IBM® DB2 Universal Database™

관리 안내서: 구현

버전 8.2

SA30-1479-01

IBM® DB2 Universal Database™

관리 안내서: 구현

버전 8.2

SA30-1479-01

이 정보 및 이 정보가 지원하는 제품을 사용하기 전에 반드시 *주의사항*에 나와 있는 일반 정보를 읽으십시오.

IBM 서적을 주문하려면 온라인을 통하거나 한국 IBM 담당자에게 문의하십시오.

- 책을 온라인으로 주문하려면 IBM Publications Center(www.ibm.com/shop/publications/order)를 방문하십시오.
- 한국 IBM 담당자에게 문의하려면 IBM Directory of Worldwide Contacts(www.ibm.com/planetwide)를 방문하십시오.

미국이나 캐나다의 DB2 마케팅 및 판매 부서에서 DB2 책을 주문하려면 1-800-IBM-4YOU(426-4968)로 전화하십시오.

IBM에 정보를 보내는 경우, IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

목차

이 책에 대한 정보

3개 볼륨으로 된 관리 안내서에서는 DB2 관계형 데이터베이스 관리 시스템(RDBMS) 제품을 사용하고 관리하는 데 필요한 정보를 제공하며, 내용은 다음과 같습니다.

- 데이터베이스 설계에 대한 정보(*관리 안내서: 계획*)
- 데이터베이스의 구현 및 관리에 대한 정보(*관리 안내서: 구현*)
- 성능 개선을 위한 데이터베이스 환경의 구성 및 조정에 대한 정보(*관리 안내서: 성능*)

이 책에서 다루어지는 여러 태스크는 서로 다른 인터페이스를 사용하여 수행될 수 있습니다.

- 그래픽 인터페이스로부터 데이터베이스를 액세스하고 조작할 수 있도록 해주는 **명령행 처리기**. 이 인터페이스에서 SQL문과 DB2 유틸리티 기능을 실행할 수도 있습니다. 이 책에 있는 예의 대부분이 이 인터페이스를 사용하여 설명되어 있습니다. 명령행 처리기 사용에 관한 자세한 정보는 *Command Reference*를 참조하십시오.
- 응용프로그램 내에서 DB2 유틸리티 기능을 실행할 수 있도록 해주는 **API**(Application Programming Interface). API 사용에 관한 자세한 정보는 *Administrative API Reference*를 참조하십시오.
- 시스템 구성, 디렉토리 관리, 시스템 백업 및 복구, 작업 스케줄 작성, 미디어 관리와 같은 관리 태스크를 수행하기 위해 사용자 인터페이스를 사용할 수 있도록 해주는 **제어 센터**. 제어 센터에는 시스템 간의 데이터 복제를 설정할 수 있도록 해주는 복제 관리도 포함되어 있습니다. 그리고 제어 센터는 그래픽 사용자 인터페이스를 통해 DB2 유틸리티 기능을 실행할 수 있게 합니다. 사용자 플랫폼에 따라 제어 센터를 호출하는 여러가지 방법이 있습니다. 예를 들어, 명령행에서 `db2cc` 명령을 사용하거나 DB2 폴더에서 제어 센터 아이콘을 선택하거나 Windows 플랫폼에서 시작 메뉴를 사용하십시오. 좀더 자세한 내용을 보려면, 제어 센터 창의 **도움말** 풀다운 메뉴에서 **시작하기**를 선택하십시오. **Visual Explain** 도구는 제어 센터로부터 호출됩니다.

  제어 센터에는 다음과 같은 세 가지 보기가 있습니다.

  - 기본. 이 보기는 데이터베이스, 테이블 및 스토어드 프로시저와 같은 필수 오브젝트에 관한 핵심 DB2 UDB 기능을 표시합니다.
  - 고급. 이 보기는 모든 오브젝트 및 조치가 사용 가능합니다. 엔터프라이즈 환경에서 작업 중이며 z/OS 또는 IMS용 DB2에 연결하려는 경우 이 보기를 사용하십시오.
  - 사용자 정의. 이 보기는 오브젝트 트리 및 오브젝트 조치를 조정할 수 있는 기능을 제공합니다.

관리 태스크를 수행하는 데 사용할 수 있는 기타 도구가 있습니다. 도구는 다음과 같습니다.

- 명령 센터를 대신하며 SQL문의 생성, 편집, 실행 및 조작, IMS 및 DB2 명령, 결과 출력에 대한 작업 및 설명된 SQL문에 대한 그래픽 액세스 플랜 표시 보기에 사용되는 명령 편집기
- 원시(native) SQL PSM(Persistent Storage Module) 스토어드 프로시저, iSeries 버전 5 릴리스 3 이상에 대한 Java 스토어드 프로시저, 사용자 정의 함수(UDF) 및 구조화된 유형에 대한 지원을 제공하는 개발 센터
- Health Center에서는 성능 및 자원 할당 문제를 해결하는 데 있어서 DBA를 지원하기 위한 도구를 제공합니다.
- 제어 센터, Health Center 및 복제 센터의 설정값을 변경하는 도구 설정값
- 작업이 자동으로 실행되도록 스케줄하는 저널
- 웨어하우스 오브젝트를 관리하는 Data Warehouse Center

이 책의 사용자

이 책은 기본적으로 로컬 또는 리모트 클라이언트가 액세스할 데이터베이스를 설계, 구현 및 관리해야 하는 데이터베이스 관리자, 시스템 관리자, 보안 관리자 및 시스템 운영자를 대상으로 합니다. 또한 프로그래머나 DB2 Universal Database™(DB2 UDB) 관계형 데이터베이스 관리 시스템의 관리 및 조작을 이해해야 하는 기타 사용자들에게도 도움이 됩니다.

이 책의 구성 방법

이 책에는 다음 주요 주제에 대한 정보가 들어 있습니다.

설계 구현

- 제 1 장 『데이터베이스를 작성하기 전에』에서는 데이터베이스와 데이터베이스 내의 오브젝트를 작성하기 전에 필요한 전제조건에 대해 설명합니다.
- 제 2 장 『DAS(DB2 Administration Server) 작성 및 사용』에서는 DAS의 정의, 작성 방법 및 사용 방법을 설명합니다.
- 제 3 장 『데이터베이스 작성』에서는 데이터베이스와 데이터베이스 내의 오브젝트를 작성하는 것과 연관된 태스크에 대해 설명합니다.
- 제 4 장 『테이블 및 기타 관련 테이블 오브젝트 작성』에서는 데이터베이스 설계를 구현할 때 특정 특성을 사용하여 테이블을 작성하는 방법을 설명합니다.
- 제 5 장 『데이터베이스 경고』에서는 데이터베이스와 데이터베이스 내의 오브젝트를 변경하고 삭제하는 것과 연관된 전제조건과 태스크에 대해 설명합니다.

- 제 6 장『테이블 및 기타 관련 테이블 오브젝트 변경』에서는 테이블을 삭제하는 방법 및 해당 테이블과 연관된 특정 특성을 수정하는 방법을 설명합니다. 또한 관련된 테이블 오브젝트의 삭제 및 수정을 보여줍니다.

데이터베이스 보안

- 제 7 장『데이터베이스 액세스 제어』에서는 데이터베이스 자원에 대한 액세스를 제어하는 방법에 대해 설명합니다.
- 제 8 장『DB2 Universal Database™(DB2 UDB) 활동 감사』에서는 원하지 않거나 예상하지 않은 데이터 액세스를 검출하고 모니터하는 방법에 대해 설명합니다.

부록

- 부록 A『이름 지정 규칙 준수』에서는 데이터베이스 및 오브젝트를 이름 지정할 때 지켜야 할 규칙을 제공합니다.
- 부록 B『자동 클라이언트 재라우트 사용』에서는 클라이언트 응용프로그램의 자동 재라우팅 및 해당 지원을 사용 가능하게 하는 방법을 설명합니다.
- 부록 C『LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스 사용』에서는 LDAP 디렉토리 서비스 사용 방법에 대한 정보가 제공됩니다.
- 부록 D『다중 데이터베이스 파티션에 대한 명령 발행』에서는 파티션된 데이터베이스 환경의 모든 파티션으로 명령을 송신하는 *db2_all*과 *rah* 쉘 스크립트를 사용하는 방법을 설명합니다.
- 부록 E『WMI(Windows Management Instrumentation) 지원 사용』에서는 WMI를 사용하여 DB2를 관리할 수 있는 방법에 대한 정보를 제공합니다.
- 부록 F『Windows NT 보안 사용』에서는 DB2가 Windows 보안 기능을 사용하여 작업하는 방법을 설명합니다.
- 부록 G『Windows 성능 모니터 사용』에서는 Windows 성능 모니터를 사용하여 DB2 성능 데이터를 수집하는 방법을 설명합니다.
- 부록 H『Windows 데이터베이스 파티션 서버 사용』에서는 파티션된 데이터베이스 서버에서 작업하는 Windows에 의해 사용되는 유틸리티를 설명합니다.
- 부록 I『다중 논리 노드 구성』에서는 파티션된 데이터베이스 환경에서 다중 논리 노드를 구성하는 방법을 설명합니다.
- 부록 J『제어 센터 확장』에서는 새 조치를 포함한 새 도구 막대 단추, 새 오브젝트 정의 및 새 조치 정의를 추가하여 제어 센터를 확장하는 방법에 대한 정보를 제공합니다.

관리 안내서: 구현 매뉴얼에서 제목이『데이터 이동 유틸리티』인 장이 이동되었습니다.

주: 데이터 이동을 위한 DB2 유틸리티에 대한 모든 정보와 *Command Reference* 및 *Administrative API Reference*의 유사한 주제는 *데이터 이동 유틸리티 안내 및 참조서*로 통합되어 있습니다.

*데이터 이동 유틸리티 안내 및 참조서*는 이들 주제에 대한 1차 단일 정보 소스입니다.

데이터 복제에 대한 자세한 정보는 *IBM DB2 Information Integrator SQL 복제 안내 및 참조서*를 참조하십시오.

관리 안내서: 구현 매뉴얼에서 제목이 『데이터베이스 복구』인 장이 이동되었습니다.

주: 데이터 백업 및 복구를 위한 방법과 도구에 대한 모든 정보와 *Command Reference* 및 *Administrative API Reference*의 유사한 주제는 *데이터 복구 및 고가용성 안내 및 참조서*에 통합되어 있습니다.

*데이터 복구 및 고가용성 안내 및 참조서*는 이들 주제에 대한 1차 단일 정보 소스입니다.

관리 안내서의 다른 볼륨에 대한 간단한 개요

관리 안내서: 계획

*관리 안내서: 계획*은 데이터베이스 설계와 관련됩니다. 이 책에서는 논리적 및 실제적인 설계 문제와 분산 트랜잭션 문제에 대해 설명합니다. 관리 안내서, 성능에 있는 특정 장 및 부록에 대해 다음에서 간략히 설명합니다.

데이터베이스 개념

- "기본 관계형 데이터베이스 개념"에서는 복구 오브젝트, 스토리지 오브젝트 및 시스템 오브젝트를 포함하여 데이터베이스 오브젝트의 개요를 나타냅니다.
- "병렬 데이터베이스 시스템"에서는 DB2에서 사용 가능한 병렬 처리 유형에 대한 소개를 제공합니다.
- "데이터 웨어하우스 정보"에서는 데이터 웨어하우스 및 데이터 웨어하우스 태스크의 개요를 제공합니다.

데이터베이스 설계

- "논리 데이터베이스 설계"에서는 논리 데이터베이스 설계에 대한 개념 및 지시사항을 설명합니다.
- "실제 데이터베이스 설계"에서는 데이터 스토리지와 관련된 고려사항을 포함하여 실제 데이터베이스 설계에 대한 지시사항을 설명합니다.
- "분산 데이터베이스 설계"에서는 단일 트랜잭션에서 다중 데이터베이스에 액세스하는 방법을 설명합니다.
- "트랜잭션 관리 프로그램 설계"에서는 분산 트랜잭션 프로세싱 환경에서 데이터베이스를 사용하는 방법에 대해 설명합니다.

부록

- ″릴리스간 비호환성″에서는 알고 있어야 할 장래의 비호환성 뿐만 아니라 버전 7과 버전 8에서 소개한 비호환성을 설명합니다.
- ″자국어 지원(NLS)″에서는 지역, 언어 및 코드 페이지의 정보가 포함된 DB2 자국어 지원에 대해 설명합니다.
- ″64비트 환경에서 대용량 페이지 지원 사용(AIX)″에서는 16MB 페이지 *크기*에 대한 지원 및 해당 지원을 사용 가능하게 하는 방법을 설명합니다.

관리 안내서: 성능

*관리 안내서: 성능*은 성능 문제와 관련됩니다. 즉, 이 주제 및 문제는 사용자 응용프로그램과 DB2 Universal Database 제품 자체의 성능을 설정, 테스트 및 개선하는 것과 연관됩니다. 관리 안내서, 성능에 있는 특정 장 및 부록에 대해 다음에서 간략히 설명합니다.

성능 개요

- ″성능 개요″에서는 DB2 UDB 성능을 관리하고 개선하기 위한 개념 및 고려사항을 소개합니다.
- ″아키텍처 및 프로세스″에서는 주요 DB2 Universal Database 아키텍처 및 프로세스에 대해 소개합니다.

응용프로그램 성능 조정

- ″응용프로그램 고려사항″에서는 응용프로그램을 설계할 때 데이터베이스 성능을 향상시키기 위한 몇몇 기술에 대해 설명합니다.
- ″환경상의 고려사항″에서는 데이터베이스 환경을 설정할 때 데이터베이스 성능을 향상시키기 위한 몇몇 기술에 대해 설명합니다.
- ″시스템 카탈로그 통계″에서는 사용자 데이터에 대한 통계를 수집하고 이를 사용하여 최적의 성능을 보장할 수 있는 방법을 설명합니다.
- ″SQL 컴파일러 이해″에서는 SQL 컴파일러로 컴파일할 때 SQL문에 발생하는 현상에 대해 설명합니다.
- ″SQL Explain 기능″에서는 SQL 컴파일러가 데이터에 액세스하기 위해 취하는 선택항목을 살펴볼 수 있는 Explain 기능에 대해 설명합니다.

시스템의 조정 및 구성

- ″조작 성능″에서는 데이터베이스 관리 프로그램이 메모리를 사용하는 방법 및 런타임 성능에 영향을 주는 기타 고려사항에 대한 개요를 설명합니다.
- ″조정자(Governor) 사용″에서는 조정자를 사용하여 데이터베이스 관리의 일부 양상을 제어하는 방법에 대해 설명합니다.
- ″구성 조정″에서는 데이터베이스 시스템의 *크기* 증가에 연관된 일부 고려사항 및 태스크에 대해 설명합니다.

- ″데이터베이스 파티션에 데이터 재분배에서는 파티션″된 데이터베이스 환경에서 파티션에 데이터를 재분배하는 데 필요한 태스크에 대해 설명합니다.
- ″벤치마크 테스트″에서는 벤치마크 테스트의 개요 및 벤치마크 테스트 수행 방법을 보여줍니다.
- ″DB2 구성″에서는 데이터베이스 관리 프로그램 데이터베이스 구성 파일과 데이터베이스 관리 프로그램, 데이터베이스 및 DAS 구성 매개변수의 값에 대해 설명합니다.

부록

- ″DB2 레지스트리 및 환경 변수″에서는 프로파일 등록 값 및 환경 변수를 설명합니다.
- ″Explain 테이블 및 정의″에서는 DB2 Explain 기능에서 사용되는 테이블 및 해당 테이블의 작성 방법을 설명합니다.
- ″SQL Explain 도구″에서는 DB2 explain 도구 즉, db2expln 및 dynexpln의 사용 방법을 설명합니다.
- ″db2exfmt -- Explain 테이블 포맷 도구″에서는 DB2 explain 도구를 사용하여 explain 테이블 데이터를 포맷하는 방법을 설명합니다.

제 1 부 설계 구현

제 1 장 데이터베이스를 작성하기 전에

데이터베이스의 설계를 판별한 후에, 데이터베이스 및 오브젝트를 작성해야 합니다. 이들 오브젝트에는 스키마, 데이터베이스 파티션 그룹, 테이블 스페이스, 테이블, 뷰, 랩퍼, 서버, 별칭, 유형 맵핑, 함수 맵핑, 별명, 사용자 정의 유형(UDT), 사용자 정의 함수(UDF) 및 자동 요약 테이블(AST)이 포함됩니다. 이들 오브젝트는 명령행 처리기의 SQL문을 사용하거나 응용프로그램의 SQL문을 통해 작성할 수 있습니다.

SQL문에 대한 정보는 *SQL 참조서* 매뉴얼을 참조하십시오. 명령행 처리기 명령에 대한 정보는 *Command Reference* 매뉴얼을 참조하십시오. API에 대한 정보는 *Administrative API Reference* 매뉴얼을 참조하십시오.

데이터베이스 오브젝트는 제어 센터를 통해 작성할 수도 있습니다. SQL문, 명령행 처리기 명령 또는 API 대신 제어 센터를 사용할 수 있습니다.

이 장에서 제어 센터를 사용하여 태스크를 완료하는 메소드는 상자에 이를 위치시켜 강조표시됩니다. 이것은 명령행을 사용하는 비교 대상 메소드가 바로 다음에 오며, 때로는 예제를 동반합니다. 일부 경우에, 오직 하나의 메소드만을 표시하는 태스크가 있을 수 있습니다. 제어 센터에서 작업할 때, 여기에 있는 개요 정보보다 더 자세히 제공하는 도움말을 사용할 수 있음을 기억하십시오.

이 장에서는 관련된 모든 오브젝트로 데이터베이스를 작성하기 전에 알아야 하는 정보에 초점을 둡니다. 데이터베이스를 작성하기 전에 수행해야 하는 여러 태스크와 마찬가지로 여러 전제조건 개념 및 주제가 있습니다.

이 장의 다음 장에는 사용자 데이터베이스 설계 구현의 일부일 수 있는 다양한 오브젝트에 대한 간단한 설명이 들어 있습니다.

이 파트의 마지막 장은 데이터베이스를 변경하기 전에 고려해야 하는 주제를 나타낸 다음 데이터베이스 오브젝트를 변경하거나 제거하는 방법을 설명합니다.

DB2 Universal Database가 운영 체제와 상호 작용한 영역의 경우, 이 장과 다음 장의 일부 주제는 운영 체제 특유의 차이점을 나타냅니다. 운영 체제의 원시(native) 성능 또는 DB2 UDB에 의해 제공된 성능 간의 차이점을 사용할 수 있습니다. 정확한 차이점에 대해서는 해당 *빠른 시작* 매뉴얼 및 운영 체제 관련 문서를 참조하십시오.

예를 들어, Windows에서는 『서비스』라고 하는 응용프로그램 유형을 지원합니다. Windows용 DB2에는 서비스라고 정의된 DB2 인스턴스가 있습니다. 서비스는 서비스 제어판 애플릿을 통한 사용자 또는 Microsoft Windows 32비트 API에 포함된 서비스 기능을 사용하는 Windows 32비트 응용프로그램에 의해 시스템 시동에서 자동으로 시작할 수 있습니다. 서비스는 시스템에 로그온한 사용자가 없어도 실행할 수 있습니다.

특별히 언급하지 않는한 Windows 9x에 대한 참조는 Windows 98 및 Windows ME를 가리킵니다. Windows NT에 대한 참조는 Windows NT, Windows 2000, Windows XP 및 Windows Server 2003을 가리킵니다. Windows에 대한 참조는 지원되는 모든 Windows 운영 체제를 가리킵니다.

인스턴스에 대한 작업

데이터베이스를 구현하기 전에 다음 전제조건에 대해 이해하고 있어야 합니다.

- 『UNIX에서 DB2 UDB 시작』
- 5 페이지의 『Windows에서 DB2 UDB 시작』
- 6 페이지의 『데이터베이스 관리 프로그램의 다중 인스턴스』
- 8 페이지의 『스키마별 오브젝트 그룹화』
- 9 페이지의 『병렬 처리』
- 13 페이지의 『데이터베이스에서 데이터 파티션 사용』
- 14 페이지의 『UNIX에서 인스턴스 중지』

UNIX에서 DB2 UDB 시작

일반적인 비즈니스 조작을 수행하는 동안 DB2 Universal Database™(DB2 UDB)를 시작하거나 중지시켜야 하는 경우가 있습니다. 예를 들어, 다음 태스크를 수행하기 전에 인스턴스를 시작해야 합니다.

- 인스턴스에서 데이터베이스에 연결
- 응용프로그램 프리컴파일
- 패키지를 데이터베이스에 바인드
- 호스트 데이터베이스에 액세스

전제조건:

시스템에서 DB2 UDB 인스턴스를 시작하려면, 다음을 수행하십시오.

1. 인스턴스에 대해 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있는 사용자 ID 및 이름으로 로그인하거나 인스턴스 소유자로 로그인하십시오.
2. 시작 스크립트를 다음과 같이 실행하십시오.

```
. INSTHOME/sqllib/db2profile      (Bourne 또는 Korn 쉘의 경우)
source INSTHOME/sqllib/db2cshrc   (C 쉘의 경우)
```

여기서, INSTHOME은 사용할 인스턴스의 홈 디렉토리입니다.

프로시저:

다음 두 방법을 사용하여 인스턴스를 시작하십시오.

1. 제어 센터를 사용하여 인스턴스를 시작하십시오.

1. **인스턴스** 폴더가 나올 때까지 오브젝트 트리를 확장하십시오.
2. 시작하고자 하는 인스턴스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **시작**을 선택하십시오.

2. 명령행을 사용하여 인스턴스를 시작하려면, 다음을 입력하십시오.

```
db2start
```

태스크 관련:

- 14 페이지의『UNIX에서 인스턴스 중지』
- 28 페이지의『현재 인스턴스 설정』
- 5 페이지의『Windows에서 DB2 UDB 시작』

Windows에서 DB2 UDB 시작

일반적인 비즈니스를 수행하는 동안 DB2 Universal Database™(DB2 UDB)를 시작하거나 중지시켜야 하는 경우가 있습니다. 예를 들어, 다음 태스크를 수행하기 전에 인스턴스를 시작해야 합니다.

- 인스턴스에서 데이터베이스에 연결
- 응용프로그램 프리컴파일
- 패키지를 데이터베이스에 바인드
- 호스트 데이터베이스에 액세스

전제조건:

db2start를 사용하여 DB2 UDB를 서비스로서 시작하려면, 사용자 어카운트가 Windows NT 운영 체제에 정의된 대로 Windows 서비스를 시작할 수 있는 올바른 특권이 있어야 합니다. 사용자 어카운트는 Administrators, Server Operators 또는 Power Users 그룹의 구성원이 될 수 있습니다.

프로시저:

다음 두 방법을 사용하여 인스턴스를 시작하십시오.

1. 제어 센터를 사용하여 인스턴스를 시작하십시오.

1. **인스턴스** 폴더가 나올 때까지 오브젝트 트리를 확장하십시오.
2. 시작하고자 하는 인스턴스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **시작**을 선택하십시오.

2. 명령행을 사용하여 인스턴스를 시작하려면, 다음을 입력하십시오.

```
db2start
```

db2start 명령이 DB2 UDB를 Windows 서비스로서 시작합니다. **db2start**를 호출할 때 ″/D″ 스위치를 지정함으로써, Windows 상의 DB2 UDB를 프로세스로서 실행할 수도 있습니다. 또한, DB2 UDB는 제어판 또는 ″NET START″ 명령을 사용하여 서비스로서 시작될 수도 있습니다.

파티션된 데이터베이스 환경에서 실행될 때, 각 데이터베이스 파티션 서버는 Windows 서비스로서 시작됩니다. 파티션된 데이터베이스 환경에서는 ″/D″ 스위치를 사용하여 DB2를 프로세스로서 시작할 수 없습니다.

태스크 관련:

- 4 페이지의 『UNIX에서 DB2 UDB 시작』
- 14 페이지의 『UNIX에서 인스턴스 중지』
- 16 페이지의 『Windows에서 인스턴스 중지』

데이터베이스 관리 프로그램의 다중 인스턴스

데이터베이스 관리 프로그램의 다중 인스턴스를 단일 서버에 작성할 수 있습니다. 이는 사용자가 실제 머신에 동일한 제품의 여러 인스턴스를 작성하여 이를 동시에 실행할 수 있음을 의미합니다. 이는 환경 설정에 있어서 융통성을 제공합니다.

다음 환경 작성을 위해 다중 인스턴스를 가질 수도 있습니다.

- 실행 환경에서 개발 환경을 분리시키기 위해
- 서비스하는 특정 응용프로그램에 대해 각각 개별적으로 조정하기 위해
- 관리자로부터 관련 정보를 보호하기 위해. 예를 들어, 급여 데이터베이스가 자신의 인스턴스에 대해 보호되어 있어, 다른 인스턴스의 소유자가 급여 데이터를 볼 수 없게 끔 합니다.

주: (UNIX® 운영 체제에 한함) 둘 이상의 인스턴스 간의 환경 충돌을 피하기 위해 각 인스턴스가 자체 고유의 파일 시스템을 갖도록 해야 합니다. 홈 파일 시스템이 공유되면 오류가 리턴됩니다.

DB2® Universal Database(DB2 UDB) 프로그램 파일은 특정 머신의 하나의 위치에 실제로 저장됩니다. 작성된 각각의 인스턴스가 다시 이 위치를 지시하고 있기 때문에, 작성된 각 인스턴스에 대해 프로그램 파일이 중복되지는 않습니다. 관련된 여러 데이터베이스가 단일 인스턴스에 놓일 수 있습니다.

인스턴스는 노드 디렉토리에 로컬 또는 리모트로 카탈로그화됩니다. 디폴트 인스턴스는 DB2INSTANCE 환경 변수에 의해 정의됩니다. 데이터베이스 작성, 응용프로그램의 강제 해제, 데이터베이스의 모니터 또는 데이터베이스 관리 구성의 갱신과 같이 인스턴스 레벨에서만 수행할 수 있는 태스크를 개발하고 유지보수하기 위해 다른 인스턴스에 **접속**할 수 있습니다. 디폴트 인스턴스가 아닌 다른 인스턴스에 접속하려고 시도할 때, 해당 인스턴스와 통신하는 방법을 판별하는 데 노드 디렉토리가 사용됩니다.

관련 개념:

- 21 페이지의『UNIX 운영 체제에서 다중 인스턴스』
- 22 페이지의『Windows 운영 체제의 다중 인스턴스』

태스크 관련:

- 23 페이지의『추가 인스턴스 작성』

관련 참조:

- *Command Reference*의『ATTACH Command』

데이터베이스 관리 프로그램의 다른 인스턴스에 접속

리모트 상태인 또다른 인스턴스에 접속하려면, **ATTACH** 명령을 사용하십시오.

전제조건:

둘 이상의 인스턴스가 이미 있어야 합니다.

프로시저:

제어 센터를 사용하여 데이터베이스 관리 프로그램의 다른 인스턴스에 접속하려면 다음을 수행하십시오.

1. **인스턴스** 폴더가 나올 때까지 오브젝트 트리를 확장하십시오.
2. 접속하려는 인스턴스를 누르십시오.
3. 선택된 인스턴스 이름을 마우스 오른쪽 단추로 누르십시오.
4. DB2 창에서 접속하려면, 사용자 ID와 암호를 입력한 후 **확인**을 누르십시오.

명령행을 사용하여 인스턴스를 접속하려면, 다음을 입력하십시오.

```
db2 attach to <instance name>
```

예를 들어, 노드 디렉토리 내의 이미 카탈로그된 testdb2라는 인스턴스에 접속합니다.

```
db2 attach to testdb2
```

testdb2 인스턴스에 대한 유지보수 활동을 수행한 후, 다음 명령을 실행하여 해당 인스턴스에서 **DETACH**(접속 해제)할 수 있습니다.

```
db2 detach
```

관련 참조:

- *Command Reference*의『ATTACH Command』
- *Command Reference*의『DETACH Command』

스키마별 오브젝트 그룹화

데이터베이스 오브젝트 이름은 단일 ID로 구성되거나, 두 개의 ID로 구성되는 *스키마 규정 오브젝트*입니다. 스키마 규정 오브젝트의 스키마 또는 상위 자리 부분은 데이터베이스의 그룹 오브젝트를 분류하거나 그룹화할 수 있는 방법을 제공합니다. 테이블, 뷰, 별명, 구별 유형, 함수, 인덱스, 패키지 또는 트리거와 같은 오브젝트가 작성될 때, 오브젝트는 스키마에 지정됩니다. 이러한 지정은 외부적으로 또는 내재적으로 완료됩니다.

명령문에서 해당 오브젝트를 참조할 때 두 부분으로 구성된 오브젝트 이름의 상위 자리 부분을 사용하는 경우 스키마를 명시적으로 사용하게 됩니다. 예를 들어, 사용자 A는 CREATE TABLE문을 다음과 같이 스키마 C로 발행합니다.

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

두 부분으로 구성된 오브젝트 이름의 상위 자리 부분을 사용하지 않는 경우 스키마를 내재적으로 사용하게 됩니다. 이 경우, 오브젝트 이름의 상위 자리 부분을 완료하는 데 사용되는 스키마 이름을 식별하는 데 CURRENT SCHEMA 특수 레지스터가 사용됩니다. CURRENT SCHEMA의 초기값은 현재 세션 사용자의 권한 부여 ID입니다. 현재 세션 중에 이 값을 변경하려는 경우, SET SCHEMA문을 사용하여 또다른 스키마 이름으로 특수 레지스터를 설정할 수 있습니다.

데이터베이스가 작성될 때 몇몇의 오브젝트는 특정 스키마내에서 작성되고 시스템 카탈로그 테이블에 저장됩니다.

동적 SQL문에서 스키마 규정 오브젝트 이름은 CURRENT SCHEMA 특수 레지스터 값을 규정되지 않은 오브젝트 이름 참조용 규정자로서 내재적으로 사용합니다. 정적 SQL 문에서 QUALIFIER 프리컴파일/바인드 옵션은 규정되지 않은 데이터베이스 오브젝트 이름에 대한 규정자를 내재적으로 지정합니다.

사용자 오브젝트를 작성하기 전에, 사용자 스키마에 오브젝트를 작성하려는지, 아니면 논리적으로 오브젝트를 그룹화하는 다른 스키마를 사용하여 작성하려는지를 고려해야 합니다. 공유될 오브젝트를 작성하는 경우, 다른 스키마 이름을 사용하는 것이 바람직합니다.

관련 개념:

- 80 페이지의『시스템 카탈로그 테이블 정의』

태스크 관련:

- 99 페이지의『스키마 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『SET SCHEMA문』
- *SQL 참조서, 볼륨 1*의『CURRENT SCHEMA 특수 레지스터』

병렬 처리

구성 매개변수를 수정하여 데이터베이스 파티션 내의 또는 파티션되지 않은 데이터베이스 내의 병렬 처리를 사용하십시오. 예를 들어, 파티션 내 병렬 처리를 사용하여 멀티프로세서(SMP) 머신에서 다중 프로세서를 사용할 수 있습니다.

파티션간 쿼리 병렬 처리 사용

프로시저:

파티션간 병렬 처리는 이 파티션의 데이터베이스 파티션 수와 데이터 분산에 기초하여 자동으로 발생합니다.

관련 개념:

- *관리 안내서: 계획*의 『파티션 및 프로세서 환경』
- *관리 안내서: 계획*의 『데이터 파티션』
- *관리 안내서: 계획*의 『데이터베이스 파티션 그룹 설계』
- *관리 안내서: 성능*의 『파티션된 데이터베이스에서의 파티션』

태스크 관련:

- 9 페이지의 『쿼리를 위해 파티션내 병렬 처리 사용』
- 13 페이지의 『데이터베이스에서 데이터 파티션 사용』
- *관리 안내서: 성능*의 『파티션에 걸친 데이터 재분배』

쿼리를 위해 파티션내 병렬 처리 사용

프로시저:

제어 센터는 특정 데이터베이스 또는 데이터베이스 관리 프로그램 구성 파일에서 각 항목의 값을 찾아내거나, 수정하는 데 사용될 수 있습니다.

특정 데이터베이스 또는 데이터베이스 관리 프로그램 구성 파일의 개별 항목 값을 알고자 하면, **GET DATABASE CONFIGURATION** 및 **GET DATABASE MANAGER CONFIGURATION** 명령을 사용하십시오. 특정 데이터베이스 또는 데이터베이스 관리 프로그램 구성 파일의 개별 항목 값을 수정하려면, **UPDATE DATABASE CONFIGURATION** 및 **UPDATE DATABASE MANAGER CONFIGURATION** 명령을 각각 사용하십시오.

파티션 내 병렬 처리에 영향을 미치는 구성 매개변수는 *max_querydegree* 및 *intra_parallel* 데이터베이스 관리 프로그램 매개변수, *dft_degree* 데이터베이스 매개변수를 포함합니다.

파티션내 병렬 처리가 발생하려면, 하나 이상의 데이터베이스 구성 매개변수, 데이터베이스 관리 프로그램 매개변수, 프리컴파일, 바인드 옵션 또는 특수 레지스트리를 수정해야합니다.

intra_parallel

데이터베이스 관리 프로그램의 파티션 내 병렬 처리 사용 가능 여부를 지정하는 데이터베이스 관리 프로그램 구성 매개변수. 디폴트값은 파티션내 병렬 처리에서 사용되지 않습니다.

max_querydegree

이 인스턴스에서 실행되는 SQL문에 사용되는 파티션 내 병렬 처리의 최대 수준을 지정하는 데이터베이스 관리 프로그램 구성 매개변수. SQL문은 파티션 내에서 병렬 조작을 실행할 때 이 매개변수에 의해 주어지는 숫자보다 더 큰 숫자를 사용하지 않습니다. *max_querydegree*의 값을 사용할 경우에는 *intra_parallel* 구성 매개변수의 값이 『예』로 설정되어 있어야 합니다. 이 구성 매개변수의 디폴트값은 -1입니다. 이 값은 시스템이 옵티마이저에 의해 결정된 병렬 처리 수준을 사용함을 의미합니다. 그 외에는 사용자 지정 값이 사용됩니다.

dft_degree

데이터베이스 구성 매개변수. DEGREE 바인드 옵션 및 CURRENT DEGREE 특수 레지스터에 대한 디폴트값을 제공합니다. 디폴트값은 1입니다. ANY 값은 시스템이 옵티마이저에 의해 결정된 병렬 처리 수준을 사용함을 의미합니다.

DEGREE

정적 SQL에 대한 프리컴파일 또는 바인드 옵션

CURRENT DEGREE

동적 SQL용 특수 레지스터

관련 개념:

- *관리 안내서: 성능*의 『응용프로그램에 대한 병렬 처리』
- *관리 안내서: 성능*의 『병렬 처리 정보』

태스크 관련:

- *관리 안내서: 성능*의 『구성 매개변수를 사용하여 DB2 구성』

관련 참조:

- *관리 안내서: 성능*의 『max_querydegree - 병렬 처리의 최대 쿼리 수준 구성 매개변수』
- *관리 안내서: 성능*의 『intra_parallel - 파티션내 병렬 처리 사용 구성 매개변수』
- *관리 안내서: 성능*의 『dft_degree - 디폴트 등급 구성 매개변수』
- *Command Reference*의 『BIND Command』
- *Command Reference*의 『PRECOMPILE Command』

- *SQL 참조서, 볼륨 1*의 『CURRENT DEGREE 특수 레지스터』

유틸리티에 대한 파티션내 병렬 처리 사용

이 절에서는 다음 유틸리티에 대한 파티션 내 병렬 처리 사용 방법의 개요를 설명합니다.

- 로드
- 인덱스 작성
- 데이터베이스 또는 테이블 스페이스 백업
- 데이터베이스 또는 테이블 스페이스 리스토어

유틸리티에 대한 파티션간 병렬 처리는 데이터베이스 파티션 수에 기초하여 자동으로 발생합니다.

데이터 로드에 병렬 처리 사용: 로드 유틸리티가 자동으로 병렬 처리를 사용하거나 또는 다음 매개변수를 **LOAD** 명령에서 사용할 수 있습니다.

- CPU_PARALLELISM
- DISK_PARALLELISM

파티션된 데이터베이스 환경에서 데이터 로드에 대한 파티션간 병렬 처리는 다중 파티션에 목표 테이블이 정의될 때 자동으로 발생합니다. 데이터 로드에 대한 파티션간 병렬 처리는 OUTPUT_DBPARTNUMBS를 지정하여 겹쳐쓸 수 있습니다. 로드 유틸리티는 또한 목표 파티션의 *크기*에 따라 데이터 파티션 병렬 처리를 인공지능식으로 사용가능화합니다. MAX_NUM_PART_AGENTS를 사용하여 로드 유틸리티가 선택한 병렬 처리의 최대 수준을 제어할 수 있습니다. ANYORDER가 지정되어 있을 때 PARTITIONING_DBPARTNUMS를 지정하여 데이터 파티션 병렬 처리를 겹쳐쓸 수 있습니다.

관련 개념:

- *데이터 이동 유틸리티 안내 및 참조서*의 『로드 개요』
- *데이터 이동 유틸리티 안내 및 참조서*의 『파티션된 데이터베이스 로드 - 개요』

인덱스 작성시 병렬 처리 사용: 인덱스를 작성할 때 병렬 처리를 사용하려면, 다음과 같은 상태여야 합니다.

- *intra_parallel* 데이터베이스 관리 프로그램 구성 매개변수가 ON 상태여야 합니다.
- 테이블은 병렬 처리에서 도움이 될 수 있도록 커야 합니다.
- SMP 머신에서 다중 프로세서가 사용 가능 상태여야 합니다.

관련 참조:

- *관리 안내서: 성능*의 『intra_parallel - 파티션내 병렬 처리 사용 구성 매개변수』
- *SQL 참조서, 볼륨 2*의 『CREATE INDEX문』

데이터베이스 또는 테이블 스페이스 백업 시 입출력 병렬 처리 사용: 데이터베이스 또는 테이블 스페이스를 백업할 때 입출력 병렬 처리를 사용하려면, 다음을 수행하십시오.

- 둘 이상의 목표 미디어를 사용하십시오.
- 다중 컨테이너를 정의하거나 다중 디스크가 있는 단일 컨테이너를 사용하고 적합한 DB2_PARALLEL_IO 레지스트리 변수를 사용하여, 병렬 I/O가 가능하도록 테이블 스페이스를 구성하십시오. 병렬 I/O를 활용하려면 컨테이너를 정의하기 전에 먼저 수행되어야 할 사항들을 고려해야 합니다. 필요한 상황에 접할 때마다 이러한 사항을 고려할 수 없으며, 데이터베이스나 테이블 스페이스를 백업해야 하는 시점에 이르기 전에 미리 계획되어 있어야 합니다.
- **BACKUP** 명령에서 PARALLELISM 매개변수를 사용하여 병렬 처리 등급을 지정하십시오.
- **BACKUP** 명령에서 WITH num-buffers BUFFERS 매개변수를 사용하여 병렬 처리 등급을 수용할 만큼 충분한 버퍼가 있는지 확인하십시오. 버퍼 수는 소유하고 있는 목표 미디어의 수에 선택된 병렬 처리 수준을 더한 값과 같아야 합니다.

  또한, 다음과 같은 백업 버퍼 크기를 사용하십시오.
  - 실행할 수 있을 만큼의 크기. 4MB 또는 8MB(1024 또는 2048 페이지)가 적당합니다.
  - 최소한 백업 중인 테이블 스페이스의 가장 큰(Extent 크기 * 컨테이너 수) 제품의 크기

관련 참조:

- *Command Reference*의 『BACKUP DATABASE Command』

데이터베이스 또는 테이블 스페이스 리스토어 시 입출력 병렬 처리 사용: 데이터베이스 또는 테이블 스페이스를 리스토어할 때 입출력 병렬 처리를 사용하려면, 다음을 수행하십시오.

- 둘 이상의 소스 미디어를 사용하십시오.
- 병렬 I/O에 대해 테이블 스페이스를 구성하십시오. 컨테이너를 정의하기 전에 이 옵션의 사용을 결정해야 합니다. 필요하다고 느낄 때마다 이 옵션의 사용을 결정할 수 없습니다. 데이터베이스 또는 테이블 스페이스의 리스토어가 필요한 시점에 이르기 전에 미리 계획되어 있어야 합니다.
- **RESTORE** 명령에서 PARALLELISM 매개변수를 사용하여 병렬 처리 등급을 지정합니다.
- **RESTORE** 명령에서 WITH num-buffers BUFFERS 매개변수를 사용하여 병렬 처리를 등급을 수용할 만큼 충분한 버퍼가 있는지 확인하십시오. 버퍼 수는 소유하고 있는 목표 미디어의 수에 선택된 병렬 처리 수준을 더한 값과 같아야 합니다.

  또한, 다음과 같은 기본 리스토어 버퍼 크기를 사용하십시오.

– 실행할 수 있을 만큼의 *크기*. 4MB 또는 8MB(1024 또는 2048 페이지)가 적당합니다.
– 최소한 리스토어 중인 테이블 스페이스의 가장 큰(Extent *크기* * 컨테이너 수) 제품의 *크기*
– 백업 버퍼 *크기*와 같거나 짝수의 배수만큼

관련 참조:

• *Command Reference*의 『RESTORE DATABASE Command』

데이터베이스에서 데이터 파티션 사용

데이터베이스를 작성하기 전에 파티션된 환경에서 데이터베이스를 사용할 것인지 결정해야 합니다. 데이터베이스 설계 결정의 일부로서, 데이터베이스 파티션을 통해 성능을 향상시킬 것인지의 여부를 결정해야 합니다.

다음은 파티션된 데이터베이스 작성과 관련한 몇 가지 고려사항입니다.

프로시저:

파티션된 데이터베이스 환경에서 실행할 때, **CREATE DATABASE** 명령 또는 sqlecrea() API를 사용하여 `db2nodes.cfg` 파일에 있는 모든 노드에서 데이터베이스를 작성할 수 있습니다.

파티션된 데이터베이스를 작성하기 전에 데이터베이스의 카탈로그 노드가 될 데이터베이스 파티션을 선택해야 합니다. 그런 다음 해당 파티션에서 직접 또는 그 파티션에 접속된 리모트 클라이언트에서 데이터베이스를 작성할 수 있습니다. 접속 후, CREATE DATABASE 명령을 실행하는 데이터베이스 파티션은 해당 특정 데이터베이스에 대한 *카탈로그 노드*가 됩니다.

카탈로그 노드는 모든 시스템 카탈로그 테이블이 저장되는 데이터베이스 파티션입니다. 시스템 테이블로의 모든 액세스는 이 데이터베이스 파티션을 거쳐야 합니다. 모든 페더레이티드 데이터베이스 오브젝트(랩퍼, 서버, 별칭 등)는 이 노드에서 시스템 카탈로그 테이블에 저장됩니다.

가능하다면, 별도의 인스턴스에서 각 데이터베이스를 작성해야 합니다. 그렇지 않은 경우, 즉 인스턴스마다 둘 이상의 데이터베이스를 작성해야 한다면, 사용 가능한 데이터베이스 파티션간에 카탈로그 노드를 확대해야 합니다. 이렇게 하면, 단일 데이터베이스 파티션에서 카탈로그 정보에 대한 경합이 줄어듭니다.

주: 카탈로그 노드를 정규적으로 백업하고 사용자 데이터를 여기에 넣지 않아야(가능하면 언제나) 하는데, 그 이유는 다른 데이터가 백업에 필요한 시간을 증가시키기 때문입니다.

데이터베이스를 작성할 때, db2nodes.cfg 파일에서 정의된 모든 데이터베이스 파티션에 걸쳐 자동으로 작성됩니다.

시스템에서 첫 번째 데이터베이스가 작성될 때, 데이터베이스 디렉토리가 형성됩니다. 이것은 사용자가 작성하는 다른 데이터베이스에 대한 정보와 함께 추가됩니다. UNIX에서 작업할 때, 시스템 데이터베이스 디렉토리는 sqldbdir이며 홈 디렉토리 아래 또는 DB2 Universal Database™(DB2 UDB)가 설치된 디렉토리 아래 sqllib 디렉토리에 위치합니다. UNIX에서 작업할 때 이 디렉토리는 공유 파일 시스템(예: UNIX 플랫폼의 NFS)에 있어야 합니다. 그 이유는 파티션된 데이터베이스를 구성하는 모든 데이터베이스 파티션에 대해 시스템 데이터베이스 디렉토리가 하나뿐이기 때문입니다. Windows에서 작업 할 때, 시스템 데이터베이스 디렉토리는 인스턴스 디렉토리에 위치합니다.

또한 sqldbdir 디렉토리에는 시스템 인텐션 파일도 상주합니다. 이것은 sqldbins라 불리며, 데이터베이스 파티션이 동기화된 상태로 있도록 합니다. 모든 데이터베이스 파티션에 걸쳐 단 하나의 디렉토리만이 존재하기 때문에 이 파일은 공유 파일 시스템(예 : UNIX 플랫폼의 NFS)에도 상주해야 합니다. 파일은 데이터베이스를 구성하는 모든 파티션에서 공유합니다.

데이터 파티션 기능을 활용하려면, 구성 매개변수가 수정되어야 합니다. 특정 데이터베이스 또는 데이터베이스 관리 프로그램 구성 파일에 있는 개별 항목의 값을 알아내려면 각각 **GET DATABASE CONFIGURATION** 및 **GET DATABASE MANAGER CONFIGURATION** 명령을 사용하십시오. 특정 데이터베이스 또는 데이터베이스 관리 프로그램 구성 파일에 있는 개별 항목을 수정하려면 각각 **UPDATE DATABASE CONFIGURATION** 및 **UPDATE DATABASE MANAGER CONFIGURATION** 명령을 사용하십시오.

파티션된 데이터베이스에 영향을 주는 데이터베이스 관리 프로그램 구성 매개변수로는 *conn_elapse*, *fcm_num_anchors*, *fcm_num_buffers*, *fcm_num_connect*, *fcm_num_rqb*, *max_connretries*, *max_coordagents*, *max_time_diff*, *num_poolagents*, 및 *stop_start_time*가 있습니다.

태스크 관련:

- *관리 안내서: 성능*의 『구성 매개변수를 사용하여 DB2 구성』

관련 참조:

- *Administrative API Reference*의 『sqlecrea - Create Database』
- *Command Reference*의 『CREATE DATABASE Command』

UNIX에서 인스턴스 중지

데이터베이스 관리 프로그램의 현재 인스턴스를 중지해야 하는 경우도 있습니다.

전제조건:

시스템에서 인스턴스를 중지시키려면, 다음을 수행해야 합니다.

1. 인스턴스에 대해 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있는 사용자 ID 및 이름으로 로그인 또는 접속하거나 인스턴스 소유자로 로그인하십시오.
2. 중지시키고자 하는 특정 데이터베이스에 연결된 모든 응용프로그램과 사용자를 표시하십시오. 중요한 응용프로그램이 실행되고 있지 않은지 확인하려면, 응용프로그램을 나열해야 합니다. 이에 대해 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 필요합니다.
3. 모든 응용프로그램과 사용자를 데이터베이스에서 강제로 중지하십시오. 사용자를 강제로 중지하려면 SYSADM 및 SYSCTRL 권한이 필요합니다.

제한사항:

db2stop 명령은 서버에서만 실행될 수 있습니다. 이 명령이 실행 중일 때에는 어떠한 데이터베이스 연결도 허용되지 않지만, 인스턴스 첨부가 있을 경우에는 인스턴스가 중지되기 전에 강제로 중지됩니다.

주: 명령행 처리기 세션이 인스턴스에 접속된 경우, **db2stop** 명령을 실행하기 전에 **종료** 명령을 실행하여 각 세션을 종료해야 합니다. **db2stop** 명령은 DB2INSTANCE 환경 변수로 정의된 인스턴스를 중지시킵니다.

프로시저:

다음 두 방법 중 하나를 사용하여 인스턴스를 중지시키십시오.

1. 제어 센터를 사용하여 인스턴스를 중지시키십시오.

   1. **인스턴스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
   2. 중지시키려면 각 인스턴스를 누르십시오.
   3. 인스턴스를 마우스 오른쪽 단추로 선택한 후, 자동(팝업) 메뉴로부터 **중지**를 선택하십시오.
   4. 중지 확인 창에서 **확인**을 누르십시오.

2. 명령행을 사용하여 인스턴스를 중지시키려면, 다음을 입력하십시오.

   ```
   db2stop
   ```

db2stop 명령을 사용하여 파티션된 데이터베이스 환경 내의 개별 파티션을 중지 또는 삭제할 수 있습니다. 파티션된 데이터베이스에서 작업할 때 다음을 사용하여 논리적 파티션을 삭제하고자 할 경우 다음을 참조하십시오.

```
db2stop drop nodenum <0>
```

해당 데이터베이스에 액세스하려는 사용자가 없는지 확인해야 합니다. 해당 데이터베이스에 액세스하려는 사용자가 있으면 오류 메시지 SQL6030N을 수신하게 됩니다.

관련 참조:

- *Command Reference*의 『db2stop - Stop DB2 Command』

- *Command Reference*의 『TERMINATE Command』

Windows에서 인스턴스 중지

데이터베이스 관리 프로그램의 현재 인스턴스를 중지해야 하는 경우도 있습니다.

전제조건:

시스템에서 인스턴스를 중지시키려면, 다음을 수행해야 합니다.

1. DB2 Universal Database™(DB2 UDB) 서비스를 중지하는 사용자 어카운트는 Windows 운영 체제에서 정의된 올바른 특권이 있어야 합니다. 사용자 어카운트는 Administrators, Server Operators 또는 Power Users 그룹의 구성원이 될 수 있습니다.
2. 중지시키고자 하는 특정 데이터베이스에 연결된 모든 응용프로그램과 사용자를 표시하십시오. 중요한 응용프로그램이 실행되고 있지 않은지 확인하려면, 응용프로그램을 나열해야 합니다. 이에 대해 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 필요합니다.
3. 모든 응용프로그램과 사용자를 데이터베이스에서 강제로 중지하십시오. 사용자를 강제로 중지하려면 SYSADM 및 SYSCTRL 권한이 필요합니다.

제한사항:

db2stop 명령은 서버에서만 실행될 수 있습니다. 이 명령이 실행 중일 때에는 어떠한 데이터베이스 연결도 허용되지 않지만, 인스턴스 첨부가 있을 경우에는 DB2 UDB가 중지되기 전에 강제로 중지됩니다.

주: 명령행 처리기 세션이 인스턴스에 접속된 경우, **db2stop** 명령을 실행하기 전에 **종료** 명령을 실행하여 각 세션을 종료해야 합니다. **db2stop** 명령은 DB2INSTANCE 환경 변수로 정의된 인스턴스를 중지시킵니다.

프로시저:

시스템에서 인스턴스를 중지시키려면, 다음 메소드 중 하나를 사용하십시오.

- **db2stop**
- 제어 센터를 사용하여 서비스 중지

1. **인스턴스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 중지시키려면 각 인스턴스를 누르십시오.
3. 인스턴스를 마우스 오른쪽 단추로 선택한 후, 자동(팝업) 메뉴로부터 **중지**를 선택하십시오.
4. 중지 확인 창에서 **확인**을 누르십시오.

- 『NET STOP』 명령을 사용하여 중지하십시오.
- 응용프로그램 내에서 인스턴스를 중지하십시오.

파티션된 데이터베이스 환경에서 DB2 UDB를 사용 중일 경우, 각 데이터베이스 파티션 서버는 서비스로서 시작됩니다. 각 서비스를 중지해야 합니다.

관련 참조:

- *Command Reference*의 『db2stop - Stop DB2 Command』

데이터베이스 작성 준비

실제 데이터베이스를 작성하기 전에 수행해야 할 작업의 일부로 많은 개념과 태스크를 고려해야 합니다. 이 개념과 태스크에는 데이터베이스 설계 및 데이터베이스 작업에 필요한 인스턴스, 디렉토리 및 기타 지원 파일의 설정 등이 포함됩니다. 여기서는 다음 주제를 다룹니다.

- 논리 및 실제 데이터베이스 특성 설계
- 인스턴스 작성
- 환경 변수 및 프로파일 레지스트리
- DB2 Administration Server
- 노드 구성 파일 작성
- 데이터베이스 구성 파일 작성
- FCM(Fast Communication Manager) 통신

논리 및 실제 데이터베이스 특성 설계

데이터베이스를 작성하기 전에, 데이터베이스 설계에 대한 논리 및 실제 데이터베이스를 작성해야 합니다. *관리 안내서: 계획*을 참조하십시오.

인스턴스 작성

인스턴스는 데이터베이스를 카탈로그화하고 구성 매개변수를 설정하는 논리 데이터베이스 관리 프로그램 환경입니다. 필요에 따라, 둘 이상의 인스턴스를 작성할 수 있습니다. 다중 인스턴스를 사용하여 다음을 수행할 수 있습니다.

- 개발 환경에 대해 하나의 인스턴스를 사용하고 프로덕션 환경에 대해 또다른 인스턴스를 사용합니다.
- 특정 환경에 맞게 인스턴스를 조정합니다.
- 관련 정보에 대한 액세스를 제한합니다.
- 각 인스턴스에 대한 SYSADM, SYSCTRL 및 SYSMAINT 권한의 지정을 제어합니다.
- 각 인스턴스에 대해 데이터베이스 관리 프로그램 구성을 최적화합니다.
- 인스턴스 실패의 영향을 제한합니다. 인스턴스가 실패하는 이벤트에서 하나의 인스턴스만 영향을 받습니다. 기타 인스턴스는 계속 정상적으로 기능할 수 있습니다.

다중 인스턴스에는 몇 가지 단점이 있습니다.

- 각 인스턴스에 추가 시스템 자원(가상 메모리 및 디스크 스페이스)이 필요합니다.
- 관리할 인스턴스가 추가되므로 더 많은 관리 작업이 필요합니다.

인스턴스 디렉토리는 데이터베이스 인스턴스에 속하는 모든 정보를 저장합니다. 일단 인스턴스 디렉토리가 작성되면, 위치를 변경할 수 없습니다. 디렉토리에는 다음이 들어 있습니다.

- 데이터베이스 관리 프로그램 구성 파일
- 시스템 데이터베이스 디렉토리
- 노드 디렉토리
- 노드 구성 파일(db2nodes.cfg)
- 예외, 레지스트리 덤프 또는 DB2® Universal Database(DB2 UDB) 프로세스에 대한 호출 스택과 같은 디버깅 정보가 들어 있는 기타 파일

UNIX® 운영 체제에서 인스턴스 디렉토리는 INSTHOME/sqllib 디렉토리에 위치하며, 여기서, INSTHOME은 인스턴스 소유자의 홈 디렉토리입니다.

Windows® 운영 체제, 인스턴스 디렉토리는 DB2 UDB가 설치된 /sqllib 서브디렉토리에 위치합니다.

파티션된 데이터베이스 시스템에서 인스턴스 디렉토리는 인스턴스에 속하는 모든 데이터베이스 파티션 서버 사이에서 공유됩니다. 그러므로 인스턴스 디렉토리는 인스턴스에 있는 모든 머신이 액세스할 수 있는 네트워크 공유 드라이브에서 작성되어야 합니다.

설치 프로시저의 일부로서, 『DB2』라고 하는 DB2 UDB의 초기 인스턴스를 작성할 수 있습니다. UNIX에서 초기 인스턴스는 이름 지정 규칙 지침 내에서 원하는 것을 지정할 수 있습니다. 인스턴스 이름은 디렉토리 구조 설정에 사용됩니다.

이 인스턴스를 즉시 사용할 수 있도록 지원하려면, 설치 중에 다음을 설정합니다.

- 환경 변수 DB2INSTANCE를 『DB2』로 설정합니다.
- DB2 레지스트리 변수 DB2INSTDEF를 『DB2』로 설정합니다.

UNIX에서 디폴트값은 이름 지정 규칙 지침 내에서 원하는 것을 지정할 수 있습니다.

Windows에서는 인스턴스 이름이 서비스 이름과 동일하므로 충돌하지 않습니다. 서비스를 작성하려면 올바른 권한이 있어야 합니다.

이들 설정은 『DB2』를 디폴트 인스턴스로 설정합니다. 디폴트로 사용되는 인스턴스를 변경할 수 있지만, 먼저 추가 인스턴스를 작성해야 합니다.

DB2 UDB를 사용하기 전에, 각 사용자에 대한 데이터베이스 환경은 인스턴스에 액세스하고 DB2 UDB 프로그램을 실행할 수 있도록 갱신되어야 합니다. 이는 모든 사용자에게 적용됩니다(관리 사용자 포함).

UNIX 운영 체제에서 데이터베이스 환경을 설정할 수 있도록 샘플 스크립트 파일이 제공됩니다. 파일은 Bourne 또는 Korn 쉘의 경우 db2profile이고, C 쉘의 경우 db2cshrc입니다. 이들 스크립트는 인스턴스 소유자의 홈 디렉토리 아래에 있는 sqllib 서브디렉토리에 위치해 있습니다. 인스턴스 소유자 또는 인스턴스의 SYSADM 그룹에 속하는 모든 사용자는 인스턴스의 모든 사용자를 위해 스크립트를 사용자 정의할 수 있습니다. 또한, 스크립트는 각 사용자를 위해 복사되고 사용자 정의할 수도 있습니다.

샘플 스크립트에는 다음을 수행하기 위한 명령문이 들어 있습니다.

- 인스턴스 소유자 홈 디렉토리의 sqllib 서브디렉토리 아래에 있는 bin, adm 및 misc 서브디렉토리를 기존의 검색 경로에 추가하여 사용자의 PATH를 갱신합니다.
- DB2INSTANCE 환경 변수를 인스턴스 이름으로 설정합니다.

관련 개념:

- 21 페이지의 『UNIX 운영 체제에서 다중 인스턴스』
- 22 페이지의 『Windows 운영 체제의 다중 인스턴스』

태스크 관련:

- 27 페이지의 『인스턴스 추가』
- 24 페이지의 『인스턴스 작성시 UNIX 세부사항』
- 25 페이지의 『인스턴스 작성시 Windows 세부사항』
- 28 페이지의 『현재 인스턴스 설정』
- 29 페이지의 『인스턴스 자동 시작』
- 29 페이지의 『다중 인스턴스 동시 실행』
- 27 페이지의 『인스턴스 나열』
- 23 페이지의 『추가 인스턴스 작성』

UNIX에서 DB2 UDB 환경 자동 설정

디폴트로, 인스턴스를 작성할 때 데이터베이스 환경이 설정된 스크립트는 현재 세션이 지속하는 동안에만 사용자 환경에 영향을 줍니다. .profile 파일을 변경하여 사용자가 Bourne 또는 Korn 쉘을 통해 로그온할 때 db2profile 스크립트를 자동으로 실행하도록 할 수 있습니다. C 쉘의 사용자의 경우, .login 파일을 변경하여 db2shrc 스크립트 파일을 수행하도록 할 수 있습니다.

프로시저:

다음 명령문 중 하나를 .profile 또는 .login 스크립트 파일에 추가하십시오.

- 하나의 스크립트 버전을 공유하는 사용자의 경우, 다음을 추가하십시오.

```
. INSTHOME/sqllib/db2profile      (Bourne 또는 Korn 쉘의 경우)
source INSTHOME/sqllib/db2cshrc   (C 쉘의 경우)
```

  여기서, INSTHOME은 사용할 인스턴스의 홈 디렉토리입니다.

- 홈 디렉토리에 사용자 정의된 스크립트 버전이 있는 사용자의 경우, 다음을 추가하십시오.

```
. USERHOME/db2profile          (Bourne 또는 Korn 쉘의 경우)
source USERHOME/db2cshrc       (C 쉘에서)
```

  여기서, USERHOME은 사용자의 홈 디렉토리입니다.

태스크 관련:

- 20 페이지의 『UNIX에서 DB2 UDB 환경 수동 설정』

UNIX에서 DB2 UDB 환경 수동 설정

프로시저:

사용할 인스턴스를 선택하려면, 다음 명령문 중 하나를 명령 프롬프트에 입력하십시오. 마침표(.) 및 스페이스가 필요합니다.

- 하나의 스크립트 버전을 공유하는 사용자의 경우, 다음을 추가하십시오.

```
. INSTHOME/sqllib/db2profile      (Bourne 또는 Korn 쉘의 경우)
source INSTHOME/sqllib/db2cshrc   (C 쉘의 경우)
```

  여기서, INSTHOME은 사용할 인스턴스의 홈 디렉토리입니다.

- 홈 디렉토리에 사용자 정의된 스크립트 버전이 있는 사용자의 경우, 다음을 추가하십시오.

```
. USERHOME/db2profile          (Bourne 또는 Korn 쉘의 경우)
source USERHOME/db2cshrc       (C 쉘에서)
```

  여기서, USERHOME은 사용자의 홈 디렉토리입니다.

동시에 둘 이상의 인스턴스에 대해 작업하려면, 별도의 창에서 사용할 각 인스턴스에 대해 스크립트를 수행하십시오. 예를 들어, test 및 prod라고 하는 두 개의 인스턴스가 있고 홈 디렉토리가 /u/test 및 /u/prod라고 가정하십시오.

창 1의 경우,

- Bourne 또는 Korn 쉘에서 다음을 입력하십시오.

```
. /u/test/sqllib/db2profile
```

- C 쉘에서 다음을 입력하십시오.

```
source /u/test/sqllib/db2cshrc
```

창 2의 경우,

- Bourne 또는 Korn 쉘에서 다음을 입력하십시오.

```
. /u/prod/sqllib/db2profile
```

- C 쉘에서 다음을 입력하십시오.

```
source /u/prod/sqllib/db2cshrc
```

창 1을 사용하여 test 인스턴스에 대해 작업하고 창 2를 사용하여 prod 인스턴스에 대해 작업하십시오.

주: which db2 명령을 입력하여 검색 경로가 올바로 설정되었는지 확인하십시오. 이 명령은 CLP 실행 파일의 절대 경로를 리턴합니다. 이것이 인스턴스의 sqllib 디렉토리 아래에 위치해 있는지 검증하십시오.

태스크 관련:

- 19 페이지의『UNIX에서 DB2 UDB 환경 자동 설정』

UNIX 운영 체제에서 다중 인스턴스

UNIX® 운영 체제에서 하나 이상의 인스턴스가 있을 수 있습니다. 그러나 동시에 하나의 DB2® Universal Database(DB2 UDB) 인스턴스내에서만 작업할 수 있습니다.

주: 둘 이상의 인스턴스 간의 환경 충돌을 피하기 위해 각 인스턴스가 자체 고유의 홈 파일 시스템을 갖도록 해야 합니다. 홈 파일 시스템이 공유되면 오류가 리턴됩니다.

인스턴스 소유자 및 그룹, 즉 모든 인스턴스에 연관된 시스템 관리(SYSADM) 그룹입니다. 인스턴스 소유자 및 SYSADM 그룹은 인스턴스 작성 프로세스 중에 지정됩니다. 하나의 사용자 ID 또는 사용자 이름은 하나의 인스턴스에만 사용될 수 있습니다. 해당 사용자 ID 또는 사용자 이름을 *인스턴스 소유자*라고도 합니다.

각 인스턴스 소유자에게는 고유 홈 디렉토리가 있어야 합니다. 인스턴스 수행에 필요한 모든 파일은 인스턴스 소유자의 사용자 ID 또는 사용자 이름의 홈 디렉토리에 작성됩니다.

시스템에서 인스턴스 소유자의 사용자 ID 또는 사용자 이름을 제거해야 하는 경우, 잠재적으로 인스턴스에 연관된 파일을 유실하고 이 인스턴스에 저장된 데이터에 액세스할 수 없게 됩니다. 이러한 이유로, 인스턴스 소유자의 사용자 ID 또는 사용자 이름을 DB2 UDB 수행에만 사용하도록 하는 것이 바람직합니다.

인스턴스 소유자의 1차 그룹 또한 중요합니다. 이 1차 그룹은 인스턴스에 대해 자동으로 시스템 관리 그룹이 되고 인스턴스에 대한 SYSADM 권한을 얻습니다. 인스턴스 소유자의 1차 그룹의 구성원인 기타 사용자 ID 또는 사용자 이름도 이 레벨의 권한을 얻습니다. 이러한 이유로, 인스턴스 소유자의 사용자 ID 또는 사용자 이름을 인스턴스 관

리를 위해 예약해 놓은 1차 그룹에 지정할 수 있습니다 (또한, 1차 그룹을 인스턴스 소유자의 사용자 ID 또는 사용자 이름에 지정하도록 하십시오. 그렇지 않으면, 시스템 디폴트 1차 그룹이 사용됩니다).

이미 인스턴스에 대해 시스템 관리 그룹으로 만들고 싶은 그룹이 있으면, 인스턴스 소유자의 사용자 ID 또는 사용자 이름을 작성할 때 이 그룹을 1차 그룹으로 지정하기만 하면 됩니다. 인스턴스에서 기타 사용자에게 관리 권한을 부여하려면, 시스템 관리 그룹으로 지정된 그룹에 사용자를 추가하십시오.

인스턴스 간에 SYSADM 권한을 분리하려면, 각 인스턴스 소유자의 사용자 ID 또는 사용자 이름이 다른 1차 그룹을 사용하십시오. 그러나 다중 인스턴스에 대해 공통 SYSADM 권한이 있도록 선택하는 경우, 다중 인스턴스에 대해 동일한 1차 그룹을 사용할 수 있습니다.

태스크 관련:

• 24 페이지의『인스턴스 작성시 UNIX 세부사항』

Windows 운영 체제의 다중 인스턴스

동일한 머신에서 DB2® Universal Database(DB2 UDB)의 다중 인스턴스를 실행할 수도 있습니다. DB2 UDB의 각 인스턴스는 자체 고유의 데이터베이스를 유지보수하며 자체 고유의 데이터베이스 관리 프로그램을 가집니다.

DB2 UDB의 인스턴스는 다음으로 구성됩니다.

- 인스턴스를 나타내는 Windows® 서비스. 이 서비스의 이름은 인스턴스 이름과 동일합니다. 서비스의 표시 이름(서비스 패널에서)은『DB2 - 』접두부가 있는 인스턴스 이름입니다. 예를 들어, 이름이 DB2인 인스턴스의 경우, 표시 이름이『DB2 - DB2』인『DB2』라는 Windows 서비스가 존재합니다.

  주: Windows 98, Windows ME 또는 클라이언트 인스턴스의 경우에는 Windows 서비스가 작성되지 않습니다.

- 인스턴스 디렉토리. 이 디렉토리에는 인스턴스와 연관된 데이터베이스 관리 프로그램 구성 파일, 시스템 데이터베이스 디렉토리, 노드 디렉토리, DCS 데이터베이스 디렉토리, 모든 진단 로그 및 덤프 파일이 포함됩니다. 인스턴스 디렉토리는 기본적으로 SQLLIB 디렉토리 내부의 서브디렉토리이며 인스턴스 이름과 동일한 이름을 가집니다. 예를 들어,『DB2』인스턴스의 인스턴스 디렉토리는 `C:\SQLLIB\DB2`이며, 여기서, `C:\SQLLIB`는 DB2 UDB가 설치된 장소입니다. 레지스트리 변수 DB2INSTPROF를 사용하여 인스턴스 디렉토리의 디폴트 위치를 변경할 수 있습니다. DB2INSTPROF 레지스트리 변수를 다른 위치로 설정하면, DB2INSTPROF가 가리키는 디렉토리 아래에 인스턴스 디렉토리가 작성됩니다. 예를 들어, `DB2INSTPROF=D:\DB2PROFS`일 경우, 인스턴스 디렉토리는 `D:\DB2PROFS\DB2`가 됩니다.

- HKEY_LOCAL_MACHINE\SOFTWARE\ IBM®\DB2\PROFILES\<instance_name> 아래의 레지스트리 키. 모든 인스턴스 레벨 레지스트리 변수가 여기에 작성됩니다.

다중 DB2 UDB 인스턴스를 동시에 실행할 수 있습니다. 인스턴스에 대한 작업을 수행하려면, 해당 인스턴스에 대해 명령을 발행하기 전에 DB2INSTANCE 환경 변수를 인스턴스의 이름으로 설정해야 합니다.

하나의 인스턴스가 다른 인스턴스의 데이터베이스에 액세스하지 못하도록 하기 위해, 인스턴스의 데이터베이스 파일은 인스턴스 이름과 동일한 이름의 디렉토리 아래에 작성됩니다. 예를 들어, 드라이브 C:에 인스턴스 DB2에 대한 데이터베이스를 작성할 때, 데이터베이스 파일은 C:\DB2라는 디렉토리 내에 작성됩니다. 마찬가지로, 드라이브 C:에 인스턴스 TEST에 대한 데이터베이스를 작성할 때, 데이터베이스 파일은 C:\TEST라는 디렉토리 내에 작성됩니다.

관련 개념:

- *데이터 복구 및 고가용성 안내 및 참조서*의 『고가용성』

태스크 관련:

- 25 페이지의 『인스턴스 작성시 Windows 세부사항』

추가 인스턴스 작성

인스턴스는 DB2 Universal Database™(DB2 UDB) 설치의 일부로서 작성되나, 비즈니스상 추가 인스턴스를 작성해야 하는 경우도 있습니다.

전제조건:

Windows에서 관리 그룹에 속하거나 UNIX 플랫폼에서 root 권한이 있으면 추가적 DB2 UDB 인스턴스를 추가할 수 있습니다. 인스턴스를 추가하는 머신은 인스턴스 소유 머신(노드 0)이 됩니다. DB2 관리 서버가 있는 머신에서 인스턴스를 추가하십시오.

프로시저:

명령행을 사용하여 인스턴스를 추가하려면, 다음을 입력하십시오.

```
db2icrt <instance_name>
```

또다른 DB2 UDB 인스턴스를 추가하기 위해 **db2icrt** 명령을 사용할 때에는 인스턴스 소유자의 로그인 이름을 제공하고, 인스턴스의 인증 유형을 선택적으로 지정해야 합니다. 인증 유형은 해당 인스턴스에서 작성된 모든 데이터베이스에 적용됩니다. 인증 유형은 사용자 인증이 일어나는 곳의 명령문입니다.

DB2INSTPROF 환경 변수를 사용하여 인스턴스 디렉토리의 위치를 DB2PATH로 변경할 수 있습니다. 인스턴스 디렉토리에 대해 쓰기 액세스가 필요합니다. DB2PATH 이외의 경로에 디렉토리를 작성하려면 **db2icrt** 명령을 입력하기 *전에* DB2INSTPROF를 설정해야 합니다.

DB2 Universal Database Enterprise - Server Edition(ESE)의 경우에는 또한 파티션된 데이터베이스 시스템인 새 인터페이스를 추가 중임을 선언해야 합니다. 추가로 둘 이상의 파티션이 있는 ESE 인스턴스에 대해 작업 중이고 FCM(Fast Communication Manager)에 대해 작업 중이면, 인스턴스 작성시 추가의 TCP/IP 포트를 정의하여 파티션 간에 다중 연결을 수행할 수 있습니다. 예를 들어, Windows 운영 체제의 경우 **db2icrt** 명령을 **-r <port range>** 매개변수와 함께 사용하십시오. 포트 범위는 다음과 같습니다.

```
-r:<base_port,end_port>
```

여기서, base_port는 FCM에서 사용할 수 있는 첫 번째 포트이며, end_port는 FCM에서 사용할 수 있는 포트 번호 범위의 마지막 포트입니다.

관련 개념:

- 246 페이지의 『서버에 대한 인증 방법』
- 252 페이지의 『리모트 클라이언트에 대한 인증 고려사항』

관련 참조:

- *Command Reference*의 『db2icrt - Create Instance Command』

인스턴스 작성시 UNIX 세부사항

UNIX 운영 체제에서 작업할 때, **db2icrt** 명령에는 다음과 같은 선택적 매개변수가 있습니다.

- -h 또는 -?

  이 매개변수는 명령에 대한 도움말 메뉴를 표시할 때 사용됩니다.

- -d

  이 매개변수는 문제점 판별 중에 사용할 디버그 모드를 설정합니다.

- -a AuthType

  이 매개변수는 인스턴스에 대한 인증 유형을 지정합니다. 유효한 인증 유형은 SERVER, SERVER_ENCRYPT 또는 CLIENT입니다. 지정하지 않을 경우, DB2 Universal Database™(DB2 UDB) 서버가 설치되면 디폴트값은 SERVER입니다. 그렇지 않으면, CLIENT로 설정됩니다.

주:

1. 인스턴스의 인증 유형은 인스턴스에 속하는 모든 데이터베이스에 적용됩니다.
2. UNIX 운영 체제에서 인증 유형 DCE는 유효하지 않습니다.

- -u FencedID

  이 매개변수는 분리(fenced) 사용자 정의 함수(UDF) 및 스토어드 프로시저가 실행되는 사용자입니다. 이는 DB2 UDB 클라이언트 또는 DB2 UDB Application Development Client를 설치할 때는 필요하지 않습니다. 기타 DB2 UDB 제품의 경우, 필수 매개변수입니다.

  주: FencedID는 『root』 또는 『bin』이 될 수 없습니다.

- -p PortName

  이 매개변수는 사용될 TCP/IP 서비스 이름 또는 포트 번호를 지정합니다. 그런 다음, 이 값은 인스턴스의 모든 데이터베이스에 대해 인스턴스의 데이터베이스 구성 파일에 설정됩니다.

- -s InstType

  서로 다른 유형의 인스턴스를 작성할 수 있게 합니다. 유효한 인스턴스 유형은 `ese`, `wse`, `client` 및 `standalone`입니다.

예:

- DB2 UDB 서버에 대해 인스턴스를 추가하려면, 다음 명령을 사용할 수 있습니다.

  ```
  db2icrt -u db2fenc1 db2inst1
  ```

- DB2 Connect Enterprise Edition만 설치한 경우, 인스턴스 이름을 분리 ID로 사용할 수도 있습니다.

  ```
  db2icrt -u db2inst1 db2inst1
  ```

- DB2 UDB 클라이언트에 대해 인스턴스를 추가하려면, 다음 명령을 사용할 수 있습니다.

  ```
  db2icrt db2inst1 -s client -u fencedID
  ```

DB2 UDB 클라이언트 인스턴스는 워크스테이션을 다른 데이터베이스 서버에 연결하려고 할 때 작성되며 해당 워크스테이션에서 로컬 데이터베이스가 필요하지 않습니다.

관련 참조:

- *Command Reference*의 『db2icrt - Create Instance Command』

인스턴스 작성시 Windows 세부사항

Windows 운영 체제에서 작업할 때, **db2icrt** 명령에는 다음과 같은 선택적 매개변수가 있습니다.

- -s InstType

  서로 다른 유형의 인스턴스를 작성할 수 있게 합니다. 유효한 인스턴스 유형은 ese, wse, client 및 standalone입니다.

- -p:InstProf_Path

  이것은 다른 인스턴스 프로파일 경로를 지정하기 위한 선택적 매개변수입니다. 경로를 지정하지 않으면, 인스턴스 디렉토리는 SQLLIB 디렉토리 아래에서 작성되고 주어진 공유 이름인 DB2는 인스턴스 이름에 연결됩니다. 읽기 및 쓰기 사용 권한은 도메인에 있는 모든 사용자에게 자동으로 권한 부여됩니다. 사용 권한을 변경하여 디렉토리에 대한 액세스를 제한할 수 있습니다.

  다른 인스턴스 프로파일 경로를 지정하면, 공유 드라이브 또는 디렉토리를 작성해야 합니다. 사용 권한이 변경되지 않았다면, 이를 통해 도메인의 모든 사용자가 인스턴스를 직접 액세스하는 기회를 허용합니다.

- -u:username,password

  파티션된 데이터베이스 환경을 작성할 때, DB2 Universal Database 서비스의 도메인/사용자 어카운트 이름과 암호를 선언해야 합니다.

- -r:base_port,end_port

  이것은 FCM(Fast Communication Manager)의 TCP/IP 포트 범위를 지정하기 위한 선택적 매개변수입니다. TCP/IP 포트 범위를 지정하는 경우, 포트 범위가 파티션된 데이터베이스 시스템의 모든 머신에서 사용 가능해야 합니다.

Windows용 DB2 Universal Database (DB2 UDB) Enterprise Server Edition에 대한 예는 다음과 같습니다.

```
db2icrt inst1 -s ese
   -p:\\machineA\db2mpp
   -u:<user account name>,<password> -r:9010,9015
```

주: 서비스 어카운트를 변경하려면, 즉 제품 설치 중 첫 번째 인스턴스가 작성될 때 작성된 디폴트 서비스를 더이상 사용하지 않으려면, 그 인스턴스를 작성하는데 사용된 도메인/사용자 어카운트 이름에 다음과 같은 고급 권한을 부여해야 합니다.

- 운영 체제의 일부로 작동시키십시오.
- 토큰 오브젝트를 작성하십시오.
- 할당량을 늘리십시오.
- 서비스로 로그온하십시오.
- 프로세스 레벨 토큰을 바꾸십시오.
- 메모리에서 페이지를 잠그십시오.

해당 인스턴스는 공유 드라이브에 액세스하고 사용자 어카운트를 인증하며 DB2 UDB를 Windows 서비스로 실행하기 위해 이러한 사용자 권한이 필요합니다. 『메모리에서 페이지 잠금』 권한은 AWE(Address Windowing Extensions) 지원을 위해 필요합니다.

관련 참조:

- *Command Reference*의 『db2icrt - Create Instance Command』

인스턴스 추가

프로시저:

추가 인스턴스를 작성했으면, 제어 센터에 해당 인스턴스의 레코드를 추가해야만 제어 센터에서 그 인스턴스에 대한 작업을 수행할 수 있습니다.

또다른 인스턴스를 추가하려면, 다음 단계를 수행하십시오.

1. 관리자 권한이 있거나 Administrators 그룹에 속하는 사용자 ID 또는 이름으로 로그온하십시오.
2. 인스턴스를 추가하려면, 다음 방법 중 하나를 사용하십시오.

   제어 센터를 사용하려면, 다음을 수행하십시오.

1. 원하는 시스템의 **인스턴스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 인스턴스 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **추가**를 선택하십시오.
3. 정보를 완료하고, **적용**을 누르십시오.

관련 개념:

- 17 페이지의 『인스턴스 작성』

태스크 관련:

- 27 페이지의 『인스턴스 나열』

인스턴스 나열

프로시저:

제어 센터를 사용하여 시스템에서 사용할 수 있는 모든 인스턴스 목록을 확보하려면, 다음을 수행하십시오.

1. **인스턴스** 폴더가 나올 때까지 오브젝트 트리를 확장하십시오.
2. 인스턴스 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **추가**를 선택하십시오.
3. 인스턴스 추가 창에서 **새로 고침**을 누르십시오.
4. 데이터베이스 인스턴스를 보려면 드롭 다운 화살표를 누르십시오.
5. 창을 나가려면 **취소**를 누르십시오.

시스템에서 사용할 수 있는 모든 인스턴스 목록을 확보하려면, 다음을 입력하십시오.

```
db2ilist
```

현재 세션에 적용되는 인스턴스를 판별하려면(지원되는 Windows 플랫폼에서) 다음을 사용하십시오.

```
set db2instance
```

관련 참조:

- *Command Reference*의 『db2ilist - List Instances Command』

현재 인스턴스 설정

프로시저:

인스턴스의 데이터베이스 관리 프로그램을 시작하거나 중지하는 명령을 실행할 경우, DB2 Universal Database™(DB2 UDB)가 명령을 현재 인스턴스에 적용합니다. DB2 UDB는 현재 인스턴스를 다음과 같이 판별합니다.

- DB2INSTANCE 환경 변수가 현재 세션에 대해 설정되면, 값은 현재 인스턴스가 됩니다. DB2INSTANCE 환경 변수를 설정하려면, 다음을 입력하십시오.

  ```
  set db2instance=<new_instance_name>
  ```

- DB2INSTANCE 환경 변수가 현재 세션에 대해 설정되지 않으면, DB2 UDB는 시스템 환경 변수에서 DB2 INSTANCE 환경 변수에 대한 설정을 사용합니다. Windows NT에서 시스템 환경 변수는 시스템 환경에서 설정됩니다. Windows 9x에서는 `autoexec.bat` 파일에 설정됩니다.
- DB2INSTANCE 환경 변수가 전혀 설정되지 않으면, DB2 UDB는 DB2INSTDEF 레지스트리 변수를 사용합니다.

  레지스트리의 전역 레벨에서 DB2INSTDEF 레지스트리 변수를 설정하려면, 다음을 입력하십시오.

  ```
  db2set db2instdef=<new_instance_name> -g
  ```

현재 세션에 적용되는 인스턴스를 판별하려면, 다음을 입력하십시오.

```
db2 get instance
```

태스크 관련:

- 33 페이지의『레지스트리 및 환경 변수 선언』

인스턴스 자동 시작

프로시저:

Windows 운영 체제에서 설치 시 작성되는 DB2 Universal Database™(DB2 UDB) 인스턴스는 디폴트로 자동 시작으로 설정됩니다. **db2icrt**를 사용하여 작성한 인스턴스는 수동 시작으로 설정됩니다. 시작 유형을 변경하려면 서비스 패널에서 DB2 UDB 서비스의 등록 정보를 변경해야 합니다.

UNIX 운영 체제에서 시스템이 재시작할 때마다 인스턴스가 자동으로 시작되게 하려면, 다음 명령을 입력하십시오.

```
db2iauto -on <instance name>
```

여기서, <instance name>은 인스턴스의 로그인 이름입니다.

UNIX 운영 체제에서 시스템이 재시작할 때마다 인스턴스가 자동으로 시작되지 않도록 하려면, 다음 명령을 입력하십시오.

```
db2iauto -off <instance name>
```

여기서, <instance name>은 인스턴스의 로그인 이름입니다.

관련 개념:

- 17 페이지의『인스턴스 작성』

관련 참조:

- *Command Reference*의『db2iauto - Auto-start Instance Command』

다중 인스턴스 동시 실행

프로시저:

제어 센터를 사용하여 동시에 다중 인스턴스를 수행하십시오.

1. **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 마우스 오른쪽 단추로 인스턴스를 누른 후, 팝업 메뉴에서 **시작**을 선택하십시오.
3. 동시에 수행하려는 모든 인스턴스를 시작할 때까지 2단계를 반복하십시오.

(Windows 전용:) 다음 명령행을 사용하여 현재 다중 인스턴스를 수행하십시오.

1. 다음을 입력하여 시작하려면 DB2INSTANCE 변수를 원하는 다른 인스턴스의 이름으로 설정하십시오.

   ```
   set db2instance=<another_instName>
   ```

2. **db2start** 명령을 입력하여 인스턴스를 시작하십시오.

관련 개념:

- 6 페이지의 『데이터베이스 관리 프로그램의 다중 인스턴스』

태스크 관련:

- 24 페이지의 『인스턴스 작성시 UNIX 세부사항』
- 25 페이지의 『인스턴스 작성시 Windows 세부사항』
- 23 페이지의 『추가 인스턴스 작성』

라이센스 관리

DB2® Universal Database(DB2 UDB) 제품에 대한 라이센스 관리는 제품에 대한 온라인 인터페이스의 제어 센터 내에 있는 라이센스 센터를 통해 주로 완료됩니다. 라이센스 센터에서 라이센스에 관한 정보, 통계, 등록된 사용자 그리고 설치된 각 제품의 현재 사용자를 검사할 수 있습니다.

제어 센터를 사용할 수 없을 때에는 **db2licm** 라이센스 관리 도구가 기본 라이센스 기능을 수행합니다. 이 명령을 사용하여 로컬 시스템에 설치된 라이센스 및 규정을 추가, 제거, 나열 및 수정할 수 있습니다.

관련 참조:

- *Command Reference*의 『db2licm - License Management Tool Command』

환경 변수 및 프로파일 레지스트리

환경 변수 및 레지스트리 변수는 데이터베이스 환경을 제어합니다.

구성 지원 프로그램(**db2ca**)을 사용하여 구성 매개변수 및 레지스트리 변수를 구성할 수 있습니다.

DB2® Universal Database(DB2 UDB) 프로파일 레지스트리가 도입되기 전에는, Windows® 워크스테이션(예를 들어)에서 환경 변수를 변경하려면 환경 변수를 변경한 후 재부팅했어야 합니다. 이제 사용자의 환경은 DB2 UDB 프로파일 레지스트리에 저장된 레지스트리 변수에 의해 제어됩니다(몇 가지 예외사항이 있음). 주어진 인스턴스에 대해 시스템 관리(SYSADM) 권한이 있는 UNIX® 운영 체제 사용자는 해당 인스턴스에 대한 레지스트리 값을 갱신할 수 있습니다. Windows 사용자는 레지스트리 변수를 갱신하는데 SYSADM 권한이 없어도 됩니다. 재부팅하지 않고 레지스트리 변수를 갱신하려면 **db2set** 명령을 사용하십시오. 갱신된 정보는 즉시 프로파일 레지스트리에 저장됩니다. DB2 UDB 레지스트리는 변경이 이루어진 후 시작된 DB2 UDB 서버 인스턴스 및 DB2 UDB 응용프로그램에 갱신된 정보를 적용합니다.

레지스트리를 갱신할 때 변경사항은 현재 실행 중인 DB2 UDB 응용프로그램 또는 사용자에게 영향을 주지 않습니다. 갱신 후에 시작되는 응용프로그램은 새 값을 사용합니다.

주: DB2 UDB 환경 변수 DB2INSTANCE 및 DB2NODE는 운영 체제에 따라 DB2 UDB 프로파일 레지스트리에 저장되지 않을 수도 있습니다. 몇몇 운영 체제에서 이러한 환경 변수를 갱신하려면, **set** 명령을 사용하십시오. 이들 변경사항은 다음에 시스템이 재부팅될 때까지 적용됩니다. UNIX 플랫폼에서 **set** 명령 대신 **export** 명령이 사용됩니다.

프로파일 레지스트리를 사용하면 환경 변수를 중앙에서 제어할 수 있습니다. 다양한 프로파일을 통해 다양한 레벨의 지원이 제공됩니다. DB2 Administration Server를 사용하면, 환경 변수를 리모트로 관리할 수도 있습니다.

다음은 네 개의 프로파일 레지스트리입니다.

- DB2 UDB 인스턴스 레벨 프로파일 레지스트리. 대부분의 DB2 UDB 환경 변수는 이 레지스트리 내에 배치됩니다. 이 레지스트리에는 특정 인스턴스용 환경 변수 설정이 있습니다. 이 레벨에서 정의된 값은 전역 레벨에서 설정을 겹쳐씁니다.
- DB2 UDB 전역 레벨 프로파일 레지스트리. 특정 인스턴스에 대해 환경 변수가 설정되어 있지 않을 경우, 이 레지스트리가 사용됩니다. 이 레지스트리에는 머신 전체에 적용되는 환경 변수 설정이 있습니다. DB2 UDB ESE에는 각 머신마다 하나의 전역 레벨 프로파일이 있습니다.
- DB2 UDB 인스턴스 노드 레벨 프로파일 레지스트리. 이 레지스트리 레벨에는 다중 파티션 환경에 있는 임의 파티션(노드)에 고유한 변수 설정값이 들어 있습니다. 이 레벨에서 정의된 값은 인스턴스 및 전역 레벨의 설정값을 겹쳐씁니다.
- DB2 UDB 인스턴스 프로파일 레지스트리. 이 레지스트리에는 이 시스템이 인식하는 모든 인스턴스 이름 목록이 들어 있습니다. db2ilist를 실행하여 시스템에서 사용 가능한 모든 인스턴스의 전체 목록을 볼 수 있습니다.

DB2 UDB는 레지스트리 값 및 환경 변수를 검사하고, 다음 순서로 해결함으로써, 운영 환경을 구성합니다.

1. set 명령으로 설정된 환경 변수(또는 UNIX 플랫폼에서 export 명령)
2. 인스턴스 노드 레벨 프로파일(**db2set -i <instance name> <nodenum>** 명령 사용)로 설정된 레지스트리 값
3. 인스턴스 노드 레벨 프로파일(**db2set -i** 명령 사용)로 설정된 레지스트리 값
4. 전역 레벨 프로파일(**db2set -g** 명령 사용)로 설정된 전역 레벨 프로파일

인스턴스 레벨 프로파일 레지스트리

파티션된 데이터베이스 환경에 대해 작업할 때 몇 가지 UNIX 및 Windows 차이점이 있습니다. 이러한 차이점은 다음 예에서 보여줍니다.

『red』, 『white』 및 『blue』로 식별할 세 개의 실제 노드가 있는 파티션된 데이터베이스 환경이 있다고 가정합니다. UNIX 플랫폼에서 인스턴스 소유자가 노드에서 다음을 실행하는 경우.

```
db2set -i FOO=BAR
```

또는

```
db2set FOO=BAR        ('-i' is implied)
```

값 F00은 현재 인스턴스의 모든 노드(즉, 『빨간색』, 『흰색』 및 『파란색』)가 볼 수 있게 됩니다.

UNIX 플랫폼에서 인스턴스 레벨 프로파일 레지스트리는 sqllib 디렉토리의 텍스트 파일에 저장됩니다. 파티션된 데이터베이스 환경에서 sqllib 디렉토리는 모든 실제 노드들이 공유하는 파일 시스템에 있습니다.

Windows 플랫폼에서 사용자가 『red』에서 동일한 명령을 수행하면 값 F00은 현재 인스턴스의 『빨간색』에서만 볼 수 있습니다. DB2 UDB는 인스턴스 레벨 프로파일 레지스트리를 Windows 레지스트리에 저장합니다. 실제 노드들 간의 공유는 없습니다. 모든 실제 시스템에서 레지스트리 변수를 설정하려면 다음과 같이 『rah』 명령을 사용하십시오.

```
rah db2set -i FOO=BAR
```

rah는 『빨간색』, 『흰색』 및 『파란색』에서 db2set 명령을 리모트로 실행합니다.

인스턴스를 소유하지 않는 머신의 레지스트리 변수가 인스턴스 소유 머신의 레지스트리 변수를 참조하도록 구성하는 데 DB2REMOTEPREG를 사용할 수 있습니다. 그러면 인스턴스 소유 머신의 레지스트리 변수를 인스턴스의 모든 머신들이 공유하는 환경이 효과적으로 작성됩니다.

위에 표시된 예를 사용하고 『빨간색』이 소유 머신이라고 가정하면 다음을 수행하여 『흰색』 및 『파란색』 머신의 DB2REMOTEPREG를 『빨간색』의 레지스트리 변수를 공유하도록 설정합니다.

```
(빨간색의 경우) 아무 조치도 수행하지 않음
(흰색 및 파란색의 경우) db2set DB2REMOTEPREG=\\red
```

DB2REMOTEPREG의 설정은 설정한 후에 변경해서는 안됩니다.

다음은 REMOTEPREG의 작업 방식입니다.

DB2 UDB가 Windows에서 레지스트리 변수를 읽을 때 먼저 DB2REMOTEPREG 값을 읽습니다. DB2REMOTEPREG가 설정된 경우 머신 이름이 DB2REMOTEPREG 변수에 지정된 리모트 머신에서 레지스트리를 엽니다. 레지스트리 변수의 후속 읽기 및 갱신은 지정된 리모트 머신으로 경로 재지정됩니다.

리모트 레지스트리에 액세스하려면 목표 머신에서 리모트 레지스트리 서비스를 실행하고 있어야 합니다. 또한 사용자 로그온 어카운트와 모든 DB2 UDB 로그온 어카운트에는 리모트 레지스트리에 대한 충분한 액세스 권한이 있어야 합니다. 따라서 DB2REMOTEPREG를 사용하려면 도메인 어카운트에 필수 레지스트리 액세스 권한을 부여할 수 있도록 Windows 도메인 환경에서 작동해야 합니다.

Microsoft® Cluster Server(MSCS) 고려사항이 있습니다. MSCS 환경에서는 DB2REMOTEPREG를 사용해서는 안됩니다. 모든 머신이 동일한 MSCS 클러스터에 속하는 MSCS 구성에서 실행하고 있는 경우 레지스트리 변수는 클러스터 레지스트리에서 유지보수됩니다. 따라서 이미 동일한 MSCS 클러스터의 모든 머신들이 공유하고 있으며 이 경우 DB2REMOTEPREG를 사용할 필요가 없습니다.

파티션이 여러 MSCS 클러스터에 걸쳐 있는 다중 파티션 장애 복구 환경에서 실행하고 있는 경우 인스턴스 소유 머신이 클러스터 레지스트리에 상주하므로 DB2REMOTEPREG를 사용하여 인스턴스 소유 머신을 나타낼 수 없습니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 33 페이지의 『레지스트리 및 환경 변수 선언』

레지스트리 및 환경 변수 선언

프로시저:

db2set 명령은 레지스트리 값(및 환경 변수)의 로컬 선언을 지원합니다.

명령에 대한 도움말 정보를 표시하려면, 다음을 사용하십시오.

```
db2set ?
```

지원되는 모든 레지스트리 변수의 전체 세트를 나열하려면, 다음을 사용하십시오.

```
db2set -lr
```

현재 또는 디폴트 인스턴스에 대해 정의된 모든 레지스트리 변수를 나열하려면, 다음을 사용하십시오.

```
db2set
```

프로파일 레지스트리에 정의된 모든 레지스트리 변수를 나열하려면, 다음을 사용하십시오.

```
db2set -all
```

현재 또는 디폴트 인스턴스에서 레지스트리 변수 값을 표시하려면, 다음을 사용하십시오.

```
db2set registry_variable_name
```

모든 레벨에서 레지스트리 변수 값을 표시하려면, 다음을 사용하십시오.

```
db2set registry_variable_name -all
```

현재 또는 디폴트 인스턴스에서 레지스트리 변수를 변경하려면, 다음을 사용하십시오.

```
db2set registry_variable_name=new_value
```

인스턴스의 모든 데이터베이스에 대한 레지스트리 변수 디폴트값을 변경하려면, 다음을 사용하십시오.

```
db2set registry_variable_name=new_value
   -i instance_name
```

인스턴스의 특정 파티션에 대한 레지스트리 변수 디폴트를 변경하려면, 다음을 사용하십시오.

```
db2set registry_variable_name=new_value
   -i instance_name node_number
```

시스템의 모든 인스턴스에 대한 레지스트리 변수 디폴트값을 변경하려면, 다음을 사용하십시오.

```
db2set registry_variable_name=new_value -g
```

LDAP(Lightweight Directory Access Protocol)를 사용할 경우, LDAP에 다음과 같이 레지스트리 변수를 설정할 수 있습니다.

- LDAP의 사용자 레벨에서 레지스트리 변수를 설정하려면 다음을 사용하십시오.

  ```
  db2set -ul
  ```

- LDAP의 전역 레벨에서 레지스트리 변수를 설정하려면 다음을 사용하십시오.

  ```
  db2set -gl user_name
  ```

LDAP 환경에서 실행할 때, 범위가 디렉토리 파티션 또는 Windows NT 도메인에 속하는 모든 머신 및 모든 사용자에게 전역인 DB2 Universal Database™(DB2 UDB) 레지스트리 변수 값을 LDAP에 설정하는 것이 가능합니다. 현재 LDAP 전역 레벨에 설정할 수 있는 두 개의 DB2 UDB 레지스트리 변수는 DB2LDAP_KEEP_CONNECTION과 DB2LDAP_SEARCH_SCOPE입니다.

예를 들어, LDAP의 전역 레벨에서 검색 범위 값을 설정하려면, 다음을 사용하십시오.

```
db2set -gl db2ldap_search_scope = value
```

여기서, *value*는 『local』, 『domain』 또는 『global』일 수 있습니다.

주:

1. DB2 UDB profile.env 파일이 db2set 명령에 의해 동시에 갱신될 경우(즉, 동일한 시간에 또는 동일한 시간에 매우 근접하게), profile.env 파일의 크기가 0으로 감소됩니다. 또한 db2set -all의 값이 불일치하는 값을 표시합니다.
2. 머신 전역 레벨에서 DB2 UDB 레지스트리 변수를 설정하는데 사용되는 -g 옵션과 LDAP 전역 레벨에 특정한 -gl은 서로 다릅니다.
3. 사용자 레벨 레지스트리 변수는 LDAP 환경에서 실행 중일 때 Windows에서만 지원됩니다.
4. 사용자 레벨의 변수 설정에는 사용자 특정 변수 설정이 포함됩니다. 사용자 레벨에 대한 변경사항은 LDAP 디렉토리에 기록됩니다.
5. 『-i』, 『-g』, 『-gl』 및 『-ul』 매개변수는 같은 명령에서 동시에 사용할 수 없습니다.
6. 일부 변수는 디폴트로 항상 전역 레벨 프로파일에서만 사용됩니다. 이러한 변수는 인스턴스 또는 노드 레벨 프로파일에서 설정할 수 없습니다(예: DB2SYSTEM 및 DB2INSTDEF).
7. UNIX에서 인스턴스에 대한 레지스트리 값을 변경하려면, 시스템 관리 권한(SYSADM)이 있어야 합니다. root 권한이 있는 사용자만이 전역 레벨 레지스트리에 있는 매개변수를 변경할 수 있습니다.

인스턴스에 대한 레지스트리 변수를 전역 프로파일 레지스트리의 디폴트값으로 재설정하려면, 다음을 사용하십시오.

```
db2set -r registry_variable_name
```

인스턴스의 노드에 대한 레지스트리 변수를 전역 프로파일 레지스트리의 디폴트값으로 다시 설정하려면, 다음을 사용하십시오.

```
db2set -r registry_variable_name node_number
```

특정 레벨에서 변수 값을 삭제하려면, 같은 명령 구문을 사용하여 변수를 설정할 수 있지만 변수 값에 대해 아무것도 지정하지 마십시오. 예를 들어, 노드 레벨에서 변수의 설정을 삭제하려면, 다음을 입력하십시오.

```
db2set registry_variable_name= -i instance_name
node_number
```

변수 값이 더 높은 프로파일 레벨에서 정의된 경우, 이를 삭제하고 사용을 제한하려면, 다음을 입력하십시오.

```
db2set registry_variable_name= -null instance_name
```

이 명령은 사용자가 지정한 매개변수에 대한 설정을 삭제하고 상위 레벨의 프로파일이 이 변수 값을 변경하지 못하도록 제한할 수 있습니다(이 경우, DB2 UDB 전역 레벨 프로파일). 그러나 사용자가 지정한 변수는 더 낮은 레벨의 프로파일(이 경우, DB2 UDB 노드 레벨 프로파일)로 설정될 수는 있습니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 36 페이지의 『Windows에서 환경 변수 설정』
- 38 페이지의 『UNIX 시스템에서 환경 변수 설정』
- 373 페이지의 『LDAP 디렉토리 파티션 또는 도메인 검색』
- 376 페이지의 『LDAP 환경의 사용자 레벨에서 DB2 레지스트리 변수 설정』

Windows에서 환경 변수 설정

프로시저:

모든 특정 레지스트리 변수를 DB2 Universal Database™(DB2 UDB) 프로파일 레지스트리에 정의하는 것이 바람직합니다. DB2 UDB 변수가 레지스트리 외부에서 설정되면, 해당 변수를 리모트로 관리할 수 없게 되며 워크스테이션은 변수 값이 적용되도록 재부팅되어야 합니다.

Windows 운영 체제에는 프로파일 레지스트리 외부에서만 설정할 수 있는 하나의 시스템 환경 변수 DB2INSTANCE가 있으나, DB2INSTANCE를 설정하지 않아도 됩니다. DB2 UDB 프로파일 레지스트리 변수인 DB2INSTDEF를 전역 레벨 프로파일에서 설정하여 DB2INSTANCE가 정의되지 않은 경우 사용할 인스턴스 이름을 지정할 수 있습니다.

Windows 상의 DB2 UDB Enterprise Server Edition 서버에는 프로파일 레지스트리 외부에서만 설정할 수 있는 두 개의 시스템 환경 변수 DB2INSTANCE 및 DB2NODE가 있습니다. DB2INSTANCE를 설정할 필요는 없습니다. DB2 UDB 프로파일 레지스트리 변수인 DB2INSTDEF를 전역 레벨 프로파일에서 설정하여 DB2INSTANCE가 정의되지 않은 경우 사용할 인스턴스 이름을 지정할 수 있습니다.

DB2NODE 환경 변수는 머신 내에서 목표 논리 노드로 요청을 경로 지정하기 위해 사용됩니다. 이 환경 변수는 DB2 UDB 프로파일 레지스트리가 아닌 응용프로그램 또는 명령이 발행되는 세션에서 설정되어야 합니다. 이 변수를 설정하지 않을 경우, 머신에서 제로(0)로 정의되는 논리 노드가 목표 논리 노드의 디폴트가 됩니다.

환경 변수의 설정을 판별하려면, **echo** 명령을 사용하십시오. 예를 들어, DB2PATH 환경 변수의 값을 점검하려면 다음을 입력하십시오.

```
echo %db2path%
```

시스템 환경 변수를 설정하려면, 다음을 수행하십시오.

Windows 9x에서: `autoexec.bat` 파일을 편집한 후 변경사항이 적용되도록 시스템을 재부팅하십시오.

Windows에서: DB2 UDB 환경 변수 DB2INSTANCE 및 DB2NODE를 다음과 같이 설정할 수 있습니다(이 설명에서는 DB2INSTANCE 사용).

- (Windows NT 및 Windows 2000에서) **시작**, **설정**, **제어판**을 선택하십시오. (Windows XP 및 Windows 서버 2003에서) **시작 —> 제어판**을 선택하십시오.
- (Windows NT 및 Windows 2000에서) **시스템** 아이콘을 더블 클릭하십시오. (Windows XP 및 Windows 서버 2003에서) Windows 주제 및 현재 선택된 보기 유형에 따라, **성능** 및 **유지보수**를 선택해야만 **시스템** 아이콘 선택이 가능한 경우도 있습니다.
- (Windows NT에서) 시스템 제어판의 시스템 환경 변수 섹션에서 다음을 수행하십시오. (Windows 2000, Windows XP 및 Windows .NET에서) 시스템 등록 정보 창에서 **고급** 탭을 선택하고 환경 변수 단추를 누른 후 다음을 수행하십시오.
  1. DB2INSTANCE 변수가 없을 경우, 다음을 수행하십시오.
     a. (Windows NT에서) 시스템 환경 변수를 선택하십시오. (Windows 2000, Windows XP 및 Windows 서버 2003에서) **새로 만들기** 단추를 누르십시오.
     b. (Windows NT에서) *변수* 필드에 있는 이름을 DB2INSTANCE로 변경하십시오. (Windows 2000, Windows XP 및 Windows Server 2003에서) *변수 이름* 필드에 DB2INSTANCE를 입력하십시오.
     c. (Windows NT에서) *값* 필드를 인스턴스 이름으로 변경하십시오(예: db2inst). (Windows 2000, Windows XP 및 Windows 서버 2003에서) *변수값* 필드에 인스턴스 이름을 입력하십시오(예: db2inst).
  2. DB2INSTANCE 변수가 이미 있으면, 새 값을 추가하십시오.
     a. DB2INSTANCE 환경 변수를 선택하십시오.
     b. *값* 필드를 db2inst와 같은 인스턴스 이름으로 변경하십시오.
  3. (Windows NT에서) 설정을 선택하십시오. (Windows 2000, Windows XP 및 Windows 서버 2003에서) 확인을 선택하십시오.
  4. 확인을 선택하십시오.
  5. 이들 변경사항을 적용하려면 시스템을 재부팅하십시오.

주: DB2INSTANCE 환경 변수는 세션(프로세스) 레벨에서도 설정될 수 있습니다. 예를 들어, TEST라고 하는 두 번째 DB2 UDB 인스턴스를 시작하려면, 명령 창에서 다음 명령을 발행하십시오.

```
   set DB2INSTANCE=TEST
db2start
```

C 쉘에서 작업할 때, 명령 창에서 다음 명령을 발행하십시오.

```
   setenv DB2INSTANCE TEST
```

프로파일 레지스트리는 다음과 같이 배치됩니다.

- Windows 운영 체제 레지스트리의 DB2 UDB 인스턴스 레벨 프로파일 레지스트리는 다음 경로를 가집니다.

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES*instance_name*

  주: *instance_name*은 DB2 UDB 인스턴스의 이름입니다.
- Windows 레지스트리의 DB2 UDB 전역 레벨 프로파일 레지스트리는 다음 경로를 가집니다.

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\GLOBAL_PROFILE
- Windows 레지스트리의 DB2 UDB 인스턴스 노드 레벨 프로파일 레지스트리는 다음 경로를 가집니다.

  ...\SOFTWARE\IBM\DB2\PROFILES*instance_name*\NODES*node_number*

  주: *instance_name*과 *node_number*는 작업 중인 데이터베이스 파티션에 대해 고유합니다.
- 요구되는 DB2 UDB 인스턴스 프로파일 레지스트리가 없습니다. 시스템에 있는 DB2 UDB 인스턴스마다 해당 경로에 키가 작성됩니다.

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES*instance_name*

  PROFILES 키 아래에 있는 키를 카운팅함으로써 인스턴스 목록을 확보할 수 있습니다.

관련 개념:

- 47 페이지의 『DB2 Administration Server』

태스크 관련:

- 38 페이지의 『UNIX 시스템에서 환경 변수 설정』

UNIX 시스템에서 환경 변수 설정

프로시저:

모든 고유 레지스트리 변수를 DB2 UDB 프로파일 레지스트리에서 정의하는 것이 바람직합니다. DB2 UDB 변수가 레지스트리 외부에서 설정되면, 해당 변수를 리모트로 관리할 수 없습니다.

UNIX 운영 체제에서 시스템 환경 변수인 DB2INSTANCE를 설정해야 합니다.

스크립트 db2profile(Korn 쉘의 경우) 및 db2cshrc(Bourne 쉘 또는 C 쉘의 경우)는 데이터베이스 환경 설정을 돕기 위해 예로서 제공됩니다. insthome/sqllib에 이들 파일이 있습니다. 여기서, insthome은 인스턴스 소유자의 홈 디렉토리입니다.

이 스크립트는 다음 명령문을 포함합니다.

- 다음 디렉토리로 사용자 경로를 갱신합니다.

  - insthome/sqllib/bin
  - insthome/sqllib/adm
  - insthome/sqllib/misc
- 실행을 위해 DB2INSTANCE를 디폴트 로컬 instance_name으로 설정합니다.

주: PATH 및 DB2INSTANCE를 제외한 기타 모든 지원 변수는 DB2 UDB 프로파일 레지스트리에 설정되어야 합니다. DB2 UDB에서 지원하지 않는 변수를 설정하려면 스크립트 파일 userprofile 및 usercshrc에 이러한 변수를 정의하십시오.

인스턴스 소유자 또는 SYSADM 사용자는 인스턴스의 모든 사용자를 위해 이 스크립트를 사용자 정의할 수 있습니다. 선택적으로, 사용자는 스크립트를 복사하고 사용자 정의할 수 있으며, 스크립트를 직접 호출하거나 스크립트를 .profile 또는 .login 파일에 추가할 수 있습니다.

현재 세션에 대한 환경 변수를 변경하려면, 다음과 유사한 명령을 발행하십시오.

- Korn 쉘의 경우,

```
DB2INSTANCE=inst1
export DB2INSTANCE
```

- Bourne 쉘의 경우

```
export DB2INSTANCE=<inst1>
```

- C 쉘의 경우

```
setenv DB2INSTANCE <inst1>
```

DB2 UDB 프로파일 레지스트리가 적절히 관리되도록 하려면, UNIX 운영 체제에서는 다음 파일 소유권 규칙을 반드시 지켜야 합니다.

- DB2 UDB 인스턴스 레벨 프로파일 레지스트리 파일은 다음 위치에 배치됩니다.

  ```
  INSTHOME/sqllib/profile.env
  ```

  이 파일의 액세스 권한 및 소유권은 다음과 같아야 합니다.

  ```
  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env
  ```

  여기서, <db2inst1> 인스턴스 소유자 및 <db2iadm1> 인스턴스 소유자 그룹입니다.

  INSTHOME은 인스턴스 소유자의 홈 경로입니다.
- DB2 UDB 전역 레벨 프로파일 레지스트리는 다음 위치에 배치됩니다.
  - AIX, Solaris 운영 환경 및 Linux 운영 체제의 경우, /var/db2/<version_id>/default.env(여기서, <version_id>는 현재 버전임).
  - HP-UX 운영 체제용 /var/opt/db2/<version_id>/default.env(여기서, <version_id>는 현재 버전임)

  이 파일의 액세스 권한 및 소유권은 다음과 같아야 합니다.

```
-rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> default.env
```

전역 레지스트리 변수를 수정하려면 사용자가 root로서 로그온되어야 합니다.

- DB2 UDB 인스턴스 노드 레벨 프로파일 레지스트리는 다음 위치에 배치됩니다.

  ```
  INSTHOME/sqllib/nodes/<node_number>.env
  ```

  디렉토리와 이 파일의 액세스 권한 및 소유권은 다음과 같아야 합니다.

  ```
  drwxrwsr-w <Instance_Owner> <Instance_Owner_Group> nodes

  -rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> <node_number>.env
  ```

  INSTHOME은 인스턴스 소유자의 홈 경로입니다.

- DB2 UDB 인스턴스 프로파일 레지스트리는 다음 위치에 배치됩니다.
  - AIX, Solaris 및 Linux 운영 체제의 경우, /var/db2/<version_id>/profiles.reg(여기서, <version_id>는 현재 버전임).
  - HP-UX 운영 체제용 /var/opt/db2/<version_id>/profiles.reg(여기서, <version_id>는 현재 버전임)

  이 파일의 액세스 권한 및 소유권은 다음과 같아야 합니다.

  ```
  -rw-r--r-- root system profiles.reg
  ```

관련 개념:

- 47 페이지의『DB2 Administration Server』

태스크 관련:

- 36 페이지의『Windows에서 환경 변수 설정』

노드 구성 파일 작성

프로시저:

데이터베이스가 파티션된 데이터베이스 환경에서 작동하려면, db2nodes.cfg라는 노드 구성 파일을 작성해야 합니다. 이 파일은 인스턴스용 홈 디렉토리의 sqllib 서브디렉토리에 먼저 배치되어야 다중 파티션에 병렬 성능을 사용하는 데이터베이스 관리 프로그램을 시작할 수 있습니다. 파일에는 인스턴스의 모든 데이터베이스 파티션에 대한 구성 정보가 수록되어 있으며, 해당 인스턴스의 모든 데이터베이스 파티션이 이 파일을 공유합니다.

Windows 고려사항

Windows에서 DB2 Universal Database™(DB2 UDB) Enterprise - Server Edition을 사용 중인 경우, 인스턴스를 작성하면 노드 구성 파일이 작성됩니다. 노드 구성 파일을 수동으로 작성 또는 수정하려고 하지 마십시오. **db2ncrt** 명령을 사용하여 인스턴스에 데이터베이스 파티션 서버를 추가할 수 있습니다. **db2ndrop** 명령을 사용하여 인스턴스에서 데이터베이스 파티션 서버를 삭제할

수 있습니다. **db2nchg** 명령을 사용하여 하나의 머신에서 다른 머신으로 데이터베이스 파티션 서버를 이동하거나, TCP/IP 호스트 이름을 변경하거나 혹은 다른 논리 포트 또는 네트워크 이름을 선택하는 등, 데이터베이스 파티션 서버 구성을 수정할 수 있습니다.

주: 인스턴스가 삭제될 때 데이터가 손실되지 않도록 하려면, `sqllib` 서브디렉토리 아래에 DB2 UDB가 작성한 것 이외의 파일이나 디렉토리를 작성해서는 안 됩니다. 여기에는 두 가지 예외사항이 있습니다. 시스템이 스토어드 프로시저를 지원하면 `sqllib` 서브디렉토리 아래의 `function` 서브디렉토리에 스토어드 프로시저 응용프로그램을 두십시오. 또 하나의 예외는 사용자 정의 함수(UDF)가 작성된 경우입니다. UDF 실행 파일은 동일한 디렉토리에 와야 합니다.

파일은 인스턴스에 속하는 각 데이터베이스 파티션당 하나의 행을 수록합니다. 각 행의 형식은 다음과 같습니다.

```
dbpartitionnum hostname [logical-port [netname]]
```

토큰은 공백에 의해 구분됩니다. 변수는 다음과 같습니다.

dbpartitionnum

0 - 9까지의 데이터베이스 파티션 번호는 노드를 고유하게 정의합니다. 데이터베이스 파티션 번호는 오름차순으로 되어 있어야 합니다. 시퀀스에는 갭이 있을 수 있습니다.

데이터베이스 파티션 번호를 지정한 후에는 이를 변경할 수 없습니다(그렇지 않으면, 데이터가 파티션되어 있는, 파티션 맵의 정보가 절충됩니다).

노드를 삭제할 경우, 해당 데이터베이스 파티션 번호를 추가하는 새 노드에 다시 사용할 수 있습니다.

데이터베이스 파티션 번호는 데이터베이스 디렉토리의 노드 이름을 생성하는데 사용됩니다. 형식은 다음과 같습니다.

```
NODEnnnn
```

*nnnn*은 데이터베이스 파티션 번호이며 왼쪽은 0으로 채워집니다. 이 데이터베이스 파티션 번호는 **CREATE DATABASE** 및 **DROP DATABASE** 명령에서도 사용됩니다.

hostname

파티션간 통신용 IP 주소의 호스트 이름. 호스트 이름에 대한 완전한 이름을 사용하십시오. `/etc/hosts` 파일 또한 완전한 이름을 사용해야 합니다. db2nodes.cfg 파일 및 `/etc/hosts` 파일에서 완전한 이름을 사용하지 않을 경우, SQL30082N RC=3 오류 메시지를 받을 수도 있습니다.

(netname이 지정될 경우 예외가 발생합니다. 이러한 상황에서는 대부분의 통신에 netname이 사용되고 **db2start**, **db2stop** 및 **db2_all**에만 hostname이 사용됩니다).

logical-port

이 매개변수는 선택적이며, 노드용 논리 포트 번호를 지정합니다. 이 번호는 데이터베이스 관리 인스턴스 이름과 함께 사용되어, etc/services 파일의 TCP/IP 서비스 이름 항목을 식별합니다.

IP 주소와 논리 포트를 조합하면 잘 알려진 주소로 사용할 수 있고, 노드 간 통신 연결을 지원하는 모든 응용프로그램에서 고유합니다.

각 hostname에 대해, 하나의 logical-port가 0이거나 공백이어야 합니다(디폴트값은 0입니다). 이 logical-port와 연관된 노드는 클라이언트가 연결될 호스트의 디폴트 노드입니다. db2profile 스크립트에서 또는 sqlesetc() API를 사용하여 DB2NODE 환경 변수로 겹쳐쓸 수 있습니다.

동일한 호스트에 여러 개의 노드가 있으면(즉, 하나의 호스트에 둘 이상의 dbpartitionnum이 있을 경우), 논리 노드에 0에서 시작하여 갭 없이 logical-port 번호를 지정해야 합니다. 순서는 중요하지 않습니다.

예를 들어, 다음 설정이 유효합니다.

```
0   cpaiss43.mach1.xxx.com   1
1   cpaiss43.mach1.xxx.com   0
2   cpaiss43.mach1.xxx.com   2
3   cpaiss44.mach1.xxx.com   0
```

netname

이 매개변수는 선택적으로서, 각각 자체의 호스트 이름을 가지고, 활동 중인 TCP/IP 인터페이스를 두 개 이상 가진 호스트를 지원하는 데 사용됩니다.

다음 예는 SP2EN1이 다중 TCP/IP 인터페이스, 즉 두 개의 논리 파티션을 가지고 있으며 SP2SW1을 DB2 UDB 인터페이스로 사용하는 RS/6000 SP 시스템용으로 가능한 노드 구성 파일을 보여줍니다. 여기서는 또한 1에서 시작하는 파티션 번호와 dbpartitionnum 시퀀스의 갭을 보여줍니다.

표 1. 데이터베이스 파티션 번호 예 테이블.

dbpartitionnum	hostname	logical-port	netname
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

임의의 편집기를 사용하여 db2nodes.cfg를 갱신할 수 있습니다 (예외는 Windows에서 편집기를 사용하지 말아야 한다는 것입니다). 그러나 데이터 파티션 작성시 데이터베이스 파티션 번호가 변경되어서는 안되므로, 파일 정보의 무결성이 보호되도록 주의

해야 합니다. **db2start**를 발행하면 노드 구성 파일이 잠기고, **db2stop**이 데이터베이스 관리를 종료하면 잠금이 해제됩니다. 파일이 잠겨 있을 때, **db2start** 명령은 필요한 경우 파일을 갱신할 수 있습니다. 예를 들어, RESTART 옵션 또는 ADDNODE 옵션을 사용하여 **db2start** 명령을 발행할 수 있습니다.

주: db2stop 명령이 실패하여 노드 구성 파일이 잠금 해제되지 않으면 **db2stop FORCE** 명령을 발행하여 잠금을 해제하십시오.

관련 개념:

- *관리 안내서: 성능*의 『스토어드 프로시저에 대한 지침』

관련 참조:

- *Command Reference*의 『db2start - Start DB2 Command』
- *Command Reference*의 『db2stop - Stop DB2 Command』
- *Command Reference*의 『CREATE DATABASE Command』
- *Command Reference*의 『DROP DATABASE Command』
- *Command Reference*의 『db2nchg - Change Database Partition Server Configuration Command』
- *Command Reference*의 『db2ncrt - Add Database Partition Server to an Instance Command』
- *Command Reference*의 『db2ndrop - Drop Database Partition Server from an Instance Command』

데이터베이스 구성 파일 작성

프로시저:

각 데이터베이스마다 하나의 *데이터베이스 구성 파일*이 작성됩니다. 이 파일의 작성이 완료되었습니다. 이 파일에는 다음과 같이 데이터베이스의 사용에 영향을 주는 여러 가지 *구성 매개변수*에 대한 값이 들어 있습니다.

- 데이터베이스 작성시 지정 또는 사용된 매개변수(예를 들어, 데이터베이스 코드 페이지, 조합 시퀀스, DB2 UDB 릴리스 레벨)
- 데이터베이스의 현재 상태를 나타내는 매개변수(예: 백업 보류 플래그, 데이터베이스 일관성 플래그, 롤 포워드 보류 플래그)
- 데이터베이스 조작시 사용할 수 있는 시스템 자원의 양을 정의하는 매개변수(예: 버퍼 풀 크기, 데이터베이스 로그, 정렬 메모리 크기)

구성 파일에서 매개변수를 수동으로 변경하지 마십시오. 지원 인터페이스를 단지 사용하기만 하십시오.

성능 추가 정보: 구성 매개변수의 대부분은 디폴트값으로 주어지지만, 데이터베이스의 최적의 성능을 이루려면 이 값을 갱신해야 합니다.

다중 파티션의 경우 둘 이상의 파티션에 걸쳐 파티션된 데이터베이스가 있으면, 구성 파일은 모든 데이터베이스 파티션에서 동일해야 합니다. SQL 컴파일러는 로컬 노드 구성 파일의 정보에 근거하여 분산 SQL문을 컴파일하고, SQL문의 요구를 충족시킬 액세스 플랜을 작성하기 때문에 일관성이 요구됩니다. 데이터베이스 파티션에서 다른 구성 파일을 유지보수하는 작업은 명령문이 준비된 데이터베이스 파티션에 따라 다른 액세스 플랜이 될 수 있습니다. 구성 파일이 모든 데이터베이스 파티션에서 동기화되게 보존하려면 **db2_all**을 사용하십시오.

관련 개념:

• 403 페이지의 『파티션된 데이터베이스 환경에서 명령 발행』

태스크 관련:

• *관리 안내서: 성능*의 『구성 매개변수를 사용하여 DB2 구성』

FCM(Fast Communication Manager) 통신

파티션된 데이터베이스 환경에서는 FCM(Fast Communication Manager)으로 데이터베이스 파티션간의 대부분의 통신을 다룹니다. 데이터베이스 파티션에서 FCM을 사용하고 다른 데이터베이스 파티션과의 통신을 사용하려면, 아래 표시된 것처럼 `etc` 디렉토리의 파티션 `services` 파일에서 서비스 항목을 작성해야 합니다. FCM은 지정된 포트를 사용하여 통신합니다. 같은 호스트에 다중 파티션이 있으면, 다음과 같이 포트 범위를 정의해야 합니다.

Windows® 고려사항

Windows 환경에서 DB2® Universal Database(DB2 UDB) Enterprise - Server Edition을 사용할 경우, TCP/IP 포트 범위가 다음에 의해 서비스 파일에 자동으로 추가됩니다.

- 인스턴스를 작성하거나 새로운 노드를 추가할 때의 설치 프로그램
- **db2icrt** 유틸리티가 새 인스턴스를 작성할 때 이 유틸리티에 의해
- **db2ncrt** 유틸리티가 머신에 첫 번째 노드를 추가할 때 이 유틸리티에 의해

서비스 항목의 구문은 다음과 같습니다.

```
DB2_instance port/tcp #comment
```

DB2_*instance*

인스턴스 값은 데이터베이스 관리 인스턴스 이름과 같습니다. 이름의 모든 문자는 소문자로 되어 있어야 합니다. 인스턴스 이름이 `db2puser`일 경우, `DB2_db2puser`를 지정하십시오.

port/**tcp**

데이터베이스 파티션에 대해 예약하고자 하는 TCP/IP 포트

#comment

항목과 연관시키고자 하는 모든 주석. 주석 앞에는 파운드 부호(#)가 와야 합니다.

etc 디렉토리의 services 파일이 공유되면, 파일에서 할당된 포트 수가 인스턴스의 다중 데이터베이스 파티션 최대 수보다 크거나 같도록 해야 합니다. 또한 포트를 할당할 때에는 백업으로 사용할 수 있는 모든 프로세서를 설명합니다.

etc 디렉토리의 services 파일이 공유되지 않을 경우에도 동일한 고려사항이 적용되며 추가로 다음 사항을 고려해야 합니다. DB2 UDB 인스턴스에 대해 정의된 항목이 etc 디렉토리의 모든 services 파일에서 동일하도록 해야 합니다(파티션된 데이터베이스에 적용되지 않는 다른 항목들은 동일하지 않아도 됩니다).

하나의 인스턴스의 동일한 호스트에 다중 데이터베이스 파티션이 있을 경우, 사용할 FCM에 대해 둘 이상의 포트를 정의해야 합니다. 이를 수행하려면 etc 디렉토리의 services 파일에 두 행을 입력하여 할당할 포트의 범위를 지정하십시오. 첫 번째 행은 첫 번째 포트를 지정하고, 두 번째 행은 포트 블록의 끝을 나타냅니다. 다음 예에서 인스턴스 sales에 대해 5개의 포트가 할당됩니다. 즉, 인스턴스의 어떠한 프로세서도 6개 이상의 데이터베이스 파티션을 갖지 않습니다. 예를 들어,

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

주: END는 반드시 대문자로 지정해야 합니다. 또한 둘다 밑줄(_) 문자도 넣어야 합니다.

관련 개념:

- *관리 안내서: 계획*의 『파티션 및 프로세서 환경』

제 2 장 DAS(DB2 Administration Server) 작성 및 사용

DAS(DB2 Administration Server)는 DB2 서버 태스크를 보조하기 위해 사용됩니다.

DB2 Administration Server

DAS(DB2® Administration Server)는 DB2 Universal Database™(DB2 UDB) 서버의 태스크를 지원하는 데에만 사용되는 제어점입니다. 구성 지원 프로그램, 제어 센터 또는 개발 센터와 같은 도구를 사용하려면 DAS가 실행 중이어야 합니다. 다음과 같은 관리 태스크를 수행할 때, DAS는 제어 센터 및 구성 지원 프로그램을 지원합니다.

- DB2 UDB 서버의 리모트 관리를 사용합니다.
- DB2 UDB 및 운영 체제 명령 스크립트의 실행 스케줄 설정 기능을 포함하여 작업 관리 기능을 제공합니다. 이러한 명령 스크립트는 사용자 정의됩니다.
- 태스크 센터를 사용하여 작업 스케줄을 정의하고, 완료된 작업 결과를 보며, DAS로부터 리모트 또는 로컬에 위치한 작업에 대해 기타 관리 태스크를 수행합니다.
- DB2 UDB 발견 유틸리티와 함께 DB2 인스턴스, 데이터베이스 및 기타 DB2 UDB Administration Server의 구성에 대한 정보를 발견하는 방법을 제공합니다. 이러한 정보는 구성 지원 프로그램 및 제어 센터가 DB2 UDB 데이터베이스로의 연결 구성을 간소화하고 자동화하는데 사용됩니다.

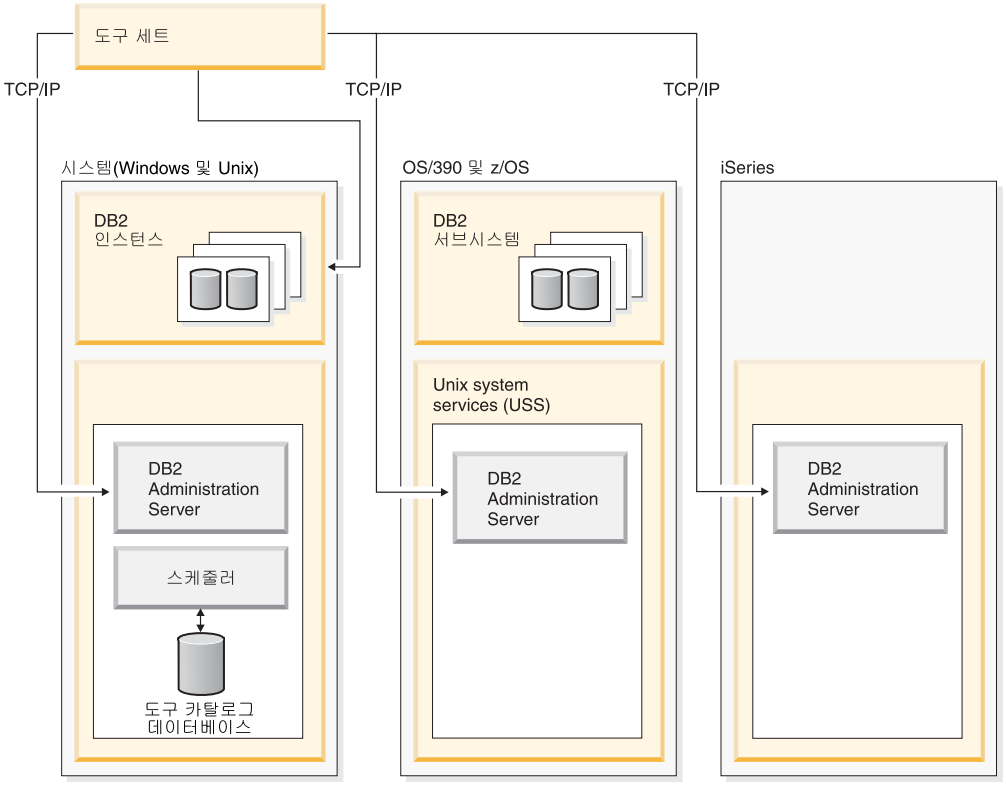


그림 1. DAS가 사용되는 시스템

한 머신에 하나의 DAS만 가질 수 있습니다. DAS는 운영 체제가 부트될 때 시작하기 위해 설치되는 동안 구성됩니다.

DAS는 제어 센터 또는 구성 지원 프로그램 또는 기타 사용 가능한 도구로부터의 클라이언트 요청을 대신하여 서버 시스템이나 호스트 시스템에서 리모트 태스크를 수행하는 데 사용됩니다.

DAS는 지원되는 모든 Windows®, UNIX®, zSeries®(OS/390 및 z/OS™에 한함) 플랫폼에서 사용 가능합니다. zSeries의 DAS는 관리 태스크에서 제어 센터, 개발 센터 및 복제 센터를 지원하는데 사용됩니다.

zSeries(OS/390 및 z/OS에 한함)에서 사용되는 DB2 Administration Server는 패키징되어 DB2 UDB의 DB2 관리 클라이언트 기능의 일부로서 전달됩니다. 제어 센터, 복제 센터 및 개발 센터와 같이 DAS를 필요로 하는 제품은 DAS 기능을 설치해야 합니다. 사용하는 운영 체제에서의 DAS 사용에 관한 자세한 내용은 IBM® 담당자에게 문의하십시오.

Windows 및 UNIX 상의 DAS에는 태스크 센터로 정의된 태스크(예: DB2 UDB 및 운영 체제 명령 스크립트)를 실행하는 스케줄러가 있습니다. 실행될 명령, 태스크와 연관된 스케줄, 통지 및 완료 조치와 같은 태스크 정보와 실행 결과는 도구 카탈로그라는

DB2 UDB 데이터베이스에 저장됩니다. 도구 카탈로그는 설치 프로그램의 일부로서 작성됩니다. 또한 제어 센터 또는 CLP에서 **CREATE TOOLS CATALOG** 명령을 사용하여 작성할 수도 있습니다.

zSeries(OS/390 및 z/OS에 한함)에서 스케줄러를 제공하지 않더라도, 제어 센터에서 제공하는 JCL 빌드 및 JCL 작성 기능을 사용하여 시스템 스케줄러로 실행될 파티션된 데이터 세트에 저장되는 JCL을 생성할 수 있습니다.

관련 개념:

- 60 페이지의 『Windows에서의 DAS에 대한 보안 고려사항』
- 66 페이지의 『ESE(Enterprise Server Edition) 시스템에서 DAS 구성』
- 67 페이지의 『관리 서버, 인스턴스 및 데이터베이스 발견』
- 72 페이지의 『DB2 Administration Server 첫 번째 실패 데이터 캡처』

태스크 관련:

- 49 페이지의 『DB2 Administration Server 작성』
- 51 페이지의 『DAS 시작 및 중지』
- 52 페이지의 『DAS 나열』
- 52 페이지의 『DAS 구성』
- 61 페이지의 『UNIX에서 DAS 갱신』
- 62 페이지의 『DAS 제거』
- 63 페이지의 『ESE(Enterprise Server Edition) 시스템과 함께 DAS 설치』
- 69 페이지의 『발견으로부터 서버 인스턴스 및 데이터베이스 숨기기』
- 69 페이지의 『발견 매개변수 설정』
- 70 페이지의 『DAS를 설정하여 구성 지원 프로그램 및 제어 센터 사용』
- 71 페이지의 『discovery에 대한 DAS 구성 갱신』
- 53 페이지의 『도구 카탈로그 데이터베이스 및 DAS 스케줄러 설정 및 구성』
- 59 페이지의 『통지 및 접속 목록 설정 및 구성』
- 59 페이지의 『DAS JVM(Java Virtual Machine) 설치』

DB2 Administration Server 작성

DB2 Administration Server는 제어 센터 및 구성 지원 프로그램과 같은 DB2 Universal Database™(DB2 UDB) 도구에 대한 지원 서비스를 제공합니다.

전제조건:

DAS를 작성하려면 UNIX 플랫폼에서 root 권한을 가지거나 서비스를 작성할 수 있는 올바른 권한을 가지는 어카운트를 사용해야 합니다.

Windows에서 특정 사용자를 식별해야 하는 경우, 로컬 Administrator 권한을 갖는 사용자를 작성하십시오. **db2admin create**를 입력하십시오. 특정 사용자 어카운트를 원하는 경우에는 **db2admin create** 명령을 발행할 때 『/USER:』 및 『/PASSWORD:』를 사용해야 합니다.

프로시저:

일반적으로, 설치 프로그램은 DB2 UDB 설치 중에 인스턴스를 소유하는 머신에서 DAS를 작성합니다. 그러나 설치 프로그램에서 이를 작성하지 못하면, DAS를 수동으로 작성할 수 있습니다.

DAS와 관련하여 설치 프로세스 중 발생하는 사항에 대한 개요로서 다음을 고려하십시오.

- Windows 플랫폼에서

  서비스를 작성할 수 있는 올바른 권한이 있는 어카운트를 사용하여 DAS를 작성할 머신에 로그온하십시오.

  DAS를 작성하는 경우, 사용자 어카운트 이름과 사용자 암호를 선택적으로 제공할 수 있습니다. 유효하면, 사용자 어카운트 이름과 암호는 DAS의 소유자를 식별합니다. DAS에 대해 작성된 사용자 ID 또는 어카운트 이름을 사용자 어카운트로 사용하지 마십시오. 어카운트 이름에 대한 암호를 『절대로 만기되지 않는 암호』로 설정합니다. DAS를 작성한 후, **db2admin setid** 명령을 사용한 사용자 어카운트 이름과 사용자 암호를 제공하여 해당 소유권을 설정하거나 변경할 수 있습니다.
- UNIX 플랫폼에서
  1. 사용자에게 root 권한이 있는지 확인하십시오.
  2. 명령 프롬프트에서 DB2 UDB 설치 경로의 instance 서브디렉토리에서 다음 명령을 발행하십시오.

     ```
     dascrt -u <DASUser>
     ```

     <DASUser>는 DB2 UDB의 사용자 및 그룹을 작성할 때 작성한 DAS 사용자의 사용자 이름입니다.

     – AIX에서

     ```
     /usr/opt/db2_08_01/instance/
     dascrt -u <DASUser>
     ```

     – HP-UX, Solaris 운영 환경 또는 Linux:

     ```
     /opt/IBM/db2/V8.1/instance/
     dascrt -u <DASUser>
     ```

관련 참조:

- 『db2admin - DB2 Administration Server Command』의 *Command Reference*

DAS 시작 및 중지

프로시저:

Windows에서 DAS를 수동으로 시작하거나 중지하려면 먼저 Administrators, Server Operators 또는 Power Users 그룹에 속하는 어카운트나 사용자 ID를 사용하여 머신에 로그온해야 합니다. Unix에서 DAS를 수동으로 시작하거나 중지하려면 어카운트나 사용자 ID를 *dasadm_group*의 일부로 만들어야 합니다. *dasadm_group*은 DAS 구성 매개변수에 지정됩니다.

Windows에서 DAS를 시작하거나 중지하려면 **db2admin start** 또는 **db2admin stop** 명령을 사용하십시오.

UNIX 운영 체제의 DB2 Universal Database™(DB2 UDB)에서 작업할 경우, 다음을 수행해야 합니다.

- DAS를 시작하려면, 다음을 수행하십시오.
  1. DAS 소유자로 로그인하십시오.
  2. 다음 중 하나를 사용하여 시동 스크립트를 수행하십시오.

     ```
     . DASHOME/das/dasprofile    (for Bourne or Korn shell)
     source DASHOME/das/dascshrc (for C shell)
     ```

     여기서, DASHOME은 DB2 Administration Server의 홈 디렉토리입니다.
  3. DAS를 시작하려면 **db2admin** 명령을 사용하십시오.

     ```
     db2admin start
     ```

  주: 각 시스템이 재부팅된 후 DAS는 자동으로 시작됩니다. **dasauto** 명령을 사용하여 DAS의 디폴트 시작 동작을 변경할 수 있습니다.
- DAS를 중지시키려면 다음을 수행하십시오.
  1. *dasadm_group*의 일부인 어카운트나 사용자 ID로 로그인하십시오.
  2. 다음과 같이 **db2admin** 명령을 사용하여 DAS를 중지시키십시오.

     ```
     db2admin stop
     ```

주: UNIX의 두 경우 모두, 이 명령을 사용하는 사용자는 DAS 소유자의 권한 부여 ID로 로그온해야 합니다. **db2admin start** 또는 **db2admin stop** 명령을 발행하려면 사용자가 *dasadm_group*에 속해야 합니다.

관련 참조:

- *Command Reference*의 『db2admin - DB2 Administration Server Command』
- *관리 안내서: 성능*의 『dasadm_group - DAS 관리 권한 그룹 이름 구성 매개변수』

DAS 나열

프로시저:

머신에서 DAS의 이름을 확보하려면 다음을 입력하십시오.

```
db2admin
```

이 명령은 DAS를 시작 또는 중지하고 새 사용자 및 암호를 작성하며 DAS를 삭제하고 DAS와 연관된 사용자 어카운트를 설정하거나 수정하는데에도 사용됩니다.

관련 참조:

- *Command Reference*의 『db2admin - DB2 Administration Server Command』

DAS 구성

프로시저:

DAS와 관련된 DB2 Administration Server 구성 매개변수의 현재 값을 보려면 다음을 입력하십시오.

```
db2 get admin cfg
```

여기에서는 제품 설치 중에 디폴트값으로 주어진 현재 값이나 이전에 구성 매개변수를 갱신할 때 주어진 값을 보여줍니다.

명령행 처리기(CLP)와 UPDATE ADMIN CONFIG 명령을 사용하여 DAS 구성 파일을 갱신하려면 DAS와 동일한 설치 레벨에 있는 인스턴스에서 CLP를 사용해야 합니다. DAS 구성 파일에 있는 개별 항목을 갱신하려면 다음을 입력하십시오.

```
db2 update admin cfg using ...
```

구성 매개변수를 권장되는 디폴트로 재설정하려면 다음을 입력하십시오.

```
db2 reset admin cfg
```

몇몇 경우에, DAS 구성 파일의 변경사항은 메모리에 로드된 후에만 적용됩니다(즉, **db2admin stop** 다음에 **db2admin start**가 실행될 때, 또는 Windows 플랫폼의 경우 서비스를 중지했다가 시작할 때). 다른 경우에는 구성 매개변수를 온라인으로 구성할 수 있습니다(즉, 변경사항을 적용하기 위해 DAS를 재시작하지 않아도 됩니다).

태스크 관련:

- *관리 안내서: 성능*의 『구성 매개변수를 사용하여 DB2 구성』

관련 참조:

- *Command Reference*의 『UPDATE ADMIN CONFIGURATION Command』

도구 카탈로그 데이터베이스 및 DAS 스케줄러 설정 및 구성

도구 카탈로그 데이터베이스에는 태스크 센터 및 제어 센터에서 작성한 태스크 정보가 들어 있습니다. 이러한 태스크는 DB2 Administration Server의 스케줄러에 의해 실행됩니다. 스케줄러 및 도구 카탈로그 데이터베이스는 항상 함께 작동하며, 둘 중 하나가 없을 경우 작동하지 않습니다. 스케줄러는 도구 카탈로그 데이터베이스를 읽기 위한 에이전트로 작동하며 해당하는 시간에 태스크를 실행하는 DB2 관리 서버의 특정 요소입니다.

전제조건:

DB2 Administration Server가 설치되어 있어야 합니다.

프로시저:

목표는 도구 카탈로그 데이터베이스 및 DAS 스케줄러를 설정하고 구성하는 것입니다.

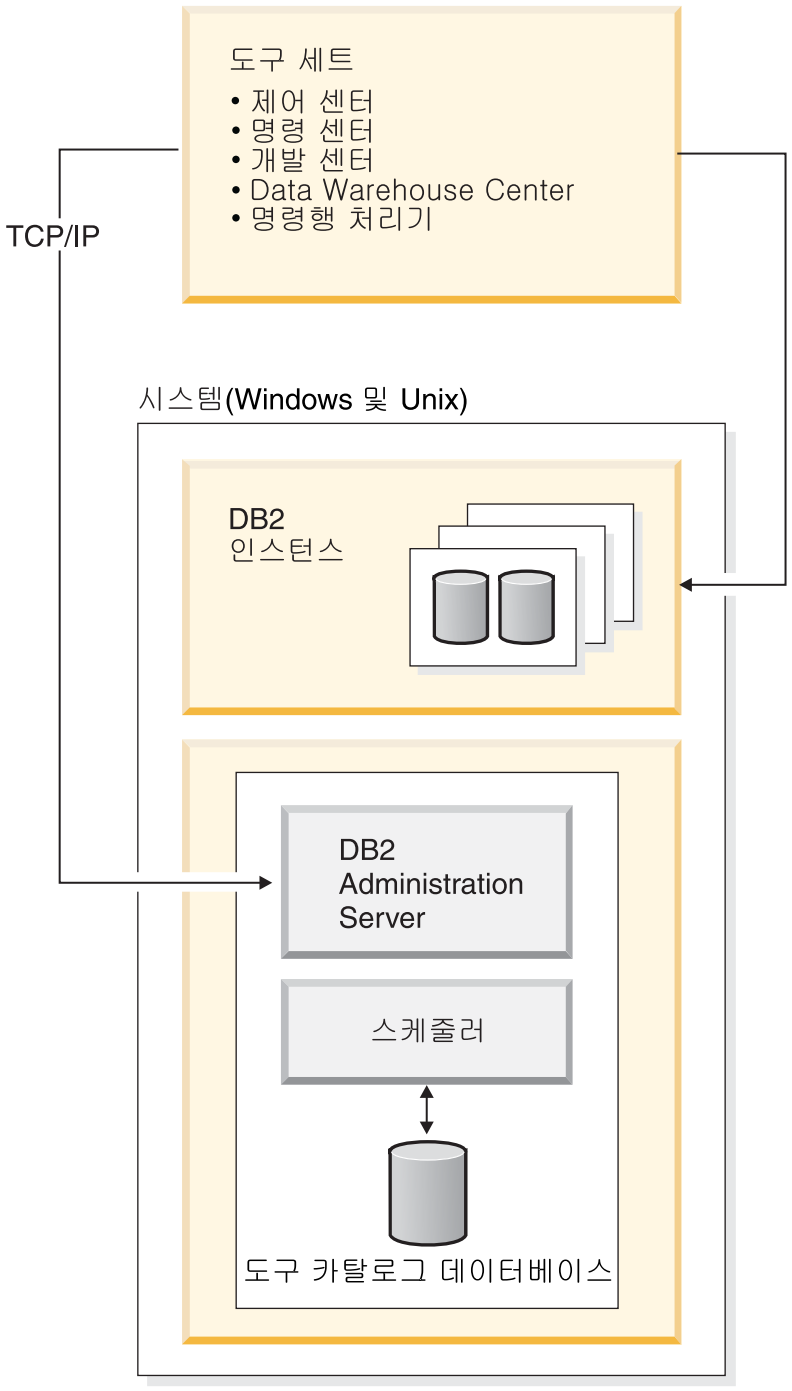


그림 2. DAS와 DB2 UDB의 다른 파트 간의 관계

DB2 관리 서버 구성 프로세스는 스케줄러에 도구 카탈로그 데이터베이스의 위치 및 스케줄러의 사용 가능화 여부를 알립니다. 디폴트로, 도구 카탈로그 데이터베이스가 작성되면 해당하는 DAS 구성이 갱신됩니다. 즉, 스케줄러가 구성되며 새 도구 카탈로그를 사용할 준비가 됩니다. DAS를 재시작할 필요가 없습니다.

도구 카탈로그 데이터베이스를 스케줄러 시스템의 로컬 또는 리모트 서버에 작성할 수 있습니다. 도구 카탈로그를 리모트 서버에 작성할 경우, 스케줄러 도구 카탈로그 데이터

베이스 인스턴스(TOOLSCAT_INST)에서 카탈로그되어야 합니다. 추가로, 스케줄러가 리모트 카탈로그에 연결하여 인증을 받을 수 있도록 **db2admin setschedid** 명령을 사용하여 스케줄러 사용자 ID를 설정해야 합니다. **db2admin** 명령의 전체 구문을 *Command Reference*에서 찾을 수 있습니다.

DAS 스케줄러는 도구 카탈로그 정보에 액세스하기 위해 JVM(Java Virtual Machine)을 필요로 합니다. JVM 정보는 DAS의 `jdk_path` DB2 Administration Server 구성 매개변수를 사용하여 지정합니다.

32비트 및 64비트 인스턴스를 모두 지원하는 플랫폼(AIX, Sun 및 HP-UX) 중 하나에서 64비트 인스턴스에 대해 도구 카탈로그를 작성하는 경우 `jdk_64_path` 구성 매개변수가 필요합니다.

제어 센터 및 태스크 센터는 클라이언트에서 직접 도구 카탈로그 데이터베이스에 액세스합니다. 따라서 제어 센터가 도구 카탈로그 데이터베이스를 사용하려면 먼저 클라이언트에서 도구 카탈로그 데이터베이스를 카탈로그화해야 합니다. 제어 센터는 자동으로 도구 카탈로그 데이터베이스에 대한 정보를 검색하고 로컬 노드 디렉토리 및 데이터베이스 디렉토리에 필요한 디렉토리 항목을 작성하는 수단을 제공합니다. 이러한 자동 카탈로그화를 위해 지원되는 유일한 통신 프로토콜은 TCP/IP입니다.

DAS 구성 매개변수 중 하나를 *exec_exp_task*라고 합니다. 이 매개변수는 스케줄러가 과거에 스케줄되었으나 아직 실행되지 않은 태스크를 실행하는지의 여부를 지정합니다. 스케줄러는 시작될 때 만기된 태스크만 발견합니다.

예를 들어, 토요일마다 실행되도록 스케줄된 작업이 있으며 스케줄러를 금요일에 껐다가 월요일에 재시작한 경우, 토요일에 실행되도록 스케줄된 작업은 이제 과거에 스케줄된 작업입니다. *exec_exp_task*가 『예』로 설정되어 있으면, 토요일 작업은 스케줄러가 재시작될 때 실행됩니다.

스케줄러에서 필요로 하는 다른 DAS 구성 매개변수는 도구 카탈로그 데이터베이스와 통지에 사용되는 SMTP(Simple Mail Transfer Protocol) 서버를 식별하는 것으로 구성됩니다.

다음 예는 이러한 매개변수가 사용되는 방법을 설명합니다.

- Windows 서버 설정 예.
  1. 도구 카탈로그 데이터베이스에 아무 이름이나 지정할 수 있습니다. 이 예에서는 도구 카탈로그 데이터베이스를 『CCMD』라고 하며 서버 머신 Host1(tcp/ip 호스트 이름 Host1)의 DB2 Universal Database™(DB2 UDB) 인스턴스 아래에 작성합니다. 주어진 데이터베이스 내에서 도구 카탈로그를 고유하게 식별하기 위해 스키마 이름이 사용됩니다. 이 예에서는 스키마 이름을 『CCADMIN』으로 가정합니다.

2. 다음을 사용하여, 포트 번호 50000을 사용하는 TCP/IP 통신용 인스턴스『DB2』를 설정합니다.

```
db2set -i DB2 DB2COMM=TCPIP
db2 update dbm cfg using svcename db2cDB2
 db2stop
db2start
```

3. db2cDB2 서비스 이름을 %SystemRoot%\system32\drivers\etc\services에 정의합니다. 즉 services에서 다음 행을 찾을 수 있습니다.

```
db2cDB2     50000/tcp     #connection port for the DB2 instance DB2
```

4. 도구 카탈로그는 CREATE TOOLS CATALOG 명령을 사용하여 작성됩니다. 그러면 지정된 데이터베이스의 카탈로그 이름에 해당하는 스키마 이름으로 도구 카탈로그 테이블과 뷰가 작성됩니다. DB2 Administration Server 구성 매개변수가 자동으로 갱신되며 스케줄러가 사용 가능하고 시작됩니다.

5. 전자 우편 통지에 사용되는 SMTP 서버가 머신 Host2(tcp/ip 호스트 이름 Host2)에 있다고 가정하십시오. 다음을 사용하여 이 정보를 DB2 Administration Server로 지정합니다.

```
db2 update admin cfg using smtp_server Host2
```

   이것은 설치 프로세스 동안에 수행할 수 있습니다. 이것을 나중에 수행할 경우에는, 위에 나와 있는 것과 같은 DB2 UDB 버전 8 CLP 명령을 통해 수동으로 DAS에 지정해야 합니다.

6. Windows에 있는 IBM Software Development Kit(SDK)는 %DB2PATH%\java\jdk 아래에 설치됩니다. 이것은 이미 DAS로 지정되어 있어야 합니다. 다음을 사용하여 JDK를 검증하고 설정(필요한 경우)할 수 있습니다.

```
db2 update admin cfg using jdk_path c:\SQLLIB\java\jdk
```

   이것은 DB2 UDB가 C:\SQLLIB 아래에 설치되어 있는 것으로 가정합니다.

주: DAS가 db2admin create로 작성될 경우 /USER 및 /PASSWORD 옵션을 사용해야 합니다. USER 어카운트가 스케줄러 프로세스에 의해 사용되었습니다. 사용하지 않으면 스케줄러가 제대로 시작되지 않습니다. USER 어카운트는 도구 카탈로그 인스턴스에 대한 SYSADM 권한을 가지고 있어야 합니다.

DAS가 db2admin create로 작성될 것이며 이 때 /USER 및 /PASSWORD 옵션을 지정하지 않을 경우 나중에 USER 정보를 갱신할 수 있습니다. 이 갱신은 다음의 명령을 실행하여 DAS에서 수행됩니다.

```
db2admin stop
db2admin setid <user account ID> <password>
db2admin start
```

- Windows 클라이언트 설정 예

1. 제어 센터가 클라이언트 머신 C1(tcp/ip 호스트 이름 C1)에서 실행 중인 것으로 가정하십시오.
2. 구성 지원 프로그램 또는 제어 센터를 사용하거나 다음 명령을 사용하여 DAS를 로컬 노드 디렉토리에 관리 서버 노드로서 카탈로그화합니다.

   ```
   db2 catalog admin tcpip node Host1 remote Host1 system Host1 ostype NT
   ```

3. 태스크 센터를 시작하고 시스템 Host1을 선택하면 태스크 센터가 로컬 디렉토리에서 도구 카탈로그 데이터베이스를 찾으려고 시도합니다(태스크 센터 대신 제어 센터가 사용될 수도 있습니다). 찾을 수 없는 경우에는 다음을 사용하여 노드 및 데이터베이스를 카탈로그화하려고 합니다.

   ```
   db2 catalog tcpip node <unique-node name>
     remote Host1 server 50000
     remote_instance DB2 system Host1 ostype NT
   db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
   ```

   자동 카탈로그화가 실패하면 구성 지원 프로그램 또는 제어 센터를 사용하여 데이터베이스를 카탈로그화할 수 있습니다. 그러면 태스크 센터에서 데이터베이스를 인식하고 사용하게 됩니다.

- AIX 서버 설정 예.
  1. 도구 카탈로그 데이터베이스에 아무 이름이나 지정할 수 있습니다. 이 예에서는 도구 카탈로그 데이터베이스를 『CCMD』라고 하며 서버 머신 Host1(tcp/ip 호스트 이름 Host1)의 db2inst1 인스턴스 아래에 작성합니다. 주어진 데이터베이스 내에서 도구 카탈로그를 고유하게 식별하기 위해 스키마 이름이 사용됩니다. 이 예에서는 스키마 이름을 『CCADMIN』으로 가정합니다.
  2. 다음을 사용하여, 포트 번호 50000을 사용하는 TCP/IP 통신용 인스턴스 db2inst1를 설정합니다.

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename xdb2inst
      db2stop
   db2start
     ```

  3. xdb2inst 서비스 이름을 /etc/services에 정의합니다. 즉 services에서 다음 행을 찾을 수 있습니다.

     ```
     xdb2inst1     50000/tcp     #connection port for the DB2 instance db2inst1
     ```

  4. 도구 카탈로그는 CREATE TOOLS CATALOG 명령을 사용하여 작성됩니다. 그러면 지정된 데이터베이스의 카탈로그 이름에 해당하는 스키마 이름으로 도구 카탈로그 테이블과 뷰가 작성됩니다. DB2 Administration Server 구성 매개변수가 자동으로 갱신되며 스케줄러가 사용 가능하고 시작됩니다.
  5. 전자 우편 통지에 사용되는 SMTP 서버가 머신 Host2(tcp/ip 호스트 이름 Host2)에 있다고 가정하십시오. 다음을 사용하여 이 정보를 DB2 Administration Server로 지정합니다.

     ```
     db2 update admin cfg using smtp_server Host2
     ```

이것은 설치 프로세스 동안에 수행할 수 있습니다. 이것을 나중에 수행할 경우에는, 위에 나와 있는 것과 같은 DB2 UDB 버전 8 CLP 명령을 통해 수동으로 DAS에 지정해야 합니다.

6. AIX에서 IBM Software Developer's Kit for Java(SDK) 버전 1.3.1은 /sqllib/java/jdk 아래 설치됩니다. 이것은 이미 DAS로 지정되어 있어야 합니다. 다음을 사용하여 JDK를 검증하고 설정(필요한 경우)할 수 있습니다.

   ```
   db2 update admin cfg using jdk_path /sqllib/java/jdk
   ```

- AIX 클라이언트 설정 예

  1. 제어 센터가 클라이언트 머신 C1(tcp/ip 호스트 이름 C1)에서 실행 중인 것으로 가정하십시오.
  2. 구성 지원 프로그램 또는 제어 센터를 사용하거나 다음 명령을 사용하여 DAS를 로컬 노드 디렉토리에 관리 서버 노드로서 카탈로그화합니다.

     ```
     db2 catalog admin tcpip node Host1 remote Host1 system Host1
       ostype AIX
     ```

  3. 태스크 센터를 시작하고 시스템 Host1을 선택하면 태스크 센터가 로컬 디렉토리에서 도구 카탈로그 데이터베이스를 찾으려고 시도합니다(태스크 센터 대신 제어 센터가 사용될 수도 있습니다). 찾을 수 없는 경우에는 다음을 사용하여 노드 및 데이터베이스를 카탈로그화하려고 합니다.

     ```
     db2 catalog tcpip node <unique-node name>
       remote Host1 server 50000
       remote_instance DB2 system Host1 ostype AIX
     db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
     ```

     자동 카탈로그화가 실패하면 구성 지원 프로그램 또는 제어 센터를 사용하여 데이터베이스를 카탈로그화할 수 있습니다. 그러면 태스크 센터에서 데이터베이스를 인식하고 사용하게 됩니다.

관련 참조:

- *관리 안내서: 성능*의 『svcename - TCP/IP 서비스 이름 구성 매개변수』
- *관리 안내서: 성능*의 『sched_enable - 스케줄러 모드 구성 매개변수』
- *관리 안내서: 성능*의 『toolscat_inst - 도구 데이터베이스 카탈로그 인스턴스 구성 매개변수』
- *관리 안내서: 성능*의 『toolscat_db - 도구 데이터베이스 카탈로그 구성 매개변수』
- *관리 안내서: 성능*의 『toolscat_schema - 도구 데이터베이스 카탈로그 스키마 구성 매개변수』
- *관리 안내서: 성능*의 『smtp_server - SMTP 서버 구성 매개변수』
- *관리 안내서: 성능*의 『jdk_path - JDK 설치 경로 DAS 구성 매개변수』
- *관리 안내서: 성능*의 『exec_exp_task - 만기된 태스크 실행 구성 매개변수』

통지 및 접속 목록 설정 및 구성

DAS(DB2 Administration Server)의 전자 우편 및 호출기 통지는 로컬 또는 리모트일 수 있습니다. 올바른 호스트 이름으로 통지를 전송하려면 접속 목록이 필요합니다.

프로시저:

스케줄러 또는 health monitor에 의한 통지를 가능하게 하는 두 개의 DAS 구성 매개변수가 있습니다.

DAS 구성 매개변수 *smtp_server*는 스케줄러가 태스크 센터를 통해 정의된 태스크 실행 완료 조치의 일부로서 전자 우편 및 호출기 통지를 보내거나 또는 Health Monitor가 전자 우편 또는 호출기를 통해 경고 통지를 보내는 데 사용하는 SMTP(Simple Mail Transfer Protocol) 서버를 식별하는 데 사용됩니다.

DAS 구성 매개변수 *contact_host*는 스케줄러 및 Health Monitor가 통지를 위해 사용하는 접속 정보가 저장되는 위치를 지정합니다. 이 위치는 DB2 Administration Server의 TCP/IP 호스트 이름으로 정의됩니다. 리모트 DAS에 *contact_host*를 위치시키면 여러 DB2 Administration Server 간에 접속 목록을 공유할 수 있습니다. 모든 파티션에서 공통된 접속 목록을 사용하도록 하려면 파티션된 데이터베이스 환경에 대해서도 이것을 설정해야 합니다. 접속 목록은 DAS 디렉토리 아래의 플랫 파일에 저장됩니다. *contact_host*를 지정하지 않을 경우, DAS는 접속 정보가 로컬인 것으로 가정합니다.

관련 참조:

- *관리 안내서: 성능*의 『smtp_server - SMTP 서버 구성 매개변수』
- *관리 안내서: 성능*의 『contact_host - 접속 목록 위치 구성 매개변수』

DAS JVM(Java Virtual Machine) 설치

프로시저:

jdk_path 구성 매개변수는 DB2 관리 서버 기능을 실행하는 데 사용될 IBM Software Developer's Kit(SDK) for Java가 설치되는 디렉토리를 지정합니다. Java 인터프리터에서 사용하는 환경 변수가 이 매개변수의 값으로부터 계산됩니다.

스케줄러는 도구 카탈로그 데이터베이스를 사용하기 위해 JVM(Java Virtual Machine)을 필요로 합니다. 스케줄러를 시작하려면 먼저 JVM을 설치해야 합니다.

UNIX 플랫폼에서 작업할 때는 이 매개변수에 대한 디폴트값이 없습니다. IBM Software Developer's Kit(SDK) for Java를 설치할 때 이 매개변수의 값을 지정해야 합니다.

Windows에서 IBM Software Developer's Kit(SDK) for Java는 %DB2PATH%\java\jdk(Windows 플랫폼에서 해당 매개변수의 디폴트값) 아래 설치됩니다. 이것은 이미 DAS로 지정되어 있어야 합니다. 다음을 사용하여 *jdk_path*의 값을 확인할 수 있습니다.

```
db2 get admin cfg
```

이 명령은 *jdk_path*가 구성 매개변수 중 하나인 DB2 Administration Server 구성 파일의 값을 표시합니다. 필요한 경우, 다음을 사용하여 이 매개변수를 설정할 수 있습니다.

```
db2 update admin cfg using jdk_path 'C:\Program Files\IBM\SQLLIB'
```

이것은 DB2 Universal Database™(DB2 UDB)가 'C:\Program Files\IBM\SQLLIB' 아래에 설치되어 있는 것으로 가정합니다.

AIX에 있는 IBM Software Developer's Kit(SDK)은 /usr/java130 아래에 설치됩니다. 필요한 경우, 다음을 사용하여 이 매개변수를 설정할 수 있습니다.

```
db2 update admin cfg using jdk_path /usr/java130
```

주: 32비트 및 64비트 인스턴스를 모두 지원하는 플랫폼(AIX, Sun 또는 HP-UX) 중 하나에서 64비트 인스턴스에 대해 도구 카탈로그를 작성하거나 사용 중인 경우, jdk_64_path 구성 매개변수를 jdk_path 매개변수 대신 사용하십시오. 이 구성 매개변수는 IBM Software Developer's Kit(SDK) for Java의 64비트 버전이 설치되는 디렉토리를 지정합니다.

관련 참조:

- *Command Reference*의 『GET ADMIN CONFIGURATION Command』
- *Command Reference*의 『UPDATE ADMIN CONFIGURATION Command』
- *관리 안내서: 성능*의 『jdk_path - JDK 설치 경로 DAS 구성 매개변수』

Windows에서의 DAS에 대한 보안 고려사항

Windows에서 DAS 서비스가 실행되는 사용자 ID를 변경해야 할 수도 있습니다.

DAS를 작성한 후 다음과 같이 **db2admin** 명령을 사용하여 로그온 어카운트를 설정하거나 변경하십시오.

```
db2admin setid <username> <password>
```

여기서, <username> 및 <password>는 로컬 Administrators 권한을 갖는 어카운트의 사용자 이름 및 암호입니다. 이 명령을 실행하기 전에 로컬 관리자 권한이 있는 어카운트나 사용자 ID를 사용하여 머신에 로그온해야 합니다.

주: 암호는 대소문자가 구분됩니다. 대문자와 소문자의 혼합 사용이 가능하며 이는 암호의 대소문자 구분이 매우 중요함을 의미합니다.

주: Windows에서는 **제어판**의 **서비스** 유틸리티를 사용하여 DAS에 대한 로그온 어카운트를 변경해서는 안됩니다. 그 이유는 로그온 어카운트에 대해 일부 필요한 액세스 권한이 설정되지 않기 때문입니다. 항상 **db2admin** 명령을 사용하여 DAS(DB2® administration server)에 대한 로그온 어카운트를 설정하거나 변경하십시오.

관련 참조:

- 『db2admin - DB2 Administration Server Command』의 *Command Reference*

UNIX에서 DAS 갱신

프로시저:

UNIX 운영 체제에서는, 프로그램 임시 수정(PTF) 또는 코드 패치를 설치하여 DB2를 갱신할 경우, 각 DAS(DB2 Administration Server) 및 인스턴스를 갱신해야 합니다. DAS를 갱신하려면, 설치된 DB2 버전 및 릴리스에 고유한 서브디렉토리 아래의 `instance` 서브디렉토리에 있는 **dasupdt** 명령을 사용하십시오.

먼저 수퍼유저 권한으로(주로 『root』로서) 머신에 로그온해야 합니다.

명령은 다음과 같이 사용됩니다.

```
dasupdt
```

이 명령과 연관된 선택적 매개변수도 있습니다.

- -h 또는 -?

  이 명령에 대한 도움말 메뉴를 표시합니다.

- -d

  문제점 분석에 사용되는 디버그 모드를 설정합니다.

- -D

  DAS를 한 경로의 상위 코드 레벨에서 다른 경로에 설치된 하위 코드 레벨로 이동시킵니다.

주: Windows에서 DAS 갱신은 설치 프로세스의 일부입니다. 사용자 조치가 필요하지 않습니다.

예:

DAS는 버전 8 설치 경로에서 버전 8.1.2 코드를 실행하고 있습니다. FixPak 3이 버전 8 설치 경로에 설치된 경우 다음의 명령을 버전 8 설치 경로에서 호출하면 DAS가 FixPak 3으로 갱신됩니다.

```
dasupdt
```

DAS는 대체 설치 경로에서 버전 8.1.2 코드를 실행하고 있습니다. FixPak 1이 다른 대체 설치 경로에 설치되어 있으면, FixPak 1 대체 설치 경로에서 호출된 다음 명령은 DAS를 FixPak 1 대체 설치 경로에서 실행 중인 FixPak 1로 갱신합니다.

```
dasupdt -D
```

관련 개념:

- 47 페이지의『DB2 Administration Server』
- 60 페이지의『Windows에서의 DAS에 대한 보안 고려사항』

DAS 제거

프로시저:

DAS를 제거하려면, 다음을 수행하십시오.

- Windows 운영 체제에서
  1. 서비스를 제거할 수 있는 올바른 권한을 가지는 어카운트 또는 사용자 ID를 사용하여 머신에 로그온하십시오.
  2. **db2admin stop**을 사용하여 DAS를 중지시키십시오.
  3. `sqllib` 서브디렉토리 아래에 있는 `db2das00` 서브디렉토리의 모든 파일을 백업(필요하면)하십시오.

     주: 이 예에서는 `db2das00`을 제거될 DAS의 이름으로 간주합니다. 한 사용자가 이름이 DB2DAS00인 DB2 Universal Database™(DB2 UDB) 인스턴스를 작성한 경우에는 DAS는 DB2DAS00 이외의 이름을 가질 수 있습니다. 이러한 경우, DAS는 DB2DAS01이라는 이름을 가집니다(이것 역시 사용될 경우에는 DB2DAS02). 접두부가『DB2DAS』인 서비스를 찾아 존재할 가능성이 있는 몇몇 DAS 목록에서 특정 DAS를 식별해야 합니다. **db2admin** 명령을 아무런 옵션 없이 사용하여 모든 DAS를 나열할 수 있습니다.
  4. **db2admin drop**을 사용하여 DAS를 제거하십시오.
- UNIX 운영 체제에서
  1. DASADM 권한이 있는 사용자로 로그인하십시오.
  2. 다음 중 하나를 사용하여 시동 스크립트를 수행하십시오.

```
. DASHOME/das/dasprofile    (for Bourne or Korn shell)
source DASHOME/das/dascshrc (for C shell)
```

여기서, DASHOME은 DAS 소유자의 홈 디렉토리입니다.

3. 다음과 같이 **db2admin** 명령을 사용하여 DAS를 중지시키십시오.

   ```
   db2admin stop
   ```

4. DAS 홈 디렉토리 아래의 das 서브디렉토리에 있는 모든 파일을 백업하십시오(필요한 경우).
5. 로그오프하십시오.
6. root로 로그인한 후 다음과 같이 **dasdrop** 명령을 사용하여 DAS를 제거하십시오.

   ```
   dasdrop
   ```

   dasdrop 명령은 설치된 DB2 UDB 버전 및 릴리스에 고유한 서브디렉토리 아래의 instance 서브디렉토리에 있습니다.

   주: dasdrop 명령은 DAS(DB2 Administration Server)의 홈 디렉토리 아래에 있는 das 서브디렉토리를 제거합니다.

관련 참조:

- *Command Reference*의 『db2admin - DB2 Administration Server Command』
- *Command Reference*의 『dasdrop - Remove a DB2 Administration Server Command』

ESE(Enterprise Server Edition) 시스템과 함께 DAS 설치

프로시저:

다음 정보는 제어 센터를 사용하여 리모트 관리를 하는 경우, DB2 Universal Database™(DB2 UDB) ESE 서버(Linux, Solaris 운영 환경, Windows NT, Windows 2000, Windows Server 2003, HP-UX 및 AIX)를 구성하는 데 필요한 단계를 보여줍니다.

설치 중에 설치 프로그램은 인스턴스를 소유하는 머신에서 단일 DAS를 작성합니다. 제어 센터 또는 구성 지원 프로그램이 다른 코디네이터 노드에 액세스할 수 있도록 하려면 다른 머신에 추가 DAS를 작성해야 합니다. 그래야만 관리 코디네이터 노드로서 작업할 때의 오버헤드를 인스턴스에서 둘 이상의 파티션으로 분산시킬 수 있습니다. 설치 프로그램이 DAS가 실행되는 모든 노드에 DAS를 작성합니다. **db2setup**을 사용하지 않는 경우에만 이를 수동으로 수행해야 합니다.

여기에 나와 있는 지시사항은 파티션된 ESE 환경에만 적용될 수 있습니다. 단일 파티션 ESE 시스템에서 실행 중일 경우에는 여기에 나와 있는 지시사항이 적용되지 않습니다.

코디네이터 함수를 분배하려면, 다음을 수행하십시오.

1. 파티션된 데이터베이스 시스템에서 선택된 추가 머신에 새 DAS를 작성하십시오.
2. 제어 센터 또는 구성 지원 프로그램에서 각 DAS를 별도의 시스템으로 카탈로그화하십시오.
3. 각 새 시스템에서 동일한 인스턴스를 카탈로그화한 후, DAS를 카탈로그화할 때 사용하는 동일한 머신 이름을 매번 지정하십시오.

구성에는 두 가지 양상이 있습니다. 즉, 하나는 DAS(DB2 Administration Server)에 필요하고 다른 하나는 DB2 UDB 인스턴스로 관리되는 목표에 권장되는 것입니다. 다음 세 절에서는 각 절마다 두 가지 구성 주제 중 하나씩을 다룹니다. 각 구성 주제 앞에는 가정된 환경을 설명하는 절이 옵니다.

환경 예

제품/버전:
DB2 UDB ESE V8.1

설치 경로:
install_path

TCP 서비스 파일:
services

DB2 UDB 인스턴스:

이름: db2inst

소유자 ID:
db2inst

인스턴스 경로:
instance_path

노드: 3 노드, db2nodes.cfg:

- 0 hostA 0 hostAswitch
- 1 hostA 1 hostAswitch
- 2 hostB 0 hostBswitch

DB 이름:
db2instDB

DAS:

이름: db2as00

owner/user ID:
db2as

인스턴스 경로:

das_path

설치/실행 호스트:

hostA

인터노드 통신 포트:

16000(hostA 및 hostB용으로 사용되지 않는 포트)

주: 사이트 고유 값을 위의 필드로 대체하십시오. 예를 들어, 다음 표에는 지원되는 각 ESE 플랫폼에 대한 경로 이름의 예가 들어 있습니다.

표 2. 지원되는 ESE 플랫폼의 경로 이름 예

경로	AIX용 DB2 UDB ESE	Solaris 운영 환경용 DB2 UDB ESE	Windows용 DB2 UDB ESE
install_path	/usr/opt/<v_r_ID>	/opt/IBM/db2/<v_r_ID>	C:\sqllib
instance_path	/home/db2inst/sqllib	/home/db2inst/sqllib	C:\profiles\db2inst
das_path	/home/db2as/das	/home/db2as/das	C:\profiles\db2as
tcp_services_file	/etc/services	/etc/services	C:\winnt\system32 \drivers\etc\services

테이블에서 <v_r_ID>는 플랫폼 고유 버전 및 릴리스 ID입니다. 예를 들어, 버전 8의 AIX용 DB2 UDB ESE에서 <v_r_ID>는 db2_08_01입니다.

DB2 UDB ESE를 설치할 때, 설치 프로그램이 인스턴스 소유 머신에 DAS를 작성합니다. 데이터베이스 파티션 서버는 DAS와 같은 머신에 있으며 인스턴스에 대한 연결 지점이 됩니다. 즉, 이 데이터베이스 파티션 서버는 제어 센터 또는 구성 지원 프로그램에서 인스턴스에 발행된 요청에 대한 코디네이터 노드입니다.

각 실제 머신에 DAS를 설치하면 각 머신이 코디네이터 노드로서 동작할 수 있습니다. 각 실제 머신은 제어 센터 또는 구성 지원 프로그램에서 별도의 DB2SYSTEM으로 나타납니다. 서로 다른 클라이언트가 서로 다른 시스템을 사용하여 파티션된 데이터베이스 서버에 연결하면, 코디네이터 노드 업무가 분산되어 수신 연결의 균형을 조절하는데 도움이 됩니다.

관련 개념:

- 47 페이지의『DB2 Administration Server』
- 66 페이지의『ESE(Enterprise Server Edition) 시스템에서 DAS 구성』

ESE(Enterprise Server Edition) 시스템에서 DAS 구성

DAS는 제어 센터 대신 특정 태스크를 수행하는 관리 제어점입니다. 실제 머신당 최대 하나의 DAS가 있을 수 있습니다. 몇 개의 머신으로 구성되는 ESE 인스턴스의 경우, 제어 센터가 ESE 인스턴스를 관리할 수 있도록 하려면 모든 머신이 DAS를 실행해야 합니다. 이 DAS(db2as)는 제어 센터 네비게이터 트리에서 목표 DB2® Universal Database(DB2 UDB) 인스턴스(db2inst)의 상위 인스턴스로서 존재하는 시스템으로 표현됩니다.

예를 들어, db2inst는 두 개의 실제 머신 또는 호스트에 분산되는 세 개의 노드로 구성됩니다. hostA 및 hostB에서 **db2as**를 실행함으로써 최소 요구사항을 충족시킬 수 있습니다.

주:

1. 호스트 A에 있는 파티션 수는 해당 호스트에서 수행될 수 있는 DAS의 수와는 상관이 없습니다. 해당 호스트에 대한 다중 논리 노드(MLN) 구성과 상관없이 hostA에서 하나의 DAS 사본만을 실행할 수 있습니다.
2. 각 머신 또는 실제 노드에는 하나의 DAS가 필요하며, **dascrt** 명령을 사용하여 개별적으로 작성해야 합니다. 태스크 센터 및 제어 센터가 제대로 작업을 수행하려면 각 머신 또는 실제 노드 상의 DAS가 실행 중이어야 합니다. ID db2as는 hostA 및 hostB에 있어야 합니다. db2as ID의 홈 디렉토리가 두 시스템 간에 교차 설치되지 않도록 해야 합니다. 또는 서로 다른 사용자 ID를 사용하여 hostA 및 hostB에 DAS를 작성할 수도 있습니다.

Windows®용 DB2 Universal Database (DB2 UDB) Enterprise Server Edition에서 구성 지원 프로그램 또는 제어 센터를 사용하여 DB2 UDB 서버에 대한 연결 구성을 자동화할 경우에는, DAS와 동일한 머신에 있는 데이터베이스 파티션 서버가 코디네이터 노드가 됩니다. 이는 클라이언트에서 데이터베이스로의 모든 실제 연결이 다른 데이터베이스 파티션 서버로 경로 지정되기 전에 코디네이터 노드로 경로 지정됨을 의미합니다.

Windows용 DB2 Universal Database (DB2 UDB) Enterprise Server Edition에서 다른 머신에 추가 DB2 Administration Server를 작성하면, 구성 지원 프로그램 또는 제어 센터가 DB2 발견을 사용하여 다른 시스템을 코디네이터 노드로 구성할 수 있습니다.

Windows용 DB2 Universal Database (DB2 UDB) Enterprise Server Edition에서 작업할 경우, DB2 UDB 리모트 명령 서비스(**db2rcmd.exe**)는 인터노드 관리 통신을 자동으로 처리합니다.

제어 센터는 523 TCP 서비스 포트를 사용하여 DAS와 통신합니다. 이 포트는 DB2 UDB 전용으로 예약되어 있습니다. 따라서 TCP 서비스 파일에 새 항목을 삽입하지 않아도 됩니다.

태스크 관련:

- 49 페이지의『DB2 Administration Server 작성』

관련 참조:

- *Command Reference*의『db2admin - DB2 Administration Server Command』

관리 서버, 인스턴스 및 데이터베이스 발견

리모트 머신으로의 연결을 구성하는데에는 두 가지 방법이 있습니다. 구성 지원 프로그램에 빌드된 발견 서비스를 사용하거나 LDAP(Lightweight Directory Access Protocol)와 같은 기존 디렉토리 서비스를 사용할 수 있습니다.

발견 서비스는 구성 지원 프로그램 및 DB2® Administration Server와 통합됩니다. 리모트 머신으로의 연결을 구성하기 위해, 사용자는 클라이언트 머신에 로그온하여 구성 지원 프로그램(CA)을 실행합니다. CA는 네트워크에 있는 모든 머신에 브로드캐스트 신호를 보냅니다. DAS가 설치되어 발견용으로 구성된 머신이 그 머신에 대한 모든 인스턴스 및 데이터베이스 정보가 들어 있는 패키지를 다시 보냄으로써 CA의 브로드캐스트 신호에 응답합니다. 그러면 CA는 이 패키지에 있는 정보를 사용하여 클라이언트 연결을 구성합니다. 발견 메소드를 사용하면 로컬 데이터베이스 및 노드 디렉토리에 리모트 서버에 대한 카탈로그 정보를 자동으로 생성할 수 있습니다.

발견 메소드를 사용할 경우에는 각 클라이언트 머신에 로그온하여 CA를 실행해야 합니다. 따라서 클라이언트 수가 많은 환경일 경우에는 이를 수행하기가 어려우며 많은 시간이 소요됩니다. 이러한 경우에는 LDAP와 같은 디렉토리 서비스를 사용할 수 있습니다.

| 알려진 발견(Known Discovery)을 통해 클라이언트에 알려진 시스템상의 인스턴스와 데
| 이터베이스를 발견할 수 있으며, 인스턴스와 데이터베이스를 발견할 수 있도록 새 시스
| 템을 추가할 수 있습니다. 검색 발견(Search Discovery)은 알려진 발견(Known
| Discovery)의 모든 기능을 제공하고 옵션을 추가하여 로컬 네트워크에서 다른 DB2
| Universal Database™(DB2 UDB) 서버를 검색할 수 있게 해줍니다.

시스템이 알려진 발견을 지원하도록 하려면 DAS 구성 파일에 있는 *discover* 매개변수를 KNOWN으로 설정하십시오. 시스템이 알려진 발견 및 검색 발견 둘 다를 지원하도록 하려면 DAS 구성 파일에 있는 *discover* 매개변수를 SEARCH로 설정하십시오(이것이 디폴트입니다). 시스템과 시스템의 모든 인스턴스 및 데이터베이스의 발견을 불가능하게 하려면 이 매개변수를 DISABLE로 설정하십시오. DAS 구성 파일에서 *discover* 매개변수를 DISABLE로 설정하면 시스템의 발견을 막을 수 있습니다.

주: 검색 발견으로 클라이언트에 리턴된 TCP/IP 호스트 이름은 **hostname** 명령을 입력할 때 DB2 UDB 서버 시스템으로 리턴된 것과 같은 호스트 이름입니다. 클라이언트에서 이 호스트 이름이 맵핑되는 IP 주소는 클라이언트 머신에서 구성된 TCP/IP 도메인 이름 서버(DNS) 또는 DNS가 구성되지 않은 경우, 클라이언트의 *hosts* 파일에 있는 맵핑 항목으로 판별됩니다. DB2 UDB 서버 시스템에 다중 어댑터 카드가 구성된 경우, 서버에 TCP/IP가 구성되어 있어야만 올바른 호스트 이름을 리턴할 수 있으며, DNS 또는 로컬 클라이언트의 *hosts* 파일이 호스트 이름을 원하는 IP 주소에 맵핑할 수 있습니다.

클라이언트에서 발견은 *discover* 매개변수를 사용하여 완료되기도 합니다. 그러나 이 경우, *discover* 매개변수는 다음과 같이 클라이언트 인스턴스(또는 클라이언트로서 작동하는 서버)에 설정됩니다.

- **KNOWN**

  KNOWN 발견은 구성 지원 프로그램 및 제어 센터가 로컬 시스템에 이미 알려진 시스템과 연관된 인스턴스 및 데이터베이스 정보를 검색하는데 사용됩니다. 도구에서 제공되는 **시스템 추가** 기능을 사용하여 새 시스템을 추가할 수 있습니다. *discover* 매개변수가 KNOWN으로 설정되면 네트워크를 검색할 수 없습니다.

- **SEARCH**

  알려진 발견의 모든 기능을 사용 가능화하고 로컬 네트워크 검색을 사용 가능화합니다. 이는 모든 검색이 로컬 네트워크로 제한됨을 의미합니다.

  이것이 선택되면 『기타 시스템(네트워크 검색)』 아이콘만이 표시됩니다. 이는 디폴트 설정입니다.

- **DISABLE**

  발견을 사용하지 않게 합니다. 이 경우, **네트워크 검색** 옵션을 『데이터베이스 마법사 추가』에서 사용할 수 없습니다.

주: *discover* 매개변수는 모든 클라이언트와 서버 인스턴스에서 디폴트값이 SEARCH가 됩니다. 모든 DB2 Administration Server에서 *discover* 매개변수의 디폴트는 SEARCH입니다.

관련 개념:

- 83 페이지의 『LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스』

태스크 관련:

- 69 페이지의 『발견으로부터 서버 인스턴스 및 데이터베이스 숨기기』
- 69 페이지의 『발견 매개변수 설정』

발견으로부터 서버 인스턴스 및 데이터베이스 숨기기

프로시저:

서버 시스템에 여러 인스턴스가 있고 이러한 인스턴스에 여러 데이터베이스가 있을 수도 있습니다. 이 중 일부를 발견 프로세스가 찾지 못하도록 숨길 수 있습니다.

클라이언트가 시스템에서 서버 인스턴스를 발견하도록 하려면, 시스템의 각 서버 인스턴스에 있는 *discover_inst* 데이터베이스 관리 프로그램 구성 매개변수를 ENABLE(디폴트값)로 설정하십시오. 이 매개변수를 DISABLE로 설정하면 이 인스턴스와 데이터베이스를 발견에서 숨길 수 있습니다.

클라이언트가 데이터베이스를 발견할 수 있도록 하려면 *discover_db* 데이터베이스 구성 매개변수를 ENABLE로 설정하십시오(이것이 디폴트값입니다). 이 매개변수를 DISABLE로 설정하면 데이터베이스를 발견에서 숨길 수 있습니다.

주: 인스턴스를 발견할 수 있도록 하려면 DAS 구성 파일에서 discover 역시 KNOWN 또는 SEARCH로 설정해야 합니다. 데이터베이스를 발견할 수 있도록 하려면 서버 인스턴스에서 *discover_inst* 매개변수도 사용 가능으로 설정해야 합니다.

관련 참조:

- *관리 안내서: 성능*의 『discover_inst - 발견 서버 인스턴스 구성 매개변수』
- *관리 안내서: 성능*의 『discover_db - 발견 데이터베이스 구성 매개변수』

발견 매개변수 설정

프로시저:

서버 시스템에서는 DAS 구성 파일에서 클라이언트에서는 데이터베이스 관리 프로그램 구성 파일에서 *discover* 매개변수를 설정합니다. 구성 지원 프로그램 또는 제어 센터를 사용하여 데이터베이스 관리 프로그램 구성 매개변수 *discover*, *discover_inst* 및 *discover_db*를 설정하십시오. 매개변수를 다음과 같이 설정하십시오.

- DAS에서

  다음 명령 프로세스를 사용하여 DAS 구성 파일에 있는 *discover* 매개변수를 갱신하십시오.

  ```
  update admin cfg using discover [ DISABLE | KNOWN |
     SEARCH ]
  ```

  DAS *discover* 구성 매개변수는 온라인으로 구성 가능하며, 이는 변경사항을 적용하기 위해 DAS를 중지하여 재시작하지 않아도 됨을 의미합니다.

  주: 검색 발견은 TCP/IP에서만 작동합니다.

- 구성 지원 프로그램에 대한 작업.

  구성 지원 프로그램을 시작하려면 명령행(모든 플랫폼의) 또는 시작 메뉴(Windows의)에서 **db2ca**를 입력하십시오. **시작 —> 프로그램 —> IBM DB2 —> 설정 도구 —> 구성 지원 프로그램**을 누르십시오.

  구성 지원 프로그램을 사용하여 데이터베이스 관리 프로그램 구성 매개변수를 설정하십시오.

  1. **구성 —> DBM 구성**을 누르십시오.
  2. 수정할 키워드를 누르십시오.
  3. 값 컬럼에서, 수정할 키워드의 값을 누르고, **확인**을 누르십시오.
  4. 다시 **확인**을 누르고, 메시지가 표시됩니다. **닫기**를 누르십시오.

제어 센터를 사용하여 *discover_inst* 및 *discover_db* 매개변수를 설정하십시오.

구성 지원 프로그램을 사용하여 구성 매개변수를 갱신할 수도 있습니다.

관련 참조:

- *관리 안내서: 성능*의 『discover_inst - 발견 서버 인스턴스 구성 매개변수』
- *관리 안내서: 성능*의 『discover_db - 발견 데이터베이스 구성 매개변수』
- *Command Reference*의 『UPDATE ADMIN CONFIGURATION Command』
- *관리 안내서: 성능*의 『discover - DAS 발견 모드 구성 매개변수』

DAS를 설정하여 구성 지원 프로그램 및 제어 센터 사용

전제조건:

네트워크에 있는 시스템에 관한 정보를 검색하려면 **discover**를 구성해야 합니다.

제한사항:

각 실제 파티션에 DAS가 있어야 합니다. 파티션에 DAS를 작성하면, DB2SYSTEM 이름이 TCP/IP 호스트 이름에 구성되고 **discover** 설정이 디폴트값 SEARCH로 설정됩니다.

프로시저:

DB2 발견은 구성 지원 프로그램 및 제어 센터에서 사용되는 기능입니다. 이 기능을 구성할 때는 DAS(DB2 Administration Server) 구성 및 인스턴스의 데이터베이스 관리 프로그램 구성을 갱신하여 DB2 발견이 올바른 정보를 검색하도록 해야 합니다.

클라이언트가 구성 지원 프로그램이나 제어 센터에서 발견 요청을 발행하면 사용 가능한 발견이 있는 각 DAS가 응답합니다. 파티션된 데이터베이스 환경에서는 각 파티션이 별도의 DB2SYSTEM 이름으로서 응답합니다. 관리 가능한 실제 인스턴스는 실제 파티션에 알려진 인스턴스에 따라 다릅니다. 인스턴스는 여러 파티션에 걸쳐 있을 수 있으므로, 동일한 인스턴스가 잠재적으로 서로 다른 시스템 이름을 통해 관리될 수 있습니다. 이 기능을 사용하면 서버 인스턴스에서의 로드 균형을 이루는데 도움이 될 수 있습니다. 예를 들어, 시스템 『S1』 및 시스템 『S2』를 통해 인스턴스 『A』를 사용할 수 있는 경우, 일부 사용자는 『S1』을 사용하여 데이터베이스를 카탈로그화하고 일부는 『S2』를 사용하여 동일한 데이터베이스를 카탈로그화할 수 있습니다. 각 사용자는 다른 코디네이터 데이터베이스 파티션을 사용하여 서버에 연결할 수 있습니다.

관련 참조:

- *Command Reference*의 『db2ilist - List Instances Command』
- *Command Reference*의 『db2ncrt - Add Database Partition Server to an Instance Command』
- *관리 안내서: 성능*의 『discover - DAS 발견 모드 구성 매개변수』

discovery에 대한 DAS 구성 갱신

제한사항:

각 실제 파티션에 DAS가 있어야 합니다. 파티션에 DAS를 작성하면, DB2SYSTEM 이름이 TCP/IP 호스트 이름에 구성되고 *discover* 설정이 디폴트값 SEARCH로 설정됩니다.

프로시저:

발견으로 검색되는 시스템 이름은 DAS(DB2 Administration Server)가 상주하는 시스템입니다. 발견은 연결이 설정될 때 이들 시스템을 코디네이터 노드로 사용합니다.

DAS 구성을 갱신할 때 DB2 Universal Database™(DB2 UDB) 시스템 목록에서 코디네이터 노드를 선택할 수 있도록 하려면, 각 DB2 관리 서버의 구성 파일에서 *discover*=SEARCH(디폴트)를 설정하십시오.

파티션된 데이터베이스 서버 환경에 둘 이상의 DAS가 있으면, 구성 지원 프로그램 또는 제어 센터의 인터페이스에서 둘 이상의 시스템에 동일한 인스턴스가 나타날 수 있습니다. 그러나 각 시스템은 인스턴스에 대해 서로 다른 통신 액세스 경로를 가집니다. 사용자는 통신에 대해 다른 DB2 UDB 시스템을 코디네이터 노드로 선택할 수 있으므로 워크로드를 재분배할 수 있습니다.

관련 참조:

- *관리 안내서: 성능*의 『기타 변수』

DB2 Administration Server 첫 번째 실패 데이터 캡처

첫 번째 실패 데이터 캡처(FFDC)는 오류 발생 시 DB2®Administration Server가 자동으로 캡처하는 진단 정보 세트에 적용되는 일반 용어입니다. 이 정보를 사용할 경우, 진단 정보를 얻기 위해 오류를 재생해야 하는 필요성이 줄어듭니다. 진단 정보는 단일 위치에 저장됩니다.

DB2 Administration Server FFDC에서 캡처하는 정보에는 다음이 포함됩니다.

- 관리 통지 로그

  이벤트가 발생하면 DB2 Administration Server는 DB2 Administration Server 로그 파일 db2dasdiag.log에 정보를 기록합니다.

- 덤프 파일

  몇몇 오류 조건에서는 실패한 프로세스 ID의 이름을 딴 외부 2진 덤프 파일에 추가 정보가 로그됩니다. 이러한 파일은 DB2 Universal Database™(DB2 UDB) 고객 지원에서 사용하도록 하기 위한 것입니다.

- 트랩 파일

  DB2 Administration Server는 트랩, 세그멘테이션 위반 또는 예외로 인해 처리를 계속할 수 없는 경우 트랩 파일을 생성합니다. 트랩 파일에는 문제점이 발생하기 전에 실행되었던 마지막 단계의 함수 흐름이 포함됩니다.

DB2 Administration Server 첫번재 실패 데이터 캡처 정보 위치

디폴트로 DB2 Administration Server FFDC 정보는 다음 위치에 놓입니다.

- Windows® 시스템에서

  DB2INSTPROF 환경 변수가 설정되어 있지 않는 경우,

  ```
  db2path\DB2DAS00\dump
  ```

  여기서, db2path는 DB2PATH 환경 변수에서 참조되는 경로이고 DB2DAS00은 DAS 서비스의 이름입니다. 아무런 인수 없이 **db2admin** 명령을 입력함으로써 DAS 이름을 확보할 수 있습니다.

  DB2INSTPROF 환경 변수가 설정되어 있는 경우,

  ```
  x:\db2instprof\DB2DAS00\dump
  ```

  여기서, x:는 DB2PATH 환경 변수에서 참조되는 드라이브이고 db2instprof는 인스턴스 프로파일 디렉토리이며 DB2DAS00은 DAS 서비스의 이름입니다.

- UNIX® 기반 시스템의 경우:

  ```
  $DASHOME/das/dump
  ```

여기서, $DASHOME은 DAS 서버의 홈 디렉토리입니다.

주: 덤프 디렉토리가 너무 커지지 않도록 정기적으로 이 디렉토리를 정리해야 합니다.

DB2 Administration Server 로그 해석

DB2 Administration Server 로그 파일(db2dasdiag.log)의 형식은 DB2 FFDC 로그 파일 db2diag.log의 형식과 유사합니다. db2dasdiag.log 파일을 해석하는 방법에 대해서는 문제점 해결 주제에 있는 관리 로그 해석에 관한 절을 참조하십시오.

관련 개념:

• 47 페이지의『DB2 Administration Server』

제 3 장 데이터베이스 작성

이 장에서는 사용자 데이터베이스 설계 구현의 일부일 수 있는 다양한 각 오브젝트에 대한 간단한 설명을 제공합니다.

이전 장은 데이터베이스를 작성하기 전에 알아야 하는 정보에 초점을 두었습니다. 또한 데이터베이스를 작성하기 전에 수행해야 하는 여러 주제와 태스크도 다루었습니다.

이 부분의 최종 장에서는 데이터베이스를 변경하기 전에 고려해야 하는 사항을 나타냅니다. 그리고 데이터베이스 오브젝트를 변경하거나 삭제하는 방법을 설명합니다.

데이터베이스 작성

전제조건:

데이터베이스를 작성하기 전, 데이터베이스의 컨텐츠, 레이아웃, 잠재적 크기 증가 및 사용을 설계하는데 충분한 시간을 들여야 합니다.

프로시저:

데이터베이스를 작성할 때, 다음과 같은 각 태스크가 수행됩니다.

- 데이터베이스가 필요로 하는 모든 시스템 카탈로그 테이블 설정
- 데이터베이스 복구 로그의 할당
- 데이터베이스 구성 파일 및 디폴트값이 설정됩니다.
- 데이터베이스 유틸리티를 데이터베이스에 바인딩

데이터베이스 특권(시스템 카탈로그 뷰에 있는 CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA 및 SELECT 특권)은 자동으로 PUBLIC에 권한 부여됩니다.

제어 센터를 사용하여 데이터베이스를 작성하려면 다음을 수행하십시오.

1. **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **데이터베이스** 폴더를 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 **작성 —> 표준** 또는 **작성 —> 자동 유지보수 사용**을 선택하십시오.
3. 이 태스크를 완료하려면 단계에 따르십시오.

연관된 주석 "Personnel DB for BSchiefer Co"와 함께 다음 명령행 처리기 명령은 디폴트 위치에 personl이라는 데이터베이스를 작성합니다.

```
CREATE DATABASE personl
  WITH "Personnel DB for BSchiefer Co"
```

데이터베이스를 사용할 때, 모든 구성 매개변수의 디폴트값을 승인하는 대신, 데이터베이스 구성을 돕는 구성 어드바이저의 사용을 요청할 수도 있습니다. **CREATE DATABASE** 명령에서 AUTOCONFIGURE 옵션을 사용하여 이를 수행할 수 있습니다.

```
CREATE DATABASE <database name>
      AUTOCONFIGURE
```

AUTOCONFIGURE 절에는 몇 개의 옵션이 있습니다. 파티션된 환경에서 데이터베이스를 작성할 때에는 AUTOCONFIGURE 절을 사용할 수 없습니다.

데이터베이스가 작성되는 동시에 세부 교착 상태 이벤트 모니터도 작성됩니다. 다른 모니터와 마찬가지로 이 이벤트 모니터와 연관된 오버헤드가 있습니다. 세부 교착 상태 이벤트 모니터를 원하지 않는 경우 다음의 명령을 사용하여 이벤트 모니터를 삭제할 수 있습니다.

```
DROP EVENT MONITOR db2detaildeadlock
```

이 이벤트 모니터가 사용한 디스크 스페이스의 양을 제한하기 위해 최대 출력 파일 수에 도달하면 이벤트 모니터가 비활성화되고 관리 통지 로그에 메시지가 기록됩니다. 더 이상 필요하지 않은 출력 파일을 제거하면 다음 데이터베이스 활성화시에 이벤트 모니터를 다시 활성화할 수 있습니다.

다른(예: 리모트) 데이터베이스 관리 프로그램 인스턴스에서 데이터베이스를 작성할 수 있습니다. 이러한 유형의 환경에서는 리모트 인스턴스를 비롯한 디폴트 인스턴스 이외의 인스턴스에 대해 인스턴스 레벨 관리를 수행할 수 있습니다.

관련 개념:

- *관리 안내서: 계획*의 『데이터베이스에 기록할 내용』
- 6 페이지의 『데이터베이스 관리 프로그램의 다중 인스턴스』
- 268 페이지의 『데이터베이스 권한』
- *관리 안내서: 계획*의 『추가 데이터베이스 설계 고려사항』

관련 참조:

- *Command Reference*의 『CREATE DATABASE Command』

초기 데이터베이스 파티션 그룹 정의

데이터베이스가 처음 작성될 때, 데이터베이스 파티션은 db2nodes.cfg 파일에 지정된 모든 파티션용으로 작성됩니다. 기타 파티션은 **ADD DBPARTITIONNUM** 및 **DROP DBPARTITIONNUM VERIFY** 명령으로 추가 또는 삭제할 수 있습니다.

세 개의 데이터베이스 파티션 그룹은 다음과 같이 정의됩니다.

- 시스템 카탈로그 테이블이 들어 있는 SYSCATSPACE 테이블 스페이스의 경우, IBMCATGROUP
- 데이터베이스 처리 중에 작성된 임시 테이블이 들어 있는 TEMPSPACE1 테이블 스페이스의 경우, IBMTEMPGROUP
- 디폴트로, 사용자 테이블과 인덱스가 들어 있는 USERSPACE1 테이블 스페이스의 경우, IBMDEFAULTGROUP

관련 개념:

- *관리 안내서: 계획*의 『데이터베이스 파티션 그룹』

관련 참조:

- *Command Reference*의 『ADD DBPARTITIONNUM Command』
- *Command Reference*의 『DROP DBPARTITIONNUM VERIFY Command』

초기 테이블 스페이스 정의

데이터베이스가 작성되면, 세 개의 테이블 스페이스가 정의됩니다.

- 시스템 카탈로그 테이블의 경우 SYSCATSPACE
- 데이터베이스 처리 동안 작성된 시스템 임시 테이블의 경우, TEMPSPACE1
- 사용자 정의 테이블 및 인덱스의 경우, USERSPACE1

주: 처음 데이터베이스를 작성할 때 사용자 임시 테이블 스페이스가 작성되지 않습니다.

CREATE DATABASE 명령을 사용하여 테이블 매개변수를 지정하지 않으면, 데이터베이스 관리 프로그램은 시스템 관리 스토리지(SMS) 디렉토리 컨테이너를 사용하여 이 테이블 스페이스를 작성합니다. 이러한 디렉토리 컨테이너는 데이터베이스용으로 작성된 서브디렉토리에 작성됩니다. 이러한 테이블 스페이스에 Extent 크기는 디폴트값으로 설정됩니다.

전제조건:

데이터베이스가 작성되어 있어야 하고 테이블 스페이스 작성 권한이 있어야 합니다.

프로시저:

제어 센터를 사용하여 초기 테이블 스페이스를 정의하려면, 다음을 수행하십시오.

1. **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **데이터베이스** 폴더를 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 **작성 —> 표준** 또는 **작성 —> 자동 유지보수 사용**을 선택하십시오.
3. 이 태스크를 완료하려면 단계에 따르십시오.

명령행을 사용하여 초기 테이블 스페이스를 정의하려면, 다음을 입력하십시오.

```
CREATE DATABASE <name>
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('<path>')
    EXTENTSIZE <value> PREFETCHSIZE <value>
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'<path>' 5000,
                               FILE'<path>' 5000)
    EXTENTSIZE <value> PREFETCHSIZE <value>
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('<path>')
  WITH "<comment>"
```

이러한 테이블 스페이스에 대해 디폴트 정의를 사용하지 않으려면, **CREATE DATABASE** 명령에 특성을 지정해야합니다. 예를 들어, 다음 명령은 Windows에서 데이터베이스를 작성할 때 사용될 수 있습니다.

```
CREATE DATABASE PERSONL
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                               FILE'd:\db2data\personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:\db2temp\personl')
  WITH "Personnel DB for BSchiefer Co"
```

이 예에서 각각의 초기 테이블 스페이스에 대한 정의는 명시적으로 제공됩니다. 디폴트 정의를 사용하지 않으려는 경우에는 테이블 스페이스에 대해 테이블 스페이스 정의를 지정하기만 하면 됩니다.

주: 파티션된 데이터베이스 환경에서 작업할 때는 컨테이너를 작성하거나 특정 파티션에 지정할 수 없습니다. 먼저, 디폴트 사용자 및 임시 테이블 스페이스로 데이터베이스를 작성해야 합니다. CREATE TABLESPACE문을 사용하여 필수 테이블 스페이스를 작성할 수 있습니다. 마지막으로 디폴트 테이블 스페이스를 삭제할 수 있습니다.

CREATE DATABASE 명령의 MANAGED BY절 코드 형식은 CREATE TABLESPACE 명령의 MANAGED BY절과 동일합니다.

관련 개념:

- 80 페이지의『시스템 카탈로그 테이블 정의』
- *관리 안내서: 계획*의『테이블 스페이스 디자인』

태스크 관련:

- 90 페이지의『테이블 스페이스 작성』

관련 참조:

- *Command Reference*의 『CREATE DATABASE Command』

버퍼 풀 작성

데이터베이스 관리자가 사용할 버퍼 풀을 새로 작성할 수 있습니다. 버퍼 풀은 데이터베이스 시스템 성능을 즉시 향상시킵니다.

테이블 스페이스에 맞게 지정된 페이지 *크기*를 통해 선택된 사용자 버퍼 풀의 페이지 *크기*를 판별합니다. 버퍼 풀에 사용할 페이지 *크기* 선택이 중요한 이유는 버퍼 풀을 작성한 후에는 페이지 *크기*를 변경할 수 없기 때문입니다.

전제조건:

명령문의 권한 부여 ID에는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

버퍼 풀을 새로 작성하기 전에 다음 질문에 답하십시오.

- 어떤 버퍼 풀 이름을 사용할 것인가
- 버퍼 풀을 즉시 작성할 것인가 아니면 데이터베이스가 비활성화 및 활성화된 다음 작성할 것인가
- 데이터베이스를 구성하는 모든 데이터베이스 파티션의 서브세트와 버퍼 풀을 연관시킬 것인가
- 페이지 수를 기초로 한 총 버퍼 풀 *크기* 및 페이지 *크기*를 포함해서 버퍼 풀 *크기*를 제어하는 매개변수와 연관시킬 값은 무엇인가
- 사용하려는 것이 확장 스토리지인가, 블록 기반 지원인가 또는 모두 사용하지 않을 것인가

프로시저:

새 버퍼 풀을 작성하려면 다음을 수행하십시오.

1. `SELECT BPNAME FROM SYSCAT.BUFFERPOOLS`로 이미 데이터베이스에 있는 버퍼 풀 이름 목록을 가져오십시오.
2. 현재 결과 목록에 표시되지 않은 버퍼 풀 이름을 선택하십시오. 이름이 『SYS』 또는 『IBM』과 같은 문자로 시작해서는 안 됩니다.
3. 작성하려는 버퍼 풀의 특성을 판별하십시오.
4. CREATE BUFFERPOOL문 실행에 필요한 올바른 권한 부여 ID가 있는지 확인하십시오.
5. CREATE BUFFERPOOL문을 실행하십시오.

태스크 관련:

- 193 페이지의 『버퍼 풀 변경』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE BUFFERPOOL문』

시스템 카탈로그 테이블 정의

시스템 카탈로그 테이블 세트는 각 데이터베이스에 대해 작성되고 유지보수됩니다. 이 테이블에는 데이터베이스 오브젝트(예: 테이블, 뷰, 인덱스, 패키지) 정의에 대한 정보와 이 오브젝트에 대해 사용자가 가진 액세스 유형에 대한 보안 정보가 들어 있습니다. 이들 테이블은 SYSCATSPACE 테이블 스페이스에 저장됩니다.

이들 테이블은 데이터베이스를 조작하는 동안에 갱신됩니다(예를 들어, 테이블이 작성될 때). 사용자가 이들 테이블을 명시적으로 작성하거나 제거할 수는 없지만, 해당 내용을 쿼리하거나 볼 수는 있습니다. 데이터베이스가 작성되면, 시스템 카탈로그 테이블 오브젝트 이외에 다음 데이터베이스 오브젝트가 시스템 카탈로그에 정의됩니다.

- SYSIBM, SYSFUN 및 SYSPROC 스키마의 루틴 세트(함수 및 프로시저).
- 시스템 카탈로그 테이블에 대한 읽기 전용 뷰의 세트가 SYSCAT 스키마에 작성됩니다.
- 갱신 가능한 카탈로그 뷰의 세트가 SYSSTAT 스키마에 생성됩니다. 이러한 갱신 가능한 뷰를 사용함으로써, 특정 통계 정보를 갱신하여 가상 데이터베이스의 성능을 조사하거나 **RUNSTATS** 유틸리티를 사용하지 않고 통계를 갱신할 수 있습니다.

데이터베이스를 작성한 후에는 시스템 카탈로그 뷰에 대한 액세스를 제한하고자 할 수도 있습니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『사용자 정의 함수』
- *SQL 참조서, 볼륨 1*의 『카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『함수 개요』

태스크 관련:

- 294 페이지의 『시스템 카탈로그 뷰 보안』

관련 참조:

- *SQL 참조서, 볼륨 1*의 『함수』

데이터베이스 디렉토리의 정의

새 데이터베이스를 설치하거나 설정할 때에는 세 개의 디렉토리가 사용됩니다.

- 로컬 데이터베이스 디렉토리
- 시스템 데이터베이스 디렉토리
- 노드 디렉토리

로컬 데이터베이스 디렉토리

로컬 데이터베이스 디렉토리 파일은 데이터베이스가 정의된 각 경로(Windows® 운영 체제에서는 『드라이브』라고 함)에 있습니다. 이 디렉토리에는 해당 위치에서 액세스할 수 있는 각 데이터베이스에 대해 하나의 항목이 들어 있습니다. 각 항목에는 다음 항목이 들어 있습니다.

- **CREATE DATABASE** 명령에서 제공된 데이터베이스 이름
- 데이터베이스 별명 이름(별명 이름이 지정되지 않으면 동일한 데이터베이스 이름으로 대신함)
- **CREATE DATABASE** 명령에서 제공된 데이터베이스를 설명하는 주석
- 데이터베이스에 대한 루트 디렉토리의 이름
- 기타 시스템 정보

관련 참조:

- *Command Reference*의 『CREATE DATABASE Command』

시스템 데이터베이스 디렉토리

시스템 데이터베이스 디렉토리 파일은 데이터베이스 관리 프로그램의 인스턴스마다 존재하며, 이 인스턴스용으로 카탈로그화된 각 데이터베이스 항목이 하나 들어 있습니다. 데이터베이스는 **CREATE DATABASE** 명령을 실행할 때 내재적으로 카탈로그화되며, **CATALOG DATABASE** 명령을 사용하여 명시적으로 카탈로그화할 수도 있습니다.

작성된 각 데이터베이스의 경우, 항목은 다음 정보가 들어 있는 디렉토리에 추가됩니다.

- **CREATE DATABASE** 명령에서 제공된 데이터베이스 이름
- 데이터베이스 별명 이름(별명 이름이 지정되지 않으면 동일한 데이터베이스 이름으로 대신함)
- **CREATE DATABASE** 명령에서 제공된 데이터베이스 주석
- 로컬 데이터베이스 디렉토리의 위치
- 데이터베이스가 *간접적*임을 나타내는 표시기. 간접적이란 데이터베이스가 현재 데이터베이스 관리 프로그램 인스턴스에 존재함을 의미합니다.
- 기타 시스템 정보

UNIX® 플랫폼 및 파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션이 항상 해당 인스턴스 홈 디렉토리의 `sqldbdir` 서브디렉토리에 있는 동일한 시스템 데이터베이스 디렉토리 파일 `sqldbdir`에 액세스하도록 해야 합니다. 동일한 `sqldbdir` 서브디렉토리에 있는 시스템 데이터베이스 디렉토리 또는 시스템 인텐션 파일 `sqldbins`가 공유 파일 시스템에 있는 다른 파일에 대한 기호 링크일 경우, 예상치 못한 오류가 발생할 수 있습니다.

태스크 관련:

- 13 페이지의『데이터베이스에서 데이터 파티션 사용』
- 87 페이지의『데이터베이스 카탈로그화』

관련 참조:

- *Command Reference*의『CREATE DATABASE Command』

데이터베이스의 대체 서버 식별

서버가 손상될 때마다, 해당 서버에 연결된 각각의 클라이언트는 연결이 종료되고 결과적으로 응용프로그램 오류를 야기하는 통신 오류를 수신합니다. 사용 가능성이 중요한 경우, 대기 노드로 서버를 페일오버하기 위한 별도의 설정 또는 기능을 구현해야 합니다. 두 가지 모두, DB2 Universal Database™(DB2 UDB) 클라이언트 코드는 페일오버 노드(IP 주소 또한 페일오버됨)에서 실행 중일 수도 있는 원래 서버로 또는 새로운 서버로 연결을 재설정하려고 시도합니다.

프로시저:

새로운 또는 대체 서버를 정의하려면, UPDATE ALTERNATE SERVER FOR DATABASE 명령을 사용하십시오. 이 명령은 시스템 데이터베이스 디렉토리에서 데이터베이스 별칭에 대한 대체 서버 정보를 갱신합니다.

관련 개념:

- 86 페이지의『자동 클라이언트 재라우트 구현』
- 353 페이지의『자동 클라이언트 재라우트 설명 및 설정』

로컬 또는 시스템 데이터베이스 디렉토리 파일 보기

시스템에 있는 데이터베이스와 연관된 정보를 보고자 합니다.

전제조건:

로컬 또는 시스템 데이터베이스 디렉토리 파일을 보려면 인스턴스 및 데이터베이스가 미리 작성되어 있어야 합니다.

프로시저:

로컬 데이터베이스 디렉토리 파일의 컨텐츠를 보려면 다음 명령을 발행하십시오. <location>은 데이터베이스의 위치를 지정합니다.

```
LIST DATABASE DIRECTORY ON <location>
```

시스템 데이터베이스 디렉토리 파일의 컨텐츠를 보려면 데이터베이스 디렉토리의 위치를 지정하지 않고 **LIST DATABASE DIRECTORY** 명령을 발행하십시오.

관련 참조:

- *Command Reference*의 『LIST DATABASE DIRECTORY Command』

노드 디렉토리

데이터베이스 관리 프로그램은 첫 번째 데이터베이스 파티션이 카탈로그화될 때 노드 *디렉토리*를 작성합니다. 데이터베이스 파티션을 카탈로그화하려면 **CATALOG NODE** 명령을 사용하십시오. 로컬 노드 디렉토리의 컨텐츠를 나열하려면 **LIST NODE DIRECTORY** 명령을 사용하십시오. 각 데이터베이스 클라이언트에서 노드 디렉토리가 작성되고 유지보수됩니다. 디렉토리에는 클라이언트가 액세스할 수 있는 하나 이상의 데이터베이스를 갖는 각 리모트 워크스테이션에 대한 항목이 들어 있습니다. DB2® 클라이언트는 데이터베이스 연결 또는 인스턴스 접속이 요청될 때마다 노드 디렉토리의 통신 끝점 정보를 사용합니다.

또한, 디렉토리의 항목에는 클라이언트에서 리모트 데이터베이스 파티션으로 통신하는 데 사용되는 통신 프로토콜 유형에 대한 정보도 들어 있습니다. 로컬 데이터베이스 파티션을 카탈로그화하면 동일한 머신에 상주하는 인스턴스의 별명도 작성됩니다.

관련 참조:

- *Command Reference*의 『CATALOG TCPIP NODE Command』
- *Command Reference*의 『LIST NODE DIRECTORY Command』
- *Command Reference*의 『CATALOG NETBIOS NODE Command』
- *Command Reference*의 『CATALOG LOCAL NODE Command』
- *Command Reference*의 『CATALOG NAMED PIPE NODE Command』

LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스

디렉토리 서비스는 분산 환경 내의 다중 시스템과 서비스에 대한 자원 정보의 저장소이며, 이들 자원에 대한 클라이언트 및 서버 액세스를 제공합니다. 클라이언트 및 서버는 이 디렉토리 서비스를 사용하여 다른 자원에 액세스하는 방법을 알아냅니다. 분산 환경에 있는 다른 자원에 대한 정보는 디렉토리 서비스 저장소에 입력되어야 합니다.

LDAP(Lightweight Directory Access Protocol)는 디렉토리 서비스에 대한 산업 표준 액세스 방법입니다. 각 데이터베이스 서버 인스턴스는 LDAP 서버로 그 존재를 퍼블리시하며 데이터베이스가 작성될 때 LDAP 디렉토리로 데이터베이스 정보를 제공합니다. 클라이언트가 데이터베이스에 연결할 때, 카탈로그에 대한 정보는 LDAP 디렉토리에서 검색될 수 있습니다. 각 클라이언트는 각 머신에 로컬로 카탈로그 정보를 저장하는 것이 더 이상 필요하지 않습니다. 클라이언트 응용프로그램은 데이터베이스에 연결하는 데 필요한 정보 요구를 찾기 위해 LDAP 디렉토리를 검색합니다.

DB2® UDB 시스템 관리자는 디렉토리 서비스를 설정하고 유지보수할 수 있습니다. 구성 지원 프로그램 또는 제어 센터에서 이러한 디렉토리 서비스의 유지보수를 지원합니다. DB2 UDB에서는 LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스를 통해 디렉토리 서비스를 사용할 수 있습니다. LDAP 디렉토리 서비스를 사용하려면, 디렉토리 정보가 저장될 수 있도록 먼저 DB2가 지원하는 LDAP 서버가 있어야 합니다.

주: Windows® 2000 도메인 환경에서 실행할 경우에는 LDAP 서버가 Windows 2000 Active Directory에 통합되어 있으므로 LDAP 서버가 이미 사용 가능합니다. 따라서 Windows 2000을 실행 중인 각 머신이 LDAP를 사용할 수 있습니다.

LDAP 디렉토리는 클라이언트 수가 많아 각 클라이언트에서 로컬 디렉토리 카탈로그를 갱신하기 어려운 엔터프라이즈 환경에서 유용합니다. 이러한 상황에 처했을 때는 한 장소(LDAP 서버의)에서 카탈로그 항목을 유지보수할 수 있도록 하나의 LDAP 서버에 디렉토리 항목을 저장하는 것이 바람직합니다. LDAP 서버를 구입 및 유지보수하는데에는 상당한 비용이 들므로, 그러한 비용을 상쇄할 수 있을 만큼 충분한 수의 클라이언트가 있을 때에만 이를 고려하십시오.

관련 개념:

- 67 페이지의『관리 서버, 인스턴스 및 데이터베이스 발견』
- 359 페이지의『LDAP(Lightweight Directory Access Protocol) 입문』

데이터베이스 파티션 그룹(노드 그룹) 작성

CREATE DATABASE PARTITION GROUP문을 사용하여 데이터베이스 파티션 그룹을 작성하십시오. 이 명령문은 테이블 스페이스 컨테이너 및 테이블 데이터가 상주할 데이터베이스 파티션 세트를 지정합니다. 이 명령문은 또한 다음 기능을 수행합니다.

- 데이터베이스 파티션 그룹에 대해 파티션 맵을 작성합니다.
- 파티션 맵 ID를 생성합니다.
- 레코드를 다음 카탈로그 테이블에 삽입합니다.
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.DBPARTITIONGROUPDEF

전제조건:

머신 및 시스템이 사용 가능해야 하며 파티션된 데이터베이스 환경을 처리할 수 있어야 합니다. DB2 Universal Database Enterprise - Server Edition이 설치되어 있어야 합니다. 데이터베이스가 있어야 합니다.

프로시저:

제어 센터를 사용하여 데이터베이스 파티션 그룹을 작성하려면 다음을 수행하십시오.

1. 오브젝트 트리를 펼쳐 **데이터베이스 파티션 그룹** 폴더를 찾으십시오.
2. **데이터베이스 파티션 그룹** 폴더를 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 **작성**을 선택하십시오.
3. 데이터베이스 파티션 그룹 작성 창에서 정보를 완료하고 화살표를 사용하여 **사용 가능한 노드** 상자에서 **선택된 데이터베이스 파티션** 상자로 노드를 이동시킨 후 **확인**을 누르십시오.

명령행을 사용하여 데이터베이스 파티션 그룹을 작성하려면 다음을 입력하십시오.

```
CREATE DATABASE PARTITION GROUP <name> ON PARTITIONS (<value>,<value>)
```

데이터베이스에 있는 데이터베이스 파티션의 서브세트에 일부 테이블을 로드하고자 한다고 가정해 보십시오. 최소한 세 개의(0에서 2) 데이터베이스 파티션으로 구성되는 데이터베이스에서 두 개의 데이터베이스 파티션(1 및 2)으로 구성된 데이터베이스 파티션 그룹을 작성하려면 다음 명령을 사용할 것입니다.

```
CREATE DATABASE PARTITION GROUP mixng12 ON PARTITIONS (1,2)
```

CREATE DATABASE 명령 또는 sqlecrea() API도 디폴트 시스템 데이터베이스 파티션 그룹 IBMDEFAULTGROUP, IBMCATGROUP 및 IBMTEMPGROUP을 작성합니다.

관련 개념:

- *관리 안내서: 계획*의 『데이터베이스 파티션 그룹』
- *관리 안내서: 계획*의 『파티션 맵』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE DATABASE PARTITION GROUP문』
- *Administrative API Reference*의 『sqlecrea - Create Database』
- *Command Reference*의 『CREATE DATABASE Command』

데이터베이스 복구 로그 정의

*데이터베이스 복구 로그*는 새 테이블의 추가 또는 기존 테이블의 갱신을 포함한 데이터베이스에서 이루어진 모든 변경사항을 기록합니다. 이 로그는 다수의 *로그 Extent*로 구성되며, *로그 파일*인 분리 파일에 각각 들어 있습니다.

데이터베이스 복구 로그를 사용하여 장애(예: 시스템 전원 중단 또는 응용프로그램 오류)가 불일치 상태에서 데이터베이스를 떠나지 않은 것을 확인할 수 있습니다. 장애가 발생한 경우, 이미 내용이 변경되었으나 커미트되지 않은 변경사항은 롤백되고, 실제로 디스크에 작성되지 않은 커미트된 모든 트랜잭션은 다시 실행됩니다. 이러한 조치로 데이터베이스의 무결성을 확인할 수 있습니다.

관련 개념:

- *데이터 복구 및 고가용성 안내 및 참조서*의 『복구 로그 이해』

자동 클라이언트 재라우트 구현

DB2® Universal Database(DB2 UDB) 클라이언트 응용프로그램이 DB2 UDB 서버와의 통신 손실 문제점을 겪을 경우, 사용자는 클라이언트가 사용자나 또다른 관리자의 개입 없이 손실을 복구하도록 할 것입니다. DB2 UDB는 DB2 UDB 서버에 대한 클라이언트 통신의 복구를 지원합니다. 통신 실패 이전에 클라이언트 연결에서 인식하는 대체 위치 설정을 수행해야 합니다.

UPDATE ALTERNATE SERVER FOR DATABASE 명령은 특정 데이터베이스에 관한 대체 서버 위치를 정의하는 데 사용됩니다. 대체 호스트 이름 및 포트 번호가 명령의 파트로 제공됩니다. 위치는 서버의 시스템 데이터베이스 디렉토리 파일에 저장됩니다.

서버 인스턴스의 특정 데이터베이스에 대체 서버 위치를 지정한 후, 대체 서버 위치 정보는 연결 프로세스의 파트로 클라이언트에 리턴됩니다. 어떠한 이유로 클라이언트 및 서버 간 통신이 손실된 경우, DB2 UDB 클라이언트 코드가 대체 서버 정보를 사용하여 연결을 재설정하려고 시도합니다. DB2 UDB 클라이언트는 원래 서버 및 대체 서버와 다시 연결하려고 시도하며, 두 서버 중에서 번갈아가며 시도합니다. 이러한 시도의 타이밍은 다양하며 매우 빠른 시도에서 시작하여 시도 간의 간격을 점차적으로 늘립니다.

연결이 성공하면, 통신 실패로 데이터베이스 연결이 재설정되었음을 표시하기 위해 SQLCODE -30108이 리턴됩니다. 호스트 이름/IP 주소 및 서비스 이름/포트 번호가 리턴됩니다. 원래 또는 대체 서버로의 클라이언트 통신 재설정이 가능하지 않을 경우 클라이언트 코드는 원래 통신 실패에 대한 오류만을 응용프로그램으로 리턴합니다.

관련 개념:

- 353 페이지의 『자동 클라이언트 재라우트 설명 및 설정』
- 354 페이지의 『자동 클라이언트 재라우트 한계』

관련 참조:

- 356 페이지의 『자동 클라이언트 재라우트 예』

데이터베이스에 유틸리티 바인딩

데이터베이스가 작성되면 데이터베이스 관리 프로그램은 db2ubind.lst에 있는 유틸리티를 데이터베이스에 바인드하려고 합니다. 이 파일은 sqllib 디렉토리의 bnd 서브디렉토리에 저장됩니다.

유틸리티를 바인딩하면 *패키지*가 작성됩니다.

주: 클라이언트로부터 이들 유틸리티를 사용하려면, 유틸리티를 명시적으로 바인드해야 합니다.

유틸리티를 데이터베이스에 바인드하거나 리바인드해야 하는 경우, 명령행 처리기를 사용하여 다음 명령을 발행하십시오.

```
connect to sample
bind @db2ubind.lst
```

주: sample 데이터베이스에 패키지를 작성하려면 이들 파일이 상주하는 디렉토리 내에 있어야 합니다. 바인드 파일은 sqllib 디렉토리의 bnd 서브디렉토리에 있습니다. 이 예에서 sample은 데이터베이스 이름입니다.

태스크 관련:

- 75 페이지의『데이터베이스 작성』

관련 참조:

- *Command Reference*의『BIND Command』

데이터베이스 카탈로그화

새로운 데이터베이스를 작성하면, 시스템 데이터베이스 디렉토리 파일에 자동으로 카탈로그화됩니다. **CATALOG DATABASE** 명령을 사용하여 시스템 데이터베이스 디렉토리 파일에 명시적으로 데이터베이스를 카탈로그화할 수도 있습니다. **CATALOG DATABASE** 명령을 사용하면, 다른 별명 이름으로 데이터베이스를 카탈로그화하거나 **UNCATALOG DATABASE** 명령으로 이전에 삭제된 데이터베이스 항목을 카탈로그화할 수 있습니다.

전제조건:

데이터베이스를 작성할 때 자동으로 데이터베이스가 카탈로그화되더라도, 여전히 수동으로 데이터베이스를 카탈로그화해야 하는 경우가 있을 수 있습니다. 그러한 경우에는 데이터베이스가 있어야 합니다.

프로시저:

다음 명령행 처리기 명령은 personl 데이터베이스를 humanres로 카탈로그화합니다.

```
CATALOG DATABASE personl AS humanres
   WITH "Human Resources Database"
```

여기서, 시스템 데이터베이스 디렉토리 항목은 데이터베이스 별명으로서 humanres를 가지는데, 이는 데이터베이스 이름(personl)과는 다릅니다.

디폴트 인스턴스가 아닌 기타 인스턴스에서 데이터베이스를 카탈로그화할 수도 있습니다. 다음 예에서 데이터베이스 B에 대한 연결은 INSTNC_C에 대한 연결입니다. 이 명령을 시도하기 전에 인스턴스 instnc_c가 로컬 노드로서 이미 카탈로그화되어 있어야 합니다.

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

주: 데이터베이스 서버 머신에 있는 데이터베이스를 카탈로그화하기 위해 클라이언트 노드에서 **CATALOG DATABASE** 명령도 사용됩니다.

주: 디폴트로 데이터베이스 디렉토리를 비롯한 디렉토리 파일이 『디렉토리 캐시 지원 (*dir_cache*)』 구성 매개변수에 의해 메모리에 캐시됩니다. 디렉토리 캐싱이 사용되면, 다른 응용프로그램이 디렉토리에 대해 수행한 변경(예를 들어, **CATALOG DATABASE** 또는 **UNCATALOG DATABASE** 명령을 사용하여)이 응용프로그램을 재시작할 때까지 효력을 가지지 않을 수도 있습니다. 명령행 처리기 세션에서 사용된 디렉토리 캐시를 새로 고치려면 **db2 terminate** 명령을 발행하십시오.

파티션된 데이터베이스에서 디렉토리 파일용 캐시는 각 데이터베이스 파티션에서 작성됩니다.

응용프로그램 레벨 캐시뿐만 아니라, 데이터베이스 관리 프로그램 레벨 캐시도 내재적으로 데이터베이스 관리 프로그램을 찾아보는 데 사용됩니다. 이러한 『공유』 캐시를 새로 고치려면, **db2stop** 및 **db2start** 명령을 발행하십시오.

태스크 관련:

- 89 페이지의 『리모트 데이터베이스 서버 머신에 대한 정보가 들어 있는 디렉토리 갱신』

관련 참조:

- *관리 안내서: 성능*의 『dir_cache - 디렉토리 캐시 지원 구성 매개변수』
- *Command Reference*의 『CATALOG DATABASE Command』
- *Command Reference*의 『TERMINATE Command』
- *Command Reference*의 『UNCATALOG DATABASE Command』

리모트 데이터베이스 서버 머신에 대한 정보가 들어 있는 디렉토리 갱신

프로시저:

구성 지원 프로그램(CA) 인터프리터의 데이터베이스 추가 마법사를 사용하여 카탈로그 항목을 작성할 수 있습니다. DB2 Application Development Client가 있으면 항목을 카탈로그화하는 응용프로그램도 작성할 수 있습니다.

주: 데이터베이스를 카탈로그화하려면 SYSADM 또는 SYSCTRL 권한이 있거나 *catalog_noauth* 구성 매개변수를 YES로 설정해야 합니다.

명령행 처리기를 사용하여 디렉토리를 갱신하려면 다음을 수행하십시오.

1. 다음 명령 중 하나를 사용하여 노드 디렉토리를 갱신하십시오.
   - APPC 연결을 가지는 노드의 경우

     ```
     db2 CATALOG APPC NODE <nodename>
       REMOTE <symbolic_destination_name> SECURITY <security_type>
     ```

     예를 들어, 다음과 같습니다.

     ```
     db2 CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
     ```

   - TCP/IP 연결을 가지는 z/OS 및 OS/390용 DB2 Universal Database 버전 5.1(또는 그 이상) 또는 AS/400용 DB2 Universal Database 버전 4.2(또는 그 이상) 데이터베이스의 경우

     ```
     db2 CATALOG TCPIP NODE <nodename>
        REMOTE <hostname> or <IP address>
        SERVER <service_name> or <port_number>
        SECURITY <security_type>
     ```

     예를 들어, 다음과 같습니다.

     ```
     db2 CATALOG TCPIP NODE MVSIPNOD REMOTE MVSHOST SERVER DB2INSTC
     ```

     OS/390 및 z/OS용 DB2에서 TCP/IP 연결에 사용되는 디폴트 포트는 446입니다.
2. DB2 Connect를 사용할 경우에는 CATALOG DCS DATABASE 명령을 사용하여 DCS 디렉토리를 갱신해야 합니다.

리모트 클라이언트가 있는 경우에는 각 리모트 클라이언트에 있는 디렉토리도 갱신해야 합니다.

관련 개념:

- *DB2 Connect 사용자 안내서*의 『시스템 데이터베이스 디렉토리 값』
- *DB2 Connect 사용자 안내서*의 『DCS 디렉토리 값』

관련 참조:

- *Command Reference*의 『CATALOG DATABASE Command』
- *Command Reference*의 『CATALOG DCS DATABASE Command』
- *Command Reference*의 『CATALOG APPC NODE Command』
- *Command Reference*의 『CATALOG TCPIP NODE Command』
- *Command Reference*의 『CATALOG NETBIOS NODE Command』
- *Command Reference*의 『CATALOG APPN NODE Command』

테이블 스페이스 작성

테이블 스페이스는 데이터베이스 시스템이 사용하는 실제 스토리지 디바이스와 데이터를 저장하는데 사용되는 논리적 컨테이너 또는 테이블 사이의 관계를 설정합니다.

전제조건:

테이블 스페이스를 작성할 때 참조할 컨테이너의 디바이스 또는 파일 이름을 알고 있어야 합니다. 추가로, 테이블 스페이스에 할당될 각 디바이스 또는 파일 이름과 연관된 스페이스를 알고 있어야 합니다.

프로시저:

데이터베이스 내에서 테이블 스페이스를 작성하면 컨테이너가 테이블 스페이스에 지정되고 정의 및 속성이 데이터베이스 시스템 카탈로그에 기록됩니다. 그런 다음, 이 테이블 스페이스 내에 테이블을 작성할 수 있습니다.

제어 센터를 사용하여 테이블 스페이스를 작성하려면 다음을 수행하십시오.

1. **테이블 스페이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **테이블 스페이스** 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **작성 —> 마법사를 사용한 테이블 스페이스**를 선택하십시오.
3. 태스크를 완료하려면, 마법사의 단계에 따르십시오.

명령행을 사용하여 SMS 테이블 스페이스를 작성하려면 다음을 입력하십시오.

```
CREATE TABLESPACE <NAME>
   MANAGED BY SYSTEM
   USING ('<path>')
```

명령행을 사용하여 DMS 테이블 스페이스를 작성하려면 다음을 입력하십시오.

```
CREATE TABLESPACE <NAME>
   MANAGED BY DATABASE
   USING (FILE'<path>' <size>)
```

다음 SQL문은 세 개의 개별 드라이브에서 세 개의 디렉토리를 사용하여 Windows에 SMS 테이블 스페이스를 작성합니다.

```
CREATE TABLESPACE RESOURCE
   MANAGED BY SYSTEM
   USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

다음 SQL문은 각각 5,000페이지인 두 개의 파일 컨테이너를 사용하여 DMS 테이블 스페이스를 작성합니다.

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (FILE'd:\db2data\acc_tbsp' 5000,
          FILE'e:\db2data\acc_tbsp' 5000)
```

이전 두 예에서 컨테이너에 대해 명시적인 이름이 제공됩니다. 그러나 상대 컨테이너 이름을 지정하면, 데이터베이스용으로 작성된 서브디렉토리에 컨테이너가 작성됩니다.

또한 지정된 경로 이름의 부분이 존재하지 않으면, 데이터베이스 관리 프로그램이 작성합니다. 서브디렉토리가 데이터베이스 관리 프로그램에 의해 작성된 경우, 서브디렉토리는 테이블 스페이스가 제거될 때 데이터베이스 관리 프로그램에 의해 삭제될 수도 있습니다.

위 예에서는 테이블 스페이스가 특정 데이터베이스 파티션 그룹과 연관되지 않은 것으로 가정합니다. 명령문에 다음 매개변수를 지정하지 않으면 디폴트 데이터베이스 파티션 그룹 IBMDEFAULTGROUP이 사용됩니다.

```
IN database_partition_group_name
```

다음 SQL문은 각각 10,000페이지의 세 개의 논리 볼륨을 사용하여 UNIX 기반 시스템에 DMS 테이블 스페이스를 작성하고 입출력 특성을 지정합니다.

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (DEVICE '/dev/rdblv6' 10000,
          DEVICE'/dev/rdblv7' 10000,
          DEVICE'/dev/rdblv8' 10000)
   OVERHEAD 12.67
   TRANSFERRATE 0.18
```

이 SQL문에서 언급된 UNIX 디바이스가 이미 존재하고 있어야 하며, 인스턴스 소유자 및 SYSADM 그룹이 이를 작성할 수 있어야 합니다.

다음 예에서는 UNIX 파티션된 데이터베이스의 ODDGROUP이라는 데이터베이스 파티션에 DMS 테이블 스페이스를 작성합니다. 이전에 ODDGROUP이 CREATE DATABASE PARTITION GROUP문으로 작성되어 있어야 합니다. 이 경우, ODDGROUP 데이터베이스 파티션 그룹은 1, 3, 5로 번호가 매겨진 데이터베이스 파티션으로 구성된 것으로 가정됩니다. 모든 데이터베이스 파티션에서 10 000 4KB 페이지에 /dev/hdisk0 디바이스를 사용하십시오. 또한, 40 000 4KB 페이지의 각 데이터베이스 파티션에 대해서도 디바이스를 선언하십시오.

```
CREATE TABLESPACE PLANS IN ODDGROUP
   MANAGED BY DATABASE
   USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
          ON DBPARTITIONNUM 1
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
          ON DBPARTITIONNUM 3
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
          ON DBPARTITIONNUM 5
```

UNIX 디바이스는 두 개의 범주(문자 직렬 디바이스 및 블록 구조 디바이스)로 분류됩니다. 모든 파일 시스템 디바이스의 경우, 각 블록 디바이스(또는 *cooked* 디바이스)에 대해 해당하는 문자 직렬 디바이스(또는 *원시* 디바이스)를 갖는 것이 일반적입니다. 블록 구조 디바이스는 일반적으로 『hd0』 또는 『fd0』와 유사한 이름으로 지정됩니다. 문자 직렬 디바이스는 일반적으로 『rhd0』, 『rfd0』 또는 『rmt0』와 유사한 이름으로 지정됩니다. 이들 문자 직렬 디바이스는 블록 디바이스보다 액세스 속도가 빠릅니다. 문자 직렬 디바이스 이름은 CREATE TABLESPACE 명령에 사용되며, 블록 디바이스 이름은 그렇지 않습니다.

오버헤드 및 전송률은 SQL문이 컴파일될 때 사용할 최상의 액세스 경로를 판별하는 데 도움이 됩니다. 현재 디폴트값은 다음과 같습니다.

- OVERHEAD 12.67 ms
- TRANSFERRATE 0.18 ms

DB2 UDB는 병렬 I/O를 사용하는 시퀀스 프리페치 기능을 사용하여 순차적 입출력의 성능을 크게 개선할 수 있습니다.

또한 디폴트값인 4KB보다 큰 페이지 크기를 사용하는 테이블 스페이스를 작성할 수도 있습니다. 다음 SQL문은 UNIX 기반 시스템에 8KB 페이지 크기의 SMS 테이블 스페이스를 작성합니다.

```
CREATE TABLESPACE SMS8K
   PAGESIZE 8192
   MANAGED BY SYSTEM
   USING ('FSMS_8K_1')
   BUFFERPOOL BUFFPOOL8K
```

연관된 버퍼 풀 또한 동일한 8KB 페이지 크기여야 한다는 점을 유의하십시오.

참조하는 버퍼 풀이 활성화될 때까지는 작성된 테이블 스페이스를 사용할 수 없습니다.

ALTER TABLESPACE SQL문은 DMS 테이블 스페이스에(에서) 컨테이너를 추가(삭제)하거나 컨테이너의 크기를 조정하고 테이블 스페이스에 대한 PREFETCHSIZE, OVERHEAD 및 TRANSFERRATE 설정을 수정하는 데 사용할 수 있습니다. 테이블 스페이스 명령문을 발행하는 트랜잭션은 시스템 카탈로그 경합을 막기 위해 가능한 빨리 커미트되어야 합니다.

주: PREFETCHSIZE 값은 EXTENTSIZE 값의 배가 되어야 합니다. 예를 들어, EXTENTSIZE가 10이면 PREFETCHSIZE는 20 또는 30이 되어야 합니다. 테이블 스페이스를 작성할 경우 프리페치 *크기*를 수동으로 설정하려면 다음과 같은 등식을 사용해야 합니다.

```
prefetch size = (number of containers) X (number of physical spindles per
                container) X extent size
```

또한 DB2 UDB가 프리페치 *크기*를 자동으로 판별할 수 있도록 고려해야 합니다.

관련 개념:

- *관리 안내서: 계획*의 『테이블 스페이스 디자인』
- *관리 안내서: 계획*의 『시스템 관리 스페이스』
- *관리 안내서: 계획*의 『데이터베이스 관리 스페이스』
- *관리 안내서: 성능*의 『시퀀스 프리페치』

태스크 관련:

- *관리 안내서: 계획*의 『64비트 환경에서 대형 페이지 지원 사용(AIX)』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLESPACE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLESPACE문』

특정 유형의 테이블 스페이스 작성

데이터베이스 관리자가 사용하고 응용프로그램 및 사용자가 사용하기 위한 서로 다른 유형의 테이블 스페이스가 있습니다.

시스템 임시 테이블 스페이스 작성

데이터베이스를 작성하면 디폴트로 시스템 임시 테이블 스페이스가 작성되나, 시스템 정렬 태스크를 위해 별도의 테이블 스페이스를 할당하고자 할 수도 있습니다.

전제조건:

시스템 임시 테이블 스페이스와 연관될 컨테이너가 있어야 합니다.

제한사항:

시스템 임시 테이블이 이러한 테이블 스페이스에 저장될 수 있으므로 데이터베이스는 항상 최소한 하나의 시스템 임시 테이블 스페이스를 가져야 합니다.

프로시저:

시스템 임시 테이블 스페이스는 시스템 임시 테이블을 저장하는 데 사용됩니다. 데이터베이스가 작성되면, 정의된 세 가지 디폴트 테이블 스페이스 중 하나는『TEMPSPACE1』이라고 하는 시스템 임시 테이블 스페이스입니다.

CREATE TABLESPACE문을 사용하여 또다른 시스템 임시 테이블 스페이스를 작성할 수 있습니다. 예를 들어,

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
   MANAGED BY SYSTEM
   USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

각 페이지 *크기*의 테이블 스페이스가 하나 이상 있어야 합니다.

시스템 임시 테이블 스페이스를 작성할 때 지정할 수 있는 유일한 데이터베이스 파티션 그룹은 IBMTEMPGROUP입니다.

태스크 관련:

- 94 페이지의『사용자 임시 테이블 스페이스 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TABLESPACE문』

사용자 임시 테이블 스페이스 작성

사용자 임시 테이블 스페이스는 데이터베이스가 작성될 때 디폴트로 작성되지 않습니다. 응용프로그램이 데이터베이스의 데이터에 대해 수행 중일 수도 있는 작업의 일부로서 사용자가 임시 테이블을 사용해야 하는 경우도 있습니다. 그러한 경우에는 사용자 임시 테이블을 작성해야 합니다.

프로시저:

사용자 임시 테이블 스페이스는 선언된 임시 테이블을 저장하는 데 사용됩니다.

CREATE TABLESPACE문을 사용하여 사용자 임시 테이블 스페이스를 작성할 수 있습니다.

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
   MANAGED BY DATABASE
   USING (FILE 'd:\db2data\user_tbsp' 5000,
          FILE 'e:\db2data\user_tbsp' 5000)
```

일반 테이블 스페이스와 같이, 사용자 임시 테이블 스페이스도 IBMTEMPGROUP을 제외한 모든 데이터베이스 파티션 그룹에 작성할 수 있습니다. 사용자 임시 테이블 스페이스를 작성할 때 사용되는 디폴트 데이터베이스 파티션 그룹은 IBMDEFAULTGROUP입니다.

DECLARE GLOBAL TEMPORARY TABLE문은 사용자 임시 테이블 스페이스 내에서 사용할 선언된 임시 테이블을 정의합니다.

태스크 관련:

- 117 페이지의『사용자 정의 임시 테이블 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TABLESPACE문』
- *SQL 참조서, 볼륨 2*의『DECLARE GLOBAL TEMPORARY TABLE문』

데이터베이스 파티션 그룹에 테이블 스페이스 작성

다중 파티션 데이터베이스 파티션 그룹에 테이블 스페이스를 배치하면, 테이블 스페이스 내의 모든 테이블이 데이터베이스 파티션 그룹의 각 파티션으로 분할되거나 파티션됩니다. 테이블 스페이스는 하나의 데이터베이스 파티션 그룹에 작성됩니다. 테이블 스페이스가 임의 데이터베이스 파티션 그룹에 작성된 후에는 계속해서 거기에 남아 있어야 합니다. 다른 데이터베이스 파티션 그룹으로 변경될 수 없습니다. CREATE TABLESPACE문이 테이블 스페이스와 데이터베이스 파티션 그룹을 연관시키는데 사용됩니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TABLESPACE문』

원시 입출력 지정

데이터를 저장할 컨테이너에 대한 작업을 수행할 때, DB2 Universal Database는 직접 디스크 액세스(원시 입출력)를 지원합니다. 이러한 유형의 지원으로 인해, DB2 Universal Database 시스템에 직접 디스크 액세스(원시) 디바이스를 접속할 수 있습니다 (유일한 예외는 Windows 9x 운영 체제입니다).

전제조건:

테이블 스페이스를 작성할 때 참조할 컨테이너의 디바이스 또는 파일 이름을 알고 있어야 합니다. 테이블 스페이스에 할당될 각 디바이스 또는 파일 이름과 연관된 스페이스의 양을 알고 있어야 합니다.

컨테이너에 데이터를 읽고 쓸 수 있는 올바른 권한이 필요합니다.

프로시저:

다음 목록은 직접 디스크 액세스 유형의 디바이스를 식별하는 실제적 및 논리적 방법을 설명합니다.

- Windows에서 실제 하드 드라이브를 지정하려면, 다음과 같은 구문을 사용하십시오.

  \\.\PhysicalDriveN

여기서, N은 시스템의 실제 드라이브 중 하나를 나타냅니다. 이 경우, N을 0, 1, 2 또는 기타 양의 정수로 바꿀 수 있습니다.

\\.\PhysicalDrive5

- Windows에서 논리 드라이브(즉, 포맷되지 않은 파티션)를 지정하려면 다음 구문을 사용하십시오.

  \\.\N:

  여기서, N:은 시스템의 논리 드라이브 이름을 나타냅니다. 예를 들어, N:은 E: 또는 임의의 기타 드라이브 이름으로 바꿀 수 있습니다. 드라이브 식별 문자의 사용으로 인한 제한을 극복하기 위해, 논리 드라이브와 함께 GUID(Globally Unique Identifier)를 사용할 수 있습니다.
- **주:** 디바이스에 기록하려면, 서비스 팩 3 이상이 설치된 Windows NT 버전 4.0이 있어야 합니다.
- UNIX 기반의 플랫폼에서 논리적 볼륨은 사용자와 응용프로그램에게 단일, 연속 및 확장 가능한 디스크 볼륨으로 나타날 수 있습니다. 이러한 방법으로 나타나더라도 비연속 물리적 파티션 또는 둘 이상의 물리적 볼륨에 있을 수 있습니다. 또한 논리적 볼륨은 단일 볼륨 그룹 내에 포함되어야 합니다. 볼륨 그룹당 256개의 논리적 볼륨 한계가 있습니다. 볼륨 그룹당 32개의 물리적 볼륨 한계가 있습니다. **mklv** 명령을 사용하여 추가 논리적 볼륨을 작성할 수 있습니다. 이 명령을 사용하여 논리적 볼륨의 이름을 지정하고, 논리적 볼륨에 할당할 논리적 파티션 수 및 위치를 비롯해 특성을 정의할 수 있습니다.

  논리적 볼륨을 작성한 후 **chlv** 명령을 사용하여 이름 및 특성을 변경하고 **extendlv** 명령을 사용하여 할당된 논리적 파티션 수를 증가시킬 수 있습니다. 작성시 논리적 볼륨의 디폴트 최대 크기는 더 크게 지정되지 않는 한 512개의 논리적 파티션입니다. **chlv** 명령은 이 제한을 겹쳐쓰는 데 사용됩니다.

  AIX 내에서 논리적 볼륨 스토리지를 설정 및 제어할 수 있는 운영 체제 명령 세트, 라이브러리 서브루틴 및 기타 도구를 LVM(Logical Volume Manager)이라고 합니다. LVM은 실제 물리적 디스크와 스토리지 스페이스의 단순하고 유연한 논리적 뷰 간의 데이터를 맵핑함으로써 디스크 자원을 제어합니다.

  mklv 및 기타 논리적 볼륨 명령에 대한 자세한 정보는 *AIX 5L 버전 5.2 시스템 관리 개념: 운영 체제 및 디바이스*를 참조하십시오.

Windows 2000 이상에서는 새로운 방법으로 DMS 원시 테이블 스페이스 컨테이너를 지정할 수 있습니다. 볼륨(즉, 기본 디스크 파티션 또는 동적 볼륨)은 작성될 때 GUID(Globally Unique Identifier)가 지정됩니다. GUID는 테이블 스페이스 정의에 컨테이너를 지정할 때 디바이스 ID로 사용할 수 있습니다. GUID는 시스템 전반에 걸쳐

고유하며, 이는 다중 파티션된 데이터베이스 구성에서 디스크 파티션 정의가 동일하더라도 GUID는 파티션마다 다름을 의미합니다.

*db2listvolumes.exe*라는 도구는 Windows 시스템에 정의된 모든 디스크 볼륨의 GUID를 보다 쉽게 표시하는데 사용할 수 있습니다(Windows 운영 체제에서만). 이 도구는 이 도구가 실행되는 현재 디렉토리에 두 개의 파일을 작성합니다. `volumes.xml`이라는 파일에는 XML 사용 브라우저에서 쉽게 볼 수 있도록 XML로 인코드된 각 디스크 볼륨에 대한 정보가 들어 있습니다. `tablespace.ddl`이라는 두 번째 파일에는 테이블 스페이스 컨테이너를 지정하는데 필요한 구문이 들어 있습니다. 이 파일을 갱신하여 테이블 스페이스 정의에 필요한 나머지 정보를 채워야 합니다. *db2listvolumes* 도구는 명령행 인수를 필요로 하지 않습니다.

태스크 관련:

- 97 페이지의『Linux에서 원시 입출력 설정』

Linux에서 원시 입출력 설정

데이터를 저장할 컨테이너에 대한 작업을 수행할 때, DB2 Universal Database는 직접 디스크 액세스(원시 입출력)를 지원합니다. 이러한 유형의 지원으로 인해, DB2 Universal Database 시스템에 직접 디스크 액세스(원시) 디바이스를 접속할 수 있습니다. Linux 환경에서 작업하는 동안 특정 정보가 있습니다.

전제조건:

테이블 스페이스를 작성할 때 참조할 컨테이너의 디바이스 또는 파일 이름을 알고 있어야 합니다. 테이블 스페이스에 할당될 각 디바이스 또는 파일 이름과 연관된 스페이스를 알고 있어야 합니다.

Linux에서 원시 입출력을 설정하기 전에 다음 사항을 요청합니다.

- 하나 이상의 여유 IDE 또는 SCSI 디스크 파티션
- `/dev/rawctl` 또는 `/dev/raw`라는 이름의 원시 디바이스 제어기. 없는 경우 다음과 같이 기호 링크를 작성하십시오.

  ```
  # ln -s /dev/your_raw_dev_ctrl /dev/rawctl
  ```

- 원시 유틸리티, 보통 Linux 분산이 제공

주: 현재 원시 입출력을 지원하는 분산 중, 원시 디바이스 노드의 이름 지정은 다음과 같은 차이점이 있습니다.

표 3. 원시 입출력을 지원하는 Linux 분산.

분산	원시 디바이스 노드	원시 디바이스 제어기
RedHat 또는 TurboLinux	`/dev/raw/raw1 - 255`	`/dev/rawctl`
SuSE	`/dev/raw1 - 63`	`/dev/raw`

DB2는 위의 원시 디바이스 제어기와 대부분 원시 디바이스 노드의 기타 이름 중 하나를 지원합니다. Linux/390의 DB2는 원시 디바이스를 지원하지 않습니다.

프로시저:

Linux에는 원시 입출력이 수행되기 전에 블록 디바이스로 바인드되어야 하는 원시 디바이스 노드의 풀이 있습니다. 블록 디바이스 바인딩 정보에 대해 원시 중앙 저장소로 작동하는 원시 디바이스 제어기가 있습니다. 바인딩은 raw라고 불리는 유틸리티를 사용하여 수행하며, 보통 Linux 분배기가 공급합니다.

Linux에서 원시 입출력을 구성하려면, 다음을 수행하십시오.

이 예에서 사용하는 원시 파티션은 /dev/sda5입니다. 이 원시 파티션에는 어떠한 가치 있는 데이터도 없어야 합니다.

단계 1. 이 파티션에서 4 096바이트 페이지의 수가 필요하면 반올림하여 계산하십시오. 예를 들어, 다음과 같습니다.

```
# fdisk /dev/sda
Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

표 4. Linux 원시 입출력 계산.

디바이스 시동	시작	끝	블록	ID	시스템
/dev/sda1	1	523	4200997	83	Linux
/dev/sda2	524	1106	4682947+	5	Extended
/dev/sda5	524	1106	4682947	83	Linux

```
Command (m for help): q
#
```

/dev/sda5의 페이지 수는 다음과 같습니다.

```
num_pages = floor( ((1106-524+1)*16065*512)/4096 )
num_pages = 11170736
```

단계 2. 이 파티션에 미사용 원시 디바이스 노드를 바인드하십시오. 바인드는 머신이 재부팅될 때마다 매번 수행해야 하며, 루트 액세스를 요청합니다. raw -a를 사용하여 어떤 원시 디바이스 노드가 이미 사용 중인지 보십시오.

```
# raw /dev/raw/raw1 /dev/sda5
/dev/raw/raw1: bound to major 8, minor 5
```

단계 3. 원시 디바이스 제어기 및 디스크 파티션에 적절한 읽기 사용 권한을 설정하십시오. 원시 디바이스에 적절한 읽기 및 쓰기 사용 권한을 설정하십시오.

단계 4. DB2에 테이블 스페이스를 작성하고, 디스크 파티션이 아니라 원시 디바이스를 지정하십시오. 예를 들어, 다음과 같습니다.

```
CREATE TABLESPACE dms1
      MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 11170736)
```

원시 디바이스의 테이블 스페이스 또한 DB2가 지원하는 모든 기타 페이지 크기에 대해 지원됩니다.

태스크 관련:

- 95 페이지의『원시 입출력 지정』

스키마 작성

테이블에 데이터를 구성할 때 테이블과 기타 관련 오브젝트를 함께 그룹화하는 것이 유익할 수도 있습니다. 이 작업은 CREATE SCHEMA문을 사용하여 스키마를 정의함으로써 완료됩니다. 스키마에 대한 정보는 연결한 데이터베이스의 시스템 카탈로그 테이블에 저장됩니다. 이 정보는 다른 오브젝트가 작성될 때, 이 스키마 내에 배치될 수 있습니다.

전제조건:

그룹화할 데이터베이스 테이블 및 기타 관련 오브젝트가 있어야 합니다.

제한사항:

이 명령문은 DBADM 권한을 가진 사용자가 발행해야 합니다.

스키마는 사용자가 IMPLICIT_SCHEMA 권한을 가진 경우에 내재적으로 작성될 수도 있습니다. 이 권한으로, 사용자는 아직 존재하지 않는 스키마 이름을 가진 오브젝트를 작성할 때마다 스키마를 내재적으로 작성합니다.

사용자에게 IMPLICIT_SCHEMA 권한이 없을 경우, 사용자의 권한 부여 ID와 동일한 이름을 갖는 스키마만을 작성할 수 있습니다.

스키마는 데이터베이스에서 고유성을 강화하는데 사용되므로, 스키마 내의 오브젝트에 대한 규정되지 않은 액세스는 허용되지 않습니다. 이것은 두 사용자가 두 개의 테이블(또는 기타 오브젝트)을 동일 이름으로 작성할 수 있는 가능성을 고려할 때 명백해집니다. 고유성을 강요하는 스키마가 없는 경우, 제3의 사용자가 테이블을 쿼리하려고 시도하면 모호성이 존재합니다. 일부 자세한 규정 없이 사용할 테이블을 정하는 것은 불가능합니다.

새로운 스키마 이름은 시스템 카탈로그에 아직 존재하지 않으며, "SYS"로 시작하지 않습니다.

프로시저:

사용자가 SYSADM 또는 DBADM 권한을 가지고 있을 경우, 유효한 이름을 사용하여 스키마를 작성할 수 있습니다. 데이터베이스가 작성될 때, IMPLICIT_SCHEMA 권한에는 PUBLIC(즉, 모든 사용자가 사용 가능한) 권한이 부여됩니다.

CREATE SCHEMA문의 일부로서 작성된 모든 오브젝트의 작성자는 스키마 소유자입니다. 이 소유자는 다른 사용자에게 스키마 특권을 권한 부여(GRANT)하거나 권한 취소(REVOKE)할 수 있습니다.

테이블 이름 규정의 일부로서 스키마 이름을 입력하지 않고도 테이블을 액세스하도록 또 다른 사용자를 허용하려면, 해당 사용자에 대해 뷰를 설정할 것을 요청합니다. 뷰의 정의에서는 사용자 스키마를 포함해서 완전한 테이블 이름을 정의하며, 사용자는 단순히 뷰 이름을 사용하여 쿼리하면 됩니다. 뷰는 뷰 정의의 일부인 사용자 스키마에 의해 완전해집니다.

제어 센터를 사용하여 스키마를 작성하십시오.

1. 데이터베이스 내에서 **스키마** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **스키마** 폴더를 마우스 오른쪽 단추로 누른 후, **작성**을 누르십시오.
3. 새로운 스키마에 대한 정보를 완료한 후, **확인**을 누르십시오.

명령행을 사용하여 스키마를 작성하십시오.

```
CREATE SCHEMA <name> AUTHORIZATION <name>
```

다음은 권한 부여 ID "joe"를 가진 개별 사용자용 스키마를 작성하는 CREATE SCHEMA문의 예입니다.

```
CREATE SCHEMA joeschma AUTHORIZATION joe
```

관련 개념:

- 8 페이지의 『스키마별 오브젝트 그룹화』
- 270 페이지의 『내재된 스키마 권한(IMPLICIT_SCHEMA) 고려사항』
- 271 페이지의 『스키마 특권』

태스크 관련:

- 101 페이지의 『스키마 설정』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE SCHEMA문』

스키마 작성에 대한 세부사항

스키마는 데이터베이스 내의 오브젝트 소유권을 구성하는 데 사용됩니다.

스키마 설정

몇 개의 스키마가 있으면, 특정 DB2 연결 내에서 발행되는 동적 SQL문에서 규정되지 않은 오브젝트 참조에 사용될 디폴트 스키마로서 하나를 지정할 수 있습니다.

프로시저:

특수 레지스터 CURRENT SCHEMA를 디폴트로 사용할 스키마로 설정하면 디폴트 스키마가 설정됩니다. 모든 사용자가 이 특수 레지스터를 설정할 수 있으며, 권한 부여는 필요하지 않습니다.

다음은 특수 레지스터 CURRENT SCHEMA 설정 방법에 대한 예입니다.

```
SET CURRENT SCHEMA = 'SCHEMA01'
```

이 명령문은 응용프로그램 내에서 사용될 수 있거나 대화식으로 발행될 수 있습니다. 일단 CURRENT SCHEMA 특수 레지스터 값이 설정되면, 이는 동적 SQL문의 규정되지 않은 오브젝트 참조용 규정자(스키마)로서 사용됩니다(데이터베이스 오브젝트로의 규정되지 않은 참조가 있는 CREATE SCHEMA문의 경우 예외).

CURRENT SCHEMA 특수 레지스터의 초기값은 현재 세션 사용자의 권한 부여 ID와 같습니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『스키마』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『SET SCHEMA문』
- *SQL 참조서, 볼륨 1*의 『예약된 스키마 이름 및 예약어』
- *SQL 참조서, 볼륨 1*의 『CURRENT SCHEMA 특수 레지스터』

제 4 장 테이블 및 기타 관련 테이블 오브젝트 작성

이 장에서는 데이터베이스 설계를 구현할 때 특정 특성을 사용하여 테이블을 작성하는 방법을 설명합니다.

테이블 작성 및 데이터 채우기

테이블은 데이터베이스 내의 데이터 주 저장소입니다. 새 데이터베이스를 작성할 때 테이블이 작성되고 데이터로 채워집니다.

전제조건:

충분한 시간을 들여 데이터를 저장할 테이블을 설계하고 구성해야 합니다.

프로시저:

테이블 안에서 데이터를 어떻게 구성할 것인지를 판별한 후, 다음 단계는 CREATE TABLE문을 사용하여 이 테이블을 작성하는 것입니다. 테이블 설명은 연결한 데이터베이스의 시스템 카탈로그에 저장됩니다.

CREATE TABLE문은 테이블에 이름을 부여합니다. 이 이름은 규정되거나 규정되지 않은 ID이며, 각 컬럼에 대한 정의입니다. 테이블 스페이스에 단 하나의 테이블만 포함되도록, 각 테이블을 개별 테이블 스페이스에 저장할 수 있습니다. 테이블이 자주 제거되고 작성될 경우, 개별 테이블 스페이스에 저장한 다음, 테이블 대신 테이블 스페이스를 제거하는 것이 훨씬 더 효과적입니다. 단일 테이블 스페이스에 여러 개의 테이블을 저장할 수도 있습니다. 파티션된 데이터베이스 환경에서 선택된 테이블 스페이스는 테이블 데이터가 저장되는 데이터베이스 파티션 그룹 및 데이터베이스 파티션도 정의합니다.

처음에는 테이블에 데이터가 들어 있지 않습니다. 데이터 행을 추가하려면, 다음 중 하나를 사용하십시오.

- INSERT 명령문
- LOAD 또는 IMPORT 명령
- 파티션된 데이터베이스 환경에서 작업할 경우, 자동 로드 프로그램 유틸리티

테이블에 데이터를 추가하는 것은 변경사항을 기록하지 않고 수행될 수 있습니다. CREATE TABLE문의 NOT LOGGED INITIALLY절이 테이블에 변경사항을 기록하지 못하게 합니다. 테이블이 작성되는 동일한 작업 단위(UOW)에서 INSERT,

DELETE, UPDATE, CREATE INDEX, DROP INDEX 또는 ALTER TABLE 조작에 의해 테이블에 이루어지는 모든 변경사항은 기록되지 않습니다. 로깅은 후속 작업 단위(UOW)에서 시작됩니다.

테이블은 하나 이상의 컬럼 정의로 구성됩니다. 한 테이블에 대해 최대 500 컬럼을 정의할 수 있습니다. 컬럼은 엔터티의 속성을 나타냅니다. 컬럼에 있는 값은 모두 동일한 유형의 정보입니다.

주: 4KB 페이지 *크기*를 사용할 때에는 최대 500 컬럼입니다. 8KB, 16KB 또는 32KB 페이지 *크기*를 사용할 때에는 최대 1012 컬럼을 사용할 수 있습니다.

컬럼 정의 포함에는 *컬럼 이름*, *데이터 유형*, 필요한 모든 *널(NULL) 속성* 또는 디폴트 값(사용자가 선택한)이 있습니다.

컬럼 이름은 컬럼에 포함된 정보를 나타내면서 쉽게 기억할 수 있는 것이어야 합니다. 이 이름은 테이블 내에서 고유한 것이어야 하지만, 같은 이름을 다른 테이블에서 사용할 수도 있습니다.

컬럼의 데이터 유형은 컬럼에 있는 값의 길이와 컬럼에 유효한 데이터의 종류를 나타냅니다. 데이터베이스 관리 프로그램은 문자열, 숫자, 날짜, 시간 및 대형 오브젝트(LOB) 데이터 유형을 사용합니다. 그래픽 문자열 데이터 유형은 복수 바이트 문자 세트를 사용하는 데이터베이스 환경에서만 사용 가능합니다. 또한 사용자 정의 구별 유형으로 컬럼을 정의할 수도 있습니다.

디폴트 속성에 대한 스펙은 값이 제공되지 않을 경우에 사용해야 할 값을 표시합니다. 디폴트값을 지정하거나 시스템 정의 디폴트값을 사용할 수 있습니다. 널(NULL) 속성 스펙이 있는(또한 없는) 컬럼에 대해 디폴트값을 지정할 수 있습니다.

널(NULL) 속성 스펙은 컬럼이 널(NULL) 값을 포함하고 있는지의 여부를 나타냅니다.

제어 센터를 사용하여 테이블을 작성하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **테이블** 폴더를 마우스 오른쪽 단추로 누른 후, **작성**을 누르십시오.
3. 태스크를 완료하려면, 마법사의 단계에 따르십시오.

명령행을 사용하여 테이블을 작성하려면, 다음을 입력하십시오.

```
CREATE TABLE <NAME>
   (<column_name>  <data_type>  <null_attribute>)
   IN <TABLE_SPACE_NAME)
```

다음은 RESOURCE 테이블 스페이스에 EMPLOYEE 테이블을 작성하는 CREATE TABLE문의 예입니다. 이 테이블은 샘플 데이터베이스에서 정의됩니다.

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL)
IN RESOURCE
```

테이블을 작성할 때에는 구조화 유형의 속성에 기초하여 테이블 컬럼을 보유하도록 선택할 수 있습니다. 이러한 테이블을 『유형이 지정된 테이블』이라고 합니다.

유형이 지정된 테이블은 또다른 유형이 지정된 테이블에서 일부 컬럼을 상속하도록 정의될 수 있습니다. 이러한 테이블을 『하위 테이블』이라고 하며 이 테이블이 상속하는 테이블을 『상위 테이블』이라고 합니다. 유형이 지정된 테이블과 하위 테이블의 조합을 『테이블 계층 구조』라고 합니다. 테이블 계층 구조에서 맨 위에 있는 테이블(상위 테이블이 없는 것)은 계층 구조의 『루트 테이블』이라고 합니다.

전역 임시 테이블을 선언하려면 DECLARE GLOBAL TEMPORARY TABLE문을 사용하십시오.

또한 쿼리 결과에 기초하여 정의된 테이블을 작성할 수도 있습니다. 이러한 유형의 테이블을 *구체화된 쿼리 테이블*이라고 합니다.

관련 개념:

- *데이터 이동 유틸리티 안내 및 참조서*의 『임포트 개요』
- *데이터 이동 유틸리티 안내 및 참조서*의 『로드 개요』
- *데이터 이동 유틸리티 안내 및 참조서*의 『플랫폼간 데이터 이동 - 파일 형식의 고려사항』
- 139 페이지의 『사용자 정의 유형(UDT)』

태스크 관련:

- 146 페이지의 『구체화된 쿼리 테이블 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 2*의 『INSERT문』
- *SQL 참조서, 볼륨 2*의 『DECLARE GLOBAL TEMPORARY TABLE문』
- *Command Reference*의 『IMPORT Command』
- *Command Reference*의 『LOAD Command』

테이블 작성 및 채우기에 대한 세부사항

테이블에는 모든 데이터가 포함됩니다. 테이블을 작성하고 테이블에 데이터를 배치할 때 고려해야 할 많은 사항이 있습니다.

테이블 스페이스 압축 입문

테이블이 디스크에 저장될 때 스페이스를 덜 차지하도록 하는 두 가지 방법이 있습니다.

- 컬럼 값이 널(NULL)인 경우, 정의된 고정 스페이스의 양을 고려하십시오.
- 컬럼 값을 쉽게 알 수 있거나 판별할 수 있는 경우(디폴트값처럼)와 레코드 포맷팅 및 컬럼 추출 시 데이터베이스 관리 프로그램이 컬럼 값을 사용할 수 있는 경우.

DB2® Universal Database(DB2 UDB)는 이러한 유형의 스페이스 절약을 가능하게 하는 선택적 레코드 형식을 가집니다. 스페이스 절약은 컬럼 레벨에서 뿐만 아니라 테이블 레벨에서도 가능합니다.

관련 개념:

- *관리 안내서: 계획*의 『데이터베이스 오브젝트에 대한 스페이스 요구사항』
- 106 페이지의 『새 테이블의 스페이스 압축』
- 195 페이지의 『기존 테이블의 스페이스 압축』

새 테이블의 스페이스 압축

테이블을 작성할 때, 선택적 VALUE COMPRESSION절을 사용하여 테이블이 테이블 레벨 및 컬럼 레벨에서 스페이스 절약 행 형식을 사용하도록 지정할 수 있습니다.

VALUE COMPRESSION을 사용하면, 정의된 가변 길이 데이터 유형(VARCHAR, VARGRAPHICS, LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB 및 DBCLOB)에 지정되었던 0 길이 데이터 및 널(NULL)은 디스크에 저장되지 않습니다. 이러한 데이터 유형과 연관된 오버헤드 값만이 디스크 스페이스를 차지합니다.

VALUE COMPRESSION을 사용할 경우에는 선택적 COMPRESS SYSTEM DEFAULT 옵션을 사용하여 추가로 디스크 스페이스 사용량을 줄일 수 있습니다. 삽입되거나 갱신된 값이 컬럼의 데이터 유형에 대한 시스템 디폴트값과 동일한 경우에는 최소한의 디스크 스페이스가 사용됩니다. 디폴트값은 디스크에 저장되지 않습니다. COMPRESS SYSTEM DEFAULT를 지원하는 데이터 유형에는 모든 숫자 유형 컬럼, 고정 길이 문자 및 고정 길이 그래픽 문자열 데이터 유형이 포함됩니다. 이는 0 및 공백을 압축할 수 있음을 의미합니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

대형 오브젝트(LOB) 컬럼 고려사항

대형 오브젝트(LOB) 컬럼이 들어 있는 테이블을 작성하기 전에, 다음과 같은 사항을 결정해야 합니다.

1. 변경사항을 LOB 컬럼에 기록하시겠습니까?

   이 변경사항을 기록하지 않으려면, 테이블을 작성할 때 NOT LOGGED절을 지정하여 로그 기능을 해제해야 합니다.

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED)
IN RESOURCE
```

   LOB 컬럼이 1GB보다 크면, 로그 작업이 해체되어야 합니다. (일반적으로 10MB보다 큰 LOB 컬럼을 기록하기를 원하지 않을 수 있습니다). 컬럼 정의에 지정된 기타 옵션처럼, 로그 옵션을 변경하는 방법은 테이블을 재작성하는 것입니다.

   변경사항을 기록하지 않도록 선택했더라도, 롤백이 시스템 생성 오류의 결과이건 응용프로그램 요구이건 상관없이, 변경사항이 롤백되도록 하기 위해 LOB 컬럼이 *음영처리*됩니다. 음영처리는 현재의 스토리지 페이지 내용이 겹쳐쓰게 되지 않도록 하는 복구 기법입니다. 즉, 수정되지 않은 기존의 페이지는 『음영』 사본으로 보존됩니다. 트랜잭션 롤백을 지원하는 데 이들 사본이 더 이상 필요하지 않으면 버립니다.

   주: RESTORE 및 ROLLFORWARD 명령을 사용하여 데이터베이스를 복구할 때, 『NOT LOGGED』 마지막 백업 이후에 작성된 LOB 데이터는 *2진 0으로 바뀝니다*.

2. LOB 컬럼에 필요한 공간을 최소화하시겠습니까?

   CREATE TABLE문의 COMPACT절을 사용하여 LOB 컬럼을 가능한 작게 작성할 수 있습니다. 예를 들어, 다음과 같습니다.

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED  COMPACT)
IN RESOURCE
```

   최소 LOB 컬럼을 가진 테이블에 추가할 경우, 특히 LOB 값의 크기가 증가할 경우(이미 만들어진 스토리지 조정으로 인한) *성능 비용*이 있습니다.

LOB가 SMS 테이블 스페이스에 위치하고 희소 파일 할당을 지원하지 않는 플랫폼에서는 COMPACT절을 사용하십시오. 희소 파일 할당은 운영 체제가 실제 디스크 스페이스를 사용하는 방법과 연관이 있습니다. 희소 파일 할당을 지원하는 운영 체제는 희소 파일 할당을 지원하지 않는 운영 체제와 비교하면 LOB 저장에 많은 실제 디스크 스페이스를 사용하지 않습니다. COMPACT 옵션은 희소 파일 할당 지원과는 별도로 훨씬 큰 실제 디스크 스페이스 『절약』을 가능하게 합니다. 어느 정도의 실제 디스크 스페이스 절약을 확보할 수 있기 때문에, COMPACT를 사용하려면 운영 체제가 희소 파일 할당을 지원하지 않는 경우의 COMPACT 사용을 고려해야 합니다.

주: DB2® 시스템 카탈로그는 LOB 컬럼을 사용하며 이전 버전에서보다 더 많은 스페이스를 차지할 수도 있습니다.

3. DB2 시스템 카탈로그에 있는 LOB 컬럼을 포함하여 LOB 컬럼의 성능을 향상시키시겠습니까?

   카탈로그 테이블에는 대형 오브젝트(LOB) 컬럼이 있습니다. LOB 데이터는 다른 데이터와 함께 버퍼 풀에 보존되지 않고, 필요할 때마다 디스크에서 읽습니다. 디스크로부터 읽을 경우, 카탈로그의 LOB 컬럼이 사용되는 DB2의 성능이 느려질 수 있습니다. 보통 파일 시스템은 자체적으로 데이터를 저장(또는 캐싱)할 장소를 가지고 있기 때문에, 파일 컨테이너에 빌드된 SMS 테이블 스페이스 또는 DMS 테이블 스페이스는 앞에서 LOB가 참조될 경우 입출력을 피할 수 있게 합니다.

관련 개념:

- *관리 안내서: 계획*의 『대형 오브젝트 데이터에 대한 스페이스 요구사항』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 1*의 『대형 오브젝트(LOB)』

제한조건 정의

이 절에서는 제한조건의 정의 방법을 설명합니다.

- 109 페이지의 『고유 제한조건 정의』
- 110 페이지의 『참조 제한조건 정의』
- 114 페이지의 『테이블 점검 제한조건 정의』
- 115 페이지의 『정보 제한조건 정의』.

제한조건에 대한 자세한 정보는 *관리 안내서: 계획*에 있는 제한조건 강제 실시의 계획에 대한 절 및 *SQL 참조서*를 참조하십시오.

고유 제한조건 정의

*고유 제한조건*은 지정된 키의 모든 값이 고유한지 확인합니다. 테이블에는 기본 키로 정의된 많아야 한 가지의 고유 제한조건을 가진 많은 고유 제한조건이 있습니다.

제한사항:

서브테이블에서는 고유 제한조건이 정의되지 않을 수도 있습니다.

테이블마다 한 개의 기본 키만 있습니다.

프로시저:

CREATE TABLE 또는 ALTER TABLE문의 UNIQUE절을 가진 고유 제한조건을 정의하십시오. 고유 키는 둘 이상의 컬럼으로 구성될 수 있습니다. 한 테이블에 둘 이상의 고유 제한조건이 있을 수 있습니다.

일단 설정되면, 고유 제한조건은 INSERT 또는 UPDATE문이 테이블의 데이터를 수정할 때 데이터베이스 관리 프로그램에 의해 자동으로 시행됩니다. 고유 제한조건은 고유 인덱스를 통해 시행됩니다.

ALTER TABLE문에 고유 제한조건이 정의되며, 해당 고유 키의 동일한 컬럼 세트에 대해 인덱스가 존재할 경우, 해당 인덱스는 고유 인덱스가 되어 제한조건에 의해 사용됩니다.

임의의 고유 제한조건을 선택하여 해당 제한조건을 *기본 키*로 사용할 수 있습니다. 기본 키는 참조 제한조건(기타 고유 제한조건과 함께)의 상위 키로 사용될 수 있습니다. CREATE TABLE 또는 ALTER TABLE문의 PRIMARY KEY절로 고유 키를 정의하십시오. 고유 키는 둘 이상의 컬럼으로 구성될 수 있습니다.

1차 인덱스는 고유 키의 값을 고유한 상태가 되도록 시행합니다. 테이블이 고유 키와 함께 작성되면, 데이터베이스 관리 프로그램은 해당 키에 대해 1차 인덱스를 작성합니다.

고유 제한조건으로 사용된 인덱스의 일부 성능 추가 정보는 다음 내용을 포함합니다.

- 인덱스가 있는 빈 테이블을 초기에 로드할 때, LOAD를 사용하는 것이 IMPORT를 사용하는 경우보다 나은 성능을 제공합니다. LOAD의 INSERT 또는 LOAD REPLACE 모드 중 어느 것을 사용하는지 여부는 문제가 되지 않습니다.
- 인덱스가 있는 기존 테이블에 상당한 양의 데이터를 추가할 경우(IMPORT INSERT 또는 LOAD INSERT 사용), LOAD는 IMPORT보다 약간 나은 성능을 제공합니다.
- 대량의 데이터를 초기에 로드하기 위해 IMPORT 명령을 사용하는 경우, 데이터를 임포트하거나 로드된 후에 고유 키를 작성하십시오. 이로 인해 테이블이 로드되는 동

안 인덱스를 유지보수하는 데 발생하는 오버헤드를 막을 수 있습니다. 그 결과, 인덱스는 최소량의 스토리지를 사용하게 됩니다.

- REPLACE 모드에서 로드 유틸리티를 사용하려면, 데이터를 로드하기 전에 고유 키를 작성하십시오. 이 경우, 로드 중에 인덱스를 작성하는 것이 로드 후에 CREATE INDEX문을 사용하는 것보다 효율적입니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『키』
- *SQL 참조서, 볼륨 1*의 『제한조건』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

참조 제한조건 정의

참조 무결성은 테이블과 컬럼 정의에 참조 제한조건을 추가함으로써 이루어집니다. 참조 제한조건이 데이터베이스 관리 프로그램에 정의되면, 테이블 및 컬럼 내의 데이터에 대한 변경이 정의된 제한조건을 토대로 점검됩니다. 요청된 조치의 완료는 제한조건 점검 결과에 따라 달라집니다.

프로시저:

참조 제한조건은 FOREIGN KEY절과, CREATE TABLE 또는 ALTER TABLE문의 REFERENCES절로 설정됩니다. 참조 제한조건을 작성하기 전에 고려해야할, 유형이 지정된 테이블이나 상위 테이블(유형이 지정된 테이블)에 대한 참조 제한조건의 효과가 있습니다.

외부 키 식별은 한 테이블 행 내의 값 또는 두 테이블 행 사이의 값에 대한 제한조건을 시행합니다. 데이터베이스 관리 프로그램은 테이블 정의에 지정된 제한조건을 검사하여 이에 따라 관계를 유지보수합니다. 목적은 하나의 데이터베이스 오브젝트가 또다른 데이터베이스 오브젝트를 참조할 때마다 무결성을 유지보수하는 것입니다.

예를 들어, 기본 키와 외부 키는 각각 부서 번호 컬럼을 가지고 있습니다. EMPLOYEE 테이블의 경우에 컬럼 이름은 WORKDEPT이고, DEPARTMENT 테이블의 경우에는 DEPTNO입니다. 두 테이블 간의 관계는 다음 제한조건으로 정의됩니다.

- EMPLOYEE 테이블에 있는 각 사원에 대해 하나의 부서 번호만 있고, 해당 번호가 DEPARTMENT 테이블에 존재합니다.
- EMPLOYEE 테이블의 각 행은 DEPARTMENT 테이블에 있는 하나의 행에만 관련됩니다. 테이블 간에는 고유한 관계가 있습니다.
- WORKDEPT에 대해 널(NULL)이 아닌 값을 가지는 EMPLOYEE 테이블 내의 각 행은 DEPARTMENT 테이블의 DEPTNO 컬럼에 있는 행과 관련이 있습니다.

- DEPARTMENT 테이블이 상위 테이블이고 EMPLOYEE 테이블이 종속 테이블입니다.

상위 테이블인 DEPARTMENT를 정의하는 SQL문은 다음과 같습니다.

```
CREATE TABLE DEPARTMENT
   (DEPTNO    CHAR(3)     NOT NULL,
    DEPTNAME  VARCHAR(29) NOT NULL,
    MGRNO     CHAR(6),
    ADMRDEPT  CHAR(3)     NOT NULL,
    LOCATION  CHAR(16),
       PRIMARY KEY (DEPTNO))
IN RESOURCE
```

종속 테이블인 EMPLOYEE를 정의하는 SQL문은 다음과 같습니다.

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO     BLOB(10m)   NOT NULL,
       FOREIGN KEY DEPT (WORKDEPT)
       REFERENCES DEPARTMENT ON DELETE NO ACTION)
IN RESOURCE
```

DEPARTMENT 테이블의 기본 키로 DEPTNO 컬럼을 지정하고 EMPLOYEE 테이블의 외부 키로 WORKDEPT를 지정함으로써, WORKDEPT 값에 대해 참조 제한조건을 정의하게 됩니다. 이 제한조건은 두 테이블의 값 사이에 참조 무결성을 시행합니다. 이 경우, EMPLOYEE 테이블에 추가되는 사원은 DEPARTMENT 테이블에서 찾을 수 있는 부서 번호를 가지고 있어야 합니다.

사원 테이블의 참조 제한조건에 대한 삭제 규칙은 NO ACTION이며, 이는 해당 부서에 사원이 있는 경우에 DEPARTMENT 테이블에서 부서를 삭제할 수 없음을 의미합니다.

앞의 예에서는 참조 제한조건을 추가하기 위해 CREATE TABLE문을 사용했지만, ALTER TABLE문을 사용할 수도 있습니다.

기타 예: 앞의 예에 사용된 것과 같은 테이블 정의가 사용됩니다. 또한 DEPARTMENT 테이블이 EMPLOYEE 테이블보다 먼저 작성됩니다. 각 부서에 관리자가 있고, 해당 관리자는 EMPLOYEE 테이블에 나열되어 있습니다. DEPARTMENT 테이블의 MGRNO는 실제로 EMPLOYEE 테이블의 외부 키입니다. 이 참조 순환으로 이 제한조건이 약간의 문제를 가져올 수 있습니다. 나중에 외부 키를 추가할 수 있습니다. 또한 CREATE SCHEMA문을 사용하여 동시에 EMPLOYEE 및 DEPARTMENT 테이블 둘 다를 작성할 수도 있습니다.

관련 개념:

- 112 페이지의 『FOREIGN KEY절』
- 113 페이지의 『REFERENCES절』

태스크 관련:

- 205 페이지의 『외부 키 추가』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE SCHEMA문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

FOREIGN KEY절

외부 키는 같은 테이블 또는 다른 테이블의 기본 키나 고유 키를 참조합니다. 외부 키 지정은 지정된 참조 제한조건에 따라 참조 무결성이 유지보수됨을 나타냅니다. CREATE TABLE 또는 ALTER TABLE문의 FOREIGN KEY절로 외부 키를 정의합니다.

외부 키의 컬럼 번호는 해당하는 상위 테이블의 기본 또는 고유 제한조건(상위 키라고 함)의 컬럼 번호와 일치해야 합니다. 또한, 키 컬럼 정의의 해당 부분은 동일한 데이터 유형과 길이를 가지고 있어야 합니다. 외부 키는 *제한조건 이름*을 지정 받을 수 있습니다. 이름을 지정하지 않으면, 자동으로 지정됩니다. 쉽게 사용하려면, *제한조건 이름*을 지정하고 시스템 생성 이름을 사용하지 않는 것이 바람직합니다.

복합 외부 키의 값은 외부 키의 각 컬럼 값이 상위 키의 해당 컬럼 값과 같을 **경우**, 상위 키의 값과 일치합니다. 널(NULL) 값이 들어 있는 외부 키는 상위 키의 값과 일치할 수 없는데, 이는 상위 키는 널(NULL) 값을 가질 수 없는 것으로 정의되어 있기 때문입니다. 그러나 널(NULL) 외부 키 값은 외부 키의 널(NULL) 이외 파트의 값에 관계없이 항상 유효합니다.

다음과 같은 규칙이 외부 키 정의에 적용됩니다.

- 한 테이블이 여러 개의 외부 키를 가질 수 있습니다.
- 외부 키는 키의 모든 파트가 널(NULL) 입력 가능 상태일 때 널(NULL) 입력 가능합니다.
- 어떤 파트가 널(NULL) 값일 경우, 외부 키 값은 널(NULL)이 됩니다.

태스크 관련:

- 109 페이지의 『고유 제한조건 정의』
- 110 페이지의 『참조 제한조건 정의』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

REFERENCES절

REFERENCES절은 테이블간의 관계에서 상위 테이블을 식별하고, 필요한 제한조건을 정의합니다. 컬럼 정의에 이를 포함시키거나, CREATE TABLE 또는 ALTER TABLE 문에서 FOREIGN KEY절을 동반한 개별 절로서 이를 포함시킬 수 있습니다.

REFERENCES절을 컬럼 제한조건으로서 지정하면, 내재된 컬럼 목록이 컬럼 이름으로 구성됩니다. 다중 컬럼에 개별 REFERENCES절이 들어 갈 수 있고, 단일 컬럼에 둘 이상의 REFERENCES절이 들어 갈 수 있음에 유의하십시오.

REFERENCES절에는 삭제 규칙이 포함되어 있습니다. 예에서는 ON DELETE NO ACTION 규칙이 사용되며, 이는 부서에 사원이 지정되어 있는 경우에는 어느 부서도 삭제될 수 없음을 의미합니다. 기타 삭제 규칙으로는 ON DELETE CASCADE, ON DELETE SET NULL 및 ON DELETE RESTRICT가 있습니다.

관련 개념:

- 112 페이지의『FOREIGN KEY절』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의『CREATE TABLE문』

유틸리티 조작 결과

로드 유틸리티는 자체 참조 및 종속 테이블을 점검 보류 상태로 둠으로써 테이블에 대한 제한조건 점검을 해제합니다. 로드 유틸리티가 완료되면, 꺼진 모든 테이블에 대해 제한조건 점검을 다시 켜야 합니다. 예를 들어, DEPARTMENT 및 EMPLOYEE 테이블만 점검 보류 상태에 있을 경우, 다음 명령문을 실행할 수 있습니다.

```
SET INTEGRITY FOR DEPARTMENT, EMPLOYEE IMMEDIATE CHECKED
```

임포트 유틸리티는 다음과 같은 방법으로 참조 제한조건의 영향을 받습니다.

- REPLACE 및 REPLACE CREATE 함수는 오브젝트 테이블에 자신을 제외한 다른 종속 테이블이 있는 경우에는 허용되지 않습니다.

  이러한 함수를 사용하려면, 먼저 상위 테이블에 있는 모든 외부 키를 제거하십시오. 임포트가 완료되면, ALTER TABLE문으로 외부 키를 재작성하십시오.
- 자체 참조 제한조건을 가진 테이블로의 임포트 성공 여부는 행이 임포트하는 순서에 따라 달라집니다.

관련 개념:

- *데이터 이동 유틸리티 안내 및 참조서*의『임포트 개요』
- *데이터 이동 유틸리티 안내 및 참조서*의『로드 개요』
- *데이터 이동 유틸리티 안내 및 참조서*의『무결성 위반에 대한 점검』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『SET INTEGRITY문』

테이블 점검 제한조건 정의

테이블 점검 제한조건은 테이블 점검 제한조건이 정의된 테이블의 각 행에 대해 시행되는 검색 조건을 지정합니다. 데이터베이스 관리 프로그램에 테이블 점검 제한조건이 정의되면, 테이블 및 컬럼 내의 데이터에 대한 삽입 및 갱신이 정의된 제한조건을 토대로 점검됩니다. 요청된 조치의 완료는 제한조건 점검 결과에 따라 달라집니다.

프로시저:

테이블이 작성되거나 변경될 때 테이블과 점검 제한조건 정의를 연관시킴으로써 테이블에 대해 테이블 점검 제한조건을 작성할 수 있습니다. 이 제한조건은 INSERT 또는 UPDATE문이 테이블에 있는 데이터를 수정할 때 자동으로 활성화됩니다. 테이블 점검 제한조건은 DELETE 또는 SELECT문에 아무런 영향을 주지 않습니다. 점검 제한조건은 유형이 지정된 테이블과 연관될 수 있습니다.

제한조건 이름은 동일한 CREATE TABLE문에서 지정된 다른 제한조건과는 달라야 합니다. 제한조건 이름을 지정하지 않으면, 시스템이 제한조건에 대해 18자의 고유 ID를 생성합니다.

테이블 점검 제한조건은 키 고유성 또는 참조 무결성 제한조건에서 다루지 않는 데이터 무결성 규칙을 시행하는 데 사용됩니다. 일부 경우에는 테이블 점검 제한조건이 도메인 검사를 구현하는 데 사용될 수도 있습니다. CREATE TABLE문에서 발행된 다음 제한조건은 모든 활동의 시작 날짜가 동일한 활동의 종료 날짜보다 늦지 않도록 합니다.

```
CREATE TABLE EMP_ACT
   (EMPNO        CHAR(6)      NOT NULL,
    PROJNO     CHAR(6)      NOT NULL,
    ACTNO      SMALLINT     NOT NULL,
    EMPTIME    DECIMAL(5,2),
    EMSTDATE   DATE,
    EMENDATE   DATE,
    CONSTRAINT ACTDATES CHECK(EMSTDATE <= EMENDATE) )
IN RESOURCE
```

앞의 예에서는 테이블 점검 제한조건을 추가하기 위해 CREATE TABLE문을 사용했지만, ALTER TABLE문도 사용될 수 있습니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『제한조건』

태스크 관련:

- 206 페이지의 『테이블 점검 제한조건 추가』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 2*의 『ALTER SERVER문』

정보 제한조건 정의

*정보 제한조건*은 SQL 컴파일러에서 사용할 수 있으나 데이터베이스 관리 프로그램에서는 강제 실행하지 않는 규칙입니다. SQL 컴파일러에는 최적화 가능하고 필수 데이터로의 액세스 경로를 개선하는 형식으로 SQL문을 변환하는 쿼리 재작성 단계가 포함됩니다. 이 제한조건의 목적은 데이터베이스 관리 프로그램이 데이터를 추가로 검증하게 하기 위한 것이 아니라 쿼리 성능을 개선하기 위한 것입니다.

프로시저:

CREATE TABLE 또는 ALTER TABLE문을 사용하여 정보 제한조건을 정의합니다. 이러한 명령문에 참조 무결성 또는 점검 제한조건을 추가합니다. 그런 다음 제한조건 속성을 이들과 연관시켜 데이터베이스 관리 프로그램이 제한조건을 강제 실행하도록 할 것인지의 여부와 쿼리 최적화에 제한조건이 사용되도록 할 것인지의 여부를 지정합니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『제한조건』
- *관리 안내서: 성능*의 『SQL 컴파일러 프로세스』
- *관리 안내서: 성능*의 『쿼리 재작성 방법 및 예』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

새 테이블에 생성된 컬럼 정의

생성된 컬럼은 저장된 값이 삽입 또는 갱신 조작을 통해 지정되기 보다는 표현식을 사용하여 계산되는 기본 테이블에 정의됩니다.

프로시저:

특정 표현식 또는 술어가 항상 사용될 것으로 알려진 테이블을 작성할 때, 하나 이상의 생성된 컬럼을 해당 테이블에 추가할 수 있습니다. 생성된 컬럼을 사용하여 테이블 데이터를 쿼리할 때 성능을 향상시킬 수 있습니다.

예를 들어, 성능이 중요할 때 표현식의 평가가 비용이 많이 들 수 있는 두 가지 경우가 있습니다.

1. 표현식의 평가가 쿼리 중 여러 번 수행된 경우
2. 계산이 복잡한 경우

쿼리 성능을 개선시키기 위해, 표현식의 결과를 포함하는 추가 컬럼을 정의할 수 있습니다. 그런 다음, 동일한 표현식을 포함하는 쿼리를 발행할 때, 생성되는 컬럼이 직접 사용될 수 있거나, 옵티마이저의 쿼리 재작성 구성요소가 표현식을 생성된 컬럼으로 바꿀 수 있습니다.

또한 생성된 컬럼에서 고유하지 않은 인덱스를 작성하는 것이 가능합니다.

쿼리가 둘 이상의 테이블의 데이터의 조인에 관련된 곳에서 생성된 컬럼을 추가하면, 옵티마이저가 가능한 더 나은 조인 전략을 선택할 수 있습니다.

다음은 CREATE TABLE문에서 생성된 컬럼을 정의하는 예입니다.

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

이 테이블을 작성한 후, 인덱스는 생성된 컬럼을 사용하여 작성될 수 있습니다. 예를 들어,

```
CREATE INDEX i1 ON t1(c4)
```

쿼리는 생성된 컬럼의 이점을 사용할 수 있습니다. 예를 들어,

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

다음과 같이 작성할 수 있습니다.

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

또다른 예는 다음과 같습니다.

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

다음과 같이 작성할 수 있습니다.

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

생성된 컬럼을 사용하여 쿼리 성능을 높일 수 있습니다. 결과적으로, 생성된 컬럼은 테이블이 작성되어 데이터가 채워진 후에 추가될 수 있습니다.

태스크 관련:

- 210 페이지의 『기존 테이블에 생성된 컬럼 정의』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE INDEX문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 2*의 『SELECT문』

사용자 정의 임시 테이블 작성

데이터베이스의 데이터를 조작하기 위해 작성하는 응용프로그램에서 사용자 정의 임시 테이블을 필요로 합니다. 데이터 조작 결과가 테이블에 임시로 저장되어야 합니다.

전제조건:

사용자 임시 테이블 스페이스는 사용자 정의 임시 테이블을 작성하기 전에 존재해야 합니다.

제한사항:

이 테이블의 설명은 지속적이지 않게 하는 시스템 카탈로그에 표시되지 않으며 다른 응용프로그램과 공유될 수 없습니다.

이 테이블을 사용하는 응용프로그램이 데이터베이스에서 종료되거나 연결해제될 때 테이블에 있는 임의의 데이터가 삭제되며 테이블이 내재적으로 삭제됩니다.

사용자 정의 임시 테이블은 다음을 지원하지 않습니다.

- LOB 유형 컬럼(또는 LOB에 기초한 구별 유형 컬럼)
- 사용자 정의 유형 컬럼
- LONG VARCHAR 컬럼
- DATALINK 컬럼

프로시저:

DECLARE GLOBAL TEMPORARY TABLE문을 사용하여 임시 테이블을 정의할 수 있습니다. 명령문은 응용프로그램 내에서 사용됩니다. 사용자 정의 임시 테이블만이 응용프로그램이 데이터베이스에서 연결해제될 때까지 지속됩니다.

임시 테이블을 정의할 수 있는 방법의 예는 다음과 같습니다.

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

이 명령문은 gbl_temp라고 하는 사용자 임시 테이블을 작성합니다. 사용자 임시 테이블은 empltabl의 컬럼과 동일한 이름 및 설명을 가진 컬럼으로 정의됩니다. 내재된 정의만이 컬럼 이름, 데이터 유형, 널 기능 특성 및 컬럼 디폴트값 속성을 포함합니다. 고유 제한조건, 외부 키 제한조건, 트리거 및 인덱스를 포함하는 다른 모든 컬럼 속성은 정의되지 않습니다. COMMIT 조작이 수행될 때, WITH HOLD 커서가 테이블에서 열려 있지 않은 경우 테이블에 있는 모든 데이터가 삭제됩니다. 사용자 임시 테이블에 작

성한 변경사항은 로그되지 않습니다. 사용자 임시 테이블은 지정된 사용자 임시 테이블 스페이스에 위치합니다. 이 테이블 스페이스가 존재해야 합니다. 그렇지 않으면 이 테이블의 선언은 실패합니다.

이 테이블을 작성할 때 ROLLBACK 또는 ROLLBACK TO SAVEPOINT를 지정할 경우, 테이블에 있는 모든 행이 삭제되도록(디폴트값인 DELETE ROWS) 또는 테이블에 있는 행이 보존되도록(PRESERVE ROWS) 지정할 수 있습니다.

태스크 관련:

- 94 페이지의『사용자 임시 테이블 스페이스 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ROLLBACK문』
- *SQL 참조서, 볼륨 2*의『SAVEPOINT문』
- *SQL 참조서, 볼륨 2*의『DECLARE GLOBAL TEMPORARY TABLE문』

새 테이블에 식별 컬럼 정의

*식별 컬럼*은 테이블에 추가되는 각 행의 고유 숫자 값을 자동으로 생성하는 방법을 DB2에 제공합니다. 테이블에 추가될 각 행을 고유하게 식별하는 데 필요한 테이블을 사용자가 아는 곳에 작성할 때, 식별 컬럼을 테이블에 추가할 수 있습니다. 테이블에 추가된 각 행의 고유 숫자 값을 생성하려면, 식별 컬럼에 고유 인덱스를 정의하거나 기본 키를 선언해야 합니다.

제한사항:

일단 테이블을 작성하면, 식별 컬럼을 포함하기 위해 테이블 설명을 변경할 수 없습니다.

행을 지정된 명시적 식별 컬럼 값으로 테이블에 삽입하는 경우, 그 다음 내부적으로 생성된 값은 갱신되어 있지 않아 테이블의 기존 값과 충돌할 수 있습니다. 식별 컬럼에 있는 값의 고유성이 기본 키 또는 식별 컬럼에 정의된 고유 인덱스에 의해 실행되는 경우 중복 값은 오류 메시지를 생성합니다.

프로시저:

식별 컬럼의 스펙을 허용하는 CREATE TABLE문에 AS IDENTITY절이 있습니다.

다음은 CREATE TABLE문에서 식별 컬럼을 정의하는 예입니다.

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

이 예에서 세 번째 컬럼은 식별 컬럼입니다. 또한 컬럼을 추가할 때 각 행을 고유하게 식별하기 위해 컬럼에서 사용되는 값을 지정할 수 있습니다. 입력된 첫 번째 행은 컬럼에 위치한 값 『100』을 가지며, 테이블에 추가되는 차후의 모든 행은 5씩 증가되는 연관된 값을 갖습니다.

식별 컬럼의 일부 추가적인 예 사용은 주문 번호, 사원 번호, 재고 번호 또는 변환 기록 번호입니다. DB2가 생성하는 식별 컬럼 값은 ALWAYS 또는 BY DEFAULT입니다.

GENERATED ALWAYS로 정의되는 식별 컬럼에는 DB2에 의해 항상 생성되는 값이 주어집니다. 응용프로그램은 명시적 값을 제공하도록 허용되지 않습니다. GENERATED BY DEFAULT로서 정의된 식별 컬럼은 식별 컬럼의 값을 명시적으로 제공하는 방법을 응용프로그램에 제공합니다. 응용프로그램이 값을 제공하지 않는 경우, DB2는 값을 생성합니다. 응용프로그램이 값을 제어하므로, DB2가 값의 고유성을 보장할 수 없습니다. GENERATED BY DEFAULT절은 기존 테이블의 내용을 복사하거나 테이블을 언로드 및 다시 로드할 의도가 있는 곳에서 데이터 보급을 위해 사용됩니다.

관련 개념:

- 121 페이지의 『IDENTITY 컬럼과 시퀀스 비교』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

시퀀스 작성

*시퀀스*는 값을 자동 생성하도록 하는 데이터베이스 오브젝트입니다. 시퀀스는 고유 키 값을 생성하는 태스크에 이상적으로 적합합니다. 응용프로그램은 시퀀스를 사용하여 데이터베이스 외부에서 고유 카운터를 생성함으로써 있을 수 있는 동시성 및 성능 문제점을 방지합니다.

제한사항:

식별 컬럼 속성과는 달리, 시퀀스는 특정 테이블 컬럼에 고정되어 있지도 고유 테이블 컬럼에 바인드되어 있지도 않으며, 단지 해당 테이블 컬럼을 통해 액세스가 가능합니다.

하나 이상의 시퀀스를 포함하는 데이터베이스를 특정 시점 이전까지 복구하면, 일부 시퀀스에 대해 중복 값 생성을 유발합니다. 가능한 중복 값을 방지하려면, 시퀀스가 있는 데이터베이스는 특정 시점 이전까지 복구해서는 안 됩니다.

NEXTVAL 또는 PREVVAL 표현식의 사용 가능 위치에 대한 몇 가지 제한사항이 있습니다.

프로시저:

시퀀스는 작성되거나 변경되어, 다음 중 한 가지 방법으로 값을 생성합니다.

- 바인드하지 않고 단순 증분 또는 감축
- 사용자 정의 한계에 대한 단순 증분 또는 감축 그리고 중지
- 사용자 정의 한계에 대한 단순 증분 또는 감축 그리고 시작 부분으로 돌아간 후 다시 시작

다음은 시퀀스 오브젝트를 작성하는 예입니다.

```
CREATE SEQUENCE order_seq
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE
   CACHE 24
```

이 예에서 시퀀스는 order_seq로 부릅니다. 이 시퀀스는 1에서 시작하여 최고 한계없이 1씩 증가합니다. 최고 한계를 지정하지 않기 때문에 시작 부분으로 되돌아가 1에서 재시작할 이유가 없습니다. CACHE 매개변수에 연관된 번호는 시퀀스 값의 최대수를 지정하여 데이터베이스 관리 프로그램이 사전 할당하고 메모리에 보존할 수 있도록 합니다.

생성된 시퀀스 번호는 다음과 같은 등록 정보를 가집니다.

- 값은 0 스케일의 임의의 exact 숫자 데이터 유형을 가집니다. 이러한 데이터 유형에는 SMALLINT, BIGINT, INTEGER 및 DECIMAL이 포함됩니다.
- 연속 값은 지정된 정수 증분으로 차이가 납니다. 디폴트 증분값은 1입니다.
- 카운터 값은 복구 가능합니다. 카운터 값은 복구가 필요할 때 로그에서 재구성할 수 있습니다.
- 값을 캐시하여 성능을 향상시킬 수 있습니다. 캐시에 값을 사전 할당하고 저장하면, 로그에 대한 동기 입출력을 줄일 수 있는데, 이때 값은 시퀀스에 대해 생성됩니다. 시스템 장애의 이벤트에서 커미트 받지 못한 모든 캐시 값은 절대 사용되지 않으며 잃어버린 것으로 간주됩니다. CACHE에 대해 지정된 값은 잃어버릴 수 있는 시퀀스 값의 최대 번호입니다.

시퀀스가 사용되는 두 가지 표현식이 있습니다.

PREVVAL 표현식은 현재 응용프로그램 프로세스 내에서 이전 명령문에 대해 지정된 시퀀스의 가장 최근에 생성된 값을 리턴합니다.

NEXTVAL 표현식은 지정된 시퀀스의 그 다음 값을 리턴합니다. NEXTVAL 표현식이 시퀀스의 이름을 지정할 때 새 시퀀스 번호를 생성합니다. 그러나 한 쿼리 내에서 동일한 시퀀스 이름을 지정하는 NEXTVAL 표현식 인스턴스가 여러 개일 경우, 시퀀스에 대한 카운터는 결과의 각 행마다 한 번만 증분되고 NEXTVAL의 모든 인스턴스가 결과의 한 행에 대해 동일한 값을 리턴합니다.

첫 번째 행에 대해서는 NEXTVAL 표현식이 있는 시퀀스 번호를, 추가 행에 대해서는 PREVVAL 표현식이 있는 시퀀스 번호를 참조함으로써, 두 개의 분리된 테이블에서 동일한 시퀀스 번호를 고유한 키 값으로 사용할 수 있습니다.

예를 들어, 다음과 같습니다.

```
INSERT INTO order (orderno, custno)
   VALUES (NEXTVAL FOR order_seq, 123456);
INSERT INTO line_item (orderno, partno, quantity)
   VALUES (PREVVAL FOR order_seq, 987654, 1)
```

관련 개념:

- 121 페이지의『IDENTITY 컬럼과 시퀀스 비교』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE SEQUENCE문』

IDENTITY 컬럼과 시퀀스 비교

IDENTITY 컬럼 및 시퀀스 사이에는 유사점이 있는 반면, 차이점이 있습니다. 각각의 특성은 데이터베이스 및 응용프로그램 설계시에 사용됩니다.

식별 컬럼은 다음 특성을 가집니다.

- 식별 컬럼은 테이블 작성시에만 테이블의 파트로 정의할 수 있습니다. 한번만 테이블을 작성하면, 변경하여 식별 컬럼을 추가할 수 없습니다. (그러나 기존 식별 컬럼 특성은 변경할 수 있습니다.)
- 식별 컬럼은 자동으로 단일 테이블의 값을 생성합니다.
- 식별 컬럼을 GENERATED ALWAYS로 정의하면, 사용된 값은 항상 데이터베이스 관리 프로그램이 생성합니다. 테이블의 내용을 수정하는 동안 응용프로그램은 소유한 값을 제공하도록 허용되지 않습니다.

식별 컬럼은 다음 특성을 가집니다.

- 시퀀스 오브젝트는 임의의 한 테이블에 고정되어 있지 않은 데이터베이스 오브젝트입니다.
- 시퀀스 오브젝트는 임의의 SQL문에서 사용되는 시퀀스 값을 생성합니다.
- 시퀀스 오브젝트가 응용프로그램에 사용되므로, 지정된 시퀀스에서 다음 값의 검색을 제어하는 데 두 가지 표현식이 사용됩니다. PREVVAL 표현식은 현재 세션 내에서 이전 명령문에 대한 지정된 시퀀스의 가장 최근 생성 값을 리턴합니다. NEXTVAL 표현식은 지정된 시퀀스의 그 다음 값을 리턴합니다. 이 표현식을 사용하면, 여러 테이블 내에서 여러 SQL문에 걸쳐 동일 값을 사용할 수 있게 됩니다.

이들 특성이 두 항목에 대한 특성의 전부는 아니지만, 이들 특성은 데이터베이스 설계 및 데이터베이스를 사용하는 응용프로그램에 따라 무엇을 사용할 것인가 판별하는 데 도움을 줍니다.

태스크 관련:

- 115 페이지의『새 테이블에 생성된 컬럼 정의』
- 119 페이지의『시퀀스 작성』
- 210 페이지의『기존 테이블에 생성된 컬럼 정의』

범위 클러스터 테이블의 예

다음의 간단한 두 예는 범위 클러스터 테이블 작성 방법을 보여줍니다. 테이블 키로서 단일 컬럼 또는 다중 컬럼을 사용하는 방법을 보여주는 예입니다. 또한 데이터 오버플로우를 허용하는 테이블 및 데이터 오버플로우를 허용하지 않는 테이블을 작성하는 방법을 보여줍니다.

첫 번째 예에서는 STUDENT_ID를 사용하여 학생을 찾을 수 있는 범위 클러스터 테이블을 보여줍니다. 각 학생 레코드에 있는 정보는 다음과 같습니다.

- 학교 ID
- 프로그램 ID
- 학생 번호
- 학생 ID
- 학생 이름
- 학생 성
- 학생 평점(GPA)

이 경우에 STUDENT_ID를 기초로 학생 레코드를 다루려 합니다. STUDENT_ID를 사용하여 학생 레코드를 추가, 갱신 및 삭제합니다.

주: 나중에 별도의 인덱스를 추가할 수도 있습니다. 단, 이 예의 용도상 테이블 작성시 테이블의 소속 및 테이블 데이터를 액세스하는 방법을 정의합니다.

이 테이블에 필요한 구문은 다음과 같습니다.

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    PROGRAM_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
```

```
   ORGANIZE BY KEY SEQUENCE
   (STUDENT_ID     STARTING FROM 1 ENDING AT 1000000)
    ALLOW OVERFLOW
;
```

각 레코드의 *크기*는 컬럼의 합으로 계산합니다. 이 경우 10바이트 헤더 + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (널(NULL) 입력 가능 컬럼) 합은 97바이트입니다. 4KB 페이지 *크기*(또는 4096바이트)를 사용하면 오버헤드 어카운팅 이후 4038바이트 또는 페이지당 42개의 레코드에 해당하는 공간이 생깁니다. 백만 명의 학생 레코드를 사용하는 경우 백만을 페이지당 42개의 레코드로 나눈 만큼 또는 23809.5페이지가 필요합니다. 반올림하여 23810 페이지가 필요합니다. 테이블 오버헤드에는 4페이지를 추가하고 Extent 맵핑에는 3페이지를 추가합니다. 결과는 미리 할당된 4KB *크기*의 23817페이지가 필요합니다. (Extent 맵핑에서는 단일 컨테이너가 해당 테이블을 보류한다고 가정합니다. 각 컨테이너에는 3페이지가 있어야 합니다.)

두 번째 예에서는 어떤 변화가 가장 먼저 생기는지 교육 위원회의 아이디어를 고려합니다. 교육 위원회의 경우, 200개의 학교가 있고 각 학교에는 35명 정원의 교실이 20개 있습니다. 이 교육 위원회는 최대 140,000명의 학생을 수용할 수 있습니다.

이런 경우 SCHOOL_ID, CLASS_ID 및 STUDENT_NUM 값의 세 가지 인수를 기초로 하여 학생 레코드를 다룰 수 있습니다. 이 세 컬럼 각각은 고유 값을 갖게 되고 다같이 학생 레코드의 추가, 갱신 및 삭제에 사용됩니다.

주: 이전 예에서처럼 다른 인덱스를 나중에 별도로 추가할 수 있습니다.

이 테이블에 필요한 구문은 다음과 같습니다.

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    CLASS_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (SCHOOL_ID      STARTING FROM 1 ENDING AT 200,
    CLASS_ID       STARTING FROM 1 ENDING AT 20,
    STUDENT_NUM    STARTING FROM 1 ENDING AT 35)
    DISALLOW OVERFLOW
;
```

이런 경우 오버플로우를 허용하지 않습니다. 왜냐하면 교육 위원회에서 각 반 정원 수를 제한하는 *규정*이 있기 때문입니다. 이 예에서 가장 큰 학급의 *크기*는 35명입니다. 해당 인수를 학급과 학교 수에서 정한 물리적 한계와 결합시키면 교육 위원회의 학생 수에 오버플로우를 허용할 이유가 없음이 명백해집니다.

학교 마다 총 학급 수가 다를 수 있습니다. 이런 경우 학급 수의 범위 정의 시 (CLASS_ID 사용) 상한선은 모든 학교를 고려할 때 가장 큰 학급 수가 되어야 합니

다. 몇몇 작은 규모의 학교(가장 큰 학교 보다 학급 수가 적은 학교)에는 사용하지 않을 수도 있는(예를 들어, 휴대용 학급을 해당 학교에 추가하지 않는 경우) 학생 레코드에 필요한 스페이스가 있음을 의미합니다.

이전 예와 마찬가지로 동일한 4KB 페이지 *크기* 및 동일한 학생 레코드 *크기*를 사용하면 페이지당 42개의 레코드가 가능합니다. 140,000개의 학생 레코드를 사용하면 3333.3 페이지 또는 반올림하여 3334 페이지가 필요합니다. 테이블 정보에 2 페이지가 필요하고 Extent 맵핑에 3 페이지가 필요합니다. 결과는 미리 할당된 4KB *크기*의 3339 페이지가 필요합니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

SQL 컴파일러가 범위 클러스터 테이블과 작동하는 방법

SQL 컴파일러가 범위 클러스터 테이블(RCT)을 다루는 방법은 보조 B+ 트리 인덱스를 가진 일반 테이블을 다루는 방법과 유사합니다. B+ 트리 인덱스를 사용하여 레코드 위치 또는 레코드 ID(RID)를 판별하기 보다는 RCT는 범위 정의에 해당하는 알고리즘 및 레코드 키 값으로 찾아보기 기능을 사용합니다. 키 값을 사용하여 RID를 빨리 얻을 수 있기 때문에 이런 방법은 인덱스가 있는 경우와 유사합니다.

SQL 컴파일러는 필요한 데이터를 얻는 최적의 액세스 경로를 판별하기 위해 테이블에 대한 통계 정보를 사용합니다. 인덱스 통계는 RUNSTATS 명령이 발행될 때 테이블 스캔 중 수집됩니다. RCT의 경우 테이블을 일반 테이블로서 모델화하고 인덱스는 기능 기반 인덱스로 모델화합니다.

오버플로우를 허용하는 범위 클러스터 테이블 작성시 테이블 레코드의 순서는 지켜지지 않습니다.

관련 개념:

- *관리 안내서: 계획*의 『범위 클러스터 테이블』
- 124 페이지의 『범위 클러스터 테이블 사용 지침』

범위 클러스터 테이블 사용 지침

DB2® Universal Database 및 범위 클러스터 테이블(RCT)을 다룰 때 참고할 지침은 다음과 같습니다.

- 키 값의 범위를 정의할 때 최소값은 선택사항입니다. 지정하지 않으면 디폴트값은 1입니다. 최소값 및 최대값으로 음수값을 허용합니다. 음수값을 사용하는 경우 최소값을 반드시 표시해야 합니다(예: ORGANIZE BY KEY SEQUENCE (F1 STARTING FROM -100 ENDING AT -10).

- 범위 클러스터 테이블 정의에 사용한 키 값과 같은 값에 대해 일반 인덱스를 작성할 수 없습니다.
- 범위 클러스터 테이블에 사용할 수 없는 일부 ALTER TABLE 옵션이 있습니다. 옵션이 물리적 테이블 구조에 영향을 미치지 않는 경우에는 옵션을 사용할 수 있습니다.
- 범위 클러스터 테이블 작성 프로세스는 필요한 디스크 스페이스를 미리 할당하기 때문에 해당 스페이스가 사용 가능하지 않은 경우 테이블 작성은 실패합니다.

관련 개념:

- *관리 안내서: 계획*의 『범위 클러스터 테이블』
- 122 페이지의 『범위 클러스터 테이블의 예』

테이블에 대한 차원 정의

*차원*은 테이블에 대한 클러스터링 키입니다. 테이블에 대해 하나 이상의 차원을 선택할 수 있습니다. 테이블에 둘 이상의 차원이 존재하면, 이 테이블은 다차원적으로 클러스터된 테이블로 간주됩니다. 이러한 테이블은 ORGANIZE BY DIMENSIONS절이 있는 CREATE TABLE문을 사용하여 작성합니다.

제한사항:

ORGANIZE BY [DIMENSIONS]절에서 사용되는 컬럼 세트는 CREATE INDEX문에 대한 규칙을 따릅니다. 이 컬럼들은 스토리지에서 데이터의 실제 순서를 유지보수하는데 사용되는 키로서 취급됩니다.

프로시저:

각 차원은 ORGANIZE BY [DIMENSIONS]절 및 하나 이상의 컬럼을 사용하여 CREATE TABLE문에 지정합니다. 차원 목록 내에서 단일 차원과 연관될 컬럼들을 묶기 위해 괄호가 사용됩니다.

원하는 수의 차원을 동시에 지정할 수 있으므로 데이터는 실제로 하나 이상의 차원에서 클러스터됩니다. 데이터는 차원 라인을 따라 Extent 또는 『블록』 단위로 구성됩니다. 차원 술어를 사용하여 데이터를 조회할 때, 해당 차원 값을 포함하는 테이블의 Extent로만 스캔을 제한할 수 있습니다. 더욱이 Extent는 디스크 상의 순차 페이지 세트이므로 이러한 스캔을 위해 매우 효율적인 프리페치를 수행할 수 있습니다.

단일 클러스터링 인덱스를 가지는 테이블은 시간이 지남에 따라 테이블 스페이스가 채워짐으로써 언클러스터될 수 있으나, 다중 차원을 가지는 테이블은 모든 차원의 클러스터링을 자동으로 계속해서 유지보수할 수 있습니다. 따라서 데이터 순서를 복원하기 위해 테이블을 재구성하지 않아도 됩니다.

지정된 각 차원에 대해 차원 블록 인덱스가 자동으로 작성됩니다. 차원 블록 인덱스는 차원을 따라 데이터에 액세스하는데 사용됩니다. 차원 블록 인덱스는 개별 행이 아닌 Extent를 가리키므로 일반 인덱스보다 훨씬 작습니다. 이러한 차원 블록 인덱스는 특정 차원 값을 포함하는 Extent에 매우 빨리 액세스하는데 사용할 수 있습니다.

복합 블록 인덱스는 자동으로 작성되며 모든 차원 키 컬럼을 포함합니다. 복합 블록 인덱스는 삽입 및 갱신 활동 동안 데이터의 클러스터링을 유지보수하는데 사용됩니다. 복합 블록 인덱스는 쿼리 처리에서 특정 차원 값을 가지는 테이블의 데이터에 액세스하는데 사용됩니다.

주: 복합 블록 인덱스의 키 파트 순서는 쿼리 처리에 복합 블록 인덱스를 사용하거나 적용하는데 영향을 미칠 수도 있습니다. 키 파트의 순서는 MDC 테이블을 작성할 때 사용하는 전체 ORGANIZE BY [DIMENSIONS]절에 있는 컬럼 순서에 의해 결정됩니다. 예를 들어, 다음을 사용하여 테이블을 작성하면,

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

컬럼 (c1,c4,c3,c2)에 복합 블록 인덱스가 작성됩니다. 차원 절에서 c1을 두 번 지정했으나, c1은 복합 블록 인덱스에 대한 키 파트로서는 처음 발견되는 순서로 한 번만 사용됩니다. 복합 블록 인덱스의 키 파트 순서는 삽입 처리 시에는 문제가 되지 않으나 쿼리 처리 시에는 문제가 될 수도 있습니다. 따라서 복합 블록 인덱스의 컬럼 순서를 (c1,c2,c3,c4)로 하는 것이 바람직할 경우에는 다음을 사용하여 테이블을 작성해야 합니다.

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

작성하려는 복합 블록 인덱스가 가지게 될 모든 컬럼이 지정된 차원에 이미 있으면 복합 블록 인덱스가 작성되지 않습니다. 예를 들어, 다음 테이블의 경우에는 복합 블록 인덱스가 작성되지 않습니다.

```
CREATE TABLE t1 (c1 int, c2 int)
   ORGANIZE BY DIMENSIONS (c1,(c2,c1))
```

관련 개념:

- *관리 안내서: 계획*의 『다차원 클러스터링 테이블』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

계층 구조 테이블 또는 유형이 지정된 테이블 작성

계층 구조 테이블은 유형이 지정된 테이블 계층 구조의 구현에 연관된 테이블입니다. 이는 계층 구조의 루트 테이블과 동시에 작성됩니다.

구조화된 유형 계층 구조 설정의 일부로서 유형이 지정된 테이블을 작성하게 됩니다. 유형이 지정된 테이블은 CREATE TYPE문으로 특성을 정의하는 오브젝트의 인스턴스를 저장하는데 사용됩니다.

전제조건:

계층 구조 테이블 또는 유형이 지정된 테이블이 작성되는 유형이 있어야 합니다.

프로시저:

CREATE TABLE문의 변형을 사용하여 계층 구조 테이블 또는 유형이 지정된 테이블을 작성할 수 있습니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블』

태스크 관련:

- 127 페이지의 『유형이 지정된 테이블에 데이터 채우기』
- 145 페이지의 『유형이 지정된 뷰 작성』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『구조화된 유형 계층 구조 작성』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블 삭제』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TYPE(구조화)문』

유형이 지정된 테이블에 데이터 채우기

구조화된 유형 계층 구조 설정의 일부로서 유형이 지정된 테이블을 작성하게 됩니다. 유형이 지정된 테이블은 CREATE TYPE문으로 특성을 정의하는 오브젝트의 인스턴스를 저장하는데 사용됩니다. 유형이 지정된 테이블을 작성한 후에는 여기에 데이터를 채워야 합니다.

전제조건:

유형이 지정된 테이블이 있어야 합니다.

프로시저:

구조화 유형을 작성하고 해당 테이블 및 하위 테이블을 작성한 다음, 유형이 지정된 테이블에 데이터를 채울 수 있습니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형 지정 테이블에서의 대체 가능성』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블』

태스크 관련:

- 126 페이지의 『계층 구조 테이블 또는 유형이 지정된 테이블 작성』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형 지정 테이블 행에서의 오브젝트 저장』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블 삭제』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TYPE(구조화)문』

다중 테이블 스페이스에 테이블 작성

테이블 데이터는 테이블 인덱스와 테이블과 연관된 긴 컬럼 데이터와 동일한 테이블 스페이스에 저장될 수 있습니다. 또한 나머지 테이블 데이터의 테이블 스페이스와는 분리하여, 개별 테이블 스페이스에 인덱스를 위치시키고, 개별 테이블 스페이스에 긴 컬럼 데이터를 위치시킬 수 있습니다.

전제조건:

CREATE TABLE문을 수행하기 전에 모든 테이블 스페이스가 있어야 합니다.

제한사항:

테이블을 일부 분리하는 것은 DMS 테이블 스페이스를 사용해야만 가능합니다.

프로시저:

제어 센터를 사용하여 다중 테이블 스페이스에 테이블을 작성하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **테이블** 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **작성**을 선택하십시오.
3. 테이블 이름을 입력한 후, **다음**을 누르십시오.
4. 사용자 테이블의 컬럼을 선택하십시오.
5. **테이블 스페이스** 페이지에서 **별도의 인덱스 스페이스 사용** 및 **별도의 long 스페이스 사용**을 누른 후, 정보를 지정하고 **완료**를 누르십시오.

명령행을 사용하여 다중 테이블 스페이스에 테이블을 작성하려면, 다음을 입력하십시오.

```
CREATE TABLE <name>
   (<column_name>  <data_type>  <null_attribute>)
    IN <table_space_name>
    INDEX IN <index_space_name>
    LONG  IN <long_space_name>
```

다음 예에서는 테이블의 여러 파트를 서로 다른 테이블 스페이스에 저장하기 위해 EMP_PHOTO 테이블이 어떻게 작성되는지를 보여줍니다.

```
CREATE TABLE EMP_PHOTO
   (EMPNO        CHAR(6)      NOT NULL,
    PHOTO_FORMAT VARCHAR(10)  NOT NULL,
    PICTURE      BLOB(100K) )
IN RESOURCE
INDEX IN RESOURCE_INDEXES
LONG  IN RESOURCE_PHOTO
```

이 예는 EMP_PHOTO 데이터가 다음과 같이 저장되도록 만듭니다.

- EMP_PHOTO 테이블용으로 작성된 인덱스는 RESOURCES_INDEXES 테이블 스페이스에 저장됩니다.
- PICTURE 컬럼에 대한 데이터는 RESOURCE_PHOTO 테이블 스페이스에 저장됩니다.
- EMPNO 및 PHOTO_FORMAT 컬럼에 대한 데이터는 RESOURCE 테이블 스페이스에 저장됩니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

파티션된 데이터베이스에 테이블 작성

파티션된 데이터베이스에서 여러 파티션에 걸쳐 테이블을 작성하면 성능상의 이점이 있습니다. 여러 데이터베이스 파티션이 데이터 검색과 연관된 작업을 분담할 수 있습니다.

전제조건:

실제로 나누어지거나 파티션된 테이블을 작성하기 전에, 다음과 같은 사항을 고려해야 합니다.

- 테이블 스페이스는 둘 이상의 데이터베이스 파티션을 확장시킬 수 있습니다. 테이블이 놓이는 파티션 수는 데이터베이스 파티션 그룹의 파티션 수에 따라 달라집니다.
- 동일한 테이블 스페이스에 배치하거나, 첫 번째 테이블 스페이스와 함께 동일한 데이터베이스 파티션 그룹과 연관된 다른 테이블 스페이스에 배치함으로써 테이블을 한 곳에 배치할 수 있습니다.

제한사항:

일단 파티션 키를 선택하면 나중에 변경이 불가능하므로 주의해서 적절한 파티션 키를 선택해야 합니다. 더욱이, 고유 인덱스(고유 키 또는 기본 키)는 파티션 키의 상위 집합으로 정의되어야 합니다. 다시 말해서, 파티션 키가 정의되면, 고유 키와 기본 키는 파티션 키(더 많은 컬럼을 가지고 있음)처럼 동일한 모든 컬럼을 포함해야 합니다.

테이블의 한 파티션의 최대 *크기*는 64GB 또는 사용 가능한 디스크 스페이스 중 더 적은 쪽입니다(이 경우, 테이블 스페이스 *크기*를 4KB 페이지 *크기*로 가정합니다). 테이블의 *크기*는 데이터베이스 파티션의 64GB(또는 사용 가능한 디스크 스페이스)에 데이터베이스 파티션 수를 곱한 만큼 클 수 있습니다. 테이블 스페이스의 페이지 *크기*가 8KB인 경우, 테이블의 *크기*는 데이터베이스 파티션의 128GB(또는 사용 가능한 디스크 스페이스)에 데이터베이스 파티션 수를 곱한 만큼 클 수 있습니다. 테이블 스페이스에 대한 페이지 *크기*가 16KB이면, 테이블의 *크기*는 256GB(또는 사용 가능한 디스크 스페이스)에 데이터베이스 파티션 수를 곱한 만큼 클 수 있습니다. 테이블 스페이스에 대한 페이지 *크기*가 32KB이면, 테이블의 *크기*는 512GB(또는 사용 가능한 디스크 스페이스)에 데이터베이스 파티션 수를 곱한 만큼 클 수 있습니다.

프로시저:

몇몇 데이터베이스 파티션의 일부가 될 테이블의 작성은 해당 테이블을 작성할 때 구체화됩니다. 파티션된 데이터베이스 환경에서 테이블을 작성할 때에는 *파티션 키*라는 추가 옵션이 있습니다. 파티션 키는 테이블의 정의 파트인 키입니다. 이 키는 데이터의 각 행이 저장된 파티션을 판별합니다.

파티션 키를 확실히 지정하지 않은 경우, 다음 디폴트값이 사용됩니다. *디폴트 파티션 키가 적합한지 확인하십시오.*

- 기본 키가 CREATE TABLE문에서 지정되지 않은 경우, 기본 키의 첫 번째 컬럼은 파티션 키로 사용됩니다.
- 기본 키가 없을 경우, 긴 필드가 아닌 첫 번째 컬럼이 사용됩니다.
- 어떠한 컬럼도 디폴트 파티션 키의 요구사항을 충족시킬 수 없는 경우, 이 옵션 없이 테이블이 작성됩니다(단일 파티션 데이터베이스 파티션 그룹의 경우에만 허용됨).

다음은 예입니다.

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
```

```
MIX_INT INTEGER NOT NULL,
MIX_INTS SMALLINT NOT NULL,
MIX_DEC DECIMAL NOT NULL,
MIX_FLT FLOAT NOT NULL,
MIX_DATE DATE NOT NULL,
MIX_TIME TIME NOT NULL,
MIX_TMSTMP TIMESTAMP NOT NULL)
IN MIXTS12
PARTITIONING KEY (MIX_INT) USING HASHING
```

앞의 예에서 테이블 스페이스는 MIXTS12이며 파티션 키는 MIX_INT입니다. 파티션 키가 확실히 지정되어 있지 않으면, 테이블 스페이스는 MIX_CNTL입니다(기본 키가 지정되어 있지 않고 파티션 키가 정의되어 있지 않을 경우, 파티션 키는 목록에서 첫 번째의 길지 않은 컬럼이 됩니다).

테이블 행 및 해당 행에 대한 모든 정보는 항상 동일한 데이터베이스 파티션에 상주합니다.

관련 개념:

- *관리 안내서: 계획*의 『데이터베이스 파티션 그룹』
- *관리 안내서: 계획*의 『데이터베이스 파티션 그룹 설계』
- *관리 안내서: 계획*의 『테이블 공동 배치』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

트리거 작성

트리거는 지정된 기본 테이블 및 유형이 지정된 테이블에서 INSERT, UPDATE 또는 DELETE절과 함께 실행되는 조치 또는 이러한 절에 의해 트리거되는 일련의 조치를 정의합니다. 트리거는 다음 경우에 사용됩니다.

- 입력 데이터의 유효성 확인을 위해
- 새로 삽입된 행의 값을 생성하기 위해
- 상호 참조 목적으로 다른 테이블을 읽기 위해
- 감사 추적 목적으로 다른 테이블에 작성하기 위해

사용자는 무결성 또는 비즈니스 규칙의 일반 양식을 지원하기 위해 트리거를 사용할 수 있습니다. 예를 들어, 트리거는 주문을 승인하기 전에 고객의 신용 한도를 검사하거나 요약 데이터 테이블을 갱신할 수 있습니다.

트리거 사용의 이점은 다음과 같습니다.

- 보다 빠른 응용프로그램 개발: 트리거는 데이터베이스에 저장되므로 모든 응용프로그램에서 이루어지는 조치를 코드화할 필요가 없습니다.

- 보다 용이한 유지보수: 일단 트리거가 정의되면, 이를 작성한 테이블이 액세스될 때 자동으로 호출됩니다.
- 비즈니스 규칙의 전역 시행: 비즈니스 규정이 변경될 경우, 트리거만 변경하고 각각의 응용프로그램은 변경하지 않아도 됩니다.

제한사항:

별칭이 있는 트리거를 사용할 수 없습니다.

트리거가 사전 트리거이면, 트리거 조치에 의해 지정되는 컬럼 이름은 식별 컬럼 이외에 생성된 컬럼일 수 없습니다. 즉, 생성된 식별 값은 사전 트리거에 가시적입니다.

atomic 트리거를 작성할 때는 명령문의 끝 문자에 주의해야 합니다. 데이터베이스 관리 프로그램은 디폴트로 『;』 명령문의 끝 표시문자를 고려합니다. 『;』 이외의 문자를 사용할 수 있도록 atomic 트리거를 작성하려면, 스크립트에서 명령문의 끝 문자를 수동으로 편집해야 합니다. 예를 들어, 『;』은 『#』과 같은 또다른 특수 문자로 바꿀 수 있습니다.

그런 다음, 다음 중 하나를 실행해야 합니다.

- 명령 편집기(명령 센터 대신)에서 선택한 스크립트 탭과 함께 도구 —> 도구 설정 메뉴에서 분리문자를 변경한 다음 스크립트를 실행하십시오. 또는
- 명령행 처리기에서 다음을 사용하십시오.

  ```
  db2 -td <delimiter> -vf <script>
  ```

  여기서, 분리문자는 대체 명령문의 끝 문자이며, <script>는 내부에 새 분리문자가 있는 수정된 스크립트입니다.

프로시저:

제어 센터를 사용하여 트리거를 작성하려면, 다음을 수행하십시오.

1. **트리거** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **트리거** 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **작성**을 선택하십시오.
3. 트리거용 정보를 지정하십시오.
4. 트리거가 호출하려는 조치를 지정한 후, **확인**을 누르십시오.

명령행을 사용하여 트리거를 작성하려면, 다음을 입력하십시오.

```
CREATE TRIGGER <name>
   <action> ON <table_name>
   <operation>
   <triggered_action>
```

다음 SQL문은 신입 사원이 입사할 때마다 사원의 수를 증가시키는 트리거를 작성하는 것으로, EMPLOYEE 테이블에 행이 추가될 때마다 COMPANY_STATS 테이블의 사원 번호(NBEMP) 컬럼에 1을 추가합니다.

```
   CREATE TRIGGER NEW_HIRED
      AFTER INSERT ON EMPLOYEE
FOR EACH ROW
      UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

트리거 내용에는 하나 이상의 SQL문, 즉 INSERT, 검색된 UPDATE, 검색된 DELETE, 전체 선택, SET 전이 변수 및 SIGNAL SQLSTATE가 포함될 수 있습니다. 트리거는 참조하는 INSERT, UPDATE 또는 DELETE문 이전 또는 이후에 활성화될 수 있습니다.

관련 개념:

- 133 페이지의 『트리거 종속성』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『INSERT, UPDATE 및 DELETE 트리거』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『응용프로그램 개발에서의 트리거』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『트리거 작성 지침』
- 134 페이지의 『트리거를 사용하여 뷰 컨텐츠 갱신』

태스크 관련:

- 225 페이지의 『트리거 삭제』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『트리거 작성』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『트리거를 사용하여 비즈니스 규칙 정의』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『트리거를 사용하여 조치 정의』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TRIGGER문』

트리거 종속성

일부 다른 오브젝트에 대한 트리거의 모든 종속성이 SYSCAT.TRIGDEP 카탈로그에 기록됩니다. 트리거는 여러 오브젝트에 따라 달라질 수 있습니다. 이러한 오브젝트 및 해당 종속 트리거는 DROP문에 자세히 기술되어 있습니다.

이들 오브젝트 중 하나가 제거되면, 트리거는 작동 불능이 되지만 정의는 카탈로그에 남아 있습니다. 이 트리거의 유효성을 재확인하려면, 카탈로그에서 정의를 검색하여 새로운 CREATE TRIGGER문을 제출해야 합니다.

트리거가 제거되면, SYSCAT.TRIGGERS 카탈로그 뷰에서 트리거에 관한 설명이 삭제되고 SYSCAT.TRIGDEP 카탈로그 뷰에서 모든 종속성이 삭제됩니다. 트리거에 대해 UPDATE, INSERT 또는 DELETE 종속성을 가지고 있는 모든 패키지도 유효하지 않게 됩니다.

종속 오브젝트가 뷰이고 뷰가 조작 불가능하게 되면, 트리거도 작동 불능으로 표시됩니다. 작동 불능이라고 표시된 트리거에 종속된 임의의 패키지는 유효하지 않습니다.

관련 개념:

- 134 페이지의『트리거를 사용하여 뷰 컨텐츠 갱신』

태스크 관련:

- 131 페이지의『트리거 작성』
- 225 페이지의『트리거 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TRIGGER문』
- *SQL 참조서, 볼륨 2*의『DROP문』

트리거를 사용하여 뷰 컨텐츠 갱신

INSTEAD OF 트리거를 사용하여 본질적으로 갱신이 불가능한 뷰에 대해 삭제, 삽입 또는 갱신 요청을 수행할 수 있습니다. 이러한 유형의 트리거를 이용하는 응용프로그램은 뷰가 테이블인 것처럼 뷰에 갱신 조작을 쓸 수 있습니다.

예를 들어, 다음 SQL문을 사용하여 뷰를 작성할 수 있습니다.

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE,
      DEPTNAME)
   AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE, DEPTNAME
      FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
```

EMPV 뷰의 본문에 있는 조인으로 인해 다음 명령문을 추가할 때까지 뷰를 사용하여 기본 테이블의 데이터를 갱신할 수 없습니다.

```
CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
      REFERENCING NEW AS NEWEMP DEFAULTS NULL FOR EACH ROW
      INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
               PHONENO, HIREDATE)
        VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
               COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                         WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                 RAISE_ERROR('70001', 'Unknown department name')),
               PHONENO, HIREDATE)
```

이 CREATE TRIGGER문은 EMPV 뷰에 대한 INSERT 요청이 수행될 수 있도록 합니다.

```
CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
      REFERENCING OLD AS OLDEMP FOR EACH ROW
DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

이 CREATE TRIGGER문은 EMPV 뷰에 대한 DELETE 요청이 수행될 수 있도록 합니다.

```
CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
      REFERENCING NEW AS NEWEMP
                  OLD AS OLDEMP
      DEFAULTS NULL FOR EACH ROW
BEGIN ATOMIC
 VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
             ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
 UPDATE EMPLOYEE AS E
            SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE) =
                (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
               COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                         WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                 RAISE_ERROR('70001', 'Unknown department name')),
               NEWEMP.PHONENO, NEWEMP.HIREDATE)
 WHERE NEWEMP.EMPNO = E.EMPNO;
END
```

이 CREATE TRIGGER문은 EMPV 뷰에 대한 UPDATE 요청이 수행될 수 있도록 합니다.

태스크 관련:

- 131 페이지의 『트리거 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TRIGGER문』

사용자 정의 함수(UDF) 또는 메소드 작성

사용자 정의 함수(UDF)는 SQL의 내장 함수에서 제공하는 지원에 확장되고 추가되며, 내장 함수가 사용되는 곳이면 어디든지 사용될 수 있습니다. 다음과 같이 UDF를 작성할 수 있습니다.

- 외부 함수: 프로그래밍 언어로 작성됨
- 전래 함수: 일부 기존의 다른 함수로부터 상속되어 구현되는 함수

다음과 같은 세 가지 유형의 UDF가 있습니다.

스칼라 호출때마다 한 가지 값의 응답을 리턴합니다. 예를 들어, SUBSTR() 내장 함수는 스칼라 함수입니다. 스칼라 UDF는 외부에 있거나 전래될 수 있습니다.

컬럼 비슷한 값(컬럼)의 세트에서 한 가지 값의 응답을 리턴합니다. 또한, 종종 DB2에서는 집계 함수라고 합니다. 컬럼 함수의 예로는 내장 함수 AVG()가 있습

니다. 외부 컬럼 UDF는 DB2에 정의될 수 없지만, 내장된 컬럼 함수 중 하나에서 전래된 컬럼 UDF는 정의될 수 있습니다. 이 사항은 구별 유형에 유용합니다.

예를 들어, 기본 유형 INTEGER로 정의된 구별 유형 SHOESIZE가 있는 경우, 내장 함수 AVG(INTEGER)에서 전래된 UDF AVG(SHOESIZE)는 정의될 수 있으며 컬럼 함수가 됩니다.

테이블 테이블을 참조하는 SQL문으로 리턴합니다. 테이블 함수는 SELECT문의 FROM 절에서만 참조됩니다. 이러한 함수는 DB2 데이터가 아닌 데이터에 SQL 언어 처리 기능을 적용하거나 그러한 데이터를 DB2 테이블로 변환하는데 사용할 수 있습니다.

예를 들어, 테이블 함수는 파일을 가져와 테이블로 변환하거나, WWW(World Wide Web)의 샘플 데이터를 테이블로 만들거나, 또는 Lotus® Notes 데이터베이스에 액세스하여 날짜, 보낸 사람 및 메일 메시지 텍스트와 같은 정보를 리턴할 수 있습니다. 이 정보는 데이터베이스의 기타 테이블과 조인될 수 있습니다.

테이블 함수는 외부 함수만 될 수 있습니다. 전래 함수는 될 수 없습니다.

기존의 UDF 정보는 SYSCAT.FUNCTIONS 및 SYSCAT.FUNCPARMS 카탈로그 뷰에 기록됩니다. 시스템 카탈로그에는 UDF에 대한 실행 가능한 코드가 들어 있지 *않습니다* (그러므로 백업 플랜 및 복구 플랜을 작성하려면, 실행 가능한 UDF 관리 방법을 고려해야 합니다).

UDF의 성능에 대한 통계는 SQL문을 컴파일할 때 중요합니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『스칼라 함수』
- *SQL 참조서, 볼륨 1*의 『사용자 정의 함수』
- *SQL 참조서, 볼륨 1*의 『테이블 함수』
- 137 페이지의 『함수 맵핑 작성』
- *관리 안내서: 성능*의 『사용자 정의 함수(UDF)에 대한 통계』
- *관리 안내서: 성능*의 『카탈로그 통계 수동 갱신에 대한 일반 규칙』
- *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 『사용자 정의 함수(UDF) 및 메소드』

태스크 관련:

- 138 페이지의 『함수 템플리트 작성』

관련 참조:

- *SQL 참조서, 볼륨 1*의 『함수』

- *SQL 참조서, 볼륨 2*의 『CREATE FUNCTION문』

사용자 정의 함수(UDF) 또는 메소드 작성에 대한 세부사항

이 섹션에서는 사용자 정의 함수나 메소드를 작성할 경우의 페더레이티드 고려사항에 대한 정보를 제공합니다.

함수 맵핑 작성

로컬 함수 또는 로컬 함수 템플리트를 하나 이상의 데이터 소스에 있는 함수와 맵핑해야 할 때 페더레이티드 데이터베이스에서 함수 맵핑을 작성하십시오. 디폴트 함수 맵핑은 많은 데이터 소스 함수에 제공됩니다.

함수 맵핑은 다음과 같은 경우에 유용합니다.

- 내장된 새 함수가 데이터 소스에서 사용 가능해집니다.
- 데이터 소스에서 사용자 정의 함수(UDF)를 로컬 함수에 맵핑해야 합니다.
- 응용프로그램은 디폴트 맵핑으로 제공되는 다른 디폴트 작동이 필요합니다.

CREATE FUNCTION MAPPING문으로 정의된 함수 맵핑은 페더레이티드 데이터베이스에 저장됩니다.

함수(또는 함수 템플리트)에는 데이터 소스 함수와 동일한 수의 입력 매개변수가 있어야 합니다. 추가로, 페더레이티드 측의 입력 매개변수의 데이터 유형은 데이터 소스 측의 입력 매개변수의 데이터 유형과 호환 가능해야 합니다. 이들 요구사항은 리턴 값에도 적용됩니다.

CREATE FUNCTION MAPPING문을 사용하여 함수 맵핑을 작성하십시오. 예를 들어, Oracle AVGNEW 함수와 서버 ORACLE1에 있는 DB2® 동등 함수 간의 함수 맵핑을 작성하려면 다음을 입력하십시오.

```
CREATE FUNCTION MAPPING ORAVGNEW FOR SYSIBM.AVG(INT) SERVER ORACLE1
OPTIONS (REMOTE_NAME 'AVGNEW')
```

페더레이티드 데이터베이스에서 SYSADM 또는 DBADM 권한 중 하나를 보유하여 이 명령문을 사용하십시오. 함수 맵핑 속성은 SYSCAT.FUNCMAPPINGS에 저장됩니다.

페더레이티드 서버는 입력 호스트 변수를 바인드하거나 LOB, LONG VARCHAR/VARGRAPHIC, DATALINK, 구별 유형 및 구조화 유형의 결과를 검색하지 않습니다. 입력 매개변수 또는 리턴된 값에 이들 유형 중 하나가 포함되면, 함수 맵핑이 작성될 수 없습니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『변환 함수와의 호스트 언어 프로그램 맵핑』

태스크 관련:

- 138 페이지의 『함수 템플리트 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE FUNCTION MAPPING문』

함수 템플리트 작성

페더레이티드 시스템에서 함수 템플리트는 함수 맵핑에 대한 『앵커』를 제공합니다. 이를 사용하여 해당 DB2 함수가 페더레이티드 서버에 없을 때 데이터 소스 함수의 맵핑을 사용할 수 있게 합니다. 함수 맵핑에서는 DB2에 함수 템플리트 또는 기존의 비슷한 함수가 있어야 합니다.

템플리트는 단지 함수 쉘일 뿐입니다. 이는 이름, 입력 매개변수 및 리턴값입니다. 함수에 대한 로컬 실행 파일이 없습니다.

제한사항:

함수에 대한 로컬 실행 파일이 없으므로, 데이터 소스에서 함수를 사용할 수 있는 경우에도 함수 템플리트 호출이 실패할 수 있습니다. 예를 들어, 다음 쿼리를 고려해 보십시오.

```
SELECT myfunc(C1)
FROM nick1
WHERE C2 < 'A'
```

DB2 및 nick1로 참조되는 오브젝트가 들어 있는 데이터 소스에 동일한 조합 시퀀스가 없으면, 함수가 데이터 소스에 있는 동안 DB2에서 비교가 완료되어야 하므로 쿼리는 실패합니다. 조합 시퀀스가 같으면, 비교 조작은 myfunc로 참조되는 기초 함수가 있는 데이터 소스에서 완료될 수 있습니다.

함수(또는 함수 템플리트)에는 데이터 소스 함수와 동일한 수의 입력 매개변수가 있어야 합니다. 페더레이티드 측의 입력 매개변수의 데이터 유형은 데이터 소스 측의 입력 매개변수의 데이터 유형과 호환 가능해야 합니다. 이들 요구사항은 리턴 값에도 적용됩니다.

프로시저:

CREATE FUNCTION문을 AS TEMPLATE 키워드와 함께 사용하여 함수 템플리트를 작성하십시오. 템플리트가 작성된 후, CREATE FUNCTION MAPPING문을 사용하여 템플리트를 데이터 소스에 맵핑합니다.

예를 들어, 서버 S1에서 함수 MTS1FUNC에 대한 함수 템플리트 및 함수 맵핑을 작성하려면, 다음을 입력하십시오.

```
CREATE FUNCTION MYFUNC(INT) RETURNS INT AS TEMPLATE

CREATE FUNCTION MAPPING S1_MYFUNC FOR MYFUNC(INT) SERVER S1 OPTIONS
(REMOTE_NAME 'MYS1FUNC')
```

관련 개념:

• 137 페이지의『함수 맵핑 작성』

관련 참조:

• *SQL 참조서, 볼륨 2*의『CREATE FUNCTION(소스 또는 템플리트)문』

사용자 정의 유형(UDT)

사용자 정의 유형(UDT)은 사용자가 데이터베이스에 작성한 이름 지정된 데이터 유형입니다. UDT는 내장 데이터 유형과 공통된 표현을 공유하는 구별 유형일 수도 있고 각각 유형을 가지는 이름 지정된 속성 시퀀스를 갖는 구조화 유형일 수도 있습니다. 구조화 유형은 유형 계층을 정의하는 또다른 구조화 유형(슈퍼 유형이라 함)의 부속 유형일 수 있습니다.

UDT는 강력한 유형 분류를 지원합니다. 즉, UDT가 다른 유형과 동일한 표현을 공유하더라도 해당 UDT 값은 동일한 UDT 또는 동일한 유형 계층의 UDT 값하고만 호환되는 것으로 간주됩니다.

SYSCAT.DATATYPES 카탈로그 뷰를 사용하면 데이터베이스에 대해 정의된 UDT를 볼 수 있습니다. 이 카탈로그 뷰에서는 또한 데이터베이스가 작성될 때 데이터베이스 관리 프로그램에 의해 정의된 데이터 유형을 보여줍니다.

UDT는 대부분의 시스템 제공 함수 또는 내장 함수의 인수로는 사용될 수 없습니다. 이러한 조작 및 기타 다른 조작을 사용하려면 사용자 정의 함수(UDF)가 제공되어야 합니다.

다음과 같은 경우에만 UDT를 제거할 수 있습니다.

- 기존 테이블에 대한 컬럼 정의에 사용되지 *않습니다*.
- 기존 입력된 테이블 또는 입력된 뷰의 유형으로 사용되지 *않습니다*.
- 제거될 수 없는 UDF 함수에 사용되지 *않습니다*. 뷰, 트리거, 테이블 점검 제한조건 또는 다른 UDF가 이에 종속적인 경우에는, UDF가 제거될 수 없습니다.

UDT가 제거되면, 종속된 다른 함수도 제거됩니다.

관련 개념:

• 141 페이지의『사용자 정의 구조화된 유형 작성』

태스크 관련:

• 140 페이지의『사용자 정의 구별 유형 작성』

관련 참조:

- *SQL 참조서, 볼륨 1*의 『사용자 정의 유형』
- *SQL 참조서, 볼륨 1*의 『데이터 유형』

사용자 정의 유형(UDT) 작성에 대한 세부사항

여기서는 구별 유형과 구조화된 유형 정의를 페더레이티드 유형 맵핑으로 설명합니다.

사용자 정의 구별 유형 작성

사용자 정의 구별 유형은 정수, 10진수 또는 문자 유형과 같은 기존 유형에서 파생된 데이터 유형입니다. CREATE DISTINCT TYPE문을 사용하여 구별 유형을 작성할 수 있습니다.

제한사항:

WITH COMPARISONS절이 CREATE DISTINCT TYPE문에 지정된 경우(예에서처럼), 동일한 구별 유형의 인스턴스가 서로 비교될 수 있습니다. 소스 데이터 유형이 대형 오브젝트(LOB), DATALINK, LONG VARCHAR 또는 LONG VARGRAPHIC 유형인 경우 WITH COMPARISONS절을 지정할 수 없습니다.

구별 유형의 인스턴스는 소스 유형에 정의된 연산의 피연산자나 함수의 인수로 사용될 수 없습니다. 마찬가지로, 소스 유형은 구별 유형을 사용하도록 정의된 인수 또는 피연산자에는 사용할 수 없습니다.

프로시저:

다음 SQL문은 구별 유형 t_educ를 smallint로 작성합니다.

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

일단 구별 유형을 작성하면, 이를 사용하여 CREATE TABLE문에서 컬럼을 정의할 수 있습니다.

```
CREATE TABLE EMPLOYEE
   (EMPNO        CHAR(6)      NOT NULL,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO      BLOB(10M)    NOT NULL,
    EDLEVEL    T_EDUC)
IN RESOURCE
```

구별 유형을 작성하면, 구별 유형과 소스 유형 사이의 캐스트 또한 생성됩니다. T_EDUC 유형 값을 SMALLINT 값으로 캐스트하고 SMALLINT 값을 T_EDUC 값으로 캐스트할 수 있습니다.

변환을 사용하여 UDT를 기본 데이터 유형으로 변환하고 기본 데이터 유형을 UDT로 변환할 수 있습니다. 변환 함수의 작성은 CREATE TRANSFORM문을 통해 이루어집니다.

변환 지원은 CREATE METHOD문 및 CREATE FUNCTION문의 확장을 통해 얻을 수도 있습니다.

관련 개념:

- 139 페이지의 『사용자 정의 유형(UDT)』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE DISTINCT TYPE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TRANSFORM문』
- *SQL 참조서, 볼륨 2*의 『CREATE METHOD문』
- *SQL 참조서, 볼륨 2*의 『CREATE FUNCTION(소스 또는 템플리트)문』
- *SQL 참조서, 볼륨 1*의 『사용자 정의 유형』
- *SQL 참조서, 볼륨 1*의 『데이터 유형』

사용자 정의 구조화된 유형 작성

구조화된 유형은 하나 이상의 속성을 포함하는 사용자 정의 유형입니다. 각 속성에는 이름과 데이터 유형이 있습니다. 속성은 유형의 인스턴스를 설명하는 등록 정보입니다. 구조화 유형은 테이블 유형으로 사용될 수 있는데, 테이블의 각 컬럼은 이름과 데이터 유형을 구조화 유형의 속성 중 하나에서 가져옵니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『사용자 정의 구조화된 유형』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『구조화된 유형 계층 구조』

태스크 관련:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『구조화된 유형 작성』의
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『구조화된 유형 계층 구조 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TYPE(구조화)문』
- *SQL 참조서, 볼륨 1*의 『사용자 정의 유형』

유형 맵핑 작성

페더레이티드 시스템에서 유형 맵핑을 통해 데이터 소스 테이블에 있는 특정 데이터 유형 및 뷰를 DB2 구별 데이터 유형에 맵핑할 수 있습니다. 유형 맵핑은 하나의 데이터 소스 또는 데이터 소스의 범위(유형, 버전)에 적용될 수 있습니다.

디폴트 데이터 유형 맵핑은 내장 데이터 소스 유형과 내장 DB2 유형에 제공됩니다. 새 데이터 유형 맵핑(사용자가 작성하는)은 SYSCAT.TYPEMAPPINGS 뷰에 나열됩니다.

제한사항:

LOB, LONG VARCHAR/VARGRAPHIC, DATALINK, 구조화 유형 또는 구별 유형에 대해 유형 맵핑을 작성할 수 없습니다.

프로시저:

CREATE TYPE MAPPING문으로 유형 맵핑을 작성합니다. 페더레이티드 데이터베이스에서 SYSADM 또는 DBADM 권한 중 하나를 보유하여 이 명령문을 사용하십시오.

유형 맵핑 명령문의 한 예는 다음과 같습니다.

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(10,2)
TO SERVER ORACLE1 TYPE NUMBER([10..38],2)
```

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TYPE MAPPING문』
- *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 『DB2와 OLE DB 간의 데이터 유형 맵핑』

뷰 작성

뷰는 하나 이상의 기본 테이블, 별칭 또는 뷰에서 파생되고 데이터 검색시 기본 테이블과 함께 교환되어 사용될 수 있습니다. 뷰에 표시된 데이터를 변경하면, 테이블의 데이터도 변경됩니다.

뷰는 감지 가능한 데이터에 대한 액세스를 제한하면서, 반면에 기타 데이터에 대한 일반적인 액세스는 좀더 많이 허용하도록 작성될 수 있습니다.

뷰 정의의 SELECT 목록이 직접 또는 간접으로 기본 테이블의 식별 컬럼 이름을 포함하는 뷰에 삽입할 때, INSERT문이 기본 테이블의 식별 컬럼을 직접 참조한 것처럼 동일한 규칙을 적용합니다.

위에서 설명된 대로 뷰를 사용하는 것 외에도, 뷰는 다음 경우에도 사용될 수 있습니다.

- 응용프로그램에 영향을 주지 않고 테이블을 변경합니다. 기본 테이블에 기초하여 뷰를 작성하면 이 작업을 수행할 수 있습니다. 기본 테이블을 사용하는 응용프로그램은 새 뷰의 작성에 영향받지 않습니다. 새 응용프로그램은 기본 테이블을 사용하는 응용프로그램과는 다른 목적으로 작성된 뷰를 사용할 수 있습니다.
- 컬럼의 값을 합산하고, 최대값 또는 평균값을 선택합니다.
- 하나 이상의 데이터 소스에 있는 정보에 액세스할 수 있습니다. CREATE VIEW문 내의 별칭을 참조하고 다중 위치/전역 뷰(뷰는 서로 다른 시스템에 위치한 다중 데이터 소스에 있는 정보를 조인할 수 있음)를 작성할 수 있습니다.

  표준 CREATE VIEW 구문을 사용하는 별칭을 참조하는 뷰를 작성할 때, 뷰 작성자 인증 ID 대신 기초를 이루는 오브젝트 또는 데이터 소스에 있는 오브젝트에 액세스하는 데 뷰 사용자의 인증 ID가 사용된다는 사실을 알려주는 경고가 표시됩니다. FEDERATED 키워드를 사용하여 이 경고가 표시되지 않도록 하십시오.

뷰를 작성하는 또 하나의 이유는 중첩 또는 공통 테이블 표현식을 사용하여 카탈로그 찾아보기를 줄이고 성능을 개선하기 위해서입니다.

전제조건:

뷰를 작성하려면, 기본 테이블, 별칭 또는 뷰의 기초가 되는 뷰가 있어야 합니다.

제한사항:

사용자는 정의에 있는 UDF를 사용하는 뷰를 작성할 수 있습니다. 그러나 이 뷰를 갱신하여 최근 함수를 포함하려면, 뷰를 제거한 후 재작성해야 합니다. 뷰가 UDF에 종속적이면, 해당 함수는 제거될 수 없습니다.

다음 SQL문은 정의에 함수가 있는 뷰를 작성합니다.

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
  AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
     FROM   EMPLOYEE
```

UDF 함수 PENSION은 공식에 따라 HIREDATE, BIRTHDATE, SALARY 및 BONUS를 비롯하여 사원이 받아야 할 현재의 수당을 계산합니다.

프로시저:

제어 센터를 사용하여 뷰를 작성하려면, 다음을 수행하십시오.

1. **뷰** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **뷰** 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **작성**을 선택하십시오.
3. 정보를 완료한 후, **확인**을 누르십시오.

명령행을 사용하여 뷰를 작성하려면, 다음을 입력하십시오.

```
CREATE VIEW <name> (<column>, <column>, <column>)
   SELECT <column_names> FROM <table_name>
   WITH CHECK OPTION
```

예를 들어, EMPLOYEE 테이블에는 아무나 사용할 수 없는 급여 정보가 들어 있습니다. 그러나 사원의 전화번호는 일반적으로 액세스할 수 있어야 합니다. 이 경우, LASTNAME 및 PHONENO 컬럼에서만 뷰가 작성될 수 있습니다. 뷰에 대한 액세스는 PUBLIC으로 권한 부여되는 반면에, 전체 EMPLOYEE 테이블에 대한 액세스는 급여 정보를 볼 수 있는 권한을 가진 사용자로 제한될 수 있습니다.

뷰를 사용하여, 응용프로그램에 대해 사용 가능한 테이블 데이터 서브세트를 만들고, 삽입되거나 갱신될 데이터의 유효성 확인을 수행할 수 있습니다. 뷰에서는 원래 테이블의 해당 컬럼 이름과는 다른 컬럼 이름을 사용할 수 있습니다.

뷰를 사용하면, 사용자 프로그램에 융통성을 더할 수 있고 일반 사용자 쿼리에서 테이블 데이터를 볼 수 있습니다.

다음 SQL문은 EMPLOYEE 테이블에 A00 부서의 모든 직원과 전화번호를 나열하는 뷰를 작성합니다.

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
   AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

이 명령문의 첫 번째 행에서는 뷰를 이름 지정하고 컬럼을 정의합니다. 이름 EMP_VIEW는 SYSCAT.TABLES의 스키마에서 고유해야 합니다. 뷰 이름은 그 안에 데이터가 들어 있지 않더라도 테이블 이름으로 나타납니다. 뷰는 EMPLOYEE 테이블의 컬럼 LASTNAME, EMPNO, PHONENO에 해당하는 DA00NAME, DA00NUM, PHONENO라는 세 개의 컬럼을 가지게 됩니다. 나열된 컬럼 이름은 SELECT문의 선택 목록에 일대일로 적용됩니다. 컬럼 이름이 지정되지 않으면, 뷰에서는 SELECT문의 결과 테이블의 컬럼과 동일한 이름을 사용합니다.

두 번째 행은 데이터베이스에서 어느 값이 선택될 것인지를 설명하는 SELECT문입니다. 여기에 ALL, DISTINCT, FROM, WHERE, GROUP BY 및 HAVING절을 포함할 수도 있습니다. 뷰에 대한 컬럼을 선택할 데이터 오브젝트의 이름(들)이 FROM절 뒤에 와야 합니다.

WITH CHECK OPTION절은 갱신되거나 뷰에 삽입된 행을 뷰 정의에 대해 검사해야 하며, 맞지 않을 경우 거부됨을 나타냅니다. 이로 인해, 데이터 무결성은 강화되지만 추가 처리가 요구됩니다. 이 절을 생략하면, 삽입 및 갱신 내용이 뷰 정의에 맞는지 검사되지 않습니다.

다음 SQL문은 SELECT AS절을 사용하여 EMPLOYEE 테이블에 대해 동일한 뷰를 작성합니다.

```
CREATE VIEW EMP_VIEW
   SELECT LASTNAME AS DA00NAME,
          EMPNO AS DA00NUM,
          PHONENO
     FROM   EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

관련 개념:

- *SQL 참조서, 볼륨 1*의 『뷰』
- 273 페이지의 『테이블 및 뷰 특권』
- 285 페이지의 『뷰로 데이터에 대한 액세스 제어』
- 134 페이지의 『트리거를 사용하여 뷰 컨텐츠 갱신』

태스크 관련:

- 145 페이지의 『유형이 지정된 뷰 작성』
- 201 페이지의 『테이블 또는 뷰에서 행 제거』
- 228 페이지의 『뷰 변경 및 삭제』
- 229 페이지의 『작동 불능 뷰 복구 중』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE VIEW문』
- *SQL 참조서, 볼륨 2*의 『INSERT문』

뷰 작성에 대한 세부사항

유형이 지정된 뷰는 사전 정의된 구조화된 유형을 기반으로 합니다.

유형이 지정된 뷰 작성

프로시저:

CREATE VIEW문을 사용하여 유형이 지정된 뷰를 작성할 수 있습니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 뷰』

태스크 관련:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 뷰』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 뷰 변경』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 뷰 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE VIEW문』

구체화된 쿼리 테이블 작성

*구체화된 쿼리 테이블*은 쿼리 결과를 기초로 정의하는 테이블입니다. 따라서 구체화된 쿼리 테이블에는 일반적으로 해당 테이블 정의의 기초가 되는 테이블에 있는 데이터를 토대로 한 사전 계산된 결과가 들어 있습니다. SQL 컴파일러가 기본 테이블보다 구체화된 쿼리 테이블에 대해 쿼리가 보다 효율적으로 실행될 것으로 판별하면, 구체화된 쿼리 테이블에 대해 쿼리가 실행되며 기본 테이블에 대해 쿼리가 실행될 때보다 빨리 결과를 얻을 수 있습니다.

제한사항:

REFRESH DEFERRED로 정의된 구체화된 쿼리 테이블은 정적 SQL 최적화에 사용되지 않습니다.

CURRENT REFRESH AGE 특수 레지스터 값을 0이 아닌 값에 설정할 때에는 주의해야 합니다. 기초가 되는 기본 테이블의 값을 표시하지 않을 수도 있는 구체화된 쿼리 테이블을 쿼리 최적화에 사용될 수 있도록 하면, 쿼리 결과가 기본 테이블의 데이터를 정확히 표시하지 않을 수도 있습니다. 이는 기본 데이터가 변경되지 않거나 데이터에 대한 지식에 기초하여 결과의 오류 정도를 수용하려는 경우에 합당합니다.

유효한 *fullselect*에 기초하여 새 기본 테이블을 작성하려는 경우, 테이블을 작성할 때 DEFINITION ONLY 키워드를 지정하십시오. 테이블 작성 조작이 완료되면, 새 테이블은 구체화된 쿼리 테이블이 아닌 기본 테이블로 간주됩니다. 예를 들어, 다음과 같이 LOAD 및 SET INTEGRITY에 사용되는 예외 테이블을 작성할 수 있습니다.

```
CREATE TABLE XT AS
(SELECT T.*, CURRENT TIMESTAMP AS TIMESTAMP,CLOB(",32K)
AS MSG FROM T) DEFINITION ONLY
```

다음은 구체화된 쿼리 테이블과 관련된 몇 가지 주요 제한사항입니다.

1. 구체화된 쿼리 테이블을 변경할 수 없습니다.
2. 구체화된 쿼리 테이블이 있는 기본 테이블은 컬럼의 길이를 변경할 수 없습니다.
3. 구체화된 쿼리 테이블로 데이터를 가져올 수 없습니다.
4. 구체화된 쿼리 테이블에 고유 인덱스를 작성할 수 없습니다.
5. 하나 이상의 별칭을 참조하는 쿼리 결과를 기초로 구체화된 쿼리 테이블을 작성할 수 없습니다.

프로시저:

복제 옵션을 사용하여 구체화된 쿼리 테이블을 작성하면 파티션된 데이터베이스 환경의 모든 노드에 걸쳐 테이블을 복제할 수 있습니다. 이러한 테이블을 『복제된 구체화된 쿼리 테이블』이라고 합니다.

구체화된 쿼리 테이블 또는 복제된 구체화된 쿼리 테이블의 분리 레벨이 쿼리의 분리 레벨 이상이면, 일반적으로 구체화된 쿼리 테이블 또는 복제된 구체화된 쿼리 테이블은 쿼리 최적화에 사용됩니다. 예를 들어, 쿼리가 커서 안정성(CS) 분리 레벨 하에서 실행 중일 경우, CS 이상의 분리 레벨 하에서 정의된 구체화된 쿼리 테이블과 복제된 구체화된 쿼리 테이블만이 최적화에 사용됩니다.

구체화된 쿼리 테이블을 작성하려면 CREATE TABLE문을 AS *fullselect*절 및 IMMEDIATE 또는 REFRESH DEFERRED 옵션과 함께 사용합니다.

구체화된 쿼리 테이블의 컬럼 이름을 고유하게 식별하는 옵션을 가집니다. 컬럼 이름의 목록은 전체 선택의 결과 테이블에 컬럼이 있을때 만큼 많은 이름을 포함해야 합니다. 전체 선택의 결과 테이블이 중복 컬럼 이름 또는 이름 없는 컬럼을 갖는다면, 컬럼 이름 목록이 제공되어야 합니다. 이름 없는 컬럼은 상수, 함수, 표현식 또는 선택 목록의 AS절을 사용하는 이름 지정되지 않은 세트 조작에서 파생됩니다. 컬럼 이름의 목록이 지정되지 않은 경우, 테이블의 컬럼은 전체 선택의 결과 세트 컬럼의 이름을 상속합니다.

구체화된 쿼리 테이블을 작성할 때는 구체화된 쿼리 테이블을 시스템이 유지보수하는지 또는 사용자가 유지보수하는지를 지정하는 옵션을 사용합니다. 디폴트는 시스템 유지보수이며, MAINTAINED BY SYSTEM절을 사용하여 명시적으로 지정할 수 있습니다. 사용자 유지보수 구체화된 쿼리 테이블은 MAINTAINED BY USER절을 사용하여 지정합니다.

시스템 유지보수 구체화된 쿼리 테이블을 작성할 경우에는 구체화된 쿼리 테이블이 기본 테이블이 변경될 때 자동으로 새로 고쳐지는지 또는 REFRESH TABLE문에 의해 새로 고쳐지는지를 지정하는 추가 옵션을 가집니다. 구체화된 쿼리 테이블이 기본 테이블이 변경될 때 자동으로 새로 고쳐지도록 하려면 REFRESH IMMEDIATE 키워드를 지정하십시오. 새로 고침은 다음과 같은 경우에 즉시 유용합니다.

- 쿼리가 액세스하는 데이터가 최신 데이터가 되도록 해야 합니다.
- 기본 테이블은 자주 변경되지 않습니다
- 새로 고침은 비용이 적게 듭니다.

이러한 상황에서 구체화된 쿼리 테이블은 사전 계산된 결과를 제공할 수 있습니다. 구체화된 쿼리 테이블의 새로 고침을 지연시키려면 REFRESH DEFERRED 키워드를 지정하십시오. REFRESH DEFERRED를 지정한 구체화된 쿼리 테이블은 기본 테이블의 변경사항을 반영하지 **않습니다**. 이러한 요구사항이 없는 경우에 구체화된 쿼리 테이블을 사용해야 합니다. 예를 들어, DSS 쿼리를 실행하면 legacy 데이터를 포함하는 구체화된 쿼리 테이블을 사용하게 됩니다.

다음과 같은 경우에 쿼리 대신 REFRESH DEFERRED를 사용하여 정의한 구체화된 쿼리 테이블을 사용할 수 있습니다.

- 다음과 같은 경우를 제외하고 지금 새로 고침 요약 테이블의 fullselect 제한사항을 따릅니다.
  - SELECT 목록이 COUNT(*) 또는 COUNT_BIG(*)을 포함할 필요가 없는 경우
  - SELECT 목록이 MAX 및 MIN 컬럼 함수를 포함할 수 있는 경우
  - HAVING절이 허용되는 경우

REFRESH DEFERRED를 사용하여 정의한 구체화된 쿼리 테이블을 새로 고치기 전에 이 테이블을 동적 쿼리에 사용할 수 있는 시간을 지정하려면 CURRENT REFRESH AGE 특수 레지스터를 사용합니다. CURRENT REFRESH AGE 특수 레지스터 값을 설정하려면 SET CURRENT REFRESH AGE문을 사용하면 됩니다.

CURRENT REFRESH AGE 특수 레지스터를 ANY 또는 99999999999999 값으로 설정하여 지연된 구체화된 쿼리를 동적 쿼리에서 사용할 수 있습니다. 9의 콜렉션은 DECIMAL(20,6) 데이터 유형을 가진 시간소인 지속 기간 값인 이 특수 레지스터에 허용되는 최대값입니다. 0 값은 REFRESH IMMEDIATE로 정의한 구체화된 쿼리 테이블만이 쿼리 처리 최적화에 사용될 수 있음을 지시합니다. 0 값으로 설정할 경우, REFRESH DEFERRED로 정의한 구체화된 쿼리 테이블은 최적화에 사용되지 않습니다.

REFRESH IMMEDIATE로 정의된 구체화된 쿼리 테이블은 정적 및 동적 쿼리에 모두 적용 가능하며 CURRENT REFRESH AGE 특수 레지스터를 사용하지 않아도 됩니다.

ENABLE QUERY OPTIMIZATION절을 사용하여 구체화된 쿼리 테이블을 정의하고, 지연된 구체화된 쿼리 테이블의 경우, CURRENT REFRESH AGE 특수 레지스터를 ANY로 설정했으면, 구체화된 쿼리 테이블은 해당 테이블로 경로 지정된 쿼리를 가집니다. 그러나 사용자 유지보수 구체화된 쿼리 테이블의 경우, CURRENT REFRESH AGE 특수 레지스터를 사용하는 것은 쿼리의 경로 재지정을 제어하는 최상의 방법이 아닙니다. CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터가 경로 지정에 사용 가능한 캐시된 데이터 종류를 표시합니다.

활동이 소스 데이터에 영향을 미치므로, 시간이 지난 구체화된 쿼리 테이블에는 더이상 정확한 데이터가 들어 있지 않습니다. REFRESH TABLE문을 사용해야 합니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『분리 레벨』

태스크 관련:

- 218 페이지의 『구체화된 쿼리 테이블 등록 정보 변경』

- 219 페이지의 『구체화된 쿼리 테이블의 데이터 새로 고침』
- 230 페이지의 『구체화된 쿼리 또는 스테이징 테이블 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 2*의 『REFRESH TABLE문』
- *SQL 참조서, 볼륨 2*의 『SET CURRENT REFRESH AGE문』
- *SQL 참조서, 볼륨 1*의 『CURRENT REFRESH AGE 특수 레지스터』
- *SQL 참조서, 볼륨 1*의 『CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 특수 레지스터』
- *SQL 참조서, 볼륨 2*의 『SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION문』

사용자 유지보수 구체화된 쿼리 테이블 작성

사용자 유지보수 구체화된 쿼리 테이블(MQT)은 요약 데이터의 테이블이 이미 존재하는 데이터베이스 시스템의 경우 유용합니다. 해당 요약 테이블을 유지보수하는 사용자 정의 응용프로그램은 일반적입니다. 기존 요약 테이블을 사용자 유지보수 MQT로 식별하면 쿼리 옵티마이저가 기존 요약 테이블을 사용하여 기본 테이블에 대한 쿼리의 결과 세트를 계산할 수 있습니다.

주: 쿼리 옵티마이저는 정적 SQL 쿼리에 대한 액세스 플랜을 선택할 때 사용자 유지보수 MQT를 사용하지 않습니다.

제한사항:

사용자 유지보수 구체화된 쿼리 테이블을 작성할 경우에도 여전히 시스템 유지보수 구체화된 쿼리 테이블과 연관된 제한사항이 적용되나 다음과 같은 예외가 있습니다.

- 구체화된 쿼리 테이블에 대해 INSERT, UPDATE 및 DELETE 조작이 허용됩니다. 그러나 기초가 되는 기본 테이블에 대해 유효성 확인이 수행되지 않습니다. 사용자가 직접 데이터를 정정해야 합니다.
- 이러한 유형의 구체화된 쿼리 테이블에 대해 LOAD, EXPORT, IMPORT 및 데이터 복제 조작을 수행할 수 있으나 유효성 확인이 수행되지 않습니다.
- 이러한 유형의 구체화된 쿼리 테이블에 대해서는 REFRESH TABLE문을 사용할 수 없습니다.
- 이러한 유형의 구체화된 쿼리 테이블에 대해서는 SET INTEGRITY ... IMMEDIATE CHECKED문을 사용할 수 없습니다.
- 사용자 유지보수 구체화된 쿼리 테이블은 REFRESH DEFERRED로 정의해야 합니다.

추가 제한사항은『구체화된 쿼리 테이블 작성』주제를 참조하십시오.

프로시저:

구체화된 쿼리 테이블을 작성하려면 CREATE TABLE문을 AS *fullselect*절 및 IMMEDIATE 또는 REFRESH DEFERRED 옵션과 함께 사용합니다.

구체화된 쿼리 테이블을 작성할 때는 구체화된 쿼리 테이블을 시스템이 유지보수하는지 또는 사용자가 유지보수하는지를 지정하는 옵션을 사용합니다. 디폴트는 시스템 유지보수이며, MAINTAINED BY SYSTEM절을 사용하여 명시적으로 지정할 수 있습니다. 사용자 유지보수 구체화된 쿼리 테이블은 MAINTAINED BY USER절을 사용하여 지정합니다.

대형 데이터베이스 환경이나 데이터 웨어하우스 환경에는 종종 사용자 유지보수 구체화된 쿼리 테이블을 유지보수하고 로그하는 사용자 정의 응용프로그램이 있습니다.

주: 옵티마이저가 사용자 유지보수 MQT를 사용할 수 있게 하려면, 쿼리 최적화 레벨을 레벨 2 또는 5 이상의 레벨로 설정해야 합니다.

태스크 관련:

- 150 페이지의『사용자 유지보수 구체화된 쿼리 테이블 채우기』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TABLE문』

사용자 유지보수 구체화된 쿼리 테이블 채우기

요약 정보를 보유할 테이블을 작성하였으면, 결과 세트를 판별할 때 옵티마이저가 사용하기 원하는 요약 데이터로 구체화된 쿼리 테이블(MQT)을 채웁니다.

전제조건:

요약 정보를 보유할 테이블이 존재하는지 확인하십시오.

프로시저:

트리거, 삽입 조작 또는 LOAD, IMPORT 및 DB2 DataPropagator 유틸리티를 사용하여 사용자 유지보수 MQT를 채울 수 있습니다. 사용자 유지보수 MQT를 처음 채울 때, LOAD 또는 IMPORT 유틸리티를 사용하여 로그 오버헤드를 방지할 수 있습니다.

다음 단계는 사용자 유지보수 MQT를 채우기 위한 일반적인 접근방식을 표시합니다.

- 새 레코드 작성 또는 기존 레코드 수정을 방지하려면 기본 테이블을 읽기 전용으로 지정하십시오.
- 기본 테이블에서 필요한 데이터를 추출하여 외부 파일에 기록하십시오.

- 외부 파일의 데이터를 MQT로 임포트하거나 로드하십시오. CHECK PENDING NO ACCESS 상태에서 테이블에 관하여 LOAD 또는 IMPORT 유틸리티를 사용할 수 있습니다.

  주: SQL 삽입 조작을 사용하여 MQT를 채우고자 하는 경우, PENDING NO ACCESS 상태를 재설정해야 합니다. 그러나 데이터를 설정 중인 상태에서 동적 SQL 조회에서 실수로 해당 MQT를 최적화하지 않게 하려면 ALTER TABLE 문의 SET MATERIALIZED QUERY절에서 DISABLE QUERY OPTIMIZATION 옵션을 사용하여 옵티마이저를 먼저 사용 불가능하게 해야 합니다. MQT를 채웠으면, ALTER TABLE문의 SET MATERIALIZED QUERY 절에서 ENABLE QUERY OPTIMIZATION 옵션을 사용하여 최적화를 사용 가능하게 해야 합니다.
- 새 MQT에 대해 SQL 쿼리를 발행하려면, CHECK PENDING NO ACCESS 상태를 재설정하십시오. 이를 수행하려면, 구체화된 뷰의 데이터 무결성을 책임질 수 있어야 합니다. 이를 수행하기 위한 명령문은 다음과 같습니다.

  ```
  DB2 SET INTEGRITY FOR example ALL IMMEDIATE UNCHECKED
  ```
- 기본 테이블 읽기/쓰기를 수행하십시오.

주: 옵티마이저가 사용자 유지보수 MQT를 사용할 수 있게 하려면, 쿼리 최적화 레벨을 레벨 2 또는 5 이상의 레벨로 설정해야 합니다.

관련 참조:

- *Command Reference*의 『IMPORT Command』
- *Command Reference*의 『LOAD Command』

스테이징 테이블 작성

*스테이징 테이블*은 지연된 구체화된 쿼리 테이블에 대한 점진적 유지보수 지원을 가능하게 합니다. 스테이징 테이블은 구체화된 쿼리 테이블에 적용되어야 할 변경사항을 수집하여 기초가 되는 테이블의 컨텐츠와 동기화합니다. 스테이징 테이블을 사용하면, 구체화된 쿼리 테이블의 즉시 새로 고침이 요청될 때 즉시 유지보수 컨텐츠로 인해 초래되는 높은 잠금 경합이 제거됩니다. 또한 REFRESH TABLE이 수행될 때마다 구체화된 쿼리 테이블을 전체적으로 재생성하지 않아도 됩니다.

구체화된 쿼리 테이블은 복잡한 쿼리, 특히 다음 중 일부 조작이 필요할 수도 있는 쿼리에 대한 응답 시간을 개선하는 강력한 방법입니다.

- 하나 이상의 차원에 대한 집계 데이터
- 하나의 테이블 그룹에 있는 데이터 조인 및 집계
- 일반적으로 액세스되는 데이터 서브세트의 데이터
- 파티션된 데이터베이스 환경의 테이블 또는 테이블 파트에서 다시 파티션된 데이터

제한사항:

다음은 스테이징 테이블과 관련된 몇 가지 주요 제한사항입니다.

1. 스테이징 테이블을 정의하는데 사용되는 쿼리는 점진적으로 유지보수가 가능해야 합니다. 즉, 즉시 새로 고침 옵션이 있는 구체화된 쿼리 테이블과 동일한 규칙을 따라야 합니다.
2. 지연된 새로 고침에만 스테이징 테이블이 지원될 수 있습니다. 쿼리 또한 스테이징 테이블과 연관된 구체화된 쿼리 테이블을 정의합니다. 구체화된 쿼리 테이블은 REFRESH DEFERRED로 정의해야 합니다.
3. 스테이징 테이블을 사용하여 새로 고침을 수행할 때, 현재 시점까지의 새로 고침만이 지원됩니다.

프로시저:

몇 가지 다른 조작이 발생하지 않으면, 불일치하거나, 불완전하거나 또는 보류 상태인 스테이징 테이블을 연관된 구체화된 쿼리 테이블을 점진적으로 새로 고치는데 사용할 수 없습니다. 이러한 조작은 스테이징 테이블의 컨텐츠를 그와 연관된 구체화된 쿼리 테이블 및 기초가 되는 테이블과 일치하게 하여 스테이징 테이블을 보류 상태에서 벗어나게 합니다. 구체화된 쿼리 테이블이 새로 고쳐지면, 해당 스테이징 테이블의 컨텐츠가 지워지고 스테이징 테이블이 정상 상태로 설정됩니다. SET INTEGRITY문을 해당 옵션과 함께 사용하여 스테이징 테이블을 의도적으로 프룬(prune)할 수도 있습니다. 프룬(prune)은 스테이징 테이블을 불일치하는 상태로 변경합니다. 예를 들어, 다음 명령문은 STAGTAB1이라는 스테이징 테이블의 프룬(prune)을 강제 실행합니다.

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

스테이징 테이블은 작성시 보류 상태에 놓이며, 해당 테이블이 기초가 되는 테이블 및 연관된 구체화된 쿼리 테이블의 컨텐츠와 일치하지 않고 불완전함을 표시하는 표시기를 가집니다. 스테이징 테이블이 기초가 되는 테이블로부터 변경사항을 수집하려면 보류 및 불일치 상태에서 벗어나야 합니다. 보류 상태에 있는 동안에는 스테이징 테이블의 기본 테이블을 수정하려는 모든 시도가 실패하며 연관된 구체화된 쿼리 테이블을 새로 고치려는 시도도 실패합니다.

스테이징 테이블을 보류 상태에서 벗어나게 하는데에는 몇 가지 방법이 있습니다.

- SET INTEGRITY FOR <staging table name> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <staging table name> IMMEDIATE CHECKED

태스크 관련:

- 146 페이지의『구체화된 쿼리 테이블 작성』
- 218 페이지의『구체화된 쿼리 테이블 등록 정보 변경』
- 219 페이지의『구체화된 쿼리 테이블의 데이터 새로 고침』
- 230 페이지의『구체화된 쿼리 또는 스테이징 테이블 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『SET INTEGRITY문』

별명 작성

별명은 테이블, 별칭 또는 뷰를 참조하는 간접 방법입니다. 그러므로 SQL문은 해당 테이블 또는 뷰의 완전한 이름과는 무관합니다. 테이블 또는 뷰 이름이 변경될 경우에는 별명 정의만 변경하면 됩니다. 또다른 별명에 대한 별명이 작성될 수 있습니다. 별명은, 기존의 테이블 또는 뷰 이름을 참조할 수 있는 테이블 점검 제한조건 정의를 제외하고는, 뷰 또는 트리거 정의와 임의의 SQL문에서 사용할 수 있습니다.

전제조건:

별명은 정의시에 존재하지 않는 테이블, 뷰 또는 별명에 대해 정의될 수 있습니다. 그러나 별명이 들어 있는 SQL문이 컴파일될 때에는 반드시 있어야 합니다.

제한사항:

별명은 테이블 이름이 사용되는 곳이면 어디든지 사용될 수 있으며, 별명의 체인을 따라 순환 또는 반복적으로 참조되지 않을 경우, 또다른 별명을 참조할 수 있습니다.

별명 이름은 기존의 테이블, 뷰 또는 별명과 동일할 수 없으며, 동일한 데이터베이스 내의 한 테이블만을 언급할 수 있습니다. CREATE TABLE 또는 CREATE VIEW문에 사용된 테이블 또는 뷰의 이름은 동일한 스키마에 있는 다른 별명 이름과 달라야 합니다.

현재 권한 부여 ID가 소유하고 있는 스키마가 아닌 다른 스키마에 별명이 존재하지 않는 한, 별명을 작성하는 데에는 특수 권한이 필요하지 않습니다. 이 예의 경우에는 DBADM 권한이 필요합니다.

별명 또는 별명이 가리키는 오브젝트가 삭제될 경우, 별명과 관련된 모든 패키지에 올바르지 않다는 표시가 주어지고 별명에 의존한 모든 뷰와 트리거에 작동 불능이라는 표시가 주어집니다.

프로시저:

제어 센터를 사용하여 별명을 작성하려면, 다음을 수행하십시오.

1. **별명** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **별명** 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **작성**을 선택하십시오.
3. 정보를 완료한 후, **확인**을 누르십시오.

명령행을 사용하여 별명을 작성하려면, 다음을 입력하십시오.

```
CREATE ALIAS <alias_name> FOR <table_name>
```

별명은 명령문 컴파일시에 테이블 또는 뷰 이름에 의해 바뀝니다. 별명 또는 별명 체인을 테이블 또는 뷰 이름으로 분석할 수 없는 경우, 오류가 발생합니다. 예를 들어, WORKERS가 EMPLOYEE의 별명이면, 컴파일할 때 다음 명령문은

```
SELECT * FROM WORKERS
```

실제로 다음과 같이 됩니다.

```
SELECT * FROM EMPLOYEE
```

다음 SQL문은 EMPLOYEE 테이블에 대한 별명 WORKERS를 작성합니다.

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

주: OS/390 또는 z/Series용 DB2는 두 가지 구별된 별명 개념 ALIAS 및 SYNONYM을 사용합니다. 이 두 개념은 다음과 같이 DB2 Universal Database와 다릅니다.

- OS/390 또는 z/Series용 DB2에서의 ALIAS
  - 작성자에게 특수 권한 또는 특권이 있어야 합니다.
  - 다른 별명을 참조할 수 없습니다.
- OS/390 또는 z/Series용 DB2에서의 SYNONYM
  - 해당 작성자만 사용할 수 있습니다.
  - 항상 규정되지 않습니다.
  - 참조 테이블이 제거될 때 제거됩니다.
  - 이름 스페이스를 테이블 또는 뷰와 공유하지 않습니다.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『별명』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE ALIAS문』

인덱스, 인덱스 확장 또는 인덱스 스펙

인덱스는 행 위치의 목록으로서 하나 이상 지정된 컬럼의 내용에 의해 분류됩니다. 인덱스는 일반적으로 테이블에 대한 액세스 속도를 높이기 위해 사용됩니다. 그러나 논리 데이터 설계 목적으로도 사용될 수 있습니다. 예를 들어, *고유 인덱스*는 컬럼에 값이 중복된 항목이 오도록 허용하지 않기 때문에, 테이블의 어떠한 행도 같지 않다고 보증합니다. 인덱스는 컬럼의 값을 오름차순 또는 내림차순으로 지정하기 위해 작성될 수도 있습니다.

인덱스 확장은 구조화 유형 또는 구별 유형 컬럼을 갖는 인덱스와 함께 사용하는 인덱스 오브젝트입니다.

인덱스 스펙은 메타데이터 구성입니다. 이는 옵티마이저에 데이터 소스 오브젝트(테이블 또는 뷰)에 대해 별칭으로 참조되는 인덱스가 있다는 것을 알려줍니다. 인덱스 스펙에는 행 위치 목록이 없습니다. 이는 인덱스의 설명일 뿐입니다. 옵티마이저는 인덱스 스펙을 사용하여 별칭으로 표시되는 오브젝트에 대한 액세스를 향상시킵니다. 처음 별칭을 작성할 때, 데이터 소스에 있는 기초가 되는 테이블에 대해 DB2®가 인식할 수 있는 형식의 인덱스가 있으면 인덱스 스펙이 작성됩니다.

주: 필요할 경우, 테이블 별칭 또는 하나의 테이블상에 뷰가 있는 뷰 별칭에서 인덱스 스펙을 작성하십시오.

다음과 같은 경우에 인덱스 또는 인덱스 스펙을 수동으로 작성하십시오.

- 성능을 향상시키려는 경우. 예를 들어, 옵티마이저가 특정 테이블 또는 별칭을 중첩된 루프 조인의 내부 테이블로 사용하도록 하려면, 인덱스가 없을 때 조인하는 컬럼에서 인덱스 스펙을 작성하십시오.
- 기본 테이블에 대한 인덱스가 해당 테이블의 별칭이 작성된 후에 추가된 경우

인덱스 스펙은 기본 테이블에 인덱스가 없을 때 작성될 수 있습니다(DB2는 CREATE INDEX문을 발행할 때 리모트 인덱스를 검사하지 않습니다). 인덱스 스펙은 UNIQUE 키워드가 지정되어도 행의 고유성을 강요하지는 않습니다.

DB2 인덱스 권장 도구는 최적의 인덱스 세트 선택을 지원하는 마법사입니다. 제어 센터를 통해 이 마법사에 액세스할 수 있습니다. 비교 가능한 유틸리티를 *db2advis*라고 합니다.

인덱스는 기본 테이블의 컬럼에 의해 정의됩니다. 이는 테이블 작성자 또는 특정 컬럼에 직접 액세스해야 한다는 점을 알고 있는 사용자가 정의할 수 있습니다. 사용자 정의 인덱스가 미리 존재하지 않는 한, 1차 인덱스 키가 자동으로 기본 키에 작성됩니다.

특정 기본 테이블에서 임의의 수의 인덱스를 정의할 수 있으며, 쿼리 성능에도 도움이 될 수 있습니다. 그러나 인덱스가 많을수록 데이터베이스 관리 프로그램은 갱신, 삭제 및 삽입 조작 동안에 더 많이 수정해야 합니다. 많은 내용이 갱신되는 테이블에 대해 대규모의 인덱스를 작성하면 요청 처리 시간이 더 길어질 수 있습니다. 그러므로 잦은 액세스로 인해 분명한 이점이 있는 경우에만 인덱스를 사용하십시오.

인덱스에서 컬럼의 최대 수는 16입니다. 유형이 지정된 테이블을 인덱스화하는 중이라면, 최대 컬럼 수는 15입니다. 인덱스 키의 최대 길이는 1024바이트입니다. 이전에 언급한 것처럼, 테이블에 있는 있는 많은 인덱스 키는 요청 처리 속도를 느리게 할 수 있습니다. 마찬가지로, 역시 큰 인덱스 키는 요청 처리 속도를 느리게 할 수 있습니다.

*인덱스 키*는 인덱스가 정의된 컬럼의 콜렉션 또는 컬럼이며, 인덱스의 사용성을 판별합니다. 인덱스 키를 구성하는 컬럼 순서는 인덱스 키 작성과 다른 점이 없지만, 인덱스의 사용 여부를 결정하는 경우 옵티마이저와는 다를 수 있습니다.

인덱스화된 테이블이 비어 있을 경우에도 인덱스가 계속 작성되지만, 테이블이 로드되거나 행이 삽입될 때까지 어떠한 인덱스 항목도 작성되지 않습니다. 테이블이 비어 있지 않으면, 데이터베이스 관리 프로그램은 CREATE INDEX문을 처리하는 동안에 인덱스 항목을 작성합니다.

*클러스터링 인덱스*의 경우, 비슷한 키 값을 가진 기존 행에 근접하여 새 행이 실제로 삽입됩니다. 이는 데이터 페이지에 대해 좀더 선형적인 액세스 패턴을 만들어 내고 좀더 효과적인 프리페치 결과를 가져오므로 쿼리시 성능상의 이점을 가져옵니다.

기본 키 인덱스가 클러스터링 인덱스가 되게 하려면, 기본 키가 CREATE TABLE에서 지정되지 않아야 합니다. 일단 기본 키가 작성되면, 연관된 인덱스는 수정될 수 없습니다. 대신, 기본 키 절 없이 CREATE TABLE을 수행하십시오. 그런 다음, CREATE INDEX문을 발행하여 클러스터링 속성을 지정하십시오. 마지막으로, ALTER TABLE 문을 사용하여 방금 작성한 인덱스에 해당하는 기본 키를 추가하십시오. 이 인덱스는 기본 키 인덱스로서 사용됩니다.

일반적으로, 클러스터링 인덱스가 고유하므로 클러스터링이 더 효과적으로 유지보수됩니다.

고유 인덱스 키의 파트는 아니지만, 인덱스에 저장/유지보수되는 컬럼 데이터를 *include* 컬럼이라고 합니다. Include 컬럼은 고유 인덱스 전용으로 지정될 수 있습니다. Include 컬럼으로 인덱스 작성시, 고유 키 컬럼만이 고유성을 위해 저장 및 고려됩니다. Include 컬럼의 사용은 인덱스 액세스가 포함될 때 데이터 검색의 성능을 개선시킵니다.

데이터베이스 관리 프로그램은 맨 아래 레벨이 리프 노드로 구성된 B+ 트리 구조를 사용하여 인덱스를 저장합니다. 리프 노드 또는 페이지는 실제 인덱스 키 값이 저장되는 위치입니다. 인덱스를 작성할 때, 온라인으로 그러한 인덱스 리프 페이지가 병합되게 할 수 있습니다. 온라인 인덱스 조각 모음은 여러 번의 삭제 및 갱신 활동 후 다수의 인덱스 리프 페이지에 몇 개의 인덱스 키만 남는 상황을 방지하는 데 사용됩니다. 이러한 상황에서 온라인 인덱스 조각 모음이 수행되지 않으면, 스페이스는 데이터 및/또는 인덱스의 재구성에 의해서만 다시 확보될 수 있습니다. 온라인으로 인덱스 페이지를 조각 모음하는 기능으로 인덱스를 작성할 것인지의 여부를 결정할 때 다음 질문을 고려해야 합니다. 스페이스가 충분할 경우, 리프 페이지에서 실제로 키가 제거될 때마다 병합할 스페이스를 점검하는 추가된 성능 비용과 병합을 완료하는 실제 비용의 효과가 인덱스의 보다 나은 스페이스 활용이라는 이점보다 크고 스페이스를 다시 확보하기 위해 재구성을 수행할 필요성이 감소한 정도 보다 작습니까?

주:

1. 온라인 인덱스 조각 모음 이후에 해제되는 페이지는 같은 테이블에 있는 다른 인덱스에만 재사용될 수 있습니다. 전체 재구성에서 해제되는 해당 페이지는 기타 오브젝트(데이터베이스 관리 스토리지로 작업할 때) 또는 디스크 스페이스(시스템 관리 스토리지로 작업할 때)에서 사용될 수 있습니다. 이외에도, 온라인 인덱스 조각 모

음은 인덱스의 비리프 페이지를 해제하지 않는 반면, 전체 재구성은 인덱스의 비리프 및 리프 페이지뿐만 아니라 인덱스의 레벨 수도 줄임으로써 인덱스를 가능한 한 작게 만듭니다.

2. 버전 8 이전에 작성된 인덱스에서는, 테이블 행 삭제 또는 갱신의 일부로서 리프 페이지에서 키가 실제로 제거됩니다. 유형 2 인덱스의 경우, 행이 삭제되거나 갱신될 때 키는 단지 삭제된 것으로 표시됩니다. 삭제 또는 갱신을 커미트한 후 정리를 수행할 때까지는 페이지에서 키가 실제로 제거되지 않습니다. 그러한 정리는 키가 삭제된 것으로 표시된 페이지를 변경하는 후속 트랜잭션에 의해 수행될 수 있습니다. REORG INDEXES 유틸리티의 CLEANUP ONLY [ALL | PAGES] 옵션을 사용하여 정리를 명시적으로 트리거할 수 있습니다.

파티션된 데이터베이스의 테이블에 대한 인덱스는 동일한 CREATE INDEX문을 사용하여 빌드됩니다. 이 인덱스는 테이블의 파티션 키에 의거하여 파티션됩니다. 테이블의 인덱스는 데이터베이스 파티션 그룹에 있는 각 노드의 해당 테이블의 로컬 인덱스로 구성됩니다. 다중 파티션 환경에서 정의된 고유 인덱스는 파티션 키의 상위 집합이어야 한다는 사실을 주의하십시오.

관련 개념:

- *SQL 참조서, 볼륨 1*의 『인덱스』
- 159 페이지의 『인덱스 사용』
- 160 페이지의 『CREATE INDEX문의 옵션』
- 165 페이지의 『사용자 정의 확장 인덱스 유형 작성』
- 277 페이지의 『인덱스 특권』

태스크 관련:

- 11 페이지의 『인덱스 작성시 병렬 처리 사용』
- 158 페이지의 『인덱스 작성』
- 220 페이지의 『기존 테이블 또는 인덱스 이름 바꾸기』
- 232 페이지의 『인덱스, 인덱스 확장 또는 인덱스 스펙 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE INDEX문』
- *SQL 참조서, 볼륨 2*의 『CREATE INDEX EXTENSION문』

인덱스, 인덱스 확장자 또는 인덱스 스펙 작성에 대한 세부사항

데이터베이스 관리자가 유지보수하는 인덱스를 사용하거나 사용자 고유의 인덱스를 지정할 수 있습니다.

인덱스 작성

*인덱스*는 테이블에 있는 행을 가리키는 하나 이상의 키 세트입니다. 인덱스는 포인터를 통해 데이터에 대한 직접 경로를 작성하므로 테이블에 있는 행을 더 효율적으로 액세스할 수 있습니다.

프로시저:

성능 추가 정보: 다음 태스크를 수행하거나

1. 테이블 작성
2. 테이블 로드
3. 인덱스 작성(COLLECT STATISTICS 옵션을 사용하지 않음)
4. RUNSTATS 수행

다음 태스크를 수행하고

1. 테이블 작성
2. 테이블 로드
3. 인덱스 작성(COLLECT STATISTICS 옵션 사용)

그런 다음, 태스크 실행의 순서가 다음과 같다는 사실을 고려하십시오.

1. 테이블 작성
2. 인덱스 작성
3. `statistics yes` 옵션을 사용한 테이블 로드가 요청됨

인덱스는 작성된 후 유지보수됩니다. 그 이후, 응용프로그램이 키 값을 사용하여 테이블 내의 행을 무작위로 액세스하고 처리할 때, 해당 키 값에 따른 인덱스는 행을 직접 액세스하는 데 사용될 수 있습니다. 이는 기본 테이블에서 행의 실제 스토리지가 순서화되지 않았기 때문에 매우 중요합니다.

테이블을 작성할 때 다차원 클러스터링(MDC) 테이블을 작성할 수 있습니다. 이러한 유형의 테이블을 작성하면 블록 인덱스가 작성됩니다. 일반 인덱스는 개별 행을 가리키는 반면, 블록 인덱스는 데이터 블록이나 Extent를 가리키며 일반 인덱스보다 훨씬 작습니다. 블록 인덱스는 일반 인덱스와 함께 동일한 테이블 스페이스에 저장됩니다.

클러스터링 인덱스가 정의되지 않은 경우 행이 삽입되면, 행이 들어갈 수 있는 가장 편리한 스토리지 위치에 이 행이 위치합니다. 특정 선택 조건에 부합하는 테이블 행을 검색하고 테이블에 인덱스가 없으면, 전체 테이블이 스캔됩니다. 인덱스는 시퀀스 검색을 길게 수행하지 않고도 데이터 검색을 최적화할 수 있습니다.

인덱스에 대한 데이터는 테이블 데이터와 동일한 테이블 스페이스에 저장되거나, 인덱스 데이터를 담고 있는 별도의 테이블 스페이스에 저장될 수 있습니다. 인덱스 데이터를 저장하는데 사용되는 테이블 스페이스는 테이블이 작성될 때 결정됩니다.

제어 센터를 사용하여 인덱스를 작성하려면, 다음을 수행하십시오.

1. **인덱스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. **인덱스** 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **작성 —> 마법사를 사용한 인덱스**를 선택하십시오.
3. 태스크를 완료하려면, 마법사의 단계에 따르십시오.

명령행을 사용하여 인덱스를 작성하려면, 다음을 입력하십시오.

```
CREATE INDEX <name> ON <table_name> (<column_name>)
```

관련 개념:

- *데이터 이동 유틸리티 안내 및 참조서*의 『로드 성능 최적화』
- 159 페이지의 『인덱스 사용』
- 160 페이지의 『CREATE INDEX문의 옵션』
- 277 페이지의 『인덱스 특권』

태스크 관련:

- 220 페이지의 『기존 테이블 또는 인덱스 이름 바꾸기』
- 232 페이지의 『인덱스, 인덱스 확장 또는 인덱스 스펙 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE INDEX문』

인덱스 사용

인덱스는 응용프로그램이 직접 사용하지 않습니다. 인덱스의 사용 여부와 잠재적으로 사용 가능한 인덱스를 결정하는 것은 옵티마이저의 책임입니다.

한 테이블에서 최적의 인덱스는 다음과 같은 인덱스입니다.

- 고속 디스크를 사용하는 인덱스
- 고도로 클러스터링된 인덱스
- 몇 개의 좁은 컬럼으로만 구성되는 인덱스
- 카디낼리티(cardinality)가 높은 컬럼을 사용하는 인덱스

관련 개념:

- *관리 안내서: 성능*의 『인덱스 플랜 추가 정보』
- *관리 안내서: 성능*의 『인덱스 성능 추가 정보』

- *관리 안내서: 성능*의 『인덱스 스캔을 통해 데이터 액세스』
- *관리 안내서: 성능*의 『표준 테이블에 대한 테이블 및 인덱스 관리』
- *관리 안내서: 성능*의 『MDC 테이블에 대한 테이블 및 인덱스 관리』

CREATE INDEX문의 옵션

고유 키가 아닌 컬럼으로 좀더 효과적으로 검색할 수 있도록 중복(비고유 인덱스)을 허용하고 인덱스화된 컬럼에 중복 값이 존재할 수 있도록 인덱스를 작성할 수 있습니다.

다음 SQL문은 EMPLOYEE 테이블의 LASTNAME 컬럼으로부터 LNAME이라는 고유하지 않은 오름차순으로 정렬된 인덱스를 작성합니다.

```
CREATE INDEX LNAME ON EMPLOYEE (LASTNAME ASC)
```

다음 SQL문은 전화번호 컬럼에 고유한 인덱스를 작성합니다.

```
CREATE UNIQUE INDEX PH ON EMPLOYEE (PHONENO DESC)
```

고유 인덱스는 인덱스화된 컬럼에 중복값이 존재하지 않도록 합니다. 제한조건은 행을 갱신하거나 새 행을 삽입하는 SQL문의 끝에 강제로 적용됩니다. 이러한 유형의 인덱스는 이미 하나 이상의 중복 값을 가지고 있는 컬럼에서는 작성될 수 없습니다.

키워드 ASC는 컬럼별로 인덱스 항목을 오름차순으로 입력하고, DESC는 컬럼별로 인덱스 항목을 내림차순으로 입력합니다. 디폴트값은 오름차순입니다.

두 컬럼에서 고유 인덱스를 작성할 수 있는데, 그 중 하나는 Include 컬럼입니다. 기본 키는 Include 컬럼이 아닌 컬럼에서 정의합니다. 두 컬럼 모두 같은 테이블의 기본 키로서 카탈로그에 표시되어 있습니다. 보통 테이블마다 한 개의 기본 키만 있습니다.

INCLUDE절은 인덱스 키 컬럼 세트에 추가되는 추가 컬럼을 지정합니다. 이 절에 포함된 어떤 컬럼도 고유성을 강요하기 위해 사용할 수 없습니다. Include 컬럼은 인덱스만의 액세스를 통해 일부 쿼리의 성능을 개선합니다. 컬럼은 고유성을 강요하는 데 사용하는 컬럼과 구별되어야 합니다(그렇지 않으면, 오류 메시지인 SQLSTATE 42711을 받게 됩니다). 컬럼의 수와 길이 속성의 합계에 대한 한계는 고유 키와 인덱스의 모든 컬럼에 적용됩니다.

기존 인덱스가 기본 키 정의와 일치하는지 판별하기 위해 확인합니다(인덱스에서 임의의 INCLUDE 컬럼은 무시). 컬럼의 순서 또는 방향(오름차순 또는 내림차순) 권장 스펙에 무관하게 같은 세트의 컬럼을 식별하면, 인덱스 정의가 일치합니다. 일치하는 인덱스 정의라면 인덱스의 설명은 시스템이 요청하는 대로 1차 인덱스고, 비고유 인덱스였다면 고유 인덱스로 변경됨을 표시하도록 변경됩니다.

이것이 카탈로그에 표시된 것처럼 같은 테이블에서 기본 키를 두 개 이상 가질 수 있는 이유입니다.

구조화 유형으로 작업할 때 사용자 정의 인덱스 유형을 작성해야 합니다. 여기서는 인덱스 유지보수, 인덱스 검색 및 인덱스 개발 기능 정의 수단이 필요합니다.

다음 SQL문은 EMPLOYEE 테이블의 LASTNAME 컬럼에 INDEX1이라고 하는 클러스터링 인덱스를 작성합니다.

```
CREATE INDEX INDEX1 ON EMPLOYEE (LASTNAME) CLUSTER
```

데이터베이스의 내부 스토리지를 효과적으로 사용하려면, ALTER TABLE문과 연관된 PCTFREE 매개변수와 함께 클러스터링 인덱스를 사용하여 올바른 페이지에 새 데이터가 삽입될 수 있도록 하십시오. 올바른 페이지에 데이터가 삽입되면, 클러스터링 순서가 유지보수됩니다. 일반적으로, 테이블에서의 INSERT 활동이 더 클수록 클러스터링을 유지보수하는 데 필요한 (테이블의) PCTFREE 값이 더 커집니다. 이 인덱스는 데이터가 실제 페이지에 놓이는 순서를 판별하므로, 하나의 클러스터링 인덱스만이 특정 테이블에 정의될 수 있습니다.

예를 들어, 이들 새로운 행의 인덱스 키 값이 항상 새로운 높은 키 값일 경우, 테이블의 클러스터링 속성은 이들 인덱스 키 값을 테이블의 끝에 위치시키려고 합니다. 다른 페이지에 여유 공간이 있으면 클러스터링을 거의 보존하지 않습니다. 이 경우, 추가 모드에 테이블을 위치시키는 것이 인덱스를 클러스터링하고 테이블이 큰 PCTFREE 값을 갖도록 변경하는 것보다 나은 선택일 수 있습니다. ALTER TABLE APPEND ON을 발행하여 추가 모드에 테이블을 위치시킬 수 있습니다.

또한, 위의 설명은 행의 *크기*를 증가시키는 UPDATE의 결과인 새로운 "오버플로우" 행에 적용합니다.

CREATE INDEX문에 ALLOW REVERSE SCANS 매개변수를 사용하여 작성한 단일 인덱스는 앞뒤로 스캔할 수 있습니다. 즉, 그러한 인덱스는 인덱스 작성시 정의한 방향으로의 스캔 및 반대 방향(역방향)으로의 스캔을 지원합니다. 명령문은 다음과 유사합니다.

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

이 경우, 인덱스(iname)는 주어진 컬럼(cname)에 내림차순 값(DESC)을 기초로 형성됩니다. 컬럼에 있는 인덱스가 내림차순 스캔이 가능하도록 정의되어 있더라도, 역 스캔을 허용함으로써 오름차순(역방향) 스캔을 수행할 수 있습니다. 양방향으로의 실제 인덱스 사용은 액세스 계획을 작성 및 고려할 때 옵티마이저에 의해 제어됩니다.

CREATE INDEX문의 MINPCTUSED절은 인덱스 리프 페이지에서 사용한 최소 공간량에 대한 임계값을 지정합니다. 이 절을 사용하면, 이 인덱스에 대해 온라인 인덱스 조각 모음이 사용 가능화됩니다. 온라인 인덱스 조각 모음이 사용 가능화되면, 다음 사항을 고려하여 온라인 인덱스 조각 모음의 수행 여부를 결정합니다. 키가 이 인덱스의 리프 페이지에서 실제로 제거된 후 페이지에서 사용한 스페이스의 백분율이 지정된 임

계값보다 작으면, 이웃하는 인덱스 리프 페이지를 점검하여 두 리프 페이지에 있는 키가 단일 인덱스 리프 페이지로 병합될 수 있는지 판별합니다.

예를 들어, 다음 SQL문은 온라인 인덱스 조각 모음이 사용 가능한 인덱스를 작성합니다.

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED 20
```

이 인덱스의 인덱스 페이지에서 실제로 키가 삭제될 때 인덱스 페이지에 있는 나머지 키가 인덱스 페이지에서 20% 이하의 스페이스를 차지하면, 이 인덱스 페이지의 키와 이웃하는 인덱스 페이지의 키를 병합하여 인덱스 페이지를 삭제하려고 합니다. 결합된 키가 모든 단일 페이지에 들어갈 수 있으면, 이 병합이 수행되고 인덱스 페이지 중 하나가 삭제됩니다.

CREATE INDEX문을 사용하면, 인덱스를 작성하면서 동시에 기초가 되는 테이블 및 기존의 모든 인덱스에 대한 읽기 및 쓰기 액세스를 허용할 수 있습니다. 인덱스를 작성하는 동안 테이블에 대한 액세스를 제한하려면 인덱스를 작성하기 전에 LOCK TABLE 문을 사용하여 테이블을 잠그십시오. 새 인덱스는 기초가 되는 테이블을 스캔함으로써 작성됩니다. 인덱스 작성 중에 테이블에 가해진 모든 변경사항이 로그됩니다. 인덱스를 새로 작성하면 변경사항을 인덱스에 적용할 수 있습니다. 인덱스 작성시, 로그된 변경사항을 빨리 적용하려면 변경사항에 대한 별도의 사본을 메모리 버퍼 스페이스(유틸리티 힙(heap)의 요청으로 할당된)에서 유지보수합니다. 이런 방법으로 인덱스 작성시 먼저 메모리에서 직접 읽은 다음 로그를 읽음으로써 변경사항을 처리할 수 있습니다. 모든 변경사항이 인덱스에 적용되면, 새 인덱스가 표시되는 동안 테이블이 Quiesce 상태가 됩니다.

고유 인덱스를 작성할 때는 테이블에 중복 키가 없어야 하며 인덱스 작성시 동시 삽입으로 인해 중복 키가 생성되지 않도록 해야 합니다. 인덱스 작성은 지연된 고유 스킴을 사용하여 중복 키를 발견하므로 중복 키는 인덱스 작성의 마지막 시점에서 발견되며, 이 시점에서 인덱스 작성이 중복 키로 인해 실패합니다.

CREATE INDEX문의 PCTFREE절은 인덱스가 작성될 때 여유 공간으로 남겨둘 각 인덱스 페이지의 비율을 지정합니다. 인덱스 페이지에 여유 공간을 더 많이 두면 페이지 파티션이 줄어듭니다. 그러면 프리페치를 증가시키는 시퀀스 인덱스 페이지를 다시 얻기 위해 테이블을 재구성할 필요성이 줄어듭니다. 그리고 프리페치는 성능을 향상시키는 주요 구성요소입니다. 또한, 항상 높은 키 값이 있는 경우, CREATE INDEX문의 PCTFREE의 값을 낮추는 방안을 고려해 볼 수도 있습니다. 이렇게 하면, 각 인덱스 페이지에서 예약된 공간을 제한할 수 있습니다.

LEVEL2 PCTFREE 절을 사용하여 시스템이 인덱스 SECOND 레벨의 각 페이지에 해당되는 여유 공간에 지정된 비율만큼 확보하도록 지시할 수 있습니다. 추후 삽입 및 갱신을 위해 인덱스를 작성할 때, 사용자가 여유 공간에 대한 비율을 지정합니다. SECOND 레벨은 리프 레벨의 바로 위 레벨입니다. 디폴트로 모든 비리프 페이지에서

최소 10 및 PCTFREE 값만큼 확보합니다. LEVEL2 PCTFREE 매개변수를 사용하여 디폴트를 겹쳐쓰기할 수 있습니다. CREATE INDEX문에서 LEVEL2 PCTFREE 정수 옵션을 사용하면 2레벨 중간 페이지에 여유 공간에 대한 정수 퍼센트가 남습니다. 여유 공간에 대한 정수 퍼센트 및 최소 10이 3레벨 이상의 중간 페이지에 남습니다. SECOND 레벨에 여유 공간을 더 남김으로써 인덱스의 SECOND 레벨에 발생하는 페이지 분열 수를 줄입니다.

인덱스 삽입 시 PAGE SPLIT SYMMETRIC, PAGE SPLIT HIGH 및 PAGE SPLIT LOW 절을 사용하면 페이지 분열 동작을 선택할 수 있습니다.

PAGE SPLIT SYMMETRIC 절은 인덱스 페이지의 중간을 분열하는 디폴트 페이지 분열 동작입니다. 인덱스에 무작위 삽입 시 또는 PAGE SPLIT HIGH 및 PAGE SPLIT LOW 절과 관련된 패턴을 따르지 않을 때는 디폴트 동작을 사용하는 것이 가장 좋습니다.

PAGE SPLIT HIGH 동작은 인덱스 범위가 증가할 때 유용합니다. 인덱스 범위가 증가하는 경우는 다음과 같습니다.

- 다중 키 파트를 가진 인덱스 및 많은 값(다중 인덱스 페이지 값)이 있고 마지막 키 파트를 제외한 모든 파트가 같은 값을 가진 경우
- 테이블에 삽입된 값이 마지막 키 파트를 제외한 기존의 모든 키 파트와 같은 값을 갖는 경우
- 삽입된 값의 마지막 키 파트가 기존의 키 파트보다 큰 경우

예를 들어, 인덱스의 키 값은 다음과 같습니다.

```
(1,1),(1,2),(1,3), ... (1,n),
(2,1),(2,2),(2,3), ... (2,n),
...
(m,1),(m,2),(m,3), ...(m,n)
```

그리고 삽입할 다음 키는 1 <= x <= m 및 y > n일 때 값 (x,y)를 가집니다. 이러한 패턴을 삽입하는 경우, PAGE SPLIT HIGH 절을 사용하여 페이지 분열로 인해 여러 페이지가 50퍼센트의 공백을 남기지 않도록 할 수 있습니다.

마찬가지로 인덱스의 범위가 감소할 때 PAGE SPLIT LOW를 사용하여 페이지에 50퍼센트의 공백을 남기지 않도록 할 수 있습니다.

주: 기본 또는 고유 키를 추가하고 기본 인덱스에 SPLIT HIGH, SPLIT LOW, PCTFREE, LEVEL2 PCTFREE, MINPCTUSED, CLUSTER 또는 ALLOW REVERSE SCANS를 사용하려면 우선 원하는 키와 매개변수를 지정하여 인덱스를 작성해야 합니다. 그런 다음 ALTER TABLE문을 사용하여 기본 또는 고유 키를 추가합니다. ALTER TABLE문은 작성된 인덱스를 수정하여 재사용합니다.

인덱스 작성의 일부로서 인덱스 통계를 수집할 수 있습니다. CREATE INDEX문을 사용할 때 키 값 통계 및 실제 통계를 사용할 수 있습니다. CREATE INDEX문의 일부로서 인덱스 통계를 수집하면, CREATE INDEX문을 완료한 즉시 RUNSTATS 유틸리티를 실행하지 않아도 됩니다.

예를 들어, 다음 SQL문은 인덱스 작성의 일부로서 기본 인덱스 통계를 수집합니다.

```
CREATE INDEX IDX1 ON TABL1 (COL1) COLLECT STATISTICS
```

복제된 요약 테이블을 가지고 있는 경우, 해당 기본 테이블은 고유 인덱스를 가져야 하며 복제된 요약 테이블을 정의하는 쿼리에 인덱스 키 컬럼을 사용하십시오.

파티션 내 병렬 처리의 경우, 인덱스 작성 중에 수행된 데이터 스캐닝 및 정렬에 다중 프로세서를 사용함으로써 인덱스 작성 성능이 향상됩니다. 다중 프로세서는 *intra_parallel*을 YES(1) 또는 ANY(-1)로 설정하여 사용할 수 있습니다. 인덱스 작성 중에 사용되는 프로세서 수는 시스템에 의해 판별되며, 구성 매개변수 dft_degree, max_querydegree, 응용프로그램 런타임 정도 또는 SQL문 컴파일 정도에 의해 영향을 받지 않습니다.

다중 파티션 데이터베이스에서 고유 인덱스는 파티션 키의 상위 집합으로 정의된 상태여야 합니다.

관련 개념:

- *관리 안내서: 성능*의 『인덱스 성능 추가 정보』
- *관리 안내서: 성능*의 『인덱스 재구성』
- *관리 안내서: 성능*의 『표준 테이블에 대한 테이블 및 인덱스 관리』
- *관리 안내서: 성능*의 『온라인 인덱스 조각 모음』
- *관리 안내서: 성능*의 『MDC 테이블에 대한 테이블 및 인덱스 관리』

태스크 관련:

- 215 페이지의 『테이블 속성 변경』

관련 참조:

- *관리 안내서: 성능*의 『max_querydegree - 병렬 처리의 최대 쿼리 수준 구성 매개변수』
- *관리 안내서: 성능*의 『intra_parallel - 파티션내 병렬 처리 사용 구성 매개변수』
- *관리 안내서: 성능*의 『dft_degree - 디폴트 등급 구성 매개변수』
- *SQL 참조서, 볼륨 2*의 『CREATE INDEX문』

사용자 정의 확장 인덱스 유형 작성

사용자 정의 인덱스 유형을 지원하기 위해, DB2® Universal Database는 인덱스가 작동하는 방법을 구성하는 기본 구성요소에 대해 사용자 고유의 논리를 작성하여 적용할 수 있도록 합니다. 대체될 수 있는 해당 구성요소는 다음과 같습니다.

- 인덱스 유지보수. 이것은 인덱스 컬럼 내용을 인덱스 키에 맵핑하는 기능을 허용합니다. 그러한 맵핑은 사용자 정의 맵핑 기능을 통해 완료됩니다. 정확하게 하나의 구성 유형 컬럼이 확장 인덱스에 참여할 수 있습니다. 일반 인덱스와 달리, 확장 인덱스를 행당 둘 이상의 인덱스 항목을 가질 수 있습니다. 행당 다중 인덱스 항목은 문서에 있는 각 키워드의 각 인덱스 항목이 있는 오브젝트로서 텍스트 문서를 저장하게 할 수 있습니다.
- 인덱스 개발. 이것은 응용프로그램 설계자가 사용자 정의 함수(UDF)와 필터링 조건(범위 술어)을 연관시키게 하며, UDF는 필터링 조건과 연관되지 않으면 옵티마이저에 불투명하게 됩니다. 이것은 DB2 각 행에 대한 개별 UDF 호출을 작성하지 못하게 하고 클라이언트 및 서버 간의 컨텍스트 전환을 하지 못하게 하여, 성능을 크게 개선시킵니다.

주: 사용자 정의 함수(UDF) 정의는 결정적이어야 하며 옵티마이저로 검색될 수 있는 외부 조치를 허용하지 말아야 합니다.

또한, 선택적 데이터 필터 기능도 지정될 수 있습니다. 옵티마이저는 사용자 정의 함수(UDF)가 평가되기 전에 페치된 튜플(tuple)에 대해 필터를 사용합니다.

구조화 유형 또는 구별 컬럼 유형만이 인덱스 확장을 사용하여 이 오브젝트에서 사용자 정의 확장 인덱스 유형을 작성할 수 있습니다. 사용자 정의 확장 인덱스 유형은 다음이 아니어야 합니다.

- 클러스터된 인덱스로 정의된 상태
- INCLUDE 컬럼이 있음

관련 개념:

- 166 페이지의『인덱스 유지보수 세부사항』
- 166 페이지의『인덱스 검색 세부사항』
- 167 페이지의『인덱스 사용에 대한 세부사항』
- 168 페이지의『인덱스 확장 정의 시나리오』

사용자 정의 확장 인덱스 유형 작성에 대한 세부사항

이 섹션에서는 사용자 고유의 확장 인덱스 유형을 만들 때 필요한 여러 가지 측면을 설명합니다.

인덱스 유지보수 세부사항

CREATE INDEX EXTENSION문을 통해 인덱스의 조작을 구성하는 두 개의 구성요소를 정의합니다.

인덱스 유지보수는 인덱스 컬럼 내용(또는 소스 키)을 목표 인덱스 키로 변환하는 프로세스입니다. 변환 프로세스는 이전에 데이터베이스에서 정의된 테이블 함수를 사용하여 정의됩니다.

FROM SOURCE KEY절은 이 인덱스 확장으로 지원되는 소스 키 컬럼에 대한 구조화 데이터 유형 또는 구별 유형을 지정합니다. 단일 매개변수 이름 및 데이터 유형이 제공되며 소스 키 컬럼과 연관됩니다.

GENERATE KEY USING절은 인덱스 키를 생성하기 위해 사용되는 사용자 정의 테이블 함수를 지정합니다. 이 함수의 출력은 TARGET KEY절 스펙에 지정되어야 합니다. 이 함수의 출력은 FILTER USING절에서 지정한 인덱스 필터링 함수의 입력으로서 사용될 수 있습니다.

관련 개념:

- 165 페이지의『사용자 정의 확장 인덱스 유형 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE INDEX EXTENSION문』

인덱스 검색 세부사항

인덱스 검색은 검색 인수를 검색 범위에 맵핑합니다.

CREATE INDEX EXTENSION문의 WITH TARGET KEY절은 GENERATE KEY USING절에 지정된 사용자 정의 테이블 함수의 출력인 목표 키 매개변수를 지정합니다. 단일 매개변수 이름 및 데이터 유형이 제공되며 목표 키 컬럼과 연관됩니다. 이 매개변수는 GENERATE KEY USING절의 사용자 정의 테이블 함수의 RETURNS 테이블 컬럼에 대응합니다.

SEARCH METHODS절은 인덱스용으로 정의된 하나 이상의 검색 메소드를 소개합니다. 각 검색 메소드는 메소드 이름, 검색 인수, 함수 생성 범위 및 선택적 인덱스 필터 함수로 구성됩니다. 각 검색 메소드는 주요 사용자 정의 인덱스의 인덱스 검색 범위가 사용자 정의 테이블 함수에 의해 생성되는 방법을 정의합니다. 더욱이, 각 검색 메소드는 특정 검색 범위에 있는 인덱스 항목이 사용자 정의 스칼라 함수에 의해 자세히 규정될 수 있는 방식을 정의하여 단일 값을 리턴합니다.

- WHEN절은 레이블을 검색 메소드와 연관시킵니다. 레이블은 인덱스 개발 규칙(사용자 정의 함수(UDF)의 PREDICATES절에 있음)에 지정된 메소드 이름과 관련된 SQL ID입니다. 범위 함수(인덱스 필터링 함수를 포함하거나 포함하지 않음)에서 인수로 사

용하도록 하나 이상의 매개변수 이름과 데이터 유형이 제공됩니다. WHEN절은 CREATE FUNCTION문의 PREDICATES절이 입력 쿼리와 일치할 때 옵티마이저가 취할 수 있는 조치를 지정합니다.

- RANGE THROUGH절은 인덱스 키 범위를 생성하는 사용자 정의 외부 테이블 함수를 지정합니다. 이것은 인덱스 키가 키 범위를 벗어날 때 옵티마이저가 연관된 UDF를 호출하지 못하게 할 수 있습니다.
- FILTER USING절은 범위 생성 함수에서 리턴된 인덱스 항목을 필터하기 위해 사용되는 CASE 표현식 또는 사용자 정의 외부 테이블 함수를 지정하는 선택적 방법입니다. 인덱스 필터 함수 또는 CASE 표현식에 의해 리턴된 값이 1이면, 인덱스 항목에 대응하는 행이 테이블에서 검색됩니다. 리턴되는 값이 1이 아닌 경우, 인덱스 항목을 버립니다. 이 기능은 두 번째 필터의 비용이 원래 메소드를 평가하는 비용과 비교할 때 더 낮고 보조 필터의 선택성이 상대적으로 낮을 때 유용합니다.

관련 개념:

- 165 페이지의『사용자 정의 확장 인덱스 유형 작성』
- 166 페이지의『인덱스 유지보수 세부사항』
- 167 페이지의『인덱스 사용에 대한 세부사항』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE INDEX EXTENSION문』

인덱스 사용에 대한 세부사항

인덱스 검색은 검색 메소드의 평가에서 발생합니다.

CREATE FUNCTION(외부 스칼라)문은 인덱스 확장용으로 정의된 검색 메소드에서 사용되는 사용자 정의 술어를 작성합니다.

PREDICATES절은 인덱스 확장을 검색할 수 있는(그리고 술어의 검색 조건에 대해 선택적 SELECTIVITY절 사용) 이 함수를 사용하여 술어를 식별합니다. PREDICATES절이 지정되면, 함수는 NO EXTERNAL ACTION을 사용한 DETERMINISTIC으로서 정의되어야 합니다.

- WHEN절은 비교 연산자(=, >, < 및 기타) 및 상수 또는 표현식(EXPRESSION AS절 사용)이 있는 술어에서 정의되는 함수의 특정 사용을 소개합니다. 술어가 동일한 비교 연산자와 제공된 상수 또는 표현식과 함께 이 함수를 사용하는 경우, 필터링 및 인덱스 검색이 사용될 수 있습니다. 상수의 사용은 결과 유형이 1 또는 0 중 하나인 부울 표현식을 다루기 위해 주로 제공됩니다. 다른 모든 경우에는, EXPRESSION AS절이 더 낫습니다.
- FILTER USING절은 결과 테이블의 추가 필터링을 수행하기 위해 사용할 수 있는 필터 함수를 식별합니다. 행이 규정하는지 여부를 판별하기 위해 사용자 정의 술어를 실행해야 하는 행의 수를 감축하는 정의된 함수(술어에서 사용)를 대체하는 더 빠른

버전입니다. 사용자 정의 술어에 의해 예상되는 결과에 근접하는 인덱스로 결과를 생성한 경우, 이 필터 함수의 응용프로그램은 중복될 수 있습니다.

- 인덱스 확장의 각 검색 메소드가 인덱스를 검색하는 규칙 세트를 선택적으로 정의할 수 있습니다. 또한 검색 목표, 검색 인수 그리고 인덱스 검색을 수행하기 위해 이들을 사용하는 방법을 기술하기 위해 인덱스 확장에서 검색 메소드를 정의할 수 있습니다.
  - SEARCH BY INDEX EXTENSION절은 인덱스 확장을 식별합니다.
  - 선택적 EXACT절은 인덱스 찾아보기가 술어 평가에서 일치함을 나타냅니다. 이 절은 인덱스 찾아보기 이후에 데이터베이스가 필터 함수나 원래의 사용자 제공 술어 함수를 적용하지 않도록 알립니다. 인덱스 찾아보기가 사용되지 않는 경우, 원래의 술어 및 필터 함수가 적용되어야 합니다. EXACT절이 사용되지 않는 경우, 원래의 사용자 제공 술어는 인덱스 찾아보기 이후에 적용됩니다. EXACT 술어는 인덱스 찾아보기가 술어와 동일한 결과를 리턴할 때 유용합니다. 이것은 쿼리 실행시 인덱스 찾아보기에서 획득한 결과에 사용자 정의 술어를 적용하지 못하게 합니다. 인덱스가 술어와 비슷한 것만을 제공하리라고 예상되는 경우, EXACT절을 지정하지 마십시오.
  - WHEN KEY절은 검색 목표를 정의합니다. 키에 대해 오직 하나의 검색 목표가 지정됩니다. WHEN KEY절 다음에 제공된 값은 정의되는 함수의 매개변수 이름을 식별합니다. 이름 지정된 매개변수의 값이 지정된 인덱스 확장에 근거한 인덱스로 처리될 때 이 절은 참으로 평가됩니다.
  - USE절은 검색 인수를 정의합니다. 검색 인수는 인덱스 확장에 정의된 어떤 메소드가 사용되는가를 식별합니다. 여기에서 제공한 메소드 이름은 인덱스 확장에 정의한 메소드와 일치해야 합니다. 하나 이상의 매개변수 값은 정의되는 함수의 매개변수 이름을 식별하며 어떤 것이 검색 목표에 지정한 매개변수 이름과 달라야 하는지 나타냅니다. 매개변수 값의 수와 각 데이터 유형은 인덱스 확장에서 메소드용으로 정의한 매개변수와 일치해야 합니다. 내장 및 구별 데이터 유형의 경우 완전히 일치해야 하며 동일한 구조 유형 내에 있어야 합니다.

관련 개념:

- 165 페이지의 『사용자 정의 확장 인덱스 유형 작성』
- 166 페이지의 『인덱스 유지보수 세부사항』
- 166 페이지의 『인덱스 검색 세부사항』
- 168 페이지의 『인덱스 확장 정의 시나리오』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE FUNCTION(외부 스칼라)문』

인덱스 확장 정의 시나리오

인덱스 확장을 정의하는 시나리오는 다음과 같습니다.

1. 구조화 유형(형태의 경우)을 정의하십시오. CREATE TYPE문을 사용하여 형태가 슈퍼 유형 및 널(Null) 모양, 포인트, 라인이며, 다각형이 하위 유형인 유형 계층 구조를 정의하십시오. 이것은 유형 모델 공간 엔터티를 구조화했습니다. 예를 들어, 상점의 위치는 포인트이며, 강의 경로는 라인이며, 비즈니스 구역의 바운더리는 다각형입니다. 최소 경계의 사각형(mbr)은 속성입니다. gtype 속성은 연관된 엔터티가 포인트, 라인 또는 다각형인지 여부를 지정합니다. 지리적인 바운더리는 numpart, numpoint 및 geometry 속성으로 모델화됩니다. 기타 모든 속성은 이 시나리오와 관련이 없으므로 무시됩니다.
2. 인덱스 확장을 작성하십시오.
   - CREATE FUNCTION문을 사용하여 키 변환(gridentry), 범위 생성(gridrange) 및 인덱스 필터링(checkduplicate 및 mbroverlap)에 사용되는 함수를 작성하십시오.
   - CREATE INDEX EXTENSION 문을 사용하여 인덱스의 구성요소에 필요한 나머지를 작성하십시오.
3. 인덱스의 인덱스 유지보수 구성요소에 대응하는 키 변환을 작성하십시오.

   ```
   CREATE INDEX EXTENSION iename (parm_name datatype, ...)
      FROM SOURCE KEY (parm_name datatype)
      GENERATE KEY USING table_function_invocation
      ...
   ```

   FROM SOURCE KEY절은 키 변환의 매개변수 및 데이터 유형을 식별합니다. GENERATE KEY USING절은 함수에서 생성된 값으로 소스 키를 맵핑하는 데 사용되는 함수를 식별합니다.
4. 인덱스의 인덱스 검색 구성요소에 대응하는 범위 생성 및 인덱스 필터 함수를 정의하십시오.

   ```
   CREATE INDEX EXTENSION iename (parm_name datatype, ...)
      ...
      WITH TARGET KEY
         WHEN method_name (parm_name datatype, ...)
         RANGE THROUGH range_producing_function_invocation
         FILTER USING index_filtering_function_invocation
   ```

   WITH TARGET KEY절은 검색 메소드 정의를 식별합니다. WHEN절은 메소드 이름을 식별합니다. RANGE THROUGH절은 사용되는 인덱스의 범위를 제한하는 데 사용되는 함수를 식별합니다. FILTER USING절은 결과 인덱스 값에서 불필요한 항목을 제거하는 데 사용되는 함수를 식별합니다.

   주: FILTER USING절은 인덱스 필터링 함수 대신 CASE 표현식을 식별할 수 있습니다.
5. 인덱스 확장을 검색하려면 술어를 정의하십시오.

   ```
   CREATE FUNCTION within (x shape, y shape)
      RETURNS INTEGER
      ...
   ```

```
PREDICATES
   WHEN = 1
      FILTER USING mbrWithin (x..mbr..xmin, ...)
      SEARCH BY INDEX EXTENSION grid_extension
      WHEN KEY (parm_name) USE method_name(parm_name)
```

PREDICATES절은 각 WHEN절로 시작되는 하나 이상의 술어를 소개합니다. WHEN절은 상수 또는 EXPRESSION AS절 중 하나가 다음에 오는 비교 연산자를 사용하여 술어에 대한 스펙을 시작합니다. FILTER USING절은 결과 테이블의 추가 필터링을 수행하기 위해 사용할 수 있는 필터 함수를 식별합니다. 이것은 규정하는 행을 판별하기 위해 사용자 정의 술어가 실행되어야 하는 행의 수를 감축하는 정의된 함수(술어에서 사용)의 비용이 저렴한 버전입니다. SEARCH BY INDEX EXTENSION절은 인덱스 검색이 발생하는 곳을 지정합니다. 인덱스 검색은 인덱스를 검색하기 위해 사용될 수 있는 인덱스 확장의 검색 메소드를 사용하여 규칙 세트를 정의합니다. WHEN KEY절은 검색 규칙을 지정합니다. 검색 규칙은 검색 메소드를 통해 인덱스 검색을 수행하는 데 사용될 수 있는 방법은 물론 검색 목표 및 검색 인수를 설명합니다.

6. 필터 함수를 정의하십시오.

   ```
   CREATE FUNCTION mbrWithin (...)
   ```

   여기에 정의된 함수는 인덱스 확장의 술어에서 사용하도록 작성됩니다.

쿼리 옵티마이저가 작성된 인덱스를 정상적으로 검색하여 쿼리 성능을 개선시키기 위해, SELECTIVITY 옵션이 함수 호출에서 사용 가능합니다. 술어가 리턴할 수 있는 행의 백분율을 알고 있는 경우에는, 함수 호출에서 SELECTIVITY 옵션을 사용하여 DB2® 옵티마이저가 보다 효과적인 액세스 경로를 선택하도록 도울 수 있습니다.

다음 예에서 within 사용자 정의 함수(UDF)는 중심 및 반경(각각 첫 번째와 두 번째 매개변수에 근거하여)을 계산하고 적절한 선택으로 명령문 문자열을 빌드합니다.

```
SELECT * FROM customer
   WHERE within(loc, circle(100, 100, 10)) = 1 SELECTIVITY .05
```

이 예에서 표시된 술어(SELECTIVITY .05)는 customer 테이블에서 95퍼센트의 행을 필더링합니다.

관련 개념:

- 165 페이지의 『사용자 정의 확장 인덱스 유형 작성』
- 166 페이지의 『인덱스 유지보수 세부사항』
- 166 페이지의 『인덱스 검색 세부사항』
- 167 페이지의 『인덱스 사용에 대한 세부사항』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE INDEX EXTENSION문』

- *SQL 참조서, 볼륨 2*의 『CREATE FUNCTION(외부 스칼라)문』

명령행 처리기를 통해 구성 어드바이저 호출

전제조건:

데이터베이스가 이미 작성되었습니다.

프로시저:

데이터베이스를 작성한 후, AUTOCONFIGURE 명령을 사용하여 구성 어드바이저를 호출할 수 있습니다. 데이터베이스 작성시 AUTOCONFIGURE 옵션을 선택하면 이 메소드를 사용할 수 있습니다.

AUTOCONFIGURE 옵션을 사용하여 여러 구성 매개변수의 값을 정의하고 해당 매개변수를 사용하는 응용프로그램의 범위를 판별할 수 있습니다. 범위는 NONE, DB ONLY 또는 DB AND DBM이 될 수 있습니다. NONE은 어떠한 값도 적용되지 않음을 의미하고, DB ONLY는 데이터베이스 구성 및 버퍼 풀 값만이 적용됨을 의미하며, DB AND DBM은 모든 매개변수 및 해당 값이 적용됨을 의미합니다.

관련 개념:

- *관리 안내서: 성능*의 『구성 매개변수』

관련 참조:

- *Command Reference*의 『AUTOCONFIGURE Command』

제 5 장 데이터베이스 경고

이 장은 데이터베이스를 변경하기 전에 고려해야 하는 사항과 데이터베이스 오브젝트를 변경하거나 삭제(drop)하는 방식에 초점을 둡니다.

인스턴스 변경

데이터베이스 설계가 구현된 후 데이터베이스 설계를 변경하고자 하는 경우가 있습니다. 이전 설계와 관련된 주요 설계 문제를 재검토해야 합니다.

전체 데이터베이스에 영향을 주는 변경을 수행하기 전에, 모든 논리 및 물리 설계 결정을 검토해야 합니다. 예를 들어, 테이블 스페이스를 변경할 때에는 SMS 또는 DMS 스토리지 유형과 관련하여 설계 결정을 검토해야 합니다.

DB2 Universal Database™(DB2 UDB) 제품 라이센스 관리의 일부로서, 라이센스 수를 늘려야 하는 경우가 있을 수도 있습니다. 제어 센터에 있는 라이센스 센터를 사용하여 설치된 제품의 사용량을 검사하고 해당 사용량을 근거로 라이센스의 수를 늘릴 수 있습니다.

다음 사항에 특히 주의를 기울여야 합니다.

- 『인스턴스 변경(UNIX에만 해당)』
- 178 페이지의 『노드 및 데이터베이스 구성 파일 변경』

인스턴스 변경(UNIX에만 해당)

인스턴스는 제품의 후속 설치 및 제거의 영향에서 가능한 벗어나도록 설계됩니다.

대부분의 경우, 기존 인스턴스는 설치 또는 제거되는 제품의 기능을 상속하거나 이에 대한 액세스를 유실합니다. 그러나 특정 실행 파일 또는 구성요소가 설치되거나 제거되면, 기존의 인스턴스는 새 시스템 구성 매개변수를 자동으로 상속하거나 모든 추가 기능에 대한 액세스를 확보하지는 않습니다. 인스턴스는 갱신되어야 합니다.

프로그램 임시 수정(PTF) 또는 패치를 설치하여 DB2® Universal Database(DB2 UDB)를 갱신할 경우, **db2iupdt** 명령을 사용하여 모든 기존 DB2 UDB 인스턴스를 갱신해야 합니다.

인스턴스를 변경하거나 삭제하기 전에 인스턴스와 인스턴스에 있는 데이터베이스 파티션 서버에 대해 이해하고 있어야 합니다.

관련 개념:

- 17 페이지의 『인스턴스 작성』

태스크 관련:

- 174 페이지의 『UNIX에서 인스턴스 구성 갱신』
- 177 페이지의 『인스턴스 제거』

관련 참조:

- *Command Reference*의 『db2iupdt - Update Instances Command』

인스턴스 변경 세부사항

인스턴스를 변경하기 전에 기존의 모든 인스턴스를 나열해야 합니다.

인스턴스 나열

프로시저:

제어 센터를 사용하여 시스템에서 사용할 수 있는 모든 인스턴스 목록을 확보하려면, 다음을 수행하십시오.

1. **인스턴스** 폴더가 나올 때까지 오브젝트 트리를 확장하십시오.
2. 인스턴스 폴더를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **추가**를 선택하십시오.
3. 인스턴스 추가 창에서 **새로 고침**을 누르십시오.
4. 데이터베이스 인스턴스를 보려면 드롭 다운 화살표를 누르십시오.
5. 창을 나가려면 **취소**를 누르십시오.

명령행을 사용해서 시스템에서 사용할 수 있는 모든 인스턴스의 목록을 확보하려면, 다음을 입력하십시오.

```
db2ilist
```

현재 세션에 적용되는 인스턴스를 판별하려면(지원되는 Windows 플랫폼에서) 다음을 사용하십시오.

```
set db2instance
```

UNIX에서 인스턴스 구성 갱신

db2iupdt 명령을 수행하면, 다음을 수행하여 지정된 인스턴스를 갱신합니다.

- 인스턴스 소유자의 홈 디렉토리 아래에 있는 `sqllib` 서브디렉토리의 파일을 바꿉니다.
- 노드 유형이 변경되면, 새 데이터베이스 관리 프로그램 구성 파일이 작성됩니다. 이는 기존의 데이터베이스 관리 프로그램 구성 파일에서 관련 값을 새 노드 유형에 대한 데이터베이스 관리 프로그램 구성 파일과 병합하여 완료됩니다. 새 데이터베이스 관리 프로그램 구성 파일이 작성되면, 이전 파일은 인스턴스 소유자의 홈 디렉토리 아래에 있는 `sqllib` 서브디렉토리의 `backup` 서브디렉토리로 백업됩니다.

프로시저:

AIX에서는 /usr/opt/db2_08_01/instance/ 디렉토리에 **db2iupdt** 명령이 있습니다. HP-UX, Solaris 운영 환경 또는 Linux에서는 /opt/IBM/db2/V8.1/instance/ 디렉토리에 **db2iupdt** 명령이 있습니다.

명령은 다음과 같이 사용됩니다.

```
db2iupdt InstName
```

InstName은 인스턴스 소유자의 로그인 이름입니다.

이 명령과 연관된 기타 선택적 매개변수가 있습니다.

- -h 또는 -?

  이 명령에 대한 도움말 메뉴를 표시합니다.

- -d

  문제점 판별 중에 사용할 디버그 모드를 설정합니다.

- -a AuthType

  인스턴스에 대한 인증 유형을 지정합니다. 유효한 인증 유형은 SERVER, SERVER_ENCRYPT 또는 CLIENT입니다. 지정되지 않은 경우, DB2 서버가 설치되면 디폴트값은 SERVER입니다. 그렇지 않으면, CLIENT로 설정됩니다. 인스턴스의 인증 유형은 인스턴스에 속하는 모든 데이터베이스에 적용됩니다.

- -e

  존재하는 각 인스턴스를 갱신하도록 허용합니다. **db2ilist**를 사용하면 존재하는 인스턴스를 표시할 수 있습니다.

- -u Fenced ID

  분리 사용자 정의 함수(UDF) 및 스토어드 프로시저가 실행되는 사용자를 지정합니다. DB2 클라이언트 또는 DB2 Software Developer's Kit를 설치하는 경우 이것은 필요하지 않습니다. 기타 DB2 제품의 경우, 필수 매개변수입니다.

  주: 분리(Fenced) ID는『root』또는『bin』이 될 수 없습니다.

- -k

  이 매개변수는 현재 인스턴스 유형을 보존합니다. 이 매개변수를 지정하지 않으면, 현재 인스턴스는 다음 순서 중에서 가장 높은 인스턴스 유형으로 업그레이드됩니다.
  - 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버(DB2 Enterprise - Extended Edition 디폴트 인스턴스 유형)
  - 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버(DB2 Universal Database Enterprise Server Edition 디폴트 인스턴스 유형)

– 클라이언트(DB2 클라이언트 디폴트 인스턴스 유형)

예:

- 인스턴스가 작성된 후에 DB2 Universal Database Workgroup Server Edition 또는 DB2 Universal Database Enterprise Server Edition을 설치했으면 다음 명령을 입력하여 해당 인스턴스를 갱신하십시오.

```
db2iupdt -u db2fenc1 db2inst1
```

- 인스턴스를 작성한 후 DB2 Connect Enterprise Edition을 설치하면, 인스턴스 이름을 분리(Fenced) ID로 사용할 수도 있습니다.

```
db2iupdt -u db2inst1 db2inst1
```

- 클라이언트 인스턴스를 갱신하려면, 다음 명령을 사용할 수 있습니다.

```
db2iupdt db2inst1
```

태스크 관련:

- 177 페이지의 『인스턴스 제거』

관련 참조:

- *Command Reference*의 『db2ilist - List Instances Command』
- *Command Reference*의 『db2iupdt - Update Instances Command』

Windows에서 인스턴스 구성 갱신

db2iupdt 명령을 수행하면, 다음을 수행하여 지정된 인스턴스를 갱신합니다.

- 인스턴스 소유자의 홈 디렉토리 아래에 있는 `sqllib` 서브디렉토리의 파일을 바꿉니다.
- 노드 유형이 변경되면, 새 데이터베이스 관리 프로그램 구성 파일이 작성됩니다. 이는 기존의 데이터베이스 관리 프로그램 구성 파일에서 관련 값을 새 노드 유형에 대한 데이터베이스 관리 프로그램 구성 파일과 병합하여 완료됩니다. 새 데이터베이스 관리 프로그램 구성 파일이 작성되면, 이전 파일은 인스턴스 소유자의 홈 디렉토리 아래에 있는 `sqllib` 서브디렉토리의 `backup` 서브디렉토리로 백업됩니다.

프로시저:

db2iupdt 명령은 `\sqllib\bin` 디렉토리에 있습니다.

명령은 다음과 같이 사용됩니다.

```
db2iupdt InstName
```

`InstName`은 인스턴스 소유자의 로그인 이름입니다.

이 명령과 연관된 기타 선택적 매개변수가 있습니다.

- /h: hostname

현재 머신에 대해 하나 이상의 TCP/IP 호스트 이름이 있으면 디폴트 TCP/IP 호스트 이름을 겹쳐씁니다.

- /p: instance profile path

  갱신된 인스턴스의 새 인스턴스 프로파일 경로를 지정합니다.

- /r: baseport,endport

  파티션된 데이터베이스 인스턴스가 다중 파티션에서 실행될 때 사용하는 TCP/IP 포트의 범위를 지정합니다.

- /u:username,password

  DB2 서비스에 대한 어카운트 이름 및 암호를 지정합니다.

태스크 관련:

- 27 페이지의『인스턴스 나열』
- 174 페이지의『UNIX에서 인스턴스 구성 갱신』
- 177 페이지의『인스턴스 제거』

인스턴스 제거

프로시저:

제어 센터를 사용하여 인스턴스를 제거하려면, 다음을 수행하십시오.

1. 제거하려는 인스턴스를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 인스턴스 이름을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 인스턴스를 제거하려면, 다음을 입력하십시오.

```
db2idrop <instance_name>
```

명령행을 사용하여 인스턴스를 제거하는 준비 및 세부사항은 다음과 같습니다.

1. 현재 인스턴스를 사용 중인 모든 응용프로그램을 중지시키십시오.
2. 각 DB2 명령 창에서 **db2 terminate** 명령을 수행하여 명령행 처리기를 중지시키십시오.
3. **db2stop** 명령을 수행하여 인스턴스를 중지시키십시오.
4. DB2INSTPROF 레지스트리 변수로 표시된 인스턴스 디렉토리를 백업하십시오.

   UNIX 운영 체제에서 INSTHOME/sqllib/sqllib 디렉토리(여기서, INSTHOME은 인스턴스 소유자의 홈 디렉토리임)에 있는 파일의 백업을 고려하십시오. 예를 들어, 데이터베이스 관리 프로그램 구성 파일, db2systm, db2nodes.cfg 파일, 사용자 정의 함수(UDF) 또는 분리 스토어드 프로시저 응용프로그램을 저장할 수 있습니다.

5. (UNIX 운영 체제에서만) 인스턴스 소유자로 로그오프하십시오.
6. (UNIX 운영 체제에서만) root 권한을 가진 사용자로 로그인하십시오.
7. **db2idrop** 명령을 발행하십시오.

   ```
   db2idrop InstName
   ```

   여기서, `InstName`은 제거 중인 인스턴스 이름입니다.

   이 명령은 인스턴스 목록에서 인스턴스 항목을 제거한 후 인스턴스 디렉토리를 제거합니다.
8. (UNIX 운영 체제에서만) 선택적으로, root 권한을 가진 사용자로서 인스턴스 소유자의 사용자 ID 및 그룹(해당 인스턴스에 대해서만 사용된 경우)을 제거하십시오. 인스턴스를 재작성할 계획이면, 제거하지 마십시오.

   이 단계는 인스턴스 소유자 및 인스턴스 소유자 그룹이 다른 목적으로 사용될 수 있으므로 선택적입니다.

db2idrop 명령은 인스턴스 목록에서 인스턴스 항목을 제거한 후, 인스턴스 소유자의 홈 디렉토리 아래에 있는 `sqllib` 서브디렉토리를 제거합니다.

주: UNIX 운영 체제에서 db2idrop 명령을 사용하여 인스턴스를 삭제하려고 하면, `sqllib` 서브디렉토리를 제거할 수 없다는 메시지가 생성되고 `adm` 서브디렉토리에 확장자가 .nfs인 몇몇 파일이 생성됩니다. `adm` 서브디렉토리는 NFS 마운트 시스템이며 해당 파일은 서버에서 제어됩니다. 디렉토리가 마운트되는 파일 서버에서 *.nfs 파일을 삭제해야 합니다. 그런 다음에 `sqllib` 서브디렉토리를 제거할 수 있습니다.

관련 참조:

- *Command Reference*의 『db2stop - Stop DB2 Command』
- *Command Reference*의 『TERMINATE Command』
- *Command Reference*의 『STOP DATABASE MANAGER Command』
- *Command Reference*의 『db2idrop - Remove Instance Command』
- *Command Reference*의 『db2ilist - List Instances Command』

노드 및 데이터베이스 구성 파일 변경

데이터베이스 구성 파일을 갱신하려면, 제어 센터에 있는 구성 어드바이저를 사용하거나 적절한 옵션으로 *db2 autoconfigure*를 실행하십시오. 구성 어드바이저는 어떤 구성 매개변수를 수정할 것인지 제안하고, 제안된 값을 제공함으로써 성능 조정을 하고, 인스턴스별 단일 데이터베이스의 메모리 요구사항의 밸런스를 맞추도록 도와줍니다.

주: 매개변수를 수정하려는 경우, 값은 다음과 같을 때까지 갱신되지 않습니다.

- 데이터베이스 매개변수의 경우, 모든 응용프로그램이 연결해제된 후에 데이터베이스에 대한 새로운 첫 번째 연결
- 데이터베이스 관리 프로그램 매개변수의 경우, 인스턴스를 중지시킨 후 다시 시작할 때

대부분의 경우, 구성 어드바이저에 의해 권장된 값은 디폴트값보다 더 나은 성능을 제공하는데, 그 이유는 워크로드 및 사용자의 특정 서버에 대한 정보에 근거한 값이기 때문입니다. 그러나 값은 데이터베이스 시스템의 성능을 반드시 최적화한 것은 아니지만, 향상시킬 목적으로 설계되었습니다. 최적화된 성능을 얻기 위해 조정이 가능한 시작 지점으로 값을 생각하십시오.

전제조건:

데이터베이스 파티션 그룹을 변경하려면(파티션 추가 또는 삭제, 기존 파티션 이동) 노드 구성 파일을 갱신해야 합니다.

데이터베이스를 변경하기로 계획 중일 경우, 구성 매개변수 값을 검토해야 합니다. 데이터베이스 사용에 기초하여 수행 중인 데이터베이스 변경의 일환으로서 때때로 일부 값을 조정할 수 있습니다.

프로시저:

제어 센터를 사용하여 데이터베이스 구성을 갱신하려면, 다음을 수행하십시오.

1. **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 변경하려는 인스턴스 또는 데이터베이스를 마우스 오른쪽 단추로 누른 후, **구성 어드바이저**를 누르십시오.
3. 각 페이지를 누른 후, 필요하면 정보를 변경하십시오.
4. **결과** 페이지를 눌러 구성 매개변수에 대해 제안된 변경사항을 검토하십시오.
5. 갱신 내용을 적용 또는 저장하려면 **완료**를 누르십시오.

명령행에서 구성 어드바이저를 사용하려면 AUTOCONFIGURE 명령을 사용하십시오.

명령행을 사용하여 데이터베이스 관리 프로그램 구성에 있는 개별 매개변수를 갱신하려면 다음을 입력하십시오.

```
UPDATE DBM CFG FOR <database_alias>
   USING <config_keyword>=<value>
```

하나 이상의 `<config_keyword>=<value>` 조합을 단일 명령에서 갱신할 수 있습니다. 데이터베이스 관리 프로그램 구성 파일에 대한 대부분의 변경사항은 메모리로 로드된 후에만 유효합니다. 서버 구성 매개변수의 경우, 이것은 START DATABASE MANAGER 명령의 수행 중 발생합니다. 클라이언트 구성 매개변수의 경우, 이것은 응용프로그램이 재시작될 때 발생합니다.

현재 데이터베이스 관리 프로그램 구성 매개변수를 보거나 인쇄하려면, GET DATABASE MANAGER CONFIGURATION 명령을 사용하십시오.

관련 개념:

- *관리 안내서: 성능*의 『벤치마크 테스팅』

태스크 관련:

- 180 페이지의 『다중 파티션에서의 데이터베이스 구성 변경』
- *관리 안내서: 성능*의 『구성 매개변수를 사용하여 DB2 구성』

관련 참조:

- *Command Reference*의 『GET DATABASE MANAGER CONFIGURATION Command』
- *Command Reference*의 『UPDATE DATABASE MANAGER CONFIGURATION Command』

다중 파티션에서의 데이터베이스 구성 변경

프로시저:

둘 이상의 파티션에 걸쳐 파티션된 데이터베이스가 있으면, 데이터베이스 구성 파일은 모든 데이터베이스 파티션에서 동일해야 합니다. SQL 컴파일러는 노드 구성 파일의 정보에 근거하여 분산 SQL문을 컴파일하고, SQL문의 요구를 충족시킬 액세스 플랜을 작성하기 때문에 일관성이 요구됩니다. 데이터베이스 파티션에서 다른 구성 파일을 유지보수하는 작업은 명령문이 준비된 데이터베이스 파티션에 따라 다른 액세스 플랜이 될 수 있습니다. 구성 파일이 모든 데이터베이스 파티션에서 유지보수되게 하려면 **db2_all**을 사용하십시오.

관련 개념:

- 403 페이지의 『파티션된 데이터베이스 환경에서 명령 발행』

태스크 관련:

- 178 페이지의 『노드 및 데이터베이스 구성 파일 변경』

데이터베이스 경고

데이터베이스를 변경할 때에는 데이터베이스를 작성할 때 만큼의 태스크를 수행해야 합니다. 이 태스크는 이전에 작성된 데이터베이스의 양상을 갱신하거나 제거합니다. 수행할 태스크에는 다음과 같은 것이 있습니다.

- 181 페이지의 『데이터베이스 삭제』
- 182 페이지의 『데이터베이스 파티션 그룹 변경』
- 182 페이지의 『테이블 스페이스 변경』
- 193 페이지의 『스키마 삭제』

- 195 페이지의 제 6 장『테이블 및 기타 관련 테이블 오브젝트 변경』
- 219 페이지의『사용자 정의 구조화된 유형 변경』
- 220 페이지의『유형이 지정된 테이블의 행 삭제 및 갱신』
- 220 페이지의『기존 테이블 또는 인덱스 이름 바꾸기』
- 223 페이지의『테이블 삭제』
- 225 페이지의『사용자 정의 임시 테이블 삭제』
- 225 페이지의『트리거 삭제』
- 226 페이지의『사용자 정의 함수(UDF), 함수 맵핑 또는 메소드 삭제』
- 227 페이지의『사용자 정의 유형(UDT) 또는 유형 맵핑 삭제』
- 228 페이지의『뷰 변경 및 삭제』
- 229 페이지의『작동 불능 뷰 복구 중』
- 230 페이지의『구체화된 쿼리 또는 스테이징 테이블 삭제』
- 231 페이지의『작동 불능 요약 테이블 복구』
- 232 페이지의『인덱스, 인덱스 확장 또는 인덱스 스펙 삭제』
- 233 페이지의『오브젝트 변경 시 명령문 종속성』

데이터베이스 삭제

프로시저:

비록 데이터베이스에 있는 일부 오브젝트를 변경할 수는 있어도, 데이터베이스 자체를 변경할 수는 없습니다. 제거하거나 재작성해야 합니다. 데이터베이스를 제거하게 되면 해당되는 모든 오브젝트, 컨테이너 및 연관된 파일이 삭제되므로 파급 효과가 매우 커집니다. 제거된 데이터베이스는 데이터베이스 디렉토리에서 제거(카탈로그 해제)됩니다.

제어 센터를 사용하여 데이터베이스를 제거하려면, 다음을 수행하십시오.

1. **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 제거하려는 데이터베이스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 선택하십시오.
3. **확정** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 데이터베이스를 제거하려면, 다음을 수행하십시오.

```
DROP DATABASE <name>
```

다음 명령은 데이터베이스 SAMPLE을 삭제합니다.

```
DROP DATABASE SAMPLE
```

주: SAMPLE 데이터베이스의 시험을 계속하려는 경우, 이를 삭제하면 안됩니다. SAMPLE 데이터베이스를 제거한 후 이를 다시 사용하려면, 재작성해야 합니다.

관련 참조:

- *Command Reference*의『GET SNAPSHOT Command』

- *Command Reference*의 『DROP DATABASE Command』
- *Command Reference*의 『LIST ACTIVE DATABASES Command』

데이터베이스 파티션 그룹 변경

프로시저:

파티션을 추가 또는 삭제한 후에는, 데이터베이스 파티션 그룹의 새 파티션 세트에 현재 데이터를 재분배해야 합니다. 이를 수행하려면 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하십시오.

관련 개념:

- *관리 안내서: 성능*의 『데이터 재분배』
- *관리 안내서: 성능*의 『데이터베이스 서버 용량 관리』

태스크 관련:

- *관리 안내서: 성능*의 『파티션에 걸친 데이터 재분배』

관련 참조:

- *Command Reference*의 『REDISTRIBUTE DATABASE PARTITION GROUP Command』

테이블 스페이스 변경

프로시저:

데이터베이스를 작성하면, 최소한 세 개의 테이블 스페이스를 작성하십시오. 카탈로그 테이블 스페이스(SYSCATSPACE), 사용자 테이블 스페이스(디폴트 이름은 USERSPACE1) 및 하나의 시스템 임시 테이블 스페이스(디폴트 이름은 TEMPSPACE1)입니다. 적어도 이 테이블 스페이스 중 하나가 있어야 합니다. 원하는 경우, 추가 사용자와 임시 테이블 스페이스를 추가할 수 있습니다.

주: 카탈로그 테이블 스페이스 SYSCATSPACE를 삭제하거나 다른 것을 작성할 수 없으므로, 페이지 크기가 4KB인 시스템 임시 테이블 스페이스가 하나 이상 있어야 합니다. 기타 시스템 임시 테이블 스페이스를 작성할 수 있습니다. 또한 테이블 스페이스를 작성한 후에 페이지 크기 또는 테이블 스페이스의 Extent 크기를 변경할 수 없습니다.

태스크 관련:

- 183 페이지의 『DMS 테이블 스페이스에 컨테이너 추가』
- 184 페이지의 『DMS 테이블 스페이스에서 컨테이너 수정』
- 188 페이지의 『파티션의 SMS 테이블 스페이스에 컨테이너 추가』
- 189 페이지의 『테이블 스페이스 이름 바꾸기』

- 190 페이지의『사용자 테이블 스페이스 삭제』
- 191 페이지의『시스템 임시 테이블 스페이스 삭제』
- 192 페이지의『사용자 임시 테이블 스페이스 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER TABLESPACE문』

테이블 스페이스 변경 세부사항

이 섹션에서는 테이블 스페이스 변경과 연관된 태스크를 검토합니다.

DMS 테이블 스페이스에 컨테이너 추가

프로시저:

테이블 스페이스에 하나 이상의 컨테이너를 추가하여 DMS 테이블 스페이스(MANAGED BY DATABASE절로 작성된 테이블 스페이스)의 *크기*를 증가시킬 수 있습니다.

테이블 스페이스에 새 컨테이너를 추가하거나 기존 컨테이너를 확장하면 테이블 스페이스의 균형 조정이 발생할 수 있습니다. 균형 조정 프로세스에는 한 위치에서 다른 위치로의 테이블 스페이스 Extent 이동이 포함됩니다. 이 프로세스 동안, 테이블 스페이스 내에서 데이터를 stripe 상태로 유지하려고 시도합니다. 균형 조정은 모든 컨테이너에서 발생하는 것이 아니라, 기존 컨테이너 구성, 새 컨테이너의 *크기*, 테이블 스페이스가 채워진 정도 등 다수의 요소에 따라 달라집니다.

기존 테이블 스페이스에 컨테이너가 추가될 때, 컨테이너가 stripe 0에서 시작되지 않을 수도 있습니다. 맵에서 컨테이너가 시작되는 위치는 데이터베이스 관리 프로그램에 의해 결정되며 추가되는 컨테이너의 *크기*에 기초합니다. 추가되는 컨테이너가 충분히 크지 않으면, 컨테이너가 맵의 마지막 stripe에서 끝나도록 배치됩니다. 컨테이너가 충분히 크면 stripe 0에서 시작되도록 배치됩니다.

새 컨테이너를 추가하고 새 stripe 세트를 작성하면 균형 조정이 발생하지 않습니다. 새 stripe 세트는 ALTER TABLESPACE문에 BEGIN NEW STRIPE SET절을 사용하여 작성합니다. ALTER TABLESPACE문에서 ADD TO STRIPE SET절을 사용하여 기존 stripe 세트에 컨테이너를 추가할 수도 있습니다.

균형을 조정하는 동안에도 테이블 스페이스에 대한 액세스는 제한되지 않습니다. 둘 이상의 컨테이너를 추가해야 할 경우, 컨테이너를 동시에 추가해야 합니다.

제어 센터를 사용하여 DMS 테이블 스페이스에 컨테이너를 추가하려면, 다음을 수행하십시오.

1. **테이블 스페이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 컨테이너를 추가하려는 테이블 스페이스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **추가**를 누르고 정보를 완료한 후, **확인**을 누르십시오.

명령행을 사용하여 DMS 테이블 스페이스에 컨테이너를 추가하려면, 다음을 입력하십시오.

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<path>' <size>, FILE '<filename>' <size>)
```

다음 예에서는 UNIX 기반 시스템에서 테이블 스페이스에 두 개의 새 디바이스 컨테이너(각각 10 000 페이지씩)를 추가하는 방법을 설명합니다.

```
ALTER TABLESPACE RESOURCE
  ADD (DEVICE '/dev/rhd9'  10000,
       DEVICE  '/dev/rhd10' 10000)
```

ALTER TABLESPACE문을 사용하여, 성능에 영향을 줄 수 있는 테이블 스페이스의 기타 등록 정보를 변경할 수 있습니다.

관련 개념:

- *관리 안내서: 성능*의 『쿼리 최적화에 영향을 미치는 테이블 스페이스』
- *관리 안내서: 계획*의 『DMS 테이블 스페이스에서 컨테이너를 추가 및 확장하는 방법』

태스크 관련:

- 188 페이지의 『파티션의 SMS 테이블 스페이스에 컨테이너 추가』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLESPACE문』

DMS 테이블 스페이스에서 컨테이너 수정

제한사항:

각 원시 디바이스는 하나의 컨테이너로서 사용될 수 있습니다. 원시 디바이스(raw device) *크기*는 작성 후에 고정됩니다. *크기* 조정 또는 확장 옵션을 사용하여 원시 디바이스 컨테이너를 확장하고자 할 경우에는, 먼저 원시 디바이스 *크기*를 점검하여 디바이스 컨테이너 *크기*를 원시 디바이스 *크기*보다 더 *크게* 확장하지 않도록 해야 합니다.

프로시저:

DMS 테이블 스페이스(MANAGED BY DATABASE절로 작성된)에 있는 컨테이너의 *크기*를 조정할 수 있습니다.

제어 센터를 사용하여 DMS 테이블 스페이스에 있는 하나 이상의 컨테이너 크기를 증가시키려면 다음을 수행하십시오.

1. **테이블 스페이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 컨테이너를 추가하려는 테이블 스페이스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **크기 조정**을 누르고 정보를 입력한 후 **확인**을 누르십시오.

DMS 테이블 스페이스에서 기존 컨테이너를 삭제하고 DMS 테이블 스페이스에 있는 기존 컨테이너의 크기를 줄일 수도 있으며, 모든 컨테이너에 있는 데이터의 균형을 조절할 필요없이 DMS 테이블 스페이스에 새 컨테이너를 추가할 수 있습니다.

삭제하거나 크기를 줄일 Extent의 수가 테이블 스페이스의 하이 워터 마크 위에 있는 여유 Extent 수 이하일 경우에만 기존 테이블 스페이스 컨테이너를 삭제하고 크기를 줄일 수 있습니다. 최대 수위 표시는 테이블 스페이스에서 최고 할당 페이지의 페이지 번호입니다. 하이 워터 마크 아래의 일부 Extent가 재사용이 가능하게 되었을 수도 있으므로, 이 표시는 테이블 스페이스의 사용된 페이지 수와 동일하지 않습니다.

테이블 스페이스의 하이 워터 마크 위에 있는 여유 Extent 수가 중요한 이유는 하이 워터 마크까지의 모든 Extent가 테이블 스페이스 내에서 동일한 논리적 위치에 있어야 하기 때문입니다. 결과적으로 테이블 스페이스에는 모든 데이터를 포함할 수 있는 충분한 공간이 있어야 합니다. 충분한 여유 공간이 없으면 오류 메시지(SQL20170N, SQLSTATE 57059)를 초래합니다.

컨테이너를 삭제하려면 ALTER TABLESPACE문에 DROP 옵션을 사용하십시오. 예를 들어, 다음과 같습니다.

```
ALTER TABLESPACE TS1 DROP (FILE 'file1', DEVICE '/dev/rdisk1')
```

기존 컨테이너의 크기를 줄이려면 RESIZE 또는 REDUCE 옵션을 사용하십시오. RESIZE 옵션을 사용하면 명령문의 일부로서 나열한 모든 컨테이너의 크기가 증가하거나 감소됩니다. 동일한 명령문을 사용하여 일부 컨테이너의 크기를 증가시키면서 다른 컨테이너의 크기를 감소시킬 수 없습니다. 컨테이너 크기에 대한 새로운 최저 한계를 알고 있는 경우에는 크기 조정 방법을 고려해야 합니다. 컨테이너의 현재 크기를 모른다면(고려하지 않으면) 축소 방법을 고려해야 합니다.

명령행을 사용하여 DMS 테이블 스페이스에 있는 하나 이상의 컨테이너 크기를 줄이려면 다음을 입력하십시오.

```
ALTER TABLESPACE <name>
   REDUCE (FILE '<filename>' <size>)
```

다음 예는 Windows 기반 시스템에서 테이블 스페이스에 있는 파일 컨테이너(1 000페이지가 이미 있는)를 줄이는 방법을 보여줍니다.

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:\hldr\finance'  200)
```

이 조치 이후에 파일 *크기*는 1 000페이지에서 800페이지로 줄어듭니다.

명령행을 사용하여 DMS 테이블 스페이스에 있는 하나 이상의 컨테이너 *크기*를 늘리려면 다음을 입력하십시오.

```
ALTER TABLESPACE <name>
   RESIZE (DEVICE '<path>' <size>)
```

다음 예는 UNIX 기반 시스템의 테이블 스페이스에서 두 개의 디바이스 컨테이너(1,000페이지가 이미 있는)를 증가시킬 수 있는 방법을 설명합니다.

```
ALTER TABLESPACE HISTORY
  RESIZE (DEVICE '/dev/rhd7'  2000,
          DEVICE  '/dev/rhd8' 2000)
```

이 조치 다음에, 두 개의 디바이스는 1,000페이지에서 2,000페이지로 증가합니다. 테이블 스페이스의 컨텐츠 균형이 컨테이너 전반에 걸쳐 재조정될 수도 있습니다. 균형을 조정하는 동안에도 테이블 스페이스에 대한 액세스는 제한되지 않습니다.

명령행을 사용하여 DMS 테이블 스페이스에서 하나 이상의 컨테이너를 확장하려면, 다음을 입력하십시오.

```
ALTER TABLESPACE <name>
   EXTEND (FILE '<filename>' <size>)
```

다음 예는 Windows 기반 시스템에서 테이블 스페이스에 있는 파일 컨테이너(각각 이미 1 000페이지가 있는)를 늘리는 방법을 보여줍니다.

```
ALTER TABLESPACE PERSNEL
  EXTEND (FILE 'e:\wrkhist1'  200
          FILE  'f:\wrkhist2'  200)
```

이 조치 이후에 두 파일의 *크기*는 1 000페이지에서 1 200페이지로 늘어납니다. 테이블 스페이스의 컨텐츠 균형이 컨테이너 전반에 걸쳐 재조정될 수도 있습니다. 재조정하는 동안에도 테이블 스페이스에 대한 액세스는 제한되지 않습니다.

테이블 작성 동안 또는 작성 후에 추가되거나, 테이블 스페이스 작성 후에 확장된 DMS 컨테이너(파일 및 원시 디바이스 컨테이너 모두)는 프리페처를 통해 병렬로 수행됩니다. 이러한 컨테이너 작성 또는 *크기* 조정 조작의 병렬 처리에서 증가를 하려면, 시스템에서 수행 중인 프리페처의 수를 증가시킵니다. 병렬에서 완료되지 않는 단 하나의 프로세스는 이 조치의 로깅과 컨테이너 작성의 경우, 컨테이너의 태그 처리입니다.

주: CREATE TABLESPACE 또는 ALTER TABLESPACE문(기존 테이블 스페이스에 새 컨테이너 추가에 관하여)의 병렬 처리를 최대화하려면, 프리페처의 수가 추가된 컨테이너의 수보다 더 많은지 또는 같은지 확인하십시오. 프리페처 수는 *num_ioservers* 데이터베이스 구성 매개변수로 제어됩니다. 새 매개변수 값을 적용

하려면 데이터베이스를 중지해야 합니다. 즉, 변경사항을 적용하려면 데이터베이스로의 모든 응용프로그램 및 사용자 연결을 끊어야 합니다.

ALTER TABLESPACE문을 사용하여, 성능에 영향을 줄 수 있는 테이블 스페이스의 기타 등록 정보를 변경할 수 있습니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLESPACE문』

컨테이너 추가 또는 삭제 후 자동 프리페치 크기 조정

컨테이너 추가 또는 삭제 후 테이블 스페이스의 프리페치 *크기* 갱신을 잊어버릴 가능성이 있을 경우, 데이터베이스 관리 프로그램이 프리페치 *크기*를 자동으로 결정하게 할 수 있습니다. 프리페치 *크기* 갱신을 잊어버린 경우, 데이터베이스의 성능이 상당히 떨어질 수도 있습니다.

DB2® Universal Database(DB2 UDB)는 버전 8.2(이상)를 사용하여 작성된 모든 테이블 스페이스의 디폴트값으로 설정됩니다. 데이터베이스 관리 프로그램은 다음과 같은 수식을 사용하여 테이블 스페이스의 프리페치 *크기*를 계산합니다.

```
prefetch size = (number of containers) X (number of physical spindles per
                container) X extent size
```

테이블 스페이스의 프리페치 *크기*를 AUTOMATIC으로 설정하지 않는 다음과 같은 세 가지 방법이 있습니다.

- 특정 프리페치 *크기*의 테이블 스페이스를 작성하십시오. 프리페치 *크기*에 대한 값을 수동으로 선택하면 테이블 스페이스와 연관된 컨테이너 수가 조정될 때마다 필요한 경우 프리페치 *크기*를 조정해야 합니다.
- 테이블 스페이스 작성시 프리페치 *크기*를 사용하지 않고 *dft_prefetch_sz* 데이터베이스 구성 매개변수를 AUTOMATIC이 아닌 값으로 설정하십시오. DB2 UDB는 테이블 스페이스 작성시 프리페치 *크기*에 대한 명시적 언급이 없을 경우 이 매개변수를 점검합니다. AUTOMATIC 이외의 값이 존재할 경우, 해당하는 값이 디폴트 프리페치 *크기*로 사용됩니다. 또한 테이블 스페이스와 연관된 컨테이너의 수가 조정될 때마다 필요한 경우 프리페치 *크기*를 조정해야 합니다.
- ALTER TABLESPACE문을 사용하여 프리페치 *크기*를 수동으로 변경하십시오.

DB2_PARALLEL_IO 사용

프리페치 요청은 테이블 스페이스의 병렬 처리에 따라, 요청이 프리페치 대기열에 제출되기 전에 몇 개의 작은 프리페치 요청으로 분할됩니다. DB2_PARALLEL_IO 레지스트리 변수는 컨테이너 당 물리적 스핀들의 수를 정의하는 데 사용되며 테이블 스페이스에 관한 병렬 I/O에 영향을 미칩니다. 병렬 I/O가 해제된 경우, 테이블 스페이스의 병렬 처리는 컨테이너 수와 같습니다. 병렬 I/O가 설정된 경우, 테이블 스페이스의 병렬 처리는 컨테이너 수를 DB2_PARALLEL_IO 레지스트리 변수에 제공된 값으로 곱한

것과 같습니다. (다르게 말하면, 테이블 스페이스의 병렬 처리는 프리페치 *크기*를 테이블 스페이스의 Extent *크기*로 나눈 것과 같습니다.)

DB2_PARALLEL_IO 레지스트리 변수가 프리페치 *크기*에 영향을 미치는 방법에 대한 몇 가지 예는 다음과 같습니다. (다음 테이블 스페이스 모두가 AUTOMATIC 프리페치 *크기*로 정의되었다고 가정하십시오.)

- DB2_PARALLEL_IO=*
  - 모든 테이블 스페이스가 디폴트값을 사용하며, 여기서 스핀들의 수는 각 컨테이너에 대하여 6입니다. 프리페치 *크기*는 병렬 I/O 설정 시 6배 더 큽니다.
  - 모든 테이블 스페이스에 병렬 I/O가 설정됩니다. 프리페치 요청이 몇 개의 작은 요청으로 분할되며, 각각은 프리페치 *크기*를 Extent *크기*로 나눈 값(또는 컨테이너 수 X 스핀들 수)과 같습니다.
- DB2_PARALLEL_IO=*:3
  - 모든 테이블 스페이스가 컨테이너 당 스핀들의 수로 3을 사용합니다.
  - 모든 테이블 스페이스에 병렬 I/O가 설정됩니다.
- DB2_PARALLEL_IO=*:3,1:1
  - 모든 테이블 스페이스가 컨테이너 당 스핀들의 수로 3을 사용합니다(1을 사용하는 테이블 스페이스 1 제외).
  - 모든 테이블 스페이스에 병렬 I/O가 설정됩니다.

태스크 관련:

- 182 페이지의 『테이블 스페이스 변경』
- 183 페이지의 『DMS 테이블 스페이스에 컨테이너 추가』
- 184 페이지의 『DMS 테이블 스페이스에서 컨테이너 수정』

파티션의 SMS 테이블 스페이스에 컨테이너 추가

제한사항:

현재 컨테이너가 없는 임의 파티션 상의 SMS 테이블 스페이스에 컨테이너를 추가할 수 있습니다.

프로시저:

명령행을 사용하여 SMS 테이블 스페이스에 컨테이너를 추가하려면, 다음을 입력하십시오.

```
ALTER TABLESPACE <name>
   ADD ('<path>')
   ON DBPARTITIONNUM (<partition_number>)
```

번호를 지정한 파티션 및 해당 범위의 파티션에 속하는 모든 파티션(또는 노드)이 해당 테이블 스페이스가 정의된 데이터베이스 파티션 그룹에 있어야 합니다. partition_number는 명시적이거나 명령문에 대해 정확히 하나의 *db-partitions-clause* 범위 내에서만 나타납니다.

다음 예에서는 테이블 스페이스 『plans』가 UNIX 기반 운영 체제에서 사용한 데이터베이스 파티션 그룹의 파티션 번호 3에 새 컨테이너를 추가하는 방법을 표시합니다.

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

태스크 관련:

- 183 페이지의 『DMS 테이블 스페이스에 컨테이너 추가』
- 184 페이지의 『DMS 테이블 스페이스에서 컨테이너 수정』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLESPACE문』

테이블 스페이스 이름 바꾸기

제한사항:

SYSCATSPACE 테이블 스페이스의 이름을 바꿀 수 없습니다.

『롤 포워드 보류』 또는 『롤 포워드 진행』 상태인 테이블 스페이스의 이름을 바꿀 수 없습니다.

백업된 이후에 이름이 바뀐 테이블 스페이스를 리스토어할 때, RESTORE DATABASE 명령에서 새로운 테이블 스페이스 이름을 사용하십시오. 이전 테이블 스페이스 이름을 사용하는 경우, 찾지 못합니다. 마찬가지로, ROLLFORWARD DATABASE 명령으로 테이블 스페이스를 롤 포워드하는 경우, 새로운 이름을 사용하십시오. 이전 테이블 스페이스 이름을 사용하는 경우, 찾지 못합니다.

프로시저:

테이블 스페이스 내에서 개별 오브젝트와 관련되지 않고 새로운 이름을 기존 테이블 스페이스에 부여할 수 있습니다. 테이블 스페이스의 이름을 바꿀 때, 해당 테이블 스페이스를 참조하는 모든 카탈로그 레코드가 변경됩니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『RENAME TABLESPACE문』

테이블 스페이스 상태 전환

프로시저:

테이블 스페이스에 연관된 컨테이너가 액세스 가능한 경우, ALTER TABLESPACE 문의 SWITCH ONLINE절은 해당 테이블 스페이스에서 OFFLINE 상태를 제거하는 데 사용합니다. 테이블 스페이스는 제거된 OFFLINE 상태를 가지고 있는 반면, 나머지 데이터베이스는 여전히 위에 있고 사용되고 있습니다.

이 절의 사용에 대한 대체는 데이터베이스로부터 모든 응용프로그램을 연결해제했다가, 데이터베이스에 응용프로그램을 다시 연결하는 것입니다. 이것은 테이블 스페이스에서 OFFLINE 상태를 제거해 줍니다.

명령행을 사용하여 테이블 스페이스에서 OFFLINE 상태를 제거하려면, 다음을 입력하십시오.

```
db2 ALTER TABLESPACE <name>
   SWITCH ONLINE
```

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLESPACE문』

사용자 테이블 스페이스 삭제

프로시저:

사용자 테이블 스페이스를 제거하는 경우, 해당 테이블 스페이스의 모든 데이터를 삭제하고, 컨테이너를 해제하고, 카탈로그 항목을 제거하십시오. 그러면 테이블 스페이스의 모든 정의된 오브젝트는 제거되거나 올바르지 않은 것으로 표시됩니다.

테이블 스페이스를 제거하여 빈 테이블 스페이스에서 컨테이너를 재사용할 수 있지만, 컨테이너를 재사용하기 전에 DROP TABLESPACE 명령을 COMMIT해야 합니다.

단일 사용자 테이블 스페이스에서 인덱스 및 LOB 데이터를 비롯한 모든 테이블 데이터가 들어 있는 사용자 테이블 스페이스를 제거할 수 있습니다. 여러 개의 테이블 스페이스에 스팬되어 있는 사용자 테이블 스페이스를 제거할 수도 있습니다. 즉, 테이블 데이터를 테이블 스페이스 하나에 보관하고 인덱스는 또다른 테이블 스페이스에 보관하여 모든 LOB를 세 번째 테이블 스페이스에 보관할 수 있습니다. 단일 명령문에서는 세 가지 테이블 스페이스를 동시에 모두 제거해야 합니다. 테이블이 들어 있는 모든 스팬된 테이블 스페이스는 이 단일 명령문의 일부가 되지 않으면, 제거 요청이 실패합니다.

제어 센터를 사용하여 사용자 테이블 스페이스를 제거하려면, 다음을 수행하십시오.

1. **테이블 스페이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 제거하려는 테이블 스페이스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 사용자 테이블을 제거하려면, 다음을 입력하십시오.

```
DROP TABLESPACE <name>
```

다음 SQL문은 ACCOUNTING 테이블 스페이스를 제거합니다.

```
DROP TABLESPACE ACCOUNTING
```

태스크 관련:

- 191 페이지의『시스템 임시 테이블 스페이스 삭제』
- 192 페이지의『사용자 임시 테이블 스페이스 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의『COMMIT문』
- *SQL 참조서, 볼륨 2*의『DROP문』

시스템 임시 테이블 스페이스 삭제

제한사항:

먼저 다른 시스템 임시 테이블 스페이스를 작성하지 않고는 페이지 크기가 4KB인 시스템 임시 테이블 스페이스를 삭제할 수 없습니다. 데이터베이스에는 항상 페이지 크기가 4KB인 시스템 임시 테이블 스페이스가 하나 이상 있어야 하므로, 새 시스템 임시 테이블 스페이스의 페이지 크기는 4KB이어야 합니다. 예를 들어, 페이지 크기가 4KB인 단일 시스템 임시 테이블 스페이스가 있고 여기에 컨테이너를 추가하고자 하며 그것이 SMS 테이블 스페이스인 경우, 먼저 적절한 수의 컨테이너가 있는 4KB 페이지 크기의 새 시스템 임시 테이블 스페이스를 추가한 다음 기존 시스템 임시 테이블 스페이스를 삭제하십시오(DMS를 사용할 경우에는 테이블 스페이스를 삭제 및 재작성하지 않고 컨테이너를 추가할 수 있습니다).

프로시저:

디폴트 테이블 스페이스 크기는 4KB입니다.

제어 센터를 사용하여 시스템 테이블 스페이스를 제거하려면, 다음을 수행하십시오.

1. **테이블 스페이스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 기타 시스템 임시 테이블 스페이스에서 **테이블 스페이스** 아이콘을 마우스의 오른쪽 단추로 선택한 후, **작성 —> 마법사를 사용한 테이블 스페이스**를 선택하십시오. 그렇지 않으면 4단계로 건너뛰십시오.
3. 필요한 경우 새로운 시스템 임시 테이블 스페이스를 작성하려면 마법사에 있는 단계에 따르십시오.
4. 창의 오른쪽에서(내용 분할창) 테이블 스페이스의 목록을 표시하려면 **테이블 스페이스** 폴더를 다시 누르십시오.
5. 제거하려는 시스템 임시 테이블 스페이스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 누르십시오.
6. **확인** 상자를 누른 후, **확인**을 누르십시오.

다음은 시스템 임시 테이블 스페이스를 작성하는 명령문입니다.

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
   MANAGED BY SYSTEM USING ('<directories>')
```

명령행을 사용하여 시스템 테이블 스페이스를 제거하려면, 다음을 입력하십시오.

```
DROP TABLESPACE <name>
```

다음 SQL문은 TEMPSPACE2라고 하는 새 시스템 임시 테이블 스페이스를 작성합니다.

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
   MANAGED BY SYSTEM USING ('d:\systemp2')
```

일단 TEMPSPACE2가 작성되면, 원래의 시스템 임시 테이블 스페이스인 TEMPSPACE1을 다음 명령으로 제거할 수 있습니다.

```
DROP TABLESPACE TEMPSPACE1
```

테이블 스페이스를 제거하여 빈 테이블 스페이스에서 컨테이너를 재사용할 수 있지만, 컨테이너를 재사용하기 전에 DROP TABLESPACE 명령을 COMMIT해야 합니다.

태스크 관련:

- 190 페이지의『사용자 테이블 스페이스 삭제』
- 192 페이지의『사용자 임시 테이블 스페이스 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TABLESPACE문』
- *SQL 참조서, 볼륨 2*의『DROP문』

사용자 임시 테이블 스페이스 삭제

프로시저:

삭제하고자 하는 사용자 임시 테이블 스페이스에 현재 정의되어 있는 선언된 임시 테이블이 없는 경우에만 사용자 임시 테이블 스페이스를 삭제할 수 있습니다. 테이블 스페이스를 제거할 때, 테이블 스페이스에 있는 모든 선언된 임시 테이블을 제거하려는 시도는 선언되지 않습니다.

주: 선언된 임시 테이블은 이를 선언한 응용프로그램이 데이터베이스에서 연결해제되면 내재적으로 제거됩니다.

태스크 관련:

- 190 페이지의『사용자 테이블 스페이스 삭제』
- 191 페이지의『시스템 임시 테이블 스페이스 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DROP문』

스키마 삭제

프로시저:

스키마를 제거하기 전에, 해당 스키마에 있는 모든 오브젝트가 제거되거나 또다른 스키마로 이동되어야 합니다. DROP문을 시도할 때에는 스키마 이름이 카탈로그에 있어야 하고, 그렇지 않으면 오류가 리턴됩니다.

제어 센터를 사용하여 스키마를 제거하려면, 다음을 수행하십시오.

1. **스키마** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 제거하려는 스키마를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 스키마를 제거하려면, 다음을 입력하십시오.

```
DROP SCHEMA <name>
```

다음 예에서는 스키마 "joeschma"가 제거됩니다.

```
DROP SCHEMA joeschma RESTRICT
```

RESTRICT 키워드는 데이터베이스에서 삭제되는 스키마의 경우 어떠한 오브젝트도 지정된 스키마에 정의될 수 없다는 규칙을 시행합니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DROP문』

버퍼 풀 변경

기존 버퍼 풀에서 작업 시 다음 태스크 중 하나를 완료해야 합니다.

- 모든 파티션 또는 단일 파티션의 버퍼 풀 크기를 수정합니다.
- 확장 스토리지 사용을 사용 가능하게 하거나 사용 불가능하게 합니다.
- 해당 버퍼 풀 정의를 새 데이터베이스 파티션 그룹에 추가합니다.
- 블록 기반 I/O에 사용되는 버퍼 풀의 블록화 영역을 수정합니다.

전제조건:

명령문의 권한 부여 ID에는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

프로시저:

1. SELECT BPNAME FROM SYSCAT.BUFFERPOOLS로 이미 데이터베이스에 있는 버퍼 풀 이름 목록을 가져오십시오.
2. 결과 목록에서 버퍼 풀 이름을 선택하십시오.
3. 변경할 내용을 결정하십시오.
4. ALTER BUFFERPOOL문 실행에 필요한 올바른 권한 부여 ID가 있는지 확인하십시오.

   주: 두 개의 키 매개변수는 IMMEDIATE 및 DEFERRED입니다. IMMEDIATE를 사용하면 버퍼 풀 *크기*를 바로 변경할 수 있습니다. 데이터베이스 공유 메모리에 있는 예약 스페이스가 새 스페이스를 할당하기에 충분하지 않은 경우 해당 명령문을 바로 실행할 수 없습니다.

   DEFERRED를 사용하면, 해당 응용프로그램을 연결 해제한 후 데이터베이스를 재활성화하여 버퍼 풀을 캐시할 수 있습니다. 예약 메모리 스페이스는 필요 없습니다. 활성화시 DB2 UDB는 시스템으로부터 필수 메모리를 할당합니다.
5. ALTER BUFFERPOOL문을 사용하여 단일 버퍼 풀 오브젝트를 변경합니다.

태스크 관련:

- 79 페이지의 『버퍼 풀 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER BUFFERPOOL문』

제 6 장 테이블 및 기타 관련 테이블 오브젝트 변경

테이블 및 관련 테이블 오브젝트의 구조 및 내용을 수정하는 데 필요한 태스크에는 다음이 포함됩니다.

- 『기존 테이블의 스페이스 압축』
- 199 페이지의 『기존 테이블에 컬럼 추가』
- 200 페이지의 『컬럼 정의 수정』
- 201 페이지의 『테이블 또는 뷰에서 행 제거』
- 221 페이지의 『MERGE문을 사용하여 테이블 및 뷰 내용 갱신』
- 203 페이지의 『식별 컬럼 정의 수정』
- 203 페이지의 『제한조건 변경』
- 210 페이지의 『기존 테이블에 생성된 컬럼 정의』
- 214 페이지의 『테이블을 volatile로 선언』
- 214 페이지의 『파티션 키 변경』
- 215 페이지의 『테이블 속성 변경』
- 218 페이지의 『구체화된 쿼리 테이블 등록 정보 변경』
- 219 페이지의 『구체화된 쿼리 테이블의 데이터 새로 고침』

테이블용 트리거를 변경할 수 없습니다. 더 이상 적합하지 않은 모든 트리거를 삭제(drop)하고(225 페이지의 『트리거 삭제』 참조) 대체 트리거를 추가해야 합니다(131 페이지의 『트리거 작성』 참조).

기존 테이블 및 기타 관련 테이블 오브젝트 수정

기존 테이블의 스페이스 압축

기존 테이블을 스페이스 압축이 가능한 레코드 형식으로 변경할 수 있습니다. 바이트 수 합이 테이블 스페이스의 허용 가능한 테이블 행 길이를 초과하지 않는 한, 스페이스 압축이 가능한 레코드 형식의 컬럼 바이트 수 합이 스페이스 압축이 불가능한 원래 레코드 형식의 컬럼 바이트 수 합을 초과해도 됩니다. 예를 들어, 페이지 *크기*가 4KB인 테이블 스페이스에서 허용 가능한 행 길이는 4005바이트입니다. 이 값을 초과하면 오류 메시지 SQL0670N이 리턴됩니다. 바이트 수 공식은 CREATE TABLE문의 일부로서 문서화됩니다.

마찬가지로, 기존 테이블을 스페이스 압축이 가능한 레코드 형식에서 스페이스 압축이 불가능한 레코드 형식으로 변경할 수 있습니다. 컬럼 바이트 수 합과 관련하여 동일한 조건이 적용되며 필요한 경우 오류 메시지 SQL0670N이 리턴됩니다.

테이블에 대해 스페이스 압축을 고려해야 하는지의 여부를 판별하려면, 대다수의 값이 시스템 디폴트값 또는 NULL과 같은 테이블이 새로운 행 형식으로부터 이득을 얻을 수 있는지 알아야 합니다. 예를 들어, INTEGER 컬럼이 있고 이 컬럼의 90%가 0(INTEGER 데이터 유형에 대한 디폴트값) 또는 널(NULL) 값을 가질 경우, 이 테이블과 함께 이 컬럼을 압축하면 새로운 행 형식으로부터 이득을 얻을 수 있으며 많은 양의 디스크 스페이스를 절약할 수 있습니다.

테이블을 변경할 때, VALUE COMPRESSION절을 사용하여 테이블이 테이블 레벨 및 컬럼 레벨에서 스페이스 행 형식을 사용하도록 지정할 수 있습니다. ACTIVATE VALUE COMPRESSION을 사용하여 테이블이 스페이스 절약 기술을 사용하도록 지정하거나, DEACTIVATE VALUE COMPRESSION을 사용하여 테이블이 테이블 데이터에 대해 더이상 스페이스 절약 기술을 사용하지 않도록 지정할 수 있습니다.

DEACTIVATE VALUE COMPRESSION을 사용하면, 해당 테이블에 있는 컬럼과 연관된 모든 COMPRESS SYSTEM DEFAULT 옵션이 내재적으로 사용 불가능화됩니다.

테이블을 새 행 형식으로 수정하면, 삽입, 로드 또는 갱신되는 모든 후속 행이 새 행 형식을 가집니다. 모든 행을 새 행 형식으로 수정하려면, 테이블 재구성을 실행하거나 행 형식을 변경하기 전에 기존 행에 대해 갱신 조작을 수행해야 합니다.

관련 개념:

- 106 페이지의 『새 테이블의 스페이스 압축』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

스토어드 프로시저를 사용한 테이블 변경

테이블은 모든 비즈니스 데이터가 저장되는 곳입니다. 데이터베이스를 작성하기 전에, 데이터베이스에 보존하려는 데이터의 유형 및 구성을 고려해야 합니다. 사용자와 사용자의 비즈니스에서 필요로 하는 모든 관련 데이터를 사용하고 조작하려면 많은 계획이 필요합니다. 그러나 상황은 변화합니다. 좋은 계획에도 불구하고, 데이터베이스의 테이블을 변경해야 하는 새로운 요구사항 또는 비즈니스 변화가 발생할 수도 있습니다.

다음 중 하나 이상의 방식으로 테이블을 변경해야 하는 상황이 있을 수 있습니다.

- 컬럼 이름 바꾸기

- 컬럼 제거
- SQL 스칼라 함수를 사용한 컬럼 유형 변경 및 기존 데이터 변형
- 컬럼 크기 증가 또는 감소
- 컬럼 디폴트값 변경
- NOT NULL에서 NULLABLE로 컬럼 변경
- 10진수의 정밀도 및 스케일 변경

이들 유형을 변경할 경우, 원래 테이블 데이터의 손실 위험을 최소화해야 합니다. DB2® Universal Database(DB2 UDB)는 사용자가 테이블을 변경할 수 있는 사용자 인터페이스 및 스토어드 프로시저를 제공합니다. 원하는 모든 테이블 변경 작업이 완료되었음을 명시적으로 표시할 때까지 원래 테이블 및 관련 데이터가 삭제되지 않습니다.

사용자 인터페이스에서 호출된 각각의 스토어드 프로시저는 데이터 삭제, 재작성 및 로드와 같은 일련의 조치를 수행하여 위에 나열된 조치를 완료합니다.

테이블에서 변경할 수 있는 항목에 관하여 한계가 있습니다. 한계는 다음과 같습니다.

- 구체화된 쿼리 테이블(MQT) 변경이 지원되지 않습니다.

  그러나 MQT를 포함하는 테이블 변경은 지원됩니다. 또한 MQT는 변경된 기본 테이블에 정의된 MQT를 ALTER TABLE 프로세스 수행 시 새로 고치지(채우지) 않습니다. MQT에서 기본 테이블이 ALTOBJ() 스토어드 프로시저에 의해 변경되는 동안, MQT 컨텐츠가 새 기본 테이블에서 완벽하게 다시 빌드되므로 기본 테이블로부터의 선택 결과의 일부가 아닌 모든 컬럼이 손실됩니다.
- 유형 테이블 또는 기존 참조 컬럼 유형 테이블의 범위인 테이블 변경이 지원되지 않습니다.
- 별칭을 사용한 원격 테이블 변경이 지원되지 않습니다.
- 테이블 내의 컬럼 시퀀스를 재정렬할 수 없습니다.
- 추가 및 이름 바꾸기가 컬럼 삭제 조치에 배타적입니다.

  즉, 해당 컬럼 조치가 하나의 단일 테이블 변경 호출에 공존할 수 없습니다.
- DATALINK 데이터 유형이 지원되지 않습니다.
- 지속적인 오브젝트 잠금이 없기 때문에 ALTOBJ() 호출 간에 오브젝트 정의가 바뀔 수도 있습니다.
- 테이블 팩 디스크립터와 연관된 Runstats 프로파일과 같은, 테이블 프로파일은 ALTER TABLE 프로세스 실행 후 손실됩니다.
- 주어진 시간에 테이블 당 하나의 ALTER TABLE 스토어드 프로시저 호출 시퀀스만이 지원됩니다. 즉, ALTOBJ() 스토어드 프로시저를 호출하였으면, 다른 ALTER

TABLE을 동일한 테이블에서 시작하기 전에 완료하거나 롤백해야 합니다. 테이블 종속성이 충돌하지 않는 한 ALTOBJ() 스토어드 프로시저를 사용하여 동시에 다중 테이블을 지정할 수 있습니다.

ALTER TABLE 조치를 수행하는 스토어드 프로시저를 사용할 때 사용 가능한 옵션을 구성하는 몇 가지 구성요소가 있습니다. 이들 구성요소는 다음과 같습니다.

- ALTER_OBJ('GENERATE','<sql statement>, 0, ?)

  이 프로시저는 모든 SQL문을 생성하여 메타데이터 테이블에 배치합니다.

  주: 생성 모드에서 SQL문 매개변수는 널(null)일 수 없으며, 변경 ID를 제공할 경우 무시됩니다.

- ALTER_OBJ('VALIDATE',NULL,123,?)

  이 프로시저는 생성된 SQL을 검증하지만 데이터의 이동을 포함하지 않습니다. 유효성을 테스트하기 위한 스크립트가 제공된 사용자 ID『123』으로 실행됩니다. 검증 결과가 메타 데이블에 저장됩니다(변경 중인 테이블의 기타 정보도 보유함).

- ALTER_OBJ('APPLY_CONTINUE_ON_ERROR',NULL,123,?)

  이 프로시저는 제공된 ID로 모든 SQL문을 실행하며, 결과를 메타 테이블에 기록합니다. SQL문은 새 테이블 빌드 방법, 종속 오브젝트 빌드 및 새 테이블 채우기를 포함합니다.

  UNDO 모드를 사용하여 기존 정의를 다시 확보할 수 있습니다(아래 참조).

  SQLCA의 스토어드 프로시저에 대해 SQLCODE 경고가 설정되며, 스토어드 프로시저의 트랜잭션이 완료됩니다.

- ALTER_OBJ('APPLY_STOP_ON_ERROR',NULL,123,?)

  이 프로시저는 제공된 ID로 각각의 SQL문을 하나씩 실행하며, 오류 발생 시 중지됩니다.

  SQLCA의 스토어드 프로시저에 대해 SQLCODE 오류가 설정되며, 스토어드 프로시저의 트랜잭션이 자동으로 롤백됩니다.

- ALTER_OBJ('UNDO',NULL,123,?)

  제공된 사용자 ID로 테이블 변경 조치에 의해 수행된 모든 변경사항을 포함하는 스크립트를 실행합니다. 해당 변경사항이 모두 미완료됩니다.

  주: ALTOBJ_UNDO에 대해 작업할 때, ID 매개변수는 널(null)일 수 없습니다.

- ALTER_OBJ('FINISH',NULL,123,?)

이 프로시저는 제공된 사용자 ID로 원래 테이블을 삭제하고, 메타 테이블에 있는 모든 항목을 정리합니다.

주: 이 모드는 기타 모든 모드와 별도로만 호출할 수 있습니다.

관련 참조:

- *SQL 참조서, 볼륨 1*의 『지원 함수 및 관리 루틴』
- *SQL Administrative Routines*에서 『ALTOBJ 프로시저』

기존 테이블에 컬럼 추가

프로시저:

컬럼 정의에는 컬럼 이름, 데이터 유형 및 필요한 제한조건이 포함됩니다.

테이블에 컬럼을 추가할 때 컬럼은 논리적으로 가장 오른쪽에 존재하는 컬럼 정의의 오른쪽에 위치합니다. 새 컬럼이 기존 테이블에 추가될 때에는, 시스템 카탈로그의 테이블 설명만이 수정되므로, 테이블로의 액세스 시간에 즉각 영향을 주지 않습니다. 기존의 레코드는 UPDATE문을 사용하여 수정될 때까지 실제로 변경되지 않았습니다. 테이블에서 기존의 행을 검색할 때, 새로운 컬럼이 정의된 방법에 따라 널(NULL) 또는 디폴트값이 새로운 컬럼에 제공됩니다. 테이블이 작성된 후 추가되는 컬럼은 NOT NULL로 정의될 수 없습니다. 이는 NOT NULL WITH DEFAULT 또는 널(NULL) 입력 가능으로 정의되어야 합니다.

제어 센터를 사용하여 기존 테이블에 컬럼을 추가하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 컬럼을 추가하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **컬럼** 페이지를 검사하고 컬럼에 대한 정보를 완료한 후, **확인**을 누르십시오.

명령행을 사용하여 기존 테이블에 컬럼을 추가하려면, 다음을 입력하십시오.

```
ALTER TABLE <table_name>
   ADD <column_name> <data_type> <null_attribute>
```

컬럼은 SQL문으로 추가될 수 있습니다. 다음 명령문은 ALTER TABLE문을 사용하여 세 개의 컬럼을 EMPLOYEE 테이블에 추가합니다.

```
ALTER TABLE EMPLOYEE
   ADD MIDINIT CHAR(1)   NOT NULL WITH DEFAULT
   ADD HIREDATE DATE
   ADD WORKDEPT CHAR(3)
```

태스크 관련:

- 200 페이지의 『컬럼 정의 수정』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

컬럼 정의 수정

프로시저:

기존 VARCHAR 또는 VARGRAPHIC 컬럼의 길이를 증가시켜서 컬럼의 특성을 수정할 수 있습니다. 문자 수는 사용되는 페이지 크기에 맞는 값까지 증가될 수 있습니다.

컬럼과 연관된 디폴트값을 수정할 수 있습니다. 새 디폴트값을 정의하면, 디폴트값을 사용하도록 표시된 모든 후속 SQL 조작에 있는 컬럼에 새 값이 사용됩니다. 새 값은 지정에 대한 규칙을 따라야 하며 CREATE TABLE문에서 설명한 것과 동일한 제한사항을 갖습니다.

주: 컬럼 생성 시 해당 명령문에서 변경한 디폴트값이 사용되지 않습니다.

제어 센터를 사용하여 기존 테이블의 길이를 수정하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 오른쪽 분할창에 있는 테이블 목록에서 컬럼을 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **컬럼** 페이지를 검사하고 컬럼을 선택한 후, **변경**을 누르십시오.
4. 컬럼에 새 바이트 수를 **길이**에 입력한 후, **확인**을 누르십시오.

명령행을 사용하여 기존 테이블의 컬럼 길이 및 유형을 수정하려면, 다음을 입력하십시오.

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   <modification_type>
```

예를 들어, 컬럼을 최대 4000자까지 증가시키려면, 다음과 유사한 것을 사용하십시오.

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DATA TYPE VARCHAR(4000)
```

다른 예에서 새 VARGRAPHIC 값을 가진 컬럼을 사용할 수 있도록 다음과 유사한 SQL문을 사용하십시오.

```
ALTER TABLE t1
   ALTER COLUMN colnam2
   SET DATA TYPE VARGRAPHIC(2000)
```

입력된 테이블의 컬럼은 변경할 수 없습니다. 그러나 아직 범위가 정의되지 않은 기존 참조 유형 컬럼에 범위를 추가할 수는 있습니다. 예를 들어, 다음과 같습니다.

```
ALTER TABLE t1
   ALTER COLUMN colnamt1
   ADD SCOPE typtab1
```

명령행을 사용하여 기존 테이블의 컬럼 디폴트값을 수정하려면, 다음을 입력하십시오.

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   SET DEFAULT 'new_default_value'
```

예를 들어, 컬럼의 디폴트값을 변경하려면, 다음과 유사하게 사용하십시오.

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DEFAULT '123'
```

태스크 관련:

- 203 페이지의『식별 컬럼 정의 수정』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER TABLE문』

테이블 또는 뷰에서 행 제거

프로시저:

행을 삭제하여 테이블 또는 뷰의 내용을 변경하면 됩니다. 뷰에서 행을 삭제하면 뷰의 기초가 되는 테이블에서 해당 행이 삭제됩니다. DELETE문은 다음과 같이 사용됩니다.

- 검색 조건에 따라 선택적으로 판별되는 한 행 이상의 삭제. 이는 *검색된 DELETE*로 불립니다.
- 커서의 현재 위치에 따라 결정되는 정확한 한 행 삭제. 이는 *위치 DELETE*로 불립니다.

DELETE문은 응용프로그램에 임베드되거나 동적 SQL문으로 발행됩니다.

수정되고 있는 테이블이 참조 제한조건을 통해 기타의 테이블과 관련되어 있다면, 행 삭제의 수행과 함께 고려사항이 있습니다. 식별된 테이블 또는 식별된 뷰의 기본 테이블이 상위인 경우, 삭제하려고 선택된 행은 삭제 규칙인 RESTRICT와의 관계에서 어떠한 종속도 가져서는 안됩니다. 더욱이, DELETE는 삭제 규칙인 RESTRICT와의 관계에서 종속을 가지고 있는 하위 행에 연쇄해서도 안됩니다.

삭제 조작이 RESTRICT 삭제 규칙으로 금지되지 않으면, 선택된 행이 삭제됩니다.

예를 들어, 테이블(DEPARTMENT)에서 부서(DEPTNO)인『D11』를 삭제하려면, 다음을 사용하십시오.

```
DELETE FROM department WHERE deptno='D11'
```

다중 행의 DELETE 수행 중에 오류가 발생하면, 테이블에는 아무런 변경도 일어나지 않습니다. 기존의 참조 제한조건에 의해 요청되어 검색 조건 및 모든 조작에 일치하는 모든 행 삭제를 미리 방지하는 오류가 발생하면, 테이블은 어떠한 변경도 되지 않습니다.

적절한 잠금이 이미 존재하지 않으면, 성공적인 DELETE문의 수행 중에 하나 이상의 독점 잠금이 획득됩니다. 잠금은 COMMIT 또는 ROLLBACK문 다음에 해제됩니다. 잠금은 기타의 응용프로그램이 테이블에서 조작을 수행하지 못하도록 합니다.

관련 개념:

- *관리 안내서: 성능*의 『잠금 및 동시처리 제어』
- *관리 안내서: 성능*의 『잠금 및 성능』
- *관리 안내서: 성능*의 『잠금에 영향을 미치는 인수』
- *관리 안내서: 성능*의 『잠금에 대한 지침』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DELETE문』

컬럼의 생성 또는 식별 등록 정보 수정

ALTER TABLE문에서 ALTER COLUMN절을 사용하여 테이블의 컬럼에 대한 생성 또는 식별 등록 정보를 추가하고 삭제할 수 있습니다.

다음 조치 중 하나를 수행할 수 있습니다.

- 기존 비생성 컬럼에 대해 작업할 때, 생성된 표현식 속성을 추가할 수 있습니다. 수정된 컬럼이 생성된 컬럼이 됩니다.
- 기존 생성된 컬럼에 대해 작업할 때, 생성된 표현식 속성을 삭제할 수 있습니다. 수정된 컬럼이 보통 비생성 컬럼이 됩니다.
- 기존 비식별 컬럼에 대해 작업할 때, 식별 속성을 추가할 수 있습니다. 수정된 컬럼이 식별 컬럼이 됩니다.
- 기존 식별 컬럼에 대해 작업할 때, 식별 속성을 삭제할 수 있습니다. 수정된 컬럼이 보통 비생성 비식별 컬럼이 됩니다.
- 기존 생성된 컬럼에 대해 작업할 때, 생성된 컬럼을 GENERATED ALWAYS에서 GENERATED BY DEFAULT로 변경할 수 있습니다. 역 또한 참입니다. 즉, 생성된 컬럼을 GENERATED BY DEFAULT에서 GENERATED ALWAYS로 변경할 수 있습니다. 이는 생성된 컬럼에 대해 작업할 때만 가능합니다.
- 사용자 정의 디폴트 컬럼에서 디폴트 속성을 삭제할 수 있습니다. 이를 수행할 때, 새 디폴트값은 널(null)입니다.
- 디폴트, 식별 또는 생성 속성을 삭제한 후 동일한 ALTER COLUMN문에서 새 디폴트, 식별 또는 생성 속성을 설정할 수 있습니다.

- CREATE TABLE 및 ALTER TABLE문 둘다에 대해 『ALWAYS』는 GENERATED절의 선택적 단어입니다. 이는 ALTER TABLE에서 사용될 때 GENERATED ALWAYS가 GENERATED와 같음을 의미합니다.

태스크 관련:

- 115 페이지의 『새 테이블에 생성된 컬럼 정의』
- 118 페이지의 『새 테이블에 식별 컬럼 정의』

식별 컬럼 정의 수정

프로시저:

임포트 또는 로드 조작으로 임포트한 테이블을 재작성하려고 하고, 또 테이블에 IDENTITY 컬럼이 있다면, 테이블의 내용을 재작성한 다음 IDENTITY 값을 1에서 생성을 시작하도록 재설정합니다. 이렇게 다시 작성된 테이블에 새 행을 삽입할 때, 다시 1에서 시작하는 IDENTITY 컬럼을 원하지 않습니다. IDENTITY 컬럼에서 중복 값을 원하지 않습니다. 이것이 발생하지 않게 하려면, 다음을 수행해야 합니다.

1. 테이블을 재작성하십시오.
2. MODIFIED BY IDENTITYOVERRIDE절을 사용하여 테이블에 데이터를 로드하십시오. 데이터가 테이블에 로드되지만 식별 값은 행에 대해 생성되지 않습니다.
3. 다음과 같이 쿼리를 수행하여 IDENTITY 컬럼에 대한 마지막 카운터 값을 구하십시오.

   ```
   SELECT MAX(<IDENTITY column>)
   ```

   이는 테이블의 IDENTITY 컬럼 값이었던 값과 동일 값을 리턴합니다.
4. 다음과 같이 ALTER TABLE문의 RESTART절을 사용하십시오.

   ```
   ALTER TABLE <table name> ALTER COLUMN <IDENTITY column>
      RESTART WITH <last counter value>
   ```
5. 테이블로 새 행을 삽입하십시오. IDENTITY 컬럼 값이 RESTART WITH절에 지정된 값을 기본으로 생성됩니다.

관련 참조:

- *SQL 참조서, 볼륨 1*의 『MAX 집계 함수』
- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *Command Reference*의 『LOAD Command』

제한조건 변경

삭제한 다음 제한조건을 변경할 수 있으며 그 자리에 새로운 제한조건을 추가할 수 있습니다. 자세한 정보는 다음을 참조하십시오.

- 204 페이지의 『제한조건 추가』

- 207 페이지의 『고유 제한조건 삭제』

제한조건 추가

ALTER TABLE문을 사용하여 제한조건을 추가하십시오. 이 명령문에 대한 자세한 정보는 *SQL 참조서* 매뉴얼을 참조하십시오.

고유 제한조건 추가

프로시저:

고유 제한조건은 기존의 테이블에 추가될 수 있습니다. 제한조건 이름은 ALTER TABLE 문에 지정된 다른 제한조건과 동일할 수 없으며, 테이블 내에서 고유해야 합니다(여기에는 정의된 참조 무결성 제한조건의 이름을 포함합니다). 명령문이 수행되기 전에 기존의 데이터가 새로운 조건에 맞는지 검사합니다.

다음 SQL문은 테이블에서 사원을 식별하는 새로운 방법을 나타내는 고유 제한조건을 EMPLOYEE 테이블에 추가합니다.

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

태스크 관련:

- 109 페이지의 『고유 제한조건 정의』
- 207 페이지의 『고유 제한조건 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

기본 키 추가

프로시저:

대형 테이블에 제한조건을 추가하려면, 테이블을 점검 보류 상태로 놓고, 제한조건을 추가한 후, 테이블에서 위반 행의 통합 목록을 검사하는 것이 훨씬 더 효율적입니다. SET INTEGRITY문을 사용하여 점검 보류 상태를 명시적으로 설정하십시오. 테이블이 상위 테이블인 경우, 모든 종속 테이블 및 하위 테이블에 대해 점검 보류가 내재적으로 설정됩니다.

제어 센터를 사용하여 기본 키를 추가하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **기본 키** 페이지에서 기본 키로서 하나 이상의 컬럼을 선택한 후, 화살표를 눌러 이동시키십시오.
4. 선택적: 기본 키의 제한조건 이름을 입력하십시오.
5. **확인**을 누르십시오.

명령행을 사용하여 기본 키를 추가하려면, 다음을 수행하십시오.

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
   PRIMARY KEY <column_name>
```

태스크 관련:

- 205 페이지의 『외부 키 추가』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『SET INTEGRITY문』

외부 키 추가

프로시저:

외부 키가 테이블에 추가되면, 다음과 같은 명령문이 들어 있는 패키지와 캐시된 동적 SQL은 올바르지 않음으로 표시됩니다.

- 외부 키가 들어 있는 테이블을 삽입하고 갱신하는 명령문
- 상위 테이블을 갱신하거나 삭제하는 명령문

제어 센터를 사용하여 외부 키를 추가하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **외부 키** 페이지에서 **추가**를 누르십시오.
4. **외부 키 추가** 창에서 상위 테이블 정보를 지정하십시오.
5. 하나 이상의 컬럼이 외부 키가 되도록 선택한 후, 화살표를 눌러 이동시키십시오.
6. 상위 테이블의 행이 삭제되거나 갱신될 때 종속 테이블에서 수행할 조치를 지정하십시오. 또한 외부 키에 대한 제한조건 이름을 추가할 수도 있습니다.
7. **확인**을 누르십시오.

명령행을 사용하여 외부 키를 추가하려면, 다음을 수행하십시오.

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
    FOREIGN KEY <column_name>
    ON DELETE <action_type>
    ON UPDATE <action_type>
```

다음 예에서는 기본 키와 외부 키를 테이블에 추가하는 ALTER TABLE문을 보여줍니다.

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
      PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
  ADD CONSTRAINT ACT_PROJ_REF
      FOREIGN KEY (PROJNO)
      REFERENCES PROJECT
      ON DELETE CASCADE
```

관련 개념:

- 233 페이지의 『오브젝트 변경 시 명령문 종속성』

태스크 관련:

- 204 페이지의 『기본 키 추가』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

테이블 점검 제한조건 추가

프로시저:

점검 제한조건은 ALTER TABLE문을 사용하여 기존의 테이블에 추가될 수 있습니다. 제한조건 이름은 ALTER TABLE문에 지정된 기타 제한조건과 동일할 수 없으며, 테이블 내에서 고유해야 합니다. (여기에는 정의된 참조 무결성 제한조건의 이름을 포함합니다.) 명령문이 수행되기 전에 기존의 데이터가 새로운 조건에 맞는지 검사합니다.

대형 테이블에 제한조건을 추가하려면, 테이블을 점검 보류 상태로 놓고, 제한조건을 추가한 후, 테이블에서 위반 행의 통합 목록을 검사하는 것이 훨씬 더 효율적입니다. SET INTEGRITY문을 사용하여 점검 보류 상태를 명시적으로 설정하십시오. 테이블이 상위 테이블인 경우, 모든 종속 테이블 및 하위 테이블에 대해 점검 보류가 내재적으로 설정됩니다.

테이블 점검 제한조건이 추가되면, 테이블을 삽입하거나 갱신하는 패키지 및 캐시된 동적 SQL이 올바르지 않음으로 표시됩니다.

제어 센터를 사용하여 테이블 점검 제한조건을 추가하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **점검 제한조건** 페이지에서 **추가**를 누르십시오.
4. **점검 제한조건 추가**에서 정보를 완료한 후 **확인**을 누르십시오.
5. **점검 제한조건** 페이지에서 **확인**을 누르십시오.

명령행을 사용하여 테이블 점검 제한조건을 추가하려면, 다음을 수행하십시오.

```
ALTER TABLE <name>
   ADD CONSTRAINT <name> (<constraint>)
```

다음 SQL문은 급여와 커미션을 합해 $25,000를 넘어야 한다는 제한조건을 EMPLOYEE 테이블에 추가합니다.

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

관련 개념:

- 233 페이지의『오브젝트 변경 시 명령문 종속성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의『SET INTEGRITY문』

고유 제한조건 삭제

ALTER TABLE문을 사용하여 제한조건을 제거합니다. 이 명령문에 대한 자세한 정보는 *SQL 참조서* 매뉴얼을 참조하십시오.

고유 제한조건 삭제

프로시저:

ALTER TABLE문을 사용하여 고유 제한조건을 명시적으로 제거할 수 있습니다. 테이블의 모든 고유 제한조건의 이름은 SYSCAT.INDEXES 시스템 카탈로그 뷰에 있습니다.

다음 SQL문은 EMPLOYEE 테이블에서 고유 제한조건 NEWID를 제거합니다.

```
ALTER TABLE EMPLOYEE
   DROP UNIQUE NEWID
```

이 고유 제한조건을 제거하면, 제한조건에 사용된 모든 패키지 또는 캐시된 동적 SQL이 올바르지 않음으로 표시됩니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

기본 키 삭제

프로시저:

제어 센터를 사용하여 기본 키를 삭제하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **기본 키** 페이지에서 오른쪽에서 삭제할 기본 키를 선택한 후, 화살표를 눌러 왼쪽에 있는 **사용 가능한 컬럼** 상자로 이동하십시오.
4. **확인**을 누르십시오.

명령행을 사용하여 기본 키를 삭제하려면, 다음을 입력하십시오.

```
ALTER TABLE <name>
  DROP PRIMARY KEY
```

외부 키 제한조건이 제거되면, 다음과 같은 사항이 들어 있는 패키지와 캐시된 동적 SQL 문이 올바르지 않음으로 표시됩니다.

- 외부 키가 들어 있는 테이블을 삽입하고 갱신하는 명령문
- 상위 테이블을 갱신하거나 삭제하는 명령문

관련 개념:

- 233 페이지의 『오브젝트 변경 시 명령문 종속성』

태스크 관련:

- 208 페이지의 『외부 키 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

외부 키 삭제

프로시저:

제어 센터를 사용하여 외부 키를 제거하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **외부 키** 페이지에서 **추가**를 누르십시오.
4. 오른쪽에서 제거할 외부 키를 선택한 후, 화살표를 눌러 왼쪽에 있는 **사용 가능한 컬럼** 상자로 이동하십시오.
5. **외부 키** 페이지에서 **확인**을 누르십시오.

명령행을 사용하여 외부 키를 제거하려면, 다음을 수행하십시오.

```
ALTER TABLE <name>
   DROP FOREIGN KEY <foreign_key_name>
```

다음 예에서는 ALTER TABLE문의 DROP PRIMARY KEY 및 DROP FOREIGN KEY절을 사용하여 테이블에서 기본 키와 외부 키를 제거합니다.

```
ALTER TABLE EMP_ACT
  DROP PRIMARY KEY
  DROP FOREIGN KEY ACT_EMP_REF
  DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
  DROP PRIMARY KEY
```

관련 개념:

- 233 페이지의 『오브젝트 변경 시 명령문 종속성』

태스크 관련:

- 208 페이지의 『기본 키 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

테이블 점검 제한조건 삭제

프로시저:

ALTER TABLE문을 사용하여 테이블 점검 제한조건을 명시적으로 제거 또는 변경하거나, DROP TABLE문의 결과로서 이를 내재적으로 제거할 수 있습니다.

테이블 점검 제한조건을 제거하면, 테이블에 INSERT 또는 UPDATE 종속성이 있는 모든 패키지와 캐시된 SQL문이 올바르지 않음으로 표시됩니다. 테이블에 대한 모든 점검 제한조건의 이름은 SYSCAT.CHECKS 카탈로그 뷰에 있습니다. 시스템이 생성한 이름을 갖는 테이블 점검 제한조건을 제거하기 전에, SYSCAT.CHECKS 카탈로그 뷰에서 이름을 찾으십시오.

제어 센터를 사용하여 테이블 점검 제한조건을 제거하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **점검 제한조건** 페이지에서 제거할 점검 제한조건을 선택하고 **제거**를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 테이블 점검 제한조건을 제거하려면, 다음을 수행하십시오.

```
ALTER TABLE <table_name>
   DROP CHECK <check_constraint_name>
```

다음 SQL문은 EMPLOYEE 테이블에서 테이블 점검 제한조건 REVENUE를 제거합니다.

```
ALTER TABLE EMPLOYEE
   DROP CHECK REVENUE
```

관련 개념:

• 233 페이지의 『오브젝트 변경 시 명령문 종속성』

태스크 관련:

• 206 페이지의 『테이블 점검 제한조건 추가』

관련 참조:

• *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

기존 테이블에 생성된 컬럼 정의

생성된 컬럼은 저장된 값이 삽입 또는 갱신 조작을 통해 지정되기 보다는 표현식을 사용하여 계산되는 기본 테이블에 정의됩니다. 생성된 컬럼은 테이블이 작성될 때 작성되거나 기존 테이블에 대한 수정으로서 작성됩니다.

전제조건:

생성된 컬럼은 동일한 비교가 정의된 데이터 유형에서만 정의될 수 있습니다. 생성된 컬럼에서 제외된 데이터 유형은 다음과 같습니다. 구조화 유형, LOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC 및 제외된 데이터 유형과 동일한 것을 사용하여 정의된 사용자 정의 유형.

생성된 컬럼은 제한조건, 고유 인덱스, 참조 제한조건, 기본 키 및 전역 임시 테이블에서 사용될 수 없습니다. LIKE 및 구체화된 뷰로 작성된 테이블은 생성된 컬럼 등록 정보를 상속하지 않습니다.

제한사항:

생성된 컬럼은 키워드 DEFAULT 없이 삽입되거나 갱신될 수 없습니다. 삽입할 때, DEFAULT를 사용하면 컬럼 목록에 컬럼을 열거할 필요가 없습니다. 대신, 생성된 컬럼은 값 목록에 있는 DEFAULT로 설정될 수 있습니다. 갱신할 때, DEFAULT는 검사하지 않고도 SET INTEGRITY에 의해 온라인으로 위치된 생성된 컬럼의 재계산을 가능하게 합니다.

트리거의 처리 순서는 사전 트리거가 헤더(갱신 이전) 또는 본문에서 생성된 컬럼을 참조할 수 없도록 요구합니다. 처리 순서에서 생성된 컬럼은 사전 트리거 이후에 처리됩니다.

db2look 유틸리티는 생성된 컬럼에 의해 생성된 점검 제한조건을 모릅니다.

복제 사용시, 목표 테이블은 맵핑에서 생성된 컬럼을 사용하지 말아야 합니다. 복제할 때 두가지 선택사항이 있습니다.

- 목표 테이블은 일반 컬럼으로서 생성된 컬럼을 정의해야 합니다.
- 목표 테이블은 맵핑에서 생성된 컬럼을 생략해야 합니다.

생성된 컬럼으로 작업할 때 몇 가지 제한사항이 있습니다.

- 생성된 컬럼은 서로 종속성을 갖지 말아야 합니다.
- 생성된 컬럼을 작성하기 위해 사용되는 표현식은 서브쿼리를 포함하지 말아야 합니다. 이것은 **SQL 데이터를 읽는** 함수와 함께 표현식을 포함합니다.
- 생성된 컬럼에 어떠한 점검 제한조건도 허용되지 않습니다.

프로시저:

생성된 컬럼을 정의하려면, 다음 단계를 수행하십시오.

1. 테이블을 점검 보류 상태로 지정하십시오.

   ```
   SET INTEGRITY FOR t1 OFF
   ```

2. 하나 이상의 생성된 컬럼을 추가하려면 테이블을 변경하십시오.

   ```
   ALTER TABLE t1 ADD COLUMN c3 DOUBLE GENERATED ALWAYS AS (c1 + c2),
                  ADD COLUMN c4 GENERATED ALWAYS AS
                     (CASE WHEN c1 > c3 THEN 1 ELSE NULL END))
   ```

3. 생성된 컬럼에 올바른 값을 지정하십시오. 다음 방법으로 이를 수행할 수 있습니다.
   - 다음을 사용하여 생성된 컬럼에 대한 값을 다시 계산하고 다시 지정하십시오.

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATED
     ```

     로그 스페이스의 부족으로 인해 이 SET INTEGRITY문이 실패하면 사용 가능한 사용 중인 로그 스페이스를 늘린 후 SET INTEGRITY문을 다시 발행하십시오.

     주: 예외 테이블을 여기에서 사용할 수 있습니다.

- 사용 가능한 사용 중인 로그 스페이스를 늘릴 수 없으면, 검색된 갱신 명령문을 사용하여 생성된 컬럼을 디폴트값으로 지정하십시오.
  a. 테이블에서 독점 잠금을 가져오십시오. 이것은 언커미트 읽기 트랜잭션을 제외하고 모두 테이블을 액세스하지 못하게 합니다. 테이블 잠금은 첫 번째 간헐적 커미트 시 해제되며, 아직 디폴트값으로 지정되지 않은 생성된 컬럼이 있는 행을 다른 트랜잭션들이 볼 수 있습니다.

     ```
     LOCK TABLE t1
     ```

  b. 생성된 컬럼의 점검을 생략하십시오.

     ```
     SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
     ```

  c. 테이블에 다른 무결성 위반이 없는지 점검하고(적용 가능한 경우) 점검 보류를 해제하십시오.

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED
     ```

  d. 생성된 컬럼을 간헐적인 커미트 및 술어를 사용해서 갱신하여 로그가 가득 차는 것을 피하십시오.

     ```
     UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <predicate>
     ```

  e. 커미트 명령문을 사용하여 트랜잭션을 완료함으로써 테이블을 잠금해제하십시오.

     ```
     COMMIT
     ```

- 사용 가능한 사용 중인 로그 스페이스를 늘릴 수 없는 경우에는 커서 방식을 사용할 수도 있습니다.
  a. 테이블에 대해 FOR UPDATE 커서를 선언하십시오. 간헐적인 커미트 후 잠금을 유지해야 하는 경우에는 WITH HOLD 옵션을 사용해야 합니다.

     ```
     DECLARE C1 CURSOR WITH HOLD FOR S1
     ```

     여기서 S1은 다음과 같이 정의됩니다.

     ```
     SELECT '0' FROM t1 FOR UPDATE OF C3, C4
     ```

  b. 커서를 여십시오.

     ```
     OPEN C1
     ```

  c. 생성된 컬럼의 점검을 생략하십시오.

     ```
     SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
     ```

  d. 테이블에 다른 무결성 위반이 없는지 점검하고(적용 가능한 경우) 점검 보류를 해제하십시오.

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED
     ```

  e. 루프를 사용하여 테이블에 있는 모든 행을 페치하고, 페치된 각 행에 대해 다음을 실행하여 생성된 컬럼을 디폴트 테이블에 지정하십시오. 커서 지속기간 동안 테이블이 잠금 상태에 있도록 하려면, 테이블이 점검 보류에서 해제된 직후에 첫 번째 페치가 수행되도록 해야 합니다.

```
UPDATE t1 SET (C3, C4) = (DEFAULT, DEFAULT) WHERE CURRENT OF C1
```

로그가 꽉 차지 않도록 하려면 간헐적인 커미트를 수행하십시오.

f. 커서를 닫고 커미트한 후 테이블을 잠금 해제하십시오.

```
CLOSE C1
COMMIT
```

- 테이블이 NOT LOGGED INITIALLY 옵션으로 작성되었습니다. 이러한 방식으로 테이블의 로깅은 생성된 컬럼 값으로 작업하는 중에 일반 포함 및 위험으로 설정 해제됩니다.

  a. NOT LOGGED INITIALLY 옵션을 활성화하십시오.

  ```
  ALTER TABLE t1 ACTIVATE NOT LOGGED INITIALLY
  ```

  b. 값을 생성하십시오.

  ```
  SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATION
  ```

  c. 트랜잭션을 커미트하여 NOT LOGGED INITIALLY 옵션을 설정해제하십시오.

  ```
  COMMIT
  ```

생성된 컬럼의 값은 표현식을 등식 점검 제한조건인 것처럼 적용하여 간단히 검사할 수 있습니다.

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

예를 들어, LOAD를 사용하여 생성된 컬럼에 값이 위치하며, 값이 생성된 표현식과 일치함을 아는 경우, 값을 검사하거나 지정하지 않고 점검 보류 상태로부터 테이블을 가져올 수 있습니다.

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

태스크 관련:

- 115 페이지의 『새 테이블에 생성된 컬럼 정의』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『COMMIT문』
- *SQL 참조서, 볼륨 2*의 『LOCK TABLE문』
- *SQL 참조서, 볼륨 2*의 『SET INTEGRITY문』
- *SQL 참조서, 볼륨 2*의 『UPDATE문』
- *Command Reference*의 『db2look - DB2 Statistics and DDL Extraction Tool Command』

테이블을 volatile로 선언

프로시저:

소멸성 테이블은 런타임 *크기*가 공백에서 대형까지 변할 수 있는 내용을 가진 테이블로 정의됩니다. 이 테이블의 소멸성 또는 극도의 변경 가능성은 RUNSTATS로 수집된 통계 사용을 부정확하게 만듭니다. 통계는 특정 시점에서 수집되고 특정 시점만을 반영합니다. 소멸성 테이블을 사용하는 액세스 플랜을 생성하면, 이는 부정확하거나 성능이 떨어지는 결과를 낳습니다. 예를 들어, 소멸성 테이블이 비어 있을 때 통계가 모아지면, 옵티마이저는 인덱스 스캔보다는 테이블 스캔을 사용하여 소멸성 테이블 액세스를 선호하는 경향이 있습니다.

이를 방지하려면, ALTER TABLE문을 사용하여 테이블을 소멸성(volatile)으로 선언해야 합니다. 테이블을 소멸성으로 선언하면, 옵티마이저는 테이블 스캔보다 인덱스 스캔 사용을 고려합니다. 선언된 소멸성 테이블을 사용하는 액세스 플랜은 해당 테이블에 대해 기존의 통계에 의존하지 않습니다.

제어 센터를 사용하여 테이블을 소멸성으로 선언하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **테이블** 페이지에서 **런타임시에 카디낼리티(cardinality) 변동** 선택란을 선택한 후 **확인**을 누르십시오.

명령행을 사용하여 테이블을 『volatile』으로 선언하려면, 다음을 수행하십시오.

```
ALTER TABLE <table_name>
    VOLATILE CARDINALITY
```

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

파티션 키 변경

프로시저:

단일 파티션 데이터베이스 파티션 그룹에 있는 테이블의 파티션 키만을 변경할 수 있습니다. 우선 기존의 파티션 키를 제거한 다음, 또다른 파티션 키를 작성하십시오.

다음 SQL문은 MIXREC 테이블에서 파티션 키 MIX_INT를 제거합니다.

```
ALTER TABLE MIXREC
   DROP PARTITIONING KEY
```

다중 파티션 데이터베이스 파티션 그룹에 있는 테이블의 파티션 키를 변경할 수 없습니다. 이를 제거하려고 할 경우, 오류가 표시됩니다.

다중 파티션 데이터베이스 파티션 그룹의 파티션 키를 변경하려면, 다음 중 하나를 수행합니다.

- 모든 데이터를 단일 파티션 데이터베이스 파티션 그룹으로 익스포트한 후 위의 지시를 따릅니다.
- 모든 데이터를 익스포트하고 테이블을 삭제하며, 테이블을 재작성하고 파티션 키를 재정의한 다음 모든 데이터를 임포트합니다.

이 방법은 대형 데이터베이스에는 사용 가능하지 않습니다. 그러므로 대형 데이터베이스 설계를 구현하기 전에 적합한 파티션 키를 정의하는 것이 중요합니다.

관련 개념:

- *관리 안내서: 계획*의 『파티션 키』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

테이블 속성 변경

프로시저:

데이터 캡처 옵션, 각 페이지의 여유 스페이스 백분율(PCTFREE), 잠금 크기 또는 추가 모드와 같은 테이블 속성을 변경할 필요가 있을 수 있습니다.

테이블의 각 페이지에 남아 있는 여유 공간의 양은 PCTFREE를 통해 지정되며, 클러스터링 인덱스의 효과적인 사용을 위한 주요 고려사항입니다. 지정할 양은 기존 데이터 및 예상 장래 데이터의 특성에 따라 달라집니다. PCTFREE는 LOAD 및 REORG로 고려되지만, 삽입, 갱신 및 임포트 활동에 의해 무시됩니다.

PCTFREE를 더 큰 값에 설정하면 더 긴 기간 동안 클러스터링을 유지보수하지만, 추가 디스크 스페이스가 필요합니다.

LOCKSIZE 매개변수를 사용하여 테이블이 액세스될 때 사용되는 잠금의 크기(세분성)를 지정할 수 있습니다. 디폴트로, 테이블이 작성될 때 행 레벨 잠금이 정의됩니다. 테이블 레벨 잠금의 사용은 취득하거나 해제하는 데 필요한 잠금 수를 제한하여 쿼리의 성능을 향상시킬 수 있습니다.

APPEND ON을 지정하여, 테이블의 전체 성능을 향상시킬 수 있습니다. 여유 공간에 대한 정보의 유지보수를 줄이고, 더 빠른 삽입을 허용합니다.

클러스터링 인덱스가 있는 테이블은 추가 모드를 설정하도록 변경할 수 없습니다. 마찬가지로, 클러스터링 인덱스는 추가 모드가 있는 테이블에서 작성될 수 없습니다.

관련 개념:

- *관리 안내서: 성능*의 『잠금 및 동시처리 제어』

- *관리 안내서: 성능*의 『잠금 속성』
- *관리 안내서: 성능*의 『잠금 및 성능』
- *관리 안내서: 성능*의 『잠금에 영향을 미치는 인수』
- *관리 안내서: 성능*의 『잠금에 대한 지침』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

식별 컬럼 변경

프로시저:

ALTER TABLE문으로 기존 식별 컬럼의 속성을 수정하십시오.

속성 컬럼이 일부의 시퀀스 특성을 가지도록 수정하는 데에는 몇 가지 방법이 있습니다.

다음과 같이 ALTER TABLE문과 식별 컬럼에 대해 고유한 일부 태스크가 있습니다.

- RESTART는 식별 컬럼과 연관된 시퀀스를 식별 컬럼이 원래 작성될 때 시작값으로서 내재적으로나 명시적으로 지정된 값으로 재설정합니다.
- RESTART WITH <numeric-constant>는 식별 컬럼과 연관된 시퀀스를 정확한 숫자 상수 값으로 재설정합니다. 숫자 상수는 식별 컬럼에 지정되는 소수점의 오른쪽에 0이 아닌 숫자가 하나도 없는 임의의 양수 또는 음수 값입니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』

시퀀스 변경

프로시저:

ALTER SEQUENCE문으로 기존 시퀀스의 속성을 수정하십시오.

수정할 수 있는 시퀀스 속성에는 다음이 포함됩니다.

- 장래 값 사이의 증분 변경
- 새로운 최소값 또는 최대값 설정
- 캐시 시퀀스 번호의 수 변경
- 시퀀스가 순환할 것인지 아닌지의 변경
- 시퀀스 번호가 요청 순서대로 생성되어야 하는지의 변경
- 시퀀스 재시작

시퀀스 작성의 일부가 아닌 두 개의 태스크가 있습니다. 태스크는 다음과 같습니다.

- RESTART. 식별 컬럼이 작성될 때 시작값으로서 내재적으로나 명시적으로 지정된 값에 시퀀스를 재설정합니다.
- RESTART WITH <numeric-constant>. 정확한 숫자 상수 값에 시퀀스를 재설정합니다. 숫자 상수는 소수점의 오른쪽에 0이 아닌 숫자가 하나도 없는 임의의 양수 또는 음수 값입니다.

시퀀스를 재시작하거나 CYCLE로 변경한 후, 중복 시퀀스 번호를 생성할 수 있습니다. 장래 시퀀스 번호만은 ALTER SEQUENCE문의 영향을 받습니다.

시퀀스의 데이터 유형은 변경할 수 없습니다. 대신, 현재 시퀀스를 제거한 다음, 새로운 데이터 유형을 지정하는 시퀀스를 새로 작성해야 합니다.

시퀀스 변경시, DB2가 사용하지 않는 캐시 시퀀스 값은 모두 잃게 됩니다.

태스크 관련:

- 217 페이지의『시퀀스 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER SEQUENCE문』

시퀀스 삭제

프로시저:

시퀀스를 삭제하려면, DROP문을 사용하십시오.

특정 시퀀스는 다음을 사용하여 제거할 수 있습니다.

```
DROP SEQUENCE sequence_name
```

여기서는 sequence_name이 제거할 시퀀스의 이름이고, 기존 시퀀스를 정확히 식별하기 위해 내재적 또는 명시적 스키마 이름을 포함합니다.

IDENTITY 컬럼에 대해 시스템이 생성하는 시퀀스는 DROP SEQUENCE문을 사용하여 제거할 수 없습니다.

시퀀스를 한번만 제거하면, 시퀀스의 모든 특권도 제거됩니다.

태스크 관련:

- 216 페이지의『시퀀스 변경』

관련 참조:

- *SQL 참조서, 볼륨 2*의『DROP문』

구체화된 쿼리 테이블 등록 정보 변경

프로시저:

구체화된 쿼리 테이블을 테이블로 변경하거나 일반 테이블을 구체화된 쿼리 테이블로 변경할 수 있으나 몇 가지 제한사항이 있습니다. 다른 테이블 유형은 변경할 수 없습니다. 일반 및 구체화된 쿼리 테이블만 변경할 수 있습니다. 예를 들어, 복제된 구체화된 쿼리 테이블을 일반 테이블로 변경하거나 일반 테이블을 복제된 구체화된 쿼리 테이블로 변경할 수는 없습니다.

일반 테이블이 구체화된 쿼리 테이블로 변경되면 이 테이블은 점검 보류 상태에 놓입니다. 이러한 방식으로 변경할 때, 구체화된 쿼리 테이블 정의의 fullselect는 원래 테이블 정의와 일치해야 합니다. 즉 다음과 같아야 합니다.

- 컬럼 수가 동일해야 합니다.
- 컬럼 이름 및 위치가 일치해야 합니다.
- 데이터 유형이 동일해야 합니다.

구체화된 쿼리 테이블이 원래의 테이블에 정의된 경우, 원래의 테이블 자체는 구체화된 쿼리 테이블로 변경할 수 없습니다. 원래 테이블에 트리거, 점검 제한조건, 참조 제한조건 또는 정의된 고유 인덱스가 있으면, 이 테이블을 구체화된 쿼리 테이블로 변경할 수 없습니다. 테이블 등록 정보를 변경하여 구체화된 쿼리 테이블을 정의할 경우, 동일한 ALTER TABLE문에서 다른 방식으로 테이블을 변경할 수 없습니다.

일반 테이블을 구체화된 쿼리 테이블로 변경할 때, 구체화된 쿼리 테이블 정의의 fullselect는 직접적으로나 혹은 뷰, 별명 또는 구체화된 쿼리 테이블을 통해 간접적으로 원래 테이블을 참조할 수 없습니다.

구체화된 쿼리 테이블을 일반 테이블로 변경하려면 다음을 사용하십시오.

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

일반 테이블을 구체화된 쿼리 테이블로 변경하려면 다음을 사용하십시오.

```
ALTER TABLE regtable
   SET SUMMARY AS <fullselect>
```

일반 테이블을 구체화된 쿼리 테이블로 변경할 때의 fullselect에 대한 제한사항은 CREATE SUMMARY TABLE문을 사용하여 요약 테이블을 작성할 때의 제한사항과 매우 유사합니다.

태스크 관련:

- 146 페이지의『구체화된 쿼리 테이블 작성』
- 219 페이지의『구체화된 쿼리 테이블의 데이터 새로 고침』
- 230 페이지의『구체화된 쿼리 또는 스테이징 테이블 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

구체화된 쿼리 테이블의 데이터 새로 고침

프로시저:

REFRESH TABLE문을 사용하여 하나 이상의 구체화된 쿼리 테이블에 있는 데이터를 새로 고칠 수 있습니다. 응용프로그램에 해당 명령문을 삽입하거나 동적으로 발행할 수 있습니다. 이 명령문을 사용하려면 새로 고칠 테이블에 대해 SYSADM 또는 DBADM 권한 또는 CONTROL 특권을 가지고 있어야 합니다.

다음 예는 구체화된 쿼리 테이블의 데이터를 새로 고치는 방법을 보여줍니다.

```
REFRESH TABLE SUMTAB1
```

태스크 관련:

- 146 페이지의 『구체화된 쿼리 테이블 작성』
- 218 페이지의 『구체화된 쿼리 테이블 등록 정보 변경』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『REFRESH TABLE문』

사용자 정의 구조화된 유형 변경

프로시저:

구조화 유형을 작성한 후, 해당 구조화 유형과 연관된 속성을 추가하거나 제거해야 하는 경우가 있습니다. ALTER TYPE(구조화)문을 사용하여 이 작업을 수행할 수 있습니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『사용자 정의 구조화된 유형』
- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『구조화된 유형 계층 구조』

태스크 관련:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『구조화된 유형 작성』의

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER TYPE(구조화)문』

유형이 지정된 테이블의 행 삭제 및 갱신

검색되거나 배치된 DELETE문을 사용하여 유형이 지정된 테이블에서 행을 삭제할 수 있습니다. 검색되거나 배치된 UPDATE문을 사용하여 유형이 지정된 테이블에서 행을 갱신할 수 있습니다.

관련 개념:

- *응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍*의 『유형이 지정된 테이블』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DELETE문』
- *SQL 참조서, 볼륨 2*의 『UPDATE문』

기존 테이블 또는 인덱스 이름 바꾸기

스키마 내에서 기존 테이블 또는 인덱스에 새 이름을 부여하고 원래의 테이블에 작성된 권한 부여와 인덱스를 유지보수할 수 있습니다.

전제조건:

이름을 바꿀 기존 테이블 또는 인덱스는 테이블 또는 인덱스를 식별하는 별명일 수 있습니다.

제한사항:

이름을 바꿀 기존 테이블 또는 인덱스는 카탈로그 테이블 또는 인덱스, 요약 테이블 또는 인덱스, 유형이 지정된 테이블, 선언된 전역 임시 테이블 또는 별칭의 이름이 아니어야 하며 테이블, 뷰 또는 별명 이외의 오브젝트의 이름이 아니어야 합니다.

기존 테이블 또는 인덱스는 다음에서 참조될 수 없습니다.

- 뷰
- 트리거
- 참조 제한조건
- 요약 테이블
- 기존 참조 컬럼의 범위

또한, 테이블 내에 점검 제한조건이 있어서는 안되며, 식별 컬럼 이외에 생성된 컬럼이 있어서도 안됩니다. 원래의 테이블에 종속된 모든 패키지 또는 캐시된 동적 SQL문은 유효하지 않습니다. 마지막으로, 원래의 테이블을 언급하는 모든 별명은 수정되지 않습니다.

해당 시스템 카탈로그 테이블을 점검하여 이름을 바꿀 테이블 또는 인덱스가 이러한 제한사항에 의해 영향을 받지 않도록 해야 합니다.

프로시저:

제어 센터를 사용하여 기존 테이블 또는 인덱스의 이름을 바꾸려면 다음을 수행하십시오.

1. 오브젝트 트리를 펼쳐 **테이블** 또는 **뷰** 폴더를 찾으십시오.
2. 이름을 바꾸고자 하는 테이블 또는 뷰를 마우스 오른쪽 단추로 누른 후 팝업 메뉴에서 **이름 바꾸기**를 선택하십시오.
3. 새 테이블 또는 뷰 이름을 입력하고 **확인**을 누르십시오.

명령행을 사용하여 기존 테이블의 이름을 바꾸려면, 다음을 입력하십시오.

```
RENAME TABLE <schema_name>.<table_name> TO <new_name>
```

다음 SQL문은 COMPANY 스키마 내의 EMPLOYEE 테이블을 EMPL로 이름 바꾸기합니다.

```
RENAME TABLE COMPANY.EMPLOYEE TO EMPL
```

명령행을 사용하여 기존 인덱스의 이름을 바꾸려면 다음을 입력하십시오.

```
RENAME INDEX <schema_name>.<index_name> TO <new_name>
```

다음 SQL문은 COMPANY 스키마 내의 EMPIND 인덱스를 MSTRIND로 이름을 바꿉니다.

```
RENAME INDEX COMPANY.EMPIND TO MSTRIND
```

패키지가 방금 이름이 바뀐 테이블이나 인덱스를 참조할 경우에는 패키지가 무효화되며 리바인드해야 합니다. 동일한 이름을 가지는 다른 인덱스의 유무와 무관하게 패키지는 내재적으로 리바인드됩니다. 더 나은 선택항목이 없는 한, 패키지는 새 이름 아래에 이전에 사용했던 인덱스를 사용합니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『RENAME문』

MERGE문을 사용하여 테이블 및 뷰 내용 갱신

DB2 Universal Database는 다른 소스(일반적으로 테이블 참조 결과)에 있는 데이터를 사용하여 테이블 또는 뷰를 갱신하는 기능을 제공합니다. 이는 MERGE문을 사용하여 수행된 갱신 유형입니다.

MERGE문 내에 지정된 지시문을 기본으로 소스가 일치하는 목표 테이블의 행을 삭제하거나 갱신할 수 있습니다. 목표 테이블에 존재하지 않는 행은 삽입할 수 없습니다.

뷰의 행을 갱신, 삭제 또는 삽입하면 뷰가 기본으로 하는 테이블에서도 해당 행이 갱신, 삭제 또는 삽입됩니다.

제한사항:

테이블 또는 뷰가 기본으로 하는 테이블에 대해 세 가지 가능한 조치(갱신, 삭제 또는 삽입)를 수행하려면 MERGE문에 연관된 권한 부여 ID가 적절한 특권을 갖고 있어야 합니다. 권한 부여 ID는 서브쿼리의 테이블 또는 기본 뷰 테이블에 대해서도 적절한 특권을 갖고 있어야 합니다.

MERGE문에 오류가 발생하는 경우, MERGE와 연관된 전체 조작 세트가 롤백됩니다.

MERGE문을 실행한 후에 생긴 목표 테이블 또는 기본 뷰 테이블에 있는 행은 갱신할 수 없습니다. 즉, MERGE문의 일부로 삽입된 행은 갱신할 수 없습니다.

뷰가 MERGE문의 목표로서 지정될 경우, 뷰에 대해 INSTEAD OF 트리거가 정의되어 있지 않으므로 갱신, 삭제 및 삽입 조작을 수행할 때마다 INSTEAD OF 트리거를 정의해야 합니다.

프로시저:

목표 테이블에 대해 이러한 조치의 조합을 갱신, 삽입 또는 수행할 경우, 명령 프롬프트에 다음을 입력하십시오.

```
MERGE INTO <table or view name>
   USING <table reference> ON <search condition>
   WHEN <match condition> THEN <modification operation or signal statement>
```

수정 조작 및 신호 명령문은 MERGE 문에 두 번 이상 지정할 수 있습니다. 목표 테이블 또는 뷰의 각 행은 단일 MERGE문에서는 한 번만 조작될 수 있습니다. 이는 목표 테이블 또는 뷰의 한 행만이 테이블 참조 결과 테이블의 한 행과 일치되는 (MATCHED) 것으로 식별될 수 있음을 의미합니다.

두 개의 테이블(운송 및 재고)이 있는 상황을 생각해 보십시오. 운송 테이블을 사용하여 행을 재고 테이블에 병합하십시오. 일치하는 행의 경우, 재고 테이블에 있는 수량에 운송 테이블에 있는 수량을 더하십시오. 그렇지 않으면, 새 부품 번호를 재고 테이블에 삽입하십시오.

```
   MERGE INTO inventory AS in
      USING (SELECT partno, description, count FROM shipment
      WHERE shipment. partno IS NOT NULL) AS sh
      ON (in.partno = sh.partno)
      WHEN MATCHED THEN
UPDATE SET
            description = sh.description
        quantity = in.quantity + sh.count
      WHEN NOT MATCHED THEN
```

```
INSERT
   (partno, description, quantity)
       VALUES (sh.partno, sh.description, sh.count)
```

이 예에는 DELETE 옵션이 없습니다. DELETE 옵션을 추가하면 좀더 복잡한 일치 조건을 사용할 수 있습니다. 여기에 문서화되지 않은 여러 가지 옵션(예: 신호 명령문 및 ELSE 절)이 있으나 SQL 참조서에는 들어있지 않습니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『MERGE문』

테이블 삭제

프로시저:

테이블은 DROP TABLE SQL문으로 삭제될 수 있습니다.

테이블이 삭제되면, 삭제된 테이블에 대한 정보가 들어 있는 SYSCAT.TABLES 카탈로그의 행과, 테이블에 의존하는 기타 오브젝트는 영향을 받습니다. 예를 들어, 다음과 같습니다.

- 모든 컬럼 이름이 삭제됩니다.
- 테이블의 컬럼에 대해 작성된 인덱스가 삭제됩니다.
- 테이블에 근거한 모든 뷰가 작동 불능 상태로 표시됩니다.
- 삭제된 테이블과 종속 뷰에 대한 모든 특권도 내재적으로 권한 취소됩니다.
- 상위 테이블 또는 종속 테이블에 있는 모든 참조 제한조건이 삭제됩니다.
- 삭제된 테이블에 종속된 모든 패키지 및 캐시된 동적 SQL문은 올바르지 않음으로 표시되며 종속 오브젝트가 재작성될 때까지 이 상태로 남아 있습니다. 계층 구조에서 삭제 중인 하위 테이블 위에 있는 모든 상위 테이블에 종속된 패키지도 여기에 포함됩니다.
- 삭제된 테이블이 참조 범위로서 정의된 모든 참조 컬럼은 『범위 없는 상태』가 됩니다.
- 별명이 정의될 수 없기 때문에 테이블에 대한 별명 정의는 효력이 없습니다.
- 삭제된 테이블에 종속적인 모든 트리거는 작동 불능으로 표시됩니다.
- DATALINK 컬럼과 링크된 모든 파일은 링크해제됩니다. 링크해제 조작은 비동기식으로 수행되는데, 파일을 기타 조작용으로 즉시 사용할 수 없다는 의미입니다.

제어 센터를 사용하여 테이블을 삭제하려면, 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 삭제하려는 테이블을 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **삭제**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 테이블을 삭제하려면, 다음을 입력하십시오.

```
DROP TABLE <table_name>
```

다음 명령문은 DEPARTMENT라고 하는 테이블을 삭제합니다.

```
DROP TABLE DEPARTMENT
```

각각의 테이블은 하위테이블이 있을 때는 삭제될 수 없습니다. 그러나 테이블 계층 구조에 있는 모든 테이블은 다음 예에서처럼 단일 DROP TABLE HIERARCHY문으로 삭제될 수 있습니다.

```
DROP TABLE HIERARCHY person
```

DROP TABLE HIERARCHY문은 삭제될 계층 구조의 루트 테이블을 이름 지정해야 합니다.

특정 테이블을 삭제하는 것과 테이블 계층 구조를 삭제하는 것에는 차이가 있습니다.

- DROP TABLE HIERARCHY는 개별 DROP 테이블 명령문으로 활성화되는 삭제된 트리거를 활성화시키지 않습니다. 예를 들어, 개별 서브테이블을 삭제하면, 상위 테이블에 대한 삭제 트리거가 활성화됩니다.
- DROP TABLE HIERARCHY는 제거된 테이블의 개별 행에 대해 로그 항목을 작성하지 않습니다. 대신 계층 구조 제거는 단일 이벤트로 로그됩니다.

관련 개념:

- 233 페이지의『오브젝트 변경 시 명령문 종속성』

태스크 관련:

- 225 페이지의『사용자 정의 임시 테이블 삭제』
- 229 페이지의『작동 불능 뷰 복구 중』

관련 참조:

- *SQL 참조서, 볼륨 2*의『DROP문』

사용자 정의 임시 테이블 삭제

사용자 정의 임시 테이블은 DECLARE GLOBAL TEMPORARY TABLE문을 사용하여 작성합니다.

전제조건:

이러한 테이블을 제거할 때, 테이블 이름은 스키마 이름 SESSION으로 규정되어야 하며 테이블을 작성한 응용프로그램에 있어야 합니다.

제한사항:

패키지는 이 테이블 유형에 종속될 수 없으므로 이러한 테이블을 제거할 때 유효합니다.

프로시저:

사용자 정의 임시 테이블이 제거되고, 임시 테이블 작성이 사용 중인 작업 단위(UOW) 또는 저장 지점을 선행한 경우, 테이블은 기능적으로 삭제되어 응용프로그램이 테이블에 액세스할 수 없습니다. 그러나 테이블은 여전히 테이블 스페이스에 예약된 일부 스페이스를 가지며 이것은 작업 단위(UOW)가 커미트되거나 저장 지점이 종료될 때까지 사용자 임시 테이블 스페이스를 삭제하지 못하게 합니다.

태스크 관련:

- 117 페이지의『사용자 정의 임시 테이블 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『DROP문』
- *SQL 참조서, 볼륨 2*의『SET SCHEMA문』

트리거 삭제

프로시저:

트리거 오브젝트는 DROP문을 사용하여 제거될 수 있지만, 이 프로시저로 인해 종속 패키지는 다음과 같이 올바르지 않음으로 표시됩니다.

- 명시적 컬럼 목록 없이 갱신 트리거가 제거될 경우, 목표 테이블에서 갱신 사용이 있는 패키지가 유효하지 않음으로 표시됩니다.
- 컬럼 목록과 함께 갱신 트리거가 제거될 경우, 패키지가 CREATE TRIGGER문의 `column-name` 목록 내의 적어도 하나의 컬럼에서 갱신 사용이 있을 경우에만 목표 테이블에 갱신 사용이 있는 패키지가 유효하지 않음으로 표시됩니다.

- 삽입 트리거가 제거될 경우, 목표 테이블에서 삽입 사용이 있는 패키지가 유효하지 않음으로 표시됩니다.
- 삭제 트리거가 제거될 경우, 목표 테이블에서 삭제 사용이 있는 패키지가 유효하지 않음으로 표시됩니다.

패키지는 응용프로그램이 명시적으로 바인드 또는 리바인드될 때까지 올바르지 않음으로 있거나, 패키지가 수행되면 데이터베이스 관리 프로그램이 자동으로 이를 리바인드합니다.

태스크 관련:

- 131 페이지의『트리거 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『DROP문』

사용자 정의 함수(UDF), 함수 맵핑 또는 메소드 삭제

사용자 정의 함수(UDF), 함수 템플리트 또는 함수 맵핑은 DROP문으로 제거될 수 있습니다.

전제조건:

기타 오브젝트는 함수 또는 함수 템플리트에 종속될 수 있습니다. 함수 맵핑을 비롯한 이러한 모든 종속성은 함수가 제거되기 전에 제거되어야 합니다. 단, 작동하지 않는 것으로 표시된 패키지는 예외입니다.

제한사항:

뷰, 트리거, 테이블 점검 제한조건 또는 또다른 UDF가 이에 종속적인 경우에는, UDF가 제거될 수 없습니다. CREATE DISTINCT TYPE문에 의해 내재적으로 생성된 함수는 제거될 수 없습니다. SYSIBM 스키마 또는 SYSFUN 스키마에 있는 함수는 제거할 수 없습니다.

프로시저:

DISABLE 맵핑 옵션으로 함수 맵핑을 사용할 수 없게 할 수 있습니다.

작동 불능으로 표시된 패키지는 내재적으로 리바인드되지 않습니다. 패키지는 BIND 또는 REBIND 명령을 사용하여 리바인드하거나 PREP 명령을 사용하여 준비해야 합니다. UDF를 제거하면, 이것을 사용한 다른 패키지 또는 캐시된 동적 SQL문이 유효하지 않음으로 표시됩니다.

함수 맵핑을 제거하면, 패키지는 유효하지 않음으로 표시됩니다. 자동 리바인드가 일어나며 옵티마이저는 로컬 함수를 사용하려고 시도합니다. 로컬 함수가 템플리트인 경우, 내재적 리바인드는 실패합니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DROP문』
- *Command Reference*의 『BIND Command』
- *Command Reference*의 『PRECOMPILE Command』
- *Command Reference*의 『REBIND Command』

사용자 정의 유형(UDT) 또는 유형 맵핑 삭제

DROP문을 사용하여 사용자 정의 유형(UDT) 또는 유형 맵핑을 제거할 수 있습니다.

제한사항:

UDT가 다음에 사용될 때는 제거할 수 없습니다.

- 기존 테이블 또는 뷰(구별 유형)에 대한 컬럼 정의에 사용
- 기존의 입력된 테이블 또는 입력된 뷰(구조화 유형)의 유형으로 사용
- 또다른 구조화 유형의 슈퍼 유형으로 사용

디폴트 유형 맵핑을 삭제할 수 없습니다. 또다른 유형 맵핑을 작성하여 이를 겹쳐쓰기만 할 수 있습니다.

데이터베이스 관리 프로그램은 이러한 구별 유형에 종속적인 모든 함수 제거를 시도할 것입니다. UDF가 제거될 수 없으면, UDT도 제거될 수 없습니다. 뷰, 트리거, 테이블 점검 제한조건 또는 또다른 UDF가 이에 종속적인 경우에는, UDF가 제거될 수 없습니다. UDT를 제거하면, 이것을 사용한 다른 패키지 또는 캐시된 동적 SQL문이 유효하지 않음으로 표시됩니다.

사용자 또는 기타 응용프로그램 개발자가 정의한 변환만 제거될 수 있다는 점에 유의하십시오. 내장된 변환과 연관된 그룹 정의는 제거될 수 없습니다.

프로시저:

DROP문은 사용자 정의 유형을 삭제하는데 사용됩니다.

UDT에 대한 변환을 작성하고 UDT를 제거할 계획이라면, 변환을 제거해야 하는지 고려해야 합니다. 이는 DROP TRANSFORM문을 통해 완료됩니다.

관련 개념:

- 139 페이지의 『사용자 정의 유형(UDT)』

태스크 관련:

- 140 페이지의 『사용자 정의 구별 유형 작성』
- 142 페이지의 『유형 맵핑 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DROP문』

뷰 변경 및 삭제

ALTER VIEW문은 참조 유형 컬럼을 변경하여 범위를 추가함으로써 기존 뷰 정의를 수정합니다. DROP문은 뷰를 삭제합니다.

전제조건:

뷰를 변경할 때에는 아직 범위가 정의되지 않은 기존 참조 유형 컬럼에 범위를 추가해야 합니다. 그리고 상위 뷰로부터 컬럼을 상속할 수 없습니다.

제한사항:

뷰의 기초가 되는 컨텐츠를 변경할 경우에는 트리거를 사용해야 합니다. 뷰에 대한 다른 변경을 수행할 경우에는 해당 뷰를 삭제한 후 다시 작성해야 합니다.

프로시저:

ALTER VIEW문의 컬럼 이름의 데이터 유형은 REF(유형이 지정된 테이블 이름 또는 유형이 지정된 뷰 이름의 유형)여야 합니다. INSTEAD OF 트리거를 통해 뷰의 컨텐츠를 수정할 수도 있습니다.

패키지 및 캐시된 동적 명령문에 올바르지 않음 표시가 되어 있더라도, 테이블 및 인덱스와 같은 기타 데이터베이스 오브젝트에는 영향을 주지 않습니다.

제어 센터를 사용하여 뷰에 대한 정의를 변경하려면 다음을 수행하십시오.

1. **뷰** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 수정하려는 뷰를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **변경**을 선택하십시오.
3. **뷰 변경** 창에서 주석을 입력하거나 수정한 후, **확인**을 누르십시오.

명령행을 사용하여 뷰를 변경하려면, 다음을 입력하십시오.

```
ALTER VIEW <view_name> ALTER <column name>
   ADD SCOPE <typed table or view name>
```

제어 센터를 사용하여 뷰를 제거하려면, 다음을 수행하십시오.

1. **뷰** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 제거하려는 뷰를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 뷰를 제거하려면, 다음을 수행하십시오.

```
DROP VIEW <view_name>
```

다음 예에서는 EMP_VIEW를 제거하는 방법을 보여줍니다.

```
DROP VIEW EMP_VIEW
```

제거될 뷰에 종속적인 모든 뷰가 작동 불능 상태로 됩니다.

테이블 계층 구조의 경우, 다음 예에서처럼 루트 뷰를 이름 지정하여 하나의 명령문에서 전체 뷰 계층 구조를 제거할 수 있습니다.

```
DROP VIEW HIERARCHY VPerson
```

관련 개념:

- 233 페이지의『오브젝트 변경 시 명령문 종속성』

태스크 관련:

- 131 페이지의『트리거 작성』
- 142 페이지의『뷰 작성』
- 229 페이지의『작동 불능 뷰 복구 중』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER VIEW문』
- *SQL 참조서, 볼륨 2*의『DROP문』

작동 불능 뷰 복구 중

프로시저:

뷰는 다음 경우 작동 불능이 될 수도 있습니다.

- 기본 테이블에 대한 특권을 취소한 결과로서.
- 테이블, 별명 또는 함수가 삭제될 경우.
- 슈퍼 뷰가 작동 불능이 될 경우(슈퍼 뷰는 또 하나의 유형이 지정된 뷰인 서브뷰의 기초가 되는 유형이 지정된 뷰입니다).
- 종속인 뷰가 제거될 경우

다음 단계는 작동 불능 뷰를 복구하는 데 도움이 됩니다.

1. 뷰를 작성하기 위해 초기에 사용된 SQL문을 판별하십시오. SYSCAT.VIEW 카탈로그 뷰의 TEXT 컬럼으로부터 이 정보를 얻을 수 있습니다.
2. 동일한 뷰 이름과 동일한 정의로 CREATE VIEW문을 사용하여 뷰를 재작성하십시오.
3. GRANT문을 사용하여 뷰에 대해 이전에 권한 부여된 모든 특권을 다시 권한 부여하십시오(작동 불능 뷰에 권한 부여된 특권이 모두 권한 취소됨을 유의하십시오).

작동 불능 뷰를 복구하지 않을 경우, DROP VIEW문으로 작동 불능 뷰를 명시적으로 제거하거나 또는 정의는 다르지만 동일한 이름을 가진 새 뷰를 작성할 수 있습니다.

작동 불능 뷰에는 SYSCAT.TABLES 및 SYSCAT.VIEWS 카탈로그 뷰의 항목만 있습니다. SYSCAT.VIEWDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS 및 SYSCAT.COLAUTH 카탈로그 뷰에 있는 모든 항목은 제거됩니다.

태스크 관련:

- 228 페이지의 『뷰 변경 및 삭제』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE VIEW문』
- *SQL 참조서, 볼륨 2*의 『DROP문』
- *SQL 참조서, 볼륨 2*의 『GRANT(테이블, 뷰 또는 별칭 특권)문』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.VIEWS 카탈로그 뷰』

구체화된 쿼리 또는 스테이징 테이블 삭제

프로시저:

구체화된 쿼리 또는 스테이징 테이블을 변경할 수는 없지만 삭제할 수는 있습니다.

테이블을 참조하는 모든 인덱스, 기본 키, 외부 키 및 점검 제한조건이 제거됩니다. 테이블을 참조하는 모든 뷰 및 트리거는 사용 불가능 상태가 됩니다. 제거되거나 작동 불능 상태로 표시된 오브젝트에 종속되는 모든 패키지는 유효하지 않음으로 표시됩니다.

제어 센터를 사용하여 구체화된 쿼리 테이블을 삭제하려면 다음을 수행하십시오.

1. **테이블** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 삭제할 구체화된 쿼리 또는 스테이징 테이블을 마우스 오른쪽 단추로 누른 후 팝업 메뉴에서 **삭제**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 구체화된 쿼리 테이블을 삭제하려면 다음을 입력하십시오.

```
DROP TABLE <table_name>
```

다음 SQL문은 구체화된 쿼리 테이블 XT를 삭제합니다.

```
DROP TABLE XT
```

구체화된 쿼리 테이블은 DROP TABLE문으로 명시적으로 삭제되거나 또는 기초가 되는 테이블이 삭제될 때 내재적으로 삭제될 수 있습니다.

스테이징 테이블은 DROP TABLE문으로 명시적으로 삭제되거나 또는 연관된 구체화된 쿼리 테이블이 삭제될 때 내재적으로 삭제될 수 있습니다.

관련 개념:

- 233 페이지의『오브젝트 변경 시 명령문 종속성』

태스크 관련:

- 146 페이지의『구체화된 쿼리 테이블 작성』
- 151 페이지의『스테이징 테이블 작성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『DROP문』

작동 불능 요약 테이블 복구

프로시저:

기본 테이블의 SELECT 특권을 권한 취소하면 요약 테이블이 *작동 불능* 상태가 될 수 있습니다.

다음 단계는 작동 불능 요약 테이블을 복구하는 데 도움이 됩니다.

- 요약 테이블을 작성하기 위해 초기에 사용된 SQL문을 판별하십시오. SYSCAT.VIEW 카탈로그 뷰의 TEXT 컬럼으로부터 이 정보를 얻을 수 있습니다.
- 동일한 요약 테이블 이름 및 동일한 정의를 통해 CREATE SUMMARY TABLE 문을 사용하여 요약 테이블을 재작성하십시오.
- GRANT문을 사용하여, 요약 테이블에 이전에 권한 부여된 모든 특권을 다시 권한 부여하십시오. (작동 불능 요약 테이블에 권한 부여된 특권이 모두 권한 취소됨을 유의하십시오.)

작동 불능 요약 테이블을 복구하지 않으려는 경우, DROP TABLE문을 사용하여 작동 불능 요약 테이블을 명시적으로 제거하거나 정의는 다르지만 동일한 이름을 사용하여 새 요약 테이블을 작성할 수 있습니다.

작동 불능 요약 테이블에는 SYSCAT.TABLES 및 SYSCAT.VIEWS 카탈로그 뷰에 있는 항목만 있습니다. SYSCAT.VIEWDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS 및 SYSCAT.COLAUTH 카탈로그 뷰의 모든 항목은 제거됩니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』
- *SQL 참조서, 볼륨 2*의 『DROP문』
- *SQL 참조서, 볼륨 2*의 『GRANT(테이블, 뷰 또는 별칭 특권)문』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.VIEWS 카탈로그 뷰』

인덱스, 인덱스 확장 또는 인덱스 스펙 삭제

제한사항:

인덱스 정의, 인덱스 확장 또는 인덱스 스펙의 절을 변경할 수 없습니다. 인덱스 또는 인덱스 확장을 제거한 후 재작성해야 합니다(인덱스 또는 인덱스 스펙을 제거해도 다른 오브젝트가 제거되지는 않지만, 일부 패키지는 유효하지 않음으로 표시될 수 있습니다).

인덱스 확장의 이름은 카탈로그에 기술된 인덱스 확장을 식별해야 합니다. RESTRICT 절은 인덱스 확장 정의에 종속되는 인덱스를 정의할 수 없다는 규칙을 시행합니다. 기본 인덱스가 이 인덱스 확장에 종속되는 경우, 삭제에 실패합니다.

기본 키 또는 고유 인덱스 키(인덱스 스펙이 아닌 한)는 명시적으로 제거될 수 없습니다. 다음 방법 중 하나를 사용하여 이를 제거해야 합니다.

- 1차 인덱스 또는 고유 제한조건이 기본 키 또는 고유 키를 사용하여 자동으로 작성된 경우, 기본 키 또는 고유 키를 삭제하면 인덱스가 삭제됩니다. 삭제는 ALTER TABLE문을 사용하여 완료됩니다.
- 1차 인덱스 또는 고유 제한조건이 사용자 정의된 경우, 기본 키 또는 고유 키는 ALTER TABLE문을 사용하여 먼저 삭제되어야 합니다. 기본 키 또는 고유 키가 제거된 후, 인덱스는 더 이상 1차 인덱스 또는 고유 인덱스로 간주되지 않고, 명시적으로 삭제될 수 있습니다.

프로시저:

제어 센터를 사용하여 인덱스, 인덱스 확장 또는 인덱스 스펙을 제거하려면, 다음을 수행하십시오.

1. **인덱스** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 제거하려는 인덱스를 마우스 오른쪽 단추로 누른 후, 팝업 메뉴에서 **제거**를 선택하십시오.
3. **확인** 상자를 누른 후, **확인**을 누르십시오.

명령행을 사용하여 인덱스, 인덱스 확장 또는 인덱스 스펙을 제거하려면, 다음을 입력하십시오.

```
DROP INDEX <index_name>
```

다음 SQL문은 PH라고 하는 인덱스를 제거합니다.

```
DROP INDEX PH
```

다음 SQL문은 IX_MAP이라고 하는 인덱스 확장을 제거합니다.

```
DROP INDEX EXTENSION ix_map RESTRICT
```

제거된 인덱스에 의존하는 모든 패키지 및 캐시된 동적 SQL문은 올바르지 않음으로 표시됩니다. 응용프로그램은 인덱스의 추가 또는 제거로 인한 변경사항의 영향을 받지 않습니다.

관련 개념:

- 233 페이지의『오브젝트 변경 시 명령문 종속성』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의『DROP문』

오브젝트 변경 시 명령문 종속성

명령문 종속성에는 패키지 및 캐시된 동적 SQL문이 포함됩니다. *패키지*란 특정 응용프로그램에 대해 가장 효과적인 방법으로 데이터에 액세스하기 위해 데이터베이스 관리 프로그램이 필요로 하는 정보가 들어 있는 데이터베이스 오브젝트입니다. *바인딩*이란 응용프로그램이 실행될 때 데이터베이스에 액세스하기 위해 데이터베이스 관리 프로그램이 필요로 하는 패키지를 작성하는 프로세스입니다.

패키지 및 캐시된 동적 SQL문은 여러 유형의 오브젝트에 종속될 수 있습니다.

이들 오브젝트는 SQL SELECT문에 포함된 테이블 또는 사용자 정의 함수와 같이 명시적으로 참조될 수 있습니다. 오브젝트는 또한 내재적으로 참조될 수도 있습니다. 예를 들어, 상위 테이블의 행이 삭제될 때 참조 제한조건에 위반되지 않도록 하기 위해 검사되어야 하는 종속 테이블이 여기에 해당됩니다. 패키지는 패키지 작성자에게 권한 부여된 특권에도 종속됩니다.

패키지 또는 캐시된 동적 SQL문이 오브젝트에 종속되고 해당 오브젝트가 삭제되면, 패키지 또는 캐시된 동적 SQL문이『유효하지 않은』상태가 됩니다. 패키지가 사용자 정의 함수(UDF)에 종송되고 해당 함수가 삭제되면, 패키지는『작동 불능』상태가 됩니다.

올바르지 않은 상태에 있는 캐시된 동적 SQL문은 다음에 사용될 때 자동으로 다시 최적화됩니다. 명령문에서 필요한 오브젝트가 제거되면, 동적 SQL문의 실행은 오류 메시지와 함께 실패하게 됩니다.

올바르지 않은 상태에 있는 패키지는 다음에 사용될 때 내재적으로 리바인드됩니다. 이러한 패키지 또한 명시적으로 리바인드될 수 있습니다. 트리거가 제거되었기 때문에 패키지가 올바르지 않은 것으로 표시된 경우, 리바인드 패키지는 더 이상 트리거를 호출하지 않습니다.

작동 불능 상태에 있는 패키지는 사용되기 전에 명시적으로 리바인드되어야 합니다.

페더레이티드 데이터베이스 오브젝트에는 비슷한 종속성이 있습니다. 예를 들어, 서버를 제거하면, 해당 서버에 연관된 별칭을 참조하는 패키지 또는 캐시된 동적 SQL은 유효하지 않게 됩니다.

일부 경우에는, 패키지를 리바인드하는 것이 불가능할 때도 있습니다. 예를 들어, 테이블이 제거된 후 재작성되지 않으면, 패키지는 리바인드될 수 없습니다. 이 경우, 오브젝트를 재작성하거나 제거된 오브젝트를 사용하지 않도록 응용프로그램을 변경해야 합니다.

대부분의 경우, 제한조건 중 하나가 제거될 경우에는 패키지를 리바인드할 수 있습니다.

다음 시스템 카탈로그 뷰는 패키지 및 패키지 종속성의 상태를 판별하는 데 도움을 줍니다.

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

관련 개념:

- *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 『BIND 명령을 사용하여 패키지 작성』
- *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 『응용프로그램, 바인드 파일 및 패키지 관련사항』
- *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 『패키지 리바인드』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『DROP문』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.PACKAGEAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.PACKAGEDEP 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.PACKAGES 카탈로그 뷰』
- *Command Reference*의 『BIND Command』

- *Command Reference*의 『REBIND Command』

제 2 부 데이터베이스 보안

제 7 장 데이터베이스 액세스 제어

데이터베이스 관리자와 시스템 관리자의 가장 중요한 책임 중의 하나는 데이터베이스 보안입니다. 데이터베이스의 보안에는 여러 가지 활동이 포함됩니다.

- 장비 또는 시스템 고장으로 인한 데이터 또는 데이터 무결성의 갑작스런 손실을 방지
- 권한 부여되지 않은 액세스로부터 가치 있는 데이터를 보호. 중요한 정보가 『권한』 없는 사용자에 의해 액세스되지 않는지 확인하십시오.
- 권한 부여되지 않은 사용자가 악의로 데이터를 삭제하거나 고치는 등의 위해를 가하지 못하도록 방지
- 299 페이지의 제 8 장 『DB2 Universal Database™(DB2 UDB) 활동 감사』에서 언급한 사용자의 데이터 액세스 모니터링

이 절에는 다음과 같은 내용이 들어 있습니다.

- 『DB2 Universal Database 설치시 보안 문제점』
- 246 페이지의 『서버에 대한 인증 방법』
- 252 페이지의 『리모트 클라이언트에 대한 인증 고려사항』
- 252 페이지의 『파티션된 데이터베이스 인증 고려사항』
- 296 페이지의 『방화벽 지원 입문』
- 257 페이지의 『특권, 권한 레벨 및 데이터베이스 권한』
- 278 페이지의 『데이터베이스 오브젝트에 대한 액세스 제어』
- 290 페이지의 『태스크 및 필수 권한 부여』
- 291 페이지의 『보안 문제에 시스템 카탈로그 사용』.

보안을 위한 계획: 데이터베이스 액세스 제어 플랜의 목표를 정의하는 것으로 시작하여, 누가 어떤 환경에서 무엇에 액세스할 것이지 지정하십시오. 사용자의 계획에는 데이터베이스 기능, 기타 프로그램의 기능 및 관리 프로시저를 사용하여 이러한 목표를 충족시킬 수 있는 방법이 다루어져야 합니다.

DB2 Universal Database 설치시 보안 문제점

보안 문제점은 제품이 설치되는 시점부터 DB2® 관리자에게 중요합니다.

DB2 설치를 완료하려면 사용자 ID, 그룹 이름 및 암호가 필요합니다. GUI 기반 DB2 설치 프로그램은 서로 다른 사용자 ID 및 그룹에 대해 서로 다른 디폴트값을 생성합니다. Linux 또는 Windows 플랫폼에서 설치하는지 여부에 따라 서로 다른 디폴트값이 생성됩니다.

- Linux 플랫폼에서, DB2 설치 프로그램은 DAS(dasusr), 인스턴스 소유자(db2inst) 및 분리 사용자(db2fenc)에 대해 서로 다른 디폴트 사용자를 작성합니다.

  DB2 설치 프로그램은 아직 존재하지 않는 사용자 ID가 작성될 수 있을 때까지 1-99의 숫자를 디폴트 사용자 이름에 추가합니다. 예를 들어, 사용자 db2inst1 및 db2inst2가 이미 존재하면 DB2 설치 프로그램은 사용자 db2inst3을 작성합니다. 10 이상의 숫자가 사용되면 이름의 문자 부분이 디폴트 사용자 ID에서 절단됩니다. 예를 들어, 사용자 ID db2fenc9가 이미 존재하면 DB2 설치 프로그램은 사용자 ID에서 c를 자른 후에 10을 추가합니다(db2fen10). 숫자 값이 디폴트 DAS 사용자에 추가되는 경우에는 절단이 발생하지 않습니다(예: dasusr24).
- Windows 플랫폼에서, DB2 설치 프로그램은 DAS 사용자, 인스턴스 소유자 및 분리 사용자에 대한 디폴트 사용자 db2admin을 작성합니다. Linux 플랫폼과는 달리 사용자 ID에는 숫자 값이 추가되지 않습니다.

관리자 이외의 사용자가 부적절하게 데이터베이스 및/또는 인스턴스를 액세스하는 위험성을 최소화하기 위해서는 디폴트 사용자 ID 이름 및/또는 암호를 변경해야 합니다.

주: 응답 파일 설치는 사용자 ID 또는 그룹 이름에 대해 디폴트값을 사용하지 않습니다. 이 값은 응답 파일에 지정되어야 합니다.

암호는 사용자를 인증할 때 매우 중요합니다. 운영 체제 레벨에서 인증 요구사항이 설정되어 있지 않으며 데이터베이스가 사용자를 인증하기 위해 운영 체제를 사용하는 경우에는 사용자의 연결이 허용됩니다. 예를 들어, Linux 운영 체제에서 정의되지 않은 암호는 널(NULL)로 처리됩니다. 이러한 상황에서 정의된 암호가 없는 임의의 사용자는 널(NULL) 암호를 갖는 것으로 간주됩니다. 운영 체제의 관점에서 이는 일치하는 것이며 사용자는 유효성이 확인되고 데이터베이스에 연결할 수 있습니다. 운영 체제가 데이터베이스에 대한 사용자의 인증을 수행하도록 하려면 운영 체제 레벨에서 암호를 사용하십시오.

주: 데이터베이스 환경이 일반 기준 요구사항을 충족하도록 하려는 경우에는 정의되지 않은 암호를 사용할 수 없습니다.

또한 DB2 Universal Database를 설치한 후에는 사용자에 부여된 디폴트 특권을 검토하고 (필요하면) 변경하십시오. 디폴트로 설치 프로세스는 각 운영 체제의 다음 사용자에게 시스템 관리(SYSADM) 특권을 부여합니다.

Windows® 9x	임의의 Windows 98 또는 Windows ME 사용자
기타 **Windows 환경**	Windows NT, Windows 2000, Windows XP 또는 Windows Server 2003에서, 관리자 그룹에 속하는 유효한 DB2 사용자 이름
Linux 플랫폼	인스턴스 소유자의 기본 그룹에 속하는 유효한 DB2 사용자 이름

SYSADM 특권은 DB2 Universal Database 내에서 사용 가능한 가장 강력한 특권 세트입니다. 따라서 이 모든 사용자가 디폴트로 SYSADM 특권을 갖는 것은 바람직하지 않습니다. DB2는 관리자에게 그룹 및 개별 사용자 ID에게 특권을 부여하고 취소하는 기능을 제공합니다.

데이터베이스 관리 프로그램 구성 매개변수 *sysadm_group*을 갱신함으로써, 관리자는 SYSADM 특권을 소유하는 사용자의 그룹을 제어할 수 있습니다. 아래의 지침에 따라 DB2 설치와 후속 인스턴스 및 데이터베이스 작성 모두에 대한 보안 요구사항을 완료해야 합니다.

(*sysadm_group*을 갱신함으로써) 시스템 관리 그룹으로 정의된 임의의 그룹이 존재해야 합니다. 이 그룹의 이름은 인스턴스 소유자에 대해 작성된 그룹으로서 식별을 쉽게 할 수 있도록 합니다. 이 그룹에 속하는 사용자 ID 및 그룹에는 개별 인스턴스에 대한 시스템 관리자 권한이 있습니다.

관리자는 특정 인스턴스와 연관된 것으로 쉽게 인식되는 인스턴스 소유자 사용자 ID를 작성하는 것을 고려해야 합니다. 이 사용자 ID는 자체 그룹 중 하나로서 위에서 작성된 SYSADM 그룹의 이름을 가져야 합니다. 다른 권장사항은 이 인스턴스 소유자 사용자 ID를 인스턴스 소유자 그룹의 구성원으로서만 사용하며 이를 기타 그룹에서 사용하지 않는 것입니다. 이는 인스턴스 또는 인스턴스 내의 임의의 오브젝트를 수정할 수 있는 사용자 ID 및 그룹의 확산을 제어해야 합니다.

작성된 사용자 ID는 인스턴스 내의 데이터 및 데이터베이스의 항목으로 허용되기 전에 인증의 제공을 위해 암호와 연관되어야 합니다. 암호를 작성하는 경우에는 해당 조직의 암호 이름 지정 지침을 따르도록 권장합니다.

액세스 토큰을 사용하여 Windows 사용자 그룹 정보 확보

액세스 토큰은 프로세스 또는 스레드의 보안 컨텍스트를 설명하는 오브젝트입니다. 액세스 토큰의 정보에는 프로세스 또는 스레드와 연관된 사용자 어카운트의 ID 및 특권이 포함됩니다.

로그온 시, 시스템은 암호를 보안 데이터베이스에 저장된 정보와 비교하여 검증합니다. 암호가 인증될 경우, 시스템은 액세스 토큰을 생성합니다. 사용자의 이름으로 실행되는 모든 프로세스는 이 액세스 토큰의 사본을 사용합니다.

또한 액세스 토큰은 캐시된 증명서를 기반으로 확보할 수 있습니다. 시스템에 인증되면, 운영 체제에서 증명서를 캐시합니다. 도메인 제어기에 접속할 수 없는 때 마지막 로그온의 액세스 토큰을 캐시에서 참조할 수 있습니다.

액세스 토큰에는 사용자가 속하는 모든 그룹 즉, 로컬 그룹 및 다양한 도메인 그룹(전역 그룹, 도메인 로컬 그룹 및 범용 그룹)에 관한 정보가 포함됩니다.

주: 액세스 토큰 지원이 사용 가능한 경우에도 클라이언트 인증을 사용한 그룹 찾아보기가 원격 연결에서 지원되지 않습니다.

액세스 토큰 지원을 사용 가능하게 하려면, **db2set** 명령을 사용하여 DB2®_GRP_LOOKUP 레지스트리 변수를 갱신해야 합니다. 레지스트리 변수 갱신 시 선택사항은 다음과 같습니다.

- TOKEN

  이 선택사항은 사용자가 속하는 모든 그룹을 사용자 어카운트가 정의된 위치에서 찾아보기 위한 액세스 토큰 지원을 사용합니다. 이 위치는 일반적으로 도메인이거나 DB2 Universal Database™(DB2 UDB) 서버에 로컬입니다.

- TOKENLOCAL

  이 선택사항은 사용자가 속하는 모든 로컬 그룹을 DB2 UDB 서버에서 찾아보기 위한 액세스 토큰 지원을 사용합니다.

- TOKENDOMAIN

  이 선택사항은 사용자가 속하는 모든 도메인 그룹을 도메인에서 찾아보기 위한 액세스 토큰 지원을 사용합니다.

액세스 토큰 지원을 사용할 경우, 어카운트 관리 인프라구조에 영향을 미치는 몇 가지 한계가 있습니다. 해당 지원이 사용될 경우, DB2 UDB는 데이터베이스에 연결 중인 사용자에 관한 그룹 정보를 수집합니다. 다른 권한 부여 ID에 종속성을 갖는 CONNECT 또는 ATTACH 요청이 성공한 이후 후속 조작은 여전히 기존의 그룹 열거를 사용해야 합니다. 중첩된 전역 그룹, 도메인 로컬 그룹 및 캐시된 증명서의 액세스 토큰 이점은 사용 가능하지 않습니다. 예를 들어, 연결 후 SET SESSION_USER를 사용하여 또다른 권한 부여 ID로 실행할 경우, 세션에 대한 새 권한 부여 ID에 부여될 권한을 점검할 때 기존의 그룹 열거만이 사용됩니다. 권한 부여 ID가 속하는 그룹에 대한 특권 부여 및 취소와 반대로, DB2 UDB에 알려진 개별 권한 부여 ID에 명시적 특권을 부여하거나 취소해야 합니다.

그룹에 SYSADM, SYSMAINT 또는 SYSCTRL을 지정하려는 경우, 지정된 그룹이 내포된 전역 그룹 또는 도메인 로컬 그룹이 아닌지 확인해야 하며 캐시된 증명서 기능은 필요하지 않습니다.

DB2 UDB가 기존 그룹 열거 방식을 사용하여 그룹 찾아보기를 수행해야 하는 위치를 표시하려면 DB2_GRP_LOOKUP 레지스트리 변수를 사용하여 그룹 찾아보기 위치를 지정해야 합니다. 예를 들어,

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

이 변수는 로컬 그룹 열거에 액세스 토큰 지원을 사용할 수 있게 해줍니다. DB2 UDB 서버에서 연결된 사용자와 다른 권한 부여 ID에 대한 그룹 찾아보기를 수행합니다.

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

이 변수는 사용자 ID가 정의된 위치에서 그룹 열거에 액세스 토큰 지원을 사용할 수 있게 해줍니다. 사용자 ID가 정의된 위치에서 연결된 사용자와 다른 권한 부여 ID에 대한 그룹 찾아보기를 수행합니다.

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

이 변수는 도메인 그룹 열거에 액세스 토큰 지원을 사용할 수 있게 해줍니다. 사용자 ID가 정의된 위치에서 연결된 사용자와 다른 권한 부여 ID에 대한 그룹 찾아보기를 수행합니다.

DYNAMICRULES RUN(디폴트값)을 사용하여 바운드된 패키지에서 동적 SQL을 사용하는 응용프로그램은 응용프로그램을 실행하는 사용자의 특권으로 실행됩니다. 이 경우, 이미 언급한 한계는 적용되지 않습니다. 여기에는 JDBC 및 DB2 CLI를 사용하도록 작성된 응용프로그램이 포함됩니다.

액세스 토큰 지원은 CLIENT 인증을 제외한 모든 인증 유형에서 사용할 수 있습니다.

주: Windows® NT 4.0 사용자의 경우, DB2 UDB 응용프로그램이 로컬 내재적 연결을 사용 중이면 액세스 토큰 지원은 프로세스 레벨의 보안 컨텍스트만을 지원합니다. 즉, 응용프로그램의 모든 스레드가 응용프로그램을 실행 중인 사용자의 보안 컨텍스트에서 실행 중인 것처럼 처리됩니다. 서로 다른 스레드에 대해 서로 다른 사용자 보안 컨텍스트가 필요할 경우, Windows 2000 이상으로 이동하거나 명시적 연결을 사용하도록 DB2 UDB 응용프로그램을 변경해야 합니다.

관련 개념:

• 239 페이지의『DB2 Universal Database 설치시 보안 문제점』

운영 체제 기반의 보안에 대한 세부사항

각 운영 체제에서 보안을 관리할 수 있는 방법을 제공합니다. 운영 체제와 연관된 보안 문제 중 일부는 이 섹션에서 설명합니다.

주: DB2 Universal Database™(DB2 UDB)가 SERVER_ENCRYPT 인증 사용 시 사용자 ID 및 암호의 암호화를 수행할 때와 DATA_ENCRYPT 인증 사용 시 사용자 ID, 암호 및 사용자 데이터의 암호화를 수행할 때 사용하는 암호 표기 루틴은 FIPS 140-2를 준수합니다.

암호 표기 루틴은 IBM Crypto for C(ICC) 버전 1.2.1에서 제공합니다. ICC에 대한 FIPS 140-2 유효성 확인 인증서 No. 384를 다음의 NIST 웹 사이트에서 찾을 수 있습니다.
http://csrc.nist.gov/cryptval/140-1/140crt/140crt384.pdf

또한 보안 규정을 다음의 NIST 웹 사이트에서 찾을 수 있습니다.
http://csrc.nist.gov/cryptval/140-1/140sp/140sp384.pdf

보안 규정에는 DB2 UDB를 FIPS 140-2를 준수하는 방법으로 설치하기 위해 수행해야 하는 지침이 있으며 자세한 정보를 보려면 5.3.2 섹션을 참조해야 합니다.

FIPS 140-2 준수는 다음과 같은 시스템에서만 사용 사용 가능합니다: AIX, Microsoft Windows, Solaris 운영 환경, Linux 및 HP-UX. 지원되는 시스템의 자세한 세부사항을 보려면 유효성 확인 인증서 및 보안 규정을 참조해야 합니다.

사용자에 대한 Windows NT 플랫폼 보안 고려사항

시스템 관리(SYSADM) 권한은 어카운트가 정의된 머신의 로컬 Administrators 그룹에 속하는 유효한 DB2® Universal Database(DB2 UDB) 사용자 어카운트에 부여됩니다.

디폴트로 Windows® 도메인 환경에서는 도메인 제어기에서 Administrators 그룹에 속하는 도메인 사용자만이 인스턴스에 대한 SYSADM 권한을 가집니다. DB2 UDB는 항상 어카운트가 정의된 머신에서 권한 부여를 수행하므로, 서버의 로컬 Administrators 그룹에 도메인 사용자를 추가할 때 도메인 사용자 SYSADM 권한이 그룹에 부여되지 않습니다.

주: Windows NT 도메인 환경에서 DB2 UDB는 요구사항 및 제한사항을 충족시키고 사용자 ID가 속하는 첫 번째 64그룹만을 인증합니다. 그룹은 64개를 초과할 수 없습니다.

PDC에서 관리자 그룹에 도메인 사용자를 추가하지 않으려면, 전역 그룹을 작성하여 SYSADM 권한을 부여하려는 사용자(도메인 및 로컬 둘다)를 추가하십시오. 이렇게 하려면, 다음 명령을 입력하십시오.

```
DB2STOP
DB2 UPDATE DBM CFG USING SYSADM_GROUP global_group
DB2START
```

관련 개념:

• 245 페이지의『사용자에 대한 UNIX 플랫폼 보안 고려사항』

Windows 로컬 시스템 어카운트 지원

Windows® 플랫폼에서 DB2® Universal Database(DB2 UDB)는 로컬 내재 연결을 사용하여 로컬 시스템 어카운트(LSA) 컨텍스트에서 실행 중인 응용프로그램을 지원합니다. 이 어카운트에서 실행될 어플리케이션을 작성 중인 개발자는 DB2 UDB에『SYS』

로 시작하는 스키마 이름을 가진 오브젝트에 관한 제한사항이 있음을 주의해야 합니다. 따라서 응용프로그램에 DB2 UDB 오브젝트를 작성하는 DDL이 포함될 경우, 다음과 같이 작성되어야 합니다.

- 정적 SQL의 경우, 디폴트값이 아닌 QUALIFIER 옵션에 대한 값으로 바운드되어야 합니다.
- 동적 SQL의 경우, 작성할 오브젝트가 DB2 UDB에서 지원하는 스키마 이름으로 내재적으로 규정되거나 또는 CURRENT SCHEMA 레지스터가 DB2 UDB에서 지원하는 스키마 이름으로 설정되어야 합니다.

LSA에 대한 그룹 정보는 DB2 UDB 인스턴스가 시작된 후 첫 번째 그룹 찾아보기 요청에서 수집되며 인스턴스가 재시작될 때까지 새로 고쳐지지 않습니다.

관련 개념:

- 239 페이지의『DB2 Universal Database 설치시 보안 문제점』

사용자에 대한 UNIX 플랫폼 보안 고려사항

DB2® Universal Database(DB2 UDB)는 직접 데이터베이스 관리자로서 동작하는 root를 지원하지 않습니다. **su - <instance owner>**를 데이터베이스 관리자로 사용해야 합니다.

보안상의 이유로, 분리 ID로서 인스턴스 이름을 사용하지 않도록 권장합니다. 그러나 분리 UDF 또는 스토어드 프로시저를 사용하려고 계획하지 않는 경우, 또다른 사용자 ID를 작성하는 대신 인스턴스 이름에 분리 ID를 설정할 수 있습니다.

권장사항은 이 그룹과 연관하여 인식되는 사용자 ID를 작성하는 것입니다. 분리 UDF 및 스토어드 프로시저에 대한 사용자는 인스턴스 작성 스크립트의 매개변수(**db2icrt... -u <FencedID>**)로서 지정됩니다. DB2 클라이언트 또는 DB2 Software Developer's Kit를 설치하는 경우 이것은 필요하지 않습니다.

관련 개념:

- 244 페이지의『사용자에 대한 Windows NT 플랫폼 보안 고려사항』

인스턴스 디렉토리 위치

UNIX®에서 **db2icrt** 명령은 인스턴스 소유자의 홈 디렉토리 아래에 기본 SQL 라이브러리(`sqllib`) 디렉토리를 작성합니다.

Windows® 운영 체제, 인스턴스 디렉토리는 DB2®가 설치된 `/sqllib` 서브디렉토리에 위치합니다.

관련 개념:

- 17 페이지의『인스턴스 작성』

태스크 관련:

- 23 페이지의『추가 인스턴스 작성』

보안 플러그인

DB2® Universal Database(DB2 UDB)에서 인증은 *보안 플러그인*을 통해 수행됩니다. 자세한 정보는 *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 "보안 플러그인"을 참조하십시오.

서버에 대한 인증 방법

인스턴스 또는 데이터베이스로의 액세스에는 사용자가 *인증이 된 상태*여야 합니다. 각 인스턴스에 대한 *인증 유형*은 사용자가 확인되는 방법과 확인될 장소를 결정합니다. 인증 유형은 서버의 데이터베이스 관리 프로그램 구성 파일에 저장됩니다. 인스턴스가 작성될 때에 처음 설정됩니다. 인스턴스마다 하나의 인증 유형이 있으며, 해당 데이터베이스 서버와 제어하에 있는 모든 데이터베이스로의 액세스를 포함합니다.

페더레이티드 데이터베이스에서 데이터 소스에 액세스하려는 경우, 페더레이티드 인증 유형에 대한 데이터 소스 데이터베이스 처리 및 정의를 고려해야 합니다.

다음 인증 유형이 제공됩니다.

SERVER

인증이 로컬 운영 체제 보안을 사용하여 서버에서 발생함을 지정합니다. 사용자 ID와 암호가 연결 또는 접속 시도 중에 지정되는 경우, 서버의 유효한 사용자 ID 및 암호의 조합과 비교되어 사용자가 인스턴스에 액세스하는 데 허가를 받았는지 판별합니다. 이것이 디폴트 보안 메커니즘입니다.

주:

1. 서버 코드는 연결이 로컬인지 리모트인지를 감지합니다. 로컬 연결의 경우, 인증이 SERVER라면, 사용자 ID와 암호가 인증에 필요하지 않습니다.
2. DB2® Universal Database(DB2 UDB)를 설치하여 일반 기준 인증 구성을 설정하려면, SERVER를 지정해야 합니다.

SERVER_ENCRYPT

서버가 암호화된 SERVER 인증 스킴을 승인하도록 지정합니다. 클라이언트 인증이 지정되지 않으면, 클라이언트는 서버에서 선택된 방법을 사용하여 인증됩니다.

클라이언트 인증이 SERVER이면 클라이언트는 사용자 ID 및 암호를 서버에 전달함으로써 인증됩니다. 클라이언트 인증이 SERVER_ENCRYPT이면 클라이언트는 암호화된 사용자 ID 및 암호화된 암호를 전달함으로써 인증됩니다.

SERVER_ENCRYPT가 클라이언트에서 지정되고 SERVER가 서버에서 지정되면, 인증 레벨 불일치로 인해 오류가 리턴됩니다.

CLIENT

운영 체제 보안을 사용하여 응용프로그램이 호출된 데이터베이스 파티션에서 인증이 발생함을 지정합니다. 연결 또는 접속 시도 중에 지정된 사용자 ID와 암호는 클라이언트 노드에서 유효한 사용자 ID 및 암호의 조합과 비교되어 사용자 ID가 인스턴스로의 액세스를 허가받았는지 판별합니다. 데이터베이스 서버에서 인증은 더 이상 발생하지 않습니다. 이를 종종 단일 사인온이라고 합니다.

사용자가 로컬 또는 클라이언트 로그인을 수행하면, 사용자는 해당 로컬 클라이언트 워크스테이션에만 알려집니다.

리모트 인스턴스가 CLIENT 인증을 가지고 있으면, 다른 두 매개변수 (*trust_allclnts* 및 *trust_clntauth*)가 최종 인증 유형을 판별합니다.

트러스트된 클라이언트 전용에 대한 CLIENT 레벨 보안

트러스트된 클라이언트는 신뢰 받는 로컬 보안 시스템을 갖는 클라이언트입니다. Windows® 9x 운영 체제를 제외한 모든 클라이언트는 트러스트된 클라이언트입니다.

CLIENT의 인증 유형이 선택되면, 추가 옵션이 선택되어 운영 환경에 고유의 보안이 없는 클라이언트를 보호합니다.

보안이 없는 클라이언트를 보호하려면, 관리자는 *trust_allclnts* 매개변수를 NO로 설정하여 트러스트된 클라이언트 인증을 선택할 수 있습니다. 이는 모든 트러스트된 플랫폼이 서버를 대신하여 사용자를 인증할 수 있음을 나타냅니다. 트러스트되지 않은 클라이언트는 서버에서 인증을 받고 사용자 ID와 암호를 제공해야 합니다. *trust_allclnts* 구성 매개변수는 사용자가 트러스트된 클라이언트인지 나타내기 위해 사용합니다. 매개변수의 디폴트값은 YES입니다.

주: 모든 클라이언트(*trust_allclnts*가 YES임)가 인증에 대한 원시(native) 안전 보안 시스템이 없는 클라이언트로 신뢰 받을 수 있습니다.

트러스트된 클라이언트라 하더라도 서버에서 완전한 인증을 받아야 할 경우가 있습니다. 트러스트된 클라이언트의 유효성을 확인하는 곳을 나타내기 위해, *trust_clntauth* 구성 매개변수를 사용합니다. 이 매개변수의 디폴트값은 CLIENT입니다.

주: 트러스트된 클라이언트만을 위해, CONNECT 또는 ATTACH 시도 중에 사용자 ID 또는 암호가 명시적으로 제공되면, 사용자의 유효성이 클라이언트에서 확인됩니다. *trust_clntauth* 매개변수는 USER 또는 USING절에서 제공된 정보의 유효성을 확인하는 곳을 판별하는데에만 사용됩니다.

DRDA® 클라이언트를 제외한 모든 클라이언트로부터 OS/390 및 z/OS용 DB2, VM 및 VSE용 DB2, 그리고 iSeries용 DB2를 보호하려면 *trust_allclnts* 매개변수를 DRDAONLY로 설정하십시오. 이 클라이언트만이 클라이언트측 인증을 수행하도록 신뢰 받을 수 있습니다. 기타 모든 클라이언트는 서버에 의해 인증되려면 사용자 ID와 암호를 제공해야 합니다.

trust_clntauth 매개변수는 위 클라이언트가 인증되는 위치를 판별하는 데 사용됩니다. *trust_clntauth*가 "client"이면, 인증은 클라이언트에서 발생합니다. *trust_clntauth*가 "server"이면, 인증은 암호가 제공되지 않을 경우는 클라이언트에서 암호가 제공되면 서버에서 발생합니다.

표 5. TRUST_ALLCLNTS 및 TRUST_CLNTAUTH 매개변수 조합을 사용한 인증 모드

TRUST_ ALLCLNTS	**TRUST_ CLNTAUTH**	신뢰성 없는 비**DRDA** 클라이언트 인증. 암호 없음	신뢰성 없는 비**DRDA** 클라이언트 인증. 암호 있음	신뢰성 있는 비**DRDA** 클라이언트 인증. 암호 없음	신뢰성 있는 비**DRDA** 클라이언트 인증. 암호 있음	**DRDA** 클라이언트 인증. 암호 없음	**DRDA** 클라이언트 인증. 암호 있음
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

KERBEROS

Kerberos 보안 프로토콜을 지원하는 운영 체제에 DB2 UDB 클라이언트 및 서버 둘다가 있을 때 사용됩니다. Kerberos 보안 프로토콜은 공유 비밀 키를 작성하기 위해 일반 암호를 사용하여 써드 파티 인증 서비스로서 인증을 수행합니다. 이 키는 사용자 증명되며 로컬 또는 네트워크 서비스가 요청될 때 모든 경우에서 사용자 식별을 검증하는 데 사용됩니다. 키는 명확한 텍스트로서 네트워크에서 사용자 이름 및 암호를 전달하는 필요성을 줄입니다. Kerberos를 사용하여 보안 프로토콜은 리모트 DB2 UDB 서버로의 단일 사인온의 사용을 가능하게 합니다. KERBEROS 인증 유형은 Windows 2000, AIX® 및 Solaris 운영 환경에서 실행되는 클라이언트 및 서버에서 지원됩니다.

Kerberos 인증은 다음과 같이 기능합니다.

1. 도메인 어카운트를 사용하여 클라이언트 머신에 로그온하는 사용자는 도메인 제어기의 Kerberos 키 분산 센터(KDC)에 인증을 요청합니다. 키 분산 센터는 클라이언트에 TGT(ticket-granting ticket)를 발행합니다.
2. 연결의 첫 번째 단계에서 서버는 DB2 UDB 서버 서비스에 대한 서비스 어카운트 이름인 목표 핵심부 이름을 클라이언트로 보냅니다. 서버의 목표 핵심부 이름 및 TGT를 사용하여 클라이언트는 도메인 제어기에 있는 TGS(ticket-granting service)에 서비스 티켓을 요청합니다. 클라이언트의

TGT와 서버의 목표 핵심부 이름이 둘 다 유효하면 TGS가 클라이언트에 서비스 티켓을 발행합니다. 이제 데이터베이스 디렉토리에 기록된 핵심부 이름을 name/instance@REALM으로 지정할 수 있습니다. (이 형식은 DB2 UDB 버전 7.1 이상을 갖춘 Windows 2000에서 승인되는 현재 DOMAIN\userID 및 userID@xxx.xxx.xxx.com에 대한 추가사항입니다.)

3. 클라이언트는 통신 채널(예: TCP/IP)을 통해 이 서비스 티켓을 서버로 보냅니다.
4. 서버는 클라이언트 서버 티켓의 유효성을 확인합니다. 클라이언트의 서비스 티켓이 유효하면 인증이 완료됩니다.

클라이언트 머신에서 데이터베이스를 카탈로그화하고 명시적으로 서버의 목표 핵심부 이름과 함께 Kerberos 인증 유형을 지정할 수 있습니다. 그러면 연결의 첫 번째 단계를 생략할 수 있습니다.

사용자 ID 및 암호를 지정하면, 클라이언트는 해당 사용자 어카운트에 대한 TGT를 요청하여 이것을 인증에 사용합니다.

KRB_SERVER_ENCRYPT

서버가 KERBEROS 인증 또는 암호화된 SERVER 인증 스킴을 승인하도록 지정합니다. 클라이언트 인증이 KERBEROS이면, 클라이언트는 Kerberos 보안 시스템을 사용하여 인증됩니다. 클라이언트 인증이 SERVER_ENCRYPT이면, 클라이언트는 사용자 ID 및 암호화된 암호를 사용하여 인증됩니다. 클라이언트 인증이 지정되지 않으면, 클라이언트는 사용 가능한 경우 Kerberos를 사용하며 그렇지 않은 경우 암호 암호화를 사용합니다. 기타 클라이언트 인증 유형에 대하여 인증 오류가 리턴됩니다. 클라이언트의 인증 유형을 KRB_SERVER_ENCRYPT로 지정할 수 없습니다.

주: Kerberos 인증 유형은 Windows 2000, Windows XP, Windows Windows Server 2003 및 AIX 운영 체제와 Solaris 운영 환경에서 실행되는 클라이언트 및 서버에서만 지원됩니다. 또한 클라이언트 및 서버 머신 둘 다 동일한 Windows 도메인에 속하거나 트러스트된 도메인에 속해야 합니다. 서버가 Kerberos를 지원하고 일부 클라이언트 머신이 Kerberos 인증을 지원할 때 이 인증 유형을 사용해야 합니다.

DATA_ENCRYPT

서버가 암호화된 SERVER 인증 스킴 및 사용자 데이터의 암호화를 승인합니다. 인증은 SERVER_ENCRYPT로 표시된 것과 정확히 동일한 방법으로 작동합니다. 자세한 정보는 해당 인증 유형을 참조하십시오.

이 인증 유형을 사용할 경우 다음과 같은 사용자 데이터가 암호화됩니다.

- SQL문
- SQL 프로그램 변수 데이터
- SQL문에 대한 서버 처리로부터의 출력 데이터 및 데이터에 대한 설명

- 쿼리 결과 생성된 응답 세트 데이터의 일부 또는 전부
- 대형 오브젝트(LOB) 데이터 스트림
- SQLDA 디스크립터

DATA_ENCRYPT_CMP

서버가 암호화된 SERVER 인증 스킴 및 사용자 데이터의 암호화를 승인합니다. 추가로, 이 인증 유형은 DATA_ENCRYPT 인증 유형을 지원하지 않는 하위 레벨 제품과의 호환성을 허용합니다. 해당 제품은 사용자 데이터를 암호화하지 않고 SERVER_ENCRYPT 인증 유형을 사용하여 연결할 수 있습니다. 새 인증 유형을 지원하는 제품은 이 유형을 사용해야 합니다. 이 인증 유형은 서버의 데이터베이스 관리자 구성 파일에서만 유효하며 CATALOG DATABASE 명령에서 사용할 경우 유효하지 않습니다.

GSSPLUGIN

서버가 GSS-API 플러그인을 사용하여 인증을 수행하도록 지정합니다. 클라이언트 인증을 지정하지 않을 경우, 서버는 *srvcon_gssplugin_list* 데이터베이스 관리 프로그램 구성 매개변수에 나열된 임의의 Kerberos 플러그인을 포함한 서버 지원 플러그인 목록을 클라이언트에 리턴합니다. 클라이언트는 목록에서 클라이언트 플러그인 디렉토리에 있는 첫 번째 플러그인을 선택합니다. 클라이언트가 목록에 있는 플러그인을 지원하지 않은 경우, Kerberos 인증 스킴(리턴된 경우)을 사용하여 인증됩니다. 클라이언트 인증이 GSSPLUGIN 인증 스킴인 경우, 클라이언트는 목록의 첫 번째 지원 플러그인을 사용하여 인증됩니다.

GSS_SERVER_ENCRYPT

서버가 플러그인 인증 또는 암호화된 서버 인증 스킴을 승인하도록 지정합니다. 클라이언트 인증이 플러그인을 통해 발생할 경우, 클라이언트는 서버 지원 플러그인의 첫 번째 클라이언트 지원 플러그인을 사용하여 인증됩니다.

클라이언트 인증을 지정하고 않고 내재된 연결을 수행 중인 경우(즉, 클라이언트가 연결 수행 시 사용자 ID 및 암호를 제공하지 않을 경우), 서버는 서버 지원 플러그인의 목록, Kerberos 인증 스킴(목록의 플러그인 중 하나가 Kerberos 기반일 경우) 및 암호화된 서버 인증 스킴을 리턴합니다. 클라이언트는 클라이언트 플러그인 디렉토리에 있는 첫 번째 지원 플러그인을 사용하여 인증됩니다. 클라이언트가 목록에 있는 임의의 플러그인을 지원하지 않을 경우, Kerberos 인증 스킴을 사용하여 인증됩니다. 클라이언트가 Kerberos 인증 스킴을 지원하지 않을 경우, 클라이언트는 암호화된 서버 인증 스킴을 사용하여 인증되며 암호가 누락된 경우 연결이 실패합니다. DB2 UDB 제공 Kerberos 플러그인이 운영 체제에 대하여 존재하거나 Kerberos 기반 플러그인이 *srvcon_gssplugin_list* 데이터베이스 관리 프로그램 구성 매개변수에 대하여 지정된 경우 클라이언트가 Kerberos 인증 스킴을 지원합니다.

클라이언트 인증을 지정하지 않고 명시적 연결을 수행 중인 경우(즉, 사용자 ID 및 암호를 모두 제공한 경우), 인증 유형은 SERVER_ENCRYPT입니다.

주:

1. 구성 파일 자체로의 액세스는 구성 파일의 정보에 의해 보호되므로, 인증 정보를 변경하려면 인스턴스로부터 부주의하게 사용자를 잠그지 마십시오. 다음 데이터베이스 관리 프로그램 구성 파일 매개변수는 다음 인스턴스로의 액세스를 제어합니다.
   - AUTHENTICATION *
   - SYSADM_GROUP *
   - TRUST_ALLCLNTS
   - TRUST_CLNTAUTH
   - SYSCTRL_GROUP
   - SYSMAINT_GROUP

   * 가장 중요한 두 매개변수를 나타내며 이 매개변수가 문제점을 일으킬 수 있습니다.

   이러한 상황이 발생하지 않도록 할 수 있는 방법이 있습니다. 우연히 DB2 UDB 시스템으로부터 자신을 잠그게 되면, 확실한 특권을 받은 로컬 운영 체제 보안 사용자를 사용하여 데이터베이스 관리 프로그램 구성 파일을 갱신하는 보통 DB2 UDB 보안 검사를 겹쳐쓸 수 있는 모든 플랫폼에서 사용 가능한 장애 안전 옵션이 있습니다. 이 사용자는 데이터베이스 관리 프로그램 구성 파일을 갱신하여 문제점을 수정하는 특권을 항상 가지고 있습니다. 그러나 이 보안 생략은 데이터베이스 관리 프로그램 구성 파일의 로컬 갱신으로 제한됩니다. 리모트로 또는 기타 DB2 UDB 명령에 대해 장애 안전 사용자를 사용할 수 없습니다. 이 특수 사용자는 다음과 같이 식별됩니다.
   - UNIX® 플랫폼: 인스턴스 소유자
   - NT 플랫폼: 로컬 『Administrators』 그룹에 속한 사용자
   - 기타 플랫폼: 기타 플랫폼에는 로컬 보안이 없으므로, 모든 사용자는 로컬 보안 검사를 전달합니다.

관련 개념:

- 252 페이지의 『리모트 클라이언트에 대한 인증 고려사항』
- 252 페이지의 『파티션된 데이터베이스 인증 고려사항』
- 423 페이지의 『Windows NT 및 Windows NT용 DB2 보안 소개』

관련 참조:

- *관리 안내서: 성능*의 『authentication - 인증 유형 구성 매개변수』
- *관리 안내서: 성능*의 『trust_allclnts - 모든 클라이언트 신뢰 구성 매개변수』
- *관리 안내서: 성능*의 『trust_clntauth - 트러스트된 클라이언트 인증 구성 매개변수』

리모트 클라이언트에 대한 인증 고려사항

리모트 액세스용 데이터베이스를 카탈로그화할 경우, 인증 유형은 데이터베이스 디렉토리 항목에 지정됩니다.

DB2® Connect를 사용하여 액세스하는 데이터베이스의 경우 값을 지정하지 않으면 SERVER 인증으로 간주됩니다.

인증 유형이 필요하지 않습니다. 인정 유형을 지정하지 않으면 디폴트값 SERVER_ENCRYPT가 사용됩니다. 그러나 서버가 SERVER_ENCRYPT를 지원하지 않으면, 클라이언트는 서버에서 지원하는 값을 사용하여 재시도하려고 합니다. 서버가 여러 인증 유형을 지원할 경우, 클라이언트는 이들 중에서 선택하지 않고 대신 오류를 리턴합니다. 올바른 인증 유형이 사용되도록 하기 위해서 오류가 리턴됩니다. 이 경우, 클라이언트는 지원되는 인증 유형을 사용하는 데이터베이스를 카탈로그화해야 합니다. 인증 유형을 지정하면, 지정된 값이 서버의 값과 일치하는 한 인증이 즉시 시작됩니다. 불일치가 발견되면, DB2 Universal Database™(DB2 UDB)가 복구를 시도합니다. 복구 작업은 차이를 조정하기 위해 보다 많은 흐름을 초래할 수 있으며, DB2 UDB가 복구에 실패할 경우에는 오류를 초래합니다. 불일치의 경우, 서버의 값은 올바른 것으로 간주합니다.

관련 개념:

• 246 페이지의『서버에 대한 인증 방법』

파티션된 데이터베이스 인증 고려사항

파티션된 데이터베이스에서 데이터베이스의 파티션마다 정의된 동일한 사용자 및 그룹 세트가 있어야 합니다. 정의가 동일하지 않으면, 사용자는 다른 파티션에 다른 사항을 실행하도록 권한이 부여됩니다. 모든 파티션에 걸쳐서 일관성이 권장됩니다.

관련 개념:

• 246 페이지의『서버에 대한 인증 방법』

Kerberos 인증 세부사항

DB2® Universal Database(DB2 UDB)는 AIX® 버전 5.2, Solaris 운영 환경 버전 8, Red Hat Enterprise Linux Server Advanced Server 2.1(32비트 Intel) 및 Windows® 2000 이상 운영 체제에 관한 Kerberos 인증 프로토콜에 대한 지원을 제공합니다.

Kerberos 지원은 서버 및 클라이언트 인증 플러그인 둘다로 사용되는『IBMkrb5』라는 GSS-API 보안 플러그인으로 제공됩니다. 라이브러리는 UNIX® 및 Linux의 경우

sqllib/security{32|64}/plugin/IBM/{client|server} 디렉토리에 위치하며 Windows의 경우 sqllib/security/plugin/IBM{client|server} 디렉토리에 위치합니다.

주: 64비트 Windows의 경우, 플러그인 라이브러리를 IBMkrb564.dll이라고 합니다. 추가로, UNIX 및 Linux 플러그인에 대한 실제 플러그인 소스 코드 IBMkrb5.C가 sqllib/samples/security/plugins 디렉토리에서 사용 가능합니다.

DB2 UDB에서 Kerberos 인증을 사용하려고 시도하기 전에 Kerberos 사용 및 구성에 대해 잘 이해하도록 적극 권장합니다.

Kerberos 설명 및 소개

Kerberos는 보안되지 않은 네트워크 환경에서 사용자를 안전하게 인증하기 위해 공유 비밀 키 시스템을 채택하는 써드 파티 네트워크 인증 프로토콜입니다. 1980년대 후반 MIT에서 처음으로 개발되었으며, IETF(Internet Engineering Task Force) RFC 1510에 의해 최신 개정판 Kerberos 5가 소개되었습니다. (IETF의 URL은 http://www.ieft.org입니다. RFC 1510에는 『Kerberos 네트워크 인증 서비스(V5)』라는 표제가 붙었습니다.) 응용프로그램 서버 및 클라이언트 간에 텍스트 사용자 ID 및 암호 쌍이 아닌 암호화된 티켓(Kerberos 키 분배 센터 또는 줄여서 KDC라고 하는 별도의 서버에서 제공하는)이 교환되는 3계층 시스템을 사용합니다. 이들 암호화된 서비스 티켓(*증명서*라고 함)의 수명은 유한하며 클라이언트 및 서버만이 이해할 수 있습니다. 이로써 누군가 티켓을 네트워크에서 가로챈 경우에도, 보안 위험이 줄어듭니다. 각각의 사용자 또는 Kerberos 용어인 *핵심부*는 KDC와 공유되는 개인용 암호화 키를 소유합니다. 집합적으로, KDC에 등록된 핵심부 및 컴퓨터 세트를 *범주*라고 합니다.

Kerberos의 주요 기능은 사용자가 Kerberos 범주 내에서 자원에 대한 자신의 ID를 한 번만 검증할 수 있도록 해주는 단일 사인온 환경을 허용하는 것입니다. 이는 DB2 UDB에 대해 작업할 때, 사용자 ID 또는 암호를 제공하지 않고 DB2 UDB 서버에 연결하거나 접속할 수 있음을 의미합니다. 또다른 이점은 핵심부용 중앙 저장소가 사용되므로 사용자 ID 관리가 단순화된다는 점입니다. 마지막으로, Kerberos는 클라이언트가 서버 ID의 유효성을 확인할 수 있게 해주는 상호 인증을 지원합니다.

Kerberos 설정

DB2 UDB 및 Kerberos 지원은 DB2 UDB 포함 전에 포함된 모든 머신에 설치되어 제대로 구성된 Kerberos 계층에 의존합니다. 여기에는 다음과 같은 요구사항이 포함되지만, 추가 요구사항이 있을 수도 있습니다.

1. 클라이언트와 서버 머신 및 핵심부가 동일한 범주 또는 트러스트된 범주(Windows 용어로 트러스트된 도메인)에 속해야 합니다.
2. 적절한 핵심부를 작성해야 합니다.
3. 해당하는 경우, 서버 키탭(keytab) 파일을 작성해야 합니다.

4. 포함된 모든 머신의 시스템 시계를 동기화해야 합니다(Kerberos는 일반적으로 5분의 시간 오차를 허용하며, 그렇지 않을 경우 증명서를 확보할 때 사전 인증 오류가 발생할 수도 있습니다).

Kerberos 설치 및 구성에 관한 세부사항은 설치된 Kerberos 제품과 함께 제공된 문서를 참조하십시오.

DB2 UDB의 유일한 관심사는 연결 중인 응용프로그램이 제공하는 증명서(즉, 인증)를 기반으로 Kerberos 보안 컨텍스트가 성공적으로 작성되었는지 여부입니다. 서명 또는 메시지 암호화 같은, 기타 Kerberos 기능은 사용되지 않습니다. 또한 사용 가능한 경우, 상호 인증이 지원됩니다.

키보드 전제조건은 다음과 같습니다.

- IBM® Network Authentication Service (NAS) Toolkit 1.3을 갖춘 AIX 버전 5.2
- SEAM(Sun Enterprise Authentication Mechanism)을 갖춘 Solaris 운영 환경 버전 8 및 IBM NAS Toolkit 1.3
- `krb5-libs` 및 `krb5-workstation` 파일 세트를 갖춘 Red Hat Enterprise Linux Advanced Server 2.1
- Windows 2000(이상 운영 체제) 서버

Kerberos 및 클라이언트 핵심부

핵심부는 2파트 또는 다중 파트 형식일 수 있습니다(즉, *name@REALM* 또는 *name/instance@REALM*).『name』파트는 권한 부여 ID(AUTHID) 맵핑에서 사용되므로, 이름이 DB2 UDB 이름 지정 규칙을 준수해야 합니다. 이는 이름이 최대 30자여야 하며 사용되는 문자의 선택사항에 관한 기존 제한사항을 준수해야 함을 의미합니다. (AUTHID 맵핑은 나중에 나오는 주제에서 설명됩니다.)

주: Windows는 Kerberos 핵심부를 도메인 사용자와 직접 연관시킵니다. 이는 Kerberos 인증이 도메인 또는 범주와 연관되지 않은 Windows 머신에서 사용 가능하지 않음을 의미합니다. 또한 Windows는 2파트 이름(즉, *name@domain*)만을 지원합니다.

핵심부 자체가 대상 데이터베이스에 서비스 티켓을 요청하고 수신할 수 있는 아웃바운드 증명서를 확보할 수 있어야 합니다. 이는 UNIX 또는 Linux에서 보통 **kinit** 명령으로 수행되며, Windows에서는 로그온 시 내재적으로 수행됩니다.

Kerberos 및 권한 부여 ID 맵핑

존재의 범위가 일반적으로 단일 머신으로 제한되는 운영 체제 사용자 ID와 달리(NIS 예외), Kerberos 핵심부는 자신의 소유가 아닌 범주에서 인증을 받을 수 있습니다. 중복된 핵심부 이름의 잠재적 문제점은 범주 이름을 사용하여 핵심부를 완전히 규정함으로써 방지됩니다. Kerberos에서 완전한 핵심부는 `name/instance@REALM` 양식을 취하

며 여기서, instance 필드는 실제로 『/』로 분리된 복수의 인스턴스(즉, name/instance1/instance2@REALM)이거나 전부 생략할 수도 있습니다. 명백한 제한사항은 범주 이름이 네트워크 내에 정의된 모든 범주 내에서 고유해야 한다는 점입니다. DB2 UDB에 대한 문제점은 핵심부에서 AUTHID로 단순 맵핑을 제공하려면, 핵심부 이름(즉, 완전한 핵심부의 『name』) 및 AUTHID 간에 일대일 맵핑이 바람직하다는 점입니다. AUTHID가 DB2 UDB에서 디폴트 스키마로 사용되며 쉽고 논리적으로 유도되어야 하므로 단순 맵핑이 필요합니다. 결과적으로, 데이터베이스 관리자는 다음과 같은 잠재적 문제점을 인식해야 합니다.

- 동일한 이름을 갖는 서로 다른 범주의 핵심부는 동일한 AUTHID로 맵핑됩니다.
- 동일한 이름을 갖지만 서로 다른 인스턴스의 핵심부는 동일한 AUTHID로 맵핑됩니다.

위의 사항을 고려할 때, 다음과 같은 권장사항이 작성됩니다.

- DB2 UDB 서버를 액세스할 모든 트러스트된 범주 내의 이름에 대하여 고유한 이름 스페이스를 유지보수하십시오.
- 인스턴스에 관계없이, 동일한 이름을 갖는 모든 핵심부는 동일한 사용자에게 속해야 합니다.

Kerberos 및 서버 핵심부

UNIX 또는 Linux에서 DB2 UDB 인스턴스에 대한 서버 핵심부 이름은 <instance name>/<fully qualified hostname>@REALM으로 가정됩니다. 이 핵심부는 Kerberos 보안 컨텍스트를 승인할 수 있어야 하며 *초기화* 수행 시 플러그인에 의해 서버 이름이 DB2 UDB로 보고되므로 DB2 UDB 인스턴스를 시작하기 전에 존재해야 합니다.

Windows에서 서버 핵심부는 DB2 UDB 서비스가 시작된 도메인 어카운트로 간주됩니다. 이에 대한 예외로 로컬 SYSTEM 어카운트에 의해 시작될 수도 있는 인스턴스가 있습니다. 이 경우, 서버 핵심부 이름은 host/<hostname>으로 보고됩니다. 이 이름은 클라이언트 및 서버가 둘다 Windows 도메인에 속할 경우에만 유효합니다.

Windows는 3파트 이름 이상을 지원하지 않습니다. 이로 인하여 Windows 클라이언트가 UNIX 서버에 연결하려고 할 때 문제점이 발생합니다. 결과적으로, UNIX Kerberos와의 상호 작동성이 필요한 경우 Windows 도메인에서 Windows 어카운트에 대한 Kerberos 핵심부 맵핑을 설정해야 할 수도 있습니다. (관련 지시사항은 해당하는 Microsoft® 문서를 참조하십시오.)

Kerberos 키탭(keytab) 파일

보안 컨텍스트 요청 승인을 원하는 UNIX 또는 Linux의 모든 Kerberos 서비스는 *키탭(keytab)* 파일에 증명서를 배치해야 합니다. 이는 DB2 UDB에 의해 서버 핵심부로 사용되는 핵심부에 적용됩니다. 디폴트 키탭 파일에서만 서버 키를 검색합니다. 키탭 파일에 키 추가에 관한 지시사항은 Kerberos 제품에 제공되는 문서를 참조하십시오.

Windows에는 키탭 파일의 개념이 없으며 시스템이 핵심부에 대한 인증서 핸들 저장 및 획득을 자동으로 처리합니다.

Kerberos 및 그룹

Kerberos는 그룹의 개념이 없는 인증 프로토콜입니다. 결과적으로 DB2 UDB는 로컬 운영 체제에 의존하여 Kerberos 핵심부에 대한 그룹 목록을 확보합니다. UNIX 또는 Linux의 경우, 각각의 핵심부에 대하여 동등한 시스템 어카운트가 있어야 합니다. 예를 들어, name@REALM 핵심부에 대하여, DB2 UDB는 운영 체제 사용자 *name*이 속하는 모든 그룹 이름을 로컬 운영 체제에서 조회하여 그룹 정보를 수집합니다. 운영 체제 사용자가 존재하지 않을 경우, AUTHID는 PUBLIC 그룹에만 속합니다. 한편 Windows는 도메인 어카운트를 Kerberos 핵심부에 자동으로 연관시키며 별도의 운영 체제 어카운트를 작성하기 위한 추가 단계가 필요하지 않습니다.

클라이언트에서 Kerberos 인증 사용 가능화

clnt_krb_plugin 데이터베이스 관리 프로그램 구성 매개변수를 사용 중인 Kerberos 플러그인의 이름으로 갱신해야 합니다. 지원되는 플랫폼에서는 이를 IBMkrb5로 설정합니다. 이 매개변수는 AUTHENTICATION 매개변수가 KERBEROS 또는 KRB_SERVER_ENCRYPT로 설정된 경우 연결 및 로컬 인스턴스 레벨 조치에 Kerberos를 사용할 수 있음을 DB2 UDB에 알립니다. 그렇지 않은 경우, 클라이언트 측 Kerberos 지원을 가정하지 않습니다.

주: Kerberos 지원이 사용 가능한지를 검증하는 점검은 수행되지 않습니다.

선택적으로 클라이언트에서 데이터베이스를 카탈로그화할 경우, 다음과 같이 인증 유형을 지정할 수 있습니다.

```
db2 catalog db testdb at node testnode authentication kerberos target
   principal service/host@REALM
```

그러나 인증 정보를 제공하지 않을 경우, 서버는 클라이언트에 서버 핵심부 이름을 전송합니다.

서버에서 Kerberos 인증 사용 가능화

srvcon_gssplugin_list 데이터베이스 관리 프로그램 구성 매개변수를 서버 Kerberos 플러그인 이름으로 갱신해야 합니다. 이 매개변수가 지원되는 플러그인 목록만을 포함할 경우에도, 하나의 Kerberos 플러그인만을 지정할 수 있습니다. 그러나 필드가 공백이고 AUTHENTICATION이 KERBEROS 또는 KRB_SERVER_ENCRYPT로 설정된 경우, 디폴트 Kerberos 플러그인(IBMkrb5)을 가정하고 사용합니다. Kerberos 인증을 사용할 경우 항상 또는 수신 연결에만 사용할 것인지 여부에 따라 AUTHENTICATION 또는 SVRCON_AUTH 매개변수를 KERBEROS 또는 KRB_SERVER_ENCRYPT로 설정해야 합니다.

Kerberos 플러그인 작성

Kerberos 플러그인 작성시 고려해야 할 다음과 같은 몇 가지 고려사항이 있습니다.

- 초기화 기능에서 DB2 UDB에 리턴되는 함수 포인터 배열의 *plugintype*을 DB2SEC_PLUGIN_TYPE_KERBEROS로 설정해야 하는 경우를 제외하고 Kerberos 플러그인을 GSS-API 플러그인으로 기록하십시오.
- 특정 조건에서 서버가 클라이언트에 서버 핵심부 이름을 보고할 수도 있습니다. 이와 같이 DRDA®는 핵심부 이름을 GSS_C_NT_USER_NAME 형식 (`server/host@REALM`)으로 선언하므로, 핵심부 이름을 GSS_C_NT_HOSTBASED_SERVICE 형식(`service@host`)으로 지정하지 않아야 합니다.
- 일반적인 상황에서 디폴트 키탭 파일은 KRB5_KTNAME 환경 변수로 지정할 수 있습니다. 그러나 서버 플러그인이 DB2 UDB 엔진 프로세스 내에서 실행되므로, 이 환경 변수를 액세스할 수 없습니다.

관련 개념:

- 246 페이지의 『서버에 대한 인증 방법』

특권, 권한 레벨 및 데이터베이스 권한

*특권*을 사용하여 사용자는 데이터베이스 자원을 작성하거나 액세스할 수 있습니다. *권한 레벨*은 특권은 물론 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작을 그룹화하는 메소드를 제공합니다. *데이터베이스 권한*을 사용하여 사용자는 데이터베이스 레벨에서 활동을 수행할 수 있습니다. 특권, 권한 레벨 및 데이터베이스 권한을 함께 사용하여 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 제어할 수 있습니다. 사용자는 필수 특권, 권한 레벨 또는 데이터베이스 권한을 보유하고 있는 오브젝트만 액세스할 수 있습니다. DB2® Universal Database(DB2 UDB)는 인증된 사용자에 대한 권한 점검을 수행할 때 이를 판별합니다.

데이터베이스 관리 프로그램은 각각의 사용자가 특정 태스크를 수행하기 위해 필요한 각 데이터베이스 함수를 사용할 수 있도록 명시적으로 또는 내재적으로 특정하게 권한 부여되어야 함을 요구합니다. *명시적* 권한 또는 특권은 사용자에게 권한 부여됩니다(데이터베이스 카탈로그에서 U의 GRANTEETYPE). *내재적* 권한 또는 특권은 사용자가 속하는 그룹에 권한 부여됩니다(데이터베이스 카탈로그에서 G의 GRANTEETYPE). 따라서 테이블을 작성하려면 사용자는 테이블 작성 권한을 부여받아야 하며 테이블을 변경하려면 사용자는 테이블 변경 권한을 부여받아야 합니다. 나머지도 마찬가지입니다.

258 페이지의 그림 3에서는 권한 및 이의 제어 범위간의 관계를 예시합니다(데이터베이스, 데이터베이스 관리 프로그램).

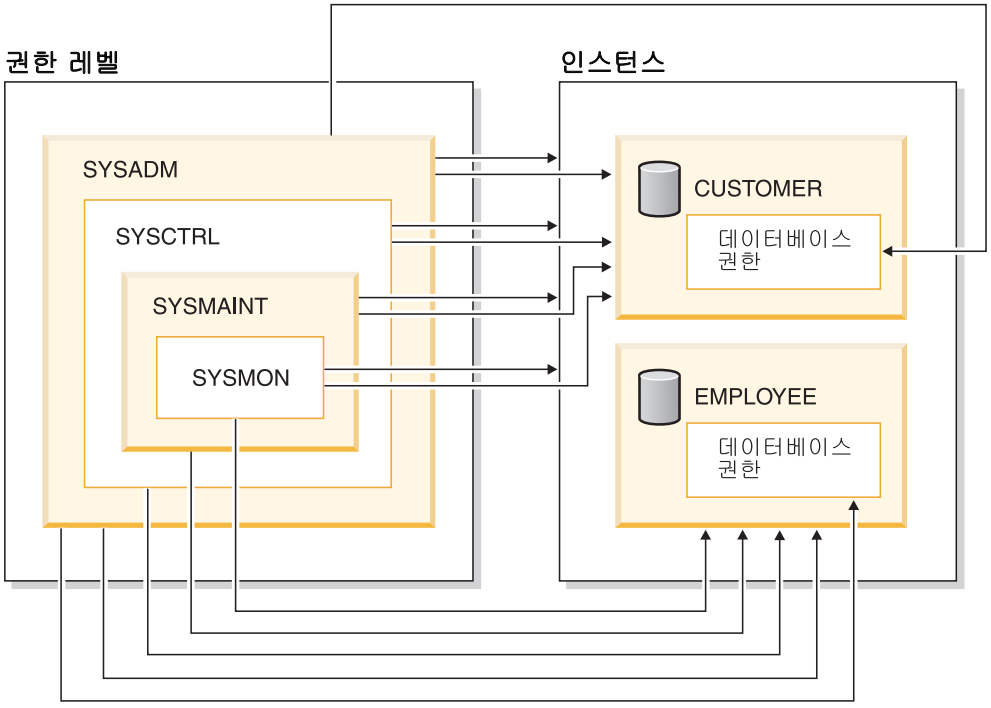


그림 3. 권한의 계층 구조

사용자 또는 그룹은 다음과 같은 권한 또는 특권 중 하나 이상을 보유할 수 있습니다.

- 관리자 권한:
  - SYSADM(시스템 관리자)

    SYSADM 권한 레벨은 데이터베이스 관리 프로그램에 의해 작성되고 유지보수되는 모든 자원에 대한 제어를 제공합니다. 시스템 관리자는 DBADM, SYSCTRL, SYSMAINT 및 SYSMON의 모든 권한을 보유하며 DBADM 권한을 부여하고 취소하는 권한을 보유합니다.

    SYSADM 권한이 있는 사용자는 데이터베이스 관리 프로그램을 제어하는 책임과 데이터의 무결성 및 안전성을 보장하는 책임을 담당하게 됩니다. SYSADM 권한은 데이터베이스의 모든 오브젝트에 대한 내재적 특권을 제공하며 데이터베이스 관리 프로그램을 액세스할 수 있는 사용자 및 이러한 액세스의 범위에 대한 제어를 제공합니다. SYSADM 권한에 대한 자세한 정보는 ″시스템 관리 권한(SYSADM)″을 참조하십시오.
  - DBADM(데이터베이스 관리자)

    DBADM 데이터베이스 권한은 단일 데이터베이스에 대한 관리자 권한을 제공합니다. 이 데이터베이스 관리자는 오브젝트의 작성, 데이터베이스 명령의 실행 및 테이블 데이터의 액세스에 필요한 특권을 보유합니다. 데이터베이스 관리자는 CONTROL 및 개별적 특권을 부여하거나 취소할 수도 있습니다. DBADM 권한에 대한 자세한 정보는 ″데이터베이스 관리 권한(DBADM)″을 참조하십시오.

- 시스템 제어 권한:
  - SYSCTRL(시스템 제어)

    SYSCTRL 권한 레벨은 시스템 자원에 영향을 주는 조작에 대한 제어를 제공합니다. 예를 들어, SYSCTRL 권한이 있는 사용자는 데이터베이스를 작성, 갱신, 중지하거나 삭제할 수 있습니다. 또한 이 사용자는 인스턴스를 중지할 수도 있지만 테이블 데이터를 액세스할 수는 없습니다. SYSCTRL 권한이 있는 사용자는 SYSMON 권한을 보유할 수도 있습니다. SYSCTRL 권한에 대한 자세한 정보는 ″시스템 제어 권한(SYSCTRL)″을 참조하십시오.
  - SYSMAINT(시스템 유지보수)

    SYSMAINT 권한 레벨은 인스턴스와 연관된 모든 데이터베이스에서 유지보수 조작을 수행하기 위해 필요한 권한을 제공합니다. SYSMAINT 권한이 있는 사용자는 데이터베이스 구성을 갱신하고, 데이터베이스 또는 테이블 스페이스를 백업하며, 기존 데이터베이스를 복원하고, 데이터베이스를 모니터할 수 있습니다. SYSCTRL과 마찬가지로, SYSMAINT는 테이블 데이터에 대한 액세스를 제공하지 않습니다. SYSMAINT 권한이 있는 사용자는 SYSMON 권한을 보유할 수도 있습니다. SYSMAINT 권한에 대한 자세한 정보는 ″시스템 유지보수 권한(SYSMAINT)″을 참조하십시오.
- SYSMON(시스템 모니터 권한)

  SYSMON 권한 레벨은 데이터베이스 시스템 모니터를 사용하기 위해 필요한 권한을 제공합니다. SYSMON 권한에 대한 자세한 정보는 ″시스템 모니터 권한(SYSMON)″을 참조하십시오.
- 데이터베이스 권한

  테이블 또는 루틴의 작성과 같은 활동을 수행하거나 데이터를 테이블로 로드하려면 특정 데이터베이스 권한이 필요합니다. 자세한 정보는 ″데이터베이스 권한″을 참조하십시오.
- 특권:

  특권은 데이터베이스 오브젝트에 대한 활동을 수행하기 위해 필요합니다(예: 인덱스 작성 및 삭제). 특권은 사용자가 수행할 수 있는 태스크를 엄격하게 정의합니다. 예를 들어, 사용자가 테이블에서 인덱스를 작성하는 특권을 보유할 수 있지만 동일 테이블에서 트리거를 작성하는 특권은 보유할 수 없습니다.
  - CONTROL 특권

    오브젝트에서 CONTROL 특권을 보유하는 경우, 사용자는 해당 데이터베이스 오브젝트를 액세스할 수 있으며 해당 오브젝트에서 기타 사용자에게 특권을 부여하거나 기타 사용자로부터 특권을 취소할 수 있습니다.

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

다른 사용자가 해당 오브젝트에 대해 CONTROL 특권을 필요로 하는 경우, SYSADM 또는 DBADM 권한이 있는 사용자는 해당 오브젝트에 대한 CONTROL 권한을 부여해야 합니다. 오브젝트 소유자에게서 CONTROL 특권을 취소할 수는 없습니다.

일부 경우에 오브젝트의 작성자는 자동으로 해당 오브젝트에 대한 CONTROL 특권을 확보합니다. 자세한 정보는 "오브젝트 작성, 소유권 및 특권"을 참조하십시오.

- 개별적 특권을 부여하여 사용자가 특정 오브젝트에서 특정 태스크를 수행하도록 할 수 있습니다.

  관리자 권한(SYSADM 또는 DBADM) 또는 CONTROL 특권이 있는 사용자는 사용자에게 권한을 부여하거나 사용자로부터 권한을 취소할 수 있습니다.

  개별적 특권 및 데이터베이스 권한은 세부 함수를 허용하지만 기타 사용자에게 동일한 특권이나 권한을 부여하는 권한이 포함되지는 않습니다. 다른 사용자에게 테이블, 뷰, 스키마, 패키지, 루틴 및 시퀀스 특권을 부여하는 권한은 GRANT 명령문에서 WITH GRANT OPTION을 통해 다른 사용자에게 확장될 수 있습니다. 그러나 WITH GRANT OPTION은 일단 부여된 권한을 취소하기 위한 특권을 사용자가 부여할 수 있도록 허용하지는 않습니다. 권한을 취소하려면 SYSADM 권한, DBADM 권한 또는 CONTROL 권한이 있어야 합니다.

  특권을 PUBLIC에 부여할 수도 있습니다. PUBLIC 특권은 모든 사용자(권한 부여 이름)에게 적용되며 여기에는 향후의 모든 사용자가 포함됩니다. 임의의 개별적 사용자가 이전에 특권이 부여되었는지 여부와는 상관이 없습니다.
- 내재적 특권은 패키지를 실행할 수 있는 특권이 있는 사용자에게 부여될 수 있습니다. 사용자가 응용프로그램을 실행할 수 있는 동안, 사용자는 패키지 내에서 사용되는 데이터 오브젝트에 대한 명시적 특권을 반드시 필요로 하지는 않습니다.

사용자 또는 그룹은 개별적 특권 또는 권한의 임의의 조합에 대해 권한 부여될 수 있습니다. 특권이 오브젝트와 연관되는 경우에는 해당 오브젝트가 존재해야 합니다. 예를 들어, 테이블이 이전에 작성되어 있지 않은 경우에는 사용자에게 해당 테이블에 대한 SELECT 특권을 부여할 수 없습니다.

주: 권한 부여 이름에 특권 및 권한이 부여되지만 이 권한 부여 이름으로 작성된 사용자가 없으면 주의가 필요합니다. 일정 시간이 지나면 이 권한 부여 이름으로 사용자가 작성될 수 있으며 이 권한 부여 이름과 연관된 모든 권한 및 특권을 자동으로 받을 수 있습니다.

REVOKE 명령문은 이전에 권한 부여된 특권을 취소하기 위해 사용됩니다. DB2 UDB에서, 권한 부여 이름에서 특권을 취소하면 모든 권한 부여 이름에 의해 부여된 특권이 취소됩니다.

권한 부여 이름으로부터 특권을 취소해도 이 권한 부여 이름에 의해 특권이 부여된 임의의 기타 권한 부여 이름으로부터 동일한 특권이 취소되지는 않습니다. 예를 들어, CLAIRE가 SELECT WITH GRANT OPTION을 RICK에게 권한 부여한 후에 RICK이 SELECT를 BOBBY 및 CHRIS에게 권한 부여한다고 가정합니다. CLAIRE가 SELECT 특권을 RICK에게서 취소해도 BOBBY 및 CHRIS는 SELECT 특권을 계속 보유합니다.

관련 개념:

- 263 페이지의 『시스템 관리 권한(SYSADM)』
- 264 페이지의 『시스템 제어 권한(SYSCTRL)』
- 265 페이지의 『시스템 유지보수 권한(SYSMAINT)』
- 266 페이지의 『데이터베이스 관리 권한(DBADM)』
- 267 페이지의 『LOAD 권한』
- 268 페이지의 『데이터베이스 권한』
- 271 페이지의 『스키마 특권』
- 273 페이지의 『테이블 스페이스 특권』
- 273 페이지의 『테이블 및 뷰 특권』
- 276 페이지의 『패키지 특권』
- 277 페이지의 『인덱스 특권』
- 277 페이지의 『시퀀스 특권』
- 278 페이지의 『데이터베이스 오브젝트에 대한 액세스 제어』
- 283 페이지의 『패키지를 통한 간접 권한』
- 278 페이지의 『루틴 특권』
- 261 페이지의 『오브젝트 작성, 소유권 및 특권』
- 267 페이지의 『시스템 모니터 권한(SYSMON)』

오브젝트 작성, 소유권 및 특권

오브젝트가 작성되면, 하나의 권한 부여 이름이 오브젝트의 *소유권*에 지정됩니다. 소유권은 사용자가 SQL문의 오브젝트를 참조할 수 있는 권한이 있음을 의미합니다.

오브젝트를 스키마 내에서 작성할 경우, 명령문의 권한 부여 ID에는 내재적 또는 명시적으로 지정된 스키마에서 오브젝트를 작성하는 데 필요한 특권이 있습니다. 즉, 권한 부여 이름은 스키마의 소유자이거나 스키마에 관한 CREATEIN 특권을 소유해야 합니다.

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

오브젝트 작성시, 명령문의 권한 부여 ID는 해당 오브젝트의 소유자입니다.

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

예를 들어, `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C! INT)`문은 `SCOTTSTUFF` 스키마 및 `SCOTTSTUFF.T1` 테이블을 작성하며, 둘 다 SCOTT에 의해 소유됩니다. BOBBY 사용자에게 `SCOTTSTUFF` 스키마에 관한 CREATEIN 특권이 부여되었고 `SCOTTSTUFF.T1` 테이블에 관한 인덱스를 작성한다고 가정하십시오. 인덱스는 스키마 이후에 작성되므로, BOBBY는 `SCOTTSTUFF.T1`에 관한 인덱스를 소유합니다.

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

- CONTROL 특권은 새로 작성된 테이블, 인덱스 및 패키지에 내재적으로 부여됩니다. 이 특권을 사용하여 오브젝트 작성자는 데이터베이스 오브젝트를 액세스하고, 해당 오브젝트에 관하여 다른 사용자에게/로부터 특권을 부여하고 취소할 수 있습니다. 서로 다른 사용자가 해당 오브젝트에 대하여 CONTROL 특권을 요구할 경우, SYSADM 또는 DBADM 권한이 있는 사용자는 CONTROL 특권을 해당 오브젝트에 부여해야 합니다. CONTROL 특권을 해당 오브젝트 소유자가 취소할 수 없습니다.
- 오브젝트 소유자에게 뷰 정의에서 참조하는 모든 테이블, 뷰 및 별칭에 관한 CONTROL 특권이 있을 경우 CONTROL 특권이 새로 작성된 뷰에 내재적으로 부여됩니다.
- 트리거, 루틴, 시퀀스, 테이블 스페이스 및 버퍼 풀과 같은 기타 오브젝트에는 연관된 CONTROL 특권이 없습니다. 그러나 오브젝트 소유자는 오브젝트와 연관된 각각의 특권을 자동으로 수신합니다(또한 지원되는 경우 GRANT문의 WITH GRANT 옵션을 사용하여 다른 사용자에게 이러한 특권을 제공할 수 있습니다). 추가로 오브

젝트 소유자는 오브젝트를 변경하거나, 오브젝트에 관한 주석을 추가하거나 또는 삭제할 수 있습니다. 이러한 권한 부여는 오브젝트 소유자에 대하여 내재적이며 취소할 수 없습니다.

관련 개념:

- 257 페이지의『특권, 권한 레벨 및 데이터베이스 권한』
- 271 페이지의『스키마 특권』
- 273 페이지의『테이블 스페이스 특권』
- 273 페이지의『테이블 및 뷰 특권』
- 276 페이지의『패키지 특권』
- 277 페이지의『인덱스 특권』
- 277 페이지의『시퀀스 특권』
- 278 페이지의『루틴 특권』

특권, 권한 및 권한 부여에 대한 세부사항

이 섹션에서 각 권한에 대해 설명합니다. 권한 다음에는 서로 다른 특권이 나옵니다.

시스템 관리 권한(SYSADM)

SYSADM 권한 레벨은 최상위 레벨의 관리 권한입니다. SYSADM 권한을 갖는 사용자는 유틸리티를 실행하고, 데이터베이스 및 데이터베이스 관리 프로그램 명령을 발행하며, 데이터베이스 관리 인스턴스 내의 데이터베이스에 있는 테이블의 데이터에 액세스할 수 있습니다. 데이터베이스, 테이블, 뷰, 인덱스, 패키지, 스키마, 서버, 별명, 데이터 유형, 함수, 프로시저, 트리거, 테이블 스페이스, 데이터베이스 파티션 그룹, 버퍼 풀 및 이벤트 모니터를 비롯하여 인스턴스에 있는 모든 데이터베이스 오브젝트를 제어할 수 있는 능력을 제공합니다.

SYSADM 권한은 *sysadm_group* 구성 매개변수로 지정한 그룹에 지정됩니다. 해당 그룹의 멤버십은 사용자 플랫폼에 사용된 보안 기능을 통해 데이터베이스 관리 프로그램 외부에서 제어됩니다.

SYSADM 권한을 가진 사용자만이 다음 기능을 수행할 수 있습니다.

- 데이터베이스 이주
- 데이터베이스 관리 프로그램 구성 파일 변경(SYSCTRL, SYSMAINT 또는 SYSMON 권한을 갖는 그룹 지정 포함)
- DBADM 권한 부여

주: SYSADM 권한을 갖는 사용자가 데이터베이스를 작성하면, 해당 사용자에게 데이터베이스에 관한 명시적 DBADM 권한이 자동으로 부여됩니다. 데이터베이스 작

성자가 SYSADM 그룹에서 제거되고 해당 사용자가 해당 데이터베이스를 DBADM으로 액세스하지 못하도록 하고자 할 경우, 사용자의 DBADM 권한을 명시적으로 취소해야 합니다.

관련 개념:

- 264 페이지의 『시스템 제어 권한(SYSCTRL)』
- 265 페이지의 『시스템 유지보수 권한(SYSMAINT)』
- 288 페이지의 『데이터 암호화』
- 267 페이지의 『시스템 모니터 권한(SYSMON)』

시스템 제어 권한(SYSCTRL)

SYSCTRL 권한은 최상위 레벨의 시스템 제어 권한입니다. 이 권한은 데이터베이스 관리 프로그램 인스턴스와 데이터베이스에 대해 유지보수 및 유틸리티 조작을 수행할 수 있는 기능을 제공합니다. 이러한 조작은 시스템 자원에 영향을 미칠 수는 있으나, 데이터베이스 내의 데이터에 직접 액세스하지는 못합니다. 시스템 제어 권한은 사용자가 중요한 데이터가 들어 있는 데이터베이스 관리 프로그램 인스턴스를 관리하기 위한 목적으로 설계되었습니다.

SYSCTRL 권한은 *sysctrl_group* 구성 매개변수로 지정한 그룹에 지정됩니다. 그룹이 지정되면, 해당 그룹 내의 멤버십은 사용자의 플랫폼에서 사용되는 보안 기능을 통해 데이터베이스 관리 외부에서 제어됩니다.

SYSCTRL 권한을 가진 사용자만이 다음을 수행할 수 있습니다.

- 데이터베이스, 노드 또는 분산 연결 서비스(DCS) 디렉토리 갱신
- 사용자가 시스템을 강제로 끄도록 함
- 데이터베이스의 작성 또는 제거
- 테이블 스페이스의 제거, 작성 또는 변경
- 새로운 데이터베이스로 리스토어

추가로 SYSCTRL 권한이 있는 사용자는 시스템 유지보수 권한(SYSMAINT) 및 시스템 모니터 권한(SYSMON)이 있는 사용자의 기능을 수행할 수 있습니다.

SYSCTRL 권한을 갖는 사용자는 데이터베이스에 연결할 수 있는 내재적 특권도 갖게 됩니다.

주: SYSCTRL 권한을 가지고 있는 사용자가 데이터베이스를 작성할 때, 데이터베이스에 대한 명시적인 DBADM 권한이 사용자에게 자동으로 권한 부여됩니다. SYSCTRL 그룹에서 데이터베이스 작성자가 제거되고 그룹이 해당 데이터베이스에 DBADM으로 액세스하지 못하도록 하고자 할 경우, 명시적으로 이 DBADM 권한을 권한 취소하십시오.

관련 개념:

- 265 페이지의 『시스템 유지보수 권한(SYSMAINT)』
- 266 페이지의 『데이터베이스 관리 권한(DBADM)』
- 267 페이지의 『시스템 모니터 권한(SYSMON)』

시스템 유지보수 권한(SYSMAINT)

SYSMAINT 권한은 시스템 제어 권한 중 두 번째로 높은 레벨입니다. 이 권한은 데이터베이스 관리 인스턴스와 데이터베이스에 대해 유지보수 및 유틸리티 조작을 수행할 수 있는 기능을 제공합니다. 이러한 조작은 시스템 자원에 영향을 미칠 수는 있으나, 데이터베이스 내의 데이터에 직접 액세스하지는 못합니다. 시스템 유지보수 권한은 사용자가 중요한 데이터가 들어 있는 데이터베이스 관리 인스턴스 내의 데이터베이스를 유지보수하기 위한 목적으로 설계되었습니다.

SYSMAINT 권한은 *sysmaint_group* 구성 매개변수로 지정한 그룹에 지정됩니다. 그룹이 지정되면, 해당 그룹 내의 멤버십은 사용자의 플랫폼에서 사용되는 보안 기능을 통해 데이터베이스 관리 외부에서 제어됩니다.

SYSMAINT 이상의 시스템 권한을 가진 사용자만이 다음을 수행할 수 있습니다.

- 데이터베이스 구성 파일 갱신
- 데이터베이스 또는 테이블 스페이스 백업
- 기존의 데이터베이스로 리스토어
- 롤 포워드 복구 수행
- 인스턴스 시작 또는 중지
- 테이블 스페이스 리스토어
- 추적 실행
- 데이터베이스 관리 인스턴스 또는 해당 데이터베이스의 데이터베이스 시스템 모니터 스냅샷

SYSMAINT, DBADM 이상의 권한을 가진 사용자만이 다음을 수행할 수 있습니다.

- 테이블 스페이스의 상태 쿼리
- 로그 실행기록 파일 갱신
- 테이블 스페이스 Quiesce
- 테이블 재구성
- **RUNSTATS** 유틸리티를 사용하여 카탈로그 통계 수집

SYSMAINT 권한을 갖는 사용자는 데이터베이스에 연결할 수 있는 내재적 특권도 갖게 되며, 시스템 모니터 권한(SYSMON)을 갖는 사용자의 기능을 수행할 수 있습니다.

관련 개념:

- 266 페이지의 『데이터베이스 관리 권한(DBADM)』
- 267 페이지의 『시스템 모니터 권한(SYSMON)』

데이터베이스 관리 권한(DBADM)

DBADM 권한은 관리 권한 중 두 번째로 높은 레벨입니다. 이것은 특정 데이터베이스에만 적용되며, 사용자가 특정 유틸리티를 실행하고, 데이터베이스 명령을 발행하고, 데이터베이스 테이블의 데이터에 액세스할 수 있도록 합니다. DBADM 권한이 부여되면, BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, QUIESCE_CONNECT 및 LOAD 데이터베이스 권한도 부여됩니다. SYSADM 권한을 가지고 있는 사용자만이 DBADM 권한을 권한 부여하거나 권한 취소할 수 있습니다. DBADM 권한을 가지고 있는 사용자는 다른 사용자에게 데이터베이스에 대한 특권을 권한 부여할 수 있고, 누가 특권을 권한 부여했는지에 관계없이 사용자에게서 임의의 특권을 권한 취소할 수 있습니다.

DBADM 이상의 권한을 가진 사용자만이 다음을 수행할 수 있습니다.

- 로그 파일 읽기
- 이벤트 모니터 작성, 활성화 및 제거

DBADM, SYSMAINT 이상의 권한을 가진 사용자는 다음을 수행할 수 있습니다.

- 테이블 스페이스의 상태 쿼리
- 로그 실행기록 파일 갱신
- 테이블 스페이스 Quiesce
- 테이블 재구성
- **RUNSTATS** 유틸리티를 사용하여 카탈로그 통계 수집

주: DBADM은 DBADM 권한이 보유된 데이터베이스에 대해서만 위의 기능을 수행할 수 있습니다.

관련 개념:

- 263 페이지의 『시스템 관리 권한(SYSADM)』
- 264 페이지의 『시스템 제어 권한(SYSCTRL)』
- 265 페이지의 『시스템 유지보수 권한(SYSMAINT)』
- 267 페이지의 『LOAD 권한』
- 268 페이지의 『데이터베이스 권한』
- 270 페이지의 『내재된 스키마 권한(IMPLICIT_SCHEMA) 고려사항』

시스템 모니터 권한(SYSMON)

SYSMON 권한은 데이터베이스 관리 프로그램 인스턴스 또는 해당 데이터베이스의 데이터베이스 시스템 모니터 스냅샷을 취하는 기능을 제공합니다. SYSMON 권한은 *sysmon_group* 구성 매개변수로 지정한 그룹에 지정됩니다. 그룹이 지정되면, 해당 그룹 내의 멤버십은 사용자의 플랫폼에서 사용되는 보안 기능을 통해 데이터베이스 관리 외부에서 제어됩니다.

SYSMON 권한을 사용하여 사용자는 다음과 같은 명령을 실행할 수 있습니다.

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

SYSMON 권한을 사용하여 사용자는 다음과 같은 API를 사용할 수 있습니다.

- db2GetSnapshot - 스냅샷 가져오기
- db2GetSnapshotSize - db2GetSnapshot() 출력 버퍼에 필요한 크기 측정
- db2MonitorSwitches - 모니터 스위치 가져오기/갱신
- db2ResetMonitor - 모니터 재설정

SYSMON 권한을 사용하여 사용자는 다음과 같은 SQL 테이블 함수를 사용할 수 있습니다.

- 이전에 SYSPROC.SNAPSHOT_FILEW를 실행하지 않은 모든 스냅샷 테이블 함수

  SYSPROC.SNAPSHOT_FILEW는 스냅샷을 취하여 내용을 파일에 저장합니다. 널(NULL) 입력 매개변수를 사용하여 스냅샷 테이블 함수를 호출할 경우, 실시간 시스템 스냅샷 대신 파일 내용이 리턴됩니다.

SYSADM, SYSCTRL 또는 SYSMAINT 권한 레벨을 갖는 사용자는 SYSMON 권한도 소유합니다.

관련 참조:

- *관리 안내서: 성능*의 『sysmon_group - 시스템 모니터 권한 그룹 이름 구성 매개변수』

LOAD 권한

테이블에 대한 INSERT 특권 및 데이터베이스 레벨의 LOAD 권한이 있는 사용자는 **LOAD** 명령을 사용하여 테이블에 데이터를 로드할 수 있습니다.

테이블에 대한 INSERT 특권 및 데이터베이스 레벨의 LOAD 권한이 있는 사용자는 이전 로드 조작이 데이터를 삽입하는 로드인 경우 **LOAD RESTART** 또는 **LOAD TERMINATE**를 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 바꾸기였던 경우에는 해당 사용자에게 DELETE 특권도 부여해야만 사용자가 **LOAD RESTART** 또는 **LOAD TERMINATE**를 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.

관련 개념:

- *데이터 이동 유틸리티 안내 및 참조서*의 『로드를 사용하는 데 필요한 특권, 권한 및 권한 부여』
- 273 페이지의 『테이블 및 뷰 특권』

관련 참조:

- *Command Reference*의 『RUNSTATS Command』
- *Command Reference*의 『QUIESCE TABLESPACES FOR TABLE Command』
- *Command Reference*의 『LIST TABLESPACES Command』
- *Command Reference*의 『LOAD Command』

데이터베이스 권한

269 페이지의 그림 4에서는 데이터베이스 권한을 보여줍니다.

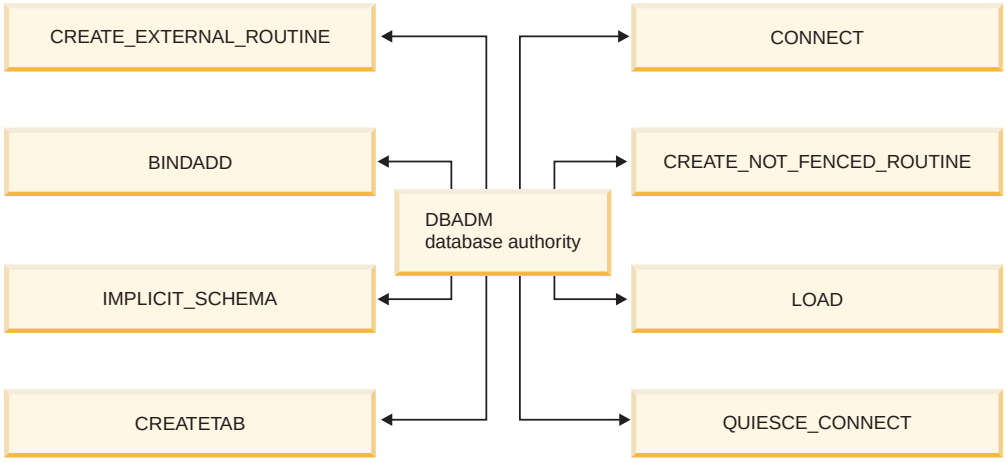


그림 4. 데이터베이스 권한

데이터베이스 권한에는 데이터베이스에 대한 조치가 전체적으로 포함됩니다. DBADM 권한이 있는 사용자는 다음과 같은 전체 데이터베이스 권한 세트를 소유합니다.

- CONNECT는 사용자가 데이터베이스에 액세스할 수 있도록 합니다.
- BINDADD는 사용자가 데이터베이스에 새로운 패키지를 작성할 수 있도록 합니다.
- CREATETAB는 사용자가 데이터베이스에 새로운 테이블을 작성할 수 있도록 합니다.
- CREATE_EXTERNAL_ROUTINE은 사용자가 응용프로그램 및 데이터베이스의 다른 사용자가 사용할 프로시저를 작성할 수 있도록 합니다.
- CREATE_NOT_FENCED_ROUTINE에서 사용자는 사용자 정의 함수(UDF) 또는 『비분리』 프로시저를 작성할 수 있습니다. 『비분리』 UDF 또는 프로시저는 데이터베이스 관리가 이러한 UDF 또는 프로시저로부터의 스토리지 또는 제어 블록을 보호하지 않기 때문에 세심하게 테스트되어야 합니다(결과적으로, 『비분리』를 실행하도록 허용된 UDF 또는 프로시저가 완벽하게 작성되어 있지 않거나 테스트되지 않으면, 시스템에 심각한 문제가 발생할 수 있습니다).

  주: CREATE_EXTERNAL_ROUTINE은 CREATE_NOT_FENCED_ROUTINE이 부여된 모든 사용자에게 자동으로 부여됩니다.
- IMPLICIT_SCHEMA는 모든 사용자가 아직 존재하지 않는 스키마 이름과 함께 CREATE문을 사용하여 오브젝트를 작성함으로써 스키마를 내재적으로 작성할 수 있도록 합니다. SYSIBM은 내재적으로 작성된 스키마의 소유자가 되며, PUBLIC에는 이 스키마에서 오브젝트를 작성할 특권이 부여됩니다.
- LOAD는 사용자가 데이터를 테이블로 로드하게 합니다.
- QUIESCE_CONNECT는 사용자가 Quiesce 상태에 있는 데이터베이스에 액세스할 수 있도록 합니다.

SYSADM 또는 DBADM 권한을 가지고 있는 사용자만이 다른 사용자에게 이러한 데이터베이스 권한을 권한 부여하고 권한 취소할 수 있습니다.

주: 데이터베이스가 작성되면, 다음 데이터베이스 권한이 자동으로 PUBLIC에 권한 부여됩니다.

- CREATETAB 데이터베이스 권한
- BINDADD 데이터베이스 권한
- CONNECT 데이터베이스 권한
- IMPLICIT_SCHEMA 데이터베이스 권한
- USERSPACE1 테이블 스페이스에서의 USE 특권
- 시스템 카탈로그 뷰에 대한 SELECT 특권

어떤 데이터베이스 권한을 제거하려면, DBADM 또는 SYSADM은 PUBLIC으로부터 명시적으로 데이터베이스 권한을 권한 취소하십시오.

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

내재된 스키마 권한(IMPLICIT_SCHEMA) 고려사항

새 데이터베이스가 작성되면, PUBLIC에 IMPLICIT_SCHEMA 데이터베이스 권한이 부여됩니다. 이 권한이 있는 모든 사용자는 오브젝트를 작성하고 아직 존재하지 않는 스키마 이름을 지정함으로써 스키마를 작성할 수 있습니다. SYSIBM은 내재적으로 작성된 스키마의 소유자가 되며, PUBLIC에는 이 스키마에서 오브젝트를 작성할 특권이 부여됩니다.

데이터베이스에서 스키마 오브젝트를 내재적으로 작성할 수 있는 사용자를 제어할 필요가 있을 경우, IMPLICIT_SCHEMA 데이터베이스 권한을 PUBLIC에서 권한 취소하십시오. 일단 이렇게 되면, 스키마 오브젝트를 작성할 수 있는 방법은 다음 세 가지 밖에 없습니다.

- 모든 사용자는 CREATE SCHEMA문에 자신의 권한 부여 이름을 사용하여 스키마를 작성할 수 있습니다.
- DBADM 권한을 갖는 모든 사용자는 아직 존재하지 않는 모든 스키마를 명시적으로 작성할 수 있으며, 또다른 사용자를 스키마 소유자로 선택적으로 지정할 수 있습니다.
- DBADM 권한이 있는 모든 사용자는 IMPLICIT_SCHEMA 데이터베이스 권한이 있어(PUBLIC과 별도로), 다른 데이터베이스 오브젝트를 작성할 때 어떠한 이름으로

도 스키마를 내재적으로 작성할 수 있습니다. SYSIBM은 내재적으로 작성된 스키마의 소유자가 되며, PUBLIC에는 스키마에서 오브젝트를 작성할 수 있는 특권이 부여됩니다.

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

스키마 특권

스키마 특권은 오브젝트 특권 범주 내에 있습니다. 오브젝트 특권은 272 페이지의 그림 5에 나와 있습니다.

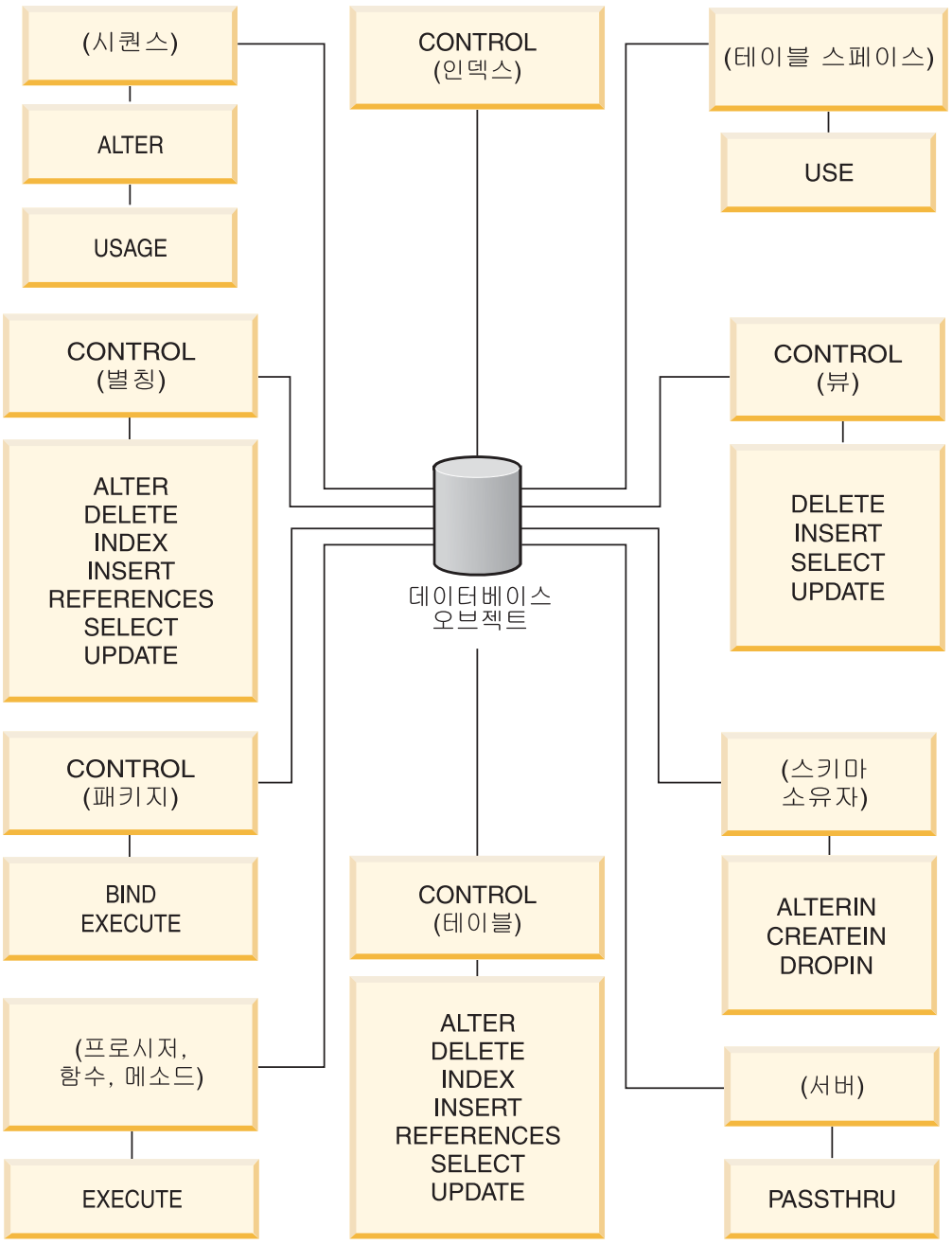


그림 5. 오브젝트 특권

스키마 권한은 데이터베이스의 스키마에 조치를 취할 수 있는 권한입니다. 사용자에게는 다음 권한이 권한 부여됩니다.

- CREATEIN은 사용자가 스키마 내에서 오브젝트를 작성하도록 합니다.
- ALTERIN은 사용자가 스키마 내에서 오브젝트를 변경하도록 합니다.
- DROPIN은 사용자가 스키마 내에서 오브젝트를 제거하도록 합니다.

스키마 소유자는 모두 이러한 특권과 기능을 다른 사용자에게 권한 부여할 권한을 가집니다. 스키마 오브젝트 내에서 조정된 오브젝트에는 테이블, 뷰, 인덱스, 패키지, 데이터 유형, 함수, 트리거, 프로시저, 별명이 있습니다.

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER SEQUENCE문』

테이블 스페이스 특권

테이블 스페이스 특권은 데이터베이스에서 테이블 스페이스에 대한 조치를 포함합니다. 사용자는 테이블 스페이스 내에서 테이블을 작성하도록 허용하는 테이블 스페이스에 대한 USE 특권을 부여할 수 있습니다.

테이블 스페이스의 소유자는 보통 SYSADM 또는 SYSCTRL 권한을 가진 작성자로 USE 특권과 이 특권을 다른 사용자에게 부여하는 능력을 갖습니다. 디폴트로, 데이터베이스 작성 시간에 테이블 스페이스 USERSPACE1에 대한 USE 특권이 PUBLIC으로 부여되지만, 이 특권은 권한 취소될 수 있습니다.

USE 특권은 SYSCATSPACE 또는 임의의 시스템 임시 테이블 스페이스에서 사용될 수 없습니다.

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

관련 참조:

- *SQL 참조서, 볼륨 2*의『CREATE TABLE문』

테이블 및 뷰 특권

테이블 및 뷰 특권에는 데이터베이스의 테이블 또는 뷰에 대한 조치가 포함됩니다. 사용자는 데이터베이스에 대해 CONNECT 권한을 가져야 다음과 같은 특권을 사용합니다.

- CONTROL은 테이블 또는 뷰를 제거할 권한을 포함하여 테이블 또는 뷰에 대한 모든 특권을 제공하고, 개별 테이블 특권을 권한 부여하거나 권한 취소합니다. CONTROL을 권한 부여하려면 SYSADM 또는 DBADM 권한이 있어야 합니다. 테이블의 작성자는 테이블에 대해 자동으로 CONTROL 특권을 가지게 됩니다. 뷰의

작성자는 뷰 정의에서 언급된 모든 테이블, 뷰 및 별칭에 대해 작성자가 CONTROL 특권을 가지고 있거나 SYSADM 또는 DBADM 권한을 가지고 있는 경우에만, 자동으로 CONTROL 특권을 부여받습니다.

- ALTER는 사용자가 테이블을 수정할 수 있도록 합니다(예를 들어, 테이블에 컬럼 또는 고유한 제한사항 추가). ALTER 특권을 갖는 사용자는 또한 테이블 또는 테이블 컬럼에서 COMMENT ON을 수행할 수도 있습니다. 테이블에서 수행할 수 있는 가능한 수정사항에 관한 정보는 ALTER TABLE 및 COMMENT문을 참조하십시오.
- DELETE는 사용자가 테이블 또는 뷰에서 행을 제거할 수 있도록 합니다.
- INDEX는 사용자가 데이터베이스에 인덱스를 작성할 수 있도록 합니다. 인덱스의 작성자는 인덱스에 대해 자동으로 CONTROL 특권을 가지게 됩니다.
- INSERT는 사용자가 테이블 또는 뷰에 행을 삽입하고 IMPORT 유틸리티를 실행하도록 합니다.
- REFERENCES는 사용자가 테이블을 관계의 상위로 지정하여, 외부 키를 작성하거나 제거할 수 있도록 합니다. 사용자는 특정 컬럼에 대해서만 이 특권을 가질 수도 있습니다.
- SELECT는 사용자가 테이블 또는 뷰에서 행을 검색하고, 테이블에 뷰를 작성하고, EXPORT 유틸리티를 실행할 수 있도록 합니다.
- UPDATE는 사용자가 테이블의 한 항목, 뷰 또는 테이블이나 뷰의 하나 이상의 특정 컬럼을 변경할 수 있도록 합니다. 사용자는 특정 컬럼에서만 이 특권을 가질 수 있습니다.

이러한 특권을 다른 사용자에게 권한 부여하는 특권은 GRANT문에서 WITH GRANT OPTION을 사용하여 권한 부여될 수도 있습니다.

주: 사용자 또는 그룹에 테이블에 대한 CONTROL 특권이 권한 부여될 경우, 해당 테이블의 다른 모든 특권에는 자동으로 WITH GRANT OPTION이 권한 부여됩니다. 나중에 사용자에게서 테이블에 대한 CONTROL 특권을 권한 취소해도, 사용자는 자동으로 권한 부여된 다른 특권을 계속 보유하게 됩니다. CONTRL 특권으로 권한 부여된 모든 특권을 권한 취소하려면, 각 특권을 명시적으로 권한 취소하거나, REVOKE문에서 ALL 키워드를 지정하십시오. 예를 들어, 다음과 같습니다.

```
REVOKE ALL
   ON EMPLOYEE FROM USER HERON
```

유형이 지정된 테이블에 대한 작업을 하는 경우, 테이블 및 뷰 특권과 관련하여 고려할 사항이 있습니다.

주: 특권은 테이블 계층 구조의 모든 레벨에서 별도로 권한 부여되어야 합니다. 그 결과, 입력된 테이블 내의 상위 테이블에서 특권을 권한 부여받은 사용자는 하위 테이블에도 간적접으로 영향을 줄 수 있습니다. 그러나 해당 하위 테이블에 필수 특권이 설정되어 있는 경우, 하위 테이블에 직접적으로 조작을 보류하십시오.

테이블 계층 구조 내 테이블 간의 상위 테이블/하위 테이블 관계는 SELECT, UPDATE 및 DELETE와 같은 조작이 해당 조작의 목표 테이블 및 모든 해당 하위 테이블(있는 경우)의 행에 영향을 줄 것임을 의미합니다. 이 작동을 *대체 가능성*이라 부를 수 있습니다. 예를 들어, 유형이 Manager_t인 하위 테이블 Manager로 유형이 Employee_t인 Employee 테이블을 작성했다고 가정합니다. 관리자도 사원의 (특정한) 일부이므로, 구조화 유형 Employee_t 및 Manager_t 간의 유형/부속 유형 관계와 테이블 Employee 및 Manager 간의 해당 테이블/하위 테이블 관계로 표시됩니다. 이 관계의 결과로서, SQL 쿼리는 다음과 같습니다.

```
SELECT * FROM Employee
```

는 사원과 관리자 모두에 대한 오브젝트 ID 및 Employee_t 속성을 리턴합니다. 이와 마찬가지로, 다음 갱신 조작은

```
UPDATE Employee SET Salary = Salary + 1000
```

정규 사원뿐 아니라 관리자의 급여를 1000달러 인상합니다.

Employee의 SELECT 특권을 가진 사용자는 Manager에 대한 명시적인 SELECT 특권을 가지고 있지 않더라도 이 SELECT 조작을 수행할 수 있습니다. 그러나 이러한 사용자는 Manage 하위 테이블에 직접 SELECT 조작을 수행할 수 없기 때문에, Manager 테이블의 상속되지 않은 컬럼에는 액세스할 수 없습니다.

마찬가지로, Employee에 대한 UPDATE 특권을 가진 사용자는 Manager에 대해 UPDATE 조작을 수행할 수 있기 때문에, Manager 테이블에 대한 명시적 UPDATE 특권을 가지고 있지 않더라도 정규 사원과 관리자 모두에게 영향을 줍니다. 그러나 이러한 사용자는 Manage 하위 테이블에 직접 UPDATE 조작을 수행할 수 없기 때문에, Manager 테이블의 상속되지 않은 컬럼은 갱신할 수 없습니다.

관련 개념:

- 277 페이지의『인덱스 특권』

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

관련 참조:

- *SQL 참조서, 볼륨 2*의『ALTER TABLE문』
- *SQL 참조서, 볼륨 2*의『CREATE VIEW문』
- *SQL 참조서, 볼륨 2*의『SELECT문』

패키지 특권

패키지란 특정 응용프로그램에 대해 가장 효과적인 방법으로 데이터에 액세스하기 위해 데이터베이스 관리자가 필요로 하는 정보가 들어 있는 데이터베이스 오브젝트입니다. 패키지 특권은 사용자가 패키지를 작성하고 조작할 수 있도록 합니다. 사용자는 다음 특권을 사용하기 위해 데이터베이스에 대해 CONNECT 권한을 가지고 있어야 합니다.

- CONTROL은 패키지를 리바인드, 제거 또는 실행할 능력과 이러한 특권을 다른 사용자에게 확대할 수 있는 능력을 제공합니다. 패키지의 작성자는 자동으로 이 특권을 받게 됩니다. CONTROL 특권이 있는 사용자에게는 BIND 및 EXECUTE 특권이 부여되며, GRANT 명령문을 사용하여 다른 사용자에게도 이러한 특권을 부여할 수 있습니다. (WITH GRANT OPTION으로 특권이 부여된 경우, BIND 또는 EXECUTE 특권을 받은 사용자는 이 특권을 다른 사용자에게 차례로 부여할 수 있습니다.) CONTROL 특권을 권한 부여하려면, 사용자에게 SYSADM 또는 DBADM 권한이 있어야 합니다.
- 패키지에 대한 BIND 특권은 사용자가 해당 패키지를 리바인드 또는 바인드하고 패키지 이름 및 작성자가 동일한 새 패키지 버전을 추가할 수 있도록 합니다.
- EXECUTE는 사용자가 패키지를 실행할 수 있도록 합니다.

주: 모든 패키지 특권은 동일한 패키지 이름 및 작성자를 공유하는 모든 버전에 적용됩니다.

이러한 패키지 특권 외에도, 사용자는 BINDADD 데이터베이스 특권을 사용하여 데이터베이스에 새로운 패키지를 작성하거나 기존의 패키지를 리바인드할 수 있습니다.

별칭으로 참조되는 오브젝트는 오브젝트를 포함하는 데이터 소스에 인증 점검을 전달해야 합니다. 뿐만 아니라, 패키지 사용자는 데이터 소스에 있는 데이터 소스 오브젝트에 대해 적합한 특권 또는 권한 레벨을 가지고 있어야 합니다.

DB2® Universal Database(DB2 UDB)는 DB2 계열 데이터 소스와 통신할 때 동적 SQL을 사용하므로, 별칭을 포함하는 패키지는 추가 권한 부여 단계가 필요할 수도 있습니다. 데이터 소스에서 패키지를 실행하는 권한 부여 ID는 해당 데이터 소스에서 동적으로 패키지를 실행할 권한을 가지고 있어야 합니다.

관련 개념:

- 268 페이지의 『데이터베이스 권한』

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

인덱스 특권

인덱스 또는 인덱스 스펙 작성자는 인덱스에 대해 자동으로 CONTROL 특권을 갖게 됩니다. 인덱스에 대한 CONTROL 특권은 인덱스를 제거할 권한이라 할 수 있습니다. 인덱스에 대한 CONTROL 특권을 권한 부여하려면, 사용자에게 SYSADM 또는 DBADM 권한이 있어야 합니다.

테이블 레벨 INDEX 특권은 사용자가 해당 테이블에 대한 인덱스를 작성할 수 있도록 합니다.

별칭 레벨 INDEX 특권은 사용자가 해당 별칭에 대한 인덱스 스펙을 작성할 수 있도록 합니다.

관련 개념:

- 273 페이지의 『테이블 및 뷰 특권』

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

시퀀스 특권

시퀀스의 작성자는 자동으로 시퀀스에 관한 USAGE 및 ALTER 특권을 받게 됩니다. 시퀀스를 위한 NEXT VALUE 및 PREVIOUS VALUE 표현식을 사용하려면 USAGE 특권이 필요합니다. 기타 사용자가 NEXT VALUE 및 PREVIOUS VALUE 표현식을 사용하도록 허용하려면 시퀀스 특권을 공용(PUBLIC)으로 권한 부여해야 합니다. 이로써 모든 사용자가 지정된 시퀀스로 표현식을 사용할 수 있게 됩니다.

사용자는 시퀀스에 관한 ALTER 특권으로 시퀀스 재시작 또는 후속 시퀀스 값에 대한 증분 변경과 같은 태스크를 수행할 수 있습니다. 시퀀스의 작성자는 ALTER 특권을 다른 사용자에게 부여할 수 있으며, WITH GRANT OPTION을 사용할 경우 해당 사용자가 차례로 이들 특권을 다른 사용자에게 부여할 수 있습니다.

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『ALTER SEQUENCE문』

루틴 특권

실행 특권에는 데이터베이스 내의 모든 유형의 루틴(예: 함수, 프로시저 및 메소드)에 대한 조치가 포함됩니다. EXECUTE 특권이 있으면, 사용자는 루틴을 호출하고, 해당 루틴을 소스로 하는 함수를 작성하며(함수에만 적용), CREATE VIEW 또는 CREATE TRIGGER 등의 DDL문에서 루틴을 참조할 수 있습니다.

외부 스토어드 프로시저, 함수 또는 메소드를 정의하는 사용자는 EXECUTE WITH GRANT 특권을 받습니다. EXECUTE 특권이 WITH GRANT OPTION을 통해 또 다른 사용자에게 부여될 경우, 해당 사용자는 EXECUTE 특권을 또다른 사용자에게 차례로 부여할 수 있습니다.

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

데이터베이스 오브젝트에 대한 액세스 제어

데이터 액세스를 제어하려면 직접 및 간접 특권, 관리자 권한, 패키지에 대해 이해해야 합니다. 이 절에서는 이러한 주제에 대해 설명하고 몇 가지 예도 제공합니다.

직접적으로 권한 부여된 특권은 시스템 카탈로그에 저장됩니다.

권한 부여는 다음 세 가지 방법으로 제어됩니다.

- 명시적 권한 부여는 GRANT 및 REVOKE문으로 제어되는 특권을 통해 제어됩니다.
- 내재적 권한 부여는 오브젝트를 작성 및 제거함으로써 제어됩니다.
- 간접 권한은 패키지와 연관됩니다.

주: 데이터베이스 그룹 이름은 GRANT 또는 REVOKE문에서 사용되거나 제어 센터에서 사용될 때 8자 이하이어야 합니다. 8자를 넘는 데이터베이스 그룹 이름이 승인되더라도, 8자를 넘는 이름을 사용하면 해당 그룹에 속하는 사용자가 데이터베이스 오브젝트에 액세스할 때 오류 메시지가 발생합니다.

관련 개념:

- 291 페이지의『보안 문제에 시스템 카탈로그 사용』

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

데이터베이스 오브젝트에 대한 액세스 제어 세부사항

GRANT 및 REVOKE문을 사용하여 데이터베이스 오브젝트에 대한 액세스를 제어합니다. 내재된 액세스 권한 부여와 간접 권한에 대해서도 설명합니다.

특권 부여

제한사항:

대부분의 데이터베이스 오브젝트에 특권을 권한 부여하려면, 사용자는 해당 오브젝트에 대해 SYSADM 권한, DBADM 권한 또는 CONTROL 특권을 가지고 있거나, WITH GRANT OPTION 특권을 가지고 있어야 합니다. 특권은 기존의 오브젝트에 대해서만 권한 부여될 수 있습니다. 그 밖의 다른 사용자에게 CONTROL 특권을 권한 부여하려면, 사용자에게 SYSADM 또는 DBADM 권한이 있어야 합니다. DBADM 권한을 권한 부여하려면, 사용자에게 SYSADM 권한이 있어야 합니다.

프로시저:

GRANT문은 권한 부여된 사용자가 특권을 권한 부여할 수 있도록 합니다. 특권을 하나의 명령문으로 하나 이상의 권한 부여 이름에 권한 부여하거나 PUBLIC에 권한 부여하여, 모든 사용자가 특권을 사용하게 할 수 있습니다. 권한 부여 이름은 개별 사용자 또는 그룹이 될 수 있습니다.

사용자와 그룹이 동일한 이름으로 존재하는 운영 체제에서 사용자 또는 그룹에 특권을 권한 부여할 것인지 여부를 지정하십시오. GRANT 및 REVOKE문은 둘다 키워드 USER 및 GROUP을 지원합니다. 이들 선택적 키워드가 사용되지 않는 경우, 데이터베이스 관리가 운영 체제 보안 기능을 검사하여 권한 부여 이름이 사용자 또는 그룹 식별 여부를 판별합니다. 권한 부여 이름이 사용자 또는 그룹 모두인 경우, 오류가 리턴됩니다.

다음 예에서는 HERON 사용자에게 EMPLOYEE 테이블에 대한 SELECT 특권을 권한 부여합니다.

```
GRANT SELECT
   ON EMPLOYEE TO USER HERON
```

다음 예에서는 HERON 그룹으로 EMPLOYEE 테이블에 대한 SELECT 특권을 권한 부여합니다.

```
GRANT SELECT
   ON EMPLOYEE TO GROUP HERON
```

관련 개념:

- 278 페이지의『데이터베이스 오브젝트에 대한 액세스 제어』

태스크 관련:

- 280 페이지의 『특권 취소』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『GRANT(데이터베이스 권한)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(인덱스 특권)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(패키지 특권)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(스키마 특권)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(테이블, 뷰 또는 별칭 특권)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(서버 특권)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(테이블 스페이스 권한 취소)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(시퀀스 특권)문』
- *SQL 참조서, 볼륨 2*의 『GRANT(루틴 특권)문』

특권 취소

REVOKE문은 권한 부여된 사용자가 다른 사용자에게 이미 권한 부여된 특권을 권한 취소할 수 있도록 합니다.

제한사항:

데이터베이스 오브젝트에 대한 특권을 권한 취소하려면, 해당 오브젝트에 대한 DBADM 권한, SYSADM 권한 또는 CONTROL 특권을 가지고 있어야 합니다. WITH GRANT OPTION 특권만으로는 특권을 권한 취소할 수 없다는 점을 유의하십시오. 또다른 사용자로부터 CONTROL 특권을 권한 취소하려면, SYSADM 또는 DBADM 권한을 가지고 있어야 합니다. DBADM 권한을 권한 취소하려면, SYSADM 권한을 가지고 있어야 합니다. 특권은 기존 오브젝트에 대해서만 권한 취소될 수 있습니다.

주: DBADM 권한 또는 CONTROL 특권을 가지고 있지 않은 사용자는 WITH GRANT OPTION을 사용하여 권한 부여한 특권을 권한 취소할 수 없습니다. 또한, 권한 취소된 사용자에게 특권을 권한 부여 받은 사용자의 권한을 연쇄적으로 권한 취소할 수 없습니다.

명시적으로 권한 부여된 테이블(또는 뷰) 특권이 DBADM 권한을 갖는 사용자로부터 권한 취소된 경우, 해당 테이블에 대해 정의된 다른 뷰에서 특권을 권한 취소할 수 **없습니다**. 이는 뷰 특권이 DBDAM 권한을 통해 사용 가능하고, 기초가 되는 테이블에 대한 명시적인 특권에 종속되지 않기 때문입니다.

프로시저:

특권이 동일한 이름을 갖는 사용자와 그룹에 모두 권한 부여되면, 특권을 권한 취소할 때 GROUP 또는 USER 키워드를 지정하십시오. 다음 예에서는 사용자 HERON에게서 EMPLOYEE 테이블의 SELECT 특권을 취소합니다.

```
REVOKE SELECT
   ON EMPLOYEE FROM USER HERON
```

다음 예에서는 그룹 HERON에게서 EMPLOYEE 테이블의 SELECT 특권을 권한 취소합니다.

```
REVOKE SELECT
   ON EMPLOYEE FROM GROUP HERON
```

그룹에서의 특권 권한 취소가 해당 그룹의 모든 구성원에서 특권을 권한 취소하는 것이 아님에 유의하십시오. 특권을 개별 이름에 직접 권한 부여한 경우, 해당 특권이 직접 권한 취소될 때까지 이름을 보존합니다.

테이블 특권이 사용자에게서 권한 취소되면, 해당 사용자가 작성한 뷰에 대한 특권도 권한 취소된 테이블 특권에 종속하는 것이므로 권한 취소됩니다. 그러나 시스템에 의해 권한 부여된 특권만이 권한 취소됩니다. 또다른 사용자에 의해 뷰에 대한 특권을 직접 권한 부여 받은 경우, 특권이 계속 보유됩니다.

특권을 그룹으로 권한 부여(GRANT)한 다음, 그룹의 한 구성원에게서 권한을 취소(REVOKE)하려는 상황이 있을 수 있습니다. 다음과 같은 오류 메시지 SQL0556N을 받지 않고 그렇게 하기 위해서는 오직 두 가지 방법만이 있습니다.

- 그룹으로부터 구성원을 제거하거나, 더 적은 수의 구성원이 있는 그룹 하나를 새로 작성하고 새 그룹에게 특권을 권한 부여(GRANT)합니다.
- 그룹으로부터 특권을 권한 취소(REVOKE)한 다음, 사용자에게 개별적으로 특권을 권한 부여합니다(권한 부여 ID).

주: 테이블 또는 뷰의 사용자가 CONTROL 특권을 권한 취소하더라도, 사용자는 여전히 특권을 다른 사용자에게 권한 부여할 수 있는 권한을 지닙니다. 사용자에게 CONTROL 특권이 주어질 경우, 다른 모든 특권 WITH GRANT OPTION도 권한 부여됩니다. 일단 CONTROL이 권한 취소되면, 다른 모든 특권이 명시적으로 권한 취소될 때까지 WITH GRANT OPTION이 남아 있습니다.

권한 취소된 특권에 종속적인 모든 패키지는 올바르지 않음으로 표시되지만, 해당 권한을 갖는 사용자에 의해 리바인드되면 다시 유효성이 확인될 수 있습니다. 특권이 연속적으로 응용프로그램 바인더에 다시 권한 부여될 경우 패키지도 재빌드될 수 있으며, 응용프로그램을 실행하면 내재적 리바인드가 성공적으로 트리거됩니다. 특권이 PUBLIC으로부터 권한 취소되면, PUBLIC 특권에 근거해야만 바인드 가능한 사용자가 바인드한 모든 패키지는 유효하지 않습니다. DBADM 권한이 사용자로부터 권한 취소되면, 데이터베이스 유틸리티와 연관된 패키지를 포함하여 해당 사용자가 바인드한 모든 패키지가 올바르지 않게 됩니다. 올바르지 않음으로 표시된 패키지를 사용하려고 시도하면 시스템이 패키지를 리바인드하려고 합니다. 이러한 리바인드 시도가 실패할 경우, 오류가 발생합니다(SQLCODE-727). 이 경우, 다음과 같은 권한을 갖는 사용자가 패키지를 명시적으로 리바인드해야 합니다.

- 패키지를 리바인드할 수 있는 권한
- 패키지 내에서 사용한 오브젝트에 대한 해당 권한

이들 패키지는 특권이 권한 취소될 때마다 리바인드되어야 합니다.

하나 이상의 특권에 근거하는 트리거를 정의하고 해당 특권 중 하나 이상을 손실한 경우, 트리거 또는 SQL 기능을 사용할 수 없습니다.

태스크 관련:

- 279 페이지의 『특권 부여』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『REVOKE(데이터베이스 권한)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(인덱스 특권)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(패키지 특권)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(스키마 특권)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(테이블, 뷰 또는 별칭 특권)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(서버 특권)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(테이블 스페이스 권한 취소)문』
- *SQL 참조서, 볼륨 2*의 『REVOKE(루틴 특권)문』

오브젝트를 작성 및 삭제하여 내재적 권한 부여 관리

프로시저:

데이터베이스 관리자는 테이블 또는 패키지와 같은 데이터베이스 오브젝트를 작성하는 사용자에게 특정 권한을 내재적으로 부여합니다. 또한, SYSADM 또는 DBADM 권한을 갖는 사용자가 오브젝트를 작성할 때에도 특권이 권한 부여됩니다. 마찬가지로, 특권은 오브젝트가 제거될 때 제거됩니다.

작성된 오브젝트가 테이블, 별칭, 인덱스 또는 패키지인 경우, 사용자는 오브젝트에 대한 CONTROL 특권을 받게 됩니다. 오브젝트가 뷰이면, 뷰 정의에서 참조된 모든 테이블, 뷰 및 별칭에 대해 사용자가 CONTROL 특권을 가지고 있는 경우에만, 뷰에 대한 CONTROL 특권이 내재적으로 부여됩니다.

명시적으로 작성된 오브젝트가 스키마일 경우, 스키마 소유자에게는 ALTERIN, CREATEIN 및 DROPIN 특권 WITH GRANT OPTION이 부여됩니다. 내재적으로 작성된 스키마는 PUBLIC에 권한 부여된 CREATEIN을 갖습니다.

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

패키지 소유권 설정

프로시저:

BIND 및 PRECOMPILE 명령은 응용프로그램 패키지를 작성하고 변경합니다. 두 경우 모두 OWNER 옵션을 사용하여 결과 패키지의 소유자 이름을 지정하십시오. 패키지 소유권의 이름 지정을 위한 간단한 규칙이 있습니다.

- 사용자는 자신을 소유자로서 지정할 수 있습니다. OWNER 옵션이 지정되지 않은 경우 이것이 디폴트값입니다.
- SYSADM 또는 DBADM 권한이 있는 ID는 OWNER 옵션을 사용하여 권한 부여 ID를 소유자로 지정할 수 있습니다.

DB2 Universal Database™(DB2 UDB) 데이터베이스 제품을 사용하여 패키지를 바인드할 수 있는 모든 운영 체제가 OWNER 옵션을 지원하는 것은 아닙니다.

관련 참조:

- *Command Reference*의 『BIND Command』
- *Command Reference*의 『PRECOMPILE Command』

패키지를 통한 간접 권한

데이터베이스 내의 데이터에 대한 액세스는 대화식 워크스테이션 세션에서 작업하고 있는 사용자뿐만 아니라, 응용프로그램에서도 요구할 수 있습니다. 패키지에는 사용자가 수 많은 데이터베이스 오브젝트에 대해 다양한 조치를 수행할 수 있도록 하는 명령문이 들어 있습니다. 각각의 이들 조치에는 하나 이상의 특권이 필요합니다.

패키지를 바인드하는 개별 및 PUBLIC에 권한 부여된 특권은 정적 SQL이 바인드될 때 권한 부여 검사용으로 사용됩니다. 그룹을 통해 권한 부여된 특권은 정적 SQL이 바인드될 때 권한 부여 검사용으로 사용되지 *않습니다*. 패키지를 바인드하는 유효한 *authID*는 패키지에서 정적 SQL문을 실행하도록 요청된 모든 특권이 명시적으로 부여되거나 패키지를 바인딩할 때 VALIDATE RUN이 지정되는 경우를 제외하고는 PUBLIC을 통해 필요한 특권이 내재적으로 부여되어야 합니다. VALIDATE RUN이 BIND때 지정된 경우, 이 패키지 내에서 정적 SQL문에 대한 모든 권한 부여 실패는 BIND를 실패하도록 야기하지 않으며 이러한 SQL문의 유효성이 런타임에서 재확인됩니다. PUBLIC, 그룹, 사용자 특권은 사용자에게 패키지를 바인드하는 적합한 권한 부여(BIND 또는 BINDADD 특권)가 있는지를 검사할 때 모두 사용됩니다.

패키지에는 정적 및 동적 SQL이 모두 포함됩니다. 정적 SQL로 패키지를 처리하려면, 사용자가 패키지에 대해 EXECUTE 특권만 가지고 있으면 됩니다. 그런 다음, 사용자는 패키지가 가지고 있는 제한 범위 내에서 패키지에 있는 정적 SQL에 대해 패키지 바인더의 특권을 간접적으로 얻을 수 있습니다.

패키지에 동적 SQL이 포함될 경우, 필수 특권은 패키지 사전 컴파일 또는 바운드 수행 시 DYNAMICRULES에 지정된 값에 따라 다릅니다. 자세한 정보는 동적 SQL에 관한 DYNAMICRULES의 효과를 설명하는 주제를 참조하십시오.

관련 개념:

- 284 페이지의 『별칭을 포함하는 패키지를 통한 간접 권한』
- *응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍*의 『동적 SQL에서 DYNAMICRULES 바인드 옵션의 효과』

관련 참조:

- *Command Reference*의 『BIND Command』

별칭을 포함하는 패키지를 통한 간접 권한

패키지에 별칭에 대한 참조가 들어 있으면, 패키지 작성자와 패키지 사용자에 대한 권한 부여 처리는 다소 복잡합니다. 패키지 작성자가 별칭이 들어 있는 패키지를 바인드하는 경우, 패키지 작성자는 데이터 소스에서 별칭이 참조하는 테이블과 뷰에 대해 인증 검사 또는 특권 검사를 통과할 필요는 없습니다. 그러나 패키지 실행자는 데이터 소스에서 인증 및 권한 부여 검사를 통과해야 합니다.

예를 들어, 패키지 작성자의 .SQC 파일에 여러 SQL문이 들어 있는 것으로 가정하십시오. 하나의 정적 명령문은 로컬 테이블을 참조합니다. 또다른 동적 명령문은 별칭을 참조합니다. 패키지가 바인드되면, 패키지 작성자의 인증 ID(authid)는 로컬 테이블 및 별칭에 대한 특권을 검증하는 데 사용됩니다. 그러나 별칭이 식별하는 데이터 소스 오브젝트에 대해서는 어떠한 검사도 수행되지 않습니다. 또다른 사용자가 패키지를 실행하면, 해당 패키지에 대해 EXECUTE 특권을 가지고 있고 사용자가 테이블을 참조하는 명령문에 대한 추가 특권 검사를 통과할 필요가 없는 것으로 가정합니다. 그러나 별칭을 참조하는 명령문의 경우, 패키지를 실행하는 사용자는 데이터 소스에서 인증 검사 및 특권 검사를 통과해야 합니다.

.SQC 파일에 동적 SQL문과 테이블 및 별칭 참조의 혼합이 들어 있으면 로컬 오브젝트 및 별칭에 대한 DB2® Universal Database(DB2 UDB) 권한 부여 점검이 유사합니다. 패키지 사용자는 명령문 내의 로컬 오브젝트(테이블, 뷰)에 대한 특권 검사를 통과해야 하며, 또한 별칭 오브젝트에 대한 특권 검사도 통과해야 합니다(패키지 사용자는 별칭이 식별하는 오브젝트가 들어 있는 데이터 소스에서 인증 및 특권 검사를 통과해야 합니다). 어느 경우에든, 패키지 사용자는 EXECUTE 특권을 가지고 있어야 합니다.

패키지 실행자의 ID와 암호는 모든 데이터 소스 인증 및 특권 처리에 사용됩니다. 이 정보는 사용자 맵핑을 작성하여 변경될 수 있습니다.

주: 별칭은 정적 SQL에서 지정될 수 없습니다. 별칭이 들어 있는 패키지와 함께 DYNAMICRULES 옵션(BIND로 설정된)을 사용하지 마십시오.

DB2 UDB는 DB2 계열 데이터 소스와 통신할 때 동적 SQL을 사용하므로, 별칭을 포함하는 패키지는 추가 권한 부여 단계가 필요할 수도 있습니다. 데이터 소스에서 패키지를 실행하는 권한 부여 ID는 해당 데이터 소스에서 동적으로 패키지를 실행할 권한을 가지고 있어야 합니다.

관련 개념:

• 283 페이지의『패키지를 통한 간접 권한』

뷰로 데이터에 대한 액세스 제어

뷰는 다음 사항을 허용함으로써 테이블에 대한 액세스 제어 또는 특권 확장의 수단을 제공합니다.

- 테이블의 지정된 컬럼에 대해서만 액세스

  테이블의 특정 컬럼에 대한 액세스만이 필요한 사용자와 응용프로그램의 경우에, 권한 부여된 사용자가 뷰를 작성하여 필요한 컬럼에만 주소지정되도록 컬럼을 제한할 수 있습니다.
- 테이블 행의 서브세트만을 액세스

  뷰 정의의 서브쿼리에 WHERE절을 지정함으로써, 권한 부여된 사용자가 뷰를 통해 주소지정되는 행을 제한할 수 있습니다.
- 데이터 소스 테이블이나 뷰에 있는 행 또는 컬럼의 서브세트에만 액세스. 별칭을 통해 데이터 소스에 액세스할 경우, 별칭을 참조하는 로컬 DB2® Universal Database (DB2 UDB) 뷰를 작성할 수 있습니다. 이들 뷰는 하나 이상의 데이터 소스에서 별칭을 참조할 수 있습니다.

  주: 둘 이상의 데이터 소스에 대한 별칭 참조사항이 들어 있는 뷰를 작성할 수 있으므로, 사용자는 하나의 뷰에서 다중 데이터 소스에 있는 데이터에 액세스할 수 있습니다. 이들 뷰를 *다중 위치 뷰*라고도 합니다. 이러한 뷰는 분산 환경을 통해 중요 테이블의 컬럼에서 정보를 조인할 때 또는 각 사용자에게 특정 오브젝트에 대한 데이터 소스에 필요한 특권이 부족할 때 유용합니다.

뷰를 작성하려면, 사용자는 뷰 정의에서 참조되는 각각의 테이블, 뷰 또는 별칭에 대하여 SYSADM 권한, DBADM 권한 또는 CONTROL이나 SELECT 특권이 있어야 합니다. 또한, 사용자는 뷰에 대해 지정된 스키마에서 오브젝트를 작성할 권한도 가지고 있어야 합니다. 즉, 스키마가 아직 존재하지 않을 경우 기존 스키마에 대한 CREATEIN 특권 또는 데이터베이스에 대한 IMPLICIT_SCHEMA 권한이 있어야 합니다.

별칭을 참조하는 뷰를 작성하는 경우, 뷰에서 별칭이 참조되는 데이터 소스 오브젝트(테이블 및 뷰)에 대해 추가 권한이 필요하지 않습니다. 그러나 뷰 사용자는 뷰를 액세스할 때 기초 데이터 소스 오브젝트에 대해 SELECT 권한 또는 동등한 권한 부여 레벨을 가지고 있어야 합니다.

사용자가 기초 오브젝트(테이블 및 뷰)에 대한 데이터 소스에서 적합한 권한을 가지고 있지 않으면, 다음을 수행할 수 있습니다.

1. 데이터 소스 테이블에서 사용자가 액세스할 수 있는 컬럼에 대해 데이터 소스 뷰를 작성합니다.
2. 이 뷰에 대한 SELECT 특권을 사용자에게 권한 부여합니다.
3. 뷰를 참조하는 별칭을 작성합니다.

그런 다음, 새로운 별칭을 참조하는 SELECT문을 발행하여 컬럼에 액세스할 수 있습니다.

다음 시나리오에서는 뷰가 정보에 대한 액세스를 제한하기 위해 사용되는 방법의 상세한 예를 제공합니다.

많은 사용자가 서로 다른 이유로 STAFF 테이블의 정보를 액세스해야 할 경우가 있습니다. 예를 들어, 다음과 같습니다.

- 인사부에서는 전체 테이블을 갱신하고 볼 수 있어야 합니다.

  이러한 요구사항은 STAFF 테이블에 대한 SELECT 및 UPDATE 특권을 그룹 PERSONNL에 권한 부여함으로써 쉽게 충족될 수 있습니다.

  ```
  GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
  ```

- 부서 관리자는 각각 부하 직원에 대한 급여 정보를 보아야 합니다.

  이러한 요구사항은 각 부서 관리자에 대한 뷰를 작성함으로써 충족될 수 있습니다. 예를 들어, 다음과 같은 뷰가 부서 번호 51의 관리자에 대해 작성될 수 있습니다.

  ```
  CREATE VIEW EMP051 AS
     SELECT NAME,SALARY,JOB FROM STAFF
     WHERE DEPT=51
  GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  권한 부여 이름 JANE을 갖는 관리자는 STAFF 테이블처럼 EMP051 뷰를 쿼리합니다. STAFF 테이블의 EMP051 뷰에 액세스하면, 이 관리자는 다음과 같은 정보를 볼 수 있습니다.

NAME	SALARY	JOB
Fraye	45150.0	Mgr
Williams	37156.5	Sales
Smith	35654.5	Sales
Lundquist	26369.8	Clerk

NAME	SALARY	JOB
Wheeler	22460.0	Clerk

- 모든 사용자는 다른 사원을 찾을 수 있어야 합니다. 이 요구사항은 STAFF 테이블의 NAME 컬럼과 ORG 테이블의 LOCATION 컬럼에 대한 뷰를 작성하여, DEPT 및 DEPTNUMB 컬럼 각각에 대해 두 개의 테이블을 조인함으로써 충족될 수 있습니다.

```
CREATE VIEW EMPLOCS AS
   SELECT NAME, LOCATION FROM STAFF, ORG
   WHERE STAFF.DEPT=ORG.DEPTNUMB
GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
```

  사원 위치 뷰에 액세스하는 사용자는 다음과 같은 정보를 볼 수 있습니다.

NAME	LOCATION
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington
Marenghi	Atlanta
O'Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco

NAME	LOCATION
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

태스크 관련:

- 142 페이지의『뷰 작성』
- 279 페이지의『특권 부여』

감사 기능을 사용하여 데이터에 대한 액세스 모니터링

DB2® Universal Database (DB2 UDB) 감사 기능은 일련의 사전 정의된 데이터베이스 이벤트에 대한 감사 추적을 생성하여 사용자가 이를 유지보수할 수 있게 합니다. 감사 기능은 데이터에 대한 액세스를 금지하지는 않지만, 데이터 오브젝트를 액세스하거나 수정하려고 시도하는 레코드를 모니터하고 보존할 수 있습니다.

감사 기능 관리자 도구 **db2audit**를 사용하려면 SYSADM 권한이 필요합니다.

관련 개념:

- 299 페이지의『DB2 Universal Database(DB2 UDB) 감사 기능 소개』

데이터 암호화

보안 플랜의 한 파트는 사용자의 데이터 암호화를 포함합니다. 암호화를 하려면, ENCRYPT, DECRYPT_BIN, DECRYPT_CHAR 및 GETHINT과 같은 암호화 및 암호 해독 내장 함수를 사용하면 됩니다.

ENCRYPT 함수는 암호 기반 암호화 메소드를 사용하여 데이터를 암호화합니다. 이 함수는 또한 사용자가 암호 힌트를 캡슐화하도록 합니다. 암호 힌트는 암호화된 데이터에 임베드되어 있습니다. 한번만 암호화되면, 데이터를 암호 해독할 수 있는 유일한 방법은 올바른 암호를 사용하는 것입니다. 이 함수를 사용하려고 선택한 개발 프로그램으로 잊어버린 암호 및 사용할 수 없는 데이터의 관리에 대해 계획해야 합니다.

ENCRYPT 함수의 결과는 VARCHAR FOR BIT DATA입니다(32 631의 한계가 있음).

CHAR, VARCHAR 및 FOR BIT DATA만을 암호화할 수 있습니다.

DECRYPT_BIN 및 DECRYPT_CHAR 함수는 암호 기반 암호 해독을 사용하여 암호를 해독합니다.

DECRYPT_BIN은 항상 VARCHAR FOR BIT DATA를 리턴하는 반면 DECRYPT_CHAR은 항상 VARCHAR을 리턴합니다. 첫 번째 인수는 CHAR FOR BIT DATA 또는 VARCHAR FOR BIT DATA이므로, 결과가 첫 번째 인수와 동일하지 않은 경우가 있습니다.

결과의 길이는 다음 8바이트 경계까지의 바이트 수에 따라 다릅니다. 결과의 길이는 선택적 힌트 매개변수가 지정된 경우 데이터 인수에 40을 더한 길이 더하기 다음 8바이트 경계까지의 바이트 수이거나 또는 선택적 힌트 매개변수가 지정되지 않은 경우 데이터 인수에 8을 더한 길이 더하기 다음 8바이트 경계까지의 바이트 수일 수 있습니다.

GETHINT 함수는 캡슐화된 암호 힌트를 리턴합니다. 암호 힌트는 데이터 소유자가 암호를 기억하도록 도와주는 문구입니다. 예를 들어, 단어 『Ocean』은 암호 "태평양(Pacific)"을 기억하는 힌트로 사용할 수 있습니다.

데이터를 암호화하는 데 사용되는 암호는 다음과 같은 두 가지 방법 중 하나로 판별합니다.

- 암호 인수. 암호는 암호화(ENCRYPT) 함수가 호출되면 명시적으로 전달되는 문자열입니다. 데이터는 주어진 암호로 암호화하고 암호 해독합니다.
- 암호화 암호 특수 레지스터. SET ENCRYPTION PASSWORD문은 암호 값을 암호화하고 암호화된 암호를 데이터베이스 관리 프로그램으로 보내어 특수 레지스터에 저장합니다. 암호 매개변수 없이 호출된 ENCRYPT, DECRYPT_BIN 및 DECRYPT_CHAR 함수는 ENCRYPTION PASSWORD 특수 레지스터의 값을 사용합니다. ENCRYPTION PASSWORD 특수 레지스터는 암호화된 양식으로만 저장됩니다.

  특수 레지스터의 초기값 또는 디폴트값은 비어 있는 문자열입니다.

암호의 유효 길이는 6에서 127까지입니다. 힌트의 유효 길이는 0에서 32까지입니다.

관련 참조:

- *SQL 참조서, 볼륨 2*의 『SET ENCRYPTION PASSWORD문』
- *SQL 참조서, 볼륨 1*의 『DECRYPT_BIN and DECRYPT_CHAR 스칼라 함수』
- *SQL 참조서, 볼륨 1*의 『ENCRYPT 스칼라 함수』
- *SQL 참조서, 볼륨 1*의 『GETHINT 스칼라 함수』

태스크 및 필수 권한 부여

모든 조직이 동일한 방식으로 작업량을 분담하지는 못합니다. 표 6에는 기타 일부 공통된 직책, 보통 작업에 수반되는 태스크, 이러한 태스크를 수행하는 데 필요한 권한 또는 특권이 나열되어 있습니다.

표 6. 공통 직책, 태스크 및 필수 권한 부여

직책	태스크	필수 권한 부여
부서 관리자	부서 시스템 감독, 데이터베이스 작성	SYSCTRL 권한. 부서에 자신의 인스턴스가 있으면 SYSADM 권한
보안 관리자	일부 또는 모든 권한 및 특권에 대해 다른 사용자에게 권한 부여	SYSADM 또는 DBADM 권한
데이터베이스 관리자	하나 이상의 데이터베이스를 설계, 개발, 조작, 보호 및 유지보수	하나 이상의 데이터베이스에 대한 DBADM 및 SYSMAINT 권한. 일부 경우에는 SYSCTRL 권한
시스템 운영자	데이터베이스를 모니터하고 백업 기능을 수행함	SYSMAINT 권한
응용프로그램 프로그래머	데이터베이스 관리 응용프로그램 개발 및 테스트, 테스트 데이터 테이블 작성	기존 패키지에 BINDADD 및 BIND, 하나 이상의 데이터베이스에 CONNECT 및 CREATETAB, 일부 특정 스키마 특권, 일부 테이블에 대한 특권 목록 CREATE_EXTERNAL_ROUTINE이 또한 필요할 수도 있습니다.
사용자 분석자	시스템 카탈로그 뷰를 검사하여 응용프로그램에 대한 데이터 요구사항 정의	카탈로그 뷰에 대해서는 SELECT, 하나 이상의 데이터베이스에 대해서는 CONNECT
프로그램 일반 사용자	응용프로그램 실행	패키지에 대해서는 EXECUTE, 하나 이상의 데이터베이스에 대해서는 CONNECT. 이 테이블 다음 주 참조
정보 센터 상담원	쿼리 사용자의 데이터 요구사항 정의, 테이블 및 뷰를 작성하고 데이터베이스 오브젝트로 액세스를 부여하여 데이터 제공	하나 이상의 데이터베이스에 대한 DBADM 권한
쿼리 사용자	SQL문을 발행하여 데이터 검색, 추가, 삭제 또는 변경 결과를 테이블로 저장	하나 이상의 데이터베이스에 대해서는 CONNECT, 작성 중인 테이블 및 뷰의 스키마에 대해서는 CREATEIN 그리고 일부 테이블 및 뷰에 대해서는 SELECT, INSERT, UPDATE, DELETE

주: 응용프로그램에 동적 SQL문이 있으면, 프로그램 일반 사용자는 EXECUTE 및 CONNECT 이외의 다른 특권(예: SELECT, INSERT, DELETE 및 UPDATE)이 필요할 수도 있습니다.

관련 개념:

- 263 페이지의 『시스템 관리 권한(SYSADM)』
- 264 페이지의 『시스템 제어 권한(SYSCTRL)』

- 265 페이지의 『시스템 유지보수 권한(SYSMAINT)』
- 266 페이지의 『데이터베이스 관리 권한(DBADM)』
- 267 페이지의 『LOAD 권한』
- 268 페이지의 『데이터베이스 권한』

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

보안 문제에 시스템 카탈로그 사용

각 데이터베이스에 대한 정보는 자동으로 시스템 카탈로그라고 하는 뷰 세트에 유지보수됩니다. 이 시스템 카탈로그는 데이터베이스가 생성될 때 작성됩니다. 시스템 카탈로그에서는 테이블, 컬럼, 인덱스, 프로그램, 특권 및 기타 오브젝트에 대해 설명합니다.

이 뷰는 사용자가 보유한 특권과 각 특권을 부여하는 사용자의 ID를 나열합니다.

SYSCAT.DBAUTH 데이터베이스 특권 나열

SYSCAT.TABAUTH 테이블 및 뷰 특권 나열

SYSCAT.COLAUTH 컬럼 특권 나열

SYSCAT.PACKAGEAUTH 패키지 특권 나열

SYSCAT.INDEXAUTH 인덱스 특권 나열

SYSCAT.SCHEMAAUTH 스키마 특권 나열

SYSCAT.PASSTHRUAUTH 서버 특권 나열

SYSCAT.ROUTINEAUTH 루틴(함수, 메소드 및 스토어드 프로시저) 특권 나열

시스템이 사용자에게 권한 부여한 특권은 SYSIBM을 권한 준 사용자로 갖게 됩니다. SYSADM, SYSMAINT 및 SYSCTRL은 시스템 카탈로그에 나열되지 않습니다.

CREATE 및 GRANT문은 시스템 카탈로그에 특권을 위치시킵니다. SYSADM 및 DBADM 권한을 가지고 있는 사용자는 시스템 카탈로그 뷰에 대해 SELECT 특권을 권한 부여하고 권한 취소할 수 있습니다.

태스크 관련:

- 292 페이지의 『부여된 특권이 있는 권한 부여 이름 검색』
- 293 페이지의 『DBADM 권한이 있는 모든 이름 검색』
- 293 페이지의 『테이블 액세스 권한이 부여된 이름 검색』
- 294 페이지의 『사용자에게 부여된 모든 특권 검색』
- 294 페이지의 『시스템 카탈로그 뷰 보안』

관련 참조:

- *SQL 참조서, 볼륨 1*의 『SYSCAT.COLAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.DBAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.INDEXAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.PACKAGEAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.SCHEMAAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.TABAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.PASSTHRUAUTH 카탈로그 뷰』
- *SQL 참조서, 볼륨 1*의 『SYSCAT.ROUTINEAUTH 카탈로그 뷰』

보안 문제에 시스템 카탈로그를 사용하는 방법에 대한 세부사항

이 섹션에서는 데이터베이스에서 어떤 사용자가 어떤 특권을 갖고 있는지 판별하는 몇 가지 방법을 검토합니다.

부여된 특권이 있는 권한 부여 이름 검색

프로시저:

모든 특권에 대한 정보가 하나의 시스템 카탈로그 뷰에 들어 있지는 않습니다. 다음 명령문에서는 특권을 가진 모든 권한 부여 이름을 검색합니다.

```
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

정기적으로, 이 명령문에 의해 검색되는 목록은 시스템 보안 기능에 정의된 사용자 및 그룹 이름의 목록과 비교되어야 합니다. 그러면 더 이상 유효하지 않은 해당 권한 부여 이름을 식별할 수 있습니다.

주: 리모트 데이터베이스 클라이언트를 지원하고 있는 경우, 리모트 클라이언트에만 권한 부여 이름이 정의되고 데이터베이스 서버 머신에는 정의되지 않도록 할 수 있습니다.

관련 개념:

- 291 페이지의『보안 문제에 시스템 카탈로그 사용』

DBADM 권한이 있는 모든 이름 검색

프로시저:

다음 명령문은 직접적으로 DBADM 권한이 권한 부여된 모든 권한 부여 이름을 검색합니다.

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

관련 개념:

- 266 페이지의『데이터베이스 관리 권한(DBADM)』
- 291 페이지의『보안 문제에 시스템 카탈로그 사용』

테이블 액세스 권한이 부여된 이름 검색

프로시저:

다음 명령문에서는 JAMES 규정자로 테이블 EMPLOYEE에 액세스하도록 직접 권한이 부여된 모든 권한 부여 이름을 검색합니다.

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
```

규정자 JAMES로 테이블 EMPLOYEE를 갱신할 수 있는 사용자를 알아내려면, 다음 명령문을 발행하십시오.

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
      (CONTROLAUTH  = 'Y' OR
       UPDATEAUTH   = 'Y' OR  UPDATEAUTH   = 'G')
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
   PRIVTYPE = 'U'
```

이는 CONTROL 또는 UPDATE 특권이 직접 권한 부여된 이름뿐만 아니라, DBADM 권한을 갖는 모든 권한 부여 이름도 검색합니다. 그러나 SYSADM 권한만을 보유하고 있는 사용자의 권한 부여 이름은 리턴되지 않습니다.

권한 부여 이름 중 일부는 개별 사용자가 아닌 그룹 ID일 수도 있음을 기억하십시오.

관련 개념:

- 273 페이지의『테이블 및 뷰 특권』
- 291 페이지의『보안 문제에 시스템 카탈로그 사용』

사용자에게 부여된 모든 특권 검색

프로시저:

시스템 카탈로그 뷰에 대해 쿼리함으로써, 사용자가 보유하고 있는 특권의 목록과 다른 사용자에게 권한 부여한 특권의 목록을 검색할 수 있습니다. 예를 들어, 다음 명령문에서는 개별 권한 부여 이름에 직접 권한 부여된 데이터베이스 특권의 목록을 검색합니다.

```
SELECT * FROM SYSCAT.DBAUTH
   WHERE GRANTEE = USER AND GRANTEETYPE = 'U'
```

다음 명령문에서는 특정 사용자에 의해 직접 권한 부여된 테이블 특권의 목록을 검색합니다.

```
SELECT * FROM SYSCAT.TABAUTH
   WHERE GRANTOR  = USER
```

다음 명령문에서는 특정 사용자에 의해 직접 권한 부여된 개별 컬럼 특권의 목록을 검색합니다.

```
SELECT * FROM SYSCAT.COLAUTH
   WHERE GRANTOR  = USER
```

이 명령문에서 키워드 USER는 사용자의 권한 부여 이름 값과 항상 같습니다. USER는 읽기 전용 특수 레지스터입니다.

관련 개념:

- 257 페이지의『특권, 권한 레벨 및 데이터베이스 권한』
- 268 페이지의『데이터베이스 권한』
- 291 페이지의『보안 문제에 시스템 카탈로그 사용』

태스크 관련:

- 279 페이지의『특권 부여』
- 280 페이지의『특권 취소』

시스템 카탈로그 뷰 보안

프로시저:

데이터베이스가 작성되는 동안, 시스템 카탈로그 뷰에 대한 SELECT 특권이 PUBLIC에 권한 부여됩니다. 대부분의 경우, 보안 문제점이 나타나지 않습니다. 그러나 매우 중요

한 데이터의 경우에 이들 테이블이 데이터베이스에 있는 모든 오브젝트를 서술하기 때문에, 적절하지 않을 수도 있습니다. 이 경우, PUBLIC으로부터 SELECT 특권을 권한 취소하고, 특정 사용자에게 필요한 SELECT 특권을 권한 부여하는 방법을 고려해 보십시오. 시스템 카탈로그 뷰에 대해 SELECT를 권한 부여하고 권한 취소하는 방법은 모든 뷰에서 동일하지만, 사용자가 이를 수행할 수 있는 SYSADM 또는 DBADM 권한 가지고 있어야 합니다.

최소한 다음 카탈로그 뷰로의 액세스를 제한하는 방법을 고려해야 합니다.

- SYSCAT.DBAUTH
- SYSCAT.TABAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.COLAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.SCHEMAAUTH

이렇게 하면, 데이터베이스에 액세스하는 모든 사용자가 사용자 특권에 대한 정보를 사용할 수 없게 됩니다. 비윤리적인 사용자가 이 정보로 데이터베이스에 대해 권한 없이 액세스할 수도 있습니다.

또한 통계가 수집되는 컬럼도 조사해야 합니다. 시스템 카탈로그에 기록된 일부 통계에는 사용자의 환경에서 중요한 데이터가 될 수 있는 데이터 값이 들어 있습니다. 이러한 통계에 관련 데이터가 들어 있으면, SYSCAT.COLUMNS 및 SYSCAT.COLDIST 카탈로그 뷰에 대해 PUBLIC으로부터 SELECT 특권을 권한 취소할 수도 있습니다.

시스템 카탈로그 뷰로의 액세스를 제한하려는 경우, 뷰를 정의하여 각 권한 부여 이름이 자체 특권 정보를 검색하도록 할 수 있습니다.

예를 들어, 다음 뷰 MYSELECTS에는 사용자의 권한 부여 이름이 SELECT 특권에 직접 권한 부여된 모든 테이블의 이름 및 소유자가 있습니다.

```
CREATE VIEW MYSELECTS AS
   SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
   WHERE GRANTEETYPE = 'U'
     AND GRANTEE = USER
     AND SELECTAUTH = 'Y'
```

이 명령문에서 키워드 USER는 권한 부여 이름 값과 항상 같습니다.

다음 명령문에서는 모든 권한 부여 이름에 사용 가능한 뷰를 작성합니다.

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

그리고 마지막으로, 기본 테이블에 대한 SELECT 특권을 권한 취소해야 합니다.

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
```

관련 개념:

- *관리 안내서: 성능*의 『카탈로그 통계』
- 268 페이지의 『데이터베이스 권한』
- 291 페이지의 『보안 문제에 시스템 카탈로그 사용』

태스크 관련:

- 279 페이지의 『특권 부여』
- 280 페이지의 『특권 취소』

방화벽 지원 입문

*방화벽*은 시스템 또는 네트워크에 대한 비승인 액세스를 방지하는데 사용되는 관련 프로그램 세트로 네트워크 게이트웨이에 위치합니다.

다음과 같은 네 가지 유형의 방화벽이 있습니다.

1. 네트워크 레벨, 패킷 필터 또는 스크리닝 라우터 방화벽
2. 클래식 응용프로그램 레벨 프록시 방화벽
3. 회선 레벨 또는 투명한 프록시 방화벽
4. SMLI(Stateful Multi-Layer Inspection) 방화벽

위에 나열된 방화벽 유형 중 하나가 통합된 기존의 방화벽 제품들이 있습니다. 위 유형의 일부가 조합되어 통합된 다른 방화벽 제품들도 많이 있습니다.

관련 개념:

- 296 페이지의 『스크리닝 라우터 방화벽』
- 297 페이지의 『응용프로그램 프록시 방화벽』
- 297 페이지의 『회선 레벨 방화벽』
- 297 페이지의 『SMLI(Stateful Multi-Layer Inspection) 방화벽』

스크리닝 라우터 방화벽

이 유형의 방화벽은 네트워크 레벨 또는 패킷 필터 방화벽이라고도 합니다. 이러한 방화벽은 들어오는 패킷을 프로토콜 속성으로 스크리닝함으로써 작동합니다. 스크리닝되는 프로토콜 속성에는 소스 또는 목적지 주소, 프로토콜 유형, 소스 또는 목적지 포트, 또는 그 외의 몇 가지 프로토콜 특정 속성들이 포함될 수 있습니다.

모든 방화벽 솔루션(SOCKS는 제외)에 대해, DB2® Universal Database(DB2 UDB)에서 사용하는 모든 포트가 수신 및 전송 패킷에 대해 열려 있도록 해야 합니다. DB2 UDB는 DB2 UDB 도구가 사용하는 DAS(DB2 Administration Server)에 포트 523

을 사용합니다. 서비스 파일을 사용하여 서버 데이터베이스 관리 프로그램 구성 파일에 있는 서비스 이름을 해당 포트 번호로 맵핑함으로써 모든 서버 인스턴스에서 사용하는 포트를 결정하십시오.

관련 개념:

• 296 페이지의『방화벽 지원 입문』

응용프로그램 프록시 방화벽

프록시 또는 프록시 서버는 웹 클라이언트와 웹 서버 사이에서 중개자 역할을 하는 기술입니다. 프록시 방화벽은 클라이언트로부터의 요청에 대한 하나의 게이트웨이로서 동작합니다. 방화벽이 클라이언트 요청을 수신하면 프록시 소프트웨어가 최종 서버 목적지 주소를 판별합니다. 응용프로그램 프록시가 주소를 변환하고 필요한 경우 추가 액세스 제어 점검 및 로그를 수행한 다음 클라이언트를 대신하여 서버에 연결합니다.

방화벽 머신에 있는 DB2® Connect 제품이 목적지 서버에 대한 프록시로서 동작할 수 있습니다. 또한 최종 목적지 서버에 대한 홉 서버로서 동작하는 방화벽 상의 DB2 Universal Database™(DB2 UDB) 서버가 응용프로그램 프록시처럼 동작할 수도 있습니다.

관련 개념:

• 296 페이지의『방화벽 지원 입문』

회선 레벨 방화벽

이 유형의 방화벽은 투명한 프록시 방화벽이라고도 합니다. 투명한 프록시 방화벽은 요청을 수정하지 않으며 프록시 인증 및 식별에 필요한 것 이외에는 응답하지 않습니다. 투명한 프록시 방화벽의 한 예가 SOCKS입니다.

DB2® Universal Database(DB2 UDB)는 SOCKS 버전 4를 지원합니다.

관련 개념:

• 296 페이지의『방화벽 지원 입문』

SMLI(Stateful Multi-Layer Inspection) 방화벽

이 유형의 방화벽은 OSI(Open System Interconnection) 모델의 일곱 계층 모두를 조사하는 정교한 양식의 패킷 필터링입니다. 각 패킷은 잘 알려진 상태에서 패킷 친화적으로 실험 및 비교되었습니다. 스크리닝 라우터 방화벽이 패킷 헤더만을 조사하는 반면, SMLI 방화벽은 데이터를 비롯하여 전체 패킷을 조사합니다.

관련 개념:

- 296 페이지의『방화벽 지원 입문』

제 8 장 DB2 Universal Database™(DB2 UDB) 활동 감사

DB2 Universal Database(DB2 UDB) 감사 기능 소개

알려지거나 예측된 데이터 액세스를 제어하는 데에는 인증, 권한 및 특권을 사용할 수 있으나, 이들 방법은 알려지지 않거나 예측되지 않은 데이터 액세스를 방지하는 데에는 충분하지 않습니다. 이러한 알 수 없거나 예측할 수 없는 데이터 액세스 유형의 발견을 돕기 위해, DB2® Universal Database(DB2 UDB)는 감사 기능을 제공합니다. 원하지 않는 데이터 액세스 및 후속 분석의 성공적인 모니터링은 데이터 액세스의 제어와 데이터에 대해 악의적이거나 부주의한 권한이 없는 액세스의 방지를 개선시킬 수 있습니다. 시스템 관리 조치를 포함하여 응용프로그램 및 각 사용자 액세스의 모니터링은 사용자 데이터베이스 시스템에서의 활동의 실행기록 레코드를 제공할 수 있습니다.

DB2 UDB 감사 기능은 일련의 사전 정의된 데이터베이스 이벤트에 대한 감사 추적을 생성하고 유지보수할 수 있게 합니다. 이 기능에서 생성된 레코드는 감사 로그 파일에 보존됩니다. 이들 레코드의 분석은 시스템 오용을 식별하는 사용 패턴을 나타낼 수 있습니다. 일단 시스템 오용이 식별되면, 이러한 시스템 오용을 감소시키거나 줄이기 위한 조치가 취해질 수 있습니다.

감사 기능은 모든 인스턴스 레벨 활동 및 데이터베이스 레벨 활동을 기록하며 인스턴스 레벨에서 작동합니다.

파티션된 데이터베이스 환경에서 작업할 때, 사용자가 연결된 파티션(코디네이터 노드) 또는 카탈로그 노드(이들 이벤트가 동일한 파티션이 아닌 경우)에서 감사할 수 있는 여러 이벤트가 발생합니다. 이것이 함축하는 의미는 감사 레코드가 둘 이상의 파티션에 의해 생성될 수 있다는 것입니다. 각 감사 레코드의 일부에는 코디네이터 노드 및 원래 노드 ID에 대한 정보가 들어 있습니다.

감사 로그(db2audit.log) 및 감사 구성 파일(db2audit.cfg)은 인스턴스의 `security` 서브디렉토리에 있습니다. 인스턴스 작성시, 가능하다면 운영 체제에 의해 이들 파일에 읽기/쓰기 사용 권한이 설정됩니다. 디폴트로, 사용 권한은 인스턴스 소유자 전용의 읽기/쓰기입니다. 이들 사용 권한을 변경하지 않도록 권장합니다.

감사 기능 관리자 도구(db2audit)의 사용자는 SYSADM 권한을 가져야 합니다.

감사 기능은 명시적으로 중지 및 시작되어야 합니다. 감사 기능이 시작되면, 기존 감사 구성 정보를 사용합니다. 감사 기능은 DB2 UDB 서버와 무관하므로, 인스턴스가 중지되더라도 사용 중인 상태로 남습니다. 사실, 인스턴스가 중지되면, 감사 레코드는 감사 로그에서 생성될 수 있습니다.

감사 기능의 권한 부여된 사용자는 감사 기능 내에서 다음 조치를 제어할 수 있습니다.

- DB2 UDB 인스턴스 내에서 감사 가능한 이벤트 기록을 시작합니다.
- DB2 UDB 인스턴스 내에서 감사 가능한 이벤트 기록을 중지시킵니다.
- 기록될 감사 가능한 이벤트 범주 선택을 비롯한 감사 기능의 작동을 구성합니다.
- 현재 감사 구성의 설명을 요청합니다.
- 인스턴스에서 보류 중인 감사 레코드를 비우고 감사 로그에 작성합니다.
- 감사 로그에서 플랫 파일 또는 ASCII 분리 파일로 감사 레코드를 포맷팅하고 복사하여 감사 레코드를 추출합니다. 로그 레코드의 분석을 위한 준비 또는 로그 레코드의 제거를 위한 준비에서 추출이 완료됩니다.
- 현재 감사 로그에서 감사 레코드를 프룬(prune)합니다.

주: 감사 유틸리티를 사용하기 전, db2audit 시작 명령을 발행하여 감사 기능을 작동시키십시오.

생성될 수 있는 감사 레코드의 다른 범주가 있습니다. 감사에 사용 가능한 이벤트 범주의 설명에서(아래), 각 범주의 이름 다음에 오는 것은 범주 유형을 식별하기 위해 사용하는 하나의 단어 키워드임을 기억해야 합니다. 감사에 사용 가능한 이벤트의 범주는 다음과 같습니다.

- 감사(AUDIT). 감사 설정이 변경되거나 감사 로그가 액세스될 때 레코드를 생성합니다.
- 권한 부여 검사(CHECKING). DB2 UDB 오브젝트 또는 기능에 액세스하거나 조작하려는 시도에 대한 권한 부여 검사 동안에 레코드를 생성합니다.
- 오브젝트 유지보수(OBJMAINT). 데이터 오브젝트를 작성하거나 제거할 때 레코드를 생성합니다.
- 보안 유지보수(SECMAINT). 오브젝트 또는 데이터베이스 특권 또는 DBADM 권한을 부여하거나 취소할 때 레코드를 생성합니다. 데이터베이스 관리 프로그램 보안 구성 매개변수 SYSADM_GROUP, SYSCTRL_GROUP 또는 SYSMAINT_GROUP이 수정될 때 레코드가 생성됩니다.
- 시스템 관리(SYSADMIN). SYSADM, SYSMAINT 또는 SYSCTRL 권한이 필요한 조작이 수행될 때 레코드를 생성합니다.
- 사용자 유효성 확인(VALIDATE). 사용자를 인증하거나 시스템 보안 정보를 검색할 때 레코드를 생성합니다.
- 조작 컨텍스트(CONTEXT). 데이터베이스 조작이 수행될 때 조작 컨텍스트를 표시하기 위해 레코드를 생성합니다. 이 범주는 감사 로그 파일을 더 잘 이해하게 합니다. 로그의 이벤트 상관자 필드를 사용할 때, 이벤트 그룹은 다시 하나의 데이터베이스 조작에 연관될 수 있습니다. 예를 들어, 동적 SQL에 대한 SQL문, 정적 SQL에 대한 패키지 ID 또는 CONNECT와 같이 수행되는 조작 유형의 표시기는 감사 결과를 분석할 때 필요한 컨텍스트를 제공할 수 있습니다.

주: 조작 컨텍스트를 제공하는 SQL문은 매우 길며 CONTEXT 레코드 내에서 완전히 표시됩니다. 이것은 CONTEXT 레코드를 매우 크게 만들 수 있습니다.

- 실패 또는 성공 중 하나를 감사하거나 둘다 감사할 수 있습니다.

데이터베이스에서의 조작은 여러 레코드를 생성할 수 있습니다. 생성되어 감사 로그로 이동되는 실제 레코드 수는 감사 기능 구성에서 지정한 대로 기록되는 이벤트의 범주 수에 따라 다릅니다. 또한 성공, 실패 또는 둘다 중 어느 것을 감사하는지 여부에 따라서도 다릅니다. 이러한 이유로 감사할 이벤트를 선택하는 것은 중요합니다.

관련 개념:

- 301 페이지의『감사 기능 동작』
- 317 페이지의『감사 기능 레코드 레이아웃(소개)』
- 335 페이지의『감사 기능 추가 정보 및 기술』

태스크 관련:

- 337 페이지의『DB2 UDB 감사 기능 활동 제어』

관련 참조:

- 303 페이지의『감사 기능 사용』
- 316 페이지의『감사 기능 메시지』

감사 기능 동작

감사 기능은 영향을 미치는 데이터베이스 인스턴스를 포함하는 감사 가능한 이벤트를 기록합니다. 이러한 이유로, 감사 기능은 DB2 UDB 인스턴스가 중지될 경우에도 작동할 수 있는 DB2® Universal Database(DB2 UDB)의 독립된 파트입니다. 감사 기능이 사용 중이면, 중지된 인스턴스가 시작될 때 인스턴스의 데이터베이스 이벤트 감사가 재개됩니다.

감사 로그에 감사 레코드를 작성하는 시기는 인스턴스에서의 데이터베이스 성능에 상당한 영향을 줍니다. 감사 레코드의 작성은 이들 레코드의 생성을 일으키는 이벤트 어커런스와 함께 동기적 또는 비동기적으로 발생할 수 있습니다. *audit_buf_sz* 데이터베이스 관리 프로그램 구성 매개변수의 값은 감사 레코드가 수행되는 시기를 판별합니다.

이 매개변수의 값이 0이면, 쓰기는 동기적으로 완료됩니다. 감사 레코드를 생성하는 이벤트가 레코드를 디스크에 작성할 때까지 대기합니다. 각 레코드와 연관된 대기는 DB2 UDB 성능이 저하되게 합니다.

*audit_buf_sz*의 값이 0보다 크면, 레코드 작성은 비동기적으로 완료됩니다. *audit_buf_sz*의 값이 0보다 커지면 내부 버퍼를 작성하기 위해 몇 개의 4KB 페이지가 사용됩니다. 감사 레코드의 그룹을 디스크에 작성하기 전에 몇 개의 감사 레코드를 보존하기 위해

내부 버퍼가 사용됩니다. 감사 레코드를 감사 이벤트의 결과로서 생성하는 명령문은 레코드가 디스크에 작성될 때까지 기다리지 않고 조작을 계속할 수 있습니다.

비동기 경우에, 감사 레코드가 얼마 동안 채워지지 않은 버퍼에 남아 있을 수 있습니다. 이것이 확장 기간 동안 발생하지 않게 하기 위해 데이터베이스 관리 프로그램은 정기적으로 감사 레코드 작성을 강제로 실행합니다. 감사 기능의 권한이 부여된 사용자는 명시적 요청으로 감사 버퍼를 비울 수 있습니다.

동기 레코드 작성이 있는지 비동기 레코드 작성이 있는지 여부에 따라 오류 발생 시기가 다릅니다. 비동기 모드의 경우, 감사 레코드가 디스크에 기록되기 전에 버퍼 처리되기 때문에 몇몇 레코드가 유실될 수 있습니다. 동기 모드에서는 오류가 발생해도 많아야 하나의 감사 레코드만 쓰기 금지되기 때문에 하나의 레코드가 유실될 수 있습니다.

ERRORTYPE 감사 기능 매개변수의 설정은 오류가 DB2 UDB와 감사 기능 간에 어떻게 관리되는지를 제어합니다. 감사 기능이 사용 중이고, ERRORTYPE 감사 기능 매개변수가 AUDIT이면, 감사 기능은 DB2 UDB의 다른 파트와 동일한 방법으로 처리됩니다. 성공적이라고 간주되는 명령문과 연관된 감사 이벤트에 대한 감사 레코드가 작성되어야 합니다(동기 모드에서 디스크로 또는 비동기 모드에서 감사 버퍼로). 이 모드를 실행할 때 오류가 발생할 때마다, 감사 레코드를 생성하는 명령문에 대해 음의 SQLCODE가 응용프로그램에 리턴됩니다. 오류 유형이 NORMAL로 설정되면, db2audit로부터의 모든 오류는 무시되며 조작의 SQLCODE가 리턴됩니다.

API 또는 SQL문과 DB2 UDB 인스턴스에 대한 감사 설정에 따라, 없음, 하나 또는 여러 감사 레코드가 특정 이벤트에 대해 생성될 수 있습니다. 예를 들어, SELECT 서브쿼리가 있는 SQL UPDATE문은 테이블에서 UPDATE 특권에 대한 권한 부여 검사의 결과가 들어 있는 하나의 감사 레코드와 테이블에서 SELECT 특권에 대한 권한 부여 점검의 결과가 들어 있는 또다른 레코드의 결과를 가져올 수 있습니다.

동적 DML(Data Manipulation Language)문의 경우, 명령문이 준비되면 모든 권한 부여 검사에 대해 감사 레코드가 생성됩니다. 이때는 어떠한 감사 검사도 발생하지 않으므로 동일한 사용자에 의한 이들 명령문의 재사용은 다시 감사되지 않습니다. 그러나 특권 정보가 들어 있는 카탈로그 테이블 중 하나가 변경되면, 다음 작업 단위(UOW)에서 캐시된 동적 SQL문에 대한 명령문 특권이 검사되며 하나 이상의 새로운 감사 레코드가 작성됩니다.

정적 DML문만 들어 있는 패키지의 경우, 감사 레코드를 생성할 수 있는 유일한 감사 가능한 이벤트는 사용자가 해당 패키지를 실행할 특권을 가지고 있는지 여부를 보기 위한 권한 부여 검사입니다. 패키지에서 정적 SQL문에 필요한 가능한 감사 레코드 작성과 권한 부여 검사는 패키지가 프리컴파일되거나 바인드될 때 수행됩니다. 패키지 내의 정적 SQL문의 실행은 감사 가능하지 않습니다. 패키지가 사용자에 의해 명시적으로 또는 시스템에 의해 내재적으로 리바인드될 때, 정적 SQL문에 의해 필요한 권한 부여 검사에 대한 감사 레코드가 생성됩니다.

권한 부여 검사가 명령문 실행 시간에서 수행되는 명령문의 경우(예: DDL, GRANT 및 REVOKE문), 이들 명령문이 사용될 때마다 감사 레코드가 생성됩니다.

주: DDL을 실행할 때, 감사 레코드의 모든 이벤트(컨텍스트 이벤트 제외)에 기록되는 섹션 번호는 명령문의 실제 섹션 번호와 상관없이 0이 됩니다.

관련 개념:

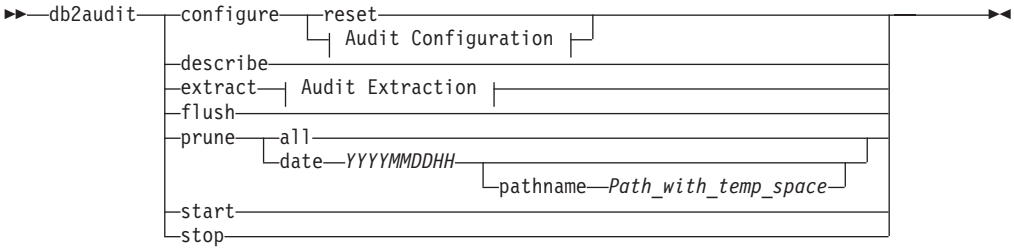
- 299 페이지의 『DB2 Universal Database(DB2 UDB) 감사 기능 소개』

관련 참조:

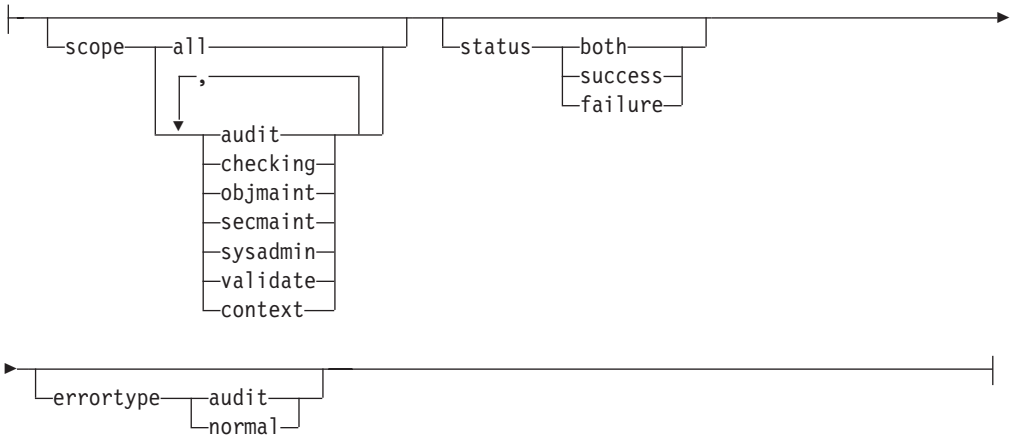
- *관리 안내서: 성능*의 『audit_buf_sz - 감사 버퍼 크기 구성 매개변수』
- 303 페이지의 『감사 기능 사용』

감사 기능 사용

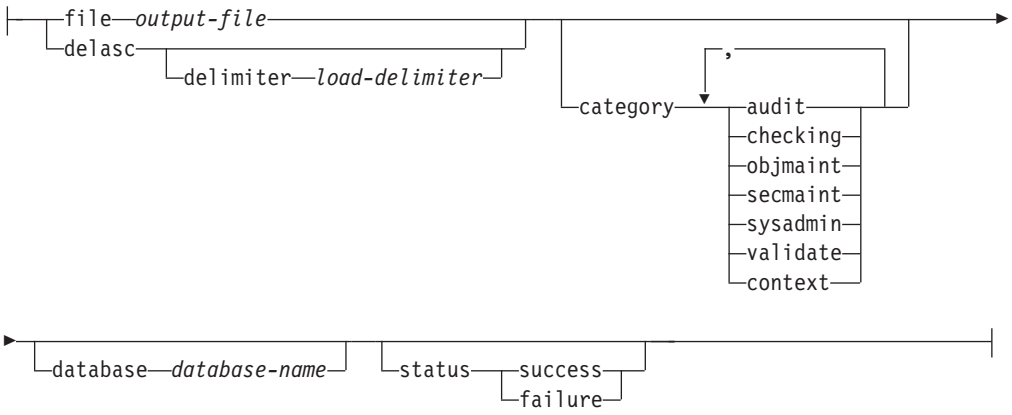
다음 구문 다이어그램의 각 파트를 검토하면, 감사 기능을 사용하는 방법을 이해하는 데 도움이 됩니다.



감사 구성:



감사 발췌:



다음은 각 매개변수의 내재된 사용 및 설명입니다.

configure

이 매개변수를 사용하면 인스턴스의 security 서브디렉토리에 있는 db2audit.cfg 구성 파일을 수정할 수 있습니다. 이 파일에 대한 갱신은 인스턴스가 종료될 때에도 발생할 수 있습니다. 인스턴스가 동적으로 사용 중일 때 발생하는 갱신은 모든 파티션에 걸쳐 DB2 Universal Database™(DB2 UDB)에 의해 수행 중인 감사에 영향을 미칩니다. 감사 기능이 시작될 때 구성 파일을 구성하면 감사 레코드가 작성되고 감사 가능한 이벤트의 *감사* 범주가 감사됩니다.

다음은 구성 파일에서 가능한 조치입니다.

- RESET. 이 조치는 구성 파일이 초기 구성으로 되돌아가게 합니다(여기서, SCOPE는 CONTEXT를 제외한 모든 범주이며, STATUS는 FAILURE이며, ERRORTYPE은 NORMAL이며, 감사 기능은 OFF입니다). 이 조치는 원래의 감사 구성 파일이 유실되거나 손상된 경우 새로운 감사 구성 파일을 작성합니다.
- SCOPE. 이 조치는 이벤트의 어떤 범주가 감사되는지를 지정합니다. 또한, 이 조치는 감사의 특정 초점을 허용하며 로그의 성장을 감소시킵니다. 가능한 한 로그되는 이벤트의 수와 유형을 제한하도록 권장합니다. 그렇지 않으면, 감사 로그가 급격히 증가합니다.

  주: 디폴트값 SCOPE는 CONTEXT를 제외한 모든 범주이며 레코드가 빠르게 생성되는 결과를 가져올 수 있음을 기억하십시오. 모드(동기 또는 비동기)와 결합된, 범주의 선택은 상당한 성능 저하를 가져오며 디스크 요구사항을 상당히 증가시키는 결과를 가져올 수 있습니다.
- STATUS. 이 조치는 성공하거나 실패한 이벤트만 또는 성공 및 실패한 이벤트 둘다를 로그하는지 여부를 지정합니다.

  주: 조작 상태가 알려지기 전에 컨텍스트 이벤트가 발생합니다. 그러므로 이러한 이벤트는 이 매개변수와 연관된 값과 무관하게 로그됩니다.

- ERRORTYPE. 이 조치는 감사 오류가 사용자에게 리턴되는지, 아니면 무시되는지 여부를 지정합니다. 이 매개변수 값은 다음과 같습니다.
  - AUDIT. 감사 기능 내에서 발생하는 오류를 포함하여 모든 오류는 DB2 UDB가 관리하며 모든 음의 SQLCODE는 호출자에게 다시 보고됩니다.
  - NORMAL. db2audit가 생성한 모든 오류는 무시되며 수행 중인 조작과 연관된 오류에 대한 SQLCODE만이 응용프로그램에 리턴됩니다.

describe

이 매개변수는 현재 감사 구성 정보 및 상태를 표준 출력에 표시합니다.

extract

이 매개변수는 감사 로그로부터 표시된 목적지로 감사 레코드의 이동을 허용합니다. 선택적 절이 지정되지 않으면, 모든 감사 레코드가 추출되어 플랫 보고서 파일에 위치합니다. *output_file*이 있는 경우, 오류 메시지가 리턴됩니다.

다음은 추출할 때 사용할 수 있는 가능한 옵션입니다.

- FILE. 추출된 감사 레코드는 파일(*output_file*)에 위치합니다. 파일 이름을 지정하지 않을 경우, 레코드가 `sqllib`의 `security` 서브디렉토리에 있는 `db2audit.out` 파일로 기록됩니다. 디렉토리를 지정하지 않을 경우, *output_file*이 현재 작업 디렉토리로 기록됩니다.
- DELASC. 추출된 감사 레코드는 DB2 UDB 관계형 테이블로 로드하는 데 적합한 컬럼 식별자가 있는 ASCII 형식으로 위치합니다. 출력은 각 범주에 하나씩 분리 파일에 위치합니다. 파일 이름은 다음과 같습니다.
  - `audit.del`
  - `checking.del`
  - `objmaint.del`
  - `secmaint.del`
  - `sysadmin.del`
  - `validate.del`
  - `context.del`

  이들 파일은 항상 `sqllib`의 `security` 서브디렉토리로 기록됩니다.

  DELASC 선택항목은 감사 로그에서 추출할 때 디폴트 감사 문자열 분리문자(『0xff』)를 겹쳐쓸 수 있게 합니다. 감사 레코드를 보류할 테이블로 로드하는 준비에서 사용하려는 새로운 분리문자가 뒤에 DELASC DELIMITER를 사용하십시오. 새로운 로드 분리문자는 단일 문자(예: !) 또는 16진수(예 : 0xff)를 표현하는 4바이트 문자열 중 하나입니다.
- CATEGORY. 감사 이벤트의 지정 범주에 대한 감사 레코드가 추출됩니다. 지정되지 않으면, 모든 범주가 추출 대상이 됩니다.

- DATABASE. 지정 데이터베이스의 감사 레코드가 추출됩니다. 지정되지 않으면, 모든 데이터베이스가 추출 대상이 됩니다.
- STATUS. 지정 상태의 감사 레코드가 추출됩니다. 지정되지 않으면, 모든 레코드가 추출 대상이 됩니다.

flush 이 매개변수는 보류 중인 모든 감사 레코드가 감사 로그에 작성되도록 강제합니다. 또한, 감사 상태는 감사 기능이 오류 상태에 있는 경우 『로그할 수 없음』에서 『로그 준비』 상태로 엔진에서 재설정됩니다.

prune 이 매개변수는 감사 로그에서 감사 레코드의 삭제를 허용합니다. 감사 기능이 사용 중이며 이벤트의 『감사』 범주가 감사용으로 지정된 경우, 감사 레코드는 감사 로그가 프룬(prune)된 후에 로그됩니다.

다음은 제거할 때 사용할 수 있는 옵션입니다.

- ALL. 감사 로그의 모든 감사 레코드가 삭제됩니다.
- DATE yyyymmddhh. 지정한 날짜/시간에 발생하거나 지정한 날짜/시간 이전에 발생한 모든 감사 레코드를 감사 로그에서 삭제하도록 지정할 수 있습니다. 사용자는 감사 로그를 제거할 때 감사 기능이 임시 공간으로서 사용할 다음을

  ```
  pathname
  ```

  선택적으로 제공할 수 있습니다. 이 임시 공간은 감사 로그가 있는 디스크가 가득 차서 제거 조작할 공간이 충분하지 않을 때 감사 로그를 제거할 수 있게 합니다.

start 이 매개변수는 db2audit.cfg 파일의 내용에 근거하여 감사 기능이 감사 이벤트를 시작할 수 있게 합니다. 파티션된 DB2 UDB 인스턴스에서 이 절이 지정되면 모든 파티션에서 감사가 시작됩니다. 이벤트의 『감사』 범주가 감사에 지정된 경우, 감사 기능이 시작될 때 감사 레코드가 로그됩니다.

stop 이 매개변수는 감사 기능이 감사 이벤트를 중지시키게 합니다. 파티션된 DB2 UDB 인스턴스에서 이 절이 지정되면 모든 파티션에서 감사가 중지됩니다. 이벤트의 『감사』 범주가 감사에 지정된 경우, 감사 기능이 중지될 때 감사 레코드가 로그됩니다.

관련 개념:

- 299 페이지의 『DB2 Universal Database(DB2 UDB) 감사 기능 소개』
- 335 페이지의 『감사 기능 추가 정보 및 기술』

관련 참조:

- *Command Reference*의 『db2audit - Audit Facility Administrator Tool Command』

DB2 테이블에서 DB2 감사 데이터에 대한 작업

다음 주제는 DB2 감사 데이터 작성 방법, 해당 데이터를 보유할 테이블 작성 방법, DB2 감사 데이터로 테이블을 채우는 방법 및 테이블에서 DB2 감사 데이터 선택 방법을 설명합니다.

DB2 테이블에서 DB2 감사 데이터에 대한 작업

DB2 감사 기능을 사용하여 데이터베이스 활동의 감사 추적을 유지보수할 경우, 디폴트로 감사 기능은 감사 레코드를 로그 파일에 배치합니다. 원하는 경우, 로그 파일의 감사 레코드를 텍스트 파일에 기록하거나 로그 파일의 감사 레코드를 컬럼 식별자가 있는 ASCII 파일로 기록한 다음, ASCII 파일의 내용을 DB2 테이블에 로드할 수 있습니다. 감사 데이터가 DB2 테이블에 있을 경우, DB2 인스턴스에 관한 활동에 대해 있을 수 있는 의문점에 응답이 될 데이터를 테이블에서 선택할 수 있습니다.

프로시저:

DB2 테이블의 감사 데이터에 대해 작업하려면, 다음을 수행하십시오.

1. DB2 감사 데이터를 보유할 테이블을 작성하십시오.
2. DB2 감사 데이터 파일을 작성하십시오.
3. 로드 유틸리티를 사용하여 테이블에 데이터를 채우십시오.
4. 테이블 데이터를 선택하십시오.

관련 개념:

- 301 페이지의『감사 기능 동작』
- 335 페이지의『감사 기능 추가 정보 및 기술』

태스크 관련:

- 311 페이지의『DB2 감사 데이터 파일 작성』
- 307 페이지의『DB2 감사 데이터를 보유할 테이블 작성』
- 312 페이지의『테이블에 DB2 감사 데이터 로드』
- 315 페이지의『테이블에서 DB2 감사 데이터 선택』

관련 참조:

- 303 페이지의『감사 기능 사용』

DB2 감사 데이터를 보유할 테이블 작성

테이블의 감사 데이터에 대해 작업하기 전에, 데이터를 보유할 테이블을 작성해야 합니다. 테이블의 데이터를 권한이 없는 사용자로부터 격리하려면 별도의 스키마에 이들 테이블을 작성해야 합니다.

전제조건:

- 스키마를 작성하는 데 필요한 권한 및 특권에 대해서는 CREATE SCHEMA문을 참조하십시오.
- 테이블을 작성하는 데 필요한 권한 및 특권에 대해서는 CREATE TABLE문을 참조하십시오.
- 테이블을 보유하는 데 사용할 테이블 스페이스를 결정하십시오. (이 주제에서는 테이블 스페이스를 작성하는 방법을 설명하지 않습니다.)

프로시저:

다음에 나오는 예는 모든 ASCII 파일의 모든 레코드를 보유할 테이블을 작성하는 방법을 보여줍니다. 원하는 경우, 이들 테이블을 포함할 별도의 스키마를 작성할 수 있습니다.

파일에 포함된 모든 데이터를 사용하지 않으려는 경우, 필요에 따라 테이블 정의에서 컬럼을 생략하거나 테이블 작성을 통과할 수 있습니다. 테이블 정의에서 컬럼을 생략하는 경우, 이들 테이블에 데이터를 로드하는 데 사용하는 명령을 수정해야 합니다.

1. DB2 명령 창을 열려면 **db2** 명령을 발행하십시오.
2. 선택적. 테이블을 보유할 스키마를 작성하십시오. 다음 명령을 발행하십시오. 이 예의 경우, 스키마는 AUDIT입니다.

   ```
   CREATE SCHEMA AUDIT
   ```

3. 선택적. AUDIT 스키마를 작성한 경우, 테이블을 작성하기 전에 스키마로 전환하십시오. 다음 명령을 발행하십시오:

   ```
   SET CURRENT SCHEMA = 'AUDIT'
   ```

4. audit.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL문을 발행하십시오.

   ```
   CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128))
   ```

5. checking.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL문을 발행하십시오.

   ```
   CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                          CATEGORY CHAR(8),
                          EVENT VARCHAR(32),
                          CORRELATOR INTEGER,
                          STATUS INTEGER,
                          DATABASE CHAR(8),
                          USERID VARCHAR(1024),
                          AUTHID VARCHAR(128),
                          NODENUM SMALLINT,
   ```

```
                                    COORDNUM SMALLINT,
                                    APPID VARCHAR(255),
                                    APPNAME VARCHAR(1024),
                                    PKGSCHEMA VARCHAR(128),
                                    PKGNAME VARCHAR(128),
                                    PKGSECNUM SMALLINT,
                                    OBJSCHEMA VARCHAR(128),
                                    OBJNAME VARCHAR(128),
                                    OBJTYPE VARCHAR(32),
                                    ACCESSAPP CHAR(18),
                                    ACCESSATT CHAR(18),
                                    PKGVER VARCHAR(64))
```

6. objmaint.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL 문을 발행하십시오.

```
CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       PACKVER VARCHAR(64))
```

7. secmaint.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL 문을 발행하십시오.

```
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       GRANTOR VARCHAR(128),
                       GRANTEE VARCHAR(128),
```

```
                                GRANTEETYPE VARCHAR(32),
                                PRIVAUTH CHAR(18),
                                PKGVER VARCHAR(64))
```

8. sysadmin.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL 문을 발행하십시오.

```
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64))
```

9. validate.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL 문을 발행하십시오.

```
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       EXECID VARCHAR(1024),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       AUTHTYPE VARCHAR(32),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64)
                       PLUGINNAME VARCHAR(32))
```

10. context.del 파일의 레코드를 포함할 테이블을 작성하려면, 다음과 같은 SQL문을 발행하십시오.

```
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
```

```
APPID VARCHAR(255),
APPNAME VARCHAR(1024),
PKGSCHEMA VARCHAR(128),
PKGNAME VARCHAR(128),
PKGSECNUM SMALLING,
STMTTEXT CLOB(2M),
PKGVER VARCHAR(64))
```

11. 테이블 작성 후, COMMIT문을 발행하여 테이블 정의가 디스크에 기록되었는지 확인하십시오.
12. 테이블을 작성하였으면, db2audit.log 파일의 감사 레코드를 컬럼 식별자가 있는 ASCII 파일로 추출할 준비가 완료됩니다.

태스크 관련:

- 101 페이지의 『스키마 설정』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『CREATE SCHEMA문』
- *SQL 참조서, 볼륨 2*의 『CREATE TABLE문』

DB2 감사 데이터 파일 작성

디폴트로, DB2 감사 기능은 감사 데이터를 db2audit.log 파일에 기록합니다. 이 파일의 레코드를 테이블에 로드할 수 없습니다. 감사 레코드를 컬럼 식별자가 있는 ASCII 파일로 추출해야 하며, 이를 사용하여 테이블을 채울 수 있습니다.

전제조건:

db2audit 명령을 사용하려면 SYSADM 권한이 필요합니다.

프로시저:

감사 기능 레코드를 컬럼 식별자가 있는 ASCII 파일에 기록하려면, 다음을 수행하십시오.

1. 감사 기능 사용에 관한 주제를 검토하여 감사하고자 하는 DB2 활동의 유형을 판별하십시오. 감사 기능에 대해 설정한 구성에 만족할 경우, 다음 명령을 발행하여 감사를 시작하십시오.

   ```
   db2audit start
   ```

2. 다음 명령을 발행하여 모든 감사 레코드가 메모리로부터 db2audit.log 파일로 옮겨졌는지 확인하십시오.

   ```
   db2audit flush
   ```

3. 다음 명령을 발행하여 db2audit.log의 감사 레코드를 컬럼 식별자가 있는 ASCII 파일로 이동하십시오.

   ```
   db2audit extract delasc
   ```

다음 파일은 sqllib의 security 서브디렉토리에 작성됩니다. 특정 이벤트 유형을 감사 중이 아닐 경우, 해당 이벤트에 대한 파일이 작성되지만 파일은 비어 있습니다.

- audit.del
- checking.del
- objmaint.del
- secmaint.del
- sysadmin.del
- validate.del
- context.del

4. 다음 명령을 발행하여 db2audit.log 파일에서 방금 추출한 감사 레코드를 삭제하십시오.

   ```
   db2audit prune date YYYYMMDDHH
   ```

   여기서, *YYYYMMDDHH*는 현재 연도, 월, 일 및 시입니다. 감사 데이터로 테이블을 채울 때 다음 단계에서 이 정보가 필요하므로 사용한 값을 기록하십시오.

   감사 기능은 계속해서 새로운 감사 레코드를 db2audit.log 파일에 기록하며, 이들 레코드는 *YYYYMMDDHH* 이후의 시간소인을 갖습니다. 이미 추출한 db2audit.log 파일에서 레코드를 프룬(prune)하면 동일한 레코드를 두 번 추출할 수 없습니다. *YYYYMMDDHH* 이후 기록된 모든 감사 레코드는 다음 번에 감사 데이터를 추출할 때 .del 파일에 기록됩니다.

5. 감사 데이터 파일을 작성한 후, 다음 단계는 로드 유틸리티를 사용하여 감사 데이터로 테이블 채우기.

관련 참조:

- *Command Reference*의 『db2audit - Audit Facility Administrator Tool Command』
- 303 페이지의 『감사 기능 사용』
- 318 페이지의 『AUDIT 이벤트에 대한 감사 레코드 레이아웃』
- 318 페이지의 『CHECKING 이벤트에 대한 감사 레코드 레이아웃』
- 324 페이지의 『OBJMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 325 페이지의 『SECMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 330 페이지의 『SYSADMIN 이벤트에 대한 감사 레코드 레이아웃』
- 333 페이지의 『VALIDATE 이벤트에 대한 감사 레코드 레이아웃』
- 334 페이지의 『CONTEXT 이벤트에 대한 감사 레코드 레이아웃』

테이블에 DB2 감사 데이터 로드

감사 데이터를 보유할 테이블을 작성하였으면, 이제 ASCII 파일의 데이터를 테이블에 로드합니다.

전제조건:

자세한 정보는 로드 유틸리티를 사용하는 데 필요한 특권, 권한 및 권한 부여에 관한 주제를 참조하십시오.

프로시저:

로드 유틸리티를 사용하여 데이터를 테이블에 로드하십시오. 각각의 테이블에 대하여 별도의 로드 명령을 발행하십시오. 테이블 정의에서 하나 이상의 컬럼을 생략한 경우, 데이터를 성공적으로 로드하는 데 사용되는 LOAD 명령의 버전을 수정해야 합니다. 또한 감사 데이터를 추출할 때 디폴트값(0xff)이 아닌 분리문자를 지정한 경우, 사용하는 LOAD 명령의 버전을 수정해야 합니다(자세한 정보는 "로드에 대한 파일 유형 수정자" 주제를 참조하십시오).

1. DB2 명령 창을 열려면 **db2** 명령을 발행하십시오.
2. AUDIT 테이블을 로드하려면, 다음 명령을 발행하십시오.

   ```
   LOAD FROM audit.del OF del MODIFIED BY CHARDEL0xff INSERT INTO schema.AUDIT
   ```

   주: 파일 이름 지정 시, 완전한 경로 이름을 사용하십시오. 예를 들어, Windows 기반 컴퓨터의 C: 드라이브에 DB2 UDB를 설치한 경우, audit.del 파일에 대한 완전한 파일 이름으로 C:\Program Files\IBM\SQLLIB*instance*\security\audit.del을 지정합니다.

   AUDIT 테이블을 로드한 후, 다음과 같은 DELETE문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오. db2audit.log 파일에서 감사 레코드를 추출한 경우, 파일의 모든 레코드는 .del 파일에 기록됩니다. 마찬가지로 .del 파일은 감사 로그가 이어서 프룬(prune)된 시간 이후 기록된 레코드를 포함합니다(**db2audit prune** 명령은 레코드를 지정된 시간으로만 프룬하므로). 다음 번에 감사 레코드를 추출하면, 새로운 .del 파일이 이전에 추출되었으나 **db2audit prune** 명령으로 삭제되지 않은(프룬 조작에 지정된 시간 이후 기록되었으므로) 레코드를 포함합니다. db2audit.log 파일이 프룬된 동일한 시간까지 테이블에서 행을 삭제하면 중복된 행이 포함되지 않으며, 감사 레코드가 손실되지 않습니다.

   ```
   DELETE FROM schema.AUDIT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   여기서, *YYYYMMDDHH*는 db2audit.log 파일을 프룬할 때 지정한 값입니다. DB2 감사 기능은 프룬 이후 계속해서 감사 레코드를 db2audit.log 파일에 기록하므로, db2audit.log 파일이 프룬된 후 기록된 감사 레코드가 테이블에서 삭제되지 않게 하려면 분과 초에 대해 0000을 지정해야 합니다.
3. CHECKING 테이블을 로드하려면, 다음 명령을 발행하십시오.

   ```
   LOAD FROM checking.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
     schema.CHECKING
   ```

CHECKING 테이블을 로드한 후, 다음과 같은 SQL문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오.

```
DELETE FROM schema.CHECKING WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

여기서, *YYYYMMDDHH*는 로그 파일을 프룬할 때 지정한 값입니다.

4. OBJMAINT 테이블을 로드하려면, 다음 명령을 발행하십시오.

```
LOAD FROM objmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.OBJMAINT
```

OBJMAINT 테이블을 로드한 후, 다음과 같은 SQL문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오.

```
DELETE FROM schema.OBJMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

여기서, *YYYYMMDDHH*는 로그 파일을 프룬할 때 지정한 값입니다.

5. SECMAINT 테이블을 로드하려면, 다음 명령을 발행하십시오.

```
LOAD FROM secmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.SECMAINT
```

SECMAINT 테이블을 로드한 후, 다음과 같은 SQL문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오.

```
DELETE FROM schema.SECMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

여기서, *YYYYMMDDHH*는 로그 파일을 프룬할 때 지정한 값입니다.

6. SYSADMIN 테이블을 로드하려면, 다음 명령을 발행하십시오.

```
LOAD FROM sysadmin.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.SYSADMIN
```

SYSADMIN 테이블을 로드한 후, 다음과 같은 SQL문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오.

```
DELETE FROM schema.SYSADMIN WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

여기서, *YYYYMMDDHH*는 로그 파일을 프룬할 때 지정한 값입니다.

7. VALIDATE 테이블을 로드하려면, 다음 명령을 발행하십시오.

```
LOAD FROM validate.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.VALIDATE
```

VALIDATE 테이블을 로드한 후, 다음과 같은 SQL문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오.

```
DELETE FROM schema.VALIDATE WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

여기서, *YYYYMMDDHH*는 로그 파일을 프룬할 때 지정한 값입니다.

8. CONTEXT 테이블을 로드하려면, 다음 명령을 발행하십시오.

```
LOAD FROM context.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.CONTEXT
```

CONTEXT 테이블을 로드한 후, 다음과 같은 SQL문을 발행하여 다음 번에 로드할 때 중복된 행이 테이블에 로드되지 않도록 하십시오.

```
DELETE FROM schema.CONTEXT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

여기서, *YYYYMMDDHH*는 로그 파일을 프루할 때 지정한 값입니다.

9. 테이블에 데이터 로드를 완료한 후, sqllib 디렉토리의 security 서브디렉토리에서 .del 파일을 삭제하십시오.
10. 테이블에 감사 데이터를 로드했으면, 테이블에서 데이터를 선택할 준비가 완료된 것입니다.

테이블을 이미 채웠으며 다시 이를 수행하려는 경우, INSERT 옵션을 사용하여 새 테이블 데이터를 기존 테이블 데이터에 추가하십시오. 이전 **db2audit extract** 조작의 레코드를 테이블에서 제거하려는 경우, REPLACE 옵션을 사용하여 테이블을 다시 로드하십시오. 어느 경우든지, 반드시 레코드를 .del 파일로 추출하기 전에 감사 레코드를 db2audit.log 파일로 비우고 레코드를 추출한 후 동일한 레코드가 테이블에 두 번 이상 로드되지 않도록 db2audit.log 파일을 프루하십시오.

태스크 관련:

- 307 페이지의『DB2 테이블에서 DB2 감사 데이터에 대한 작업』

테이블에서 DB2 감사 데이터 선택

감사 데이터가 테이블에 성공적으로 로드된 경우, 추가 분석을 위해 테이블에서 데이터를 선택할 수 있습니다.

전제조건:

테이블에서 데이터를 선택하는 데 필요한 권한 및 특권에 관한 정보는 SELECT문에 관한 주제를 참조하십시오.

프로시저:

테이블에서 모든 행을 선택하려면, 다음을 수행하십시오.

1. DB2 명령 창을 열려면 **db2** 명령을 발행하십시오.
2. 감사 데이터를 선택할 각각의 테이블에 대하여 다음과 같은 양식의 SQL문을 발행하십시오.

```
SELECT * FROM schema.table
```

예를 들어, AUDIT 스키마의 CHECKING 테이블에서 모든 데이터를 선택하려면, 다음 명령문을 사용하십시오.

```
SELECT * FROM AUDIT.CHECKING
```

수행하는 SELECT문은 데이터에 관하여 수행하려는 분석의 유형을 반영해야 합니다. 예를 들어, 해당 권한 부여 ID가 수행 중이던 활동의 유형을 판별하려면 권한 부여 ID(authid)에 따라 레코드를 선택할 수 있습니다.

```
SELECT * FROM AUDIT.CHECKING WHERE AUTHID = authorization ID
```

여기서, *authorization ID*는 데이터를 분석하려는 사용자 ID입니다.

감사 데이터에 포함시킬 수 있는 값에 대한 설명은 테이블에 대한 해당하는 감사 레코드 배치 주제 및 테이블에 대한 가능한 리턴 값의 목록을 참조하십시오.

관련 참조:

- *SQL 참조서, 볼륨 1*의 『Subselect』
- *SQL 참조서, 볼륨 2*의 『SELECT문』
- 318 페이지의 『AUDIT 이벤트에 대한 감사 레코드 레이아웃』
- 318 페이지의 『CHECKING 이벤트에 대한 감사 레코드 레이아웃』
- 321 페이지의 『가능한 CHECKING 액세스 승인 이유 목록』
- 322 페이지의 『가능한 CHECKING 액세스 시도 유형 목록』
- 324 페이지의 『OBJMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 325 페이지의 『SECMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 326 페이지의 『가능한 SECMAINT 특권 또는 권한 목록』
- 330 페이지의 『SYSADMIN 이벤트에 대한 감사 레코드 레이아웃』
- 331 페이지의 『가능한 SYSADMIN 감사 이벤트 목록』
- 333 페이지의 『VALIDATE 이벤트에 대한 감사 레코드 레이아웃』
- 334 페이지의 『CONTEXT 이벤트에 대한 감사 레코드 레이아웃』
- 334 페이지의 『가능한 CONTEXT 감사 이벤트 목록』

감사 기능 메시지

SQL1322N 감사 로그 파일에 기록할 때 오류가 발생했습니다.

설명: 감사 이벤트를 감사 로그 파일에 기록하도록 DB2 Universal Database™(DB2 UDB) 감사 기능이 호출될 때 오류가 발생했습니다. 감사 로그가 있는 파일 시스템에 공간이 없습니다.

사용자 응답: 시스템 관리자는 이 파일 시스템에서 공간을 해제하거나 감사 로그의 크기를 줄이도록 프룬(prune) 해야 합니다.

더 많은 공간이 사용 가능하면, db2audit를 사용하여 메모리의 데이터를 비우고 감사 프로그램을 준비 상태로 재설정하십시오. 삭제한 레코드를 복구할 수 없으면, 해당 추출이 발생했는지 또는 로그를 제거하기 전에 로그 사본을 작성했는지 확인하십시오.

sqlcode: -1322

sqlstate: 50830

SQL1323N 감사 구성 파일에 액세스할 때 오류가 발생했습니다.

설명: 감사 구성 파일(db2audit.cfg)을 열 수 없거나 올바르지 않습니다. 이 오류는 db2audit.cfg 파일이 없거나 손상될 때 발생할 수 있습니다.

사용자 응답: 다음 조치 중 하나를 취하십시오.

- 파일의 저장 버전에서 리스토어하십시오.
- 다음을 발행하여 감사 기능 구성 파일을 재설정하십시오.

```
db2audit reset
```

sqlcode: -1323

sqlstate: 57019

관련 개념:

- 299 페이지의『DB2 Universal Database(DB2 UDB) 감사 기능 소개』

감사 기능 레코드 레이아웃(소개)

DELASC 추출 옵션을 사용하여 감사 로그에서 감사 레코드가 추출되면, 각 레코드는 다음 테이블에 표시된 형식 중 하나를 가집니다. 각 테이블은 샘플 레코드의 내용을 표시하여 시작합니다. 레코드의 각 항목 설명이 해당 테이블에서 한 번에 하나의 행에 표시됩니다. 항목이 중요하면, 해당 항목의 이름이 강조표시(**굵은체**)됩니다. 이들 항목에는 사용자에게 가장 관심 있는 정보가 들어 있습니다.

주:

1. 샘플 레코드의 모든 필드가 값을 갖지는 않습니다.
2. 『액세스가 시도됨』과 같은 일부 필드는 비트맵으로서 컬럼 식별자가 있는 ASCII 형식으로 저장됩니다. 그러나 이 플랫 보고서 파일에서 이들 필드는 비트맵 값을 표현하는 문자열 세트로서 나타납니다.
3. CHECKING, OBJMAINT, SECMAINT, SYSADMIN, VALIDATE 및 CONTEXT 이벤트에 대한 레코드 레이아웃에『패키지 버전』이라는 새 필드가 추가되었습니다.

관련 참조:

- 318 페이지의『AUDIT 이벤트에 대한 감사 레코드 레이아웃』
- 318 페이지의『CHECKING 이벤트에 대한 감사 레코드 레이아웃』
- 324 페이지의『OBJMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 325 페이지의『SECMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 330 페이지의『SYSADMIN 이벤트에 대한 감사 레코드 레이아웃』
- 333 페이지의『VALIDATE 이벤트에 대한 감사 레코드 레이아웃』
- 334 페이지의『CONTEXT 이벤트에 대한 감사 레코드 레이아웃』

감사 기능 레코드 레이아웃의 세부사항

이 섹션에서는 다양한 감사 기능 레코드 레이아웃을 보여줍니다.

AUDIT 이벤트에 대한 감사 레코드 레이아웃

표 7. AUDIT 이벤트에 대한 감사 레코드 레이아웃

timestamp=1998-06-24-11.54.05.151232;**category=AUDIT;audit event=START;**
event correlator=0;event status=0;
userid=boss;authid=BOSS;		

이름	**형식**	**설명**
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. AUDIT
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 CONFIGURE, DB2AUD, EXTRACT, FLUSH, PRUNE, START, STOP 및 UPDATE_ADMIN_CFG입니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.
이벤트 상태	INTEGER	SQLCODE가 표시하는 감사 이벤트의 상태 성공 이벤트 > = 0 실패 이벤트 < 0
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

CHECKING 이벤트에 대한 감사 레코드 레이아웃

표 8. CHECKING 이벤트에 대한 감사 레코드 레이아웃

timestamp=1998-06-24-08.42.11.622984;**category=CHECKING;audit event=CHECKING_OBJECT;**
event correlator=2;**event status=0;**
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SYSSH200;
package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
access approval reason=DBADM;access attempted=STORE;		

이름	**형식**	**설명**
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. CHECKING
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 CHECKING_OBJECT 및 CHECKING_FUNCTION입니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.

표 8. CHECKING 이벤트에 대한 감사 레코드 레이아웃 (계속)

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SYSSH200;
  package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
  access approval reason=DBADM;access attempted=STORE;
```

이름	형식	설명
이벤트 상태	INTEGER	SQLCODE가 표시하는 감사 이벤트의 상태 성공 이벤트 > = 0 실패 이벤트 < 0
데이터베이스 이름	CHAR(8)	이벤트가 생성된 데이터베이스의 이름. 이것이 인스턴스 레벨 감사 이벤트인 경우 공백입니다.
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID
원래 노드 번호	SMALLINT	감사 이벤트가 발생한 노드 번호
코디네이터 노드 번호	SMALLINT	코디네이터 노드의 노드 번호
응용프로그램 ID	VARCHAR(255)	감사 이벤트 발생시 사용 중인 응용프로그램 ID
응용프로그램 이름	VARCHAR(1024)	감사 이벤트 발생시 사용 중인 응용프로그램 이름
패키지 스키마	VARCHAR(128)	감사 이벤트시 사용 중인 패키지의 스키마
패키지 이름	VARCHAR(128)	감사 이벤트 발생시 사용 중인 패키지 이름
패키지 섹션 번호	SMALLINT	감사 이벤트 발생시 사용 중인 패키지의 섹션 번호
오브젝트 스키마	VARCHAR(128)	감사 이벤트가 생성된 오브젝트의 스키마
오브젝트 이름	VARCHAR(128)	감사 이벤트가 생성된 오브젝트의 이름
오브젝트 유형	VARCHAR(32)	감사 이벤트가 생성된 오브젝트의 유형. 가능한 값은 『감사 레코드 오브젝트 유형』이라는 주제에서 표시됩니다.
액세스 승인 이유	CHAR(18)	액세스가 이 감사 이벤트에 승인된 이유를 나타냅니다. 가능한 값은 『가능한 CHECKING 액세스 승인 이유 목록』이라는 주제에서 표시됩니다.
액세스가 시도됨	CHAR(18)	시도된 액세스 유형을 나타냅니다. 가능한 값은 『가능한 CHECKING 액세스 시도 유형』이라는 주제에서 표시됩니다.
패키지 버전	VARCHAR (64)	감사 이벤트 발생시 사용 중인 패키지 버전.

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

관련 참조:

- 321 페이지의 『가능한 CHECKING 액세스 승인 이유 목록』
- 322 페이지의 『가능한 CHECKING 액세스 시도 유형 목록』
- 320 페이지의 『감사 레코드 오브젝트 유형』

감사 레코드 오브젝트 유형

표 9. 감사 이벤트를 기초로 한 감사 레코드 오브젝트 유형

오브젝트 유형	CHECKING 이벤트	OBJMAINT 이벤트	SECMAINT 이벤트
NONE	X	X	X
TABLE	X	X	X
VIEW	X	X	X
ALIAS	X	X	
FUNCTION	X	X	X
INDEX	X	X	X
INDEX EXTENSION		X	
PACKAGE	X	X	X
PACKAGE CACHE	X		
DATA_TYPE		X	
NODEGROUP	X	X	
SCHEMA	X	X	X
STORED_PROCEDURE	X	X	X
METHOD_BODY	X	X	X
BUFFERPOOL	X	X	
SEQUENCE	X	X	
TABLESPACE	X	X	X
EVENT_MONITOR	X	X	
TRIGGER		X	
DATABASE	X		X
INSTANCE	X		
FOREIGN_KEY		X	
PRIMARY_KEY		X	
UNIQUE_CONSTRAINT		X	
CHECK_CONSTRAINT		X	
WRAPPER	X	X	
SERVER	X	X	X
NICKNAME	X	X	X
USER MAPPING	X	X	
SERVER OPTION	X	X	
TYPE&TRANSFORM	X	X	
TYPE MAPPING	X	X	
FUNCTION MAPPING	X	X	
SUMMARY TABLES	X	X	X
JAR_FILE		X	
ALL	X		
REOPT_VALUES	X		

관련 참조:

- 318 페이지의『CHECKING 이벤트에 대한 감사 레코드 레이아웃』
- 324 페이지의『OBJMAINT 이벤트에 대한 감사 레코드 레이아웃』
- 325 페이지의『SECMAINT 이벤트에 대한 감사 레코드 레이아웃』

가능한 CHECKING 액세스 승인 이유 목록

다음은 가능한 CHECKING 액세스 승인 이유의 목록입니다.

0x0000000000000001 ACCESS DENIED
액세스가 승인되지 않고 거부되었습니다.

0x0000000000000002 SYSADM
액세스가 승인되었으며, 응용프로그램/사용자가 SYSADM 권한을 가집니다.

0x0000000000000004 SYSCTRL
액세스가 승인되었으며, 응용프로그램/사용자가 SYSCTRL 권한을 가집니다.

0x0000000000000008 SYSMAINT
액세스가 승인되었으며, 응용프로그램/사용자가 SYSMAINT 권한을 가집니다.

0x0000000000000010 DBADM
액세스가 승인되었으며, 응용프로그램/사용자가 DBADM 권한을 가집니다.

0x0000000000000020 DATABASE PRIVILEGE
액세스가 승인되었으며, 응용프로그램/사용자가 데이터베이스에서 명시적 특권을 가집니다.

0x0000000000000040 OBJECT PRIVILEGE
액세스가 승인되었으며, 응용프로그램/사용자가 오브젝트 또는 기능에서 명시적 특권을 가집니다.

0x0000000000000080 DEFINER
액세스가 승인되었으며, 응용프로그램/사용자가 오브젝트 또는 기능의 정의자입니다.

0x0000000000000100 OWNER
액세스가 승인되었으며, 응용프로그램/사용자가 오브젝트 또는 기능의 소유자입니다.

0x0000000000000200 CONTROL
액세스가 승인되었으며, 응용프로그램/사용자가 오브젝트 또는 기능에서 CONTROL 특권을 가집니다.

0x0000000000000400 BIND
액세스가 승인되었으며, 응용프로그램/사용자가 패키지에서 바인드 특권을 가집니다.

0x0000000000000800 SYSQUIESCE

액세스가 승인되었으며, 인스턴스 또는 데이터베이스가 Quiesce 모드에 있으면 응용프로그램/사용자가 연결 또는 접속할 수도 있습니다.

0x0000000000001000 SYSMON

액세스가 승인되었으며, 응용프로그램/사용자가 SYSMON 권한을 가집니다.

관련 참조:

- 318 페이지의 『CHECKING 이벤트에 대한 감사 레코드 레이아웃』
- 322 페이지의 『가능한 CHECKING 액세스 시도 유형 목록』

가능한 CHECKING 액세스 시도 유형 목록

다음은 가능한 CHECKING 액세스가 시도됨 유형의 목록입니다.

0x0000000000000002 ALTER

오브젝트 변경 시도

0x0000000000000004 DELETE

오브젝트 삭제 시도

0x0000000000000008 INDEX

인덱스 사용 시도

0x0000000000000010 INSERT

오브젝트로 삽입 시도

0x0000000000000020 SELECT

테이블 또는 뷰의 쿼리 시도

0x0000000000000040 UPDATE

오브젝트에서의 데이터 갱신 시도

0x0000000000000080 REFERENCE

오브젝트 간의 참조 제한조건 설정 시도

0x0000000000000100 CREATE

오브젝트 작성 시도

0x0000000000000200 DROP

오브젝트 제거 시도

0x0000000000000400 CREATEIN

또다른 스키마 내에서 오브젝트 작성 시도

0x0000000000000800 DROPIN

또다른 스키마 내에서 발견된 오브젝트 제거 시도

0x0000000000001000 ALTERIN

또다른 스키마 내에서 발견된 오브젝트 변경 또는 수정 시도

0x0000000000002000 EXECUTE

응용프로그램 실행 시도, 또는 루틴 호출, 해당 루틴을 소스로 하는 함수 작성 (함수에만 적용됨) 또는 DDL문에 루틴 참조 시도.

0x0000000000004000 BIND

응용프로그램 바인드 또는 준비 시도

0x0000000000008000 SET EVENT MONITOR

이벤트 모니터 스위치 설정 시도

0x0000000000010000 SET CONSTRAINTS

오브젝트에서 제한조건 설정 시도

0x0000000000020000 COMMENT ON

오브젝트에서 주석 작성 시도

0x0000000000040000 GRANT

오브젝트의 특권을 또다른 사용자 ID로 권한 부여 시도

0x0000000000080000 REVOKE

사용자 ID에서 오브젝트의 특권 권한 취소 시도

0x0000000000100000 LOCK

오브젝트 잠금 시도

0x0000000000200000 RENAME

오브젝트 이름 바꾸기 시도

0x0000000000400000 CONNECT

오브젝트 연결 시도

0x0000000000800000 Member of SYS Group

SYS 그룹의 구성원에 액세스하거나 사용하려고 시도

0x0000000001000000 Access All

보유된 오브젝트에서 모든 필수 특권으로 명령문 실행 시도(DBADM/SYSADM에만 사용)

0x0000000002000000 Drop All

다중 오브젝트 제거 시도

0x0000000004000000 LOAD

테이블 스페이스에서 테이블 로드 시도

0x0000000008000000 USE

테이블 스페이스에서 테이블 작성 시도

0x0000000010000000 SET SESSION_USER

SET SESSION_USER문 실행 시도.

0x0000000020000000 FLUSH

FLUSH문 실행 시도

0x0000000040000000 STORE

EXPLAIN_PREDICATE 테이블에서 다시 최적화된 명령문의 값을 보기 위한 시도.

관련 참조:

- 318 페이지의『CHECKING 이벤트에 대한 감사 레코드 레이아웃』
- 321 페이지의『가능한 CHECKING 액세스 승인 이유 목록』

OBJMAINT 이벤트에 대한 감사 레코드 레이아웃

표 10. OBJMAINT 이벤트에 대한 감사 레코드 레이아웃

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

이름	형식	설명
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. OBJMAINT
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 CREATE_OBJECT, RENAME_OBJECT 및 DROP_OBJECT입니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.
이벤트 상태	INTEGER	SQLCODE가 표시하는 감사 이벤트의 상태 성공 이벤트 > = 0 실패 이벤트 < 0
데이터베이스 이름	CHAR(8)	이벤트가 생성된 데이터베이스의 이름. 이것이 인스턴스 레벨 감사 이벤트인 경우 공백입니다.
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID
원래 노드 번호	SMALLINT	감사 이벤트가 발생한 노드 번호
코디네이터 노드 번호	SMALLINT	코디네이터 노드의 노드 번호
응용프로그램 ID	VARCHAR(255)	감사 이벤트 발생시 사용 중인 응용프로그램 ID
응용프로그램 이름	VARCHAR(1024)	감사 이벤트 발생시 사용 중인 응용프로그램 이름
패키지 스키마	VARCHAR(128)	감사 이벤트시 사용 중인 패키지의 스키마

표 10. OBJMAINT 이벤트에 대한 감사 레코드 레이아웃 (계속)

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

이름	형식	설명
패키지 이름	VARCHAR(128)	감사 이벤트 발생시 사용 중인 패키지 이름
패키지 섹션 번호	SMALLINT	감사 이벤트 발생시 사용 중인 패키지의 섹션 번호
오브젝트 스키마	VARCHAR(128)	감사 이벤트가 생성된 오브젝트의 스키마
오브젝트 이름	VARCHAR(128)	감사 이벤트가 생성된 오브젝트의 이름
오브젝트 유형	VARCHAR(32)	감사 이벤트가 생성된 오브젝트의 유형. 가능한 값은 『감사 레코드 오브젝트 유형』이라는 주제에서 표시됩니다.
패키지 버전	VARCHAR (64)	감사 이벤트 발생시 사용 중인 패키지 버전.

관련 개념:

- 299 페이지의 『DB2 Universal Database(DB2 UDB) 감사 기능 소개』

관련 참조:

- 320 페이지의 『감사 레코드 오브젝트 유형』

SECMAINT 이벤트에 대한 감사 레코드 레이아웃

표 11. SECMAINT 이벤트에 대한 감사 레코드 레이아웃

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS; object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

이름	형식	설명
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. SECMAINT
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 GRANT, REVOKE, IMPLICIT_GRANT, IMPLICIT_REVOKE, SET_SESSION_USER, 및 UPDATE_DBM_CFG 입니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.
이벤트 상태	INTEGER	SQLCODE가 표시하는 감사 이벤트의 상태 성공 이벤트 > = 0 실패 이벤트 < 0

표 11. SECMAINT 이벤트에 대한 감사 레코드 레이아웃 (계속)

timestamp=1998-06-24-11.57.45.188101;**category=SECMAINT;audit event=GRANT;**
event correlator=4;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.boss.980624155728;application name=db2bp;
package schema=NULLID;package name=SQLC28A1;
package section=0;**object schema=BOSS; object name=T1;object type=TABLE;**
grantor=BOSS;grantee=WORKER;grantee type=USER;**privilege=SELECT;**		

이름	**형식**	**설명**
데이터베이스 이름	CHAR(8)	이벤트가 생성된 데이터베이스의 이름. 이것이 인스턴스 레벨 감사 이벤트인 경우 공백입니다.
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID
원래 노드 번호	SMALLINT	감사 이벤트가 발생한 노드 번호
코디네이터 노드 번호	SMALLINT	코디네이터 노드의 노드 번호
응용프로그램 ID	VARCHAR(255)	감사 이벤트 발생시 사용 중인 응용프로그램 ID
응용프로그램 이름	VARCHAR(1024)	감사 이벤트 발생시 사용 중인 응용프로그램 이름
패키지 스키마	VARCHAR(128)	감사 이벤트시 사용 중인 패키지의 스키마
패키지 이름	VARCHAR(128)	감사 이벤트 발생시 사용 중인 패키지 이름
패키지 섹션 번호	SMALLINT	감사 이벤트 발생시 사용 중인 패키지의 섹션 번호
오브젝트 스키마	VARCHAR(128)	감사 이벤트가 생성된 오브젝트의 스키마
오브젝트 이름	VARCHAR(128)	감사 이벤트가 생성된 오브젝트의 이름
오브젝트 유형	VARCHAR(32)	감사 이벤트가 생성된 오브젝트의 유형. 가능한 값은 『감사 레코드 오브젝트 유형』이라는 주제에서 표시됩니다.
권한 준 사용자	VARCHAR(128)	권한을 부여한 사용자 ID
권한 받은 사용자	VARCHAR(128)	특권 또는 권한이 권한 부여되거나 권한 취소된 권한 받은 사용자 ID
권한 받은 사용자 유형	VARCHAR(32)	권한 부여되거나 권한 취소된 권한 받은 사용자의 유형. 가능한 값은 USER, GROUP 또는 BOTH입니다.
특권 또는 권한	CHAR(18)	특권 또는 권한 유형이 권한 부여되거나 권한 취소되었음을 나타냅니다. 가능한 값은 『가능한 SECMAINT 특권 또는 권한 목록』이라는 주제에서 표시됩니다.
패키지 버전	VARCHAR (64)	감사 이벤트 발생시 사용 중인 패키지 버전.

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

관련 참조:

- 326 페이지의 『가능한 SECMAINT 특권 또는 권한 목록』
- 320 페이지의 『감사 레코드 오브젝트 유형』

가능한 SECMAINT 특권 또는 권한 목록

다음은 가능한 SECMAINT 특권 또는 권한의 목록입니다.

0x0000000000000001 Control Table

테이블 또는 뷰에서 권한 부여되거나 권한 취소된 특권 제어

0x0000000000000002 ALTER TABLE

테이블을 변경하기 위해 권한 부여되거나 권한 취소된 특권

0x0000000000000004 ALTER TABLE with GRANT

허용된 특권 권한 부여로 테이블을 변경하기 위해 권한 부여되거나 권한 취소된 특권

0x0000000000000008 DELETE TABLE

테이블 또는 뷰를 제거하기 위해 권한 부여되거나 권한 취소된 특권

0x0000000000000010 DELETE TABLE with GRANT

허용된 특권 권한 부여로 테이블을 제거하기 위해 권한 부여되거나 권한 취소된 특권

0x0000000000000020 Table Index

인덱스에서 권한 부여되거나 권한 취소된 특권

0x0000000000000040 Table Index with GRANT

허용된 특권 권한 부여로 인덱스에서 권한 부여되거나 권한 취소된 특권

0x0000000000000080 Table INSERT

테이블 또는 뷰의 삽입에서 권한 부여되거나 권한 취소된 특권

0x0000000000000100 Table INSERT with GRANT

허용된 특권 권한 부여로 테이블의 삽입에서 권한 부여되거나 권한 취소된 특권

0x0000000000000200 Table SELECT

테이블의 선택에서 권한 부여되거나 권한 취소된 특권

0x0000000000000400 Table SELECT with GRANT

허용된 특권 권한 부여로 테이블의 선택에서 권한 부여되거나 권한 취소된 특권

0x0000000000000800 Table UPDATE

테이블 또는 뷰의 갱신에서 권한 부여되거나 권한 취소된 특권

0x0000000000001000 Table UPDATE with GRANT

허용된 특권 권한 부여로 테이블 또는 뷰의 갱신에서 권한 부여되거나 권한 취소된 특권

0x0000000000002000 Table REFERENCE

테이블의 참조에서 권한 부여되거나 권한 취소된 특권

0x0000000000004000 Table REFERENCE with GRANT

허용된 특권 권한 부여로 테이블의 참조에서 권한 부여되거나 권한 취소된 특권

0x0000000000020000 CREATEIN Schema

스키마에서 권한 부여되거나 권한 취소된 CREATEIN 특권

0x0000000000040000 CREATEIN Schema with GRANT

허용된 특권 권한 부여로 스키마에서 권한 부여되거나 권한 취소된 CREATEIN 특권

0x0000000000080000 DROPIN Schema

스키마에서 권한 부여되거나 권한 취소된 DROPIN 특권

0x0000000000100000 DROPIN Schema with GRANT

허용된 특권 권한 부여로 스키마에서 권한 부여되거나 권한 취소된 DROPIN 특권

0x0000000000200000 ALTERIN Schema

스키마에서 권한 부여되거나 권한 취소된 ALTERIN 특권

0x0000000000400000 ALTERIN Schema with GRANT

허용된 특권 권한 부여로 스키마에서 권한 부여되거나 권한 취소된 ALTERIN 특권

0x0000000000800000 DBADM Authority

권한 부여되거나 권한 취소된 DBADM 권한

0x0000000001000000 CREATETAB Authority

권한 부여되거나 권한 취소된 Createtab 권한

0x0000000002000000 BINDADD Authority

권한 부여되거나 권한 취소된 Bindadd 권한

0x0000000004000000 CONNECT Authority

권한 부여되거나 권한 취소된 CONNECT 권한

0x0000000008000000 Create not fenced Authority

권한 부여되거나 권한 취소된 분리 권한을 작성하지 않음

0x0000000010000000 Implicit Schema Authority

권한 부여되거나 권한 취소된 내재된 스키마 권한

0x0000000020000000 Server PASSTHRU

이 서버에 대해 pass-through 기능을 사용하기 위해 권한 부여되거나 권한 취소된 특권(페더레이티드 데이터베이스 데이터 소스)

0x0000000100000000 Table Space USE

테이블 스페이스에서 테이블을 작성하기 위해 권한 부여되거나 권한 취소된 특권

0x0000000200000000 Table Space USE with GRANT

특권이 허용된 테이블 스페이스에서 테이블을 작성하기 위해 권한 부여되거나 권한 취소된 특권

0x0000000400000000 Column UPDATE

테이블에 있는 하나 이상의 특정 컬럼에 대한 갱신에서 권한 부여되거나 권한 취소된 특권

0x0000000800000000 Column UPDATE with GRANT

특권이 허용된 테이블의 하나 이상의 특정 컬럼에 대한 갱신에서 권한 부여되거나 권한 취소된 특권

0x0000001000000000 Column REFERENCE

테이블에서 하나 이상의 특정 컬럼에 대한 참조에서 권한 부여되거나 권한 취소된 특권

0x0000002000000000 Column REFERENCE with GRANT

특권이 허용된 테이블의 하나 이상의 특정 컬럼에 대한 참조에서 권한 부여되거나 권한 취소된 특권

0x0000004000000000 LOAD Authority

권한 부여되거나 권한 취소된 LOAD 권한

0x0000008000000000 Package BIND

패키지에서 권한 부여되거나 권한 취소된 BIND 특권

0x0000010000000000 Package BIND with GRANT

특권 부여가 허용된 상태에서 패키지에 대해 부여되거나 취소된 BIND 특권.

0x0000020000000000 EXECUTE

패키지 또는 루틴에 대해 부여되거나 취소된 EXECUTE 특권.

0x0000040000000000 EXECUTE with GRANT

특권 부여가 허용된 상태에서 패키지 또는 루틴에 대해 부여되거나 취소된 EXECUTE 특권.

0x0000080000000000 EXECUTE IN SCHEMA

스키마에 있는 모든 루틴에 대해 부여되거나 취소된 EXECUTE 특권.

0x0000100000000000 EXECUTE IN SCHEMA with GRANT

특권 부여가 허용된 상태에서 스키마에 있는 모든 루틴에 대해 부여되거나 취소된 EXECUTE 특권.

0x0000020000000000 EXECUTE IN TYPE

임의 유형의 모든 루틴에 대해 부여되거나 취소된 EXECUTE 특권.

0x0000400000000000 EXECUTE IN TYPE with GRANT

특권 부여가 허용된 상태에서 임의 유형의 모든 루틴에 대해 부여되거나 취소된 EXECUTE 특권.

0x0000800000000000 CREATE EXTERNAL ROUTINE

부여되거나 취소된 CREATE EXTERNAL ROUTINE 특권.

0x0001000000000000 QUIESCE_CONNECT

부여되거나 취소된 QUIESCE_CONNECT 특권.

관련 참조:

• 325 페이지의『SECMAINT 이벤트에 대한 감사 레코드 레이아웃』

SYSADMIN 이벤트에 대한 감사 레코드 레이아웃

표 12. SYSADMIN 이벤트에 대한 감사 레코드 레이아웃

timestamp=1998-06-24-11.54.04.129923;**category=SYSADMIN;audit event=DB2AUDIT;**
event correlator=1;event status=0;
userid=boss;authid=BOSS;
application id=*LOCAL.boss.980624155404;application name=db2audit;		

이름	**형식**	**설명**
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. SYSADMIN
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 이 표 다음에 오는 목록에 표시됩니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.
이벤트 상태	INTEGER	SQLCODE가 표시하는 감사 이벤트의 상태 성공 이벤트 > = 0 실패 이벤트 < 0
데이터베이스 이름	CHAR(8)	이벤트가 생성된 데이터베이스의 이름. 이것이 인스턴스 레벨 감사 이벤트인 경우 공백입니다.
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID
원래 노드 번호	SMALLINT	감사 이벤트가 발생한 노드 번호
코디네이터 노드 번호	SMALLINT	코디네이터 노드의 노드 번호
응용프로그램 ID	VARCHAR(255)	감사 이벤트 발생시 사용 중인 응용프로그램 ID
응용프로그램 이름	VARCHAR(1024)	감사 이벤트 발생시 사용 중인 응용프로그램 이름
패키지 스키마	VARCHAR(128)	감사 이벤트시 사용 중인 패키지의 스키마
패키지 이름	VARCHAR(128)	감사 이벤트 발생시 사용 중인 패키지 이름
패키지 섹션 번호	SMALLINT	감사 이벤트 발생시 사용 중인 패키지의 섹션 번호
패키지 버전	VARCHAR (64)	감사 이벤트 발생시 사용 중인 패키지 버전.

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

관련 참조:

- 331 페이지의 『가능한 SYSADMIN 감사 이벤트 목록』

가능한 SYSADMIN 감사 이벤트 목록

다음은 가능한 SYSADMIN 감사 이벤트의 목록입니다.

표 13. SYSADMIN 감사 이벤트

START_DB2
STOP_DB2
CREATE_DATABASE
DROP_DATABASE
UPDATE_DBM_CFG
UPDATE_DB_CFG
CREATE_TABLESPACE
DROP_TABLESPACE
ALTER_TABLESPACE
RENAME_TABLESPACE
CREATE_NODEGROUP
DROP_NODEGROUP
ALTER_NODEGROUP
CREATE_BUFFERPOOL
DROP_BUFFERPOOL
ALTER_BUFFERPOOL
CREATE_EVENT_MONITOR
DROP_EVENT_MONITOR
ENABLE_MULTIPAGE
MIGRATE_DB_DIR
DB2TRC
DB2SET
ACTIVATE_DB
ADD_NODE
BACKUP_DB
CATALOG_NODE
CATALOG_DB
CATALOG_DCS_DB
CHANGE_DB_COMMENT
DEACTIVATE_DB
DROP_NODE_VERIFY
FORCE_APPLICATION
GET_SNAPSHOT
LIST_DRDA_INDOUBT_TRANSACTIONS
MIGRATE_DB
RESET_ADMIN_CFG
RESET_DB_CFG
RESET_DBM_CFG
RESET_MONITOR
RESTORE_DB
ROLLFORWARD_DB
SET_RUNTIME_DEGREE
SET_TABLESPACE_CONTAINERS
UNCATALOG_DB
UNCATALOG_DCS_DB
UNCATALOG_NODE
UPDATE_ADMIN_CFG
UPDATE_MON_SWITCHES
LOAD_TABLE
DB2AUDIT
SET_APPL_PRIORITY
CREATE_DB_AT_NODE
KILLDBM
MIGRATE_SYSTEM_DIRECTORY
DB2REMOT
DB2AUD
MERGE_DBM_CONFIG_FILE
UPDATE_CLI_CONFIGURATION
OPEN_TABLESPACE_QUERY
SINGLE_TABLESPACE_QUERY
CLOSE_TABLESPACE_QUERY
FETCH_TABLESPACE
OPEN_CONTAINER_QUERY
FETCH_CONTAINER_QUERY
CLOSE_CONTAINER_QUERY
GET_TABLESPACE_STATISTICS
DESCRIBE_DATABASE
ESTIMATE_SNAPSHOT_SIZE
READ_ASYNC_LOG_RECORD
PRUNE_RECOVERY_HISTORY
UPDATE_RECOVERY_HISTORY
QUIESCE_TABLESPACE
UNLOAD_TABLE
UPDATE_DATABASE_VERSION
CREATE_INSTANCE
DELETE_INSTANCE
SET_EVENT_MONITOR
GRANT_DBADM
REVOKE_DBADM
GRANT_DB_AUTHORITIES
REVOKE_DB_AUTHORITIES
REDIST_NODEGROUP

관련 참조:

• 330 페이지의『SYSADMIN 이벤트에 대한 감사 레코드 레이아웃』

VALIDATE 이벤트에 대한 감사 레코드 레이아웃

표 14. VALIDATE 이벤트에 대한 감사 레코드 레이아웃

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;
  event correlator=2;event status=-1092;
  database=FOO;userid=boss;authid=BOSS;execution id=newton;
application id=*LOCAL.newton.980624124210;application name=testapp;
  auth type=SERVER;
```

이름	형식	설명
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. VALIDATE
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 GET_GROUPS, GET_USERID, AUTHENTICATE_PASSWORD 및 VALIDATE_USER입니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.
이벤트 상태	INTEGER	SQLCODE가 표시하는 감사 이벤트의 상태 성공 이벤트 > = 0 실패 이벤트 < 0
데이터베이스 이름	CHAR(8)	이벤트가 생성된 데이터베이스의 이름. 이것이 인스턴스 레벨 감사 이벤트인 경우 공백입니다.
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID
실행 ID	VARCHAR(1024)	감사 이벤트시 사용 중인 실행 ID
원래 노드 번호	SMALLINT	감사 이벤트가 발생한 노드 번호
코디네이터 노드 번호	SMALLINT	코디네이터 노드의 노드 번호
응용프로그램 ID	VARCHAR(255)	감사 이벤트 발생시 사용 중인 응용프로그램 ID
응용프로그램 이름	VARCHAR(1024)	감사 이벤트 발생시 사용 중인 응용프로그램 이름
인증 유형	VARCHAR(32)	감사 이벤트시 인증 유형
패키지 스키마	VARCHAR(128)	감사 이벤트시 사용 중인 패키지의 스키마
패키지 이름	VARCHAR(128)	감사 이벤트 발생시 사용 중인 패키지 이름
패키지 섹션 번호	SMALLINT	감사 이벤트 발생시 사용 중인 패키지의 섹션 번호
패키지 버전	VARCHAR (64)	감사 이벤트 발생시 사용 중인 패키지 버전.
플러그인 이름	VARCHAR(*32*)	감사 이벤트 발생 시 사용 중인 플러그인 이름.

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

CONTEXT 이벤트에 대한 감사 레코드 레이아웃

표 15. CONTEXT 이벤트에 대한 감사 레코드 레이아웃

timestamp=1998-06-24-08.42.41.476840;**category=CONTEXT;audit event=EXECUTE_IMMEDIATE;**
event correlator=3;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;
package section=203;**text=create table audit(c1 char(10), c2 integer);**		

이름	**형식**	**설명**
시간소인	CHAR(26)	감사 이벤트의 날짜 및 시간
범주	CHAR(8)	감사 이벤트의 범주. 가능한 값은 다음과 같습니다. CONTEXT
감사 이벤트	VARCHAR(32)	특정 감사 이벤트 가능한 값은 이 표 다음에 오는 목록에 표시됩니다.
이벤트 상관자	INTEGER	감사되는 조작에 대한 상관 ID. 단일 이벤트와 연관된 감사 레코드를 식별하기 위해 사용될 수 있습니다.
데이터베이스 이름	CHAR(8)	이벤트가 생성된 데이터베이스의 이름. 이것이 인스턴스 레벨 감사 이벤트인 경우 공백입니다.
사용자 ID	VARCHAR(1024)	감사 이벤트시의 사용자 ID
권한 부여 ID	VARCHAR(128)	감사 이벤트시의 권한 부여 ID
원래 노드 번호	SMALLINT	감사 이벤트가 발생한 노드 번호
코디네이터 노드 번호	SMALLINT	코디네이터 노드의 노드 번호
응용프로그램 ID	VARCHAR(255)	감사 이벤트 발생시 사용 중인 응용프로그램 ID
응용프로그램 이름	VARCHAR(1024)	감사 이벤트 발생시 사용 중인 응용프로그램 이름
패키지 스키마	VARCHAR(128)	감사 이벤트시 사용 중인 패키지의 스키마
패키지 이름	VARCHAR(128)	감사 이벤트 발생시 사용 중인 패키지 이름
패키지 섹션 번호	SMALLINT	감사 이벤트 발생시 사용 중인 패키지의 섹션 번호
명령문 텍스트(명령문)	CLOB(2M)	적용 가능한 경우 SQL문의 텍스트. SQL문이 사용 가능하지 않은 경우 널(NULL)입니다.
패키지 버전	VARCHAR (64)	감사 이벤트 발생시 사용 중인 패키지 버전.

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

관련 참조:

- 334 페이지의 『가능한 CONTEXT 감사 이벤트 목록』

가능한 CONTEXT 감사 이벤트 목록

다음은 가능한 CONTEXT 감사 이벤트의 목록입니다.

표 16. CONTEXT 감사 이벤트

CONNECT	SET_APPL_PRIORITY
CONNECT_RESET	RESET_DB_CFG
ATTACH	GET_DB_CFG
DETACH	GET_DFLT_CFG
DARI_START	UPDATE_DBM_CFG
DARI_STOP	SET_MONITOR
BACKUP_DB	GET_SNAPSHOT
RESTORE_DB	ESTIMATE_SNAPSHOT_SIZE
ROLLFORWARD_DB	RESET_MONITOR
OPEN_TABLESPACE_QUERY	OPEN_HISTORY_FILE
FETCH_TABLESPACE	CLOSE_HISTORY_FILE
CLOSE_TABLESPACE_QUERY	FETCH_HISTORY_FILE
OPEN_CONTAINER_QUERY	SET_RUNTIME_DEGREE
CLOSE_CONTAINER_QUERY	UPDATE_AUDIT
FETCH_CONTAINER_QUERY	DBM_CFG_OPERATION
SET_TABLESPACE_CONTAINERS	DISCOVER
GET_TABLESPACE_STATISTIC	OPEN_CURSOR
READ_ASYNC_LOG_RECORD	CLOSE_CURSOR
QUIESCE_TABLESPACE	FETCH_CURSOR
LOAD_TABLE	EXECUTE
UNLOAD_TABLE	EXECUTE_IMMEDIATE
UPDATE_RECOVERY_HISTORY	PREPARE
PRUNE_RECOVERY_HISTORY	DESCRIBE
SINGLE_TABLESPACE_QUERY	BIND
LOAD_MSG_FILE	REBIND
UNQUIESCE_TABLESPACE	RUNSTATS
ENABLE_MULTIPAGE	REORG
DESCRIBE_DATABASE	REDISTRIBUTE
DROP_DATABASE	COMMIT
CREATE_DATABASE	ROLLBACK
ADD_NODE	REQUEST_ROLLBACK
FORCE_APPLICATION	IMPLICIT_REBIND

관련 참조:

• 334 페이지의 『CONTEXT 이벤트에 대한 감사 레코드 레이아웃』

감사 기능 추가 정보 및 기술

CHECKING 이벤트로 작업할 때 대부분의 경우, 감사 레코드의 오브젝트 유형 필드는 필수 특권 또는 권한이 오브젝트에 액세스하려는 사용자 ID에 의해 보유되는지 여부를 알아보기 위해 검사되는 오브젝트입니다. 예를 들어, 사용자가 컬럼을 추가하여 테이블을 변경하려고 하는 경우, CHECKING 이벤트 감사 레코드는 시도된 액세스가 『ALTER』이고 검사 중인 오브젝트 유형이 『TABLE』이었음을 나타냅니다(주: 검사되어야 하는 것은 테이블 특권이므로 컬럼이 아닙니다).

그러나 사용자 ID가 오브젝트를 작성 또는 바인드하거나, 오브젝트를 삭제할 수 있게 하는 데이터베이스 권한이 있는지 여부를 확인하는 것이 검사에 포함되면, 데이터베이스에 대한 검사가 있다고 하더라도, 오브젝트 유형 필드는 (데이터베이스 자체보다는) 작성, 바인드 또는 삭제되는 오브젝트를 지정합니다.

테이블에서 인덱스를 작성할 때, 인덱스를 작성하는 특권이 필요하므로, CHECKING 이벤트 감사 레코드는 『작성』보다는 『인덱스』의 액세스 시도 유형을 갖습니다.

기존의 패키지를 바인딩할 때, OBJMAINT 이벤트 감사 레코드가 패키지의 DROP용으로 작성된 다음, 또다른 OBJMAINT 이벤트 감사 레코드는 패키지의 새로운 사본의 CREATE용으로 작성됩니다.

SQL DDL(Data Definition Language)은 성공적인 것으로 로그되는 OBJMAINT 또는 SECMAINT 이벤트를 생성할 수 있습니다. 그러나 이벤트 로깅 이후에 뒤따르는 오류가 ROLLBACK을 유발할 수 있습니다. 이것은 오브젝트를 작성하지 않은 채로 두거나, GRANT 또는 REVOKE 조치를 완료하지 않은 채로 둡니다. 이 경우, CONTEXT 이벤트의 사용이 중요합니다. 이러한 CONTEXT 이벤트 감사 레코드, 특히 이벤트를 종료하는 명령문은 시도된 조작의 완료 특성을 나타냅니다.

DB2® Universal Database (DB2 UDB) 관계 테이블로 로드하는데 적합한 컬럼 식별자가 있는 ASCII 형식의 감사 레코드를 추출할 때, 명령문 텍스트 필드에서 사용되는 분리문자에 대해 명확히 알고 있어야 합니다. 이것은 컬럼 식별자가 있는 ASCII 파일을 추출할 때 완료될 수 있으며, 다음을 사용하여 완료됩니다.

```
db2audit extract delasc delimiter <load delimiter>
```

*load delimiter*는 단일 문자(예: ")이거나 16진수(예: 『0xff』)를 표현하는 4바이트 문자열입니다. 유효한 명령의 예는 다음과 같습니다.

```
db2audit extract delasc
db2audit extract delasc delimiter !
db2audit extract delasc delimiter 0xff
```

추출할 때 분리문자로서 디폴트 로드 분리문자(『0xff』) 이외의 문자를 사용한 경우, LOAD 명령에 MODIFIED BY 옵션을 사용하십시오. 분리문자로서 『0xff』가 사용된 LOAD 명령의 부분 예는 다음과 같습니다.

```
db2 load from context.del of del modified by chardel0xff replace into ...
```

이것은 디폴트 로드 문자열 분리문자인 『0xff』를 겹쳐씁니다.

관련 개념:

• 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

관련 참조:

• 303 페이지의 『감사 기능 사용』

DB2 UDB 감사 기능 활동 제어

프로시저:

감사 기능 활동 제어에 대한 설명의 일부로서, 다음과 같은 간단한 시나리오를 사용합니다. *newton* 사용자는 테이블을 연결하고 작성하는 *testapp* 응용프로그램을 실행합니다. 이와 동일한 응용프로그램이 아래 설명하는 각 예에서 사용됩니다.

극단적인 예를 표시하며 시작합니다. 모든 성공 및 실패 감사 이벤트를 감사하도록 판별했으므로, 다음과 같은 방법으로 감사 기능을 구성합니다.

```
db2audit configure scope all status both
```

주: 이것은 가능한 모든 감사 가능 이벤트에 대한 감사 레코드를 작성합니다. 결과적으로, 많은 레코드가 감사 로그에 작성되며, 이것은 데이터베이스 관리 프로그램의 성능을 저하시킵니다. 이러한 극단적인 경우는 시범 용도로만 표시되며, 위에서 표시한 명령이 있는 감사 기능을 구성하도록 권장하지 않습니다.

이 구성(『db2audit start』 사용)을 사용하여 감사 기능을 시작한 다음 *testapp* 응용프로그램을 실행하면, 다음 레코드가 생성되어 감사 로그에 위치합니다. 로그에서 감사 레코드를 추출하여, 응용프로그램이 수행한 두 가지 조치에 대해 생성된 다음과 같은 레코드를 볼 수 있습니다.

조치 작성된 레코드의 유형

CONNECT

```
timestamp=1998-06-24-08.42.10.555345;category=CONTEXT;
audit event=CONNECT;event correlator=2;database=FOO;
application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.10.944374;category=VALIDATE;
audit event=AUTHENTICATION;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;execution id=newton;
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;

timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
```

```
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
object name=FOO;object type=DATABASE;access approval reason=DATABASE;
access attempted=CONNECT;

timestamp=1998-06-24-08.42.11.801554;category=CONTEXT;
audit event=COMMIT;event correlator=2;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;
```

CREATE TABLE

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;event correlator=3;database=FOO;
userid=boss;authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
package section=203;text=create table audit(c1 char(10), c2 integer);

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;

timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;
audit event=CREATE_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;

timestamp=1998-06-24-08.42.42.018900;category=CONTEXT;
audit event=COMMIT;event correlator=3;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
```

이와 같이, 가능한 모든 감사 이벤트 및 유형의 감사를 요청하는 감사 구성에서 생성된 중요한 몇 개의 감사 레코드가 있습니다.

대부분의 경우, 감사하려는 이벤트에 대해 더 제한되거나 강조되는 뷰의 감사 기능을 구성합니다. 예를 들어, 실패한 이벤트만을 감사하려면, 감사 기능은 다음과 같이 구성될 수 있습니다.

```
db2audit configure scope audit,checking,objmaint,secmaint,sysadmin,
   validate status failure
```

주: 이 구성은 초기 감사 구성이거나 감사 구성이 재설정될 때 발생하는 구성입니다.

이 구성을 사용하여 감사 기능을 시작한 다음 *testapp* 응용프로그램을 실행하면, 다음 레코드가 생성되어 감사 로그에 위치합니다(그리고 이전에 *testapp*를 실행하지 않은 것으로 가정합니다). 로그에서 감사 레코드를 추출하여, 응용프로그램이 수행한 두 가지 조치에 대해 생성된 다음과 같은 레코드를 볼 수 있습니다.

조치 작성된 레코드의 유형

CONNECT

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;
```

CREATE TABLE

```
(none)
```

이벤트 시도가 실패한 경우에만 가능한 모든 감사 이벤트(CONTEXT 제외)의 감사를 요청하는 감사 구성에서 생성된 훨씬 더 적은 감사 레코드가 있습니다. 사용자는 감사 구성을 변경하여 생성된 감사 레코드의 유형 및 특성을 제어할 수 있습니다.

감사 기능은 감사하려는 감사 레코드가 오브젝트에 대해 성공적으로 특권이 권한 부여될 때 감사 레코드를 작성할 수 있게 합니다. 이 경우, 다음과 같이 감사 기능을 구성할 수 있습니다.

```
db2audit configure scope checking status success
```

이 구성을 사용하여 감사 기능을 시작한 다음 *testapp* 응용프로그램을 실행하면, 다음 레코드가 생성되어 감사 로그에 위치합니다(그리고 이전에 *testapp*를 실행하지 않은 것으로 가정합니다). 로그에서 감사 레코드를 추출하여, 응용프로그램이 수행한 두 가지 조치에 대해 생성된 다음과 같은 레코드를 볼 수 있습니다.

조치 작성된 레코드의 유형

CONNECT

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;
```

CREATE TABLE

```
(none)
```

관련 개념:

- 317 페이지의 『감사 기능 레코드 레이아웃(소개)』

관련 참조:

- 303 페이지의 『감사 기능 사용』

제 3 부 부록

부록 A. 이름 지정 규칙 준수

일반 이름 지정 규칙

모든 오브젝트 및 사용자의 이름 지정에 대하여 규칙이 존재합니다. 일부 규칙은 작업 중인 플랫폼에 고유한 것입니다. 예를 들어, 이름에 대소문자를 사용하는 것과 관련된 규칙이 있습니다.

- UNIX® 플랫폼에서는 소문자로 이름을 지정해야 합니다.
- Windows® 플랫폼에서는 대문자, 소문자 및 대소문자 혼용으로 이름을 지정할 수 있습니다.

다르게 지정하지 않을 경우, 모든 이름은 다음과 같은 문자를 포함할 수 있습니다.

- A - Z. 대부분의 이름에서 사용될 경우, 문자 A - Z가 소문자에서 대문자로 변환됩니다.
- 0 - 9.
- ! % ( ) { } . - ^ ~ _(밑줄) @, #, $ 및 스페이스.
- \(백슬래시).

이름은 숫자나 밑줄 문자로 시작할 수 없습니다.

테이블, 뷰, 컬럼, 인덱스 또는 권한 부여 ID를 이름 지정할 때 SQL 예약어를 사용하지 마십시오.

운영 체제 및 DB2® Universal Database(DB2 UDB)에 대해 작업 중인 위치에 따라 서로 다르게 작동할 수도 있는 그밖의 특수 문자가 있습니다. 그러나 작동할 수도 있기는 하지만, 작동한다는 보장은 없습니다. 데이터베이스에서 오브젝트 이름을 지정할 때 이들 기타 특수 문자를 사용하지 않도록 권장합니다.

또한 오브젝트 이름 지정 규칙, 워크스테이션 이름 지정 규칙, NLS 환경에서의 이름 지정 규칙 및 Unicode 환경에서의 이름 지정 규칙을 고려해야 합니다.

관련 개념:

- 344 페이지의『DB2 UDB 오브젝트 이름 지정 규칙』
- 349 페이지의『워크스테이션 이름 지정 규칙』
- 347 페이지의『사용자, 사용자 ID 및 그룹 이름 지정 규칙』
- 348 페이지의『페더레이티드 데이터베이스 오브젝트 이름 지정 규칙』

DB2 UDB 오브젝트 이름 지정 규칙

모든 오브젝트는 일반 이름 지정 규칙을 따릅니다. 추가로 일부 오브젝트는 추가 제한 사항이 동반된 테이블에 표시됩니다.

표 17. 데이터베이스, 데이터베이스 별명 및 인스턴스 이름 지정 규칙

오브젝트	지침
• 데이터베이스 • 데이터베이스 별명 • 인스턴스	• 데이터베이스 이름은 카탈로그된 위치 내에서 고유해야 합니다. DB2® Universal Database(DB2 UDB)의 UNIX® 기반 구현에서는 이 위치가 디렉토리 경로인 반면, Windows® 구현에서는 논리적 디스크입니다. • 데이터베이스 별명 이름은 시스템 데이터베이스 디렉토리 내에서 고유해야 합니다. 새 데이터베이스가 작성되면 별명의 디폴트값은 데이터베이스 이름입니다. 결과적으로 데이터베이스 별명으로 존재하는 이름의 데이터베이스가 없는 경우에도 이 이름을 사용하는 데이터베이스를 작성할 수 없습니다. • 데이터베이스, 데이터베이스 별명 및 인스턴스 이름은 8바이트까지 지정할 수 있습니다. • Windows NT®, Windows 2000, Windows XP 및 Windows Server 2003 시스템에서 인스턴스에는 서비스 이름과 동일한 이름을 사용할 수 없습니다. **주:** 잠재적인 문제점을 피하려면, 통신 환경에서 데이터베이스를 사용하려는 경우 데이터베이스 이름에 특수 문자 @, # 및 $를 사용하지 마십시오. 또한 이러한 문자는 모든 키보드에 공통되지 않으므로, 데이터베이스를 다른 언어로 사용할 계획인 경우에는 이런 문자를 사용하지 마십시오.

표 18. 데이터베이스 오브젝트 이름 지정 규칙

오브젝트	지침
• 별명 • 버퍼 풀 • 컬럼 • 이벤트 모니터 • 인덱스 • 메소드 • 노드 그룹 • 패키지 • 패키지 버전 • 스키마 • 스토어드 프로시저 • 테이블 • 테이블 스페이스 • 트리거 • UDF • UDT • 뷰	다음 경우를 *제외*하고는 18바이트까지 사용할 수 있습니다. • 최대 128바이트를 사용할 수 있는 테이블 이름(뷰 이름, 요약 테이블 이름, 별명 및 상관 이름). • 최대 30바이트까지 사용할 수 있는 컬럼 이름 • 최대 8바이트까지 사용할 수 있는 패키지 이름 • 최대 30바이트까지 사용할 수 있는 스키마 이름 • 최대 64바이트까지 사용할 수 있는 패키지 버전 • 오브젝트 이름에는 다음 문자도 사용할 수 있습니다. – 유효한 강조 문자(예: ö) – 멀티바이트 스페이스를 제외한 멀티바이트 문자(멀티바이트 환경의 경우) • 패키지 이름과 패키지 버전에는 마침표(.), 하이픈(-) 및 콜론(:)도 사용할 수 있습니다.

표 19. 페더레이티드 데이터베이스 오브젝트 이름 지정 규칙

오브젝트	지침
• 함수 맵핑 • 인덱스 스펙 • 별칭 • 서버 • 유형 맵핑 • 사용자 맵핑 • 랩퍼	• 별칭, 맵핑, 인덱스 스펙, 서버 및 랩퍼 이름은 128바이트를 초과할 수 없습니다. • 서버와 별칭 옵션 및 옵션 설정은 255바이트로 제한됩니다. • 페더레이티드 데이터베이스 오브젝트의 이름은 다음 문자를 또한 포함할 수 있습니다. – 유효한 악센트 문자(예: ö) – 멀티바이트 스페이스를 제외한 멀티바이트 문자(멀티바이트 환경의 경우)

분리 ID 및 오브젝트 이름:

키워드를 사용할 수 있습니다. SQL 키워드로 해석할 수도 있는 컨텍스트에서 키워드를 사용할 경우, 분리 ID로 지정해야 합니다.

분리 ID를 사용하면 이름 지정 규칙을 위반하는 오브젝트를 작성할 수 있지만, 나중에 오브젝트를 사용할 때 오류가 발생할 수 있습니다. 예를 들어, 이름에 + 또는 - 부호가 포함된 컬럼을 작성한 후 나중에 인덱스에서 해당 컬럼을 사용할 경우, 테이블 재구성을 시도할 때 문제점이 발생합니다.

추가 스키마 이름 정보:

- 사용자 정의 유형(UDT)은 8바이트를 초과하는 스키마 이름을 사용할 수 없습니다.
- 다음 스키마 이름은 예약어이며 사용할 수 없습니다: SYSCAT, SYSFUN, SYSIBM, SYSSTAT.
- 나중에 잠재적인 이주 문제점을 방지하려면, SYS로 시작하는 스키마 이름을 사용하지 마십시오. 데이터베이스 관리 프로그램은 SYS로 시작하는 스키마 이름을 사용하여 트리거, 사용자 정의 유형 또는 사용자 정의 함수를 작성하도록 허용하지 않습니다.
- 스키마 이름으로 SESSION을 사용하지 않도록 권장합니다. 선언된 임시 테이블은 SESSION으로 규정되어야 합니다. 따라서 응용프로그램이 영구 테이블의 이름과 동일한 이름의 임시 테이블을 선언할 수 있지만, 이 경우 응용프로그램 논리가 매우 복잡해집니다. 선언된 임시 테이블을 처리할 때를 제외하고, SESSION 스키마를 사용하지 마십시오.

관련 개념:

- 343 페이지의 『일반 이름 지정 규칙』

분리 ID 및 오브젝트 이름

키워드를 사용할 수 있습니다. SQL 키워드로 해석할 수도 있는 컨텍스트에서 키워드를 사용할 경우, 분리 ID로 지정해야 합니다.

분리 ID를 사용하면 이름 지정 규칙을 위반하는 오브젝트를 작성할 수 있지만, 나중에 오브젝트를 사용할 때 오류가 발생할 수 있습니다. 예를 들어, 이름에 + 또는 - 부호가 포함된 컬럼을 작성한 후 나중에 인덱스에서 해당 컬럼을 사용할 경우, 테이블 재구성을 시도할 때 문제점이 발생합니다.

관련 개념:

- 343 페이지의 『일반 이름 지정 규칙』

사용자, 사용자 ID 및 그룹 이름 지정 규칙

표 20. 사용자, 사용자 ID 및 그룹 이름 지정 규칙

오브젝트	지침
• 그룹 이름 • 사용자 이름 • 사용자 ID	• 그룹 이름은 최대 30문자까지 포함할 수 있습니다. • UNIX® 기반 시스템의 사용자 ID는 최대 8자까지 포함할 수 있습니다. • 최대 30문자까지 사용할 수 있는 Windows®의 사용자 이름. Windows NT®, Windows 2000, Windows XP 및 Windows Server 2003에서는 현재 20자까지만 사용할 수 있습니다. • 클라이언트 인증을 사용하지 않은 경우, 사용자 이름이 8자보다 긴 Windows NT, Windows 2000, Windows XP 및 Windows Server 2003에 대한 비Windows 32비트 클라이언트 연결은 사용자 이름과 암호가 명시적으로 지정되는 경우 지원됩니다. • 이름 및 ID는 다음과 같을 수 없습니다. – USERS, ADMINS, GUESTS, PUBLIC, LOCAL 또는 모든 SQL 예약어 – IBM®, SQL 또는 SYS로 시작. – 강조 문자 포함.

주:

1. 일부 운영 체제에서는 대소문자가 구분되는 USER ID 및 암호를 사용할 수 있습니다. 이 경우인지를 알아 보려면 운영 체제 문서를 참조하십시오.
2. 성공적인 CONNECT 또는 ATTACH로부터 리턴된 권한 부여 ID는 8자로 절단됩니다. 생략 부호(...)가 권한 부여 ID에 추가되며 SQLWARN 필드에는 절단을 나타내는 경고가 포함됩니다.
3. 사용자 ID 및 암호의 뒤 공백은 제거됩니다.

관련 개념:

- 343 페이지의『일반 이름 지정 규칙』
- 348 페이지의『페더레이티드 데이터베이스 오브젝트 이름 지정 규칙』

페더레이티드 데이터베이스 오브젝트 이름 지정 규칙

표 21. 페더레이티드 데이터베이스 오브젝트 이름 지정 규칙

오브젝트	지침
• 함수 맵핑 • 인덱스 스펙 • 별칭 • 서버 • 유형 맵핑 • 사용자 맵핑 • 랩퍼	• 별칭, 맵핑, 인덱스 스펙, 서버 및 랩퍼 이름은 128바이트를 초과할 수 없습니다. • 서버와 별칭 옵션 및 옵션 설정은 255바이트로 제한됩니다. • 페더레이티드 데이터베이스 오브젝트의 이름은 다음 문자를 또한 포함할 수 있습니다. – 유효한 악센트 문자(예: ö) – 멀티바이트 스페이스를 제외한 멀티바이트 문자(멀티바이트 환경의 경우)

관련 개념:

• 343 페이지의 『일반 이름 지정 규칙』

스키마 이름 사용에 관한 추가 제한사항 및 권장사항

- 사용자 정의 유형(UDT)은 8바이트를 초과하는 스키마 이름을 사용할 수 없습니다.
- 다음 스키마 이름은 예약어이며 사용할 수 없습니다: SYSCAT, SYSFUN, SYSIBM, SYSSTAT.
- 나중에 잠재적인 이주 문제점을 방지하려면, SYS로 시작하는 스키마 이름을 사용하지 마십시오. 데이터베이스 관리 프로그램은 SYS로 시작하는 스키마 이름을 사용하여 트리거, 사용자 정의 유형 또는 사용자 정의 함수를 작성하도록 허용하지 않습니다.
- 스키마 이름으로 SESSION을 사용하지 않도록 권장합니다. 선언된 임시 테이블은 SESSION으로 규정되어야 합니다. 따라서 응용프로그램이 영구 테이블의 이름과 동일한 이름의 임시 테이블을 선언할 수 있지만, 이 경우 응용프로그램 논리가 매우 복잡해집니다. 선언된 임시 테이블을 처리할 때를 제외하고, SESSION 스키마를 사용하지 마십시오.

관련 개념:

• 343 페이지의 『일반 이름 지정 규칙』

서버에서 암호 유지보수

암호 유지보수 태스크를 수행해야 하는 경우도 있습니다. 이런 태스크는 서버에서 필수이며 이런 태스크가 수행되는 동안 많은 사용자가 서버 환경에서 작업을 못하거나 편하게 작업할 수 없으므로, 이러한 태스크 수행에는 상당한 어려움이 따릅니다. DB2® Universal Database(DB2 UDB)는 서버에 있지 않고도 암호를 갱신하고 검증하는 방

법을 제공합니다. 예를 들어, OS/390®용 DB2 버전 5는 이 사용자 암호 변경 방법을 지원합니다. 오류 메시지 SQL1404N "암호 만기"가 표시되면 다음과 같이 CONNECT 문을 사용하여 암호를 변경하십시오.

```
CONNECT TO <database> USER <userid> USING <password>
   NEW <new_password> CONFIRM <new_password>
```

DB2 UDB 구성 지원 프로그램(CA)의 『암호 변경』 대화 상자를 사용하여 암호를 변경할 수도 있습니다.

관련 개념:

- 343 페이지의 『일반 이름 지정 규칙』
- 344 페이지의 『DB2 UDB 오브젝트 이름 지정 규칙』
- 349 페이지의 『워크스테이션 이름 지정 규칙』
- 347 페이지의 『사용자, 사용자 ID 및 그룹 이름 지정 규칙』
- 348 페이지의 『페더레이티드 데이터베이스 오브젝트 이름 지정 규칙』
- 346 페이지의 『분리 ID 및 오브젝트 이름』
- 348 페이지의 『스키마 이름 사용에 관한 추가 제한사항 및 권장사항』

워크스테이션 이름 지정 규칙

*워크스테이션 이름*은 로컬 워크스테이션에 있는 데이터베이스 서버, 데이터베이스 클라이언트 또는 DB2® Universal Database(DB2 UDB) Personal Edition의 NetBIOS 이름을 지정합니다. 이 이름은 데이터베이스 관리 프로그램 구성 파일에 저장됩니다. 워크스테이션 이름을 *워크스테이션 nname*이라고 합니다.

또한 사용자가 지정하는 이름은 다음과 같은 특성을 갖습니다.

- 1 - 8자를 포함할 수 있음
- &, # 또는 @를 포함할 수 없음
- 네트워크 내에서 고유해야 함

파티션된 데이터베이스 시스템에는 전체 파티션된 데이터베이스 시스템을 나타내는 하나의 워크스테이션 *nname*만이 존재하지만, 각각의 노드에는 자체 유도된 고유한 NetBIOS *nname*이 있습니다.

파티션된 데이터베이스 시스템을 나타내는 워크스테이션 *nname*은 인스턴스를 소유하는 데이터베이스 파티션 서버에 대한 데이터베이스 관리 프로그램 구성 파일에 저장됩니다.

각 노드의 고유한 *nname*은 워크스테이션 *nname* 및 노드 번호의 유도된 조합입니다.

노드에 인스턴스가 없을 경우, NetBIOS *nname*이 다음과 같이 유도됩니다.

1. 인스턴스 소유 머신의 첫 번째 워크스테이션 *nname* 문자가 노드 NetBIOS *nname*의 첫 번째 문자로 사용됩니다.
2. 그 다음 1 - 3자는 노드 번호를 나타냅니다. 범위는 1 - 999입니다.
3. 나머지 문자는 인스턴스 소유 머신의 워크스테이션 *nname*에서 가져옵니다. 나머지 문자의 수는 인스턴스 소유 머신의 워크스테이션 *nname* 길이에 따라 다릅니다. 이 숫자는 0에서 4 사이일 수 있습니다.

예를 들어, 다음과 같습니다.

인스턴스 소유 머신의 *워크스테이션 nname*	노드 번호	유도된 노드 **NetBIOS** *nname*
GEORGE	3	G3ORGE
A	7	A7
B2	94	B942
N0076543	21	N216543
GEORGE5	1	G1RGE5

설치 도중 디폴트 워크스테이션 *nname*을 변경한 경우, 워크스테이션 *nname*의 마지막 4자는 NetBIOS *nname* 충돌 유도를 최소화할 수 있도록 NetBIOS 네트워크에서 고유해야 합니다.

관련 개념:

• 343 페이지의 『일반 이름 지정 규칙』

NLS 환경의 이름 지정 규칙

데이터베이스 이름에서 사용될 수 있는 기본 문자 세트는 1바이트 대소문자 라틴 문자(A...Z, a...z), 아라비아 숫자(0...9) 및 밑줄 문자(_)로 구성됩니다. 호스트 데이터베이스 제품과의 호환성을 제공하기 위해 세 개의 특수 문자(#, @ 및 $)가 이 목록에 추가되었습니다. 특수 문자 #, @ 및 $는 NLS 호스트(EBCDIC) 불변 문자 세트에는 포함되지 않기 때문에 NLS 환경에서는 주의하여 사용하십시오. 사용 중인 코드 페이지에 따라 확장 문자 세트의 문자를 사용할 수도 있습니다. 다중 코드 페이지 환경에서 데이터베이스를 사용할 경우, 모든 코드 페이지가 확장 문자 세트에서 사용하려고 하는 어떠한 영문자라도 지원한다는 것을 확인해야 합니다.

데이터베이스 오브젝트(테이블 및 뷰)를 이름 지정할 때 프로그램 레이블, 호스트 변수, 커서 및 확장 문자 세트의 요소(예를 들어, 구별할 수 있는 표시가 있는 문자)를 사용할 수 있습니다. 정확히 어떤 문자가 사용 가능한지는 사용 중인 코드 페이지에 의해 좌우됩니다.

DBCS ID용 확장 문자 세트 정의:

DBCS 환경에서, 확장 문자 세트는 기본 문자 세트의 모든 문자로 구성되어 있으며, 다음도 추가됩니다.

- 2바이트 스페이스를 제외한 DBCS 코드 페이지의 모든 2바이트 문자가 유효합니다.
- 2바이트 스페이스는 특수 문자입니다.
- 혼합 코드 페이지에서 사용 가능한 1바이트 문자. 다음과 같이 다양한 범주에 할당됩니다.

범주	복합 코드 페이지 내의 유효한 코드 포인트
숫자	x30-39
문자	x23-24, x40-5A, x61-7A, xA6-DF(코드 페이지 932 및 942의 경우 A6-DF)
특수 문자	다른 모든 유효한 1바이트 문자 코드 포인트

관련 개념:

- 343 페이지의 『일반 이름 지정 규칙』
- 344 페이지의 『DB2 UDB 오브젝트 이름 지정 규칙』
- 349 페이지의 『워크스테이션 이름 지정 규칙』

유니코드 환경의 이름 지정 규칙

UCS-2 데이터베이스에서 모든 ID는 멀티 바이트 UTF-8입니다. 따라서, DB2® Universal Database(DB2 UDB)가 확장 문자 세트(예를 들어, 강조 문자 또는 멀티 바이트 문자)에 있는 문자 사용을 허용한 ID에 모든 UCS-2 문자를 사용할 수 있습니다.

클라이언트는 환경이 지원하는 모든 문자를 입력할 수 있으며, ID에 있는 모든 문자는 데이터베이스 관리 프로그램에 의해 UTF-8로 변환됩니다. UCS-2 데이터베이스에 대한 ID에 자국어 문자를 지정할 때 두 가지 점을 고려해야 합니다.

- 각 비ASCII 문자는 2바이트 또는 4바이트를 요구합니다. 따라서, *n*바이트 ID는 비ASCII 문자에 대한 ASCII 문자 비율에 따라 *n*/4과 *n* 문자 사이에서만 보유할 수 있습니다. 한 개 또는 두 개의 비ASCII(예: 강조 문자)만 가지고 있을 경우, 한계는 *n* 문자에 더 가깝습니다. 반면에 완전히 비ASCII(예: 일본어)인 ID의 경우, *n*/4 - *n*/3 문자만 사용될 수 있습니다.
- ID가 다른 클라이언트 환경에서 입력되는 경우, ID는 이 클라이언트에 사용할 수 있는 문자의 공통 서브세트를 사용하여 정의되어야 합니다. 예를 들어, UCS-2 데이터베이스가 라틴-1, 아랍어 및 일본어 환경에서 액세스되는 경우, 모든 ID는 현실적으로 ASCII로 제한되어야 합니다.

관련 개념:

- 343 페이지의 『일반 이름 지정 규칙』
- 344 페이지의 『DB2 UDB 오브젝트 이름 지정 규칙』

- 349 페이지의『워크스테이션 이름 지정 규칙』

부록 B. 자동 클라이언트 재라우트 사용

이 부록은 자동 클라이언트 재라우트 작동 방법 및 사용자 환경에서 제대로 작동하게 하는 방법을 설명합니다.

자동 클라이언트 재라우트 설명 및 설정

서버가 손상될 때마다, 해당 서버에 연결된 각각의 클라이언트는 연결이 종료되고 결과적으로 응용프로그램 오류를 야기하는 통신 오류를 수신합니다. 사용 가능성이 중요한 경우, 대기 노드로 서버를 페일오버하기 위한 별도의 설정 또는 기능을 구현해야 합니다. 두 가지 모두, DB2® Universal Database(DB2 UDB) 클라이언트 코드는 페일오버 노드(IP 주소 또한 페일오버됨)에서 실행 중일 수도 있는 원래 서버로 또는 새로운 서버로 연결을 재설정하려고 시도합니다.

연결이 재설정되면, 응용프로그램은 트랜잭션 실패를 알리는 오류 메시지를 수신하지만 응용프로그램은 다음 트랜잭션을 계속할 수 있습니다.

자동 클라이언트 재라우트 기능의 주요 목표는 DB2 UDB 클라이언트 응용프로그램이 통신 손실에서 복구하여 최소한의 인터럽트로 작업을 계속할 수 있게 하는 것입니다. 이름이 적용될 경우, 재라우트가 연속 조작 지원의 핵심이 됩니다. 그러나 재라우트는 클라이언트 연결에 식별되는 대체 위치가 있을 때에만 가능합니다.

자동 클라이언트 재라우트 기능은 다음과 같은 구성 가능 환경 내에서 사용할 수 있습니다.

1. 데이터 파티션 기능(DPF)이 있는 Enterprise Server Edition(ESE)
2. Data Propagator(DPROPR) 스타일 복제
3. 고가용성 클러스터 멀티프로세서(HACMP)
4. 고가용성 재해 복구(HADR).

   자동 클라이언트는 HADR과 결합하여 작동하면서 액세스 중인 데이터베이스의 페일오버 후에 클라이언트 응용프로그램이 최소한의 인터럽트로 작업을 계속할 수 있게 해줍니다.

DB2 UDB가 통신 손실로부터 복구하기 위해서는, 통신 손실이 발생하기 전에 대체 서버 위치를 지정해야 합니다. UPDATE ALTERNATE SERVER FOR DATABASE 명령을 사용하여 대체 서버를 지정할 수 있습니다. 지정된 대체 서버 위치를 모든 클라이언트에 적용하려면, 대체 서버 위치를 서버측에서 지정해야 합니다. 클라이언트 인스턴스에서 설정된 경우 대체 서버가 무시됩니다.

예를 들어, 데이터베이스가 『N1』이라는 노드에 있다고 가정하십시오(호스트 이름 XXX 및 포트 번호 YYY). 데이터베이스 관리자가 대체 서버 위치를 포트 번호 123의 호스트 이름 = AAA로 설정하고자 합니다. 다음은 데이터베이스 관리자가 노드 N1(서버 인스턴스의)에서 실행하는 명령입니다.

```
db2 update alternate server for database db2 using hostname AAA port 123
```

데이터베이스 관리자가 서버 인스턴스의 특정 데이터베이스에 대체 서버 위치를 지정한 후, 대체 서버 위치는 연결 시 클라이언트로 되돌려집니다. 어떠한 이유로 통신이 손실된 경우, DB2 UDB 클라이언트 코드가 서버에서 리턴된 대체 서버 정보를 사용하여 연결을 재설정할 수 있습니다. 대체 서버가 서버에 지정되지 않은 경우, 자동 클라이언트 재라우트가 처리되지 않습니다.

일반적으로 대체 서버를 지정할 경우, 통신 오류(sqlcode -30081) 또는 sqlcode -1224 감지 시 자동 클라이언트 재라우트가 사용 가능하게 됩니다. 그러나 고가용성 재해 복구(HADR) 환경에서는 sqlcode -1776이 HADR 대기 서버로부터 리턴될 경우 또한 사용 가능하게 됩니다.

관련 개념:

- 354 페이지의 『자동 클라이언트 재라우트 한계』

관련 참조:

- 356 페이지의 『자동 클라이언트 재라우트 예』

자동 클라이언트 재라우트 한계

자동 클라이언트 재라우트 기능을 사용할 때 다음과 같은 몇 가지 한계가 있습니다.

- 자동 클라이언트 재라우트는 DB2® Universal Database(DB2 UDB) 서버 또는 DB2 Connect™ 서버에 연결할 때 사용되는 통신 프로토콜이 TCP/IP인 경우에만 지원됩니다. 이는 연결이 TCP/IP 이외의 다른 프로토콜을 사용 중인 경우, 자동 클라이언트 재라우트 기능이 사용 가능하지 않음을 의미합니다. DB2 UDB가 루프백에 대해 설정된 경우에도, 자동 클라이언트 재라우트 기능을 채택하려면 TCP/IP 통신 프로토콜을 사용해야 합니다.
- 연결이 대체 서버 위치로 재설정된 경우, 동일한 데이터베이스 별칭에 대한 모든 새로운 연결이 대체 서버 위치로 연결됩니다. 원래 위치에 관한 문제점이 수정된 경우 새로운 연결을 원래 위치로 설정하고자 할 때, 다음과 같은 몇 가지 옵션을 선택할 수 있습니다.
  - 대체 서버를 오프라인화해야 하며 연결을 원래 서버로 다시 페일오버할 수 있도록 합니다. (이 옵션은 원래 서버가 대체 서버에 대한 대체 위치로 설정될 수 있도록 UPDATE ALTERNATE SERVER 명령을 사용하여 카탈로그화되었다고 가정합니다.)

  - 새 연결에서 사용할 새 데이터베이스 별명을 카탈로그화할 수 있습니다.
  - 데이터베이스 항목을 카탈로그 해제한 후 다시 카탈로그화할 수 있습니다.
- Linux, UNIX® 및 Windows® 운영 체제용 DB2 UDB는 클라이언트 및 서버 양측에서 자동 클라이언트 재라우트 기능을 지원합니다. 기타 DB2 UDB 제품군은 현재 이 기능을 지원하지 않습니다.
- z/OS™용 DB2 UDB에서 데이터 공유 Sysplex를 설정하면 연결에 사용 가능한 서버 목록이 리턴될 수도 있습니다. 그러나 통신 실패 발생 시 목록은 메모리에만 유지됩니다. 이 때, DB2 UDB 클라이언트는 목록을 사용하여 연결할 적절한 대체 서버의 위치를 판별합니다.
- 대체 호스트 서버에 설치된 DB2 UDB 서버는 원래 호스트 서버에 설치된 DB2 UDB와 비교할 때 동일한 버전이어야 합니다(보다 높은 FixPak일 수 있음).
- 클라이언트 머신의 데이터베이스 디렉토리를 갱신할 수 있는지 여부에 관계없이, 대체 서버 정보는 항상 메모리에 유지됩니다. 다시 말해서 데이터베이스 디렉토리를 갱신할 권한이 없을 경우(또는 읽기 전용 데이터베이스 디렉토리일 경우), 메모리가 응용프로그램 간에 공유되지 않으므로 다른 응용프로그램이 대체 서버를 판별하고 사용할 수 없습니다.
- 동일한 인증이 모든 대체 위치에 적용됩니다. 이는 대체 위치가 원래 위치와 다른 인증 유형을 가질 경우 클라이언트가 데이터베이스 연결을 재설정할 수 없음을 의미합니다.
- 통신 실패가 발생하면, 전역 임시 테이블, ID, 시퀀스, 커서, 페더레이티드 처리용 서버 옵션(SET SERVER OPTION) 및 특수 레지스터와 같은 모든 세션 자원이 손실됩니다. 응용프로그램은 작업 처리를 계속하기 위해 세션 자원을 재설정해야 합니다. DB2 UDB는 통신 오류 이전 발행된 특수 레지스터 명령문을 재실행하므로, 연결 재설정 후 특수 레지스터 명령문을 실행할 필요가 없습니다. 그러나 일부 특수 레지스터는 재실행되지 않습니다. 태스크는 다음과 같습니다.
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE
  - SET SESSION AUTHORIZATION
  - SET TRANSFORM GROUP

    주: 클라이언트가 CLI, JCC 유형 2 또는 유형 4 드라이버를 사용 중인 경우, 연결이 재설정된 후 원래 서버에 대해 준비된 SQL문에 대하여 새 서버를 사용하여 내재적으로 다시 준비됩니다. 그러나 Embedded SQL 루틴(예를 들어, SQC 또는 SQX 응용프로그램)의 경우 다시 준비되지 않습니다.

자동 클라이언트 재라우트를 수행하는 대신 DNS 항목을 사용하여 DNS 항목에 대한 대체 IP 주소를 지정할 수 있습니다. 의도는 2차 IP 주소(대체 서버 위치)를 DNS 항

목에 지정하는 것이며, 클라이언트가 대체 서버에 관하여 알지 못하지만 연결 시 DB2 UDB가 DNS 항목에 대한 IP 주소를 대체합니다.

관련 개념:

- 86 페이지의『자동 클라이언트 재라우트 구현』
- 353 페이지의『자동 클라이언트 재라우트 설명 및 설정』

태스크 관련:

- 82 페이지의『데이터베이스의 대체 서버 식별』

관련 참조:

- 356 페이지의『자동 클라이언트 재라우트 예』

자동 클라이언트 재라우트 예

다음은 클라이언트 응용프로그램에 대한 자동 클라이언트 재라우트의 예입니다(의사(pseudo) 코드만으로 표시됨).

```
        int checkpoint = 0;

        check_sqlca(unsigned char *str, struct sqlca *sqlca)
        {
           if (sqlca->sqlcode == -30081)
           {
              // as communication is lost, terminate the application right away
              exit(1);
           }
  else
           {
              // print out the error
              printf(...);

         if (sqlca->sqlcode == -30108)
         {
            // connection is re-established, re-execute the failed transaction
                 if (checkpoint == 0)
                 {
                     goto checkpt0;
                 }
            else if (checkpoint == 1)
                 {
                    goto checkpt1;
                 }
                 else if (checkpoint == 2)
                 {
                     goto checkpt2;
                 }
                 ....
                 exit;
         }
           }
        }

main( )
        {
```

```
   connect to mydb;
   check_sqlca("connect failed", &sqlca);

checkpt0:
   EXEC SQL set current schema XXX;
   check_sqlca("set current schema XXX failed", &sqlca);

   EXEC SQL create table t1...;
   check_sqlca("create table t1 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 1;
   }

checkpt1:
   EXEC SQL set current schema YYY;
   check_sqlca("set current schema YYY failed", &sqlca);

   EXEC SQL create table t2...;
   check_sqlca("create table t2 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 2;
   }
...
}
```

클라이언트 머신에서 『hornet』 노드를 참조하는 『mydb』라는 데이터베이스가 카탈로그화됩니다. 『hornet』 또한 노드 디렉토리에서 카탈로그화됩니다(포트 번호 456의 호스트 이름 『hornet』).

예 1(비-HADR 데이터베이스 포함)

『hornet』 서버에서(호스트 이름은 포트 번호를 사용하는 hornet과 같음), 『mydb』 데이터베이스가 작성됩니다. 추가로 『mydb』 데이터베이스가 또한 대체 서버에서 작성됩니다(포트 번호 456의 호스트 이름 『montero』). 또한 다음과 같이 『hornet』 서버의 『mydb』 데이터베이스에 대하여 대체 서버를 갱신해야 합니다.

```
db2 update alternate server for database mydb using hostname montero port 456
```

위의 샘플 응용프로그램에서 자동 클라이언트 재라우트 기능이 설정되지 않은 경우 `create table t1`문에서 통신 오류가 발생하면, 응용프로그램이 종료됩니다. 자동 클라이언트 재라우트 기능이 설정된 경우, DB2 UDB는 『hornet』 호스트(포트 456)에 대한 연결을 다시 설정하려고 시도합니다. 계속 실패할 경우, DB2 UDB는 대체 서버 위치(포트 456의 『montero』 호스트)를 시도합니다. 대체 서버 위치에 대한 연결에서 통

신 오류가 없다고 가정할 때, 응용프로그램은 계속해서 후속 명령문을 실행할 수 있습니다(또한 실패한 트랜잭션을 재실행할 수 있음).

예 2(HADR 데이터베이스 포함)

『hornet』 서버에서(호스트 이름은 포트 번호를 사용하는 hornet과 같음), 1차 데이터베이스 『mydb』가 작성됩니다. 2차 데이터베이스가 포트 456의 『montero』 호스트에서 또한 작성됩니다. 1차 및 2차 두 데이터베이스 모두에 대하여 HADR을 설정하는 방법에 관한 정보가 *데이터 복구 및 고가용성 안내 및 참조서*에 있습니다. 또한 다음과 같이 『mydb』 데이터베이스에 대하여 대체 서버를 갱신해야 합니다.

```
db2 update alternate server for database mydb using hostname montero port 456
```

위의 샘플 응용프로그램에서 자동 클라이언트 재라우트 기능이 설정되지 않은 경우 `create table t1`문에서 통신 오류가 발생하면, 응용프로그램이 종료됩니다. 자동 클라이언트 재라우트 기능이 설정된 경우, DB2 UDB는 『hornet』 호스트(포트 456)에 대한 연결을 다시 설정하려고 시도합니다. 계속 실패할 경우, DB2 UDB는 대체 서버 위치(포트 456의 『montero』 호스트)를 시도합니다. 대체 서버 위치에 대한 연결에서 통신 오류가 없다고 가정할 때, 응용프로그램은 계속해서 후속 명령문을 실행할 수 있습니다(또한 실패한 트랜잭션을 재실행할 수 있음).

관련 개념:

- 353 페이지의 『자동 클라이언트 재라우트 설명 및 설정』

태스크 관련:

- 82 페이지의 『데이터베이스의 대체 서버 식별』

부록 C. LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스 사용

LDAP(Lightweight Directory Access Protocol) 입문

LDAP(Lightweight Directory Access Protocol)는 디렉토리 서비스에 대한 산업 표준 액세스 방법입니다. 디렉토리 서비스는 분산 환경 내의 다중 시스템과 서비스에 대한 자원 정보의 저장소이며, 이들 자원에 대한 클라이언트 및 서버 액세스를 제공합니다. 각 데이터베이스 서버 인스턴스는 LDAP 서버로 그 존재를 발행하며 데이터베이스가 작성될 때 LDAP 디렉토리로 데이터베이스 정보를 제공합니다. 클라이언트가 데이터베이스에 연결할 때, 카탈로그에 대한 정보는 LDAP 디렉토리에서 검색될 수 있습니다. 각 클라이언트는 각 머신에 로컬로 카탈로그 정보를 저장하는 것이 더 이상 필요하지 않습니다. 클라이언트 응용프로그램은 데이터베이스에 연결하는 데 필요한 정보 요구를 찾기 위해 LDAP 디렉토리를 검색합니다.

클라이언트가 LDAP Directory Server를 한 번만 검색하도록 캐싱 메커니즘이 존재합니다. LDAP Directory Server에서 정보를 검색하면, *dir_cache* 데이터베이스 관리 프로그램 구성 매개변수 및 DB2LDAPCACHE 레지스트리 변수 값을 근거로 로컬 머신에 저장되거나 캐시됩니다. *dir_cache* 데이터베이스 관리 프로그램 구성 매개변수를 사용하여 메모리 캐시에 데이터베이스, 노드 및 DCS 디렉토리 파일을 저장합니다. 응용프로그램이 닫힐 때까지 응용프로그램은 디렉토리 캐시를 사용합니다. DB2LDAPCACHE 레지스트리 변수를 사용하여 로컬 디스크 캐시에 데이터베이스, 노드 및 DCS 디렉토리 파일을 저장합니다.

- DB2LDAPCACHE=NO이고 *dir_cache*=NO이면, 항상 LDAP에서 정보를 읽습니다.
- DB2LDAPCACHE=NO이고 *dir_cache*=YES이면, LDAP로부터 정보를 한 번 읽어서 DB2® 캐시로 삽입합니다.
- DB2LDAPCACHE=YES이거나 설정되지 않은 경우, LDAP로부터 정보를 한 번 읽어서 로컬 데이터베이스, 노드 및 DCS 디렉토리로 캐시합니다.

주: DB2LDAPCACHE 레지스트리 변수는 데이터베이스 및 노드 디렉토리에만 적용할 수 있습니다.

관련 개념:

- 83 페이지의 『LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스』
- 361 페이지의 『Active Directory 지원』
- 378 페이지의 『LDAP 지원 및 DB2 Connect』
- 379 페이지의 『LDAP 환경에서의 보안 고려사항』

- 380 페이지의 『Active Directory에 대한 보안 고려사항』
- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』
- 381 페이지의 『DB2 오브젝트 클래스 및 속성이 있는 LDAP 디렉토리 스키마 확장』

태스크 관련:

- 362 페이지의 『Active Directory를 사용할 수 있도록 DB2 구성』
- 363 페이지의 『IBM LDAP 환경에서 DB2 구성』
- 364 페이지의 『LDAP 사용자 작성』
- 365 페이지의 『DB2 응용프로그램에 대한 LDAP 사용자 구성』
- 366 페이지의 『설치 후 DB2 서버 등록』
- 368 페이지의 『DB2 서버에 대한 프로토콜 정보 갱신』
- 369 페이지의 『ATTACH를 위한 노드 별명 카탈로그화』
- 370 페이지의 『DB2 서버 등록 해제』
- 370 페이지의 『LDAP 디렉토리에 데이터베이스 등록』
- 371 페이지의 『LDAP 환경에서 리모트 서버에 접속』
- 372 페이지의 『LDAP 디렉토리에서 데이터베이스 등록 해제』
- 372 페이지의 『로컬 데이터베이스 및 노드 디렉토리에서 LDAP 항목 새로 고침』
- 373 페이지의 『LDAP 디렉토리 파티션 또는 도메인 검색』
- 374 페이지의 『LDAP에 호스트 데이터베이스 등록』
- 376 페이지의 『LDAP 환경의 사용자 레벨에서 DB2 레지스트리 변수 설정』
- 377 페이지의 『설치 완료 후 LDAP 지원 사용 가능화』
- 378 페이지의 『LDAP 지원 사용 불가능화』
- 381 페이지의 『Active Directory의 디렉토리 스키마 확장』

관련 참조:

- 360 페이지의 『지원되는 LDAP 클라이언트 및 서버 구성』
- 383 페이지의 『Active Directory의 DB2 오브젝트』
- 391 페이지의 『DB2에서 사용하는 LDAP 오브젝트 클래스 및 속성』
- 384 페이지의 『Netscape LDAP 디렉토리 지원 및 속성 정의』

지원되는 LDAP 클라이언트 및 서버 구성

다음 표는 지원되는 LDAP 클라이언트 및 서버 구성을 요약합니다.

표 22. 지원되는 LDAP 클라이언트 및 서버 구성

	IBM SecureWay Directory	Microsoft Active Directory	Netscape LDAP 서버
IBM LDAP 클라이언트	지원	지원	지원
Microsoft LDAP/ADSI 클라이언트	지원	지원	지원

IBM SecureWay Directory 버전 3.1은 Windows NT, AIX, Solaris 운영 환경 및 HP-UX에서 사용 가능한 LDAP 버전 3 서버입니다. SecureWay 디렉토리는 AIX 및 iSeries(AS/400)에서 기본 운영 체제의 일부로서 OS/390보안 서버와 함께 제공됩니다.

DB2 Universal Database™(DB2 UDB)의 32비트 버전은 AIX, Solaris 운영 환경, HP-UX 11.11, Windows, Linux IA32 및 Linux/390에서 IBM LDAP 클라이언트를 지원합니다.

Microsoft Active Directory는 LDAP 버전 3 서버이며 Windows 2000 서버 운영 체제의 일부로서 사용 가능합니다.

Microsoft LDAP 클라이언트는 Windows 운영 체제에 포함되어 있습니다.

Windows 운영 체제에서 실행할 때, DB2에서는 IBM LDAP 클라이언트와 Microsoft LDAP 클라이언트 중 하나를 사용하여 IBM SecureWay Directory Server에 액세스할 수 있도록 지원합니다. 명시적으로 IBM LDAP 클라이언트를 선택하려면, **db2set** 명령을 사용하여 DB2LDAP_CLIENT_PROVIDER 레지스트리 변수를 『IBM』에 설정하십시오.

관련 개념:

- 359 페이지의 『LDAP(Lightweight Directory Access Protocol) 입문』
- 361 페이지의 『Active Directory 지원』

Active Directory 지원

DB2® Universal Database (DB2 UDB)는 다음과 같이 Active Directory를 사용합니다.

1. DB2 데이터베이스 서버는 ibm_db2Node 오브젝트로 Active Directory에 발행됩니다. ibm_db2Node 오브젝트 클래스는 SCP(ServiceConnectionPoint 오브젝트 클래스의 서브클래스입니다. 각 ibm_db2Node 오브젝트에는 클라이언트 응용프로그램을 DB2 UDB 데이터베이스 서버에 연결하게 하는 프로토콜 구성 정보가 들어 있습니다. 새로운 데이터베이스가 작성되면, 데이터베이스는 ibm_db2Node 오브젝트 아래의 ibm_db2Database 오브젝트로서 Active Directory에 발행됩니다.
2. 리모트 데이터베이스에 연결하면, DB2 클라이언트는 ibm_db2Database 오브젝트에 대해 LDAP 인터페이스를 통해 Active Directory를 쿼리합니다. 데이터베이스 서

버에 연결하는 프로토콜 통신(바인딩 정보)은 ibm_db2Database 오브젝트가 작성되는 ibm_db2Node 오브젝트에서 얻습니다.

도메인 제어기에 있는 *Active Directory 사용자 및 컴퓨터* 관리 콘솔(MMC)을 사용하여 ibm_db2Node 및 ibm_db2Database 오브젝트에 대한 등록 정보 페이지를 볼 수 있습니다. 등록 정보 페이지를 설치하려면 다음과 같이 regsrv32 명령을 실행하여 DB2 오브젝트에 대한 등록 정보 페이지를 등록하십시오.

```
regsvr32 %DB2PATH%\bin\db2ads.dll
```

도메인 제어기에 있는 *Active Directory 사용자 및 컴퓨터* 관리 콘솔(MMC)을 사용하여 오브젝트를 볼 수 있습니다. 이 관리 도구로 가려면 시작 —> 프로그램 —> 관리 도구 —> Active Directory 사용자 및 컴퓨터를 차례로 선택하십시오.

주: 컴퓨터 오브젝트 아래에 있는 DB2 UDB 오브젝트를 표시하려면 보기 메뉴에서 *사용자, 그룹 및 컴퓨터를 컨테이너로*를 선택하십시오.

주: 도메인 제어기에 DB2 UDB가 설치되어 있지 않는 경우에도, %DB2PATH%\bin에 있는 db2ads.dll 파일과 %DB2PATH%\msg\locale-name에 있는 자원 DLL db2adsr.dll을 도메인 제어기의 로컬 디렉토리로 복사함으로써 DB2 UDB 오브젝트의 등록 정보 페이지를 볼 수 있습니다. (이들 두 개의 복사된 파일이 위치하는 디렉토리는 PATH 사용자/시스템 환경 변수에 있는 디렉토리 중 하나여야 합니다.) 그런 다음, 로컬 디렉토리에서 regsvr32 명령을 실행하여 DLL을 등록할 수 있습니다.

관련 개념:

- 380 페이지의 『Active Directory에 대한 보안 고려사항』

태스크 관련:

- 362 페이지의 『Active Directory를 사용할 수 있도록 DB2 구성』
- 381 페이지의 『Active Directory의 디렉토리 스키마 확장』

관련 참조:

- 383 페이지의 『Active Directory의 DB2 오브젝트』

Active Directory를 사용할 수 있도록 DB2 구성

프로시저:

Microsoft Active Directory에 액세스하려면, 다음과 같은 조건이 충족되어야 합니다.

1. DB2 Universal Database™(DB2 UDB)를 실행하는 머신은 Windows 2000 또는 Windows Server 2003 도메인에 속해야 합니다.

2. Microsoft LDAP 클라이언트가 설치되어 있습니다. Microsoft® LDAP 클라이언트는 Windows 2000, Windows XP 및 Windows Server 2003 운영 체제에 포함됩니다. Windows 98, Windows NT 또는 Windows Me의 경우, Active Directory 클라이언트 확장(wldap32.dll)이 시스템 디렉토리 아래 있음을 검증할 필요가 있습니다.
3. LDAP 지원을 사용 가능하게 하십시오. Windows 2000, Windows XP, 또는 Windows Server 2003의 경우, LDAP 지원은 설치 프로그램으로 사용 가능합니다. Windows 98, Windows NT 또는 Windows Me의 경우 **db2set** 명령을 사용하여 DB2_ENABLE_LDAP 레지스트리 변수를 『YES』로 설정하여 명시적으로 LDAP을 사용 가능하게 하십시오.
4. Active Directory에서 정보를 읽기 위해 DB2 UDB를 실행할 때 도메인 사용자 어카운트로 로그온하십시오.

관련 개념:

- 361 페이지의 『Active Directory 지원』
- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 365 페이지의 『DB2 응용프로그램에 대한 LDAP 사용자 구성』

IBM LDAP 환경에서 DB2 구성

프로시저:

IBM LDAP 환경에서 DB2를 사용하기 전에, 각 머신에서 다음을 구성해야 합니다.

- LDAP 지원을 사용 가능하게 하십시오. Windows 2000의 경우, LDAP 지원은 설치 프로그램으로 사용 가능합니다. Windows 98 또는 Windows NT의 경우, **db2set** 명령을 사용하여 DB2_ENABLE_LDAP 레지스트리 변수를 『YES』로 설정하여 명시적으로 LDAP를 사용 가능하게 해야 합니다. 모든 Windows 운영 체제에서 사용할 수 있는 디폴트 LDAP 클라이언트는 Microsoft 사의 것입니다. IBM LDAP 클라이언트를 사용하려면, **db2set** 명령을 사용하여 DB2LDAP_CLIENT_PROVIDER 레지스트리 변수를 『IBM』으로 설정해야 합니다.
- LDAP 서버의 TCP/IP 호스트 이름 및 포트 번호. 이 값은 DB2LDAPHOST 응답 키워드를 사용하여 자동 설치 중 입력되거나, 사용자가 DB2SET 명령을 사용하여 나중에 수동으로 설정할 수 있습니다.

  ```
  db2set DB2LDAPHOST=<hostname[:port]>
  ```

  여기서, hostname은 LDAP 서버의 TCP/IP 호스트 이름이며 [:port]는 포트 번호입니다. 포트 번호가 지정되지 않으면, DB2는 디폴트 LDAP 포트(389)를 사용합니다.

DB2 오브젝트는 LDAP 기본 식별 이름(baseDN)에 위치합니다. IBM SecureWay LDAP Directory Server 버전 3.1을 사용 중인 경우, DB2는 서버에서 동적으로 이 정보를 얻을 수 있으므로 기본 구분 이름을 구성할 필요는 없습니다. 그러나 IBM eNetwork Directory Server 버전 2.1을 사용하는 경우, DB2SET 명령을 사용하여 각 머신의 LDAP 기본 식별 이름을 구성해야 합니다.

```
db2set DB2LDAP_BASEDN=<baseDN>
```

여기서, baseDN은 LDAP 서버에 정의된 LDAP 접미부의 이름입니다. 이 LDAP 접미부는 DB2 오브젝트을 포함하는 데 사용됩니다.

- LDAP 사용자의 식별 이름(DN) 및 암호. 이것은 사용자가 LDAP를 사용하여 DB2 사용자 고유 정보를 저장하려고 계획하는 경우에만 필요합니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 362 페이지의 『Active Directory를 사용할 수 있도록 DB2 구성』
- 364 페이지의 『LDAP 사용자 작성』

관련 참조:

- *Command Reference*의 『db2set - DB2 Profile Registry Command』

LDAP 사용자 작성

프로시저:

DB2는 사용자 레벨에서 DB2 레지스트리 변수 및 CLI 구성을 설정하도록 지원합니다. 이것은 AIX 및 Solaris 플랫폼에서 사용 가능하지 않습니다. 사용자 레벨 지원은 다중 사용자 환경에서 사용자 고유 설정을 제공합니다. 시스템 환경이나 또다른 사용자 환경을 간섭하지 않고 각 로그온 사용자가 자신의 환경을 사용자 정의할 수 있는 Windows NT 터미널 서버가 그 예입니다.

IBM LDAP 디렉토리를 사용할 때, 사용자 레벨 정보를 LDAP에 저장하기 전에 LDAP 사용자를 정의해야 합니다. 다음 방법 중 하나로 LDAP 사용자를 작성할 수 있습니다.

- 사용자 오브젝트의 모든 속성을 포함할 LDIF 파일을 작성한 후, LDIF 임포트 유틸리티를 실행하여 오브젝트를 LDAP 디렉토리로 임포트하십시오. IBM LDAP 서버의 LDIF 유틸리티는 『LDIF2DB』입니다.
- IBM SecureWay LDAP Directory Server 버전 3.1에만 사용 가능한 DMT(Directory Management Tool)를 사용하여 사용자 오브젝트를 작성하십시오.

개인 오브젝트의 속성이 들어 있는 LDIF 파일이 다음과 유사하게 표시됩니다.

```
File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 UDB Development, ou=Toronto, o=ibm, c=ca
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

다음은 IBM LDIF 임포트 유틸리티를 사용하여 LDIF 파일을 임포트하는 LDIF 명령의 예입니다.

```
LDIF2DB -i newuser.ldif
```

주:

1. LDAP 서버 머신에서 LDIF2DB 명령을 실행해야 합니다.
2. LDAP 사용자가 자신의 오브젝트를 추가, 삭제, 읽기 및 작성할 수 있도록 LDAP 사용자 오브젝트에 필수 액세스(ACL)를 부여해야 합니다. 사용자 오브젝트의 ACL을 권한 부여하려면, LDAP Directory Server 웹 관리 도구를 사용하십시오.

태스크 관련:

- 363 페이지의『IBM LDAP 환경에서 DB2 구성』
- 365 페이지의『DB2 응용프로그램에 대한 LDAP 사용자 구성』

DB2 응용프로그램에 대한 LDAP 사용자 구성

프로시저:

Microsoft LDAP 클라이언트를 사용할 때, LDAP 사용자는 운영 체제 사용자 어카운트와 동일합니다. 그러나 IBM LDAP 클라이언트에 대한 작업을 수행할 때와 DB2를 사용하기 전에 현재 로그온된 사용자에 대한 LDAP 사용자 식별 이름(DN) 및 암호를 구성해야 합니다. 이것은 db2ldcfg 유틸리티를 사용하여 완료될 수 있습니다.

```
db2ldcfg -u <userDN> -w <password> -> set the user's DN and password
         -r                        -> clear the user's DN and password
```

예를 들어, 다음과 같습니다.

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 UDB Development,ou=Toronto,o=ibm,c=ca"
         -w password
```

태스크 관련:

- 364 페이지의『LDAP 사용자 작성』

관련 참조:

- *Command Reference*의『db2ldcfg - Configure LDAP Environment Command』

설치 후 DB2 서버 등록

프로시저:

클라이언트 응용프로그램이 DB2 서버 인스턴스로 연결하기 위해 사용하는 프로토콜 구성 정보를 발행하려면, 각 DB2 서버 인스턴스를 LDAP에 등록해야 합니다. 데이터베이스 서버 인스턴스를 등록할 때, 노드 *이름*을 지정해야 합니다. 노드 이름은 클라이언트 응용프로그램이 서버에 연결하거나 접속할 때 사용됩니다. **CATALOG LDAP NODE** 명령을 사용하여 LDAP 노드에 대한 또하나의 별명 이름을 카탈로그화할 수 있습니다.

주: Windows 2000 도메인 환경에서 작업하는 경우, 설치 중에 DB2 서버 인스턴스가 다음 정보로 Active Directory에 자동으로 등록됩니다.

```
nodename: TCP/IP hostname
protocol type: TCP/IP
```

TCP/IP 호스트 이름이 8자보다 긴 경우, 8자로 잘립니다.

REGISTER 명령은 다음과 같이 나타납니다.

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
```

protocol절은 이 데이터베이스 서버에 연결할 때 사용하는 통신 프로토콜을 지정합니다.

다중 실제 머신을 포함하는 DB2 Universal Database Enterprise Server Edition에 대한 인스턴스를 작성할 때는 각 머신당 한 번씩 **REGISTER** 명령을 호출해야 합니다. 모든 머신에 대해 **REGISTER** 명령을 발행하려면 **rah** 명령을 사용하십시오.

주: 각 머신은 LDAP에서 고유 이름을 가져야 하므로 동일한 ldap_node_name을 사용할 수 없습니다. **REGISTER** 명령에서 ldap_node_name을 각 머신의 호스트 이름으로 대체하고자 할 것입니다. 예를 들어, 다음과 같습니다.

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

『<>』는 **rah** 명령이 실행되는 각 머신의 호스트 이름으로 대체됩니다. 다중 DB2 Universal Database Enterprise Server Edition 인스턴스가 있는 드문 경우에, **rah** 명령에서 인스턴스와 호스트 인덱스의 조합을 노드 이름으로 사용할 수도 있습니다.

리모트 DB2 서버에 대해 **REGISTER** 명령을 발행할 수도 있습니다. 이를 수행하려면, 리모트 서버를 등록할 때 리모트 컴퓨터 이름, 인스턴스 이름 및 프로토콜 구성 매개변수를 지정해야 합니다. 이 명령은 다음과 같이 사용될 수 있습니다.

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
   hostname <host_name>
   svcename <tcpip_service_name>
   remote <remote_computer_name>
   instance <instance_name>
```

컴퓨터 이름에 대해 다음 규칙이 사용됩니다.

- TCP/IP가 구성되어 있을 경우, 컴퓨터 이름은 TCP/IP 호스트 이름과 동일해야 합니다.
- APPN이 구성된 경우, 컴퓨터 이름으로서 상대 LU 이름을 사용하십시오.

고가용성 또는 장애 복구 환경에서 실행 중이고 TCP/IP를 통신 프로토콜로서 사용할 경우, *클러스터* IP 주소가 사용되어야 합니다. 클러스터 IP 주소를 사용하여 클라이언트가 각 머신에 대해 별도의 TCP/IP 노드를 카탈로그화하지 않고도 머신에 있는 서버에 연결할 수 있도록 합니다. 클러스터 IP 주소는 다음과 같이 `hostname`절을 사용하여 지정됩니다.

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
   hostname n.nn.nn.nn
```

여기서, `n.nn.nn.nn`은 클러스터 IP 주소입니다.

관련 개념:

- 403 페이지의 『rah 및 db2_all 명령 개요』

태스크 관련:

- 368 페이지의 『DB2 서버에 대한 프로토콜 정보 갱신』
- 369 페이지의 『ATTACH를 위한 노드 별명 카탈로그화』
- 370 페이지의 『DB2 서버 등록 해제』
- 371 페이지의 『LDAP 환경에서 리모트 서버에 접속』

관련 참조:

- *Command Reference*의 『REGISTER Command』
- *Command Reference*의 『CATALOG LDAP NODE Command』

DB2 서버에 대한 프로토콜 정보 갱신

프로시저:

LDAP에 있는 DB2 서버 정보는 현재 정보로 보존되어야 합니다. 예를 들어, 프로토콜 구성 매개변수 또는 서버 네트워크 주소를 변경하면 LDAP에 대해서도 갱신해야 합니다.

로컬 머신의 LDAP에 있는 DB2 서버를 갱신하려면, 다음 명령을 사용하십시오.

```
db2 update ldap ...
```

갱신될 수 있는 프로토콜 구성 매개변수의 예는 다음을 포함합니다.

- TCP/IP 호스트 이름 및 서비스 이름 또는 포트 번호 매개변수
- APPC 프로토콜 정보(예: TP 이름, 상대 LU 또는 모드)
- NetBIOS 워크스테이션 이름

리모트 DB2 서버 프로토콜 구성 매개변수를 갱신하려면, node절이 있는 UPDATE LDAP 명령을 사용하십시오.

```
db2 update ldap
   node <node_name>
   hostname <host_name>
   svcename <tcpip_service_name>
```

태스크 관련:

- 366 페이지의 『설치 후 DB2 서버 등록』
- 369 페이지의 『ATTACH를 위한 노드 별명 카탈로그화』
- 371 페이지의 『LDAP 환경에서 리모트 서버에 접속』

관련 참조:

- *Command Reference*의 『UPDATE LDAP NODE Command』

다른 서버로 LDAP 클라이언트 재라우트

시스템 실패 시 클라이언트를 재라우트하는 기능과 같이, 동일한 기능이 LDAP에 대해 작업할 때에도 사용 가능합니다.

전제조건:

DB2_ENABLE_LDAP 레지스트리 변수가 『Yes』로 설정됩니다.

프로시저:

LDAP 환경 내에서 모든 데이터베이스 및 노드 디렉토리 정보는 LDAP 서버에서 유지보수됩니다. 클라이언트는 LDAP 디렉토리에서 정보를 검색합니다. DB2LDAPCACHE 레지스트리 변수가 『Yes』로 설정된 경우 해당 정보가 로컬 데이터베이스 및 노드 디렉토리에서 갱신됩니다.

LDAP에서 DB2 Universal Database™(DB2 UDB)를 나타내는 데이터베이스에 대하여 대체 서버를 정의하려면 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 명령을 사용하십시오.

이 대체 서버 정보가 설정되면 연결 시 클라이언트로 리턴됩니다.

주: 서버에서 UPDATE ALTERNATE SERVER FOR DATABASE 명령을 입력할 때(서버 인스턴스에서 『FOR LDAP DATABASE』가 아님을 주의)와, LDAP 지원이 서버에서 사용 가능한 경우(DB2_ENABLE_LDAP=Yes) 및 LDAP 사용자 ID 및 암호가 캐시된 경우(이전에 **db2ldcfg**가 실행됨), 데이터베이스에 대한 대체 서버가 LDAP 서버에서 자동으로 또는 내재적으로 갱신됩니다. 이 작업은 UPDATE ALTERNATE SERVER FOR LDAP DATABASE를 명시적으로 입력한 것처럼 수행됩니다.

관련 개념:

- 86 페이지의 『자동 클라이언트 재라우트 구현』
- 353 페이지의 『자동 클라이언트 재라우트 설명 및 설정』

ATTACH를 위한 노드 별명 카탈로그화

프로시저:

DB2 서버에 대한 노드 이름은 LDAP의 서버를 등록할 때 지정되어야 합니다. 응용프로그램은 데이터베이스 서버에 접속하기 위해 노드 이름을 사용합니다. 노드 이름이 응용프로그램에서 하드 코딩된 경우와 같이 다른 노드 이름이 필요할 경우, CATALOG LDAP NODE 명령을 사용하여 변경하십시오. 명령은 다음과 같습니다.

```
db2 catalog ldap node <ldap_node_name>
   as <new_alias_name>
```

LDAP 노드를 카탈로그 해제하려면, UNCATALOG LDAP NODE 명령을 사용하십시오. 명령은 다음과 같습니다.

```
db2 uncatalog ldap node <ldap_node_name>
```

태스크 관련:

- 366 페이지의 『설치 후 DB2 서버 등록』
- 371 페이지의 『LDAP 환경에서 리모트 서버에 접속』

관련 참조:

- *Command Reference*의 『CATALOG LDAP NODE Command』
- *Command Reference*의 『UNCATALOG LDAP NODE Command』

DB2 서버 등록 해제

프로시저:

LDAP에서 인스턴스의 등록을 해제하면 이 인스턴스에 대한 모든 노드 또는 별명, 오브젝트 및 데이터베이스 오브젝트도 제거합니다.

로컬 또는 리모트 머신에 있는 DB2 서버의 등록 해제는 이 서버에 대해 지정된 LDAP 노드 이름을 필요로 합니다.

```
db2 deregister db2 server in ldap
   node <node_name>
```

DB2 서버가 등록 해제될 때, 이 DB2 서버의 동일 인스턴스를 참조하는 모든 LDAP 노드 항목 및 LDAP 데이터베이스 항목도 카탈로그 해제됩니다.

태스크 관련:

- 366 페이지의 『설치 후 DB2 서버 등록』

관련 참조:

- *Command Reference*의 『DEREGISTER Command』

LDAP 디렉토리에 데이터베이스 등록

프로시저:

인스턴스에서 데이터베이스를 작성하는 동안 데이터베이스는 자동으로 LDAP에 등록됩니다. 이 등록은 클라이언트 머신에 데이터베이스 및 노드를 카탈로그화하지 않고도 리모트 클라이언트가 데이터베이스에 연결할 수 있게 합니다. 클라이언트가 데이트베이스에 연결하려 할 때, 데이터베이스가 로컬 머신의 데이터베이스 디렉토리에 존재하지 않으면, LDAP 디렉토리가 검색됩니다.

이 이름이 LDAP 디렉토리에 이미 있을 경우, 데이터베이스가 로컬 머신에서 여전히 작성되지만 LDAP 디렉토리에 이름 지정 충돌이 있음을 나타내는 경고 메시지가 리턴됩니다. 이러한 이유로 LDAP 디렉토리에서 데이터베이스를 수작업으로 카탈로그화할 수 있습니다. 사용자는 CATALOG LDAP DATABASE 명령을 사용하여 LDAP에 리모트 서버의 데이터베이스를 등록할 수 있습니다. 리모트 데이터베이스를 등록할 때, 리모트 데이터베이스 서버를 나타내는 LDAP 노드의 이름을 지정합니다. 데이터베이스를 등록하기 **전에** REGISTER DB2 SERVER IN LDAP 명령을 사용하여 리모트 데이터베이스 서버를 LDAP에 등록**해야 합니다**.

LDAP에 데이터베이스를 수작업으로 등록하려면, CATALOG LDAP DATABASE 명령을 사용하십시오.

```
db2 catalog ldap database <dbname>
   at node <node_name>
   with "My LDAP database"
```

태스크 관련:

- 366 페이지의『설치 후 DB2 서버 등록』
- 372 페이지의『LDAP 디렉토리에서 데이터베이스 등록 해제』

관련 참조:

- *Command Reference*의『CATALOG LDAP DATABASE Command』

LDAP 환경에서 리모트 서버에 접속

프로시저:

LDAP 환경에서 ATTACH 명령에 LDAP 노드 이름을 사용하여 리모트 데이터베이스 서버에 접속할 수 있습니다.

```
db2 attach to <ldap_node_name>
```

클라이언트 응용프로그램이 처음으로 노드 또는 데이터베이스에 접속할 경우, 노드가 로컬 노드 디렉토리에 없으므로 데이터베이스 관리 프로그램이 LDAP 디렉토리에서 목표 노드 항목을 검색합니다. LDAP 디렉토리에서 항목이 발견되지 않으면, 리모트 서버의 프로토콜 정보가 검색됩니다. 데이터베이스에 연결하며 항목이 LDAP 디렉토리에 있는 경우, 데이터베이스 정보도 검색됩니다. 이 정보를 사용하여 데이터베이스 관리가 자동으로 로컬 머신에 데이터베이스 항목 및 노드 항목을 카탈로그화합니다. 클라이언트 응용프로그램이 다음 번에 같은 노드 또는 데이터베이스에 접속할 때에는 LDAP 디렉토리를 검색하지 않고 로컬 데이터베이스 디렉토리에 있는 정보를 사용합니다.

자세히 설명하면 클라이언트가 LDAP 서버는 한 번만 검색하도록 캐싱 메커니즘이 존재합니다. 정보를 검색할 경우, *dir_cache* 데이터베이스 관리 프로그램 구성 매개변수 및 DB2LDAPCACHE 레지스트리 변수 값을 근거로 로컬 머신에 저장되거나 캐시됩니다.

- DB2LDAPCACHE=NO이고 *dir_cache*=NO이면, 항상 LDAP에서 정보를 읽습니다.
- DB2LDAPCACHE=NO이고 *dir_cache*=YES이면, LDAP에서 정보를 한 번 읽어서 DB2 캐시로 삽입합니다.
- DB2LDAPCACHE=YES이거나 설정되지 않은 경우, LDAP 서버로부터 정보를 한 번 읽어서 로컬 데이터베이스, 노드 및 DCS 디렉토리로 캐시합니다.

주: LDAP 정보의 캐싱은 사용자 레벨 CLI 또는 DB2 프로파일 레지스트리 변수에 적용할 수 없습니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 366 페이지의 『설치 후 DB2 서버 등록』
- 368 페이지의 『DB2 서버에 대한 프로토콜 정보 갱신』
- 369 페이지의 『ATTACH를 위한 노드 별명 카탈로그화』
- 370 페이지의 『LDAP 디렉토리에 데이터베이스 등록』

관련 참조:

- *Command Reference*의 『ATTACH Command』

LDAP 디렉토리에서 데이터베이스 등록 해제

프로시저:

데이터베이스는 다음 경우 LDAP로부터 자동으로 등록 해제됩니다.

- 데이터베이스가 제거된 경우
- 소유하는 인스턴스가 LDAP로부터 등록 해제된 경우

다음 명령을 사용하여 수동으로 LDAP로부터 데이터베이스를 등록 해제할 수 있습니다.

```
db2 uncatalog ldap database <dbname>
```

태스크 관련:

- 370 페이지의 『LDAP 디렉토리에 데이터베이스 등록』

관련 참조:

- *Command Reference*의 『UNCATALOG LDAP DATABASE Command』

로컬 데이터베이스 및 노드 디렉토리에서 LDAP 항목 새로 고침

프로시저:

LDAP 정보는 변경되기 쉬우므로 로컬 및 노드 디렉토리에 있는 LDAP 항목을 새로 고칠 필요가 있습니다. 로컬 데이터베이스 및 노드 디렉토리는 LDAP에 있는 항목을 캐시하는 데 사용됩니다.

자세히 설명하면 클라이언트가 LDAP 서버는 한 번만 검색하도록 캐싱 메커니즘이 존재합니다. 정보를 검색할 경우, *dir_cache* 데이터베이스 관리 프로그램 구성 매개변수 및 DB2LDAPCACHE 레지스트리 변수 값을 근거로 로컬 머신에 저장되거나 캐시됩니다.

- DB2LDAPCACHE=NO이고 *dir_cache*=NO이면, 항상 LDAP에서 정보를 읽습니다.
- DB2LDAPCACHE=NO이고 *dir_cache*=YES이면, LDAP에서만 정보를 읽어 DB2 캐시로 삽입합니다.
- DB2LDAPCACHE=YES이거나 설정되지 않은 경우, LDAP 서버로부터 정보를 한 번 읽어서 로컬 데이터베이스, 노드 및 DCS 디렉토리로 캐시합니다.

주: LDAP 정보의 캐싱은 사용자 레벨 CLI 또는 DB2 프로파일 레지스트리 변수에 적용할 수 없습니다.

LDAP 자원을 참조하는 데이터베이스 항목을 새로 고치려면, 다음 명령을 사용하십시오.

```
db2 refresh ldap database directory
```

LDAP 자원을 참조하는 로컬 머신의 노드 항목을 새로 고치려면, 다음 명령을 사용하십시오.

```
db2 refresh ldap node directory
```

새로 고치는 작업의 일부로서, 로컬 데이터베이스 및 노드 디렉토리에 저장되어 있는 모든 LDAP 항목이 제거됩니다. 다음에 응용프로그램이 데이터베이스 또는 노드에 액세스할 때, 응용프로그램은 LDAP로부터 직접 정보를 읽고 로컬 데이터베이스 또는 노드 디렉토리에 새로운 항목을 생성합니다.

시기 적절한 방법으로 새로 고침이 수행되었는지 확인하기 위해 다음을 수행하고자 할 수 있습니다.

- 정기적으로 실행되는 새로 고침을 스케줄함
- 시스템 시동시 REFRESH 명령을 실행함
- 모든 클라이언트 머신에 대해 REFRESH 명령을 호출하기 위해 사용 가능한 관리 패키지를 사용함
- DB2LDAPCACHE=『NO』를 설정하여 데이터베이스, 노드 및 DCS 디렉토리에 LDAP 정보가 캐시되는 것을 피할 수 있습니다.

관련 개념:

- *관리 안내서: 성능*의『DB2 레지스트리 및 환경 변수』

관련 참조:

- *관리 안내서: 성능*의 『dir_cache - 디렉토리 캐시 지원 구성 매개변수』
- *Command Reference*의 『REFRESH LDAP Command』

LDAP 디렉토리 파티션 또는 도메인 검색

프로시저:

DB2 UDB는 현재 LDAP 디렉토리 파티션 및 현재 활성 디렉토리 도메인을 Windows 2000 환경에서 검색합니다. 다중 LDAP 디렉토리 파티션 또는 도메인이 있는 환경에서 검색 영역을 설정할 수 있습니다. 예를 들어, 현재 파티션 또는 도메인에서 정보가 발견되지 않으면, 모든 파티션 또는 도메인의 자동 검색이 요청될 수 있습니다. 반면에 검색 영역이 로컬 머신만 검색하도록 제한될 수도 있습니다.

검색 영역은 DB2 UDB 프로파일 레지스트리 변수인 DB2LDAP_SEARCH_SCOPE를 통해 제어됩니다. 검색 영역 값을 LDAP의 전역 레벨에서 설정하려면, *db2set* 명령에 『-gl』 옵션을 사용하십시오. 이 옵션은 『LDAP의 전역』을 나타냅니다.

```
db2set -gl db2ldap_search_scope = <value>
```

사용 가능한 값은 『local』, 『domain』 또는 『global』을 포함합니다. 디폴트값은 현재 디렉토리 파티션으로 검색 범위를 제한하는 『domain』입니다. LDAP의 검색 영역을 설정하면 전체 엔터프라이즈에 대한 디폴트 검색 영역을 설정할 수 있습니다. 예를 들어, 새로운 데이터가 작성된 다음에 검색 영역을 『global』로 초기화하고자 할 수 있습니다. 이는 모든 클라이언트 머신이 다른 모든 파티션이나 도메인을 검색하게 하여 특정 파티션 또는 도메인에 정의된 데이터베이스를 찾을 수 있도록 합니다. 각 클라이언트에 대한 최초 연결 또는 접속 후 항목이 각 머신에 기록된 다음, 검색 영역을 『local』로 변경할 수 있습니다. 일단 『local』로 변경되면, 각 클라이언트는 어떠한 파티션 또는 도메인도 스캔하지 않습니다.

주: DB2 UDB 프로파일 레지스트리 변수 DB2LDAP_KEEP_CONNECTION 및 DB2LDAP_SEARCH_SCOPE만 LDAP의 전역 레벨에 변수 설정을 지원하는 레지스트리 변수입니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 33 페이지의 『레지스트리 및 환경 변수 선언』

LDAP에 호스트 데이터베이스 등록

프로시저:

호스트 데이터베이스를 LDAP에 등록할 때, 두 개의 가능한 구성이 있습니다.

- 호스트 데이터베이스에 대한 직접 연결
- 게이트웨이를 통한 호스트 데이터베이스에 대한 연결

처음 경우에, 사용자는 LDAP에서 호스트 서버를 등록한 후, 호스트 서버의 노드 이름을 지정하는 LDAP에 호스트 데이터베이스를 카탈로그화합니다. 두 번째 경우에, 사용

자는 LDAP에서 게이트웨이 서버를 등록한 후, 게이트웨이 서버의 노드 이름을 지정하는 LDAP에 호스트 데이터베이스를 카탈로그화합니다.

DB2® Connect 게이트웨이에서 LDAP 지원이 사용 가능하고 게이트웨이 데이터베이스 디렉토리에 데이터베이스가 없는 경우, DB2 UDB는 LDAP를 찾으며 발견된 정보를 보존하려고 합니다.

양쪽 경우를 표시하는 예로서, 다음을 고려하십시오. NIAGARA_FALLS라고 하는 호스트 데이터베이스가 있음을 가정하십시오. APPN 및 TCP/IP를 사용하여 들어오는 연결을 승인할 수 있습니다. 클라이언트가 DB2 Connect를 가지고 있지 않아서 호스트에 직접 연결할 수 없는 경우, 『goto@niagara』라고 하는 게이트웨이를 사용하여 연결합니다.

다음 단계를 완료해야 합니다.

1. APPN 연결성을 위해서는 호스트 데이터베이스 서버를 LDAP에 등록하십시오. REMOTE 및 INSTANCE절은 임의입니다. NODETYPE절은 『DCS』로 설정되어 이것이 호스트 데이터베이스 서버임을 나타냅니다.

   ```
   db2 register ldap as nfappn appn network CAIBMOML partnerlu NFLU
      mode IBMRDB remote mvssys instance msvinst nodetype dcs
   ```

2. TCP/IP 연결성을 위해서는 호스트 데이터베이스 서버를 LDAP에 등록하십시오. 서버의 TCP/IP 호스트 이름은 『myhost』이며 포트 번호는 『446』입니다. 1단계와 유사하며, NODETYPE절은 『DCS』로 설정되어 이것이 호스트 데이터베이스 서버임을 나타냅니다.

   ```
   db2 register ldap as nftcpip tcpip hostname myhost svcename 446
      remote mvssys instance mvsinst nodetype dcs
   ```

3. TCP/IP 연결성을 위해서는 DB2 Connect 게이트웨이 서버를 LDAP에 등록하십시오. 게이트웨이 서버의 TCP/IP 호스트 이름은 『niagara』이며 포트 번호는 『50000』입니다.

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

4. APPN 연결성을 사용하여 LDAP에서 호스트 데이터베이스를 카탈로그화하십시오. 호스트 데이터베이스 이름은 『NIAGARA_FALLS』이며, 데이터베이스 별명 이름은 『nftcpip』입니다. GWNODE절은 DB2 Connect 게이트웨이 서버의 노드 이름을 지정하기 위해 사용됩니다.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
      gwnode whasf authentication dcs
   ```

5. APPN 연결성을 사용하여 LDAP에서 호스트 데이터베이스를 카탈로그화하십시오.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nfappn at node nfappn
      gwnode whasf authentication dcs
   ```

위에 표시한 등록 및 카탈로그를 완료한 후, TCP/IP를 사용하여 호스트에 연결하려는 경우 『nftcpip』에 연결합니다. APPN을 사용하여 호스트에 연결하려는 경우, 『nfappn』에 연결합니다. 클라이언트 워크스테이션에 DB2 Connect를 갖지 않은 경우, TCP/IP를 사용하여 게이트웨이를 통해 연결이 진행되며, 거기에서 『nftcpip』 또는 『nfappn』을 사용하는지 여부에 따라 TCP/IP 또는 APPN을 각각 사용하여 호스트에 연결합니다.

각 클라이언트가 수동으로 데이터베이스와 노드를 각 머신에 로컬로 카탈로그화할 필요가 없도록 LDAP에 호스트 데이터베이스 정보를 수동으로 구성할 수 있습니다. 프로세스는 다음과 같습니다.

1. 호스트 데이터베이스 서버를 LDAP에 등록하십시오. REMOTE, INSTANCE 및 NODETYPE절을 각각 사용하여 REGISTER 명령에서 호스트 데이터베이스 서버에 대해 리모트 컴퓨터 이름, 인스턴스 이름 및 노드 유형을 지정해야 합니다. REMOTE절은 호스트 이름 또는 호스트 서버 머신의 LU 이름 중 하나로 설정될 수 있습니다. INSTANCE절은 8자 이하인 문자열로 설정될 수 있습니다. 예를 들어, 인스턴스 이름은 『DB2』로 설정될 수 있습니다. NODE TYPE절은 『DCS』로 설정되어야 이것이 호스트 데이터베이스 서버임을 나타낼 수 있습니다.
2. CATALOG LDAP DATABASE 명령을 사용하여 호스트 데이터베이스를 LDAP에 등록하십시오. PARMS절을 사용하여 DRDA 매개변수를 추가 지정할 수 있습니다. 데이터베이스 인증 유형은 『DCS』로 설정되어야 합니다.

관련 참조:

- *Command Reference*의 『REGISTER Command』
- *Command Reference*의 『CATALOG LDAP DATABASE Command』

LDAP 환경의 사용자 레벨에서 DB2 레지스트리 변수 설정

프로시저:

LDAP 환경하에서 DB2 프로파일 레지스트리 변수는 사용자의 DB2 환경을 사용자 정의할 수 있는 사용자 레벨에 설정될 수 있습니다. DB2 프로파일 레지스트리 변수를 사용자 레벨에 설정하려면, `-ul` 옵션을 사용하십시오.

```
db2set -ul <variable>=<value>
```

주: 이것은 AIX 또는 Solaris 운영 환경에서 지원되지 않습니다.

DB2에는 캐싱하는 메커니즘이 있습니다. 사용자 레벨의 DB2 프로파일 레지스트리 변수는 로컬 머신에서 캐시됩니다. `-ul` 매개변수가 지정된 경우, DB2는 항상 캐시에서 DB2 레지스트리 변수를 읽어옵니다. 캐시는 다음 경우에 새로 고쳐집니다.

- 사용자 레벨에서 DB2 레지스트리 변수를 갱신하거나 재설정하는 경우
- 서버 레벨에서 LDAP 프로파일 변수를 새로 고치는 명령은 다음과 같습니다.

```
db2set -ur
```

태스크 관련:

- 33 페이지의 『레지스트리 및 환경 변수 선언』

관련 참조:

- *Command Reference*의 『db2set - DB2 Profile Registry Command』

설치 완료 후 LDAP 지원 사용 가능화

프로시저:

설치 프로세스 완료에 이어 일부 지점에서 LDAP 지원을 사용 가능하게 하려면, 각 머신에서 다음 프로시저를 사용하십시오.

- LDAP 지원 2진 파일을 설치하십시오. 설치 프로그램을 실행하여 사용자 설치로부터 LDAP 디렉토리 개발 지원을 선택하십시오. 설치 프로그램이 2진 파일을 설치하고 DB2 프로파일 레지스트리 변수 DB2_ENABLE_LDAP를 『YES』로 설정합니다.

  주: Windows 98, Windows NT 및 UNIX 플랫폼의 경우, **db2set** 명령으로 DB2_ENABLE_LDAP 레지스트리 변수를 『YES』로 설정하여 LDAP를 명시적으로 사용 가능하게 해야 합니다.

- (UNIX 플랫폼에서만) 다음 명령을 사용하여 LDAP 서버의 TCP/IP 호스트 이름 및 (선택적) 포트 번호를 선언하십시오.

  ```
  db2set DB2LDAPHOST=<base_domain_name>[:port_number]
  ```

  여기서, base_domain_name은 LDAP 서버의 TCP/IP 호스트 이름이며, [:port]는 포트 번호입니다. 포트 번호가 지정되지 않으면, DB2는 디폴트 LDAP 포트(389)를 사용합니다.

  DB2 오브젝트는 LDAP 기본 식별 이름(baseDN)에 위치합니다. IBM SecureWay LDAP Directory Server 버전 3.1을 사용 중인 경우, DB2는 서버에서 동적으로 이 정보를 얻을 수 있으므로 기본 구분 이름을 구성할 필요는 없습니다. 그러나 IBM eNetwork Directory Server 버전 2.1을 사용하는 경우, DB2SET 명령을 사용하여 각 머신의 LDAP 기본 식별 이름을 구성해야 합니다.

  ```
  db2set DB2LDAP_BASEDN=<baseDN>
  ```

  여기서, baseDN은 LDAP 서버에 정의된 LDAP 접미부의 이름입니다. 이 LDAP 접미부는 DB2 오브젝트을 포함하는 데 사용됩니다.

- REGISTER LDAP AS 명령을 사용하여 LDAP에 DB2 서버의 현재 인스턴스를 등록하십시오. 예를 들어, 다음과 같습니다.

  ```
  db2 register ldap as <node-name> protocol tcpip
  ```

- LDAP에 등록하고자 하는 데이터베이스가 있으면, CATALOG LDAP DATABASE 명령을 실행하십시오. 예를 들어, 다음과 같습니다.

  ```
  db2 catalog ldap database <dbname> as <alias_dbname>
  ```

- LDAP 사용자의 식별 이름(DN) 및 암호를 입력하십시오. 이것은 사용자가 LDAP를 사용하여 DB2 사용자 고유 정보를 저장하려고 계획하는 경우에만 필요합니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

태스크 관련:

- 378 페이지의 『LDAP 지원 사용 불가능화』

관련 참조:

- *Command Reference*의 『REGISTER Command』
- *Command Reference*의 『db2set - DB2 Profile Registry Command』
- *Command Reference*의 『CATALOG LDAP DATABASE Command』

LDAP 지원 사용 불가능화

프로시저:

LDAP 지원을 사용 불가능하게 하려면, 다음 프로시저를 사용하십시오.

- DB2 서버의 각 인스턴스의 경우, LDAP로부터 DB2 서버를 등록 해제하십시오.

  ```
  db2 deregister db2 server in ldap node <nodename>
  ```

- DB2 프로파일 레지스트리 변수 DB2_ENABLE_LDAP를 『NO』로 설정하십시오.

태스크 관련:

- 33 페이지의 『레지스트리 및 환경 변수 선언』
- 377 페이지의 『설치 완료 후 LDAP 지원 사용 가능화』

관련 참조:

- *Command Reference*의 『DEREGISTER Command』

LDAP 지원 및 DB2 Connect

DB2® Connect 게이트웨이에서 LDAP 지원이 사용 가능하고 게이트웨이 데이터베이스 디렉토리에 데이터베이스가 없는 경우, DB2는 LDAP를 찾으며 발견된 정보를 보존하려고 합니다.

관련 개념:

- 359 페이지의 『LDAP(Lightweight Directory Access Protocol) 입문』

• 379 페이지의 『LDAP 환경에서의 보안 고려사항』

LDAP 환경에서의 보안 고려사항

LDAP 디렉토리에 있는 정보를 액세스하기 전에, 응용프로그램 또는 사용자가 LDAP 서버에 의해 인증됩니다. 이러한 인증 프로세스를 LDAP 서버로의 *바인딩*이라고 합니다.

익명의 사용자가 정보를 추가, 삭제 또는 수정하는 것을 막기 위해 LDAP 디렉토리에 저장된 정보에 대해 액세스 제어를 적용하는 것은 중요한 일입니다.

액세스 제어는 디폴트로 상속되며 컨테이너 레벨에서 적용될 수 있습니다. 새로운 오브젝트가 작성될 때, 이것은 상위 오브젝트와 동일한 보안 속성을 상속합니다. LDAP 서버에 사용할 수 있는 관리 도구를 사용하여 컨테이너 오브젝트에 대한 액세스 제어를 정의할 수 있습니다.

디폴트로, 액세스 제어는 다음과 같이 정의됩니다.

- LDAP에 있는 데이터베이스 및 노드 항목의 경우, 모든 사용자(또는 모든 익명의 사용자)가 읽기 액세스 권한을 가집니다. 디렉토리 관리자 및 오브젝트의 소유자 또는 작성자만 읽기/쓰기 액세스 권한을 가집니다.
- 사용자 프로파일의 경우, 프로파일 소유자와 디렉토리 관리자는 읽기/쓰기 액세스 권한을 가집니다. 디렉토리 관리자 권한을 가지고 있지 않은 사용자는 또다른 사용자의 프로파일에 액세스할 수 없습니다.

주: 권한 부여 검사는 LDAP 서버에 의해 항상 수행되며, DB2®에 의해서는 수행되지 않습니다. LDAP 권한 부여 점검은 DB2 권한 부여와 관련이 없습니다. SYSADM 권한을 가지는 어카운트 또는 권한 ID는 LDAP 디렉토리에 대해 액세스 권한이 없을 수도 있습니다.

LDAP 명령 또는 API를 실행할 때, 바인드 식별 이름(bindDN) 및 암호가 지정되지 않은 경우, DB2는 요청된 명령을 수행하는 충분한 권한이 없을 수 있는 디폴트 자격 사항을 사용하여 LDAP 서버에 바인드하여 오류를 리턴합니다.

DB2 명령 또는 API에 대해 USER 및 PASSWORD절을 사용하여 사용자의 bindDN 및 암호를 명시적으로 지정할 수 있습니다.

관련 개념:

• 380 페이지의 『Active Directory에 대한 보안 고려사항』

Active Directory에 대한 보안 고려사항

DB2® 데이터베이스 및 노드 오브젝트는 Active Directory에 DB2 서버가 설치된 머신의 컴퓨터 오브젝트 아래에 작성됩니다. Active Directory에서 데이터베이스 서버를 등록하거나 데이터베이스를 카탈로그화하려면, 컴퓨터 오브젝트에서 오브젝트를 작성 또는 갱신하기에 충분한 액세스를 가질 필요가 있습니다.

디폴트로, 컴퓨터 오브젝트 아래에 있는 오브젝트는 인증된 사용자가 읽을 수 있으며 관리자(Administrators, Domain Administrators 및 Enterprise Administrators 그룹에 속하는 사용자)가 갱신할 수 있습니다. 특정 사용자 또는 그룹에 대한 액세스를 부여하려면, 다음과 같이 *Active Directory 사용자 및 컴퓨터* 관리 콘솔(MMC)을 사용하십시오.

1. *Active Directory 사용자 및 컴퓨터* 관리 도구를 시작하십시오.

   (시작 —> 프로그램 —> 관리 도구 —> Active Directory 사용자 및 컴퓨터)
2. *보기* 아래에서 *고급 기능*을 선택하십시오.
3. *Computers* 컨테이너를 선택하십시오.
4. DB2가 설치된 서버 머신을 나타내는 컴퓨터 오브젝트를 마우스 오른쪽 단추로 누른 후, 등록 *정보*를 선택하십시오.
5. *보안* 탭을 선택한 후, 지정된 사용자 또는 그룹에 필수 액세스를 추가하십시오.

사용자 레벨에서의 DB2 레지스트리 변수 및 CLI 설정은 사용자 오브젝트 아래의 DB2 등록 정보 오브젝트에 유지보수됩니다. 사용자 레벨에서 DB2 레지스트리 변수 또는 CLI 설정을 설정하려면, 사용자가 사용자 오브젝트 아래에서 오브젝트를 작성할 충분한 액세스를 가져야 합니다.

디폴트로, 관리자만이 사용자 오브젝트 아래에서 오브젝트를 작성할 액세스를 갖습니다. 사용자 레벨에서 DB2 레지스트리 변수 또는 CLI 설정을 설정하게 하는 액세스를 부여하려면, 다음과 같이 *Active Directory 사용자 및 컴퓨터* 관리 콘솔(MMC)을 사용하십시오.

1. *Active Directory 사용자 및 컴퓨터* 관리 도구를 시작하십시오.

   (시작 —> 프로그램 —> 관리 도구 —> Active Directory 사용자 및 컴퓨터)
2. Users 컨테이너 아래에서 사용자 오브젝트를 선택하십시오.
3. 사용자 오브젝트를 마우스 오른쪽 단추로 누른 후, 등록 *정보*를 선택하십시오.
4. *보안* 탭을 선택하십시오.
5. 추가 단추를 사용하여 목록에 사용자 이름을 추가하십시오.
6. 『쓰기』 및 『모든 하위 오브젝트 작성』 액세스를 부여하십시오.

7. 고급 설정을 사용하여, 『이 오브젝트 및 모든 하위 오브젝트』에 적용하는 사용 권한을 설정하십시오.
8. 『이 오브젝트로 전파하도록 상위에서 상속가능한 사용 권한 허용』 선택란을 선택하십시오.

관련 개념:

• 379 페이지의 『LDAP 환경에서의 보안 고려사항』

DB2 오브젝트 클래스 및 속성이 있는 LDAP 디렉토리 스키마 확장

LDAP 디렉토리 스키마는 LDAP 디렉토리 항목에 저장된 정보에 대해 오브젝트 클래스 및 속성을 정의합니다. 오브젝트 클래스는 필수 및 선택적 속성 세트로 이루어져 있습니다. LDAP 디렉토리의 모든 항목은 연관된 오브젝트 클래스를 가집니다.

DB2®를 LDAP에 정보를 저장하기 전에, LDAP 서버의 디렉토리 스키마가 DB2가 사용하는 오브젝트 클래스 및 속성을 포함해야 합니다. 새로운 오브젝트 클래스 및 속성을 기본 스키마에 추가하는 프로세스를 디렉토리 스키마 확장이라고 합니다.

주: IBM® SecureWay® LDAP Directory v3.1을 사용 중인 경우, DB2 UDB 버전 8.1 이전 버전에 필요한 모든 오브젝트 클래스 및 속성은 기본 스키마에 포함됩니다. 이 때 DB2 오브젝트 클래스 및 속성을 사용하여 기본 스키마를 확장할 필요는 없으나, 기본 스키마에는 포함되지 않는 두 개의 새로운 속성이 DB2 UDB 버전 8.2에 대하여 존재합니다. 이 경우 두 개의 새로운 DB2 UDB 속성으로 기본 스키마를 확장해야 합니다.

관련 개념:

• 386 페이지의 『IBM SecureWay Directory Server의 디렉토리 스키마 확장』

태스크 관련:

• 381 페이지의 『Active Directory의 디렉토리 스키마 확장』

Active Directory의 디렉토리 스키마 확장

프로시저:

DB2 Universal Database™(DB2 UDB)가 Windows 2000 Active Directory에 정보를 저장하기 전에, 디렉토리 스키마가 새로운 DB2 UDB 오브젝트 클래스 및 속성을 포함하도록 확장될 필요가 있습니다. 새로운 오브젝트 클래스 및 속성을 디렉토리 스키마에 추가하는 프로세스를 *스키마 확장*이라고 합니다.

Windows 도메인의 일부인 임의 머신에서 DB2 UDB의 처음 설치 이전에 DB2 UDB 스키마 설치 프로그램 **db2schex**를 실행하여 Active Directory에 대한 스키마를 확장해야 합니다.

db2schex 프로그램은 CD-ROM 제품에 있습니다. CD-ROM에서 이 프로그램의 위치는 db2 디렉토리, windows 서브디렉토리 및 utilities 서브디렉토리 아래에 있습니다. 예를 들면, 다음과 같습니다.

```
x:\db2\windows\utilities\
```

여기서, x:는 CD-ROM 드라이브입니다.

명령은 다음과 같이 사용됩니다.

```
db2schex
```

이 명령에 연관된 기타 선택적 절이 있습니다.

- -b UserDN

  사용자 식별 이름 지정

- -w Password

  바인드 암호 지정

- -u

  스키마 설치 제거

- -k

  오류를 무시하고 설치 제거를 계속 강제 수행

주:

1. UserDN 및 암호가 지정되지 않은 경우, **db2schex**는 현재 로그된 사용자로서 바인드합니다.
2. userDN절은 Windows NT 사용자 이름으로서 지정될 수 있습니다.
3. 스키마를 갱신하려면, 스키마 관리자의 구성원이어야 하거나 스키마를 갱신하기 위한 권한이 위임되어 있어야 합니다.

디렉토리 스키마를 확장하려면 DB2 UDB 버전 8.2 제품과 함께 제공되는 **db2schex.exe** 명령을 실행해야 합니다.

Windows용 DB2 제품의 이전 버전에 제공된 **db2schex.exe** 명령을 실행한 경우, DB2 UDB 버전 8.2에 제공되는 이 명령을 다시 실행하면 ibm-db2Database 클래스에 다음과 같은 두 가지 선택적 속성이 추가됩니다.

```
   ibm-db2AltGwPtr
ibm-db2NodePtr
```

Windows용 DB2 UDB 제품의 이전 버전과 함께 제공된 **db2schex.exe** 명령을 실행하지 않은 경우, DB2 버전 8.2에 제공되는 이 명령을 다시 실행하면 DB2 UDB LDAP 지원을 위한 모든 클래스 및 속성이 추가됩니다.

예:

- DB2 UDB 스키마를 설치하려면, 다음을 입력하십시오.

```
db2schex
```

- DB2 UDB 스키마를 설치하고 바인드 DN 및 암호를 지정하려면, 다음을 입력하십시오.

```
db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
         -w password
```

  또는

```
db2schex -b Administrator -w password
```

- DB2 UDB 스키마를 설치 제거하려면, 다음을 입력하십시오.

```
db2schex -u
```

- DB2 UDB 스키마를 설치 제거하고 오류를 무시하려면, 다음을 입력하십시오.

```
db2schex -u -k
```

Active Directory의 DB2 UDB 스키마 설치 프로그램은 다음과 같은 태스크를 수행합니다.

주:

1. 어느 서버가 스키마 마스터인지 검출
2. 스키마 마스터인 도메인 제어기에 바인드
3. 사용자가 클래스와 속성을 스키마에 추가할 수 있는 권한을 가지고 있는지 확인
4. 스키마 마스터가 쓰기 가능한지 확인(즉, 레지스트리에서 안전 interlock이 제거됨)
5. 새로운 속성 작성
6. 새로운 오브젝트 클래스 작성
7. 오류를 검출하고, 오류가 발생하면 프로그램은 스키마에 대한 변경사항을 롤백

관련 개념:

- 381 페이지의 『DB2 오브젝트 클래스 및 속성이 있는 LDAP 디렉토리 스키마 확장』

Active Directory의 DB2 오브젝트

DB2는 두 위치에서 Active Directory에 오브젝트를 작성합니다.

1. DB2 데이터베이스 및 노드 오브젝트는 DB2 서버가 설치된 머신의 컴퓨터 오브젝트 아래에 작성됩니다. Windows NT 도메인에 속하지 않는 DB2 서버 머신의 경우, DB2 데이터베이스 및 노드 오브젝트는 『System』 컨테이너 아래에 작성됩니다.
2. 사용자 레벨에서의 DB2 레지스트리 변수 및 CLI 설정은 사용자 오브젝트 아래의 DB2 등록 정보 오브젝트에 저장됩니다. 이들 오브젝트에는 해당 사용자에게 고유한 정보가 들어 있습니다.

관련 참조:

• 391 페이지의 『DB2에서 사용하는 LDAP 오브젝트 클래스 및 속성』

Netscape LDAP 디렉토리 지원 및 속성 정의

Netscape LDAP 서버에 대한 지원 레벨은 v4.12 이상입니다.

Netscape LDAP 서버 버전 4.12 이상에서 Netscape 디렉토리 서버는 응용프로그램이 `slapd.user_oc.conf` 및 `slapd.user_at.conf` 파일에 속성 및 오브젝트 클래스 정의를 추가함으로써 스키마를 확장할 수 있도록 합니다. 이 두 파일은 다음 디렉토리에 위치합니다.

```
<Netscape_install path>\slapd-<machine_name>\config
```

주: iPlan Directory Server 5.0을 사용할 경우에는, 이 제품과 함께 제공되는 문서에서 스키마 확장에 대한 자세한 지시사항을 검토해야 합니다.

다음과 같이 DB2 속성을 `slapd.user_at.conf`에 추가해야 합니다.

```
########################################################################
#
# IBM DB2 Universal Database
# Attribute Definitions
#
# bin -> binary
# ces -> case exact string
# cis -> case insensitive string
# dn -> distinguished name
#
########################################################################

attribute binProperty                   1.3.18.0.2.4.305     bin
attribute binPropertyType               1.3.18.0.2.4.306     cis
attribute cesProperty                   1.3.18.0.2.4.307     ces
attribute cesPropertyType               1.3.18.0.2.4.308     cis
attribute cisProperty                   1.3.18.0.2.4.309     cis
attribute cisPropertyType               1.3.18.0.2.4.310     cis
attribute propertyType                  1.3.18.0.2.4.320     cis
attribute systemName                    1.3.18.0.2.4.329     cis
attribute db2nodeName                   1.3.18.0.2.4.419     cis
attribute db2nodeAlias                  1.3.18.0.2.4.420     cis
attribute db2instanceName               1.3.18.0.2.4.428     cis
attribute db2Type                       1.3.18.0.2.4.418     cis
attribute db2databaseName               1.3.18.0.2.4.421     cis
```

```
 attribute db2databaseAlias                1.3.18.0.2.4.422     cis
 attribute db2nodePtr                      1.3.18.0.2.4.423     dn
 attribute db2gwPtr                        1.3.18.0.2.4.424     dn
 attribute db2additionalParameters         1.3.18.0.2.4.426     cis
 attribute db2ARLibrary                    1.3.18.0.2.4.427     cis
 attribute db2authenticationLocation       1.3.18.0.2.4.425     cis
 attribute db2databaseRelease              1.3.18.0.2.4.429     cis
 attribute DCEPrincipalName                1.3.18.0.2.4.443     cis
```

다음과 같이 DB2 오브젝트를 slapd.user_oc.conf 파일에 추가해야 합니다.

```
#################################################################################
#
# IBM DB2 Universal Database
# Object Class Definitions
#
#################################################################################

objectclass eProperty
        oid 1.3.18.0.2.6.90
        requires
                objectClass
        allows
                cn,
                propertyType,
                binProperty,
                binPropertyType,
                cesProperty,
                cesPropertyType,
                cisProperty,
cisPropertyType

objectclass eApplicationSystem
        oid 1.3.18.0.2.6.84
        requires
                objectClass,
                systemName


objectclass DB2Node
        oid 1.3.18.0.2.6.116
        requires
                objectClass,
db2nodeName
        allows
                db2nodeAlias,
                host,
                db2instanceName,
                db2Type,
                description,
                protocolInformation

objectclass DB2Database
        oid 1.3.18.0.2.6.117
        requires
                objectClass,
                db2databaseName,
db2nodePtr
        allows
```

```
                    db2databaseAlias,
                    description,
                    db2gwPtr,
                    db2additionalParameters,
                    db2authenticationLocation,
                    DCEPrincipalName,
                    db2databaseRelease,
db2ARLibrary
```

DB2 스키마 정의를 추가한 후에는 모든 변경사항이 적용되도록 디렉토리 서버를 재시작해야 합니다.

관련 참조:

- 391 페이지의 『DB2에서 사용하는 LDAP 오브젝트 클래스 및 속성』

IBM SecureWay Directory Server의 디렉토리 스키마 확장

IBM® SecureWay® LDAP Directory Server 버전 3.1 이상을 사용 중인 경우, DB2® Universal Database(DB2 UDB) 버전 8.2 이전에 필요한 모든 오브젝트 클래스 및 속성이 기본 스키마에 포함됩니다. 버전 8.2에서 도입된 새 DB2 UDB 속성으로 기본 스키마를 확장하려면 다음을 실행하십시오.

```
ldapmodify -c -h <machine_name>:389 -D <dn> -w <password> -f altgwnode.ldif
```

다음은 `altgwnode.ldif` 파일에 대한 설명입니다.

```
dn:                 cn=schema
changetype:         modify
add:                attributetypes
attributetypes:     (
 1.3.18.0.2.4.3092
 NAME 'db2altgwPtr'
 DESC 'DN pointer to DB2 alternate gateway (node) object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:                ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3092
 DBNAME ('db2altgwPtr' 'db2altgwPtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)

dn:                 cn=schema
changetype:         modify
add:                attributetypes
attributetypes:     (
 1.3.18.0.2.4.3093
 NAME 'db2altnodePtr'
 DESC 'DN pointer to DB2 alternate node object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:                ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3093
 DBNAME ('db2altnodePtr' 'db2altnodePtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)

dn:             cn=schema
changetype:     modify
replace:        objectclasses
objectclasses: (
 1.3.18.0.2.6.117
 NAME 'DB2Database'
 DESC 'DB2 database'
 SUP cimSetting
 MUST ( db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName ) )
```

altgwnode.ldif 및 altgwnode.readme 파일은 다음 URL에서 찾을 수 있습니다. ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

DB2 스키마 정의를 추가한 후에는 모든 변경사항이 적용되도록 디렉토리 서버를 재시작해야 합니다.

관련 개념:

- 381 페이지의 『DB2 오브젝트 클래스 및 속성이 있는 LDAP 디렉토리 스키마 확장』
- 388 페이지의 『Sun One Directory Server의 디렉토리 스키마 확장』

태스크 관련:

- 381 페이지의 『Active Directory의 디렉토리 스키마 확장』

Sun One Directory Server의 디렉토리 스키마 확장

Sun One Directory Server는 Netscape 또는 iPlanet 디렉토리 서버라고도 합니다.

Sun One Directory Server를 사용자의 환경에서 작동하게 하려면, 60ibmdb2.ldif 파일을 다음 디렉토리에 추가하십시오.

Windows®에서 iPlanet을 C:\iPlanet\Servers에 설치한 경우, 위의 파일을 .\sldap-<machine_name>\config\schema에 추가하십시오.

UNIX®에서 iPlanet을 /usr/iplanet/servers에 설치한 경우, 위의 파일을 ./slapd-<machine_name>/config/schema에 추가하십시오.

다음은 파일의 내용입니다.

```
#################################################################################
# IBM&regtm; DB2&regtm; Universal Database
#################################################################################
dn: cn=schema
#################################################################################
# Attribute Definitions (Before V8.2)
#################################################################################
attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.5               X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'                   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
#################################################################################
# Attribute Definitions (V8.2)
#################################################################################
attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'             SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
```

```
########################################################################
# Object Class Definitions
# DB2Database for V8.2 has the above two new optional attributes.
########################################################################
objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'          SUP top STRUCTURAL
 MAY ( cn $ propertyType $ binProperty $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty $ cisPropertyType )
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem' SUP top STRUCTURAL MUST systemName
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'            SUP top STRUCTURAL MUST db2nodeName
 MAY ( db2instanceName $ db2nodeAlias $ db2Type $ description $ host $ protocolInformation )
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'        SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName $ description )
 X-ORIGIN 'IBM DB2' )
```

60ibmdb2.ldif 및 60ibmdb2.readme 파일은 다음 URL에서 찾을 수 있습니다. ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

DB2 스키마 정의를 추가한 후에는 모든 변경사항이 적용되도록 디렉토리 서버를 재시작해야 합니다.

관련 개념:

- 381 페이지의 『DB2 오브젝트 클래스 및 속성이 있는 LDAP 디렉토리 스키마 확장』
- 386 페이지의 『IBM SecureWay Directory Server의 디렉토리 스키마 확장』

태스크 관련:

- 381 페이지의 『Active Directory의 디렉토리 스키마 확장』

DB2에서 사용하는 LDAP 오브젝트 클래스 및 속성

다음 표에서는 DB2에서 사용되는 오브젝트 클래스를 설명합니다.

표 23. cimManagedElement

클래스	**cimManagedElement**
Active Directory LDAP 표시장치 이름	적용할 수 없음
Active Directory 공통 이름(cn)	적용할 수 없음
설명	많은 시스템 관리 오브젝트 클래스의 기본 클래스를 IBM 스키마에서 제공합니다.
SubClassOf	top
필수 속성(s)	
선택적 속성(s)	설명
유형	abstract
OID(오브젝트 ID)	1.3.18.0.2.6.132
GUID(전역 고유 ID)	b3afd63f-5c5b-11d3-b818-002035559151

표 24. cimSetting

클래스	**cimSetting**
Active Directory LDAP 표시장치 이름	적용할 수 없음
Active Directory 공통 이름(cn)	적용할 수 없음
설명	구성 및 설정의 기본 클래스를 IBM 스키마에서 제공합니다.
SubClassOf	cimManagedElement
필수 속성	
선택적 속성	settingID
유형	abstract
OID(오브젝트 ID)	1.3.18.0.2.6.131
GUID(전역 고유 ID)	b3afd64d-5c5b-11d3-b818-002035559151

표 25. *eProperty*

클래스	**eProperty**
Active Directory LDAP 표시장치 이름	ibm-eProperty
Active Directory 공통 이름(cn)	ibm-eProperty
설명	사용자 환경설정 등록 정보에 대한 응용프로그램 고유의 설정을 지정하는 데 사용됩니다.
SubClassOf	cimSetting
필수 속성	
선택적 속성	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
유형	structural
OID(오브젝트 ID)	1.3.18.0.2.6.90
GUID(전역 고유 ID)	b3afd69c-5c5b-11d3-b818-002035559151

표 26. *DB2Node*

클래스	**DB2Node**
Active Directory LDAP 표시장치 이름	ibm-db2Node
Active Directory 공통 이름(cn)	ibm-db2Node
설명	DB2 서버 설명
SubClassOf	eSap / ServiceConnectionPoint
필수 속성(s)	db2nodeName
선택적 속성(s)	db2nodeAlias db2instanceName db2Type host / dNSHostName(주 2 참조) protocolInformation/ServiceBindingInformation
유형	structural
OID(오브젝트 ID)	1.3.18.0.2.6.116
GUID(전역 고유 ID)	b3afd65a-5c5b-11d3-b818-002035559151

표 26. DB2Node (계속)

클래스	**DB2Node**
특수 주	1. *DB2Node* 클래스는 IBM SecureWay 디렉토리 아래의 *eSap* 오브젝트 클래스와 Microsoft Active Directory 아래의 *ServiceConnectionPoint* 오브젝트에서 파생됩니다. 2. *host*는 IBM SecureWay 환경에서 사용됩니다. *dNSHostName* 속성은 Microsoft Active Directory 아래에서 사용됩니다. 3. *protocolInformation*은 IBM SecureWay 환경 아래에서만 사용됩니다. Microsoft Active Directory의 경우, ServiceConnectionPoint 클래스에서 상속된 *ServiceBindingInformation* 속성은 프로토콜 정보를 포함하기 위해 사용됩니다.

DB2Node 오브젝트에 있는 *protocolInformation*(IBM SecureWay Directory에서) 또는 *ServiceBindingInformation*(Microsoft Active Directory에서) 속성에는 DB2 데이터베이스 서버에 바인드할 통신 프로토콜 정보가 들어 있습니다. 지원되는 네트워크 프로토콜을 설명하는 토큰으로 구성됩니다. 각 토큰은 세미콜론으로 구분됩니다. 토큰 사이에는 스페이스를 두지 않습니다. 별표(*)는 선택적 매개변수를 지정하기 위해 사용될 수 있습니다.

TCP/IP에 대한 토큰은 다음과 같습니다.

- 『TCPIP』
- 서버 호스트 이름 또는 IP 주소
- 서비스 이름(svcename) 또는 포트 번호(예: 50000)
- (선택적) 보안(『NONE』 또는 『SOCKS』)

APPN에 대한 토큰은 다음과 같습니다.

- 『APPN』
- 네트워크 ID
- 상대 LU
- 트랜잭션 프로그램(TP) 이름(지원 응용프로그램 TP만이 서비스 TP(HEX의 TP)를 지원하지 않음)
- 모드
- 보안(『NONE』, 『SAME』 또는 『PROGRAM』)
- (선택적) LAN 어댑터 주소
- (선택적) 암호 변경 LU

주: Windows용 DB2 UDB 클라이언트에서, APPN 정보가 로컬 SNA 스택에서 구성되지 않은 경우 그리고 LAN 어댑터 주소 및 선택적 암호 변경 LU가 LDAP에 있는 경우 DB2 UDB 클라이언트는 이 정보를 사용하여 스택 구성 방법을 아는 경우 SNA 스택을 구성합니다.

NetBIOS에 대한 토큰은 다음과 같습니다.

- 『NETBIOS』
- 서버 NetBIOS 워크스테이션 이름

Named Pipe에 대한 토큰은 다음과 같습니다.

- 『NPIPE』
- 서버의 컴퓨터 이름
- 서버의 인스턴스 이름

표 27. DB2Database

클래스	**DB2Database**
Active Directory LDAP 표시장치 이름	ibm-db2Database
Active Directory 공통 이름(cn)	ibm-db2Database
설명	DB2 데이터베이스 설명
SubClassOf	top
필수 속성	db2databaseName db2nodePtr
선택적 속성	db2databaseAlias db2additionalParameters db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName db2altgwPtr db2altnodePtr
유형	structural
OID(오브젝트 ID)	1.3.18.0.2.6.117
GUID(전역 고유 ID)	b3afd659-5c5b-11d3-b818-002035559151

표 28. db2additionalParameters

속성	**db2additionalParameters**
Active Directory LDAP 표시장치 이름	ibm-db2AdditionalParameters

표 28. *db2additionalParameters (계속)*

속성	**db2additionalParameters**
Active Directory 공통 이름(cn)	ibm-db2AdditionalParameters
설명	호스트 데이터베이스 서버에 연결할 때 사용되는 모든 추가 매개변수 포함
구문	대소문자 비구분 문자열
최대 길이	1024
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.426
GUID(전역 고유 ID)	b3afd315-5c5b-11d3-b818-002035559151

표 29. *db2authenticationLocation*

속성	**db2authenticationLocation**
Active Directory LDAP 표시장치 이름	ibm-db2AuthenticationLocation
Active Directory 공통 이름(cn)	ibm-db2AuthenticationLocation
설명	인증이 발생하는 곳 지정
구문	대소문자 비구분 문자열
최대 길이	64
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.425
GUID(전역 고유 ID)	b3afd317-5c5b-11d3-b818-002035559151
주	올바른 값: CLIENT, SERVER, DCS, DCE, KERBEROS, SVRENCRYPT 또는 DCSENCRYPT

표 30. *db2ARLibrary*

속성	**db2ARLibrary**
Active Directory LDAP 표시장치 이름	ibm-db2ARLibrary
Active Directory 공통 이름(cn)	ibm-db2ARLibrary
설명	응용프로그램 리퀘스터 라이브러리의 이름
구문	대소문자 비구분 문자열
최대 길이	256
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.427
GUID(전역 고유 ID)	b3afd316-5c5b-11d3-b818-002035559151

표 31. *db2databaseAlias*

속성	**db2databaseAlias**
Active Directory LDAP 표시장치 이름	ibm-db2DatabaseAlias
Active Directory 공통 이름(cn)	ibm-db2DatabaseAlias
설명	데이터베이스 별명 이름(s)
구문	대소문자 비구분 문자열
최대 길이	1024

표 31. *db2databaseAlias (계속)*

속성	**db2databaseAlias**
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.422
GUID(전역 고유 ID)	b3afd318-5c5b-11d3-b818-002035559151

표 32. *db2databaseName*

속성	**db2databaseName**
Active Directory LDAP 표시장치 이름	ibm-db2DatabaseName
Active Directory 공통 이름(cn)	ibm-db2DatabaseName
설명	데이터베이스 이름
구문	대소문자 비구분 문자열
최대 길이	1024
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.421
GUID(전역 고유 ID)	b3afd319-5c5b-11d3-b818-002035559151

표 33. *db2databaseRelease*

속성	**db2databaseRelease**
Active Directory LDAP 표시장치 이름	ibm-db2DatabaseRelease
Active Directory 공통 이름(cn)	ibm-db2DatabaseRelease
설명	데이터베이스 릴리스 번호
구문	대소문자 비구분 문자열
최대 길이	64
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.429
GUID(전역 고유 ID)	b3afd31a-5c5b-11d3-b818-002035559151

표 34. *db2nodeAlias*

속성	**db2nodeAlias**
Active Directory LDAP 표시장치 이름	ibm-db2NodeAlias
Active Directory 공통 이름(cn)	ibm-db2NodeAlias
설명	노드 별명 이름(s)
구문	대소문자 비구분 문자열
최대 길이	1024
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.420
GUID(전역 고유 ID)	b3afd31d-5c5b-11d3-b818-002035559151

표 35. *db2nodeName*

속성	**db2nodeName**
Active Directory LDAP 표시장치 이름	ibm-db2NodeName

표 35. *db2nodeName (계속)*

속성	**db2nodeName**
Active Directory 공통 이름(cn)	ibm-db2NodeName
설명	노드 이름
구문	대소문자 비구분 문자열
최대 길이	64
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.419
GUID(전역 고유 ID)	b3afd31e-5c5b-11d3-b818-002035559151

표 36. *db2nodePtr*

속성	**db2nodePtr**
Active Directory LDAP 표시장치 이름	ibm-db2NodePtr
Active Directory 공통 이름(cn)	ibm-db2NodePtr
설명	데이터베이스를 소유하는 데이터베이스 서버를 나타내는 노드(DB2Node) 오브젝트에 대한 포인터
구문	식별 이름
최대 길이	1000
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.423
GUID(전역 고유 ID)	b3afd31f-5c5b-11d3-b818-002035559151
특수 주	이 관계는 클라이언트가 데이터베이스에 연결하기 위해 프로토콜 통신 정보를 검색할 수 있도록 합니다.

표 37. *db2altnodePtr*

속성	**db2altnodePtr**
Active Directory LDAP 표시장치 이름	ibm-db2AltNodePtr
Active Directory 공통 이름(cn)	ibm-db2AltNodePtr
설명	대체 데이터베이스 서버를 나타내는 노드(DB2Node) 오브젝트에 대한 포인터
구문	식별 이름
최대 길이	1000
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.3093
GUID(전역 고유 ID)	5694e266-2059-4e32-971e-0778909e0e72

표 38. *db2gwPtr*

속성	**db2gwPtr**
Active Directory LDAP 표시장치 이름	ibm-db2GwPtr
Active Directory 공통 이름(cn)	ibm-db2GwPtr
설명	게이트웨이 서버를 나타내며 데이터베이스가 액세스될 수 있는 노드 오브젝트에 대한 포인터
구문	식별 이름

표 38. *db2gwPtr (계속)*

속성	**db2gwPtr**
최대 길이	1000
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.424
GUID(전역 고유 ID)	b3afd31b-5c5b-11d3-b818-002035559151

표 39. *db2altgwPtr*

속성	**db2altgwPtr**
Active Directory LDAP 표시장치 이름	ibm-db2AltGwPtr
Active Directory 공통 이름(cn)	ibm-db2AltGwPtr
설명	대체 게이트웨이 서버를 나타내는 노드 오브젝트에 대한 포인터
구문	식별 이름
최대 길이	1000
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.3092
GUID(전역 고유 ID)	70ab425d-65cc-4d7f-91d8-084888b3a6db

표 40. *db2instanceName*

속성	**db2instanceName**
Active Directory LDAP 표시장치 이름	ibm-db2InstanceName
Active Directory 공통 이름(cn)	ibm-db2InstanceName
설명	데이터베이스 서버 인스턴스의 이름
구문	대소문자 비구분 문자열
최대 길이	256
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.428
GUID(전역 고유 ID)	b3afd31c-5c5b-11d3-b818-002035559151

표 41. *db2Type*

속성	**db2Type**
Active Directory LDAP 표시장치 이름	ibm-db2Type
Active Directory 공통 이름(cn)	ibm-db2Type
설명	데이터베이스 서버의 유형
구문	대소문자 비구분 문자열
최대 길이	64
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.418
GUID(전역 고유 ID)	b3afd320-5c5b-11d3-b818-002035559151
주	데이터베이스 서버의 유효한 유형: SERVER, MPP, DCS

표 42. *DCEPrincipalName*

속성	**DCEPrincipalName**
Active Directory LDAP 표시장치 이름	ibm-DCEPrincipalName
Active Directory 공통 이름(cn)	ibm-DCEPrincipalName
설명	DCE 핵심부 이름
구문	대소문자 비구분 문자열
최대 길이	2048
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.443
GUID(전역 고유 ID)	b3afd32d-5c5b-11d3-b818-002035559151

표 43. *cesProperty*

속성	**cesProperty**
Active Directory LDAP 표시장치 이름	ibm-cesProperty
Active Directory 공통 이름(cn)	ibm-cesProperty
설명	이 속성의 값은 응용프로그램 고유의 환경설정 구성 매개변수를 제공하기 위해 사용될 수 있습니다. 예를 들어, 값은 XML 형식화 데이터를 포함할 수 있습니다. 이 모든 속성 값은 cesPropertyType 속성 값에서 동종이어야 합니다.
구문	대소문자 비구분 문자열
최대 길이	32700
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.307
GUID(전역 고유 ID)	b3afd2d5-5c5b-11d3-b818-002035559151

표 44. *cesPropertyType*

속성	**cesPropertyType**
Active Directory LDAP 표시장치 이름	ibm-cesPropertyType
Active Directory 공통 이름(cn)	ibm-cesPropertyType
설명	이 속성의 값은 구문, 의미 또는 cesProperty 속성의 모든 값의 다른 특성을 설명하기 위해 사용될 수 있습니다. 예를 들어, 『XML』의 값은 cesProperty 속성의 모든 값이 XML 구문으로서 코드화됨을 나타내기 위해 사용될 수 있습니다.
구문	대소문자 비구분 문자열
최대 길이	128
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.308
GUID(전역 고유 ID)	b3afd2d6-5c5b-11d3-b818-002035559151

표 45. *cisProperty*

속성	**cisProperty**
Active Directory LDAP 표시장치 이름	ibm-cisProperty
Active Directory 공통 이름(cn)	ibm-cisProperty

표 45. *cisProperty (계속)*

속성	**cisProperty**
설명	이 속성의 값은 응용프로그램 고유의 환경설정 구성 매개변수를 제공하기 위해 사용될 수 있습니다. 예를 들어, 값은 INI 파일을 포함할 수 있습니다. 이 모든 속성 값은 cisPropertyType 속성 값에서 동종이어야 합니다.
구문	대소문자 비구분 문자열
최대 길이	32700
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.309
GUID(전역 고유 ID)	b3afd2e0-5c5b-11d3-b818-002035559151

표 46. *cisPropertyType*

속성	**cisPropertyType**
Active Directory LDAP 표시장치 이름	ibm-cisPropertyType
Active Directory 공통 이름(cn)	ibm-cisPropertyType
설명	이 속성의 값은 구문, 의미 또는 cisProperty 속성의 모든 값의 다른 특성을 설명하기 위해 사용될 수 있습니다. 예를 들어, 『INI 파일』의 값은 cisProperty 속성의 모든 값이 INI 파일임을 나타내기 위해 사용될 수 있습니다.
구문	대소문자 비구분 문자열
최대 길이	128
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.310
GUID(전역 고유 ID)	b3afd2e1-5c5b-11d3-b818-002035559151

표 47. *binProperty*

속성	**binProperty**
Active Directory LDAP 표시장치 이름	ibm-binProperty
Active Directory 공통 이름(cn)	ibm-binProperty
설명	이 속성의 값은 응용프로그램 고유의 환경설정 구성 매개변수를 제공하기 위해 사용될 수 있습니다. 예를 들어, 값에는 2진 암호화 Lotus 123 등록 정보 세트가 들어 있을 수 있습니다. 이 속성의 모든 값은 해당 binPropertyType 속성 값에서 동종이어야 합니다.
구문	2진
최대 길이	250000
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.305
GUID(전역 고유 ID)	b3afd2ba-5c5b-11d3-b818-002035559151

표 48. *binPropertyType*

속성	**binPropertyType**
Active Directory LDAP 표시장치 이름	ibm-binPropertyType
Active Directory 공통 이름(cn)	ibm-binPropertyType

표 48. *binPropertyType (계속)*

속성	**binPropertyType**
설명	이 속성의 값은 구문, 의미 또는 binProperty 속성의 모든 값의 다른 특성을 설명하기 위해 사용될 수 있습니다. 예를 들어, 『Lotus 123』의 값은 binProperty 속성의 모든 값이 2진 코드화 Lotus 123 등록 정보임을 나타내기 위해 사용될 수 있습니다.
구문	대소문자 비구분 문자열
최대 길이	128
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.306
GUID(전역 고유 ID)	b3afd2bb-5c5b-11d3-b818-002035559151

표 49. *PropertyType*

속성	**PropertyType**
Active Directory LDAP 표시장치 이름	ibm-propertyType
Active Directory 공통 이름(cn)	ibm-propertyType
설명	이 속성의 값은 eProperty 오브젝트의 의미 특성을 설명합니다.
구문	대소문자 비구분 문자열
최대 길이	128
다중 값	다중 값
OID(오브젝트 ID)	1.3.18.0.2.4.320
GUID(전역 고유 ID)	b3afd4ed-5c5b-11d3-b818-002035559151

표 50. *settingID*

속성	**settingID**
Active Directory LDAP 표시장치 이름	적용할 수 없음
Active Directory 공통 이름(cn)	적용할 수 없음
설명	eProperty와 같은 오브젝트 항목에서 파생한 cimSetting를 식별하기 위해 사용될 수 있는 이름 지정 속성
구문	대소문자 비구분 문자열
최대 길이	256
다중 값	단일 값
OID(오브젝트 ID)	1.3.18.0.2.4.325
GUID(전역 고유 ID)	b3afd596-5c5b-11d3-b818-002035559151

관련 개념:

- 359 페이지의 『LDAP(Lightweight Directory Access Protocol) 입문』

부록 D. 다중 데이터베이스 파티션에 대한 명령 발행

파티션된 데이터베이스 환경에서 명령 발행

파티션된 데이터베이스 시스템에서 인스턴스의 머신 또는 데이터베이스 파티션 서버(노드)에서 실행되도록 명령을 발행할 수 있습니다. 이 경우, **rah** 명령 또는 **db2_all** 명령을 사용하십시오. **rah** 명령을 사용하면 인스턴스의 머신에서 실행되도록 명령을 발행할 수 있습니다. 인스턴스의 데이터베이스 파티션 서버에서 명령이 실행되게 하려면, **db2_all** 명령을 실행하십시오. 다음 절에서는 이들 명령의 개요를 제공합니다. 제공되는 정보는 파티션된 데이터베이스 시스템에만 적용됩니다.

주:

1. UNIX® 기반 플랫폼에서 로그인 쉘이 Korn 쉘 또는 다른 쉘이 될 수 있으나, 서로 다른 쉘은 특수 문자를 포함하는 명령을 처리하는 방법에서 차이를 나타냅니다.
2. Windows NT에서 **rah** 명령 또는 **db2_all** 명령을 실행하려면 관리자 그룹의 구성원이 되는 사용자 어카운트로 로그온하십시오.

명령의 영역을 판별하려면, *Command Reference*를 참조하십시오. 이 책에서는 명령이 단일 데이터베이스 파티션 서버에서 실행되는지 또는 이들 전체에서 실행되는지를 보여줍니다. 명령이 하나의 데이터베이스 파티션 서버에서 실행되고 있을 때 명령이 이들 전체에서 실행되게 하려면 **db2_all**을 사용하십시오. 예외는 머신의 모든 논리 노드(데이터베이스 파티션 서버)에서 실행되는 **db2trc** 명령입니다. 모든 머신의 모든 논리 노드에서 **db2trc**를 실행하려면 **rah**를 사용하십시오.

관련 개념:

- 403 페이지의 『rah 및 db2_all 명령 개요』
- 405 페이지의 『rah 및 db2_all 명령 지정』

관련 참조:

- 404 페이지의 『rah 및 db2_all 명령 설명』

rah 및 db2_all 명령 개요

하나의 데이터베이스 파티션 서버에서 또다른 서버 이후에 연속적으로 명령을 실행할 수 있고 혹은 명령을 병렬로 실행할 수 있습니다. UNIX® 기반 플랫폼에서 명령을 병렬로 실행할 경우, 출력을 버퍼로 보내고 화면에 표시하기 위해 수집할 수 있으며(디폴트 활동), 명령이 발행된 머신에 출력을 표시할 수 있습니다. Windows NT에서 명령을 병렬로 실행하면 출력은 명령이 발행된 머신에 표시됩니다.

rah 명령을 사용하려면, 다음을 입력하십시오.

```
rah command
```

db2_all 명령을 사용하려면, 다음을 입력하십시오.

```
db2_all command
```

rah 구문에 대한 도움말을 보려면, 다음을 입력하십시오.

```
rah "?"
```

명령은 다중 명령을 시퀀스대로 실행하는 것을 포함하여 대화식 프롬프트에서 입력할 수 있는 것입니다. UNIX 기반 플랫폼에서 세미콜론(;)을 사용하여 다중 명령을 구분할 수 있습니다. Windows NT에서 앰퍼샌드(&)를 사용하여 다중 명령을 구분하십시오. 마지막 명령 다음에는 분리문자를 사용하지 않도록 하십시오.

다음 예에서는 노드 구성 파일에 지정된 모든 데이터베이스 파티션의 데이터베이스 구성을 변경하기 위해 **db2_all** 명령을 사용하는 방법을 보여줍니다. 다음과 같이 문자가 큰따옴표로 묶여 있으므로, 요청은 동시에 실행됩니다.

```
db2_all ";UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

관련 개념:

- 403 페이지의 『파티션된 데이터베이스 환경에서 명령 발행』
- 405 페이지의 『rah 및 db2_all 명령 지정』

관련 참조:

- 404 페이지의 『rah 및 db2_all 명령 설명』

rah 및 db2_all 명령 설명

다음 명령을 사용할 수 있습니다.

명령	설명
rah	모든 머신에서 명령을 실행합니다.
db2_all	지정한 모든 데이터베이스 파티션 서버에서 명령을 실행합니다.
db2_kill	다중 데이터베이스 파티션 서버에서 실행 중인 프로세스를 즉시 중지시키고, 모든 데이터베이스 파티션 서버의 자원을 모두 정리합니다. 이 명령은 데이터베이스의 불일치를 나타냅니다. IBM 서비스의 지시가 있는 경우를 제외하고, 이 명령을 발행하지 *마십시오*.
db2_call_stack	UNIX 기반 플랫폼에서 모든 데이터베이스 파티션 서버에서 실행 중인 프로세스가 syslog에 call traceback을 작성하게 합니다.

Windows NT에서 모든 데이터베이스 파티션 서버에서 실행 중인 프로세스가 인스턴스 디렉토리의 P*xxxx*.*nnn* 파일에 call traceback을 작성하는데, 여기서, P*xxxx*는 프로세스 ID이고 *nnn*은 노드 번호입니다.

UNIX 기반 플랫폼에서 이들 명령은 다음과 같이 내재된 특정 설정이 **rah**를 실행합니다.

- 모든 머신에서 병렬 실행
- 각각 /tmp/$USER/db2_kill과 /tmp/$USER/db2_call_stack에서의 버퍼 명령 출력

Windows NT에서 이들 명령을 사용하면 모든 머신에서 **rah**가 병렬로 실행됩니다.

관련 개념:

- 403 페이지의『rah 및 db2_all 명령 개요』
- 405 페이지의『rah 및 db2_all 명령 지정』
- 407 페이지의『UNIX 기반 플랫폼에서 명령 병렬 실행』

rah 및 db2_all 명령 지정

다음 방법으로 명령을 지정할 수 있습니다.

- 명령행에서 매개변수
- 매개변수를 지정하지 않을 경우 프롬프트에 응답

명령에 다음 특수 문자가 들어 있는 경우, 프롬프트 방법을 사용해야 합니다.

```
| & ; < > ( )  { } [ ] unsubstituted $
```

명령행에 매개변수로 명령을 지정할 때 명령이 위에 나열된 특수 문자를 포함하면, 문자를 큰따옴표로 묶어야 합니다.

주: UNIX® 기반 플랫폼에서 명령은 프롬프트에 입력한 것처럼 명령 실행기록에 추가해야 됩니다.

명령의 모든 특수 문자는 정상적으로 입력할 수 있습니다(\를 제외하고 따옴표로 묶음). 명령에 \를 삽입하는 경우, 반드시 두 개의 백슬래시(\\)로 입력해야 합니다.

주: UNIX 기반 플랫폼에서 Korn 쉘을 사용하고 있지 않으면 명령의 모든 특수 문자가 정상적으로 입력될 수 있습니다(" \ 대체되지 않은 $ 및 작은따옴표(')의 경우를 제외하고 따옴표로 묶이지 않은 상태로). 명령에 이들 특수 문자 중 하나를 넣을 경우, 반드시 세 개의 백슬래시(\\\)를 앞에 입력해야 합니다. 예를 들어, 명령에 \ 하나를 넣으려면, 네 개의 백슬래시(\\\\)를 입력해야 합니다.

명령에 큰따옴표(")를 사용하려면 \\\"과 같이 세 개의 백슬래시를 앞에 입력해야 합니다.

주:

1. UNIX 기반 플랫폼에서 명령 쉘이 작은따옴표가 붙은 문자열 내에 작은따옴표를 입력하는 방법을 제공하지 않는 한, 명령에 작은따옴표(')를 포함시킬 수 없습니다.
2. Windows NT에서 명령 창이 작은따옴표가 붙은 문자열 내에 작은따옴표를 입력하는 방법을 제공하지 않는 한, 명령에 작은따옴표(')를 포함시킬 수 없습니다.

백그라운드의 stdin으로부터 읽기에 논리를 포함하는 임의의 korn-shell 쉘 스크립트를 실행할 때, 프로세스가 터미널에서 중지하지 않고 읽을 수 있는 소스로 stdin의 경로를 명시적으로 재지정해야 합니다(SIGTTIN 메시지). stdin의 경로를 재지정하려면, 다음과 같은 양식으로 스크립트를 실행할 수 있습니다.

```
shell_script </dev/null &
```

제공되는 입력이 없는 경우

비슷한 방법으로, 백그라운드에서 db2_all을 실행할 때는 </dev/null을 항상 지정해야 합니다. 예를 들어, 다음과 같습니다.

```
db2_all ";run_this_command" </dev/null &
```

이렇게 함으로써 stdin의 경로를 재지정할 수 있고 터미널에서 중지되는 것을 방지할 수 있습니다.

사용자가 리모트 명령으로부터의 출력에 무관할 때, 이 메소드의 대체는 다음과 같이 db2_all 접두부의 『daemonize』 옵션을 사용하는 것입니다.

```
db2_all ";daemonize_this_command" &
```

관련 개념:

- 407 페이지의 『UNIX 기반 플랫폼에서 명령 병렬 실행』
- 409 페이지의 『추가 rah 정보(Solaris 및 AIX 전용)』

태스크 관련:

- 415 페이지의 『Windows NT에서 rah에 대한 디폴트 환경 프로파일 설정』

관련 참조:

- 404 페이지의 『rah 및 db2_all 명령 설명』
- 409 페이지의 『rah 명령 접두부 시퀀스』
- 413 페이지의 『rah 명령 제어』

UNIX 기반 플랫폼에서 명령 병렬 실행

주: 이 절의 정보는 UNIX® 기반 플랫폼에만 적용됩니다.

디폴트로, 명령은 각 머신에서 순차적으로 실행되지만 특정 접두부 시퀀스를 가지도록 명령에 접두부를 추가함으로써, 백그라운드 rshell을 사용하여 명령이 병렬로 실행되도록 지정할 수 있습니다. rshell이 백그라운드에서 실행되고 각 명령이 리모트 머신의 버퍼 파일에 출력을 추가하는 경우, 이 프로세스는 다음 두 단계로 출력을 검색합니다.

1. 리모트 명령이 완료된 후
2. 일부 프로세스가 실행되고 있는 경우, 이 프로세스가 완료된 후에 발생할 rshell 종료 이후

버퍼 파일 이름은 디폴트로 /tmp/$USER/rahout이나, 환경 변수 $RAHBUFDIR/$RAHBUFNAME로 버퍼 파일 이름을 지정할 수 있습니다.

명령이 동시에 실행되도록 지정할 경우, 디폴트로 이 스크립트는 모든 호스트로 보내는 명령에 접두부로 추가 명령을 첨부하여 $RAHBUFDIR 및 $RAHBUFNAME을 버퍼 파일에 사용할 수 있는지 검사합니다. 그런 다음, $RAHBUFDIR를 작성합니다. 이 과정이 수행되지 않게 하려면, 환경 변수RAHCHECKBUF=no를 익스포트하십시오. 디렉토리가 존재하고 사용 가능한 지 알고 있는 경우, 이 방법을 통해 시간을 절약할 수 있습니다.

rah를 사용하여 다중 머신에서 명령을 동시에 실행하기 전에 다음을 확인하십시오.

- 디렉토리 /tmp/$USER이 각 머신의 ID에 대해 있는지 확인하십시오. 아직 없는 경우, 디렉토리를 작성하려면 다음을 실행하십시오.

  ```
  rah ")mkdir /tmp/$USER"
  ```

- 다음과 같이 .kshrc(Korn 쉘 구문) 또는 .profile에 다음 행을 추가하고 이를 현재 세션에 입력하십시오.

  ```
  export RAHCHECKBUF=no
  ```

- 리모트 명령을 실행하는 각 머신 ID가 **rah**를 실행하는 ID에 대한 .rhosts 파일에 항목을 갖는지와 **rah**를 실행하는 ID가 리모트 명령을 실행하는 각 머신 ID에 대한 .rhosts 파일에 항목을 갖는지 확인하십시오.

관련 개념:

- 409 페이지의 『추가 rah 정보(Solaris 및 AIX 전용)』

태스크 관련:

- 408 페이지의 『UNIX 기반 플랫폼에서 rah 프로세스 모니터링』

관련 참조:

- 409 페이지의 『rah 명령 접두부 시퀀스』

- 416 페이지의『UNIX 기반 플랫폼에서 rah 문제점 판별』

UNIX 기반 플랫폼에서 rah 프로세스 모니터링

프로시저:

주: 이 절의 정보는 UNIX 기반 플랫폼에만 적용됩니다.
아직 실행 중인 리모트 명령이 있거나 버퍼된 출력이 아직 누적되고 있는 경우, rah에 의해 시작된 프로세스는 다음 활동을 모니터합니다.

- 명령이 실행되지 않았음을 표시하는 터미널에 메시지 작성
- 버퍼링된 출력 검색

정보 메시지는 환경 변수 RAHWAITTIME으로 제어되어 일정 간격으로 작성됩니다. 이를 지정하는 방법에 대한 세부사항은 도움말 정보를 참조하십시오. 모든 정보 메시지는 RAHWAITTIME=0을 익스포트하면 전혀 표시되지 않습니다.

1차 모니터링 프로세스는(ps 명령에 의해 표시된 대로) 명령 이름이 **rahwaitfor**이 됩니다. 첫 번째 정보 메시지에서 프로세스의 pid(프로세스 id)를 가리킵니다. 다른 모든 모니터링 프로세스는 **rah** 스크립트를 실행하는 **ksh** 명령(또는 기호 링크 이름)으로 나타납니다. 원하는 경우, 다음 명령으로 모든 모니터링 프로세스를 중지시킬 수 있습니다.

```
kill <pid>
```

여기서, <pid>는 1차 모니터링 프로세서의 프로세스 ID입니다. 신호 수를 지정하지 마십시오. 디폴트값 15를 남겨 두십시오. 이 값은 리모트 명령에는 전혀 영향을 주지 않으나, 버퍼링된 출력의 자동 표시를 방해합니다. **rah**의 단일 실행 중에 다른 시간에 실행되는 다른 모니터링 프로세스가 둘 이상 있을 수 있습니다. 그러나 현재 설정을 중지시키려 할 때에는 더 이상의 프로세스가 시작되지 않습니다.

정기적인 로그인 쉘이 Korn 쉘(예: /bin/ksh)이 아니면 **rah**를 사용할 수 있으나, 다음 특수 문자를 포함하는 명령을 입력하는 방법에는 약간 다른 규칙이 적용될 수 있습니다.

```
"  unsubstituted $ '
```

자세한 정보는 rah "?". 또한, UNIX 기반 환경에서 리모트 명령을 실행하는 ID에서의 로그인 쉘이 Korn 쉘이 아니면 **rah**를 실행하는 ID에서의 로그인 쉘도 Korn 쉘이면 안 됩니다(**rah**는 리모트 ID의 쉘이 로컬 ID를 근거로 하는 Korn 쉘인지에 대해 결정합니다). 이 쉘은 작은따옴표로 묶인 문자열에 대해 대체 또는 특수 처리를 수행해서는 안 됩니다. 있는 그대로 두십시오.

관련 개념:

- 407 페이지의『UNIX 기반 플랫폼에서 명령 병렬 실행』

• 409 페이지의 『추가 rah 정보(Solaris 및 AIX 전용)』

추가 rah 정보(Solaris 및 AIX 전용)

성능을 향상시키기 위해 rah가 대형 시스템에서 tree_logic을 사용하도록 확장되었습니다. 즉, rah는 목록에 있는 노드 수를 검사하여 해당 수가 임계값을 초과할 경우, 목록의 서브세트를 구성하여 이들 노드에 재귀 호출을 보냅니다. 이들 노드에서 순환적으로 호출된 rah는 목록에 있는 모든 노드에 명령을 보내는 표준 논리(이제 ″leaf-of-tree″ 논리)를 따라갈 만큼 목록이 작아질 때까지는 같은 논리를 따라갑니다. 임계값은 환경 변수 RAHTREETHRESH에 의해 지정될 수 있거나 디폴트값으로 15가 됩니다.

물리 노드당 다중 논리 노드 시스템의 경우, db2_all은 특정 물리 노드에 재귀 호출을 보내도록 하며, 이 노드는 같은 물리 노드에 있는 다른 논리 노드에 rsh하여 물리 노드 간 전송도 감소시킵니다(이 점은 db2_all에만 적용되며 rah에는 적용되지 않는데, 그 이유는 rah가 항상 구별된 물리 노드에만 보내기 때문입니다).

관련 개념:

• 407 페이지의 『UNIX 기반 플랫폼에서 명령 병렬 실행』

태스크 관련:

• 408 페이지의 『UNIX 기반 플랫폼에서 rah 프로세스 모니터링』

rah 명령 접두부 시퀀스

접두부 시퀀스는 한 개 이상의 특수 문자로 되어 있습니다. 공백을 개입시키지 말고 명령 문자 바로 앞에 한 개 이상의 접두부 시퀀스를 입력하십시오. 둘 이상의 시퀀스를 지정하기 위해 임의의 시퀀스로 입력을 수행할 수 있으나, 다중 문자 시퀀스 내의 문자는 순서대로 입력하십시오. 접두부 시퀀스를 입력하는 경우, 다음 예에 나오는 것처럼 접두부 시퀀스를 포함하여 전체 명령을 큰따옴표로 묶어야 합니다.

- UNIX 기반 플랫폼에서

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- Windows NT에서

```
rah "||db2 get db cfg for sample"
```

접두부 시퀀스는 다음과 같습니다.

시퀀스	목적
|	백그라운드에서 시퀀스대로 명령을 실행합니다.
|&	일부 프로세스가 여전히 실행 중이더라도 백그라운드에서 명령을 시퀀스대로 실행하고, 모든 리모트 명령이 완료된 이후 명령을 종료합니다. 하위 프로세스(UNIX 기반 플랫폼에서) 또는 백그라운드 프로세스

(Windows에서)가 여전히 실행 중인 경우에는 지연될 수도 있습니다. 이 경우, 명령은 별도의 백그라운드 프로세스를 시작하여 명령 종료 이후에 생성된 리모트 출력을 검색하고, 이를 원래 머신에 다시 작성합니다.

주: UNIX 기반 플랫폼에서 &를 지정하면 많은 **rsh** 명령이 요구되므로 성능 저하를 가져옵니다.

|| 백그라운드에서 명령을 병렬로 실행합니다.

||& 백그라운드에서 명령을 병렬로 실행하고 위의 |& 경우 대한 설명대로 모든 리모트 명령이 완료된 후에 명령을 종료합니다.

주: UNIX 기반 플랫폼에서 &를 지정하면 많은 **rsh** 명령이 요구되므로 성능 저하를 가져옵니다.

; 이는 ||&과 같습니다. 이것은 사용할 수 있는 다른 짧은 양식입니다.

주: UNIX 기반 플랫폼에서 ;를 지정하면 많은 **rsh** 명령이 요구되므로 ||에 비해 성능 저하를 가져옵니다.

] 명령 실행 전에 사용자의 프로파일이 dot 실행을 사전에 보류합니다.

주: UNIX 기반 플랫폼에서만 사용 가능합니다.

} 명령을 실행하기 전에 $RAHENV에 이름 지정된 파일(`.kshrc`)의 dot 실행을 사전에 보류합니다.

주: UNIX 기반 플랫폼에서만 사용 가능합니다.

]} 명령을 실행하기 전에 $RAHENV(`.kshrc`)에 이름 지정된 파일 실행 이전에 사용자 프로파일의 dot 실행을 보류합니다.

주: UNIX 기반 플랫폼에서만 사용 가능합니다.

) 사용자의 프로파일 및 $RAHENV에 이름 지정된 파일이 실행되는 것을 막습니다.

주: UNIX 기반 플랫폼에서만 사용 가능합니다.

' 머신으로 명령 호출을 반향합니다.

< 이 시스템을 제외한 모든 시스템으로 보냅니다.

<<-nnn< 데이터베이스 파티션 서버 *nnn*을 제외한 전체(노드 번호 *nnn*을 제외한 `db2nodes.cfg`의 모든 데이터베이스 파티션 서버, 이 테이블에서 최종 접두부 다음 첫 번째 단락 참조)로 보냅니다.

<<+**nnn**< 데이터베이스 파티션 서버 *nnn*(노드 번호가 *nnn*인 db2nodes.cfg의 데이터베이스 파티션 서버, 이 테이블에서 최종 접두부 시퀀스 다음 첫 번째 단락 참조)에만 보냅니다.

(blank character)

stdin, stdout 및 stderr이 모두 닫혀진 채로 백그라운드에서 리모트 명령을 실행합니다. 이 옵션은 백그라운드에서 명령을 실행할 때, 즉 \ 또는 ;를 포함하는 접두부 시퀀스에서만 유효합니다. 이 옵션을 사용하면 명령을 좀더 빨리(리모트 명령이 시작되자마자) 완료할 수 있습니다. **rah** 명령행에 이 접두부 시퀀스를 지정한 경우 명령을 작은따옴표로 묶거나, 큰따옴표로 묶고 앞에 \를 사용하십시오. 예를 들어,

```
rah ';  mydaemon'
```

또는

```
rah ";\  mydaemon"
```

rah 명령은 백그라운드 프로세스로 실행할 경우, 출력이 리턴되기를 기다리지 않습니다.

> <>를 머신 이름으로 대체하십시오.

" () 대신 머신 인덱스로 사용하고, ## 대신 노드 번호를 사용합니다.

주:

1. 머신 인덱스는 데이터베이스 시스템의 머신과 연관된 번호입니다. 다중 논리 노드를 실행하고 있지 않은 경우, 머신에 대한 머신 인덱스는 노드 구성 파일의 해당 머신에 대한 노드 번호에 해당합니다. 다중 논리 노드 환경에서 머신에 대한 머신 인덱스를 구하려면, 다중 논리 노드가 실행되는 머신에 대한 중복된 항목을 계산하지 않도록 하십시오. 예를 들어, MACH1이 두 개의 논리 노드에서 실행되고 MACH2도 두 개의 논리 노드에서 실행되는 경우, 노드 구성 파일에서 MACH3에 대한 노드 번호는 5가 됩니다. 그러나 MACH3에 대한 머신 인덱스는 3이 됩니다.

   Windows NT에서, 노드 구성 파일을 편집하지 마십시오. 머신 인덱스를 구하려면, **db2nlist** 명령을 사용하십시오.
2. "가 지정되면, 머신 목록에서 중복된 사항이 제거되지 않습니다.

<<-nnn< 및 <<+nnn< 접두부 시퀀스를 사용할 경우, *nnn*은 db2nodes.cfg 파일의 *nodenum* 값과 일치하는 1, 2 또는 3자리 파티션 번호가 됩니다.

주: 접두부 시퀀스는 명령의 일부로 인식해야 합니다. 명령의 일부로 접두부 시퀀스를 지정하는 경우, 접두부 시퀀스를 포함하여 전체 명령을 큰따옴표로 묶어야 합니다.

관련 개념:

- 405 페이지의 『rah 및 db2_all 명령 지정』
- 407 페이지의 『UNIX 기반 플랫폼에서 명령 병렬 실행』

관련 참조:

- 404 페이지의 『rah 및 db2_all 명령 설명』

파티션된 환경의 머신 목록 지정

프로시저:

디폴트로, 머신 목록은 노드 구성 파일, db2nodes.cfg로부터 가져옵니다. 다음을 수행하여 파일을 겹쳐쓸 수 있습니다.

- 환경 변수 RAHOSTFILE을 익스포트하거나(UNIX 기반 플랫폼) 이 변수를 설정하여(Windows NT) 머신 목록을 포함하는 파일의 경로 이름 지정
- 환경 변수 RAHOSTLIST를 익스포트하거나(UNIX 기반 플랫폼) 이 변수를 설정하여(Windows NT) 스페이스로 구분된 일련의 이름으로 목록을 명시적으로 지정

  주: 이들 환경 변수가 모두 지정되면 RAHOSTLIST가 우선됩니다.

주: Windows NT에서 노드 구성 파일에 일관되지 않은 내용이 삽입되지 않게 하려면, 이를 수동으로 편집하지 않도록 하십시오. 인스턴스에서 머신 목록을 구하려면 **db2nlist** 명령을 사용하십시오.

태스크 관련:

- 412 페이지의 『파티션된 환경의 머신 목록에서 중복 항목 제거』

파티션된 환경의 머신 목록에서 중복 항목 제거

프로시저:

한 머신의 다중 논리 노드(데이터베이스 파티션 서버)에서 DB2 Universal Database Enterprise Server Edition을 실행 중인 경우, db2nodes.cfg 파일에 해당 머신에 대한 여러 개의 항목이 포함됩니다. 이 상황에서 **rah** 명령에는 사용자가 이 명령을 각 머신에 대해 한번씩만 실행할지 또는 db2nodes.cfg 파일에 나열된 각 논리 노드에 대해 한번씩 실행할지 여부를 지정하십시오. 머신을 지정하려면 **rah** 명령을 사용하십시오. 논리 노드를 지정하려면 **db2_all** 명령을 사용하십시오.

주: UNIX 기반 플랫폼에서 머신을 지정하는 경우, **rah**는 보통 다음 예외 상황을 제외하고 머신 목록에서 중복된 머신을 제거합니다. 논리 노드를 지정할 때, **db2_all** 이 명령의 다음 사항을 사전에 보류할 경우는 예외 상황입니다.

```
export DB2NODE=nnn  (for Korn shell syntax)
```

여기서, *nnn*은 db2nodes.cfg 파일의 해당 행으로부터 가져온 노드 번호이므로, 이 명령의 경로는 원하는 데이터베이스 파티션 서버로 지정됩니다.

논리 노드를 지정할 때에는 <<-nnn<과 <<+nnn< 접두부 시퀀스를 사용하여 하나만을 제외한 모든 논리 노드를 포함하도록 목록을 제한하거나 하나의 데이터베이스 파티션 서버만을 지정할 수 있습니다. 카탈로그 노드에서 먼저 명령을 실행하려 하며, 명령이 완료될 때 같은 명령을 다른 모든 데이터베이스 파티션 서버에서 병렬로 실행 가능할 경우, 이 방법을 사용할 수 있습니다. 이 방법은 보통 **db2 restart database** 명령을 실행할 때 필요합니다. 이를 위해 카탈로그 노드의 노드 번호를 알아야 합니다.

rah 명령을 사용하여 **db2 restart database**를 실행하면 머신 목록에서 중복된 항목이 제거됩니다. 그러나 "접두부"를 지정하는 경우, 접두부를 사용하는 것이 각 머신이 아닌 각 데이터베이스 파티션 서버에 보내는 것으로 간주되므로, 중복된 항목이 제거되지 않습니다.

태스크 관련:

- 412 페이지의『파티션된 환경의 머신 목록 지정』

관련 참조:

- *Command Reference*의『RESTART DATABASE Command』
- 409 페이지의『rah 명령 접두부 시퀀스』

rah 명령 제어

다음 환경 변수를 사용하여 **rah** 명령을 제어할 수 있습니다.

표 51.

이름	의미	디폴트값
$RAHBUFDIR **주:** UNIX 기반 플랫폼에서만 사용 가능합니다.	버퍼 디렉토리	/tmp/$USER
$RAHBUFNAME **주:** UNIX 기반 플랫폼에서만 사용 가능합니다.	버퍼 파일 이름	rahout
$RAHOSTFILE(UNIX 기반 플랫폼); RAHOSTFILE(Windows NT)	호스트 목록이 들어 있는 파일	db2nodes.cfg
$RAHOSTLIST(UNIX 기반 플랫폼); RAHOSTLIST(Windows NT)	문자열로 된 호스트 목록	$RAHOSTFILE에서 얻음

표 51. (계속)

이름	의미	디폴트값
$RAHCHECKBUF **주:** UNIX 기반 플랫폼에서만 사용 가능합니다.	″no″로 설정되면, 검사를 생략합니다.	설정 안됨
$RAHSLEEPTIME (UNIX 기반 플랫폼); RAHSLEEPTIME (Windows NT)	스크립트가 병렬로 실행되는 명령으로부터 초기 출력을 기다리는 시간(초)	**db2_kill**의 경우는 86400초, 기타 모든 경우는 200초
$RAHWAITTIME (UNIX 기반 플랫폼); RAHWAITTIME (Windows NT)	Windows NT에서 리모트 작업이 아직 실행 중인 연속되는 검사 간의 간격(초) UNIX 기반 플랫폼에서 리모트 작업이 아직 실행 중임과 `rah: <pid> 기다리는 중...` 메시지가 나타나는 것을 확인하는 연속되는 검사 간의 간격(초) 모든 플랫폼에서 양의 정수를 지정하십시오. 메시지 출력을 제어하기 위해 사용되는 0으로 시작하는 접두부 값의 경우, RAHWAITTIME=045를 익스포트하십시오. **rah**는 작업 완료 상태를 검출하기 위해 이 검사를 따르지 않으므로 낮은 값을 지정할 필요는 없습니다.	45초
$RAHENV **주:** UNIX 기반 플랫폼에서만 사용 가능합니다.	$RAHDOTFILES=E, K, PE 또는 B이면 실행 가능한 파일 이름을 지정하십시오.	$ENV
$RAHUSER(UNIX 기반 플랫폼); RAHUSER(Windows NT)	UNIX 기반 플랫폼에서 리모트 명령이 실행될 때의 사용자 ID Windows NT에서 DB2 리모트 명령 서비스와 연관된 로그온 어카운트	$USER

주: UNIX 기반 플랫폼에서 리모트 쉘이 설정한 값(있는 경우)이 아닌 **rah**가 실행되고 있는 $RAHENV의 값이 사용됩니다.

관련 참조:

- 414 페이지의 『UNIX 기반 플랫폼에서 $RAHDOTFILES 사용』

UNIX 기반 플랫폼에서 $RAHDOTFILES 사용

주: 이 절의 정보는 UNIX 기반 플랫폼에만 적용됩니다.

다음은 접두부 시퀀스가 지정되지 않을 때 실행되는 .file입니다.

P .profile

E $RAHENV에 이름 지정된 파일(`.kshrc`)

K E와 같음

PE $RAHENV에 이름 지정된 파일(.kshrc)이 뒤에 나오는 .profile

B PE와 같음

N 없음

주: 로그인 쉘이 Korn 쉘이 아닌 경우, 실행할 dot 파일을 Korn 쉘 프로세스에서 실행하고 Korn 쉘 구문을 준수해야 합니다. 따라서 로그인 쉘이 C 쉘인 경우, **rah**에 의해 실행되는 명령에 대해 .cshrc 환경 변수를 설정하려면, 다음과 같이 .cshrc와 상응하는 Korn 쉘 INSTHOME/.profile을 작성하고 INSTHOME/.cshrc에 지정해야 합니다.

```
setenv RAHDOTFILES P
```

또는 .cshrc에 상응하는 Korn 쉘 INSTHOME/.kshrc를 작성하고 INSTHOME/.cshrc에 지정해야 합니다.

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

또한, tty가 없는 경우(**rsh**에 의해 호출될 때), .cshrc가 stdout에 작성되지 못하게 해야 합니다. 다음과 같이 stdout에 작성하는 행을 넣어 확인할 수 있습니다.

```
if { tty -s } then echo "executed .cshrc";
endif
```

관련 참조:

• 413 페이지의『rah 명령 제어』

Windows NT에서 rah에 대한 디폴트 환경 프로파일 설정

프로시저:

주: 이 절의 정보는 Windows NT에만 적용됩니다.
rah 명령에 대한 디폴트 환경 프로파일을 설정하려면, 인스턴스 디렉토리에 작성되는 db2rah.env 파일을 사용하십시오. 이 파일은 다음 형식을 가져야 합니다.

```
; This is a comment line
DB2INSTANCE=instancename
DB2DBDFT=database
; End of file
```

rah에 대한 환경을 초기화하는 데 필요한 모든 환경 변수를 지정할 수 있습니다.

관련 개념:

• 405 페이지의『rah 및 db2_all 명령 지정』

UNIX 기반 플랫폼에서 rah 문제점 판별

주: 이 절의 정보는 UNIX 기반 플랫폼에만 적용됩니다.
다음은 **rah**를 실행할 때 발생할 수 있는 문제점을 처리하는 방법입니다.

1. **rah** 정지(또는 오랜 시간 경과)

   문제점의 원인은 다음과 같습니다.

   - **rah**는 출력을 버퍼링할 필요가 있으며 RAHCHECKBUF=no를 익스포트하지 않았음을 판별했습니다. 따라서, 명령을 실행하기 전에 **rah**는 모든 머신에 명령을 보내어 버퍼 디렉토리가 있는지 검사하고, 없는 경우 이를 작성합니다.
   - 명령을 보내고 있는 하나 이상의 머신이 응답하지 않습니다. **rsh** 명령은 결과적으로 시간종료되지만, 시간종료 간격은 보통 60초로 매우 깁니다.

2. 다음과 같은 메시지가 수신되었습니다.

   - 올바르지 않은 로그인
   - 사용 권한 거부

   머신 중 하나가 .hosts 파일에 적절히 정의된 **rah**를 실행하는 ID를 가지고 있지 않거나, **rah**를 실행하는 ID가 .rhosts 파일에 적절히 정의된 머신 중 하나를 가지고 있지 않습니다.

3. 명령이 머신에서 예상된 경과 시간 내에 실행되어 완료되어도, 백그라운드 rshell을 사용하여 명령을 병렬로 실행하는 경우에는 **rah**가 이를 감지하고 쉘 프롬프트로 올리는 데 많은 시간이 필요합니다.

   rah를 실행하는 ID가 .rhosts 파일에 적절히 정의된 머신 중 하나를 가지고 있지 않습니다.

4. 쉘 명령행으로부터 실행할 때 **rah**가 제대로 실행되더라도, 다음과 같이 rsh를 사용하여 리모트로 **rah**를 실행하면

   ```
   rsh somewher -l $USER db2_kill
   ```

   rah는 완료될 수 없습니다.

   이것은 정상적입니다. **rah**는 백그라운드 모니터링 프로세스를 시작하며 이 프로세스를 빠져나간 후에도 계속 실행됩니다. 수행하고 있는 명령과 연관된 모든 프로세스를 자체 종료할 때까지 프로세스는 정상적으로 지속됩니다. **db2_kill**의 경우, 이것은 모든 데이터베이스 관리 프로그램의 종료를 의미합니다. 명령이 **rahwaitfor** 및 kill <process_id>인 프로세스를 찾아 모니터링 프로세스를 종료할 수 있습니다. 신호 수를 지정하지 마십시오. 대신, 디폴트값 15를 사용하십시오.

5. 다중 **rah** 명령이 같은 $RAHUSER하에서 발행되어 있지 않을 때, **rah**로부터의 출력이 제대로 표시되지 않거나 **rah**가 $RAHBUFNAME이 존재하지 않음을 제대로 보고하지 않습니다.

이것은 **rah**의 다중 동시 실행이 출력을 버퍼링하기 위해 동일한 버퍼 파일(예: $RAHBUFDIR/$RAHBUFNAME)을 사용하려 하기 때문입니다. 이 문제점을 막으려면, 각 동시 **rah** 명령에 대해 다음 ksh에서처럼 $RAHBUFNAME을 사용하십시오.

```
export RAHBUFNAME=rahout
rah ";$command_1" &
export RAHBUFNAME=rah2out
rah ";$command_2" &
```

또는, 쉘이 다음과 같은 고유한 이름을 자동으로 선택하게 만드는 방법을 사용하십시오.

```
RAHBUFNAME=rahout.$$ db2_all "....."
```

어떤 방법을 사용하든지 디스크 스페이스에 한계가 있을 경우, 어떤 지점에서 버퍼 파일을 정리해야 하는지 확인해야 합니다. **rah**는 버퍼 파일을 지운 다음, 같은 버퍼 파일을 지정하는 다음 번에 기존 파일을 재사용해도 실행 마지막에 버퍼 파일을 지웁니다.

6. 다음을 입력하면

   ```
   rah '"print from ()'
   ```

   다음 메시지를 수신합니다.

   ```
   ksh: syntax error at line 1 : (' unexpected
   ```

   () 및 ## 대체를 위한 전제조건:

   - **rah**가 아닌 **db2_all**을 사용하십시오.
   - RAHOSTFILE을 익스포트하거나 /sqllib/db2nodes.cfg 파일을 디폴트 파일로 지정하여 RAHOSTFILE이 사용되는지 확인하십시오. 이러한 전제조건 없이, **rah**는 있는 그대로 () 및 ##을 남겨둡니다. 명령 **print from ()**이 유효하지 않으므로 오류가 수신됩니다.

   명령을 병렬로 실행할 때 성능 추가 정보를 얻으려면, &에서 제공한 함수가 필요하지 않는 한 |&보다는 |를, ||&보다는 ||를 사용하십시오. &를 지정하면, 더 많은 **rsh** 명령이 필요하므로 성능이 저하됩니다.

관련 참조:

- 413 페이지의『rah 명령 제어』

부록 E. WMI(Windows Management Instrumentation) 지원 사용

WMI(Windows Management Instrumentation) 입문

관리 하부구조 표준을 설정하고 다양한 하드웨어 및 소프트웨어 관리 시스템의 정보를 조합하는 방법을 제공하는 industry initiative가 있습니다. 이러한 initiative를 웹 기반 엔터프라이즈 관리(WBEM)라고 합니다. WBEM은 DMTF(Desktop Management Task Force)에 의해 구동되는 산업 표준인 CIM(Common Information Model) 스키마를 기반으로 합니다.

Microsoft® WMI(Windows® Management Instrumentation)는 지원되는 Windows 플랫폼에 대한 WBEM initiative의 구현입니다. WMI는 Windows 엔터프라이즈 네트워크에서 유지보수 및 엔터프라이즈 네트워크 구성요소 관리 비용을 줄이는데 도움이 됩니다. WMI는 다음을 제공합니다.

- Windows 조작, 구성 및 상태의 일관된 모델.
- 관리 정보로의 액세스를 가능하게 하는 COM API.
- 다른 Windows 관리 서비스의 조작 능력.
- 벤더가 새 디바이스, 응용프로그램 및 기타 개선된 기능을 지원하기 위해 다른 WMI 제공자를 작성할 수 있는 유연하고 확장 가능한 아키텍처.
- 자세한 정보 쿼리를 작성하는 WQL(WMI Query Language).
- 관리 응용프로그램 개발자가 Visual Basic 또는 WSH(Windows Scripting Host) 스크립트를 작성하기 위한 API.

WMI 아키텍처는 두 파트로 구성됩니다.

1. CIMOM(CIM Object Manager)와 CIMOM 오브젝트 저장소라고 하는 관리 데이터용 중앙 스토리지 영역을 포함하는 관리 하부구조. CIMOM은 응용프로그램이 일정한 방식으로 관리 데이터에 액세스할 수 있도록 합니다.
2. WMI 제공자. WMI 제공자는 CIMOM과 관리되는 오브젝트 사이의 중개자입니다. WMI 제공자는 WMI API를 사용하여 CIMOM에 관리되는 오브젝트의 데이터를 제공하고 관리 응용프로그램을 대신하여 요청을 처리하며 이벤트 통지를 생성합니다.

WMI 제공자는 관리되는 오브젝트와 CIMOM 사이에서 중개자 역할을 하는 표준 COM 또는 DCOM 서버입니다. CIMOM이 관리 응용프로그램으로부터 CIMOM 오브젝트 저

장소에 없는 데이터나 이벤트에 대한 요청을 수신하면, CIMOM은 이 요청을 WMI 제공자로 보냅니다. WMI 제공자는 특정 도메인에 속하는 관리되는 오브젝트의 데이터 및 이벤트 통지를 제공합니다.

관련 개념:

• 420 페이지의 『DB2 Universal Database와 WMI(Windows Management Instrumentation) 통합』

DB2 Universal Database와 WMI(Windows Management Instrumentation) 통합

WMI(Windows® Management Instrumentation)는 DB2® 성능 카운터 및 내장 PerfMon 제공자를 사용하여 스냅샷 모니터에 액세스할 수 있습니다.

WMI는 내장 레지스트리 제공자를 사용하여 DB2 프로파일 레지스트리 변수에 액세스할 수 있습니다.

WMI SDK(WMI Software Development Kit)에는 몇 개의 내장 제공자가 있습니다.

- PerfMon 제공자
- 레지스트리 이벤트 제공자
- 레지스트리 제공자
- Windows NT® 이벤트 로그 제공자
- Win32 제공자
- WDM 제공자

WMI는 내장 Windows NT 이벤트 로그 제공자를 사용하여 이벤트 로그에 있는 DB2 오류에 액세스할 수 있습니다.

DB2 Universal Database™(UDB)에는 다음과 같은 관리되는 오브젝트에 액세스하는 DB2 WMI 관리 제공자와 샘플 WMI 스크립트 파일이 있습니다.

1. 파티션된 인스턴스를 포함한 데이터베이스 서버 인스턴스. 다음 조작을 수행할 수 있습니다.
   - 인스턴스 제거
   - 데이터베이스 관리 프로그램 매개변수 구성
   - DB2 서버 서비스 시작/중지/상태 조회
   - 통신 설치 또는 설정
2. 데이터베이스. 다음 조작을 수행할 수 있습니다.
   - 데이터베이스 제거
   - 데이터베이스 매개변수 구성

- 데이터베이스 작성/삭제
- 데이터베이스 백업/복원/롤 포워드

WMI 응용프로그램을 실행하기 전에 DB2 WMI 제공자를 시스템에 등록해야 합니다. 등록하려면 다음 명령을 입력하십시오.

- mofcomp %DB2PATH%\bin\db2wmi.mof

  이 명령은 DB2 WMI 스키마 정의를 시스템에 로드합니다.

- regsvr %DB2PATH%\bin\db2wmi.dll

  이 명령은 DB2 WMI 제공자 COM DLL을 Windows에 등록합니다.

두 명령에서 %DB2PATH%는 DB2가 설치된 경로입니다. 또한 db2wmi.mof는 DB2 WMI 스키마 정의가 포함되는 .MOF 파일입니다.

WMI 하부구조를 통합하면 다음과 같은 이점이 있습니다.

1. WMI에서 제공하는 도구를 사용하여 Windows 기반 환경에서 DB2 서버를 관리하는 스크립트를 쉽게 작성할 수 있습니다. 인스턴스 나열, 데이터베이스 작성 및 삭제, 구성 매개변수 갱신 등의 간단한 태스크를 수행하는 샘플 Visual Basic(VBS) 스크립트가 제공됩니다. 샘플 스크립트는 Windows용 DB2 Application Development 제품에 들어 있습니다.
2. WMI를 사용하여 다양한 태스크를 수행하는 강력한 관리 응용프로그램을 작성할 수 있습니다. 수행하는 태스크에는 다음과 같은 것이 있습니다.
   - 시스템 정보 표시
   - DB2 성능 모니터링
   - DB2 시스템 자원 소모 모니터링

   이러한 유형의 관리 응용프로그램을 통해 시스템 이벤트와 DB2 이벤트를 둘 다 모니터링함으로써 데이터베이스를 보다 잘 관리할 수 있습니다.
3. 기존의 COM 및 Visual Basic 프로그래밍 지식 및 기술을 사용할 수 있습니다. COM 또는 Visual Basic 인터페이스를 제공함으로써, 프로그래머가 엔터프라이즈 관리 응용프로그램을 개발할 때 시간을 절약할 수 있습니다.

관련 개념:

- 419 페이지의『WMI(Windows Management Instrumentation) 입문』

부록 F. Windows NT 보안 사용

Windows NT 및 Windows NT용 DB2 보안 소개

Windows® NT 도메인은 고유한 이름으로 참조되는 클라이언트 및 서버 컴퓨터 배열로서, 보안 액세스 관리 프로그램(SAM)이라고 하는 단일 사용자 어카운트 데이터베이스를 공유합니다. 도메인에 있는 컴퓨터 중 하나가 도메인 제어기입니다. 도메인 제어기는 사용자와 도메인 간 상호작용의 모든 측면을 관리합니다. 도메인 제어기는 도메인 사용자 어카운트 데이터베이스에 있는 정보를 사용하여 도메인 어카운트에 로그온하는 사용자를 인증합니다. 각 도메인에서 하나의 도메인 제어기가 기본 도메인 제어기(PDC)입니다. 도메인에는 기본 도메인 제어기가 없거나 기본 도메인 제어기가 사용 불가능한 경우에 사용자 어카운트를 인증하는 백업 도메인 제어기(BDC)가 있을 수도 있습니다. 백업 도메인 제어기에는 PDC의 마스터 사본과 정기적으로 동기화되는 SAM 데이터베이스의 사본이 있습니다.

도메인 자원에 액세스하려면 기본 도메인 제어기에서 사용자 어카운트, 사용자 ID 및 암호를 정의하기만 하면 됩니다.

Windows NT® 서버 설치 시의 설치 프로시저 도중, 다음을 선택할 수 있습니다.

- 새 도메인에 기본 도메인 제어기 작성
- 알려진 도메인에 백업 도메인 제어기 작성
- 알려진 도메인에 독립형 서버 작성

새 도메인에서 『제어기』를 선택하면 해당 서버가 기본 도메인 제어기가 됩니다.

사용자는 로컬 머신에 로그온할 수 있거나, 머신이 Windows NT 도메인에 설치될 경우 도메인에 로그온할 수 있습니다. Windows NT용 DB2®는 이 두 옵션을 모두 지원합니다. 사용자를 인증하기 위해, DB2는 먼저 로컬 머신을 검사한 후 현재 도메인에 대한 도메인 제어기를 검사하고, 마지막으로 도메인 제어기에 알려진 신뢰성 있는 도메인을 검사합니다.

Windows NT용 DB2가 작동하는 방법을 보려면, DB2 인스턴스가 서버 인증을 요구한다고 가정하십시오. 구성은 다음과 같습니다.

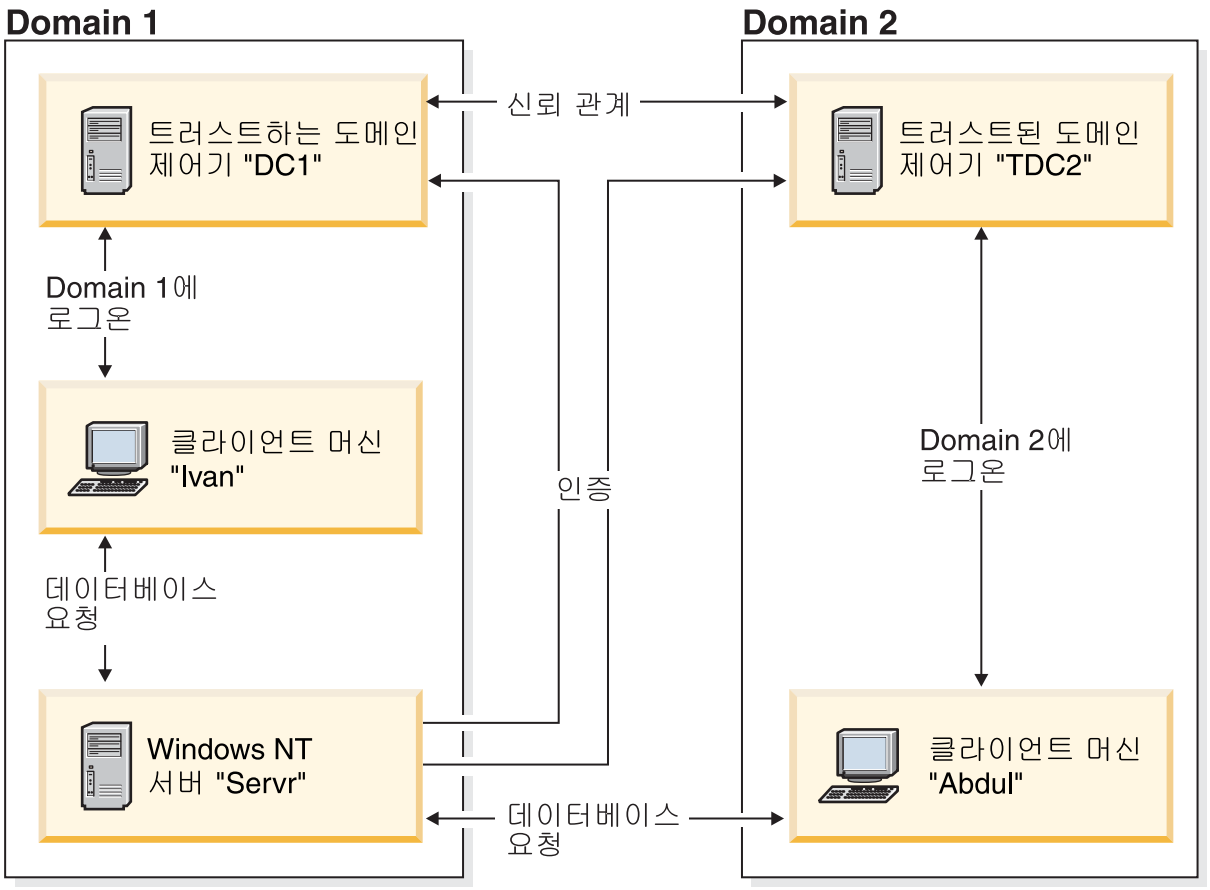


그림 6. Windows NT 도메인 사용 인증

각 머신은 클라이언트 머신에서 Windows 9x가 실행되지 않는 한, 보안 데이터베이스인 보안 액세스 관리(SAM) 데이터베이스가 있습니다. Windows 9x 머신에는 SAM 데이터베이스가 없습니다. DC1은 클라이언트 머신, Ivan 및 Windows NT 서버인 Servr용 DB2가 등록되는 도메인 제어기입니다. TDC2는 DC1과 클라이언트 머신인 Abdul용 트러스트된 도메인으로, TDC2 도메인의 구성원입니다.

관련 개념:

- 428 페이지의 『Windows NT에서의 그룹 및 사용자 인증』

태스크 관련:

- 427 페이지의 『DB2 UDB에서 백업 도메인 제어기 사용』
- 430 페이지의 『백업 도메인 제어기에서 DB2 설치』
- 431 페이지의 『그룹 및 도메인 보안으로 Windows NT용 DB2 인증』

Windows NT용 DB2 서버 인증 시나리오

1. Abdul은 TDC2 도메인에 로그온합니다(즉, TDC2 SAM 데이터베이스에 알려집니다).

2. 그런 다음, Abdul을 SRV3에 상주하기 위해 카탈로그에 등록되는 DB2 데이터베이스에 연결합니다.

```
db2 connect to remotedb user Abdul using fredpw
```

3. SRV3은 Abdul이 알려진 곳을 판별합니다. 이 정보를 찾기 위해 사용되는 API는 신뢰성 있는 도메인을 시도하기 전에 먼저 로컬 머신(SRV3)을 검색한 후 도메인 제어기(DC1)를 검색합니다. 사용자 이름 Abdul이 TDC2에서 발견됩니다. 이 검색 순서에서는 사용자 또는 그룹에 대해 하나의 이름 공간이 필요합니다.
4. 그런 다음, SRV3은 다음을 수행합니다.
   a. TDC2에 대해 사용자 이름과 암호의 유효성을 확인합니다.
   b. TDC2에 요청하여 Abdul이 관리자인지 찾습니다.
   c. TDC2에 요청하여 Abdul의 모든 그룹을 열거합니다.

관련 개념:

• 423 페이지의 『Windows NT 및 Windows NT용 DB2 보안 소개』

Windows NT용 DB2 클라이언트 인증 시나리오 및 Windows NT 클라이언트 머신

1. 관리자인 Dale은 SRV3에 로그온하여 데이터베이스 인스턴스에 대한 인증을 클라이언트로 변경합니다.

```
db2 update dbm cfg using authentication client
 db2stop
db2start
```

2. Windows 클라이언트 머신에서 Ivan은 DC1 도메인에 로그온합니다(즉, DC1 SAM 데이터베이스에 알려집니다).
3. 그런 다음, Ivan을 SRV3에 상주시키기 위해 카탈로그에 등록되는 DB2 데이터베이스에 연결합니다.

```
DB2 CONNECT to remotedb user Ivan using johnpw
```

4. Ivan의 머신은 사용자 이름과 암호의 유효성을 확인합니다. 이 정보를 찾기 위해 사용되는 API는 신뢰성 있는 도메인을 시도하기 전에 먼저 로컬 머신(Ivan)을 검색한 후 도메인 제어기(DC1)를 검색합니다. 사용자 이름 Ivan이 DC1에서 발견됩니다.
5. 그러면 Ivan의 머신은 DC1에 대해 사용자 이름과 암호의 유효성을 확인합니다.
6. 그런 다음, SRV3은 다음을 수행합니다.
   a. Ivan이 알려진 곳을 판별합니다.
   b. DC1에 요청하여 Ivan이 관리자인지 찾습니다.
   c. DC1에 요청하여 Ivan의 모든 그룹을 나열합니다.

주: DB2 데이터베이스에 연결을 시도하기 전에, DB2 보안 서비스가 시작되었는지 확인하십시오. 보안 서비스는 Windows 설치의 일부로서 설치됩니다. 그런 다음 DB2

가 설치되고 Windows NT 서비스로 『등록』되나, 자동으로 시작되지는 않습니다. DB2 보안 서비스를 시작하려면, NET START DB2NTSECSERVER 명령을 입력하십시오.

관련 개념:

- 423 페이지의 『Windows NT 및 Windows NT용 DB2 보안 소개』

Windows NT용 DB2 클라이언트 인증 시나리오 및 Windows 9x 클라이언트 머신

1. 관리자인 Dale은 SRV3에 로그온하여 데이터베이스 인스턴스에 대한 인증을 클라이언트로 변경합니다.

```
db2 update dbm cfg using authentication client
 db2stop
db2start
```

2. Windows 9x 클라이언트 머신에 있는 Ivan이 DC1 도메인에 로그온합니다(즉 그는 DC1 SAM 데이터베이스에 알려져 있습니다).
3. 그런 다음, Ivan을 SRV3에 상주시키기 위해 카탈로그에 등록되는 DB2 데이터베이스에 연결합니다.

```
db2 connect to remotedb user Ivan using johnpw
```

4. Ivan의 Windows 9x 머신은 사용자 이름 및 암호의 유효성을 확인할 수 없습니다. 그러므로 사용자 이름과 암호는 유효한 것으로 가정합니다.
5. 그런 다음, SRV3은 다음을 수행합니다.
   a. Ivan이 알려진 곳을 판별합니다.
   b. DC1에 요청하여 Ivan이 관리자인지 찾습니다.
   c. DC1에 요청하여 Ivan의 모든 그룹을 나열합니다.

주: Windows 9x 클라이언트는 주어진 사용자 이름 및 암호의 유효성을 확인할 수 없으므로, Windows 9x에서의 클라이언트 인증은 본질적으로 안전하지 못합니다. 그러나 Windows 9x 머신이 Windows NT 보안 제공업체에 대한 액세스 권한을 가지면, 유효성이 확인된 pass-through 로그온이 가능하도록 Windows 9x 시스템을 구성함으로써 어느 정도의 보안을 부여할 수 있습니다. 이러한 방식으로 Windows 9x 시스템을 구성하는 방법에 대해서는 Windows 9x에 대한 Microsoft 문서를 참조하십시오.

관련 개념:

- 423 페이지의 『Windows NT 및 Windows NT용 DB2 보안 소개』
- 427 페이지의 『전역 그룹 지원(Windows에서)』

전역 그룹 지원(Windows에서)

DB2® Universal Database(DB2 UDB)는 전역 그룹을 또한 지원합니다. 전역 그룹을 사용하려면, 로컬 그룹안에 전역 그룹이 포함되어 있어야 합니다. 사용자가 구성원으로 있는 모든 그룹을 DB2 UDB가 열거할 때, 사용자가 간접적으로 구성원인 로컬 그룹도 나열합니다(자체가 하나 이상의 로컬 그룹 구성원인 전역 그룹의 효력으로).

전역 그룹은 두 가지 상황에서 사용됩니다.

- 로컬 그룹에 포함. 로컬 그룹에 사용 권한을 부여해야 합니다.
- 도메인 제어기에 포함. 전역 그룹에 사용 권한을 부여해야 합니다.

관련 개념:

- 428 페이지의『Windows NT에서의 그룹 및 사용자 인증』

DB2 UDB에서 백업 도메인 제어기 사용

프로시저:

DB2™(DB2 UDB)에 대해 사용하는 서버가 백업 도메인 제어기로도 작동할 경우, DB2 UDB를 백업 도메인 제어기를 사용하도록 구성하면 DB2 UDB를 향상시키고 네트워크 트래픽을 줄일 수 있습니다.

DB2DMNBCKCTLR 레지스트리 변수를 설정하여 백업 도메인 제어기를 DB2 UDB에 지정합니다.

DB2 UDB 서버가 백업 도메인 제어기인 도메인의 이름을 알면, 다음을 사용하십시오.

```
db2dmnbckctlr=<domain_name>
```

여기서, domain_name은 대문자이어야 합니다.

DB2 UDB가 백업 도메인 제어기인 로컬 머신에 대한 도메인을 판별하도록 하려면, 다음을 사용하십시오.

```
DB2DMNBCKCTLR=?
```

주: 백업 도메인 제어기가 1차 도메인 제어기와 비동기화되어 보안을 노출시킬 수 있으므로 DB2 UDB는 디폴트로 기존의 백업 도메인 제어기를 사용하지 않습니다. 도메인 제어기는 기본 도메인 제어기의 보안 데이터베이스가 갱신될 때 동기화에서 벗어나지만, 변경사항은 백업 도메인 제어기에 전파되지 않습니다. 이러한 상황은 네트워크 지연이 있거나 컴퓨터 브라우저 서비스가 작동하지 않을 경우에 발생할 수 있습니다.

태스크 관련:

- 430 페이지의 『백업 도메인 제어기에서 DB2 설치』

Windows NT용 DB2로 사용자 인증

운영 체제 인증 방법 때문에 사용자 인증은 Windows NT 사용자에게 문제점을 야기할 수 있습니다. 이 절은 Windows NT용 DB2하에서 사용자 인증에 대한 고려사항을 설명합니다.

- 『Windows NT용 DB2 사용자 이름 및 그룹 이름 제한사항』
- 430 페이지의 『Windows NT용 DB2 보안 서비스』
- 430 페이지의 『백업 도메인 제어기에서 DB2 설치』
- 431 페이지의 『그룹 및 도메인 보안으로 Windows NT용 DB2 인증』

Windows NT용 DB2 사용자 이름 및 그룹 이름 제한사항

다음은 이 환경에서의 제한사항입니다.

- 사용자 이름 및 그룹 이름은 DB2 Universal Database™(DB2 UDB) 내에서 30자로 제한됩니다.
- Windows NT하의 사용자 이름은 대소문자가 구별되지 않지만, 암호는 대소문자가 구별됩니다.
- 사용자 이름 및 그룹 이름은 대문자 및 소문자의 조합일 수 있습니다. 그러나 DB2 UDB 내에서 사용될 때 보통 대문자로 변환됩니다. 예를 들어, 데이터베이스에 연결하고 schema1.table1 테이블을 작성하는 경우, 이 테이블은 데이터베이스 내에서 SCHEMA1.TABLE1로서 저장됩니다. 소문자 오브젝트 이름을 사용하려는 경우, 오브젝트 이름을 따옴표로 묶어 명령행 처리기에서 명령을 발행하거나, 써드 파티 ODBC 프론트엔드 도구를 사용하십시오.
- 사용자는 64개 이하의 그룹에만 속할 수 없습니다.
- DB2 UDB는 단일 이름 공간을 지원합니다. 즉 트러스트된 도메인 환경에서 실행 중일 때, 다중 도메인에 있거나 서버 머신의 로컬 SAM 및 다른 도메인에 있는 것과 동일한 이름의 사용자 어카운트를 가질 수 없습니다.

관련 개념:

- 428 페이지의 『Windows NT에서의 그룹 및 사용자 인증』
- 429 페이지의 『Windows NT에 있는 도메인 간의 신뢰 관계』

Windows NT에서의 그룹 및 사용자 인증

Windows® NT에서는 『사용자 관리 프로그램』이라는 Windows NT® 관리 도구를 사용하여 사용자 어카운트를 작성함으로써 사용자가 정의됩니다.

다른 어카운트(구성원이라고도 함)를 포함하는 어카운트가 그룹입니다. 그룹은 Windows NT 관리자에게 각 사용자를 개별적으로 유지보수할 필요 없이 그룹 내의 사용자에게 동시에 권한 및 허용을 부여할 수 있는 능력을 제공합니다. 사용자 어카운트와 마찬가지로, 그룹은 SAM(Security Access Manager) 데이터베이스에서 정의되고 유지보수됩니다.

두 가지 유형의 그룹이 있습니다.

- 로컬 그룹. 로컬 그룹에는 로컬 어카운트 데이터베이스에서 작성된 사용자 어카운트가 포함될 수 있습니다. 로컬 그룹이 도메인의 일부가 되는 머신에 있으면, 로컬 그룹에는 해당 Windows NT 도메인의 도메인 어카운트 및 그룹도 포함될 수 있습니다. 워크스테이션에서 로컬 그룹을 작성하면, 이 로컬 그룹은 해당 워크스테이션에 고유합니다.
- 전역 그룹. 전역 그룹은 도메인 제어기에만 있으며 해당 도메인의 SAM 데이터베이스의 사용자 어카운트가 포함됩니다. 즉, 전역 그룹에는 해당 전역 그룹이 작성된 도메인의 사용자 어카운트만 포함될 수 있으며 다른 그룹은 구성원으로서 포함될 수 없습니다. 전역 그룹은 자체 도메인의 서버 및 워크스테이션에서 사용될 수 있습니다.

관련 개념:

- 429 페이지의 『Windows NT에 있는 도메인 간의 신뢰 관계』
- 427 페이지의 『전역 그룹 지원(Windows에서)』

태스크 관련:

- 431 페이지의 『그룹 및 도메인 보안으로 Windows NT용 DB2 인증』

관련 참조:

- 428 페이지의 『Windows NT용 DB2 사용자 이름 및 그룹 이름 제한사항』

Windows NT에 있는 도메인 간의 신뢰 관계

신뢰 관계는 두 도메인 간의 관리 및 통신 링크입니다. 두 도메인 간의 신뢰 관계는 사용자 어카운트 및 전역 그룹이 정의된 도메인이 아닌 다른 도메인에서 그러한 어카운트를 사용할 수 있도록 합니다. 인증 과정 없이 트러스트된 도메인에 있는 사용자 어카운트 및 전역 그룹의 권한 및 허용의 유효성을 확인할 수 있도록 어카운트 정보가 공유됩니다. 신뢰 관계는 둘 이상의 도메인을 단일의 관리 단위로 결합함으로써 사용자 관리를 단순화합니다.

신뢰 관계에는 두 개의 도메인이 있습니다.

- 트러스트하는 도메인. 이 도메인은 다른 도메인이 그들을 대신하여 사용자를 인증할 것임을 신뢰합니다.
- 트러스트된 도메인. 이 도메인은 다른 도메인을 대신하여(신뢰 하에) 사용자를 인증합니다.

신뢰 관계는 이행적이지 않습니다. 이는 도메인 간에 각 방향으로 명시적 신뢰 관계를 수립해야 함을 의미합니다. 예를 들어, 트러스트하는 도메인은 트러스트된 도메인이 아닐 수도 있습니다.

관련 개념:

- 428 페이지의 『Windows NT에서의 그룹 및 사용자 인증』
- 427 페이지의 『전역 그룹 지원(Windows에서)』

관련 참조:

- 428 페이지의 『Windows NT용 DB2 사용자 이름 및 그룹 이름 제한사항』

Windows NT용 DB2 보안 서비스

DB2® Universal Database(DB2 UDB)에서 사용자 이름 및 암호 인증을 DB2 시스템 제어기로 통합했습니다. 보안 서비스는 인증 CLIENT용으로 구성되는 서버로 클라이언트를 연결할 때에만 요구됩니다.

관련 개념:

- 423 페이지의 『Windows NT 및 Windows NT용 DB2 보안 소개』

백업 도메인 제어기에서 DB2 설치

프로시저:

Windows NT 4.0 환경에서는 기본 또는 백업 제어기에서 사용자를 인증할 수 있습니다. 이 기능은 하나의 중앙 기본 도메인 제어기와 각 사이트에 하나 이상의 백업 도메인 제어기(BDC)가 있는 대형 분산 LAN에서 매우 중요합니다. 그런 다음, 사용자는 인증을 위해 기본 도메인 제어기(PDC)에 호출을 요청하는 대신 해당 사이트의 백업 도메인 제어기에서도 인증될 수 있습니다.

이 경우, 백업 도메인 제어기를 갖는 이점은 사용자가 더 빠르게 인증되며 LAN은 BDC가 없었을 때처럼 혼잡하지 않습니다.

다음 조건하에서 BDC에서 인증이 발생할 수 있습니다.

- Windows NT용 DB2 서버는 백업 도메인 제어기에 설치됩니다.
- DB2DMNBCKCTLR 프로파일 레지스트리 변수가 제대로 설정되어 있습니다.

DB2DMNBCKCTLR 프로파일 레지스트리 변수가 설정되지 않거나 공백으로 설정되어 있는 경우, Windows NT용 DB2는 기본 도메인 제어기에서 인증을 수행합니다.

DB2DMNBCKCTLR용으로 선언된 유일하게 유효한 설정은 『?』 또는 도메인 이름입니다.

DB2DMNBCKCTLR 프로파일 레지스트리 변수가 물음표로 설정되면 (DB2DMNBCKCTLR=?), Windows NT용 DB2는 다음과 같은 조건하에서 백업 도메인 제어기에서 인증을 수행합니다.

- cachedPrimaryDomain은 이 머신이 속하는 도메인의 이름으로 설정된 레지스트리 값입니다. **HKEY_LOCAL_MACHINE —> Software —> Microsoft —> Windows NT —> Current Version —> WinLogon** 아래에서 이 설정을 찾을 수 있습니다.
- 서버 관리 프로그램은 백업 도메인 제어기를 사용 중이며 사용 가능한 것으로 표시합니다. 즉, 이 머신의 아이콘은 회색으로 표시되지 않습니다.
- DB2 Windows NT 서버의 레지스트리는 시스템이 지정된 도메인에서 백업 도메인 제어기임을 나타냅니다.

정상적인 환경에서 설정 DB2DMNBCKCTLR=?은 작동하지만 모든 환경에서 작동하는 것은 아닙니다. 도메인에서 서버에 대해 제공된 정보는 동적이며, 컴퓨터 브라우저는 이 정보를 정확하고 현재의 것으로 보존하기 위해 실행 중이어야 합니다. 대형 LAN은 실행 중인 컴퓨터 브라우저가 아닐 수 있으므로 서버 관리 프로그램의 정보가 현재의 것이 아닐 수 있습니다. 이 경우, 백업 도메인 제어기에서 Windows NT용 DB2가 인증하도록 알리는 두 번째 메소드가 있습니다. DB2DMNBCKCTLR=*xxx*를 설정하십시오. 여기서, *xxx*는 DB2 서버의 Windows NT 도메인 이름입니다. 이 설정을 사용하여, 인증은 다음과 같은 조건에 근거하여 백업 도메인 제어기에서 발생할 수 있습니다.

- cachedPrimaryDomain은 이 머신이 속하는 도메인의 이름으로 설정된 레지스트리 값입니다. **HKEY_LOCAL_MACHINE —> Software —> Microsoft—> Windows NT —> Current Version —> WinLogon** 아래에서 이 설정을 찾을 수 있습니다.
- 머신은 지정된 도메인의 백업 도메인 제어기로서 구성됩니다 (머신이 백업 도메인 제어기로서 설정되어 또다른 도메인용인 경우, 이 설정은 오류를 발생시킵니다).

태스크 관련:

- 427 페이지의 『DB2 UDB에서 백업 도메인 제어기 사용』

그룹 및 도메인 보안으로 Windows NT용 DB2 인증

프로시저:

DB2 Universal Database™(DB2 UDB)에서는 특권을 부여하거나 권한 레벨을 정의할 때 로컬 그룹 또는 전역 그룹을 지정할 수 있습니다. 사용자 어카운트가 로컬 또는 전역 그룹에 명시적으로 정의되거나 로컬 그룹의 구성원으로 정의되는 전역 그룹의 구성원이 됨으로써 내재적으로 로컬 또는 전역 그룹에 정의될 경우, 해당 사용자는 그 그룹의 구성원으로 판별됩니다.

Windows NT용 DB2는 다음과 같은 유형의 그룹을 지원합니다.

- 로컬 그룹
- 전역 그룹
- 로컬 그룹 및 전역 그룹의 구성원

  Windows NT용 DB2는 사용자가 찾은 보안 데이터베이스를 사용하여 사용자가 구성원인 로컬 및 전역 그룹을 열거합니다. DB2 UDB는 사용자 어카운트가 있는지 여부에 관계없이 DB2 UDB가 설치된 로컬 Windows NT 서버에서 그룹 열거가 발생하게 하는 겹쳐쓰기를 제공합니다. 이 겹쳐쓰기는 다음과 같은 명령을 사용하여 이루어집니다.

  – 전역 설정의 경우

    ```
    db2set -g DB2_GRP_LOOKUP=local
    ```

  – 인스턴스 설정의 경우

    ```
    db2set -i <instance name> DB2_GRP_LOOKUP=local
    ```

    이 명령을 발행한 후에는 변경사항이 적용되도록 DB2 UDB 인스턴스를 중지했다가 다시 시작해야 합니다. 그런 다음 로컬 그룹을 작성하여 해당 로컬 그룹에 도메인 어카운트 또는 전역 그룹을 포함시키십시오.

  설정된 모든 DB2 프로파일 레지스트리 변수를 보려면, 다음을 입력하십시오.

  ```
  db2set -all
  ```

DB2_GRP_LOOKUP 프로파일 레지스트리 변수가 로컬로 설정된 경우, DB2 UDB는 로컬 머신에서만 사용자를 찾으려고 합니다. 사용자가 로컬 머신에 없는 경우나, 로컬 또는 전역 그룹의 구성원으로서 정의되지 않은 경우, 인증에 실패합니다. DB2는 도메인의 또다른 머신 또는 도메인 제어기에서 사용자를 찾으려고 시도하지 **않습니다**.

DB2_GRP_LOOKUP 프로파일 레지스트리 변수가 설정되지 않은 경우

1. DB2 UDB는 우선 동일한 머신에서 사용자를 찾으려고 시도합니다.
2. 사용자 이름이 로컬로 정의된 경우, 사용자는 로컬로 인증됩니다.
3. 로컬로 사용자를 찾지 못한 경우, DB2 UDB는 도메인에서 사용자 이름을 찾은 다음 신뢰성 있는 도메인에서 찾으려고 시도합니다.

자원 도메인의 기본 또는 백업 도메인 제어기인 머신에서 DB2 UDB가 실행 중이면 DB2는 모든 트러스트된 도메인에 있는 모든 도메인 제어기를 찾을 수 있습니다. 그 이유는 도메인 제어기에서 실행 중일 경우에만 트러스트된 도메인에 있는 백업 도메인 제어기의 도메인 이름을 알 수 있기 때문입니다.

DB2 UDB가 도메인 제어기에서 실행되지 않을 경우에는 다음을 발행해야 합니다.

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

이 명령은 DB2 UDB에 자체 도메인에 있는 도메인 제어기를 사용하여 어카운트 도메인에 있는 도메인 제어기의 이름을 찾도록 지시합니다. 즉 DB2 UDB는 특정 사용자 어카운트가 도메인 x에 정의되어 있음을 알게 될 경우, 도메인 x의 도메인 제어기를 찾기 보다는 자체 도메인에 있는 도메인 제어기에 그러한 요청을 보냅니다. 어카운트 도메인에 있는 도메인 제어기의 이름이 찾아지고 DB2 UDB가 실행 중인 머신으로 리턴됩니다. 이 방법에는 두 가지 이점이 있습니다.

1. 기본 도메인 제어기가 사용 불가능할 때 백업 도메인 제어기를 찾습니다.
2. 기본 도메인 제어기가 위치상으로 멀리 있을 때 가까운 백업 도메인 제어기를 찾습니다.

관련 개념:

• 428 페이지의『Windows NT에서의 그룹 및 사용자 인증』

정렬된 도메인 목록을 사용한 인증

트러스트된 도메인 포레스트에 사용자 ID를 두 번 이상 정의할 수 있습니다. 트러스트된 도메인 포레스트는 네트워크를 통해 상호 관련된 도메인 콜렉션입니다. 한 도메인의 사용자가 다른 도메인의 또다른 사용자와 동일한 사용자 ID를 사용할 수 있습니다. 이 경우 다음과 같은 작업을 수행하고자 할 때 문제점이 발생할 수도 있습니다.

- 동일한 사용자 ID를 갖지만 서로 다른 도메인에 위치하는 다중 사용자 인증
- 그룹을 기반으로 특권을 부여하거나 권한 취소하기 위한 그룹 찾아보기
- 암호 유효성 확인
- 네트워크 트래픽 제어.

프로시저:

도메인 포리스트(forest)에서 다중 사용자가 동일한 사용자 ID를 가질 가능성에서 발행하는 문제점을 방지하려면, **db2set** 및 레지스트리 변수 DB2DOMAINLIST를 사용하여 정의된 대로 정렬된 도메인 목록을 사용해야 합니다. 순서를 설정할 경우, 목록에 포함시킬 도메인을 쉼표로 분리해야 합니다. 사용자 인증 시 도메인 검색 순서에 관하여 신중한 의사결정을 수행해야 합니다.

도메인 목록에서 훨씬 아랫쪽에 있는 도메인에 표시된 사용자 ID는 도메인 액세스를 위한 인증을 받고자 할 경우 이름을 변경해야 합니다.

액세스 제어는 도메인 목록을 통해 수행할 수 있습니다. 예를 들어, 사용자의 도메인이 목록에 없을 경우, 사용자는 연결을 수행할 수 없습니다.

주: DB2DOMAINLIST 레지스트리 변수는 CLIENT 인증이 데이터베이스 관리 프로그램 구성에서 설정된 경우에만 유효하며 Windows NT 도메인 환경에서 Windows NT 데스크탑으로부터의 단일 사인온이 필요한 경우 사용됩니다.

관련 개념:

• 423 페이지의『Windows NT 및 Windows NT용 DB2 보안 소개』

Windows NT용 DB2 도메인 보안 지원

다음 예는 Windows NT용 DB2가 어떻게 도메인 보안을 지원하는가를 나타냅니다. 이 첫 번째 예에서 사용자 이름 및 로컬 그룹은 동일한 도메인에 있으므로 연결이 작동합니다. 두 번째 예에서 사용자 이름 및 로컬 또는 전역 그룹은 다른 도메인에 있으므로 연결이 작동하지 않습니다.

성공적인 연결의 예: 사용자 이름 및 로컬 또는 전역 그룹은 동일한 도메인에 있으므로 다음 시나리오에서 연결이 작동합니다.

사용자 이름 및 로컬 또는 전역 그룹은 데이터베이스 서버가 실행 중이지만 서로 동일한 도메인에 있어야 하는 도메인에 정의될 필요가 없음을 기억하십시오.

표 52. 도메인 제어기를 사용한 성공적 연결

Domain1	**Domain2**
Domain2와의 신뢰 관계가 있습니다.	• Domain1과의 신뢰 관계가 있습니다. • 로컬 또는 전역 그룹 grp2가 정의됩니다. • 사용자 이름 id2가 정의됩니다. • 사용자 이름 id2는 grp2의 일부입니다.

DB2 서버는 이 도메인에서 실행합니다. 다음과 같은 DB2 명령이 발행됩니다.
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

로컬 또는 전역 도메인이 스캔되지만 id2가 없습니다. 도메인 보안이 스캔됩니다.	
	사용자 이름 id2가 이 도메인에 있습니다. DB2는 이 사용자 이름에 대한 추가 정보를 얻습니다(즉, 이 사용자 이름은 grp2의 일부입니다).
사용자 이름 및 로컬 또는 전역 그룹은 동일한 도메인에 있으므로 연결이 작동합니다.	

관련 개념:

• 428 페이지의『Windows NT에서의 그룹 및 사용자 인증』

태스크 관련:

• 431 페이지의『그룹 및 도메인 보안으로 Windows NT용 DB2 인증』

부록 G. Windows 성능 모니터 사용

Windows 성능 모니터 소개

Windows®용 DB2® Universal Database(DB2 UDB)에 대해 작업할 경우, 다음과 같은 도구를 사용하여 성능을 모니터할 수 있습니다.

- **DB2 Performance Expert**

  멀티플랫폼용 DB2 Performance Expert, 버전 1.1은 DB2 UDB 성능 관련 정보를 기반으로 자체 관리 및 자원 조정 변경사항을 통합, 보고, 분석 및 권장합니다.

- **DB2 UDB Health Center**

  Health Center의 기능은 성능 관련 정보에 대해 작업하기 위한 서로 다른 메소드를 제공합니다. 이들 기능은 제어 센터의 성능 모니터 기능을 어느 정도 대신합니다.

- **Windows 성능 모니터**

  Windows 성능 모니터를 사용하여 데이터베이스 및 시스템 성능을 모니터하고 시스템에 등록된 성능 데이터 제공업체로부터 정보를 검색할 수 있습니다. Windows는 또한 다음을 포함하여 머신 조작의 모든 측면에 대한 성능 정보 데이터를 제공합니다.

  – CPU 사용
  – 메모리 활용
  – 디스크 활동
  – 네트워크 활동

태스크 관련:

- 436 페이지의 『Windows 성능 모니터에 DB2 등록』
- 436 페이지의 『DB2 성능 정보에 대한 리모트 액세스 사용』
- 437 페이지의 『DB2 UDB 및 DB2 Connect 성능 값 표시』
- 439 페이지의 『리모트 DB2 UDB 성능 정보 액세스』
- 439 페이지의 『DB2 성능 값 재설정』

관련 참조:

- 438 페이지의 『Windows 성능 오브젝트』

Windows 성능 모니터에 DB2 등록

프로시저:

설치 프로그램은 자동으로 DB2를 Windows 성능 모니터에 등록합니다.

Windows 성능 모니터가 DB2 Universal Database™(DB2 UDB) 및 DB2 Connect 성능 정보에 액세스할 수 있도록 하려면 Windows용 DB2 성능 카운트에 대한 DLL을 등록해야 합니다. 그러면 Win32 성능 API를 사용하는 다른 모든 Windows 응용 프로그램도 성능 데이터를 확보할 수 있습니다.

Windows용 DB2 성능 카운터 DLL(DB2Perf.DLL)을 설치하여 Windows 성능 모니터에 등록하려면 다음을 입력하십시오.

```
db2perfi -i
```

DLL을 등록하면 레지스트리의 서비스 옵션에 새로운 키를 작성합니다. 한 항목은 카운터 지원을 제공하는 DLL의 이름을 지정하고, 다른 세 항목은 해당 DLL에 제공된 함수의 이름을 제공합니다. 이들 함수는 다음과 같습니다.

- **열기**

  프로세스에서 DLL이 시스템에 의해 최초로 로드될 때 호출됩니다.

- **수집**

  DLL에서 성능 정보를 요청하기 위해 호출됩니다.

- **닫기**

  DLL이 언로드될 때 호출됩니다.

관련 참조:

- *Command Reference*의 『db2perfi - Performance Counters Registration Utility Command』

DB2 성능 정보에 대한 리모트 액세스 사용

프로시저:

Windows용 DB2 워크스테이션이 다른 Windows 머신에 네트워크로 연결되어 있을 경우, 이 절에 설명되어 있는 기능을 사용할 수 있습니다.

다른 Windows용 DB2 머신에서 Windows 성능 오브젝트를 보려면 DB2 Universal Database™(DB2 UDB)에 관리자 사용자 이름 및 암호를 등록해야 합니다. (디폴트 Windows 성능 모니터 사용자 이름인 **SYSTEM**은 DB2 UDB 예약어이므로 사용할 수 없습니다). 이름을 등록하려면, 다음을 입력하십시오.

```
db2perfr -r username password
```

주: 사용된 username은 DB2 UDB 이름 지정 규칙에 따라야 합니다.

사용자 이름과 암호 데이터는 관리자와 SYSTEM 어카운트만 액세스할 수 있는 보안 하에 레지스트리에 있는 키에 보관됩니다. 관리자 암호를 레지스트리에 저장할 때 보안 문제점을 피하기 위해 데이터를 암호화합니다.

주:

1. 일단 사용자 이름과 암호 조합이 DB2 UDB에 등록되고 나면, 성능 모니터의 로컬 인스턴스조차 해당 사용자 이름과 암호 사용에 대해 명시적으로 로그온합니다. 이는 DB2 UDB에 등록된 사용자 이름 정보가 일치하지 않을 경우, 성능 모니터의 로컬 세션이 DB2 UDB 성능 정보를 표시하지 않는다는 것을 의미합니다.
2. 사용자 이름 및 암호 조합은 Windows 보안 데이터베이스에 저장된 사용자 이름 및 암호 값과 일치하도록 유지보수되어야 합니다. Windows 보안 데이터베이스에서 사용자 이름 또는 암호가 변경되면, 리모트 성능 모니터링에 사용되는 사용자 이름 및 암호 조합도 재설정해야 합니다.
3. 등록 해제하려면, 다음을 입력하십시오.

   ```
   db2perfr -u <username> <password>
   ```

관련 개념:

- 343 페이지의 『일반 이름 지정 규칙』

관련 참조:

- *Command Reference*의 『db2perfr - Performance Monitor Registration Tool Command』

DB2 UDB 및 DB2 Connect 성능 값 표시

프로시저:

성능 모니터를 사용하여 DB2 Universal Database™(DB2 UDB) 및 DB2 Connect 성능 값을 표시하려면, **추가 대상(Add to)** 상자로부터 값을 표시하고자 하는 성능 카운터를 선택하기만 하십시오. 이 상자는 성능 데이터를 제공하는 성능 오브젝트 목록을 표시합니다. 한 오브젝트를 선택하여 오브젝트가 제공하는 카운터 목록을 보십시오.

성능 오브젝트는 다중 인스턴스도 가질 수 있습니다. 예를 들어, LogicalDisk 오브젝트는 『% 디스크 읽기 시간』 및 『디스크 바이트/초』와 같은 카운터를 제공하며, 머신에 있는 『C:』 및 『D:』를 포함하는 각 논리 드라이브에 대한 인스턴스도 소유합니다.

관련 개념:

- 435 페이지의 『Windows 성능 모니터 소개』

관련 참조:

- 438 페이지의 『Windows 성능 오브젝트』

Windows 성능 오브젝트

Windows에서는 다음과 같은 성능 오브젝트를 제공합니다.

- **DB2 데이터베이스 관리 프로그램**

  이 오브젝트는 단일 Windows 인스턴스에 대한 일반 정보를 제공합니다. 모니터 중인 DB2 Universal Database™(DB2 UDB) 인스턴스는 오브젝트 인스턴스로 나타납니다.

  실용적 및 성능상의 이유로, 한 번에 하나의 DB2 UDB 인스턴스로부터 성능 정보만 가져올 수 있습니다. 성능 모니터가 표시하는 DB2 UDB 인스턴스는 성능 모니터 프로세스의 db2instance 레지스트리 변수에 의해 관리됩니다. 동시 실행하는 다중 DB2 UDB 인스턴스를 가지고 있으며 둘 이상으로부터 성능 정보를 보기 원한다면, 모니터될 각 DB2 UDB 인스턴스에 대한 적당한 값에 db2instance를 설정하여, 성능 모니터의 별도 세션을 시작해야 합니다.

  파티션된 데이터베이스 시스템이 실행될 때, 한 번에 하나의 DB2 인스턴스로부터 성능 정보만 가져올 수 있습니다. 디폴트로, 디폴트 노드의 성능 정보(즉, 논리 포트 0을 가진 노드)가 표시됩니다. 또다른 노드의 성능 정보를 보려면, 모니터되는 노드의 노드 번호에 설정된 DB2NODE 환경 변수로 성능 모니터의 각 세션을 시작해야 합니다.

- **DB2 UDB 데이터베이스**

  이 오브젝트는 특정 데이터베이스에 대한 정보를 제공합니다. 정보는 현재 활동 중인 각 데이터베이스에 대해 사용 가능합니다.

- **DB2 응용프로그램**

  이 오브젝트는 특정 DB2 UDB 응용프로그램에 대한 정보를 제공합니다. 정보는 현재 활동 중인 각 DB2 UDB 응용프로그램에 대해 사용 가능합니다.

- **DB2 DCS 데이터베이스**

  이 오브젝트는 특정 DCS 데이터베이스에 대한 정보를 제공합니다. 정보는 현재 활동 중인 각 데이터베이스에 대해 사용 가능합니다.

- **DB2 DCS 응용프로그램**

  이 오브젝트는 특정 DB2 DCS 응용프로그램에 대한 정보를 제공합니다. 정보는 현재 활동 중인 각 DB2 DCS 응용프로그램에 대해 사용 가능합니다.

Windows 성능 모니터가 나열하는 오브젝트는 Windows 머신에 무엇이 설치되어 있고 어떤 응용프로그램이 활동 중인지에 따라 다릅니다. 예를 들어, DB2 UDB가 설치되어 있고 데이터베이스 관리 프로그램이 시작되어 있으면, DB2 데이터베이스 관리 프로그램 오브젝트가 나열됩니다. 머신상에서 일부 DB2 UDB 데이터베이스와 응용프로그램도 현재 활동 중이라면, DB2 데이터베이스 및 DB2 응용프로그램 오브젝트가 마찬가지로 나열됩니다. Windows 시스템을 DB2 Connect 게이트웨이로 사용하고 몇몇 DCS 데이터베이스 및 응용프로그램이 현재 활동 중이면, DB2 DCS 데이터베이스 및 DB2 DCS 응용프로그램 오브젝트가 나열됩니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

리모트 DB2 UDB 성능 정보 액세스

프로시저:

DB2 성능 정보에 대한 리모트 액세스 사용에 대해서는 이미 앞에서 언급되었습니다. **추가 대상(Add to)** 상자에서 또다른 컴퓨터를 선택하여 모니터하십시오. 해당 컴퓨터에서 사용 가능한 모든 성능 오브젝트의 목록이 표시됩니다.

리모트 컴퓨터의 DB2 성능 오브젝트를 모니터하기 위해서는 해당 컴퓨터에 설치되어 있는 DB2 UDB 또는 DB2 Connect 코드의 레벨이 버전 6 이상이어야 합니다.

관련 개념:

- 435 페이지의 『Windows 성능 모니터 소개』

DB2 성능 값 재설정

프로시저:

응용프로그램이 DB2 모니터 API를 호출할 때, 리턴된 정보는 일반적으로 DB2 Universal Database™(DB2 UDB) 서버가 시작된 이후에 누적된 값입니다. 그러나 종종 다음을 수행하는 데 유용합니다.

- 성능 값 재설정
- 테스트 실행
- 값을 다시 재설정
- 테스트 다시 실행

데이터베이스 성능 값을 재설정하려면, **db2perfc** 프로그램을 사용하십시오. 다음을 입력하십시오.

```
db2perfc
```

디폴트로, 이 명령은 활동 중인 모든 DB2 UDB 데이터베이스에 대한 성능 값을 재설정합니다. 또한, 재설정하려는 데이터베이스의 목록을 지정할 수도 있습니다. DCS 데이터베이스에 대한 성능 값이 재설정되어야 한다는 것을 지정하기 위해 -d 옵션을 사용할 수도 있습니다. 예를 들어, 다음과 같습니다.

```
db2perfc
db2perfc dbalias1 dbalias2 ... dbaliasn

db2perfc  -d
db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

첫 번째 예는 활동 중인 모든 DB2 UDB 데이터베이스에 대한 성능 값을 재설정합니다. 그 다음 예는 특정 DB2 UDB 데이터베이스에 대한 값을 재설정합니다. 세 번째 예는 활동 중인 모든 DB2 DCS 데이터베이스에 대한 성능 값을 재설정합니다. 마지막 예는 특정 DB2 DCS 데이터베이스에 대한 값을 재설정합니다.

db2perfc 프로그램은 관련 DB2 UDB 서버 인스턴스에 대한 데이터베이스 성능 정보(즉, **db2perfc**를 실행하는 세션의 DB2INSTANCE에 들어 있는)에 현재 액세스 중인 모든 프로그램에 대한 값을 재설정합니다.

또한 **db2perfc**를 호출하여, **db2perfc** 명령이 실행될 때 DB2 UDB 성능 정보를 리모트 액세스할 때 표시되는 값을 재설정합니다.

주: 응용프로그램이 전역이 아닌 로컬로 보는 특정 데이터베이스의 데이터를 재설정할 수 있도록 하는 DB2 UDB API, sqlmrset가 있습니다.

관련 참조:

- *Administrative API Reference*의 『db2ResetMonitor - Reset Monitor』
- *Command Reference*의 『db2perfc - Reset Database Performance Values Command』

부록 H. Windows 데이터베이스 파티션 서버 사용

Windows 환경에서 사용자 구성의 특성을 변경하기 위해 작업할 때, 관련된 태스크는 특정 유틸리티를 사용하여 수행됩니다.

여기에 표시된 유틸리티는 다음과 같습니다.

- 『인스턴스의 데이터베이스 파티션 서버 나열』
- 442 페이지의 『인스턴스에 데이터베이스 파티션 서버 추가(Windows)』
- 443 페이지의 『데이터베이스 파티션 변경(Windows)』
- 445 페이지의 『인스턴스에서 데이터베이스 파티션 삭제(Windows)』

인스턴스의 데이터베이스 파티션 서버 나열

프로시저:

Windows에서는 **db2nlist** 명령을 사용하여 인스턴스에 참여하는 데이터베이스 파티션 서버 목록을 확보하십시오.

명령은 다음과 같이 사용됩니다.

```
db2nlist
```

표시된 대로 이 명령을 사용하면, 디폴트 인스턴스는 현재 인스턴스가 됩니다 (DB2INSTANCE 환경 변수로 설정). 특정 인스턴스를 지정하려면, 다음을 사용하여 인스턴스를 지정할 수 있습니다.

```
db2nlist /i:instName
```

여기서, `instName`은 원하는 특정 인스턴스 이름입니다.

또한, 다음을 사용하여 선택적으로 각 파티션 서버의 상태를 요청할 수도 있습니다.

```
db2nlist /s
```

각 데이터베이스 파티션 서버의 상태는 시작 중, 실행 중, 중지 중 또는 중지됨 중 하나일 수 있습니다.

태스크 관련:

- 442 페이지의 『인스턴스에 데이터베이스 파티션 서버 추가(Windows)』
- 443 페이지의 『데이터베이스 파티션 변경(Windows)』
- 445 페이지의 『인스턴스에서 데이터베이스 파티션 삭제(Windows)』

인스턴스에 데이터베이스 파티션 서버 추가(Windows)

프로시저:

Windows에서 **db2ncrt** 명령을 사용하여 인스턴스에 데이터베이스 파티션 서버(노드)를 추가하십시오.

주: 인스턴스에 이미 데이터베이스가 있으면, **db2ncrt** 명령을 사용하지 마십시오. 대신, **db2start addnode** 명령을 사용하십시오. 이렇게 하면, 데이터베이스는 새 데이터베이스 파티션 서버에 올바로 추가됩니다. 파일을 변경하면 파티션된 데이터베이스 시스템에 불일치가 생길 수 있으므로 `db2nodes.cfg` 파일을 **편집하지 마십시오.**

명령에는 다음과 같은 필수 매개변수가 있습니다.

```
   db2ncrt /n:node_number
/u:username,password
/p:logical_port
```

- /n:

  데이터베이스 파티션 서버를 식별하기 위한 고유한 노드 번호. 번호는 오름차순으로 1 - 999일 수 있습니다.

- /u:

  DB2 서비스의 로그온 어카운트 이름 및 암호

- /p:logical_port

  논리 포트가 0이 아닐 때 데이터베이스 파티션 서버에 사용되는 논리 포트. 지정되지 않은 경우, 지정된 논리 포트 번호는 0입니다.

논리 포트 매개변수는 머신에서 최초 노드를 작성할 때에만 선택적입니다. 논리 노드를 작성하는 경우, 이 매개변수를 지정하고 사용 중이지 않은 논리 포트 번호를 선택해야 합니다. 여러 제한사항이 있습니다.

- 모든 머신에는 논리 포트 0을 가진 데이터베이스 파티션 서버가 있어야 합니다.
- 포트 번호는 `%SystemRoot%\system32\drivers\etc` 디렉토리에 있는 서비스 파일에 FCM 통신용으로 예약된 포트 범위를 초과할 수 없습니다. 예를 들어, 현재 인스턴스의 네 개의 포트 범위를 예약하는 경우 최대 포트 번호는 3입니다(포트 1, 2 및 3이며, 포트 0은 디폴트 논리 노드입니다). 포트 범위는 **db2icrt**가 `/r:base_port, end_port` 매개변수와 함께 사용될 때 정의됩니다.

또한 여러 개의 선택적 매개변수도 있습니다.

- /g:network_name

  데이터베이스 파티션 서버에 대한 네트워크 이름을 지정합니다. 이 매개변수를 지정하지 않은 경우, DB2는 사용자 시스템에서 검출한 최초 IP 주소를 사용합니다.

머신에 다중 IP 주소가 있으며 데이터베이스 파티션 서버의 특정 IP 주소를 지정하려는 경우, 이 매개변수를 사용합니다. 네트워크 이름 또는 IP 주소를 사용하여 *network_name* 매개변수를 입력할 수 있습니다.

- /h:host_name

  호스트 이름이 로컬 호스트 이름이 아닐 때 내부 통신에 대한 FCM에서 사용되는 TCP/IP 호스트 이름. 이 매개변수는 리모트 머신에서 데이터베이스 파티션 서버를 추가하는 경우 필요합니다.

- /i:instance_name

  인스턴스 이름. 디폴트값은 현재 인스턴스입니다.

- /m:machine_name

  노드가 있는 Windows 워크스테이션의 컴퓨터 이름. 디폴트값은 로컬 머신의 컴퓨터 이름입니다.

- /o:instance_owning_machine

  인스턴스를 소유하는 머신의 컴퓨터 이름. 디폴트값은 로컬 머신입니다. 이 매개변수는 **db2ncrt** 명령이 인스턴스를 소유하는 머신이 아닌 머신에서 호출될 때 필요합니다.

예를 들어, 새로운 데이터베이스 파티션 서버를 인스턴스 소유 머신 MYMACHIN에 있는 인스턴스 TESTMPP에 추가하고(다중 논리 노드를 실행할 수 있도록), 이 새로운 노드를 논리 포트 1을 사용하는 노드 2로서 알려지게 하려는 경우, 다음을 입력하십시오.

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

관련 참조:

- *Command Reference*의 『db2start - Start DB2 Command』
- *Command Reference*의 『db2icrt - Create Instance Command』
- *Command Reference*의 『db2ncrt - Add Database Partition Server to an Instance Command』

데이터베이스 파티션 변경(Windows)

프로시저:

Windows에서 **db2nchg** 명령을 사용하여 다음을 수행하십시오.

- 하나의 머신에서 또다른 머신으로 데이터베이스 파티션을 이동시킵니다.
- 머신의 TCP/IP 호스트 이름을 변경합니다.

다중 네트워크 어댑터를 사용하려고 계획하는 경우, 이 명령을 사용하여 『netname』 필드의 TCP/IP 주소를 *db2nodes.cfg* 파일에 지정해야 합니다.

- 다른 논리 포트 번호를 사용합니다.
- 데이터베이스 파티션 서버(노드)에 대해 다른 이름을 사용합니다.

명령에는 다음과 같은 필수 매개변수가 있습니다.

```
db2nchg /n:node_number
```

매개변수 /n:은 변경하려는 데이터베이스 파티션 서버 구성의 노드 번호입니다. 이 매개변수는 필수입니다.

선택적 매개변수는 다음을 포함합니다.

- /i:instance_name

  이 데이터베이스 파티션 서버가 참여하는 인스턴스를 지정합니다. 이 매개변수를 지정하지 않은 경우, 디폴트값은 현재 인스턴스입니다.

- /u:username,password

  DB2 Universal Database™(DB2 UDB) 서비스의 로그온 어카운트 이름 및 암호를 변경합니다. 이 매개변수를 지정하지 않은 경우, 로그온 어카운트 및 암호는 동일하게 남아 있습니다.

- /p:logical_port

  데이터베이스 파티션 서버에 대한 논리 포트를 변경합니다. 데이터베이스 파티션 서버를 다른 머신으로 이동시키려는 경우 이 매개변수를 지정해야 합니다. 이 매개변수를 지정하지 않은 경우, 논리 포트 번호는 변경되지 않습니다.

- /h:host_name

  내부 통신을 위해 FCM에 의해 사용되는 TCP/IP 호스트 이름을 변경합니다. 이 매개변수를 지정하지 않은 경우, 호스트 이름은 변경되지 않습니다.

- /m:machine_name

  데이터베이스 파티션 서버를 또다른 머신으로 이동시킵니다. 데이터베이스 파티션 서버는 인스턴스에 기존 데이터베이스가 없는 경우에만 이동될 수 있습니다.

- /g:network_name

  데이터베이스 파티션 서버에 대한 네트워크 이름을 변경합니다.

  머신에 다중 IP 주소가 있으며 데이터베이스 파티션 서버의 특정 IP 주소를 사용하려는 경우, 이 매개변수를 사용하십시오. 네트워크 이름 또는 IP 주소를 사용하여 `network_name`을 입력할 수 있습니다.

예를 들어, 인스턴스 TESTMPP에 참여하는 노드 2에 지정된 논리 포트를 변경하고, 논리 포드 3을 사용하려면, 다음과 같은 명령을 입력하십시오.

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 UDB는 리모트 머신에 있는 인스턴스 레벨의 DB2 UDB 레지스트리 변수에 액세스할 수 있는 능력을 제공합니다. 현재 DB2 UDB 레지스트리 변수는 세 가지 레벨(머신 또는 전역 레벨, 인스턴스 레벨 및 노드 레벨)에 저장됩니다. DB2REMOTEPREG를 사용하면 인스턴스 레벨(노드 레벨 포함)에 저장된 레지스트리 변수를 또다른 머신으로 경로 재지정할 수 있습니다. DB2REMOTEPREG를 설정하면 DB2 UDB는 DB2REMOTEPREG가 가리키는 머신에서 DB2 UDB 레지스트리 변수에 액세스합니다. db2set 명령은 다음과 같이 사용됩니다.

```
db2set DB2REMOTEPREG=<remote workstation>
```

여기서, <remote workstation>은 리모트 워크스테이션 이름입니다.

주: 모든 DB2 UDB 인스턴스 프로파일 및 인스턴스 목록은 고유 리모트 머신 이름에 위치하므로 이 옵션을 설정할 때에는 주의해야 합니다.

이들 기능을 설정 DBINSTPROF와 함께 사용하여 레지스트리를 담고 있는 동일한 머신의 리모트 LAN 드라이브를 가리키도록 할 수 있습니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

관련 참조:

- *Command Reference*의 『db2nchg - Change Database Partition Server Configuration Command』

인스턴스에서 데이터베이스 파티션 삭제(Windows)

프로시저:

Windows에서 데이터베이스가 없는 인스턴스에서 데이터베이스 파티션 서버(노드)를 삭제하려면 **db2ndrop** 명령을 사용하십시오. 데이터베이스 파티션 서버를 제거하면, 노드 번호는 새 데이터베이스 파티션 서버에서 재사용될 수 있습니다.

인스턴스에서 데이터베이스 파티션 서버를 제거할 때에는 주의하십시오. 인스턴스에서 인스턴스를 소유하는 데이터베이스 파티션 서버 노드 0을 제거하면, 인스턴스는 사용할 수 없게 됩니다. 인스턴스를 제거하려면, **db2idrop** 명령을 사용하십시오.

주: 이 인스턴스에 데이터베이스가 있으면, **db2ndrop** 명령을 사용하지 마십시오. 대신, **db2stop drop nodenum** 명령을 사용하십시오. 이렇게 하면, 데이터베이스는 데

이터베이스 파티션에서 올바로 제거됩니다. 파일을 변경하면 파티션된 데이터베이스 시스템에 불일치가 생길 수 있으므로 db2nodes.cfg 파일을 **편집하지 마십시오.**

다중 논리 노드를 실행 중인 머신에서 논리 포트 0을 지정한 노드를 삭제하려는 경우, 논리 포트 0에 지정된 노드를 삭제하기 전에 다른 논리 포트에 지정된 다른 모든 노드를 삭제해야 합니다. 각 데이터베이스 파티션 서버는 논리 포트 0에 지정된 노드를 가져야 합니다.

명령에는 다음과 같은 매개변수가 있습니다.

```
db2ndrop /n:node_number /i:instance_name
```

- /n:

  데이터베이스 파티션 서버를 식별하기 위한 고유한 노드 번호. 이것은 필수 매개변수입니다. 번호는 오름차순으로 0 - 999일 수 있습니다. 노드 0은 인스턴스를 소유하는 머신를 나타낸다는 것을 상기하십시오.

- /i:instance_name

  인스턴스 이름. 이 매개변수는 선택적입니다. 인스턴스 이름을 제공하지 않은 경우, 디폴트값은 현재 인스턴스(DB2INSTANCE 레지스트리 변수로 설정)입니다.

관련 개념:

- *관리 안내서: 성능*의 『DB2 레지스트리 및 환경 변수』

관련 참조:

- *Command Reference*의 『db2stop - Stop DB2 Command』
- *Command Reference*의 『db2idrop - Remove Instance Command』
- *Command Reference*의 『db2ndrop - Drop Database Partition Server from an Instance Command』

부록 I. 다중 논리 노드 구성

다중 논리 노드를 사용해야 하는 경우

일반적으로 각 머신에 하나의 데이터베이스 파티션 서버가 지정되도록 DB2® Universal Database (DB2 UDB) Enterprise Server Edition을 구성합니다. 그러나 동일한 머신에서 실행 중인 여러 데이터베이스 파티션 서버를 갖는 것이 유리한 여러 상황이 있습니다. 이것은 구성에 머신보다 더 많은 노드가 들어 있을 수 있음을 의미합니다. 이 경우에, *동일한 인스턴스에 참여하는 경우 다중 논리 노드*를 머신이 실행 중이라고 합니다. 다른 인스턴스에 참여하는 경우, 이 머신은 다중 논리 노드를 호스트하고 있지 *않습니다*.

다중 논리 노드 지원을 사용하여, 이러한 유형의 구성에서 선택할 수 있습니다.

- 각 머신이 오직 하나의 데이터베이스 파티션 서버를 갖는 표준 구성
- 머신이 둘 이상의 데이터베이스 파티션 서버를 갖는 다중 논리 노드 구성
- 다중 논리 노드가 각 여러 머신에서 실행하는 구성

다중 논리 노드를 사용하는 구성은 대칭 멀티프로세서(SMP) 아키텍처를 갖는 머신에서 시스템이 쿼리를 실행할 때 유용합니다. 머신에서 다중 논리 노드를 구성하는 기능은 머신이 실패할 경우 유용합니다. 머신이 실패하는 경우(데이터베이스 파티션 또는 서버가 실패하도록 함), DB2START NODENUM 명령을 사용하여 또다른 머신에서 데이터베이스 파티션 서버를 재시작할 수 있습니다. 이것은 사용자 데이터를 사용 가능하게 합니다.

또다른 이점은 다중 논리 노드가 SMP 하드웨어 구성을 활용할 수 있다는 것입니다. 그리고 데이터베이스 파티션은 더 작으므로, 데이터베이스 파티션 및 테이블 스페이스를 백업 및 리스토어하고, 인덱스를 작성하는 것과 같은 태스크를 수행할 때 더 나은 성능을 얻을 수 있습니다.

태스크 관련:

- 448 페이지의 『다중 논리 노드 구성』

관련 참조:

- *Command Reference*의 『db2start - Start DB2 Command』

다중 논리 노드 구성

프로시저:

다음 두 가지 방법 중 하나로 다중 논리 노드를 구성할 수 있습니다.

- db2nodes.cfg 파일에 논리 노드(데이터베이스 파티션)를 구성합니다. 그런 다음, DB2START 명령과 연관된 API를 사용하여 논리 노드와 리모트 노드를 모두 시작할 수 있습니다.

  주: Windows NT의 경우, 시스템에 데이터베이스가 없는 경우 *db2ncrt*를 사용하여 노드를 추가하거나, 하나 이상의 데이터베이스가 있는 경우 DB2START ADDNODE 명령을 사용하여 노드를 추가해야 합니다. Windows NT 내에서 *db2nodes.cfg* 파일은 수동으로 편집될 수 없습니다.
- 다른 논리 데이터베이스 파티션(노드)이 이미 실행 중인 또다른 프로세서에서 논리 노드를 재시작합니다. 이 방법을 사용하면 db2nodes.cfg에서 논리 데이터베이스 파티션에 대해 지정된 호스트 이름과 포트 번호를 겹쳐쓸 수 있습니다.

db2nodes.cfg에 논리 데이터베이스 파티션(노드)을 구성하려면, 해당 노드에 논리 포트 번호를 할당하는 항목을 파일에 작성해야 합니다. 다음 구문을 사용해야 합니다.

```
nodenumber hostname logical-port netname
```

주: Windows NT의 경우, 시스템에 데이터베이스가 없는 경우 *db2ncrt*를 사용하여 노드를 추가하거나, 하나 이상의 데이터베이스가 있는 경우 DB2START ADDNODE 명령을 사용하여 노드를 추가해야 합니다. Windows NT 내에서 *db2nodes.cfg* 파일은 수동으로 편집될 수 없습니다.

Windows NT에서의 *db2nodes.cfg* 파일의 형식은 Unix에서의 동일한 파일과 비교할 때 다릅니다. Windows NT에서 컬럼 형식은 다음과 같습니다.

```
nodenumber hostname computername logical_port netname
```

hostname에 완전한 이름을 사용하십시오. /etc/hosts 파일 또한 완전한 이름을 사용해야 합니다. db2nodes.cfg 파일 및 /etc/hosts 파일에서 완전한 이름을 사용하지 않을 경우, SQL30082N RC=3 오류 메시지를 받을 수도 있습니다.

FCM 통신의 etc 디렉토리의 services 파일에 충분한 포트를 정의했는지 확인해야 합니다.

관련 개념:

- 447 페이지의 『다중 논리 노드를 사용해야 하는 경우』

태스크 관련:

- 178 페이지의 『노드 및 데이터베이스 구성 파일 변경』

- 40 페이지의『노드 구성 파일 작성』

관련 참조:

- *Command Reference*의『db2start - Start DB2 Command』
- *Command Reference*의『db2ncrt - Add Database Partition Server to an Instance Command』

부록 J. 제어 센터 확장

제어 센터에 플러그인 아키텍처 도입

새 *플러그인* 아키텍처를 사용하여 추가 기능을 제공함으로써 DB2 Universal Database 제어 센터를 확장할 수 있습니다.

플러그인 아키텍처의 개념은 제어 센터 팝업 메뉴에서 주어진 오브젝트에 대한 항목을 추가하고 제어 센터 트리에 오브젝트를 추가하며 도구 모음에 새 단추를 추가할 수 있는 능력을 제공하는 것입니다. 사용자가 구현해야 하는 Java™ 인터페이스 세트가 이 도구와 함께 제공됩니다. 이 인터페이스는 제어 센터에 포함될 추가 조치를 전달하기 위해 사용됩니다.

플러그인 확장 파일(db2plug.zip)은 제어 센터 도구가 시동될 때 로드됩니다. 이 ZIP 파일의 *크기*에 따라 도구의 시동 시간이 길어질 수 있습니다. 그러나 플러그인 ZIP 파일이 대부분의 사용자에게 작을 것이고 충격이 최소한일 것입니다.

관련 개념:

- 452 페이지의 『샘플 플러그인 컴파일 및 실행』
- 454 페이지의 『제어 센터 확장으로서 플러그인 작성』
- 451 페이지의 『제어 센터 플러그인 개발자에 대한 지시사항』

제어 센터 플러그인 개발자에 대한 지시사항

db2plug.zip 파일에는 여러 플러그인이 포함될 수 있으므로, 플러그인 개발자는 제어 센터용 플러그인을 작성할 때 다음 지시사항을 따라야 합니다.

- Java™ 패키지를 사용하여 플러그인 클래스가 고유한 이름을 가지도록 하십시오. Java 패키지 이름 지정 규칙을 따르십시오. 인터넷 도메인의 거꾸로 된 이름을 패키지 이름의 접두부로 사용하십시오(예: com.companyname). 모든 패키지 이름 또는 적어도 고유한 접두부만이라도 소문자를 사용해야 합니다.
- db2plug.zip은 `sqllib` 디렉토리 아래의 도구 디렉토리에 설치해야 합니다. V8 이전인 경우에는 `sqllib` 디렉토리 아래의 `cc` 디렉토리에 db2plug.zip을 설치해야 합니다.
- 제어 센터용 플러그인을 작성할 때 db2plug.zip 파일이 이미 있으면 기존 db2plug.zip에 플러그인 클래스를 추가해야 합니다. 사용자 고유의 db2plug.zip 파일로 기존 db2plug.zip 파일을 겹쳐쓰면 안됩니다. 기존 db2plug.zip에 플러그인을 추가하려면 다음과 같은 zip 명령을 사용해야 합니다.

```
zip -r0 db2plug.zip com\companyname\myplugin\*.class
```

여기서 사용자의 플러그인 패키지 이름은 com.companyname.myplugin입니다.

- db2plug.zip에 있는 모든 클래스는 제어 센터가 시작될 때 로드됩니다. db2plug.zip 파일에는 모든 CCExtension 클래스 파일 및 com.ibm.db2.tools.cc.navigator 패키지의 클래스를 확장하거나 구현하는 클래스가 포함되어야 합니다. 이들 클래스가 직접 사용하지 않는 다른 클래스는 db2plug.zip에 포함될 필요가 없습니다. 이러한 클래스 파일은 제어 센트가 시작될 때 성능에 미치는 영향을 최소화하기 위해 별도의 jar 파일에 저장할 수 있습니다. 이 조치는 추가 클래스의 수가 많은 경우에 좋습니다. jar 파일은 sqllib 디렉토리 아래의 tools디렉토리에 두어야 합니다. **db2cc** 명령을 사용하여 제어 센터를 시작할 때 jar 파일이 자동으로 *classpath*에 포함됩니다.
- CCObject를 구현하는 플러그인 클래스는 인수가 없는 디폴트 컨스트럭터를 제공하여 제어 센터에서 Class.newInstance()를 호출할 수 있도록 합니다.
- 가능하면 내부 클래스를 사용하지 마십시오. 일반적으로 CCTreeObject를 구현하여 제어 센터에서 새 플러그인 오브젝트를 작성하는 플러그인 클래스는 내부 클래스로 선언해서는 안됩니다. 그러면 제어 센터가 이들 클래스를 인스턴스화하지 못하게 됩니다.
- **db2cc -tf filename**을 사용하여 플러그인이 올바로 로드되었는지 테스트하십시오. 지정된 파일 이름에 제어 센터 추적 정보를 넣습니다. 전체 경로 이름을 제공하지 않는 경우, 추적 파일은 sqllib의 tools 디렉토리에 기록됩니다. 플러그인 관련 추적 명령문에는 단어 『Plugin』이 포함됩니다. 텍스트 『PluginLoader』가 포함된 행을 찾아보고 클래스가 로드되었는지 확인할 수 있습니다.

관련 개념:

- 452 페이지의 『샘플 플러그인 컴파일 및 실행』
- 454 페이지의 『제어 센터 확장으로서 플러그인 작성』

관련 참조:

- *Command Reference*의 『db2cc - Start Control Center Command』

샘플 플러그인 컴파일 및 실행

제어 센터 플러그인 기능은 다음 절과 해당 플러그인 샘플 프로그램 Example1.java, Example2.java, Example3.java, Example3Folder.java 및 Example3Child.java에 자세히 설명되어 있습니다. 이러한 샘플 java 파일은 DB2® Application Development 클라이언트와 함께 설치됩니다. Windows® 플랫폼에서 이러한 샘플 프로그램은 DRIVE:\sqllib\samples\java\plugin에 있으며, 여기서, DRIVE:는 DB2가 설치된 드라이브를 나타냅니다. UNIX® 플랫폼에서 이러한 샘플은/u/db2inst1/sqllib/samples/java/plugin에 있으며, 여기서, /u/db2inst1은 DB2가 설치된 디렉토리를 나타냅니다.

주: 플러그인 샘플 프로그램에는 아직 여기에 반영되지 않은 갱신사항이 포함될 수 있습니다. 코드 예와 Java 문서는 여기에 표시된 내용과 차이가 있는 경우에 가장 최신 정보로 간주되어야 합니다.

샘플 플러그인을 실행하려면 Java™ 아카이브 파일의 규칙에 따라 ZIP해야 합니다. ZIP 파일(db2plug.zip)은 *classpath*에 있어야 합니다. Windows 운영 체제에서는 db2plug.zip을 DRIVE:\sqllib\tools 디렉토리에 두십시오. 여기서, DRIVE:는 DB2가 설치된 드라이브를 나타냅니다. UNIX 플랫폼에서는 db2plug.zip을 /u/db2inst1/sqllib/tools 디렉토리에 두십시오. 여기서, /u/db2inst1은 DB2가 설치된 디렉토리를 나타냅니다.

주: **db2cc** 명령은 도구 디렉토리에 있는 db2plug.zip을 가리키도록 *classpath*를 설정합니다.

샘플(함께 움직이는 Example3, Example3Folder 및 Example3Child는 제외)은 서로 충돌할 수도 있으므로, 동일한 db2plug.zip으로 ZIP 압축해서는 안됩니다.

이러한 샘플 Java 파일을 컴파일하려면 *classpath*에 다음을 포함시켜야 합니다.

- Windows 플랫폼에서는 다음을 사용하십시오.
  - DRIVE: \sqllib\java\Common.jar
  - DRIVE: \sqllib\tools\db2navplug.jar

  여기서, DRIVE는 DB2가 설치된 드라이브를 나타냅니다.
- UNIX 플랫폼에서는 다음을 사용하십시오.
  - /u/db2inst1/sqllib/java/Common.jar
  - /u/db2inst1/sqllib/tools/db2navplug.jar

  여기서, /u/db2inst1은 DB2가 설치된 디렉토리를 나타냅니다.

샘플 Java 파일의 컴파일로 생성된 모든 클래스를 포함하는 db2plug.zip을 작성하십시오. 이 파일을 압축해서는 안됩니다. 예를 들어, 다음 명령을 발행하십시오.

```
zip -r0 db2plug.zip *.class
```

이 명령은 모든 클래스 파일을 db2plug.zip 파일에 위치시키며 상대 경로 정보를 보존합니다.

관련 개념:

- 454 페이지의 『제어 센터 확장으로서 플러그인 작성』
- 451 페이지의 『제어 센터 플러그인 개발자에 대한 지시사항』

관련 참조:

- *Command Reference*의 『db2cc - Start Control Center Command』

제어 센터 확장으로서 플러그인 작성

플러그인을 작성하는 첫 번째 단계는 CCExtension 인터페이스를 구현하는 클래스를 정의하는 것입니다. 이 클래스에는 제어 센터에 의해 로드되는 플러그인 클래스 목록이 포함됩니다. 데이터베이스 및 테이블과 같은 표준 제어 센터 오브젝트에 메뉴 항목을 추가하거나 트리에 표시할 사용자 고유의 오브젝트를 작성하고자 할 경우에는, CCObject 인터페이스를 구현하고 getObjects 메소드에서 이러한 CCObject의 배열을 리턴하는 클래스를 작성합니다. 도구 모음 단추를 추가하려면 CCToolbarAction을 구현하고 getToolbarActions 메소드에서 CCToolbarAction 배열을 리턴합니다.

이러한 인터페이스는 각각 다음에 문서화되어 있습니다.

- Windows® 플랫폼의 경우, `DRIVE:\sqllib\samples\java\plugin\doc`. 여기서, `DRIVE:`는 DB2®가 설치된 드라이브를 나타냅니다.
- UNIX® 플랫폼의 경우, `/u/db2inst1/sqllib/samples/java/plugin/doc`. 여기서, `/u/db2inst1`은 DB2가 설치된 디렉토리를 나타냅니다.

태스크 관련:

- 455 페이지의 『도구 모음 단추를 추가하는 플러그인 작성』
- 456 페이지의 『기본 메뉴 조치 작성』
- 458 페이지의 『메뉴 항목 위치 지정』
- 459 페이지의 『기본 메뉴 조치 구분자 작성』
- 459 페이지의 『서브메뉴 작성』
- 460 페이지의 『특정 이름을 가지는 오브젝트에만 메뉴 항목 추가』
- 461 페이지의 『트리에 다중 오브젝트를 포함하는 폴더 추가』
- 463 페이지의 『폴더에 예제 오브젝트 추가』
- 465 페이지의 『플러그인 트리 오브젝트에 대한 속성 설정』
- 467 페이지의 『작성 조치 추가』
- 469 페이지의 『다중 선택 지원과 함께 제거 조치 추가』
- 470 페이지의 『변경 조치 추가』
- 472 페이지의 『isConfigurable()로 구성 기능 사용 불가능화』
- 472 페이지의 『isEditable()을 사용하여 오브젝트 변경 기능 사용 불가능화』
- 472 페이지의 『hasConfigurationDefaults()를 사용하여 구성 대화 상자에서 디폴트 단추 사용 불가능화』

플러그인 태스크 설명

다음과 같은 플러그인 태스크를 설명합니다.

1. 도구 모음 단추를 추가하는 플러그인 작성
2. 새 메뉴 항목을 데이터베이스 오브젝트에 추가하는 플러그인 작성
3. 트리의 데이터베이스 아래에 플러그인 오브젝트를 추가하는 플러그인 작성
4. isConfigurable()로 구성 기능 사용 불가능화
5. isEditable()을 사용하여 오브젝트 변경 기능 사용 불가능화
6. hasConfigurationDefaults()를 사용하여 구성 대화 상자에서 디폴트 단추 사용 불가능화

도구 모음 단추를 추가하는 플러그인 작성

프로시저:

이 예에서는 단지 도구 모음 단추를 추가하므로 getObjects는 널(NULL) 배열을 리턴합니다.

```
import com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import javax.swing.*;

public class Example1 implements CCExtension {

  public CCObject[] getObjects () {
      return null;
  }

}
```

com.ibm.db2.tools.cc.navigator 패키지가 임포트됨에 유의하십시오. 이 클래스는 CCToolbarAction 인터페이스를 구현하며, 이 인터페이스는 세 개의 메소드 getHoverHelpText, getIcon 및 actionPerformed를 구현해야 합니다. 제어 센터는 getHoverHelpText를 사용하여 사용자가 도구 모음 단추 위에 마우스를 놓을 때 나타나는 작은 텍스트 상자를 표시합니다. getIcon을 사용하여 단추에 대한 아이콘을 지정합니다. 사용자가 단추를 누를 때 제어 센터는 actionPerformed를 호출합니다. 다음은 사용자가 누를 때 콘솔에 메시지를 작성하는 X라는 단추를 추가하는 예입니다. 이 단추는 제어 센터의 이미지 저장소 클래스에 있는 새로 고침 아이콘을 사용합니다.

```
class Example1ToolbarAction implements CCToolbarAction {

  public String getHoverHelpText() { return "X"; }

  public ImageIcon getIcon() {
   return CommonImageRepository.getCommonIcon(CommonImageRepository.WC_NV_
REFRESH);
  }
```

```
  public void actionPerformed(ActionEvent e) {
   System.out.println("I've been clicked");
  }

}
```

최종 단계는 Example1의 getToolbarActions 메소드를 구현하여 새 클래스의 인스턴스를 리턴하는 것입니다.

```
  public CCToolbarAction[] getToolbarActions () {
   return new CCToolbarAction[] { new Example1ToolbarAction() };
  }
```

관련 개념:

• 452 페이지의『샘플 플러그인 컴파일 및 실행』

새 메뉴 항목을 데이터베이스 오브젝트에 추가하는 플러그인 작성

다음 프로시저에서는 새 메뉴 항목을 데이터베이스 오브젝트에 추가하는 플로그인 작성 방법을 개략적으로 설명합니다.

1. 기본 메뉴 조치 작성
2. 메뉴 항목 위치 지정
3. 기본 메뉴 조치 구분자 작성
4. 서브메뉴 작성
5. 특정 이름을 가진 오브젝트에만 메뉴 항목 추가

기본 메뉴 조치 작성

프로시저:

조금 더 고급인 이 주제에서는 데이터베이스 오브젝트의 팝업 메뉴에 새 명령을 추가합니다.

예 1에서와 같이, 첫 번째 단계는 CCExtension을 확장하는 클래스를 작성합니다.

```
import com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import javax.swing.*;

public class Example2 implements CCExtension {

  public CCToolbarAction[] getToolbarActions () {
      return null;
  }

}
```

두 번째 단계는 다음과 같이 트리에 데이터베이스 오브젝트에 대한 CCObject를 작성하는 것입니다.

```
class CCDatabase implements CCObject {

  public String getName () { return null; }
  public boolean isEditable () { return true; }
  public boolean isConfigurable () { return true; }

  public int getType () { return UDB_DATABASE; }
}
```

제어 센터 내장 오브젝트(이 예에서는 데이터베이스 오브젝트)에 메뉴 항목을 추가하는 기능 이외의 다른 기능을 사용하지 않으므로, 대부분의 함수를 널(NULL) 또는 true를 리턴하는 것으로 구현합니다. 이 오브젝트가 DB2 UDB 데이터베이스 오브젝트를 나타내도록 지정하기 위해 CCObject에서 해당 유형을 상수인 UDB_DATABASE로 지정합니다. 이 예에서는 클래스의 이름을 CCDatabase로 지정하나, 동일한 zip 파일에 사용자 플러그인과 동일한 다른 벤더의 플러그인이 있을 수 있으므로 가능한 한 고유한 클래스 이름을 사용해야 합니다. Java 패키지를 사용하면 고유한 클래스 이름을 지정하는데 도움이 됩니다.

CCExtension의 getObjects 메소드는 다음과 같이 CCDatabase의 인스턴스를 포함하는 배열을 리턴합니다.

```
public CCObject[] getObjects () {
  return new CCObject[] { new CCDatabase() };
}
```

유형이 UDB_DATABASE인 다중 CCObject 서브클래스를 작성할 수 있으나, isEditable 또는 isConfigurable 메소드에서 리턴되는 값이 충돌할 경우, false를 리턴하는 오브젝트가 true를 리턴하는 오브젝트를 겹쳐씁니다.

구현해야 할 나머지 메소드는 getMenuActions입니다. 이것은 CCMenuAction 배열을 리턴하므로, 먼저 이러한 인스턴스를 구현하는 클래스를 작성합니다.

getMenuText 및 actionPerformed 메소드를 구현해야 합니다. 메뉴에 표시되는 텍스트는 getMenuText를 사용하여 확보합니다. 사용자가 메뉴 항목을 누르면, 트리거되는 이벤트가 actionPerformed를 호출합니다.

다음 클래스 예는 단일 데이터베이스 오브젝트를 선택할 때 ″Example2a Action″이라는 메뉴 항목을 표시합니다. 사용자가 이 메뉴 항목을 누르면, 콘솔에 ″Example2a menu item actionPerformed″ 메시지가 표시됩니다.

```
class Example2AAction implements CCMenuAction {

  public String getMenuText () { return "Example2a Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2a menu item actionPerformed");
  }

}
```

마지막으로 CCObject에 다음을 추가하여 UDB Database CCObject에 이 메뉴 항목을 첨부하십시오.

```
public CCMenuAction[] getMenuActions () {
   return new CCMenuAction[] { new Example2AAction() };
}
```

관련 개념:

- 452 페이지의『샘플 플러그인 컴파일 및 실행』

태스크 관련:

- 458 페이지의『메뉴 항목 위치 지정』
- 459 페이지의『기본 메뉴 조치 구분자 작성』
- 459 페이지의『서브메뉴 작성』
- 460 페이지의『특정 이름을 가지는 오브젝트에만 메뉴 항목 추가』

메뉴 항목 위치 지정

프로시저:

기본 메뉴 항목 작성시, 메뉴 내에서 메뉴 항목의 위치를 지정하지 않았습니다. 메뉴에 플러그인 메뉴 항목을 추가할 때의 디폴트 동작은 이들을 끝에(단, 새로 고침 및 필터 메뉴 항목 앞에) 추가하는 것입니다.

0에서부터 메뉴에 있는 항목 수(새로 고침 및 필터 메뉴 항목을 포함하지 않은)까지의 위치 번호를 지정하기 위해 이러한 동작을 겹쳐쓸 수 있습니다. 다음과 같이 CCMenuAction 서브클래스를 변경하여 Positionable을 구현한 다음 getPosition 메소드를 구현하십시오.

```
class Example2BAction implements CCMenuAction, Positionable {

  public String getMenuText () { return "Example2B Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2B menu item actionPerformed");
  }

  public int getPosition() {
     return 0;
  }
}
```

위치 번호 0을 지정하면 메뉴 항목이 목록에서 첫 번째 위치에 놓이고 플러그인 메뉴 항목을 제외한 항목 수와 동일한 위치 번호를 지정하면 메뉴 항목이 맨 아래(단, 새로 고침 및 필터메뉴 항목 앞에)에 놓입니다. 디폴트 동작이 수행되도록, 즉 메뉴 항목이 맨 아래의 새로 고침 및 필터 메뉴 항목 앞에 놓이도록 Positionable.POSITION_BOTTOM 값을 지정할 수도 있습니다. 메뉴 항목이 POSITION_BOTTOM에 놓이는 UDB_DATABASE 유형의 CCObject 오브젝트가 둘

이상이면, UDB_DATABASE 유형의 CCObject가 CCExtension에 있는 getObjects 오브젝트로부터 리턴되는 순서에 따라 메뉴 항목의 순서가 결정됩니다.

다음과 같이 CCDatabase를 변경하여 메뉴에 Example2BAction을 추가하십시오.

```
    public CCMenuAction[] getMenuActions () {
      return new CCMenuAction[] { new Example2AAction(),
                                  new Example2BAction() };
  }
```

태스크 관련:

- 456 페이지의 『기본 메뉴 조치 작성』
- 459 페이지의 『기본 메뉴 조치 구분자 작성』
- 459 페이지의 『서브메뉴 작성』
- 460 페이지의 『특정 이름을 가지는 오브젝트에만 메뉴 항목 추가』

기본 메뉴 조치 구분자 작성

프로시저:

구분자를 추가하려면 구분자 인터페이스를 구현하는 CCMenuAction을 작성하십시오. 다른 모든 메소드(Positionable을 구현할 경우에는 getPosition 제외)는 무시됩니다.

```
  class Example2CSeparator implements CCMenuAction, Separator, Positionable {

    public String getMenuText () { return null; }

    public void actionPerformed (ActionEvent e) {}

    public int getPosition() {
       return 1;
    }
  }

    public CCMenuAction[] getMenuActions () {
      return new CCMenuAction[] { new Example2AAction(),
                                    new Example2BAction(),
                                  new Example2CSeparator() };
  }
```

태스크 관련:

- 456 페이지의 『기본 메뉴 조치 작성』
- 458 페이지의 『메뉴 항목 위치 지정』
- 459 페이지의 『서브메뉴 작성』
- 460 페이지의 『특정 이름을 가지는 오브젝트에만 메뉴 항목 추가』

서브메뉴 작성

프로시저:

서브메뉴는 단지 CCMenuAction의 배열입니다. 메뉴 항목에 서브메뉴를 포함시키려면 SubMenuParent 인터페이스를 구현해야 합니다. 그런 다음 각 서브메뉴 항목에 대한 CCMenuAction을 구현하여 SubMenuParent 인터페이스의 getSubMenuActions 메소드로부터 이들을 배열로서 리턴해야 합니다. 비 플러그인 서브메뉴로의 메뉴 항목 추가는 지원되지 않습니다. 또한 SubMenuParent는 제어 센터로부터 ActionEvent를 수신하지 않습니다. 다음은 한 예입니다.

```
class Example2DAction implements CCMenuAction, SubMenuParent {

  public String getMenuText () { return "Example2D Action"; }

  public void actionPerformed (ActionEvent e) {}

  public CCMenuAction[] getSubMenuActions() {
     return new CCMenuAction[] { new Example2DSubMenuAction() };
  }

}

class Example2DSubMenuAction implements CCMenuAction {

  public String getMenuText () { return "Example2D Sub-Menu Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2D sub-menu menu item actionPerformed");
  }

}
```

한 번 더 이 새 메뉴 항목을 CCDatabase에 추가하십시오.

```
  public CCMenuAction[] getMenuActions () }
    return new CCMenuAction[] { new Example2AAction(),
                                new Example2BAction(),
                                new Example2CSeparator(),
                                new Example2DAction() };
  }
```

태스크 관련:

- 456 페이지의 『기본 메뉴 조치 작성』
- 458 페이지의 『메뉴 항목 위치 지정』
- 459 페이지의 『기본 메뉴 조치 구분자 작성』
- 460 페이지의 『특정 이름을 가지는 오브젝트에만 메뉴 항목 추가』

특정 이름을 가지는 오브젝트에만 메뉴 항목 추가

프로시저:

현재로선 제어 센터에 표시하는 모든 데이터베이스가 사용자가 작성한 플러그인 메뉴 항목을 표시합니다. CCDatabase의 getName 메소드에서 특정 이름을 리턴함으로써 이러한 메뉴 항목을 해당 이름을 가지는 데이터베이스로만 제한할 수 있습니다. 이름은

완전한 이름이어야 합니다. 이 경우에는 데이터베이스를 참조하므로, getName 메소드에서 리턴하는 것에 시스템, 인스턴스 및 데이터베이스 이름을 포함시켜야 합니다. 이들 이름은 " - "로 구분됩니다. 다음은 시스템 이름이 MYSYSTEM, 인스턴스 이름이 DB2, 그리고 데이터베이스 이름이 SAMPLE일 경우의 예입니다.

```
class CCDatabase implements CCObject {
  ...
  public String getName () { return "MYSYSTEM - DB2 - SAMPLE"; }
  ...
}
```

태스크 관련:

- 456 페이지의『기본 메뉴 조치 작성』
- 458 페이지의『메뉴 항목 위치 지정』
- 459 페이지의『기본 메뉴 조치 구분자 작성』
- 459 페이지의『서브메뉴 작성』

트리의 데이터베이스 아래에 플러그인 오브젝트를 추가하는 플러그인 작성

다음 프로시저에서는 트리의 데이터베이스 아래에 플러그인 오브젝트를 추가하는 플러그인 작성 방법을 개략적으로 설명합니다.

1. 트리에 다중 오브젝트를 포함하는 폴더 추가
2. 폴더에 예제 오브젝트 추가
3. 플러그인 트리 오브젝트에 대한 속성 설정
4. 작성 조치 추가
5. 제거 조치 추가
6. 변경 조치 추가

트리에 다중 오브젝트를 포함하는 폴더 추가

프로시저:

이 예에서는 제어 센터 트리에서 데이터베이스 아래에 플러그인 오브젝트가 표시되게 하기 위해 CCObject 대신 CCTreeObject를 구현합니다. 먼저 이 오브젝트에 대한 CCTreeObject 구현을 작성해야 합니다. 트리에 다중 오브젝트를 놓아야 할 경우에는, 데이터베이스 아래에 직접 이들 모두를 위치시키기보다는 하나의 폴더를 작성하는 것이 일반적입니다. 다음은 폴더의 초기 버전입니다.

```
public class Example3Folder implements CCTreeObject {
  private String parentName = null;
  public boolean isEditable () { return false; }
  public boolean isConfigurable () { return false; }
  public CCTableObject getChildren () { return null; }
  public void setParentName(String name)
  {
    parentName = name;
```

```
    }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public CCMenuAction[] getMenuActions () { return null; }

    public String getName () { return "Example3 Folder"; }

    public void getData (Object[] data) {
      data[0] = this;
    }

    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }

    public Icon getIcon (int iconState) {
      switch (iconState) {
        case CLOSED_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);

        case OPEN_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_OPEN_
FOLDER);

        default:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);
      }
    }

  }
```

getType이 이제 CCTypeFactory 클래스를 이용함에 주의하십시오. CCTypeFactory의 목적은 제어 센터가 플러그인을 고유한 유형을 가지는 것으로 식별할 수 있도록 두 개의 오브젝트가 동일한 유형 번호를 사용하지 않도록 하는 것입니다. 새 폴더는 내장 CC 오브젝트 유형 중 하나가 아닌 새 유형이므로 작성 가능한 다른 새 유형의 유형 번호와 충돌하지 않고 내장 유형의 유형 번호와도 충돌하지 않는 새 유형 번호를 가져야 합니다.

getIcon 메소드는 사용자가 열린 폴더인지 혹은 닫힌 폴더인지를 알 수 있도록 하는 iconState에 대한 매개변수를 취합니다. 이리하여 위와 같이 사용자 아이콘이 사용자 상태와 일치하도록 할 수 있습니다.

데이터베이스를 선택할 때 폴더가 트리뿐만 아니라 세부사항 보기에도 표시되도록 하기 위해 getData는 해당 값이 플러그인 오브젝트 그 자체인 단일 컬럼을 리턴해야 합니다. getData 메소드는 데이터 배열의 첫 번째 요소에 *this* 참조를 지정합니다. 이로 인해 세부사항 보기의 동일한 컬럼에 아이콘 및 이름이 둘 다 나타날 수 있습니다. 제어 센터는 사용자가 CCTableObject 서브클래스를 리턴하는 것을 알게 될 때 Example3Folder에 있는 getIcon 및 getName을 호출할 수 있음을 압니다.

다음 단계는 다음과 같이 CCDatabase 클래스를 작성하여 CCTreeObject를 구현하고 getChildren 메소드에서 Example3Folder의 인스턴스를 포함하는 CCTableObject 배열을 리턴하는 것입니다.

```
import java.util.*;

  class CCDatabase implements CCTreeObject {
    private String parentName = null;
    private Vector childVector;

    public CCDatabase() {
      childVector = new Vector();
      childVector.addElement(new Example3Folder());
    }

    public CCTableObject[] getChildren() {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }

    public void setParentName(String name)
    {
      parentName = name;
    }

    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public int getType () { return UDB_DATABASE; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
  }
```

관련 개념:

- 452 페이지의 『샘플 플러그인 컴파일 및 실행』

태스크 관련:

- 463 페이지의 『폴더에 예제 오브젝트 추가』
- 465 페이지의 『플러그인 트리 오브젝트에 대한 속성 설정』
- 467 페이지의 『작성 조치 추가』
- 469 페이지의 『다중 선택 지원과 함께 제거 조치 추가』
- 470 페이지의 『변경 조치 추가』

폴더에 예제 오브젝트 추가

프로시저:

첫 번째 단계는 다음과 같이 하위 오브젝트에 대한 CCObject 구현을 작성하는 것입니다.

```
  class Example3Child implements CCTableObject {
    private String parentName = null;
    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
    public void setParentName(String name)
    {
      parentName = name;
    }
    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }
  }
```

그 다음에는 이러한 Exercise3Child 오브젝트의 벡터를 보존하도록 다음과 같이 Example3Folder를 수정하십시오.

```
  public class Example3Folder implements CCObject {
    private String parentName = null;
    private Vector childVector;
    ...

    public Example3Folder() {
      childVector = new Vector();
    }

    ...
    public CCTableObject[] getChildren () {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }
    public void setParentName(String name)
    {
      parentName = name;
    }
    ...
  }
```

간단하게 하기 위해 이 예에서 getChildren은 childVector라고 하는 벡터에 저장된 하위 배열을 리턴합니다.

실제 플러그인은 getChildren이 호출되면 하위를 재구성해야 합니다. 그러면 목록이 마지막으로 표시된 이후로 제어 센터 외부에서 작성되었거나 변경되었을 수 있는 새 하위 오브젝트 또는 변경된 하위 오브젝트가 포함될 수 있는 목록을 새로 고칩니다. 하위를 지속 스토리지에 저장하고 읽으면 유실되지 않습니다.

실제 플러그인에서도 getChildren로 리턴된 하위 목록은 제어 센터 트리에서 이 오브젝트의 상위가 되는 오브젝트에 따라 다릅니다. 상위 정보는 setParentName 메소드에 대한 제어 센터 호출로 제공되는 parentName 문자열에 있습니다.

주: 이 예에서 새로 고침이 트리의 데이터베이스 오브젝트 이상으로부터 제어 센터에서 수행되는 경우 Example3 폴더 아래의 하위 목록이 유실됩니다. 새로 고침을 수행하면 새 Example3Folder가 제어 센터에 의해 구성되기 때문입니다. 이 코드 예가 지속 스토리지에서 하위를 읽으면 하위가 유실되지 않습니다. 예를 간단하게 하기 위해 이를 수행하지 않습니다.

관련 개념:

- 452 페이지의『샘플 플러그인 컴파일 및 실행』

태스크 관련:

- 461 페이지의『트리에 다중 오브젝트를 포함하는 폴더 추가』
- 465 페이지의『플러그인 트리 오브젝트에 대한 속성 설정』
- 467 페이지의『작성 조치 추가』
- 469 페이지의『다중 선택 지원과 함께 제거 조치 추가』
- 470 페이지의『변경 조치 추가』

플러그인 트리 오브젝트에 대한 속성 설정

프로시저:

트리를 플러그인 폴더까지 펼쳐 이 폴더를 선택할 경우, 세부사항 분할창에 아무런 컬럼이 없음을 알게 될 것입니다. 이는 getColumns의 Example3Child 구현이 널(NULL)을 리턴하기 때문입니다. 이를 변경하려면 먼저 몇 개의 CCColumn 구현을 작성하십시오. 차후의 예에서 런타임에 컬럼 중 하나의 값을 변경하는 방법에 대해 설명할 것이며 각 오브젝트에는 절대로 변하지 않는 하나의 컬럼이 있어야 하므로, 여기서는 두 개의 컬럼을 작성합니다. 변하지 않는 컬럼을『Name』이라 하고 변하는 컬럼을『State』라고 합니다.

```
class NameColumn implements CCColumn {
   getName() { return "Name"; }
   getColumnClass { return CCTableObject.class; }
}

class StateColumn implements CCColumn {
   getName() { return "State"; }
   getColumnClass { return String.class; }
}
```

지원되는 클래스 유형에는 CCTableObject 클래스, java.util.Date 클래스 및 Java primitives(예: java.lang.Intger)와 동등한 클래스가 포함됩니다.

이 두 컬럼을 포함하도록 Example3Child의 getColumns 메소드를 변경하십시오.

```
class Example3Child implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

상위 클래스도 동일한 컬럼을 포함하도록 변경해야 합니다.

```
class Example3Folder implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

이제 세부사항 보기에서 각 행에 표시될 값을 설정해야 합니다. getData로 패스되는 오브젝트의 요소를 설정하여 이를 수행하십시오. 각 컬럼의 데이터 클래스는 getColumnClass가 해당 컬럼에 리턴하는 클래스와 일치해야 합니다.

```
class Example3Child implements CCTableObject {
  ...
  private String name;
  private String state;

  public Exampe3Child(String name, String state) {
    this.name = name;
    this.state = state;
  }
  ...
  public void getData (Object[] data) {
    data[0] = this;
    data[1] = state;
  }
  ...
}
```

이 경우, CCTableObject 클래스의 첫 번째 컬럼은 this 값을 가집니다. 이는 제어 센터가 getName이 리턴하는 텍스트와 getIcon이 리턴하는 아이콘을 둘 다 렌더링할 수 있도록 합니다. 따라서 그 다음 단계는 이들을 구현하는 것입니다. 도구 모음 단추에는 예 1에서 사용한 새로 고침 아이콘을 사용합니다.

```
class Example3Child implements CCTableObject {
  ...
  public String getName () {
    return name;
  }
  public Icon getIcon () {
return CommonImageRepository.getScaledIcon(CommonImageRepository.WC_NV_
```

```
REFRESH);
    }
    ...
  }
```

지금까지의 작업 결과를 보기 위해, 다음 연습에서 제거할 샘플 하위 오브젝트를 작성할 수 있습니다. childVector가 생성될 때 Example3Folder에 Example3Child의 인스턴스를 추가하십시오.

```
public class Example3Folder implements CCTreeObject {
  ...
  public Example3Folder() {
    childVector = new Vector();
    childVector.addElement(new Example3Child("Plugin1", "State1"));
  }
  ...
}
```

관련 개념:

- 452 페이지의『샘플 플러그인 컴파일 및 실행』

태스크 관련:

- 461 페이지의『트리에 다중 오브젝트를 포함하는 폴더 추가』
- 463 페이지의『폴더에 예제 오브젝트 추가』
- 467 페이지의『작성 조치 추가』
- 470 페이지의『변경 조치 추가』

작성 조치 추가

프로시저:

사용자들이 런타임에 폴더 아래에 오브젝트를 작성할 수 있도록 하려면, 벡터를 갱신하고 클래스를 Observable로 하며 해당 사용자가 이벤트를 트리거할 때 notifyObserver를 호출하기만 하면 됩니다. 제어 센터는 관찰 가능한(Observable인) 모든 CCTableObject의 관찰자(Observer)로서 스스로를 자동으로 등록합니다.

먼저 하위 오브젝트의 벡터에 하위 오브젝트를 추가하는 메소드를 Example3Folder에 추가하십시오.

```
public class Example3Folder implements CCTreeObject, Observable {
  ...
  public void addChild(Example3Child child) {
    childVector.addElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_ADDED,
                    child));
  }
  ...
}
```

위 코드에서는 CCObjectCollectionEvent라는 새 클래스를 notifyObserver에 대한 인수로서 사용합니다. CCObjectCollectionEvent는 CCObject(예: 제어 센터 트리에 있는 폴더) 콜렉션에서의 변경사항을 나타내는 이벤트입니다. 제어 센터는 Observable을 확장하는 모든 CCObject를 관찰하며 트리 및 세부사항 보기를 갱신함으로써 CCObjectCollectionEvent에 응답합니다. 세 가지 유형의 이벤트(추가, 제거 및 변경)가 있습니다.

CCObjectCollectionEvent는 세 개의 인수를 취합니다. 첫 번째는 이벤트를 트리거한 오브젝트입니다. 두 번째는 이벤트 유형으로 OBJECT_ADDED, OBJECT_ALTERED 또는 OBJECT_REMOVED가 될 수 있습니다. 마지막 인수는 작성되는 새 오브젝트입니다.

다음으로, 사용자가 새 addChild 메소드의 호출을 트리거할 수 있게 하는 메뉴 항목을 폴더에 추가하십시오.

```
class CreateAction implements CCMenuAction {
  private int pluginNumber = 0;
  public String getMenuText () { return "Create"; }

  public void actionPerformed (ActionEvent e) {
Example3Folder folder = (Example3Folder)((Vector)e.getSource()).elementAt(0);
    folder.addChild(new Example3Child("Plugin " + ++pluginNumber, "State1"));
  }
}
```

ActionEvent에는 항상 조치가 호출된 모든 오브젝트의 벡터가 포함됩니다. 이 조치는 Example3Folder에서만 호출되고 폴더는 하나만 있을 수 있으므로, 벡터의 첫 번째 오브젝트를 캐스트하고 거기서 addChild를 호출합니다.

마지막 단계는 폴더에 메뉴 조치를 추가하는 것이며, 이로써 이제는 이전에 추가된 샘플 오브젝트를 제거할 수 있습니다.

```
public class Example3Folder extends Observable implements CCTreeObject {
  private CCMenuAction[] menuActions =
     new CCMenuAction[] { new CreateChildAction(); }
  ...
  public Example3Folder() {
    childVector = new Vector();
  }
  ...
  public CCMenuAction[] getMenuActions () {
    retun menuActions;
  }
  ...
}
```

관련 개념:

• 452 페이지의 『샘플 플러그인 컴파일 및 실행』

태스크 관련:

• 461 페이지의 『트리에 다중 오브젝트를 포함하는 폴더 추가』

- 463 페이지의『폴더에 예제 오브젝트 추가』
- 465 페이지의『플러그인 트리 오브젝트에 대한 속성 설정』
- 469 페이지의『다중 선택 지원과 함께 제거 조치 추가』
- 470 페이지의『변경 조치 추가』

다중 선택 지원과 함께 제거 조치 추가

프로시저:

이제 사용자들이 원하는 만큼의 플러그인 인스턴스를 작성할 수 있으므로, 다음으로는 이들을 삭제하는 기능도 제공하고자 할 것입니다. 먼저 하위 오브젝트를 제거하고 제어 센터에 통지하는 메소드를 Example3Folder에 추가하십시오.

```
public class Example3Folder extends Observable implements CCTreeObject {

  public void removeChild(Example3Child child) {
    childVector.removeElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_REMOVED,
                    child));
  }
}
```

그 다음 단계는 Example3Child에 메뉴 조치를 추가하는 것입니다. 사용자가 동시에 여러 오브젝트를 제거할 수 있도록 이 CCMenuAction이 MultiSelectable을 구현하도록 할 것입니다. 이 조치의 소스는 Example3Folder가 아닌 Example3Child 오브젝트의 벡터가 될 것이므로, Example3Folder를 다른 방식으로(예: 컨스트럭터에서) 메뉴 조치에 패스해야 합니다.

```
class RemoveAction implements CCMenuAction, MultiSelectable {
  private Example3Folder folder;

  public RemoveAction(Example3Folder folder) {
     this.folder = folder;
  }

  public String getMenuText () { return "Remove"; }

  public int getSelectionMode () { return MultiSelectable.MULTI_HANDLE_ONE; }

  public void actionPerformed (ActionEvent e) {
    Vector childrenVector = (Vector)e.getSource();
    for (int i = 0; i < childrenVector.size(); i++) {
      folder.removeChild((Example3Child)childrenVector.elementAt(i));
    }
  }
}
```

MultiSelectable을 구현하려면 getSelectionMode를 구현해야 합니다. 이 경우에는 MULTI_HANDLE_ONE을 리턴하도록 되어 있으며, 이는 여러 오브젝트가 선택될 때에도 이 메뉴 항목이 메뉴에 나타나며 선택된 모든 오브젝트에 대해 actionPerformed가 한 번만 호출됨을 의미합니다.

이제 Example3Child에 새 메뉴 조치를 추가하십시오. 여기에는 폴더에서 패스할 새 매개변수를 Example3Child 컨스트럭터에 추가하는 것도 포함됩니다.

```
class Example3Child implements CCTableObject {
  ...
  private CCMenuAction[] menuActions;

  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new RemoveAction(folder) };
  }
  ...
  public CCMenuAction[] getMenuActions () {
    return menuActions;
  }
}
```

새 컨스트럭터를 사용하도록 CreateAction을 변경하십시오.

```
class CreateAction implements CCMenuAction {
  ...
  public void actionPerformed (ActionEvent e) {
    ...
 folder.addChild(new Example3Child(folder, "Plugin " + ++pluginNumber,
"State 1"));
  }
}
```

관련 개념:

- 452 페이지의 『샘플 플러그인 컴파일 및 실행』

태스크 관련:

- 461 페이지의 『트리에 다중 오브젝트를 포함하는 폴더 추가』
- 463 페이지의 『폴더에 예제 오브젝트 추가』
- 465 페이지의 『플러그인 트리 오브젝트에 대한 속성 설정』
- 467 페이지의 『작성 조치 추가』
- 470 페이지의 『변경 조치 추가』

변경 조치 추가

프로시저:

제어 센터가 플러그인과 관련하여 청취하는 마지막 유형의 이벤트는 OBJECT_ALTERED 이벤트입니다. 이 예에서 이 기능을 설명할 수 있도록 앞의 예에서『State』컬럼을 작성했습니다. 변경 조치가 호출될 때 상태 값을 증분시킬 것입니다.

첫 번째 단계는 상태를 변경하는 메소드를 작성하는 것이나, 이번에는 폴더가 아닌 Example3Child에서입니다. 이 경우, 첫 번째 및 세 번째 인수 모두 Example3Child입니다. Observable을 확장하십시오.

```
class Example3Child extends Observable implements CCTableObject {
  ...
  public void setState(String state) {
    this.state = state;
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                        CCObjectCollectionEvent.OBJECT_ALTERED, this));
  }
  ...
}
```

다음으로, 변경에 대한 메뉴 조치를 작성하여 Example3Child의 CCMenuAction 배열에 추가하십시오. AlterAction 클래스는 CCDefaultMenuAction 인터페이스도 구현하여 사용자가 제어 센터에서 Example3Child 오브젝트를 두 번 누르면 호출되는 디폴트 조치로 변경을 정의합니다.

```
class AlterAction implements CCMenuAction, CCDefaultMenuAction {
  private int stateNumber = 1;
  public String getMenuText () { return "Alter"; }

  public void actionPerformed (ActionEvent e) {
    ((Example3Child)((Vector)e.getSource()).elementAt(0)).setState("State "
        + ++stateNumber);
  }
}

class Example3Child implements CCTableObject {
  ...
  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new AlterAction(),
                                       new RemoveAction(folder) };
  }
  ...
}
```

관련 개념:

- 452 페이지의『샘플 플러그인 컴파일 및 실행』

태스크 관련:

- 461 페이지의『트리에 다중 오브젝트를 포함하는 폴더 추가』
- 463 페이지의『폴더에 예제 오브젝트 추가』
- 465 페이지의『플러그인 트리 오브젝트에 대한 속성 설정』

- 467 페이지의 『작성 조치 추가』
- 469 페이지의 『다중 선택 지원과 함께 제거 조치 추가』

isConfigurable()로 구성 기능 사용 불가능화

프로시저:

유형이 UDB_DATABASE 또는 UDB_INSTANCE인 CCObject의 isConfigurable 메소드에서 false 값을 리턴하면 데이터베이스 또는 인스턴스 팝업 메뉴에서 구성 메뉴 항목이 제거됩니다.

관련 개념:

- 452 페이지의 『샘플 플러그인 컴파일 및 실행』

isEditable()을 사용하여 오브젝트 변경 기능 사용 불가능화

프로시저:

변경 조치를 지원하는 제어 센터 오브젝트는 해당 오브젝트에 대한 플러그인을 작성하고 isEditable 메소드에서 false를 리턴함으로써 해당 조치가 제거되도록 할 수 있습니다. 오브젝트의 완전한 이름이 해당 오브젝트의 getName 메소드에 의해 리턴되는 값과 일치하는 오브젝트인 경우에만 변경 조치가 제거되도록 지정할 수도 있습니다.

추가적 기능은 찾아보기 조치를 추가하는 것입니다. UDB 테이블, 뷰 및 인덱스에 대해서만 이를 수행할 수 있습니다. 찾아보기 조치는 db2plug.zip에 BrowseTable, BrowseView 또는 BrowseIndex라는 파일을 놓음으로써 추가됩니다. 이러한 파일은 비어 있을 수도 있으나, 이름은 반드시 BrowseTable, BrowseView 또는 BrowseIndex이어야 합니다. 찾아보기 단추를 추가하는 것은 특정 이름을 가지는 오브젝트로 제한할 수 없습니다.

관련 개념:

- 452 페이지의 『샘플 플러그인 컴파일 및 실행』

hasConfigurationDefaults()를 사용하여 구성 대화 상자에서 디폴트 단추 사용 불가능화

프로시저:

UDB 데이터베이스 및 인스턴스에 대한 구성 대화 상자에는 값을 다시 DB2 디폴트로 설정하는 단추가 있습니다. 사용자 고유의 디폴트 세트가 있으며 다른 사용자들이 실수로 이러한 단추를 누르지 못하게 하고자 할 경우, 플러그인을 사용하여 이 단추들을 사용 불가능화할 수 있습니다. CCObject를 구현하는 오브젝트를 작성하여 그 유형을 UDB_DATABASE 또는 UDB_INSTANCE로 설정하십시오. 이 오브젝트가 또한

Defaultable 인터페이스를 구현하도록 하십시오. 이 인터페이스에는 hasConfigurationDefaults라는 메소드가 포함됩니다. 이 메소드에서 false를 리턴하면 구성 대화 상자의 모든 디폴트 단추가 사용 불가능화됩니다. 이 예는 UDB 데이터베이스 구성의 디폴트 단추를 사용 불가능화합니다.

관련 개념:

• 452 페이지의『샘플 플러그인 컴파일 및 실행』

부록 K. DB2 Universal Database 기술 정보

DB2 문서 및 도움말

다음 도구 및 메소드를 통해 DB2® 기술 정보를 볼 수 있습니다.

- DB2 정보 센터
  - 주제
  - DB2 도구에 대한 도움말
  - 샘플 프로그램
  - 자습서
- 다운로드 가능한 PDF 파일, CD의 PDF 파일 및 인쇄 서적
  - 안내서
  - 참조 매뉴얼
- 명령행 도움말
  - 명령 도움말
  - 메시지 도움말
  - SQL 상태 도움말
- 설치된 소스 코드
  - 샘플 프로그램

ibm.com®에서 기술서, 백서 및 Redbooks™와 같은 추가 DB2 Universal Database™ 기술 정보에 온라인으로 액세스할 수 있습니다. DB2 정보 관리 소프트웨어 라이브러리 사이트 www.ibm.com/software/data/pubs/에 액세스하십시오.

DB2 문서 갱신사항

IBM®은 정기적으로 문서 FixPak 및 DB2 정보 센터의 기타 문서 갱신사항을 사용할 수 있도록 합니다. DB2 정보 센터 http://publib.boulder.ibm.com/infocenter/db2help/에 액세스하는 경우, 항상 최신 정보를 보게 됩니다. DB2 정보 센터를 로컬로 설치한 경우, 갱신사항을 보려면 이를 수동으로 설치해야 합니다. 새 정보가 사용 가능할 때 문서 갱신사항을 통해 *DB2 정보 센터 CD*에서 설치한 정보를 갱신할 수 있습니다.

정보 센터는 PDF 또는 하드카피 책보다 자주 갱신됩니다. 최신 DB2 기술 정보를 가져오려면 사용 가능한 문서 갱신사항을 설치하거나 DB2 정보 센터, www.ibm.com으로 이동하십시오.

관련 개념:

- *CLI Guide and Reference, Volume 1*의 『CLI sample programs』
- *응용프로그램 개발 안내서: 응용프로그램 빌드 및 실행*의 『Java 샘플 프로그램』
- 476 페이지의 『DB2 정보 센터』

태스크 관련:

- 495 페이지의 『DB2 도구에서 문맥 도움말 호출』
- 486 페이지의 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신』
- 496 페이지의 『명령행 처리기에서 메시지 도움말 호출』
- 496 페이지의 『명령행 처리기에서 명령 도움말 호출』
- 497 페이지의 『명령행 처리기에서 SQL 상태 도움말 호출』

관련 참조:

- 488 페이지의 『DB2 PDF 및 인쇄 문서』

DB2 정보 센터

DB2® 정보 센터는 DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator 및 DB2 Query Patroller™를 포함하여 DB2 계열의 제품을 최대한 이용하는 데 필요한 모든 정보에 액세스할 수 있도록 합니다. 또한, DB2 정보 센터에는 복제, 데이터 웨어하우징 및 DB2 Extender와 같은 주요 DB2 기능과 구성요소에 대한 정보가 포함됩니다.

Mozilla 1.0 이상 또는 Microsoft® Internet Explorer 5.5 이상에서 DB2 정보 센터를 보는 경우, 다음 기능을 사용할 수 있습니다. 일부 기능을 사용하려면 JavaScript™ 지원을 사용 가능하게 해야 합니다.

융통성있는 설치 옵션

사용자 필요에 가장 적합한 옵션을 사용하여 DB2 문서를 보기로 선택할 수 있습니다.

- 문서를 항상 최신 상태로 유지하기 위해 다음 IBM® 웹 사이트 (http://publib.boulder.ibm.com/infocenter/db2help/)에 호스트되는 DB2 정보 센터에서 직접 모든 문서에 액세스할 수 있습니다.
- 갱신 작업을 최소화하고 인트라넷 내에서 네트워크 트래픽을 유지하려면 인트라넷의 단일 서버에 DB2 문서를 설치하십시오.
- 융통성을 최대화하고 네트워크 연결 종속성을 줄이려면 사용자의 컴퓨터에 DB2 문서를 설치하십시오.

검색 **검색** 텍스트 필드에 검색 용어를 입력하여 DB2 정보 센터에서 모든 항목을 검색할 수 있습니다. 용어를 따옴표로 묶어서 완전히 일치하는 항목을 검색하고, 와일드카드 연산자(*, ?) 및 Boolean 연산자(AND, NOT, OR)를 사용하여 검색을 구체화할 수 있습니다.

태스크 위주 목차

DB2 문서의 단일 목차에서 항목을 찾을 수 있습니다. 목차는 주로 수행하려는 태스크 종류별로 구성되지만, 제품 개요, 목표, 참조 정보, 인덱스 및 용어집 항목도 포함됩니다.

- 제품 개요에서는 DB2 계열의 사용 가능한 제품 간의 관계, 이 제품이 제공하는 기능 및 이 제품의 최신 릴리스 정보를 설명합니다.
- 설치, 관리 및 개발과 같은 목표 범주는 빠르게 태스크를 완료하고 이 태스크를 완료하는 백그라운드 정보에 대한 심도 깊은 이해를 도울 수 있는 항목을 포함합니다.
- 참조 항목은 명령문 및 명령 구문, 메시지 도움말, 구성 매개변수를 포함하여 주제에 대한 자세한 정보를 제공합니다.

목차에 현재 항목 표시

목차 프레임에서 **새로 고침/현재 항목 표시** 단추를 누르거나 컨텐츠 프레임에서 **목차에 표시** 단추를 눌러 목차에서 현재 항목이 들어가는 위치를 표시할 수 있습니다. 이 기능은 여러 파일에서 관련 항목에 대한 여러 링크를 수행하거나 검색 결과에서 항목에 도달한 경우에 유용합니다.

인덱스 인덱스에서 모든 문서에 액세스할 수 있습니다. 인덱스는 인덱스 용어에 따라 알파벳순으로 구성됩니다.

용어집 용어집을 사용하여 DB2 문서에 사용된 용어의 정의를 찾아볼 수 있습니다. 용어집은 용어집의 용어에 따라 알파벳순으로 구성됩니다.

통합된 지역 정보

DB2 정보 센터는 브라우저 환경설정에 설정된 선호 언어로 정보를 표시합니다. 항목을 선호 언어로 표시할 수 없는 경우, DB2 정보 센터는 항목을 영어 버전으로 표시합니다.

iSeries™ 기술 정보는 www.ibm.com/eserver/iseries/infocenter/의 IBM eServer™ iSeries 정보 센터를 참조하십시오.

관련 개념:

- 478 페이지의『DB2 정보 센터 설치 시나리오』

태스크 관련:

- 486 페이지의『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신』
- 487 페이지의『DB2 정보 센터에서 선호 언어로 항목 표시』
- 485 페이지의『DB2 정보 센터 호출』
- 480 페이지의『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(UNIX)』
- 483 페이지의『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(Windows)』

DB2 정보 센터 설치 시나리오

작업 환경이 다르면 DB2® 정보 액세스 방법에 대한 요구사항도 다를 수 있습니다. DB2 정보 센터는 IBM® 웹 사이트, 조직 네트워크의 서버 또는 사용자 컴퓨터에 설치된 버전에서 액세스할 수 있습니다. 세 경우 모두, 문서는 브라우저를 사용하여 볼 수 있는 항목 기반 정보의 구조화된 웹 형태인 DB2 정보 센터에 포함됩니다. 디폴트로, DB2 제품은 IBM 웹 사이트의 DB2 정보 센터에 액세스합니다. 그러나 인트라넷 서버 또는 사용자 컴퓨터의 DB2 정보 센터에 액세스하려면 제품 미디어 팩에 있는 DB2 정보 센터 CD를 사용하여 DB2 정보 센터를 설치해야 합니다. 세 가지 설치 시나리오와 함께 다음에 나오는 DB2 문서 액세스 옵션 요약 정보를 참조하면 사용자 및 사용자 작업 환경에 가장 적합한 DB2 정보 센터 액세스 메소드 및 고려해야 하는 설치 사항을 판별하는 데 도움이 됩니다.

DB2 문서 액세스 옵션 요약:

다음 표는 사용자 작업 환경에서 DB2 정보 센터의 DB2 제품 문서에 액세스하기 위해 사용할 수 있는 옵션에 대한 권장사항을 제공합니다.

인터넷 액세스	인트라넷 액세스	권장사항
예	예	IBM 웹 사이트의 DB2 정보 센터에 액세스하거나, 인트라넷 서버에 설치된 DB2 정보 센터에 액세스하십시오.
예	아니오	IBM 웹 사이트의 DB2 정보 센터에 액세스하십시오.
아니오	예	인트라넷 서버에 설치된 DB2 정보 센터에 액세스하십시오.
아니오	아니오	로컬 컴퓨터의 DB2 정보 센터에 액세스하십시오.

시나리오: 사용자 컴퓨터에서 DB2 정보 센터 액세스:

Tsu-Chen은 소도시에서 인터넷 액세스를 제공하는 로컬 ISP가 없는 공장을 소유하고 있습니다. 그는 재고, 제품 주문, 은행 계좌 정보 및 비즈니스 비용을 관리하기 위해 DB2 Universal Database™를 구입했습니다. 그는 이전에 DB2 제품을 사용해 본 경험이 없으므로 DB2 제품 문서에서 이 제품의 사용 방법을 학습해야 합니다.

Tsu-Chen은 일반 설치 옵션을 사용하여 컴퓨터에 DB2 Universal Database를 설치한 후, DB2 문서에 액세스하려고 합니다. 그러나 브라우저는 열려는 페이지를 찾을 수 없음을 나타내는 오류 메시지를 표시합니다. 그는 DB2 제품의 설치 매뉴얼을 점검하여 자신의 컴퓨터에서 DB2 문서에 액세스하려면 DB2 정보 센터를 설치해야 함을 알게 됩니다. 그는 미디어 팩에서 *DB2 정보 센터 CD*를 찾아서 이를 설치합니다.

Tsu-Chen은 이제 운영 체제의 응용프로그램 실행 프로그램에서 DB2 정보 센터에 액세스할 수 있으며, DB2 제품을 사용하여 비즈니스의 성공을 증가시킬 수 있는 방법을 학습할 수 있습니다.

시나리오: IBM 웹 사이트에서 DB2 정보 센터 액세스:

Colin은 교육 기관의 정보 기술 상담자입니다. 그는 데이터베이스 기술 및 SQL 전문가이고 DB2 Universal Database를 사용하여 북미 전체의 회사에서 이러한 주제를 다루는 세미나를 주최합니다. 그는 세미나 일부에서 교육 도구로 DB2 문서를 사용합니다. 예를 들어, SQL 교육 과정 중에 데이터베이스 쿼리의 기본 구문 및 고급 구문을 가르치기 위한 한 방법으로 SQL에 관한 DB2 문서를 사용합니다.

Colin이 교육하는 대부분의 회사에서는 인터넷 액세스가 가능합니다. 이러한 상황 때문에 그는 최신 버전의 DB2 Universal Database를 설치할 때, 그의 이동 컴퓨터가 IBM 웹 사이트의 DB2 정보 센터에 액세스할 수 있도록 구성합니다. 이 구성 방식은 그가 세미나 중에 최신 DB2 문서에 온라인으로 액세스 가능하도록 합니다.

그러나 Colin은 가끔 여행 중에 인터넷에 액세스할 수 없습니다. 이로 인해 문제가 생기며, 특히 세미나 준비를 위해 DB2 문서에 액세스해야 하는 경우에 문제가 됩니다. 이와 같은 상황을 피하기 위해 그는 이동 컴퓨터에 DB2 정보 센터 사본을 설치했습니다.

Colin은 원할 때 DB2 문서의 사본을 사용할 수 있게 하는 융통성에 만족합니다. 그는 **db2set** 명령을 사용하여 상황에 따라 IBM 웹 사이트 또는 이동 컴퓨터의 DB2 정보 센터에 액세스할 수 있도록 이동 컴퓨터의 레지스트리 변수를 쉽게 구성할 수 있습니다.

시나리오: 인트라넷 서버에서 DB2 정보 센터 액세스:

Eva는 생명 보험 회사의 고위 데이터베이스 관리자로 근무하고 있습니다. 그녀의 관리 책임은 회사의 UNIX® 데이터베이스 서버에 최신 버전의 DB2 Universal Database를 설치하고 구성하는 것입니다. 이 회사는 최근에 직원들에게 보안상의 이유로 직장에서 인터넷 액세스를 제공하지 않음을 통보했습니다. 회사에 네트워크 환경이 구축되어 있기 때문에 그녀는 회사의 데이터 웨어하우스를 정기적으로 사용하는 모든 직원(영업 대표, 영업 관리자 및 업무 분석가)이 DB2 문서에 액세스할 수 있도록 하기 위해 DB2 정보 센터 사본을 인트라넷 서버에 설치하기로 결정했습니다.

Eva는 데이터베이스 팀에게 응답 파일을 사용하여 모든 직원의 컴퓨터에 최신 버전의 DB2 Universal Database를 설치하도록 지시합니다. 이는 각 컴퓨터가 인트라넷 서버의 호스트 이름 및 포트 번호를 사용하여 DB2 정보 센터에 액세스할 수 있도록 구성하기 위한 것입니다.

그러나 Eva 팀의 하위 데이터베이스 관리자인 Migual이 이를 잘못 이해하여 인트라넷 서버의 DB2 정보 센터에 액세스하도록 DB2 Universal Database를 구성하는 대신 몇몇 직원의 컴퓨터에 DB2 정보 센터 사본을 설치합니다. 이 상황을 해결하기 위해 Eva는 Migual에게 **db2set** 명령을 사용하여 이러한 각 컴퓨터에서 DB2 Information Center 레지스트리 변수(호스트 이름의 경우 DB2_DOCHOST, 포트 번호의 경우

DB2_DOCPORT)를 변경하도록 지시합니다. 이제, 네트워크의 적절한 모든 컴퓨터가 DB2 정보 센터에 액세스할 수 있으며, 직원들이 DB2 문서에서 DB2 질문에 대한 답변을 찾을 수 있습니다.

관련 개념:

- 476 페이지의『DB2 정보 센터』

태스크 관련:

- 486 페이지의『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신』
- 480 페이지의『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(UNIX)』
- 483 페이지의『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(Windows)』
- 『DB2 정보 센터에 액세스하기 위해 위치 설정: 일반 GUI 도움말』

관련 참조:

- *Command Reference*의『db2set - DB2 Profile Registry Command』

DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(UNIX)

DB2 제품 문서에는 세 가지 방법(IBM 웹 사이트에서, 인트라넷 서버에서 또는 컴퓨터에 설치된 버전에서)으로 액세스할 수 있습니다. 디폴트로, DB2 제품은 IBM 웹 사이트의 DB2 문서에 액세스합니다. 인트라넷 서버 또는 사용자 컴퓨터의 DB2 문서에 액세스하려면, *DB2 정보 센터 CD*에서 문서를 설치해야 합니다. DB2 설치 마법사를 사용하여 설치 환경설정을 정의하고 UNIX 운영 체제를 사용하는 컴퓨터에 DB2 정보 센터를 설치할 수 있습니다.

전제조건:

이 절에서는 UNIX 컴퓨터에 DB2 정보 센터를 설치하는 데 필요한 하드웨어, 운영 체제, 소프트웨어 및 통신 요구사항을 나열합니다.

- **하드웨어 요구사항**

  다음 프로세서 중 하나가 필요합니다.

  - PowerPC(AIX)
  - HP 9000(HP-UX)
  - Intel 32비트(Linux)
  - Solaris UltraSPARC 컴퓨터(Solaris 운영 환경)

- **운영 체제 요구사항**

  다음 운영 체제 중 하나가 필요합니다.

  - IBM AIX 5.1(PowerPC)
  - HP-UX 11i(HP 9000)

- Red Hat Linux 8.0(Intel 32비트)
- SuSE Linux 8.1(Intel 32비트)
- Sun Solaris 버전 8(Solaris 운영 환경 UltraSPARC 컴퓨터)

주: DB2 정보 센터는 DB2 클라이언트가 지원되는 UNIX 운영 체제 서브세트에서 실행됩니다. 따라서, IBM 웹 사이트에서 DB2 정보 센터에 액세스하거나, DB2 정보 센터를 인트라넷 서버에 설치하여 액세스하는 것이 바람직합니다.

- **소프트웨어 요구사항**
  - 다음 브라우저가 지원됩니다.
    - Mozilla 버전 1.0 이상
- DB2 설치 마법사는 그래픽 설치 프로그램입니다. 컴퓨터에서 DB2 설치 마법사를 실행하려면 그래픽 사용자 인터페이스를 렌더링할 수 있는 X Window System 소프트웨어가 구현되어 있어야 합니다. DB2 설치 마법사를 실행하기 전에 표시 화면을 올바르게 익스포트했는지 확인해야 합니다. 예를 들어, 명령 프롬프트에서 다음 명령을 입력하십시오.

  ```
  export DISPLAY=9.26.163.144:0.
  ```

- **통신 요구사항**
  - TCP/IP

프로시저:

DB2 설치 마법사를 사용하여 DB2 정보 센터를 설치하려면 다음을 수행하십시오.

1. 시스템에 로그온하십시오.
2. DB2 정보 센터 제품 CD를 시스템에 넣고 마운트하십시오.
3. 다음 명령을 입력하여 CD가 마운트된 디렉토리로 변경하십시오.

   ```
   cd /cd
   ```

   여기서, */cd*는 CD의 마운트 지점을 나타냅니다.
4. **./db2setup** 명령을 입력하여 DB2 설치 마법사를 시작하십시오.
5. IBM DB2 설치 런치패드가 열립니다. DB2 정보 센터의 설치로 직접 이동하려면 **제품 설치**를 누르십시오. 온라인 도움말을 참조하여 나머지 단계를 진행할 수 있습니다. 온라인 도움말을 호출하려면 **도움말**을 누르십시오. **취소**를 누르면 언제라도 설치를 종료할 수 있습니다.
6. **설치하려는 제품 설치** 페이지에서 **다음**을 누르십시오.
7. **DB2 설치 마법사 시작** 페이지에서 **다음**을 누르십시오. DB2 설치 마법사가 프로그램 설치 프로세스를 안내합니다.
8. 설치를 계속하려면, 라이센스 계약을 승인해야 합니다. **라이센스 계약** 페이지에서 **라이센스 계약의 조항을 승인합니다**를 선택하고 **다음**을 누르십시오.

9. **설치 조치 선택** 페이지에서 **이 컴퓨터에 DB2 정보 센터 설치**를 선택하십시오. 나중에 응답 파일을 사용하여 이 컴퓨터 또는 다른 컴퓨터에 DB2 정보 센터를 설치하려면 **응답 파일에 설정값 저장**을 선택하십시오. **다음**을 누르십시오.
10. **설치할 언어 선택** 페이지에서 DB2 정보 센터를 설치할 언어를 선택하십시오. **다음**을 누르십시오.
11. **DB2 정보 센터 포트 지정** 페이지에서 들어오는 통신을 위한 DB2 정보 센터를 구성하십시오. 설치를 계속하려면 **다음**을 누르십시오.
12. **파일 복사 시작** 페이지에서 설치 선택사항을 검토하십시오. 설정값을 변경하려면 **뒤로**를 누르십시오. DB2 정보 센터 파일을 컴퓨터로 복사하려면 **설치**를 누르십시오.

응답 파일을 사용하여 DB2 정보 센터를 설치할 수도 있습니다.

디폴트로, 설치 로그 db2setup.his, db2setup.log 및 db2setup.err는 /tmp 디렉토리에 위치합니다.

db2setup.log 파일은 오류를 포함하여 모든 DB2 제품 설치 정보를 캡처합니다. db2setup.his 파일은 컴퓨터의 모든 DB2 제품 설치를 기록합니다. DB2는 db2setup.log 파일을 db2setup.his 파일에 추가합니다. db2setup.err 파일은 예외 및 트랩 정보와 같은 Java에 의해 리턴되는 오류 출력을 캡처합니다.

설치가 완료되면, UNIX 운영 체제에 따라 DB2 정보 센터가 다음 디렉토리 중 하나에 설치됩니다.

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris 운영 환경: /opt/IBM/db2/V8.1

관련 개념:

- 476 페이지의 『DB2 정보 센터』
- 478 페이지의 『DB2 정보 센터 설치 시나리오』

태스크 관련:

- *설치 및 구성 보충 설명서*의 『응답 파일을 사용하여 DB2 설치(UNIX)』
- 486 페이지의 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신』
- 487 페이지의 『DB2 정보 센터에서 선호 언어로 항목 표시』
- 485 페이지의 『DB2 정보 센터 호출』
- 483 페이지의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(Windows)』

DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(Windows)

DB2 제품 문서에는 세 가지 방법(IBM 웹 사이트에서, 인트라넷 서버에서 또는 컴퓨터에 설치된 버전에서)으로 액세스할 수 있습니다. 디폴트로, DB2 제품은 IBM 웹 사이트의 DB2 문서에 액세스합니다. 인트라넷 서버 또는 사용자 컴퓨터의 DB2 문서에 액세스하려면, *DB2 정보 센터 CD*에서 DB2 문서를 설치해야 합니다. DB2 설치 마법사를 사용하여 설치 환경설정을 정의하고 Windows 운영 체제를 사용하는 컴퓨터에 DB2 정보 센터를 설치할 수 있습니다.

전제조건:

이 절에서는 Windows에 DB2 정보 센터를 설치하는 데 필요한 하드웨어, 운영 체제, 소프트웨어 및 통신 요구사항을 나열합니다.

- **하드웨어 요구사항**

  다음 프로세서 중 하나가 필요합니다.

  – 32비트 컴퓨터: Pentium 또는 Pentium 호환 CPU

- **운영 체제 요구사항**

  다음 운영 체제 중 하나가 필요합니다.

  – Windows 2000

  – Windows XP

  주: DB2 정보 센터는 DB2 클라이언트가 지원되는 Windows 운영 체제 서브세트에서 실행됩니다. 따라서 IBM 웹 사이트에서 DB2 정보 센터에 액세스하거나, DB2 정보 센터를 인트라넷 서버에 설치하여 액세스하는 것이 바람직합니다.

- **소프트웨어 요구사항**

  – 다음 브라우저가 지원됩니다.

  - Mozilla 버전 1.0 이상

  - Internet Explorer 버전 5.5 또는 6.0(Windows XP의 경우, 버전 6.0)

- **통신 요구사항**

  – TCP/IP

제한사항:

- DB2 정보 센터를 설치할 수 있는 관리 특권이 있는 계정이 필요합니다.

프로시저:

DB2 설치 마법사를 사용하여 DB2 정보 센터를 설치하려면 다음을 수행하십시오.

1. DB2 정보 센터 설치를 위해 정의한 계정으로 시스템에 로그온하십시오.

2. CD를 드라이브에 넣으십시오. 사용 가능한 경우, 자동 실행 기능이 IBM DB2 설치 런치패드를 시작합니다.
3. DB2 설치 마법사는 시스템 언어를 판별하고 해당 언어의 설치 프로그램을 시작합니다. 영어가 아닌 다른 언어로 설치 프로그램을 실행하거나 설치 프로그램의 자동 시작에 실패하는 경우, DB2 설치 마법사를 수동으로 시작할 수 있습니다.

   DB2 설치 마법사를 수동으로 시작하려면 다음을 수행하십시오.

   a. **시작**을 누른 후 **실행**을 선택하십시오.
   b. **열기** 필드에 다음 명령을 입력하십시오.

      ```
      x:\setup.exe /i 2-letter language identifier
      ```

      여기서, *x:*는 CD 드라이브를 나타내고, *2-letter language identifier*는 설치 프로그램이 실행되는 언어를 나타냅니다.
   c. **예**를 누르십시오.
4. IBM DB2 설치 런치패드가 열립니다. DB2 정보 센터의 설치로 직접 이동하려면 **제품 설치**를 누르십시오. 온라인 도움말을 참조하여 나머지 단계를 진행할 수 있습니다. 온라인 도움말을 호출하려면 **도움말**을 누르십시오. **취소**를 누르면 언제라도 설치를 종료할 수 있습니다.
5. **설치하려는 제품 설치** 페이지에서 **다음**을 누르십시오.
6. **DB2 설치 마법사 시작** 페이지에서 **다음**을 누르십시오. DB2 설치 마법사가 프로그램 설치 프로세스를 안내합니다.
7. 설치를 계속하려면, 라이센스 계약을 승인해야 합니다. **라이센스 계약** 페이지에서 **라이센스 계약의 조항을 승인합니다**를 선택하고 **다음**을 누르십시오.
8. **설치 조치 선택** 페이지에서 **이 컴퓨터에 DB2 정보 센터 설치**를 선택하십시오. 나중에 응답 파일을 사용하여 이 컴퓨터 또는 다른 컴퓨터에 DB2 정보 센터를 설치하려면 **응답 파일에 설정값 저장**을 선택하십시오. **다음**을 누르십시오.
9. **설치할 언어 선택** 페이지에서 DB2 정보 센터를 설치할 언어를 선택하십시오. **다음**을 누르십시오.
10. **DB2 정보 센터 포트 지정** 페이지에서 들어오는 통신을 위한 DB2 정보 센터를 구성하십시오. 설치를 계속하려면 **다음**을 누르십시오.
11. **파일 복사 시작** 페이지에서 설치 선택사항을 검토하십시오. 설정값을 변경하려면 **뒤로**를 누르십시오. DB2 정보 센터 파일을 컴퓨터로 복사하려면 **설치**를 누르십시오.

응답 파일을 사용하여 DB2 정보 센터를 설치할 수 있습니다. **db2rspgn** 명령을 사용하여 기존 설치에 기초하는 응답 파일을 생성할 수도 있습니다.

설치시 발생한 오류에 대한 정보는 'My Documents'\DB2LOG\ 디렉토리에 있는 db2.log 및 db2wi.log 파일을 참조하십시오. 'My Documents' 디렉토리 위치는 사용자 컴퓨터 설정값에 따라 달라집니다.

db2wi.log 파일은 최신 DB2 설치 정보를 캡처합니다. db2.log는 DB2 제품 설치의 실행기록을 캡처합니다.

관련 개념:

- 476 페이지의 『DB2 정보 센터』
- 478 페이지의 『DB2 정보 센터 설치 시나리오』

태스크 관련:

- *설치 및 구성 보충 설명서*의 『응답 파일을 사용하여 DB2 제품 설치(Windows)』
- 486 페이지의 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신』
- 487 페이지의 『DB2 정보 센터에서 선호 언어로 항목 표시』
- 485 페이지의 『DB2 정보 센터 호출』
- 480 페이지의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(UNIX)』

관련 참조:

- *Command Reference*의 『db2rspgn - Response File Generator Command (Windows)』

DB2 정보 센터 호출

DB2 정보 센터는 DB2 Universal Database, DB2 Connect, DB2 Information Integrator 및 DB2 Query Patroller와 같은 Linux, UNIX 및 Windows 운영 체제용 DB2 제품을 사용하는 데 필요한 모든 정보에 액세스할 수 있도록 합니다.

다음 위치 중 하나에서 DB2 정보 센터를 호출할 수 있습니다.

- DB2 UDB 클라이언트 또는 서버가 설치된 컴퓨터
- DB2 정보 센터가 설치된 인트라넷 서버 또는 로컬 컴퓨터
- IBM 웹 사이트

전제조건:

DB2 정보 센터를 호출하기 전에 다음을 수행하십시오.

- *선택적*: 브라우저가 사용자 선호 언어로 항목을 표시하도록 구성하십시오.
- *선택적*: DB2 클라이언트가 사용자 컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 사용하도록 구성하십시오.

프로시저:

DB2 UDB 클라이언트 또는 서버가 설치된 컴퓨터에서 DB2 정보 센터를 호출하려면 다음을 수행하십시오.

- 시작 메뉴에서(Windows 운영 체제): **시작** → **프로그램** → **IBM DB2** → **정보** → **정보 센터**를 누르십시오.

- 명령행 프롬프트에서, 다음을 수행하십시오.
  - Linux 및 UNIX 운영 체제의 경우, **db2icdocs** 명령을 발행하십시오.
  - Windows 운영 체제의 경우, **db2icdocs.exe** 명령을 발행하십시오.

웹 브라우저에서 인트라넷 서버 또는 로컬 컴퓨터에 설치된 DB2 정보 센터를 열려면 다음을 수행하십시오.

- 웹 페이지(http://<host-name>:<port-number>/)를 여십시오. 여기서, <host-name>은 호스트 이름을 나타내고, <port-number>는 DB2 정보 센터가 사용 가능한 포트 번호를 나타냅니다.

웹 브라우저에서 IBM 웹 사이트의 DB2 정보 센터를 열려면 다음을 수행하십시오.

- 웹 페이지(publib.boulder.ibm.com/infocenter/db2help/)를 여십시오.

관련 개념:

- 476 페이지의 『DB2 정보 센터』
- 478 페이지의 『DB2 정보 센터 설치 시나리오』

태스크 관련:

- 495 페이지의 『DB2 도구에서 문맥 도움말 호출』
- 486 페이지의 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신』
- 496 페이지의 『명령행 처리기에서 명령 도움말 호출』
- 『DB2 정보 센터에 액세스하기 위해 위치 설정: 일반 GUI 도움말』

관련 참조:

- *Command Reference*의 『HELP Command』

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신

사용 가능한 DB2 정보 센터는 http://publib.boulder.ibm.com/infocenter/db2help/에서 새 문서 또는 변경된 문서를 사용하여 주기적으로 갱신됩니다. 또한 IBM은 DB2 정보 센터 갱신사항을 컴퓨터 또는 인트라넷 서버로 다운로드하여 설치할 수 있도록 합니다. DB2 정보 센터를 갱신해도 DB2 클라이언트 또는 서버 제품이 갱신되지는 않습니다.

전제조건:

인터넷에 연결된 컴퓨터에 액세스할 수 있어야 합니다.

프로시저:

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 갱신하려면 다음을 수행하십시오.

1. 다음 IBM 웹 사이트(http://publib.boulder.ibm.com/infocenter/db2help/)에 호스트되는 DB2 정보 센터를 여십시오.
2. 환영 페이지의 다운로드 섹션에 있는 서비스 및 지원 표제 아래에서 **DB2 Universal Database 문서** 링크를 누르십시오.
3. 새로 고친 최신 문서 이미지 레벨과 설치한 문서 레벨을 비교하여 DB2 정보 센터 버전이 오래된 버전인지 판별하십시오. 설치한 문서 레벨은 DB2 정보 센터 환영 페이지에 나열됩니다.
4. 더 최신 버전의 DB2 정보 센터가 사용 가능하면, 운영 체제에 적용되는 새로 고친 최신 *DB2 정보 센터* 이미지를 다운로드하십시오.
5. 새로 고친 *DB2 정보 센터* 이미지를 설치하려면 웹 페이지에 제공되는 지시사항을 따르십시오.

관련 개념:

• 478 페이지의 『DB2 정보 센터 설치 시나리오』

태스크 관련:

• 485 페이지의 『DB2 정보 센터 호출』
• 480 페이지의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(UNIX)』
• 483 페이지의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치(Windows)』

DB2 정보 센터에서 선호 언어로 항목 표시

DB2 정보 센터는 브라우저 환경설정에 지정된 언어로 항목을 표시합니다. 항목이 선호 언어로 변환되지 않는 경우, DB2 정보 센터는 항목을 영어로 표시합니다.

프로시저:

Internet Explorer 브라우저에서 선호 언어로 항목을 표시하려면 다음을 수행하십시오.

1. Internet Explorer에서 **도구 —> 인터넷 옵션 —> 언어...** 단추를 누르십시오. 언어 환경설정 창이 열립니다.
2. 선호 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
   - 목록에 새 언어를 추가하려면 **추가...** 단추를 누르십시오.

     주: 언어를 추가해도 반드시 컴퓨터에 선호 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. 페이지를 새로 고쳐서 선호 언어로 DB2 정보 센터를 표시하십시오.

Mozilla 브라우저에서 선호 언어로 항목을 표시하려면 다음을 수행하십시오.

1. Mozilla에서 **편집 —> 환경설정 —> 언어** 단추를 누르십시오. 환경설정 창에 언어 패널이 표시됩니다.
2. 선호 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
   - 목록에 새 언어를 추가하려면 **추가...** 단추를 눌러 언어 추가 창에서 언어를 선택하십시오.
   - 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
3. 페이지를 새로 고쳐서 선호 언어로 DB2 정보 센터를 표시하십시오.

관련 개념:

• 476 페이지의 『DB2 정보 센터』

DB2 PDF 및 인쇄 문서

다음 표는 공식적인 책 이름, 문서 번호 및 PDF 파일 이름을 제공합니다. 하드카피 책을 주문하려면 공식적인 책 이름을 알아야 합니다. PDF 파일을 인쇄하려면 PDF 파일 이름을 알아야 합니다.

DB2 문서는 다음 표제로 범주화됩니다.

- 핵심 DB2 정보
- 관리 정보
- 응용프로그램 개발 정보
- 비즈니스 인텔리전스 정보
- DB2 Connect 정보
- 시작하기 정보
- 자습서 정보
- 선택적 구성요소 정보
- 릴리스 정보

다음 표는 DB2 라이브러리의 각 책에 대한 하드카피를 주문하고, 해당 책의 PDF를 인쇄하거나 보는 데 필요한 정보를 설명합니다. DB2 라이브러리의 각 책에 대한 전체 설명은 IBM Publications Center(www.ibm.com/shop/publications/order)에서 사용 가능합니다.

핵심 DB2 정보

이 정보는 모든 DB2 사용자에게 기본적으로 필요합니다. 이 정보는 프로그래머, 데이터베이스 관리자 및 DB2 Connect, DB2 Warehouse Manager 또는 기타 DB2 제품에 대한 작업을 수행하는 모든 사용자에게 유용합니다.

표 53. 핵심 DB2 정보

이름	문서 번호	PDF 파일 이름
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x81
IBM DB2 Universal Database 용어집	문서 번호 없음	db2t0x81
IBM DB2 Universal Database 메시지 참조서, 볼륨 1	GA30-1496, 하드카피로 사용할 수 없음	db2m1x81
IBM DB2 Universal Database 메시지 참조서, 볼륨 2	GA30-1497, 하드카피로 사용할 수 없음	db2m2x81
IBM DB2 Universal Database 새로운 기능	SA30-1495	db2q0x81

관리 정보

이 정보는 DB2 데이터베이스, 데이터 웨어하우스 및 페더레이티드 시스템을 효율적으로 설계, 구현 및 유지보수하는 데 필요한 주제를 설명합니다.

표 54. 관리 정보

이름	문서 번호	PDF 파일 이름
IBM DB2 Universal Database 관리 안내서: 계획	SA30-1481	db2d1x81
IBM DB2 Universal Database 관리 안내서: 구현	SA30-1479	db2d2x81
IBM DB2 Universal Database 관리 안내서: 성능	SA30-1480	db2d3x81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database 데이터 이동 유틸리티 안내 및 참조서	SA30-1486	db2dmx81
IBM DB2 Universal Database 데이터 복구 및 고가용성 안내 및 참조서	SA30-1487	db2hax81
IBM DB2 Universal Database Data Warehouse Center 관리 안내서	SA30-1509	db2ddx81
IBM DB2 Universal Database SQL 참조서, 볼륨 1	SA30-1498	db2s1x81
IBM DB2 Universal Database SQL 참조서, 볼륨 2	SA30-1499	db2s2x81
IBM DB2 Universal Database 시스템 모니터 안내 및 참조서	SA30-1500	db2f0x81

응용프로그램 개발 정보

이 정보는 DB2 Universal Database(DB2 UDB)로 작업하는 응용 프로그램 개발자나 프로그래머에게 특히 유용합니다. 지원되는 언어 및 컴파일러와 Embedded SQL, ODBC, JDBC, SQLJ 및 CLI 등 지원되는 여러 프로그래밍 인터페이스를 사용하여 DB2 UDB에 액세스하는 데 필요한 문서 정보도 포함되어 있습니다. DB2 정보 센터를 사용하는 경우, 소스 코드의 HTML 버전에 액세스하여 샘플 프로그램을 사용할 수도 있습니다.

표 55. 응용프로그램 개발 정보

이름	문서 번호	PDF 파일 이름
IBM DB2 Universal Database 응용프로그램 개발 안내서: 응용프로그램 빌드 및 실행	SA30-1482	db2axx81
IBM DB2 Universal Database 응용프로그램 개발 안내서: 클라이언트 응용프로그램 프로그래밍	SA30-1483	db2a1x81
IBM DB2 Universal Database 응용프로그램 개발 안내서: 서버 응용프로그램 프로그래밍	SA30-1484	db2a2x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center 응용프로그램 통합 안내서	SA30-1510	db2adx81
IBM DB2 XML Extender 관리 및 프로그래밍	SA30-1516	db2sxx81

비즈니스 인텔리전스 정보

이 정보는 DB2 Universal Database의 분석적 기능 및 데이터 웨어하우징을 개선하는 구성요소 사용 방법을 설명합니다.

표 56. 비즈니스 인텔리전스 정보

이름	문서 번호	PDF 파일 이름
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition 설치 안내서	GA30-1508	db2idx81

표 56. 비즈니스 인텔리전스 정보 (계속)

이름	문서 번호	PDF 파일 이름
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

DB2 Connect 정보

이 범주의 정보는 DB2 Connect Enterprise Edition 또는 DB2 Connect Personal Edition을 사용하여 메인프레임 및 중간 범위 서버의 데이터에 액세스하는 방법을 설명합니다.

표 57. DB2 Connect 정보

이름	문서 번호	PDF 파일 이름
IBM 연결성 보충 설명서	문서 번호 없음	db2h1x81
DB2 Connect Enterprise Edition용 IBM DB2 Connect 빠른 시작	GA30-1489	db2c6x81
DB2 Connect Personal Edition용 IBM DB2 Connect 빠른 시작	GA30-1490	db2c1x81
IBM DB2 Connect 사용자 안내서	SA30-1491	db2c0x81

시작하기 정보

이 범주에 대한 정보는 서버, 클라이언트 및 기타 DB2 제품을 설치 및 구성하는 데 유용합니다.

표 58. 시작하기 정보

이름	문서 번호	PDF 파일 이름
DB2 Clients용 IBM DB2 Univeral Database 빠른 시작	GA30-1488, 하드카피로 사용할 수 없음	db2itx81
DB2 Servers용 IBM DB2 Universal Database 빠른 시작	GA30-1492	db2isx81
DB2 Personal Edition용 IBM DB2 Universal Database 빠른 시작	GA30-1494	db2i1x81
IBM DB2 Universal Database 설치 및 구성 보충 설명서	GA30-1493, 하드카피로 사용할 수 없음	db2iyx81
DB2 Data Links Manager용 IBM DB2 Universal Database 빠른 시작	GA30-1485	db2z6x81

자습서 정보

이 정보는 DB2 기능을 소개하고 다양한 태스크를 수행하는 방법을 설명합니다.

표 59. 자습서 정보

이름	문서 번호	PDF 파일 이름
비즈니스 인텔리전스 자습서: 데이터 웨어하우스 소개	문서 번호 없음	db2tux81
비즈니스 인텔리전스 자습서: 데이터 웨어하우징에서의 확장 레슨	문서 번호 없음	db2tax81
정보 카탈로그 센터 자습서	문서 번호 없음	db2aix81
e-business용 Video Central 자습서	문서 번호 없음	db2twx81
Visual Explain 자습서	문서 번호 없음	db2tvx81

선택적 구성요소 정보

이 범주에 대한 정보는 선택적 DB2 구성요소에 대한 작업을 수행하는 방법을 설명합니다.

표 60. 선택적 구성요소 정보

이름	문서 번호	PDF 파일 이름
IBM DB2 Cube Views 설치 및 사용자 안내서	SA30–1910	db2aax81
IBM DB2 Query Patroller 안내서 : 설치, 관리 및 사용	GA30–1921	db2dwx81
IBM DB2 Spatial Extender 및 Geodetic Extender 사용자 안내 및 참조서	SA30-1515	db2sbx81
IBM DB2 Universal Database Data Links Manager 관리 및 안내서 참조서	SA30-1507	db2z0x82
DB2 Net Search Extender 관리 및 사용자 안내서 **주:** 이 문서에 대한 HTML은 HTML 문서 CD로 부터 설치되지 *않습니다.*	SA30-1506	N/A

릴리스 정보

릴리스 정보는 제품 릴리스 및 FixPak 레벨별 추가 정보를 제공합니다. 또한 각각의 릴리스 및 FixPak에 통합된 문서 갱신사항을 요약합니다.

표 61. 릴리스 정보

이름	문서 번호	PDF 파일 이름
DB2 릴리스 정보	주 참조	주 참조
DB2 설치 정보	제품 CD-ROM에서만 사용 가능	사용할 수 없음

주: 릴리스 정보는 다음에서 사용 가능합니다.

- 제품 CD의 XHTML 및 텍스트 형식
- PDF 문서 CD의 PDF 형식

*알려진 문제점 및 임시 해결책*과 *릴리스 간의 비호환성*에 대해 설명하는 릴리스 정보 부분도 DB2 정보 센터에 표시됩니다.

UNIX 기반 플랫폼에서 텍스트 형식의 릴리스 정보를 보려면 `Release.Notes` 파일을 참조하십시오. 이 파일은 `DB2DIR/Readme/%L` 디렉토리에 있으며, 여기서 `%L`은 로케일 이름을 나타내고 `DB2DIR`은 다음을 나타냅니다.

- AIX 운영 체제: `/usr/opt/db2_08_01`
- 다른 모든 UNIX 기반 운영 체제: `/opt/IBM/db2/V8.1`

관련 개념:

- 475 페이지의 『DB2 문서 및 도움말』

태스크 관련:

- 493 페이지의 『PDF 파일에서 DB2 책 인쇄』
- 494 페이지의 『인쇄된 DB2 책 주문』
- 495 페이지의 『DB2 도구에서 문맥 도움말 호출』

PDF 파일에서 DB2 책 인쇄

DB2 PDF 문서 CD의 PDF 파일에서 DB2 책을 인쇄할 수 있습니다. Adobe Acrobat Reader를 사용하여 책 전체 또는 특정 페이지를 인쇄할 수 있습니다.

전제조건:

Adobe Acrobat Reader가 설치되어 있는지 확인하십시오. Adobe Acrobat Reader를 설치해야 하는 경우, Adobe 웹 사이트(www.adobe.com)에서 제공됩니다.

프로시저:

PDF 파일에서 DB2 책을 인쇄하려면, 다음을 수행하십시오.

1. *DB2 PDF 문서* CD를 넣으십시오. UNIX 운영 체제인 경우, DB2 PDF 문서 CD를 마운트하십시오. UNIX 운영 체제에서 CD를 마운트하는 방법에 대한 세부사항은 *빠른 시작* 책을 참조하십시오.
2. index.htm을 여십시오. 이 파일은 브라우저 창에서 열립니다.
3. 보려는 PDF 제목을 누르십시오. PDF는 Acrobat Reader에서 열립니다.
4. **파일 → 인쇄**를 선택하여 책에서 원하는 부분을 인쇄하십시오.

관련 개념:

- 476 페이지의 『DB2 정보 센터』

태스크 관련:

- *DB2 Server용 빠른 시작*의 『CD-ROM 마운트(AIX)』
- *DB2 Server용 빠른 시작*의 『CD-ROM 마운트(HP-UX)』
- *DB2 Server용 빠른 시작*의 『CD-ROM 마운트(Linux)』
- 494 페이지의 『인쇄된 DB2 책 주문』
- *DB2 Server용 빠른 시작*의 『CD-ROM 마운트(Solaris 운영 환경)』

관련 참조:

- 488 페이지의 『DB2 PDF 및 인쇄 문서』

인쇄된 DB2 책 주문

하드카피 책을 사용하려면 다음의 세 가지 방법 중 하나로 주문할 수 있습니다.

프로시저:

인쇄된 책은 일부 국가 또는 지역에서 주문할 수 있습니다. 국가 또는 지역에서 서비스를 사용할 수 있는지 확인하려면 국가 또는 지역의 IBM 책 웹 사이트를 점검하십시오. 책을 주문할 수 있는 경우, 다음을 수행하십시오.

- IBM의 허가된 판매업체나 마케팅 담당자에게 문의하십시오. 해당 지역의 IBM 담당자를 찾으려면 IBM Worldwide Directory of Contacts(www.ibm.com/planetwide)를 점검하십시오.
- 미국은 1-800-879-2755로, 캐나다는 1-800-IBM-4YOU로 문의하십시오.
- IBM Publications Center(http://www.ibm.com/shop/publications/order)를 방문하십시오. 모든 국가에서 IBM Publications Center를 통해 책을 주문할 수 있는 것은 아닙니다.

DB2 제품이 사용 가능해질 때, 인쇄된 책은 *DB2 PDF 문서 CD*에서 PDF 형식으로 제공되는 책과 동일합니다. *DB2 정보 센터 CD*에 표시되는 인쇄 책의 내용도 동일합니다. 그러나 DB2 정보 센터 CD에는 PDF 책에서 볼 수 없는 일부 추가 내용도 들어 있습니다(예: SQL 관리 루틴 및 HTML 샘플). DB2 PDF 문서 CD에서 사용 가능한 모든 책을 하드카피로 주문할 수 있는 것은 아닙니다.

주: DB2 정보 센터는 PDF 또는 하드카피 책보다 더 자주 갱신됩니다. 최신 정보를 보려면 문서 갱신사항이 사용 가능할 때 이를 설치하거나, DB2 정보 센터(http://publib.boulder.ibm.com/infocenter/db2help/)를 참조하십시오.

태스크 관련:

- 493 페이지의 『PDF 파일에서 DB2 책 인쇄』

관련 참조:

- 488 페이지의『DB2 PDF 및 인쇄 문서』

DB2 도구에서 문맥 도움말 호출

문맥 도움말은 특정 창, 노트북, 마법사 또는 어드바이저와 연관된 태스크나 제어사항에 대한 정보를 제공합니다. 문맥 도움말은 그래픽 사용자 인터페이스가 있는 DB2 관리 및 개발 도구에서 사용 가능합니다. 다음 두 유형의 문맥 도움말이 있습니다.

- 각 창 또는 노트북에 있는 **도움말** 단추를 통해 액세스하는 도움말
- 마우스 커서를 필드 또는 제어 위에 올려 놓거나, 창, 노트북, 마법사 또는 어드바이저에서 필드 또는 제어를 선택한 후 F1을 누를 때 표시되는 팝업 정보 창인 정보 팝업 상자

도움말 단추를 사용하여 개요, 전제조건 및 태스크 정보에 액세스할 수 있습니다. 정보 팝업 상자는 개별 필드 및 제어사항에 대해 설명합니다.

프로시저:

문맥 도움말을 호출하려면 다음을 수행하십시오.

- 창 및 노트북 도움말을 보려면 DB2 도구 중 하나를 시작한 후, 창 또는 노트북을 여십시오. 창 또는 노트북의 오른쪽 하단 구석에서 **도움말** 단추를 눌러 문맥 도움말을 호출하십시오.

  각 DB2 도구 센터의 맨 위에 있는 **도움말** 메뉴 항목에서 문맥 도움말에 액세스할 수도 있습니다.

  마법사 또는 어드바이저 내에서 문맥 도움말을 보려면 첫 번째 페이지의 태스크 개요 링크를 누르십시오.
- 창 또는 노트북의 개별 제어사항에 대한 정보 팝업 상자 도움말을 보려면 제어를 누른 후, **F1**을 누르십시오. 제어 세부사항을 포함하는 팝업 정보가 노란색 창에 표시됩니다.

  주: 필드 또는 제어 위에 마우스 커서를 올려 놓음으로써 정보 팝업 상자를 표시하려면 도구 설정 노트북의 **문서** 페이지에서 **자동으로 정보 팝업 상자 표시** 선택란을 선택하십시오.

  정보 팝업 상자와 마찬가지로, 진단 팝업 정보는 컨텍스트별 도움말의 다른 양식이며, 데이터 입력 규칙을 포함합니다. 진단 팝업 정보는 올바르지 않거나 충분하지 않은 데이터를 입력할 때 나타나는 자주색 창에 표시됩니다. 진단 팝업 정보는 다음 경우에 표시됩니다.

  – 필수 필드

  – 뒤에 정확한 형식이 오는 데이터의 필드(예: 날짜 필드)

태스크 관련:

- 485 페이지의 『DB2 정보 센터 호출』
- 496 페이지의 『명령행 처리기에서 메시지 도움말 호출』
- 496 페이지의 『명령행 처리기에서 명령 도움말 호출』
- 497 페이지의 『명령행 처리기에서 SQL 상태 도움말 호출』
- 『DB2 정보 센터에 액세스: 개념 도움말』
- 『DB2 UDB 도움말 사용법: 일반 GUI 도움말』
- 『DB2 정보 센터에 액세스하기 위해 위치 설정: 일반 GUI 도움말』
- 『문서 등록 정보 설정: 일반 GUI 도움말』

명령행 처리기에서 메시지 도움말 호출

메시지 도움말은 메시지의 원인 및 오류에 대한 응답으로 수행해야 할 조치를 설명합니다.

프로시저:

메시지 도움말을 호출하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? XXXnnnnn
```

여기서, *XXXnnnnn*은 유효한 메시지 ID를 나타냅니다.

예를 들어, `? SQL30081`은 SQL30081 메시지에 대한 도움말을 표시합니다.

관련 개념:

- *메시지 참조서 볼륨 1*의 『메시지 소개』

관련 참조:

- *Command Reference*의 『db2 - Command Line Processor Invocation Command』

명령행 처리기에서 명령 도움말 호출

명령행 도움말은 명령행 처리기의 명령 구문을 설명합니다.

프로시저:

명령 도움말을 호출하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? command
```

여기서, *command*는 키워드이거나 전체 명령입니다.

예를 들어, ? catalog는 모든 CATALOG 명령에 대한 도움말을 표시하고 ? catalog database는 CATALOG DATABASE 명령에 대한 도움말만 표시합니다.

태스크 관련:

- 495 페이지의『DB2 도구에서 문맥 도움말 호출』
- 485 페이지의『DB2 정보 센터 호출』
- 496 페이지의『명령행 처리기에서 메시지 도움말 호출』
- 497 페이지의『명령행 처리기에서 SQL 상태 도움말 호출』

관련 참조:

- *Command Reference*의『db2 - Command Line Processor Invocation Command』

명령행 처리기에서 SQL 상태 도움말 호출

DB2 Universal Database는 SQL문의 결과가 될 수 있는 조건에 대한 SQLSTATE 값을 리턴합니다. SQLSTATE 도움말은 SQL 상태 및 SQL 상태 클래스 코드의 의미를 설명합니다.

프로시저:

SQL 상태 도움말을 호출하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? sqlstate 또는 ? class code
```

여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자를 나타냅니다.

예를 들어, ? 08003은 08003 SQL 상태에 대한 도움말을 표시하고 ? 08은 08 클래스 코드에 대한 도움말을 표시합니다.

태스크 관련:

- 485 페이지의『DB2 정보 센터 호출』
- 496 페이지의『명령행 처리기에서 메시지 도움말 호출』
- 496 페이지의『명령행 처리기에서 명령 도움말 호출』

DB2 자습서

DB2® 자습서를 사용하여 DB2 Universal Database의 다양한 측면을 학습할 수 있습니다. 자습서는 응용프로그램 개발, SQL 쿼리 성능 조정, 데이터 웨어하우스에 대한 작업, 메타데이터 관리 및 DB2를 사용한 웹 서비스 개발 영역에 대한 단계별 지시사항 및 레슨을 제공합니다.

시작하기 전에:

http://publib.boulder.ibm.com/infocenter/db2help/의 정보 센터에서 자습서의 XHTML 버전을 볼 수 있습니다.

일부 자습서 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크의 전제조건에 대한 설명은 각각의 자습서를 참조하십시오.

DB2 Universal Database 자습서:

자습서를 보려면 다음 목록에서 해당 자습서 제목을 누르십시오.

비즈니스 인텔리전스 자습서: Data Warehouse Center 소개
Data Warehouse Center를 사용하여 데이터 웨어하우징 태스크를 소개합니다.

비즈니스 인텔리전스 자습서: 데이터 웨어하우징의 확장 레슨
Data Warehouse Center를 사용하여 고급 데이터 웨어하우징 태스크를 수행합니다.

정보 카탈로그 센터 자습서
정보 카탈로그 센터를 사용하여 메타데이터를 찾아 사용하기 위해 정보 카탈로그를 작성 및 관리합니다.

Visual Explain 자습서
Visual Explain을 사용하여 성능을 향상시키기 위해 SQL문을 분석, 최적화 및 조정합니다.

DB2 문제점 해결 정보

DB2® 제품을 사용하는 데 있어 도움이 되는 광범위한 문제점 해결 및 문제점 판별 정보가 있습니다.

DB2 문서
문제점 해결 정보는 DB2 정보 센터 및 DB2 라이브러리를 구성하는 PDF 책 전체에서 찾을 수 있습니다. DB2 정보 센터 탐색 트리(브라우저 창의 왼쪽 분할창에서)의 "지원 및 문제점 해결" 분기에서 DB2 문제점 해결 문서의 전체 목록을 볼 수 있습니다.

DB2 기술 지원 웹 사이트
문제가 있는 경우, 가능한 원인 및 솔루션을 찾을 수 있는 도움말을 보려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 책, 기술 정보, APAR(Authorized Program Analysis Report), FixPak 및 내부 DB2 오류 코드의 최신 목록에 대한 링크 및 기타 자원이 포함되어 있습니다. 이러한 기술 자료를 검색하고 문제에 대해 가능한 솔루션을 찾을 수 있습니다.

DB2 기술 지원 웹 사이트 http://www.ibm.com/software/data/db2/udb/winos2unix/support에 액세스하십시오.

DB2 문제점 판별 자습서 시리즈

DB2 제품을 사용하는 동안 발생할 수 있는 문제점을 빠르게 식별하고 해결하는 방법에 대한 정보를 보려면 DB2 문제점 판별 자습서 시리즈 웹 사이트를 참조하십시오. 그 중 하나의 자습서가 사용 가능한 DB2 문제점 판별 기능 및 도구를 소개하고 사용하는 시기를 결정하는 데 도움을 줍니다. 기타 자습서는 "데이터베이스 엔진 문제점 판별", "성능 문제점 판별" 및 "응용프로그램 문제점 판별"과 같은 관련 항목을 다룹니다.

DB2 기술 지원 사이트 http://www.ibm.com/software/data/support/pdm/db2tutorials.html에서 DB2 문제점 판별 자습서의 전체 세트를 참조하십시오.

관련 개념:

- 476 페이지의 『DB2 정보 센터』
- *Troubleshooting Guide*에서 『Introduction to problem determination - DB2 Technical Support tutorial』

액세스 기능

액세스 가능성 기능을 사용하면 거동이 불편하거나 시력 장애가 있는 사용자와 같이 신체적 장애가 있는 사용자가 소프트웨어 제품을 편리하게 사용할 수 있습니다. 다음은 DB2® 버전 8 제품에서 제공하는 주요 액세스 기능입니다.

- DB2에서는 마우스 대신 키보드를 사용하여 모든 기능을 조작할 수 있습니다. 자세한 정보는 『키보드 입력 및 탐색』을 참조하십시오.
- DB2 사용자 인터페이스에서 글꼴 크기 및 색상을 사용자 정의할 수 있습니다. 자세한 정보는 500 페이지의 『액세스 가능한 표시』를 참조하십시오.
- DB2 제품이 Java™ Accessibility API를 사용하는 특수 액세스 기능 응용프로그램을 지원합니다. 자세한 정보는 500 페이지의 『보조 기술과의 호환성』을 참조하십시오.
- DB2 문서가 액세스 가능 형식으로 제공됩니다. 자세한 정보는 500 페이지의 『액세스 가능한 문서』를 참조하십시오.

키보드 입력 및 탐색

키보드 입력

키보드만 사용하여 DB2 도구를 조작할 수 있습니다. 키 또는 키 조합을 사용하여, 마우스를 사용하여 수행할 수 있는 조작을 수행할 수 있습니다. 표준 운영 체제 조작을 위해 표준 운영 체제 키 입력이 사용됩니다.

키 또는 키 조합을 사용하여 조작을 수행하는 방법에 대한 자세한 정보는 단축키: 일반 GUI 도움말을 참조하십시오.

키보드 탐색

키 또는 키 조합을 사용하여 DB2 도구 사용자 인터페이스를 탐색할 수 있습니다.

키 또는 키 조합을 사용하여 DB2 도구를 탐색하는 방법에 대한 자세한 정보는 단축키 : 일반 GUI 도움말을 참조하십시오.

키보드 초점

UNIX® 운영 체제에서, 키 입력이 적용되는 활성 창 영역은 강조표시됩니다.

액세스 가능한 표시

DB2 도구에는 시력이 낮거나 시각 장애가 있는 사용자를 위해 액세스 가능성을 향상시키는 기능이 있습니다. 이러한 액세스 가능성 개선 사항에는 사용자 정의가 가능한 글꼴 등록 정보에 대한 지원이 포함됩니다.

글꼴 설정

도구 설정 노트북을 사용하여 메뉴 및 대화 상자 창에 표시되는 텍스트의 색상, *크기* 및 글꼴을 선택할 수 있습니다.

글꼴 설정 지정에 대한 자세한 정보는 메뉴 및 텍스트의 글꼴 변경: 일반 GUI 도움말을 참조하십시오.

색상과 무관

이 제품의 기능을 사용하기 위해 색상을 구분할 필요는 없습니다.

보조 기술과의 호환성

DB2 도구 인터페이스는 DB2 제품에서 화면 판독기 및 기타 보조 기술을 사용할 수 있도록 하는 Java Accessibility API를 지원합니다.

액세스 가능한 문서

DB2 문서는 대부분의 웹 브라우저에서 볼 수 있는 XHTML 1.0 형식으로 제공됩니다. XHTML은 브라우저의 표시 환경설정에 따라 문서를 볼 수 있도록 합니다. 또한 화면 판독기 및 기타 보조 기술을 사용할 수 있습니다.

구문 다이어그램은 점분리 10진수 형식으로 제공됩니다. 이 형식은 화면 판독기를 사용하여 온라인 문서에 액세스하는 경우에만 사용할 수 있습니다.

관련 개념:

- 501 페이지의『점분리 10진수 구문 다이어그램』

태스크 관련:

- 『단축키: 일반 GUI 도움말』
- 『메뉴 및 텍스트의 글꼴 변경: 일반 GUI 도움말』

점분리 10진수 구문 다이어그램

구문 다이어그램은 화면 판독기를 사용하여 정보 센터에 액세스하는 사용자를 위해 점분리 10진수 형식으로 제공됩니다.

점분리 10진수 형식으로 각 구문 요소는 별도의 행에 작성됩니다. 두 개 이상의 구문 요소가 항상 함께 표시되는 경우(또는 항상 함께 표시되지 않는 경우), 이러한 구문 요소는 하나의 복합 구문 요소로 간주되어 동일한 행에 표시될 수 있습니다.

각 행은 점분리 10진수(예: 3, 3.1, 3.1.1)로 시작합니다. 이 숫자를 올바르게 인식하려면 화면 판독기가 구두점을 판독하도록 설정되었는지 확인하십시오. 동일한 점분리 10진수가 있는 모든 구문 요소(예: 3.1이 있는 모든 구문 요소)는 서로 대체하여 사용할 수 없습니다. 3.1 USERID와 3.1 SYSTEMID 행을 인식하면, 구문에 둘다가 아니라 USERID 또는 SYSTEMID 중에서 하나를 포함할 수 있습니다.

점분리 10진수를 지정하는 레벨은 중첩 레벨입니다. 예를 들어, 점분리 10진수 3의 구문 요소 다음에 점분리 10진수 3.1의 여러 구문 요소가 오는 경우, 3.1의 모든 구문 요소는 3의 구문 요소에 종속됩니다.

특정 단어 및 기호는 점분리 10진수 다음에 사용되어 구문 요소에 대한 정보를 추가합니다. 때때로 이 단어 및 기호는 구문 요소가 자체적으로 시작할 때 발생할 수도 있습니다. 식별하기 쉽도록 단어 또는 기호가 구문 요소의 일부분이면 앞에 백슬래시(\) 문자가 옵니다. * 기호는 점분리 10진수 다음에 사용되어 구문 요소의 반복을 표시합니다. 예를 들어, 점분리 10진수 3의 구문 요소 *FILE은 3 * FILE 형식으로 지정됩니다. 3* FILE 형식은 구문 요소 FILE의 반복을 표시합니다. 3* * FILE 형식은 구문 요소 * FILE의 반복을 표시합니다.

쉼표와 같은 문자는 구문 요소 문자열을 구분하는 데 사용되어, 구문에서 구분되는 항목 바로 앞에 표시됩니다. 이 문자는 각 항목과 동일한 행에 표시되거나 관련 항목과 동일한 점분리 10진수가 있는 별도의 행에 표시될 수 있습니다. 행은 구문 요소에 대한 정보를 지정하는 다른 기호도 표시할 수 있습니다. 예를 들어, 5.1*, 5.1 LASTRUN 및 5.1 DELETE 행은 둘 이상의 LASTRUN 및 DELETE 구문 요소를 사용할 경우에 요소를 쉼표로 구분해야 한다는 의미입니다. 구분자를 지정하지 않은 경우, 공백을 사용하여 각 구문 요소를 구분한다고 가정합니다.

구문 요소 앞에 % 기호가 오는 경우, 다른 위치에서 정의된 참조를 표시합니다. % 기호 다음의 문자열은 리터럴의 이름이기 보다는 구문 조각의 이름입니다. 예를 들어, 2.1 %OP1 행은 구문 조각 OP1을 구분해야 한다는 의미입니다.

특정 단어 및 기호는 점분리 10진수 다음에 사용됩니다.

- ? 는 선택적 구문 요소임을 의미합니다. 점분리 10진수 뒤에 오는 ? 기호는 대응하는 점분리 10진수가 있는 모든 구문 요소 및 그 하위 구문 요소가 선택적이라는 표

시입니다. 점분리 10진수가 있는 구문 요소가 유일한 경우, ? 기호는 구문 요소와 동일한 행에 표시됩니다(예: 5? NOTIFY). 점분리 10진수가 있는 구문 요소가 둘 이상인 경우, ? 기호는 뒤에 선택적인 구문 요소가 오는 행에 자체적으로 표시됩니다. 예를 들어, 5 ?, 5 NOTIFY 및 5 UPDATE 행의 경우, 구문 요소 NOTIFY 및 UPDATE는 선택적입니다. 즉 둘 중 하나를 선택하거나 선택하지 않을 수 있습니다. ? 기호는 철도 다이어그램의 생략 행과 같습니다.

- ! 는 디폴트 구문 요소임을 의미합니다. 점분리 10진수 뒤에 오는 ! 기호는 구문 요소가 동일한 점분리 10진수를 공유하는 모든 구문 요소의 디폴트 옵션이라는 표시입니다. 동일한 점분리 10진수를 공유하는 구문 요소 하나에만 ! 기호를 지정할 수 있습니다. 예를 들어, 2? FILE, 2.1! (KEEP) 및 2.1 (DELETE) 행을 인식한 경우, (KEEP)은 FILE 키워드의 디폴트 옵션입니다. 이 예에서 FILE 키워드를 포함하지만 옵션을 지정하지 않은 경우, 디폴트 옵션 KEEP이 적용됩니다. 디폴트 옵션은 다음 상위 점분리 10진수에도 적용됩니다. 이 예에서 FILE 키워드를 생략한 경우, 디폴트값 FILE(KEEP)을 사용합니다. 그러나 2? FILE, 2.1, 2.1.1! (KEEP) 및 2.1.1 (DELETE) 행을 인식한 경우, 디폴트 옵션 KEEP을 다음 상위 점분리 10진수 2.1(연관된 키워드가 없는)에만 적용하고 2? FILE에는 적용하지 않습니다. 키워드 FILE을 생략하는 경우에는 사용되지 않습니다.
- *는 0번 이상 반복할 수 있는 구문 요소입니다. 뒤에 * 기호가 오는 점분리 10진수는 이 구문 요소를 0번 이상 사용할 수 있다는 표시입니다. 즉 선택적으로 반복할 수 있다는 의미입니다. 예를 들어, 행 5.1* 데이터 영역을 인식한 경우, 하나의 데이터 영역 또는 둘 이상의 데이터 영역을 포함시키거나 데이터 영역을 포함시키지 않을 수 있습니다. 3*, 3 HOST 및 3 STATE 행을 인식하면 HOST, STATE 또는 둘 다 포함하거나 전혀 포함하지 않을 수 있습니다.

  주:

  1. 점분리 10진수 다음에 별표(*)가 있고 이 점분리 10진수를 가진 항목이 하나인 경우, 동일한 항목을 두 번 이상 반복할 수 있습니다.
  2. 점분리 10진수 다음에 별표(*)가 있고 이 점분리 10진수를 가진 여러 항목이 있는 경우, 목록에서 둘 이상의 항목을 사용할 수 있습니다. 앞의 예에서 HOST STATE는 쓸 수 있지만 HOST HOST는 쓸 수 없습니다.
  3. * 기호는 철도 구문 다이어그램의 루프백 행과 같습니다.
- +는 한 번 이상 포함되어야 하는 구문 요소를 의미합니다. 뒤에 + 기호가 오는 점분리 10진수는 이 구문 요소가 한 번 이상 포함되어야 함을 나타냅니다. 즉, 최소 한 번 포함되어야 하고 반복할 수 있습니다. 예를 들어, 행 6.1+ 데이터 영역을 인식한 경우, 최소 하나의 데이터 영역을 포함시켜야 합니다. 행 2+, 2 HOST 및 2 STATE를 인식한 경우, HOST, STATE 또는 둘다를 포함시켜야 합니다. * 기호와 마찬가지로, + 기호는 특정 항목이 해당 점분리 10진수가 있는 유일한 항목인 경우에 항목을 반복할 수 있습니다. * 기호와 유사한 + 기호는 철도 구문 다이어그램의 루프백 행과 동등합니다.

관련 개념:

- 499 페이지의 『액세스 기능』

태스크 관련:

- 『단축키: 일반 GUI 도움말』

관련 참조:

- *SQL 참조서, 볼륨 2*의 『구문 도표를 읽는 방법』

DB2 Universal Database 제품의 일반 기준 인증

DB2 Universal Database의 인증은 평가 보증 레벨 4(EAL4)의 일반 기준 항목에서 평가됩니다. 일반 항목에 대한 자세한 내용은 http://niap.nist.gov/cc-scheme/을 참조하십시오.

부록 L. 주의사항

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.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-Ku
Tokyo 106, 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 면책 사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통고없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 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) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이센스 사용자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 사용료 지불 포함) 사용할 수 있습니다.

이 정보에 기술된 라이센스가 있는 프로그램 및 이 프로그램에 대해 사용 가능한 모든 라이센스가 있는 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 성능은 개발 레벨 상태의 시스템에서 측정되었을 수 있으므로, 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한, 일부 성능은 추정치일 수도 있으므로 실제 결과는 다를 수 있습니다. 이 문서의 사용자는 해당 데이터를 사용자의 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.

IBM의 향후 방향 또는 의도에 관한 모든 언급은 별도의 통지없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이들 샘플 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 암시하지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 그 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. _연도_. All rights reserved.

상표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표로서 DB2 UDB 문서 라이브러리에 있는 문서 중 적어도 하나 이상에 사용되었습니다.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

다음 용어는 기타 회사의 상표 또는 등록상표로서, DB2 UDB 문서 라이브러리에 있는 문서 중 적어도 하나 이상에 사용되었습니다.

Microsoft, Windows, Windows NT 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

Intel 및 Pentium은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표입니다.

Java 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.

UNIX는 미국 및 기타 국가에서 사용되는 Open Group의 등록상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

C

D

E

F

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

[특수 문자]

IBM에 문의

미국에서는 다음 번호로 IBM에 문의하십시오.

- 고객 서비스를 받으려면 1-800-IBM-SERV(1-800-426-7378)
- 사용 가능한 서비스 옵션을 알려면 1-888-426-4343
- DB2 마케팅 및 판매에 대해서는 1-800-IBM-4YOU(426-4968)

캐나다에서는 다음 번호로 IBM에 문의하십시오.

- 고객 서비스를 받으려면 1-800-IBM-SERV(1-800-426-7378)
- 사용 가능한 서비스 옵션을 알려면 1-800-465-9600
- DB2 마케팅 및 판매에 대해서는 1-800-IBM-4YOU(1-800-426-4968)

해당 국가 및 지역의 IBM 지사를 찾으려면 http://www.ibm.com/planetwide에서 IBM의 Directory of Worldwide Contacts를 확인하십시오.

제품 정보

DB2 Universal Database 제품에 관한 정보는 전화 또는 http://www.ibm.com/software/data/db2/udb의 WWW(World Wide Web)에서 사용 가능합니다.

이 사이트에는 기술 라이브러리, 책 주문, 제품 다운로드, 뉴스 그룹, FixPak, 뉴스 및 웹 자원 링크에 대한 최신 정보가 포함되어 있습니다.

미국에 거주하는 분은 다음 번호 중 하나를 선택하여 문의하십시오.

- 제품을 주문하거나 일반 정보를 얻으려면 1-800-IBM-CALL(1-800-426-2255)
- 책을 주문하려면 1-800-879-2755

미국 이외의 지역에서 IBM에 문의하는 방법에 대한 정보는 www.ibm.com/planetwide의 IBM Worldwide 페이지를 참조하십시오.

IBM 한글 지원에 관한 설문

IBM® DB2 Universal Database™
관리 안내서: 구현
버전 8.2

SA30-1479-01

FAX : (02) 3787-0123

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다. 독자 여러분의 좋은 의견을 기다립니다.

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까?
   그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다.
   어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

SA30-1479-01

Spine information:

IBM® DB2™ Universal Database

관리 안내서: 구현

버전 8.2