DDL문을 포함하여 COPY 작업 중에 발생하는 각 오류에 대한 오류 메시지가 들어있는 출력 파일.

  LOADTABLE.msg
  로드 유틸리티(목표 데이터베이스에서 데이터를 다시 채우는 데 사용되는)의 각 호출에 의해 생성되는 메시지가 들어있는 출력 파일.

  LOADTABLE.err
  로드 중에 실패했거나 목표 데이터베이스에서 채워져야 하는 테이블의 이름이 들어있는 출력 파일입니다. 자세한 내용은 『실패한 스키마 복사 조작 재시작』 주제를 참조하십시오.

이들 파일은 시간소인이 추가되며 한 번의 실행에서 생성되는 모든 파일은 동일한 시간소인을 갖습니다.

자동으로 생성된 스크립트를 사용하여 경로 재지정된 리스토어 수행

경로 재지정된 리스토어 작업을 수행할 때 백업 이미지에 저장된 실제 컨테이너의 위치를 지정하고 변경될 각 테이블 스페이스에 대한 컨테이너의 전체 세트를 제공해야 합니다. 다음 프로시저를 사용하여 기존 백업 이미지를 기초로 경로 재지정된 리스토어 스크립트를 생성하고, 생성된 스크립트를 수정한 후 스크립트를 실행하여 경로 재지정된 리스토어를 수행하십시오.

데이터베이스가 이전에 DB2 백업 유틸리티를 사용하여 백업된 경우에만 경로 재지정 리스토어를 수행할 수 있습니다.

- 데이터베이스가 존재하는 경우 스크립트를 생성하려면 데이터베이스에 연결할 수 있어야 합니다. 그러므로 데이터베이스가 업그레이드 또는 응급 복구가 필요한 경우 이것은 경로 재지정 리스토어 스크립트를 생성하려고 시도하기 전에 수행되어야 합니다.
- 파티션된 데이터베이스 환경에서 작업 중이고 목표 데이터베이스가 존재하지 않는 경우, 모든 데이터베이스 파티션에서 동시에 경로 재지정 리스토어 스크립트를 생성하는 명령을 실행할 수 없습니다. 대신 경로 재지정 리스토어 스크립트를 생성하는 명령은 카탈로그 파티션에서 시작하여 한 번에 하나의 데이터베이스 파티션에서 실행되어야 합니다.

  또는 먼저 목표 데이터베이스와 동일한 이름이 있는 더미 데이터베이스를 작성할 수 있습니다. 더미 데이터베이스가 작성된 후 모든 데이터베이스 파티션에서 동시에 경로 재지정 리스토어 스크립트를 생성할 수 있습니다.
- RESTORE 명령을 사용하여 스크립트를 생성할 때 REPLACE EXISTING 옵션을 지정하는 경우에도 주석 처리된 스크립트에 REPLACE EXISTING 옵션이 나타납니다.
- 보안상의 이유 때문에 암호가 생성된 스크립트에 나타나지 않습니다. 암호를 수동으로 채워야 합니다.
- 제어 센터의 리스토어 마법사를 사용하여 경로 재지정된 리스토어에 대한 스크립트를 생성할 수 없습니다.

스크립트를 사용하여 경로 재지정 리스토어를 수행하려면 다음을 수행하십시오.

1. 리스토어 유틸리티를 사용하여 경로 재지정 리스토어 스크립트를 생성하십시오. 리스토어 유틸리티는 명령행 처리기(CLP) 또는 db2Restore API를 통해 호출할 수 있습니다. 다음은 REDIRECT 옵션 및 GENERATE SCRIPT 옵션을 사용하는 RESTORE DATABASE 명령의 예입니다.

```
db2 restore db test from /home/jseifert/backups taken at 20050304090733
         redirect generate script test_node0000.clp
```

이것은 test_node0000.clp라는 클라이언트에 경로 재지정 리스토어 스크립트를 작성합니다.

2. 텍스트 편집기에서 경로 재지정 리스토어 스크립트를 열어서 필요한 수정을 수행하십시오. 다음을 수정할 수 있습니다.
   - 리스토어 옵션
   - 자동 스토리지 경로
   - 컨테이너 레이아웃 및 경로
3. 수정된 경로 재지정 리스토어 스크립트를 실행하십시오. 예를 들면, 다음과 같습니다.

   ```
   db2 -tvf test_node0000.clp
   ```

RESTORE DATABASE

RESTORE DATABASE 명령은 DB2 백업 유틸리티를 사용하여 백업한 손상된 데이터베이스를 다시 작성합니다. 리스토어된 데이터베이스는 백업 사본이 작성될 때의 상태와 동일한 상태가 됩니다. 이 유틸리티는 다른 이미지로 데이터베이스를 겹쳐쓸 수 있고 백업 사본을 새 데이터베이스로 리스토어할 수도 있습니다.

다양한 운영 체제 및 하드웨어 플랫폼 사이에 DB2 데이터베이스 시스템에서 지원되는 리스토어 조작에 관한 정보는 *Data Recovery and High Availability Guide and Reference*에서 『Backup and restore operations between different operating systems and hardware platforms』를 참조하십시오.

리스토어 유틸리티를 사용하여 DB2 Universal Database 버전 8, DB2 버전 9.1 또는 DB2 버전 9.5에서 백업한 백업 이미지를 DB2 버전 9.7에서 리스토어할 수도 있습니다. 데이터베이스 업그레이드가 필요한 경우 리스토어 조작 종료 시 자동으로 호출됩니다.

백업 작업 시 데이터베이스를 롤 포워드 복구에 사용할 수 있는 경우 리스토어 조작을 완료한 후 롤 포워드 유틸리티를 호출하여 데이터베이스를 이전 상태로 가져올 수 있습니다.

이 유틸리티는 또한 테이블 스페이스 레벨 백업을 리스토어할 수도 있습니다.

증분 이미지와 이전 캡처에서 차이점만 캡처하는 이미지(『델타 이미지』)는 운영 체제나 단어 크기(32비트 또는 64비트)가 다를 때 리스토어할 수 없습니다.

하나의 환경에서 다른 환경으로의 성공적인 리스토어 조작 후에는 증분이 아닌 백업을 가져올 때까지 증분 또는 델타 백업이 허용되지 않습니다. (이는 동일 환경에서의 리스토어 조작 다음의 제한사항이 아닙니다.)

하나의 환경에서 다른 환경으로의 리스토어 조작이 성공해도, 몇 가지 고려할 사항이 있습니다. 패키지는 사용 이전에 리바인드해야 합니다(BIND 명령, REBIND 명령 또는 db2rbind 유틸리티를 사용하여). SQL 프로시저는 삭제하고 다시 작성해야 합니다. 그리고 모든 외부 라이브러리는 새 플랫폼에서 재빌드해야 합니다. (동일한 환경으로 리스토어할 경우에는 이 사항을 고려하지 않아도 됩니다.)

기존 데이터베이스와 기존 컨테이너에 대해 실행되는 리스토어 조작은 동일한 컨테이너 및 테이블 스페이스 맵을 재사용합니다.

새 데이터베이스에 대해 실행되는 리스토어 조작은 모든 컨테이너를 다시 획득하고 최적화된 테이블 스페이스 맵을 재빌드합니다. 새 데이터베이스에 대해 실행되는 리스토어 조작은 모든 컨테이너를 다시 획득하고 최적화된 테이블 스페이스 맵을 재빌드합니다.

범위

이 명령은 명령이 실행되는 노드에만 영향을 미칩니다.

권한 부여

기존 데이터베이스를 리스토어하려면 다음 중 하나가 있어야 합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*

새 데이터베이스로 리스토어하려면 다음 중 하나가 있어야 합니다.

- *sysadm*
- *sysctrl*

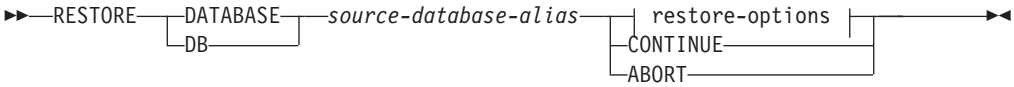
필수 연결

필수 연결은 리스토어 조치 유형에 따라 다릅니다.

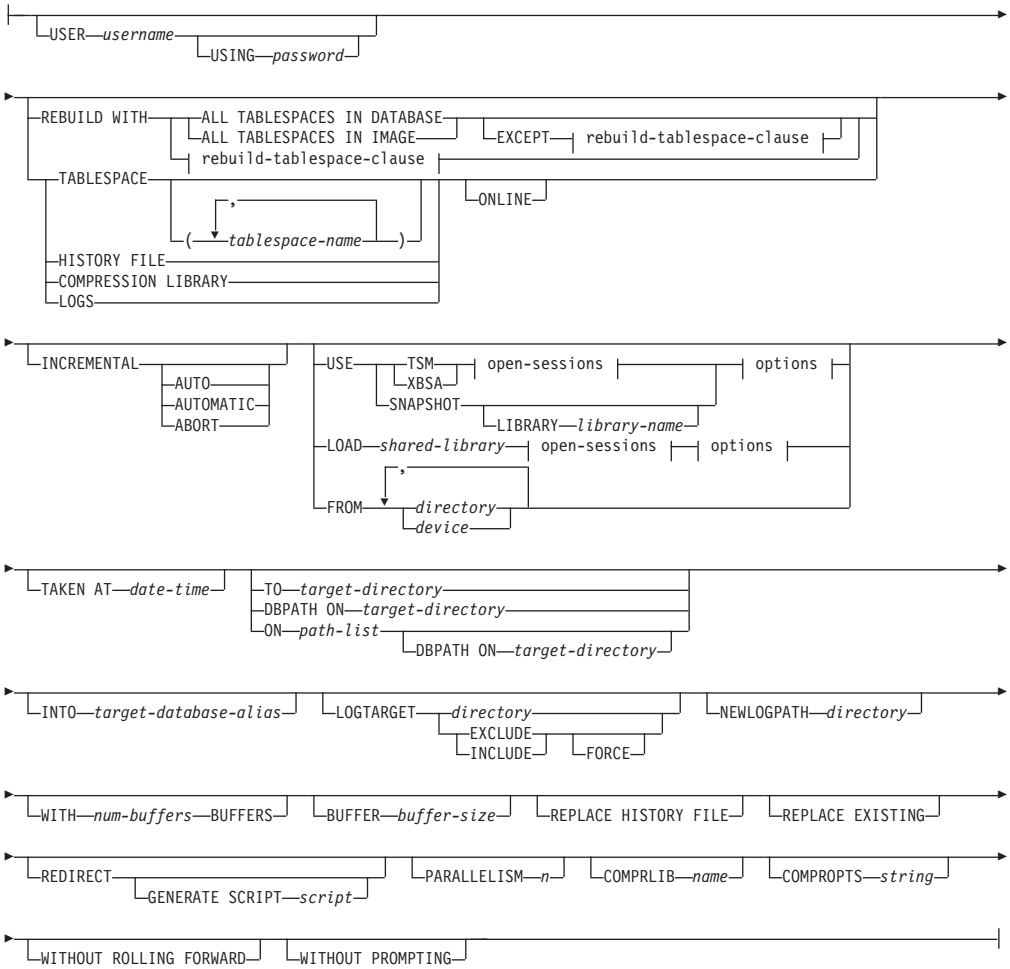
- 기존 데이터베이스로 리스토어하려면 데이터베이스에 연결해야 합니다. 이 명령은 지정된 데이터베이스에 대한 독점 연결을 자동으로 설정합니다.
- 새 데이터베이스로 리스토어하려면 인스턴스 및 데이터베이스에 연결해야 합니다. 데이터베이스를 작성하려면 인스턴스에 접속해야 합니다.

  현재 인스턴스와 다른 인스턴스에서 새 데이터베이스로 리스토어하려면, 먼저 새 데이터베이스가 상주할 인스턴스에 접속해야 합니다. 새 인스턴스는 로컬 또는 리모트 상태일 수 있습니다. 현재 인스턴스는 DB2INSTANCE 환경 변수 값으로 정의됩니다.
- 스냅샷 리스토어의 경우 *인스턴스* 및 *데이터베이스*에 연결해야 합니다.

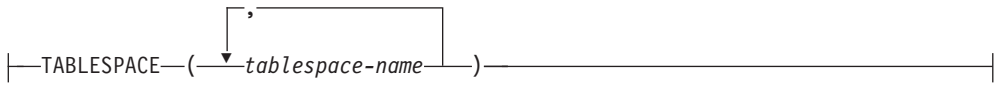
명령 구문



restore-options:



rebuild-tablespace-clause:



open-sessions:

옵션:

명령 매개변수

DATABASE *source-database-alias*

백업을 가져온 소스 데이터베이스의 별명.

CONTINUE

컨테이너가 재정의되었으며, 경로 재지정된 리스토어 조작의 최종 단계를 수행해야 함을 지정합니다.

ABORT

이 매개변수는

- 경로 재지정된 리스토어 조작을 중지합니다. 이는 하나 이상의 단계를 반복해야 하는 오류가 발생할 때 유용합니다. ABORT 옵션이 있는 RESTORE DATABASE를 발행하면, REDIRECT 옵션이 있는 RESTORE DATABASE를 포함하여 경로 재지정된 리스토어 조작의 각 단계를 반복해야 합니다.
- 완료 이전에 증분 리스토어 조작을 종료합니다.

USER *username*

데이터베이스가 리스토어되는 사용자 이름을 식별합니다.

USING *password*

사용자 이름을 인증하는 데 사용되는 암호. 암호를 생략하면 사용자가 이를 입력하도록 프롬프트가 표시됩니다.

REBUILD WITH ALL TABLESPACES IN DATABASE

이미지가 리스토어될 때 데이터베이스에 알려진 모든 테이블 스페이스와 함께 데이터베이스를 리스토어합니다. 이 리스토어는 데이터베이스가 이미 존재하는 경우 겹쳐씁니다.

REBUILD WITH ALL TABLESPACES IN DATABASE EXCEPT *rebuild-tablespace-clause*

목록에 지정된 테이블 스페이스를 제외하고, 이미지가 리스토어될 때 데이터베이스에 알려진 모든 테이블 스페이스와 함께 데이터베이스를 리스토어합니다. 이 리스토어는 데이터베이스가 이미 존재하는 경우 겹쳐씁니다.

REBUILD WITH ALL TABLESPACES IN IMAGE

리스토어 중인 이미지의 테이블 스페이스만 데이터베이스와 함께 리스토어합니다. 이 리스토어는 데이터베이스가 이미 존재하는 경우 겹쳐씁니다.

REBUILD WITH ALL TABLESPACES IN IMAGE EXCEPT *rebuild-tablespace-clause*

목록에 지정된 테이블 스페이스를 제외하고, 리스토어 중인 테이블 스페이스만 데이터베이스와 함께 리스토어합니다. 이 리스토어는 데이터베이스가 이미 존재하는 경우 겹쳐씁니다.

REBUILD WITH *rebuild-tablespace-clause*

지정된 테이블 스페이스 목록만 데이터베이스와 함께 리스토어합니다. 이 리스토어는 데이터베이스가 이미 존재하는 경우 겹쳐씁니다.

TABLESPACE *tablespace-name*

리스토어될 테이블 스페이스를 지정하는 데 사용되는 이름 목록.

ONLINE

이 키워드는 테이블 스페이스 레벨 리스토어 조작을 수행하는 경우에만 적용되며, 백업 이미지가 온라인에서 리스토어될 수 있도록 허용하기 위해 지정합니다. 이는 백업 이미지가 리스토어되는 동안 다른 에이전트가 데이터베이스에 연결할 수 있으며, 지정된 테이블 스페이스를 리스토어하는 중에 다른 테이블 스페이스의 데이터를 사용할 수 있음을 의미합니다.

HISTORY FILE

이 키워드는 백업 이미지에서 실행기록 파일만 리스토어할 때 지정됩니다.

COMPRESSION LIBRARY

이 키워드는 백업 이미지에서 압축 라이브러리만 리스토어할 때 지정됩니다. 오브젝트가 백업 이미지에 있는 경우 데이터베이스 디렉토리로 리스토어됩니다. 오브젝트가 백업 이미지에 없는 경우 리스토어 조작이 실패합니다.

LOGS

이 키워드는 백업 이미지에 포함된 로그 파일 세트만 리스토어할 때 지정됩니다. 백업 이미지에 로그 파일이 없는 경우 리스토어 조작이 실패합니다. 이 옵션을 지정하면, LOGTARGET 옵션도 지정해야 합니다.

INCREMENTAL

추가 매개변수 없이, INCREMENTAL은 수동의 누적된 리스토어 조작을 지정합니다. 수동 리스토어 중, 사용자는 리스토어에 포함되는 이미지마다 수동으로 각각의 리스토어 명령을 발행해야 합니다. 마지막, 첫 번째, 두 번째, 세 번째, 그리고 마지막 이미지까지, 이 순서로 수행해야 합니다.

INCREMENTAL AUTOMATIC/AUTO

자동 누적 리스토어 조작을 지정합니다.

INCREMENTAL ABORT

진행 중인 수동 누적 리스토어 조작의 중단을 지정합니다.

USE

TSM Tivoli Storage Manager에서 관리되는 출력에서 데이터베이스가 리스토어됨을 지정합니다.

XBSA XBSA 인터페이스가 사용되도록 지정합니다. 백업 서비스 API(XBSA)는 백업 또는 아카이브용으로 데이터 스토리지 관리를 필요로 하는 응용프로그램 또는 기능에 대한 개방 API입니다.

SNAPSHOT

스냅샷 백업에서 데이터가 리스토어되도록 지정합니다.

다음 매개변수가 있는 SNAPSHOT 매개변수를 사용할 수 없습니다.

- INCREMENTAL
- TO
- ON
- DBPATH ON
- INTO
- NEWLOGPATH
- WITH *num-buffers* BUFFERS
- BUFFER
- REDIRECT
- REPLACE HISTORY FILE
- COMPRESSION LIBRARY
- PARALLELISM
- COMPRLIB
- OPEN *num-sessions* SESSIONS
- HISTORY FILE
- LOGS

또한 테이블 스페이스 목록을 수반하는 리스토어 조작(REBUILD WITH 옵션이 포함된)에서는 SNAPSHOT 매개변수를 사용할 수 없습니다.

스냅샷 백업 이미지에서 데이터를 리스토어할 때 디폴트 동작은 시간소인이 제공되지 않는 경우 모든 컨테이너, 로컬 볼륨 디렉토리, 데이터베이스 경로(DBPATH), 1차 로그 및 가장 최신 스냅샷 백업의 미러 로그 경로를 포함하여 데이터베이스를 구성하는 모든 경로의 FULL DATABASE OFFLINE 리스토어입니다(INCLUDE LOGS는 EXCLUDE LOGS가 명시적으로 언급된 경우를 제외하고는 모든 스냅샷 백업의 디폴트임). 시간소인이 제공되는 경우에는 스냅샷 백업 이미지가 리스토어됩니다.

LIBRARY *library-name*

IBM Data Server로의 통합은 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage® SAN Volume Controller
- IBM Enterprise Storage Server® Model 800
- IBM System Storage™ DS6000™
- IBM System Storage DS8000®
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

다른 스토리지 하드웨어가 있으며 해당 스토리지 하드웨어의 DB2 ACS API 드라이버가 있는 경우, LIBRARY 매개변수를 사용하여 DB2 ACS API 드라이버를 지정할 수 있습니다.

LIBRARY 매개변수의 값은 완전한 라이브러리 파일 이름입니다.

OPTIONS

"options-string"

리스토어 조작에 사용되는 옵션을 지정합니다. 문자열은 큰따옴표 없이 입력된 그대로 정확히 DB2 ACS API 드라이버로 전달됩니다. **VENDOROPT** 데이터베이스 구성 매개변수를 사용하여 스냅샷 리스토어 조작을 위한 벤더 특정 옵션을 지정할 수 없습니다. 대신 리스토어 유틸리티의 OPTIONS 매개변수를 사용해야 합니다.

@ *file-name*

리스토어 조작에 사용될 옵션이 DB2 서버에 있는 파일에 들어 있음을 지정합니다. 이 문자열은 벤더 지원 라이브러리로 패스됩니다. 파일은 완전한 파일 이름이어야 합니다.

OPEN *num-sessions* **SESSIONS**

TSM 또는 벤더 제품과 함께 사용되는 입출력 세션 수를 지정합니다.

FROM *directory/device*

백업 이미지가 있는 디렉토리 또는 디바이스의 완전한 경로 이름. USE TSM, FROM 및 LOAD가 생략된 경우 디폴트 값은 클라이언트 컴퓨터의 현재 작업 디렉토리입니다. 이 목표 디렉토리 또는 디바이스는 목표 서버/인스턴스에 있어야 합니다.

몇 개의 항목이 지정되고 마지막 항목이 테이프 디바이스이면 사용자가 다른 테이프를 입력하도록 프롬프트됩니다. 유효한 응답은 다음과 같습니다.

c 계속. 경고 메시지를 생성한 디바이스를 사용하여 계속합니다(예를 들어, 새로운 테이프가 마운트된 경우 계속함).

d 디바이스 종료. 경고 메시지를 생성한 디바이스만 사용을 중지합니다(예를 들어, 더 이상 테이프가 없을 때 종료함).

t 종료. 유틸리티에서 요청된 일부 조치 수행에 실패한 후에는 리스토어 조작을 중단합니다.

LOAD *shared-library*

사용될 벤더 백업 및 리스토어 I/O 기능이 들어 있는 공유 라이브러리 이름(Windows 운영 체제의 DLL). 이름에는 전체 경로가 포함될 수 있습니다. 전체 경로가 제공되지 않을 경우 디폴트 값은 User Exit 프로그램이 있는 경로입니다.

TAKEN AT® *date-time*

데이터베이스 백업 이미지의 시간소인. 시간소인은 백업 작업이 완료된 후에 표시되며, 백업 이미지에 대한 경로 이름의 일부입니다. *yyyymmddhhmmss* 양식으로 지정됩니다. 부분 시간소인을 지정할 수도 있습니다. 예를 들어, 시간소인이 20021001010101 및 200210020101O1인 두 개의 다른 백업 이미지가 존재하는 경우 20021002를 지정하면 시간소인이 200210020101O1인 이미지가 사용됩니다. 이 매개변수의 값을 지정하지 않은 경우 소스 미디어에는 단 하나의 백업 이미지만 있어야 합니다.

TO *target-directory*

이 매개변수는 목표 데이터베이스 디렉토리를 언급합니다. 유틸리티가 기존 데이터베이스로 리스토어 중인 경우에는 이 매개변수는 무시됩니다. 지정하는 드라이브 및 디렉토리는 로컬이어야 합니다. 백업 이미지에 자동 스토리지가 사용 가능한 데이터베이스가 포함되면, 데이터베이스 디렉토리는 변경되고 데이터베이스에 연관된 스토리지 경로는 변경되지 않습니다.

DBPATH ON *target-directory*

이 매개변수는 목표 데이터베이스 디렉토리를 언급합니다. 유틸리티가 기존 데이터베이스로 리스토어 중인 경우에는 이 매개변수는 무시됩니다. 지정하는 드라이브 및 디렉토리는 로컬이어야 합니다. 백업 이미지에 자동 스토리지가 사용 가능하고 ON 매개변수가 지정되지 않은 데이터베이스가 포함되면, 이 매개변수는 TP 매개변수와 동의어로, 데이터베이스 디렉토리만 변경되고 데이터베이스에 연관된 스토리지 경로는 변경되지 않습니다.

ON *path-list*

이 매개변수는 자동 스토리지 데이터베이스와 연관된 스토리지 경로를 재정의합니다. 자동 스토리지에 대해 사용 가능하도록 설정되지 않은 데이터베이스에 이 매개변수를 사용하면 오류가 발생합니다(SQL20321N). 백업 이미지에 정의된 기존 스토리지 경로는 더 이상 사용되지 않으며 자동 스토리지 테이블 스페

이스의 경로는 자동으로 새 경로로 재지정됩니다. 자동 스토리지 데이터베이스에 대해 이 매개변수를 지정하지 않으면, 스토리지 경로는 백업 이미지에 정의된 대로 유지됩니다.

하나 이상의 경로를 지정할 수 있으며 각각은 쉼표로 구분됩니다. 각 경로에는 절대 경로 이름이 있어야 하며 로컬에 있어야 합니다. 데이터베이스가 아직 디스크에 존재하지 않고 DBPATH ON 매개변수를 지정하지 않은 경우, 목표 데이터베이스 디렉토리로 첫 번째 경로가 사용됩니다.

다중 파티션 데이터베이스의 경우 ON *path-list* 옵션은 카탈로그 파티션에서만 지정할 수 있습니다. 카탈로그 파티션은 ON 옵션이 사용될 때 다른 파티션이 리스토어되기 전에 리스토어해야 합니다. 새 스토리지 경로를 사용하는 카탈로그 파티션의 리스토어에서는 카탈로그에 없는 모든 노드가 RESTORE_PENDING 상태가 됩니다. 카탈로그에 없는 노드는 리스토어 명령에 ON절을 지정하지 않고 병렬로 리스토어될 수 있습니다.

일반적으로 멀티파티션 데이터베이스의 각 파티션에 대해 동일한 스토리지 경로를 사용해야 하며 RESTORE DATABASE 명령 실행 이전에 그 경로가 존재해야 합니다. 한 가지 예외는 데이터베이스 파티션 표현식을 스토리지 경로 내에서 사용하는 경우입니다. 이를 수행하면 데이터베이스 파티션 번호를 스토리지 경로에 반영하여 결과로 생성되는 경로 이름이 파티션마다 다릅니다.

인수 " $N" ([blank]$N)을 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 한가지 형식으로만 사용할 수 있습니다.

표 49. . 연산자는 왼쪽에서 오른쪽으로 계산됩니다. %는 모듈러스 연산자를 표시합니다. 예에서 데이터베이스 파티션 번호는 10으로 가정합니다.

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

INTO *target-database-alias*

목표 데이터베이스 별명. 목표 데이터베이스가 존재하지 않으면 작성됩니다.

데이터베이스 백업을 기존 데이터베이스로 리스토어할 때, 리스토어된 데이터베이스는 기존 데이터베이스의 별명 및 데이터베이스 이름을 상속합니다. 데이터베이스 백업을 기존에 없는 데이터베이스로 리스토어하는 경우에는 사용자가 지정하는 별명 및 데이터베이스 이름의 새 데이터베이스가 작성됩니다. 이 새 데이터베이스 이름은 리스토어하는 시스템에서 고유해야 합니다.

LOGTARGET *directory*

비스냅샷 리스토어:

백업 이미지에서 로그 파일을 추출하는 목표 디렉토리로 사용될, 데이터베이스 서버에 있는 기존 디렉토리의 절대 경로 이름. 이 옵션을 지정하는 경우 백업 이미지에 포함된 모든 로그 파일이 목표 디렉토리로 추출됩니다. 이 옵션을 지정하지 않으면 백업 이미지에 포함된 로그 파일은 추출되지 않습니다. 백업 이미지에서 로그 파일만 추출하려면 LOGS 옵션을 지정하십시오.

스냅샷 리스토어:

INCLUDE

스냅샷 이미지에서 로그 디렉토리 볼륨을 리스토어합니다. 이 옵션을 지정하고 백업 이미지에 로그 디렉토리가 포함된 경우 해당 디렉토리가 리스토어됩니다. 디스크의 기존 로그 디렉토리 및 로그 파일은 백업 이미지의 로그 디렉토리와 충돌하지 않으면 그대로 유지됩니다. 디스크의 기존 로그 디렉토리가 백업 이미지의 로그 디렉토리와 충돌하는 경우에는 오류가 리턴됩니다.

EXCLUDE

로그 디렉토리 볼륨을 리스토어하지 않습니다. 이 옵션을 지정하면 로그 디렉토리가 백업 이미지에서 리스토어되지 않습니다. 디스크의 기존 로그 디렉토리 및 로그 파일은 백업 이미지의 로그 디렉토리와 충돌하지 않으면 그대로 유지됩니다. 이로 인해 데이터베이스에 속한 경로가 리스토어되고 로그 디렉토리가 내재적으로 리스토어되어 로그 디렉토리가 겹쳐써지면 오류가 리턴됩니다.

FORCE

스냅샷 이미지를 리스토어할 때 현재 데이터베이스의 기존 로그 디렉토리가 겹쳐쓰여지고 교체되도록 허용합니다. 이 옵션을 사용하지 않으면, 스냅샷 이미지의 로그 디렉토리와 충돌하는 디스크의 기존 로그 디렉토리 및 로그 파일로 인해 리스토어가 실패합니다. 리스토어에서 기존 로그 디렉토리를 겹쳐쓰고 교체할 수 있음을 표시하려면 이 옵션을 사용하십시오.

주: 이 옵션을 사용할 때는 주의해야 하며 항상 복구에 필요할 수 있는 모든 로그를 백업하고 아카이브했는지 확인해야 합니다.

주: LOGTARGET이 지정된 비스냅샷 리스토어가 아닌 경우 디폴트 LOGTARGET 디렉토리는 LOGTARGET EXCLUDE입니다.

NEWLOGPATH *directory*

리스토어 조작 후 활성 로그 파일에 사용될 디렉토리의 절대 경로 이름. 이 매개변수에는 매개변수가 지정된 리스토어 조작에만 적용된다는 점을 제외하고, **newlogpath** 데이터베이스 구성 매개변수와 같은 기능이 있습니다. 이 매개변수는 백업 이미지의 로그 경로가 리스토어 조작 후 사용에 적합하지 않을 때 (예를 들어, 경로가 더 이상 유효하지 않거나 다른 데이터베이스에서 사용 중인 경우) 사용할 수 있습니다.

WITH *num-buffers* **BUFFERS**

사용될 버퍼 수. DB2 데이터베이스 시스템은 사용자가 명시적으로 값을 입력하지 않는 경우를 제외하고는 자동으로 이 매개변수에 대한 최적 값을 선택합니다. 여러 개의 소스를 읽거나 PARALLELISM 값이 증가된 경우 성능을 개선하려면 더 많은 버퍼를 사용할 수 있습니다.

BUFFER *buffer-size*

리스토어 조작에 사용되는 버퍼의 *크기*(페이지 수). DB2 데이터베이스 시스템은 사용자가 명시적으로 값을 입력하지 않는 경우를 제외하고는 자동으로 이 매개변수에 대한 최적 값을 선택합니다. 이 매개변수의 최소값은 8페이지입니다.

리스토어 버퍼 *크기*는 백업 작업 중에 지정된 백업 버퍼 *크기*의 양의 배수입니다. 올바르지 않은 버퍼 *크기*를 지정하면 버퍼는 승인할 수 있는 가장 작은 크기로 할당됩니다.

REPLACE HISTORY FILE

리스토어 조작이 디스크의 실행기록 파일을 백업 이미지의 실행기록 파일로 교체해야 함을 지정합니다.

REPLACE EXISTING

목표 데이터베이스 별명과 같은 별명의 데이터베이스가 이미 있는 경우, 이 매개변수는 리스토어 유틸리티가 기존 데이터베이스를 리스토어된 데이터베이스로 교체하도록 지정합니다. 이는 리스토어 유틸리티를 호출하는 스크립트에 유용합니다. 명령행 처리기는 기존 데이터베이스의 삭제를 검증할 것을 사용자에게 프롬프트하지 않기 때문입니다. WITHOUT PROMPTING 매개변수를 지정하는 경우 REPLACE EXISTING를 지정할 필요가 없지만, 그러면 일반적으로 사용자 개입이 필요한 이벤트가 발생하는 경우에 작업이 실패합니다.

REDIRECT

경로 재지정된 리스토어 조작을 지정합니다. 경로 재지정된 리스토어 조작을 완료하려면 이 명령 다음에 하나 이상의 SET TABLESPACE CONTAINERS 명령과, CONTINUE 옵션이 있는 RESTORE DATABASE 명령이 차례로 있어야 합니다. 경로 재지정된 단일 리스토어 조작과 연관된 모든 명령은 동일한 창 또는 CLP 세션에서 호출해야 합니다.

GENERATE SCRIPT *script*

지정된 파일 이름으로 리스토어 경로 재지정 스크립트를 작성합니다. 스크립트 이름은 상대적 또는 절대적일 수 있으며 스크립트는 클라이언트 측에서 생성됩니다. 클라이언트 측에서 파일을 작성할 수 없으면 오류 메시지(SQL9304N)가 리턴됩니다. 파일이 이미 존재하면 겹쳐씁니다. 추가 사용 정보는 아래의 예를 참조하십시오.

WITHOUT ROLLING FORWARD

데이터베이스가 성공적으로 리스토어된 후에 롤 포워드 보류 상태가 되지 않음을 지정합니다.

리스토어 조작이 성공한 후에 데이터베이스가 롤 포워드 보류 상태가 되면 데이터베이스를 다시 사용하기 전에 ROLLFORWARD 명령을 호출해야 합니다.

온라인 백업 이미지에서 리스토어할 때 이 옵션을 지정한 경우 오류 SQL2537N이 리턴됩니다.

백업 이미지가 복구 가능한 데이터베이스의 이미지인 경우 REBUILD 옵션과 함께 WITHOUT ROLLING FORWARD를 지정할 수 없습니다.

PARALLELISM *n*

리스토어 조작 중에 작성될 버퍼 조작자 수를 지정합니다. DB2 데이터베이스 시스템은 사용자가 명시적으로 값을 입력하지 않는 경우를 제외하고는 자동으로 이 매개변수에 대한 최적 값을 선택합니다.

COMPRLIB *name*

압축 해제를 수행하기 위해 사용할 라이브러리의 이름을 표시합니다(예: Windows의 경우 `db2compr.dll`, Linux/UNIX 시스템의 경우 `libdb2compr.so`). 이 이름은 서버에 있는 파일을 나타내는 완전한 경로이어야 합니다. 이 매개변수를 지정하지 않으면 DB2가 이미지에 저장된 라이브러리 사용을 시도합니다. 백업이 압축되지 않은 경우 이 매개변수 값은 무시됩니다. 지정된 라이브러리를 로드할 수 없으면 리스토어 조작이 실패합니다.

COMPROPTS *string*

압축 해제 라이브러리의 초기화 루틴으로 전달되는 2진 데이터 블록을 설명합니다. DB2 데이터베이스 시스템은 이 문자열을 클라이언트에서 서버로 직접 전달하므로 모든 바이트 리버설 또는 코드 페이지 변환 문제는 압축 해제 라이브러리가 직접 처리합니다. 데이터 블록의 첫 번째 문자가 『@』인 경우 DB2 데이터베이스 시스템은 나머지 데이터를 서버에 있는 파일의 이름으로 해석합니다. 그런 다음 DB2 데이터베이스 시스템은 *string*의 컨텐츠를 이 파일의 컨텐츠로 바꾸고 대신 초기화 루틴으로 새 값을 전달합니다. 문자열의 최대 길이는 1 024바이트입니다.

WITHOUT PROMPTING

리스토어 조작이 의도하지 않게 실행됨을 지정합니다. 보통 사용자가 개입해야

하는 조치는 오류 메시지를 표시합니다. 분리성 매체(예: 테이프 또는 디스켓)를 사용할 경우에는 이 옵션을 지정해도 디바이스 종료 시 사용자에게 프롬프트가 표시됩니다.

예:

1. 다음 예에서 데이터베이스 WSDB는 번호가 0부터 3까지인 모두 네 개의 데이터베이스 파티션에 정의됩니다. 경로 /dev3/backup은 모든 데이터베이스 파티션에서 액세스할 수 있습니다. 다음 오프라인 백업 이미지는 /dev3/backup에서 사용할 수 있습니다.

   ```
   wsdb.0.db2inst1.NODE0000.CATN0000.20020331234149.001
   wsdb.0.db2inst1.NODE0001.CATN0000.20020331234427.001
   wsdb.0.db2inst1.NODE0002.CATN0000.20020331234828.001
   wsdb.0.db2inst1.NODE0003.CATN0000.20020331235235.001
   ```

   먼저 카탈로그 파티션을 리스토어한 후 /dev3/backup 디렉토리에서 WSDB 데이터베이스의 다른 모든 데이터베이스 파티션을 리스토어하려면, 데이터베이스 파티션 중 하나에서 다음 명령을 발행하십시오.

   ```
   db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
   TAKEN AT 20020331234149
     INTO wsdb REPLACE EXISTING'
   db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
   TAKEN AT 20020331234427
     INTO wsdb REPLACE EXISTING'
   db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
   TAKEN AT 20020331234828
     INTO wsdb REPLACE EXISTING'
   db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
   TAKEN AT 20020331235235
     INTO wsdb REPLACE EXISTING'
   ```

   db2_all 유틸리티는 지정된 데이터베이스 파티션마다 리스토어 명령을 발행합니다. db2_all을 사용하여 리스토어를 수행할 때, 항상 REPLACE EXISTING 및/또는 WITHOUT PROMPTING을 지정해야 합니다. 그렇지 않으면 프롬프트가 표시되는 경우에 작업이 정지된 것처럼 보입니다. 이는 db2_all이 사용자 프롬프팅을 지원하지 않기 때문입니다.

2. 다음은 별명이 MYDB인 데이터베이스에 대한 일반적인 경로 재지정된 리스토어 시나리오입니다.

   a. REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

      ```
      restore db mydb replace existing redirect
      ```

      1단계가 성공적으로 완료되면, 3단계 완료 이전에 다음을 발행하여 리스토어 조작을 중단할 수 있습니다.

      ```
      restore db mydb abort
      ```

   b. 컨테이너를 재정의해야 하는 테이블 스페이스마다 SET TABLESPACE CONTAINERS 명령을 발행하십시오. 예를 들면, 다음과 같습니다.

```
set tablespace containers for 5 using
   (file 'f:¥ts3con1' 20000, file 'f:¥ts3con2' 20000)
```

리스토어된 데이터베이스의 컨테이너가 이 단계에서 지정된 컨테이너인지 검증하려면, LIST TABLESPACE CONTAINERS 명령을 발행하십시오.

c. 1 및 2단계를 성공적으로 완료한 후 다음을 발행하십시오.

```
restore db mydb continue
```

이는 경로 재지정된 리스토어 조작의 최종 단계입니다.

d. 3단계가 실패하거나 리스토어 조작이 중단된 경우, 경로 재지정된 리스토어를 1단계부터 재시작할 수 있습니다.

3. 다음은 복구 가능한 데이터베이스의 주별 증분 백업 전략 샘플입니다. 주별 전체 데이터베이스 백업 조작과 일별 비누적(델타) 백업 조작 및 주중 누적(증분) 백업 조작이 포함됩니다.

```
(Sun) backup db mydb use tsm
(Mon) backup db mydb online incremental delta use tsm
(Tue) backup db mydb online incremental delta use tsm
(Wed) backup db mydb online incremental use tsm
(Thu) backup db mydb online incremental delta use tsm
(Fri) backup db mydb online incremental delta use tsm
(Sat) backup db mydb online incremental use tsm
```

금요일 아침에 작성된 이미지의 자동 데이터베이스 리스토어의 경우 다음을 발행하십시오.

```
restore db mydb incremental automatic taken at (Fri)
```

금요일 아침에 작성된 이미지의 수동 데이터베이스 리스토어의 경우 다음을 발행하십시오.

```
restore db mydb incremental taken at (Fri)
restore db mydb incremental taken at (Sun)
restore db mydb incremental taken at (Wed)
restore db mydb incremental taken at (Thu)
restore db mydb incremental taken at (Fri)
```

4. 리모트 사이트로의 전송을 위해 로그를 포함하는 백업 이미지를 생성하려면 다음을 발행하십시오.

```
backup db sample online to /dev3/backup include logs
```

백업 이미지를 리스토어하려면 LOGTARGET 경로를 제공하고 ROLLFORWARD 중에 이 경로를 지정하십시오.

```
restore db sample from /dev3/backup logtarget /dev3/logs
  rollforward db sample to end of logs and stop overflow log path /dev3/logs
```

5. 로그를 포함하는 백업 이미지에서 로그 파일만 검색하려면 다음을 발행하십시오.

```
restore db sample logs from /dev3/backup logtarget /dev3/logs
```

6. USE TSM OPTIONS 키워드를 사용하여 리스토어 조작에 사용할 TSM 정보를 지정할 수 있습니다. Windows 플랫폼에서는 -fromowner 옵션을 생략하십시오.
   - 구분된 문자열 지정:

     ```
     restore db sample use TSM options '"-fromnode=bar -fromowner=dmcinnis"'
     ```

   - 완전한 파일 지정:

     ```
     restore db sample use TSM options @/u/dmcinnis/myoptions.txt
     ```

     myoptions.txt 파일에는 다음 정보가 들어 있습니다. -fromnode=bar -fromowner=dmcinnis
7. 다음은 새 스토리지 경로로 사용하는 다중 파티션 자동 스토리지 사용 가능 데이터베이스의 단순 리스토어입니다. 데이터베이스는 원래 하나의 스토리지 경로 /myPath0으로 작성되었습니다.
   - 카탈로그 파티션에서 restore db mydb on /myPath1,/myPath2를 발행하십시오.
   - 모든 비카탈로그 파티션에서 restore db mydb를 발행하십시오.
8. 자동이 아닌 스토리지 데이터베이스에 대한 다음 명령의 스크립트 출력은

   ```
   restore db sample from /home/jseifert/backups taken at 20050301100417 redirect
   generate script SAMPLE_NODE0000.clp
   ```

   다음과 유사합니다.

   ```
   -- *********************************************************************************
   -- ** 자동으로 작성된 경로 재지정 리스토어 스크립트
   -- *********************************************************************************
   UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
   SET CLIENT ATTACH_DBPARTITIONNUM  0;
   SET CLIENT CONNECT_DBPARTITIONNUM 0;
   -- *********************************************************************************
   -- ** 경로 재지정된 리스토어 초기화
   -- *********************************************************************************
   RESTORE DATABASE SAMPLE
   -- USER  '<username>'
   -- USING '<password>'
   FROM '/home/jseifert/backups'
   TAKEN AT 20050301100417
   -- DBPATH ON '<target-directory>'
   INTO SAMPLE
   -- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/SQLOGDIR/'
   -- WITH <num-buff> BUFFERS
   -- BUFFER <buffer-size>
   -- REPLACE HISTORY FILE
   -- REPLACE EXISTING
   REDIRECT-- PARALLELISM <n>
   -- WITHOUT ROLLING FORWARD
   -- WITHOUT PROMPTING
   ;
   -- *********************************************************************************
   -- ** 테이블 스페이스 정의
   -- *********************************************************************************
   -- *********************************************************************************
   -- ** 테이블 스페이스 이름                    = SYSCATSPACE
   -- **   테이블 스페이스 ID                    = 0
   -- **   테이블 스페이스 유형                  = 시스템 관리 스페이스
   -- **   테이블 스페이스 컨텐츠 유형           = 모든 데이터
   ```

```
-- **   테이블 스페이스 페이지 크기(바이트)      = 4096
-- **   테이블 스페이스 Extent 크기(페이지)      = 32
-- **   자동 스토리지 사용                       = 아니오
-- **   총 페이지 수                             = 5572
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
);
-- ********************************************************************************
-- ** 테이블 스페이스 이름                     = TEMPSPACE1
-- **   테이블 스페이스 ID                       = 1
-- **   테이블 스페이스 유형                     = 시스템 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형              = 시스템 임시 데이터
-- **   테이블 스페이스 페이지 크기(바이트)      = 4096
-- **   테이블 스페이스 Extent 크기(페이지)      = 32
-- **   자동 스토리지 사용                       = 아니오
-- **   총 페이지 수                             = 0
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- ********************************************************************************
-- ** 테이블 스페이스 이름                     = USERSPACE1
-- **   테이블 스페이스 ID                       = 2
-- **   테이블 스페이스 유형                     = 시스템 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형              = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)      = 4096
-- **   테이블 스페이스 Extent 크기(페이지)      = 32
-- **   자동 스토리지 사용                       = 아니오
-- **   총 페이지 수                             = 1
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0002.0'
);
-- ********************************************************************************
-- ** 테이블 스페이스 이름                     = DMS
-- **   테이블 스페이스 ID                       = 3
-- **   테이블 스페이스 유형                     = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형              = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)      = 4096
-- **   테이블 스페이스 Extent 크기(페이지)      = 32
-- **   자동 스토리지 사용                       = 아니오
-- **   자동 크기 조정 사용 가능                 = 아니오
-- **   총 페이지 수                             = 2000
-- **   사용 가능한 페이지 수                    = 1960
-- **   상위 워터 마크(페이지)                   = 96
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   /tmp/dms1                                              1000
, FILE   /tmp/dms2                                              1000
);
-- ********************************************************************************
-- ** 테이블 스페이스 이름                     = RAW
-- **   테이블 스페이스 ID                       = 4
-- **   테이블 스페이스 유형                     = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형              = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)      = 4096
-- **   테이블 스페이스 Extent 크기(페이지)      = 32
-- **   자동 스토리지 사용                       = 아니오
-- **   자동 크기 조정 사용 가능                 = 아니오
-- **   총 페이지 수                             = 2000
-- **   사용 가능한 페이지 수                    = 1960
-- **   상위 워터 마크(페이지)                   = 96
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 4
```

```
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                        1000
, DEVICE '/dev/hdb2'                                        1000
);
-- *********************************************************************************
-- ** 경로 재지정된 리스토어 시작
-- *********************************************************************************
RESTORE DATABASE SAMPLE CONTINUE;
-- *********************************************************************************
-- ** 파일의 끝(EOF)
-- *********************************************************************************
```

9. 자동 스토리지 데이터베이스에 대한 다음 명령의 스크립트 출력은

```
restore db test from /home/jseifert/backups taken at 20050304090733 redirect
generate script TEST_NODE0000.clp
```

다음과 유사합니다.

```
-- *********************************************************************************
-- ** 자동으로 작성된 경로 재지정 리스토어 스크립트
-- *********************************************************************************
UPDATE COMMAND OPTIONS USING S ON Z ON TEST_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM  0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- *********************************************************************************
-- ** 경로 재지정된 리스토어 초기화
-- *********************************************************************************
RESTORE DATABASE TEST
-- USER  '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050304090733
ON '/home/jseifert'
-- DBPATH ON <target-directory>
INTO TEST
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00002/SQLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- *********************************************************************************
-- ** 테이블 스페이스 정의
-- *********************************************************************************
-- *********************************************************************************
-- ** 테이블 스페이스 이름                      = SYSCATSPACE
-- **   테이블 스페이스 ID                      = 0
-- **   테이블 스페이스 유형                    = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형             = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)     = 4096
-- **   테이블 스페이스 Extent 크기(페이지)     = 4
-- **   자동 스토리지 사용                      = 예
-- **   자동 크기 조정 사용 가능                = 예
-- **   총 페이지 수                            = 6144
-- **   사용 가능한 페이지 수                   = 6140
-- **   상위 워터 마크(페이지)                  = 5968
-- *********************************************************************************
-- *********************************************************************************
-- ** 테이블 스페이스 이름                      = TEMPSPACE1
-- **   테이블 스페이스 ID                      = 1
-- **   테이블 스페이스 유형                    = 시스템 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형             = 시스템 임시 데이터
-- **   테이블 스페이스 페이지 크기(바이트)     = 4096
-- **   테이블 스페이스 Extent 크기(페이지)     = 32
-- **   자동 스토리지 사용                      = 예
-- **   총 페이지 수                            = 0
```

```
-- *********************************************************************************
-- *********************************************************************************
-- ** 테이블 스페이스 이름                          = USERSPACE1
-- **   테이블 스페이스 ID                          = 2
-- **   테이블 스페이스 유형                        = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형                 = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)         = 4096
-- **   테이블 스페이스 Extent 크기(페이지)         = 32
-- **   자동 스토리지 사용                          = 예
-- **   자동 크기 조정 사용 가능                    = 예
-- **   총 페이지 수                                = 256
-- **   사용 가능한 페이지 수                       = 224
-- **   상위 워터 마크(페이지)                      = 96
-- *********************************************************************************
-- *********************************************************************************
-- ** 테이블 스페이스 이름                          = DMS
-- **   테이블 스페이스 ID                          = 3
-- **   테이블 스페이스 유형                        = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형                 = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)         = 4096
-- **   테이블 스페이스 Extent 크기(페이지)         = 32
-- **   자동 스토리지 사용                          = 아니오
-- **   자동 크기 조정 사용 가능                    = 아니오
-- **   총 페이지 수                                = 2000
-- **   사용 가능한 페이지 수                       = 1960
-- **   상위 워터 마크(페이지)                      = 96
-- *********************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   '/tmp/dms1'                                        1000
, FILE   '/tmp/dms2'                                        1000
);
-- *********************************************************************************
-- ** 테이블 스페이스 이름                          = RAW
-- **   테이블 스페이스 ID                          = 4
-- **   테이블 스페이스 유형                        = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형                 = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)         = 4096
-- **   테이블 스페이스 Extent 크기(페이지)         = 32
-- **   자동 스토리지 사용                          = 아니오
-- **   자동 크기 조정 사용 가능                    = 아니오
-- **   총 페이지 수                                = 2000
-- **   사용 가능한 페이지 수                       = 1960
-- **   상위 워터 마크(페이지)                      = 96
-- *********************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                        1000
, DEVICE '/dev/hdb2'                                        1000
);
-- *********************************************************************************
-- ** 경로 재지정된 리스토어 시작
-- *********************************************************************************
RESTORE DATABASE TEST CONTINUE;
-- *********************************************************************************
-- ** 파일의 끝(EOF)
-- *********************************************************************************
```

10. 다음은 SNAPSHOT 옵션을 사용하는 RESTORE DB 명령의 예입니다.

    스냅샷 이미지에서 로그 디렉토리 볼륨을 리스토어하고 프롬프트하지 않습니다.

    ```
    db2 restore db sample use snapshot LOGTARGET INCLUDE without prompting
    ```

    로그 디렉토리 볼륨을 리스토어하지 않고 프롬프트도 하지 않습니다.

    ```
    db2 restore db sample use snapshot LOGTARGET EXCLUDE without prompting
    ```

로그 디렉토리 볼륨을 리스토어하지 않고 프롬프트도 하지 않습니다. LOGTARGET을 지정하지 않은 경우, 디폴트는 LOGTARGET EXCLUDE입니다.

```
db2 restore db sample use snapshot without prompting
```

충돌하는 로그 디렉토리를 포함하는 스냅샷 이미지를 리스토어할 때 프롬프트 없이 현재 데이터베이스의 기존 로그 디렉토리 위에 겹쳐쓰고 이 디렉토리가 교체되도록 허용합니다.

```
db2 restore db sample use snapshot LOGTARGET EXCLUDE FORCE without prompting
```

충돌하는 로그 디렉토리를 포함하는 스냅샷 이미지를 리스토어할 때 프롬프트 없이 현재 데이터베이스의 기존 로그 디렉토리 위에 겹쳐쓰고 이 디렉토리가 교체되도록 허용합니다.

```
db2 restore db sample use snapshot LOGTARGET INCLUDE FORCE without prompting
```

사용 시 참고사항

- `db2 restore db <name>` 양식의 RESTORE DATABASE 명령은 데이터베이스 이미지를 사용하여 전체 데이터베이스 리스토어를 수행하고 테이블 스페이스 이미지에서 발견된 테이블 스페이스의 테이블 스페이스 리스토어 조작을 수행합니다. `db2 restore db <name> tablespace` 양식의 RESTORE DATABASE 명령은 이미지에서 발견된 테이블 스페이스의 테이블 스페이스 리스토어를 수행합니다. 또한 이와 같은 명령과 함께 테이블 스페이스 목록이 제공되면, 명시적으로 나열된 테이블 스페이스가 리스토어됩니다.
- 온라인 백업의 리스토어 조작 다음에 롤 포워드 복구를 수행해야 합니다.
- 백업 이미지가 압축된 경우 DB2 데이터베이스 시스템은 이 상태를 발견하고 리스토어하기 전에 데이터를 자동으로 압축 해제합니다. 라이브러리가 db2Restore API에 지정된 경우 데이터 압축 해제에 사용됩니다. 그렇지 않으면, 백업 이미지에 라이브러리가 저장되어 있는지, 그리고 이미지가 사용되는 라이브러리가 존재하는지 확인합니다. 마지막으로, 백업 이미지에 저장된 라이브러리가 없는 경우에는 데이터를 압축 해제할 수 없으며 리스토어 조작이 실패합니다.
- 백업 이미지에서 압축 라이브러리를 리스토어 중인 경우(COMPRESSION LIBRARY 옵션을 지정하여 명시적으로, 또는 압축된 백업의 일반 리스토어를 수행하여 내재적으로) 리스토어 조작은 백업이 수행된 동일한 플랫폼 및 운영 체제에서 수행해야 합니다. DB2에서 일반적으로 두 시스템을 수반하는 플랫폼 간 리스토어 조작을 지원하는 경우에도, 백업이 수행된 플랫폼이 리스토어가 수행되는 플랫폼과 다른 경우 리스토어 조작이 실패합니다.
- 백업된 SMS 테이블 스페이스는 SMS 테이블 스페이스로만 리스토어할 수 있습니다. DMS 테이블 스페이스로, 또는 그 반대로 리스토어할 수 없습니다.
- 로그 파일을 포함하는 백업 이미지에서 로그 파일을 리스토어하려면 DB2 서버에 있는 완전하고 유효한 경로를 제공하여 LOGTARGET 옵션을 지정해야 합니다. 해당

조건이 충족되면 리스토어 유틸리티는 이미지의 로그 파일을 목표 경로에 기록합니다. 로그를 포함하지 않는 백업 이미지 리스토어 중에 LOGTARGET을 지정한 경우, 리스토어 조작은 테이블 스페이스 데이터 리스토어 시도 전에 오류를 리턴합니다. 유효하지 않거나 읽기 전용의 LOGTARGET 경로를 지정한 경우에도 리스토어 조작이 오류와 함께 실패합니다.

- RESTORE DATABASE 명령이 발행될 때 LOGTARGET 경로에 로그 파일이 있으면 경고 프롬프트가 사용자에게 리턴됩니다. WITHOUT PROMPTING을 지정한 경우에는 이 경고가 리턴되지 않습니다.
- LOGTARGET이 지정된 리스토어 조작 중에 로그 파일을 추출할 수 없으면 리스토어 조작이 실패하고 오류가 리턴됩니다. 백업 이미지에서 추출 중인 로그 파일이 LOGTARGET 경로에 있는 기존 파일과 이름이 같은 경우 리스토어 조작이 실패하고 오류가 리턴됩니다. 리스토어 데이터베이스 유틸리티는 LOGTARGET 디렉토리의 기존 로그 파일을 겹쳐쓰지 않습니다.
- 백업 이미지에서는 저장된 로그 세트만 리스토어할 수 있습니다. 로그 파일만 리스토어할 것을 표시하려면 LOGTARGET 경로와 함께 LOGS 옵션을 지정하십시오. LOGTARGET 경로 없이 LOGS 옵션만 지정하면 오류가 발생합니다. 이 작업 모드에서 로그 파일을 리스토어하는 중에 문제점이 발생하면 리스토어 조작이 즉시 종료되고 오류가 리턴됩니다.
- 자동 증분 리스토어 조작 중에는 리스토어 조작의 목표 이미지에 포함된 로그 파일만 백업 이미지에서 검색됩니다. 증분 리스토어 프로세스 중에 참조되는 중간 이미지에 포함된 로그 파일은 중간 백업 이미지에서 추출되지 않습니다. 수동 증분 리스토어 조작 중에 LOGTARGET 경로는 발행할 최종 리스토어 명령에만 지정해야 합니다.
- 오프라인 전체 데이터베이스 백업과 오프라인 증분 데이터베이스 백업은 나중 데이터베이스 버전으로 리스토어할 수 있지만, 온라인 백업은 리스토어할 수 없습니다. 다중 파티션 데이터베이스의 경우, 카탈로그 파티션을 개별적으로 먼저 리스토어하고, 그 다음으로 나머지 데이터베이스 파티션을 리스토어합니다(병렬로 또는 순차적으로). 그러나 리스토어 조작에서 수행된 내재된 데이터베이스 업그레이드는 실패할 수 있습니다. 다중 파티션 데이터베이스에서는 하나 이상의 데이터베이스 파티션에서 실패할 수 있습니다. 이와 같은 경우, 데이터베이스를 성공적으로 업그레이드하기 위해 카탈로그 파티션에서 발행된 단일 UPGRADE DATABASE 명령을 RESTORE DATABASE 명령 다음에 발행할 수 있습니다.

스냅샷 리스토어

일반(비스냅샷) 리스토어처럼, 스냅샷 백업 이미지를 리스토어할 때 디폴트 동작은 로그 디렉토리를 리스토어하지 않는 것입니다(LOGTARGET EXCLUDE).

DB2 관리 프로그램이 리스토어하려는 기타 경로에서 로그 디렉토리의 그룹 ID가 공유되는 것을 발견하면 오류가 리턴됩니다. 이 경우 LOGTARGET INCLUDE 또는 LOGTARGET INCLUDE FORCE를 지정해야 합니다. 로그 디렉토리가 리스토어의 일부여야 하기 때문입니다.

DB2 관리 프로그램은 백업 이미지의 경로 리스토어가 수행되기 전에 기존 로그 디렉토리를 모두 저장하려고 합니다(기본, 미러 및 오버플로우).

로그 디렉토리를 리스토어하려고 하는데 DB2 관리 프로그램이 디스크의 기존 로그 디렉토리가 백업 이미지의 로그 디렉토리와 충돌하는 것을 발견하면 DB2 관리 프로그램이 오류를 보고합니다. 이와 같은 경우 LOGTARGET INCLUDE FORCE를 지정하면 이 오류가 표시되지 않고 이미지의 로그 디렉토리가 리스토어되어 기존의 모든 사항은 삭제됩니다.

LOGTARGET EXCLUDE 옵션이 지정되고 로그 디렉토리 경로가 데이터베이스 디렉토리(즉, /NODExxxx/SQLxxxxx/SQLOGDIR/)에 있는 특수한 상황도 있습니다. 이 경우 리스토어는 로그 디렉토리를 데이터베이스 경로로 겹쳐쓰고 포함된 모든 컨텐츠가 리스토어됩니다. DB2 관리 프로그램이 이 시나리오를 발견하고 로그 파일이 이 로그 디렉토리에 있는 경우 오류가 보고됩니다. LOGTARGET EXCLUDE FORCE를 지정하면, 이 오류가 표시되지 않고 백업 이미지의 해당 로그 디렉토리는 디스크의 충돌하는 로그 디렉토리를 겹쳐씁니다.

일시중단된 입출력 및 온라인 분할 미러 지원을 통한 고가용성

IBM Data Server 일시중단 입출력 지원을 사용하면 데이터베이스가 오프라인이 아니어도 기본 데이터베이스의 미러링된 사본을 분할할 수 있습니다. 이를 사용하여 기본 데이터베이스가 실패하는 경우 인계받을 대기 데이터베이스를 매우 신속하게 작성하여 수 있습니다.

디스크 미러링은 데이터를 두 개의 분리된 하드 디스크에 동시에 쓰는 프로세스입니다. 하나의 데이터 사본을 다른 데이터 사본의 미러라고 합니다. 미러를 분할하는 것은 두 사본을 분리하는 프로세스입니다.

디스크 미러링을 사용하여 기본 데이터베이스의 보조 사본을 유지보수할 수 있습니다. IBM Data Server 일시중단 입출력 기능을 사용하면 데이터베이스가 오프라인이 아니어도 데이터베이스의 기본 및 보조 미러된 사본을 분할할 수 있습니다. 기본 및 보조 데이터베이스 사본이 분할되면, 기본 데이터베이스가 실패하는 경우 보조 데이터베이스가 작업을 인계받을 수 있습니다.

IBM Data Server 백업 유틸리티를 사용하여 대형 데이터베이스를 백업하지 않으려는 경우, 일시중단된 입출력 및 분할 미러 기능을 사용하여 미러된 이미지에서 사본을 작성할 수 있습니다. 이 접근 방식은 또한 다음을 수행합니다.

- 프로덕션 머신에서 백업 작업 오버헤드를 제거합니다.
- 시스템을 클론할 빠른 방법을 나타냅니다.
- 유휴 대기 장애 복구의 급속 구현을 나타냅니다. 초기 리스토어 작업이 없는데 롤 포워드 작업이 너무 느린 것으로 증명되거나 오류가 발생하면 다시 초기화가 아주 빠릅니다.

db2inidb 명령은 분할 미러를 초기화하여 다음과 같이 사용 가능하도록 합니다.

- 클론 데이터베이스로서
- 대기 데이터베이스로서
- 백업 이미지로서

이 명령은 분할 미러에 대해서만 발행할 수 있으며 분할 미러를 사용하기 전에 실행해야 합니다.

파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션에 대해 동시에 입출력 쓰기를 일시중단하지 않아도 됩니다. 하나 이상의 데이터베이스 파티션으로 구성된 서브세트를 일시중단하여 오프라인 백업을 수행하기 위한 분할 미러를 작성할 수 있습니다. 카탈로그 파티션이 서브세트에 포함되는 경우, 일시중단할 마지막 데이터베이스 파티션이어야 합니다.

파티션된 데이터베이스 환경의 경우 데이터베이스 파티션의 분할 이미지를 사용하려면 모든 데이터베이스 파티션에서 db2inidb 명령을 실행해야 합니다. 도구는 db2_all 명령을 사용하여 모든 데이터베이스 파티션에서 동시에 실행할 수 있습니다. 그러나 RELOCATE USING 옵션을 사용하는 경우에는 db2_all 명령을 사용하여 모든 파티션에서 동시에 db2inidb를 실행할 수 없습니다. 데이터베이스 파티션마다 변경될 데이터베이스 파티션의 NODENUM 값을 포함하는 개별 구성 파일을 제공해야 합니다. 예를 들어, 데이터베이스 이름이 변경되는 경우 모든 데이터베이스 파티션이 영향을 받으며 각 데이터베이스 파티션의 개별 구성 파일과 함께 db2relocatedb 명령을 실행해야 합니다. 단일 데이터베이스 파티션에 속하는 컨테이너를 이동하는 경우 db2relocatedb 명령은 해당 데이터베이스 파티션에서 한 번만 실행해야 합니다.

주: 분할 미러에 볼륨 디렉토리를 포함하여 데이터베이스를 구성하는 모든 컨테이너 및 디렉토리가 포함되어 있는지 확인하십시오. 이 정보를 수집하려면 분할해야 하는 데이터베이스의 모든 파일 및 디렉토리를 표시하는 DBPATHS 관리 뷰를 참조하십시오.

db2inidb - 미러된 데이터베이스 초기화

분할 미러 환경에서 미러된 데이터베이스를 초기화합니다. 미러된 데이터베이스는 기본 데이터베이스의 클론으로서 초기화하거나 롤 포워드 보류 상태로 두거나 기본 데이터베이스를 리스토어하기 위한 백업 이미지로서 사용할 수 있습니다. 이 명령은 분할 미러 데이터베이스에 대해서만 실행할 수 있으며 분할 미러를 사용하려면 이 명령을 실행해야 합니다.

권한 부여

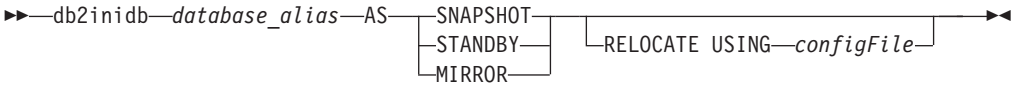
다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*

필수 연결

없음

명령 구문



명령 매개변수

database_alias
초기화될 데이터베이스의 별명을 지정합니다.

SNAPSHOT
미러된 데이터베이스가 기본 데이터베이스의 클론으로 초기화됨을 지정합니다.

STANDBY
데이터베이스가 롤 포워드 보류 상태가 됨을 지정합니다. 기본 데이터베이스의 새 로그는 페치되며 대기 데이터베이스에 적용될 수 있습니다. 기본 데이터베이스 작동이 중단되면 대신 대기 데이터베이스를 사용할 수 있습니다.

MIRROR
미러된 데이터베이스가 기본 데이터베이스를 리스토어하는 데 사용할 수 있는 백업 이미지로서 사용되는 것으로 지정합니다.

RELOCATE USING *configFile*
데이터베이스 파일 위치가 데이터베이스를 스냅샷, 대기 또는 미러로서 초기화하기 전에 지정된 *configFile*에 나열된 정보를 기반으로 다시 지정되는 것으로 지정합니다. *configFile* 형식은 432 페이지의 『db2relocatedb - 데이터베이스 재배치』에서 설명합니다.

사용 시 참고사항

db2inidb *database_alias* as mirror 명령을 실행하기 전에 `db2 connect to` *database-alias* 조작을 실행하지 마십시오. 초기화하기 전에 분할된 미러 데이터베이스에 대한 연결을 시도하면 롤 포워드 복구 중에 필요한 로그 파일이 지워집니다. 연결 시 데이터베이스를 일시중단할 때의 상태로 데이터를 다시 설정합니다. 데이터베이스가

일시중단된 시점과 일치하는 것으로 표시되면 DB2 데이터베이스는 응급 복구가 필요하지 않은 것으로 판단하여 나중에 사용할 수 있도록 로그를 비웁니다. 로그를 비운 경우 롤 포워드를 시도하면 SQL4970N 오류 메시지가 리턴됩니다.

파티션된 데이터베이스 환경의 경우 데이터베이스 파티션의 분할 미러를 사용하려면 모든 데이터베이스 파티션에서 db2inidb를 실행해야 합니다. db2inidb는 db2_all 명령을 사용하여 모든 데이터베이스 파티션에서 동시에 실행할 수 있습니다.

그러나 RELOCATE USING 옵션을 사용하는 경우에는 db2_all 명령을 사용하여 모든 파티션에서 동시에 db2inidb를 실행할 수 있습니다. 각 파티션마다 변경될 데이터베이스 파티션의 NODENUM 값을 포함하는 개별 구성 파일을 제공해야 합니다. 예를 들어, 데이터베이스 이름이 변경되는 경우 모든 데이터베이스 파티션이 영향을 받으며 각 데이터베이스 파티션의 개별 구성 파일과 함께 db2relocatedb 명령을 실행해야 합니다. 단일 데이터베이스 파티션에 속하는 컨테이너를 이동하는 경우 db2relocatedb 명령은 해당 데이터베이스 파티션에서 한 번만 실행해야 합니다.

RELOCATE USING *configFile* 매개변수가 지정되고 데이터베이스 위치가 변경되면 지정된 *configFile*이 데이터베이스 디렉토리에 복사되며 이름이 `db2path.cfg`로 바뀝니다. 이 파일은 후속 응급 복구 또는 롤 포워드 복구 중에 로그 파일이 처리됨에 따라 컨테이너 경로의 이름을 바꾸는 데 사용됩니다.

클론 데이터베이스를 초기화하는 경우, 지정된 *configFile*은 응급 복구가 완료된 후 데이터베이스 디렉토리에서 자동으로 제거됩니다.

대기 데이터베이스 또는 미러된 데이터베이스를 초기화하는 경우, 지정된 *configFile*은 롤 포워드 복구가 완료 또는 취소된 후 데이터베이스 디렉토리에서 자동으로 제거됩니다. db2inidb가 실행된 후 `db2path.cfg` 파일에 새 컨테이너 경로를 추가할 수 있습니다. 이는 원래 데이터베이스에서 CREATE 또는 ALTER TABLESPACE 조작이 완료되고 대기 데이터베이스에 다른 경로를 사용해야 하는 경우 필요합니다.

db2relocatedb - 데이터베이스 재배치

이 명령은 사용자가 제공하는 구성 파일에 지정된 대로 데이터베이스의 이름을 바꾸거나 데이터베이스 또는 데이터베이스의 일부(예: 컨테이너 및 로그 디렉토리)를 재배치합니다. 이 도구는 DB2 인스턴스와 데이터베이스 자원 파일에 대해 필요한 변경사항을 작성합니다.

권한 부여

없음

명령 구문

명령 매개변수

-f *configFilename*

데이터베이스 재배치에 필요한 구성 정보를 포함하는 파일의 이름을 지정합니다. 상대적 또는 절대적 파일 이름이 될 수 있습니다. 구성 파일의 형식은 다음과 같습니다.

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
...
STORAGE_PATH=oldStoragePath1,newStoragePath1
STORAGE_PATH=oldStoragePath2,newStoragePath2
...
```

여기서,

DB_NAME

재배치할 데이터베이스의 이름을 지정합니다. 데이터베이스 이름이 변경되는 경우 이전 이름과 새 이름 모두 지정해야 합니다. 필수 필드입니다.

DB_PATH

재배치할 데이터베이스의 원래 경로를 지정합니다. 데이터베이스 경로가 변경되는 경우, 이전 경로와 새 경로 모두 지정해야 합니다. 필수 필드입니다.

INSTANCE

데이터베이스가 있는 인스턴스를 지정합니다. 데이터베이스가 새 인스턴스로 이동하는 경우, 이전 인스턴스와 새 인스턴스 모두 지정해야 합니다. 필수 필드입니다.

NODENUM

데이터베이스 노드의 노드 번호가 변경됨을 지정합니다. 디폴트값은 0입니다.

LOG_DIR

로그 경로의 해당 위치에서 변경을 지정합니다. 로그 경로가 변경되는 경우, 이전 경로와 새 경로 모두 지정해야 합니다. 로그 경로가 데이터베이스 경로 아래에 있는 경우(경로가 자동으로 갱신되는 경우) 이 스펙은 선택적입니다.

CONT_PATH

테이블 스페이스 컨테이너의 해당 위치에서 변경을 지정합니다. 이전 및 새 컨테이너 경로 모두 지정해야 합니다. 여러 개의 컨테이너 경로 변경사항이 작성되는 경우 여러 개의 CONT_PATH 라인을 제공할 수 있습니다. 컨테이너 경로가 데이터베이스 경로 아래에 있는 경우(경로가 자동으로 갱신되는 경우) 이 스펙은 선택적입니다. 동일한 이전 경로가 공통적인 새 경로로 교체되는 두 개 이상의 컨테이너를 변경하는 경우, 단일 CONT_PATH 항목을 사용할 수 있습니다. 이와 같은 경우, 별표(*)는 이전 경로와 새 경로 모두에서 와일드 카드로 사용할 수 있습니다.

STORAGE_PATH

자동 스토리지가 사용 가능한 데이터베이스에만 적용 가능합니다. 데이터베이스에 대한 스토리지 경로 중 하나의 위치에서 변경을 지정합니다. 이전 스토리지 경로와 새 스토리지 경로 모두 지정해야 합니다. 몇 개의 스토리지 경로 변경사항이 작성되는 경우 여러 개의 STORAGE_PATH 라인을 제공할 수 있습니다.

공백 라인이나 주석 문자(#)로 시작하는 라인은 무시됩니다.

예:

예 1

/home/db2inst1 경로에 있는 db2inst1 인스턴스에서 데이터베이스 TESTDB의 이름을 PRODDB로 변경하려면 다음 구성 파일을 작성하십시오.

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

구성 파일을 relocate.cfg로 저장하고 다음 명령을 사용하여 데이터베이스 파일을 변경하십시오.

```
db2relocatedb -f relocate.cfg
```

예 2:

데이터베이스 DATAB1을 경로 /dbpath의 인스턴스 jsmith에서 인스턴스 prodinst로 이동하려면 다음을 수행하십시오.

1. 디렉토리 /dbpath/jsmith의 파일을 /dbpath/prodinst로 이동하십시오.
2. db2relocatedb 명령과 함께 다음 구성 파일을 사용하여 데이터베이스 파일 변경사항을 작성하십시오.

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

예 3

데이터베이스 PRODDB는 경로 /databases/PRODDB의 inst1 인스턴스에 있습니다. 두 테이블 스페이스 컨테이너의 위치는 다음과 같이 변경해야 합니다.

- SMS 컨테이너 /data/SMS1은 /DATA/NewSMS1로 이동해야 합니다.
- DMS 컨테이너 /data/DMS1은 /DATA/DMS1로 이동해야 합니다.

실제 디렉토리 및 파일이 새 위치로 이동된 후, db2relocatedb 명령과 함께 다음 구성 파일을 사용하여 새 위치를 인식하도록 데이터베이스 파일 변경사항을 작성할 수 있습니다.

```
DB_NAME=PRODDB
DB_PATH=/databases/PRODDB
INSTANCE=inst1
NODENUM=0
CONT_PATH=/data/SMS1,/DATA/NewSMS1
CONT_PATH=/data/DMS1,/DATA/DMS1
```

예 4

데이터베이스 TESTDB는 인스턴스 db2inst1에 있으며 경로 /databases/TESTDB에서 작성되었습니다. 테이블 스페이스는 다음 컨테이너를 사용하여 작성되었습니다.

```
TS1
TS2_Cont0
TS2_Cont1
/databases/TESTDB/TS3_Cont0
/databases/TESTDB/TS4/Cont0
/Data/TS5_Cont0
/dev/rTS5_Cont1
```

TESTDB는 새 시스템으로 이동됩니다. 새 시스템의 인스턴스는 newinst가 되고 데이터베이스의 위치는 /DB2가 됩니다.

데이터베이스를 이동할 때 /databases/TESTDB/db2inst1 디렉토리에 존재하는 모든 파일은 /DB2/newinst 디렉토리로 이동해야 합니다. 이는 처음 5개의 컨테이너가 이 이동의 일부로 재배치됨을 의미합니다. (처음 세 개는 데이터베이스 디렉토리에 상대적이고 다음 두 개는 데이터베이스 경로에 상대적입니다.) 이 컨테이너는 데이터베이스 디렉토리나 데이터베이스 경로 내에 위치하므로, 구성 파일에 나열하지 않아도 됩니다. 나머지 두 개의 컨테이너가 새 시스템의 다른 위치로 이동되는 경우, 그 컨테이너는 구성 파일에 나열해야 합니다.

실제 디렉토리 및 파일이 새 위치로 이동된 후, db2relocatedb와 함께 다음 구성 파일을 사용하여 새 위치를 인식하도록 데이터베이스 파일 변경사항을 작성할 수 있습니다.

```
DB_NAME=TESTDB
DB_PATH=/databases/TESTDB,/DB2
INSTANCE=db2inst1,newinst
NODENUM=0
CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

예 5

데이터베이스 TESTDB는 데이터베이스 파티션 서버 10 및 20에 두 개의 데이터베이스 파티션을 가지고 있습니다. 인스턴스는 servinst이고 데이터베이스 경로는 두 데이터베이스 파티션 서버 모두 /home/servinst입니다. 데이터베이스의 이름은 SERVDB로 변경되고 데이터베이스 경로는 두 데이터베이스 파티션 서버 모두에서 /databases로 변경됩니다. 또한 로그 디렉토리가 데이터베이스 파티션 서버 20에서, /testdb_logdir로부터 /servdb_logdir로 변경됩니다.

변경사항은 두 데이터베이스 파티션 모두에 대해 작성되므로, 각 데이터베이스 파티션에 대해 구성 파일을 작성하고 해당되는 구성 파일을 사용하여 각 데이터베이스 파티션 서버에서 db2relocatedb를 실행해야 합니다.

데이터베이스 파티션 서버 10에서, 다음 구성 파일이 사용됩니다.

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=10
```

데이터베이스 파티션 서버 20에서, 다음 구성 파일이 사용됩니다.

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=20
LOG_DIR=/testdb_logdir,/servdb_logdir
```

예 6

데이터베이스 MAINDB는 경로 /home/maininst의 maininst 인스턴스에 있습니다. 네 개의 테이블 스페이스 컨테이너의 위치는 다음과 같이 변경해야 합니다.

```
/maininst_files/allconts/C0 needs to be moved to /MAINDB/C0
/maininst_files/allconts/C1 needs to be moved to /MAINDB/C1
/maininst_files/allconts/C2 needs to be moved to /MAINDB/C2
/maininst_files/allconts/C3 needs to be moved to /MAINDB/C3
```

실제 디렉토리 및 파일이 새 위치로 이동된 후, db2relocatedb 명령과 함께 다음 구성 파일을 사용하여 새 위치를 인식하도록 데이터베이스 파일 변경사항을 작성할 수 있습니다.

모든 컨테이너에 유사한 변경이 작성됩니다. 즉, /maininst_files/allconts/는 와일드 카드 문자가 있는 단일 항목을 사용할 수 있도록 /MAINDB/로 교체됩니다.

```
DB_NAME=MAINDB
DB_PATH=/home/maininst
INSTANCE=maininst
NODE_NUM=0
CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

사용 시 참고사항

데이터베이스가 속하는 인스턴스가 변경되는 경우, 이 명령을 실행하기 전에 다음을 수행하여 인스턴스 및 데이터베이스 지원 파일의 변경사항이 작성되는지 확인해야 합니다.

- 데이터베이스가 다른 인스턴스로 이동되는 경우 새 인스턴스를 작성하십시오. 새 인스턴스는 데이터베이스가 현재 상주하는 인스턴스와 동일한 릴리스 레벨에 있어야 합니다.
- 복사하는 데이터베이스에 속하는 파일 및 디바이스를 새 인스턴스가 있는 시스템으로 복사하십시오. 필요에 따라 경로 이름을 변경해야 합니다. 그러나 데이터베이스 파일이 이동되는 디렉토리에 이미 데이터베이스가 있는 경우, 실수로 기존 sqldbdir 파일 위에 겹쳐써서 기존 데이터베이스에 대한 참조를 제거할 수 있습니다. 이 시나리오에서, db2relocatedb 유틸리티는 사용할 수 없습니다. db2relocatedb 대신, 경로 재지정된 리스토어 조작을 사용할 수 있습니다.
- 인스턴스 소유자가 소유하도록 복사된 파일/디바이스의 사용 권한을 변경하십시오.

ALTER TABLESPACE MANAGED BY AUTOMATIC STORAGE 문을 사용하여 자동 스토리지를 사용하도록 변환된 테이블 스페이스에 대한 기존의 사용자 작성 컨테이너를 이동하는 경우에는 db2relocatedb 명령을 사용할 수 없습니다.

인스턴스가 변경되는 경우 도구는 새 인스턴스 소유자가 실행해야 합니다.

파티션된 데이터베이스 환경에서 이 도구는 변경해야 하는 모든 데이터베이스 파티션에 대해 실행해야 합니다. 데이터베이스 파티션마다 변경될 데이터베이스 파티션의 NODENUM 값을 포함하는 개별 구성 파일을 제공해야 합니다. 예를 들어, 데이터베이스 이름이 변경되는 경우 모든 데이터베이스 파티션이 영향을 받으며 각 데이터베이스 파티션의 개별 구성 파일과 함께 db2relocatedb 명령을 실행해야 합니다. 단일 데이터베이스 파티션에 속하는 컨테이너를 이동하는 경우 db2relocatedb 명령은 해당 데이터베이스 파티션에서 한 번만 실행해야 합니다.

로드 진행 중이거나 LOAD RESTART 또는 LOAD TERMINATE 명령이 완료되기를 기다리고 있는 데이터베이스를 재배치하는 경우에는 db2relocatedb 명령을 사용할 수 없습니다.

제한사항: 파티션된 데이터베이스 환경에서, 노드가 동일한 디바이스에 있는 두 개 이상의 논리적 파티션 중 하나인 경우 전체 노드를 재배치할 수 없습니다.

db2look - DB2 통계 및 DDL 추출 도구

필요한 데이터 정의 언어(DDL)문을 추출하여 테스트 데이터베이스에서 프로덕션 데이터베이스의 데이터베이스 오브젝트를 다시 생성합니다. db2look 명령은 오브젝트 유형별로 DDL문을 생성합니다.

이 도구는 테스트 데이터베이스의 오브젝트에 대해 통계를 복제하는 데 사용되는 필수 UPDATE문을 생성할 수 있습니다. 테스트 데이터베이스의 레지스트리 변수 및 쿼리 옵티마이저 관련 구성 매개변수가 프로덕션 데이터베이스의 변수 및 매개변수와 일치하도록 UPDATE DATABASE CONFIGURATION 및 UPDATE DATABASE MANAGER CONFIGURATION 명령과 db2set 명령을 생성하기 위해 사용할 수도 있습니다.

종종 테스트 시스템에 프로덕션 시스템의 데이터 서브세트가 포함되도록 하는 것이 편리합니다. 그러나 이와 같은 테스트 시스템에 대해 선택된 액세스 플랜은 프로덕션 시스템에 대해 선택되는 액세스 플랜과 반드시 같을 필요는 없습니다. 테스트 시스템의 구성 매개변수 및 카탈로그 통계를 갱신하여 프로덕션 시스템의 것과 일치하도록 해야 합니다. 이 도구를 사용하면 액세스 플랜이 프로덕션 시스템에서 사용되는 것과 유사한 테스트 데이터베이스를 작성할 수 있습니다.

db2look 명령이 생성하는 DDL문을 점검해야 합니다. 원래 SQL 오브젝트의 모든 특성을 정확히 재생성하지 못할 수도 있기 때문입니다. 파티션된 데이터베이스 환경의 테이블 스페이스에 대해서는 일부 데이터베이스 파티션이 활성 상태가 아닌 경우 DDL이 완전하지 않을 수 있습니다. ACTIVATE 명령을 사용하여 모든 데이터베이스 파티션이 활성 상태인지 확인하십시오.

권한 부여

시스템 카탈로그 테이블에 대한 SELECT 특권

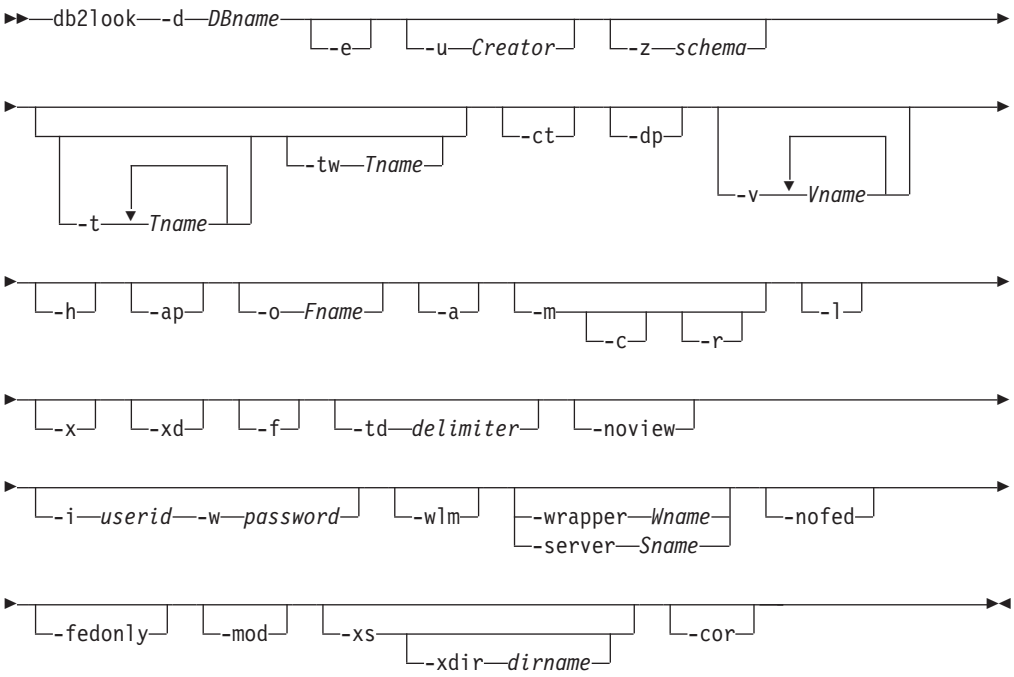
일부 경우(예: 테이블 스페이스 컨테이너 DDL을 생성하는 경우)에는 다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *sysmon*
- *dbadm*

필수 연결

없음

명령 구문



명령 매개변수

-d *DBname*

쿼리할 프로덕션 데이터베이스의 별명 이름. *DBname*은 Linux, UNIX 및 Windows용 DB2 데이터베이스 또는 z/OS용 DB2 버전 9.1(z/OS용 DB2) 데이터베이스의 이름이 될 수 있습니다. *DBname*이 z/OS용 DB2 데이터베이스인 경우, db2look 유틸리티는 OS/390 및 z/OS 오브젝트에 대한 DDL 및 UPDATE 통계 명령문을 추출합니다. 이 DDL 및 UPDATE 통계 명령문은 Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터베이스에 적용 가능한 명령문으로 z/OS용 DB2 데이터베이스에는 적용되지 않습니다. 이는 OS/390 및 z/OS 오브젝트를 추출하고 Linux, UNIX 및 Windows용 DB2 데이터베이스 데이터베이스에서 이 오브젝트를 재작성하려고 하는 사용자에게 유용합니다.

*DBname*이 z/OS용 DB2 데이터베이스이면, db2look 명령의 출력은 다음으로 제한됩니다.

- 테이블, 인덱스, 뷰 및 사용자 정의 구별 유형에 대한 DDL 생성
- 테이블, 컬럼, 컬럼 분포 및 인덱스에 대한 UPDATE 통계 명령문 생성

-e 데이터베이스 오브젝트에 대한 DDL문을 추출합니다. 다음 데이터베이스 오브젝트에 대한 DDL은 -e 옵션을 사용할 때 추출됩니다.

- 감사 규정
- 스키마

- 테이블(해당 테이블과, 파티션된 테이블에 대한 파티션 레벨 INDEX IN절에 대해 하나 이상이 존재하는 경우 인라인 길이 포함)
- 뷰
- 구체화된 쿼리 테이블(MQT)
- 별명
- 인덱스(파티션된 테이블의 파티션된 인덱스 포함)
- 트리거
- 시퀀스
- 사용자 정의 구별 유형
- 기본 키, 참조 무결성 및 점검 제한조건
- 사용자 정의 구조화 유형
- 사용자 정의 함수(UDF)
- 사용자 정의 메소드
- 사용자 정의 변환
- 랩퍼
- 서버
- 사용자 맵핑
- 별칭
- 유형 맵핑
- 함수 템플리트
- 함수 맵핑
- 인덱스 스펙
- 스토어드 프로시저
- 역할
- 트러스트된 컨텍스트
- 전역 변수
- 보안 레이블 구성요소
- 보안 규정
- 보안 레이블

db2look 명령이 생성하는 DDL을 사용하여 사용자 정의 함수(UDF)를 다시 작성할 수 있습니다. 그러나 특정의 사용자 정의 함수가 참조하는 사용자 소스 코드(예: EXTERNAL NAME절)가 사용 가능해야 사용자 정의 함수를 사용할 수 있습니다.

-u *Creator*

작성자 ID. 이 작성자 ID를 가지고 있는 오브젝트로 출력을 제한합니다. 옵션 -a를 지정한 경우 이 매개변수는 무시됩니다. 출력에는 작동 불능 오브젝트가 포함되지 않습니다. 작동 불능 오브젝트를 표시하려면 -a 옵션을 사용하십시오.

-z *schema*

스키마 이름. 이 스키마 이름을 가지고 있는 오브젝트로 출력을 제한합니다. 출력에는 작동 불능 오브젝트가 포함되지 않습니다. 작동 불능 오브젝트를 표시하려면 -a 옵션을 사용하십시오. 이 매개변수가 지정되지 않으면 모든 스키마 이름의 오브젝트가 추출됩니다. -a 옵션을 지정하면 이 매개변수는 무시됩니다. 이 옵션은 페더레이티드 DDL에 대해 무시됩니다.

-t *Tname1 Tname2 ... TnameN*

테이블 이름 목록. 테이블 목록의 특정 테이블로 출력을 제한합니다. 최대 테이블 수는 30입니다. 테이블 이름은 공백으로 구분됩니다. 대소문자 구분 이름과 2바이트 문자 세트(DBCS) 이름은 백슬래시와 큰따옴표 분리문자 안에 묶어야 합니다(예: ₩" MyTabLe ₩"). 다중 단어 테이블 이름의 경우, 따옴표 안에 분리문자를 놓아서(예: "₩"My Table₩"") 명령행 처리기가 쌍을 단어별로 평가하지 않도록 해야 합니다. 다중 단어 테이블 이름을 백슬래시와 큰따옴표 분리문자로 묶지 않으면(예: "My Table"), 모든 단어가 대문자로 변환되고 db2look 명령은 대문자 테이블을 찾습니다(예: "MY TABLE"). -l와 함께 -t를 사용하는 경우, 이 조합은 DB2 버전 9.5에서 파티션된 테이블을 지원하지 않습니다.

-tw *Tname*

*Tname*에 지정된 패턴 기준과 일치하는 테이블 이름에 대한 DDL을 생성합니다. 리턴된 모든 테이블의 모든 종속 오브젝트에 대한 DDL도 생성합니다. *Tname*은 단일 값만 가능합니다. *Tname*의 밑줄 문자(_)는 단일 문자를 나타냅니다. 퍼센트 부호(%)는 0개 이상의 문자로 된 문자열을 나타냅니다. *Tname*의 다른 문자는 단지 문자 자체를 나타냅니다. -tw를 지정한 경우 -t 옵션이 무시됩니다.

-ct 오브젝트 작성 시간별로 DDL을 생성합니다. 오브젝트 작성 시간별로 DDL을 생성하면 일부 오브젝트 DDL이 올바른 종속성 순서로 표시되지 않을 수도 있습니다. db2look 명령은 -ct 옵션도 지정하는 경우 -e, -a, -u, -z, -t, -tw, -v, -l, -noview, -wlm 옵션만 지원합니다.

-dp CREATE문 이전에 DROP문을 생성합니다. 삭제(drop)된 오브젝트에 의존하는 오브젝트가 있는 경우 DROP문이 작동하지 않을 수 있습니다. 예를 들어, 삭제된 스키마에 의존하는 테이블이 있으면 스키마 삭제에 실패하고, 사용자 정의 유형/함수에 의존하는 다른 유형, 함수, 트리거 또는 테이블이 있는 경우 사용자 정의 유형/함수 삭제에 실패합니다. 유형이 지정된 테이블의 경우, 루트 테이블에 대해서만 DROP TABLE HIERARCHY 문이 생성됩니다. DROP문

은 인덱스, 기본 및 외부 키, 제한조건에 대해 생성되지 않습니다. 테이블이 삭제될 때 항상 삭제되기 때문입니다. 테이블에 RESTRICT ON DROP 속성이 있으면 테이블을 삭제할 수 없습니다.

-v *Vname1 Vname2 ... VnameN*
지정된 뷰에 대한 DDL을 생성합니다. 최대 뷰 수는 30입니다. -t 옵션을 지정한 경우 -v 옵션이 무시됩니다. 대소문자 구분, DBCS 및 다중 단어 테이블 이름에 관한 규칙은 뷰 이름에도 적용됩니다.

-h 도움말 정보를 표시합니다. 이 옵션이 지정되면 다른 모든 옵션은 무시되고 도움말 정보만 표시됩니다.

-ap 감사 규정을 다른 데이터베이스 오브젝트와 연관시키는데 필요한 AUDIT USING 문을 생성합니다.

-o *Fname*
출력을 *filename*.sql에 기록합니다. 이 옵션을 지정하지 않으면 출력은 표준 출력에 기록됩니다. 확장자와 함께 파일 이름을 지정하면 출력은 해당 파일에 기록됩니다.

-a 이 옵션을 지정하면 출력은 특정 작성자 ID에 의해 작성된 오브젝트로 제한되지 않습니다. 작동 불능 오브젝트를 포함하여, 모든 사용자가 작성한 모든 오브젝트가 고려됩니다. 예를 들어, -e 옵션과 함께 이 옵션을 지정하면 데이터베이스에 있는 모든 오브젝트에 대해 DDL문이 추출됩니다. -m 옵션과 함께 이 옵션을 지정하면 데이터베이스에 있는 모든 사용자 작성 테이블 및 인덱스에 대해 UPDATE 통계 명령문이 추출됩니다. -u나 -a 모두 지정하지 않으면 환경 변수 USER가 사용됩니다. UNIX 운영 체제에서는 이 변수를 명시적으로 설정하지 않아도 됩니다. 그러나 Windows 시스템에는 USER 환경 변수의 디폴트값이 없습니다. SYSTEM 변수에서 사용자 변수를 설정하거나 세션에 대해 set USER=*username*을 발행해야 합니다.

-m 테이블, 통계 뷰, 컬럼 및 인덱스에 대해 통계를 복제하는 데 사용되는 필수 UPDATE문을 생성합니다.

  -c -m 옵션과 함께 이 옵션을 지정하는 경우, db2look 명령은 COMMIT, CONNECT 및 CONNECT RESET 문을 생성하지 않습니다. 디폴트 조치는 이 명령문을 생성하는 것입니다.

  -r -m 옵션과 함께 이 옵션을 지정하는 경우, db2look 명령은 RUNSTATS 명령을 생성하지 않습니다. 디폴트 조치는 RUNSTATS 명령을 생성하는 것입니다.

  주: 다른 데이터베이스에 대해 가상 갱신 모드(-m 옵션)에서 db2look 명령을 사용하여 작성된 명령 프로세서 스크립트를 실행하려는 경우(예를 들어, 테스트 데이터베이스의 카탈로그 통계가 프로덕션 데이터베이스의 통계와 일치하도록 하기 위해) 두 데이터베이스 모두 동일한 코드 세트 및 지역을 사용해야 합니다.

-l 이 옵션을 지정하는 경우, db2look 명령은 사용자 정의 테이블 스페이스, 데이터베이스 파티션 그룹 및 버퍼 풀에 대해 DDL을 생성합니다. 다음 데이터베이스 오브젝트에 대한 DDL은 -l 옵션을 사용할 때 추출됩니다.

- 사용자 정의 테이블 스페이스
- 사용자 정의 데이터베이스 파티션 그룹
- 사용자 정의 버퍼 풀

-x 이 옵션을 지정하는 경우, db2look 명령은 권한 부여 DDL(예: GRANT문)을 생성합니다.

지원되는 권한은 다음과 같습니다.

- 테이블: ALTER, SELECT, INSERT, DELETE, UPDATE, INDEX, REFERENCE, CONTROL
- 뷰: SELECT, INSERT, DELETE, UPDATE, CONTROL
- 인덱스: CONTROL
- 스키마: CREATEIN, DROPIN, ALTERIN
- 데이터베이스: ACCESSCTRL, BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, DATAACCESS, DBADM, EXPLAIN, IMPLICIT_SCHEMA, LOAD, QUIESCE_CONNECT, SECADM, SQLADM, WLMADM
- 사용자 정의 함수(UDF): EXECUTE
- 사용자 정의 메소드: EXECUTE
- 스토어드 프로시저: EXECUTE
- 패키지: CONTROL, BIND, EXECUTE
- 컬럼: UPDATE, REFERENCES
- 테이블 스페이스: USE
- 시퀀스: USAGE, ALTER
- 워크로드: USAGE
- 전역 변수
- 역할
- 보안 레이블
- 면제

-xd 이 옵션을 지정하는 경우, db2look 명령은 오브젝트 작성 시 SYSIBM에 의해 권한이 부여된 오브젝트에 대한 권한 부여 DDL을 포함하여 모든 권한 부여 DDL을 생성합니다.

-f 쿼리 옵티마이저에 영향을 주는 구성 매개변수와 레지스트리 변수를 추출하려면 이 옵션을 사용하십시오.

-td *delimiter*

db2look 명령으로 생성되는 SQL문의 명령문 분리문자를 지정합니다. 이 옵션을 지정하지 않는 경우 디폴트는 세미콜론(;)입니다. -e 옵션이 지정된 경우 이 옵션을 사용할 것을 권장합니다. 이 경우, 추출된 오브젝트에는 트리거 또는 SQL 루틴이 포함될 수 있습니다.

-noview

이 옵션을 지정하면 CREATE VIEW DDL문이 추출되지 않습니다.

-i *userid*

리모트 데이터베이스에 대해 작업할 때 이 옵션을 사용하십시오.

-w *password*

-i 옵션과 함께 사용하는 경우, 사용자는 이 매개변수를 사용하여 리모트 시스템에 있는 데이터베이스에 대해 db2look 명령을 실행할 수 있습니다. 리모트 시스템에 로그온하기 위해 db2look에서 사용자 ID 및 암호가 사용됩니다. 리모트 데이터베이스에 대해 작업하는 경우, 리모트 데이터베이스의 버전은 로컬 데이터베이스와 같아야 합니다. db2look 명령에는 하위 레벨 또는 상위 레벨 지원이 없습니다.

-wlm 이 옵션은 다음에 대해 CREATE문과 ALTER문을 생성하기 위해 제공할 수 있는 WLM 특정 DDL 출력을 생성합니다.

- 막대 그래프
- WLM 이벤트 모니터
- 서비스 클래스
- 워크로드
- 임계값
- 작업 클래스 세트
- 작업 조치 세트

-wrapper *Wname*

해당 랩퍼에 적용되는 페더레이티드 오브젝트에 대한 DDL문을 생성합니다. 생성될 수 있는 페더레이티드 DDL문은 CREATE WRAPPER, CREATE SERVER, CREATE USER MAPPING, CREATE NICKNAME, CREATE TYPE MAPPING, CREATE FUNCTION ... AS TEMPLATE, CREATE FUNCTION MAPPING, CREATE INDEX SPECIFICATION, GRANT(별명, 서버, 인덱스에 특권 부여)입니다. 하나의 랩퍼 이름만 지원됩니다. 하나 미만 또는 두 개 이상의 이름을 지정하면 오류가 리턴됩니다. 이 옵션은 관계형이 아닌 데이터 소스를 지원하지 않습니다.

-server *Sname*

해당 서버에 적용되는 페더레이티드 오브젝트에 대한 DDL문을 생성합니다. 생성될 수 있는 페더레이티드 DDL문은 CREATE WRAPPER, CREATE

SERVER, CREATE USER MAPPING, CREATE NICKNAME, CREATE TYPE MAPPING, CREATE FUNCTION ... AS TEMPLATE, CREATE FUNCTION MAPPING, CREATE INDEX SPECIFICATION, GRANT(별명, 서버, 인덱스에 특권 부여)입니다. 하나의 서버 이름만 지원됩니다. 하나 미만 또는 두 개 이상의 이름을 지정하면 오류가 리턴됩니다. 이 옵션은 관계형이 아닌 데이터 소스를 지원하지 않습니다.

-nofed 페더레이티드 DDL문이 생성되지 않음을 지정합니다. 이 옵션을 지정하면 -wrapper 및 -server 옵션은 무시됩니다.

-fedonly
페더레이티드 DDL문만 생성됨을 지정합니다.

-mod 각 모듈 및 각 모듈에 정의된 모든 오브젝트에 대해 DDL을 생성합니다.

-xs 목표 데이터베이스에서 XML 스키마와 DTD를 등록하는 데 필요한 모든 파일을 익스포트하고 파일 등록에 적절한 명령을 생성합니다. 익스포트되는 XSR 오브젝트 세트는 -u, -z 및 -a 옵션으로 제어됩니다.

-xdir *dirname*
익스포트된 XML 관련 파일을 제공된 경로에 위치시킵니다. 이 옵션이 지정되지 않으면, 모든 XML 관련 파일이 현재 디렉토리로 익스포트됩니다.

-cor CREATE OR REPLACE절로 DDL문을 생성하며 이 경우 해당 절에 처음부터 해당 문이 포함되어 있는지 여부는 상관이 없습니다.

예:

- 데이터베이스 DEPARTMENT에서 사용자 walid가 작성한 오브젝트에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

```
db2look -d department -u walid -e -o db2look.sql
```

- 데이터베이스 DEPARTMENT에서 사용자 walid가 작성한, 스키마 이름이 ianhe인 오브젝트에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

```
db2look -d department -u walid -z ianhe -e -o db2look.sql
```

- 데이터베이스 DEPARTMENT에서 사용자 walid가 작성한 데이터베이스 오브젝트에 대한 통계를 복제하기 위한 UPDATE문을 생성합니다. 출력은 db2look.sql 파일로 보냅니다.

```
db2look -d department -u walid -m -o db2look.sql
```

- 사용자 walid가 작성한 오브젝트에 대한 DDL문과 동일한 사용자가 작성한 데이터베이스 오브젝트에 대한 통계를 복제하기 위한 UPDATE문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

```
db2look -d department -u walid -e -m -o db2look.sql
```

- 데이터베이스 DEPARTMENT에서 모든 사용자가 작성한 오브젝트에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -a -e -o db2look.sql
  ```

- 모든 사용자 정의 데이터베이스 파티션 그룹, 버퍼 풀 및 테이블 스페이스에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -l -o db2look.sql
  ```

- 옵티마이저 관련 데이터베이스 및 데이터베이스 관리 프로그램 구성 매개변수에 대한 UPDATE문과, 옵티마이저 관련 레지스트리 변수에 대한 db2set문을 데이터베이스 DEPARTMENT에서 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -f -o db2look.sql
  ```

- 데이터베이스 DEPARTMENT에서 모든 오브젝트에 대한 DDL, 데이터베이스 DEPARTMENT에서 모든 테이블 및 인덱스에 대한 통계를 복제하기 위한 UPDATE문, GRANT 권한 부여 명령문, 옵티마이저 관련 데이터베이스 및 데이터베이스 관리 프로그램 구성 매개변수에 대한 UPDATE문, 옵티마이저 관련 레지스트리 변수에 대한 db2set문, 그리고 데이터베이스 DEPARTMENT에서 모든 사용자 정의 데이터베이스 파티션 그룹, 버퍼 풀 및 테이블 스페이스에 대한 DDL을 생성합니다. 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -a -e -m -l -x -f -o db2look.sql
  ```

- 원래 작성자가 작성한 오브젝트를 포함하여, 데이터베이스 DEPARTMENT에서 모든 오브젝트에 대한 모든 권한 부여 DDL문을 생성합니다. (이 경우, 권한 부여는 오브젝트 작성 시 SYSIBM에 의해 부여되었습니다.) db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -xd -o db2look.sql
  ```

- 데이터베이스 DEPARTMENT에서 모든 사용자가 작성한 오브젝트에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -a -e -td % -o db2look.sql
  ```

  출력은 CLP에서 읽을 수 있습니다.

  ```
  db2 -td% -f db2look.sql
  ```

- CREATE VIEW 문을 제외하고, 데이터베이스 DEPARTMENT에서 오브젝트에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -e -noview -o db2look.sql
  ```

- 데이터베이스 DEPARTMENT에서 지정된 테이블에 관련된 오브젝트에 대한 DDL문을 생성합니다. db2look 출력은 db2look.sql 파일로 보냅니다.

  ```
  db2look -d department -e -t tab1 ₩"My TaBlE2₩" -o db2look.sql
  ```

- 페더레이티드 데이터베이스 FEDDEPART에서 모든 오브젝트(페더레이티드 및 비페더레이티드)에 대한 DDL문을 생성합니다. 페더레이티드 DDL문의 경우, 지정된 랩퍼 FEDWRAP에 적용되는 명령문만 생성됩니다. db2look 출력은 표준 출력으로 보냅니다.

  ```
  db2look -d feddepart -e -wrapper fedwrap
  ```

- 비페더레이티드 DDL문만 포함하는 스크립트 파일을 생성합니다. 다음 시스템 명령은 페더레이티드 데이터베이스(FEDDEPART)에 대해 실행될 수 있으며 아직은 페더레이티드 데이터베이스가 아닌 데이터베이스에 대해 실행될 때 발견되는 것과 같은 출력만 생성합니다. db2look 출력은 `out.sql` 파일로 보냅니다.

  ```
  db2look -d feddepart -e -nofed -o out
  ```

- 데이터베이스 DEPARTMENT에서 스키마 이름이 `walid`인 오브젝트에 대한 DDL문을 생성합니다. 포함된 XML 스키마와 DTD를 등록하는 데 필요한 파일이 현재 디렉토리로 익스포트됩니다. db2look 출력은 `db2look.sql` 파일로 보냅니다.

  ```
  db2look -d department -z walid -e -xs -o db2look.sql
  ```

- 데이터베이스 DEPARTMENT에서 모든 사용자가 작성한 오브젝트에 대한 DDL문을 생성합니다. 포함된 XML 스키마와 DTD를 등록하는 데 필요한 파일이 `/home/ofer/ofer/` 디렉토리로 익스포트됩니다. db2look 출력은 표준 출력으로 보냅니다.

  ```
  db2look -d department -a -e -xs -xdir /home/ofer/ofer/
  ```

- 데이터베이스 DEPARTMENT에서 독점적으로 WLM 특정 DDL을 생성합니다.

  ```
  db2look -d department -wlm
  ```

  데이터베이스 DEPARTMENT에서 모든 오브젝트에 대한 DDL을 생성합니다.

  ```
  db2look -d department -wlm -e -l
  ```

사용 시 참고사항

Windows 운영 체제에서는 DB2 명령 창에서 db2look 명령을 실행해야 합니다.

기존 옵션 중 몇 개의 옵션은 페더레이티드 환경을 지원합니다. 다음 db2look 명령행 옵션은 페더레이티드 환경에서 사용됩니다.

- -ap

  사용된 경우, AUDIT USING 문이 생성됩니다.

- -e

  사용된 경우, 페더레이티드 DDL문이 생성됩니다.

- -x

  사용된 경우, 페더레이티드 오브젝트에 특권을 부여하기 위한 GRANT문이 생성됩니다.

- -xd

  사용된 경우, 페더레이티드 오브젝트에 시스템에서 부여한 특권을 추가하기 위한 페더레이티드 DDL문이 생성됩니다.

- -f

  사용된 경우, 데이터베이스 관리 프로그램 구성에서 페더레이티드 관련 정보가 추출됩니다.

- -m

  사용된 경우, 별칭에 대한 통계가 추출됩니다.

- -wlm

  사용된 경우, WLM 특정 DDL이 출력됩니다.

페더레이티드 DDL문을 작성하려면 데이터베이스 관리 프로그램 구성에서 페더레이티드 시스템을 사용할 수 있는 기능을 사용 가능하도록 설정해야 합니다. db2look 명령이 스크립트 파일을 생성하면, **federated** 구성 매개변수를 YES로 설정한 후 스크립트를 실행해야 합니다.

CREATE USER MAPPING 문의 리모트 암호를 추가하도록 출력 스크립트를 수정해야 합니다.

DB2 계열 인스턴스를 데이터 소스로 정의하기 위해 사용되는 CREATE SERVER 문에 AUTHORIZATION 및 PASSWORD를 추가하여 db2look 명령 출력 스크립트를 수정해야 합니다.

-tw 옵션은 다음과 같이 사용합니다.

- DEPARTMENT 데이터베이스에서 이름이 abc로 시작하는 테이블과 연관되는 오브젝트에 대한 DDL문을 생성하고 출력을 db2look.sql 파일로 보내려면 다음 명령을 사용하십시오.

  ```
  db2look -d department -e -tw abc% -o db2look.sql
  ```

- DEPARTMENT 데이터베이스에서 이름의 두 번째 문자가 d인 테이블과 연관되는 오브젝트에 대한 DDL문을 생성하고 출력을 db2look.sql 파일로 보내려면 다음 명령을 사용하십시오.

  ```
  db2look -d department -e -tw _d% -o db2look.sql
  ```

- db2look 명령은 *Tname* 인수에 지정된 패턴과 일치하는 테이블 이름을 평가할 때 LIKE 술어를 사용합니다. LIKE 술어가 사용되므로, _ 문자나 % 문자가 테이블 이름의 일부인 경우 백슬래시(₩) Escape 문자를 _ 또는 % 바로 앞에 사용해야 합니다. 이 경우, _과 %는 *Tname*에서 와일드 카드 문자로 사용할 수 없습니다. 예를

들어, DEPARTMENT 데이터베이스에서 이름의 첫 번째 및 마지막 위치에 퍼센트 부호없는 테이블과 연관되는 오브젝트에 대한 DDL문을 생성하려면 다음 명령을 사용하십시오.

```
db2look -d department -e -tw string₩%string
```

- 대소문자 구분, DBCS 및 다중 단어 테이블 및 뷰 이름은 백슬래시와 큰따옴표로 묶어야 합니다. 예를 들면, 다음과 같습니다.

  ```
  ₩"My TabLe₩"
  ```

  멀티바이트 문자 세트(MBCS) 또는 2바이트 문자 세트(DBCS) 이름이 백슬래시와 큰따옴표 분리문자로 묶이지 않고, 소문자와 같은 바이트를 포함하는 경우, 대문자로 변환되므로 db2look 명령은 변환된 이름의 데이터베이스 오브젝트를 찾습니다. 결과적으로, DDL문은 추출되지 않습니다.

- -tw 옵션은 -x 옵션(GRANT 특권을 생성하는 경우), -m 옵션(테이블 및 컬럼 통계를 리턴하는 경우) 및 -l 옵션(사용자 정의 테이블 스페이스, 데이터베이스 파티션 그룹 및 버퍼 풀에 대한 DDL을 생성하는 경우)과 함께 사용할 수 있습니다. -t 옵션을 -tw 옵션과 함께 지정하면, -t 옵션(및 연관된 *Tname* 인수)은 무시됩니다.
- -tw 옵션은 페더레이티드 데이터 소스나, z/OS용 DB2, i용 DB2 또는 VSE & VM용 DB2 서버에 있는 테이블(및 해당되는 연관 오브젝트)에 대한 DDL을 생성하는 데 사용할 수 없습니다.
- -tw 옵션은 CLP를 통해서만 지원됩니다.

데이터베이스 파티셔닝 기능을 사용하여 시스템에서 DDL을 요청하는 경우, 비활성 데이터베이스 파티션에 존재하는 테이블 스페이스에 대한 DDL 대신 경고 메시지가 표시됩니다. 모든 테이블 스페이스에 대해 적절한 DDL이 생성되도록 하려면 모든 데이터베이스 파티션을 활성화해야 합니다.

배열 유형의 보안 레이블 구성요소에 대한 DDL을 추출할 때, 추출된 DDL은 내부 표현(즉, 해당 배열에서의 요소 인코딩)이, db2look 추출이 수행된 데이터베이스 내의 해당되는 보안 레이블 구성요소의 내부 표현과 일치하는 보안 레이블 구성요소를 생성할 수 없습니다. 이는 배열 유형의 보안 레이블 구성요소가 변경되고 하나 이상의 요소가 추가된 경우에 발생할 수 있습니다. 이와 같은 경우, 하나의 테이블에서 추출되어 다른 테이블(db2look 출력에서 작성된)로 이동된 데이터에는 해당되는 보안 레이블 값이 수반되지 않으므로 새 테이블 보호가 손상될 수 있습니다.

관련 정보

별칭 컬럼 및 인덱스 이름

이주용 응용프로그램 변경

제 6 장 파일 형식 및 데이터 유형

익스포트/임포트/로드 유틸리티 파일 형식

DB2 익스포트, 임포트 및 로드 유틸리티에서 지원하는 다섯 가지 운영 체제 파일 형식을 설명합니다.

DEL 컬럼 식별자가 있는 ASCII. 다양한 데이터베이스 관리 프로그램 및 파일 관리 프로그램 사이에서의 데이터 교환에 적합합니다. 데이터를 저장하는 이 공통 접근 방식은 특수 문자 분리문자를 사용하여 컬럼 값을 분리합니다.

ASC 컬럼 식별자 없는 ASCII. 정렬된 컬럼 데이터를 포함하는 일반 텍스트 파일을 작성하는 다른 응용프로그램에서 데이터를 로드하거나 임포트하는 경우에 적합합니다.

PC/IXF

IXF(Integration Exchange Format)의 PC 버전. 데이터베이스 관리 프로그램에서 선호되는 데이터 교환 방법입니다. PC/IXF는 내부 테이블의 외부 표현을 포함하는 데이터베이스 테이블의 구조화된 설명입니다.

WSF 워크시트 형식. Lotus 1-2-3 및 Symphony와 같은 제품에서 데이터 교환에 적합합니다. 로드 유틸리티에서는 이 파일 형식을 지원하지 않습니다.

이 형식은 사용되지 않으며 추후 릴리스에서 제거됩니다.

CURSOR

SQL 쿼리에서 선언된 cursor입니다. 이 파일 유형은 로드 유틸리티에서만 지원됩니다.

DEL, WSF 또는 ASC 데이터 파일 형식을 사용하는 경우 파일을 임포트하기 전에 해당 컬럼 이름 및 데이터 유형을 포함하여 테이블을 정의합니다. 운영 체제 파일 필드의 데이터 유형은 데이터베이스 테이블의 해당 데이터 유형으로 변환됩니다. 임포트 유틸리티는 가능한 채우기 또는 절단이 적용되어 임포트된 문자 데이터 및 다른 유형의 숫자 필드로 임포트된 숫자 데이터를 포함하여 사소한 비호환 문제점이 있는 데이터를 승인합니다.

PC/IXF 데이터 파일 형식을 사용하는 경우 임포트 조작을 시작하기 전에 테이블은 미리 존재하지 않아도 됩니다. 그러나 사용자 정의 구별 유형(UDT)은 정의해야 합니다. 그렇지 않으면 정의되지 않은 이름 오류(SQL0204N)가 수신됩니다. 마찬가지로 PC/IXF 데이터 파일 형식을 익스포트하는 경우 출력 파일에 UDT가 저장됩니다.

CURSOR 파일 유형을 사용하는 경우 로드 조작을 시작하기 전에 해당 컬럼 이름 및 데이터 유형을 포함하여 테이블을 정의해야 합니다. SQL 쿼리의 컬럼 유형은 목표 테

이블의 대응하는 컬럼 유형과 호환 가능해야 합니다. 로드 조작을 시작하기 전에 지정된 cursor를 열지 않아도 됩니다. 로드 유틸리티는 cursor를 사용하여 행을 페치했는지 여부에 상관없이 지정된 cursor와 연관된 쿼리의 전체 결과를 처리합니다.

플랫폼 교차 데이터 이동 - 파일 형식 고려사항

호환성은 여러 플랫폼 사이에서 데이터를 익스포트, 임포트 또는 로드할 경우 매우 중요합니다. 다음 절에서는 서로 다른 운영 체제에서 데이터를 이동하는 경우 PC/IXF, 컬럼 식별자가 있는 ASCII(DEL) 및 WSF 파일 형식 고려사항에 대해 설명합니다.

PC/IXF 파일 형식

PC/IXF는 여러 플랫폼 간 데이터를 이동하는 경우 권장되는 파일 형식입니다. PC/IXF 파일을 사용하면 머신에 독립된 방식으로 로드 유틸리티 또는 임포트 유틸리티에서 보통 머신에 종속된 숫자 데이터를 처리할 수 있습니다. 예를 들어 숫자 데이터는 Intel® 및 기타 하드웨어 아키텍처에서 서로 다르게 저장 및 처리됩니다.

DB2 계열의 모든 제품에서 PC/IXF 파일의 호환성을 제공하기 위해 익스포트 유틸리티는 Intel 형식으로 숫자 데이터를 포함한 파일을 작성하고 임포트 유틸리티는 이 형식을 예상합니다.

하드웨어 플랫폼에 따라 DB2 제품은 익스포트 및 임포트 조작 중 Intel 및 Intel 이외 형식(바이트 리버설 사용) 사이에서 숫자 값을 변환합니다.

DB2 데이터베이스의 UNIX 기반 구현에서는 익스포트 중 다중 파트로 구성된 PC/IXF 파일을 작성하지 않습니다. 그러나 DB2에서 작성한 다중 파트 PC/IXF 파일을 임포트할 수는 있습니다. 이러한 유형의 파일을 임포트하는 경우 모든 파트가 동일한 디렉토리에 있어야 합니다. 그렇지 않으면 오류가 리턴됩니다.

DB2 익스포트 유틸리티의 UNIX 기반 구현으로 작성된 단일 파트 PC/IXF 파일은 Windows용 DB2 데이터베이스에서 임포트할 수 있습니다.

컬럼 식별자가 있는 ASCII(DEL) 파일 형식

DEL 파일은 작성된 운영 체제에 따라 다른 점이 있습니다. 다음은 다른 점입니다.

- 행 구분문자
  - UNIX 기반 텍스트 파일은 줄 바꾸기(LF) 문자를 사용합니다.
  - 비UNIX 기반 텍스트 파일은 캐리지 리턴/줄 바꾸기(CRLF) 시퀀스를 사용합니다.
- EOF 문자
  - UNIX 기반 텍스트 파일은 EOF 문자를 포함하지 않습니다.
  - 비UNIX 기반 텍스트 파일은 EOF 문자(X'1A')를 포함합니다.

DEL 익스포트 파일은 텍스트 파일이므로 한 운영 체제에서 다른 운영 체제로 전송할 수 있습니다. 파일 전송 프로그램은 텍스트 모드로 파일을 전송할 때 운영 체제에 종속된 다른 점을 처리할 수 있습니다. 2진 모드에서 행 구분자 및 EOF 문자의 변환은 수행되지 않습니다.

주: 문자 데이터 필드에 행 구분자가 포함된 경우 파일 전송 중에도 변환됩니다. 이 변환으로 데이터가 예기치 않게 변경될 수 있으므로 여러 플랫폼 사이에서 데이터를 이동하는 경우 DEL 익스포트 파일은 사용하지 않습니다. 대신 PC/IXF 파일 형식을 사용합니다.

WSF 파일 형식

WSF 형식 파일의 숫자 데이터는 Intel 머신 형식을 사용하여 저장됩니다. 이 형식을 사용하면 Lotus WSF 파일을 다른 Lotus 운영 환경(예: Intel 기반 및 UNIX 기반 시스템)에서 전송 및 사용할 수 있습니다.

주: 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

내부 형식의 이러한 일관성으로 DB2 제품에서 익스포트된 WSF 파일은 다른 플랫폼에서 실행하는 Lotus 1-2-3 또는 Symphony에서 사용될 수 있습니다. DB2 제품은 서로 다른 플랫폼에서 작성된 WSF 파일도 임포트할 수 있습니다.

텍스트 모드가 아닌 2진 모드에서 운영 체제 사이에 WSF 파일을 전송합니다.

주: 다른 플랫폼의 DB2 데이터베이스 사이에서 데이터를 전송할 때 WSF 파일 형식은 사용하지 않습니다. 데이터가 유실될 수 있기 때문입니다. 대신 PC/IXF 파일 형식을 사용합니다.

컬럼 식별자가 있는 ASCII(DEL) 파일 형식

컬럼 식별자가 있는 ASCII(DEL) 파일은 행 및 컬럼 분리문자를 포함하는 순차 ASCII 파일입니다. 각 DEL 파일은 행 및 컬럼 순서로 정렬된 셀 값으로 구성된 ASCII 문자의 스트림입니다. 데이터 스트림의 행은 행 분리문자로 구분되며 각 행에서 개별 셀 값은 컬럼 분리문자로 구분됩니다.

다음 표는 임포트할 수 있거나 익스포트 조치로 생성할 수 있는 DEL 파일의 형식에 대해 설명합니다.

```
DEL file ::= Row 1 data || Row delimiter ||
             Row 2 data || Row delimiter ||
             .
             .
             .
             Row n data || Optional row delimiter
```

```
Row i data ::= Cell value(i,1) || Column delimiter ||
               Cell value(i,2) || Column delimiter ||
               .
               .
               .
               Cell value(i,m)

Row delimiter ::= ASCII line feed sequence[a]

Column delimiter ::= Default value ASCII comma (,)[b]

Cell value(i,j) ::= Leading spaces
                 || ASCII representation of a numeric value
                    (integer, decimal, or float)
                 || Delimited character string
                 || Non-delimited character string
                 || Trailing spaces

Non-delimited character string ::= A set of any characters except a
                                   row delimiter or a column delimiter

Delimited character string ::= A character string delimiter ||
                               An extended character string ||
                               A character string delimiter ||
                               Trailing garbage

Trailing garbage ::= A set of any characters except a row delimiter
                     or a column delimiter

Character string delimiter ::= Default value ASCII double quotation
                               marks (")[c]

extended character string ::= || A set of any characters except a
                                 row delimiter or a character string
                                 delimiter if the NODOUBLEDEL
                                 modifier is specified
                              || A set of any characters except a
                                 row delimiter or a character string
                                 delimiter if the character string
                                 is not part of two consecutive
                                 character string delimiters
                              || A set of any characters except a
                                 character string delimiter if the
                                 character string delimiter is not
                                 part of two consecutive character
                                 string delimiters, and the DELPRIORITYCHAR
                                 modifier is specified

End-of-file character ::= Hex '1A' (Windows operating system only)

ASCII representation of a numeric value[d] ::= Optional sign '+' or '-'
    || 1 to 31 decimal digits with an optional decimal point before,
       after, or between two digits
    || Optional exponent

Exponent ::= Character 'E' or 'e'
    || Optional sign '+' or '-'
    || 1 to 3 decimal digits with no decimal point

Decimal digit ::= Any one of the characters '0', '1', ... '9'

Decimal point ::= Default value ASCII period (.)[e]
```

- [a] 레코드 구분 문자는 개행 문자, ASCII x0A로 가정합니다. Windows 운영 체제에서 생성된 데이터는 캐리지 리턴/줄 바꾸기 2바이트 표준 0x0D0A를 사용할 수 있

습니다. EBCDIC 코드 페이지의 데이터는 EBCDIC LF 문자(0x25)를 레코드 구분 문자로 사용해야 합니다. LOAD 명령에서 codepage 파일 유형 수정자를 사용하여 EBCDIC 데이터를 로드할 수 있습니다.

- [b] 컬럼 분리문자는 coldel 파일 유형 수정자로 지정할 수 있습니다.
- [c] 문자열 분리문자는 chardel 파일 유형 수정자로 지정할 수 있습니다.

  주: 분리문자의 디폴트 우선순위는 다음과 같습니다.

  1. 레코드 구분 문자
  2. 문자 분리문자
  3. 컬럼 분리문자

- [d] 숫자 값의 ASCII 표현이 지수를 포함하는 경우 FLOAT 상수입니다. 소수점이 있지만 지수가 없으면 DECIMAL(10진) 상수입니다. 소수점 및 지수가 없으면 INTEGER(정수) 상수입니다.
- [e] 소수점 문자는 decpt 파일 유형 수정자로 지정할 수 있습니다.

익스포트 유틸리티는 열 데이터에 임베드된 모든 문자열 분리문자 바이트(디폴트는 큰따옴표 또는 x22임)를 두 개의 문자열 분리문자 바이트(이중 사용)로 교체합니다. 임포트 구문 분석 루틴이 컬럼의 시작 또는 끝을 정의하는 문자열 분리문자 바이트와 열 데이터에 임베드된 문자열 분리문자 바이트를 구별할 수 있도록 하기 위해서입니다. 익스포트 유틸리티 이외의 다른 응용프로그램에서 익스포트한 DEL 파일을 사용하는 경우 주의해야 합니다. 'FOR BIT' 2진 열 데이터에서 동일한 이중 문자열 분리문자 사용이 나타나는지 주의합니다.

DEL 데이터 유형 설명

표 50. DEL 파일 형식에 대해 승인할 수 있는 데이터 유형 양식

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
BIGINT	-9 223 372 036 854 775 808에서 9 223 372 036 854 775 807 사이의 범위에 포함된 INTEGER 상수.	-9 223 372 036 854 775 808에서 9 223 372 036 854 775 807 사이의 범위에 포함된 숫자 값을 ASCII로 표시합니다. 10진수 및 부동수는 정수값으로 절단됩니다.
BLOB, CLOB	문자 분리문자(예: 큰따옴표)로 묶인 문자 데이터.	컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열. 문자열은 데이터베이스 컬럼 값으로 사용됩니다.
BLOB_FILE, CLOB_FILE	각 BLOB/CLOB 컬럼의 문자 데이터는 개별 파일에 저장되고 파일 이름은 문자 분리문자로 묶입니다.	데이터를 보유하는 파일의 컬럼 식별자가 있는 또는 컬럼 식별자 없는 이름.

표 50. DEL 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
CHAR	문자 분리문자(예: 큰따옴표)로 묶인 문자 데이터.	컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열. 필요한 경우 데이터베이스 컬럼 너비와 일치시키기 위해 문자열은 스페이스(X'20')로 채워지거나 절단됩니다.
DATE	문자 분리문자가 없는 *yyyymmdd* (연도, 월, 일). 예를 들어 다음과 같습니다. 19931029 또는 DATESISO 옵션을 사용하여 모든 데이터 값을 ISO 형식으로 익스포트하도록 지정할 수도 있습니다.	*yyyymmdd* 양식의 컬럼 식별자 없는 문자열 또는 목표 데이터베이스의 지역 코드와 일관된 ISO 형식의 데이터 값을 포함하는 컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열.
DBCLOB(DBCS만 해당)	그래픽 데이터는 구분된 문자열로 익스포트됩니다.	구분 또는 컬럼 식별자 없는 문자열로, 길이(바이트)가 짝수입니다. 문자열은 데이터베이스 컬럼 값으로 사용됩니다.
DBCLOB_FILE(DBCS만 해당)	각 DBCLOB 컬럼의 문자 데이터는 개별 파일에 저장되고 파일 이름은 문자 분리문자로 묶입니다.	데이터를 보유하는 파일의 컬럼 식별자가 있는 또는 컬럼 식별자 없는 이름.
DB2SECURITYLABEL	열 데이터는 인용 부호(")로 묶인 "원시" 데이터로 익스포트됩니다. SELECT 문에서 SECLABEL_TO_CHAR 스칼라 함수를 사용하여 보안 레이블 문자열 형식으로 값을 변환합니다.	데이터 파일의 값은 디폴트로 해당 보안 레이블의 내부 표현을 구성하는 실제 바이트로 가정합니다. 값은 인용 부호(" ")로 구분된다고 가정합니다.
DECIMAL	필드의 정밀도 및 스케일을 익스포트하는 DECIMAL 상수. decplusblank 파일 유형 수정자를 사용하여 양의 10진수 값에서 플러스 부호(+) 대신, 공백이 앞에 오도록 지정할 수 있습니다.	필드를 임포트하는 데이터베이스 컬럼의 범위를 오버플로우하지 않는 숫자 값을 나타내는 ASCII 표현. 입력 값의 소수점 뒤로 데이터베이스 컬럼에서 수용할 수 있는 것보다 더 많은 자릿수가 있으면 초과된 자릿수가 절단됩니다.
FLOAT(long)	-10E307에서 10E307 사이의 FLOAT 상수.	-10E307에서 10E307 사이의 숫자 값을 나타내는 ASCII 표현.
GRAPHIC(DBCS만 해당)	그래픽 데이터는 구분된 문자열로 익스포트됩니다.	구분 또는 컬럼 식별자 없는 문자열로, 길이(바이트)가 짝수입니다. 필요한 경우 데이터베이스 컬럼 너비와 일치시키기 위해 문자열은 2바이트 스페이스(X'8140')로 채워지거나 절단됩니다.

표 50. DEL 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
INTEGER	-2 147 483 648에서 2 147 483 647 사이의 INTERGER 상수.	-2 147 483 648에서 2 147 483 647 사이의 숫자 값을 나타내는 ASCII 표현. 10진수 및 부동수는 정수값으로 절단됩니다.
LONG VARCHAR	문자 분리문자(예: 큰따옴표)로 묶인 문자 데이터.	컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열. 문자열은 데이터베이스 컬럼 값으로 사용됩니다.
LONG VARGRAPHIC(DBCS만 해당)	그래픽 데이터는 구분된 문자열로 익스포트됩니다.	구분 또는 컬럼 식별자 없는 문자열로, 길이(바이트)가 짝수입니다. 문자열은 데이터베이스 컬럼 값으로 사용됩니다.
SMALLINT	-32 768에서 32 767 사이의 INTEGER 상수.	-32 768에서 32 767 사이의 숫자 값을 나타내는 ASCII 표현. 10진수 및 부동수는 정수값으로 절단됩니다.
TIME	*hh.mm.ss*(시간, 분, 초). ISO 형식의 시간 값은 문자 분리문자로 묶습니다. 예를 들어 다음과 같습니다. ″09.39.43″	목표 데이터베이스의 지역 코드와 일관된 형식의 시간 값을 포함하는 컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열.
TIMESTAMP	*yyyy-mm-dd-hh.mm.ss.nnnnnn*(연도, 월, 일, 시간, 분, 초, 마이크로초). 문자열은 문자 분리문자로 묶인 날짜 및 시간을 나타냅니다.	데이터베이스의 스토리지에서 승인할 수 있는 시간소인 값을 포함하는 컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열.
VARCHAR	문자 분리문자(예: 큰따옴표)로 묶인 문자 데이터.	컬럼 식별자가 있는 또는 컬럼 식별자 없는 문자열. 필요한 경우 데이터베이스 컬럼의 최대 너비와 일치시키기 위해 문자열이 절단됩니다.
VARGRAPHIC(DBCS만 해당)	그래픽 데이터는 구분된 문자열로 익스포트됩니다.	구분 또는 컬럼 식별자 없는 문자열로, 길이(바이트)가 짝수입니다. 필요한 경우 데이터베이스 컬럼의 최대 너비와 일치시키기 위해 문자열이 절단됩니다.

DEL 파일 예

다음은 DEL 파일에 대한 예입니다. 각 라인은 줄 바꾸기 시퀀스로 종료됩니다. Windows 운영 체제의 경우 각 라인은 캐리지 리턴/줄 바꾸기 시퀀스로 종료됩니다.

```
"Smith, Bob",4973,15.46
"Jones, Bill",12345,16.34
"Williams, Sam",452,193.78
```

다음 예에서는 컬럼 식별자 없는 문자열 사용을 보여줍니다. 컬럼 분리문자는 문자 데이터에 쉼표가 포함되었으므로 세미콜론으로 변경됩니다.

```
Smith, Bob;4973;15.46
Jones, Bill;12345;16.34
Williams, Sam;452;193.78
```

주:

1. 스페이스(X'20')는 유효한 분리문자가 아닙니다.
2. 첫 번째 문자 앞에 나오거나 셀 값의 마지막 문자 뒤에 오는 스페이스는 임포트 중 버려집니다. 셀 값에 임베드된 스페이스는 버리지 않습니다.
3. 마침표(.)는 유효한 문자열 분리문자가 아닙니다. 시간소인 값의 마침표와 충돌하기 때문입니다.
4. 순수 DBCS(그래픽), 혼합 DBCS 및 EUC의 경우, 분리문자는 x00 - x3F의 범위로 제한됩니다.
5. EBCDIC 코드 페이지에 지정된 DEL 데이터의 경우 분리문자는 시프트 인(Shift-In) 및 시프트 아웃(Shift-Out) DBCS 문자와 동시에 사용할 수 없습니다.
6. Windows 운영 체제의 경우 문자 분리문자에 없는 EOF 문자(X'1A')가 처음 나타나면 이는 EOF를 나타냅니다. 후속 데이터는 임포트되지 않습니다.
7. 널(NULL) 값은 정상적으로 하나가 나타나는 경우 셀 값의 부재 또는 스페이스 문자열로 표시됩니다.
8. 일부 제품에서는 문자 필드를 254 또는 255바이트로 제한하므로 익스포트할 때 최대 길이가 254바이트가 넘는 문자 컬럼을 선택하면 익스포트 유틸리티에서 경고 메시지가 생성됩니다. 임포트 유틸리티는 가장 긴 LONG VARCHAR 및 LONG VARGRAPHIC 컬럼과 같이 긴 필드를 사용합니다.

데이터 이동 시 분리문자 고려사항

컬럼 식별자가 있는 ASCII(DEL) 파일을 이동하는 경우 구분 문자 인식 문제점 때문에 이동할 데이터가 우연히 변경되지 않도록 해야 합니다. 이러한 오류를 방지하기 위해 DB2에서는 여러 제한사항을 시행하고 여러 파일 유형 수정자를 제공합니다.

분리문자 제한사항

선택한 구분 문자를 이동할 데이터로 처리하지 않도록 하는 여러 제한사항이 있습니다. 먼저 분리문자는 상호 배타적입니다. 두 번째로 분리문자는 줄 바꾸기 문자, 캐리지 리턴 또는 공백일 수 없습니다. 또한 디폴트 소수점(.)은 문자열 분리문자일 수 없습니다. 마지막으로 DBCS 환경에서 파이프(|) 문자 분리문자는 지원되지 않습니다.

다음 문자는 ASCII 계열 코드 페이지 및 EBCDIC 계열 코드 페이지에서 서로 다르게 지정됩니다.

- 시프트 인(Shift-In) 및 시프트 아웃(Shift-Out) 문자는 EBCDIC MBCS 데이터 파일에서 분리문자로 사용할 수 없습니다.
- MBCS, EUC 또는 DBCS 코드 페이지의 분리문자는 0x40보다 클 수 없습니다. 단, EBCDIC MBCS 데이터의 디폴트 소수점(0x4b)은 제외됩니다.
- ASCII 코드 페이지 또는 EBCDIC MBCS 코드 페이지에서 데이터 파일의 디폴트 분리문자는 다음과 같습니다.
  - 문자열 분리문자: "(0x22, 큰따옴표)
  - 컬럼 분리문자: **,**(0x2c, 쉼표)
- EBCDIC SBCS 코드 페이지의 데이터 파일에 대한 디폴트 분리문자는 다음과 같습니다.
  - 문자열 분리문자: "(0x7F, 큰따옴표 기호)
  - 컬럼 분리문자: **,**(0x6B, 쉼표)
- ASCII 데이터 파일의 디폴트 소수점은 0x2e(마침표)입니다.
- EBCDIC 데이터 파일의 디폴트 소수점은 0x4B(마침표)입니다.
- 서버의 코드 페이지가 클라이언트의 코드 페이지와 다른 경우 디폴트가 아닌 분리문자의 16진 표현을 지정해야 합니다. 예를 들면, 다음과 같습니다.

```
db2 load from ... modified by chardel0x0C coldelX1e ...
```

데이터 이동 중 분리문자에 대한 문제

2바이트 분리문자

디폴트로 DEL 파일의 문자 기반 필드에서 필드에 있는 문자 분리문자의 인스턴스는 2바이트 분리문자로 표시합니다. 예를 들어 문자 분리문자가 큰따옴표라고 가정하고 텍스트 I am 6″ tall.을 익스포트하면 DEL 파일의 출력 텍스트는 ″I am 6″″ tall.″을 읽습니다. 반대로 DEL 파일의 입력 텍스트가 ″What a ″″nice″″ day!″를 읽으면 텍스트는 What a ″nice″ day!

로 임포트됩니다. **nodoubledel**
2바이트 분리문자 동작은 nodoubledel 파일 유형 수정자를 지정하여 임포트, 익스포트 및 로드 유틸리티에서 사용하지 않도록 할 수 있습니다. 그러나 구문 분석 오류를 피하려면 2바이트 분리문자 동작이 있어야 합니다. 익스포트에서 nodoubledel을 사용하면 문자 필드에 있을 때 문자 분리문자가 이중으로 처리되지 않습니다. 임포트 및 로드에서 nodoubledel을 사용하면 2바이트 분리문자가 문자 분리문자의 리터럴 인스턴스로 해석되지 않습니다.

nochardel
익스포트에서 nochardel 파일 유형 수정자를 사용하면 문자 필드가 문자 분리문자로 묶이지 않습니다. 임포트 및 로드에서 nochardel을 사용하면 문자 분리문자가 특수 문자로 처리되지 않으며 실제 데이터로 해석됩니다.

chardel
디폴트 분리문자 및 데이터를 혼동하지 않도록 기타 파일 유형 수정자를 사용할 수 있습니다. chardel 파일 유형 수정자는 단일 문자, x를 문자열 분리문자로 지정하여 큰 따옴표(디폴트) 대신 사용합니다.

coldel
마찬가지로 디폴트 쉼표를 컬럼 분리문자로 사용하지 않으려면 coldel을 사용하여 단일 문자, x를 열 데이터 분리문자로 지정합니다.

delprioritychar
DEL 파일 이동과 관련된 또 다른 문제로 분리문자의 올바른 선행 순서를 유지해야 합니다. 분리문자의 디폴트 우선순위는 행, 문자, 컬럼입니다. 그러나 일부 응용프로그램은 우선순위가 문자, 행, 컬럼이기도 합니다. 예를 들어 디폴트 우선순위를 사용하는 다음 DEL 데이터 파일이 있습니다.

```
"Vincent <row delimiter> is a manager",<row delimiter>
```

여기서 Vincent 및 is a manager와 같은 두 개 행을 포함한다고 해석됩니다. <row delimiter>가 문자 분리문자(")보다 우선되기 때문입니다. delprioritychar을 사용하면 문자 분리문자(")가 행 분리문자(<row delimiter>)보다 우선됩니다. 즉, 동일한 DEL 파일은 Vincent is a manager와 같은 하나의 행을 포함하는 것으로 해석됩니다.

컬럼 식별자 없는 ASCII(ASC) 파일 형식

컬럼 식별자 없는 ASCII 형식(임포트 및 로드 유틸리티에서 ASC라고도 함)은 고정 길이 및 가변 길이와 같은 두 가지 변형체로 사용됩니다. 고정 길이 ASC의 경우 모든 레코드는 고정 길이입니다. 가변 길이 ASC의 경우 레코드는 행 분리문자(항상 줄 바꾸기임)로 구분됩니다. 컬럼 식별자 없는 ASCII에서 *컬럼 식별자 없는*이라는 용어는 컬럼 값이 분리문자로 분리되지 않음을 나타냅니다.

ASC 데이터를 임포트 또는 로드하는 경우 reclen 파일 유형 수정자를 지정하면 데이터 파일이 고정 길이 ASC임을 나타냅니다. 이를 지정하지 않으면 데이터 파일이 유연한 길이 ASC임을 의미합니다.

컬럼 식별자 없는 ASCII 형식은 워드 프로세서를 포함하여 데이터에 대한 종횡배열(columnar) 형식을 사용하는 ASCII 제품에서 데이터를 교환할 때 사용할 수 있습니다. 각 ASC 파일은 행 및 컬럼으로 정렬된 데이터 값으로 구성된 ASCII 문자의 스트림입니다. 데이터 스트림의 행은 행 분리문자로 구분됩니다. 행 내 각 컬럼은 시작-종료 위치 쌍(IMPORT 매개변수에서 지정함)으로 정의됩니다. 각 쌍은 바이트 위치로 지정된 행 내 위치를 나타냅니다. 행 내 첫 번째 위치는 바이트 위치 1입니다. 각 위치 쌍의 첫 번째 요소는 컬럼이 시작되는 바이트이고 각 위치 쌍의 두 번째 요소는 컬럼이 종료되는 바이트입니다. 컬럼에서 오버랩할 수 있습니다. ASC 파일의 모든 행에서 컬럼 정의가 동일합니다.

ASC 파일은 다음으로 정의됩니다.

```
ASC file ::= Row 1 data || Row delimiter ||
             Row 2 data || Row delimiter ||
             .
             .
             .
             Row n data

Row i data ::= ASCII characters || Row delimiter

Row Delimiter ::= ASCII line feed sequence[a]
```

- [a] 레코드 구분 문자는 개행 문자, ASCII x0A로 가정합니다. Windows 운영 체제에서 생성된 데이터는 캐리지 리턴/줄 바꾸기 2바이트 표준 0x0D0A를 사용할 수 있습니다. EBCDIC 코드 페이지의 데이터는 EBCDIC LF 문자(0x25)를 레코드 구분 문자로 사용해야 합니다. LOAD 명령에서 codepage 파일 유형 수정자를 사용하여 EBCDIC 데이터를 로드할 수 있습니다. 레코드 구분 문자는 데이터 필드 부분으로 해석되지 않습니다.

ASC 데이터 유형 설명

표 51. ASC 파일 형식에 대해 승인할 수 있는 데이터 유형 양식

데이터 유형	임포트 유틸리티에 대해 승인할 수 있는 양식
BIGINT	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 상수가 승인됩니다. -9 223 372 036 854 775 808에서 9 223 372 036 854 775 807 사이의 범위를 벗어난 경우 개별 값이 거부됩니다. 10진수는 정수값으로 절단됩니다. 쉼표, 마침표 또는 콜론은 소수점으로 간주됩니다. 천단위 구분자는 허용되지 않습니다. 시작 및 종료 위치로 너비가 50바이트를 초과하지 않는 필드를 지정해야 합니다. 정수, 10진수 및 부동 소수점의 가수는 31자릿수를 초과할 수 없습니다. 부동 소수점의 지수는 3자릿수를 초과할 수 없습니다.
BLOB/CLOB	문자열. 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 절단됩니다. ASC *공백 절단* 옵션이 적용되면 원래 또는 절단된 문자열에서 뒤 공백이 스트라이핑됩니다.
BLOB_FILE, CLOB_FILE, DBCLOB_FILE(DBCS만 해당)	데이터를 보유하는 파일의 컬럼 식별자가 있는 또는 컬럼 식별자 없는 이름.
CHAR	문자열. 문자열은 필요한 경우 목표 컬럼의 너비와 일치시키기 위해 오른쪽에서 스페이스로 채워지거나 절단됩니다.
DATE	목표 데이터베이스의 지역 코드와 일관된 형식의 날짜 값을 나타내는 문자열. 시작 및 종료 위치로 날짜의 외부 표현 범위에 포함된 필드 너비를 지정해야 합니다.
DBCLOB(DBCS만 해당)	짝수 바이트 문자열. 홀수 바이트 문자열은 유효하지 않으며 승인되지 않습니다. 유효한 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 절단됩니다.

표 51. ASC 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	임포트 유틸리티에 대해 승인할 수 있는 양식
DECIMAL	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 상수가 승인됩니다. 임포트하는 데이터베이스 컬럼 범위를 벗어나면 개별 값이 거부됩니다. 입력 값의 소수점 뒤로 데이터베이스 컬럼의 스케일보다 더 많은 자릿수가 있으면 초과된 자릿수가 절단됩니다. 쉼표, 마침표 또는 콜론은 소수점으로 간주됩니다. 천단위 구분자는 허용되지 않습니다. 시작 및 종료 위치로 너비가 50바이트를 초과하지 않는 필드를 지정해야 합니다. 정수, 10진수 및 부동 소수점의 가수는 31자릿수를 초과할 수 없습니다. 부동 소수점의 지수는 3자릿수를 초과할 수 없습니다.
FLOAT(long)	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 상수가 승인됩니다. 모든 값이 유효합니다. 쉼표, 마침표 또는 콜론은 소수점으로 간주됩니다. FLOAT 상수의 지수 시작으로 대문자 또는 소문자 E가 승인됩니다. 시작 및 종료 위치로 너비가 50바이트를 초과하지 않는 필드를 지정해야 합니다. 정수, 10진수 및 부동 소수점의 가수는 31자릿수를 초과할 수 없습니다. 부동 소수점의 지수는 3자릿수를 초과할 수 없습니다.
GRAPHIC(DBCS만 해당)	짝수 바이트 문자열. 홀수 바이트 문자열은 유효하지 않으며 승인되지 않습니다. 유효한 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 2바이트 스페이스(0x8140)로 채워지거나 절단됩니다.
INTEGER	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 상수가 승인됩니다. -2 147 483 648에서 2 147 483 647 사이의 범위를 벗어난 경우 개별 값이 거부됩니다. 10진수는 정수값으로 절단됩니다. 쉼표, 마침표 또는 콜론은 소수점으로 간주됩니다. 천단위 구분자는 허용되지 않습니다. 시작 및 종료 위치로 너비가 50바이트를 초과하지 않는 필드를 지정해야 합니다. 정수, 10진수 및 부동 소수점의 가수는 31자릿수를 초과할 수 없습니다. 부동 소수점의 지수는 3자릿수를 초과할 수 없습니다.
LONG VARCHAR	문자열. 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 절단됩니다. ASC *공백 절단* 옵션이 적용되면 원래 또는 절단된 문자열에서 뒤 공백이 스트라이핑됩니다.
LONG VARGRAPHIC(DBCS만 해당)	짝수 바이트 문자열. 홀수 바이트 문자열은 유효하지 않으며 승인되지 않습니다. 유효한 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 절단됩니다.
SMALLINT	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 상수가 승인됩니다. -32 768에서 32 767 사이의 범위를 벗어난 경우 개별 값이 거부됩니다. 10진수는 정수값으로 절단됩니다. 쉼표, 마침표 또는 콜론은 소수점으로 간주됩니다. 천단위 구분자는 허용되지 않습니다. 시작 및 종료 위치로 너비가 50바이트를 초과하지 않는 필드를 지정해야 합니다. 정수, 10진수 및 부동 소수점의 가수는 31자릿수를 초과할 수 없습니다. 부동 소수점의 지수는 3자릿수를 초과할 수 없습니다.

표 51. ASC 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	임포트 유틸리티에 대해 승인할 수 있는 양식
TIME	목표 데이터베이스의 지역 코드와 일관된 형식의 시간 값을 나타내는 문자열. 시작 및 종료 위치로 시간의 외부 표현 범위에 포함된 필드 너비를 지정해야 합니다.
TIMESTAMP	데이터베이스의 스토리지에서 승인할 수 있는 시간소인 값을 나타내는 문자열. 시작 및 종료 위치로 시간소인의 외부 표현 범위에 포함된 필드 너비를 지정해야 합니다.
VARCHAR	문자열. 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 절단됩니다. ASC *공백 절단* 옵션이 적용되면 원래 또는 절단된 문자열에서 뒤 공백이 스트라이핑됩니다.
VARGRAPHIC(DBCS만 해당)	짝수 바이트 문자열. 홀수 바이트 문자열은 유효하지 않으며 승인되지 않습니다. 유효한 문자열은 필요한 경우 목표 컬럼의 최대 길이와 일치시키기 위해 오른쪽에서 절단됩니다.

ASC 파일 예

다음은 ASC 파일에 대한 예입니다. 각 라인은 줄 바꾸기 시퀀스로 종료됩니다. Windows 운영 체제의 경우 각 라인은 캐리지 리턴/줄 바꾸기 시퀀스로 종료됩니다.

```
Smith, Bob       4973     15.46
Jones, Suzanne   12345    16.34
Williams, Sam    452123  193.78
```

주:

1. ASC 파일은 컬럼 이름을 포함하지 않는다고 가정합니다.
2. 문자열은 분리문자로 묶지 *않습니다*. ASC 파일에서 컬럼의 데이터 유형은 데이터베이스 테이블에 있는 목표 컬럼의 데이터 유형으로 판별됩니다.
3. 널(NULL)은 다음 경우에 널(NULL) 입력 가능 데이터베이스 컬럼으로 임포트됩니다.
   - 숫자, DATE, TIME 또는 TIMESTAMP 데이터베이스 컬럼에 대해 공백인 필드가 목표가 됨
   - 시작 및 종료 위치 쌍이 없는 필드가 지정됨
   - 시작 및 종료 위치가 영(0)인 위치 쌍이 지정됨
   - 데이터 행이 너무 짧아서 목표 컬럼의 올바른 값을 포함할 수 없음
   - NULL INDICATORS 로드 옵션이 사용되고 널(NULL) 표시기 컬럼에 N 또는 사용자가 지정한 기타 값이 있음
4. 목표 컬럼이 널(NULL) 입력 가능하지 않고 숫자, DATE, TIME 또는 TIMESTAMP 컬럼으로 공백인 필드를 임포트하려고 하면 행이 거부됩니다.

5. 입력 데이터가 목표 컬럼과 호환 가능하지 않고 컬럼이 널(NULL) 입력 가능하면 오류가 발견되었는지에 따라 널(NULL)이 임포트되거나 행이 거부됩니다. 컬럼이 널(NULL) 입력 가능하지 않으면 행이 거부됩니다. 발견한 호환 불가능성을 지정하며 메시지 파일에 메시지가 작성됩니다.

IXF 파일 형식의 PC 버전

IXF의 PC 버전(PC/IXF) 파일 형식은 데이터베이스 관리 프로그램에서 채택한 IXF(Integration Exchange Format (IXF) 데이터 교환 아키텍처입니다. IXF 아키텍처는 특별히 관계형 데이터베이스 구조 및 데이터 교환을 위해 디자인되었습니다. PC/IXF 아키텍처를 사용하면 데이터베이스 관리 프로그램에서 요구사항 및 수신한 제품의 특이사항을 예상하지 않고도 데이터베이스를 익스포트할 수 있습니다. 마찬가지로 PC/IXF 파일을 임포트하는 제품은 PC/IXF 아키텍처만 이해하면 됩니다. 파일을 익스포트한 제품의 특징은 상관하지 않아도 됩니다. PC/IXF 파일 아키텍처는 익스포트 및 임포트하는 데이터베이스 시스템 모두의 독립성을 유지합니다.

IXF 아키텍처는 특정 관계형 데이터베이스 제품에서 지원되지 않을 수도 있는 일부 유형을 포함하여 다양한 관계형 데이터 유형을 지원하는 일반적인 관계형 데이터베이스 교환 형식입니다. PC/IXF 파일 형식은 이러한 유연성을 보존합니다. 예를 들어 PC/IXF 아키텍처는 1바이트 문자 문자열(SBCS) 및 2바이트 문자 문자열(DBCS) 데이터 유형을 모두 지원합니다. 모든 구현이 모든 PC/IXF 데이터 유형을 지원하지는 않습니다. 그러나 임포트 중 지원되지 않는 데이터 유형의 발견 및 배치에 대한 제한된 구현이 제공됩니다.

일반적으로 PC/IXF 파일은 가변 길이 레코드의 연속된 시퀀스로 구성됩니다. 파일은 다음에 표시된 순서로 다음 레코드 유형을 포함합니다.

- 레코드 유형 H의 한 헤더 레코드
- 레코드 유형 T의 한 테이블 레코드
- 레코드 유형 C의 여러 컬럼 디스크립터 레코드(테이블의 각 컬럼에 레코드 하나)
- 레코드 유형 D의 여러 데이터 레코드(테이블의 각 행이 하나 이상의 D 레코드로 표시됨)

PC/IXF 파일은 레코드 유형 A의 응용프로그램 레코드(H 유형 이후 임의의 위치에 배치됨)를 포함할 수 있습니다. 이러한 레코드는 응용프로그램이 PC/IXF 파일에서 PC/IXF 형식으로 정의되지 않은 추가 데이터를 포함할 수 있도록 PC/IXF 파일에 허용됩니다. A 레코드에서 응용프로그램 ID가 함축하는 데이터 형식 및 컨텐츠에 대한 특정 정보를 포함하지 않는 PC/IXF 파일을 읽는 프로그램은 A 레코드를 무시합니다.

PC/IXF 파일의 모든 레코드는 레코드 길이 표시기로 시작합니다. 이는 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 오른쪽 맞춤 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. PC/IXF 파

일을 읽는 프로그램은 현재 레코드 끝과 다음 레코드 시작을 찾으려면 이 레코드 길이를 사용해야 합니다. 정의된 모든 필드를 포함하도록 H, T 및 C 레코드가 충분히 커야 하며 해당 레코드 길이 필드는 실제 길이와 일치해야 합니다. 그러나 추가 데이터(예: *새* 필드)가 이러한 레코드 중 하나 끝에 추가되면 PC/IXF 파일을 읽는 기존 프로그램은 추가 데이터를 무시하고 경고 메시지만 생성합니다. 그러나 PC/IXF 파일을 쓰는 프로그램은 정의된 모든 필드를 포함하는 데 필요한 정확한 길이의 H, T 및 C 레코드를 작성해야 합니다.

PC/IXF 파일에 LLS(LOB Location Specifier) 컬럼이 있으면 각 LLS 컬럼에는 고유한 D 레코드가 있어야 합니다. D 레코드는 익스포트 유틸리티에서 자동으로 작성되지만 써드 파티 도구를 사용하여 PC/IXF 파일을 생성하는 경우 수동으로 작성해야 합니다. 또한 널(NULL) 값인 항목을 포함하여 테이블에 각 LOB 컬럼에 대한 LLS가 있어야 합니다. LOB 컬럼이 널(NULL)이면 널(NULL)인 LOB를 나타내는 LLS를 작성해야 합니다.

각 XML 컬럼의 D 레코드 항목에 XDS 자체 앞에 XDS(XML Data Specifier) 길이를 표시하는 2바이트 리틀 엔디안이 있습니다.

예를 들어 다음 XDS를 참조하십시오.

```
XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

여기서 D 레코드의 다음 바이트로 표시됩니다.

```
0x3D 0x00 XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

PC/IXF 파일 레코드는 문자 데이터를 포함하는 필드로 구성됩니다. 임포트 및 익스포트 유틸리티는 다음 두 가지 경우를 제외하고 목표 데이터베이스의 CPGID를 사용하여 이 문자 데이터를 해석합니다.

- A 레코드의 IXFADATA 필드.

  IXFADATA 필드에 포함된 문자 데이터의 코드 페이지 환경은 특별 A 레코드를 작성 및 처리하는 응용프로그램에 의해 설정됩니다. 즉, 구현에 따라 환경이 달라집니다.

- D 레코드의 IXFDCOLS 필드.

  IXFDCOLS 필드에 포함된 문자 데이터의 코드 페이지 환경은 특정 컬럼 및 해당 데이터를 정의하는 C 레코드에 포함된 정보의 기능입니다.

H, T 및 C 레코드의 숫자 필드와 D 및 A 레코드의 접두부 영역은 선행 영(0)이나 공백을 사용하여 정수값을 나타내는 1바이트 문자 표를 오른쪽부터 채워야 합니다. 값이 영(0)이면 오른쪽으로 맞춰진 하나 이상의 영(0) 문자(공백이 아님)를 표시합니다. 이러한 숫자 필드 중 하나가 사용되지 않으면(예: 데이터 유형으로 길이가 내재적으로 결정되는 IXFCLENG) 공백으로 채워야 합니다. 숫자 필드는 다음과 같습니다.

```
IXFHRECL, IXFTRECL, IXFCRECL, IXFDRECL, IXFARECL,
IXFHHCNT, IXFHSBCP, IXFHDBCP, IXFTCCNT, IXFTNAML,
IXFCLENG, IXFCDRID, IXFCPOSN, IXFCNAML, IXFCTYPE,
IXFCSBCP, IXFCDBCP, IXFCNDIM, IXFCDSIZ, IXFDRID
```

주: 데이터베이스 관리 프로그램 PC/IXF 파일 형식은 System/370™과 동일하지 않습니다.

PC/IXF 레코드 유형

다음과 같은 다섯 가지 기본 PC/IXF 레코드 유형이 있습니다.

- 헤더
- 테이블
- 컬럼 디스크립터
- 데이터
- 응용프로그램

다음은 DB2에서 사용하는 여섯 가지 응용프로그램 부속 유형입니다.

- 인덱스
- 계층 구조
- 서브테이블
- 연속
- 종료
- ID

각 PC/IXF 레코드는 일련의 필드로 정의됩니다. 이 필드는 필수 필드이며 표시된 순서대로 나타나야 합니다.

HEADER RECORD

FIELD NAME	LENGTH	TYPE	COMMENTS
IXFHRECL	06-BYTE	CHARACTER	record length
IXFHRECT	01-BYTE	CHARACTER	record type = 'H'
IXFHID	03-BYTE	CHARACTER	IXF identifier
IXFHVERS	04-BYTE	CHARACTER	IXF version
IXFHPROD	12-BYTE	CHARACTER	product
IXFHDATE	08-BYTE	CHARACTER	date written
IXFHTIME	06-BYTE	CHARACTER	time written
IXFHHCNT	05-BYTE	CHARACTER	heading record count
IXFHSBCP	05-BYTE	CHARACTER	single byte code page
IXFHDBCP	05-BYTE	CHARACTER	double byte code page
IXFHFIL1	02-BYTE	CHARACTER	reserved

다음은 헤더 레코드에 포함된 필드입니다.

IXFHRECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. H 레코드는 정의된 모든 필드를 포함하도록 충분히 길어야 합니다.

IXFHRECT

IXF 레코드 유형으로, 이 레코드에 대해 H로 설정됩니다.

IXFHID

파일 형식 ID로, 이 파일에 대해 IXF로 설정됩니다.

IXFHVERS

파일 작성 시 사용되는 PC/IXF 형식 레벨로, '0002'로 설정됩니다.

IXFHPROD

파일을 작성하는 프로그램에서 자체 프로그램을 식별하는 데 사용할 수 있는 필드입니다. 이 필드가 채워지면 처음 6바이트로 파일을 작성하는 제품을 식별하고 마지막 6바이트로 작성한 제품의 버전 또는 릴리스를 표시합니다. 데이터베이스 관리 프로그램에서는 이 필드를 사용하여 데이터베이스 관리 프로그램 특정 데이터 존재를 알립니다.

IXFHDATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다.

IXFHTIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 선택적 필드이며 공백으로 둘 수 있습니다.

IXFHHCNT

이 파일에 있는 H, T 및 C 레코드 수로, 첫 번째 데이터 레코드 앞에 나옵니다. 이 계수에 A 레코드는 포함되지 않습니다.

IXFHSBCP

1바이트 코드 페이지 필드로, SBCS CPGID 또는 '00000'으로 표시되는 1바이트 문자를 포함합니다.

익스포트 유틸리티는 이 필드를 익스포트된 데이터베이스 테이블의 SBCS CPGID와 동일하게 설정합니다. 예를 들어 테이블 SBCS CPGID가 850이면 이 필드는 '00850'을 포함합니다.

IXFHDBCP

2바이트 코드 페이지 필드로, DBCS CPGID 또는 '00000'으로 표시되는 1바이트 문자를 포함합니다.

익스포트 유틸리티는 이 필드를 익스포트된 데이터베이스 테이블의 DBCS CPGID와 동일하게 설정합니다. 예를 들어 테이블 DBCS CPGID가 301이면 이 필드는 '00301'을 포함합니다.

IXFHFIL1

호스트 IXF 파일의 예약 필드와 일치하도록 2개 공백으로 설정된 필드를 예비로 할당합니다.

```
TABLE RECORD

   FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     -------    ---------   -------------

   IXFTRECL       006-BYTE   CHARACTER   record length
   IXFTRECT       001-BYTE   CHARACTER   record type = 'T'
   IXFTNAML       003-BYTE   CHARACTER   name length
   IXFTNAME       256-BYTE   CHARACTER   name of data
   IXFTQULL       003-BYTE   CHARACTER   qualifier length
   IXFTQUAL       256-BYTE   CHARACTER   qualifier
   IXFTSRC        012-BYTE   CHARACTER   data source
   IXFTDATA       001-BYTE   CHARACTER   data convention = 'C'
   IXFTFORM       001-BYTE   CHARACTER   data format = 'M'
   IXFTMFRM       005-BYTE   CHARACTER   machine format = 'PC'
   IXFTLOC        001-BYTE   CHARACTER   data location = 'I'
   IXFTCCNT       005-BYTE   CHARACTER   'C' record count
   IXFTFIL1       002-BYTE   CHARACTER   reserved
   IXFTDESC       030-BYTE   CHARACTER   data description
   IXFTPKNM       257-BYTE   CHARACTER   primary key name
   IXFTDSPC       257-BYTE   CHARACTER   reserved
   IXFTISPC       257-BYTE   CHARACTER   reserved
   IXFTLSPC       257-BYTE   CHARACTER   reserved
```

다음은 테이블 레코드에 포함된 필드입니다.

IXFTRECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. T 레코드는 정의된 모든 필드를 포함하도록 충분히 길어야 합니다.

IXFTRECT

IXF 레코드 유형으로, 이 레코드에 대해 T로 설정됩니다.

IXFTNAML

IXFTNAME 필드에서 테이블 이름의 길이(바이트)입니다.

IXFTNAME

테이블 이름입니다. 각 파일에 하나의 테이블만 있으면 오직 정보용 필드입니다. 데이터를 임포트할 때 데이터베이스 관리 프로그램에서는 이 필드를 사용하지 않습니다. PC/IXF 파일을 작성할 때 데이터베이스 관리 프로그램에서는 이 필드에 DOS 파일 이름 및 가능한 경로 정보를 작성합니다.

IXFTQULL

IXFTQUAL 필드에서 테이블 이름 규정자의 길이(바이트)입니다.

IXFTQUAL

테이블 이름 규정자로, 관계형 시스템에서 테이블 작성자를 식별합니다. 오직 정보용 필드입니다. 파일을 작성하는 프로그램에서 이 필드에 작성할 데이터를 포함하지 않는 경우 기본 채우기 값은 공백입니다. 파일을 읽는 프로그램은 이 필드를 인쇄 또는 표시하거나 정보용 필드에 저장할 수 있지만 계산은 이 필드의 컨텐츠에 종속되지 않습니다.

IXFTSRC

데이터의 원래 소스를 표시하는 데 사용됩니다. 오직 정보용 필드입니다. 파일을 작성하는 프로그램에서 이 필드에 작성할 데이터를 포함하지 않는 경우 기본 채우기 값은 공백입니다. 파일을 읽는 프로그램은 이 필드를 인쇄 또는 표시하거나 정보용 필드에 저장할 수 있지만 계산은 이 필드의 컨텐츠에 종속되지 않습니다.

IXFTDATA

데이터를 설명하는 데 사용되는 규칙입니다. 이 필드는 임포트 및 익스포트의 경우 C로 설정됩니다. 이는 개별 컬럼 속성이 다음 컬럼 디스크립터(C) 레코드에서 설명되며 해당 데이터가 PC/IXF 규칙을 따름을 의미합니다.

IXFTFORM

숫자 데이터를 저장하는 데 사용되는 규칙입니다. 이 필드는 M으로 설정됩니다. 이는 데이터의 숫자 데이터(D)가 IXFTMFRM 필드에서 지정한 머신(내부) 형식으로 저장됨을 의미합니다.

IXFTMFRM

PC/IXF 파일에서 머신 데이터의 형식입니다. 데이터베이스 관리 프로그램에서는 이 필드가 PC*bbb*로 설정된 경우 파일 읽기 또는 쓰기만 수행합니다. 여기서 *b*는 공백을 나타내고 PC에서는 PC/IXF 파일의 데이터가 IBM PC 머신 형식이 되도록 지정합니다.

IXFTLOC

데이터의 위치입니다. 데이터베이스 관리 프로그램에서는 1의 값을 지원합니다. 이는 데이터가 이 파일 내부에 있음을 의미합니다.

IXFTCCNT

이 테이블에서 C 레코드의 수입니다. 정수값을 나타내는 오른쪽 맞춤 문자입니다.

IXFTFIL1

호스트 IXF 파일의 예약 필드와 일치하도록 2개 공백으로 설정된 필드를 예비로 할당합니다.

IXFTDESC

테이블에 대한 설명 데이터입니다. 오직 정보용 필드입니다. 파일을 작성하는 프로그램에서 이 필드에 작성할 데이터를 포함하지 않는 경우 기본 채우기 값은 공백입니다. 파일을 읽는 프로그램은 이 필드를 인쇄 또는 표시하거나 정보용 필드에 저장할 수 있지만 계산은 이 필드의 컨텐츠에 종속되지 않습니다. 컬럼이 디폴트로 널(NULL)이 아니고 워크스테이션 데이터베이스에서 테이블 이름을 가져온 경우 이 필드는 NOT NULL WITH DEFAULT를 포함합니다.

IXFTPKNM

테이블에 정의된 1차 키(있는 경우)의 이름입니다. 이름은 널(null)로 끝나는 문자열로 저장됩니다.

IXFTDSPC

이 필드는 나중에 사용하도록 예약됩니다.

IXFTISPC

이 필드는 나중에 사용하도록 예약됩니다.

IXFTLSPC

이 필드는 나중에 사용하도록 예약됩니다.

COLUMN DESCRIPTOR RECORD

FIELD NAME	LENGTH	TYPE	COMMENTS
IXFCRECL	006-BYTE	CHARACTER	record length
IXFCRECT	001-BYTE	CHARACTER	record type = 'C'
IXFCNAML	003-BYTE	CHARACTER	column name length
IXFCNAME	256-BYTE	CHARACTER	column name
IXFCNULL	001-BYTE	CHARACTER	column allows nulls
IXFCDEF	001-BYTE	CHARACTER	column has defaults
IXFCSLCT	001-BYTE	CHARACTER	column selected flag
IXFCKPOS	002-BYTE	CHARACTER	position in primary key
IXFCCLAS	001-BYTE	CHARACTER	data class
IXFCTYPE	003-BYTE	CHARACTER	data type
IXFCSBCP	005-BYTE	CHARACTER	single byte code page
IXFCDBCP	005-BYTE	CHARACTER	double byte code page
IXFCLENG	005-BYTE	CHARACTER	column data length
IXFCDRID	003-BYTE	CHARACTER	'D' record identifier
IXFCPOSN	006-BYTE	CHARACTER	column position
IXFCDESC	030-BYTE	CHARACTER	column description
IXFCLOBL	020-BYTE	CHARACTER	lob column length
IXFCUDTL	003-BYTE	CHARACTER	UDT name length
IXFCUDTN	256-BYTE	CHARACTER	UDT name
IXFCDEFL	003-BYTE	CHARACTER	default value length
IXFCDEFV	254-BYTE	CHARACTER	default value
IXFCREF	001-BYTE	CHARACTER	reference type
IXFCNDIM	002-BYTE	CHARACTER	number of dimensions
IXFCDSIZ	varying	CHARACTER	size of each dimension

다음은 컬럼 디스크립터 레코드에 포함된 필드입니다.

IXFCRECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. C 레코드는 정의된 모든 필드를 포함하도록 충분히 길어야 합니다.

IXFCRECT

IXF 레코드 유형으로, 이 레코드에 대해 C로 설정됩니다.

IXFCNAML

IXFCNAME 필드에서 컬럼 이름의 길이(바이트)입니다.

IXFCNAME

컬럼 이름입니다.

IXFCNULL

이 컬럼에서 NULL 값이 허용되는지 지정합니다. 유효한 설정은 Y 또는 N입니다.

IXFCDEF

이 필드에 디폴트값이 정의되는지를 지정합니다. 유효한 설정은 Y 또는 N입니다.

IXFCSLCT

사용하지 않는 필드로, 이전에 데이터에서 컬럼 서브세트 선택을 허용하는 데 사용되었습니다. PC/IXF 파일을 작성하는 프로그램은 항상 이 필드에 Y를 저장합니다. PC/IXF 파일을 읽는 프로그램은 필드를 무시합니다.

IXFCKPOS

1차 키에 포함된 컬럼의 위치입니다. 컬럼이 1차 키에 포함되지 않은 경우 올바른 값의 범위는 01 - 16 또는 N입니다.

IXFCCLAS

IXFCTYPE 필드에서 사용할 데이터 유형의 클래스입니다. 데이터베이스 관리 프로그램에서는 관계형 유형(R)만 지원합니다.

IXFCTYPE

컬럼의 데이터 유형입니다.

IXFCSBCP

SBCS CPGID로 표시되는 1바이트 문자를 포함합니다. 이 필드에서는 이 컬럼에 대한 D 레코드의 IXFDCOLS 필드에 나타나는 1바이트 문자 데이터의 CPGID를 지정합니다.

이 필드의 시맨틱은 IXFCTYPE 필드에 지정된 컬럼의 데이터 유형에 따라 달라집니다.

- 문자열 컬럼의 경우 이 필드는 정상적으로 H 레코드에서 IXFHSBCP 필드 값과 동일한 0이 아닌 값을 포함해야 합니다. 그러나 다른 값도 허용됩니다. 이 값이 영(0)이면 컬럼은 비트 문자열 데이터를 포함한다고 해석됩니다.
- 숫자 컬럼의 경우 이 필드는 의미를 지니지 않습니다. 익스포트 유틸리티에서는 0으로 설정되며 임포트 유틸리티에서는 무시됩니다.
- 날짜 또는 시간 컬럼의 경우 이 필드는 의미를 지니지 않습니다. 익스포트 유틸리티에서는 IXFHSBCP 필드 값으로 설정되며 임포트 유틸리티에서는 무시됩니다.
- 그래픽 컬럼의 경우 이 필드는 영(0)이어야 합니다.

IXFCDBCP

DBCS CPGID로 표시되는 1바이트 문자를 포함합니다. 이 필드에서는 이 컬럼에 대한 D 레코드의 IXFDCOLS 필드에 나타나는 2바이트 문자 데이터의 CPGID를 지정합니다.

이 필드의 시맨틱은 IXFCTYPE 필드에 지정된 컬럼의 데이터 유형에 따라 달라집니다.

- 문자열 컬럼의 경우 이 필드는 H 레코드에서 IXFHDBCP 필드 값과 동일한 값 또는 영(0)이어야 합니다. 그러나 다른 값도 허용됩니다. IXFCSBCP 필드 값이 영(0)이면 이 필드 값도 영(0)이어야 합니다.
- 숫자 컬럼의 경우 이 필드는 의미를 지니지 않습니다. 익스포트 유틸리티에서는 0으로 설정되며 임포트 유틸리티에서는 무시됩니다.
- 날짜 또는 시간 컬럼의 경우 이 필드는 의미를 지니지 않습니다. 익스포트 유틸리티에서는 0으로 설정되며 임포트 유틸리티에서는 무시됩니다.
- 그래픽 컬럼의 경우 이 필드는 IXFHDBCP 필드 값과 같아야 합니다.

IXFCLENG

설명할 컬럼 *크기*에 대한 정보를 제공합니다. 일부 데이터 유형에서 이 필드는 사용되지 않으며 공백으로 두어야 합니다. 기타 데이터 유형의 경우 이 필드는 컬럼 길이를 지정하는 정수를 나타내는 오른쪽 맞춤 문자를 포함합니다. 기타 데이터 유형의 경우 이 필드는 두 개의 서브필드(정밀도를 나타내는 3바이트 및 스케일을 나타내는 2바이트)로 구분됩니다. 이 두 서브필드에서는 정수를 나타내는 오른쪽 맞춤 문자를 포함합니다. 버전 9.7부터, 시간소인 데이터 유형의 경우 이 필드는 시간소인 정밀도를 지정하는 정수를 나타내는 오른쪽 맞춤 문자를 포함합니다.

IXFCDRID

D 레코드 ID입니다. 이 필드는 정수값을 나타내는 오른쪽 맞춤 문자를 포함합니다. 여러 D 레코드를 사용하여 PC/IXF 파일에서 각 데이터 행을 포함할 수 있습니다. 이 필드에서는 컬럼의 데이터를 포함하는 D 레코드(행 데이터에 기여하는 여러 D 레코드 중)를 지정합니다. 값이 1이면(예: 001) 컬럼의 데이터

가 데이터 행의 첫 번째 D 레코드에 있음을 의미합니다. 첫 번째 C 레코드에서 IXFCDRID 값은 1입니다. 모든 후속 C 레코드에서 IXFCDRID 값은 이전 C 레코드 값과 같거나 1이 더 커야 합니다.

IXFCPOSN

이 필드 값은 테이블 데이터 행을 나타내는 D 레코드 중 하나에서 컬럼 데이터를 찾는 데 사용됩니다. D 레코드의 IXFDCOLS 필드에서 이 컬럼 데이터의 시작 위치에 해당합니다. 널(NULL) 입력 가능한 컬럼인 경우 IXFCPOSN은 널(NULL) 표시기를 가리킵니다. 그렇지 않으면 데이터 자체를 가리킵니다. 컬럼에서 가변 길이 데이터를 포함하는 경우 데이터 자체는 현재 길이 표시기로 시작됩니다. D 레코드의 IXFDCOLS 필드에서 첫 번째 바이트에 대한 IXFCPOSN 값은 1(영(0)이 아님)입니다. 컬럼이 새 D 레코드에 있는 경우 IXFCPOSN 값은 1입니다. 그렇지 않으면 IXFCPOSN 값은 데이터 값을 오버랩하지 않도록 컬럼에서 적당한 등급만큼 계속 증가해야 합니다.

IXFCDESC

컬럼에 대한 설명 정보입니다. 오직 정보용 필드입니다. 파일에 작성하는 프로그램에서 이 필드에 작성할 데이터를 포함하지 않는 경우 기본 채우기 값은 공백입니다. 파일을 읽는 프로그램은 이 필드를 인쇄 또는 표시하거나 정보용 필드에 저장할 수 있지만 계산은 이 필드의 컨텐츠에 종속되지 않습니다.

IXFCLOBL

이 컬럼에 정의된 long 또는 LOB의 길이(바이트)입니다. 이 컬럼이 long 또는 LOB가 아니면 이 필드 값은 000입니다.

IXFCUDTL

IXFCUDTN 필드에서 사용자 정의 유형 이름의 길이(바이트)입니다. 이 컬럼의 유형이 UDT가 아니면 이 필드 값은 000입니다.

IXFCUDTN

이 컬럼의 데이터 유형으로 사용되는 사용자 정의 유형의 이름입니다.

IXFCDEFL

IXFCDEFV 필드에서 디폴트값의 길이(바이트)입니다. 이 컬럼에 디폴트값이 없으면 이 필드 값은 000입니다.

IXFCDEFV

정의된 경우 이 컬럼의 디폴트값을 지정합니다.

IXFCREF

컬럼이 계층 구조에 포함된 경우 이 필드에서는 컬럼 유형(데이터 컬럼(D) 또는 참조 컬럼(R))을 지정합니다.

IXFCNDIM

컬럼의 차원 수입니다. 이 버전의 PC/IXF에서 배열은 지원되지 않습니다. 따라서 이 필드는 영(0)의 정수값을 표시하는 문자를 포함해야 합니다.

IXFCDSIZ

각 차원의 *크기* 또는 범위입니다. 이 필드의 길이는 차원당 5바이트입니다. 배열은 지원되지 않으므로(즉, 차원 수는 0이어야 함) 이 필드 길이는 영(0)이며 이 필드는 실제로 존재하지 않습니다.

```
DATA RECORD

   FIELD NAME     LENGTH    TYPE        COMMENTS
   ----------     -------   ---------   -------------
   IXFDRECL       06-BYTE   CHARACTER   record length
   IXFDRECT       01-BYTE   CHARACTER   record type = 'D'
   IXFDRID        03-BYTE   CHARACTER   'D' record identifier
   IXFDFIL1       04-BYTE   CHARACTER   reserved
   IXFDCOLS       varying   variable    columnar data
```

다음은 데이터 레코드에 포함된 필드입니다.

IXFDRECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (*총 레코드 크기* - 6바이트)입니다. 각 D 레코드는 레코드에 저장된 마지막 데이터 컬럼의 현재 어커런스에 대해 모든 유효 데이터를 포함하도록 충분히 길어야 합니다.

IXFDRECT

IXF 레코드 유형으로, 이 레코드에 대해 D로 설정됩니다. 테이블에서 데이터 값을 포함함을 의미합니다.

IXFDRID

레코드 ID로, 데이터 행에 기여하는 여러 D 레코드의 시퀀스에서 특정 D 레코드를 식별합니다. 데이터 행의 첫 번째 D 레코드인 경우 이 필드 값은 1, 데이터 행의 두 번째 D 레코드인 경우 이 필드 값은 2와 같은 방식으로 지정됩니다. 각 데이터 행에 대해 C 레코드에서 호출된 모든 D 레코드 ID는 실제로 존재해야 합니다.

IXFDFIL1

호스트 IXF 파일의 예약 필드와 일치하고 가능한 시프트 아웃(Shift-Out) 문자의 위치를 보유하도록 4개 공백으로 설정된 필드를 예비로 할당합니다.

IXFDCOLS

종횡배열(columnar) 데이터의 영역입니다. 데이터 레코드(D 레코드)의 데이터 영역은 하나 이상의 컬럼 항목으로 구성됩니다. 각 컬럼 디스크립터 레코드에 대한 하나의 컬럼 항목으로, D 레코드와 동일한 D 레코드 ID를 보유합니다. D 레코드에서 컬럼 항목의 시작 위치는 C 레코드의 IXFCPOSN 값으로 표시됩니다.

컬럼 항목 데이터의 형식은 컬럼이 널(NULL) 입력 가능한지에 따라 달라집니다.

- 널(NULL) 입력 가능 컬럼인 경우(IXFCNULL 필드가 Y로 설정됨) 컬럼 항목 데이터는 널(NULL) 표시기를 포함합니다. 널(NULL) 입력 가능 컬럼이 아닌 경우 표시기는 실제 데이터베이스 값을 포함하는 데이터 유형별 정보 앞에 표시됩니다. 널(NULL) 표시기는 널(NULL)이 아닌 경우 x'0000', 널(NULL)인 경우 x'FFFF'로 설정되는 2바이트 값입니다.
- 널(NULL) 입력 가능 컬럼이 아닌 경우 컬럼 항목 데이터는 실제 데이터베이스 값을 포함하는 데이터 유형별 정보만 포함합니다.

가변 길이 데이터 유형인 경우 데이터 유형별 정보는 현재 길이 표시기를 포함합니다. 현재 길이 표시기는 IXFTMFRM 필드에 지정된 양식으로 표시되는 2바이트 정수입니다.

D 레코드 데이터 영역의 길이는 32 771바이트를 초과할 수 없습니다.

```
APPLICATION RECORD

   FIELD NAME     LENGTH    TYPE        COMMENTS
   ----------     -------   ---------   -------------
   IXFARECL       06-BYTE   CHARACTER   record length
   IXFARECT       01-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       12-BYTE   CHARACTER   application identifier
   IXFADATA       varying   variable    application-specific data
```

다음은 응용프로그램 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 A 레코드를 작성할 때 식별하는 응용프로그램 ID입니다. 데이터베이스 관리 프로그램에서 작성한 PC/IXF 파일의 경우 이 필드의 처음 6자는 데이터베이스 관리 프로그램을 식별하는 상수로 설정되고 마지막 6자는 데이터베이스 관리 프로그램의 릴리스 또는 버전이나 A 레코드를 작성한 다른 응용프로그램을 식별하는 A 레코드를 포함할 수 있습니다.

IXFADATA

이 필드는 응용프로그램 종속 보완 데이터를 포함합니다. 이 데이터의 양식 및 컨텐츠는 A 레코드를 작성한 프로그램 및 A 레코드를 처리할 수 있는 다른 응용프로그램에만 알려집니다.

```
DB2 INDEX RECORD

   FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
   IXFARECL       006-BYTE   CHARACTER   record length
   IXFARECT       001-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       012-BYTE   CHARACTER   application identifier =
                                            'DB2    02.00'
   IXFAITYP       001-BYTE   CHARACTER   application specific data type =
                                            'I'
   IXFADATE       008-BYTE   CHARACTER   date written from the 'H' record
   IXFATIME       006-BYTE   CHARACTER   time written from the 'H' record
   IXFANDXL       002-BYTE   SHORT INT   length of name of the index
   IXFANDXN       256-BYTE   CHARACTER   name of the index
   IXFANCL        002-BYTE   SHORT INT   length of name of the index creator
   IXFANCN        256-BYTE   CHARACTER   name of the index creator
   IXFATABL       002-BYTE   SHORT INT   length of name of the table
   IXFATABN       256-BYTE   CHARACTER   name of the table
   IXFATCL        002-BYTE   SHORT INT   length of name of the table creator
   IXFATCN        256-BYTE   CHARACTER   name of the table creator
   IXFAUNIQ       001-BYTE   CHARACTER   unique rule
   IXFACCNT       002-BYTE   CHARACTER   column count
   IXFAREVS       001-BYTE   CHARACTER   allow reverse scan flag
   IXFAPCTF       002-BYTE   CHARACTER   amount of pct free
   IXFAPCTU       002-BYTE   CHARACTER   amount of minpctused
   IXFAEXTI       001-BYTE   CHARACTER   reserved
   IXFACNML       002-BYTE   SHORT INT   length of name of the columns
   IXFACOLN       varying    CHARACTER   name of the columns in the index
```

각 사용자 정의 인덱스에 대해 이 유형의 레코드 하나가 지정됩니다. 이 레코드는 테이블의 모든 C 레코드 다음에 위치합니다. 다음은 DB2 인덱스 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (*총 레코드 크기* - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 이 A 레코드를 작성할 때 DB2를 식별하는 응용프로그램 ID입니다.

IXFAITYP

DB2 응용프로그램 레코드의 부속 유형 ″I″가 되도록 지정합니다.

IXFADATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다. 이 필드는 IXFHDATE 값과 동일해야 합니다.

IXFATIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 IXFHTIME 값과 동일해야 합니다.

IXFANDXL

IXFANDXN 필드에서 인덱스 이름의 길이(바이트)입니다.

IXFANDXN

인덱스의 이름입니다.

IXFANCL

IXFANCN 필드에서 인덱스 작성자 이름의 길이(바이트)입니다.

IXFANCN

인덱스 작성자의 이름입니다.

IXFATABL

IXFATABN 필드에서 테이블 이름의 길이(바이트)입니다.

IXFATABN

테이블 이름입니다.

IXFATCL

IXFATCN 필드에서 테이블 작성자 이름의 길이(바이트)입니다.

IXFATCN

테이블 작성자의 이름입니다.

IXFAUNIQ

인덱스 유형을 지정합니다. 올바른 값은 1차 키의 경우 P, 고유 인덱스의 경우 U, 비고유 인덱스의 경우 D입니다.

IXFACCNT

인덱스 정의에 있는 컬럼 수를 지정합니다.

IXFAREVS

이 인덱스에서 역방향 스캔이 허용되는지를 지정합니다. 올바른 값은 역방향 스캔의 경우 Y이고 역방향 스캔이 아닌 경우 N입니다.

IXFAPCTF

여유 공간으로 할당할 인덱스 페이지의 백분율을 지정합니다. 올바른 값의 범위는 -1 - 99입니다. -1 또는 영(0)의 값이 지정되면 시스템 디폴트값이 사용됩니다.

IXFAPCTU

두 인덱스 페이지를 병합하기 전에 여유 공간으로 할당해야 하는 인덱스 페이지의 최소 백분율을 지정합니다. 올바른 값의 범위는 00 - 99입니다.

IXFAEXTI

나중에 사용하도록 예약됩니다.

IXFACNML

IXFACOLN 필드에서 컬럼 이름의 길이(바이트)입니다.

IXFACOLN

이 인덱스에 포함된 컬럼의 이름입니다. 올바른 값의 양식은 *+name-name...* 입니다. 여기서 +는 컬럼에서 오름차순을 지정하고 -는 컬럼에서 내림차순을 지정합니다.

```
DB2 HIERARCHY RECORD

   FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
   IXFARECL       006-BYTE   CHARACTER   record length
   IXFARECT       001-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       012-BYTE   CHARACTER   application identifier =
                                            'DB2    02.00'
   IXFAXTYP       001-BYTE   CHARACTER   application specific data type =
                                            'X'
   IXFADATE       008-BYTE   CHARACTER   date written from the 'H' record
   IXFATIME       006-BYTE   CHARACTER   time written from the 'H' record
   IXFAYCNT       010-BYTE   CHARACTER   'Y' record count for this hierarchy
   IXFAYSTR       010-BYTE   CHARACTER   starting column of this hierarchy
```

이 유형의 레코드 하나를 사용하여 계층 구조를 설명합니다. 모든 서브테이블 레코드(아래 참조)는 계층 구조 레코드 바로 다음에 있어야 하며 계층 구조 레코드는 테이블의 모든 C 레코드 다음에 배치됩니다. 다음은 DB2 계층 구조 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (*총 레코드 크기* - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 이 A 레코드를 작성할 때 DB2를 식별하는 응용프로그램 ID입니다.

IXFAXTYP

DB2 응용프로그램 레코드의 부속 유형 ″X″가 되도록 지정합니다.

IXFADATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다. 이 필드는 IXFHDATE 값과 동일해야 합니다.

IXFATIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 IXFHTIME 값과 동일해야 합니다.

IXFAYCNT

이 계층 구조 레코드 이후에 나오는 예상 서브테이블 레코드 수를 지정합니다.

IXFAYSTR

익스포트된 데이터의 시작 지점에서 서브테이블 레코드의 인덱스를 지정합니다. 계층 구조 익스포트가 루트가 아닌 서브테이블에서 시작되면 이 서브테이블의 모든 상위 테이블이 익스포트됩니다. IXF 파일 내 이 서브테이블의 위치는 이 필드에도 저장됩니다. 첫 번째 X 레코드는 인덱스가 영(0)인 컬럼을 의미합니다.

```
DB2 SUBTABLE RECORD

   FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
   IXFARECL       006-BYTE   CHARACTER   record length
   IXFARECT       001-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       012-BYTE   CHARACTER   application identifier =
                                            'DB2    02.00'
   IXFAYTYP       001-BYTE   CHARACTER   application specific data type =
                                            'Y'
   IXFADATE       008-BYTE   CHARACTER   date written from the 'H' record
   IXFATIME       006-BYTE   CHARACTER   time written from the 'H' record
   IXFASCHL       003-BYTE   CHARACTER   type schema name length
   IXFASCHN       256-BYTE   CHARACTER   type schema name
   IXFATYPL       003-BYTE   CHARACTER   type name length
   IXFATYPN       256-BYTE   CHARACTER   type name
   IXFATABL       003-BYTE   CHARACTER   table name length
   IXFATABN       256-BYTE   CHARACTER   table name
   IXFAPNDX       010-BYTE   CHARACTER   subtable index of parent table
   IXFASNDX       005-BYTE   CHARACTER   starting column index of current
                                            table
   IXFAENDX       005-BYTE   CHARACTER   ending column index of current
                                            table
```

이 유형의 레코드 하나를 사용하여 계층 구조에 포함된 서브테이블을 설명합니다. 계층 구조에 속한 모든 서브테이블 레코드는 함께 저장해야 하며 해당되는 계층 구조 레코드 바로 다음에 나와야 합니다. 서브테이블은 하나 이상의 컬럼으로 구성되며 각 컬럼은 컬럼 레코드에서 설명됩니다. 서브테이블의 각 컬럼은 연속된 C 레코드 세트에서 설명되어야 합니다. 다음은 DB2 서브테이블 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉,

(총 레코드 *크기* - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 이 A 레코드를 작성할 때 DB2를 식별하는 응용프로그램 ID입니다.

IXFAYTYP

DB2 응용프로그램 레코드의 부속 유형 ″Y″가 되도록 지정합니다.

IXFADATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다. 이 필드는 IXFHDATE 값과 동일해야 합니다.

IXFATIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 IXFHTIME 값과 동일해야 합니다.

IXFASCHL

IXFASCHN 필드에서 서브테이블 스키마 이름의 길이(바이트)입니다.

IXFASCHN

서브테이블 스키마의 이름입니다.

IXFATYPL

IXFATYPN 필드에서 서브테이블 이름의 길이(바이트)입니다.

IXFATYPN

서브테이블의 이름입니다.

IXFATABL

IXFATABN 필드에서 테이블 이름의 길이(바이트)입니다.

IXFATABN

테이블 이름입니다.

IXFAPNDX

상위 서브테이블의 서브테이블 레코드 인덱스입니다. 이 서브테이블이 계층 구조의 루트인 경우 이 필드는 -1 값을 포함합니다.

IXFASNDX

이 서브테이블을 구성하는 컬럼 레코드의 시작 인덱스입니다.

IXFAENDX

이 서브테이블을 구성하는 컬럼 레코드의 종료 인덱스입니다.

```
DB2 CONTINUATION RECORD

   FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
   IXFARECL       006-BYTE   CHARACTER   record length
   IXFARECT       001-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       012-BYTE   CHARACTER   application identifier =
                                            'DB2    02.00'
   IXFACTYP       001-BYTE   CHARACTER   application specific data type = 'C'
   IXFADATE       008-BYTE   CHARACTER   date written from the 'H' record
   IXFATIME       006-BYTE   CHARACTER   time written from the 'H' record
   IXFALAST       002-BYTE   SHORT INT   last diskette volume number
   IXFATHIS       002-BYTE   SHORT INT   this diskette volume number
   IXFANEXT       002-BYTE   SHORT INT   next diskette volume number
```

이 레코드는 파일이 최종 볼륨이 아닌 한, 다중 볼륨 IXF 파일에 포함된 각 파일 종료 지점에 있습니다. 또한 파일이 첫 번째 볼륨이 아닌 한, 다중 볼륨 IXF 파일에 포함된 각 파일의 시작 지점에서도 찾을 수 있습니다. 이 레코드의 용도는 파일 순서를 유지하는 것입니다. 다음은 DB2 연속 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 이 A 레코드를 작성할 때 DB2를 식별하는 응용프로그램 ID입니다.

IXFACTYP

DB2 응용프로그램 레코드의 부속 유형 ″C″가 되도록 지정합니다.

IXFADATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다. 이 필드는 IXFHDATE 값과 동일해야 합니다.

IXFATIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 IXFHTIME 값과 동일해야 합니다.

IXFALAST

이 필드는 리틀 엔디안(little-endian) 형식의 2진 필드입니다. 값은 IXFATHIS 값보다 작아야 합니다.

IXFATHIS

이 필드는 리틀 엔디안(little-endian) 형식의 2진 필드입니다. 연속 볼륨에서 이 필드 값은 연속적이어야 합니다. 첫 번째 볼륨에는 1 값이 있습니다.

IXFANEXT

이 필드는 리틀 엔디안(little-endian) 형식의 2진 필드입니다. 레코드가 파일 시작 지점에 있지 않는 한, 값은 IXFATHIS보다 1이 더 커야 합니다. 시작 지점에 있는 경우 값은 영(0)이어야 합니다.

```
DB2 TERMINATE RECORD

   FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
   IXFARECL       006-BYTE   CHARACTER   record length
   IXFARECT       001-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       012-BYTE   CHARACTER   application identifier =
                                            'DB2    02.00'
   IXFAETYP       001-BYTE   CHARACTER   application specific data type =
                                            'E'
   IXFADATE       008-BYTE   CHARACTER   date written from the 'H' record
   IXFATIME       006-BYTE   CHARACTER   time written from the 'H' record
```

이 레코드는 IXF 파일의 종료 지점에 있는 EOF 마커입니다. 다음은 DB2 종료 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 *크기* - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 이 A 레코드를 작성할 때 DB2를 식별하는 응용프로그램 ID입니다.

IXFAETYP

DB2 응용프로그램 레코드의 부속 유형 ″E″가 되도록 지정합니다.

IXFADATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다. 이 필드는 IXFHDATE 값과 동일해야 합니다.

IXFATIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 IXFHTIME 값과 동일해야 합니다.

```
DB2 IDENTITY RECORD

   FIELD NAME     LENGTH    TYPE        COMMENTS
   ----------     -------   ---------   -------------
   IXFARECL       06-BYTE   CHARACTER   record length
   IXFARECT       01-BYTE   CHARACTER   record type = 'A'
   IXFAPPID       12-BYTE   CHARACTER   application identifier
   IXFATYPE       01-BYTE   CHARACTER   application specific record type = 'S'
   IXFADATE       08-BYTE   CHARACTER   application record creation date
   IXFATIME       06-BYTE   CHARACTER   application record creation time
   IXFACOLN       06-BYTE   CHARACTER   column number of the identity column
   IXFAITYP       01-BYTE   CHARACTER   generated always ('Y' or 'N')
   IXFASTRT       33-BYTE   CHARACTER   identity START AT value
   IXFAINCR       33-BYTE   CHARACTER   identity INCREMENT BY value
   IXFACACH       10-BYTE   CHARACTER   identity CACHE value
   IXFAMINV       33-BYTE   CHARACTER   identity MINVALUE
   IXFAMAXV       33-BYTE   CHARACTER   identity MAXVALUE
   IXFACYCL       01-BYTE   CHARACTER   identity CYCLE ('Y' or 'N')
   IXFAORDR       01-BYTE   CHARACTER   identity ORDER ('Y' or 'N')
   IXFARMRL       03-BYTE   CHARACTER   identity Remark length
   IXFARMRK      254-BYTE   CHARACTER   identity Remark value
```

다음은 DB2 ID 레코드에 포함된 필드입니다.

IXFARECL

레코드 길이 표시기입니다. 레코드 길이 표시기 다음에 나오는, PC/IXF 레코드 부분의 길이(바이트)를 지정하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. 각 A 레코드는 최소한 전체 IXFAPPID 필드를 포함하도록 충분히 길어야 합니다.

IXFARECT

IXF 레코드 유형으로, 이 레코드에 대해 A로 설정됩니다. 응용프로그램 레코드임을 의미합니다. 응용프로그램 ID에서 내포하는 데이터의 컨텐츠 및 형식에 대한 특정 정보를 포함하지 않는 프로그램에서는 이 레코드를 무시합니다.

IXFAPPID

응용프로그램에서 이 A 레코드를 작성할 때 DB2를 식별하는 응용프로그램 ID입니다.

IXFATYPE

응용프로그램 특정 레코드 유형입니다. 이 필드 값은 항상 ″S″입니다.

IXFADATE

파일이 작성된 날짜(*yyyymmdd* 양식)입니다. 이 필드는 IXFHDATE 값과 동일해야 합니다.

IXFATIME

파일이 작성된 시간(*hhmmss* 양식)입니다. 이 필드는 IXFHTIME 값과 동일해야 합니다.

IXFACOLN

테이블에서 ID 컬럼의 컬럼 수입니다.

IXFAITYP

ID 컬럼의 유형입니다. 값이 ″Y″이면 ID 컬럼은 항상 GENERATED입니다. 다른 모든 값은 컬럼 유형이 GENERATED BY DEFAULT임을 나타낸다고 해석됩니다.

IXFASTRT

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 START AT 값입니다.

IXFAINCR

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 INCREMENT BY 값입니다.

IXFACACH

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 CACHE 값입니다. ″1″ 값은 NO CACHE 옵션에 해당합니다.

IXFAMINV

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 MINVALUE입니다.

IXFAMAXV

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 MAXVALUE입니다.

IXFACYCL

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 CYCLE 값입니다. ″Y″ 값은 CYCLE 옵션에 해당하며 다른 값은 NO CYCLE에 해당합니다.

IXFAORDR

테이블 작성 시 CREATE TABLE문에 제공된 ID 컬럼의 ORDER 값입니다. ″Y″ 값은 ORDER 옵션에 해당하며 다른 값은 NO ORDER에 해당합니다.

IXFARMRL

IXFARMRK 필드에서 설명의 길이(바이트)입니다.

IXFARMRK

ID 컬럼과 연관된 사용자가 입력한 설명입니다. 오직 정보용 필드입니다. 데이터를 임포트할 때 데이터베이스 관리 프로그램에서는 이 필드를 사용하지 않습니다.

PC/IXF 데이터 유형

표 52. PC/IXF 데이터 유형

이름	**IXFCTYPE** 값	설명
BIGINT	492	IXFTMFRM에서 지정된 양식의 8바이트 정수. -9 223 372 036 854 775 808 및 9 223 372 036 854 775 807 사이의 정수로 표시합니다. IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다. IXFCLENG는 사용되지 않으며 공백을 포함해야 합니다.
BLOB, CLOB	404, 408	가변 길이 문자열. 문자열의 최대 길이는 컬럼 디스크립터 레코드의 IXFCLENG 필드에 포함되며 32 767바이트를 초과할 수 없습니다. 문자열 자체는 앞에 현재 길이 표시기가 나옵니다. 이때 표시기는 문자열 길이(바이트)를 지정하는 4바이트 정수입니다. 문자열은 코드 페이지의 IXFCSBCP에서 표시됩니다. 다음은 BLOB에만 적용됩니다. IXFCSBCP가 영(0)이면 문자열은 비트 데이터이고 변환 프로그램에 의해 변환될 수 없습니다. 다음은 CLOB에만 적용됩니다. IXFCDBCP가 영(0)이 아니면 문자열은 코드 페이지의 IXFCDBCP에서 2바이트 문자도 포함할 수 있습니다.
BLOB_LOCATION_ SPECIFIER 및 DBCLOB_ LOCATION_ SPECIFIER	960, 964, 968	고정 길이 필드로, 255바이트를 초과할 수 없습니다. LLS(LOB Location Specifier)는 코드 페이지의 IXFCSBCP에서 표시됩니다. IXFCSBCP가 영(0)이면 LLS는 비트 데이터이고 변환 프로그램에 의해 변환될 수 없습니다. IXFCDBCP가 영(0)이 아니면 문자열은 코드 페이지의 IXFCDBCP에서 2바이트 문자도 포함할 수 있습니다. LLS 길이는 IXFCLENG에 저장되므로 원래 LOB의 실제 길이는 유실됩니다. LLS의 길이를 사용하여 LOB가 작성되므로 이 유형의 컬럼을 포함하는 PC/IXF 파일은 LOB 필드를 재작성하는 데 사용할 수 없습니다.

표 52. PC/IXF 데이터 유형 (계속)

이름	**IXFCTYPE 값**	설명
BLOB_FILE, CLOB_FILE, DBCLOB_FILE	916, 920, 924	*name_length* 및 *name* 필드가 채워진 SQLFILE 구조를 포함하는 고정 길이 필드. 구조의 길이는 컬럼 디스크립터 레코드의 IXFCLENG 필드에 포함되며 255바이트를 초과할 수 없습니다. 파일 이름은 코드 페이지의 IXFCSBCP에서 표시됩니다. IXFCDBCP가 영(0)이 아니면 파일 이름은 코드 페이지의 IXFCDBCP에서 2바이트 문자도 포함할 수 있습니다. IXFCSBCP가 영(0)이면 파일 이름은 비트 데이터이고 변환 프로그램에 의해 변환될 수 없습니다. 구조의 길이는 IXFCLENG에 저장되므로 원래 LOB의 실제 길이는 유실됩니다. *sql_lobfile_len*의 길이를 사용하여 LOB가 작성되므로 BLOB_FILE, CLOB_FILE 또는 DBCLOB_FILE 유형의 컬럼을 포함하는 IXF 파일은 LOB 필드를 재작성하는 데 사용할 수 없습니다.
CHAR	452	고정 길이 문자열. 문자열 길이는 컬럼 디스크립터 레코드의 IXFCLENG 필드에 포함되며 254바이트를 초과할 수 없습니다. 문자열은 코드 페이지의 IXFCSBCP에서 표시됩니다. IXFCDBCP가 영(0)이 아니면 문자열은 코드 페이지의 IXFCDBCP에서 2바이트 문자도 포함할 수 있습니다. IXFCSBCP가 영(0)이면 문자열은 비트 데이터이고 변환 프로그램에 의해 변환될 수 없습니다.
DATE	384	그레고리력에 따르는 특정 시점. 각 날짜는 ISO(International Standards Organization) 형식의 10바이트 문자열(*yyyy-mm-dd*)입니다. 연도 부분의 범위는 `0001`에서 `9999` 사이입니다. 월 부분의 범위는 `01`에서 `12` 사이입니다. 일 부분의 범위는 `01`에서 *n* 사이입니다. 여기서 *n*은 월에 따라 달라집니다. 이때 유년 및 월의 일 수에 대한 일반 규칙을 사용합니다. 어떤 부분에서도 선행 영(0)은 생략될 수 없습니다. IXFCLENG는 사용되지 않으며 공백을 포함해야 합니다. 모든 PC ASCII 코드 페이지에서 DATE 내 유효한 문자는 불변값이므로 IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다.

표 52. PC/IXF 데이터 유형 (계속)

이름	**IXFCTYPE 값**	설명
DBCLOB	412	2바이트 문자의 가변 길이 문자열. 컬럼 디스크립터 레코드의 IXFCLENG 필드에서는 문자열에서 2바이트 문자의 최대 수를 지정하며 16 383을 초과할 수 없습니다. 문자열 자체는 앞에 현재 길이 표시기가 나옵니다. 이때 표시기는 문자열 길이(바이트)(즉, 이 정수 값은 문자열 길이의 절반에 해당함)를 지정하는 4바이트 정수입니다. C 레코드의 IXFCDBCP에 지정된 대로 문자열은 DBCS 코드 페이지에 있습니다. 문자열이 2바이트 문자 데이터로만 구성되었으므로 IXFCSBCP는 영(0)이어야 합니다. 주위에 시프트 인(Shift-In) 또는 시프트 아웃(Shift-Out) 문자는 없습니다.
DECIMAL	484	정밀도가 P(컬럼 디스크립터 레코드에서 IXFCLENG의 처음 3바이트로 지정됨) 및 스케일 S(IXFCLENG의 마지막 2바이트로 지정됨)의 압축 10진수. 압축 10진수의 길이(바이트)는 (P+2)/2입니다. 정밀도는 1에서 31 사이(경계 포함)의 홀수여야 합니다. 압축 10진수는 IXFTMFRM으로 지정된 내부 형식입니다. 여기서 PC의 압축 10진수는 System/370의 압축 10진수와 동일하게 정의됩니다. IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다.
DECFLOAT	996	10진 부동 소수점 값은 소수점이 있는 IEEE 754r 숫자입니다. 소수점 위치는 각 10진 부동 소수점 값에 저장됩니다. 10진 부동 소수점 숫자의 범위는 정밀도 자릿수가 16 또는 34이고 지수 범위가 각각 10-383에서 10+384 또는 10-6143에서 10+6144 사이인 숫자입니다. 16자릿수 값의 스토리지 길이는 8바이트이고 34자릿수 값의 스토리지 길이는 16바이트입니다.
FLOATING POINT	480	IXFCLENG이 설정된 값(8 또는 4)에 따라 결정되는 long(8바이트) 또는 short(4바이트) 부동 소수점 숫자입니다. 데이터는 IXFTMFRM에서 지정한 대로의 내부 머신 양식입니다. IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다. 4바이트 부동 소수점은 데이터베이스 관리 프로그램에서 지원되지 않습니다.
GRAPHIC	468	2바이트 문자의 고정 길이 문자열. 컬럼 디스크립터 레코드의 IXFCLENG 필드에서는 문자열에서 2바이트 문자의 수를 지정하며 127을 초과할 수 없습니다. 문자열의 실제 길이는 IXFCLENG 필드 값의 2배(바이트)입니다. C 레코드의 IXFCDBCP에 지정된 대로 문자열은 DBCS 코드 페이지에 있습니다. 문자열이 2바이트 문자 데이터로만 구성되었으므로 IXFCSBCP는 영(0)이어야 합니다. 주위에 시프트 인(Shift-In) 또는 시프트 아웃(Shift-Out) 문자는 없습니다.

표 52. PC/IXF 데이터 유형 (계속)

이름	**IXFCTYPE 값**	설명
INTEGER	496	IXFTMFRM에서 지정된 양식의 4바이트 정수. -2 147 483 648 및 +2 147 483 647 사이의 정수로 표시합니다. IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다. IXFCLENG는 사용되지 않으며 공백을 포함해야 합니다.
LONGVARCHAR	456	가변 길이 문자열. 문자열의 최대 길이는 컬럼 디스크립터 레코드의 IXFCLENG 필드에 포함되며 32 767바이트를 초과할 수 없습니다. 문자열 자체는 앞에 현재 길이 표시기가 나옵니다. 이때 표시기는 문자열 길이(바이트)를 지정하는 2바이트 정수입니다. 문자열은 코드 페이지의 IXFCSBCP에서 표시됩니다. IXFCDBCP가 영(0)이 아니면 문자열은 코드 페이지의 IXFCDBCP에서 2바이트 문자도 포함할 수 있습니다. IXFCSBCP가 영(0)이면 문자열은 비트 데이터이고 변환 프로그램에 의해 변환될 수 없습니다.
LONG VARGRAPHIC	472	2바이트 문자의 가변 길이 문자열. 컬럼 디스크립터 레코드의 IXFCLENG 필드에서는 문자열에서 2바이트 문자의 최대 수를 지정하며 16 383을 초과할 수 없습니다. 문자열 자체는 앞에 현재 길이 표시기가 나옵니다. 이때 표시기는 문자열 길이(바이트)(즉, 이 정수 값은 문자열 길이의 절반에 해당함)를 지정하는 2바이트 정수입니다. C 레코드의 IXFCDBCP에 지정된 대로 문자열은 DBCS 코드 페이지에 있습니다. 문자열이 2바이트 문자 데이터로만 구성되었으므로 IXFCSBCP는 영(0)이어야 합니다. 주위에 시프트 인(Shift-In) 또는 시프트 아웃(Shift-Out) 문자는 없습니다.
SMALLINT	500	IXFTMFRM에서 지정된 양식의 2바이트 정수. -32 768 및 +32 767 사이의 정수로 표시합니다. IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다. IXFCLENG는 사용되지 않으며 공백을 포함해야 합니다.
TIME	388	24시간제에 따르는 특정 시점. 각 시간은 ISO 형식의 8바이트 문자열입니다(*hh.mm.ss*). 시간 부분의 범위는 `00`에서 `24` 사이이고 기타 부분의 범위는 `00`에서 `59` 사이입니다. 시간이 `24`이면 다른 부분은 `00`입니다. 가장 낮은 시간은 `00.00.00`이고 가장 높은 시간은 `24.00.00`입니다. 어떤 부분에서도 선행 영(0)은 생략될 수 없습니다. IXFCLENG는 사용되지 않으며 공백을 포함해야 합니다. 모든 PC ASCII 코드 페이지에서 TIME 내 유효한 문자는 불변값이므로 IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다.

표 52. PC/IXF 데이터 유형 (계속)

이름	IXFCTYPE 값	설명
TIMESTAMP	392	분수 초 정밀도로 표시한 날짜 및 시간. 각 시간 소인은 *yyyy-mm-dd-hh.mm.ss.nnnnnn* 양식의 문자열입니다(연도, 월, 일, 시간, 분, 초, 분수 초). 버전 9.7부터 시간소인 정밀도는 컬럼 디스크립터 레코드의 IXFCLENG 필드에 포함되며 12를 초과할 수 없습니다. 버전 9.7 이전의 경우 IXFCLENG는 사용되지 않으며 공백을 포함해야 합니다. 모든 PC ASCII 코드 페이지에서 TIMESTAMP 내 유효한 문자는 불변값이므로 IXFCSBCP 및 IXFCDBCP는 유효(significant) 숫자가 아니며 영(0)이어야 합니다.
VARCHAR	448	가변 길이 문자열. 문자열의 최대 길이(바이트)는 컬럼 디스크립터 레코드의 IXFCLENG 필드에 포함되며 254바이트를 초과할 수 없습니다. 문자열 자체는 앞에 현재 길이 표시기가 나옵니다. 이때 표시기는 문자열 길이(바이트)를 지정하는 2바이트 정수입니다. 문자열은 코드 페이지의 IXFCSBCP에서 표시됩니다. IXFCDBCP가 영(0)이 아니면 문자열은 코드 페이지의 IXFCDBCP에서 2바이트 문자도 포함할 수 있습니다. IXFCSBCP가 영(0)이면 문자열은 비트 데이터이고 변환 프로그램에 의해 변환될 수 없습니다.
VARGRAPHIC	464	2바이트 문자의 가변 길이 문자열. 컬럼 디스크립터 레코드의 IXFCLENG 필드에서는 문자열에서 2바이트 문자의 최대 수를 지정하며 127을 초과할 수 없습니다. 문자열 자체는 앞에 현재 길이 표시기가 나옵니다. 이때 표시기는 문자열 길이(바이트)(즉, 이 정수 값은 문자열 길이의 절반에 해당함)를 지정하는 2바이트 정수입니다. C 레코드의 IXFCDBCP에 지정된 대로 문자열은 DBCS 코드 페이지에 있습니다. 문자열이 2바이트 문자 데이터로만 구성되었으므로 IXFCSBCP는 영(0)이어야 합니다. 주위에 시프트 인(Shift-In) 또는 시프트 아웃(Shift-Out) 문자는 없습니다.

PC/IXF 문자 또는 그래픽 컬럼에서 IXFCSBCP 및 IXFCDBCP 값의 모든 조합이 유효합니다. 유효하지 않은 (IXFCSBCP,IXFCDBCP) 조합의 PC/IXF 문자 또는 그래픽 컬럼은 유효하지 않은 데이터 유형입니다.

표 53. 유효한 PC/IXF 데이터 유형

PC/IXF 데이터 유형	유효한 (IXFCSBCP,IXFCDBCP) 쌍	유효하지 않은 (IXFCSBCP,IXFCDBCP) 쌍
CHAR, VARCHAR 또는 LONG VARCHAR	(0,0), (x,0) 또는 (x,y)	(0,y)
BLOB	(0,0)	(x,0), (0,y) 또는 (x,y)
CLOB	(x,0), (x,y)	(0,0), (0,y)

표 53. 유효한 PC/IXF 데이터 유형 (계속)

PC/IXF 데이터 유형	유효한 (IXFCSBCP,IXFCDBCP) 쌍	유효하지 않은 (IXFCSBCP,IXFCDBCP) 쌍
GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC 또는 DBCLOB	(0,y)	(0,0), (x,0) 또는 (x,y)
주: x 및 y는 영(0)이 아닙니다.		

PC/IXF 데이터 유형 설명

표 54. PC/IXF 파일 형식에 대해 승인할 수 있는 데이터 유형 양식

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
BIGINT	데이터베이스 컬럼과 동일한 BIGINT 컬럼이 작성됩니다.	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 컬럼이 승인됩니다. -9 223 372 036 854 775 808에서 9 223 372 036 854 775 807 사이의 범위를 벗어난 경우 개별 값이 거부됩니다.
BLOB	PC/IXF BLOB 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	다음 경우에 PC/IXF CHAR, VARCHAR, LONG VARCHAR, BLOB, BLOB_FILE 또는 BLOB_LOCATION_SPECIFIER 컬럼이 승인 가능합니다. • 데이터베이스 컬럼이 FOR BIT DATA로 표시됨 • PC/IXF 컬럼 1바이트 코드 페이지 값이 데이터베이스 컬럼의 SBCS CPGID와 동일하고 PC/IXF 컬럼 2바이트 코드 페이지 값이 0 또는 데이터베이스 컬럼의 DBCS CPGID와 동일합니다. PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC BLOB 컬럼도 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.

표 54. PC/IXF 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
CHAR	PC/IXF CHAR 컬럼이 작성됩니다. 데이터베이스 컬럼 길이, SBCS CPGID 값 및 DBCS CPGID 값이 PC/IXF 컬럼 디스크립터 레코드로 복사됩니다.	다음 경우에 PC/IXF CHAR, VARCHAR 또는 LONG VARCHAR 컬럼이 승인 가능합니다. • 데이터베이스 컬럼이 FOR BIT DATA로 표시됨 • PC/IXF 컬럼 1바이트 코드 페이지 값이 데이터베이스 컬럼의 SBCS CPGID와 동일하고 PC/IXF 컬럼 2바이트 코드 페이지 값이 0 또는 데이터베이스 컬럼의 DBCS CPGID와 동일합니다. 데이터베이스 컬럼이 FOR BIT DATA로 표시된 경우 PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC 컬럼도 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼 길이와 호환 가능해야 합니다. 필요한 경우 데이터는 1바이트 스페이스 오른쪽에 채워집니다(x'20').
CLOB	PC/IXF CLOB 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	PC/IXF 컬럼 1바이트 코드 페이지 값이 데이터베이스 컬럼의 SBCS CPGID와 동일하고 PC/IXF 컬럼 2바이트 코드 페이지 값이 0 또는 데이터베이스 컬럼의 DBCS CPGID와 동일하면 PC/IXF CHAR, VARCHAR, LONG VARCHAR, CLOB, CLOB_FILE 또는 CLOB_LOCATION_SPECIFIER 컬럼이 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.
DATE	데이터베이스 컬럼과 동일한 DATE 컬럼이 작성됩니다.	DATE 유형의 PC/IXF 컬럼은 일반 입력입니다. 임포트 유틸리티는 호환되지 않는 길이를 제외하고 모든 문자 유형의 컬럼을 승인하려고 합니다. PC/IXF 파일의 문자 컬럼은 목표 데이터베이스의 지역 코드와 일관된 형식의 날짜를 포함해야 합니다.
DBCLOB	PC/IXF DBCLOB 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	PC/IXF 컬럼 2바이트 코드 페이지 값이 데이터베이스 컬럼 값과 동일하면 PC/IXF GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, DBCLOB_FILE 또는 DBCLOB_LOCATION_SPECIFIER 컬럼이 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.

표 54. PC/IXF 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
DECIMAL	데이터베이스 컬럼과 동일한 DECIMAL 컬럼이 작성됩니다. 컬럼의 정밀도 및 스케일은 컬럼 디스크립터 레코드에 저장됩니다.	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 컬럼이 승인됩니다. 임포트하는 DECIMAL 컬럼 범위를 벗어나면 개별 값이 거부됩니다.
DECFLOAT	데이터베이스 컬럼과 동일한 DECFLOAT 컬럼이 작성됩니다. 컬럼의 정밀도는 컬럼 디스크립터 레코드에 저장됩니다.	SMALLINT, INTEGER, BIGINT(DECFLOAT(34)로만 해당), DECIMAL, FLOAT, REAL, DOUBLE 또는 DECFLOAT(16)(DECFLOAT(34)로만 해당) 유형의 컬럼이 승인됩니다. 기타 숫자 컬럼 유형은 DECFLOAT에 대해서만 유효하지만 값이 목표 정밀도를 벗어나면 값이 올림됩니다.
FLOAT	데이터베이스 컬럼과 동일한 FLOAT 컬럼이 작성됩니다.	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 컬럼이 승인됩니다. 모든 값이 범위에 포함됩니다.
GRAPHIC(DBCS만 해당)	PC/IXF GRAPHIC 컬럼이 작성됩니다. 데이터베이스 컬럼 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	PC/IXF 컬럼 2바이트 코드 페이지 값이 데이터베이스 컬럼 값과 동일하면 PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC 컬럼이 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼 길이와 호환 가능해야 합니다. 필요한 경우 데이터는 2바이트 스페이스 오른쪽에 채워집니다(x'8140').
INTEGER	데이터베이스 컬럼과 동일한 INTEGER 컬럼이 작성됩니다.	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 컬럼이 승인됩니다. -2 147 483 648에서 2 147 483 647 사이의 범위를 벗어난 경우 개별 값이 거부됩니다.
LONG VARCHAR	PC/IXF LONG VARCHAR 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	다음 경우에 PC/IXF CHAR, VARCHAR 또는 LONG VARCHAR 컬럼이 승인 가능합니다. • 데이터베이스 컬럼이 FOR BIT DATA로 표시됨 • PC/IXF 컬럼 1바이트 코드 페이지 값이 데이터베이스 컬럼의 SBCS CPGID와 동일하고 PC/IXF 컬럼 2바이트 코드 페이지 값이 0 또는 데이터베이스 컬럼의 DBCS CPGID와 동일합니다. 데이터베이스 컬럼이 FOR BIT DATA로 표시된 경우 PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC 컬럼도 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.

표 54. PC/IXF 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
LONG VARGRAPHIC(DBCS만 해당)	PC/IXF LONG VARGRAPHIC 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	PC/IXF 컬럼 2바이트 코드 페이지 값이 데이터베이스 컬럼 값과 동일하면 PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC 컬럼이 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.
SMALLINT	데이터베이스 컬럼과 동일한 SMALLINT 컬럼이 작성됩니다.	숫자 유형(SMALLINT, INTEGER, BIGINT, DECIMAL 또는 FLOAT)의 컬럼이 승인됩니다. -32 768에서 32 767 사이의 범위를 벗어난 경우 개별 값이 거부됩니다.
TIME	데이터베이스 컬럼과 동일한 TIME 컬럼이 작성됩니다.	TIME 유형의 PC/IXF 컬럼은 일반 입력입니다. 임포트 유틸리티는 호환되지 않는 길이를 제외하고 모든 문자 유형의 컬럼을 승인하려고 합니다. PC/IXF 파일의 문자 컬럼은 목표 데이터베이스의 지역 코드와 일관된 형식의 시간 데이터를 포함해야 합니다.
TIMESTAMP	데이터베이스 컬럼과 동일한 TIMESTAMP 컬럼이 작성됩니다.	TIMESTAMP 유형의 PC/IXF 컬럼은 일반 입력입니다. 임포트 유틸리티는 호환되지 않는 길이를 제외하고 모든 문자 유형의 컬럼을 승인하려고 합니다. PC/IXF 파일의 문자 컬럼은 시간소인에 대한 입력 형식의 데이터를 포함해야 합니다.
VARCHAR	데이터베이스 컬럼의 최대 길이가 254인 경우 PC/IXF VARCHAR 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이가 254보다 큰 경우 PC/IXF LONG VARCHAR 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	다음 경우에 PC/IXF CHAR, VARCHAR 또는 LONG VARCHAR 컬럼이 승인 가능합니다. • 데이터베이스 컬럼이 FOR BIT DATA로 표시됨 • PC/IXF 컬럼 1바이트 코드 페이지 값이 데이터베이스 컬럼의 SBCS CPGID와 동일하고 PC/IXF 컬럼 2바이트 코드 페이지 값이 0 또는 데이터베이스 컬럼의 DBCS CPGID와 동일합니다. 데이터베이스 컬럼이 FOR BIT DATA로 표시된 경우 PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC 컬럼도 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.

표 54. PC/IXF 파일 형식에 대해 승인할 수 있는 데이터 유형 양식 (계속)

데이터 유형	익스포트 유틸리티에서 작성한 파일의 양식	임포트 유틸리티에 대해 승인할 수 있는 양식
VARGRAPHIC(DBCS 만 해당)	데이터베이스 컬럼의 최대 길이가 127인 경우 PC/IXF VARGRAPHIC 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이가 127 보다 큰 경우 PC/IXF LONG VARGRAPHIC 컬럼이 작성됩니다. 데이터베이스 컬럼의 최대 길이, SBCS CPGID 값 및 DBCS CPGID 값이 컬럼 디스크립터 레코드로 복사됩니다.	PC/IXF 컬럼 2바이트 코드 페이지 값이 데이터베이스 컬럼 값과 동일하면 PC/IXF GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC 컬럼이 승인 가능합니다. PC/IXF 컬럼이 고정 길이인 경우 해당 길이는 데이터베이스 컬럼의 최대 길이와 호환 가능해야 합니다.

데이터베이스로 PC/IXF 파일 임포트를 제어하는 일반 규칙

데이터베이스 관리 프로그램 임포트 유틸리티는 SBCS 또는 DBCS 환경에서 PC/IXF 파일을 임포트할 때 다음 일반 규칙을 적용합니다.

- 임포트 유틸리티는 PC/IXF 형식 파일만(IXFHID = 'IXF')승인합니다. 다른 형식의 IXF 파일은 임포트할 수 없습니다.
- 임포트 유틸리티는 1024개 컬럼을 초과하는 PC/IXF 파일은 거부합니다.
- IXF 형식으로 익스포트하는 경우 ID가 IXF 형식에서 지원하는 최대 *크기*를 초과하면 익스포트 조작에 성공해도 CREATE 모드를 사용하는 후속 임포트 조작에서 결과로 생성된 데이터 파일을 사용할 수 없습니다. SQL27984W가 리턴됩니다.

  주: IMPORT 명령의 CREATE 및 REPLACE_CREATE 옵션은 사용되지 않으며 추후 릴리스에서 제거됩니다.
- PC/IXF 레코드의 IXFHSBCP 값은 SBCS CPGID와 같거나 목표 데이터베이스의 IXFHSBCP/IXFHDBCP 및 SBCS/DBCS CPGID 사이에 변환표가 있어야 합니다. IXFHDBCP 값은 '00000' 또는 목표 데이터베이스의 DBCS CPGID여야 합니다. 두 조건 중 하나를 만족하지 못하면 FORCEIN 옵션이 지정되지 않는 한, 임포트 유틸리티는 PC/IXF 파일을 거부합니다.
- 유효하지 않은 데이터 유형 - 새 테이블

  PC/IXF 파일을 *새* 테이블로 임포트하는 작업은 IMPORT 명령에서 CREATE 또는 REPLACE_CREATE 키워드로 지정됩니다. 새 테이블로 임포트하는 중 데이터 유형이 유효하지 않은 PC/IXF 컬럼이 선택되면 임포트 유틸리티가 종료됩니다. 전체 PC/IXF 파일이 거부되고 테이블이 작성되지 않으며 데이터가 임포트되지 않습니다.

- 유효하지 않은 데이터 유형 - 기존 테이블

  PC/IXF 파일을 *기존* 테이블로 임포트하는 작업은 IMPORT 명령에서 INSERT, INSERT_UPDATE, REPLACE 또는 REPLACE_CREATE 키워드로 지정됩니다. 기존 테이블로 임포트하는 중 데이터 유형이 유효하지 않은 PC/IXF 컬럼이 선택되면 다음 두 조치 중 하나가 가능합니다.

  – 목표 테이블 컬럼이 널(NULL) 입력 가능한 경우 유효하지 않은 PC/IXF 컬럼의 모든 값이 무시되고 테이블 컬럼 값이 널(NULL)로 설정됩니다.
  – 목표 테이블 컬럼이 널(NULL) 입력 가능하지 않은 경우 임포트 유틸리티가 종료됩니다. 전체 PC/IXF 파일이 거부되고 데이터가 임포트되지 않습니다. 기본 테이블은 변경되지 않고 남아 있습니다.

- 새 테이블로 임포트하는 중 널(NULL) 입력 가능한 PC/IXF 컬럼에서 널(NULL) 입력 가능한 데이터베이스 컬럼이 생성되고 널(NULL) 입력 가능하지 않은 PC/IXF 컬럼에서 널(NULL) 입력 가능하지 않은 데이터베이스 컬럼이 생성됩니다.
- 널(NULL) 입력 가능하지 않은 PC/IXF 컬럼은 널(NULL) 입력 가능한 데이터베이스 컬럼으로 임포트할 수 있습니다.
- 널(NULL) 입력 가능한 PC/IXF 컬럼은 널(NULL) 입력 가능하지 않은 데이터베이스 컬럼으로 임포트할 수 있습니다. PC/IXF 컬럼에 널(NULL) 값이 발견되면 임포트 유틸리티는 PC/IXF 행에서 널(NULL) 값을 포함하는 모든 컬럼 값을 거부하고 (전체 행이 거부됨) 다음 PC/IXF 행으로 처리를 계속합니다. 즉, 목표 테이블 컬럼(널(NULL)임)이 널(NULL) 입력 가능하지 않은 경우 널(NULL) 값을 포함하는 PC/IXF 행에서 데이터가 임포트되지 않습니다.
- 호환되지 않는 컬럼 - 새 테이블

  새 데이터베이스 테이블로 임포트하는 중 목표 데이터베이스 컬럼과 호환되지 않는 PC/IXF 컬럼이 선택되면 임포트 유틸리티가 종료됩니다. 전체 PC/IXF 파일이 거부되고 테이블이 작성되지 않으며 데이터가 임포트되지 않습니다.

  주: IMPORT의 FORCEIN 옵션은 호환 가능한 컬럼 범위를 확장합니다.

- 호환되지 않는 컬럼 - 기존 테이블

  기존 데이터베이스 테이블로 임포트하는 중 목표 데이터베이스 컬럼과 호환되지 않는 PC/IXF 컬럼이 선택되면 다음 두 조치 중 하나가 가능합니다.

  – 목표 테이블 컬럼이 널(NULL) 입력 가능한 경우 PC/IXF 컬럼의 모든 값이 무시되고 테이블 컬럼 값이 널(NULL)로 설정됩니다.
  – 목표 테이블 컬럼이 널(NULL) 입력 가능하지 않은 경우 임포트 유틸리티가 종료됩니다. 전체 PC/IXF 파일이 거부되고 데이터가 임포트되지 않습니다. 기본 테이블은 변경되지 않고 남아 있습니다.

  주: IMPORT의 FORCEIN 옵션은 호환 가능한 컬럼 범위를 확장합니다.

- 유효하지 않은 값

  임포트 중 목표 데이터베이스 컬럼에 대해 유효하지 않은 PC/IXF 컬럼 값이 발견되면 임포트 유틸리티는 PC/IXF 행에서 유효하지 않은 값을 포함하는 모든 컬럼 값을 거부하고(전체 행이 거부됨) 다음 PC/IXF 행으로 처리를 계속합니다.

데이터베이스로 PC/IXF 파일 임포트를 제어하는 데이터 유형별 규칙

- 유효한 PC/IXF 숫자 컬럼은 호환 가능한 숫자 데이터베이스 컬럼으로 임포트할 수 있습니다. 4바이트 부동 소수점 데이터를 포함하는 PC/IXF 컬럼은 해당 데이터가 유효하지 않은 데이터 유형이므로 임포트되지 않습니다.
- 데이터베이스 날짜/시간 컬럼은 컬럼 길이 및 값 호환성 제한사항에 따라 대응하는 PC/IXF 날짜/시간 컬럼(DATE, TIME 및 TIMESTAMP) 및 PC/IXF 문자 컬럼(CHAR, VARCHAR 및 LONG VARCHAR)의 값을 승인할 수 있습니다.
- 유효한 PC/IXF 문자 컬럼(CHAR, VARCHAR 또는 LONG VARCHAR)은 FOR BIT DATA로 표시된 *기존* 데이터베이스 문자 컬럼으로 항상 임포트할 수 있습니다.
  - IXFCSBCP 및 SBCS CPGID가 일치해야 합니다.
  - IXFCSBCP/IXFCDBCP 및 SBCS/DBCS에 대한 변환표가 있어야 합니다.
  - 한 세트가 모두 영(0)이어야 합니다(FOR BIT DATA).

  IXFCSBCP가 영(0)이 아니면 IXFCDBCP는 영(0) 또는 목표 데이터베이스 컬럼의 DBCS CPGID여야 합니다.

  두 조건 중 하나를 만족하지 못하면 PC/IXF 및 데이터베이스 컬럼은 호환되지 않습니다.

  유효한 PC/IXF 문자 컬럼을 *새* 데이터베이스 테이블로 임포트하는 경우 IXFCSBCP 값은 영(0) 또는 데이터베이스의 SBCS CPGID이거나 변환표가 있어야 합니다. IXFCSBCP가 영(0)이면 IXFCDBCP도 영(0)이어야 합니다. 그렇지 않으면 PC/IXF 컬럼은 유효하지 않은 데이터 유형입니다. IMPORT에서 새 테이블에 FOR BIT DATA로 표시된 문자 컬럼을 작성합니다. IXFCSBCP가 영(0)이 아니고 데이터베이스의 SBCS CPGID와 같으면 IXFCSBCP 값은 영(0) 또는 데이터베이스의 DBCS CPGID와 같아야 합니다. 이 경우 유틸리티는 새 테이블에서 SBCS 및 DBCS CPGID 값이 데이터베이스의 해당 값과 동일한 문자 컬럼을 작성합니다. 이 조건을 만족하지 못하면 PC/IXF 및 데이터베이스 컬럼은 호환되지 않습니다.

  FORCEIN 옵션을 사용하여 코드 페이지 등식 검사를 겹쳐쓸 수 있습니다. 그러나 IXFCSBCP가 영(0)이고 IXFCDBCP가 영(0)이 아닌 PC/IXF 문자 컬럼은 유효하지 않은 데이터 유형이므로 FORCEIN을 지정한 경우에도 임포트할 수 없습니다.
- 유효한 PC/IXF 그래픽 컬럼(GRAPHIC, VARGRAPHIC 또는 LONG VARGRAPHIC)은 FOR BIT DATA로 표시된 *기존* 데이터베이스 문자 컬럼으로

항상 임포트할 수 있습니다. 그러나 다른 모든 데이터베이스 컬럼과는 호환되지 않습니다. FORCEIN 옵션을 사용하여 이 제한을 완화할 수 있습니다. 그러나 IXFCSBCP가 영(0)이 아니거나 IXFCDBCP가 영(0)인 PC/IXF 그래픽 컬럼은 유효하지 않은 데이터 유형이므로 FORCEIN을 지정한 경우에도 임포트할 수 없습니다.

유효한 PC/IXF 그래픽 컬럼을 데이터베이스 그래픽 컬럼으로 임포트하는 경우 IXFCDBCP 값은 목표 데이터베이스 컬럼의 DBCS CPGID와 동일해야 합니다. 즉, 두 컬럼의 2바이트 코드 페이지가 일치해야 합니다.

- PC/IXF 파일을 기존 데이터베이스 테이블로 임포트하는 중 길이가 목표 컬럼의 최대 길이보다 큰 고정 길이 문자열(CHAR 또는 GRAPHIC)이 선택되면 컬럼은 호환되지 않습니다.
- PC/IXF 파일을 기존 데이터베이스 테이블로 임포트하는 중 길이가 목표 컬럼의 최대 길이보다 큰 가변 길이 문자열(VARCHAR, LONG VARCHAR, VARGRAPHIC 또는 LONG VARGRAPHIC)이 선택되면 컬럼은 *호환 가능*합니다. 개별 값은 데이터베이스 관리 프로그램 INSERT문으로 제어하는 호환성 규칙에 따라 처리되며 목표 데이터베이스 컬럼에서 너무 긴 PC/IXF 값은 유효하지 않습니다.
- 고정 길이 데이터베이스 *문자* 컬럼(즉, CHAR 컬럼)으로 임포트된 PC/IXF 값의 경우 필요하면 데이터베이스 컬럼 길이와 동일한 길이 값을 확보하기 위해 1바이트 스페이스(0x20)를 오른쪽부터 채웁니다. 고정 길이 데이터베이스 *그래픽* 컬럼(즉, GRAPHIC 컬럼)으로 임포트된 PC/IXF 값의 경우 필요하면 데이터베이스 컬럼 길이와 동일한 길이 값을 확보하기 위해 2바이트 스페이스(0x8140)를 오른쪽부터 채웁니다.
- PC/IXF VARCHAR 컬럼의 최대 길이는 254바이트이므로 최대 길이 *n*(254 *n* 4001)의 데이터베이스 VARCHAR 컬럼을 최대 길이 *n*의 PC/IXF LONG VARCHAR 컬럼으로 익스포트해야 합니다.
- PC/IXF LONG VARCHAR 컬럼의 최대 길이가 32 767바이트이고 데이터베이스 LONG VARCHAR 컬럼의 최대 길이 제한이 32 700바이트이지만 32 700바이트를 초과하고 32 768바이트 미만인 PC/IXF LONG VARCHAR 컬럼은 여전히 유효하며 데이터베이스 LONG VARCHAR 컬럼으로 임포트할 수 있지만 데이터가 유실될 수 있습니다.
- PC/IXF VARGRAPHIC 컬럼의 최대 길이는 127바이트이므로 최대 길이 *n*(127 *n* 2001)의 데이터베이스 VARGRAPHIC 컬럼을 최대 길이 *n*의 PC/IXF LONG VARGRAPHIC 컬럼으로 익스포트해야 합니다.
- PC/IXF LONG VARGRAPHIC 컬럼의 최대 길이가 16 383바이트이고 데이터베이스 LONG VARGRAPHIC 컬럼의 최대 길이 제한이 16 350바이트이지만 16 350바이트를 초과하고 16 384바이트 미만인 PC/IXF LONG VARGRAPHIC 컬럼은 여전히 유효하며 데이터베이스 LONG VARGRAPHIC 컬럼으로 임포트할 수 있지만 데이터가 유실될 수 있습니다.

표 55 및 표 56에서는 FORCEIN 옵션 없이 새 또는 기존 데이터베이스 테이블로의 PC/IXF 파일 임포트를 요약합니다.

표 55. FORCEIN 옵션 숫자 유형을 포함하지 않는 PC/IXF 파일 임포트 요약

	데이터베이스 컬럼 데이터 유형					
PC/IXF COLUMN DATA TYPE	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**DFP**	**FLT**
-SMALLINT	N					
	E	E	E	E[a]	E	E
-INTEGER		N				
	E[a]	E	E	E[a]	E	E
-BIGINT			N			
	E[a]	E[a]	E	E[a]	E	E
-DECIMAL				N		
	E[a]	E[a]	E[a]	E[a]	E	E
-DECFLOAT						
	E[a]	E[a]	E[a]	E[a]	E	E[a]
-FLOAT						N
	E[a]	E[a]	E[a]	E[a]	E	E
[a] 목표 숫자 데이터 유형 범위를 벗어나면 개별 값이 거부됩니다.						

표 56. FORCEIN 옵션 문자, 그래픽 및 날짜/시간 유형을 포함하지 않는 PC/IXF 파일 임포트 요약

	데이터베이스 컬럼 데이터 유형						
PC/IXF COLUMN DATA TYPE	**(0,0)**	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
-(0,0)	N						
	E				E[c]	E[c]	E[c]
-(SBCS,0)		N	N				
	E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)			N		E[c]	E[c]	E[c]
	E		E				
-GRAPHIC				N			
	E			E			
-DATE					N		
					E		
-TIME						N	
						E	
-TIME STAMP							N
							E

표 56. FORCEIN 옵션 문자, 그래픽 및 날짜/시간 유형을 포함하지 않는 PC/IXF 파일 임포트 요약 (계속)

	데이터베이스 컬럼 데이터 유형						
PC/IXF COLUMN DATA TYPE	**(0,0)**	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**

[b] 데이터 유형이 DBCS 환경에서만 사용 가능합니다.

[c] 유효한 날짜 또는 시간 값이 아니면 개별 값이 거부됩니다.

[d] 데이터 유형은 DBCS 환경에서 사용 불가능합니다.

주:

1. 테이블은 유효한 모든 PC/IXF 및 데이터베이스 관리 프로그램 데이터 유형으로 구성된 매트릭스입니다. PC/IXF 컬럼을 데이터베이스 컬럼으로 임포트할 수 있으면 데이터베이스 관리 프로그램 데이터 유형 매트릭스 컬럼 및 PC/IXF 데이터 유형 매트릭스 행의 교차 지점에서 매트릭스 셀에 문자가 표시됩니다. 'N'은 유틸리티가 새 데이터베이스 테이블을 작성함을 나타냅니다(표시된 데이터 유형의 데이터베이스 컬럼이 작성됨). 'E'는 유틸리티가 기존 데이터베이스 테이블로 데이터를 임포트함을 나타냅니다(표시된 데이터 유형의 데이터베이스 컬럼이 유효한 목표임).
2. 문자열 데이터 유형은 코드 페이지 속성으로 구별됩니다. 이러한 속성은 정렬된 쌍(SBCS,DBCS)으로 표시됩니다. 여기서,
   - SBCS는 영(0)이거나 문자 데이터 유형의 1바이트 코드 페이지 속성에 대해 영(0)이 아닌 값으로 표시됩니다.
   - DBCS는 영(0)이거나 문자 데이터 유형의 2바이트 코드 페이지 속성에 대해 영(0)이 아닌 값으로 표시됩니다.
3. 테이블에서 PC/IXF 문자 컬럼을 데이터베이스 문자 컬럼으로 임포트할 수 있다고 표시하면 각각의 코드 페이지 속성 쌍에 대한 값이 코드 페이지 등식을 제어하는 규칙을 만족합니다.

PC/IXF 및 버전 0 System/370 IXF 간 다른 점

다음에서는 여러 호스트 데이터베이스 제품에서 사용하는 버전 0 System/370 IXF 및 데이터베이스 관리 프로그램에서 사용하는 PC/IXF 간 다른 점에 대해 설명합니다.

- PC/IXF 파일은 EBCDIC 지향이 아닌 ASCII입니다. PC/IXF 파일은 컬럼 디스크립터 레코드의 실제 코드 페이지 값 사용 및 H 레코드의 새 코드 페이지 ID를 포함하여 코드 페이지 ID를 크게 확장합니다. 또한 문자 데이터 컬럼을 FOR BIT DATA로 표시하는 메커니즘이기도 합니다. FOR BIT DATA 컬럼은 특별한 의미를 지닙니다. PC/IXF 파일 형식 및 기타 IXF 또는 데이터베이스 파일 형식 간 변환이 FOR BIT DATA 컬럼에 포함된 값에서 코드 페이지 변환을 수행할 수 없기 때문입니다.
- 머신 데이터 양식만 허용됩니다. 즉, IXFTFORM 필드는 항상 M 값을 포함해야 합니다. 또한 머신 데이터는 PC 양식이어야 합니다. 즉, IXFTMFRM 필드는 PC 값

을 포함해야 합니다. 이는 PC/IXF 데이터 레코드의 데이터 부분에서 정수, 부동 소수점 숫자 및 10진수가 PC 양식이어야 함을 나타냅니다.

- 응용프로그램(A) 레코드는 PC/IXF 파일에서 H 레코드 다음에 모든 위치에서 허용됩니다. IXFHHCNT 필드 값이 계산될 때 계수에 포함되지 않습니다.
- 모든 PC/IXF 레코드는 레코드 길이 표시기로 시작합니다. 이는 레코드 길이 표시기를 포함하지 않는 PC/IXF 레코드의 길이(바이트)를 포함하는 정수 값을 6바이트 문자로 표시합니다. 즉, (총 레코드 크기 - 6바이트)입니다. 레코드 길이 필드는 PC 프로그램에서 레코드 경계를 식별할 수 있도록 하기 위해 제공됩니다.
- 가변 길이 데이터의 압축 스토리지를 활용하고 필드가 여러 레코드로 분할될 때 복잡한 처리를 피하기 위해 PC/IXF는 버전 0 IXF X 레코드를 지원하지 않지만 D 레코드 ID는 지원합니다. 가변 길이 필드 또는 널(NULL) 입력 가능한 필드가 데이터 D 레코드의 마지막 필드이면 언제나 PC/IXF 파일에 전체 필드 최대 길이를 작성하지 않아도 됩니다.

FORCEIN 옵션

forcein 파일 유형 수정자는 PC/IXF 파일의 데이터 및 목표 데이터베이스 간 코드 페이지 차이에도 불구하고 PC/IXF 파일 임포트를 허용합니다. 호환 가능 컬럼 정의에도 추가로 유연성을 제공합니다.

forcein의 일반 시맨틱

다음 일반 시맨틱은 SBCS 또는 DBCS 환경에서 forcein 파일 유형 수정자를 사용하는 경우 적용됩니다.

- forcein 파일 유형 수정자는 주의하여 사용해야 합니다. 보통 이 옵션을 해제하고 임포트하는 것이 좋습니다. 그러나 PC/IXF 데이터 교환 아키텍처의 일반적인 특성 때문에 일부 PC/IXF 파일은 개입 없이는 임포트할 수 없는 데이터 유형 또는 값을 포함할 수 있습니다.
- forcein을 사용하여 *새* 테이블에 임포트하는 경우 기존 테이블에 임포트하는 경우와 다른 결과가 나타납니다. 기존 테이블에 각 PC/IXF 데이터 유형에 대한 사전 정의된 목표 데이터 유형이 있습니다.
- lobsinfile 파일 유형 수정자로 LOB 데이터를 익스포트하고 다른 코드 페이지에서 다른 클라이언트로 파일을 이동하면 기타 데이터와는 달리 별도의 파일에 있는 CLOBS 및 DBCLOBS는 데이터베이스로 임포트 또는 로드할 때 클라이언트 코드 페이지로 변환되지 않습니다.

forcein의 코드 페이지 시맨틱

다음 코드 페이지 시맨틱은 SBCS 또는 DBCS 환경에서 forcein 파일 유형 수정자를 사용하는 경우 적용됩니다.

- forcein 파일 유형 수정자는 모든 임포트 유틸리티 코드 페이지 비교를 사용 불가능하게 합니다.

  이 규칙은 컬럼 레벨 및 새 또는 기존 데이터베이스 테이블로 임포트하는 경우 파일 레벨에서 코드 페이지 비교에 적용됩니다. 컬럼 레벨(예: 데이터 유형)의 경우 이 규칙은 다음 데이터베이스 관리 프로그램 및 PC/IXF 데이터 유형: 문자(CHAR, VARCHAR 및 LONG VARCHAR) 및 그래픽(GRAPHIC, VARGRAPHIC 및 LONG VARGRAPHIC)에만 적용됩니다. 제한사항은 기타 데이터 유형의 코드 페이지 속성이 데이터 유형 값의 해석과는 무관하다는 점에서 비롯됩니다.
- forcein은 데이터 유형을 판별하는 코드 페이지 속성의 검사를 해제하지 않습니다.

  예를 들어 데이터베이스 관리 프로그램에서는 CHAR 컬럼을 FOR BIT DATA 속성으로 선언할 수 있습니다. 이러한 선언에서는 컬럼의 SBCS CPGID 및 DBCS CPGID를 영(0)으로 설정합니다. 문자열이 아닌 비트 문자열로 컬럼을 식별하는 이러한 CPGID의 영(0) 값입니다.
- forcein은 코드 페이지 변환을 함축하지 않습니다.

  forcein 파일 유형 수정자에 반응하는 데이터 유형의 값은 "그대로" 복사됩니다. 코드 페이지 환경의 변경을 고려하여 코드 포인트 맵핑은 사용되지 않습니다. 고정 길이 목표 컬럼의 경우 스페이스를 포함한 임포트된 값을 채워야 합니다.
- forcein을 사용하여 *기존* 테이블에 데이터를 임포트하는 경우:
  - 목표 데이터베이스 및 컬럼의 코드 페이지 값이 항상 전달됩니다.
  - PC/IXF 파일 및 컬럼의 코드 페이지 값이 무시됩니다.

  이 규칙에서 forcein 사용 여부를 적용합니다. 데이터베이스 관리 프로그램에서는 데이터베이스를 작성한 후 데이터베이스 또는 컬럼 코드 페이지 값의 변경을 허용하지 않습니다.
- forcein을 사용하여 *새* 테이블에 임포트하는 경우:
  - 목표 데이터베이스의 코드 페이지 값이 전달됩니다.
  - IXFCSBCP = IXFCDBCP = 0인 PC/IXF 문자 컬럼은 FOR BIT DATA로 표시된 테이블 컬럼을 생성합니다.
  - 기타 모든 PC/IXF 문자 컬럼은 SBCS 및 DBCS CPGID 값이 데이터베이스와 동일한 테이블 문자 컬럼을 생성합니다.
  - PC/IXF 그래픽 컬럼은 SBCS CPGID가 "정의되지 않음"이고 DBCS CPGID가 데이터베이스와 같은 테이블 그래픽 컬럼을 생성합니다(DBCS 데이터베이스 환경만 해당).

forcein에 대한 예

IXFCSBCP = ’00897’ 및 IXFCDBCP = ’00301’과 같이 설정된 PC/IXF CHAR 컬럼을 고려합니다. 이 컬럼은 SBCS CPGID = ’00850’ 및 DBCS CPGID = ’00000’와 같은 데이터베이스 CHAR 컬럼으로 임포트됩니다. forcein을 사용하지 않으면 유틸리티는 종료되고 데이터는 임포트되지 않거나 PC/IXF 컬럼 값이 무시되고 데이터베이스 컬럼은 널(NULL) 값을 포함합니다(데이터베이스 컬럼이 널(NULL) 입력 가능한 경우). forcein을 사용하면 코드 페이지 비호환성을 무시하고 유틸리티가 진행됩니다. 기타 데이터 유형 비호환성(예: 길이)이 없는 경우 PC/IXF 컬럼의 값은 ″그래도″ 임포트되며 데이터베이스 컬럼 코드 페이지 환경에서 해석이 가능합니다.

다음 두 표에서 다음을 보여줍니다.

- 지정된 코드 페이지 속성의 PC/IXF 파일 데이터 유형을 임포트할 때 *새* 데이터베이스 테이블에서 작성된 컬럼의 코드 페이지 속성.
- 유효하지 않거나 호환되지 않는 경우 임포트 유틸리티에서 PC/IXF 데이터 유형을 거부합니다.

표 57. SBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(새 테이블) 요약. 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다. 이 경우 항목 3 및 4는 forcein 없이 성공적으로 사용할 수 있습니다.

PC/IXF 데이터 유형의 코드 페이지 속성	**데이터베이스 테이블 컬럼의 코드 페이지 속성**	
	forcein을 포함하지 않는 경우	**forcein을 포함하는 경우**
(0,0)	(0,0)	(0,0)
(a,0)	(a,0)	(a,0)
(x,0)	거부	(a,0)
(x,y)	거부	(a,0)
(a,y)	거부	(a,0)
(0,y)	거부	(0,0)
주: 1. 표 58의 참고를 참조하십시오.		

표 58. DBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(새 테이블) 요약. 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다.

PC/IXF 데이터 유형의 코드 페이지 속성	**데이터베이스 테이블 컬럼의 코드 페이지 속성**	
	forcein을 포함하지 않는 경우	**forcein을 포함하는 경우**
(0,0)	(0,0)	(0,0)
(a,0)	(a,b)	(a,b)
(x,0)	거부	(a,b)
(a,b)	(a,b)	(a,b)
(x,y)	거부	(a,b)
(a,y)	거부	(a,b)
(x,b)	거부	(a,b)
(0,b)	(-,b)	(-,b)
(0,y)	거부	(-,b)

표 58. DBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(새 테이블) 요약 (계속). 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다.

PC/IXF 데이터 유형의 코드 페이지 속성	데이터베이스 테이블 컬럼의 코드 페이지 속성	
	forcein을 포함하지 않는 경우	**forcein**을 포함하는 경우

주:

1. PC/IXF 데이터 유형의 코드 페이지 속성은 정렬된 쌍으로 표시됩니다. 여기서 x는 영(0)이 아닌 1바이트 코드 페이지 값을 나타내고 y는 영(0)이 아닌 2바이트 코드 페이지 값을 나타냅니다. '-'는 정의되지 않은 코드 페이지 값을 나타냅니다.
2. 다양한 코드 페이지 속성 쌍에서 다른 문자의 사용은 계획된 것입니다. 다른 문자는 다른 규칙을 함축합니다. 예를 들어 PC/IXF 데이터 유형이 (x,y)로 표시되고 데이터베이스 컬럼이 (a,y)로 표시되는 경우 x는 a와 같지 않지만 PC/IXF 파일 및 데이터베이스는 동일한 2바이트 코드 페이지 값 y를 보유합니다.
3. `forcein` 코드 페이지 시맨틱은 문자 및 그래픽 데이터 유형에만 영향을 줍니다.
4. 새 테이블을 포함하는 데이터베이스에서 코드 페이지 속성이 (a,0)라고 가정합니다. 따라서 새 테이블의 모든 문자 컬럼에서 코드 페이지 속성은 (0,0) 또는 (a,0)입니다.

   DBCS 환경에서 새 테이블을 포함하는 데이터베이스에서 코드 페이지 속성이 (a,b)라고 가정합니다. 따라서 새 테이블의 모든 그래픽 컬럼에서 코드 페이지 속성은 (-,b)이고 모든 문자 컬럼에서 코드 페이지 속성은 (a,b)입니다. SBCS CPGID는 그래픽 데이터 유형에 대해 정의되지 않았으므로 '-'로 표시됩니다.
5. 결과의 데이터 유형은 `forcein`에 대한 데이터 유형 시맨틱에서 설명한 규칙으로 판별됩니다.
6. `reject` 결과는 유효하지 않거나 호환되지 않는 데이터 유형의 규칙이 반영된 것입니다.

다음 두 표에서 다음을 보여줍니다.

- 임포트 유틸리티는 지정된 코드 페이지 속성이 적용되는 *기존* 테이블 컬럼(목표 컬럼)에서의 다양한 코드 페이지 속성을 포함하는 PC/IXF 데이터 유형 사용을 승인합니다.
- 임포트 유틸리티는 특정 코드 페이지 속성을 포함하는 PC/IXF 데이터 유형을 표시된 코드 페이지 속성이 적용되는 *기존* 테이블 컬럼으로 임포트할 것을 허용하지 않습니다. 유효하지 않거나 호환되지 않는 경우 유틸리티에서 PC/IXF 데이터 유형을 거부합니다.

표 59. SBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(기존 테이블) 요약. 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다.

PC/IXF 데이터 유형의 코드 페이지 속성	데이터베이스 컬럼의 코드 페이지 속성	임포트 결과	
		forcein을 포함하지 않는 경우	**forcein**을 포함하는 경우
(0,0)	(0,0)	승인	승인
(a,0)	(0,0)	승인	승인
(x,0)	(0,0)	승인	승인
(x,y)	(0,0)	승인	승인
(a,y)	(0,0)	승인	승인

표 59. SBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(기존 테이블) 요약 (계속). 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다.

PC/IXF 데이터 유형의 코드 페이지 속성	데이터베이스 컬럼의 코드 페이지 속성	임포트 결과	
		forcein을 포함하지 않는 경우	**forcein**을 포함하는 경우
(0,y)	(0,0)	승인	승인
(0,0)	(a,0)	널(null) 또는 거부	승인
(a,0)	(a,0)	승인	승인
(x,0)	(a,0)	널(null) 또는 거부	승인
(x,y)	(a,0)	널(null) 또는 거부	승인
(a,y)	(a,0)	널(null) 또는 거부	승인
(0,y)	(a,0)	널(null) 또는 거부	널(null) 또는 거부

주:

1. 502 페이지의 표 57의 참고를 참조하십시오.
2. `null or reject` 결과는 유효하지 않거나 호환되지 않는 데이터 유형의 규칙이 반영된 것입니다.

표 60. DBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(기존 테이블) 요약. 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다.

PC/IXF 데이터 유형의 코드 페이지 속성	데이터베이스 컬럼의 코드 페이지 속성	임포트 결과	
		forcein을 포함하지 않는 경우	**forcein**을 포함하는 경우
(0,0)	(0,0)	승인	승인
(a,0)	(0,0)	승인	승인
(x,0)	(0,0)	승인	승인
(a,b)	(0,0)	승인	승인
(x,y)	(0,0)	승인	승인
(a,y)	(0,0)	승인	승인
(x,b)	(0,0)	승인	승인
(0,b)	(0,0)	승인	승인
(0,y)	(0,0)	승인	승인
(0,0)	(a,b)	널(null) 또는 거부	승인
(a,0)	(a,b)	승인	승인
(x,0)	(a,b)	널(null) 또는 거부	승인
(a,b)	(a,b)	승인	승인
(x,y)	(a,b)	널(null) 또는 거부	승인
(a,y)	(a,b)	널(null) 또는 거부	승인
(x,b)	(a,b)	널(null) 또는 거부	승인
(0,b)	(a,b)	널(null) 또는 거부	널(null) 또는 거부
(0,y)	(a,b)	널(null) 또는 거부	널(null) 또는 거부

표 60. DBCS에 대한 임포트 유틸리티 코드 페이지 시맨틱(기존 테이블) 요약 (계속). 이 테이블에서는 a 및 x 사이에 변환표가 없다고 가정합니다.

PC/IXF 데이터 유형의 코드 페이지 속성	데이터베이스 컬럼의 코드 페이지 속성	임포트 결과	
		forcein을 포함하지 않는 경우	forcein을 포함하는 경우
(0,0)	(-,b)	널(null) 또는 거부	승인
(a,0)	(-,b)	널(null) 또는 거부	널(null) 또는 거부
(x,0)	(-,b)	널(null) 또는 거부	널(null) 또는 거부
(a,b)	(-,b)	널(null) 또는 거부	널(null) 또는 거부
(x,y)	(-,b)	널(null) 또는 거부	널(null) 또는 거부
(a,y)	(-,b)	널(null) 또는 거부	널(null) 또는 거부
(x,b)	(-,b)	널(null) 또는 거부	널(null) 또는 거부
(0,b)	(-,b)	승인	승인
(0,y)	(-,b)	널(null) 또는 거부	승인

주:

1. 502 페이지의 표 57의 참고를 참조하십시오.
2. null or reject 결과는 유효하지 않거나 호환되지 않는 데이터 유형의 규칙이 반영된 것입니다.

forcein에 대한 데이터 유형 시맨틱

forcein 파일 유형 수정자는 동일하지 않거나 호환되지 않는 데이터 유형의 목표 데이터베이스 컬럼으로의 특정 PC/IXF 컬럼 임포트를 허용합니다. 다음 데이터 유형 시맨틱은 SBCS 또는 DBCS 환경(참고한 경우 제외)에서 forcein을 사용할 때 적용됩니다.

- SBCS 환경에서 forcein은 다음의 임포트를 허용합니다.
  - 데이터베이스 문자 컬럼(영(0)이 아닌 SBCS CPGID 및 DBCS CPGID = 0)으로의 PC/IXF BIT 데이터 유형(PC/IXF 문자 컬럼에서 IXFCSBCP = 0 = IXFCDBCP임). 기존 테이블만 해당
  - 데이터베이스 문자 컬럼으로의 PC/IXF MIXED 데이터 유형(IXFCSBCP 및 IXFCDBCP는 영(0)이 아님). 새 테이블 및 기존 테이블 모두 해당.
  - 데이터베이스 FOR BIT DATA 컬럼(SBCS CPGID = 0 = DBCS CPGID)으로의 PC/IXF GRAPHIC 데이터 유형. 새 테이블만 해당(기존 테이블에서는 항상 허용됨).
- forcein 파일 유형 수정자는 유효한 PC/IXF 데이터 유형의 범위를 확장하지 않습니다.

  데이터 유형이 유효한 PC/IXF 데이터 유형으로 정의되지 않은 PC/IXF 컬럼은 forcein을 사용하든, 사용하지 않든 임포트에서 유효하지 않습니다.
- DBCS 환경에서 forcein은 다음의 임포트를 허용합니다.
  - 데이터베이스 문자 컬럼으로의 PC/IXF BIT 데이터 유형

– 데이터베이스 그래픽 컬럼으로의 PC/IXF BIT 데이터 유형. 그러나 PC/IXF BIT 컬럼이 고정 길이인 경우 해당 길이는 짝수여야 합니다. 고정 길이 PC/IXF BIT 컬럼 길이가 홀수이면 데이터베이스 그래픽 컬럼과 호환 가능하지 않습니다. 가변 길이 PC/IXF BIT 컬럼은 길이가 홀수이든, 짝수이든, 가변 길이 컬럼의 홀수 길이 값이 데이터베이스 그래픽 컬럼으로 임포트할 때 유효하지 않은 값이어도 *호환 가능*합니다.
– 데이터베이스 문자 컬럼으로의 PC/IXF MIXED 데이터 유형.

표 61에서는 `forcein`이 지정된 새 또는 기존 데이터베이스 테이블로의 PC/IXF 파일 임포트를 요약합니다.

표 61. `forcein`을 포함하는 PC/IXF 파일 임포트 요약

	데이터베이스 컬럼 데이터 유형											
PC/IXF COLUMN DATA TYPE	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRA PH[b]**	**DATE**	**TIME**	**TIME STAMP**
-SMALLINT	N											
	E	E	E	E[a]	E							
-INTEGER		N										
	E[a]	E	E	E[a]	E							
-BIGINT			N									
	E[a]	E[a]	E	E[a]	E							
-DECIMAL				N								
	E[a]	E[a]	E[a]	E[a]	E							
-FLOAT					N							
	E[a]	E[a]	E[a]	E[a]	E							
-(0,0)						N						
						E	E w/F	E w/F	E w/F	E[c]	E[c]	E[c]
-(SBCS,0)							N	N				
						E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)							N w/F[d]	N		E[c]	E[c]	E[c]
						E	E w/F	E				
-GRAPHIC						N w/F[d]			N			
						E			E			
-DATE										N		
										E		
-TIME											N	
											E	

표 61. forcein을 포함하는 PC/IXF 파일 임포트 요약 (계속)

	데이터베이스 컬럼 데이터 유형											
PC/IXF COLUMN DATA TYPE	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRA PH[b]**	**DATE**	**TIME**	**TIME STAMP**
-TIME STAMP												N
												E

주: forcein으로만 PC/IXF 컬럼을 데이터베이스 컬럼에 임포트할 수 있으면 'N' 또는 'E'와 함께 문자열 'w/F'가 표시됩니다. 'N'은 유틸리티가 새 데이터베이스 테이블을 작성함을 나타내고 'E'는 유틸리티가 기존 데이터베이스 테이블로 데이터를 임포트함을 나타냅니다. forcein 파일 유형 수정자는 문자 및 그래픽 데이터 유형의 호환성에만 영향을 줍니다.

[a] 목표 숫자 데이터 유형 범위를 벗어나면 개별 값이 거부됩니다.

[b] 데이터 유형이 DBCS 환경에서만 사용 가능합니다.

[c] 유효한 날짜 또는 시간 값이 아니면 개별 값이 거부됩니다.

[d] 목표 데이터베이스에서 소스 PC/IXF 데이터 유형을 지원하지 않는 경우에만 적용됩니다.

[e] 데이터 유형은 DBCS 환경에서 사용 불가능합니다.

워크시트 파일 형식(WSF)

Lotus 1-2-3 및 Symphony 제품에서는 각 새 릴리스에 추가된 추가 기능과 함께 동일한 기본 형식을 사용합니다. 데이터베이스 관리 프로그램에서는 모든 Lotus 제품에서 동일한 워크시트 레코드의 서브세트를 지원합니다. 즉, 데이터베이스 관리 프로그램에서 지원하는 Lotus 1-2-3 및 Symphony 제품의 릴리스에서 3자리 확장자를 포함한 모든 파일이 허용됩니다(예: WKS, WK1, WRK, WR1, WJ2).

주: 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

각 WSF 파일은 하나의 워크시트를 나타냅니다. 데이터베이스 관리 프로그램에서는 다음 규칙을 사용하여 워크시트를 해석하고 익스포트 조작으로 생성된 워크시트의 일관성을 제공합니다.

- 첫 번째 행의 셀(ROW 값 0)은 전체 워크시트에 대한 설명 정보를 위해 예약됩니다. 이 행의 모든 데이터는 선택적입니다. 임포트 중에는 무시됩니다.
- 두 번째 행의 셀(ROW 값 1)은 컬럼 레이블에 대해 사용됩니다.
- 나머지 행은 데이터 행(테이블의 데이터 행 또는 레코드)입니다.
- 열 표제 아래 셀 값은 특정 컬럼 또는 필드에 대한 값입니다.
- 실제 셀 내용 레코드가 없으면 널(NULL) 값이 표시됩니다. 예를 들어 셀 내용 레코드의 행 내 특정 컬럼에 대한 정수, 숫자, 레이블 또는 공식 레코드가 없는 경우가 이에 해당합니다.

주: 널(NULL) 값인 행은 임포트 또는 익스포트되지 않습니다.

익스포트 조작 중 WSF 형식과 호환되는 파일을 작성하려는 경우 일부 데이터가 유실될 수 있습니다.

WSF 파일은 Lotus 코드 포인트 맵핑을 사용합니다. 이는 DB2 데이터베이스에서 지원되는 기존 코드 페이지와 동일할 필요는 없습니다. 결과적으로 WSF 파일을 임포트 또는 익스포트하는 경우 데이터는 Lotus 코드 포인트 또는 응용프로그램 코드 페이지가 사용하는 코드 포인트에서 변환됩니다. DB2에서는 Lotus 코드 포인트 및 코드 페이지 437, 819, 850, 860, 863 및 865에서 정의된 코드 포인트 간 변환을 지원합니다.

주: 멀티바이트 문자 세트 사용자의 경우 변환은 수행되지 않습니다.

데이터 이동 시 유니코드 고려사항

유니코드가 아닌 데이터베이스에 연결된 유니코드 클라이언트에서 익스포트, 임포트 및 로드 유틸리티가 사용되는 경우 이러한 유틸리티는 지원되지 않습니다.

이 절에서 설명하는 대로, 유니코드 데이터베이스에 대해서는 DEL, ASC 및 PC/IXF 파일 형식이 지원됩니다. WSF 형식은 지원되지 않습니다.

유니코드 데이터베이스에서 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하면 모든 문자 데이터가 응용프로그램 코드 페이지로 변환됩니다. 문자열 및 그래픽 문자열 데이터 모두 클라이언트의 동일한 SBCS 또는 MBCS 코드 페이지로 변환됩니다. 이는 모든 데이터베이스의 익스포트 시 예상되는 동작으로 전체 컬럼 식별자가 있는 ASCII 파일은 하나의 코드 페이지만 포함할 수 있으므로 변경 불가능합니다. 따라서 컬럼 식별자가 있는 ASCII 파일로 익스포트하면 응용프로그램 코드 페이지에 있는 UCS-2 문자만 저장됩니다. 다른 문자는 응용프로그램 코드 페이지의 디폴트 대체 문자로 교체됩니다. UTF-8 클라이언트(코드 페이지 1208)의 경우 모든 UCS-2 문자는 UTF-8 클라이언트에서 지원되므로 데이터는 유실되지 않습니다.

ASCII 파일(DEL 또는 ASC)에서 유니코드 데이터베이스로 임포트하는 경우 문자열 데이터는 응용프로그램 코드 페이지에서 UTF-8로 변환되고 그래픽 문자열 데이터는 응용프로그램 코드 페이지에서 UCS-2로 변환됩니다. 데이터는 유실되지 않습니다. 다른 코드 페이지로 저장된 ASCII 데이터를 임포트하려는 경우 IMPORT 명령을 실행하기 전에 데이터 파일 코드 페이지를 변경해야 합니다. **DB2CODEPAGE** 레지스트리 변수를 ASCII 데이터 파일의 코드 페이지로 설정하거나 `codepage` 파일 유형 수정자를 사용하여 데이터 파일의 코드 페이지를 지정할 수 있습니다.

SBCS 및 MBCS 클라이언트에서 유효한 ASCII 분리문자의 범위는 현재 해당 클라이언트에 대해 IBM DB2 V9.1이 지원하는 범위와 동일합니다. 일반 제한사항을 적용하는 경우 UTF-8의 유효한 분리문자 범위는 X'01'에서 X'7F' 사이입니다.

유니코드 데이터베이스에서 PC/IXF 파일로 익스포트하면 문자열 데이터가 클라이언트의 SBCD/MBCS 코드 페이지로 변환됩니다. 그래픽 문자열 데이터는 변환되지 않으며 UCS-2(코드 페이지 1200)로 저장됩니다. 데이터는 유실되지 않습니다.

PC/IXF 파일에서 유니코드 데이터베이스로 임포트하는 경우 문자열 데이터는 PC/IXF 헤더에 저장된 SBCS/MBCS 코드 페이지에 있다고 가정하고 그래픽 문자열 데이터는 PC/IXF 헤더에 저장된 DBCS 코드 페이지에 있다고 가정합니다. 문자열 데이터는 임포트 유틸리티에 의해 PC/IXF 헤더에 지정된 코드 페이지에서 클라이언트의 코드 페이지로, 그 다음 클라이언트 코드 페이지에서 UTF-8(INSERT문 사용)로 변환됩니다. 그래픽 문자열 데이터는 임포트 유틸리티에 의해 PC/IXF 헤더에 지정된 DBCS 코드 페이지에서 UCS-2(코드 페이지 1200)로 직접 변환됩니다.

로드 유틸리티는 데이터베이스에 직접 데이터를 배치하고 디폴트로 ASC 또는 DEL 파일의 데이터가 데이터베이스의 코드 페이지에 있다고 가정합니다. 따라서 디폴트로 ASCII 파일에서 코드 페이지 변환은 수행되지 않습니다. 데이터 파일에 대한 코드 페이지를 명시적으로 지정한 경우(codepage 수정자 사용) 로드 유틸리티는 이 정보를 사용하여 데이터를 로드하기 전에 지정된 코드 페이지에서 데이터베이스 코드 페이지로 변환합니다. PC/IXF 파일의 경우 로드 유틸리티는 항상 IXF 헤더에 지정된 코드 페이지에서 데이터베이스 코드 페이지로 변환합니다(CHAR의 경우 1208, GRAPHIC의 경우 1200).

DBCLOB 파일의 코드 페이지는 항상 UCS-2의 1200입니다. CLOB 파일의 코드 페이지는 임포트, 로드 또는 익스포트할 데이터 파일의 코드 페이지와 동일합니다. 예를 들어 PC/IXF 형식을 사용하여 데이터를 로드 또는 임포트하는 경우 CLOB 파일은 PC/IXF 헤더에 지정된 코드 페이지에 있다고 가정합니다. DBCLOB 파일이 ASC 또는 DEL 형식인 경우 로드 유틸리티는 CLOB 데이터가 데이터베이스의 코드 페이지에 있다고 가정하지만 임포트 유틸리티는 클라이언트 응용프로그램의 코드 페이지에 있다고 가정합니다.

nochecklengths 수정자는 항상 유니코드 데이터베이스에 대해 지정됩니다. 이유는 다음과 같습니다.

- SBCS는 DBCS 코드 페이지가 없는 데이터베이스에 연결될 수 있습니다.
- 일반적으로 UTF-8 형식의 문자열 길이는 클라이언트 코드 페이지의 길이와 다릅니다.

코드 페이지 1394, 1392 및 5488에 대한 고려사항

임포트, 익스포트 및 로드 유틸리티를 사용하여 중국어 코드 페이지 GB 18030(코드 페이지 ID 1392 및 5488) 및 일본어 코드 페이지 ShiftJISX 0213(코드 페이지 ID 1394)

에서 DB2 유니코드 데이터베이스로 데이터를 전송할 수 있습니다. 또한 익스포트 유틸리티를 사용하여 DB2 유니코드 데이터베이스에서 GB 18030 또는 ShiftJIS X0213 코드 페이지 데이터로 데이터를 전송할 수 있습니다.

예를 들어 다음 명령은 리모트로 연결된 클라이언트에 있는 Shift_JISX0213 데이터 파일 u/jp/user/x0213/data.del을 MYTABLE로 로드합니다.

```
db2 load client from /u/jp/user/x0213/data.del
of del modified by codepage=1394 insert into mytable
```

여기서 MYTABLE은 DB2 유니코드 데이터베이스에 있습니다.

유니코드 클라이언트 및 유니코드 서버 간 연결만 지원되므로 로드, 임포트 또는 익스포트 유틸리티를 사용하기 전에 유니코드 클라이언트를 사용하거나 DB2 레지스트리 변수를 **DB2CODEPAGE**로 설정해야 합니다.

코드 페이지 1394, 1392 또는 5488에서 유니코드로 변환하면 데이터가 확장됩니다. 예를 들어 2바이트 문자는 GRAPHIC 컬럼에서 2개의 16비트 유니코드 문자로 저장될 수 있습니다. 유니코드 데이터베이스의 목표 컬럼 너비가 확장된 유니코드 바이트를 포함하도록 커야 합니다.

비호환성

유니코드 데이터베이스에 연결된 응용프로그램의 경우 그래픽 문자열 데이터는 항상 UCS-2(코드 페이지 1200)입니다. 유니코드가 아닌 데이터베이스에 연결된 응용프로그램의 경우 그래픽 문자열 데이터는 응용프로그램의 DBCS 코드 페이지에 있거나, 응용프로그램 코드 페이지가 SBCS인 경우 허용되지 않습니다. 예를 들어 932 클라이언트가 유니코드가 아닌 일본어 데이터베이스에 연결된 경우 그래픽 문자열 데이터는 코드 페이지 301 양식입니다. 유니코드 데이터베이스에 연결된 932 클라이언트 응용프로그램의 경우 그래픽 문자열 데이터는 UCS-2 인코딩입니다.

문자 세트 및 자국어 지원(NLS)

DB2 데이터 이동 유틸리티에서는 다음 자국어 지원(NLS)을 제공합니다.

- 임포트 및 익스포트 유틸리티에서는 클라이언트 코드 페이지에서 서버 코드 페이지로의 자동 코드 페이지 변환을 제공합니다.
- 로드 유틸리티의 경우 DEL 및 ASX 파일에서 codepage 수정자를 사용하여 모든 코드 페이지에서 서버 코드 페이지로 데이터를 변환할 수 있습니다.
- 모든 유틸리티에서 IXF 데이터는 원래 코드 페이지(IXF 파일로 저장됨)에서 서버 코드 페이지로 자동 변환됩니다.

문자 데이터의 축약 또는 확장과 관련하여 동일하지 않은 코드 페이지가 때때로 나타날 수 있습니다. 예를 들어 일본어 또는 대만어 EUC(Extended UNIX Code) 및 2바이트 문자 세트(DBCS)는 동일한 문자를 다른 길이로 인코드할 수 있습니다. 정상적으

로 입력 데이터 길이와 목표 컬럼 길이 비교는 데이터를 읽기 전에 수행됩니다. 입력 길이가 목표 길이보다 크면 널(NULL) 입력 가능한 경우 해당 컬럼에 널(NULL) 값이 입력됩니다. 그렇지 않으면 요청이 거부됩니다. nochecklengths 파일 유형 수정자가 지정되면 초기 비교는 수행되지 않고 데이터를 임포트 또는 로드하려고 합니다. 변환 완료 후 데이터가 너무 길면 행이 거부됩니다. 그렇지 않으면 데이터가 임포트 또는 로드됩니다.

XML 데이터 이동

로드, 임포트 및 익스포트 유틸리티를 통해 XML 데이터 이동이 지원됩니다. 테이블을 오프라인으로 설정하지 않고도 XML 컬럼이 포함된 테이블을 이동하는 작업은 ADMIN_MOVE_TABLE 스토어드 프로시저를 통해 지원됩니다.

XML 데이터 임포트

임포트 유틸리티를 사용하여 XML 문서를 일반 관계형 테이블에 삽입할 수 있습니다. 잘 구성된 XML 문서만 임포트할 수 있습니다.

임포트할 XML 문서의 위치를 지정하려면 IMPORT 명령의 XML FROM 옵션을 사용하십시오. XMLVALIDATE 옵션은 임포트된 문서의 유효성 확인 방법을 지정합니다. IMPORT 명령을 사용하여 지정한 스키마에 대해, 소스 XML 문서 내부의 스키마 위치 힌트를 통해 식별된 스키마에 대해, 또는 주 데이터 파일에서 XML 데이터 지정자가 식별한 스키마를 통해 임포트된 XML 데이터의 유효성을 확인하도록 선택할 수 있습니다. XMLPARSE 옵션을 사용하여 XML 문서 임포트 시 공백을 처리하는 방법을 지정할 수도 있습니다. xmlchar 및 xmlgraphic 파일 유형 수정자를 사용하면 임포트된 XML 데이터의 인코딩 특성을 지정할 수 있습니다.

XML 데이터 로드

로드 유틸리티는 볼륨이 큰 XML 데이터를 테이블에 삽입할 수 있는 효율적인 방법을 제공합니다. 이 유틸리티를 사용하면 사용자 정의 커서에서 로드하는 기능과 같이 임포트 유틸리티에서 사용 불가능한 특정 옵션도 사용할 수 있습니다.

IMPORT 명령과 마찬가지로, LOAD 명령을 사용하여 로드할 XML 데이터 위치, XML 데이터의 유효성 확인 옵션 및 공백 처리 방법을 지정할 수 있습니다. IMPORT의 경우 xmlchar 및 xmlgraphic 파일 유형 수정자를 사용하여 로드된 XML 데이터의 인코딩 특성을 지정할 수 있습니다.

XML 데이터 익스포트

XML 데이터 유형이 있는 컬럼을 하나 이상 포함하는 테이블에서 데이터를 익스포트할 수 있습니다. 익스포트된 XML 데이터는 익스포트된 관계형 데이터를 포함하는 주 데이터 파일과는 별개의 파일에 저장됩니다. 익스포트된 각 XML 문서에 대한 정보는

익스포트된 주 데이터 파일에 XML 데이터 지정자(XDS)로 표시됩니다. XDS는 XML 문서가 저장된 시스템 파일의 이름, 이 파일 내부에 있는 XML 문서의 정확인 위치와 길이 및 XML 문서 유효성 확인에 사용되는 XML 스키마를 지정하는 문자열입니다.

EXPORT 명령의 XMLFILE, XML TO 및 XMLSAVESCHEMA 매개변수를 사용하여 익스포트된 XML 문서를 저장하는 방법에 대한 세부사항을 지정할 수 있습니다. xmlinsepfiles, xmlnodeclaration, xmlchar 및 xmlgraphic 파일 유형 수정자를 사용하면 익스포트된 XML 데이터의 인코딩 및 스토리지 위치에 대한 세부사항을 지정할 수 있습니다.

온라인 테이블 이동

ADMIN_MOVE_TABLE 스토어드 프로시저는 데이터가 온라인으로 남아있어서 액세스할 수 있는 동안에 활성 테이블의 데이터를 이름이 같은 새 테이블 오브젝트로 이동합니다. 테이블에는 XML 데이터 유형이 있는 컬럼이 하나 이상 포함될 수 있습니다. 비용, 스페이스, 이동 성능 및 트랜잭션 오버헤드보다 가용성이 중요한 경우에는 오프라인 테이블 이동 대신 온라인 테이블 이동을 사용하십시오.

프로시저에서 수행하는 연산마다 한 번씩 호출하여 프로시저를 한 번 또는 여러 번 호출할 수 있습니다. 여러 호출을 사용할 때 이동 취소 또는 목표 테이블을 갱신하기 위해 오프라인 상태로 전환하는 시기 제어와 같은 추가 옵션이 제공됩니다.

XML 데이터 이동 시 중요한 고려사항

XML 데이터 임포트 또는 익스포트 시 고려해야 할 여러 제한사항, 전제조건 및 유의사항이 있습니다. XML 데이터를 임포트하거나 익스포트하기 전에 이러한 고려사항을 검토하십시오.

XML 데이터 익스포트 또는 임포트 시 다음 고려사항을 유념하십시오.

- 익스포트된 XML 데이터는 항상 익스포트된 관계형 데이터를 포함하는 주 데이터 파일과 다른 파일에 저장됩니다.
- 디폴트로 익스포트 유틸리티는 XML 데이터를 유니코드로 작성합니다. xmlchar 파일 유형 수정자를 사용하여 XML 데이터를 문자 코드 페이지로 작성하거나 xmlgraphic 파일 유형 수정자를 사용하여 응용프로그램 코드에 관계 없이 XML 데이터를 UTF-16(그래픽 코드 페이지)으로 작성하십시오.
- XML 데이터를 유니코드가 아닌 데이터베이스에 저장할 수 있으며 XML 컬럼에 삽입되는 데이터를 삽입 이전에 데이터베이스 코드 페이지에서 UTF-8로 변환할 수 있습니다. XML 구문 분석 중에 대체 문자가 사용되지 않도록 하려면 삽입할 문자 데이터가 데이터베이스 코드 페이지의 파트인 코드 포인트로만 구성되어야 합니다. enable_xmlchar 구성 매개변수를 no로 설정하면 XML 구문 분석 중에 문자 데이터 유형 삽입이 차단되어 BIT DATA, BLOB 또는 XML과 같이 코드 페이지 변환을 수행하지 않는 데이터 유형에 삽입하는 것이 제한됩니다.

- XML 데이터 임포트 또는 로드 시, 임포트할 XML 문서에 인코드 속성이 포함된 선언 태그가 들어 있지 않은 경우에는 XML 데이터가 유니코드 형식인 것으로 가정합니다. `xmlchar` 파일 유형 수정자를 사용하여 임포트할 XML 문서가 문자 코드 페이지로 인코드됨을 나타낼 수 있으며, `xmlgraphic` 파일 유형 수정자를 사용하여 임포트할 XML 문서가 UTF-16으로 인코드됨을 나타낼 수 있습니다.
- 임포트 및 로드 유틸리티는 잘 구성되지 않은 문서를 포함하는 행을 거부합니다.
- 임포트 또는 로드 유틸리티에 XMLVALIDATE 옵션이 지정된 경우, 일치하는 스키마와 대조하여 정상적으로 유효성이 확인된 문서에는 문서를 테이블에 삽입할 때 유효성을 검증하는 데 사용되는 스키마에 대한 정보가 주석으로 포함됩니다. 일치하는 스키마와 대조하여 유효성을 확인하는 데 실패한 문서를 포함하는 행은 거부됩니다.
- 임포트 또는 로드 유틸리티에 XMLVALIDATE 옵션이 지정되고 다중 XML 스키마를 사용하여 XML 문서 유효성을 확인하는 경우에는 카탈로그 캐시 크기 구성 매개변수 **catalogcache_sz**를 늘려야 합니다. **catalogcache_sz** 값을 늘리는 것이 타당하지 않거나 가능하지 않은 경우 단일 임포트 또는 로드 명령을 더 적은 수의 스키마 문서를 사용하는 다중 명령으로 분리할 수 있습니다.
- XQuery 명령문을 지정하여 XML 데이터를 익스포트하면 잘 구성되지 않은 XML 문서인 쿼리 및 XPath 데이터 모델(XDM) 인스턴스를 익스포트할 수 있습니다. XMl 데이터 유형으로 정의된 컬럼에는 완료된, 잘 구성된 XML 문서만 포함시킬 수 있기 때문에 제대로 구성되지 않은 익스포트된 XML 문서를 바로 XML 컬럼에 임포트할 수 없습니다.
- 로드하는 동안 **CPU_PARALLELISM** 설정은 통계를 수집 중인 경우 1로 감소됩니다.
- XML 로드 조작을 진행하려면 공유 정렬 메모리를 사용해야 합니다. **SHEAPTHRES_SHR** 또는 **INTRA_PARALLEL**을 사용하거나 연결 집중기(connection concentrator)를 설정하십시오. 디폴트로 **SHEAPTHRES_SHR**이 설정되어 공유 정렬 메모리를 디폴트 구성에서 사용할 수 있습니다.
- XML 문서가 포함된 테이블 로드 시에는 LOAD 명령의 **SOURCEUSEREXIT** 옵션 또는 **SAVECOUNT** 매개변수를 지정할 수 없습니다.
- LOB 파일의 경우, LOAD 명령 사용 시 XML 파일이 서버 측에 상주해야 합니다.
- 파티션된 데이터베이스 환경에서 다중 데이터베이스 파티션에 XML 데이터를 로드하는 경우 모든 데이터베이스 파티션에서 XML 데이터가 포함된 파일에 액세스할 수 있어야 합니다. 예를 들어, 파일을 복사하거나 NFS 마운트를 작성하여 파일에 액세스할 수 있도록 만들 수 있습니다.

임포트 및 익스포트 시 LOB 및 XML 파일 동작

LOB와 XML 파일은 데이터 임포트 및 익스포트 시 사용할 수 있는 특정 동작과 호환성을 공유합니다.

익스포트

데이터 익스포트 시 LOB TO 옵션에 하나 이상의 LOB 경로가 지정된 경우 익스포트 유틸리티는 적절한 LOB 파일에 각 LOB 값을 차례로 작성하도록 여러 경로 사이를 순환합니다. 이와 비슷하게 XML TO 옵션에 하나 이상의 XML 경로가 지정된 경우 익스포트 유틸리티는 적절한 XML 파일에 각 XQuery 및 XPath 데이터 모델(XDM) 인스턴스를 차례로 작성하도록 여러 경로 사이를 순환합니다. 디폴트로, LOB 값 및 XDM 인스턴스는 익스포트된 관계형 데이터가 작성된 경로에 작성됩니다. LOBSINSEPFILES 또는 XMLINSEPFILES 파일 유형 수정자를 설정하지 않은 경우에는 LOB 파일 및 XML 파일 모두 동일한 파일에 병합된 다중 값을 갖습니다.

LOBFILE 옵션을 사용하면 익스포트 유틸리티에서 생성한 LOB 파일의 기본 이름을 지정할 수 있습니다. 마찬가지로 XMLFILE 옵션을 사용하면 익스포트 유틸리티에서 생성한 XML 파일의 기본 이름을 지정할 수 있습니다. 디폴트 LOB 파일 기본 이름은 익스포트된 데이터 파일의 이름으로 확장자는 .lob입니다. 디폴트 XML 파일 기본 이름은 익스포트된 데이터 파일의 이름으로 확장자는 .xml입니다. 따라서 익스포트된 LOB 파일 또는 XML 파일의 전체 이름은 기본 이름과 그 뒤에 표시되는 3자리까지 채워진 숫자 확장자 및 확장자 .lob 또는 .xml로 구성됩니다.

임포트 데이터 임포트 시 LOB 위치 지정자(LLS)는 XML 목표 컬럼과 호환 가능하고 XML 데이터 지정자(XDS)는 LOB 목표 컬럼과 호환 가능합니다. LOBS FROM 옵션을 지정하지 않으면 임포트할 LOB 파일이 입력 관계형 데이터 파일과 동일한 경로에 상주한다고 가정합니다. 마찬가지로 XML FROM 옵션을 지정하지 않으면 임포트할 XML 파일이 입력 관계형 데이터 파일과 동일한 경로에 상주한다고 가정합니다.

익스포트 예

다음 예에서 모든 LOB 값은 /mypath/t1export.del.001.lob 파일에 저장되고 모든 XDM 인스턴스는 /mypath/t1export.del.001.xml 파일에 저장됩니다.

```
EXPORT TO /mypath/t1export.del OF DEL MODIFIED BY LOBSINFILE
   SELECT * FROM USER.T1
```

다음 예에서 첫 번째 LOB 값은 /lob1/t1export.del.001.lob 파일에 저장되고 두 번째는 /lob2/t1export.del.002.lob 파일에 저장되며 세 번째는 /lob1/t1export.del.001.lob에 추가되고 네 번째는 /lob2/t1export.del.002.lob에 추가됩니다.

```
EXPORT TO /mypath/t1export.del OF DEL LOBS TO /lob1,/lob2
   MODIFIED BY LOBSINFILE SELECT * FROM USER.T1
```

다음 예에서 첫 번째 XDM 인스턴스는 /xml1/xmlbase.001.xml 파일에 저장되고 두 번째는 /xml2/xmlbase.002.xml 파일에 저장되며 세 번째는 /xml1/xmlbase.003.xml에 저장되고 네 번째는 /xml2/xmlbase.004.xml에 저장됩니다.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /xml1,/xml2 XMLFILE xmlbase
     MODIFIED BY XMLINSEPFILES SELECT * FROM USER.T1
```

임포트 예

단일 XML 컬럼 및 다음과 같은 IMPORT 명령이 포함된 ″mytable″ 테이블에서,

```
IMPORT FROM myfile.del of del LOBS FROM /lobpath XML FROM /xmlpath
MODIFIED BY LOBSINFILE XMLCHAR replace into mytable
```

″myfile.del″에 다음 데이터가 포함되어 있는 경우,

```
mylobfile.001.lob.123.456/
```

임포트 유틸리티는 /lobpath/mylobfile.001.lob 파일에서 XML 문서를 임포트하려 하며 파일 오프셋 123에서 시작하고 길이는 456바이트입니다.

XML 데이터 지정자(XDS)가 아닌 LOB 위치 지정자(LLS)가 값을 참조하므로 ″mylobfile.001.lob″ 파일은 XML 경로와 반대되는 LOB 경로에 있는 것으로 가정합니다.

XMLCHAR 파일 유형 수정자가 지정되었으므로 문서는 문자 코드 페이지에서 인코드되는 것으로 가정합니다.

XML 데이터 지정자

익스포트, 임포트 및 로드 유틸리티를 사용하여 이동한 XML 데이터는 주 데이터 파일과 다른 파일에 저장되어야 합니다. XML 데이터는 주 데이터 파일에 XML 데이터 지정자(XDS)로 표시됩니다.

XDS는 ″XDS″라는 XML 태그로 표시되는 문자열입니다. 이 태그에는 컬럼의 실제 XML 데이터에 대한 정보를 설명하는 속성이 있습니다. 이러한 정보에는 실제 XML 데이터가 포함된 파일 이름 및 해당 파일에 있는 XML 데이터의 오프셋 및 길이가 포함됩니다. XDS의 속성에 대해 아래에서 설명합니다.

FIL XML 데이터가 들어 있는 파일의 이름. Named Pipe를 지정할 수 없습니다. Named Pipe에서 XML 문서를 임포트 또는 로드하는 것은 지원되지 않습니다.

OFF FIL 속성으로 이름 지정된 파일에 있는 XML 데이터의 바이트 오프셋입니다. 여기서 오프셋은 0부터 시작합니다.

LEN FIL 속성으로 이름 지정된 파일에 있는 XML 데이터의 길이(바이트)입니다.

SCH 해당 XML 문서의 유효성 확인에 사용되는 XML 스키마의 완전한 SQL ID

입니다. SQL ID의 스키마 및 이름 구성요소는 각각 이 XML 스키마에 해당하는 SYSCAT.XSROBJECTS 카탈로그 테이블 행의 ″OBJECTSCHEMA″ 및 ″OBJECTNAME″ 값으로 저장됩니다.

XDS는 데이터 파일에서 문자 필드로 해석되며 파일 형식의 문자 컬럼에 대한 구문 분석 동작이 수행됩니다. 예를 들어, 컬럼 식별자가 있는 ASCII 파일 형식(DEL)에 문자 분리문자가 XDS에 있으면 이중으로 표시해야 합니다. 속성 값에 포함된 특수 문자(<, >, &, ', ″)는 항상 이스케이프되어야 합니다. 대소문자를 구분하는 오브젝트 이름은 " 문자 엔티티 사이에 두어야 합니다.

예

값이 abc&"def".del인 FIL 속성을 검토합니다. 컬럼 식별자가 있는 ASCII 파일(문자 분리문자가 ″ 문자)에 이 XDS를 포함시키기 위해 ″ 문자는 이중으로 사용되고 특수 문자는 이스케이프됩니다.

```
<XDS FIL=""abc&amp;&quot;def&quot;.del"" />
```

다음 예는 컬럼 식별자가 있는 ASCII 데이터 파일에 표시되는 XDS를 나타냅니다. XML 데이터는 xmldocs.xml.001 파일에 저장되며 바이트 오프셋 100에서 시작하고 길이는 300바이트입니다. 이 XDS가 큰따옴표로 구분되는 ASCII 파일에 있기 때문에 XDS 태그 자체의 큰따옴표는 이중으로 표시해야 합니다.

```
"<XDS FIL = ""xmldocs.xml.001"" OFF=""100"" LEN=""300"" />"
```

다음 예는 완전한 SQL ID ANTHONY.purchaseOrderTest를 표시합니다. ID에서 대소문자를 구분하는 부분은 XDS의 " 문자 엔티티 사이에 두어야 합니다.

```
"<XDS FIL='/home/db2inst1/xmlload/a.xml' OFF='0' LEN='6758'
   SCH='ANTHONY.&quot;purchaseOrderTest&quot;' />"
```

쿼리 및 XPath 데이터 모델

SQL에서 사용 가능한 XQuery 함수를 사용하거나 XQuery를 직접 호출하여 데이터베이스 테이블에서 XML 데이터에 액세스할 수 있습니다. 쿼리 및 XPath 데이터 모델(XDM)의 인스턴스는 잘 구성된 XML 문서, 노드 시퀀스, 원자 값 시퀀스 또는 노드와 원자값의 조합 중 하나입니다.

EXPORT 명령을 사용하여 개별 XDM 인스턴스를 하나 이상의 XML 파일에 기록할 수 있습니다.

제 2 부 부록

부록 A. 임포트 및 로드 유틸리티 간 다른 점

다음 표는 DB2 로드 및 임포트 유틸리티 간 중요한 다른 점에 대해 요약하여 설명합니다.

임포트 유틸리티	로드 유틸리티
많은 양의 데이터 이동 시 느려집니다.	많은 데이터를 이동하는 경우 임포트 유틸리티보다 빠릅니다. 로드 유틸리티는 데이터베이스에 직접 형식화된 페이지를 작성하기 때문입니다.
파티션 내 병렬 처리 사용을 제한합니다. 파티션 내 병렬 처리는 ALLOW WRITE ACCESS 모드에서 임포트 유틸리티를 동시에 호출한 경우에만 수행할 수 있습니다.	파티션 내 병렬 처리를 사용합니다. 일반적으로 SMP(Symmetric Multi-Processor) 머신이 필요합니다.
FASTPARSE가 지원되지 않습니다.	FASTPARSE가 지원됩니다. 사용자 제공 데이터에 대한 데이터 검사가 감소합니다.
계층 데이터를 지원합니다.	계층 데이터를 지원하지 않습니다.
PC/IXF 형식으로 지원되는 테이블, 계층 구조 및 인덱스를 작성합니다.	테이블 및 인덱스가 있어야 합니다.
구체화된 쿼리 테이블로의 임포트가 지원되지 않습니다.	구체화된 쿼리 테이블로의 로드가 지원됩니다.
WSF 형식이 지원됩니다.	WSF 형식이 지원되지 않습니다.
BINARYNUMERICS가 지원되지 않습니다.	BINARYNUMERICS가 지원됩니다.
PACKEDDECIMAL이 지원되지 않습니다.	PACKEDDECIMAL이 지원됩니다.
ZONEDDECIMAL이 지원되지 않습니다.	ZONEDDECIMAL이 지원됩니다.
GENERATED ALWAYS로 정의된 컬럼을 겹쳐 쓸 수 없습니다.	`generatedoverride` 및 `identityoverride` 파일 유형 수정자를 사용하여 GENERATED ALWAYS로 정의된 컬럼을 겹쳐쓸 수 있습니다.
테이블, 뷰 및 별칭으로의 임포트를 지원합니다.	테이블로의 로드만 지원합니다.
모든 행이 로그됩니다.	최소 로깅이 수행됩니다.
트리거를 지원합니다.	트리거를 지원하지 않습니다.
임포트 조작이 인터럽트되고 *commitcount*가 지정되면 테이블은 사용 가능하며 마지막 COMMIT까지 로드된 행을 포함합니다. 사용자는 임포트 조작을 재시작하거나 테이블을 그대로 승인할 수 있습니다.	로드 조작이 인터럽트되고 *savecount*가 지정된 경우 테이블은 로드 보류 상태로 남아 있으며 로드 조작을 다시 시작하거나 로드 종료 조작이 호출되거나 로드 조작을 시도하기 전에 작성된 백업 이미지에서 테이블 스페이스를 리스토어할 때까지 사용할 수 없습니다.
필요한 스페이스는 대략적으로 가장 큰 인덱스 크기의 110%입니다. 이 스페이스는 데이터베이스 내 임시 테이블 스페이스에서 가져옵니다.	필요한 스페이스는 대략적으로 테이블에 정의된 모든 인덱스 크기의 합과 동일하며 최대 이 크기의 2배까지 가능합니다. 이 스페이스는 데이터베이스 내 임시 스페이스에서 가져옵니다.
모든 제한조건은 임포트 조작 중 유효성이 확인됩니다.	로드 유틸리티는 고유성을 검사하고 생성된 컬럼 값을 계산하지만 SET INTEGRITY를 사용하여 다른 모든 제한조건을 검사해야 합니다.

임포트 유틸리티	로드 유틸리티
임포트 조작 중 키 값이 한 번에 하나의 인덱스로 삽입됩니다.	데이터를 로드한 후 키 값이 정렬되고 인덱스가 빌드됩니다.
갱신된 통계가 필요한 경우 임포트 조작 후 runstats 유틸리티를 실행해야 합니다.	테이블의 모든 데이터를 교체하는 경우 로드 조작 중 통계를 수집할 수 있습니다.
DB2 Connect를 통해 호스트 데이터베이스로 임포트할 수 있습니다.	호스트 데이터베이스로 로드할 수 없습니다.
임포트 파일은 임포트 유틸리티가 호출되는 클라이언트에 있어야 합니다.	지정한 옵션에 따라 로드 파일 또는 파이프는 로드 유틸리티를 호출한 리모트로 연결된 클라이언트 또는 데이터베이스를 포함하는 데이터베이스 파티션에 있을 수 있습니다. **주:** LOB 및 XML 데이터는 서버 측에서 읽을 수만 있습니다.
백업 이미지는 필요하지 않습니다. 임포트 유틸리티는 SQL 삽입을 사용하므로 활동은 로그되고 실패 시 이러한 조작을 복구하기 위해 백업은 필요하지 않습니다.	로드 조작 중 백업 이미지를 작성할 수 있습니다.

부록 B. 익스포트, 임포트 및 로드 유틸리티에서 사용하는 바인드 파일

다음 표는 디폴트 분리 레벨을 포함하는 바인드 파일과 함께 바인드 파일을 사용할 유틸리티 및 해당 용도를 표시합니다.

바인드 파일(디폴트 분리 레벨)	유틸리티/용도
db2ueiwi.bnd(CS)	임포트/익스포트. 테이블 컬럼 및 인덱스에 대한 정보를 쿼리하는 데 사용됩니다.
db2uexpm.bnd(CS)	익스포트. 익스포트 조작에서 지정된 쿼리에서 페치하는 데 사용됩니다.
db2uimpm.bnd(RS)	임포트. INSERT, REPLACE 또는 REPLACE_CREATE 옵션을 사용하는 경우 소스 데이터 파일에서 목표 테이블로 데이터를 삽입하는 데 사용됩니다. **주:** 참고: IMPORT 명령의 CREATE 및 REPLACE_CREATE 옵션은 사용되지 않으며 추후 릴리스에서 제거됩니다.
db2uipkg.bnd(CS)	임포트. 바인드 옵션을 확인하는 데 사용됩니다.
db2ucktb.bnd(CS)	로드. 로드 조작에 대한 일반 초기화 프로세스를 수행하는 데 사용됩니다.
db2ulxld.bnd(CS)	로드. 커서 조작에서 로드 중 제공되는 쿼리를 처리하는 데 사용됩니다.
db2uigsi.bnd(UNIX 기반 시스템의 경우 RS, 나머지 다른 플랫폼의 경우 RR)	임포트/익스포트. 임포트 교체 조작에 대한 참조 제한조건을 검사하고 인덱스를 삭제하는 데 사용됩니다. IXF 파일을 익스포트하는 경우 ID 컬럼 정보를 검색하는 데 사용됩니다.
db2uqtpd.bnd(RR)	임포트/익스포트. 계층 테이블에 대한 처리를 수행하는 데 사용됩니다.
db2uimtb.bnd(RS)	임포트. 임포트 조작에 대한 일반 초기화 프로세스를 수행하는 데 사용됩니다.
db2uImpInsUpdate.bnd(RS)	임포트. INSERT_UPDATE 옵션을 사용하는 경우 소스 데이터 파일에서 목표 테이블로 데이터를 삽입하는 데 사용됩니다. INSERT BUF 옵션과 함께 바인드될 수 없습니다.

부록 C. 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.

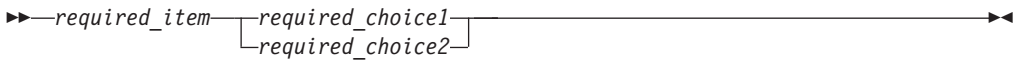
필수 항목은 수평선(주 경로)에 나타납니다.

선택 항목은 기본 경로 아래에 나타납니다.

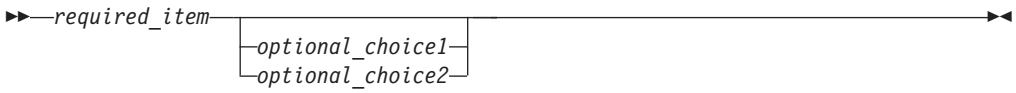
선택적 항목이 주 경로 위에 나타나는 경우 해당 항목은 실행 시 적용되지 않으며 판독성을 위해서만 사용됩니다.

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

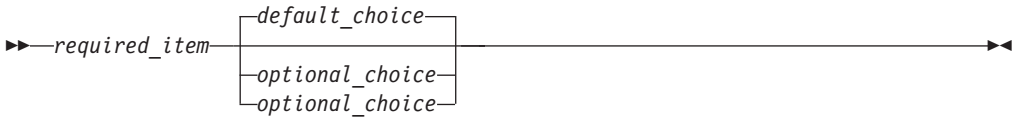
항목 중 하나를 *반드시* 선택해야 하는 경우 스택 중 하나의 항목이 주 경로에 나타납니다.



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
```

부록 D. 데이터 이동 문제점에 대한 데이터 수집

데이터 이동 명령을 수행하는 동안 문제점이 발생하고 문제점의 원인을 판별할 수 없는 경우, 직접 또는 IBM Software Support에서 문제점을 진단 및 해결하는 데 사용할 수 있는 진단 데이터를 수집하십시오.

발생하는 상황에 맞게 다음 목록의 데이터 콜렉션 지시사항을 따르십시오.

- db2move 명령과 관련된 문제점에 대한 데이터를 수집하려면, 명령을 실행한 디렉토리로 이동하십시오. 명령에 지정한 조치에 따라 다음 파일을 찾으십시오.
  - COPY 조치의 경우, COPY.*timestamp*.ERR 및 COPYSCHEMA.*timestamp*.MSG 파일을 찾으십시오. LOAD_ONLY 또는 DDL_AND_LOAD 모드도 지정한 경우, LOADTABLE.*timestamp*.MSG 파일도 찾으십시오.
  - EXPORT 조치의 경우, EXPORT.out 파일을 찾으십시오.
  - IMPORT 조치의 경우, IMPORT.out 파일을 찾으십시오.
  - LOAD 조치의 경우, LOAD.out 파일을 찾으십시오.
- EXPORT, IMPORT 또는 LOAD 명령과 관련된 문제점에 대한 데이터를 수집하려면, 명령에 MESSAGES 매개변수가 포함되었는지 여부를 판별하십시오. 포함되었다면, 출력 파일을 수집하십시오. 별도로 지정하지 않으면 이러한 유틸리티는 현재 디렉토리 및 디폴트 드라이브를 대상으로 사용합니다.
- REDISTRIBUTE 명령과 관련된 문제점에 대한 데이터를 수집하려면, *"databasename.database_partition_groupname.timestamp"*(Linux 및 UNIX의 경우) 및 *"databasename.database_partition_groupname.date.time"*(Windows의 경우) 파일을 찾으십시오. 이 파일은 각각 *$HOME*/sqllib/db2dump 디렉토리 또는 $DB2PATH¥sqllib¥redist(여기서 *$HOME*은 인스턴스 소유자이 홈 디렉토리임)에 있습니다.

부록 E. 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 센터 주제는 PDF 또는 하드카피 서적보다 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. DB2 정보 관리 라이브러리 소프트웨어 사이트 http://www.ibm.com/software/data/sw-library/에 액세스하십시오.

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 위치에 특정 예를 제공해주십시오. 특정 주제 또는 도움말 파일에 대한 피드백을 보내실 경우, 제목 및 URL을 알려주십시오.

DB2 고객 지원에 문의할 때 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 대해 설명합니다. 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 책이 나와 있는 경우에도, 사용 국가 또는 지역에서 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 62. 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 62. 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 62. 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 63. 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 64. 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 지역에 해당되지는 않습니다. 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. *DB2 PDF 문서* DVD의 일부 소프트카피 서적은 인쇄할 수 없다는 점에 유의하십시오. 예를 들면, *DB2 메시지 참조서*의 어떤 볼륨도 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 최신 및 완료된 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 정보에 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 월드 와이드 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 530 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 영어로 주제 항목을 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 추가했다고 컴퓨터에 원하는 언어로 항목을 표시하는 데 필요한 글꼴이 설치되는 것은 아닙니다.

– 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.

3. 브라우저 캐시를 지운 후 페이지를 새로 고쳐 원하는 언어로 DB2 정보 센터를 표시하십시오.

- Firefox 또는 Mozilla 브라우저에서 원하는 언어로 주제 항목을 표시하려면 다음을 수행하십시오.
  1. **도구 —> 옵션 —> 고급** 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 **추가...** 단추를 눌러 언어 추가 창에서 언어를 선택합니다.
     – 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
  3. 브라우저 캐시를 지운 후 페이지를 새로 고쳐 원하는 언어로 DB2 정보 센터를 표시하십시오.

일부 브라우저 및 운영 체제 조합에서 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 등록된 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 아주 잠시 동안만 사용 불가능하다는 점입니다. 또한 자동 갱신은 주기적으로 실행되는 기타 일괄처리 작업의 일부로 실행되도록 설정 가능합니다.
- 수동 갱신 - 갱신 프로세스 중에 기능이나 언어를 추가하려는 경우 사용하십시오. 예를 들어, 로컬 정보 센터는 기본적으로 영어와 프랑스어로 설치되어 있으며 기존 정보 센터의 기능 및 언어 갱신 외에도 수동 갱신으로 독어도 설치할 수 있습니다. 단,

수동 갱신을 수행하려면 정보 센터를 중지한 다음 갱신하고 재시작해야 합니다. 정보 센터는 갱신 프로세스 동안에는 사용할 수 없습니다.

프로시저

이 주제는 자동 갱신 프로세스에 대한 설명입니다. 수동 갱신에 대한 지시사항은 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신』 주제를 참조하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 자동으로 갱신하려면 다음을 수행하십시오.

1. Linux 운영 체제에서,
   a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.
   b. 설치 디렉토리에서 `doc/bin` 디렉토리까지 탐색하십시오.
   c. 다음과 같이 `ic-update` 스크립트를 실행하십시오.

      ```
      ic-update
      ```
2. Windows 운영 체제에서,
   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` 디렉토리까지 탐색하십시오.
   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. 어떤 갱신사항이 사용 가능한지 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 연결된 머신을 사용하여 갱신 사이트를 로컬 파일 시스템에 미러해야 DB2 정보 센터가 설치됩니다. 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 섹션 다음에 나오는 명령은 관리자로 실행해야 합니다. 전체 관리자 권한으로 명령 프롬프트 또는 그래픽 도구를 열려면 단축키를 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.
   - Windows에서는 **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 정보 센터** 서비스를 마우스 오른쪽 단추로 누른 후 **중지**를 선택하십시오.
   - Linux에서는 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 독립형 모드에서 정보 센터를 시작하십시오.
   - Windows 사용자:
     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 디렉토리까지 탐색하십시오.
     d. 다음과 같이 help_start.bat 파일을 실행하십시오.

        ```
        help_start.bat
        ```
   - Linux 사용자:

a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 /opt/ibm/db2ic/V9.7 디렉토리에 디폴트로 설치됩니다.
b. 설치 디렉토리에서 doc/bin 디렉토리까지 탐색하십시오.
c. 다음과 같이 help_start 스크립트를 실행하십시오.

```
help_start
```

독립형 정보 센터를 표시하기 위해 시스템의 기본 웹 브라우저가 열립니다.

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.
   - Windows에서 설치 디렉토리의 doc₩bin 디렉토리를 탐색한 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 중지하는 데 Ctrl-C 또는 다른 메소드를 사용하지 마십시오.
   - Linux에서 설치 디렉토리의 doc/bin 디렉토리를 탐색한 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 중지하는 데 다른 메소드를 사용하지 마십시오.
7. DB2 정보 센터를 재시작하십시오.
   - Windows에서는 **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 정보 센터** 서비스를 마우스 오른쪽 단추로 누른 후 **시작**을 선택하십시오.
   - Linux에서는 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 start
     ```

갱신된 DB2 정보 센터에는 새로 갱신된 주제가 표시됩니다.

DB2 자습서

DB2 자습서는 DB2 제품의 다양한 측면에 대해 학습하는 데 유용합니다. 각 레슨은 단계별 지시사항을 제공합니다.

시작하기 전에

http://publib.boulder.ibm.com/infocenter/db2help/의 정보 센터에서 XHTML 버전의 자습서를 볼 수 있습니다.

일부 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크에 대한 전제조건 설명은 자습서를 참조하십시오.

DB2 자습서

자습서를 보려면 제목을 누르십시오.

『**pureXML®**』(*pureXML Guide*)
XML 데이터를 저장하고 원시 XML 데이터 스토어로 기본 조작을 수행하려면 DB2 데이터베이스를 설정하십시오.

*Visual Explain 자습서*의 『**Visual Explain**』
Visual Explain을 사용하여 성능을 향상시킬 수 있도록 SQL문을 분석, 최적화 및 조정합니다.

DB2 문제점 해결 정보

DB2 데이터베이스 제품을 사용하는 데 도움이 되는 광범위한 문제를 해결하고 판별할 수 있는 정보가 있습니다.

DB2 문서
문제점 해결 정보는 *DB2 문제점 해결 안내서* 또는 *DB2 정보 센터*의 데이터베이스 기본 섹션을 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 등의 정보가 있습니다.

DB2 기술 지원 웹 사이트
문제점이 있는 경우 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 대한 가능한 솔루션을 찾을 수 있습니다.

http://www.ibm.com/software/data/db2/support/db2_9/에서 DB2 기술 지원 웹 사이트에 액세스하십시오.

이용약관

다음 조건에 따라 이 책을 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 귀하 사업장 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서의 2차적 저작물을 만들거나 본 문서 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 본 문서의 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 본 문서의 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 F. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. 비IBM 제품에 대한 정보는 이 책을 처음 발행할 때의 정보에 기초하고 있으며 변경될 수 있습니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트 문자 세트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

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

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등) 하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 다른 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 되는 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 샘플 프로그램은 어떠한 보증없이 ″있는 그대로″ 제공됩니다. IBM은 샘플 프로그램의 사용으로 인해 발생하는 모든 손해에 대해 책임을 지지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. *_enter* 연도_. All rights reserved.

상표

IBM, IBM 로고 및 ibm.com®은 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 기타 회사의 상표입니다. 현재 IBM 상표 목록은 웹 “저작권 및 상표 정보”(www.ibm.com/legal/kr/copytrade.shtml)에 있습니다.

다음 표장은 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]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[카]

[타]

[파]

[하]

A

B

C

D

E

F

G

I

K

L

M

N

P

R

S

T

U

V

W

X

SA30-3969-00

Spine information:

Linux, UNIX 및 Windows용 IBM DB2 9.7

데이터 이동 유틸리티 안내서 및 참조서

IBM