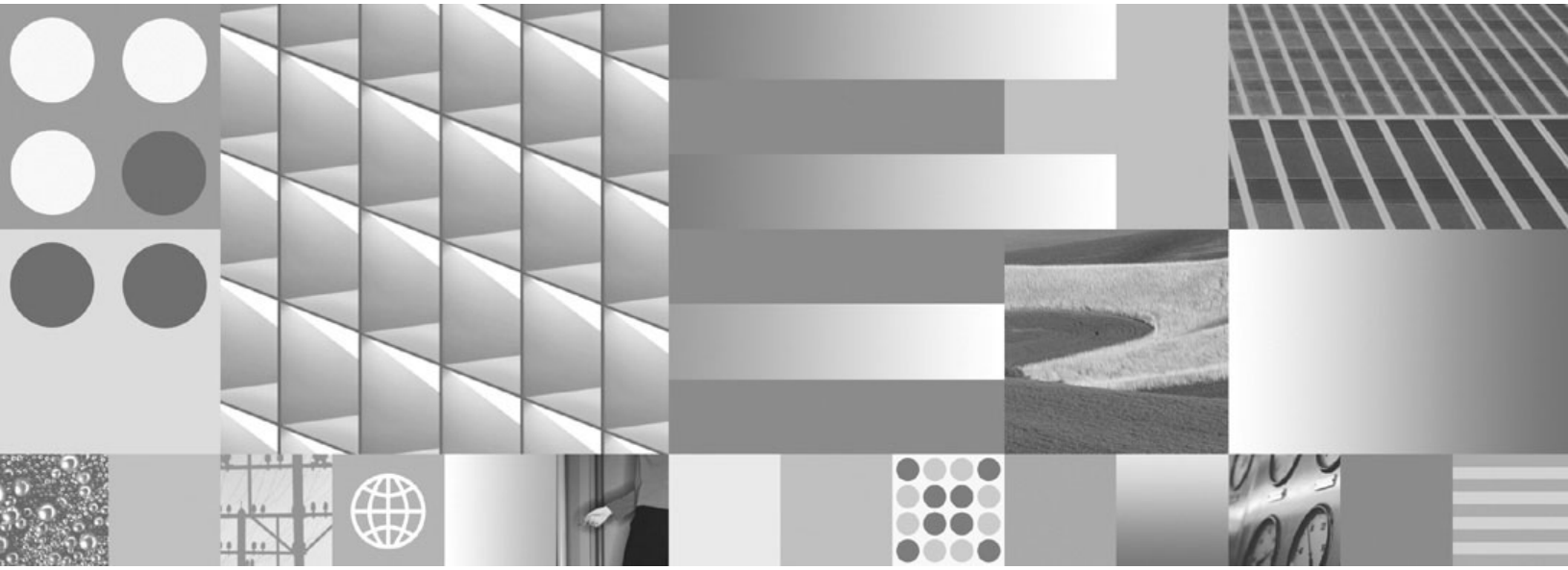
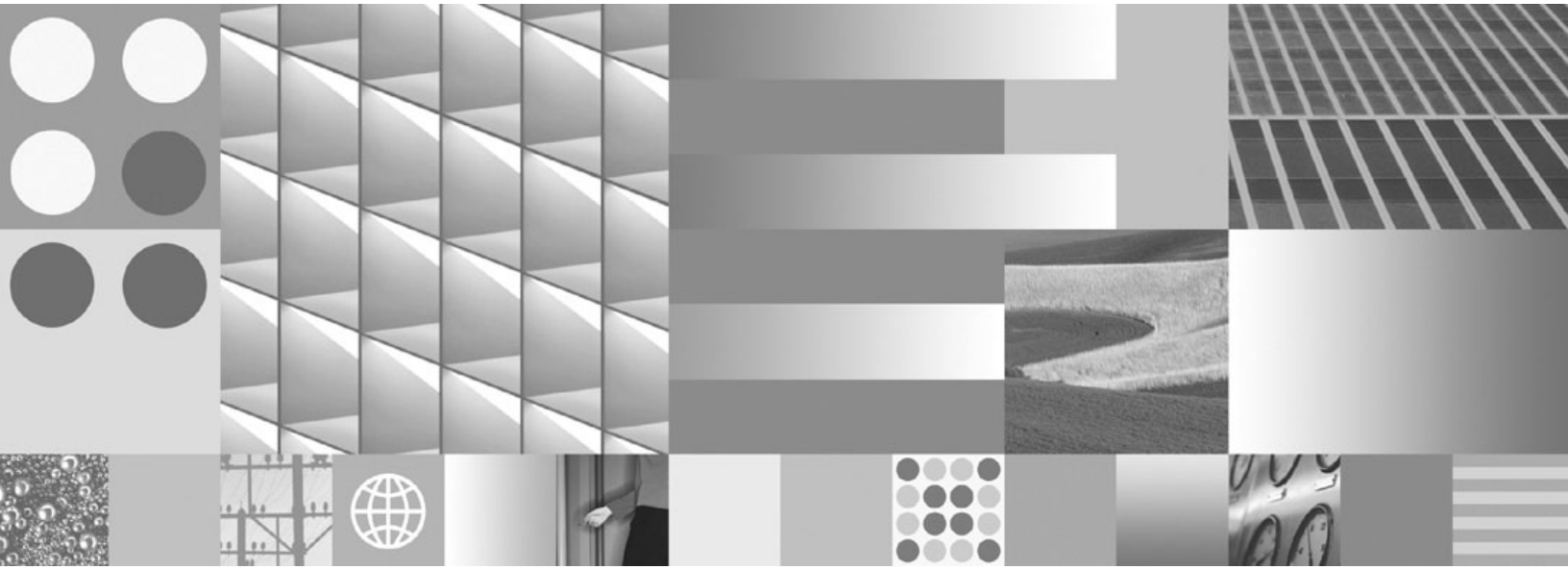
Linux, UNIX 및 Windows용
IBM DB2 9.7



파티셔닝 및 클러스터링 안내서

SA30-3973-00

Linux, UNIX 및 Windows용
IBM DB2 9.7



파티셔닝 및 클러스터링 안내서

SA30-3973-00

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 509 페이지의 부록 E『주의사항』에서 일반 정보를 읽어 보십시오.

개정판 주의사항

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

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

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

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

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

목차

이 책에 대한 정보

파티셔닝 및 클러스터링 안내서를 시작합니다!

DB2® 관계형 데이터베이스 관리 시스템은 관리자 및 시스템 운영자가 데이터베이스 성능을 효과적으로 향상시키고 하드웨어 자원에 많은 데이터베이스 오브젝트를 분배하도록 하는 파티셔닝 및 클러스터링 기능에 매우 영향을 받습니다. 병렬 처리와 스토리지 용량을 사용하기 위해 빠른 데이터 검색 및 계속 증가하는 하드웨어 자원에 오브젝트를 분배하는 기능은 궁극적으로 생산성을 향상시킵니다. 이 책에는 DB2 라이브러리의 주제를 구성한 콜렉션이 있으며 이는 데이터베이스 파티션, 테이블 파티션, 테이블 클러스터, 테이블 범위 클러스터, 다차원적으로 클러스터된 테이블 및 병렬 처리의 계획, 설계, 구현, 사용 및 유지보수에만 집중된 단일 종합 정보 소스입니다.

이 책의 사용자

이 책은 기본적으로 로컬 및 원격 클라이언트가 액세스하는 파티션되거나 클러스터된 데이터베이스를 설계, 구현 또는 유지보수해야 하는 데이터베이스 관리자, 시스템 관리자, 보안 관리자 및 시스템 운영자를 대상으로 합니다. 이 책에는 파티셔닝, 클러스터링 및 병렬 처리 기능이 포함되어 있으므로 DB2 관계형 데이터베이스 관리 시스템의 종합 정보 소스와 관리 및 운영에 대한 이해가 모두 필요한 응용프로그램 개발자 및 기타 사용자도 이 책을 사용할 수 있습니다. 향후 여기에서 설명된 주요 기능을 모두 또는 일부 구현하려는 사용자에게 이 책은 훌륭한 정보 자원이 될 것입니다.

이 책의 구성

DB2 라이브러리의 주제를 모은 이 콜렉션은 DB2 파티셔닝, 클러스터링 및 병렬 처리 기능에만 집중된 단일 종합 정보 소스입니다. 편리성과 효율성을 위해 이 책은 여섯 개의 부분으로 나뉘어져 있으며 이 중 처음 다섯 부분은 관리자, 시스템 운영자 및 응용프로그램 개발자를 고려한 주요 관리 주제에 대한 것입니다. 이 책의 주요 부분에 포함된 주제는 DB2 라이브러리의 다른 책의 내용을 나타내는 주제와 맵핑될 수 있으므로 주제가 다른 DB2 기능 및 오브젝트의 호스트와 관련된 경우 자세한 일반 정보를 쉽게 상호 참조할 수 있습니다. 예를 들어 4 부의 20 장에서 다차원적으로 클러스터된 테이블 최적화 전략이 향상된 성능을 나타내는 방법에 대한 주제를 읽은 후 특정 예제 주제가 맵핑되는 *데이터베이스 성능 조정* 책을 참고하여 구성될 수 있는 일반 테이블의 기타 일반 성능 향상을 조사할 수도 있습니다. 아래의 테이블 1에서 유사한 주제에 따라 다른 DB2 오브젝트 및 기능에 대한 추가 정보를 참조할 수 있는 다른 책과 맵핑된 이 책의 주요 부분을 볼 수 있습니다.

표 1. DB2 라이브러리의 다른 책에 맵핑된 이 책의 부분

파티셔닝 및 클러스터링 안내서의 부분	DB2 라이브러리의 책에 맵핑
제 1 부. 계획 및 설계 고려사항	*데이터베이스 관리 개념 및 구성 참조서* *데이터베이스 보안 안내서*
제 2 부. 설치 고려사항	*데이터베이스 관리 개념 및 구성 참조서* *DB2 Server 설치*
제 3 부. 구현 및 유지보수	*데이터 이동 유틸리티 안내서 및 참조서* *데이터 복구 및 고가용성 안내서 및 참조서* *데이터베이스 관리 개념 및 구성 참조서* *DB2 버전 9.7로 업그레이드* *데이터베이스 모니터링 안내서 및 참조서* *Visual Explain 자습서* *XQuery Reference*
제 4 부. 성능 문제	*데이터베이스 관리 개념 및 구성 참조서* *문제점 해결 및 데이터베이스 성능 조정* *Visual Explain 자습서*
제 5 부. 관리 API, 명령, SQL문	*관리 API 참조서* *관리 루틴 및 뷰* *명령어 참조서* *Developing ADO.NET and OLE DB Applications* *Developing Embedded SQL Applications* *Developing Java Applications* *Developing Perl, PHP, Python, and Ruby on Rails Applications* *Developing User-defined Routines(SQL and External)* *Getting Started with Database Application Development* *SQL 참조서, 볼륨 1* *SQL 참조서, 볼륨 2*
제 6 부. 부록	*데이터 복구 및 고가용성 안내서 및 참조서* *DB2 Server 설치* *SQL 참조서, 볼륨 1*

이 책의 여러 장에 설명된 주요 주제 영역은 다음과 같습니다.

제 1 부. 계획 및 설계 고려사항

다음의 모든 장에는 파티션닝, 클러스터링 또는 병렬 데이터베이스 시스템에 사용될 데이터베이스/테이블의 계획 및 설계와 관련된 개념 정보가 있습니다.

- 제 1 장 『파티션된 데이터베이스 및 테이블』에서는 데이터베이스와 테이블을 파티셔닝하는 기능 및 장점에 대한 관련 개념을 소개합니다.
- 제 2 장 『범위 클러스터된 테이블』에서 범위 클러스터된 테이블 사용의 기능 및 장점과 관련된 일반 개념 정보가 있습니다.
- 제 3 장 『다차원적으로 클러스터된(MDC) 테이블』에서는 테이블의 데이터를 클러스터링하는 명쾌한 방법인 다차원 클러스터링 사용에 대해 설명합니다.
- 제 4 장 『병렬 데이터베이스 시스템』에서는 병렬 처리를 사용하여 성능을 극적으로 향상시키는 방법에 대해 설명합니다.

제 2 부. 설치 고려사항

다음 장에서는 데이터베이스 파티셔닝 준비에 필요한 사전 설치 및 설치 태스크에 대한 정보를 제공합니다.

- 제 5 장 『설치 전제조건』에서는 파티션된 데이터베이스 환경에 포함될 DB2 서버 준비와 관련된 전제조건 및 제한사항에 대해 설명합니다.
- 제 6 장 『설치 전』에서는 UNIX® 및 Linux® 시스템의 경우 추가 사전 설치 태스크 및 고려사항에 대해 설명합니다.
- 제 7 장 『DB2 서버 제품 설치』에서는 데이터베이스 파티션 서버 설치 방법 및 파티션된 데이터베이스 환경 설정 방법에 대해 설명합니다.
- 제 8 장 『설치 후』에서는 Windows®, UNIX 및 Linux 시스템에서 설치를 검증하는 방법에 대해 설명합니다.

제 3 부. 구현 및 유지보수

계획, 설계 및 설치 단계가 완료되면 다음 장에서 이미 준비가 된 기능 및/또는 오브젝트를 구현하고 유지보수하는 방법에 대해 설명합니다.

- 제 9 장 『데이터베이스 작성 전에』에서는 병렬 처리 사용, 파티션된 데이터베이스 환경 작성, 노드/파티션 작성 및 구성 그리고 데이터베이스와 파티션 간의 통신 설정과 같은 데이터베이스 작성 전에 고려해야 할 사항에 대해 설명합니다.
- 제 10 장 『파티션된 데이터베이스 환경 작성 및 관리』에서는 데이터베이스 파티션 및 파티션 그룹의 작성 및 관리 방법에 대해 설명합니다.
- 제 11 장 『테이블 및 관련 테이블 오브젝트 작성』에서는 파티션된 테이블, 범위 클러스터된 테이블 및 MDC 테이블 작성 방법에 대한 정보를 제공합니다.
- 제 12 장 『데이터베이스 변경』에서는 인스턴스 및/또는 데이터베이스 변경 방법에 대해 설명합니다.

- 제 13 장『테이블 및 다른 관련 테이블 오브젝트 변경』에서는 파티션된 테이블의 수정 방법에 대한 정보를 제공합니다.
- 제 14 장『로드』에서는 병렬 처리, 다차원적으로 클러스터된 테이블 및 파티션된 테이블의 경우 로드 고려사항에 대해 설명합니다.
- 제 15 장『파티션된 데이터베이스 환경에 데이터 로딩』에서는 파티션된 데이터베이스 환경에서 데이터 로드 조작을 시작, 다시 시작 또는 종료하는 방법에 대해 설명합니다.
- 제 16 장『파티션된 데이터베이스 환경의 이주』에는 파티션된 데이터베이스 이주에 대한 간단한 개요 및 자세한 정보를 위한 참조가 있습니다.
- 제 17 장『스냅샷 및 이벤트 모니터』에는 CREATE EVENT MONITOR 문 사용 방법에 대한 설명과 더불어 테이블 재구성을 모니터하거나 파티션된 데이터베이스 시스템의 글로벌 상태를 평가하기 위한 스냅샷 모니터 결과 사용에 대한 관련 정보가 있습니다.
- 제 18 장『좋은 백업 및 복구 전략 개발』에서는 장애가 발생하기 전에 백업 및 복구 전략 개발에 도움이 되는 파티션된 데이터베이스 환경에서의 응급 복구에 대한 개념을 설명합니다.
- 제 19 장『문제점 해결』에는 문제점 해결에 대한 간단한 개요와 인스턴스의 모든 컴퓨터 또는 모든 데이터베이스 파티션 서버에 db2trc와 같은 문제점 해결에 사용할 수 있는 명령을 발행하는 방법에 대한 유용한 정보가 있습니다.

제 4 부. 성능 문제

다음 장에는 파티션된 환경 및/또는 클러스터된 환경의 성능을 향상시킬 수 있는 관련 정보가 있습니다.

- 제 20 장『데이터베이스 설계 시 성능 문제』에서는 테이블 파티셔닝 및 다차원 클러스터링의 최적화 전략과 함께 이 두 가지를 향상시키는 기능에 대해 설명합니다.
- 제 21 장『인덱스』에는 파티션된 테이블의 인덱스를 이해하는데 도움이 되는 개념 정보가 있습니다.
- 제 22 장『디자인 어드바이저』에서는 디자인 어드바이저를 사용하여 데이터 분배 및 새 인덱스 작성에 대한 권장사항, 구체화된 쿼리 테이블 및 다차원적으로 클러스터된 테이블 뿐만 아니라 단일 파티션에서 다중 파티션 데이터베이스로의 이주에 대한 정보를 얻는 방법에 대해 설명합니다.
- 제 23 장『동시성 관리』에서는 잠금 모드에 대한 정보를 제공합니다.
- 제 24 장『에이전트 관리』에서는 서비스 응용프로그램 요청에 익숙한 데이터베이스 에이전트를 최적화하는 방법에 대해 설명합니다.
- 제 25 장『액세스 최적화 계획』에는 액세스 플랜 향상 방법, 옵티마이저가 데이터 액세스 전략을 최적화하기 위해 다양한 스캔의 정보를 사용하는 방

법에 대한 설명과 파티션된 데이터베이스 환경, 클러스터된 테이블 및/또는 병렬 처리를 사용하는 시스템에서 성능을 향상시킬 수 있는 조인 전략에 대한 정보가 있습니다.

- 제 26 장 『데이터 재분배』에서는 데이터 재분배 수행 여부를 판별하고 재분배를 해야할 경우 데이터베이스 파티션에 데이터를 재분배하는 방법에 대해 설명합니다.
- 제 27 장 『자체 성능 조정 메모리 구성』에서는 파티션된 데이터베이스 환경에서 자체 성능 조정 메모리 기능 사용에 대해 설명하고 구성 권장사항을 제공합니다.
- 제 28 장 『DB2 구성 매개변수 및 변수』에서는 다중 파티션에 데이터베이스 구성 매개변수 및 환경 변수 설정 방법에 대한 정보를 제공하고 파티션된 데이터베이스 환경 및 병렬 처리 기능과 관련된 매개변수 및 변수를 나열합니다.

제 5 부. 관리 API, 명령, SQL문

다음 장에서는 파티션된 데이터베이스 환경과 관련된 API, 명령 및 SQL 요소에 대한 정보를 종합적으로 통합합니다.

- 제 29 장 『관리 API』에서는 파티션된 데이터베이스 환경에만 관련된 API에 대한 정보를 제공합니다.
- 제 30 장 『명령』에서는 파티션된 데이터베이스 환경에만 관련된 명령에 대한 정보를 제공합니다.
- 제 31 장 『SQL 언어 요소』에서는 데이터베이스 파티션 호환 가능 데이터 유형 및 특수 레지스터를 제공합니다.
- 제 32 장 『SQL 함수』에서는 파티션된 데이터베이스 환경에만 관련된 SQL 함수에 대해 설명합니다.
- 제 33 장 『SQL문』에서는 파티션된 데이터베이스 환경에만 관련된 SQL문에 대해 설명합니다.
- 제 34 장 『지원되는 관리 SQL 루틴 및 뷰』에서는 파티션된 데이터베이스 환경에만 관련된 SQL 루틴 및 뷰에 대해 설명합니다.

제 6 부. 부록

- 부록 A 『비루트 사용자로 설치』에서는 UNIX 및 Linux 시스템에 비루트 사용자로 DB2 제품 설치에 대해 설명합니다.
- 부록 B 『백업 사용』에서는 BACKUP DATABASE 명령을 사용하는 방법에 대해 설명합니다.
- 부록 C 『파티션된 데이터베이스 환경 카탈로그 뷰』에서는 파티션된 데이터베이스 환경에 특정한 카탈로그 보기를 나열합니다.

강조표시 규칙

이 책에서 사용되는 강조표시 규칙은 다음과 같습니다.

굵은체	명령, 키워드 및 시스템에서 이름이 사전 정의된 기타 항목을 나타냅니다.
이탤릭체	다음 중 하나를 나타냅니다. • 사용자가 제공해야 하는 이름 또는 값(변수) • 일반 강조 • 새 용어 소개 • 다른 정보 소스에 대한 참조
모노스페이스	다음 중 하나를 나타냅니다. • 파일 및 디렉토리 • 명령 프롬프트 또는 창에 입력하도록 지시되는 정보 • 특정 데이터 값의 예 • 시스템에 표시될 텍스트와 유사한 텍스트의 예 • 시스템 메시지의 예 • 프로그래밍 코드 샘플

제 1 부 계획 및 설계 고려사항

제 1 장 파티션된 데이터베이스 및 테이블

파티션된 데이터베이스 환경 설정

데이터베이스를 작성하기 전에 다중 파티션 데이터베이스를 작성 여부를 결정해야 합니다. 데이터베이스 설계 결정의 일부로서, 데이터베이스 파티션에서 제공할 수 있는 성능을 향상시킬 것인지 여부를 결정해야 합니다.

파티션된 데이터베이스 환경에서도 CREATE DATABASE 명령 또는 sqlecrea() 함수를 사용하여 데이터베이스를 작성합니다. 사용되는 메소드가 어느 것이든 db2nodes.cfg 파일에 나열된 임의의 파티션을 통해 요청할 수 있습니다. db2nodes.cfg 파일은 데이터베이스 파티션 서버 구성 파일입니다. (이전에는 노드 구성 파일로 알려져 있었습니다.)

Windows 운영 체제 환경을 제외하고 임의의 편집기를 사용하여 데이터베이스 파티션 서버 구성 파일(db2nodes.cfg)의 컨텐츠를 보고 갱신할 수 있습니다. Windows 운영 체제 환경에서 db2ncrt 및 db2nchg 명령을 사용하여 데이터베이스 파티션 서버 구성 파일을 작성 및 변경하십시오.

다중 파티션 데이터베이스를 작성하기 전에, 데이터베이스의 카탈로그 파티션이 될 데이터베이스 파티션을 선택해야 합니다. 그런 다음 해당 데이터베이스 파티션에서 직접 또는 그 데이터베이스 파티션에 접속된 리모트 클라이언트에서 데이터베이스를 작성할 수 있습니다. 접속하여 CREATE DATABASE 명령을 실행하는 데이터베이스 파티션이 해당 특정 데이터베이스에 대한 *카탈로그 파티션*이 됩니다.

카탈로그 파티션은 모든 시스템 카탈로그 테이블이 저장되는 데이터베이스 파티션입니다. 시스템 테이블로의 모든 액세스는 이 데이터베이스 파티션을 거쳐야 합니다. 모든 페더레이티드 데이터베이스 오브젝트(예: 랩퍼, 서버 및 별칭)는 이 데이터베이스 파티션에서 시스템 카탈로그 테이블에 저장됩니다.

가능하다면, 별도의 인스턴스에서 각 데이터베이스를 작성해야 합니다. 그렇지 않은 경우(즉, 인스턴스마다 둘 이상의 데이터베이스를 작성해야 할 경우), 사용 가능한 데이터베이스 파티션 간에 카탈로그 파티션을 확대해야 합니다. 이렇게 하면, 단일 데이터베이스 파티션에서 카탈로그 정보에 대한 경합이 줄어듭니다.

주: 다른 데이터로 인해 백업에 필요한 시간이 증가하므로 카탈로그 파티션은 정기적으로 백업해야 하며 사용자 데이터를 카탈로그 파티션에 저장하지 않아야(가능할 때마다) 합니다.

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 DB2가 설치된 디렉토리 아래의 sqllib 디렉토리에 있습니다. UNIX에서 작업할 때, 이 디렉토리는 파티션된 데이터베이스 환경을 구성하는 모든 데이터베이스 파티션에 대한 시스템 데이터베이스 디렉토리가 하나만 있으므로 공유 파일 시스템 (예: UNIX 플랫폼의 NFS)에 있어야 합니다. Windows에서 작업할 때, 시스템 데이터베이스 디렉토리는 인스턴스 디렉토리에 있습니다.

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 데이터베이스 파티션에서 공유됩니다.

데이터베이스 파티셔닝의 이점을 취하려면 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 데이터베이스 환경에 영향을 미치는 데이터베이스 관리 프로그램 구성 매개변수에는 **conn_elapse**, **fcm_num_buffers**, **fcm_num_channels**, **max_connretries**, **max_coordagents**, **max_time_diff**, **num_poolagents** 및 **stop_start_time**이 포함됩니다.

다중 데이터베이스 파티션 전반에 데이터베이스 파티셔닝

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

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

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

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

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

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

행은 다음과 같이 데이터베이스 파티션에 배치됩니다.

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

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

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

공동 배치된 테이블은 다음과 같아야 합니다.

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

파티션된 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.

데이터베이스 파티션 그룹

데이터베이스 파티션 그룹은 데이터베이스에 속하도록 정의된 하나 이상의 데이터베이스 파티션 세트입니다. 데이터베이스에 테이블을 작성하려면, 테이블 스페이스가 저장될 데이터베이스 파티션 그룹을 먼저 작성한 다음 테이블이 저장될 테이블 스페이스를 작성하십시오.

데이터베이스에서 하나 이상의 데이터베이스 파티션을 이름이 지정된 서브세트로 정의할 수 있습니다. 정의된 각 서브세트를 *데이터베이스 파티션 그룹*이라고 합니다. 두 개 이상의 데이터베이스 파티션이 들어 있는 서브세트는 *다중 파티션 데이터베이스 파티션 그룹*이라고 합니다. 다중 파티션 데이터베이스 파티션 그룹은 같은 인스턴스에 속한 데이터베이스 파티션으로만 정의될 수 있습니다. 데이터베이스 파티션 그룹은 최소 하나의 데이터베이스 파티션을 포함하거나 데이터베이스의 모든 데이터베이스 파티션을 포함할 수 있습니다.

7 페이지의 그림 1에서 다음 다섯 개의 데이터베이스 파티션이 있는 데이터베이스의 예를 보여줍니다.

- 데이터베이스 파티션 그룹은 데이터베이스 파티션 중 하나를 제외한 모두를 포함합니다(데이터베이스 파티션 그룹 1).
- 데이터베이스 파티션 그룹에는 하나의 데이터베이스 파티션만 있습니다(데이터베이스 파티션 그룹 2).
- 데이터베이스 파티션 그룹에는 두 개의 데이터베이스 파티션이 있습니다 (데이터베이스 파티션 그룹 3).
- 데이터베이스 파티션 그룹 2에 있는 데이터베이스 파티션은 데이터베이스 파티션 그룹 1과 공유(및 겹침)합니다.

- 데이터베이스 파티션 그룹 3에 있는 하나의 데이터베이스 파티션은 데이터베이스 파티션 그룹 1과 공유(및 겹침)합니다.

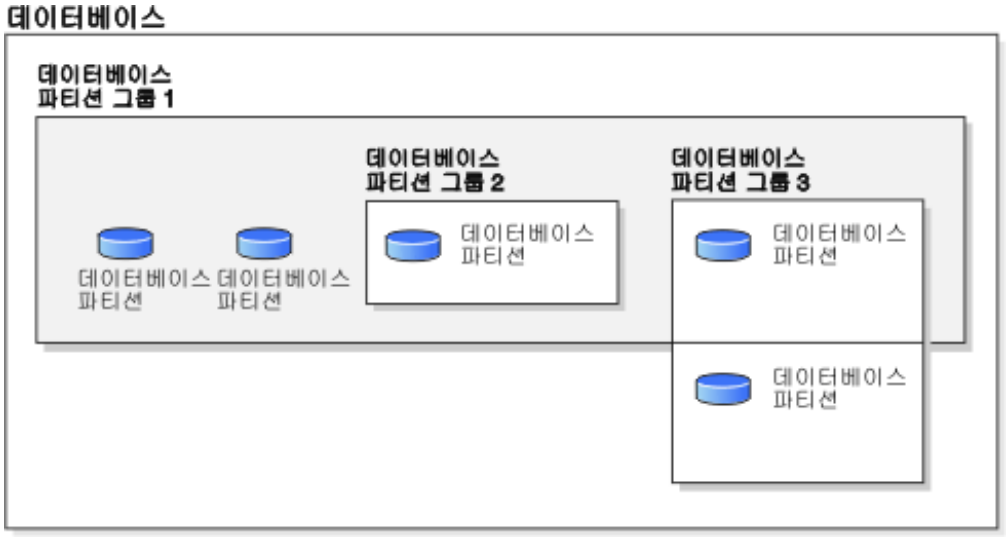


그림 1. 데이터베이스 내에 있는 데이터베이스 파티션 그룹

CREATE DATABASE PARTITION GROUP문을 사용하여 새 데이터베이스 파티션 그룹을 작성하십시오. 이 그룹은 ALTER DATABASE PARTITION GROUP문을 사용하여 수정할 수 있습니다. 데이터는 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션으로 분배되고 데이터베이스 파티션 그룹에 하나 이상의 데이터베이스 파티션을 추가하거나 삭제할 수 있습니다. 다중 파티션 데이터베이스 파티션 그룹을 사용하려면, 몇 가지 데이터베이스 파티션 그룹 설계 고려사항을 알아보아야 합니다.

데이터베이스 시스템 구성의 일부인 각 데이터베이스 파티션은 `db2nodes.cfg`라고 하는 *데이터베이스 파티션 구성 파일*에 이미 정의되어 있어야 합니다. 데이터베이스 파티션 그룹은 최소 하나의 데이터베이스 파티션부터 데이터베이스 시스템에 정의된 전체 데이터베이스 파티션까지 포함할 수 있습니다.

데이터베이스 파티션 그룹이 작성되거나 수정될 때 *분산 맵*이 데이터베이스 파티션 그룹과 연관됩니다. 데이터베이스 관리 프로그램은 *분산 키* 및 해싱 알고리즘과 함께 분산 맵을 사용하여 데이터베이스 파티션 그룹의 어떤 데이터베이스 파티션에 지정 데이터 행을 저장할 것인지 결정합니다.

파티션되지 않은 데이터베이스에서는 분산 키 또는 분산 캡이 필요하지 않습니다. 데이터베이스 파티션은 데이터베이스의 일부로서 사용자 데이터, 인덱스, 구성 파일, 트랜잭션 로그로 구성됩니다. 데이터베이스 관리 프로그램은 데이터베이스가 작성될 때 작성된 디폴트 데이터베이스 파티션 그룹을 사용합니다. IBMCATGROUP은 시스템 카탈로그가 들어 있는 테이블 스페이스에 대한 디폴트 데이터베이스 파티션 그룹입니다. IBMTEMPGROUP은 시스템 임시 테이블 스페이스에 대한 디폴트 데이터베이스 파티

션 그룹입니다. IBMDEFAULTGROUP은 사용자가 선택한 사용자 정의 테이블이 들어 있는 테이블 스페이스에 대한 디폴트 데이터베이스 파티션 그룹입니다. 선언된 임시 테이블 또는 작성된 임시 테이블 스페이스는 IBMDEFAULTGROUP 또는 사용자가 작성한 데이터베이스 파티션 그룹에 작성할 수 있지만, IBMTEMPGROUP에는 작성할 수 없습니다.

데이터베이스 파티션 그룹에서 다음과 같이 작업할 수 있습니다.

- 데이터베이스 파티션 그룹을 작성합니다.
- 데이터베이스 파티션 그룹과 연관된 주석을 변경합니다.
- 데이터베이스 파티션 그룹에 데이터베이스 파티션을 추가합니다.
- 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제합니다.
- 데이터베이스 파티션 그룹에서 테이블 데이터를 재분배합니다.

데이터베이스 파티션 그룹 디자인

단일 파티션 데이터베이스를 사용할 경우 데이터베이스 파티션 그룹을 설계할 때 고려할 사항이 없습니다. DB2 디자인 어드바이저는 데이터베이스 파티션 그룹을 권장할 때 사용할 수 있는 도구입니다. DB2 디자인 어드바이저는 제어 센터에서 액세스하고 명령행 처리기에서 db2advis를 사용하여 액세스할 수 있습니다.

다중 파티션 데이터베이스 파티션 그룹을 사용하는 경우, 설계 시 다음과 같은 사항을 고려하십시오.

- 다중 파티션 데이터베이스 파티션 그룹에서는 이 그룹이 분산 키의 수퍼 세트일 경우에만 고유 인덱스를 작성할 수 있습니다.
- 데이터베이스의 데이터베이스 파티션 수에 따라, 단일 파티션 데이터베이스 파티션 그룹과 다중 파티션 데이터베이스 파티션 그룹이 여러 개 있을 수 있습니다.
- 각 데이터베이스 파티션에는 고유한 번호가 지정되어야 합니다. 하나 이상의 데이터베이스 파티션 그룹에 같은 데이터베이스 파티션이 있을 수 있습니다.
- 시스템 카탈로그 테이블이 들어 있는 데이터베이스 파티션을 빠르게 복구하려면 동일한 데이터베이스 파티션에 사용자 테이블을 배치하지 마십시오. 이를 위해서는 IBMCATGROUP 데이터베이스 파티션 그룹의 데이터베이스 파티션을 포함하지 않는 데이터베이스 파티션 그룹에 사용자 테이블을 배치하면 됩니다.

더 큰 테이블과의 공동 *배치(collocation)*를 이용하려는 경우가 아니라면, 작은 테이블을 단일 파티션 데이터베이스 파티션 그룹에 배치해야 합니다. 공동 배치란 같은 데이터베이스 파티션에 있는 관련 데이터가 포함된 다른 테이블의 행을 배치하는 것입니다. 공동 배치된 테이블을 사용하면 Linux, UNIX 및 Windows용 DB2 데이터베이스에서 조인 전략을 보다 효율적으로 사용할 수 있습니다. 공동 배치된 테이블은 단일 파티션 데이터베이스 파티션 그룹에 상주할 수 있습니다. 테이블이 다중 파티션 데이터베이스 파티션 그룹에 있거나 분산 키의 컬럼 수와 같거나 해당 컬럼의 데이터 유형이 호환 가

능한 경우 공동 배치된 것으로 간주됩니다. 공동 배치된 테이블에서 동일한 분산 키 값을 갖는 행은 동일한 데이터베이스 파티션에 배치됩니다. 테이블은 동일한 데이터베이스 파티션 그룹에서 별도의 테이블 스페이스에 있을 수도 있는데 이 경우에도 공동 배치된 것으로 간주됩니다.

중간 크기의 테이블을 너무 많은 데이터베이스 파티션으로 확장시키면 안됩니다. 예를 들어, 100MB의 테이블은 32개의 파티션으로 구성된 데이터베이스 파티션 그룹보다 16개의 파티션으로 구성된 데이터베이스 파티션 그룹에서 더 잘 수행됩니다.

데이터베이스 파티션 그룹을 사용하여 온라인 트랜잭션 처리(OLTP) 테이블을 결정 지원(DSS) 테이블에서 분리하면 OLTP 트랜잭션의 성능이 역으로는 영향을 받지 않게 됩니다.

분산 맵

파티션된 데이터베이스 환경에서 데이터베이스 관리 프로그램은 필요한 데이터의 위치를 알아야 합니다. 데이터베이스 관리 프로그램은 *분산 맵*이라는 맵을 사용하여 데이터를 찾습니다.

분산 맵은 다중 파티션 데이터베이스 파티션 그룹의 경우 32 768개의 항목을 포함하고, 단일 파티션 데이터베이스 파티션 그룹의 경우 단일 항목을 포함하는, 내부적으로 생성된 배열입니다. 단일 파티션 데이터베이스 파티션 그룹의 경우, 분산 맵은 데이터베이스 테이블의 모든 행이 저장된 데이터베이스 파티션의 파티션 번호가 들어 있는 하나의 항목만을 보유합니다. 다중 파티션 데이터베이스 파티션 그룹의 경우, 각 데이터베이스 파티션이 전체 맵에 골고루 분산되도록 순서대로 사용하는 방식으로 데이터베이스 파티션 그룹의 번호가 지정됩니다. 도시의 지도가 격자를 사용하여 각 섹션으로 구분되듯이, 데이터베이스 관리 프로그램은 *분산 키*를 사용하여 데이터 저장 위치(데이터베이스 파티션)를 판별합니다.

예를 들어, 4개의 데이터베이스 파티션(번호 0–3)에 작성된 데이터베이스가 있다고 가정합니다. 이 데이터베이스의 IBMDEFAULTGROUP 데이터베이스 파티션 그룹에 대한 분산 맵은 다음과 같습니다.

```
0 1 2 3 0 1 2 ...
```

데이터베이스 파티션 그룹이 데이터베이스 파티션 1과 2를 사용하여 데이터베이스에 작성되면, 해당 데이터베이스 파티션 그룹에 대한 분산 맵은 다음과 같습니다.

```
1 2 1 2 1 2 1 ...
```

데이터베이스에 로드될 테이블의 분산 키가 1에서 500 000 사이의 정수일 경우, 분산 키는 0에서 32 767 사이의 번호로 해시됩니다. 번호는 해당 행의 데이터베이스 파티션을 선택하도록 분산 맵에 인덱스로 사용됩니다 .

그림 2은 분산 키 값(c1, c2, c3)을 갖는 행이 번호 2에 맵핑되는 방법입니다. 이 때, 파티션 2는 데이터베이스 n5를 참조합니다.

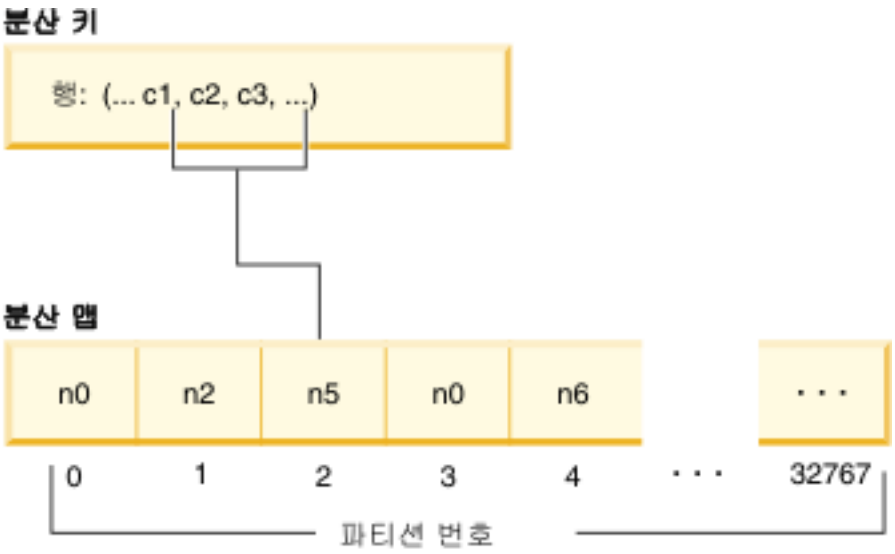


그림 2. 분산 맵을 사용한 데이터 분산

분산 맵을 사용하면 다중 파티션 데이터베이스에 데이터 저장 위치를 융통성 있게 제어할 수 있습니다. 데이터베이스의 데이터베이스 파티션에서 데이터 분산을 변경해야 할 경우, 데이터 재분배 유틸리티를 사용할 수 있습니다. 이 유틸리티를 사용하면 데이터 분산을 재조정할 수 있습니다.

db2GetDistMap API를 사용하여 볼 수 있는 분산 맵의 사본을 얻을 수 있습니다. 분산 정보를 얻기 위해 sqlugtpi API를 계속 사용하는 경우, 이 API는 4096개의 항목을 포함하는 분산 맵만 검색할 수 있으므로 오류 메시지 SQL2768N을 리턴할 수 있습니다.

분산 키

*분산 키*는 특정 데이터 행이 저장된 데이터베이스 파티션을 판별하는 데 사용되는 컬럼(또는 컬럼 그룹)입니다. 분산 키는 CREATE TABLE문을 사용하여 테이블에 정의됩니다. 데이터베이스 파티션 그룹에서 둘 이상의 데이터베이스 파티션에 분포된 테이블스페이스의 테이블에 대한 분산 키가 정의되어 있지 않을 경우, 디폴트로 기본 키의 첫 번째 컬럼에서 분산 키가 작성됩니다.

기본 키가 지정되어 있지 않을 경우, 테이블에서 정의된 long이 아닌 첫 번째 필드 컬럼이 디폴트 분산 키가 됩니다. (*Long*에는 모든 long 데이터 유형 및 모든 대형 오브젝트(LOB) 데이터 유형이 포함됩니다.) 단일 파티션 데이터베이스 파티션 그룹과 연관된 테이블 스페이스에 테이블을 작성하고 분산 키를 갖고자 할 경우, 분산 키를 명시적으로 정의해야 합니다. 디폴트로 파티션 키는 작성되지 않습니다.

디폴트 분산 키에 대한 요구사항을 충족하는 컬럼이 없는 경우, 분산 키 없이 테이블이 작성됩니다. 분산 키가 없는 테이블은 단일 파티션 데이터베이스 파티션 그룹에만 허용됩니다. ALTER TABLE문을 사용하여 분산 키를 나중에 추가 또는 삭제할 수 있습니다. 분산 키 변경은 단일 파티션 데이터베이스 파티션 그룹과 연관된 테이블 스페이스의 테이블에 대해 수행될 수 있습니다.

올바른 분산 키를 선택하는 것이 중요하므로 다음 사항을 고려해야 합니다.

- 테이블에 액세스하는 방법
- 쿼리 워크로드의 특성
- 데이터베이스 시스템에서 사용하는 조인 전략

공동 배치가 주요 고려사항이 아닐 경우, 테이블에 적합한 분산 키를 선택하는 것이 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션에 데이터를 균등하게 분배하는 방법입니다. 데이터베이스 파티션 그룹과 연관된 테이블 스페이스에 있는 각 테이블의 분산 키가 테이블의 공동 배치 여부를 결정합니다. 다음과 같은 경우 테이블은 공동 배치된 것으로 간주됩니다.

- 동일한 데이터베이스 파티션 그룹에 있는 테이블 스페이스에 테이블이 배치될 경우
- 각 테이블의 분산 키에 동일한 수의 컬럼이 있는 경우
- 해당 컬럼의 데이터 유형이 파티션과 호환되는 경우

이런 특징 때문에 같은 분산 키 값이 있는 공동 배치된 테이블의 행은 같은 데이터베이스 파티션에 있습니다.

분산 키가 적합하지 않을 경우 데이터 분산이 한쪽으로 편중될 수 있습니다. 데이터가 편중되게 분산된 컬럼과 구별 값이 별로 없는 컬럼을 분산 키로 선택할 수 없습니다. 이런 구별 값 수는 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션에 행이 균등하게 분배될만큼 많아야 합니다. 분산 키의 *크기*에 비례해 분산 알고리즘을 적용하는데 필요한 노력이 증가합니다. 분산 키는 17컬럼 이상일 수 없으며, 컬럼 수가 적을수록 성능이 향상됩니다. 분산 키에 불필요한 컬럼이 있어서는 안됩니다.

분산 키를 정의할 때 다음 사항을 고려하십시오.

- BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC, XML 또는 구조화된 데이터 유형만 있는 다중 파티션 테이블을 작성할 수 없습니다.
- 분산 키 정의를 변경할 수 없습니다.
- 분산 키는 가장 자주 조인되는 컬럼을 포함해야 합니다.
- 분산 키는 GROUP BY절에 자주 참여하는 컬럼으로 구성되어야 합니다.
- 모든 고유 키 또는 기본 키는 분산 키 컬럼을 모두 포함해야 합니다.
- 온라인 트랜잭션 처리(OLTP) 환경에서 분산 키에 있는 모든 컬럼은 상수 또는 호스트 변수와 함께 등호(=) 술어를 사용하여 트랜잭션에 참여해야 합니다. 예를 들어, 다음과 같이 트랜잭션에 자주 사용되는 사원 번호 *emp_no*가 있다고 가정합니다.

```
UPDATE emp_table SET ... WHERE
emp_no = host-variable
```

이 경우 EMP_NO 컬럼은 EMP_TABLE에 대한 유용한 단일 컬럼 분산 키를 작성합니다.

*데이터베이스 파티션*은 테이블에서 각 행의 배치가 결정되는 방법입니다. 이 방법은 다음과 같이 작동됩니다.

1. 해싱 알고리즘은 분산 키 값에 적용되고 영(0)과 32 767 사이의 번호를 생성합니다.
2. 분산 맵은 데이터베이스 파티션 그룹이 작성될 때 작성됩니다. 각 번호는 분산 맵을 채우기 위해 라운드 로빈 방식으로 순서대로 반복됩니다.
3. 번호는 분산 맵에 대한 인덱스로 사용됩니다. 분산 맵의 해당 위치에 있는 숫자는 행이 저장된 데이터베이스 파티션의 번호입니다.

테이블 공동 배치

둘 이상의 테이블이 특정 쿼리에 대한 응답으로 데이터를 자주 제공하는 경우가 있습니다. 이 경우, 관련 데이터를 이들 테이블에서 가능하면 서로 가까운 곳에 배치할 수 있습니다. 데이터베이스가 물리적으로 둘 이상의 데이터베이스 파티션으로 나누어진 환경에서는 파티션된 테이블의 관련된 부분을 가능한 한 가까이 둘 수 있는 방법이 있어야 합니다. 바로 이러한 기능을 *테이블 공동 배치*라고 합니다.

테이블이 같은 데이터베이스 파티션 그룹에 저장되는 경우 및 분산 키가 서로 호환되는 경우 테이블이 공동 배치됩니다. 두 테이블을 동일한 데이터베이스 파티션 그룹에 배치할 경우 공통 분산 맵을 사용하게 됩니다. 테이블은 다른 테이블 스페이스에 있을 수 있지만 테이블 스페이스는 같은 데이터베이스 파티션 그룹과 연관되어야 합니다. 각 분산 키의 해당 컬럼 데이터 유형은 *파티션 호환 가능* 유형이어야 합니다.

Linux, UNIX 및 Windows용 DB2 데이터베이스 소프트웨어는 조인 또는 서브쿼리를 위해 둘 이상의 테이블에 액세스할 때, 조인될 데이터가 동일한 데이터베이스 파티션에 위치하는지 인식할 수 있습니다. 이 경우, DB2는 데이터베이스 파티션 간에 데이터를 이동시키지 않고 데이터가 저장된 데이터베이스 파티션에서 조인 또는 서브쿼리를 수행할 수 있습니다. 이 기능으로 성능이 현저하게 향상됩니다.

파티션 호환성

분산 키에 있는 해당 컬럼의 기본 데이터 유형을 비교한 다음, *파티션 호환 가능*으로 선언할 수 있습니다. 파티션 호환 가능 데이터 유형은 동일한 값을 갖는 각 유형의 두 변수가 동일한 해싱 알고리즘에 의해 동일한 번호에 맵핑된다는 특성을 갖습니다.

파티션 호환성의 특성은 다음과 같습니다.

- 기본 데이터 유형은 동일한 기본 데이터 유형의 다른 유형과 호환됩니다.

- DATE, TIME 및 TIMESTAMP 데이터 유형에는 내부 형식이 사용됩니다. 이들은 서로 호환되지 않으며, 이들 중 어느 것도 문자 또는 그래픽 데이터 유형과 호환되지 않습니다.
- 널(null) 값 허용은 파티션 호환성에 영향을 주지 않습니다.
- 조합은 파티션 호환성에 영향을 줍니다. 로케일 구분 UCA 기반 조합은 조합의 강도(S) 속성이 무시되는 경우를 제외하고 조합에서 완전 일치를 필요로 합니다. 기타 모든 조합은 파티션 호환성 판별 목적으로 같다고 간주됩니다.
- FOR BIT DATA를 사용하여 정의된 문자 컬럼은 로케일 구분 UCA 기반 조합 이외의 조합이 사용되는 경우 FOR BIT DATA가 없는 문자 컬럼과만 호환 가능합니다.
- 호환 가능한 데이터 유형의 널(NULL) 값은 동일하게 취급되지만, 호환 가능하지 않은 데이터 유형의 널(NULL) 값은 동일하게 취급되지 않을 수 있습니다.
- 사용자 정의 유형의 기본 데이터 유형은 파티션 호환성을 분석하는데 사용됩니다.
- 분산 키의 동일한 값의 소수 부분은 소수점 이하 자릿수와 전체 자릿수가 다르더라도 동일하게 취급됩니다.
- 문자열(CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC) 뒤의 공백은 해시 알고리즘에서 무시됩니다.
- BIGINT, SMALLINT 및 INTEGER는 호환 가능한 데이터 유형입니다.
- 로케일 구분 UCA 기반 조합이 사용되는 경우, CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC는 호환 가능한 데이터 유형입니다. 기타 조합이 사용되는 경우, 다른 길이의 CHAR 및 VARCHAR은 호환 가능한 유형이고 GRAPHIC 및 VARGRAPHIC는 호환 가능한 유형이지만 CHAR 및 VARCHAR은 GRAPHIC 및 VARGRAPHIC와 호환 가능한 유형이 아닙니다.
- 파티션 호환성은 LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB 및 BLOB 데이터 유형에는 적용되지 않는데, 이런 데이터 유형은 분산 키로 지원되지 않기 때문입니다.

파티션된 테이블

파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 복수의 스토리지 오브젝트, 호출된 *데이터 파티션* 또는 *범위*에 나뉘어져 있는 데이터 조직 스킴을 사용합니다.

데이터 파티션 또는 범위는 테이블 행 서브세트를 포함하고 있고 기타 행 세트와 별도로 저장되는 테이블의 파트입니다. 제공된 테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 제공된 스펙을 기본으로 다중 데이터 파티션 또는 범위로 파티션됩니다. 이러한 데이터 파티션 또는 범위는 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘의 조합에 있을 수 있습니다. PARTITION BY절을 사용하여 테이블이 작성된 경우 테이블이 파티션됩니다.

지정되는 모든 테이블 스페이스가 동일한 페이지 *크기*, Extent *크기* 스토리지 메커니즘(DMS, SMS) 및 유형(REGULAR 또는 LARGE)을 가져야 하며 모든 테이블 스페이스가 동일한 *데이터베이스 파티션 그룹*에 있어야 합니다.

파티션된 테이블은 테이블 데이터의 롤인(rolling in) 및 롤 아웃(rolling out)을 단순화하며 파티션된 테이블은 일반 테이블에 비해 월등히 많은 데이터를 포함할 수 있습니다. 최대 32767개의 데이터 파티션이 있는 파티션된 테이블을 작성할 수 있습니다. 파티션된 테이블에 데이터 파티션을 추가, 접속 또는 접속 해제할 수 있으며 한 테이블에 있는 복수의 데이터 파티션 범위를 한 테이블 스페이스에 저장할 수 있습니다.

파티션된 테이블의 인덱스는 파티션되거나 파티션되지 않을 수 있습니다. 파티션되지 않거나 파티션된 인덱스는 모두 단일 파티션된 테이블에 함께 존재할 수 있습니다.

제한사항: 파티션된 계층 또는 임시 테이블, 범위 클러스터 테이블 및 파티션된 뷰는 지원되지 않습니다.

테이블 파티션

테이블 파티션은 테이블 데이터가 하나 이상의 테이블 컬럼 값에 따라 *범위* 또는 *데이터 파티션*이라고 하는 복수의 스토리지 오브젝트에 나누어져 있는 데이터 조직 스킴입니다. 각 데이터 파티션은 따로 저장됩니다. 이러한 스토리지 오브젝트는 서로 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘의 조합에 있을 수 있습니다.

스토리지 오브젝트는 개별 테이블처럼 동작하므로 ALTER TABLE ...ATTACH 명령문을 사용하여 기존의 테이블을 파티션된 테이블에 통합함으로써 보다 쉽고 빠르게 롤인(roll-in)을 수행할 수 있습니다. 마찬가지로, ALTER TABLE ...DETACH 명령문을 사용하여 롤아웃(roll-out)도 쉽게 수행할 수 있습니다. 또한, 쿼리 처리는 관련이 없는 데이터를 스캔하지 않기 위해 데이터 분리를 사용하는데, 이것은 많은 데이터 웨어하우스 스타일 쿼리의 쿼리 성능을 향상시킵니다.

테이블 데이터는 CREATE TABLE문의 PARTITION BY절에 지정된 대로 파티션됩니다. 이 정의에 사용된 컬럼을 테이블 파티션 키 컬럼이라고 합니다.

이 조직 스킴은 단독으로 또는 다른 조직 스킴과 조합하여 사용될 수 있습니다. CREATE TABLE문의 DISTRIBUTE BY 및 PARTITION BY절을 조합하여 복수의 테이블 스페이스에 걸쳐 있는 데이터베이스 파티션으로 데이터를 분산시킬 수 있습니다. 조직 스킴에는 다음이 포함됩니다.

- DISTRIBUTE BY HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

테이블 파티션 기능은 Linux, UNIX 및 Windows용 DB2 Enterprise Server Edition 버전 9.1 이상에서 사용할 수 있습니다.

테이블 파티션의 장점

다음 환경 중 하나가 사용자 또는 사용자의 조직에 적용될 경우 테이블 파티션의 여러 가지 장점을 고려하십시오.

- 데이터 웨어하우스에서 테이블 데이터의 롤인 및 롤아웃을 쉽게 이용할 수 있음
- 데이터 웨어하우스에 *크기*가 큰 테이블이 포함되어 있음
- 이전 릴리스나 경쟁업체 데이터베이스 제품에서 버전 9.1 데이터베이스로 이주를 고려할 수 있음
- HSM(Hierarchical Storage Management) 솔루션을 보다 효율적으로 사용해야 함

테이블 파티션은 테이블 데이터를 쉽게 롤인 및 롤아웃하고 쉽게 관리할 수 있으며, 인덱스를 유연하게 배치하고 쿼리 처리 성능을 향상시킬 수 있는 기능을 제공합니다.

효율적인 롤인 및 롤아웃

테이블 파티션을 사용하면 테이블 데이터를 효율적으로 롤인 및 롤아웃할 수 있습니다. ALTER TABLE문의 ATTACH PARTITION 및 DETACH PARTITION절을 사용하여 이를 수행할 수 있습니다. 파티션된 테이블 데이터를 롤인하면 새 행을 쉽게 파티션된 테이블에 추가 데이터 파티션으로서 통합할 수 있습니다. 파티션된 테이블 데이터를 롤아웃하면 후속 제거 또는 아카이브시 파티션된 테이블로부터 데이터 범위를 쉽게 분리할 수 있습니다.

보다 용이한 대형 테이블 관리

개별 데이터 파티션에 대해 관리 태스크를 수행할 수 있기 때문에 테이블 레벨 관리가 보다 유연합니다. 이 태스크에는 데이터 파티션 연결 해제 및 재접속, 개별 데이터 파티션 백업 및 복구 그리고 개별 인덱스 재구성이 포함됩니다. 조작을 일련의 보다 작은 조작으로 나누어 시간을 많이 소모하는 유지보수 조작 시간을 단축할 수 있습니다. 예를 들어, 데이터 파티션이 별도의 테이블 스페이스에 있을 경우, 백업 조작은 데이터 파티션별로 데이터 파티션에 대해 작업할 수 있습니다. 따라서 파티션된 테이블의 데이터 파티션을 한 번에 하나씩 백업할 수 있습니다.

유연한 인덱스 배치

인덱스를 다른 테이블 스페이스에 배치하여 인덱스의 배치를 보다 세분하게 제어할 수 있습니다. 이 새 디자인의 장점은 다음과 같습니다.

- 인덱스 삭제 및 온라인 인덱스 작성 시 성능이 향상됩니다.
- 테이블의 각 인덱스 사이에서 모든 테이블 스페이스 특성에 대해 다른 값을 사용할 수 있습니다(예: 각 인덱스에 대한 다른 페이지 *크기*는 더 나은 스페이스 활용을 보장함).
- 제한된 입출력 경합은 테이블의 인덱스 데이터에 더 효율적인 동시 액세스를 제공합니다.

- 개별 인덱스가 삭제되면 익덱스를 재구성할 필요없이 시스템이 즉시 스페이스를 사용 할 수 있습니다.
- 인덱스 재구성을 수행하도록 선택한 경우 개별 인덱스가 재구성됩니다.

DMS 및 SMS 테이블 스페이스 둘 다 테이블과 다른 위치에 인덱스를 사용할 수 있습니다.

비즈니스 인텔리전스 스타일 쿼리의 성능 개선

쿼리의 술어를 기본으로 자동으로 데이터 파티션을 제거할 수 있도록 쿼리 처리가 향상되었습니다. *데이터 파티션 제거*라고 하는 이 기능은 여러 의사 결정 지원 쿼리에 도움이 됩니다.

다음 예에서는 *customer* 테이블을 작성합니다. 이때, l_shipdate >= '01/01/2006' 및 l_shipdate <= '03/31/2006'인 행은 테이블 스페이스 ts1에 저장되고, l_shipdate >= '04/01/2006' 및 l_shipdate <= '06/30/2006'인 행은 스페이스 ts2에 저장됩니다.

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
IN ts1, ts2, ts3, ts4, ts5
PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

데이터 파티션 및 범위

파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 복수의 스토리지 오브젝트, 호출된 *데이터 파티션* 또는 *범위*에 나뉘어져 있는 데이터 조직 스킴을 사용합니다. 테이블을 작성할 때 각 데이터 파티션에 지정된 범위를 자동 또는 수동으로 생성할 수 있습니다.

DB2 라이브러리에서 여러 가지 방법으로 데이터 파티션을 참조할 수 있습니다. 다음 목록은 가장 일반적인 참조를 나타냅니다.

- DATAPARTITIONNAME은 작성 시 제공된 테이블의 데이터 파티션에 지정된 영구 이름입니다. 이 컬럼 값은 SYSCAT.DATAPARTITIONS 카탈로그 뷰에 저장됩니다. 이 이름은 접속 또는 접속 해제 조작에서는 보존되지 않습니다.
- DATAPARTITIONID은 작성 시 제공된 테이블의 데이터 파티션에 지정된 영구 ID입니다. 이는 제공된 테이블의 특정 데이터 파티션을 고유하게 식별하는데 사용됩니다. 이 ID는 접속 또는 접속 해제 조작에서는 보존되지 않습니다. 이 값은 시스템에서 생성되며 여러 유틸리티에서 생성된 출력에 나타날 수 있습니다.
- SEQNO는 테이블의 다른 데이터 파티션 범위에 대한 특정 데이터 파티션 범위의 순서를 나타냅니다. 모든 시각적 및 접속된 데이터 파티션 다음에 접속 해제된 데이터 파티션이 정렬됩니다.

데이터 조직 스킴

테이블 파티셔닝의 시작과 함께 DB 데이터베이스는 세 개의 레벨 데이터 조직 스킴을 제안합니다. CREATE TABLE문에는 데이터 구성 방법을 표시하는 알고리즘을 포함하는 세 개의 절이 있습니다.

다음 세 개의 절은 어떤 조합으로도 함께 사용할 수 있는 데이터 소속 레벨을 보여줍니다.

- DISTRIBUTE BY로 데이터를 데이터베이스 파티셔닝에 고르게 퍼뜨립니다(쿼리간 병렬 처리를 위해 그리고 각 데이터베이스 파티션 전반에 밸런스를 맞추기 위해)(데이터베이스 파티셔닝).
- PARTITION BY로 단일 차원의 비슷한 값을 가진 행을 동일한 데이터 파티션에 분류합니다(테이블 파티셔닝).
- ORGANIZE BY로 다중 차원에서 비슷한 값을 가진 행을 동일한 테이터 테이블 Extent에 분류합니다(다차원 클러스터링).

이 구문으로 예상 데이터 소속 알고리즘을 사용할 수 있을 뿐만 아니라 절 간의 일관성을 유지할 수 있습니다. 각 구문을 단독으로 또는 다른 절과 조합하여 사용할 수 있습니다. CREATE TABLE문의 DISTRIBUTE BY 및 PARTITION BY절을 조합하여 여러 테이블 스페이스에 걸쳐 있는 데이터베이스 파티션으로 데이터를 분산시킬 수 있습니다. 이 방법은 Informix® Dynamic Server 및 Informix Extended Parallel Server 하이브리드 기능과 유사한 동작을 허용합니다.

단일 테이블에서 각 데이터 조직 스킴에 사용되는 절을 조합하여 더 복잡한 파티셔닝 스킴을 작성할 수 있습니다. 예를 들어, DB2 Database Partitioning Feature(DPF)은 호환 가능할 뿐만 아니라 테이블 파티셔닝도 보완합니다.

그림 3. 테이블이 월별 판매 데이터를 표시하는 테이블에서 파티셔닝 조직 스킴이 다중 데이터 파티션으로 파티션되는 방법을 보여줍니다. 테이블은 두 개의 테이블 스페이스(ts1 및 ts2)로 되어 있습니다.

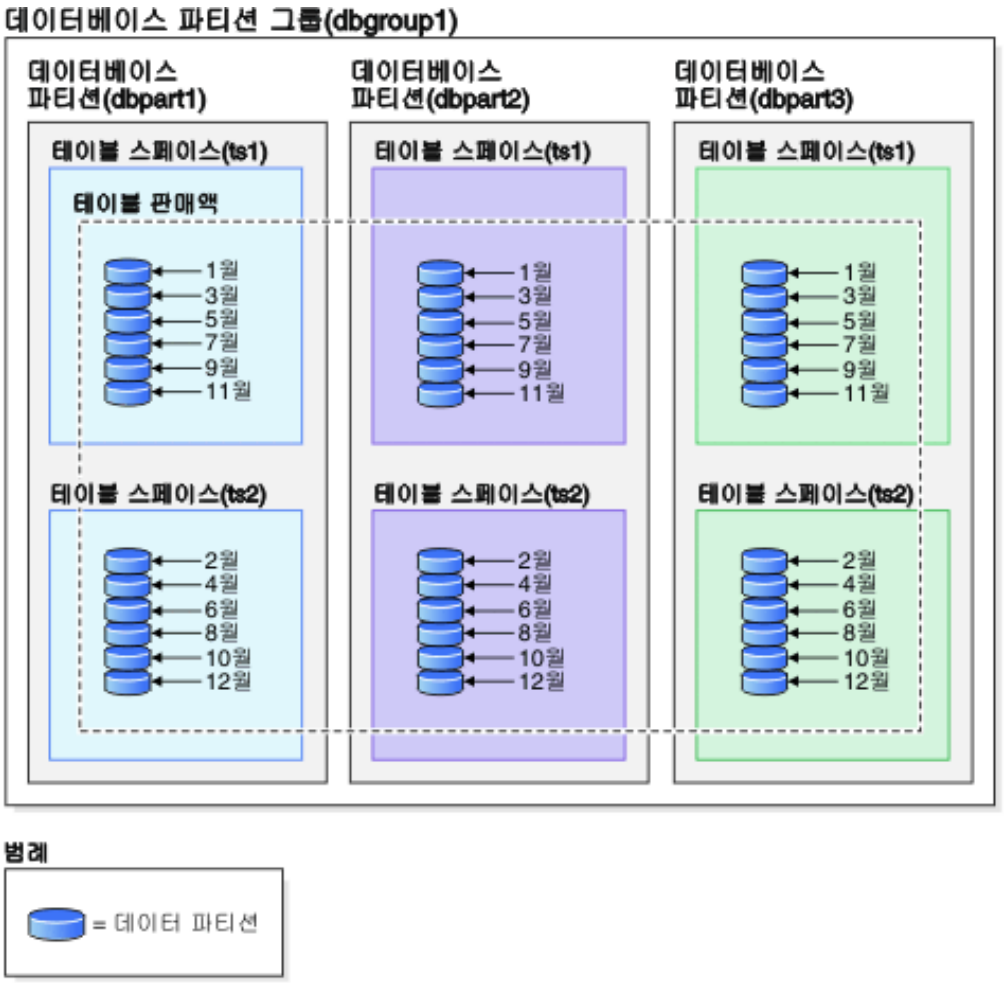


그림 4. 데이터베이스 파티셔닝 및 테이블 파티셔닝의 보완 조직 스킴을 설명합니다. 월별 판매 데이터를 표시하는 테이블은 두 테이블 스페이스(ts1 및 ts2)에 걸쳐 있는 여러 데이터 파티션으로 파티션되어 있습니다. 이때 두 테이블 스페이스는 데이터베이스 파티션 그룹(dbgroup1)의 다중 데이터베이스 파티션(dbpart1, dbpart2, dbpart3)에 분산되어 있습니다.

다차원적으로 클러스터된(MDC)과 테이블 파티셔닝의 두드러진 차이는 다중 차원과 단일 차원이라는 점입니다. MDC는 큐브(다중 차원의 테이블)에 적합한 반면, 테이블 파티셔닝은 DATE 컬럼과 같이 데이터베이스 디자인에서 중심인 단일 차원이 있는 경우에 알맞습니다. MDC와 테이블 파티셔닝은 이와 같은 조건이 모두 충족될 때 상호 보완합니다. 이것은 20 페이지의 그림 5에 설명되어 있습니다.

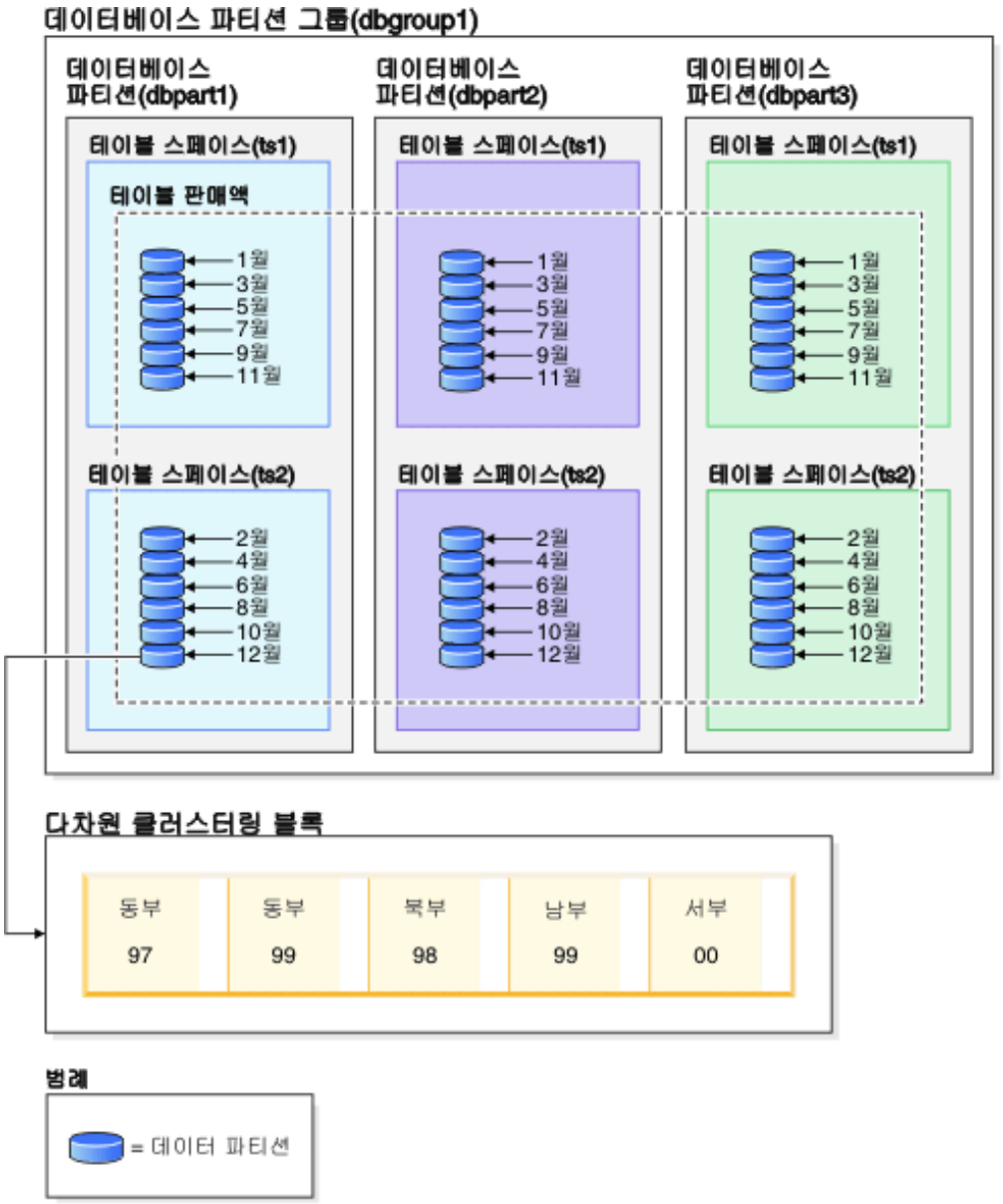


그림 5. 판매 테이블의 데이터가 ts1 및 ts2 테이블 스페이스에 파티션되어 있고 여러 데이터베이스 파티션에 분산되어 있을 뿐만 아니라 날짜와 지역 차원에 대한 값이 비슷한 행끼리 그룹화된 데이터베이스 파티셔닝, 테이블 파티셔닝 및 다차원 조직 스킴에 대한 표시입니다.

위에서 열거한 어떤 사항과도 연결하여 사용할 수 없는 다른 데이터 조직 스킴이 있습니다. 이 스킴은 ORGANIZE BY KEY SEQUENCE로, 테이블이 작성(RCT)되었을 때 해당 기록이 예약된 행 안에 각 기록을 삽입할 때 사용됩니다.

데이터 소속 용어

데이터베이스 파티셔닝

테이블의 하나 이상의 분산 키 컬럼에 있는 해시 값과 데이터베이스 파티션의 분산 맵 사용을 기준으로 테이블 데이터가 여러 데이터베이스 파티션에 걸쳐 나

눠진 데이터 조직 스킴입니다. 특정 테이블의 데이터는 CREATE TABLE문의 DISTRIBUTE BY HASH절에서 제공되는 권장 스펙에 기반하여 분배됩니다.

데이터베이스 파티션

자체 사용자 데이터, 인덱스, 구성 파일 및 트랜잭션 로그로 구성된 데이터베이스 파티션 서버의 부분입니다. 데이터베이스 파티션은 논리적이거나 물리적일 수 있습니다.

테이블 파티션

하나 이상의 테이블 컬럼에 있는 값에 따라 테이블 데이터가 여러 데이터 파티션에 분배되어 있는 데이터 조직 스킴입니다. 제공된 테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 제공된 스펙을 기본으로 복수의 스토리지 오브젝트로 파티션됩니다. 이러한 스토리지 오브젝트는 다른 테이블 스페이스에 있을 수 있습니다.

데이터 파티션

테이블 행의 집합으로, 다른 행의 집합으로부터 따로 저장되어 있고, CREATE TABLE문의 PARTITION BY RANGE절에서 제공한 권장 스펙으로 그룹화되어 있습니다.

다차원 클러스터링(MDC)

데이터가 하나 이상의 차원에서 실제로 블록으로 구성되는 테이블 또는 ORGANIZE BY DIMENSIONS로 지정된 클러스터링 키입니다.

각 데이터 조직 스킴의 장점

각 데이터 조직 스킴의 장점을 이해하면 데이터베이스 시스템 요구사항을 플랜, 설계 또는 재평가할 때 최선의 접근 방법을 판별할 수 있습니다. 표 2는 일반 고객 요구사항의 상위 레벨 뷰를 제공하고 이러한 요구사항을 다양한 데이터 조직 스킴이 충족시킬 수 있는 방법을 보여줍니다.

표 2. 데이터베이스 파티션 기능에서 테이블 파티셔닝 사용

문제점	권장 스킴	설명
데이터 롤아웃	테이블 파티션	접속 해제로 대형 데이터를 최소의 장애로 롤아웃합니다.
병렬 쿼리 실행(쿼리 성능)	데이터베이스 파티션 기능	쿼리 성능을 향상하기 위해 쿼리 병렬 처리를 제공합니다.
데이터 파티션 제거(쿼리 성능)	테이블 파티션	쿼리 성능을 향상하기 위해 데이터 파티션 제거를 제공합니다.
쿼리 성능 최대화	둘 다	쿼리 병렬 처리와 데이터 파티션 제거를 함께 사용하면 상호 보충하여 최대의 쿼리 성능을 가져옵니다.
과중한 관리자 워크로드	데이터베이스 파티션 기능	각 데이터베이스 파티션에 태스크를 많이 실행함

표 3. MDC 테이블에 테이블 파티션 사용

문제점	권장 스킴	설명
롤아웃 동안 데이터 사용 가능성	테이블 파티션	DETACH PARTITION절을 사용하여 최소한의 장애로 대형 데이터 롤아웃
쿼리 성능	둘 다	다중 차원 쿼리를 위해 MDC가 최선입니다. 테이블 파티션이 데이터 파티션을 제거하여 돕습니다.
최소 재구성	MDC	MDC가 클러스터링을 유지보수하여 재구성의 필요성을 줄입니다.

주: 테이블 파티션이 지금 UNION ALL 뷰에 권장됩니다.

DB2 및 Informix 데이터베이스의 데이터 조직 스킴

테이블 파티션은 테이블 데이터가 하나 이상의 테이블 컬럼 값에 따라 범위 또는 데이터 파티션이라고 하는 복수의 스토리지 오브젝트에 나누어져 있는 데이터 조직 스킴입니다. 각 데이터 파티션은 따로 저장됩니다. 이러한 스토리지 오브젝트는 서로 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘의 조합에 있을 수 있습니다.

테이블 데이터는 CREATE TABLE문의 PARTITION BY절에 지정된 대로 파티션됩니다. 이 정의에 사용된 컬럼을 테이블 파티션 키 컬럼이라고 합니다. DB2 테이블 파티셔닝은 Informix Dynamic Server 및 Informix Extended Parallel Server에서 제공하는 데이터 소속에 대한 데이터 단편화 접근 방법에 맵핑됩니다.

Informix 접근 방법

Informix는 Informix 제품에서 *분할화*라고 불리는 여러 개의 데이터 소속 스킴을 지원합니다. 단편화 중 더 일반적으로 사용되는 유형은 FRAGMENT BY EXPRESSION입니다. 이러한 단편화 유형은 테이블의 각 조각과 연결된 표현식이 있는 CASE 명령문과 매우 유사하게 작업합니다. 행의 위치를 결정하기 위해 이러한 표현식을 점검합니다.

Informix 및 DB2 데이터베이스 시스템 비교

DB2 데이터베이스는 Informix 데이터 조직 스킴에 직접 맵핑되는 다양한 보완 기능을 제공하여, 고객이 비교적 쉽게 Informix 구문을 DB2 구문으로 변환할 수 있도록 합니다. DB2 데이터베이스 관리 프로그램은 생성된 컬럼 및 CREATE TABLE문의 PARTITION BY RANGE절을 조합하여 복잡한 Informix 스킴을 처리합니다. 23 페이지의 표 4에 Informix 및 DB2 데이터베이스 제품에 사용된 데이터 조직 스킴이 비교되어 있습니다.

표 4. 모든 Informix 및 DB2 데이터 조직 스킴의 맵핑

데이터 조직 스킴	**Informix** 구문	**DB2** 버전 **9.1** 구문
• Informix: 표현식 기반 • DB2: 테이블 파티션	FRAGMENT BY EXPRESSION	PARTITION BY RANGE
• Informix: 라운드 로빈 • DB2: 디폴트값	FRAGMENT BY ROUND ROBIN	구문 없음: DB2 데이터베이스 관리 프로그램은 자동으로 데이터를 컨테이너로 분산시킴
• Informix: 범위 분산 • DB2: 테이블 파티션	FRAGMENT BY RANGE	PARTITION BY RANGE
• Informix: 시스템 정의 해시 • DB2: 데이터베이스 파티션	FRAGMENT BY HASH	DISTRIBUTE BY HASH
• Informix: HYBRID • DB2: 테이블 파티션이 있는 데이터베이스 파티션	FRAGMENT BY HYBRID	DISTRIBUTE BY HASH, PARTITION BY RANGE
• Informix: n/a • DB2: 다차원 클러스터링	n/a	ORGANIZE BY DIMENSION

예

다음 예에서는 표현식 스킴을 사용한 임의의 Informix 단편과 동등한 결과를 생성하도록 DB2 데이터베이스를 완성하는 방법에 대한 세부사항을 제공합니다.

예 1: 다음의 기본적인 테이블 작성 명령문에 Informix 단편화 및 이와 동등한 DB2 데이터베이스 시스템용 테이블 파티션 구문이 나와 있습니다.

Informix 구문:

```
CREATE TABLE demo(a INT) FRAGMENT BY EXPRESSION
 a = 1 IN db1,
 a = 2 IN db2,
 a = 3 IN db3;
```

DB2 구문:

```
CREATE TABLE demo(a INT) PARTITION BY RANGE(a)
 (STARTING(1) IN db1,
 STARTING(2) IN db2,
 STARTING(3) ENDING(3) IN db3);
```

Informix XPS는 데이터가 한 표현식에서는 여러 코서버에 걸쳐 분포되어 있고 두 번째 표현식에서는 코서버 내부에 분포되어 있는 하이브리드라고 알려진 2레벨 분할화 스킴을 지원합니다. 그러면 쿼리에서 데이터 파티션 제거 기능을 이용할 수 있을 뿐 아니라 쿼리에서 모든 공동 서버(co-server)를 사용할 수 있습니다(즉, 모든 공동 서버(co-server)에 데이터가 있음).

DB2 데이터베이스 시스템은 CREATE TABLE문의 DISTRIBUTE BY 및 PARTITION BY절의 조합을 사용하여 Informix 하이브리드(hybrid)와 동등한 조직 스킴을 얻을 수 있습니다.

예 2: 다음 예에 절이 결합된 구문이 나와 있습니다.

Informix 구문

```
CREATE TABLE demo(a INT, b INT) FRAGMENT BY HYBRID HASH(a)
 EXPRESSION b = 1 IN dbsl1,
   b = 2 IN dbsl2;
```

DB2 구문

```
CREATE TABLE demo(a INT, b INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1);
```

또한 다차원 클러스터링을 사용하여 추가 레벨의 데이터 조직도 확보할 수 있습니다.

```
CREATE TABLE demo(a INT, b INT, c INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1)
 ORGANIZE BY DIMENSIONS(c);
```

따라서 **a** 컬럼의 값이 동일한 모든 행은 동일한 데이터베이스 파티션에 있습니다. **b** 컬럼의 값이 동일한 모든 행은 동일한 테이블 스페이스에 있습니다. **a** 및 **b** 컬럼의 값이 제공된 경우, **c** 컬럼의 값이 동일한 모든 행은 디스크에 함께 클러스터됩니다. 이 유형의 쿼리를 만족시키려면 단일 데이터베이스 파티션의 단일 테이블 스페이스에 있는 단 하나 또는 여러 Extent(블록)를 스캔해야 하기 때문에 이 접근 방법은 OLAP 유형의 드릴 다운 조작에 이상적입니다.

공통 응용프로그램 문제점에 적용되는 테이블 파티션

다음 절에서는 DB2 테이블 파티션의 여러 가지 기능을 공통 응용프로그램 문제점에 적용하는 방법에 대해 설명합니다. 각 절에서 여러 Informix 단편화 스킴을 동등한 DB2 테이블 파티션 스킴에 맵핑하는 우수 사례를 특히 주의하여 살펴보십시오.

단순 데이터 파티션 범위에 대한 고려사항

테이블 파티션의 가장 일반적인 응용프로그램 중 하나는 날짜 키를 기본으로 대형 사실 테이블을 파티션하는 것입니다. 고유한 크기의 날짜 범위를 작성해야 하는 경우 자동으로 생성된 CREATE TABLE 구문 양식을 사용하는 것을 고려하십시오.

예

예 1: 다음 예는 자동으로 생성된 구문 양식을 보여줍니다.

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
```

```
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber  INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice  DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag  CHAR(1),
 l_linestatus  CHAR(1),
 l_shipdate  DATE,
 l_commitdate  DATE,
 l_receiptdate  DATE,
 l_shipinstruct  CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44))
  PARTITION BY RANGE(l_shipdate)
  (STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH);
```

이 구문에서는 1992년에서 1993년까지 매달 하나씩 24개의 범위를 작성합니다. l_shipdate를 사용하여 해당 범위를 벗어난 행을 삽입하면 오류가 발생합니다.

예 2: 다음 Informix 구문과 앞의 예를 비교해 보십시오.

```
create table orders
(
 l_orderkey  decimal(10,0) not null,
 l_partkey  integer,
 l_suppkey  integer,
 l_linenumber integer,
 l_quantity  decimal(12,2),
 l_extendedprice decimal(12,2),
 l_discount  decimal(12,2),
 l_tax   decimal(12,2),
 l_returnflag char(1),
 l_linestatus char(1),
 l_shipdate  date,
 l_commitdate date,
 l_receiptdate date,
 l_shipinstruct char(25),
 l_shipmode  char(10),
 l_comment  varchar(44)
) fragment by expression
l_shipdate < '1992-02-01' in ldbs1,
l_shipdate >= '1992-02-01' and l_shipdate < '1992-03-01' in ldbs2,
l_shipdate >= '1992-03-01' and l_shipdate < '1992-04-01' in ldbs3,
l_shipdate >= '1992-04-01' and l_shipdate < '1992-05-01' in ldbs4,
l_shipdate >= '1992-05-01' and l_shipdate < '1992-06-01' in ldbs5,
l_shipdate >= '1992-06-01' and l_shipdate < '1992-07-01' in ldbs6,
l_shipdate >= '1992-07-01' and l_shipdate < '1992-08-01' in ldbs7,
l_shipdate >= '1992-08-01' and l_shipdate < '1992-09-01' in ldbs8,
l_shipdate >= '1992-09-01' and l_shipdate < '1992-10-01' in ldbs9,
l_shipdate >= '1992-10-01' and l_shipdate < '1992-11-01' in ldbs10,
l_shipdate >= '1992-11-01' and l_shipdate < '1992-12-01' in ldbs11,
l_shipdate >= '1992-12-01' and l_shipdate < '1993-01-01' in ldbs12,
l_shipdate >= '1993-01-01' and l_shipdate < '1993-02-01' in ldbs13,
l_shipdate >= '1993-02-01' and l_shipdate < '1993-03-01' in ldbs14,
l_shipdate >= '1993-03-01' and l_shipdate < '1993-04-01' in ldbs15,
l_shipdate >= '1993-04-01' and l_shipdate < '1993-05-01' in ldbs16,
```

```
l_shipdate >= '1993-05-01' and l_shipdate < '1993-06-01' in ldbs17,
l_shipdate >= '1993-06-01' and l_shipdate < '1993-07-01' in ldbs18,
l_shipdate >= '1993-07-01' and l_shipdate < '1993-08-01' in ldbs19,
l_shipdate >= '1993-08-01' and l_shipdate < '1993-09-01' in ldbs20,
l_shipdate >= '1993-09-01' and l_shipdate < '1993-10-01' in ldbs21,
l_shipdate >= '1993-10-01' and l_shipdate < '1993-11-01' in ldbs22,
l_shipdate >= '1993-11-01' and l_shipdate < '1993-12-01' in ldbs23,
l_shipdate >= '1993-12-01' and l_shipdate < '1994-01-01' in ldbs24,
l_shipdate >= '1994-01-01' in ldbs25;
```

Informix 구문은 예상 범위를 벗어난 날짜를 포착하기 위해 맨 위와 맨 아래에 개방된 종료 범위를 제공합니다. MINVALUE 및 MAXVALUE를 사용하는 범위를 추가하여 Informix 구문과 일치하도록 DB2 구문을 수정할 수 있습니다.

예 3: 다음 예는 Informix 구문과 일치하도록 예 1을 수정한 것입니다.

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag CHAR(1),
 l_linestatus CHAR(1),
 l_shipdate  DATE,
 l_commitdate DATE,
 l_receiptdate DATE,
 l_shipinstruct CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44)
) PARTITION BY RANGE(l_shipdate)
 (STARTING MINVALUE,
  STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH,
  ENDING MAXVALUE);
```

이 기술을 사용하면 테이블에 임의의 날짜를 삽입할 수 있습니다.

생성된 컬럼을 사용한 표현식에 의한 파티션

DB2 데이터베이스가 직접 표현식을 사용한 파티션을 지원하지는 않지만 생성된 컬럼에서의 파티션은 지원하므로 동일한 결과를 얻을 수 있습니다.

이 방법을 사용할지 여부를 판별하기 전에 다음과 같은 사용 지침을 고려하십시오.

- 생성된 컬럼은 실제 디스크 스페이스를 사용하는 실제 컬럼입니다. 생성된 컬럼을 사용하는 테이블이 약간 클 수 있습니다.
- 파티션된 테이블을 파티션하는 컬럼의 생성된 컬럼 표현식을 변경하는 것은 지원되지 않습니다. 이를 변경하려 하면 SQL0190 메시지가 발행됩니다. 새 데이터 파티션을 다음 절에 설명된 방법으로 생성된 컬럼을 사용하는 테이블에 추가하려면 일반적으

로 생성된 컬럼을 정의하는 표현식을 변경해야 합니다. 생성된 컬럼을 정의하는 표현식을 변경하는 것은 현재 지원되지 않습니다.

- 테이블에서 생성된 컬럼을 사용할 때 데이터 파티션 제거를 적용할 수 있는 경우에 적용되는 제한사항이 있습니다.

예

예 1: 다음은 생성된 컬럼을 사용하는데 적절한 Informix 구문을 사용합니다. 이 예에서 파티션될 컬럼은 캐나다 주 및 지역 이름을 보유합니다. 주 목록은 변경 가능성이 없기 때문에 생성된 컬럼 표현식은 변경되지 않습니다.

```
CREATE TABLE customer (
 cust_id  INT,
 cust_prov CHAR(2))
 FRAGMENT BY EXPRESSION
  cust_prov = "AB" IN dbspace_ab
  cust_prov = "BC" IN dbspace_bc
  cust_prov = "MB" IN dbspace_mb
  ...
  cust_prov = "YT" IN dbspace_yt
  REMAINDER IN dbspace_remainder;
```

예 2: 이 예에서, DB2 테이블은 생성된 컬럼을 사용하여 파티션됩니다.

```
CREATE TABLE customer (
 cust_id     INT,
 cust_prov   CHAR(2),
 cust_prov_gen GENERATED ALWAYS AS (CASE
  WHEN cust_prov = 'AB' THEN 1
  WHEN cust_prov = 'BC' THEN 2
  WHEN cust_prov = 'MB' THEN 3
  ...
  WHEN cust_prov = 'YT' THEN 13
  ELSE 14 END))
 IN tbspace_ab, tbspace_bc, tbspace_mb, .... tbspace_remainder
 PARTITION BY RANGE (cust_prov_gen)
  (STARTING 1 ENDING 14 EVERY 1);
```

Case문의 표현식은 FRAGMENT BY EXPRESSION절의 해당 표현식과 일치합니다. Case문은 원래의 각 표현식을 생성된 컬럼(이 예에서는 cust_prov_gen)에 저장된 숫자에 맵핑합니다. 이 컬럼은 디스크에 저장된 실제 컬럼이므로 DB2가 표현식을 사용하여 직접 파티션을 지원할 때보다 약간의 스페이스를 더 사용할 수 있습니다. 이 예에서는 간단한 형식의 구문을 사용합니다. 따라서 데이터 파티션을 배치할 테이블 스페이스는 CREATE TABLE문의 IN절에 나열되어 있어야 합니다. Long 양식의 구문을 사용하려면 각 데이터 파티션에 대해 별도의 IN절이 필요합니다.

주: 이 기술은 모든 FRAGMENT BY EXPRESSION절에 적용될 수 있습니다.

테이블 파티셔닝 키

테이블 파티션 키는 테이블에 있는 하나 이상의 컬럼으로 구성된 순서화된 세트입니다. 테이블 파티션 키 컬럼의 값은 각 테이블 행이 속하는 데이터 파티션을 판별하는 데 사용됩니다.

테이블에 대한 테이블 파티션 키를 정의하려면 PARTITION BY절과 함께 CREATE TABLE문을 사용하십시오.

테이블 파티션의 장점을 최대한 이용하려면 효율적인 테이블 파티션 키 컬럼을 선택해야 합니다. 다음 가이드라인을 사용하여 파티션된 테이블에 대해 가장 효율적인 테이블 파티션 키 컬럼을 선택할 수 있습니다.

- 데이터 롤아웃과 일치하는 범위 단위 정의. 주, 월 또는 분기를 사용하는 것이 가장 일반적입니다.
- 데이터 롤인 크기와 일치하는 범위 정의. 날짜 또는 시간 컬럼에 따라 데이터를 파티션하는 것이 가장 일반적입니다.
- 파티션 제거 시 유리한 컬럼의 파티션

지원되는 데이터 유형

표 5는 테이블 파티션 키 컬럼으로 사용할 수 있도록 지원되는 데이터 유형(동의어 포함)을 표시합니다.

표 5. 지원되는 데이터 유형

데이터 유형 컬럼 1	데이터 유형 컬럼 2
SMALLINT	INTEGER
INT	BIGINT
FLOAT	REAL
DOUBLE	DECIMAL
DEC	DECFLOAT
NUMERIC	NUM
CHARACTER	CHAR
VARCHAR	DATE
TIME	GRAPHIC
VARGRAPHIC	CHARACTER VARYING
TIMESTAMP	CHAR VARYING
CHARACTER FOR BIT DATA	CHAR FOR BIT DATA
VARCHAR FOR BIT DATA	CHARACTER VARYING FOR BIT DATA
CHAR VARYING FOR BIT DATA	사용자 정의 유형(구별)

지원되지 않는 데이터 유형

파티션된 테이블에 다음 데이터 유형이 나타날 수는 있으나 테이블 파티션 키 컬럼으로는 사용할 수 없습니다.

- 사용자 정의 유형(구조화됨)
- LONG VARCHAR
- LONG VARCHAR FOR BIT DATA
- BLOB
- BINARY LARGE OBJECT
- CLOB
- CHARACTER LARGE OBJECT
- DBCLOB
- LONG VARGRAPHIC
- REF
- C의 경우에는 가변 길이 문자열
- Pascal의 경우에는 가변 길이 문자열
- XML

CREATE TABLE문의 EVERY절을 사용하여 자동으로 데이터 파티션을 생성하도록 선택한 경우 한 컬럼만이 테이블 파티션 키로 사용될 수 있습니다. CREATE TABLE 문의 PARTITION BY절을 사용하여 수동으로 데이터 파티션을 생성하도록 선택한 경우 다음 예에 표시된 것처럼 복수의 컬럼이 테이블 파티션 키로서 사용될 수 있습니다.

```
CREATE TABLE sales (year INT, month INT)
   PARTITION BY RANGE(year, month)
    (STARTING FROM (2001, 1) ENDING (2001,3) IN tbsp1,
     ENDING (2001,6) IN tbsp2, ENDING (2001,9)
     IN tbsp3, ENDING (2001,12) IN tbsp4,
     ENDING (2002,3) IN tbsp5, ENDING (2002,6)
     IN tbsp6, ENDING (2002,9) IN tbsp7,
     ENDING (2002,12) IN tbsp8)
```

이 예에서는 2001년과 2002년 사이의 각 분기마다 하나씩 8개의 데이터 파티션이 생성됩니다.

주:

1. 복수의 컬럼이 테이블 파티션 키로 사용될 경우, 뒤(trailing) 컬럼이 앞(leading) 컬럼에 종속된다는 점에서 이 컬럼은 복합 키(인덱스의 복합 키와 유사함)로 간주됩니다. 각 시작 또는 종료값(모든 컬럼)은 512자 이내로 지정되어야 합니다. 이 한계는 SYSCAT.DATAPARTITIONS 카탈로그 뷰의 LOWVALUE 및 HIGHVALUE 컬럼 크기에 해당합니다. 512자 이상을 사용하여 시작 또는 종료값을 지정하면 오류 SQL0636N, 이유 코드 9가 발생합니다.

2. 테이블 파티션은 다차원이 아닌 다중 컬럼입니다. 테이블 파티션에서, 사용된 모든 컬럼은 단일 차원의 일부입니다.

생성된 컬럼

생성된 컬럼은 테이블 파티션 키로 사용될 수 있습니다. 이 예에서는 매월마다 하나, 즉 12개의 데이터 파티션을 작성합니다. 모든 해의 1월에 해당하는 모든 행은 첫 번째 데이터 파티션에 배치되고 2월에 해당하는 행은 두 번째 데이터 파티션에 배치됩니다.

예 1

```
CREATE TABLE monthly_sales (sales_date date,
 sales_month int GENERATED ALWAYS AS (month(sales_date)))
     PARTITION BY RANGE (sales_month)
         (STARTING FROM 1 ENDING AT 12 EVERY 1);
```

주:

1. 테이블 파티션 키에 사용된 생성된 컬럼의 표현식을 변경하거나 삭제할 수 없습니다. 테이블 파티션 키에 사용된 컬럼의 생성된 컬럼 표현식을 추가하는 것은 허용되지 않습니다. 테이블 파티션 키에 사용된 컬럼의 생성된 컬럼 표현식을 추가, 삭제 또는 변경하면 오류가 발생합니다(SQL0270N rc=52).
2. 생성된 컬럼이 단순(monotonic)하지 않거나 옵티마이저가 컬럼이 단순(monotonic)함을 발견하지 못할 경우 데이터 파티션 제거가 범위 술어로 사용되지 않습니다. 단순(monotonic)하지 않은 표현식이 존재할 경우, 데이터 파티션 제거는 등식 또는 IN 술어에 대해서만 발생할 수 있습니다. 단순성에 대한 자세한 설명은 56 페이지의 『MDC 테이블을 작성 시 고려사항』의 내용을 참조하십시오.

파티션된 테이블에 대한 로드 고려사항

다음 일반 제한사항을 제외하고 목표 테이블이 파티션된 경우 모든 기존 로드 기능이 지원됩니다.

- 파티셔닝 에이전트 수가 1보다 큰 경우 일관성 지점은 지원되지 않습니다.
- 데이터 파티션의 서브세트로 데이터를 로드하면서 나머지 데이터 파티션은 완전히 온라인 상태로 남는 조건은 지원되지 않습니다.
- 로드 조작에서 사용하는 예외 테이블은 파티션할 수 없습니다.
- 목표 테이블에 XML 컬럼이 포함된 경우 예외 테이블을 지정할 수 없습니다.
- 로드 유틸리티를 삽입 모드 또는 재시작 모드로 실행하고 로드 목표 테이블에 접속 해제된 종속이 있는 경우 고유 인덱스는 재빌드할 수 없습니다.
- MDC 테이블 로드와 비슷하게 파티션된 테이블을 로드할 때 입력 데이터 레코드의 정확한 순서는 보존되지 않습니다. 순서는 셀 또는 데이터 파티션에서만 유지됩니다.
- 각 데이터베이스 파티션에서 다중 포맷터를 활용하는 로드 조작은 입력 레코드의 대략적인 순서만 보존합니다. 각 데이터베이스 파티션에서 단일 포맷터를 실행하면 셀

또는 테이블 파티셔닝 키로 입력 레코드를 그룹화합니다. 각 데이터베이스 파티션에서 단일 포맷터를 실행하려면 명시적으로 1의 CPU_PARALLELISM을 요청합니다.

일반 로드 동작

로드 유틸리티는 올바른 데이터 파티션에 데이터 레코드를 삽입합니다. 스플리터(splitter)와 같은 외부 유틸리티를 사용하여 로드 전에 입력 데이터를 파티션하는 경우 관련 요구사항은 없습니다.

로드 유틸리티는 접속 해제 또는 접속된 데이터 파티션에 액세스하지 않습니다. 데이터는 표시되는 데이터 파티션에만 삽입됩니다. 표시되는 데이터 파티션은 접속 또는 접속 해제된 상태가 아닙니다. 또한 로드 교체 조작은 접속 또는 접속 해제된 데이터 파티션을 절단하지 않습니다. 로드 유틸리티는 카탈로그 시스템 테이블에서 잠금을 획득하므로 로드 유틸리티는 커미트되지 않은 ALTER TABLE 트랜잭션을 기다립니다. 이러한 트랜잭션은 카탈로그 테이블의 관련 행에서 배타적 잠금을 획득합니다. 로드 조작을 계속 진행하려면 배타적 잠금을 종료해야 합니다. 즉, 로드 조작을 실행하는 동안 커미트되지 않은 ALTER TABLE ...ATTACH, DETACH 또는 ADD PARTITION 트랜잭션이 있을 수 있습니다. 접속 또는 접속 해제된 데이터 파티션이 목표로 지정된 입력 소스 레코드가 거부되고 하나가 지정되면 예외 테이블에서 검색될 수 있습니다. 목표 테이블 데이터 파티션의 일부가 접속 또는 접속 해제된 상태임을 나타내는 정보 메시지가 메시지 파일에 작성됩니다. 목표 테이블에 대응하는 관련 카탈로그 테이블 행을 잠그면 로드 유틸리티를 실행하는 동안 ALTER TABLE ...ATTACH, DETACH 또는 ADD PARTITION 조작을 실행하여 목표 테이블의 파티셔닝을 변경하지 못합니다.

유효하지 않은 행 처리

로드 유틸리티가 가시적 데이터 파티션에 속하지 않는 레코드를 발견하면 해당 레코드는 거부되고 로드 유틸리티는 처리를 계속합니다. 제한조건 위반 범위 때문에 거부된 레코드 수는 명시적으로 표시되지 않지만 거부된 레코드의 전체 수에 포함됩니다. 범위 위반으로 레코드가 거부된 경우 행 경고 수는 늘어나지 않습니다. 범위 위반을 발견했음을 나타내는 단일 메시지(SQL0327N)가 로드 유틸리티 메시지 파일에 작성되지만 레코드당 메시지는 로그되지 않습니다. 또한 목표 테이블의 모든 컬럼 이외에도 예외 테이블은 특별 행에서 나타나는 위반 유형을 설명하는 컬럼을 포함합니다. 파티션할 수 없는 데이터를 포함하여 유효하지 않은 데이터를 포함하는 행은 덤프 파일에 작성됩니다.

예외 테이블 삽입은 자원 소모가 많으므로 예외 테이블에 삽입할 제한조건 위반을 제어할 수 있습니다. 예를 들어 로드 유틸리티의 디폴트 동작은 범위 제한조건 또는 고유 제한조건 위반으로 거부된 행을 삽입하는 것입니다. 그렇지 않고 유효한 경우 예외 테이블로 삽입됩니다. 각각 FOR EXCEPTION절에서 NORANGEEXC 또는 NOUNIQUEEXC를 지정하여 이 동작을 끌 수 있습니다. 이러한 제한조건 위반을 예외 테이블에 삽입하지 않도록 하거나 예외 테이블을 지정하지 않는 경우 범위 제한조건 또는 고유 제한조건을 위반하는 행에 대한 정보는 유실됩니다.

실행기록 파일

목표 테이블이 파티션된 경우 해당하는 실행기록 파일 항목은 목표 테이블에 포함된 테이블 스페이스 목록을 포함하지 않습니다. 다른 조작 세분 단위 ID('T' 대신 'R')는 파티션된 테이블에서 로드 조작을 실행함을 나타냅니다.

로드 조작 종료

로드 교체를 완전히 종료하면 모든 가시적 데이터 파티션이 절단되고 로드 삽입을 종료하면 모든 가시적 데이터 파티션을 로드 전의 길이로 절단합니다. 인덱스는 로드 복사 단계에서 실패한 ALLOW READ ACCESS 로드 조작 종료 중 유효성이 확인되지 않습니다. 또한 인덱스를 처리하는 ALLOW NO ACCESS 로드 조작을 종료하는 경우에도 인덱스 유효성이 확인되지 않습니다. 인덱스 모드가 재빌드이거나 인덱스가 불일치 상태로 남게 되는 증분 유지보수 중에 키가 삽입되었기 때문입니다. 다중 목표로 데이터를 로드하는 경우 로드 복구 조작에 영향을 주지 않습니다. 단, 로드 단계 중 수행된 일관성 지점부터 로드 조작을 재시작하는 기능은 예외입니다. 이 경우 목표 테이블이 파티션되었으면 SAVECOUNT 로드 옵션이 무시됩니다. 이 동작은 MDC 목표 테이블로 데이터를 로드할 때와 일관됩니다.

생성된 컬럼

생성된 컬럼이 파티셔닝, 차원 또는 분산 키에 있는 경우 `generatedoverride` 파일 유형 수정자는 무시되고 `generatedignore` 파일 유형 수정자가 지정된 경우와 같이 로드 유틸리티에서 값을 생성합니다. 이 경우 올바르지 않은 생성된 컬럼 값을 로드하면 잘못된 물리적 위치(예: 잘못된 데이터 파티션, MDC 블록 또는 데이터베이스 파티션)에 레코드를 배치할 수 있습니다. 예를 들어 레코드가 잘못된 데이터 파티션에 배치되면 세트 무결성은 이 레코드를 다른 물리적 위치로 이동해야 합니다. 온라인 세트 무결성 조작 중 수행할 수 없습니다.

데이터 사용 가능성

현재 ALLOW READ ACCESS 로드 알고리즘은 파티션된 테이블로 확장됩니다. ALLOW READ ACCESS 로드 조작에서는 동시 판독기에서 로딩 및 비로딩 데이터 파티션 모두를 포함하여 전체 테이블에 액세스할 수 있습니다.

데이터 파티션 상태

로드에 성공한 후 가시적 데이터 파티션은 특정 조건에서 무결성 설정 보류 또는 읽기 액세스 전용 테이블 상태로 변경될 수 있습니다. 로드 조작을 유지보수할 수 없는 테이블에 대한 제한조건이 있으면 데이터 파티션은 이 상태로 배치될 수 있습니다. 이러한 제한조건으로는 점검 제한조건 및 접속 해제된 구체화된 쿼리 테이블을 포함할 수 있습니다. 실패한 로드 조작은 보이는 모든 데이터 파티션을 로드 보류 테이블 상태로 설정합니다.

오류 분리

데이터 파티션 레벨의 오류 분리는 지원되지 않습니다. 오류 분리는 오류가 발생하지 않는 데이터 파티션에서 로드를 계속하고 오류로 발생한 데이터 파티션을 중지하는 작업

을 의미합니다. 오류는 다른 데이터베이스 파티션 사이에서 분리될 수 있지만 로드 유틸리티는 가시적 데이터 파티션의 서브세트에서 트랜잭션을 커미트하고 나머지 가시적 데이터 파티션을 롤백할 수 없습니다.

기타 고려사항

- 인덱스가 유효하지 않은 항목으로 표시되면 증분 인덱싱은 지원되지 않습니다. 재빌드해야 하거나 접속 해제된 종속 항목에서 SET INTEGRITY문으로 유효성을 확인해야 하는 경우 인덱스는 유효하지 않은 항목으로 간주됩니다.
- 범위로 파티션되거나 해시로 분산되거나 차원으로 구성되는 알고리즘 조합을 사용하여 파티션된 테이블로 로드하는 작업이 지원됩니다.
- 로드로 영향을 받는 오브젝트 및 테이블 스페이스 ID의 목록을 포함하는 로그 레코드의 경우 이러한 레코드(LOAD START 및 COMMIT(PENDING LIST))의 크기는 상당히 커질 수 있으므로 다른 응용프로그램에서 사용 가능한 사용 중인 로그 스페이스 크기가 줄어듭니다.
- 테이블이 파티션 및 분산된 경우 파티션된 데이터베이스 로드가 모든 데이터베이스 파티션에 영향을 주지 않을 수도 있습니다. 출력 데이터베이스 파티션의 오브젝트만 변경됩니다.
- 로드 조작 중 파티션된 테이블의 메모리 소모는 테이블 수만큼 증가합니다. 단, 전체 메모리 요구사항의 작은 비율만 데이터 파티션 수에 비례하므로 전체 증가분이 선형으로 나타나지는 않습니다.

복제된 구체화된 쿼리 테이블

*설명*은 테이블의 데이터를 판별하는 데에도 사용되는 쿼리에 의해 정의된 테이블입니다. 설명을 사용하면 쿼리 성능을 높일 수 있습니다. 데이터베이스 관리 프로그램에서 쿼리의 분할 영역을 구체화된 쿼리 테이블을 사용하여 해결할 수 있다고 판별하는 경우 구체화된 쿼리 테이블을 사용하기 위해 쿼리를 다시 쓸 수 있습니다.

파티션된 데이터베이스 환경에서 구체화된 쿼리 테이블을 복제하고 그것을 사용하여 쿼리 성능을 향상시킬 수 있습니다. *복제된 구체화된 쿼리 테이블*은 단일 파티션 데이터베이스 파티션 그룹에 작성된 테이블을 기반으로 하지만, 다른 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션에 복제할 테이블을 기반으로 할 수도 있습니다. 복제된 구체화된 쿼리 테이블을 작성하려면, CREATE TABLE문을 REPLICATED 키워드와 함께 호출하십시오.

복제된 구체화된 쿼리 테이블을 사용하면 일반적으로 공동 배치되지 않은 테이블을 공동 배치할 수 있습니다. 대형 사실 테이블과 소형 차원 테이블이 있는 조인의 경우, 복제된 구체화된 쿼리 테이블은 특히 유용합니다. 모든 Replica를 갱신할 때의 영향뿐만 아니라 필요한 추가 스토리지를 최소화하기 위해, 복제될 테이블은 작고 자주 갱신되어야 합니다.

주: 드물게 갱신되는 대형 테이블을 복제하는 경우도 고려해야 하는데, 공동 배치(collocation)를 통해 얻을 수 있는 성능상의 이점으로 한 번 복제할 때의 부담이 상쇄됩니다.

복제된 테이블을 정의하는데 사용되는 subselect절에 적절한 술어를 지정하면, 선택한 컬럼이나 선택한 행 중 한 가지 또는 둘 다를 복제할 수 있습니다.

데이터베이스 파티션 그룹에서 테이블 스페이스

다중 파티션 데이터베이스 파티션 그룹에 테이블 스페이스를 배치하면, 테이블 스페이스 내의 모든 테이블이 데이터베이스 파티션 그룹의 각 데이터베이스 파티션으로 분할되거나 파티션됩니다.

테이블 스페이스는 하나의 데이터베이스 파티션 그룹에 작성됩니다. 테이블 스페이스가 임의 데이터베이스 파티션 그룹에 작성된 후에는 계속 남아 있어야 합니다. 다른 데이터베이스 파티션 그룹으로 변경될 수 없습니다. CREATE TABLESPACE문이 테이블 스페이스와 데이터베이스 파티션 그룹을 연관시키는데 사용됩니다.

테이블 파티션 및 다중 클러스터링 테이블

한 테이블이 다차원 클러스터링 및 파티션될 수 있습니다. 다차원 클러스터링되면서 또한 파티션되는 테이블에서 컬럼을 테이블 파티션 범위 파티션 스펙 및 MDC 키에서 사용할 수 있습니다. 이는 어느 하나의 함수만 사용하여 달성할 수 있는 것보다 더 정교한 데이터 파티션 및 블록 제거를 달성하는데 유용합니다. 또한 테이블이 파티션되는 곳과는 다른 컬럼을 MDC 키에 대해 지정하는 것이 유용한 많은 응용프로그램이 있습니다. MDC는 다차원인데 반해 테이블 파티션은 다중 컬럼임을 주의해야 합니다. 파티션된 테이블의 인덱스는 파티션되거나 파티션되지 않을 수 있습니다. MDC 블록 인덱스는 항상 파티션되지 않습니다.

메인스트림 DB2 데이터 웨어하우스의 특성

다음 권장사항은 DB2 V9.1에서 새로운 기능인 전형적인 메인스트림 웨어하우스에 초점을 두었습니다. 다음 특성이 가정됩니다.

- 데이터베이스가 다중 머신 또는 다중 AIX 논리적 파티션에서 실행됩니다.
- 데이터베이스 파티션 기능(DPF)이 사용됩니다(테이블이 DISTRIBUTE BY HASH 절을 사용하여 작성됩니다).
- 4 - 50개의 데이터 파티션이 있습니다.
- MDC 및 테이블 파티션이 고려되고 있는 테이블이 주요 사실 테이블입니다.
- 테이블이 1억 - 1000억 개의 행을 갖습니다.
- 새 데이터가 다양한 시간 프레임(야간, 주별, 월별)에 로드됩니다.
- 일별 처리 볼륨은 1만 - 1000만 레코드입니다.

- 데이터 볼륨은 변합니다. 가장 큰 월은 가장 작은 월 크기의 5X입니다. 마찬가지로 가장 큰 차원(제품 행, 지역)은 크기 범위의 5X입니다.
- 1 - 5년의 자세한 데이터가 보유됩니다.
- 만기된 데이터는 월별 또는 분기별로 롤아웃됩니다.
- 테이블은 광범위한 쿼리 유형을 사용합니다. 그러나 워크로드는 대부분 OLTP 워크로드에 상대적으로 다음 특성을 갖는 분석 쿼리입니다.
  - 최고 2백만 행을 갖는 더 큰 결과 세트
  - 대부분 또는 모든 쿼리는 기본 테이블이 아니라 적중하는 보기
- 범위(BETWEEN절), 목록의 항목 등으로 데이터를 선택하는 SQL절

메인스트림 DB2 V9.1 데이터 웨어하우스 사실 테이블의 특성

일반적인 웨어하우스 사실 테이블은 다음 설계를 사용할 수 있습니다.

- 월 컬럼에 데이터 파티션을 작성하십시오.
- 롤 아웃하려는 각 기간(예: 한 달, 3개월)에 대한 데이터 파티션을 정의하십시오.
- 일에 대한 MDC 차원을 작성하고 1 - 4개의 추가 차원을 작성하십시오. 일반적인 차원은 제품 행과 지역입니다.
- 모든 데이터 파티션 및 MDC 클러스터가 모든 데이터베이스 파티션에 분산됩니다.

MDC 및 테이블 파티션은 중첩되는 일련의 이점을 제공합니다. 다음 표는 조직에서 잠재적인 요구를 나열하고 이전에 식별된 특성을 기초로 권장되는 조직 스킴을 식별합니다.

표 6. MDC 테이블에 테이블 파티션 사용

문제점	권장 스킴	권장사항
롤아웃 동안 데이터 사용 가능성	테이블 파티션	DETACH PARTITION절을 사용하여 최소한의 장애로 대형 데이터 롤아웃
쿼리 성능	테이블 파티션 및 MDC	다중 차원 쿼리를 위해 MDC가 최선입니다. 테이블 파티션이 데이터 파티션을 제거하여 돕습니다.
최소 재구성	MDC	MDC가 클러스터링을 유지보수하여 재구성의 필요성을 줄입니다.
일반적인 오프라인 창 동안 1개월 이상의 데이터를 롤아웃합니다.	테이블 파티션	데이터 파티션이 이 필요성을 전적으로 다룹니다. MDC는 아무 것도 추가하지 않으며 그 자체로는 적합하지 않습니다.
마이크로 오프라인 창(1분 미만) 동안 1개월 이상의 데이터를 롤아웃합니다.	테이블 파티션	데이터 파티션이 이 필요성을 전적으로 다룹니다. MDC는 아무 것도 추가하지 않으며 그 자체로는 적합하지 않습니다.

표 6. MDC 테이블에 테이블 파티션 사용 (계속)

문제점	권장 스킴	권장사항
어떤 서비스도 중지하지 않으면서 쿼리를 제출하는 비즈니스 사용자에 대해 테이블을 완전히 사용 가능하게 유지하면서 1개월 이상의 데이터를 롤아웃합니다.	MDC	MDC만 이 필요성을 다소 다룹니다. 테이블 파티션은 테이블이 오프라인이 되는 짧은 기간으로 인해 적합하지 않을 수 있습니다.
매일 데이터 로드(ALLOW READ ACCESS 또는 ALLOW NO ACCESS)	테이블 파티션 및 MDC	MDC가 이 이점의 대부분을 제공합니다. 테이블 파티션은 증분 이점을 제공합니다.
"연속" 데이터 로드(ALLOW READ ACCESS)	테이블 파티션 및 MDC	MDC가 이 이점의 대부분을 제공합니다. 테이블 파티션은 증분 이점을 제공합니다.
"전통적인 BI" 쿼리에 대한 쿼리 실행 성능	테이블 파티션 및 MDC	MDC가 큐브/다차원 쿼리에 특히 좋습니다. 테이블 파티션은 파티션 제거를 통해 도움이 됩니다.
재구성 필요성을 피하거나 태스크 수행과 연관된 어려움을 줄여서 재구성 어려움을 최소화합니다.	MDC	MDC는 재구성할 필요성을 줄이는 클러스터링을 유지보수합니다. MDC가 사용되는 경우 데이터 파티션은 증분 이점을 제공하지 않습니다. 그러나 DC가 사용되지 않는 경우 테이블 파티션이 일부 과정 그레인 클러스터링을 파티션 레벨에서 유지보수하여 재구성 필요성을 줄이는데 도움이 됩니다.

예 1:

키 컬럼 YearAndMonth 및 Province를 갖는 테이블을 고려하십시오. 이 테이블을 플랜하는 합리적인 접근법은 데이터 파티션당 2개월의 날짜를 파티션하는 것일 수 있습니다. 또한 Province별로 구성하여, 임의의 날짜 범위 2개월 내에서 특정 지방에 대한 모든 행이 37 페이지의 그림 6에 표시된 것처럼 함께 클러스터되도록 할 수 있습니다.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

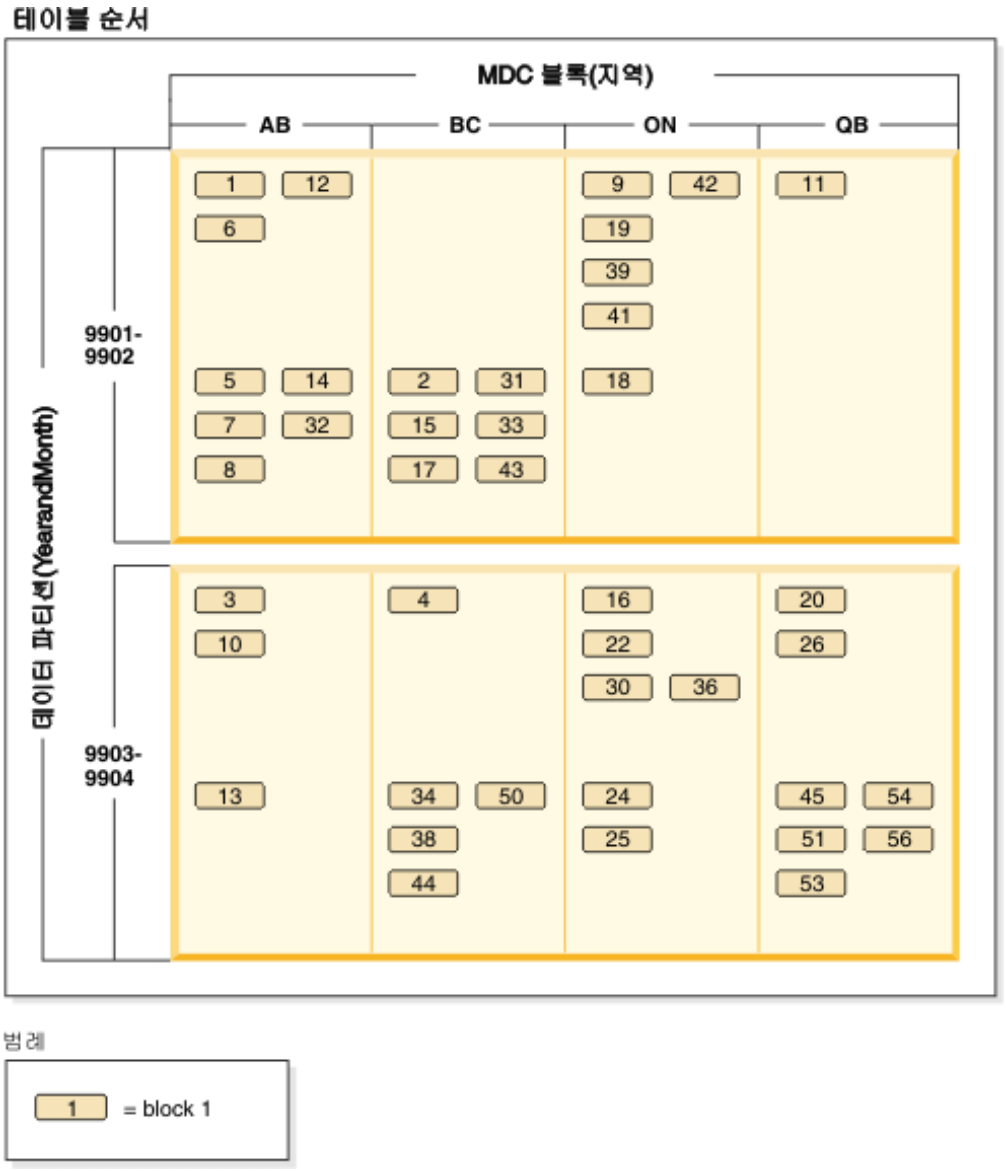


그림 6. YearAndMonth로 파티션되고 Province로 구성되는 테이블

예 2

38 페이지의 그림 7에서 보는 것처럼 ORGANIZE BY절에 YearAndMonth를 추가하여 더 정교한 세분화도를 달성할 수 있습니다.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

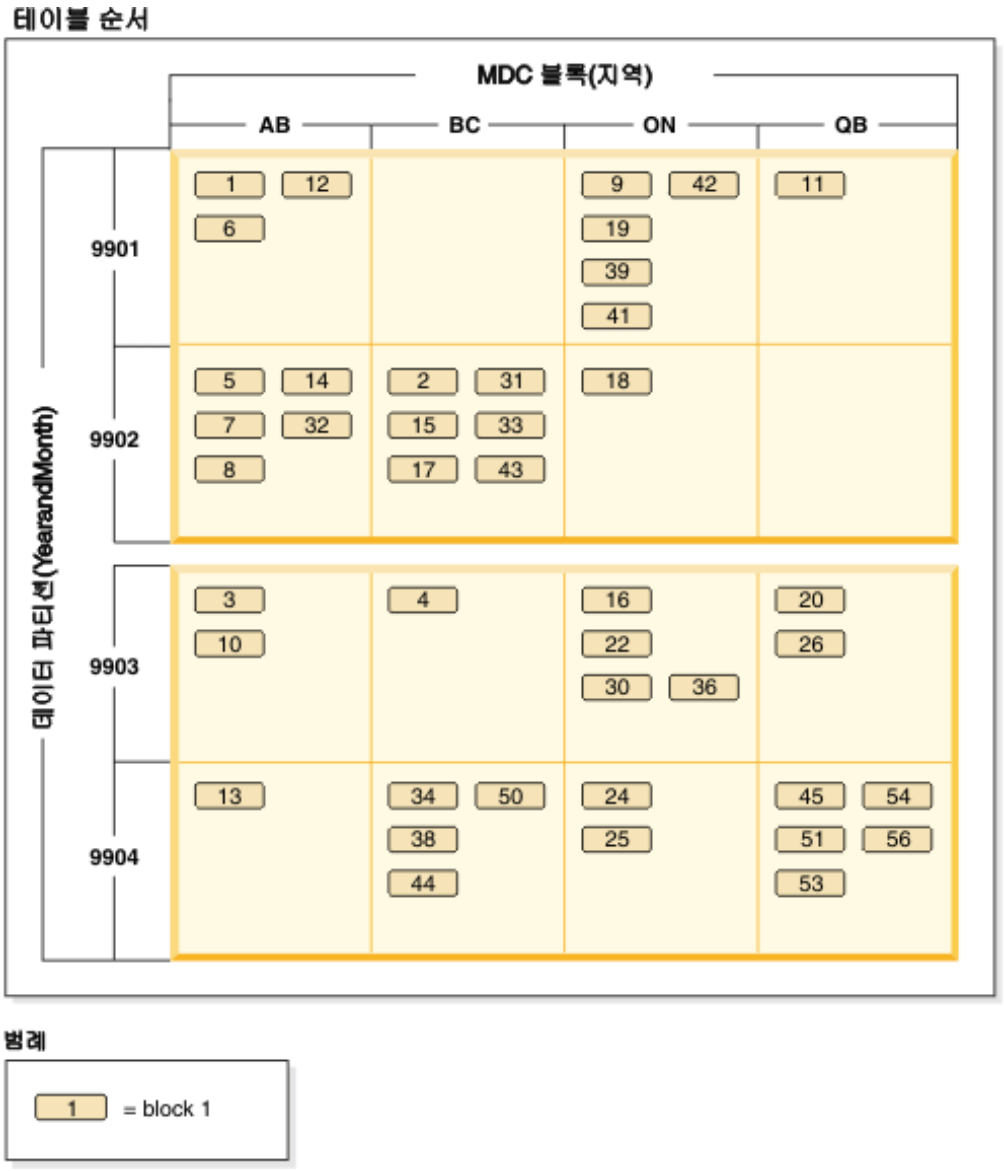


그림 7. YearAndMonth로 파티션되고 Province 및 YearAndMonth별로 구성되는 테이블

각 범위에 단일 값만 존재하도록 파티션되는 경우에 MDC 키에 테이블 파티션 컬럼을 포함시켜서 얻는 것이 없습니다.

고려사항

- 기본 테이블과 비교할 때, MDC 테이블 및 파티션된 테이블이 둘 다 추가 스토리지가 필요합니다. 이러한 스토리지 필요성은 부가적이지만, 이점이 있는 타당한 것으로 간주됩니다.
- 파티션된 데이터베이스 환경에서 MDC 기능을 테이블 파티션과 결합하지 않을 것을 선택하는 경우, 일반적으로 여기서 논의하는 시스템 유형에 해당하는 경우인 데이터 분산을 자신있게 예측할 수 있는 경우에 테이블 파티션이 최적입니다. 그렇지 않으면, MDC를 고려해야 합니다.

제 2 장 범위 클러스터 테이블(RCT)

범위 클러스터 테이블(RCT)은 테이블에서 레코드를 찾기 위해 사용되는 내부 ID인 사전 결정된 레코드 ID(RID)가 테이블의 각 레코드에 있는 테이블 레이아웃 스킴입니다.

데이터를 보유하는 각 테이블에 대해서 어떤 테이블 유형이 자신의 요구와 가장 부합하는지 고려하십시오. 예를 들어 촘촘하지 않게 클러스터되는(일정하게 증가하지 않음) 데이터 레코드의 경우에는 일반 테이블 및 인덱스를 사용하도록 고려하십시오. 데이터 레코드가 해당 키에 중복(고유하지 않은) 값을 갖는 경우 범위 클러스터 테이블을 사용해서는 안 됩니다. 사용하려는 범위 클러스터 테이블용 디스크에 일정량의 스토리지를 미리 할당할 수 없는 경우 이런 유형의 테이블을 사용해서는 안 됩니다. 이러한 요소들은 범위 클러스터 테이블로 사용할 수 있는 데이터인지 판별하는 데 유용합니다.

범위 클러스터 테이블 구조화 관련한 장점

범위 클러스터 테이블(RCT) 사용에는 몇 가지 이점이 있습니다.

- 직접 액세스

  액세스는 범위 클러스터 테이블 키의 RID 맵핑 기능을 통해 가능합니다.

- 낮은 유지보수

  B+ 트리와 같은 보조 구조를 삽입, 갱신 또는 삭제할 때마다 갱신할 필요가 없습니다.

- 적은 로그 횟수

  비슷한 크기의 일반 테이블 및 관련된 B+ 트리 인덱스와 비교할 경우 범위 클러스터 테이블에 대해 로그가 덜 수행됩니다.

- 적은 필수 버퍼 풀 메모리 필요

  보조 구조를 저장하기 위해 메모리를 추가할 필요가 없습니다.

- B+ 트리 테이블의 순서 등록 정보

  레코드의 순서는 B+ 트리 테이블이 추가 레벨 또는 B+ 트리 다음 키 잠금 스킴없이 획득한 순서와 동일합니다. RCT를 사용하면 일반 B+ 트리 인덱스에 비교했을 때 코드 경로 길이가 감소합니다. 그러나 이런 이점을 얻으려면 범위 클러스터 테이블을 DISALLOW OVERFLOW로 작성해야 하고 데이터는 조밀해야 합니다.

- 적은 수의 인덱스

  각 키를 디스크상의 위치에 맵핑하는 것은 다른 경우에 필요할 수 있는 인덱스를 하나 적게 사용하여 테이블을 작성할 수 있다는 것을 의미합니다. 범위 클러스터 테이

블을 사용하면 테이블의 데이터 액세스에 필요한 응용프로그램 요구사항에 따라 별도의 보조 인덱스가 필요하지 않습니다. 특히 응용프로그램에 인덱스가 필요한 경우에도 계속해서 일반 인덱스를 작성할 수 있습니다.

범위 클러스터 테이블(RCT) 비호환성

이러한 고려사항 이외에도 범위 클러스터 테이블이 사용될 수 있는 위치를 제한하거나 이 테이블에 대해 작업하지 않는 기타 유틸리티를 제한하는 일부 비호환성이 있습니다.

범위 클러스터 테이블에 대한 제한사항은 다음과 같습니다.

- 파티션된 테이블에 대한 범위 클러스터 테이블은 지원되지 않습니다.

  범위 클러스터링을 사용하여 파티션된 테이블을 작성하려 하면 오류 메시지 SQL0270 rc=87이 리턴됩니다.
- 선언된 임시 테이블 및 작성된 임시 테이블은 지원되지 않습니다.

  이 임시 테이블은 범위 클러스터 등록 정보를 사용할 수 없습니다.
- 자동 요약 테이블(AST)이 지원되지 않습니다.

  이 테이블은 범위 클러스터 등록 정보를 사용할 수 없습니다.
- 로드 유틸리티가 지원되지 않습니다.

  행은 임포트 조작이나 병렬 삽입 응용프로그램을 통해 한 번에 하나씩 삽입해야 합니다.
- REORG TABLE 유틸리티가 지원되지 않습니다.

  DISALLOW OVERFLOW로 정의된 범위 클러스터 테이블은 재구성할 필요가 없습니다. ALLOW OVERFLOW로 정의된 해당 범위 클러스터 테이블은 여전히 이 오버플로우 영역의 데이터를 재구성할 수 없습니다.
- 한 논리적 시스템만의 범위 클러스터 테이블

  데이터베이스 파티션 기능(DPF)이 있는 ESE(Enterprise Server Edition)에서 범위 클러스터 테이블은 둘 이상의 데이터베이스 파티션이 포함된 데이터베이스에 존재할 수 없습니다.
- 디자인 어드바이저는 범위 클러스터 테이블을 권장하지 않습니다.
- 범위 클러스터 테이블이 정의에 따라 이미 클러스터되었습니다.

  이는 다음의 클러스터링 스킴이 범위 클러스터 테이블과 호환되지 않음을 의미합니다.

  – 다차원 클러스터(MDC) 테이블

  – 클러스터링 인덱스

- 값 및 디폴트 압축이 지원되지 않습니다.
- 범위 클러스터 테이블에서의 역방향 스캔이 지원되지 않습니다.
- IMPORT 명령의 REPLACE 옵션이 지원되지 않습니다.
- ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY문의 WITH EMPTY TABLE 옵션이 지원되지 않습니다.ACTIVATE NOT LOGGED INITIALLY문이 지원되지 않습니다.

범위 클러스터 테이블(RCT) 및 범위밖 레코드 키 값

CREATE TABLE문과 ALLOW OVERFLOW 옵션을 사용하여 오버플로우 레코드를 허용하는 범위 클러스터 테이블(RCT)의 동작을 제어하십시오. 이런 방법으로, 정의된 범위 내에 있는 테이블에 필요한 모든 페이지가 즉시 할당되도록 하십시오.

일단 작성되면, 허용되는 오버플로우 옵션으로 테이블이 작성되었는지 또는 허용되지 않는 오버플로우 옵션으로 테이블이 작성되었는지와 관계없이 정의된 범위에 해당하는 키를 가진 모든 레코드는 동일한 방법으로 동작합니다. 정의된 범위밖에 해당하는 키를 가진 레코드가 있는 경우에는 다릅니다. 그런 경우, 테이블이 오버플로우 레코드를 허용하면 레코드는 오버플로우 영역에 위치하게 되는데 오버플로우 영역은 동적으로 할당됩니다. 정의된 범위밖에서 더 많은 레코드가 추가될 경우 레코드는 증가하는 오버플로우 영역에 위치합니다. 오버플로우 영역은 조치의 일부로 액세스해야 하기 때문에 해당 오버플로우 영역이 포함된 테이블 조치에는 더 많은 처리 시간이 걸립니다. 오버플로우 영역이 클수록 오버플로우 영역을 액세스하는데 걸리는 시간은 길어집니다. 늘어난 오버플로우 영역을 사용한 후에는 데이터를 테이블에서 새 범위 클러스터 테이블(새 확장 범위를 사용해 정의한)로 익스포트하여 영역 크기를 줄이십시오.

범위 클러스터 테이블에 위치한 레코드의 레코드 키 값이 허용되거나 정의된 범위밖에 있지 않도록 할 수도 있습니다. 이런 유형의 RCT 경우에는 CREATE TABLE문에서 DISALLOW OVERFLOW 옵션을 사용해야 합니다. 일단 이런 유형의 RCT를 작성하면 레코드 키 값이 허용 또는 정의된 범위를 벗어날 경우 오류 메시지가 나타날 수도 있습니다.

범위 클러스터 테이블(RCT) 잠금

정상적인 처리에서는 레코드 잠금을 사용하여 한 번에 하나의 응용프로그램 또는 한 사용자만 레코드 또는 레코드 그룹에 액세스할 수 있도록 합니다. 범위 클러스터 테이블을 사용하면 키 및 다음 키 잠금 대신 『이산 잠금』을 사용합니다.

이러한 방법을 사용할 경우, 응용프로그램 또는 사용자가 요청한 작업에 의해 영향을 받았거나 영향을 받을 수 있는 모든 레코드가 잠깁니다. 설정되는 잠금 수는 분리 레벨에 따라 다릅니다.

현재는 비어 있지만 미리 할당된 범위 클러스터 테이블의 행에 대한 규정이 잠깁니다. 이런 방법을 사용하면 다음 키 잠금을 수행할 필요가 없습니다. 결과적으로, 조밀한 범위 클러스터 테이블에 필요한 잠금 수가 줄어듭니다.

제 3 장 다차원적으로 클러스터된(MDC) 테이블

다차원적으로 클러스터된 테이블

다차원적으로 클러스터된(MDC)은 유연성있고 연속적이며 자동적인 방식으로 다중 차원에서 테이블의 데이터를 클러스터링하는 세련된 메소드를 제공합니다. MDC는 쿼리 성능을 상당히 개선시킵니다.

또한 MDC는 데이터 유지보수(예: 재구성)와 삽입, 갱신, 삭제 조작 중의 인덱스 유지보수 조작의 오버헤드를 상당히 줄일 수 있습니다. MDC는 주로 데이터 웨어하우스 및 대형 데이터베이스 환경에서 사용하지만 온라인 트랜잭션 처리(OLTP) 환경에서도 사용할 수 있습니다.

일반 및 MDC 테이블 비교

일반 테이블에는 레코드 기반 인덱스가 있습니다. 인덱스의 모든 클러스터링은 단일 차원으로 제한됩니다. 버전 8 이전에 데이터베이스 관리 프로그램에서는 클러스터링 인덱스를 통한 데이터의 단일 차원 클러스터링만 지원했습니다. 테이블에 레코드가 삽입되고 갱신될 때 데이터베이스 관리 프로그램은 클러스터링 인덱스를 사용하여 인덱스의 키 순서대로 페이지에서 데이터의 실제 순서를 유지보수하려고 시도합니다.

클러스터링 인덱스는 클러스터링 인덱스의 키가 포함된 술어가 있는 범위 쿼리의 성능을 상당히 향상시킵니다. 양호한 클러스터링 인덱스를 통해 성능이 개선됩니다. 이는 테이블의 일부만 액세스되므로 보다 효율적인 프리페치를 수행할 수 있기 때문입니다.

그러나 클러스터링 인덱스를 사용하는 데이터 클러스터링에는 몇 가지 단점이 있습니다. 첫 번째, 스페이스가 데이터 페이지에서 일정 시간에 걸쳐 채워지므로 클러스터링이 보장되지 않습니다. 삽입 조작은 동일하거나 유사한 클러스터링 키 값을 갖는 페이지 근처의 페이지에 레코드를 추가하려고 시도하지만, 이상적인 위치에서 스페이스가 발견되지 않으면 테이블의 다른 위치에 레코드가 삽입됩니다. 따라서 테이블을 다시 클러스터링하고 향후 클러스터된 삽입 요청을 수용하기 위한 여유 공간이 있는 페이지를 설정하기 위해 주기적 테이블 재구성이 필요합니다.

두 번째, 하나의 인덱스만 『클러스터링』 인덱스로 지정할 수 있으며 모든 다른 인덱스는 클러스터링이 해제됩니다. 왜냐하면 데이터는 하나의 차원에서만 실제로 클러스터될 수 있기 때문입니다. 이러한 제한사항은 모든 인덱스가 버전 8.1 이전이므로 클러스터링 인덱스가 레코드 기반이라는 사실과 관련이 있습니다.

세 번째, 레코드 기반 인덱스에는 테이블의 모든 단일 레코드에 대한 포인터가 포함되어 있으므로 크기가 매우 클 수 있습니다.

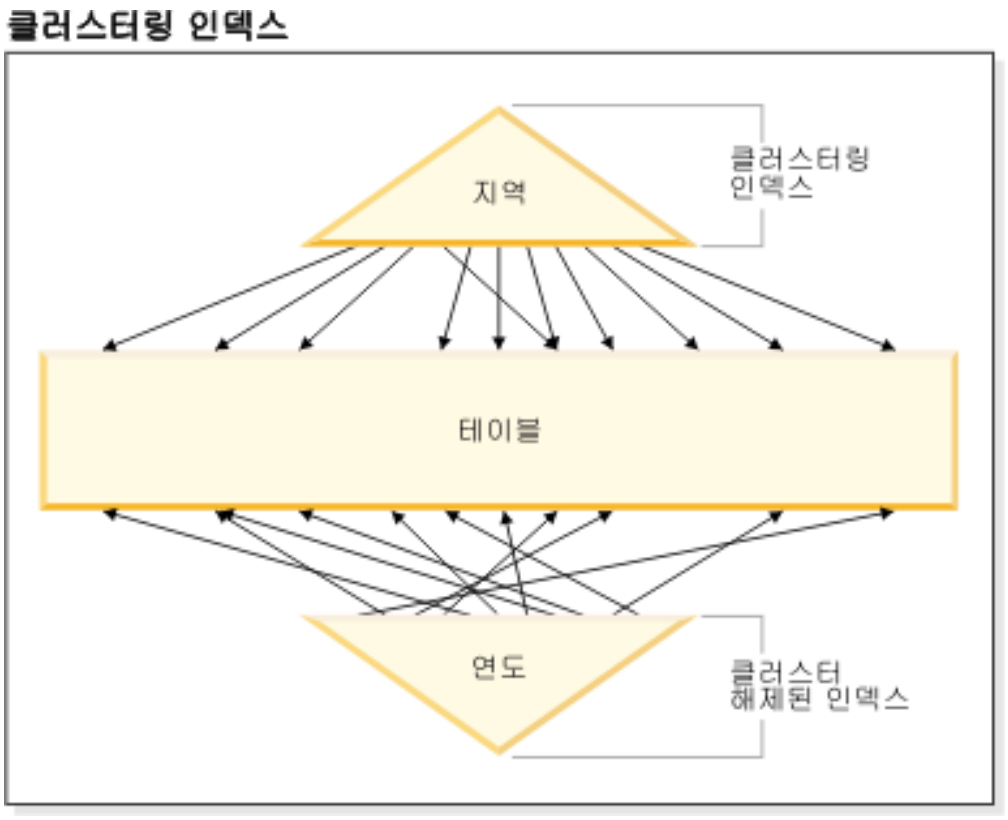


그림 8. 클러스터링 인덱스가 있는 일반 테이블

그림 8의 테이블에는 두 개의 레코드 기반 인덱스가 정의되어 있습니다.

- 『지역』의 클러스터링 인덱스
- 『연도』의 다른 인덱스

『지역』 인덱스는 클러스터링 인덱스입니다. 이는 키가 인덱스에서 스캔될 때, 대응되는 레코드가 대부분 테이블에서 동일하거나 이웃에 있는 페이지에서 발견되어야 함을 의미합니다. 이와는 반대로 『연도』 인덱스는 클러스터링되지 않았습니다. 이는 키가 해당 인덱스에서 스캔될 때, 대응되는 레코드가 테이블 전체를 통해 무작위의 페이지에서 발견될 수 있음을 의미합니다. 클러스터링 인덱스에서의 스캔은 I/O 성능이 더 높고 해당 인덱스에 대해 데이터가 보다 잘 클러스터되어 있을수록 순차 프리페치로부터 보다 많은 이점을 얻습니다.

MDC에서는 블록 기반의 인덱스를 처음으로 사용합니다. 『블록 인덱스』는 개별 레코드 대신에 블록 또는 레코드 그룹을 가리킵니다. 클러스터링 값에 따라 물리적으로 MDC 테이블의 데이터를 블록으로 구성한 후에 블록 인덱스를 사용하여 이러한 블록을 액세스함으로써, MDC는 클러스터된 인덱스의 모든 단점을 해결할 뿐 아니라 많은 추가적인 성능상의 이점도 제공할 수 있습니다.

첫 번째, MDC는 둘 이상의 키 또는 차원에서 동시에 테이블을 실제로 클러스터할 수 있게 합니다. 따라서 MDC에서 단일 차원 클러스터링의 이점은 다중 차원 또는 클러스터링 키로 확장됩니다. 테이블의 하나 이상의 지정된 차원의 클러스터링이 있으면 쿼리 성능이 향상됩니다. 이러한 쿼리 액세스는 올바른 차원 값이 있는 레코드가 포함된 페이지에만 액세스할 뿐만 아니라 이러한 규정 페이지는 블록 또는 Extent로 그룹화됩니다.

두 번째, 클러스터링 인덱스가 있는 테이블은 일정 시간이 지나면 클러스터링이 해제될 수 있지만, 대부분의 경우 MDC 테이블은 모든 차원의 클러스터링을 자동 및 연속으로 유지보수 및 보증할 수 있습니다. 그러면 데이터의 실제 순서를 리스토어하기 위해 MDC 테이블을 자주 재구성할 필요가 없어집니다. 블록 내의 레코드 순서는 항상 유지보수되지만 블록의 실제 순서화(즉, 블록 인덱스 스캔에서 한 블록에서 다른 블록으로)는 삽입 시(또는 어떤 경우에는 초기 로드 시에도) 유지보수되지 않습니다.

세 번째, MDC에서 클러스터된 인덱스는 블록 기반입니다. 이 인덱스는 일반 레코드 기반 인덱스보다 매우 작으므로 디스크 스페이스를 훨씬 덜 차지하며 스캔 속도가 빠릅니다.

MDC 테이블 차원 선택

일단 다차원적으로 클러스터된 테이블에 대해 작업하기로 결정한 경우, 선택하는 차원은 테이블을 사용하며 블록 레벨 클러스터링의 이점을 활용할 쿼리의 유형에 의존할 뿐 아니라 실제 데이터의 양 및 분산에 보다 중요하게 의존합니다.

MDC의 도움을 받을 쿼리

테이블에 대한 클러스터링 차원 선택 시 첫 번째 고려사항은 블록 레벨에서 클러스터링의 도움을 받을 쿼리를 판별하는 것입니다. 일반적으로 데이터에 대해 수행될 작업을 구성하는 쿼리를 기반으로 하여 차원을 선택할 경우 몇 가지 후보들이 있습니다. 이 후보들의 순위는 중요합니다. 등호 또는 범위 술어 쿼리에 포함된 컬럼(특히 카디낼리티가 낮은 컬럼)은 클러스터링 차원에서 가장 많은 이점을 나타내므로 이에 대한 후보로 고려해야 합니다. 차원 테이블과의 스타 조인과 관련된 MDC 사실 테이블에 외부 키에 대한 차원 작성을 고려할 수도 있습니다. 둘 이상의 차원에 대한 자동 및 연속 클러스터링과 Extent 또는 블록 레벨의 클러스터링에는 성능상의 이점이 있다는 점을 유념해야 합니다.

다차원 클러스터링의 이점을 이용할 수 있는 다수의 쿼리가 있습니다. 이러한 쿼리의 예는 다음과 같습니다. 이러한 몇 개의 예에서 c1, c2 및 c3 차원이 있는 MDC 테이블 t1이 있다고 가정합니다. 기타 예에서 color 및 nation 차원이 있는 MDC 테이블 mdctable이 있다고 가정합니다.

예 1

```
SELECT .... FROM t1 WHERE c3 < 5000
```

이 쿼리는 단일 차원에 대한 범위 술어가 있으므로 c3에 대한 차원 블록 인덱스를 사용하여 테이블에 액세스하도록 내부적으로 다시 작성할 수 있습니다. 값이 5000 미만인 키의 블록 ID(BID)를 찾기 위해 인덱스를 스캔하며, 최소 관련 스캔을 결과 블록 세트에 적용하여 실제 레코드를 검색합니다.

예 2

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

이 쿼리는 단일 차원에 대한 IN 술어가 있으며 블록 인덱스 기반 스캔을 트리거할 수 있습니다. 이 쿼리는 c2에 대한 차원 블록 인덱스를 사용하여 테이블에 액세스하기 위해 내부적으로 다시 작성할 수 있습니다. 값이 1과 2037인 키의 BID를 찾기 위해 인덱스를 스캔하며 최소 관련 스캔을 결과 블록 세트에 적용하여 실제 레코드를 검색합니다.

예 3

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND NATION='USA'
```

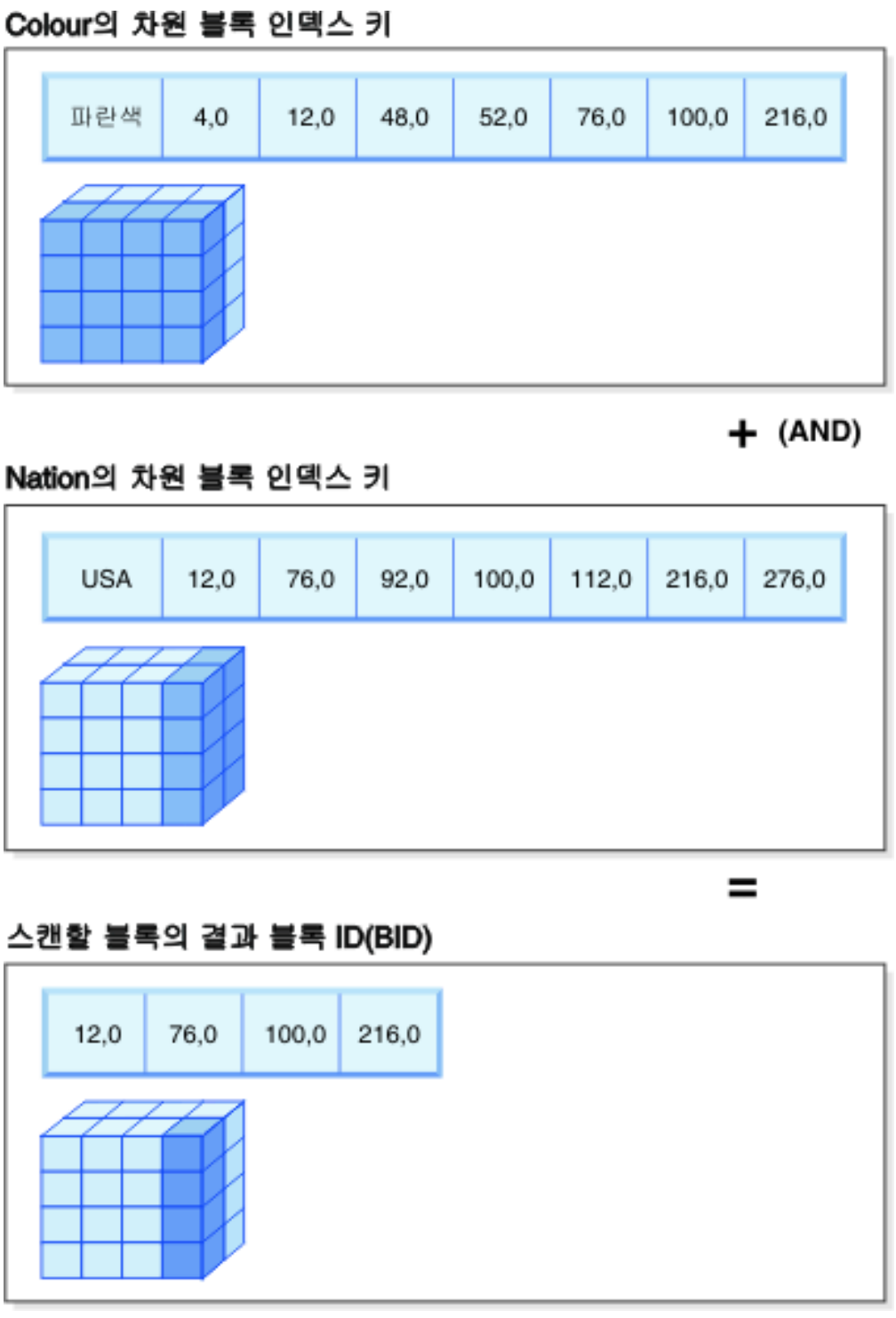


그림 9. 두 블록 인덱스에서 논리적 AND 연산을 사용하는 쿼리 요청

이 쿼리 요청을 수행하기 위해 다음과 같은 작업이 수행됩니다(그림 9 참조).

- 차원 블록 인덱스 찾아보기가 수행됩니다. 하나는 Blue 조각에 관한 것이며 다른 하나는 USA 조각에 관한 것입니다.

- 블록 논리적 AND 연산이 두 조각의 교차 지점을 판별하기 위해 수행됩니다. 즉, 논리적 AND 연산은 두 조각 모두에서 찾은 해당 블록만 판별합니다.
- 테이블에서 결과 블록의 최소 관련 스캔이 수행됩니다.

예 4

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

이 쿼리는 논리적 AND 연산뿐만 아니라 c2 및 c3에 대한 범위 술어와 c1에 대한 등호를 포함합니다. 이 쿼리는 각각의 차원 블록 인덱스에서 테이블에 액세스하기 위해 내부적으로 다시 작성할 수 있습니다.

- c2 블록 인덱스의 스캔은 값이 100을 초과하는 키의 BID를 찾습니다.
- c3 블록 인덱스의 스캔은 값이 1000과 5000 사이인 키의 BID를 찾습니다.
- c1 블록 인덱스의 스캔은 값이 '16/03/1999'인 키의 BID를 찾습니다.

그러면 논리적 AND 연산이 각 블록 스캔의 결과 BID에서 수행되어 교집합을 찾으며, 최소 관련 스캔이 블록의 결과 세트에 적용되어 실제 레코드를 찾습니다.

예 5

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR NATION='USA'
```

이 쿼리 요청을 수행하기 위해 다음과 같은 작업이 수행됩니다.

- 차원 블록 인덱스 찾아보기가 수행됩니다. 조각당 하나가 대응됩니다.
- 논리적 OR 연산이 두 조각의 합집합을 찾기 위해 수행됩니다.
- 테이블에서 결과 블록의 최소 관련 스캔이 수행됩니다.

예 6

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

이 쿼리는 OR 연산뿐만 아니라 c1 차원에 대한 범위 술어와 c2 차원에 대한 IN 술어를 포함합니다. 이 쿼리는 차원 블록 인덱스 c1 및 c2에 대한 테이블에 액세스하기 위해 내부적으로 다시 작성할 수 있습니다. c1 차원 블록 인덱스의 스캔은 5000 미만의 값을 찾고, c2 차원 블록 인덱스의 다른 스캔은 1, 2 및 3 값을 찾습니다. 논리적 OR 연산은 각 블록 인덱스 스캔에서 결과 BID를 찾은 다음, 최소 관련 스캔을 결과 세트에 적용하여 결과 목록 세트에 실제 레코드를 찾습니다.

예 7

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

이 쿼리는 AND 연산뿐만 아니라 c1 차원에 대한 등호 술어와 차원이 아닌 컬럼에 대한 다른 범위 술어를 포함합니다. 이 쿼리는 c1에 대한 차원 블록 인덱스에 액세스하기 위해 내부적으로 다시 작성하여 c1이 15인 테이블의 조각에서 블록 목록을 얻을 수

있습니다. c4에 RID 인덱스가 있는 경우, 인덱스 스캔을 수행하여 c4가 12 미만인 레코드의 RID를 검색할 수 있으며 블록의 결과 목록은 이 레코드 목록을 사용하여 논리적 AND 연산을 수행합니다. 이러한 교집합은 c1이 15인 블록에서 찾을 수 없는 RID를 제거하며 규정되는 블록에서 발견되는 해당 RID의 목록만 테이블에서 검색됩니다.

c4에 대한 RID 인덱스가 없는 경우, 규정 블록 목록에서 블록 인덱스를 스캔할 수 있으며 각 블록의 최소 관련 스캔 중에 술어 c4 < 12가 발견된 각 레코드에 적용될 수 있습니다.

예 8

부품 번호의 행 ID(RID) 인덱스 및 색상, 연도, 국가에 대한 차원이 있는 시나리오가 제공되면 다음과 같은 쿼리가 가능합니다.

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND PARTNO < 1000
```

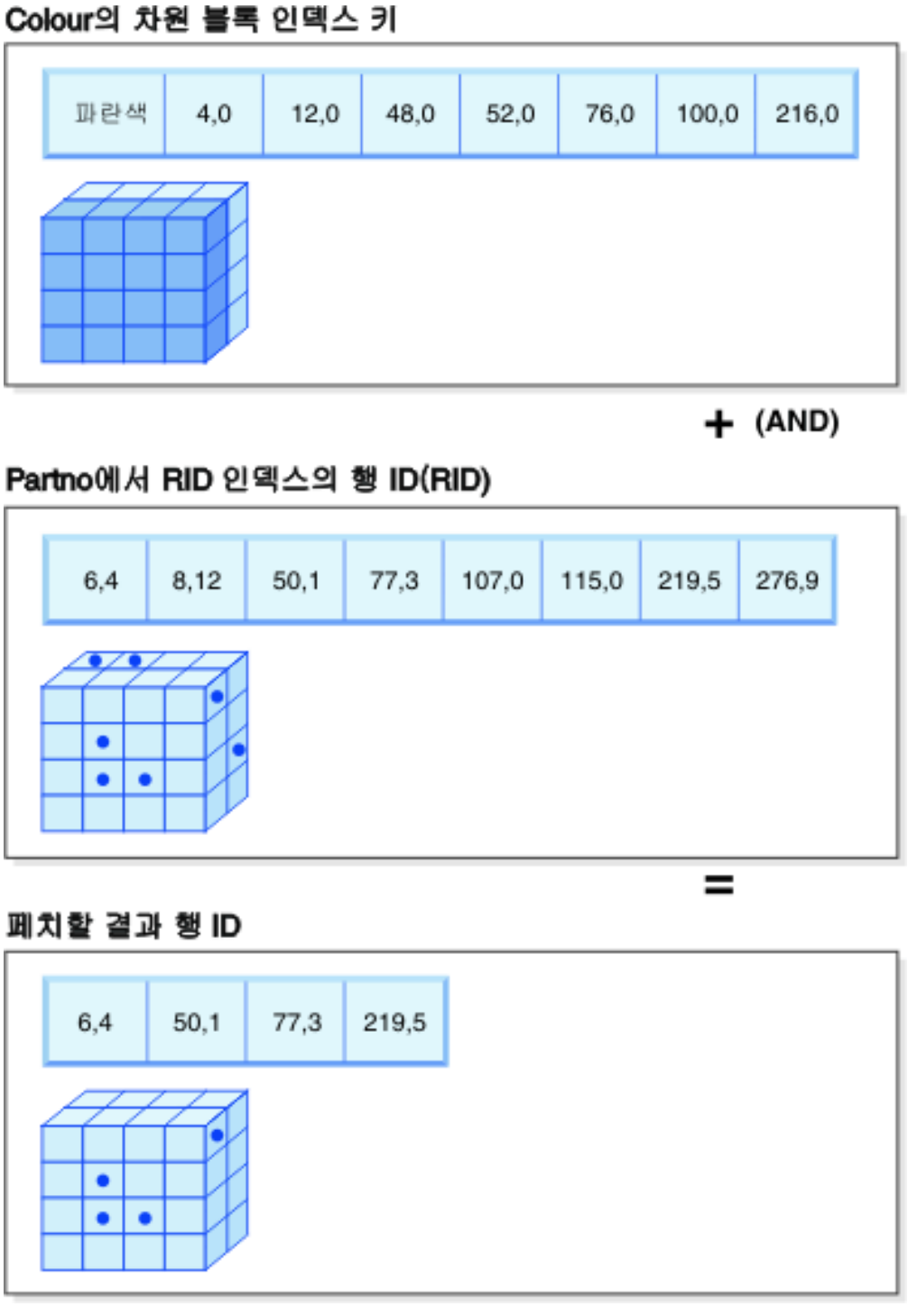


그림 10. 블록 인덱스 및 행 ID(RID) 인덱스에서 논리적 AND 연산을 사용하는 쿼리 요청

이 쿼리 요청을 수행하기 위해 다음과 같은 작업이 수행됩니다(그림 10 참조).

- 차원 블록 인덱스 찾아보기 및 RID 인덱스 찾아보기가 수행됩니다.
- 술어 조건을 충족하는 해당 행 및 조각의 교집합을 판별하기 위해 논리적 AND 조작이 블록 및 RID와 함께 사용됩니다.
- 규정되는 블록에도 속하는 해당 RID만 검색됩니다.

예 9

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR PARTNO < 1000
```

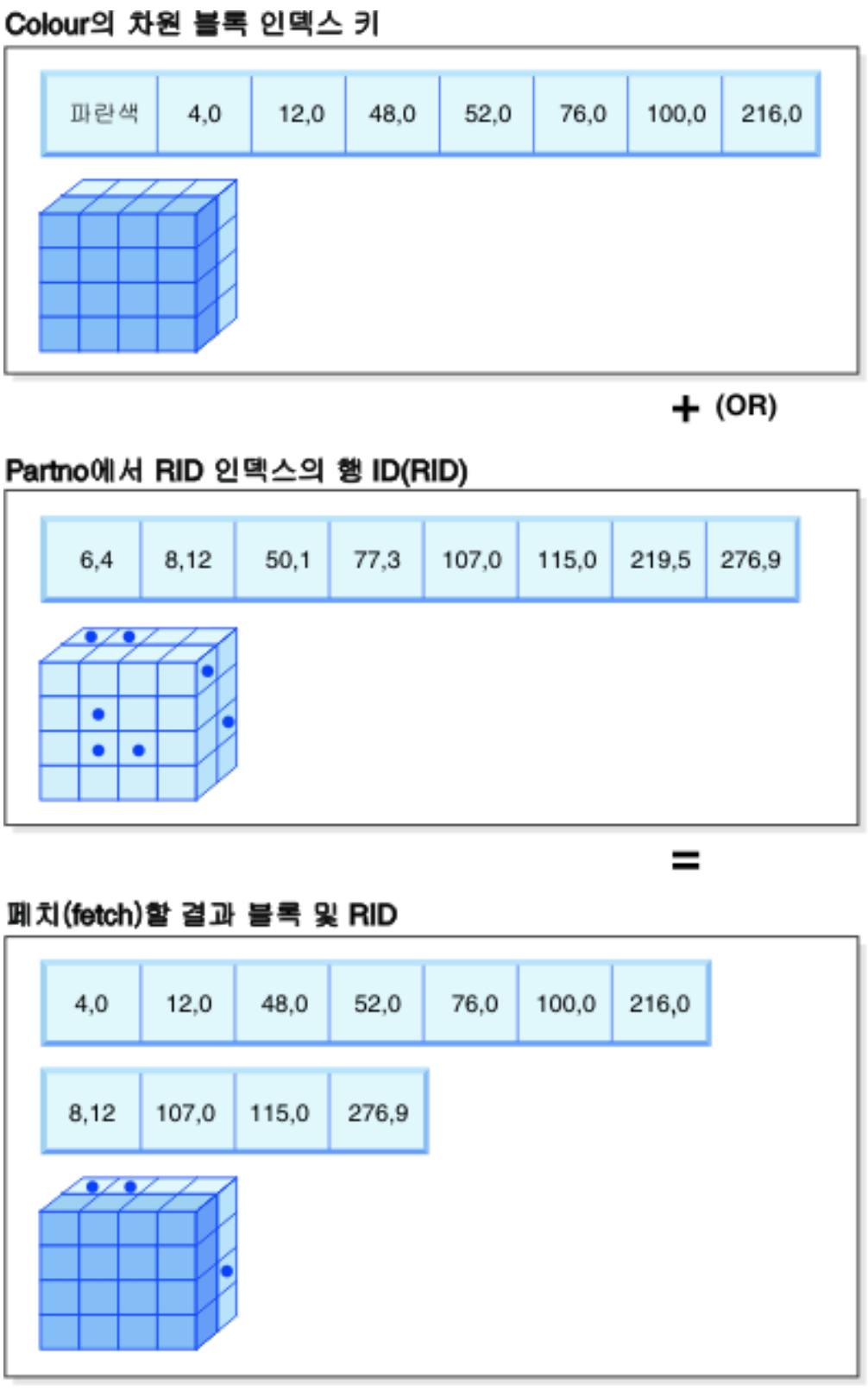


그림 11. OR 연산을 사용하는 블록 인덱스 및 행 ID가 작동하는 방식

이 쿼리 요청을 수행하기 위해 다음과 같은 작업이 수행됩니다(그림 11 참조).

- 차원 블록 인덱스 찾아보기 및 RID 인덱스 찾아보기가 수행됩니다.

- 술어 조건을 충족하는 해당 행 및 조각의 교집합을 판별하기 위해 논리적 OR 조작이 블록 및 RID와 함께 사용됩니다.
- 규정되는 블록의 모든 행 및 술어 조건을 충족하는 규정되는 블록의 밖에 있는 추가 RID가 검색됩니다. 각각의 블록의 최소 관련 스캔이 자체 레코드를 검색하기 위해 수행되며, 이러한 블록 밖에 있는 추가 레코드가 개별적으로 검색됩니다.

예 10

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

이 쿼리는 c1에 대한 범위 술어와 논리적 OR 연산뿐만 아니라 차원이 아닌 컬럼 c4에 대한 등호 술어를 포함합니다. c4 컬럼에 RID 인덱스가 있는 경우, 이는 c1의 차원 블록 인덱스 및 c4의 RID 인덱스를 사용하여 논리적 OR 연산을 수행하기 위해 내부적으로 다시 기록될 수 있습니다. c4에 대한 인덱스가 없는 경우, 모든 레코드를 확인해야 하기 때문에 테이블 스캔을 대신 선택할 수도 있습니다. 논리적 OR 연산은 4 미만의 값에 대해 c1의 블록 인덱스 스캔을 사용함은 물론 100 값에 대해 c4의 RID 인덱스 스캔을 사용합니다. 해당 블록 내의 모든 레코드가 규정되며 해당 블록 밖에 있는 레코드에 대한 모든 추가 RID도 검색되므로 최소 관련 스캔이 규정되는 각각의 블록에서 수행됩니다.

예 11

```
SELECT .... FROM t1,d1,d2,d3
   WHERE t1.c1 = d1.c1 and d1.region = 'NY'
            AND t2.c2 = d2.c3 and d2.year='1994'
      AND t3.c3 = d3.c3 and d3.product='basketball'
```

이 쿼리에는 스타 조인이 있습니다. 이 예에서 t1은 사실 테이블이고 여기에는 차원 테이블의 기본 키 d1, d2 및 d3에 해당하는 외부 키 c1, c2 및 c3이 있습니다. 차원 테이블이 MDC 테이블일 필요는 없습니다. Region, Year 및 Product는 일반 또는 블록 인덱스를 사용하여 인덱스화할 수 있는 관련 차원 테이블의 컬럼입니다(차원 테이블이 MDC 테이블인 경우). c1, c2 및 c3 값에서 사실 테이블에 액세스하는 경우에는 해당 컬럼에서 차원 블록 인덱스의 블록 인덱스 스캔을 수행한 다음 결과 BID를 사용하여 논리적 AND 조작을 수행할 수 있습니다. 블록 목록이 있는 경우, 각 블록에 대해 최소 관련 스캔을 수행하여 레코드를 얻을 수 있습니다.

셀 밀도

적합한 차원 및 Extent 크기에 대해 작성된 선택 항목은 MDC 설계에 **매우** 중요합니다. 이러한 요소들은 테이블의 예상 셀 밀도를 결정합니다. 이는 셀의 레코드 수와 무관하게 Extent가 모든 기존 셀에 대해 할당되기 때문에 중요합니다. 올바른 선택 항목은 블록 기반 인덱스 및 다차원 클러스터링을 활용하므로 결과적으로 성능이 향상됩니다. 목표는 블록을 가득 채움으로써 다차원 클러스터링의 이점을 최대한 활용하고 최적으로 스페이스를 활용하는 것입니다.

따라서 다차원 테이블을 설계할 때 가장 중요한 고려사항은 현재 및 예상 데이터를 기반으로 하는 테이블의 예상 셀 밀도입니다. 쿼리 수행 결과에 따라 차원 세트를 선택할 수 있는데 이로 인해 각 차원의 가능한 수를 기반으로 하는 테이블의 잠재적 셀 수가 매우 커집니다. 테이블의 가능한 셀 수는 각 차원의 카디널리티(cardinality)에 대한 데카르트 곱과 같습니다. 예를 들어, 차원 Day, Region 및 Product에 대한 테이블을 클러스터하는 데 데이터가 5년까지 수용하는 경우, 테이블에는 `1821 days * 12 regions * 5 products = 109 260`개의 가능한 셀이 있을 수 있습니다. 레코드 수가 적은 셀에는 해당 셀의 레코드를 저장하기 위해, 전체 페이지 블록이 계속 할당되어야 합니다. 블록 *크기*가 큰 경우, 이 테이블은 실제로 요구되는 *크기*보다 훨씬 더 클 수 있습니다.

최적 셀 밀도에 기여할 수 있는 다음과 같은 몇 가지 설계 인수들이 있습니다.

- 차원의 수를 변경합니다.
- 하나 이상의 차원의 세분화도를 변경합니다.
- 테이블 스페이스의 블록(Extent) *크기* 및 페이지 *크기*를 변경합니다.

최상의 설계가 가능하도록 하려면 다음 단계를 수행하십시오.

1. 후보 차원을 식별하십시오.

   블록 레벨 클러스터링의 혜택을 받을 쿼리를 판별하십시오. 다음 특성 중 일부 또는 모두를 갖는 컬럼에 대한 잠재적 워크로드를 검사하십시오.

   - 임의의 IN-목록 술어의 범위 및 등호
   - 데이터의 롤인 또는 롤아웃
   - Group-by 및 Order-by절
   - Join절(특히 스타 스키마 환경에서)
2. 셀 개수를 예측하십시오.

   후보 차원의 세트에서 구성된 테이블에서 사용할 수 있는 잠재적인 셀 수를 식별하십시오. 데이터에서 발생하는 차원 값의 고유 조합 수를 판별하십시오. 테이블이 존재하는 경우에는 테이블에 대한 차원이 될 각 컬럼의 개별 값 수를 단순히 선택하여 현재 데이터에 대해 정확한 수를 판별할 수 있습니다. 또는 테이블에 대한 통계를 보유하는 경우에는 차원 후보에 대한 컬럼 카디널리티를 곱해서 예상값을 판별할 수 있습니다.

   주: 테이블이 파티션된 데이터베이스 환경에 있고 분산 키가 고려된 어떤 차원에도 관련되어 있지 않으면, 모든 데이터를 취하고 데이터베이스 파티션 수로 나누어서 셀 당 평균 데이터량을 판별해야 합니다.
3. 스페이스 점유율 또는 밀도를 예측하십시오.

   평균적으로 소수의 행만 저장된 부분적으로 채워진 한 개의 블록이 각각의 셀에 있다고 가정하십시오. 행 수가 점점 줄어듦에 따라 부분적으로 채워진 블록이 늘어나

게 됩니다. 또한 일반적으로(데이터 왜곡이 없거나 거의 없다고 가정함) 셀당 레코드 수는 테이블의 레코드 수를 셀 수로 나누면 구할 수 있습니다. 그러나 테이블이 파티션된 데이터베이스 환경에 있을 경우 블록이 데이터베이스 파티션 기반으로 데이터에 할당되기 때문에 각 데이터베이스 파티션에서 셀당 존재하는 레코드 수를 고려해야 합니다. 파티션된 데이터베이스 환경에서 스페이스 밀도 및 점유율을 계산하는 경우에는 전체 테이블이 아니라 각 데이터베이스 파티션에 평균적인 셀별 레코드 수를 고려해야 합니다. 자세한 내용은 『다차원적으로 클러스터된(MDC) 테이블 작성, 배치 및 사용』을 참조하십시오.

밀도 향상을 위한 몇 가지 방법은 다음과 같습니다.

- 부분적으로 채워진 블록이 스페이스를 덜 차지하도록 블록 *크기*를 줄이십시오.

  Extent *크기*를 적절히 작게 유지하여 각 블록의 *크기*를 줄이십시오. 각 셀에 부분적으로 채워진 블록은 있지만 내부에 적은 수의 레코드를 지닌 한 개의 블록만 있는 경우에는 스페이스를 덜 소모합니다. 그러나 레코드가 많은 셀의 경우, 이러한 셀을 수용하려면 보다 많은 블록이 필요합니다. 그러면 블록 인덱스에서 이들 셀의 블록 ID(BID) 수가 증가하며 블록이 보다 빠르게 비워지고 가득 차게 됨에 따라 이러한 인덱스가 더욱 커지며 잠재적으로 이들 인덱스에 대한 보다 많은 삽입 및 삭제 조작이 발생합니다. 또한, 이러한 보다 많이 채워진 셀 값에 대해 테이블의 클러스터된 데이터의 그룹이 더 작아지는 반면, 클러스터된 데이터의 더 큰 그룹 수는 줄어듭니다.
- 차원 수를 줄이거나 생성된 컬럼에서 셀의 세분화도를 늘려서 셀 수를 줄이십시오.

  보다 낮은 카디널리티(cardinality)를 제공하기 위해 하나 이상의 차원을 보다 낮은 세분화도로 롤업할 수 있습니다. 예를 들어, Region 및 Product에 대한 이전 예의 데이터를 계속 클러스터할 수 있지만 Day의 차원을 YearAndMonth의 차원으로 대체해야 합니다. 그러면 가능한 셀 수는 3600이고 YearAndMonth, Region 및 Product에 대해 60(12달 x 5년), 12 및 5의 카디널리티가 제공됩니다. 각 셀은 보다 큰 범위의 값을 보유하며 레코드 수도 상당히 줄어듭니다.

  또한 대부분이 날짜의 월인지 아니면 분기 또는 일인지의 여부와 같이 관련된 컬럼에서 일반적으로 사용되는 술어를 고려해야 합니다. 이 점은 차원의 세분화도를 변경하는 것이 바람직한지 여부에 영향을 줍니다. 예를 들어, 대부분의 술어가 특정 일에 대한 것이며 월에 대한 테이블을 클러스터한 경우, Linux, UNIX 및 Windows용 DB2 데이터베이스는 YearAndMonth의 블록 인덱스를 사용하여 원하는 날짜가 포함된 월의 범위를 빠르게 좁히고 연관된 블록에만 액세스합니다. 그러나 블록을 스캔할 때 일 술어를 적용하여 규정하는 일 수를 결정해야 합니다. 하지만 일에 대해 클러스터할 경우(그리고 일에 높은 카디널리티(cardinality)가 있는 경우), 일에 대한 블록 인덱스를 사용하여 스캔할 블록을 결정할 수 있으며 일 술어는 규정하는 각 셀의 첫 번째 레코드에만 다시 적용해야

합니다. 이 경우, 사용자 정의 함수(UDF)를 사용하여 12개의 다른 값이 포함된 Region 컬럼을 Regions West, North, South 및 East로 롤업하는 것과 같이 다른 차원 중 하나를 롤업하여 셀의 밀도를 증가시키는 것이 바람직할 수 있습니다.

MDC 테이블을 작성 시 고려사항

MDC 테이블을 작성할 때 고려해야 할 요소에는 여러 가지가 있습니다. 다음 절에서는 MDC 테이블을 작성, 배치 및 사용하는 방법에 대한 결정이 현재 데이터베이스 환경(예: 파티션된 데이터베이스가 존재 또는 없음) 및 MDC 테이블에 대한 차원 선택에 따라 어떻게 영향을 받을 수 있는지에 대해 설명합니다. 또한 DB2 디자인 어드바이저 및 일부 문제에 대한 권장사항을 제공하기 위해 이 도구를 어떻게 사용하는지에 대해서도 설명합니다.

기존 테이블의 데이터를 MDC 테이블로 이동

쿼리 성능을 개선하고 데이터 웨어하우스 또는 대형 데이터베이스 환경에서 데이터 유지보수 조작의 오버헤드를 줄이기 위해 일반 테이블에서 다차원적으로 클러스터된(MDC) 테이블로 데이터를 이동할 수 있습니다. 기존 테이블에서 MDC 테이블로 데이터를 이동시키려면 데이터를 익스포트하고 원래의 테이블을 삭제한 후(선택사항), 다차원적으로 클러스터된(MDC) 테이블을 작성하여(CREATE TABLE문에서 ORGANIZE BY DIMENSIONS절 사용) MDC 테이블에 데이터를 로드하십시오.

SYSPROC.ALTOBJ라고 하는 ALTER TABLE 프로시저는 기존 테이블에서 MDC 테이블로 데이터 변환을 수행하는데 사용될 수 있습니다. 이 프로시저는 DB2 디자인 어드바이저에서 호출됩니다. 테이블 간의 데이터 변환에 필요한 시간은 오래 걸릴 수 있으며 테이블의 크기 및 변환이 필요한 데이터의 양에 따라 달라집니다.

ALTOBJ 프로시저는 테이블 변경을 수행할 때 다음을 수행합니다.

- 테이블의 모든 종속 오브젝트 삭제
- 테이블 이름 바꾸기
- 새 정의를 사용하여 테이블 작성
- 테이블의 모든 종속 오브젝트 다시 작성
- 테이블의 기존 데이터를 새 테이블에서 필요한 데이터로 변환. 즉, 이전 테이블에서 데이터를 선택하고 컬럼 함수를 사용하여 이전 데이터 유형에서 새 데이터 유형으로 변환할 수 있는 새 테이블로 해당 데이터를 로드합니다.

SMS 테이블 스페이스의 MDC 테이블

MDC 테이블을 SMS 테이블 스페이스에 저장하려는 경우에는 다중 파일 할당을 사용해야 합니다. (다중 페이지 파일 할당은 버전 8.2 이상에서 새로 작성된 데이터베이스에 대한 디폴트값입니다.) 이 이유는 MDC 테이블은 항상 전체 Extent에 의해 확장되

며 이러한 Extent의 모든 페이지가 물리적으로 연속된다는 사실이 중요하기 때문입니다. 따라서 다중 페이지 파일 할당을 사용하지 않는 것이 스페이스 측면에서 아무런 이점이 없습니다. 더구나 다중 페이지 파일 할당을 사용하는 경우에는 각 Extent의 페이지가 물리적으로 연속될 가능성이 매우 증대됩니다.

DB2 디자인 어드바이저의 MDC 어드바이저 기능

DB2 디자인 어드바이저(db2advis)에는 MDC 기능이 있습니다. 이 기능은 MDC 테이블에서 사용할 클러스터링 차원을 권장하며, 여기에는 워크로드 성능을 개선하기 위해 기본 컬럼에서의 개략화가 포함됩니다. *개략화(coarsification)*는 클러스터링 차원의 카디널리티(개별 값의 수)를 줄이는 작업의 수학적 표현을 의미합니다. 개략화의 일반적인 예로는 일, 주, 월 또는 분기로 개략화가 이루어질 수 있는 날짜를 들 수 있습니다.

DB2 디자인 어드바이저의 MDC 기능을 사용하려면 데이터베이스에 최소 여러 개의 데이터 Extent가 있어야 합니다. DB2 디자인 어드바이저는 데이터를 사용하여 데이터 밀도 및 카디널리티를 표시합니다.

데이터베이스 테이블에 데이터가 없는 경우, DB2 디자인 어드바이저는 MDC를 권장하지 않습니다. 데이터베이스에 빈 테이블이 있지만 채워진 데이터베이스를 의미하는 통계 모형이 있는 경우에도 마찬가지입니다.

권장사항에는 차원의 개략화를 정의하는 잠재적, 생성된 컬럼의 식별이 포함됩니다. 권장사항에는 가능한 블록 크기가 포함되어 있지 않습니다. 테이블 스페이스의 Extent 크기는 MDC 테이블에 대한 권장사항을 작성할 때 사용됩니다. 권장 MDC 테이블은 기존 테이블과 동일한 테이블 스페이스에서 작성되므로 동일한 Extent 크기를 가진다고 가정합니다. MDC 차원에 대한 권장사항은 테이블 스페이스의 Extent 크기에 따라 변경될 수 있습니다. 왜냐하면 Extent 크기가 블록이나 셀에 적합한 레코드의 수에 영향을 주기 때문입니다. 이는 셀의 밀도에 직접적으로 영향을 줍니다.

테이블에 대해 단일 또는 다중 차원을 권장할 수 있지만 단일 컬럼 차원(복합 컬럼 차원이 아닌 차원)만 고려됩니다. MDC 기능은 결과적인 MDC 솔루션에서 셀의 카디널리티(cardinality)를 줄이는 것을 목적으로 하며, 지원되는 대부분의 데이터 유형에 대해 개략화를 권장합니다. 데이터 유형 예외에는 CHAR, VARCHAR, GRAPHIC 및 VARGRAPH 데이터 유형이 포함됩니다. 지원되는 모든 데이터 유형은 INTEGER로 캐스트되며 생성된 표현식을 통해 개략화됩니다.

DB2 디자인 어드바이저의 MDC 기능을 사용하는 목적은 성능을 향상시키는 MDC 솔루션을 선택하기 위함입니다. 두 번째 목적은 데이터베이스의 스토리지 확장을 최적 레벨로 제한되도록 유지하는 것입니다. 통계적인 메소드가 각 테이블에서 최대 스토리지 확장을 결정하는데 사용됩니다.

어드바이저에서 수행되는 분석 조작에는 블록 인덱스 액세스의 이점은 물론 테이블의 차원에 대한 삽입, 갱신, 삭제 조작에 미치는 MDC의 영향도 포함됩니다. 테이블에서

의 이러한 조치로 인해 레코드가 셀 간에 이동될 수도 있습니다. 또한 분석 조작은 특정 MDC차원에서의 데이터 구성에 따른 임의의 테이블 확장이 잠재적으로 성능에 미치는 영향을 모델링합니다.

MDC 기능은 db2advis 유틸리티에서 -m <advise type> 플래그를 사용하여 실행합니다. 『C』 권장 유형은 다차원적으로 클러스터된 테이블을 나타내는데 사용됩니다. 권장 유형은 『I』(인덱스용), 『M』(구체화된 쿼리 테이블용), 『C』(MDC용) 및 『P』(파티션된 데이터베이스 환경용)입니다. 권장 유형은 서로간에 조합해서 사용이 가능합니다.

주: DB2 디자인 어드바이저는 크기가 12 Extent 미만인 테이블은 탐색하지 않습니다.

어드바이저는 권장사항을 제안할 때 MQT 및 일반 기본 테이블 모두를 분석합니다.

MDC 기능의 출력에는 다음이 포함됩니다.

- MDC 솔루션에 나타나는 개략화된 차원의 각 테이블에 대해 생성된 컬럼 표현식
- 각 테이블에 대해 권장되는 ORGANIZE BY절

권장사항은 Explain 기능의 일부인 ADVISE 테이블 및 표준 출력으로 보고됩니다.

MDC 테이블 및 파티션된 데이터베이스 환경

다차원 클러스터링을 파티션된 데이터베이스 환경과 함께 사용할 수 있습니다. 실제로 MDC는 파티션된 데이터베이스 환경을 보완할 수 있습니다. 파티션된 데이터베이스 환경은 다음 목적을 위해 다수의 물리적 또는 논리적 노드에 테이블의 데이터를 분산하는데 사용됩니다.

- 다중 시스템을 사용하여 처리 요청을 병렬로 늘리기
- 단일 데이터베이스 파티션의 한계를 넘어 테이블의 물리적 크기 늘리기
- 데이터베이스 확장성 개선

테이블을 분배하는 이유는 테이블이 MDC 테이블인지 또는 일반 테이블인지 여부와 무관합니다. 예를 들어, 분산 키를 구성하기 위해 컬럼을 선택하는 규칙은 동일합니다. MDC 테이블에 대한 분산 키에는 컬럼이 테이블의 차원의 일부를 구성하는지 여부와 상관없이 모든 컬럼이 포함될 수 있습니다.

분산 키가 테이블의 차원과 동일하면 각 데이터베이스 파티션에는 테이블의 다른 부분이 포함됩니다. 예를 들어, 예제 MDC 테이블이 두 데이터베이스 파티션에서 색상별로 파티션되면 Color 컬럼이 데이터 분할을 위해 사용됩니다. 그 결과 빨간색 및 파란색 조각이 한 파티션에 있고 노란색 조각이 다른 파티션에 있을 수 있습니다. 분산 키가 테이블의 차원과 동일하지 않으면 각 데이터베이스 파티션은 각 조각에서 데이터의 서브세트를 갖습니다. 차원을 선택하고 셀 점유율을 계산 할 때(『셀 밀도』 절 참조), 일반적으로 셀당 전체 데이터 양은 모든 데이터를 모아 이를 파티션 수로 나누어서 결정된다는 점을 참고하십시오.

다중 차원이 있는 MDC 테이블

특정 술어가 쿼리에서 많이 사용될 것 같으면 ORGANIZE BY DIMENSIONS절을 사용하여, 관련된 컬럼에서 테이블을 클러스터할 수 있습니다.

예 1

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

예 1의 테이블은 하나의 논리적 큐브를 형성하는 3개의 원시(native) 컬럼 내의 값에 클러스터됩니다(즉, 차원이 3개임). 해당 조각이나 셀의 블록만 관련된 관계 연산자에 의해 처리되도록 하나 이상의 차원에 대한 쿼리 처리 중 테이블을 논리적으로 조각으로 나눌 수 있습니다. 블록 *크기*(페이지 수)는 테이블의 Extent *크기*가 된다는 점을 참고하십시오.

둘 이상의 컬럼을 기반으로 하는 차원이 있는 MDC 테이블

각 차원은 하나 이상의 컬럼으로 구성될 수 있습니다. 예를 들어, 두 컬럼이 포함된 차원에 클러스터되는 테이블을 작성할 수 있습니다.

예 2

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

예 2에서, 테이블은 두 개의 차원인 c1과 (c3, c4)에서 클러스터됩니다. 따라서 쿼리를 처리할 때 c1 차원이나 복합 (c3, c4) 차원에서 테이블을 논리적 조각으로 나눌 수 있습니다. 테이블의 블록 수는 예 1의 테이블과 동일하지만 차원 블록 인덱스는 하나 적습니다. 예 1에는 컬럼 c1, c3 및 c4에 대해 각각 하나씩 세 개의 차원 블록 인덱스가 있습니다. 예 2에는 컬럼 c1에 대해 하나와 컬럼 c3 및 c4에 대해 하나인 두 개의 차원 블록 인덱스가 있습니다. 이 두 예의 차이점은 예 1에서는 c4와만 관련된 쿼리가 c4에 대한 차원 블록 인덱스를 사용하여 관련 데이터의 블록에 빠르게 직접 액세스할 수 있다는 점입니다. 예 2에서는 c4가 차원 블록 인덱스의 두 번째로 중요한 부분이므로 c4만 관련된 쿼리에는 추가 처리가 필요합니다. 그러나 예 2에서는 유지보수 및 저장할 블록 인덱스가 하나 줄게 됩니다.

DB2 디자인 어드바이저는 둘 이상의 컬럼이 포함된 차원을 권장하지 않습니다.

차원으로서의 컬럼 표현식이 있는 MDC 테이블

컬럼 표현식은 차원 클러스터링에도 사용할 수 있습니다. 컬럼 표현식에 대해 클러스터하는 기능은 지리적 위치 또는 지역으로 주소를 롤업하거나 날짜를 주, 월 또는 연도로 롤업하는 것과 같이 차원을 보다 낮은 세분화도로 롤업하는 경우에 유용합니다. 이러한 방식으로 차원의 롤업을 구현하기 위해, 생성된 컬럼을 사용할 수 있습니다. 이러한 유

형의 컬럼 정의는 차원을 나타낼 수 있는 표현식을 사용하여 컬럼을 작성할 수 있게 합니다. 예 3에서 명령문은 하나의 기본 컬럼 및 두 개의 컬럼 표현식에 대해 클러스터된 테이블을 작성합니다.

예 3

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
      c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
            ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

예 3에서, 컬럼 c5는 컬럼 c3 및 c4를 기반으로 하는 표현식인 반면, 컬럼 c6은 컬럼 c1을 적절한 때에 보다 낮은 세분화도로 롤업합니다. 이 명령문은 컬럼 c2, c5 및 c6의 값에 따라 테이블을 클러스터합니다.

생성된 컬럼 차원에 대한 범위 쿼리

단수 컬럼 함수가 필요한 생성된 컬럼 차원의 범위 쿼리 표현식은 생성된 컬럼의 차원에 대한 범위 술어를 유도하기 위해 단순해야 합니다. 생성된 컬럼에 차원을 작성하는 경우, 기본 컬럼에 대한 쿼리는 한 가지 예외를 제외하고는 생성된 컬럼의 블록 인덱스를 사용하여 성능을 향상시킵니다. 차원 블록 인덱스에서 범위 스캔을 사용하는 기본 컬럼(예: 날짜)에 대한 범위 쿼리의 경우, CREATE TABLE문에서 컬럼을 생성하기 위해 사용되는 표현식은 단순해야 합니다. 컬럼 표현식에는 유효한 모든 표현식(사용자 정의 함수(UDF) 포함)이 포함될 수 있지만, 표현식이 단순하지 않으면 등호 또는 IN 술어가 기본 컬럼에 있을 때 이 술어만 블록 인덱스를 사용하여 쿼리를 충족시킬 수 있습니다.

예를 들어, 생성된 컬럼 'month'에서 차원을 갖는 MDC 테이블을 작성한다고 가정합시다(`month = INTEGER (date)/100`). 차원(month) 쿼리의 경우에는 블록 인덱스 스캔이 수행될 수 있습니다. 기본 컬럼(date) 쿼리의 경우에는 블록 인덱스 스캔을 사용하여 스캔할 블록을 좁힐 수도 있으며 그런 다음 'date'의 술어를 해당 블록에만 있는 행에 적용할 수 있습니다.

컴파일러는 블록 인덱스 스캔에서 사용할 추가 술어를 생성합니다. 예를 들어, 다음의 쿼리에서

```
SELECT * FROM MDCTABLE WHERE DATE > "1999-03-03" AND DATE < "2000-01-15"
```

컴파일러는 추가 술어 『month >= 199903』 및 『month <= 200001』을 생성하는데, 이는 차원 블록 인덱스 스캔에 대한 술어로 사용할 수 있습니다. 결과 블록을 스캔하는 중에 원래 술어는 블록의 행에 적용됩니다.

단순하지 않은 표현식에서는 해당 차원에 등호 술어만 적용할 수 있습니다. 단순하지 않은 함수의 좋은 예로는 예 3의 컬럼 c6에 대한 정의에서 볼 수 있는 MONTH( )가 있습니다. c1 컬럼이 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경

우, 이 함수는 1 - 12 범위 내의 정수 값을 리턴합니다. 함수의 결과가 결정적이더라도 실제로는 단계 함수(즉, 순환 패턴)의 결과와 비슷합니다.

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

이 예의 날짜가 계속해서 증가하더라도 MONTH(date)는 증가하지 않습니다. 보다 자세하게 말하자면 date1이 date2보다 클 때 MONTH(date1)이 MONTH(date2)보다 항상 크거나 같다고 보장할 수는 없습니다. 이 상태가 바로 단순성에 필요한 조건입니다. 이러한 비단순성은 허용되지만 기본 컬럼에 대한 범위 술어가 차원에 대한 범위 술어를 생성할 수 없다는 점에서 차원을 제한합니다. 그러나 예를 들어, `where month(c1) between 4 and 6`과 같이 표현식에 대한 범위 술어는 허용됩니다. 시작 키로 4를 사용하고 중지 키로 6을 사용하여 차원에 대한 인덱스를 평상시와 같이 사용할 수 있습니다.

이 함수를 단순하게 하려면 연도를 월의 상위 부분으로 포함시켜야 합니다. 이는 INTEGER 내장 함수의 확장을 제공하므로 날짜에 대한 단순한 표현식을 쉽게 정의할 수 있습니다. INTEGER(date)는 날짜의 정수 표현을 리턴하며 이 표현을 나누어 연도와 월의 정수 표현을 찾을 수 있습니다. 예를 들어, `INTEGER(date('2000/05/24'))`는 `20000524`를 리턴하므로 `INTEGER(date('2000/05/24'))/100 = 200005`입니다. INTEGER(date)/100은 단순 함수입니다.

마찬가지로 내장 함수 DECIMAL 및 BIGINT도 확장하여 사용할 수 있으므로 단순 함수를 파생시킬 수 있습니다. DECIMAL(timestamp)는 시간소인의 10진수 표현을 리턴하므로 이를 단순한 표현식에 사용하여 월, 일, 시, 분 등에 대한 증가 값을 파생시킬 수 있습니다. BIGINT(date)는 INTEGER(date)와 마찬가지로 날짜의 큰 정수 표현을 리턴합니다.

데이터베이스 관리 프로그램은 테이블에 대해 생성된 컬럼을 작성할 때나 차원 절의 표현식에서 차원을 작성할 때 가능한 한 표현식의 단순성을 판별합니다. DATENUM( ), DAYS( ), YEAR( )와 같은 특정 함수는 단순성을 유지하는 것으로 간주할 수 있습니다. 또한, 컬럼 및 상수의 나누기, 곱하기 또는 더하기와 같은 다양한 수학 표현식은 단순성을 유지합니다. DB2가 표현식이 단순성을 유지하지 않는 것으로 판별하거나 이를 판별할 수 없는 경우, 차원은 기본 컬럼에 등호 술어만 사용하도록 지원합니다.

MDC 테이블에 대한 로드 고려사항

정기적으로 데이터 웨어하우스에 데이터를 롤인하는 경우, MDC 테이블을 자신에게 유리하게 사용할 수 있습니다. MDC 테이블에서는 먼저 테이블에서 이전에 비운 블록을 로드에 재사용한 다음 테이블을 확장하고 나머지 데이터에 대해 새 블록을 추가합니다.

예를 들어, 월에 대한 모든 데이터와 같은 데이터 세트를 삭제한 후, 로드 유틸리티를 사용하여 다음 달의 데이터를 롤인하고 삭제(커미트됨) 이후 비워진 블록을 다시 사용할 수 있습니다. 또한 지연된 정리에 MDC 롤아웃 기능을 사용하도록 선택할 수 있습니다. 롤아웃(삭제에 해당함)이 커미트된 후 블록은 비어 있지 않으므로 아직 재사용할 수 없습니다. 레코드 ID(RID) 기반 인덱스를 유지보수하기 위해 백그라운드 프로세스가 호출됩니다. 유지보수가 완료되면 블록이 비워지고 재사용될 수 있습니다.

데이터를 MDC 테이블로 로드하는 경우, 입력 데이터는 정렬되거나 정렬되지 않을 수 있습니다. 정렬되지 않는 경우에는 다음 사항을 고려하십시오.

- *util_heap_sz* 구성 매개변수를 늘리십시오.

  MDC 테이블을 로드할 때 로드 유틸리티의 성능을 향상시키려면 *util_heap_sz* 데이터베이스 구성 매개변수 값을 늘려야 합니다. mdc-load 알고리즘의 경우 유틸리티에서 사용 가능한 추가 메모리가 있으면 성능이 더 향상됩니다. 그러면 로드 단계 중 수행된 데이터 클러스터링 동안 디스크 입출력이 줄어듭니다. LOAD 명령을 사용하여 여러 MDC 테이블을 동시에 로드하는 경우 적절히 *util_heap_sz*를 늘려야 합니다.
- LOAD 명령의 DATA BUFFER절을 사용하여 제공된 값을 늘리십시오.

  이 값을 늘리면 단일 로드 요청에 영향을 줍니다. 유틸리티 힙 *크기*는 다중 동시 로드 요청의 가능성을 수용할 정도로 충분히 커야 합니다. LOAD 명령의 DATA BUFFER 옵션이 지정되면 값도 증가해야 합니다.
- 버퍼 풀에 사용되는 페이지 *크기*가 임시 테이블 스페이스에 대한 최대 페이지 *크기*와 동일하도록 하십시오.

  로드 단계 중 블록 맵 유지보수를 위한 추가 로깅이 수행됩니다. 할당된 Extent당 약 2개의 추가 로그 레코드가 있습니다. 좋은 성능을 유지하려면 *logbufsz* 데이터베이스 구성 매개변수를 이를 고려한 값으로 설정해야 합니다.

다음 제한사항은 다차원적으로 클러스터된(MDC) 테이블로 데이터를 로드하는 경우 적용됩니다.

- LOAD 명령의 SAVECOUNT 옵션은 지원되지 않습니다.
- 이러한 테이블은 고유한 여유 공간을 관리하므로 `totalfreespace` 파일 유형 수정자는 지원되지 않습니다.
- MDC 테이블에 대해 `anyorder` 파일 유형 수정자가 필요합니다. `anyorder` 수정자 없이 MDC 테이블로 로드가 실행되면 유틸리티에서 명시적으로 사용 가능해집니다.

MDC 테이블에서 LOAD 명령을 사용하는 경우 고유 제한조건 위반은 다음과 같이 처리됩니다.

- 로드 조작 전에 테이블에 고유 키가 포함되고 중복 레코드를 테이블로 로드한 경우 원래 레코드는 남아 있으며 새 레코드는 삭제 단계 중에 삭제됩니다.
- 로드 조작 전에 테이블에 고유 키가 없고 고유 키 및 중복 레코드를 테이블로 로드한 경우 고유 키를 포함하는 레코드 중 하나만 로드되고 나머지는 삭제 단계 중에 삭제됩니다.

  주: 로드되는 레코드와 삭제되는 레코드를 판별하는 명시적인 기술은 없습니다.

로드는 블록 바운더리에서 시작되므로, 이는 새 셀에 속하는 데이터에 사용하거나 테이블을 처음으로 채우는 경우에 가장 적합합니다.

모든 MDC 테이블에는 블록 인덱스가 있으므로 MDC 로드 조작은 항상 빌드 단계를 포함합니다.

MDC 테이블에 대한 로깅 고려사항

RID 인덱스에 의해 이전에 또는 다른 때에 인덱스화된 컬럼이 현재 차원이므로 블록 인덱스를 통해 인덱스화되는 경우, 인덱스 유지보수 및 로그는 상당히 줄어듭니다.

전체 블록의 최종 레코드가 삭제된 경우에만 데이터베이스 관리 프로그램은 블록 인덱스에서 BID를 제거하고 이 인덱스 조작을 로그해야 합니다. 마찬가지로 레코드가 새 블록에 삽입될 때만(논리적 셀의 첫 번째 레코드이거나 현재 가득 찬 블록의 논리적 셀에 삽입하는 경우) 데이터베이스 관리 프로그램은 블록 인덱스에 BID를 삽입하고 이 조작을 로그해야 합니다. 블록이 2 - 256 페이지의 레코드일 수 있으므로 이 블록 인덱스 유지보수 및 로그는 상대적으로 적습니다. 테이블과 RID 인덱스에 대한 삽입 및 삭제는 계속 로그됩니다. 롤 아웃 삭제의 경우, 삭제된 레코드는 로그되지 않습니다. 대신에, 페이지의 일부를 재형식화하여 레코드를 포함하는 페이지를 비어 있는 것처럼 보이게 합니다. 다시 형식화된 부분의 변경사항은 로그되지만 레코드 자체는 로그되지 않습니다.

MDC 테이블에 대한 블록 인덱스 고려사항

MDC 테이블에 대한 차원을 정의하는 경우에는 차원 블록 인덱스가 작성됩니다. 또한 다중 차원이 정의된 경우에는 복합 블록 인덱스도 작성될 수 있습니다. 그러나 DC 테이블에 대해 하나의 차원만 정의한 경우, DB2는 차원 블록 인덱스와 복합 블록 인덱스의 역할을 모두 수행할 하나의 블록 인덱스만 작성합니다.

마찬가지로 컬럼 A 등(컬럼 A, 컬럼 B)에 차원이 있는 MDC 테이블을 작성하는 경우, DB2는 컬럼 A에 차원 블록 인덱스를 작성하고 컬럼 A, 컬럼 B에 차원 블록 인덱스를 작성합니다. 복합 블록 인덱스는 테이블에 있는 모든 차원의 블록 인덱스이므로 컬럼 A, 컬럼 B의 차원 블록 인덱스도 복합 블록 인덱스의 역할을 수행합니다.

복합 블록 인덱스는 특정 차원 값이 있는 테이블의 데이터에 액세스하기 위한 쿼리 처리에도 사용됩니다. 복합 블록 인덱스에 있는 주요 파트의 순서에 따라 쿼리 처리에 대한 사용 또는 적용 가능성이 변경될 수 있습니다. 주요 파트의 순서는 MDC 테이블을 작성할 때 사용된 전체 ORGANIZE BY DIMENSIONS절에 있는 컬럼의 순서에 따라 결정됩니다. 예를 들어, 다음 명령문을 사용하여 테이블을 작성하면,

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
      ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

컬럼(c4,c3,c1,c2)에 복합 블록 인덱스가 작성됩니다. c1이 차원 절에 두 번 지정되었어도 먼저 발견된 것을 우선적으로 하여 복합 블록 인덱스의 주요 파트로 한 번만 사용된다는 점을 참고하십시오. 복합 블록 인덱스의 키 부분 순서는 삽입 처리시에는 달라지지 않으나 쿼리 처리시에는 달라질 수도 있습니다. 따라서 컬럼 순서(c1,c2,c3,c4)가 복합 블록 인덱스보다 바람직한 경우, 다음 명령문을 사용하여 테이블을 작성해야 합니다.

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

MDC 테이블에 대한 블록 인덱스

이 주제에서는 블록 인덱스를 사용하여 MDC 테이블에서 레코드를 구성하는 방법을 표시합니다.

MDC 테이블 65 페이지의 그림 12는 동일『지역』및『연도』값을 지닌 레코드가 별도의 블록 또는 Extent로 그룹화되도록 물리적으로 구성되어 있습니다. Extent는 디스크에서 연속된 페이지 세트이므로 이러한 레코드의 그룹은 물리적으로 연속된 데이터 페이지에 클러스터되어 있습니다. 각각의 테이블 페이지는 정확히 하나의 블록에 속하며 모든 블록은 *크기*가 동일합니다(즉, 동일한 페이지 수). 블록의 *크기*는 테이블 스페이스의 Extent *크기*와 동일하므로, 블록 바운더리는 Extent 바운더리와 함께 정렬됩니다. 이 경우에는 두 개의 블록 인덱스가 작성됩니다. 하나는『지역』차원에 대한 것이며 다른 하나는『연도』차원에 대한 것입니다. 이러한 블록 인덱스에는 테이블의 블록에 대한 포인터만 포함되어 있습니다.『지역』이『동부』와 동일한 모든 레코드에 대해『지역』블록 인덱스를 스캔하면 규정에 맞는 두 개의 블록이 발견됩니다.『지역』이『동부』와 동일한 모든 레코드 또는 해당 레코드만 이 두 블록에서 발견되며 연속된 페이지 또는 Extent의 두 세트에서 클러스터됩니다. 이와 동시에, 완전히 독립적으로 1999 및 2000 사이의 레코드에 대해『연도』인덱스를 스캔하면 규정에 맞는 세 개의 블록이 발견됩니다. 이러한 각각의 세 블록에 대한 데이터 스캔은 모든 레코드 또는 1999 및 2000 사이의 해당 레코드만 리턴하며 각각의 블록 내의 순차 페이지에서 클러스터된 이 레코드들을 찾습니다.

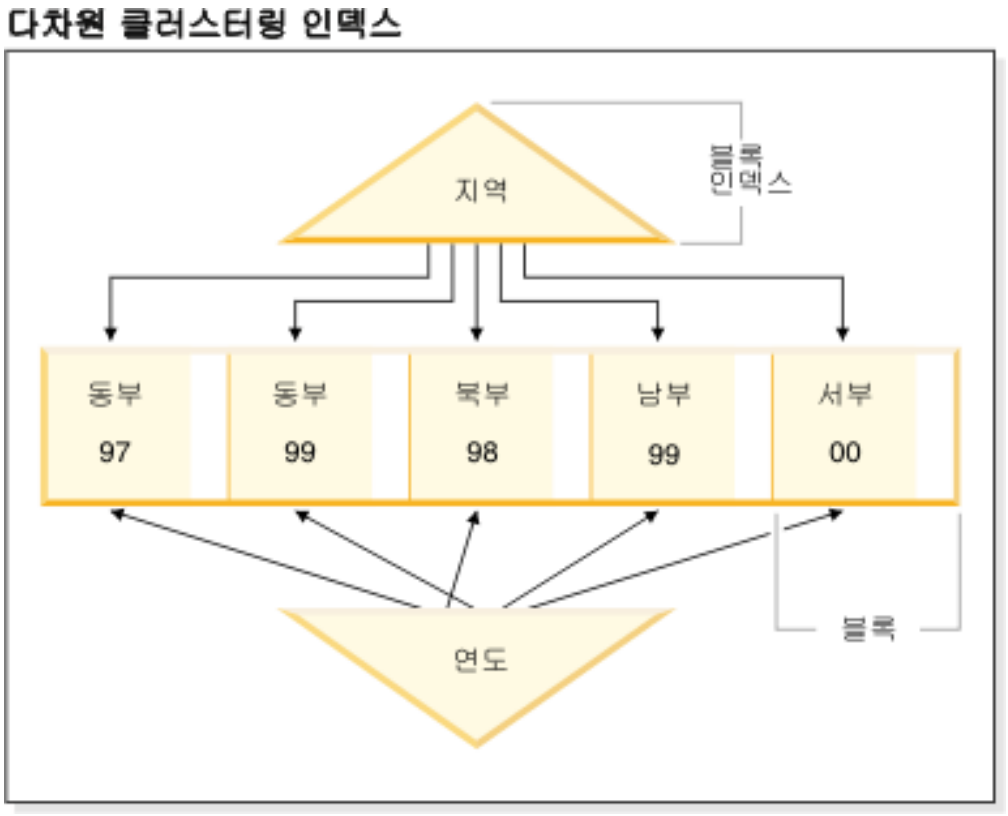


그림 12. 다차원적으로 클러스터된 테이블

이러한 클러스터링 개선사항과 함께 MDC 테이블은 다음과 같은 이점을 제공합니다.

- 블록 인덱스는 레코드 기반 인덱스에 비해 *크기*가 매우 작으므로 조사 및 스캔이 훨씬 빠릅니다.
- 블록 인덱스 및 대응되는 데이터 구성은 세부적인 『데이터베이스 파티션 제거』 또는 선택적 테이블 액세스를 허용합니다.
- 블록 인덱스를 사용하는 쿼리에는 감소된 인덱스 *크기*, 블록의 최적화된 프리페치, 대응되는 데이터의 보증된 클러스터링과 같은 이점이 있습니다.
- 일부 쿼리의 경우에는 감소된 잠금 및 술어 평가가 가능합니다.
- 블록 인덱스는 로그 및 유지보수에 대해 이와 연관된 오버헤드가 훨씬 적습니다. 이는 블록에 첫 번째 레코드를 추가하거나 블록에서 마지막 레코드를 제거하는 경우에만 갱신이 필요하기 때문입니다.
- 롤인된 데이터는 이전에 롤아웃된 데이터에 의해 남겨진 연속 스페이스를 재사용합니다.

주: 단일 차원으로만 정의된 경우라도 MDC 테이블은 이러한 MDC 속성으로부터 이점을 얻을 수 있으며, 클러스터링 인덱스가 있는 일반 테이블에 대한 실용적인 대안이 될 수 있습니다. 이러한 결정은 워크로드를 구성하는 쿼리, 테이블에 있는 데이터의 특

성 및 분산을 포함하여 다수의 인수를 기반으로 해야 합니다. 45 페이지의 『MDC 테이블 차원 선택』 및 56 페이지의 『MDC 테이블을 작성 시 고려사항』의 내용을 참조하십시오.

테이블을 작성할 때 데이터를 클러스터할 차원으로 하나 이상의 키를 지정할 수 있습니다. 이들 각 차원은 인덱스 키와 마찬가지로 하나 이상의 컬럼으로 구성될 수 있습니다. *차원 블록 인덱스*는 지정된 차원마다 자동으로 작성되며 옵티마이저가 각 차원을 따라 데이터에 빠르고 효율적으로 액세스하는데 사용됩니다. 모든 차원에 대해 모든 컬럼이 포함된 *복합 블록 인덱스*도 자동으로 작성되어, 삽입 및 갱신 활동 중에 데이터의 클러스터링을 유지보수하는데 사용됩니다. 복합 블록 인덱스는 단일 차원에 아직 모든 차원 키 컬럼이 포함되지 않은 경우에만 작성됩니다. 복합 블록 인덱스도 컬럼 차원의 서브세트에서 또는 전체에서 값을 충족하는 데이터에 효율적으로 액세스하기 위해 옵티마이저에 의해 선택될 수 있습니다.

주: 쿼리 처리 중 이 인덱스의 유용성은 해당 주요 파트의 키 순서에 의존합니다. CREATE TABLE 명령문의 ORGANIZE BY절에 지정된 차원을 구문 분석할 때 파트의 순서는 구문 분석기에서 발견하는 컬럼의 순서에 따라 결정됩니다. 자세한 정보는 63 페이지의 『MDC 테이블에 대한 블록 인덱스 고려사항』의 내용을 참조하십시오.

블록 인덱스는 구조적으로 일반 인덱스와 동일하지만 레코드 대신 블록을 가리킨다는 점만 다릅니다. 블록 인덱스의 *크기*는 일반 인덱스보다 작은데, 블록 *크기*를 페이지의 평균 레코드 수와 곱한 인수만큼 작습니다. 블록의 페이지 *크기*는 테이블 스페이스의 Extent *크기*와 동일합니다. 범위는 2 - 256 페이지입니다. 페이지 *크기*는 4KB, 8KB, 16KB 또는 32KB입니다.

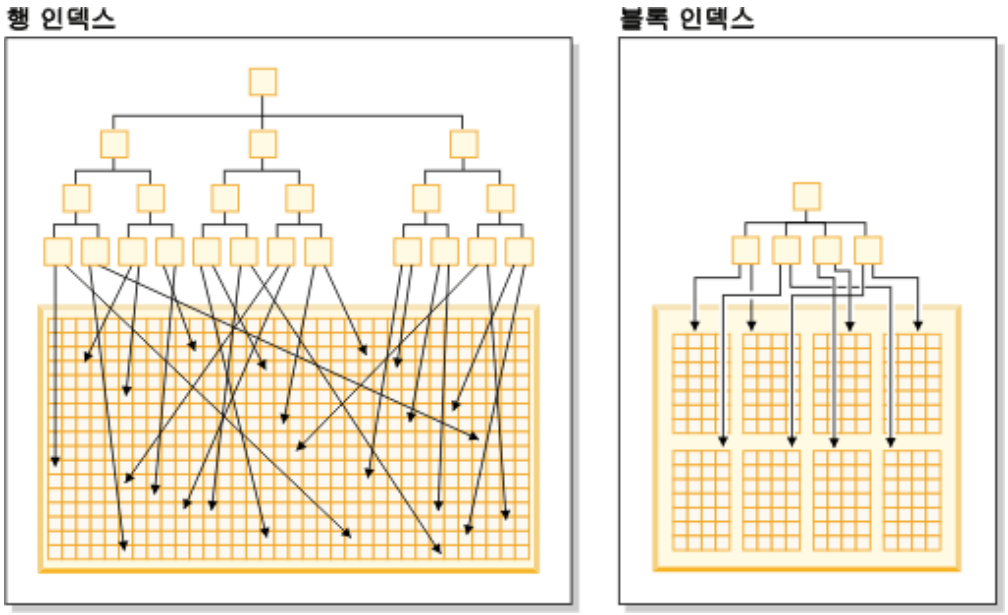


그림 13. 행 인덱스와 블록 인덱스의 차이점

66 페이지의 그림 13에 표시된 대로, 블록 인덱스에는 각 행에 대한 단일 항목과 비교하여 각 블록에 대한 단일 인덱스 항목이 있습니다. 결과적으로 블록 인덱스는 디스크 사용량을 많이 줄일 수 있으며 매우 빠른 데이터 액세스를 제공합니다.

MDC 테이블에서는 차원 값의 고유한 모든 조합이 하나의 논리적 *셀*을 형성하며 이 셀은 실제로 둘 이상의 페이지 블록으로 구성될 수 있습니다. 논리적 셀은 해당 논리적 셀의 차원 값을 갖는 레코드를 저장하기 위해 이와 연관된 양의 블록만 갖습니다. 특정 논리적 셀의 차원 값을 갖는 테이블에 레코드가 없으면 해당 논리적 셀에 대해 블록이 할당되지 않습니다. 특정 차원 키 값을 가진 데이터를 포함하는 블록 세트를 *조각*이라고 합니다.

시나리오: 다차원적으로 클러스터된(MDC) 테이블

MDC에 대한 작업 방법 시나리오로, 전국 소매업자들에 대한 판매 데이터를 기록하는 『Sales』라고 하는 MDC 테이블이 있다고 가정합시다. 이 테이블은 『YearAndMonth』 및 『Region』 차원을 따라 클러스터됩니다. 테이블의 레코드는 블록에 저장되는데, 블록에는 Extent를 채우는데 충분한 연속된 페이지가 있습니다.

68 페이지의 그림 14과 같이 블록은 직사각형으로 표시되고 테이블에 있는 할당된 Extent의 논리적인 순서에 따라 번호가 매겨집니다. 다이어그램의 격자는 이들 블록의 논리적 데이터베이스 파티션을 나타내며 각 정사각형은 논리적 셀을 나타냅니다. 격자의 컬럼이나 행은 특정 차원의 분할을 나타냅니다. 예를 들어, 『Region』 컬럼에 'South-central' 값을 포함하는 모든 레코드는 격자의 'South-central' 컬럼에 의해 정의된 조각에 포함된 블록에서 실제로 이 조각의 각 블록에는 『Region』 필드에 'South-central'이 있는 레코드만 포함됩니다. 따라서 블록의 『Region』 필드에 'South-central'이 있는 레코드가 포함된 경우에만 격자의 컬럼이나 이 조각에 블록이 포함됩니다.

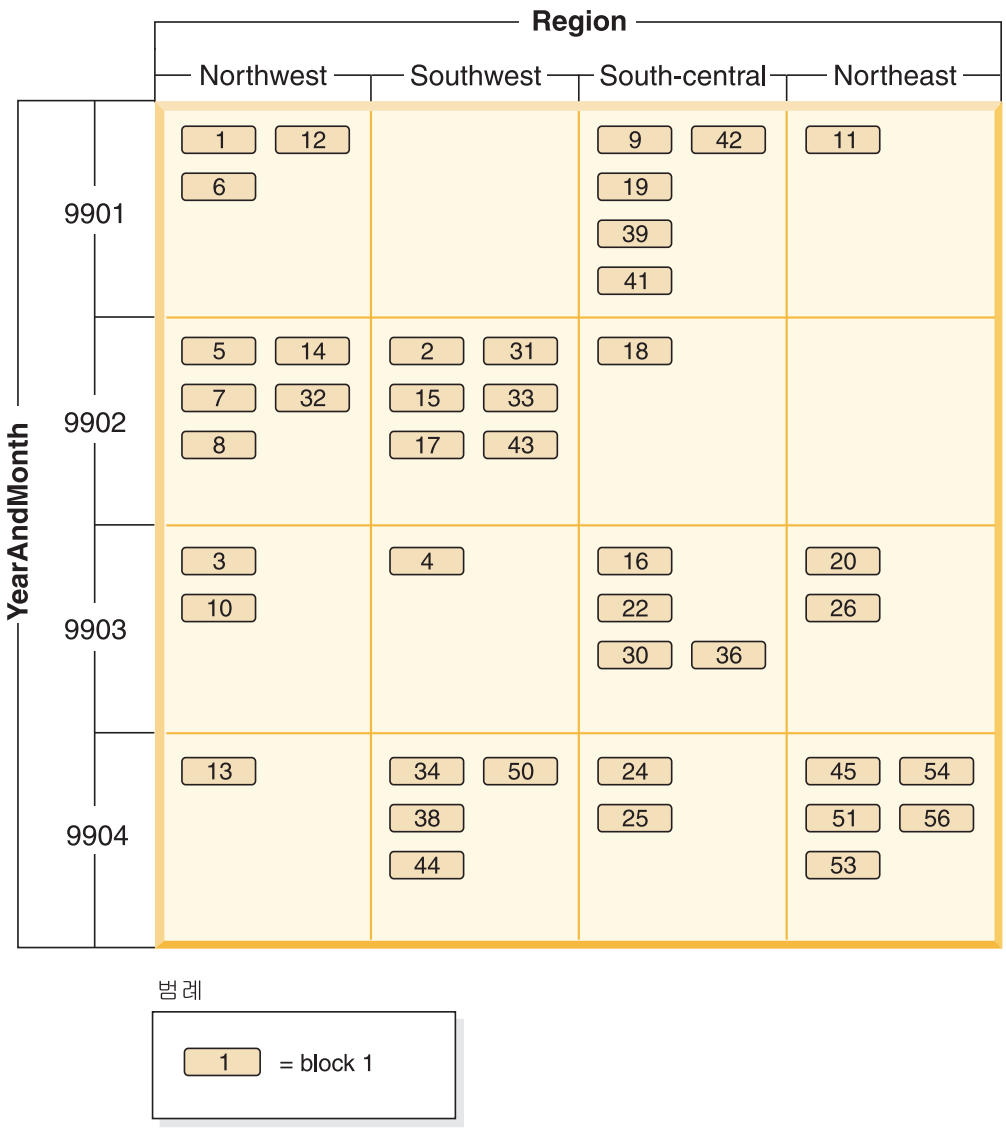


그림 14. 'Region' 및 'YearAndMonth' 차원을 포함하는 Sales라고 하는 다차원 테이블

조각을 구성하는 블록 또는 특정 차원 키 값이 있는 모든 레코드가 포함된 블록을 동등하게 판별하기 위해 테이블을 작성할 때 차원마다 차원 블록 인덱스가 자동으로 작성됩니다.

69 페이지의 그림 15와 같이 차원 블록 인덱스가 『YearAndMonth』 차원에 작성되며 『Region』 차원에는 다른 차원 블록 인덱스가 작성됩니다. 각 차원 블록 인덱스는 리프 레벨에서 키가 레코드 ID(RID) 대신 블록 ID(BID)를 가리킨다는 점을 제외하고는 기존의 RID 인덱스와 동일한 방식으로 구조화됩니다. RID는 물리적 페이지 번호 및 슬롯 번호(레코드가 있는 페이지의 슬롯)에 의해 테이블에서 레코드의 위치를 식별합니다. BID는 더미 슬롯(0) 및 해당 Extent의 첫 번째 페이지의 물리적 페이지 번호에 의해 블록을 표현합니다. 블록의 모든 페이지가 해당 항목에서 시작하여 물리적으로 연속되며 사용자는 블록의 크기를 알고 있으므로 이 BID를 사용하여 블록의 모든 레코드를 찾을 수 있습니다.

조각 또는 블록 세트가 차원 특정 키 값을 갖는 모든 레코드가 있는 페이지를 포함하는 경우, 연관된 차원 블록 인덱스에 해당 키 값에 대한 BID 목록으로 표시됩니다.

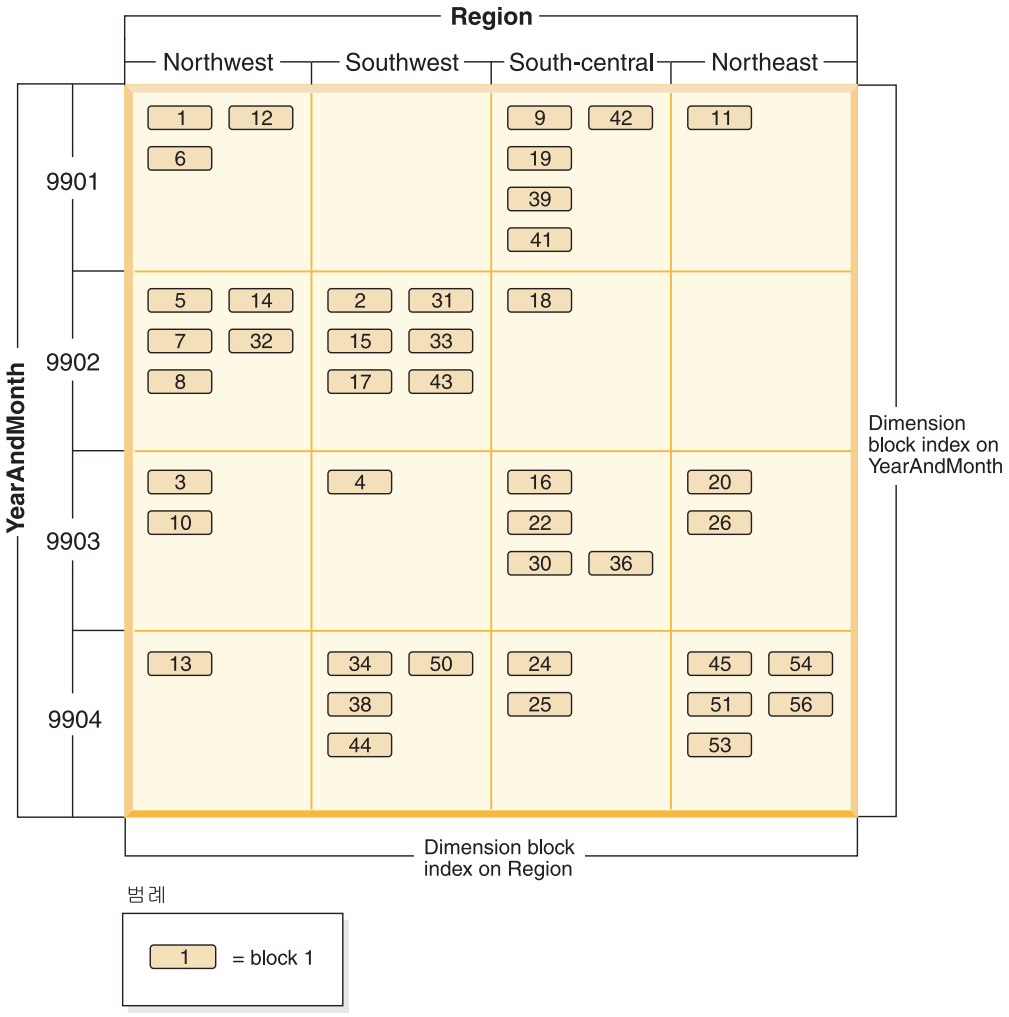


그림 15. 차원 블록 인덱스를 표시하는 'Region' 및 'YearAndMonth' 차원이 있는 Sales 테이블

70 페이지의 그림 16은 『Region』의 차원 블록 인덱스 키가 표시되는 방식을 나타냅니다. 키는 'South-central'이라고 하는 키 값과 BID 목록으로 구성됩니다. 각 BID는 블록 위치를 포함합니다. 70 페이지의 그림 16에 나열된 블록 번호는 Sales 테이블의 격자에 있는 'South-central' 조각에서 찾을 수 있는 것과 동일합니다(68 페이지의 그림 14 참조).

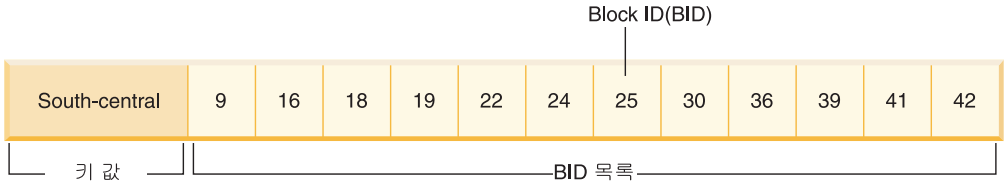


그림 16. 'Region'의 차원 블록 인덱스 키

마찬가지로 『YearAndMonth』 차원에 대해 '9902'를 가지고 있는 모든 레코드가 포함된 블록 목록을 찾기 위해, 그림 17에 표시된 대로 『YearAndMonth』 차원 블록 인덱스에서 이 값을 찾습니다.

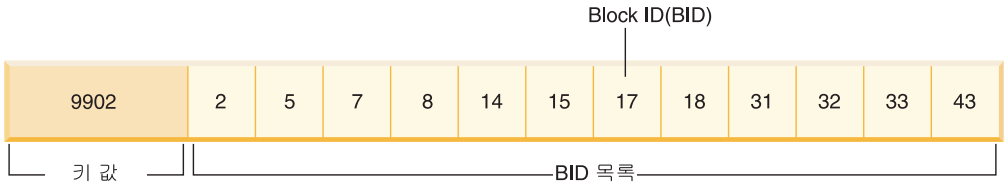


그림 17. 'YearAndMonth'의 차원 블록 인덱스 키

MDC 테이블에 대한 블록 인덱스 및 쿼리 성능

MDC 테이블의 임의의 블록 인덱스에서 수행되는 스캔은 클러스터된 데이터 액세스를 제공합니다. 왜냐하면 지정된 차원 값을 갖는 데이터를 포함하도록 보장된 테이블의 순차 페이지의 세트와 각각의 BID가 대응되기 때문입니다. 더구나 차원 또는 조각은 임의의 다른 차원 또는 조각의 클러스터 인수를 조정하지 않고 자체 블록 인덱스를 통해 서로간에 독립적으로 액세스될 수 있습니다. 이는 다차원 클러스터링의 다차원성을 제공합니다.

블록 인덱스 액세스를 사용하는 쿼리는 성능을 향상시키는 여러 가지의 요인들의 이점을 활용할 수 있습니다. 첫 번째, 블록 인덱스는 일반 인덱스보다 *크기*가 훨씬 작으므로 블록 인덱스 스캔이 매우 효율적입니다. 두 번째, 데이터 페이지의 프리페치는 블록 인덱스가 사용될 때 순차적 발견에 의존하지 않습니다. DB2는 인덱스를 미리 보고 큰 블록 입출력을 사용하여 블록의 데이터 페이지를 메모리에 프리페치하고 데이터 페이지가 테이블에서 액세스될 때 입출력을 초래하지 않도록 보장합니다. 세 번째, 테이블의 데이터는 순차 페이지에서 클러스터되며 I/O를 최적화하고 테이블의 선택된 부분으로 결과 세트를 로컬화합니다. 네 번째, 블록 기반의 버퍼 풀이 Extent *크기*인 블록 *크기*로 사용되는 경우에는 MDC 블록이 디스크의 순차 페이지에서 메모리의 순차 페이지로 프리페치되며 이에 따라 성능 측면에서 클러스터링의 효과가 더욱 증대됩니다. 마지막으로 각 블록의 레코드는 자체 데이터 페이지의 최소 관련 스캔을 사용하여 검색되며, 이는 종종 RID 기반 검색을 통하는 것보다 신속한 데이터 스캔 방법입니다.

쿼리는 블록 인덱스를 사용하여 특정 차원 값 또는 값 범위를 갖는 테이블 부분을 축소할 수 있습니다. 이는 세밀한 형태의『데이터베이스 파티션 제거』즉, 블록 제거를 제공합니다. 이는 테이블에 대해 보다 정확하게 동시적으로 변환될 수 있습니다. 왜냐하면 다른 쿼리, 로드, 삽입, 갱신 및 삭제가 이 쿼리의 데이터 세트와 상호작용 하지 않고도 테이블의 다른 블록에 액세스할 수 있기 때문입니다.

Sales 테이블이 3개의 차원으로 클러스터되는 경우, 개별 차원 블록 인덱스를 사용하여, 테이블의 모든 차원 서브세트에 대한 쿼리를 충족시키는 레코드가 포함된 블록 세트를 찾을 수 있습니다. 테이블에『연도/달』,『지역』및『제품』차원이 있는 경우, 이 테이블은 그림 18에 표시된 대로 논리적 큐브로 간주할 수 있습니다.

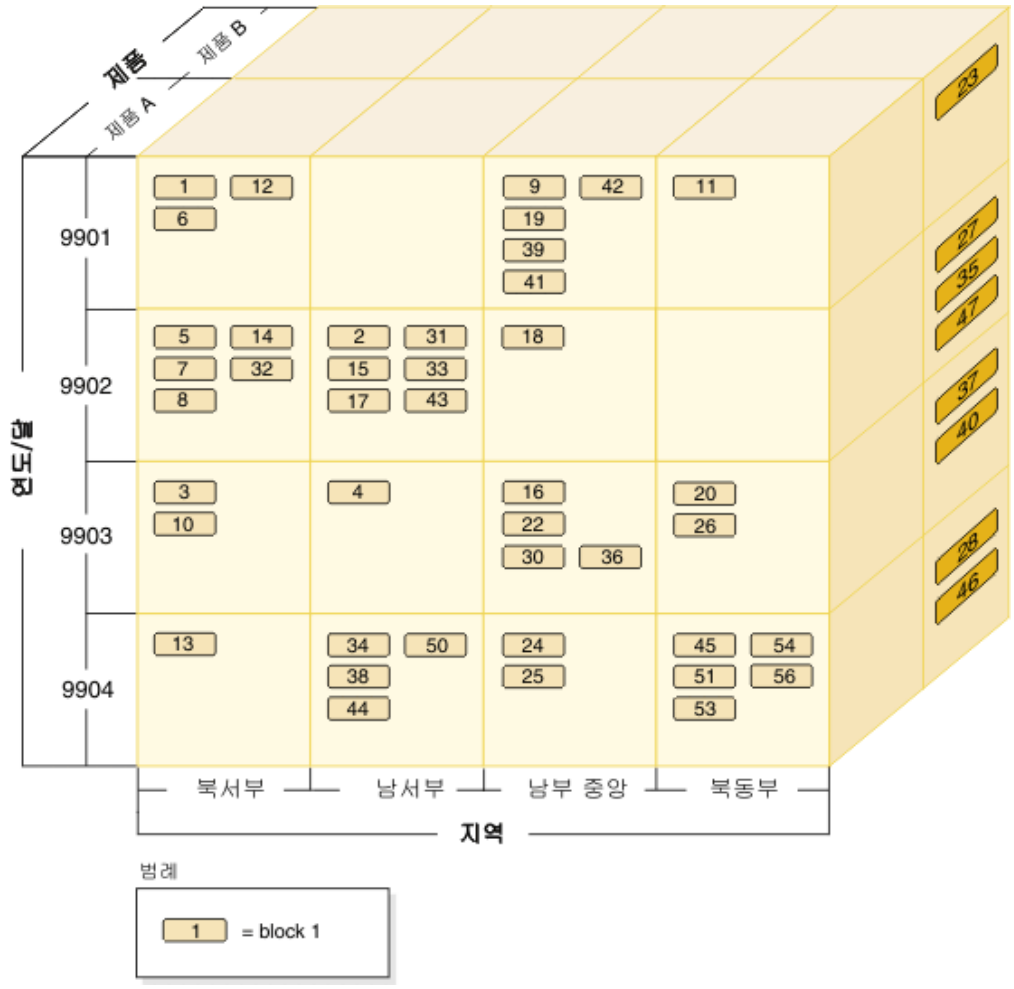


그림 18. '지역', '연도/달' 및 '제품' 차원이 있는 다차원 테이블

그림 18에 표시된 MDC 테이블에 대해서는 4개의 블록 인덱스가 작성됩니다.『연도/달』,『지역』및『제품』의 각 개별 차원에 대한 블록 인덱스와 이들 모든 차원 컬럼을 키로 사용하는 다른 블록 인덱스가 작성됩니다.『제품』이『제품 A』이고『지역』이『북동부』인 레코드를 모두 검색하기 위해 데이터베이스 관리 프로그램은 먼저『제품』차원 블록 인덱스에서 제품 A 키를 검색합니다(72 페이지의 그림 19 참조). 그런 후 데이터베

이스 관리 프로그램은『지역』차원 블록 인덱스에서『북동부』키를 찾아『지역』이『북동부』인 모든 레코드가 포함된 블록을 판별합니다(그림 20 참조).

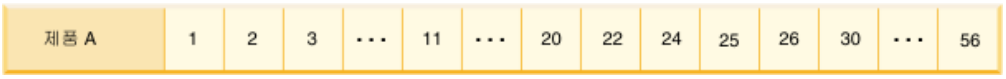


그림 19. '제품'의 차원 블록 인덱스 키

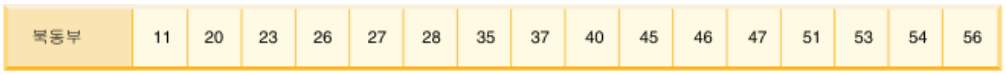


그림 20. '지역'의 차원 블록 인덱스 키

블록 인덱스 스캔은 논리적 AND 및 논리적 OR 연산자의 사용을 통해 결합될 수 있으며 스캔할 블록의 결과 목록은 클러스터된 데이터 액세스도 제공합니다.

위의 예제를 사용하여 두 차원 값을 갖는 모든 레코드가 포함된 블록 세트를 찾으려면 두 조각의 교집합을 찾아야 합니다. 이는 두 블록 인덱스 키의 BID 목록에서 논리적 AND 연산을 사용하면 됩니다. 공통 BID 값은 11, 20, 26, 45, 54, 51, 53 및 56입니다.

다음 예제에서는 블록 인덱스가 있는 논리적 OR 연산을 사용하여 두 차원이 포함된 술어를 갖는 쿼리를 충족시키는 방법을 예시합니다. 73 페이지의 그림 21에서는 두 차원이『Colour』및『Nation』인 MDC 테이블을 가정합니다. 목표는『Colour』가『blue』이거나『Nation』이름인『USA』인 조건을 충족시키는 MDC 테이블의 해당 레코드를 모두 검색하는 것입니다.

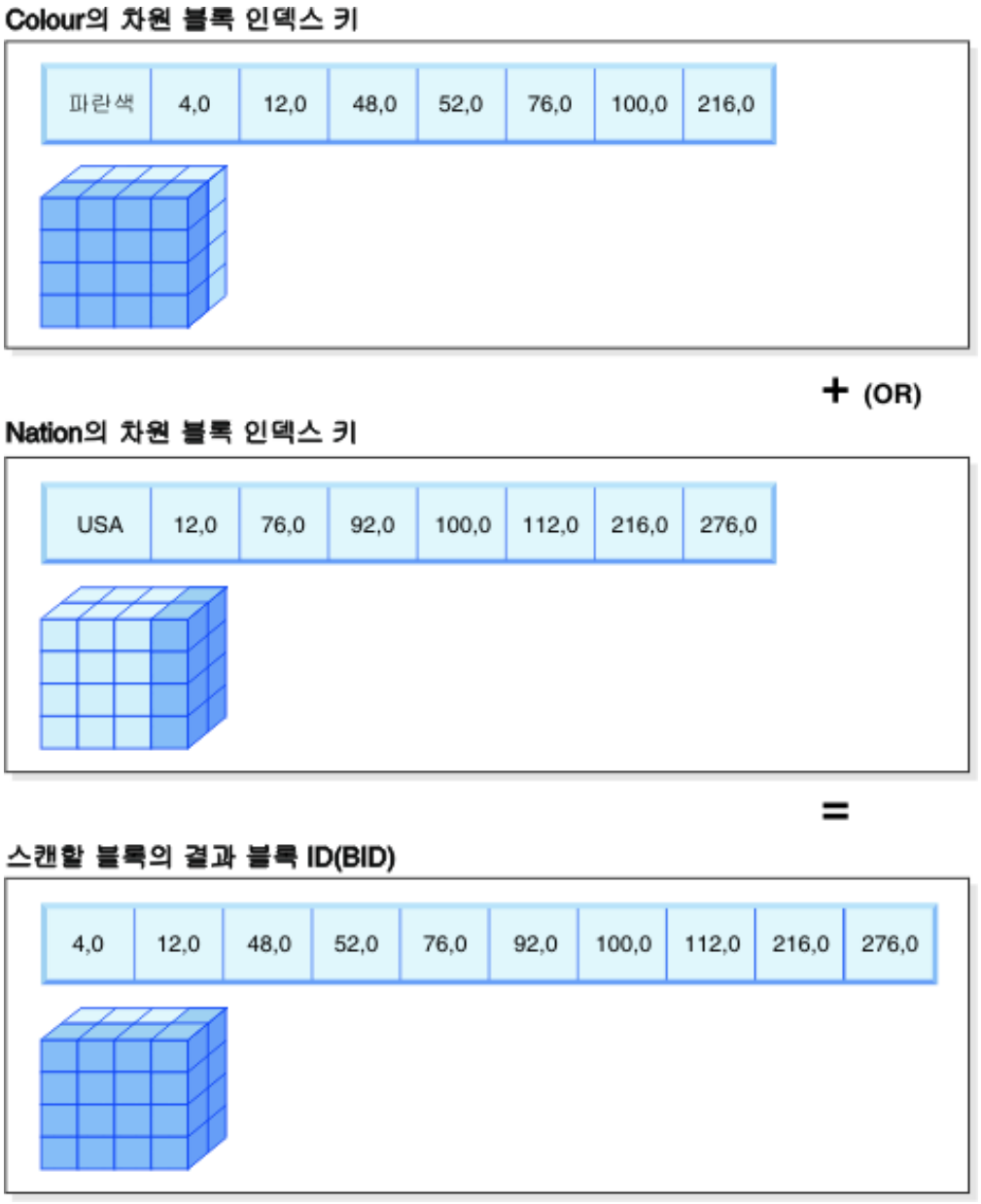


그림 21. 논리적 OR 연산을 블록 인덱스에서 사용하는 방법

이 다이어그램은 두 가지 별도의 블록 인덱스 스캔의 결과가 결합되어 술어 제한사항을 충족하는 값 범위를 결정하는 방식을 보여줍니다. (해당 번호는 레코드 ID(RID), 슬롯 필드를 표시합니다.)

SELECT문의 술어를 기반으로 하여 두 가지 별도의 차원 블록 인덱스 스캔이 수행됩니다. 하나는 파란색 조각에 대한 것이며 다른 하나는 USA 조각에 대한 것입니다. 두 조각의 합집합을 찾고 두 조각 모두에서 찾은 결합된 블록 세트를 판별하기 위해 논리적 OR 연산은 메모리에서 수행됩니다(중복된 블록 제거 포함).

일반 데이터베이스 관리 프로그램에 스캔할 블록 목록이 있으면 데이터베이스 관리 프로그램에는 각 블록에 대한 최소 관련 스캔을 수행할 수 있습니다. 블록 프리페치가 수

행될 수 있으며, 각 블록은 디스크에 Extent로 저장되고 버퍼 풀에서 하나의 단위로 읽을 수 있으므로 블록당 하나의 I/O만 포함됩니다. 술어를 데이터에 적용해야 하는 경우에는 차원 술어를 블록의 한 레코드에만 적용해야 합니다. 이는 블록의 모든 레코드는 동일 차원 키 값을 갖도록 보장되기 때문입니다. 다른 술어가 있는 경우, 데이터베이스 관리 프로그램은 블록의 나머지 레코드에서만 이 술어를 점검하면 됩니다.

또한 MDC 테이블은 일반 RID 기반 인덱스를 지원합니다. RID 및 블록 인덱스는 모두 인덱스에서 논리적 AND 연산 또는 논리적 OR 연산을 사용하여 결합될 수 있습니다. 블록 인덱스는 옵티마이저에게 선택할 추가 액세스 플랜을 제공하며 기존의 액세스 플랜(RID 스캔, 조인, 테이블 스캔 등)을 사용하지 못하도록 합니다. 옵티마이저는 특정 쿼리에 대해 가능한 다른 모든 액세스 플랜을 비롯한 블록 인덱스 플랜의 비용을 계산하여 가장 비용이 적게 드는 플랜을 선택합니다.

DB2 디자인 어드바이저는 MDC 테이블에 RID 기반 인덱스를 권장하거나 테이블에 대한 MDC 차원을 권장하는데 유용합니다.

INSERT 연산 중에 클러스터링 자동 유지보수

MDC 테이블에서 데이터 클러스터링을 자동으로 유지보수하려면 복합 블록 인덱스를 사용하면 됩니다. 이 기능은 INSERT 조작 중 테이블의 차원을 따라 데이터의 실제 클러스터링을 동적으로 관리하고 유지보수하는데 사용됩니다.

레코드가 포함된 테이블의 각 셀에 대해서만 이 복합 블록 인덱스에서 키를 찾을 수 있습니다. 따라서 이 블록 인덱스는 INSERT 중에 사용되어 논리적 셀이 테이블에 존재하는지 여부를 빠르고 효율적으로 판별할 수 있으며, 그런 경우에만 해당 셀의 특정 차원 값 세트를 갖는 레코드가 포함된 블록을 정확하게 판별합니다.

삽입이 발생하는 경우:

- 삽입될 레코드의 차원 값에 대응되는 논리적 셀이 있는지 복합 블록 인덱스를 조사합니다.
- 논리적 셀의 키가 인덱스에서 발견되면 블록 ID(BID) 목록은 논리적 셀의 차원 값을 갖는 테이블의 전체 블록 목록을 제공합니다(75 페이지의 그림 22 참조.) 이는 레코드 삽입 스페이스를 검색할 테이블의 Extent 수를 제한합니다.
- 논리적 셀의 키가 인덱스에 없는 경우 또는 이 값을 포함하는 Extent가 가득 차 있는 경우에는 새 블록이 논리적 셀에 지정됩니다. 가능한 경우에는 페이지의 다른 새 Extent(새 블록)로 테이블을 확장하기 전에 테이블에서 빈 블록의 재사용이 먼저 발생합니다.

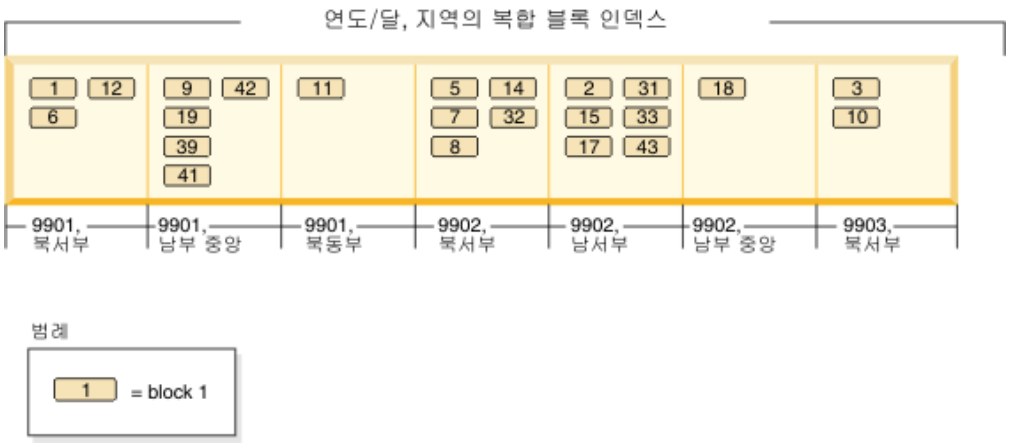


그림 22. '연도/달', '지역'의 복합 블록 인덱스

특정 차원 값을 갖는 데이터 레코드는 해당 값을 갖는 레코드만 포함하거나 모든 레코드를 포함하는 블록 세트에서 발견됩니다. 블록은 연속된 디스크 페이지로 구성됩니다. 결과적으로 이 레코드에 대한 액세스는 순차적이며 클러스터링을 제공합니다. 이러한 클러스터링은 레코드의 차원 값을 갖는 셀의 블록에만 레코드가 삽입되도록 함으로써 자동으로 시간에 따라 유지보수됩니다. 논리적 셀의 기존 블록이 가득 차 있으면, 빈 블록이 재사용되거나 새 블록이 해당 논리적 셀에 대한 블록 세트에 할당되거나 추가됩니다. 블록에서 데이터 레코드를 비우면 블록 ID(BID)가 블록 인덱스에서 제거됩니다. 이렇게 하면 나중에 다른 논리적 셀이 이용할 수 있도록 모든 논리적 셀 값으로부터 블록을 분리합니다. 따라서 테이블에 존재하는 데이터만 수용하기 위해 필요에 따라 셀 및 연관된 블록 인덱스 항목이 동적으로 테이블에 추가되고 테이블에서 제거됩니다. 논리적 블록 인덱스는 해당 값을 갖는 레코드가 포함된 블록에 셀 값을 맵핑하므로, 복합 블록 인덱스를 사용하여 이와 같은 작업을 관리할 수 있습니다.

클러스터링이 자동으로 이러한 방식으로 유지보수되므로, 데이터를 다시 클러스터링하기 위해 MDC 테이블 재구성이 필요하지 않습니다. 그러나 계속해서 재구성을 사용하여 스페이스를 요청할 수 있습니다. 예를 들어, 셀에 성긴 블록이 여러 개 있는데, 이 블록의 데이터를 소수의 블록에 맞출 수 있는 경우와 테이블에 다수의 포인터 오버플로우 쌍이 있는 경우, 테이블 재구성은 각각의 논리적 셀에 속하는 레코드를 필요한 최소 수의 블록으로 압축함은 물론 포인터-오버플로우 쌍도 제거합니다.

다음 예는 쿼리 처리에 복합 블록 인덱스를 사용할 수 있는 방법을 보여줍니다. 그림 22의 테이블에서 『영역』이 '북서부'이고 『연도/달』이 '9903'인 모든 레코드를 찾으려는 경우, 데이터베이스 관리 프로그램은 76 페이지의 그림 23에 표시된 대로 복합 블록 인덱스에서 키 값 9903, 북서부를 찾습니다. 키는 '9903, 북서부'라고 하는 키 값과 BID 목록으로 구성됩니다. 나열되는 BID는 3과 10뿐이며 판매 테이블에서 이들 두 특정 값을 가진 레코드가 포함된 블록이 두 개뿐임을 확인할 수 있습니다.

그림 23. '연도', '지역'의 복합 블록 인덱스의 키

삽입 중 복합 블록 인덱스의 사용을 보여주기 위해 차원 값이 9903과 북서부인 다른 레코드를 삽입하는 예를 사용하십시오. 데이터베이스 관리 프로그램은 복합 블록 인덱스에서 이 키 값을 찾아보고 3블록 및 10블록의 BID를 찾습니다. 이들 블록에는 모든 레코드와 이들 차원 키 값을 가진 레코드만 포함됩니다. 사용 가능한 스페이스가 있으면 데이터베이스 관리 프로그램은 새 레코드를 이 블록 중 하나에 삽입합니다. 이 블록의 어떤 페이지에도 스페이스가 없는 경우, 데이터베이스 관리 프로그램은 테이블에 새 블록을 할당하거나 테이블에서 이전에 비워진 블록을 사용합니다. 이 예에서 블록 48은 테이블에서 사용하고 있지 않은 블록입니다. 데이터베이스 관리 프로그램은 레코드를 블록에 삽입하며, 블록의 BID를 복합 블록 인덱스 및 각각의 차원 블록 인덱스에 추가하여 이 블록을 현재 논리적 셀과 연관시킵니다. 그림 24는 블록 48을 추가한 후의 차원 블록 인덱스의 키 그림입니다.

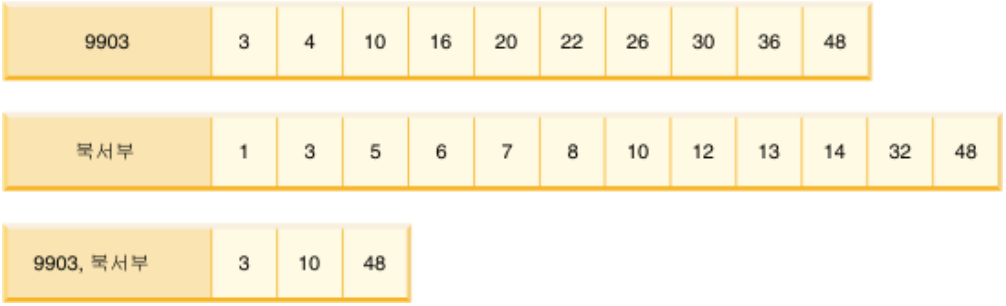


그림 24. 블록 48을 추가한 후의 차원 블록 인덱스 키

MDC 테이블에 대한 블록 맵

블록을 비울 때 블록 인덱스에서 해당 BID를 제거하면 현재 논리적 셀 값에서 블록이 분리됩니다. 그러면 블록을 다른 논리적 셀에서 다시 사용할 수 있습니다. 이에 따라 새 블록으로 테이블을 확장할 필요가 줄어듭니다.

새 블록이 필요하면 이를 위해 테이블을 검색할 필요 없이 이전에 비워진 블록을 신속히 찾아야 합니다.

블록 맵은 MDC 테이블에서 빈 블록을 보다 잘 찾을 수 있도록 하기 위해 사용되는 구조입니다. 블록 맵은 별도 오브젝트로 저장됩니다.

- SMS에서는 별도의 .BKM 파일로 저장됨
- DMS에서는 오브젝트 테이블의 새 오브젝트 디스크립터로 저장됨

블록 맵은 테이블의 각 블록에 대한 항목이 포함된 배열입니다. 각 항목은 블록에 대한 상태 비트 세트로 구성됩니다.

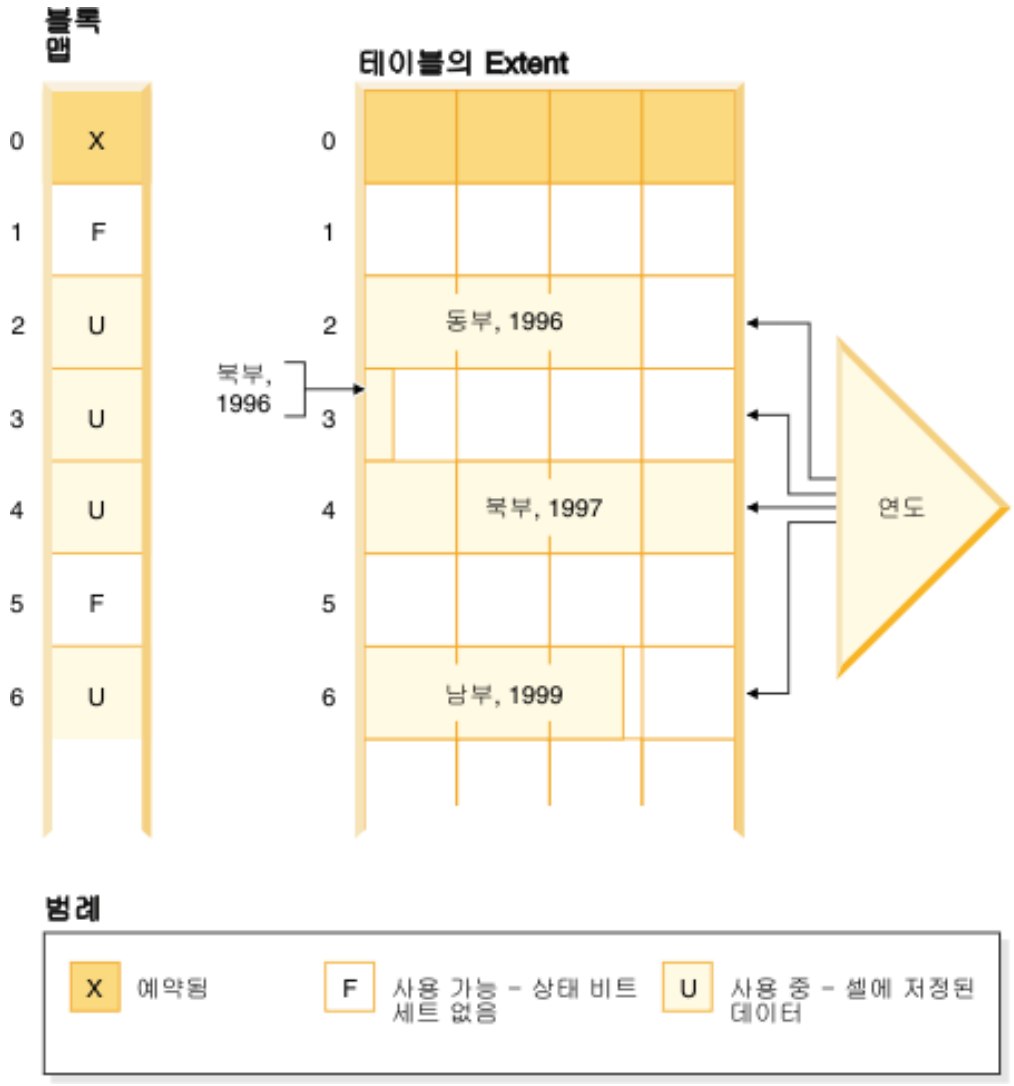


그림 25. 블록 맵의 작동 방법

그림 25에서 왼쪽 측면은 테이블의 각 블록에 대해 각기 다른 항목을 지닌 블록 맵 배열을 보여줍니다. 오른쪽 측면은 테이블의 각 Extent가 사용되는 방식을 보여줍니다. 대부분이 사용 중이며 일부는 사용 중이 아닙니다. 그리고 레코드는 블록 맵에서 사용 중인 것으로 표시된 블록에서만 발견됩니다. 간단히 하기 위해 두 차원 블록 인덱스 중 하나만 다이어그램에 표시했습니다.

주:

1. 블록 맵에서 사용 중으로 표시된 블록에 대해서만 블록 인덱스에 포인터가 있습니다.
2. 첫 번째 블록은 예약되어 있습니다. 이 블록에는 테이블에 대한 시스템 레코드가 포함되어 있습니다.

사용 중이 아닌 블록은 비어 있는(FREE) 블록(비트 세트가 없는 블록)에 대한 블록 맵을 스캔하여 셀에서 사용하기 위해 쉽게 찾을 수 있습니다.

테이블 스캔은 또한 블록 맵을 사용하여 현재 데이터가 포함된 Extent만 액세스할 수 있습니다. 사용 중이지 않은 모든 Extent는 테이블 스캔에 전혀 포함될 필요가 없습니다. 예를 들어, 이 예제의 테이블 스캔(77 페이지의 그림 25)은 첫 번째 예약된 Extent 및 다음의 빈 Extent를 건너뛰어 테이블의 세 번째 Extent(Extent 2)에서 시작되고, 테이블에서 블록 2, 3 및 4를 스캔하며, 다음 Extent를 건너뛴 후(Extent의 데이터 페이지를 전혀 건드리지 않고) 그 지점에서 스캔을 계속합니다.

MDC 테이블에서 삭제

레코드를 MDC 테이블에서 삭제하는 경우, 블록의 마지막 레코드가 아니면 데이터베이스 관리 프로그램은 단지 레코드만 제거하며 테이블에 정의된 모든 레코드 기반 인덱스에서 해당 RID를 삭제합니다.

그러나 삭제가 블록의 마지막 레코드를 제거하는 경우, 데이터베이스 관리 프로그램은 자체 IN_USE 상태 비트를 변경하고 모든 블록 인덱스에서 블록의 BID를 제거함으로써 블록을 비웁니다. 또한 레코드 기반 인덱스가 있으면 RID가 이 인덱스에서 제거됩니다.

주: 따라서 블록 인덱스 항목을 레코드 기반 인덱스에서 삭제된 행당 한 번만 제거하는 대신에 전체 블록당 한 번만 또는 블록이 완전히 비워진 경우에만 제거해야 합니다.

MDC 테이블의 갱신

MDC 테이블에서 비차원 값의 갱신은 일반 테이블에서 작업하는 것과 같은 위치에서 수행됩니다. 갱신이 가변 길이 컬럼에 영향을 주며 레코드가 더 이상 페이지에 맞지 않으면 충분한 스페이스의 다른 페이지가 발견됩니다.

이 새 페이지에 대한 검색은 동일 블록 내에서 시작됩니다. 해당 블록에 스페이스가 없으면, 새 레코드 삽입 알고리즘은 충분한 스페이스가 있는 논리적 셀에서 페이지를 찾는데 사용됩니다. 스페이스가 셀에서 발견되지 않으며 새 블록을 셀에 추가해야 하는 경우가 아니면 블록 인덱스를 갱신할 필요가 없습니다.

레코드는 자신이 속한 논리적 셀을 변경하므로, 차원 값 갱신은 현재 레코드를 삭제한 후 변경된 레코드를 삽입하는 것으로 간주됩니다. 현재 레코드의 삭제로 인해 블록이 비워지면 블록 인덱스를 갱신해야 합니다. 마찬가지로 새 레코드의 삽입으로 새 블록으로의 삽입이 필요하면 블록 인덱스를 갱신해야 합니다.

블록 인덱스는 첫 번째 레코드를 블록에 삽입하거나 블록에서 마지막 레코드를 삭제하는 경우에만 갱신해야 합니다. 따라서 유지보수 및 로깅을 위해 블록 인덱스와 연관된 인덱스 오버헤드는 일반 인덱스와 연관된 오버헤드보다 훨씬 작습니다. 일반 인덱스였던 모든 블록 인덱스의 경우에는 유지보수 및 로깅 오버헤드가 매우 감소합니다.

MDC 테이블은 기존 테이블과 마찬가지로 취급됩니다. 즉, 트리거, 참조 무결성, 뷰 및 구체화된 쿼리 테이블을 MDC 테이블에 대해 정의할 수 있습니다.

다차원 클러스터링 Extent 관리

다차원적으로 클러스터된(MDC) 테이블 내에서 데이터 Extent 비우기는 MDC 테이블의 재구성을 통해 완료됩니다.

MDC 테이블 내에서 블록 맵은 테이블에 속하는 모든 데이터 Extent를 추적하고 이에 관한 데이터가 있는 블록 및 Extent와 데이터가 없는 블록 및 Extent를 표시합니다. 데이터가 있는 블록은 『사용 중』으로 표시됩니다. 삭제 또는 롤아웃이 일어날 때마다 블록 맵이 있는 블록 항목이 더 이상 『사용 중』으로 표시되지 않고 MDC 테이블에서 재사용할 수 있게 여유 공간이 됩니다.

그러나 이 블록 및 Extent는 테이블 스페이스 내의 기타 오브젝트에 사용될 수 없습니다. MDC 테이블에서 이 여유 공간 데이터 Extent를 MDC 테이블 재구성을 통해 해제할 수 있습니다. MDC 테이블에서 Extent 비우기는 DMS 테이블 스페이스에서 MDC 테이블에만 지원됩니다.

REORG TABLE 명령은 RECLAIM EXTENTS ONLY 옵션을 사용하여 MDC 테이블의 독점 사용 Extent를 비우고 테이블 스페이스 내의 기타 데이터베이스 오브젝트에 사용할 스페이스를 사용 가능하게 합니다.

또한 이 옵션은 Extent를 비우는 동안 MDC 테이블에 대한 동시 액세스 제어를 허용합니다. 쓰기 액세스는 디폴트이고, 읽기 액세스 및 액세스 없음도 또한 동시 액세스를 제어하기 위한 선택사항입니다.

MDC 테이블이 범위 또는 파티션된 데이터베이스인 경우에도 디폴트로 모든 데이터 또는 데이터베이스 파티션의 Extent 비우기가 발생합니다. 파티션 이름(데이터 파티션의 경우) 또는 파티션 번호(데이터베이스 파티션의 경우)를 지정하여 특정 파티션에서만 Extent를 비우도록 명령을 실행하는 옵션이 있습니다.

REORG TABLE 명령은 Extent를 비우는 데 사용될 수 있고 db2Reorg API는 Extent 옵션 재개도 또한 허용합니다.

데이터베이스에 대한 자동 유지보수 활동의 Extent 파트를 비우기 위해 자동 지원이 사용 가능합니다. MDC 테이블의 Extent를 비우는 재구성을 사용하려면, **AUTO_MAINT**, **AUTO_TBL_MAINT** 및 **AUTO_REORG** 데이터베이스 구성 매개변수는 모두 『ON』 값을 가지고 있어야 합니다. 이 데이터베이스의 구성 매개변수 구성은 자동 유지보수 구성 마법사를 사용하거나 명령행을 사용하여 수행할 수 있습니다. 데이터베이스 파티셔닝 기능이 사용 가능한 DB2 인스턴스에서 카탈로그 파티션의 매개변수 구성을 실행해야 합니다.

유지보수 규정은 사용되지 않은 Extent를 비우기 위해 MDC 테이블의 자동 재구성이 일어나는 시간을 제어합니다. DB2 시스템 스토어드 프로시저 AUTOMAINT_SET_POLICY 및 AUTOMAINT_SET_POLICYFILE은 이 유지보수 규정을 설정하는 데 사용됩니다. XML은 자동화된 유지보수 규정을 저장하는 데 사용됩니다.

테이블 파티션 및 다중 클러스터링 테이블

한 테이블이 다차원 클러스터링 및 파티션될 수 있습니다. 다차원 클러스터링되면서 또한 파티션되는 테이블에서 컬럼을 테이블 파티션 범위 파티션 스펙 및 MDC 키에서 사용할 수 있습니다. 이는 어느 하나의 함수만 사용하여 달성할 수 있는 것보다 더 정교한 데이터 파티션 및 블록 제거를 달성하는데 유용합니다. 또한 테이블이 파티션되는 곳과는 다른 컬럼을 MDC 키에 대해 지정하는 것이 유용한 많은 응용프로그램이 있습니다. MDC는 다차원인데 반해 테이블 파티션은 다중 컬럼임을 주의해야 합니다. 파티션된 테이블의 인덱스는 파티션되거나 파티션되지 않을 수 있습니다. MDC 블록 인덱스는 항상 파티션되지 않습니다.

메인스트림 DB2 데이터 웨어하우스의 특성

다음 권장사항은 DB2 V9.1에서 새로운 기능인 전형적인 메인스트림 웨어하우스에 초점을 두었습니다. 다음 특성이 가정됩니다.

- 데이터베이스가 다중 머신 또는 다중 AIX 논리적 파티션에서 실행됩니다.
- 데이터베이스 파티션 기능(DPF)이 사용됩니다(테이블이 DISTRIBUTE BY HASH 절을 사용하여 작성됩니다).
- 4 - 50개의 데이터 파티션이 있습니다.
- MDC 및 테이블 파티션이 고려되고 있는 테이블이 주요 사실 테이블입니다.
- 테이블이 1억 - 1000억 개의 행을 갖습니다.
- 새 데이터가 다양한 시간 프레임(야간, 주별, 월별)에 로드됩니다.
- 일별 처리 볼륨은 1만 - 1000만 레코드입니다.

- 데이터 볼륨은 변합니다. 가장 큰 월은 가장 작은 월 *크기*의 5X입니다. 마찬가지로 가장 큰 차원(제품 행, 지역)은 *크기* 범위의 5X입니다.
- 1 - 5년의 자세한 데이터가 보유됩니다.
- 만기된 데이터는 월별 또는 분기별로 롤아웃됩니다.
- 테이블은 광범위한 쿼리 유형을 사용합니다. 그러나 워크로드는 대부분 OLTP 워크로드에 상대적으로 다음 특성을 갖는 분석 쿼리입니다.
  - 최고 2백만 행을 갖는 더 큰 결과 세트
  - 대부분 또는 모든 쿼리는 기본 테이블이 아니라 적중하는 보기
- 범위(BETWEEN절), 목록의 항목 등으로 데이터를 선택하는 SQL절

메인스트림 DB2 V9.1 데이터 웨어하우스 사실 테이블의 특성

일반적인 웨어하우스 사실 테이블은 다음 설계를 사용할 수 있습니다.

- 월 컬럼에 데이터 파티션을 작성하십시오.
- 롤 아웃하려는 각 기간(예: 한 달, 3개월)에 대한 데이터 파티션을 정의하십시오.
- 일에 대한 MDC 차원을 작성하고 1 - 4개의 추가 차원을 작성하십시오. 일반적인 차원은 제품 행과 지역입니다.
- 모든 데이터 파티션 및 MDC 클러스터가 모든 데이터베이스 파티션에 분산됩니다.

MDC 및 테이블 파티션은 중첩되는 일련의 이점을 제공합니다. 다음 표는 조직에서 잠재적인 요구를 나열하고 이전에 식별된 특성을 기초로 권장되는 조직 스킴을 식별합니다.

표 7. MDC 테이블에 테이블 파티션 사용

문제점	권장 스킴	권장사항
롤아웃 동안 데이터 사용 가능성	테이블 파티션	DETACH PARTITION절을 사용하여 최소한의 장애로 대형 데이터 롤아웃
쿼리 성능	테이블 파티션 및 MDC	다중 차원 쿼리를 위해 MDC가 최선입니다. 테이블 파티션이 데이터 파티션을 제거하여 돕습니다.
최소 재구성	MDC	MDC가 클러스터링을 유지보수하여 재구성의 필요성을 줄입니다.
일반적인 오프라인 창 동안 1개월 이상의 데이터를 롤아웃합니다.	테이블 파티션	데이터 파티션이 이 필요성을 전적으로 다룹니다. MDC는 아무 것도 추가하지 않으며 그 자체로는 적합하지 않습니다.
마이크로 오프라인 창(1분 미만) 동안 1개월 이상의 데이터를 롤아웃합니다.	테이블 파티션	데이터 파티션이 이 필요성을 전적으로 다룹니다. MDC는 아무 것도 추가하지 않으며 그 자체로는 적합하지 않습니다.

표 7. MDC 테이블에 테이블 파티션 사용 (계속)

문제점	권장 스킴	권장사항
어떤 서비스도 중지하지 않으면서 쿼리를 제출하는 비즈니스 사용자에 대해 테이블을 완전히 사용 가능하게 유지하면서 1개월 이상의 데이터를 롤아웃합니다.	MDC	MDC만 이 필요성을 다소 다룹니다. 테이블 파티션은 테이블이 오프라인이 되는 짧은 기간으로 인해 적합하지 않을 수 있습니다.
매일 데이터 로드(ALLOW READ ACCESS 또는 ALLOW NO ACCESS)	테이블 파티션 및 MDC	MDC가 이 이점의 대부분을 제공합니다. 테이블 파티션은 증분 이점을 제공합니다.
"연속" 데이터 로드(ALLOW READ ACCESS)	테이블 파티션 및 MDC	MDC가 이 이점의 대부분을 제공합니다. 테이블 파티션은 증분 이점을 제공합니다.
"전통적인 BI" 쿼리에 대한 쿼리 실행 성능	테이블 파티션 및 MDC	MDC가 큐브/다차원 쿼리에 특히 좋습니다. 테이블 파티션은 파티션 제거를 통해 도움이 됩니다.
재구성 필요성을 피하거나 태스크 수행과 연관된 어려움을 줄여서 재구성 어려움을 최소화합니다.	MDC	MDC는 재구성할 필요성을 줄이는 클러스터링을 유지보수합니다. MDC가 사용되는 경우 데이터 파티션은 증분 이점을 제공하지 않습니다. 그러나 DC가 사용되지 않는 경우 테이블 파티션이 일부 과정 그레인 클러스터링을 파티션 레벨에서 유지보수하여 재구성 필요성을 줄이는데 도움이 됩니다.

예 1:

키 컬럼 YearAndMonth 및 Province를 갖는 테이블을 고려하십시오. 이 테이블을 플랜하는 합리적인 접근법은 데이터 파티션당 2개월의 날짜를 파티션하는 것일 수 있습니다. 또한 Province별로 구성하여, 임의의 날짜 범위 2개월 내에서 특정 지방에 대한 모든 행이 37 페이지의 그림 6에 표시된 것처럼 함께 클러스터되도록 할 수 있습니다.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

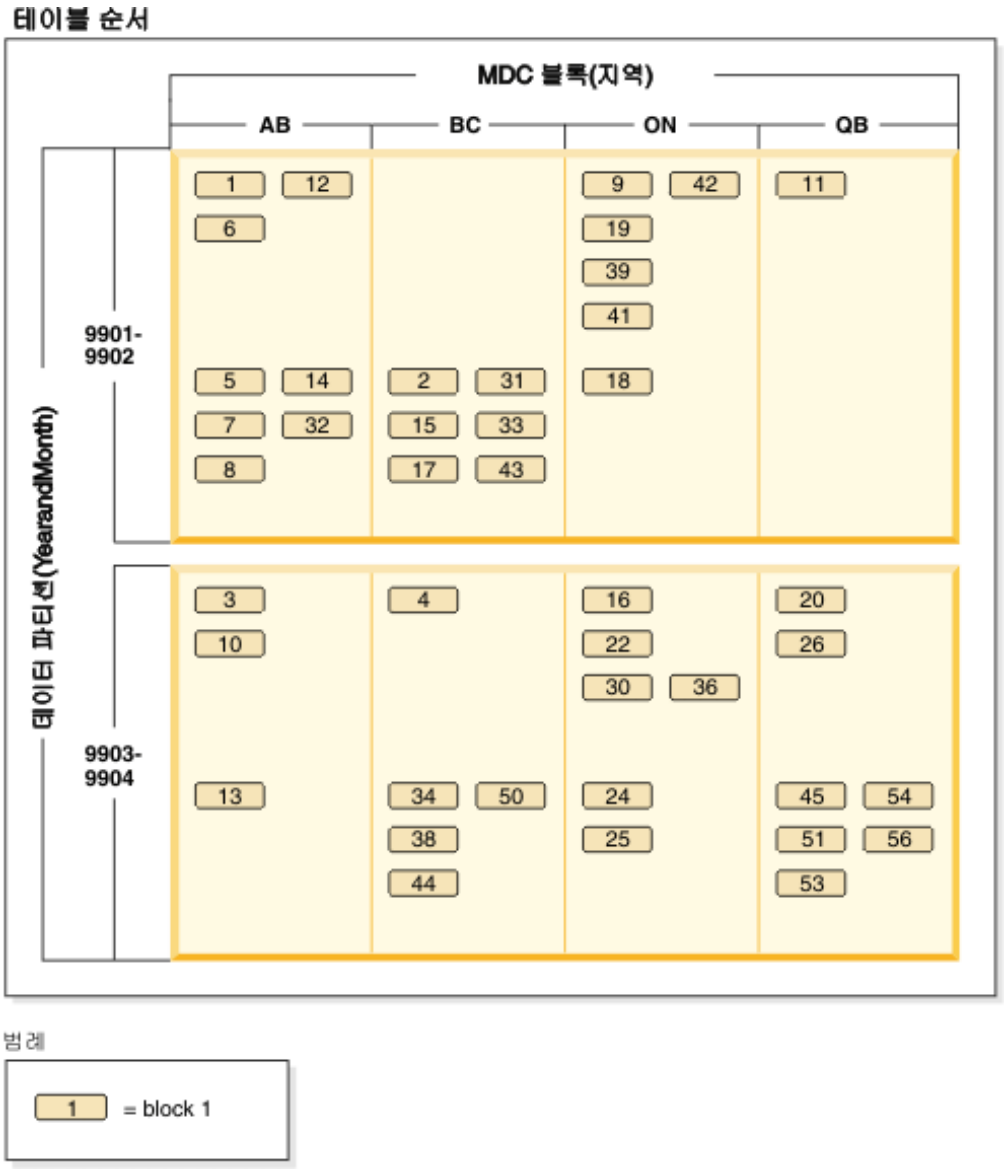


그림 26. YearAndMonth로 파티션되고 Province로 구성되는 테이블

예 2

38 페이지의 그림 7에서 보는 것처럼 ORGANIZE BY절에 YearAndMonth를 추가하여 더 정교한 세분화도를 달성할 수 있습니다.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

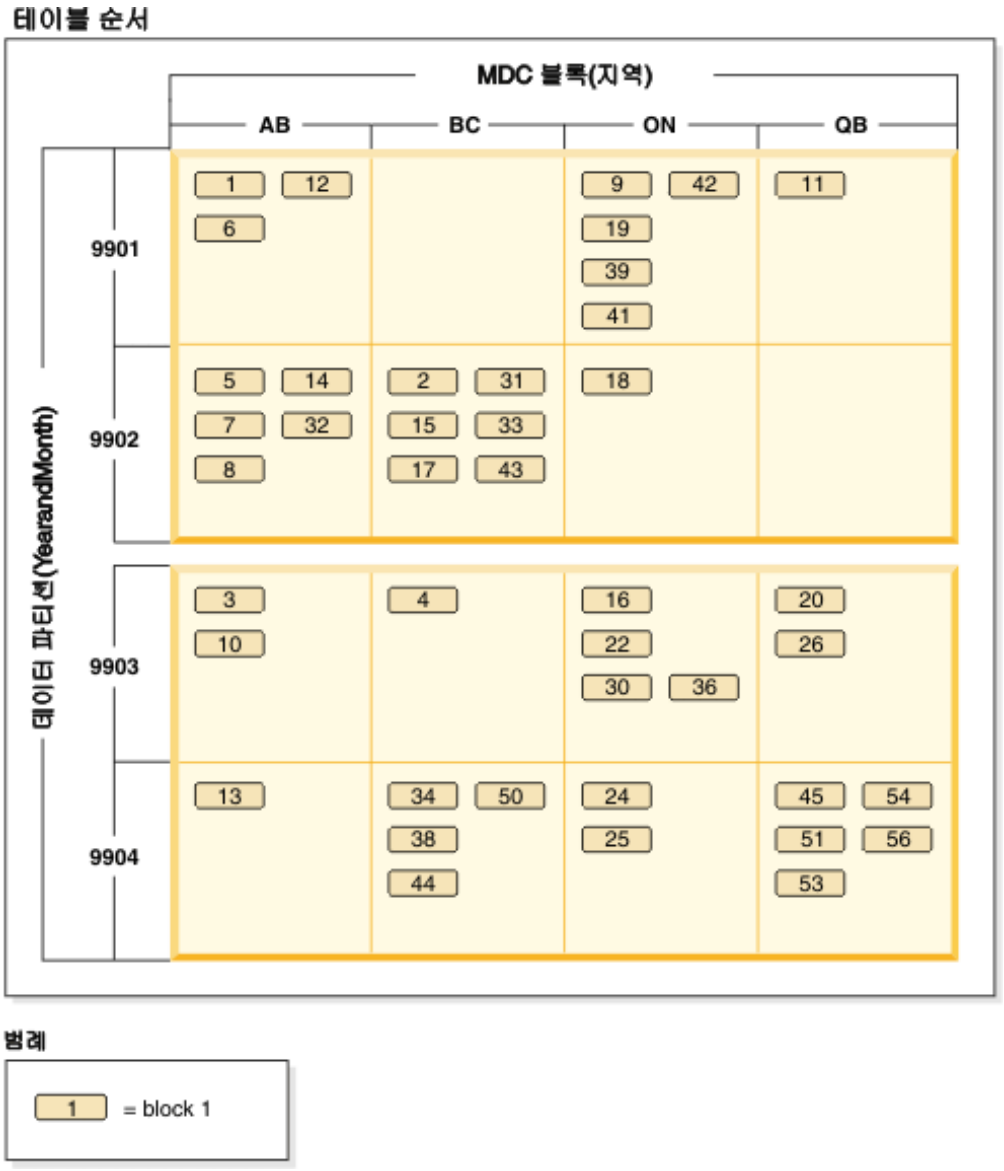


그림 27. YearAndMonth로 파티션되고 Province 및 YearAndMonth별로 구성되는 테이블

각 범위에 단일 값만 존재하도록 파티션되는 경우에 MDC 키에 테이블 파티션 컬럼을 포함시켜서 얻는 것이 없습니다.

고려사항

- 기본 테이블과 비교할 때, MDC 테이블 및 파티션된 테이블이 둘 다 추가 스토리지가 필요합니다. 이러한 스토리지 필요성은 부가적이지만, 이점이 있는 타당한 것으로 간주됩니다.
- 파티션된 데이터베이스 환경에서 MDC 기능을 테이블 파티션과 결합하지 않을 것을 선택하는 경우, 일반적으로 여기서 논의하는 시스템 유형에 해당하는 경우인 데이터 분산을 자신있게 예측할 수 있는 경우에 테이블 파티션이 최적입니다. 그렇지 않으면, MDC를 고려해야 합니다.

제 4 장 병렬 데이터베이스 시스템

병렬 처리

데이터베이스 쿼리와 같은 태스크의 구성요소는 병렬로 실행되어 극적으로 성능을 향상시킬 수 있습니다. 태스크 속성, 데이터베이스 구성 및 하드웨어 환경은 모두 DB2 데이터베이스 제품이 병렬로 태스크를 수행하는 방식을 결정합니다.

이러한 고려사항은 서로 관련되며, 데이터베이스의 실제 설계 및 논리적 설계로 작업할 때 함께 고려해야 합니다. 다음은 DB2 데이터베이스 시스템에서 지원하는 병렬 처리 유형입니다.

- 입출력
- 쿼리
- 유틸리티

입/출력(I/O) 병렬 처리

하나의 테이블 스페이스에 대해 복수의 컨테이너가 존재할 때, 데이터베이스 관리 프로그램은 *병렬 입출력*을 활용할 수 있습니다. 병렬 입출력은 동시에 둘 이상의 입출력 디바이스에 쓰거나 읽는 프로세스를 나타내며 처리량을 상당히 개선시킬 수 있습니다.

쿼리 병렬 처리

쿼리 병렬 처리에는 쿼리 간 병렬 처리와 쿼리 내의 병렬 처리의 두 가지 유형이 있습니다.

*쿼리 간 병렬 처리*는 여러 응용프로그램에서 동시에 쿼리하는 데이터베이스의 기능입니다. 각 쿼리는 서로 독립적으로 실행되지만, 데이터베이스 관리 프로그램은 이들을 Each query runs independently of the 모두 동시에 실행합니다. DB2 데이터베이스 제품은 이 병렬 처리 유형을 항상 지원합니다.

*쿼리 내 병렬 처리*는 *파티션 내 병렬 처리* 및 *파티션 간 병렬 처리* 중 하나 또는 둘 다를 사용하여 동시에 하나의 쿼리에서 여러 부분을 처리하는 것입니다.

파티션 내 병렬 처리

*파티션 내 병렬 처리*는 하나의 쿼리를 여러 부분으로 분해하는 기능입니다. 일부 DB2 유틸리티에서 이 병렬 처리 유형을 수행하기도 합니다.

파티션 내 병렬 처리는 인덱스 작성, 데이터베이스 로드 또는 SQL 쿼리와 같이 일반적으로 단일 데이터베이스 조작으로 간주되는 조작을 여러 부분으로 세분합니다. 이런 조작의 일부 또는 전부를 *단일 데이터베이스 파티션 내에서* 병렬로 실행할 수 있습니다.

그림 28은 병렬로 실행 가능하며, 쿼리가 연속 방식으로 실행될 경우보다 더 빨리 리턴되는 네 부분으로 분해된 쿼리를 나타냅니다. 각 조각은 서로에 대한 사본입니다. 파티션 내 병렬 처리를 사용하려면, 데이터베이스를 적절하게 구성해야 합니다. 사용자가 직접 병렬 처리 정도를 선택하거나 시스템이 임의로 선택하게 할 수 있습니다. 병렬 처리 정도는 병렬로 실행하는 쿼리 조각의 개수입니다.

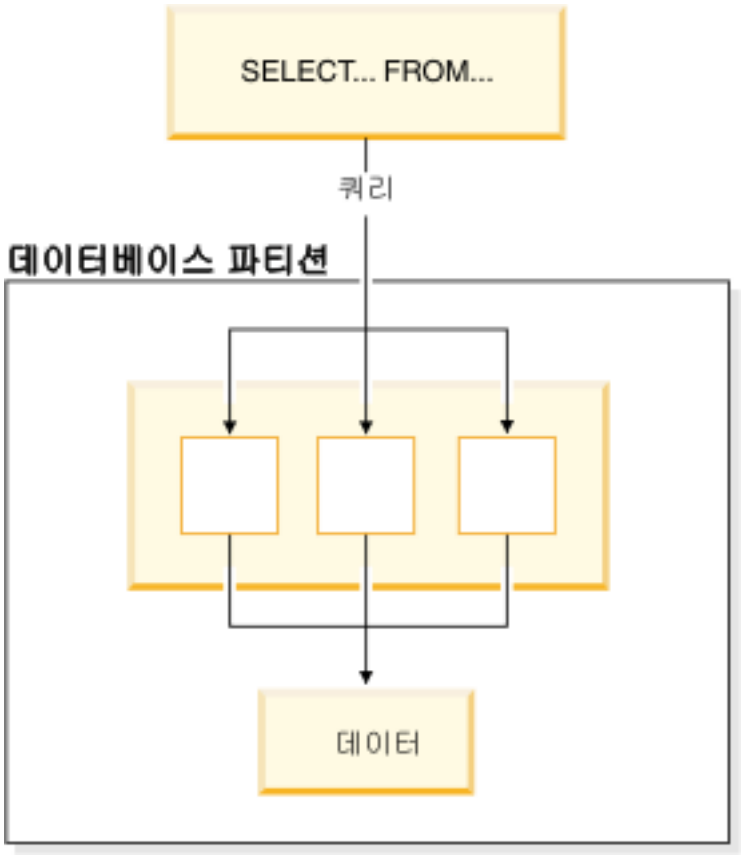


그림 28. 파티션 내 병렬 처리

파티션 간 병렬 처리

*파티션 간 병렬 처리*는 하나 이상의 머신에서 파티션된 데이터베이스의 다중 파티션에 하나의 쿼리를 여러 부분으로 분해하는 기능입니다. 쿼리는 병렬로 실행됩니다. 일부 DB2 유틸리티에서 이 병렬 처리 유형을 수행하기도 합니다.

파티션 간 병렬 처리는 인덱스 작성, 데이터베이스 로드 또는 SQL 쿼리와 같이 일반적으로 단일 데이터베이스 조작으로 간주되는 조작을 여러 부분으로 세분합니다. 이런 조작의 일부 또는 전부는 *하나 이상의 머신에 있는 파티션된 데이터베이스의 다중 파티션에서* 병렬로 실행할 수 있습니다.

그림 29는 병렬로 실행 가능하며, 단일 데이터베이스 파티션에서 쿼리가 연속 방식으로 실행될 경우보다 더 빨리 리턴되는 네 부분으로 분해된 쿼리를 나타냅니다.

병렬 처리 정도는 대개 작성하는 데이터베이스 파티션 수와 데이터베이스 파티션 그룹의 정의 방식에 따라 결정됩니다.

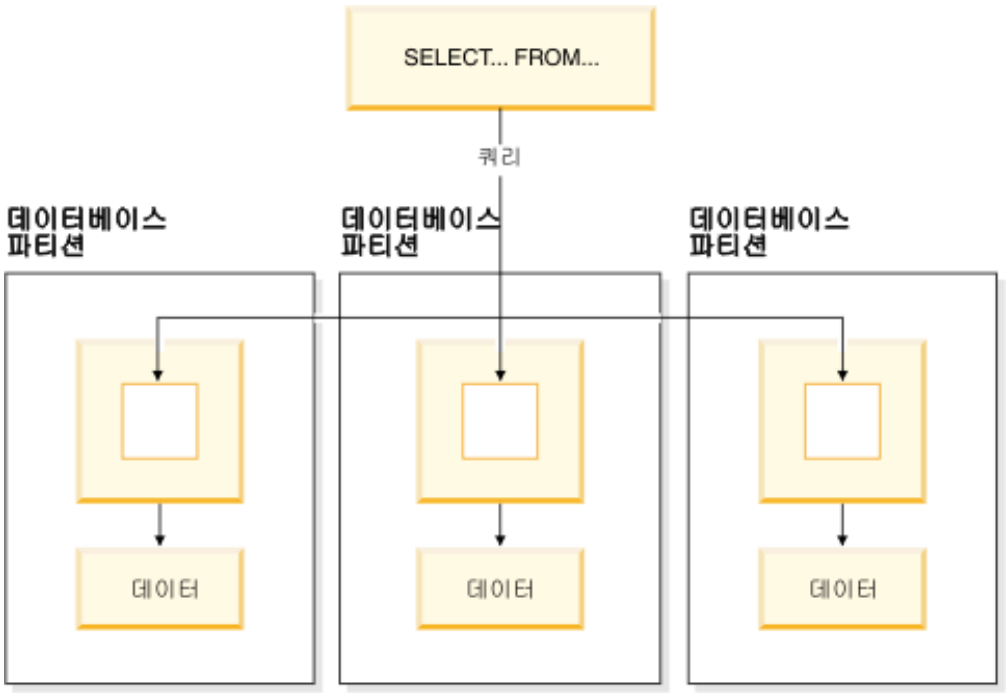


그림 29. 파티션 간 병렬 처리

동시 파티션 간 및 파티션 내 병렬 처리

파티션 내 병렬 처리와 파티션 간 병렬 처리를 동시에 사용할 수 있습니다. 이 조합은 두 가지 병렬 처리 차원을 제공하므로 쿼리를 훨씬 빠르게 처리할 수 있습니다.

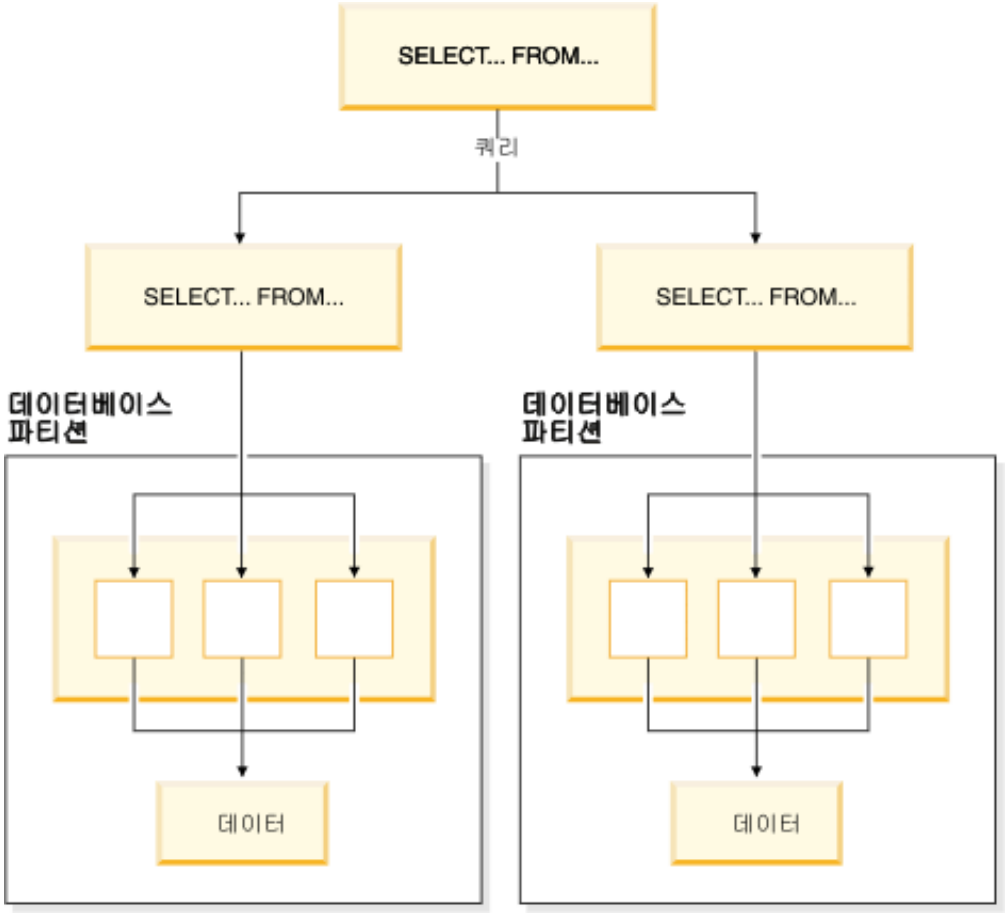


그림 30. 동시 파티션 간 및 파티션 내 병렬 처리

유틸리티 병렬 처리

DB2 유틸리티는 파티션 내 병렬 처리를 사용할 수 있습니다. 파티션 간 병렬 처리를 모두 사용할 수 있습니다. 파티션 데이터베이스가 여러 개 있으면, 유틸리티는 각 데이터베이스 파티션에서 병렬로 실행합니다.

로드 유틸리티는 파티션 내 병렬 처리와 입출력 병렬 처리를 사용할 수 있습니다. 데이터 로드는 CPU를 많이 사용하는 태스크입니다. 로드 유틸리티는 데이터의 구문 분석 및 형식화와 같은 태스크에 대해 멀티 프로세서를 사용합니다. 또한 병렬 입출력 서버를 사용하여 데이터를 컨테이너에 병렬로 기록할 수도 있습니다.

파티션된 데이터베이스 환경에서 LOAD 명령은 테이블이 있는 각 데이터베이스 파티션에서 병렬 호출을 수행하여 파티션 내, 파티션 간 및 입출력 병렬 처리를 사용합니다.

인덱스 작성 시 데이터 스캔과 이어지는 정렬은 병렬로 처리됩니다. DB2 시스템은 인덱스 작성 시 입출력 병렬 처리와 파티션 내 병렬 처리를 이용합니다. 이는 CREATE INDEX 명령문을 실행할 때, 재시작(인덱스가 유효하지 않다고 표시될 경우) 및 데이터 재구성 중에 인덱스 작성의 속도를 높이는데 유용합니다.

데이터의 백업 및 리스토어는 입출력이 많이 발생하는 태스크입니다. DB2 시스템은 백업 및 리스토어 조작을 실행할 경우, 입출력 병렬 처리와 파티션 내 병렬 처리를 이용합니다. 백업은 병렬로 다중 테이블 스페이스 컨테이너로부터 읽고, 병렬로 복수의 백업 미디어를 비동기적으로 기록하여 입출력 병렬 처리를 이용합니다.

파티션된 데이터베이스 환경

데이터베이스 파티션 기능(DPF)은 데이터베이스 관리 프로그램을 병렬, 다중 파티션 환경으로 확장시킵니다.

- *데이터베이스 파티션*은 고유의 데이터, 인덱스, 구성 파일 및 트랜잭션 로그로 이루어진 데이터베이스의 일부분입니다. 데이터베이스 파티션을 노드 또는 데이터베이스 노드라고도 합니다. 파티션된 데이터베이스 환경은 데이터베이스 파티션에 있는 데이터의 분산을 지원합니다.
- *단일 파티션 데이터베이스*는 하나의 데이터베이스 파티션만 있는 데이터베이스입니다. 데이터베이스의 모든 데이터는 하나의 데이터베이스 파티션에 저장됩니다. 이런 경우, 데이터베이스 파티션 그룹에는 추가 기능이 없습니다.
- *다중 파티션 데이터베이스*는 두 개 이상의 데이터베이스 파티션을 가진 데이터베이스입니다. 테이블은 하나 이상의 데이터베이스 파티션에 있을 수 있습니다. 테이블이 다중 데이터베이스 파티션으로 구성된 데이터베이스 파티션 그룹에 있으면, 테이블의 각 행은 여러 데이터베이스 파티션에 나뉘어 저장됩니다.

일반적으로 단일 데이터베이스 파티션은 각 물리적 머신에 있으며, 각 데이터베이스 파티션의 데이터베이스 관리 프로그램은 시스템의 프로세서를 사용하여 데이터베이스에서 전체 데이터의 해당 부분을 관리합니다.

데이터는 여러 데이터베이스 파티션에 분산되어 있기 때문에 다중 물리적 머신의 멀티프로세서를 사용하여 정보에 대한 요청을 충족시킬 수 있습니다. 데이터 검색 및 갱신 요청은 자동으로 하위 요청으로 분할되며, 적용 가능한 데이터베이스 파티션에서 병렬 상태로 실행됩니다. 데이터베이스가 여러 데이터베이스 파티션으로 나뉜다는 사실을 SQL 문을 발행하는 사용자는 명확히 알 수 있습니다.

해당 사용자의 *코디네이터 파티션*으로 알려진 데이터베이스 파티션을 통해 사용자 상호작용이 발생합니다. 코디네이터 파티션은 같은 데이터베이스 파티션에서 응용프로그램으로 실행되고, 리모트 응용프로그램의 경우 해당 응용프로그램이 연결된 데이터베이스 파티션에서 실행됩니다. 모든 데이터베이스 파티션은 코디네이터 파티션으로 사용될 수 있습니다.

데이터베이스 관리 프로그램은 데이터베이스의 여러 데이터베이스 파티션에 데이터를 저장할 수 있게 합니다. 즉, 데이터는 물리적으로는 둘 이상의 데이터베이스 파티션에 저

장되지만, 마치 같은 장소에 있는 것처럼 액세스될 수 있습니다. 다중 파티션 데이터베이스의 데이터에 액세스하는 응용프로그램과 사용자는 데이터의 물리적 위치를 알 필요가 없습니다.

데이터는 물리적으로 분할되어 있어도, 논리적으로는 하나의 전체 단위로 사용되고 관리됩니다. 사용자는 분산 키를 선언하여 데이터 분산 방식을 선택할 수 있습니다. 또한 데이터를 저장할 테이블 스페이스 및 관련 데이터베이스 파티션 그룹을 선택함으로써 데이터를 분배할 데이터베이스 파티션 수도 결정할 수 있습니다. 분배 및 복제에 대한 제안은 DB2 디자인 어드바이저를 사용하여 수행될 수 있습니다. 이외에도 갱신할 수 있는 분산 맵을 해싱 알고리즘과 함께 사용하여 분산 키 값과 데이터베이스 파티션 간의 맵핑을 지정할 수 있습니다. 이를 통해 데이터의 각 행의 위치와 검색 방법이 결정됩니다. 결과적으로, 보다 작은 테이블은 하나 이상의 데이터베이스 파티션에 저장할 수 있도록 하는 동시에 큰 테이블의 워크로드는 다중 파티션 데이터베이스에 분배할 수 있습니다. 각 데이터베이스 파티션에는 저장된 데이터에 대한 로컬 인덱스가 있으므로 로컬 데이터 액세스 성능이 향상됩니다.

주: 모든 테이블은 데이터베이스의 모든 데이터베이스 파티션으로 나눌 수 있습니다. 데이터베이스 관리 프로그램은 *부분 클러스터링 해제*를 지원하는데, 이 기능은 테이블 및 테이블 스페이스를 시스템의 데이터베이스 파티션 서브세트에 분할할 수 있는 기능입니다.

각 데이터베이스 파티션에 테이블을 배치할 때 고려해야 할 또 다른 사항은 구체화된 쿼리 테이블(Materialized Query Table)을 사용한 다음 해당 테이블을 복제하는 것입니다. 필요한 정보를 포함하는 구체화된 쿼리 테이블을 작성한 다음 이 테이블을 각 데이터베이스 파티션으로 복제할 수 있습니다.

DB2 데이터베이스 제품의 루트 서버가 아닌 설치는 데이터베이스 파티셔닝을 지원하지 않습니다. 그 결과로 노드 추가 연산을 실행할 수 없습니다. `db2nodes.cfg` 파일을 수동으로 갱신해서는 안됩니다. 수동 갱신의 결과로 SQL6031N 오류가 표시됩니다.

데이터베이스 파티션 및 프로세서 환경

*용량*은 데이터베이스에 액세스할 수 있는 사용자 및 응용프로그램 수를 나타냅니다. 용량은 대부분 메모리, 에이전트, 잠금, 입출력 및 스토리지 관리에 따라 결정됩니다. *확장성*은 데이터베이스의 규모를 확대하면서도 동일한 조작 특성과 응답 시간을 유지하는 기능입니다.

이 절은 다음 하드웨어 환경에 대한 개요를 제공합니다.

- 단일 프로세서상의 단일 데이터베이스 파티션(단일 프로세서)
- 다중 프로세서가 있는 단일 데이터베이스 파티션(SMP)
- 다중 데이터베이스 파티션 구성

- 하나의 프로세서가 있는 여러 데이터베이스 파티션(MPP)
- 다중 프로세서가 있는 여러 데이터베이스 파티션(SMP의 클러스터)
- 논리 데이터베이스 파티션

각 환경에 대한 용량 및 확장성에 대해 설명합니다.

단일 프로세서상의 단일 데이터베이스 파티션

이 환경은 메모리와 디스크로 구성되며 하나의 CPU만 포함합니다(그림 31 참조). 이런 환경을 독립형 데이터베이스, 클라이언트/서버 데이터베이스, 직렬 데이터베이스, 단일 프로세서 시스템, 단일 노드 또는 병렬이 아닌 환경이라고도 합니다.

이 환경의 데이터베이스는 데이터 및 시스템 자원(단일 프로세서 또는 CPU 포함)이 단일 데이터베이스 관리 프로그램에 의해 관리되는 부서나 소규모 사무실에서 주로 사용합니다.

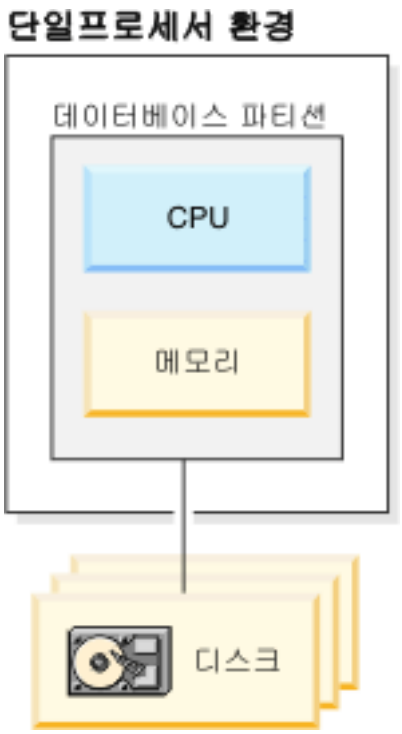


그림 31. 단일 프로세서상의 단일 데이터베이스 파티션

용량 및 확장성

이 환경에서는 더 많은 디스크를 추가할 수 있습니다. 각 디스크에 대해 하나 이상의 입출력 서버를 사용하면 동시에 둘 이상의 입출력 조작을 수행할 수 있습니다.

단일 프로세서 시스템은 프로세서가 다룰 수 있는 디스크 스페이스 크기에 따라 제한됩니다. 워크로드가 증가하면, 추가할 수 있는 메모리 또는 디스크와 같은 기타 구성요소와 관계없이 사용자 요청을 빠르게 처리할 수 없습니다. 최대 용량 또는 확장성에 도달

한 경우 다중 프로세서가 있는 단일 데이터베이스 파티션 시스템으로 이동하는 것이 좋습니다.

다중 프로세서가 있는 단일 데이터베이스 파티션

이 환경은 일반적으로 같은 머신 안에서 동일한 성능을 가진 여러 개의 프로세서로 구성되며(그림 32 참조), *대칭적 멀티 프로세서(SMP)* 시스템이라고 합니다. 디스크 스페이스 및 메모리 같은 자원을 *공유*합니다.

멀티 프로세서를 사용할 수 있으면 여러 데이터베이스 조작을 더 빨리 완료할 수 있습니다. DB2 데이터베이스 시스템은 처리 속도를 향상시키기 위해 사용 가능한 프로세서로 단일 쿼리 작업을 나눌 수도 있습니다. 데이터 로드, 테이블 스페이스 백업 및 리스토어 그리고 기존 데이터의 인덱스 작성과 같은 다른 데이터베이스 조작은 멀티 프로세서를 이용할 수 있습니다.

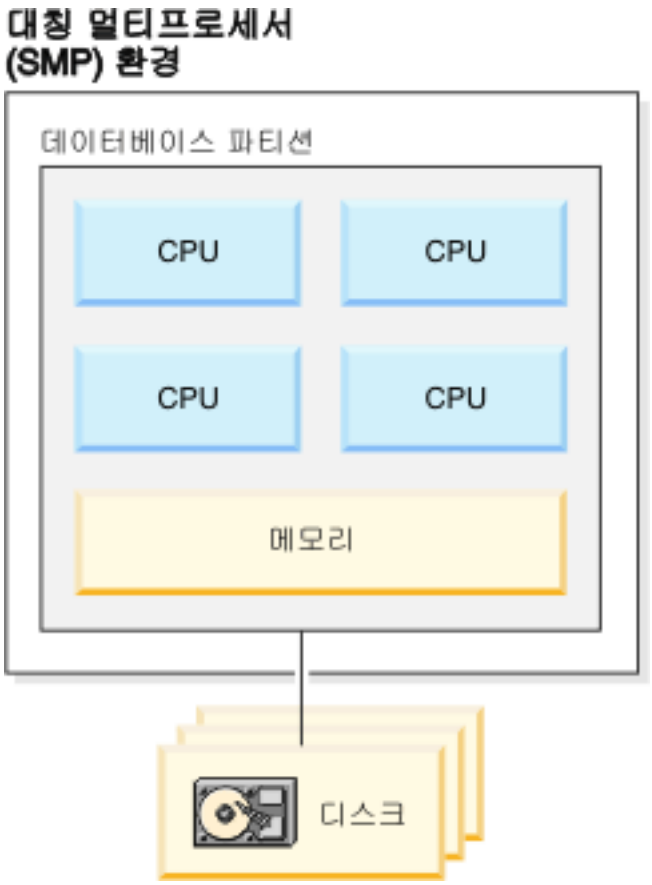


그림 32. 단일 파티션 데이터베이스 대칭 멀티프로세서 환경

용량 및 확장성

이 환경에서는 더 많은 프로세서를 추가할 수 있습니다. 그러나 여러 프로세서가 같은 데이터에 액세스하려고 하므로, 이 환경에서 비즈니스 조작이 계속 증가하면 한계가 나타날 수 있습니다. 공유 메모리 및 공유 디스크를 사용하면, 모든 데이터베이스 데이터를 효과적으로 공유하는 것입니다.

디스크 수를 늘려 프로세서와 연관된 데이터베이스 파티션의 입출력 용량을 늘릴 수 있습니다. 특별히 입출력 요청을 처리하는 입출력 서버를 설정할 수 있습니다. 각 디스크에 대해 하나 이상의 입출력 서버를 사용하면 동시에 둘 이상의 입출력 조작을 수행할 수 있습니다.

최대 용량 또는 확장성에 도달하면, 다중 데이터베이스 파티션이 있는 시스템으로 이동하는 것이 좋습니다.

다중 데이터베이스 파티션 구성

데이터베이스를 각 자체 소유 머신에서 다중 데이터베이스 파티션으로 나눌 수 있습니다. 이 경우 각 파티션은 자체 머신에 있습니다. 다중 파티션 데이터베이스가 있는 다중 머신은 그룹화할 수도 있습니다. 이 절은 다음 데이터베이스 파티션 구성을 설명합니다.

- 하나의 프로세서가 있는 시스템의 여러 데이터베이스 파티션
- 다중 프로세서가 있는 시스템의 여러 데이터베이스 파티션
- 논리 데이터베이스 파티션

하나의 프로세서가 있는 여러 데이터베이스 파티션

이 환경에는 여러 개의 데이터베이스 파티션이 있습니다. 각 데이터베이스 파티션은 해당 소유 머신에 있으며 자체 소유 프로세서, 메모리 및 디스크(94 페이지의 그림 33)를 가집니다. 모든 머신은 통신 기능에 의해 연결됩니다. 이런 환경을 클러스터, 단일 프로세서의 클러스터, 대량 병렬 처리(MPP) 환경 및 비공유 구성이라고도 합니다. 비공유 구성이란 이름을 통해 이 환경에서의 자원 배열 상태를 알 수 있습니다. SMP 환경과는 달리, MPP 환경에서는 메모리나 디스크를 공유하지 않으므로 제한사항이 없습니다.

파티션된 데이터베이스 환경에서 데이터베이스는 둘 이상의 데이터베이스 파티션에 물리적으로 나뉘지만 논리적으로는 하나일 수 있습니다. 따라서 데이터가 분산된 사실을 대부분의 사용자가 명확히 알 수 있습니다. 여러 데이터베이스 관리 프로그램으로 작업을 나눌 수 있으며, 각 데이터베이스 파티션의 각 데이터베이스 관리 프로그램은 자신에게 해당되는 데이터베이스의 부분에 대해 작업합니다.

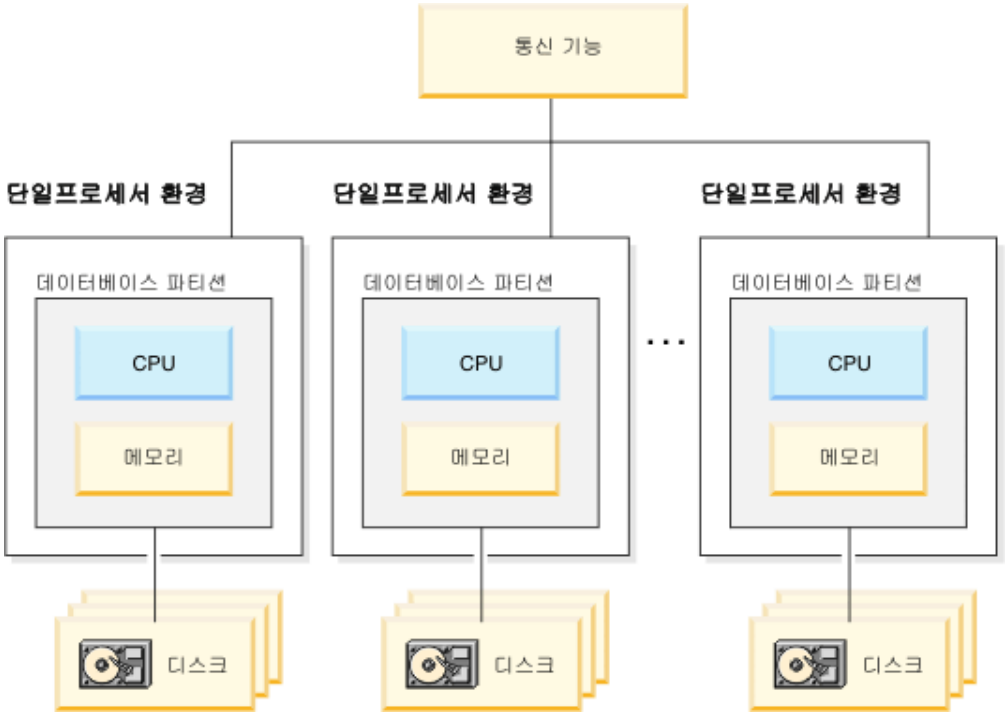


그림 33. 대량 병렬 처리(MPP) 환경

용량 및 확장성

이 환경에서는 더 많은 데이터베이스 파티션(노드)을 구성에 추가할 수 있습니다. 일부 플랫폼에서 최대 노드 수는 512입니다. 그러나 많은 수의 머신과 인스턴스를 관리하는 데는 실질적인 한계가 있습니다.

최대 용량 또는 확장성에 도달하면, 각 데이터베이스 파티션에 다중 프로세서가 있는 시스템으로 이동하는 것이 좋습니다.

다중 프로세서가 있는 시스템의 여러 데이터베이스 파티션

각 데이터베이스 파티션에 단일 프로세서가 있는 구성에 대한 대안은 각 파티션에 다중 프로세서가 있는 구성입니다. 이를 *SMP 클러스터*라고 합니다(95 페이지의 그림 34).

이 구성은 SMP 병렬 처리와 MPP 병렬 처리의 이점을 합한 것입니다. 다중 프로세서를 통해 쿼리가 단일 데이터베이스 파티션에서 실행될 수 있습니다. 또한 다중 데이터베이스 파티션을 통해 쿼리가 병렬로 실행될 수도 있습니다.

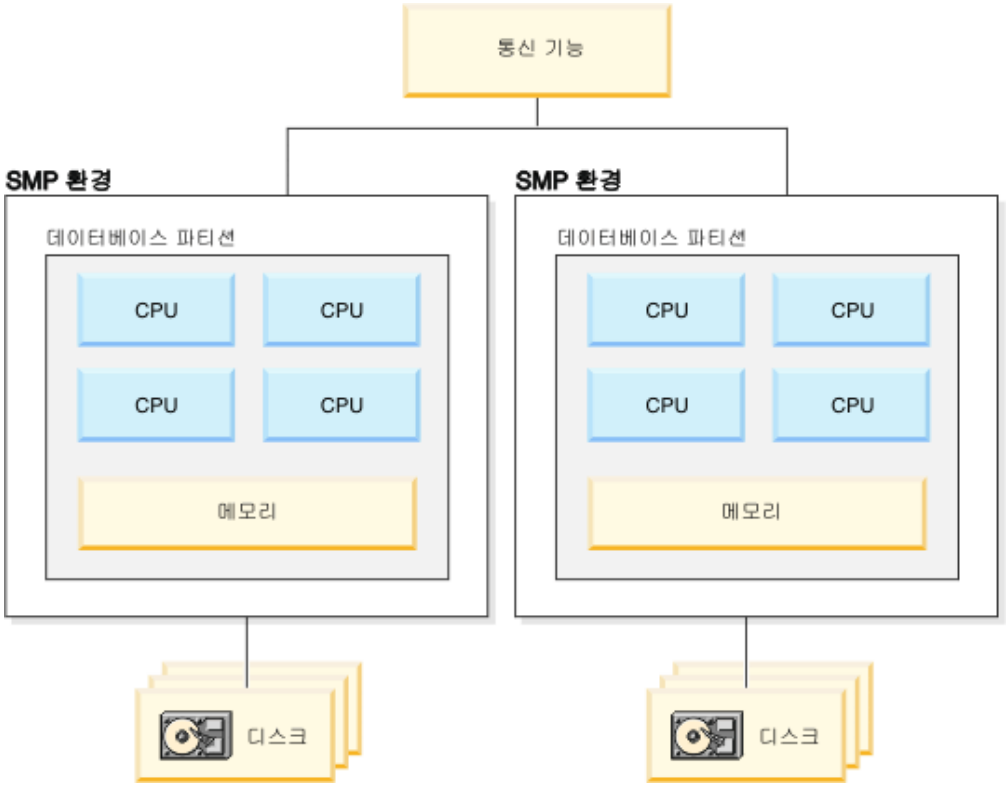


그림 34. 클러스터에 있는 다수의 대칭적 멀티프로세서(SMP) 환경

용량 및 확장성

이 환경에서는 더 많은 데이터베이스 파티션을 추가할 수 있으며, 기존 데이터베이스 파티션에 더 많은 프로세서를 추가할 수 있습니다.

논리 데이터베이스 파티션

논리 데이터베이스 파티션은 전체 머신 제어가 제공되지 않는다는 점에서 물리적 파티션과는 다릅니다. 머신은 자원을 공유하지만, 데이터베이스 파티션은 자원을 공유하지 않습니다. 프로세서는 공유되지만, 디스크와 메모리는 공유되지 않습니다.

논리 데이터베이스 파티션은 확장성을 제공합니다. 다중 논리 파티션에서 실행되는 다중 데이터베이스 관리 프로그램은 단일 데이터베이스 관리 프로그램이 사용하는 것보다 더 완벽하게 사용 가능한 자원을 사용할 수 있습니다. 96 페이지의 그림 35는 더 많은 데이터베이스 파티션을 추가하면 SMP 머신에서 더 많은 확장성을 얻을 수 있으며, 특히 프로세서가 많은 머신의 경우에 확장성을 더 많이 얻을 수 있음을 보여줍니다. 데이터베이스를 분배하면, 각 파티션을 개별적으로 관리하고 복구할 수 있습니다.

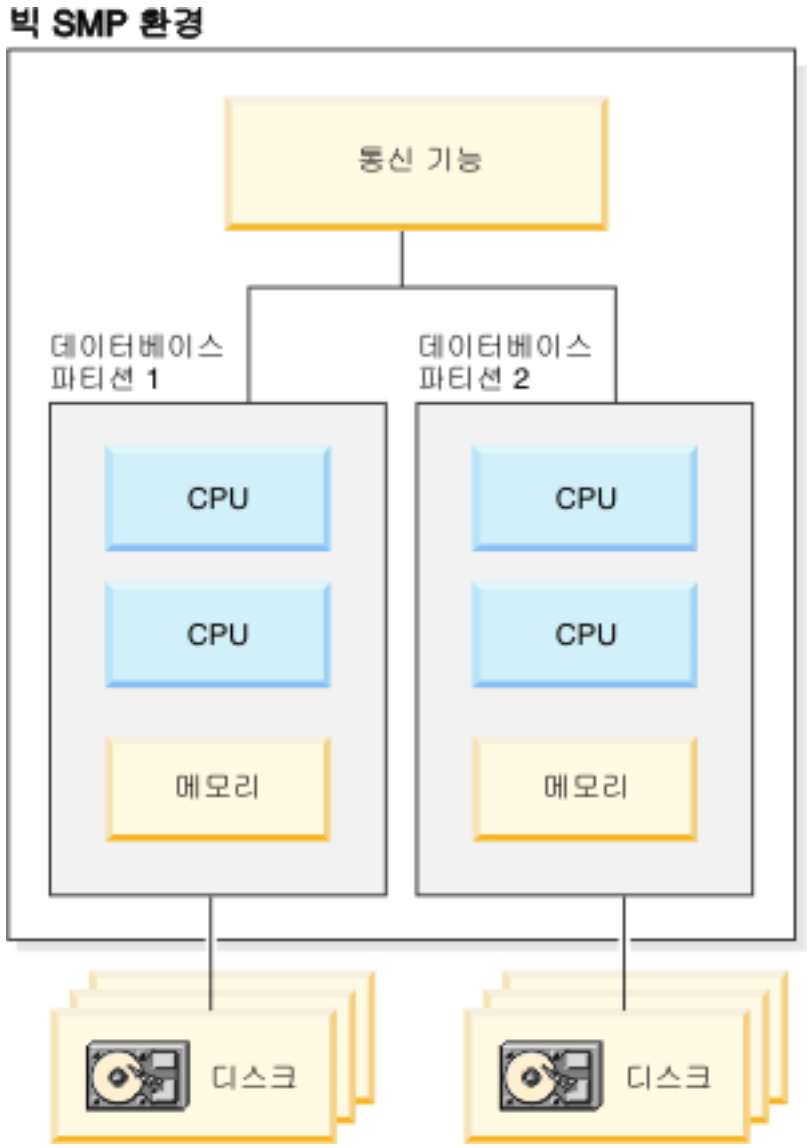


그림 35. 대칭적 멀티 프로세서(SMP) 환경의 파티션된 데이터베이스

97 페이지의 그림 36에서는 그림 35에 표시된 구성을 두 배로 늘려 처리 능력을 증가시킬 수 있다는 것을 보여줍니다.

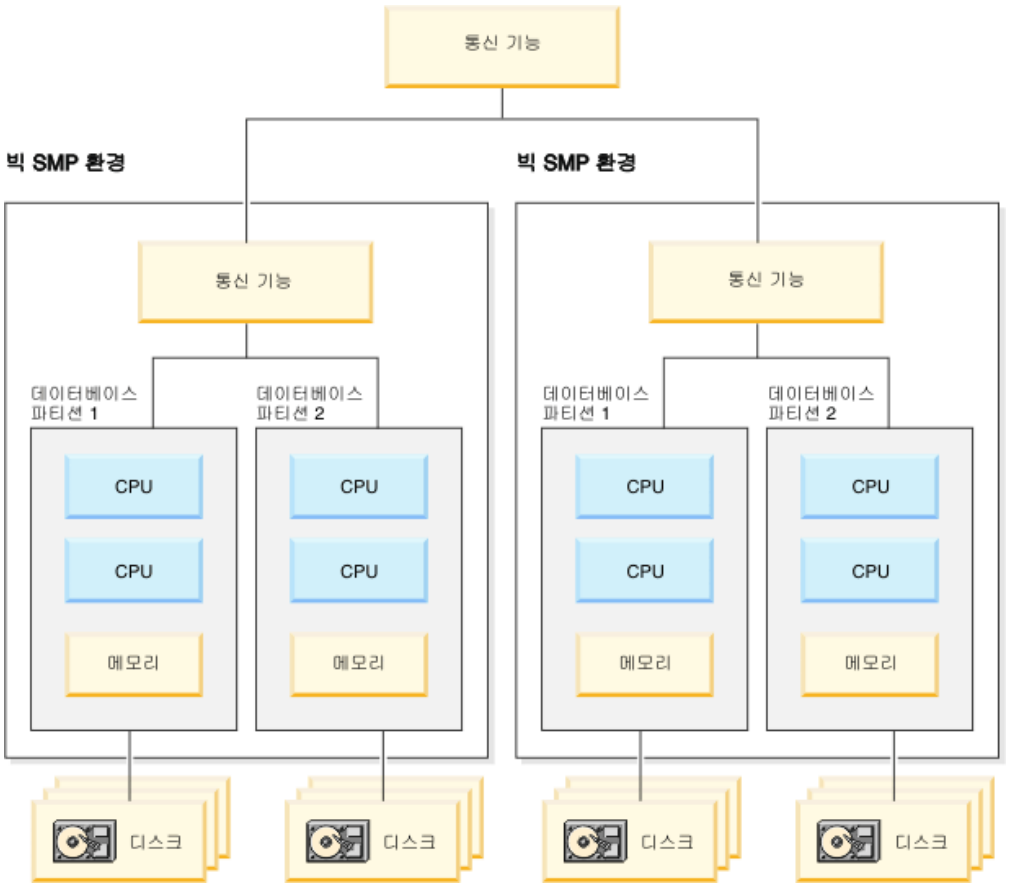


그림 36. 함께 클러스터된 대칭적 멀티프로세서 환경의 파티션된 데이터베이스

주: 둘 이상의 데이터베이스 파티션이 같은 머신에 (프로세서 수에 관계없이) 공존하면 고가용성 구성 및 장애 복구 전략 설계에서 융통성을 높입니다. 머신 고장의 경우, 같은 데이터베이스의 다른 데이터베이스 파티션을 포함하는 2차 머신으로 데이터베이스 파티션이 자동으로 이동되어 재시작될 수 있습니다.

각 하드웨어 환경에 가장 적합한 병렬 처리 요약

다음 표는 다양한 하드웨어 환경에 가장 적합한 병렬 처리의 유형을 요약한 것입니다.

표 8. 각 하드웨어 환경에서 가능한 병렬 처리 유형

하드웨어 환경	입출력 병렬 처리	쿼리 간 병렬 처리	
		파티션 내 병렬 처리	파티션 간 병렬 처리
단일 데이터베이스 파티션, 단일 프로세서	예	아니오(1)	아니오
단일 데이터베이스 파티션, 다중 프로세서(SMP)	예	예	아니오
다중 데이터베이스 파티션, 하나의 프로세서(MPP)	예	아니오(1)	예

표 8. 각 하드웨어 환경에서 가능한 병렬 처리 유형 (계속)

하드웨어 환경	입출력 병렬 처리	쿼리 간 병렬 처리	
		파티션 내 병렬 처리	파티션 간 병렬 처리
다중 데이터베이스 파티션, 멀티 프로세서(SMP 클러스터)	예	예	예
논리 데이터베이스 파티션	예	예	예
주: (1) 단일 시스템에서조차, 특히 실행하는 쿼리가 CPU를 완전히 사용하지 못하는 경우(예: 쿼리가 입출력 바운드일 경우) 병렬 처리 정도를 1보다 큰 값으로 설정하는 것이 좋습니다.			

제 2 부 설치 고려사항

제 5 장 설치 요구사항

DB2 서버 설치(Windows)

이 태스크는 Windows에서 DB2 설치 마법사를 시작하는 방법을 설명합니다. DB2 설치 마법사를 사용하여 설치를 정의하고 시스템에 DB2 데이터베이스 제품을 설치할 수 있습니다.

시작하기 전에

DB2 설치 마법사를 사용하기 전에 다음을 확인하십시오.

- 파티션된 데이터베이스 환경을 설정할 계획이면 ″파티션된 데이터베이스 환경 설정″을 참조하십시오.
- 시스템이 설치, 메모리 및 디스크 요구사항에 맞는지 확인하십시오.
- Windows에서 LDAP을 사용하여 DB2 서버를 Active Directory에 등록하려면 설치하기 전에 디렉토리 스키마를 확장해야 합니다.
- IBM® Tivoli® Monitoring for Databases: DB2 Agent를 사용할 계획인 경우, DB2 제품을 설치하기 전에 세부사항, 한계 및 제한에 관한 내용은 ″DB2 설치 프로그램을 사용한 IBM Tivoli Monitoring for Databases: DB2 Agent 설치″ 주제를 참조하십시오.
- 설치를 수행하려면 권장되는 사용자 권한을 가진 로컬 관리자 어카운트가 있어야 합니다. LocalSystem이 DAS 및 DB2 인스턴스 사용자로 사용되는 DB2 데이터베이스 서버이고, 데이터베이스 파티셔닝 기능을 사용하지 않는 경우, 승격된 특권을 가진 비관리자가 설치를 수행할 수 있습니다.

  주: 비관리자 어카운트가 제품 설치를 수행할 경우, DB2 데이터베이스 제품을 설치하기 전에 VS2005 런타임 라이브러리가 설치되어 있어야 합니다. VS2005 런타임 라이브러리는 DB2 데이터베이스 제품을 설치하기 전에 운영 체제에 필요합니다. VS2005 런타임 라이브러리는 Microsoft® 런타임 라이브러리 다운로드 웹 사이트에 있습니다. 32비트 시스템을 위해 `vcredist_x86.exe`를 선택하거나 64비트 시스템을 위해 `vcredist_x64.exe`를 선택하는 두 가지 선택사항이 있습니다.
- 필수는 아니지만, 설치 프로그램에서 재부트하지 않고 컴퓨터에서 파일을 갱신할 수 있도록 모든 프로그램을 닫을 것을 권장합니다.
- 가상 드라이브 또는 맵핑되지 않은 네트워크 드라이브(예: Windows 탐색기의 ₩₩*hostname*₩*sharename*)에서의 DB2 제품 설치는 지원되지 않습니다. DB2 제품을 설치하기 전에 네트워크 드라이브를 Windows 드라이브 이름(예를 들면 Z:)으로 맵핑해야 합니다.

제한사항

- 사용자 어카운트로 실행 중인 DB2 설치 마법사의 인스턴스를 두 개 이상 가질 수 없습니다.
- DB2 사본 이름 및 인스턴스 이름은 숫자 값으로 시작할 수 없습니다. DB2 사본 이름은 문자 A-Z, a-z 및 0-9로 구성된 64자리 영어 문자로 제한됩니다.
- DB2 사본 이름 및 인스턴스 이름은 모든 DB2 사본 사이에서 고유해야 합니다.
- XML 기능의 사용은 하나의 데이터베이스 파티션만 있는 데이터베이스로 제한됩니다.
- 다음 중 하나가 이미 설치된 경우 동일한 경로에 다른 DB2 데이터베이스 제품을 설치할 수 없습니다.
  - IBM Data Server Runtime Client
  - IBM Data Server Driver Package
  - *DB2 정보 센터*
- DB2 설치 마법사 필드에는 영어가 아닌 문자는 허용되지 않습니다.
- Windows Vista 또는 Windows 2008 이상에서 확장된 보안을 사용하는 경우, 로컬 DB2 명령 및 응용프로그램을 실행하려면 로컬 관리자의 디폴트 특권을 제한하는 추가 보안 기능(사용자 액세스 제어) 때문에 사용자는 DB2ADMNS 또는 DB2USERS 그룹에 속해야 합니다. 사용자가 이러한 그룹 중 하나에 속하지 않으면 로컬 DB2 구성 또는 응용프로그램 데이터에 대한 읽기 권한을 가지지 못합니다.

프로시저

DB2 설치 마법사를 시작하려면 다음을 수행하십시오.

1. DB2 설치에 정의한 로컬 관리자 어카운트로 시스템에 로그온하십시오.
2. DB2 데이터베이스 제품 DVD가 있으면 드라이브에 삽입하십시오. 사용 가능한 경우, 자동 실행 기능이 DB2 설치 런치패드를 자동으로 시작합니다. 자동 실행이 작동하지 않으면 Windows 탐색기를 사용하여 DB2 데이터베이스 제품 DVD를 찾은 후 setup 아이콘을 더블 클릭하여 DB2 설치 런치패드를 시작하십시오.
3. Passport Advantage에서 DB2 데이터베이스 제품을 다운로드한 경우, 실행할 수 있는 파일을 실행하여 DB2 데이터베이스 제품 설치 파일을 추출하십시오. Windows 탐색기를 사용하여 DB2 설치 파일을 찾은 후 setup 아이콘을 더블 클릭하여 DB2 설치 런치패드를 시작하십시오.
4. DB2 설치 런치패드에서 설치 전제조건 및 릴리스 정보를 보거나 바로 설치를 진행할 수 있습니다. 최신 정보에 대한 릴리스 정보와 설치 요구사항을 검토할 수 있습니다.
5. **제품 설치** 및 제품 설치 창을 누르면 설치할 수 있는 제품이 표시됩니다.

컴퓨터에 기존 DB2 데이터베이스 제품이 설치되어 있지 않은 경우, **새로 설치**를 눌러 설치를 시작하십시오. DB2 설치 마법사 프롬프트를 따라 설치를 진행하십시오.

최소한 하나의 기존 DB2 데이터베이스 제품이 컴퓨터에 설치되어 있는 경우, 다음을 수행할 수 있습니다.

- **새로 설치**를 눌러 새 DB2 사본을 작성할 수 있습니다.
- **기존 제품으로 설치**를 눌러 기존 DB2 사본을 업그레이드하고 기존 DB2 사본에 기능을 추가하거나 기존 DB2 버전 8, 버전 9.1, 또는 버전 9.5 사본을 업그레이드하거나 추가(add-on) 제품을 설치하십시오.

6. DB2 설치 마법사는 시스템 언어를 판별하고 해당 언어로 설치 프로그램을 시작합니다. 온라인 도움말을 참조하여 나머지 단계를 진행할 수 있습니다. 온라인 도움말을 호출하려면 **도움말**을 누르거나 **F1**을 누르십시오. **취소**를 누르면 언제든지 설치를 끝낼 수 있습니다.

결과

디폴트로, DB2 데이터베이스 제품은 *Program_Files*₩IBM₩sqllib 디렉토리에 설치되며, 여기서, *Program_Files*는 Program Files 디렉토리의 위치를 나타냅니다.

이 디렉토리가 이미 사용 중인 시스템에서 설치하는 경우, DB2 데이터베이스 제품 설치 경로에는 *_xx*가 추가되며, 여기서, *xx*는 01에서 시작하여 설치된 DB2 사본의 수에 따라 증가합니다.

또한, 자체 DB2 데이터베이스 제품 설치 경로를 지정할 수도 있습니다.

다음 단계

- 설치를 검증하십시오.
- 필요한 설치 후 태스크를 수행하십시오.

설치 중에 발생한 오류에 대한 정보는 My Documents₩DB2LOG₩ 디렉토리에 있는 설치 로그 파일을 검토하십시오. 로그 파일의 형식은 DB2-*ProductAbrrev*-*DateTime*.log(예: DB2-ESE-Tue Apr 04 17_04_45 2008.log)입니다.

Vista 64비트에 새 DB2 제품을 설치하고 32비트 OLE DB Provider를 사용할 경우 IBMDADB2 DLL을 수동으로 등록해야 합니다. 이 DLL을 등록하려면 다음 명령을 실행하십시오.

```
c:₩windows₩SysWOW64₩regsvr32 /s c:₩Program_Files₩IBM₩SQLLIB₩bin₩ibmdadb2.dll
```

여기서 *Program_Files*는 Program Files 디렉토리의 위치를 나타냅니다.

DB2 데이터베이스 제품에서 로컬 컴퓨터 또는 네트워크의 다른 컴퓨터에 있는 DB2 문서에 액세스하도록 하려면 *DB2 정보 센터*를 설치해야 합니다. *DB2 정보 센터*에는

DB2 데이터베이스 시스템 및 DB2 관련 제품에 관한 문서가 포함되어 있습니다. *DB2 정보 센터*가 로컬에 설치되어 있지 않으면 디폴트로 DB2 정보는 웹에서 액세스됩니다.

DB2 Express Edition 및 DB2 Workgroup Server Edition 메모리 한계

DB2 Express Edition 설치 시 인스턴스의 최대 허용 메모리는 4GB입니다.

DB2 Workgroup Server Edition 설치 시 인스턴스의 최대 허용 메모리는 16GB입니다.

인스턴스에 할당되는 메모리 양은 **INSTANCE_MEMORY** 데이터베이스 관리 프로그램 구성 매개변수에 의해 판별됩니다.

버전 9.1 또는 9.5에서 업그레이드 시 중요한 참고사항:

- 버전 9.1 DB2 데이터베이스 제품의 메모리 구성이 허용 한계를 초과하는 경우, 현재 버전으로 업그레이드한 후 DB2 데이터베이스 제품이 시작되지 않습니다.
- 자체 성능 조정 메모리 관리자는 전체 인스턴스 메모리 한계를 라이센스 한계 범위 이상으로 증가시키지 않습니다.

파티션된 DB2 서버의 환경 준비(Windows)

이 주제에서는 DB2 제품의 파티션된 설치에 알맞게 Windows 환경을 준비하는 데 필요한 단계에 대해 설명합니다.

각 참여 컴퓨터는 같은 운영 체제를 가져야 합니다.

설치에 알맞게 Windows 환경을 준비하려면 다음을 수행하십시오.

1. 기본 컴퓨터 및 참여 컴퓨터가 동일한 Windows 도메인에 속하는지 확인하십시오. 제어판을 통해 액세스할 수 있는 시스템 등록 정보 대화 상자를 사용하여 컴퓨터가 속하는 도메인을 확인하십시오.
2. 기본 컴퓨터와 참여 컴퓨터의 시간 및 날짜 설정값이 일치하는지 확인하십시오. 일관성을 고려하여 모든 컴퓨터의 GMT 시간 차가 1시간 이내여야 합니다.

   제어판에서 액세스 가능한 날짜/시간 등록 정보 대화 상자를 사용하여 시스템 날짜 및 시간을 수정할 수 있습니다. 이 제한사항을 변경하려면 `max_time_diff` 구성 매개변수를 사용하십시오. 디폴트값은 `max_time_diff = 60`으로, 이는 60분 미만의 차이를 허용합니다.
3. 파티션된 데이터베이스 환경에 참여하는 각 컴퓨터 오브젝트가 "위임에 대해 컴퓨터 신뢰" 특권이 플래그 표시되어 있는지 확인하십시오. 활성 디렉토리 사용자 및 컴퓨터 콘솔에서 각 컴퓨터의 어카운트 등록 정보 대화 상자의 일반 탭에서 "위임에 대해 컴퓨터 신뢰" 선택란이 선택되어 있음을 확인할 수 있습니다.
4. 모든 컴퓨터가 TCP/IP를 사용하여 서로 통신할 수 있는지 확인하십시오.

a. 한 참여 컴퓨터에서 컴퓨터의 호스트 이름을 리턴하도록 hostname 명령을 입력하십시오.

b. 다른 참여 컴퓨터에서 다음 명령을 입력하십시오.

```
ping hostname
```

여기서 *hostname*은 기본 컴퓨터의 호스트 이름입니다. 테스트에 성공하면 다음과 같은 내용이 출력됩니다.

```
Pinging ServerA.ibm.com [9.21.27.230] with 32 bytes of data:

Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
```

모든 참여 컴퓨터가 TCP/IP를 사용하여 서로 통신할 수 있는지 확인할 때까지 이 단계를 반복하십시오. 각 컴퓨터에는 정적 IP 주소가 있어야 합니다.

다중 네트워크 어댑터를 사용할 계획이라면 데이터베이스 파티션 서버 간의 통신에 사용할 어댑터를 지정할 수 있습니다. 설치 완료 후 db2nchg 명령을 사용하여 db2nodes.cfg 파일에 네트이름 필드를 지정하십시오.

5. 설치 중에 DB2 Administration Server 사용자 어카운트를 제공해야 합니다. 이는 DB2 Administration Server(DAS)에 사용할 로컬 또는 도메인 사용자 어카운트입니다. DAS는 GUI 도구를 지원하고 관리 태스크를 지원하기 위해 사용되는 관리 서비스입니다. 지금 새 사용자를 정의하거나 DB2 설치 마법사를 사용하여 작성할 수 있습니다. DB2 설치 마법사를 사용하여 새 도메인 사용자를 작성할 경우, 설치를 수행하는 데 사용되는 어카운트에 도메인 사용자를 작성할 권한이 있어야 합니다.
6. 인스턴스 소유 파티션을 설치할 기본 컴퓨터에서 사용자는 로컬 *관리자* 그룹에 속하는 도메인 사용자 어카운트를 가져야 합니다. DB2를 설치할 때 이 사용자로 로그온합니다. 각 참여 컴퓨터의 로컬 *관리자* 그룹에 같은 사용자 어카운트를 추가해야 합니다. 이 사용자는 *운영 체제의 일부로 작동* 사용자 권한을 가지고 있어야 합니다.
7. 인스턴스의 모든 컴퓨터가 동일한 로컬 드라이브 이름에 데이터베이스 디렉토리를 갖고 있는지 확인하십시오.GET DATABASE CONFIGURATION 명령을 실행하고 DFTDBPATH DBM 구성 매개변수의 값을 확인하여 이 조건을 확인할 수 있습니다.
8. 설치 중에 DB2 인스턴스에 연관될 도메인 사용자 어카운트를 제공해야 합니다. 모든 DB2 인스턴스에는 하나의 사용자만 지정됩니다. DB2는 인스턴스가 시작될 때 이 사용자 이름을 사용하여 로그온합니다. 지금 사용자를 정의하거나 DB2 설치 마법사를 사용하여 새 도메인 사용자를 작성할 수 있습니다.

파티션된 환경에 새 노드를 추가할 때, DB2 사본 이름은 모든 컴퓨터에서 동일해야 합니다.

DB2 설치 마법사를 사용하여 새 도메인 사용자를 작성할 경우, 설치를 수행하는 데 사용되는 어카운트에 도메인 사용자를 작성할 권한이 있어야 합니다. 인스턴스 사용자 도메인 어카운트는 모든 참여 컴퓨터의 로컬 *관리자* 그룹에 속해야 하며 다음과 같은 사용자 권한을 부여 받습니다.

- 운영 체제의 일부로 활동
- 토큰 오브젝트 작성
- 메모리의 페이지 잠금
- 서비스로서 로그온
- 할당량 증가
- 프로세스 레벨 토큰 바꾸기

확장 보안을 선택할 경우, 어카운트도 DB2ADMNS 그룹의 구성원이어야 합니다. DB2ADMNS 그룹에 이미 이러한 특권이 부여되어 있으므로 특권은 이미 어카운트에 명시적으로 추가되어 있습니다.

FCM(Fast Communication Manager) (Windows)

FCM(Fast Communication Manager)은 동일한 인스턴스에 속하는 DB2 서버 제품에 통신 지원을 제공합니다. 각 데이터베이스 파티션 서버마다 하나의 FCM 송신기 및 하나의 FCM 수신기 디먼이 있어서 데이터베이스 파티션 서버 간에 통신을 제공하여 에이전트 요청을 처리하고 메시지 버퍼를 전달합니다. FCM 디먼은 사용자가 인스턴스를 시작할 때 시작됩니다.

데이터베이스 파티션 서버 사이에 통신에 실패하거나 통신을 다시 설정하는 경우, FCM 스레드는 정보를 갱신합니다. 데이터베이스 시스템 모니터를 사용하여 이 정보를 쿼리할 수 있습니다. FCM 디먼은 또한 적합한 조치를 트리거하기도 합니다. 적합한 조치의 예는 영향 받은 트랜잭션의 롤백입니다. FCM 구성 매개변수 설정시 데이터베이스 시스템 모니터를 사용하면 도움이 됩니다.

fcm_num_buffers 데이터베이스 관리 프로그램 구성 매개변수로 FCM 메시지 버퍼 수를 지정하고 *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수로 FCM 채널 수를 지정할 수 있습니다. *fcm_num_buffers* 및 *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수는 디폴트값으로 AUTOMATIC으로 설정됩니다. 이러한 매개변수 중 하나라도 AUTOMATIC으로 설정되면 FCM은 자원 사용을 모니터하고 점차적으로 자원을 릴리스합니다. 이러한 매개변수는 AUTOMATIC으로 설정한 상태로 남겨두는 것이 좋습니다.

DB2 서버 제품 설치 개요(Linux 및 UNIX)

이 주제에는 AIX®, HP-UX, Linux 및 Solaris에 DB2 서버 제품을 설치하는 단계가 요약되어 있습니다.

DB2 서버 제품을 설치하려면 다음을 수행하십시오.

1. DB2 제품 전제조건을 검토하십시오.
2. 적용 가능한 경우 DB2 업그레이드 정보를 검토하십시오.
3. HP-UX, Linux 및 Solaris에서 커널 매개변수를 수정하십시오. 설치를 계속하기 전에 Linux on x86_32를 제외한 모든 플랫폼에 64비트 커널을 설치해야 합니다. 그렇지 않으면 설치에 실패합니다.
4. 설치 미디어 준비:

   제품 DVD

   DB2 제품 DVD가 자동 마운트되지 않으면 DB2 제품 DVD를 마운트하십시오.

   설치 이미지

   설치 이미지를 다운로드한 경우 파일의 압축을 푸십시오.
5. 사용 가능한 메소드 중 하나를 사용하여 DB2 제품을 설치하십시오.
   - DB2 설치 마법사
   - db2_install 명령
   - 응답 파일을 사용하여 자동 설치
   - 페이로드 파일 전개

   DB2의 경우 DB2 설치 마법사를 사용하여 다음과 같은 설치 및 구성 태스크를 수행할 수 있습니다.
   - DB2 설치 유형(일반, 최소 또는 사용자 정의) 선택
   - DB2 제품 설치 위치 선택
   - 추후에 제품 인터페이스 및 메시지의 디폴트 언어로 지정할 수 있는 언어 설치
   - IBM Tivoli SA MP(System Automation for Multiplatforms) 설치 또는 업그레이드 (Linux 및 AIX)
   - DB2 인스턴스 설정
   - DB2 Administration Server 설정(DAS 사용자 설정 등)
   - DB2 텍스트 검색 서버 설정
   - 관리 담당자 및 Health Monitor 통지 설정
   - 인스턴스 설정 및 구성(인스턴스 사용자 설정 포함)
   - Informix 데이터 소스 지원 설정
   - DB2 도구 카탈로그 준비

- DB2 정보 센터 포트 지정
- 응답 파일 작성

6. DB2 설치 마법사 이외의 방법으로 DB2 서버를 설치한 경우 설치 후 구성 단계가 필요합니다.

DB2 설치 메소드

이 주제에서는 DB2 설치 메소드에 대한 정보를 제공합니다. 다음 표는 운영 체제별로 사용 가능한 설치 메소드입니다.

표 9. 운영 체제별 설치 메소드

설치 메소드	Windows	Linux 또는 UNIX
DB2 설치 마법사	예	예
응답 파일 설치	예	예
db2_install 명령	아니오	예
페이로드 파일 전개	아니오	예

다음 목록에서는 DB2 설치 메소드에 대해 설명합니다.

DB2 설치 마법사

DB2 설치 마법사는 Linux, UNIX 및 Windows 운영 체제에서 사용 가능한 GUI 설치 프로그램입니다. DB2 설치 마법사는 DB2 제품을 설치하고 초기 설정 및 구성 태스크를 수행하는 데 필요한 사용하기 쉬운 인터페이스를 제공합니다.

DB2 설치 마법사는 이 설치를 다른 머신에 복사하는 데 사용할 수 있는 DB2 인스턴스 및 응답 파일을 작성할 수도 있습니다.

주: Linux 및 UNIX 플랫폼에서 비루트 설치의 경우, 단 하나의 DB2 인스턴스만이 존재할 수 있습니다. DB2 설치 마법사는 비루트 설치를 자동으로 작성합니다.

Linux 및 UNIX 플랫폼에서는 DB2 설치 마법사를 표시하려면 X 서버가 필요합니다.

응답 파일 설치

응답 파일은 설치 및 구성 값이 들어 있는 텍스트 파일입니다. DB2 설치 파일이 이 파일을 읽으며 여기에 지정된 값에 따라 설치가 수행됩니다.

응답 파일 설치는 자동 설치로 언급될 수도 있습니다.

응답 파일을 사용하는 또 하나의 장점은 DB2 설치 마법사를 사용하여 설정할 수 없는 매개변수에 대한 액세스를 제공한다는 점입니다.

Linux 및 UNIX 운영 체제에서, 고유 응용프로그램에 DB2 설치 이미지를 임베드한 경우, 응용프로그램에서는 컴퓨터에서 판독 가능한 양식으로 설치 프로

그램에서 설치 진행 정보 및 프롬프트를 수신할 수 있습니다. 이 동작은 INTERACTIVE 응답 파일 키워드에 의해 제어됩니다.

다음과 같은 여러 가지 메소드로 응답 파일을 작성할 수 있습니다.

응답 파일 생성 프로그램 사용

응답 파일 생성 프로그램을 사용하여 기존 설치를 복제하는 응답 파일을 작성할 수 있습니다. 예를 들어, IBM Data Server Client를 설치하고 클라이언트를 완전히 구성한 다음, 응답 파일을 생성하여 클라이언트의 설치 및 구성을 다른 컴퓨터로 복제할 수 있습니다.

DB2 설치 마법사 사용

DB2 설치 마법사는 DB2 설치 마법사가 진행됨에 따라 사용자가 선택하는 선택사항을 기본으로 응답 파일을 작성할 수 있습니다. 선택사항은 시스템의 위치에 저장할 수 있는 응답 파일에 기록됩니다. 파티션된 데이터베이스 설치를 선택할 경우, 두 개의 응답 파일(인스턴스 소유 컴퓨터용 파일과 참여 컴퓨터용 파일)이 생성됩니다.

이 설치 메소드의 이점 중 하나는 설치를 수행하지 않고도 응답 파일을 작성할 수 있다는 점입니다. 이 기능은 DB2 제품을 설치하는 데 필요한 옵션을 캡처하는 데 사용할 수 있습니다. 응답 파일은 나중에 사용자가 지정한 정확한 옵션에 따라 DB2 제품을 설치하는 데 사용할 수 있습니다.

db2cfexp 명령을 사용하여 클라이언트 또는 서버 프로파일을 익스포트하여 클라이언트 또는 서버 구성을 저장한 다음 db2cfimp 명령을 사용하여 프로파일을 쉽게 임포트할 수 있습니다. db2cfexp 명령으로 익스포트된 클라이언트 또는 서버 프로파일은 CLIENT_IMPORT_PROFILE 키워드를 사용하여 응답 파일 설치 중에 임포트할 수도 있습니다.

설치를 수행하고 데이터 소스를 카탈로그화한 후 클라이언트 또는 서버 프로파일을 익스포트해야 합니다.

각 DB2 제품이 제공하는 샘플 응답 파일 사용자 정의

응답 파일 생성 프로그램 또는 DB2 설치 마법사를 사용하여 응답 파일을 작성하는 대신 수동으로 샘플 응답 파일을 수정할 수도 있습니다. 샘플 응답 파일은 DB2 제품 DVD에 제공됩니다. 샘플 응답 파일은 각 제품에 유효한 모든 키워드에 대한 세부사항을 제공합니다.

db2_install 명령(Linux 및 UNIX 플랫폼 전용)

db2_install 명령은 영어 인터페이스 지원을 사용하여 지정한 DB2 제품의 모든 구성요소를 설치합니다. -L 매개변수를 사용하여 지원할 추가 언어를 선택할 수 있습니다. 구성요소를 선택하거나 선택 취소할 수 없습니다.

db2_install 명령을 사용하여 지정된 DB2 제품에 대한 모든 구성요소를 설치하는 경우, 사용자 및 그룹 작성, 인스턴스 작성 또는 구성을 수행하지 않습니다. 이 설치 메소드는 설치 후에 구성이 완료되는 경우 바람직한 메소드입니다. 설치 중에 DB2 제품을 구성하려면 DB2 설치 마법사 사용을 고려하십시오.

Linux 및 UNIX 운영 체제에서, 고유 응용프로그램에 DB2 설치 이미지를 임베드한 경우, 응용프로그램에서는 컴퓨터에서 판독 가능한 양식으로 설치 프로그램에서 설치 진행 정보 및 프롬프트를 수신할 수 있습니다.

이 설치 메소드에서는 제품 파일 전개 후에 수동 구성이 필요합니다.

페이로드 파일 전개(Linux 및 UNIX 전용)

이 메소드는 대부분의 사용자에게 권장되지는 않으나 향상된 설치 메소드입니다. 사용자가 직접 페이로드 파일을 설치해야 합니다. 페이로드 파일은 설치 가능한 구성요소에 대한 모든 파일 및 메타데이터를 포함하는 압축된 tarball입니다.

이 설치 메소드에서는 제품 파일 전개 후에 수동 구성이 필요합니다.

주: DB2 제품 설치는 더 이상 Linux 및 UNIX 플랫폼에서 운영 체제 패키지가 아니므로 결과적으로 설치시 운영 체제 명령을 사용할 수 없습니다. DB2 설치와 인터페이스 및 쿼리에 사용된 기존 스크립트를 변경해야 합니다.

DB2 설치 마법사를 사용하여 DB2 서버 설치(Linux 및 UNIX)

이 태스크는 Linux 및 UNIX 운영 체제에서 DB2 설치 마법사를 시작하는 방법을 설명합니다. DB2 설치 마법사는 설치 환경 설정을 정의하고 시스템에 DB2 데이터베이스 제품을 설치하는 데 사용됩니다.

시작하기 전에

DB2 설치 마법사를 사용하기 전에 다음을 확인하십시오.

- 파티션된 데이터베이스 환경을 설정하려면 이 주제의 맨 아래에 있는 관련 링크에 따라 이를 수행하는 방법을 참조하십시오.
- 시스템이 설치, 메모리 및 디스크 요구사항에 맞는지 확인하십시오.
- 지원되는 브라우저가 설치되어 있는지 확인하십시오(Firefox 2.0+, Mozilla 1.7+ 또는 SeaMonkey 1.1.4).
- 루트 또는 비루트 권한을 사용하여 DB2 서버를 설치할 수 있습니다. 비루트 설치에 관한 자세한 정보는 관련 링크를 참조하십시오.
- DB2 데이터베이스 제품 이미지를 사용할 수 있어야 합니다. DB2 설치 이미지는 실제 DB2 데이터베이스 제품 DVD를 구매하거나 Passport Advantage®에서 설치 이미지를 다운로드하여 얻을 수 있습니다.
- 비영어 버전의 DB2 데이터베이스 제품을 설치할 경우, 적절한 자국어 패키지가 있어야 합니다.

- DB2 설치 마법사는 그래프 형식 설치 프로그램입니다. 머신에서 DB2 설치 마법사를 실행하려면 그래픽 사용자 인터페이스를 렌더링할 수 있는 X Windows 소프트웨어가 있어야 합니다. X Windows 서버가 실행 중인지 확인하십시오. 표시 화면을 적절하게 익스포트했는지 확인하십시오. 예를 들면,
  `export DISPLAY=9.26.163.144:0.`입니다.
- 보안 소프트웨어를 사용하는 환경에서는 DB2 설치 마법사를 시작하기 전에 수동으로 필수 DB2 사용자를 작성해야 합니다.
- IBM Tivoli Monitoring for Databases: DB2 Agent를 사용할 계획인 경우, DB2 제품을 설치하기 전에 세부사항, 한계 및 제한에 관한 내용은 ″DB2 설치 프로그램을 사용한 IBM Tivoli Monitoring for Databases: DB2 Agent 설치″ 주제를 참조하십시오.

제한사항

- 사용자 어카운트로 실행 중인 DB2 설치 마법사의 인스턴스를 두 개 이상 가질 수 없습니다.
- XML 기능의 사용은 코드 세트 UTF-8로 정의되어 있고 하나의 데이터베이스 파티션만 있는 데이터베이스로 제한됩니다.
- DB2 설치 마법사 필드에는 영어가 아닌 문자는 허용되지 않습니다.

프로시저

DB2 설치 마법사를 시작하려면 다음을 수행하십시오.

1. 실제 DB2 데이터베이스 제품 DVD가 있는 경우, 다음 명령을 입력하여 DB2 데이터베이스 제품 DVD가 마운트되어 있는 디렉토리로 변경하십시오.

   ```
   cd /dvdrom
   ```

   여기서, */dvdrom*은 DB2 데이터베이스 제품 DVD의 마운트 지점을 나타냅니다.
2. DB2 데이터베이스 제품 이미지를 다운로드한 경우, 제품 파일을 추출하고 untar해야 합니다.
   a. 제품 파일을 추출하십시오.

      ```
      gzip -d product.tar.gz
      ```

      여기서, *product*는 다운로드한 제품의 이름입니다.
   b. 제품 파일을 Untar하십시오.

      Linux 운영 체제

      ```
      tar -xvf product.tar
      ```

      AIX, HP-UX 및 Solaris 운영 체제

      ```
      gnutar -xvf product.tar
      ```

      여기서, *product*는 다운로드한 제품의 이름입니다.

c. 디렉토리를 변경하십시오.

```
cd ./product
```

여기서, *product*는 다운로드한 제품의 이름입니다.

주: 자국어 패키지를 다운로드한 경우에는 이를 동일한 디렉토리에 압축을 푸십시오. 그러면 동일한 디렉토리에 서브디렉토리(예: `./nlpack`)가 작성되고 설치 프로그램이 프롬프트 없이 자동으로 설치 이미지를 찾을 수 있습니다.

3. 데이터베이스 제품 이미지가 있는 디렉토리에서 ./db2setup 명령을 입력하여 DB2 설치 마법사를 시작하십시오.
4. IBM DB2 설치 런치패드가 열립니다. 이 창에서 설치 요구사항 및 릴리스 정보를 보거나 바로 설치를 진행할 수 있습니다. 또한, 최신 정보에 대한 설치 요구사항 및 릴리스 정보를 검토해야 합니다.
5. **제품 설치** 및 **제품 설치** 창을 누르면 설치할 수 있는 제품이 표시됩니다.

   새로 설치를 눌러 설치를 실행하십시오. DB2 설치 마법사 프롬프트를 따라 설치를 진행하십시오.

   설치를 시작한 후 DB2 설치 마법사의 설치 패널에서 항목을 선택하십시오. 설치 도움말을 사용하여 나머지 단계를 진행할 수 있습니다. 설치 도움말을 호출하려면 **도움말**을 누르거나 F1을 누르십시오. **취소**를 누르면 언제든지 설치를 끝낼 수 있습니다.

결과

비루트 설치의 경우, DB2 데이터베이스 제품은 항상 *$HOME*`/sqllib` 디렉토리에 설치되는데, 여기서 *$HOME*은 비루트 사용자의 홈 디렉토리입니다.

루트 설치의 경우, 디폴트로 DB2 데이터베이스 제품은 다음 디렉토리 중 하나에 설치됩니다.

AIX, HP-UX 및 Solaris
/opt/IBM/db2/V9.7

Linux /opt/ibm/db2/V9.7

이 디렉토리가 이미 사용 중인 시스템에 설치하는 경우, DB2 데이터베이스 제품 설치 경로에는 *_xx*가 추가되는데, 여기서, *_xx*는 01에서 시작하여 설치한 DB2 사본의 수에 따라 증가하는 숫자입니다.

또한, 자체 DB2 데이터베이스 제품 설치 경로를 지정할 수도 있습니다.

DB2 설치 경로의 규칙은 다음과 같습니다.

- 소문자 글자(a–z), 대문자 글자(A–Z) 및 밑줄 문자(_)를 사용할 수 있어야 함

- 128자 미만이어야 함
- 스페이스를 포함할 수 없음
- 영어가 아닌 문자를 포함할 수 없음

설치 로그 파일은 다음과 같습니다.

- DB2 설치 로그 파일. 이 파일은 오류를 포함한 모든 DB2 설치 정보를 캡처합니다.
  - 루트 설치의 경우, DB2 설치 로그 파일 이름은 db2setup.log입니다.
  - 비루트 설치의 경우, DB2 설치 로그 파일 이름은 db2setup_*username*.log입니다. 여기서, *username*은 설치 수행 시 사용한 비루트 사용자 ID입니다.
- DB2 오류 로그 파일. 이 파일은 Java™에서 리턴되는 모든 오류 출력(예: 예외 및 트랩 정보)을 캡처합니다.
  - 루트 설치의 경우, DB2 오류 로그 파일 이름은 db2setup.err입니다.
  - 비루트 설치의 경우, DB2 오류 로그 파일 이름은 db2setup_*username*.err입니다. 여기서, *username*은 설치 수행 시 사용한 비루트 사용자 ID입니다.

디폴트로 이러한 로그 파일은 /tmp 디렉토리에 위치합니다. 로그 파일의 위치를 지정할 수 있습니다.

더 이상 db2setup.his 파일은 존재하지 않습니다. 대신 DB2 설치 프로그램은 DB2 설치 로그 파일의 사본을 DB2_DIR/install/logs/ 디렉토리에 저장한 후 db2install.history로 이름을 바꿉니다. 이름이 이미 존재하는 경우, DB2 설치 프로그램은 db2install.history.*xxxx*로 이름을 바꿉니다. 여기서, *xxxx*는 해당 머신에 있는 설치의 수에 따라 0000-9999의 숫자입니다.

각 설치 사본마다 별도의 실행기록 파일 목록을 갖고 있습니다. 설치 사본이 제거되면, 이 설치 경로 아래의 실행기록 파일도 제거됩니다. 이 복사 조치는 설치가 거의 종료되는 시점에 수행되며, 완료 전에 프로그램이 중지 또는 중단될 경우 실행기록 파일이 작성되지 않습니다.

다음 단계

- 설치를 검증하십시오.
- 필요한 설치 후 태스크를 수행하십시오.

자국어 팩은 DB2 데이터베이스 제품을 설치한 후에 자국어 팩이 상주하는 디렉토리에서 ./db2setup 명령을 실행하여 설치할 수도 있습니다.

Linux x86의 경우, DB2 데이터베이스 제품에서 로컬 컴퓨터 또는 네트워크의 다른 컴퓨터에 있는 DB2 문서에 액세스하도록 하려면 *DB2 정보 센터*를 설치해야 합니다. *DB2 정보 센터*에는 DB2 데이터베이스 시스템 및 DB2 관련 제품에 관한 문서가 포함되어 있습니다.

DB2 Express Edition 및 DB2 Workgroup Server Edition 메모리 한계

DB2 Express Edition 설치 시 인스턴스의 최대 허용 메모리는 4GB입니다.

DB2 Workgroup Server Edition 설치 시 인스턴스의 최대 허용 메모리는 16GB입니다.

인스턴스에 할당되는 메모리 양은 **INSTANCE_MEMORY** 데이터베이스 관리 프로그램 구성 매개변수에 의해 판별됩니다.

버전 9.1 또는 9.5에서 업그레이드 시 중요한 참고사항:

- 버전 9.1 또는 9.5 DB2 데이터베이스 제품의 메모리 구성이 허용 한계를 초과하는 경우, 현재 버전으로 업그레이드한 후 DB2 데이터베이스 제품이 시작되지 않습니다.
- 자체 성능 조정 메모리 관리자는 전체 인스턴스 메모리 한계를 라이센스 한계 범위 이상으로 증가시키지 않습니다.

FCM(Fast Communication Manager) (Linux 및 UNIX)

FCM(Fast Communication Manager)은 데이터베이스 파티션 기능(DPF)을 사용하는 DB2 서버 제품에 통신 지원을 제공합니다.

다중 파티션 인스턴스의 경우, 각 데이터베이스 파티션 서버마다 하나의 FCM 송신기 디먼 및 하나의 FCM 수신기 디먼이 있어서 데이터베이스 파티션 서버 간에 통신을 제공하여 에이전트 요청을 처리하고 메시지 버퍼를 전달합니다. FCM 디먼은 사용자가 다중 파티션 인스턴스를 시작할 때 시작됩니다.

데이터베이스 파티션 서버간에 통신이 실패하거나, 통신을 재설정하는 경우에는 FCM 디먼이 정보를 갱신합니다. 데이터베이스 시스템 모니터를 사용하여 이 정보를 쿼리할 수 있습니다. FCM 디먼은 또한 적합한 조치를 트리거하기도 합니다. 적합한 조치의 예는 영향 받은 트랜잭션의 롤백입니다. FCM 구성 매개변수 설정 시 데이터베이스 시스템 모니터를 사용하면 도움이 됩니다.

fcm_num_buffers 데이터베이스 관리 프로그램 구성 매개변수로 FCM 메시지 버퍼 수를 지정할 수 있습니다. 또한, *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수로 FCM 채널 수를 지정할 수도 있습니다. *fcm_num_buffers* 및 *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수는 디폴트값으로 AUTOMATIC으로 설정됩니다. 이러한 매개변수 중 하나라도 AUTOMATIC으로 설정되면 FCM은 자원 사용을 모니터하고 점차적으로 자원을 릴리스합니다. 이러한 매개변수는 AUTOMATIC으로 설정한 상태로 남겨두는 것이 좋습니다.

제 6 장 설치 전

추가 파티션된 데이터베이스 환경 사전 설치 태스크(Linux 및 UNIX)

파티션된 DB2 설치에 대한 환경 설정 갱신(AIX)

이 태스크에서 파티션된 데이터베이스 시스템에 참여할 각 컴퓨터에서 갱신해야 하는 환경 설정값을 설명합니다.

AIX 환경 설정을 갱신하려면 다음을 수행하십시오.

1. 루트 권한을 가진 사용자로 컴퓨터에 로그인하십시오.
2. 다음 명령을 입력하여 AIX maxuproc(사용자당 최대 프로세스 수) 디바이스 속성을 *4096*으로 설정하십시오.

   ```
   chdev -l sys0 -a maxuproc='4096'
   ```

   주: 다른 이미지가 실행 중인 경우 bosboot/reboot를 64비트 커널로 전환해야 합니다.
3. 파티션된 데이터베이스 시스템에 참여하는 모든 워크스테이션의 TCP/IP 네트워크 매개변수를 다음 값으로 설정하십시오. 이러한 값이 이 매개변수에 대한 최소값입니다. 네트워크 관련 매개변수가 이미 더 높은 값으로 설정되어 있는 경우에는 변경하지 마십시오.

   ```
   thewall       = 65536
   sb_max        = 1310720
   rfc1323       = 1
   tcp_sendspace = 221184
   tcp_recvspace = 221184
   udp_sendspace = 65536
   udp_recvspace = 65536
   ipqmaxlen     = 250
   somaxconn     = 1024
   ```

   모든 네트워크 관련 매개변수의 현재 설정값을 나열하려면 다음 명령을 입력하십시오.

   ```
   no -a | more
   ```

   매개변수를 설정하려면 다음 명령을 입력하십시오.

   ```
   no -o parameter_name=value
   ```

   각 부분에 대한 설명은 다음과 같습니다.

   - *parameter_name*은 설정할 매개변수입니다.
   - *value*는 이 매개변수에 설정할 값입니다.

예를 들어, tcp_sendspace 매개변수를 221184로 설정하려면 다음 명령을 입력하십시오.

```
no -o tcp_sendspace=221184
```

4. 신속한 상호 연결을 사용 중이라면 *css0*에 대한 *spoolsize* 및 *rpoolsize*를 다음 값으로 설정해야 합니다.

   ```
   spoolsize     16777216
   rpoolsize     16777216
   ```

   이 매개변수의 현재 설정값을 나열하려면 다음 명령을 입력하십시오.

   ```
   lsattr -l css0 -E
   ```

   이 매개변수를 설정하려면 다음 명령을 입력하십시오.

   ```
   /usr/lpp/ssp/css/chgcss -l css0 -a spoolsize=16777216
   /usr/lpp/ssp/css/chgcss -l css0 -a rpoolsize=16777216
   ```

   시스템을 조정할 때 /tftpboot/tuning.cst 파일을 사용하지 않을 경우, DB2DIR/misc/rc.local.sample 샘플 스크립트 파일을 사용하여 설치 후 네트워크 관련 매개변수를 갱신할 수 있습니다. 여기서 DB2DIR은 DB2 제품이 설치된 경로입니다. 설치 후 이 샘플 스크립트 파일을 사용하여 네트워크 관련 매개변수를 갱신하려면 다음 단계를 수행하십시오.

   a. 다음 명령을 입력하여, 이 스크립트 파일을 /etc 디렉토리로 복사하여 루트로 실행할 수 하십시오.

      ```
      cp /usr/opt/db2_09_01/misc/rc.local.sample /etc/rc.local
      chown root:sys /etc/rc.local
      chmod 744 /etc/rc.local
      ```

   b. /etc/rc.local 파일을 검토하고 필요하다면 이를 갱신하십시오.

   c. 머신이 재부트될 때마다 /etc/rc.local 스크립트가 실행되도록 /etc/inittab 파일에 항목을 추가하십시오. mkitab 명령을 사용하여 /etc/inittab 파일에 항목을 추가할 수 있습니다. 이 항목을 추가하려면 다음 명령을 입력하십시오.

      ```
      mkitab "rclocal:2:wait:/etc/rc.local > /dev/console 2>&1"
      ```

   d. 다음 명령을 입력하여 /etc/inittab 파일에 /etc/rc.nfs 항목이 포함되어 있는지 확인하십시오.

      ```
      lsitab rcnfs
      ```

   e. 다음 명령을 입력하여 시스템을 재부팅하지 않고 네트워크 매개변수를 갱신하십시오.

      ```
      /etc/rc.local
      ```

5. DB2 ESE의 파티션된 설치를 실행하기에 충분한 페이징 스페이스가 있는지 확인하십시오. 페이징 스페이스가 충분하지 않은 경우, 운영 체제는 가장 많은 가상 메모리를 사용하는 프로세스를 종료합니다(즉 DB2 프로세스 중 하나일 가능성이 큼). 사용가능한 페이징 스페이스를 확인하려면 다음 명령을 입력하십시오.

```
lsps -a
```

이 명령은 다음과 유사한 출력을 표시합니다.

```
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
   paging00    hdisk1            rootvg          60MB      19     yes   yes    lv
   hd6         hdisk0            rootvg          60MB      21     yes   yes    lv
   hd6         hdisk2            rootvg          64MB      21     yes   yes    lv
```

컴퓨터에 설치된 실제 메모리 양의 두 배 정도를 페이징 공간으로 사용해야 합니다.

6. 소형에서 중간 크기 정도의 파티션된 데이터베이스 시스템을 작성할 경우에는 인스턴스 소유 컴퓨터의 네트워크 파일 시스템 디먼(NFSD) 수가 다음에 근접해야 합니다.

   컴퓨터 상의 biod 수 × 인스턴스 내의 컴퓨터 수

   이상적으로는, 각 컴퓨터에서 10개의 biod 프로세스를 실행해야 합니다. 위의 공식에 따라 10개의 biod 프로세스가 있는 네 개의 컴퓨터 시스템에서는 40개의 NFSD를 사용합니다.

   대형 시스템을 설치할 경우에는 컴퓨터상에 최대 120개의 NFSD가 있을 수 있습니다.

   NFS에 대한 자세한 내용은 해당 NFS 문서를 참조하십시오.

ESE 워크스테이션으로 명령을 분배하기 위한 작업 집합 설정(AIX)

AIX의 파티션된 데이터베이스 환경에서 파티션된 데이터베이스 시스템에 참여하는 워크스테이션 세트에 명령을 분배하기 위해 작업 집합을 설정할 수 있습니다. dsh 명령으로 명령을 워크스테이션에 분배할 수 있습니다.

이 기능을 사용하면 AIX에서 파티션된 데이터베이스 시스템을 설치 또는 관리할 때 오류 발생을 줄여 사용자 환경의 모든 컴퓨터에서 동일한 명령을 신속하게 실행할 수 있습니다.

작업 집합에 포함시키려는 각 컴퓨터의 호스트 이름을 알아야 합니다.

루트 권한이 있는 사용자로 CWS(Control Workstation)에 로그온해야 합니다.

파티션된 데이터베이스 시스템에 참여할 모든 워크스테이션에 대한 호스트 이름 목록을 가진 파일을 준비하십시오. 워크스테이션 목록에 따라 명령을 분배시키도록 작업 집합을 설정하려면 다음과 같이 수행하십시오.

1. 작업 집합에 참여할 모든 워크스테이션에 대한 호스트 이름을 나열하는 eeelist.txt 파일을 작성하십시오.

예를 들어, workstation1 및 workstation2라는 두 개의 워크스테이션을 가진 작업 집합을 작성한다고 가정하십시오. eeelist.txt의 내용은 다음과 같습니다.

```
workstation1
workstation2
```

2. 작업 집합 환경 변수를 갱신하십시오. 이 목록을 갱신하려면 다음 명령을 입력하십시오.

   ```
   export WCOLL=path/eeelist.txt
   ```

   여기서, *path*는 *eeelist.txt*가 작성된 위치이고, *eeelist.txt*는 작업 집합의 워크스테이션을 나열하도록 작성한 파일의 이름입니다.

3. 다음 명령을 입력하여 작업 집합내의 이름이 원하는 워크스테이션이 맞는지 확인하십시오.

   ```
   dsh -q
   ```

   다음과 유사한 출력이 표시됩니다.

   ```
   Working collective file /eeelist.txt:
   workstation1
   workstation2
   Fanout: 64
   ```

NFS가 실행 중인지 확인(Linux 및 UNIX)

데이터베이스 파티션 환경을 설치하기 전에 파티션된 데이터베이스 시스템에 참여할 각 컴퓨터에서 NFS(Network File System)가 실행되고 있는지 확인해야 합니다.

각 컴퓨터에서 NFS가 실행 중이어야 합니다.

각 컴퓨터에서 NFS가 실행 중인지 확인하려면 다음을 수행하십시오.

AIX 운영 체제

각 컴퓨터에서 다음 명령을 입력하십시오.

```
lssrc -g nfs
```

NFS 프로세스의 Status 필드는 active로 표시되어야 합니다.

각 시스템에서 NFS가 실행 중인지 확인했으면 DB2 제품에 필요한 특정 NFS 프로세스를 점검하십시오. 필수 프로세스는 다음과 같습니다.

```
rpc.lockd
rpc.statd
```

HP-UX 및 Solaris 운영 체제

각 컴퓨터에서 다음 명령을 입력하십시오.

```
showmount -e hostname
```

로컬 시스템을 점검하려면 *hostname* 매개변수 없이 showmount 명령을 입력하십시오.

NFS가 활성화되어 있지 않은 경우, 다음과 같은 메시지를 수신합니다.

```
showmount: ServerA: RPC: Program not registered
```

각 시스템에서 NFS가 실행 중인지 확인했으면 DB2 제품에 필요한 특정 NFS 프로세스를 점검하십시오.

```
rpc.lockd
rpc.statd
```

다음 명령을 사용하여 이러한 프로세스를 점검할 수 있습니다.

```
ps -ef | grep rpc.lockd
ps -ef | grep rpc.statd
```

Linux 운영 체제

각 컴퓨터에서 다음 명령을 입력하십시오.

```
showmount -e hostname
```

로컬 시스템을 점검하려면 *hostname* 매개변수 없이 showmount 명령을 입력하십시오.

NFS가 활성화되어 있지 않은 경우, 다음과 같은 메시지를 수신합니다.

```
showmount: ServerA: RPC: Program not registered
```

각 시스템에서 NFS가 실행 중인지 확인했으면 DB2 제품에 필요한 특정 NFS 프로세스를 점검하십시오. 필수 프로세스는 `rpc.statd`입니다.

ps -ef | grep rpc.statd 명령을 사용하여 이 프로세스에 대해 점검할 수 있습니다.

이 두 프로세스가 실행 중이지 않은 경우, 운영 체제 문서를 참고하십시오.

참여 컴퓨터에서 포트 범위 사용 가능성 확인(Linux 및 UNIX)

이 태스크에서는 참여 컴퓨터의 포트 범위 사용 가능성을 확인하는 데 필요한 단계를 설명합니다. 포트 범위는 FCM(Fast Communication Manager)에서 사용합니다. FCM은 데이터베이스 파티션 서버 간의 통신을 조절하는 DB2 기능입니다.

참여 컴퓨터 포트 범위 사용 가능성 확인은 인스턴스 소유 데이터베이스 파티션 서버를 설치한 후에, 그리고 참가하는 데이터베이스 파티션 서버를 설치하기 전에 실행해야 합니다.

인스턴스 소유 데이터베이스 파티션 서버를 기본 컴퓨터에 설치할 경우, DB2는 파티션된 데이터베이스 환경에 참여하는 지정된 논리적 데이터베이스 파티션 서버 수에 따라 포트 범위를 예약합니다. 디폴트 범위는 네 개의 포트입니다. 파티션된 데이터베이스 환

경에 참여하는 각 서버의 경우, FCM 포트에 대해 /etc/services 파일을 수동으로 구성해야 합니다. FCM 포트의 범위는 참여하는 컴퓨터에서 사용할 논리적 파티션 수에 따라 다릅니다. 최소 두 개의 항목 **DB2_<instance>** 및 **DB2_<instance>_END**가 필요합니다. 참여 컴퓨터에 지정되는 FCM 포트의 다른 요구사항은 다음과 같습니다.

- 시작 포트 번호는 기본 컴퓨터의 시작 포트 번호와 일치해야 합니다.
- 후속 포트는 순차적으로 번호가 지정되어야 합니다.
- 지정된 포트 번호는 사용 가능해야 합니다.

서비스 파일을 변경하려면 루트 권한이 필요합니다.

참여 컴퓨터에서의 포트 범위 사용 가능성을 확인하려면 다음을 수행하십시오.

1. /etc/services 디렉토리에 있는 서비스 파일을 여십시오.
2. DB2 FCM(Fast Communication Manager)용으로 예약된 포트를 찾으십시오. 항목은 다음과 같아야 합니다.

```
        DB2_db2inst1         60000/tcp
        DB2_db2inst1_1       60001/tcp
        DB2_db2inst1_2       60002/tcp
    DB2_db2inst1_END 60003/tcp
```

   DB2는 60000 다음의 사용 가능한 처음 네 개의 포트를 예약합니다.
3. 각 참여 컴퓨터에서 서비스 파일을 열고 기본 컴퓨터의 service 파일에서 DB2 FCM용으로 예약된 포트가 사용 중이 아닌지 확인하십시오.
4. 참여 컴퓨터에서 필요한 포트를 사용 중인 경우, 모든 컴퓨터에 사용 가능한 포트 범위를 식별하고 기본 컴퓨터의 서비스 파일을 포함하여 각 서비스 파일을 갱신하십시오.

인스턴스 소유 데이터베이스 파티션 서버를 기본 컴퓨터에 설치한 후 참여 데이터베이스 파티션 서버에 DB2 제품을 수동으로 설치해야 합니다. 파티션된 서버에 대해 생성된 응답 파일(디폴트 이름은 db2ese_addpart.rsp)을 사용할 수 있으며 FCM 포트에 대해 /etc/services 파일을 수동으로 구성해야 합니다. FCM 포트의 범위는 현재 머신에서 사용할 논리적 파티션 수에 따라 다릅니다. 최소 항목은 연속하여 사용 가능한 포트 번호가 있는 DB2_ 및 DB2__END의 두 개 항목입니다. 참여하는 각 머신에서 사용되는 FCM 포트 번호에는 동일한 시작 포트 번호가 있어야 하며 후속 포트는 순차적으로 번호가 매겨져야 합니다.

파티션된 DB2 서버에 대한 파일 시스템 작성(Linux)

이 태스크는 파티션된 데이터베이스 시스템을 설정하는 단계의 일부입니다. 이 태스크에서는 다음 작업을 수행하는 방법에 대해 설명합니다.

- DB2 홈 파일 시스템 작성

- 홈 파일 시스템 NFS 익스포트
- 참여하는 각 컴퓨터로부터 홈 파일 시스템 NFS 마운트

파티션된 데이터베이스 시스템에 참여할 모든 머신에서 사용할 수 있는 파일 시스템이 있어야 합니다. 이 파일 시스템은 인스턴스 홈 디렉토리로 사용됩니다.

단일 데이터베이스 인스턴스에 대해 둘 이상의 머신을 사용하는 구성의 경우, NFS(Network File System)는 이 파일 시스템을 공유하는 데 사용됩니다. 일반적으로, 클러스터의 한 머신은 NFS를 사용하여 파일 시스템을 익스포트하는 데 사용되며, 클러스터의 나머지 머신은 이 머신에서 NFS 파일 시스템을 마운트합니다. 파일 시스템을 익스포트하는 머신에는 로컬로 마운트된 파일 시스템이 있습니다.

자세한 명령 정보는 Linux 분산 문서를 참조하십시오.

파일 시스템을 작성하려면 다음과 같이 수행하십시오.

1. 한 머신에서, 디스크 파티션을 선택하거나 fdisk를 사용하여 디스크 파티션을 작성하십시오.
2. mkfs와 같은 유틸리티를 사용하여 이 파티션에서 파일 시스템을 작성하십시오. 파일 시스템은 필수 DB2 프로그램 파일과 데이터베이스가 필요로 하는 스페이스를 포함할 수 있도록 충분히 커야 합니다.
3. 작성된 파일 시스템을 로컬로 마운트하고, 시스템이 재부트될 때마다 이 파일 시스템이 마운트되도록 /etc/fstab 파일에 항목을 추가하십시오. 예를 들어, 다음과 같습니다.

   ```
   /dev/hda1    /db2home    ext3   defaults   1 2
   ```

4. 부트 시 Linux에서 NFS 파일 시스템을 자동으로 익스포트하려면 항목을 /etc/exports 파일에 추가하십시오. 클러스터에 참여하는 모든 호스트 이름과 머신이 가질 수 있는 모든 이름을 포함시켜야 합니다. 또한 "루트" 옵션을 사용하여 클러스터의 각 머신이 익스포트된 파일 시스템에 대해 루트 권한을 갖도록 해야 합니다.

   /etc/exports 파일은 다음과 같은 정보 유형을 포함한 ASCII 파일입니다.

   ```
   /db2home machine1_name(rw) machine2_name(rw)
   ```

   NFS 디렉토리를 익스포트하려면 다음 명령을 실행하십시오.

   ```
   /usr/sbin/exports -r
   ```

5. 클러스터의 나머지 각 머신에서, /etc/fstab 파일에 항목을 추가하여 부트 시 파일 시스템을 자동으로 NFS 마운트하십시오. 다음 예와 같이, 마운트 지점 옵션을 지정할 때 파일 시스템이 부트 시 마운트되고 읽기/쓰기가 가능하며 하드 마운트되었는지 확인하고 bg(백그라운드) 옵션을 포함하며 setuid 프로그램이 제대로 실행될 수 있는지 확인하십시오.

```
fusion-en:/db2home /db2home nfs rw,timeo=7,
    hard,intr,bg,suid,lock
```

여기서, *fusion-en*은 머신 이름을 나타냅니다.

6. 다음 명령을 입력하여 클러스터의 나머지 각 머신에서 익스포트된 파일 시스템을 NFS 마운트합니다.

   ```
   mount /db2home
   ```

   mount 명령이 실패하면, showmount 명령을 사용하여 NFS 서버의 상태를 점검하십시오. 예를 들어, 다음과 같습니다.

   ```
   showmount -e fusion-en
   ```

   여기서, *fusion-en*은 머신 이름을 나타냅니다.

   이 showmount 명령은 fusion-en이라는 머신에서 익스포트된 파일 시스템을 나열해야 합니다. 이 명령이 실패하면 NFS 서버가 시작되지 않을 수 있습니다. 서버를 수동으로 시작하려면 NFS 서버에서 루트로 다음 명령을 실행하십시오.

   ```
   /etc/rc.d/init.d/nfs restart
   ```

   현재 실행 레벨이 3이라고 가정하면 /etc/rc.d/rc3.d 디렉토리에서 K20nfs를 S20nfs로 이름을 바꾸어 부트 시 자동으로 이 명령을 실행할 수 있습니다.

7. 다음 단계를 완료했는지 확인하십시오.
   a. 클러스터의 단일 머신에서, 인스턴스 및 홈 디렉토리로 사용될 파일 시스템을 작성했습니다.
   b. 단일 데이터베이스 인스턴스에 대해 둘 이상의 머신을 사용하는 구성의 경우, NFS를 사용하여 이 파일 시스템을 익스포트했습니다.
   c. 클러스터의 나머지 각 머신에서 익스포트된 파일 시스템을 마운트했습니다.

파티션된 데이터베이스 시스템에 대한 DB2 홈 파일 시스템 작성(AIX)

이 태스크는 파티션된 데이터베이스 시스템을 설정하는 단계의 일부입니다. 이 태스크에서는 다음 작업을 수행하는 방법에 대해 설명합니다.

- DB2 홈 파일 시스템 작성
- 홈 파일 시스템 NFS 익스포트
- 참여하는 각 컴퓨터로부터 홈 파일 시스템 NFS 마운트

DB2 제품 DVD의 컨텐츠 크기 정도로 홈 파일 시스템을 작성하는 것이 좋습니다. 다음 명령을 사용하여 크기를 점검할 수 있습니다(KB 단위로 표시됨).

```
du -sk <DVD mounting point>
```

DB2 인스턴스에는 약 200MB의 스페이스가 필요합니다. 충분한 여유 공간이 없는 경우, 내용을 디스크로 복사하는 대신 참여하는 각 컴퓨터로부터 DB2 제품 DVD를 마운트할 수 있습니다.

다음 사항이 있어야 합니다.

- 파일 시스템을 작성하기 위한 루트 권한
- 파일 시스템이 실제 위치할 볼륨 그룹이 작성되어 있어야 함

DB2 홈 파일 시스템을 작성, NFS 익스포트 및 NFS 마운트하려면 다음 단계를 수행하십시오.

DB2 홈 파일 시스템 작성

루트 권한이 있는 사용자로 파티션된 데이터베이스 시스템의 기본 컴퓨터(ServerA)에 로그온하여 파티션된 데이터베이스 시스템에 대한 홈 파일 시스템 /db2home을 작성하십시오.

1. **smit jfs** 명령을 입력하십시오.
2. **저널 파일 시스템 추가** 아이콘을 누르십시오.
3. **표준 저널 파일 시스템 추가** 아이콘을 누르십시오.
4. **볼륨 그룹 이름** 목록에서 이 파일 시스템이 실제 위치할 기존 볼륨 그룹을 선택하십시오.
5. 파일 시스템의 *크기*(**파일 시스템의 크기(512바이트 블록)(Num.)** 필드)를 설정하십시오. 이 *크기*는 512바이트 블록에 열거되므로 인스턴스 홈 디렉토리에 파일 시스템을 작성해야만 하는 경우, 약 90MB인 180,000을 사용할 수 있습니다. 설치를 실행하기 위해 제품 DVD 이미지를 복사해야 하는 경우, 약 1GB인 2,000,000의 값으로 이를 작성할 수 있습니다.
6. **마운트 위치** 필드에 이 파일 시스템에 대한 마운트 위치를 입력하십시오. 예에서 마운트 위치는 /db2home입니다.
7. **시스템 재시작시 자동으로 마운트하시겠습니까?** 필드를 예로 설정하십시오.

   나머지 필드는 디폴트값으로 남겨두어도 됩니다.
8. **확인**을 누르십시오.

DB2 홈 파일 시스템 익스포트

1. 파티션된 데이터베이스 시스템에 참여할 모든 컴퓨터에서 /db2home이 파일 시스템을 사용할 수 있도록 파일 시스템을 NFS 익스포트하십시오.
   a. **smit nfs** 명령을 입력하십시오.
   b. **NFS(Network File System)** 아이콘을 누르십시오.
   c. **익스포트 목록에 디렉토리 추가** 아이콘을 누르십시오.

d. **익스포트할 디렉토리의 경로 이름** 필드에 익스포트할 경로 이름 및 디렉토리(예: /db2home)를 입력하십시오.

e. **루트 액세스가 허용된 호스트** 필드에 파티션된 데이터베이스 시스템에 참여할 각 워크스테이션 이름을 입력하십시오. 각 이름 간의 분리문자로서 쉼표(,)를 사용하십시오. (예: ServerA, ServerB, ServerC). 신속한 상호 연결을 사용할 경우, 이 필드에 각 워크스테이션마다 신속한 상호 연결의 이름도 지정하는 것이 바람직합니다. 나머지 필드는 디폴트값으로 남겨두어도 됩니다.

f. **확인**을 누르십시오.

2. 로그아웃하십시오.

참여하는 각 컴퓨터로부터 DB2 홈 파일 시스템 마운트

참여하는 *각* 컴퓨터(ServerB, ServerC, ServerD)에 로그온한 후 다음 단계를 수행하여 익스포트한 파일 시스템을 NFS 마운트하십시오.

1. **smit nfs** 명령을 입력하십시오.
2. **NFS(Network File System)** 아이콘을 누르십시오.
3. **마운트할 파일 시스템 추가** 아이콘을 누르십시오.
4. **마운트 위치의 경로 이름(경로)** 필드에 마운트 위치의 경로 이름을 입력하십시오.

   마운트 위치의 경로 이름은 DB2 홈 디렉토리를 작성해야 하는 위치입니다. 예를 들면, /db2home입니다.
5. **리모트 디렉토리의 경로 이름** 필드에 리모트 디렉토리의 경로 이름을 입력하십시오.

   이 예에서는 **마운트 위치의 경로 이름(경로)** 필드에 입력했던 값과 동일한 값을 입력해야 합니다.
6. **리모트 디렉토리의 상주 호스트** 필드에 사용자가 파일 시스템을 익스포트할 머신의 *호스트 이름*을 입력하십시오.

   이 값은 마운트할 파일 시스템이 작성된 머신의 호스트 이름입니다.

   성능을 개선하기 위해 사용자가 작성한 파일 시스템을 신속한 상호 연결을 통해 NFS 마운트할 수도 있습니다. 신속한 상호 연결을 사용하여 이 파일 시스템을 마운트하려면 **리모트 디렉토리의 상주 호스트** 필드에 호스트의 이름을 입력해야 합니다.

   몇 가지 이유로 신속한 상호 연결을 사용할 수 없게 된 경우, 파티션된 데이터베이스 시스템에 참여한 모든 워크스테이션은 DB2 홈 디렉토리에 대한 액세스 권한을 상실합니다.

7. **마운트(지금 마운트, /etc/filesystems에 항목 추가, 모두)?** 필드를 모두로 설정하십시오.
8. **시스템 재시작 시에 /etc/filesystems의 내용으로 디렉토리를 마운트합니다.** 필드를 예로 설정하십시오.
9. **이 NFS 파일 시스템의 모드** 필드를 읽고 쓰기로 설정하십시오.
10. **파일 시스템 마운트 방법(소프트, 하드)** 필드를 소프트로 설정하십시오.

    소프트 마운트는 컴퓨터에서 리모트로 디렉토리를 마운트하려고 무한정 시도하지 않음을 의미합니다. 하드 마운트는 사용자 머신이 디렉토리 마운트를 무한정 시도하게 된다는 것을 의미합니다. 이로 인해 시스템 손상 시 문제가 발생할 수 있습니다. 이 필드는 소프트로 설정하는 것이 바람직합니다.

    나머지 필드는 디폴트값으로 남겨두어도 됩니다.
11. 이 파일 시스템이 **이 파일 시스템에서 SUID와 sgid 프로그램의 실행을 허용하시겠습니까?** 필드가 예로 설정되어 마운트되었는지 확인하십시오. 이것이 디폴트 설정입니다.
12. **확인**을 누르십시오.
13. 로그아웃하십시오.

파티션된 데이터베이스 환경에서 DB2 서버 설치에 필수 사용자 작성(Linux)

DB2 데이터베이스를 작동하려면 세 개의 사용자와 그룹이 필요합니다. 여기서 설명하는 사용자 및 그룹 이름은 다음 표와 같습니다. 시스템 이름 지정 규칙 및 DB2 이름 지정 규칙에 맞는 자체 사용자 및 그룹 이름을 지정할 수 있습니다 .

DB2 설치 마법사를 사용하여 DB2 제품을 설치할 경우, DB2 설치 마법사는 다음과 같은 사용자를 작성합니다.

표 10. 필수 사용자 및 그룹

필수 사용자	사용자 이름	그룹 이름
인스턴스 소유자	db2inst1	db2iadm1
분리(Fenced) 소유자	db2fenc1	db2fadm1
DB2 Administration Server 사용자	dasusr1	dasadm1

DB2 Administration Server 사용자가 기존 사용자이면 이 사용자는 설치 이전에 모든 참여 컴퓨터에 존재해야 합니다. DB2 설치 마법사를 사용하여 인스턴스 소유 컴퓨터에서 DB2 Administration Server용으로 새 사용자를 작성할 경우, 참여 컴퓨터에 응답 파일을 설치하는 중에도 필요하면 새 사용자가 작성됩니다. 이 사용자가 이미 참여 컴퓨터에 있는 경우, 사용자는 같은 1차 그룹을 가져야 합니다.

전제조건

- 사용자 및 그룹을 작성하려면 root 권한이 있어야 합니다.
- 보안 소프트웨어로 사용자 및 그룹을 관리할 경우, DB2 사용자 및 그룹을 정의할 때 추가 단계가 필요할 수도 있습니다.

제한사항

사용자가 작성한 사용자 이름은 운영 체제의 이름 지정 규칙과 DB2의 이름 지정 규칙을 모두 따라야 합니다.

이 세 사용자 모두를 작성하려면 다음을 수행하십시오.

1. 기본 컴퓨터에 로그온하십시오.
2. 다음 명령을 입력하여 인스턴스 소유자에 대한 그룹(예: db2iadm1), UDF 또는 스토어드 프로시저를 실행하는 그룹(예: db2fadm1) 및 DB2 Administration Server를 소유할 그룹(예: dasadm1)을 작성하십시오.

   ```
   groupadd -g 999 db2iadm1
      groupadd -g 998 db2fadm1
      groupadd -g 997 dasadm1
   ```

   사용하고 있는 특정 번호는 현재 어떤 머신에도 존재하지 않아야 합니다.
3. 다음 명령을 사용하여 이전 단계에서 작성한 각 그룹에 속하는 사용자를 작성하십시오. 각 사용자의 홈 디렉토리는 이전에 작성하여 공유한 DB2 홈 디렉토리(db2home)가 됩니다.

   ```
   useradd -u 1004 -g db2iadm1 -m -d /db2home/db2inst1 db2inst1
   useradd -u 1003 -g db2fadm1 -m -d /db2home/db2fenc1 db2fenc1
   useradd -u 1002 -g dasadm1 -m -d /home/dasusr1 dasusr1
   ```
4. 작성된 각 사용자에 대한 초기 암호를 다음 명령으로 설정하십시오.

   ```
   passwd db2inst1    passwd db2fenc1    passwd dasusr1
   ```
5. 로그아웃하십시오.
6. 작성된 각 사용자(db2inst1, db2fenc1 및 dasusr1)로 기본 컴퓨터에 로그인하십시오. 작성된 사용자가 시스템에 처음 로그온했으므로 각 사용자의 암호를 변경하라고 요청을 받습니다.
7. 로그아웃하십시오.
8. 파티션된 데이터베이스 환경에 참여할 각 컴퓨터에 정확히 동일한 사용자 및 그룹 어카운트를 작성하십시오.

파티션된 데이터베이스 환경에서 DB2 서버 설치에 필수 사용자 작성(AIX)

DB2 데이터베이스를 작동하려면 세 개의 사용자와 그룹이 필요합니다. 여기서 설명하는 사용자 및 그룹 이름은 다음 표와 같습니다. 시스템 이름 지정 규칙 및 DB2 이름 지정 규칙에 맞는 자체 사용자 및 그룹 이름을 지정할 수 있습니다 .

DB2 설치 마법사를 사용하여 DB2 제품을 설치할 경우, DB2 설치 마법사는 다음과 같은 사용자를 작성합니다.

표 11. 필수 사용자 및 그룹

필수 사용자	사용자 이름	그룹 이름
인스턴스 소유자	db2inst1	db2iadm1
분리(Fenced) 소유자	db2fenc1	db2fadm1
DB2 Administration Server 사용자	dasusr1	dasadm1

DB2 Administration Server 사용자가 기존 사용자이면 이 사용자는 설치 이전에 모든 참여 컴퓨터에 존재해야 합니다. DB2 설치 마법사를 사용하여 인스턴스 소유 컴퓨터에서 DB2 Administration Server용으로 새 사용자를 작성할 경우, 참여 컴퓨터에 응답 파일을 설치하는 중에도 필요하면 새 사용자가 작성됩니다. 이 사용자가 이미 참여 컴퓨터에 있는 경우, 사용자는 같은 1차 그룹을 가져야 합니다.

전제조건

- 사용자 및 그룹을 작성하려면 root 권한이 있어야 합니다.
- 보안 소프트웨어로 사용자 및 그룹을 관리할 경우, DB2 사용자 및 그룹을 정의할 때 추가 단계가 필요할 수도 있습니다.

제한사항

사용자가 작성한 사용자 이름은 운영 체제의 이름 지정 규칙과 DB2의 이름 지정 규칙을 모두 따라야 합니다.

이 세 사용자 모두를 작성하려면 다음을 수행하십시오.

1. 기본 컴퓨터에 로그온하십시오.
2. 다음 명령을 입력하여 인스턴스 소유자에 대한 그룹(예: db2iadm1), UDF 또는 스토어드 프로시저를 실행하는 그룹(예: db2fadm1) 및 DB2 Administration Server를 소유할 그룹(예: dasadm1)을 작성하십시오.

   ```
   mkgroup id=999 db2iadm1
      mkgroup id=998 db2fadm1
      mkgroup id=997 dasadm1
   ```

3. 다음 명령을 사용하여 이전 단계에서 작성한 각 그룹에 속하는 사용자를 작성하십시오. 각 사용자의 홈 디렉토리는 이전에 작성하여 공유한 DB2 홈 디렉토리(db2home)가 됩니다.

   ```
   mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1 home=/db2home/db2inst1
         core=-1 data=491519 stack=32767 rss=-1 fsize=-1 db2inst1
   mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1 home=/db2home/db2fenc1
         db2fenc1
   mkuser id=1002 pgrp=dasadm1 groups=dasadm1 home=/home/dasusr1
         dasusr1
   ```

4. 작성된 각 사용자에 대한 초기 암호를 다음 명령으로 설정하십시오.

```
passwd db2inst1
passwd db2fenc1
passwd dasusr1
```

5. 로그아웃하십시오.
6. 작성된 각 사용자(db2inst1, db2fenc1 및 dasusr1)로 기본 컴퓨터에 로그인하십시오. 작성된 사용자가 시스템에 처음 로그온했으므로 각 사용자의 암호를 변경하라고 요청을 받습니다.
7. 로그아웃하십시오.
8. 파티션된 데이터베이스 환경에 참여할 각 컴퓨터에 정확히 동일한 사용자 및 그룹 어카운트를 작성하십시오.

제 7 장 DB2 서버 제품 설치

파티션된 데이터베이스 환경 설정

이 주제에서는 파티션된 데이터베이스 환경을 설정하는 방법에 대해 설명합니다. DB2 설치 마법사를 사용하여 인스턴스 소유 데이터베이스 서버를 설치하고 참여하는 데이터베이스 서버를 작성하는 데 사용하는 응답 파일을 작성할 수 있습니다.

주: 파티션된 데이터베이스 환경은 비루트 설치에서는 지원되지 않습니다.

데이터베이스 파티션은 자체 데이터, 인덱스, 구성 파일 및 트랜잭션 로그로 구성된 데이터베이스의 파트입니다. 파티션된 데이터베이스는 둘 이상의 파티션이 있는 데이터베이스입니다.

전제조건

- 참여하는 모든 컴퓨터에 복사해야 하는 InfoSphere™ Warehouse Activation CD 라이센스 키가 있는지 확인하십시오.
- 파티션된 데이터베이스 환경에 참여할 각 컴퓨터에서 같은 수의 연속 포트가 사용 중이면 안 됩니다. 예를 들어, 파티션된 데이터베이스 환경은 네 개의 컴퓨터로 구성되고 네 개의 각 컴퓨터에는 같은 네 개의 연속 포트가 사용될 수 없습니다. 인스턴스 작성 중에, 현재 서버의 논리적 파티션 수와 동일한 포트 수가 Linux 및 UNIX에서 `/etc/services`에 예약되고 Windows에서는 `%SystemRoot%₩system32₩drivers₩etc₩services`에 예약됩니다. 이 포트는 FCM(Fast Communication Manager) 프로그램에서 사용됩니다. 예약된 포트의 형식은 다음과 같게 됩니다.

  ```
  DB2_InstanceName
  DB2_InstanceName_1
  DB2_InstanceName_2
  DB2_InstanceName_END
  ```

  필수 항목은 시작(DB2_InstanceName) 및 종료(DB2_InstanceName_END) 포트입니다. 다른 항목은 서비스 파일에 예약되어 있으므로 다른 항목에서는 이 포트를 사용하지 않습니다.
- 참여한 다중 DB2 데이터베이스 서버를 지원하려면 DB2를 설치할 컴퓨터가 액세스 가능한 도메인에 속해 있어야 합니다. 그러나 컴퓨터가 도메인에 속해 있지 않더라도 이 컴퓨터에 로컬 파티션을 추가할 수 있습니다.
- Linux 및 UNIX 시스템의 경우, 파티션된 데이터베이스 시스템에는 리모트 쉘 유틸리티가 필요합니다. DB2는 다음과 같은 리모트 쉘 유틸리티를 지원합니다.

– rsh

– ssh

디폴트로 DB2는 리모트 DB2 노드에서 명령을 실행할 때 rsh를 사용합니다(예를 들어, 리모트 DB2 데이터베이스 파티션을 시작할 때). DB2 디폴트 값을 사용하려면 rsh-server 패키지가 설치되어 있어야 합니다. DB2 제품을 설치 시 보안 문제에 대한 자세한 정보는 관련 링크를 참조하십시오.

rsh 리모트 쉘 유틸리티를 사용하도록 선택한 경우 inetd(또는 xinetd)도 설치되어 실행 중이어야 합니다. ssh 리모트 쉘 유틸리티를 사용하도록 선택한 경우, DB2 설치가 완료된 즉시 DB2RSHCMD 레지스트리 변수를 설정해야 합니다. 이 레지스트리 변수를 설정하지 않으면 rsh가 사용됩니다.

- Linux 및 UNIX 운영 체제에서, IP 주소가 머신의 완전한 호스트 이름에 맵핑되는 경우 etc 디렉토리 아래의 hosts 파일에 『127.0.0.2』 항목이 포함되어 있지 않은지 확인하십시오.

파티션된 데이터베이스 환경을 설정하려면 다음을 수행하십시오.

1. DB2 설치 마법사를 사용하여 인스턴스 소유 서버를 설치하십시오. 자세한 지시사항을 보려면 사용자 플랫폼에 해당하는 "DB2 서버 설치" 주제를 참조하십시오.
   - **설치 선택, 응답 파일 작성 또는 둘 다 선택** 창에서 **응답 파일에 내 설치 설정 저장** 옵션을 선택하십시오. 설치가 완료되면 두 개의 파일(PROD_ESE.rsp 및 PROD_ESE_addpart.rsp)이 DB2 설치 마법사에 지정된 디렉토리로 복사됩니다. PROD_ESE.rsp 파일은 인스턴스 소유 데이터베이스 서버의 응답 파일입니다. PROD_ESE_addpart.rsp 파일은 참여한 데이터베이스 서버의 응답 파일입니다.
   - **DB2 인스턴스의 파티션 옵션 설정** 창에서 **다중 파티션 인스턴스**를 선택한 후 최대 논리 파티션 수를 입력하십시오.
2. 파티션된 데이터베이스 환경에 참여한 모든 컴퓨터가 DB2 설치 이미지를 사용할 수 있도록 설정하십시오.
3. 참여한 데이터베이스 서버 응답 파일(PROD_ESE_addpart.rsp)을 분배하십시오.
4. db2setup 명령(Linux 및 UNIX) 또는 setup 명령 (Windows)을 사용하여 참여하는 각 컴퓨터에서 DB2 데이터베이스 서버를 설치하십시오.

   Linux 및 UNIX

   DB2 제품 코드가 사용 가능한 디렉토리로 이동하여 다음 명령을 실행하십시오.

   ```
   ./db2setup -r /responsefile_directory/response_file_name
   ```

   Windows

   ```
   setup -u x:¥responsefile_directory¥response_file_name
   ```

   예를 들어, PROD_ESE_addpart.rsp를 응답 파일로 사용하는 명령이 있습니다.

Linux 및 UNIX

DB2 제품 코드가 사용 가능한 디렉토리로 이동하여 다음 명령을 실행하십시오.

```
./db2setup -r /db2home/PROD_ESE_addpart.rsp
```

여기서, /db2home은 응답 파일을 복사한 디렉토리입니다.

Windows

```
setup -u c:¥resp_files¥PROD_ESE_addpart.rsp
```

여기서, c:¥resp_files¥는 응답 파일을 복사한 디렉토리입니다.

5. (Linux 및 UNIX 전용) db2nodes.cfg 파일을 구성하십시오. DB2 설치는 현재 컴퓨터에 사용할 논리적 파티션의 최대 수만을 예약하고 db2nodes.cfg 파일을 구성하지는 않습니다. db2nodes.cfg 파일을 구성하지 않은 경우에는 인스턴스는 여전히 단일 파티션된 인스턴스입니다.
6. 참여한 서버에서 서비스 파일을 갱신하여 DB2 인스턴스의 해당하는 FCM 포트를 정의하십시오. 서비스 파일이 다음 위치에 있습니다.
   - /etc/services(Linux 및 UNIX)
   - %SystemRoot%¥system32¥drivers¥etc¥services(Windows)
7. Windows 2000 이상의 파티션된 데이터베이스 환경에서는 데이터 및 자원을 보호하기 위해 DB2 Remote Command Service 보안 기능을 시작하십시오.

   완전한 보안을 위해 컴퓨터(LocalSystem 어카운트의 컨텍스트에서 서비스가 실행 중인 경우) 또는 위임할 사용자(사용자의 로그온 컨텍스트에서 서비스가 시작된 경우)를 시작하십시오.

   DB2 Remote Command Service 보안 기능을 시작하려면 다음을 수행하십시오.

   a. 도메인 제어기에서 Active Directory 사용자 및 컴퓨터 창을 열고 **시작**을 누르고 **프로그램 → 관리 도구 → Active Directory 사용자 및 컴퓨터**를 선택하십시오.
   b. 오른쪽 창 패널에서 시작할 컴퓨터 또는 사용자를 마우스 오른쪽 단추로 누른 후 **등록 정보**를 선택하십시오.
   c. **일반** 탭을 누르고 **위임에 대해 컴퓨터 신뢰** 선택란을 선택하십시오. 사용자 설정의 경우, **어카운트** 탭을 누르고 **어카운트 옵션** 그룹에 있는 **어카운트가 위임에 대해 신뢰됨** 선택란을 선택하십시오. **어카운트가 중요하여 위임할 수 없음** 상자가 선택되지 않았는지 확인하십시오.
   d. **확인**을 눌러 위임을 위한 컴퓨터 또는 사용자를 시작하십시오.

   시작해야 하는 각 사용자 또는 컴퓨터에 대해 이 단계를 시작하십시오. 보안 변경을 적용하려면 컴퓨터를 재시작해야 합니다.

응답 파일을 사용하여 참여 컴퓨터에 데이터베이스 파티션 서버 설치(Windows)

이 태스크에서는 DB2 설치 마법사로 작성한 응답 파일을 사용하여 참여 컴퓨터에 데이터베이스 파티션 서버를 설치합니다.

전제조건

- DB2 사본을 DB2 설치 마법사를 사용하여 기본 컴퓨터에 설치하십시오.
- 참여 컴퓨터를 설치하기 위한 응답 파일을 작성했고 이를 참여 컴퓨터로 복사했습니다.
- 참여 컴퓨터에 대한 관리자 권한이 필요합니다.

응답 파일을 사용하여 추가 데이터베이스 파티션 서버를 설치하려면 다음과 같이 수행하십시오.

1. DB2 설치에 정의한 로컬 관리자 어카운트로 컴퓨터에 로그온하십시오.
2. DB2 제품 DVD가 포함된 디렉토리로 변경하십시오. 예를 들어, 다음과 같습니다.

   ```
   cd c:¥db2dvd
   ```

   여기서, db2dvd는 DB2 제품 DVD가 포함된 디렉토리의 이름입니다.
3. 명령 프롬프트에서 다음과 같이 setup 명령을 입력하십시오.

   ```
   setup -u responsefile_directory¥response_file_name
   ```

   다음 예에서는 Addpart.file 응답 파일을 c:¥responsefile 디렉토리에서 찾을 수 있습니다. 따라서 명령은 다음과 같습니다.

   ```
   setup -u c:¥reponsefile¥Addpart.file
   ```
4. 설치 완료 시 로그 파일의 메시지를 점검하십시오. My Documents¥DB2LOG¥ 디렉토리에서 로그 파일을 찾을 수 있습니다. 로그 파일 끝에서 다음과 유사한 출력이 표시되어야 합니다.

   ```
   === Logging stopped: 5/9/2007  10:41:32 ===
   MSI (c) (C0:A8) [10:41:32:984]: Product: DB2
   Enterprise Server Edition - DB2COPY1 -- Installation
   operation completed successfully.
   ```
5. 인스턴스 소유 데이터베이스 파티션 서버를 기본 컴퓨터에 설치할 경우, DB2 제품은 파티션된 데이터베이스 환경에 참여하는 지정된 논리적 데이터베이스 파티션 서버 수에 따라 포트 범위를 예약합니다. 디폴트 범위는 네 개의 포트입니다. 파티션된 데이터베이스 환경에 참여하는 각 서버의 경우, FCM 포트에 대해 /etc/services 파일을 수동으로 구성해야 합니다. FCM 포트의 범위는 참여하는 컴퓨터에서 사용할 논리적 파티션 수에 따라 다릅니다. 최소 두 개의 항목 **DB2_<instance>** 및 **DB2_<instance>_END**가 필요합니다. 참여 컴퓨터에 지정되는 FCM 포트의 다른 요구사항은 다음과 같습니다.
   - 시작 포트 번호는 기본 컴퓨터의 시작 포트 번호와 일치해야 합니다.
   - 후속 포트는 순차적으로 번호가 지정되어야 합니다.

- 지정된 포트 번호는 사용 가능해야 합니다.

각 참여 컴퓨터에 로그온하고 이들 단계를 반복하십시오.

DB2 제품에서 로컬 컴퓨터 또는 네트워크의 다른 컴퓨터에 있는 DB2 문서에 액세스가 가능하려면 DB2 정보 센터를 설치해야 합니다. DB2 정보 센터에는 DB2 데이터베이스 시스템 및 DB2 관련 제품에 대한 문서가 포함되어 있습니다.

응답 파일을 사용하여 참여 컴퓨터에 데이터베이스 파티션 서버 설치(Linux 및 UNIX)

이 태스크에서는 DB2 설치 마법사로 작성한 응답 파일을 사용하여 참여 컴퓨터에 데이터베이스 파티션 서버를 설치합니다.

전제조건

- DB2 설치 마법사를 사용하여 기본 컴퓨터에 DB2를 설치하고 참여 컴퓨터에 설치에 필요한 응답 파일을 작성해야 합니다.
- 참여 컴퓨터에 대한 루트 권한이 필요합니다.

응답 파일을 사용하여 추가 데이터베이스 파티션 서버를 설치하려면 다음과 같이 수행하십시오.

1. 파티션된 데이터베이스 환경에 참여할 컴퓨터에 루트로 로그온하십시오.
2. DB2 제품 DVD의 내용을 복사한 디렉토리로 변경하십시오. 예를 들어, 다음과 같습니다.

   ```
   cd /db2home/db2dvd
   ```

3. db2setup 명령을 다음과 같이 입력하십시오.

   ```
   ./db2setup -r /responsefile_directory/response_file_name
   ```

   이 예에서 응답 파일 AddPartitionResponse.file은 /db2home 디렉토리에 저장되었습니다. 이 예에서 명령은 다음과 같습니다.

   ```
   ./db2setup -r /db2home/AddPartitionResponse.file
   ```

4. 설치 완료 시 로그 파일의 메시지를 점검하십시오.

각 참여 컴퓨터에 로그온하고 응답 파일 설치를 수행해야 합니다.

DB2 제품에서 로컬 컴퓨터 또는 네트워크의 다른 컴퓨터에 있는 DB2 문서에 액세스가 가능하려면 DB2 정보 센터를 설치해야 합니다. DB2 정보 센터에는 DB2 데이터베이스 시스템 및 DB2 관련 제품에 대한 문서가 포함되어 있습니다.

제 8 장 설치 후

설치 확인

파티션된 데이터베이스 환경 설치 확인(Windows)

DB2 서버 설치가 완료되었는지 확인하려면, 샘플 데이터베이스를 작성한 후 SQL 명령을 실행하여 샘플 데이터를 검색하고 참여한 모든 데이터베이스 파티션 서버로 데이터가 분산되었는지 확인하십시오.

모든 설치 단계를 완료해야 합니다.

SAMPLE 데이터베이스를 작성하려면 다음과 같이 수행하십시오.

1. SYSADM 권한을 가진 사용자로 기본 컴퓨터(ServerA)에 로그온하십시오.
2. db2sampl 명령을 입력하여 SAMPLE 데이터베이스를 작성하십시오.

   이 명령이 처리되는 데에는 2-3분 정도 소요됩니다. 명령 프롬프트가 리턴되면, 해당 프로세스가 완료될 것입니다.

   SAMPLE 데이터베이스는 작성될 때 자동으로 데이터베이스 별명 SAMPLE을 사용하여 카탈로그됩니다.
3. db2start 명령을 입력하여 데이터베이스 관리 프로그램을 시작하십시오.
4. DB2 명령 창에서 다음 DB2 명령을 입력하여 SAMPLE 데이터베이스에 연결한 다음 부서 20에서 작업하는 모든 직원 목록을 검색하십시오.

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```
5. 데이터베이스 파티션 서버로 데이터가 분산되었는지 확인하려면 DB2 명령 창에서 다음 명령을 입력하십시오.

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   employee 테이블에서 사용하는 데이터베이스 파티션이 목록으로 출력됩니다. 특정 출력은 데이터베이스에 있는 데이터베이스 파티션 수와 직원 테이블이 작성된 테이블 스페이스에 의해 사용되는 데이터베이스 파티션 그룹에 있는 데이터베이스 파티션 수에 따라 달라집니다.

설치를 확인한 다음 SAMPLE 데이터베이스를 제거하여 디스크 스페이스를 늘릴 수 있습니다. 그러나 샘플 데이터베이스를 사용하려면 샘플 데이터를 보존하는 것이 유용합니다.

db2 drop database sample 명령을 입력하여 SAMPLE 데이터베이스를 삭제하십시오.

파티션된 데이터베이스 서버 설치 검증(Linux 및 UNIX)

db2val 도구를 사용하여 설치 파일, 인스턴스, 데이터베이스 작성, 해당 데이터베이스에 대한 연결, DPF 환경의 상태에 대한 유효성 검사를 통해 DB2 사본의 코어 기능을 확인합니다. 자세한 내용은 『DB2 사본 유효성 확인』을 참조하십시오. DPF 환경의 상태는 노드가 2개 이상일 경우에만 검증됩니다. 또한 DB2 서버 설치가 완료되었는지 확인하려면, 샘플 데이터베이스를 작성한 후 SQL 명령을 실행하여 샘플 데이터를 검색하고 참여한 모든 데이터베이스 파티션 서버로 데이터가 분산되었는지 확인하십시오.

다음 단계를 시작하기 전에 모든 설치 단계를 완료해야 합니다.

SAMPLE 데이터베이스를 작성하려면 다음과 같이 수행하십시오.

1. 인스턴스 소유 사용자로 기본 컴퓨터(ServerA)에 로그온하십시오. 이 예에서는 db2inst1이 인스턴스 소유 사용자입니다.
2. db2sampl 명령을 입력하여 SAMPLE 데이터베이스를 작성하십시오. 디폴트로 샘플 데이터베이스는 인스턴스 소유 홈 디렉토리에 작성됩니다. 예에서는 /db2home/db2inst1/이 인스턴스 소유자의 홈 디렉토리입니다. 인스턴스 소유자의 홈 디렉토리가 디폴트 데이터베이스 경로입니다.

   이 명령이 처리되는 데에는 2-3분 정도 소요됩니다. 완료 메시지는 없지만, 명령 프롬프트가 리턴되면, 해당 프로세스가 완료될 것입니다.

   SAMPLE 데이터베이스는 작성될 때 자동으로 데이터베이스 별명 SAMPLE을 사용하여 카탈로그됩니다.
3. db2start 명령을 입력하여 데이터베이스 관리 프로그램을 시작하십시오.
4. DB2 명령 창에서 다음 DB2 명령을 입력하여 SAMPLE 데이터베이스에 연결한 다음 부서 20에서 작업하는 모든 직원 목록을 검색하십시오.

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```
5. 데이터베이스 파티션 서버로 데이터가 분산되었는지 확인하려면 DB2 명령 창에서 다음 명령을 입력하십시오.

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   employee 테이블에서 사용하는 데이터베이스 파티션이 목록으로 출력됩니다. 특정 출력은 다음에 따라 다릅니다.
   - 데이터베이스의 데이터베이스 파티션 수
   - 데이터베이스 파티션 그룹에서 employee 테이블이 작성된 테이블 스페이스가 사용하는 데이터베이스 파티션 수.

설치를 확인한 다음 SAMPLE 데이터베이스를 제거하여 디스크 스페이스를 늘릴 수 있습니다. db2 drop database sample 명령을 입력하여 SAMPLE 데이터베이스를 삭제하십시오.

제 3 부 구현 및 유지보수

제 9 장 데이터베이스 작성 전

파티션된 데이터베이스 환경 설정

데이터베이스를 작성하기 전에 다중 파티션 데이터베이스를 작성 여부를 결정해야 합니다. 데이터베이스 설계 결정의 일부로서, 데이터베이스 파티션에서 제공할 수 있는 성능을 향상시킬 것인지 여부를 결정해야 합니다.

파티션된 데이터베이스 환경에서도 CREATE DATABASE 명령 또는 sqlecrea() 함수를 사용하여 데이터베이스를 작성합니다. 사용되는 메소드가 어느 것이든 `db2nodes.cfg` 파일에 나열된 임의의 파티션을 통해 요청할 수 있습니다. `db2nodes.cfg` 파일은 데이터베이스 파티션 서버 구성 파일입니다. (이전에는 노드 구성 파일로 알려져 있었습니다.)

Windows 운영 체제 환경을 제외하고 임의의 편집기를 사용하여 데이터베이스 파티션 서버 구성 파일(`db2nodes.cfg`)의 컨텐츠를 보고 갱신할 수 있습니다. Windows 운영 체제 환경에서 db2ncrt 및 db2nchg 명령을 사용하여 데이터베이스 파티션 서버 구성 파일을 작성 및 변경하십시오.

다중 파티션 데이터베이스를 작성하기 전에, 데이터베이스의 카탈로그 파티션이 될 데이터베이스 파티션을 선택해야 합니다. 그런 다음 해당 데이터베이스 파티션에서 직접 또는 그 데이터베이스 파티션에 접속된 리모트 클라이언트에서 데이터베이스를 작성할 수 있습니다. 접속하여 CREATE DATABASE 명령을 실행하는 데이터베이스 파티션이 해당 특정 데이터베이스에 대한 *카탈로그 파티션*이 됩니다.

카탈로그 파티션은 모든 시스템 카탈로그 테이블이 저장되는 데이터베이스 파티션입니다. 시스템 테이블로의 모든 액세스는 이 데이터베이스 파티션을 거쳐야 합니다. 모든 페더레이티드 데이터베이스 오브젝트(예: 랩퍼, 서버 및 별칭)는 이 데이터베이스 파티션에서 시스템 카탈로그 테이블에 저장됩니다.

가능하다면, 별도의 인스턴스에서 각 데이터베이스를 작성해야 합니다. 그렇지 않은 경우(즉, 인스턴스마다 둘 이상의 데이터베이스를 작성해야 할 경우), 사용 가능한 데이터베이스 파티션 간에 카탈로그 파티션을 확대해야 합니다. 이렇게 하면, 단일 데이터베이스 파티션에서 카탈로그 정보에 대한 경합이 줄어듭니다.

주: 다른 데이터로 인해 백업에 필요한 시간이 증가하므로 카탈로그 파티션은 정기적으로 백업해야 하며 사용자 데이터를 카탈로그 파티션에 저장하지 않아야(가능할 때마다) 합니다.

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 DB2가 설치된 디렉토리 아래의 sqllib 디렉토리에 있습니다. UNIX에서 작업할 때, 이 디렉토리는 파티션된 데이터베이스 환경을 구성하는 모든 데이터베이스 파티션에 대한 시스템 데이터베이스 디렉토리가 하나만 있으므로 공유 파일 시스템 (예: UNIX 플랫폼의 NFS)에 있어야 합니다. Windows에서 작업할 때, 시스템 데이터베이스 디렉토리는 인스턴스 디렉토리에 있습니다.

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 데이터베이스 파티션에서 공유됩니다.

데이터베이스 파티셔닝의 이점을 취하려면 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 데이터베이스 환경에 영향을 미치는 데이터베이스 관리 프로그램 구성 매개변수에는 **conn_elapse**, **fcm_num_buffers**, **fcm_num_channels**, **max_connretries**, **max_coordagents**, **max_time_diff**, **num_poolagents** 및 **stop_start_time**이 포함됩니다.

노드 구성 파일 작성

파티션된 데이터베이스 환경에서 데이터베이스를 작동시키려면 db2nodes.cfg라는 노드 구성 파일을 작성해야 합니다.

여러 데이터베이스 파티션에서 병렬 기능을 사용하여 데이터베이스 관리 프로그램을 시작하려면 인스턴스의 홈 디렉토리 아래에 있는 sqllib 서브디렉토리에 이 파일이 있어야 합니다. 이 파일에는 모든 데이터베이스 파티션에 대한 구성 정보가 포함되어 있으며 해당 인스턴스의 모든 데이터베이스 파티션에서 이 파일을 공유합니다.

Windows 고려사항

Windows에서 DB2 Enterprise Server Edition을 사용하는 경우 인스턴스를 작성하면 노드 구성 파일이 작성됩니다. 노드 구성 파일을 수동으로 작성 또는 수정하면 안됩니다. db2ncrt 명령을 사용하여 인스턴스에 데이터베이스 파티션 서버를 추가할 수 있습니다. db2ndrop 명령을 사용하여 인스턴스에서 데이터베이스 파티션 서버를 삭제할 수 있습니다. db2nchg 명령을 사용하여 하나의 컴퓨터에서 다른 컴퓨터로 데이터베이스 파티션 서버 수정, TCP/IP 호스트 이름 변경 또는 다른 논리적 포트 번호 또는 네트워크 이름 선택을 비롯하여 데이터베이스 파티션 서버 구성을 변경할 수 있습니다.

주: 인스턴스가 삭제되는 경우 데이터 손실을 예방하기 위해 데이터베이스 관리 프로그램에서 작성한 서브디렉토리 이외의 `sqllib` 서브디렉토리 아래에 파일 또는 디렉토리를 작성하면 안됩니다. 두 가지 예외가 있습니다. 시스템에서 스토어드 프로시저를 지원하는 경우 `sqllib` 서브디렉토리의 함수 서브디렉토리에 스토어드 프로시저 응용프로그램을 저장합니다. 다른 예외는 사용자 정의 함수(UDF)가 작성되는 경우입니다. UDF 실행 파일은 동일한 디렉토리에서 허용됩니다.

이 파일에는 인스턴스에 속한 각 데이터베이스 파티션에 대한 하나의 라인이 포함되어 있습니다. 각 라인의 형식은 다음과 같습니다.

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

토큰은 공백으로 구분됩니다. 변수는 다음과 같습니다.

dbpartitionnum

0에서 999 사이일 수 있는 데이터베이스 파티션 번호는 데이터베이스 파티션을 고유하게 정의합니다. 데이터베이스 파티션 번호는 오름차순이어야 합니다. 시퀀스에는 갭이 있을 수 있습니다.

데이터베이스 파티션 번호는 지정되면 변경할 수 없습니다. 그렇지 않으면 데이터가 분산되는 방법을 지정하는 분산 맵의 정보가 손상될 수 있습니다.

데이터베이스 파티션을 삭제하면 추가한 새 데이터베이스 파티션에 데이터베이스 파티션 번호를 사용할 수 있습니다.

데이터베이스 파티션 번호는 데이터베이스 디렉토리에 데이터베이스 파티션 이름을 생성하는 데 사용됩니다. 데이터베이스 파티션 번호의 형식은 다음과 같습니다.

NODE *nnnn*

*nnnn*은 왼쪽이 0으로 채워진 데이터베이스 파티션 번호입니다. 또한 데이터베이스 파티션 번호는 CREATE DATABASE 및 DROP DATABASE 명령에서 사용됩니다.

hostname

파티션 간 통신에 필요한 IP 주소의 호스트 이름입니다. 호스트 이름에 완전한 이름을 사용하십시오. 또한 `/etc/hosts` 파일에서도 완전한 이름을 사용해야 합

니다. db2nodes.cfg 파일 및 /etc/hosts 파일에 완전한 이름이 사용되지 않으면 오류 메시지 SQL30082N RC=3이 표시될 수 있습니다.

네트이름이 지정된 경우는 예외입니다. 이러한 경우 네트이름은 대부분의 통신에 사용되고 호스트 이름은 db2start, db2stop 및 db2_all에만 사용됩니다.

logical-port

이 매개변수는 선택적이고 데이터베이스 파티션의 논리적 포트 번호를 지정합니다. 이 번호는 데이터베이스 관리 프로그램 인스턴스 이름과 함께 사용되어 etc/services 파일의 TCP/IP 서비스 이름 항목을 식별합니다.

IP 주소와 논리적 포트의 조합은 잘 알려진 주소로 사용되고 데이터베이스 파티션 간에 통신 연결을 지원할 수 있도록 모든 응용프로그램에서 고유해야 합니다.

각 호스트 이름의 경우 *logical-port*는 0(영) 또는 공백이어야 합니다(디폴트값 : 0). *logical-port*와 연결된 데이터베이스 파티션은 클라이언트가 연결된 호스트의 디폴트 노드입니다. 이 값은 db2profile 스크립트의 **DB2NODE** 환경 변수 또는 sqlesetc() API로 겹쳐 쓸 수 있습니다.

netname

이 매개변수는 선택적이며 고유한 호스트 이름이 있는 하나 이상의 활성 TCP/IP 인터페이스가 있는 호스트를 지원하는 데 사용됩니다.

다음 예에서는 SP2EN1에 여러 TCP/IP 인터페이스와 두 개의 논리적 파티션이 있는 시스템에 대해 가능한 노드 구성 파일을 보여주고 SP2SW1을 DB2 데이터베이스 인터페이스로 사용합니다. 또한 1(0이 아님)에서 시작하고 *dbpartitionnum* 시퀀스에 갭이 있는 데이터베이스 파티션 번호를 보여줍니다.

표 12. 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		

원하는 편집기를 사용하여 db2nodes.cfg 파일을 갱신할 수 있습니다. (예외: 편집기는 Windows에서 사용하면 안됩니다.) 그러나 데이터베이스 파티션에서는 START DBM 발행 시 노드 구성 파일이 잠겨 있고 STOP DBM이 데이터베이스 관리 프로그램을 중지한 후에는 이 파일의 잠금이 해제되어야 하므로 파일에서 정보의 무결성을 보호하도록 주의해야 합니다. 필요한 경우 파일이 잠겨 있으면 START DBM 명령으로 파일을 갱신할 수 있습니다. 예를 들어 **RESTART** 옵션 또는 **ADD DBPARTITIONNUM** 옵션을 사용하여 START DBM을 발행할 수 있습니다.

주: STOP DBM 명령에 실패하고 노드 구성 파일을 잠금을 해제하지 못한 경우 STOP DBM **FORCE**를 발행하여 해당 파일의 잠금을 해제하십시오.

DB2 노드 구성 파일의 형식

db2nodes.cfg 파일은 DB2 인스턴스에 참여하는 데이터베이스 파티션을 정의하는 데 사용됩니다. 데이터베이스 파티션 서버 통신에 신속한 상호 연결을 사용하려는 경우, db2nodes.cfg 파일은 신속한 상호 연결의 호스트 이름 또는 IP 주소를 지정하는 데에도 사용됩니다.

Linux 및 UNIX 운영 체제에서의 db2nodes.cfg 파일의 형식은 다음과 같습니다.

```
dbpartitionnum hostname logicalport netname resourcesetname
```

dbpartitionnum, *hostname*, *logicalport*, *netname* 및 *resourcesetname*은 다음 섹션에 정의되어 있습니다.

Windows 운영 체제에서의 db2nodes.cfg 파일의 형식은 다음과같습니다.

```
dbpartitionnum hostname computername logicalport netname resourcesetname
```

Windows 운영 체제에서 db2nodes.cfg에 대한 이러한 입력은 db2ncrt 또는 START DBM ADD DBPARTITIONNUM 명령을 사용하여 추가됩니다. 또한, 이러한 입력은 db2nchg 명령을 사용하여 수정됩니다. 이러한 행을 직접 추가하거나 이 파일을 편집하면 안됩니다.

dbpartitionnum

0과 999 사이의 고유 번호로서 파티션된 데이터베이스 시스템 내의 데이터베이스 파티션 서버입니다.

파티션된 데이터베이스 시스템의 크기를 조정하려면 각 데이터베이스 파티션 서버에 대한 항목을 db2nodes.cfg 파일에 추가하십시오. 추가적인 데이터베이스 파티션 서버에 대해 선택한 *dbpartitionnum* 값은 오름차순이어야 하지만 이 시퀀스에 갭이 존재할 수 있습니다. 논리적 파티션을 추가하고 노드를 이 파일에서 논리적으로 그룹화하려면 *dbpartitionnum* 값 사이에 갭을 넣도록 선택할 수 있습니다.

이 항목은 필수 항목입니다.

hostname

FCM이 사용할 데이터베이스 파티션 서버의 TCP/IP 호스트 이름입니다. 이 항목은 필수 항목입니다. 정규 호스트 이름이 권장됩니다.

db2nodes.cfg 파일에 IP 주소 대신 호스트 이름이 제공된 경우, 데이터베이스 관리 프로그램은 동적으로 호스트 이름을 분석하려고 시도합니다. 머신의 OS 설정에 따라 로컬 또는 등록된 DNS(Domain Name Server) 찾아보기를 통해 분석됩니다.

DB2 버전 9.1부터는 TCP/IPv4 및 TCP/IPv6 프로토콜이 둘 다 지원됩니다. 호스트 이름을 분석하기 위한 메소드가 변경되었습니다.

버전 9.1 이전 릴리스에서 사용되었던 메소드로 db2nodes.cfg 파일에 정의된 문자열을 분석했으며 버전 9.1 이상의 메소드에서는 db2nodes.cfg 파일에 단축 이름이 정의된 경우 FQDN(Fully Qualified Domain Name)을 분석하려고 시도합니다. 완전한 호스트 이름에 대해 구성된 단축 이름을 지정하면 호스트 이름을 분석하는 프로세스가 불필요하게 지연됩니다.

호스트 이름 분석이 필요한 DB2 명령에서 이러한 지연을 방지하기 위해 다음과 같은 일시적인 해결책을 사용하십시오.

1. db2nodes.cfg 파일 및 운영 체제 호스트 이름 파일에 단축 이름이 지정된 경우, 운영 체제 호스트 파일에 호스트 이름에 대한 단축 이름 및 FQDN(Fully Qualified Domain Name)을 지정하십시오.
2. DB2 서버가 IPv4 포트에서 대기 중이라는 것을 알고 있는 경우, IPv4 주소만을 사용하려면 다음 명령을 발행하십시오.

   ```
   db2 catalog tcpip4 node db2tcp2 remote 192.0.32.67 server db2inst1
   with "Look up IPv4 address from 192.0.32.67"
   ```

3. DB2 서버가 IPv6 포트에서 대기 중이라는 것을 알고 있는 경우, IPv6 주소만을 사용하려면 다음 명령을 발행하십시오.

   ```
   db2 catalog tcpip6 node db2tcp3 1080:0:0:0:8:800:200C:417A server
   50000 with "Look up IPv6 address from 1080:0:0:0:8:800:200C:417A"
   ```

logicalport

데이터베이스 파티션 서버용 논리 포트 번호입니다. 이 필드는 논리 데이터베이스 파티션 서버를 실행 중인 워크스테이션에서 특정 데이터베이스 파티션 서버를 지정하는 데 사용됩니다.

설치 시 파티션 간 통신을 위해 DB2는 포트 범위(예: 60000 - 60003)를 /etc/services 파일에 예약합니다. db2nodes.cfg의 이 *logicalport* 필드는 해당 범위 중 어떤 포트를 특정 논리적 파티션 서버에 지정할 것인지를 지정합니다.

이 필드에 대한 항목이 없을 경우 디폴트값은 0입니다. 그러나 *netname* 필드에 대한 항목을 추가할 경우에는 *logicalport* 필드에 숫자를 입력해야 합니다.

논리적 데이터베이스 파티션을 사용할 경우, 지정된 *logicalport* 값은 0에서 시작하고 오름차순으로 계속*되어야 합니다*(예: 0,1,2).

또한, 하나의 데이터베이스 파티션 서버에 대해 *logicalport* 항목을 지정할 경우, db2nodes.cfg 파일에 나열된 각 데이터베이스 파티션 서버에 대해서도 *logicalport*를 지정해야 합니다.

이 필드는 논리 데이터베이스 파티션이나 신속한 상호 연결을 사용하지 않은 경우에만 선택적입니다.

netname

FCM 통신을 위해 신속한 상호 연결의 호스트 이름 또는 IP 주소를 지정합니다.

이 필드에 대한 항목이 지정되면 데이터베이스 파티션 서버 간의 모든 통신(db2start, db2stop 및 db2_all 명령으로 인한 통신은 제외)이 신속한 상호 연결을 통해 처리됩니다.

이 매개변수는 데이터베이스 파티션 통신에 신속한 상호 연결을 사용할 경우에만 필요합니다.

resourcesetname

*resourcesetname*은 노드가 시작될 운영 체제 자원을 정의합니다.
*resourcesetname*은 프로세스 친화도를 지원하기 위한 것이며, MLN(Multiple Logical Node)에 사용됩니다. 이는 quadname으로 알려진 문자열 유형 필드와 함께 제공됩니다.

이 매개변수는 AIX, HP-UX 및 Solaris 운영 체제에서만 지원됩니다.

AIX에서는 이러한 개념을 ″자원 세트″라고 하고 Solaris 운영 체제에서는 ″프로젝트″라고 합니다. 자원 관리에 대한 자세한 정보는 운영 체제 문서를 참조하십시오.

HP-UX에서 *resourcesetname* 매개변수는 PRM 그룹의 이름입니다. 자세한 정보는 HP에서 제공하는 ″HP-UX 프로세스 자원 관리자 사용자 안내서 (B8733-90007)″ 문서를 참조하십시오.

Windows 운영 체제에서 논리 노드에 대한 프로세스 친화도는 **DB2PROCESSORS** 레지스트리 변수를 통해 정의될 수 있습니다.

Linux 운영 체제에서 *resourcesetname* 컬럼은 시스템의 NUMA(Non-Uniform Memory Access) 노드에 해당하는 번호를 정의합니다. 시스템 유틸리티 numactl은 2.6 커널뿐 아니라 NUMA 규정 지원에서도 사용 가능해야 합니다.

resourcesetname 매개변수를 사용하는 경우 *netname* 매개변수를 지정해야 합니다.

구성 예

다음 구성 예를 사용하여 사용자 환경에 해당하는 구성을 판별하십시오.

한 대의 컴퓨터, 네 개의 데이터베이스 파티션 서버

클러스터 환경을 사용하지 않으면서 ServerA라는 하나의 실제 워크스테이션에 네 개의 데이터베이스 파티션 서버를 지정하려면 db2nodes.cfg 파일을 다음과 같이 갱신해야 합니다.

```
0          ServerA        0
1          ServerA        1
2          ServerA        2
3          ServerA        3
```

두 대의 컴퓨터, 컴퓨터당 하나의 데이터베이스 파티션 서버

파티션된 데이터베이스 시스템에 ServerA와 ServerB라는 두 개의 실제 워크스테이션을 포함시키려면 db2nodes.cfg 파일을 다음과 같이 갱신하십시오.

```
0          ServerA        0
1          ServerB        0
```

두 대의 컴퓨터, 컴퓨터당 세 개의 데이터베이스 파티션

파티션된 데이터베이스 시스템에 ServerA와 ServerB라는 두 개의 물리적 워크스테이션을 포함시키고 ServerA가 세 개의 데이터베이스 파티션 서버를 실행 중이면 db2nodes.cfg 파일을 다음과 같이 갱신하십시오.

```
4          ServerA        0
6          ServerA        1
8          ServerA        2
9          ServerB        0
```

두 대의 컴퓨터, 고속 스위치가 있는 세 개의 데이터베이스 파티션 서버

파티션된 데이터베이스 시스템에 ServerA와 ServerB라는(ServerB는 두 개의 데이터베이스 파티션 서버를 실행 중) 두 대의 컴퓨터를 포함시키고 switch1과 switch2라는 신속한 상호 연결을 사용할 경우, db2nodes.cfg 파일을 다음과 같이 갱신하십시오.

```
0          ServerA        0              switch1
1          ServerB        0              switch2
2          ServerB        1              switch2
```

resourcesetname을 사용한 예

이러한 제한사항은 다음의 예에 적용됩니다.

- 이 예는 구성에 신속한 상호 연결이 없을 경우 *resourcesetname*을 사용하는 것을 나타냅니다.
- *netname*은 네 번째 컬럼이며, 전환 이름이 없고 *resourcesetname*을 사용하려는 경우 *hostname*도 해당 컬럼에 지정할 수 있습니다. *resourcesetname*이 정의된 경우, 이는 다섯 번째 매개변수입니다. 자원 그룹 지정은 db2nodes.cfg 파일에 다섯 번째 컬럼으로만 표시될 수 있습니다. 이는 자원 그룹을 지정하기 위해서는 다섯 번째 컬럼도 입력해야 함을 의미합니다. 네 번째 컬럼은 고속 스위치용입니다.
- 고속 스위치가 없거나 이를 사용하지 않는 경우에는 *hostname*(두 번째 컬럼과 동일함)을 입력해야 합니다. 즉, DB2 데이터베이스 관리 시스템은 db2nodes.cfg 파일에서 컬럼 갭(또는 이를 교환하는 것)을 지원하지 않습니다. 이 제한사항은 처음 세 컬럼에 이미 적용되어 있으며 현재는 5개의 컬럼에 모두 적용됩니다.

AIX 예

다음 예에서는 AIX 운영 체제용 자원 세트를 설정하는 방법을 설명합니다.

이 예에는 32개의 프로세서와 8개의 논리적 데이터베이스 파티션(MLN)이 있는 한 개의 물리적 노드가 있습니다. 이 예는 각 MLN에 프로세스 친화도를 제공하는 방법을 설명합니다.

1. /etc/rset에 자원 세트를 정의하십시오.

```
DB2/MLN1:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00000,sys/cpu.00001,sys/cpu.00002,sys/cpu.00003

DB2/MLN2:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00004,sys/cpu.00005,sys/cpu.00006,sys/cpu.00007

DB2/MLN3:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00008,sys/cpu.00009,sys/cpu.00010,sys/cpu.00011

DB2/MLN4:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00012,sys/cpu.00013,sys/cpu.00014,sys/cpu.00015

DB2/MLN5:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00016,sys/cpu.00017,sys/cpu.00018,sys/cpu.00019

DB2/MLN6:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00020,sys/cpu.00021,sys/cpu.00022,sys/cpu.00023

DB2/MLN7:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00024,sys/cpu.00025,sys/cpu.00026,sys/cpu.00027

DB2/MLN8:
        owner     = db2inst1
        group     = system
        perm      = rwr-r-
        resources = sys/cpu.00028,sys/cpu.00029,sys/cpu.00030,sys/cpu.00031
```

2. 다음 명령을 입력하여 메모리 친화도를 사용 가능으로 설정하십시오.

```
vmo -p -o memory_affinity=1
```

3. 자원 세트를 사용하기 위한 인스턴스 권한을 부여하십시오.

```
chuser capabilities=
        CAP_BYPASS_RAC_VMM,CAP_PROPAGATE,CAP_NUMA_ATTACH  db2inst1
```

4. 자원 세트 이름을 db2nodes.cfg의 다섯 번째 컬럼으로 추가하십시오.

```
1 regatta 0 regatta DB2/MLN1
2 regatta 1 regatta DB2/MLN2
3 regatta 2 regatta DB2/MLN3
4 regatta 3 regatta DB2/MLN4
5 regatta 4 regatta DB2/MLN5
6 regatta 5 regatta DB2/MLN6
7 regatta 6 regatta DB2/MLN7
8 regatta 7 regatta DB2/MLN8
```

HP-UX 예

이 예에서는 4개의 CPU 및 4개의 MLN이 있는 머신에서 CPU 공유를 위해 PRM 그룹을 사용하는 방법을 표시하며, MLN 당 24%의 CPU 공유를 설정하고 4%는 다른 응용프로그램을 위해 남겨두고자 합니다. DB2 인스턴스 이름은 db2inst1입니다.

1. /etc/prmconf의 GROUP 섹션을 편집하십시오.

```
   OTHERS:1:4::
db2prm1:50:24::
  db2prm2:51:24::
    db2prm3:52:24::
  db2prm4:53:24::
```

2. /etc/prmconf에 인스턴스 소유자 항목을 추가하십시오.

```
db2inst1::::OTHERS,db2prm1,db2prm2,db2prm3,db2prm4
```

3. 다음 명령을 입력하여 그룹을 초기화하고 CPU 관리 프로그램을 사용 가능하게 하십시오.

```
prmconfig -i
prmconfig -e CPU
```

4. db2nodes.cfg에 PRM 그룹 이름을 다섯 번째 컬럼으로 추가하십시오.

```
1 voyager 0 voyager db2prm1
2 voyager 1 voyager db2prm2
3 voyager 2 voyager db2prm3
4 voyager 3 voyager db2prm4
```

대화식 GUI 도구인 xprm을 사용하여 PRM 구성(단계 1-3)을 완료할 수 있습니다.

Linux 예

Linux 운영 체제에서 *resourcesetname* 컬럼은 시스템의 NUMA(Non-Uniform Memory Access) 노드에 해당하는 번호를 정의합니다. NUMA 규정을 지원하는 2.6 커널에 추가하여 numactl 시스템 유틸리티를 사용할 수 있습니다. Linux 운영 체제에서 NUMA 지원에 관한 자세한 정보는 numact1의 man 페이지를 참조하십시오.

이 예에서는 NUMA 노드와 연관된 각 논리 노드가 있는 네 개의 NUMA 노드 컴퓨터를 설정하는 방법이 나와 있습니다.

1. 시스템에 NUMA 기능이 있는지 확인하십시오.
2. 다음 명령을 발행하십시오.

   ```
   $ numactl --hardware
   ```

   다음과 유사한 출력이 표시됩니다.

   ```
   available: 4 nodes (0-3)
   node 0 size: 1901 MB
   node 0 free: 1457 MB
   node 1 size: 1910 MB
   node 1 free: 1841 MB
   node 2 size: 1910 MB
   node 2 free: 1851 MB
   node 3 size: 1905 MB
   node 3 free: 1796 MB
   ```

3. 이 예에서는 시스템에 네 개의 NUMA 노드가 있습니다. db2nodes.cfg 파일을 다음과 같이 편집하여 각 MLN을 시스템의 NUMA 노드와 연관시키십시오.

   ```
   0 hostname 0 hostname 0
   1 hostname 1 hostname 1
   2 hostname 2 hostname 2
   3 hostname 3 hostname 3
   ```

Solaris 예

다음은 Solaris 버전 9에 대한 프로젝트를 설정하는 방법의 예입니다.

이 예에는 8개의 프로세서가 있는 하나의 실제 노드가 있습니다. 디폴트 프로젝트에 하나의 CPU가 사용되고 응용프로그램 서버에 세 개의 CPU가 사용되며 DB2에 네 개의 CPU가 사용됩니다. 인스턴스 이름은 db2inst1입니다.

1. 편집기를 사용하여 자원 풀 구성 파일을 작성하십시오. 이 예에서 파일은 pool.db2라고 합니다. 내용은 다음과 같습니다.

   ```
   create system hostname
      create pset pset_default (uint pset.min = 1)
      create pset db0_pset (uint pset.min = 1; uint pset.max = 1)
      create pset db1_pset (uint pset.min = 1; uint pset.max = 1)
      create pset db2_pset (uint pset.min = 1; uint pset.max = 1)
      create pset db3_pset (uint pset.min = 1; uint pset.max = 1)
      create pset appsrv_pset (uint pset.min = 3; uint pset.max = 3)
      create pool pool_default (string pool.scheduler="TS";
                boolean pool.default = true)
      create pool db0_pool (string pool.scheduler="TS")
      create pool db1_pool (string pool.scheduler="TS")
      create pool db2_pool (string pool.scheduler="TS")
      create pool db3_pool (string pool.scheduler="TS")
      create pool appsrv_pool (string pool.scheduler="TS")
      associate pool pool_default (pset pset_default)
      associate pool db0_pool (pset db0_pset)
      associate pool db1_pool (pset db1_pset)
      associate pool db2_pool (pset db2_pset)
      associate pool db3_pool (pset db3_pset)
      associate pool appsrv_pool (pset appsrv_pset)
   ```

2. 다음과 같이 /etc/project 파일을 편집하여 DB2 프로젝트 및 appsrv 프로젝트를 추가하십시오.

```
system:0::::
   user.root:1::::
   noproject:2::::
   default:3::::
   group.staff:10::::
   appsrv:4000:App Serv project:root::project.pool=appsrv_pool
   db2proj0:5000:DB2 Node 0 project:db2inst1,root::project.pool=db0_pool
   db2proj1:5001:DB2 Node 1 project:db2inst1,root::project.pool=db1_pool
   db2proj2:5002:DB2 Node 2 project:db2inst1,root::project.pool=db2_pool
   db2proj3:5003:DB2 Node 3 project:db2inst1,root::project.pool=db3_pool
```

3. 자원 풀을 작성하십시오. # poolcfg -f pool.db2.
4. 자원 풀을 활성화하십시오. # pooladm -c
5. 프로젝트 이름을 db2nodes.cfg 파일의 다섯 번째 컬럼으로 추가하십시오.

```
0 hostname 0 hostname db2proj0
   1 hostname 1 hostname db2proj1
   2 hostname 2 hostname db2proj2
   3 hostname 3 hostname db2proj3
```

파티션된 데이터베이스 환경에서 머신 목록 지정

디폴트로 컴퓨터 목록은 노드 구성 파일인 db2nodes.cfg에서 가져옵니다.

다음을 수행하여 파일을 겹쳐쓸 수 있습니다.

- 환경 변수 RAHOSTFILE을 익스포트하거나(Linux 및 UNIX 플랫폼) 설정하여(Windows) 컴퓨터 목록을 포함하는 파일의 경로 이름 지정
- 환경 변수 RAHOSTLIST를 익스포트하거나(Linux 및 UNIX 플랫폼) 이 변수를 설정하여(Windows) 스페이스로 구분된 일련의 이름으로 목록을 명시적으로 지정

  주: 이들 환경 변수가 모두 지정되면 RAHOSTLIST가 우선됩니다.

주: Windows에서 노드 구성 파일에 일관되지 않은 내용이 삽입되지 않게 하려면, 이를 수동으로 편집하지 않도록 하십시오. 인스턴스에서 컴퓨터 목록을 구하려면 db2nlist 명령을 사용하십시오.

파티션된 데이터베이스 환경에서 머신 목록의 중복 항목 제거

한 컴퓨터의 다중 논리 데이터베이스 파티션 서버에서 DB2 Enterprise Server Edition을 실행 중인 경우, db2nodes.cfg 파일에 해당 컴퓨터에 대한 여러 개의 항목이 포함됩니다.

이 경우 rah 명령에는 사용자가 이 명령을 각 컴퓨터에서 한 번만 실행할지 또는 db2nodes.cfg 파일에 나열된 각 논리 데이터베이스 파티션에 대해 한 번 실행할지 여부를 지정해야 합니다. 컴퓨터를 지정하려면 rah 명령을 사용하고, 논리 데이터베이스 파티션을 지정하려면 db2_all 명령을 사용하십시오.

주: Linux 및 UNIX 플랫폼에서 컴퓨터를 지정하는 경우, rah는 보통 다음 예외 상황을 제외하고 컴퓨터 목록에서 중복된 컴퓨터를 제거합니다. 논리 데이터베이스 파티션을 지정할 때, db2_all이 명령의 다음 사항을 사전에 보류할 경우는 예외 상황입니다.

```
export DB2NODE=nnn  (Korn 쉘 구문의 경우)
```

여기서, nnn은 db2nodes.cfg 파일의 해당 행에서 가져온 데이터베이스 파티션 번호이므로, 이 명령의 경로는 원하는 데이터베이스 파티션 서버로 지정됩니다.

논리 데이터베이스 파티션을 지정할 때에는 <<-nnn< 및 <<+nnn< 접두부 시퀀스를 사용하여 하나만을 제외한 모든 논리 데이터베이스 파티션을 포함하도록 목록을 제한하거나 하나만을 지정할 수 있습니다. 최초 데이터베이스 파티션을 카탈로그화하기 위한 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 파티션의 데이터베이스 파티션 번호를 알아야 합니다.

rah 명령을 사용하여 db2 restart database를 실행하면 컴퓨터 목록에서 중복된 항목이 제거됩니다. 그러나 " 접두부를 지정하는 경우, " 접두부를 사용하는 것은 각 컴퓨터가 아닌 각 데이터베이스 파티션 서버로 보내는 것으로 간주되므로, 중복된 항목이 제거되지 않습니다.

노드 구성 파일 갱신(Linux 및 UNIX)

이 태스크에서는 참여 컴퓨터에 대한 항목을 포함시키기 위해 db2nodes.cfg 파일을 갱신하는 단계를 설명합니다.

인스턴스 소유자의 홈 디렉토리에 있는 노드 구성 파일(db2nodes.cfg)에는 서버가 파티션된 데이터베이스 인스턴스 환경에 참여하고 있음을 DB2에 알려주는 구성 정보가 포함되어 있습니다. 파티션된 데이터베이스 환경에는 각 인스턴스에 대한 db2nodes.cfg 파일이 있습니다.

db2nodes.cfg 파일에는 인스턴스에 참여할 각 서버에 대한 항목이 하나 포함되어야 합니다. 인스턴스 작성 시 db2nodes.cfg 파일이 자동으로 작성되고 인스턴스 소유 서버에 대한 항목이 추가됩니다.

예를 들어, DB2 설치 마법사를 사용하여 ServerA라는 인스턴스 소유 서버에 DB2 인스턴스를 작성하면 다음과 같이 db2nodes.cfg 파일이 갱신됩니다.

```
0     ServerA     0
```

전제 조건

- 참여하는 모든 컴퓨터에 DB2 응용프로그램을 설치해야 합니다.
- 기본 컴퓨터에 DB2 인스턴스가 있어야 합니다.
- SYSADM 권한을 가진 사용자여야 합니다.

- 다음 조건 중 하나라도 적용되면 DB2 노드 구성 파일 주제 형식으로 제공되는 파일 형식 정보 및 구성 예를 검토하십시오.
  - 데이터베이스 파티션 서버 간 통신에 고속 전환을 사용할 계획입니다.
  - 파티션된 구성에 여러 논리적 파티션이 있습니다.

제한사항

프로시저 섹션의 단계에서 사용되는 hostname은 완전한 hostname이어야 합니다.

db2nodes.cfg 파일을 갱신하려면 다음과 같이 수행하십시오.

1. 인스턴스 소유자로 로그온하십시오(예에서는 db2inst1이 인스턴스 소유자임).
2. 다음 명령을 입력하여 DB2 인스턴스가 중지되었는지 확인하십시오.

   INSTHOME/sqllib/adm/db2stop

   여기서 *INSTHOME*은 인스턴스 소유자의 홈 디렉토리입니다. (db2nodes.cfg 파일은 인스턴스가 실행 중일때 잠기므로, 인스턴스가 중지되어야만 편집할 수 있습니다.)

   예를 들어, 인스턴스 홈 디렉토리가 /db2home/db2inst1인 경우, 다음 명령을 입력하십시오.

   ```
   /db2home/db2inst1/sqllib/adm/db2stop
   ```

3. 각 DB2 인스턴스에 대한 항목을 .rhosts 파일에 추가하십시오. 다음을 추가하여 파일을 갱신하십시오.

   ```
   <hostname> <db2instance>
   ```

   여기서 <hostname>은 데이터베이스 서버의 TCP/IP 호스트 이름이고 <db2instance>는 데이터베이스 서버에 액세스하는 데 사용하는 인스턴스의 이름입니다.

4. 참여하는 각 서버의 db2nodes.cfg 파일에 항목을 추가하십시오. 맨 처음 db2nodes.cfg 파일에는 다음과 같은 항목이 하나 있어야 합니다.

   ```
   0     ServerA     0
   ```

   이 항목은 데이터베이스 파티션 서버 번호(노드 번호), 데이터베이스 파티션 서버가 있는 서버의 TCP/IP 호스트 이름 및 데이터베이스 파티션 서버의 논리적 포트 번호를 포함합니다.

   예를 들어, 네 개의 컴퓨터를 가진 파티션된 구성 및 각 컴퓨터에 데이터베이스 파티션 서버를 설치하는 경우, 갱신된 db2nodes.cfg는 다음과 유사하게 표시됩니다.

   ```
   0          ServerA        0
   1          ServerB        0
     2     ServerC     0
     3     ServerD     0
   ```

5. db2nodes.cfg 파일 갱신을 완료하면 INSTHOME/sqllib/adm/db2start 명령을 입력하십시오. 여기서 *INSTHOME*은 인스턴스 소유자의 홈 디렉토리입니다. 예를 들어, 인스턴스 홈 디렉토리가 /db2home/db2inst1인 경우, 다음 명령을 입력하십시오.

   ```
   /db2home/db2inst1/sqllib/adm/db2start
   ```

6. 로그아웃하십시오.

다중 논리적 파티션 설정

일반적으로 각 컴퓨터에 하나의 데이터베이스 파티션 서버가 지정되도록 DB2 Enterprise Server Edition을 구성하십시오. 그러나 동일한 컴퓨터에서 여러 개의 데이터베이스 파티션 서버를 실행하는 것이 유리한 여러 상황이 있습니다.

이는 구성이 컴퓨터보다 더 많은 데이터베이스 파티션을 포함할 수 있음을 의미합니다. 이 경우, *동일한* 인스턴스에 참여한다면 *다중 논리 파티션* 또는 *다중 논리* 노드가 컴퓨터에서 실행 중이라고 합니다. 다른 인스턴스에 참여하는 경우, 이 컴퓨터는 다중 논리 파티션을 호스트하고 있지 *않은 것입니다*.

다중 논리 파티션 지원을 사용하여, 다음 세 가지 유형의 구성 중에서 선택할 수 있습니다.

- 각 컴퓨터에 오직 하나의 데이터베이스 파티션 서버만 있는 표준 구성
- 한 컴퓨터에 둘 이상의 데이터베이스 파티션 서버가 있는 다중 논리 파티션 구성
- 다중 논리 파티션이 각 여러 컴퓨터에서 실행되는 구성

다중 논리 파티션을 사용하는 구성은 대칭 멀티프로세서(SMP) 아키텍처를 갖는 컴퓨터에서 시스템이 쿼리를 실행할 때 유용합니다. 한 컴퓨터에서 다중 논리 파티션을 구성하는 기능은 컴퓨터가 실패할 경우 유용합니다. 컴퓨터가 실패하는 경우(데이터베이스 파티션 서버의 장애를 야기함) START DBM DBPARTITIONNUM 명령을 사용하여 다른 컴퓨터에서 데이터베이스 파티션 서버를 다시 시작할 수 있습니다. 이것은 사용자 데이터를 사용 가능하게 합니다.

또 다른 이점은 다중 논리 파티션이 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.

다중 논리적 파티션 구성

다중 논리적 파티션 구성은 두 가지 방법이 있습니다.

- db2nodes.cfg 파일에 논리 파티션(데이터베이스 파티션)을 구성합니다. 그런 다음, db2start 명령 및 연관된 API를 사용하여 논리 파티션과 리모트 파티션을 모두 시작할 수 있습니다.

주: Windows의 경우, 시스템에 데이터베이스가 없는 경우 db2ncrt를 사용하여 파티션을 추가하거나, 하나 이상의 데이터베이스가 있는 경우 db2start addnode 명령을 사용하여 노드를 추가해야 합니다. Windows에서는 db2nodes.cfg 파일을 수동으로 편집해서는 안 됩니다.

- 다른 논리 파티션(노드)이 이미 실행 중인 다른 프로세서에서 논리 파티션을 재시작합니다. 이 방법을 사용하면 db2nodes.cfg에서 논리 파티션에 대해 지정된 호스트 이름과 포트 번호를 겹쳐쓸 수 있습니다.

db2nodes.cfg에 논리 파티션(노드)을 구성하려면, 해당 데이터베이스 파티션에 대한 논리 포트 번호를 할당하는 항목을 파일에 작성해야 합니다. 다음 구문을 사용해야 합니다.

```
nodenumber hostname logical-port netname
```

주: Windows의 경우, 시스템에 데이터베이스가 없는 경우 db2ncrt를 사용하여 파티션을 추가하거나, 하나 이상의 데이터베이스가 있는 경우 db2start addnode 명령을 사용하여 노드를 추가해야 합니다. Windows에서는 db2nodes.cfg 파일을 수동으로 편집해서는 안 됩니다.

Windows에서 db2nodes.cfg 파일의 형식은 UNIX의 동일한 파일과 비교할 때 다릅니다. Windows에서 컬럼 형식은 다음과 같습니다.

```
nodenumber hostname computername logical_port netname
```

호스트 이름에 완전한 이름을 사용하십시오. /etc/hosts 파일도 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 서비스 파일에 충분한 포트를 정의했는지 확인해야 합니다.

파티션 간 쿼리 병렬 처리 사용

파티션 간 병렬 처리는 이러한 데이터베이스 파티션의 데이터베이스 파티션 수와 데이터 분산에 기초하여 자동으로 발생합니다.

주: 데이터베이스 파티션 내에서 또는 파티션되지 않은 데이터베이스 내에서 병렬 처리를 사용하려면 구성 매개변수를 수정하십시오. 예를 들어, 파티션 내 병렬 처리를 사용하여 대칭적 다중 프로세서(SMP) 머신에서 다중 프로세서를 사용할 수 있습니다.

데이터 로딩에 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 파티션 간 병렬 처리는 다중 데이터베니스 파티션에 목표 테이블이 정의될 때 자동으로 발생합니다. 데이터 로드의 파티션 간 병렬 처리는 OUTPUT_DBPARTNUMS를 지정하여 겹쳐쓸 수 있습니다. 로드 유틸리티는 목표 데이터베이스 파티션 *크기*에 따라 데이터베이스 파티션 병렬 처리를 인공 지능식으로 사용 가능하게 할 수도 있습니다. MAX_NUM_PART_AGENTS를 사용하여 로드 유틸리티가 선택한 병렬 처리의 최대 수준을 제어할 수 있습니다. 데이터베이스 파티션 병렬 처리는 ANYORDER도 지정되어 있는 경우 PARTITIONING_DBPARTNUMS를 지정하여 겹쳐쓸 수 있습니다.

인덱스 작성에 병렬 처리 사용

인덱스를 작성할 때 병렬 처리를 사용하려면, 다음과 같은 상태여야 합니다.

- 테이블은 병렬 처리에서 도움이 될 수 있도록 커야 합니다.
- SMP 컴퓨터에서 다중 프로세서가 사용 가능 상태여야 합니다.

데이터베이스 또는 테이블 스페이스를 백업할 때 입출력 병렬 처리 사용

데이터베이스 또는 테이블 스페이스를 백업할 때 입출력 병렬 처리를 사용하려면, 다음을 수행하십시오.

1. 둘 이상의 목표 미디어를 사용하십시오.
2. 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. 필요할 때마다 이러한 사항을 고려할 수 없으므로, 데이터베이스나 테이블 스페이스를 백업하기 전에 미리 계획을 세워야 합니다.
3. BACKUP 명령에서 PARALLELISM 매개변수를 사용하여 병렬 처리 수준을 지정하십시오.
4. BACKUP 명령에서 WITH `num-buffers` BUFFERS 매개변수를 사용하여 병렬 처리 수준을 수용할 만큼 버퍼가 충분한지 확인하십시오. 버퍼 수는 소유하고 있는 목표 미디어의 수에 선택된 병렬 처리 수준을 더한 값과 같아야 합니다.

   또한 다음과 같은 백업 버퍼 *크기*를 사용하십시오.

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

데이터베이스 또는 테이블 스페이스를 리스토어할 때 입출력 병렬 처리 사용

데이터베이스 또는 테이블 스페이스를 리스토어할 때 입출력 병렬 처리를 사용하려면, 다음을 수행하십시오.

- 둘 이상의 소스 미디어를 사용하십시오.
- 병렬 I/O에 대한 테이블 스페이스를 구성하십시오. 컨테이너를 정의하기 전에 이 옵션의 사용을 결정해야 합니다. 필요할 때마다 이 옵션의 사용을 결정할 수 없으므로, 데이터베이스 또는 테이블 스페이스를 리스토어하기 전에 미리 계획을 세워야 합니다.
- RESTORE 명령에서 PARALLELISM 매개변수를 사용하여 병렬 처리 수준을 지정하십시오.
- RESTORE 명령에서 WITH num-buffers BUFFERS 매개변수를 사용하여 병렬 처리 수준을 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 *크기*를 사용하십시오.

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

쿼리에 파티션 내 병렬 처리 사용

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 매개변수에는 *max_querydegree* 및 *intra_parallel* 데이터베이스 관리 프로그램 매개변수, *dft_degree* 데이터베이스 매개변수가 포함됩니다.

파티션 내 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 파티션 내 병렬 처리에서 사용되지 않습니다.

max_querydegree

이 인스턴스에서 실행되는 SQL문에 사용되는 파티션 내 병렬 처리의 최대 수준을 지정하는 데이터베이스 관리 프로그램 구성 매개변수. SQL문은 데이터베이스 파티션 내에서 병렬 조작을 실행할 때 이 매개변수가 제공하는 숫자보다 큰 숫자를 사용하지 않습니다. *intra_parallel* 구성 매개변수의 *max_querydegree* 의 사용 여부 값이 『예』로 설정되어 있어야 합니다. 이 구성 매개변수의 디폴트값은 -1입니다. 이 값은 시스템이 옵티마이저가 판별한 병렬 처리 수준을 사용함을 의미합니다. 그렇지 않으면 사용자 지정 값을 사용합니다.

dft_degree

DEGREE 바인드 옵션 및 CURRENT DEGREE 특수 레지스터에 대한 디폴트값을 제공하는 데이터베이스 구성 매개변수입니다. 1. 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

데이터 서버 용량 관리

데이터 서버 용량이 현재 또는 추후의 요구에 부합하지 않는 경우 디스크 스페이스를 추가하고 추가 컨테이너를 작성하거나, 메모리를 추가하여 용량을 늘릴 수 있습니다. 이러한 간단한 방법으로 필요한 용량을 추가하지 못하는 경우에는 프로세서 또는 실제 파티션을 추가하십시오. 환경을 변경하여 시스템을 확장하는 경우에는 이러한 변경이 데이터 로딩 또는 백업 및 데이터베이스 리스토어와 같은 데이터베이스 프로시저에 미칠 수 있는 영향에 유의해야 합니다.

프로세서 추가

단일 프로세서를 가진 단일 파티션 데이터베이스 구성을 최대 용량까지 사용하는 경우 프로세서를 추가하거나 논리적 파티션을 추가하십시오. 프로세서를 추가하면 처리 능력이 향상되는 장점이 있습니다. SMP 시스템에서 프로세서는 메모리 및 스토리지 시스템 자원을 공유합니다. 모든 프로세서가 하나의 시스템에 있으므로 시스템 간의 통신 및 시스템 간 태스크 조정과 같은 추가적인 오버헤드 고려사항이 없습니다. 로드, 백업 및 리스토어와 같은 유틸리티에서 추가적인 프로세서를 사용할 수 있습니다.

주: Solaris 운영 체제와 같은 일부 운영 체제에서는 동적으로 프로세서를 온라인 및 오프라인으로 설정할 수 있습니다.

프로세서를 추가하는 경우 사용되는 프로세서 수를 판별하는 몇몇 데이터베이스 구성 매개변수를 검토하고 수정하십시오. 다음 데이터베이스 구성 매개변수가 사용되는 프로세서 수를 판별하며, 매개변수를 갱신해야 할 수도 있습니다.

- 디폴트 등급(dft_degree)
- 최대 병렬 처리 수준(max_querydegree)
- 파티션 내 병렬 처리 사용(intra_parallel)

응용프로그램에서 병렬 처리를 수행하는 방법을 판별하는 매개변수도 평가해야 합니다.

통신에 TCP/IP를 사용하는 환경에서는 DB2TCPCONNMGRS 레지스트리 변수의 값을 검토하십시오.

추가적인 컴퓨터 추가

기존 파티션된 데이터베이스 환경이 있는 경우에는 추가적인 컴퓨터(단일 프로세서 또는 다중 프로세서) 및 스토리지 자원을 환경에 추가하여 처리 능력과 데이터 스토리지 용량을 늘릴 수 있습니다. 컴퓨터 간에 메모리와 스토리지 자원이 공유되지 않습니다. 이 방법을 선택하면 스토리지 및 컴퓨터 전반에서 데이터와 사용자 액세스 밸런스를 유지할 수 있는 장점이 있습니다.

새 컴퓨터와 스토리지를 추가한 후에 START DATABASE MANAGER 명령을 사용하여 새 컴퓨터에 새 데이터베이스 파티션 서버를 추가할 수 있습니다. 추가한 각각의 새 데이터베이스 파티션 서버에 있는 인스턴스의 데이터베이스마다 새 데이터베이스 파티션이 작성 및 구성됩니다. 대부분의 경우 새 데이터베이스 파티션 서버 추가 후 인스턴스를 재시작할 필요가 없습니다.

FCM(Fast Communication Manager)

FCM(Fast Communication Manager) (Windows)

FCM(Fast Communication Manager)은 동일한 인스턴스에 속하는 DB2 서버 제품에 통신 지원을 제공합니다. 각 데이터베이스 파티션 서버마다 하나의 FCM 송신기 및 하나의 FCM 수신기 디먼이 있어서 데이터베이스 파티션 서버 간에 통신을 제공하여 에이전트 요청을 처리하고 메시지 버퍼를 전달합니다. FCM 디먼은 사용자가 인스턴스를 시작할 때 시작됩니다.

데이터베이스 파티션 서버 사이에 통신에 실패하거나 통신을 다시 설정하는 경우, FCM 스레드는 정보를 갱신합니다. 데이터베이스 시스템 모니터를 사용하여 이 정보를 쿼리

할 수 있습니다. FCM 디먼은 또한 적합한 조치를 트리거하기도 합니다. 적합한 조치의 예는 영향 받은 트랜잭션의 롤백입니다. FCM 구성 매개변수 설정시 데이터베이스 시스템 모니터를 사용하면 도움이 됩니다.

fcm_num_buffers 데이터베이스 관리 프로그램 구성 매개변수로 FCM 메시지 버퍼 수를 지정하고 *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수로 FCM 채널 수를 지정할 수 있습니다. *fcm_num_buffers* 및 *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수는 디폴트값으로 AUTOMATIC으로 설정됩니다. 이러한 매개변수 중 하나라도 AUTOMATIC으로 설정되면 FCM은 자원 사용을 모니터하고 점차적으로 자원을 릴리스합니다. 이러한 매개변수는 AUTOMATIC으로 설정한 상태로 남겨두는 것이 좋습니다.

FCM(Fast Communication Manager) (Linux 및 UNIX)

FCM(Fast Communication Manager)은 데이터베이스 파티션 기능(DPF)을 사용하는 DB2 서버 제품에 통신 지원을 제공합니다.

다중 파티션 인스턴스의 경우, 각 데이터베이스 파티션 서버마다 하나의 FCM 송신기 디먼 및 하나의 FCM 수신기 디먼이 있어서 데이터베이스 파티션 서버 간에 통신을 제공하여 에이전트 요청을 처리하고 메시지 버퍼를 전달합니다. FCM 디먼은 사용자가 다중 파티션 인스턴스를 시작할 때 시작됩니다.

데이터베이스 파티션 서버간에 통신이 실패하거나, 통신을 재설정하는 경우에는 FCM 디먼이 정보를 갱신합니다. 데이터베이스 시스템 모니터를 사용하여 이 정보를 쿼리할 수 있습니다. FCM 디먼은 또한 적합한 조치를 트리거하기도 합니다. 적합한 조치의 예는 영향 받은 트랜잭션의 롤백입니다. FCM 구성 매개변수 설정 시 데이터베이스 시스템 모니터를 사용하면 도움이 됩니다.

fcm_num_buffers 데이터베이스 관리 프로그램 구성 매개변수로 FCM 메시지 버퍼 수를 지정할 수 있습니다. 또한, *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수로 FCM 채널 수를 지정할 수도 있습니다. *fcm_num_buffers* 및 *fcm_num_channels* 데이터베이스 관리 프로그램 구성 매개변수는 디폴트값으로 AUTOMATIC으로 설정됩니다. 이러한 매개변수 중 하나라도 AUTOMATIC으로 설정되면 FCM은 자원 사용을 모니터하고 점차적으로 자원을 릴리스합니다. 이러한 매개변수는 AUTOMATIC으로 설정한 상태로 남겨두는 것이 좋습니다.

FCM 통신을 사용하여 데이터베이스 파티션 간 통신 사용

파티션된 데이터베이스 환경에서는 FCM(Fast Communication Manager)으로 데이터베이스 파티션 간의 대부분 통신이 이루어집니다.

데이터베이스 파티션에서 FCM을 사용하고 다른 데이터베이스 파티션과의 통신을 사용하려면, 아래 표시된 것처럼 `etc` 디렉토리의 데이터베이스 파티션의 `services` 파일에

서 서비스 항목을 작성해야 합니다. FCM은 지정된 포트를 사용하여 통신합니다. 같은 호스트에 다중 데이터베이스 파티션을 정의한 경우, 다음과 같이 포트 범위를 정의해야 합니다.

FCM(Fast Communication Manager)에 대한 메모리를 수동으로 구성하기 전에, FCM 버퍼(*fcm_num_buffers*) 수 및 FCM 버퍼(*fcm_num_channels*) 수에 대한 디폴트 설정인 자동 설정으로 시작하도록 권장합니다. 해당 설정이 적절한지 판별하려면 FCM 활동에 대한 시스템 모니터 데이터를 사용하십시오.

Windows 고려사항

Windows 환경에서 DB2 Enterprise Server Edition을 사용할 경우, TCP/IP 포트 범위는 다음에 의해 서비스 파일에 자동으로 추가됩니다.

- 인스턴스를 작성하거나 새 데이터베이스 파티션을 추가할 때의 설치 프로그램
- 새 인스턴스를 작성할 때의 db2icrt 유틸리티
- 컴퓨터에 첫 번째 데이터베이스 파티션을 추가할 때의 db2ncrt 유틸리티

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 경우, 동일한 고려사항이 적용되며 추가로 다음 사항을 고려해야 합니다. DB2 데이터베이스 인스턴스에 대해 정의된 항목이 `etc` 디렉토리의 모든 `services` 파일에서 동일하도록 해야 합니다(파티션된 데이터베이스 환경에 적용되지 않는 다른 항목은 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, 다음과 같습니다.

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

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

데이터베이스 파티션 서버 간 통신 사용(Linux 및 UNIX)

이 태스크에서는 파티션된 데이터베이스 시스템에 참여하는 데이터베이스 파티션 서버 간에 통신을 사용 가능하게 하는 방법에 대해 설명합니다. 데이터베이스 파티션 서버 간의 통신은 FCM(Fast Communication Manager)으로 처리됩니다. FCM을 사용하려면, 파티션된 데이터베이스 시스템의 각 컴퓨터에 있는 /etc/services 파일에서 포트 또는 포트 범위를 예약해야 합니다.

루트 권한을 가진 사용자 ID가 있어야 합니다.

인스턴스에 참여하는 모든 컴퓨터에서 이 태스크를 수행해야 합니다.

FCM용으로 예약하는 포트 수는 인스턴스의 컴퓨터에서 호스트하거나 잠정적으로 호스트하는 데이터베이스 파티션의 최대 수와 동일합니다.

다음 예에서 db2nodes.cfg 파일에 이들 항목이 포함되어 있습니다.

```
0 server1 0
1 server1 1
2 server2 0
3 server2 1
4 server2 2
5 server3 0
6 server3 1
7 server3 2
8 server3 3
```

FCM 포트가 60000에서 시작하여 번호가 매겨진다고 가정합시다. 이 상황에서

- server1은 두 개의 데이터베이스 파티션에 두 개의 포트(60000, 60001)를 사용합니다.
- server2는 세 개의 데이터베이스 파티션에 세 개의 포트(60000, 60001, 60002)를 사용합니다.
- server3는 네 개이 데이터베이스 파티션에 대해 네 개의 포트(60000, 60001, 60002, 60003)를 사용합니다.

모든 컴퓨터는 60000, 60001, 60002 및 60003을 예약해야 하는데, 이것이 인스턴스의 모든 컴퓨터에서 필요로 하는 최대 포트 범위입니다.

HACMP™(High Availability Cluster Multi-Processing) 또는 Tivoli System Automation 과 같은 고가용성 솔루션을 사용해 한 시스템에서 다른 시스템으로 데이터베이스 파티션을 복구하는 경우 필요한 포트를 고려해야 합니다. 예를 들어 네 개의 데이터베이스 파티션을 호스트하는 시스템의 경우 다른 시스템에 있는 두 개의 데이터베이스 파티션이 해당 시스템에 대해 복구되는 경우를 대비하여 6개의 포트를 계획해야 합니다.

인스턴스를 작성할 경우, 기본 컴퓨터에 포트 범위가 예약됩니다. 기본 컴퓨터를 인스턴스 소유 컴퓨터라고도 합니다. 그러나 /etc/services 파일에 추가된 원래 포트 범위가 사용자 요구에 충분하지 않을 경우 수동으로 항목을 추가하여 예약된 포트 범위를 확장해야 합니다.

/etc/services을 사용하여 파티션된 데이터베이스 환경에서 서버 간에 통신을 사용가능하게 하려면 다음을 수행하십시오.

1. 루트 권한이 있는 사용자로 기본 컴퓨터(인스턴스 소유 컴퓨터)에 로그온하십시오.
2. 인스턴스를 작성하십시오.
3. /etc/services 파일에 예약된 디폴트 포트 범위를 확인하십시오. 기본 구성 외에도 FCM 포트는 다음과 유사하게 나타나야 합니다.

```
db2c_db2inst1         50000/tcp
#Add FCM port information
     DB2_db2inst1         60000/tcp
     DB2_db2inst1_1       60001/tcp
     DB2_db2inst1_2       60002/tcp
   DB2_db2inst1_END 60003/tcp
```

디폴트로 첫 번째 포트가 연결 요청용으로 예약되어 있고 60000 이상에서 사용 가능한 처음 네 개의 포트가 FCM 통신용으로 예약되어 있습니다. 인스턴스 소유 데이터베이스 파티션 서버용으로 하나의 포트와, 설치 후 컴퓨터에 추가하도록 선택할 수 있는 논리적 데이터베이스 파티션 서버용으로 세 개의 포트가 사용됩니다.

포트 범위에는 시작 및 END 항목이 포함되어야 합니다. 중간 항목은 선택적입니다. 명시적으로 중간 값을 포함시키는 것은 다른 응용프로그램이 이들 포트를 사용하지 못하게 하는 데에는 유용하지만 데이터베이스 관리 프로그램이 이들 항목의 유효성을 검증하지 못하게 합니다.

DB2 포트 항목은 다음과 같은 형식을 사용합니다.

```
DB2_instance_name_suffix  port_number/tcp  # comment
```

각 부분에 대한 설명은 다음과 같습니다.

- *instance_name*은 파티션된 인스턴스 이름입니다.
- *suffix*는 첫 번째 FCM 포트에 사용되지 않습니다. 중간 항목은 최하위와 최상위 포트 사이의 항목입니다. 첫 번째 및 마지막 FCM 포트 사이에 중간 항목을 포함시키는 경우 *suffix*는 각 추가 포트에 대해 1씩 증가되는 정수로 구성됩니다.

예를 들어 고유성을 유지하기 위해 두 번째 포트에는 1로 번호가 매겨지고 세 번째 포트에는 2로 번호가 매겨집니다. 단어 END는 마지막 항목의 *suffix*로 사용해야 합니다.

- *port_number*는 사용자가 데이터베이스 파티션 서버 통신용으로 예약한 포트 번호입니다.
- *comment*는 항목을 설명하는 선택적 주석입니다.

4. FCM 통신을 위해 충분한 포트를 예약하도록 하십시오. 예약된 포트 범위가 충분하지 않은 경우 새 항목을 파일에 추가하십시오.
5. 인스턴스에 참여하는 각 컴퓨터에 루트 사용자로 로그온하고 동일한 항목을 `/etc/services` 파일에 추가하십시오.

제 10 장 파티션된 데이터베이스 환경 작성 및 관리

초기 데이터베이스 파티션 그룹

데이터베이스가 초기에 작성될 때, 데이터베이스 파티션은 db2nodes.cfg 파일에 지정된 모든 데이터베이스 파티션용으로 작성됩니다. 나머지 데이터베이스 파티션은 ADD DBPARTITIONNUM 및 DROP DBPARTITIONNUM VERIFY 명령으로 추가 또는 제거할 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

데이터베이스 파티션 그룹 작성

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

컴퓨터 및 시스템이 사용 가능해야 하며 파티션된 데이터베이스 환경을 처리할 수 있어야 합니다. DB2 Enterprise Server Edition이(가) 구매 및 설치되어 있어야 하며 데이터베이스가 있어야 합니다.

이 명령문은 또한 다음 기능을 수행합니다.

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

제어 센터를 사용하여 데이터베이스 파티션 그룹을 작성하려면 다음을 수행하십시오.

1. **데이터베이스 파티션 그룹** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.

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 데이터베이스 파티션 상자에서 선택된 데이터베이스 파티션 상자로 데이터베이스 파티션을 이동시킨 후 **확인**을 누르십시오.

명령행을 사용하여 데이터베이스 파티션 그룹을 작성하려면 다음을 입력하십시오.

```
CREATE DATABASE PARTITION GROUP db-partition-group-name
   ON DBPARTITIONNUM (db-partition-number1,db-partition-number1)
```

또는

```
CREATE DATABASE PARTITION GROUP db-partition-group-name
   ON DBPARTITIONNUMS (db-partition-number1
   TO db-partition-number2)
```

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

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

또는

```
CREATE DATABASE PARTITION GROUP mixng12 ON DBPARTITIONNUMS (1 TO 2)
```

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

데이터베이스 파티션 그룹에서 테이블 스페이스

다중 파티션 데이터베이스 파티션 그룹에 테이블 스페이스를 배치하면, 테이블 스페이스 내의 모든 테이블이 데이터베이스 파티션 그룹의 각 데이터베이스 파티션으로 분할되거나 파티션됩니다.

테이블 스페이스는 하나의 데이터베이스 파티션 그룹에 작성됩니다. 테이블 스페이스가 임의 데이터베이스 파티션 그룹에 작성된 후에는 계속 남아 있어야 합니다. 다른 데이터베이스 파티션 그룹으로 변경될 수 없습니다. CREATE TABLESPACE문이 테이블 스페이스와 데이터베이스 파티션 그룹을 연관시키는데 사용됩니다.

데이터베이스 파티션 관리

제어 센터에서 파티션 보기를 사용하여 파티션을 시작 및 중지하고 파티션을 삭제 및 추적하고 진단 로그를 표시할 수 있습니다.

데이터베이스 파티션에 대해 작업하거나 DB2 로그를 보려면 인스턴스에 대한 접속 권한이 필요합니다. SECADM 또는 ACCESSCTRL 권한이 있는 사람은 누구나 특정 인스턴스에 액세스할 수 있는 권한을 부여할 수 있습니다.

IBM Support에서 요청할 경우, 표시되는 옵션을 사용하여 추적 유틸리티를 실행하십시오. 추적 유틸리티는 DB2 조작에 대한 정보를 기록하고 이 정보를 읽을 수 있는 형식으로 변경합니다. 자세한 정보는 『db2trc - 추적: DB2』 주제를 참조하십시오.

주의: DB2 고객 서비스 센터나 기술 지원 센터에서 요청한 경우에만 추적 기능을 사용하십시오.

진단 로그 창을 사용하여 DB2 추적 유틸리티가 로그한 텍스트 정보를 보십시오.

파티션 뷰가 다음 정보를 표시합니다.

노드 번호

이 컬럼에는 아이콘과 노드 번호가 포함되어 있습니다. 노드 번호는 고유 번호이고, 0에서 99까지 가능합니다. 번호는 db2nodes.cfg 파일에 저장되어 있습니다. 노드 번호는 오름차순으로 표시되지만 순서에 간격이 있을 수 있습니다.

일단 지정한 노드 번호는 변경될 수 없습니다. 이 세이프가드는 파티션 맵의 정보(데이터가 파티션되는 방법을 자세히 설명하는)가 손상되지 않도록 합니다.

호스트 이름

호스트 이름은 FCM(Fast Communication Manager)이 내부 통신용으로 사용하는 IP 주소입니다. (그러나, 스위치 이름이 지정된 경우, FCM은 스위치 이름을 사용합니다. 이 경우 호스트 이름이 START DBM, STOP DBM 및 db2_all에서만 사용됩니다.) 호스트 이름은 db2nodes.cfg 파일에 저장됩니다.

포트 번호

포트 번호는 노드에 대한 논리 포트 번호입니다. 이 번호는 데이터베이스 관리 인스턴스 이름과 함께 사용되어, /etc/services 파일의 TCP/IP 서비스 이름 항목을 식별합니다. 이 번호는 db2nodes.cfg 파일에 저장됩니다.

IP 주소(호스트 이름)와 논리 포트의 결합은 잘 알려진 주소로 사용되며, 노드 간의 통신 연결을 지원하기 위해 모든 응용프로그램간에 고유해야 합니다.

표시된 각 호스트 이름에 대해 한 개의 포트 번호는 0입니다. 포트 번호 0은 클라이언트가 연결하는 호스트의 디폴트 노드를 표시합니다. (이 작동을 대체하려면, db2profile 스크립트의 **DB2NODE** 환경 변수를 사용하십시오.)

스위치 이름

스위치 이름(또는 네트이름)은 각기 자체의 호스트 이름을 가지는 사용 중인 TCP/IP 인터페이스가 여러 개 있는 호스트를 지원하기 위해 사용됩니다. 또한 고속 스위치를 공유하는 노드 간 빠른 통신(FCM이라고도 함)에도 사용됩니다.

스위치 이름은 db2nodes.cfg 파일에 저장됩니다. 스위치 이름이 db2nodes.cfg 파일에 지정되지 않은 경우, 호스트 이름과 같은 이름입니다.

Resourcesetname

resourcesetname은 노드가 시작되는 운영 체제 자원을 정의합니다. resourcesetname은 프로세스 친화도 지원을 위해 다중 논리적 참고(MLN)에 사용되며 이전에 quadname으로 알려진 문자열 유형 필드에 제공됩니다.

1. 파티션 보기 열기:제어 센터에서 파티션을 볼 인스턴스를 찾을 때까지 오브젝트 트리를 펼치십시오. 원하는 인스턴스에서 마우스 오른쪽 단추를 눌러 팝업 메뉴에서 **열기 → 파티션**을 선택하십시오. 파티션 보기가 열립니다.
2. 파티션을 시작하려면 하나 이상의 파티션을 강조표시한 후 **파티션 → 시작**을 선택하십시오. 선택된 파티션이 시작됩니다.
3. 파티션을 중지하려면 하나 이상의 파티션을 강조표시한 후 **파티션 → 중지**를 선택하십시오. 선택된 파티션이 중지됩니다.
4. 파티션에서 추적 유틸리티를 실행하려면 다음을 수행하십시오.
   a. DB2 추적 창을 여십시오. 파티션을 강조표시한 후 **파티션 → 서비스 → 추적**을 선택하십시오. DB2 추적 창이 열립니다.
   b. 추적 옵션을 지정하십시오.
   c. **시작**을 눌러 정보 레코딩을 시작하고 **중지**를 눌러 정보 레코딩을 중지한 후 정보를 파일에 저장하려면 **다른 이름으로 저장**을 누르십시오.
   d. 선택사항: 로그를 보십시오.
   e. 요청이 있으면, 추적 결과물을 IBM Support로 보내십시오.

파티션된 데이터베이스 환경에서 데이터베이스 파티션 추가

시스템이 실행 중이거나 중지될 때 데이터베이스 파티션을 파티션된 데이터베이스에 추가할 수 있습니다. 새 서버를 추가하는데 시간이 걸리기 때문에, 데이터베이스 관리 프로그램이 이미 실행 중일 때 추가해야 합니다.

ADD DBPARTITIONNUM 명령을 사용하여 데이터베이스 파티션을 시스템에 추가하십시오. 이 명령은 다음 방법으로 호출될 수 있습니다.

- START DBM의 옵션으로
- 명령행 처리기 ADD DBPARTITIONNUM 명령을 사용하여 호출될 수 있습니다.
- API 함수 sqleaddn을 사용하여 호출될 수 있습니다.
- API 함수 sqlepstart를 사용하여 호출될 수 있습니다.

시스템이 중지되어 있는 경우, START DBM를 사용합니다. 시스템이 실행 중인 경우, 기타 모든 선택항목을 사용할 수 있습니다.

ADD DBPARTITIONNUM 명령을 사용하여 새 데이터베이스 파티션을 시스템에 추가하면, 인스턴스의 모든 기존 데이터베이스가 새 데이터베이스 파티션에 확장됩니다. 또한 데이터베이스에 대한 임시 테이블 스페이스에 사용할 컨테이너를 지정할 수도 있습니다. 컨테이너는 다음이 될 수 있습니다.

- 각 데이터베이스의 카탈로그 파티션에 대해 정의된 컨테이너와 같습니다. (이 컨테이너는 디폴트값입니다.)
- 다른 데이터베이스 파티션에 정의된 것과 같습니다.
- 전혀 작성되지 않았습니다. 데이터베이스를 사용하기 전에, ALTER TABLESPACE 문을 사용하여 각 데이터베이스에 임시 테이블 스페이스 컨테이너를 추가해야 합니다.

주: 카탈로그 해제된 데이터베이스는 새 데이터베이스 파티션을 추가할 때 인식되지 않습니다. 카탈로그 해제된 데이터베이스는 새 데이터베이스 파티션에 존재하지 않습니다. 새 데이터베이스 파티션에서 데이터베이스에 연결하려고 하면 오류 메시지 SQL1013N이 리턴됩니다.

하나 이상의 데이터베이스 파티션 그룹이 새 데이터베이스 파티션을 포함할 수 있도록 변경되어야 새 데이터베이스 파티션의 데이터베이스를 사용하여 데이터를 포함할 수 있습니다.

시스템에 데이터베이스 파티션을 추가하는 것만으로 단일 파티션 데이터베이스에서 다중 파티션 데이터베이스로 변경할 수 없습니다. 이는 데이터를 데이터베이스 파티션에 재분배하려면 영향을 받은 각 테이블에 분산 키가 있어야 하기 때문입니다. 다중 파티션 데이터베이스에서 테이블이 작성될 때 분산 키가 자동으로 생성됩니다. 단일 파티션 데이터베이스에서는 CREATE TABLE 또는 ALTER TABLE SQL문으로 분산 키를 명시적으로 작성할 수 있습니다.

주: 시스템에 정의된 데이터베이스가 없고 UNIX 운영 체제에서 Enterprise Server Edition을 실행 중인 경우, `db2nodes.cfg` 파일을 편집하여 새 데이터베이스 파티션 정의를 추가하십시오. 데이터베이스가 존재할 때만 적용되므로 설명된 프로시저를 사용하지 마십시오.

Windows 고려사항: Windows에서 Enterprise Server Edition을 사용 중이고 인스턴스에 데이터베이스가 없는 경우, db2ncrt 명령을 사용하여 데이터베이스 시스템을 스케일하십시오. 그러나 이미 데이터베이스가 있는 경우, START DBM ADD DBPARTITIONNUM 명령을 사용하여 시스템을 스케일할 때 기존의 각 데이터베이스에 데이터베이스 파티션이 작성되어 있는지 확인하십시오. Windows에서 파일에 불일치가 발생할 수 있으므로 노드 구성 파일(`db2nodes.cfg`)을 수동으로 편집해서는 안됩니다.

실행 중인 데이터베이스 시스템에 데이터베이스 파티션 추가

파티션된 데이터베이스 환경이 실행되고 있고 응용프로그램이 데이터베이스에 연결되어 있는 동안 새 데이터베이스 파티션을 파티션된 데이터베이스 시스템에 추가할 수 있습니다. 그러나 데이터베이스 관리 프로그램을 종료하여 재시작한 때부터 모든 데이터베이스에서 새로 추가한 데이터베이스 파티션을 사용할 수 있습니다.

명령행을 사용하여 실행 중인 데이터베이스 관리 프로그램에 데이터베이스 파티션을 추가하려면 다음을 수행하십시오.

1. 기존 데이터베이스 파티션에서, START DBM 명령을 실행하십시오.

   모든 플랫폼에서 DBPARTITIONNUM, ADD DBPARTITIONNUM, HOSTNAME, PORT 및 NETNAME 매개변수에 대한 새 데이터베이스 파티션 값을 지정하십시오. Windows 플랫폼에서 COMPUTER, USER 및 PASSWORD 매개변수를 지정할 수도 있습니다.

   또한 데이터베이스와 함께 작성되어야 하는 임시 테이블 스페이스 컨테이너 정의에 대한 소스를 지정할 수도 있습니다. 테이블 스페이스 정보를 제공하지 않은 경우, 임시 테이블 스페이스 컨테이너 정의가 각 데이터베이스에 대한 카탈로그 파티션으로부터 검색됩니다.

   START DBM 명령이 완료되면 새 서버가 중지됩니다.
2. STOP DBM 명령을 실행하여 모든 데이터베이스 파티션에서 데이터베이스 관리 프로그램을 중지시키십시오.

   시스템의 모든 데이터베이스 파티션를 중지시키는 경우, 노드 구성 파일은 새 데이터베이스 파티션을 포함하도록 갱신됩니다. 노드 구성 파일은 STOP DBM이 실행될 때까지 새 서버 정보로 갱신되지 않습니다. 이것은 ADD DBPARTITIONNUM 명령(ADD DBPARTITIONNUM 매개변수를 START DBM 명령으로 지정할 때 호출됨)이 올바른 데이터베이스 파티션에서 실행되도록 하기 위해서 입니다. 유틸리티가 종료되면 새 서버 파티션이 중지됩니다.
3. START DBM 명령을 실행하여 데이터베이스 관리 프로그램을 중지시키십시오.

   새로 추가된 데이터베이스 파티션은 시스템의 나머지 파티션과 함께 이제 시작됩니다.

   시스템의 모든 데이터베이스 파티션이 실행 중이면, 전체 시스템 활동(예: 데이터베이스 작성 또는 삭제)을 실행할 수 있습니다.

   주: 새 db2nodes.cfg 파일에 액세스하기 위해 모든 데이터베이스 파티션 서버에 대해 START DBM 명령을 두 번 발행해야 할 수도 있습니다.

4. 선택사항: 새 데이터베이스 파티션을 포함하도록 데이터베이스 파티션 그룹을 변경하십시오. 또한 이 조치는 새 데이터베이스 파티션에 데이터를 재분배할 때 선택사항이 될 수 있습니다.
5. 선택사항: 새 데이터베이스 파티션에 데이터를 재분배하십시오. 새 데이터베이스 파티션을 이용하려는 경우 이 조치는 실제로 선택적이 아닙니다. 또한 데이터베이스 파티션 그룹 변경 옵션을 재분배 연산의 파트로 포함할 수 있습니다. 그렇지 않으면, 새 데이터베이스 파티션에 데이터를 재분배하기 전에 새 데이터베이스 파티션을 포함하도록 데이터베이스 파티션 그룹 변경을 별도의 조치로 수행해야 합니다.
6. 선택사항: 새 데이터베이스 파티션에서 모든 데이터베이스를 백업하십시오. 선택적이지만 이는 특히 이전 및 새 데이터베이스 파티션에 걸쳐 데이터를 재분배한 경우 새 데이터베이스 파티션 및 기타 데이터베이스 파티션용으로 포함하는 것이 도움이 됩니다.

데이터베이스 파티션을 추가하기 위해 온라인으로 작업 시 제한사항

인스턴스에 새 데이터베이스 파티션을 추가한 후 이 데이터베이스 파티션의 상태는 원래 데이터베이스 파티션의 상태에 따라 결정됩니다. 응용프로그램이 WITH HOLD 커서를 사용하는 경우 인스턴스에 새 데이터베이스 파티션을 추가한 후 이 데이터베이스 파티션을 인식할 수도, 인식하지 못할 수도 있습니다.

단일 파티션 데이터베이스 인스턴스에 새 데이터베이스 파티션 추가 시:

- 데이터베이스 파티션이 추가될 때 원래 데이터베이스 파티션이 위에 있으면 데이터베이스 파티션 추가 연산이 완료될 때 새 데이터베이스 파티션은 아래에 있습니다.
- 데이터베이스 파티션이 추가될 때 원래 데이터베이스 파티션이 아래에 있으면 데이터베이스 파티션 추가 연산이 완료될 때 새 데이터베이스 파티션은 위에 있습니다.

데이터베이스 파티션 추가 연산이 실행되기 전에 시작되는 WITH HOLD 커서를 사용하면 응용프로그램이 데이터베이스 파티션 추가 연산이 완료될 때 새 데이터베이스 파티션을 인식하지 않습니다. 데이터베이스 파티션 추가 연산이 실행되기 전에 WITH HOLD 커서가 닫히는 경우, 응용프로그램이 데이터베이스 파티션 추가 연산이 완료될 때 새 데이터베이스 파티션을 인식합니다.

중지된 데이터베이스 시스템에 데이터베이스 파티션 추가(Windows)

파티션된 데이터베이스 시스템이 중지될 때 새 데이터베이스 파티션을 파티션된 데이터베이스 시스템에 추가할 수 있습니다. 데이터베이스 관리 프로그램이 재시작되면 새로 추가된 데이터베이스 파티션을 모든 데이터베이스에서 사용할 수 있습니다.

데이터베이스 파티션을 작성하기 전에 새 서버를 설치해야 합니다.

명령행을 사용하여 중지된 파티션된 데이터베이스 서버에 데이터베이스 파티션을 추가하려면 다음을 수행하십시오.

1. 모든 데이터베이스 파티션을 중지하려면 STOP DBM을 발행하십시오.
2. 새 서버에서 ADD DBPARTITIONNUM 명령을 실행하십시오.

   시스템에 이미 존재하는 모든 데이터베이스에 대해 데이터베이스 파티션이 로컬로 작성됩니다. 새 데이터베이스 파티션의 데이터베이스 매개변수는 디폴트로 설정되며, 각 데이터베이스 파티션은 사용자가 데이터를 파티션에 이동시킬 때까지 비어 있습니다. 데이터베이스 구성 매개변수 값을 갱신하여 다른 데이터베이스 파티션에서 값을 일치시키십시오.
3. START DBM 명령을 실행하여 데이터베이스 시스템을 시작하십시오. 노드 구성 파일(cfg)은 새 서버 설치 중에 새 서버를 포함하도록 데이터베이스 관리 프로그램에서 이미 갱신되었습니다.
4. 다음과 같이 새 데이터베이스 파티션에서 구성 파일을 갱신하십시오.
   a. 기존 데이터베이스 파티션에서, START DBM 명령을 실행하십시오.

      COMPUTER, USER 및 PASSWORD 매개변수뿐만 아니라 DBPARTITIONNUM, ADDDBPARTITIONNUM, HOSTNAME, PORT 및 NETNAME 매개변수에 대한 새 데이터베이스 파티션 값도 지정하십시오.

      또한 데이터베이스와 함께 작성되어야 하는 임시 테이블 스페이스 컨테이너 정의에 대한 소스를 지정할 수도 있습니다. 테이블 스페이스 정보를 제공하지 않은 경우, 임시 테이블 스페이스 컨테이너 정의가 각 데이터베이스에 대한 카탈로그 파티션으로부터 검색됩니다.

      START DBM 명령이 완료되면 새 서버가 중지됩니다.
   b. STOP DBM 명령을 실행하여 전체 데이터베이스 관리 프로그램을 중지시키십시오.

      시스템의 모든 데이터베이스 파티션를 중지시키는 경우, 노드 구성 파일은 새 데이터베이스 파티션을 포함하도록 갱신됩니다. 노드 구성 파일은 STOP DBM이 실행될 때까지 새 서버 정보로 갱신되지 않습니다. 이것은 ADD DBPARTITIONNUM 명령(ADDDBPARTITIONNUM 매개변수를 START DBM 명령으로 지정할 때 호출됨)이 올바른 데이터베이스 파티션에서 실행되도록 하기 위해서 입니다. 유틸리티가 종료되면 새 서버 파티션이 중지됩니다.
5. START DBM 명령을 실행하여 데이터베이스 관리 프로그램을 중지시키십시오.

   새로 추가된 데이터베이스 파티션은 시스템의 나머지 파티션과 함께 이제 시작됩니다.

   시스템의 모든 데이터베이스 파티션이 실행 중이면, 전체 시스템 활동(예: 데이터베이스 작성 또는 삭제)을 실행할 수 있습니다.

주: 새 db2nodes.cfg 파일에 액세스하기 위해 모든 데이터베이스 파티션 서버에 대해 START DBM 명령을 두 번 발행해야 할 수도 있습니다.

6. 선택사항: 새 데이터베이스 파티션을 포함하도록 데이터베이스 파티션 그룹을 변경하십시오. 또한 이 조치는 새 데이터베이스 파티션에 데이터를 재분배할 때 선택사항이 될 수 있습니다.
7. 선택사항: 새 데이터베이스 파티션에 데이터를 재분배하십시오. 새 데이터베이스 파티션을 이용하려는 경우 이 조치는 실제로 선택적이 아닙니다. 또한 데이터베이스 파티션 그룹 변경 옵션을 재분배 연산의 파트로 포함할 수 있습니다. 그렇지 않으면, 새 데이터베이스 파티션에 데이터를 재분배하기 전에 새 데이터베이스 파티션을 포함하도록 데이터베이스 파티션 그룹 변경을 별도의 조치로 수행해야 합니다.
8. 선택사항: 새 데이터베이스 파티션에서 모든 데이터베이스를 백업하십시오. 선택적이지만 이는 특히 이전 및 새 데이터베이스 파티션에 걸쳐 데이터를 재분배한 경우 새 데이터베이스 파티션 및 기타 데이터베이스 파티션용으로 포함하는 것이 도움이 됩니다.

중지된 데이터베이스 시스템에 데이터베이스 파티션 추가(UNIX)

파티션된 데이터베이스 시스템이 중지될 때 새 데이터베이스 파티션을 추가할 수 있습니다. 데이터베이스 관리 프로그램이 재시작되면 새로 추가된 데이터베이스 파티션을 모든 데이터베이스에서 사용할 수 있습니다.

데이터베이스 파티션을 작성하기 위해서는 먼저 서버가 존재하지 않는 경우 새 서버를 설치해야 합니다. 또한 준비사항은 다음 태스크를 포함해야 합니다.

- 실행 파일을 액세스할 수 있도록 만듦(공유 파일 시스템 마운트 또는 로컬 사본을 사용하여)
- 운영 체제 파일과 기존 프로세스의 운영 체제 파일 동기화
- sqllib 디렉토리를 공유 파일 시스템으로서 액세스할 수 있는지 확인
- 관련 운영 체제 매개변수(예: 프로세스의 최대 수)가 적절한 값으로 설정되었는지 확인

또한 모든 데이터베이스 파티션에서 /etc 디렉토리의 .hosts 파일에 등록하거나 이름 서버로 호스트 이름을 등록해야 합니다. rsh 또는 rah를 사용하여 리모트 명령을 실행하기 위해서는 컴퓨터의 호스트 이름이 .rhosts에 등록되어 있어야 합니다.

명령행을 사용하여 중지된 파티션된 데이터베이스 서버에 데이터베이스 파티션을 추가하려면 다음을 수행하십시오.

1. 모든 데이터베이스 파티션을 중지하려면 STOP DBM을 발행하십시오.
2. 새 서버에서 ADD DBPARTITIONNUM 명령을 실행하십시오.

   시스템에 이미 존재하는 모든 데이터베이스에 대해 데이터베이스 파티션이 로컬로 작성됩니다. 새 데이터베이스 파티션의 데이터베이스 매개변수는 디폴트로 설정되며,

각 데이터베이스 파티션은 사용자가 데이터를 파티션에 이동시킬 때까지 비어 있습니다. 데이터베이스 구성 매개변수 값을 갱신하여 다른 데이터베이스 파티션에서 값을 일치시키십시오.

3. START DBM 명령을 실행하여 데이터베이스 시스템을 시작하십시오. 노드 구성 파일(cfg)은 새 서버 설치 중에 새 서버를 포함하도록 데이터베이스 관리 프로그램에서 이미 갱신되었습니다.
4. 다음과 같이 새 데이터베이스 파티션에서 구성 파일을 갱신하십시오.
   a. 기존 데이터베이스 파티션에서, START DBM 명령을 실행하십시오.

      COMPUTER, USER 및 PASSWORD 매개변수뿐만 아니라 DBPARTITIONNUM, ADD DBPARTITIONNUM, HOSTNAME, PORT 및 NETNAME 매개변수에 대한 새 데이터베이스 파티션 값도 지정하십시오.

      또한 데이터베이스와 함께 작성되어야 하는 임시 테이블 스페이스 컨테이너 정의에 대한 소스를 지정할 수도 있습니다. 테이블 스페이스 정보를 제공하지 않은 경우, 임시 테이블 스페이스 컨테이너 정의가 각 데이터베이스에 대한 카탈로그 파티션으로부터 검색됩니다.

      START DBM 명령이 완료되면 새 서버가 중지됩니다.

   b. STOP DBM 명령을 실행하여 전체 데이터베이스 관리 프로그램을 중지시키십시오.

      시스템의 모든 데이터베이스 파티션를 중지시키는 경우, 노드 구성 파일은 새 데이터베이스 파티션을 포함하도록 갱신됩니다. 노드 구성 파일은 STOP DBM이 실행될 때까지 새 서버 정보로 갱신되지 않습니다. 이것은 ADD DBPARTITIONNUM 명령(ADD DBPARTITIONNUM 매개변수를 START DBM 명령으로 지정할 때 호출됨)이 올바른 데이터베이스 파티션에서 실행되도록 하기 위해서 입니다. 유틸리티가 종료되면 새 서버 파티션이 중지됩니다.
5. START DBM 명령을 실행하여 데이터베이스 관리 프로그램을 중지시키십시오.

   새로 추가된 데이터베이스 파티션은 시스템의 나머지 파티션과 함께 이제 시작됩니다.

   시스템의 모든 데이터베이스 파티션이 실행 중이면, 전체 시스템 활동(예: 데이터베이스 작성 또는 삭제)을 실행할 수 있습니다.

   주: 새 `db2nodes.cfg` 파일에 액세스하기 위해 모든 데이터베이스 파티션 서버에 대해 START DBM 명령을 두 번 발행해야 할 수도 있습니다.
6. 선택사항: 새 데이터베이스 파티션을 포함하도록 데이터베이스 파티션 그룹을 변경하십시오. 또한 이 조치는 새 데이터베이스 파티션에 데이터를 재분배할 때 선택사항이 될 수 있습니다.

7. 선택사항: 새 데이터베이스 파티션에 데이터를 재분배하십시오. 새 데이터베이스 파티션을 이용하려는 경우 이 조치는 실제로 선택적이 아닙니다. 또한 데이터베이스 파티션 그룹 변경 옵션을 재분배 연산의 파트로 포함할 수 있습니다. 그렇지 않으면, 새 데이터베이스 파티션에 데이터를 재분배하기 전에 새 데이터베이스 파티션을 포함하도록 데이터베이스 파티션 그룹 변경을 별도의 조치로 수행해야 합니다.
8. 선택사항: 새 데이터베이스 파티션에서 모든 데이터베이스를 백업하십시오. 선택적이지만 이는 특히 이전 및 새 데이터베이스 파티션에 걸쳐 데이터를 재분배한 경우 새 데이터베이스 파티션 및 기타 데이터베이스 파티션용으로 포함하는 것이 도움이 됩니다.

또한 다음과 같이 구성 파일을 수동으로 갱신할 수도 있습니다.

1. db2nodes.cfg 파일을 편집하고 여기에 새 데이터베이스 파티션을 추가하십시오.
2. 다음 명령을 발행하여 새 데이터베이스 파티션을 시작하십시오. `START DBM DBPARTITIONNUM partitionnum`

   새 데이터베이스 파티션에 지정할 수를 partitionnum 값으로 지정하십시오.
3. 새 서버가 논리 파티션이어야 하는 경우(즉, 데이터베이스 파티션이 0이 아닌 경우), db2set 명령을 사용하여 DBPARTITIONNUM 레지스트리 변수를 갱신하십시오. 추가 중인 데이터베이스 파티션의 수를 지정하십시오.
4. 새 데이터베이스 파티션에서 ADD DBPARTITIONNUM 명령을 실행하십시오.

   이 명령은 시스템에 이미 존재하는 모든 데이터베이스에 대해 데이터베이스 파티션을 로컬로 작성합니다. 새 데이터베이스 파티션의 데이터베이스 매개변수는 디폴트로 설정되며, 각 데이터베이스 파티션은 사용자가 데이터를 파티션에 이동시킬 때까지 비어 있습니다. 데이터베이스 구성 매개변수 값을 갱신하여 다른 데이터베이스 파티션에서 값을 일치시키십시오.
5. ADD DBPARTITIONNUM 명령이 완료되면 START DBM 명령을 발행하여 시스템의 다른 데이터베이스 파티션을 시작하십시오.

   모든 데이터베이스 파티션이 성공적으로 시작될 때까지 전체 시스템 활동(예: 데이터베이스 작성 또는 삭제)을 수행하지 마십시오.

데이터베이스 파티션 추가 시 오류 복구

데이터베이스 관리 프로그램에서 시스템 버퍼 풀을 작성하여 모든 버퍼 풀 페이지 크기에 디폴트 자동 지원을 제공하므로 존재하지 않는 버퍼 풀로 인해 데이터베이스 파티션 추가에 실패하지 않습니다. 그러나 이러한 시스템 버퍼 풀 중 하나를 사용할 경우, 데이터베이스 관리 프로그램에서 작성한 시스템 버퍼 풀이 매우 작기 때문에 성능에 상당한 영향을 미칠 수 있습니다. 시스템 버퍼 풀이 사용될 경우, 관리 통지 로그에 메시지가 작성됩니다.

시스템 버퍼 풀은 다음 환경의 데이터베이스 파티션 추가 시나리오에서 사용됩니다.

- 페이지 *크기*가 4KB의 디폴트 페이지 *크기*와 다른 하나 이상의 시스템 임시 테이블 스페이스가 있는 파티션된 데이터베이스 환경에 데이터베이스 파티션을 추가합니다. 데이터베이스 파티션이 작성되면, IBMDEFAULTDP 버퍼 풀만 존재하고 이 버퍼 풀의 페이지 *크기*는 4KB입니다.

  다음 예를 고려해 보십시오.

  1. START DBM 명령을 사용하여 현재 다중 파티션 데이터베이스에 데이터베이스 파티션을 추가합니다.

     ```
     START DBM DBPARTITIONNUM 2 ADD DBPARTITIONNUM HOSTNAME newhost PORT 2
     ```

  2. `db2nodes.cfg` 파일을 새 데이터베이스 파티션 설명으로 수동 갱신한 후 ADD DBPARTITIONNUM 명령을 사용합니다.

  이러한 문제를 방지하는 한 가지 방법은 ADD DBPARTITIONNUM 또는 START DBM 명령에 WITHOUT TABLESPACES절을 지정하는 것입니다. 이렇게 한 후, CREATE BUFFERPOOL 문을 사용하여 해당 SIZE 및 PAGESIZE 값을 사용하여 버퍼 풀을 작성하고 ALTER TABLESPACE문을 사용하여 시스템 임시 테이블 스페이스를 버퍼 풀과 연관시켜야 합니다.
- 페이지 *크기*가 4KB의 디폴트 페이지 *크기*와 다른 하나 이상의 테이블 스페이스가 있는 기존 데이터베이스 파티션 그룹에 데이터베이스 파티션을 추가합니다. 새 데이터베이스 파티션에서 작성된 디폴트가 아닌 페이지 *크기* 버퍼 풀이 테이블 스페이스에 대해 활성화되지 않았기 때문에 발생합니다.

  주: 이전 버전에서, 이 명령은 데이터베이스 파티션 그룹 키워드 대신 NODEGROUP 키워드를 사용합니다.

  다음 예를 고려하십시오.

  – ALTER DATABASE PARTITION GROUP문을 사용하여 다음과 같이 데이터베이스 파티션 그룹에 데이터베이스 파티션을 추가합니다.

    ```
    START DBM
       CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (2)
    ```

    이 문제를 방지하는 한 가지 방법은 다음 ALTER 데이터베이스 파티션 그룹을 발행하기 전에 각 페이지 *크기*에 대한 버퍼 풀을 작성한 다음, 데이터베이스에 다시 연결하는 것입니다.

    ```
    START DBM
       CONNECT TO mpp1
       CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
    CONNECT RESET
       CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (2)
    ```

주: 데이터베이스 파티션 그룹에 디폴트 페이지 *크기*의 테이블 스페이스가 있는 경우 메시지 SQL1759W가 리턴됩니다.

데이터베이스 파티션 삭제

데이터베이스가 사용하지 않는 데이터베이스 파티션을 삭제하여 다른 용도의 컴퓨터 여유 공간을 확보할 수 있습니다.

DROP DBPARTITIONNUM VERIFY 명령 또는 sqledrpn API를 발행하여 데이터베이스 파티션을 사용 중이 아닌지 확인하십시오.

- 메시지 SQL6034W(어떠한 데이터베이스에서도 사용되지 않은 데이터베이스 파티션)를 수신한 경우, 데이터베이스 파티션을 삭제할 수 있습니다.
- 메시지 SQL6035W(데이터베이스에 사용 중인 데이터베이스 파티션)를 수신하는 경우, REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 삭제할 데이터베이스 파티션에서 데이터베이스 별명과 다른 데이터베이스 파티션으로 데이터를 재분배하십시오.

또한 이 데이터베이스 파티션이 코디네이터 역할을 한 모든 트랜잭션이 성공적으로 커미트되거나 롤백되었는지 확인하십시오. 이를 위해 다른 서버에서 응급 복구를 할 수도 있습니다. 예를 들어, 코디네이터 파티션을 삭제하고 코디네이터 파티션이 삭제되기 전에 트랜잭션에 참여한 다른 데이터베이스 파티션이 손상되면, 손상된 데이터베이스 파티션은 인다우트 트랜잭션 결과에 대해 코디네이터 파티션을 쿼리할 수 없습니다.

명령행을 사용하여 데이터베이스 파티션을 삭제하려면 DROP DBPARTITIONNUM 매개변수와 함께 STOP DBM 명령을 발행하여 데이터베이스 파티션을 삭제하십시오. 명령이 성공적으로 완료한 후에 시스템은 중지됩니다.그런 다음 데이터베이스 관리 프로그램을 START DBM 명령과 함께 시작하십시오.

인스턴스에 데이터베이스 파티션 서버 나열

Windows에서는 db2nlist 명령을 사용하여 인스턴스에 참여하는 데이터베이스 파티션 서버 목록을 확보할 수 있습니다.

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, instName은 원하는 특정 인스턴스 이름입니다.

다음을 사용하여 선택적으로 각 데이터베이스 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.

인스턴스에 데이터베이스 파티션 서버 추가 (Windows)

Windows에서 db2ncrt 명령을 사용하여 인스턴스에 데이터베이스 파티션 서버를 추가하십시오.

주: 인스턴스에 이미 데이터베이스가 있으면, db2ncrt 명령을 사용하지 마십시오. 대신 START DBM ADD DBPARTITIONNUM 명령을 사용하십시오. 이것은 데이터베이스가 새 데이터베이스 파티션 서버에 올바르게 추가되도록 보장합니다. 파일을 변경하면 파티션된 데이터베이스 환경에 불일치가 생길 수 있으므로 db2nodes.cfg 파일을 **편집하지 마십시오.**

명령에는 다음과 같은 필수 매개변수가 있습니다.

```
db2ncrt /n:partition_number
        /u:username,password
        /p:logical_port
```

/n:partition_number

데이터베이스 파티션 서버를 식별하기 위한 고유한 데이터베이스 파티션 번호이며 번호는 오름차순으로 1 - 999일 수 있습니다.

/u:username,password

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 컴퓨터에서 최초 데이터베이스 파티션을 작성할 때에만 선택적입니다. 논리 데이터베이스 파티션을 작성하는 경우, 이 매개변수를 지정하고 사용 중이지 않은 논리 포트 번호를 선택해야 합니다. 여기에는 여러 개의 제한사항이 있습니다.

- 모든 컴퓨터에는 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 데이터베이스 파티션입니다.) 포트 범위는 db2icrt가 /r:base_port, end_port 매개변수와 함께 사용될 때 정의됩니다.

또한 여러 개의 선택적 매개변수도 있습니다.

/g:network_name

데이터베이스 파티션 서버의 네트워크 이름을 지정합니다. 이 매개변수를 지정하지 않은 경우, DB2는 사용자 시스템에서 검출한 최초 IP 주소를 사용합니다.

컴퓨터에 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 컴퓨터에서 데이터베이스 파티션 서버를 추가하는 경우 필요합니다.

/i:instance_name

인스턴스 이름. 디폴트값은 현재 인스턴스입니다.

/m:computer_name

데이터베이스 파티션이 있는 Windows 워크스테이션의 컴퓨터 이름. 디폴트값은 로컬 컴퓨터의 컴퓨터 이름입니다.

/o:instance_owning_computer

인스턴스를 소유하는 컴퓨터의 컴퓨터 이름. 디폴트값은 로컬 컴퓨터입니다. 이 매개변수는 인스턴스 소유 컴퓨터가 아닌 모든 컴퓨터에서 db2ncrt 명령이 호출될 때 필수입니다.

예를 들어, 새 데이터베이스 파티션 서버를 인스턴스 소유 컴퓨터 MYMACHIN에 있는 인스턴스 TESTMPP에 추가하고(다중 논리 데이터베이스 파티션을 실행할 수 있도록), 이 새 데이터베이스 파티션을 논리 포트 1을 사용하여 데이터베이스 파티션 2로 알리려면, 다음을 입력하십시오.

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

파티션 추가 마법사를 사용하여 인스턴스에 데이터베이스 파티션 추가

파티션 추가 마법사를 사용하여 파티션을 작성하고 이를 하나 이상의 데이터베이스 파티션 그룹에 추가합니다. 제일 먼저 새 파티션을 인스턴스에 추가하고 해당 파티션을 하나 이상의 데이터베이스 파티션 그룹에 지정한 다음 고급 선택사항을 선택합니다.

데이터베이스 파티션 그룹에 대해 작업하려면 DBADM 권한이 있어야 합니다.

1. 파티션 추가 마법사를 여십시오.
   a. 제어 센터에서 작업할 인스턴스 오브젝트를 찾을 때까지 오브젝트 트리를 펼치십시오. 오브젝트를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 파티션 추가를 누르십시오. 파티션 추가 런치패드가 열립니다.
   b. **파티션 추가** 단추를 누르십시오. 파티션 추가 마법사가 열립니다.

2. 적용 가능한 각 마법사 페이지를 완료하십시오. 자세한 정보는 첫 번째 페이지의 마법사 개요 링크를 누르십시오. 마법사가 파티션을 추가하기에 충분한 정보를 완료하면 **완료** 누름 단추를 사용할 수 있습니다.

데이터베이스 파티션 변경(Windows)

Windows에서 db2nchg 명령을 사용하여 데이터베이스 파티션을 변경하십시오.

- 하나의 컴퓨터에서 다른 컴퓨터로 데이터베이스 파티션을 이동시킵니다.
- 컴퓨터의 TCP/IP 호스트 이름을 변경하십시오.

  다중 네트워크 어댑터를 사용하려는 경우, 이 명령을 사용하여 db2nodes.cfg 파일에 ″netname″ 필드의 TCP/IP 주소를 지정해야 합니다.
- 다른 논리 포트 번호를 사용합니다.
- 데이터베이스 파티션 서버에 대해 다른 이름을 사용합니다.

명령에는 다음과 같은 필수 매개변수가 있습니다.

```
db2nchg /n:node_number
```

매개변수 /n:은 변경하려는 데이터베이스 파티션 서버의 구성 번호입니다. 이 매개변수는 필수입니다.

선택적 매개변수는 다음과 같습니다.

/i:instance_name
이 데이터베이스 파티션 서버가 참여하는 인스턴스를 지정합니다. 이 매개변수를 지정하지 않은 경우, 디폴트값은 현재 인스턴스입니다.

/u:username,password
DB2 데이터베이스 서비스에 대한 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 컴퓨터로 이동시키려는 경우 이 매개변수를 지정해야 합니다. 이 매개변수를 지정하지 않은 경우, 논리 포트 번호는 변경되지 않습니다.

/h:host_name
FCM이 내부 통신에 사용하는 TCP/IP 호스트 이름을 변경하십시오. 이 매개변수를 지정하지 않은 경우, 호스트 이름은 변경되지 않습니다.

/m:computer_name
데이터베이스 파티션 서버를 다른 컴퓨터로 이동시킵니다. 데이터베이스 파티션 서버는 인스턴스에 기존 데이터베이스가 없는 경우에만 이동될 수 있습니다.

/g:network_name

데이터베이스 파티션 서버에 대한 네트워크 이름을 변경합니다.

컴퓨터에 다중 IP 주소가 있고 데이터베이스 파티션 서버의 특정 IP 주소를 사용하려는 경우, 이 매개변수를 사용하십시오. 네트워크 이름 또는 IP 주소를 사용하여 network_name을 입력할 수 있습니다.

예를 들어, 인스턴스 TESTMPP에 참여하는 데이터베이스 파티션 2에 지정된 논리 포트를 논리 포드 3을 사용하도록 변경하려면, 다음 명령을 입력하십시오.

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 데이터베이스 관리 프로그램은 리모트 컴퓨터에서 인스턴스 레벨의 DB2 데이터베이스 시스템 레지스트리 변수에 액세스할 수 있도록 합니다. 현재 DB2 데이터베이스 시스템 레지스트리 변수가 세 개의 다른 레벨(컴퓨터 또는 전역 레벨, 인스턴스 레벨, 데이터베이스 파티션 레벨)에 저장됩니다. DB2REMOTEPREG를 사용하면 인스턴스 레벨(데이터베이스 파티션 레벨 포함)에 저장된 레지스트리 변수를 다른 컴퓨터로 경로 재지정할 수 있습니다. DB2REMOTEPREG가 설정되면, DB2 데이터베이스 관리 프로그램은 DB2REMOTEPREG가 가리키는 컴퓨터에서 DB2 데이터베이스 시스템 레지스트리 변수에 액세스합니다. db2set 명령은 다음과 같이 표시됩니다.

```
db2set DB2REMOTEPREG=<remote workstation>
```

여기서 <리모트 워크스테이션>은 리모트 워크스테이션 이름입니다.

주:

- 모든 DB2 데이터베이스 인스턴스 프로파일 및 인스턴스 목록이 지정된 리모트 컴퓨터 이름에 있으므로 이 위치를 설정할 경우에는 주의해야 합니다.
- 환경에 도메인의 사용자가 포함된 경우, DB2 인스턴스 서비스와 연관된 로그온 어카운트가 도메인 어카운트인지 확인하십시오. DB2 인스턴스에 도메인 레벨에서 그룹을 열거하는데 적절한 특권이 있어야 합니다.

이들 기능을 설정 DBINSTPROF와 함께 사용하여 레지스트리가 있는 동일한 컴퓨터의 리모트 LAN 드라이브를 가리키도록 할 수 있습니다.

데이터베이스 파티션에서 SMS 테이블 스페이스에 컨테이너 추가

현재 컨테이너가 없는 데이터베이스 파티션의 SMS 테이블 스페이스에 컨테이너를 추가만 할 수 있습니다.

명령행을 사용하여 SMS 테이블 스페이스에 컨테이너를 추가하려면, 다음을 입력하십시오.

```
ALTER TABLESPACE <name>
      ADD ('<path>')
ON DBPARTITIONNUM (<database partition_number>)
```

번호로 지정된 데이터베이스 파티션 및 데이터베이스 파티션 범위의 모든 파티션이 테이블 스페이스가 정의된 데이터베이스 파티션 그룹에 있어야 합니다. 데이터베이스 partition_number는 명시적으로만 보이거나 명령문에 대해 정확히 하나의 *db-partitions-clause* 범위 내에서만 보일 수 있습니다.

다음 예에서는 테이블 스페이스 『plans』가 UNIX 기반 운영 체제에서 사용한 데이터베이스 파티션 그룹의 데이터베이스 파티션 번호 3에 새 컨테이너를 추가하는 방법을 보여줍니다.

```
ALTER TABLESPACE plans
      ADD ('/dev/rhdisk0')
ON DBPARTITIONNUM (3)
```

인스턴스에서 데이터베이스 파티션 삭제(Windows)

Windows에서 데이터베이스가 없는 인스턴스에서 데이터베이스 파티션 서버를 삭제하려면 db2ndrop 명령을 사용하십시오. 데이터베이스 파티션 서버를 제거하면, 데이터베이스 파티션 번호는 새 데이터베이스 파티션 서버에서 재사용될 수 있습니다.

인스턴스에서 데이터베이스 파티션 서버를 제거할 때에는 주의하십시오. 인스턴스에서 인스턴스를 소유하는 데이터베이스 파티션 서버 0을 제거하면, 인스턴스는 사용할 수 없게 됩니다. 인스턴스를 제거하려면, db2idrop 명령을 사용하십시오.

주: 이 인스턴스에 데이터베이스가 있으면, db2ndrop 명령을 사용하지 마십시오. 대신 STOP DBM DROP DBPARTITIONNUM 명령을 사용하십시오. 이렇게 하면, 데이터베이스는 데이터베이스 파티션에서 올바로 제거됩니다. 파일을 변경하면 파티션된 데이터베이스 환경에 불일치가 생길 수 있으므로 db2nodes.cfg 파일을 **편집하지 마십시오.**

다중 논리 데이터베이스 파티션 실행 중인 컴퓨터에서 논리 포트 0을 지정한 데이터베이스 파티션을 삭제하려는 경우, 논리 포트 0에 지정된 데이터베이스 파티션을 삭제하기 전에 다른 논리 포트에 지정된 다른 모든 데이터베이스 파티션을 삭제해야 합니다. 각 데이터베이스 파티션 서버는 논리 포트 0에 지정된 데이터베이스 파티션을 가져야 합니다.

명령에는 다음과 같은 매개변수가 있습니다.

```
db2ndrop /n:dbpartitionnum /i:instance_name
```

/n:dbpartitionnum

데이터베이스 파티션 서버를 식별하기 위한 고유한 데이터베이스 파티션 번호(dbpartitionnum)입니다. 이것은 필수 매개변수입니다. 번호는 오름차순으로 0 - 999일 수 있습니다. 데이터베이스 파티션 0은 인스턴스를 소유하는 컴퓨터를 나타낸다는 것을 상기하십시오.

/i:instance_name

인스턴스 이름(instance_name)입니다. 이 매개변수는 선택적입니다. 인스턴스 이름을 제공하지 않은 경우, 디폴트값은 현재 인스턴스(DB2INSTANCE 레지스트리 변수로 설정)입니다.

파티션 삭제 런치패드를 사용하여 인스턴스에서 데이터베이스 파티션 삭제

파티션 삭제 런치패드를 사용하여 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제하고, 데이터베이스 파티션 그룹에 데이터를 재분배하며 인스턴스에서 파티션을 삭제해야 하는 태스크에 대해 안내합니다.

이 태스크에 대한 정보

주: 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제하는 경우 해당 데이터베이스 파티션이 즉시 삭제되지 않습니다. 대신에, 데이터베이스 파티션 그룹에서 데이터를 재분배할 경우 삭제되는 데이터베이스 파티션에서 데이터를 이동시킬 수 있도록 삭제하려는 데이터베이스 파티션이 플래그됩니다.

데이터베이스 파티션 그룹에서 데이터를 재분배하기 이전 및 이후에 인스턴스에 있는 모든 데이터베이스를 백업하는 것이 좋습니다. 데이터베이스를 백업하지 않으면 데이터베이스가 손상되는 경우 손상된 데이터베이스를 복구할 수 없을 수도 있습니다.

프로시저

파티션 삭제 런치패드를 사용하여 파티션을 삭제하려면 다음과 같이 수행하십시오.

1. 선택사항: 백업 마법사를 사용하여 데이터를 백업하십시오.
2. 파티션 삭제 런치패드를 여십시오.
   a. 파티션 창을 여십시오. 제어 센터에서 파티션을 보려는 인스턴스를 찾을 때까지 오브젝트 트리를 펼치십시오. 원하는 인스턴스에서 마우스 오른쪽 단추를 누르고 **데이터베이스 파티션 서버 열기**를 선택하십시오. 선택된 인스턴스의 파티션 창이 열립니다.
   b. 삭제하려는 파티션을 선택하십시오.
   c. 선택된 파티션을 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 **삭제**를 누르십시오. 파티션 삭제 런치패드가 열립니다.
3. 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제하십시오.
   a. 데이터베이스 파티션 그룹에서 삭제하려는 데이터베이스 파티션을 확인하십시오.

      주:

      - 인스턴스에서 파티션을 삭제하기 전에 먼저 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제해야 합니다.

- 이 조작을 수행해도 데이터베이스 파티션이 즉시 삭제되지 않습니다. 대신에, 데이터베이스 파티션 그룹에서 데이터를 재분배할 경우 삭제되는 데이터베이스 파티션에서 데이터를 이동시킬 수 있도록 삭제하려는 데이터베이스 파티션이 플래그됩니다.

4. 데이터베이스 파티션 그룹에서 데이터를 재분배하십시오.
5. 인스턴스에서 파티션을 삭제하십시오.
   a. 인스턴스에서 파티션 삭제 확인 창을 여십시오.
      - 위에서 설명한 대로 파티션 창을 여십시오.
      - 삭제하려는 파티션을 선택하십시오.
      - 선택된 파티션을 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 **삭제**를 누르십시오. 파티션 삭제 런치패드가 열립니다.
      - **인스턴스에서 파티션 삭제** 누름 단추를 누르십시오. 인스턴스에서 파티션 삭제 확인 창이 열립니다.
   b. **삭제** 컬럼에서 선택된 인스턴스의 파티션을 삭제할지 검증하십시오.
   c. **확인**을 눌러 파티션을 삭제할 시기를 스케줄할 수 있는 창을 여십시오.
6. 선택사항: 백업 마법사를 사용하여 데이터를 백업하십시오.

시나리오: 데이터베이스 내에 있는 데이터 파티셔닝

이 시나리오에서는 새 데이터베이스 파티션을 데이터베이스에 추가하고 데이터베이스 파티션 간에 데이터를 재분배하는 방법을 표시합니다. REDISTRIBUTE DATABASE PARTITION GROUP 명령은 데이터베이스 파티션 그룹 내에서 서로 다른 테이블 세트의 데이터를 재분배하는 방법의 표시 파트로 설명됩니다.

시나리오:

데이터베이스 DBPG1에는 (0, 1)로 지정된 두 개의 데이터베이스 파티션과 데이터베이스 파티션 그룹 정의(0, 1)가 있습니다.

다음 테이블 스페이스는 데이터베이스 파티션 그룹 DBPG_1에 정의되어 있습니다.

- 테이블 스페이스 TS1 - 이 테이블 스페이스에는 두 개의 테이블 T1 및 T2가 있습니다.
- 테이블 스페이스 TS2 - 이 테이블 스페이스에는 세 개의 테이블 T3, T4 및 T5가 정의되어 있습니다.

DBPG1의 데이터베이스 파티션 간 데이터 재분배:

세 개의 새 데이터베이스 파티션을 데이터베이스에 추가하려면 다음 명령을 발행하십시오.

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME3>
 PORT <PORT3>;
```

```
START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME4>
 PORT <PORT4>;

START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME5>
 PORT <PORT5>;

STOP DBM;

START DBM;
```

다음 재분배 명령은 DBPG_1 정의를 (0, 1)에서 (0, 1, 3, 4, 5)로 변경하고 데이터도 재분배합니다.

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM ADD DBPARTITIONNUM (3 TO 5) STOP AT 2006-03-10-07.00.00.000000;
```

테이블 T1, T2 및 T3에 대해 명령이 성공적으로 실행된 후 STOP AT 옵션 스펙으로 인해 중지되었다고 가정합시다.

데이터베이스 파티션 그룹에 대한 데이터 재분배를 중단하고 테이블 T1, T2 및 T3에 대한 변경사항을 되돌리려면 다음 명령을 발행하십시오.

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD
 RECOVERABLE ABORT;
```

데이터 재분배 중에 오류나 일시중단이 발생했거나 재분배 연산을 계속하지 않으려는 경우 데이터 재분배를 중단할 수 있습니다. 이 시나리오의 경우, 이 명령이 성공적으로 실행되었고 테이블 T1 및 T2가 원래 상태로 되돌려졌다고 가정하십시오.

5000 4K 페이지를 DATA BUFFER로 사용하여 T5 및 T4만 재분배하려면 다음을 수행하십시오.

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM ADD DBPARTITIONNUM (3 TO 5) TABLE (T5, T4) ONLY  DATA BUFFER 5000;
```

명령이 성공적으로 실행된 경우, 테이블 T4 및 T5의 데이터가 성공적으로 재분배되었습니다.

지정된 순서로 테이블 T1, T2 및 T3의 데이터 재분배를 완료하기 위해서는 다음을 발생하십시오.

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 CONTINUE TABLE (T1) FIRST;
```

TABLE (T1) FIRST를 지정하면 데이터베이스 관리 프로그램에서 테이블 T1을 맨 먼저 처리하도록 강제 실행되어 해당 테이블이 기타 테이블 이전에 온라인(읽기 전용) 상태로 리턴할 수 있습니다. 기타 모든 테이블은 데이터베이스 관리 프로그램에서 판별한 순서대로 처리됩니다.

주:

- ADD DBPARTITIONNUM 옵션 및 DROP DBPARTITIONNUM 옵션은 지정할 필요가 없습니다. 그 대신에 REDISTRIBUTE DATABASE PARTITION GROUP 명령이 실행되기 전에 데이터베이스 파티션을 추가

또는 삭제하기 위해 ALTER DATABASE PARTITION GROUP문이 사용될 수 있습니다. 이 경우 REDISTRIBUTE DATABASE PARTITION GROUP 명령은 파티션을 추가 또는 삭제하는 것이 아니라 단지 지정된 옵션에 따라 데이터를 재분배합니다.

- 사용자는 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 실행하기 전에 해당 데이터베이스의 오프라인 데이터베이스 백업을 받는 것이 권장됩니다. 이 조치는 위 예제에 표시되지 않았습니다.
- REDISTRIBUTE DATABASE PARTITION GROUP 명령은 롤 포워드 복구 가능하지 않습니다. 이 문제에 대한 충분한 논의는 『REDISTRIBUTE DATABASE PARTITION GROUP 명령』을 참조하십시오.
- REDISTRIBUTE DATABASE PARTITION GROUP 명령이 완료된 후 이 명령이 액세스한 모든 테이블 스페이스가 BACKUP PENDING 상태로 남게 됩니다. 이러한 테이블 스페이스는 들어 있는 테이블이 쓰기 활동용으로 액세스 가능해지기 전에 백업되어야 합니다.

위 단계에서는 재분배 명령의 변형을 사용하여 데이터베이스 파티션 간에 데이터를 재분배할 수 있는 방법에 대해 설명합니다.

파티션된 데이터베이스 환경에서 명령 실행

파티션된 데이터베이스 환경에서 인스턴스의 컴퓨터 또는 데이터베이스 파티션 서버(노드)에서 실행되도록 명령을 발행할 수 있습니다. 이 경우, rah 명령 또는 db2_all 명령을 사용하십시오. rah 명령을 사용하면 인스턴스의 컴퓨터에서 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 환경에만 적용됩니다.

Windows에서 rah 명령 또는 db2_all 명령을 실행하려면 관리자 그룹의 구성원이 되는 사용자 어카운트로 로그온하십시오.

Linux 및 UNIX 플랫폼에서 로그인 쉘이 Korn 쉘 또는 다른 쉘이 될 수 있으나, 서로 다른 쉘은 특수 문자를 포함하는 명령을 처리하는 방법에서 차이를 나타냅니다.

또한 Linux 및 UNIX 플랫폼에서 rah는 **DB2RSHCMD** 레지스트리 변수로 지정된 리모트 쉘 프로그램을 사용합니다. 두 개의 리모트 쉘 프로그램, ssh(추가 보안이 필요한 경우) 또는 rsh(또는 HP-UX의 경우 remsh) 중에서 선택할 수 있습니다. **DB2RSHCMD**가 설정되지 않으면 rsh(또는 HP-UX의 경우 remsh)가 사용됩니다. ssh 리모트 쉘 프로그램은 UNIX 운영 체제 환경에서 텍스트 상태의 암호 전송을 막는데 사용됩니다.

DB2 DPF 인스턴스에 사용할 ssh의 단일 버전을 구성하는 방법에 대한 자세한 설명은 다음 항목을 참조하십시오. ″OpenSSH를 사용하려면 DB2 Universal Database for UNIX를 구성하십시오.″

명령 범위를 판별하려면 명령이 단일 데이터베이스 파티션 서버에서 실행되는지 또는 이들 전체에서 실행되는지를 보여주는 *명령어 참조서*를 참조하십시오. 명령이 하나의 데이터베이스 파티션 서버에서 실행되고 있을 때 명령이 이들 전체에서 실행되게 하려면 db2_all을 사용하십시오. 예외는 컴퓨터의 모든 논리 노드(데이터베이스 파티션 서버)에서 실행되는 db2trc 명령입니다. 모든 컴퓨터의 모든 논리 노드에서 db2trc를 실행하려면 rah를 사용하십시오.

rah 및 db2_all 명령 개요

하나의 데이터베이스 파티션 서버에서 또 다른 서버 이후에 연속적으로 명령을 실행할 수 있고 혹은 명령을 병렬로 실행할 수 있습니다. Linux 및 UNIX 플랫폼에서 명령을 병렬로 실행할 경우 출력을 버퍼로 보내 표시를 위해 수집하도록(디폴트 동작) 선택할 수 있거나 출력을 명령이 발행된 컴퓨터에 표시할 수 있습니다.

Windows에서 명령을 병렬로 실행하면 출력은 명령이 발행된 컴퓨터에 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. Linux 및 UNIX 플랫폼에서 세미콜론(;)을 사용하여 다중 명령을 구분할 수 있습니다. Windows에서는 앰퍼샌드(&)를 사용하여 다중 명령을 구분합니다. 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 ";DB2 GET DB CFG FOR sample USING LOGFILSIZ 100"
```

rah 및 db2_all 명령 지정

명령행에서 rah 명령을 매개변수로 지정하거나 매개변수를 지정하지 않을 경우 프롬프트에 응답하여 지정할 수 있습니다.

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: Linux 및 UNIX 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: Linux 및 UNIX 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. Linux 및 UNIX 플랫폼에서 명령 쉘이 작은따옴표가 붙은 문자열 내에 작은따옴표를 입력하는 방법을 제공하지 않는 한, 명령에 작은따옴표(')를 포함시킬 수 없습니다.
2. Windows에서 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 양식의 스크립트를 실행할 수 있습니다.

```
shell_script </dev/null &
```

(제공되는 입력이 없는 경우)

비슷한 방법으로, 백그라운드에서 db2_all을 실행하려면 </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.

리모트 명령의 출력이 필요 없는 경우, 이 메소드 대신 다음과 같이 db2_all 접두부에 『daemonize』 옵션을 사용할 수도 있습니다.

```
db2_all ";daemonize_this_command" &
```

병렬 명령 실행(Linux, UNIX)

디폴트로, 명령은 각 컴퓨터에서 순차적으로 실행되지만 특정 접두부 시퀀스를 가지도록 명령에 접두부를 추가함으로써, 백그라운드 rshell을 사용하여 명령이 병렬로 실행되도록 지정할 수 있습니다. rshell이 백그라운드에서 실행되고 각 명령이 리모트 컴퓨터의 버퍼 파일에 출력을 추가하는 경우,

주: 이 절의 정보는 Linux and UNIX 플랫폼에만 적용됩니다.

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/rahout이지만, 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 컴퓨터에서 명령을 동시에 실행하기 전에 다음을 확인하십시오.

- 디렉토리 /tmp/$USER이 각 컴퓨터의 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 컴퓨터 ID가 rah를 실행하는 ID에 대한 .rhosts 파일에 항목을 갖는지와 rah를 실행하는 ID가 리모트 명령을 실행하는 각 컴퓨터 ID에 대한 .rhosts 파일에 항목을 갖는지 확인하십시오.

트리 논리를 사용하도록 rah 명령 확장 (AIX 및 Solaris)

성능 향상을 위해 대형 시스템에서 rah에 use 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 디폴트 값으로 15가 됩니다.

물리 노드당 다중 논리 노드 시스템의 경우, db2_all은 개별 물리 노드에 재귀 호출을 보내며, 이 노드는 같은 물리 노드에 있는 다른 논리 노드에 rsh하여 물리 노드 간 전송량도 줄어듭니다. (이는 db2_all에만 적용되며 rah에는 적용되지 않는데, 그 이유는 rah는 항상 개별 물리 노드에만 보내기 때문입니다.)

rah 및 db2_all 명령

이 주제는 rah 및 db2_all 명령에 대한 설명을 포함합니다.

명령 **설명**

rah 모든 컴퓨터에서 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 Software Support의 지시가 있는 경우 지속된 트랩을 복구하도록 지시된 경우를 제외하고, 이 명령을 발행하지 *마십시오*.

db2_call_stack

Linux 및 UNIX 플랫폼의 경우 모든 데이터베이스 파티션 서버에서 실행 중인 프로세스가 syslog에 call traceback을 기록합니다.

Linux 및 UNIX 플랫폼에서 이러한 명령은 다음과 같은 내재된 특정 설정을 통해 rah를 실행합니다.

- 모든 컴퓨터에서 병렬 실행
- 각각 `/tmp/$USER/db2_kill`과 `/tmp/$USER/db2_call_stack`에서의 버퍼 명령 출력

db2_call_stack 명령은 Windows에서 사용할 수 *없습니다*. 대신 db2pd -stack 명령을 사용하십시오.

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.

- Linux 및 UNIX 플랫폼의 경우:

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- Windows의 경우:

```
rah "||db2 get db cfg for sample"
```

접두부 시퀀스는 다음과 같습니다.

순서	목적
|	백그라운드에서 시퀀스대로 명령을 실행합니다.
|&	일부 프로세스가 여전히 실행 중이더라도 백그라운드에서 명령을 시퀀스대로 실행하고, 모든 리모트 명령이 완료된 이후 명령을 종료합니다. 하위 프로세스(Linux and UNIX 플랫폼) 또는 백그라운드 프로세스(Windows)가 여전히 실행 중인 경우에는 지연될 수도 있습니다. 이 경우, 명령은 별도의 백그라운드 프로세스를 시작하여 명령 종료 이후에 생성된 리모트 출력을 검색한 다음, 이를 원래 컴퓨터에 다시 작성합니다. **주:** Linux 및 UNIX 플랫폼에서 &를 지정하면 많은 rsh 명령이 요구되므로 성능이 저하됩니다.
||	백그라운드에서 명령을 병렬로 실행합니다.
||&	백그라운드에서 명령을 병렬로 실행하고 위의 |& 경우 대한 설명대로 모든 리모트 명령이 완료된 후에 명령을 종료합니다. **주:** Linux 및 UNIX 플랫폼에서 &를 지정하면 많은 rsh 명령이 요구되므로 성능이 저하됩니다.
;	위의 ||&와 같습니다. 이것은 사용할 수 있는 다른 짧은 양식입니다. **주:** Linux 및 UNIX 플랫폼에서 ;를 지정하면 많은 rsh 명령이 요구되므로 ||에 비해 성능이 저하됩니다.
]	명령 실행 전에 사용자의 프로파일이 dot 실행을 사전에 보류합니다. **주:** Linux 및 UNIX 플랫폼에서만 사용 가능합니다.

} 명령을 실행하기 전에 $RAHENV에 이름 지정된 파일(.kshrc)의 dot 실행을 사전에 보류합니다.

주: Linux 및 UNIX 플랫폼에서만 사용 가능합니다.

]} 명령을 실행하기 전에 $RAHENV(.kshrc)에 이름 지정된 파일 실행 이전에 사용자 프로파일의 dot 실행을 보류합니다.

주: Linux 및 UNIX 플랫폼에서만 사용 가능합니다.

) 사용자의 프로파일 및 $RAHENV에 이름 지정된 파일이 실행되는 것을 막습니다.

주: Linux 및 UNIX 플랫폼에서만 사용 가능합니다.

' 명령 호출을 컴퓨터로 반향합니다.

< 이 컴퓨터를 제외한 모든 컴퓨터로 보냅니다.

<<-nnn<

데이터베이스 파티션 서버 *nnn*을 제외한 전체(노드 번호 *nnn*을 제외한 db2nodes.cfg의 모든 데이터베이스 파티션 서버, 이 테이블에서 최종 접두부 다음 첫 번째 단락 참조)로 보냅니다.

<<+nnn<

데이터베이스 파티션 서버 *nnn*으로만 보냅니다. 데이터베이스 파티션 번호가 *nnn* 인 db2nodes.cfg의 데이터베이스 파티션 서버는 이 테이블의 마지막 접두부 시퀀스 뒤에 오는 첫 번째 단락을 참조하십시오.

(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 좀 더 빨리(리모트 명령이 시작되자마자) 완료할 수 있습니다. rah 명령행에 이 접두부 시퀀스를 지정한 경우 명령을 작은따옴표로 묶거나, 큰따옴표로 묶고 앞에 ₩를 사용하십시오. 예를 들어, 다음과 같습니다.

```
rah '; mydaemon'
```

또는

```
rah ";₩ mydaemon"
```

rah 명령은 백그라운드 프로세스로 실행할 경우, 출력이 리턴되기를 기다리지 않습니다.

> >를 컴퓨터 이름으로 대체하십시오.

″ ()를 컴퓨터 인덱스로 대체하고, ##를 데이터베이스 파티션 번호로 대체합니다.

주:

1. 컴퓨터 인덱스는 데이터베이스 시스템의 컴퓨터와 연관된 번호입니다. 다중 논리 파티션을 실행하고 있지 않은 경우, 컴퓨터에 대한 컴퓨터 인덱스는 노드 구성 파일의 해당 컴퓨터에 대한 데이터베이스 파티션 번호에 해당합니다. 다중 논리 파티션 데이터베이스 환경에서 컴퓨터에 대한 컴퓨터 인덱스를 얻으려면, 다중 논리 파티션이 실행되는 컴퓨터에 대해 중복된 항목을 계산하지 않도록 하십시오. 예를 들어, MACH1이 두 개의 논리 파티션에서 실행되고 MACH2도 두 개의 논리 파티션에서 실행되는 경우, 노드 구성 파일에서 MACH3에 대한 데이터베이스 파티션 번호는 5가 됩니다. 그러나 MACH3에 대한 컴퓨터 인덱스는 3이 됩니다.

   Windows에서는 노드 구성 파일을 편집하지 마십시오. 컴퓨터 인덱스를 얻으려면, db2nlist 명령을 사용하십시오.
2. "를 지정하면, 컴퓨터 목록에서 중복된 사항이 제거되지 않습니다.

`<<-nnn<` 및 `<<+nnn<` 접두부 시퀀스를 사용할 경우 *nnn*은 `db2nodes.cfg` 파일의 *nodenum* v값과 일치해야 하는 1, 2 또는 3자리 데이터베이스 파티션 번호입니다.

주: 접두부 시퀀스는 명령의 일부로 인식해야 합니다. 명령의 일부로 접두부 시퀀스를 지정하는 경우, 접두부 시퀀스를 포함하여 전체 명령을 큰따옴표로 묶어야 합니다.

rah 명령 제어

이 주제는 rah 명령을 제어할 환경 변수를 나열합니다.

표 13. rah 명령으로 제어하는 환경 변수

이름	의미	디폴트값
$RAHBUFDIR **주:** Linux 및 UNIX 플랫폼에서만 사용 가능합니다.	버퍼 디렉토리	/tmp/$USER
$RAHBUFNAME **주:** Linux 및 UNIX 플랫폼에서만 사용 가능합니다.	버퍼 파일 이름	rahout
$RAHOSTFILE(Linux 및 UNIX 플랫폼); RAHOSTFILE (Windows)	호스트 목록이 들어 있는 파일	`db2nodes.cfg`
$RAHOSTLIST(Linux 및 UNIX 플랫폼); RAHOSTLIST (Windows)	문자열로 된 호스트 목록	$RAHOSTFILE에서 얻음

표 13. rah 명령으로 제어하는 환경 변수 (계속)

이름	의미	디폴트값
$RAHCHECKBUF **주:** Linux 및 UNIX 플랫폼에서만 사용 가능합니다.	″no″로 설정되면, 검사를 생략합니다.	설정 안됨
$RAHSLEEPTIME (Linux 및 UNIX 플랫폼); RAHSLEEPTIME (Windows)	스크립트가 병렬로 실행되는 명령으로부터 초기 출력을 기다리는 시간(초)	db2_kill의 경우는 86400초, 기타 모든 경우는 200초
$RAHWAITTIME (Linux 및 UNIX platforms); RAHWAITTIME (Windows)	Windows에서 리모트 작업이 아직 실행 중인 연속되는 검사 간의 간격(초) Linux 및 UNIX 플랫폼에서 리모트 작업이 아직 실행 중임과 `rah: waiting for pid> ...` 메시지가 나타나는 것을 확인하는 연속되는 검사간의 간격(초). 모든 플랫폼에서 양의 정수를 지정하십시오. 메시지 출력을 제어하기 위해 사용되는 0으로 시작하는 접두부 값의 경우, RAHWAITTIME=045를 익스포트하십시오. rah는 작업 완료 상태를 검출하기 위해 이 검사를 따르지 않으므로 낮은 값을 지정할 필요는 없습니다.	45초
$RAHENV **주:** Linux 및 UNIX 플랫폼에서만 사용 가능합니다.	$RAHDOTFILES=E, K, PE 또는 B이면 실행 가능한 파일 이름을 지정하십시오.	$ENV
$RAHUSER(Linux 및 UNIX platforms); RAHUSER(Windows)	Linux 및 UNIX 플랫폼에서 리모트 명령이 실행될 때의 사용자 ID Windows에서 DB2 리모트 명령 서비스와 연관된 로그온 어카운트	$USER

주: Linux 및 UNIX 플랫폼에서 리모트 쉘이 설정한 값(있는 경우)이 아닌 rah가 실행되고 있는 $RAHENV의 값이 사용됩니다.

rah(Linux 및 UNIX)로 실행되는 . 파일 지정

이 주제에서는 접두부 시퀀스가 지정되지 않은 경우에 실행되는 . 파일을 나열합니다.

주: 이 절의 정보는 Linux and UNIX 플랫폼에만 적용됩니다.

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, rah에 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(rsh에 의해 호출될 때), .cshrc가 stdout에 기록되지 못하게 해야 합니다. 다음과 같이 stdout에 작성하는 행을 넣어 확인할 수 있습니다.

```
if { tty -s } then echo "executed .cshrc";
endif
```

rah를 사용하여 문제점 판별 (Linux, UNIX)

이 주제에서는 rah를 실행할 때 발생할 수 있는 문제점을 처리하는 방법입니다.

주: 이 절의 정보는 Linux and UNIX 플랫폼에만 적용됩니다.

1. rah 정지(또는 오랜 시간 경과)

   문제점의 원인은 다음과 같습니다.

   - rah는 출력을 버퍼링할 필요가 있음을 판별했는데 사용자가 RAHCHECKBUF=no를 익스포트하지 않았습니다. 따라서 명령을 실행하기 전에 rah는 모든 컴퓨터에 명령을 보내어 버퍼 디렉토리가 있는지 검사하고, 없는 경우 이를 작성합니다.
   - 명령을 보내고 있는 하나 이상의 컴퓨터가 응답하지 않습니다. rsh 명령은 결과적으로 시간종료되지만, 시간종료 간격은 보통 60초로 매우 깁니다.

2. 다음과 같은 메시지가 수신되었습니다.
   - 올바르지 않은 로그인
   - 사용 권한 거부

   컴퓨터 중 하나가 .hosts 파일에 적절히 정의된 rah를 실행하는 ID를 가지고 있지 않거나, rah를 실행하는 ID가 .rhosts 파일에 적절히 정의된 컴퓨터 중 하나를 가지고 있지 않습니다. DB2RSHCMD 레지스트리 변수가 ssh를 사용하도록 구성된 경우에는 각 컴퓨터의 ssh 클라이언트 및 서버가 제대로 구성되지 않을 수 있습니다.

주: 데이터베이스 파티션 간 명확한 텍스트의 암호 전송에 관한 보안의 필요성이 증가할 수 있습니다. 이것은 사용하는 리모트 쉘 프로그램에 따라 다릅니다. rah는 DB2RSHCMD 레지스트리 변수에 의해 지정된 리모트 쉘 프로그램을 사용합니다. 두 개의 리모트 쉘 프로그램, ssh(추가 보안이 필요한 경우) 또는 rsh(또는 HP-UX의 경우 remsh) 중에서 선택할 수 있습니다. 이 레지스트리 변수가 설정되지 않으면 rsh(또는 HP-UX의 경우 remsh)가 사용됩니다.

3. 백그라운드 리모트 쉘을 사용하여 명령을 병렬로 실행하는 경우 명령이 컴퓨터에서 예상된 경과 시간 내에 실행되어 완료되어도, rah가 이를 감지하고 쉘 프롬프트로 올리는데 많은 시간이 필요합니다.

   rah를 실행하는 ID가 `.rhosts` 파일에 적절히 정의된 컴퓨터 중 하나를 가지고 있지 않거나, DB2RSHCMD 레지스트리 변수가 ssh를 사용하도록 구성된 경우에는 각 컴퓨터의 ssh 클라이언트 및 서버가 제대로 구성되지 않을 수 있습니다.

4. 쉘 명령행으로부터 실행할 때 rah가 제대로 실행되더라도, 다음과 같이 rsh를 사용하여 리모트로 rah를 실행하면

   ```
   rsh somewher -l $USER db2_kill
   ```

   rah이 완료됩니다.

   이것은 정상적입니다. rah는 백그라운드 모니터링 프로세스를 시작하며 이 프로세스를 빠져 나간 후에도 계속 실행됩니다. 수행하고 있는 명령과 연관된 모든 프로세스를 자체 종료할 때까지 프로세스는 정상적으로 지속됩니다. db2_kill의 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 $RAHUSER에서 발행되어 있지 않을 때, rah로부터의 출력이 제대로 표시되지 않거나 rah가 $RAHBUFNAME이 존재하지 않음을 제대로 보고하지 않습니다.

   이것은 rah의 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. rah는 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:

   - rah가 아닌 db2_all을 사용하십시오.
   - RAHOSTFILE을 익스포트하거나 /sqllib/db2nodes.cfg 파일을 디폴트 파일로 지정하여 RAHOSTFILE이 사용되는지 확인하십시오. 이러한 전제조건없이, rah는 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 리모트 쉘 명령이 필요하므로 성능이 저하됩니다.

rah 프로세스 모니터링(Linux, UNIX)

아직 실행 중인 리모트 명령이 있거나 버퍼된 출력이 아직 누적되고 있는 경우, rah에 의해 시작된 프로세스는 실행되지 않은 명령을 표시하는 터미널에 메시지를 쓰고 버퍼된 출력을 검색하는 활동을 모니터합니다.

주: 이 절의 정보는 Linux and UNIX 플랫폼에만 적용됩니다.

정보 메시지는 환경 변수 RAHWAITTIME으로 제어되어 일정 간격으로 작성됩니다. 이를 지정하는 방법에 대한 세부사항은 도움말 정보를 참조하십시오. 모든 정보 메시지는 RAHWAITTIME=0을 익스포트하면 표시되지 않습니다.

1차 모니터링 프로세스는(ps 명령에 의해 표시된 대로) 명령 이름이 rahwaitfor이 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. rah의 단일 실행 중에 다른 시간에 실

행되는 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 rah를 사용할 수 있으나. 다음 특수 문자를 포함하는 명령을 입력하는 방법에는 약간 다른 규칙이 적용될 수 있습니다.

```
"  unsubstituted $ '
```

자세한 정보는 rah "?". 또한 Linux 및 UNIX 환경에서 리모트 명령을 실행하는 ID에서의 로그인 쉘이 Korn 쉘이 아니면 rah를 실행하는 ID에서의 로그인 쉘도 Korn 쉘이면 안 됩니다 (rah는 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.

Windows에서 rah에 대한 디폴트 환경 프로파일 설정

rah 명령에 대한 디폴트 환경 프로파일을 설정하려면, 인스턴스 디렉토리에 작성되는 db2rah.env 파일을 사용하십시오.

주: 이 절의 정보는 Windows에만 적용됩니다.

이 파일은 다음 형식을 가져야 합니다.

```
   ; This is a comment line
DB2INSTANCE=instancename
DB2DBDFT=database
; End of file
```

rah에 대한 환경을 초기화하는데 필요한 모든 환경 변수를 지정할 수 있습니다.

제 11 장 테이블 및 기타 관련 테이블 오브젝트 작성

파티션된 데이터베이스 환경의 테이블

파티션된 데이터베이스 환경에서 여러 데이터베이스 파티션에 걸쳐 테이블을 작성하면 성능상 이점이 있습니다. 여러 데이터베이스 파티션이 데이터 검색과 연관된 작업을 분담할 수 있습니다.

실제로 분할되거나 분산된 테이블을 작성하기 전에, 다음과 같은 사항을 고려해야 합니다.

- 테이블 스페이스는 둘 이상의 데이터베이스 파티션을 확장시킬 수 있습니다. 테이블이 놓이는 데이터베이스 파티션의 수는 데이터베이스 파티션 그룹의 데이터베이스 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.

몇몇 데이터베이스 파티션의 일부가 될 테이블의 작성은 해당 테이블을 작성할 때 구체화됩니다. 파티션된 데이터베이스 환경에서 테이블을 작성할 때에는 *분산 키*라는 추가 옵션이 있습니다. 분산 키는 테이블의 정의 파트인 키입니다. 이 키는 데이터의 각 행이 저장된 데이터베이스 파티션을 판별합니다.

분산 키를 확실히 지정하지 않은 경우, 다음 디폴트값이 사용됩니다. *디폴트 분산 키가 적합한지 확인하십시오.*

- 기본 키가 CREATE TABLE문에서 지정되지 않은 경우, 기본 키의 첫 번째 컬럼은 분산 키로 사용됩니다.
- 다중 파티션 데이터베이스 파티션 그룹에 대해 1차 키가 없는 경우 Long 필드가 아닌 첫 번째 컬럼이 사용됩니다.
- 어떠한 컬럼도 디폴트 분산 키의 요구사항을 충족시킬 수 없는 경우, 이 옵션 없이 테이블이 작성됩니다(단일 파티션 데이터베이스 파티션 그룹의 경우에만 허용됨).

일단 분산 키를 선택하면 나중에 변경이 불가능하므로 주의해서 적절한 분산 키를 선택해야 합니다. 더욱이, 고유 인덱스(고유 키 또는 기본 키)는 분산 키의 수퍼 세트로 정의되어야 합니다. 다시 말해서 분산 키를 정의할 경우, 고유 키와 기본 키는 분산 키(컬럼은 더 많을 수 있음)와 동일한 모든 컬럼을 포함해야 합니다.

테이블의 데이터베이스 파티션 *크기*는 테이블 스페이스 유형 및 사용된 페이지 *크기*와 연관된 특정 한계의 소량 및 사용 가능한 디스크 스페이스의 양입니다. 예를 들어, 4KB 페이지 *크기*의 대형 DMS 테이블 스페이스를 가정하면 테이블 *크기*는 8TB에 데이터

베이스 파티션 수를 곱한 소량 및 사용 가능한 디스크 스페이스의 양입니다. 데이터베이스 관리 프로그램 페이지 크기 한계의 전체 목록에 대한 관련 링크를 참조하십시오.

명령행을 사용하여 파티션된 데이터베이스 환경에 테이블을 작성하려면 다음을 입력하십시오.

```
CREATE TABLE name>
        (<column_name>  <data_type>  <null_attribute>)
       IN <tagle_space_name>
         INDEX IN <index_space_name>
         LONG  IN <long_space_name>
  DISTRIBUTE BY HASH (<column_name>)
```

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
                     DISTRIBUTE BY HASH (MIX_INT)
```

앞의 예에서 테이블 스페이스는 MIXTS12이며 분산 키는 MIX_INT입니다. 분산 키가 확실히 지정되어 있지 않으면, 테이블 스페이스는 MIX_CNTL입니다. (기본 키가 지정되어 있지 않고 분산 키가 정의되어 있지 않을 경우 분산 키는 목록에서 첫 번째의 길지 않은 컬럼이 됩니다).

테이블 행 및 해당 행에 대한 모든 정보는 항상 동일한 데이터베이스 파티션에 상주합니다.

파티션된 테이블의 대형 오브젝트(LOB) 동작

파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 데이터 파티션 또는 범위라고 하는 여러 스토리지 오브젝트로 나누어지는 데이터 구성 스킴을 사용합니다. 제공된 테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 제공된 스펙을 기본으로 복수의 스토리지 오브젝트로 파티션됩니다. 이러한 스토리지 오브젝트는 서로 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘의 조합에 있을 수 있습니다.

파티션된 테이블에 대한 대형 오브젝트는 디폴트로 이에 대응하는 데이터 오브젝트와 동일한 테이블 스페이스에 저장됩니다. 이는 단 하나의 테이블 스페이스를 사용하거나

다중 테이블 스페이스를 사용하는 파티션된 테이블에 적용됩니다. 파티션된 테이블의 데이터가 다중 테이블 스페이스에 저장되어 있는 경우, 대형 오브젝트(LOB) 데이터도 또한 다중 테이블 스페이스에 저장됩니다.

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

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

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

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

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

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

파티션된 테이블 작성

파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 복수의 스토리지 오브젝트, 호출된 데이터 파티션 또는 범위에 나뉘어져 있는 데이터 조직 스킴을 사용합니다. 제공된 테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 제공된 스펙을 기본으로 복수의 스토리지 오브젝트로 파티션됩니다. 이러한 스토리지 오브젝트는 서로 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘의 조합에 있을 수 있습니다.

테이블을 작성하려면, 명령문의 권한 부여 ID가 보유하는 특권이 최소한 다음과 같은 권한 또는 특권 중 하나를 포함해야 합니다.

- 다음 중 하나를 비롯하여 테이블에 사용되는 모든 테이블 스페이스에 대한 USE 특권 및 데이터베이스에 대한 CREATETAB 권한
  - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- DBADM 권한

CREATE TABLE문을 사용하여 파티션된 테이블을 작성할 수 있습니다.

명령행에서 파티션된 테이블을 작성하려면 CREATE TABLE문을 실행하십시오.

```
CREATE TABLE <NAME> (<column_name>  <data_type>  <null_attribute>) IN
 <table space list> PARTITION BY RANGE (<column expression>)
 STARTING FROM <constant> ENDING <constant> EVERY <constant>
```

예를 들어, 다음 명령문은 ≥ 1 및 ≤ 20인 행이 PART0(첫 번째 데이터 파티션)에 있고 21 ≤ ≤ 40인 행이 PART1(두 번째 데이터 파티션)에 있으며, 최대 81 ≤ ≤ 100이 PART4(마지막 데이터 파티션)에 있는 테이블을 작성합니다.

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE (a) (STARTING FROM (1)
 ENDING AT (100) EVERY (20))
```

파티션된 테이블에 대한 범위 정의

파티션된 테이블을 작성할 때 각 데이터 파티션의 범위를 지정할 수 있습니다. 파티션된 테이블은 테이블의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 다중 데이터 파티션으로 나뉘어진 데이터 조직 스킴을 사용합니다.

제공된 테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 제공된 스펙을 기본으로 복수의 스토리지 오브젝트로 파티션됩니다. 범위는 PARTITION BY절의 STARTING FROM 및 ENDING AT 값에 의해 지정됩니다.

각 데이터 파티션의 범위를 완전히 정의하려면 충분한 바운더리를 지정해야 합니다. 다음은 파티션된 테이블에 범위를 정의할 때 고려해야 할 지침입니다.

- STARTING절은 데이터 파티션 범위의 하한을 지정합니다. 이 절은 최저 데이터 파티션 범위에는 필수입니다(MINVALUE로 바운더리 정의 가능). 최저 데이터 파티션 범위는 하한이 지정된 데이터 파티션입니다.
- ENDING(또는 VALUES)절은 데이터 파티션 범위의 상한을 지정합니다. 이 절은 최고 데이터 파티션 범위에는 필수입니다(MAXVALUE로 바운더리 정의 가능). 최고 데이터 파티션 범위는 상한이 지정된 데이터 파티션입니다.
- 데이터 파티션에 ENDING절을 지정하지 않을 경우, 그 다음 큰 데이터 파티션에 STARTING절을 지정해야 합니다. 마찬가지로, STARTING절을 지정하지 않을 경우 이전 데이터 파티션에 ENDING절을 지정해야 합니다.
- MINVALUE는 사용되는 컬럼 유형에 사용 가능한 모든 값보다 작은 값을 지정합니다. MINVALUE와 INCLUSIVE 또는 EXCLUSIVE를 함께 지정할 수는 없습니다.
- MAXVALUE는 사용되는 컬럼 유형에 사용 가능한 모든 값보다 큰 값을 지정합니다. MAXVALUE와 INCLUSIVE 또는 EXCLUSIVE를 함께 지정할 수는 없습니다.
- INCLUSIVE는 지정된 값과 동일한 모든 값이 이 바운더리를 포함하는 데이터 파티션에 포함됨을 나타냅니다.
- EXCLUSIVE는 지정된 값과 동일한 모든 값이 이 바운더리를 포함하는 데이터 파티션에 포함되지 않음을 나타냅니다.
- CREATE TABLE문의 NULL절은 데이터 파티션 배치를 고려할 때 널(NULL) 값을 최대값으로 정렬할지 또는 최소값으로 정렬할지 여부를 지정합니다. 디폴트로 널(NULL) 값은 최대값으로 정렬됩니다. 테이블 파티션 키 컬럼의 널(NULL) 값은 양의 무한대로 처리되며 MAXVALUE로 종료되는 범위에 배치됩니다. 이러한 데이터 파티션을 정의하지 않으면 널(NULL) 값은 범위를 벗어난 것으로 간주됩니다. 테이블 파티션 키 컬럼에서 널(NULL) 값을 제외시키려면 NOT NULL 제한조건을 사용하십시오. LAST는 널(NULL) 값을 정렬된 값 목록의 맨 마지막에 표시하도록 지정합니다. FIRST는 널(NULL) 값을 정렬된 값 목록의 맨 앞에 표시하도록 지정합니다.
- Long 양식의 구문을 사용할 경우, 각 데이터 파티션마다 최소한 하나의 바운드가 지정되어 있어야 합니다.

추가 정보:테이블에 데이터 파티션을 정의하기 전에 테이블이 데이터 파티션을 이용하는 방법과 파티션 컬럼으로 선택한 컬럼에 영향을 미치는 요소를 이해하는 것이 중요합니다.

각 데이터 파티션에 지정된 범위는 자동 또는 수동으로 생성될 수 있습니다.

자동으로 생성

자동 생성은 많은 데이터 파티션을 신속하고 쉽게 작성하는 단순한 메소드입니다. 이 메소드는 날짜 또는 숫자를 기본으로 범위의 크기가 동일한 경우에 적당합니다.

예 1 및 2에서는 CREATE TABLE문을 사용하여 자동으로 각 데이터 파티션에 지정된 범위를 정의 및 생성하는 방법을 보여줍니다.

예 1

다음과 같이 범위가 정의된 테이블 작성 명령문을 발행하십시오.

```
CREATE TABLE lineitem (
    l_orderkey    DECIMAL(10,0) NOT NULL,
    l_quantity    DECIMAL(12,2),
    l_shipdate    DATE,
    l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
         PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

이 명령문에서는 다음과 같이 각 한 개의 키 값을 갖는 12개의 데이터 파티션이 생성됩니다. (l_shipdate) >= ('1/1/1992'), (l_shipdate) < ('3/1/1992'), (l_shipdate) < ('4/1/1992'), (l_shipdate) < ('5/1/1992'), ..., (l_shipdate) < ('12/1/1992'), (l_shipdate) < ('12/31/1992').

전체 시작 바운드('1/1/1992')는 포함(디폴트)되기 때문에 첫 번째 데이터 파티션의 시작값도 포함됩니다. 마찬가지로, 전체 종료 바운드('12/31/1992')가 포함(디폴트)되기 때문에 마지막 데이터 파티션의 종료 바운드도 포함됩니다. 나머지 STARTING 값은 포함되며 나머지 ENDING 값은 모두 제외됩니다. 각 데이터 파티션마다 n개의 키 값이 있습니다. 여기서 n은 EVERY절에 제공됩니다. 각 데이터 파티션의 종료 범위를 찾으려면 (start + every) 공식을 사용하십시오. EVERY 값이 START 및 END 범위로 균일하게 분할되지 않을 경우 마지막 데이터 파티션에 들어 있는 키 값의 수가 적을 수 있습니다.

예 2

다음과 같이 범위가 정의된 테이블 작성 명령문을 발행하십시오.

```
CREATE TABLE t(a INT, b INT)
 PARTITION BY RANGE(b) (STARTING FROM (1) EXCLUSIVE ENDING AT (1000) EVERY (100))
```

이 명령문으로 각 100개의 키 값(1 < b <= 101, 101 < b <= 201, ..., 901 < b <= 1000)을 갖는 10개의 데이터 파티션이 생성됩니다.

전체 시작 바운드(1)가 제외(exclusive)되기 때문에 첫 번째 데이터 파티션(b > 1 및 b <= 101)의 시작값이 제외됩니다. 마찬가지로 전체 종료 바운드(1000)가 포함되기 때문에 마지막 데이터 파티션(b > 901 b <= 1000)의 종료 바운드도 포함됩니다. 나머지 시작값은 모든 제외되며 나머지 종료값은 모두 포함됩니다. 각 데이터 파티션마다 n개의 키 값이 있습니다. 여기서 n은 EVERY절에 제공됩니다. 마지막으로, 전체 절의 시

작 및 종료 바운드가 모두 제외될 경우, 전체 시작 바운드(1)가 제외되므로 첫 번째 데이터 파티션의 시작값이 제외됩니다. 마찬가지로, 전체 종료 바운드(1000)가 제외되므로 마지막 데이터 파티션의 종료 바운드가 제외됩니다. 나머지 STARTING 값은 모두 제외되고 ENDING 값은 모두 포함됩니다. 각 데이터 파티션(마지막 데이터 파티션은 제외)마다 n개의 키 값을 보유합니다. n은 EVERY절에 제공됩니다.

수동으로 생성

수동 생성은 PARTITION BY절에 나열된 각 범위에 대한 새 데이터 파티션을 작성합니다. 이 형식의 구문은 범위를 정의할 때 보다 많은 유연성을 제공하므로 데이터 및 LOB 배치 옵션이 증가합니다. 예 3 및 4에 CREATE TABLE문을 사용하여 수동으로 데이터 파티션에 지정된 범위를 정의 및 생성하는 방법이 나와 있습니다.

예 3

이 명령문은 생성된 두 날짜 컬럼 모두에 대한 파티션을 생성합니다. CREATE TABLE 구문의 자동으로 생성되는 양식 사용 및 각 범위의 한 쪽만 지정했음을 참고하십시오. 다른 쪽 범위는 인접한 데이터 파티션의 값이고 INCLUSIVE 옵션을 사용한다고 가정합니다.

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)), item_id int NOT NULL,
cust_id int NOT NULL) PARTITION BY RANGE (inv_year, inv_month)
(PART Q1_02 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
PART Q2_02 ENDING (2002, 6) INCLUSIVE,
PART Q3_02 ENDING (2002, 9) INCLUSIVE,
PART Q4_02 ENDING (2002,12) INCLUSIVE,
PART CURRENT ENDING (MAXVALUE, MAXVALUE));
```

범위 내에 갭이 허용됩니다. CREATE TABLE 구문은 이전 데이터 파티션의 ENDING 값과 정렬되지 않는 값을 범위의 STARTING 값으로 지정할 수 있게 함으로써 갭을 지원합니다.

예 4

값 101 - 200의 갭을 가진 테이블을 작성합니다.

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE(a)
      (STARTING FROM (1) ENDING AT (100),
   STARTING FROM (201) ENDING AT (300))
```

데이터 파티션을 추가 또는 제거 가능한 ALTER TABLE문을 사용하면 범위에 갭이 생길 수 있습니다.

파티션된 테이블에 행을 삽입할 경우, 해당 키 값과 키가 속하는 범위를 기본으로 적절한 데이터 파티션에 자동으로 배치됩니다. 행이 테이블에 대해 정의된 범위 외부에 배치되면 삽입에 실패하고 다음 오류가 응용프로그램으로 리턴됩니다.

```
SQL0327N  The row cannot be inserted into table <tablename>
   because it is outside the bounds of the defined data partition ranges.
   SQLSTATE=22525
```

제한사항

- 테이블 레벨 제한사항:
  - 구문(EVERY절 포함)의 자동 생성된 양식을 사용하여 작성된 테이블은 테이블 파티션 키의 숫자 또는 날짜 시간 유형을 사용하도록 제한됩니다.
- 명령문 레벨 제한사항:
  - MINVALUE 및 MAXVALUE는 자동으로 생성된 구문 양식에서는 지원되지 않습니다.
  - 범위는 오름차순입니다.
  - 단 하나의 컬럼이 구문의 자동 생성된 양식에 지정될 수 있습니다.
  - EVERY절의 증분은 0보다 커야 합니다.
  - ENDING절은 STARTING절보다 크거나 같아야 합니다.

데이터 파티션의 데이터, 인덱스 및 Long 데이터 배치

고유 특성에 따라 파티션된 테이블을 작성하면 특정 테이블 스페이스에 테이블 및 연관된 테이블 오브젝트의 다양한 파트를 배치할 수 있습니다.

테이블 작성 시 전체 테이블 데이터 및 연관된 테이블 오브젝트가 배치되는 테이블 스페이스를 지정할 수 있습니다. 또는 특정 테이블 스페이스에 테이블의 인덱스, Long 또는 대형 데이터 또는 테이블 파티션을 배치할 수 있습니다. 모든 테이블 스페이스는 동일한 데이터베이스 파티션 그룹에 있어야 합니다.

CREATE TABLE문에는 특정 테이블 스페이스 내에 테이블 데이터 및 연관된 테이블 오브젝트를 배치하기 위해 이 기능을 설명하는 다음 절이 있습니다.

```
CREATE TABLE <table name> IN <table space name1>
   INDEX IN <table space name2>
   LONG IN <table space name3>
   PARTITIONED BY ... PARTITION <partition name> | boundary specification | IN <table space name4>
   INDEX IN <table space name5>
          LONG IN <table space name6>
```

파티션된 테이블의 각 파티션은 다른 테이블 스페이스에 배치될 수 있습니다.

또한 CREATE INDEX ... IN <table space name1>문을 사용하여 파티션된 테이블에 사용자가 작성한 파티션되지 않은 인덱스의 테이블 스페이스를 지정할 수 있습니다. 이는 CREATE TABLE ... INDEX IN <table space name2>문에 지정된 인덱스 테이블 스페이스와 다를 수 있습니다. CREATE INDEX문의 IN절은 파티션된 테이블에만 사용됩니다. INDEX IN절이 CREATE TABLE 또는 CREATE INDEX문에 지정되지 않으면, 인덱스는 테이블의 첫 번째 보이는 파티션 또는 접속된 파티션과 동일한 테이블 스페이스에 위치합니다.

시스템이 생성한 파티션되지 않은 인덱스(예: XML 컬럼 경로 인덱스)는 CREATE TABLE문의 INDEX IN 절에 지정된 테이블 스페이스에 배치됩니다.

XML 데이터가 있는 파티션된 테이블에서 XML 영역 인덱스는 항상 테이블 데이터와 같은 방법으로 파티션됩니다. 파티션된 인덱스의 테이블 스페이스는 파티션 레벨에 정의되어 있습니다.

XML 데이터는 테이블의 Long 데이터에 사용된 테이블 스페이스에 상주합니다. 파티션된 테이블의 XML 데이터 배치는 Long 데이터 배치 규칙을 따릅니다.

Long 데이터의 테이블 스페이스는 사용자가 명시적으로 지정하거나 데이터베이스 관리 프로그램에서 내재적으로 판별할 수 있습니다. 파티션된 테이블의 경우, 테이블 레벨 LONG IN절은 파티션 레벨 LONG IN 절과 함께 사용될 수 있습니다. 둘 다 지정되어 있는 경우, 파티션 레벨 LONG IN 절이 모든 테이블 레벨 LONG IN 절에 우선됩니다.

기존 테이블 및 뷰를 파티션된 테이블로 이주

데이터를 파티션되지 않은 테이블에서 빈 파티션된 테이블로 이주하려면 LOAD 명령을 사용하십시오.

다음 세 가지 방법으로 기존의 테이블 또는 뷰를 파티션된 테이블로 이주할 수 있습니다.

- 일반 테이블을 이주할 경우, 새로운 빈 파티션된 테이블을 작성한 후 LOAD from CURSOR를 사용하여 이전 테이블에서 파티션된 테이블로 중간 단계 없이 직접 데이터를 이동하십시오.
- 일반 테이블을 이주할 때 익스포트 유틸리티 또는 높은 성능의 언로드를 사용하여 소스 테이블을 언로드하고 새로운 빈 파티션된 테이블을 작성한 후 LOAD 명령으로 빈 파티션된 테이블에 데이터를 채우십시오.
- Union All 뷰를 이주할 경우, 단일 더미 데이터 파티션을 사용하여 파티션된 테이블을 작성한 후 모든 테이블을 접속시키십시오.

예 1: 일반 테이블 t1을 가지고 있다고 가정하십시오.

```
CREATE TABLE t1 (c1 int, c2 int);
```

새로운 빈 파티션된 테이블을 작성하십시오.

```
CREATE TABLE sales_dp (c1 int, c2 int)
   PARTITION BY RANGE (c1)
   (STARTING FROM 0 ENDING AT 10 EVERY 2);
```

테이블 t1에 데이터를 채우십시오.

```
INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
```

플랫 파일 데이터의 세 번째 사본이 작성되지 않게 하려면 LOAD 명령을 발행하여 SQL 쿼리에서 직접 새 파티션된 테이블로 데이터가 입력되게 하십시오.

```
   SELECT * FROM t1;
   DECLARE c1 CURSOR FOR SELECT * FROM t1;
   LOAD FROM c1 of CURSOR INSERT INTO sales_dp;SELECT * FROM sales_dp;

SELECT * FROM sales_dp;
```

이전 테이블을 삭제하려면 다음을 수행하십시오.

```
DROP TABLE t1;
```

UNION ALL 뷰 변환

UNION ALL 뷰의 파티션되지 않은 데이터를 파티션된 테이블로 변환할 수 있습니다. UNION ALL 뷰는 대형 테이블을 관리하고 분기 제거 성능을 향상시킴과 동시에 테이블 데이터를 쉽게 롤인 및 롤아웃하는 데 사용됩니다. 테이블 파티션에서 이 모든 작업이 가능하며 관리가 쉽습니다. ALTER TABLE ...ATTACH 조작을 사용할 경우, 기본 테이블의 데이터를 이동하지 않고 변환을 완료할 수 있습니다. 변환 후 파티션되지 않은 인덱스 및 종속 보기 또는 구체화된 쿼리 테이블(MQT)을 다시 작성해야 합니다.

권장되는 전략은 단일 더미 데이터 파티션을 가진 파티션된 테이블을 작성한 다음 UNION ALL 뷰의 모든 테이블을 접속하는 것입니다. 범위가 겹치는 문제점을 방지하기 위해서는 프로세스 초기에 더미 데이터 파티션을 삭제해야 합니다.

예 2

UNION의 첫 번째 테이블에 대한 테이블 작성 구문은 다음과 같습니다.

```
 CREATE TABLE sales_0198(
  sales_date DATE NOT NULL,
  prod_id INTEGER,
  city_id INTEGER,
  channel_id INTEGER,
  revenue DECIMAL(20,2),
   CONSTRAINT ck_date
   CHECK
   (sales_date BETWEEN '01-01-1998' AND '01-31-1998'));
```

union all 뷰의 보기 구문을 작성하십시오.

```
 CREATE VIEW all_sales AS
  (
  SELECT * FROM sales_0198
  WHERE sales_date BETWEEN '01-01-1998' AND '01-31-1998'
  UNION ALL
  SELECT * FROM sales_0298
  WHERE sales_date BETWEEN '02-01-1998' AND '02-28-1998'
  UNION ALL
  ...
```

```
  UNION ALL
  SELECT * FROM sales_1200
  WHERE sales_date BETWEEN '12-01-2000' AND '12-31-2000'
  );
```

단일 더미 파티션이 있는 파티션된 테이블을 작성하십시오. 접속될 첫 번째 데이터 파티션과 겹치지 않도록 범위를 선택해야 합니다.

```
 CREATE TABLE sales_dp (
  sales_date DATE NOT NULL,
  prod_id INTEGER,
  city_id INTEGER,
  channel_id INTEGER,
  revenue DECIMAL(20,2))
  PARTITION BY RANGE (sales_date)
  (PART dummy STARTING FROM '01-01-1900' ENDING AT '01-01-1900');
```

첫 번째 테이블을 접속하십시오.

```
ALTER TABLE sales_dp ATTACH PARTITION
STARTING FROM '01-01-1998' ENDING AT '01-31-1998'
FROM sales_0198;
```

더미 파티션을 삭제하십시오.

```
 ALTER TABLE sales_dp DETACH PARTITION dummy
 INTO dummy;
  DROP TABLE dummy;
```

나머지 파티션을 접속하십시오.

```
ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '02-01-1998' ENDING AT '02-28-1998' FROM sales_0298;

...

ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '12-01-2000' ENDING AT '12-31-2000' FROM sales_1200;
```

SET INTEGRITY문을 발행하여 접속된 데이터 파티션을 온라인으로 만드십시오.

```
SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;
```

적당하게 인덱스를 작성하십시오.

변환 고려사항

소스 컬럼 및 목표 컬럼 모두에 대해 특정 컬럼의 SYSCAT.COLUMNS IMPLICITVALUE 필드의 값이 널(NULL)이 아니고 값이 일치하지 않을 경우에는 데이터 파티션을 접속할 수 없습니다. 이 경우 소스 테이블을 삭제한 다음 다시 작성해야 합니다.

다음 조건 중 하나를 만족할 경우에는 컬럼이 SYSCAT.COLUMNS IMPLICITVALUE 필드에 널(NULL)이 아닌 값을 가질 수 있습니다.

- 컬럼이 ALTER TABLE ...ADD COLUMN문을 실행함으로써 작성된 경우
- IMPLICITVALUE 필드가 접속 중에 소스 테이블에서 전파된 경우
- IMPLICITVALUE 필드가 접속 해제 중에 소스 테이블로부터 상속된 경우
- IMPLICITVALUE 필드가 V8에서 V9로 이주하는 중에 설정된 경우. IMPLICITVALUE 필드는 추가된 컬럼으로 판별되거나 추가된 컬럼일 수 있습니다. 데이터베이스가 컬럼이 추가되는지 여부를 확실히 알 수 없는 경우 추가된 것으로 처리됩니다. 추가된 컬럼은 ALTER TABLE ...ADD COLUMN문을 실행할 때 작성되는 컬럼입니다.

이러한 불일치가 발생하지 않게 하려면, 항상 정의된 동일한 컬럼을 사용하여 접속 조치와 관련된 소스 및 목표 테이블을 작성하는 것이 좋습니다. 특히, ALTER TABLE 문을 사용하여 컬럼을 접속 조치의 목표 테이블에 추가하지 마십시오.

파티션된 테이블에 대해 작업할 때 불일치가 발생하지 않게 하는 법에 대한 우수 사례는 239 페이지의 『파티션된 테이블에 데이터 파티션을 접속하기 위한 지침』의 내용을 참조하십시오.

기존 인덱스를 파티션된 인덱스로 이주

시스템 작성 및 사용자 작성 인덱스는 파티션되지 않음에서 파티션됨으로 이주해야 할 수도 있습니다. 사용자 작성 인덱스는 대부분의 이주에서 테이블 및 인덱스에 대한 사용 가능성을 유지하면서 이주할 수 있습니다. 1차 키 제한조건 또는 고유 제한조건을 강제 실행하기 위해 사용된 시스템 작성 인덱스는 이주가 수행되는 동안 제한조건을 유지시킬 수 없습니다.

제품의 이전 릴리스에서 작성된 인덱스는 파티션되지 않을 수 있습니다. 여기에는 사용자 작성 인덱스 또는 데이터베이스 관리 프로그램에서 고유 및 1차 제한조건을 강제 실행하기 위해 작성된 시스템 작성 인덱스가 모두 포함될 수 있습니다.

사용자가 작성한 인덱스는 인덱스를 사용하여 데이터를 계속 사용할 수 있는 한 파티션되지 않음에서 파티션됨으로 이주할 수 있습니다. 파티션된 인덱스는 이에 대응하는 파티션되지 않은 인덱스와 동일한 키로 작성할 수 있습니다. 파티셔닝 인덱스가 작성되는 동안에는 현재 인덱스 및 인덱스가 작성되는 테이블을 계속 사용할 수 있습니다. 한 번 파티션된 인덱스가 작성되면, 이에 대응하는 파티션되지 않은 인덱스를 삭제하고 원하는 경우 새 파티션된 인덱스의 이름을 바꿀 수 있습니다.

이 태스크는 기존의 파티션되지 않은 인덱스에서 파티션된 인덱스로 이동하는 것입니다.

다음은 사용자가 작성한 파티션되지 않은 인덱스를 파티션된 인덱스로 변환하는 예제입니다.

```
UPDATE COMMAND OPTIONS USING C OFF;
CREATE INDEX data_part ON sales(sale_date) PARTITIONED;
COMMIT;
DROP INDEX dateidx;
RENAME INDEX data_part TO dateidx;
COMMIT;
```

다음은 데이터베이스 관리 프로그램에서 작성한 파티션되지 않은 인덱스를 파티션된 인덱스로 변환하는 예제입니다. 이 경우, 원래 제한조건 삭제와 새 제한조건 작성 사이에는 일정 시간 간격이 유지됩니다.

```
ALTER TABLE employees DROP CONSTRAINT emp_uniq;
ALTER TABLE employees ADD CONSTRAINT emp_uniq UNIQUE (employee_id);
```

파티션된 구체화된 쿼리 테이블(MQT) 동작

구체화된 쿼리 테이블(MQT)의 모든 유형은 파티션된 테이블로 지원됩니다. 파티션된 MQT로 작업할 때 접속 및 접속 해제된 데이터 파티션을 가장 효과적으로 관리할 수 있도록 도와주는 여러 개의 지침이 있습니다.

다음 지침 및 제한사항은 파티션된 MQT 또는 접속 해제된 종속이 있는 파티션된 테이블을 작업할 때 적용됩니다.

- DETACH PARTITION 조작을 발행하고 접속 해제된 데이터 파티션(이 종속 테이블을 접속 해제된 종속 테이블이라고 함)과 관련하여 유지보수해야 하는 종속 테이블이 있는 경우, 처음에는 새로 접속 해제된 테이블에 액세스할 수 없습니다. SYSCAT.TABLES 카탈로그 뷰의 TYPE 컬럼에 테이블이 L이라고 표시됩니다. 이를 접속 해제된 테이블이라고 합니다. 접속 해제되면, SET INTEGRITY문을 실행하여 증분 방식으로 접속 해제된 종속 테이블을 유지보수할 때까지 테이블을 읽거나 수정하거나 또는 삭제할 수 없습니다. 모든 접속 해제된 종속 테이블에서 SET INTEGRITY문을 실행하면, 접속 해제된 테이블은 완전히 액세스가 가능한 일반 테이블이 됩니다.
- 접속 해제된 테이블이 아직 액세스 가능하지 않음을 확인하려면 SYSCAT.TABDETACHEDDEP 카탈로그 뷰를 쿼리하십시오. 액세스 불가능한 접속 해제된 테이블이 발견되면, SET INTEGRITY문을 IMMEDIATE CHECKED 옵션과 함께 모든 접속 해제된 종속에 실행하여 접속 해제된 테이블을 일반 액세스 가능한 테이블로 전환하십시오. 모든 접속 해제된 종속이 유지보수되기 전에 접속 해제된 테이블에 액세스를 시도하면 SQL20285N 오류 코드가 리턴됩니다.
- DATAPARTITIONNUM 함수는 구체화된 쿼리 테이블(MQT) 정의에 사용할 수 없습니다. 이 함수를 사용해서 MQT를 작성하려고 시도하면 이 함수는 오류(SQLCODE SQL20058N, SQLSTATE 428EC)를 리턴합니다.
- 접속 해제된 데이터 파티션이 있는 테이블에 인덱스를 작성할 때, 데이터 파티션과 관련하여 대폭 새로 고쳐야 하는 종속 구체화된 쿼리 테이블(MQT)이 접속 해제된

데이터 파티션에 없으면 접속 해제된 데이터 파티션의 데이터를 인덱스가 포함하지 않습니다. 이 경우, 접속 해제된 데이터 파티션에 대한 데이터가 인덱스에 포함됩니다.

- MQT에 접속된 데이터 파티션이 있는 테이블은 변경할 수 없습니다.
- 파티션된 스테이징 테이블을 지원하지 않습니다.
- MQT에 바로 첨부할 수 없습니다. 세부사항은 예제 1을 참조하십시오.

예 1: 파티션된 MQT를 일반 테이블로 변환

ATTACH 조작이 파티션된 MQT에서 직접적으로 지원되지 않지만, 파티션된 MQT를 테이블 데이터의 지정 롤인 및 롤아웃을 실행하는 일반 테이블로 변환한 다음 다시 테이블을 MQT로 변환하여 동일한 효과를 획득할 수 있습니다. 다음 CREATE TABLE 및 ALTER TABLE문은 해당 효과를 설명합니다.

```
CREATE TABLE lineitem (
    l_orderkey    DECIMAL(10,0) NOT NULL,
    l_quantity    DECIMAL(12,2),
    l_shipdate    DATE,
    l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
        PARTITION BY RANGE(l_shipdate)
        (STARTING ('1/1/1992') ENDING ('12/31/1993') EVERY 1 MONTH);
CREATE TABLE lineitem_ex (
    l_orderkey    DECIMAL(10,0) NOT NULL,
    l_quantity    DECIMAL(12,2),
    l_shipdate    DATE,
    l_year_month  INT,
    ts            TIMESTAMP,
    msg           CLOB(32K));

CREATE TABLE quan_by_month (
    q_year_month, q_count) AS
      (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
          FROM lineitem
            GROUP BY l_year_month)
       DATA INITIALLY DEFERRED REFRESH IMMEDIATE
          PARTITION BY RANGE(q_year_month)
          (STARTING (199201) ENDING (199212) EVERY (1),
            STARTING (199301) ENDING (199312) EVERY (1));
CREATE TABLE quan_by_month_ex(
    q_year_month  INT,
    q_count       INT NOT NULL,
    ts            TIMESTAMP,
    msg           CLOB(32K));

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
CREATE INDEX qbmx ON quan_by_month(q_year_month);

ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;

SET INTEGRITY FOR li_reuse OFF;
ALTER TABLE li_reuse ALTER l_year_month SET GENERATED ALWAYS AS
(YEAR(l_shipdate)*100 + MONTH(l_shipdate));

LOAD FROM part_mqt_rotate.del OF DEL MODIFIED BY GENERATEDIGNORE
MESSAGES load.msg REPLACE INTO li_reuse;

DECLARE load_cursor CURSOR FOR
    SELECT l_year_month, COUNT(*)
```

```
          FROM li_reuse
        GROUP BY l_year_month;
LOAD FROM load_cursor OF CURSOR MESSAGES load.msg
    REPLACE INTO qm_reuse;

ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
    ENDING '1/31/1994' FROM li_reuse;

SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN lineitem USE lineitem_ex;

ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
    ENDING 199401 FROM qm_reuse;

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED
    FOR EXCEPTION IN quan_by_month USE quan_by_month_ex;

ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
    (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
          FROM lineitem
              GROUP BY l_year_month)
                  DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

SET INTEGRITY문을 IMMEDIATE CHECKED 옵션과 함께 사용해 무결성 위반에 대해 첨부된 데이터 파티션을 점검하십시오. 이 단계는 테이블을 MQT로 다시 변경하기 전에 실행해야 하는 필수 단계입니다. SET INTEGRITY문을 IMMEDIATE UNCHECKED 옵션과 함께 사용해 MQT의 필수 완전 새로 고침을 생략할 수 있습니다. MQT의 인덱스는 최적의 성능을 위해 필요합니다. SET INTEGRITY문과 함께 적절한 곳에 예외 테이블을 사용할 것을 권장합니다.

일반적으로 파티션된 MQT를 역시 파티션된 대형 사실 테이블에 작성합니다. 대형 사실 테이블의 테이블 데이터에 롤아웃 또는 롤인을 할 경우, 예 2의 설명처럼 파티션된 MQT를 수동으로 조정해야 합니다.

예 2: 파티션된 MQT를 수동으로 조정

MQT(quan_by_month)를 변경하여 일반 파티션된 테이블로 변환하십시오.

```
ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

사실 테이블(lineitem) 및 MQT에서 롤아웃될 데이터를 접속 해제한 다음 li_reuse 스테이징 테이블을 롤인할 새 데이터로 리로드하십시오.

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;

LOAD FROM part_mqt_rotate.del OF DEL MESSAGES load.msg REPLACE INTO li_reuse;

ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;
```

삽입 전에 qm_reuse를 프룬하십시오. 이는 subselect 데이터를 삽입하기 전에 접속 해제된 데이터를 삭제합니다. 프룬은 로드의 데이터 파일이 subselect의 내용인 MQT로 로드를 바꿔 이루어집니다.

```
db2 load from datafile.del of del replace into qm_reuse
```

INSERT INTO ... (SELECT ...)를 사용하여 수동으로 테이블을 새로 고칠 수 있습니다. 이는 새 데이터에만 필요하기 때문에 명령문을 접속하기 전에 발행해야 합니다.

```
INSERT INTO qm_reuse
    (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
              FROM li_reuse
         GROUP BY l_year_month);
```

이제 새 데이터를 사실 테이블에 롤인할 수 있습니다.

```
ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
ENDING '1/31/1994' FROM TABLE li_reuse;
SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR
EXCEPTION IN li_reuse USE li_reuse_ex;
```

다음으로 데이터를 MQT에 롤인합니다.

```
ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM TABLE qm_reuse;
SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
```

데이터 파티션을 접속한 후에 새 데이터가 범위 안에 있는지 확인하기 위해 새 데이터를 검증해야 합니다.

```
ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
    (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
          FROM lineitem
              GROUP BY l_year_month)
                  DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

데이터는 SET INTEGRITY문으로 유효성이 확인되면 액세스 가능합니다. REFRESH TABLE 조작이 지원되지만 이 시나리오는 ATTACH PARTITION 및 DETACH PARTITION 조작을 통한 파티션된 MQT의 수동 유지보수를 설명합니다. SET INTEGRITY문의 MMEDIATE UNCHECKED절을 통해 데이터가 유효성 확인되었다고 사용자가 표시합니다.

범위 클러스터 테이블(RCT) 작성

범위 클러스터 테이블(RCT)에 사용된 알고리즘

알고리즘은 레코드의 키 값과 테이블에 있는 특정 행의 위치가 같게 하는데 사용됩니다. 기본 알고리즘은 단순합니다. 대부분의 기본 양식(둘 이상의 컬럼 대신 단일 컬럼을 사용하여 키 구성)에서 알고리즘은 시퀀스 번호를 논리적 행 번호에 맵핑합니다.

또한 알고리즘은 레코드 키를 사용하여 논리적 페이지 번호 및 슬롯 번호를 결정합니다. 이러한 프로세스를 통해 레코드 즉, 테이블의 특정 행에 매우 빨리 액세스할 수 있습니다.

해싱은 키 값 순서를 보존하지 않기 때문에 알고리즘에는 해싱이 포함되지 않습니다. 키 값 순서를 보존하면 시간이 지나도 테이블 데이터를 재구성할 필요가 없기 때문에 키 값 순서 보존이 필요합니다.

테이블에 있는 각 레코드 키는 다음과 같은 특성을 가져야 합니다.

- 고유
- 널(NULL) 아님
- 정수(SMALLINT, INTEGER 또는 BIGINT)
- 일정한 증가
- 키의 각 컬럼에 따라 사전 결정된 범위 세트 내에 있음

  테이블 작성 시 ALLOW OVERFLOW 옵션을 사용하면 키 값이 정의된 범위를 초과할 수 있습니다. 테이블 작성 시 DISALLOW OVERFLOW 옵션을 사용하면 키 값이 정의된 범위를 초과할 수 없습니다. 이런 경우, 범위에서 지정한 바운더리 밖에서 레코드가 삽입된 경우에는 SQL 오류 메시지가 리턴됩니다.

시퀀스 키 범위가 촘촘하게 클러스터된(조밀한) 응용프로그램은 범위 클러스터 테이블의 좋은 예입니다. 이런 유형의 키를 사용하여 범위 클러스터 테이블을 작성하면 테이블에 있는 행의 논리적 위치를 생성하는데 해당 키가 사용됩니다. 이 프로세스를 사용하면 별도의 인덱스가 필요하지 않습니다.

범위 클러스터 테이블(RCT) 인덱스

범위 클러스터 테이블(RCT)에서 인덱스는 레코드에서 키 기반 레코드를 찾고 시작 및 중지 스캔을 적용하고 데이터를 수직으로 분배합니다. RCT를 사용할 경우, 고려하지 않은 인덱스의 유일한 등록 정보는 데이터의 수직 파티션입니다.

일반 테이블과의 다른 점

범위 클러스터 테이블(RCT)을 사용하기로 결정한 경우에는 일반 테이블과 구분되는 몇 가지 특성을 고려해야 합니다.

- 범위 클러스터 테이블에는 여유 공간 제어 레코드(FSCR)가 없습니다.
- 스페이스가 사전 할당됩니다.

  테이블 레코드가 채워지지 않았을 때도 테이블에서 사용할 테이블 스페이스를 미리 할당하거나 예약합니다. 테이블 작성 시 테이블에 레코드가 없어도 페이지의 전체 범위가 미리 할당됩니다. 사전 할당은 저장할 최대 레코드의 수 및 레코드 *크기*에 따라 결정됩니다.

  – VARCHAR와 같은 가변 길이 필드를 각 레코드에 사용하는 경우 필드의 최대 길이가 사용되고 전체 레코드 *크기*는 고정 길이가 됩니다. 최대 레코드 수와 함께 각 레코드의 전체 고정 길이를 사용하여 필요한 스페이스를 결정합니다.

  – 따라서 효과적으로 활용될 수 없는 추가 스페이스가 할당됩니다.

  - 키 값이 조밀하지 않은 경우에는 사용하지 않은 스페이스가 있으며 범위 스캔 성능이 떨어집니다.
  - 해당 키 값이 포함된 행이 아직 데이터베이스에 삽입되지 않은 경우에도 범위 스캔은 범위 내의 모든 가능한 레코드를 찾아야 합니다.
- 스키마 수정은 허용되지 않습니다.

  범위 클러스터 테이블에서 스키마 수정이 필요하면 테이블에 대한 새 스키마 이름 및 이전 테이블의 모든 데이터가 포함되도록 테이블을 다시 작성해야 합니다. 특히 다음 사항을 고려하십시오.

  - 키 범위의 변경은 지원되지 않습니다.

    테이블의 범위를 변경해야 하는 경우에는 원하는 범위의 새 테이블을 작성해야 하며 새 테이블을 이전 테이블의 데이터로 채워야 하므로 이는 중요합니다.
- 중복 키 값은 허용되지 않습니다.
- 정의된 범위밖의 키 값은 허용되지 않습니다.

  이는 DISALLOW OVERFLOW에 정의된 범위 클러스터 테이블의 경우에만 해당합니다.

  - NULL 값은 명시적으로 허용되지 않습니다.
- 범위 클러스터 인덱스가 구체화되지 않습니다.

  시스템 카탈로그에서 RCT 키 등록 정보를 가진 인덱스를 표시하고 옵티마이저를 사용하여 인덱스를 선택해도 인덱스가 디스크상에 구체화되지는 않습니다. 일반 테이블을 사용하면 테이블과 관련된 각 인덱스에 스페이스도 제공해야 합니다. RCT를 사용하면 RCT 인덱스에는 스페이스가 필요하지 않습니다. 옵티마이저는 해당 RCT 인덱스를 참조하는 시스템 카탈로그에 있는 정보를 사용하여 테이블에 필요한 올바른 액세스 메소드를 선택하도록 합니다.
- 범위 클러스터 테이블 인덱스와 동일한 정의에 기본 또는 고유 키를 작성하는 것은 중복되므로 허용되지 않습니다.
- 범위 클러스터 테이블에는 원본 키 값 순서 즉, 테이블에 있는 행을 클러스터링하는 기능이 있습니다.

범위 클러스터 테이블(RCT) 사용 지침

범위 클러스터 테이블(RCT)에 대한 작업 시 따라야 할 몇 가지 지침이 있습니다.

- 키 값의 범위를 정의할 때 최소값은 선택사항입니다. 지정하지 않으면 디폴트값은 1입니다. 최소값 및 최대값으로 음수값을 허용합니다. 음수값을 사용하는 경우 최소값을 반드시 표시해야 합니다. 예를 들어, 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.

SQL 컴파일러의 범위 클러스터 테이블(RCT)에 대한 작업 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 판별하는 대신, 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 테이블은 일반 테이블로 모델화되고 인덱스는 기능 기반 인덱스로 모델화됩니다.

오버플로우를 허용하는 범위 클러스터 테이블 작성 시 테이블 레코드의 순서는 지켜지지 않습니다.

시나리오: 범위 클러스터 테이블(RCT)

간단한 이 시나리오는 범위 클러스터 테이블 작성 방법을 보여줍니다. 이 시나리오는 테이블 키로서 단일 컬럼 또는 다중 컬럼을 사용하는 방법을 보여줍니다. 또한 데이터 오버플로우를 허용하는 테이블 및 데이터 오버플로우를 허용하지 않는 테이블을 작성하는 방법을 보여줍니다.

시나리오 1: 범위 클러스터 테이블(RCT) 작성

이 시나리오는 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 경우, 학생 레코드는 STUDENT_ID에만을 기초로합니다. STUDENT_ID를 사용하여 학생 레코드를 추가, 갱신 및 삭제합니다.

주: 나중에 별도의 인덱스를 추가할 수도 있습니다. 단, 이 예의 용도상 테이블 작성 시 테이블의 구성 및 테이블 데이터에 액세스하는 방법이 정의됩니다.

다음은 이 테이블에 필요한 구문입니다.

```
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.)

시나리오 2: 범위 클러스터 테이블(RCT) 작성 (오버플로우 허용되지 않음)

첫 번째 변형인 이 시나리오에서는 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.

이 경우 학생 레코드는 SCHOOL_ID, CLASS_ID 및 STUDENT_NUM 값의 세 가지 인수를 기초로합니다. 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.

다음은 이 테이블에 필요한 구문입니다.

```
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 규정이 있기 때문입니다. 이 시나리오에서 가장 큰 학급의 크기는 35명입니다. 해당 인수를 학급과 학교 수에 대한 실제 한계와 결합시키면 교육 위원회의 학생 수에 오버플로우를 허용할 이유가 없음이 명백해집니다.

학교 마다 총 학급 수가 다를 수 있습니다. 이런 경우 학급 수의 범위 정의시 (CLASS_ID 사용) 상한선은 모든 학교를 고려할 때 가장 큰 학급 수가 되어야 합니다. 즉, 몇몇 작은 규모의 학교(가장 규모가 큰 학교보다 학급 수가 적은 학교)의 경우, 예를 들어 간이 학급이 학교에 추가되지 않으면 한 번도 사용되지 않을 수 있는 학생 레코드 스페이스가 생길 수 있습니다.

이전 예와 마찬가지로 동일한 4KB 페이지 크기 및 동일한 학생 레코드 크기를 사용하면 페이지당 42개의 레코드가 가능합니다. 140,000개의 학생 레코드를 사용하면 3333.3페이지 또는 반올림하여 3334페이지가 필요합니다. 테이블 정보에 2페이지가 필요하고 Extent 맵핑에 3페이지가 필요합니다. 그 결과 4KB 크기의 3339페이지가 미리 할당되어야 합니다.

MDC 테이블을 작성 시 고려사항

MDC 테이블을 작성할 때 고려해야 할 요소에는 여러 가지가 있습니다. 다음 절에서는 MDC 테이블을 작성, 배치 및 사용하는 방법에 대한 결정이 현재 데이터베이스 환경(예: 파티션된 데이터베이스가 존재 또는 없음) 및 MDC 테이블에 대한 차원 선택에 따라 어떻게 영향을 받을 수 있는지에 대해 설명합니다. 또한 DB2 디자인 어드바이저 및 일부 문제에 대한 권장사항을 제공하기 위해 이 도구를 어떻게 사용하는지에 대해서도 설명합니다.

기존 테이블의 데이터를 MDC 테이블로 이동

쿼리 성능을 개선하고 데이터 웨어하우스 또는 대형 데이터베이스 환경에서 데이터 유지보수 조작의 오버헤드를 줄이기 위해 일반 테이블에서 다차원적으로 클러스터된(MDC) 테이블로 데이터를 이동할 수 있습니다. 기존 테이블에서 MDC 테이블로 데이터를 이동시키려면 데이터를 익스포트하고 원래의 테이블을 삭제한 후(선택사항), 다차원적으로 클러스터된(MDC) 테이블을 작성하여(CREATE TABLE문에서 ORGANIZE BY DIMENSIONS절 사용) MDC 테이블에 데이터를 로드하십시오.

SYSPROC.ALTOBJ라고 하는 ALTER TABLE 프로시저는 기존 테이블에서 MDC 테이블로 데이터 변환을 수행하는데 사용될 수 있습니다. 이 프로시저는 DB2 디자인 어드바이저에서 호출됩니다. 테이블 간의 데이터 변환에 필요한 시간은 오래 걸릴 수 있으며 테이블의 크기 및 변환이 필요한 데이터의 양에 따라 달라집니다.

ALTOBJ 프로시저는 테이블 변경을 수행할 때 다음을 수행합니다.

- 테이블의 모든 종속 오브젝트 삭제
- 테이블 이름 바꾸기
- 새 정의를 사용하여 테이블 작성
- 테이블의 모든 종속 오브젝트 다시 작성
- 테이블의 기존 데이터를 새 테이블에서 필요한 데이터로 변환. 즉, 이전 테이블에서 데이터를 선택하고 컬럼 함수를 사용하여 이전 데이터 유형에서 새 데이터 유형으로 변환할 수 있는 새 테이블로 해당 데이터를 로드합니다.

SMS 테이블 스페이스의 MDC 테이블

MDC 테이블을 SMS 테이블 스페이스에 저장하려는 경우에는 다중 파일 할당을 사용해야 합니다. (다중 페이지 파일 할당은 버전 8.2 이상에서 새로 작성된 데이터베이스에 대한 디폴트값입니다.) 이 이유는 MDC 테이블은 항상 전체 Extent에 의해 확장되며 이러한 Extent의 모든 페이지가 물리적으로 연속된다는 사실이 중요하기 때문입니다. 따라서 다중 페이지 파일 할당을 사용하지 않는 것이 스페이스 측면에서 아무런 이점이 없습니다. 더구나 다중 페이지 파일 할당을 사용하는 경우에는 각 Extent의 페이지가 물리적으로 연속될 가능성이 매우 증대됩니다.

DB2 디자인 어드바이저의 MDC 어드바이저 기능

DB2 디자인 어드바이저(db2advis)에는 MDC 기능이 있습니다. 이 기능은 MDC 테이블에서 사용할 클러스터링 차원을 권장하며, 여기에는 워크로드 성능을 개선하기 위해 기본 컬럼에서의 개략화가 포함됩니다. *개략화(coarsification)*는 클러스터링 차원의 카디널리티(개별 값의 수)를 줄이는 작업의 수학적 표현을 의미합니다. 개략화의 일반적인 예로는 일, 주, 월 또는 분기로 개략화가 이루어질 수 있는 날짜를 들 수 있습니다.

DB2 디자인 어드바이저의 MDC 기능을 사용하려면 데이터베이스에 최소 여러 개의 데이터 Extent가 있어야 합니다. DB2 디자인 어드바이저는 데이터를 사용하여 데이터 밀도 및 카디널리티를 표시합니다.

데이터베이스 테이블에 데이터가 없는 경우, DB2 디자인 어드바이저는 MDC를 권장하지 않습니다. 데이터베이스에 빈 테이블이 있지만 채워진 데이터베이스를 의미하는 통계 모형이 있는 경우에도 마찬가지입니다.

권장사항에는 차원의 개략화를 정의하는 잠재적, 생성된 컬럼의 식별이 포함됩니다. 권장사항에는 가능한 블록 크기가 포함되어 있지 않습니다. 테이블 스페이스의 Extent 크

기는 MDC 테이블에 대한 권장사항을 작성할 때 사용됩니다. 권장 MDC 테이블은 기존 테이블과 동일한 테이블 스페이스에서 작성되므로 동일한 Extent 크기를 가진다고 가정합니다. MDC 차원에 대한 권장사항은 테이블 스페이스의 Extent 크기에 따라 변경될 수 있습니다. 왜냐하면 Extent 크기가 블록이나 셀에 적합한 레코드의 수에 영향을 주기 때문입니다. 이는 셀의 밀도에 직접적으로 영향을 줍니다.

테이블에 대해 단일 또는 다중 차원을 권장할 수 있지만 단일 컬럼 차원(복합 컬럼 차원이 아닌 차원)만 고려됩니다. MDC 기능은 결과적인 MDC 솔루션에서 셀의 카디널리티(cardinality)를 줄이는 것을 목적으로 하며, 지원되는 대부분의 데이터 유형에 대해 개략화를 권장합니다. 데이터 유형 예외에는 CHAR, VARCHAR, GRAPHIC 및 VARGRAPH 데이터 유형이 포함됩니다. 지원되는 모든 데이터 유형은 INTEGER로 캐스트되며 생성된 표현식을 통해 개략화됩니다.

DB2 디자인 어드바이저의 MDC 기능을 사용하는 목적은 성능을 향상시키는 MDC 솔루션을 선택하기 위함입니다. 두 번째 목적은 데이터베이스의 스토리지 확장을 최적 레벨로 제한되도록 유지하는 것입니다. 통계적인 메소드가 각 테이블에서 최대 스토리지 확장을 결정하는데 사용됩니다.

어드바이저에서 수행되는 분석 조작에는 블록 인덱스 액세스의 이점은 물론 테이블의 차원에 대한 삽입, 갱신, 삭제 조작에 미치는 MDC의 영향도 포함됩니다. 테이블에서의 이러한 조치로 인해 레코드가 셀 간에 이동될 수도 있습니다. 또한 분석 조작은 특정 MDC차원에서의 데이터 구성에 따른 임의의 테이블 확장이 잠재적으로 성능에 미치는 영향을 모델링합니다.

MDC 기능은 db2advis 유틸리티에서 `-m <advise type>` 플래그를 사용하여 실행합니다. 『C』 권장 유형은 다차원적으로 클러스터된 테이블을 나타내는데 사용됩니다. 권장 유형은 『I』(인덱스용), 『M』(구체화된 쿼리 테이블용), 『C』(MDC용) 및 『P』(파티션된 데이터베이스 환경용)입니다. 권장 유형은 서로간에 조합해서 사용이 가능합니다.

주: DB2 디자인 어드바이저는 크기가 12 Extent 미만인 테이블은 탐색하지 않습니다.

어드바이저는 권장사항을 제안할 때 MQT 및 일반 기본 테이블 모두를 분석합니다.

MDC 기능의 출력에는 다음이 포함됩니다.

- MDC 솔루션에 나타나는 개략화된 차원의 각 테이블에 대해 생성된 컬럼 표현식
- 각 테이블에 대해 권장되는 ORGANIZE BY절

권장사항은 Explain 기능의 일부인 ADVISE 테이블 및 표준 출력으로 보고됩니다.

MDC 테이블 및 파티션된 데이터베이스 환경

다차원 클러스터링을 파티션된 데이터베이스 환경과 함께 사용할 수 있습니다. 실제로 MDC는 파티션된 데이터베이스 환경을 보완할 수 있습니다. 파티션된 데이터베이스 환경은 다음 목적을 위해 다수의 물리적 또는 논리적 노드에 테이블의 데이터를 분산하는데 사용됩니다.

- 다중 시스템을 사용하여 처리 요청을 병렬로 늘리기
- 단일 데이터베이스 파티션의 한계를 넘어 테이블의 물리적 *크기* 늘리기
- 데이터베이스 확장성 개선

테이블을 분배하는 이유는 테이블이 MDC 테이블인지 또는 일반 테이블인지 여부와 무관합니다. 예를 들어, 분산 키를 구성하기 위해 컬럼을 선택하는 규칙은 동일합니다. MDC 테이블에 대한 분산 키에는 컬럼이 테이블의 차원의 일부를 구성하는지 여부와 상관없이 모든 컬럼이 포함될 수 있습니다.

분산 키가 테이블의 차원과 동일하면 각 데이터베이스 파티션에는 테이블의 다른 부분이 포함됩니다. 예를 들어, 예제 MDC 테이블이 두 데이터베이스 파티션에서 색상별로 파티션되면 Color 컬럼이 데이터 분할을 위해 사용됩니다. 그 결과 빨간색 및 파란색 조각이 한 파티션에 있고 노란색 조각이 다른 파티션에 있을 수 있습니다. 분산 키가 테이블의 차원과 동일하지 않으면 각 데이터베이스 파티션은 각 조각에서 데이터의 서브세트를 갖습니다. 차원을 선택하고 셀 점유율을 계산 할 때(『셀 밀도』 절 참조), 일반적으로 셀당 전체 데이터 양은 모든 데이터를 모아 이를 파티션 수로 나누어서 결정된다는 점을 참고하십시오.

다중 차원이 있는 MDC 테이블

특정 술어가 쿼리에서 많이 사용될 것 같으면 ORGANIZE BY DIMENSIONS절을 사용하여, 관련된 컬럼에서 테이블을 클러스터할 수 있습니다.

예 1

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

예 1의 테이블은 하나의 논리적 큐브를 형성하는 3개의 원시(native) 컬럼 내의 값에 클러스터됩니다(즉, 차원이 3개임). 해당 조각이나 셀의 블록만 관련된 관계 연산자에 의해 처리되도록 하나 이상의 차원에 대한 쿼리 처리 중 테이블을 논리적으로 조각으로 나눌 수 있습니다. 블록 *크기*(페이지 수)는 테이블의 Extent *크기*가 된다는 점을 참고하십시오.

둘 이상의 컬럼을 기반으로 하는 차원이 있는 MDC 테이블

각 차원은 하나 이상의 컬럼으로 구성될 수 있습니다. 예를 들어, 두 컬럼이 포함된 차원에 클러스터되는 테이블을 작성할 수 있습니다.

예 2

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

예 2에서, 테이블은 두 개의 차원인 c1과 (c3, c4)에서 클러스터됩니다. 따라서 쿼리를 처리할 때 c1 차원이나 복합 (c3, c4) 차원에서 테이블을 논리적 조각으로 나눌 수 있습니다. 테이블의 블록 수는 예 1의 테이블과 동일하지만 차원 블록 인덱스는 하나 적습니다. 예 1에는 컬럼 c1, c3 및 c4에 대해 각각 하나씩 세 개의 차원 블록 인덱스가 있습니다. 예 2에는 컬럼 c1에 대해 하나와 컬럼 c3 및 c4에 대해 하나인 두 개의 차원 블록 인덱스가 있습니다. 이 두 예의 차이점은 예 1에서는 c4와만 관련된 쿼리가 c4에 대한 차원 블록 인덱스를 사용하여 관련 데이터의 블록에 빠르게 직접 액세스할 수 있다는 점입니다. 예 2에서는 c4가 차원 블록 인덱스의 두 번째로 중요한 부분이므로 c4만 관련된 쿼리에는 추가 처리가 필요합니다. 그러나 예 2에서는 유지보수 및 저장할 블록 인덱스가 하나 줄게 됩니다.

DB2 디자인 어드바이저는 둘 이상의 컬럼이 포함된 차원을 권장하지 않습니다.

차원으로서의 컬럼 표현식이 있는 MDC 테이블

컬럼 표현식은 차원 클러스터링에도 사용할 수 있습니다. 컬럼 표현식에 대해 클러스터하는 기능은 지리적 위치 또는 지역으로 주소를 롤업하거나 날짜를 주, 월 또는 연도로 롤업하는 것과 같이 차원을 보다 낮은 세분화도로 롤업하는 경우에 유용합니다. 이러한 방식으로 차원의 롤업을 구현하기 위해, 생성된 컬럼을 사용할 수 있습니다. 이러한 유형의 컬럼 정의는 차원을 나타낼 수 있는 표현식을 사용하여 컬럼을 작성할 수 있게 합니다. 예 3에서 명령문은 하나의 기본 컬럼 및 두 개의 컬럼 표현식에 대해 클러스터된 테이블을 작성합니다.

예 3

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
      c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
            ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

예 3에서, 컬럼 c5는 컬럼 c3 및 c4를 기반으로 하는 표현식인 반면, 컬럼 c6은 컬럼 c1을 적절한 때에 보다 낮은 세분화도로 롤업합니다. 이 명령문은 컬럼 c2, c5 및 c6의 값에 따라 테이블을 클러스터합니다.

생성된 컬럼 차원에 대한 범위 쿼리

단수 컬럼 함수가 필요한 생성된 컬럼 차원의 범위 쿼리 표현식은 생성된 컬럼의 차원에 대한 범위 술어를 유도하기 위해 단순해야 합니다. 생성된 컬럼에 차원을 작성하는 경우, 기본 컬럼에 대한 쿼리는 한 가지 예외를 제외하고는 생성된 컬럼의 블록 인덱스를 사용하여 성능을 향상시킵니다. 차원 블록 인덱스에서 범위 스캔을 사용하는 기본 컬럼(예: 날짜)에 대한 범위 쿼리의 경우, CREATE TABLE문에서 컬럼을 생성하기 위

해 사용되는 표현식은 단순해야 합니다. 컬럼 표현식에는 유효한 모든 표현식(사용자 정의 함수(UDF) 포함)이 포함될 수 있지만, 표현식이 단순하지 않으면 등호 또는 IN 술어가 기본 컬럼에 있을 때 이 술어만 블록 인덱스를 사용하여 쿼리를 충족시킬 수 있습니다.

예를 들어, 생성된 컬럼 'month'에서 차원을 갖는 MDC 테이블을 작성한다고 가정합시다(`month = INTEGER (date)/100`). 차원(month) 쿼리의 경우에는 블록 인덱스 스캔이 수행될 수 있습니다. 기본 컬럼(date) 쿼리의 경우에는 블록 인덱스 스캔을 사용하여 스캔할 블록을 좁힐 수도 있으며 그런 다음 'date'의 술어를 해당 블록에만 있는 행에 적용할 수 있습니다.

컴파일러는 블록 인덱스 스캔에서 사용할 추가 술어를 생성합니다. 예를 들어, 다음의 쿼리에서

```
SELECT * FROM MDCTABLE WHERE DATE > "1999-03-03" AND DATE < "2000-01-15"
```

컴파일러는 추가 술어 『month >= 199903』 및 『month <= 200001』을 생성하는데, 이는 차원 블록 인덱스 스캔에 대한 술어로 사용할 수 있습니다. 결과 블록을 스캔하는 중에 원래 술어는 블록의 행에 적용됩니다.

단순하지 않은 표현식에서는 해당 차원에 등호 술어만 적용할 수 있습니다. 단순하지 않은 함수의 좋은 예로는 예 3의 컬럼 c6에 대한 정의에서 볼 수 있는 MONTH( )가 있습니다. c1 컬럼이 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우, 이 함수는 1 - 12 범위 내의 정수 값을 리턴합니다. 함수의 결과가 결정적이더라도 실제로는 단계 함수(즉, 순환 패턴)의 결과와 비슷합니다.

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

이 예의 날짜가 계속해서 증가하더라도 MONTH(date)는 증가하지 않습니다. 보다 자세하게 말하자면 date1이 date2보다 클 때 MONTH(date1)이 MONTH(date2)보다 항상 크거나 같다고 보장할 수는 없습니다. 이 상태가 바로 단순성에 필요한 조건입니다. 이러한 비단순성은 허용되지만 기본 컬럼에 대한 범위 술어가 차원에 대한 범위 술어를 생성할 수 없다는 점에서 차원을 제한합니다. 그러나 예를 들어, `where month(c1) between 4 and 6`과 같이 표현식에 대한 범위 술어는 허용됩니다. 시작 키로 4를 사용하고 중지 키로 6을 사용하여 차원에 대한 인덱스를 평상시와 같이 사용할 수 있습니다.

이 함수를 단순하게 하려면 연도를 월의 상위 부분으로 포함시켜야 합니다. 이는 INTEGER 내장 함수의 확장을 제공하므로 날짜에 대한 단순한 표현식을 쉽게 정의할 수 있습니다. INTEGER(date)는 날짜의 정수 표현을 리턴하며 이 표현을 나누어 연도와 월의 정수 표현을 찾을 수 있습니다. 예를 들어, `INTEGER(date('2000/05/24'))`는 `20000524`를 리턴하므로 `INTEGER(date('2000/05/24'))/100 = 200005`입니다. INTEGER(date)/100은 단순 함수입니다.

마찬가지로 내장 함수 DECIMAL 및 BIGINT도 확장하여 사용할 수 있으므로 단순 함수를 파생시킬 수 있습니다. DECIMAL(timestamp)는 시간소인의 10진수 표현을 리턴하므로 이를 단순한 표현식에 사용하여 월, 일, 시, 분 등에 대한 증가 값을 파생시킬 수 있습니다. BIGINT(date)는 INTEGER(date)와 마찬가지로 날짜의 큰 정수 표현을 리턴합니다.

데이터베이스 관리 프로그램은 테이블에 대해 생성된 컬럼을 작성할 때나 차원 절의 표현식에서 차원을 작성할 때 가능한 한 표현식의 단순성을 판별합니다. DATENUM( ), DAYS( ), YEAR( )와 같은 특정 함수는 단순성을 유지하는 것으로 간주할 수 있습니다. 또한, 컬럼 및 상수의 나누기, 곱하기 또는 더하기와 같은 다양한 수학 표현식은 단순성을 유지합니다. DB2가 표현식이 단순성을 유지하지 않는 것으로 판별하거나 이를 판별할 수 없는 경우, 차원은 기본 컬럼에 등호 술어만 사용하도록 지원합니다.

제 12 장 데이터베이스 변경

인스턴스 변경

다중 데이터베이스 파티션 전반에 데이터베이스 구성 변경

둘 이상의 데이터베이스 파티션에 걸쳐 분산된 데이터베이스가 있으면, 데이터베이스 구성 파일은 모든 데이터베이스 파티션에서 동일해야 합니다.

SQL 컴파일러는 노드 구성 파일의 정보에 근거하여 분산 SQL문을 컴파일하고, SQL 문의 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_all을 사용하십시오.

데이터베이스 변경

데이터베이스 파티션 그룹 변경

데이터베이스 파티션 그룹에 데이터베이스 파티션을 추가 또는 삭제(drop) ALTER DATABASE PARTITION GROUP문을 사용하십시오. 데이터베이스 파티션을 추가 또는 삭제한 후 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 현재 데이터를 데이터베이스 파티션 그룹의 새 데이터베이스 파티션 세트에 재분배하십시오.

제어 센터에서 데이터베이스 파티션 관리

제어 센터의 데이터베이스 파티션 뷰를 사용하여 데이터베이스 파티션에 대해 작업할 수 있습니다.

데이터베이스 파티션에 대해 작업하려면 인스턴스에 접속할 권한이 필요합니다. SECADM 또는 ACCESSCTRL 권한이 있는 사람은 누구나 특정 인스턴스에 액세스할 수 있는 권한을 부여할 수 있습니다.

데이터베이스 파티션을 구성하거나 데이터베이스 파티션을 롤 포워드 보류 상태에서 제거하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있어야 합니다.

데이터베이스 파티션 보기를 사용하여 데이터베이스 파티션을 재시작, 롤 포워드 보류 상태에서 제거, 백업, 리스토어 또는 구성 어드바이저를 사용하여 데이터베이스 파티션을 구성합니다.

제어 센터에서 데이터베이스 파티션 뷰를 열려면 다음과 같이 수행하십시오.

1. 제어 센터에서 데이터베이스 파티션을 보려는 파티션된 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. 원하는 파티션된 데이터베이스를 마우스 오른쪽 단추로 누르고 메뉴 목록에서 데이터베이스 파티션 열기를 선택하십시오.
3. 선택된 파티션된 데이터베이스의 데이터베이스 파티션 뷰가 열립니다.

데이터베이스 파티션을 구성하려면 다음과 같이 수행하십시오.

1. 데이터베이스 파티션 뷰에서 원하는 데이터베이스 파티션을 선택하십시오.
2. 데이터베이스 파티션을 선택하고 마우스 오른쪽 단추로 눌러 목록에서 구성 어드바이저를 선택하십시오.
3. 구성 어드바이저가 열립니다. 구성 어드바이저를 사용하여 데이터베이스 구성 매개변수의 값을 지정하십시오.

제 13 장 테이블 및 기타 관련 테이블 오브젝트 변경

파티션된 테이블 변경

파티션된 테이블에서 ALTER TABLE문에 관련된 모든 절이 지원됩니다. 또한 ALTER TABLE문을 사용하여 새 데이터 파티션을 추가하고 새 데이터 파티션을 롤인(접속)하며 기존의 데이터 파티션을 롤아웃(접속 해제)할 수 있습니다.

데이터 파티션을 접속 해제하도록 파티션을 변경하기 위해서는 사용자가 다음과 같은 권한 또는 특권을 갖고 있어야 합니다.

- DETACH 조작을 수행하는 사용자는 소스 테이블에서 AITER, SELECT 및 DELETE를 수행하는데 필요한 권한을 갖고 있어야 합니다.
- 또한 목표 테이블을 작성하는데 필요한 권한도 갖고 있어야 합니다. 따라서, 데이터 파티션을 접속 해제하도록 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권이 목표 테이블에 대해 최소한 다음과 같은 권한 또는 특권 중 하나를 포함해야 합니다.
  - DBADM 권한
  - 다음 권한 중 하나를 비롯하여 테이블에 사용되는 테이블 스페이스에 대한 USE 특권 및 데이터베이스에 대한 CREATETAB 권한
    - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

데이터 파티션에 접속하도록 파티션된 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권이 소스 테이블에 대해 최소한 다음과 같은 권한 또는 특권 중 하나를 포함해야 합니다.

- 소스 테이블에 대한 DATAACCESS 권한 또는 SELECT 특권 및 소스 테이블의 스키마에 대한 DBADM 권한 또는 DROPIN 특권
- 소스 테이블에 대한 CONTROL 특권

데이터 파티션을 추가하도록 파티션된 테이블을 변경하기 위해서는 명령문의 권한 부여 ID 가 보유하는 특권에 새 파티션이 추가되는 테이블 스페이스를 사용하는 특권이 있어야 하고 소스 테이블에 대해 최소한 다음과 같은 권한 또는 특권 중 하나가 포함되어 있어야 합니다.

- ALTER 특권
- CONTROL 특권

- DBADM
- 테이블 스키마에 대한 ALTERIN 특권

사용 지침

- PARTITION절과 함께 발행된 각 ALTER TABLE문은 다른 SQL문에 있어야 합니다.
- ALTER TABLE...PARTITION 조작을 포함하는 SQL문에서는 다른 ALTER 조작이 허용되지 않습니다. 예를 들어, 단일 SQL문으로 데이터 파티션에 접속하고 테이블에 컬럼을 추가할 수 없습니다.
- 복수의 ALTER문을 실행한 다음 단일 SET INTEGRITY문을 실행할 수 있습니다.

명령행에서 파티션된 테이블을 변경하려면 ALTER TABLE문을 발행하십시오.

파티션된 테이블 변경에 대한 지침 및 제한사항

이 주제에서는 가장 일반적인 테이블 변경 조치와 접속 및 접속 해제 데이터 파티션에서의 특별 고려사항을 식별합니다.

외부 키 제한조건에 대한 점검 추가 또는 변경

접속 및 접속 해제된 외부 키 제한조건에 대한 점검 추가가 지원됩니다. 파티션된 테이블이 접속 해제된 파티션 'D' 상태에 있을 때 1차 또는 고유 제한조건을 추가하면 시스템이 제한조건을 강제 실행하기 위해 새 파티션된 인덱스를 생성해야 하는 경우 오류가 리턴됩니다.

컬럼 추가

접속된 데이터 파티션이 있는 테이블에 컬럼을 추가하면 접속된 데이터 파티션에도 컬럼이 추가됩니다. 접속 해제된 데이터 파티션이 있는 테이블에 컬럼을 추가하면, 접속 해제된 데이터 파티션은 더 이상 테이블에 실제로 연관되어 있지 않기 때문에 접속 해제된 데이터 파티션에는 컬럼이 추가되지 않습니다.

컬럼 변경

접속된 데이터 파티션이 있는 테이블의 컬럼을 변경하면 접속된 데이터 파티션의 컬럼도 변경됩니다. 접속 해제된 데이터 파티션이 있는 테이블의 컬럼을 변경하면, 접속 해제된 데이터 파티션은 더 이상 테이블에 실제로 연관되어 있지 않기 때문에 접속 해제된 데이터 파티션에서 컬럼이 변경되지 않습니다.

생성된 컬럼 추가

접속 또는 접속 해제된 파티션된 테이블에 생성된 컬럼을 추가할 경우, 다른 유형의 컬럼을 추가하기 위한 규칙을 고려해야 합니다.

파티션되지 않은 인덱스 추가 또는 수정

접속된 데이터 파티션이 있는 테이블에서 인덱스를 작성, 재작성 또는 재구성할 경우, SET INTEGRITY문은 모든 접속된 데이터 파티션에 대한 모든 인덱스를 유지보수하기 때문에 접속된 데이터 파티션의 데이터는 인덱스에 포함

되지 않습니다. 접속 해제된 데이터 파티션이 있는 테이블에서 인덱스를 작성, 재작성 또는 재구성할 경우, 접속 해제된 데이터 파티션에 데이터 파티션과 관련하여 대폭으로 새로 고쳐야 하는 접속 해제된 종속 테이블 또는 스테이징 테이블이 없다면 접속 해제된 데이터 파티션의 데이터는 인덱스에 포함되지 않습니다. 이 경우, 접속 해제된 데이터 파티션에 대한 데이터가 인덱스에 포함됩니다.

파티션된 인덱스 추가 또는 수정

접속된 데이터 파티션이 있는 위치에서 파티션된 인덱스를 작성할 때 접속된 각 데이터 파티션의 인덱스 파티션이 작성됩니다. 접속된 데이터 파티션을 온라인으로 가져오기 위해 SET INTEGRITY문이 실행될 때까지 접속된 데이터 파티션의 인덱스 파티션에 대한 인덱스 항목이 보이지 않습니다. 인덱스 작성은 접속된 데이터 파티션을 포함하므로, 고유 파티션된 인덱스를 작성하면 접속된 데이터 파티션에서 중복 키 값이어서 인덱스 작성에 실패한 행을 찾을 수 있습니다. 사용자는 이 문제점을 피하기 위해 접속된 파티션이 있는 위치에 파티션된 인덱스의 작성을 시도하지 않는 것이 좋습니다.

접속 해제된 데이터 파티션이 있는 테이블에서 파티션된 인덱스 작성이 지원되며 접속 해제된 데이터 파티션에 데이터를 포함하지 않습니다. 여기서 유일한 예외는 테이블에 접속 해제된 종속 테이블이 있는 경우 파티션된 인덱스 작성이 접속 해제된 종속 테이블이 있는 파티션된 테이블에서 지원되지 않는다는 것입니다. 이 상황에서 파티션된 인덱스를 작성하려고 시도하면 SQL 상태가 55019로 됩니다.

파티션된 인덱스를 재작성하면 접속 또는 접속 해제된 데이터 파티션에 인덱스 파티션이 포함될 수 있습니다. 접속된 데이터 파티션의 경우, 접속된 파티션 데이터가 데이터베이스의 대부분의 활동에서 보이지 않기 때문에 접속된 데이터 파티션에서 유효하지 않은 인덱스 파티션의 재작성 비용이 대부분의 사용자에게 영향을 주지 않습니다. SET INTEGRITY는 접속된 데이터 파티션의 인덱스 파티션이 액세스되는 경우가 가장 일반적인 예이므로, SET INTEGRITY는 접속된 데이터 파티션에서 유효하지 않은 인덱스 파티션의 재작성을 시작합니다. 접속 해제된 데이터 파티션에 관하여 증분적으로 유지보수해야 하는 종속 테이블이 있는 경우에만 접속 해제된 데이터 파티션에 대해 유효하지 않은 인덱스 파티션의 재작성이 발생할 수 있습니다.

각 인덱스 파티션을 인식하여 파티션된 인덱스 재구성을 수행합니다. 접속 또는 접속 해제된 데이터 파티션의 인덱스 파티션은 인덱스 재구성 연산에 포함되지 않습니다.

WITH EMPTY TABLE

접속된 데이터 파티션이 있는 테이블을 비울 수 없습니다.

ADD MATERIALIZED QUERY AS

MQT에 접속된 데이터 파티션이 있는 테이블은 변경할 수 없습니다.

데이터 파티션에 저장된 추가 테이블 속성 변경

다음 테이블 속성은 데이터 파티션에도 저장됩니다. 이러한 속성을 변경하면 접속된 데이터 파티션에는 반영되나 접속 해제된 데이터 파티션에는 반영되지 않습니다.

- DATA CAPTURE
- VALUE COMPRESSION
- APPEND
- COMPACT/LOGGED FOR LOB COLUMNS
- ACTIVATE NOT LOGGED INITIALLY (WITH EMPTY TABLE)

파티션을 추가(ADD), 접속(ATTACH) 또는 접속 해제(DETACH)하기 위해 테이블 변경 시 XML 인덱스에 대한 특수 고려사항

XML 영역 인덱스 및 컬럼 경로 인덱스는 테이블 변경 시 파티션을 추가, 접속 또는 접속 해제하여 적용됩니다. 시스템이 생성한 XML 컬럼 경로 인덱스는 항상 파티션되지 않습니다. 파티션되지 않은 관계형 인덱스와 마찬가지로 XML 컬럼에서 파티션되지 않은 인덱스는 파티션된 테이블의 모든 데이터 파티션 사이에서 공유되는 독립 오브젝트입니다.

XML 영역 인덱스

ADD PARTITION은 추가되는 새로운 빈 데이터 파티션에 대해 새 영역 인덱스 파티션을 작성합니다. 영역 인덱스 파티션의 새 항목이 SYSINDEXPARTITIONS 테이블에 추가됩니다. 새 파티션에서 파티션된 인덱스 오브젝트의 테이블 스페이스는 ADD PARTITION 절에서 INDEX IN <table space>로 판별됩니다. ADD PARTITION 절에 대한 INDEX IN <table space>가 지정되지 않은 경우, 파티션된 인덱스 오브젝트의 테이블 스페이스는 디폴트로 대응하는 데이터 파티션에 사용된 테이블 스페이스와 동일합니다.

파티션된 테이블에서 시스템이 생성한 XML 영역 인덱스는 항상 파티션되어 있습니다. 파티션된 인덱스는 테이블의 테이블 파티셔닝 스킴에 따라 인덱스 데이터가 인덱스 파티션이라는 다중 스토리지 오브젝트로 접속 해제되는 인덱스 소속 스킴을 사용합니다. 각 인덱스 파티션은 해당 데이터 파티션의 테이블 행만 참조합니다.

ATTACH의 경우, XML 컬럼으로 파티션된 테이블의 영역 인덱스는 항상 파티션되므로 소스 테이블의 영역 인덱스는 ATTACH 조작 완료 후 새 테이블 파티션의 새 영역 인덱스 파티션으로 보존됩니다. 데이터 및 인덱스 오브젝트는 이동하지 않으므로 카탈로그 테이블 항목을 갱신해야 합니다. 소스 테이블의 영역 인덱스에 대한 카탈로그 테이블 항목은 ATTACH에 의해 제거되고 하나의 영역 인덱스 파티션이

SYSINDEXPARTITIONS 테이블에 추가됩니다. 풀 ID 및 오브젝트 ID는 소스 테이블에서와 동일한 상태로 남아 있습니다. 인덱스 ID(IID)는 목표의 영역 인덱스의 ID와 일치하도록 수정됩니다.

DETACH 조작 완료 후 영역 인덱스는 접속 해제된 테이블에서 보존됩니다. 접속 해제되는 파티션과 연관된 인덱스 파티션 항목은 SYSINDEXPARTITIONS 테이블에서 제거됩니다. 하나의 새 영역 인덱스 항목은 접속 해제된 테이블의 SYSINDEXES 카탈로그 테이블에 추가되며, 이 테이블은 DETACH 전의 영역 인덱스 파티션과 동일한 풀 ID와 오브젝트 ID를 갖습니다.

XML 데이터에 대한 인덱스

XML 데이터에 대한 파티션되지 않은 인덱스는 ATTACH 및 DETACH 조작 중에 파티션되지 않은 관계형 인덱스처럼 처리됩니다.

소스 테이블의 인덱스는 ATTACH 조작 중에 삭제됩니다. 이는 논리적 및 실제 XML 인덱스에 모두 적용됩니다. 시스템 카탈로그의 항목은 ATTACH 조작 중에 제거됩니다.

무결성 설정은 목표 테이블에서 XML 데이터에 대한 파티션되지 않은 인덱스를 유지보수하기 위해 ATTACH 이후에 실행해야 합니다.

DETACH의 경우, 소스 테이블의 XML 컬럼에 대한 파티션되지 않은 인덱스는 목표 테이블에서 상속하지 않습니다.

XML 컬럼 경로 인덱스

소스 및 목표 테이블의 XML 컬럼 경로 인덱스는 롤인 및 롤아웃 연산 중에 유지보수됩니다.

ATTACH의 경우, DB2 데이터베이스 관리 프로그램은 목표 테이블에서 파티션되지 않은 XML 컬럼 경로 인덱스를 유지보수합니다(이것은 기타 파티션되지 않은 인덱스와는 다르게 ATTACH 조작 완료 후 SET INTEGRITY 중에 유지보수됩니다). 그 후 소스 테이블의 XML 컬럼 경로 인덱스가 삭제되고 해당 카탈로그 항목은 목표 테이블의 컬럼 경로 인덱스가 파티션되지 않아 제거됩니다.

롤아웃의 경우, XML 컬럼 경로 인덱스가 파티션되지 않았고 파티션되지 않은 인덱스가 독립형 목표 테이블로 이전되지 않았음을 상기하십시오. 그러나 테이블이 외부 사용자에게 액세스 가능하도록 하려면 XML 컬럼 경로 인덱스(각 컬럼에 하나)가 XML 컬럼이 있는 테이블에 존재해야 하므로, 목표 테이블을 사용하기 전에 먼저 목표 테이블에서 XML 컬럼 경로 인덱스를 작성해야 합니다. 컬럼 경로 인덱스가 작성되는 시간은 DETACH 연산 중에 접속 해제된 종속 테이블이 있는지 여부에 따라 다릅니다. 접속

해제된 종속 테이블이 없는 경우 DETACH 연산 중에 경로 인덱스가 작성됩니다. 그렇지 않으면, 분리 종속 오브젝트를 유지보수하기 위해 SET INTEGRITY 또는 MQT 새로 고침으로 경로 인덱스가 작성됩니다.

DETACH 후 목표 테이블에 작성된 XML 컬럼 경로 인덱스는 해당 테이블에서 다른 모든 인덱스와 함께 같은 인덱스 오브젝트에 상주합니다.

데이터 파티션 접속

테이블 파티션을 사용하면 테이블 데이터를 효율적으로 롤인 및 롤아웃할 수 있습니다. ATTACH PARTITION 절이 있는 ALTER TABLE문을 사용하여 데이터 롤인을 매우 쉽게 수행할 수 있습니다. 이 절은 기존 테이블(소스 테이블)을 가져와 목표 테이블에 새 데이터 파티션으로 접속시킵니다. 새로 접속한 데이터 파티션은 접속 명령문이 완료된 후 처음에는 쿼리에 사용 불가능하지만 테이블의 나머지는 접속 후 온라인으로 남아 있습니다. SET INTEGRITY는 파티션을 온라인으로 가져오기 전에 파티션되지 않은 인덱스의 범위 검사, 제한 검사 및 유지보수를 수행하기 위해 필요합니다.

데이터 파티션을 접속할 수 있도록 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권이 소스 테이블에 대해 최소한 다음과 같은 권한 또는 특권 중 하나를 포함해야 합니다.

- 소스 테이블에 대한 DATAACCESS 권한 또는 SELECT 특권 및 소스 테이블의 스키마에 대한 DBADM 권한 또는 DROPIN 특권
- 소스 테이블에 대한 CONTROL 특권

ATTACH PARTITION절은 기존의 테이블(소스 테이블)을 가져와 목표 테이블에 새 데이터 파티션으로 접속시킵니다. 처음에는 쿼리 시 새로 접속된 데이터 파티션에 액세스할 수 없습니다. 테이블의 나머지 부분은 온라인으로 유지됩니다. 접속된 데이터 파티션을 온라인으로 만들려면 SET INTEGRITY문을 호출해야 합니다.

제한사항 및 사용법 지침

데이터 파티션을 접속하려면 다음 조건을 만족해야 합니다.

- 새 데이터 파티션을 접속할 테이블(즉 목표 테이블)은 기존의 파티션된 테이블이어야 합니다.
- 소스 테이블은 기존의 파티션되지 않은 테이블이거나 하나의 데이터 파티션만 있고 접속(ATTACHED) 또는 접속 해제(DETACHED)된 데이터 파티션이 없는 파티션된 테이블이어야 합니다. 복수의 데이터 파티션을 접속하려면 복수의 ATTACH문을 발행해야 합니다.
- 소스 테이블이 계층 구조를 가져서는 안 됩니다(유형이 지정된 테이블).
- 소스 테이블이 범위 클러스터 테이블(RCT)이어서는 안 됩니다.
- 소스 테이블과 목표 테이블의 테이블 정의가 일치해야 합니다.

- 소스 및 목표 테이블의 컬럼 수, 유형 및 순서가 일치해야 합니다.
- 두 테이블의 경우, 디폴트값 포함 여부에 관계없이 컬럼이 일치해야 합니다. ALTER TABLE ADD COLUMN을 사용하여 소스 컬럼을 작성할 경우 (즉 SYSCOLUMNS.ADD_DEFAULT = 'Y'임) existDefault 값 (SYSCOLUMNS.ADDED_DEFAULT)은 목표 컬럼의 existDefault 값과 동일해야 합니다.
- 두 테이블의 경우, 컬럼은 널(NULL) 허용 여부에 관계없이 일치해야 합니다.
- 소스 및 목표 테이블에서 VALUE COMPRESSION 및 SYSTEM COMPRESSION DEFAULT 값을 비롯한 압축 절이 일치해야 합니다.
- 데이터 캡처 옵션과 함께 APPEND절을 사용하고 로그되지 않은 초기 옵션도 일치해야 합니다.
- 목표 컬럼은 생성된 컬럼이나 소스 컬럼은 생성된 컬럼이 아니어도 데이터 파티션을 접속할 수 있습니다. 이 SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FORCE GENERATED문은 접속된 행의 생성된 컬럼에 대한 값을 생성합니다. 생성된 컬럼과 일치하는 컬럼은 유형 및 널 가능성이 일치해야 합니다. 이 컬럼에 대한 디폴트값은 필요하지 않습니다. 권장되는 접속은 ATTACH라는 소스 테이블이 생성된 컬럼에 올바른 생성 값을 가지는 것입니다. 그러면 FORCE GENERATED 옵션을 사용하지 않아도 됩니다. 다음 명령문을 사용할 수 있습니다.

  ```
  SET INTEGRITY FOR T GENERATED COLUMN IMMEDIATE UNCHECKED
  (생성된 컬럼의 점검을 생략하도록 시스템에게 지시함)
  SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION
  IN T USE T_EX (접속된 파티션에 대한 무결성 점검은 수행하나
  생성된 컬럼이 올바른지는 점검하지 않음)
  ```

- 목표 컬럼은 식별 컬럼이고 소스 컬럼은 식별 컬럼이 아니어도 데이터 파티션을 접속할 수 있습니다. SET INTEGRITY IMMEDIATE CHECKED문은 접속된 행에 대한 식별 값을 생성하지 않습니다. SET INTEGRITY FOR T GENERATE IDENTITY ALLOW WRITE ACCESS IMMEDIATE CHECKED문은 접속된 행의 식별 값을 채웁니다. 식별 컬럼과 일치하는 컬럼은 유형 및 널 가능성도 일치해야 합니다. 이 컬럼에 대한 디폴트값은 필요하지 않습니다. 권장되는 방법은 스테이징 테이블에서 올바른 식별 값을 입력하는 것입니다. 그러면 소스 테이블에서 식별 값이 이미 보장되기 때문에 접속(ATTACH) 이후에 GENERATE IDENTITY 옵션을 사용할 필요가 없습니다.
- 데이터가 데이터베이스 파티션에 분산되어 있는 테이블의 경우, 소스 테이블도 동일한 분산 키 및 동일한 분산 맵을 사용하여 동일한 데이터베이스 파티션 그룹에 분산되어야 합니다.
- 소스 테이블은 삭제 가능해야 합니다(즉 RESTRICT DROP 세트가 있어서는 안 됨).
- DATAPARTITIONNAME을 지정할 경우 목표 테이블에 이미 존재해서는 안 됩니다.

- 목표 테이블이 다차원적으로 클러스터된(MDC) 테이블인 경우, 소스 테이블도 MDC 테이블이어야 합니다.
- 파티션되지 않은 테이블을 사용하여 소스 테이블의 데이터 테이블 스페이스와 목표 테이블의 데이터 테이블 스페이스의 유형(즉 DMS 또는 SMS), 페이지 크기, Extent 크기 및 데이터베이스 파티션 그룹이 일치해야 합니다. 프리페치 크기가 일치하지 않으면 경고가 리턴됩니다. 소스 테이블의 인덱스 테이블 스페이스는 목표 테이블의 파티션된 인덱스에 사용된 인덱스 테이블 스페이스와 유형, 페이지 크기, Extent 크기 및 데이터베이스 파티션 그룹이 일치해야 합니다. 소스 테이블의 대형 테이블 스페이스와 목표 테이블의 대형 테이블 스페이스는 유형, 데이터베이스 파티션 그룹 및 페이지 크기가 일치해야 합니다. 파티션된 테이블 사용 시 소스 테이블의 데이터 테이블 스페이스는 목표 테이블의 데이터 테이블 스페이스와 유형, 페이지 크기, Extent 크기 및 데이터베이스 파티션 그룹이 일치해야 합니다.
- 구조화된 컬럼이나 XML 또는 LOB 컬럼이 있는 파티션된 테이블에 ALTER TABLE ATTACH문을 발행하는 경우, 소스 테이블의 구조화된 컬럼이나 XML 또는 LOB 컬럼의 INLINE LENGTH는 목표 테이블에서 이에 대응하는 구조화된 컬럼이나 XML 또는 LOB 컬럼의 INLINE LENGTH와 일치해야 합니다.
- REQUIRE MATCHING INDEXES 절이 ATTACH PARTITION절과 함께 사용될 때 목표 테이블에서 파티션된 모든 인덱스가 소스 테이블의 인덱스와 일치하지 않는 경우 SQL20307N이 리턴됩니다.
- 목표에서 파티션된 각 고유 인덱스에 대해 일치하는 인덱스가 없는 소스 테이블을 접속하면 오류 SQL20054N과 함께 접속 실패를 야기합니다.
- 테이블에 지연된 인덱스 정리 연산이 MDC 롤아웃의 결과로 진행 중에 있을 때 파티션된 인덱스에 대해서는 지연된 인덱스 정리 메커니즘을 사용하는 MDC 롤아웃이 지원되지 않기 때문에 접속 연산 중에 보존되어 재빌드되지 않고 롤아웃된 블록의 비동기 인덱스 정리가 보류 중인 RID 인덱스가 소스 테이블에 있는 경우 접속 연산이 허용되지 않습니다.
- 목표 테이블의 데이터 형식과는 다른 XML 데이터 형식을 사용한 소스 테이블 접속은 지원되지 않습니다.
- 테이블에 버전 9.5 또는 이전 XML 레코드 형식을 사용하는 XML 컬럼이 들어 있는 경우, 버전 9.7 또는 이후 레코드 형식을 사용하는 XML 컬럼을 포함하는 파티션된 테이블에 해당 테이블을 접속하는 기능이 지원되지 않습니다.

  테이블을 접속하기 위해서는 먼저 목표 파티션된 테이블의 레코드 형식과 일치하도록 테이블의 XML 레코드 형식을 갱신해야 합니다. 다음 두 메소드 중 어느 하나는 테이블의 XML 레코드 형식을 갱신합니다.

  – ADMIN_MOVE_TABLE 프로시저를 사용하여 테이블에서 온라인 테이블 이동을 수행하십시오.
  – 다음 단계를 수행하십시오.

1. EXPORT 명령을 사용하여 테이블 데이터 복사를 작성하십시오.
2. TRUNCATE문을 사용하여 테이블의 모든 행을 삭제하고 테이블에 할당된 스토리지를 해제하십시오.
3. LOAD 명령을 사용하여 데이터를 테이블에 추가하십시오.

테이블의 XML 레코드 형식이 갱신된 후 테이블을 목표 파티션된 테이블에 접속하십시오.

접속 연산을 실행하기 전에 목표 테이블의 각 파티션된 인덱스와 일치하는 소스 테이블의 인덱스를 작성하십시오. 파티션된 인덱스가 일치하면 롤인 연산의 효율성이 더 증가하고 필요한 활성 로그 스페이스가 더 감소합니다. 소스 테이블의 인덱스가 적절히 준비되지 않은 경우 해당 인덱스를 유지보수하기 위해 데이터베이스 관리 프로그램이 필요합니다. 롤인이 파티션된 인덱스를 유지보수하는 데 추가적인 비용을 초래하지 않도록 하기 위해 접속 파티션 연산 시 REQUIRE MATCHING INDEXES를 지정할 수 있습니다. 목표에서 파티션된 인덱스와 일치하는 인덱스가 소스 테이블에 없는 경우에는 접속 연산이 실패합니다. 이 경우에는 정정 조치를 수행하고 접속 연산을 재실행할 수 있습니다.

또한 접속 연산을 실행하기 전에 소스 테이블의 모든 추가 인덱스를 삭제하십시오. 추가 인덱스는 목표 테이블에 일치하는 인덱스가 없거나 목표 테이블에서 파티션되지 않은 인덱스와 일치하는 소스 테이블의 인덱스입니다. 접속 연산을 실행하기 전에 추가 인덱스를 삭제하면 연산 실행 속도가 더 빨라집니다. 그렇지 않으면, 추가 인덱스가 연산 속도를 늦추므로 ATTACH PARTITION 연산 중에 DB2 시스템에서 추가 인덱스를 삭제해야 합니다.

예를 들어, 12개의 데이터 파티션(해당 연도의 각 월에 하나씩)을 가지고 있는 『orders』라는 파티션된 테이블이 존재한다고 가정합니다. 독립 테이블 『neworders』도 또한 존재합니다. 각 월말에 파티션된 『orders』 테이블에 접속됩니다.

1. 『orders』 테이블에서 파티션된 인덱스를 작성하십시오.

   ```
   CREATE INDEX idx_delivery_date ON orders(delivery) PARTITIONED;
   CREATE INDEX idx_order_price ON orders(price) PARTITIONED;
   ```

2. 『neworders』 테이블에서 대응하는 인덱스를 작성하여 접속 연산을 준비하십시오.

   ```
   CREATE INDEX idx_delivery_date_for_attach ON neworders(delivery);
   CREATE INDEX idx_order_price_for_attach ON neworders(price);
   ```

3. 접속 연산에는 두 개의 단계가 있습니다.
   a. ATTACH. 『orders』 테이블에서 파티션된 인덱스와 일치하는 『neworders』 테이블의 인덱스는 보존됩니다.

      ```
      ALTER TABLE orders ATTACH PARTITION part_jan2009 STARTING FROM ('01/01/2009')
         ENDING AT ('01/31/2009') FROM TABLE neworders;
      ```

『orders』 테이블은 Set Integrity Pending(무결성 설정 보류) 상태로 자동 배치됩니다. idx_delivery_date_for_attach 및 idx_order_price_for _attach 둘 다 접속 조작 완료 후 『orders』 테이블의 파트가 됩니다. 이 연산 중에는 데이터 이동이 발생하지 않습니다.

b. SET INTEGRITY. 새로 접속된 파티션에서 범위 점검이 완료되었습니다. 존재하는 모든 제한조건이 강제 실행됩니다.

```
SET INTEGRITY FOR orders IMMEDIATE CHECKED;
```

목표 테이블에 파티션되지 않은 인덱스가 존재하는 경우, SET INTEGRITY는 인덱스 유지보수 작업을 기타 태스크(예: 새로 접속된 파티션의 데이터에 대한 범위 유효성 확인 및 제한조건 점검)와 함께 수행해야 합니다. 파티션되지 않은 인덱스 유지보수에는 새로 접속된 파티션의 데이터 볼륨, 각 파티션되지 않은 인덱스의 키 크기 및 파티션되지 않은 인덱스 수에 비례하는 대량의 활성 로그 스페이스가 필요합니다.

소스 테이블이 파티션되지 않은 경우, 새 데이터 파티션의 각 파티션된 인덱스에는 소스 테이블의 테이블 스페이스 ID 및 오브젝트 ID를 사용하여 SYSINDEXPARTITIONS에 새 항목이 제공됩니다. ID 정보는 SYSINDEXES(테이블이 파티션되지 않은 경우) 또는 SYSINDEXPARTITIONS (테이블이 파티션된 경우)에서 가져옵니다. 인덱스 ID는 일치하는 목표 테이블 파티션 및 인덱스에서 가져옵니다.

소스 테이블이 파티션된 경우, 목표 테이블의 파티션된 인덱스와 일치하는 소스 테이블의 파티션된 인덱스는 접속 연산의 파트로 보존됩니다. SYSINDEXPARTITIONS 인덱스 파티션 항목이 갱신되어 새 색인 ID로 새 목표 테이블의 인덱스 파티션임을 표시합니다.

데이터 파티션 접속 시 소스 테이블에서 새 파티션의 목표 테이블로 데이터 및 인덱스에 대한 일부 통계가 이전됩니다. 특히 목표의 새 파티션에 대한 SYSDATAPARTITIONS 및 SYSINDEXPARTITIONS의 모든 필드에는 소스의 데이터가 채워집니다. 소스 테이블이 파티션되지 않은 경우, 이 통계는 SYSTABLES 및 SYSINDEXES에서 옵니다. 소스 테이블이 단일 파티션으로 파티션된 테이블인 경우에는 이 통계가 단일 소스 파티션의 SYSDATAPARTITIONS 및 SYSINDEXPARTITIONS에서 옵니다.

주: 이전된 통계는 SYSINDEXES 및 SYSTABLES의 통계 집계에 영향을 주지 않기 때문에 ATTACH 조작 완료 후 RUNSTATS를 실행해야 합니다.

파티션된 테이블에 데이터 파티션을 접속하기 위한 지침

이 주제는 ALTER TABLE ...ATTACH PARTITION문을 발행하여 데이터 파티션을 파티션된 테이블에 첨부하려고 할 때 발생할 수 있는 여러 불일치 유형을 수정하는 지침을 제공합니다. 소스 테이블을 수정하여 목표 테이블의 특성을 일치시키거나, 목표 테이블을 수정하여 소스 테이블의 특성을 일치시켜 테이블 간의 계약을 얻을 수 있습니다.

소스 테이블은 목표 테이블에 첨부할 기존 테이블입니다. 목표 테이블은 새 데이터 파티션을 첨부할 테이블입니다.

권장되는 첨부 수행 방법은 목표 테이블에 사용한 것과 동일한 CREATE TABLE문을 PARTITION BY절없이 소스 테이블에 사용하는 것입니다. 호환성에 대한 소스 또는 목표 테이블의 특성을 수정하기가 어려울 경우, 목표 테이블에 호환 가능한 새 소스 테이블을 작성할 수 있습니다. 새 소스 작성에 대한 세부사항은 *기존 테이블과 같은 테이블 작성*을 참조하십시오.

불일치 발생을 방지하려면 234 페이지의 『데이터 파티션 접속』 절의 제한사항 및 사용 지침을 참조하십시오. 해당 절에서는 데이터 파티션을 첨부하기 위해 만족시켜야 할 개요 조건을 설명합니다. 목록에 있는 조건을 만족시키지 못하면 SQL20408N 또는 SQL20307N 오류가 리턴됩니다.

다음 절에서는 발생할 수 있는 불일치의 여러 유형을 설명하고 테이블 간의 계약을 처리하기 위한 권장되는 단계를 제공합니다.

(값) 압축 절(SYSCAT.TABLES의 COMPRESSION 컬럼)이 일치하지 않습니다. (SQL20307N 이유 코드 2)

값 압축 계약을 처리하려면 다음 명령문 중 하나를 사용하십시오.

```
ALTER TABLE... ACTIVATE VALUE COMPRESSION
 또는
ALTER TABLE... DEACTIVATE VALUE COMPRESSION
```

행 압축 계약을 처리하려면 다음 명령문 중 하나를 사용하십시오.

```
ALTER TABLE... COMPRESS YES
 또는
ALTER TABLE... COMPRESS NO
```

테이블의 APPEND 모드가 일치하지 않습니다. (SQL20307N 이유 코드 3)

APPEND 모드 계약을 처리하려면 다음 명령문 중 하나를 사용하십시오.

```
ALTER TABLE ... APPEND ON
 또는
ALTER TABLE ... APPEND OFF
```

소스 및 목표 테이블의 코드 페이지가 일치하지 않습니다. (SQL20307N 이유 코드 4)

새 소스를 작성하십시오.

소스 테이블이 둘 이상의 데이터 파티션 또는 접속되거나 접속 해제된 데이터 파티션이 있는 파티션된 테이블입니다. (SQL20307N 이유 코드 5)

단일 데이터 파티션이 표시될 때까지 다음 명령을 사용하여 데이터 파티션을 소스 테이블로부터 접속 해제하십시오.

```
ALTER TABLE ... DETACH PARTITION
```

필요한 SET INTEGRITY 명령문을 포함하십시오. 소스 테이블에 인덱스가 있을 경우 해당 소스 테이블을 즉시 첨부할 수 없습니다. 접속 해제된 데이터 파티션은 모든 인덱스가 접속 해제된 키를 정리할 때까지 접속 해제되어 있습니다. 즉시 첨부하려면 소스 테이블에서 인덱스를 삭제하십시오. 또는 새 소스를 작성하십시오.

소스 테이블은 시스템 테이블, 뷰, 입력된 테이블, ORGANIZED BY KEY SEQUENCE 테이블, 작성된 임시 테이블 또는 선언된 임시 테이블입니다. (SQL20307N 이유 코드 6)

새 소스를 작성하십시오.

목표 및 소스 테이블이 동일합니다. (SQL20307N 이유 코드 7)

테이블 자체에 테이블을 추가할 수 없습니다. 소스 또는 목표 테이블로 사용할 올바른 테이블을 판별하십시오.

NOT LOGGED INITIALLY절을 소스 테이블과 목표 테이블 모두에 지정하지 않고 소스 테이블 또는 목표 테이블 중 하나에만 지정했습니다. (SQL20307N 이유 코드 8)

처음에 로그되지 않은 테이블을 COMMIT문을 발행하여 로그하거나 로그된 테이블을 다음 명령문을 입력하여 로그 해제하십시오.

```
ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY
```

DATA CAPTURE CHANGES절이 소스 테이블과 목표 테이블 모두에 지정되지 않고 소스 테이블 또는 목표 테이블 중 하나에만 지정되었습니다. (SQL20307N 이유 코드 9)

데이터 캡처 변경이 설정되지 않은 테이블에서 데이터 캡처를 사용 가능하게 하려면 다음 명령문을 실행하십시오.

```
ALTER TABLE ... DATA CAPTURE CHANGES
```

데이터 캡처 변경이 설정된 테이블에서 데이터 캡처 변경을 사용 불가능하게 하려면 다음 명령문을 실행하십시오.

```
ALTER TABLE ... DATA CAPTURE NONE
```

테이블의 분산 절이 일치하지 않습니다. 분산 키는 소스 테이블 및 목표 테이블에서 동일해야 합니다. (SQL20307N 이유 코드 10)

새 소스 테이블을 작성할 것을 권장합니다. 여러 데이터베이스 파티션에 있는 분산 키를 변경할 수 없습니다. 단일 파티션 데이터베이스에서 테이블의 분산 키를 변경하려면 다음 명령문을 실행하십시오.

```
ALTER TABLE ... DROP DISTRIBUTION;
    ALTER TABLE ... ADD DISTRIBUTION(key-specification)
```

접속 연산 중에 누락된 인덱스가 있는 경우에 오류가 리턴됩니다((SQL20307N 이유 코드 18).

접속 연산은 목표 테이블에서 파티션된 인덱스에 대응하는 소스 테이블의 누락된 인덱스를 내재적으로 빌드합니다. 누락된 인덱스의 내재된 작성이 완료되기까지 시간이 걸립니다. 접속 연산이 누락된 인덱스에 직면하는 경우에 작성할 옵션 및 오류 조건이 있습니다. 옵션은 ERROR ON MISSING INDEXES라고 하며 접속 연산 옵션 중 하나입니다. 이 경우가 발생할 때 리턴된 오류는 SQL20307N, SQLSTATE 428GE, 이유 코드 18입니다. 불일치한 인덱스에 대한 정보는 관리 로그에 배치됩니다.

접속 연산은 목표 테이블에서 파티션된 인덱스와 일치하지 않는 소스 테이블의 인덱스를 삭제합니다. 이 불일치 인덱스의 식별 및 삭제는 완료되기까지 시간이 걸립니다. 접속 연산을 시도하기 전에 먼저 이 인덱스를 삭제해야 합니다.

목표 테이블의 불일치 인덱스가 고유 인덱스이거나 XML 인덱스가 접속 연산 중에 REJECT INVALID VALUES절로 정의되어 있는 경우 오류가 리턴됩니다 (SQL20307N 이유 코드 17).

소스 테이블에 일치하는 인덱스가 없는 목표 테이블에 파티션된 인덱스가 있고 ERROR ON MISSING INDEXES가 사용되지 않은 경우, 다음 결과를 기대할 수 있습니다.

1. 목표 테이블의 불일치 인덱스가 고유 인덱스이거나 XML 인덱스가 REJECT INVALID VALUES 절로 정의되어 있는 경우 접속 연산이 실패하고 오류 메시지 SQL20307N, SQLSTATE 428GE, 이유 코드 17을 리턴합니다.
2. 목표 테이블의 불일치 인덱스가 이전 위치에서 조건을 충족하지 않는 경우, 소스 테이블의 인덱스 오브젝트가 접속 연산 중에 유효하지 않음으로 표시됩니다. 접속 연산이 성공적으로 완료되지만 새 데이터 파티션의 인덱스 오브젝트는 유효하지 않음으로 표시됩니다. SET INTEGRITY 연산은 새로 접속된 파티션에서 인덱스 오브젝트를 재빌드하는 데 사용됩니다. 일반적으로 이것은 데이터 파티션 접속 후에 수행하는 다음 연산입니다. 인덱스 재작성은 시간이 걸립니다.

관리 로그는 소스 및 목표 테이블의 인덱스 간에 모든 불일치에 대한 세부사항을 포함합니다.

테이블 중 하나에만 ORGANIZE BY DIMENSIONS절이 지정되었거나 소속 차원이 다릅니다. (SQL20307N 이유 코드 11)

새 소스를 작성하십시오.

컬럼(TYPENAME)의 데이터 유형이 일치하지 않습니다. (SQL20408N 이유 코드 1)

데이터 유형의 불일치를 수정하려면 다음 명령문을 발행하십시오.

```
ALTER TABLE ... ALTER COLUMN ... SET DATA TYPE...
```

컬럼(NULLS)의 널 가능성이 일치하지 않습니다. (SQL20408N 이유 코드 2)

여러 테이블 중의 한 테이블에서 일치하지 않는 컬럼의 널 가능성을 변경하려면 다음 명령문을 발행하십시오.

```
ALTER TABLE... ALTER COLUMN... DROP NOT NULL
 또는
ALTER TABLE... ALTER COLUMN... SET NOT NULL
```

컬럼의 내재된 디폴트값(SYSCAT.COLUMNS IMPLICITVALUE)이 호환되지 않습니다. (SQL20408N 이유 코드 3)

새 소스 테이블을 작성하십시오. 목표 테이블 컬럼과 소스 테이블 컬럼이 둘 다 내재된 디폴트값(IMPLICITVALUE가 NULL이 아닌 경우)을 가질 경우 내재된 디폴트값이 정확히 일치해야 합니다.

IMPLICITVALUE가 목표 테이블의 컬럼에 대해 널(NULL)이 아니고 IMPLICITVALUE가 소스 테이블의 상응하는 컬럼에 대해 널(NULL)이 아닌 경우, 테이블에 대한 원래 CREATE TABLE문 다음에 각 컬럼이 추가됩니다. 이 경우, 이 컬럼에 대해 IMPLICITVALUE에 저장된 값이 일치해야 합니다.

버전 9.1 이전 테이블로부터의 이주를 통해 또는 버전 9.1 이전 테이블의 데이터 파티션 접속을 통해 IMPLICITVALUE가 널(NULL)이 아닌 경우도 있습니다. 그 이유는 원래 CREATE TABLE문 다음에 컬럼이 추가되었는지 여부를 시스템이 알 지 못하기 때문입니다. 데이터베이스가 컬럼이 추가되는지 여부를 확실히 알 수 없는 경우 추가된 것으로 처리됩니다. 추가된 컬럼은 ALTER TABLE ...ADD COLUMN문을 실행할 때 작성되는 컬럼입니다. 이 경우, 접속을 진행하면 컬럼 값이 손상될 수 있으므로 명령문이 허용되지 않습니다. 소스 테이블에서 새 테이블로 데이터를 복사하고(NULL인 이 컬럼에 IMPLICITVALUE 사용) 새 테이블을 접속 조치의 소스 테이블로 사용해야 합니다.

컬럼의 코드 페이지(COMPOSITE_CODEPAGE)가 일치하지 않습니다. (SQL20408N 이유 코드 4)

새 소스 테이블을 작성하십시오.

시스템 압축 디폴트 절(COMPRESS)이 일치하지 않습니다. (SQL20408N 이유 코드 5)

컬럼의 시스템 압축을 변경하려면 다음 명령문 중 하나를 발행하여 불일치를 수정하십시오.

```
ALTER TABLE ... ALTER COLUMN ...  COMPRESS SYSTEM DEFAULT
 또는
ALTER TABLE ... ALTER COLUMN ... COMPRESS OFF
```

ATTACH PARTITION 중에 목표 테이블의 파티션된 인덱스와 소스 테이블 인덱스를 일치시키는 조건

목표 테이블에서 파티션된 인덱스의 모든 인덱스 키 컬럼은 소스 테이블에서 인덱스의 인덱스 키 컬럼과 일치해야 합니다. 인덱스의 다른 모든 인덱스 등록 정보가 동일한 경우, 소스 테이블의 인덱스가 목표 테이블에서 파티션된 인덱스에 대한 일치사항으로 간주됩니다. 즉, 소스 테이블의 인덱스는 목표 테이블의 인덱스로 사용될 수 있습니다. 여기서 테이블은 인덱스가 일치사항으로 간주되는지 여부를 판별하는 데 사용될 수 있습니다.

아래 테이블은 목표 인덱스가 파티션된 경우에만 유용하고 적용 가능합니다. 목표 인덱스 등록 정보는 일치사항으로 간주되는 모든 경우에 소스 인덱스에서 가정됩니다.

표 14. 목표 인덱스 등록 정보가 소스 인덱스 등록 정보와 다른 경우 소스 인덱스의 일치 여부 판별.

규칙 번호	목표 인덱스 등록 정보	소스 인덱스 등록 정보	소스 인덱스가 일치합니까?
1.	고유하지 않음	고유	예
2.	고유	고유하지 않음	아니오
3.	컬럼 X는 내림차순임	컬럼 X는 오름차순임	아니오
4.	컬럼 X는 오름차순임	컬럼 X는 내림차순임	아니오
5.	파티션됨	파티션되지 않음	아니오. 참고: 이는 소스 테이블이 파티션되었다고 가정합니다.
6.	pctfree n1	pctfree n2	예
7.	level2pctfree n1	level2pctfree n2	예
8.	minpctused n1	minpctused n2	예
9.	역 스캔 승인 불가	역 스캔 허용	예, 실제 인덱스 구조는 역방향 스캔이 허용되는지 여부에 관계없이 동일합니다.
10.	역 스캔 허용	역 스캔 승인 불가	예, (9)와 같은 이유.
11.	pagesplit [L|H|S]	pagesplit [L|H|S]	예
12.	샘플 통계	자세한 통계	예
13.	자세한 통계	샘플 통계	예
14.	클러스터되지 않음	CLUSTER	예

표 14. 목표 인덱스 등록 정보가 소스 인덱스 등록 정보와 다른 경우 소스 인덱스의 일치 여부 판별. (계속)

규칙 번호	목표 인덱스 등록 정보	소스 인덱스 등록 정보	소스 인덱스가 일치합니까?
15.	CLUSTER	클러스터되지 않음	예. 인덱스는 클러스터링 인덱스가 되지만 데이터가 재구성될 때까지 데이터는 이 인덱스에 따라 클러스터되지 않습니다. 접속 후 파티션 레벨 인식을 사용하여 이 인덱스 파티션에 따라 데이터를 클러스터할 수 있습니다.
16.	유효하지 않은 값 무시	유효하지 않은 값 거부	예
17.	유효하지 않은 값 거부	유효하지 않은 값 무시	아니오. 유효하지 않은 값 거부의 목표 인덱스 등록 정보를 고려해야 하며, 소스 테이블에는 이 인덱스 제한조건을 위반하는 행이 있을 수 있습니다.
18.	인덱스 압축이 사용 가능합니다.	인덱스 압축이 사용되지 않습니다.	예. 참고: 인덱스가 재빌드될 때까지 기본적인 인덱스 데이터의 압축이 발생하지 않습니다.
19.	인덱스 압축이 사용되지 않습니다.	인덱스 압축이 사용 가능합니다.	예. 참고: 인덱스가 재빌드될 때까지 인덱스 데이터 압축 해제가 발생하지 않습니다.

주: 규칙 번호 5에서, DB2 9.5 또는 이전 버전을 사용하여 작성된 다차원적으로 클러스터된(MDC) 테이블(파티션되지 않은 블록 인덱스 포함)을 DB2 9.7 또는 이후 버전을 사용하여 작성된 새 MDC 파티션된 테이블(파티션된 블록 인덱스 포함)에 접속하려는 경우 및 ERROR ON MISSING INDEXES절이 사용되는 경우 ALTER TABLE ... ATTACH PARTITON 문이 실패하고 오류 메시지 SQL20307N, SQLSTATE 428GE, 이유 코드 18을 리턴합니다. ERROR ON MISSING INDEXES절을 제거하면 데이터베이스 관리 프로그램이 접속 연산 중에 인덱스를 유지보수하므로 접속이 완료됩니다. 오류 메시지 SQL20307N, SQLSTATE 428GE, 이유 코드 18을 수신한 경우, ERROR ON MISSING INDEXES절 제거를 고려해야 합니다.

데이터 파티션 접속 해제

테이블 파티션을 사용하면 테이블 데이터를 효율적으로 롤인 및 롤아웃할 수 있습니다. ALTER TABLE문의 ATTACH PARTITION 및 DETACH PARTITION절을 사용하여 효율성을 높일 수 있습니다.

파티션된 테이블에서 데이터 파티션을 접속 해제하려면 다음과 같은 권한 또는 특권이 있어야 합니다.

- DETACH 조작을 수행하는 사용자는 소스 테이블에서 ALTER, SELECT 및 DELETE 조작을 수행하는데 필요한 권한을 갖고 있어야 합니다.

- 또한 목표 테이블을 작성하는데 필요한 권한도 갖고 있어야 합니다. 따라서, 데이터 파티션을 접속 해제하도록 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권이 목표 테이블에 대해 최소한 다음과 같은 권한 또는 특권 중 하나를 포함해야 합니다.
  - DBADM 권한
  - 다음 권한 중 하나를 비롯하여 테이블에 사용되는 테이블 스페이스에 대한 USE 특권 및 데이터베이스에 대한 CREATETAB 권한
    - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

주: 데이터 파티션을 접속 해제할 경우, 명령문의 권한 부여 ID가 효율적으로 CREATE TABLE문을 수행하기 위해서는 해당 조작을 수행하기 위한 필수 특권을 갖고 있어야 합니다. ALTER TABLE문의 권한 부여 ID는 마치 사용자가 CREATE TABLE문을 발행한 것처럼 CONTROL 권한을 가진 새 테이블의 정의자가 됩니다. 변경된 테이블의 특권은 새 테이블로 이동되지 않습니다. ALTER TABLE문의 권한 부여 ID 및 DBADM 또는 SYSADM만이 ALTER TABLE ... DETACH PARTITION문 실행 직후에 데이터에 액세스할 수 있습니다.

파티션된 테이블 데이터를 롤아웃하면 파티션된 테이블에서 데이터 범위를 쉽게 분리할 수 있습니다. 데이터 파티션을 별도의 테이블로 접속 해제한 경우, 여러 방법으로 테이블을 처리할 수 있습니다. 분리된 테이블을 삭제(이 조작을 수행하면 데이터 파티션의 데이터가 제거됨)하고, 이를 아카이브하거나 별도의 테이블로 사용할 수 있으며, 이를 실행기록 테이블과 같은 다른 파티션된 테이블에 접속시킬 수 있습니다. 또는 원본 또는 일반 다른 파티션된 테이블을 조작, 정리, 변환 및 재접속할 수 있습니다.

제한사항

소스 테이블이 다차원적으로 클러스터된 테이블(MDC)인 경우 ALTER TABLE ...DETACH 조작과 동일한 작업 단위(UOW)로 새로 접속 해제된 테이블에 액세스할 수 없습니다. MDC 테이블은 파티션된 블록 인덱스를 지원하지 않습니다. 이 경우, ALTER TABLE ...DETACH 조작이 커미트된 후 테이블에 처음으로 액세스할 때 블록 인덱스가 작성됩니다. 소스 테이블에 분리 시간 이전에 파티션된 다른 어떤 인덱스가 있는 경우, 목표 테이블의 인덱스 오브젝트는 블록 인덱스 작성을 고려하여 유효하지 않은 것으로 표시됩니다. 그 결과 블록 인덱스가 작성되는 동안 액세스 시간이 늘어나고 모든 파티션된 인덱스가 재작성됩니다.

DETACH 조작을 수행하려면 다음 조건을 만족시켜야 합니다.

- 접속 해제할 테이블(소스 테이블)이 존재하고 파티션된 테이블이어야 합니다.
- 접속 해제할 데이터 파티션은 소스 테이블에 존재해야 합니다.

- 소스 테이블은 둘 이상의 데이터 파티션을 갖고 있어야 합니다. 파티션된 테이블은 최소한 하나의 데이터 파티션을 갖고 있어야 합니다. 가시적이며 접속된 데이터 파티션만이 이 컨텍스트에 관련되어 있습니다. 접속된 데이터 파티션은 접속되어 있으나 아직 SET INTEGRITY문에 의해 유효성이 확인되지 않은 데이터 파티션입니다.
- DETACH 조작을 사용하여 작성할 테이블 이름(목표 테이블)이 존재하지 않아야 합니다.
- 강제 실행된 참조 무결성(RI) 관계에서 상위인 테이블에서는 DETACH 조작이 허용되지 않습니다.
- 접속 해제된 데이터 파티션(이 종속 테이블을 접속 해제된 종속 테이블이라고 함)과 관련하여 유지보수해야 하는 종속 테이블이 있는 경우, 처음에는 새로 접속 해제된 테이블에 액세스할 수 없습니다. 테이블의 SYSCAT.TABLES 카탈로그 뷰의 TYPE 컬럼에 L이라고 표시됩니다. 이를 접속 해제된 테이블이라고 합니다. 접속 해제되면, SET INTEGRITY문을 실행하여 증분 방식으로 접속 해제된 종속 테이블을 유지보수할 때까지 테이블을 읽거나 수정하거나 또는 삭제할 수 없습니다. 모든 접속 해제된 종속 테이블에서 SET INTEGRITY문을 실행하면, 접속 해제된 테이블은 완전히 액세스가 가능한 일반 테이블이 됩니다.

파티션된 테이블을 변경하고 테이블에서 데이터 파티션을 접속 해제하려면, 명령행에서 ALTER TABLE문을 DETACH PARTITION 절과 함께 발행하십시오.

접속 해제되는 데이터 파티션의 소스 테이블에 정의된 각 인덱스 파티션은 목표 테이블의 인덱스가 됩니다. 인덱스 오브젝트는 접속 해제 조작 중에 전혀 이동되지 않습니다. 그러나 접속 해제되는 테이블 파티션의 인덱스 파티션에 대한 메타데이터는 카탈로그 테이블 SYSINDEXPARTITIONS에서 제거되고 새 인덱스 항목은 접속 해제 조작의 결과로 새 테이블의 SYSINDEXES에 추가됩니다. 원래 인덱스 ID(IID)는 보존되고 소스 테이블에서와 같이 고유하게 유지됩니다.

목표 테이블에 잔존하는 인덱스의 인덱스 이름은 SQLyymmddhhmmssxxx 양식을 사용하여 시스템에서 생성합니다. 이 인덱스의 스키마는 SYSTEM 스키마에 있는 모든 경로 인덱스, 영역 인덱스 및 MDC 블록 인덱스를 제외하고 목표 테이블의 스키마와 동일합니다. 인덱스는 접속 해제된 테이블로 이전되지만 제한조건은 이전되지 않으므로 고유한 1차 키 제한조건을 강제 실행하는 인덱스와 같이 기타 시스템이 생성한 인덱스에는 목표 테이블의 스키마가 있습니다. RENAME 명령을 사용하여 SYSTEM 스키마에 없는 인덱스 이름을 바꿀 수 있습니다.

새로 접속 해제된 독립형 테이블에 대한 동일한 제한조건을 적용하려면 접속 해제 조작 완료 후 목표 테이블에서 ALTER TABLE...ADD CONSTRAINT을 실행하십시오. 소스 테이블에서 인덱스가 파티션된 경우 제한조건을 충족하는 데 필요한 모든 인덱스는 목표 테이블에 이미 존재합니다.

소스 테이블 작성 시 지정된 테이블 레벨 INDEX IN 옵션은 목표 테이블에서 상속하지 않지만 파티션 레벨 INDEX IN(지정된 경우) 또는 분리 파티션의 디폴트 인덱스 테이블 스페이스는 여전히 목표 테이블의 인덱스 테이블 스페이스입니다.

데이터 파티션 접속 해제 시 일부 통계는 접속 해제되는 파티션에서 목표 테이블로 이전됩니다. 특히 파티션된 인덱스에 대한 SYSINDEXPARTITIONS의 통계는 새로 접속 해제된 테이블에 대한 항목 SYSINDEXES로 이전됩니다. SYSDATAPARTITIONS의 통계는 새로 접속 해제된 테이블의 SYSTABLES로 복사됩니다.

주: 접속 해제 조작 완료 이후에는 많은 통계가 이전되지 않으므로 새로 접속 해제된 테이블과 소스 테이블에서 모두 DETACH 조작 완료 후 RUNSTATS를 실행해야 합니다.

데이터 파티션 접속 해제의 속성

ALTER TABLE문의 DETACH PARTITION절을 사용하여 파티션된 테이블에서 데이터 파티션을 접속 해제하면, 이는 독립형, 파티션되지 않은 목표 테이블이 됩니다. 새 목표 테이블의 대부분의 속성은 소스 테이블로부터 상속됩니다. 소스 테이블로부터 상속되지 않은 속성은 DETACH 조작을 실행하는 사용자가 목표 테이블을 작성한 것처럼 설정됩니다.

DETACH 이후 목표 테이블은 소스 테이블에 정의된 모든 파티션된 인덱스를 상속합니다. 이 인덱스는 시스템이 생성한 인덱스 또는 사용자 정의 인덱스를 모두 포함합니다. 인덱스 오브젝트는 접속 해제 조작 중에 전혀 이동되지 않습니다. 접속 해제되는 데이터 파티션의 인덱스 파티션 메타데이터는 SYSINDEXPARTITIONS 카탈로그에서 제거됩니다. 새 항목은 새 테이블의 SYSINDEXES에 추가됩니다. 소스 테이블에서 지정된 파티션된 인덱스의 인덱스 ID(IID)는 목표 테이블의 인덱스 IID가 됩니다(테이블에 대해서는 IID가 고유하게 남아 있으며 분리 중에 변경되지 않습니다).

새 테이블에 잔존하는 인덱스의 인덱스 이름은 SQLyymmddhhmmssxxx 양식으로 시스템에서 생성됩니다. 경로 인덱스, 영역 인덱스 및 MDC 블록 인덱스는 SYSTEM 스키마의 파트로 작성됩니다. 다른 모든 인덱스는 새 테이블 스키마의 파트로 작성됩니다. 인덱스는 새 테이블로 이전되므로 고유한 1차 키 제한조건을 강제 실행하는 인덱스와 같이 시스템이 생성한 인덱스는 새 테이블 스키마의 파트로 작성됩니다. 소스 테이블에 대한 제한조건은 DETACH 이후 목표 테이블에서 상속하지 않습니다.

RENAME문을 사용하여 다른 시간에 SYSTEM 스키마에 없는 인덱스 이름을 바꿀 수 있습니다.

접속 해제 조작 완료 후 새 테이블에서 ALTER TABLE ... ADD CONSTRAINT문을 사용하여 소스 테이블에서와 같이 새 테이블에서 동일한 제한조건을 강제 실행할 수 있습니다.

소스 테이블의 테이블 레벨 INDEX IN절에 지정된 테이블 스페이스 위치는 새 목표 테이블에 상속되지 않습니다. 정확히 말하자면, 파티션 레벨 INDEX IN절에 지정된 테이블 스페이스 위치나 새 테이블의 디폴트 인덱스 테이블 스페이스는 새 테이블의 인덱스 테이블 스페이스 위치로 지속됩니다.

목표 테이블이 상속하는 속성

목표 테이블이 상속하는 속성은 다음과 같습니다.

- 다음과 같은 컬럼 정의를 상속합니다.
  - 컬럼 이름
  - 데이터 유형(CHAR 및 DECIMAL과 같은 길이 및 정밀도를 갖는 유형의 길이 및 정밀도를 포함함)
  - 널 가능성
  - 컬럼 디폴트값
  - 인라인 길이
  - 코드 페이지(SYSCAT.COLUMNS 카탈로그 뷰의 CODEPAGE 컬럼)
  - LOB용 로깅(SYSCAT.COLUMNS 카탈로그 뷰의 LOGGED 컬럼)
  - LOB 간략화(SYSCAT.COLUMNS 카탈로그 뷰의 COMPACT 컬럼)
  - 압축(SYSCAT.COLUMNS 카탈로그 뷰의 COMPRESS 컬럼)
  - 숨겨진 컬럼 유형(SYSCAT.COLUMNS 카탈로그 뷰의 HIDDEN 컬럼)
  - 컬럼 순서
- 소스 테이블에 다차원적으로 클러스터된(MDC) 테이블이 있는 경우 목표 테이블도 동일한 차원 컬럼을 사용하여 정의된 MDC 테이블입니다. 소스 테이블이 MDC인 경우, 접속 해제와 동일한 작업 단위에서는 새로 접속 해제된 테이블에 대한 액세스가 허용되지 않습니다.
- 블록 인덱스 정의. DETACH 조작이 커미트된 후, 새로 접속 해제된 독립 테이블에 처음 액세스할 때 인덱스가 재빌드됩니다.
- 테이블 스페이스 ID 및 테이블 오브젝트 ID는 소스 테이블이 아닌 데이터 파티션으로부터 상속됩니다. 이는 DETACH 조작 중에는 테이블 데이터가 이동하지 않기 때문입니다. 카탈로그 중에 소스 데이터 파티션에 있는 SYSCAT.DATAPARTITIONS 카탈로그 뷰의 TBSPACEID 컬럼은 SYSCAT.TABLES 카탈로그 뷰의 TBSPACEID 컬럼이 됩니다. 테이블 스페이스 이름으로 변환된 다음에는 목표 테이블에 있는 SYSCAT.TABLES 카탈로그 뷰의 TBSPACE 컬럼이 됩니다. 소스 데이터 파티션에 있는 SYSCAT.DATAPARTITIONS 카탈로그 뷰의 PARTITIONOBJECTID 컬럼은 목표 테이블에 있는 SYSCAT.TABLES 카탈로그 뷰의 TABLEID 컬럼이 됩니다.

- 소스 데이터 파티션에 있는 SYSCAT.DATAPARTITIONS 카탈로그 뷰의 LONG_TBSPACEID 컬럼은 테이블 스페이스 이름으로 변환된 다음 목표 테이블에 있는 SYSCAT.TABLES의 LONG_TBSPACE 컬럼이 됩니다.
- 소스 데이터 파티션(파티션 레벨 인덱스 테이블 스페이스)의 SYSDATAPARTITIONS에 있는 INDEX_TBSPACEID 컬럼 값은 테이블 스페이스 이름으로 변환되고 목표 테이블의 SYSTABLES에 있는 INDEX_TBSPACE 값이 됩니다. CREATE TABLE 문에서 테이블 레벨 INDEX IN <table space>에 지정된 인덱스 테이블 스페이스는 목표 테이블에서 상속하지 않습니다.
- 테이블 스페이스 위치
- 멀티파티션 데이터베이스의 분산 맵 ID(SYSCAT.TABLES 카탈로그 뷰의 PMAP_ID)
- 여유 공간 비율(SYSCAT.TABLES 카탈로그 뷰의 PCTFREE 컬럼)
- 추가 모드(SYSCAT.TABLES 카탈로그 뷰의 APPEND_MODE 컬럼)
- 선호하는 잠금 세분화도(SYSCAT.TABLES 카탈로그 뷰의 LOCKSIZE 컬럼)
- 데이터 캡처(SYSCAT.TABLES 카탈로그 뷰의 DATA_CAPTURE 컬럼)
- VOLATILE(SYSCAT.TABLES 카탈로그 뷰의 VOLATILE 컬럼)
- DROPRULE(SYSCAT.TABLES 카탈로그 뷰의 DROPRULE 컬럼)
- 압축(SYSCAT.TABLES 카탈로그 뷰의 COMPRESSION 컬럼)
- 최대 여유 공간 검색(SYSCAT.TABLES 카탈로그 뷰의 MAXFREESPACESEARCH 컬럼)

주: 파티션된 계층 또는 임시 테이블, 범위 클러스터 테이블 및 파티션된 뷰는 지원되지 않습니다.

소스 테이블로부터 상속되지 않는 속성

소스 테이블로부터 상속되지 않는 속성은 다음과 같습니다.

- 목표 테이블 유형은 상속되지 않습니다. 목표 테이블은 항상 일반 테이블입니다.
- 특권 및 권한
- 스키마
- 생성된 컬럼, 식별 컬럼, 점검 제한조건, 참조 제한조건. 소스 컬럼이 생성된 컬럼 또는 식별 컬럼인 경우, 해당 목표 컬럼은 명시적 디폴트값을 갖고 있지 않습니다. 즉, 디폴트값이 NULL입니다.
- 테이블 레벨 인덱스 테이블 스페이스(SYSCAT.TABLES 카탈로그 뷰의 INDEX_TBSPACE 컬럼). DETACH 조작으로 인해 생긴 테이블 인덱스는 테이블과 동일한 테이블 스페이스에 있게 됩니다.
- 트리거
- 1차 키 제한조건 및 고유 키 제한조건

- 파티션되지 않은 인덱스 통계는 상속되지 않습니다.
- 속성 목록에 언급되지 않은 기타 모든 속성은 소스 테이블에서 명시적으로 상속됩니다.

파티션된 테이블에 데이터 파티션 추가

테이블을 작성한 후 ALTER TABLE문을 사용하여 파티션된 테이블을 수정할 수 있습니다. 특별히, ADD PARTITION절을 사용하여 새 데이터 파티션을 기존의 파티션된 테이블에 추가할 수 있습니다. 데이터가 외부 소스로부터 롤인되기 보다는 소량으로 반입되거나 데이터를 직접 파티션된 테이블로 삽입 또는 로드할 경우, 데이터 파티션에 데이터가 항상 추가되는 상황에서는 파티션된 테이블에 데이터 파티션을 추가하는 것이 데이터 파티션을 접속하는 것보다 유리합니다. 특정 예로 1월 데이터용 데이터 파티션으로 데이터를 매일 로드하거나 개별 행을 삽입하는 것을 들 수 있습니다.

특정 테이블 스페이스 위치에 새 데이터 파티션을 추가하려면, IN절이 ALTER TABLE ADD PARTITION문의 옵션으로 추가됩니다.

데이터 파티션의 테이블 스페이스 위치에서 독립된 특정 테이블 스페이스 위치에 새 데이터 파티션의 파티션된 인덱스를 추가하려면, 파티션 레벨 INDEX IN절이 ALTER TABLE ADD PARTITION문의 옵션으로 추가됩니다. INDEX IN 옵션이 지정된 경우, 디폴트로 새 데이터 파티션의 모든 파티션된 인덱스는 데이터 파티션과 같은 테이블 스페이스에 상주합니다. 파티션된 테이블에 파티션된 인덱스가 존재하는 경우, ADD PARTITION이 새 파티션에 대응하는 빈 인덱스 파티션을 작성합니다. 새 인덱스 파티션 항목은 각 파티션된 인덱스에 대해 시스템 카탈로그 테이블 SYSIBM.SYSINDEXPARTITIONS에 삽입됩니다.

데이터 파티션의 테이블 스페이스 위치에서 독립된 특정 테이블 스페이스 위치에 새 데이터 파티션의 Long 데이터, LOB 또는 XML 데이터를 추가하려면, 파티션 레벨 LONG IN절이 ALTER TABLE ADD PARTITION문의 옵션으로 추가됩니다.

제한사항 및 사용법 지침

- 데이터 파티션을 파티션되지 않은 테이블에 추가할 수는 없습니다. 기존 테이블을 파티션된 테이블로 이주하는 방법에 대한 자세한 정보는 207 페이지의 『기존 테이블 및 뷰를 파티션된 테이블로 이주』의 내용을 참조하십시오.
- 각 새 데이터 파티션 값의 범위는 STARTING 및 ENDING절에 의해 판별됩니다.
- STARTING 및 ENDING절 중 하나 또는 둘 다를 제공해야 합니다.
- 새 범위가 기존 데이터 파티션의 범위와 겹치지 않아야 합니다.
- 첫 번째 기존 데이터 파티션 앞에 새 데이터 파티션을 추가할 경우 STARTING절을 지정해야 합니다. 이 범위를 개방된 종료 범위로 만들려면 MINVALUE를 사용하십시오.

- 마찬가지로, 마지막 기존 데이터 파티션 뒤에 새 데이터 파티션을 추가하려면 ENDING 절을 지정해야 합니다. 이 범위를 개방된 종료 범위로 만들려면 MAXVALUE를 사용하십시오.
- STARTING절을 생략할 경우, 데이터베이스는 이전 데이터 파티션의 종료 바운드 직후에 시작 바운드를 조작합니다. 마찬가지로, ENDING절을 생략할 경우 데이터베이스는 다음 데이터 파티션 바운드 직전에 종료 바운드를 조작합니다.
- start절 및 end절 구문은 CREATE TABLE문에 지정된 구문과 동일합니다.
- ADD PARTITION에 IN, INDEX IN 또는 LONG IN절을 지정하지 않을 경우, CREATE TABLE문에 사용된 방법과 동일한 방법을 사용하여 데이터 파티션을 배치할 테이블 스페이스를 선택합니다.
- 패키지는 ALTER TABLE ...ADD PARTITION 조작 중에 무효화됩니다.
- 새로 추가된 데이터 파티션은 ALTER TABLE문이 커미트되면 사용 가능합니다.

ADD 조작에 대한 STARTING 또는 ENDING 바운드를 생략하는 것은 범위 값에 있는 갭을 채울 때에도 사용됩니다. 다음은 시작 바운드만이 지정된 ADD 조작을 사용하여 갭을 채우는 예입니다.

```
CREATE TABLE hole (c1 int) PARTITION BY RANGE (c1)
(STARTING FROM 1 ENDING AT 10, STARTING FROM 20 ENDING AT 30);
DB20000I  SQL 명령이 완료되었습니다.

ALTER TABLE hole ADD PARTITION STARTING 15;
DB20000I  SQL 명령이 완료되었습니다.

SELECT SUBSTR(tabname, 1,12) tabname,
SUBSTR(datapartitionname, 1, 12) datapartitionname,
seqno, SUBSTR(lowvalue, 1, 4) lowvalue, SUBSTR(highvalue, 1, 4) highvalue
FROM SYSCAT.DATAPARTITIONS WHERE TABNAME='HOLE' ORDER BY seqno;

TABNAME DATAPARTITIONNAME SEQNO LOWVALUE HIGHVALUE
------------ ---------------- ----------- -------- ---------
HOLE PART0 0 1 10
HOLE PART2 1 15 20
HOLE PART1 2 20 30

3 레코드가 선택되었습니다.
```

예 1: 기존의 파티션된 테이블에 범위가 901에서 1000인 값을 포함하는 데이터 파티션을 추가합니다. 판매액 테이블에 9개의 범위(0-100, 101-200, ..., 801-900)가 포함되어 있다고 가정합니다. 예에서는 STARTING절의 제외(exclusion)에 표시된 대로 테이블의 맨 끝에 추가 범위를 추가합니다.

```
 ALTER TABLE sales ADD PARTITION dp10
(ENDING AT 1000 INCLUSIVE)
```

데이터 파티션의 테이블 스페이스 위치에서 독립된 특정 테이블 스페이스 위치에 새 데이터 파티션의 파티션된 인덱스를 추가하려면, 파티션 레벨 INDEX IN절이 ALTER TABLE ADD PARTITION문의 옵션으로 추가됩니다. INDEX IN 옵션이 지정된 경우, 디폴트로 새 데이터 파티션의 모든 파티션된 인덱스는 데이터 파티션과 같은 테이

블 스페이스에 상주합니다. 파티션된 테이블에 파티션된 인덱스가 존재하는 경우, ADD PARTITION이 새 파티션에 대응하는 빈 인덱스 파티션을 작성합니다. 새 인덱스 파티션 항목은 각 파티션된 인덱스에 대해 시스템 카탈로그 테이블 SYSIBM.SYSINDEXPARTITIONS에 삽입됩니다.

예제 2: 데이터 파티션의 나머지 부분에서 긴 데이터와 인덱스를 분리하여 기존의 파티션된 테이블에 데이터 파티션을 추가하십시오.

```
ALTER TABLE newbusiness ADD PARTITION IN tsnewdata
INDEX IN tsnewindex LONG IN tsnewlong
```

데이터 파티션 삭제

ALTER TABLE문을 DETACH PARTITION절과 함께 사용하여 데이터 파티션을 삭제할 수 있습니다. 그러나 DROP TABLE문을 사용하여 접속 해제된 테이블을 삭제할 수 있습니다.

파티션된 테이블에서 데이터 파티션을 접속 해제하기 위해서는 사용자가 다음과 같은 권한 또는 특권을 갖고 있어야 합니다.

- DETACH 조작을 수행하는 사용자는 소스 테이블에서 AlTER, SELECT 및 DELETE를 수행하는데 필요한 권한을 갖고 있어야 합니다.
- 또한 목표 테이블을 작성하는데 필요한 권한도 갖고 있어야 합니다. 따라서, 데이터 파티션을 접속 해제하도록 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권이 목표 테이블에 대해 최소한 다음 권한 중 하나를 포함해야 합니다.
  - DBADM 권한
  - 다음 권한 중 하나를 비롯하여 테이블에 사용되는 테이블 스페이스에 대한 USE 특권 및 데이터베이스에 대한 CREATETAB 권한
    - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

테이블을 삭제하기 위해서는 사용자가 다음과 같은 권한 또는 특권을 갖고 있어야 합니다.

- SYSCAT.TABLES의 DEFINER 컬럼에 기록된 정의자이거나 최소한 다음 특권 중 하나를 갖고 있어야 합니다.
  - DBADM 권한
  - 테이블의 스키마에 대한 DROPIN 특권
  - 테이블에 대한 CONTROL 특권

주: 데이터베이스 파티션을 접속 해제할 경우 명령문의 권한 부여 ID가 효율적으로 CREATE TABLE문을 발행하려면 해당 조작을 수행하기 위한 필수 권한이 있어야 합니다. 테이블 스페이스는 접속 해제된 데이터 파티션이 이미 상주하는 장소입니다. ALTER TABLE문의 권한 부여 ID는 마치 사용자가 CREATE TABLE문을 발행한 것처럼 CONTROL 권한을 가진 새 테이블의 정의자가 됩니다. 변경된 테이블의 특권은 새 테이블로 이동되지 않습니다. ALTER TABLE문의 권한 부여 ID 및 DBADM 또는 SYSADM만이 ALTER TABLE ... DETACH PARTITION 조작 직후에 데이터에 액세스할 수 있습니다.

명령행에서 파티션된 테이블의 데이터 파티션을 접속 해제하려면 ALTER TABLE문을 DETACH PARTITION절과 함께 발행하십시오.

DB2 명령행 처리기(CLP)에서 테이블을 삭제할 수 있습니다.

명령행에서 테이블을 삭제하려면 DROP TABLE문을 실행하십시오. 다음 예에서, dec01 데이터 파티션은 주식 테이블에서 접속 해제되어 정크 테이블에 배치됩니다. 연관된 데이터 파티션을 효율적으로 삭제하여 정크 테이블을 삭제할 수 있습니다.

```
ALTER TABLE stock DETACH PART dec01 INTO junk;
DROP TABLE junk;
```

주: DETACH 조작을 가능한 빠르게 작성할 수 있도록 소스 테이블에서의 인덱스 정리가 백그라운드 비동기 인덱스 정리 프로세스를 사용하여 자동으로 수행됩니다. 접속 해제된 종속 테이블이 있는 경우, 접속 해제되어도 접속 해제된 데이터 파티션이 독립형 테이블이 되지는 않습니다. 이 경우, SET INTEGRITY문을 발행하여 접속 해제를 완료하고 테이블에 액세스할 수 있게 만들어야 합니다.

시나리오: 파티션된 테이블의 데이터 회전

DB2 데이터베이스에서 데이터 회전은 사용하지 않는 데이터를 테이블에서 제거한 다음 새 데이터를 추가하여 데이터 파티션의 스페이스를 재사용하는 방법입니다. 테이블 파티션 기능을 사용하면, 사용하지 않는 데이터에서 데이터 파티션을 접속 해제한 후 최신 데이터에 새 데이터 파티션을 접속할 수 있습니다.

파티션된 테이블에서 데이터 파티션을 접속 해제하기 위해서는 사용자가 다음과 같은 권한 또는 특권을 갖고 있어야 합니다.

- DETACH 조작을 수행하는 사용자는 소스 테이블에서 AlTER, SELECT 및 DELETE를 수행하는데 필요한 권한을 갖고 있어야 합니다.
- 또한 목표 테이블을 작성하는데 필요한 권한도 갖고 있어야 합니다. 따라서, 데이터 파티션을 접속 해제하도록 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권이 목표 테이블에 대해 최소한 다음과 같은 권한 또는 특권 중 하나를 포함해야 합니다.
  - DBADM 권한

  - 다음 권한 중 하나를 비롯하여 테이블에 사용되는 테이블 스페이스에 대한 USE 특권 및 데이터베이스에 대한 CREATETAB 권한
    - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

데이터 파티션을 접속하도록 테이블을 변경하려면 사용자가 다음과 같은 권한 또는 특권이 있어야 합니다.

- 접속을 수행하는 사용자는 ALTER 및 목표 테이블에 INSERT하기 위한 권한이 있어야 합니다.
- 사용자는 소스 테이블에서 SELECT 및 소스 테이블을 DROP 가능 해야 합니다. 따라서, 데이터 파티션을 접속하도록 테이블을 변경하기 위해서는 명령문의 권한 부여 ID가 보유하는 특권에 소스 테이블에 대해 최소한 다음 중 하나를 포함해야 합니다.
  - 소스 테이블에 대한 DATAACCESS 권한 또는 SELECT 특권 및 소스 테이블의 스키마에 대한 DBADM 권한 또는 DROPIN 특권
  - 소스 테이블에 대한 CONTROL 특권

명령행에서 파티션된 테이블의 데이터를 회전하려면 ALTER TABLE문을 발행하십시오. 다음 예는 2001년 12월 데이터를 제거한 후 이를 2003년 12월의 최신 데이터로 바꾸어 주식 테이블을 갱신하는 방법을 보여줍니다.

1. *주식* 테이블에서 이전 데이터를 제거하십시오.

   ```
   ALTER TABLE stock DETACH PARTITION dec01 INTO newtable;
   ```

2. 새 데이터를 로드하십시오. REPLACE 옵션과 함께 LOAD를 사용하면 기존의 데이터를 겹쳐씁니다.

   ```
   LOAD FROM data_file OF DEL REPLACE INTO newtable
   ```

   주: 접속 해제된 종속 테이블이 있으면 접속 해제된 테이블을 로드하기 전에 접속 해제된 종속 테이블에서 SET INTEGRITY문을 실행해야 합니다.

3. 원할 경우, 데이터 정리를 수행하십시오. 데이터 정리 활동에는 다음이 포함됩니다.
   - 누락된 값 채우기
   - 불일치 및 미완료 데이터 삭제
   - 다중 소스로부터 도착한 중복 데이터 제거
   - 데이터 변환
     - 정규화(다른 방법으로 동일한 값을 표시하는 다른 소스에서 도착한 데이터는 데이터를 웨어하우스로 롤인하는 중에 조정되어야 합니다.)
     - 집계(웨어하우스에 저장하기에 너무 자세한 원시 데이터는 롤인 중에 사전 집계해야 합니다.)

4. 새 데이터를 새 범위로서 접속하십시오.

```
ALTER TABLE stock ATTACH PARTITION dec03
STARTING '12/01/2003' ENDING '12/31/2003'
 FROM newtable;
```

   데이터 파티션을 접속하면 쿼리를 드레인하고 패키지가 무효화됩니다.

5. SET INTEGRITY문을 사용하여 인덱스 및 기타 종속 오브젝트를 갱신하십시오. SET INTEGRITY문 실행 중에 읽기 및 쓰기 액세스가 허용됩니다.

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

시나리오: 파티션된 테이블 데이터 롤인 및 롤아웃

이 사나리오에는 각 월이 시작할 때 새 데이터가 롤인되고 특정 일에 기준하여 기존 데이터가 롤아웃되는 데이터 웨어하우스의 일반 관리 조작을 설명합니다.

시나리오 1은 테이블에서 사용 안하는 데이터를 제거하여 DETACH 조작(롤아웃)을 다룹니다. 변형에는 데이터 삭제 및 다른 테이블로 데이터 이동이 포함됩니다. 두 개의 시나리오는 모두 테이블에 새 데이터를 로드하여 ADD 조작과 ATTACH 조작(롤인)을 다룹니다. 다음과 같은 변형을 포함합니다.

1. 데이터 변환, 파티션되지 않은 테이블에 로드 후 데이터 파티션 접속(전형적인 추출, 변환 및 로드(ETL))
2. 파티션되지 않은 테이블에 데이터 로드, 데이터 변환
3. 데이터 파티션 접속

시나리오 1: 파티션된 테이블을 사용할 때 롤아웃 조작은 단순히 적절한 데이터 파티션에 대한 DETACH 조작

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
COMMIT WORK
```

DETACH 조작을 가속화하기 위해 소스 테이블에서의 인덱스 정리가 백그라운드 비동기 인덱스 정리 프로세스를 통해 자동으로 수행됩니다. 소스 테이블에 정의된 접속 해제 종속이 없다면 SET INTEGRITY문을 발행하여 DETACH 조작을 완료할 필요가 없습니다.

테이블 삭제 대신 테이블을 다른 테이블에 접속하거나 테이블을 절단하여 새 데이터를 재접속하기 전에 새 데이터를 로드하는 테이블로 사용합니다. 접속 해제된 종속이 있는 저장 테이블을 제외하고 이러한 조작을 즉시, 심지어 비동기 인덱스 정리가 완료되기 전에 수행할 수 있습니다.

접속 해제된 테이블이 아직 액세스 가능하지 않음을 확인하려면 SYSCAT.TABDETACHEDDEP 카탈로그 뷰를 쿼리하십시오. 액세스 불가능한 접속 해제된 테이블이 발견되면, SET INTEGRITY문을 IMMEDIATE CHECKED 옵션과

함께 모든 접속 해제된 종속에 실행하여 접속 해제된 테이블을 일반 액세스 가능한 테이블로 전환하십시오. 모든 접속 해제된 종속이 유지보수되기 전에 접속 해제된 테이블에 액세스를 시도하면 SQL20285N 오류 코드가 리턴됩니다.

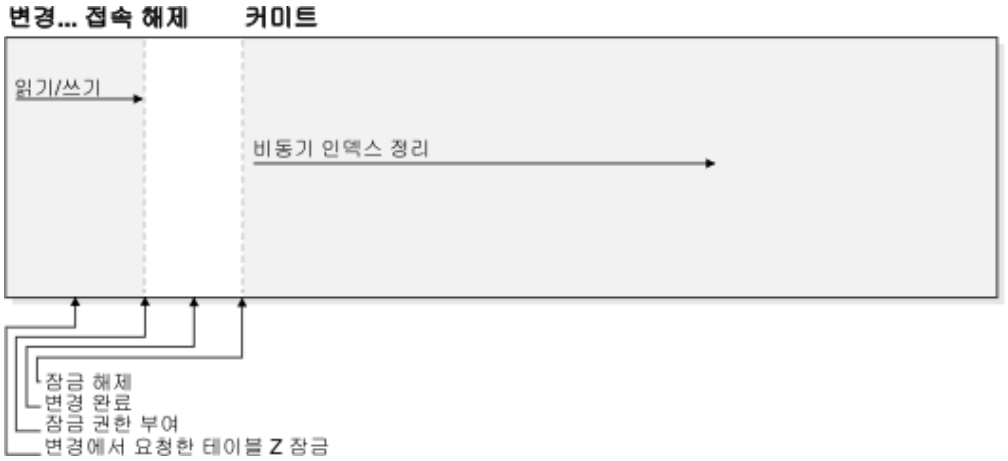


그림 37. 이 그룹은 DETACH 조작 도중 데이터 사용 가능성의 단계를 보여줍니다. 비동기 인덱스 정리는 접속 해제된 종속이 없는 경우 DETACH 조작 커미트 직후 시작됩니다. 그렇지 않을 경우 접속 해제된 종속 유지보수가 커미트된 후 비동기 인덱스 정리가 시작됩니다.

시나리오 2: 비어 있는 범위 새로 작성

다음 시나리오에는 데이터를 파티션되지 않은 테이블에 로드한 후 해당 데이터 파티션을 나머지 테이블에 추가하는 단계를 설명합니다.

```
ALTER TABLE stock ADD PARTITION dec03
STARTING FROM '12/01/2003' ENDING AT '12/31/2003';
```

이 ALTER TABLE ... ADD 조작은 저장 테이블에 대해 실행 중인 쿼리를 드레인하고 패키지를 무효로 만듭니다. 즉, 기존 쿼리가 ADD 조작이 계속하기 전에 정상적으로 완료됩니다. ADD 조작이 발행된 후에는 주식 테이블에 액세스하는 모든 새 쿼리가 잠금을 블록합니다.

테이블에 데이터 로드:

```
LOAD FROM data_file OF DEL INSERT
INTO stock ALLOW READ ACCESS;
```

SET INTEGRITY문을 사용하여 제한조건을 확인하고 종속 구체화된 쿼리 테이블(MQT)을 새로 고칩니다.

```
SET INTEGRITY FOR stock ALLOW READ
ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;

COMMIT WORK;
```

추가 정보: 테이블에 제한조건 또는 MQT 정의가 없을 경우, ALTER TABLE ...ADD PARTITION 다음에 LOAD 조작을 하는 것이 LOAD 조작 다음에 ALTER TABLE

...ATTACH를 수행하는 것과 비교했을 때, SET INTEGRITY문없이 새 데이터를 사용 가능하게 만들 수 있다는 이점이 있습니다. 새 데이터 파티션을 추가하고 테이블에 직접 데이터를 로드할 경우 불리한 제약이 있습니다. ALTER TABLE ... ADD PARTITION문을 사용하면 로드 조작 자체와 모든 후속 SET INTEGRITY문 실행시 둘 다 테이블을 갱신할 수는 큰 단점이 있습니다. ALTER TABLE ... ADD PARTITION문 및 ALTER TABLE ... ATTACH PARTITION문은 둘 다 패키지를 무효화시키는 반면에, ALTER ... ATTACH PARTITION 조작 후 LOAD 명령은 데이터 사용 가능성을 향상시킵니다. 그러나 ALTER TABLE ... ADD PARTITION문 뒤의 IMPORT 명령 또는 일반 INSERT문은 데이터가 대형 블록에 롤인되지 않고 조금만 있는 상황에 적합합니다. 롤인된 데이터가 데이터 파티션 바운더리와 일치하지 않을 경우 데이터 파티션 추가를 할 수 있습니다.

시나리오 3: 파티션된 테이블을 사용할 때 롤인 조작은 단순히 새로 로드된 데이터 파티션의 ATTACH 조작

이 시나리오에서 기존 파티션된 테이블에 새 범위의 데이터를 로드하기 쉽도록 ATTACH가 사용됩니다. 일반적으로 데이터는 비어 있는 새 테이블에 로드되어 목표 테이블에 영향을 주지 않고 필요한 데이터 정리와 점검을 수행합니다. 데이터 준비가 완료되면 새로 로드된 데이터파티션에 접속합니다.

```
CREATE TABLE dec03(.....);
LOAD FROM data_file OF DEL REPLACE INTO dec03;
```

테이블 데이터에 롤인하기 전, 데이터가 접속되기 전에 데이터를 정리해야 합니다. 데이터 정리 활동에는 다음이 포함됩니다.

- 누락된 값 채우기
- 불일치 및 미완료 데이터 삭제
- 다중 소스로부터 도착한 중복 데이터 제거
- 데이터 변환
  - 정규화(다른 방법으로 동일한 값을 표시하는 다른 소스의 데이터는 데이터를 웨어하우스로 롤인하는 중에 조정되어야 합니다.)
  - 집계(웨어하우스에 저장하기에 너무 자세한 원시 데이터는 롤인 중에 사전 집계해야 합니다.)

다음으로 데이터를 다음과 같이 롤인합니다.

```
ALTER TABLE stock ATTACH PARTITION dec03
STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
FROM dec03;
```

ATTACH 조작시 STARTING 및 ENDING절이 하나 또는 둘 다 제공되어야 하고, 하한(STARTING)이 상한(ENDING)보다 적거나 같아야 합니다. 또 새로 접속된 데이터 파티션이 목표 테이블의 기존 데이터 파티션 범위를 겹치면 안됩니다. 최고 범위를

MAXVALUE로 정의한 후 높은 범위를 새로 추가하면 기존 범위가 새 범위에 겹치므로 추가가 실패합니다. 이러한 제한사항은 MINVALUE에도 적용됩니다. 범위에 있는 기존 갭에 들어가지 않으면 새 데이터 파티션을 추가 또는 접속할 수 없습니다. 사용자가 지정하지 않는 바운더리는 테이블이 작성될 때 결정됩니다.

ALTER TABLE ... ATTACH 조작은 모든 쿼리를 드레인하고 판매 품목 테이블에 종속된 패키지를 무효화합니다. 즉, 기존 쿼리가 ATTACH 조작이 계속하기 전에 정상적으로 완료됩니다. ATTACH 조작이 발행된 후에는 주식 테이블에 액세스하는 모든 새 쿼리가 잠금을 블록합니다. 저장 테이블은 이 전환 동안 z-잠금(완전 액세스 불가능) 상태입니다. 접속된 데이터 파티션의 데이터가 아직 SET INTEGRITY문으로 유효화되지 않았기 때문에 데이터가 표시되지 않습니다. **추가 정보:** COMMIT WORK 문을 ATTACH 조작 후에 즉시 발행하여 테이블을 사용 가능하게 하십시오.

```
COMMIT WORK;
```

새로 접속한 데이터가 범위 안에 있는지 확인하려면 SET INTEGRITY문이 필요합니다. 인덱스 및 MQT와 같은 다른 종속 오브젝트의 유지보수에도 SET INTEGRITY문이 필요합니다. SET INTEGRITY문이 커미트되어야 새 데이터가 표시됩니다. 온라인 SET INTEGRITY가 사용되었을 경우 저장 테이블의 기존 데이터를 읽기 및 쓰기 모두 사용 가능합니다. SET INTEGRITY가 실행 중일 때 디폴트값은 ALLOW NO ACCESS 모드입니다.

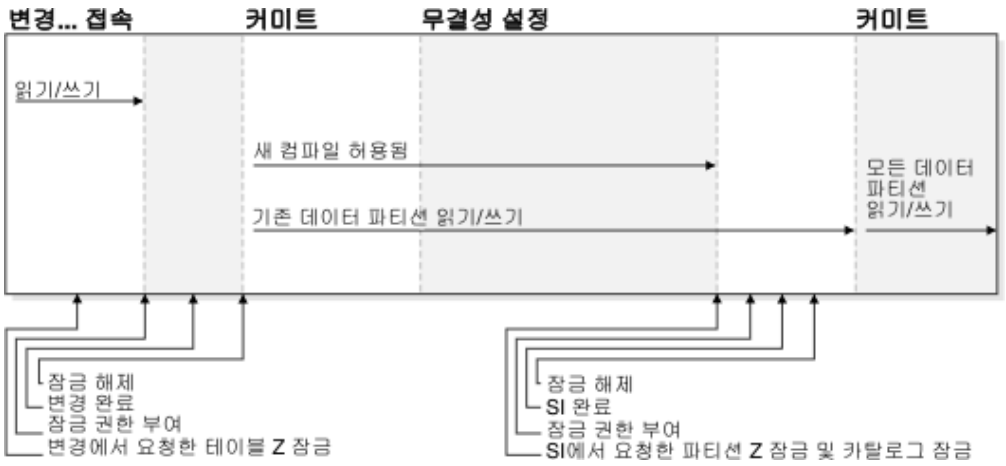


그림 38. 이 그림은 ATTACH 조작 도중 데이터 사용 가능성의 단계를 보여줍니다.

주: SET INTEGRITY가 실행 중일 때 테이블에 DDL 또는 유틸리티 유형 조작을 실행할 수 없습니다. 유틸리티 유형 조작으로 LOAD, REORG, REDISTRIBUTE, ALTER TABLE(예: 컬럼 추가, ADD, ATTACH, DETACH, ALTER를 사용하여 『원래 로그되지 않음』에 TRUNCATE) 및 INDEX CREATE가 있으며 이 외의 조작도 가능합니다.

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

무결성 설정으로 새로 접속된 데이터 파티션의 데이터가 유효한지 확인합니다.

다음으로 테이블을 사용 가능하도록 트랜잭션을 커미트합니다.

```
COMMIT WORK;
```

범위 밖의 모든 행 또는 다른 제한조건을 위반하는 행은 stock_ex 예외 테이블로 이동됩니다. stock_ex를 쿼리하여 위반 행을 조사 및 정리한 다음 테이블에 다시 삽입할 수 있습니다.

제 14 장 로드

병렬 처리 및 로딩

로드 유틸리티는 SMP(Symmetric Multiprocessor) 환경과 같이 다중 프로세서 또는 다중 스토리지 디바이스를 사용하는 하드웨어 구성의 이점을 활용합니다.

로드 유틸리티를 사용하여 많은 데이터를 병렬 처리하는 여러 가지 방법이 있습니다. 한 가지 방법은 다중 스토리지 디바이스를 사용하는 것입니다. 그러면 로드 조작 중 입출력 병렬 처리가 가능합니다(그림 39 참조). 또 다른 방법은 SMP 환경에서 다중 프로세서를 사용하는 것입니다. 그러면 파티션 내 병렬 처리가 가능합니다(그림 40 참조). 두 방법을 함께 사용하여 보다 빨리 데이터를 로드할 수도 있습니다.

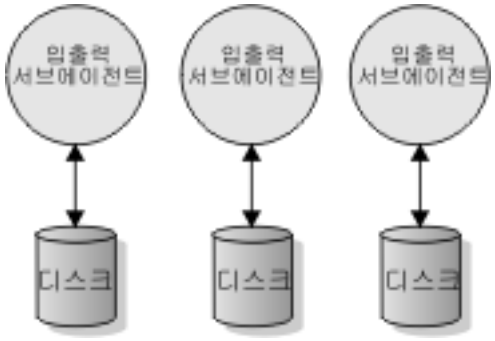


그림 39. 데이터 로드 시 입출력 병렬 처리 활용

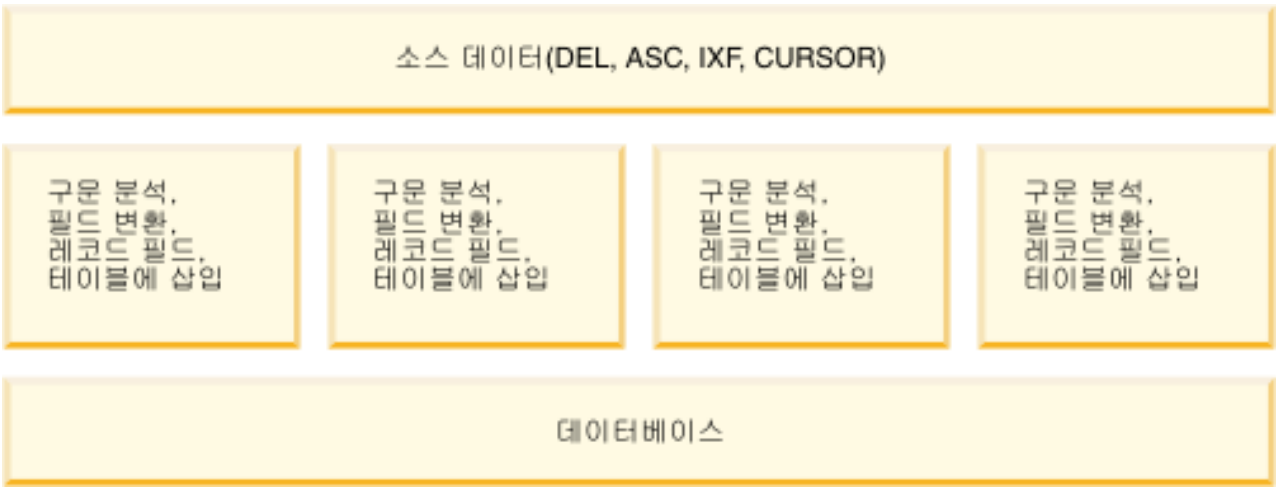


그림 40. 데이터 로드 시 파티션 내 병렬 처리 활용

다차원적으로 클러스터된 고려사항

다음 제한사항은 다차원적으로 클러스터된(MDC) 테이블로 데이터를 로드하는 경우 적용됩니다.

- LOAD 명령의 SAVECOUNT 옵션은 지원되지 않습니다.
- 이러한 테이블은 고유한 여유 공간을 관리하므로 `totalfreespace` 파일 유형 수정자는 지원되지 않습니다.
- MDC 테이블에 대해 `anyorder` 파일 유형 수정자가 필요합니다. `anyorder` 수정자 없이 MDC 테이블로 로드가 실행되면 유틸리티에서 명시적으로 사용 가능해집니다.

MDC 테이블에서 LOAD 명령을 사용하는 경우 고유 제한조건 위반은 다음과 같이 처리됩니다.

- 로드 조작 전에 테이블에 고유 키가 포함되고 중복 레코드를 테이블로 로드한 경우 원래 레코드는 남아 있으며 새 레코드는 삭제 단계 중에 삭제됩니다.
- 로드 조작 전에 테이블에 고유 키가 없고 고유 키 및 중복 레코드를 테이블로 로드한 경우 고유 키를 포함하는 레코드 중 하나만 로드되고 나머지는 삭제 단계 중에 삭제됩니다.

  주: 로드되는 레코드와 삭제되는 레코드를 판별하는 명시적인 기술은 없습니다.

성능 고려사항

MDC 테이블을 로드할 때 로드 유틸리티의 성능을 향상시키려면 *util_heap_sz* 데이터베이스 구성 매개변수 값을 늘려야 합니다. mdc-load 알고리즘의 경우 유틸리티에서 사용 가능한 추가 메모리가 있으면 성능이 더 향상됩니다. 그러면 로드 단계 중 수행된 데이터 클러스터링 동안 디스크 입출력이 줄어듭니다. LOAD 명령의 DATA BUFFER 옵션이 지정되면 값도 증가해야 합니다. LOAD 명령을 사용하여 여러 MDC 테이블을 동시에 로드하는 경우 적절히 *util_heap_sz*를 늘려야 합니다.

모든 MDC 테이블에는 블록 인덱스가 있으므로 MDC 로드 조작은 항상 빌드 단계를 포함합니다.

로드 단계 중 블록 맵 유지보수를 위한 추가 로깅이 수행됩니다. 할당된 Extent당 약 2개의 추가 로그 레코드가 있습니다. 좋은 성능을 유지하려면 *logbufsz* 데이터베이스 구성 매개변수를 이를 고려한 값으로 설정해야 합니다.

인덱스를 포함하는 시스템 임시 테이블은 MDC 테이블로 데이터를 로드하는 데 사용됩니다. 테이블 *크기*는 로드된 구별 셀의 수에 비례합니다. 테이블에서 각 행의 *크기*는 MDC 차원 키의 *크기*에 비례합니다. 로드 조작 중 이 테이블의 처리로 인한 디스크 입출력을 최소화하려면 임시 테이블 스페이스의 버퍼 풀이 충분히 커야 합니다.

파티션된 테이블에 대한 로드 고려사항

다음 일반 제한사항을 제외하고 목표 테이블이 파티션된 경우 모든 기존 로드 기능이 지원됩니다.

- 파티셔닝 에이전트 수가 1보다 큰 경우 일관성 지점은 지원되지 않습니다.
- 데이터 파티션의 서브세트로 데이터를 로드하면서 나머지 데이터 파티션은 완전히 온라인 상태로 남는 조건은 지원되지 않습니다.
- 로드 조작에서 사용하는 예외 테이블은 파티션할 수 없습니다.
- 목표 테이블에 XML 컬럼이 포함된 경우 예외 테이블을 지정할 수 없습니다.
- 로드 유틸리티를 삽입 모드 또는 재시작 모드로 실행하고 로드 목표 테이블에 접속 해제된 종속이 있는 경우 고유 인덱스는 재빌드할 수 없습니다.
- MDC 테이블 로드와 비슷하게 파티션된 테이블을 로드할 때 입력 데이터 레코드의 정확한 순서는 보존되지 않습니다. 순서는 셀 또는 데이터 파티션에서만 유지됩니다.
- 각 데이터베이스 파티션에서 다중 포맷터를 활용하는 로드 조작은 입력 레코드의 대략적인 순서만 보존합니다. 각 데이터베이스 파티션에서 단일 포맷터를 실행하면 셀 또는 테이블 파티셔닝 키로 입력 레코드를 그룹화합니다. 각 데이터베이스 파티션에서 단일 포맷터를 실행하려면 명시적으로 1의 CPU_PARALLELISM을 요청합니다.

일반 로드 동작

로드 유틸리티는 올바른 데이터 파티션에 데이터 레코드를 삽입합니다. 스플리터(splitter)와 같은 외부 유틸리티를 사용하여 로드 전에 입력 데이터를 파티션하는 경우 관련 요구사항은 없습니다.

로드 유틸리티는 접속 해제 또는 접속된 데이터 파티션에 액세스하지 않습니다. 데이터는 표시되는 데이터 파티션에만 삽입됩니다. 표시되는 데이터 파티션은 접속 또는 접속 해제된 상태가 아닙니다. 또한 로드 교체 조작은 접속 또는 접속 해제된 데이터 파티션을 절단하지 않습니다. 로드 유틸리티는 카탈로그 시스템 테이블에서 잠금을 획득하므로 로드 유틸리티는 커미트되지 않은 ALTER TABLE 트랜잭션을 기다립니다. 이러한 트랜잭션은 카탈로그 테이블의 관련 행에서 배타적 잠금을 획득합니다. 로드 조작을 계속 진행하려면 배타적 잠금을 종료해야 합니다. 즉, 로드 조작을 실행하는 동안 커미트되지 않은 ALTER TABLE ...ATTACH, DETACH 또는 ADD PARTITION 트랜잭션이 있을 수 있습니다. 접속 또는 접속 해제된 데이터 파티션이 목표로 지정된 입력 소스 레코드가 거부되고 하나가 지정되면 예외 테이블에서 검색될 수 있습니다. 목표 테이블 데이터 파티션의 일부가 접속 또는 접속 해제된 상태임을 나타내는 정보 메시지가 메시지 파일에 작성됩니다. 목표 테이블에 대응하는 관련 카탈로그 테이블 행을 잠그면 로드 유틸리티를 실행하는 동안 ALTER TABLE ...ATTACH, DETACH 또는 ADD PARTITION 조작을 실행하여 목표 테이블의 파티셔닝을 변경하지 못합니다.

유효하지 않은 행 처리

로드 유틸리티가 가시적 데이터 파티션에 속하지 않는 레코드를 발견하면 해당 레코드

는 거부되고 로드 유틸리티는 처리를 계속합니다. 제한조건 위반 범위 때문에 거부된 레코드 수는 명시적으로 표시되지 않지만 거부된 레코드의 전체 수에 포함됩니다. 범위 위반으로 레코드가 거부된 경우 행 경고 수는 늘어나지 않습니다. 범위 위반을 발견했음을 나타내는 단일 메시지(SQL0327N)가 로드 유틸리티 메시지 파일에 작성되지만 레코드당 메시지는 로그되지 않습니다. 또한 목표 테이블의 모든 컬럼 이외에도 예외 테이블은 특별 행에서 나타나는 위반 유형을 설명하는 컬럼을 포함합니다. 파티션할 수 없는 데이터를 포함하여 유효하지 않은 데이터를 포함하는 행은 덤프 파일에 작성됩니다.

예외 테이블 삽입은 자원 소모가 많으므로 예외 테이블에 삽입할 제한조건 위반을 제어할 수 있습니다. 예를 들어 로드 유틸리티의 디폴트 동작은 범위 제한조건 또는 고유 제한조건 위반으로 거부된 행을 삽입하는 것입니다. 그렇지 않고 유효한 경우 예외 테이블로 삽입됩니다. 각각 FOR EXCEPTION절에서 NORANGEEXC 또는 NOUNIQUEEXC를 지정하여 이 동작을 끌 수 있습니다. 이러한 제한조건 위반을 예외 테이블에 삽입하지 않도록 하거나 예외 테이블을 지정하지 않는 경우 범위 제한조건 또는 고유 제한조건을 위반하는 행에 대한 정보는 유실됩니다.

실행기록 파일

목표 테이블이 파티션된 경우 해당하는 실행기록 파일 항목은 목표 테이블에 포함된 테이블 스페이스 목록을 포함하지 않습니다. 다른 조작 세분 단위 ID('T' 대신 'R')는 파티션된 테이블에서 로드 조작을 실행함을 나타냅니다.

로드 조작 종료

로드 교체를 완전히 종료하면 모든 가시적 데이터 파티션이 절단되고 로드 삽입을 종료하면 모든 가시적 데이터 파티션을 로드 전의 길이로 절단합니다. 인덱스는 로드 복사 단계에서 실패한 ALLOW READ ACCESS 로드 조작 종료 중 유효성이 확인되지 않습니다. 또한 인덱스를 처리하는 ALLOW NO ACCESS 로드 조작을 종료하는 경우에도 인덱스 유효성이 확인되지 않습니다. 인덱스 모드가 재빌드이거나 인덱스가 불일치 상태로 남게 되는 증분 유지보수 중에 키가 삽입되었기 때문입니다. 다중 목표로 데이터를 로드하는 경우 로드 복구 조작에 영향을 주지 않습니다. 단, 로드 단계 중 수행된 일관성 지점부터 로드 조작을 재시작하는 기능은 예외입니다. 이 경우 목표 테이블이 파티션되었으면 SAVECOUNT 로드 옵션이 무시됩니다. 이 동작은 MDC 목표 테이블로 데이터를 로드할 때와 일관됩니다.

생성된 컬럼

생성된 컬럼이 파티셔닝, 차원 또는 분산 키에 있는 경우 `generatedoverride` 파일 유형 수정자는 무시되고 `generatedignore` 파일 유형 수정자가 지정된 경우와 같이 로드 유틸리티에서 값을 생성합니다. 이 경우 올바르지 않은 생성된 컬럼 값을 로드하면 잘못된 물리적 위치(예: 잘못된 데이터 파티션, MDC 블록 또는 데이터베이스 파티션)에 레코드를 배치할 수 있습니다. 예를 들어 레코드가 잘못된 데이터 파티션에 배치되면 세트 무결성은 이 레코드를 다른 물리적 위치로 이동해야 합니다. 온라인 세트 무결성 조작 중 수행할 수 없습니다.

데이터 사용 가능성

현재 ALLOW READ ACCESS 로드 알고리즘은 파티션된 테이블로 확장됩니다. ALLOW READ ACCESS 로드 조작에서는 동시 판독기에서 로딩 및 비로딩 데이터 파티션 모두를 포함하여 전체 테이블에 액세스할 수 있습니다.

데이터 파티션 상태

로드에 성공한 후 가시적 데이터 파티션은 특정 조건에서 무결성 설정 보류 또는 읽기 액세스 전용 테이블 상태로 변경될 수 있습니다. 로드 조작을 유지보수할 수 없는 테이블에 대한 제한조건이 있으면 데이터 파티션은 이 상태로 배치될 수 있습니다. 이러한 제한조건으로는 점검 제한조건 및 접속 해제된 구체화된 쿼리 테이블을 포함할 수 있습니다. 실패한 로드 조작은 보이는 모든 데이터 파티션을 로드 보류 테이블 상태로 설정합니다.

오류 분리

데이터 파티션 레벨의 오류 분리는 지원되지 않습니다. 오류 분리는 오류가 발생하지 않는 데이터 파티션에서 로드를 계속하고 오류로 발생한 데이터 파티션을 중지하는 작업을 의미합니다. 오류는 다른 데이터베이스 파티션 사이에서 분리될 수 있지만 로드 유틸리티는 가시적 데이터 파티션의 서브세트에서 트랜잭션을 커미트하고 나머지 가시적 데이터 파티션을 롤백할 수 없습니다.

기타 고려사항

- 인덱스가 유효하지 않은 항목으로 표시되면 증분 인덱싱은 지원되지 않습니다. 재빌드해야 하거나 접속 해제된 종속 항목에서 SET INTEGRITY문으로 유효성을 확인해야 하는 경우 인덱스는 유효하지 않은 항목으로 간주됩니다.
- 범위로 파티션되거나 해시로 분산되거나 차원으로 구성되는 알고리즘 조합을 사용하여 파티션된 테이블로 로드하는 작업이 지원됩니다.
- 로드로 영향을 받는 오브젝트 및 테이블 스페이스 ID의 목록을 포함하는 로그 레코드의 경우 이러한 레코드(LOAD START 및 COMMIT(PENDING LIST))의 *크기*는 상당히 커질 수 있으므로 다른 응용프로그램에서 사용 가능한 사용 중인 로그 스페이스 *크기*가 줄어듭니다.
- 테이블이 파티션 및 분산된 경우 파티션된 데이터베이스 로드가 모든 데이터베이스 파티션에 영향을 주지 않을 수도 있습니다. 출력 데이터베이스 파티션의 오브젝트만 변경됩니다.
- 로드 조작 중 파티션된 테이블의 메모리 소모는 테이블 수만큼 증가합니다. 단, 전체 메모리 요구사항의 작은 비율만 데이터 파티션 수에 비례하므로 전체 증가분이 선형으로 나타나지는 않습니다.

제 15 장 파티션된 데이터베이스 환경에서 데이터 로드

로드 개요 - 파티션된 데이터베이스 환경

다중 파티션 데이터베이스에서 많은 데이터가 많은 데이터베이스 파티션에 교차하여 있습니다. 분산 키는 데이터의 각 부분이 있는 데이터베이스 파티션을 판별하는 데 사용됩니다. 데이터를 올바른 데이터베이스 파티션으로 로드하려면 데이터를 *분산*해야 합니다.

다중 파티션 데이터베이스에서 데이터를 로드하는 경우 로드 유틸리티에서는 다음을 수행할 수 있습니다.

- 입력 데이터를 병렬로 분산
- 해당 데이터베이스 파티션에서 동시에 데이터 로드
- 한 시스템에서 다른 시스템으로 데이터 전송

데이터를 다중 파티션 데이터베이스로 로드하는 작업은 테이블 잠금과 같은 데이터베이스 파티션 자원을 획득하는 *설정 단계*와 데이터베이스 파티션으로 데이터를 로드하는 로드 *단계*의 두 단계로 수행됩니다. LOAD 명령의 ISOLATE_PART_ERRS 옵션을 사용하여 이 단계 중 하나를 수행하는 동안 오류를 처리하는 방법 및 하나 이상의 데이터베이스 파티션에서 발생한 오류가 오류가 발생하지 않은 데이터베이스 파티션의 로드 조작에 영향을 미치는 방법을 선택할 수 있습니다.

다중 파티션 데이터베이스로 데이터를 로드하는 경우 다음 방법 중 하나를 사용할 수 있습니다.

PARTITION_AND_LOAD
데이터가 해당 데이터베이스 파티션에서 동시에 분산(병렬 방식일 수 있음) 및 로드됩니다.

PARTITION_ONLY
데이터가 분산되고(병렬 방식일 수 있음) 로드하는 각 데이터베이스 파티션에서 지정된 위치의 파일에 작성됩니다. 각 파일은 여러 데이터베이스 파티션에서 데이터를 분산하는 방법을 지정하고 LOAD_ONLY 모드를 사용하여 데이터베이스로 파일을 로드할 수 있는 파티션 헤더를 포함합니다.

LOAD_ONLY
데이터는 여러 데이터베이스 파티션에서 이미 분산되었다고 가정합니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다.

LOAD_ONLY_VERIFY_PART
데이터는 여러 데이터베이스 파티션에서 이미 분산되었다고 가정합니다. 그러나

데이터 파일은 파티션 헤더를 포함하지 않습니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. 로드 조작 중 올바른 데이터베이스 파티션에 있는지 각 행을 검사합니다. 데이터베이스 파티션 위반을 포함하는 행은 dumpfile 파일 유형 수정자가 지정된 경우 덤프 파일에 배치됩니다. 그렇지 않으면 행을 버립니다. 데이터베이스 파티션 위반이 로드하는 특정 데이터베이스 파티션에 있으면 해당 데이터베이스 파티션에 대한 단일 경고가 로드 메시지 파일에 작성됩니다.

ANALYZE

모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다.

개념 및 용어

다음 용어는 다중 데이터베이스 파티션을 포함하는 파티션된 데이터베이스 환경에서 로드 유틸리티의 동작 및 조작을 논의할 때 사용됩니다.

- *코디네이터 파티션*은 로드 조작을 수행하기 위해 연결하는 데이터베이스 파티션입니다. PARTITION_AND_LOAD, PARTITION_ONLY 및 ANALYZE 모드의 경우 LOAD 명령의 CLIENT 옵션을 지정하지 않는 한, 데이터 파일은 이 데이터베이스 파티션에 있다고 가정합니다. CLIENT를 지정하면 로드할 데이터가 리모트로 연결된 클라이언트에 있음을 표시합니다.
- PARTITION_AND_LOAD, PARTITION_ONLY 및 ANALYZE 모드의 경우 *사전 파티셔닝 에이전트*는 사용자 데이터를 읽고 데이터를 분산하는 *파티셔닝 에이전트*에 라운드 로빈 방식으로 데이터를 분산합니다. 이 프로세스는 항상 코디네이터 파티션에서 수행됩니다. 로드 조작에 대해 데이터베이스 파티션당 최대 하나의 파티셔닝 에이전트가 허용됩니다.
- PARTITION_AND_LOAD, LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART 모드의 경우 *로드 에이전트*는 각 출력 데이터베이스 파티션에서 실행되며 해당 데이터베이스 파티션으로 데이터를 로드하는 작업을 조정합니다.
- *파일 에이전트*로 로드는 PARTITION_ONLY 로드 조작 중에 각 출력 데이터베이스 파티션에서 실행됩니다. 파티셔닝 에이전트에서 데이터를 받아 해당 데이터베이스 파티션의 파일에 작성합니다.
- SOURCEUSEREXIT 옵션에서는 로드 유틸리티가 사용자 정의 스크립트 또는 실행 파일(여기서 *User Exit*로 참조됨)을 실행할 수 있도록 하는 기능을 제공합니다.

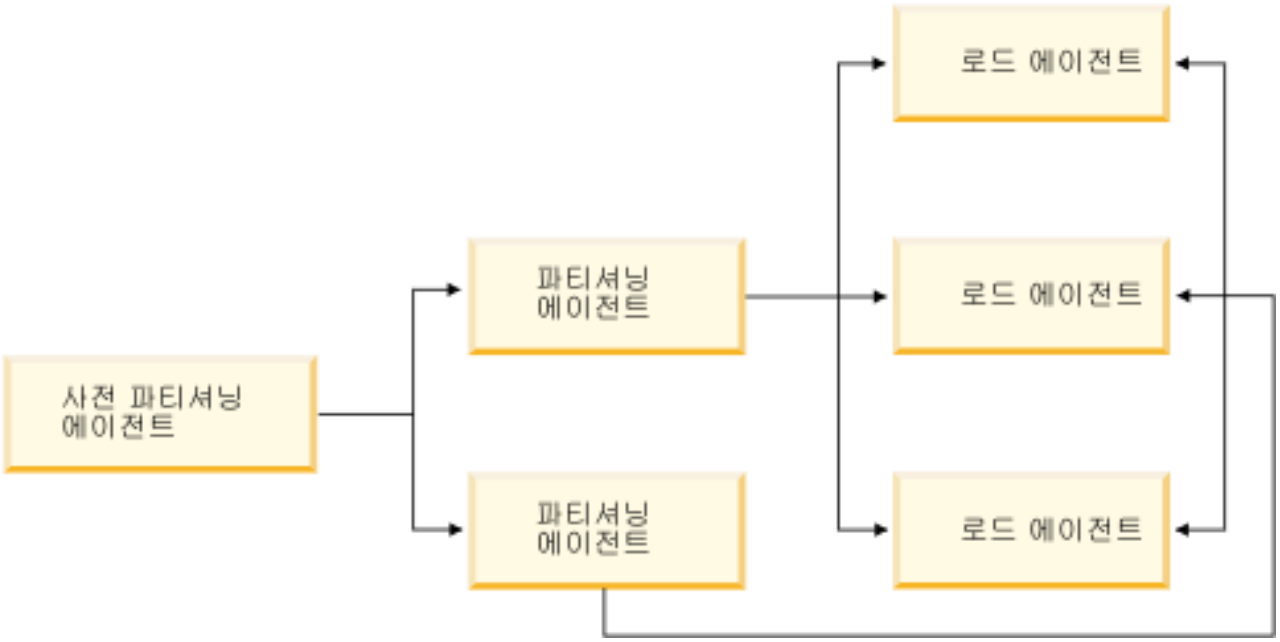


그림 41. 파티션된 데이터베이스 로드 개요. 사전 파티셔닝 에이전트가 소스 데이터를 읽으며 약 절반의 데이터가 두 파티셔닝 에이전트 중 각각에 전송되어 데이터를 분산하고 세 개의 데이터베이스 파티션 중 하나로 전송합니다. 각 데이터베이스 파티션의 로드 에이전트에서 데이터를 로드합니다.

파티션된 데이터베이스 환경에서 데이터 로드 - 힌트 및 추가 정보

다음은 다중 파티션 데이터베이스에서 테이블을 로드하기 전에 고려할 몇 가지 정보입니다.

- 적은 양의 데이터를 기반으로 유틸리티를 사용하여 로드 구성 옵션에 익숙해져야 합니다.
- 입력 데이터가 이미 정렬되었거나 일부 선택된 순서로 배치된 경우 로드 프로세스 중 해당 순서를 유지하려면 분산 시 하나의 데이터베이스 파티션만 사용해야 합니다. 병렬 분산으로는 데이터를 수신과 동일한 순서로 로드하도록 보장할 수 없습니다. 로드 유틸리티는 `anyorder` 수정자가 LOAD 명령에 지정되지 않은 경우 디폴트로 단일 파티셔닝 에이전트를 선택합니다.
- 대형 오브젝트(LOB)가 별도의 파일에서 로드되는 경우(즉, 로드 유틸리티를 통해 `lobsinfile` 수정자를 사용하는 경우) LOB 파일을 포함하는 모든 디렉토리는 로드 수행 시 모든 데이터베이스 파티션에 대한 읽기 액세스 권한이 있어야 합니다. LOB에 대한 작업을 수행할 때 LOAD *lob-path* 매개변수는 완전한 형식이어야 합니다.
- 로드 조작 시작 시 로드하는 일부 데이터베이스 파티션 또는 연관된 테이블 스페이스나 테이블이 오프라인임을 발견한 경우에도 ISOLATE_PART_ERRS 옵션을

SETUP_ERRS_ONLY 또는 SETUP_AND_LOAD_ERRS로 설정하여 다중 파티션 데이터베이스에서 작업을 강제로 계속 실행할 수 있습니다.

- STATUS_INTERVAL 로드 구성 옵션을 사용하여 다중 파티션 데이터베이스에서 실행하는 작업의 진행을 모니터합니다. 로드 조작 시 지정된 간격으로 사전 파티셔닝 에이전트에서 읽은 데이터 크기(MB)를 표시하는 메시지가 생성됩니다. 이 메시지는 사전 파티셔닝 에이전트 메시지 파일로 덤프됩니다. 로드 조작 중 이 파일의 컨텐츠를 보려면 코디네이터 파티션에 연결하고 목표 테이블에서 LOAD QUERY 명령을 실행합니다.
- 분산 프로세스에 참여하는 데이터베이스 파티션(PARTITIONING_DBPARTNUMS 옵션에서 정의함)이 로드하는 데이터베이스 파티션(OUTPUT_DBPARTNUMS 옵션에서 정의함)과 다른 경우 CPU 주기에 대한 경합이 적으므로 더 나은 성능이 예상됩니다. 다중 파티션 데이터베이스로 데이터를 로드하는 경우 분산 또는 로드 조작에 참여하지 않는 데이터베이스 파티션에서 로드 유틸리티를 호출합니다.
- LOAD 명령에서 MESSAGES 매개변수를 지정하면 로드 조작 종료 시 참조할 메시지 파일을 사전 파티셔닝, 파티셔닝 및 로드 에이전트에서 저장합니다. 로드 조작 중 이 파일의 컨텐츠를 보려면 원하는 데이터베이스 파티션에 연결하고 목표 테이블에서 LOAD QUERY 명령을 실행합니다.
- 로드 유틸리티는 통계를 수집할 하나의 출력 데이터베이스 파티션만 선택합니다. RUN_STAT_DBPARTNUM 데이터베이스 구성 옵션은 데이터베이스 파티션을 지정하는 데 사용할 수 있습니다.
- 다중 파티션 데이터베이스로 데이터를 로드하기 전에 디자인 어드바이저를 실행하여 각 테이블에 대한 최상의 파티션을 판별합니다. 자세한 정보는 *문제점 해결 및 데이터베이스 성능 조정*의 『디자인 어드바이저』를 참조하십시오.

문제점 해결

로드 유틸리티가 정지되면 다음을 수행할 수 있습니다.

- STATUS_INTERVAL 매개변수를 사용하여 다중 파티션 데이터베이스 로드 조작의 진행을 모니터합니다. 상태 간격 정보는 코디네이터 파티션의 사전 파티셔닝 에이전트 메시지 파일로 덤프됩니다.
- 파티셔닝 에이전트 메시지 파일에서 각 데이터베이스 파티션의 파티셔닝 에이전트 프로세스 상태를 확인합니다. 오류 없이 로드가 진행되는 경우 TRACE 옵션이 설정되면 이 메시지 파일에서 여러 레코드에 대한 추적 메시지가 있어야 합니다.
- 로드 메시지 파일에서 로드 오류 메시지가 있는지 확인합니다.

  주: 이 파일이 존재하려면 LOAD 명령의 MESSAGES 옵션을 지정해야 합니다.
- 로드 프로세스 중 하나에서 오류가 발생했음을 알리는 오류가 발견되면 현재 로드 조작을 인터럽트합니다.

파티션된 데이터베이스 환경에서 데이터 로드

로드 유틸리티를 사용하여 파티션된 데이터베이스 환경으로 데이터를 로드합니다.

다중 파티션 데이터베이스로 테이블을 로드하기 전에:

1. *svcename* 데이터베이스 관리 프로그램 구성 매개변수 및 **DB2COMM** 프로파일 레지스트리 변수를 올바르게 설정해야 합니다. 이는 로드 유틸리티가 사전 파티셔닝 에이전트에서 파티셔닝 에이전트로, 파티셔닝 에이전트에서 로드하는 데이터베이스 파티션으로 데이터를 전송하기 위해 TCP/IP를 사용하므로 중요합니다.
2. 로드 유틸리티를 호출하기 전에 데이터를 로드할 데이터베이스에 연결하거나 내재적으로 연결할 수 있어야 합니다. 로드 유틸리티는 COMMIT문을 실행하므로 로드 조작을 시작하기 전에 COMMIT 또는 ROLLBACK문을 실행하여 모든 트랜잭션을 완료하고 잠금을 해제해야 합니다. PARTITION_AND_LOAD, PARTITION_ONLY 또는 ANALYZE 모드를 사용하는 경우 로드할 데이터 파일은 이 데이터베이스 파티션에 있어야 합니다. 이때 다음은 예외 조건입니다.
   a. CLIENT 옵션이 지정된 경우. 이때 데이터는 클라이언트 머신에 있어야 합니다.
   b. 입력 소스 유형이 CURSOR인 경우. 이때 입력 파일은 없습니다.
3. 각 테이블에 대해 최상의 데이터베이스 파티션을 판별하도록 디자인 어드바이저를 실행합니다. 자세한 정보는 *문제점 해결 및 데이터베이스 성능 조정*의 『디자인 어드바이저』를 참조하십시오.

다음 제한사항은 다중 파티션 데이터베이스에서 로드 유틸리티를 사용하여 데이터를 로드하는 경우 적용됩니다.

- 로드 조작에 대한 입력 파일 위치는 테이프 디바이스일 수 없습니다.
- ANALYZE 모드를 사용하지 않는 한, ROWCOUNT 옵션은 지원되지 않습니다.
- 분산에 필요한 ID 컬럼이 목표 테이블에 있고 `identityoverride` 파일 유형 수정자가 지정되지 않은 경우 또는 다중 데이터베이스 파티션을 사용하여 데이터를 분산하고 로드하는 경우 LOAD 명령에서 0보다 큰 SAVECOUNT를 사용하는 작업은 지원되지 않습니다.
- ID 컬럼이 분산 키의 일부를 구성하는 경우 PARTITION_AND_LOAD 모드만 지원됩니다.
- LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART 모드는 LOAD 명령의 CLIENT 옵션과 함께 사용할 수 없습니다.
- LOAD_ONLY_VERIFY_PART 모드는 CURSOR 입력 소스 유형에서 사용할 수 없습니다.
- 분산 오류 분리 모드 LOAD_ERRS_ONLY 및 SETUP_AND_LOAD_ERRS는 LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션과 함께 사용할 수 없습니다.

- OUTPUT_DBPARTNUMS 및 PARTITIONING_DBPARTNUMS 옵션으로 지정된 데이터베이스 파티션이 오버랩되지 않는 경우 다중 로드 조작을 동시에 실행하여 동일한 테이블로 데이터를 로드할 수 있습니다. 예를 들어 데이터베이스 파티션 0 - 3까지 테이블이 정의된 경우 하나의 로드 조작이 데이터베이스 파티션 0 및 1로 데이터를 로드하는 동안, 두 번째 로드 조작은 데이터베이스 파티션 2 및 3으로 데이터를 로드할 수 있습니다.
- 컬럼 식별자 없는 ASCII(ASC) 및 컬럼 식별자가 있는 ASCII(DEL) 파일만 다중 데이터베이스 파티션에서 분산될 수 있습니다. PC/IXF 파일은 분산될 수 없지만 LOAD_ONLY_VERIFY_PART 모드에서 로드 조작을 사용하여 다중 데이터베이스 파티션에 분산된 테이블로 PC/IXF 파일을 로드할 수 있습니다.

다음 예에서는 LOAD 명령을 사용하여 다양한 유형의 로드 조작을 시작하는 방법을 보여줍니다. 다음 예에서 사용하는 데이터베이스에는 5개의 데이터베이스 파티션(0, 1, 2, 3, 4)이 있습니다. 각 데이터베이스 파티션에는 로컬 디렉토리 /db2/data/가 있습니다. 2개 테이블, TABLE1 및 TABLE2가 데이터베이스 파티션 0, 1, 3 및 4에 정의되어 있습니다. 클라이언트에서 로드하는 경우 데이터베이스 파티션에 포함되지 않은 리모트 클라이언트에 대한 액세스 권한이 사용자에게 부여됩니다.

서버 파티션에서 로드

분산 및 로드 예

이 시나리오에서는 데이터베이스 파티션에 연결됩니다. 이 데이터베이스 파티션이 TABLE1이 정의된 데이터베이스 파티션인지 여부와는 상관이 없습니다. 데이터 파일 load.del은 이 데이터베이스 파티션의 현재 작업 디렉토리에 있습니다. load.del에서 TABLE1이 정의된 모든 데이터베이스 파티션으로 데이터를 로드하려면 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

주: 이 예에서 파티션된 데이터베이스 환경의 모든 구성 매개변수에 대해 디폴트값이 사용됩니다. MODE 매개변수는 디폴트로 PARTITION_AND_LOAD로 설정되며 OUTPUT_DBPARTNUMS 옵션은 디폴트로 TABLE1이 정의된 모든 데이터베이스 파티션으로 설정됩니다. PARTITIONING_DBPARTNUMS는 디폴트로 아무것도 지정하지 않았을 때 데이터베이스 파티션 선택 시 LOAD 명령 규칙에 따라 선택된 데이터베이스 파티션으로 설정됩니다.

데이터베이스 파티션 3 및 4에서 데이터가 분산되는 로드 조작을 수행하려면 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                    PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

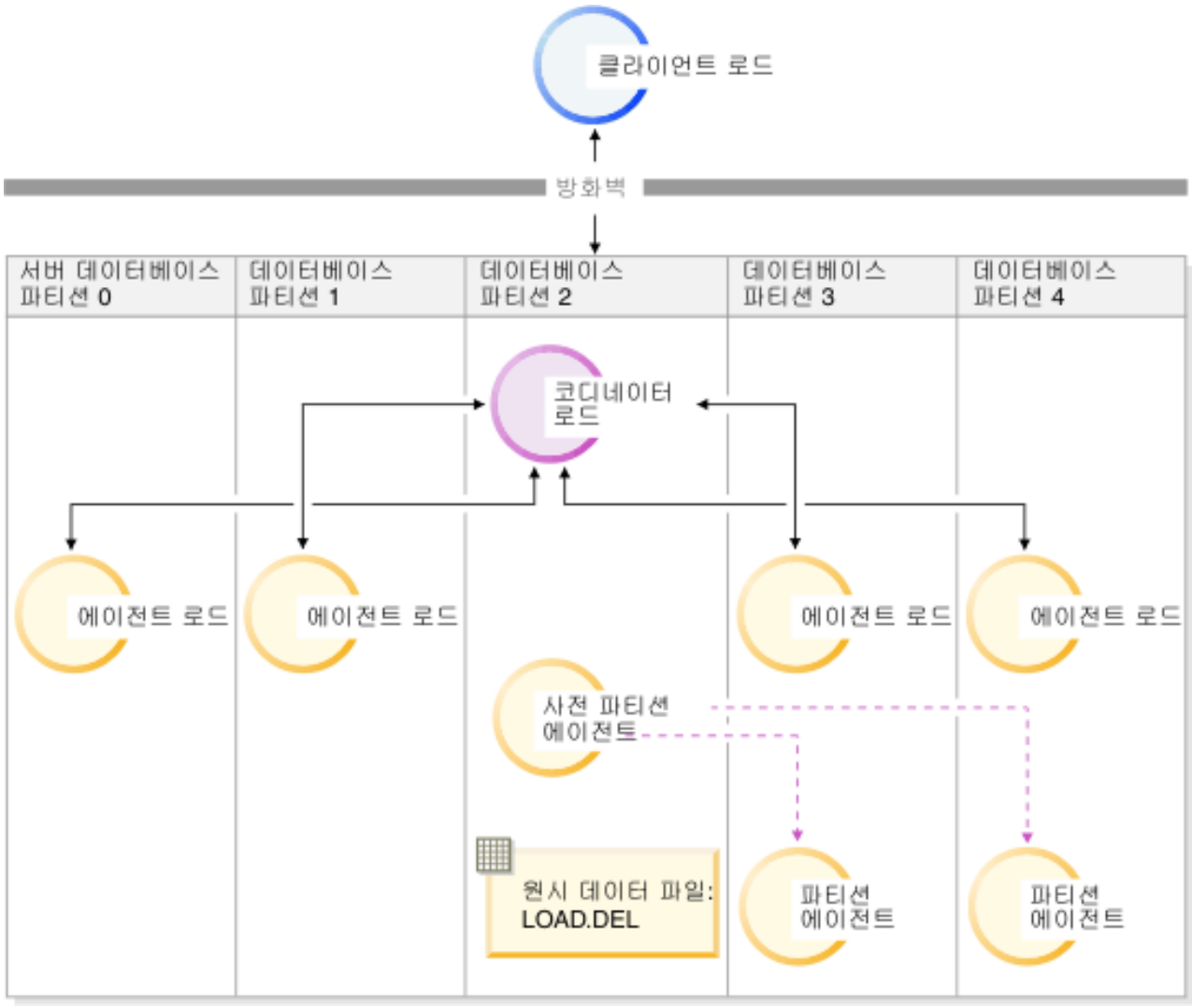


그림 42. 데이터베이스 파티션 3 및 4로 데이터 로드. 이 그림에서는 이전 명령을 실행할 때 나타나는 동작을 보여줍니다. 데이터는 데이터베이스 파티션 3 및 4로 로드됩니다.

분산만 수행하는 예

이 시나리오에서는 데이터베이스 파티션에 연결됩니다. 이 데이터베이스 파티션이 TABLE1이 정의된 데이터베이스 파티션인지 여부와는 상관이 없습니다. 데이터 파일 `load.del`은 이 데이터베이스 파티션의 현재 작업 디렉토리에 있습니다. 데이터베이스 파티션 3 및 4를 사용하여 `load.del`을 TABLE1이 정의된 모든 데이터베이스 파티션에 분산(로드가 아님)하려면 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                               PART_FILE_LOCATION /db2/data
                                               PARTITIONING_DBPARTNUMS (3,4)
```

그러면 `load.del.xxx` 파일이 각 데이터베이스 파티션의 `/db2/data` 디렉토리에 저장됩니다. 여기서 `xxx`는 데이터베이스 파티션 번호의 3자리 표시입니다.

데이터베이스 파티션 0에서 실행하는 파티셔닝 에이전트 하나만 사용하여 load.del 파일을 데이터베이스 파티션 1 및 3에 분산하려면(PARTITIONING_DBPARTNUMS의 디폴트값임) 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
 PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                          PART_FILE_LOCATION /db2/data
                                          OUTPUT_DBPARTNUMS (1,3)
```

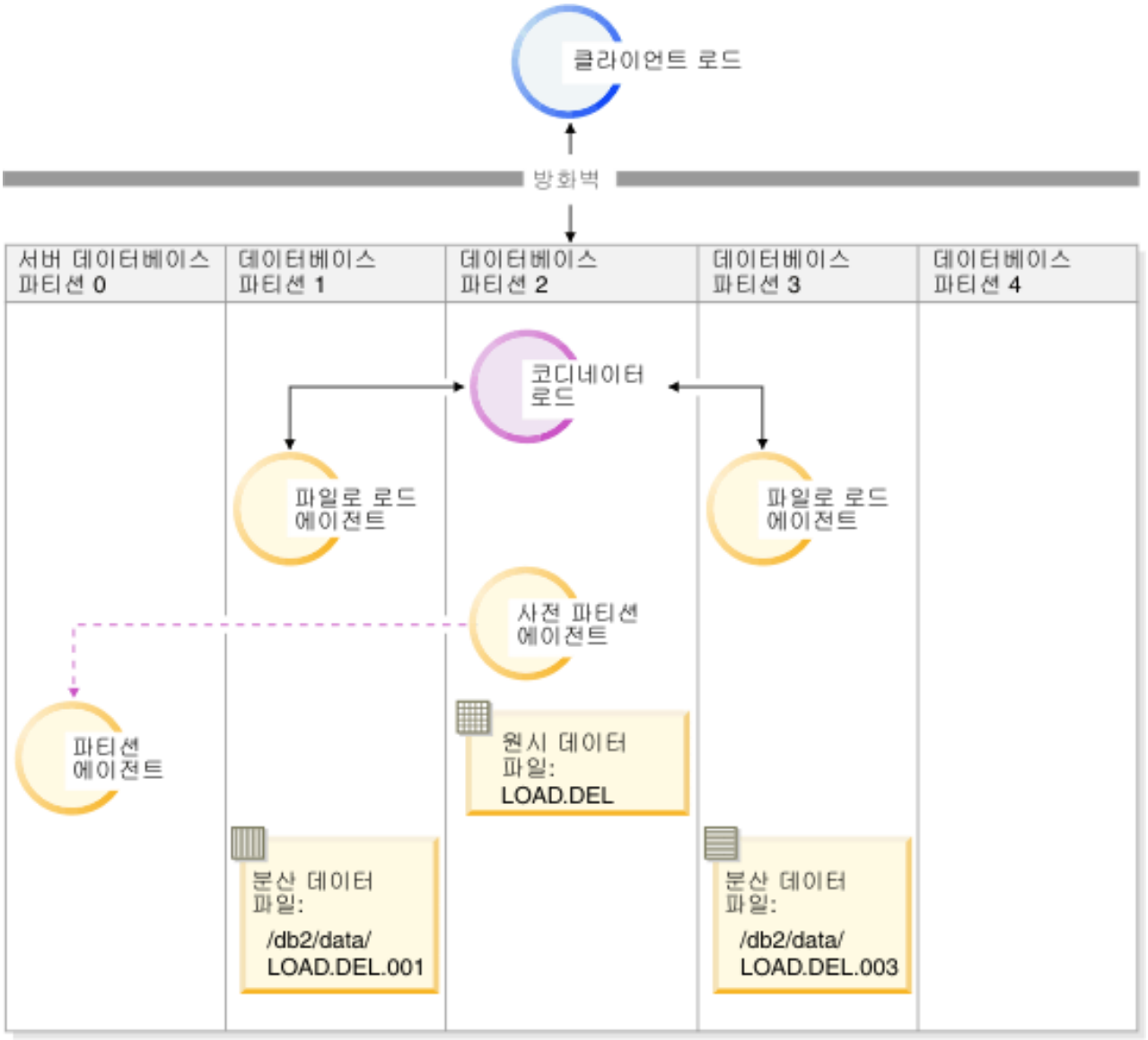


그림 43. 하나의 파티셔닝 에이전트를 사용하여 데이터베이스 파티션 1 및 3으로 데이터 로드. 이 그림에서는 이전 명령을 실행할 때 결과로 나타나는 동작을 보여줍니다. 데이터는 데이터베이스 파티션 0에서 실행하는 하나의 파티셔닝 에이전트를 사용하여 데이터베이스 파티션 1 및 3으로 로드됩니다.

로드만 수행하는 예

이미 PARTITION_ONLY 모드로 로드 조작을 수행했으며 로드하는 각 데이터베이스 파티션의 /db2/data 디렉토리에 있는 파티션된 파일을 TABLE1이 정의된 모든 데이터베이스 파티션으로 로드하려는 경우 다음 명령을 실행합니다.

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
                                    PART_FILE_LOCATION /db2/data
```

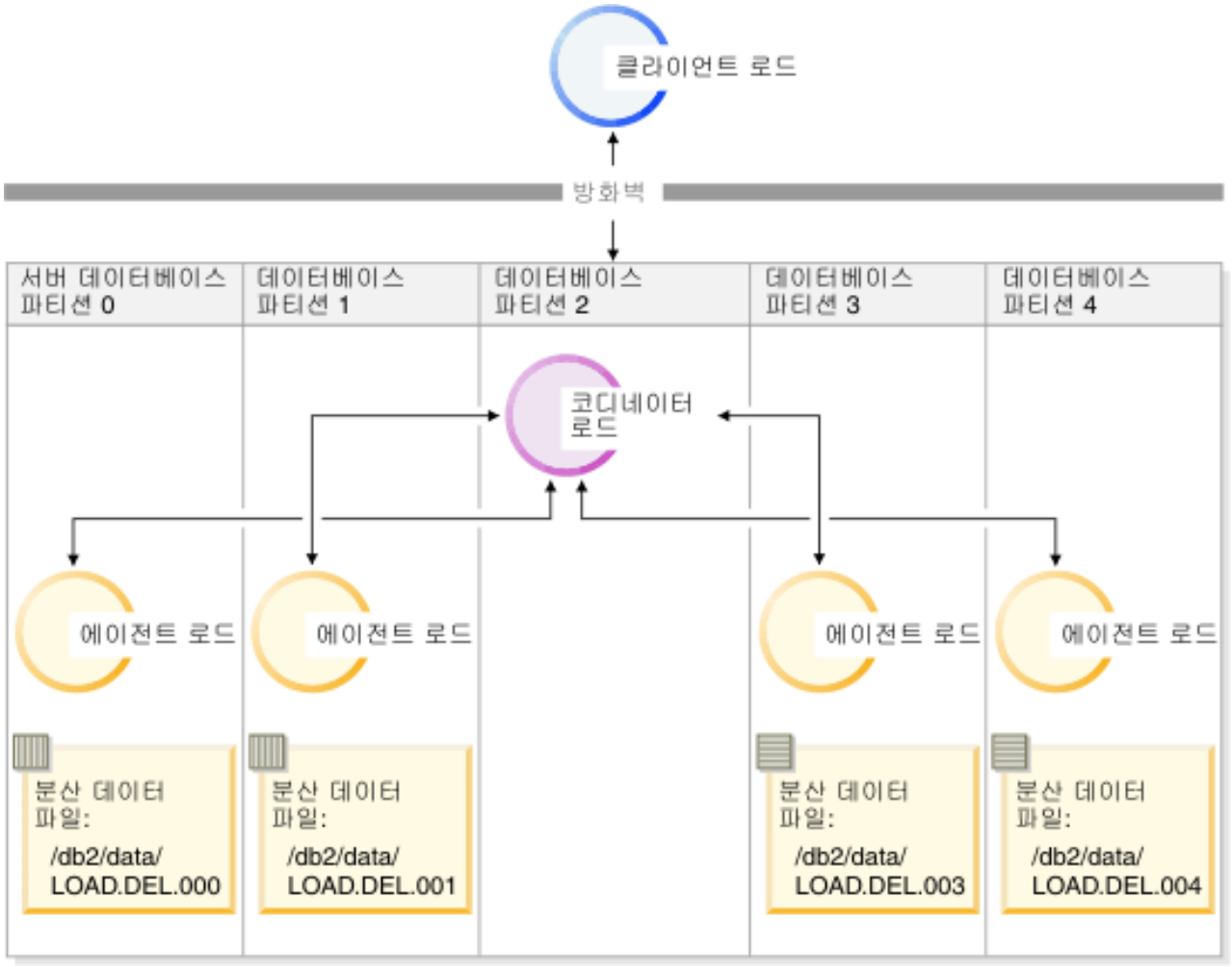


그림 44. 특정 테이블이 정의된 모든 데이터베이스 파티션으로 데이터 로드. 이 그림에서는 이전 명령을 실행할 때 나타나는 동작을 보여줍니다. 분산된 데이터는 TABLE1이 정의된 모든 데이터베이스 파티션으로 로드됩니다.

데이터베이스 파티션 4로만 로드하려면 다음 명령을 실행하십시오.

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
                                    PART_FILE_LOCATION /db2/data
                                    OUTPUT_DBPARTNUMS (4)
```

분산 맵 헤더 없이 사전 분산된 파일 로드

LOAD 명령을 사용하여 분산 헤더 없이 여러 데이터베이스 파티션으로 직접 데이터 파일을 로드할 수 있습니다. 데이터 파일이 TABLE1이 정의된 각 데이터베이스 파티션의 /db2/data 디렉토리에 있고 이름이 load.del.xxx(여기서 xxx는 데이터베이스 파티션 번호)이면 다음 명령을 실행하여 파일을 로드할 수 있습니다.

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                          REPLACE INTO TABLE1
                          PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                PART_FILE_LOCATION /db2/data
```

데이터베이스 파티션 1로만 데이터를 로드하려면 다음 명령을 실행하십시오.

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                          REPLACE INTO TABLE1
                          PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                PART_FILE_LOCATION /db2/data
                                                OUTPUT_DBPARTNUMS (1)
```

주: 로드된 데이터베이스 파티션에 속하지 않은 행은 거부되고 덤프 파일이 지정된 경우 이 파일에 배치됩니다.

리모트 클라이언트에서 다중 파티션 데이터베이스로 로드

리모트 클라이언트에 있는 파일에서 다중 파티션 데이터베이스로 데이터를 로드하려면 데이터 파일이 서버 파티션에 없음을 표시하도록 LOAD 명령의 CLIENT 옵션을 지정해야 합니다. 예를 들어, 다음과 같습니다.

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

주: LOAD_ONLY 또는 LOAD_ONLY_VERIFY_PART 모드는 CLIENT 옵션과 함께 사용할 수 없습니다.

cursor에서 로드

단일 파티션 데이터베이스에서는 cursor에서 다중 파티션 데이터베이스로 로드할 수 있습니다. 이 예에서 PARTITION_ONLY 및 LOAD_ONLY 모드의 경우 PART_FILE_LOCATION 옵션에서는 완전한 파일 이름을 지정해야 합니다. 이 이름은 각 출력 데이터베이스 파티션에서 로드 또는 작성된 분산 파일의 완전한 기본 파일 이름입니다. 목표 테이블에 LOB 컬럼이 있는 경우 지정된 기본 이름으로 다중 파일을 작성할 수 있습니다.

다음에 TABLE2로 로드하는 경우 SELECT * FROM TABLE1문의 응답 세트에 있는 모든 행을 이름이 /db2/data/select.out.xxx(여기서 xxx는 데이터베이스 파티션 번호임)인 각 데이터베이스 파티션으로 분산하려면 다음 명령을 실행합니다.

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                       PARTITIONED DB CONFIG MODE PARTITION_ONLY
                       PART_FILE_LOCATION /db2/data/select.out
```

위 조작으로 생성된 데이터 파일은 다음 LOAD 명령을 실행하여 로드할 수 있습니다.

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                        PARTITIONED CB CONFIG MODE LOAD_ONLY
                        PART_FILE_LOCATION /db2/data/select.out
```

LOAD QUERY 명령을 사용하여 파티션된 데이터베이스 환경에서 로드 조작 모니터링

파티션된 데이터베이스 환경에서 로드 조작 중 로드 프로세스를 실행하는 데이터베이스 파티션에서 일부 로드 프로세스로 메시지 파일이 작성됩니다.

로드 조작 실행 중 생성되는 모든 정보, 경고 및 오류 메시지는 메시지 파일에 저장됩니다. 사용자가 볼 수 있는 메시지 파일을 생성하는 로드 프로세스는 로드 에이전트, 사전 파티셔닝 에이전트 및 파티셔닝 에이전트입니다. 메시지 파일의 컨텐츠는 로드 조작을 완료한 후에만 볼 수 있습니다.

로드 조작 중 개별 데이터베이스 파티션에 연결하고 목표 테이블에서 LOAD QUERY 명령을 실행할 수 있습니다. CLP에서 명령을 실행한 경우 이 명령은 LOAD QUERY 명령에 지정된 테이블의 해당 데이터베이스 파티션에 현재 있는 모든 메시지 파일의 컨텐츠를 표시합니다.

예를 들어 테이블 TABLE1은 데이터베이스 WSDB의 데이터베이스 파티션 0 - 3에 정의되어 있습니다. 데이터베이스 파티션 0에 연결한 상태에서 다음 LOAD 명령을 실행하십시오.

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

이 명령은 데이터베이스 파티션 0, 1, 2, 3에서 실행하는 로드 에이전트, 데이터베이스 파티션 1에서 실행하는 파티셔닝 에이전트 및 데이터베이스 파티션 0에서 실행하는 사전 파티셔닝 에이전트를 포함하는 로드 조작을 시작합니다.

데이터베이스 파티션 0에는 사전 파티셔닝 에이전트에 대한 하나의 메시지 파일과 해당 데이터베이스 파티션의 로드 에이전트에 대한 메시지 파일 하나가 포함되어 있습니다. 이 파일의 컨텐츠를 동시에 보려면 새 세션을 시작하고 CLP에서 다음 명령을 실행하십시오.

```
set client connect_node 0
connect to wsdb
load query table table1
```

데이터베이스 파티션 1에는 로드 에이전트에 대한 하나의 파일과 파티셔닝 에이전트에 대한 하나의 파일이 포함되어 있습니다. 이 파일의 컨텐츠를 보려면 새 세션을 시작하고 CLP에서 다음 명령을 실행하십시오.

```
set client connect_node 1
connect to wsdb
load query table table1
```

주: STATUS_INTERVAL 로드 구성 옵션으로 생성된 메시지는 사전 파티셔닝 에이전트 메시지 파일에 표시됩니다. 로드 조작 중 이 메시지를 보려면 코디네이터 파티션에 연결하고 LOAD QUERY 명령을 실행해야 합니다.

메시지 파일의 컨텐츠 저장

db2Load API를 통해 로드 조작이 시작되면 메시지 옵션(piLocalMsgFileName)을 지정해야 합니다. 메시지 파일은 서버에서 클라이언트로 가져와 사용자가 볼 수 있도록 저장됩니다.

CLP로 시작된 다중 파티션 데이터베이스 로드 조작의 경우 메시지 파일은 콘솔에 표시되지 않으며 보유되지 않습니다. 다중 파티션 데이터베이스 로드가 완료된 후 이러한 파일의 컨텐츠를 보거나 저장하려면 LOAD 명령의 MESSAGES 옵션을 지정해야 합니다. 이 옵션을 사용하면 로드 조작을 완료한 후 각 데이터베이스 파티션의 메시지 파일이 클라이언트 머신으로 전송되고 MESSAGES 옵션에 표시된 기본 이름을 사용하여 파일에 저장됩니다. 다중 파티션 데이터베이스 로드 조작의 경우 로드 프로세스에 대응하여 생성된 파일 이름이 아래 나열되어 있습니다.

프로세스 유형	파일 이름
로드 에이전트	<message-file-name>.LOAD.<dbpartition-number>
파티셔닝 에이전트	<message-file-name>.PART.<dbpartition-number>
사전 파티셔닝 에이전트	<message-file-name>.PREP.<dbpartition-number>

예를 들어 MESSAGES 옵션에서 `/wsdb/messages/load`를 지정하면 데이터베이스 파티션 2의 로드 에이전트 메시지 파일은 `/wsdb/messages/load.LOAD.002`입니다.

주: CLP에서 시작된 다중 파티션 데이터베이스 로드 조작에서는 MESSAGES 옵션을 사용하는 것이 좋습니다.

파티션된 데이터베이스 환경에서 로드 조작 다시 시작, 재시작 또는 종료

파티션된 데이터베이스 환경에서 실패한 로드 조작 이후에 수행해야 하는 단계는 실패한 시점에 따라 달라집니다.

다중 파티션 데이터베이스에서 로드 프로세스는 다음 두 단계로 구성됩니다.

1. *설정 단계*에서는 출력 데이터베이스 파티션의 테이블 잠금과 같은 데이터베이스 파티션 레벨의 자원이 확보됩니다.

일반적으로 설정 단계에서 실패하면 재시작 및 종료 조작은 필요하지 않습니다. 실패한 로드 조작에 지정된 오류 분리 모드에 따라 수행해야 하는 작업이 달라집니다.

로드 조작에서 설정 단계 오류를 분리하지 않도록 지정한 경우 전체 로드 조작이 취소되고 각 데이터베이스 파티션의 테이블은 로드 조작 전의 상태로 롤백됩니다.

로드 조작에서 설정 단계 오류를 분리하도록 지정한 경우 설정 단계가 성공하면 데이터베이스 파티션에서 로드 조작이 계속되지만 실패한 각 데이터베이스 파티션의 테이블은 로드 조작 전의 상태로 롤백됩니다. 즉, 일부 파티션이 설정 단계에서 실패하고 또 다른 파티션이 로드 단계에서 실패한 경우 단일 로드 조작이 서로 다른 단계에서 실패할 수 있음을 의미합니다.

2. 로드 *단계*에서는 데이터를 형식화하여 데이터베이스 파티션의 테이블로 로드합니다.

   다중 파티션 데이터베이스 로드 조작의 로드 단계 중 하나 이상의 데이터베이스 파티션에서 로드 조작에 실패하면 로드 RESTART 또는 TERMINATE 명령을 실행해야 합니다. 다중 파티션 데이터베이스에서의 데이터 로드는 단일 트랜잭션으로 수행되므로 이 작업이 필요합니다.

   로드 실패 원인이 되는 문제점을 수정할 수 있으면 로드 RESTART를 선택해야 합니다. 로드 재시작 조작을 시작하면 모든 데이터베이스 파티션에서 중단점 지점부터 로드 조작이 계속되므로 시간을 절약할 수 있습니다.

   초기 로드 조작 전의 상태로 테이블을 되돌리려면 로드 TERMINATE를 선택해야 합니다.

프로시저:

로드 실패 시점 판별

파티션된 환경에서 로드 조작에 실패한 경우 처음 수행해야 하는 작업은 실패한 파티션 및 각 파티션에서 실패한 단계를 판별하는 것입니다. 파티션 요약을 보면 알 수 있습니다. CLP에서 load 명령을 실행한 경우 로드 끝에 파티션 요약이 표시됩니다(아래 예 참조). db2Load API에서 load 명령을 실행한 경우 db2PartLoadOut 구조의 poAgentInfoList 필드에 파티션 요약이 포함됩니다.

지정된 파티션에서 ″Agent Type″이 ″LOAD″인 항목이 있으면 해당 파티션은 로드 단계에 도달한 것입니다. 그렇지 않으면 설정 단계에서 실패합니다. 음수 SQL 코드는 실패했음을 의미합니다. 다음 예에서 로드는 로드 단계 중 파티션 1에서 실패했습니다.

에이전트 유형	노드	SQL 코드	결과
LOAD	000	+00000000	성공
LOAD	001	-00000289	오류. 재시작 필요

```
LOAD            002       +00000000     성공
_____________________________________________________________________
LOAD            003       +00000000     성공
.
.
.
```

실패한 로드 다시 시작, 재시작 또는 종료

SETUP_ERRS_ONLY 또는 SETUP_AND_LOAD_ERRS를 지정하여 ISOLATE_PART_ERRS 옵션으로 수행하는 로드만 설정 단계 중에 실패합니다. 이 단계 중에 하나 이상의 출력 데이터베이스 파티션에서 로드에 실패하면 LOAD REPLACE 또는 LOAD INSERT 명령을 실행할 수 있습니다. OUTPUT_DBPARTNUMS 옵션을 사용하여 실패한 해당 데이터베이스 파티션만 지정합니다.

로드 단계 중 하나 이상의 출력 데이터베이스 파티션에서 로드에 실패하면 로드 RESTART 또는 로드 TERMINATE 명령을 실행합니다.

설정 단계 중 하나 이상의 출력 데이터베이스 파티션에서, 그리고 로드 단계 중 하나 이상의 출력 데이터베이스 파티션에서 로드에 실패하면 이전에 설명한 대로 로드 단계 중 실패한 로드 하나 및 설정 단계 중 실패한 로드 하나와 같은 두 개의 로드 조작을 수행하여 실패한 로드를 다시 시작해야 합니다. 이러한 유형의 실패한 로드 조작을 효과적으로 실행 취소하려면 로드 TERMINATE 명령을 실행합니다. 그러나 명령을 실행한 후에는 모든 파티션을 고려해야 합니다. 설정 단계 중 실패한 파티션에서 테이블은 변경되지 않고 로드 단계 중 실패한 파티션에서 모든 변경 사항은 실행 취소되기 때문입니다.

예를 들어 TABLE1은 데이터베이스 WSDB의 데이터베이스 파티션 0 - 3에 정의되어 있습니다. 다음 명령이 실행됩니다.

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

설정 단계 중 출력 데이터베이스 파티션 1에서 실패했습니다. 설정 단계 오류는 분리되므로 로드 조작은 계속되지만 로드 단계 중 파티션 3에서 실패했습니다. 로드 조작을 다시 시작하려면 다음 명령을 실행합니다.

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

주: 로드 재시작 조작의 경우 LOAD RESTART 명령에 지정된 옵션은 처리되므로 원래 LOAD 명령에 지정된 옵션과 동일해야 합니다.

MODE X

다중 파티션 데이터베이스 로드 중 로드 조작에서 사용하는 모드를 지정합니다. PARTITION_AND_LOAD가 디폴트입니다. 가능한 값은 다음과 같습니다.

- PARTITION_AND_LOAD. 데이터가 해당 데이터베이스 파티션에서 동시에 분산(병렬 방식일 수 있음) 및 로드됩니다.
- PARTITION_ONLY. 데이터가 분산되고(병렬 방식일 수 있음) 로드하는 각 데이터베이스 파티션에서 지정된 위치의 파일에 작성됩니다. CURSOR 이외의 파일 유형에서 각 데이터베이스 파티션의 출력 파일 이름 형식은 `filename.xxx`입니다. 여기서 `filename`은 LOAD 명령에 지정된 입력 파일 이름이고 `xxx`는 3자리 데이터베이스 파티션 번호입니다. CURSOR 파일 유형의 경우 각 데이터베이스 파티션에서 출력 파일 이름은 PART_FILE_LOCATION 옵션으로 판별됩니다. 각 데이터베이스 파티션에서 분산 위치를 지정하는 방법에 대한 세부사항은 PART_FILE_LOCATION 옵션을 참조하십시오.

  주:

  1. 이 모드는 CLI 로드 조작에서 사용할 수 없습니다.
  2. 분산에 필요한 ID 컬럼이 테이블에 포함된 경우 `identityoverride` 파일 유형 수정자를 지정하지 않는 한 이 모드는 지원되지 않습니다.
  3. CURSOR 파일 유형에 대해 생성된 분산 파일은 DB2 릴리스 사이에서 호환 가능하지 않습니다. 즉, 이전 릴리스에서 생성된 파일 유형 CURSOR의 분산 파일은 LOAD_ONLY 모드를 사용하여 로드할 수 없습니다. 마찬가지로 현재 릴리스에서 생성된 파일 유형 CURSOR의 분산 파일은 LOAD_ONLY 모드를 사용하여 추후 릴리스에서 로드할 수 없습니다.
- LOAD_ONLY. 데이터는 이미 분산되었다고 가정합니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. CURSOR 이외의 파일 유형에서 각 데이터베이스 파티션의 입력 파일 이름 형식은 `filename.xxx`입니다. 여기서 `filename`은 LOAD 명령에 지정된 파일 이름이고 `xxx`는 3자리 데이터베이스 파티션 번호입니다. CURSOR 파일 유형의 경우 각 데이터베이스 파티션에서 입력 파일 이름은 PART_FILE_LOCATION 옵션으로 판별됩니다. 각 데이터베이스 파티션에서 분산 위치를 지정하는 방법에 대한 세부사항은 PART_FILE_LOCATION 옵션을 참조하십시오.

  주:

1. 이 모드는 CLI 로드 조작 또는 LOAD 명령의 CLIENT 옵션이 지정된 경우에 사용할 수 없습니다.
2. 분산에 필요한 ID 컬럼이 테이블에 포함된 경우 identityoverride 파일 유형 수정자를 지정하지 않는 한 이 모드는 지원되지 않습니다.

- LOAD_ONLY_VERIFY_PART. 데이터는 이미 분산되었다고 가정합니다. 그러나 데이터 파일은 파티션 헤더를 포함하지 않습니다. 분산 프로세스는 건너뛰고 데이터는 해당 데이터베이스 파티션에 동시에 로드됩니다. 로드 조작 중 올바른 데이터베이스 파티션에 있는지 각 행을 검사합니다. 데이터베이스 파티션 위반을 포함하는 행은 dumpfile 파일 유형 수정자가 지정된 경우 덤프 파일에 배치됩니다. 그렇지 않으면 행을 버립니다. 데이터베이스 파티션 위반이 로드하는 특정 데이터베이스 파티션에 있으면 해당 데이터베이스 파티션에 대한 단일 경고가 로드 메시지 파일에 작성됩니다. 각 데이터베이스 파티션의 입력 파일 이름 형식은 filename.xxx입니다. 여기서 filename은 LOAD 명령에 지정된 파일 이름이고 xxx는 3자리 데이터베이스 파티션 번호입니다. 각 데이터베이스 파티션에서 분산 위치를 지정하는 방법에 대한 세부사항은 PART_FILE_LOCATION 옵션을 참조하십시오.

  주:

  1. 이 모드는 CLI 로드 조작 또는 LOAD 명령의 CLIENT 옵션이 지정된 경우에 사용할 수 없습니다.
  2. 분산에 필요한 ID 컬럼이 테이블에 포함된 경우 identityoverride 파일 유형 수정자를 지정하지 않는 한 이 모드는 지원되지 않습니다.

- ANALYZE. 모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다.

PART_FILE_LOCATION X

PARTITION_ONLY, LOAD_ONLY 및 LOAD_ONLY_VERIFY_PART 모드에서 이 매개변수를 사용하여 분산된 파일의 위치를 지정할 수 있습니다. 이 위치는 OUTPUT_DBPARTNUMS 옵션에서 지정한 각 데이터베이스 파티션에 있어야 합니다. 지정한 위치가 상대 경로 이름인 경우 경로는 현재 디렉토리에 추가되어 분산된 파일의 위치를 작성합니다.

CURSOR 파일 유형의 경우 이 옵션을 지정하고 위치는 완전한 파일 이름을 참조해야 합니다. 이 이름은 PARTITION_ONLY 모드에서 각 출력 데이터베이스 파티션에 작성된 분산 파일의 완전한 기본 파일 이름 또는 LOAD_ONLY 모드에서 각 데이터베이스 파티션에서 읽을 파일 위치입니다.
PARTITION_ONLY 모드를 사용하면 목표 테이블에 LOB 컬럼이 있는 경우 지정된 기본 이름으로 다중 파일을 작성할 수 있습니다.

CURSOR 이외의 파일 유형인 경우 이 옵션을 지정하지 않으면 분산 파일에 대해 현재 디렉토리가 사용됩니다.

OUTPUT_DBPARTNUMS X

X는 데이터베이스 파티션 번호 목록입니다. 데이터베이스 파티션 번호는 로드 조작을 수행할 데이터베이스 파티션을 나타냅니다. 데이터베이스 파티션 번호는 테이블이 정의된 데이터베이스 파티션의 서브세트여야 합니다. 모든 데이터베이스 파티션은 디폴트로 선택됩니다. 목록은 괄호로 묶어야 하며 목록의 항목은 쉼표로 구분되어야 합니다. 범위는 허용됩니다(예: 0, 2에서 10, 15).

PARTITIONING_DBPARTNUMS X

X는 분산 프로세스에서 사용된 데이터베이스 파티션 번호 목록입니다. 목록은 괄호로 묶어야 하며 목록의 항목은 쉼표로 구분되어야 합니다. 범위는 허용됩니다(예: 0, 2에서 10, 15). 분산 프로세스에 지정된 데이터베이스 파티션은 로드할 데이터베이스 파티션과 다를 수 있습니다.

PARTITIONING_DBPARTNUMS가 지정되지 않은 경우 로드 유틸리티는 최적의 성능을 얻기 위해 사용할 데이터베이스 파티션 및 필요한 데이터베이스 파티션 수를 판별합니다.

anyorder 파일 유형 수정자가 LOAD 명령에 지정되지 않은 경우 로드 세션에서는 하나의 파티셔닝 에이전트만 사용됩니다. 더 나아가 OUTPUT_DBPARTNUMS 옵션에 하나의 데이터베이스 파티션만 지정된 경우 또는 로드 조작의 코디네이터 파티션이 OUTPUT_DBPARTNUMS의 요소가 아닌 경우 분산 프로세스에서 로드 조작의 코디네이터 파티션이 사용됩니다. 그렇지 않으면 OUTPUT_DBPARTNUMS의 첫 번째 데이터베이스 파티션(코디네이터 파티션이 아님)이 분산 프로세스에 사용됩니다.

anyorder 파일 유형 수정자가 지정되면 분산 프로세스에 사용되는 데이터베이스 파티션 수는 다음과 같이 판별됩니다. (OUTPUT_DBPARTNUMS의 파티션 수/4 + 1).

MAX_NUM_PART_AGENTS X

로드 세션에서 사용할 최대 파티셔닝 에이전트 수를 지정합니다. 디폴트값은 25입니다.

ISOLATE_PART_ERRS X

개별 데이터베이스 파티션에서 발생하는 오류에 로드 조작이 대응하는 방식을 표시합니다. LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션이 지정되지 않는 한(이 경우 디폴트는 NO_ISOLATION임) 디폴트는 LOAD_ERRS_ONLY입니다. 가능한 값은 다음과 같습니다.

- SETUP_ERRS_ONLY. 설정 중 데이터베이스 파티션에서 발생하는 오류(예: 데이터베이스 파티션에 액세스하는 중 발생하는 문제점 또는 데이터베이스 파티션의 테이블 또는 테이블 스페이스에 액세스하는 중 발생하는 문제점)로 실패한 데이터베이스 파티션에서 로드 조작이 중지되지만 나머지 데이터베이스 파티션에서는 계속 진행됩니다. 데이터를 로드하는 중 데이터베이스 파티션에서 발생하는 오류로 전체 조작에 실패합니다.

- LOAD_ERRS_ONLY. 설정 중 데이터베이스 파티션에서 발생한 오류로 전체 로드 조작에 실패할 수 있습니다. 데이터를 로드하는 중 오류가 발생하면 오류가 발생한 데이터베이스 파티션에서 로드 조작이 중지됩니다. 실패가 발생하거나 모든 데이터를 로드할 때까지 나머지 데이터베이스 파티션에서 로드 조작이 계속됩니다. 새로 로드된 데이터는 로드 재시작 조작을 수행하여 성공적으로 완료할 때까지 보이지 않습니다.

  주: 이 모드는 LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션이 지정된 경우 사용할 수 없습니다.
- SETUP_AND_LOAD_ERRS. 이 모드에서는 설정 또는 데이터 로드 중 발생하는 데이터베이스 파티션 레벨의 오류로 관련된 데이터베이스 파티션에서만 처리가 중지됩니다. LOAD_ERRS_ONLY 모드와 마찬가지로 데이터를 로드하는 중 파티션 오류가 발생하면 새로 로드된 데이터는 로드 재시작 조작을 수행하여 성공적으로 완료할 때까지 보이지 않습니다.

  주: 이 모드는 LOAD 명령의 ALLOW READ ACCESS 및 COPY YES 옵션이 지정된 경우 사용할 수 없습니다.
- NO_ISOLATION. 로드 조작 중 오류가 발생하면 로드 조작에 실패합니다.

STATUS_INTERVAL X

X는 읽은 데이터의 볼륨을 알리는 간격을 표시합니다. 측정 단위는 메가바이트(MB)입니다. 디폴트값은 100MB입니다. 올바른 값은 1에서 4000 사이의 정수입니다.

PORT_RANGE X

X는 내부 통신에서 소켓을 작성하는 데 사용하는 TCP 포트 범위를 표시합니다. 디폴트 범위는 6000에서 6063 사이입니다. 호출 시 정의되는 경우 **DB2ATLD_PORTS** 레지스트리 변수의 값이 PORT_RANGE 로드 구성 옵션의 값을 교체합니다. **DB2ATLD_PORTS** 레지스트리 변수의 경우 범위는 다음 형식으로 제공해야 합니다.

```
<lower-port-number:higher-port-number>
```

CLP에서 형식은 다음과 같습니다.

```
( lower-port-number, higher-port-number )
```

CHECK_TRUNCATION

입출력 시 프로그램에서 데이터 레코드가 절단되었는지 확인하도록 지정합니다. 디폴트 동작은 입출력 시 데이터가 절단되었는지 확인하지 않는 것입니다.

MAP_FILE_INPUT X

X는 분산 맵의 입력 파일 이름을 지정합니다. 이 매개변수는 분산 맵이 사용자 정의된 경우 사용자 정의된 분산 맵을 포함하는 파일을 가리킬 때 지정해야 합니다. 사용자 정의된 분산 맵은 db2gpmap 프로그램을 사용하여 데이터베이스

시스템 카탈로그 테이블에서 맵을 추출하거나 LOAD 명령의 ANALYZE 모드를 통해 최적의 맵을 생성하여 작성할 수 있습니다. ANALYZE 모드를 사용하여 생성된 맵은 로드 조작을 계속 진행하려면 데이터베이스의 각 데이터베이스 파티션으로 이동되어야 합니다.

MAP_FILE_OUTPUT X

X는 분산 맵의 출력 파일 이름을 나타냅니다. 출력 파일은 파티셔닝을 수행할 때 데이터베이스 파티션이 데이터베이스 파티션 그룹에 참여한다고 가정하고 LOAD 명령을 실행하여 데이터베이스 파티션에서 작성됩니다. PARTITIONING_DBPARTNUMS에서 정의한 대로 참여하지 않는 데이터베이스 파티션에서 LOAD 명령을 호출하면 출력 파일은 PARTITIONING_DBPARTNUMS 매개변수로 정의된 첫 번째 데이터베이스 파티션에서 작성됩니다. 다음의 파티션된 데이터베이스 환경 설정을 고려합니다.

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

serv3에서 다음 LOAD 명령을 실행하면 serv1에 분산 맵이 작성됩니다.

```
LOAD FROM file OF ASC METHOD L ( ...) INSERT INTO table CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

이 매개변수는 ANALYZE 모드가 지정된 경우 사용해야 합니다. 모든 데이터베이스 파티션에서 균등하게 분산하는 최적의 분산 맵이 생성됩니다. 이 매개변수를 지정하지 않고 ANALYZE 모드가 지정되면 프로그램은 오류로 종료됩니다.

TRACE X

해시 값의 출력 및 데이터 변환 프로세스의 덤프를 검토해야 하는 경우 추적할 레코드 수를 지정합니다. 디폴트값은 0입니다.

NEWLINE

입력 데이터 파일이 각 레코드가 개행 문자로 구분된 ASC 파일이고 reclen 파일 유형 수정자가 LOAD 명령에 지정된 경우 사용합니다. 이 옵션을 지정하면 각 레코드에 개행 문자가 있는지 확인합니다. reclen 파일 유형 수정자에 지정된 대로 레코드 길이도 확인합니다.

DISTFILE X

이 옵션을 지정하면 로드 유틸리티가 지정된 이름으로 데이터베이스 파티션 분산 파일을 생성합니다. 데이터베이스 파티션 분산 파일은 32 768개 정수를 포함하며 목표 테이블의 분산 맵에서 각 항목에 하나씩 존재합니다. 파일의 각 정수는 대응하는 분산 맵 항목으로 해시되는 로드할 입력 파일의 행 수를 나타냅니다. 이 정보는 데이터에서 비대칭 항목을 식별하고 유틸리티의 ANALYZE

모드를 사용하여 테이블에서 새 분산 맵을 생성해야 하는지 여부를 결정하는 데에도 도움이 됩니다. 이 옵션을 지정하지 않으면 로드 유틸리티의 디폴트 동작은 분산 파일을 생성하지 않는 것입니다.

주: 이 옵션을 지정하면 로드 조작에 대해 최대 하나의 파티셔닝 에이전트를 사용합니다. 명시적으로 다중 파티셔닝 에이전트를 요청해도 하나만 사용됩니다.

OMIT_HEADER

분산 맵 헤더를 분산 파일에 포함하지 않도록 지정합니다. 지정하지 않으면 헤더가 생성됩니다.

RUN_STAT_DBPARTNUM X

LOAD 명령에 STATISTICS YES 매개변수가 지정되면 하나의 데이터베이스 파티션에서만 통계가 수집됩니다. 이 매개변수에서는 통계를 수집할 데이터베이스 파티션을 지정합니다. 값이 -1이거나 전혀 지정되지 않으면 출력 데이터베이스 파티션 목록의 첫 번째 데이터베이스 파티션에서 통계가 수집됩니다.

파티션된 데이터베이스 환경의 로드 세션 - CLP 예

다음 예에서는 다중 파티션 데이터베이스로 데이터를 로드하는 방법에 대해 설명합니다.

데이터베이스에는 0부터 3까지 번호가 지정된 4개의 데이터베이스 파티션이 있습니다. 데이터베이스 WSDB는 모든 데이터베이스 파티션에 정의되어 있으며 테이블 TABLE1은 마찬가지로 모든 데이터베이스 파티션에 정의된 디폴트 데이터베이스 파티션 그룹에 있습니다.

예 1

데이터베이스 파티션 0에 있는 사용자 데이터 파일 load.del에서 TABLE1로 데이터를 로드하려면 데이터베이스 파티션 0에 연결하고 다음 명령을 실행하십시오.

```
load from load.del of del replace into table1
```

로드 조작에 성공하면 출력은 다음과 같습니다.

```
에이전트 유형  노드     SQL 코드     결과
------------------------------------------------
LOAD           000      +00000000    성공
------------------------------------------------
LOAD           001      +00000000    성공.
------------------------------------------------
LOAD           002      +00000000    성공
------------------------------------------------
LOAD           003      +00000000    성공
------------------------------------------------
PARTITION      001      +00000000    성공.
------------------------------------------------
PRE_PARTITION  000      +00000000    성공.
```

```
결과:          4 / 4 LOAD가 완료되었습니다.

파티션 에이전트 요약:
읽은 행 수                  = 100000
거부된 행 수                = 0
파티션된 행 수              = 100000

LOAD 에이전트 요약:
읽은 행 수                  = 100000
건너뛴 행 수                = 0
로드된 행 수                = 100000
거부된 행 수                = 0
삭제된 행 수                = 0
커미트된 행 수              = 100000
```

출력에서는 각 데이터베이스 파티션에 하나의 로드 에이전트가 있고 각각 성공적으로 실행되었음을 표시합니다. 또한 데이터베이스 파티션 1에서 실행하는 하나의 파티셔닝 에이전트와 코디네이터 파티션에서 실행하는 하나의 사전 파티셔닝 에이전트가 있음을 표시합니다. 이 프로세스는 일반 SQL 리턴 코드 0을 표시하며 성공적으로 완료됩니다. 통계 요약에서는 사전 파티셔닝 에이전트가 100,000개 행을 읽고 파티셔닝 에이전트가 100,000개 행을 분산시켰고 로드 에이전트에서 로드한 모든 행의 합계가 100,000개임을 표시합니다.

예 2

다음 예에서 PARTITION_ONLY 모드로 데이터가 TABLE1에 로드됩니다. 분산된 출력 파일은 /db/data 디렉토리의 각 출력 데이터베이스 파티션에 저장됩니다.

```
load from load.del of del replace into table1 partitioned db config mode
   partition_only part_file_location /db/data
```

load 명령의 출력은 다음과 같습니다.

```
에이전트 유형  노드     SQL 코드     결과
------------------------------------------------------------
LOAD_TO_FILE   000      +00000000    성공.
------------------------------------------------------------
LOAD_TO_FILE   001      +00000000    성공.
------------------------------------------------------------
LOAD_TO_FILE   002      +00000000    성공.
------------------------------------------------------------
LOAD_TO_FILE   003      +00000000    성공.
------------------------------------------------------------
PARTITION      001      +00000000    성공.
------------------------------------------------------------
PRE_PARTITION  000      +00000000    성공.
------------------------------------------------------------

파티션 에이전트 요약:
읽은 행 수                  = 100000
거부된 행 수                = 0
파티션된 행 수              = 100000
```

출력에서는 각 출력 데이터베이스 파티션에서 실행하는 파일로 로드하는 에이전트가 있으며 이 에이전트가 성공적으로 실행되었음을 표시합니다. 데이터베이스 파티션 1에서 실행하는 파티셔닝 에이전트와 코디네이터 파티션에서 실행하는 사전 파티셔닝 에이전트가 있습니다. 통계 요약에서는 사전 파티셔닝 에이전트에서 100,000개 행을 성공적으로 읽고 파티셔닝 에이전트에서 100,000개 행이 성공적으로 분산되었음을 나타냅니다. 테이블로 로드된 행이 없으므로 로드된 행 수에 대한 요약은 나타나지 않습니다.

예 3

위의 PARTITION_ONLY 로드 조작 중 생성된 파일을 로드하려면 다음 명령을 실행하십시오.

```
load from load.del of del replace into table1 partitioned db config mode
    load_only part_file_location /db/data
```

load 명령의 출력은 다음과 같습니다.

```
에이전트 유형  노드     SQL 코드     결과
_____________________________________________
LOAD           000      +00000000    성공
_____________________________________________
LOAD           001      +00000000    성공.
_____________________________________________
LOAD           002      +00000000    성공
_____________________________________________
LOAD           003      +00000000    성공
_____________________________________________
결과:          4 / 4 LOAD가 완료되었습니다.
_____________________________________________

LOAD 에이전트 요약:
읽은 행 수                = 100000
건너뛴 행 수              = 0
로드된 행 수              = 100000
거부된 행 수              = 0
삭제된 행 수              = 0
커미트된 행 수            = 100000
```

출력에서는 각 출력 데이터베이스 파티션의 로드 에이전트가 성공적으로 실행되었으며 모든 로드 에이전트에서 로드된 행 수가 100,000개임을 표시합니다. 분산이 수행되지 않았으므로 분산된 행 수는 표시되지 않습니다.

예 4 - 실패한 로드 조작

다음 LOAD 명령을 발행한 경우

```
load from load.del of del replace into table1
```

로드 조작 중 로드하는 데이터베이스 파티션에서 테이블 스페이스의 공간이 부족한 경우 다음 출력이 리턴됩니다.

```
SQL0289N  테이블 스페이스 "DMS4KT"에 새 페이지를 할당할 수 없습니다.
SQLSTATE=57011

에이전트 유형  노드     SQL 코드     결과
------------------------------------------------------------------------
LOAD           000      +00000000    성공
------------------------------------------------------------------------
LOAD           001      -00000289    오류. 재시작 필요
------------------------------------------------------------------------
LOAD           002      +00000000    성공
------------------------------------------------------------------------
LOAD           003      +00000000    성공
------------------------------------------------------------------------
PARTITION      001      +00000000    성공.
------------------------------------------------------------------------
PRE_PARTITION  000      +00000000    성공.
------------------------------------------------------------------------
결과:          3 / 4 LOAD가 완료되었습니다.
------------------------------------------------------------------------

파티션 에이전트 요약:
읽은 행 수                  = 0
거부된 행 수                = 0
파티션된 행 수              = 0

LOAD 에이전트 요약:
읽은 행 수                  = 0
건너뛴 행 수                = 0
로드된 행 수                = 0
거부된 행 수                = 0
삭제된 행 수                = 0
커미트된 행 수              = 0
```

출력에서는 로드 조작에서 오류 SQL0289를 리턴함을 표시합니다. 데이터베이스 파티션 요약에서는 데이터베이스 파티션 1에서 스페이스가 부족함을 표시합니다. 데이터베이스 파티션 1에서 테이블 스페이스의 컨테이너에 추가 스페이스를 추가하면 다음과 같이 로드 조작을 재시작할 수 있습니다.

```
load from load.del of del restart into table1
```

이주 및 버전 호환성

DB2_PARTITIONEDLOAD_DEFAULT 레지스트리 변수는 다중 파티션 데이터베이스에서 이전 DB2® Universal Database™ 버전 8 로드 동작으로 되돌리는 데 사용될 수 있습니다.

주: DB2_PARTITIONEDLOAD_DEFAULT 레지스트리 변수는 사용되지 않으며 추후 릴리스에서 제거됩니다.

다중 파티션 데이터베이스에서 LOAD 명령의 이전 DB2 UDB 버전 8 동작으로 되돌리면 추가 파티션된 데이터베이스 구성 옵션을 지정하지 않고도 단일 데이터베이스 파티션에 올바른 분산 헤더를 포함하는 파일을 로드할 수 있습니다.

DB2_PARTITIONEDLOAD_DEFAULT 값을 NO로 설정하면 됩니다. 단일 데이터베이스 파티션에서 LOAD 명령을 실행하는 기존 스크립트를 수정하지 않으려면 이 옵션을 사용하도록 선택할 수 있습니다. 예를 들어 4개의 데이터베이스 파티션 그룹으로 구성된 데이터베이스 파티션 그룹에 있는 테이블의 데이터베이스 파티션 3으로 분산 파일을 로드하려면 다음 명령을 실행합니다.

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

그 다음 DB2 명령행 처리기에서 다음 명령을 실행하십시오.

```
CONNECT RESET

SET CLIENT CONNECT_NODE 3

CONNECT TO DB MYDB

LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

다중 파티션 데이터베이스의 경우 다중 파티션 데이터베이스 로드 구성 옵션이 지정되지 않으면 테이블이 정의된 모든 데이터베이스 파티션에서 로드 조작을 수행합니다. 입력 파일에 분산 헤더가 필요하지 않으며 MODE 옵션은 디폴트로 PARTITION_AND_LOAD로 지정됩니다. 단일 데이터베이스 파티션을 로드하려면 OUTPUT_DBPARTNUMS 옵션을 지정해야 합니다.

제 16 장 파티션된 데이터베이스 이주 환경

파티션된 데이터베이스 이주

파티션된 데이터베이스 이주 환경에 모든 데이터베이스 파티션 서버에 최근 데이터베이스 제품 릴리스를 설치하고 인스턴스를 이주한 다음 데이터베이스를 이주해야 합니다.

카탈로그 데이터베이스 파티션 서버 또는 다른 데이터베이스 파티션 서버에서 데이터베이스 파티션 서버를 이주할 수 있습니다. 이주 프로세스가 실패하면 카탈로그 데이터베이스 파티션 서버 또는 다른 데이터베이스 파티션 서버에서 이주를 다시 시도할 수 있습니다.

이러한 정렬의 이주는 매우 중요하므로 이주 절차, 전제조건 및 제한사항은 이 책에서 다루는 범위가 아닙니다. 자세한 설명은 *이주 설명서*의 『파티션된 데이터베이스 이주 환경』 주제에 있으며 이 주제에서 이주 수행 전에 검토할 여러 가지 다른 주제를 참조할 수 있습니다.

제 17 장 스냅샷 및 이벤트 모니터 사용

스냅샷 모니터 데이터를 사용하여 파티션된 테이블의 재구성 모니터

다음 정보는 테이블 재구성의 전역 상태를 모니터링하는 데 가장 유용한 몇 가지 방법에 대해 설명합니다.

파티션된 테이블의 전반적인 테이블 재구성 상태를 나타내는 별도의 데이터 그룹은 없습니다. 파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 데이터 파티션 또는 범위라고 하는 여러 스토리지 오브젝트로 나누어지는 데이터 구성 스킴을 사용합니다. 그러나 재구성 중인 개별 데이터 파티션 데이터 그룹의 요소 값에서 테이블 재구성의 전역 상태를 추측할 수 있습니다. 다음 정보는 테이블 재구성의 전역 상태를 모니터링하는 데 가장 유용한 몇 가지 방법에 대해 설명합니다.

재구성 중인 데이터 파티션 수 판별

테이블 이름 및 스키마 이름이 동일한 테이블 데이터의 모니터 데이터 블록 수를 카운트하여 테이블에서 재구성 중인 총 파티션 수를 판별할 수 있습니다. 이 값은 재구성이 시작된 데이터 파티션 수를 표시합니다. 예 1 및 예 2는 세 개의 데이터 파티션을 재구성 중임을 나타냅니다.

재구성 중인 데이터 파티션 식별

단계 시작 시간으로 재구성 중인 현재 데이터 파티션을 추측할 수 있습니다 (reorg_phase_start). SORT/BUILD/REPLACE 단계 중에, 재구성 중인 데이터 파티션에 해당하는 모니터 데이터는 최근 단계 시작 시간을 표시합니다. INDEX_RECREATE 단계 중에는 모든 데이터 파티션의 단계 시작 시간이 동일합니다. 예 1 및 예 2에 INDEX_RECREATE 단계가 표시되어 있습니다. 따라서 모든 데이터 파티션의 시작 시간이 동일합니다.

인덱스 재빌드 요구사항 식별

재구성 중인 데이터 파티션 중 하나에 해당하는 최대 재구성 단계 요소 (reorg_max_phase)의 값을 확보하는 데 인덱스 재빌드가 필요한지 여부를 판별할수 있습니다. reorg_max_phase의 값이 3 또는 4이면 인덱스를 재빌드해야 합니다. 예 1 및 2에서 reorg_max_phase 값이 3임을 알 수 있습니다. 이는 인덱스 재빌드가 필요함을 표시합니다.

다음 출력 샘플은 세 개의 데이터 파티션이 있는 테이블을 포함하는 3 노드 서버에서 출력된 것입니다.

```
CREATE TABLE  sales (c1 INT, c2 INT, c3 INT)
    PARTITION BY RANGE (c1)
                (PART P1 STARTING FROM (1)  ENDING AT (10) IN parttbs,
         PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
         PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
    DISTRIBUTE BY (c2)
```

실행된 명령문:

```
REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS
```

예 1:

```
GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL
```

관련 테이블에 대한 테이블 정보만 포함하도록 출력을 수정했습니다.

```
           Table Snapshot

First database connect timestamp     = 06/28/2005 13:46:43.061690
Last reset timestamp                 = 06/28/2005 13:46:47.440046
Snapshot timestamp                   = 06/28/2005 13:46:50.964033
Database name                        = DPARTDB
Database path                        = /work/sales/NODE0000/SQL00001/
Input database alias                 = DPARTDB
Number of accessed tables            = 5

Table List
 Table Schema        = NEWTON
 Table Name          = SALES
 Table Type          = User
 Data Partition Id   = 0
 Data Object Pages   = 3
 Rows Read           = 12
 Rows Written        = 1
 Overflows           = 0
 Page Reorgs         = 0
 Table Reorg Information:
   Node number       = 0
   Reorg Type        =
        Reclaiming
        Table Reorg
        Allow No Access
        Recluster Via Table Scan
        Reorg Data Only
   Reorg Index       = 0
   Reorg Tablespace  = 3
 Long Temp space ID  = 3
   Start Time        = 06/28/2005 13:46:49.816883
   Reorg Phase       = 3 - Index Recreate
   Max Phase         = 3
   Phase Start Time  = 06/28/2005 13:46:50.362918
   Status            = Completed
   Current Counter   = 0
   Max Counter       = 0
   Completion        = 0
   End Time          = 06/28/2005 13:46:50.821244
```

```
Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.822701
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543


Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.814813
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 1
Data Object Pages   = 3
Rows Read           = 8
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
```

```
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.014617
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.026278
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543

Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.006392
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

```
Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 2
Data Object Pages   = 3
Rows Read           = 4
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.199971
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.223742
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543

Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
```

```
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
 Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.179922
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

예 2

GET SNAPSHOT FOR TABLES ON DPARTDB **AT DBPARTITIONNUM** 2

관련 테이블에 대한 테이블 정보만 포함하도록 출력을 수정했습니다.

```
                        Table Snapshot

First database connect timestamp     = 06/28/2005 13:46:43.617833
Last reset timestamp                 =
Snapshot timestamp                   = 06/28/2005 13:46:51.016787
Database name                        = DPARTDB
Database path                        = /work/sales/NODE0000/SQL00001/
Input database alias                 = DPARTDB
Number of accessed tables            = 3

Table List
 Table Schema        = NEWTON
 Table Name          = SALES
 Table Type          = User
 Data Partition Id   = 0
 Data Object Pages   = 1
 Rows Read                  = 0
 Rows Written        = 0
 Overflows           = 0
 Page Reorgs         = 0
 Table Reorg Information:
   Node number       = 2
   Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
   Reorg Index       = 0
   Reorg Tablespace  = 3
 Long Temp space ID  = 3
   Start Time        = 06/28/2005 13:46:49.814813
   Reorg Phase       = 3 - Index Recreate
   Max Phase         = 3
   Phase Start Time  = 06/28/2005 13:46:50.344277
```

```
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 1
Data Object Pages   = 1
Rows Read                   = 0
Rows Written        = 0
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.006392
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 2
Data Object Pages   = 1
Rows Read           = 4
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
```

```
   Start Time         = 06/28/2005 13:46:50.179922
   Reorg Phase        = 3 - Index Recreate
   Max Phase          = 3
   Phase Start Time   = 06/28/2005 13:46:50.344277
   Status             = Completed
   Current Counter    = 0
   Max Counter        = 0
   Completion         = 0
   End Time           = 06/28/2005 13:46:50.803619
```

예 3

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

관련 테이블에 대한 테이블 정보의 서브세트만 포함하도록 출력을 수정했습니다.

```
...   TBSP_NAME TABNAME LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_STATUS ...
     --------- ------- ----------------- ---------- -----------
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
```

9 레코드가 선택되었습니다.

이 쿼리에서 생성된 출력(계속).

```
... LOCK_ESCALATION LOCK_ATTRIBUTES DATA_PARTITION_ID DBPARTITIONNUM
   --------------- --------------- ----------------- --------------
...               0 INSERT          2                 2
...               0 NONE            -                 2
...               0 NONE            2                 2
...               0 INSERT          0                 0
...               0 NONE            -                 0
...               0 NONE            0                 0
...               0 INSERT          1                 1
...               0 NONE            -                 1
...               0 NONE            1                 1
```

예 4:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

관련 테이블에 대한 테이블 정보의 서브세트만 포함하도록 출력을 수정했습니다.

```
... TABSCHEMA TABNAME TAB_FILE_ID TAB_TYPE   DATA_OBJECT_PAGES ROWS_WRITTEN ...
... --------- ------- ----------- ---------- ----------------- ------------ ...
... NEWTON    SALES             2 USER_TABLE                 1            1 ...
... NEWTON    SALES             4 USER_TABLE                 1            1 ...
... NEWTON    SALES             3 USER_TABLE                 1            1 ...
```

3 레코드가 선택되었습니다.

이 쿼리에서 생성된 출력(계속).

```
... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID DATA_PARTITION_ID
... ----------------- ----------- -------------- ------- -----------------
...                 0           0              0       3                 0
...                 0           0              2       3                 2
...                 0           0              1       3                 1
```

예 5:

```
SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;
```

관련 테이블에 대한 테이블 정보의 서브세트만 포함하도록 출력을 수정했습니다.

```
   REORG_PHASE   REORG_MAX_PHASE REORG_TYPE                                  ...
   ------------- --------------- ------------------------------------------
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...

 9 레코드가 선택되었습니다.
```

이 쿼리에서 생성된 출력(계속).

```
... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
    ------------------------- -------------- -----------------
... COMPLETED                3              2                 0
... COMPLETED                3              2                 1
... COMPLETED                3              2                 2
... COMPLETED                3              1                 0
... COMPLETED                3              1                 1
... COMPLETED                3              1                 2
... COMPLETED                3              0                 0
... COMPLETED                3              0                 1
... COMPLETED                3              0                 2
```

예 6: 테이블 Reorg 정보에는 재구성 조작의 일부로 Extent를 재개하는 방법에 대한 정보가 포함됩니다. 다음 예는 관련된 출력을 나타냅니다.

```
db2 -v "get snapshot for tables on wsdb"

 Table Reorg Information:
   Reorg Type         =
        Reclaim Extents
        Allow Write Access
   Reorg Index        = 0
   Reorg Tablespace   = 0
   Start Time         = 10/22/2008 15:49:35.477532
   Reorg Phase        = 12 - Release
   Max Phase          = 3
```

주: SQLM_DBMON_VERSION9_7 이전 모니터 버전의 스냅샷 요청은 요청하는 클라이언트에 재개 Reorg 상태를 리턴하지 않습니다.

파티션된 데이터베이스 시스템의 전역 스냅샷

파티션된 데이터베이스 시스템에서 스냅샷 모니터를 사용하여 현재 파티션, 지정된 파티션 또는 모든 파티션의 스냅샷을 찍을 수 있습니다. 파티션된 데이터베이스의 모든 파티션에서 전역 스냅샷을 찍는 경우 결과가 리턴되기 전에 데이터가 집계됩니다.

다음과 같이 여러 요소 유형에 관해 데이터가 집계됩니다.

- **카운터, 시간 및 게이지(gauge)**

  인스턴스의 각 파티션에서 수집된 모든 like 값 합계가 포함됩니다. 예를 들어, GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL은 파티션된 데이터베이스 인스턴스에 있는 모든 파티션의 데이터베이스에서 읽은 행(rows_read) 수를 리턴합니다.

- **워터 마크(water mark)**

  파티션된 데이터베이스 시스템에서 임의의 파티션에 대해 발견된 최고(상위 워터 마크) 또는 최저(하위 워터 마크) 값을 리턴합니다. 리턴된 값이 중요한 경우에는 특정 파티션이 너무 많이 사용되는지 또는 인스턴스 전반에 문제점이 있는지 여부를 판별하기 위해 개별 파티션의 스냅샷을 찍습니다.

- **시간소인**

  스냅샷 모니터 인스턴스 에이전트가 접속한 파티션의 시간소인 값으로 설정됩니다. 모든 시간소인 값은 `timestamp` 모니터 스위치를 통해 제어되는 점에 유의하십시오.

- **정보**

  작업을 방해할 수도 있는 파티션에 관한 가장 중요한 정보를 리턴합니다. 예를 들어, `appl_status` 요소의 상태가 하나의 파티션에서는 UOW 실행 중이고 다른 파티션에서는 잠금 대기인 경우 잠금 대기는 응용프로그램 실행을 보류시키는 상태이므로 잠금 대기가 리턴됩니다.

파티션된 데이터베이스서 개별 파티션 또는 모든 파티션에 대해 카운터를 재설정하고 모니터 스위치를 설정하며 모니터 스위치 설정을 검색할 수도 있습니다.

주: 전역 스냅샷을 찍는 중에 하나 이상의 파티션에서 오류가 발생하는 경우에는 스냅샷이 성공한 파티션에서 데이터가 수집되고 경고 (sqlcode 1629)도 리턴됩니다. 모니터 스위치의 갱신이나 전역 가져오기 또는 카운터 재설정이 하나 이상의 파티션에서 실패하는 경우 해당 파티션에서는 스위치 설정 또는 데이터 재설정이 이루어지지 않습니다.

파티션된 데이터베이스의 이벤트 모니터 작성

파티션된 데이터베이스 시스템에서 파일 또는 파이프 이벤트 모니터를 작성하는 경우 수집하려는 모니터링 데이터의 범위를 결정해야 합니다.

시작하기 전에

파티션된 데이터베이스의 이벤트 모니터를 작성하려면 SQLADM 또는 DBADM 권한이 있어야 합니다.

이 태스크에 대한 정보

이벤트 모니터에서는 운영 체제 프로세스 또는 스레드를 사용하여 이벤트 레코드를 기록합니다. 이러한 프로세스 또는 스레드가 실행되는 데이터베이스 파티션을 모니터 파티션이라고 합니다. 파일 및 파이프 이벤트 모니터는 모니터 파티션에서 로컬로 발생하거나 DB2 데이터베이스 관리 프로그램이 실행 중인 모든 파티션에서 전역으로 발생하므로 모니터링 이벤트가 될 수 있습니다. 전역 이벤트 모니터는 모든 파티션의 활동이 포함된 모니터링 파티션에 대한 단일 추적을 기록합니다. 이벤트 모니터가 로컬인지 또는 전역인지 여부는 해당 모니터링 범위에 따라서 판별됩니다.

모니터 파티션과 모니터 범위 둘 다 CREATE EVENT MONITOR문을 통해 지정됩니다.

모니터 파티션이 활성 상태인 경우에만 이벤트 모니터를 활성화할 수 있습니다. SET EVENT MONITOR문을 사용하여 이벤트 모니터를 활성화했으나 모니터 파티션은 아직 활성화되지 않은 경우에는 다음에 모니터 파티션을 시작할 때 이벤트 모니터가 활성화됩니다. 또한 이벤트 모니터가 명시적으로 비활성화되거나 인스턴스가 명시적으로 비활성화될 때까지 이벤트 모니터 활성화가 자동으로 발생합니다. 예를 들어, 데이터베이스 파티션 0의 경우에는 다음과 같습니다.

```
db2 connect to sample
db2 create event monitor foo ... on dbpartitionnum 2
db2 set event monitor foo state 1
```

위의 명령을 실행하면 데이터베이스 파티션 2에서 데이터베이스 `sample`이 활성화될 때마다 이벤트 모니터 `foo`가 자동으로 활성화됩니다. 이러한 자동 활성화는 `db2 set event monitor foo state 0`이 발행되거나 파티션 2가 중지될 때까지 발생합니다.

테이블에 기록 이벤트 모니터에서는 로컬 또는 전역 범위의 개념을 적용할 수 없습니다. 테이블에 기록 이벤트 모니터가 활성화되면 모든 파티션에서 이벤트 모니터 프로세스가 실행됩니다. (명확히 설명하자면 이벤트 모니터 프로세스는 목표 테이블이 상주하는 데이터베이스 파티션 그룹에 속하는 파티션에서 실행됩니다.) 이벤트 모니터 프로세스가 실행 중인 각 파티션에도 동일한 목표 테이블 세트가 있습니다. 이러한 테이블의

데이터는 모니터링 데이터의 개별 파티션 뷰를 나타내므로 서로 다릅니다. 각 파티션의 이벤트 모니터 목표 테이블에서 원하는 값에 액세스하는 SQL문을 발행하여 모든 파티션에서 집계 값을 가져올 수 있습니다.

각 목표 테이블의 첫 번째 컬럼 이름은 PARTITION_KEY로 지정되며 테이블의 파티셔닝 키로 사용됩니다. 각 이벤트 모니터 프로세스에서 프로세스가 실행되는 데이터베이스 파티션에 데이터를 삽입할 수 있도록 이 컬럼의 값이 선택됩니다. 즉, 삽입 조작은 이벤트 모니터 프로세스가 실행되는 데이터베이스 파티션에서 로컬로 수행됩니다. 모든 데이터베이스 파티션에서 PARTITION_KEY 필드는 같은 값을 포함합니다. 즉, 데이터 파티션이 삭제되고 데이터 재분배가 수행되는 경우 삭제된 데이터베이스 파티션에 있는 모든 데이터는 고르게 분배되지 않고 다른 데이터베이스 파티션으로 이동됩니다. 따라서 데이터베이스 파티션을 제거하기 전에 제거할 데이터베이스 파티션에 있는 모든 테이블 행을 삭제하는 것이 좋습니다.

또한 PARTITION_NUMBER로 이름이 지정된 컬럼을 각 테이블에 정의할 수 있습니다. 이 컬럼에는 데이터가 삽입된 파티션의 번호가 포함되어 있습니다. PARTITION_KEY 컬럼과 달리 PARTITION_NUMBER 컬럼은 필수가 아닙니다.

목표 테이블이 정의된 테이블 스페이스는 이벤트 모니터 데이터를 기록할 모든 파티션에 존재해야 합니다. 이 규칙을 준수하지 않으면 테이블 스페이스가 없는 파티션(이벤트 모니터가 있음)의 로그에 레코드가 작성되지 않습니다. 테이블 스페이스가 없는 파티션에 이벤트는 여전히 작성되고 오류는 리턴되지 않습니다. 이러한 동작 때문에 사용자는 특정 파티션에만 존재하는 테이블 스페이스를 작성하여 모니터할 파티션의 서브세트를 선택할 수 있습니다.

테이블에 기록 이벤트 모니터를 활성화하는 동안에는 FIRST_CONNECT 및 EVMON_START의 CONTROL 테이블 행만 카탈로그 데이터베이스 파티션에 삽입됩니다. 이 때 제어 테이블에 대한 테이블 스페이스가 카탈로그 데이터베이스 파티션에 있어야 합니다. 테이블 스페이스가 카탈로그 데이터베이스 파티션에 없으면 이러한 삽입이 수행되지 않습니다.

테이블에 기록 이벤트 모니터 활성화 시 파티션이 아직 활성되지 않은 경우 해당 파티션이 다음에 활성화될 때 이벤트 모니터가 활성화됩니다.

추가되는 즉시 온라인 상태인 데이터베이스 파티션을 추가하는 경우 이벤트 모니터는 새 파티션을 즉시 인식하지 않습니다. 새 파티션에 대한 데이터를 수집하여 기록하려면 다음 중 하나를 수행해야 합니다.

- 전역 이벤트 모니터의 경우 이벤트 모니터를 재시작하십시오.
- 테이블에 기록 이벤트 모니터의 경우 이벤트 모니터를 삭제, 재작성 및 재시작하십시오.

.

주: 자세한 교착 상태 연결 이벤트의 잠금 목록에는 잠금 대기 중인 파티션의 응용프로그램에 보유된 잠금만 포함됩니다. 예를 들어, 교착 상태에 관련된 응용프로그램이 노드 20에서 잠금 대기 중인 경우 노드 20의 해당 응용프로그램에 보유된 잠금만 목록에 포함됩니다.

프로시저

1. 모니터할 파티션을 지정하십시오.

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3
   ```

   dlmon은 이벤트 모니터 이름을 나타냅니다.

   /tmp/dlevents는 이벤트 모니터가 이벤트 파일을 작성할 디렉토리 경로(UNIX의 경우) 이름입니다.

   3은 모니터할 파티션 번호를 나타냅니다.

2. 이벤트 모니터 데이터를 로컬 또는 전역 범위에서 수집할지 여부를 지정하십시오. 모든 파티션에서 이벤트 모니터 보고서를 수집하려면 다음 명령문을 발행하십시오.

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3 GLOBAL
   ```

   교착 상태 및 세부사항이 포함된 교착 상태 이벤트 모니터만 GLOBAL로 정의할 수 있습니다. 모든 파티션이 교착 상태 관련 이벤트 레코드를 파티션 3에 보고합니다.

3. 로컬 파티션에서만 이벤트 모니터 보고서를 수집하려면 다음 명령문을 발행하십시오.

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3 LOCAL
   ```

   이 동작은 파티션된 데이터베이스의 파일 및 파이프 이벤트 모니터의 디폴트 동작입니다. 테이블에 기록 이벤트 모니터에서는 LOCAL 및 GLOBAL 절이 무시됩니다.

4. 기존 이벤트 모니터의 모니터 파티션 및 범위 값을 검토할 수 있습니다. 다음 명령문을 사용하여 SYSCAT.EVENTMONITORS 테이블에서 이 쿼리를 수행하십시오.

   ```
   SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
   ```

결과

이벤트 모니터가 작성되어 활성화되면 지정된 이벤트가 발생하는 경우 모니터링 데이터를 기록합니다.

제 18 장 좋은 백업 및 복구 전략 개발

응급 복구

데이터베이스에서의 트랜잭션 또는 작업 단위(UOW)는 예상치 않게 인터럽트될 수 있습니다. 작업 단위(UOW)에 속한 변경 모두가 완료 및 커미트되기 전에 실패한 경우 데이터베이스는 불일치한 상태 및 사용 불가능한 상태로 남아 있습니다. *응급 복구*는 데이터베이스가 일관성 있는 사용 가능한 상태로 다시 이동되는 프로세스입니다. 불완전한 트랜잭션을 롤백하고 손상되었을 때 메모리에 남아 있는 커미트된 트랜잭션을 완료하면 됩니다(그림 45). 데이터베이스가 일관성 있고 사용 가능한 상태이면 "일관성 지점"이라고 하는 지점에 도달합니다.

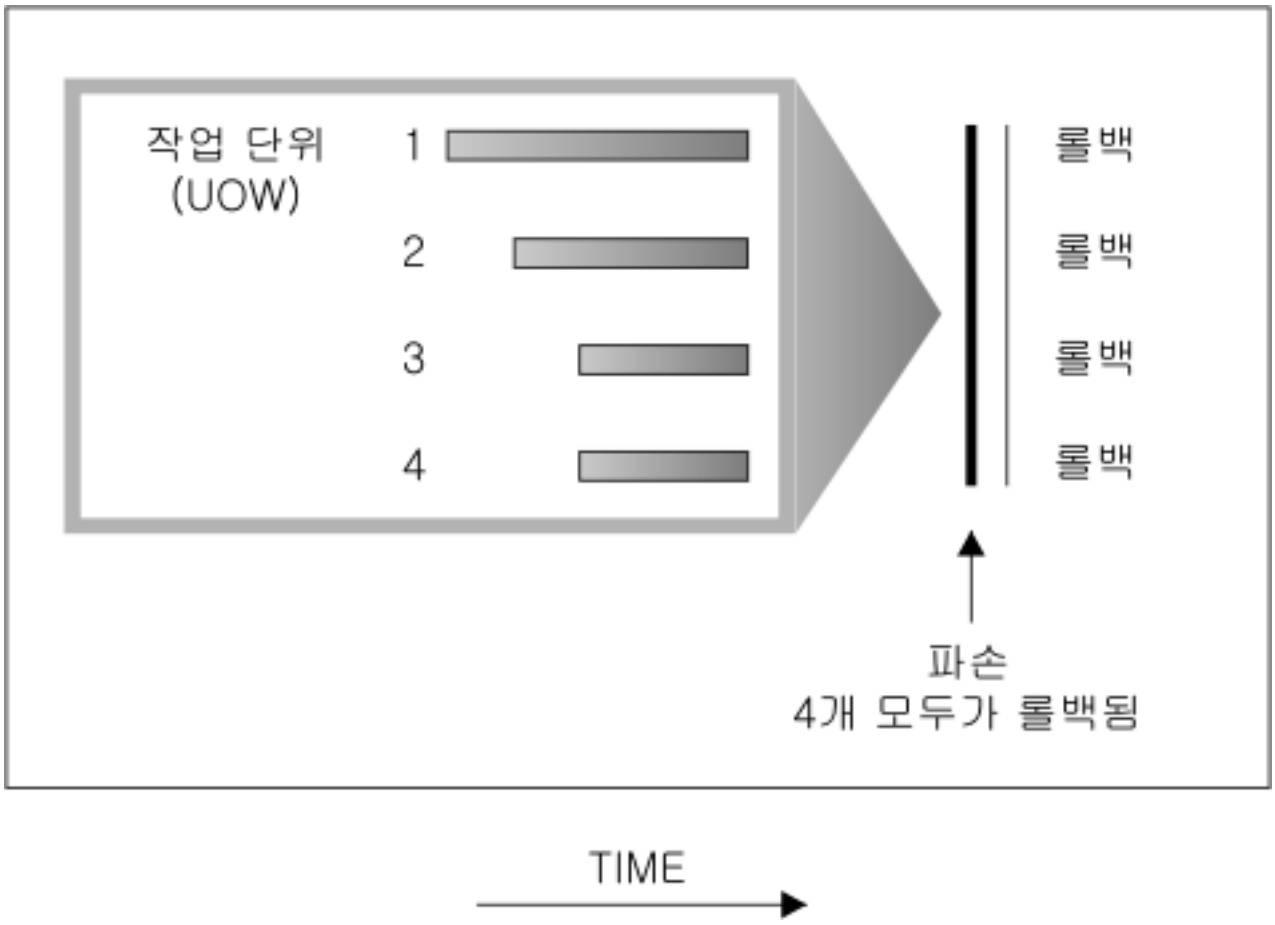


그림 45. 작업 단위(UOW) 롤백(응급 복구)

*트랜잭션 실패*는 데이터베이스 또는 데이터베이스 관리 프로그램이 비정상적으로 종료되는 심각한 오류 또는 조건으로 인해 발생합니다. 실패 시 디스크에 플러시되지 않은

부분적으로 완료된 작업 단위(UOW)는 데이터베이스에서 불일치 상태로 남게 됩니다. 트랜잭션 실패 이후에는 데이터베이스를 복구해야 합니다. 트랜잭션 실패를 야기하는 조건으로는 다음이 있습니다.

- 머신에서 전력 장애가 발생한 경우. 이로 인해 이 머신의 데이터베이스 관리 프로그램 및 데이터베이스 파티션이 중지됩니다.
- 메모리 손상 또는 디스크, CPU 또는 네트워크 장애와 같은 하드웨어 장애.
- DB2가 중단되는 심각한 운영 체제 오류.
- 응용프로그램의 비정상적 종료.

데이터베이스 관리 프로그램이 미완료된 작업 단위(UOW)를 자동으로 롤백하도록 하려면 자동 재시작(*autorestart*) 데이터베이스 구성 매개변수를 ON으로 설정하여 사용합니다. 이는 디폴트값입니다. 자동 재시작 동작을 사용하지 않으려면 *autorestart* 데이터베이스 구성 매개변수를 OFF로 설정합니다. 데이터베이스 오류가 발생한 경우 RESTART DATABASE 명령을 실행해야 합니다. 손상되기 전에 데이터베이스 입출력이 일시중단된 경우 응급 복구를 계속하려면 RESTART DATABASE 명령의 WRITE RESUME 옵션을 지정해야 합니다. 데이터베이스 재시작 조작을 시작하면 관리 통지 로그가 기록됩니다.

응급 복구가 포워드 복구가 가능한 데이터베이스(즉, *logarchmeth1* 구성 매개변수가 OFF로 설정되지 않음)에 적용되는 경우 개별 테이블 스페이스에 기인하는 응급 복구 중 오류가 발생하면 해당 테이블 스페이스가 오프라인 상태가 되고 수리할 때까지 액세스할 수 없습니다. 응급 복구가 계속됩니다. 응급 복구를 완료하면 데이터베이스의 다른 테이블 스페이스에 액세스할 수 있으며 데이터베이스에 대한 연결을 설정할 수 있습니다. 그러나 오프라인 상태의 테이블 스페이스가 시스템 카탈로그를 포함하는 테이블 스페이스인 경우 연결을 허용하기 전에 먼저 수리해야 합니다.

파티션된 데이터베이스 환경에서 트랜잭션 장애 복구

파티션된 데이터베이스 환경에서 트랜잭션에 실패하면 일반적으로 실패한 데이터베이스 파티션 서버 및 트랜잭션에 참여한 기타 데이터베이스 파티션 서버 모두에서 데이터베이스 복구를 수행해야 합니다.

- 실패한 조건을 정정한 후 실패한 데이터베이스 파티션 서버에서 응답 복구가 수행됩니다.
- 실패를 발견한 후 사용 중인 다른 데이터베이스 파티션 서버에서 즉시 *데이터베이스 파티션 장애 복구*가 수행됩니다.

파티션된 데이터베이스 환경에서 트랜잭션이 제출된 데이터베이스 파티션 서버가 코디네이터 파티션이며 트랜잭션에서 처리하는 첫 번째 에이전트가 코디네이터 에이전트입니다. 코디네이터 에이전트는 다른 데이터베이스 파티션 서버에서 작업을 분산하고 트랜잭션에 사용되는 항목을 추적합니다. 응용프로그램이 트랜잭션의 COMMIT문을 실행

하면 코디네이터 에이전트는 2단계로 구성된 커미트 프로토콜을 사용하여 트랜잭션을 커미트합니다. 첫 번째 단계 중 코디네이터 파티션은 트랜잭션에 참여하는 모든 다른 데이터베이스 파티션 서버에 PREPARE 요청을 분산합니다. 그러면 서버는 다음 중 하나로 응답합니다.

READ-ONLY
이 서버에서 데이터가 변경되지 않습니다.

YES 이 서버에서 데이터가 변경되었습니다.

NO 오류 때문에 서버에서 커미트 준비가 되지 않음

서버 중 하나가 NO로 응답하면 트랜잭션이 롤백됩니다. 그렇지 않으면 코디네이터 파티션은 두 번째 단계를 시작합니다.

두 번째 단계 중 코디네이터 파티션은 COMMIT 로그 레코드를 작성하고 YES로 응답한 모든 서버에 COMMIT 요청을 분산합니다. 다른 모든 데이터베이스 파티션 서버가 커미트된 후 코디네이터 파티션에 COMMIT 수신확인을 전송합니다. 코디네이터 에이전트가 모든 참여 서버에서 모든 COMMIT 수신확인을 받으면 트랜잭션이 완료됩니다. 이때 포인트 코디네이터는 FORGET 로그 레코드를 작성합니다.

활성 데이터베이스 파티션 서버에서 트랜잭션 장애 복구

데이터베이스 파티션 서버가 다른 서버가 중지되었음을 감지한 경우 실패한 데이터베이스 파티션 서버와 연관된 모든 작업이 중지됩니다.

- 여전히 사용 중인 데이터베이스 파티션 서버가 응용프로그램의 코디네이터 파티션이고 장애가 발생한 데이터베이스 파티션 서버에서 응용프로그램이 실행된 경우(COMMIT는 준비되지 않음) 장애 복구를 수행하기 위해 코디네이터 에이전트가 인터럽트됩니다. 코디네이터 에이전트가 COMMIT 처리의 두 번째 단계를 진행하는 경우 응용프로그램에 SQL0279N이 리턴되고 데이터베이스 연결이 끊어집니다. 그렇지 않으면 코디네이터 에이전트는 트랜잭션에 참여하는 다른 모든 서버에 ROLLBACK 요청을 분산시키고 응용프로그램에 SQL1229N이 리턴됩니다.
- 장애가 발생한 데이터베이스 파티션 서버가 응용프로그램의 코디네이터 파티션인 경우 장애 복구를 수행하기 위해 사용 중인 서버의 응용프로그램에서 여전히 작동하는 에이전트가 인터럽트됩니다. 트랜잭션이 아직 준비된 상태가 아닌 각 데이터베이스 파티션에서 트랜잭션이 로컬로 롤백됩니다. 트랜잭션이 준비 상태인 해당 데이터베이스 파티션에서 트랜잭션은 인다우트(Indoubt) 상태가 됩니다. 코디네이터 데이터베이스 파티션은 일부 데이터베이스 파티션에서 트랜잭션이 인다우트(Indoubt)임을 인식하지 못합니다. 코디네이터 데이터베이스 파티션이 사용 불가능하기 때문입니다.
- 장애가 발생한 데이터베이스 파티션 서버에 응용프로그램이 연결되었지만(장애 전에 연결됨) 로컬 데이터베이스 파티션 서버 또는 장애가 발생한 데이터베이스 파티션 서버가 코디네이터 파티션이 아니면 이 응용프로그램에서 작동하는 에이전트가 인터럽트됩니다. 코디네이터 파티션은 다른 데이터베이스 파티션 서버에 ROLLBACK 또

는 DISCONNECT 메시지를 보냅니다. 코디네이터 파티션이 SQL0279를 리턴하면 여전히 사용 중인 데이터베이스 파티션 서버에서 트랜잭션은 인다우트(Indoubt) 상태만 가능합니다.

장애가 발생한 서버로 요청을 보내는 모든 프로세스(예: 에이전트 또는 교착 상태 검출기)에 요청을 보낼 수 없음을 알립니다.

실패한 데이터베이스 파티션 서버에서 트랜잭션 장애 복구

트랜잭션 실패로 데이터베이스 관리 프로그램이 비정상적으로 종료되면 RESTART 옵션과 함께 db2start 명령을 실행하여 데이터베이스 파티션이 재시작되면 데이터베이스 관리 프로그램을 재시작할 수 있습니다. 데이터베이스 파티션을 재시작할 수 없으면 db2start를 실행하여 다른 데이터베이스 파티션에서 데이터베이스 관리 프로그램을 재시작할 수 있습니다.

데이터베이스 관리 프로그램이 비정상적으로 종료되면 서버의 데이터베이스 파티션은 불일치 상태로 남을 수 있습니다. 사용 가능하게 하려는 경우 다음과 같이 데이터베이스 파티션 서버에서 응급 복구를 트리거할 수 있습니다.

- 명시적으로 RESTART DATABASE 명령 사용
- *autorestart* 데이터베이스 구성 매개변수가 ON으로 설정된 경우 명시적으로 CONNECT 요청 사용

응급 복구는 완료된 모든 트랜잭션의 효과가 데이터베이스 안에 적용되도록 사용 중인 로그 파일의 로그 레코드를 다시 적용합니다. 변경 사항이 다시 적용된 후 커미트되지 않은 모든 트랜잭션은 로컬로 롤백됩니다(단, 인다우트(Indoubt) 트랜잭션은 *제외*됨). 파티션된 데이터베이스 환경에서 다음과 같은 두 가지 인다우트(Indoubt) 트랜잭션 유형이 있습니다.

- 코디네이터 파티션이 아닌 데이터베이스 파티션 서버에서 준비되었지만 아직 커미트되지 않은 트랜잭션은 인다우트(in doubt) 상태가 됩니다.
- 코디네이터 파티션에서 커미트되었지만 아직 완료된 것으로 로그되지 않은 트랜잭션(즉, FORGET 레코드가 아직 작성되지 않음)은 인다우트(in doubt) 상태가 됩니다. 코디네이터 에이전트가 응용프로그램에서 작동하는 모든 서버에서 모든 COMMIT 수신확인을 받지 못한 경우 이 상황이 나타납니다.

응급 복구에서는 다음 중 하나를 수행하여 인다우트(Indoubt) 트랜잭션을 모두 해결하려고 합니다. 수행하는 조치는 데이터베이스 파티션 서버가 응용프로그램의 코디네이터 파티션인지에 따라 달라집니다.

- 재시작된 서버가 응용프로그램의 코디네이터 파티션이 아니면 코디네이터 파티션으로 조회 메시지를 보내 트랜잭션 결과를 찾습니다.

- 재시작된 서버가 응용프로그램의 코디네이터 파티션이면 코디네이터 에이전트가 COMMIT 수신확인을 계속 기다리고 있음을 나타내는 메시지가 다른 모든 에이전트(종속 에이전트)로 전송됩니다.

응급 복구는 모든 인다우트(Indoubt) 트랜잭션을 해결할 수는 없습니다. 예를 들어 일부 데이터베이스 파티션 서버가 사용 불가능할 수 있습니다. 트랜잭션에서 다른 데이터베이스 파티션을 사용하기 전에 코디네이터 파티션이 응급 복구를 완료하면 응급 복구로 인다우트(Indoubt) 트랜잭션을 해결할 수 없습니다. 각 데이터베이스 파티션이 독립적으로 응급 복구를 수행했기 때문입니다. 이 경우 SQL 경고 메시지 SQL1061W가 리턴됩니다. 인다우트(Indoubt) 트랜잭션이 자원을 보유하는 경우(예: 잠금 및 사용 중인 로그 스페이스) 데이터베이스를 변경할 수 없는 지점으로 이동할 수 있습니다. 인다우트(Indoubt) 트랜잭션이 사용 중인 로그를 보유하고 있기 때문입니다. 따라서 응급 복구 이후 인다우트(Indoubt) 트랜잭션이 남아 있는지 판별하고 가능한 한 신속하게 인다우트(Indoubt) 트랜잭션을 해결하는 데 필요한 모든 데이터베이스 파티션 서버를 복구해야 합니다.

주: 파티션된 데이터베이스 서버 환경에서 RESTART 데이터베이스 명령은 노드별로 실행됩니다. 모든 노드에서 데이터베이스를 재시작하도록 하려면 다음의 권장 명령을 사용합니다.

```
db2_all "db2 restart database <database_name>"
```

인다우트(Indoubt) 트랜잭션을 해결해야 하는 하나 이상의 서버를 제시간에 복구할 수 없고 다른 서버의 데이터베이스 파티션에 액세스해야 하는 경우 경험을 바탕으로 인다우트(Indoubt) 트랜잭션을 수동으로 해결해야 합니다. LIST INDOUBT TRANSACTIONS 명령을 사용하여 서버에서 인다우트(Indoubt) 트랜잭션을 쿼리, 커미트 및 롤백할 수 있습니다.

주: LIST INDOUBT TRANSACTIONS 명령은 분산 트랜잭션 환경에서도 사용됩니다. 두 가지 유형의 인다우트(Indoubt) 트랜잭션을 서로 구별하기 위해 LIST INDOUBT TRANSACTIONS 명령에서 리턴된 출력의 *originator* 필드는 다음 중 하나를 표시합니다.

- DB2 Enterprise Server Edition. 이는 파티션된 데이터베이스 환경에서 트랜잭션이 시작되었음을 나타냅니다.
- XA. 이는 분산 환경에서 트랜잭션이 시작되었음을 나타냅니다.

장애가 발생한 데이터베이스 파티션 서버 식별

데이터베이스 파티션 서버에서 장애가 발생한 경우 일반적으로 응용프로그램은 다음 SQLCODE 중 하나를 수신합니다. 장애가 발생한 데이터베이스 관리 프로그램을 감지하는 방법은 수신된 SQLCODE에 따라 달라집니다.

SQL0279N

이 SQLCODE는 COMMIT 처리 중 트랜잭션에서 사용하는 데이터베이스 파티션 서버가 종료되는 경우 수신됩니다.

SQL1224N

이 SQLCODE는 장애가 발생한 데이터베이스 파티션이 트랜잭션에 대한 코디네이터 파티션인 경우 수신됩니다.

SQL1229N

이 SQLCODE는 장애가 발생한 데이터베이스 파티션이 트랜잭션에 대한 코디네이터 파티션이 아닌 경우 수신됩니다.

장애가 발생한 데이터베이스 파티션 서버를 판별하는 작업은 2단계 프로세스로 구성됩니다.

1. SQLCA를 검사하여 장애를 발견한 파티션 서버를 찾으십시오. SQLCODE SQL1229N과 연관된 SQLCA는 *sqlerrd* 필드의 6번째 배열 위치에서 오류를 발견한 서버의 노드 번호를 포함합니다. 이때 서버에서 작성된 노드 번호는 `db2nodes.cfg` 파일의 노드 번호에 대응합니다.
2. 실패한 서버의 노드 수에 대해 1단계에서 찾은 서버의 관리 통지 로그를 검사하십시오.

주: 프로세서에서 다중 논리 노드를 사용하는 경우 한 논리 노드에서 장애가 발생하면 동일한 프로세서의 다른 논리 노드에서도 장애가 발생합니다.

데이터베이스 파티션 서버의 실패로부터 복구

데이터베이스 파티션 서버의 실패로부터 복구하려면 다음 단계를 수행하십시오.

1. 실패를 유발한 문제점을 정정하십시오.
2. 임의의 데이터베이스 파티션 서버에서 db2start 명령을 실행하여 데이터베이스 관리 프로그램을 재시작하십시오.
3. 실패한 데이터베이스 파티션 서버에서 RESTART DATABASE 명령을 실행하여 데이터베이스를 재시작하십시오.

파티션된 데이터베이스 재빌드

파티션된 데이터베이스를 재빌드하려면 각 데이터베이스 파티션을 개별적으로 재빌드하십시오. 각 데이터베이스 파티션의 경우 카탈로그 파티션에서 시작하여 먼저 필요한 모든 테이블 스페이스를 리스토어하십시오. 리스토어되지 않는 모든 테이블 스페이스는 리스토어 보류 상태에 놓입니다. 모든 데이터베이스 파티션이 리스토어된 후, 카탈로그 파티션에서 ROLLFORWARD DATABASE 명령을 발행하여 모든 데이터베이스 파티션을 롤 포워드합니다.

주: 나중에 원래 재빌드 단계에 포함되지 않았던 모든 테이블 스페이스를 리스토어해야 하는 경우, 나중에 테이블 스페이스를 롤 포워드할 때 롤 포워드 유틸리티가 데이터베이스 파티션 사이의 모든 데이터를 동기화된 상태로 유지하는지 확인해야 합니다. 테이블 스페이스가 원래 리스토어 및 롤 포워드 조작 중에 누락되는 경우, 이는 데이터에 액세스하려는 시도가 있고 데이터 액세스 오류가 발생할 때까지 발견되지 않을 수 있습니다. 그런 다음 누락된 테이블 스페이스를 리스토어하고 롤 포워드하여 나머지 파티션과 다시 동기화되도록 해야 합니다.

테이블 스페이스 레벨 백업 이미지를 사용하여 파티션된 데이터베이스를 재빌드하려면 다음 예를 고려하십시오.

이 예에서 세 개의 데이터베이스 파티션이 있는 SAMPLE이라는 복구 가능한 데이터베이스가 있습니다.

- 데이터베이스 파티션 1은 테이블 스페이스 SYSCATSPACE, USERSP1 및 USERSP2를 포함하며 카탈로그 파티션입니다.
- 데이터베이스 파티션 2는 테이블 스페이스 USERSP1 및 USERSP3를 포함합니다.
- 데이터베이스 파티션 3은 테이블 스페이스 USERSP1, USERSP2 및 USERSP3을 포함합니다.

다음 백업이 작성되었으며, BKxy는 파티션 y의 백업 번호 x를 나타냅니다.

- BK11은 SYSCATSPACE, USERSP1 및 USERSP2의 백업입니다.
- BK12는 USERSP2 및 USERSP3의 백업입니다.
- BK13은 USERSP1, USERSP2 및 USERSP3의 백업입니다.
- BK21은 USERSP1의 백업입니다.
- BK22는 USERSP1의 백업입니다.
- BK23은 USERSP1의 백업입니다.
- BK31은 USERSP2의 백업입니다.
- BK33은 USERSP2의 백업입니다.
- BK42는 USERSP3의 백업입니다.
- BK43은 USERSP3의 백업입니다.

다음 프로시저는 CLP를 통해 발행되는 RESTORE DATABASE 및 ROLLFORWARD DATABASE 명령을 사용하여 로그 끝까지 전체 데이터베이스 재빌드를 보여줍니다.

1. 데이터베이스 파티션 1에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db sample rebuild with all tablespaces in database
      taken at BK31 without prompting
   ```

2. 데이터베이스 파티션 2에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db sample rebuild with tablespaces in database
   taken at BK42 without prompting
```

3. 데이터베이스 파티션 3에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db sample rebuild with all tablespaces in database
   taken at BK43 without prompting
```

4. 카탈로그 파티션에서, TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db sample to end of logs
```

5. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db sample stop
```

이 때 데이터베이스는 모든 데이터베이스 파티션에서 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

---

db2adutl을 사용한 데이터 복구

다음 예는 db2adutl 명령과 *logarchopt1* 및 *vendoropt* 데이터베이스 구성 매개변수를 사용하여 상호 노드 복구를 수행하는 방법을 보여줍니다.

다음 예의 경우 컴퓨터 1은 이름이 bar이고 AIX를 실행 중입니다. 이 머신의 소유자는 roecken입니다. bar의 데이터베이스 이름은 zample입니다. 컴퓨터 2의 이름은 dps입니다. 이 머신 역시 AIX를 실행 중이며 regress9가 소유하고 있습니다.

PASSWORDACCESS = generate

컴퓨터 1

1. 로그가 TSM으로 아카이브되도록 데이터베이스를 설정하십시오. zample 데이터베이스의 데이터베이스 구성 매개변수 *logarchmeth1*을 갱신하십시오.

```
bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
```

   다음 정보가 리턴됩니다.

```
DB20000I  UPDATE DATABASE CONFIGURATION 명령이 완료되었습니다.
```

   주: 데이터베이스 구성을 갱신하기 전에, 데이터베이스의 오프라인 백업을 수행해야 할 수도 있습니다.

2. 응용프로그램을 강제로 해제하십시오.

```
db2 force applications all
```

3. 모든 응용프로그램이 강제로 해제되었는지 검증하십시오.

```
db2 list applications
```

어떤 데이터도 리턴되지 않았음을 알리는 메시지가 수신되어야 합니다.

주: 파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션에 대해 이 단계를 수행해야 합니다.

4. 데이터베이스 백업을 수행하십시오.

```
db2 backup db zample use tsm
```

다음과 유사한 정보가 리턴됩니다.

```
백업이 완료되었습니다. 이 백업 이미지에 대한 시간소인은
20040216151025입니다.
```

주: 파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션에 대해 이 단계를 수행해야 합니다. 데이터베이스 파티션에서 이 단계를 수행하는 순서는 온라인 백업 또는 오프라인 백업 중 어느 백업을 수행하는지 여부에 따라 다릅니다. 자세한 정보는 487 페이지의 『백업 사용』을 참조하십시오.

5. zample 데이터베이스에 연결한 후 데이터베이스 안에서 테이블을 작성하십시오.
6. 데이터를 새 테이블로 로드하십시오. 이 예에서는 a 테이블이 있고, 데이터는 mr이라고 하는 컬럼 식별자가 있는 ASCII 파일에서 로드됩니다. COPY YES 옵션은 로드되는 데이터의 사본을 작성하기 위해 지정하고 USE TSM 옵션은 데이터 사본이 Tivoli Storage Manager에 저장됨을 지정합니다.

주: 데이터베이스가 롤 포워드 복구에 사용 가능한 경우에만 COPY YES 옵션을 지정할 수 있습니다. 즉, *logarchmeth1* 데이터베이스 구성 매개변수는 USEREXIT 또는 LOGRETAIN으로 설정해야 합니다.

```
bar:/home/roecken> db2 load from mr of del modified by noheader replace
      into a copy yes use tsm
```

유틸리티는 해당 프로세스를 표시하기 위해 일련의 메시지를 리턴합니다.

```
SQL3109N  유틸리티가 파일 파일 "/home/roecken/mr"에서 데이터를 로드하기 시작합니다.

   SQL3500W  유틸리티가 "02/16/2004 15:12:13.392633"에
             "LOAD" 단계를 시작 중입니다.

   SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "0".

   SQL3520W  일관성 지점 로드에 성공했습니다.

   SQL3110N  유틸리티가 처리를 완료했습니다. 입력 파일에서
             "1"개의 행을 읽었습니다.

   SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "1".

   SQL3520W  일관성 지점 로드에 성공했습니다.

   SQL3515W  유틸리티가 "02/16/2004 15:12:13.445718"에
             "LOAD" 단계를 시작 중입니다.


   읽은 행 수               = 1
```

```
건너뛴 행 수              = 0
로드된 행 수              = 1
거부된 행 수              = 0
삭제된 행 수              = 0
커미트된 행 수            = 1
```

이제 하나의 백업 이미지, 하나의 로드 사본 이미지 및 하나의 로그 파일이 TSM에 있어야 합니다. zample 데이터베이스에 대한 쿼리는 다음과 같이 실행할 수 있습니다.

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
```

다음 정보가 리턴됩니다.

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

7. 상호 노드 복구를 사용하려면 다른 노드와 어카운트에 bar 컴퓨터에 있는 오브젝트에 대한 액세스 권한이 부여되어야 합니다. 이 예에서는 노드 dps와 사용자 regress9에 액세스가 부여됩니다.

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
      on nodename dps for db zample
   ```

   다음 정보가 리턴됩니다.

   ```
   Successfully added permissions for regress9 to access ZAMPLE on node dps.
   ```

   db2adutl grant 조작의 결과를 쿼리하려면 다음 명령을 실행하십시오.

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
   ```

   다음 정보가 리턴됩니다.

```
Node                 Username             Database Name   Type
-------------------------------------------------------------
DPS                  regress9             ZAMPLE          A
-------------------------------------------------------------
 Access Types:  B - backup images   L - logs   A - both
```

PASSWORDACCESS = 환경 생성

컴퓨터 2

컴퓨터 2 dps가 아직 설정되지 않았습니다. zample 데이터베이스에 대한 dps에서의 db2adutl 쿼리는 다음 결과를 리턴합니다.

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
--- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.


dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
    bar owner roecken
--- Database directory is empty ---

Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

zample 데이터베이스가 아직 dps 컴퓨터에 존재하지 않습니다.

1. zample 데이터베이스를 dps 컴퓨터로 리스토어하십시오.

```
dps:/home/regress9> db2 restore db zample use tsm options
"'-fromnode=bar -fromowner=roecken'" without prompting
```

다음 정보가 리턴됩니다.

```
DB20000I  RESTORE DATABASE 명령이 완료되었습니다.
```

주: zample 데이터베이스가 이미 dps에 존재하면, OPTIONS 매개변수를 생략하고 데이터베이스 구성 매개변수 *vendoropt*가 사용됩니다. 이 구성 매개변수는 백업 또는 리스토어 작업에 대한 OPTIONS 매개변수보다 우선합니다.

zample 데이터베이스에 대한 롤 포워드 작업은 롤 포워드 유틸리티가 로그 파일을 찾을 수 없어서 실패합니다. 다음과 같은 롤 포워드 작업은

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

다음 오류를 리턴합니다.

```
SQL4970N  "0" 노드에서 로그 파일이 누락되어 "ZAMPLE" 데이터베이스의
롤 포워드 복구가 지정된 중지점(로그의 끝 또는 특정 시점)에 도달할 수
없습니다.
```

2. 롤 포워드 유틸리티가 다른 머신에서 로그 파일을 강제로 찾도록 하려면 적절한 *logarchopt* 값을 구성해야 합니다(이 상황에서는 *logarchopt1* 데이터베이스 구성 매개변수).

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

3. 롤 포워드 유틸리티가 로드 사본 이미지를 사용할 수 있도록, *vendoropt* 데이터베이스 구성 매개변수도 설정해야 합니다.

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

4. zample 데이터베이스는 이제 롤 포워드될 수 있습니다.

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

다음 정보가 리턴됩니다.

```
                                 Rollforward Status

입력 데이터베이스 별명                = zample
상태를 리턴한 노드 수                 = 1

노드 번호                            = 0
롤 포워드 상태                       = not pending
다음에 읽을 로그 파일                 =
처리된 로그 파일                     = S0000000.LOG - S0000000.LOG
최종 커미트된 트랜잭션                = 2004-02-16-20.10.38.000000 UTC

DB20000I  ROLLFORWARD 명령이 완료되었습니다.
```

PASSWORDACCESS = 프롬프트 환경

PROMPT 환경에서는 추가 정보가 필요합니다(특히, 오브젝트가 작성된 머신의 TSM 노드 이름 및 암호).

db2adutl의 경우, dsm.sys 파일(Windows 기반 플랫폼에서 *dsm.opt* 파일)을 갱신하고 NODENAME bar(bar은 소스 컴퓨터의 이름이므로)을 서버 절에 추가하십시오.

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
owner roecken password *******
```

다음 정보가 리턴됩니다.

```
Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

1. 데이터베이스가 없으면 비어 있는 zample 데이터베이스를 작성하십시오. zample 데이터베이스가 이미 존재하면 이 단계와 데이터베이스 구성을 갱신하는 다음 단계를 건너뛸 수 있습니다.

   ```
   dps:/home/regress9> db2 create db zample
   ```

2. zample 데이터베이스의 데이터베이스 구성 매개변수 *tsm_nodename*을 갱신하십시오.

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

3. zample 데이터베이스의 데이터베이스 구성 매개변수 *tsm_password*를 갱신하십시오.

   ```
   dps:/home/regress9> db2 update db cfg for zample using
   tsm_password ********
   ```

4. zample d데이터베이스를 리스토어하십시오.

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   리스토어 작업이 성공적으로 완료되지만 경고가 발행됩니다.

   ```
   SQL2540W  리스토어는 성공적이었으나, 인터럽트 없음 모드에서 처리하는
   동안 데이터베이스 리스토어 중에 경고 "2539"이(가)
   발견되었습니다.
   ```

   다시, 이 때 롤 포워드 유틸리티는 올바른 로그 파일을 찾을 수 없습니다.

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   다음 오류 메시지가 리턴됩니다.

   ```
   SQL1268N  "0" 노드에서 "ZAMPLE" 데이터베이스에 대해 로그 파일
   "S0000000.LOG"를 검색하는 동안 -2112880618" 오류가 발생하여 롤 포워드
   복구가 중지되었습니다.
   ```

5. 데이터베이스 리스토어 작업이 데이터베이스 구성 파일을 교체하므로, TSM 데이터베이스 구성 값을 올바른 값으로 설정해야 합니다. 먼저 *tsm_nodename* 구성 매개변수를 재설정해야 합니다.

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

6. *tsm_password* 데이터베이스 구성 매개변수를 재설정해야 합니다.

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
   ```

7. *logarchopt1* 데이터베이스 구성 매개변수를 재설정하여, 롤 포워드 유틸리티가 올바른 로그 파일을 찾을 수 있도록 해야 합니다.

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

8. 로드 복구 파일도 사용할 수 있도록 *vendoropt* 데이터베이스 구성 매개변수도 재설정해야 합니다.

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

9. 데이터베이스 구성 매개변수가 설정되면, 데이터베이스는 롤 포워드될 수 있습니다.

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   zample 데이터베이스의 ROLLFORWARD QUERY STATUS 명령은 다음을 표시합니다.

```
                                 Rollforward Status

입력 데이터베이스 별명                = zample
상태를 리턴한 노드 수                 = 1

노드 번호                             = 0
롤 포워드 상태                        = not pending
다음에 읽을 로그 파일                 =
처리된 로그 파일                      = S0000000.LOG - S0000000.LOG
최종 커미트된 트랜잭션                = 2004-02-16-20.10.38.000000 UTC

DB20000I  ROLLFORWARD 명령이 완료되었습니다.
```

파티션된 데이터베이스 환경에서 클럭 동기화

데이터베이스 파티션 서버 사이에 상대적으로 동기화된 시스템 클럭이 유지되도록 하여 원활한 데이터베이스 조작 및 무제한 포워드 복구성이 가능하도록 해야 합니다. 데이터베이스 파티션 서버 사이의 시간 차이와, 트랜잭션에 대해 가능한 작동 및 통신 지연은 *max_time_diff*(노드 사이의 최대 시간 차이) 데이터베이스 관리 프로그램 구성 매개변수에 지정된 값보다 작아야 합니다.

로그 레코드 시간소인에 파티션된 데이터베이스 환경에서 트랜잭션의 시퀀스를 반영하기 위해 DB2는 각 머신의 SQLOGCTL.LFH 파일에 저장된 시스템 클럭 시간소인을 로그 레코드 시간소인의 기준으로 사용합니다. 그러나 시스템 클럭이 우선 설정된 경우에는 로그 클럭이 시스템 클럭에 따라 자동으로 설정됩니다. 시스템 클럭을 다시 설정할 수 있어도, 로그 클럭은 다시 설정할 수 없으므로 시스템 클럭이 이 시간과 일치할 때까지 동일한 진행 시간으로 유지됩니다. 그런 다음 클럭이 동기화됩니다. 이는 데이터베이스 노드에서의 단기 시스템 클럭 오류가 데이터베이스 로그 시간소인에는 오래 지속되는 영향을 줄 수 있다는 것을 의미합니다.

예를 들어, 연도가 2003년일 때 데이터베이스 파티션 서버 A에서의 시스템 클럭이 잘못하여 2005년 11월 7일로 설정되고, 그 실수가 해당 데이터베이스 파티션 서버에 있는 데이터베이스 파티션에서 갱신 트랜잭션이 커미트된 *후에* 정정되었다고 가정합니다. 데이터베이스가 계속 사용 중이고 시간이 경과하면서 정기적으로 갱신되는 경우, 2003년 11월 7일과 2005년 11월 7일 사이의 임의 시점은 가상으로 롤 포워드 복구를 통해 도달할 수 없습니다. 데이터베이스 파티션 서버 A에서의 COMMIT가 완료되면, 데이터베이스 로그의 시간 소인은 2005로 설정되고, 로그 클럭은 시스템 클럭이 2005년 11월 7일 시간과 일치할 때까지 2005년 11월 7일로 유지됩니다. 이 시간 프레임 내의 특정 시점으로 롤 포워드하려고 하면 조작은 지정된 중지점(2003년 11월 7일)을 넘은 첫 번째 시간소인에서 중지합니다.

DB2가 시스템 클럭에 대한 갱신사항을 제어할 수 없어도, *max_time_diff* 데이터베이스 관리 프로그램 구성 매개변수는 이 유형의 문제점이 발생할 수 있는 기회를 감소시킵니다.

- 이 매개변수의 구성 가능 값 범위는 1분 - 24시간입니다.

- 비카탈로그 파티션에 대해 첫 번째 연결 요청이 작성될 때 데이터베이스 파티션 서버는 해당 시간을 데이터베이스의 카탈로그 파티션으로 보냅니다. 그러면 카탈로그 파티션은 연결을 요청하는 데이터베이스 파티션의 시간과 자체의 시간이 *max_time_diff* 매개변수에 지정된 범위 내에 있는지 점검합니다. 이 범위를 초과하면 연결이 거부됩니다.
- 데이터베이스에서 세 개 이상의 데이터베이스 파티션 서버와 관련되는 갱신 트랜잭션은 갱신이 커미트되기 전에 참여하는 데이터베이스 파티션 서버의 클럭이 동기화된 상태인지 검증해야 합니다. 두 개 이상의 데이터베이스 파티션 서버가 *max_time_diff* 에서 허용하는 한계를 초과하는 시간 차이를 수반하는 경우, 트랜잭션은 올바르지 않은 시간이 다른 데이터베이스 파티션 서버로 전달되지 않도록 롤백됩니다.

제 19 장 문제점 해결

DB2 데이터베이스 문제점 해결

일반적으로 문제점 해결 프로세스에는 문제점을 분리 및 식별한 후 해결책을 찾아야 합니다. 이 섹션에서는 DB2 제품의 특정 기능과 관련된 문제점 해결 정보를 제공합니다.

일반적인 문제점이 식별되면, 찾은 내용이 점검목록 양식으로 이 섹션에 추가됩니다. 점검목록을 통해 해결책을 찾을 수 없으면, 추가 진단 데이터를 수집하여 직접 분석하거나 분석을 위해 IBM Software Support에 데이터를 제출할 수 있습니다.

다음 질문을 통해 적절한 문제점 해결 태스크를 수행할 수 있습니다.

1. 알려진 FixPack을 모두 적용했습니까? 그렇지 않다면, *DB2 Server 설치*의 『FixPack 적용』을 고려하십시오.
2. 다음을 수행할 때 문제점이 발생합니까?
   - DB2 데이터베이스 서버 또는 클라이언트 설치. 그렇다면, 이 책의 『설치 문제점에 대한 데이터 수집』 주제를 참조하십시오.
   - 인스턴스 또는 DB2 Administration Server(DAS) 작성, 삭제, 갱신 또는 업그레이드. 그렇다면, 이 책의 『DAS 및 인스턴스 관리 문제점에 대한 데이터 수집』 주제를 참조하십시오.
   - EXPORT, IMPORT, LOAD 또는 db2move 명령을 사용하여 데이터 이동. 그렇다면, 이 책의 『데이터 이동 문제점에 대한 데이터 수집』 주제를 참조하십시오.

문제점이 이러한 범주 중 하나에 해당하지 않는 경우에는 IBM Software Support에 문의하는 경우 기본 진단 데이터가 여전히 필요할 수 있습니다. 을 수행해야 합니다.

파티션된 데이터베이스 환경 문제점 해결

파티션된 데이터베이스 환경에서 명령 실행

파티션된 데이터베이스 환경에서 인스턴스의 컴퓨터 또는 데이터베이스 파티션 서버(노드)에서 실행되도록 명령을 발행할 수 있습니다. 이 경우, rah 명령 또는 db2_all 명령을 사용하십시오. rah 명령을 사용하면 인스턴스의 컴퓨터에서 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 환경에만 적용됩니다.

Windows에서 rah 명령 또는 db2_all 명령을 실행하려면 관리자 그룹의 구성원이 되는 사용자 어카운트로 로그온하십시오.

Linux 및 UNIX 플랫폼에서 로그인 쉘이 Korn 쉘 또는 다른 쉘이 될 수 있으나, 서로 다른 쉘은 특수 문자를 포함하는 명령을 처리하는 방법에서 차이를 나타냅니다.

또한 Linux 및 UNIX 플랫폼에서 rah는 **DB2RSHCMD** 레지스트리 변수로 지정된 리모트 쉘 프로그램을 사용합니다. 두 개의 리모트 쉘 프로그램, ssh(추가 보안이 필요한 경우) 또는 rsh(또는 HP-UX의 경우 remsh) 중에서 선택할 수 있습니다. **DB2RSHCMD**가 설정되지 않으면 rsh(또는 HP-UX의 경우 remsh)가 사용됩니다. ssh 리모트 쉘 프로그램은 UNIX 운영 체제 환경에서 텍스트 상태의 암호 전송을 막는데 사용됩니다. DB2 DPF 인스턴스에 사용할 ssh의 단일 버전을 구성하는 방법에 대한 자세한 설명은 다음 항목을 참조하십시오. ″OpenSSH를 사용하려면 DB2 Universal Database for UNIX를 구성하십시오.″

명령 범위를 판별하려면 명령이 단일 데이터베이스 파티션 서버에서 실행되는지 또는 이들 전체에서 실행되는지를 보여주는 *명령어 참조서*를 참조하십시오. 명령이 하나의 데이터베이스 파티션 서버에서 실행되고 있을 때 명령이 이들 전체에서 실행되게 하려면 db2_all을 사용하십시오. 예외는 컴퓨터의 모든 논리 노드(데이터베이스 파티션 서버)에서 실행되는 db2trc 명령입니다. 모든 컴퓨터의 모든 논리 노드에서 db2trc를 실행하려면 rah를 사용하십시오.

제 4 부 성능 문제

제 20 장 데이터베이스 설계 시 성능 문제

성능 확장 기능

테이블 파티션 및 다중 클러스터링 테이블

한 테이블이 다차원 클러스터링 및 파티션될 수 있습니다. 다차원 클러스터링되면서 또한 파티션되는 테이블에서 컬럼을 테이블 파티션 범위 파티션 스펙 및 MDC 키에서 사용할 수 있습니다. 이는 어느 하나의 함수만 사용하여 달성할 수 있는 것보다 더 정교한 데이터 파티션 및 블록 제거를 달성하는데 유용합니다. 또한 테이블이 파티션되는 곳과는 다른 컬럼을 MDC 키에 대해 지정하는 것이 유용한 많은 응용프로그램이 있습니다. MDC는 다차원인데 반해 테이블 파티션은 다중 컬럼임을 주의해야 합니다. 파티션된 테이블의 인덱스는 파티션되거나 파티션되지 않을 수 있습니다. MDC 블록 인덱스는 항상 파티션되지 않습니다.

메인스트림 DB2 데이터 웨어하우스의 특성

다음 권장사항은 DB2 V9.1에서 새로운 기능인 전형적인 메인스트림 웨어하우스에 초점을 두었습니다. 다음 특성이 가정됩니다.

- 데이터베이스가 다중 머신 또는 다중 AIX 논리적 파티션에서 실행됩니다.
- 데이터베이스 파티션 기능(DPF)이 사용됩니다(테이블이 DISTRIBUTE BY HASH 절을 사용하여 작성됩니다).
- 4 - 50개의 데이터 파티션이 있습니다.
- MDC 및 테이블 파티션이 고려되고 있는 테이블이 주요 사실 테이블입니다.
- 테이블이 1억 - 1000억 개의 행을 갖습니다.
- 새 데이터가 다양한 시간 프레임(야간, 주별, 월별)에 로드됩니다.
- 일별 처리 볼륨은 1만 - 1000만 레코드입니다.
- 데이터 볼륨은 변합니다. 가장 큰 월은 가장 작은 월 크기의 5X입니다. 마찬가지로 가장 큰 차원(제품 행, 지역)은 크기 범위의 5X입니다.
- 1 - 5년의 자세한 데이터가 보유됩니다.
- 만기된 데이터는 월별 또는 분기별로 롤아웃됩니다.
- 테이블은 광범위한 쿼리 유형을 사용합니다. 그러나 워크로드는 대부분 OLTP 워크로드에 상대적으로 다음 특성을 갖는 분석 쿼리입니다.
  - 최고 2백만 행을 갖는 더 큰 결과 세트
  - 대부분 또는 모든 쿼리는 기본 테이블이 아니라 적중하는 보기
- 범위(BETWEEN절), 목록의 항목 등으로 데이터를 선택하는 SQL절

메인스트림 DB2 V9.1 데이터 웨어하우스 사실 테이블의 특성

일반적인 웨어하우스 사실 테이블은 다음 설계를 사용할 수 있습니다.

- 월 컬럼에 데이터 파티션을 작성하십시오.
- 롤 아웃하려는 각 기간(예: 한 달, 3개월)에 대한 데이터 파티션을 정의하십시오.
- 일에 대한 MDC 차원을 작성하고 1 - 4개의 추가 차원을 작성하십시오. 일반적인 차원은 제품 행과 지역입니다.
- 모든 데이터 파티션 및 MDC 클러스터가 모든 데이터베이스 파티션에 분산됩니다.

MDC 및 테이블 파티션은 중첩되는 일련의 이점을 제공합니다. 다음 표는 조직에서 잠재적인 요구를 나열하고 이전에 식별된 특성을 기초로 권장되는 조직 스킴을 식별합니다.

표 15. MDC 테이블에 테이블 파티션 사용

문제점	권장 스킴	권장사항
롤아웃 동안 데이터 사용 가능성	테이블 파티션	DETACH PARTITION절을 사용하여 최소한의 장애로 대형 데이터 롤아웃
쿼리 성능	테이블 파티션 및 MDC	다중 차원 쿼리를 위해 MDC가 최선입니다. 테이블 파티션이 데이터 파티션을 제거하여 돕습니다.
최소 재구성	MDC	MDC가 클러스터링을 유지보수하여 재구성의 필요성을 줄입니다.
일반적인 오프라인 창 동안 1개월 이상의 데이터를 롤아웃합니다.	테이블 파티션	데이터 파티션이 이 필요성을 전적으로 다룹니다. MDC는 아무 것도 추가하지 않으며 그 자체로는 적합하지 않습니다.
마이크로 오프라인 창(1분 미만) 동안 1개월 이상의 데이터를 롤아웃합니다.	테이블 파티션	데이터 파티션이 이 필요성을 전적으로 다룹니다. MDC는 아무 것도 추가하지 않으며 그 자체로는 적합하지 않습니다.
어떤 서비스도 중지하지 않으면서 쿼리를 제출하는 비즈니스 사용자에 대해 테이블을 완전히 사용 가능하게 유지하면서 1개월 이상의 데이터를 롤아웃합니다.	MDC	MDC만 이 필요성을 다소 다룹니다. 테이블 파티션은 테이블이 오프라인이 되는 짧은 기간으로 인해 적합하지 않을 수 있습니다.
매일 데이터 로드(ALLOW READ ACCESS 또는 ALLOW NO ACCESS)	테이블 파티션 및 MDC	MDC가 이 이점의 대부분을 제공합니다. 테이블 파티션은 증분 이점을 제공합니다.
"연속" 데이터 로드(ALLOW READ ACCESS)	테이블 파티션 및 MDC	MDC가 이 이점의 대부분을 제공합니다. 테이블 파티션은 증분 이점을 제공합니다.
"전통적인 BI" 쿼리에 대한 쿼리 실행 성능	테이블 파티션 및 MDC	MDC가 큐브/다차원 쿼리에 특히 좋습니다. 테이블 파티션은 파티션 제거를 통해 도움이 됩니다.

표 15. MDC 테이블에 테이블 파티션 사용 (계속)

문제점	권장 스킴	권장사항
재구성 필요성을 피하거나 태스크 수행과 연관된 어려움을 줄여서 재구성 어려움을 최소화합니다.	MDC	MDC는 재구성할 필요성을 줄이는 클러스터링을 유지보수합니다. MDC가 사용되는 경우 데이터 파티션은 증분 이점을 제공하지 않습니다. 그러나 DC가 사용되지 않는 경우 테이블 파티션이 일부 과정 그레인 클러스터링을 파티션 레벨에서 유지보수하여 재구성 필요성을 줄이는데 도움이 됩니다.

예 1:

키 컬럼 YearAndMonth 및 Province를 갖는 테이블을 고려하십시오. 이 테이블을 플랜하는 합리적인 접근법은 데이터 파티션당 2개월의 날짜를 파티션하는 것일 수 있습니다. 또한 Province별로 구성하여, 임의의 날짜 범위 2개월 내에서 특정 지방에 대한 모든 행이 37 페이지의 그림 6에 표시된 것처럼 함께 클러스터되도록 할 수 있습니다.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

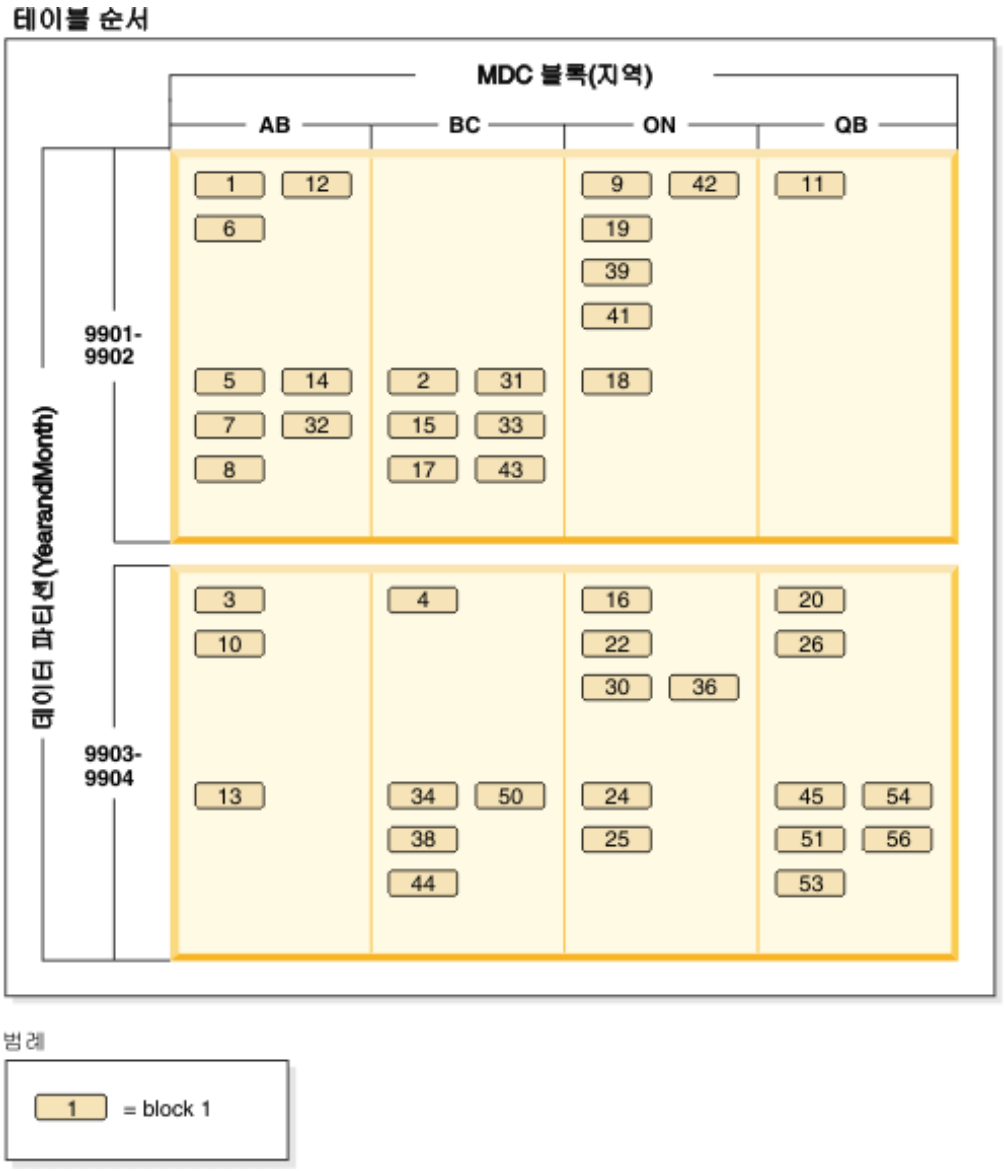


그림 46. YearAndMonth로 파티션되고 Province로 구성되는 테이블

예 2

38 페이지의 그림 7에서 보는 것처럼 ORGANIZE BY절에 YearAndMonth를 추가하여 더 정교한 세분화도를 달성할 수 있습니다.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

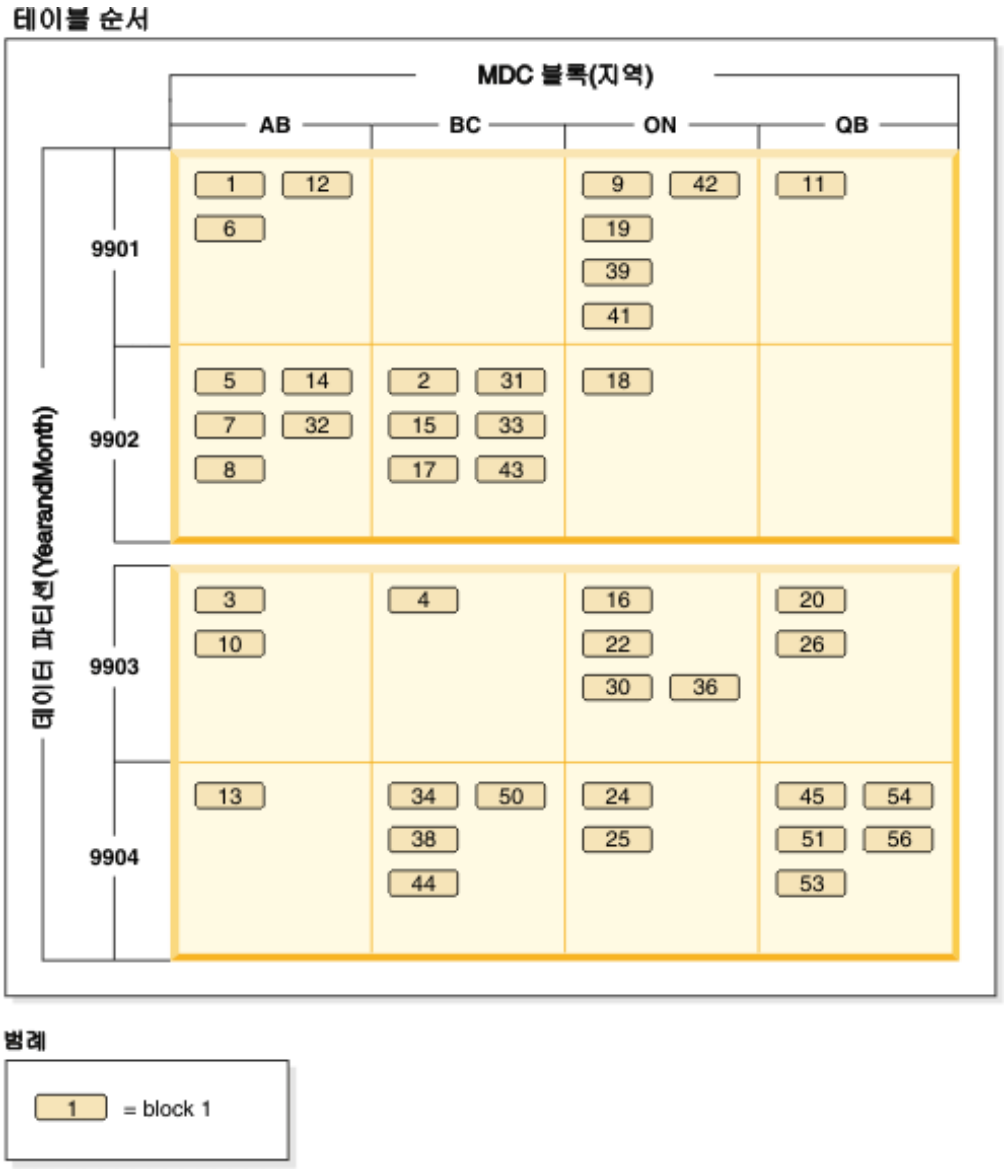


그림 47. YearAndMonth로 파티션되고 Province 및 YearAndMonth별로 구성되는 테이블

각 범위에 단일 값만 존재하도록 파티션되는 경우에 MDC 키에 테이블 파티션 컬럼을 포함시켜서 얻는 것이 없습니다.

고려사항

- 기본 테이블과 비교할 때, MDC 테이블 및 파티션된 테이블이 둘 다 추가 스토리지가 필요합니다. 이러한 스토리지 필요성은 부가적이지만, 이점이 있는 타당한 것으로 간주됩니다.
- 파티션된 데이터베이스 환경에서 MDC 기능을 테이블 파티션과 결합하지 않을 것을 선택하는 경우, 일반적으로 여기서 논의하는 시스템 유형에 해당하는 경우인 데이터 분산을 자신있게 예측할 수 있는 경우에 테이블 파티션이 최적입니다. 그렇지 않으면, MDC를 고려해야 합니다.

파티션된 테이블에 대한 최적화 전략

데이터 파티션 제거는 쿼리 술어를 기반으로 테이블에서 데이터 파티션의 서브세트에만 액세스하여 쿼리에 응답해야 함을 결정하는 데이터베이스 서버의 기능을 참조합니다. 데이터 파티션 제거는 특히 파티션된 테이블에 대해 결정 지원 쿼리를 실행할 때 유용합니다.

파티션된 테이블은 테이블의 하나 이상의 테이블 파티션 키 컬럼의 값에 따라 테이블 데이터가 데이터 파티션 또는 범위라고 하는 여러 스토리지 오브젝트로 나누어지는 데이터 구성 스킴을 사용합니다. 테이블의 데이터는 CREATE TABLE문의 PARTITION BY 절에서 제공된 스펙을 기반으로 여러 스토리지 오브젝트로 파티션됩니다. 이러한 스토리지 오브젝트는 서로 다른 테이블 스페이스, 동일한 테이블 스페이스 또는 둘의 조합에 있을 수 있습니다.

다음 예에서는 데이터 파티션 제거의 성능 이점을 보여줍니다.

```
create table custlist(
  subsdate date, province char(2), accountid int)
  partition by range(subsdate) (
    starting from '1/1/1990' in ts1,
    starting from '1/1/1991' in ts1,
    starting from '1/1/1992' in ts1,
    starting from '1/1/1993' in ts2,
    starting from '1/1/1994' in ts2,
    starting from '1/1/1995' in ts2,
    starting from '1/1/1996' in ts3,
    starting from '1/1/1997' in ts3,
    starting from '1/1/1998' in ts3,
    starting from '1/1/1999' in ts4,
    starting from '1/1/2000' in ts4,
    starting from '1/1/2001'
    ending '12/31/2001' in ts4)
```

2000년의 고객 정보에만 관심이 있다고 가정해봅니다.

```
select * from custlist
  where subsdate between '1/1/2000' and '12/31/2000'
```

333 페이지의 그림 48에 표시된 대로, 데이터베이스 서버가 테이블 스페이스 TS4에서 하나의 데이터 파티션에만 액세스하여 이 쿼리를 해석해야 함을 결정합니다.

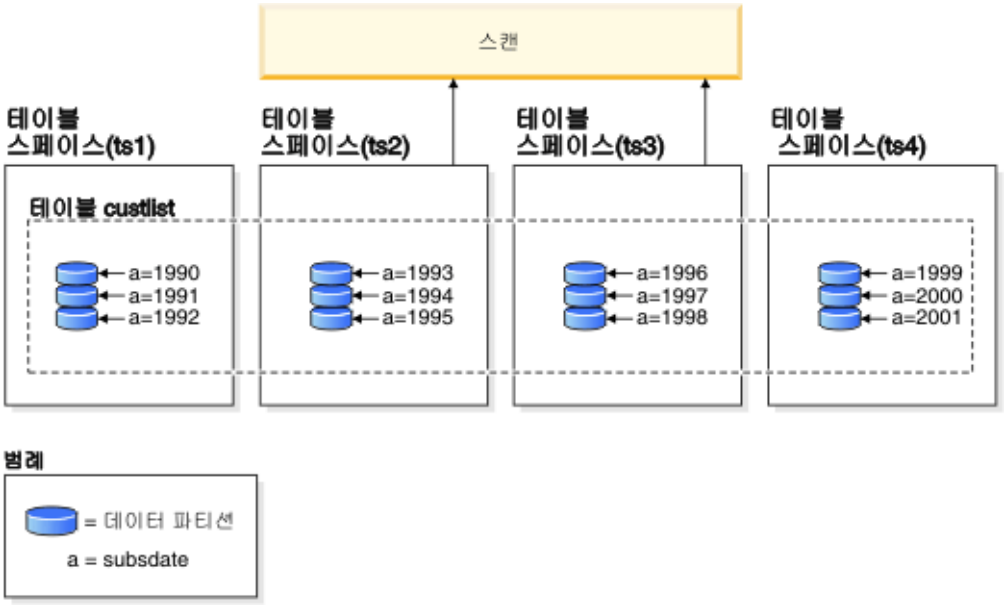


그림 48. 데이터 파티션 제거의 성능 이점

데이터 파티션 제거의 또 다른 예는 다음 스킴을 기반으로 합니다.

```
create table multi (
  sale_date date, region char(2))
  partition by (sale_date) (
    starting '01/01/2005'
    ending '12/31/2005'
    every 1 month)

create index sx on multi(sale_date)

create index rx on multi(region)
```

다음 쿼리를 발행한다고 가정해봅니다.

```
select * from multi
  where sale_date between '6/1/2005'
    and '7/31/2005' and region = 'NW'
```

테이블 파티셔닝 없이, 하나의 가능한 플랜은 인덱스 ANDing입니다. Index ANDing은 다음 태스크를 수행합니다.

- 각 인덱스에서 모든 관련 인덱스 항목을 읽습니다.
- 행 ID(RID) 두 세트를 모두 저장합니다.
- RID를 일치하여 두 인덱스에서 발생하는 항목을 판별합니다.
- RID를 사용하여 행을 페치합니다.

334 페이지의 그림 49에서 설명된 대로, 테이블 파티셔닝을 통해 인덱스를 읽어 REGION 및 SALE_DATE 모두에 대한 일치 항목을 찾음으로써 일치하는 행을 신속하게 검색

할 수 있습니다.

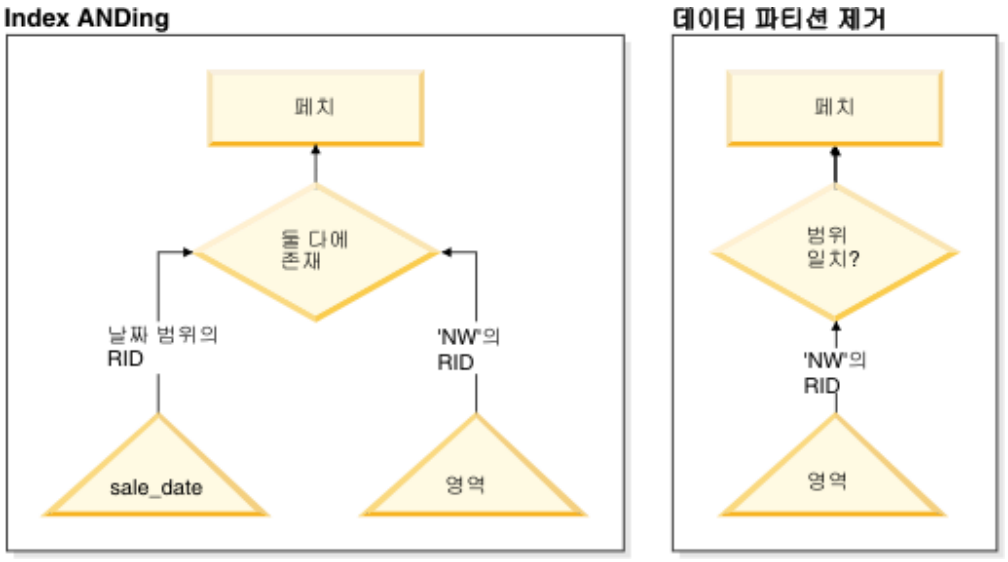


그림 49. 테이블 파티셔닝 및 인덱스 ANDing에 대한 옵티마이저 의사 결정 경로

DB2 Explain

Explain 기능을 사용하여 쿼리 옵티마이저에 의해 선택된 데이터 파티션 제거 플랜을 결정할 수도 있습니다. 『DP Elim Predicates』 정보에서는 다음 쿼리를 해석하기 위해 스캔되는 데이터 파티션을 보여줍니다.

```
   select * from custlist
     where subsdate between '12/31/1999' and '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
    FALSE
DPLSTPRT: (List of data partitions accessed)
    9-11
DPNUMPRT: (Number of data partitions accessed)
   3

DP Elim Predicates:
------------------
Range 1)
  Stop  Predicate: (Q1.A <= '01/01/2001')
 Start Predicate: ('12/31/1999' <= Q1.A)



Objects Used in Access Plan:
 --------------------------

  Schema: MRSRINI
  Name:    CUSTLIST
  Type:    Data Partitioned Table
   Time of creation:      2005-11-30-14.21.33.857039
```

```
Last statistics update:    2005-11-30-14.21.34.339392
 Number of columns:      3
 Number of rows:      100000
Width of rows:      19
Number of buffer pool pages:    1200
 Number of data partitions:    12
 Distinct row values:      No
 Tablespace name:      <VARIOUS>
```

복수 컬럼 지원

데이터 파티션 제거는 복수 컬럼이 테이블 파티션 키로 사용될 경우 작동합니다. 예를 들어, 다음과 같습니다.

```
create table sales (
  year int, month int)
 partition by range(year, month) (
    starting from (2001,1)
    ending at (2001,3) in ts1,
   ending at (2001,6) in ts2,
   ending at (2001,9) in ts3,
   ending at (2001,12) in ts4,
   ending at (2002,3) in ts5,
   ending at (2002,6) in ts6,
   ending at (2002,9) in ts7,
   ending at (2002,12) in ts8)

select * from sales where year = 2001 and month < 8
```

쿼리 옵티마이저에서 TS1, TS2 및 TS3의 데이터 파티션에만 액세스하여 이 쿼리를 해석해야 한다고 추론합니다.

주: 복수 컬럼이 테이블 파티션 키를 구성하는 경우, 테이블 파티션 키에 사용되는 비선행 컬럼이 독립적이지 않기 때문에 복합 키의 선행 컬럼에 술어가 있는 경우에만 데이터 파티션 제거가 가능합니다.

다중 범위 지원

다중 범위가 있는 데이터 파티션(즉, 함께 OR되는 데이터 파티션)에서 데이터 파티션 제거가 가능합니다. 위의 예에서 작성된 SALES 테이블을 사용하여 다음 쿼리를 실행하십시오.

```
select * from sales
  where (year = 2001 and month <= 3)
    or (year = 2002 and month >= 10)
```

데이터베이스 서버는 2001년의 첫 번째 분기 및 2002년의 마지막 분기의 데이터에만 액세스합니다.

생성된 컬럼

생성된 컬럼을 테이블 파티션 키로 사용할 수 있습니다. 예를 들어, 다음과 같습니다.

```
create table sales (
  a int, b int generated always as (a / 5))
  in ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10
  partition by range(b) (
    starting from (0)
    ending at (1000) every (50))
```

이 경우, 생성된 컬럼의 술어가 데이터 파티션 제거에 사용됩니다. 또한 컬럼 생성에 사용된 표현식이 단순할 경우, 데이터베이스 서버에서 소스 컬럼의 술어를 생성된 컬럼의 술어로 변환하여 생성된 컬럼에서 데이터 파티션 제거를 가능하게 합니다. 예를 들어, 다음과 같습니다.

```
select * from sales where a > 35
```

데이터베이스 서버가 a(a > 35)에서 b(b > 7)에 추가 술어를 생성하여 데이터 파티션 제거를 허용합니다.

Join 술어

Join 술어가 테이블 액세스 등급으로 밀려 내려가는 경우 데이터 파티션 제거에서 Join 술어가 사용될 수도 있습니다. Join 술어는 중첩된 루프 조인(NLJN)의 내부 조인에서 테이블 액세스 등급으로만 밀려 내려갑니다.

다음 테이블을 고려해보겠습니다.

```
create table t1 (a int, b int)
  partition by range(a,b) (
    starting from (1,1)
    ending (1,10) in ts1,
    ending (1,20) in ts2,
    ending (2,10) in ts3,
    ending (2,20) in ts4,
    ending (3,10) in ts5,
    ending (3,20) in ts6,
    ending (4,10) in ts7,
    ending (4,20) in ts8)

create table t2 (a int, b int)
```

다음 두 개의 술어가 사용됩니다.

```
P1: T1.A = T2.A
P2: T1.B > 15
```

이 예에서는 조인의 외부 값을 알 수 없기 때문에 컴파일 시간에 액세스되는 정확한 데이터 파티션을 판별할 수 없습니다. 이 경우 및 호스트 변수 또는 매개변수 표시문자가 사용되는 경우, 필요한 값이 바인드될 때 런타임에서 데이터 파티션 제거가 발생합니다.

런타임 동안 T1이 NLJN의 내부인 경우, 술어를 기반으로 T2.A의 모든 외부 값에 대해 데이터 파티션 제거가 동적으로 발생합니다. 런타임 동안, 액세스될 테이블 스페이스 TS6에 데이터 파티션을 규정하는 외부 값 T2.A = 3에 대해 술어 T1.A = 3 및 T1.B > 15가 적용됩니다.

테이블 T1 및 T2의 컬럼 A에 다음 값이 있다고 가정해봅니다.

외부 테이블 **T2:** 컬럼 **A**	내부 테이블 **T1:** 컬럼 **A**	내부 테이블 **T1:** 컬럼 **B**	내부 테이블 **T1:** 데이터 파티션 위치
2	3	20	TS6
3	2	10	TS3
3	2	18	TS4
	3	15	TS6
	1	40	TS3

중첩된 루프 조인을 수행하기 위해 (내부 테이블에 대한 테이블 스캔 가정), 데이터베이스 관리 프로그램은 다음 단계를 수행합니다.

1. T2에서 첫 번째 행을 읽습니다. A의 값은 2입니다.
2. T2.A 값(2)을 Join 술어 T1.A = T2.A의 컬럼 T2.A에 바인드합니다. 술어가 T1.A = 2가 됩니다.
3. 술어 T1.A = 2 및 T1.B > 15를 사용하여 데이터 파티션 제거를 적용합니다. 이렇게 하면 테이블 스페이스 TS4에 데이터 파티션이 규정됩니다.
4. T1.A = 2 및 T1.B > 15를 적용한 후, 행이 발견될 때까지 테이블 T1의 테이블 스페이스 TS4에서 데이터 파티션을 스캔합니다. 발견된 첫 번째 규정 행은 T1의 행 3입니다.
5. 일치하는 행을 조인합니다.
6. 다음 일치(T1.A = 2 및 T1.B > 15)가 발견될 때까지 테이블 T1의 테이블 스페이스 TS4에서 데이터 파티션을 스캔합니다. 행이 더 이상 발견되지 않습니다.
7. T2의 모든 행이 처리될 때까지 다음 행 T2에 대해 1-6단계를 반복합니다(A 값을 3으로 교체).

XML 데이터에 대한 인덱스

파티션되지 않은 XML 인덱스는 파티션된 테이블의 기타 관계형 인덱스가 유지보수되는 것과 동일한 방법으로 테이블 삽입, 갱신 및 삭제 조작 동안 데이터베이스 관리 프로그램에 의해 유지보수됩니다. 파티션된 테이블에서 XML 데이터에 대한 파티션되지 않은 인덱스는 파티션되지 않은 테이블에서 XML 데이터에 대한 인덱스와 동일한 방법으로 사용되어 쿼리 처리 속도를 높입니다. 쿼리 술어를 사용하여 파티션된 테이블에서 데이터 파티션의 서브세트에만 액세스하여 쿼리에 응답해야 함을 결정할 수 있습니다.

XML 컬럼에 대한 인덱스와 데이터 파티션 제거는 함께 작동하여 쿼리 성능을 향상시킬 수 있습니다. 다음 파티션된 테이블을 고려해보십시오.

```
create table employee (a int, b xml, c xml)
  index in tbspx
  partition by (a) (
    starting 0 ending 10,
    ending 20,
    ending 30,
    ending 40)
```

Now consider the following query:

```
select * from employee
  where a > 21
  and xmlexist('$doc/Person/Name/First[.="Eric"]'
    passing "EMPLOYEE"."B" as "doc")
```

옵티마이저는 술어 a > 21을 기반으로 처음 두 파티션을 즉시 제거할 수 있습니다. 쿼리 플랜에서 옵티마이저에 의해 컬럼 B의 XML 데이터에 대한 파티션되지 않은 인덱스가 선택된 경우, XML 데이터에 대한 인덱스를 사용한 인덱스 스캔에서 옵티마이저의 데이터 파티션 제거 결과를 이용할 수 있고 관계형 데이터 파티션 제거 술어에 의해 제거되지 않은 파티션에 속하는 결과만 리턴할 수 있습니다.

MDC 테이블에 대한 최적화 전략

다차원적으로 클러스터된(MDC) 테이블을 작성하는 경우 옵티마이저는 추가 최적화 전략을 적용할 수 있으므로 여러 쿼리의 성능이 향상될 수 있습니다. 이러한 전략은 주로 블록 인덱스의 향상된 효율성에 기초하지만 여러 차원에서 클러스터링하는 장점으로 인해 데이터 검색 속도도 빨라집니다.

MDC 테이블 최적화 전략은 파티션 내 병렬 처리 및 파티션 간 병렬 처리의 성능 이점도 활용할 수 있습니다. 다음과 같은 MDC 테이블의 특정 이점을 참조하십시오.

- 차원 블록 인덱스 찾아보기는 테이블의 필수 분할 영역을 식별하고 필수 블록만 신속하게 스캔할 수 있습니다.
- 블록 인덱스는 레코드 ID(RID) 인덱스보다 작으므로 찾아보기 속도가 빠릅니다.
- 인덱스 AND 및 OR 작업을 블록 레벨에서 수행하고 RID와 결합할 수 있습니다.
- 데이터를 Extent에서 클러스터링할 수 있으므로 검색 속도가 빨라집니다.
- 롤아웃을 사용할 수 있는 경우 행을 신속하게 삭제할 수 있습니다.

REGION 및 MONTH 컬럼에 정의된 차원을 사용하여 SALES라고 하는 MDC 테이블에 대한 다음의 간단한 예를 참조하십시오.

```
select * from sales
  where month = 'March' and region = 'SE'
```

이 쿼리에서 옵티마이저는 차원 블록 인덱스 찾아보기를 수행하여 March와 SE 영역이 발생한 블록을 찾습니다. 그런 다음 이러한 블록만 스캔하여 결과 세트를 신속하게 페치합니다.

롤아웃 삭제

조건이 롤아웃을 사용하여 삭제를 허용하는 경우 이와 같이 MDC 테이블에서 행을 삭제하는 효율적인 방법이 사용됩니다. 필수 조건은 다음과 같습니다.

- DELETE문은 위치 지정된 DELETE가 아닌 검색된 DELETE입니다(명령문이 WHERE CURRENT OF절을 사용하지 않음).
- WHERE절이 없거나(모든 행을 삭제함) WHERE절의 조건만 차원이 적용됩니다.
- 테이블은 DATA CAPTURE CHANGES절을 사용하여 정의되어 있지 않습니다.
- 테이블은 참조 무결성 관계에서 상위가 아닙니다.
- 테이블에 ON DELETE 트리거가 정의되어 있지 않습니다.
- 테이블은 즉시 새로 고친 MQT에서 사용되지 않습니다.
- 외부 키가 테이블의 차원 컬럼에 대한 서브세트인 경우 연쇄 삭제 조작을 롤아웃할 수 있습니다.
- 트리거링 SQL 조작(CREATE TRIGGER문의 OLD TABLE AS절에 지정됨) 이전에 영향을 받은 행 세트를 식별하는 임시 테이블에 대해 실행되는 SELECT문에 DELETE문을 표시할 수 없습니다.

롤아웃 삭제 시, 삭제된 레코드는 로그되지 않습니다. 대신에, 페이지의 일부를 재형식화하여 레코드를 포함하는 페이지를 비어 있는 것처럼 보이게 합니다. 다시 형식화된 부분의 변경사항은 로그되지만 레코드 자체는 로그되지 않습니다.

디폴트 동작 *즉시 정리 롤아웃*은 삭제 시 RID 인덱스를 정리합니다.
DB2_MDC_ROLLOUT 레지스트리 변수를 IMMEDIATE로 설정하거나 SET CURRENT MDC ROLLOUT MODE문에서 IMMEDIATE를 지정하여 이 모드를 지정할 수도 있습니다. 표준 삭제 조작과 비교하여 인덱스 갱신의 로깅에는 변경이 없으므로 성능 향상은 여기에 포함된 RID 인덱스의 수에 따라 다릅니다. 총 시간 및 로그 스페이스의 퍼센트에 따라 RID 인덱스의 수가 적으면 향상 수준이 높습니다.

절약된 로그 스페이스의 양은 다음 공식을 사용하여 추정할 수 있습니다.

```
S + 38*N - 50*P
```

여기서 *N*은 삭제된 레코드의 수이고 *S*는 널(NULL) 표시기 및 VARCHAR 길이와 같은 오버헤드를 포함하여 삭제된 레코드의 전체 크기이고 *P*는 삭제된 레코드가 들어 있는 블록의 페이지 수입니다. 이 수치는 실제 로그 데이터의 감소를 나타냅니다. 롤백을 위해 예약된 스페이스가 절약되었으므로 필요한 활성 로그 스페이스에서 절약되는 양은 해당 값의 두 배입니다.

또는 *지연된 정리 롤아웃*을 사용하여 트랜잭션이 커미트된 후 RID 인덱스를 갱신할 수 있습니다. **DB2_MDC_ROLLOUT** 레지스트리 변수를 DEFER로 설정하거나 SET CURRENT MDC ROLLOUT MODE문에 DEFERRED를 지정하여 이 모드를 지정할 수도 있습니다. 지연된 롤아웃에서 RID 인덱스는 삭제가 커미트된 후 백그라운드에서 비동기로 정리됩니다. 이 롤아웃 방법으로 인해 삭제가 매우 큰 경우 또는 많은 RID 인덱스가 테이블에 존재하는 경우 삭제 시간이 상당히 빨라질 수 있습니다. 지연된 인덱스 정리 중에는 인덱스가 병렬식으로 정리되지만 즉시 인덱스 정리에서는 인덱스의 각 행이 하나씩 정리되므로 전체 정리 조작의 속도가 증가합니다. 또한 비동기 인덱스 정리가 인덱스 키 대신 인덱스 페이지의 인덱스 갱신을 로그하므로 DELETE문의 트랜잭션 로그 스페이스 요구사항이 상당히 줄어듭니다.

주: 지연된 정리 롤아웃에는 데이터베이스 힙에서 가져오는 추가 메모리 자원이 필요합니다. 데이터베이스 관리 프로그램이 필요한 메모리 구조를 할당할 수 없는 경우 지연된 정리 롤아웃이 실패하고 관리 통지 로그에 메시지가 기록됩니다.

지연된 정리 롤아웃을 사용하는 시기

삭제 성능이 가장 중요한 요인이며 테이블에 RID 인덱스가 정의된 경우 지연된 정리 롤아웃을 사용하십시오. 인덱스 정리 이전에 롤아웃된 블록의 인덱스 기반 스캔을 수행할 경우 롤아웃된 데이터의 양에 따라 다소 성능 저하가 있습니다. 즉시 인덱스 정리와 지연된 인덱스 정리 중에서 결정할 때 다음의 사항도 고려해야 합니다.

- 삭제 조작의 *크기*

  삭제가 매우 큰 경우 지연된 정리 롤아웃을 선택하십시오. 여러 개의 작은 MDC 테이블에서 차원 DELETE문을 자주 발행하는 경우 비동기로 인덱스 오브젝트를 정리하는 오버헤드가 삭제 조작 중 시간이 절약되는 장점보다 클 수 있습니다.
- 인덱스의 수 및 유형

  테이블에 행 레벨 처리가 필요한 여러 RID 인덱스가 있는 경우 지연된 정리 롤아웃을 사용하십시오.
- 블록 사용 가능성

  삭제 조작으로 해제된 블록 스페이스를 DELETE문이 커미트된 후 즉시 사용할 수 있도록 하려면 즉시 정리 롤아웃을 사용하십시오.
- 로그 스페이스

  로그 스페이스가 제한된 경우 삭제 규모가 크면 지연된 정리 롤아웃을 사용하십시오.
- 메모리 제한조건

  지연된 정리 롤아웃은 지연된 정리가 보류 중인 모든 테이블에서 추가 데이터베이스 힙 스페이스를 사용합니다.

삭제 중 롤아웃 동작을 사용 불가능하게 하려면 **DB2_MDC_ROLLOUT** 레지스트리 변수를 OFF로 설정하거나 SET CURRENT MDC ROLLOUT MODE문에서 NONE을 지정하십시오.

제 21 장 인덱스

파티션된 테이블의 인덱스

파티션된 테이블에서의 인덱스 동작

파티션된 테이블의 인덱스는 파티션되지 않은 테이블의 인덱스와 유사하게 작동하지만 파티션된 인덱스인지 또는 파티션되지 않은 인덱스인지 여부에 따라 다른 스토리지 모델을 사용하여 저장됩니다.

파티션되지 않은 일반 테이블의 인덱스는 모두 공유 인덱스 오브젝트에 상주하는 반면, 파티션된 테이블에서 *파티션되지 않은 인덱스*는 데이터 파티션이 여러 테이블 스페이스에 걸쳐 있다 하더라도 단일 테이블 스페이스에서 자체 인덱스 오브젝트에 작성됩니다. 데이터베이스 관리 스페이스(DMS)와 시스템 관리 스페이스(SMS) 테이블 스페이스 모두 테이블 데이터와 다른 위치에서의 인덱스 사용을 지원합니다. 각 파티션되지 않은 인덱스는 대형 테이블 스페이스를 비롯한 해당 고유 테이블 스페이스에 배치될 수 있습니다. 각 인덱스 테이블 스페이스에서는 데이터 파티션과 동일한 스토리지 메커니즘을 사용해야 합니다(DMS 또는 SMS). 대형 테이블 스페이스의 인덱스는 최대 2^{29}페이지를 포함할 수 있습니다. 모든 테이블 스페이스는 동일한 데이터베이스 파티션 그룹에 있어야 합니다.

*파티션된 인덱스*는 테이블의 파티셔닝 스킴에 따라 인덱스 데이터가 여러 *인덱스* 파티션으로 나뉘어지는 인덱스 구성 스킴을 사용합니다. 각 인덱스 파티션은 해당 데이터 파티션의 테이블 행만 참조합니다. 주어진 데이터 파티션의 모든 인덱스 파티션은 동일한 인덱스 오브젝트에 상주합니다.

XML 데이터에 대한 사용자 작성 인덱스만 파티션되지 않을 수 있습니다. 시스템 생성 XML 영역은 항상 파티션되지만 시스템 생성 컬럼 경로 인덱스는 항상 파티션되지 않습니다.

파티션되지 않은 인덱스의 장점은 다음과 같습니다.

- 각 인덱스에 대해 다른 테이블 스페이스 특성을 정의하는 기능 (예를 들어, 다양한 페이지 *크기*는 더 나은 스페이스 이용률을 보장하는 데 도움이 될 수 있음)
- 인덱스를 서로 독립적으로 재구성할 수 있다는 점
- 인덱스 삭제(drop) 조작 성능 향상
- 인덱스 데이터로의 보다 효율적인 동시 액세스를 제공하는 데 도움이 되는 감소된 입출력 경쟁

- 개별 인덱스가 삭제될 때 인덱스를 재구성할 필요 없이 시스템에서 스페이스가 즉시 사용 가능하게 된다는 점

파티션된 인덱스의 장점은 다음과 같습니다.

- 향상된 데이터 롤인 및 롤아웃 성능
- 인덱스가 파티션되기 때문에 인덱스 페이지 경쟁이 줄어듦
- 다음 결과를 초래할 수 있는 각 인덱스 파티션에 대한 인덱스 B-트리 구조.
  - 인덱스 파티션의 B-트리는 일반적으로 테이블의 모든 데이터를 참조하는 인덱스에 비해 레벨 수가 적기 때문에 삽입, 갱신, 삭제 및 스캔 성능 향상
  - 파티션 제거가 적용될 때 스캔 성능 및 동시성 향상. 파티션된 인덱스 스캔과 파티션되지 않은 인덱스 스캔 둘 다에 파티션 제거를 사용할 수 있지만, 각 인덱스 파티션은 해당 데이터 파티션에 대한 키만 포함하므로 파티션된 인덱스 스캔의 경우 보다 효과적입니다. 따라서 파티션되지 않은 인덱스를 통한 유사 쿼리에 비해 스캔하는 키와 인덱스 페이지 수가 줄어들 수 있습니다.

파티션되지 않은 인덱스는 항상 인덱스 컬럼에 순서를 보존하지만 파티션된 인덱스는 특정 시나리오(예: 파티션된 컬럼이 인덱스 컬럼과 일치하지 않고 두 개 이상의 파티션에 액세스할 경우)에서는 파티션에서 일부 순서를 유실할 수 있습니다.

온라인 인덱스 작성 중에 테이블에 대한 동시 읽기 및 쓰기 액세스가 허용됩니다. 이러한 인덱스가 빌드된 후, 인덱스 작성 중에 이루어진 테이블 변경사항은 새 인덱스에 적용됩니다. 인덱스 작성이 완료되고 트랜잭션이 커미트될 때까지 테이블에 대한 쓰기 액세스가 차단됩니다. 파티션된 인덱스의 경우, 인덱스 파티션 작성 중에 이루어진 데이터 파티션 변경사항이 적용되는 동안만 각 데이터 파티션은 읽기 전용 액세스로 Quiesce 상태가 됩니다.

파티션된 인덱스는 ALTER TABLE...ATTACH PARTITION문을 사용하여 데이터를 롤링하는 경우에 특히 유용합니다. 파티션되지 않은 인덱스가 있는 경우(테이블에 XML 데이터가 있는 경우 XML 컬럼 경로 인덱스 제외), 파티션 첨부 후에 SET INTEGRITY 문을 실행하십시오. 이는 파티션되지 않은 인덱스 유지보수, 범위 유효성 확인, 제한조건 검사 및 구체화된 쿼리 테이블(MQT) 유지보수에 필요합니다. 파티션되지 않은 인덱스 유지보수는 시간이 걸릴 수 있으며 대량의 로그 스페이스 공간을 필요로 합니다. 이러한 유지보수 비용이 들지 않게 하려면 파티션된 인덱스를 사용하십시오.

345 페이지의 그림 50에서는 파티션된 테이블의 파티션되지 않은 두 개의 인덱스(각각의 인덱스는 별도의 상주함)를 표시합니다.

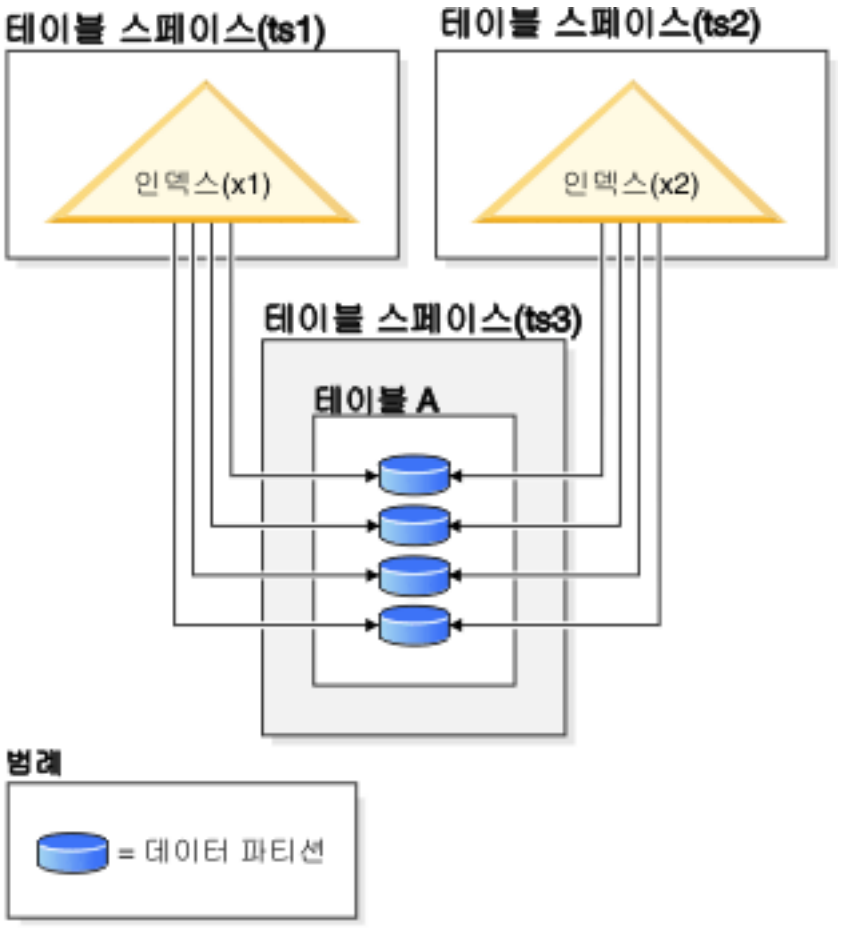


그림 50. 파티션된 테이블의 파티션되지 않은 인덱스

346 페이지의 그림 51에서는 두 개의 데이터베이스 파티션에 걸쳐 있고 단일 테이블 스페이스에 상주하는 파티션된 테이블의 파티션되지 않은 인덱스를 표시합니다.

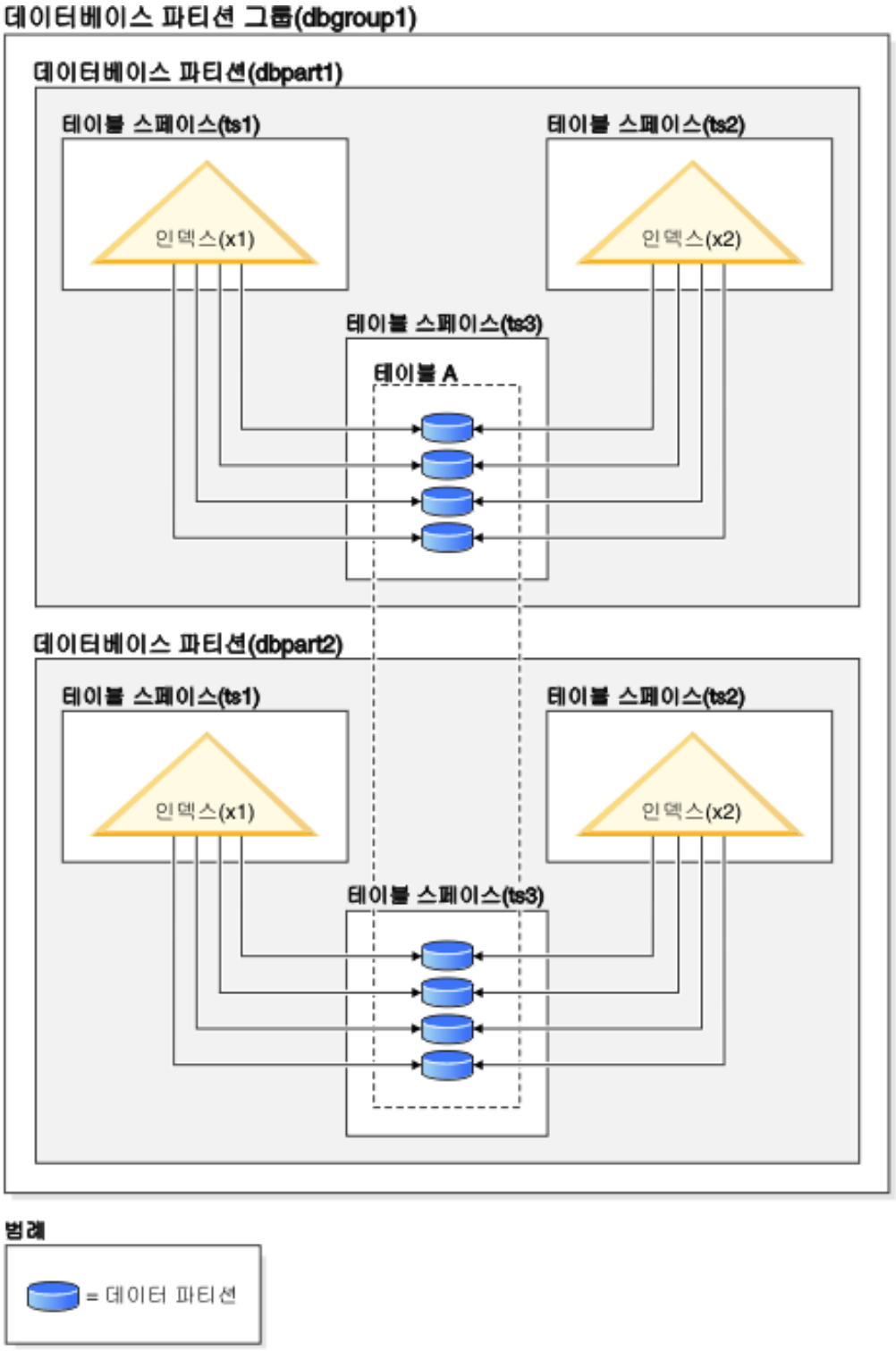


그림 51. 분산되고 파티션된 테이블에서 파티션되지 않은 인덱스

347 페이지의 그림 52에서는 파티션된 테이블의 파티션된 인덱스와 파티션되지 않은 인덱스 혼합을 표시합니다.

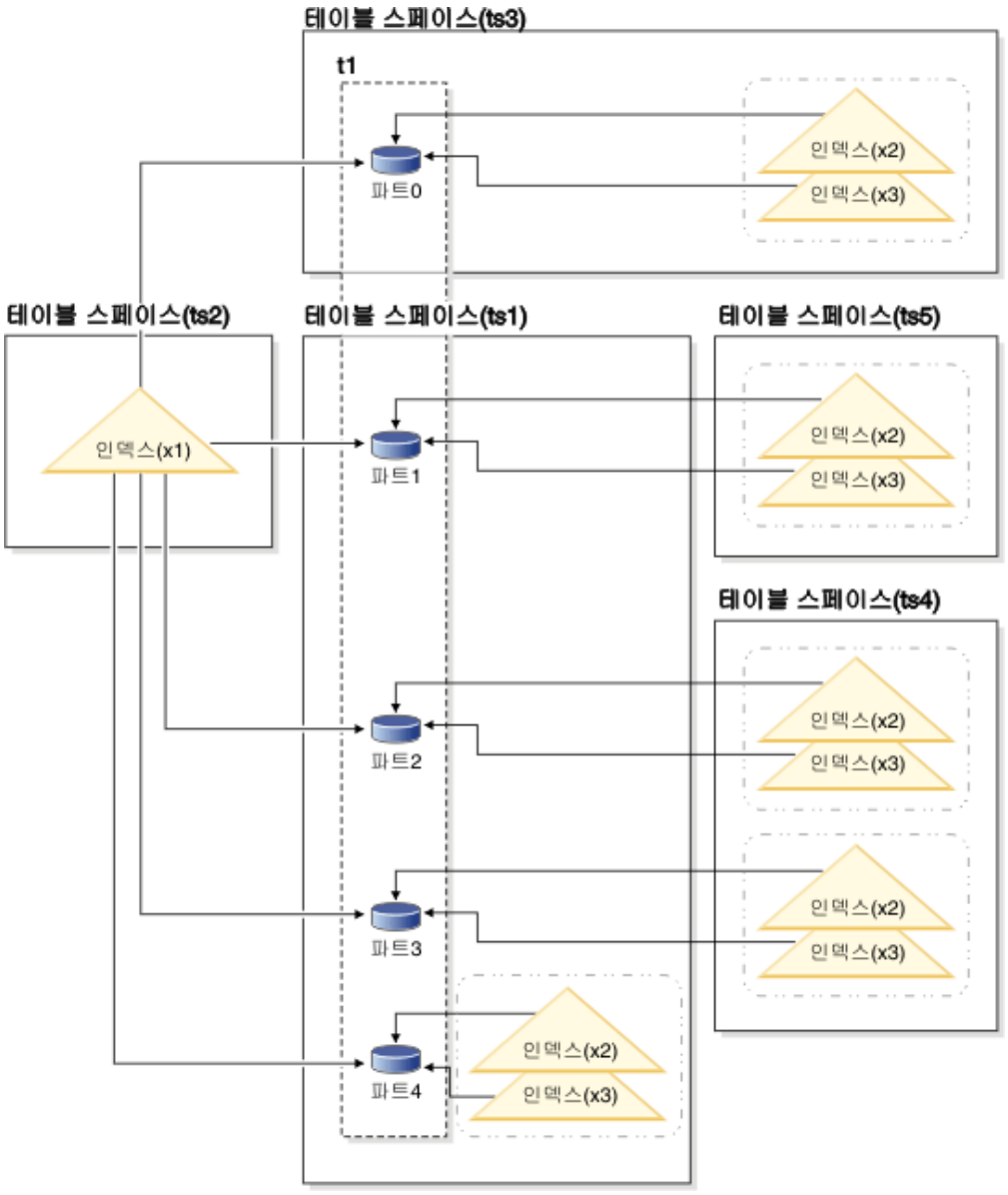


그림 52. 파티션된 테이블의 파티션된 인덱스 및 파티션되지 않은 인덱스

파티션되지 않은 인덱스 X1은 모든 데이터 파티션의 행을 참조합니다. 대조적으로, 파티션된 인덱스 X2 및 X3은 연관된 데이터 파티션의 행만 참조합니다. 테이블 스페이스 TS3은 연관된 데이터 파티션의 테이블 스페이스를 공유하는 인덱스 파티션도 표시합니다. 이는 파티션된 인덱스의 경우 디폴트입니다.

파티션되지 않은 인덱스와 파티션된 인덱스의 디폴트 위치를 겹쳐쓸 수 있습니다(겹쳐쓰는 방법은 각각 다를 수 있음). 파티션되지 않은 인덱스를 사용하여 인덱스 작성 시 테이블 스페이스를 지정할 수 있습니다. 파티션된 인덱스의 경우, 테이블을 작성할 때 저장될 테이블 스페이스 인덱스 파티션을 결정해야 합니다.

파티션되지 않은 인덱스

파티션되지 않은 인덱스의 인덱스 위치를 겹쳐쓰려면, 인덱스의 대체 테이블 스페이스 위치를 지정할 수 있게 하는 CREATE INDEX문에서 IN절을 사용하십시오. 필요하면 테이블 스페이스마다 다른 인덱스를 배치할 수 있습니다. 파티션되지 않은 인덱스를 배치할 위치를 지정하지 않고 파티션된 테이블을 작성한 후 테이블 스페이스를 지정하지 않는 CREATE INDEX문을 사용하여 인덱스를 작성할 경우, 인덱스가 첫 번째 첨부, 또는 표시된 데이터 파티션의 테이블 스페이스에서 작성됩니다. 다음 세 가지 가능한 경우 각각이 경우 1부터 순서대로 평가되어 인덱스가 작성될 위치를 판별합니다. 인덱스에 대한 테이블 스페이스 배치를 판별하기 위한 이 평가는 일치하는 경우가 발견되면 중지됩니다.

경우 1:
인덱스 테이블 스페이스가 CREATE INDEX...IN
*tbspace*문에서 지정된 경우 이 인덱스에 대해 지정된 테이블 스페이스를
사용하십시오.

경우 2:
인덱스 테이블 스페이스가 CREATE TABLE...
INDEX IN *tbspace*문에 지정된 경우 이 인덱스에 지정된
테이블 스페이스를 사용하십시오.

경우 3:
테이블 스페이스가 지정되지 않은 경우 첫 번째 첨부, 또는
표시된 데이터 파티션에서 사용되는 테이블 스페이스를 선택하십시오.

파티션된 인덱스

디폴트로, 인덱스 파티션은 참조하는 데이터 파티션과 동일한 테이블 스페이스에 배치됩니다. 이 디폴트 동작을 겹쳐쓰려면, CREATE TABLE문을 사용하여 정의하는 각 데이터 파티션에 INDEX IN절을 사용해야 합니다. 즉, 파티션된 테이블에 파티션된 인덱스를 사용하려면 테이블을 작성할 때 인덱스 파티션을 저장할 위치를 예상해야 합니다. 파티션된 인덱스 작성 시 INDEX IN절을 사용하려고 하면 오류 메시지를 수신합니다.

예 1: 파티션된 테이블 SALES(a int, b int, c int)의 경우 고유 인덱스 A_IDX를 작성하십시오.

```
create unique index a_idx on sales (a)
```

테이블 SALES는 파티션되어 있기 때문에 인덱스 a_idx도 파티션된 인덱스로 작성됩니다.

예 2: 인덱스 B_IDX를 작성하십시오.

```
create index b_idx on sales (b)
```

예 3: 파티션된 인덱스에서 인덱스 파티션의 디폴트 위치를 겹쳐쓰려면 파티션된 테이블을 작성할 때 정의하는 각 파티션에 INDEX IN절을 사용하십시오. 다음 예에서 테이블 Z의 인덱스는 테이블 스페이스 TS3에 작성됩니다.

```
create table z (a int, b int)
      partition by range (a) (starting from (1)
      ending at (100)  index in ts3)

   create index c_idx on z (a) partitioned
```

파티션된 테이블에서 파티션되지 않은 인덱스 클러스터링

클러스터링 인덱스는 일반 테이블에 대해 제공하는 이점과 동일한 이점을 파티션된 테이블에 제공합니다. 그러나 클러스터링 인덱스를 선택할 때 테이블 파티션 키 정의에서 주의를 기울여야 합니다.

클러스터링 키를 사용하여 파티션된 테이블의 클러스터링 인덱스를 작성할 수 있습니다. 데이터베이스 서버는 클러스터링 인덱스를 사용하여 각 데이터 파티션 내에서 로컬로 데이터를 클러스터하려고 시도합니다. 클러스터된 삽입 조작 동안 적절한 레코드 ID(RID)를 찾기 위해 인덱스 찾아보기가 수행됩니다. 이 RID는 레코드를 삽입할 스페이스를 찾을 때 테이블에서 시작점으로 사용됩니다. 좋은 성능의 최적의 클러스터링을 얻기 위해서는 인덱스 컬럼과 테이블 파티션 키 컬럼 사이에 상관이 있어야 합니다. 이러한 상관을 보장하는 한 가지 방법은 다음 예에 표시된 것처럼 인덱스 컬럼에 테이블 파티션 키 컬럼으로 접두부를 지정하는 것입니다.

```
partition by range (month, region)
create index...(month, region, department) cluster
```

데이터베이스 서버에서 이 상관을 강제 적용하지는 않지만, 좋은 클러스터링을 얻기 위해 인덱스의 모든 키가 파티션 ID별로 함께 그룹화될 가능성이 있습니다. 예를 들어, 테이블이 분기로 파티션되어 있고 클러스터링 인덱스가 날짜로 정의되어 있다고 가정해 보십시오. 분기와 날짜 간에 관계가 있고, 데이터 파티션의 모든 키가 인덱스 내에서 함께 그룹화되므로 좋은 성능의 최적의 데이터 클러스터링을 얻을 수 있습니다. 350 페이지의 그림 53에서는 클러스터링이 테이블 파티션 키와 상관될 경우에만 최적의 스캔 성능이 확보된다는 것을 보여줍니다.

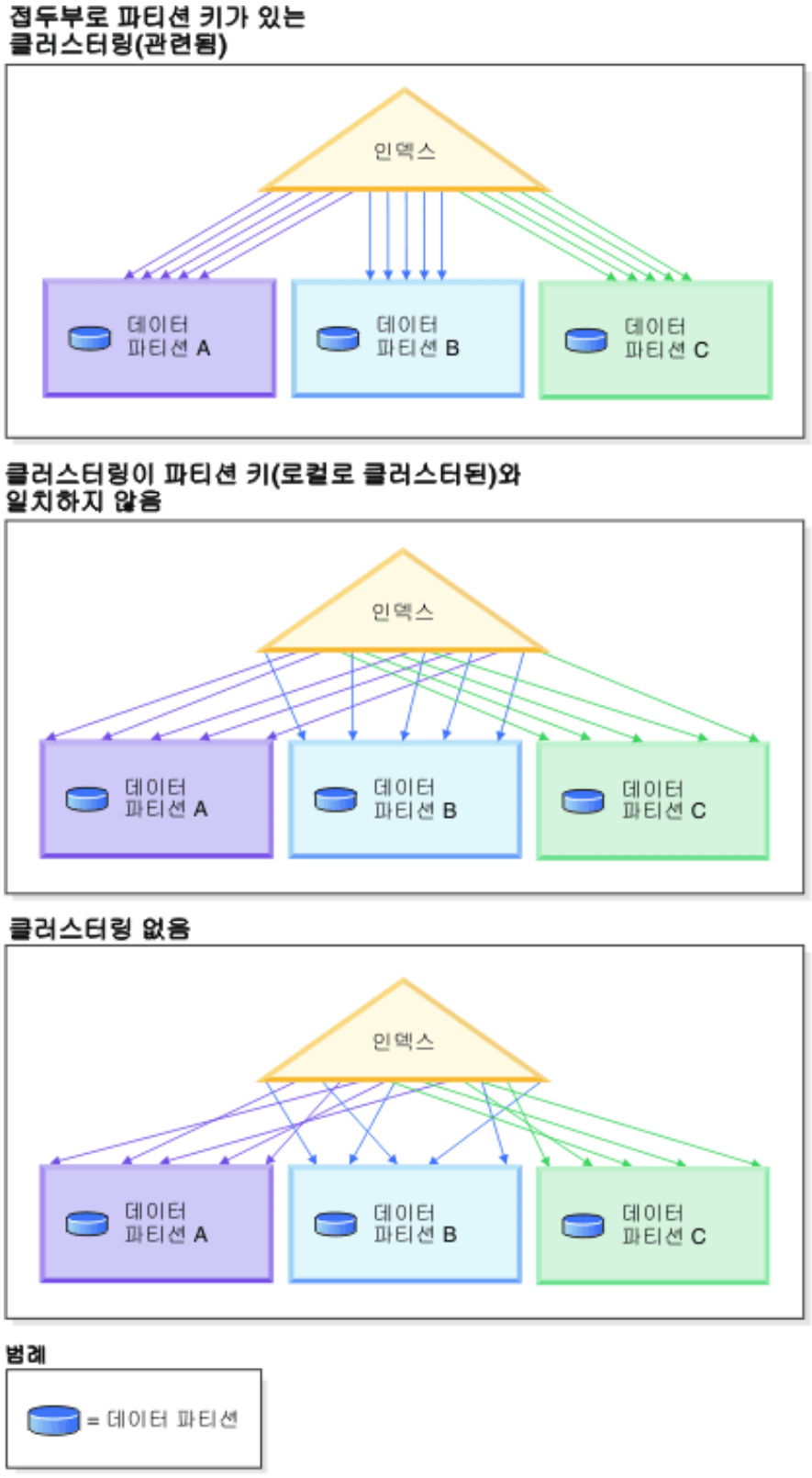


그림 53. 파티션된 테이블에서 클러스터된 인덱스의 가능한 효과

클러스터링의 이점은 다음과 같습니다.

- 각 데이터 파티션 내에서 행이 키 순서로 됩니다.
- 스캐너가 첫 번째 페이지의 첫 번째 행을 페치한 후 같은 페이지의 각 행을 페치한 후 다음 페이지로 넘어가므로 클러스터링 인덱스는 키 순서로 테이블을 횡단하는 스캔의 성능을 향상시킵니다. 즉, 지정된 시간에 테이블의 하나의 페이지만 버퍼 풀에 있어야 합니다. 반대로, 테이블이 클러스터되지 않은 경우, 다른 페이지에서 행이 페치될 수 있습니다. 버퍼 풀에서 전체 테이블을 수용할 수 없는 경우, 대부분의 페이지가 두 번 이상 페치될 수 있어 스캔 속도를 크게 저하시킵니다.

클러스터링 키가 테이블 파티션 키와 상관되지 않지만 데이터가 로컬로 클러스터되는 경우, 버퍼 풀에 각 데이터 파티션의 한 페이지를 수용할 충분한 스페이스가 있으면 클러스터된 인덱스의 전체 이점을 얻을 수 있습니다. 이는 특정 데이터 파티션에서 페치된 각 행이 같은 파티션에서 이전에 페치된 행 근처에 있기 때문입니다(350 페이지의 그림 53의 두 번째 예 참조).

제 22 장 디자인 어드바이저

디자인 어드바이저를 사용하여 단일 파티션 데이터베이스에서 다중 파티션 데이터베이스로 변환

디자인 어드바이저를 사용하면 단일 파티션 데이터베이스에서 다중 파티션 데이터베이스로 변환하는 데 도움이 될 수 있습니다.

이 태스크에 대한 정보

새 인덱스, 구체화된 쿼리 테이블(MQT) 및 다차원적으로 클러스터된(MDC) 테이블에 대한 권장사항을 제공하는 것 외에도 디자인 어드바이저는 데이터 분배에 대한 권장사항을 제공할 수 있습니다.

프로시저

1. db2licm 명령을 사용하여 데이터베이스 파티셔닝 기능(DPF) 라이센스 키를 등록하십시오.
2. 다중 파티션 데이터베이스 파티션 그룹에서 적어도 하나의 테이블 스페이스를 작성하십시오.

   주: 디자인 어드바이저는 기존 테이블 스페이스로의 데이터 재분배만 권장할 수 있습니다.
3. db2advis 명령에서 파티셔닝 옵션을 지정하여 디자인 어드바이저를 실행하십시오.
4. 디자인 어드바이저에 의해 생성된 DDL문을 실행하기 전에 db2advis 출력 파일을 약간 수정하십시오. 디자인 어드바이저가 생성하는 DDL 스크립트를 실행할 수 있으려면 먼저 데이터베이스 파티셔닝이 설정되어야 하므로, 리턴되는 스크립트에서 권장사항이 주석으로 제공됩니다. 권장사항에 따라 테이블을 변환할지 여부는 사용자가 결정해야 합니다.

제 23 장 동시성 관리

MDC 테이블 및 RID 인덱스 스캔에 대한 잠금 모드

테이블 또는 RID 인덱스 스캔 동안 다차원적으로 클러스터된(MDC) 테이블이 갖는 잠금의 유형은 적용되는 분리 수준 및 사용되는 데이터 액세스 플랜에 따라 달라집니다.

다음 테이블에서는 여러 액세스 플랜에 대한 각 분리 수준에서 MDC 테이블에 대해 얻어지는 잠금의 유형을 보여줍니다. 각 항목은 테이블 잠금, 블록 잠금 및 행 잠금의 세 부분으로 되어 있습니다. 하이픈은 특정 잠금 단위가 사용 가능하지 않음을 표시합니다.

테이블 9-14는 데이터 페이지 읽기가 지연될 경우 RID 인덱스 스캔에 대해 얻어지는 잠금의 유형을 표시합니다. UR 분리 수준에서, 인덱스의 Include 컬럼에 술어가 있는 경우, 분리 수준이 CS로 업그레이드되고 잠금이 IS 테이블 잠금, IS 블록 잠금 또는 NS 행 잠금으로 업그레이드됩니다.

- 표 1. 술어가 없는 테이블 스캔에 대한 잠금 모드
- 표 2. 차원 컬럼에만 술어가 있는 테이블 스캔에 대한 잠금 모드
- 표 3. 기타 술어(sargs, resids)가 있는 테이블 스캔에 대한 잠금 모드
- 표 4. 술어가 없는 RID 인덱스 스캔에 대한 잠금 모드
- 표 5. 단일 규정 행이 있는 RID 인덱스 스캔에 대한 잠금 모드
- 표 6. 시작 및 중지 술어만 있는 RID 인덱스 스캔에 대한 잠금 모드
- 표 7. 인덱스 술어만 있는 RID 인덱스 스캔에 대한 잠금 모드
- 표 8. 기타 술어(sargs, resids)가 있는 RID 인덱스 스캔에 대한 잠금 모드
- 표 9. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 RID 인덱스 스캔
- 표 10. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 RID 인덱스 스캔 후
- 표 11. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어(sargs, resids)가 있는 RID 인덱스 스캔
- 표 12. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어(sargs, resids)가 있는 RID 인덱스 스캔 후
- 표 13. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 RID 인덱스 스캔
- 표 14. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 RID 인덱스 스캔 후

주: 잠금 모드는 SELECT문의 *lock-request-clause*를 사용하여 명시적으로 변경할 수 있습니다.

표 16. 술어가 없는 테이블 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

표 17. 차원 컬럼에만 술어가 있는 테이블 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-

표 18. 기타 술어(sargs, resids)가 있는 테이블 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

표 19. 술어가 없는 RID 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

표 20. 단일 규정 행이 있는 RID 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

표 21. 시작 및 중지 술어만 있는 RID 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

표 22. 인덱스 술어만 있는 RID 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

표 23. 기타 술어(sargs, resids)가 있는 RID 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

표 24. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 RID 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	

표 24. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 RID 인덱스 스캔 (계속)

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

표 25. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 RID 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

표 26. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어(sargs, resids)가 있는 RID 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

표 27. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어(sargs, resids)가 있는 RID 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

표 28. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 RID 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/IS/S	IX/IX/S		IX/IX/X	

표 28. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 RID 인덱스 스캔 (계속)

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

표 29. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 RID 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

MDC 블록 인덱스 스캔에 대한 잠금 모드

블록 인덱스 스캔 동안 다차원적으로 클러스터된(MDC) 테이블이 갖는 잠금의 유형은 적용되는 분리 수준 및 사용되는 데이터 액세스 플랜에 따라 달라집니다.

다음 테이블에서는 여러 액세스 플랜에 대한 각 분리 수준에서 MDC 테이블에 대해 얻어지는 잠금의 유형을 보여줍니다. 각 항목은 테이블 잠금, 블록 잠금 및 행 잠금의 세 부분으로 되어 있습니다. 하이픈은 특정 잠금 단위가 사용 가능하지 않음을 표시합니다.

테이블 5-12는 데이터 페이지 읽기가 지연될 경우 블록 인덱스 스캔에 대해 얻어지는 잠금의 유형을 표시합니다.

- 표 1. 술어가 없는 인덱스 스캔에 대한 잠금 모드
- 표 2. 차원 컬럼에만 술어가 있는 인덱스 스캔에 대한 잠금 모드
- 표 3. 시작 및 중지 술어만 있는 인덱스 스캔에 대한 잠금 모드
- 표 4. 술어가 있는 인덱스 스캔에 대한 잠금 모드
- 표 5. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 블록 인덱스 스캔
- 표 6. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 블록 인덱스 스캔 후
- 표 7. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 차원 컬럼에만 술어가 있는 블록 인덱스 스캔

- 표 8. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 차원 컬럼에만 술어가 있는 블록 인덱스 스캔 후
- 표 9. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 블록 인덱스 스캔
- 표 10. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 블록 인덱스 스캔 후
- 표 11. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 기타 술어(sargs, resids)가 있는 블록 인덱스 스캔
- 표 12. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 기타 술어(sargs, resids)가 있는 블록 인덱스 스캔 후

주: 잠금 모드는 SELECT문의 *lock-request-clause*를 사용하여 명시적으로 변경할 수 있습니다.

표 30. 술어가 없는 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--

표 31. 차원 컬럼에만 술어가 있는 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

표 32. 시작 및 중지 술어만 있는 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-

표 33. 술어가 있는 인덱스 스캔에 대한 잠금 모드

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

표 34. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 블록 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

표 35. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 술어가 없는 블록 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--

표 36. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 차원 컬럼에만 술어가 있는 블록 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

표 37. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 차원 컬럼에만 술어가 있는 블록 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--

표 38. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 블록 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

표 39. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 시작 및 중지 술어만 있는 블록 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	

표 40. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 기타 술어(sargs, resids)가 있는 블록 인덱스 스캔

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

표 41. 지연된 데이터 페이지 액세스에 사용된 인덱스 스캔에 대한 잠금 모드: 기타 술어(sargs, resids)가 있는 블록 인덱스 스캔 후

분리 수준	읽기 전용 및 앰비규어스 스캔	커서 조작		검색된 갱신 또는 삭제	
		스캔	Where current of	스캔	갱신 또는 삭제
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

파티션된 테이블에서의 잠금 동작

전체 테이블 잠금 외에도, 파티션된 테이블의 각 데이터 파티션에 대한 잠금이 있습니다.

이는 파티션되지 않은 테이블에 비해 더 미세한 세분화도와 증가된 동시성을 제공합니다. 데이터 파티션 잠금은 테이블 함수, 관리 뷰, 이벤트 모니터 및 db2pd 명령의 출력에서 식별됩니다.

테이블에 액세스할 때 테이블 잠금이 우선 획득된 후 필요에 따라 파티션 잠금이 획득됩니다. 액세스 메소드 및 분리 수준에서 결과 세트에 표시되지 않은 데이터 파티션의 잠금을 요구할 수도 있습니다. 이러한 데이터 파티션 잠금은 획득된 후 테이블 잠금만큼 오래 보유될 수도 있습니다. 예를 들어, 인덱스에 대한 커서 안정성(CS) 스캔에서 이전에 액세스한 데이터 파티션에 대한 잠금을 유지하여 나중에 데이터 파티션 잠금을 다시 획득하는 비용을 줄일 수 있습니다.

데이터 파티션 잠금은 또한 테이블 스페이스에 대한 액세스를 보장하는 비용도 감당합니다. 파티션되지 않은 테이블의 경우, 테이블 스페이스 액세스는 테이블 잠금에 의해 처리됩니다. 테이블 레벨에서 독점 또는 공유 잠금이 있는 경우라도 데이터 파티션 잠금이 발생합니다.

보다 미세한 세분화도를 통해 한 트랜잭션에서 특정 데이터 파티션에 대한 독점 액세스를 가질 수 있고 다른 트랜잭션에서 다른 데이터 파티션에 액세스하는 동안 행 잠금을 피할 수 있습니다. 이는 모두 갱신에 대해 선택한 플랜의 결과이거나 데이터 파티션 레벨에 대한 잠금의 에스컬레이션 때문일 수 있습니다. 많은 액세스 메소드에서 테이블 잠금은 데이터 파티션이 공유 또는 독점 모드로 잠긴다 하더라도, 일반적으로 의도 잠금입니다. 이는 동시성을 증가시켜 줍니다. 그러나 데이터 파티션 레벨에서 비의도 잠금이 필요하고 플랜에서 모든 데이터 파티션에 액세스할 수 있다고 표시할 경우, 동시 트랜잭션 간 데이터 파티션 교착 상태를 방지하기 위해 테이블 레벨에서 비의도 잠금이 선택될 수도 있습니다.

LOCK TABLE문

파티션 테이블의 경우, LOCK TABLE문에 의해 획득되는 유일한 잠금은 테이블 레벨 잠금입니다. 이것은 후속 데이터 처리 언어(DML) 명령문에 의한 행 잠금을 예방하고, 행, 블록 또는 데이터 파티션 레벨에서의 교착 상태를 방지합니다. IN EXCLUSIVE MODE 옵션을 사용하여 인덱스 갱신 시 독점 액세스를 보장할 수 있으며, 이는 대형 갱신 동안 인덱스의 증가를 제한하는 데 유용합니다.

ALTER TABLE문에서 LOCKSIZE TABLE 옵션의 효과

LOCKSIZE TABLE 옵션은 의도 잠금 없는 독점 또는 공유 모드에서 테이블이 잠기도록 보장합니다. 파티션된 테이블의 경우, 이 잠금 전략은 테이블 잠금과 데이터 파티션 잠금에 모두 적용됩니다.

행 및 블록 레벨 잠금 에스컬레이션

파티션된 테이블에서 행 및 블록 레벨 잠금은 데이터 파티션 레벨로 에스컬레이션될 수 있습니다. 이렇게 될 경우, 데이터 파티션이 공유, 독점 또는 강한 독점 모드로 에스컬레이션된다 하더라도 다른 데이터 파티션이 영향을 받지 않으므로, 다른 트랜잭션에서 테이블에 더 잘 액세스할 수 있습니다. 에스컬레이션에 대한 통지 로그 항목에는 테이블의 이름 및 영향을 받는 데이터 파티션이 포함됩니다.

파티션되지 않은 인덱스에 대한 독점 액세스는 잠금 에스컬레이션에 의해 보장될 수 없습니다. 독점 액세스의 경우, 다음 조건 중 하나가 참이어야 합니다.

- 명령문이 독점 테이블 레벨 잠금을 사용해야 함
- 명시적 LOCK TABLE IN EXCLUSIVE MODE문을 실행함
- 테이블에 LOCKSIZE TABLE 속성이 있어야 함

파티션된 인덱스의 경우, 인덱스 파티션에 대한 독점 액세스는 독점 또는 강한 독점 액세스 모드로의 데이터 파티션 잠금 에스컬레이션으로 보장됩니다.

잠금 정보 해석

SNAPLOCK 관리 뷰가 파티션된 테이블에 대해 리턴되는 잠금 정보를 해석하는 데 도움이 될 수 있습니다. 오프라인 인덱스 재구성 동안 다음과 같은 SNAPLOCK 관리 뷰가 캡처되었습니다.

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE,
LOCK_MODE, LOCK_ESCALATION FROM SYSIBMADM.SNAPLOCK where TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%'
ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE

TABNAME         TAB_FILE_ID          TBSP_NAME       DATA_PARTITION_ID LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_ESCALATION
--------------- -------------------- --------------- ----------------- ------------------ ---------- ---------------
TP1                           32768 -                              -1 TABLE_LOCK         Z                       0
TP1                               4 USERSPACE1                     0 TABLE_PART_LOCK    Z                       0
TP1                               5 USERSPACE1                     1 TABLE_PART_LOCK    Z                       0
TP1                               6 USERSPACE1                     2 TABLE_PART_LOCK    Z                       0
TP1                               7 USERSPACE1                     3 TABLE_PART_LOCK    Z                       0
```

```
TP1                             8 USERSPACE1                    4 TABLE_PART_LOCK    Z                    0
TP1                             9 USERSPACE1                    5 TABLE_PART_LOCK    Z                    0
TP1                            10 USERSPACE1                    6 TABLE_PART_LOCK    Z                    0
TP1                            11 USERSPACE1                    7 TABLE_PART_LOCK    Z                    0
TP1                            12 USERSPACE1                    8 TABLE_PART_LOCK    Z                    0
TP1                            13 USERSPACE1                    9 TABLE_PART_LOCK    Z                    0
TP1                            14 USERSPACE1                   10 TABLE_PART_LOCK    Z                    0
TP1                            15 USERSPACE1                   11 TABLE_PART_LOCK    Z                    0
TP1                             4 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                             5 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                             6 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                             7 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                             8 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                             9 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            10 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            11 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            12 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            13 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            14 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            15 USERSPACE1                    - TABLE_LOCK         Z                    0
TP1                            16 USERSPACE1                    - TABLE_LOCK         Z                    0

  26 record(s) selected.
```

이 예에서는 파티션된 테이블 TP1에 대한 액세스 및 동시성을 제어하는 데 DATA_PARTITION_ID -1 및 TABLE_LOCK 유형의 잠금 오브젝트가 사용됩니다. 각 데이터 파티션에 대한 대부분의 액세스 및 동시성을 제어하는 데에는 TABLE_PART_LOCK 유형의 잠금 오브젝트가 사용됩니다.

이 출력에는 DATA_PARTITION_ID에 대한 값이 없는 TABLE_LOCK 유형의 추가 잠금 오브젝트가 캡처되어 있습니다 (TAB_FILE_ID 4-16). 이 유형의 잠금(오브젝트의 TAB_FILE_ID 및 TBSP_NAME이 파티션된 테이블의 데이터 파티션 또는 인덱스에 해당하는)은 온라인 백업 유틸리티에서 동시성을 제어하는 데 사용될 수 있습니다.

제 24 장 에이전트 관리

파티션된 데이터베이스의 에이전트

파티션된 데이터베이스 환경 또는 파티션 내 병렬 처리가 사용 가능한 환경에서 각 데이터베이스 파티션에는 서브에이전트를 작성할 수 있는 자체의 에이전트 풀이 있습니다.

이 풀로 인해 필요할 때 또는 작업을 완료했을 때마다 서브에이전트를 작성 및 파괴할 필요가 없습니다. 서브에이전트는 풀에서 연관 에이전트로 계속 남아 있을 수 있으며 데이터베이스 관리 프로그램은 서브에이전트가 연결되는 응용프로그램 또는 새 응용프로그램의 새 요청에 사용할 수 있습니다.

시스템의 성능 및 메모리 사용에 대한 영향은 에이전트 풀이 조정된 방식과 깊은 관련이 있습니다. 에이전트 풀 *크기*의 데이터베이스 관리 프로그램 구성 매개변수(**num_poolagents**)는 데이터베이스 파티션의 응용프로그램과 계속 연관시킬 수 있는 총 에이전트 및 서브에이전트 수에 영향을 줍니다. 풀 *크기*가 너무 작고 풀이 가득 찬 경우 서브에이전트는 작업 중인 응용프로그램과의 연관을 취소하고 종료됩니다. 서브에이전트를 계속 작성하고 응용프로그램과 다시 연관시켜야 하므로 성능이 저하됩니다.

디폴트로 100 값을 사용하여 **num_poolagents**는 AUTOMATIC으로 설정되며 데이터베이스 관리 프로그램은 풀링할 유휴 에이전트 수를 자동으로 관리합니다.

num_poolagents의 값이 수동으로 너무 낮게 설정된 경우 한 응용프로그램이 풀을 연관된 서브에이전트로 가득 채울 수 있습니다. 그런 다음 다른 응용프로그램이 새 서브에이전트를 요구하지만 에이전트 풀에 서브에이전트가 없는 경우 다른 응용프로그램의 에이전트 풀에서 비활성 서브에이전트를 재활용합니다. 이러한 동작으로 자원을 완전히 활용할 수 있습니다.

num_poolagents의 값이 수동으로 너무 높게 설정된 경우 다른 태스크에 사용할 수 없는 데이터베이스 관리 프로그램 자원을 사용하여 오랜 기간 동안 풀에서 계속 사용되지 않습니다.

연결 집중기가 사용 가능한 경우 **num_poolagents**의 값은 한 번에 풀에서 유휴 상태일 수 있는 정확한 에이전트 수를 반드시 반영하지는 않습니다. 높은 워크로드 활동을 처리하기 위해 임시로 에이전트가 필요할 수도 있습니다.

데이터베이스 에이전트 외에 다음과 같은 기타 비동기 데이터베이스 관리 프로그램 활동이 자체 프로세스 또는 스레드로 실행됩니다.

- 데이터베이스 입출력 서버 또는 입출력 프리페처
- 데이터베이스 비동기 페이지 클리너

- 데이터베이스 로그 프로그램
- 데이터베이스 교착 상태 검출기
- 통신 및 IPC 리스너
- 테이블 스페이스 컨테이너 재조정 프로그램

제 25 장 액세스 플랜 최적화

인덱스 액세스 및 클러스터 비율

옵티마이저는 액세스 플랜을 선택할 때 디스크에서 버퍼 풀로 페이지를 페치하는 데 필요한 입출력 수를 추정합니다. 이미 버퍼 풀에 있는 페이지에서 행을 읽는 데 추가적인 입출력이 필요하지 않으므로 이 추정 값에는 버퍼 풀 사용량 예측이 포함됩니다.

인덱스 스캔의 경우 시스템 카탈로그의 정보는 옵티마이저가 버퍼 풀로 데이터 페이지를 읽는 입출력 비용을 추정하는 데 도움이 됩니다. SYSCAT.INDEXES 뷰에서 다음 컬럼의 정보를 사용합니다.

- CLUSTERRATIO 정보는 이 인덱스와 관련하여 테이블 데이터가 클러스터된 수준을 표시합니다. 이 수가 높으면 인덱스 키 시퀀스로 행이 잘 정렬되어 있습니다. 테이블 행이 인덱스 키 시퀀스에 가까운 경우 페이지가 버퍼에 있는 동안 데이터 페이지에서 행을 읽을 수 있습니다. 이 컬럼의 값이 -1인 경우 옵티마이저는 PAGE_FETCH_PAIRS 및 CLUSTERFACTOR 정보(사용 가능한 경우)를 사용합니다.
- PAGE_FETCH_PAIRS 컬럼에는 CLUSTERFACTOR 정보와 함께 다양한 크기의 버퍼 풀로 데이터 페이지를 읽는 데 필요한 입출력 수를 모델링한 숫자 쌍이 있습니다. DETAILED절을 지정하여 인덱스에 대해 RUNSTATS 명령을 호출하는 경우에만 이러한 컬럼의 데이터를 수집합니다.

인덱스 클러스터링 통계가 사용 불가능한 경우 옵티마이저는 인덱스와 관련하여 데이터의 클러스터링 수준이 낮음을 가정하는 디폴트값을 사용합니다. 데이터의 클러스터링 수준은 성능에 상당한 영향을 미칠 수 있으며 테이블에 정의된 인덱스 중 하나를 100%에 가까운 클러스터링 수준으로 유지해야 합니다. 일반적으로 인덱스의 키가 클러스터링 인덱스 키의 수퍼 세트를 표시하거나 두 인덱스의 키 컬럼 사이에 실제 상관이 있는 경우를 제외하고는 하나의 인덱스만 100% 클러스터될 수 있습니다.

테이블을 재구성할 때 행을 클러스터링하고 삽입 처리 중 계속 클러스터링을 유지하는 데 사용되는 인덱스를 지정할 수 있습니다. 갱신 및 삽입 조작은 테이블을 인덱스와 관련하여 거의 클러스터링하지 않으므로 테이블을 주기적으로 재구성해야 합니다. 삽입, 갱신 또는 삭제 조작을 자주 수행하는 테이블의 재구성 수를 줄이려면 ALTER TABLE 문에 PCTFREE절을 지정하십시오.

MDC 테이블에 대한 테이블 및 인덱스

다차원적으로 클러스터된(MDC) 테이블에 대한 테이블 및 인덱스 구성은 표준 테이블 구성과 동일한 논리 구조에 기초합니다.

표준 테이블과 마찬가지로 MDC 테이블은 컬럼으로 구분된 데이터 행이 들어 있는 페이지로 구성됩니다. 각 페이지의 행은 레코드 ID(RID)로 식별합니다. 그러나 MDC 테이블의 페이지는 Extent 크기 블록으로 그룹화됩니다. 예를 들어, 그림 54에서는 Extent 크기가 4인 테이블을 표시합니다. 0 - 3의 처음 네 페이지는 테이블을 첫 번째 블록을 나타냅니다. 4 - 7의 다음 네 페이지는 테이블의 두 번째 블록을 나타냅니다.

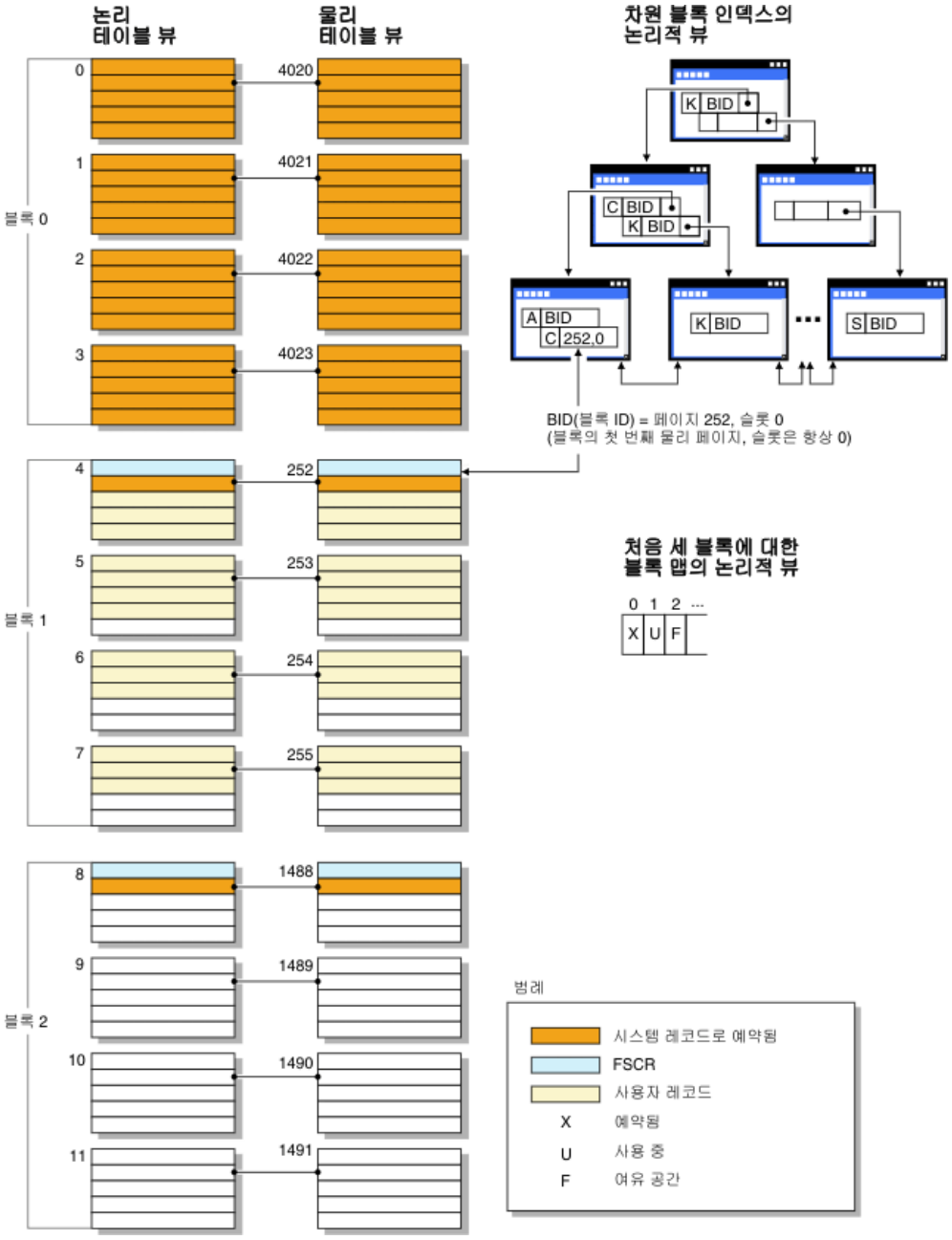


그림 54. MDC 테이블의 논리 테이블, 레코드 및 인덱스 구조

첫 번째 블록에는 DB2 서버가 테이블을 관리하는 데 사용하는 FSCR(Free Space Control Record)을 포함한 특수 내부 레코드가 들어 있습니다. 연속 블록에서 첫 번째

페이지에 FSCR이 있습니다. FSCR은 블록의 각 페이지에 있는 새 레코드의 여유 공간을 맵핑합니다. 이 사용 가능한 여유 공간은 테이블에 레코드를 삽입할 때 사용됩니다.

이름이 의미하는 바와 같이 MDC 테이블은 여러 차원에서 데이터를 클러스터합니다. 각 차원은 CREATE TABLE문의 ORGANIZE BY DIMENSIONS절에 지정하는 컬럼 또는 컬럼 세트로 판별합니다. MDC 테이블을 작성할 때 다음과 같은 두 개의 인덱스가 자동으로 작성됩니다.

- 단일 차원에 대해 차지하는 각 블록을 지정하는 포인터가 포함된 차원 블록 인덱스
- 모든 차원 키 컬럼이 들어 있으며 삽입 및 갱신 활동 중 클러스터링을 유지보수하는 데 사용되는 복합 블록 인덱스

옵티마이저는 특정 쿼리의 가장 효율적인 액세스 플랜을 판별할 때 차원 블록 인덱스를 사용하는 액세스 플랜을 고려합니다. 쿼리에 차원 값에 대한 술어가 있는 경우 옵티마이저는 차원-블록 인덱스를 사용하여 이러한 값이 포함된 Extent를 식별하고 Extent에서 페치할 수 있습니다. Extent는 디스크에서 실제로 인접하는 페이지이므로 입출력이 최소화되고 이로 인해 성능이 향상됩니다.

데이터 액세스 플랜의 분석이 이러한 인덱스로 쿼리 성능이 향상됨을 표시하는 경우 특정 RID 인덱스를 작성할 수도 있습니다.

클러스터링

시간이 경과함에 따라 갱신으로 인해 데이터 페이지의 행 위치가 변경되어 인덱스와 해당 데이터 페이지 사이에 존재하는 *클러스터링* 등급이 낮아집니다.

선택된 키에 대해 테이블을 재구성하면 데이터가 다시 클러스터링됩니다. 클러스터된 인덱스는 기본 테이블에 있는 데이터의 시퀀스 액세스를 향상시킬 수 있기 때문에 범위 술어를 갖는 컬럼에 가장 유용합니다. 그 결과, 비슷한 값이 동일한 데이터 페이지에 위치하므로 페이지 페치(fetch)가 더 적어집니다.

일반적으로 한 테이블 내에서 한 인덱스만 높은 등급으로 클러스터화될 수 있습니다.

인덱스에 적용할 클러스터링 등급을 점검하려면 해당 노드를 더블 클릭하여 인덱스 통계 창을 표시하십시오. 클러스터 비율 또는 클러스터 인수 값이 이 창에 표시됩니다. 값이 낮은 경우 테이블의 데이터를 재구성하십시오.

파티션 내 병렬 처리의 최적화 전략

SQL문이 컴파일될 때 병렬 처리 수준이 지정된 경우 옵티마이저는 단일 데이터베이스 파티션에서 쿼리를 병렬식으로 실행할 액세스 플랜을 선택할 수 있습니다.

런타임에 쿼리를 실행하기 위해 서브에이전트라고 하는 여러 데이터베이스 에이전트가 작성됩니다. 서브에이전트 수는 SQL문 컴파일 시 지정된 병렬 처리 수준 이하입니다.

액세스 플랜을 병렬 처리하기 위해 옵티마이저는 액세스 플랜을 각 서브에이전트가 실행하는 부분과 코디네이팅 에이전트가 실행하는 부분으로 나눕니다. 서브에이전트는 테이블 큐를 통해 데이터를 코디네이팅 에이전트 또는 기타 서브에이전트로 전달합니다. 파티션된 데이터베이스 환경에서 서브에이전트는 다른 데이터베이스 파티션의 서브에이전트에서 테이블 큐를 통해 데이터를 보내거나 받을 수 있습니다.

파티션 내 병렬 스캔 전략

관계형 스캔 및 인덱스 스캔은 동일한 테이블 또는 인덱스에서 병렬식으로 수행할 수 있습니다. 병렬 관계형 스캔의 경우 테이블은 서브에이전트에 지정된 페이지 또는 행 범위로 나눕니다. 서브에이전트는 지정된 범위를 스캔하고 현재 범위에서 작업을 완료하면 다른 범위가 지정됩니다.

병렬 인덱스 스캔의 경우 인덱스 키 값 및 키 값의 인덱스 항목 수에 따라 인덱스를 레코드 범위로 나눕니다. 서브에이전트에 레코드 범위를 지정하여 병렬 인덱스 스캔을 병렬 테이블 스캔과 같이 진행합니다. 서브에이전트가 현재 범위에서 작업을 완료하면 새 범위가 지정됩니다.

옵티마이저는 스캔 단위(페이지 또는 행) 및 스캔 세분화도를 판별합니다.

병렬 스캔은 서브에이전트들로 균일하게 작업을 분산하는 기능을 제공합니다. 병렬 스캔의 목표는 서브에이전트들 간 로드 균형을 맞추고 모두 동일하게 사용 중인 상태를 유지하기 위한 것입니다. 사용 중인 서브에이전트 수가 사용 가능한 프로세서 수와 동일하며 디스크가 입출력 요청으로 과부하되지 않은 경우 머신 자원을 효과적으로 사용 중입니다.

다른 액세스 플랜 전략으로 쿼리 실행 시 데이터 불균형이 발생할 수 있습니다. 옵티마이저는 서브에이전트들 간 데이터 균형을 유지하는 병렬 전략을 선택합니다.

파티션 내 병렬 정렬 전략

옵티마이저는 다음의 병렬 정렬 전략 중 하나를 선택할 수 있습니다.

- 라운드 로빈 정렬

  이 정렬은 *재분산 정렬*이라고도 합니다. 이 방법은 공유 메모리를 사용하여 모든 서브에이전트에 가능하면 균일하게 데이터를 효율적으로 재분산합니다. 라운드 로빈 알고리즘을 사용하여 균일한 분산을 제공합니다. 먼저 각 서브에이전트의 개별 정렬을 작성합니다. 삽입 단계 중 라운드 로빈 방식으로 각 개별 정렬에 서브에이전트를 삽입하면 데이터가 더욱 균일하게 분산됩니다.
- 파티션된 정렬

  각 서브에이전트마다 정렬을 작성한다는 점에서 라운드 로빈 정렬과 비슷합니다. 서브에이전트는 정렬 컬럼에 해시 함수를 적용하여 행을 삽입해야 하는 정렬을 판별합

니다. 예를 들어, 병합 조인의 내부 및 외부 테이블이 파티션된 정렬인 경우 서브에이전트는 병합 조인을 사용하여 해당 테이블 부분을 조인하고 병렬식으로 실행할 수 있습니다.

- 복제된 정렬

  이 정렬은 각 서브에이전트에 모든 정렬 출력이 필요한 경우 사용됩니다. 하나의 정렬이 작성되고 행이 정렬에 삽입될 때 서브에이전트가 동기화됩니다. 정렬이 완료되면 각 서브에이전트는 전체 정렬을 읽습니다. 행 수가 작으면 이 정렬을 사용하여 데이터 스트림의 재조정을 유지할 수 있습니다.

- 공유 정렬

  이 정렬은 서브에이전트가 라운드 로빈 정렬과 비슷한 방식으로 서브에이전트들에 데이터를 분산하기 위해 정렬된 결과에서 병렬 스캔을 연다는 점을 제외하고 복제된 정렬과 동일합니다.

파티션 내 병렬 임시 테이블

서브에이전트들은 협력하여 행을 동일한 테이블에 삽입함으로써 임시 테이블을 작성할 수 있습니다. 이 테이블을 *공유 임시 테이블*이라고 합니다. 서브에이전트는 데이터 스트림을 복제하거나 분할할 것인지 여부에 따라 공유 임시 테이블에서 개인용 스캔 또는 병렬 스캔을 열 수 있습니다.

파티션 내 병렬 집계 전략

집계 연산은 서브에이전트가 병렬식으로 수행할 수 있습니다. 집계 연산의 경우 그룹화 컬럼에서 데이터를 정렬해야 합니다. 서브에이전트가 그룹화 컬럼 값 세트의 모든 행을 받을 것으로 보장할 수 있는 경우 전체 집계를 수행할 수 있습니다. 이전 파티션된 정렬로 인해 그룹화 컬럼에서 스트림이 이미 분할된 경우 이러한 상태가 발생할 수 있습니다.

그렇지 않으면 서브에이전트는 부분 집계를 수행하고 다른 전략을 사용하여 집계를 완료할 수 있습니다. 이러한 일부 전략은 다음과 같습니다.

- 병합 테이블 큐를 통해 코디네이터 에이전트로 부분적으로 집계된 데이터를 보내십시오. 코디네이터 에이전트가 집계를 완료합니다.
- 파티션된 정렬로 부분적으로 집계된 데이터를 삽입하십시오. 정렬은 그룹화 컬럼에서 분할되며 그룹화 컬럼 세트의 모든 행이 하나의 정렬 파티션에 포함됨을 확신합니다.
- 처리 균형 맞추기 위해 스트림을 복제해야 하는 경우 부분적으로 집계된 데이터를 복제된 정렬로 삽입할 수 있습니다. 각 서브에이전트는 복제된 정렬을 사용하여 집계를 완료하고 집계 결과의 동일한 사본을 수신합니다.

파티션 내 병렬 조인 전략

조인 연산은 서브에이전트가 병렬식으로 수행할 수 있습니다. 병렬 조인 전략은 데이터 스트림의 특성에 따라 판별됩니다.

조인의 내부 및 외부 테이블에서 데이터 스트림을 파티셔닝하거나 복제하여 조인을 병렬 처리할 수 있습니다. 예를 들어, 병렬 스캔을 위해 외부 스트림을 파티션하고 각 서브에이전트가 내부 스트림을 다시 독립적으로 평가하는 경우 중첩된 루프 조인이 병렬 처리될 수 있습니다. 내부 및 외부 스트림이 파티션된 정렬을 위해 가치 파티션된 경우 병합 조인을 병렬 처리할 수 있습니다.

조인

*조인*은 일반 정보 도메인에 기초하여 둘 이상의 테이블에 있는 데이터를 결합하는 프로세스입니다. 한 테이블의 행을 다른 테이블의 행과 쌍으로 지정합니다(해당 행의 정보가 결합 기준(*Join 술어*)에 따라 일치하는 경우).

예를 들어, 다음 두 개의 테이블을 참조하십시오.

TABLE1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

TABLE2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

PROJ_ID 컬럼의 값이 동일하도록 TABLE1과 TABLE2를 조인하려면 다음 SQL문을 사용하십시오.

```
select proj, x.proj_id, name
  from table1 x, table2 y
  where x.proj_id = y.proj_id
```

이 경우 적합한 Join 술어는 `where x.proj_id = y.proj_id`입니다.

이 쿼리로 다음의 결과 세트가 작성됩니다.

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

Join 술어의 특성과 테이블 및 인덱스 통계를 기초로 판별된 비용에 따라 옵티마이저는 다음의 조인 방법 중 하나를 선택합니다.

- 중첩된 루프 조인

- 병합 조인
- 해시 조인

두 테이블이 조인된 경우 한 테이블을 외부 테이블로 선택하고 다른 테이블을 조인의 내부 테이블로 간주합니다. 다른 테이블에 먼저 액세스하고 한 번만 스캔합니다. 내부 테이블을 여러 번 스캔하는지 여부는 조인 유형과 사용 가능한 인덱스에 따라 다릅니다. 쿼리가 두 개 이상의 테이블을 조인하는 경우에도 옵티마이저는 한 번에 두 테이블만 조인합니다. 필요하면 중간 결과를 보유할 임시 테이블이 작성됩니다.

INNER 또는 LEFT OUTER JOIN과 같은 명시적 조인 연산자를 제공하여 조인에서 테이블을 사용하는 방식을 판별할 수 있습니다. 그러나 이러한 방식으로 쿼리를 변경하기 전에 옵티마이저가 테이블을 조인하는 방법을 판별한 후 쿼리 성능을 분석하여 조인 연산자 추가 여부를 결정하도록 해야 합니다.

쿼리 최적화에 대한 데이터베이스 파티션 그룹 영향

파티션된 데이터베이스 환경에서 옵티마이저는 쿼리에 가장 적합한 액세스 플랜을 판별하면 테이블 공동 배치를 인식하고 사용합니다.

조인 쿼리에서 테이블을 자주 사용하는 경우 조인되는 각 테이블의 행을 동일한 데이터베이스 파티션에 배치하는 방식으로 테이블들을 여러 데이터베이스 파티션들로 분산해야 합니다. 조인 연산 중 두 개의 조인된 테이블의 데이터 공동 배치는 한 데이터베이스 파티션에서 다른 데이터베이스 파티션으로 데이터가 이동되는 것을 예방합니다. 두 테이블을 동일한 데이터베이스 파티션 그룹에 배치하여 데이터를 공동 배치하십시오.

테이블의 *크기*에 따라 데이터를 추가 데이터베이스 파티션에 분산시키면 쿼리를 실행하는 추정 시간이 줄어듭니다. 테이블 수, 테이블 *크기*, 테이블에서 데이터 위치 및 쿼리 유형(예: 조인 필요 여부)이 모두 쿼리 비용에 영향을 줍니다.

파티션된 데이터베이스의 조인 전략

파티션된 데이터베이스 환경의 조인 전략은 파티션되지 않은 데이터베이스 환경의 전략과 다를 수 있습니다. 표준 조인 방법에 추가 기술을 적용하여 성능을 향상시킬 수 있습니다.

자주 조인하는 테이블의 경우 테이블 공동 배치를 고려해야 합니다. 파티션된 데이터베이스 환경에서 *테이블 공동 배치*는 동일한 수의 호환 가능한 파티션 키가 있는 두 개의 테이블이 동일한 데이터베이스 파티션 그룹에 저장된 경우 발생하는 상태를 나타냅니다. 이러한 상태가 발생하면 데이터가 저장된 데이터베이스 파티션에서 조인 처리가 수행될 수 있으며 결과 세트만 코디네이터 데이터베이스 파티션으로 이동해야 합니다.

테이블 큐

파티션된 데이터베이스 환경에서 조인 기술에 대한 설명에서는 다음의 용어를 사용합니다.

- *테이블 큐*(TQ라고도 함)는 데이터베이스 파티션 간 또는 단일 파티션 데이터베이스의 프로세서 간 행을 전송하는 메커니즘입니다.
- *방향지정 테이블 큐*(DTQ라고도 함)는 행이 수신 데이터베이스 파티션 중 하나로 해시되는 테이블 큐입니다.
- *브로드캐스트 테이블 큐*(BTQ라고도 함)는 행이 모든 수신 데이터베이스로 전송되지만 해시되지 않는 테이블 큐입니다.

테이블 큐를 사용하여 다음과 같이 테이블 데이터를 전달합니다.

- 파티션 간 병렬 처리 사용 시 한 데이터베이스 파티션에서 다른 데이터베이스 파티션으로
- 파티션 내 병렬 처리 사용 시 데이터베이스 파티션 내에서
- 단일 파티션 데이터베이스 사용 시 데이터베이스 파티션 내에서

각 테이블 큐는 한 방향으로 데이터를 전달합니다. 컴파일러는 테이블 큐가 필요한 위치를 결정하고 플랜에 포함시킵니다. 플랜이 실행되면 데이터베이스 파티션들 간 연결이 테이블 큐를 시작합니다. 프로세스가 종료되면 테이블 큐가 닫힙니다.

여러 가지 유형의 테이블 큐가 있습니다.

- 비동기 테이블 큐

  이러한 테이블 큐는 응용프로그램에서 패치 요청에 앞서서 행을 읽으므로 비동기라고 합니다. FETCH문이 발행된 경우 테이블 큐에서 행을 검색합니다.

  SELECT문에서 FOR FETCH ONLY절을 지정할 때 비동기 테이블 큐를 사용합니다. 행만 페치하는 경우 비동기 테이블 큐가 빠릅니다.
- 동기 테이블 큐

  이러한 테이블 큐는 응용프로그램에서 발행한 각 FETCH문에 대한 한 행을 읽으므로 동기라고 합니다. 각 데이터베이스 파티션에서 커서는 해당 데이터베이스 파티션에서 읽을 다음 행에 위치합니다.

  SELECT문에서 FOR FETCH ONLY절을 지정하지 않는 경우 동기 테이블 큐를 사용합니다. 파티션된 데이터베이스 환경에서 행을 갱신하는 경우 데이터베이스 관리 프로그램은 동기 테이블 큐를 사용합니다.
- 병합 테이블 큐

  이러한 테이블 큐는 순서를 보존합니다.
- 비병합 테이블 큐

*일반 테이블 큐*라고도 하는 이러한 테이블 큐는 순서를 보존하지 않습니다.

- 리스너 테이블 큐(LTQ라고도 함)

  이러한 테이블 큐는 상관 서브쿼리에서 사용합니다. 상관값은 서브쿼리까지 전달되며 이러한 유형의 테이블 큐를 사용하여 결과는 상위 쿼리 블록까지 다시 전달됩니다.

파티션된 데이터베이스의 조인 방법

일부 조인 방법은 공동 배치 조인, 브로드캐스트 외부 테이블 조인, 방향지정 외부 테이블 조인, 방향지정 내부 테이블 및 외부 테이블 조인, 브로드캐스트 내부 테이블 조인 및 방향지정 내부 테이블 조인과 같은 파티션된 조인 환경에 사용할 수 있습니다.

다음 다이어그램에서 q1, q2 및 q3은 테이블 큐를 나타냅니다. 참조된 테이블은 두 개의 데이터베이스 파티션으로 나뉘며 화살표는 테이블 큐를 보내는 방향을 나타냅니다. 코디네이터 데이터베이스 파티션은 데이터베이스 파티션 0입니다.

공동 배치 조인

공동 배치 조인은 데이터가 상주하는 데이터베이스 파티션에서 로컬로 발생합니다. 데이터베이스 파티션은 조인이 완료된 후 다른 데이터베이스 파티션으로 데이터를 보냅니다. 옵티마이저가 공동 배치 조인을 고려할 수 있도록 조인된 테이블을 공동 배치해야 하며 해당 분산 키 쌍이 모두 동등 Join 술어에 있어야 합니다. 378 페이지의 그림 55에서 예를 제공합니다.

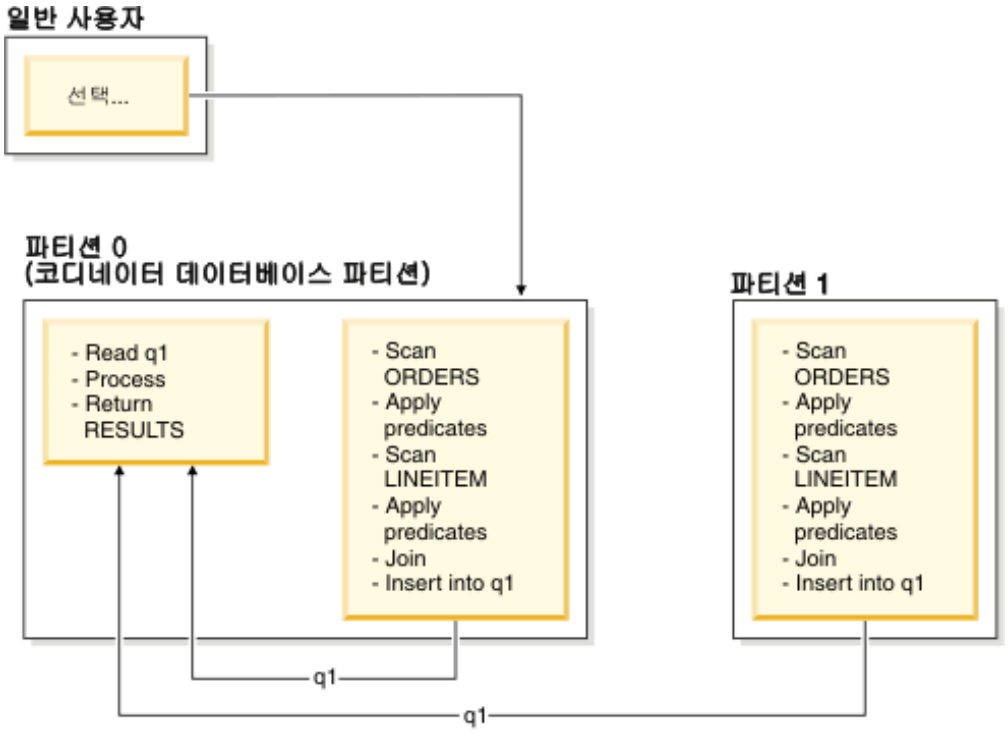


그림 55. 공동 배치 조인 예

LINEITEM 및 ORDERS 테이블은 모두 ORDERKEY 컬럼에서 파티션됩니다. 각 데이터베이스 파티션에서 로컬로 조인을 수행합니다. 이 예에서 Join 술어는 `orders.orderkey = lineitem.orderkey`로 가정합니다.

복제된 구체화된 쿼리 테이블(MQT)로 공동 배치 조인의 가능성이 증가합니다.

브로드캐스트 외부 테이블 조인

브로드캐스트 외부 테이블 조인은 조인된 테이블 간 동등 Join 술어가 없는 경우 사용할 수 있는 병렬 조인 전략을 나타냅니다. 또한 가장 비용 효율적인 조인 방법으로 입증된 다른 상황에서 사용할 수 있습니다. 예를 들어, 하나의 매우 큰 테이블이 있고 하나의 매우 작은 테이블이 있으며 Join 술어 컬럼에서 이러한 테이블이 분할되지 않은 경우 브로드캐스트 외부 테이블 조인이 발생할 수 있습니다. 두 테이블을 분할하는 대신 작은 테이블을 큰 테이블로 브로드캐스트하는 것이 저렴할 수 있습니다. 379 페이지의 그림 56에서 예를 제공합니다.

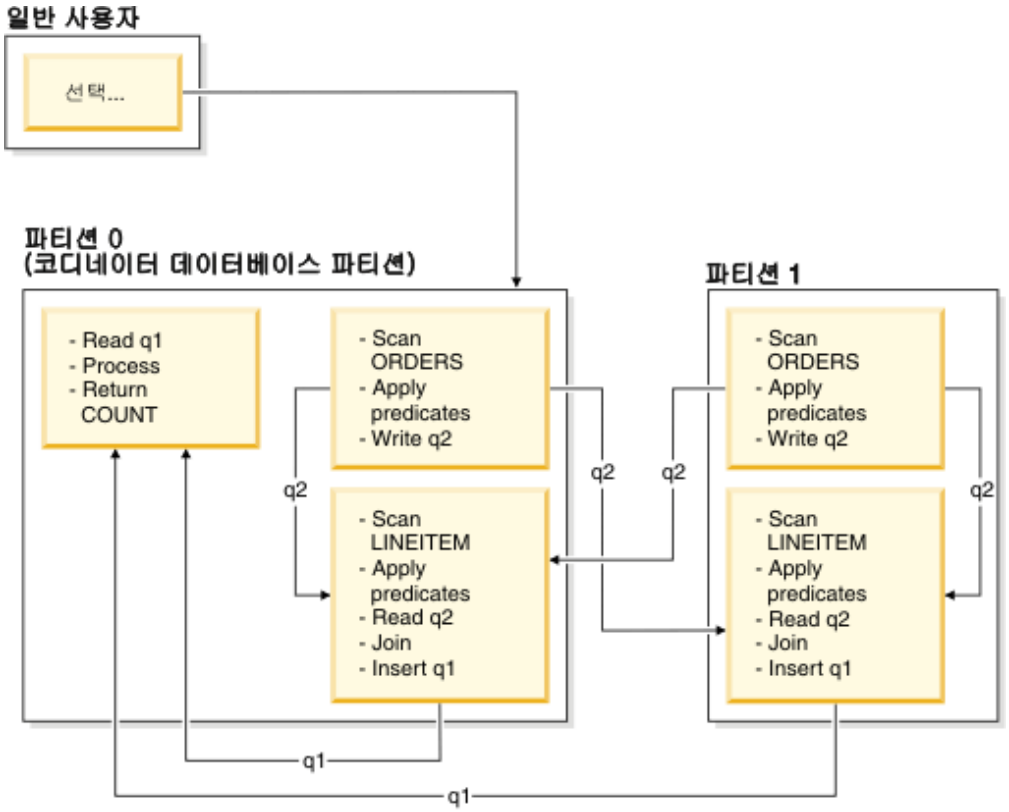
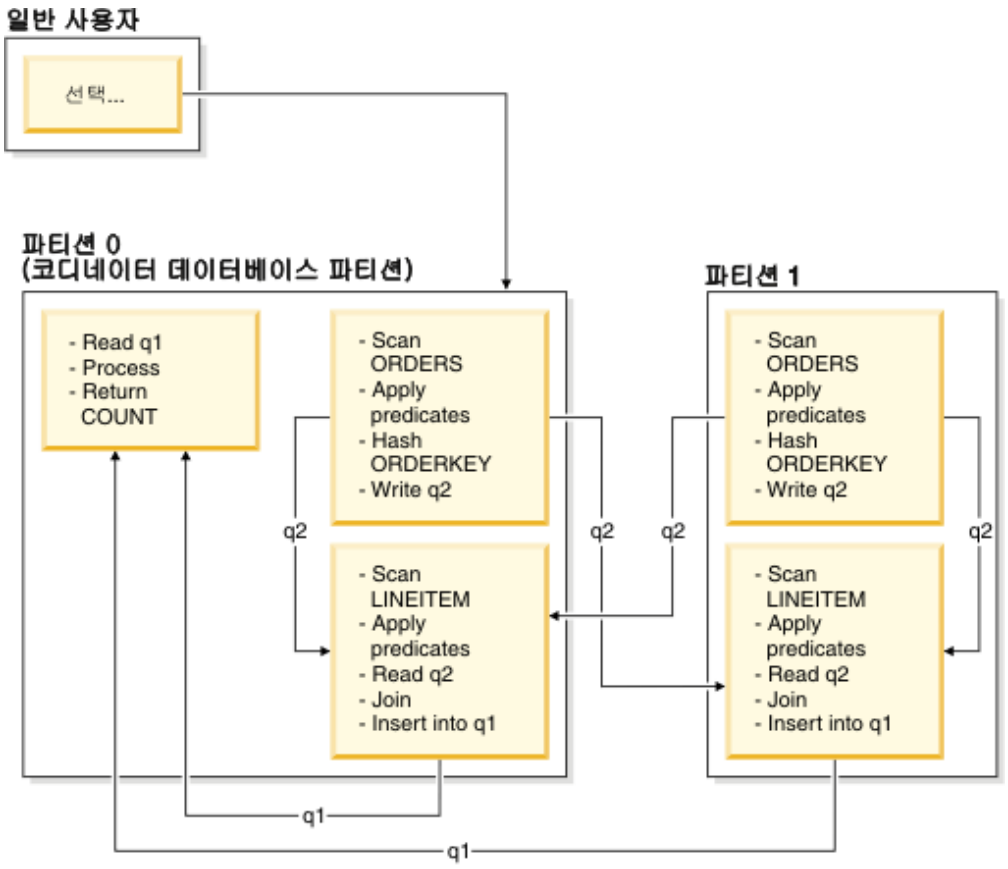


그림 56. 브로드캐스트 외부 테이블 조인 예

ORDERS 테이블을 LINEITEM 테이블이 있는 모든 데이터베이스 파티션으로 보냅니다. 테이블 큐 q2를 내부 테이블의 모든 데이터베이스 파티션으로 브로드캐스트합니다.

방향지정 외부 테이블 조인

방향지정 외부 테이블 조인 전략에서 외부 테이블의 각 행은 내부 테이블의 분할 속성에 따라 내부 테이블의 한 부분으로 보냅니다. 이 데이터베이스 파티션에서 조인이 발생합니다. 380 페이지의 그림 57에서 예를 제공합니다.



LINEITEM 테이블은 ORDERKEY 컬럼에서 파티션됩니다. ORDERS 테이블은 다른 컬럼에서 파티션됩니다. ORDERS 테이블을 해시하고 LINEITEM 테이블의 올바른 데이터베이스 파티션으로 보냅니다. 이 예에서 Join 술어는 `orders.orderkey = lineitem.orderkey`로 가정합니다.

그림 57. 방향지정 외부 테이블 조인 예

방향지정 내부 테이블 및 외부 테이블 조인

방향지정 내부 테이블 및 외부 테이블 조인 전략에서 조인 컬럼의 값에 따라 모든 외부 및 내부 테이블의 행을 데이터베이스 파티션 세트로 방향지정합니다. 이러한 데이터베이스 파티션에서 조인이 발생합니다. 381 페이지의 그림 58에서 예를 제공합니다.

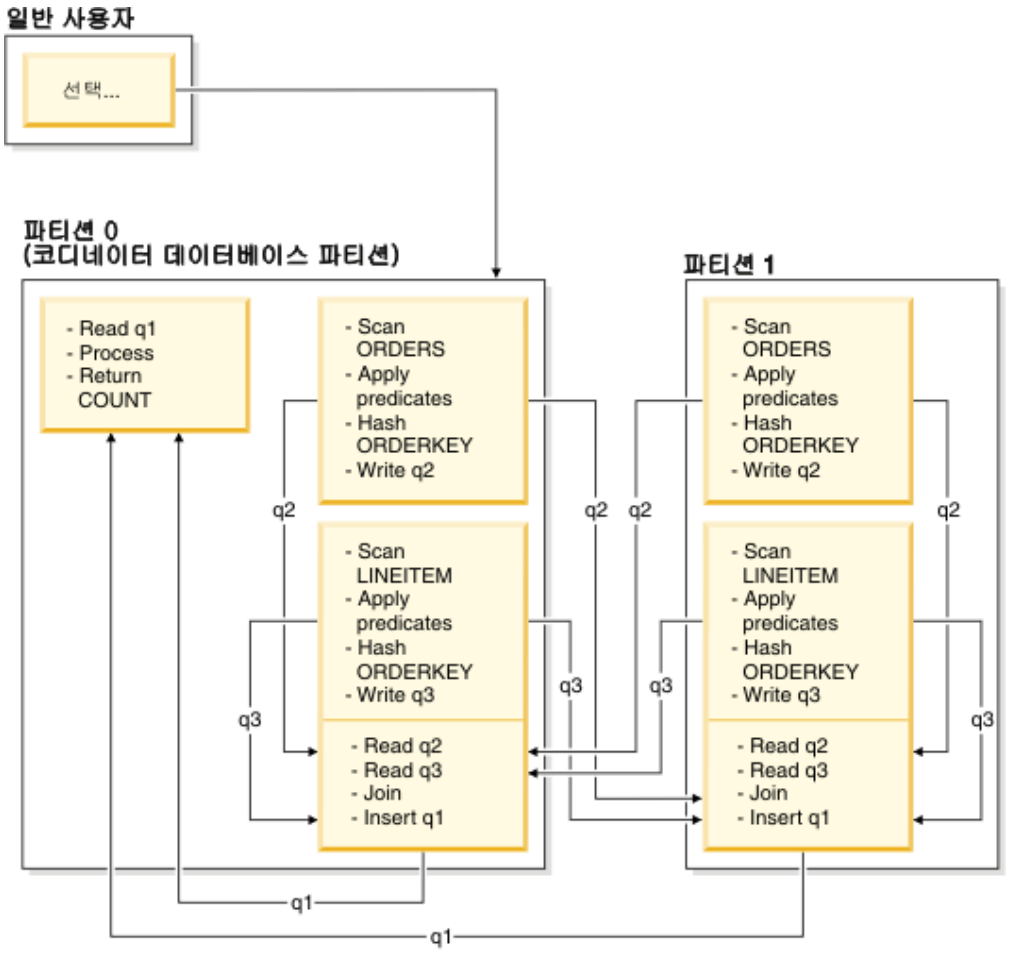
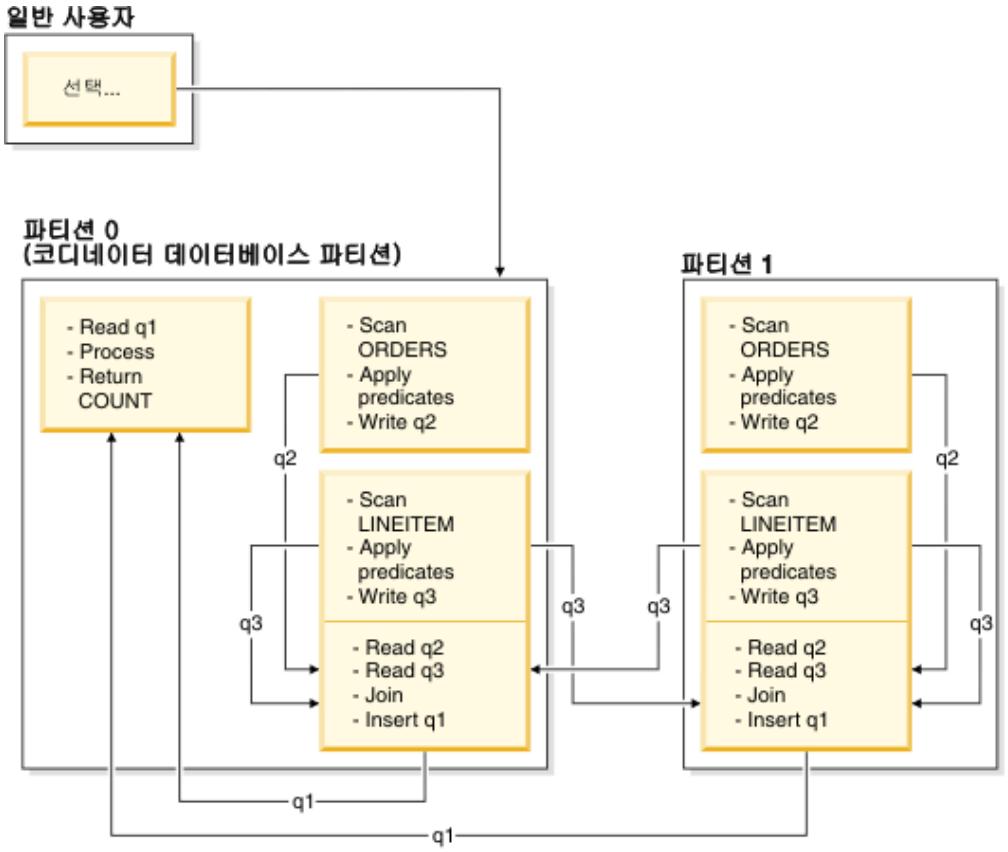


그림 58. 방향지정 내부 테이블 및 외부 테이블 조인 예

어느 테이블도 ORDERKEY 컬럼에서 파티션되지 않습니다. 두 테이블을 모두 해시하고 조인된 새 데이터베이스 파티션으로 보냅니다. 테이블 큐 q2 및 q3를 모두 방향지정합니다. 이 예에서 Join 술어는 `orders.orderkey = lineitem.orderkey`로 가정합니다.

브로드캐스트 내부 테이블 조인

브로드캐스트 내부 테이블 조인 전략에서 내부 테이블을 외부 테이블의 모든 데이터베이스 파티션으로 브로드캐스트합니다. 382 페이지의 그림 59에서 예를 제공합니다.

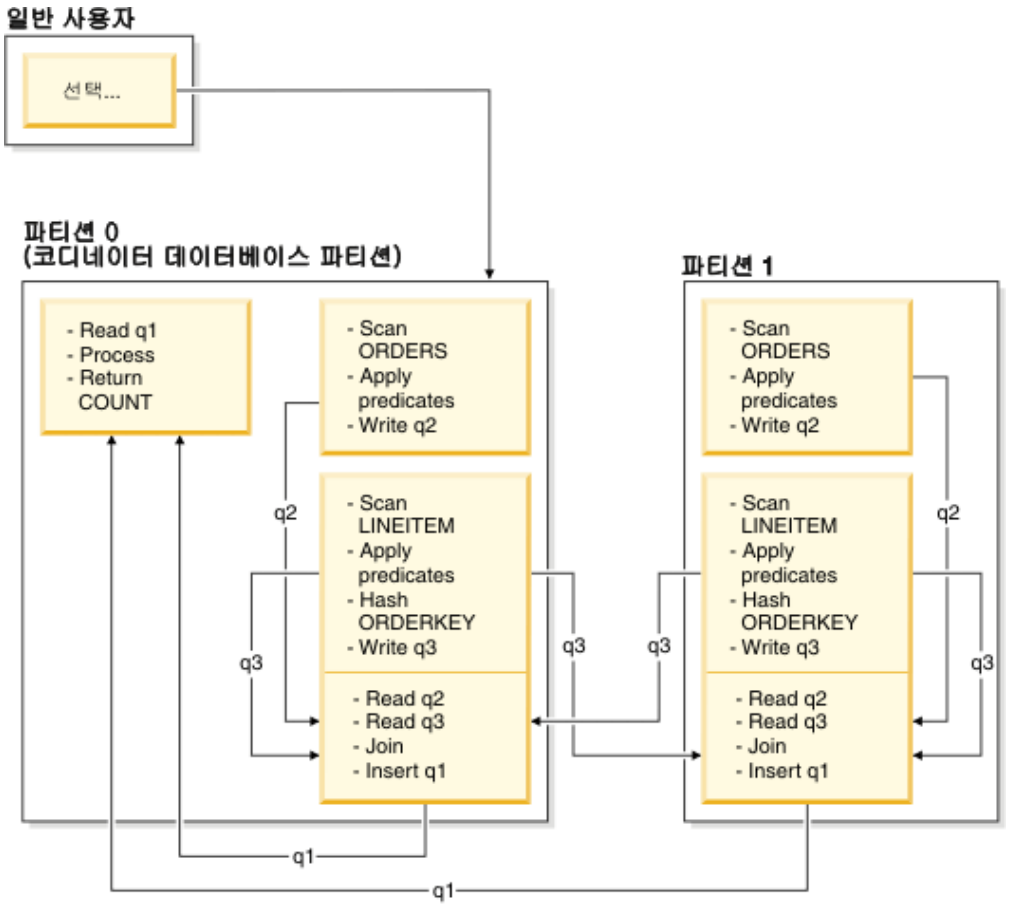


LINEITEM 테이블을 ORDERS 테이블이 있는 모든 데이터베이스 파티션으로 보냅니다. 테이블 큐 q3를 외부 테이블의 모든 데이터베이스 파티션으로 브로드캐스트합니다.

그림 59. 브로드캐스트 내부 테이블 조인 예

방향지정 내부 테이블 조인

방향지정 내부 테이블 조인 전략에서 내부 테이블의 각 행은 외부 테이블의 분할 속성에 따라 외부 테이블에 있는 하나의 데이터베이스 파티션으로 보냅니다. 이 데이터베이스 파티션에서 조인이 발생합니다. 383 페이지의 그림 60에서 예를 제공합니다.



ORDERS 테이블은 ORDERKEY 컬럼에서 파티션됩니다. LINEITEM 테이블은 다른 컬럼에서 파티션됩니다. LINEITEM 테이블을 해시하고 ORDERS 테이블의 올바른 데이터베이스 파티션으로 보냅니다. 이 예에서 Join 술어는 `orders.orderkey = lineitem.orderkey`로 가정합니다.

그림 60. 방향지정 내부 테이블 조인 예

파티션된 데이터베이스 환경에서 복제된 구체화된 쿼리 테이블

복제된 구체화된 쿼리 테이블(MQT)을 통해 데이터베이스가 테이블 데이터의 사전 계산된 값을 관리할 수 있으므로 파티션된 데이터베이스 환경에서 자주 실행하는 조인의 성능이 향상됩니다.

이 컨텍스트에서 복제된 MQT는 데이터베이스 내 복제와 관계가 있다는 점을 참고하십시오. 데이터베이스 내 복제는 서브스크립션, 제어 테이블, 다른 데이터베이스 및 다른 운영 체제에 있는 데이터와 관련되어 있습니다.

다음 예에서 아래 사항을 참조하십시오.

- SALES 테이블은 다중 파티션 테이블 스페이스 REGIONTABLESPACE에 있으며 REGION 컬럼에서 분할되어 있습니다.

- EMPLOYEE 및 DEPARTMENT 테이블은 단일 파티션 데이터베이스 파티션 그룹에 있습니다.

EMPLOYEE 테이블의 정보에 따라 복제된 MQT를 작성하십시오.

```
create table r_employee as (
  select empno, firstnme, midinit, lastname, workdept
    from employee
  )
  data initially deferred refresh immediate
  in regiontablespace
  replicated
```

복제된 MQT의 내용을 갱신하십시오.

```
refresh table r_employee
```

REFRESH문을 사용한 후 다른 테이블과 마찬가지로 복제된 테이블에 대해 runstats 유틸리티를 호출해야 합니다.

다음의 쿼리는 직원별 급여, 부서 총액 및 전체 총액을 계산합니다.

```
select d.mgrno, e.empno, sum(s.sales)
  from department as d, employee as e, sales as s
  where
    s.sales_person = e.lastname and
    e.workdept = d.deptno
  group by rollup(d.mgrno, e.empno)
  order by d.mgrno, e.empno
```

한 데이터베이스 파티션에만 상주하는 EMPLOYEE 테이블을 사용하는 대신 데이터베이스 관리 프로그램은 SALES 테이블이 저장된 각 데이터베이스 파티션에서 복제된 MQT인 R_EMPLOYEE를 사용합니다. 조인을 수행할 때 네트워크를 통해 직원 정보를 각 데이터베이스 파티션으로 이동할 필요가 없으므로 성능이 향상됩니다.

공동 배치 조인에서 복제된 구체화된 쿼리 테이블

복제된 MQT는 조인의 공동 배치에서도 도움이 됩니다. 예를 들어, 스타 스키마에 20개의 데이터베이스 파티션에 분산된 대형 사실 테이블이 있으면 사실 테이블과 차원 테이블 간 조인은 이러한 테이블이 공동 배치된 경우 가장 효율적입니다. 모든 테이블이 동일한 데이터베이스 파티션 그룹에 있는 경우 공동 배치 조인을 위해 최대 하나의 차원 테이블이 올바로 파티션됩니다. 사실 테이블의 조인 컬럼이 사실 테이블의 분배 키에 해당하지 않으므로 공동 배치 조인에서 다른 차원 테이블을 사용할 수 없습니다.

C1에서 분할된 FACT 테이블(C1, C2, C3, ...), C1에서 분할된 DIM1 테이블(C1, dim1a, dim1b, ...), C2에서 분할된 DIM2 테이블(C2, dim2a, dim2b, ...) 등을 참조하십시오. 이 경우 `dim1.c1 = fact.c1` 술어가 공동 배치되므로 FACT와 DIM1 간 조인이 가장 좋습니다. 이 두 테이블은 모두 C1 컬럼에서 분할됩니다.

그러나 FACT는 C2 컬럼이 아닌 C1 컬럼에서 분할되므로 DIM2와 dim2.c2 = fact.c2 술어가 관련된 조인을 공동 배치할 수 없습니다. 이 경우 사실 테이블의 데이터베이스 파티션 그룹에서 DIM2를 복제할 수 있으므로 각 데이터베이스 파티션에서 조인이 로컬로 발생합니다.

복제된 MQT를 작성할 때 소스 테이블은 데이터베이스 파티션 그룹에서 단일 파티션 테이블 또는 다중 파티션 테이블이 될 수 있습니다. 대부분의 경우 복제된 테이블은 작으며 단일 파티션 데이터베이스 파티션 그룹에 배치될 수 있습니다. 테이블에서 컬럼의 서브세트만 지정하거나 술어를 통해 규정 행 수를 제한하여 복제할 데이터를 제한할 수 있습니다.

복제된 MQT는 다중 파티션 데이터베이스 파티션 그룹에서도 작성할 수 있으므로 소스 테이블의 사본이 모든 데이터베이스 파티션에서 작성됩니다. 대형 사실 테이블과 차원 테이블 간 조인은 소스 테이블을 모든 데이터베이스 파티션으로 브로드캐스트하는 경우보다 이 환경에서 로컬로 발생할 가능성이 높습니다.

복제된 테이블의 인덱스는 자동으로 작성되지 않습니다. 소스 테이블의 인덱스와 다른 인덱스를 작성할 수 있습니다. 그러나 소스 테이블에 없는 제한조건 위반을 예방하기 위해 소스 테이블에 동일한 제한조건이 발생하는 경우에도 복제된 테이블에서 고유 인덱스를 작성하거나 제한조건을 정의할 수 없습니다.

복제된 테이블을 쿼리에 직접 참조할 수 있지만 특정 데이터베이스 파티션의 테이블 데이터를 보기 위해 DBPARTITIONNUM 스칼라 함수를 복제된 테이블과 함께 사용할 수 없습니다.

DB2 Explain 기능을 사용하여 복제된 MQT를 액세스 플랜이 쿼리에 사용했는지 여부를 판별하십시오. 옵티마이저가 선택한 액세스 플랜이 복제된 MQT를 사용하는지 여부는 조인할 데이터에 따라 다릅니다. 옵티마이저가 데이터베이스 파티션 그룹의 다른 데이터베이스 파티션으로 원래의 소스 테이블을 브로드캐스트하는 것이 저렴하다고 판별한 경우 복제된 MQT를 사용하지 않을 수도 있습니다.

레슨 4. 파티션된 데이터베이스 환경에서 액세스 플랜 개선

여러 조정 활동 수행 시 기본 쿼리에 사용되는 액세스 플랜 및 관련 창을 변경하는 방법을 학습합니다.

일러스트레이션이 포함된 일련의 예제를 통해 **runstats** 명령을 사용하고 적합한 인덱스를 추가하여 단순 쿼리에 대해서도 액세스 플랜의 예상 총 비용을 개선할 수 있는 방법을 학습합니다.

Visual Explain 사용 경험이 늘어나면 다른 쿼리 조정 방법을 찾을 수 있습니다.

액세스 플랜 그래프에 대한 작업

네 개의 샘플 Explain 스냅샷을 예제로 사용하여 조정이 데이터베이스 성능에 얼마나 중요한 파트인지 학습합니다.

Explain 스냅샷과 연관된 쿼리는 1 - 4번입니다. 각 쿼리에서는 동일한 SQL 또는 XQuery 명령문을 사용합니다(레슨 1에 설명되어 있음).

```
SELECT S.ID,SNAME,O.DEPTNAME,SALARY+COMM
FROM ORG O, STAFF S
WHERE
 O.DEPTNUMB = S.DEPT AND
 S.JOB <> 'Mgr' AND
 S.SALARY+S.COMM > ALL ( SELECT ST.SALARY*.9
                         FROM STAFF ST
                         WHERE ST.JOB='Mgr' )
ORDER BY S.NAME
```

그러나 각각의 반복되는 쿼리에서는 이전 실행에서보다 더 많은 조정 기술을 사용합니다. 예를 들어, 쿼리 1에는 성능 조정이 없지만 쿼리 4에서는 성능 조정이 가장 많이 수행됩니다. 쿼리 간의 다른점은 아래 설명되어 있습니다.

쿼리 1
인덱스 및 통계 없이 쿼리 실행
쿼리 2
쿼리에서 테이블 및 인덱스의 현재 통계 수집
쿼리 3
쿼리에서 테이블을 조인하는 데 사용되는 컬럼에 대한 인덱스 작성
쿼리 4
테이블 컬럼에 대한 추가 인덱스 작성

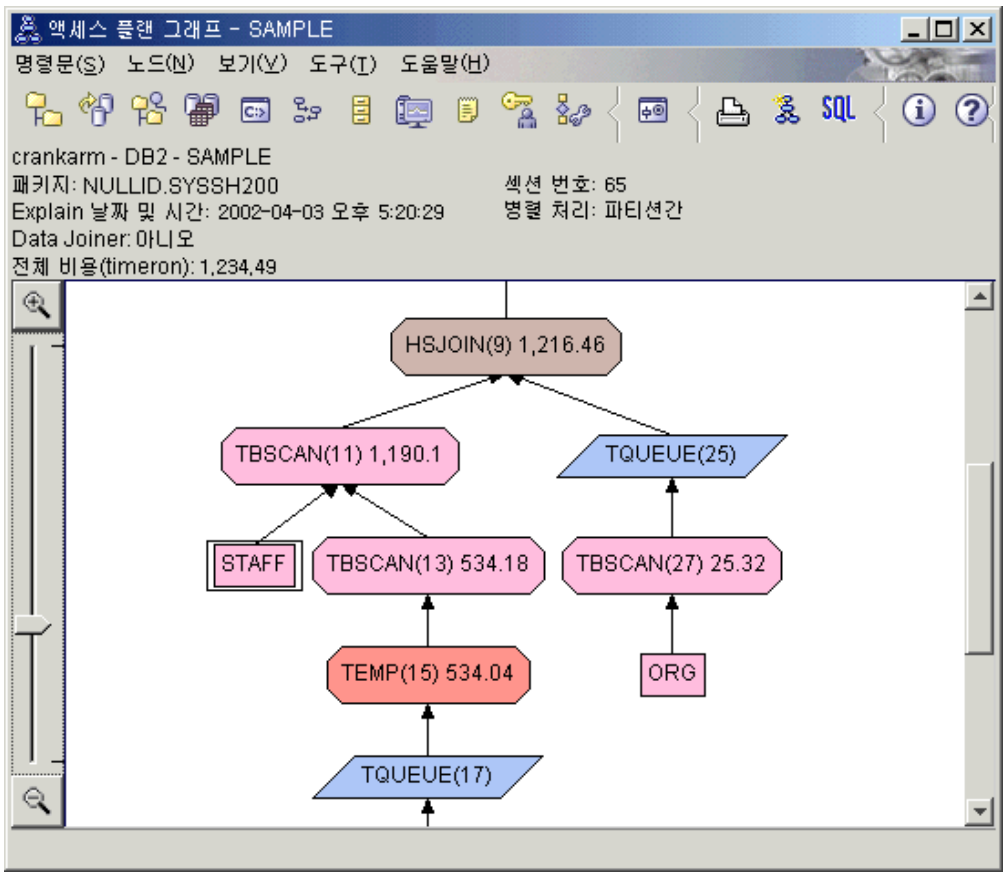
해당 예는 파티션 간 병렬 처리를 사용하여 7개의 실제 노드가 있는 RS/6000® SP™ 머신에서 생성되었습니다.

파티션된 데이터베이스 환경에서 인덱스 및 통계 없이 쿼리 실행

이 예제에서는 인덱스 및 통계 없이 SQL 쿼리에 사용할 액세스 플랜을 작성합니다.

해당 쿼리에 대한 액세스 플랜 그래프를 보려면 다음과 같이 수행하십시오(쿼리 1).

1. 제어 센터에서 SAMPLE 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 **Explain문 실행기록 표시**를 선택하십시오.
   Explain문 실행기록 창이 열립니다.
3. 쿼리 번호 1로 식별되는 엔트리를 더블 클릭하십시오. **쿼리 번호** 컬럼을 찾기 위해 오른쪽으로 스크롤해야 할 수도 있습니다.
   명령문의 액세스 플랜 그래프 창이 열립니다.



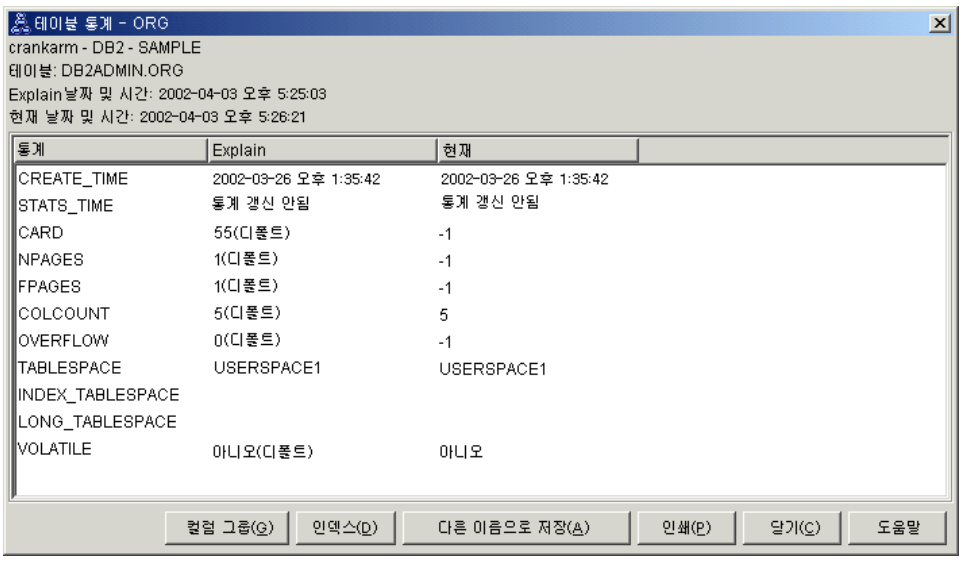
다음 질문에 대한 답은 쿼리 개선 방법을 검토하는 데 도움이 됩니다.

1. 쿼리의 각 테이블에 현재 통계가 존재합니까?

   쿼리의 각 테이블에 현재 통계가 존재하는지 점검하려면 액세스 플랜 그래프에서 각 테이블 노드 를 더블 클릭하십시오. 해당 테이블 통계 창이 열리면 **Explain** 컬럼의 **STATS_TIME** 행에 "통계가 갱신되지 않음"이라는 단어가 표시됩니다. 이는 스냅샷 작성 당시 통계를 수집하지 않았음을 나타냅니다.

   현재 통계가 존재하지 않는 경우에는 옵티마이저에서 디폴트 통계를 사용하며 이는 실제 통계와 다를 수 있습니다. 디폴트 통계는 테이블 통계 창의 **Explain** 컬럼에서 "디폴트"라는 단어를 통해 식별됩니다.

   ORG 테이블에 관한 테이블 통계 창의 정보에 따라, 옵티마이저에서 Explain된 값 옆에 표시된 대로 디폴트 통계를 사용했습니다. **STATS_TIME** 행에 표시된 바와 같이 스냅샷 작성 당시 실제 통계를 사용할 수 없었기 때문에 디폴트 통계를 사용했습니다.



2. 이 액세스 플랜에서 가장 효율적인 방법을 사용하여 데이터에 액세스합니까?

   이 액세스 플랜에는 인덱스 스캔이 아니라 테이블 스캔이 포함되어 있습니다. 테이블 스캔은 8각형으로 표시되며 다음과 같이 레이블이 지정됩니다. TBSCAN 연산자. 인덱스 스캔이 사용된 경우에는 다이아몬드형으로 표시되며 다음과 같이 레이블이 지정됩니다. IXSCAN . 적은 양의 데이터를 추출하는 경우 테이블에 관해 작성된 인덱스를 사용하는 것이 테이블 스캔보다 비용이 적게 듭니다.

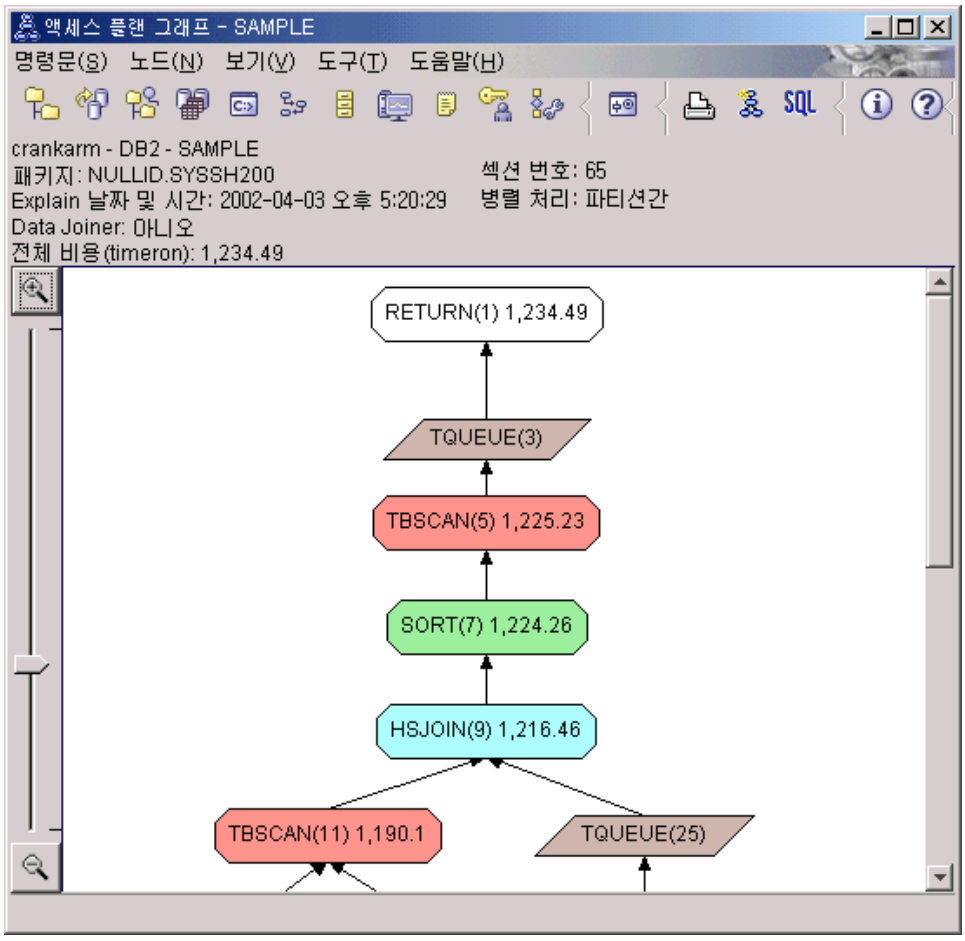
3. 이 플랜의 효율성은 어느 정도입니까?

   액세스 플랜이 실제 통계를 기본으로 하는 경우에만 액세스 플랜의 효율성을 판별할 수 있습니다. 옵티마이저가 액세스 플랜에서 디폴트 통계를 사용했으므로 플랜의 효율성을 판별할 수 없습니다.

   일반적으로, 나중에 개정된 액세스 플랜과 비교할 수 있도록 액세스 플랜의 추정 비용 총계를 기록해 두어야 합니다. 각 노드에 나열되는 비용은 쿼리의 첫 번째 단계부터 해당 노드까지 누적된 값입니다.

   주: 파티션된 데이터베이스의 경우 이 값은 대부분의 자원을 사용하는 노드의 누적 비용입니다.

   액세스 플랜 그래프 창에서 총 비용은 약 1,234 timeron으로 그래프의 맨 위에 있는 **RETURN (1)**에 표시됩니다. 계산된 비용 총계 또한 창의 맨 위 영역에 표시됩니다.



다음 내용

쿼리 2로 이동합니다.

쿼리 2에서는 runstats 실행 이후 기본 쿼리에 대한 액세스 플랜을 검토합니다. runstats 명령을 사용하여 쿼리에서 액세스한 모든 테이블에 대한 현재 통계를 옵티마이저에 제공합니다.

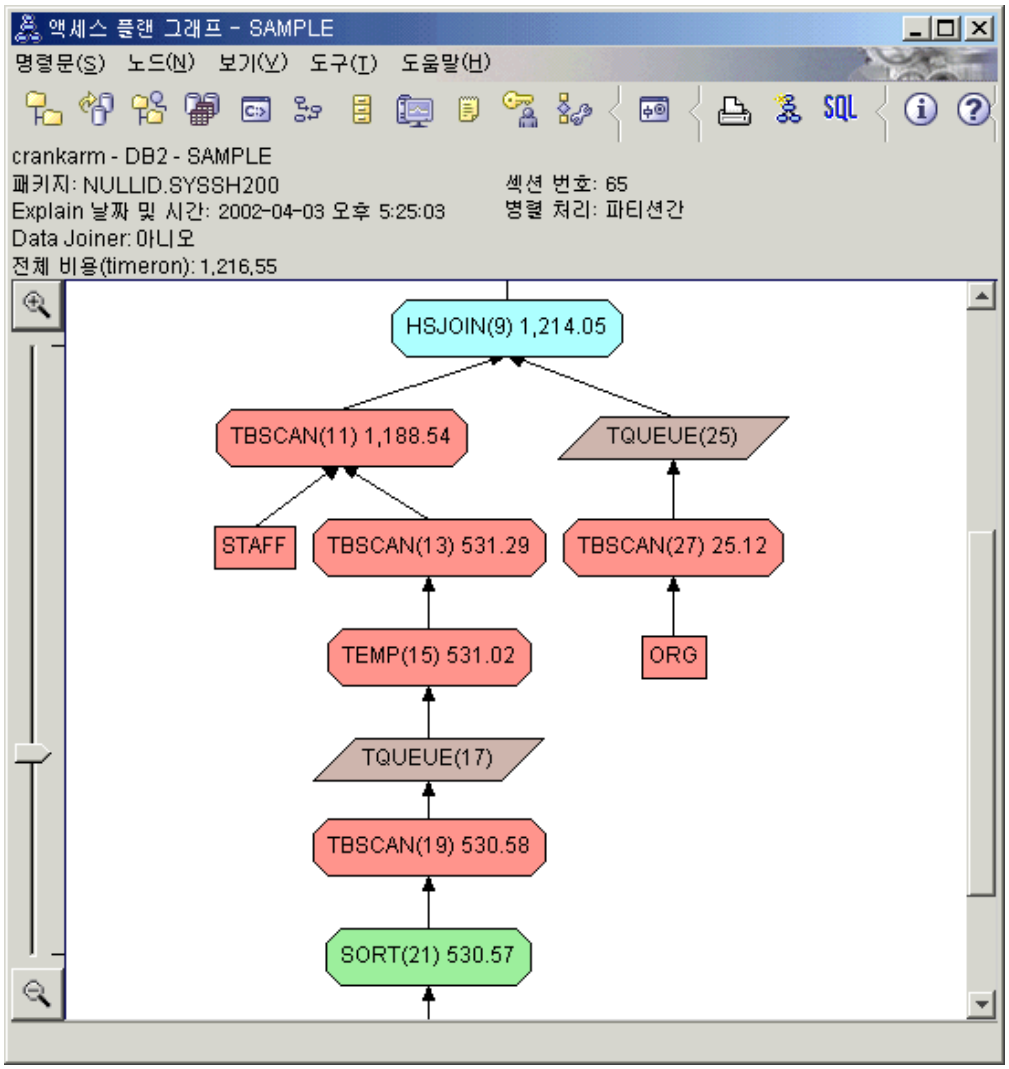
파티션된 데이터베이스 환경에서 runstats를 사용하여 테이블 및 인덱스에 관한 현재 통계 수집

이 예제에서는 runstats 명령을 사용하여 현재 통계를 수집함으로써 쿼리 1에서 설명한 액세스 플랜을 빌드합니다.

마지막으로 runstats 명령을 실행한 이후 중요한 갱신 활동이 발생했거나 새 인덱스가 작성된 경우에는 특히, runstats 명령을 사용하여 테이블 및 인덱스에 대한 현재 통계를 수집해야 합니다. 이렇게 하면 옵티마이저에 가장 정확한 정보를 제공함으로써 이를 바탕으로 최선의 액세스 플랜을 판별할 수 있습니다. 현재 통계가 사용 불가능한 경우에는 옵티마이저가 부정확한 디폴트 통계를 기본으로 비효율적인 액세스 플랜을 선택합니다.

테이블을 갱신한 *이후* runstats 명령을 사용하십시오. 그렇지 않으면 옵티마이저에 테이블이 비어 있는 것으로 표시됩니다. 연산자 세부사항 창에서 카디널리티(cardinality)가 0인 경우 이러한 문제가 발생합니다. 이런 경우 테이블 갱신을 완료한 후 runstats 명령을 다시 실행하고 영향을 받은 테이블의 Explain 스냅샷을 다시 작성하십시오.

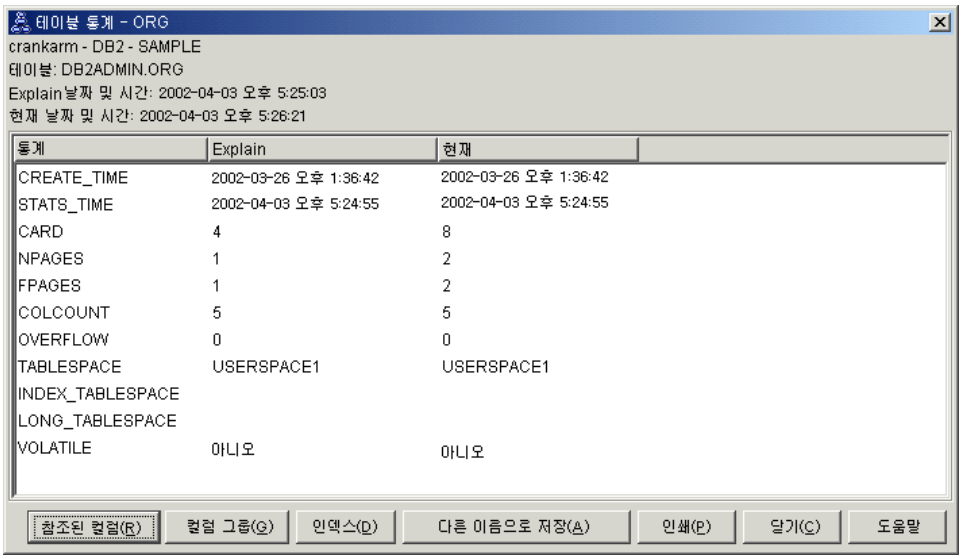
이 쿼리(쿼리 2)의 액세스 플랜 그래프를 보려면 Explain문 실행기록 창에서 쿼리 번호 2로 식별된 엔트리를 더블 클릭하십시오. 이 명령문 실행과 관련된 액세스 플랜 그래프 창이 열립니다.



다음 질문에 대한 답은 쿼리 개선 방법을 검토하는 데 도움이 됩니다.

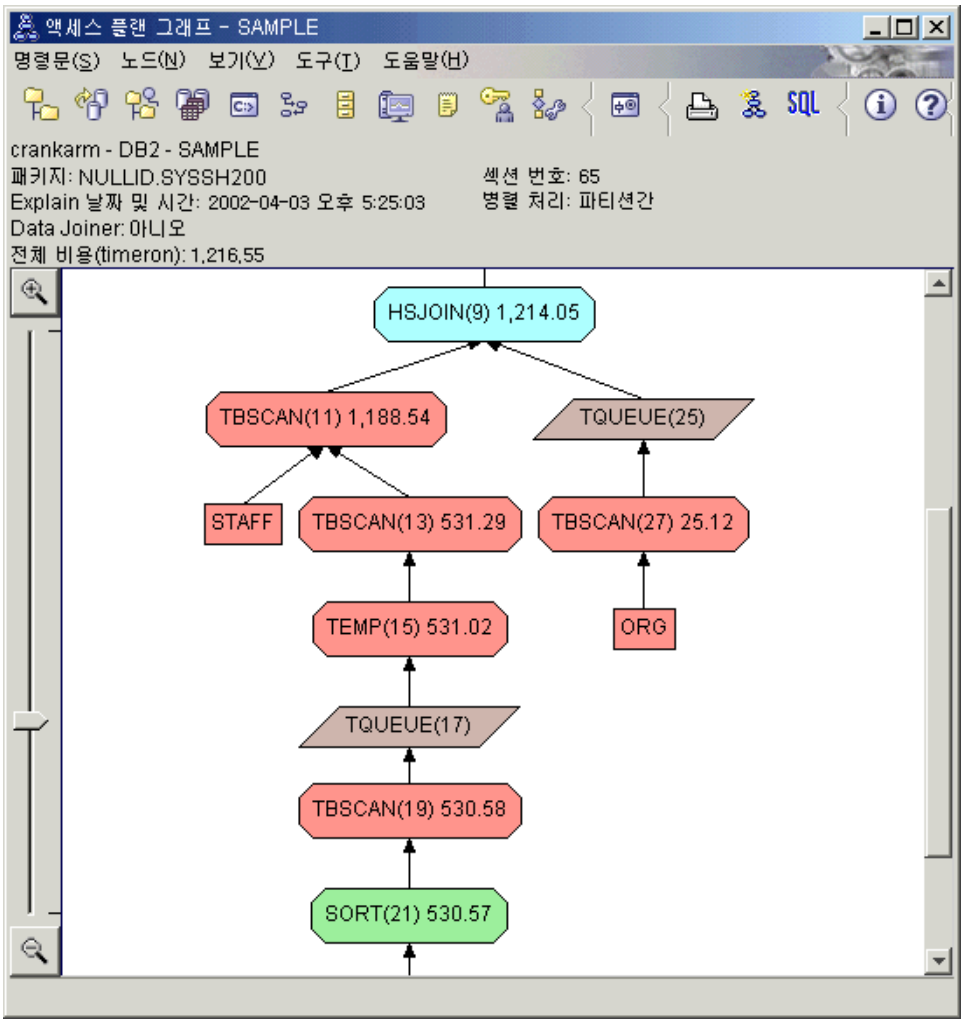
1. 쿼리의 각 테이블에 현재 통계가 존재합니까?

   ORG 테이블의 테이블 통계 창에 옵티마이저가 실제 통계를 사용했음이 표시됩니다. STATS_TIME 값은 통계가 수집된 실제 시간입니다. 통계 정밀도는 runstats 명령 실행 이후 테이블 컨텐츠에 중요한 변경사항이 있었는지 여부에 따라 다릅니다.

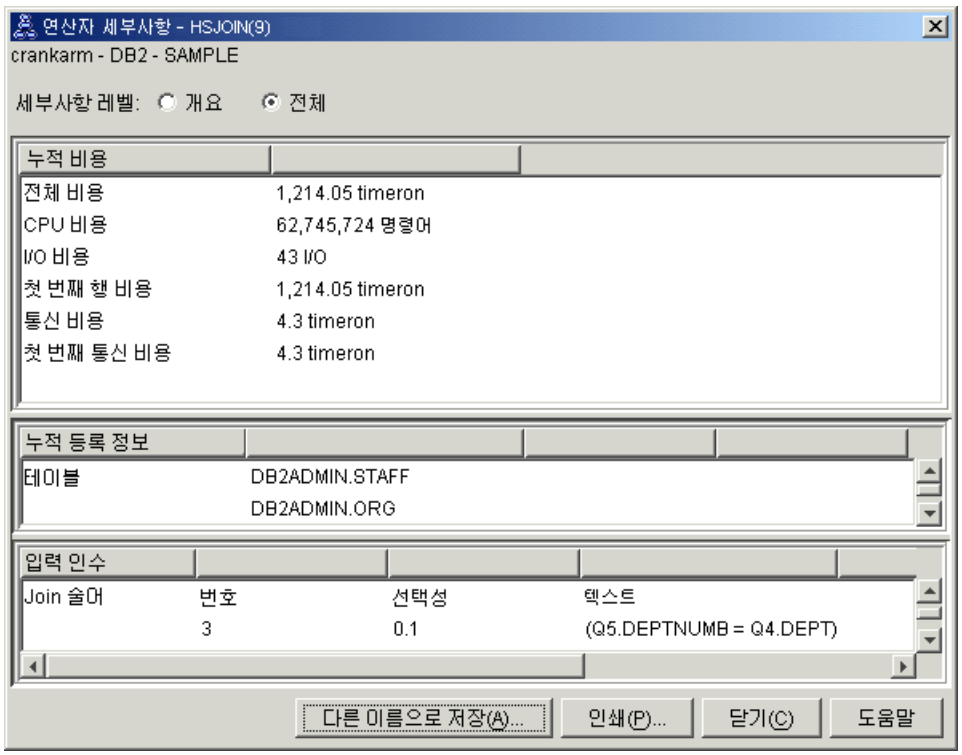


2. 이 액세스 플랜에서 가장 효율적인 방법을 사용하여 데이터에 액세스합니까?

   쿼리 1에서와 마찬가지로 쿼리 2의 액세스 플랜에서는 테이블 스캔(TBSCAN 연산자)을 사용하고 인덱스 스캔을 사용하지 않습니다(IXSCAN). 현재 통계가 존재하는 경우에도 쿼리에서 사용된 컬럼에 대한 인덱스가 없기 때문에 인덱스 스캔이 수행되지 않았습니다. 쿼리를 개선하기 위한 방법 중 하나로, 옵티마이저에 테이블을 조인하는 데 사용되는 컬럼(즉, Join술어). 이 예에서는 첫 번째 병합 스캔 조인(HSJOIN (9))입니다.



HSJOIN (9) 연산자에 관한 연산자 세부사항 창에서 **입력 인수**의 **Join 술어** 섹션을 검토하십시오. 이 조인 연산에 사용되는 컬럼은 **텍스트** 컬럼에 나열되어 있습니다. 이 예에서는 DEPTNUMB 및 DEPT 컬럼입니다.



3. 이 액세스 플랜의 효율성은 어느 정도입니까?

   최신 통계를 기본으로 하는 액세스 플랜은 항상 실질적인 계산된 비용을 산출합니다(단위는 timeron). 쿼리 1의 계산된 비용은 디폴트 통계를 기본으로 한 것이기 때문에 두 액세스 플랜 그래프의 비용을 비교하여 더 효율적인 비용을 판별할 수 없습니다. 비용이 높거나 낮은지 여부는 관계 없습니다. 유효한 효율성 측정값을 얻으려면 실제 통계를 기본으로 하는 액세스 플랜 비용을 비교해야 합니다.

다음 내용

쿼리 3으로 이동합니다.

쿼리 3에서는 DEPTNUMB 및 DEPT 컬럼에 대한 인덱스를 추가하는 경우의 효과에 대해 검토합니다. Join 술어에서 사용되는 컬럼에 대한 인덱스를 추가하면 성능이 개선됩니다.

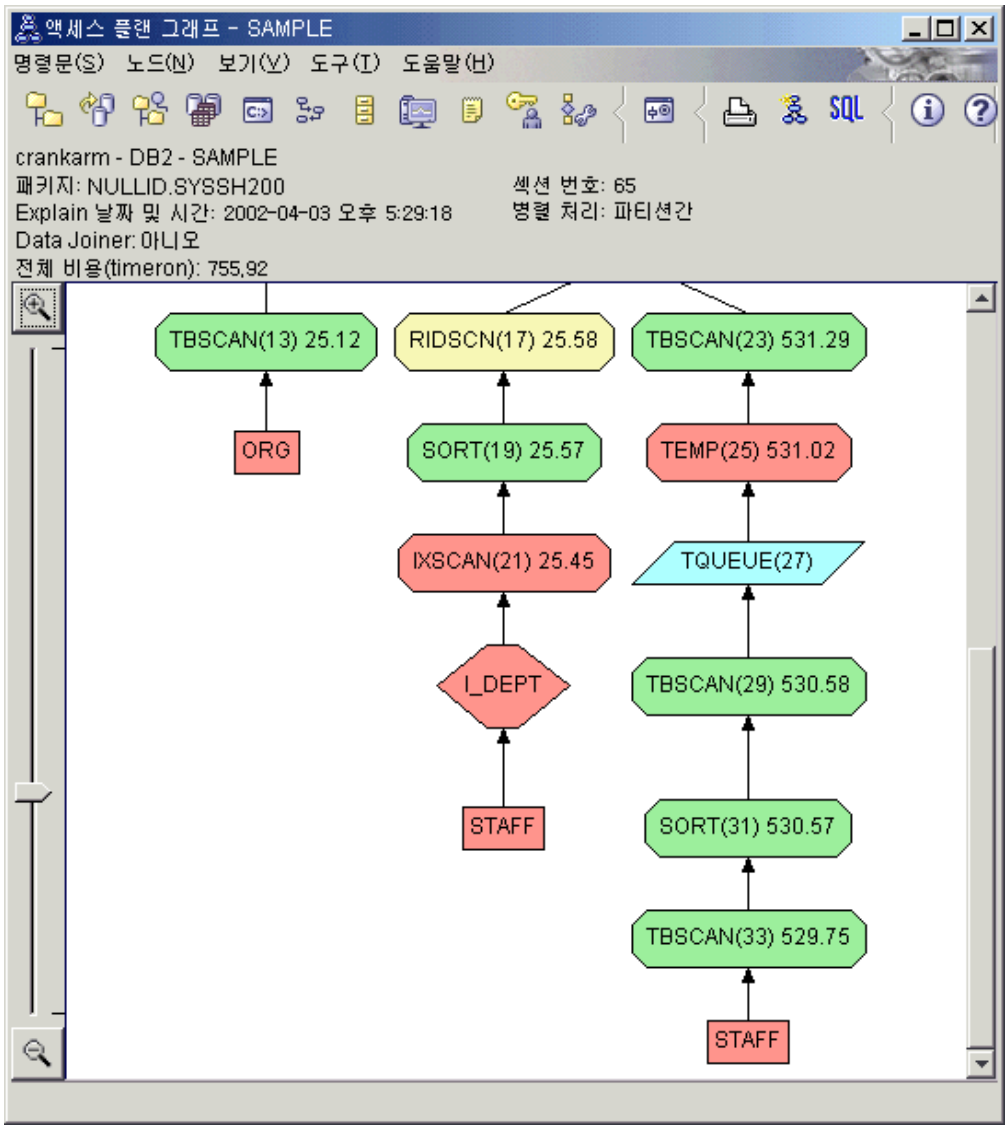
파티션된 데이터베이스 환경에서 쿼리의 테이블을 조인하는 데 사용되는 컬럼에 대한 인덱스 작성

이 예에서는 STAFF 테이블의 DEPT 컬럼 및 ORG 테이블의 DEPTNUMB 컬럼에 대한 인덱스를 작성하여 쿼리 2에서 설명한 액세스 플랜을 빌드합니다.

주: 디자인 어드바이저를 사용하여 권장 인덱스를 작성할 수 있습니다.

이 쿼리(쿼리 3)의 액세스 플랜 그래프를 보려면 Explain문 실행기록 창에서 쿼리 번호 3으로 식별된 엔트리를 더블 클릭하십시오. 이 명령문 실행과 관련된 액세스 플랜 그래프 창이 열립니다.

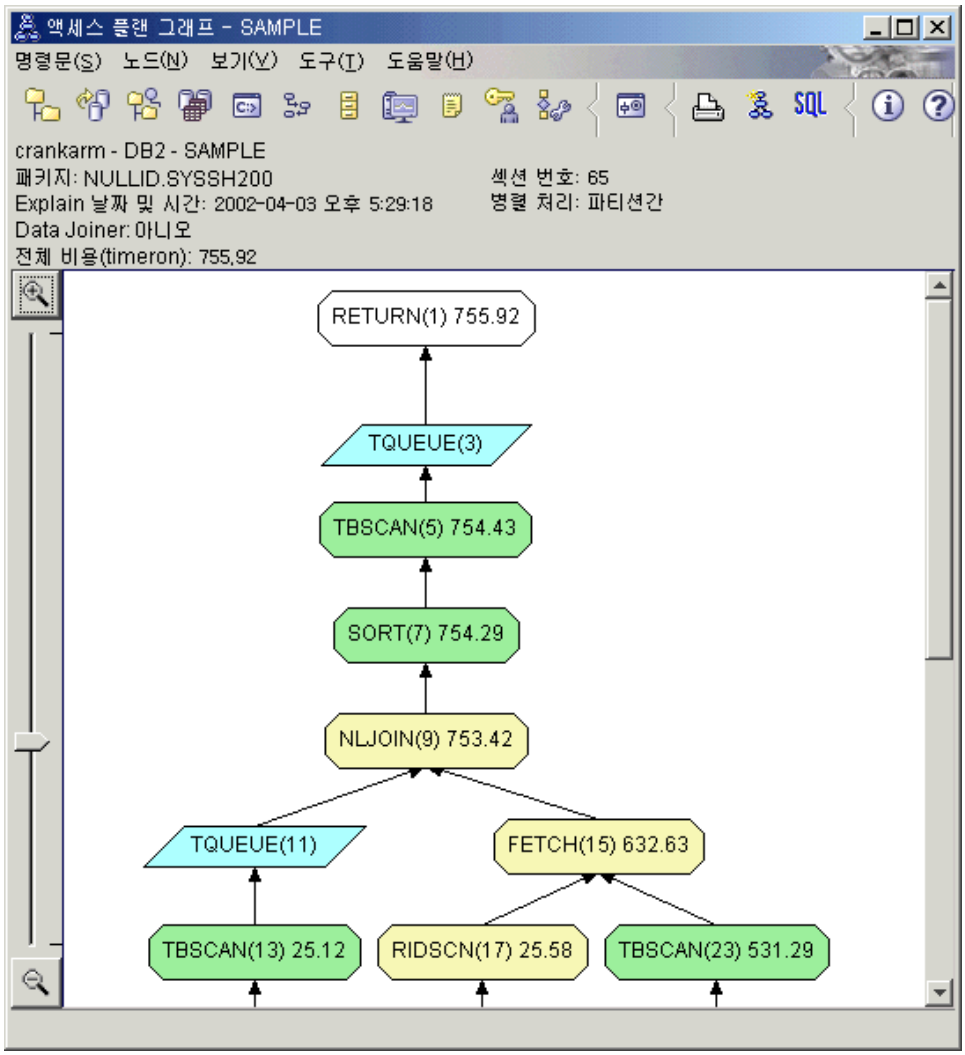
주: DEPTNUM에 관한 인덱스가 작성되었어도 옵티마이저가 이를 사용하지 않았습니다.



다음 질문에 대한 답은 쿼리 개선 방법을 검토하는 데 도움이 됩니다.

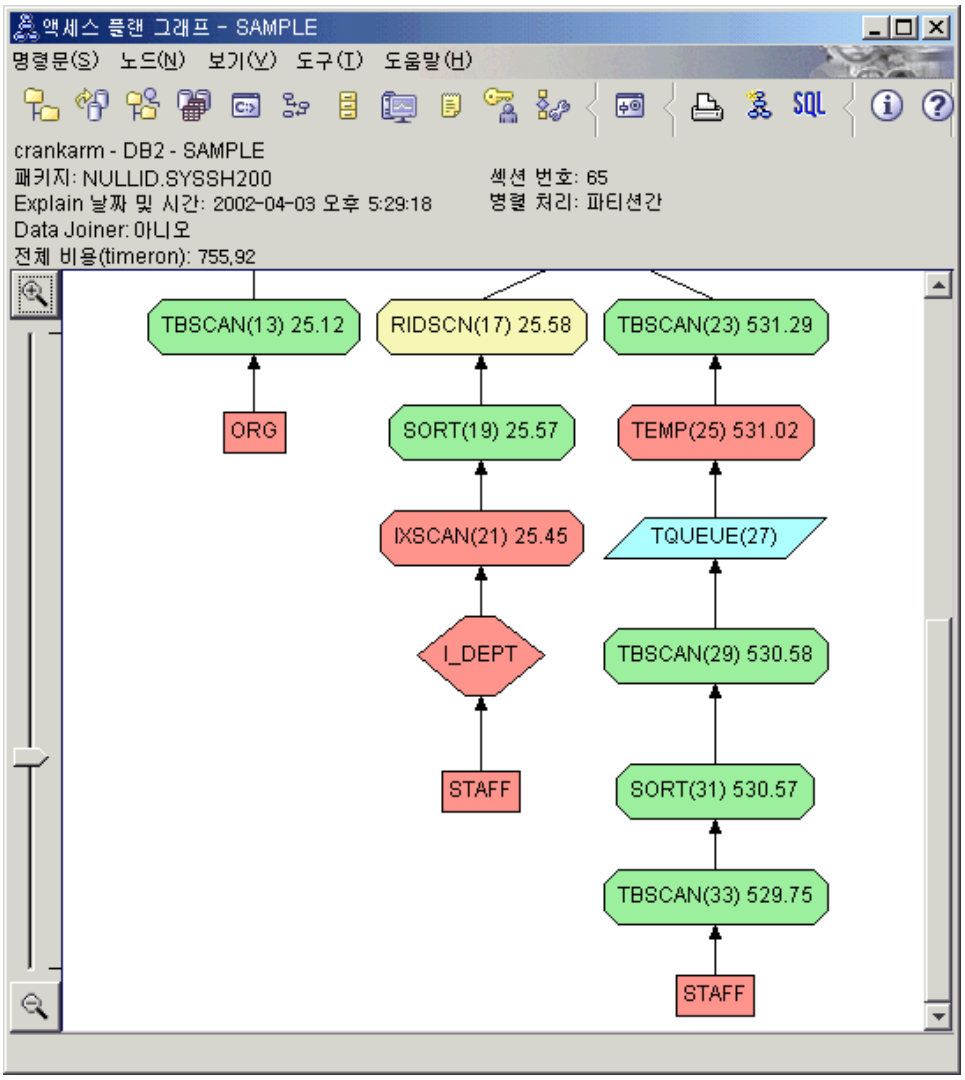
1. 액세스 플랜에서 인덱스에 관해 변경된 사항은 무엇입니까?

   새 다이아몬드형 노드 **I_DEPT**가 STAFF 테이블 바로 위에 추가되었습니다. 이 노드는 DEPT에 대해 작성된 인덱스를 나타내며, 옵티마이저가 테이블 스캔 대신 인덱스 스캔을 사용하여 검색할 행을 판별함을 표시합니다.

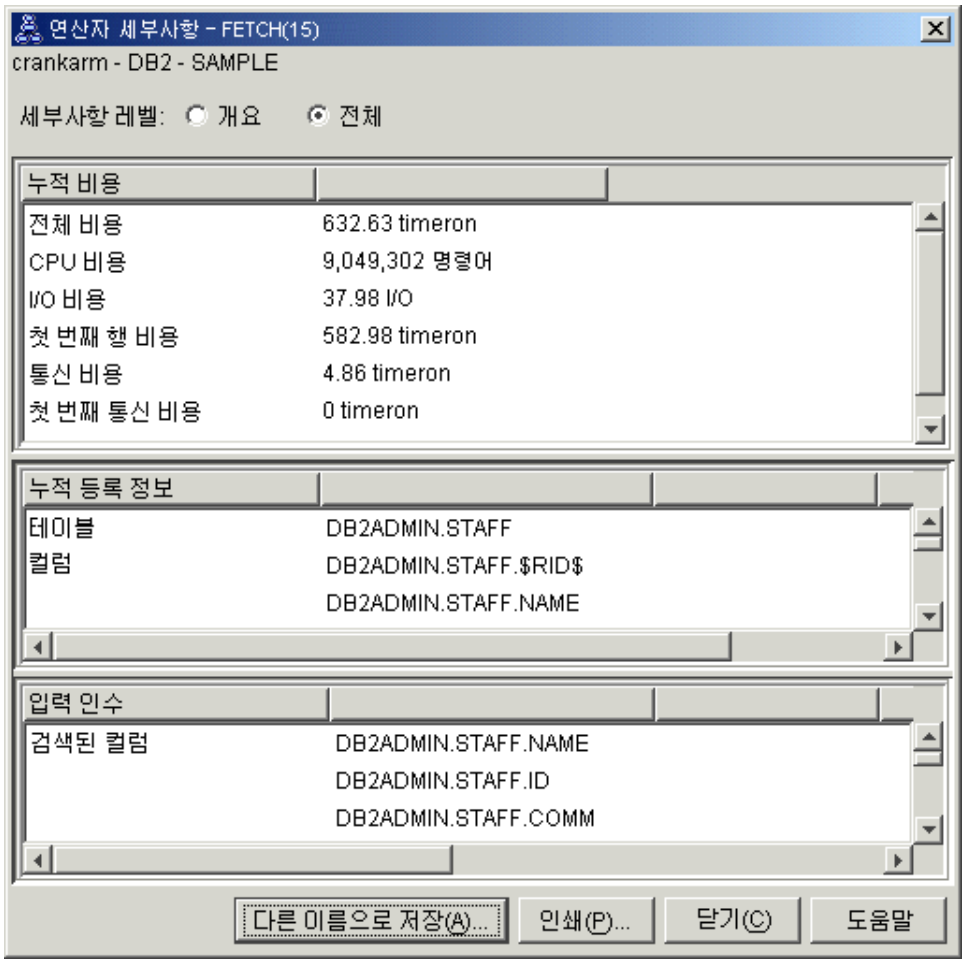


2. 이 액세스 플랜에서 가장 효율적인 방법을 사용하여 데이터에 액세스합니까?

   이 쿼리의 액세스 플랜에서는 ORG 테이블의 DEPTNUMB 컬럼에 대한 인덱스 작성(FETCH (15)와 IXSCAN (21) 발생) 및 STAFF 테이블의 DEPT 컬럼에 대한 인덱스 작성 영향을 표시합니다. 쿼리 2에는 이 인덱스가 없습니다. 따라서 해당 예에서 테이블 스캔이 사용되었습니다.



FETCH (15) 연산자의 연산자 세부사항 창에 이 연산에서 사용 중인 컬럼이 표시됩니다.



이전 액세스 플랜에서 사용된 전체 테이블 스캔보다 비용이 덜 들도록 인덱스와 페치의 조합이 계산됩니다.

3. 이 액세스 플랜의 효율성은 어느 정도입니까?

   이 액세스 플랜은 이전 예에서 사용된 것보다 더 효율적입니다. 누적 비용은 쿼리 2의 대략 1,214 timeron에서 쿼리 3의 755 timeron 가량으로 줄어듭니다.

다음 내용

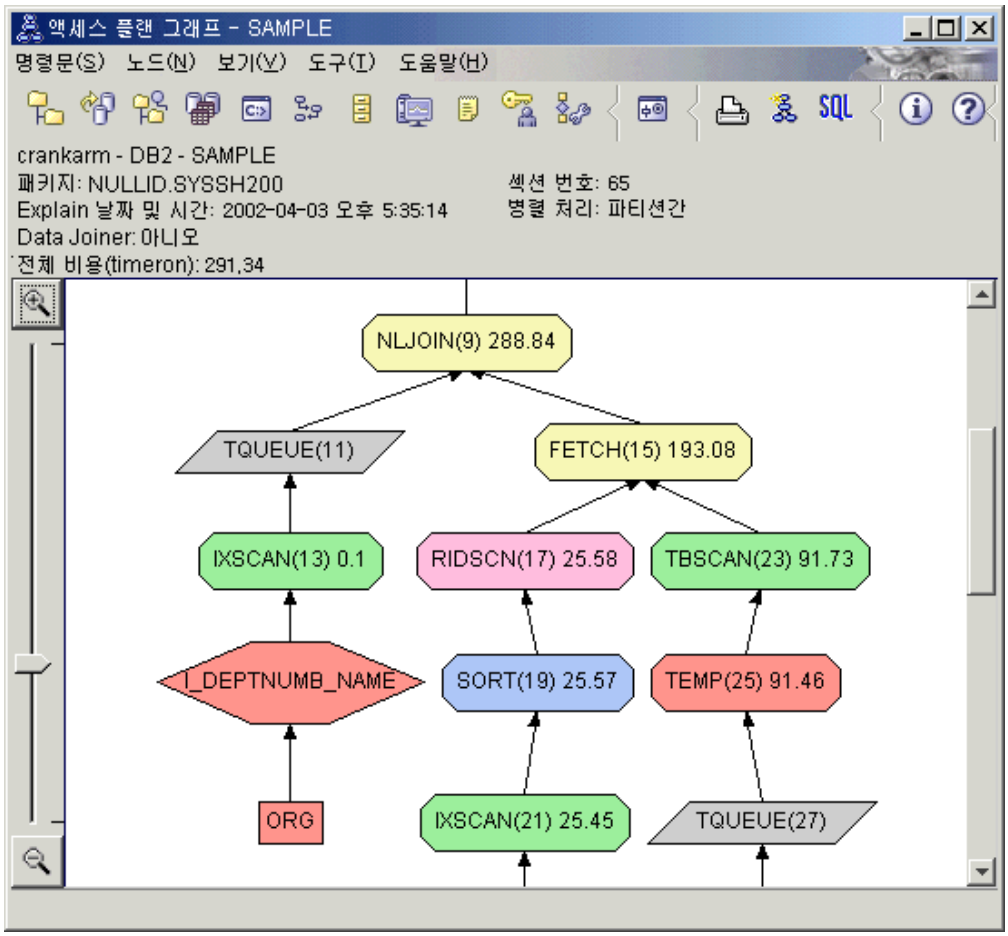
쿼리 4로 이동합니다.

쿼리 4에서는 페치 및 인덱스 스캔이 페치가 없는 단일 인덱스 스캔으로 감소합니다. 추가적인 인덱스를 작성하면 액세스 플랜의 계산된 비용을 줄일 수 있습니다.

파티션된 데이터베이스 환경에서 테이블 컬럼에 대한 추가적인 인덱스 작성

이 예에서는 STAFF 테이블의 JOB 컬럼에 대한 인덱스를 작성하고 ORG 테이블의 기존 인덱스에 DEPTNAME을 추가하여 쿼리 3에서 설명한 액세스 플랜을 빌드합니다. (개별 인덱스를 추가하면 추가적인 액세스가 필요할 수 있습니다.)

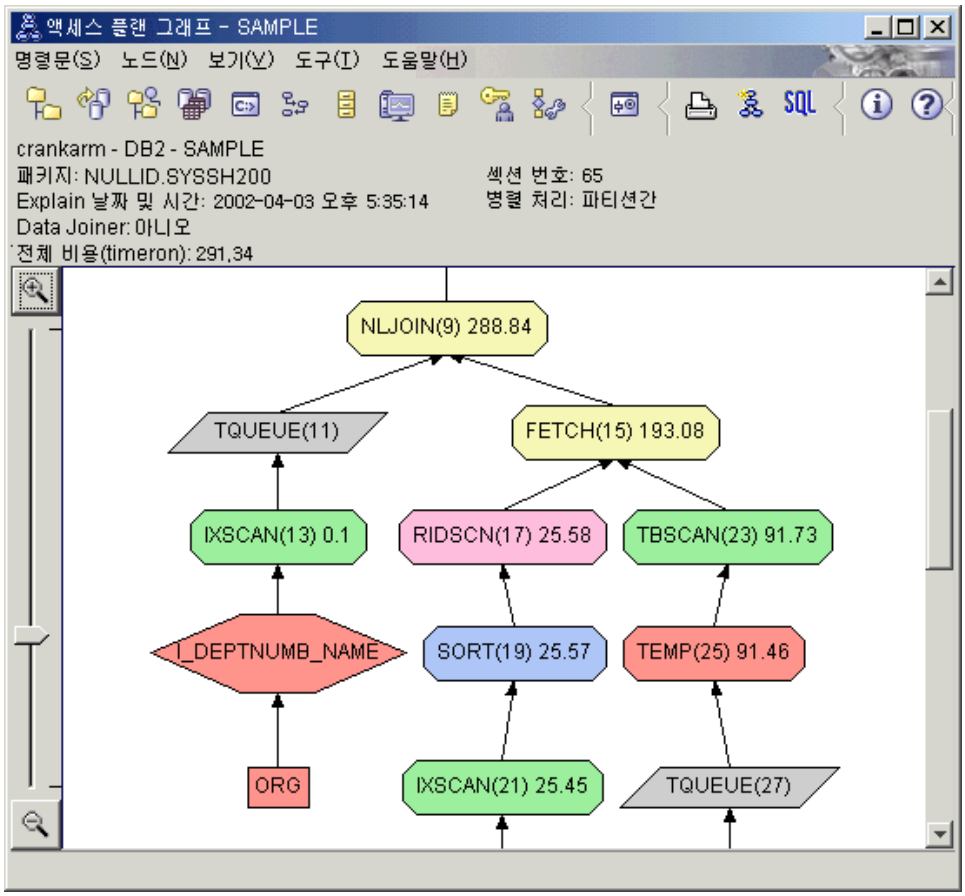
이 쿼리(쿼리 4)의 액세스 플랜 그래프를 보려면 Explain문 실행기록 창에서 쿼리 번호 4으로 식별된 엔트리를 더블 클릭하십시오. 이 명령문 실행과 관련된 액세스 플랜 그래프 창이 열립니다.



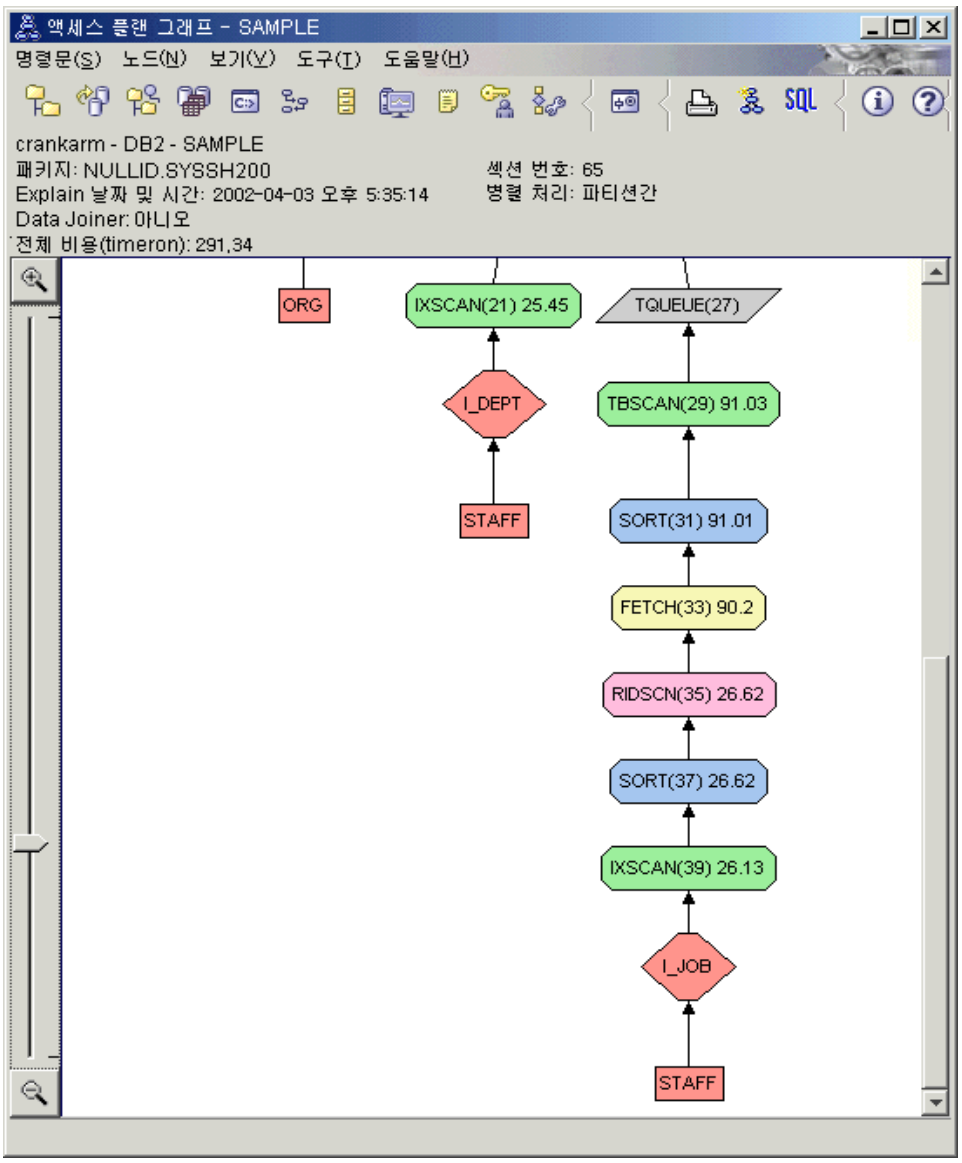
다음 질문에 대한 답은 쿼리 개선 방법을 검토하는 데 도움이 됩니다.

1. 추가적인 인덱스를 작성함으로써 이 프로세스 플랜에서 변경되는 사항은 무엇입니까?

   액세스 플랜 그래프의 중간 분할 영역에서, ORG 테이블과 관련하여 이전 테이블 스캔이 인덱스 스캔 IXSCAN (7)로 변경된 점에 유의하십시오. ORG 테이블의 인덱스에 DEPTNAME 컬럼을 추가함으로써 옵티마이저가 테이블 스캔과 관련된 액세스를 세분화할 수 있도록 했습니다.



액세스 플랜 그래프의 맨 아래 분할 영역에서, STAFF 테이블과 관련하여 이전 인덱스 스캔 및 페치가 인덱스 스캔 전용 IXSCAN (39)로 변경된 점에 유의하십시오. STAFF 테이블의 JOB 인덱스를 작성함으로써 옵티마이저가 페치와 관련된 추가 액세스를 제거할 수 있도록 했습니다.



2. 이 액세스 플랜의 효율성은 어느 정도입니까?

   이 액세스 플랜은 이전 예에서 사용된 것보다 비용이 덜 듭니다. 누적 비용은 쿼리 3의 대략 753 timeron에서 쿼리 4의 288 timeron 가량으로 줄었습니다.

다음 내용

사용자 고유 SQL 또는 XQuery 명령문 성능을 개선합니다.

성능을 개선하기 위해 수행할 수 있는 추가적인 단계에 대한 자세한 정보는 *DB2 정보 센터*를 참조하십시오. 그런 다음 Visual Explain으로 돌아와 조치 결과에 액세스할 수 있습니다.

제 26 장 데이터 재분배

데이터 재분배는 스토리지 스페이스 사용을 밸런싱하고 데이터베이스 시스템 성능 또는 기타 시스템 요구사항을 개선할 수 있도록 파티션이 추가 또는 제거될 때 파티션된 데이터베이스 환경에서 데이터를 주로 이동하기 위해 수행될 수 있는 데이터베이스 관리 연산입니다.

다음 인터페이스 중 하나를 사용하여 데이터 재분배를 수행할 수 있습니다.

- REDISTRIBUTE DATABASE PARTITION GROUP 명령
- ADMIN_CMD 시스템 정의 프로시저
- STEPWISE REDISTRIBUTE_DBPG 시스템 정의 프로시저
- sqludrdt API

파티션된 데이터베이스 내에서 다음 이유 중 하나로 데이터 재분배가 완료됩니다.

- 새 데이터베이스 파티션이 데이터베이스 환경에 추가되거나 기존 데이터베이스 파티션이 제거될 때마다 데이터를 재조정하기 위해.
- 파티션 전반에 사용자 특정의 데이터 분산을 도입하기 위해.
- 특정 파티션 내에서 sensitive 데이터를 분리하여 해당 데이터를 안전하게 하기 위해.

카탈로그 데이터베이스 파티션에서 데이터베이스에 연결하고 지원되는 인터페이스 중 하나를 사용하여 특정 파티션 그룹의 데이터 재분배 연산을 시작하여 데이터 재분배를 수행합니다. 데이터 재분배는 파티션 그룹 내에서 테이블 분산 키 정의의 존재 여부에 달려 있습니다. 테이블 내의 데이터 행에 대한 분산 키 값은 데이터 행이 저장되는 파티션을 판별하는 데 사용됩니다. 분산 키는 다중 파티션 데이터베이스 파티션 그룹에서 테이블이 작성되면 자동으로 생성되거나 CREATE TABLE 또는 ALTER TABLE문을 사용하여 명시적으로 정의할 수 있습니다. 디폴트로 데이터 재분배 중에 지정된 노드 그룹 내의 각 테이블에 대한 테이블 데이터는 데이터베이스 파티션들 사이에서 균등하게 분할 및 재분배됩니다. 그러나 데이터를 분배하는 방법을 정의하는 입력 분산 맵을 지정하여 기타 분배(예: 편중된 분배)를 성취할 수 있습니다. 분산 맵은 장차 사용하기 위해 데이터 재분배 연산 중에 생성하거나 수동으로 작성할 수 있습니다.

데이터 재분배에 대한 제한사항

데이터 재분배에 대한 제한사항은 데이터 재분배를 계속하기 전이나 데이터 재분배에 관련된 문제점을 해결할 때 참고하는 것이 중요합니다.

다음 제한사항은 데이터 재분배에 적용됩니다.

- 테이블에 파티션 키 정의가 없는 파티션의 데이터 재분배가 제한됩니다.
- 데이터 재분배가 진행 중일 때:
  - 데이터베이스 파티션 그룹에서 다른 재분배 연산 시작이 제한됩니다.
  - 데이터베이스 파티션 그룹 삭제가 제한됩니다.
  - 데이터베이스 파티션 그룹 변경이 제한됩니다.
  - 데이터베이스 파티션 그룹에서 임의의 테이블에 대한 ALTER TABLE문 실행이 제한됩니다.
  - 데이터 재분배 중에 테이블에서 인덱스 새로 작성이 제한됩니다.
  - 데이터 재분배 중에 테이블에 정의된 인덱스 삭제가 제한됩니다.
  - 데이터 재분배 중에 테이블에서 데이터 쿼리가 제한됩니다.
  - 데이터 재분배 중에 테이블 갱신이 제한됩니다.
- NOT ROLLFORWARD RECOVERABLE 옵션이 지정된 경우 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 시작된 데이터 재분배 중에 데이터베이스의 테이블 갱신이 제한됩니다. 갱신사항을 작성할 수 있지만 데이터 재분배가 인터럽트되는 경우 데이터에 대한 변경사항이 분실될 수 있으므로 이 실행은 단호히 금지됩니다.
- REDISTRIBUTE DATABASE PARTITION GROUP 명령이 발행되고 NOT ROLLFORWARD RECOVERABLE 옵션이 지정됩니다.
  - 데이터 재분배의 파트로 작성된 데이터 변경사항은 롤 포워드 복구가 불가능합니다.
  - 데이터베이스가 다른 방법으로 복구 가능한 경우, 테이블 스페이스는 파티션 내에서 첫 번째 테이블에 액세스한 후 BACKUP PENDING 상태로 됩니다. 테이블을 이 상태에서 해제하려면 재분배 연산 완료 시 테이블 스페이스 변경사항의 백업을 받아야 합니다.
  - 데이터 재분배 중에 데이터베이스 파티션 그룹의 테이블 데이터를 갱신할 수 없습니다. 데이터는 읽기 전용입니다. 활발하게 재분배 중인 테이블에는 액세스할 수 없습니다.
- 입력된 (계층 구조) 테이블에서, REDISTRIBUTE DATABASE PARTITION GROUP 명령이 사용되고 TABLE 옵션이 ONLY 값에 지정되는 경우, 테이블 이름은 루트 테이블 전용 이름으로 제한됩니다. 서브테이블 이름을 지정할 수 없습니다.

- 범위 파티션된 테이블의 경우, 데이터 파티션된 테이블의 범위 간에 데이터 이동이 제한됩니다. 그러나 데이터베이스 파티션 간에 데이터 이동에 대해서는 데이터 재분배가 지원됩니다.
- 파티션된 테이블의 경우, 다음 두 가지 조건이 모두 참이 아니면 데이터 재분배가 제한됩니다.
  - 파티션된 테이블의 systables.access_mode 카탈로그 테이블에 액세스 모드 FULL ACCESS가 있습니다.
  - 파티션된 테이블에 현재 접속 또는 접속 해제 중인 파티션이 없습니다.
- 복제된 구체화된 쿼리 테이블의 경우 데이터베이스 파티션 그룹의 데이터에 복제된 구체화된 쿼리 테이블이 포함된 경우, 데이터를 재분배하기 전에 이들 테이블을 삭제해야 합니다. 데이터가 재분배된 후, 구체화된 쿼리 테이블을 재작성해야 합니다.
- 다차원 클러스터형(MDC) 테이블을 포함하는 데이터베이스 파티션에 대해서는 정리 보류 중인 롤아웃된 블록을 포함하는 데이터베이스 파티션 그룹에 다차원 클러스터형 테이블이 있는 경우 REDISTRIBUTE DATABASE PARTITION GROUP 명령 사용이 제한되고 정상적으로 진행되지 않습니다. 데이터 재분배를 재개 또는 재시작하기 위해서는 먼저 이 MDC 테이블을 정리해야 합니다.
- DB2 카탈로그 보기에 현재 "재분배 진행 중" 상태에 있는 것으로 표시되어 있는 테이블의 삭제가 제한됩니다. 이 상태에서 테이블을 삭제하려면 먼저 ABORT 또는 CONTINUE 옵션과 적절한 테이블 목록을 사용해 REDISTRIBUTE DATABASE PARTITION GROUP 유틸리티를 실행하여 테이블 재분배가 완료되거나 중단되도록 합니다.

데이터 재분배의 필요성 여부 판별

데이터베이스 파티션 그룹 또는 테이블에 대한 현재 데이터 분산 판별은 데이터 재분배가 필수인지 판별하는 데 도움이 될 수 있고 데이터 분산 방법을 지정하는 데 사용될 수 있는 사용자 정의 분산 맵을 작성하기 위해 사용될 수 있습니다.

새 데이터베이스 파티션이 데이터베이스 파티션 그룹에 추가되거나 기존 데이터베이스 파티션이 데이터베이스 파티션 그룹에서 삭제되는 경우, 모든 데이터베이스 파티션 간에 데이터를 밸런싱하기 위해 데이터 재분배를 수행해야 합니다.

데이터베이스 파티션 그룹에서 추가 또는 삭제된 데이터베이스 파티션이 없는 경우, 데이터 재분배는 보통 데이터베이스 파티션 그룹의 데이터베이스 파티션 간에 불균등한 데이터의 분배가 있을 때만 표시됩니다. 어떤 경우에는 불균등한 데이터 분산이 바람직할 수 있다는 점을 유념하십시오. 예를 들어 일부 데이터베이스 파티션이 특히 강력한 머신에 상주하는 경우, 해당 데이터베이스 파티션에 다른 파티션보다 더 큰 데이터 볼륨을 포함되어 있는 것이 도움이 될 수 있습니다.

데이터베이스 파티션 그룹의 데이터베이스 파티션 간 현재 데이터 분산에 대한 정보를 얻으려면, 데이터베이스 파티션 그룹의 가장 큰 테이블(또는 대표 테이블)에 대해 다음 쿼리를 실행하십시오.

```
SELECT DBPARTITIONNUM(column_name), COUNT(*) FROM table_name
      GROUP BY DBPARTITIONNUM(column_name)
      ORDER BY DBPARTITIONNUM(column_name) DESC
```

여기서 column_name은 table_name 테이블의 분산 키 이름입니다.

쿼리의 결과가 데이터베이스 파티션 간에 데이터 분산이 원하는 대로가 아님을 표시하는 경우, 해시 파티션에 걸쳐 데이터 분산을 얻으려면 다음 쿼리를 실행하십시오.

```
SELECT PARTITION(column_name), COUNT(*) FROM table_name
         GROUP BY PARTITION(column_name)
   ORDER BY PARTITION(column_name) DESC
```

이 쿼리의 출력은 REDISTRIBUTE DATABASE PARTITION GROUP 명령의 USING DISTFILE 옵션이 지정될 때 필요한 분산 파일을 구성하기 위해 용이하게 사용될 수 있습니다(분산 파일 형식에 대한 설명은 REDISTRIBUTE DATABASE PARTITION GROUP 명령에 대한 명령 참조 절을 참조하십시오).

USING DISTFILE 옵션이 지정되면, REDISTRIBUTE DATABASE PARTITION GROUP 명령은 데이터베이스 파티션 간 데이터의 균등 분산으로 결과되는 데이터베이스 파티션 그룹의 새 파티션 맵을 생성하기 위해 파일의 정보를 사용합니다.

균등 분배를 원하지 않는 경우, 사용자는 재분배 연산을 위한 독자적인 목표 파티션 맵을 구성할 수 있고 이 맵은 REDISTRIBUTE DATABASE PARTITION GROUP 명령의 USING TARGETMAP 옵션을 사용하여 지정될 수 있습니다.

이 조사를 수행한 후 데이터가 균등 분배되는지 여부 또는 데이터 재분배가 필수인지 여부를 알 수 있습니다. 데이터 재분배가 필요한 경우, 지원되는 인터페이스 중 하나를 사용하여 시스템 유지보수 기회 중에 이 작업을 수행하기로 계획할 수 있습니다.

REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 데이터베이스 파티션 전반에 데이터 재분배

REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 데이터 재분배를 성공적으로 수행할 수 있습니다. 이것은 데이터 재분배 수행을 위해 권장되는 인터페이스입니다.

제한사항

- 402 페이지의 『데이터 재분배에 대한 제한사항』 참조

REDISTRIBUTE DATABASE PARTITION GROUP 명령을 사용하여 데이터베이스 파티션 그룹의 데이터베이스 파티션 전반에 데이터를 재분배하려면 다음을 수행하십시오.

1. 데이터베이스 백업 수행 BACKUP 명령을 참조하십시오.
2. 시스템 카탈로그 테이블을 포함하는 데이터베이스 파티션에 연결하십시오. CONNECT 명령을 참조하십시오.
3. REDISTRIBUTE DATABASE PARTITION GROUP 명령을 실행합니다.

   주: DB2 제품의 이전 버전에서, 이 명령은 데이터베이스 파티션 그룹 키워드 대신 NODEGROUP 키워드를 사용합니다.

   다음 인수를 지정하십시오.

   데이터베이스 파티션 그룹 이름
   재분배될 데이터 내에 데이터베이스 파티션 그룹을 지정해야 합니다.

   UNIFORM
   선택사항: 데이터가 일정하게 분산되거나 일정하게 분산된 상태를 지속하도록 지정합니다. UNIFORM은 분산 유형이 지정되지 않은 경우 디폴트이므로, 다른 분산 유형이 지정되지 않은 경우 이 옵션을 생략하는 것도 유효합니다.

   USING DISTFILE *distfile-name*
   선택사항: 사용자 정의된 분산을 원하는 경우 원하는 데이터 왜곡을 정의한 데이터를 포함하는 분산 파일의 파일 경로 이름을 지정합니다. 이 파일의 컨텐츠는 목표 분산 맵을 생성하는 데 사용됩니다.

   USING TARGETMAP *targetmap-name*
   선택사항: 목표 데이터 재분배 맵을 사용해야 하는 경우 목표 재분배 맵을 포함하는 파일 이름을 지정합니다.

   세부사항은 REDISTRIBUTE DATABASE PARTITION GROUP 명령행 유틸리티 정보를 참조하십시오.
4. 명령이 인터럽트되지 않고 실행됩니다. 명령이 완료되면 데이터 재분배가 성공적으로 처리된 경우 다음을 수행하십시오.
   - BACKUP PENDING 상태에 있는 데이터베이스 파티션 그룹에 있는 모든 테이블 스페이스를 백업하십시오. 또는 전체 데이터베이스 백업이 수행될 수 있습니다. 참고: 데이터베이스가 복구 가능하고 REDISTRIBUTE DATABASE PARTITION GROUP 명령의 NOT ROLLFORWARD RECOVERABLE 옵션이 사용되는 경우 테이블 스페이스가 BACKUP PENDING 상태로만 됩니다.
   - 재분배 전에 삭제되는 복제된 구체화된 쿼리 테이블을 재작성하십시오.

- REDISTRIBUTE DATABASE PARTITION GROUP 명령의 STATISTICS NONE 옵션이 지정되었거나 NOT ROLLFORWARD RECOVERABLE 옵션이 생략된 경우(두 경우 모두 데이터 재분배 중에 통계가 수집되지 않았음을 의미함) 및 통계 프로파일을 소유하는 데이터베이스 파티션 그룹에 테이블이 있는 경우, 쿼리에 대한 데이터 액세스 플랜을 선택할 때 SQL 컴파일러 및 옵티마이저가 사용할 데이터 분산 통계를 수집하려면 RUNSTATS 명령을 실행하십시오.
- NOT ROLLFORWARD RECOVERABLE 옵션이 지정된 경우, 다음 경로에 위치한 제어 파일을 삭제하십시오.
  - Linux 및 UNIX 운영 체제: DIAGPATH/redist/*db_name*/*db_partitiongroup_name*/*timestamp*/
  - Windows 운영 체제: DIAGPATH₩redist₩*db_name*₩*db_partitiongroup_name*₩*timestamp*₩

데이터 재분배가 성공적으로 완료되었으며 데이터 재분배 프로세스에 대한 정보가 재분배 로그 파일에서 사용 가능합니다. 사용된 재분배 맵에 대한 정보는 DB2 설명 테이블에서 찾을 수 있습니다.

데이터베이스 파티션 그룹에서 데이터 재분배

데이터 재분배 마법사를 사용하여 데이터베이스 파티션에 대한 효과적인 재분배 플랜을 작성하고 데이터를 재분배하십시오. 먼저 재분배 메소드와 전략을 선택한 다음 추가로 고급 선택항목을 선택하십시오.

데이터베이스 파티션 그룹으로 작업하려면 *sysadm* 또는 *dbadm* 권한이 있어야 합니다.

데이터베이스 파티션 그룹에 데이터를 재분배하려면 다음을 수행하십시오.

1. 테이블 데이터 재분배 마법사 열기: 제어 센터에서 오브젝트 트리를 **데이터베이스 파티션 그룹** 폴더를 찾을 때까지 펼치십시오. 기존 데이터베이스 파티션 그룹이 창의 오른쪽 컨텐츠 영역에 표시됩니다. 작업할 데이터베이스 파티션을 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 **재분배**를 선택하십시오. 데이터 재분배 마법사가 열립니다.

   또한 *파티션 추가 런치패드를 사용하여 데이터베이스 파티션 추가* 또는 *파티션 삭제 런치패드를 사용하여 인스턴스에서 데이터베이스 파티션 삭제*에서 데이터 재분배 마법사를 열 수 있습니다.
2. 적용 가능한 각 마법사 페이지를 완료하십시오. 자세한 정보는 첫 번째 페이지의 마법사 개요 링크를 누르십시오. **완료** 누름 단추는 마법사가 데이터를 재분배하기에 충분한 정보를 지정한 경우에 사용 가능합니다.

데이터 재분배에 대한 로그 스페이스 요구사항

데이터 재분배 연산을 성공적으로 수행하기 위해서는 데이터 재분배가 인터럽트되지 않도록 데이터 재분배 연산을 시작하기 전에 먼저 충분한 로그 파일 스페이스를 할당해야 합니다.

필요한 로그 파일 공간의 수량은 사용되는 REDISTRIBUTE DATABASE PARTITION GROUP 명령의 옵션을 포함하여 다중 인수에 의해 결정됩니다.

REDISTRIBUTE DATABASE PARTITION GROUP 명령이 사용되고 NOT ROLLFORWARD RECOVERABLE 옵션이 사용되지 **않는** 경우 또는 데이터 재분배가 롤 포워드 복구 불가능한 어떤 다른 지원되는 인터페이스에서 재분배가 수행되는 경우:

- 로그는 데이터가 재분배될 각 데이터베이스 파티션에서 INSERT 및 DELETE 조작을 실행할 수 있을 만큼 커야 합니다. 가장 과중한 로깅 요구사항은 대부분의 데이터를 유실할 데이터베이스 파티션이나, 대부분의 데이터를 얻을 데이터베이스 파티션에 있습니다.
- 더 많은 데이터베이스 파티션으로 이동할 경우, 현재 데이터베이스 파티션의 비율을 사용하여 INSERT 및 DELETE 조작 횟수를 계산하십시오. 예를 들어, 재분배 전에 일정하게 분산된 데이터의 재분배을 고려하십시오. 네 개의 데이터베이스 파티션에서 5개의 데이터베이스 파티션으로 이동할 경우, 네 개의 원래 데이터베이스 파티션에서 약 20 퍼센트가 새 데이터베이스 파티션으로 이동됩니다. 이것은 네 개의 원래 데이터베이스 파티션 각각에서 20 퍼센트의 DELETE 조작이 발생하고 INSERT 조작 모두가 새 데이터베이스 파티션에서 발생함을 의미합니다.
- 분산 키에 많은 널(NULL) 값이 들어 있는 경우와 같이, 일정하지 않은 데이터 분산을 고려하십시오. 이 경우, 분산 키의 NULL 값을 포함하는 모든 행은 이전 분산 스킴 아래의 하나의 데이터베이스 파티션에서, 그리고 새 분산 스킴 아래의 다른 데이터베이스 파티션으로 이동됩니다. 결과적으로, 두 개의 데이터베이스 파티션에서 필요한 로그 스페이스의 양이 증가하고 일정한 분산을 가정하여 계산되는 양을 초과하게 됩니다.
- 각 테이블의 재분배는 단일 트랜잭션입니다. 이러한 이유로, 로그 스페이스를 계산할 때 변경 백분율(예: 20 퍼센트)에 가장 큰 테이블의 *크기*를 곱합니다. 그러나 가장 큰 테이블은 일정하게 분산되지만, 두 번째로 큰 테이블(예: 현저하게 팽창된 하나 이상의 데이터베이스 파티션)이 있을 수도 있습니다. 이런 경우, 가장 큰 테이블 대신 일정하지 않게 분산된 테이블을 사용할 것을 고려하십시오.

주: 데이터베이스 파티션에서 삽입되고 삭제될 최대 데이터의 양을 계산한 후, 사용 중인 로그의 최대 *크기*를 판별하십시오. 이 계산이 사용 중인 로그 한계인 1024GB를 초과할 경우, 데이터 재분배는 단계별로 완료해야 합니다. ″makepmap″ 유틸리티를 사용

하여 각 단계마다 하나씩, 일련의 목표 분산 맵을 생성하십시오. 또한 logsecond 데이터베이스 구성 매개변수를 -1로 설정하여 대부분의 로그 스페이스 문제점을 피할 수도 있습니다.

REDISTRIBUTE DATABASE PARTITION GROUP 명령이 사용되고 NOT ROLLFORWARD RECOVERABLE 옵션이 사용되는 경우 또는 데이터 재분배가 롤 포워드 복구 불가능한 어떤 다른 지원되는 인터페이스에서 재분배가 수행되는 경우:

- 데이터 재분배의 파트로 행이 이동될 때는 로그 레코드가 작성되지 않습니다. 이는 로그 파일 스페이스 요구사항을 크게 감소시키지만 이 옵션이 데이터베이스의 롤 포워드 복구가 수행될 때 사용되면 재분배 연산 로그 레코드가 롤 포워드될 수 없고 롤 포워드 연산의 파트로 처리된 모든 테이블이 UNAVAILABLE 상태로 남아 있습니다. NOT ROLLFORWARD RECOVERABLE 옵션 사용 결과에 대한 논의는 명령 참조를 참조하십시오.
- 데이터 재분배 중인 데이터베이스 파티션 그룹이 테이블에 긴 필드(LF) 또는 대형 오브젝트(LOB) 데이터가 있는 테이블을 포함하는 경우, 각 데이터 행에 대해 로그 레코드가 작성되므로 데이터 재분배 중에 생성된 로그 레코드 수가 더 높아집니다. 이 경우, 데이터베이스 파티션당 로그 스페이스 요구사항이 해당 파티션에서 이동하는 데이터(즉, 송신 및 수신되는 데이터) 양의 약 1/3이라고 예상하십시오. LF/LOB 데이터의 존재 여부에 관계없이 수신 파티션에는 단일 유형의 로그 레코드가 작성되며, 그 결과 이러한 로그 레코드 수는 이동 중인 데이터 양: Extent 할당 로그 레코드에 따라 결정됩니다. 그러나 이 로그 레코드에 필요한 전체 공간이 작고 이동 중인 전체 사용자 데이터 중 극소량에 불과합니다.

이벤트 로그 파일 재분배

데이터 재분배 중에 이벤트 로깅이 수행됩니다. 이벤트 정보는 오류 복구를 수행하기 위해 나중에 사용될 수 있는 이벤트 로그 파일에 로깅됩니다.

데이터 재분배가 수행되면 처리되는 각 테이블에 대한 정보가 단일 재분배 이벤트 로그 파일에 로깅됩니다.

이벤트 로그 파일 이름은 `database-name.database-partition-group-name.timestamp.log`와 같이 형식화됩니다. 로그 파일은 다음에 위치합니다.

- Linux® 및 UNIX® 기반 시스템의 `homeinst/sqllib/redist` 디렉토리.
- Windows® 운영 체제의 `DB2INSTPROF¥instance¥redist` 디렉토리. 여기서 DB2INSTPROF는 DB2INSTPROF 레지스트리 변수 값입니다.

다음은 이벤트 로그 파일 이름의 예입니다.

```
DB819.NG1.2007062419415651.log
```

이 이벤트 로그 파일은 2007년 6월 24일 현지 시간 오후 4시 41분에 작성된 NG1이라는 데이터베이스 파티션 그룹이 있는 DB819라는 데이터베이스에서 재분배 연산용입니다.

이벤트 로그 파일의 세 가지 기본 사용은 다음과 같습니다.

- 재분배 연산에 대한 일반 정보(예: 이전 및 새 분산 맵)를 제공합니다.
- 유틸리티에서 지금까지 재분배된 테이블을 추적하는 데 도움이 되는 정보를 사용자에게 제공합니다.
- 테이블에 사용되는 인덱싱 모드, 테이블이 성공적으로 재분배되었는지 여부 표시 및 테이블에서 재분배 연산 시작 및 종료 시간을 포함하여 재분배된 각 테이블에 대한 정보를 제공합니다.

재분배 로그 파일 항목 및 데이터 재분배 중에 오류 복구 방법에 대한 자세한 정보는 다음을 참조하십시오.

STEPWISE_REDISTRIBUTE_DBPG 프로시저를 사용하여 데이터베이스 파티션 그룹 재분배

STEPWISE_REDISTRIBUTE_DBPG 시스템 정의 프로시저를 사용하여 데이터 재분배를 수행할 수 있습니다.

STEPWISE_REDISTRIBUTE_DBPG 시스템 정의 프로시저 및 기타 시스템 정의 프로시저를 사용하여 데이터베이스 파티션 그룹 재분배를 수행할 수 있습니다.

다음 단계에서는 수행해야 할 작업에 대해 설명하고 이 단계를 설명하는 예제는 다음과 같습니다.

1. *ANALYZE_LOG_SPACE 프로시저 - 로그 스페이스 분석 정보 검색*을 사용하여 로그 스페이스 사용 가능성 및 데이터 왜곡과 관련하여 데이터베이스 파티션 그룹을 분석하십시오.

   analyze_log_space 함수는 로그 스페이스 분석 결과의 결과 세트(열린 커서)를 리턴하며 여기에는 제공된 데이터베이스 파티션 그룹의 각 데이터베이스 파티션에 대한 필드가 포함됩니다.
2. *GENERATE_DISTFILE 프로시저 - 데이터 분산 파일 생성*을 사용하여 주어진 테이블의 데이터 분산 파일을 작성하십시오.

   generate_distfile 기능은 지정된 테이블에 대해 데이터 분산 파일을 생성하고 제공된 파일 이름을 사용하여 해당 파일을 저장합니다.
3. *STEPWISE_REDISTRIBUTE_DBPG 프로시저 - 데이터베이스 파티션 그룹의 파트 재분배*를 사용하여 데이터베이스 파티션 그룹의 단계별 재분배 플랜의 내용을 작성하고 보고하십시오.

4. *GET_SWRD_SETTINGS 프로시저 - 재분배 정보 검색* 및 *SET_SWRD_SETTINGS 프로시저 - 재분배 레지스트리 작성 또는 변경*을 사용하여 주어진 테이블의 데이터 분산 파일을 작성하십시오.

   get_swrd_settings 기능은 주어진 데이터베이스 파티션 그룹의 기존 재분배 레지스트리 레코드를 읽습니다.

   set_swrd_settings 기능은 재분배 레지스트리를 작성하거나 변경합니다. 레지스트리가 존재하지 않을 경우 레지스트리를 새로 작성하여 레코드를 추가합니다. 레지스트리가 이미 있는 경우 overwriteSpec을 사용하여 겹쳐써야 하는 필드 값을 식별합니다. overwriteSpec 필드는 이 기능을 사용하여 갱신할 필요가 없는 필드에 대한 널(NULL) 입력을 취합니다.

5. *STEPWISE_REDISTRIBUTE_DBPG 프로시저 - 데이터베이스 파티션 그룹의 일부 재분배*를 사용하여 플랜에 따라 데이터베이스 파티션 그룹을 재분배하십시오.

   stepwise_redistribute_dbpg 기능은 입력 및 설정 파일에 따라 데이터베이스 파티션 그룹의 일부분을 재분배합니다.

사용 예

다음은 AIX에서 CLP 스크립트의 예입니다.

```
# ------------------------------------------------------------------------------------
# Set the database you wish to connect to
# ------------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------------
# Set the target database partition group name
# ------------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------------
# Specify the table name and schema
# ------------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------------
# Specify the name of the data distribution file
# ------------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
# Invoke call statements in clp
# ------------------------------------------------------------------------------------
db2start
db2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analysing the effect of adding a database partition without applying the changes - a 'what if'
# hypothetical analysis
#
# - In the following case, the hypothesis is adding database partition 40, 50 and 60 to the
#   database partition group, and for database partitions 10,20,30,40,50,60, using a respective
#   target ratio of 1:2:1:2:1:2.
#
# NOTE: in this example only partitions 10, 20 and 30 actually exist in the database
```

```
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ------------------------------------------------------------------------------------
# Analysing the effect of dropping a database partition without applying the changes
#
# - In the following case, the hypothesis is dropping database partition 30 from the database
#   partition group, and redistributing the data in database partitions 10 and 20 using a
#   respective target ratio of 1 : 1
#
# NOTE: In this example all database partitions 10, 20 and 30 should exist in the database
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------------
# Generate a data distribution file to be used by the redistribute process
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------------
# Write a step wise redistribution plan into a registry
#
# Setting the 10th parameter to 1, may cause a currently running step wise redistribute
# stored procedure to complete the current step and stop, until this parameter is reset
# to 0, and the redistribute stored procedure is called again.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------------
# Report the content of the step wise redistribution plan for the given database
# partition group.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------------
# Redistribute the database partition group "dbpgName" according to the redistribution
# plan stored in the registry by set_swrd_settings. It starting with step 3 and
# redistributes the data until 2 steps in the redistribution plan are completed.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

제 27 장 자체 성능 조정 메모리 구성

파티션된 데이터베이스 환경에서 자체 성능 조정 메모리

파티션된 데이터베이스 환경에서 자체 성능 조정 메모리 기능을 사용할 경우, 이 기능으로 시스템을 적절히 조정할지 여부를 판별하는 몇 가지 인수가 있습니다.

파티션된 데이터베이스에 대해 자체 성능 조정 메모리가 사용될 경우, 단일 데이터베이스 파티션이 조정 파티션으로 지정되고 모든 메모리 조정 결정은 해당 데이터베이스 파티션의 메모리 및 워크로드 특성을 기반으로 합니다. 해당 파티션에 대한 조정이 결정된 후, 다른 데이터베이스 파티션에 메모리 조정이 분산되어 모든 데이터베이스 파티션이 유사한 구성을 유지하도록 해 줍니다.

단일 조정 파티션 모델에서는 모든 데이터베이스 파티션이 유사한 메모리 요구사항을 지니고 있을 경우에만 기능이 사용된다고 가정합니다. 파티션된 데이터베이스에서 자체 성능 조정 메모리를 사용할지 여부를 결정할 때 다음 지침을 사용하십시오.

파티션된 데이터베이스에 대한 자체 성능 조정 메모리가 권장되는 경우

모든 데이터베이스 파티션이 유사한 메모리 요구사항을 지니고 유사한 하드웨어에서 실행되고 있는 경우, 자체 성능 조정 메모리를 수정 없이 사용할 수 있습니다. 이러한 유형의 환경은 다음 특성을 공유합니다.

- 모든 데이터베이스 파티션이 동일한 하드웨어에 있고, 여러 논리적 데이터베이스 파티션이 여러 물리적 데이터베이스 파티션에 고르게 분산되어 있습니다.
- 완벽하거나 완벽에 가깝게 데이터가 분산되어 있습니다.
- 워크로드가 데이터베이스 파티션에 걸쳐 고르게 분산되어 있습니다. 즉, 다른 데이터베이스 파티션보다 하나 이상의 힙에 대해 더 높은 메모리 요구사항을 지닌 데이터베이스 파티션이 없습니다.

이러한 환경에서 모든 데이터베이스 파티션이 균등하게 구성되어 있는 경우, 자체 성능 조정 메모리가 시스템을 적절히 구성합니다.

파티션된 데이터베이스에 대한 자체 성능 조정 메모리가 조건부로 권장되는 경우

환경에서 대부분의 데이터베이스 파티션이 유사한 메모리 요구사항을 지니고 유사한 하드웨어에서 실행되고 있는 경우, 초기 구성에서 약간의 주의를 기울이기만 하면 자체 성능 조정 메모리를 사용할 수 있습니다. 이러한 시스템에는 데이터에 대한 하나의 데이터베이스 파티션 세트와 훨씬 더 작은 코디네이터 파티션 및 카탈로그 파티션 세트가

있을 수 있습니다. 이런 환경에서는 코디네이터 파티션과 카탈로그 파티션을 데이터가 포함된 데이터베이스 파티션과 다르게 구성하는 것이 좋습니다.

데이터가 포함된 모든 데이터베이스 파티션에서 자체 성능 조정 메모리가 사용되어야 하고, 이러한 데이터베이스 파티션 중 하나가 조정 파티션으로 지정되어야 합니다. 그리고 코디네이터 파티션과 카탈로그 파티션이 다르게 구성될 수 있기 때문에 해당 파티션에서 자체 성능 조정 메모리를 사용하지 않아야 합니다. 코디네이터 파티션과 카탈로그 파티션에서 자체 성능 조정 메모리를 사용하지 않으려면 이러한 파티션에서 **self_tuning_mem** 데이터베이스 구성 매개변수를 해제(OFF)로 설정하십시오.

파티션된 데이터베이스에 대한 자체 성능 조정 메모리가 권장되지 않는 경우

각 데이터베이스 파티션의 메모리 요구사항이 다른 경우, 또는 크게 다른 하드웨어에서 다른 데이터베이스 파티션이 실행되고 있는 경우 자체 성능 조정 메모리 기능을 사용하지 않는 것이 좋습니다. 모든 파티션에서 **self_tuning_mem** 데이터베이스 구성 매개변수를 해제(OFF)로 설정하여 이 기능을 사용하지 않을 수 있습니다.

다른 데이터베이스 파티션의 메모리 요구사항 비교

다른 데이터베이스 파티션의 메모리 요구사항이 충분히 유사한지 여부를 판별하는 가장 좋은 방법은 스냅샷 모니터를 참조하는 것입니다. 다음 스냅샷 요소가 모든 데이터베이스 파티션에서 유사할 경우(20% 이하 차이), 데이터베이스 파티션이 메모리 요구사항이 충분히 유사한 것으로 고려될 수 있습니다.

`get snapshot for database on <dbname>` 명령을 발행하여 다음 데이터를 수집하십시오.

```
현재 보유된 잠금 수                          = 0
잠금 대기 수                                 = 0
데이터베이스의 잠금 대기 시간(밀리초)       = 0
사용 중인 잠금 목록 메모리(바이트)           = 4968
잠금 에스컬레이션 수                         = 0
배타적 잠금 에스컬레이션 수                  = 0

할당된 총 공유 정렬 힙                       = 0
공유 정렬 힙 상위 워터 마크                  = 0
포스트 임계값 정렬(공유 메모리)              = 0
정렬 오버플로우 수                           = 0

패키지 캐시 찾아보기 수                      = 13
패키지 캐시 삽입 수                          = 1
패키지 캐시 오버플로우 수                    = 0
패키지 캐시 상위 워터 마크 수(Bytes)         = 655360

해시 조인 수                                 = 0
해시 루프 수                                 = 0
해시 조인 오버플로우 수                      = 0
작은 해시 조인 오버플로우 수                 = 0
포스트 임계값 해시 조인 수(공유 메모리)      = 0
```

```
OLAP 기능 수                                = 0
OLAP 기능 오버플로우 수                     = 0
활성 OLAP 기능                              = 0
```

get snapshot for bufferpools on <dbname> 명령을 발행하여 다음 데이터를 수집하십시오.

```
버퍼 풀 데이터 논리적 읽기 수              = 0
버퍼 풀 데이터 실제 읽기 수                = 0
버퍼 풀 인덱스 논리적 읽기 수              = 0
버퍼 풀 인덱스 실제 읽기 수                = 0
전체 버퍼 풀 읽기 시간(밀리초)             = 0
전체 버퍼 풀 쓰기 시간(밀리초)             = 0
```

파티션된 데이터베이스 환경에서 자체 성능 조정 메모리 사용

파티션된 데이터베이스 환경에서 자체 성능 조정 메모리를 사용할 경우, 메모리 구성을 모니터하고 구성 변경사항을 다른 모든 데이터베이스 파티션에 전파하여 모든 참여 데이터베이스 파티션 간에 일관성 있는 구성을 유지하는 단일 데이터베이스 파티션(*조정 파티션*이라고 함)이 있습니다.

조정 파티션은 버퍼 풀의 수 및 파티션 그룹에서 데이터베이스 파티션의 수와 같은 여러 특성을 기반으로 선택됩니다.

- 현재 어떤 데이터베이스 파티션이 조정 파티션으로 지정되어 있는지 판별하려면 ADMIN_CMD 프로시저를 다음과 같이 호출하십시오.

  ```
  CALL SYSPROC.ADMIN_CMD('get stmm tuning dbpartitionnum')
  ```

- 조정 파티션을 변경하려면 ADMIN_CMD 프로시저를 다음과 같이 호출하십시오.

  ```
  CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum <partitionnum>')
  ```

  조정 파티션은 비동기적으로 또는 다음 데이터베이스 시작 시 갱신됩니다. 메모리 조정 프로그램에서 조정 파티션을 자동으로 선택하도록 하려면 *partitionnum* 값으로 -1을 입력하십시오.

파티션된 데이터베이스 환경에서 메모리 조정 프로그램 시작

파티션된 데이터베이스 환경에서는, 자체 성능 조정 메모리를 사용하려면 모든 파티션이 활성이어야 하므로, 명시적 ACTIVATE DATABASE 명령에 의해 데이터베이스가 활성화된 경우에만 메모리 조정 프로그램이 시작됩니다.

특정 데이터베이스 파티션에 대해 자체 성능 조정 메모리 사용 안함

- 데이터베이스 파티션의 서브세트에 대해 자체 성능 조정 메모리를 사용하지 않으려면 해당 데이터베이스 파티션에 대해 **self_tuning_mem** 데이터베이스 구성 매개변수를 해제(OFF)로 설정하십시오.

- 특정 데이터베이스 파티션의 구성 매개변수에 의해 제어되는 메모리 소비자의 서브세트에 대해 자체 성능 조정 메모리를 사용하지 않으려면 관련 구성 매개변수의 값 또는 버퍼 풀 *크기*를 MANUAL 또는 해당 데이터베이스 파티션에 대한 특정 값으로 설정하십시오. 실행되는 모든 파티션에서 자체 성능 조정 메모리 구성 매개변수 값이 일관성이 있는 것이 좋습니다.
- 특정 데이터베이스 파티션의 특정 버퍼 풀에 대해 자체 성능 조정 메모리를 사용하지 않으려면 자체 성능 조정 메모리가 사용되지 않을 파티션과 *크기* 값을 지정하는 ALTER BUFFERPOOL문을 발행하십시오.

  특정 데이터베이스 파티션의 버퍼 풀의 *크기*를 지정하는 ALTER BUFFERPOOL문은 SYSCAT.BUFFERPOOLDBPARTITIONS 카탈로그 뷰에서 해당 버퍼 풀에 대한 예외 항목을 작성합니다(또는 기존 항목 갱신). 버퍼 풀에 대한 예외 항목이 존재하는 경우, 디폴트 버퍼 풀 *크기*가 AUTOMATIC으로 설정되어 있으면 해당 버퍼 풀이 자체 성능 조정 조작에 참여하지 않습니다. 버퍼 풀이 자체 성능 조정에 사용될 수 있도록 예외 항목을 제거하려면 다음을 수행하십시오.

  1. ALTER BUFFERPOOL문을 발행하고 버퍼 풀 *크기*를 특정 값으로 설정하여 이 버퍼 풀에 대해 자체 성능 조정을 사용하지 마십시오.
  2. 다른 ALTER BUFFERPOOL문을 발행하여 이 데이터베이스 파티션의 버퍼 풀의 *크기*를 디폴트로 설정하십시오.
  3. 다른 ALTER BUFFERPOOL문을 발행하고 버퍼 풀 *크기*를 AUTOMATIC으로 설정하여 이 버퍼 풀에 대해 자체 성능 조정을 사용하십시오.

비균일 환경에서 자체 성능 조정 메모리 사용

원칙적으로는, 데이터가 모든 데이터베이스 파티션 간에 고르게 분산되어야 하고, 각 파티션에서 실행되는 워크로드에 유사한 메모리 요구사항이 있어야 합니다. 데이터 분산이 비대칭으로 되어 하나 이상의 데이터베이스 파티션에 다른 데이터베이스 파티션보다 상당히 많거나 적은 데이터가 포함될 경우, 이러한 이례적 데이터베이스 파티션이 자체 성능 조정에 사용되지 않아야 합니다. 데이터베이스 파티션 간에 메모리 요구사항이 비대칭인 경우에도 마찬가지입니다. 이러한 경우는 예를 들어, 자원 중심 정렬이 한 파티션에서만 수행될 경우나 일부 데이터베이스 파티션이 다른 데이터베이스 파티션보다 더 많은 메모리 및 다른 하드웨어와 연관된 경우에 발생할 수 있습니다. 이런 유형의 환경에서 일부 데이터베이스 파티션에 자체 성능 조정 메모리가 여전히 사용될 수 있습니다. 비대칭 환경에서 자체 성능 조정 메모리를 이용하려면 유사한 데이터 및 메모리 요구사항을 지닌 데이터베이스 파티션 세트를 식별하고 자체 성능 조정에 사용하십시오. 나머지 파티션의 메모리는 수동으로 구성해야 합니다.

제 28 장 DB2 구성 매개변수 및 변수

다중 파티션에서 데이터베이스 구성

데이터베이스 관리 프로그램은 다중 파티션의 모든 데이터베이스 구성 요소를 볼 수 있는 단일 뷰를 제공합니다. 이는 각 데이터베이스 파티션에 대해 db2_all 명령을 호출하지 않아도 모든 데이터베이스 파티션에서 데이터베이스 구성을 갱신하거나 재설정할 수 있음을 의미합니다.

데이터베이스가 상주하는 파티션에서 하나의 관리 명령 또는 하나의 SQL문을 실행하여 파티션에서 구성을 갱신할 수 있습니다. 디폴트로, 데이터베이스 구성 갱신 또는 재설정 방법은 *모든 데이터베이스 파티션*에 있습니다.

명령 스크립트 및 응용프로그램의 이전 버전과의 호환을 위해 세 가지 옵션을 사용할 수 있습니다.

- 다음과 같이 db2set 명령을 사용하여 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 레지스트리 변수를 TRUE로 설정하십시오.

  ```
  DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
  ```

  주: 레지스트리 변수 설정은 ADMIN_CMD 프로시저를 사용하여 작성한 UPDATE DATABASE CONFIGURATION 또는 RESET DATABASE CONFIGURATION 요청에는 적용되지 않습니다.

- **DBPARTITIONNUM** 매개변수를 UPDATE DATABASE CONFIGURATION 또는 RESET DATABASE CONFIGURATION 명령이나 ADMIN_CMD 프로시저와 함께 사용하십시오. 예를 들어, 모든 데이터베이스 파티션에서 데이터베이스 구성을 갱신하려면 다음과 같이 ADMIN_CMD 프로시저를 호출하십시오.

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG USING sortheap 1000')
  ```

  단일 데이터베이스 파티션을 갱신하려면 다음과 같이 ADMIN_CMD 프로시저를 호출하십시오.

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```

- **DBPARTITIONNUM** 매개변수를 db2CfgSet API와 함께 사용하십시오. **db2Cfg** 구조의 플래그는 데이터베이스 구성에 필요한 값을 단일 데이터베이스 파티션에 적용할지 여부를 나타냅니다. 플래그를 설정하는 경우 다음과 같이 **DBPARTITIONNUM** 값도 제공해야 합니다.

  ```
  #define db2CfgSingleDbpartition          256
  ```

db2CfgSingleDbpartition 값을 설정하지 않으면
DB2_UPDDBCFG_SINGLE_DBPARTITION 레지스트리 변수를 TRUE로 설정하거나 *versionNumber*를 버전 9.5보다 낮은 버전 번호로 설정하지 않는 한, 데이터베이스 관리 프로그램이나 데이터베이스 구성 매개변수를 설정하는 db2CfgSet API와 관련하여 데이터베이스 구성에 필요한 값이 모든 데이터베이스 파티션에 적용됩니다.

데이터베이스를 버전 9.7로 업그레이드하면 기존 데이터베이스 구성 매개변수가, 일반적인 규칙으로는 데이터베이스 업그레이드 이후에도 해당 값을 보유합니다. 그러나 디폴트값 및 일부 기존 매개변수를 사용하여 추가되는 새 매개변수는 새 버전 9.7 디폴트값으로 설정됩니다. 기존 데이터베이스 구성 매개변수의 변경에 대한 자세한 내용은 *DB2 버전 9.7로 업그레이드*의 "DB2 서버 동작 변경" 주제를 참조하십시오. 업그레이드된 데이터베이스와 관련하여 후속 데이터베이스 구성 갱신 또는 재설정 요청은 디폴트로 모든 데이터베이스 파티션에 적용됩니다.

기존 갱신 또는 재설정 명령 스크립트의 경우 이전에 언급한 것과 동일한 규칙이 모든 데이터베이스 파티션에 적용됩니다. UPDATE DATABASE CONFIGURATION 또는 RESET DATABASE CONFIGURATION 명령의 **DBPARTITIONNUM** 옵션을 포함하도록 스크립트를 수정하거나 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 레지스트리 변수를 설정할 수 있습니다.

db2CfgSet API를 호출하는 기존 응용프로그램에서는 버전 9.5 이상에 해당하는 명령어를 사용해야 합니다. 버전 9.5 이전의 동작을 수행하려면
DB2_UPDDBCFG_SINGLE_DBPARTITION 레지스트리 변수를 설정하거나 버전 9.5 이상의 버전 번호로 API를 호출하도록 응용프로그램을 수정할 수 있습니다(특정 데이터베이스 파티션의 데이터베이스 구성을 갱신하거나 재설정하기 위한 새 db2CfgSingleDbpartition 플래그 및 새 **dbpartitionnum** 필드 포함).

주: 데이터베이스 구성 값이 불일치하는 경우 각 데이터베이스 파티션을 개별적으로 갱신하거나 재설정할 수 있습니다.

파티션된 데이터베이스 환경 변수

DB2CHGPWD_EEE

- 운영 체제: AIX, Linux 및 Windows의 DB2 ESE
- 디폴트=NULL, 값: YES 또는 NO
- 이 변수는 다른 사용자가 AIX 또는 Windows ESE 시스템에서 암호를 변경하도록 허용하는지 여부를 지정합니다. 모든 데이터베이스 파티션 또는 노드의 암호가 Windows의 Windows 도메인 제어기나 AIX의 LDAP를 사용하여 중앙집중식으로 유지되는지 확인하십시오. 중앙집중식으로 유지되지 않는 경우, 모든 데이터베이스 파티션 또는 노드 전체에서 암호가 불일치할 수

있습니다. 그 결과, 변경을 수행하기 위해 사용자가 연결하는 데이터베이스 파티션에서만 암호가 변경될 수 있습니다.

DB2_FCM_SETTINGS

- 운영 체제: Linux
- 디폴트=YES, 값: FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]. FCM_MAXIMIZE_SET_SIZE의 디폴트값은 YES입니다.
- FCM_MAXIMIZE_SET_SIZE 토큰을 사용하여 **DB2_FCM_SETTINGS** 레지스트리 변수를 설정하여 FCM(Fast Communication Manager) 버퍼에 디폴트 2GB의 스페이스를 사전 할당할 수 있습니다. 이 기능을 사용하려면 토큰의 값이 YES 또는 TRUE여야 합니다.

DB2_FORCE_OFFLINE_ADD_PARTITION

- 운영 체제: 모두
- 디폴트=FALSE, 값: FALSE 또는 TRUE
- 이 변수를 사용하여 데이터베이스 파티션 서버 추가 조작이 오프라인으로 수행되도록 지정할 수 있습니다. 디폴트 설정인 FALSE는 데이터베이스를 오프라인으로 설정하지 않고 DB2 데이터베이스 파티션 서버를 추가할 수 있음을 표시합니다. 그러나 조작을 오프라인으로 수행할 경우 또는 일부 제한사항으로 인해 데이터베이스가 온라인 상태일 때 데이터베이스 파티션 서버를 추가할 수 없는 경우, **DB2_FORCE_OFFLINE_ADD_PARTITION**을 TRUE로 설정하십시오. 이 변수가 TRUE로 설정된 경우, 버전 9.5 및 이전 버전의 동작에 따라 새 DB2 데이터베이스 파티션 서버가 추가됩니다. 즉, 인스턴스가 종료된 후 재시작될 때까지 새 데이터베이스 파티션 서버가 인스턴스에 표시되지 않습니다.

DB2_NUM_FAILOVER_NODES

- 운영 체제: 모두
- 디폴트=2, 값: 0 - 필수 데이터베이스 파티션 수
- 장애 복구 시 머신에서 시작되어야 하는 추가 데이터베이스 파티션 수를 지정하려면 **DB2_NUM_FAILOVER_NODES**를 설정하십시오.

  DB2 데이터베이스 고가용성 솔루션에서 데이터베이스 서버가 실패하는 경우, 실패한 머신의 데이터베이스 파티션을 다른 머신에서 재시작할 수 있습니다. FCM(Fast Communication Manager)은 **DB2_NUM_FAILOVER_NODES**를 사용하여 이 장애 복구를 쉽게 하기 위해 각 머신에서 예약할 메모리 크기를 계산합니다.

  예를 들어, 다음 구성을 고려하십시오.

  – 머신 A에는 두 개의 데이터베이스 파티션이 있습니다(1 및 2).

  – 머신 B에는 두 개의 데이터베이스 파티션이 있습니다(3 및 4).

– A와 B 모두에서 **DB2_NUM_FAILOVER_NODES**는 2로 설정됩니다.

START DBM에서 FCM은 한 머신이 실패하는 경우 실패한 머신의 두 데이터베이스 파티션을 다른 머신에서 시작할 수 있도록 최대 네 개의 데이터베이스 파티션을 관리하기 위해 A와 B 모두에서 충분한 메모리를 예약합니다. 머신 A가 실패하는 경우, 데이터베이스 파티션 1 및 2를 머신 B에서 재시작할 수 있습니다. 머신 B가 실패하는 경우, 데이터베이스 파티션 3 및 4를 머신 A에서 재시작할 수 있습니다.

DB2_PARTITIONEDLOAD_DEFAULT

- 운영 체제: 지원되는 모든 ESE 플랫폼
- 디폴트=YES, 값: YES 또는 NO
- **DB2_PARTITIONEDLOAD_DEFAULT** 레지스트리 변수를 사용하면 ESE 특정 로드 옵션이 지정되지 않은 경우 사용자가 ESE 환경에서 로드 유틸리티의 디폴트 동작을 변경할 수 있습니다. 디폴트값은 YES이며 이는 ESE 환경에서 ESE 특정 로드 옵션을 지정하지 않은 경우 목표 테이블이 정의된 모든 데이터베이스 파티션에서 로딩이 시도되도록 지정합니다. 값이 NO이면 로드 유틸리티가 현재 연결되어 있는 데이터베이스 파티션에서만 로딩이 시도됩니다.

  주: 이 변수는 사용되지 않으며 추후 릴리스에서는 제거됩니다. LOAD 명령에는 동일한 동작을 수행하는 데 사용할 수 있는 여러 가지 옵션이 있습니다. LOAD 명령에 다음을 지정하여 이 변수의 NO 설정과 동일한 결과를 얻을 수 있습니다. `PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x`, 여기서 x는 데이터를 로드하려는 파티션의 파티션 번호입니다.

DB2PORTRANGE

- 운영 체제: Windows
- 값: nnnn:nnnn
- 이 값은 다른 머신에서 작성된 모든 추가 데이터베이스 파티션의 포트 범위도 동일하도록 FCM에서 사용되는 TCP/IP 포트 범위로 설정됩니다.

파티션된 데이터베이스 환경 구성 매개변수

통신

conn_elapse - 연결 경과 시간

이 매개변수는 두 개의 데이터베이스 파티션 서버 간 TCP/IP 연결을 설정해야 하는 제한 시간(초)을 지정합니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

온라인으로 구성 가능

전파 클래스

즉시

디폴트 [범위]

10 [0–100]

수치 단위

초

이 매개변수에 지정된 시간 안에 연결 시도가 성공하면 통신이 설정됩니다. 실패한 경우 연결을 다시 시도합니다. *max_connretries* 매개변수에 지정된 횟수만큼 연결을 시도했으나 항상 시간종료된 경우 오류가 발행됩니다.

fcm_num_buffers - FCM 버퍼 수

이 매개변수는 데이터베이스 서버 간 및 데이터베이스 서버 내에서 내부 통신(메시지)에 사용되는 4KB 버퍼 수를 지정합니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

- 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버
- 로컬 클라이언트가 있는 데이터베이스 서버
- 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

온라인으로 구성 가능

전파 클래스

즉시

디폴트 [범위]

32비트 플랫폼

Automatic [128 - 65 300]

64비트 플랫폼

Automatic [128 - 524,288]

- 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버: 디폴트는 1024임

- 로컬 클라이언트가 있는 데이터베이스 서버: 디폴트는 512임
- 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버: 디폴트는 4096임

단일 파티션 데이터베이스 시스템에서 이 매개변수는 *intra_parallel* 매개변수를 디폴트값 NO에서 YES로 변경하여 파티션 내 병렬 처리가 사용 가능한 경우에만 사용됩니다.

초기값 및 AUTOMATIC 속성을 모두 설정할 수 있습니다.

AUTOMATIC으로 설정된 경우 FCM은 자원 사용도를 모니터하고 30분 안에 사용되지 않은 경우 자원을 늘리거나 줄일 수 있습니다. 자원을 늘리거나 줄이는 양은 플랫폼에 따라 다르며 특히 Linux에서는 시작 값보다 25%만 늘릴 수 있습니다. 데이터베이스 관리 프로그램이 인스턴스 시작 시 지정된 자원 수를 할당할 수 없는 경우 인스턴스를 시작할 수 있을 때까지 구성 값을 조금씩 다시 늘립니다.

동일한 머신에 여러 개의 논리 노드가 있는 경우 이 매개변수의 값을 늘려야 할 수도 있습니다. 또한 시스템의 사용자 수, 시스템의 데이터베이스 파티션 서버 수 또는 복잡한 응용프로그램으로 인해 메시지 버퍼를 모두 사용한 경우 이 매개변수의 값을 늘려야 합니다.

여러 개의 논리 노드를 사용 중인 경우 하나의 *fcm_num_buffers* 버퍼 풀을 동일한 버퍼의 모든 논리 노드가 공유합니다. 풀의 크기는 *fcm_num_buffers* 값에 해당 실제 머신의 논리 노드 수를 곱하여 판별합니다. 사용 중인 값을 다시 조사하십시오. 여러 논리 노드가 상주하는 머신에 할당할 전체 FCM 버퍼 수를 고려하십시오.

fcm_num_channels - FCM 채널 수

이 매개변수는 각 데이터베이스 파티션의 FCM 채널 수를 지정합니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

- 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버
- 로컬 클라이언트가 있는 데이터베이스 서버
- 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버
- 로컬 클라이언트가 있는 Satellite 데이터베이스 서버

매개변수 유형

온라인으로 구성 가능

전파 클래스

즉시

디폴트 [범위]

UNIX 32비트 플랫폼

자동, 시작 값은 256, 512, 2 048 [128 - 120 000]

UNIX 64비트 플랫폼

자동, 시작 값은 256, 512, 2 048 [128 - 524 288]

Windows 32비트

자동, 시작 값은 10 000 [128 - 120 000]

Windows 64비트

자동, 시작 값은 256, 512, 2 048 [128 - 524 288]

- 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버의 경우 시작 값은 512입니다.
- 로컬 클라이언트가 있는 데이터베이스 서버의 경우 시작 값은 256입니다.
- 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 환경 서버의 경우 시작 값은 2 048입니다.

파티션되지 않은 데이터베이스 환경에서는 *intra_parallel* 매개변수가 활성이어야 *fcm_num_channels*를 사용할 수 있습니다.

FCM 채널은 DB2 엔진에서 실행 중인 EDU 간의 논리적 통신 엔드 포인트를 나타냅니다. 제어 순서(요청 및 응답)와 데이터 순서(테이블 큐 데이터) 둘 다 채널에 의존하여 파티션 간에 데이터를 전송합니다.

AUTOMATIC으로 설정된 경우 FCM은 채널 사용량을 모니터하며 요구사항 변화에 따라 증분하여 자원을 할당하고 해제합니다.

max_connretries - 노드 연결 재시도 수

이 매개변수는 두 개의 데이터베이스 파티션 서버 간 TCP/IP 연결을 설정하는 최대 연결 시도 수를 지정합니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

온라인으로 구성 가능

전파 클래스

즉시

디폴트 [범위]

5 [0–100]

두 개의 데이터베이스 파티션 서버 간 통신 설정 시도가 실패한 경우(예: *conn_elapse* 매개변수에 지정된 값과 같아짐) *max_connretries*는 데이터베이스 파티션 서버에 대한 연결 재시도 수를 지정합니다. 이 매개변수에 지정된 값을 초과한 경우 오류가 리턴됩니다.

max_time_diff - 노드 간의 최대 시간 차이

이 매개변수는 노드 구성 파일에 나열된 데이터베이스 파티션 서버 간에 허용되는 최대 시간 차이를 분 단위로 지정합니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

구성 가능

디폴트 [범위]

60 [1 - 1 440]

수치 단위

분

각 데이터베이스 파티션 서버에는 자신의 시스템 클럭이 있습니다. 둘 이상의 데이터베이스 파티션 서버가 트랜잭션과 연관되어 있고 이들 서버 클럭 간의 시간 차이가 MAX_TIME_DIFF 매개변수에서 지정한 시간보다 클 경우 트랜잭션이 거부되고 SQLCODE가 리턴됩니다. (트랜잭션은 데이터 수정이 연결된 경우에만 거부됩니다.)

데이터베이스가 파티션된 환경에서도 SQLCODE가 리턴됩니다. 이 환경에서는 DB2가 시스템 클럭을 SQLOGCTL.LFH 로그 제어 파일에 저장된 가상 시간소인(VTS)과 비교합니다. .LFH 파일의 시간소인이 시스템 시간보다 작은 경우 시스템 클럭이 이 시간과 일치할 때까지 데이터베이스 로그의 시간이 VTS로 설정됩니다. 다중 노드 간의 시간 차이가 MAX_TIME_DIFF 매개변수보다 작지만 SQL1473N 오류 메시지도 리턴됩니다.

DB2는 *세계 표준시*(UTC)를 사용하므로 이 매개변수를 설정할 때 다른 시간대는 고려사항이 아닙니다. 세계 표준시는 그리니치 평균시와 같습니다.

start_stop_time - 시작 및 중지 시간종료

이 매개변수는 모든 데이터베이스 파티션 서버가 START DBM 또는 STOP DBM 명령에 응답해야 하는 시간을 분 단위로 지정합니다. ADD DBPARTITIONNUM 조작의 시간종료 값으로도 사용됩니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

로컬 및 리모트 클라이언트가 있는 데이터베이스 서버

매개변수 유형

온라인으로 구성 가능

전파 클래스

즉시

디폴트 [범위]

10 [1 - 1 440]

수치 단위

분

지정된 시간 내에 DB2START 명령에 응답하지 않는 데이터베이스 파티션 서버는 해당 인스턴스의 홈 디렉토리 아래의 sqllib 서브디렉토리의 log 서브디렉토리에 있는 db2start 오류 로그에 메시지를 전송합니다. 재시작하기 전에 이 노드에서 DB2STOP을 실행해야 합니다.

지정된 시간 내에 DB2STOP 명령에 응답하지 않는 데이터베이스 파티션 서버는 해당 인스턴스의 홈 디렉토리 아래의 sqllib 서브디렉토리의 log 서브디렉토리에 있는 db2stop 오류 로그에 메시지를 전송합니다. 응답하지 않는 각 데이터베이스 파티션 서버에 대해 db2stop을 실행하거나 모두에 대해 실행할 수 있습니다. (이미 중지된 서버는 중지되었음을 알리는 메시지를 리턴합니다.)

다중 파티션 데이터베이스에서 db2start 또는 db2stop 조작이 *start_stop_time* 데이터베이스 관리 프로그램이 지정하는 값 내에 완료되지 않는 경우, 시간종료된 데이터베이스 파티션은 내부적으로 제거됩니다. *start_stop_time* 값이 낮은 많은 데이터베이스 파티션이 있는 환경에서 이 동작이 발생할 수 있습니다. 이 동작을 해결하려면 *start_stop_time*의 값을 늘리십시오.

DB2START, START DATABASE MANAGER 또는 ADD DBPARTITIONNUM 명령 중 하나를 사용하여 새 데이터베이스 파티션을 추가하는 경우, 데이터베이스 파티션 추가 조작은 인스턴스 내의 각 데이터베이스가 자동 스토리지에 대해 사용 가능한지 여부를 판별해야 합니다. 이는 각 데이터베이스의 카탈로그 파티션과 통신하여 수행됩니다. 자동 스토리지가 사용되는 경우 스토리지 경로 정의가 이 통신의 파트로 검색됩니다. 마찬가지로, 데이터베이스 파티션이 있는 시스템 임시 테이블 스페이스를 작성하는 경우 조작은 다른 데이터베이스 파티션 서버와 통신하여 해당 서버에 있는 데이터베이스 파티션의 테이블 스페이스 정의를 검색해야 합니다. *start_stop_time* 매개변수의 값을 결정할 때 이러한 요소를 고려해야 합니다.

병렬 처리

intra_parallel - 파티션 내 병렬 처리 사용

이 매개변수는 데이터베이스 관리 프로그램이 파티션 내 병렬 처리를 사용할 수 있는지 여부를 지정합니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

- 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버
- 로컬 클라이언트가 있는 데이터베이스 서버
- 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

구성 가능

디폴트 [범위]

NO (0) [SYSTEM (-1), NO (0), YES (1)]

-1 값은 데이터베이스 관리 프로그램을 실행 중인 하드웨어에 따라 매개변수 값을 『YES』 또는 『NO』로 설정합니다.

이 매개변수가 ″YES″인 경우 병렬 성능 향상을 활용할 수 있는 일부 조작은 데이터베이스 쿼리 및 인덱스 작성입니다.

주:

- 병렬 인덱스 작성에서는 이 구성 매개변수를 사용하지 않습니다.
- 이 매개변수 값을 변경하면 패키지가 데이터베이스에 리바인드되고 일부 성능 저하가 발생할 수 있습니다.

max_querydegree - 병렬 처리의 최대 쿼리 수준

이 매개변수는 이 데이터베이스 관리 프로그램 인스턴스에서 실행 중인 SQL문에 사용되는 파티션 내 병렬 처리의 최대 수준을 지정합니다. SQL문은 명령문 실행 시 데이터베이스 파티션에서 이 병렬 조작 수를 초과하여 사용하지 않습니다.

구성 유형

데이터베이스 관리 프로그램

적용 대상

- 로컬 및 리모트 클라이언트가 있는 데이터베이스 서버
- 로컬 클라이언트가 있는 데이터베이스 서버
- 로컬 및 리모트 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

온라인으로 구성 가능

전파 클래스

명령문 경계

디폴트 [범위]

-1 (ANY) [ANY, 1 - 32 767](ANY는 시스템이 판별함을 의미함)

데이터베이스 파티션이 SQL문에 대해 파티션 내 병렬 처리를 사용할 수 있도록 하려면 *intra_parallel* 구성 매개변수를 『YES』로 설정해야 합니다. 병렬 인덱스 작성 시 더 이상 *intra_parallel* 매개변수가 필요하지 않습니다.

이 구성 매개변수의 디폴트값은 -1입니다. 이 값은 시스템이 옵티마이저가 판별한 병렬 처리 수준을 사용함을 의미합니다. 그렇지 않으면 사용자 지정 값을 사용합니다.

주: CURRENT DEGREE 특수 레지스터 또는 DEGREE 바인드 옵션을 사용하여 명령문 컴파일 시 SQL문의 병렬 처리 수준을 지정할 수 있습니다.

활성 응용프로그램에 대한 병렬 처리의 최대 쿼리 수준은 SET RUNTIME DEGREE 명령을 사용하여 수정할 수 있습니다. 사용되는 실제 런타임 수준은 다음의 값 중 낮은 값입니다.

- *max_querydegree* 구성 매개변수
- 응용프로그램 런타임 등급
- SQL문 컴파일 등급

이 구성 매개변수는 쿼리에만 적용됩니다.

제 5 부 관리 API, 명령, SQL문

제 29 장 관리 API

sqleaddn - 파티션된 데이터베이스 환경에 데이터베이스 파티션 추가

데이터베이스 파티션 서버에 데이터베이스 파티션을 추가합니다.

범위

이 API는 실행된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

sqleaddn API 매개변수

+pAddNodeOptions

입력. 선택적 sqle_addn_options 구조의 포인터. 이 구조를 사용하여 작성한 모든 데이터베이스 파티션의 시스템 임시 테이블 스페이스 정의가 있는 경우 해당 소스 데이터베이스 파티션 서버를 지정합니다. 지정하지 않으면(즉, NULL 포인터를 지정하면) 시스템 임시 테이블 스페이스 정의는 카탈로그 파티션의 정의와 동일하게 됩니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgaddn API 특정 매개변수

addnOptionsLen

입력. 선택적 sqle_addn_options 구조 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수.

사용 시 참고사항

이 API는 데이터베이스 파티션 서버가 한 개의 데이터베이스를 포함하며 파티션 추가 조작 시에 데이터베이스가 카탈로그되지 않은 환경에 추가되는 경우에만 사용해야 합니다. 이 상황에서는 데이터베이스가 카탈로그되지 않기 때문에 파티션 추가 조작이 데이터베이스를 인식하지 못하고 새 데이터베이스 파티션 서버의 데이터베이스에 대해 데이터베이스 파티션을 작성하지 않습니다. 새 데이터베이스 파티션 서버에서 데이터베이스 파티션 연결 시도는 오류를 초래합니다. 데이터베이스는 sqleaddn API를 사용하여 새 데이터베이스 파티션 서버에서 데이터베이스에 대한 데이터베이스 파티션을 작성하기 전에 우선 카탈로그해야 합니다.

이 API는 둘 이상의 데이터베이스가 있고 이 데이터베이스 중 하나 이상이 파티션 추가 조작 시에 카탈로그된 경우 사용하면 안됩니다. 이 상황에서는 sqlecran API를 사용하여 파티션 추가 조작 시에 카탈로그되지 않은 각 데이터베이스에 대해 데이터베이스 파티션을 작성하십시오. 카탈로그 해제된 각 데이터베이스는 sqlecran API를 사용하여 새 데이터베이스 파티션 서버에서 데이터베이스에 대해 데이터베이스 파티션을 작성하기 전에 우선 카탈로그해야 합니다.

새 데이터베이스 파티션을 추가하기 전에 작성해야 하는 컨테이너에 필요한 스토리지가 충분한지 확인하십시오.

노드 추가 조작은 인스턴스에 있는 모든 데이터베이스의 새 데이터베이스 파티션 서버에 비어 있는 데이터베이스 파티션을 작성합니다. 새 데이터베이스 파티션의 구성 매개변수는 디폴트값으로 설정됩니다.

주: 카탈로그 해제된 데이터베이스는 새 데이터베이스 파티션을 추가할 때 인식되지 않습니다. 카탈로그 해제된 데이터베이스는 새 데이터베이스 파티션에 존재하지 않습니다. 새 데이터베이스 파티션에서 데이터베이스에 연결하려고 하면 오류 메시지 SQL1013N이 리턴됩니다.

로컬에서 데이터베이스 파티션 작성 시 노드 추가 조작이 실패하면 제거 단계가 되고 이 경우 작성된 모든 데이터베이스가 로컬에서 삭제됩니다. 즉, 데이터베이스 파티션은 추가 중인 데이터베이스 파티션 서버(즉, 로컬 데이터베이스 파티션 서버)에서만 제거됩

니다. 기존 데이터베이스 파티션은 모든 다른 데이터베이스 파티션 서버에서 이전과 동일하게 유지됩니다. 이 상황이 실패하면 추가적인 제거는 수행되지 않으며 오류가 리턴됩니다.

새 데이터베이스 파티션 서버의 데이터베이스 파티션은 ALTER DATABASE PARTITION GROUP 문을 사용하여 데이터베이스 파티션 서버를 데이터베이스 파티션 그룹에 추가해야 사용자 데이터를 포함하는 데 사용할 수 있습니다.

이 API는 데이터베이스 작성 또는 데이터베이스 삭제 조작이 진행 중인 경우 실패합니다. API는 조작이 완료될 때 다시 호출할 수 있습니다.

데이터베이스의 자동 스토리지 사용 가능 여부를 판별하기 위해 sqleaddn API가 인스턴스의 각 데이터베이스에 대해 카탈로그 파티션과 통신합니다. 자동 스토리지가 사용 가능한 경우 스토리지 경로 정의는 통신 중에 검색됩니다. 이와 유사하게 시스템 임시 테이블 스페이스가 데이터베이스 파티션과 같이 작성되는 경우 tsqleaddn API는 테이블 스페이스 정의를 검색하기 위해 파티션된 데이터베이스 환경의 다른 데이터베이스 파티션 서버와 통신해야 합니다. **start_stop_time** 데이터베이스 관리 프로그램 구성 매개변수를 사용하여 다른 데이터베이스 파티션 서버가 자동 스토리지 및 테이블 스페이스 정의에 응답해야 하는 시간을 분 단위로 지정합니다. 이 시간이 초과되면 API가 실패합니다. **start_stop_time** 값을 증가시키고 API를 다시 호출하십시오.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

sqlecran - 데이터베이스 파티션 서버에 데이터베이스 작성

API를 호출하는 데이터베이스 파티션 서버에서만 데이터베이스를 작성합니다. 이 API는 일반적인 용도로는 사용되지 않습니다. 예를 들어 데이터베이스 파티션 서버의 데이터베이스 파티션이 손상되어 재작성해야 하는 경우에 db2Restore와 같이 사용해야 합니다. 이 API를 잘못 사용하면 시스템에서 불일치가 발생하기 때문에 주의해서 사용해야 합니다.

주: 이 API를 사용하여 손상되어 삭제된 데이터베이스 파티션을 재작성하는 경우 이 데이터베이스 파티션 서버의 데이터베이스는 리스토어 보류 상태가 됩니다. 데이터베이스 파티션을 재작성한 후에 데이터베이스는 이 데이터베이스 파티션 서버에서 바로 리스토어해야 합니다.

범위

이 API는 호출된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

인스턴스. 다른 데이터베이스 파티션 서버에서 데이터베이스를 작성하려면 우선 해당 데이터베이스 파티션 서버에 접속해야 합니다. 데이터베이스 연결은 처리 중에 이 API에서 임시로 작성됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
        unsigned short dbNameLen,
        struct sqlca * pSqlca,
        void * pReserved,
        char * pDbName);
```

sqlecran API 매개변수

pDbName

입력. 작성할 데이터베이스 이름이 포함된 문자열. NULL이면 안됩니다.

pReserved

입력. 영(0)을 가리키거나 널(NULL)로 설정되는 여분의 포인터. 나중에 사용하도록 예약됩니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgcran API 특정 매개변수

reservedLen

입력. pReserved 길이로 예약되었습니다.

dbNameLen

입력. 데이터베이스 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수.

사용 시 참고사항

데이터베이스가 제대로 작성되면 리스토어 보류 상태가 됩니다. 데이터베이스는 사용하기 전에 이 데이터베이스 파티션 서버에서 리스토어해야 합니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

---

sqledpan - 데이터베이스 파티션 서버에서 데이터베이스 삭제

지정한 데이터베이스 파티션 서버에서 데이터베이스를 삭제합니다. 파티션된 데이터베이스 환경에서만 실행할 수 있습니다.

범위

이 API는 호출된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음. 호출 지속 기간 동안 인스턴스가 접속됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
```

```
unsigned short DbAliasLen,
struct sqlca * pSqlca,
void * pReserved2,
char * pDbAlias);
```

sqledpan API 매개변수

pDbAlias

입력. 삭제할 데이터베이스 별명이 포함된 문자열. 이 이름은 시스템 데이터베이스 디렉토리에서 실제 데이터베이스 이름을 참조하는 데 사용됩니다.

pReserved

예약됨. NULL이어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgdpan API 특정 매개변수

Reserved1

나중에 사용하도록 예약됩니다.

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수.

pReserved2

영(0)을 가리키거나 널(NULL)로 설정되는 여분의 포인터. 나중에 사용하도록 예약됩니다.

사용 시 참고사항

이 API를 잘못 사용하면 시스템에서 불일치가 발생하기 때문에 주의해서 사용해야 합니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

---

sqledrpn - 데이터베이스 파티션 서버 삭제 가능 여부 점검

데이터베이스에서 데이터베이스 파티션 서버를 사용 중인지 확인합니다. 데이터베이스 파티션 서버가 삭제 가능한지를 나타내는 메시지가 리턴됩니다.

범위

이 API는 발행된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

sqledrpn API 매개변수

Action

요청된 조치. 유효한 값은 SQL_DROPNODE_VERIFY입니다.

pReserved

예약됨. NULL이어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgdrpn API 특정 매개변수

Reserved1

pReserved2 길이로 예약되었습니다.

pReserved2

NULL로 설정되거나 0을 지시하는 여분의 포인터. 나중에 사용되도록 예약되어 있습니다.

사용 시 참고사항

데이터베이스 파티션 서버가 사용되고 있지 않음을 나타내는 메시지가 리턴되면 db2stop 명령을 DROP NODENUM과 같이 사용하여 db2nodes.cfg 파일에서 데이터베이스 파티션 서버 항목을 제거하십시오. 그러면 파티션된 데이터베이스 환경에서 데이터베이스 파티션 서버가 제거됩니다.

데이터베이스 파티션 서버가 사용 중임을 나타내는 메시지가 리턴되면 다음 조치를 수행해야 합니다.

1. 삭제할 데이터베이스 파티션 서버가 인스턴스의 각 데이터베이스에 대해 데이터베이스 파티션을 갖게 됩니다. 이 데이터베이스 파티션 중 하나라도 데이터를 포함한 경우 이 데이터베이스 파티션을 사용하여 데이터베이스 파티션 그룹에 다시 분배하십시오. 삭제 중이지 않은 데이터베이스 파티션 서버에 있는 데이터베이스 파티션의 데이터를 이동하기 위해 데이터베이스 파티션 그룹을 다시 분배하십시오.
2. 데이터베이스 파티션 그룹을 재분배하고 나면 데이터베이스 파티션을 사용하고 있는 모든 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제하십시오. 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 제거하려면 sqludrdt API 또는 ALTER DATABASE PARTITION GROUP 문을 사용할 수 있습니다.
3. 데이터베이스 파티션 서버에 정의된 이벤트 모니터를 삭제하십시오.
4. sqledrpn을 재실행하여 데이터베이스 파티션 서버의 데이터베이스 파티션이 더 이상 사용되지 않도록 하십시오.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

---

sqlugrpn - 행에 대해 데이터베이스 파티션 서버 번호 가져오기

DB2 9.7을 시작으로 이 API는 더 이상 사용되지 않습니다. db2GetRowPartNum(행에 대해 데이터베이스 파티션 서버 번호 가져오기) API를 사용하여 행에 대한 데이터베이스 파티션 번호 및 데이터베이스 파티션 서버 번호를 리턴하십시오. 이 API가 사용되는 경우 SQL2768N 메시지가 리턴됩니다.

분산 키 값을 바탕으로 데이터베이스 파티션 번호 및 데이터베이스 파티션 서버 번호를 리턴합니다. 응용프로그램은 이 정보를 사용하여 특정 테이블 행이 저장되는 데이터베이스 파티션 서버를 판별합니다.

파티셔닝 데이터 구조인 sqlupi가 이 API의 입력입니다. 구조는 sqlugtpi API로 리턴할 수 있습니다. 다른 입력은 해당 분산 키 값의 문자 표시입니다. 출력은 분산 전략 및 분산 맵의 해당 데이터베이스 파티션 서버 번호로 생성된 데이터베이스 파티션 번호입니다. 분산 맵 정보가 제공되지 않으면 데이터베이스 파티션 번호만 리턴됩니다. 이는 데이터 분산을 분석할 때 유용할 수 있습니다.

이 API를 호출할 때 데이터베이스 관리 프로그램은 실행 중일 필요는 없습니다.

범위

이 API는 db2nodes.cfg 파일의 데이터베이스 파티션 서버에서 호출해야 합니다. 클라이언트와 서버 사이의 코드 페이지 및 엔디안의 차이로 인해 데이터베이스 파티셔닝 정

보에 오류가 발생할 수 있기 때문에 이 API는 클라이언트에서 호출하면 안됩니다.

권한 부여

없음

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

sqlugrpn API 매개변수

num_ptrs

ptr_array의 포인터 수. 값은 part_info 매개변수에 지정한 값과 동일해야 합니다. 즉, part_info->sqld.

ptr_array

part_info에 지정한 분산 키의 각 파트에 해당하는 값의 문자 표시를 지시하는 포인터 배열. 널(NULL) 값이 필요한 경우 해당 포인터가 널(NULL)로 설정됩니다. 생성된 컬럼의 경우 이 함수는 행에 대해 값을 생성하지 않습니다. 사용자가 행의 올바른 파티셔닝을 위해 값을 제공해야 합니다.

ptr_lens

part_info에 지정된 파티셔닝 키의 각 부분에 해당하는 값에 대한 문자 표시 길이가 포함된 부호없는 정수의 배열.

territory_ctrycode

목표 데이터베이스의 국가/지역 코드. 이 값은 GET DATABASE CONFIGURATION 명령을 사용하여 데이터베이스 구성 파일에서 가져올 수도 있습니다.

codepage

목표 데이터베이스의 코드 페이지. 이 값은 GET DATABASE CONFIGURATION 명령을 사용하여 데이터베이스 구성 파일에서 가져올 수도 있습니다.

part_info

sqlupi 구조의 포인터

part_num

데이터베이스 파티션 번호를 저장하는 데 사용되는 2바이트의 부호있는 정수의 포인터.

node_num

노드 번호를 저장하는 데 사용되는 SQL_PDB_NODE_TYPE 필드의 포인터. 포인터가 널(NULL)인 경우 노드 번호가 리턴되지 않습니다.

chklvl 입력 매개변수에서 수행되는 점검 레벨을 지정하는 부호없는 정수. 값을 영(0)으로 지정하는 경우 점검이 수행되지 않습니다. 0이 아닌 값을 지정하면 모든 입력 매개변수가 점검됩니다.

sqlca 출력. sqlca 구조의 포인터

dataformat

분산 키 값 표시를 지정합니다. 가능한 값은 다음과 같습니다.

SQL_CHARSTRING_FORMAT

모든 분산 키 값은 문자열로 표시됩니다. 이는 디폴트값입니다.

SQL_IMPLIEDDECIMAL_FORMAT

내재된 소수점 위치는 컬럼 정의로 별됩니다. 예를 들어 컬럼 정의가 DECIMAL(8,2)인 경우 값 12345는 123.45로 처리됩니다.

SQL_PACKEDDECIMAL_FORMAT

모든 10진수 컬럼 분산 키 값은 10진수 형식으로 압축됩니다.

SQL_BINARYNUMERICS_FORMAT

모든 숫자 키 값은 빅 엔디안(big-endian) 2진 형식입니다.

pReserved1

나중에 사용하도록 예약됩니다.

pReserved2

나중에 사용하도록 예약됩니다.

사용 시 참고사항

운영 체제에서 지원되는 데이터 유형은 분산 키로 정의된 유형과 동일합니다.

주: CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC 데이터 유형은 이 API를 호출하기 전에 데이터베이스 코드 페이지로 변환해야 합니다.

숫자 및 날짜 시간 데이터 유형의 경우 문자 표시는 API가 호출되는 각 시스템의 코드 페이지에 있어야 합니다.

node_num이 널(null)인 경우 분산 맵을 입력해야 합니다. 즉, part_info 매개변수의 pmaplen 필드(part_info->pmaplen)는 2 또는 8192입니다. 그 외의 경우에는 SQLCODE -6038이 리턴됩니다. 분산 키를 정의해야 합니다. 즉, part_info 매개변수의 sqld 필드(part_info->sqld)는 0보다 커야 합니다. 그 외의 경우에는 SQLCODE -2032가 리턴됩니다.

널(null) 값을 허용하지 않는 컬럼에 널(NULL) 값을 지정하면 SQLCODE -6039가 리턴됩니다.

입력 문자열의 모든 앞 공백 및 뒤 공백은 스트립되며 뒤 공백만 스트립될 수 있는 CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC 데이터 유형은 예외입니다.

제 30 장 명령

REDISTRIBUTE DATABASE PARTITION GROUP

데이터베이스 파티션 그룹에 있는 데이터베이스 파티션 간에 데이터를 재분배합니다. 특정 시스템 요구사항에 맞게 데이터의 목표 분산이 일정(디폴트)하거나 사용자 지정될 수 있습니다.

REDISTRIBUTE DATABASE PARTITION GROUP 명령은 데이터베이스 파티션 그룹에 있는 모든 파티션 사이에 데이터를 재분배합니다. 이는 데이터베이스 파티션 그룹에 있는 모든 오브젝트에 영향을 주며 하나의 오브젝트로만 제한될 수 없습니다.

이 명령은 카탈로그 데이터베이스 파티션에서만 발행할 수 있습니다. 어떤 데이터베이스 파티션이 각 데이터베이스에 대한 카탈로그 데이터베이스 파티션인지 판별하려면 LIST DATABASE DIRECTORY 명령을 사용하십시오.

범위

이 명령은 데이터베이스 파티션 그룹에 있는 모든 데이터베이스 파티션에 영향을 줍니다.

권한 부여

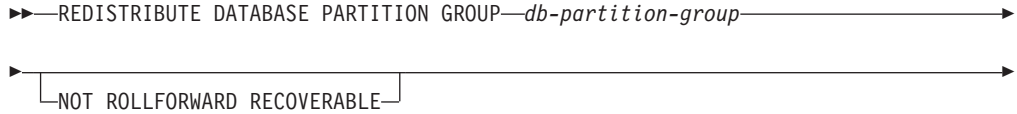
다음 권한 중 하나가 필요합니다.

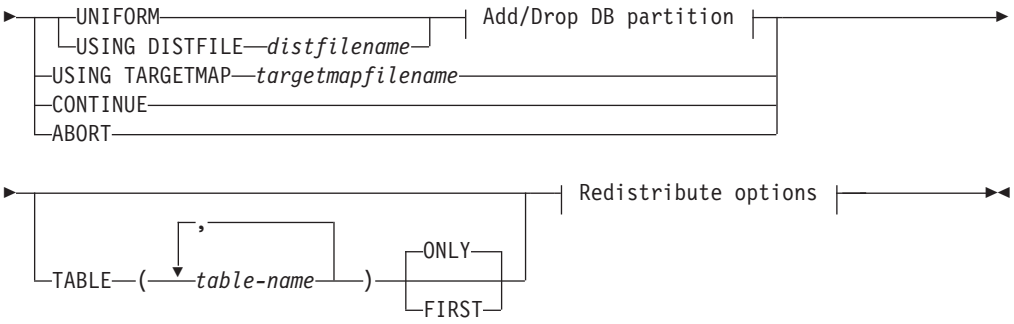
- SYSADM
- SYSCTRL
- DBADM

또한 다음과 같은 권한 그룹 중 하나도 필요합니다.

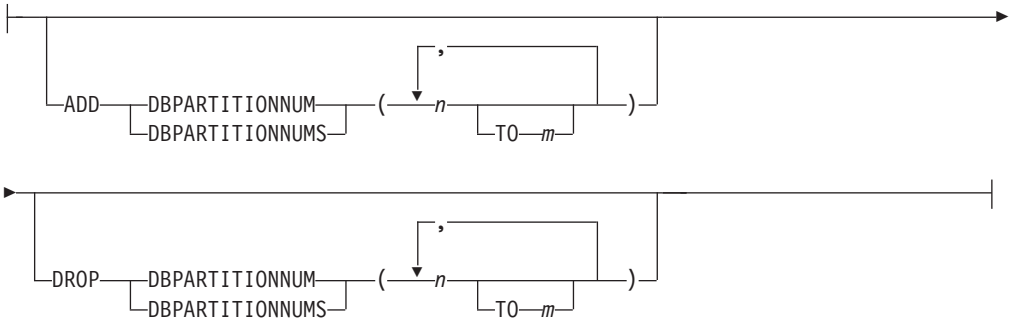
- 재분배 중인 데이터베이스 파티션 그룹의 모든 테이블에 대한 DELETE, INSERT, SELECT 특권
- DATAACCESS 권한

명령 구문

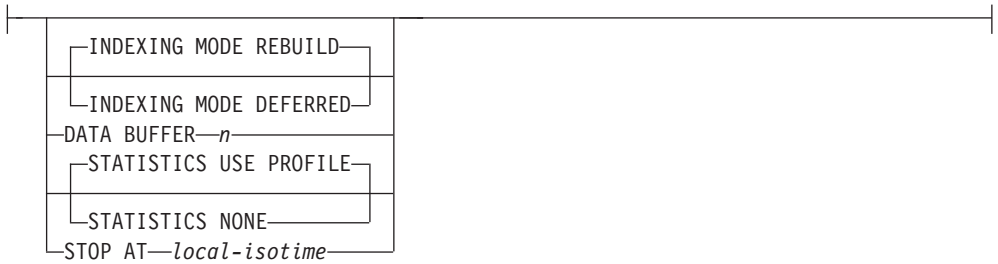




Add/Drop DB partition:



Redistribute options:



명령 매개변수

DATABASE PARTITION GROUP *db-partition-group*

데이터베이스 파티션 그룹 이름. 이 한 부분으로 된 이름은 SYSCAT.DBPARTITIONGROUPS 카탈로그 테이블에 설명된 데이터베이스 파티션 그룹을 식별합니다. 데이터베이스 파티션 그룹은 현재 재분배될 수 없습니다.

주: IBMCATGROUP 및 IBMTEMPGROUP 데이터베이스 파티션 그룹에 있는 테이블을 재분배할 수 없습니다.

NOT ROLLFORWARD RECOVERABLE

이 옵션 사용 시, REDISTRIBUTE DATABASE PARTITION GROUP 명령은 롤 포워드 복구 가능하지 않습니다.

- 내부 삽입 및 삭제 조작 대신 대량으로 데이터가 이동됩니다. 이렇게 하면 테이블을 스캔하고 액세스하는 시간을 줄여 더 나은 성능을 가져옵니다.
- 각 삽입 및 삭제 조작에는 로그 레코드가 더 이상 필요하지 않습니다. 이는 데이터 재분배 시 사용자 시스템에 사용 중인 대량의 로그 스페이스 및 로그 아카이브 스페이스 관리가 필요하지 않음을 의미합니다. 과거에 대량의 사용 중인 로그 스페이스와 스토리지 요구사항으로 인해 단일 데이터 재분배 조작을 여러 개의 더 작은 재분배 태스크로 구분함으로써 단말 간 데이터 재분배 조작 완료에 더 많은 시간이 걸릴 수 있는 상황에서는 특히 유용합니다.
- REDISTRIBUTE DATABASE PARTITION GROUP 명령에 NOT ROLLFORWARD RECOVERABLE 옵션을 사용하는 경우 재분배 조작 시에 XML 컬럼을 포함하는 테이블에 대해 INDEXING MODE DEFERRED 옵션이 사용됩니다. 테이블에 XML 컬럼이 포함되지 않은 경우 재분배 조작은 명령 실행 시에 지정한 인덱스 모드를 사용합니다.

이 옵션이 사용되지 않으면, 모든 행 이동의 확장 로깅이 수행되어 인터럽트, 오류 또는 다른 비즈니스 필요에 따라 나중에 데이터베이스를 복구할 수 있습니다.

UNIFORM

데이터가 해시 파티션에 균등하게 분산되도록 지정하지만(즉, 모든 해시 파티션이 동일한 행 수를 갖는 것으로 간주됨) 각 데이터베이스 파티션에 동일한 해시 파티션 수가 맵핑되지 않습니다. 재분배 후 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션은 대략 동일한 수의 해시 파티션을 갖게 됩니다.

USING DISTFILE *distfilename*

분산 키 값의 분산이 균등하지 않을 경우, 이 옵션을 사용하여 데이터베이스 파티션 그룹의 데이터베이스 파티션에서 균등한 데이터 재분배를 달성하십시오.

32 768 해시 파티션에서 현재 데이터 분산을 표시하려면 *distfilename*을 사용하십시오.

각 해시 파티션이 표시하는 데이터의 양을 표시하려면 행 계수, 바이트 볼륨 또는 기타 수치를 사용하십시오. 이 유틸리티는 파티션과 연관된 정수 값을 해당 파티션의 가중치로 읽습니다. *distfilename*이 지정될 때 유틸리티는 데이터베이스 파티션 그룹의 데이터베이스 파티션에서 데이터를 가능한 균등하게 재분배하는 데 사용하는 목표 분산 맵을 생성합니다. 재분배 후, 데이터베이스 파티션 그룹에 있는 각 데이터베이스 파티션의 가중치는 대략적으로 동일합니다(데이터베이스 파티션의 가중치는 해당 데이터베이스 파티션에 맵핑되는 모든 해시 파티션의 가중치 합계임).

예를 들어, 입력 분산 파일에는 다음과 같은 항목이 포함될 수 있습니다.

```
10223
1345
112000
0
100
...
```

예에서, 해시 파티션 2의 가중치는 112000이며 파티션 3(가중치 0)에는 데이터가 전혀 맵핑되지 않습니다.

*distfilename*은 32 768 양의 정수값을 문자 형식으로 포함해야 합니다. 이들 값의 합계는 4 294 967 295 이하가 되어야 합니다.

*distfilename*의 경로를 지정하지 않은 경우 현재 디렉토리를 사용합니다.

USING TARGETMAP *targetmapfilename*

*targetmapfilename*에 지정된 파일은 목표 분산 맵으로 사용됩니다. 데이터 재분배는 이 파일에 따라 수행됩니다. 경로를 지정하지 않으면 현재 디렉토리를 사용합니다.

목표 맵에 포함된 데이터베이스 파티션이 데이터베이스 파티션 그룹에 있지 않으면 오류가 리턴됩니다. REDISTRIBUTE DATABASE PARTITION GROUP 명령을 실행하기 전에 ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM문을 발행하십시오.

목표 맵에서 제외된 데이터베이스 파티션이 데이터베이스 파티션 그룹에 있는 경우, 이 데이터베이스 파티션은 파티션되는 데 포함되지 않습니다. 이러한 데이터베이스 파티션은 REDISTRIBUTE DATABASE PARTITION GROUP 명령 전후에 ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM문을 사용하여 삭제될 수 있습니다.

CONTINUE

이전에 실패하거나 중지된 REDISTRIBUTE DATABASE PARTITION GROUP 조작을 계속 수행합니다. 아무 것도 발생하지 않으면 오류가 리턴됩니다.

ABORT

이전에 실패하거나 중지된 REDISTRIBUTE DATABASE PARTITION GROUP 조작을 중단합니다. 아무 것도 발생하지 않으면 오류가 리턴됩니다.

ADD

DBPARTITIONNUM *n*

TO *m*

n 또는 *n* **TO** *m*은 데이터베이스 파티션 그룹에 추가된 데이터베이스 파티션 번호 목록을 지정합니다. 지정된 파티션은 데이터베이스 파티션 그룹에 정의되어 있지 않아야 합니다(SQLSTATE 42728). 이것은 ADD

DBPARTITIONNUM 절이 지정된 ALTER DATABASE PARTITION GROUP문을 실행하는 것과 같습니다.

DBPARTITIONNUMS *n*

TO *m*

n 또는 *n* **TO** *m*은 데이터베이스 파티션 그룹에 추가된 데이터베이스 파티션 번호 목록을 지정합니다. 지정된 파티션은 데이터베이스 파티션 그룹에 정의되어 있지 않아야 합니다(SQLSTATE 42728). 이것은 ADD DBPARTITIONNUM 절이 지정된 ALTER DATABASE PARTITION GROUP문을 실행하는 것과 같습니다.

주: 이 옵션을 사용하여 데이터베이스 파티션을 추가하면, 테이블 스페이스의 컨테이너는 데이터베이스 파티션 그룹에서 가장 번호가 낮은 기존 파티션의 해당 테이블 스페이스의 컨테이너에 기준합니다. 이로 인해 컨테이너 간의 이름지정 충돌이 발생하는 경우, 이는 새 파티션이 기존 컨테이너와 동일한 실제 머신에 있을 경우 발생할 수 있으며, 이때 이 옵션을 사용하지 말아야 합니다. 대신 ALTER DATABASE PARTITION GROUP문은 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 발행하기 전에 WITHOUT TABLESPACES 옵션과 함께 사용되어야 합니다. 그러면 수동으로 적절한 이름을 지정하여 테이블 스페이스 컨테이너를 작성할 수 있습니다.

DROP

DBPARTITIONNUM *n*

TO *m*

n 또는 *n* **TO** *m*은 데이터베이스 파티션 그룹에서 삭제되는 데이터베이스 파티션 번호 목록을 지정합니다. 지정된 파티션은 데이터베이스 파티션 그룹에 정의되어 있어야 합니다(SQLSTATE 42729). 이것은 DROP DBPARTITIONNUM 절이 지정된 ALTER DATABASE PARTITION GROUP문을 실행하는 것과 같습니다.

DBPARTITIONNUMS *n*

TO *m*

n 또는 *n* **TO** *m*은 데이터베이스 파티션 그룹에서 삭제되는 데이터베이스 파티션 번호 목록을 지정합니다. 지정된 파티션은 데이터베이스 파티션 그룹에 정의되어 있어야 합니다(SQLSTATE 42729). 이것은 DROP DBPARTITIONNUM 절이 지정된 ALTER DATABASE PARTITION GROUP문을 실행하는 것과 같습니다.

TABLE *tablename*

재분배 처리를 위한 테이블 순서를 지정합니다.

ONLY

테이블 순서 다음에 **ONLY** 키워드(디폴트)가 오면, 지정된 테이블만이 재분배됩니다. 남아 있는 테이블은 나중에 REDISTRIBUTE CONTINUE 명령으로 처리될 수 있습니다. 이는 디폴트값입니다.

FIRST

테이블 순서 다음에 **FIRST** 키워드가 표시되면, 지정된 테이블이 표시된 순서로 재분배되고 데이터베이스 파티션 그룹에 남아있는 테이블은 무작위 순서로 재분배됩니다.

INDEXING MODE

NOT ROLLFORWARD RECOVERABLE 옵션이 지정되면 이 매개변수는 재분배 중 인덱스가 유지되는 방식을 지정합니다. 가능한 값은 다음과 같습니다.

REBUILD

인덱스는 스크래치에서 재빌드됩니다. 이 옵션을 사용하기 위해 인덱스를 유효하게 할 필요는 없습니다. 이 옵션의 사용 결과로, 인덱스 페이지가 디스크에서 함께 클러스터됩니다.

DEFERRED

재분배는 인덱스를 유지하려고 하지 않습니다. 인덱스는 새로 고칠 필요가 있는 것으로 표시됩니다. 그러한 인덱스에 대한 최초 액세스는 재빌드를 강제하거나, 데이터베이스 재시작 시 인덱스를 재빌드할 수 있습니다.

주: 비MDC 테이블의 경우 테이블에 유효하지 않은 인덱스가 있으면, INDEXING MODE DEFERRED를 지정하지 않은 경우 **REDISTRIBUTE DATABASE PARTITION GROUP** 명령이 자동으로 인덱스를 재빌드합니다. MDC 테이블의 경우 **INDEXING MODE DEFERRED**를 지정하면, 유틸리티가 MDC 테이블을 처리하기 위해 복합 인덱스를 필요로 하므로 테이블 재분배가 시작되기 전에 유효하지 않은 복합 인덱스가 재빌드됩니다.

DATA BUFFER *n*

유틸리티에서 데이터 전송에 필요한 버퍼 스페이스로 사용할 4KB 페이지의 수를 지정합니다. 지정된 값이 지원되는 최소값보다 작으면, 최소값이 사용되며 경고는 리턴하지 않습니다. 이 메모리는 유틸리티 힙에서 바로 할당되며 *크기*는 **util_heap_sz** 데이터베이스 구성 매개변수로 수정할 수 있습니다. 값이 지정되지 않으면 처음에 각 테이블 처리 시 런타임에 유틸리티에 의해 적절한 디폴트

값이 계산됩니다. 특히, 테이블의 시간 재분배 시 유틸리티 힙에서 사용 가능한 메모리의 50%를 사용하고 다양한 테이블 등록 정보를 고려하는 것이 디폴트입니다.

STOP AT *local-isotime*

이 옵션이 지정되면, 각 테이블의 데이터 재분배를 시작하기 전에 *local-isotime*을 현재 로컬 시간소인과 비교합니다. 지정된 *local-isotime*이 현재 로컬 시간소인과 같거나 이전이면, 유틸리티는 경고 메시지로 중지합니다. 중지 시 진행 중인 테이블의 데이터 재분배 처리는 인터럽트 없이 완료됩니다. 테이블의 새 데이터 재분배 처리는 시작되지 않습니다. **CONTINUE** 옵션을 사용하여 처리되지 않은 테이블을 재분배할 수 있습니다. 이 *local-isotime* 값은 조합된 날짜 및 시간을 식별하는 7파트 문자열 시간소인으로 지정됩니다. 형식은 로컬 시간으로 표시된 *yyyy-mm-dd-hh.mm.ss.nnnnnn*(연도, 월, 일, 시, 분, 초, 마이크로초)입니다.

STATISTICS

이 옵션은 통계 프로파일이 있는 테이블의 통계를 유틸리티가 수집해야 함을 지정합니다. 이 옵션 지정은 데이터 재분배 완료 이후 별도로 RUNSTATS 명령을 발행하는 것보다 더 효과적입니다.

USE PROFILE

통계 프로파일이 있는 테이블에 대한 통계가 수집됩니다. 통계 프로파일이 없는 테이블의 경우, 어떤 것도 완료되지 않습니다. 이는 디폴트 값입니다.

NONE

테이블의 통계는 수집되지 않습니다.

예: 재분배 단계

노드 그룹에서 노드를 삭제하거나 추가할 수 있습니다. 다음은 노드 그룹에 새 노드를 추가하고 데이터를 재분배하기 위한 단계입니다. 추가된 데이터베이스 파티션은 분산 맵에 없지만 데이터베이스 파티션 그룹에 있는 테이블 스페이스에 대한 컨테이너는 작성되었습니다. 데이터베이스 파티션은 데이터베이스 파티션 그룹 재분배 조작이 완료되었을 때 데이터베이스 파티션이 분산 맵에 추가됩니다.

1. 재분배해야 할 노드 그룹을 식별하십시오. 이 문서에서 재분배해야 하는 노드 그룹은 『sampleNodegrp』입니다.
2. 재분배하기 전에 사용하지 않도록 설정하거나 제거해야 하는 오브젝트를 식별하십시오.
   a. MQT 복제: 이 유형의 MQT는 REDISTRIBUTE 유틸리티의 일부로 지원되지 않습니다. 재분배를 실행하기 전에 삭제하고 이후에 다시 작성해야 합니다.

```
SELECT tabschema, tabname
  FROM syscat.tables
  WHERE partition_mode = 'R'
```

b. 테이블에 기록 이벤트 모니터: 재분배되는 데이터베이스 파티션 그룹에 있는 테이블을 포함하는 자동으로 활성화되는 모든 테이블에 기록 이벤트 모니터를 사용하지 않도록 설정해야 합니다.

```
SELECT distinct evmonname FROM syscat.eventtables E
JOIN syscat.tables T on T.tabname = E.tabname AND T.tabschema = E.tabschema
JOIN syscat.tablespaces S on S.tbspace = T.tbspace AND S.ngname = 'sampleNodegrp'
```

c. Explain 테이블: 단일 파티션 노드 그룹에서 Explain 테이블을 작성할 것을 권장합니다. 그러나 재분배가 필요한 노드 그룹에 정의된 경우, 재분배 이전에 삭제하고 재분배 완료 시 재정의할 것을 고려할 수 있습니다(현재까지 생성된 데이터를 유지보수하지 않아도 되는 경우).

d. 테이블 액세스 모드 및 로드 상태: 재분배되는 노드 그룹의 모든 테이블이 전체 액세스 모드 상태이고 보류 중이거나 진행 중인 상태의 로드가 없어야 합니다.

```
SELECT distinct trim(T.creator) || ₩'.₩' || trim(T.name)
  AS name, T.access_mode, A.load_status
  FROM sysibm.systables T, sysibm.sysnodegroups N, sysibmadm.admintabinfo A
  WHERE T.pmap_id = N.pmap_id
   AND A.tabschema = T.creator
   AND A.tabname = T.name
   AND N.name = 'sampleNodegrp'
   AND (T.access_mode <> 'F' or A.load_status is not null)
```

e. 통계 프로파일: 통계 프로파일이 테이블에 대해 정의된 경우 테이블 통계는 재분배 프로세스의 일부로 갱신 가능합니다. REDISTRIBUTE 유틸리티가 테이블의 통계를 갱신하는 경우 재분배를 위해 모든 데이터가 스캔되어 RUNSTATS에 대해 추가 데이터 스캔이 필요하지 않기 때문에 입출력이 감소합니다.

```
RUNSTATS on table schema.table
  USE PROFILE runstats_profile
  SET PROFILE ONLY
```

3. 데이터베이스 구성을 검토하십시오. **util_heap_sz**는 데이터베이스 파티션 간의 데이터 이동 처리에 매우 중요합니다. 재분배 지속기간 동안 가능한 많은 메모리를 **util_heap_sz**에 할당하십시오. 인덱스 재빌드가 재분배의 일부로 수행되는 경우 충분한 **sortheap**이 필요합니다. 재분배 성능을 향상시키기 위해서는 **util_heap_sz** 및 **sortheap**을 필요한 만큼 늘리십시오.

4. 새 데이터베이스 파티션에 사용할 데이터베이스 구성 설정을 검색하십시오. 데이터베이스 파티션을 추가할 때 디폴트 데이터베이스 구성이 사용됩니다. 결과적으로, 전체 웨어하우스 간의 균형을 조정하기 위해 REDISTRIBUTE 명령이 실행되기 전에 새 노드에서 데이터베이스 구성을 갱신하는 것이 중요합니다.

```
SELECT name,
  CASE WHEN deferred_value_flags = 'AUTOMATIC'
    THEN deferred_value_flags
    ELSE substr(deferred_value,1,20)
    END  AS deferred_value
  FROM sysibmadm.dbcfg
```

```
WHERE dbpartitionnum = existing-node
  AND deferred_value != ''
  AND name NOT IN ('hadr_local_host','hadr_local_svc','hadr_peer_window',
                   'hadr_remote_host','hadr_remote_inst','hadr_remote_svc',
                   'hadr_syncmode','hadr_timeout','backup_pending','codepage',
                   'codeset','collate_info','country','database_consistent',
                   'database_level','hadr_db_role','log_retain_status',
                   'loghead','logpath','multipage_alloc','numsegs','pagesize',
                   'release','restore_pending','restrict_access',
                   'rollfwd_pending','territory','user_exit_status',
                   'number_compat','varchar2_compat','database_memory')
```

5. 최근 복구점을 확인하려면 재분배 프로세스를 시작하기 전에 데이터베이스(재분배할 노드 그룹의 테이블 스페이스)를 백업하십시오.
6. db2nodes.cfg 파일을 갱신하고 새 데이터 BCU 데이터베이스 파티션 스펙을 추가하여 DB2에 새 데이터 BCU를 정의하고 ADD NODE WITHOUT TABLESPACES 명령을 사용하여 DB2에 새 데이터베이스 파티션을 정의하십시오.

   ```
   db2start nodenum x export DB2NODE=x
   db2 add node without tablespaces
   db2stop nodenum x
   ```

   주: 데이터 BCU에 대해 첫 번째 논리적 포트가 아니면, 연속 논리적 포트에 대한 위의 명령 시퀀스 전후에 첫 번째 논리적 포트 번호 시작 및 중지를 실행하십시오.
7. 새로 정의된 데이터베이스 파티션에 대해 시스템 임시 테이블 스페이스 컨테이너를 정의하십시오.

   ```
   ALTER TABLESPACE tablespace_name ADD container_information
   ON dbpartitionnums (x to y)
   ```
8. 데이터 BCU에 걸쳐 있는 데이터베이스 파티션 그룹에 새 논리 데이터베이스 파티션을 추가하십시오.

   ```
   ALTER DATABASE PARTITION GROUP partition_group_name
     ADD dbpartitionnums (x to y)
     WITHOUT TABLESPACES
   ```
9. 새로 정의된 데이터베이스 파티션에 대해 영구 데이터 테이블 스페이스 컨테이너를 정의하십시오.

   ```
   ALTER TABLESPACE tablespace_name ADD container_information
   ON dbpartitionnums (x to y)
   ```
10. 4단계에서 검색한 데이터베이스 구성 설정을 새 데이터베이스 파티션에 적용하십시오(또는 구성 지원의 새 DB2 9.5 단일 뷰를 사용하여 모든 데이터베이스 파티션에 대해 단일 UPDATE DB CFG 명령을 실행하십시오).
11. 재분배할 데이터베이스 파티션 그룹의 정의를 캡처한 후 이 그룹에 존재하는 복제된 MQT를 삭제하십시오.

    ```
    db2look -d dbname -e -z schema -t replicated_MQT_table_names
    -o repMQTs.clp
    ```

12. 재분배할 데이터베이스 파티션 그룹에 존재하는 테이블에 기록 이벤트 모니터를 사용하지 않도록 설정하십시오.

    ```
    SET EVENT MONITOR monitor_name STATE 0
    ```

13. REDISTRIBUTE 유틸리티를 실행하여 모든 데이터베이스 파티션 사이에 일정하게 재분배하십시오. 다음은 단순한 재분배 명령을 보여줍니다.

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP sampleNodegrp NOT ROLLFORWARD RECOVERABLE uniform;
    ```

사용자는 또한 REDISTRIBUTE 명령의 입력으로 테이블 목록을 지정하여 테이블이 처리될 순서를 지시할 것을 고려해야 합니다. REDISTRIBUTE 유틸리티는 데이터를 이동시킵니다(압축 및 간략화). 선택적으로, 인덱스가 재빌드되고 통계 프로파일이 정의된 경우 통계가 갱신됩니다. 따라서 이전 명령 대신 다음 스크립트를 실행할 수 있습니다.

```
REDISTRIBUTE DATABASE PARTITION GROUP sampleNodegrp NOT ROLLFORWARD
RECOVERABLE uniform TABLE (tab1, tab2,...) FIRST;
```

NOT ROLLFORWARD RECOVERABLE 옵션 사용 결과

REDISTRIBUTE DATABASE PARTITION GROUP 명령이 발행되고 NOT ROLLFORWARD RECOVERABLE 옵션이 지정되면, 이동된 각 행의 로그 레코드 쓰기를 최소화하는 최소 로깅 전략이 사용됩니다. 모든 데이터 이동을 완전히 로그하는 접근 방식이 대형 시스템에 가능하며, 비현실적인 활성 영구 로그 스페이스를 요구하며 일반적으로 성능이 더 저하된 성능 특성을 가지므로 재분배 조작의 사용 편리성에는 이러한 유형의 로깅이 중요합니다. 그러나 이러한 최소 로깅 모델의 결과로서, REDISTRIBUTE DATABASE PARTITION GROUP 명령이 복구 가능한 롤 포워드가 *아님*을 인식하는 것이 중요합니다. 이것은 재분배 조작을 통해 데이터베이스를 롤 포워드하는 모든 조작이 UNAVAILABLE 상태로 남아 있는 재분배 조작으로 모든 테이블을 처리하는 결과를 가져옴을 의미합니다. 그러한 테이블은 삭제만 가능하며, 이들 테이블의 데이터를 복구할 방법이 없음을 의미합니다. 그 이유는 복구 가능한 데이터베이스의 경우, REDISTRIBUTE DATABASE PARTITION GROUP 유틸리티가 NOT ROLLFORWARD RECOVERABLE 옵션으로 발행되면 사용하는 모든 테이블 스페이스를 BACKUP PENDING 상태로 만들어 정상적인 재분배 조작의 끝에서 사용자가 강제로 모든 재분배 테이블 스페이스를 백업하게 하기 때문입니다. 재분배 조작 이후의 백업 시, 사용자는 재분배 조작 자체를 통해 롤 포워드하지 말아야 합니다.

재분배 유틸리티에서 롤 포워드 복구 가능성이 없다는 점은 간과할 수 없는 중요한 문제입니다. 재분배에 의해 처리되는 테이블 스페이스를 사용자가 백업하는 재분배 주기의 종료 지점을 포함하여 재분배 조작이 실행되는 동안 데이터베이스의 테이블 갱신을 허용한 경우(재분배되는 데이터베이스 파티션 그룹 외부의 테이블인 경우에도) 데이터베이스 컨테이너 손상과 같은 심각한 장애의 경우 해당 갱신사항이 손실될 수 있습니다. 그러한 갱신이 유실되는 이유는 재분배 조작이 복구 가능한 롤 포워드가 아니기 때문입니다. 재분배 조작 이전의 백업에서 데이터베이스를 리스토어하는 것이 필요한 경우, 위에서 설명한 대로 재분배를 통해 롤 포워드하지 않는 재분배 조작 중 수행된 갱

신을 재생하기 위해 로그를 롤 포워드하는 것이 가능하지 않고, 재분배된 테이블을 사용 불가능한 상태로 둡니다. 그러므로 이 상황에서 수행할 수 있는 유일한 것은 롤 포워드 없이 재분배하기 전에 수행한 백업에서 데이터베이스를 리스토어하는 것입니다. 그러면 재분배 조작을 다시 수행할 수 있습니다. 그러나 원래의 재분배 조각 중 발생한 모든 갱신은 유실됩니다.

이러한 사항은 매우 중요합니다. 재분배 조작 중 갱신이 유실되지 않음을 보장하려면, 다음 중 하나가 참이어야 합니다.

- 영향을 받는 테이블 스페이스를 백업 중인 명령 완료 이후 기간을 포함하여 REDISTRIBUTE DATABASE PARTITION GROUP 명령의 조작 중 갱신하는 것을 피합니다.
- 재분배 조작 중 적용되는 갱신은 반복 가능한 소스에서 오는데, 이는 언제든지 다시 적용될 수 있음을 의미합니다. 예를 들어, 갱신 소스가 파일에 저장되는 데이터이며 일괄처리 중 갱신이 적용되는 경우, 데이터베이스 리스토어를 필요로 하는 실패에서 조차도 갱신이 유실되지 않는 이유는 언제든지 다시 갱신을 적용할 수 있기 때문입니다.

재분배 조작 중 데이터베이스 갱신을 허용하는 경우, 필요하면 사용자는 그러한 갱신이 적절한지 아니면 데이터베이스 리스토어 이후에 갱신을 반복할 수 있는지 여부를 결정해야 합니다.

주: REDISTRIBUTE DATABASE PARTITION GROUP 명령을 조작하는 동안 모든 실패가 이러한 문제점을 유발하지는 않습니다. 사실상 대부분은 문제점을 유발하지 않습니다. REDISTRIBUTE DATABASE PARTITION GROUP 명령은 완전하게 재시작할 수 있으며 이는 작업 중간에 유틸티리가 실패하는 경우 **CONTINUE** 또는 **ABORT** 옵션을 사용하여 간편하게 계속하거나 중단할 수 있음을 나타냅니다. 위에서 언급하는 실패는 재분배 조작 이전에 수행된 백업에서 리스토어하는 것이 필요한 실패입니다.

사용 시 참고사항

- **NOT ROLLFORWARD RECOVERABLE** 옵션을 지정하고 데이터베이스가 복구 가능한 데이터베이스인 경우 유틸리티가 테이블 스페이스에 처음 액세스할 때 BACKUP PENDING 상태가 됩니다. 해당 테이블 스페이스의 모든 테이블은 테이블 스페이스가 백업될 때까지 읽기 전용으로 됩니다.
- 재분배 조작이 실행 중이면, 처리되는 각 테이블의 시작 및 종료 시간과 같은 정보와 재분배 조작에 대한 일반 정보가 있는 이벤트 로그 파일을 생성합니다. 이 이벤트 로그 파일은 작성됩니다.
  - 서브디렉토리와 파일 이름에 대해 *database-name.database-partition-group-name.timestamp*.log 형식을 사용하는 Linux 및 UNIX 시스템의 homeinst/sqllib/redist 디렉토리.

– 서브디렉토리와 파일 이름에 대해 *database-name.database-partition-group-name.timestamp*.log 형식을 사용하는 Windows 운영 체제의 **DB2INSTPROF**₩*instance*₩redist 디렉토리(여기서 **DB2INSTPROF**는 **DB2INSTPROF** 레지스트리 변수 값입니다).
– 시간소인 값은 명령이 발행된 시간입니다.

재분배 이벤트 로그에 대한 자세한 정보는 『재분배 이벤트 로그 파일』 주제를 참조하십시오.

- 유틸리티 프로그램은 처리 중 간헐적인 COMMIT를 수행합니다.
- 재분배가 일어난 테이블에 종속성을 갖는 모든 패키지는 무효화됩니다. 데이터베이스 파티션 그룹 재분배 조작이 완료된 후 이러한 패키지를 명시적으로 리바인드하는 것이 바람직합니다. 명시적으로 리바인드하면 유효하지 않은 패키지에 대한 첫 번째 SQL 요청을 실행할 때 초기 지연이 발생하지 않습니다. 재분배 메시지 파일에는 재분배가 일어난 모든 테이블 목록이 들어 있습니다.
- 디폴트로 재분배 유틸리티는 통계 프로파일이 있는 테이블의 통계를 갱신합니다. 통계 프로파일이 없는 테이블의 경우, 재분배 조작이 완료된 후에 RUNSTATS 명령을 발행하거나 db2Runstats API를 호출하여 해당 테이블의 테이블 및 인덱스 통계를 별도로 갱신하는 것이 좋습니다.
- DATA CAPTURE CHANGES로 정의된 복제된 테이블이나 복제된 구체화된 쿼리 테이블을 포함하는 데이터베이스 파티션 그룹은 재분배할 수 없습니다.
- 데이터베이스 파티션 그룹에 기존의 선언된 임시 테이블이나 작성된 임시 테이블이 있는 사용자 임시 테이블 스페이스가 있으면 재분배를 수행할 수 없습니다.
- **INDEXING MODE**와 같은 옵션은 해당 옵션이 적용되지 않는 테이블에서 경고 없이 무시됩니다. 예를 들어, **INDEXING MODE**는 인덱스가 없는 테이블에서 무시됩니다.
- 재분배 조작을 시작하기 전에 로드 보류 상태의 테이블이 없는지 확인하십시오. LOAD QUERY 명령을 사용하여 테이블 상태를 점검할 수 있습니다.
- 데이터베이스 파티션 서버 추가 요청이 보류 중 또는 진행 중인 경우에는 REDISTRIBUTE DATABASE PARTITION GROUP 명령이 실패할 수 있습니다(SQLSTATE 55071). 새 데이터베이스 파티션 서버가 온라인으로 인스턴스에 추가되고 모든 응용프로그램이 새 데이터베이스 파티션 서버를 인지하는 것이 아니라면 이 명령도 실패합니다(SQLSTATE 55077).

호환성

DB2 버전 9.5 이전의 XML 레코드 형식을 사용하는 XML 컬럼을 포함하는 테이블은 재분배할 수 없습니다. 테이블을 새 형식으로 이주하려면 ADMIN_MOVE_TABLE 스토어드 프로시저를 사용하십시오.

버전 8 이전 버전과의 호환성:

- **NODEGROUP** 키워드는 **DATABASE PARTITION GROUP**으로 대체될 수 있습니다.

db2nchg - 데이터베이스 파티션 서버 구성 변경

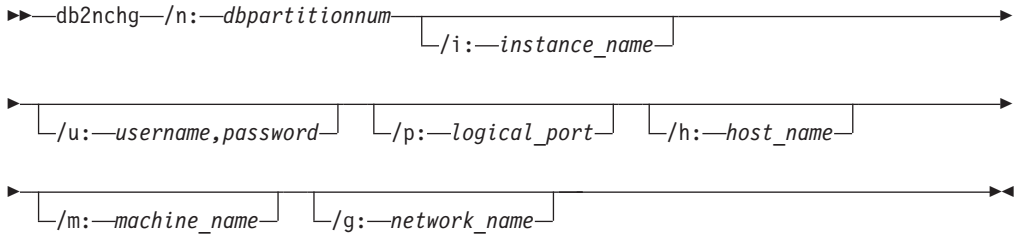
데이터베이스 파티션 서버 구성을 수정합니다. 여기에는 한 머신에서 다른 머신으로 데이터베이스 파티션 서버(노드) 이동, 머신의 TCP/IP 호스트 이름 변경 및 데이터베이스 파티션 서버(노드)에 대한 다른 논리적 포트 번호 또는 다른 네트워크 이름 선택이 포함됩니다. 이 명령은 데이터베이스 파티션 서버가 중지되는 경우에만 사용할 수 있습니다.

이 명령은 Windows 운영 체제에서만 사용 가능합니다.

권한 부여

로컬 관리자

명령 구문



명령 매개변수

/n:*dbpartitionnum*

변경될 데이터베이스 파티션 서버의 구성의 데이터베이스 파티션 번호를 지정합니다.

/i:*instance_name*

이 데이터베이스 파티션 서버가 참여하는 인스턴스를 지정합니다. 매개변수가 지정되지 않는 경우 디폴트는 현재 인스턴스입니다.

/u:*username,password*

사용자 이름 및 암호를 지정합니다. 매개변수가 지정되지 않는 경우 기존 사용자 이름과 암호가 적용됩니다.

/p:*logical_port*

데이터베이스 파티션 서버에 대한 논리적 포트를 지정합니다. 데이터베이스 파티션 서버를 다른 머신으로 이동하려면 이 매개변수를 지정해야 합니다. 매개변수가 지정되지 않는 경우 논리적 포트 번호는 변경되지 않습니다.

/h:*host_name*

FCM이 내부 통신에 사용하는 TCP/IP 호스트 이름을 지정합니다. 이 매개변수가 지정되지 않는 경우 호스트 이름은 동일하게 남습니다.

/m:*machine_name*

데이터베이스 파티션 서버가 상주할 머신을 지정합니다. 데이터베이스 파티션 서버는 인스턴스에 기존 데이터베이스가 없는 경우에만 이동될 수 있습니다.

/g:*network_name*

데이터베이스 파티션 서버에 대한 네트워크 이름을 변경합니다. 이 매개변수를 사용하여 머신에 다중 IP 주소가 있을 때 데이터베이스 파티션 서버에 특정 IP 주소를 적용할 수 있습니다. 네트워크 이름 또는 IP 주소를 입력할 수 있습니다.

예:

인스턴스 TESTMPP에 참여하는 데이터베이스 파티션 2에 지정된 논리적 포트를 논리적 포트 3으로 변경하려면 다음 명령을 입력하십시오.

```
db2nchg /n:2 /i:TESTMPP /p:3
```

---

db2ncrt - 인스턴스에 데이터베이스 파티션 서버 추가

인스턴스에 데이터베이스 파티션 서버(노드)를 추가합니다.

이 명령은 Windows 운영 체제에서만 사용할 수 있습니다.

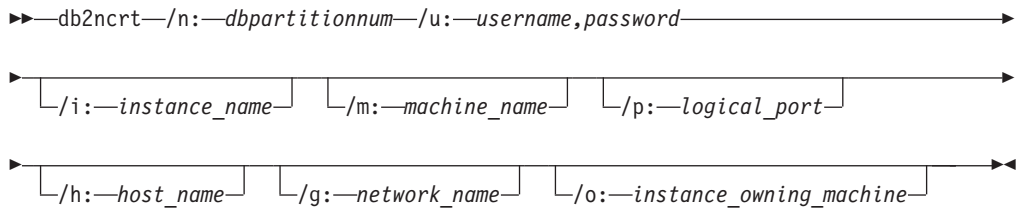
범위

데이터베이스 파티션 서버가 인스턴스가 이미 존재하는 컴퓨터에 추가되면 데이터베이스 파티션 서버는 논리적 데이터베이스 파티션 서버로서 컴퓨터에 추가됩니다. 데이터베이스 파티션 서버가 인스턴스가 존재하지 않는 컴퓨터에 추가되는 경우 인스턴스가 추가되고 컴퓨터는 새 실제 데이터베이스 파티션 서버가 됩니다. 인스턴스에 데이터베이스가 있는 경우 이 명령을 사용하지 않아야 합니다. 대신 START DATABASE MANAGER 명령이 ADD DBPARTITIONNUM 옵션과 함께 발행되어야 합니다. 이것은 데이터베이스가 새 데이터베이스 파티션 서버에 올바르게 추가되도록 보장합니다. 또한 데이터베이스가 작성된 인스턴스에 데이터베이스 파티션 서버를 추가할 수 있습니다. 파일을 변경하면 파티션된 데이터베이스 환경에서 불일치를 유발할 수 있으므로 `db2nodes.cfg` 파일은 편집해서는 안됩니다.

권한 부여

새 데이터베이스 파티션 서버가 추가되는 컴퓨터의 로컬 관리자 권한.

명령 구문



명령 매개변수

/n:*dbpartitionnum*

데이터베이스 파티션 서버를 식별하는 고유한 데이터베이스 파티션 번호입니다. 입력되는 번호는 1 - 999 범위에 있을 수 있습니다.

/u:*username,password*

DB2에 대한 로그온 어카운트 이름 및 암호를 지정합니다.

/i:*instance_name*

인스턴스 이름을 지정합니다. 매개변수가 지정되지 않는 경우 디폴트는 현재 인스턴스입니다.

/m:*machine_name*

데이터베이스 파티션 서버가 있는 Windows 워크스테이션의 컴퓨터 이름을 지정합니다. 데이터베이스 파티션 서버가 리모트 컴퓨터에 추가되는 경우 이 매개변수는 필수입니다.

/p:*logical_port*

데이터베이스 파티션 서버에 사용되는 논리적 포트 번호를 지정합니다. 이 매개변수를 지정하지 않으면 지정되는 논리적 포트 번호는 0입니다. 논리적 데이터베이스 파티션 서버를 작성할 때 이 매개변수를 지정해야 하며 사용 중이 아닌 논리적 포트 번호를 선택해야 합니다. 다음 제한사항을 주의하십시오.

- 모든 컴퓨터는 논리적 포트가 0인 데이터베이스 파티션 서버를 가져야 합니다.
- 포트 번호는 x:₩winnt₩system32₩drivers₩etc₩ 디렉토리에서 FCM 통신에 대해 보존되는 포트 범위를 초과하지 않아야 합니다. 예를 들어 4 포트의 범위가 현재 인스턴스에 대해 보존되는 경우 최대 포트 번호는 3입니다. 포트 0은 디폴트 논리적 데이터베이스 파티션 서버에 사용됩니다.

/h:*host_name*

FCM이 내부 통신에 사용하는 TCP/IP 호스트 이름을 지정합니다. 데이터베이스 파티션 서버가 리모트 컴퓨터에 추가될 때 이 매개변수는 필수입니다.

/g:*network_name*

데이터베이스 파티션 서버의 네트워크 이름을 지정합니다. 매개변수가 지정되지 않는 경우 시스템에서 발견되는 첫 번째 IP 주소가 사용됩니다. 이 매개변수를

사용하여 컴퓨터에 다중 IP 주소가 있을 때 데이터베이스 파티션 서버에 특정 IP 주소를 적용할 수 있습니다. 네트워크 이름 또는 IP 주소를 입력할 수 있습니다.

/o:*instance_owning_machine*

인스턴스 소유 컴퓨터의 컴퓨터 이름을 지정합니다. 디폴트는 로컬 컴퓨터입니다. 이 매개변수는 인스턴스 소유 컴퓨터가 아닌 모든 컴퓨터에서 db2ncrt 명령이 호출될 때 필수입니다.

예:

새 데이터베이스 파티션 서버가 데이터베이스 파티션 2로 알려지고 논리적 포트 1을 사용하는 인스턴스 소유 컴퓨터 SHAYER의 TESTMPP 인스턴스에 새 데이터베이스 파티션 서버를 추가하려면 다음 명령을 입력하십시오.

```
db2ncrt /n:2 /u:QBPAULZ¥paulz,g1reeky /i:TESTMPP /m:TEST /p:1 /o:SHAYER /h:TEST
```

db2ndrop - 인스턴스에서 데이터베이스 파티션 서버 삭제

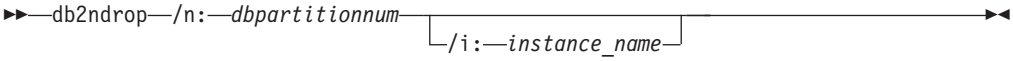
데이터베이스가 없는 인스턴스에서 데이터베이스 파티션 서버(노드)를 삭제(drop)합니다. 데이터베이스 파티션 서버가 삭제되는 경우 데이터베이스 파티션 번호가 새 데이터베이스 파티션 서버에 재사용될 수 있습니다. 이 명령은 데이터베이스 파티션 서버가 중지되는 경우에만 사용할 수 있습니다.

이 명령은 Windows 운영 체제에서만 사용 가능합니다.

권한 부여

데이터베이스 파티션 서버가 삭제될 머신의 로컬 관리자 권한.

명령 구문



명령 매개변수

/n:*dbpartitionnum*

데이터베이스 파티션 서버를 식별하는 고유한 데이터베이스 파티션 번호입니다.

/i:*instance_name*

인스턴스 이름을 지정합니다. 매개변수가 지정되지 않는 경우 디폴트는 현재 인스턴스입니다.

예:

```
db2ndrop /n:2 /i=KMASCI
```

사용 시 참고사항

인스턴스 소유 데이터베이스 파티션 서버(dbpartitionnum 0)가 인스턴스에서 삭제되는 경우 인스턴스는 사용 불가능하게 됩니다. 인스턴스를 삭제하려면 db2idrop 명령을 사용하십시오.

이 인스턴스에 데이터베이스가 있는 경우 이 명령을 사용해서는 안됩니다. 대신 `db2stop drop nodenum` 명령을 사용해야 합니다. 이것은 데이터베이스 파티션 서버가 파티션 데이터베이스 환경에서 올바르게 제거되도록 합니다. 또한 데이터베이스가 존재하는 인스턴스에서 데이터베이스 파티션 서버를 삭제할 수도 있습니다. 파일을 변경하면 파티션된 데이터베이스 환경에서 불일치를 유발할 수 있으므로 `db2nodes.cfg` 파일은 편집해서는 안됩니다.

다중 논리적 데이터베이스 파티션 서버를 실행 중인 머신에서 논리적 포트 0에 지정된 데이터베이스 파티션 서버를 삭제하려면, 다른 논리적 포트에 지정된 다른 모든 데이터베이스 파티션 서버가 먼저 삭제되어야 합니다. 각 데이터베이스 파티션 서버는 논리적 포트 0에 지정된 데이터베이스 파티션 서버가 있어야 합니다.

제 31 장 SQL 언어 요소

데이터 유형

데이터베이스 파티션 호환 가능 데이터 유형

*데이터베이스 파티션 호환성*은 파티션 키의 해당 컬럼에 대한 기본 데이터 유형 사이에서 정의됩니다. 데이터베이스 파티션 호환 데이터 유형은 한 유형에서 하나씩 동일한 값의 두 변수가 동일한 파티션 함수에 의해 동일한 분배 맵 인덱스에 맵핑되는 등록 정보를 갖습니다.

462 페이지의 표 42에서는 데이터베이스 파티션에서 데이터 유형의 호환성을 나타냅니다.

데이터베이스 파티션 호환성은 다음과 같은 등록 정보를 갖습니다.

- 내부 형식은 DATE, TIME 및 TIMESTAMP에 사용됩니다. 이들은 서로 호환되지 않으며, 이들 중 어느 것도 문자 또는 그래픽 데이터 유형과 호환되지 않습니다.
- 널(null) 값 허용은 파티션 호환성에 영향을 주지 않습니다.
- 조합은 파티션 호환성에 영향을 줍니다. 로케일 구분 UCA 기반 조합은 조합의 강도(S) 속성이 무시되는 경우를 제외하고 조합에서 완전 일치를 필요로 합니다. 기타 모든 조합은 파티션 호환성 판별 목적으로 같다고 간주됩니다.
- FOR BIT DATA를 사용하여 정의된 문자 컬럼은 로케일 구분 UCA 기반 조합 이외의 조합이 사용되는 경우 FOR BIT DATA가 없는 문자 컬럼과만 호환 가능합니다.
- 호환되는 데이터 유형의 널(NULL) 값들은 동일하게 취급됩니다. 호환되지 않는 데이터 유형의 널(NULL) 값들에 대해서는 서로 다른 결과가 산출될 수 있습니다.
- UDT의 기본 데이터 유형을 사용하여 데이터베이스 파티션 호환성을 분석합니다.
- 분배 키에서 동일한 값의 시간소인은 시간소인 정밀도가 다르더라도 동일하게 취급됩니다.
- 분산 키의 동일한 값의 소수 부분은 소수점 이하 자릿수와 전체 자릿수가 다르더라도 동일하게 취급됩니다.
- 문자열(CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC)에 있는 후행 공백은 시스템이 제공하는 해시 기능에 의해 무시됩니다.
- 로케일 구분 UCA 기반 조합이 사용되는 경우, CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC는 호환 가능한 데이터 유형입니다. 기타 조합이 사용되는 경우, CHAR 및 VARCHAR은 호환 가능한 유형이고 GRAPHIC 및 VARGRAPHIC는

호환 가능한 유형이지만 CHAR 및 VARCHAR은 GRAPHIC 및 VARGRAPHIC와 호환 가능한 유형이 아닙니다. 길이가 서로 다른 CHAR 또는 VARCHAR는 호환되는 데이터 유형입니다.

- 같은 DECFLOAT 값은 정밀도가 다르더라도 동일하게 취급됩니다. 숫자적으로 같은 DECFLOAT 값은 유효 숫자 수가 다르더라도 동일하게 취급됩니다.
- 분산 키의 일부로 지원되지 않는 데이터 유형은 데이터베이스 파티션 호환성에 적용할 수 없습니다. 여기에는 데이터 유형이 BLOB, CLOB, DBCLOB, XML, 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형인 컬럼이 포함됩니다.

표 42. 데이터베이스 파티션 호환성

피연산자	2진 정수	10진수	부동 소수점	10진 부동 소수점	문자열	그래픽 문자열	날짜	시간	시간소인	구별 유형
2진 정수	예	없음	없음	없음	없음	없음	없음	없음	없음	[1]
10진수	없음	예	없음	없음	없음	없음	없음	없음	없음	[1]
부동 소수점	없음	없음	예	없음	없음	없음	없음	없음	없음	[1]
10진 부동 소수점	없음	없음	없음	예	없음	없음	없음	없음	없음	[1]
문자열	없음	없음	없음	없음	예[2]	2, 3	없음	없음	없음	[1]
그래픽 문자열	없음	없음	없음	없음	2, 3	예[2]	없음	없음	없음	[1]
날짜	없음	없음	없음	없음	없음	없음	예	없음	없음	[1]
시간	없음	없음	없음	없음	없음	없음	없음	예	없음	[1]
시간소인	없음	없음	없음	없음	없음	없음	없음	없음	예	[1]
구별 유형	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]

주:

[1] 구별 유형 값은 구별 유형의 소스 데이터 유형 또는 소스 데이터 유형이 같은 다른 구별 유형과 호환할 수 있는 데이터베이스 파티션입니다. 구별 유형의 소스 데이터 유형은 분산 키의 일부로 지원되는 데이터 유형이어야 합니다. 사용자 정의 구별 유형(UDT) 값은 데이터베이스 파티션 호환 소스 유형의 다른 UDT 또는 UDT의 소스 유형과 호환되는 파티션입니다. 구별 유형은 BLOB, CLOB, DBCLOB 또는 XML을 기반으로 할 수 없습니다.

[2] 문자 및 그래픽 문자열 유형은 호환 가능한 조합을 가진 경우 호환 가능합니다.

[3] 문자 및 그래픽 문자열 유형은 로케일 구분 UCA 기반 조합이 적용되는 경우 호환 가능합니다. 그렇지 않으면, 호환 가능한 유형이 아닙니다.

특수 레지스터

CURRENT DBPARTITIONNUM

CURRENT DBPARTITIONNUM 특수 레지스터는 명령문에 대한 코디네이터 노드 번호를 식별하는 INTEGER 값을 지정합니다. 응용프로그램에서 발행되는 명령문의 경우 코디네이터는 응용프로그램이 연결하는 데이터베이스 파티션입니다. 루틴에서 발행되는 명령문의 경우 코디네이터는 루틴이 호출되는 데이터베이스 파티션입니다.

루틴 내에서 SQL문에서 사용될 때 CURRENT DBPARTITIONNUM은 호출 명령문에서 상속되지 않습니다.

CURRENT DBPARTITIONNUM은 데이터베이스 인스턴스가 데이터베이스 파티셔닝을 지원하기 위해 정의되지 않은 경우 0을 리턴합니다. (즉, db2nodes.cfg 파일이 없는 경우입니다. 파티션된 데이터베이스의 경우 db2nodes.cfg 파일이 존재하며 데이터베이스 파티션 정의가 들어있습니다.)

CURRENT DBPARTITIONNUM은 CONNECT문을 통해 변경할 수 있지만 특정 조건에서만 가능합니다.

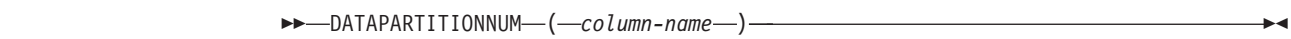
버전 8 이전의 버전과의 호환성을 위해 DBPARTITIONNUM 대신 NODE 키워드를 사용할 수 있습니다.

예: 호스트 변수 APPL_NODE(정수)를 응용프로그램이 연결되는 데이터베이스 파티션의 번호로 설정하십시오.

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

제 32 장 SQL 함수

DATAPARTITIONNUM



스키마는 SYSIBM입니다.

DATAPARTITIONNUM 함수는 행이 있는 데이터 파티션의 시퀀스 번호(SYSDATAPARTITIONS.SEQNO)를 리턴합니다. 데이터 파티션은 범위별로 정렬되며 시퀀스 번호는 0에서 시작됩니다. 예를 들어, DATAPARTITIONNUM 함수는 최하위 범위의 데이터 파티션에 있는 행에 대해 0을 리턴합니다.

인수는 테이블의 컬럼에 대해 규정된 이름 또는 규정되지 않은 이름일 수 있습니다. 행 레벨 정보가 리턴되므로 지정된 컬럼에 관계 없이 결과가 동일합니다. 컬럼의 데이터 유형에는 제한이 없습니다.

*column-name*에서 뷰의 컬럼을 참조할 경우, 뷰의 컬럼에 대한 표현식은 기본 테이블의 컬럼을 참조해야 하며 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다.

결과의 데이터 유형은 INTEGER이며 절대 널(NULL)이 될 수 없습니다.

이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 함수가 모든 데이터 유형을 인수로 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

DATAPARTITIONNUM 함수는 점검 제한조건 또는 생성된 컬럼에 정의에 사용할 수 없습니다(SQLSTATE 42881). DATAPARTITIONNUM 함수는 구체화된 쿼리 테이블(MQT) 정의에 사용할 수 없습니다(SQLSTATE 428EC).

예:

- ```
  SELECT DATAPARTITIONNUM (EMPNO)
     FROM EMPLOYEE
  ```

  DATAPARTITIONNUM이 리턴한 시퀀스 번호(예: 0)를 다른 SQL문(예: ALTER TABLE...DETACH PARTITION)에서 사용할 수 있는 데이터 파티션 이름으로 변환하기 위해 SYSCAT.DATAPARTITIONS 카탈로그 뷰를 쿼리할 수 있습니다. 다음 예에 표시된 대로 WHERE절의 DATAPARTITIONNUM에서 얻은 SEQNO를 포함시키십시오.
  ```

```
SELECT DATAPARTITIONNAME
FROM SYSCAT.DATAPARTITIONS
WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
```

결과 값은 'PART0'입니다.

DBPARTITIONNUM

스키마는 SYSIBM입니다.

DBPARTITIONNUM 함수는 행의 데이터베이스 파티션 번호를 리턴합니다. 예를 들어, SELECT절에 사용하는 경우 결과 세트에 있는 각 행의 데이터베이스 파티션 번호를 리턴합니다.

인수는 테이블의 컬럼에 대해 규정된 이름 또는 규정되지 않은 이름일 수 있습니다. 행 레벨 정보가 리턴되므로 지정된 컬럼에 관계 없이 결과가 동일합니다. 컬럼의 데이터 유형에는 제한이 없습니다.

*column-name*에서 뷰의 컬럼을 참조할 경우, 뷰의 컬럼에 대한 표현식은 기본 테이블의 컬럼을 참조해야 하며 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다.

데이터베이스 파티션 번호가 DBPARTITIONNUM 함수에 의해 리턴되는 특정 행 (및 테이블)은 함수를 사용하는 SQL문의 컨텍스트에서 결정됩니다.

전이 변수 및 테이블에서 리턴되는 데이터베이스 파티션 번호는 분산 키 컬럼의 현재 전이 값에서 파생됩니다. 예를 들어, 사전 삽입 트리거에서 새 전이 변수의 현재 값을 제공하면 이 함수는 프로젝트된 데이터베이스 파티션 번호를 리턴합니다. 그러나 분산 키 컬럼 값은 이후의 사전 삽입 트리거에서 수정될 수 있습니다. 따라서 데이터베이스에 삽입될 때 행의 최종 데이터베이스 파티션 번호는 프로젝트된 값과 다를 수 있습니다.

결과의 데이터 유형은 INTEGER이며 절대 널(NULL)이 될 수 없습니다. `db2nodes.cfg` 파일이 없는 경우 결과는 0입니다.

이 함수는 사용자 정의 함수를 작성할 때 소스 함수로 사용할 수 없습니다. 함수가 모든 데이터 유형을 인수로 허용하므로 사용자 정의 구별 유형을 지원하기 위해 추가 시그니처를 작성할 필요가 없습니다.

DBPARTITIONNUM 함수는 점검 제한조건 내의 복제된 테이블이나 생성된 컬럼의 정의에서 사용할 수 없습니다(SQLSTATE 42881).

이전 DB2 제품 버전과의 호환성을 위해 DBPARTITIONNUM 대신 NODENUMBER를 지정할 수 있습니다.

예를 들면, 다음과 같습니다.

- EMPLOYEE 테이블에 있는 제공된 직원의 행이 DEPARTMENT 테이블에 있는 직원 부서의 설명과 다른 데이터베이스 파티션에 있는 인스턴스 수를 계산하십시오.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- 두 테이블의 행이 동일한 데이터베이스 파티션에 있도록 EMPLOYEE 및 DEPARTMENT 테이블을 조인하십시오.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- EMPLOYEE 테이블의 BEFORE 트리거를 사용하여 이름이 EMPINSERTLOG1인 테이블에서 EMPLOYEE 테이블의 새 행에 대한 프로젝트된 데이터베이스 파티션 번호를 로그하십시오.

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

제 33 장 SQL문

ALTER DATABASE PARTITION GROUP

ALTER DATABASE PARTITION GROUP문은 다음을 수행하는 데 사용됩니다.

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

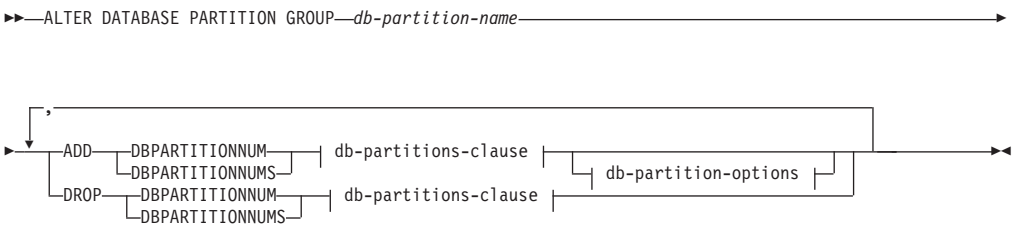
호출

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

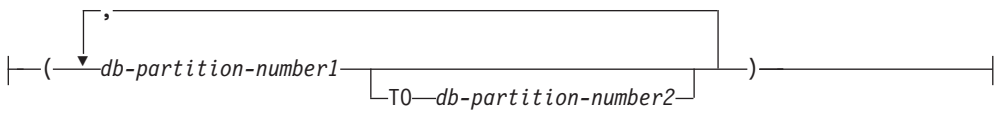
권한 부여

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

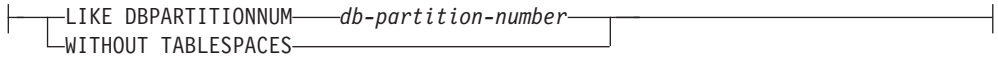
구문



db-partitions-clause:



db-partition-options:



설명

db-partition-name

데이터베이스 파티션 그룹의 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이

것은 SQL ID(일반 또는 분리 ID)입니다. 카탈로그에 기술된 데이터베이스 파티션 그룹이어야 합니다. IBMCATGROUP과 IBMTEMPGROUP은 지정할 수 없습니다(SQLSTATE 42832).

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

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

db-partitions-clause
데이터베이스 파티션 그룹에 추가 또는 삭제할 데이터베이스 파티션을 지정합니다.

db-partition-number1
특정 데이터베이스 파티션 번호를 지정하십시오.

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

db-partition-options

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

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

WITHOUT TABLESPACES

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

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

Rules

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

이스 파티션 서버가 온라인으로 인스턴스에 추가되고 모든 응용프로그램이 새 데이터베이스 파티션 서버를 인식하지 않는 경우, 이 명령문도 실패할 수 있습니다 (SQLSTATE 55077).

주

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

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

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

예 :

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

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

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

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

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

CREATE DATABASE PARTITION GROUP

CREATE DATABASE PARTITION GROUP문은 데이터베이스 내에 새로운 데이터베이스 파티션 그룹을 정의한 후 데이터베이스 파티션 그룹에 데이터베이스 파티션을 지정하고, 시스템 카탈로그에 데이터베이스 파티션 그룹 정의를 기록합니다.

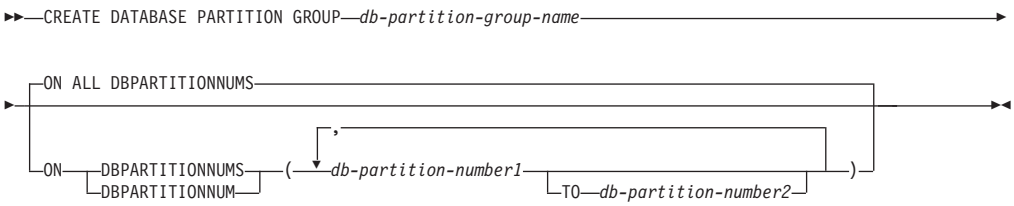
호출

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

권한 부여

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

구문



설명

db-partition-group-name

데이터베이스 파티션 그룹의 이름을 지정합니다. 이 이름은 한 부분의 이름으로, 이것은 SQL ID(일반 또는 분리 ID)입니다. *db-partition-group-name*은 카탈로그에 이미 존재하는 데이터베이스 파티션 그룹을 식별하면 안됩니다(SQLSTATE 42710). *db-partition-group-name*은 'SYS' 또는 'IBM'으로 시작하면 안됩니다(SQLSTATE 42939).

ON ALL DBPARTITIONNUMS

데이터베이스 파티션 그룹 작성 시 데이터베이스에 대해 정의된 모든 데이터베이스 파티션(db2nodes.cfg 파일)에 데이터베이스 파티션 그룹이 정의되도록 지정합니다.

데이터베이스 파티션이 데이터베이스 시스템에 추가되면, ALTER DATABASE PARTITION GROUP문은 데이터베이스 파티션 그룹(IBMDEFAULTGROUP 포함)에 새 데이터베이스 파티션을 포함하도록 실행되어야 합니다. 또한 REDISTRIBUTE DATABASE PARTITION GROUP 명령을 발행하여 데이터를 데이터베이스 파티션으로 이동시켜야 합니다.

ON DBPARTITIONNUMS

데이터베이스 파티션 그룹에 있는 데이터베이스 파티션을 지정합니다. DBPARTITIONNUM은 DBPARTITIONNUMS와 동의어입니다.

db-partition-number1

데이터베이스 파티션 번호를 지정합니다. (이전 버전과 호환되도록 NODE*nnnnn* 형식의 *node-name*을 지정할 수 있습니다.)

TO *db-partition-number2*

데이터베이스 파티션 번호의 범위를 지정하십시오. *db-partition-number2* 값은 *db-partition-number1* 값보다 크거나 같아야 합니다(SQLSTATE 428A9). 지정된 데이터베이스 파티션 번호 사이 및 번호를 포함하는 모든 데이터베이스 파티션은 데이터베이스 파티션 그룹에 포함됩니다.

Rules

- 번호 순으로 지정된 각 데이터베이스 파티션은 db2nodes.cfg 파일에 정의되어야 합니다(SQLSTATE 42729).
- ON DBPARTITIONNUMS절에 나열된 각 *db-partition-number*는 한 번만 표시되어야 합니다(SQLSTATE 42728).
- 유효한 *db-partition-number*는 0과 999 사이입니다(SQLSTATE 42729).
- 데이터베이스 파티션 서버 요청이 보류 또는 진행 중인 경우 CREATE DATABASE PARTITION GROUP문이 실패할 수 있습니다(SQLSTATE 55071). 새 데이터베이스 파티션 서버가 온라인으로 인스턴스에 추가되고 모든 응용프로그램이 새 데이터베이스 파티션 서버를 인식하지 않는 경우, 이 명령문도 실패할 수 있습니다(SQLSTATE 55077).

주

- 이 명령문은 데이터베이스 파티션 그룹에 대한 분산 맵을 작성합니다. 각 분산 맵에 대해 분산 맵 ID(PMAP_ID)가 생성됩니다. 이 정보는 카탈로그에 기록되며, SYSCAT.DBPARTITIONGROUPS와 SYSCAT.PARTITIONMAPS에서 검색할 수 있습니다. 분산 맵의 각 항목은 해시된 모든 행이 있는 목표 데이터베이스 파티션을 지정합니다. 단일 파티션 데이터베이스 파티션 그룹의 경우, 해당하는 분산 맵에는 단 한 개의 항목만 있습니다. 다중 파티션 데이터베이스 파티션 그룹의 경우, 해당하는 분산 맵의 32768 항목이 디폴트값이며 데이터베이스 파티션 번호가 라운드 로빈 방식으로 맵 항목에 할당됩니다.
- *호환성*: 이전 버전의 DB2 제품과의 호환성을 위해,
  - DBPARTITIONNUM 대신 NODE를 지정할 수 있습니다.
  - DBPARTITIONNUMS 대신 NODES를 지정할 수 있습니다.
  - DATABASE PARTITION GROUP 대신 NODEGROUP을 지정할 수 있습니다.

예:

0, 1, 2, 5, 7 및 8로 정의된 6개의 데이터베이스 파티션이 있는 파티션 데이터베이스가 있다고 가정하십시오.

- 여섯 개의 모든 데이터베이스 파티션에 MAXGROUP이라는 데이터베이스 파티션 그룹을 작성할 경우, 명령문은 다음과 같습니다.

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- 데이터베이스 파티션 0, 1, 2, 5 및 8에 MEDGROUP이라는 데이터베이스 파티션 그룹을 작성할 경우, 명령문은 다음과 같습니다.

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- 데이터베이스 파티션 7에 MINGROUP이라는 단일 파티션 데이터베이스 파티션 그룹을 작성할 경우, 명령문은 다음과 같습니다.

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

제 34 장 지원되는 관리 SQL 루틴 및 뷰

ADMIN_CMD 스토어드 프로시저 및 연관된 관리 SQL 루틴

ADMIN_CMD 프로시저를 사용한 GET STMM TUNING DBPARTITIONNUM 명령

사용자 선호 자체 성능 조정 메모리 관리자(STMM) 조정 데이터베이스 파티션 번호 및 현재 STMM 조정 데이터베이스 파티션 번호를 보고하기 위해 카탈로그 테이블을 읽는 데 사용됩니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권에는 다음과 같은 권한 또는 특권 중 하나 이상을 포함해야 합니다

- DBADM
- SECADM
- SQLADM
- ACCESSCTRL
- DATAACCESS
- SYSIBM.SYSTUNINGINFO의 SELECT

필수 연결

데이터베이스

명령 구문

예

```
CALL SYSPROC.ADMIN_CMD( 'get stmm tuning dbpartitionnum' )
```

이 쿼리의 출력 예는 다음과 같습니다.

```
결과 세트 1
  --------------

    USER_PREFERRED_NUMBER CURRENT_NUMBER
  --------------------- --------------
                                     2              2
```

```
  1 레코드가 선택됨.

Return Status = 0
```

사용 시 참고사항

사용자 선호 자체 성능 조정 메모리 관리자(STMM) 조정 데이터베이스 파티션 번호(USER_PREFERRED_NUMBER)는 사용자에 의해 설정되며 사용자가 메모리 조정 프로그램을 실행하려 하는 데이터베이스 파티션을 지정합니다. 데이터베이스가 실행되는 동안 조정 파티션은 한시간에 여러 번 비동기적으로 갱신됩니다. 따라서 사용자가 선호하는 STMM 파티션 번호를 갱신한 후 리턴된 CURRENT_NUMBER와 USER_PREFERRED_NUMBER는 동기화된 상태가 아닐 수 있습니다. 이를 해결하려면 CURRENT_NUMBER기 비동기적으로 갱신되기를 기다리거나 데이터베이스를 중지한 후 다시 시작하여 CURRENT_NUMBER를 강제로 갱신하십시오.

ADMIN_CMD 프로시저를 사용하는 UPDATE STMM TUNING DBPARTITIONNUM 명령

사용자 선호 자체 성능 조정 메모리 관리자(STMM) 조정 데이터베이스 파티션을 갱신하십시오.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권에는 다음과 같은 권한 중 하나 이상을 포함해야 합니다.

- DBADM
- DATAACCESS
- SQLADM

필수 연결

데이터베이스

명령 구문

```
►►—UPDATE—STMM—TUNING—DBPARTITIONNUM—partitionnum———————————————►◄
```

명령 매개변수

partitionnum

*partitionnum*dms 정수입니다. -1 또는 nonexistent 데이터베이스 파티션 수가 사용된 경우, DB2가 자동으로 적합한 데이터베이스 파티션을 선택하여 STMM 메모리 조정 프로그램을 실행합니다.

예

사용자 선호 자체 성능 조정 메모리 관리자(STMM) 조정 데이터베이스 파티션을 데이터베이스 파티션 3으로 갱신하십시오.

```
CALL SYSPROC.ADMIN_CMD( 'update stmm tuning dbpartitionnum 3' )
```

사용 시 참고사항

STMM 조정 프로세스는 사용자 선호 STMM 조정 데이터베이스 파티션 수 값의 변경사항을 정기적으로 점검합니다. *partitionnum*이 존재하고 활성 데이터베이스 파티션인 경우 STMM 조정 프로세스는 사용자 선호 STMM 조정 데이터베이스 파티션으로 이동합니다. 이 명령이 STMM 조정 데이터베이스 파티션 수를 한 번 변경하면 현재 STMM 조정 데이터베이스 파티션 수가 즉시 변경됩니다.

명령 실행 상태는 CALL문에서 생성된 SQLCA에 리턴됩니다.

이 명령은 ADMIN_CMD 프로시저에서 해당 변경사항을 커미트합니다.

구성 관리 SQL 루틴 및 뷰

DB_PARTITIONS

DB_PARTITIONS 테이블 함수는 db2nodes.cfg 파일의 컨텐츠를 테이블 형식으로 리턴합니다.

구문

스키마는 SYSPROC입니다.

권한 부여

DB_PARTITIONS 테이블 함수에 대한 EXECUTE 특권

테이블 함수 매개변수

함수에 입력 매개변수가 없습니다.

예

3개의 논리적 파티션을 갖는 데이터베이스에서 정보를 검색합니다.

```
SELECT * FROM TABLE(DB_PARTITIONS()) AS T
```

이 쿼리의 출력 예는 다음과 같습니다.

```
PARTITION_NUMBER HOST_NAME                PORT_NUMBER SWITCH_NAME
---------------- --------------------...- ----------- -----------
                               0 jessicae.torolab.ibm.com           0 jessicae
                               1 jessicae.torolab.ibm.com           1 jessicae
                               2 jessicae.torolab.ibm.com           2 jessicae

  3 record(s) selected.
```

리턴되는 정보

표 43. DB_PARTITIONS 테이블 함수에서 리턴한 정보

컬럼 이름	데이터 유형	설명
PARTITION_NUMBER	SMALLINT	파티션된 데이터베이스 환경에서 데이터베이스 파티션 서버를 식별하는 0에서 999 사이의 고유한 숫자
HOST_NAME	VARCHAR(128)	데이터베이스 파티션 서버의 TCP/IP 호스트 이름
PORT_NUMBER	SMALLINT	데이터베이스 파티션 서버의 포트 번호
SWITCH_NAME	VARCHAR(128)	데이터베이스 파티션 통신을 위한 고속 상호 접속 또는 스위치의 이름

Stepwise 재분배 관리 SQL 루틴

STEPWISE_REDISTRIBUTE_DBPG 프로시저 - 데이터베이스 파티션 그룹의 일부 재분배

STEPWISE_REDISTRIBUTE_DBPG 프로시저는 이 프로시저에 지정된 입력 및 SET_SWRD_SETTINGS 프로시저에 의해 작성되거나 갱신된 설정 파일에 따라 데이터베이스 파티션 그룹의 일부를 재분배합니다.

구문

스키마는 SYSPROC입니다.

프로시저 매개변수

inDBPGroup

목표 데이터베이스 파티션 그룹의 이름을 지정하는 VARCHAR(128) 유형의 입력 인수.

inStartingPoint

사용할 시작 지점을 지정하는 SMALLINT 유형의 입력 인수. 매개변수가 NULL이 아닌 양수로 설정될 경우, STEPWISE_REDISTRIBUTE_DBPG 프로시저는 설정 파일에 지정된 *nextStep* 값을 사용하는 대신 이 값을 사용합니다. 이 옵션은 STEPWISE_REDISTRIBUTE_DBPG 프로시저를 특정 단계부터 재실행할 때 유용합니다. 매개변수가 NULL로 설정될 경우, *nextStep* 값이 사용됩니다.

inNumSteps

실행할 단계 수를 지정하는 SMALLINT 유형의 입력 인수. 매개변수가 NULL이 아닌 양수로 설정될 경우, STEPWISE_REDISTRIBUTE_DBPG 프로시저는 설정 파일에 지정된 *stageSize* 값을 사용하는 대신 이 값을 사용합니다. 이 옵션은 설정값에 지정된 것과 다른 단계 수로 STEPWISE_REDISTRIBUTE_DBPG 프로시저를 재실행할 때 유용합니다. 예를 들어, 스케줄된 단계에 5개의 단계가 있으며 재분배 프로세스가 3단계에서 실패한 경우, 오류 조건을 정정한 후 나머지 세 단계를 실행하도록 STEPWISE_REDISTRIBUTE_DBPG 프로시저를 호출할 수 있습니다. 매개변수가 NULL로 설정될 경우, *stageSize* 값이 사용됩니다. 숫자가 무제한인 것을 나타내기 위해 이 프로시저에서 -2값을 사용할 수 있습니다.

주: REDISTRIBUTE DATABASE PARTITION GROUP 명령의 **NOT ROLLFORWARD RECOVERABLE** 옵션과 동등한 옵션을 지정할 매개변수가 없습니다. 로깅은 STEPWISE_REDISTRIBUTE_DBPG 프로시저를 사용할 때 수행되는 행 데이터 재분배에 항상 수행됩니다.

권한 부여

- STEPWISE_REDISTRIBUTE_DBPG 프로시저에 대한 EXECUTE 특권
- SYSADM, SYSCTRL 또는 DBADM

예

SET_SWRD_SETTINGS 프로시저에 의해 레지스트리에 저장된 재분배 플랜에 따라 데이터베이스 파티션 그룹 "IBMDEFAULTGROUP"을 재분배하십시오. 3단계에서 시작하며 재분배 계획에서 두 단계가 완료될 때까지 데이터를 재분배합니다.

```
CALL SYSPROC.STEPWISE_REDISTRIBUTE_DBPG('IBMDEFAULTGROUP', 3, 2)
```

사용 시 참고사항

STEPWISE_REDISTRIBUTE_DBPG 프로시저 실행이 시작된 후 SET_SWRD_SETTINGS 프로시저를 사용하여 *processState*에 대한 레지스트리 값을 1로 갱신한 경우, 이 프로세스는 다음 단계가 시작될 때 중지되고 경고 메시지가 리턴됩니다.

재분배 프로세스를 통해 SQL COMMIT문을 호출한 후에는 유형 2 연결에서 재분배 프로세스를 실행하는 것이 지원되지 않습니다.

제 6 부 부록

부록 A. 루트 서버가 아닌 사용자로 설치

DB2 제품을 비루트 사용자로 설치

대부분의 DB2 데이터베이스 제품을 비루트 사용자로 설치할 수 있습니다.

시작하기 전에

DB2 데이터베이스 제품을 비루트 사용자로 설치하기 전에 루트 설치와 비루트 설치의 차이점과 비루트 설치의 제한사항을 알아야 합니다. 비루트 설치에 관한 자세한 정보는 『비루트 설치의 개요(Linux 및 UNIX)』를 참조하십시오.

비루트 사용자로 DB2 데이터베이스 제품을 설치하기 위한 전제조건은 다음과 같습니다.

- 설치 DVD를 마운트할 수 있어야 하거나 또는 마운트된 설치 DVD가 있어야 합니다.
- DB2 인스턴스의 소유자로 사용할 수 있는 유효한 사용자 ID가 있어야 합니다.

  사용자 ID의 제한사항 및 요구사항은 다음과 같습니다.
  - guests, admins, users 및 local 이외의 기본 그룹이 있어야 함
  - 소문자 글자(a–z), 숫자(0–9) 및 밑줄 문자(_)를 사용할 수 있어야 함
  - 8자 미만이어야 함
  - IBM, SYS, SQL 또는 숫자로 시작할 수 없음
  - DB2 예약어(USERS, ADMINS, GUESTS, PUBLIC 또는 LOCAL) 또는 SQL 예약어일 수 없음
  - DB2 인스턴스 ID, DAS ID 또는 분리 ID에 대한 루트 특권을 가진 모든 사용자 ID를 사용할 수 없음
  - 강조 문자를 포함할 수 없음
  - 새 사용자 ID를 작성하는 대신 기존 사용자 ID를 지정한 경우에는 사용자 ID에서 다음 사항을 확인하십시오.
    - 사용자 ID가 잠겨 있지 않은지
    - 만기된 암호가 있지 않은지
- 설치 중인 제품에 존재하는 하드웨어 및 소프트웨어 전제조건은 루트 사용자와 마찬가지로 비루트 사용자에게 적용됩니다.
- AIX 버전 5.3에서 비동기 입출력(AIO)이 사용 가능해야 합니다. 시스템에서 IOCP(I/O Completion Port)를 사용 가능하도록 설정하십시오.
- 홈 디렉토리는 유효한 DB2 경로여야 합니다.

DB2 설치 경로의 규칙은 다음과 같습니다.

- 소문자 글자(a–z), 대문자 글자(A–Z) 및 밑줄 문자(_)를 사용할 수 있어야 함
- 128자 미만이어야 함
- 스페이스를 포함할 수 없음
- 영어가 아닌 문자를 포함할 수 없음

이 태스크에 대한 정보

DB2 데이터베이스 제품을 비루트 사용자로 설치하는 것은 비루트 사용자에게 쉬운 작업입니다. 즉, 비루트 사용자가 비루트 사용자로서 로그인하는 것 외에는 DB2 데이터베이스 제품을 설치하기 위해 수행해야 하는 특별한 사항이 없습니다.

프로시저

비루트 설치를 수행하려면 다음을 수행하십시오.

1. 비루트 사용자로 로그인하십시오.
2. 사용 가능한 방법을 사용하여 DB2 데이터베이스 제품을 설치하십시오. 옵션은 다음과 같습니다.
   - DB2 설치 마법사(GUI 설치)
   - db2_install 명령
   - 응답 파일이 있는 db2setup 명령(자동 설치)

     주: 비루트 사용자는 DB2 데이터베이스 제품이 설치된 디렉토리를 선택할 수 없으므로 응답 파일에서 **FILE** 키워드는 무시됩니다.
3. DB2 데이터베이스 제품을 설치한 후에 비루트 DB2 인스턴스를 사용하려면 새 로그인 세션을 열어야 합니다. 또한 DB2 인스턴스 환경에 *$HOME*/sqllib/db2profile(본 쉘 및 콘 쉘 사용자용) 또는 *$HOME*/sqllib/db2chsrc(C 쉘 사용자용)을 설정하는 경우에는 동일한 로그인 세션을 사용할 수도 있습니다. 여기서 *$HOME*은 비루트 사용자의 홈 디렉토리입니다.

다음 단계

DB2 데이터베이스 제품을 설치한 후에 운영 체제 사용자 프로세스 자원 한계(ulimits)를 확인하십시오. 최소 ulimit 값에 부합하지 않으면 DB2 엔진에는 운영 자원 부족 오류가 예기치 않게 발생할 수 있습니다. 이러한 오류로 인해 DB2 데이터베이스 시스템 정지가 발생할 수 있습니다.

부록 B. 백업 사용

백업 사용

데이터베이스 데이터를 복사하고 실패 또는 원본에 대한 손상의 경우에 다른 매체에 저장하려면 BACKUP DATABASE 명령을 사용하십시오. 전체 데이터베이스, 데이터베이스 파티션 또는 선택된 테이블 스페이스만 백업할 수 있습니다.

시작하기 전에

백업될 데이터베이스에 연결될 필요는 없습니다. 데이터베이스 백업 유틸리티가 자동으로 지정된 데이터베이스에 연결하고 이 연결은 백업 작업이 완료될 때 종료됩니다. 백업될 데이터베이스에 연결된 경우 BACKUP DATABASE 명령이 실행될 때 연결이 끊어지며 백업 작업은 계속됩니다.

데이터베이스는 로컬 또는 리모트일 수 있습니다. Tivoli Storage Manager(TSM) 또는 DB2 ACS(Advanced Copy Services) 같은 스토리지 관리 제품을 사용 중이 아니면 백업 이미지는 데이터베이스 서버에 남아 있습니다.

오프라인 백업을 수행 중인 경우 및 ACTIVATE DATABASE문을 사용하여 데이터베이스를 활성화한 경우, 오프라인 백업을 실행하기 전에 데이터베이스를 비활성화해야 합니다. 데이터베이스에 대한 사용 중인 연결이 있는 경우 데이터베이스를 비활성화하려면 SYSADM 권한이 있는 사용자가 데이터베이스에 연결하고 다음 명령을 실행해야 합니다.

```
CONNECT TO database-alias
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE database-alias
```

파티션된 데이터베이스 환경에서는 BACKUP DATABASE 명령을 사용하여 데이터베이스 파티션을 개별적으로 백업하거나, ON DBPARTITIONNUM 명령 매개변수를 사용하여 한 번에 여러 데이터베이스 파티션을 백업하거나 ALL DBPARTITIONNUMS 매개변수를 사용하여 모든 데이터베이스 파티션을 동시에 백업할 수 있습니다. LIST NODES 명령을 사용하여 백업하려는 파티션에 사용자 테이블이 있는 데이터베이스 파티션을 식별할 수 있습니다.

단일 시스템 뷰(SSV) 백업을 사용 중이 아니면, 파티션된 데이터베이스 환경에서 *오프라인* 백업을 수행하려는 경우 다른 모든 데이터베이스 파티션에서 카탈로그 파티션을 별도로 백업해야 합니다. 예를 들어 카탈로그 파티션을 먼저 백업한 후 다른 데이터베이스 파티션을 백업할 수 있습니다. 이는 백업 작업에서 카탈로그 파티션에 대한 독점 데

이터베이스 연결이 필요하여 이 동안에는 다른 데이터베이스 파티션이 연결할 수 없기 때문에 필요합니다. *온라인* 백업을 수행 중인 경우 모든 데이터베이스 파티션(카탈로그 파티션 포함)을 동시에 또는 임의의 순서로 백업할 수 있습니다.

또한 명령 편집기를 사용하여 데이터베이스 파티션을 백업할 수도 있습니다. 이 방식은 롤 포워드 복구를 지원하지 않기 때문에 이러한 노드에 있는 데이터베이스를 정기적으로 백업해야 합니다. `db2nodes.cfg` 파일에 대한 가능한 손상에 대한 보호 장치로서, 작성하는 모든 백업 사본과 함께 이 파일의 사본을 보존해야 합니다.

분산 요청(DR) 시스템에서 백업 작업은 분산 요청(DR) 데이터베이스 및 데이터베이스 카탈로그에 저장되는 메타데이터(랩퍼, 서버, 별명 등)에 적용됩니다. 데이터 소스 오브젝트(테이블 및 뷰)는 분산 요청(DR) 데이터베이스에 저장되지 않으면 백업되지 않습니다.

데이터베이스가 데이터베이스 관리 프로그램의 이전 릴리스를 사용하여 작성되었고 데이터베이스가 업그레이드되지 않은 경우 데이터베이스를 백업하기 전에 업그레이드해야 합니다.

이 태스크에 대한 정보

다음 제한사항이 백업 유틸리티에 적용됩니다.

- 테이블 스페이스 백업 작업 및 테이블 스페이스 리스토어 작업은 서로 다른 테이블 스페이스가 관련되는 경우에도 동시에 실행될 수 없습니다.
- 파티션된 데이터베이스 환경에서 롤 포워드 복구를 수행할 수 있기 원하는 경우, 노드 목록에 있는 데이터베이스를 정기적으로 백업해야 하며 시스템에 있는 나머지 노드(해당 데이터베이스에 대한 사용자 데이터를 포함하지 않는 것까지도)의 백업 이미지가 최소한 하나 있어야 합니다. 두 상황은 데이터베이스에 대한 사용자 데이터를 포함하지 않는 데이터베이스 파티션 서버에서 데이터베이스 파티션의 백업 이미지가 필요합니다.
  - 마지막 백업을 작성한 후 데이터베이스 시스템에 데이터베이스 파티션 서버를 추가했고, 이 데이터베이스 파티션 서버에서 포워드 복구를 수행해야 합니다.
  - 특정 시점 복구가 사용되는데, 이 복구는 시스템의 모든 데이터베이스 파티션이 롤 포워드 보류 상태에 있어야 합니다.
- DMS 테이블 스페이스의 온라인 백업 작업은 다음 작업과 호환되지 않습니다.
  - 로드
  - 재구성(온라인 및 오프라인)
  - 테이블 스페이스 삭제
  - 테이블 절단
  - 인덱스 작성
  - 처음에 로그되지 않음(CREATE TABLE 및 ALTER TABLE문과 함께 사용됨)

- 현재 사용 중인 데이터베이스의 오프라인 백업을 수행하려고 시도하면 오류가 수신됩니다. 오프라인 백업을 실행하기 전에 DEACTIVATE DATABASE 명령을 실행하여 데이터베이스가 활성이 아닌지 확인할 수 있습니다.

명령행 처리기(CLP), 제어 센터의 데이터베이스 백업 마법사를 통해서, BACKUP DATABASE 매개변수가 있는 ADMIN_CMD 프로시저나 *db2Backup* API를 실행하여 백업 유틸리티를 호출할 수 있습니다.

다음은 CLP를 통해 실행되는 BACKUP DATABASE 명령의 예입니다.

```
db2 backup database sample to c:¥DB2Backups
```

프로시저

데이터베이스 백업 마법사를 열려면 다음을 수행하십시오.

1. 제어 센터에서 백업하려는 데이터베이스나 테이블 스페이스 오브젝트를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. 오브젝트를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 백업을 선택하십시오. 데이터베이스 백업 마법사가 열립니다. .

다음 단계

제어 센터에 있는 문맥 도움말 기능을 통해 세부사항 정보가 제공됩니다.

오프라인 백업을 수행한 경우, 백업이 완료된 후 데이터베이스를 재활성화해야 합니다.

```
ACTIVATE DATABASE sample
```

부록 C. 파티션된 데이터베이스 환경 카탈로그 뷰

SYSCAT.BUFFERPOOLDBPARTITIONS

각 행은 버퍼 풀과 데이터베이스 파티션의 조합을 나타내며 해당 파티션의 버퍼 풀 크기는 SYSCAT.BUFFERPOOLS에 표시된 동일한 데이터베이스 파티션 그룹에 있는 다른 파티션의 디폴트 크기와 다릅니다.

표 44. SYSCAT.BUFFERPOOLDBPARTITIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
BUFFERPOOLID	INTEGER		내부 버퍼 풀 ID.
DBPARTITIONNUM	SMALLINT		데이터베이스 파티션 번호.
NPAGES	INTEGER		이 데이터베이스 파티션의 이 버퍼 풀의 페이지 수.

SYSCAT.DATAPARTITIONEXPRESSION

각 행은 테이블 파티셔닝 키의 해당 파트에 대한 표현식을 나타냅니다.

표 45. SYSCAT.DATAPARTITIONEXPRESSION 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
TABSCHEMA	VARCHAR(128)		파티션된 테이블의 스키마 이름.
TABNAME	VARCHAR(128)		파티션된 테이블의 규정되지 않은 이름.
DATAPARTITIONKEYSEQ	INTEGER		1에서 시작하는 표현식 키 파트 시퀀스.
DATAPARTITIONEXPRESSION	CLOB(32K)		SQL 구문의 시퀀스에서 이 항목에 대한 표현식.
NULLSFIRST	CHAR(1)		• N = 이 표현식의 널(NULL) 값이 비교적 높음 • Y = 이 표현식의 널(NULL) 값이 비교적 낮음

SYSCAT.DATAPARTITIONS

각 행은 데이터 파티션을 나타냅니다. 메모:

- 데이터 파티션 통계는 테이블이 여러 데이터베이스 파티션에 작성된 경우 하나의 데이터베이스 파티션을 나타냅니다.

표 46. SYSCAT.DATAPARTITIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
DATAPARTITIONNAME	VARCHAR(128)		데이터 파티션 이름.
TABSCHEMA	VARCHAR(128)		이 데이터 파티션이 속하는 테이블의 스키마 이름.

표 46. SYSCAT.DATAPARTITIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
TABNAME	VARCHAR(128)		이 데이터 파티션이 속하는 테이블의 규정되지 않은 이름.
DATAPARTITIONID	INTEGER		데이터 파티션 ID.
TBSPACEID	INTEGER	Y	이 데이터 파티션이 저장되는 테이블 스페이스의 ID입니다. STATUS가 'I'일 때는 널(NULL) 값입니다.
PARTITIONOBJECTID	INTEGER	Y	테이블 스페이스 내의 데이터 파티션 ID.
LONG_TBSPACEID	INTEGER	Y	Long 데이터가 저장된 테이블 스페이스 ID. STATUS가 'I'일 때는 널(NULL) 값입니다.
ACCESS_MODE	CHAR(1)		데이터 파티션의 액세스 제한 상태입니다. 이러한 상태는 무결성 설정 보류 상태에 있는 오브젝트나 SET INTEGRITY문으로 처리된 오브젝트에만 적용됩니다. 가능한 값은 다음과 같습니다. • D = 데이터 이동 없음 • F = 전체 액세스 • N = 권한 없음 • R = 읽기 전용 액세스
STATUS	VARCHAR(32)		• A = 데이터 파티션이 새로 접속됨 • D = 데이터 파티션이 접속 해제됨 • I = 카탈로그의 항목이 비동기 인덱스 정리 중에만 유지보수되는 접속 해제된 데이터 파티션. STATUS 값이 'I'인 행은 접속 해제된 파티션을 참조하는 모든 인덱스 레코드가 삭제되었을 때 제거됨 • 비어 있는 문자열 = 데이터 파티션을 볼 수 있음(정상 상태) 바이트 2 - 32는 나중에 사용하기 위해 예약됩니다.
SEQNO	INTEGER		데이터 파티션 시퀀스 번호(0에서 시작).
LOWINCLUSIVE	CHAR(1)		• N = 낮은 키 값이 포함되지 않음 • Y = 낮은 키 값이 포함됨
LOWVALUE	VARCHAR(512)		이 데이터 파티션에 대한 낮은 키 값(SQL 값의 문자열 표시).
HIGHINCLUSIVE	CHAR(1)		• N = 높은 키 값이 포함되지 않음 • Y = 높은 키 값이 포함됨
HIGHVALUE	VARCHAR(512)		이 데이터 파티션에 대한 높은 키 값(SQL 값의 문자열 표시).
CARD	BIGINT		데이터 파티션에 있는 총 행 수이며, 통계가 수집되지 않는 경우 -1입니다.
OVERFLOW	BIGINT		데이터 파티션에 있는 오버플로우 레코드의 총수이며, 통계가 수집되지 않는 경우 -1입니다.

표 46. SYSCAT.DATAPARTITIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
NPAGES	BIGINT		데이터 파티션의 행이 존재하는 페이지의 총수입니다. 통계가 수집되지 않는 경우 -1입니다.
FPAGES	BIGINT		데이터 파티션에 있는 페이지의 총수이며, 통계가 수집되지 않는 경우 -1입니다.
ACTIVE_BLOCKS	BIGINT		데이터 파티션에 있는 활성 블록의 총수 또는 -1입니다. 다차원적으로 클러스터된(MDC) 테이블에만 적용됩니다.
INDEX_TBSPACEID	INTEGER		이 데이터 파티션에 대한 모든 파티션된 인덱스를 보유하는 테이블 스페이스의 ID.
AVGROWSIZE	SMALLINT		이 데이터 파티션에 있는 압축 및 압축되지 않은 행 모두의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
PCTROWSCOMPRESSED	REAL		데이터 파티션에 있는 전체 행 수의 백분율로 표현되는 압축된 행입니다. 통계가 수집되지 않는 경우 -1입니다.
PCTPAGESAVED	SMALLINT		행 압축의 결과로 데이터 파티션에 저장되는 페이지의 대략적인 백분율입니다. 이 값에는 데이터 파티션의 각 사용자 데이터에 대한 오버헤드 바이트가 포함되지만 사전 오버헤드가 이용하는 스페이스는 포함되지 않습니다. 통계가 수집되지 않는 경우 -1입니다.
AVGCOMPRESSEDROWSIZE	SMALLINT		이 데이터 파티션에 있는 압축된 행의 평균 길이(바이트)이며, 통계가 수집되지 않는 경우 -1입니다.
AVGROWCOMPRESSIONRATIO	REAL		데이터 파티션의 압축된 행의 경우 이는 행별 평균 압축비입니다. 즉, 평균 압축되지 않은 행 길이를 평균 압축 행 길이로 나눈 값입니다. 통계가 수집되지 않는 경우 -1입니다.
STATS_TIME	TIMESTAMP	Y	이 오브젝트에 대해 기록된 통계가 마지막으로 변경된 시간입니다. 통계가 수집되지 않은 경우 널(NULL)입니다.
LASTUSED	DATE		나중에 사용하도록 예약됩니다.

SYSCAT.DBPARTITIONGROUPDEF

각 행은 데이터베이스 파티션 그룹에 포함되는 데이터베이스 파티션을 나타냅니다.

표 47. SYSCAT.DBPARTITIONGROUPDEF 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
DBPGNAME	VARCHAR(128)		데이터베이스 파티션을 포함하는 데이터베이스 파티션 그룹의 이름.
DBPARTITIONNUM	SMALLINT		데이터베이스 파티션 그룹에 포함되는 데이터베이스 파티션의 파티션 번호입니다. 유효한 파티션 번호는 0 - 999입니다(경계 포함).

표 47. SYSCAT.DBPARTITIONGROUPDEF 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 값 가능	설명
IN_USE	CHAR(1)		데이터베이스 파티션의 상태. • A = 새로 추가된 데이터베이스 파티션이 분산 맵에 없지만, 데이터베이스 파티션 그룹의 테이블 스페이스에 대한 컨테이너가 작성됩니다. 데이터베이스 파티션 그룹 재분배 조작이 완료되었을 때 데이터베이스 파티션이 분산 맵에 추가됩니다. • D = 데이터베이스 파티션 그룹 재분배 조작이 완료될 때 데이터베이스 파티션이 삭제됩니다. • T = 새로 추가된 데이터베이스 파티션이 분산 맵에 없으므로 WITHOUT TABLESPACES절을 사용하여 추가되었습니다. 컨테이너는 데이터베이스 파티션 그룹의 테이블 스페이스에 추가되어야 합니다. • Y = 데이터베이스 파티션이 분산 맵에 있습니다.

SYSCAT.DBPARTITIONGROUPS

각 행은 데이터베이스 파티션 그룹을 나타냅니다.

표 48. SYSCAT.DBPARTITIONGROUPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
DBPGNAME	VARCHAR(128)		데이터베이스 파티션 그룹 이름
OWNER	VARCHAR(128)		데이터베이스 파티션 그룹 소유자의 권한 부여 ID
OWNERTYPE	CHAR(1)		• S = 소유자가 시스템임 • U = 소유자가 개별 사용자임
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS 카탈로그 뷰의 분산 맵에 대한 ID.
REDISTRIBUTE_PMAP_ID	SMALLINT		현재 재분배에 사용되고 있는 분산 맵에 대한 ID입니다. 재분배가 현재 진행 중이 아닌 경우 -1입니다.
CREATE_TIME	TIMESTAMP		데이터베이스 파티션 그룹의 작성 시간.
DEFINER[1]	VARCHAR(128)		데이터베이스 파티션 그룹 소유자의 권한 부여 ID
REMARKS	VARCHAR(254)	Y	사용자가 제공하는 주석 또는 널(NULL) 값입니다.

주:

1. 이전 버전과의 호환성을 위해 DEFINER 컬럼이 포함됩니다. OWNER를 참조하십시오.

SYSCAT.PARTITIONMAPS

각 행은 테이블의 분산 해싱을 기초로 데이터베이스 파티션 그룹의 데이터베이스 파티션 사이에 테이블의 행을 분배하는 데 사용되는 분산 맵을 나타냅니다.

표 49. SYSCAT.PARTITIONMAPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 값 가능	설명
PMAP_ID	SMALLINT		분산 맵의 ID.
PARTITIONMAP	BLOB(65536)		다중 파티션 데이터베이스 파티션 그룹에 대한 32768 개의 2바이트 정수의 벡터인 분산 맵입니다. 단일 파티션 데이터베이스 파티션 그룹의 경우 단일 파티션의 파티션 번호가 지정된 하나의 항목이 있습니다.

부록 D. DB2 기술 정보 개요

DB2 기술 정보는 다음 도구 및 메소드를 통해 사용할 수 있습니다.

- DB2 정보 센터
  - 주제 항목(태스크, 개념 및 참조 항목)
  - DB2 도구에 대한 도움말
  - 샘플 프로그램
  - 자습서
- DB2 서적
  - PDF 파일(다운로드)
  - PDF 파일(DB2 PDF DVD)
  - 인쇄된 서적
- 명령행 도움말
  - 명령 도움말
  - 메시지 도움말

주: DB2 정보 센터 주제는 PDF 또는 하드카피 서적보다 자주 갱신됩니다. 최신 정보를 보려면 사용 가능한 문서 갱신사항을 설치하거나 ibm.com에서 DB2 정보 센터를 참조하십시오.

ibm.com에서 추가 DB2 기술 정보(예: 기술 노트, 백서 및 IBM Redbooks® 서적)를 온라인으로 액세스할 수 있습니다. DB2 정보 관리 라이브러리 소프트웨어 사이트 http://www.ibm.com/software/data/sw-library/에 액세스하십시오.

문서 피드백

DB2 문서에 대한 피드백을 환영합니다. DB2 문서를 향상시키는 방법에 대해서 제안사항이 있는 경우 db2docs@ca.ibm.com으로 전자 우편을 보내십시오. DB2 문서 팀에서는 고객의 모든 피드백을 읽지만 직접 응답할 수는 없습니다. 고객의 문제를 더 잘 이해할 수 있도록 가능한 위치에 특정 예를 제공해주십시오. 특정 주제 또는 도움말 파일에 대한 피드백을 보내실 경우, 제목 및 URL을 알려주십시오.

DB2 고객 지원에 문의할 때 이 전자 우편 주소를 사용하지 마십시오. 문서에서 해결할 수 없는 DB2 기술 문제점이 있는 경우, 해당 지역의 IBM 서비스 센터에 도움을 요청하십시오.

DB2 기술 라이브러리(하드카피 또는 PDF 형식)

다음 표는 IBM Publications Center(www.ibm.com/shop/publications/order)에서 사용할 수 있는 DB2 라이브러리에 대해 설명합니다. PDF 형식의 영문 DB2 버전 9.7 매뉴얼 및 번역된 버전은 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947에서 다운로드할 수 있습니다.

표에 인쇄할 수 있는 책이 나와 있는 경우에도, 사용 국가 또는 지역에서 해당 책을 사용할 수 없을 수도 있습니다.

매뉴얼이 갱신될 때마다 문서 번호가 증가합니다. 다음 사항을 참조하여 읽고 있는 매뉴얼이 최신 버전인지 확인하십시오.

주: *DB2 정보 센터*는 PDF 또는 하드카피 서적보다 자주 갱신됩니다.

표 50. DB2 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
관리 API 참조서	SA30-3958-00	예	2009년 8월
관리 루틴 및 뷰	SA30-3955-00	아니오	2009년 8월
Call Level Interface Guide and Reference, Volume 1	SC27-2437-00	예	2009년 8월
Call Level Interface Guide and Reference, Volume 2	SC27-2438-00	예	2009년 8월
명령어 참조서	SA30-3959-00	예	2009년 8월
데이터 이동 유틸리티 안내서 및 참조서	SA30-3969-00	예	2009년 8월
데이터 복구 및 고가용성 안내서 및 참조서	SA30-3970-00	예	2009년 8월
데이터베이스 관리 개념 및 구성 참조서	SA30-3951-00	예	2009년 8월
데이터베이스 모니터링 안내서 및 참조서	SA30-3953-00	예	2009년 8월
데이터베이스 보안 안내서	SA30-3971-00	예	2009년 8월
DB2 Text Search Guide	SC27-2459-00	예	2009년 8월
Developing ADO.NET and OLE DB Applications	SC27-2444-00	예	2009년 8월
Developing Embedded SQL Applications	SC27-2445-00	예	2009년 8월
Developing Java Applications	SC27-2446-00	예	2009년 8월
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-2447-00	아니오	2009년 8월

표 50. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Developing User-defined Routines(SQL and External)	SC27-2448-00	예	2009년 8월
Getting Started with Database Application Development	GI11-9410-00	예	2009년 8월
Linux 및 Windows에서 DB2 설치 및 관리 시작하기	GA30-3960-00	예	2009년 8월
자국어 안내서	SA30-3972-00	예	2009년 8월
DB2 Server 설치	GA30-3962-00	예	2009년 8월
IBM Data Server Client 설치	GA30-3963-00	아니오	2009년 8월
Message Reference Volume 1	SC27-2450-00	아니오	2009년 8월
Message Reference Volume 2	SC27-2451-00	아니오	2009년 8월
Net Search Extender Administration and User's Guide	SC27-2469-00	아니오	2009년 8월
파티셔닝 및 클러스터링 안내서	SA30-3973-00	예	2009년 8월
pureXML Guide	SC27-2465-00	예	2009년 8월
Query Patroller 관리 및 사용자 안내서	SA30-3974-00	아니오	2009년 8월
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC27-2468-00	아니오	2009년 8월
SQL Procedural Languages: Application Enablement and Support	SC27-2470-00	예	2009년 8월
SQL 참조서, 볼륨 1	SA30-3956-00	예	2009년 8월
SQL 참조서, 볼륨 2	SA30-3957-00	예	2009년 8월
문제점 해결 및 데이터베이스 성능 조정	SA30-3952-00	예	2009년 8월
DB2 버전 9.7로 업그레이드	SA30-3961-00	예	2009년 8월
Visual Explain 자습서	SA30-3968-00	아니오	2009년 8월
DB2 버전 9.7의 새로운 내용	SA30-3967-00	예	2009년 8월
Workload Manager Guide and Reference	SC27-2464-00	예	2009년 8월

표 50. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
XQuery Reference	SC27-2466-00	아니오	2009년 8월

표 51. DB2 Connect 특정 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
DB2 Connect Personal Edition 설치 및 구성	SA30-3965-00	예	2009년 8월
DB2 Connect Server 설치 및 구성	SA30-3966-00	예	2009년 8월
DB2 Connect 사용자 안내서	SA30-3964-00	예	2009년 8월

표 52. Information Integration 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	예	2009년 8월
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	예	2009년 8월
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	아니오	2009년 8월
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	예	2009년 8월
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	예	2009년 8월

인쇄된 DB2 서적 주문

인쇄된 DB2 서적이 필요한 경우, 대부분 온라인으로 구매할 수 있으나 모든 국가 또는 지역에 해당되지는 않습니다. 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. *DB2 PDF 문서* DVD의 일부 소프트카피 서적은 인쇄할 수 없다는 점에 유의하십시오. 예를 들면, *DB2 메시지 참조서*의 어떤 볼륨도 인쇄된 서적으로 사용할 수 없습니다.

DB2 PDF 문서 DVD에서 사용할 수 있는 다수의 DB2 서적의 인쇄된 버전은 IBM에서 유료로 주문할 수 있습니다. 주문하는 위치에 따라 IBM Publications Center에서 온라인으로 서적을 주문할 수도 있습니다. 해당 국가 또는 지역에서 온라인 주문이

불가능하면, 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. DB2 PDF 문서 DVD의 모든 서적을 인쇄할 수는 없다는 점에 유의하십시오.

주: 가장 최신 및 완료된 DB2 문서는 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7의 DB2 정보 센터에서 유지보수됩니다.

인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.

- 해당 국가 또는 지역에서 인쇄된 DB2 서적을 온라인으로 주문할 수 있는지 여부를 확인하려면 http://www.ibm.com/shop/publications/order의 IBM Publications Center를 확인하십시오. 서적 주문 정보에 액세스하려면 국가/지역/언어를 선택한 다음 해당 위치에서 주문 지시사항을 따르십시오.
- 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.
  1. 다음 웹 사이트 중 하나에서 해당 지역 담당자에 대한 문의처 정보를 찾으십시오.
     - www.ibm.com/planetwide에 있는 IBM 월드 와이드 문의처 디렉토리
     - http://www.ibm.com/shop/publications/order의 IBM Publications 웹 사이트. 사용 지역의 해당 서적 홈 페이지에 액세스하려면 해당 국가, 지역 또는 언어를 선택해야 합니다. 이 페이지에서 "이 제품의 정보" 링크를 수행하십시오.
  2. 전화로 주문할 경우, 주문할 DB2 서적을 지정하십시오.
  3. 담당자에게 주문하려는 서적의 제목 및 문서 번호를 제공하십시오. 서적의 제목 및 문서 번호는 498 페이지의 『DB2 기술 라이브러리(하드카피 또는 PDF 형식)』를 참조하십시오.

명령행 처리기에서 SQL 상태 도움말 표시

DB2 제품은 SQL문의 결과로 나타나는 상태에 대한 SQLSTATE 값을 리턴합니다. SQLSTATE 도움말은 SQL 상태 및 SQL 상태 클래스 코드의 의미를 설명합니다.

SQL 상태 도움말을 시작하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? sqlstate or ? class code
```

여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자를 나타냅니다.
예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

DB2 정보 센터의 다른 버전에 액세스

DB2 버전 9.7 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/입니다.

DB2 버전 9.5 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/입니다.

DB2 버전 9 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9/입니다.

DB2 버전 8 주제에 대한 버전 8 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v8/입니다.

DB2 정보 센터에서 원하는 언어로 항목 표시

DB2 정보 센터는 브라우저 환경 설정에 지정된 언어로 주제 항목을 표시합니다. 주제가 원하는 언어로 변환되지 않은 경우, DB2 정보 센터는 영어로 주제 항목을 표시합니다.

- Internet Explorer 브라우저에서 원하는 언어로 항목을 표시하려면 다음을 수행하십시오.
  1. Internet Explorer에서 **도구 —> 인터넷 옵션 —> 언어...** 단추를 누르십시오. 언어 환경 설정 창이 열립니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 누르십시오.

       주: 언어를 추가했다고 컴퓨터에 원하는 언어로 항목을 표시하는 데 필요한 글꼴이 설치되는 것은 아닙니다.
     - 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
  3. 브라우저 캐시를 지운 후 페이지를 새로 고쳐 원하는 언어로 DB2 정보 센터를 표시하십시오.
- Firefox 또는 Mozilla 브라우저에서 원하는 언어로 주제 항목을 표시하려면 다음을 수행하십시오.
  1. **도구 —> 옵션 —> 고급** 대화 상자의 **언어** 섹션에서 단추를 선택하십시오. 환경 설정 창에 언어 패널이 표시됩니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 눌러 언어 추가 창에서 언어를 선택합니다.

– 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.

3. 브라우저 캐시를 지운 후 페이지를 새로 고쳐 원하는 언어로 DB2 정보 센터를 표시하십시오.

일부 브라우저 및 운영 체제 조합에서 운영 체제의 국가별 설정을 선택한 로케일 및 언어로 변경해야 합니다.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신

로컬로 설치된 DB2 정보 센터는 주기적으로 갱신해야 합니다.

시작하기 전에

DB2 버전 9.7 정보 센터는 등록된 상태여야 합니다. 자세한 내용은 *DB2 Server 설치*의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치』 주제를 참조하십시오. 정보 센터 설치에 적용되는 모든 전제조건 및 제한사항은 정보 센터 갱신에도 적용됩니다.

이 태스크에 대한 정보

기존의 DB2 정보 센터는 자동 또는 수동으로 갱신할 수 있습니다.

- 자동 갱신 - 기존 정보 센터 기능 및 언어를 갱신합니다. 자동 갱신의 또 다른 이점으로는 갱신 동안 정보 센터를 아주 잠시 동안만 사용 불가능하다는 점입니다. 또한 자동 갱신은 주기적으로 실행되는 기타 일괄처리 작업의 일부로 실행되도록 설정 가능합니다.
- 수동 갱신 - 갱신 프로세스 중에 기능이나 언어를 추가하려는 경우 사용하십시오. 예를 들어, 로컬 정보 센터는 기본적으로 영어와 프랑스어로 설치되어 있으며 기존 정보 센터의 기능 및 언어 갱신 외에도 수동 갱신으로 독어도 설치할 수 있습니다. 단, 수동 갱신을 수행하려면 정보 센터를 중지한 다음 갱신하고 재시작해야 합니다. 정보 센터는 갱신 프로세스 동안에는 사용할 수 없습니다.

프로시저

이 주제는 자동 갱신 프로세스에 대한 설명입니다. 수동 갱신에 대한 지시사항은 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신』 주제를 참조하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 자동으로 갱신하려면 다음을 수행하십시오.

1. Linux 운영 체제에서,
   a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.
   b. 설치 디렉토리에서 `doc/bin` 디렉토리까지 탐색하십시오.

c. 다음과 같이 ic-update 스크립트를 실행하십시오.

```
ic-update
```

2. Windows 운영 체제에서,
   a. 명령 창을 여십시오.
   b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.
   c. 설치 디렉토리에서 doc₩bin 디렉토리까지 탐색하십시오.
   d. 다음과 같이 ic-update.bat 파일을 실행하십시오.

```
ic-update.bat
```

결과

DB2 정보 센터가 자동으로 재시작됩니다. 갱신사항이 사용 가능한 경우, 정보 센터에는 새로 갱신된 주제가 표시됩니다. 정보 센터 갱신을 사용할 수 없는 경우, 메시지가 로그에 추가됩니다. 로그 파일은 doc₩eclipse₩configuration 디렉토리에 있습니다. 이 로그 파일 이름은 임의로 생성된 번호입니다. (예: 1239053440785.log).

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신

DB2 정보 센터를 로컬로 설치한 경우, IBM으로부터 문서 갱신사항을 받아 설치할 수 있습니다.

로컬로 설치된 DB2 정보 센터를 수동으로 갱신하려면 다음을 수행하십시오.

1. 컴퓨터에서 DB2 정보 센터를 중지한 후 독립형 모드에서 다시 시작하십시오. 독립형 모드에서 정보 센터를 실행하면 사용자의 네트워크와 연결된 다른 사용자는 정보 센터에 액세스할 수 없으므로 갱신사항을 적용할 수 있습니다. DB2 정보 센터의 워크스테이션 버전은 항상 독립형 모드에서 실행됩니다. .
2. 어떤 갱신사항이 사용 가능한지 확인하려면 갱신 기능을 사용하십시오. 설치해야 할 갱신사항이 있는 경우, 갱신 기능을 사용하여 이를 가져온 후 설치할 수 있습니다.

   주: 인터넷에 연결되지 않은 머신에 DB2 정보 센터 갱신사항을 설치해야 할 경우, 인터넷에 연결된 머신을 사용하여 갱신 사이트를 로컬 파일 시스템에 미러해야 DB2 정보 센터가 설치됩니다. 네트워크 상에 문서 갱신사항을 설치하려는 사용자가 많을 경우에는 갱신 사이트를 로컬로 미러링하거나 갱신 사이트의 프록시를 작성하여 갱신을 수행하면 각 개인에게 필요한 시간을 줄일 수 있습니다.
   갱신 패키지가 사용 가능하면 갱신 기능을 사용하여 패키지를 가져오십시오. 그러나 갱신 기능은 독립형 모드에서만 사용할 수 있습니다.
3. 독립형 정보 센터를 중지한 후 컴퓨터에서 DB2 정보 센터를 재시작하십시오.

주: Windows 2008, Windows Vista 이상의 경우 이 섹션 다음에 나오는 명령은 관리자로 실행해야 합니다. 전체 관리자 권한으로 명령 프롬프트 또는 그래픽 도구를 열려면 단축키를 마우스 오른쪽 단추로 누른 후 **관리자로 실행**을 선택하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 갱신하려면 다음을 수행하십시오.

1. DB2 정보 센터를 중지하십시오.
   - Windows에서는 **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 정보 센터** 서비스를 마우스 오른쪽 단추로 누른 후 **중지**를 선택하십시오.
   - Linux에서는 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 독립형 모드에서 정보 센터를 시작하십시오.
   - Windows 사용자:
     a. 명령 창을 여십시오.
     b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.
     c. 설치 디렉토리에서 doc₩bin 디렉토리까지 탐색하십시오.
     d. 다음과 같이 help_start.bat 파일을 실행하십시오.

        ```
        help_start.bat
        ```
   - Linux 사용자:
     a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 /opt/ibm/db2ic/V9.7 디렉토리에 디폴트로 설치됩니다.
     b. 설치 디렉토리에서 doc/bin 디렉토리까지 탐색하십시오.
     c. 다음과 같이 help_start 스크립트를 실행하십시오.

        ```
        help_start
        ```

   독립형 정보 센터를 표시하기 위해 시스템의 기본 웹 브라우저가 열립니다.
3. **갱신** 단추( )를 누르십시오. (JavaScript™가 브라우저에서 사용 가능해야 합니다.) 정보 센터의 오른쪽 패널에서 **갱신사항 찾기**를 누르십시오. 기존 문서의 갱신사항 목록이 표시됩니다.
4. 설치 프로세스를 시작하려면 설치할 선택란을 체크한 후 **갱신사항 설치**를 누르십시오.
5. 설치 프로세스가 완료되면 **완료**를 누르십시오.
6. 독립형 정보 센터를 중지하십시오.

- Windows에서 설치 디렉토리의 doc₩bin 디렉토리를 탐색한 후 다음과 같이 help_end.bat 파일을 실행하십시오.

  ```
  help_end.bat
  ```

  주: help_end 일괄처리 파일에는 help_start 일괄처리 파일로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start.bat를 중지하는 데 Ctrl-C 또는 다른 메소드를 사용하지 마십시오.
- Linux에서 설치 디렉토리의 doc/bin 디렉토리를 탐색한 후 다음과 같이 help_end 스크립트를 실행하십시오.

  ```
  help_end
  ```

  주: help_end 스크립트에는 help_start 스크립트로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start 스크립트를 중지하는 데 다른 메소드를 사용하지 마십시오.

7. DB2 정보 센터를 재시작하십시오.
   - Windows에서는 **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 정보 센터** 서비스를 마우스 오른쪽 단추로 누른 후 **시작**을 선택하십시오.
   - Linux에서는 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 start
     ```

갱신된 DB2 정보 센터에는 새로 갱신된 주제가 표시됩니다.

DB2 자습서

DB2 자습서는 DB2 제품의 다양한 측면에 대해 학습하는 데 유용합니다. 각 레슨은 단계별 지시사항을 제공합니다.

시작하기 전에

http://publib.boulder.ibm.com/infocenter/db2help/의 정보 센터에서 XHTML 버전의 자습서를 볼 수 있습니다.

일부 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크에 대한 전제조건 설명은 자습서를 참조하십시오.

DB2 자습서

자습서를 보려면 제목을 누르십시오.

『**pureXML®**』(*pureXML Guide*)

XML 데이터를 저장하고 원시 XML 데이터 스토어로 기본 조작을 수행하려면 DB2 데이터베이스를 설정하십시오.

*Visual Explain 자습서*의 『**Visual Explain**』

Visual Explain을 사용하여 성능을 향상시킬 수 있도록 SQL문을 분석, 최적화 및 조정합니다.

DB2 문제점 해결 정보

DB2 데이터베이스 제품을 사용하는 데 도움이 되는 광범위한 문제를 해결하고 판별할 수 있는 정보가 있습니다.

DB2 문서

문제점 해결 정보는 *DB2 문제점 해결 안내서* 또는 *DB2 정보 센터*의 데이터베이스 기본 섹션을 참조하십시오. DB2 진단 도구 및 유틸리티를 사용하여 문제점을 찾아내고 식별하는 방법, 가장 일반적인 문제점에 대한 솔루션 및 DB2 데이터베이스 제품에서 발생할 수 있는 문제점을 해결하는 방법 등의 정보가 있습니다.

DB2 기술 지원 웹 사이트

문제점이 있는 경우 원인 및 솔루션을 찾으려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 서적, 기술 노트, APAR(Authorized Program Analysis Report 또는 버그 수정), FixPack 및 기타 자원에 대한 링크가 있습니다. 이러한 기술 자료를 검색하여 문제에 대한 가능한 솔루션을 찾을 수 있습니다.

http://www.ibm.com/software/data/db2/support/db2_9/에서 DB2 기술 지원 웹 사이트에 액세스하십시오.

이용약관

다음 조건에 따라 이 책을 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 귀하 사업장 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서의 2차적 저작물을 만들거나 본 문서 또는 그 일부를 복제, 배포 또는 전시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 이 책이나 이 책에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대한 어떠한 허가나 라이센스 또는 권한도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 본 문서의 사용이 IBM의 이익을 해친다고 판단되거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 이 사이트에서 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 경우에만 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 본 문서의 내용에 대해 어떠한 보증도 제공하지 않습니다. 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 현 상태대로 제공합니다.

부록 E. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. 비IBM 제품에 대한 정보는 이 책을 처음 발행할 때의 정보에 기초하고 있으며 변경될 수 있습니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트 문자 세트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

지적 자산 라이센스
법정 및 지적 자산 법률
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책 사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

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

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠. 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등) 하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 다른 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 되는 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 샘플 프로그램은 어떠한 보증없이 ″있는 그대로″ 제공됩니다. IBM은 샘플 프로그램의 사용으로 인해 발생하는 모든 손해에 대해 책임을 지지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. *_enter* 연도_. All rights reserved.

상표

IBM, IBM 로고 및 ibm.com®은 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 기타 회사의 상표입니다. 현재 IBM 상표 목록은 웹 “저작권 및 상표 정보”(www.ibm.com/legal/kr/copytrade.shtml)에 있습니다.

다음 표장은 기타 회사의 상표 또는 등록상표입니다.

- Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.
- Java 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.
- UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.
- Intel®, Intel 로고, Intel Inside®, Intel Inside 로고, Intel® Centrino®, Intel Centrino 로고, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium® 및 Pentium®은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표 또는 등록상표입니다.
- Microsoft, Windows, Windows NT® 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

[숫자]

A

B

C

D

E

F

G

H

I

L

M

N

R

S

U

V

W

X

SA30-3973-00

Spine information:

Linux, UNIX 및 Windows용 IBM DB2 9.7

파티셔닝 및 클러스터링 안내서

IBM