IBM® DB2® Universal Database

관리 안내서: 성능

버전 7

SA30-0989-01

IBM® DB2® Universal Database

관리 안내서: 성능

버전 7

SA30-0989-01

이 책의 정보와 지원하는 제품을 사용하기 전에 반드시 719 페이지의 『부록F. 주의사항』을 읽으십시오.

이 문서에는 IBM의 특허 정보가 들어 있습니다. 이 정보는 사용권 계약하에서 제공되며 저작권법의 보호를 받습니다. 이 책에 있는 정보는 어떠한 제품도 보증하지 않으며, 이 책에 제공된 어떤 내용도 이와 같이 해석되어서는 안됩니다.

책에 대한 주문은 한국 IBM 담당자 또는 해당 지역의 IBM 지방 사무소로 문의하십시오.

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

© Copyright International Business Machines Corporation 1993, 2001. All rights reserved.

목차

© Copyright IBM Corp. 1993, 2001

이 책에 관하여

관리 안내서는 세 가지 볼륨으로 되어 있으며 2000년 문제를 준비하는 DB2* 관계형 데이터베이스 관리 시스템(RDBMS) 제품을 사용하고 관리하는 데 필요한 정보를 제공하고, 다음과 같은 내용을 다루고 있습니다.

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

이 책에서 설명하는 여러 타스크는 서로 다른 인터페이스를 사용하여 수행될 수 있습니다.

- 그래픽 인터페이스로부터 데이터베이스에 액세스하고 조작할 수 있도록 해주는 **명령행 처리기**. 이 인터페이스에서 SQL문과 DB2 유틸리티 기능을 실행할 수도 있습니다. 이 책의 예 중 대부분이 이 인터페이스를 사용하여 설명되어 있습니다. 명령행 처리기 사용에 대한 자세한 내용은 *Command Reference*를 참조하십시오.
- **API**. API를 통해 응용프로그램 내에서 DB2 유틸리티 기능을 실행할 수 있습니다. API 사용에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.
- 시스템 구성, 디렉토리 관리, 시스템 백업 및 복구, 작업 스케줄링, 미디어 관리와 같은 관리 타스크를 그래픽으로 수행할 수 있도록 해주는 **제어 센터**. 제어 센터에는 시스템 간에 그래픽으로 데이터 복제를 설정하는 복제 관리가 있습니다. 또한 제어 센터에서 그래픽 사용자 인터페이스를 통해 DB2 유틸리티 기능을 실행할 수 있습니다. 플랫폼에 따라 제어 센터를 호출하는 다른 방법이 있습니다. 예를 들어, 명령행에서 `db2cc` 명령을 사용하거나 (OS/2에서) DB2 폴더에서 제어 센터 아이콘을 선택하거나 또는 Windows 플랫폼에서 패널 시작을 사용하십시오. 소개 도움말을 보려면 제어 센터 창의 **도움말** 풀다운 메뉴에서 **시작하기**를 선택하십시오. 제어 센터에서 **Visual Explain** 및 **성능 모니터** 도구가 호출됩니다.

© Copyright IBM Corp. 1993, 2001

관리 타스크를 수행하는 데 사용할 수 있는 다른 도구가 있습니다. 도구는 다음과 같습니다.

- 스크립트라는 작은 응용프로그램을 저장하는 스크립트 센터. 이 스크립트는 운영 체제 명령뿐 아니라 SQL문, DB2 명령도 수록하고 있습니다.
- 다른 DB2 조작에서 기인한 메시지를 모니터하는 경보 센터
- 제어 센터, 경보 센터, 복제의 설정을 변경하는 도구 설정
- 작업이 자동으로 수행되도록 스케줄하는 저널
- 웨어하우스 오브젝트를 관리하는 Data Warehouse Center

이 책의 사용자

이 책은 기본적으로 지역 또는 원격 클라이언트에 의해 액세스될 데이터베이스를 설계, 구현 및 유지보수해야 하는 데이터베이스 관리자, 시스템 관리자, 보안 관리자 및 시스템 운영자를 대상으로 합니다. 또한 프로그래머나 DB2 관계형 데이터베이스 관리 시스템의 관리 및 조작을 이해해야 하는 기타 다른 사용자에게도 도움이 됩니다.

이 책의 구성 방식

이 책에는 다음 주요 주제에 대한 정보가 수록되어 있습니다.

성능 개요

- 제1장 성능 요소에서는 DB2 UDB 성능을 관리 및 향상시키기 위한 개념 및 고려사항을 소개합니다.
- 제2장 아키텍처 및 프로세스 개요에서는 기초가 되는 DB2 Universal Database 아키텍처와 프로세스를 소개합니다.

응용프로그램 성능 조정

- 제3장 응용프로그램 고려사항에서는 응용프로그램을 설계할 때 데이터베이스 성능을 향상시킬 수 있는 몇몇 기법에 대해 설명합니다.
- 제4장 환경상의 고려사항에서는 데이터베이스 환경을 설정할 때 데이터베이스 성능을 향상시킬 수 있는 몇몇 기법에 대해 설명합니다.

- 제5장 시스템 카탈로그 통계에서는 데이터에 대한 통계의 수집 방식과 최적 성능을 보장하는 데 사용된 방식에 대해 설명합니다.
- 제6장 SQL 컴파일러의 이해에서는 SQL문이 SQL 컴파일러를 사용하여 컴파일될 때 발생하는 내용에 대해 설명합니다.
- 제7장 SQL Explain 기능에서는 SQL 컴파일러가 데이터에 액세스하기 위해 취하는 선택사항을 살펴볼 수 있는 Explain 기능에 대해 설명합니다.

시스템의 조정 및 구성

- 제8장 수행시의 성능에서는 데이터베이스 관리 프로그램이 런타임 성능에 영향을 주는 메모리 및 기타 고려사항을 사용하는 방법에 대한 개요를 제공합니다.
- 제9장 조정자(governor) 사용에서는 조정자를 사용하여 데이터베이스 관리의 여러 측면을 제어하는 방법에 대해 설명합니다.
- 제10장 프로세서 추가를 통해 구성 조정에서는 데이터베이스 시스템의 크기 증가와 관련된 고려사항과 타스크를 소개합니다.
- 제11장 데이터베이스 파티션에 걸친 데이터 재분산에서는 파티션된 데이터베이스 환경에서 파티션 전반에 걸쳐 데이터를 재분산하는 데 필요한 타스크에 대해 설명합니다.
- 제12장 벤치마크 테스팅에서는 벤치마크 테스트의 개념 및 수행 방법에 대해 설명합니다.
- 제13장 DB2 구성에서는 데이터베이스 관리 프로그램 및 데이터베이스 구성 파일과 구성 매개변수의 값에 대해 설명합니다.

부록

- 부록A. DB2 레지스트리 및 환경 변수에서는 프로파일 등록 값과 환경 변수에 대해 설명합니다.
- 부록B. Explain 테이블 및 정의에서는 DB2 Explain 기능에 의해 사용되는 테이블에 대한 정보와 이들 테이블을 작성하는 방법을 제공합니다.
- 부록C. SQL Explain 도구>에서는 DB2 Explain 도구인 db2expln과 dynexpln 사용에 대한 정보를 제공합니다.
- 부록D. db2exfmt - Explain 테이블 형식 도구에서는 DB2 Explain 테이블의 내용을 형식화합니다.

- 부록E. DB2 라이브러리 사용에서는 마법사, 온라인 도움말, 메시지 및 책을 포함하여 DB2 라이브러리 구조에 대한 정보를 제공합니다.

관리 안내서의 다른 볼륨에 대한 간략한 개요

관리 안내서: 계획

*관리 안내서: 계획*은 데이터베이스 설계에 관련되어 있습니다. 이 책은 논리적 및 물리적 설계 관련 문제, 분산 트랜잭션 관련 문제 그리고 고가용성 주제에 대해 설명합니다. 해당 볼륨의 특정 장 및 부록에서는 다음에 대해 간략히 설명합니다.

DB2 Universal Database의 세계

- "DB2 Universal Database 관리"에는 DB2 Universal Database에 대한 소개 및 개요가 있습니다.

데이터베이스 개념

- "기본 관계형 데이터베이스 개념"에는 복구 오브젝트, 저장 오브젝트 및 시스템 오브젝트를 포함하여 데이터베이스 오브젝트에 대한 개요가 있습니다.
- "연합 시스템"에서는 단일 명령문에서 두 개 이상의 DBMS 또는 데이터베이스를 참조하는 사용자 제출 SQL문과 응용프로그램을 지원하는 데이터베이스 관리 시스템(DBMS)인 연합 시스템에 대해 설명합니다.
- "병렬 데이터베이스 시스템"에서는 DB2에서 사용 가능한 병렬 처리 유형에 대한 소개를 제공합니다.
- "데이터 웨어하우징 정보"에서는 데이터 웨어하우징 및 데이터 웨어하우징 타스크의 개요를 제공합니다.
- "Spatial Extender 정보"에서는 Spatial Extender를 소개하며 그 목적과 처리하는 데이터에 대해 설명합니다.

데이터베이스 설계

- "논리 데이터베이스 설계"에서는 논리 데이터베이스 설계의 개념과 지침에 대해 설명합니다.
- "물리 데이터베이스 설계"에서는 데이터 저장영역과 관련된 고려사항을 포함하여 물리 데이터베이스 설계의 지침에 대해 설명합니다.

분산 트랜잭션 프로세싱

- "분산 데이터베이스 설계"에서는 단일 트랜잭션에서 다중 데이터베이스에 액세스하는 방법에 대해 설명합니다.
- "트랜잭션 관리 프로그램 설계"에서는 CICS와 같은 분산 트랜잭션 프로세싱 환경에서 데이터베이스를 사용하는 방법에 대해 설명합니다.

고가용성 시스템

- "고가용성 및 Failover 지원 소개"에는 DB2가 제공하는 고가용성 failover 지원에 대한 개요가 있습니다.

부록

- "데이터베이스 이주 계획"에서는 버전 7로의 데이터베이스 이주에 대한 정보에 대해 설명합니다.
- "릴리스 간 비호환성"에서는 버전 7을 포함하여 릴리스간에 도입되는 비호환성에 대해 설명합니다.
- "자국어 지원(NLS)"에서는 국가, 언어, 코드 페이지에 대한 정보를 포함하여 DB2 자국어 지원에 대해 설명합니다.

관리 안내서: 구현

*관리 안내서: 구현*은 데이터베이스 설계의 구현에 관련되어 있습니다. 해당 볼륨의 특정 장 및 부록에서는 다음에 대해 간략히 설명합니다.

제어 센터 사용 관리

- "GUI 도구를 사용한 DB2 관리"에서는 데이터베이스 관리에 사용되는 그래픽 사용자 인터페이스(GUI) 도구에 대해 설명합니다.

설계 구현

- "데이터베이스를 작성하기 전에"에서는 데이터베이스를 작성하기 전의 전제조건에 대해 설명합니다.
- "데이터베이스 작성"에서는 데이터베이스 및 관련된 데이터베이스 오브젝트 작성과 연관되는 타스크에 대해 설명합니다.

- ″데이터베이스 변경″에서는 데이터베이스와, 데이터베이스나 관련된 데이터베이스 오브젝트의 수정 또는 삭제와 연관되는 타스크를 변경하기 전에 수행해야 하는 사항에 대해 설명합니다.

데이터베이스 보안

- ″데이터베이스 액세스 제어″에서는 데이터베이스 자원에 대한 액세스를 제어하는 방법에 대해 설명합니다.
- ″DB2 활동 감사″에서는 원하지 않거나 예상하지 않은 데이터 액세스를 검출하고 모니터하는 방법에 대해 설명합니다.

데이터 이동

- ″데이터 이동 유틸리티″에서는 한 페이지에 걸쳐 데이터를 이동시키고 *데이터 이동 유틸리티 안내 및 참조서* 책으로 넘어 가는 여러 가지 방법을 소개합니다.

복구

- ″데이터베이스 복구″에서는 한페이지에 걸쳐 데이터베이스 백업, 복원 및 롤 포워드의 개념을 소개합니다. 자세한 내용은 *Data Recovery and High Availability Guide and Reference*를 참조하십시오.

부록

- ″DCE(Distributed Computing Environment) 디렉토리 서비스 사용″에서는 DCE 디렉토리 서비스를 사용하는 방법에 대해 설명합니다.
- ″데이터베이스 복구에 대한 User Exit″에서는 데이터베이스 로그 파일과 함께 User Exit 프로그램을 사용하는 방법과 몇몇 User Exit 샘플 프로그램에 대해 설명합니다.
- ″여러 데이터베이스 파티션 서버로 명령 발행″에서는 파티션된 데이터베이스 환경의 모든 파티션으로 명령을 전송하는 *db2_all*과 *rah* 쉘 스크립트를 사용하는 방법에 대해 설명합니다.
- ″Windows NT용 DB2가 Windows NT 보안을 사용하여 작업하는 방법″에서는 DB2가 Windows NT 보안을 사용하여 작업하는 방법에 대해 설명합니다.
- ″Windows NT 성능 모니터 사용″에서는 Windows NT 성능 모니터에서 DB2를 등록하고 성능 정보를 사용하는 방법에 대한 정보를 제공합니다.

- ″Windows NT 또는 Windows 2000 데이터베이스 파티션 서버에서 작업″은 Windows NT 또는 Windows 2000에서 데이터베이스 파티션 서버와 함께 사용할 수 있는 유틸리티에 대한 정보를 제공합니다.
- ″논리 노드 구성″에서는 파티션된 데이터베이스 환경에서 여러 논리 노드를 구성하는 방법에 대해 설명합니다.
- ″고속 노드간 통신″에서는 DB2 Universal Database와 함께 사용하기 위해 가상 인터페이스 아키텍처를 사용 가능하게 하는 방법에 대해 설명합니다.
- ″LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스″에서는 LDAP 디렉토리 서비스를 사용하는 방법에 대한 정보를 제공합니다.
- ″제어 센터 확장″에서는 새 조치를 포함하여 새 도구 막대 버튼을 추가하고 새 오브젝트 정의 및 새 조치 정의를 추가하여 제어 센터를 확장하는 방법에 대한 정보를 제공합니다.

제1부 성능 개요

© Copyright IBM Corp. 1993, 2001

제1장 성능 요소

성능은 특정 워크로드가 주어진 컴퓨터 시스템이 작동하는 방식입니다. 성능은 하나 이상의 시스템 응답 시간, 처리량, 사용 가능성을 통해 측정됩니다. 성능은 또한 다음 사항에 따라 영향받습니다.

- 사용 가능한 자원
- 이러한 자원이 얼마나 제대로 사용되고 공유되는지

일반적으로, 시스템의 비용 이익 비율을 향상시키려고 할 때 성능 조정을 수행합니다. 특정 목표에는 다음이 포함됩니다.

- 처리 비용을 높이지 않고 대형의, 더 많은 요구 및 워크로드를 처리하는 것(예를 들면, 새 하드웨어를 구입하거나 더 많은 프로세서 시간을 사용하지 않고 워크로드를 높이는 것이 포함됩니다.)
- 처리 비용을 높이지 않고, 시스템 응답 시간을 단축하고, 처리량을 높이는 것
- 사용자에 대한 서비스에 악영향을 미치지 않고 처리 비용을 줄이는 것

성능을 기술 측면에서 비용 측면으로 전환하는 것은 쉽지 않습니다. 성능 조정은 비용이 많이 들기 때문에(사용자의 시간 및 프로세서 시간을 통해) 조정 프로젝트에 임하기 전에 가능한 이점에 대한 비용을 산출해야 합니다. 일부는 다음과 같은 면에서 확실히 이점이 있습니다.

- 좀더 효율적인 자원의 사용
- 시스템에 좀더 많은 사용자를 추가하는 능력

빠른 응답 시간으로 인한 좀더 많은 사용자의 충족도와 같은 이점은 별로 없습니다. 이 두 이점을 모두 고려해야 합니다.

마법사가 DB2에 통합됨으로써 성능 관련 관리 타스크를 완료하는 데 많은 도움이 됩니다. 이 타스크는 사용자가 시간을 거의 소비하지 않으며 많은 성능 향상을 달성할 수 있는 타스크입니다. 마법사는 각 타스크에서 한 단계씩 진행할 수 있도록 안내합니다. 마법사는 제어 센터와 클라이언트 구성 지원 프로그램을 통해 사용할 수 있습니다.

© Copyright IBM Corp. 1993, 2001

성능 구성 마법사는 비지니스 요구사항에 맞도록 구성 매개변수를 갱신하여 데이터베이스의 성능을 조정하는 데 도움이 됩니다. 이 마법사는 데이터베이스 작성 마법사에 이르기까지 데이터베이스의 성능을 향상시키는 데 도움이 됩니다. 다른 마법사는 각 테이블 및 일반 데이터 액세스의 성능 향상을 돕는 데 사용 가능합니다. 이 마법사의 내용에는 테이블 작성, 색인화 및 다중 사이트 갱신 구성 마법사가 포함됩니다. 마법사는 오브젝트에서 오른쪽 마우스 버튼을 눌러 제어 센터에서 찾을 수 있습니다.

지침 조정

다음 지침은 성능 조정에 대한 전반적인 접근 방법을 개발하는 데 도움이 됩니다.

보상 감소 법칙을 기억하십시오. 맨 처음이 성능이 가장 우수합니다. 일반적으로, 그 다음부터 조금씩 변경을 할수록 이점이 줄어들고 노력은 더 많이 필요하게 됩니다.

조정을 위한 조정을 하지 마십시오. 식별된 제한조건을 해결하려면 조정하십시오. 성능 문제점의 1차 원인이 아닌 자원을 조정하면 주요 제한조건을 해결할 때까지 응답 시간이 별 차이가 없거나 효과가 없으며, 실제로 후속 조정 작업을 더욱 어렵게 만들 수 있습니다. 중요한 개선 가능성이 있는 경우, 이는 응답 시간의 주요 인수가 되는 자원의 성능을 향상시키는 데 달려 있습니다.

전체 시스템을 고려하십시오. 하나의 매개변수나 시스템만 별도로 조정할 수 없습니다. 조정을 하기 전에 시스템 전반에 어떤 영향을 미칠 것인지를 고려해야 합니다.

한 번에 하나의 매개변수만 변경하십시오. 한 번에 둘 이상의 성능 조정 변수를 변경하지 마십시오. 모든 변경이 장점을 갖고 있다고 확신하더라도, 각 변경의 영향력이 얼마나 되는지는 평가할 방법이 없습니다. 한 번에 둘 이상의 매개변수를 변경하여 사용자가 행한 교환(trade-off)을 효율적으로 평가할 수 없습니다. 한 분야를 개선하기 위해 매개변수를 조정할 때마다 거의 항상 생각지 않은 다른 분야에 영향을 미치게 됩니다. 한 번에 하나의 매개변수를 변경하여 변경이 원하는 사항을 수행하는지 평가하는 벤치마크를 가질 수 있게 합니다.

*레벨별로 측정 및 재구성*하십시오. 한 번에 하나의 매개변수만을 변경해야 하는 이유와 동일한 이유로, 한 번에 하나의 시스템 레벨만 조정하십시오. 시스템 내의 다음과 같은 레벨 목록을 지침으로 사용할 수 있습니다.

- 하드웨어
- 운영 체제
- 응용프로그램 서버(AS) 및 리퀘스터(AR)
- 데이터베이스 관리 프로그램
- SQL문
- 응용프로그램

하드웨어 및 소프트웨어 문제점을 점검하십시오. 일부 성능 문제점은 하드웨어, 소프트웨어 또는 둘다에 서비스를 적용하여 정정될 수 있습니다. 간단히 서비스를 적용하여 해결할 수 있을 경우에는 시스템 모니터링 및 조정에 시간을 너무 낭비하지 마십시오.

하드웨어를 갱신하기 전에 문제점을 이해하십시오. 추가 기억영역이나 프로세서가 있으면 즉시 성능이 향상될 것 같더라도 병목 현상이 어디서 발생하는지 시간을 두고 살펴보십시오. 병목 현상을 처리할 처리 능력이나 채널이 없음을 발견한 경우에만 추가 디스크 저장영역을 구입하십시오.

조정을 시작하기 전에 프로시듀어를 백업해 두십시오. 앞에서 주의를 주었듯이, 일부 조정은 예상치 않은 성능 결과를 초래할 수 있습니다. 조정으로 인해 성능이 저하되면 원위치한 다음 다른 조정을 시도해야 합니다. 이전 설정이 간단히 다시 호출만 하면 되는 방식으로 저장되어 있으면 틀린 정보를 원위치하는 것은 훨씬 간단해집니다.

디스크 저장영역

시스템을 구성하는 하드웨어가 시스템 성능에 미칠 수 있는 영향에 대해서는 이미 언급하였습니다. 성능의 하드웨어 영향에 대한 예로, 디스크 저장영역과 연관되는 일부 영향을 고려하십시오.

성능에 영향을 미치는 디스크 저장영역 관리 방법에는 네 가지가 있습니다.

- **저장영역을 나누는 방법**

색인과 데이터간, 테이블 공간 사이, 버퍼 풀 사이에 제한된 저장영역을 나누는 방법이 다른 상황에서 각각 수행되는 방법을 많이 좌우합니다.

- **낭비되는 저장영역**

  자체에서 낭비되는 저장영역은 이를 사용하는 시스템의 성능에는 영향을 미치지 않으나, 다른 곳에서 성능을 향상시키기 위해 사용될 수 있는 자원을 나타낼 수 있습니다.

- **디스크 I/O 분산**

  여러 디스크 저장영역 장치 및 제어기에서의 디스크 I/O 요구의 균형 유지 정도에 따라 데이터베이스 관리 프로그램이 디스크에서 정보를 검색하는 시간이 얼마나 빨라지는지가 결정됩니다.

- **저장영역 부족**

  사용 가능한 저장영역이 한계에 도달하면 전반적으로 성능이 저하될 수 있습니다.

성능 향상 프로세스

다음 프로세스를 사용하여 임의의 시스템 성능을 향상시키십시오.

1. 성능 목표를 정의하십시오.
2. 성능 표시기를 설정하십시오.
3. 성능 모니터링 플랜을 개발하십시오.
4. 플랜을 수행하십시오.
5. 목표를 충족시키는지를 판별하기 위해 측정치를 분석하십시오. 충족하면 성능 모니터링 자체가 시스템 자원을 사용하기 때문에 측정 횟수를 줄이는 것을 고려하십시오. 그렇지 않으면 다음 단계를 계속하십시오.
6. 시스템 내의 주요 제한조건을 판별하십시오.
7. 교환 조건을 제시할 수 있는 위치와 어떤 자원에서 추가 로드가 발생할 것인지를 결정하십시오. (거의 모든 조정에 시스템 자원과 여러 성능 요소간의 교환이 포함됩니다.)

8. 시스템 구성을 조정하십시오. 둘 이상의 조정 옵션을 변경하는 것이 필요하다고 생각될 경우, 변경사항을 한 번에 하나씩 구현하십시오. 임의의 레벨에 남아 있는 옵션이 없으면 자원의 한계에 도달한 것이며 하드웨어를 업그레이드해야 합니다.
9. 위의 4단계로 리턴하여 시스템 모니터를 계속하십시오.

주기적으로 또는 시스템이나 워크로드에 중요한 변경을 한 후에는 다음을 수행하십시오.

- 위의 1단계로 리턴하십시오.
- 목표 및 표시기를 다시 검토하십시오.
- 모니터링 및 조정 전략을 다시 세우십시오.

시스템을 조정할 수 있는 한계

시스템의 효율을 향상시키는 데에는 한계가 있습니다. 시스템 성능 향상에 얼마만큼의 비용과 시간을 들여야 하는지와 추가로 얼마만큼의 시간과 비용을 들여야만 시스템 사용자에게 도움이 되는지를 고려해 보십시오.

시스템은 조정 없이도 어느 정도 적당히 수행될 수 있으나, 기대치만큼은 수행하지 못할 수도 있습니다. 각 데이터베이스는 고유합니다. 사용자의 데이터베이스와 이를 사용할 응용프로그램을 개발하면 곧바로 사용 가능한 조정 매개변수를 검토하고 사용자의 상황을 반영할 수 있도록 이를 사용자 정의하는 방법을 배우도록 하십시오. 일부 상황에서 시스템을 조정하면 약간의 이점이 있을 수 있습니다. 그러나 대부분의 상황에서는 이점이 큽니다.

마법사는 제어 센터에서 사용이 가능하며 데이터베이스 매개변수를 조정하는 데 도움이 됩니다. 성능 구성 마법사는 제어 센터에서 조정하려는 데이터베이스를 오른쪽 마우스 버튼으로 누르면 찾을 수 있습니다.

시스템에 성능 병목 현상이 발생하면 조정은 더욱 진가를 발휘하게 됩니다. 성능 한계에 근접할 때 시스템에서 약 10% 정도 사용자 수를 늘리면 응답 시간은 10% 이상 증가합니다. 이 상황에서는 시스템을 조정하여 이러한 성능 저하 현상을 개선할 방법을 강구해야 합니다. 그러나 조정이 도움이 되지 않는 부분이 있습니다. 이 시점에서는 사용자 환경 내에서 목표와 기대치를 수정해야 합니다. 아니면 더

많은 디스크 저장영역, 더 빠른 CPU, 추가 CPU, 더 많은 주 메모리, 더 빠른 통신 링크 또는 이들 변경사항을 조합하는 것을 고려하여 시스템 환경을 변경해야 합니다.

비공식적인 접근 방법

성능 목표를 설정하고 포괄적인 방법으로 모니터 및 조정할 시간이 부족하면 사용자의 의견을 들어 성능을 조정해야 합니다. 성능 관련 문제점이 있는지를 알아 보십시오. 몇 가지 간단한 질문을 하여 문제점을 찾아 내거나 어디서부터 문제점을 찾아야 할지를 판별할 수 있습니다. 예를 들어, 사용자에게 다음과 같은 질문을 할 수 있습니다.

- 『느린 응답』이라는 것은 무엇을 의미합니까? 예상하는 것보다 10% 느리다는 것입니까, 아니면 10배 느리다는 것입니까?
- 언제 문제점을 인식하셨습니까? 최근입니까, 아니면 항상 그랬습니까?
- 같은 문제점에 대해 불만이 있는 다른 사용자가 있습니까? 불만이 있는 사람이 한두 사람입니까, 아니면 그룹 전체입니까?
- (사용자 그룹 전체가 어려움을 겪고 있으면 이들이 모두 동일한 근거리 통신망에 연결되어 있습니까?)
- 사용자가 겪고 있는 문제점이 특정 트랜잭션이나 응용프로그램에 연관되어 있습니까?
- 문제점이 점심 시간과 같은 일정 기간에만 나타납니까, 아니면 하루 종일 나타납니까?

모두 함께 배치

기존의 DB2 아키텍처는 주요 개념 및 프로세스를 이해하는 것이 다른 성능 관련 문제를 이해하는 데 도움이 되므로 매우 중요합니다. 저장영역 아키텍처, 데이터 관리, 처리 모델 및 메모리 모델과 같은 주제는 모두 초기에 다음 장에 제시됩니다. 자세한 내용은 11 페이지의 『제2장 아키텍처 및 프로세스 개요』를 참조하십시오.

응용프로그램 성능 조정은 사용자의 응용프로그램 및 데이터베이스와의 상호작용과 연관된 성능 주제와 관련이 있습니다. 응용프로그램 자체에 관련된 주제(동시

성, 잠금, 최적화 클래스, 조회 결과 세트 제어, 행 블로킹, 복합 SQL 사용)가 실려 있습니다. 또한 응용프로그램 성능에 관련된 문자 변환, 저장 프로시듀어, 데이터베이스 활성화 및 병렬 처리의 장점에 대해서도 간략하게 설명합니다. 자세한 내용은 45 페이지의 『제3장 응용프로그램 고려사항』을 참조하십시오.

조회 최적화에 관련된 주제(조회 최적화에 영향을 미치는 구성 매개변수, 조회 최적화에서 노드 그룹 및 테이블 공간의 영향, 색인이 조회 최적화에 미칠 수 있는 막대한 영향)가 실려 있습니다. 자세한 내용은 101 페이지의 『제4장 환경상의 고려사항』을 참조하십시오.

시스템 카탈로그 통계는 응용프로그램이 얼마나 잘 데이터에 액세스할 수 있는지에 큰 영향을 미칩니다. 다음 주제는 통계(RUNSTATS 유틸리티, 분산 통계, 색인 통계 및 사용자가 갱신할 수 있는 통계)와 연관되어 있습니다. 자세한 내용은 125 페이지의 『제5장 시스템 카탈로그 통계』를 참조하십시오.

SQL 컴파일러는 각 응용프로그램을 취하여 해당 응용프로그램에 대한 최상의 액세스 플랜을 결정합니다. 응용프로그램 내의 각 조회는 평가되며 조회의 목적을 가장 명확히 정의하도록 설계된 여러 다른 조작을 거칠 수 있습니다. 그런 다음 각 조회에 대해 다른 액세스 메소드(스캔 및 조인)를 검토하여 조회가 요청한 데이터를 가장 빨리 검색할 수 있는 방법을 판별합니다. 병렬 처리의 효과도 고려합니다. 자세한 내용은 167 페이지의 『제6장 SQL 컴파일러의 이해』를 참조하십시오.

응용프로그램 조회에서 발생하는 내용을 이해하는 데 도움이 되는 DB2 제품 내에서 사용 가능한 여러 가지 도구가 있습니다. 이 도구는 응용프로그램 성능에 영향을 미치는 조건에 대한 설명에 관련되어 있습니다. 자세한 내용은 243 페이지의 『제7장 SQL Explain 기능』을 참조하십시오.

개별 응용프로그램의 조정과 함께 이러한 응용프로그램이 수행하는 데이터베이스의 성능도 고려해야 합니다. 데이터베이스의 성능은 거의 메모리 활용도에 따라 결정됩니다. 성능(버퍼 풀, 데이터 프리페치, 병렬 I/O, 정렬 기능, 테이블 내의 데이터 재구성 필요성, 데이터베이스 에이전트 개념)에 관련하여 서라운드 메모리에 대한 여러 주제가 실려 있습니다. 자세한 내용은 273 페이지의 『제8장 수행시의 성능』을 참조하십시오.

응용프로그램이 데이터베이스를 사용하는 방법을 관리하도록 설정할 수 있는 조정자(governor)가 있습니다. 자세한 내용은 323 페이지의 『제9장 조정자(governor) 사용』을 참조하십시오.

데이터베이스의 성능을 향상시키기 위해 프로세서의 수 및 데이터베이스 파티션의 수를 증가시킬 수 있습니다. 자세한 내용은 341 페이지의 『제10장 프로세서 추가를 통해 구성 조정』을 참조하십시오.

일단 데이터베이스 파티션의 수를 증가시키면 데이터베이스 내의 데이터가 데이터베이스 파티션에 고루 분포하는지 또는 제대로 재분산되는지를 확인하려고 합니다. 자세한 내용은 355 페이지의 『제11장 데이터베이스 파티션에 걸친 데이터 재분산』을 참조하십시오.

데이터베이스가 제대로 수행되는지를 판별하기 위해 벤치마크 테스트를 수행할 수 있습니다. 벤치마크 테스트에 대한 방법론, 벤치마크 테스트를 준비하는 방법, 벤치마크 프로그램의 작성 및 벤치마크 테스트의 수행 모두 중요한 주제입니다. 자세한 내용은 363 페이지의 『제12장 벤치마크 테스팅』를 참조하십시오.

379 페이지의 『제13장 DB2 구성』에 확장된 데이터베이스 관리 프로그램 세트 및 데이터베이스 구성 매개변수가 개별적으로 제시됩니다.

이러한 성능 주제에 관련된 추가 정보도 들어 있습니다. 부록에는 다음이 포함됩니다.

- 575 페이지의 『부록A. DB2 레지스트리 및 환경 변수』
- 609 페이지의 『부록B. Explain 테이블 및 정의』
- 643 페이지의 『부록C. SQL Explain 도구』
- 695 페이지의 『부록D. db2exfmt - Explain 테이블 형식 도구』

제2장 아키텍처 및 프로세스 개요

DB2에 대한 데이터베이스 조작 성능에 대해 작업할 때 DB2 아키텍처 및 프로세스에 관련되는 기본 개념에 대해 이해해야 합니다. 이 장에는 DB2 Universal Database 작동 방법에 대한 내용을 제공하기 위한 충분한 정보가 수록되어 있습니다. 이후의 장에서도 여기에 수록된 주제 중 일부에 대한 세부사항이 제공되지만, 이 장의 내용은 이후의 내용을 이해할 수 있도록 하는 문맥을 작성합니다.

첫 번째 그림은 DB2 UDB에 대한 아키텍처와 프로세스의 개요를 보여줍니다.

© Copyright IBM Corp. 1993, 2001

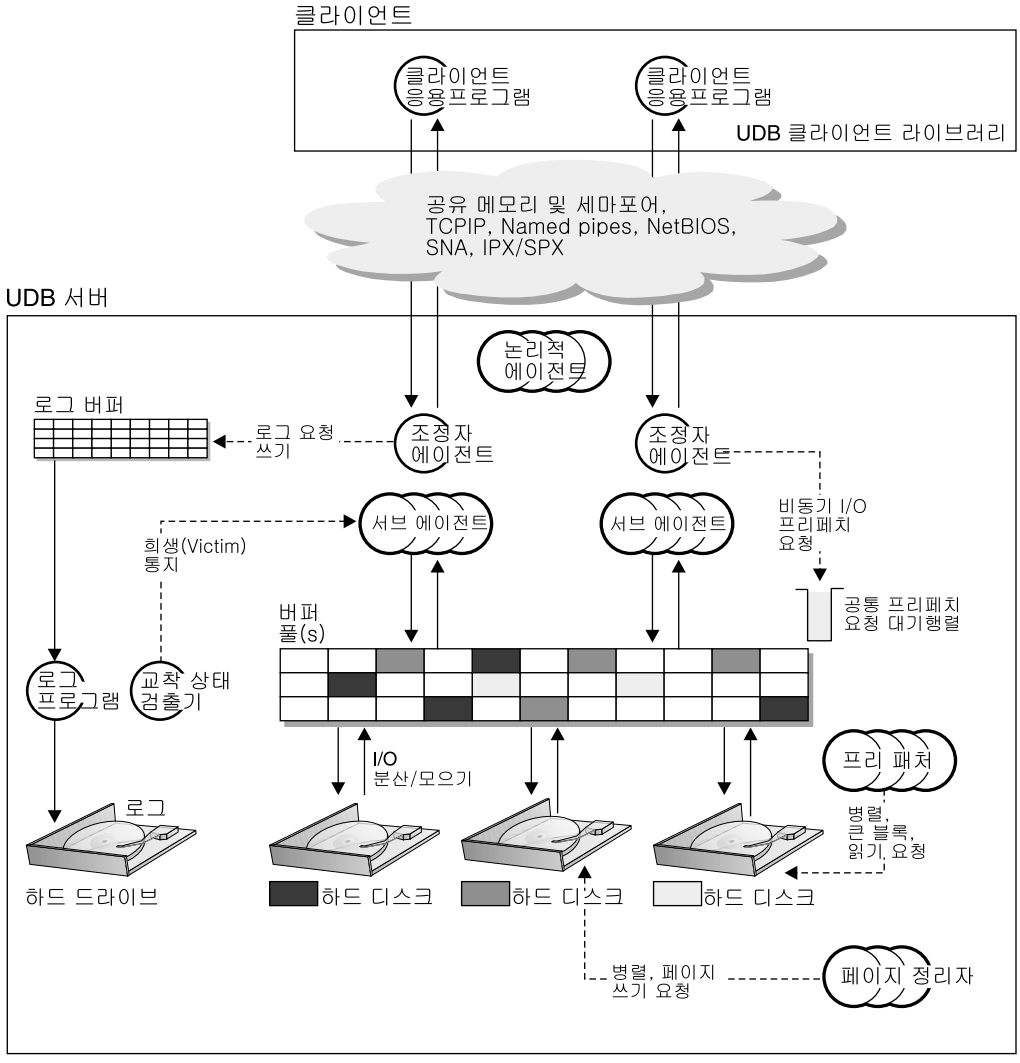


그림 1. 아키텍처 및 프로세스 개요

클라이언트측에서 볼 때 DB2 Universal Database 클라이언트 라이브러리와 링크되어 있는 지역 또는 원격 응용프로그램이 있습니다.

클라이언트와 DB2 Universal Database 서버 사이에는 지역 또는 원격 클라이언트와 서버 사이의 통신 수단을 나타내는 『cloud』가 있습니다. 지역 클라이언트는 공유 메모리와 세마포어를 사용하여 통신하고, 원격 클라이언트는 Named Pipes(NPIPE), TCP/IP, NetBIOS, IPX/SPX 또는 SNA와 같은 프로토콜을 사용합니다.

서버측에서 볼 때 활동은 **EDU**(Engine Dispatchable Unit) 단위로 제어됩니다. 이 장의 모든 그림에서 EDU는 원이나 원 그룹으로 표시됩니다. EDU는 Windows 기반 플랫폼과 OS/2에서는 스레드로(모두 단일 프로세스 내에서) 그리고 UNIX에서는 프로세스로 구현됩니다. 가장 일반적인 유형의 EDU는 DB2 에이전트입니다. 이러한 EDU는 응용프로그램 대신 대부분의 SQL 처리를 수행합니다. EDU의 다른 예로는 다양한 유형의 I/O 처리를 담당하는 DB2 프리페처와 페이지 정리자를 들 수 있습니다. 자세한 내용은 311 페이지의 『데이터베이스 에이전트』를 참조하십시오.

각각의 클라이언트 응용프로그램에는 『조정자 에이전트』라고 하는 고유 EDU가 지정됩니다. 이 에이전트는 해당 응용프로그램에 대한 처리를 조정하고 그 응용프로그램과 통신합니다. 클라이언트 응용프로그램 요청 처리시 함께 작동되도록 지정된 서브에이전트 세트가 있을 수도 있습니다. 대칭적 멀티프로세싱 환경에서처럼 서버가 상주하는 머신에 여러 프로세서가 있으면 클라이언트 응용프로그램 요청이 해당 프로세스를 이용할 수 있도록 여러 서브에이전트가 지정되었을 수 있습니다.

모든 에이전트와 서브에이전트는 EDU의 작성 및 파괴를 최소화하는 풀링 알고리즘을 사용하여 관리됩니다.

버퍼 풀은 사용자 테이블 데이터, 색인 데이터 및 카탈로그 데이터의 데이터베이스 페이지가 임시로 이동되고 수정되는 저장 메모리의 영역입니다. 버퍼 풀은 디스크보다는 메모리에서 데이터에 액세스하는 것이 훨씬 빠르므로 전체 데이터베이스 성능에 영향을 미치는 주 요소입니다. 응용프로그램에 필요한 것보다 더 많은 데이터가 버퍼 풀에 있으면 디스크 저장영역에서 데이터를 찾는 데 소요되는 시간에 비해 이 데이터에 액세스하는 데 필요한 시간이 덜 듭니다. 자세한 내용은 282 페이지의 『데이터베이스 버퍼 풀 관리』를 참조하십시오.

프리페처 및 페이지 정리자 EDU와 함께 버퍼 풀의 구성은 응용프로그램에 필요한 데이터의 사용 가능성 결과와 데이터 액세스 속도를 제어합니다.

프리페처는 응용프로그램에 데이터가 필요하기 전에 디스크에서 데이터를 검색하여 버퍼 풀로 이동시키도록 표시됩니다. 예를 들어, 많은 볼륨의 데이터를 통해 스캔해야 하는 응용프로그램은 데이터 프리페처가 없을 경우 디스크에서 버퍼 풀로 데이터가 이동되기를 기다려야 합니다. 응용프로그램의 에이전트는 비동기 우선 읽기 요청을 공통 프리페치 대기행렬로 전송합니다. 프리페처가 사용 가능하게 되면 그 프리페처는 큰 블록이나 분산 읽기 입력 조작을 통해 해당 요청을 구현하여 요청된 페이지를 디스크에서 버퍼 풀로 가져옵니다. 데이터베이스 데이터 저장영역에 여러 디스크가 있으면 데이터가 그 디스크에서 스트

라이핑될 수 있음을 의미합니다. 이렇게 데이터가 스트라이핑되면 프리페처가 동시에 여러 디스크를 사용하여 데이터를 검색할 수 있습니다. 자세한 내용은 292 페이지의 『버퍼 풀로 데이터 프리페치』 및 295 페이지의 『프리페치 및 병렬 I/O를 위한 I/O 서버 구성』을 참조하십시오.

프리페처는 데이터를 버퍼 풀로 가져오는 데 사용됩니다. 페이지 정리자는 데이터를 버퍼 풀에서 디스크 외부로 이동시키는 데 사용됩니다.

페이지 정리자는 응용프로그램 에이전트와는 독립적인 백그라운드 EDU로, 더 이상 필요하지 않은 버퍼 풀에서 페이지를 찾아서 기록합니다. 페이지 정리자는 버퍼 풀에 프리페처에 의해 검색되는 페이지를 위한 공간이 있도록 할 수 있습니다.

독립적인 프리페처와 페이지 정리자 EDU가 없으면 응용프로그램 에이전트가 버퍼 풀과 디스크 저장영역 사이의 모든 데이터 읽기 및 쓰기를 수행해야 합니다.

여러 응용프로그램이 데이터베이스의 데이터에 대해 작업할 경우, 두 개 이상의 응용프로그램 사이에 『교착 상태』가 발생할 가능성이 있습니다. 교착 상태는 다음 그림에 나와 있습니다.

교착 상태 개념

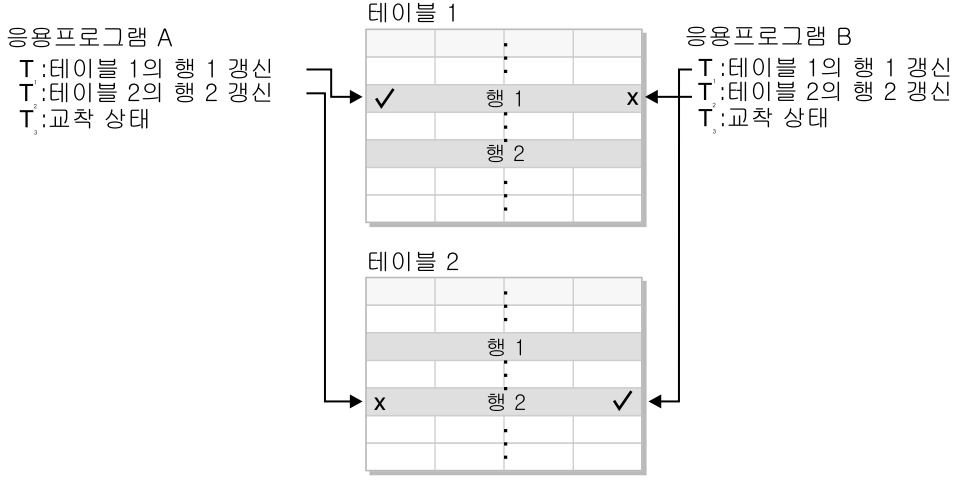


그림 2. 교착 상태 검출기

『교착 상태』란 여러 개의 응용프로그램이, 다른 응용프로그램이 데이터 잠금을 해제하기를 기다리는 것을 의미합니다. 대기 중인 응용프로그램 각각은 잠금을 통해 다른 응용프로그

램에 필요한 데이터를 보유합니다. 이렇게 잠긴 데이터는 다른 응용프로그램에 필요한 데이터를 보유하는 하나 이상의 다른 응용프로그램에 필요합니다. 다른 응용프로그램이 보유된 데이터에 대한 잠금을 해제하기를 기다리는 상호적 대기로 인해 교착 상태가 발생하므로, 응용프로그램은『다른』응용프로그램이 보유된 데이터에 대한 잠금을 해제할 때까지 영원히 기다릴 수도 있습니다. 다른 응용프로그램은 필요한 데이터에 대한 잠금을 자발적으로 해제하지는 않습니다. 이러한 교착 상태를 중단하기 위한 프로세스가 필요합니다.

이름에서 제시하는 것처럼 교착 상태 검출기는 잠금 상태에서 기다리는 에이전트에 대한 정보를 모니터합니다. 교착 상태 검출기는 교착 상태에서 응용프로그램 중 하나를 임의로 선택하여『자원하는』응용프로그램에서 현재 보유하고 있는 잠금을 해제합니다. 해당 응용프로그램의 잠금을 해제하면 대기 중인 다른 응용프로그램에 필요한 데이터가 사용 가능하게 됩니다. 그러면 일반적으로 대기 중인 응용프로그램이 필요한 데이터를 사용할 수 있도록 해제되어 데이터베이스의 데이터에 대한 조치를 완료하게 됩니다.

버퍼 풀에 있는 데이터 페이지에 대한 변경사항이 로그됩니다. 로그 버퍼가 존재하며 로그 프로그램 EDU와 연관됩니다. 데이터베이스에 있는 데이터 레코드를 갱신하는 에이전트는 버퍼 풀에서 연관 페이지를 갱신하여 로그 레코드를 기록합니다. 로그 레코드에는 변경을 재실행하거나 실행 취소하기 위해 필요한 정보가 들어 있습니다. 버퍼 풀의 페이지나 로그 버퍼의 로그 레코드는 디스크에 기록되지 않으므로 성능을 즉시 최적화할 수 없습니다. 로그 프로그램 EDU와 버퍼 풀 관리 프로그램은 서로 협조하여 연관된 로그 레코드가 로그에 기록되기 전에 데이터 페이지가 디스크에 기록되지 않도록 하는 쓰기 우선 로깅(WAL;Write Ahead Logging) 프로토콜을 구현합니다. WAL 프로토콜은 로그에 항상 충분한 정보가 있어서 손상으로부터 복구하고 데이터베이스 일관성을 복원할 수 있도록 합니다. 페이지에서의 미확약된 갱신이 디스크에 기록되면 응급 복구에서 연관된 로그 레코드의 실행 취소 정보를 사용하여 갱신 실행을 취소합니다. 확약된 갱신이 디스크에 대한 갱신이 아니면 응급 복구에서 연관된 로그 레코드의 재실행 정보를 사용하여 갱신을 재실행합니다.

주: COMMIT에서 트랜잭션의 모든 로그 레코드가 아직 비워지지 않았으면 디스크까지 비워집니다.

저장영역 아키텍처

저장영역 아키텍처 설명에서는 다음 사항을 고려합니다.

- 『데이터베이스 디렉토리』
- 18 페이지의 『테이블 공간』

데이터베이스 디렉토리

데이터베이스를 작성할 때 기본 정보를 포함하여 데이터베이스에 대한 정보는 디렉토리 내에 배치됩니다. 디렉토리 구조는 CREATE DATABASE 명령에서 제공하는 정보를 기초로 하는 위치에서 작성됩니다. 데이터베이스를 작성할 때 경로나 드라이브의 위치를 지정하지 않을 경우, 기본 위치가 사용됩니다.

데이터베이스를 작성할 위치를 명시적으로 언급하는 것이 좋습니다.

CREATE DATABASE 명령에서 지정하는 디렉토리에서 인스턴스의 이름을 사용하여 서브디렉토리가 작성됩니다. 이 서브디렉토리는 같은 디렉토리하의 서로 다른 인스턴스에서 작성된 데이터베이스가 같은 경로를 사용하지 않도록 합니다. 인스턴스 이름 서브디렉토리 다음에 NODE0000이라는 서브디렉토리가 작성됩니다. 이 서브디렉토리는 다중 논리적 파티션된 데이터베이스 환경에서 파티션을 차별화하는 데 사용됩니다. 노드 디렉토리 다음에는 SQL00001이라는 서브디렉토리가 작성됩니다. 이 서브디렉토리는 데이터베이스 토큰을 사용하여 이름이 지정되며 작성되는 데이터베이스를 나타냅니다. 이것은 또한 CREATE DATABASE 명령에서 지정한 디렉토리의 해당 인스턴스에서 작성한 데이터베이스를 차별화하는 데도 사용됩니다.

디렉토리 구조는 다음과 같이 나타납니다.

```
<your_directory>/<your_instance>/NODE0000/SQL00001/
```

데이터베이스 디렉토리에는 CREATE DATABASE 명령의 부분으로 작성된 몇 개의 파일이 들어 있습니다. 버퍼 풀 정보는 파일 SQLBP.1 및 SQLBP.2에 포함됩니다. 테이블 공간 정보는 파일 SQLSPCS.1 및 SQLSPCS.2에 포함됩니다. 이들 파일의 정보를 백업할 수 있도록 각각 두 가지의 이 파일이 있습니다.

데이터베이스 구성 정보는 SQLDBCON에 포함됩니다. 실행기록 파일 DB2RHIST.ASC 및 해당 백업 DB2RHIST.BAK는 사용자가 읽을 수 있는 파일, 이 파일에는 백업, 복원, 테이블 로드, 테이블 재구성, 테이블 공간 변경 및 기타 데이터베이스 변경사항에 대한 실행기록 정보가 들어 있습니다.

로그 제어 파일 SQLOGCTL.LFH에는 사용 중인 로그에 대한 정보가 들어 있습니다. 복구 처리에서는 이 파일의 정보를 사용하여 로그에서 어느 정도 뒤로 가면 복구를 시작할 수 있는지 판별합니다. SQLOGDIR 서브디렉토리에는 실제 로그 파일이 있습니다.

주: 로그 서브디렉토리가 데이터에 사용되는 서브디렉토리와 다른 디스크에 맵핑되는지 확인해야 합니다. 그러면 디스크 문제점이 데이터나 로그 둘 중에 하나로 제한될 수 있습니다. 또한 로그 파일과 데이터베이스 컨테이너가 같은 디스크 헤드를 이동하기 위해 경쟁하지 않게 되므로 실제적으로 성능상의 이점도 제공할 수 있습니다. *newlogpath* 데이터베이스 구성 매개변수를 사용하여 로그 서브디렉토리의 위치를 변경할 수 있습니다.

SQLT* 서브디렉토리가 작성되며 이 서브디렉토리에는 조작 데이터베이스에 필요한 기본 시스템 관리 공간(SMS) 테이블 공간이 있습니다. 세 가지의 기본 테이블 공간이 작성됩니다.

- SQLT0000.0 서브디렉토리에는 시스템 카탈로그 테이블이 있는 카탈로그 테이블 공간이 있습니다.
- SQLT0001.0 서브디렉토리에는 기본 임시 테이블 공간이 있습니다.
- SQLT0002.0 서브디렉토리에는 기본 사용자 데이터 테이블 공간이 있습니다.

테이블 공간을 고려할 때 『컨테이너』도 읽습니다. SMS 테이블 공간의 경우, 컨테이너는 운영 체제 디렉토리입니다.

각 서브디렉토리나 컨테이너에서는 SQLTAG.NAM이라는 파일이 작성됩니다. 이 파일은 사용 중인 서브디렉토리를 표시하여 후속 테이블 공간 작성시 이러한 서브디렉토리를 사용하지 않도록 합니다. 파일에 저장되는 파일 유형 사이에 구별하기 위해 서로 다른 이름 확장자를 사용하는 다른 파일도 컨테이너 서브디렉토리에서 작성됩니다. 확장자는 다음과 같습니다.

- SQL*.DAT(비 Long 테이블 데이터 포함)

- SQL*.LF(LONG VARCHAR 또는 LONG VARGRAPHIC 데이터 포함)
- SQL*.LB(BLOB, CLOB 또는 DBCLOB 데이터 포함)
- SQL*.LBA(SQL*.LB 파일에 대한 할당 공간과 여유 공간 정보 포함)
- SQL*.INX(색인 테이블 데이터 포함)
- SQL*.DTR(SQL*.DAT 파일의 재구성을 위한 임시 데이터 포함)
- SQL*.LFR(SQL*.LF 파일의 재구성을 위한 임시 데이터 포함)
- SQL*.RLB(SQL*.LB 파일의 재구성을 위한 임시 데이터 포함)
- SQL*.RBA(SQL*.LBA 파일의 재구성을 위한 임시 데이터 포함)

테이블 공간

두 가지 유형의 테이블 공간인 시스템 관리 공간(SMS)과 데이터베이스 관리 공간(DMS)이 지원됩니다. 각 유형에는 서로 다른 환경에 적합하도록 만들어 주는 고유한 특성이 있습니다. 테이블 공간 설계 및 변경에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

SMS 테이블 공간

시스템 관리 공간(SMS) 테이블 공간은 운영 체제 파일에 데이터를 저장합니다. 테이블 공간의 데이터는 시스템의 모든 컨테이너에서 extent별로 스트라이핑됩니다. **extent**는 데이터베이스에 대해 정의된 연속 페이지 그룹입니다. 테이블 공간의 각 테이블에는 모든 컨테이너가 사용하는 고유한 파일 이름이 부여됩니다. 파일 확장자는 파일에 저장된 데이터의 유형을 나타냅니다. 각 테이블의 시작 extent는 컨테이너 전체적으로『라운드 로빈』방식으로 배치됩니다. 그러면 공간 요구량이 테이블 공간의 모든 컨테이너에 고르게 분산됩니다. 이는 작은 테이블이 많을 때 아주 중요합니다.

공간 할당은 추가 공간에 대한 요구가 있을 때 수행됩니다. 기본적으로 공간은 한 번에 한 페이지가 할당됩니다.

DMS 테이블 공간

데이터베이스 관리 공간(DMS) 테이블 공간을 사용할 경우, 데이터베이스 관리 프로그램이 저장 공간을 제어합니다. 장치 또는 파일 목록이 선택되어 DMS 테이블 공간이 정의될 때 테이블 공간에 속하게 됩니다. 장치 또는 파일의 공간은 DB2 데이터베이스 관리 프로그램에 의해 관리됩니다. SMS 테이블 공간 및 컨테이너에

서처럼 DMS 테이블 공간 및 데이터베이스 관리 프로그램은 extent별 스트라이핑을 사용하여 모든 컨테이너에 걸쳐 데이터 분산이 고르게 되도록 합니다.

DMS 테이블 공간은 SMS 테이블 공간과 다릅니다. DMS 테이블 공간의 경우, 테이블 공간이 작성되고 필요할 때 할당되지 않을 경우에 공간이 할당됩니다.

또한 데이터 배치도 두 유형의 테이블 공간에서 다릅니다. 예를 들어, 효율적인 테이블 스캔이 필요하다고 간주할 경우, extent의 페이지가 실제로 연속적인 것은 중요합니다. SMS를 사용할 경우, 운영 체제의 파일 시스템은 각각의 논리 파일 페이지가 실제로 배치되는 곳을 결정합니다. 페이지는 파일 시스템에서의 다른 활동 레벨과, 배치 결정에 사용되는 알고리즘에 따라 할당될 수도, 할당되지 않을 수도 있습니다. 그러나 DMS를 사용할 경우, 데이터베이스 관리 프로그램은 직접 디스크와 인터페이스하므로 페이지가 실제로 연속되도록 할 수 있습니다.

저장영역에서의 페이지 연속 배치에 대해 이러한 일반적인 사항에 대한 한 가지 예외가 있습니다. DMS 테이블 공간에 대해 작업할 때 두 가지의 컨테이너 옵션인 원시 장치와 파일이 있습니다. 파일 컨테이너에 대해 작업할 때 데이터베이스 관리 프로그램은 테이블 공간 작성시 전체 컨테이너를 할당합니다. 전체 테이블 공간에 대한 이러한 초기 할당의 결과로, 파일 시스템은 할당을 수행하지 않더라도 실제 할당이 일반적으로 연속하여 이루어집니다. 이때 항상 연속된다고 보증할 수는 없습니다. 원시 장치 컨테이너에 대해 작업할 때 데이터베이스 관리 프로그램은 전체 장치의 제어를 취하여 extent의 페이지가 연속되도록 합니다.

SMS 테이블 공간과는 달리, DMS 테이블 공간을 구성하는 컨테이너는 용량이 동등에 가까울 필요는 없습니다. 그러나 컨테이너는 용량면에서 동등하거나, 동등에 가까워야 합니다. 또한 컨테이너가 가득 차면 DMS 테이블 공간은 다른 컨테이너의 사용 가능한 여유 공간을 사용합니다.

DMS 테이블 공간에 대해 작업할 때 각각의 컨테이너를 서로 다른 디스크와 연관시키는 것을 고려해야 합니다. 그러면 큰 테이블 공간 용량과 병렬 I/O 조작의 이점을 이용할 수 있습니다.

CREATE TABLESPACE문은 데이터베이스 내에서 새 테이블 공간을 작성하여 컨테이너를 테이블 공간에 지정한 후 테이블 공간 정의 및 속성을 카탈로그에 기록합니다. 테이블 공간을 작성할 때 정의되는 것 중의 하나가 Extent 크기입니다.

extent는 테이블 공간 내에서의 공간 할당 단위입니다. 이것은 단순히 연속 페이지 세트입니다. Extent *크기*는 연속 페이지 수입니다. 하나의 테이블(또는 색인과 같은 다른 오브젝트)만 단일 extent의 페이지를 사용할 수 있습니다. 테이블 공간에서 작성된 모든 오브젝트(테이블, 색인 및 기타 오브젝트)는 논리적 테이블 공간 주소 맵에서 할당된 extent입니다. 하나의 extent는 한 번에 단 하나의 오브젝트에만 속합니다. extent 할당은 SMP(Space Map Pages)를 통해 관리됩니다.

논리적 테이블 공간 주소 맵에서 첫 번째 extent는 내부 제어 정보를 포함하는 테이블 공간에 대한 헤더이고, 두 번째 extent는 테이블 공간에 대한 SMP(Space Map Pages)의 첫 번째 extent입니다. SMP extent는 테이블 공간에서 일정한 간격으로 분산됩니다. 각각의 SMP extent는 단순히 현재 SMP extent에서 다음 SMP extent로의 extent 비트맵입니다. 비트맵은 사용 중이거나 사용되고 있지 않은 중간 extent를 추적하는 데 사용됩니다.

SMP를 따르는 다음 extent는 테이블 공간에 대한 오브젝트 테이블입니다. 오브젝트 테이블은 테이블 공간에 존재하는 사용자 오브젝트와 해당되는 첫 번째 EMP(Extent Map Page) extent가 위치하는 곳을 추적하는 내부 테이블입니다. 각 오브젝트에는 논리적 테이블 공간 주소 맵에 저장되는 오브젝트의 각 페이지에 대한 맵을 제공하는 고유한 EMP가 있습니다.

다음 그림은 DMS 테이블 공간에 대한 논리적 주소 맵을 보여줍니다.

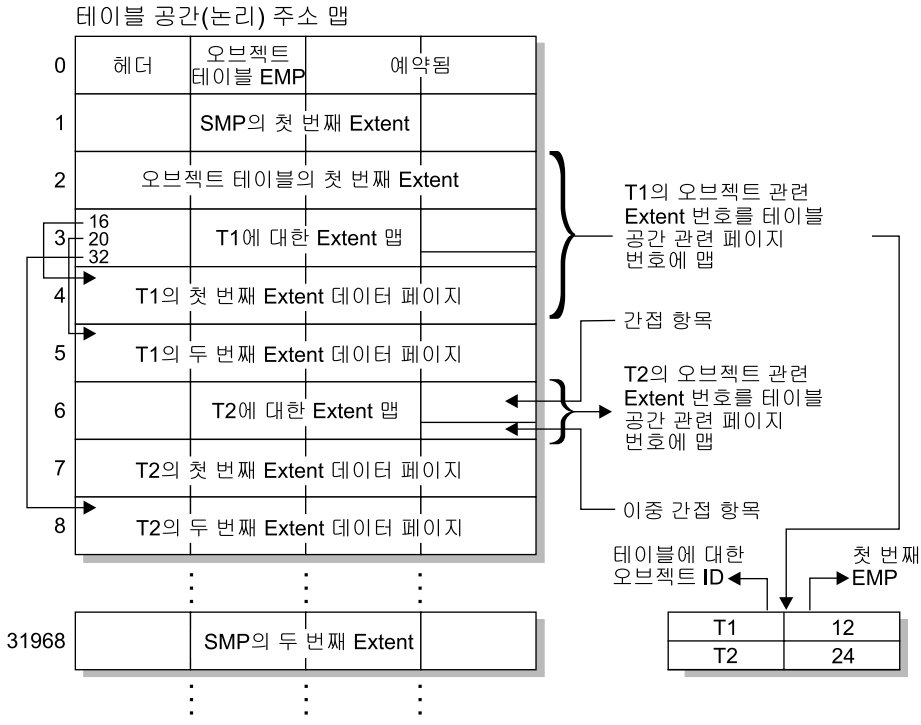


그림 3. DMS 테이블 공간

오브젝트 테이블은 오브젝트 식별자를 테이블의 첫 번째 EMP extent 위치에 맵핑하는 내부 관계 테이블입니다. 이 EMP extent는 직접 또는 간접으로 오브젝트의 모든 extent를 정밀하게 표시합니다. 각 EMP에는 하나의 항목 배열이 있습니다. 각 항목은 오브젝트 관련 extent 번호를 오브젝트 extent가 위치한 테이블 공간 관련 페이지 번호에 맵핑합니다. 직접 EMP 항목은 오브젝트 관련 주소를 직접 테이블 공간 관련 주소에 맵핑합니다. 첫 번째 EMP extent의 마지막 EMP 페이지에는 간접 항목이 있습니다. 간접 EMP 항목은 EMP 페이지에 맵핑되고, 이 페이지는 다시 오브젝트 페이지에 맵핑됩니다. 첫 번째 EMP extent의 마지막 EMP 페이지에 있는 최종 16개 항목에 이중 간접 항목들이 있습니다.

논리적 테이블 공간 주소 맵의 extent는 테이블 공간과 연관되는 컨테이너에서 라운드 로빈 방식으로 스트라이핑됩니다.

SMS와 DMS 테이블 공간 비교

SMS와 DMS 테이블 공간을 비교할 때 SMS 테이블 공간은 일반적인 목적에 대해 탁월한 선택입니다. SMS 테이블 공간은 매우 작은 관리 비용으로 아주 좋은 성능을 제공합니다. DMS 테이블 공간은 상위 성능을 찾을 때 최상의 선택입니다.

장치 컨테이너는 파일 컨테이너나 SMS 테이블 공간을 사용하여 데이터를 이동할 때 이중 버퍼링이 발생할 수 있으므로 최상의 성능을 제공합니다. (이중 버퍼링은 데이터가 데이터베이스 관리 프로그램 레벨에서 처음으로 버퍼링된 후 다시 파일 시스템 레벨에서 버퍼링될 때 발생할 수도 있습니다.)

데이터 관리

데이터베이스 작성, 테이블 공간 작성, 테이블 작성 및 테이블에 데이터 배치 후에는 테이블 구성 방법과 해당 테이블 데이터 검색에 색인을 사용하는 방법을 알아야 합니다.

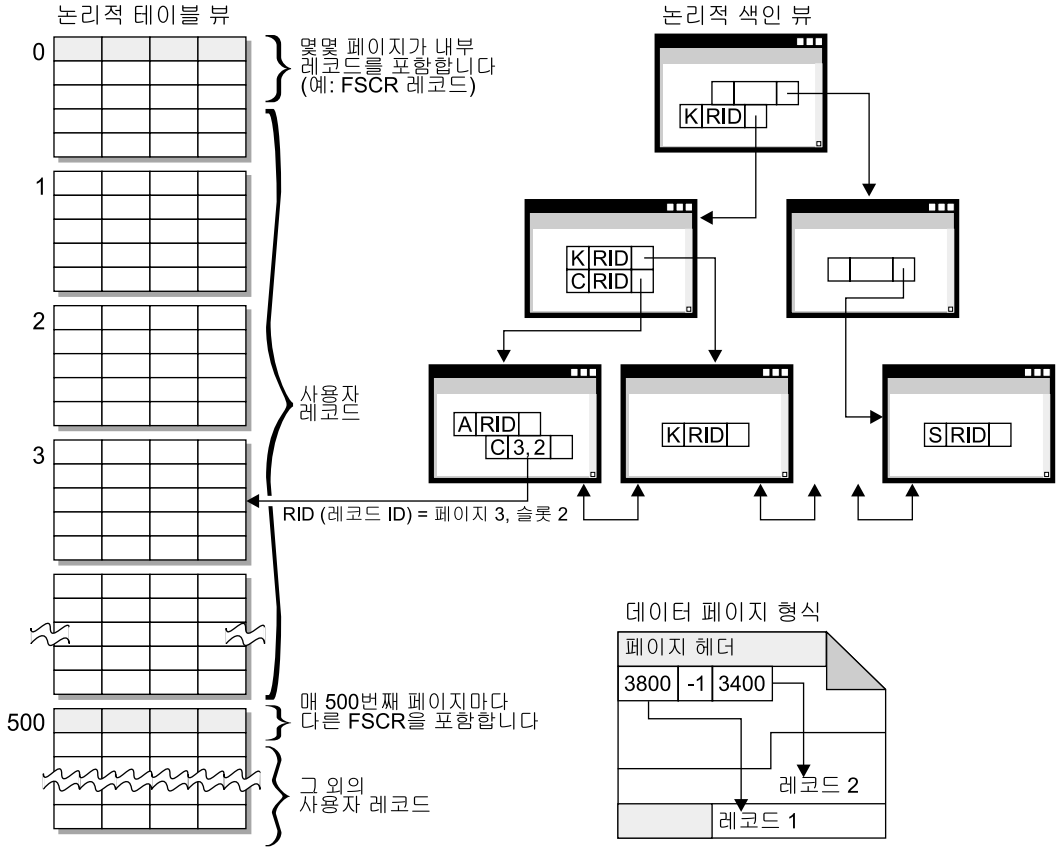


그림 4. 테이블, 레코드 및 색인

논리적으로 테이블 데이터는 데이터 페이지의 목록으로 구성됩니다. 그리고 이들 데이터 페이지는 테이블 공간의 Extent *크기*를 기초로 하여 함께 논리적으로 그룹화됩니다. 예를 들어, Extent *크기*가 4일 경우, 페이지 0 - 3은 첫 번째 extent의 부분이고, 페이지 4 - 7은 두 번째 extent의 부분입니다.

각각의 데이터 페이지 내에 포함되는 레코드 수는 데이터 페이지 *크기*와 레코드 *크기*에 따라 다를 수 있습니다. 최대 255개의 레코드가 한 페이지에 적합할 수 있습니다. 대부분의 페이지에는 사용자 레코드만 있습니다. 그러나 페이지 수가 적으면 테이블을 관리하기 위해 DB2가 사용하는 특수 내부 레코드가 포함됩니다. 예를 들어, 500번째 데이터 페이지마다 FSCR(Free Space Control Record)이 있습니다. 이들 레코드는 다음 FSCR에 도달할 때까지 다음 500개의 데이터 페이지 각각에 대해 존재하는 여유 공간량을 정밀하게 표시합니다. 이 사용 가능한 여유 공간은 레코드를 테이블에 삽입할 때 사용됩니다.

논리적으로 색인 페이지는 주어진 키 값을 가지고 있는 테이블 데이터에서 레코드를 효율적으로 찾을 수 있는 B 트리로 구성됩니다. 색인 페이지의 엔터티 수는 수정되지는 않지만, 키 *크기*에 따라 다를 수 있습니다. DMS 테이블 공간에 있는 테이블의 경우, 색인 페이지의 레코드 ID(RID)는 오브젝트 관련 페이지 번호가 아니라 테이블 공간 관련 페이지 번호를 사용합니다. 그러면 **색인 스캔**을 사용하여 맵핑에 대한 EMP(Extent Map page) 없이도 데이터 페이지에 직접 액세스할 수 있습니다.

각 데이터 페이지의 형식은 다음과 같습니다. 페이지 헤더는 각 데이터 페이지를 시작합니다. 페이지 헤더 다음에는 슬롯 디렉토리가 있습니다. 슬롯 디렉토리의 각 항목은 페이지에서의 서로 다른 레코드에 해당됩니다. 항목 자체는 레코드가 시작하는 데이터 페이지로의 바이트 오프셋입니다. -1 항목은 삭제된 레코드에 해당됩니다.

레코드 ID 및 페이지

레코드 ID(RID)는 3바이트의 페이지 번호와 1바이트의 슬롯 번호로 구성됩니다. 일단 RID를 식별하는 데 색인이 사용되면 RID는 해당 페이지에서 올바른 데이터 페이지와 슬롯 번호를 확보하는 데 사용됩니다. 슬롯의 내용은 페이지 내에서 찾는 레코드의 시작 부분까지의 바이트 오프셋입니다. 일단 레코드에 RID가 할당되면 테이블이 재구성될 때까지 변경되지 않습니다.

데이터 페이지 및 RID 형식

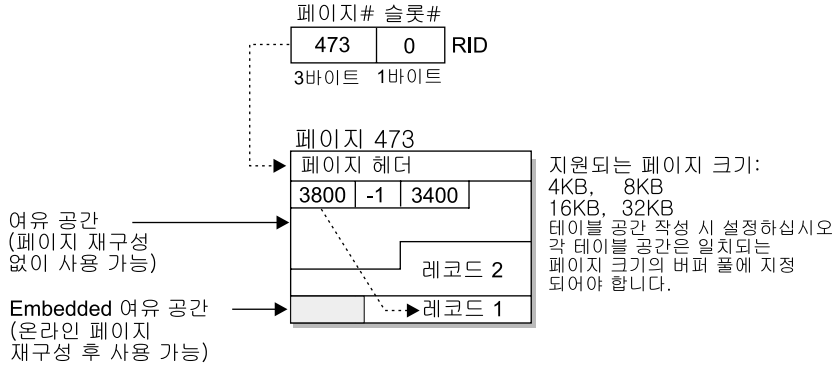


그림 5. 데이터 페이지 및 RID 형식

테이블이 재구성될 때 레코드 삭제 후 데이터 페이지에 남아 있는 삽입된 여유 공간은 사용할 수 있는 여유 공간으로 변환됩니다. RID는 그러한 사용할 수 있는 여유 공간을 이용하기 위해 데이터 페이지에서 레코드 이동을 기초로 재정의됩니다.

DB2는 서로 다른 페이지 *크기*를 지원합니다. 순차적으로 행에 액세스하려는 경향이 있는 워크로드에 대해서는 더 큰 페이지 *크기*를 사용하십시오. 예를 들어, 순차 액세스는 결정 지원 응용프로그램이나 임시 테이블이 광범위하게 사용되는 곳에 사용됩니다. 더 무작위로 액세스하려는 경향이 있는 워크로드에 대해서는 더 작은 페이지 *크기*를 사용하십시오. 예를 들어, 무작위 액세스는 OLTP 환경에서 사용됩니다.

테이블 재구성에 대한 자세한 내용은 304 페이지의 『카탈로그 및 사용자 테이블 재구성』을 참조하십시오.

공간 관리

SQL INSERT문을 사용하여 새 정보를 테이블에 배치할 수 있습니다. 이렇게 할 경우, 작업 완료를 위해 준수하는 INSERT 검색 알고리즘이 있습니다. 먼저, 충분한 공간이 있는 페이지를 찾는 데 FSCR(Free Space Control Record)이 사용됩니다. 그러나 FSCR에서 페이지에 충분한 여유 공간이 있음을 알려 주어도, 다른 트랜잭션의 미확약된 DELETE에 의해 『예약』되어 있으므로 그 공간을 사용하지

못할 수도 있습니다. 결과적으로, 모든 트랜잭션이 자주 COMMIT하도록 해야 합니다. 그렇지 않으면 미확약된 여유 공간을 사용할 수 없게 됩니다.

테이블의 모든 FSCR이 검색되는 것은 아닙니다. DB2MAXFSCRSEARCH 레지스트리 변수는 INSERT를 시도할 때 고려되는 FSCR 수를 제한합니다. 이 레지스트리 변수의 기본값은 5입니다. 5개의 FSCR 내에서 공간이 발견되지 않으면 삽입되는 레코드는 테이블 끝에 추가됩니다. 그리고 INSERT 속도를 최적화하기 위해 후속 레코드도 두 개의 extent가 채워질 때까지 테이블 끝에 추가됩니다. 일단 두 개의 extent가 채워지면 다음 INSERT가 마지막 검색이 종료된 FSCR에서 검색을 재개합니다.

주: DB2MAXFSCRSEARCH의 값은 중요합니다. 테이블이 더 빠르게 커질 수도 있는 상황에서 INSERT 속도에 대해 최적화하려면 이 레지스트리 변수를 작은 값으로 설정하십시오. INSERT 속도가 느려질 수도 있는 상황에서 공간 재사용에 대해 최적화하려면 이 레지스트리 변수를 큰 값으로 설정하십시오.

일단 전체 테이블을 검색하면 더 검색하지 않아도 삽입될 레코드가 추가됩니다. FSCR을 사용하는 검색은 테이블에서 DELETE와 같은 조작 다음에 공간이 만들어질 때까지 수행되지 않습니다.

두 가지의 다른 검색 알고리즘 옵션이 있습니다. 첫 번째는 APPEND MODE입니다. 이 모드에서 새 행은 항상 테이블의 끝에 추가됩니다. FSCR의 검색이나 유지보수는 전혀 수행되지 않습니다. 이 옵션은 ALTER TABLE APPEND ON문을 사용할 때 가능하고, 저널과 같이 커지기만 하는 테이블의 성능을 향상시킬 수 있습니다. 두 번째는 테이블에서 클러스터링 색인을 정의하는 것입니다. 이 경우, 데이터베이스 관리 프로그램은 유사한 색인 키 값을 가지고 있는 다른 레코드와 같은 페이지에 레코드를 삽입하려고 합니다. 해당 페이지에 공간이 없으면 레코드를 주변 페이지에 배치하려고 합니다. 계속 성공하지 못하면 위에 설명된 FSCR 검색 알고리즘이 사용됩니다. 한 가지의 사소한 차이는 최초 적합 방식보다는 최악 적합 방식을 사용한다는 것입니다. 이 최악 적합 방식은 더 많은 여유 공간이 있는 페이지를 선택하는 경향이 있습니다. 이는 해당 키 값을 사용하여 행에 대한 새 클러스터링 영역을 설정하기 위해 수행됩니다.

테이블에서 클러스터링 색인을 정의할 때 테이블을 로드하거나 재구성하기 전에 ALTER TABLE... PCTFREE를 사용하십시오. PCTFREE절은 로드 및 재구성

후에 테이블의 데이터 페이지에서 여유 공간으로 제공된 백분율 값을 그대로 둡니다. 이렇게 하면 클러스터 색인 조작이 원하는 페이지에서 여유 공간을 찾을 수 있는 가능성이 커집니다.

데이터 페이지 및 오버플로우 레코드

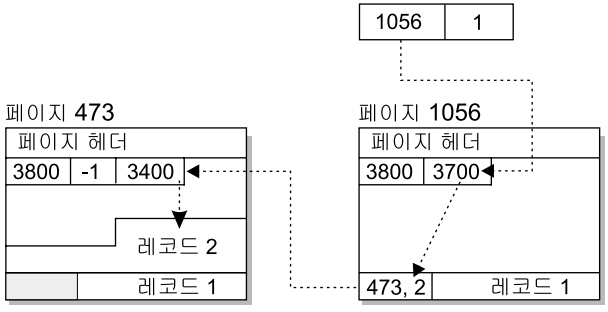


그림 6. 데이터 페이지 및 오버플로우 레코드

오버플로우 레코드는 갱신 요청으로 기존 레코드가 확장되어 현재 페이지에 맞출 수 없을 경우에 발생할 수 있습니다. 확장된 레코드는 충분한 공간이 있는 다른 페이지에서 오버플로우 레코드로 삽입됩니다. 원래의 RID는 오버플로우 레코드의 새 RID를 포함하는 포인터 레코드로 변환됩니다. 테이블의 색인은 원래 RID를 보존하며 읽혀진 여분의 페이지는 요청된 데이터 레코드를 확보하기 위해 필요합니다. 오버플로우 레코드가 많으면 많은 여분의 페이지가 읽혀져서 테이블에 액세스하는 성능이 느려집니다. 테이블을 재구성하면 오버플로우 레코드가 제거됩니다. 그러나 가능할 때마다 레코드를 확장시키는 갱신 요청을 피하여 오버플로우 레코드가 발생하지 않도록 해야 합니다.

색인 관리

DB2 색인은 쓰기 우선 로깅을 사용하는 효율적인 높은 동시성 색인 관리 방법을 기초로 하는 최적화된 B 트리 구현입니다.

최적화된 B 트리 구현에는 단일 색인이 정방향이나 역방향으로(동시에 두 방향을 모두 사용할 수는 없음)의 스캔을 지원할 수 있도록 리프 페이지에 대한 양방향 포인터가 있습니다. 색인 페이지 분할은 90/10 분할이 사용되는 상위 키 페이지에서는 제외하고 보통 반에서 올바르게 작동됩니다. 즉, 색인 키 중에서 상위 10%는 새 페이지에 배치됩니다. 이러한 유형의 색인 페이지 분할은 INSERT 요청이 종종 새 상위 키로 완료되는 워크로드에 유용합니다.

색인의 페이지는 페이지상의 마지막 색인 키가 제거될 때 사용 가능하게 됩니다. 색인을 작성할 때 MINPCTUSED절을 선택하면 이 규칙에 대한 예외가 발생합니다. 이 절의 사용은 색인이 온라인에서 재구성될 수 있음을 나타냅니다. 그리고 이 절과 함께 제공된 값이 색인 리프 페이지에서 사용되는 최소 공간 백분율에 대한 임계값임을 나타냅니다. 색인 페이지에서 키가 삭제된 후 페이지에서 사용되는 공간 백분율이 제공된 값과 같거나 그 이하이면, 나머지 키를 이웃하는 페이지의 키와 병합하려고 합니다. 충분한 공간이 있으면 병합이 수행되고 색인 리프 페이지는 삭제됩니다. 이 절을 사용하면 공간 재사용을 개선할 수 있지만, 사용되는 값이 너무 크면 병합하려고 할 때 소요되는 시간은 증가되고 성공할 가능성도 줄어듭니다. 이 절의 값은 항상 50% 마만이어야 바람직합니다.

CREATE INDEX문의 INCLUDE절은 키 컬럼 외에도 색인 리프 페이지에서 지정된 컬럼을 포함시킬 수 있게 합니다. 그렇게 하면 색인 전용 액세스에 적합한 조회 수가 증가됩니다. 그러나 이렇게 하면 색인 공간 요구량도 증가되어 포함된 컬럼이 자주 갱신될 경우에 색인 유지보수 비용이 증가될 수도 있습니다. 색인 B 트리 순서지정은 키 컬럼을 사용하여 수행되며 포함된 컬럼은 사용하지 않습니다.

잠금

데이터베이스 관리 프로그램은 동시처리 제어를 제공하며 잠금을 사용하여 자원 및 데이터에 대한 액세스를 제어합니다. **잠금**은 응용프로그램을 데이터베이스 관리 프로그램 자원이나 데이터 레코드와 연관시킵니다. 잠금은 다른 응용프로그램이 같은 자원이나 데이터 레코드에 액세스할 수 있는 방법을 제어합니다.

데이터베이스 관리 프로그램은 다음을 기초로 레코드 레벨 잠금이나 테이블 레벨 잠금 중 적절한 잠금을 사용합니다.

- 사전 처리 컴파일 시간이나 응용프로그램이 데이터베이스에 바인드될 때 지정되는 분리 레벨. 분리 레벨은 미확약 읽기(UR), 커서 안정성(CS), 읽기 안정성(RS) 또는 반복 읽기(RR) 중 하나가 될 수 있습니다. 미확약된 데이터에 대한 액세스, 갱신사항 유실 금지, 반복 불가능한 데이터 읽기 허용 그리고 팬텀 읽기 금지를 제어하는 데 서로 다른 분리 레벨이 사용됩니다. 응용프로그램 필요성에 충족되는 최소 분리 레벨을 사용하십시오.

- 최적화 알고리즘에 의해 선택된 액세스 플랜. 테이블 스캔, 색인 스캔 및 기타 데이터 액세스 메소드 각각에서는 데이터에 대한 액세스 유형이 서로 달라야 합니다.
- 테이블의 LOCKSIZE 속성. ALTER TABLE문의 LOCKSIZE절은 테이블에 액세스할 때 잠금 세분성이 사용됨을 나타냅니다. 행 잠금에 해당되는 ROW나 테이블 잠금에 해당되는 TABLE 중에서 선택할 수 있습니다. 읽기 전용 테이블에 대해 ALTER TABLE... LOCKSIZE TABLE을 사용하십시오. 그러면 데이터베이스 활동에 필요한 잠금 수가 줄어듭니다.
- 잠금으로 충당할 메모리의 양. 잠금으로 충당할 메모리의 양은 *locklist* 데이터베이스 구성 매개변수에 의해 제어됩니다. 잠금 목록이 채워질 경우, 성능은 잠금 레벨 자동 업그레이드로 인해 저하되고 데이터베이스에 있는 공유 오브젝트에서의 동시성은 감소됩니다. 잠금 레벨 자동 업그레이드가 자주 발생할 경우, *locklist*나 *maxlocks*의 값을 증가시키십시오.

모든 트랜잭션이 자주 COMMIT하여 보유된 잠금을 해제하는지 확인하십시오.

일반적으로, 다음 중 하나에 해당되는 경우가 아니면 레코드 레벨 잠금이 사용됩니다.

- 선택된 분리 레벨이 미확약 읽기(UR)입니다.
- 선택된 분리 레벨이 반복 읽기(RR)이고 액세스 플랜에서는 술어가 없는 스캔을 요구합니다.
- 테이블의 LOCKSIZE 속성이 『TABLE』입니다.
- 잠금 목록이 채워져 있고 잠금 레벨 자동 업그레이드가 발생합니다.
- LOCK TABLE문을 통해 확보된 명시적 테이블 잠금이 있습니다. LOCK TABLE문은 동시적인 응용프로그램 프로세스가 테이블을 변경하거나 테이블을 사용하지 못하도록 합니다.

잠금 레벨 자동 업그레이드는 하나 이상의 레코드 잠금이 하나의 테이블 잠금으로 변환되는 것입니다. 독점 잠금 레벨 자동 업그레이드는 확보된 테이블 잠금이 독점 잠금인 잠금 레벨 자동 업그레이드입니다. 잠금 레벨 자동 업그레이드는 동시성을 감소시키므로 피해야 합니다.

레코드 잠금 지속기간은 사용되는 분리 레벨에 따라 다릅니다.

- 미확약 읽기(UR) 스캔: 어떤 레코드 잠금도 레코드가 변경되지 않으면 보유되지 않습니다.
- 커서 안정성(CS) 스캔: 레코드 잠금은 커서가 레코드에 위치하고 있는 동안만 보유됩니다.
- 읽기 안정성(RS) 스캔: 규정하는 레코드 잠금만 트랜잭션 지속기간 동안 보유됩니다.
- 반복 읽기(RR) 스캔: 모든 레코드 잠금이 트랜잭션 지속기간 동안 보유됩니다. 이 분리 레벨이 필요하지 않거나 원하지 않는 환경에 있을 경우, DB2_RR_TO_RS 레지스트리 변수를 사용하십시오. 그러면 데이터베이스 관리 프로그램이 RR 의미를 사용하는 데 필요한 여분의 잠금을 피하게 되어 결국 성능이 향상됩니다.

이 주제에 대한 자세한 내용은 55 페이지의 『잠금』을 참조하십시오.

로깅

두 가지의 로깅 전략 중에서 선택할 수 있습니다.

- 로그 레코드가 로그 파일을 채워서 초기 로그 파일에 있는 초기 로그 레코드를 덮어쓰는 순환 로깅. 덮어쓴 로그 레코드는 복구할 수 없습니다.
- 로그 파일이 로그 레코드로 채워지면 아카이브되는 로그 레코드 보유. 새 로그 파일은 로그 레코드에 사용할 수 있게 됩니다. 로그 파일을 보유하면 **롤 포워드 복구**가 가능합니다. 롤 포워드 복구는 로그에 기록되는 완료된 작업 단위(트랜잭션)를 기초로 데이터베이스에 대한 변경사항을 다시 적용합니다. 롤 포워드 복구가 로그의 끝까지 또는 로그 끝 이전의 특정 시점까지 이루어지도록 지정할 수 있습니다.

어느 것을 선택하더라도, 일반 데이터 및 색인 페이지에 대한 모든 변경사항은 로그 버퍼에 기록됩니다. 로그 버퍼의 데이터는 다음과 같이 강제로 디스크에 놓입니다.

- 해당되는 데이터 페이지가 디스크에 강제로 놓이기 전에. 이를 『쓰기 우선 로깅』이라고 합니다.
- COMMIT시 또는 데이터베이스 구성 매개변수를 그룹화할 COMMITS 수 (*mincommit*)의 값에 도달한 후에.

- 로그 버퍼가 거의 가득 찰 경우. 로그 프로그램 프로세서는 『로그 버퍼가 가득 찼음』 조건을 방지하기 위해 디스크에 로그 데이터를 기록합니다.

주: 트랜잭션이 COMMIT문을 사용하여 완료될 때 변경된 모든 페이지는 디스크로 비워져서 복구할 수 있도록 합니다.

트랜잭션이 짧을 경우, 로그 I/O는 COMMIT시의 잦은 로그 비우기로 인해 『병목 현상』을 일으킬 수 있습니다. 그러한 환경에서는 *mincommit* 구성 매개변수를 1보다 큰 값으로 설정해야 『병목 현상』을 제거할 수 있습니다. 1보다 큰 값을 사용할 경우, 몇 개의 트랜잭션에 대한 COMMIT가 보유되거나 『일괄처리』될 수 있습니다. COMMIT할 첫 번째 트랜잭션은 (*mincommit* - 1)보다 많은 트랜잭션이 COMMIT할 때까지 기다립니다. 이 때 로그는 강제로 디스크에 놓이고 모든 트랜잭션이 해당 응용프로그램에 대해 응답합니다. 결과는 몇 개의 개별 로그 I/O 대신 단 하나의 로그 I/O입니다.

현저한 응답 시간 저하를 피하기 위해 (*mincommit* - 1)개의 다른 트랜잭션이 COMMIT하도록 각 트랜잭션이 1초까지만 기다립니다. 1초를 초과할 경우, 대기 중인 트랜잭션은 로그 자체를 강제로 열어서 해당 응용프로그램에 응답합니다. 그러면 *mincommit*를 설정할 수 있어서 더 적은 수의 트랜잭션이 처리되는 동안 성능에 너무 영향을 주지 않도록 할 수 있습니다.

대형 오브젝트(LOB)와 LONG VARCHAR에 대한 변경사항은 음영 페이징을 통해 추적됩니다. LOB 컬럼 변경사항은 로그 유지가 사용되고 LOB 컬럼이 CREATE TABLE문에 정의되어 있지만 NOT LOGGED절은 사용하지 않으면 로그되지 않습니다. LONG 또는 LOB 데이터 유형의 할당 페이지에 대한 변경사항은 일반 데이터 페이지처럼 로그됩니다.

갱신시 발생하는 사항

에이전트가 페이지를 갱신할 때 어떤 상황이 로그와 데이터 페이지에 대해 발생할까요? 여기에 설명되는 프로토콜은 트랜잭션에 필요한 I/O를 최소화하고 복구도 할 수 있게 합니다.

먼저, 갱신될 페이지가 고정되어 독점 잠금으로 래치됩니다. 변경사항을 재실행하고 실행 취소하는 방법을 설명하는 로그 레코드가 로그 버퍼에 기록됩니다. 이 조치의 부분으로, 로그 순차 번호(LSN)가 확보되어 갱신되는 페이지의 페이지 헤더

에 저장됩니다. 변경은 페이지에 대해 수행됩니다. 결국, 페이지는 래치되지 않아서 수정되지 않습니다. 페이지는 디스크에 기록되지 않은 변경사항이 있으므로 『더티(dirty)』 페이지로 간주됩니다. 로그 버퍼도 갱신되었습니다.

로그 버퍼의 데이터와 『더티(dirty)』 데이터 페이지는 둘 다 강제로 디스크에 놓입니다. 성능상의 이유로, 이러한 I/O는 편리한 지점까지(예를 들어, 시스템 로드시 멈추어 있는 동안)나, 복구할 수 있도록 해야 하거나 바인드된 복구 시간까지 지연됩니다. 더 특별하게는, 『더티(dirty)』 페이지가 다음과 같이 강제로 디스크에 놓입니다.

- 다른 에이전트가 희생(victim)으로 이를 선택할 경우.
- 페이지 정리자가 다음의 결과로 페이지에서 작동될 경우.
  - 다른 에이전트가 희생으로 이를 선택합니다.
  - *chngpgs_thresh* 데이터베이스 구성 매개변수 백분율 값을 초과합니다. 일단 초과되면 비동기 페이지 정리자가 『기동(wake-up)』되어 변경된 페이지를 디스크에 기록합니다.
  - *softmax* 데이터베이스 구성 매개변수 백분율 값을 초과합니다. 일단 초과되면 비동기 페이지 정리자가 『기동(wake-up)』되어 변경된 페이지를 디스크에 기록합니다.
- NOT LOGGED INITIALLY절이 호출되고 COMMIT가 발행되는 테이블의 부분으로 페이지가 갱신된 경우. COMMIT시 변경된 모든 페이지는 디스크로 비워져서 복구할 수 있도록 합니다.

로그 버퍼는 다음과 같이 로그 프로그램 EDU(Engine Dispatchable Unit)에 의해 강제로 디스크에 놓입니다.

- 해당되는 데이터 페이지가 디스크에 강제로 놓이기 전에. 이를 『쓰기 우선 로깅』이라고 합니다.
- COMMIT시 또는 데이터베이스 구성 매개변수를 그룹화할 COMMITS 수 (*mincommit*)의 값에 도달한 후에.
- 로그 버퍼가 거의 가득 찰 경우. 로그 프로그램 프로세서는 『로그 버퍼가 가득 찼음』 조건을 방지하기 위해 디스크에 로그 데이터를 기록합니다.

프로세스 모델

지역 및 원격 응용프로그램 프로세스는 같은 데이터베이스에 대해 작업할 수 있습니다. 원격 응용프로그램은 데이터베이스 머신에서 원격인 머신으로부터 데이터베이스를 시작하는 응용프로그램입니다. 지역 응용프로그램은 서버 머신에서 직접 데이터베이스에 접속됩니다.

다음 그림에서 각 원은 UNIX 플랫폼에서는 『프로세스』로, Windows NT와 OS/2에서는 『스레드』로 알려진 EDU를 나타냅니다.

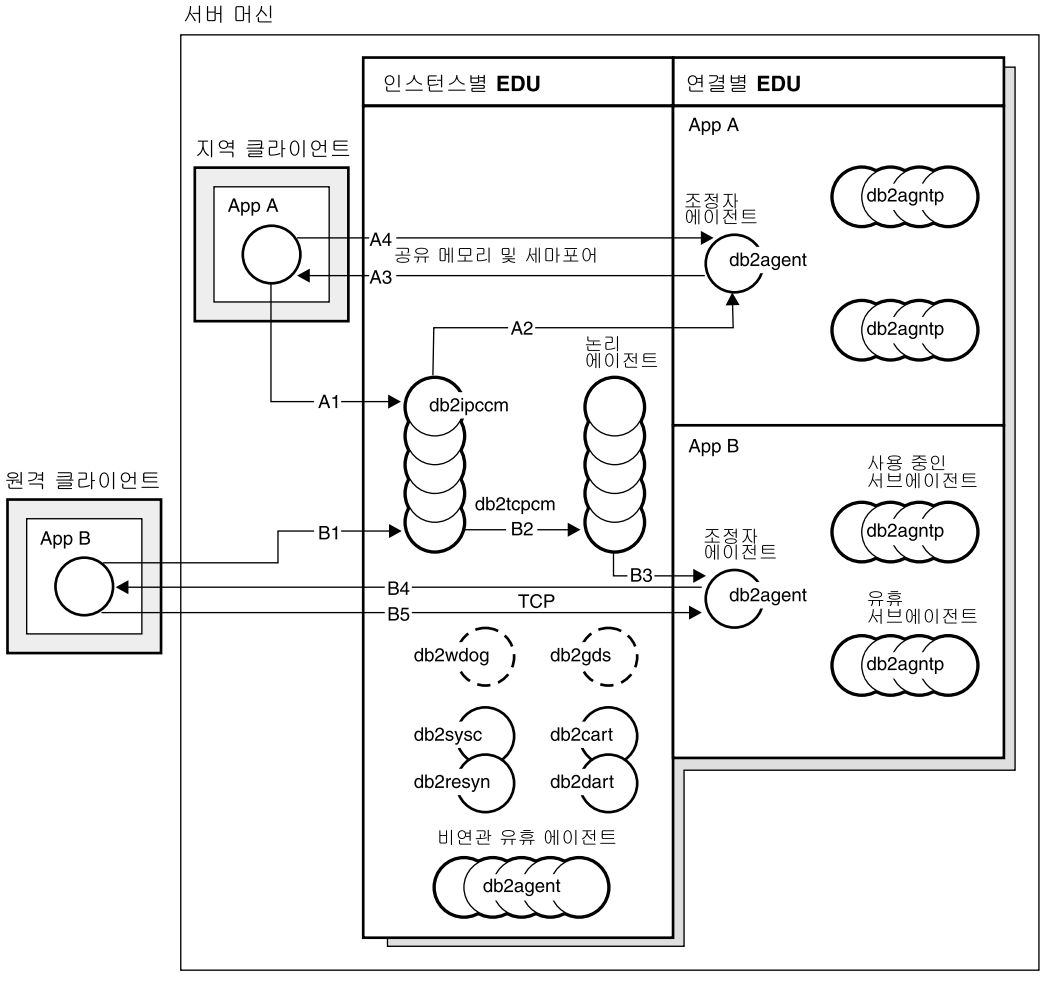


그림 7. 프로세스 모델 개요

데이터베이스에서 수행되는 응용프로그램에서 원하는 작업이 처리되려면 먼저 응용프로그램과 데이터베이스 관리 프로그램 사이의 통신 수단이 설정되어야 합니다.

위의 그림에 있는 A1에서 지역 클라이언트는 먼저 db2ipccm EDU에 대해 작업하여 통신을 설정합니다. A2에 있는 이 EDU는 지역 클라이언트로부터의 응용프로그램 요청에 대한 조정자 에이전트가 되는 db2agent EDU에 대해 작업합니다. 조정자 에이전트는 A3에서 클라이언트 응용프로그램에 접속하여 A4에서 클라이언트 응용프로그램과 데이터베이스 사이의 공유 메모리 및 세마포어 통신을 설정합니다. 지역 클라이언트에 있는 응용프로그램은 데이터베이스에 연결됩니다.

위의 그림에 있는 B1에서 원격 클라이언트는 먼저 db2tcpcm EDU에 대해 작업하여 통신을 설정합니다. 다른 통신 프로토콜을 선택하였으면 해당되는 EDU가 사용됩니다. B2에 있는 db2tcpcm EDU는 논리 에이전트에 대해 작업합니다. B3에 있는 이 EDU는 원격 클라이언트로부터의 응용프로그램 요청에 대한 조정자 에이전트가 되는 db2agent EDU에 대해 작업합니다. 조정자 에이전트는 B4에서 클라이언트 응용프로그램에 접속하여 B5에서 클라이언트 응용프로그램과 데이터베이스 사이의 TCP/IP 통신을 설정합니다. 원격 클라이언트에 있는 응용프로그램은 데이터베이스에 연결됩니다.

이 그림에서 유의할 기타 사항은 다음과 같습니다.

- 에이전트의 클래스에는 논리 에이전트와 작업자 에이전트라는 두 개의 에이전트가 있습니다. 논리 에이전트는 데이터베이스 관리 프로그램에 대한 연결된 응용프로그램을 나타냅니다. 작업자 에이전트는 응용프로그램 요청을 수행하지만, 특정 응용프로그램에 영구적으로 접속되지는 않습니다.
- 활동 조정자 에이전트, 서브에이전트, 비활동 에이전트 및 유휴 에이전트와 같은 네 가지 유형의 작업자 에이전트가 있습니다.
- 논리 에이전트에 의해 표시되는 클라이언트 응용프로그램의 각 프로세스나 스레드는 활동 중인 조정자 에이전트에 링크됩니다.
- 파티션된 데이터베이스 환경에서 그리고 파티션 내 병렬 처리가 가능한 환경에서, 조정자 에이전트는 데이터베이스 요청을 서브에이전트(db2agntp)에 분산시킵니다. 서브에이전트가 응용프로그램에 대한 요청을 수행합니다.
- 유휴 에이전트가 새 작업을 기다리는 에이전트 풀(db2agent)이 있습니다.
- 클라이언트 연결, 로그, 2단계 확약, 백업 및 복원 타스크 그리고 기타 타스크를 관리하는 다른 EDU가 있습니다.

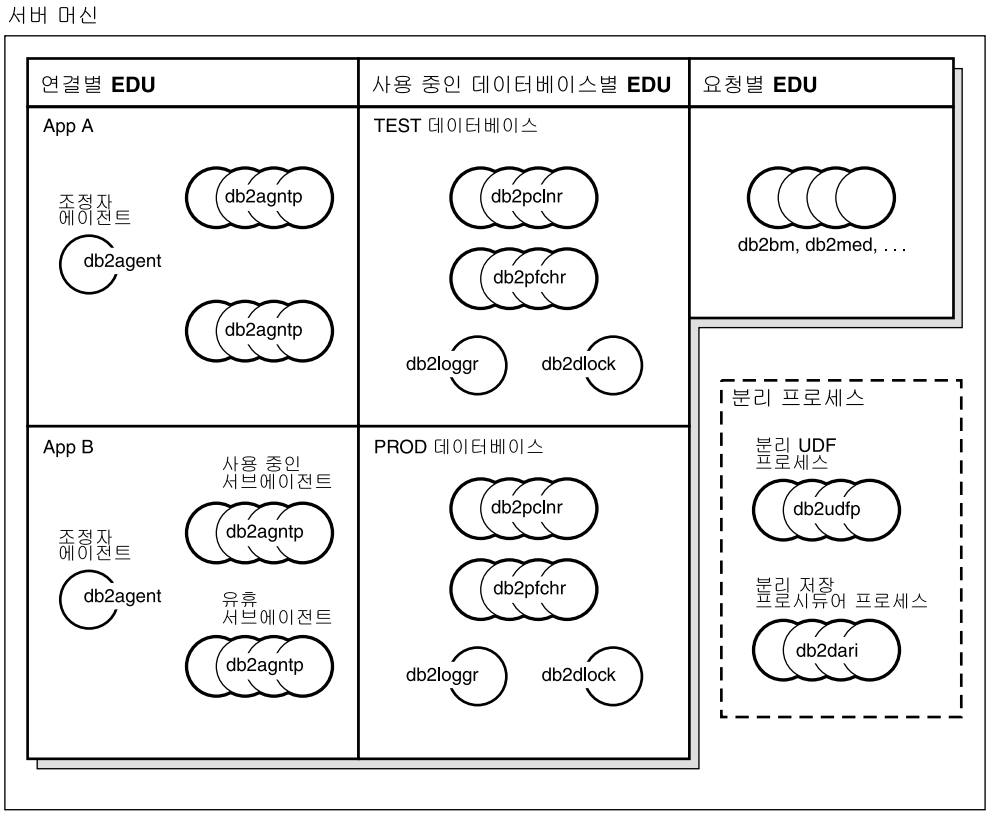


그림 8. 프로세스 모델 부분 2

이 그림은 서버 머신 환경의 부분인 추가 EDU를 보여줍니다. 사용 중인 각 데이터베이스에는 프리페처(db2pfchr) 및 페이지 정리자(db2pclnr)에 대한 고유한 공유 풀과, 자체의 고유한 로그 프로그램(db2loggr) 및 교착 상태 검출기(db2dlock)가 있습니다.

그림의 오른쪽 하단에 있는 『db2udfp』 및 『db2dari』 원은 각각 분리(fenced) 사용자 정의 함수(UDF)와 저장 프로시듀어로 DB2 Universal Database 내에서 수행되는 프로세스를 나타냅니다. 이 프로세스들은 작성 및 파괴와 연관되는 비용을 최소화하기 위해 관리됩니다. *keepdari* 데이터베이스 관리 프로그램 구성 매개변수의 기본값은 『YES』로, 이 값은 다음 저장 프로시듀어 호출에서 다시 사용할 수 있도록 저장 프로시듀어 프로세스를 보존합니다.

주: 에이전트의 주소 공간에서 직접 수행되는 비분리 UDF 및 저장 프로시듀어도 있습니다. 이러한 방식으로 작업할 경우, 성능이 향상됩니다. 그러나 에이전트의 주소 공간에 대한 무제한 액세스가 발생할 수 있으므로 사용하기 전에 테스트해야 합니다.

자세한 내용은 *응용프로그램 개발 안내서*을 참조하십시오.

다중 파티션 처리 모델은 단일 파티션 처리 모델의 논리적 확장입니다. 사실, 단일 공통 코드 기본은 두 가지의 조작 모드를 모두 지원합니다. 다음 그림은 이전의 두 그림에서 본 단일 파티션 처리 모델과 다중 파티션 처리 모델 사이의 유사점과 차이점을 보여줍니다.

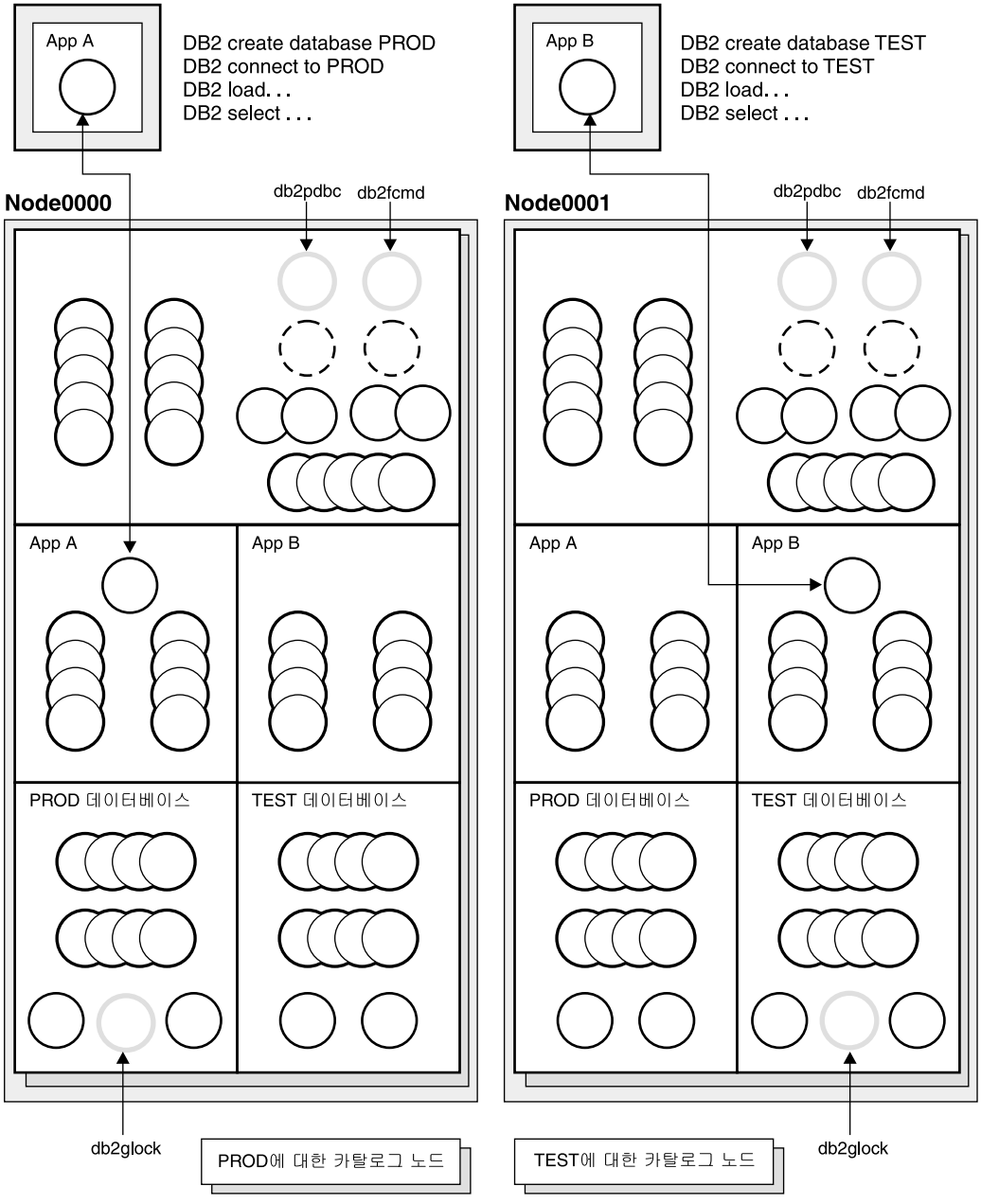


그림 9. 프로세스 모델 및 다중 파티션

대다수의 EDU는 단일 파티션 처리 모델과 다중 파티션 처리 모델 사이에 같습니다.

다중 파티션(또는 노드) 환경에서 파티션 중 하나는 카탈로그 노드입니다. 카탈로그는 데이터베이스에서 오브젝트에 관련되는 모든 정보를 보존합니다.

위의 그림에 표시된 것처럼 응용프로그램 A는 Node0000에서 PROD 데이터베이스를 작성하므로, PROD 데이터베이스에 대한 카탈로그도 이 노드에서 작성됩니다. 마찬가지로, 응용프로그램 B는 Node0001에서 TEST 데이터베이스를 작성하므로, TEST 데이터베이스에 대한 카탈로그가 이 노드에서 작성됩니다. 항상 시스템 환경에 있는 노드에서 각 데이터베이스에 대한 카탈로그와 연관되는 여분의 활동을 균형있게 분산할 것을 원하므로, 서로 다른 노드에서 데이터베이스를 작성할 수 있습니다.

인스턴스와 연관되는 추가 EDU(db2pdbc 및 db2fcmd)가 있으며 이 EDU는 다중 파티션된 데이터베이스 환경에 있는 각 노드에서 발견됩니다. 이 EDU는 데이터베이스 파티션에서 요청을 조정하고 빠른 통신 관리 프로그램(FCM)을 사용하기 위해 필요합니다.

또한 데이터베이스에 대한 카탈로그 노드와 연관되는 추가 EDU(db2glock)도 있습니다. 이 EDU는 사용 중인 데이터베이스가 위치한 노드에서 전역 교착 상태를 제어합니다.

응용프로그램에서의 각 CONNECT는 데이터베이스에서 논리 에이전트에 의해 제시되고 단일 조정자 에이전트를 야기시킵니다. 조정자 에이전트는 응용프로그램이 연결된 파티션에 존재합니다. 그러면 이 파티션은 해당 응용프로그램에 대한 『조정자 노드』가 됩니다. 조정자 노드는 또한 *SET CLIENT CONNECT_NODE* 명령을 사용하여 설정될 수도 있습니다. 응용프로그램에서의 데이터베이스 요청 부분들은 다른 파티션에서 조정자 노드에 의해 서브에이전트로 전송되고, 다른 파티션에서의 모든 결과는 다시 응용프로그램으로 전송되기 전에 조정자 노드에서 통합됩니다.

CREATE DATABASE 명령이 발행된 데이터베이스 파티션을 해당 데이터베이스에 대한 『카탈로그 노드』라고 합니다. 이것은 카탈로그 테이블이 저장되는 해당 데이터베이스 파티션에 있습니다. 일반적으로, 모든 사용자 테이블은 노드 세트에서 파티션됩니다.

주: 몇 개의 파티션이라도 같은 머신에서 수행되도록 구성할 수 있습니다. 이를 『다중 논리 파티션』이나 『다중 논리 노드』 구성이라고 합니다. 그러한 구성은 주 메모리가 아주 큰 대형 SMP 머신에서 아주 유용합니다. 이 환경에서 파티션 사이의 통신은 공유 메모리와 세마포어를 사용하여 최적화할 수 있습니다.

메모리 모델

메모리는 데이터베이스 내에서 수행되는 작업량에 영향을 미치므로 중요합니다. 데이터베이스 내의 영역 사이에 사용 가능한 메모리를 나누는 방법은 데이터베이스가 얼마나 제대로 수행되는지를 제어하는 기본적인 방법입니다. 구성 매개변수를 통해 서로 다른 힙(heap) 사이의 메모리 나누기를 제어합니다. 이 절에는 이들 키 구성 매개변수와 그 매개변수가 제어하는 메모리 부분에 대해 설명합니다. 이 주제에 대한 자세한 내용은 274 페이지의 『DB2의 메모리 사용법』을 참조하십시오.

파티션의 모든 EDU(Engine Dispatchable Unit)는 인스턴스 공유 메모리에 접속됩니다. 데이터베이스 내에서 작업을 수행하는 모든 EDU는 해당 데이터베이스의 데이터베이스 공유 메모리에 접속됩니다. 특정 응용프로그램 대신 작업하는 모든 EDU는 해당 응용프로그램에 대한 응용프로그램 공유 메모리 영역에 접속됩니다. 이러한 유형의 공유 메모리는 파티션 내 또는 파티션 간 병렬 처리를 사용할 경우에만 할당됩니다. 마지막으로, 각 EDU에는 고유한 개인용 메모리가 있습니다.

데이터베이스 관리 프로그램 공유 메모리라고도 하는 인스턴스 공유 메모리는 데이터베이스가 시작될 때 할당됩니다. 인스턴스 공유 메모리에서 다른 모든 메모리가 접속/할당됩니다. FCM(Fast communication mgager)이 사용될 경우 이 메모리에서 버퍼가 사용됩니다. FCM은 특정 데이터베이스 환경에서 데이터베이스 서버 사이나 데이터베이스 서버 내에서의 내부 통신(기본적으로 메시지)에 사용됩니다. 첫 번째 응용프로그램이 데이터베이스에 연결되거나 접속될 때 데이터베이스 공유, 응용프로그램 공유 및 에이전트 개인용 메모리 영역이 할당됩니다.

데이터베이스 전역 메로리라고도 하는 데이터베이스 공유 메모리는 데이터베이스가 활성화되거나 처음으로 연결될 때 할당됩니다. 이 메모리는 데이터베이스에 연결할 수 있는 모든 응용프로그램에서 사용됩니다. 다음과 같이 서로 다른 많은 메모리 영역이 데이터베이스 공유 메모리에 포함됩니다.

- 버퍼 풀
- 잠금 목록
- 데이터베이스 힙(heap) - 로그 버퍼와 카탈로그 캐쉬가 포함됩니다.
- 유틸리티 힙(heap)
- 패키지 캐쉬

데이터베이스 관리 프로그램 구성 매개변수 *numdb*는 동시에 사용 중일 수 있는 지역 데이터베이스의 수를 지정합니다. 파티션된 데이터베이스 환경에서 이 매개변수는 데이터베이스 파티션 서버에서 사용 중인 데이터베이스 파티션 수를 제한합니다. *numdb* 매개변수의 값은 할당되는 총 메모리 양에 영향을 미칠 수도 있습니다.

응용프로그램 전역 메모리라고도 하는 응용프로그램 공유 메모리는 응용프로그램이 데이터베이스에 연결될 때 할당됩니다. 이러한 할당은 파티션된 데이터베이스 환경이나, 데이터베이스 관리 프로그램 구성 매개변수 *intra_parallel*을 사용할 경우에만 발생합니다. 이 메모리는 데이터를 공유하고 에이전트 사이에 활동을 조정하기 위해 응용프로그램 대신 에이전트 작업에 사용됩니다.

데이터베이스 구성 매개변수 *maxappls*는 상한을 데이터베이스에 연결하는 응용프로그램 수로 설정합니다. 데이터베이스에 접속되는 각 응용프로그램으로 인해 일부 개인용 메모리가 할당되므로, 동시적인 응용프로그램의 수를 크게 하면 더 많은 메모리를 사용할 수도 있습니다.

특정 extent에 대해 최대 응용프로그램 수도 데이터베이스 관리 프로그램 구성 매개변수 *maxagents*(또는, 파티션된 환경의 경우 *max_coordagents*)에 의해 제어됩니다. *maxagents* 매개변수는 상한을 파티션에 있는 총 데이터베이스 관리 프로그램 에이전트 수로 설정합니다. 이 데이터베이스 관리 프로그램 에이전트에는 활동 조정자 에이전트, 서브에이전트, 비활동 에이전트 및 유휴 에이전트가 포함됩니다.

에이전트 개인용 메모리는 해당 에이전트에 특정 응용프로그램에 대해 작업이 지정될 때 할당됩니다. 에이전트 개인용 메모리는 에이전트에 대해 할당되고, 정렬 힙 및 응용프로그램 힙과 같은 특정 에이전트에 의해서만 사용될 메모리 할당사항을 포함하고 있습니다.

몇 가지의 특수 공유 메모리 유형이 있습니다.

- 에이전트/지역 응용프로그램 공유 메모리. 이 메모리는 SQL 요청에 사용되며 에이전트와 해당 클라이언트 응용프로그램 사이의 통신에 응답합니다.
- UDF/에이전트 공유 메모리. 이 메모리는 분리(fenced) UDF나 저장 프로시듀어를 수행하는 에이전트에 의해 접속됩니다. 이는 통신 영역으로 사용됩니다.

- 확장된 저장영역. 확장된 버퍼 풀로 사용되는 공유 메모리의 매우 큰(4GB 이상) 영역. 에이전트/프리페처/페이지 정리자는 이 영역에 영구적으로 접속되지는 않지만, 필요에 따라 그 영역 내에서 개별 세그먼트에 접속됩니다.

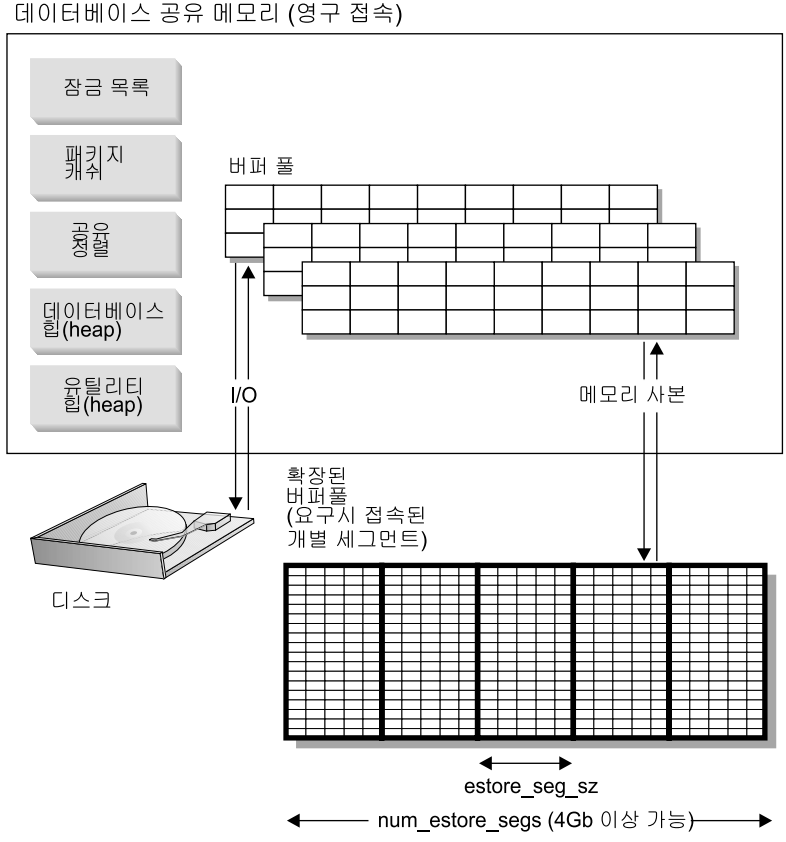


그림 10. 버퍼 풀 및 확장된 저장영역

확장 저장영역은 주 버퍼 풀에 대한 확장된 look-a-side 버퍼 역할을 합니다. 이 주제에 대한 자세한 내용은 321 페이지의 『메모리 확장』을 참조하십시오. 4GB보다 더 클 수도 있으며 주 메모리가 큰 머신을 이용할 수 있는 탁월한 방법입니다. 확장 저장영역 캐쉬는 메모리 세그먼트에 의해 정의됩니다.

실제 주소 지정 가능 메모리의 일부를 확장 저장영역 캐쉬로 사용하려고 할 때 이 메모리가 머신에서 더 이상 JFS(Journaled File System)저널된 파일시스템 캐쉬 또는 프로세스

개인 주소 공간과 같은 다른 목적에는 사용될 수 없다는 것에 주의해야 합니다. 추가 실제 주소 지정 가능 메모리를 확장 저장영역 캐쉬에 지정하면 시스템 페이징을 높일 수 있습니다.

다음 데이터베이스 구성 매개변수는 확장 저장영역에 사용 가능한 메모리 양 및 *크기*에 영향을 미칩니다.

- *num_estore_segs*는 확장 저장영역 메모리 세그먼트의 수를 정의합니다.
- *estore_seg_sz*는 각각의 확장 메모리 세그먼트의 *크기*를 정의합니다.

각 테이블 공간에는 버퍼 풀이 지정됩니다. 확장 저장영역 캐쉬는 하나 이상의 특정 버퍼 풀과 항상 연관되어 있어야 합니다. 확장 저장영역 캐쉬의 페이지 *크기*는 연관되어 있는 버퍼 풀의 페이지 *크기*와 일치해야 합니다.

확장 저장영역 캐쉬에 대한 자세한 내용은 321 페이지의 『메모리 확장』을 참조하십시오.

제2부 응용프로그램 성능 조정

© Copyright IBM Corp. 1993, 2001

제3장 응용프로그램 고려사항

응용프로그램의 런타임 성능에 영향을 미치는 많은 인수가 있습니다. 이 장에서는 응용프로그램을 설계하고 코딩할 때 고려해야 할 다음 주제 항목에 대해 설명합니다.

- 동시성
- 잠금
- 최적화 클래스 조정
- 성능 향상을 위한 결과 세트에 대한 제한사항
- 행 블로킹
- 조회 조정
- 복합 SQL
- 성능 고려사항 및 문자 변환
- 저장 프로시듀어
- 데이터베이스 활성화
- 응용프로그램의 병렬 처리

응용프로그램의 성능에 영향을 미치는 추가 정보에 대해서는 *응용프로그램 개발 안내서* 및 *CLI Guide and Reference*를 참조하십시오. 예를 들면, 다음과 같습니다.

- Embedded 정적 SQL을 사용하여 프로그램 작성
- Embedded 동적 SQL을 사용하여 프로그램 작성
- DB2 콜 레벨 인터페이스를 사용하여 프로그램 작성

동시성

관계형 데이터베이스에서 데이터의 무결성은 다중 사용자가 데이터에 액세스하고 변경할 때 유지보수되어야 합니다. *동시성*은 상호작용하는 다중 사용자 또는 응용프로그램에 의해 동시에 자원을 공유하는 것입니다. 데이터베이스 관리 프로그램은 예기치 않은 결과의 발생을 막기 위해 이 액세스를 제어합니다.

© Copyright IBM Corp. 1993, 2001

- *갱신사항 유실.* 응용프로그램 A와 응용프로그램 B는 모두 데이터베이스로부터 같은 행을 읽고, 이 응용프로그램이 읽은 데이터에 근거한 해당 컬럼 중의 하나의 컬럼에 대한 새 값을 계산합니다. A가 새 값으로 행을 갱신하면 B도 또한 행을 갱신하여 A에 의해 수행된 갱신은 유실됩니다.
- *미확약된 데이터에 대한 액세스.* 응용프로그램 A는 데이터베이스의 값을 갱신하고 응용프로그램 B는 값이 확약되기 전에 해당 값을 읽습니다. 그러면 A의 값이 나중에 확약되지 않고 백아웃(back out)되는 경우, B에 의해 수행된 계산은 미확약된(아마도 유효하지 않은) 데이터에 근거합니다.
- *반복 불가능한 읽기.* 일부 응용프로그램은 다음과 같은 이벤트 순서를 포함합니다. 응용프로그램 A는 데이터베이스로부터 행을 읽은 다음, 다른 SQL 요청의 처리를 계속합니다. 한편에서, 응용프로그램 B는 행을 변경하거나 삭제하고 변경사항을 확약합니다. 이후에, 응용프로그램 A가 원래 행을 다시 읽으려고 하는 경우, 수정된 행을 수신하거나 원래 행이 삭제된 것을 발견합니다.
- *팬텀 읽기 현상.* 팬텀 읽기 현상은 다음과 같은 경우에 발생합니다.
  1. 응용프로그램은 검색 기준에 근거한 행 세트를 읽는 조회를 실행합니다.
  2. 다른 응용프로그램은 새 데이터를 삽입하거나 사용자 응용프로그램의 조회에 부합하는 기존의 데이터를 갱신합니다.
  3. 응용프로그램은 1단계로부터 (같은 작업 단위(UOW) 안에서) 조회를 반복합니다.

  조회가 반복되면(3단계) 몇 가지의 추가(『팬텀』) 행이 조회가 처음 실행되는 경우(1단계)에 리턴되지 않은 결과 세트의 부분으로 리턴됩니다.

*분리 레벨*은 데이터가 액세스 중인 다른 프로세스로부터 데이터가 잠겨진 또는 분리된 방식을 결정합니다. 분리 레벨은 작업 단위(UOW)의 지속기간에 대해 유효합니다. WITH HOLD절을 사용하여 DECLARE CURSOR문으로 선언된 커서를 사용하는 응용프로그램은 OPEN CURSOR가 수행되는 작업 단위(UOW)의 지속기간에 대해 선택된 분리 레벨을 유지합니다. (자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오.) 분리 레벨의 지정 방법에 대한 자세한 내용은 52 페이지의 『분리 레벨 지정』을 참조하십시오.

DB2는 다음 분리 레벨을 지원합니다.
- 반복 읽기(RR)
- 읽기 안정성(RS)

- 커서 안정성(CS)
- 미확약 읽기(UR)

(일부 DRDA 데이터베이스 서버는 *확약 없는* 분리 레벨을 지원합니다. 기타 데이터베이스의 경우, 미확약 읽기(UR) 분리 레벨과 같이 작동합니다. 이 분리 레벨에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.)

또한 다음 내용을 참조하십시오.

- 51 페이지의『분리 레벨 선택』
- 52 페이지의『분리 레벨 지정』.

하나의 명령문에서 둘 이상의 데이터베이스 관리 시스템(DBMS) 또는 데이터베이스를 참조하는 SQL문을 제출하는 응용프로그램과 사용자를 지원하는 *연합 데이터베이스 시스템*에서 작업을 할 수 있습니다. DB2 연합 시스템은 데이터베이스 오브젝트에 대해 *위치 투명성*을 제공합니다. 예를 들어, 테이블 및 뷰에 대한 정보가 이동하면 해당 정보에 대한 참조(*별칭*이라고 함)는 정보를 요청한 응용프로그램을 변경하지 않고도 갱신될 수 있습니다. 응용프로그램이 별칭에 액세스할 때 DB2는 데이터 소스 데이터베이스 관리 프로그램의 동시처리 제어 프로토콜에 의존하여 분리 레벨을 확인합니다. (*데이터 소스*는 DBMS 및 데이터로 구성됩니다.) DB2는 논리적으로 동등한 데이터 소스에서 요청된 분리 레벨과 일치시키려고 하지만, 결과는 데이터 소스 성능에 따라 다를 수도 있습니다. 별칭에 액세스하는 응용프로그램 작성에 대한 자세한 내용은 *응용프로그램 개발 안내서* 매뉴얼을 참조하십시오.

각 분리 레벨에 대한 자세한 설명은 성능 영향이 낮아지는 순서로, 하지만 데이터에 액세스하여 갱신할 때 요구되는 주의가 커지는 순서로 나옵니다.

반복 읽기(RR)

반복 읽기(RR)는 작업 단위(UOW) 안에서 응용프로그램이 참조하는 모든 행을 잠급니다. 반복 읽기(RR)를 사용하여 커서가 열린 같은 작업 단위(UOW) 안에서 응용프로그램에 의해 두 번 발행된 SELECT문은 매번 같은 결과를 가져옵니다. 반복 읽기(RR)를 통해 갱신사항 유실, 미확약된 데이터에 대한 액세스 및 팬텀 행이 가능하지 않습니다.

반복 읽기(RR) 응용프로그램은 작업 단위(UOW)가 완료할 때까지 필요한 시간 만큼의 행에서 검색하고 조작할 수 있습니다. 그러나 다른 응용프로그램은 작업 단위(UOW)가 완료할 때까지 결과 테이블에 영향을 주는 행을 갱신, 삭제 또는 삽입할 수 없습니다. 반복 읽기(RR) 응용프로그램은 다른 응용프로그램의 미확약된 변경사항을 볼 수 없습니다.

반복 읽기(RR)를 통해 참조된 모든 행은 검색된 행과는 다르게 잠겨집니다. 적합한 잠금이 수행되어 다른 응용프로그램은 사용자 조회에 의해 참조된 행 목록에 추가된 행을 조회가 재실행되는 경우에 삽입하거나 갱신할 수 없습니다. 이렇게 하면, 팬텀 행은 발생하지 않습니다. 10 000행을 스캔하고 술어를 이 행에 적용하는 경우, 10행만이 규정되라도 모든 10 000행에서 잠금을 보유합니다.

주: 반복 읽기(RR) 분리 레벨을 통해 리턴된 모든 데이터는 응용프로그램이 데이터를 보고, 비록 임시 테이블이나 행 블로킹이 사용되더라도 변경되지 않은 상태로 남아 있습니다.

반복 읽기(RR)는 많은 잠금 수를 필요로 하고 보유하기 때문에, 이러한 잠금은 *locklist* 및 *maxlocks* 구성 매개변수의 결과로서 사용 가능한 잠금 수를 초과할 수 있습니다. (443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』 및 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』을 참조하십시오.) 최적화 알고리즘은 잠금 레벨 자동 업그레이드를 방지하기 위해 잠금 레벨 자동 업그레이드가 발생할 것 같은 경우 색인 스캔용 단일 테이블 레벨 잠금을 즉시 확보하도록 선택할 수 있습니다. (잠금 레벨 자동 업그레이드에 대한 자세한 내용은 61 페이지의 『잠금 레벨 자동 업그레이드』를 참조하십시오.) 데이터베이스 관리 프로그램이 사용자 대신으로 LOCK TABLE문을 발행하는 경우에 작동합니다. 확보된 테이블 레벨 잠금을 원하지 않으면 트랜잭션에서 사용 가능한 충분한 잠금이 있는지 확인하거나 읽기 안정성(RS) 분리 레벨을 사용하십시오.

읽기 안정성(RS)

읽기 안정성(RS)은 응용프로그램이 작업 단위(UOW) 안에서 검색하는 행만을 잠급니다. 작업 단위(UOW) 중 모든 규정 행 읽기는 작업 단위(UOW)가 완료되지 않을 때까지 다른 응용프로그램 프로세스에 의해 변경되지 않습니다. 그리고 다른 응용프로그램에 의해 변경된 행은 변경이 해당 프로세스에 의해 확약될 때까지 읽을 수 없습니다. 즉, 『반복 불가능한 읽기』 작동은 가능하지 **않습니다**.

반복 읽기(RR)와는 달리, 읽기 안정성(RS)을 통해 응용프로그램이 두 번 이상 같은 조회를 발행하는 경우, 추가 *팬텀 행*(*팬텀 읽기 현상*)을 볼 수 있습니다. 10 000 행을 스캔하는 예를 재호출하면 읽기 안정성(RS)은 규정하는 행만을 잠급니다. 그러므로 읽기 안정성(RS)을 통해 10행만이 검색되고 이 10행에서만 잠금이 보유됩니다. 이 예의 내용인, 잠금이 10 000행에서 모두 보유되는 반복 읽기(RR)와 대조하십시오. 보유된 잠금은 공유 상태, 다음 공유 상태, 갱신 또는 독점적 잠금일 수 있습니다. (잠금 속성에 대한 자세한 내용은 56 페이지의 『잠금 속성』을 참조하십시오.)

주: 읽기 안정성(RS) 분리 레벨을 통해 리턴된 모든 데이터는 응용프로그램이 데이터를 *보고*, 비록 임시 테이블이나 행 블로킹이 사용되더라도 변경되지 않은 상태로 남아 있습니다.

읽기 안정성(RS) 분리 레벨의 목적 중 하나는 데이터의 안정적 뷰뿐만 아니라 높은 동시성을 모두 제공하는 것입니다. 이 목적을 달성하려면 최적화 알고리즘에서는 잠금 레벨 자동 업그레이드가 발생할 때까지 테이블 레벨 잠금을 확보할 수 없습니다. (잠금 레벨 자동 업그레이드에 대한 자세한 내용은 61 페이지의 『잠금 레벨 자동 업그레이드』를 참조하십시오.)

읽기 안정성(RS) 분리 레벨은 다음을 모두 포함한 응용프로그램에 가장 적합합니다.

- 동시 환경에서 조작
- 작업 단위(UOW)의 지속기간에 대해 안정적인 규정 행이 필요
- 작업 단위(UOW) 안에서 같은 조회를 두 번 이상 발행하지 않거나, 같은 작업 단위(UOW)에서 두 번 이상 발행된 같은 응답을 조회가 확보할 필요는 없습니다.

커서 안정성(CS)

커서 안정성(CS)은 커서가 행에 위치하는 동안 응용프로그램의 트랜잭션에 의해 액세스된 모든 행을 잠급니다. 이 잠금은 다음 행이 페치되거나 트랜잭션이 종료될 때까지 유효한 상태로 남아 있습니다. 그러나 행의 데이터가 변경된 경우, 잠금은 변경이 데이터베이스에 확약될 때까지 보유되어야 합니다.

다른 응용프로그램은 갱신 가능한 커서가 행에 위치하는 동안 커서 안정성(CS) 응용프로그램이 검색한 행을 갱신하거나 삭제할 수 없습니다. 커서 안정성(CS) 응용프로그램은 다른 응용프로그램의 미확약된 변경사항을 볼 수 없습니다.

커서 안정성(CS)을 사용하는 경우, 10 000행을 스캔하는 예를 재호출하면, 현재의 커서 위치 아래 행에 잠금만 있게 됩니다. 해당 행을 이동하는 경우(해당 행을 갱신하지 않는 경우) 잠금은 제거됩니다.

커서 안정성(CS)을 통해 반복 불가능 읽기와 팬텀 읽기 현상 모두 가능합니다. 커서 안정성(CS)은 기본 분리 레벨이고, 다른 응용프로그램으로부터의 확약된 행을 보는 동안 최대의 동시성을 원하는 경우 사용되어야 합니다.

미확약 읽기(UR)

미확약 읽기(UR)를 사용하여 응용프로그램은 다른 트랜잭션의 미확약된 변경사항에 액세스합니다. 다른 응용프로그램이 테이블을 삭제하거나 변경하지 않으면 응용프로그램은 읽는 중인 행의 밖으로 다른 응용프로그램을 잠그지 않습니다. 미확약 읽기(UR)는 읽기 전용 커서와 갱신 가능한 커서에서는 다르게 작업합니다.

읽기 전용 커서는 다른 트랜잭션의 미확약된 변경사항에 대부분 액세스할 수 있습니다. 그러나 다른 트랜잭션에 의해 작성되거나 삭제 중인 테이블, 뷰, 색인은 트랜잭션이 처리중인 동안에는 사용 가능하지 않습니다. 다른 트랜잭션에 의한 변경사항은 확약되거나 구간 복원되기 전에 읽을 수 있습니다.

주: 미확약 읽기(UR) 분리 레벨 아래에서 조작 중인 갱신 가능한 커서는 마치 분리 레벨이 커서 안정성(CS)인 것처럼 작동합니다.

분리 레벨 UR을 사용하여 응용프로그램을 수행할 때 응용프로그램은 분리 레벨 CS를 사용하여 수행할 수 있습니다. 이런 현상은 응용프로그램에서 커서가 불명확하게 사용되었기 때문입니다. 불명확한 커서는 블로킹 옵션으로 인해 분리 레벨 CS로 자동 레벨 업그레이드될 수 있습니다. BLOCKING 옵션의 기본값은 UNAMBIG입니다. 이것은 불명확한 커서가 분리 레벨의 CS 단계로 갱신될 수 있고 레벨 자동 업그레이드되는 것을 의미합니다. 이러한 레벨 자동 업그레이드를 방지하기 위한 두가지 선택사항이 있습니다. 첫 번째 선택사항은 응용프로그램에서 커서를 수정하여 커서가 FOR READ ONLY절을 포함하도록 SELECT문을 수정하여 명확하게 합니다. 두 번째 선택사항은 커서가 응용프로그램에서 왼쪽으로 불명

확하면 그 프로그램을 사전 처리 컴파일하거나 전체 BLOCKING ALL 지정으로 바인드해야 합니다. 이러한 방법은 프로그램을 수행할 때 모든 불명확한 커서가 읽기 전용으로 처리되도록 합니다.

미확약 읽기(UR)를 사용하는 경우 10 000행을 스캔하는 예를 재호출하면 행 잠금을 획득하지 않습니다.

미확약 읽기(UR)를 통해 반복 불가능 읽기 작동과 팬텀 읽기 현상 모두 가능합니다.

미확약 읽기(UR) 분리 레벨은 읽기 전용 테이블에 대한 조회에, 또는 Select 문만 실행하고 다른 응용프로그램로부터 미확약된 데이터를 보는 것을 신경 쓰지 않는 경우에 일반적으로 가장 많이 사용됩니다.

분리 레벨 선택

표1에서는 *응용프로그램 개발 안내서* 매뉴얼에 설명된 원하지 않는 결과에 의한 다른 분리 레벨을 요약합니다.

표 1. 분리 레벨 요약

분리 레벨	미확약된 데이터에 대한 액세스	반복 불가능 읽기	팬텀 읽기 현상
반복 읽기(RR)	가능하지 않음	가능하지 않음	가능하지 않음
읽기 안정성(RS)	가능하지 않음	가능하지 않음	가능함
커서 안정성(CS)	가능하지 않음	가능함	가능함
미확약 읽기(UR)	가능함	가능함	가능함

표2에서는 응용프로그램에 대해 초기 분리 레벨 선택에 도움이 되는 간단한 경험(heuristic)을 설명합니다. 이 테이블을 시작점으로 간주하며, 사용자의 요구사항에 더 적합한 다른 값이 되는 인수에 대한 다양한 레벨의 이전 논의를 참조하십시오.

표 2. 분리 레벨 선택 지침

응용프로그램 유형	필요한 높은 데이터 안정성	필요하지 않은 높은 데이터 안정성
읽기/쓰기 트랜잭션	RS	CS
읽기 전용 트랜잭션	RR 또는 RS	UR

응용프로그램에 대한 적합한 분리 레벨을 선택하는 것은 해당 응용프로그램에서 허용되지 않는 현상을 피하기 위해 매우 중요합니다. 분리 레벨은 응용프로그램간의 분리 등급뿐만 아니라, 잠금을 얻고 해제하는 데 필요한 CPU와 메모리 자원이 분리 레벨에 따라 달라지므로 개별 응용프로그램의 성능 특성에도 영향을 줍니다. 또한 잠재적인 교착 상태도 분리 레벨에 따라 달라집니다.

분리 레벨 지정

분리 레벨은 사전 처리 컴파일 시간이나 응용프로그램이 데이터베이스에 바인드될 때 지정됩니다. 지원된 컴파일 언어로 작성된 응용프로그램의 경우, 명령행 처리기 PREP 또는 BIND 명령의 ISOLATION 옵션을 사용하십시오. 분리 레벨은 PREP 또는 BIND API를 사용하여 지정될 수도 있습니다.

바인드 파일이 사전 처리 컴파일 시간에 작성된 경우, 분리 레벨은 바인드 파일에 저장됩니다. 분리 레벨이 바인드 시간에 지정되지 않은 경우, 기본값은 사전 처리 컴파일 동안 사용된 분리 레벨입니다.

지정된 분리 레벨이 없는 경우, 커서 안정성(CS)의 기본값이 사용됩니다.

다음 조회를 실행하여 패키지의 분리 레벨을 결정할 수 있습니다.

```
SELECT ISOLATION FROM SYSCAT.PACKAGES
  WHERE PKGNAME = 'XXXXXXXX'
  AND PKGSCHEMA = 'YYYYYYYY'
```

여기서 *XXXXXXXX*는 패키지 이름이며, *YYYYYYYY*는 패키지의 스키마 이름입니다. 이름은 모두 대문자여야 합니다.

데이터베이스가 작성되면 REXX의 SQL용 다른 분리 레벨을 지원하는 데 사용된 다중 바인드 파일은 데이터베이스(REXX를 지원하는 해당 서버에서)에 바인드됩니다. 또한 데이터베이스가 작성될 때 다른 명령행 처리기 패키지도 데이터베이스에 바인드됩니다. 바인드 파일에 대한 자세한 내용은 *응용프로그램 개발 안내서*을 참조하십시오.

REXX와 명령행 처리기는 커서 안정성(CS)의 기본 분리 레벨을 사용하여 데이터베이스에 연결합니다. 다른 분리 레벨에 대한 변경사항은 연결 상태를 변경하지 않습니다. CONNECTABLE AND UNCONNECTED 상태 또는 IMPLICITLY

CONNECTABLE 상태에서 실행되어야 합니다. (연결 상태에 대한 자세한 내용은 *SQL 참조서*의 CONNECT TO문을 참조하십시오.)

사용 중인 분리 레벨은 SQLISL REXX 변수 값을 확인하여 REXX 응용프로그램에 의해 점검될 수 있습니다. CHANGE SQLISL 명령이 실행될 때마다 값은 갱신됩니다.

DB2_RR_TO_RS 프로파일 레지스트리 변수를 사용하여 다음 키 잠금을 최소화하고 동시성 및 성능을 증가시킬 수 있습니다. 그러나 DB2_RR_TO_RS 프로파일 레지스트리 변수의 사용으로 반복 읽기(RR) 분리 레벨을 방지합니다. 모든 응용프로그램을 데이터베이스에 액세스할 때 RR 의미를 요구하지 않으려면 변수를 "예"로 설정하는 것이 좋습니다. 변수를 유효하게 하려면 데이터베이스 관리 프로그램을 중지시킨 다음 시작해야 합니다. `db2start`에 이어 이 변경은 전체 인스턴스에 영향을 줍니다. 변수를 설정한 후 RR을 사용하여 사용자 테이블에 액세스하는 요청을 수신하면 대신 일기 안정성(RS) 분리 레벨을 사용하도록 요청이 내부적으로 수정됩니다. 이 경우, 경고가 제공되지 않습니다.

응용프로그램을 준비하거나 바인딩할 때 패키지 레벨에 분리 레벨을 설정하는 것 것 이외에도, 명령문 레벨에 분리 레벨을 설정할 수 있습니다. 명령문 레벨의 분리 레벨은 WITH-절을 사용하여 지정됩니다.

다음은 SQL문이 명령문 레벨 분리를 지원합니다.

- SELECT문
- SELECT INTO
- 검색된 DELETE
- INSERT
- 검색된 UPDATE
- DECLARE CURSOR

명령문 레벨 분리의 사용과 연관된 같은 조건이 있습니다.

- WITH-절은 부속 조회에 사용될 수 없습니다.
- WITH UR 옵션은 읽기 전용 조작에만 적용됩니다. 다른 상황에서 WITH UR 옵션이 사용되면 명령문은 자동으로 『UR』에서 『CS』로 변경됩니다.

- 명령문에 대한 기본 분리 레벨은 명령문이 바인드된 패키지의 분리 레벨입니다.
- 명령문 레벨의 분리 레벨은 명령문이 나타나는 패키지에 대해 지정된 분리 레벨을 겹쳐씁니다.

명령행 처리기를 사용하는 경우, CHANGE ISOLATION LEVEL 명령을 사용하여 분리 레벨을 변경할 수 있습니다. 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

DB2 콜 레벨 인터페이스(DB2 CLI)의 경우, DB2 CLI 구성 부분으로 분리 레벨을 변경할 수 있습니다.
런타임 시 CLI 내에서 속성이 SQL_ATTR_TXN_ISOLATION인 SQLSetConnectAttr 함수를 사용합니다. 이는 *ConnectionHandle*에 의해 참조되는 현재 연결에 대한 트랜잭션 분리 레벨을 설정합니다. db2cli.ini 파일 내에서 TXNISOLATION 키워드를 사용할 수도 있습니다.

주: JDBC 및 SQLJ는 DB2에서 CLI를 사용하여 구현됩니다. 이는 db2cli.ini 설정이 JDBC 및 SQLJ를 사용하여 수행되고 기록되는 것에 영향을 미칠 수도 있음을 의미합니다. 자세한 내용은 *CLI Guide and Reference* 매뉴얼을 참조하십시오.

런타임 시 JDBC나 SQLJ에 대해 작업할 때 java.sql 인터페이스 연결 내에서 setTransactionIsolation 메소드를 사용하여 분리 레벨을 설정할 수 있습니다. 자세한 내용은 *응용프로그램 개발 안내서* 매뉴얼의 『Java에서의 프로그래밍』 장에서 참조하십시오.

SQLJ에 대해 작업할 때 db2profc SQLJ 최적화 알고리즘을 수행하면 패키지가 작성됩니다. 사용될 분리 레벨을 포함하도록 이 패키지에 대한 최종 옵션을 지정할 수 있습니다. 자세한 내용은 *응용프로그램 개발 안내서* 매뉴얼의 『Java에서의 프로그래밍』 장을 참조하십시오.

또한 상업적으로 많이 작성된 응용프로그램은 분리 레벨을 선택할 수 있는 메소드도 제공합니다. 자세한 내용은 *CLI Guide and Reference* 매뉴얼을 참조하십시오.

선언된 임시 테이블 및 동시성

선언된 임시 테이블은 선언된 응용프로그램만 사용할 수 있으므로 동시성 관련 문제가 없습니다. 이러한 유형의 테이블은 응용프로그램이 이를 선언하는 시점으로부터 응용프로그램이 완료되거나 연결 해제될 때까지만 존재합니다.

잠금

데이터베이스 관리 프로그램은 동시처리 제어를 제공하고 잠금을 사용하여 제어되지 않은 액세스를 방지합니다. *잠금*은 다른 응용프로그램이 동일한 자원에 액세스할 수 있는 방법을 제어하기 위해 데이터베이스 관리 프로그램 자원을 응용프로그램과 연관시키는 수단입니다. 연관된 자원을 가진 응용프로그램은 잠금을 보유하거나 소유한다고 할 수 있습니다.

데이터베이스 관리 프로그램은 다른 응용프로그램에 의해 작성된 미확약된 데이터에 대한 액세스를 차단하기 위해 잠금을 부여합니다(미확약 읽기(UR) 분리 레벨이 사용되지 않은 경우). 이 원칙은 데이터 무결성(즉, 데이터의 동시성과 보안)을 보호합니다. 잠금은 또한 행을 갱신(반복 읽기(RR) 응용프로그램의 경우처럼)할 수 없습니다.

데이터 무결성을 충족시키기 위해 데이터베이스 관리 프로그램은 데이터베이스 관리 프로그램 제어하에서 내재적으로 잠금을 획득합니다. 미확약 읽기(UR) 분리 레벨의 경우를 제외하면, 응용프로그램이 미확약 데이터를 다른 프로세스로부터 숨기는 것을 확인하기 위해 내재적으로 잠금을 요청할 필요가 없습니다.

잠금의 기본 원칙으로 인해 대부분의 경우에 잠금을 제어하는 조치를 취할 필요는 없습니다. 여전히 응용프로그램은 일반 매개변수에 기반하여 잠금을 획득합니다. 지역 상황을 알면 이런 매개변수를 변경하여 시스템 자원을 더 효율적으로 사용할 수 있습니다. 사용자를 지원하기 위해 잠금에 대해 다음 주제가 논의됩니다.

- 잠금 속성
- 잠금 및 응용프로그램 성능
- 잠금에 영향을 주는 인수
- LOCK TABLE문
- 릴리스로 커서 닫기
- 잠금 고려사항 요약.

잠금 속성

데이터베이스 관리 프로그램 잠금은 다음 기본 속성을 갖습니다.

모드 잠긴 오브젝트의 동시 사용자에 대해 허용된 액세스 유형뿐만 아니라 잠금 소유자에 대해 허용된 액세스 유형. 가끔 잠금의 *상태*로서 나타냅니다.

오브젝트

잠금되는 자원. 명시적으로 잠금 가능한 유일한 오브젝트 유형은 테이블입니다. 또한 데이터베이스 관리 프로그램은 행, 테이블 및 테이블 공간과 같은 다른 유형의 자원에 잠금을 부여합니다. 잠금되는 오브젝트는 잠금의 *세분성*을 나타냅니다.

지속기간

잠금이 유지되는 기간. 잠금 지속기간은 45 페이지의 『동시성』에서 설명한 분리 레벨에 의해 영향을 받습니다.

다음 표에서 모드와 그 영향은 자원에 대한 제어가 증가하는 순서로 표시됩니다.

표 3. 잠금 모드 요약

잠금 모드	적용 가능한 오브젝트 유형	설명
IN(잠금 없음)	테이블 공간, 테이블	잠금 소유자는 미확약된 데이터를 포함하여 테이블의 어떤 데이터든지 읽을 수 있지만, 갱신할 수는 없습니다. 잠금 소유자가 획득한 행 잠금은 없습니다. 다른 동시 응용프로그램은 테이블을 읽거나 갱신할 수 있습니다.
IS(부분 공유)	테이블 공간, 테이블	잠금 소유자는 잠긴 테이블에서 데이터를 읽을 수는 있지만, 이 데이터를 갱신할 수는 없습니다. 응용프로그램이 IS 테이블 잠금을 보유하는 경우, 응용프로그램은 읽는 행에서 S 잠금 또는 NS 잠금을 획득합니다. 두 가지 경우에서 다른 동시 응용프로그램은 테이블을 읽거나 갱신할 수 있습니다.
NS(다음 키 공유)	행	잠금 소유자와 모든 동시 응용프로그램은 잠긴 행을 읽을 수는 있지만, 갱신할 수는 없습니다. 이 잠금을 S 잠금 대신 테이블 행에서 획득합니다. 이 때 분리 레벨은 응용프로그램에 대해 RS 또는 CS 중 하나입니다.
S(공유)	행, 테이블	잠금 소유자와 모든 동시 응용프로그램은 잠긴 데이터를 읽을 수는 있지만, 갱신할 수는 없습니다. 테이블의 개별 행에서 S 잠금이 수행될 수 있습니다. 테이블이 S 잠금이 된 경우, 행 잠금은 필요하지 않습니다.

표 3. 잠금 모드 요약 (계속)

잠금 모드	적용 가능한 오브젝트 유형	설명
IX(부분 독점)	테이블 공간, 테이블	잠금 소유자와 동시 응용프로그램은 테이블의 데이터를 읽고 갱신할 수 있습니다. 잠금 소유자가 데이터를 읽는 경우, 행마다 S, NS, X 또는 U 잠금을 획득합니다. 잠금 소유자가 갱신하는 각 행에서는 X 잠금도 확보합니다. 다른 동시 응용프로그램은 테이블을 읽고 갱신할 수 있습니다.
SIX(부분 독점이 있는 공유)	테이블	잠금 소유자는 테이블의 데이터를 읽고 갱신할 수 있습니다. 잠금 소유자는 갱신하는 행에서 X 잠금을 획득하지만, 읽는 행에서는 어떤 잠금도 획득하지 않습니다. 다른 동시 응용프로그램은 테이블을 읽을 수 있습니다.
U(갱신)	행, 테이블	잠금 소유자는 잠긴 행이나 테이블의 데이터를 갱신할 수 있습니다. 잠금 소유자는 행을 갱신하기 전에 그 행에서 X 잠금을 획득합니다. 다른 작업 단위(UOW)는 잠긴 행이나 테이블에서 데이터를 읽을 수는 있지만 갱신할 수는 없습니다.
NX(다음 키 독점)	행	잠금 소유자는 잠긴 행을 읽을 수는 있지만, 갱신할 수는 없습니다. 이 모드는 NS 잠금과 호환되는 것을 제외하고는 X 잠금과 유사합니다.
NW(다음 키 약한 독점)	행	행이 카탈로그가 아닌 테이블의 색인으로 삽입되는 경우, 이 잠금을 다음 행에서 획득합니다. 잠금 소유자는 잠긴 행을 읽을 수는 있지만, 갱신할 수는 없습니다. 이 모드는 W 및 NS 잠금과 호환되는 것을 제외하고는 X 및 NX 잠금과 유사합니다.
X(독점)	행, 테이블	잠금 소유자는 잠긴 행이나 테이블의 데이터를 읽고 갱신할 수 있습니다. 테이블에서 독점 잠금이 수행될 수 있습니다. 이는 해당 테이블의 행에서 어떤 행 잠금도 획득하지 않음을 의미합니다. 미확약 읽기(UR) 응용프로그램만이 잠긴 테이블에 액세스할 수 있습니다.
W(약한 독점)	행	행이 카탈로그가 아닌 테이블로 삽입되는 경우, 이 잠금을 행에서 획득합니다. 잠금 소유자는 잠긴 행을 변경할 수 있습니다. NW 잠금과 호환성이 되는 것을 제외하고 이 잠금은 X 잠금과 유사합니다. 미확약 읽기(UR) 응용프로그램만이 잠긴 행에 액세스할 수 있습니다.
Z(강한 독점)	테이블 공간, 테이블	테이블이 변경되거나 삭제된 경우, 테이블의 색인이 작성되거나 삭제되는 경우 또는 테이블이 재인식되는 경우와 같은 조건의 테이블에서 이 잠금을 획득합니다. 다른 동시 응용프로그램은 테이블을 읽거나 갱신할 수 없습니다.

주: 테이블과 테이블 공간만이『intent』잠금 모드를 확보할 수 있습니다. 즉, 잠금 모드는 행에 대해 확보되지 않습니다.

잠금 및 응용프로그램 성능

응용프로그램 프로그래머는 잠금 사용과 사용에 따른 응용프로그램 성능에 미치는 영향과 관계있는 여러 관련 인수를 알고 있어야 합니다. 이 인수는 다음과 같습니다.

- 동시성 및 세분성
- 잠금 호환성
- 잠금 변환
- 잠금 레벨 자동 업그레이드
- 잠금 대기 및 시간종료
- 교착 상태

동시성 및 세분성

응용프로그램에 의해 보유된 잠금은 다른 응용프로그램에 의한 액세스를 할 수 없습니다. 그러므로 최대 동시성의 경우, 행 레벨 잠금은 테이블 잠금보다 좋습니다. 그러나 잠금에는 관리하기 위해서 저장 및 처리 시간이 필요합니다. 따라서 저장 및 처리 시간을 최소화하는 데에는 단일 테이블 잠금이 여러 행 잠금보다 좋습니다.

ALTER TABLE문의 LOCKSIZE절을 통해 행 또는 테이블 레벨에서 잠금 *크기* (세분성)를 정의할 수 있습니다. 기본적으로 행 잠금이 사용됩니다. ALTER TABLE에 정의된 영구 테이블 잠금에서는 S 및 X 테이블 잠금만이 사용됩니다. 응용프로그램은 행 잠금처럼 많이 획득하거나 해제할 필요가 없으므로 성능이 향상됩니다. 다음 경우에 LOCK TABLE문을 사용하는 단일 트랜잭션 테이블 잠금보다는 ALTER TABLE문을 사용하는 영구 테이블 잠금을 확보하는 것을 더 선호할 수 있습니다.

- 테이블은 읽기 전용이며, 항상 S 잠금을 필요로 합니다. 테이블 레벨 잠금은 테이블에서 S 잠금을 확보하게 하여 성능을 향상시킵니다.
- 유지보수를 위해 단일 사용자가 테이블에 액세스하며, 사용자는 제한된 기간 동안 X 잠금을 요구합니다. 테이블에서 ALTER TABLE을 통해 테이블 레벨 잠금을 변경하면 테이블 레벨에서 X 잠금이 제공됩니다. 일단 사용자가 완료하면 ALTER TABLE을 사용하여 테이블을 행 레벨 잠금으로 리턴할 수 있습니다.

ALTER TABLE문을 사용하면 정상적인 잠금 레벨 자동 업그레이드의 발생을 방지하지 못합니다.

그리고 ALTER TABLE을 사용하여 테이블 레벨까지 잠금을 밀어 올리는 것이 전역 접근으로, 해당 테이블을 사용하는 모든 응용프로그램과 사용자에게 영향을 줍니다. 개별 응용프로그램의 경우, 다른 선택사항은 LOCK TABLE문을 사용하는 것입니다. 이 명령을 사용하면 위의 두 번째 글머리표에 언급된 대로 데이터베이스 레벨이 아닌 응용프로그램 레벨에서 테이블 잠금으로 이동할 수 있습니다.

잠금 호환성

표4에서는 다른 프로세스가 주어진 상태에서 같은 자원의 잠금을 보유하거나 요청하는 경우 잠금 요청의 허용 여부를 나타냅니다. **아니오**는 호환되지 않은 모든 잠금이 다른 프로세스에 의해 해제될 때까지 리퀘스터가 대기해야 한다는 사실을 나타냅니다. 시간종료는 잠금 대기시 발생할 수 있습니다. **예**는 잠금이 권한 부여된 것(다른 사용자가 자원을 기다리지 않는 경우)을 나타냅니다.

표 4. 잠금 유형 호환성

	보유된 자원 상태												
요청 중인 상태	**없음**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**NX**	**X**	**Z**	**NW**	**W**
없음	예	예	예	예	예	예	예	예	예	예	예	예	예
IN	예	예	예	예	예	예	예	예	예	예	아니오	예	예
IS	예	예	예	예	예	예	예	예	아니오	아니오	아니오	아니오	아니오
NS	예	예	예	예	예	아니오	아니오	예	예	아니오	아니오	예	아니오
S	예	예	예	예	예	아니오	아니오	예	아니오	아니오	아니오	아니오	아니오
IX	예	예	예	아니오	아니오	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오
SIX	예	예	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오
U	예	예	예	예	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오
NX	예	예	아니오	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오
X	예	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오
Z	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오
NW	예	예	아니오	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	예
W	예	예	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	예	아니오

표 4. 잠금 유형 호환성 (계속)

요청 중인 상태	보유된 자원 상태												
	없음	IN	IS	NS	S	IX	SIX	U	NX	X	Z	NW	W

주:

I	잠금
N	없음
NS	다음 키 공유
S	공유
NX	다음 키 독점
X	독점
U	갱신
Z	강한 독점
NW	다음 키 약한 독점
W	약한 독점

이러한 잠금 유형에 대한 자세한 내용은 56 페이지의 『잠금 속성』을 참조하십시오.

주:

- 예 - 요청된 잠금을 즉시 **허용**
- 아니오 - 릴리스될 보유 잠금이나 발생할 시간종료에 대한 **대기**

응용프로그램 A는 응용프로그램 B 역시 액세스하려는 테이블에서 잠금을 보유한다고 가정합니다. 데이터베이스 관리 프로그램은 응용프로그램 B 대신 몇몇 특정 모드의 잠금을 요청합니다. A가 보유한 잠금 모드는 B가 요청한 잠금을 허용하고, 이 두 잠금(또는 모드)은 호환성이 있다고 할 수 있습니다.

응용프로그램 B에 대해 요청된 잠금 모드는 응용프로그램 A가 보유한 잠금과 호환되지 않는 경우, 응용프로그램 B는 계속 실행할 수 없습니다. 그 대신, 응용프로그램 A가 잠금을 해제하고 다른 모든 기존의 호환되지 않은 잠금이 해제될 때까지 대기해야 합니다.

잠금 변환

잠금 변환은 프로세스가 이미 잠금을 보유한 데이터 오브젝트에 액세스할 경우에 발생하고, 액세스 모드에는 이미 보유한 잠금보다 더 제한적인 잠금이 필요합니다. 같은 데이터 오브젝트에 여러 번 잠금을 (조회를 통해 간접적으로) 요청할 수 있지만, 프로세스는 항상 데이터 오브젝트에 하나의 잠금만을 보유할 수 있습니다. 이미 보유한 잠금 모드를 변경하는 조작을 *변환*이라고 합니다.

행에 발생하는 변환은 간단합니다. 예에서처럼 X가 필요하고 S 또는 U를 보유한 경우 변환이 발생합니다.

테이블과 행에 대해 고유한 잠금 모드가 있습니다. 그러나 IX(부분 독점) 및 S(공유) 잠금은 잠금 변환에 대해 특별한 경우입니다. S 또는 IX가 다른 잠금보다 더 제한적이라고 할 수는 없습니다. 이 잠금 중 하나를 보유하고 다른 잠금이 필요한 경우, 변환은 SIX(부분 독점이 있는 공유) 잠금에서 결과적으로 발생합니다. 모든 다른 변환은 요청된 모드가 덜 제한적인 경우, 요청된 잠금 모드가 보유된 잠금 모드가 되게 합니다.

행을 갱신하는 조회는 변환을 두 번 일으킬 수도 있습니다. 행이 색인 액세스를 통해 읽혀지고 S로 잠금되었다고 가정하십시오. 행을 포함하는 테이블에는 커버하는 잠금이 있습니다. 잠금 유형이 IX가 아닌 IS라고 가정하십시오. 그 결과로 행이 변경되는 경우, 테이블 잠금은 IX로 변환되고 행은 X로 변환됩니다.

상기한 바와 같이, 잠금 응용프로그램은 조회 실행시 흔히 내재적으로 발생합니다. 다른 조회와 테이블 및 색인 조합에서 얻어진 잠금의 종류를 알면 응용프로그램을 설계하고 조정하는 데 도움이 될 수 있습니다. 이 주제에 대한 자세한 내용은 66 페이지의 『잠금에 영향을 주는 인수』를 참조하십시오.

잠금 레벨 자동 업그레이드

잠금 레벨 자동 업그레이드는 보유할 잠금 수를 줄이는 내부 메커니즘입니다. 레벨 자동 업그레이드는 여러 행 잠금(단일 테이블에서)에서 단일 테이블 잠금까지 분포합니다.

잠금 레벨 자동 업그레이드는 현재 너무 많은 잠금(모든 유형의)을 보유한 경우에 발생합니다.

잠금 레벨 자동 업그레이드는 에이전트가 잠금 목록의 할당을 초과한 경우에 발생할 수 있습니다(443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』 참조).

이러한 레벨 자동 업그레이드는 내부적으로 처리됩니다. 외부적으로 검출할 수 있는 결과는 하나 이상의 테이블에서 동시 액세스가 감소한다는 것입니다. 보통, 적합하게 구성된 데이터베이스에서 잠금 레벨 자동 업그레이드는 거의 발생하지 않습니다.

잠금 레벨 자동 업그레이드는, 예를 들어, 응용프로그램 설계자가 대형 테이블의 색인을 사용하여 성능과 동시성을 증가시킬 경우에 발생할 수 있습니다. 그러나 응용프로그램은 테이블에서 높은 비율 레코드에 액세스합니다. 데이터베이스 관리 프로그램은 (이 경우에) 그렇게 많은 테이블이 잠기고, S 또는 X 테이블만 잠그는 것이 아니라 개별적으로 각 레코드도 잠그는 것을 예측할 수 없습니다. 이러한 경우에 대한 해결책으로 그리고 응용프로그램 설계자에게 문의한 후 데이터베이스 설계자는 이 트랜잭션에 대해 LOCK TABLE문을 사용하는 것이 좋습니다.

때때로, 레벨 자동 업그레이드 요청을 (내재적으로) 수신하는 프로세스는 모든 테이블에서 레코드 잠금을 거의 보유하지 않습니다. 이 레벨 자동 업그레이드에 대한 이유는 하나의 프로세스(또는 여러 개의 프로세스)가 많은 잠금을 보유할 수 있지만(프로세스당 잠금에서 잠금 수가 데이터베이스 구성 매개변수 값 이하이더라도), 레벨 자동 업그레이드 요청을 트리거할 만큼 충분한 것은 아닙니다. 프로세스는 다른 잠금을 요청하지 않거나 트랜잭션 종료를 제외한 데이터베이스에 다시 액세스합니다. 그런 다음 다른 프로세스는 레벨 자동 업그레이드 요청을 트리거하는 잠금을 요청할 수 있습니다.

잠금 레벨 자동 업그레이드로 인해 승인할 수 없는 레벨로 동시성이 줄어들 경우, 다음을 수행할 수 있습니다.

- *db2diag.log* 내용에서 레벨 자동 업그레이드에 대한 정보를 점검하십시오. 레벨 자동 업그레이드되는 각 테이블에 대해 정보가 기록됩니다. 기록되는 정보의 유형은 다음과 같습니다.
  - 현재 보유되는 잠금 수
  - 잠금 레벨 자동 업그레이드를 완료하는 데 필요한 잠금 수
  - 레벨 자동 업그레이드되는 각 테이블의 테이블 식별자 정보 및 테이블 이름
  - 현재 보유되는 테이블이 아닌 잠금의 수
  - 레벨 자동 업그레이드의 일부로 획득된 새 테이블 레벨 잠금. 일반적으로, 이는 『S』 또는 공유 잠금, 『X』 또는 독점 잠금일 수 있습니다.
  - 새 테이블 잠금 레벨 획득 결과로부터 발생된 내부 리턴 코드

  현재 동적 SQL문도 기록될 수 있습니다. DIAGLEVEL 데이터베이스 관리 프로그램 구성 매개변수가 4일 경우, 동적 SQL문이 기록되면 기록되는 정보에는 테이블 잠금 자동 업그레이드 앞의 현재 SQL문이 포함되게 됩니다.

DIAGLEVEL이 2 이상인 **경우**, 잠금 레벨 자동 업그레이드가 실패하면 기록되는 정보에는 레벨 자동 업그레이드에 실패한 테이블과 현재 SQL문(현재 사용 가능하며, 이전에 기록되지 않은 경우)이 포함됩니다.

이 정보를 사용하여 아래에서 언급하는 몇 가지 사항을 기준으로 적절한 조치를 수행할 수 있습니다.

이 유형의 정보를 기록하려면 데이터베이스 관리 프로그램 구성 매개변수 DIAGLEVEL을 3(기본값) 또는 4로 설정해야 합니다.

- 데이터베이스 구성 파일의 *maxlocks* 및/또는 *locklist* 매개변수의 값을 증가시켜 허용된 잠금 수를 증가시키십시오. (자세한 내용은 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』 및 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』을 참조하십시오.) 다른 프로세스에 의한 테이블로의 동시 액세스가 가장 중요한 경우에 이를 선택할 수 있습니다. 그러나 레코드 레벨 잠금 확보로 인해 오버헤드는 테이블로의 동시 액세스에 의해 저장된 프로세스보다 다른 프로세스에서 지연이 더 많이 발생할 수 있습니다. (파티션된 데이터베이스에서 이 매개변수를 변경 중이면 해당 매개변수가 모든 파티션에서 갱신되었는지 확인하십시오.)
- 레벨 자동 업그레이드 또는 구간 복원일 수도 있고 아닐 수도 있는 위반 프로세스의 위치를 찾아내고 조정하여 명시적으로 LOCK TABLE문을 발행하십시오.
- 분리(isolation) 등급을 변경하십시오. 이렇게 하면 동시성이 감소될 수도 있다는 점에 유의하십시오.
- 확약 빈도를 증가시키십시오. 이렇게 하면 주어진 시간에 기존의 잠금 수를 줄이게 됩니다. 분리 레벨 및 동시성에 대한 자세한 내용은 45 페이지의 『동시성』을 참조하십시오.

잠금 대기 및 시간종료

비정상적인 상황에서 잠금 시간종료를 검출하지 않으면 응용프로그램은 잠금이 해제되기를 무한정 기다려야 합니다. 예를 들어, 트랜잭션이 다른 사용자의 응용프로그램에 의해 보유된 잠금을 기다리고 다른 사용자는 응용프로그램이 잠금을 해제할 트랜잭션이 확약하는 응용프로그램과의 상호작용을 수행하지 않고 워크스테이션을 떠나는 경우에 이러한 상황이 발생할 수 있습니다. 이러한 상황이 발생하면, 분명 응용프로그램 성능이 더 나빠집니다. 이 경우에 프로그램을 *정지*시키지 않

으려면, *locktimeout* 구성 매개변수를 사용하여 응용프로그램이 잠금을 확보하기 위해 기다리는 최대 시간을 설정할 수 있습니다. (자세한 내용은 445 페이지의 『잠금 시간종료(locktimeout)』를 참조하십시오.)

이 매개변수를 사용하면 분산 작업 단위(DUOW) 응용프로그램에서 특히 전역 교착 상태를 피할 수 있도록 도와줍니다. 잠금이 시간종료하는 경우, 즉 잠금 요청이 보류 중인 시간이 *locktimeout* 값보다 큰 경우, 응용프로그램은 오류를 수신하고 트랜잭션은 구간 복원됩니다. 예를 들어, *program1*이 *program2*에 의해 이미 보유된 잠금을 획득하려는 경우, *program1*은 시간종료가 만기되면 이유 코드 68을 가진 SQLCODE -911(SQLSTATE 40001)을 리턴합니다.

데이터베이스 관리 프로그램 구성 매개변수 *diaglevel*이 4로 설정되고 잠금 요청이 시간종료한 경우, db2diag.log에 더 많은 정보가 있을 수 있습니다. 발견되는 정보에는 오브젝트, 잠금 모드 그리고 오브젝트에서 잠금을 보유하는 응용프로그램이 포함됩니다. 현재 동적 SQL문이나 정적 패키지 이름이 있을 수도 있습니다.

교착 상태

데이터베이스 관리 프로그램에서 데이터베이스를 사용하는 프로세스에 의한 잠금 경합으로 인해 교착 상태가 발생합니다. 예를 들어, 프로세스 1은 X(독점) 모드의 테이블 A를 잠그고, 프로세스 2는 X 모드의 테이블 B를 잠급니다. 그런데 프로세스 1이 X 모드의 테이블 B를 잠그고 프로세스 2가 X 모드의 테이블 A를 잠그려고 하는 경우, 프로세스는 교착 상태가 됩니다. 교착 상태에서 두 프로세스는 두 번째 잠금 요청이 권한 부여될 때까지 일시중단되고 한 프로세스가 확약 또는 구간 복원을 수행할 때까지 어떤 요청도 권한 부여되지 않습니다. 이 상태는 외부 에이전트가 프로세스 중 하나를 활성화하고 프로세스가 구간 복원을 수행하도록 강제(force)할 때까지 무기한으로 유지됩니다.

잠금 시스템에서 교착 상태는 교착 상태 검출기라고 하는 비동기 시스템 백그라운드 프로세스에 의해 데이터베이스 관리 프로그램에서 처리됩니다. 교착 상태 검출기는 *dlchktime* 구성 매개변수에서 결정한 대로 정기적으로 사용할 수 있게 됩니다(442 페이지의 『교착 상태 점검을 위한 시간 간격(dlchktime)』 참조). 교착 상태 검출기가 사용 중인 경우, 교착 상태에 대한 잠금 시스템을 검사합니다. 데이터베이스가 파티션된 경우, 각 파티션은 *잠금 그래프*를 전역 교착 상태가 검출된 시스템 카탈로그 뷰를 가지고 있는 데이터베이스 파티션에 전송합니다.

교착 상태가 있으면 교착 상태 검출기는 구간 복원하기 위해 교착 상태가 된 프로세스를 선택합니다. 선택된 프로세스는 작동 중이며, 이유 코드가 2인 SQLCODE -911(SQLSTATE 40001)과 함께 호출 응용프로그램으로 리턴됩니다. 데이터베이스 관리 프로그램은 선택된 프로세스를 자동으로 구간 복원합니다. 구간 복원이 완료되면 희생(victim) 프로세스에 속한 잠금은 해제되고 교착 상태에 관련된 다른 프로세스는 결국 계속할 수 있습니다.

교착 상태 검출기에 대한 적합한 간격 선택은 성능이 좋은지를 확인하는 데 필요합니다. 간격이 너무 짧으면 불필요한 오버헤드를 일으키고, 간격이 너무 길면 교착 상태는 승인할 수 없는 시간량에 대한 프로세스를 연기합니다. 예를 들어, 가동(wake up) 간격이 30분으로 설정되면 교착 상태는 거의 30분이 됩니다. 응용프로그램 설계자는 교착 상태를 해결할 때, 가능한 지연을 교착 상태를 검출하는 오버헤드와 균형을 유지해야 합니다.

파티션된 데이터베이스의 경우, 모든 파티션에서 간격이 동일해야 합니다. (*dlchktime* 구성 매개변수는 모든 파티션에서 동일한 값으로 갱신되어야 합니다.) 다른 파티션에서보다 카탈로그 노드 값이 작으면 팬텀 교착 상태가 검출됩니다. 다른 파티션에서보다 카탈로그 노드의 값이 더 큰 경우, 교착 상태가 검출되기 전에 세 개 이상의 간격이 지나간 것처럼 나타납니다. 많은 교착 상태가 파티션된 데이터베이스에서 검출되는 경우, 잠금 대기 및 통신 대기용 계정에 대한 *dlchktime* 매개변수의 값을 증가시켜야 합니다.

데이터베이스에 액세스하는 둘 이상의 독립된 프로세스를 가진 응용프로그램이 교착 상태와 같은 방식으로 구조화되는 경우, 또 다른 문제가 발생할 수 있습니다. 여러 프로세스가 읽기 및 쓰기에 대해 같은 테이블에 액세스하는 예가 있습니다. 프로세스가 처음에는 읽기 전용 SQL 조회를 수행하고 같은 테이블에서 SQL 갱신을 수행하는 경우, 같은 데이터에 대한 프로세스간의 잠재적인 경합 때문에 교착 상태가 발생할 기회는 증가합니다. 예를 들어, 두 프로세스가 테이블을 먼저 읽고 테이블을 갱신하는 경우, 프로세스 A는 행에서 X 잠금을 확보하려 하며, 여기서 프로세스 B는 반대로 S 잠금을 가지는 상태로 들어갑니다. 결과적으로, 교착 상태가 될 수 있습니다. 이런 교착 상태를 피하려면 수정하려고 데이터에 액세스하는 응용프로그램은 선택을 수행할 때 FOR UPDATE OF절을 사용해야 합니다.이 절을 통해 프로세스 A가 데이터를 읽으려고 할 경우, U 잠금이 부여된 것을 확인합니다.

주: 교착 상태가 발생하는 시기를 기록할 모니터를 정의하는 것을 고려하고자 할 수 있습니다. *SQL 참조서*에 설명된 CREATE EVENT문을 사용하여 모니터를 작성하십시오.

연합 시스템 환경에서 응용프로그램이 별칭에 액세스할 경우, 응용프로그램이 요청한 데이터는 데이터 소스에서의 교착 상태로 인해 사용하지 못할 수 있습니다. 이러한 상황이 발생하면 DB2는 데이터 소스의 교착 상태 처리 기능을 통해 잠금을 해결합니다. 둘 이상의 데이터 소스에 걸쳐 교착 상태가 발생할 경우, DB2는 데이터 소스 시간종료 메커니즘을 통해 교착 상태를 중단시킵니다.

데이터베이스 관리 프로그램 구성 매개변수 *diaglevel*이 4로 설정되고 잠금 요청이 교착 상태로 인해 실패할 경우, db2diag.log에 더 많은 정보가 있을 수 있습니다. 발견되는 정보에는 오브젝트, 잠금 모드 그리고 오브젝트에서 잠금을 보유하는 응용프로그램이 포함됩니다. 현재 동적 SQL문이나 정적 패키지 이름이 있을 수도 있습니다.

잠금에 영향을 주는 인수

데이터베이스 관리 프로그램 잠금의 모드 및 세분성은 인수(응용프로그램이 수행하는 처리 유형, 응용프로그램이 데이터에 액세스하는 방법, 사용자가 지정할 수 있는 여러 매개변수)의 조합에 의해 결정됩니다.

응용프로그램 처리

잠금 속성을 결정하기 위해 처리는 네 가지 유형 중 하나로 분류될 수 있습니다.

읽기 전용

이 유형은 원래 읽기 전용이고(커서에 대한 자세한 내용은 *SQL 참조서* 참조), 명시적인 FOR READ ONLY절이 있거나, 불분명하지만 PREP 또는 BIND 명령에 지정된 BLOCKING 옵션 값 때문에 SQL 컴파일러가 읽기 전용일 것이라고 생각되는 모든 Select 문을 포함합니다. 여기에는 공유 잠금(S 또는 IS)만이 필요합니다.

변경할 의도

이 유형은 FOR UPDATE절을 가진, 또는 SQL 컴파일러가 미결정문을 번역함으로써 변경하려고 하는 모든 Select 문을 포함합니다. 이 유형에서

는 공유 잠금과 갱신 잠금을 사용합니다. (행의 경우 S, U, X를 사용하고, 테이블의 경우 IX, U, X를 사용합니다.)

변경 이 유형은 UPDATE, INSERT, DELETE는 포함하지만, UPDATE WHERE CURRENT OF 또는 DELETE WHERE CURRENT OF는 포함하지 않습니다. 이 유형은 독점 잠금이 필요합니다(X 또는 IX).

제어된 커서

이 유형은 UPDATE WHERE CURRENT OF와 DELETE WHERE CURRENT OF를 포함합니다. 이 유형은 또한 독점 잠금이 필요합니다 (X 또는 IX).

목표 테이블에 대해 삽입, 갱신 또는 삭제할 명령문에는 부속 Select 문의 결과에 근거하여 두 가지 처리 유형이 있습니다. 부속 선택에서 리턴된 테이블에 대한 잠금은 읽기 전용 처리 규칙에 의해 결정됩니다. 목표 테이블에 대한 잠금은 변경 처리 규칙에 의해 결정됩니다.

액세스 경로

*액세스 경로*는 특정 테이블 참조로부터 데이터를 검색하는 최적화 알고리즘에 의해 선택된 메소드입니다. (자세한 내용은 183 페이지의 『데이터 액세스 개념 및 최적화』를 참조하십시오.) 최적화 알고리즘에 의해 선택된 액세스 경로는 잠금 모드에 심각한 영향을 미칩니다. 예를 들어, 색인 스캔을 사용하여 특정 행을 위치 지정하는 경우, 최적화 알고리즘은 테이블에 대해 행 레벨 잠금(IS)을 선택합니다. 이러한 액세스 유형은 사원 번호(EMPNO)에 대한 색인이 있는 EMPLOYEE 테이블로부터 다음 명령문을 사용하여 한 명의 사원에 대한 정보를 선택하는 데 사용됩니다.

```
SELECT *
   FROM   EMPLOYEE
  WHERE EMPNO = '000310';
```

마찬가지로, 사용된 색인이 없는 경우, 전체 테이블은 선택된 행을 찾기 위해 순서에서 스캔되어야 하고 하나의 테이블 레벨 잠금(S)을 얻을 수 있습니다. 예를 들어, 이 유형의 액세스는 컬럼 SEX에 색인이 없는 다음 경우처럼 명령문을 사용하여 모든 남자 사원을 선택하는 데 사용될 수 있습니다.

```
SELECT *
   FROM   EMPLOYEE
  WHERE SEX = 'M';
```

다음 표에서는 액세스 플랜 종류에 따라 확보된 잠금 개요를 제공합니다. 컬럼 표제 정의에 대한 자세한 내용은 66 페이지의 『응용프로그램 처리』를 참조하십시오. 액세스 메소드 정의에 대한 자세한 내용은 183 페이지의 『데이터 액세스 개념 및 최적화』를 참조하십시오. *제어된 커서* 유형의 처리 방식은 응용프로그램이 갱신하거나 삭제할 행을 찾을 때까지 기존 커서의 잠금 모드를 사용함을 유의하십시오. 이러한 처리 유형의 경우, 커서의 잠금 모드에 관계없이 독점 잠금은 갱신 또는 삭제를 수행하기 위해 항상 확보됩니다.

다음 표에서 하나의 잠금 모드만 나타나는 경우, 테이블 레벨 잠금 모드입니다. 두 잠금 모드 중 첫 번째 모드는 테이블 레벨 잠금 모드이고 두 번째 모드는 행 레벨 잠금 모드입니다.

표 5. 테이블 스캔용 잠금 모드

분리 레벨	읽기 전용	변경할 의도	변경
액세스 메소드: *술어가 없는 테이블 스캔*			
RR	S	U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
액세스 메소드: *술어가 있는 테이블 스캔*			
RR	S	U	U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

표 6. 색인 스캔용 잠금 모드

분리 레벨	읽기 전용	변경할 의도	변경
액세스 메소드: *술어가 없는 색인 스캔*			
RR	S	IX / U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X

표 6. 색인 스캔용 잠금 모드 (계속)

분리 레벨	읽기 전용	변경할 의도	변경
액세스 메소드: *단일 규정 행을 사용한 색인 스캔*			
RR	IS / S	IX / U	IX / X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
액세스 메소드: *시작 및 중지 술어만을 사용한 색인 스캔*			
RR	IS / S	IX / S	IX / X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
액세스 메소드: *술어가 있는 색인 스캔*			
RR	IS / S	IX / S	IX / U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

70 페이지의 표7에는 행 목록을 다음과 같이 하기 위해 데이터 페이지 읽기가 지연된 경우에 대한 잠금 모드가 나와 있습니다.

- 다중 색인을 사용하여 규정을 더욱 확실히 합니다. 자세한 내용은 190 페이지의 『다중 색인 액세스』를 참조하십시오.
- 효율적인 프리페치로 정렬됩니다. 자세한 내용은 294 페이지의 『목록 프리페치의 이해』를 참조하십시오.

데이터 페이지의 지연된 액세스는, 행에 대한 액세스가 두 단계에서 발생하여 잠금 시나리오가 훨씬 복잡하게 되는 것을 암시합니다. 분리 레벨에 따라 결정되는 두 가지 주요 범주가 있습니다. 반복 읽기(RR) 분리 레벨은 트랜잭션 종료까지 획득한 잠금을 유지하기 때문에 첫 번째 단계에서 획득한 잠금이 보유되고 두 번째 단계에서 잠금을 더 획득할 필요가 없습니다. 읽기 안정성(RS) 및 커서 안정성(CS) 분리 레벨의 경우, 잠금은 두 번째 단계 중에 획득해야 합니다. 동시성을 최대화하기 위해 잠금은 첫 번째 단계에서 획득되지 않고 모든 술어의 재적용에 따라 규정 행만 리턴되도록 합니다.

표 7. 지연된 데이터 페이지 액세스에 사용된 색인 스캔용 잠금 모드

분리 레벨	읽기 전용	변경할 의도	변경
액세스 메소드: *술어가 없는 색인 스캔*			
RR	IS / S	IX / S	X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
액세스 메소드: *술어가 없는 색인 스캔 이후의 지연된 데이터 페이지 액세스*			
RR	IN	IX / S	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
액세스 메소드: *술어가 있는 색인 스캔*			
RR	IS / S	IX / S	IX / S
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
액세스 메소드: *시작 및 중지 술어만을 사용한 색인 스캔*			
RR	IS / S	IX / S	IX / X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
액세스 메소드: *술어가 있는 색인 스캔 이후의 지연된 데이터 페이지 액세스*			
RR	IN	IX / S	IX / S
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

사용자가 액세스 경로를 제어하지 않고, 최적화 알고리즘에 의해 액세스 경로가 선택됩니다.

사용된 액세스 경로는 모드와 잠금의 세분성에 영향을 줄 수 있습니다. 예를 들어, 반복 읽기(RR) 분리 레벨을 사용한 응용프로그램에서 술어가 없는 테이블 스

캔을 사용한 UPDATE 조회는 테이블에서 X 잠금을 사용합니다. 색인을 통해 행을 찾은 경우, 데이터베이스 관리 프로그램은 테이블의 개별 행을 잠그도록 선택할 수 있습니다.

선언된 임시 테이블 및 잠금

선언된 임시 테이블은 선언된 응용프로그램만 사용할 수 있으므로 잠기지 않습니다. 이러한 유형의 테이블은 응용프로그램이 이를 선언하는 시점으로부터 응용프로그램이 완료되거나 연결 해제될 때까지만 존재합니다.

LOCK TABLE문

응용프로그램에서 LOCK TABLE문을 사용하여 초기 잠금 모드를 획득하는 규칙을 대체할 수 있습니다.

명령문은 전체 테이블을 잠급니다. LOCK TABLE문에 지정된 테이블만 잠깁니다. 지정된 테이블의 상위 테이블과 종속 테이블은 잠기지 않습니다. 액세스될 수 있는 다른 테이블을 잠그는 것이 동시성과 성능에 의하여 기대하는 결과를 얻는 데 필요한지를 결정해야 합니다. 작업 단위(UOW)가 확약되거나 구간 복원될 때까지 잠금은 해제되지 않습니다.

여러 명의 사용자가 테이블을 공유하는 경우, 다음과 같은 이유로 테이블을 잠그려 할 수 있습니다.

LOCK TABLE IN SHARE MODE

지정된 시간 내에 일관성이 있는 데이터에 액세스하려 합니다. 즉, 특정 시점의 테이블에 대한 현재 데이터를 말하는 것입니다. 테이블에서 활동이 빈번히 일어나는 경우, 전체 테이블이 안정성이 있는지 확인하는 유일한 방법은 테이블을 잠그는 것 뿐입니다. 예를 들어, 응용프로그램은 테이블의 스냅샷을 하려고 합니다. 그러나 응용프로그램이 테이블의 일부 행을 처리해야 하는 동안, 다른 응용프로그램은 사용자가 아직 처리하지 않은 행을 갱신하고 있습니다. 이 갱신은 반복 읽기(RR)를 사용하여 허용할 수 있지만, 사용자가 원하는 조치는 아닙니다.

다른 방법으로, 응용프로그램에서 LOCK TABLE IN SHARE MODE 문을 발행할 수 있습니다. 검색 여부에 관계없이 행을 변경할 수 없습니다. 그러면 필요한 만큼의 행을 검색하여 검색한 행이 검색 전과 같이 변경되지 않은 것을 알 수 있습니다.

LOCK TABLE IN SHARE MODE를 사용하여 다른 사용자는 테이블에서 데이터를 검색할 수 있습니다. 그러나 행을 갱신, 삭제하거나 테이블로 행을 삽입할 수 없습니다.

LOCK TABLE IN EXCLUSIVE MODE

대형 테이블 부분을 갱신하려고 합니다. 다른 모든 사용자가 행이 갱신될 때마다 행을 잠그는 것보다 테이블에 액세스하지 않고, 모든 변경사항이 확약된 후에 행의 잠금을 해제하는 것이 비용도 덜 들고 더 효율적입니다.

독점 모드에서 LOCK TABLE을 사용하여 다른 모든 사용자가 잠겨집니다. 미확약 읽기(UR) 응용프로그램이 아니면 어떠한 응용프로그램도 테이블에 액세스할 수 없습니다.

LOCK TABLE문에 대한 자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오.

LOCK TABLE문을 사용하지 않는 다른 방법은 ALTER TABLE문을 LOCKSIZE 매개변수와 함께 사용하는 방법입니다. LOCKSIZE 매개변수를 사용하면 ROW 잠금 또는 TABLE 잠금을 선택할 수 있습니다. 모든 선택사항은 다음 테이블에 액세스될 때 선택된 잠금의 세분성이 됩니다. 테이블 작성시 ROW 잠금을 선택하는 것은 기본 잠금 *크기*를 선택하는 것과 다르지 않습니다. TABLE 잠금을 선택하면 획득해야 할 잠금의 수를 제한하여 조회의 성능을 향상시킬 수 있습니다. 그러나 전체 테이블에서 모든 잠금이 보유되기 때문에 동시성은 감소됩니다. 이 선택사항을 선택해도 정상적으로 발생하는 잠금 레벨 자동 업그레이드를 막지 못합니다. ALTER TABLE문에 자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오.

릴리스로 커서 닫기

WITH RELEASE절을 포함하는 CLOSE CURSOR문을 사용하여 커서를 닫으면 데이터베이스 관리 프로그램은 커서용으로 보유되어 있는 모든 읽기 잠금(있을 경

우)을 해제하려고 합니다. 읽기 잠금은 IS, S이고, U는 S, NS와 함께 테이블 잠금이며, U는 행 잠금입니다. 잠금 모드에 대한 자세한 내용은 56 페이지의 『잠금 속성』을 참조하십시오.

WITH RELEASE절은 CS 또는 UR 분리 레벨 아래에서 조작 중인 커서에 영향을 주지 않습니다. RS 또는 RR 분리 레벨 아래에서 조작 중인 커서용으로 지정하는 경우, WITH RELEASE절은 해당 분리 레벨의 여러 보증을 종료합니다. 특히, RS 커서에서 *반복 불가능 읽기* 현상이 발생하고, RR 커서에서도 *반복 불가능 읽기* 현상이나 *팬텀 읽기* 현상이 발생합니다.

원래 RR 또는 RS인 커서는 WITH RELEASE절을 사용하여 닫혀진 후에 다시 열리고 새 읽기 잠금이 획득됩니다.

CLOSE CURSOR문에 대해 다른 기본 절과의 비교에 대한 자세한 내용은 87 페이지의 『DECLARE CURSOR WITH HOLD문』을 참조하십시오.

DB2 CLI 연결 속성인 SQL_ATTR_CLOSE_BEHAVIOR를 CLI 응용프로그램에 사용하여 CLOSE CURSOR WITH RELEASE와 같은 결과를 얻을 수 있습니다. 자세한 내용은 *CLI Guide and Reference*의 `SQLSetConnectAttr()`절을 참조하십시오.

잠금 고려사항 요약

다음은 잠금에 대해 기억해야 할 사항입니다.

- 작은 작업 단위(UOW)(빈번한 COMMIT문)는 많은 사용자에 의한 동시 데이터 액세스를 승격합니다. 응용프로그램이 논리적으로 일관성 지점에 있을 때, 즉 변경한 데이터가 일관성이 있는 경우, COMMIT문을 포함시키십시오. COMMIT를 발행하는 경우, 잠금이 해제됩니다(WITH HOLD로 선언된 커서와 연관된 테이블 잠금의 경우 제외).
- 응용프로그램이 행만을 읽더라도 잠금은 획득하므로 읽기 전용 작업 단위(UOW)를 확약하는 것은 여전히 중요합니다. 공유 잠금은 읽기 전용 응용프로그램에서 반복 읽기(RR) 분리 레벨, 읽기 안정성(RS) 분리 레벨, 커서 안정성(CS) 분리 레벨에 의해 획득되기 때문입니다. 반복 읽기(RR) 및 읽기 안정성(CS)을 통해 모든 잠금은 COMMIT가 발행될 때까지 보유되며 WITH RELEASE절을 사

용하여 사용자의 커서를 닫지 않으면 다른 프로세스는 잠긴 데이터를 갱신할 수 없습니다. 또한 카탈로그 잠금은 동적 SQL을 사용하여 미확약 읽기(UR) 응용프로그램에서도 획득할 수 있습니다.

- 데이터베이스 관리 프로그램은 미확약 읽기(UR) 분리 레벨을 사용하지 않으면 응용프로그램이 미확약된 데이터를 검색하지 않음을 확인합니다. (다른 응용프로그램에 의해 갱신되었으나, 아직 확약되지 않은 행이 여기에 해당됩니다.)
- LOCK TABLE문을 발행하여 보호하려는 전체 테이블을 잠글 수 있습니다.
  - 다른 응용프로그램은 행을 검색하지만, 행을 갱신, 삭제 또는 삽입할 수는 없습니다.
  - 다른 응용프로그램이(미확약 읽기(UR) 분리 레벨을 사용하는 응용프로그램 제외) 테이블의 행에 액세스할 수 없습니다.
- WITH RELEASE절을 포함하는 CLOSE CURSOR문을 사용하여 커서를 닫으면 데이터베이스 관리 프로그램은 커서용으로 보유되어 있는 모든 읽기 잠금(있을 경우)을 해제하려고 합니다.
- 파티션된 데이터베이스에서 잠금에 영향을 주는 구성 매개변수를 변경 중이면 데이터베이스의 모든 파티션에서 변경이 모두 수행되는지 확인하십시오.

최적화 클래스 조정

SQL 조회가 컴파일되는 경우, 많은 최적화 기술을 사용하여 해당 조회에 대한 가장 효율적인 액세스 플랜을 결정할 수 있습니다. 좀더 나은 최적화 기술을 사용하면 다음 결과를 가져옵니다.

1. 런타임 성능 향상
2. 조회 컴파일 시간 증가
3. 시스템 자원 사용 증가

이러한 이유로, 최적화 클래스를 설정하여 조회를 최적화하는 데 적용되는 기술의 수를 제한할 수 있습니다. 특히 다음 경우에 사용할 수 있습니다.

- 매우 작은 데이터베이스 또는 매우 간단한 동적 조회
- 데이터베이스 서버에서 컴파일시 사용 가능한 제한된 메모리
- 조회 컴파일(예: PREPARE) 시간을 줄이려는 경우

다음에 설명된 조회 최적화 클래스를 선택할 수 있습니다. 그러나 클래스 0과 클래스 9는 특별한 경우에만 사용해야 합니다. 클래스 5가 기본값입니다. 클래스 0, 1, 2는 Greedy 조인 열거 알고리즘을 사용합니다. 복합 조회의 경우, 이 알고리즘은 대체 플랜을 거의 고려할 수 없고 클래스 3 이상의 클래스보다 컴파일 시간이 훨씬 적게 걸립니다. 클래스 3 이상의 클래스는 동적 프로그래밍 조인 열거 알고리즘을 사용합니다. 이 알고리즘은 대체 플랜을 고려하고 증가하는 테이블 수처럼 클래스 0, 1, 2보다 컴파일 시간이 훨씬 오래 걸립니다.

0 - 이 클래스는 최소의 최적화를 사용하여 액세스 플랜을 생성하도록 최적화 알고리즘에 지시합니다. 예를 들면, 다음과 같습니다.

- 최적화 알고리즘에는 non-uniform 분산 통계를 고려하지 않습니다.
- 기본 조회 재작성 규칙만이 적용됩니다. (조회 재작성에 대한 자세한 내용은 171 페이지의 『SQL 컴파일러에 의한 조회 재작성』을 참조하십시오.)
- Greedy 조인 열거가 발생합니다. (자세한 내용은 204 페이지의 『최적의 조인을 선택하기 위한 검색 전략』을 참조하십시오.)
- 중첩된 루프 조인과 색인 스캔 액세스 메소드만이 작동 가능합니다. (자세한 내용은 197 페이지의 『조인 개념』 및 184 페이지의 『색인 스캔 개념』을 참조하십시오.)
- 목록 프리페치와 색인 AND 작업은 생성된 액세스 메소드에서 사용되지 않도록 작동 불가능하게 됩니다.
- 스타 조인 전략은 고려되지 않습니다.

이 클래스는 가장 낮은 가능 조회 컴파일 오버헤드를 필요로 하는 특별한 경우에만 사용될 수 있습니다. 색인화가 잘 되어 있는 테이블에 액세스하는 매우 간단한 동적 SQL문을 전체적으로 구성하는 응용프로그램이 조회 최적화 클래스 0이 적합한 좋은 예입니다.

1 - 이 클래스는 버전 1에는 없는 일부 낮은 비용의 기능을 추가하여 DB2/6000 버전 1과 어느정도 비교할 수 있는 최적화 등급을 사용하도록 최적화 알고리즘에 지시합니다.

- 최적화 알고리즘은 non-uniform 분산 통계를 고려하지 않습니다.
- DB2/6000 버전 1에서 제공된 규칙을 포함한 조회 재작성 규칙의 부속 집합만이 적용됩니다.

- Greedy 조인 열거가 발생합니다. (자세한 내용은 204 페이지의 『최적의 조인을 선택하기 위한 검색 전략』을 참조하십시오.)
- 목록 프리페치와 색인 AND 작업은 생성된 액세스 메소드에서 사용되지 않도록 작동 불가능하게 됩니다.

  주: 색인 AND 작업은 스타 조인에서 발견되는 세미조인에 대해 작업할 때도 계속 사용됩니다.

최적화 클래스 1은 병합 스캔 조인과 테이블 스캔이 또한 사용 가능한 것을 제외하면 클래스 0과 유사합니다.

2 - 이 클래스는 복합 조회에 대해 클래스 3 이상의 클래스보다 더 낮은 컴파일 비용을 확실히 유지하는 반면에, 클래스 1의 비용을 확실히 향상시키는 최적화를 사용하도록 최적화 알고리즘에 지시합니다. 특히,

- 빈도와 quantile non-uniform 분산 통계를 포함한 사용 가능한 통계가 모두 사용됩니다.
- 요약 테이블로의 조회 경로지정을 포함한 모든 조회 재작성 규칙이 적용되는데, 매우 드문 경우에만 적용 가능한 계산 집약적인 규칙은 제외됩니다.
- Greedy 조인 열거가 사용됩니다.
- 목록 프리페치 및 요약 테이블 라우팅을 포함한 다양한 범위의 액세스 메소드가 고려됩니다.
- 적용 가능한 경우, 스타 조인 전략은 고려되지 않습니다.

최적화 클래스 2는 동적 프로그래밍보다는 Greedy 조인 열거를 사용하는 점을 제외하면 클래스 5와 매우 유사합니다. 이 클래스는 Greedy 조인 열거 알고리즘을 사용하는 모든 최적화 클래스 중 최고의 최적화를 가지고 있으며, 이 최적화는 복합 조회에 대한 대체 방법을 거의 고려하지 않아 클래스 3 이상의 클래스보다 컴파일 시간이 적게 걸립니다. 그러므로 결정 지원 또는 온라인 분석 처리(OLAP) 환경에서 매우 복합 조회용으로 사용하는 것이 좋습니다. 이 경우, 같은 조회가 거의 실행하지 않아서 다음 조회가 발생할 때까지 조회의 액세스 플랜은 캐쉬에 남아 있지 않습니다.

3 - 이 클래스는 액세스 플랜을 생성하기 위해 수행될 적당한 최적화의 양을

요청합니다. 이 클래스는 MVS/ESA용 DB2 또는 OS/390의 조회 최적화 특성에 가장 가까이 일치합니다. 이 최적화 클래스에는 다음과 같은 특성이 있습니다.

- non-uniform 분산 통계는 사용 가능한 경우, 발생하는 값을 추적하는 데 사용됩니다.
- 조인에 대한 부속 조회 변환을 포함한 대부분의 조회 재작성 규칙이 적용됩니다.
- 동적 프로그래밍 조인 열거(204 페이지의 『최적의 조인을 선택하기 위한 검색 전략』 참조)
  - 제한된 복합 내부 테이블 사용(206 페이지의 『복합 테이블』 참조)
  - 『찾아보기』 테이블을 포함한 스타 스키마용으로 제한된 Cartesian 제품 사용(204 페이지의 『스타 조인의 검색 전략』 참조)
- 목록 프리페치, 색인 ADDing 및 스타 조인을 포함한 광범위한 액세스 메소드가 고려됩니다.

이 클래스는 광범위한 응용프로그램에 적합합니다. 이 클래스를 사용하면 최적화 알고리즘은 네 개 이상의 조인을 가진 조회에 대한 탁월한 액세스 플랜을 선택할 수 있는 좋은 기회가 생깁니다. 그러나 최적화 알고리즘은 기본 조회 최적화 클래스로 선택된 더 나은 플랜을 고려하는 데 실패할 수 있습니다.

5 - 이 클래스는 상당한 양의 최적화를 사용하여 액세스 플랜을 생성하도록 최적화 알고리즘에 지시합니다. 예를 들어, 클래스 5에는 다음과 같은 특성이 있습니다.

- 빈도와 quantile non-uniform 분산 통계를 포함한 사용 가능한 모든 통계
- 요약 테이블로의 조회 경로지정을 포함한 모든 조회 재작성 규칙이 적용되는데, 매우 드문 경우에만 적용 가능한 계산 집약적인 규칙은 제외됩니다.
- 동적 프로그래밍 조인 열거(204 페이지의 『최적의 조인을 선택하기 위한 검색 전략』 참조)
  - 제한된 복합 내부 테이블 사용(206 페이지의 『복합 테이블』 참조)

– 『찾아보기』 테이블을 포함한 스타 스키마용으로 제한된 Cartesian 제품 사용(204 페이지의 『스타 조인의 검색 전략』 참조)

- 목록 프리페치, 색인 ADDing 및 요약 테이블 경로지정을 포함한 광범위한 액세스 메소드가 고려됩니다.

최적화 알고리즘이 복잡한 동적 SQL 조회에 대해 보장되지 않은 추가 자원과 처리 시간을 검출하는 경우, 최적화는 줄어듭니다. 감소되는 extent 또는 크기는 머신 크기와 술어의 수에 따라 결정됩니다.

조회 최적화 알고리즘이 수행된 조회 최적화 양을 줄이는 경우, 보통 적용되는 조회 재작성 규칙을 모두 계속 적용합니다. 그러나 Greedy 조인 열거 메소드를 사용하지 않고, 고려되는 액세스 플랜 조합 수를 줄입니다.

조회 최적화 클래스 5는 트랜잭션과 복합 조회를 모두 구성하는 혼합된 환경에 적합한 탁월한 선택사항입니다. 이 최적화 클래스는 대부분의 다양한 조회 변환과 다른 조회 최적화 기술을 효과적으로 적용하도록 설계되었습니다.

7 - 이 클래스는 상당한 양의 최적화를 사용하여 액세스 플랜을 생성하도록 최적화 알고리즘에 지시합니다. 복잡한 동적 SQL 조회에 대한 조회 최적화 양을 줄이지 않는 점을 제외하면 조회 최적화 클래스 5와 매우 유사합니다.

9 - 이 클래스는 사용 가능한 모든 최적화 기술을 사용하도록 최적화 알고리즘에 지시합니다. 여기에는 다음이 포함됩니다.

- 사용 가능한 모든 통계
- 모든 조회 재작성 규칙
- Cartesian 제품과 제한되지 않는 복합 내부를 포함한 모든 가능한 조인 열거
- 모든 액세스 메소드

이 클래스는 최적화 알고리즘에 의해 고려되는 가능한 액세스 플랜의 수를 크게 확장할 수 있습니다. 이 클래스를 사용하여 대형 테이블을 통하여 매우 복잡하고 장시간에 걸친 조회에 대한 더 나은 액세스 플랜을 생성할 수 있는 범위가 더 넓은 최적화를 결정해야 합니다. Explain 및 성능 측정을 사용하여 더 나은 플랜이 있는 것을 검증해야 합니다.

최적화 클래스 설정 방법

특정 조회 최적화 클래스를 요청하는 방법은 정적 SQL 또는 동적 SQL의 사용 여부에 따라 결정됩니다.

- *정적 SQL*문은 PREP 및 BIND 명령에 지정된 최적화 클래스를 사용합니다. SYSCAT.PACKAGES 카탈로그 테이블의 QUERYOPT 컬럼은 패키지를 바인드하는 데 사용된 최적화 클래스를 기록합니다. 패키지가 내재적으로 또는 REBIND PACKAGE 명령을 사용하여 리바인드되는 경우, 같은 최적화 클래스는 정적 SQL문에 사용됩니다. 이러한 정적 SQL문에 사용된 최적화 클래스를 변경하려면 BIND 명령을 사용해야 합니다. 최적화 클래스를 지정하지 않은 경우, DB2는 *dft_queryopt* 데이터베이스 구성 매개변수가 지정한 기본 최적화를 사용합니다.
- *동적 SQL*문은 SQL SET문을 사용하여 설정된 CURRENT QUERY OPTIMIZATION 특수 레지스터에 의해 지정된 최적화 클래스를 사용합니다. 예를 들어, 다음 명령문은 최적화 클래스를 1로 설정합니다.

  ```
  SET CURRENT QUERY OPTIMIZATION = 1
  ```

  동적 SQL문이 같은 최적화 클래스를 항상 사용하는 것을 확인하기 위해 응용프로그램에 이 SET문을 포함시키려고 합니다. 자세한 내용은 *SQL 참조서*를 참조하십시오.

  CURRENT QUERY OPTIMIZATION 레지스터가 설정되지 않은 경우, 기본 조회 최적화 클래스를 사용하여 동적문이 바인드됩니다. 동적 SQL과 정적 SQL에 대한 기본값은 구성 가능한 데이터베이스 매개변수 *dft_queryopt* 값에 따라 결정됩니다. 기본값을 변경하지 않으면 클래스 5가 기본 조회 최적화 클래스입니다. (이 매개변수에 대한 자세한 내용은 515 페이지의 『기본 조회 최적화 클래스(dft_queryopt)』를 참조하십시오.) 바인드 옵션과 특수 레지스터에 대한 기본값은 *dft_queryopt* 데이터베이스 구성 매개변수로부터 확보됩니다.

최적화 필요 정도

기본 조회 최적화 클래스와 함께 적당한 자원량을 사용하여 대부분의 명령문은 적합하게 최적화됩니다. 주어진 최적화 클래스에서 조회 컴파일 시간과 자원 소비는 조회의 복잡도, 특히 조인 및 부속 조회 수에 의해 주로 영향을 받습니다. 그러나 컴파일 시간과 자원의 사용은 다양한 최적화 클래스에서 수행된 최적화의 양에 의

해서도 영향을 받습니다. 다른 최적화 클래스의 경우, 간단한 조회에서보다 매우 복합 조회에 대한 조회 컴파일 시간과 자원 소비에서 더 큰 차이가 나타날 수 있습니다.

다음 내용은 사용할 최적화 클래스를 선택하는 데 도움을 줍니다.

- 기본 조회 최적화 클래스를 사용한 시작
- 기본 클래스를 제외한 클래스를 사용하려면 먼저 클래 1, 2 또는 3을 시도하십시오.
- 매우 짧은 런타임 조회 즉, 조회에 걸리는 시간이 1초 미만인 조회에 대해 낮은 최적화 클래스(0 또는 1)를 사용하십시오. (낮은 최적화 클래스를 선택할 때의 추가 기준에 대한 다음 논의를 참조하십시오.)
- 같은 컬럼에서 많은 Join 술어를 가진 테이블이 있는 경우와 컴파일 시간이 중요한 경우, 최적화 클래스 1 또는 2를 사용하십시오.
- 오랜 시간 수행하는 조회 즉, 조회에 걸리는 시간이 30초 이상인 조회에 대해 더 높은 최적화 클래스(3, 5 또는 7)를 사용하십시오.
- 보통의 경우에 최적화 클래스 9를 사용하면 안됩니다.
- 오랜 시간 수행하는 조회의 경우, db2batch를 사용하여 컴파일하는 데 걸린 시간과 실행에 걸린 시간을 결정하여 조회를 수행하십시오.
  - 대부분의 시간이 컴파일하는 데 걸리며 최적화 클래스를 줄입니다.
  - 대부분의 시간이 실행하는 데 걸리며 더 높은 최적화 클래스를 고려합니다.

조회 최적화 클래스 1, 2, 3, 5, 7은 모두 일반적인 목적으로 사용하는 데 적합하다는 사실에 유의하십시오.

조회 컴파일 시간에서 더 많은 감소를 요구하고 클래스 0에서 실행될 SQL(예를 들어, 아주 간단한 명령문) 종류를 알고 있는 경우에만 클래스 0을 고려하십시오. 이 SQL은 다음과 같은 특성을 수반합니다.

- 하나 또는 적은 테이블에만 액세스합니다.
- 하나 또는 적은 테이블에만 페치합니다.
- 완전한 고유 색인을 사용합니다.

온라인 트랜잭션 처리(OLTP) 트랜잭션은 이러한 SQL 종류의 좋은 예입니다.

복합 조회에는 최상의 액세스 플랜을 선택하기 위해 서로 다른 최적화의 양이 필요합니다. 다음 특성을 나타내는 조회에 대해 더 높은 최적화 클래스를 사용하는 것을 고려할 수도 있습니다.

- 대형 테이블에 대한 액세스
- 많은 수의 술어
- 많은 부속 조회
- 많은 조인
- UNION 및 INTERSECT와 같은 많은 집합 연산자
- 많은 규정 행
- GROUP BY 및 HAVING 조작
- 중첩 테이블 표현식
- 많은 수의 뷰

완전히 정규화된 데이터베이스에 대한 결정 지원 조회 또는 월말 보고 조회는 적어도 기본 조회 최적화 클래스에서 사용되어야 하는 복합 조회의 좋은 예입니다.

더 높은 조회 최적화 클래스를 사용하는 또 다른 이유는 조회 생성자에 의해 생성된 SQL 때문입니다. 많은 조회 생성자는 비효율적인 SQL을 작성합니다. 조회 생성자에 의해 생성된 조회를 포함한 제대로 기록되지 않은 조회에는 좋은 액세스 플랜을 선택할 수 있는 추가 최적화가 필요합니다. 조회 최적화 클래스 2 이상의 클래스를 사용하면 제대로 기록되지 않은 SQL 조회를 향상시킬 수 있습니다.

정적 SQL 또는 동적 SQL 사용과, 같은 동적 SQL이 반복 실행되었는지는 또한 중요한 고려사항입니다. 정적 SQL의 경우, 조회 컴파일 시간과 자원은 한 번만 사용되고 결과 플랜은 여러 번 사용될 수 있습니다. 일반적으로, 정적 SQL은 기본 조회 최적화 클래스를 항상 사용해야 합니다. 동적문은 런타임 시 바인드되고 실행됩니다. 그래서 동적문에 대한 추가 최적화 오버헤드가 전체 성능을 향상시키는지 고려해야 합니다. 그러나 같은 동적 SQL문이 반복 실행되는 경우, 선택된 액세스 플랜이 캐쉬됩니다. 조회 최적화 클래스를 선택하기 위해 명령문은 정적 SQL 문처럼 처리될 수 있습니다.

(정적 SQL 및 동적 SQL 사용에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.)

추가 최적화로부터 이점을 얻을 수 있는 조회가 있지만, 컴파일 시간 및 자원 사용에 대해 확신하지 못하거나 걱정하는 경우, 일부 벤치마크 테스트를 수행할 수 있습니다. 이 테스트를 하면 다른 최적화 클래스로부터 얻은 이점을 수량화하는 데 도움을 줄 수 있습니다. 일반 기술 및 db2batch 도구의 특정 사용에 대해서는 363 페이지의 『제12장 벤치마크 테스팅』를 참조하십시오. 벤치마크 테스트를 설계하고 수행하는 경우, 응용프로그램에서 SQL문이 동적인지 정적인지를 고려하십시오.

- **동적** SQL문의 경우, 테스트시 명령문의 평균 런타임을 비교해야 합니다. 다음의 공식을 사용하면 평균 런타임을 계산하는 데 도움이 됩니다.

  $$\frac{\text{컴파일 시간} + \text{모든 반복에 대한 실행 시간의 합}}{\text{반복 수}}$$

  여기서 반복 수는 컴파일되는 시간마다 SQL문 실행이 예상되는 시간의 수를 나타냅니다.

  주: 초기 컴파일을 따라서 동적 SQL문은 환경에 대한 변경에 재컴파일이 필요한 경우에만 재컴파일됩니다. 일단 캐쉬되면 후속 PREPARE문은 환경이 변경되지 않은 것으로 간주하는 캐쉬된 명령문을 다시 사용하기 때문에 SQL문은 컴파일될 필요가 없습니다. (동적 SQL문으로 작업하는 경우 성능을 향상시킬 수 있는 캐쉬에 대한 자세한 내용은 401 페이지의 『카탈로그 캐쉬 크기(catalogcache_sz)』 및 410 페이지의 『패키지 캐쉬 크기(pckcachesz)』를 참조하십시오.)

- **정적** SQL문의 경우, 테스트시 명령문 런타임을 비교해야 합니다.

  주: 정적 SQL의 컴파일 시간을 비교하는 동안, 명령문에 대한 전체(컴파일 및 수행) 시간을 의미 있는 문맥에서 사용하기 어렵습니다. 전체 시간을 비교하면 정적 SQL문은 바인드될 때마다 여러 번 수행될 수 있고 런타임 중 보통 바인드되지 않는다는 사실을 인식하지 않습니다.

성능 향상을 위한 결과 세트에 대한 제한사항

SELECT문은 탐색 기준을 충족시키는 행 세트를 정의합니다. DB2 최적화 알고리즘은 응용프로그램이 모든 규정 행을 검색하는 것으로 가정합니다. 이러한 가정은 OLTP 및 일괄처리 환경에서 가장 적절합니다. 단, 『찾아보기』 응용프로그램에서는, 조회가 매우 큰 잠재적 응답 세트를 정의하나 첫 번째 몇 행, 일반적으로 화면을 채우는 데 필요한 행 수만큼만을 검색하는 것이 보편적입니다.

모든 규정 행 검색시 최적화 알고리즘이 세운 기본 가정은 저장된 데이터의 정보를 갱신하거나 삭제하지 않는 응용프로그램에 대해서는 최적이 아닐 수도 있습니다.

성능을 향상시키기 위해 결과 테이블을 제한하거나 수정하는 SELECT문을 수정하는 데에는 다섯 가지 방법이 있습니다. 다음과 같습니다.

- FOR UPDATE절
- FOR READ/FETCH ONLY절
- OPTIMIZE FOR n ROWS절
- FETCH FIRST n ROWS ONLY절
- DECLARE CURSOR WITH HOLD문

FOR UPDATE절

FOR UPDATE절은 바로 뒤에 오는 UPDATE문에 의해 갱신될 수 있는 컬럼을 식별합니다. FOR UPDATE절이 컬럼 이름 없이 지정되면 테이블 또는 뷰의 모든 갱신 가능 컬럼이 포함됩니다. 컬럼 이름이 지정되면 각 이름은 규정화되지 않아야 하며 테이블 또는 뷰의 컬럼을 식별해야 합니다.

FOR UPDATE절은 다음에 해당할 경우에는 사용할 수 없습니다.

- SELECT문과 연관된 커서는 삭제할 수 없습니다.
- 선택된 컬럼 중 하나 이상이 카탈로그 테이블에서 갱신 불가능한 컬럼이며, FOR UPDATE절에서 제외되지 않았습니다.

DB2 CLI 연결 속성인 SQL_ATTR_ACCESS_MODE가 CLI 응용프로그램에 사용되어 동일한 결과를 생성합니다. 자세한 내용은 *CLI Guide and Reference*의 `SQLSetConnectAttr()`절을 참조하십시오.

FOR READ 또는 FETCH ONLY절

FOR READ ONLY절은 결과 테이블이 읽기 전용이 되도록 합니다. FOR FETCH ONLY절은 동일한 의미를 가집니다.

일부 결과 테이블은 정의에 의해 읽기 전용이 됩니다. 예를 들어, 뷰의 SELECT 문에서 생성된 결과 테이블은 읽기 전용으로 정의됩니다. 이 경우에 FOR READ ONLY를 지정할 수도 있으나, 영향을 미치지는 않습니다.

갱신 및 삭제가 허용되는 결과 테이블의 경우, FOR READ ONLY를 지정하면 FETCH 조작의 성능이 향상될 수도 있습니다. 이러한 가능한 성능 향상은 데이터베이스 관리 프로그램이 데이터에 대해 독점 잠금을 하기보다는 블로킹을 할 수 있을 때 발생합니다. 조회가 파티션된 UPDATE 또는 DELETE문에 사용되는 경우를 제외하고 성능 향상을 위해서는 FOR READ ONLY절을 사용해야 합니다.

DB2 CLI 연결 속성인 SQL_ATTR_ACCESS_MODE가 CLI 응용프로그램에 사용되어 동일한 결과를 생성합니다. 자세한 내용은 *CLI Guide and Reference*의 SQLSetConnectAttr()절을 참조하십시오.

OPTIMIZE FOR n ROWS절

OPTIMIZE FOR절은 응용프로그램이 결과의 서브세트만을 검색할 것인지 또는 첫 번째 몇 행의 검색에 우선순위를 둘 것인지를 선언할 수 있는 메커니즘을 제공합니다. 일단 응용프로그램의 의도가 이해되면 최적화 알고리즘은 첫 번째 몇 행 검색에 대한 응답 시간을 최소화하는 액세스 플랜에 우선권을 줄 수 있습니다. 또한 단일 블록으로서 클라이언트에 전송된 행의 수(88 페이지의 『행 블로킹』 참조)는 OPTIMIZE FOR절의 『n』 값에 의해 바인드됩니다. 따라서 OPTIMIZE FOR절은 서버가 데이터베이스에서 규정 행을 검색하는 방법과 규정 행이 클라이언트로 리턴되는 방법에 모두 영향을 미칩니다.

예를 들어, 정기적으로 높은 급료를 받는 사원에 대한 사원 테이블을 조회한다고 가정해 보십시오.

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
   FROM   EMPLOYEE
ORDER BY SALARY DESC
```

SALARY 컬럼에서 내림차순의 색인을 정의하였습니다. 그러나 사원의 순서가 사원 번호에 의해 결정되므로, 급료 색인이 매우 부적절하게 클러스터될 가능성이 높습니다. 많은 임의의 비동기 I/O를 방지하기 위해 노력하는 최적화 알고리즘은 규정하는 모든 행의 식별자가 정렬되어야 하는 목록 프리페치 액세스 메소드(294 페이지의 『목록 프리페치의 이해』 참조)를 사용하도록 선택합니다. 이는 첫 번째 규정 행이 응용프로그램으로 리턴되기 전에 지연을 초래할 수 있습니다. 다음과 같이 명령문에 OPTIMIZE FOR절을 추가하면

```
  SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
     FROM   EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

최적화 알고리즘은 모든 가능성 중에서 단지 최고의 급료를 받는 20명의 사원만이 검색됨을 알고 곧바로 SALARY 색인을 사용하려고 합니다. 블로킹 가능한 행 수에 관계없이 20행마다 한 개의 행 블록이 클라이언트로 리턴됩니다.

OPTIMIZE FOR절을 사용하면 최적화 알고리즘은 정렬과 같은 대량 조작 또는 행 흐름을 방해하는 조작을 막는 액세스 플랜을 지원할 수 있습니다. OPTIMIZE FOR 1 ROW를 사용할 경우, 액세스 경로에 영향을 줄 가능성이 가장 높습니다. 결과적으로, 이 절을 사용하면 다음과 같은 효과가 있습니다.

- 복합 내부가 있는 조인 순서는 임시 테이블이 필요하므로 가능성이 줄어듭니다.
- 조인 메소드가 변경될 수 있습니다. 중첩된 루프 조인은 오버헤드 비용이 낮고 주로 몇 개의 행만을 검색하려고 할 경우에 더욱 효과적이므로 선택 가능성이 가장 높습니다.
- ORDER BY절에 일치하는 색인이 선택될 가능성이 더 높습니다. 이러한 상황은 ORDER BY에 정렬이 필요하지 않기 때문에 발생합니다.
- 목록 프리페치 액세스 메소드에는 정렬이 필요하므로 선택 가능성이 줄어듭니다.
- 순차 프리페치는 사용자가 적은 수의 행을 보려는 것으로 추정하므로 DB2에 의해 요청될 가능성이 적습니다.
- 조인 조회에서는 ORDER BY절에 필요한 순서화를 제공하는 색인이 외부 테이블에 있을 경우, ORDER BY절의 컬럼을 가진 테이블이 외부 테이블로서 선택되기 쉽습니다.

OPTIMIZE FOR절이 모든 최적화 클래스(74 페이지의 『최적화 클래스 조정』 참조)에 적용되기는 하지만, 최적화 클래스 3 이상에 대해 최적으로 작용합니다. 최적화 클래스 3 미만에서 Geeedy 조인 열거 메소드(204 페이지의 『최적의 조인을 선택하기 위한 검색 전략』 참조)를 사용할 경우, 처음 몇 행의 신속한 검색에는 도움이 되지 않는 다중 테이블 조인용 액세스 플랜이 생성됩니다.

OPTIMIZE FOR절은 모든 규정 행의 검색을 막지 않습니다. 그러나 규정 행을 모두 검색하는 데 걸린 총 경과 시간은 최적화 알고리즘이 전체 응답 세트를 최적화하는 데 걸린 시간보다 오래 걸릴 수가 있습니다.

호출 레벨 인터페이스(DB2 CLI 또는 ODBC)를 사용하는 패키지 응용프로그램이 있을 경우, `db2cli.ini` 구성 파일의 OPTIMIZEFORNROWS 키워드를 사용하여 DB2 CLI가 자동으로 각 조회 명령문 끝에 OPTIMIZE FOR절을 추가하도록 할 수 있습니다. 자세한 내용은 *CLI Guide and Reference* 매뉴얼을 참조하십시오.

별칭으로부터 데이터를 선택할 경우, 결과는 데이터 소스 지원에 따라 다를 수 있습니다. 별칭으로 참조되는 데이터 소스가 OPTIMIZE FOR절을 지원하고 DB2 최적화 알고리즘이 절이 들어 있는 전체 조회를 데이터 소스로 푸시다운하면 절은 원격 SQL로 생성되어 데이터 소스로 전송됩니다. 데이터 소스가 이 절을 지원하지 않거나 최적화 알고리즘이 절을 지역적으로 실행하기로 결정한 경우(최소 비용 플랜), OPTIMIZE FOR절은 DB2에서 지역적으로 적용됩니다. 이 경우, DB2 최적화 알고리즘은 액세스 플랜에 조회의 처음 몇 행을 검색할 때 걸리는 응답 시간을 최소화하기 위한 선호사항을 제공하며, 플랜 작성시 최적화 알고리즘에 사용 가능한 옵션은 한계가 있으며, OPTIMIZE FOR절로부터 얻은 성능은 대단하지 않습니다.

FETCH FIRST절과 OPTIMIZE FOR절을 모두 지정할 경우, 두 값 중 더 적은 값이 통신 버퍼 *크기*를 설정하는 데 사용됩니다. 최적화를 위해 두 값은 각각 서로 별개인 것으로 간주됩니다. 두 절간의 상호작용에 대한 자세한 내용은 90 페이지의 『SELECT문 사용』을 참조하십시오.

FETCH FIRST n ROWS ONLY절

OPTIMIZE FOR *n* ROWS절은 모든 규정 행의 검색을 막지 않습니다. 그러나 모든 규정 행을 검색하는 데 걸린 총 경과 시간은 최적화 알고리즘이 전체 응답 세트를 최적화하는 데 걸린 시간보다 오래 걸릴 수 있습니다.

FETCH FIRST *n* ROWS ONLY절은 SELECT문 내에서 검색할 수 있는 최대 행 수를 설정합니다. 결과 테이블을 처음 몇 행으로 제한하면 성능을 향상시킬 수 있습니다. 이 절이 지정되지 않은 SELECT를 기본으로 한 결과 테이블에 있을 수 있는 행 수에 관계없이 n행만이 검색됩니다.

FETCH FIRST절과 OPTIMIZE FOR절을 모두 지정할 경우, 두 값 중 더 적은 값이 통신 버퍼 *크기*를 설정하는 데 사용됩니다. 최적화를 위해 두 값은 각각 서로 별개인 것으로 간주됩니다. 두 절간의 상호작용에 대한 자세한 내용은 90 페이지의 『SELECT문 사용』을 참조하십시오.

DECLARE CURSOR WITH HOLD문

커서를 WITH HOLD절이 들어 있는 DECLARE CURSOR문을 사용하여 선언하면 임의의 열린 커서는 트랜잭션이 확약될 때 열린 상태로 남아 있습니다. 또한 열린 WITH HOLD 커서의 현재 커서 위치를 보호하는 잠금을 제외한 모든 잠금은 해제됩니다.

커서를 WITH HOLD절이 들어 있는 DECLARE CURSOR문을 사용하여 선언하면 열린 모든 커서는 트랜잭션이 ROLLBACK으로 종료되면 닫힙니다. 또한 모든 잠금은 해제되고 LOB 위치 지정자는 사용 가능해집니다.

CLOSE CURSOR문에 대한 다른 기본 절과의 비교에 대한 자세한 내용은 72 페이지의 『릴리스로 커서 닫기』를 참조하십시오.

DB2 CLI 연결 속성인 SQL_ATTR_CURSOR_HOLD를 CLI 응용프로그램에 사용하여 동일한 결과를 얻을 수 있습니다. 자세한 내용은 *CLI Guide and Reference* 의 『SQLSetStmtAttr - Set Options Related to a Statement』절을 참조하십시오.

콜 레벨 인터페이스(DB2 CLI 또는 ODBC)를 사용하는 패키지 응용프로그램이 있는 경우, `db2cli.ini` 구성 파일의 CURSORHOLD 키워드를 사용하여 DB2

CLI가 자동으로 선언된 모든 커서에 대해 WITH HOLD절이 설정된 것으로 가정하도록 할 수 있습니다. 자세한 내용은 *CLI Guide and Reference*의 트랜잭션 구성 키워드 절을 참조하십시오.

행 블로킹

행 블록킹은 한 번의 조작으로 행의 블록을 검색함하여 데이터베이스 관리 프로그램의 오버헤드를 줄이는 기술입니다. 이들 행은 캐쉬에 저장되고 응용프로그램의 각 FETCH 요청은 캐쉬에서 다음 행을 확보합니다. 블록 내의 모든 행이 처리되면 다른 행 블록이 데이터베이스 관리 프로그램에 의해 검색됩니다.

응용프로그램이 OPEN CURSOR 요청을 발행하면 캐쉬가 할당되고, 커서가 닫히면 캐쉬는 할당되지 않습니다. 캐쉬 *크기*는 I/O 블록에 메모리를 할당하는 데 사용된 구성 매개변수에 의해 결정됩니다. 사용되는 매개변수는 클라이언트가 지역인지 원격인지에 따라 결정됩니다.

- *지역 응용프로그램*의 경우, 매개변수 *aslheapsz*를 사용하여 행 블로킹에 캐쉬를 할당합니다. (이 매개변수에 대한 자세한 내용은 429 페이지의 『응용프로그램 지원 계층 힙 *크기*(aslheapsz)』를 참조하십시오.)
- *원격 응용프로그램*의 경우, 클라이언트 워크스테이션의 매개변수 *rqrioblk*를 사용하여 행 블로킹에 *캐쉬*를 할당합니다. 캐쉬는 데이터베이스 클라이언트에 할당됩니다. (이 매개변수에 대한 자세한 내용은 432 페이지의 『클라이언트 I/O 블록 *크기*(rqrioblk)』를 참조하십시오.).

지역 응용프로그램의 경우, 다음 공식을 사용하여 블록마다 리턴될 행 수를 계산할 수 있습니다.

- *aslheapsz*는 메모리 페이지에 있습니다.
- 4 096은 페이지당 바이트 수입니다.
- *orl*은 출력 행 길이(바이트 단위)입니다.

블록당 행 수 = `aslheapsz * 4096 / orl`

원격 응용프로그램의 경우, 다음 공식을 사용하여 블록마다 리턴될 행 수를 계산할 수 있습니다.

- *rqrioblk*은 메모리(바이트 단위)입니다.
- *orl*은 출력 행 길이(바이트 단위)입니다.

블록당 행 수 = rqrioblk / orl

SELECT문에 FETCH FIRST *n* ROWS ONLY절 또는 OPTIMIZE FOR *n* ROWS절을 사용할 경우, 블록당 행 수는 다음 중 최소값임에 유의하십시오.

- 위의 공식에서 계산된 값
- FETCH FIRST절의 *n* 값
- OPTIMIZE FOR절의 *n* 값

PREP 및 BIND 명령에 BLOCKING 옵션을 사용하여 다음 유형의 행 블로킹을 지정하십시오.

UNAMBIG
: 블로킹은 읽기 전용 커서와 『FOR UPDATE OF』로 지정되지 않은 커서에서 발생합니다. 불명확한 커서는 갱신 가능한 것으로 처리됩니다.

ALL
: 블로킹은 읽기 전용 커서와 『FOR UPDATE OF』로 지정되지 않은 커서에서 발생합니다. 불명확한 커서는 읽기 전용으로 처리됩니다.

NO
: 블로킹이 모든 커서에서 발생하는 것은 아닙니다. 불명확한 커서는 읽기 전용으로 처리됩니다.

이들 유형의 행 블로킹에 대한 자세한 내용은 *Command Reference* 매뉴얼의 PREP 및 BIND 명령 설명을 참조하십시오.

PREP 및 BIND 명령에서 지정된 옵션이 없는 경우, 기본 행 블로킹 유형은 UNAMBIG입니다. 명령행 처리기와 콜 레벨 인터페이스의 경우, 기본 행 블로킹 유형은 ALL입니다.

커서에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

조회 조정

이 절에서는 응용프로그램에서 사용자가 SQL문을 세분 조정하는 데 도움을 주는 특정 고려사항과 지침에 대해 설명합니다. 일반적으로, 이 지침은 시스템 자원 사용과 대형 테이블의 데이터에 액세스하는 데 필요한 시간을 최소화하는 프로그램을 설계하는 데 도움을 줍니다. SQL문이 컴파일될 때 발생하는 최적화 양에 따라 SQL문을 세분 조정할 필요는 없습니다. SQL 컴파일러는 SQL을 더 효율적인

양식으로 재작성할 수 있습니다. 자세한 내용은 171 페이지의 『SQL 컴파일러에 의한 조회 재작성』 및 74 페이지의 『최적화 클래스 조정』을 참조하십시오.

최적화 알고리즘에 의해 선택된 액세스 플랜은 환경상의 고려사항과 시스템 카탈로그 통계를 포함한 다른 인수에 의해서도 영향을 받는다는 사실에 유의하는 것도 중요합니다. 응용프로그램의 성능을 테스트하는 벤치마크를 유도하는 경우, 액세스 플랜을 향상시킬 수 있게 조정할 수 있습니다.

SELECT문 사용

SQL 언어는 융통성이 높은 고급 언어입니다. 이러한 결과로, *Select* 문은 같은 데이터를 검색하기 위해 기록될 수 있습니다. 그러나 다른 양식과 다른 최적화 클래스에 따라 성능은 달라질 수 있습니다.

SQL 컴파일러(조회 재작성과 최적화 단계를 포함함)는 코딩한 조회에 대한 결과 세트를 생성하는 액세스 플랜을 선택합니다. 그러므로 다음 지침에서 많이 설명된 내용처럼 사용자의 조회를 필요한 데이터만 확보하도록 코딩해야 합니다.

SELECT문 사용 시기에 대한 지침

*select*문 사용에 대한 지침은 다음과 같습니다.

- 선택 목록에서 필요한 컬럼만을 지정하십시오. 별표(*)로 모든 컬럼을 지정하는 것이 더 간단하지만, 불필요한 처리와 원하지 않는 컬럼의 리턴이 발생할 수 있습니다.
- 필요한 행에 대한 응답 세트만을 제한하는 술어를 사용하여 선택된 행 수를 제한하십시오. (다른 술어 유형 및 술어가 관련 성능에 미치는 영향에 다한 자세한 내용은 195 페이지의 『술어에 관련된 용어』을 참조하십시오.)
- 사용하려는 행 수가 리턴될 수 있는 전체 행 수보다 상당히 적은 경우, *select*문으로 OPTIMIZE FOR절을 지정하십시오. 이 절을 사용하면 통신 버퍼에서 블록화된 행 수뿐만 아니라 액세스 플랜의 선택사항에 영향을 줍니다. (자세한 내용은 88 페이지의 『행 블로킹』을 참조하십시오.)
- 검색되는 행 수가 작으면 FETCH FIRST n ROWS ONLY절에 OPTIMIZE FOR k ROWS절을 추가할 필요가 없습니다. 그러나 n이 크고 후속의 k 행에 가능한 지연으로 첫 번째 k 행을 확보하여 최적화하려는 경우, 둘 다 지정하십시오. 통신 버퍼는 n과 k 중 더 작은 크기로 설정됩니다.

```
SELECT EMPNAME, SALARY FROM EMPLOYEE
   ORDER BY SALARY DESC
      FETCH FIRST 100 ROWS ONLY
   OPTIMIZE FOR 20 ROWS
```

- FOR READ ONLY(또는 FOR FETCH ONLY)절을 지정하면 사용자의 조회가 행 블로킹을 이용할 수 있어 성능을 향상시킬 수 있습니다. 지정된 이 절을 사용한 조회에 의해 검색된 행에서 독점 잠금은 절대로 보유되지 않으므로, 데이터 동시성을 또한 향상시킬 수 있습니다. 또한 추가 조회 재작성이 발생할 수 있습니다. 연합 시스템에서는 BLOCKING ALL BIND와 함께 FOR READ ONLY(또는 FOR FETCH ONLY)절을 지정하면 별칭에 대한 조회의 성능을 향상시킬 수 있습니다.
- FOR UPDATE OF절을 지정하면 갱신될 커서 성능을 또한 향상시킬 수 있으며 데이터베이스 관리 프로그램이 처음부터 더 적합한 잠금 레벨을 선택하여 잠재적인 교착 상태(64 페이지의 『교착 상태』 참조)와 잠금 변환(60 페이지의 『잠금 변환』 참조)을 피할 수 있게 하여 성능을 향상시킬 수 있습니다.
- 가능하면 숫자 데이터 유형 변환은 항상 피하십시오. 값을 비교하는 경우, 같은 데이터 유형을 가진 항목을 사용하는 것이 더 효율적입니다. 변환이 필요한 경우, 제한된 정밀도로 인한 부정확성과 런타임 변환으로 인한 성능 비용이 발생할 수 있습니다.

  가능하면 다음 데이터 유형을 사용하십시오.
  - 짧은 컬럼에 대해 변하지 않는 문자
  - 부동 소수 또는 십진수보다는 정수
  - 문자보다는 날짜시간
  - 문자보다는 숫자
- DISTINCT 또는 ORDER BY와 같은 절이나 조작이 들어 있는 SQL문에는 조작을 수행하기 위해 순서화된 데이터가 필요합니다. 정렬 조작이 사용되는 기회를 줄이려는 경우, 필요하지 않으면 이 절의 스펙을 생략하십시오.
- 테이블에서 행의 존재를 확인하려면 다음을 사용하지 마십시오.

  ```
  SELECT COUNT(*) FROM  TABLENAME
  ```

  그리고 매우 작은 테이블이 아니면 0이 아닌 값을 확인하십시오. 테이블이 커질수록 모든 행을 계수면 성능에 영향이 미칩니다. 그 대신, 단일 행을 선택하

도록 제시됩니다. 커서를 열고 한 행을 페치하거나 단일 행(SELECT INTO)을 선택하여 수행될 수 있습니다. (둘 이상의 행이 *select*문에 있는 경우, SQLCODE -811 오류를 반드시 점검하십시오.)

- 갱신 활동이 적고 테이블이 큰 경우, 술어로 자주 사용되는 컬럼에서 색인을 정의하십시오.
- 여러 술어 절에 같은 컬럼이 나타날 경우, IN 목록을 사용할 것을 고려해 보십시오.
- 호스트 변수와 함께 사용되는 큰 IN 목록의 경우, 호스트 변수의 부속 집합을 loopin하면 성능이 향상됩니다.

다음의 제안사항은 여러 테이블에 액세스하는 *select*문에 특별히 적용합니다.

- 테이블을 조인하는 경우 Join 술어를 사용하십시오. (Join 술어는 조인의 다른 테이블에서 두 컬럼간의 비교입니다.)
- 조인이 더 효율적으로 처리될 수 있도록 Join 술어의 컬럼에서 색인을 정의하십시오. 이렇게 하면 여러 테이블에 액세스하는 select문이 들어 있는 UPDATE 및 DELETE문에서도 도움이 될 수 있습니다.
- 가능하면 Join 술어를 가진 표현식이나 OR절을 사용하지 마십시오. 이 경우, 데이터베이스 관리 프로그램이 일부 조인 기술을 사용할 수 없습니다. 그 결과, 가장 효율적인 조인 메소드를 선택하지 못할 수도 있습니다.
- 가능하면 파티션된 데이터베이스 환경에서 조인된 테이블이 조인 컬럼에서 모두 파티션되었는지 확인하십시오.

자세한 내용은 197 페이지의 『조인 개념』을 참조하십시오.

또한 조인 및 부속 조회를 가진 SQL문 코딩에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

복합 SQL

복합 SQL을 사용하여 여러 개의 SQL문을 하나의 실행 블록으로 그룹화할 수 있습니다. 블록(*부속 명령문*) 내에 포함된 SQL문은 개별적으로 실행될 수 있습니다. 그러나 명령문 블록을 작성한 다음 실행하여 데이터베이스 관리 프로그램 오버헤드를 줄일 수 있습니다. 원격 클라이언트의 경우, 복합 SQL은 네트워크를 통해 전송되어야 할 요청의 수도 줄여줍니다.

복합 SQL에는 다음의 두 가지 종류가 있습니다.

- **최소단위**

  모든 부속 명령문이 성공적으로 완료되거나 부속 명령문 중 하나가 오류로 종료한 경우, 응용프로그램은 데이터베이스 관리 프로그램으로부터 오류를 수신합니다. 하나의 부속 명령문이라도 오류로 종료하면, 전체 블록은 오류로 종료한 것으로 간주되며, 블록 내에서 데이터베이스에 대한 변경사항은 모두 구간 복원됩니다.

- **비최소단위**

  응용프로그램은 모든 부속 명령문이 완료되면 데이터베이스 관리 프로그램으로부터 응답을 수신합니다. 블록 내의 모든 부속 명령문은 바로 이전의 부속 명령문이 성공적으로 완료되었는지 여부에 관계없이 실행됩니다. NOT ATOMIC 복합 SQL을 포함하고 있는 작업 단위(UOW)가 구간 복원될 때에만 명령문의 그룹을 구간 복원할 수 있습니다.

- 최소단위 복합 SQL은 DB2 Connect와 함께 지원되지 않습니다.
- 복합 SQL은 저장 프로시듀어(DARI 루틴이라고도 함) 내에서 지원됩니다.
- 복합 SQL은 다음을 통해 지원됩니다.
  - Embedded 정적 SQL(*SQL 참조서* 매뉴얼 참조)
  - DB2 CLI(*CLI Guide and Reference* 매뉴얼 참조)
  - JDBC(*응용프로그램 개발 안내서* 매뉴얼 참조)

동적 복합 명령문

동적 복합 명령문 그룹과 그 밖의 SQL문은 함께 실행 블록으로 됩니다. 동적 복합 명령문 내에서는 SQL 변수 및 SQLSTATE와 연관된 조건을 선언할 수 있고 여러 개의 SQL 절차 명령문을 가질 수 있습니다. 동적 복합 명령문에서 오류가 발생하면 모든 이전 SQL문은 구간 복원되고 동적 복합 명령문의 나머지 SQL문은 처리되지 않습니다.

동적 복합 명령문이 트리거, SQL 함수, SQL 메소드에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 실행 명령문은 동적으로 준비될 수 있습니다. 명령문을 호출할 때는 특권은 필요하지 않지만 복합 명령문 내 Embedded SQL 문을 호출할 때는 명령문과 연관된 권한 부여 ID에 필수 특권이 있어야 합니다.

변수는 변수 선언의 부속 명령문에 나타납니다. 조건은 조건 선언의 SQLSTATE 값에 기반하여 부속 명령문에 나타납니다. 동적 복합 명령문은 단일 명령문으로 DB2에 의해 컴파일됩니다. 이 명령문은 제어 흐름 논리는 적지만 상당한 데이터 흐름을 있는 짧은 스크립트에 효과적으로 사용될 수 있습니다. 중첩된 복합 제어 흐름이 있는 더 큰 구조에 대해서는 SQL 프로시듀어 사용을 고려하십시오.

동적 복합 명령문 내에서 사용될 수 있는 몇 가지 제어 흐름 논리문이 있습니다. 이러한 논리문으로는 FOR문, IF문, ITERATE문 및 WHILE문이 있습니다. 이러한 명령문과 기타 지원되는 명령문에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

성능 고려사항 및 문자 변환

응용프로그램과 데이터베이스가 서로 다른 코드 페이지를 사용하고 있는 경우, 가능한 경우 하나의 코드 페이지에서 다른 코드 페이지로의 데이터 맵핑이 발생합니다. 이 때 응용프로그램과 데이터베이스 코드 페이지 사이에서 데이터를 적절히 맵핑하기 위해서는 데이터 변환이 요구되는 경우도 생깁니다.

이러한 맵핑과 데이터 변환은 데이터베이스 코드 페이지와는 다른 코드 페이지에서 수행 중인 응용프로그램의 처리 시간에 오버헤드를 초래할 수도 있습니다. 응용프로그램 및 데이터베이스가 동일한 코드 페이지 또는 식별 조합 순서를 사용할 경우, 응용프로그램의 성능을 향상시킬 수 있습니다.

코드 페이지 변환

다음은 문자 변환이 발생할 수 있는 상황입니다.

- 데이터베이스에 액세스하는 클라이언트 또는 응용프로그램이 데이터베이스의 코드 페이지와 다른 코드 페이지에서 수행 중일 때 발생할 수 있습니다.

  데이터베이스 변환은 데이터베이스 서버 머신에서 발생하는데, 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로, 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 발생합니다.
- 파일을 가져오는(또는 로드하는) 클라이언트 또는 응용프로그램이 자져올(또는 로드되는) 파일과 다른 코드 페이지에서 수행할 때 발생할 수 있습니다.
- DB2 Connect를 사용하여 DRDA 서버에서 데이터에 액세스할 때 발생할 수 있습니다.

문자 변환은 다음에 대해서는 발생하지 **않습니다**.

- 파일 이름
- 목표로 지정되거나 돌아올 데이터, 2진 데이터용 속성으로 지정된 컬럼 또는 결과가 2진 데이터용이거나 BLOB 데이터인 SQL 조작에서 사용된 데이터
- 설치된 EUC나 UCS-2로, 또는 설치된 ECU나 UC2-2에서 지원된 변환 함수를 가진 DB2 제품이나 플랫폼. 응용프로그램을 수행할 때 SQLCODE -332(SQLSTATE 57017)를 수신합니다.

EUC 코드 페이지 지원 및 지국어 지원(NLS) 고려사항에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

운영 체제 환경에 따라 복수 바이트 코드 페이지를 변환하는 경우, DB2 데이터베이스 관리 프로그램은 변환 함수와 변환 테이블 또는 DBCS 변환 API를 사용합니다.

주: 복수 바이트 코드 페이지간의 문자열 변환은 ECU를 가진 DBCS와 같이 문자열의 길이를 늘리거나 줄입니다.

일부 국가의 PC DBCS, EUC, UCS-2 코드 세트에서 다른 문자로 지정된 코드 포인트는 같은 문자를 정렬하는 경우 다른 결과를 나타냅니다. 다른 국가의 코드 세트를 통해 정렬이 필요한 경우에도 *관리 안내서: 계획*을 참조하십시오.

EUC(Extended UNIX Code) 코드 페이지 지원

C 또는 C++ 응용프로그램에서 그래픽 데이터를 사용하는 호스트 변수를 사용하면 특수 사전 처리 컴파일러, 응용프로그램 성능, 응용프로그램 설계 관련 문제를 포함한 특수 고려사항이 필요합니다.

응용프로그램이 EUC 코드 세트를 필요로 하도록 개발된 경우, *Administrative API Reference* 매뉴얼을 참조해야 합니다.

그래픽 데이터에 대한 데이터베이스 및 클라이언트 응용프로그램 지원(즉, 2바이트 문자)은 일본어 및 대만어 ECU 코드 페이지 모두에 있는 많은 문자를 사용할 경우에 2바이트 넓이 제한사항을 극복해야 합니다. 이 EUC 코드 페이지의 그래픽 데이터는 UCS-2 코드 세트를 사용하여 저장되고 조작됩니다.

저장 프로시듀어

데이터베이스 응용프로그램 환경에서 대부분의 상황은 반복적으로 일어납니다(예: 고정 데이터 세트 수신, 데이터베이스에 대해 동일한 다중 요청 수행 또는 고정 데이터 세트 리턴). 저장 프로시듀어를 사용하면, 원격 데이터베이스를 한 번만 호출하여 사전 프로그래밍된 프로시듀어를 실행할 수 있습니다. 한 번의 호출로 데이터베이스에 대한 여러 액세스를 나타낼 수도 있습니다.

원격 데이터베이스에 대해 단일 SQL문을 처리하면 두 가지 전송(한 번의 요청 및 한 번의 수신)을 송신해야 합니다. 그러나 응용프로그램에는 여러 SQL문이 들어 있을 수 있습니다. 저장 프로시듀어가 없으면 응용프로그램에서 작업을 완료하는 데 많은 전송이 필요합니다.

데이터베이스 클라이언트가 저장 프로시듀어를 사용하는 경우, 전체 프로세스에는 두 가지 전송만 필요합니다. 즉, 네트워크 전송의 수를 줄이는 것입니다. 저장 프로시듀어를 호출하려면 저장 프로시듀어를 호출하기 전에 먼저 프로시듀어를 포함하고 있는 데이터베이스에 요청하는 응용프로그램이 연결되어야 합니다.

일반적으로, 이러한 저장 프로시듀어는 데이터베이스 에이전트와는 별도의 프로세스에서 수행됩니다. 이렇게 별도로 수행되기 때문에 저장 프로시듀어와 에이전트 프로세스는 라우터를 통해서만 통신해야 합니다. 저장 프로시듀어에 대해 가능한 최상의 성능을 얻으려는 경우, 저장 프로시듀어를 『신뢰(trusted)』 또는 『비분리(not

fenced)』 상태로 식별할 수 있습니다. 그 결과 데이터베이스 에이전트 프로세스에서 프로시듀어를 직접 수행할 수 있습니다. 여기서 『신뢰』와 『비분리』의 의미는 다음과 같습니다.

- *비분리*는 저장 프로시듀어를 데이터베이스 에이전트가 사용하는 데이터베이스 제어 구조와 분리시키지 않음을 의미합니다.
- *신뢰*는 관리자가 저장 프로시듀어가 사고로 또는 고의적으로 데이터베이스 제어 구조를 손상시키지 않을 것을 확신하고 있음을 의미합니다. 즉, 저장 프로시듀어가 데이터베이스의 무결성을 해치지 않는 방식으로 작동됨을 확신할 수 있습니다.

이 두 용어는 같은 의미입니다. 즉, 저장 프로시듀어가 "비분리(not fenced)"일 경우, "신뢰(trusted)"도 됩니다. 데이터베이스가 손상을 입을 위험이 있기 때문에 최대한의 성능을 얻으려면 비분리 저장 프로시듀어를 **사용해야** 합니다. 또한 비분리 저장 프로시듀어로 수행하기 전에 해당 프로시듀어의 코딩이 제대로 되어 있는지 확인한 다음 충분한 테스트를 거쳐야 합니다. 이러한 비분리 저장 프로시듀어 중 하나를 수행하는 중에 치명적인 오류가 발생하면 데이터베이스 관리 프로그램은 오류가 응용프로그램 또는 데이터베이스 관리 프로그램 코드에서 발생했는지를 판단하여 적당한 복구를 수행해야 합니다.

이는 복구를 넘어서 데이터베이스 관리 프로그램을 훼손하여, 결국 비분리(not fenced) 저장 프로시듀어가 데이터를 유실시키거나 데이터베이스를 훼손할 가능성이 있습니다. 신뢰성 있는 저장 프로시듀어를 수행할 때 대단한 주의가 필요합니다. 거의 모든 경우에 응용프로그램의 적절한 성능 분석으로 인해 이 옵션을 사용하지 않고도 원하는 성능을 얻을 수 있습니다. 예를 들어, 성능은 트리거 사용을 통해 향상시킬 수 있습니다.

다음 두 가지 방법으로 저장 프로시듀어를 비분리(not fenced)로 작성할 수 있습니다.

- CREATE PROCEDURE 명령을 사용하고 NOT FENCED절을 지정합니다.
- 플랫폼에 대한 *빠른 시작* 매뉴얼에서 정의된 대로 특정 디렉토리에 프로시듀어를 지정합니다. (이 방법은 Java 저장 프로시듀어에는 해당되지 않습니다.)

저장 프로시듀어를 수행하려면 프로시듀어를 수행하는 응용프로그램을 호출하는 일반 사용자에게는 런타임 시 다음 특권이 있어야 합니다.

- 저장 프로시듀어와 연관된 패키지에 대한 EXECUTE 또는 CONTROL 특권
- SYSADM 또는 DBADM 권한

저장 프로시듀어를 사용한 프로그램 작성에 대한 자세한 내용은 *응용프로그램 개발 안내서* 매뉴얼을 참조하십시오.

데이터베이스 활성화

데이터베이스가 시작될 때 몇 가지 유형의 데이터가 캐쉬됩니다. 예를 들어, 데이터 버퍼는 버퍼 풀에 캐쉬되며, 패키지와 동적 SQL문은 패키지 캐쉬에 캐쉬됩니다.

어떠한 사용자도 데이터베이스에 연결할 수 없는 짧은 기간이 빈번히 발생하며 이 기간이 몇몇 사용자가 데이터베이스에 연결되는 기간과 섞이게 될 경우, 캐쉬가 자주 손상되기 때문에 캐쉬으로 인한 이점은 사라집니다. 이러한 상황을 피하려면 다음 명령을 발행하여 데이터베이스를 활성화하는 것을 고려해 보십시오.

```
DB2 ACTIVATE DATABASE database
```

이 명령은 지정된 데이터베이스를 활성화하고 필요한 모든 서비스를 시작하여 모든 응용프로그램을 데이터베이스에 연결하고 사용할 수 있도록 합니다. ACTIVATE DATABASE에 의해 초기화된 데이터베이스는 DEACTIVATE DATABASE 또는 *db2stop*에 의해 종료될 수 있습니다. 이들 명령에 대한 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

응용프로그램의 병렬 처리

DB2에 의해 지원된 병렬 처리 환경의 유형은 대칭 멀티프로세서(SMP) 머신이 필요한 유형입니다. 이 환경에서 둘 이상의 프로세서는 데이터베이스에 대한 액세스를 공유합니다. 이렇게 하면 프로세서 간에 분리된 복합 SQL 요청의 병렬 실행을 할 수 있습니다.

CURRENT DEGREE 특수 레지스터 또는 DEGREE 바인드 옵션을 사용하여 응용프로그램 컴파일시 구현할 병렬 처리 수준을 지정할 수 있습니다. "수준"은 간단히 말해 동시에 실행되는 조회 부분의 수를 나타냅니다. 프로세서 수와 병렬 처리 수준에서 선택된 값 사이의 뚜렷한 관계는 없습니다. 하드웨어 플랫폼에서 사

용하는 사용 가능한 전체 프로세서 수가 응용프로그램 수행시 필요하지 않습니다. 이 수의 이상 또는 이하를 선택할 수 있습니다.

각 병렬 처리 수준은 시스템 메모리와 CPU 오버헤드를 추가합니다.

병렬 처리를 이용할 때 일부 구성 매개변수는 성능의 최적화를 위해 수정해야 합니다. 병렬 처리 수준이 높은 환경에서는 필요에 따라 공유 메모리 양을 제어하고 프리페치하는 구성 매개변수를 검토하고 수정해야 합니다. 병렬 조작 및 파티션된 데이터베이스 환경에 관련된 매개변수 목록에 대한 자세한 내용은 535 페이지의 『파티션된 데이터베이스』를 참조하십시오.

파티션 내 병렬 처리를 제어하고 관리하는 데 사용할 수 있는 세 가지의 구성 매개변수가 있습니다. 첫 번째는 *intra_parallel* 데이터베이스 관리 프로그램 구성매개변수로, 병렬 처리 지원을 사용하거나 사용하지 않도록 합니다. 두 번째는 *max_querydegree* 데이터베이스 구성 매개변수로, 데이터베이스에서의 조회에 대한 병렬 처리 수준의 상한을 설정합니다. 이 값은 CURRENT DEGREE 특수 레지스터와 DEGREE 바인드 옵션을 대체합니다. 세 번째 구성 매개변수는 *dft_degree* 데이터베이스 구성 매개변수입니다. 이 매개변수는 CURRENT DEGREE 특수 레지스터와 DEGREE 바인드 옵션의 기본값을 설정합니다.

둘 이상의 병렬 처리 수준을 사용할 경우의 응용프로그램 사용 및 포함에 대한 자세한 내용은 *응용프로그램 개발 안내서* 매뉴얼을 참조하십시오.

조회가 DEGREE = ANY로 수행하면 데이터베이스 관리 프로그램은 프로세서의 수와 조회 특성을 포함하여 인수의 수를 근거로 파티션 내 병렬 처리 수준을 선택합니다. 런타임 시 사용되는 실제 수준은 이들 인수에 따라 프로세서의 수보다 적을 수 있습니다.

데이터베이스 활동에 따라 조회 실행 이전에 명령문이 컴파일되고 조정될 수 있는 경우, 병렬 처리 수준은 SQL 최적화 알고리즘에 의해 결정됩니다. 시스템이 너무 많이 사용되면 병렬 처리 수준은 SQL 최적화 알고리즘이 선택한 것보다 낮을 수 있습니다. 이는 파티션 내 병렬 처리가 시스템 자원을 적극적으로 사용하여 다른 데이터베이스 사용자의 성능에 역효과를 미칠 만큼 조회 경과 시간을 단축하기 때문에 발생할 수 있습니다.

SQL 최적화 알고리즘이 선택한 병렬 처리 수준은 액세스 플랜을 표시하기 위해 SQL Explain 기능을 사용하여 찾을 수 있습니다. 런타임 시 사용되는 병렬 처리 수준은 데이터베이스 시스템 모니터를 사용하여 찾을 수 있습니다. SQL Explain 기능 및 관련 도구에 대한 자세한 내용은 243 페이지의 『제7장 SQL Explain 기능』 및 643 페이지의 『부록C. SQL Explain 도구』를 참조하십시오. 추가 모니터 정보는 *시스템 모니터 안내 및 참조서*를 참조하십시오.

주: 병렬 처리 "수준"은 하드웨어 환경과는 상관없이 별도로 설정할 수 있습니다. 즉, SMP 머신 없이 병렬 처리 수준을 사용할 수 있습니다. 예를 들어, 단일 프로세서 머신에서 "I/O-바인드" 조회의 경우, 수준을 "2" 이상으로 선언하면 좋습니다. 이 경우, 단일 프로세서는 새 조회 작업을 하기 전에 I/O 타스크가 완료되기까지 기다리지 않아도 됩니다. 수준을 "2" 이상으로 선언하면 단일 프로세서 머신에서 I/O 병렬 처리를 직접 제어할 수 없습니다. Load와 같은 유틸리티는 이러한 선언과 상관없이 I/O 병렬 처리를 제어할 수 있습니다. ANY라는 키워드는 *dft_degree*를 변경할 때에도 사용할 수 있습니다. ANY를 사용하면 최적화 알고리즘이 파티션 내 병렬 처리 수준을 판별하게 됩니다.

여러 경우에서 *데이터베이스 에이전트*를 사용하여 병렬 실행을 조정할 수 있습니다. 자세한 정보 및 데이터베이스 에이전트에 영향을 미치는 여러 데이터베이스 관리 프로그램 구성 매개변수 목록에 대한 자세한 내용은 311 페이지의 『데이터베이스 에이전트』를 참조하십시오.

제4장 환경상의 고려사항

응용프로그램의 설계 및 코딩시 고려해야 하는 인수(45 페이지의 『제3장 응용프로그램 고려사항』 참조) 외에도, 응용프로그램에 대해 선택된 액세스 플랜에 영향을 미칠 수 있는 환경 인수가 있습니다.

- 조회 최적화에 영향을 주는 구성 매개변수
- 조회 최적화에 미치는 노드 그룹의 영향
- 조회 최적화에 미치는 테이블 공간의 영향
- 색인화가 조회 최적화에 미치는 영향
- 연합 데이터베이스 조회에 영향을 미치는 서버 옵션

SQL 최적화 알고리즘에 영향을 미치는 인수에 대한 자세한 내용은 125 페이지의 『제5장 시스템 카탈로그 통계』를 참조하십시오.

응용프로그램과 환경을 조정하는 경우, 위의 영역에서 변경한 후에 리바인드하십시오. 이는 최상의 액세스 플랜이 사용되고 있음을 확인합니다.

조회 최적화에 영향을 주는 구성 매개변수

몇 가지 구성 매개변수가 SQL 컴파일러에 의해 선택된 액세스 플랜에 영향을 미칩니다. 대부분의 매개변수가 하나의 데이터베이스 파티션에 적합하고 일부 매개변수는 파티션된 데이터베이스에만 적합합니다. 파티션된 데이터베이스에서 구성 매개변수에 대해 작업할 때 각 매개변수에 사용된 값과 모든 파티션에서 사용된 값은 같은 것이 좋습니다.

연합 시스템에서 작업할 때 조회의 대부분이 별칭을 액세스할 경우에는 환경을 변경하기 전에 전송 중인 조회의 유형을 고려하십시오. 예를 들어, 버퍼 풀은 데이터 소스로부터 페이지를 캐쉬하지 않습니다. 그렇기 때문에 *buffpage* 매개변수 값을 늘리더라도, 최적화 알고리즘이 별칭을 포함하는 조회에 대한 액세스 플랜을 작성할 때 추가로 대안을 고려한다고는 보장할 수 없습니다. (데이터 소스는 연합 시스템 내의 DBMS 및 데이터입니다.) 또한 최적화 알고리즘은 데이터 소스 테이블

© Copyright IBM Corp. 1993, 2001

의 지역 구체화는 최소 비용이 드는 방법이거나 정렬 조작에 대한 필수 단계라는 결정을 내릴 수 있습니다. 이 경우, DB2 Universal Database에서 사용 가능한 자원을 늘리면 성능이 향상될 수 있습니다. 자세한 내용은 119 페이지의 『연합 데이터베이스 조회에 영향을 미치는 서버 옵션』 및 396 페이지의 『데이터베이스 공유 메모리』를 참조하십시오.

다음은 SQL 컴파일러가 선택한 액세스 플랜에 영향을 미치는 구성 매개변수 목록입니다.

- 396 페이지의 『버퍼 풀 *크기*(buffpage)』.

  액세스 플랜을 선택할 때 최적화 알고리즘은 디스크에서 버퍼 풀로 페이지를 페치하는 I/O 비용을 고려합니다. 계산 과정을 통해 최적화 알고리즘은 조회를 충족시키는 데 필요한 I/O 횟수를 추정합니다. 이미 버퍼 풀에 들어 있는 페이지의 행을 읽는 데에는 추가 물리적 I/O가 필요하지 않기 때문에, 이러한 예측에는 버퍼 풀 사용에 대한 예상치도 포함됩니다. 최적화 알고리즘은 버퍼 풀에 페이지가 있는지를 추정할 때 BUFFERPOOLS 시스템 카탈로그 테이블의 *npages* 컬럼 값을 고려합니다.

  테이블을 읽는 데 드는 I/O 비용은 다음과 같은 요소에 영향을 미칠 수 있습니다.

  – 202 페이지의 『외부 및 내부 테이블 결정』에 설명된 대로 두 개의 테이블을 조인하는 방법
  – 데이터를 읽는 데 클러스터되지 않은 색인이 사용될지의 여부(192 페이지의 『클러스터된 색인』 참조)

  한 데이터베이스에 둘 이상의 버퍼 풀이 있을 수 있습니다. 또한 한 파티션된 데이터베이스에 둘 이상의 버퍼 풀이 있을 수도 있습니다. 데이터베이스의 각 파티션에 선택적으로 새 버퍼 풀을 추가하거나, 모든 파티션에 버퍼 풀을 추가할 수 있습니다. BUFFERPOOLS 및 BUFFERPOOLSNODE 시스템 카탈로그 테이블의 *npages* 컬럼은 파티션된 데이터베이스의 추정을 위해 최적화 알고리즘을 사용합니다.

- 514 페이지의 『기본 등급(dft_degree)』.

  dft_degree 구성 매개변수는 CURRENT DEGREE 특수 레지스터 및 DEGREE 바인드 옵션에 대한 기본값을 지정합니다. 1 값은 파티션 내 병렬 처리가 없음

을 의미합니다. -1 값은 최적화 알고리즘이 프로세서 수 및 조회 유형에 기초하여 파티션 내 병렬 처리 수준을 결정한다는 의미입니다.

- 515 페이지의 『기본 조회 최적화 클래스(dft_queryopt)』.

  SQL 조회를 컴파일할 때에는 조회 최적화 클래스를 사용하여 다양한 수준의 최적화 알고리즘을 사용할 수 있습니다. 적당한 조회 최적화 클래스 선택에 대한 자세한 내용은 74 페이지의 『최적화 클래스 조정』을 참조하십시오.

- 458 페이지의 『사용 중인 응용프로그램의 평균 수(avg_appls)』.

  avg_appls 매개변수는 선택된 액세스 플랜에 대하여 런타임 시 얼마나 많은 버퍼 풀이 사용 가능한지를 SQL 최적화 알고리즘이 예측하는 데 도움을 줍니다. 이 매개변수의 값이 높을수록 최적화 알고리즘이 버퍼 풀 사용에 더 보수적인 조회의 액세스 플랜을 선택하는 데 영향을 미칠 수 있습니다. 이 매개변수에 1의 값을 넣으면 최적화 알고리즘은 버퍼 풀 전체를 응용프로그램에 사용 가능한 것으로 처리하게 됩니다.

- 414 페이지의 『정렬 힙 *크기*(sortheap)』.

  정렬시 최종 정렬된 데이터 목록을 저장하기 위해 임시 테이블이 필요하지 않은 경우, 정렬은 『파이프 정렬(piped sort)』로 간주됩니다. 즉, 정렬의 결과가 단일, 순차 액세스로 읽습니다. 파이프 정렬(piped sort)은 비 파이프 정렬보다 성능에서 더 좋은 결과를 보여주므로 가능하면 사용됩니다. (파이프 정렬과 비교하여 비 파이프 정렬 정의에 대한 자세한 내용은 217 페이지의 『정렬이 최적화 알고리즘에 미치는 영향』을 참조하십시오.)

  액세스 플랜을 선택할 때 최적화 알고리즘은 정렬이 아래와 같은 방법으로 파이프되는지 여부를 평가하는 것을 포함하여 정렬 조작에 드는 비용을 계산합니다.

  – 정렬될 데이터의 양을 예측
  – *sortheap* 매개변수를 보고 파이프 정렬에 사용될 공간이 충분한지를 결정합니다.

- 자세한 내용은 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』 및 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』을 참조하십시오.

  사용 중인 분리 레벨(45 페이지의 『동시성』 참조)이 **반복 읽기(RR)**인 경우, SQL 최적화 알고리즘은 *locklist* 및 *maxlocks* 매개변수 값을 고려하여 행 레벨 잠금

이 테이블 잠금 레벨로 레벨 자동 업그레이드될 것인지를 결정합니다. 최적화 알고리즘이 테이블 액세스에 대한 잠금 레벨 자동 업그레이드가 발생하리라고 예측하는 경우, 최적화 알고리즘은 조회 실행 중에 잠금 레벨 자동 업그레이드에 대한 오버헤드를 발생시키기보다는 액세스 플랜에 대한 테이블 레벨 잠금을 선택합니다.

- 552 페이지의 『CPU 속도(cpuspeed)』.

  특정 조작을 수행하는 데 드는 비용을 계산하기 위해 SQL 최적화 알고리즘에서 CPU 속도가 사용됩니다. 최적화 알고리즘은 다양한 I/O 비용 계산과 더불어 이러한 CPU 비용 계산을 사용하여 조회를 위한 최상의 액세스 플랜을 선택합니다.

  머신의 CPU 속도는 선택된 액세스 플랜에 큰 영향을 미칠 수 있습니다. 이 구성 매개변수는 데이터베이스가 설치 또는 이주될 때 자동으로 적정 값으로 설정됩니다. 테스트 시스템에 실제 환경을 모델링하거나 하드웨어상의 변경사항이 가져올 영향을 평가하는 경우에만 이 매개변수를 조정해야 합니다. 다른 하드웨어 환경을 모델링하기 위해 이 매개변수를 사용하면 해당 환경에 대해 선택될 액세스 플랜을 검토해 볼 수 있습니다.

- 417 페이지의 『명령문 힙 *크기*(stmtheap)』.

  명령문 힙(heap)의 *크기*는 최적화 알고리즘이 서로 다른 액세스 경로를 선택하는 데 영향을 미치지 않습니다. 그러나 복잡한 SQL문에 수행하게 될 최적화의 수준에는 영향을 미칠 수 있습니다.

  stmtheap 매개변수가 충분히 큰 값으로 설정되지 않으면 명령문 처리에 사용할 수 있는 메모리가 충분하지 않음을 나타내는 SQL 경고를 수신하게 됩니다. 예를 들어, SQLCODE +437(SQLSTATE 01602)은 명령문을 컴파일하는 데 사용된 최적화의 양이 조회 최적화 클래스를 지정할 때 요청한 양보다 작음을 나타냅니다. 자세한 내용은 74 페이지의 『최적화 클래스 조정』을 참조하십시오.

- 543 페이지의 『병렬의 처리 최대 조회 수준(max_querydegree)』.

  이 매개변수가 "ANY" 값을 가지는 경우, 최적화 알고리즘은 사용될 병렬 처리 수준을 선택합니다. "ANY"를 제외한 값이 나타나는 경우, 사용자 지정 값을 사용하여 응용프로그램에서 병렬 처리 수준을 결정합니다.

- 551 페이지의 『통신 대역폭(comm_bandwidth)』.

최적화 알고리즘이 통신 대역폭을 사용하여 액세스 경로를 결정합니다. 최적화 알고리즘은 이 매개변수의 값을 사용하여 파티션된 데이터베이스의 데이터베이스 파티션 서버간의 조작을 수행하는 비용을 계산합니다.

자세한 내용은 380 페이지의 『구성 매개변수 조정』을 참조하십시오.

조회 최적화에 미치는 노드 그룹의 영향

파티션된 데이터베이스에서 테이블 조합은 최적화 알고리즘에 의해 인식되고 조회에 대한 최상의 액세스 플랜을 결정할 경우에 사용됩니다. 조인 조회와 자주 연관되는 테이블은 파티션된 데이터베이스의 파티션간에 분리된 경우 조인되는 각 테이블로부터 같은 데이터베이스 파티션에 위치한 행을 가져야 한다고 가정합니다. 조인 조작 동안, 조인의 일부인 두 테이블로부터의 데이터 조합으로 파티션간에 데이터를 이동시킬 수 없습니다. 두 테이블을 동일한 노드 그룹에 배치할 경우, 테이블에서 데이터가 조합됩니다.

테이블 조합에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

또한 파티션된 데이터베이스 내에서, 그리고 테이블의 *크기*에 따라 더 많은 파티션으로의 데이터 확장은 조회를 실행하는 시간(또는 비용)을 줄입니다. 테이블 수, 테이블 *크기*, 테이블에서 데이터의 위치, 조회 유형(조회가 위의 설명처럼 필요한 경우)은 모두 조회 비용에 영향을 미칩니다.

조회 최적화에 미치는 테이블 공간의 영향

테이블 공간의 특정한 특성이 SQL 컴파일러에 의해 선택된 액세스 플랜에 영향을 미칠 수 있습니다.

- 컨테이너 특성

  컨테이너 특성은 조회를 실행할 때 드는 I/O 비용에 큰 영향을 미칠 수 있습니다. 액세스 플랜을 선택할 때 SQL 최적화 알고리즘은 서로 다른 테이블 공간에서 데이터에 액세스하는 데 드는 비용의 차이를 포함한 I/O 비용을 고려합니다. 최적화 알고리즘은 SYSCAT.TABLESPACES 시스템 카탈로그의 두 컬럼을 사용하여 테이블 공간에 있는 데이터에 액세스하는 데 드는 I/O 비용을 계산합니다.

– OVERHEAD 이는 데이터를 메모리로 읽어들이기 전에 컨테이너에 필요한 예상 시간(밀리초 단위)을 제공합니다. 이 오버헤드 활동에는 디스크 탐색 시간(seek time)을 비롯한 디스크 대기 시간(latency time), 컨테이너의 I/O 제어기 오버헤드가 포함됩니다.

  다음 공식을 사용하여 오버헤드 비용을 계산할 수 있습니다.

  ```
  OVERHEAD = 밀리초 단위의 평균 탐색 시간
             + (0.5 * 회전 대기 시간)
  ```

  여기서

  - 0.5는 반 회전의 평균 오버헤드를 나타냅니다.
  - 회전 대기 시간은 다음과 같은 방법으로 한 번 회전하는 데 걸리는 시간을 계산합니다(밀리초 단위).

    ```
    (1 / RPM) * 60 * 1000
    ```

    여기서

    - 분당 회전 수로 나누어서 한 번 회전하는 데 걸리는 시간을 구합니다.
    - 분당 60초를 곱합니다.
    - 초당 1000밀리초를 곱합니다.

  예를 들어, 디스크가 1분에 7 200회 회전한다고 가정하십시오. 그러면 회전 대기 공식을 사용하여 다음과 같은 식을 세울 수 있습니다.

  ```
  (1 / 7200) * 60 * 1000 = 8.328밀리초
  ```

  위의 식은 가정된 평균 탐색 시간인 11밀리초를 사용하여 OVERHEAD 추정치 계산에 사용될 수 있습니다.

  ```
  OVERHEAD = 11 + (0.5 * 8.328)
           = 15.164
  ```

  계산 결과, OVERHEAD 값은 약 15밀리초로 예상됩니다.

– TRANSFERRATE. 이것은 한 페이지의 데이터를 메모리로 읽어들이는 데 필요한 시간의 추정치(밀리초 단위)를 제공합니다.

  각 테이블 공간 컨테이너가 단일 물리적 디스크이면 다음 공식을 사용하여 전송 비용(페이지당 밀리초)을 계산할 수 있습니다.

  ```
  TRANSFERRATE = (1 / spec_rate) * 1000 / 1 024 000 * page_size
  ```

여기서

- spec_rate는 초당 MB 단위의 전송률에 대한 디스크 스펙을 나타냅니다.
- spec_rate로 나누어 MB당 초 수를 구합니다.
- 초당 1000밀리초를 곱합니다.
- MB당 1 024 000바이트로 나눕니다.
- 페이지 크기(바이트 단위)를 곱합니다(예를 들어, 4KB 페이지의 경우, 4 096바이트임).

예를 들어, 디스크에 대한 스펙 비율이 초당 3MB라고 가정하십시오. 다음과 같은 식을 세울 수 있습니다.

```
TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
             = 1.333248
```

계산된 TRANSFERRATE 값은 약 페이지당 1.3밀리초입니다.

테이블 공간 컨테이너가 단일 물리적 디스크가 아니고 디스크 배열(예: RAID)인 경우, 사용할 TRANSFERRATE를 결정할 때 추가로 고려해야 할 사항이 있습니다. 배열이 상대적으로 작은 경우, 병목 현상이 디스크 레벨에 있는 것으로 가정하고 spec_rate에 디스크 수를 곱할 수 있습니다. 그러나 컨테이너를 구성하는 배열 내의 디스크 수가 많으면 병목 현상은 디스크 레벨이 아니고 다른 I/O 서브시스템 구성요소(예: 디스크 제어기, I/O 버스 또는 시스템 버스) 중 하나의 레벨에 있을 수 있습니다. 이런 경우에는 I/O 처리량 성능이 spec_rate에 디스크 수를 곱한 것이라고 할 수 없습니다. 대신, 순차 스캔 중에 실제 I/O 비율(MB 단위)을 측정해야 합니다. 예를 들어, 순차 스캔은 select count(*) from big_table일 수 있으며 크기는 MB 단위입니다. 결과를 big_table이 상주하는 테이블 공간을 구성하는 컨테이너 수로 나누십시오. 위의 공식에 있는 spec_rate 대신 이 결과를 사용하십시오. 예를 들어, 네 개의 컨테이너 테이블 공간에 있는 테이블을 스캔하는 중에 측정된 순차 I/O 비율(100MB)은 컨테이너당 25MB 또는 TRANSFERRATE가 페이지당 (1/25) * 1000 / 1024000 * 4096 = 0.16 밀리초임을 의미할 수 있습니다.

테이블 공간에 지정된 각각의 컨테이너는 서로 다른 물리적 디스크에 상주할 수도 있습니다. 최상의 결과를 얻기 위해 주어진 테이블 공간에 사용된 모든 물리

적 디스크에는 같은 OVERHEAD 및 TRANSFERRATE 특성이 있어야 합니다. 이들 특성이 같지 않을 경우, OVERHEAD 및 TRANSFERRATE 값을 설정할 때에는 평균값을 사용해야 합니다.

이 컬럼에 대한 미디어 특정 값은 하드웨어 스펙이나 경험을 통해 구할 수 있습니다. 이들 값은 CREATE TABLESPACE 및 ALTER TABLESPACE문에서 지정할 수도 있습니다.

위에서 언급한 것처럼 컨테이너로서 디스크 배열을 사용하는 환경에서는 경험이 특히 중요할 수 있습니다. 데이터를 이동시키고 이를 플랫폼 특정 측정 유틸리티와 함께 사용하는 간단한 조회를 작성해야 합니다. 그런 다음 테이블 공간 내의 다른 컨테이너 구성에서 조회를 다시 수행할 수 있습니다. CREATE 및 ALTER TABLESPACE문을 사용하여 환경 내에서 데이터를 전송하는 방법을 변경할 수 있습니다.

이 두 값을 통한 I/O 비용 정보는 여러 방면(데이터에 액세스하는 데 색인을 사용할지 여부, 조인에서 내부 및 외부 테이블에 대해 어떤 테이블을 선택할지와 같은)에서 최적화 알고리즘에 영향을 미칠 수 있습니다.

- 프리페치

  테이블 공간에서 데이터에 액세스하는 I/O 비용을 고려할 때 최적화 알고리즘은 디스크에서 데이터 및 색인 페이지를 프리페치하는 것이 조회의 성능에 미칠 잠재적 영향도 고려합니다. 데이터 및 색인 페이지를 프리페치하면 버퍼 풀로 데이터를 읽어들일 때 연관된 오버헤드와 대기 시간을 줄일 수 있습니다. 자세한 내용은 292 페이지의 『버퍼 풀로 데이터 프리페치』를 참조하십시오.

  최적화 알고리즘은 SYSCAT.TABLESPACES의 PREFETCHSIZE 및 EXTENTSIZE 컬럼의 정보를 사용하여 테이블 공간에 대해 발생할 프리페치의 양을 예측합니다.

  – EXTENTSIZE는 (CREATE TABLESPACE문을 사용하여) 테이블 공간을 작성할 때에만 설정될 수 있습니다. 기본 확장 *크기*는 32페이지(각 4KB)로 충분한 *크기*입니다.

  – PREFETCHSIZE는 테이블 공간을 작성할 때와 ALTER TABLESPACE 문을 사용할 때 설정될 수 있습니다. 기본 프리페치 *크기*는 운영 체제에 따라 다른 DFT_PREFETCH_SZ 데이터베이스 구성 매개변수 값에 의해 결정됩니다. 이 매개변수의 *크기*를 지정하려면 452 페이지의 『기본 프리페치 *크기*

(dft_prefetch_sz)』에 설명되어 있는 권장사항을 검토하고, 필요에 따라 변경하여 데이터의 이동 성능을 향상시키십시오.

다음은 RESOURCE 테이블 공간의 특성을 변경하기 위한 구문의 예입니다.

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

테이블 공간에 임의의 변경을 한 후에는 응용프로그램을 리바인드하고, RUNSTATS 유틸리티를 통해 색인에 대한 최종 통계를 수집하여 최상의 액세스 플랜이 사용되도록 하십시오.

색인화가 조회 최적화에 미치는 영향

사용자는 색인이 사용되는 시기를 결정하지 않는다는 점을 기억하십시오. 최적화 알고리즘에서 사용 가능한 테이블 및 색인 정보에 근거하여 이를 결정합니다. 그러나 사용자가 성능을 향상시킬 수 있는 필요한 색인을 작성하여 프로세스에서 중요한 역할을 합니다. 색인을 작성하거나 프리페치 *크기*를 변경한 후에는 색인에 대한 통계를 수집하고(RUNSTATS 유틸리티 사용) 현재 기반에서 통계를 최신 정보로 유지하는 것이 중요합니다. 즉, 이는 사용자가 작성할 수 있는 색인의 종류 및 색인을 작성하는 방법을 이해하고 있어야 한다는 의미입니다.

색인화 및 비색인화

데이터베이스 조회에서 참조된 각 테이블은 테이블에 색인이 없는 경우, 테이블에서 테이블 스캔이 수행되어야 합니다. 테이블이 클수록 테이블 스캔도 더 오래 걸립니다. *테이블 스캔*은 데이터베이스 관리 프로그램이 테이블의 각 행에 순차적으로 액세스할 때 발생합니다. 이는 데이터베이스 관리 프로그램이 색인을 사용하여 데이터에 액세스할 때 발생하는 *색인 스캔*과 비교할 수 있습니다. (자세한 내용은 184 페이지의 『색인 스캔 개념』을 참조하십시오.)

SELECT문에서 색인 컬럼이 참조되고 테이블 스캔보다 색인 스캔이 더 빠른 방법이라고 최적화 알고리즘이 판단한 경우, 색인이 선택됩니다. 색인 파일은 일반적으로 더 작고, 특히 테이블의 규모가 커질수록 전체 테이블을 읽는 데 시간이 덜

걸립니다. 또한 색인 전체가 스캔될 필요도 없습니다. 색인에 적용되는 술어는 데이터 페이지로부터 읽어야 하는 행 수를 줄여줍니다.

각각의 색인 항목은 검색 키 값과, 해당 값을 가지고 있는 행을 가리키는 포인터로 구성되어 있습니다. 값은 ALLOW REVERSE SCANS 매개변수가 CREATE INDEX문에 지정된 경우에만 역방향으로 검색될 수 있습니다. 따라서 오른쪽 술어가 있을 경우, 검색을 제한할 수 있습니다. 색인을 사용하여 행을 테이블에서 읽은 후 데이터베이스 관리 프로그램이 행을 정렬할 필요가 없이, 순서화된 순서로 행을 확보할 수 있습니다. ALLOW REVERSE SCANS를 지정하면 색인을 사용하여 정방향 및 역방향으로 순서대로 직접 행을 확보할 수 있습니다. 자세한 내용은 *SQL 참조서*를 참조하십시오.

고유 색인에는 검색 키 값 및 행 포인터와 함께 컬럼이 포함될 수 있습니다.

주: 색인이 최적화 알고리즘에 의해 선택되거나, 데이터베이스 관리 프로그램에 의해 사용되는지 여부를 제어할 수 없습니다. 예를 들어, 조회의 결과가 조회 중인 테이블에 존재하는 색인에 의해 순서화된 순서에서 생성되는 것이 보장될 수 없습니다. 조회 처리 중에 데이터베이스 관리 프로그램은 최적화 알고리즘으로 색인을 선택하고 이 색인을 사용하지만, 반드시 그렇게 할 필요는 없습니다. ORDER BY절만 있으면 결과 세트의 순서를 『보장할』 수 있습니다.

색인은 액세스 시간을 크게 줄일 수 있지만, 성능에 좋지 않은 영향을 줄 수도 있습니다. 색인을 작성하기 전에 디스크 공간과 처리 시간에 미치는 다중 색인의 영향을 고려하십시오.

- 색인은 모두 일정량의 저장영역 또는 디스크 공간을 차지하고 있습니다. 정확한 양은 테이블의 크기 및 색인에 포함된 컬럼의 크기와 수에 따라 다릅니다.
- 테이블에 대하여 INSERT 또는 DELETE 조작을 수행하여 해당 테이블에 대한 각각의 색인을 추가로 갱신해야 합니다. 이는 색인 키를 변경시킨 모든 UPDATE 조작의 경우에도 해당됩니다.
- LOAD 유틸리티는 색인을 재빌드하거나 기존의 색인에 첨부합니다.
- indexfreespace MODIFIED BY 매개변수를 LOAD 명령에 지정하여 색인이 작성될 때 사용된 PCTFREE 색인을 겹쳐쓸 수 있습니다.
- 각 색인은 잠재적으로 조회에 대하여 대체 액세스 경로를 추가하는데, 최적화 알고리즘이 이를 고려하기 때문에 조회 컴파일 시간이 증가하게 됩니다.

응용프로그램의 필요를 충족시키려면 색인은 신중하게 선택해야 합니다.

특정 패키지에서 색인이 사용되는지의 여부를 결정하기 위해 243 페이지의 『제7장 SQL Explain 기능』에서 설명된 SQL Explain 기능을 사용할 수도 있습니다.

색인 보조 프로그램 사용

DB2 색인 보조 프로그램은 테이블 데이터에 대해 최적의 색인 세트를 선택할 수 있도록 도와주는 도구입니다. 이 도구를 다음과 같이 여러 가지 방법으로 확보할 수 있습니다.

- 제어 센터를 통해 색인 폴더를 선택하고 마우스 버튼 2를 누른 다음 **작성 —> 마법사를 사용한 색인**을 선택하여 이 도구에 액세스할 수 있습니다.
- `db2advis`를 입력하여 명령행에서 이 도구에 액세스할 수 있습니다.

DB2 색인 권장 도구에 대한 자세한 내용은 265 페이지의 『SQL 조언 기능』을 참조하십시오.

보다 큰 색인 키 사용

255바이트보다 긴 컬럼을 색인 키의 부분으로서 지정할 수 있습니다. DB2_INDEX_2BYTEVARLEN 레지스트리 변수는 색인 키의 길이가 1로 저장되는 대신에 2바이트로 사용되는 것을 허용합니다.

몇 가지 SQL문은 레지스트리 변수 변경사항에 의해 영향을 받습니다. 다음과 같습니다.

- CREATE TABLE. 기본, 외래 및 변수 키 파트가 있는 고유 키는 *크기*가 255바이트보다 클 수 있습니다.
- CREATE INDEX. 고유 색인 및 포함 컬럼을 포함하여 변수 키 파트를 가지는 모든 색인은 *크기*가 255바이트보다 클 수 있습니다.
- ALTER TABLE. 기본, 외래 및 변수 키 파트가 있는 고유 키는 *크기*가 255바이트보다 클 수 있습니다. 고유 색인 및 포함 컬럼을 포함하여 변수 키 파트를 가지는 모든 색인은 *크기*가 255바이트보다 클 수 있습니다.

  255바이트로의 외부 키 제한은 레지스트리 변수의 값에 상관없이 제거됩니다. 외부 키에 상응하는 기본 키의 참 조건은 어떤 제한 또는 한계도 강제합니다.

더 큰 색인 키를 사용하기 위해 기존 색인을 변환하려면 색인을 삭제하고 DB2_INDEX_2BYTEVARLEN 레지스트리 변수를 ON으로 설정한 다음 색인을 재작성합니다(더 큰 컬럼 사용).

구문 설명을 포함한 SQL문에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

색인화에 대한 지침

어떤 색인을 작성할 것인지는 데이터와 사용 목적에 따라 다릅니다. 다음 지침은 어떤 색인이 가장 유용한지를 판단하는 데 도움이 될 수 있습니다.

- 적용되는 곳마다 CREATE UNIQUE INDEX문을 사용하여 기본 키와 고유 키를 정의하십시오. (자세한 내용은 *SQL 참조서*를 참조하십시오.) 고유 색인은 최적화 알고리즘이 정렬과 같은 조작을 수행하는 것을 피하는 데 도움이 될 수 있습니다.
- 데이터 검색의 성능을 향상시키려면 INCLUDE 컬럼이 있는 고유 색인을 정의하십시오. 다음과 같은 경우에 컬럼이 고유 색인의 INCLUDE 컬럼으로서 적합합니다.
  - 자주 액세스되기 대문에 색인 전용 액세스를 통해 이점을 얻을 수 있는 경우
  - 색인 스캔 범위를 제한할 필요가 없는 경우
  - 색인 키의 순서나 고유성에 영향을 받지 않는 경우

  INCLUDE 컬럼에 대한 자세한 내용은 *관리 안내서: 계획*의 『색인 또는 색인 스펙 작성』 장을 참조하십시오.
- SYSCAT.TABLES 카탈로그 뷰의 NPAGES 컬럼에 기록된 대로 몇 개 이상의 데이터 페이지를 가진 테이블에 대한 빈번한 조회를 최적화하려면 색인을 사용하십시오. 다음을 수행하십시오.
  - 테이블을 조인할 때 사용할 컬럼에 대하여 색인을 작성하십시오.
  - 정기적으로 특정 값에 대한 검색을 하게 될 컬럼에 대하여 색인을 작성하십시오.
- 어떤 순서가 기본적으로 사용되는지 또는 요청되었는지에 따라 키의 순서(오름차순 또는 내림차순)를 결정하십시오. 값은 ALLOW REVERSE SCANS 매개변수가 CREATE INDEX문에 지정된 경우에만 역방향으로 검색될 수 있습

니다. 색인은 양방향으로 스캔될 수는 있지만, 정방향 스캔(즉, 색인이 작성될 때 지정된 순서)이 역방향 스캔보다 약간 빠릅니다. 자세한 내용은 *SQL 참조서*를 참조하십시오.

- 컬럼에 대한 다른 색인 키의 부분 키가 되는 색인은 작성하지 않도록 하십시오. 예를 들어, 컬럼 a, b, c에 대한 색인이 있는 경우, a와 b에 대한 두 번째 색인은 꼭 유용한 것은 아닙니다.
- 상위 테이블에 대한 삭제 및 갱신 조작의 성능을 향상시키려면 외부 키에 대한 색인을 사용하십시오.
- 데이터를 정렬하는 데 자주 사용될 컬럼에 대한 색인을 사용하십시오.
- 다중 컬럼 색인을 작성할 때 첫 번째 키 컬럼에 대해 둘 이상의 선택사항이 있을 경우, 『=』(equijoin) 술어와 함께 가장 자주 지정되는 것을 선택하거나 구별값이 가장 큰 컬럼을 먼저 지정하십시오.
- 임의로 모든 컬럼에 대한 색인을 작성하면 디스크 공간이 많이 소모되고 준비 시간도 길어지게 됩니다. 특히 동적 프로그래밍 조인 열거가 있는 최적화 클래스가 사용되는 복합 조회의 경우에 해당됩니다(74 페이지의 『최적화 클래스 조정』 참조).
- 다음은 하나의 테이블에 정의하는 *대표적인* 색인의 수입니다. 이 숫자는 데이터베이스의 기본적인 사용을 근거로 한 것입니다.
  - 온라인 트랜잭션 처리(OLTP) 환경에서는 하나 또는 두 개의 색인만을 가져야 합니다.
  - 조회(읽기 전용) 환경에서는 6개 이상의 색인을 가질 수 있습니다.
  - 조회와 OLTP가 혼합된 환경에서는 2 - 5개까지의 색인을 가질 수 있습니다.
- 새로 삽입된 행이 해당 색인에 따라 클러스터링되도록 하려면 클러스터링 색인 정의를 고려해 보십시오. 클러스터링 색인은 테이블을 재구성할 필요성을 크게 줄입니다.

  주: 클러스터링 색인이 정의되면 각 데이터 페이지에 예약된 사용 가능 공간으로 테이블이 로드되어 해당 페이지에 삽입이 발행할 수 있도록 해야 합니다. (PCTFREE 키워드를 ALTER TABLE문에 사용하거나 LOAD 명령의 pagefreespace MODIFIED BY절을 사용하여 사용 가능 공간이 예약됩니다.)

- 색인을 작성할 때에는 PCTFREE 키워드 사용을 고려해 보십시오. PCTFREE는 향후의 색인 갱신을 위해 색인 페이지에 공간을 예약합니다. 그러면 페이지 분할 빈도가 줄어들고 성능이 향상됩니다.
- 색인을 작성할 때에는 MINPCTUSED 옵션 사용을 고려해 보십시오. MINPCTUSED는 색인 리프 페이지에서 사용되는 최소 공간에 대한 임계값을 지정하여 온라인 색인 재구성을 가능하게 합니다. 이를 사용하면 데이터 및 색인을 오프라인으로 재구성해야 할 필요가 줄어듭니다.

주: 색인은 선언된 임시 테이블에 대해서는 지원하지 않습니다.

다음은 색인을 작성하여 성능을 향상시킬 수 있는 대표적인 경우입니다.

- 색인이 조회의 WHERE절에서 사용된 컬럼과 가장 빈번하게 처리되는 트랜잭션에 대하여 작성될 수 있습니다.

  WHERE절

  ```
  WHERE WORKDEPT='A01' OR WORKDEPT='E21'
  ```

  은 해당 값이 자주 발생하지 않으면 WORKDEPT에 대한 색인에서 보통 이점이 있습니다.
- 조합 순서로 되어 있는 행을 순서지정하기 위해 하나 이상의 컬럼에 대하여 색인을 작성할 수 있습니다. 순서지정은 ORDER BY절뿐만이 아니라, DISTINCT 및 GROUP BY절과 같은 다른 기능에 의해서도 요구될 수 있습니다.

  다음 예는 DISTINCT절을 사용한 것입니다.

  ```
  SELECT DISTINCT WORKDEPT
       FROM   EMPLOYEE
  ```

  데이터베이스 관리 프로그램은 중복된 값을 제거하기 위해 WORKDEPT에 대해 오름차순 또는 내림차순으로 정의된 색인을 사용할 수도 있습니다. 이 동일한 색인이 다음 예에서처럼 GROUP BY절로 값을 그룹화하는 데에도 사용될 수 있습니다.

  ```
  SELECT WORKDEPT, AVERAGE(SALARY)
       FROM   EMPLOYEE
  GROUP BY WORKDEPT
  ```

- 색인이 명령문에서 참조된 각각의 컬럼의 이름을 지정하기 위해 작성될 수 있습니다. 색인이 이 방법으로 지정된 경우에는 테이블 액세스를 피함으로써 결과 색인 전용 액세스 방식 데이터가 더욱 효율적으로 검색될 수 있습니다.

  예를 들어, 다음의 SQL문이 발행된다고 가정하십시오.

  ```
  SELECT LASTNAME
       FROM   EMPLOYEE
   WHERE WORKDEPT IN ('A00','D11','D21')
  ```

  색인이 EMPLOYEE 테이블의 LASTNAME 및 WORKDEPT 컬럼에 대하여 정의되어 있는 경우, 명령문은 전체 테이블을 스캔하는 대신 색인을 스캔하여 더욱 효율적으로 처리될 수 있습니다. 술어가 WORKDEPT에 대한 것이므로, 이 컬럼이 색인의 첫 번째 컬럼이 되어야 함에 유의하십시오.

- 색인의 INCLUDE 컬럼은 테이블의 색인 사용을 향상시키는 또다른 방법입니다. 이전 예를 사용하여 다음과 같이 고유 색인을 정의할 수 있습니다.

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  lastname을 색인 키의 일부가 아닌 INCLUDE 컬럼으로 지정하는 것은 lastname은 색인의 리프 페이지에만 저장된다는 의미입니다.

색인 관리시 성능에 관련된 추가 정보

다음의 내용은 색인을 적절하게 사용하고 관리하는 것이 어떻게 성능에 영향을 미치는지를 이해하는 데 도움이 됩니다.

1. **색인 작성**

   대형 테이블에서 색인을 작성하고 SMP 머신을 가진 경우, intra_parallel을 YES(1) 또는 SYSTEM(-1)으로 설정하여 병렬 성능 향상을 이용할 수 있습니다.

   다중 프로세서는 데이터를 스캔하고 정렬하는 데 사용할 수 있습니다. 색인 작성 중 다중 프로세서가 도움이 되지 않는 경우에는 *indexsort* 데이터베이스 구성 매개변수가 NO로 설정될 경우뿐입니다. (이 매개변수의 기본값은 YES입니다.) 이 매개변수는 색인 작성 중 색인 키 정렬이 수행하는지를 제어합니다.

2. **색인 테이블 공간**

   색인은 다른 테이블 데이터를 저장하는 데 사용하는 테이블 공간과는 다른 테이블 공간에 저장되어 있을 수도 있습니다. 이는 읽기/쓰기 헤드의 움직임을 줄

임으로써 디스크 저장을 더욱 효율적으로 사용할 수 있도록 해줍니다. 또한 사용자의 색인 테이블 공간을 작성하여 더 빠른 물리적 장치에 이들이 저장될 수도 있습니다.

색인 페이지가 많은 데이터 페이지로 인해 버퍼 밖으로 밀려나지 않게 보호하는 별도의 버퍼 풀에 테이블 공간을 지정할 수 있습니다.

색인이 별도의 테이블 공간에 놓이지 않으면 데이터와 색인 페이지가 모두 같은 Extent *크기*와 프리페치량을 사용합니다. 색인에 대해 다른 테이블 공간을 사용할 경우, 테이블 공간의 모든 특성에 대해 다른 값을 선택하는 옵션을 가지게 됩니다. 일반적으로 색인이 테이블보다 작고 여러 컨테이너에 걸쳐 있는 경우, 8과 16 같은 더 작은 Extent *크기*를 찾는 것이 보통입니다. 자세한 내용은 194 페이지의 『색인 페이지 프리페치』를 참조하십시오. 테이블 공간에 대하여 더 빠른 장치를 사용하는 방법이 105 페이지의 『조회 최적화에 미치는 테이블 공간의 영향』에 설명된 대로 SQL 최적화 알고리즘에 의해 고려됩니다. 테이블 공간에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

3. **클러스터링 등급**

   SQL문이 순서화를 요구하고(예: ORDER BY, GROUP BY, DISTINCT) 이를 충족시킬 수 있는 적절한 색인이 있는 경우, 데이터베이스 관리 프로그램이 색인을 선택하지 않을 수도 있습니다. 이는 다음과 같은 경우에 발생할 수 있습니다.

   - 색인 클러스터링이 제대로 되어 있지 않습니다(SYSCAT.INDEXES의 CLUSTERRATIO 및 CLUSTERFACTOR 컬럼 참조).
   - 테이블이 작아서 테이블을 스캔하고 메모리에서 응답 세트를 정렬하는 데 비용이 더 적게 듭니다.
   - 테이블 액세스에 대해 경쟁하는 색인이 있습니다.

   클러스터링 색인을 작성한 후에 REORG 또는 정렬 및 LOAD를 수행하는 것이 좋습니다. 일반적으로, 테이블은 하나의 색인에 대해서만 클러스터링될 수 있습니다. 테이블과 색인은 해당 테이블의 클러스터링 색인의 순서에 따라 빌드되어야 합니다. 클러스터링 색인은 RUNSTATS 유틸리티가 수집하는 CLUSTERRATIO 또는 CLUSTERFACTOR 통계를 개선하여 특정한 순서의 데이터를 유지보수하려고 합니다.

또한 해당 테이블을 로드하거나 재구성하기 전에 테이블을 변경하는 경우 PCTFREE 사용을 고려해 보십시오. 클러스터링을 유지보수하려면 각 테이블에는 각 데이터 페이지에 대해 사용할 수 있는 추가 삽입용 공간이 있어야 합니다. 공간을 사용할 수 있는 경우, 기존 데이터를 사용하여 추가 삽입이 클러스터링될 수 있습니다. 그 결과, 추가 데이터의 클러스터링을 위해 각 페이지에 일정 비율의 여유 공간을 남겨둔 후 데이터를 테이블에 로드하는 것을 고려할 수 있습니다. 우선 테이블을 작성한 다음 PCTFREE 매개변수를 사용하여 테이블을 변경하여 이 작업을 수행할 수 있습니다. 마찬가지로, 데이터를 재구성하기 전에 PCTFREE 매개변수를 사용하여 테이블을 변경하도록 고려해야 합니다. 그렇지 않으면 PCTFREE가 설정되지 않은 경우에 재구성을 통해 모든 추가 공간이 제거됩니다.

클러스터링은 현재 갱신 동안에 유지보수되지 않습니다. 즉, 클러스터링 색인에 있는 키 값이 변경되는 레코드를 갱신하는 경우, 레코드는 클러스터링 순서를 유지보수하기 위해 새 페이지로 이동할 필요가 없습니다. UPDTAE를 사용하는 대신 클러스터링을 유지보수하려면 DELETE를 사용한 다음 INSERT를 사용하십시오.

4. **RUNSTATS 유틸리티**

   새 색인을 작성한 후 색인 통계를 수집하려면 RUNSTATS 유틸리티를 사용해야 합니다. 통계 정보는 최적화 알고리즘이 색인을 사용하여 액세스 성능을 향상시킬 수 있는지 여부를 결정할 수 있도록 해줍니다. 이 주제에 대한 자세한 내용은 127 페이지의 『RUNSTATS 유틸리티를 사용하여 통계 수집』을 참조하십시오.

5. **색인 재구성**

   색인을 통해 최상의 성능을 얻으려면 색인을 정기적으로 재구성하는 것을 고려해야 합니다. 테이블에 대한 갱신으로 색인 페이지 프리페치가 덜 효율적일 수 있습니다. 색인 페이지 프리페치의 효율성을 유지하려면 색인을 재구성해야 합니다.

   색인을 삭제한 다음 재작성하거나 REORG 유틸리티를 사용하여 색인을 재구성할 수 있습니다. 자세한 내용은 304 페이지의 『카탈로그 및 사용자 테이블 재구성』을 참조하십시오.

재구성을 자주 하지 않기 위해 색인 작성시 PCTFREE를 지정할 수 있습니다. 색인 작성시 PCTFREE 매개변수를 지정하면 해당 색인이 작성될 때 각 색인 리프 페이지에 빈 공간이 남게 됩니다. 그 결과, 색인과 관련된 향후 활동 중에 색인 페이지 분할의 가능성이 적은 색인에 레코드가 삽입될 수 있습니다. 색인 페이지 분할로 인해 색인 페이지가 연속되거나 순차적이지 않게 됩니다. 그러면 색인 페이지 프리페치를 수행할 능력이 떨어지게 됩니다. 색인에 대해 적절한 PCTFREE를 선택하면 색인을 재구성하는 빈도가 제거되거나 줄어듭니다.

주: 재구성 중에 색인이 재작성되는 경우, 색인을 작성할 때 지정한 PCTFREE가 사용됩니다.

색인을 삭제한 다음 재작성하면 대략 연속되거나 순차적인 새 페이지 세트를 얻게 됩니다. 이로써 색인 페이지 프리페치가 향상됩니다.

비용이 더 들더라도, REORG 유틸리티를 사용하면 데이터 페이지의 클러스터링을 확실히 보장할 수 있습니다. 이 클러스터링은 많은 데이터 페이지에 액세스하는 색인 스캔에 대해 많은 이점이 있습니다.

대칭 멀티프로세서(SMP) 환경에서 작업하는 경우, REORG 유틸리티는 *intra_parallel*이 YES나 ANY일 때 다중 프로세서를 사용합니다.

6. **EXPLAIN 사용**

   가장 자주 사용되는 조회에서 정기적으로 EXPLAIN을 수행하고 각 색인이 적어도 한 번은 사용되는지 점검하십시오. 색인이 어떠한 조회에서도 사용되지 않는 경우, 해당 색인의 삭제를 고려하십시오.

   또한 EXPLAIN을 사용하여 대형 테이블에서 테이블 스캔이 중첩된 루프 조인의 내부 테이블처럼 처리되는지 보십시오. Join 술어 컬럼의 색인은 누락되거나 Join 술어 적용시 효과가 없는 것으로 생각될 수 있음을 나타냅니다. 또는 Join 술어가 없을 수도 있습니다.

7. **소멸성 테이블**

   소멸성 테이블은 런타임 시 크기가 공백에서 대형까지 변할 수 있는 테이블의 내용과 같은 특성을 가진 테이블입니다. 이 유형의 테이블에 대한 액세스 플

랜을 생성하는 경우, 최적화 알고리즘은 색인 스캔보다는 테이블 스캔을 사용하여 내용(기본 행 수: cardinality)이 심하게 변하는 테이블에 (부적절하게) 액세스하려고 합니다.

ALTER TABLE...VOLATILE문을 사용하여 『소멸성』 테이블을 선언하면 최적화 알고리즘이 소멸성 테이블에서 색인 스캔을 사용할 수 있게 됩니다. 최적화 프로그램은 통계에 상관없이 다음 경우에 색인 스캔(테이블 스캔이 아닌)을 사용합니다.

- 참조된 모든 컬럼이 색인에 있는 경우
- 색인이 색인 스캔의 술어에 적용될 수 있는 경우

테이블이 입력된 테이블인 경우, ALTER TABLE...VOLATILE문은 입력된 테이블 계층 구조의 루트 테이블에서만 지원될 수 있습니다. 이 주제에 대한 자세한 내용은 *관리 안내서: 계획* 또는 *SQL 참조서*를 참조하십시오.

연합 데이터베이스 조회에 영향을 미치는 서버 옵션

연합 시스템은 DB2 DBMS(연합 데이터베이스)와 하나 이상의 데이터 소스로 구성됩니다. CREATE SERVER문을 발행하면 데이터 소스가 연합 데이터베이스에 대해 식별됩니다. 이 명령문을 발행하면 DB2 및 지정된 데이터 소스를 포함한 연합 시스템 조작의 기능을 정제 및 제어할 수 있는 서버 옵션을 제공할 수 있습니다. 서버 옵션은 나중에 ALTER SERVER문을 사용하여 변경할 수 있습니다. CREATE SERVER 및 ALTER SERVER문에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

주: 서버를 작성하고 서버 옵션을 지정하려면 우선 분산 Distributed Join 설치 옵션을 설치하고 데이터베이스 관리 프로그램 매개변수 *federated*를 YES로 설정해야 합니다.

서버 옵션 및 해당 값은 조회 분석 푸시다운, 전역 최적화 및 연합 데이터베이스 조작의 여러 기능을 이용합니다. 예를 들어, CREATE SERVER문에서 서버 옵션 값에 따라 특정 성능 통계를 지정할 수 있습니다. 즉, *cpu_ratio* 옵션을 데이터 소스 및 연합 서버 CPU의 상대 속도를 나타내는 값으로 설정할 수 있습니다. 그런 다음 *io_ratio* 옵션을 데이터 소스 및 연합 서버 I/O 장치의 상대 비율을 나타내는 값으로 설정할 수 있습니다. CREATE SERVER를 수행하면 이 데이터는

SYSCAT.SERVEROPTIONS 카탈로그 뷰에 추가되며, 최적화 알고리즘은 이를 데이터 소스에 대한 액세스 플랜을 개발하는 데 사용합니다. 통계가 변경되면(예를 들어, 데이터 소스 CPU가 업그레이드될 경우에 발생할 수 있음) ALTER SERVER문을 사용하여 변경된 내용으로 SYSCAT.SERVEROPTIONS를 갱신할 수 있습니다. 그러면 최적화 알고리즘은 갱신사항을 사용하여 데이터 소스에 대한 그 다음 액세스 플랜을 개발합니다.

표 8. 서버 옵션 및 설정값

옵션	유효한 설정값	기본 설정값
collating_sequence	코드 세트 및 국가 정보에 기반하여 데이터 소스가 연합 데이터베이스와 동일한 기본 조합 순서를 사용할지 여부를 지정합니다. 데이터 소스가 DB2의 조합 순서와 다른 조합 순서를 사용할 경우, DB2의 조합 순서에 따라 대부분의 조작은 데이터 소스에서 원격으로 평가될 수 없습니다. 예에서는 다른 조합 순서를 사용하여 데이터 소스에서 별칭 문자 컬럼에 대해 MAX 컬럼 함수를 실행합니다. MAX 함수는 원격 데이터 소스에서 평가될 경우 결과가 다를 수 있기 때문에, DB2는 총계 조작과 MAX 함수를 지역적으로 수행합니다. 조회에 등호가 들어 있으면 조합 순서가 다르더라도 조회의 해당 부분을 데이터 소스로 푸시다운할 수 있습니다('N'으로 설정). 예를 들어, 술어 C1 = 'A'는 데이터 소스로 푸시다운할 수 있습니다. 물론, 데이터 소스에서의 조합 순서가 대소문자를 구별하지 않을 경우에는 그러한 조회를 푸시다운할 수 없습니다. 데이터 소스가 대소문자를 구별하지 않을 경우, C1= 'A' 및 C1 = 'a'의 결과는 동일하며, 이는 대소문자 구별 환경(DB2)에서는 승인되지 않습니다. 관리자는 데이터 소스 조합 순서와 일치하는 특정 조합 순서를 사용하여 연합 데이터베이스를 작성할 수 있습니다. 이 접근 방법은 모든 데이터 소스가 동일한 조합 순서를 사용하거나 대부분 또는 모든 컬럼 함수가 동일한 조합 순서를 사용하는 데이터 소스에 대해 지정될 경우, 성능이 향상됩니다. 'Y' 데이터 소스의 조합 순서가 연합 데이터베이스의 조합 순서와 동일합니다. 'N' 데이터 소스의 조합 순서가 연합 데이터베이스의 조합 순서와 동일하지 않습니다. 'I' 데이터 소스의 조합 순서가 연합 데이터베이스의 조합 순서와 다르며, 대소문자를 구별하지 않습니다. (예를 들면, 'TOLLESON' 및 'TolLESon'이 동일하게 간주됩니다.)	'N'

표 8. 서버 옵션 및 설정값 (계속)

옵션	유효한 설정값	기본 설정값
comm_rate	연합 서버 및 연관된 데이터 소스 사이의 통신 비율을 지정합니다. 초당 메가바이트로 표시됩니다. 유효 값은 0보다 크고 2147483648보다 작습니다. 값은 12와 같이 정수로만 표시될 수 있습니다.	'2'
connectstring	OLE DB 제공업체에 연결하는 데 필요한 초기화 등록 정보를 지정합니다. 연결 문자열의 완전한 구문 및 의미는 *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998*에 있는 "Data Link API of the OLE DB Core Components"를 참조하십시오.	없음
cpu_ratio	연합 서버의 CPU 속도보다 데이터 소스의 CPU 속도가 얼마나 빠른지 또는 느린지를 나타냅니다. 유효 값은 0보다 크고 $1x10^{23}$보다 작습니다. 값은 123E10, 123 또는 1.21E4와 같이 유효한 이중 표기로 표시될 수 있습니다.	'1.0'
dbname	연합 서버가 액세스하도록 할 데이터 소스 데이터베이스의 이름. DB2 계열 데이터 소스에 필요하며, Oracle** 데이터 소스에는 적용되지 않습니다. Oracle 인스턴스에 하나의 데이터베이스만 있기 때문입니다. DB2의 경우, 이 값은 인스턴스 내의 특정 데이터베이스에 해당되며, OS/390용 DB2일 경우에는 데이터베이스 LOCATION 값에 해당됩니다.	없음
fold_id(이 테이블의 끝에 있는 주 1 및 4 참조)	연합 서버가 인증을 위해 데이터 소스로 전송한 사용자 ID에 적용됩니다. 유효 값은 다음과 같습니다. 'U' 연합 서버는 사용자 ID를 데이터 소스로 전송하기 전에 대문자로 변경합니다. 이는 DB2 계열 및 Oracle** 데이터 소스에 대한 논리적 선택사항입니다(이 테이블의 끝에 있는 주 2 참조). 'N' 연합 서버는 사용자 ID를 데이터 소스로 전송하기 전에 아무것도 수행하지 않습니다(이 테이블의 끝에 있는 주 2 참조). 'L' 연합 서버는 사용자 ID를 데이터 소스로 전송하기 전에 소문자로 변경합니다. 이들 설정값 중 아무것도 사용되지 않으면, 연합 서버는 대문자로 사용자 ID를 데이터 소스로 전송하려 합니다. 사용자 ID가 실패하면 서버는 소문자로 전송을 시도합니다.	없음

표 8. 서버 옵션 및 설정값 (계속)

옵션	유효한 설정값	기본 설정값
fold_pw(이 테이블의 끝에 있는 주 1, 3 및 4 참조)	연합 서버가 인증을 위해 데이터 소스로 전송하는 암호에 적용됩니다. 유효 값은 다음과 같습니다. 'U' 연합 서버는 암호를 데이터 소스로 전송하기 전에 대문자로 변경합니다. 이는 DB2 계열 및 Oracle** 데이터 소스에 대한 논리적 선택사항입니다. 'N' 연합 서버는 암호를 데이터 소스로 전송하기 전에 아무것도 수행하지 않습니다. 'L' 연합 서버는 암호를 데이터 소스로 전송하기 전에 소문자로 변경합니다. 이들 설정값 중 아무것도 사용되지 않으면 연합 서버는 대문자로 암호를 데이터 소스로 전송하려 합니다. 암호가 실패하면 서버는 소문자로 전송을 시도합니다.	없음
io_ratio	연합 서버의 I/O 시스템 속도보다 데이터 소스의 I/O 시스템 속도가 얼마나 빠른지 또는 느린지를 나타냅니다. 유효 값은 0보다 크고 $1x10^{23}$보다 작습니다. 값은 123E10, 123 또는 1.21E4와 같이 유효한 이중 표기로 표시될 수 있습니다.	'1.0'
node	데이터 소스가 해당 RDBMS에서 인스턴스로 정의되는 이름입니다. 모든 데이터 소스에 필요합니다. DB2 계열 데이터 소스의 경우, 이 이름은 연합 데이터베이스의 DB2 노드 디렉토리에 지정되어 있습니다. 이 디렉토리를 보려면 **db2 list node directory** 명령을 발행하십시오. Oracle** 데이터 소스의 경우, 이 이름은 Oracle** `tnsnames.ora` 파일에 지정된 서버 이름입니다. Windows NT 플랫폼에서 이 이름에 액세스하려면 Oracle** SQL Net 쉬운 구성 도구의 **구성 정보 보기** 옵션을 지정하십시오.	없음

표 8. 서버 옵션 및 설정값 (계속)

옵션	유효한 설정값	기본 설정값
password	데이터 소스로 암호를 전송할지 여부를 지정합니다. 'Y' 암호가 항상 데이터 소스로 전송되며 유효성이 확인됩니다. 이는 기본값입니다. 'N' 암호가 데이터 소스로 전송되지 않으며(사용자 맵핑에 상관없이) 유효성이 확인되지 않습니다. 'ENCRYPTION' 암호가 항상 데이터 소스로 암호화 형식을 사용하여 전송되며 유효성이 확인됩니다. 암호화된 암호를 지원하는 DB2 계열 데이터 소스에 대해서만 유효합니다.	'Y'
plan_hints	*플랜 추가 정보*가 사용 가능한지 여부를 지정합니다. 플랜 추가 정보는 데이터 소스 최적화 알고리즘에 대한 추가 정보를 제공하는 명령문의 단편입니다. 특정 조회 유형의 경우, 이 정보는 조회 성능을 향상시킬 수 있습니다. 플랜 추가 정보는 데이터 소스 최적화 알고리즘이 색인 사용 여부, 사용할 색인 또는 사용할 테이블 조인 순서를 결정하는 데 도움을 줄 수 있습니다. 'Y' 데이터 소스가 플랜 추가 정보를 지원할 경우, 데이터 소스에서 플랜 추가 정보를 사용할 수 있습니다. 'N' 데이터 소스에서 플랜 추가 정보를 사용할 수 없습니다.	'N'
pushdown	'Y' DB2는 데이터 소스에서 조작을 평가하도록 합니다. 'N' DB2는 원격 데이터 소스에 있는 컬럼만을 검색하며 데이터 소스가 조인과 같은 다른 조작은 평가하지 못하게 합니다.	'Y'

표 8. 서버 옵션 및 설정값 (계속)

옵션	유효한 설정값	기본 설정값
varchar_no_trailing_blanks	이 데이터 소스가 공백이 아닌 문자로 채워진 varchar 비교 의미를 사용할지 여부를 지정합니다. 뒤 공백이 없는 변수 길이 문자열의 경우, 일부 DBMS의 공백이 아닌 문자로 채워진 비교 의미는 DB2의 비교 의미와 동일한 결과를 리턴합니다. 데이터 소스에 있는 모든 VARCHAR 테이블/뷰 컬럼에 뒤 공백이 포함되어 있지 않으면 이 서버 옵션을 데이터 소스의 경우에는 'Y'로 설정 고려하십시오. 이 옵션은 종종 Oracle** 데이터 소스에도 사용됩니다. 별칭(뷰를 포함하여)을 사용할 가능성이 있는 모든 오브젝트도 고려하도록 하십시오. 'Y' 이 데이터 소스는 DB2와 유사한 공백이 아닌 문자로 채워진 비교 의미를 가집니다. 'N' 이 데이터 소스는 DB2와 동일한 공백이 아닌 문자로 채워진 비교 의미를 갖지 않습니다.	'N'

120 페이지의 표8에 대한 주:

1. 이 필드는 인증에 지정된 값에 상관없이 적용됩니다.
2. DB2는 사용자 ID를 대문자로 저장하므로 'N'과 'U' 값은 논리적으로 서로 동일합니다.
3. fold_pw의 설정은 암호에 대한 설정이 'N'이면 영향을 미치지 않습니다. 암호가 전송되지 않기 때문에 대소문자는 인수가 될 수 없습니다.
4. 이 옵션에 널(NULL)을 설정하는 것은 피하도록 하십시오. DB2가 사용자 ID와 암호를 해석하기 위해 여러 번의 시도를 하기 때문에 널(NULL)로 설정할 수 있으나 성능이 저하됩니다. (데이터 소스 인증을 성공적으로 전달하기 전에 DB2가 네 번까지 사용자 ID 및 암호를 전송하는 것은 가능합니다.)

제5장 시스템 카탈로그 통계

SQL 조회를 최적화할 때 SQL 컴파일러가 내리는 결정은, 최적화 알고리즘 모델의 데이터베이스 내용에 영향을 크게 받습니다. 이러한 데이터 모델은 최적화 알고리즘에 의해 특정 조회를 해결하는 데 사용될 수 있는 대체 액세스 경로에 드는 비용을 계산하는 데 사용됩니다.

데이터 모델의 가장 중요한 요소는 데이터베이스에 포함된 데이터에 대해 수집하여 시스템 카탈로그 테이블에 저장해 놓은 통계 세트입니다. 여기에는 테이블, 별칭, 색인, 컬럼 및 사용자 정의 함수(UDF) 등이 포함됩니다. 데이터 통계상의 변경은 원하는 데이터에 액세스하는 가장 효율적인 메소드로서 선택되는 액세스 플랜의 선택사항에 변화를 가져오게 됩니다.

최적화 알고리즘에 대한 데이터 모델을 정의하는 데 도움이 되는 통계의 예에는 다음과 같은 내용이 있습니다.

- 테이블의 페이지 수 및 비어 있지 않은 페이지의 수
- 행이 원래의 페이지에서 다른(오버플로우) 페이지로 이동한 정도
- 테이블의 행의 수
- 컬럼에 있는 구별 값의 수
- 색인의 클러스터링 등급. 즉, 테이블에 있는 행의 물리적 순서가 색인을 따르는 extent
- 색인 레벨의 수 및 각각의 색인에서 리프 페이지의 수
- 자주 사용된 컬럼 값의 발생 수(135 페이지의 『분산 통계 수집 및 사용』 참조)
- 컬럼에 표시되는 값의 범위에 걸친 컬럼 값의 분산(135 페이지의 『분산 통계 수집 및 사용』 참조)
- 사용자 정의 함수(UDF)에 대한 비용 계산

오브젝트에 대한 통계는 명확하게 요청될 때에만 시스템 카탈로그 테이블에서 갱신됩니다. 통계의 일부 또는 전체는 다음과 같은 방법으로 갱신될 수 있습니다.

- RUNSTATS(통계 수행) 유틸리티 사용(127 페이지의 『RUNSTATS 유틸리티를 사용하여 통계 수집』 참조)

© Copyright IBM Corp. 1993, 2001

- 통계 수집 옵션을 지정하여 LOAD 사용
- 사전 정의된 카탈로그 뷰(view) 세트에 대하여 작업하는 SQL UPDATE문 코딩(150 페이지의『사용자가 갱신할 수 있는 카탈로그 통계』참조). 사용자 정의 함수(UDF)에 대한 통계는 반드시 이 방법으로 갱신되어야 함에 주의하십시오(157 페이지의『사용자 정의 함수(UDF)에 대한 통계 갱신』참조). UDF의 경우를 제외하면 카탈로그는 테스트 시스템상에 제품 환경을 모델링하는 경우나『이 경우에는 어떻게 될까를 분석(what-if analysis)』하는 경우에는 수동으로만 갱신되어야 합니다. 생산 시스템에서는 통계가 갱신되어서는 안됩니다.

연합 데이터베이스 시스템에서 데이터 소스로부터 별칭에 대한 새 통계를 수집하는 유일한 방법은 별칭을 삭제하고 데이터 소스에서 동등한 RUNSTATS를 수행한 다음 별칭을 새로 작성하는 것입니다. 별칭이 작성될 때마다 데이터 소스 카탈로그로부터 기존 테이블에 대한 통계가 수집됩니다.

기존 테이블에 있는 데이터 정의 정보가 변경되면 별칭을 삭제한 다음 재작성해야 합니다. 예를 들어, 컬럼이 테이블 정의에 추가된 경우입니다.

또한 조회 성능이 저하된 경우에도 별칭을 재작성해야 합니다. 다른 접근 방법은 SYSSTAT.TABLES에서 수동으로 통계를 갱신하는 것입니다.

뷰에 대한 별칭을 작성할 때에는 주의를 기울이십시오. 이 별칭이 리턴하는 행 수와 같은 통계 정보는 이 뷰를 평가하는 데 필요한 실제 비용을 반영하지 않을 수도 있습니다. 뷰가 SELECT 목록에 적용되는 컬럼 함수가 없는 하나의 기본 테이블에 정의된 경우, 최적화 알고리즘에 사용 가능한 통계 정보는 정확해야 합니다. 뷰가 복잡한 경우에는 최적화 알고리즘이 효율적인 데이터 액세스 플랜을 생성할 수 있도록 연합 데이터베이스 시스템의 DB2 Universal Database 서버에 있는 뷰 기본 테이블에 대한 별칭을 통해 새 뷰를 작성하는 것을 고려해 보십시오.

추가 정보

SYSCAT 및 SYSSTAT 카탈로그에는 수집된 통계에 대한 정보가 들어 있습니다. 자세한 내용은 *SQL 참조서*를 참조하십시오.

- 모든 카탈로그 뷰 및 포함된 컬럼에 대한 내용

- 갱신 가능한 모든 카탈로그 뷰 및 포함된 컬럼에 대한 내용. 카탈로그 테이블의 통계 컬럼에 대해서만 관심을 가지고 있는 경우, 이 부분도 참조할 수 있습니다.
- 테이블 통계에 대한 내용
- 컬럼 통계에 대한 내용
- 컬럼 분산 통계에 대한 내용
- 색인 통계에 대한 내용
- 사용자 정의 함수(UDF) 통계에 대한 내용

RUNSTATS 유틸리티를 사용하여 통계 수집

RUNSTATS 유틸리티는 시스템 카탈로그 테이블의 통계를 갱신하여 조회 최적화 프로세스를 도와줍니다. 이 통계가 없으면 데이터베이스 관리 프로그램은 SQL문의 성능에 부정적인 영향을 미치는 결정을 내릴 수도 있습니다. RUNSTATS 유틸리티는 테이블이나 색인 또는 둘다에 포함되어 있는 데이터에 대한 통계를 수집할 수 있도록 해줍니다.

다음과 같은 상황에서 액세스 플랜 선택 프로세스에 정확한 정보를 제공하려면 RUNSTATS 유틸리티를 사용하여 테이블 및 색인 데이터 모두에 근거한 통계를 수집하십시오.

- 테이블에 데이터가 로드되어 있는 경우 및 적절한 색인이 작성되어 있는 경우
- 테이블이 REORG 유틸리티를 사용하여 재구성되어 있는 경우
- 테이블과 해당 색인에 영향을 미치는 광범위한 갱신, 삭제 및 삽입 작업이 있는 경우(여기서 『광범위한』이라는 단어의 의미는 테이블과 색인 데이터의 10%에서 20%까지 영향을 받는 경우를 말합니다.)
- 성능이 아주 중요한 요소로 간주되는 응용프로그램을 바인딩하기 전
- 이전의 통계와 비교하는 것이 바람직할 경우. 주기적으로 통계를 수행하여 성능상의 문제점을 조기에 발견할 수 있습니다.
- 프리페치량이 변경될 때
- REDISTRIBUTE NODEGROUP 유틸리티를 사용할 경우

파티션된 데이터베이스에서 작업할 때에는 단일 노드에서 RUNSTATS 조작을 실행하여 테이블 및 해당 색인과 관련된 통계를 수집하십시오. (유틸리티가 실행되는

노드는 명령을 발행할 노드가 테이블 데이터를 가지고 있는지의 여부에 따라 결정됩니다. 자세한 내용은 『RUNSTATS가 실행되는 데이터베이스 파티션』을 참조하십시오.) 카탈로그에 저장된 통계는 테이블 레벨의 정보를 표시한다고 가정되기 때문에, 데이터베이스 관리 프로그램이 수집하는 노드 레벨 통계에는 테이블이 파티션된 노드의 수가 적당히 곱해집니다. 이렇게 하면 모든 노드에서 RUNSTATS를 실행하고 이 통계를 총계하여 수집되는 실제 통계의 근사치를 제공합니다.

주: DB2 조회 최적화 알고리즘은 속성 값(데이터)이 시스템의 데이터베이스 파티션에 걸쳐 똑같이 균등하게 배치된 것으로 간주합니다. 데이터의 위치가 같지 않을 경우, 표시 테이블 분산을 가지고 있는 것으로 생각되는 데이터베이스 파티션에서 이 명령을 수행해야 합니다.

RUNSTATS가 실행되는 데이터베이스 파티션

테이블에서 RUNSTATS을 호출할 경우에는 테이블이 저장된 데이터베이스에 연결되어야 하지만, 명령을 발행한 데이터베이스 파티션이 이 테이블에 대한 파티션을 포함해야 할 필요는 없습니다.

- 테이블에 대한 파티션이 들어 있는 데이터베이스 파티션에서 RUNSTATS를 발행할 경우, 유틸리티는 이 데이터베이스 파티션에서 실행됩니다.
- 테이블 파티션을 포함하지 않는 데이터베이스 파티션에서 RUNSTATS를 발행하면 요청은 테이블에 대한 파티션을 보유하고 있는 노드 그룹의 첫 번째 데이터베이스 파티션으로 전송됩니다. 그러면 유틸리티는 이 데이터베이스 파티션에서 실행됩니다.

통계 분석

통계를 분석하면 재구성이 필요한 시기를 알 수 있습니다. 이는 다음 정보를 통해 알 수 있습니다.

- 색인의 클러스터링

  클러스터 비율 통계가 수집된 경우, 값의 범위는 0에서 100까지입니다. 클러스터 인수 통계를 수집하면 값은 0에서 1 사이에 있게 됩니다. 이 두 클러스터링 통계 중 하나만이 SYSCAT.INDEXES 카탈로그에 기록됩니다. 일반적으로, 테

이블의 색인 중 하나의 색인만 클러스터링의 등급이 높을 수 있습니다. -1 값은 어떠한 통계도 사용할 수 없음을 나타내는 데 사용됩니다.

비율 값을 비교하려는 경우, 클러스터 인수에 100을 곱하여 클러스터링의 양에 대한 백분율 값을 구하십시오.

색인 전용 액세스가 **아닌** 색인 스캔의 경우, 클러스터 비율이 높을 때 더 나은 성능을 보일 수도 있습니다. 이러한 유형의 스캔의 경우, 클러스터 비율이 낮으면 더 많은 I/O 작업을 하게 되는데, 이는 각각의 데이터 페이지에 최초로 액세스한 후 다시 그 페이지에 액세스할 때, 계속 버퍼 풀에 있을 가능성이 더 적기 때문입니다. 버퍼 크기를 늘리면 클러스터되어 있지 않은 색인의 성능을 향상시킬 수 있습니다. 데이터베이스 관리 프로그램이 클러스터 비율이 낮은 색인에 대한 색인 스캔 성능을 향상시키는 방법에 대한 자세한 내용은 294 페이지의 『목록 프리페치의 이해』를 참조하고, 최적화 알고리즘이 색인 통계를 사용하는 방법에 대한 자세한 내용은 192 페이지의 『클러스터된 색인』을 참조하십시오.

테이블 데이터가 특정 색인에 의해 초기에 클러스터되고 위의 클러스터링 정보에 의해 데이터가 현재 동일한 색인에 대해 잘못 클러스터되어 있다고 판단되는 경우, 테이블을 재구성하여 해당 색인에 따라 데이터를 다시 클러스터하려고 할 수 있습니다.

- 행의 오버플로우

  오버플로우 번호는 원래의 페이지에 맞지 않는 행의 번호를 나타냅니다. 이는 VARCHAR 컬럼이 더 긴 값을 가지고 갱신될 때 발생할 수 있습니다. 이러한 경우에 있어서 포인터는 원래의 행 위치를 여전히 가리키고 있습니다. 행의 내용을 찾기 위해서는 데이터베이스 관리 프로그램이 포인터를 따라가야 하고, 그렇게 되면 처리 시간도 길어지고 I/O 횟수도 증가되기 때문에, 이 방법은 성능을 저하시킬 수 있습니다.

  오버플로우 행의 수가 많아질수록 테이블 데이터 재구성으로 얻어지는 잠재적인 이점도 증가하게 됩니다. 테이블 데이터를 재구성하면 행의 오버플로우가 제거됩니다.

- 파일 페이지 비교

행이 있는 페이지의 수가 테이블이 포함하는 전체 페이지 수와 비교될 수 있습니다. 테이블 스캔을 위해 비어 있는 페이지를 읽어들입니다. 비어 있는 페이지는 행 전체 범위가 삭제되면 생기게 됩니다.

비어 있는 페이지의 수가 증가할수록 테이블 재구성의 필요성도 커집니다. 테이블을 재구성하면 비어 있는 페이지를 이용하여 테이블에서 사용하고 있는 공간을 압축할 수 있습니다. 이렇게 디스크 공간을 더욱 효율적으로 사용할 수 있을 뿐 아니라, 미사용 페이지를 이용하여 테이블 스캔의 성능을 향상시킬 수도 있는데, 이는 버퍼 풀로 더 적은 수의 페이지를 읽어들이기 때문입니다.

- 리프(leaf) 페이지 수

  리프 페이지 수는 전체 색인 스캔에 얼마나 많은 색인 페이지 I/O가 필요한지 예측합니다.

  무작위 갱신 활동으로 인해 필요한 최소 공간량 이상으로 색인 크기를 증가시키는 페이지 분할이 발생할 수 있습니다. 테이블 재구성 중에 색인이 재빌드되면 가능한 최소 공간량으로 각 색인을 빌드할 수 있습니다. 색인에 필요한 최소 공간 요구량에 대한 자세한 내용은 109 페이지의 『색인화가 조회 최적화에 미치는 영향』 또는 *관리 안내서: 계획*의 『색인 또는 색인 스펙 작성』 절을 참조하십시오.

  주: 색인이 재빌드된 경우, 기본 10%의 여유 공간이 색인 페이지마다 남아 있습니다. 처음으로 색인을 작성할 때 PCTFREE 매개변수를 사용하여 여유 공간량을 늘릴 수 있습니다. 그런 다음 색인을 재구성할 때마다 PCTFREE 값이 사용됩니다. 색인을 재구성하는 데 필요한 시간을 줄이려면 여유 공간이 10%를 넘어야 합니다. 여유 공간은 추가 색인 삽입을 수용하는 데 사용됩니다.

RUNSTATS는 성능이 데이터베이스 내의 변경사항과 얼마나 관계가 있는지 판단하는 데 도움을 줄 수도 있습니다. 통계에서는 테이블 내의 데이터 분산을 보여줍니다. 이를 규칙적으로 사용하면 RUNSTATS는 일정 기간이 경과한 후 테이블과 색인에 대한 데이터를 제공하고, 그 결과 시간이 지남에 따라 볼 수 있는 데이터 모델의 성능 경향이 파악됩니다.

조회 최적화 알고리즘은 새 통계가 제공된 다른 액세스 플랜을 선택하는 경우도 있으므로, 통계를 수행한 후 응용프로그램을 리바인드해야하는 것이 이상적입니다.

사용 가능한 시간이 한 번에 모든 통계를 수집하기에 충분하지 않은 경우, 수집 가능한 통계의 일부만 갱신하기 위해 RUNSTATS를 주기적으로 수행하도록 선택할 수 있습니다. 선택적인 부분 갱신으로 RUNSTATS를 수행하는 기간 사이의 테이블에서 활동의 결과로 불일치가 나타나면 바로 경고 메시지(SQL0437W, 이유 코드 6)가 발행됩니다. 예를 들어, 먼저 RUNSTATS를 사용하여 테이블 분산 통계를 수집합니다. 그런 다음 RUNSTATS를 사용하여 색인 통계를 수집합니다. 테이블에서 활동의 결과로 불일치가 검출되면 바로 테이블 분산 통계는 삭제되고 경고 메시지가 발행됩니다. 이러한 상황이 발생하면 RUNSTATS를 수행하여 테이블 분산 통계를 수집하는 것이 좋습니다.

정기적으로 RUNSTATS를 사용하여 테이블 및 색인 통계를 동시에 수집해야 하며, 색인 통계와 테이블 통계가 동기화되었는지 확인해야 합니다. 색인 통계는 RUNSTATS의 마지막 수행으로부터 수집된 대부분의 테이블 및 컬럼 통계를 보유합니다. 최종적으로 테이블 통계가 수집된 후 테이블이 광범위하게 수정될 경우, 해당 테이블에 대해서만 색인 통계를 수집하여 두 세트의 통계가 동기화 상태를 벗어나게 됩니다.

다음과 같은 상황에서 색인 데이터만을 근거로 통계를 수집하려고 할 수 있습니다.

- RUNSTATS 유틸리티가 수행된 이후로 새 색인이 작성되고, 테이블 데이터에 대한 통계를 수집하려고 하지 않을 경우
- 색인의 첫 번째 컬럼에 영향을 미치는 데이터에 많은 변경사항이 있는 경우

RUNSTATS 유틸리티를 사용하여 다양한 레벨의 통계 정보를 수집할 수도 있습니다. 테이블에 대해서는 기본 레벨의 통계를 수집하거나, 테이블 내의 컬럼 값에 대한 분산 통계 정보를 수집할 수도 있습니다(135 페이지의 『분산 통계 수집 및 사용』 참조). 색인에 대해서는 기본 레벨의 통계 정보를 수집할 수도 있고, 최적화 알고리즘이 색인 스캔의 I/O 비용을 계산하는 데 더 많은 도움이 될 수 있는 세부사항 통계 정보를 수집할 수도 있습니다. (이러한 『세부사항』 통계에 대한 자세한 내용은 192 페이지의 『클러스터된 색인』을 참조하십시오.)

주: 통계는 LONG, 대형 오브젝트(LOB) 또는 구조화된 유형 컬럼에 대해서는 수집되지 않습니다. 행 유형의 경우, 테이블 레벨 통계 NPAGES, FPAGES 및

OVERFLOW는 서브테이블용으로는 수집되지 않습니다. 통계는 확장 색인이나 선언된 임시 테이블에 대해서 수집되지 않습니다.

다음 표에서는 RUNSTATS 유틸리티에 의해 갱신된 카탈로그 통계를 보여줍니다.

표 9. 테이블 통계(SYSCAT.TABLES 및 SYSSTAT.TABLES)

통계	설명	RUNSTATS 옵션	
		테이블	색인
FPAGES	테이블에서 사용하는 페이지의 수	예	예
NPAGES	행이 들어 있는 페이지의 수	예	예
OVERFLOW	오버플로우되는 행의 수	예	아니오
CARD	테이블의 행 수(기본 행 수(cardinality))	예	예(주 2)

주:

1. 파티션된 데이터베이스의 경우, 데이터베이스 파티션 계수를 데이터베이스 파티션 수에 곱한 값으로부터 각 통계 값이 추정됩니다.
2. 테이블에 색인이 정의되지 않고 사용자가 색인 통계를 요청할 경우, 새 CARD 통계가 갱신되지 않습니다. 이전의 CARD 통계는 보유됩니다.

표 10. 컬럼 통계(SYSCAT.COLUMNS 및 SYSSTAT.COLUMNS)

통계	설명	RUNSTATS 옵션	
		테이블	색인
COLCARD	컬럼 기본 행 수(cardinality)	예(주 1)	예(주 2)
AVGCOLLEN	평균 컬럼 길이	예	예(주 2)
HIGH2KEY	컬럼의 두 번째 상위 값	예	예(주 2)
LOW2KEY	컬럼의 두 번째 하위 값	예	예(주 2)
NUMNULLS	컬럼 내의 NULL의 수	예	예(주 2)

주:

1. COLCARD는 테이블의 모든 컬럼에 대하여 추정됩니다. 파티션된 데이터베이스에서 컬럼이 테이블에 대한 단일 컬럼 파티션 키일 경우, 데이터베이스 파티션 계수를 데이터베이스 파티션 수에 곱한 값으로부터 계수 값이 추정됩니다.
2. 컬럼 통계는 색인 키의 첫 번째 컬럼에 대하여 수집됩니다.

표 11. 색인 통계(SYSCAT.INDEXES 및 SYSSTAT.INDEXES)

통계	설명	RUNSTATS 옵션	
		테이블	색인
NLEAF	색인 리프 페이지의 수	아니오	예(주 3)
NLEVELS	색인 레벨의 수	아니오	예
CLUSTERRATIO	테이블 데이터의 클러스터링 등급	아니오	예(주 2)
CLUSTERFACTOR	클러스터링의 좀더 자세한 등급	아니오	세부사항(주 1,2)
DENSITY	색인(주 4)이 차지한 페이지 범위에서 페이지 수에 대한 SEQUENTIAL_PAGES의 비율(백분율)	아니오	예
FIRSTKEYCARD	색인의 첫 번째 컬럼 내 구별 값의 수	아니오	예(주 3)
FIRST2KEYCARD	색인의 첫 번째 두 개의 컬럼 내 구별 값의 수	아니오	예(주 3)
FIRST3KEYCARD	색인의 첫 번째 세 개의 컬럼 내 구별 값의 수	아니오	예(주 3)
FIRST4KEYCARD	색인의 첫 번째 네 개의 컬럼 내 구별 값의 수	아니오	예(주 3)
FULLKEYCARD	색인의 모든 컬럼 내 구별 값의 수	아니오	예(주 3)
PAGE_FETCH_PAIRS	서로 다른 버퍼 크기에 대한 페이지 페치 예측	아니오	세부사항(주 1,2)
SEQUENTIAL_PAGES	서로 간격이 거의 없는, 색인 키순으로 디스크에 배치된 리프 페이지의 수	아니오	예

표 11. 색인 통계(SYSCAT.INDEXES 및 SYSSTAT.INDEXES) (계속)

통계	설명	RUNSTATS 옵션	
		테이블	색인
주: 1. 세부사항 색인 통계는 RUNSTATS 명령에 DETAILED절을 지정하거나, RUNSTATS API를 호출할 때 statsopt 매개변수에 대하여 A, Y 또는 X를 지정하여 수집됩니다. 2. 테이블의 크기가 적절하지 않으면 CLUSTERFACTOR 및 PAGE_FETCH_PAIRS는 DETAILED절로 수집되지 않습니다. 테이블이 약 25페이지 보다 크면 CLUSTERFACTOR와 PAGE_FETCH_PAIRS 통계가 수집됩니다. 이 경우, CLUSTERRATIO는 -1(수집되지 않음)입니다. 테이블이 비교적 작은 경우, CLUSTERFACTOR 및 PAGE_FETCH_PAIRS는 구해지지 않는 반면, RUNSTATS에 의해 CKUSTERRATIO만 구해집니다. DETAILED 절이 지정되지 않으면 CLUSTERRATIO 통계만 수집됩니다. 3. 파티션된 데이터베이스의 경우, 데이터베이스 파티션 계수를 데이터베이스 파티션 수에 곱한 값으로부터 값이 추정됩니다. 4. 이 통계 방법을 사용하여 해당 테이블에 속한 색인이 들어 있는 파일에서 페이지의 백분율을 측정합니다. 정의된 색인이 하나밖에 없는 테이블의 경우, DENSITY는 보통 100이어야 합니다. 색인 페이지가 프리페치된 경우, 최적화 알고리즘이 DENSITY를 사용하여 평균적으로 다른 색인에서 얼마나 많은 비관련 페이지를 읽어들이는지 계산합니다.		

표 12. 컬럼 분산 통계(SYSCAT.COLDIST 및 SYSSTAT.COLDIST)

통계	설명	RUNSTATS 옵션	
		테이블	색인
DISTCOUNT	TYPE이 Q일 경우, COLVALUE 통계 이하인 구별 값 수	분산(주 2)	아니오
TYPE	행이 빈번하게 사용되는 값 또는 quantile 통계를 제공하는지 여부에 대한 표시기	분산	아니오
SEQNO	테이블의 행을 고유하게 식별하는 데 도움이 되는 순차 번호의 빈도순 순위	분산	아니오
COLVALUE	자주 사용되는 데이터 값 또는 quantile 통계가 수집되는 데이터 값	분산	아니오
VALCOUNT	데이터 값이 컬럼에서 발생하는 빈도 또는 quantile 빈도, 데이터 값 이하인 값의 수(COLVALUE)	분산	아니오

표 12. 컬럼 분산 통계(SYSCAT.COLDIST 및 SYSSTAT.COLDIST) (계속)

통계	설명	RUNSTATS 옵션	
		테이블	색인

주:

1. 컬럼 분산 통계는 RUNSTATS 명령에 WITH DISTRIBUTION절을 지정하거나, RUNSTATS API를 호출할 때 `statsopt` 매개변수에 대하여 `A`, `D` 또는 `Y`를 지정하여 수집할 수 있습니다. 컬럼 값에서 충분한 일관성이 부족하지 않으면 분산 통계는 수집되지 **않습니다**.
2. 컬럼이 색인의 첫 번째 키 컬럼인 경우에만 DISTCOUNT가 수집됩니다.
3. 파티션된 데이터베이스의 경우, 데이터베이스 파티션 계수를 데이터베이스 파티션 수에 곱한 값이 VALCOUNT를 예측합니다. TYPE이 'F'이고 컬럼이 테이블의 단일 컬럼 파티션 키일 경우에 이에 대한 예외가 발생하는데, 이 때 VALCOUNT는 단순히 데이터베이스 파티션의 계수입니다.

컬럼 분산 통계에 대한 자세한 내용은『분산 통계 수집 및 사용』을 참조하십시오.

사용자 정의 함수(UDF)에 대한 통계는 RUNSTATS 유틸리티로 수집할 수 없습니다. 이 함수에 대한 통계는 수동으로 갱신해야 합니다. 150 페이지의『사용자가 갱신할 수 있는 카탈로그 통계』및 157 페이지의『사용자 정의 함수(UDF)에 대한 통계 갱신』을 참조하십시오.

분산 통계 수집 및 사용

데이터베이스 관리 프로그램은 간단한 방법으로 두 가지 유형의 통계(『자주 사용되는 값 통계』및『quantile』)를 수집, 유지보수 및 사용할 수 있으며, 이 통계는 컬럼 내의 데이터 값의 분산을 추정합니다. 최적화 알고리즘이 이 통계를 사용함으로써 주어진 등호 또는 범위 술어를 충족시키는 컬럼의 행 수를 훨씬 더 정확하게 추정할 수 있습니다. 이러한 정확한 추정은 다시 최적화 알고리즘이 최적 플랜을 선택할 수 있는 가능성을 높입니다.

RUNSTATS문 명령에 WITH DISTRIBUTION절을 사용하여 이러한 데이터 값의 분산에 대한 통계를 수집할 수도 있습니다. 이러한 추가 통계를 수집하는 것은 RUNSTATS 유틸리티에 대하여 추가 오버헤드가 되기도 하지만, SQL 컴파일러는 이 정보를 사용하여 최상의 액세스 플랜을 선택하는 데 도움이 될 수 있습니다.

간혹, 데이터베이스 관리 프로그램은 분산 통계를 수집하지 않으며, 이 때 아무런 오류도 리턴되지 않습니다. 예를 들면, 다음과 같습니다.

- *num_freqvalues* 및 *num_quantiles* 구성 매개변수는 0으로 설정하여 분산 통계 수집을 원하지 않는 것을 나타냅니다. 이 매개변수에 대한 자세한 내용은 다음을 참조하십시오.
  - 140 페이지의『얼마나 많은 통계를 유지해야 하는가?』
  - 516 페이지의『보유되어 자주 사용되는 값의 수(num_freqvalues)』
  - 517 페이지의『컬럼에 대한 quantile 수(num_quantiles)』
- 분산 통계를 사용하지 않고도 데이터 분산을 알 수 있는 경우입니다. 예를 들면, 두 번 이상 나오는 데이터 값이 없는 컬럼, 즉 컬럼 내의 모든 데이터 값이 고유한 경우입니다.
- 데이터 유형이 통계가 수집되지 않는 종류인 경우입니다. 즉, 컬럼이 Long 필드 또는 대형 오브젝트(LOB) 데이터 유형을 사용하여 정의된 것입니다.
- quantile 통계의 경우, 단 하나의 NULL이 아닌 값이 컬럼 내에 존재하는 경우입니다.

색인의 첫 번째 컬럼에 대해 분산 통계가 정확합니다. 각 추가 컬럼의 경우, 정확한 통계를 계산하려면 너무 많은 시간과 메모리가 필요하기 때문에, 데이터베이스 관리 프로그램은 해쉉 및 샘플링 기법을 사용하여 분산 통계를 추정합니다. 이러한 기법은 상당한 정확도를 지닌 통계 기법으로 인정받고 있습니다.

분산 통계는 SYSSTAT.COLDIST를 갱신하고 분산 통계가 더 이상 필요하지 않은 컬럼에 대해 모든 COLVALUE 및 VALCOUNT 값은 0이나 -1로 설정하여 제거할 수 있습니다.

다음에서는 이러한 분산 통계에 대하여 사용자가 이해하고 사용하는 데 도움이 될 내용을 제공하고 있습니다.

- 분산 통계의 이해
- 언제 분산 통계를 사용해야 하는가?
- 얼마나 많은 통계를 유지해야 하는가?
- 최적화 알고리즘은 어떻게 분산 통계를 사용하는가?
- 생산 데이터베이스 모델링
- 컬럼에 대한 분산 통계 갱신 규칙

분산 통계의 이해

고정된 숫자인 N>=1의 경우, 컬럼에서 *가장 자주 사용되는 값 N*은 빈도가 가장 높은 데이터 값(즉, 중복 횟수), 빈도가 두 번째로 높은 데이터 값, 빈도가 N번째로 높은 데이터 값으로 구성됩니다. 이러한 *자주 사용되는 값 통계*는 『N』개의 데이터 값과 컬럼에서의 이들 데이터 값의 빈도로 구성됩니다.

컬럼에 대한 *K-quantile*은 가장 작은 데이터 값인 V이므로, 적어도 『K』 행은 V보다 작거나 같은 데이터 값을 가지고 있습니다. K-quantile은 컬럼의 행을 증가하는 데이터 값에 따라 정렬하여 계산될 수 있습니다. K-quantile은 정렬된 컬럼의 K번째 행에 있는 데이터 값입니다.

예를 들면, 다음과 같은 데이터 컬럼을 고려해 보십시오.

```
C1
--
 B
 E
 Y
 B
 F
 G
 E
 A
 J
 K
 E
 L
```

이 컬럼은 다음과 같은 순서로 값을 얻기 위해 정렬될 수 있습니다.

```
C1'
--
 A
 B
 B
 E
 E
 E
 F
 G
 J
 K
 L
 Y
```

컬럼 C1에는 9개의 구별 데이터 값이 있습니다. N = 2에 대해 자주 사용되는 값 통계는 다음과 같습니다.

```
SEQNO     COLVALUE      VALCOUNT
-----     ---------     --------
  1           E             3
  2           B             2
```

수집될 quantile 수가 5이면(517 페이지의 『컬럼에 대한 quantile 수 (num_quantiles)』 참조) K = 1, 3, 6, 9 및 12인 경우, 이 컬럼의 K-quantile은 다음과 같습니다.

```
SEQNO     COLVALUE      VALCOUNT
-----     ---------     --------
  1           A             1
  2           B             3
  3           E             6
  4           J             9
  5           Y            12
```

이 예에서 정렬된 컬럼의 6번째 행이 E와 같은 데이터 값을 가지고 있기 때문에 (그리고 원래의 컬럼의 6행이 E보다 같거나 작은 데이터 값을 가지고 있기 때문에) 6-quantile은 E와 같습니다.

quantile 값이 공통의 값이면 같은 quantile 값이 두 번 이상 발생할 수 있습니다. 두 개의 quantile의 최대값은 주어진 값에 대하여 저장됩니다. 두 개의 이 quantile 중 첫 번째는 COLVALUE보다 확실히 작은 행의 수를 제공하는 VALCOUNT를 가지고 있으며, 두 번째는 COLVALUE보다 작거나 같은 행의 수를 제공합니다.

언제 분산 통계를 사용해야 하는가?

주어진 테이블에 대해 분산 통계를 사용해야 할지의 여부를 결정하려면 두 가지 인수가 고려되어야 합니다.

1. 정적 또는 동적 SQL 사용

   분산 통계는 동적 SQL 및 호스트 변수를 사용하지 않는 정적 SQL에 아주 유용합니다. 호스트 변수와 함께 SQL을 사용하면 최적화 알고리즘은 분산 통계를 제한하여 사용합니다.

2. 데이터 분산에 있어서의 일관성 결여

테이블에서 최소한 하나의 컬럼에 매우『균일하지 않은(non-uniform)』데이터 분산이 있고, 컬럼이 등호(equality) 또는 범위 술어에 빈번하게 나타나는 경우, 분산 통계를 사용하는 것이 좋습니다. 즉, 다음과 같은 절에서 사용하십시오.

```
WHERE C1 = KEY;
WHERE C1 IN(KEY1, KEY2, KEY3);
WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
WHERE C1 <= KEY;
WHERE C1 BETWEEN KEY1 AND KEY2;
```

데이터 분산에는 두 가지 유형의 비균일성(non-uniformity)이 존재하는데, 이 두 가지가 함께 발생할 수도 있습니다.

- 한 가지 유형은 최상위의 데이터 값과 최하위의 데이터 값 사이에 데이터가 균등하게 분산되어 있지 않고, 어느 지점에 몰려 있는 경우입니다. 예를 들어, 다음 컬럼에서처럼 데이터가 범위(5,10)에 클러스터되어 있습니다.

  ```
  C1
  -----
    0.0
    5.1
    6.3
    7.1
    8.2
    8.4
    8.5
    9.1
   93.6
  100.0
  ```

  이러한 종류의 비균일성(non-uniformity)이 존재하는 경우, quantile을 유지하는 것이 유용합니다.

  다음 예에서는 컬럼에 데이터 분산상의 비균일성이 존재하는지 여부를 판단하는 데 도움이 될 수 있는 조회를 보여줍니다.

  ```
  SELECT C1, COUNT(*) AS OCCURRENCES
    FROM T1
  GROUP BY C1
  ORDER BY OCCURRENCES DESC;
  ```

- 비균일성의 또 다른 유형은 아래에서처럼 특정 데이터 값이 다른 나머지 데이터 값에 비해 월등히 높은 빈도 수를 보이는 경우입니다.

데이터 값	빈도
20	5
30	10
40	10
50	25
60	25
70	20
80	5

이러한 유형의 비균일성이 보이는 경우, quantile 및 자주 사용되는 값 통계를 유지하는 것이 유용합니다.

RUNSTATS 명령에 WITH DISTRIBUTION절을 사용하거나, RUNSTATS API를 호출할 때 `statsopt` 매개변수에 대하여 `D`, `E` 또는 `A`를 지정하여 분산 통계를 수집할 수 있습니다. API에 대한 자세한 내용은 *Administrative API Reference* 매뉴얼을 참조하십시오.

얼마나 많은 통계를 유지해야 하는가?

많은 컬럼 분산 통계를 유지하면 최적화 알고리즘이 액세스 플랜을 선택하는 데 도움이 되지만, 한편으로 이러한 통계를 수집하고 조회를 컴파일하는 데 드는 비용도 함께 증가합니다. 통계 힙(heap)의 *크기*(419 페이지의 『통계 힙 *크기* (stat_heap_sz)』 참조)로 인해 계산되고 저장될 수 있는 통계의 숫자가 제한될 수도 있습니다.

분산 통계가 요청되면 데이터베이스 관리 프로그램은 기본값으로 컬럼에 대해 가장 자주 사용되는 10개의 값을 저장합니다. 실제적으로 10과 100개 사이의 자주 사용되는 값이 있으면 충분합니다. 그러나 나머지 값의 빈도가 서로 거의 비슷하거나, 가장 자주 사용되는 값의 빈도와 비교하여 무시할 정도가 되도록, 충분히 자주 사용되는 값 통계가 확보되어야 하는 것이 이상적입니다.

수집하기 위해 자주 사용되는 값의 수를 설정하려면 516 페이지의 『보유되어 자주 사용되는 값의 수(num_freqvalues)』에 설명된 대로 *num_freqvalues* 구성 매개변수를 사용하십시오. 이 통계는 두 번 이상 발생하는 데이터 값에 대해서만 수집되기 때문에 데이터베이스 관리 프로그램은 자주 사용되는 값 통계 수보다 작은 수를 수집할 수 있습니다. quantile 통계만을 수집하는 경우, 이 매개변수는 0으로 설정할 수 있습니다.

분산 통계가 요청되면 데이터베이스 관리 프로그램은 기본값으로 컬럼에 대해 20개의 quantile을 저장합니다. 이 값을 사용하면 단면 범위 술어(>, >=, < 또는 <=)의 최대 추정 오류는 약 2.5%이고 BETWEEN 술어의 최대 추정 오류는 5%로 보장됩니다. quantile 수를 판별할 때 다음과 같이 대략적으로 판별할 수 있습니다.

- 범위 조회의 행 수를 추정할 때 허용할 만한 최대 오류를 백분율 P로 결정하십시오.
- 술어가 BETWEEN 술어일 경우 quantile의 수는 약 100/P이고, 술어가 다른 유형의 범위 술어(<, <=, > 또는 >=)일 경우 quantile의 수는 50/P입니다.

예를 들어, 25개의 quantile은 BETWEEN 술어에 대해 4%와 ″>″ 술어에 대해 2%로 최대 추정 오류를 일으키게 됩니다. 일반적으로 적어도 10개의 quantile은 유지되어야 하고, 극단적으로 균일하지 않은(non-uniform) 데이터에 한해서만 51개 이상의 quantile이 필요합니다.

quantile의 수를 설정하려면 517 페이지의 『컬럼에 대한 quantile 수(num_quantiles)』에 설명된 대로 *num_quantiles* 구성 매개변수를 사용하십시오. 단지 자주 사용되는 값 통계만을 수집하는 경우, 이 매개변수는 0으로 설정될 수 있습니다. 이 매개변수를 『1』로 설정하면 전 범위의 값이 하나의 quantile에 맞게 되므로, 역시 quantile 통계를 수집할 수 없습니다.

최적화 알고리즘은 어떻게 분산 통계를 사용하는가?

분산 통계를 수집하고 저장하는 이유가 무엇인가? 그 대답은 바로 최적화 알고리즘이 가장 경제적인 액세스 플랜을 선택하기 위한 등호(equality) 또는 범위 술어를 충족시키는 컬럼의 행 수를 추정하는 데 필요하기 때문입니다. 더 정확한 추정을 할수록 최적화 알고리즘이 최적의 액세스 플랜을 선택할 가능성이 더욱 커집니다. 예를 들어, 다음 조회를 고려해 보십시오.

```
SELECT C1, C2
  FROM TABLE1
 WHERE C1 = 'NEW YORK'
   AND C2 <= 10
```

그리고 여기에 C1과 C2에 대한 색인이 있다고 가정해 보십시오. 가능한 한 가지 액세스 플랜은 C1에 대한 색인을 사용하여 C1 = 'NEW YORK'인 모든 행을 검색한 다음 각각의 검색되는 행이 C2 <= 10인지 확인하는 것입니다. 또 다른 액세스

플랜은 C2에 대한 색인을 사용하여 C2 <= 10인 모든 행을 검색한 다음 각각의 검색되는 행이 C1 = 'NEW YORK'인지 확인하는 것입니다. 일반적으로, 위의 조회를 실행하는 데 드는 기본 비용은 행을 검색하는 비용이고, 따라서 최소한의 검색 횟수를 요구하는 플랜을 선택하는 것이 바람직합니다. 최상의 플랜을 선택하기 위해서는 각각의 술어를 충족시키는 행 수를 추정하는 것이 필요합니다.

분산 통계를 요청하지 않은 경우, 최적화 알고리즘은 하나의 컬럼에 대하여 두 번째 최상위 데이터 값(HIGH2KEY), 두 번째 최하위 데이터 값(LOW2KEY), 구별 값 수(COLCARD) 및 행 수(CARD)만을 유지보수합니다. 그런 다음 컬럼 내의 데이터 값의 빈도 수가 모두 같고 데이터 값은 상위 값과 하위 값(HIGH2KEY, LOW2KEY) 사이에 균등하게 분산되어 있다는 가정하에 등호(equality) 또는 범위 술어를 충족시키는 행 수가 추정됩니다. 특히 C1 = KEY 등호 술어를 충족시키는 행 수는 CARD/COLCARD로 추정되고, C1 BETWEEN KEY1 AND KEY2 범위 술어를 충족시키는 행 수는 다음과 같이 추정됩니다.

```
    KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

이러한 추정은 컬럼 내의 데이터 값 분산이 논리적으로 고르게 되어 있는 경우에만 정확합니다. 분산 통계를 사용할 수 없고, 데이터 값의 빈도가 서로 차이가 크거나, 데이터 값이 해당 범위(LOW_KEY,HIGH_KEY)의 몇몇 부분에 클러스터되어 있는 경우, 추정은 크기 순서로 정렬하는 것에서 종료되고, 최적화 알고리즘은 적절하지 못한 액세스 플랜을 선택할 수도 있습니다.

분산 통계를 사용할 수 있는 경우, 자주 사용되는 값 통계를 사용하여 등호 술어를 충족시키는 행 수를 계산하고, 자주 사용되는 값 통계 및 quantile을 사용하여 범위 술어를 충족시키는 행 수를 계산하여 위에서 설명한 오류를 크게 줄일 수 있습니다.

등호 술어에 대한 영향 예

우선 C1 = KEY 양식의 술어를 고려해 보십시오. KEY가 가장 자주 사용되는 값 N개 중 하나이면 최적화 알고리즘은 단순히 카탈로그에 저장되어 있는 KEY의 빈도를 사용합니다. KEY가 가장 자주 사용되는 값 N개 중 하나가 아니면 최적화 알

고리즘은(COLCARD - N) 자주 사용되지 않는 값이 균등한 분산을 갖는다는 가정하에 술어를 충족하는 행 수를 추정합니다. 즉, 행 수는 다음과 같이 추정됩니다.

```
CARD - NUM_FREQ_ROWS
--------------------          (2)
   COLCARD - N
```

여기서, NUM_FREQ_ROWS는 가장 자수 사용되는 값 N개 중 하나와 같은 값을 갖는 행의 수의 합입니다.

예를 들어, 데이터 값의 빈도가 다음과 같은 컬럼(C)을 고려해 보십시오.

데이터 값	빈도
1	2
2	3
3	40
4	4
5	1

가장 자주 사용되는 값(즉, N = 1)에만 근거한 자주 사용되는 값 통계가 가능하다고 가정해 보십시오. 이 컬럼에 대하여 CARD = 50이고, COLCARD = 5입니다. C = 3 술어에 대하여 정확하게 40행이 이를 충족시킵니다. 데이터가 고르게 분산되어 있다고 가정하면 해당 술어를 충족시키는 행 수는 50/5 = 10으로 추정되고, 오류는 -75%입니다. 자주 사용되는 값 통계를 사용하여 오류 없이 행 수는 40으로 추정됩니다.

마찬가지로, 2행이 술어 C = 1를 충족시킵니다. 자주 사용되는 값 통계가 없으면 술어를 충족시키는 행 수는 10으로 추정되고, 오류는 400%입니다. 다음 공식을 사용하여 추정 오류를 계산할 수 있습니다(백분율로).

```
   산출되는 행  - 실제 행
----------------------------  X 100
        실제 행
```

자주 사용되는 값 통계(N = 1)를 사용하면 최적화 알고리즘은 위의 공식 (2)를 사용하여 다음과 같은 값을 가진 행 수를 추정할 수 있습니다.

```
(50 - 40)
--------- = 3
 (5 - 1)
```

또한 오류는 다음에 표시된 크기 순서대로 감소됩니다.

```
 3 - 2
------- = 50%
  2
```

범위 술어를 충족시키는 행 수는 아래의 예에 표시된 대로 quantile을 사용하여 추정될 수 있습니다. 아래에 나와 있는 컬럼 (C)를 고려해 보십시오.

```
    C
-------
   0.0
   5.1
   6.3
   7.1
   8.2
   8.4
   8.5
   9.1
  93.6
 100.0
```

그리고 K = 1, 4, 7 및 10에 대하여 K-quantile이 사용 가능하다고 가정해 보십시오.

```
 K   K-quantile
---  ----------
 1      0.0
 4      7.1
 7      8.5
10    100.0
```

우선 술어 C <= 8.5를 살펴보십시오. 위에 제공된 데이터에 대하여 정확히 7개의 행이 이 술어를 충족시킵니다. 데이터가 고르게 분산되어 있다고 가정하고, KEY1을 LOW2KEY로 대체하여, 위의 공식 (1)을 사용하면 해당 술어를 충족시키는 행 수는 아래와 같이 추정될 수 있습니다.

```
 8.5 - 5.1
---------- x 10 *= 0
93.6 - 5.1
```

여기서 *=는 『거의 같음』을 의미합니다. 이 추정에서의 오류는 대략 -100%입니다.

quantile을 사용하면 동일한 이 술어(C <= 8.5)를 충족시키는 행 수는 8.5를 K-quantile 값 중 하나로 배치하고 대응 값 K, 즉 7을 추정치로 사용하여 추정됩니다. 이 경우, 오류는 0으로 감소합니다.

우선 술어 C <= 10를 살펴보십시오. 정확히 8개의 행이 이 술어를 충족시킵니다. 이전의 예와는 달리, 10이라는 값은 저장된 K-quantile 중 하나가 아닙니다. 데이터가 고르게 분산되어 있다고 가정하고, 공식 (1)을 사용하면 해당 술어를 충족시키는 행 수가 1로 추정되면 오류는 -87.5%입니다.

최적화 알고리즘은 quantiles을 사용하여 술어를 충족시키는 행 수를 r_1 + r_2로 추정합니다. 여기서 r_1은 술어 C <= 8.5를 충족시키는 행 수이고, r_2는 술어 C > 8.5 AND C <= 10을 충족시키는 행 수입니다. 위의 예에서처럼 r_1 = 7입니다. r_2를 산출하기 위해 최적화 알고리즘은 선형 보간법(linear interpolation)을 사용합니다.

```
        10 - 8.5
r_2 *= ---------- x (number of rows with value > 8.5 and <= 100.0)
        100 - 8.5
        10 - 8.5
r_2 *= ---------- x (10 - 7)
        100 - 8.5
        1.5
r_2 *= ---- x (3)
        91.5
r_2 *= 0
```

최종 추정값은 r_1 + r_2 *= 7이고, 오류는 -12.5%뿐입니다.

위의 예에서 quantile을 사용하여 추정의 정확도를 향상시킨 이유는 실제 데이터가 5 - 10의 범위에 "클러스터되어" 있기 때문인데, 표준 추정에 사용되는 공식은 데이터 값이 0에서 100 사이에 고르게 분산되어 있다고 가정합니다.

quantile을 사용하여 서로 다른 데이터 값의 빈도가 크게 차이가 나는 경우에도 정확도를 향상시킬 수 있습니다. 다음과 같은 빈도의 데이터 값을 갖는 컬럼을 고려해 보십시오.

데이터 값	빈도
20	5
30	5
40	15

```
  50          50
  60          15
  70           5
  80           5
```

K = 5, 25, 75, 95 및 100에 대하여 K-quantile이 가능하다고 가정해 보십시오.

```
 K     K-quantile
----   ----------
  5       20
 25       40
 75       50
 95       70
100       80
```

또한 세 개의 가장 발생 빈도가 높은 값에 근거한 자주 사용되는 값 통계를 사용할 수 있다고 가정해 보십시오.

그리고 술어가 C BETWEEN 20 AND 30과 같다고 고려해 보십시오. 데이터 값의 분산으로부터 정확히 10개의 행이 이 술어를 충족시키고 있음을 알 수 있습니다. 데이터가 고르게 분산되어 있다고 가정하고, 공식 (1)을 사용하면 해당 술어를 충족시키는 행 수는 다음과 같이 추정됩니다.

```
30 - 20
-------  x 100 = 25
70 - 30
```

이것은 오류가 150%에 달합니다.

자주 사용되는 값 통계 및 quantile을 사용하여, 술어를 충족시키는 행 수는 r_1 + r_2로 추정됩니다. 여기서 r_1은 술어 (C = 20)을 충족시키는 행 수이고, r_2은 술어 C > 20 AND C <= 30을 충족시키는 행 수입니다. 공식 (2)를 사용하면 r_1은 다음과 같이 추정됩니다.

```
100 - 80
-------- = 5
  7 - 3
```

선형 보간법을 사용하면 r_2는 다음과 같이 계산됩니다.

```
30 - 20
------- x (# rows with value > 20 and <= 40)
40 - 20
```

```
  30 - 20
= ------- x (25 - 5)
  40 - 20
= 10
```

최종 계산값은 15이고, 3의 인수씩 오류를 줄였습니다.

세부사항 색인 통계의 수집 및 사용

옵션으로, 최적화 알고리즘이 해당 색인을 사용하여 테이블에 액세스하는 비용을 더 잘 계산할 수 있도록 도움을 주는 색인에서 더 세부사항 통계를 수집할 수 있습니다. 이것은 두 방법 중 하나로 수행될 수 있습니다. 첫 번째는 RUNSTATS 명령에서 DETAILED절을 사용하는 것이고, 두 번째는 RUNSTATS API를 호출할 때 statsopt 매개변수에 대해 A, Y 또는 X를 지정하는 것입니다. DETAILED 통계인 PAGE_FETCH_PAIRS와 CLUSTERFACTOR는 테이블의 *크기*가 충분한 경우, 즉 약 25페이지인 경우에만 수집됩니다. 이 경우, CLUSTERFACTOR는 0과 1 사이의 값이고, CLUSTERRATIO는 -1(수집되지 않음)입니다. 25페이지보다 적은 테이블의 경우, CLUSTERFACTOR는 -1(수집되지 않음)이고, DETAILED 절이 해당 테이블에서 색인에 대해 지정된 경우라도 CLUSTERRATIO는 0과 100 사이입니다.

세부사항 색인 통계 이해

DETAILED 통계는 다른 버퍼 *크기*하에서 전체 색인 스캔이 수행된 경우, 테이블의 데이터 페이지에 액세스하는 데 필요한 물리적 I/O의 수를 간단한 방식으로 캡쳐하려 합니다. RUNSTATS는 색인의 페이지를 통해 스캔할 때 다른 버퍼 *크기*를 모델링하고 발생한 페이지 오류 빈도에 대한 추정을 수집합니다. 예를 들어, 사용 가능한 하나의 버퍼 페이지를 가지고, 색인에 의한 모든 새 페이지를 참조하면 페이지 오류를 일으키며, 더 나쁜 상황에서는 모든 행은 대부분 CARDINALITY I/O가 일어나는 다른 페이지를 참조할 수 있습니다. 다른 극단적 상황에서 버퍼의 *크기*가 커서 전체 테이블(최대 버퍼 *크기*가 될 수 있는)을 보유할 수 있는 경우, 테이블 NPAGES의 각 페이지를 정확히 한 번 물리적으로 읽습니다. 그러므로 물리적 I/O 수는 단일하고 버퍼 *크기*의 기능이 증가하지 않아야 합니다.

RUNSTATS는 구분적인 선형 곡선을 이런 추정치에 맞추고, 해당 곡선은 PAGE_FETCH_PAIRS 통계에서 11쌍의 문자열로 저장됩니다. 각 쌍의 첫 번째

값은 가정의 버퍼 *크기*이고, 각 쌍의 두 번째 값은 해당 색인 스캔에서 전체적으로 사용 가능한 *크기*의 버퍼를 가진 전체 색인 스캔을 페치하는 추정 물리적 I/O 수입니다. 그런 다음 최적화 알고리즘은 PAGE_FETCH_PAIRS 통계를 사용하여 해당 색인을 사용한 전체 또는 부분 색인 스캔의 데이터 페이지 페치에 대한 물리적 I/O 수를 추정합니다.

색인에 대해 PAGE_FETCH_PAIRS에 저장된 곡선 모양은 해당 색인의 클러스터링 작동에 따라 결정됩니다.

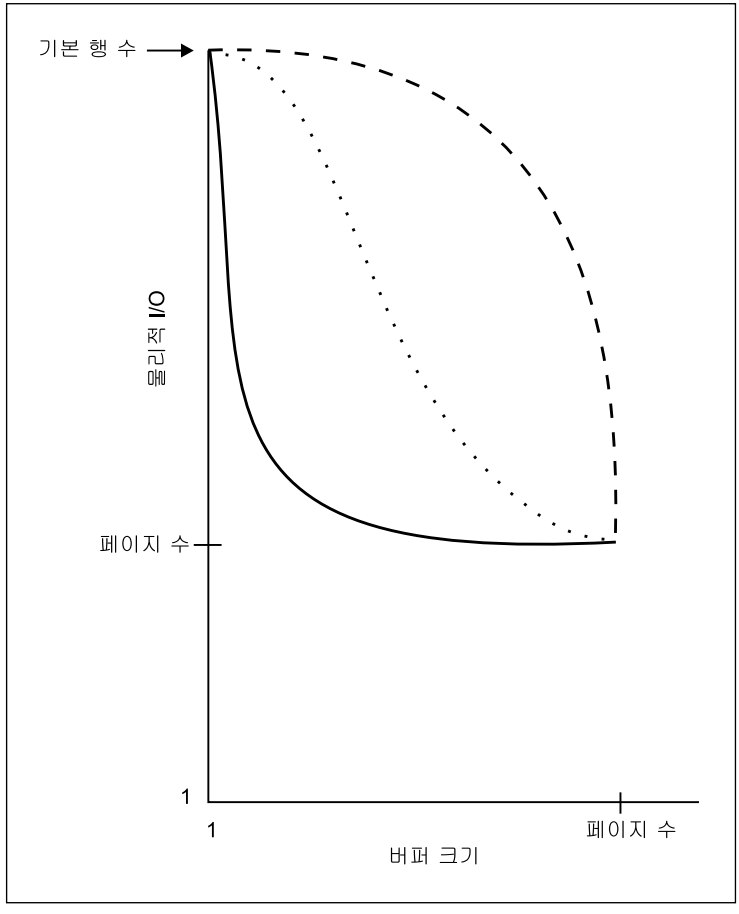


그림 11. 클러스터된 세 개의 곡선과 클러스터되지 않은 색인

다음과 같은 세 개의 곡선 유형이 있습니다.

1. 곡선 1(파선)은 재참조된 페이지가 버퍼에 있기 전의 테이블과 거의 같은 *크기*의 버퍼가 필요한 클러스터되지 않은 색인입니다. 이 곡선은 같은 페이지를 참조하는 상황이 색인의 키 값 전반에 걸쳐 널리 퍼져 있는 것을 나타내어 중간 *크기*의 버퍼로는 같은 페이지를 여러 번 재참조하는 데 충분하지 않습니다. 이런 상태는 최악의 시나리오이며, 제대로 수행되려면 최대의 버퍼 공간이 필요합니다. 최적화 알고리즘은 이러한 색인에서 목록 프리페치 액세스 전략을 사용하여 색인의 규정 키 값에 대해 데이터 페이지 액세스를 클러스터하는 시도를 합니다. 이 색인이 자주 사용되는 경우, 재구성에 가장 적합한 고려사항이 됩니다.
2. 곡선 2(실선)는 더욱 지역적으로 클러스터되지 않습니다. 매우 작은 버퍼의 경우, 곡선 1과 같이 클러스터되지 않습니다. 그러나 일단 적은 수의 버퍼 페이지가 최근에 참조된 데이터를 대부분 포함할 수 있게 되면 데이터 페이지 사용 비율은 크게 향상됩니다. 여기서는 어느 정도 다음과 같은 유리한 상황을 나타냅니다. 색인은 특별히 클러스터되지 않지만, 같은 데이터 페이지에 대한 참조로 색인의 키 값간에 서로 가까운 근접성을 갖게 되는 것입니다.
3. 곡선 3(점선)은 버퍼 증가시 균등한 비율로 향상되는 두 곡선 사이에 있습니다. 이 곡선은 클러스터되지 않은 색인에 있는 훨씬 흔한 경우이고, DETAILED 색인이 없을 때 최적화 알고리즘이 가정하는 사항을 나타냅니다.

세부사항 색인 통계를 언제 사용해야 합니까?

DETAILED 색인 통계는 조회가 색인에 없는 모든 컬럼을 참조할 경우에 사용해야 합니다. 또한 DETAILED 색인 통계는 다음 경우에 사용해야 합니다.

- 변하는 클러스터링 등급을 갖는 복수의 클러스터되지 않은 색인의 경우
- 클러스터링 등급이 키 값간에 일치하지 않는 경우
- 색인의 값이 일치하지 않도록 갱신되는 경우

사전 지식 없이 그리고 변하는 버퍼 *크기*에서 색인 스캔을 강제로 시도하지 않으면서, 모니터를 사용하여 발생하는 물리적 I/O를 관찰하지 않고, 이런 상황을 판별하는 것은 매우 어려운 일입니다. 이러한 상황의 발생 여부를 판별하는 가장 손쉬운 방법은 색인에서 DETAILED 통계를 수집하여 결과가 선형이 아닌 PAGE_FETCH_PAIRS인 경우, 수집한 통계를 보유하는 것입니다.

사용자가 갱신할 수 있는 카탈로그 통계

선택된 시스템 카탈로그 통계를 갱신할 수 있는 능력을 사용하여 다음과 같은 작업을 할 수 있습니다.

- 생산 시스템 통계를 사용하여 개발 시스템에서 조회의 성능을 모델링해 볼 수 있습니다.
- 『이 경우에는 어떻게 될까(what-if)』와 같은 조회 성능 분석을 수행할 수 있습니다.

최적화 알고리즘이 해당 조회에 대한 최상의 액세스 플랜을 찾는 데 장애가 되므로 생산 시스템에서는 통계를 갱신해서는 **안 됩니다.**

이러한 통계에 사용되는 컬럼의 값을 갱신하려면 SYSSTAT 스키마에 정의되어 있는 뷰에 대하여 SQL UPDATE문을 사용하십시오. 다음과 같은 경우에 통계를 갱신할 수 있습니다.

- 명시적 CONTROL 특권을 보유한 테이블. 이러한 테이블에 대해서는 컬럼과 색인에 대한 통계도 갱신할 수 있습니다.
- 연합 데이터베이스 시스템에서 명시적 CONTROL 특권을 보유한 별칭. 이러한 별칭에 대해서는 컬럼과 색인에 대한 통계도 갱신할 수 있습니다. 갱신은 지역 메타데이터에만 영향을 미침에 유의하십시오. (데이터 소스 테이블 통계는 변경되지 않습니다.) 이러한 갱신은 DB2 최적화 알고리즘에 의해 생성된 전역 액세스 전략에만 영향을 미칩니다.
- 소유하고 있는 사용자 정의 함수(UDF)(지침은 157 페이지의 『사용자 정의 함수(UDF)에 대한 통계 갱신』 참조)

또한 사용자 ID가 데이터베이스에 대하여 명시적인 DBADM 권한을 가지고 있는 경우, 즉 SYSCAT.DBAUTH 테이블에 DBADM 권한을 가지고 있는 것으로 기록되어 있는 경우, 이러한 통계를 갱신할 수 있습니다. 단지, DBADM 그룹에 속해 있는 것만으로는 이 권한을 명시적으로 제공하는 것은 아닙니다.

이러한 뷰를 사용하여 DBADM은 모든 사용자에 대한 통계 행을 볼 수 있습니다. DBADM 권한을 가지고 있지 않은 사용자는 CONTROL 특권이 부여된 오브젝트에 대한 통계를 가지고 있는 행만을 볼 수 있습니다.

다음에서는 EMPLOYEE 테이블에 대한 테이블 통계 갱신의 예를 보여줍니다.

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
     FPAGES = 1000,
     OVERFLOW = 2
WHERE TABSCHEMA = 'userid'
  AND TABNAME   = 'EMPLOYEE'
```

카탈로그 통계를 갱신할 때에는 주의를 기울여야 합니다. 임의로 갱신사항은 이후의 조회 성능에 심각한 영향을 미칠 수도 있습니다. 다음과 같은 방법을 사용하여 이러한 테이블에 적용한 갱신사항을 바꿀 수 있습니다.

- (작업 단위(UOW))가 확약되지 않은 것으로 가정하고) 변경이 일어난 작업 단위(UOW)를 구간 복원합니다.
- RUNSTATS 유틸리티를 사용하여 카탈로그 통계를 다시 계산하고 새로 고칠 수 있습니다.
- 카탈로그 통계를 갱신하여 통계가 수집되지 않았음을 나타냅니다. (예를 들어, 컬럼 NPAGES를 -1로 설정한다는 것은 페이지 수 통계가 수집되지 않았다는 것입니다.)
- 카탈로그 통계를 갱신 전에 가지고 있던 데이터로 바꿉니다. 이 방법은 159 페이지의『생산 데이터베이스 모델링』에 설명되어 있는 *db2look* 도구를 사용하여 변경 전의 통계를 캡쳐한 경우에만 가능합니다.

일부 경우, 최적화 알고리즘은 특수한 통계 값이나 값의 조합이 유효하지 않다고 판별하고, 기본값을 사용하여 경고를 발행합니다. 그러나 통계를 갱신할 때 대부분의 유효성 확인 작업이 수행되므로, 이러한 상황이 발생하는 것은 흔하지 않습니다.

추가 정보: 카탈로그 통계 갱신에 대한 자세한 내용은 다음을 참조하십시오.

- 152 페이지의『카탈로그 통계 갱신에 대한 규칙』
- 152 페이지의『테이블 및 별칭 통계 갱신 규칙』
- 153 페이지의『컬럼 통계 갱신 규칙』
- 154 페이지의『컬럼에 대한 분산 통계 갱신 규칙』
- 155 페이지의『색인 통계 갱신 규칙』
- 157 페이지의『사용자 정의 함수(UDF)에 대한 통계 갱신』
- 159 페이지의『생산 데이터베이스 모델링』

카탈로그 통계 갱신에 대한 규칙

카탈로그 통계를 갱신할 때 가장 중요한 일반 규칙은 여러 통계의 유효 값, 범위, 형식을 통계 뷰에 저장해야 한다는 점입니다. 또한 다양한 통계간의 관계의 일관성을 유지하는 것도 중요합니다.

예를 들어, SYSSTAT.COLUMNS의 COLCARD는 SYSSTAT.TABLES의 CARD(컬럼의 구별 값은 행 수보다 클 수 없음)보다 적어야 합니다. COLCARD를 100에서 25로 줄이고, CARD를 200에서 50으로 줄인다고 가정해 보십시오. SYSCAT.TABLES을 먼저 갱신할 경우, 오류가 발생합니다(CARD는 COLCARD보다 적어야 하므로). 올바른 순서는 SYSCAT.COLUMNS의 CARD를 먼저 갱신한 다음 SYSSTAT.TABLES의 CARD를 갱신하는 것입니다. COLCARD를 100에서 250으로 증가시키고, CARD를 200에서 300으로 증가시키는 경우에는 반대의 상황이 연출됩니다. 이 경우, CARD를 먼저 갱신한 다음 COLCARD를 갱신해야 합니다.

갱신된 통계와 다른 통계간에 충돌이 검출되면 오류가 발행됩니다. 그러나 충돌이 발생할 때마다 오류가 발행되는 것은 아닙니다. 일부 경우, 특히 관련된 두 통계가 서로 다른 카탈로그에 있을 경우에는 충돌을 검출하기 어려우며 오류로 보고하기도 힘듭니다. 이러한 이유로, 충돌을 일으키지 않도록 주의해야 합니다.

카탈로그 통계를 갱신하기 전에 검토해야 할 가장 일반적인 사항은 다음과 같습니다.

1. 숫자 통계 값은 -1이거나 0 이상의 값이어야 합니다.
2. 백분율로 표현되는 숫자 통계값(예: SYSSTAT.INDEXES의 CLUSTERRATIO)은 0에서 100 사이의 값이어야 합니다.

주: 행 유형의 경우, 테이블 레벨 통계 NPAGES, FPAGES 및 OVERFLOW는 서브테이블용으로는 갱신 가능하지 않습니다.

테이블 및 별칭 통계 갱신 규칙

SYSTAT.TABLES에서는 네 가지의 통계 값(CARD, FPAGES, NPAGES 및 OVERFLOW)만을 갱신할 수 있습니다. 다음 사항에 유의하십시오.

1. CARD는 해당 테이블에 대응하는 SYSSTAT.COLUMNS의 모든 COLCARD 값보다 커야 합니다.
2. CARD는 NPAGES보다 커야 합니다.
3. FPAGES는 NPAGES보다 커야 합니다.
4. NPAGES는 모든 색인의 PAGE_FETCH_PAIRS 컬럼에 있는 모든 "페치" 값 이하여야 합니다(이 통계가 색인에 해당되는 경우).
5. CARD는 모든 색인의 PAGE_FETCH_PAIRS 컬럼 모든 "페치" 값 이하일 수 없습니다(이 통계가 색인에 해당되는 경우).

연합 데이터베이스 시스템 내에서 작업할 때 원격 뷰를 통해 수동으로 별칭에 대한 통계를 제공/갱신할 경우에는 주의를 기울여야 합니다. 이 별칭이 리턴하는 행 수와 같은 통계 정보는 이 원격 뷰를 평가하는 데 필요한 실제 비용을 반영하지 않을 수도 있으므로, DB2 최적화 알고리즘이 잘못된 판단을 내리게 할 수 있습니다. 통계 갱신을 이용할 수 있는 경우로는 SELECT 목록에 적용되는 컬럼 함수가 없는 하나의 기본 테이블에 정의된 원격 뷰가 포함됩니다. 복잡한 뷰에는 각 조회를 조정해야 하는 복잡한 조정 프로세스가 필요할 수 있습니다. 그보다는 DB2 최적화 알고리즘이 뷰에 대한 비용을 좀더 정확히 계산하는 방법을 할 수 있도록 별칭을 통해 지역 뷰를 작성하는 것을 고려해 보십시오.

컬럼 통계 갱신 규칙

SYSSTAT.COLUMNS의 통계를 갱신할 때는 다음 지침을 따르십시오. 컬럼 분산 통계 갱신에 대한 자세한 내용은 154 페이지의 『컬럼에 대한 분산 통계 갱신 규칙』을 참조하십시오.

1. HIGH2KEY 및 LOW2KEY(SYSSTAT.COLUMNS의)는 다음의 규칙을 준수해야 합니다.
   - HIGH2KEY 및 LOW2KEY 값의 데이터 유형은 통계가 산출된 사용자 컬럼의 데이터 유형에 해당해야 합니다. HIGH2KEY는 VARCHAR 컬럼이므로 값을 따옴표로 묶어야 합니다. 예를 들어, HIGH2KEY를 INTEGER 사용자 컬럼에 대해 25로 설정하려면, 갱신문에 SET HIGH2KEY = '25'가 포함되어야 합니다.
   - HIGH2KEY, LOW2KEY 값의 길이는 33과 목표 컬럼 데이터 유형의 최대 길이 중에서 작은 것이어야 합니다.

- 해당 컬럼에 4 이상의 구별 값이 있는 경우, HIGH2KEY는 LOW2KEY 보다 커야 합니다. 컬럼 내의 구별 값이 3 미만인 경우, HIGH2KEY는 LOW2KEY와 같은 값이 될 수도 있습니다.

2. 컬럼의 기본 행 수(SYSSTAT.COLUMNS의 COLCARD 통계)는 해당 테이블의 기본 행 수(SYSSTAT.TABLES의 CARD 통계)보다 커야 합니다.
3. 컬럼에서 널(NULL) 수(SYSSTAT.COLUMNS의 NUMNULLS 통계)는 해당 테이블의 기본 행 수(SYSSTAT.TABLES의 CARD 통계)보다 커야 합니다.
4. 다음과 같은 데이터 유형을 갖는 컬럼에 대해서는 통계가 지원되지 않습니다. LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB.

컬럼에 대한 분산 통계 갱신 규칙

150 페이지의 『사용자가 갱신할 수 있는 카탈로그 통계』에서는 카탈로그 통계를 갱신하는 방법에 관한 일반 정보를 제공합니다. 컬럼 분산 통계 갱신을 시도하기 전에 해당 절을 참조하려고 할 수 있습니다.

카탈로그의 모든 통계에 일관성을 유지하려면 분산 통계를 갱신할 때 신중해야 합니다. 특히, 각각의 컬럼에 대해 자주 사용되는 값 통계 및 quantile에 대한 카탈로그 항목은 다음과 같은 제한조건을 충족시켜야 합니다.

- (SYSSTAT.COLDIST 카탈로그의) 자주 사용되는 값 통계. 제한조건은 다음과 같습니다.
  - VALCOUNT 컬럼의 값은 SEQNO 값이 증가하는 경우에 불변 또는 감소해야 합니다.
  - COLVALUE 컬럼의 값 수는 컬럼 구별 값의 수 이하여야 하며, 이는 SYSSTAT.COLUMNS 카탈로그 뷰의 COLCARD 컬럼에 저장되어 있습니다.
  - VALCOUNT 컬럼 값의 합계는 컬럼의 행 수 이하여야 하며, 이는 SYSSTAT.TABLES 카탈로그 뷰의 CARD 컬럼에 저장됩니다.
  - 대부분의 경우, COLVALUE 컬럼의 값은 컬럼의 두 번째 최상위 데이터 값과 두 번째 최하위 데이터 값 사이의 값이어야 하며, 이는 SYSSTAT.COLUMNS 카탈로그 뷰의 HIGH2KEY 및 LOW2KEY 컬럼에 각각 저

장되어 있습니다. HIGH2KEY보다 더 자주 사용되는 값이 하나 있고, LOW2KEY보다 덜 자주 사용되는 값이 하나 있습니다.

- (SYSSTAT.COLDIST 카탈로그의) quantile. 제한조건은 다음과 같습니다.
  - COLVALUE의 컬럼 값은 SEQNO 값이 증가하는 경우에 불변 또는 감소해야 합니다.
  - VALCOUNT 컬럼의 값은 SEQNO 값이 증가하는 경우에 증가해야 합니다.
  - COLVALUE 컬럼의 가장 큰 값은 VALCOUNT 컬럼에 컬럼의 행 수와 같은 해당 항목을 가지고 있어야 합니다.
  - 대부분의 경우, COLVALUE 컬럼의 값은 컬럼의 두 번째 최상위 데이터 값과 두 번째 최하위 데이터 값 사이의 값이어야 하며, 이는 SYSSTAT.COLUMNS 카탈로그 뷰의 HIGH2KEY 및 LOW2KEY 컬럼에 각각 저장되어 있습니다.

『R』 행이 있는 컬럼 C1에 대하여 분산 통계가 가능하고, 『(F x R)』 행이 있으며 상대적으로 같은 비율의 데이터 값을 갖는 컬럼에 대응하기 위하여 통계를 변경하려고 한다고 가정해 보십시오. F 인수를 사용하여 자주 사용되는 값 통계의 규모를 증가시키려면 VALCOUNT 컬럼의 각각의 입력항목에 F를 곱해야 합니다. 마찬가지로, F 인수를 사용하여 quantile의 규모를 증가시키려면 VALCOUNT 컬럼의 각 항목에 F를 곱해야 합니다. 이 규칙을 따르지 않으면 최적화 알고리즘이 잘못된 필터 인수를 사용하므로 조회를 수행할 때 예기치 않은 성능상의 문제점이 발생할 수 있습니다.

색인 통계 갱신 규칙

SYSSTAT.INDEXES의 통계를 갱신할 때에는 다음 규칙을 따르십시오.

1. PAGE_FETCH_PAIRS(SYSSTAT. INDEXES의)는 다음 규칙을 따라야 합니다.
   - PAGE_FETCH_PAIRS 통계의 개별 값은 일련의 공백 분리문자로 구분되어야 합니다.
   - PAGE_FETCH_PAIRS 통계의 개별 값은 10자릿수보다 길어서는 안 되고, 최대 정수 값(MAXINT = 2147483647)보다 작아야 합니다.
   - CLUSTERFACTOR가 0보다 큰 경우, 항상 유효한 PAGE_FETCH_PAIRS 값이 있어야 합니다.

- 하나의 PAGE_FETCH_PAIR 통계 안에는 정확히 11쌍의 값이 있어야 합니다.
- PAGE_FETCH_PAIRS의 버퍼 *크기* 항목의 값은 반드시 오름차순으로 되어야 합니다.
- PAGE_FETCH_PAIRS 항목의 버퍼 *크기* 값은 MIN(NPAGES, 524287) 보다 커서는 안됩니다. 여기서 NPAGES는 (SYSSTAT.TABLES의) 해당 테이블의 페이지 수입니다.
- PAGE_FETCH_PAIRS의 『페치』 항목은 내림차순이어야 하고, 각각의 『페치』 항목은 NPAGES보다 작아야 합니다. PAGE_FETCH_PAIRS 항목의 『페치』 *크기* 값은 해당 테이블의 CARD(기본 행 수) 통계보다 크지 않아야 합니다.
- (SYSSTAT.TABLES에 있는) 버퍼 *크기* 값이 두 개의 연속된 쌍에서 같은 값인 경우, 페이지 페치 값도 양쪽 쌍에서 같은 값이어야 합니다.

유효한 PAGE_FETCH_UPDATE는 다음과 같습니다.

```
PAGE_FETCH_PAIRS =
 '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
  260 300 280 300 300 300'
```

여기서

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. CLUSTERRATIO 및 CLUSTERFACTOR(SYSSTAT.INDEXES에 있는)는 다음의 규칙을 따라야 합니다.
   - CLUSTERRATIO의 유효 값은 -1이거나 0에서 100 사이의 값입니다.
   - CLUSTERFACTOR의 유효 값은 -1이거나 0에서 1 사이의 값입니다.
   - 적어도 CLUSTERRATIO 및 CLUSTERFACTOR 중 하나의 값은 항상 -1이어야 합니다.
   - CLUSTERFACTOR 값이 양수이면 반드시 유효한 PAGE_FETCH_PAIR 통계와 함께 있어야 합니다.
3. 다음 규칙은 FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD 및 FULLKEYCARD에 적용됩니다.

- FIRSTKEYCARD는 단일 컬럼 색인의 FULLKEYCARD와 같아야 합니다.
- FIRSTKEYCARD는 해당 컬럼의 COLCARD(SYSSTAT.COLUMNS에 있는)와 같아야 합니다.
- 이 색인 통계가 관련이 없을 경우, 이를 -1로 설정해야 합니다. 예를 들어, 3컬럼만 있는 색인을 가지고 있을 경우, FIRST4KEYCARD를 -1로 설정하십시오.
- 다중 컬럼 색인의 경우, 모든 통계가 관련되어 있으면 그 관계는 다음과 같아야 합니다.

```
FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
   <= FULLKEYCARD <= CARD
```

4. 다음과 같은 규칙이 SEQUENTIAL_PAGES 및 DENSITY에 적용됩니다.
   - SEQUENTIAL_PAGES의 유효 값은 -1이거나 0에서 NLEAF 사이의 값입니다.
   - DENSITY의 유효 값은 -1이거나 0에서 100 사이의 값입니다.

사용자 정의 함수(UDF)에 대한 통계 갱신

SYSSTAT.FUNCTIONS 카탈로그 뷰를 사용하여 사용자 정의 함수(UDF)에 대한 통계를 갱신할 수 있습니다. 이러한 통계를 사용할 수 있는 경우, 최적화 알고리즘은 이를 사용하여 다양한 액세스 플랜에 대한 비용을 계산합니다. 통계를 사용할 수 없는 경우 통계 컬럼의 값은 -1이 되며, 최적화 알고리즘은 간단한 UDF를 가정하는 기본값을 사용하게 됩니다.

다음 표에서는 UDF에 대하여 갱신할 수 있는 통계 컬럼에 관한 정보를 제공합니다.

표 13. 함수 통계(SYSCAT.FUNCTIONS 및 SYSSTAT.FUNCTIONS)

통계	설명
IOS_PER_INVOC	함수가 실행될 때마다 실행된 읽기/쓰기 요청의 예상 수
INSTS_PER_INVOC	함수가 실행될 때마다 실행된 머신 명령어의 예상 수
IOS_PER_ARGBYTE	입력 인수 바이트마다 실행된 읽기/쓰기 요청의 예상 수

표 13. 함수 통계(SYSCAT.FUNCTIONS 및 SYSSTAT.FUNCTIONS) (계속)

통계	설명
INSTS_PER_ARGBYTES	입력 인수 바이트마다 실행된 머신 명령어의 예상 수
PERCENT_ARGBYTES	함수가 실제로 처리하는 입력 인수 바이트의 예상 평균 백분율
INITIAL_IOS	함수가 호출된 처음 시간이나 마지막 시간에만 실행된 읽기/쓰기 요청의 예상 수
INITIAL_INSTS	함수가 호출된 처음 시간이나 마지막 시간에만 실행된 머신 명령어의 예상 수
CARDINALITY	테이블 함수에 의해 생성 행의 예상 수

예를 들어, 미국인의 구두 *크기*를 *그*에 대응하는 유럽인의 구두 *크기*로 변환하는 UDF(EU_SHOE)를 고려해 보십시오. (두 구두 *크기*는 UDT일 수 있습니다.) 이 UDF에 대해서는 다음과 같이 통계 컬럼을 설정해야 합니다.

- INSTS_PER_INVOC는 다음을 요구하는 머신 명령어 예상 수로 설정되어야 합니다.
  - EU_SHOE 호출
  - 출력 문자열 초기화
  - 결과 리턴
- INSTS_PER_INVOC는 입력 문자열을 유럽인의 구두 *크기*로 변환해야 하는 머신 명령어의 예상 수로 설정되어야 합니다.
- PERCENT_ARGBYTES는 전체 입력 문자열이 변환된 것을 나타내는 100으로 설정됩니다.
- INITIAL_INSTS, IOS_PER_INVOC, IOS_PER_ARGBYTE, INITIAL_IOS는 모두 0으로 설정되어야 하는데, 그 이유는 UDF는 계산만을 수행하기 때문입니다.

PERCENT_ARGBYTES는 전체 입력 문자열을 항상 처리하지는 않는 함수에 의해 사용됩니다. 예를 들어, UDF(LOCATE)를 고려하는데, 이것은 두 인수 입력을 승인하여 두 번째 인수 안에 첫 번째 인수가 먼저 발생하는 시작 위치를 리턴합니다. 첫 번째 인수 길이가 짧아서 두 번째 인수와 관련이 거의 없게 되고, 평균적으로 두 번째 인수의 75퍼센트가 검색되는 경우를 가정해 보십시오. 이 정보에

근거하여 PERCENT_ARGBYTES는 75로 설정되어야 합니다. 75%라는 평균값은 다음과 같은 추가 가정을 근거로 합니다.

- 검색될 전체 두 번째 인수를 발생하는 첫 번째 인수가 30분 동안 발견되지 않습니다.
- 첫 번째 인수는 두 번째 인수 안의 어디서나 똑같이 나타나며, 첫 번째 인수가 발견되면 (평균적으로) 검색될 두 번째 인수의 절반이 결과로 생깁니다.

INITIAL_INSTS 또는 INITIAL_IOS를 사용하여 함수가 호출된 처음 시간이나 마지막 시간에만 실행된 머신 명령어의 예상 수 또는 읽기/쓰기 요청의 예상 수를 기록할 수 있습니다. 예를 들어, 이것을 사용하여 스크래치패드 영역을 설정하는 비용을 기록할 수 있습니다.

사용자 정의 함수(UDF)에 의해 사용된 I/O와 명령어에 대한 정보를 얻기 위해서 운영 체제용 프로그래밍 언어 컴파일러에 의해 제공되거나 사용 가능한 모니터링 도구에 의해 제공된 출력을 사용할 수 있습니다.

생산 데이터베이스 모델링

때때로 생산 시스템 데이터의 부속 집합을 테스트 시스템에 갖고 있기를 원하는 경우가 있습니다. 그러나 테스트 시스템에 대한 카탈로그 통계와 구성 매개변수가 생산 시스템의 통계 및 매개변수와 일치하도록 갱신되어 있지 않으면 테스트 시스템용으로 선택된 액세스 플랜은 생산 시스템에서 선택될 것과 반드시 같지는 않습니다.

생산의 카탈로그 통계와 일치하는 테스트 데이터베이스의 카탈로그 통계를 작성하는 데 필요한 갱신문을 생성하기 위해 생산 데이터베이스에 대해 수행될 수 있는 생산성 도구인 *db2look*이 제공됩니다. 이러한 갱신문은 모방 모드(`-m` 옵션)에서 *db2look*을 사용하여 생성될 수 있습니다. 이 경우, *db2look*은 생산 데이터베이스의 카탈로그 통계를 모방하는 데 필요한 모든 명령문이 들어 있는 명령 프로세서 스크립트를 생성합니다. 이는 테스트 환경에서 Visual Explain을 통해 SQL문을 분석할 때 유용합니다.

db2look -e 명령으로 DDL문을 발췌하여 테이블, 뷰, 색인 및 데이터베이스 내의 다른 오브젝트를 포함한 데이터베이스 데이터 오브젝트를 재작성할 수 있습니다.

다른 데이터베이스에 대해 이 명령으로부터 작성된 명령 프로세서 스크립트를 수행하여 데이터베이스를 재작성할 수 있습니다. -e 옵션을 -m 옵션과 함께 사용할 수 있습니다.

테스트 시스템에서 db2look에 의해 생성된 갱신문을 수행한 후, 테스트 시스템은 생산에서 생성될 액세스 플랜의 유효성을 확인하는 데 사용될 수 있습니다. 최적화 알고리즘은 테이블 공간의 유형과 구성을 사용하여 I/O 비용을 계산하기 때문에 테스트 시스템은 같은 테이블 공간 기하학 또는 레이아웃을 가지고 있어야 합니다. 즉, 같은 유형의 같은 수의 컨테이너입니다. SMS 또는 DMS가 해당합니다.

db2look 도구는 *bin* 서브디렉토리에 들어 있습니다.

이 생산성 도구를 사용하는 방법에 관한 자세한 정보를 보려면 명령행에 다음 명령을 입력하십시오.

```
db2look -h
```

또한 이 도구에 대한 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

제어 센터에서는 『Generate SQL - Object Name』이라는 *db2look* 유틸리티에 대한 인터페이스를 제공합니다. 제어 센터를 사용하면 유틸리티로부터 생성된 결과 파일이 스크립트 센터에 통합됩니다. 제어 센터로부터 *db2look* 명령을 스케줄할 수 있습니다. 한 가지 차이점은 *db2look* 명령을 사용하면 하나의 호출에서 최대 30개의 테이블을 분석할 수 있는 반면, 제어 센터를 사용하면 하나의 테이블 분석만을 수행할 수 있다는 것입니다. LaTex 및 Graphical 출력은 제어 센터에서는 지원되지 않는다는 점에 유의하십시오.

OS/390 데이터베이스에 대해 db2look 유틸리티를 수행할 수도 있습니다. db2look 유틸리티는 OS/390 오브젝트에 대한 DDL 및 UPDATE 통계 명령문을 발췌합니다. 이 유틸리티는 OS/390 오브젝트를 발췌하고 DB2 Universal Database(UDB) 데이터베이스에서 이 오브젝트를 재작성하려고 할 경우에 매우 유용합니다. db2look 유틸리티에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

DB2 UDB 통계와 OS/390 통계 사이에는 약간의 차이가 있습니다. db2look 유틸리티는 적용 가능할 경우 OS/390용 DB2에서 DB2 UDB로의 적절한 변환을 수

행하고 OS/390용 DB2 대응부가 존재하지 않는 DB2 UDB 통계를 기본값(-1)으로 설정합니다. 다음은 db2look 유틸리티가 OS/390용 DB2 통계를 DB2 UDB 통계에 맵핑하는 방식입니다. 아래 설명에서 『UDB_x』는 DB2 UDB 통계 컬럼을 나타내고, 『S390_x』는 OS/390용 DB2 통계 컬럼을 나타냅니다.

1. 테이블 레벨 통계.

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   S390_FPAGES는 없습니다. 그러나 OS/390용 DB2에는 테이블의 행을 포함하는 활동 중인 테이블 공간 페이지의 백분율을 나타내는 PCTPAGES라고 하는 다른 통계가 있습니다. 그러므로 다음과 같이 S390_NPAGES 및 S390_PCTPAGES를 기초로 UDB_FPAGES를 계산할 수 있습니다.

   ```
   UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
   ```

   UDB_OVERFLOW에 맵핑할 S390_OVERFLOW가 없습니다. 그러므로 db2look 유틸리티는 이를 기본값으로 설정합니다.

   ```
   UDB_OVERFLOW=-1
   ```

2. 컬럼 레벨 통계.

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   UDB_AVGCOLLEN에 맵핑할 S390_AVGCOLLEN이 없으므로 db2look 유틸리티는 이를 기본값으로 설정합니다.

   ```
   UDB_AVGCOLLEN=-1
   ```

3. 색인 레벨 통계.

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS
   UDB_FIRSTKEYCARD= S390_FIRSTKEYCARD
   UDB_FULLKEYCARD = S390_FULLKEYCARD
   UDB_CLUSTERRATIO= S390_CLUSTERRATIO

OS/390 대응부가 없는 다른 통계는 기본값으로 설정됩니다. 다음과 같습니다.

```
UDB_FIRST2KEYCARD = -1
UDB_FIRST3KEYCARD = -1
UDB_FIRST4KEYCARD  = -1
UDB_CLUSTERFACTOR = -1
UDB_SEQUENTIAL_PAGES = -1
UDB_DENSITY = -1
```

4. 컬럼 분산 통계.

   OS/390용 DB2 SYSIBM.SYSCOLUMNS에는 두 가지 유형의 통계가 있습니다. 자주 사용되는 값에 대해서는 『F』를, 기본 행 수(cardinality)에 대해서는 『C』를 입력하십시오. 유형 『F』의 항목만 UDB용 DB2에 적용할 수 있으며, 이들 항목은 고려될 항목입니다.

   ```
   UDB_COLVALUE = S390_COLVALUE
   UDB_VALCOUNT = S390_FrequencyF * S390_CARD
   ```

   또한 OS/390용 DB2 SYSIBM.SYSCOLUMNS에는 SEQNO 컬럼이 없지만, 이는 UDB용 DB2에 필요합니다. 따라서 db2look는 자동으로 이를 생성합니다.

부속 요소 통계

부속 요소 통계를 수집 및 사용하는 옵션이 제공됩니다. 이것은 데이터가 일련의 부속 필드 또는 공백으로 분리된 부속 요소 양식의 구조를 가질 때 데이터 컬럼의 내용에 대한 통계입니다.

예를 들어, 각 행이 문서를 설명하는 테이블 DOCUMENTS가 데이터베이스에 포함되어 있고 텍스트 검색 목적으로 이 문서와 관련한 목록을 포함하는 KEYWORDS라는 컬럼이 DOCUMENTS에 있다고 가정해 보십시오. KEYWORDS의 값은 다음과 같습니다.

```
'database simulation analytical business intelligence'
'simulation model fruitfly reproduction temperature'
'forestry spruce soil erosion rainfall'
'forest temperature soil precipitation fire'
```

이 예에서 각 컬럼 값은 5개의 부속 요소로 구성되며 각 요소는 공백 하나로 다른 요소와 구분되는 단어(키워드)입니다.

% match_all 문자를 사용하여 이러한 컬럼에 LIKE 술어를 지정하는 조회의 경우:

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%simulation%'
```

최적화 프로그램이 이름별로 컬럼의 부속 요소 구조에 대한 기본 통계를 알고 있으면 많은 이점이 있습니다.

SUB_COUNT
부속 요소의 평균 수

SUB_DELIM_LENGTH
이 문맥에서 분리문자가 있는 곳에 각 부속 요소를 구분하는 각 분리문자의 평균 길이는 하나 이상의 연속 공백 문자입니다.

각 분리문자가 단일 공백 문자이므로 KEYWORDS 컬럼 예에서 SUB_COUNT는 5이고, SUB_DELIM_LENGTH는 1입니다.

시스템 관리자는 DB2_LIKE_VARCHAR 레지스트리 변수에 대한 확장 수단으로 이들 통계 사용과 콜렉션을 제어합니다. 이 레지스트리 변수는 DB2 UDB 최적화 알고리즘이 다음 양식의 술어를 처리하는 방식에 영향을 줍니다.

```
COLUMN LIKE '%xxxxxx'
```

여기서 *xxxxxx*는 문자열입니다. 즉, % 문자로 시작하는 검색 값의 LIKE 술어입니다. (% 문자로 끝날 수도, 그렇지 않을 수도 있습니다.) 이것을 이후로는 ″와일드카드 LIKE 술어″라고 합니다. 모든 술어에 대해 최적화 알고리즘은 술어와 일치하는 행 수를 추정해야 합니다. 와일드카드 LIKE 술어의 경우, 최적화 알고리즘은 일치된 COLUMN이 일련의 요소 병합 구조로 전체 컬럼을 형성하고 선행 및 후행 % 문자를 제외한 문자열의 길이에 기반한 각 요소의 길이를 추정한다고 가정합니다. 새 구문은 다음과 같습니다.

```
   db2set  DB2_LIKE_VARCHAR=[Y|N|S|num1][,Y|N|num2]
여기서,
   - 첫 번째 용어(쉼표 앞)는 다음을 의미하되,
     양수 부속 요소 통계가 없는 컬럼에 대해서만이
     아닙니다.
     S                DB2 버전 2에서 사용된 대로 알고리즘을 사용합니다.
     N                고정 길이 부속 요소 알고리즘 사용
     Y    (기본값)    변수 길이 부속 요소 알고리즘 사용
                      (알고리즘 매개변수에 대한 기본값으로)
     num1             가변 길이 부속 요소 알고리즘 사용
                      및 num1을 알고리즘 매개변수로 사용
```

```
- 두 번째 조건          (쉼표 뒤) 의미:
   N    (기본값)       부속 요소 통계를 수집하거나 사용하지 마십시오.
   Y                   부속 요소 통계를 수집하십시오.
                       양수 부속 요소 통계를 갖는 컬럼의 경우,
                       알고리즘 매개변수의 기본값과 함께 이러한 통계를
                       사용하는 변수 길이 부속 요소 알고리즘을
                       사용합니다.
   num2                부속 요소 통계를 수집하십시오.
                       양수 부속 요소 통계를 갖는 컬럼의 경우,
                       알고리즘 매개변수로 num2와 함께 이러한 통계를
                       사용하는 변수 길이 부속 요소 알고리즘을
                       사용합니다.
```

DB2_LIKE_VARCHAR이 첫 번째 조건만 포함하는 경우, 부속 요소 통계는 수집되지 않고 이전에 수집된 통계는 무시됩니다. 지정된 값은 최적화 알고리즘이 전과 같은 방식으로 와일드카드 LIKE 술어의 선택성을 계산하는 방법에 영향을 줍니다.

- 값이 S인 경우, 최적화 알고리즘은 DB2 버전 2에 사용된 것과 동일한 알고리즘을 사용하고, 이 알고리즘을 부속 요소 모델이라고 가정하지 않습니다.
- 값이 N인 경우, 최적화 프로그램은 부속 요소 모델을 가정하는 알고리즘을 사용하고, COLUMN이 변수 길이로 정의되어 있더라도 고정 길이라고 가정합니다.
- 값이 Y(기본값) 또는 부동 소수점 상수인 경우, 최적화 프로그램은 부속 요소 모델을 가정하는 알고리즘을 사용하고, COLUMN이 가변 길이로 정의되어 있으면 가변 길이라고 인식합니다. 또한 데이터가 아닌 조회 자체에서 부속 요소 통계를 추정합니다. 이 알고리즘은 요소의 길이가 % 문자로 둘러싸인 문자열보다 얼마나 긴지를 지정하는 매개변수("알고리즘 매개변수")를 포함합니다.
- 값이 Y인 경우, 최적화 알고리즘은 알고리즘 매개변수에 대해 기본값 1.9를 사용합니다.
- 값이 부동 소수점 상수인 경우, 최적화 알고리즘은 알고리즘 매개변수에 지정된 값을 사용합니다. 이 상수는 0에서 6.2 내에 있어야 합니다.

DB2_LIKE_VARCHAR 값이 두 조건을 포함하고 두 번째 조건이 Y 또는 부동 소수점 상수인 경우 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 유형의 1바이트 문자 세트 문자열 컬럼에 대한 부속 요소 통계는 RUNSTATS 작업 도중에 수집되고 와일드카드 LIKE 술어를 포함하는 조회의 컴파일 도중에 사용됩니다. 최적화 알고리즘은 부속 요소 모델을 추정하고 알고리즘 매개변수와 함

| 께 SUB_COUNT 및 SUB_DELIM_LENGTH 통계를 사용하는 알고리즘을 사
| 용하여 술어의 선택성을 계산합니다. 알고리즘 매개변수는 추론 알고리즘이 지정
| 된 방식과 동일한 방식으로 지정됩니다.

- 값이 Y인 경우, 최적화 알고리즘은 알고리즘 매개변수에 대해 기본값 1.9를 사
  용합니다.
- 값이 부동 소수점 상수인 경우, 최적화 알고리즘은 알고리즘 매개변수에 지정된
  값을 사용합니다. 이 상수는 0에서 6.2 내에 있어야 합니다.

| 컴파일 도중 최적화 알고리즘은 부속 요소 통계가 조회에 포함된 컬럼에 수집되지
| 않았음을 발견한 경우 "추론" 부속 요소 알고리즘을 사용합니다. 즉,
| DB2_LIKE_VARCHAR의 첫 번째 조건이 지정된 경우에만 사용됩니다. 따라서
| 최적화 알고리즘에 의해 사용될 부속 요소 통계의 순서에 대해
| DB2_LIKE_VARCHAR의 두 번째 조건이 RUNSTATS 도중, 그리고 컴파일 도
| 중에 설정되어야 합니다.

| 부속 요소 통계의 값은 SYSIBM.SYSCOLUMNS를 조회하여 볼 수 있습니다. 예
| 를 들면, 다음과 같습니다.

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

| SUB_COUNT 및 SUB_DELIM_LENGTH 컬럼은 SYSSTAT.COLUMNS 통
| 계 뷰에 나타나지 않으므로 갱신될 수 없습니다.

| **주:** RUNSTATS는 이 옵션이 사용되면 더 오래 실행할 수 있습니다. 예를 들어,
| DETAILED 및 DISTRIBUTION 옵션이 사용되지 않으면 RUNSTATS는
| 15%와 40% 사이에서 5개의 문자 컬럼이 있는 테이블에 더 오래 실행할 수
| 있습니다. DETAILED 또는 DISTRIBUTION 옵션이 지정되어 있으면 백분
| 율 오버헤드는 오버헤드의 절대 양이 동일한 경우에도 적습니다. 이 옵션 사
| 용을 고려 중이면 조회 성능의 향상에 대한 이 오버헤드를 평가하십시오.

제6장 SQL 컴파일러의 이해

SQL 조회가 컴파일될 때『최상의』액세스 플랜이 시스템 카탈로그에서 실행되거나 저장되기 전에 여러 단계가 수행됩니다.

파티션된 데이터베이스 환경에서 SQL 컴파일러가 SQL 조회에서 수행한 모든 작업은 연결하는 데이터베이스 파티션에서 발생합니다. 수행되기 전에 컴파일된 조회는 데이터베이스의 모든 데이터베이스 파티션에 전송됩니다.

다음 주제에서는 SQL 컴파일러가 수행하는 단계에 대한 자세한 정보를 제공합니다.

- SQL 컴파일러의 개요
- SQL 컴파일러에 의한 조회 재작성
- 조작 병합
- 조작 이동
- 술어 변환
- 데이터 액세스 개념 및 최적화
- 파티션 내 병렬 처리에 대한 최적화 전략
- 자동 요약 테이블
- 연합 데이터베이스 조회 컴파일러 단계

다음 절에서는 컴파일러가 생성하는 결과에 영향을 미칠수 있는 컴파일 외적인 인수에 대한 정보를 제공합니다.

- 45 페이지의『제3장 응용프로그램 고려사항』
- 101 페이지의『제4장 환경상의 고려사항』
- 125 페이지의『제5장 시스템 카탈로그 통계』

243 페이지의『제7장 SQL Explain 기능』에서는 SQL 컴파일러가 선택한 액세스 플랜을 조사하는 방법에 대하여 설명합니다.

© Copyright IBM Corp. 1993, 2001

SQL 컴파일러의 개요

SQL 컴파일러는 사용자가 실행할 수 있는 액세스 플랜을 생성하기 전에 몇 단계의 작업을 수행합니다. 그림12에서는 이들 단계를 보여줍니다.

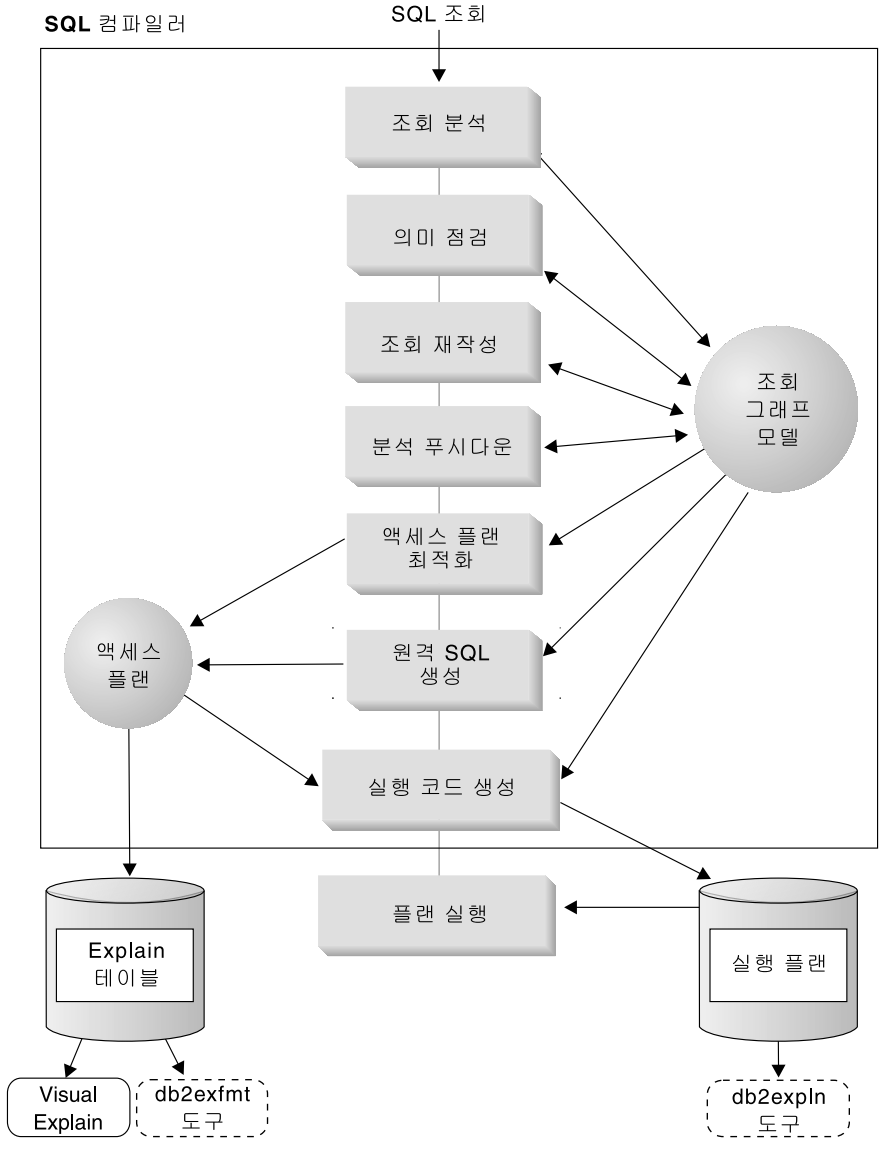


그림 12. SQL 컴파일러가 수행하는 단계

이 도표에서는 **조회 그래프 모델**이 SQL 컴파일러의 주요 구성요소임을 보여줍니다. *조회 그래프 모델*은 내부, 메모리 내의 데이터베이스로서, 아래에 설명된 대로 조회 컴파일 프로세스 전반에 걸쳐 조회를 나타내는 데 사용됩니다.

- **조회 분석**

  SQL 컴파일러의 첫 번째 타스크는 구문의 유효성을 확인하기 위해 SQL 조회를 분석하는 것입니다. 어떤 구문 오류가 검출된 경우, SQL 컴파일러는 처리를 중단하고 SQL 문을 컴파일하려고 한 응용프로그램으로 적절한 SQL 오류를 리턴합니다. 분석이 완료되면 조회의 내부 표현이 작성됩니다.

- **의미 점검**

  컴파일러의 두 번째 타스크는 명령문 부분들 사이에 불일치가 있는지 확인하는 것입니다. 이러한 간단한 의미 점검 예로는 YEAR 스칼라 함수에 대하여 지정된 컬럼의 데이터 유형이 날짜 시간 데이터 유형임을 확인하는 것입니다. 또한 이 두 번째 단계에서 컴파일러는 참조 제한조건, 테이블 점검 제한조건, 트리거와 뷰 등의 작동 의미를 조회 그래프 모델에 추가합니다.

  조회 그래프 모델은 조회 블록, 부속 조회, 상관, 파생된 테이블, 표현식, 데이터 유형, 데이터 유형 변환, 코드 페이지 변환, 파티션 키를 포함한 조회에 대한 모든 의미를 수록하고 있습니다.

- **조회 재작성**

  SQL 컴파일러의 세 번째 단계는 조회 그래프 모델에서 제공하는 전역 의미를 사용하여 조회를 더 쉽게 최적화할 수 있는 양식으로 변환하는 작업입니다. 예를 들어, 컴파일러는 술어를 이동시키고, 적용되는 레벨을 변경하며, 잠재적으로 조회 성능을 향상시킬 수 있습니다. 이러한 유형의 조작 이동을 *일반 술어 푸시다운*이라고 합니다. 자세한 내용은 171 페이지의 『SQL 컴파일러에 의한 조회 재작성』을 참조하십시오.

  파티션된 데이터베이스 환경에서 작업할 때 일부 조회 조작은 다음 작업이 관련된 것처럼 보다 강도 높게 이루어집니다.

  – 총계

  – 행의 재분산

  – 상관 부속 조회

  상관 부속 조회는 부속 조회를 벗어나는 테이블 컬럼에 대한 참조가 포함되는 부속 조회입니다.

  이러한 환경에서는 일부 조회의 경우 조회 재작성의 일부로서 상관 해제가 발생할 수 있습니다.

전송된 조회는 조회 그래프 모델에 저장됩니다.

- **분석 푸시다운(연합 데이터베이스)**

  이 단계의 주요 타스크는 조작을 데이터 소스에서 원격으로 평가할 수 있는지(『푸시다운하여』) 여부를 DB2 최적화 알고리즘에 권고하는 것입니다. 이러한 유형의 푸시다운 활동은 데이터 소스 조회에 국한되며 일반 술어 푸시다운 조작에 대한 확장입니다.

  이 단계는 연합 데이터베이스 조회를 실행하고 있지 않으면 생략됩니다. 자세한 내용은 226 페이지의 『분석 푸시다운』을 참조하십시오.

- **액세스 플랜 최적화**

  SQL 컴파일러의 SQL 최적화 알고리즘 부분은 입력으로 조회 그래프 모델을 사용하여, 사용자의 요청을 충족시킬 수 있는 여러 가지 대체 실행 플랜을 생성합니다. 최적화 알고리즘은 테이블, 색인, 컬럼 및 함수 등에 대한 통계를 사용하여 각각의 대체 플랜을 실행하는 데 드는 비용을 계산하고, 가장 적은 비용이 드는 플랜을 선택합니다. 최적화 알고리즘은 조회 그래프 모델을 사용하여 조회 의미를 분석하고, 색인, 기본 테이블, 파생된 테이블, 부속 조회, 상관 및 순환을 포함한 광범위한 인수에 대한 정보를 얻습니다.

  최적화 알고리즘 부분은 다른 유형의 푸시다운 조작을 고려할 수도 있습니다. *총계 및 정렬*은 이러한 조작의 평가를 데이터베이스 관리 서비스 구성요소로 푸시다운함으로써 성능을 향상시킬 수 있습니다. 자세한 내용은 218 페이지의 『총계 및 정렬 푸시다운 연산자』를 참조하십시오.

  최적화 알고리즘은 페이지 *크기*를 선택할 때 *크기*가 다른 버퍼가 있는지를 고려합니다. 대칭 멀티프로세서(SMP) 환경에서 선택된 플랜의 조회 내 병렬 처리 가능성을 증진하는 능력뿐 아니라, 환경은 파티션된 데이터베이스를 포함한다는 사실도 고려됩니다. 이 정보는 최적화 알고리즘이 조회에 대한 최상의 액세스 플랜을 선택하는 것을 돕는 데 사용됩니다. 자세한 내용은 183 페이지의 『데이터 액세스 개념 및 최적화』를 참조하십시오.

  SQL 컴파일러의 이번 단계에서의 출력은 『액세스 플랜』입니다. 이 액세스 플랜은 Explain 테이블에 캡쳐된 정보에 대한 근거를 제공합니다. 액세스 플랜을 생성하는 데 사용되는 정보는 Explain 스냅샷을 통해 캡쳐될 수 있습니다. (Explain에 대한 자세한 내용은 243 페이지의 『제7장 SQL Explain 기능』을 참조하십시오.)

- **원격 SQL 생성(연합 데이터베이스)**

  DB2 최적화 알고리즘가 선택한 최종 플랜은 원격 데이터 소스에서 작동할 수 있는 단계 세트로 구성될 수 있습니다. 각 데이터 소스에서 수행될 조작의 경우, 원격 SQL 생성 단계는 데이터 소스 SQL dialect를 기본으로 하여 효율적인 SQL문을 작성합니다.

이 단계는 연합 데이터베이스 조회를 실행하고 있지 않으면 생략됩니다. 자세한 내용은 235 페이지의 『원격 SQL 생성 및 전역 최적화』를 참조하십시오.

- **『실행』 코드 생성**

  SQL 컴파일러의 마지막 작업 단계는 *액세스 플랜*과 조회 그래프 모델을 사용하여 조회에 대한 실행 액세스 플랜 또는 섹션을 작성하는 것입니다. 이 코드 생성 단계에서는 조회에 대해 한 번만 계산되어야 하는 표현식의 반복 실행을 피하기 위해 조회 그래프 모델의 정보를 사용합니다. 이러한 최적화가 가능한 예로는 코드 페이지 변환 및 호스트 변수 사용과 같은 경우가 있습니다.

  정적 SQL의 액세스 플랜에 대한 정보는 시스템 카탈로그 테이블에 저장됩니다. 패키지가 실행되면 데이터베이스 관리 프로그램은 시스템 카탈로그 테이블에 저장된 정보를 사용하여 데이터에 액세스할 방법을 결정하고 조회 결과를 제공합니다. *db2expln* 도구에 의해 사용되는 정보가 바로 이 정보입니다. (Explain에 대한 자세한 내용은 243 페이지의 『제7장 SQL Explain 기능』을 참조하십시오.)

좋은 성능을 기대하는 경우, RUNSTATS는 조회에서 사용된 테이블에서 주기적으로 수행되는 것이 좋습니다. 최적화 알고리즘은 데이터의 특성에 대한 관련 통계 정보를 갖게 됩니다. 새로운 통계를 사용하면 사용자의 응용프로그램을 리바인드할 필요가 있습니다. RUNSTATS가 수행되지 않는(또는 최적화 알고리즘이 RUNSTATS가 비어 있거나 거의 빈 테이블에서 수행된다고 생각하는) 경우, 최적화 알고리즘은 기본값을 사용하거나 디스크에 테이블을 저장하는 데 사용된 파일 페이지 수(FPAGES)에 근거하는 통계를 도출하려는 시도를 합니다.

SQL 컴파일러에 의한 조회 재작성

SQL 컴파일러에는 보다 쉽게 최적화할 수 있는 양식으로 SQL문을 변환하여 결과적으로 선택된 액세스 경로를 향상시킬 수 있는 조회 재작성 단계가 들어 있습니다. 조회 재작성은 부속 조회나 조인이 많이 들어 있는 조회와 같이 매우 복합 조회의 경우 특히 유용합니다. 조회 생성 프로그램 도구는 종종 이런 유형의 매우 복합 조회를 작성합니다.

최적화 클래스를 변경하여 SQL문에 적용되는 조회 재작성 규칙의 수에 영향을 줄 수 있습니다(74 페이지의 『최적화 클래스 조정』 참조).

Explain 기능 또는 Visual Explain의 사용을 통해 조회 재작성 결과의 일부를 볼 수 있습니다.

SQL 컴파일러가 수행하는 재작성에는 다음 세 가지 주요 범주가 있습니다.

- 조작 병합
- 조작 이동
- 술어 변환

조작 병합

SQL 컴파일러는 가장 적은 수의 조작, 특히 SELECT 조작을 갖도록 조회를 구성하려고 할 때 조회 조작을 병합하기 위한 조회를 재작성합니다. 다음 예는 SQL 컴파일러가 병합할 수 있는 일부 조작을 설명하기 위해 제공됩니다.

- 예 - 뷰 병합

  SELECT문에서 뷰를 사용하면 테이블의 조인 순서를 제한할 수 있으며 테이블의 중복 조인이 발생할 수도 있습니다. 조회 재작성 중에 뷰를 병합하여 이러한 제한사항을 없앨 수 있습니다.

- 예 - 부속 조회에서 조인으로의 변환

  최적화 알고리즘이 SELECT문에서 부속 조회를 찾을 경우, 테이블의 순서 처리에 대한 선택으로 제한될 수 있습니다.

- 예 - 중복 조인 제거

  조회 재작성 중에 최적화될 SELECT문을 더 단순화할 수 있도록 중복 조인을 제거할 수 있습니다.

- 예 - 공유 총계

  다른 함수를 사용할 때 조회를 재작성하면 수행되어야 하는 계산의 수를 줄일 수 있습니다.

예 - 뷰 병합

EMPLOYEE 테이블의 다음과 같은 두 개의 뷰에 액세스한다고 가정해 보십시오. 하나는 교육 수준이 높은 사원을 보여주고, 다른 하나는 수입이 $35,000보다 많은 사원을 보여줍니다.

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
     FROM   EMPLOYEE
 WHERE EDLEVEL > 17
```

```
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
     FROM   EMPLOYEE
 WHERE SALARY > 35000
```

이제 교육 수준이 높고 수입이 $35,000보다 많은 사원을 나열하는 다음의 조회를 수행한다고 가정해 보십시오.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

조회 재작성 중에 이들 두 개의 뷰는 병합되어 다음 조회를 작성하게 됩니다.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

사용자가 작성한 SELECT문을 포함하는 두 개의 뷰로부터 SELECT문을 병합하면 최적화 알고리즘은 액세스 플랜을 선택할 때보다 많은 선택사항을 고려할 수 있습니다. 또한 병합된 두 개의 뷰가 동일한 기본 테이블을 사용하는 경우, 추가 재작성 작업이 174 페이지의 『예 - 중복 조인 제거』에 설명된 대로 수행될 수 있습니다.

예 - 부속 조회에서 조인으로의 변환

SQL 컴파일러는 다음의 부속 조회를 포함하는 조회를 가져옵니다.

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
     FROM   EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
         FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

그리고 이것을 다음과 같은 양식의 조인 조회로 변환합니다.

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

일반적으로 조인은 부속 조회보다 훨씬 더 효율적으로 실행됩니다.

예 - 중복 조인 제거

종종 불필요한 조인을 포함하는 조회가 작성되거나 생성될 수 있습니다. 또한 다음과 같은 조회가 172 페이지의 『예 - 뷰 병합』에 설명된 대로 조회 재작성 단계 중에 생성될 수 있습니다.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

이 조회에서 SQL 컴파일러는 조인을 제거하고 이 조회를 단순화할 수 있습니다.

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
     FROM   EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

다른 예에서는 EMPLOYEE 및 DEPARTMENT 샘플 테이블간의 부서 번호에 참조 제한조건이 있다고 가정합니다. 첫 번째 뷰가 작성됩니다.

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

그런 다음, 다음과 같은 조회가 작성됩니다.

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

다음과 같이 조회가 변경됩니다.

```
SELECT LASTNAME, SALARY
     FROM   EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

이러한 상황에서는 조회를 재작성할 수 있다는 사실을 알더라도 기존 테이블에 대한 액세스 권한이 없기 때문에 작성하지 못할 수 있습니다. 사용자는 뷰에 대한 액세스 권한만을 가질 수 있습니다. 따라서 이러한 유형의 최적화는 데이터베이스 관리 프로그램 내에서 수행되어야 합니다.

다음과 같은 경우에 참조 무결성 조인 내에 중복이 있을 수 있습니다.

- 뷰가 조인을 사용하여 정의됩니다.
- 조회가 자동으로 생성됩니다.

  예를 들어, 조회 관리 프로그램 내에 사용자가 최적화된 조회를 작성하지 못하게 하는 자동화된 도구가 있습니다.

예 - 공유 총계

조회 내에 여러 개의 함수를 사용하면 시간이 걸리는 몇 가지 계산을 생성할 수 있습니다. 조회 내에서 수행될 계산의 수를 줄이면 더 개선된 플랜을 생성할 수 있습니다. SQL 컴파일러는 다음과 같은 여러 개의 함수를 사용하는 조회를 가져와서

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

다음 방법으로 조회를 변환합니다.

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

이 재작성은 조회에서 두 개의 덧셈과 두 개의 계수를 한 개의 덧셈과 한 개의 계수로 줄입니다.

조작 이동

SQL 컴파일러는 조회 조작을 이동하여 최소수의 조작과 술어를 갖는 조회를 구성할 수 있도록 조회를 재작성합니다. 다음 예는 SQL 컴파일러가 이동할 수 있는 일부 조작을 설명하기 위해 제공됩니다.

- 예 - DISTINCT 제거

  조회 재작성 중에 최적화 알고리즘은 DISTICT 조작이 수행되는 위치를 이동하여 이 조작의 비용을 줄일 수 있습니다. 제공된 예에서 DISTINCT 조작이 완전히 제거됩니다.

- 예 - 일반 술어 푸시다운

  조회 재작성 중에 점더 많은 선택적 술어가 가능한 한 좀더 일찍 조회에 적용될 수 있도록 적용되는 술어의 순서를 변경할 수 있습니다.

- 예 - 상관 해제

  파티션된 데이터베이스 환경에서 데이터베이스 파티션 사이에서의 결과 세트 이동은 비효율적입니다. 다른 데이터베이스 파티션으로 브로드캐스트해야 하는 데이터의 크기와 브로드캐스트의 수를 줄이는 것은 조회 재작성의 목적 중 하나입니다.

예 - DISTINCT 제거

EMPNO 컬럼이 EMPLOYEE 테이블의 기본 키로 정의된 경우,

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
     FROM   EMPLOYEE
```

위의 조회는 DISTINCT절을 제거하여 다음과 같이 재작성됩니다.

```
SELECT EMPNO, FIRSTNME, LASTNAME
     FROM   EMPLOYEE
```

위의 예에서는 기본 키가 선택되기 때문에 SQL 컴파일러는 리턴되는 각 행이 이미 고유할 것이라는 것을 알고 있습니다. 이 경우, DISTINCT 키워드는 중복됩니다. 조회가 재작성되지 않은 경우, 최적화 알고리즘은 컬럼이 고유하다는 것을 보장할 수 있도록 필요한 처리(예: 정렬)를 포함하는 플랜을 빌드하게 됩니다.

예 - 일반 술어 푸시다운

일반적으로, 술어가 적용되는 레벨을 변경하면 성능을 향상시킬 수 있습니다. 예를 들면,『D11』부서의 모든 사원의 목록을 제공하는 다음 뷰가 있는 경우,

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM   EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

그리고 다음 조회가 있는 경우,

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

컴파일러의 조회 재작성 단계는 술어 LASTNAME = 'BROWN'을 뷰 D11_EMPLOYEE에 푸시다운합니다. 이렇게 하면 해당 술어가 더 빠르고 결국에는 보다 더 효율적으로 적용될 수 있습니다. 이 예에서 실행되는 실제 조회는 다음과 같습니다.

```
SELECT FIRSTNME, PHONENO
     FROM   EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

술어의 푸시다운은 뷰로 제한되지 않습니다. 술어가 푸시다운될 수 있는 다른 상황에는 UNION, BROUP BY 및 파생된 테이블(중첩 테이블 표현식 또는 공통 테이블 표현식)이 있습니다.

예 - 상관 해제

파티션된 데이터베이스 환경에서 SQL 컴파일러는 다음 조회를 재작성할 수 있습니다.

프로그래밍 프로젝트에서 작업 중이며 급여가 평균보다 적은 모든 사원을 찾으십시오.

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
 WHERE P.EMPNO = E.EMPNO
   AND P.PROJNAME LIKE '%PROGRAMMING%'
   AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
        FROM EMPLOYEE E1, PROJECT P1
        WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
          AND P1.PROJNO = A.PROJNO
          AND E1.EMPNO = P1.EMPNO)
```

이 조회는 상관 관계가 있고 PROJECT와 EMPLOYEE가 모두 PROJNO에 대해 파티션될 수 없기 때문에 각 프로젝트에서 각 데이터베이스 파티션으로의 브로드캐스트가 가능합니다. 또한 부속 조회는 여러 번 평가되어야 합니다.

SQL 컴파일러는 다음과 같이 조회를 재작성할 수 있습니다.

- 프로그래밍 프로젝트인 DIST_PROJS에서 작업 중인 사원의 구별 목록을 결정하십시오. 각 프로젝트마다 한 번만 총계가 수행되도록 구별되어야 합니다.

```
WITH DIST_PROJS(PROJNO, EMPNO) AS
(SELECT DISTINCT PROJNO, EMPNO
 FROM PROJECT P1
 WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
```

- 프로그래밍 프로젝트에서 작업 중인 사원의 구별 목록을 사용하여 이를 사원 테이블에 조인해서 프로젝트당 평균 보상인 AVG_PER_PROJ를 확보하십시오.

```
AVG_PER_PROJ(PROJNO, AVG_COMP) AS
(SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
 FROM EMPLOYEE E1, DIST_PROJS P2
 WHERE E1.EMPNO = P2.EMPNO
 GROUP BY P2.PROJNO)
```

- 그러면 다음과 같은 새 조회를 얻게 됩니다.

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
 FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
 WHERE P.EMPNO = E.EMPNO
   AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND P.PROJNO = A.PROJNO
   AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
```

재작성된 SQL 조회는 프로젝트당 AVG_COMP(AVG_PRE_PROJ)를 계산한 다음, 결과를 EMPLOYEE 테이블을 포함하는 모든 데이터베이스 파티션으로 브로드캐스트할 수 있습니다.

술어 변환

SQL 컴파일러는 기존의 술어를 특정 조회에 대해 보다 최적화된 술어로 변환하도록 조회를 재작성합니다. 다음 예는 SQL 컴파일러가 변환할 수 있는 일부 술어를 설명하기 위해 제공됩니다.

- 예 - 암시적 술어의 추가

  조회 재작성 중에 최적화 알고리즘이 조회에 대한 최상의 액세스 플랜을 선택할 때 추가 테이블 조인을 고려할 수 있도록 술어가 해당 조회에 추가될 수 있습니다.

- 예 - OR에서 IN으로의 변환

조회 재작성 중에 더 효율적인 액세스 플랜이 선택될 수 있도록 OR 술어가 IN 술어로 변환될 수 있습니다. SQL 컴파일러는 IN 술어에서 OR 술어로의 변환이 더 효율적인 액세스 플랜의 선택을 허용하는 경우, 이러한 변환도 수행할 수 있습니다.

예 - 암시적 술어의 추가

다음 조회는 『E01』에게 보고하는 부서의 관리자들과 해당 관리자가 담당하는 프로젝트의 목록을 생성합니다.

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
  FROM DEPARTMENT DEPT,
       EMPLOYEE   EMP,
       PROJECT    PROJ
 WHERE DEPT.ADMRDEPT = 'E01'
   AND DEPT.MGRNO = EMP.EMPNO
   AND EMP.EMPNO  = PROJ.RESPEMP
```

조회를 재작성하면 다음 암시적 술어를 추가합니다.

```
DEPT.MGRNO = PROJ.RESPEMP
```

이 재작성의 결과, 최적화 알고리즘은 해당 조회에 대한 최상의 액세스 플랜을 선택하려고 시도할 때 추가 조인을 고려할 수 있습니다.

위의 술어 전이 폐쇄 이외에, 조회를 재작성하면 등호 술어에 의해 암시되는 전이성에 기초하여 추가 지역 술어도 유도합니다. 예를 들면, 다음의 조회는 부서 번호가 『E00』보다 큰 부서와 이 부서에서 일하는 사원의 이름을 나열합니다.

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNO > 'E00'
```

이 조회의 경우, 재작성 단계는 다음의 암시적 술어를 추가합니다.

```
EMP.WORKDEPT > 'E00'
```

이 재작성의 결과, 최적화 알고리즘은 조인되는 행의 수를 줄입니다.

예 - OR에서 IN으로의 변환

다음의 예에서처럼 동일한 컬럼에 대한 둘 이상의 등호 술어를 연결하는 OR절이 있다고 가정해 보십시오.

```
 SELECT *
    FROM   EMPLOYEE
WHERE DEPTNO = 'D11'
   OR DEPTNO = 'D21'
   OR DEPTNO = 'E21'
```

DEPTNO 컬럼에 대한 색인이 없는 경우, OR절을 다음의 IN 술어로 변환하면 조회를 보다 효율적으로 처리할 수 있습니다.

```
 SELECT *
    FROM   EMPLOYEE
WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

주: 일부 경우에는 색인 OR을 수행할 수 있도록 데이터베이스 관리 프로그램이 IN 술어를 OR절의 세트로 변환할 수도 있습니다. 색인 OR에 대한 자세한 내용은 190 페이지의 『다중 색인 액세스』를 참조하십시오.

컬럼 상관에 대한 계정

두 테이블을 조인하는 Join 술어가 둘 이상인 조인으로 구성되는 조회가 포함된 응용프로그램이 있을 수 있습니다. 이것은 복잡하게 들릴지도 모르지만, 이러한 상황은 테이블 사이의 유사한, 관련 컬럼간 관계를 판별하려고 할 때 보통 발생하는 상황입니다.

예를 들어, 제조업체는 다양한 색상, 신축성 그리고 양질의 원료로 제품을 만듭니다. 완성된 제품은 만들어진 원료와 같은 색상과 신축성을 갖게 됩니다. 제조업체는 다음과 같은 조회를 발행합니다.

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR       =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

이 조회는 모든 제품의 이름과 양질의 원료를 리턴합니다. 다음과 같이 두 가지 Join 술어가 있습니다.

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

DB2 UDB 최적화 알고리즘은 이 조회를 실행하기 위한 플랜을 선택할 경우, 두 술어의 선택성을 계산하고 이 술어가 서로 관계가 없다고 간주합니다. 다시 말해서, 각 색상마다 모든 종류의 신축성이 생길 수 있으며 거꾸로 모든 신축성 레벨마다 모든 색상의 원료가 있다는 것을 의미합니다. 그 다음에는 술어 쌍의 전체적인 선택성을 계산하기 위해 통계를 사용하여 각 테이블에 있는 신축성의 레벨과 색상의 종류를 계산합니다. 이를 근거로, 예를 들어, 병합 조인보다 중첩된 루프 조인을 선택하거나 또는 그 반대로 선택할 수 있습니다.

그러나 이러한 두 술어는 연관되어 있습니다. 예를 들어, 신축성이 높은 원료는 적은 수의 색상에서만 가능하고, 신축성이 거의 없는 원료는 적은 수의 색상에서만 가능할 수 있습니다(신축성 있는 것과는 다른). 술어의 결합된 선택성은 더 적으므로(더 적은 수의 행을 제거) 조회는 더 많은 행을 리턴합니다. 이러한 경우를 보려면 각 색상마다 신축성 레벨이 하나이고 또 그 반대도 성립하는 극히 드문 경우를 생각해 보십시오. 이제 술어는 다른 한쪽에 의해 내재되기 때문에 이 중 어느 쪽이든 논리적으로 완전히 생략될 수 있습니다. 최적화 알고리즘의 플랜 선택사항은 더 이상 최상의 것일 수 없습니다. 예를 들어, 중첩된 루프 조인 플랜이 선택되지만 병합 조인이 더 빠를 수 있습니다.

다른 데이터베이스 제품에 대해 데이터베이스 관리자가 술어 중 하나의 선택성을 낮추어서 카탈로그에서 통계를 갱신하여 이 성능 문제점을 해결하려고 했지만, 이 접근 방법은 다른 조회에 원하지 않는 부가 작용을 일으킬 수 있습니다.

다음의 경우, DB2 UDB 최적화 알고리즘은 Join 술어의 상관을 검출하여 보충하려고 합니다.

1. 상관 컬럼, 즉 상관 술어에 나타나는 테이블의 컬럼에서 고유 색인을 정의합니다.
2. 레지스트리 변수 DB2_CORRELATED_PREDICATES를『NO』로 설정하지 마십시오.

위의 예에서 다음을 포함하는 고유 색인을 정의할 수 있습니다.

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

또는

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

또는 둘 다

상관을 검출하려면 이 색인의 비포함 컬럼은 다른 어떠한 컬럼도 아닌 상관 컬럼이어야 합니다. 색인에는 선택적으로 포함 컬럼이 있을 수 있습니다.

일반적으로, Join 술어에는 상관 컬럼이 세 개 이상일 수 있으므로 이를 모두 포함하는 고유 색인을 정의해야 합니다.

대부분의 경우, 테이블 하나에 있는 상관 컬럼은 기본 키를 형성합니다. 기본 키는 항상 고유하므로, 상관 컬럼에 기본 키가 있으면 또다른 고유 색인을 정의할 필요가 없습니다.

이를 수행한 후 테이블의 통계가 최신 것인지와 어떠한 이유로든 참 값에서 변경되지 않았다는 것을 확인해야 합니다(예: 최적화 알고리즘에 영향을 주기 위해).

최적화 알고리즘은 고유 색인 통계의 FIRSTnKEYCARD 및 FULLKEYCARD 정보를 사용하여 상관 경우를 검출하고 상관 술어의 결합된 선택성을 동적으로 조정합니다. 그러므로 더욱 정확한 조인 크기 및 비용의 추정치를 얻을 수 있습니다.

JOIN 술어 상관 외에도, 최적화 알고리즘은 유형 COL =『constant』의 간단한 등호 술어를 사용하여 상관에 대해 계산합니다. 예를 들어, 각각 MAKE(즉, 제조업체), MODEL, STYLE(즉, 세단, 스테이션 웨건, 스포츠), YEAR, COLOR를 가지고 있는 서로 다른 유형의 자동차 테이블을 고려해 보십시오. COLOR의 술어는 MAKE, MODEL, STYLE 또는 YEAR의 술어와 독립적인 것처럼 보입니다. 거의 모든 제조업체가 해마다 모델 및 스타일 각각에서 같은 표준 색상을 사용할 수 있게 하기 때문입니다. 그러나 술어 MAKE 및 MODEL는 하나의 자동차 메이커만 특정 이름의 모델을 만들기 때문에 확실하게 독립적이지는 않습니다. 두 개 이상의 자동차 메이커에서 사용되는 동일한 모델 이름은 자동차 메이커에서 거의 원하지 않을 것입니다. 두 개의 컬럼인 MAKE 및 MODEL에 색인이 존재할 경우, 최적화 알고리즘은 색인으로부터 통계를 사용하여 결합된 구별 값 수를 판별하고 두 컬럼 사이의 상관에 대한 선택성 또는 기본 행 수(cardinality) 추정을 조정합니다. Join 술어가 아닌 그러한 술어의 경우, 최적화 알고리즘이 조정하기 위한 고유 색인을 가지고 있지 않아도 됩니다.

데이터 액세스 개념 및 최적화

SQL문을 컴파일할 때 SQL 최적화 알고리즘은 요청을 충족시키는 여러 가지 방법의 실행 비용을 계산합니다. 이러한 평가에 근거하여 최적화 알고리즘은 최적의 액세스 플랜이라고 여겨지는 것을 선택합니다. *액세스 플랜*은 SQL문을 해석(resolve)하는 데 필요한 조작 순서를 지정합니다. 응용프로그램이 바인드되면 *패키지*가 작성됩니다. 이 패키지에는 해당 응용프로그램의 모든 정적 SQL문에 대한 액세스 플랜이 있습니다. 동적 SQL문에 대한 액세스 플랜은 응용프로그램이 실행되는 시점에 작성됩니다.

테이블의 데이터에 액세스하는 방법에는 두 가지가 있습니다. 곧바로 테이블을 읽는 방법(관계 스캔)과 테이블의 색인에 먼저 액세스하는 방법(색인 스캔)이 있습니다.

*관계 스캔*은 데이터베이스 관리 프로그램이 테이블의 모든 행에 순차적으로 액세스할 때 발생합니다. 색인 스캔이 작동하는 방식에 대해서는 184 페이지의 『색인 스캔 개념』을 참조하고, 어떤 조건하에서 각각의 스캔 유형이 사용되는지에 대해서는 194 페이지의 『관계 스캔 대 색인 스캔』을 참조하십시오.

다음에서는 테이블의 데이터에 액세스하고 조회 결과를 얻기 위하여 액세스 플랜에서 사용될 수 있는 기타의 방법에 대하여 설명합니다.

- 195 페이지의 『술어에 관련된 용어』
- 197 페이지의 『조인 개념』
- 210 페이지의 『파티션된 데이터베이스에서의 조인 전략』
- 217 페이지의 『정렬이 최적화 알고리즘에 미치는 영향』

기타 관련 항목

- 74 페이지의 『최적화 클래스 조정』에서는 SQL 컴파일러가 평가한 대체 액세스 플랜 수를 제어하는 정보에 대해 설명합니다.
- 243 페이지의 『제7장 SQL Explain 기능』에서는 SQL 컴파일러가 선택한 액세스 플랜에 대한 정보를 얻는 방법에 대해 설명합니다.

색인 스캔 개념

*색인 스캔*은 데이터베이스 관리 프로그램이 다음 내용 중 일부 또는 모두를 수행하기 위해 색인에 액세스할 때 발생합니다.

- 기본 테이블에 액세스하기 전에 (색인의 일정 범위 안에 있는 행을 스캔하여) 규정 행 세트의 범위를 좁혀 가는 경우. 색인 *스캔 범위*(스캔의 시작점 및 중지점)는 색인 컬럼이 비교되는 조회의 값에 의해 결정됩니다.
- 조회 결과를 정렬하는 경우
- 요청하는 데이터를 완전 검색하는 경우. 요청된 데이터가 모두 색인에 있는 경우, 기본 테이블에 액세스하지 않습니다. 이를 *색인 전용 액세스*라고 합니다.

스캔은 색인이 정의된 것과 반대 방향으로 수행될 수도 있습니다. 자세한 내용은 *SQL 참조서*의 CREATE INDEX문에 있는 ALLOW REVERSE SCANS 옵션을 참조하십시오.

다음 추가 주제가 제공됩니다.

- 색인 구조
- 범위를 정하기 위한 색인 스캔
- 데이터 순서화를 위한 색인 스캔
- 색인 전용 액세스
- 다중 색인 액세스
- 클러스터된 색인
- 색인 페이지 프리페치

색인 구조

데이터베이스 관리 프로그램은 B+ 트리 구조를 사용하여 색인을 저장합니다. B+ 트리는 다음 도표(RID는 행 ID를 의미함)에서처럼 하나 이상의 레벨을 가지고 있습니다.

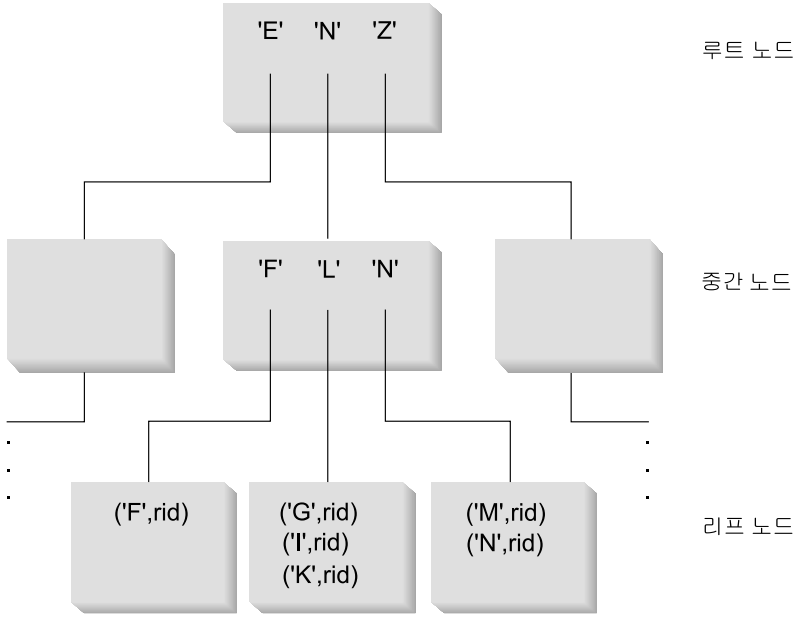


그림 13. B+ 트리 구조

최상위 레벨을 루트 노드*(root node)*라고 합니다. 최하위 레벨은 *리프 노드(leaf node)*로 구성되는데, 여기에는 테이블 내의 실제 행에 대한 포인터뿐만 아니라 실제 색인 키 값도 저장됩니다. 루트 노드와 리프 노드 레벨 사이의 레벨을 *중간 노드(intermediate node)*라고 합니다.

특정 색인 키 값을 찾으려 할 때 색인 관리 프로그램은 루트 노드에서 시작하여 색인 트리를 검색합니다. 루트 노드는 다음 레벨에 있는 각각의 노드에 대한 하나의 키를 가지고 있습니다. 이들 각각의 키 값은 다음 레벨에 있는 해당 노드에 대한 기존의 가장 큰 키 값입니다. 예를 들어, 그림13에서처럼 색인이 세 개의 레벨로 되어 있을 경우, 색인 관리 프로그램은 색인 키 값을 찾기 위해 찾으려는 키보다 크거나 같은 첫 번째 키 값을 루트 노드에서 검색합니다. 이 루트 노드 키는 특정 중간 노드를 가리키고 있습니다. 이 중간 노드에서 어느 리프 노드로 이동해야 할지를 결정하기 위해 똑같은 프로시듀어가 반복됩니다. 최종 색인 키는 리프 노드에 있습니다. 그림13을 사용할 경우, 찾을 키는 『I』입니다. 『I』보다 크거나 같은 루트 노드의 첫 번째 키는 『N』입니다. 이것은 다음 레벨에 있는 중간 노드를 가리킵니다. 『I』보다 크거나 같은 중간 노드의 첫 번째 키는 『L』입니다. 이것은 『I』에 대한 색인 키와 해당 행 ID(기본 테이블에 있는 해당 행의 행 ID)가 들어 있는 특정 리프 노드를 가리킵니다.

주: 리프 노드 레벨에는 이전 리프 포인터가 있을 수 있습니다. 이는 트리를 탐색하여 색인에서 특정 키 값을 찾기 때문에 큰 이점이 될 수 있으며, 색인 관리 프로그램은 양방향으로 리프 노드를 스캔하여 값의 범위를 검색할 수 있습니다. 양방향 스캔 기능은 색인이 ALLOW REVERSE SCANS 매개변수를 사용하여 작성된 경우에만 가능합니다.

자세한 내용은 *SQL 참조서*에서 CREATE INDEX문에 대한 옵션을 참조하십시오.

범위를 정하기 위한 색인 스캔

특정 조회에 색인을 사용할 수 있는지를 결정하기 위해 최적화 알고리즘은 색인의 각 컬럼이 다음을 충족시킬 수 있는지 보기 위해 첫 번째 컬럼부터 평가합니다.

- 명령문의 WHERE절에 있는 EQUAL 술어
- WHERE절의 기타 술어

술어는 WHERE절 검색 조건의 한 요소로서, 비교 조작을 표현하거나 내재합니다. 색인 스캔의 범위를 정하는 데 사용될 수 있는 술어는 다음 중 한 가지가 해당되는 색인 컬럼과 관련된 것입니다.

- 색인 컬럼이 상수, 호스트 변수, 상수 값을 평가하는 표현식 또는 키워드와 같은지의 여부가 테스트됩니다.
- 색인 컬럼에 대한 테스트는『IS NULL』또는『IS NOT NULL』입니다.
- 이 테스트는 기본 부속 조회(즉, ANY, ALL 또는 SOME을 가지고 있지 않은 것)와 같은지를 테스트하는 것이며, 부속 조회는 직접적인 상위 조회 블록(즉, 이 부속 조회가 부속 선택된 SELECT)에 대한 상관 컬럼 참조를 가지고 있지 않습니다.
- 테스트는 아래에 설명된 조건을 충족시키는 부등호(inequality) 술어입니다.

예를 들어, 다음 정의를 갖는 색인이 있는 경우,

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

다음 술어는 색인 IX1의 스캔 범위를 정하는 데 사용될 수 있습니다.

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
```

또는

```
WHERE  MGR  = :hv1
  AND  NAME = :hv2
  AND  DEPT = :hv3
```

두 번째 예에서 WHERE 술어가 색인에 키 컬럼이 나타난 것과 같은 순서로 지정되어 있을 필요가 없음에 주의하십시오. 그리고 이 예에서 호스트 변수가 사용되더라도, 매개변수 표시문자, 표현식 또는 상수는 동일한 효과를 나타냅니다.

CREATE INDEX문의 ALLOW REVERSE SCANS 매개변수를 사용하여 작성된 단일 색인은 양방향으로 스캔될 수 있습니다. 즉, 이러한 색인 지원은 색인이 작성될 때 정의된 방향으로 스캔하며, 역방향으로도 스캔될 수 있습니다. 명령문은 다음과 같습니다.

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

이 경우, cname의 내림차순 값에 따라 색인(iname)이 구성됩니다. 컬럼의 색인이 내림차순으로 스캔하도록 정의되어 있더라도, 역스캔이 허용되므로 오름차순으로 스캔을 수행할 수 있습니다. 실제로 색인을 양방향으로 사용하는 것은 사용자에 의해 제어되는 것이 아니라, 액세스 플랜을 작성하고 고려할 때 최적화 알고리즘에 의해 제어됩니다.

다음 WHERE절에서 NAME 및 DEPT에 대한 술어는 색인 스캔 범위를 정하는 데 사용되지만, SALARY 또는 YEARS에 대한 술어는 그렇지 않습니다.

```
WHERE  NAME   = :hv1
  AND  DEPT   = :hv2
  AND  SALARY = :hv4
  AND  YEARS  = :hv5
```

이것은 처음 두 개의 색인 키 컬럼으로부터 이 컬럼을 분리시키는 키 컬럼(MGR)이 있어서 순서화 작업이 종료되기 때문입니다. 그러나 일단 범위가 NAME = :hv1 및 DEPT = :hv2 술어에 의해 결정되면 나머지 술어는 나머지 색인 키 컬럼에 대해 평가될 수 있습니다.

위에서 설명한 등호(equality) 술어 외에도, 특정 부등호 술어가 색인 스캔의 범위를 정하는 데 사용될 수 있습니다. 다음은 두 가지 유형의 부등호 술어, 즉 완전한(strict) 부등호와 포함(inclusive) 부등호에 대한 설명입니다.

완전한 부등호(strict inequality) 술어 범위 구분 술어에 사용할 수 있는 완전한 부등호 연산자는 > 및 <입니다.

색인 스캔을 위한 범위를 정하기 위해서는 완전 부등호 술어를 갖는 하나의 컬럼만 고려됩니다. 다음의 예에서는 NAME 및 DEPT 컬럼에 대한 술어가 범위를 정하는 데 사용될 수 있지만, MGR 컬럼은 사용될 수 없습니다.

```
WHERE  NAME  = :hv1
  AND  DEPT   > :hv2
  AND  DEPT   < :hv3
  AND  MGR    < :hv4
```

포함 부등호(Inclusive Inequality) 술어: 다음은 범위 구분 술어에 사용할 수 있는 포함 부등호 연산자입니다.

- >= 및 <=
- BETWEEN
- LIKE

색인 스캔의 범위를 정하기 위해서는 포함 부등호 술어를 갖는 여러 개의 컬럼이 고려됩니다. 다음의 예에서는 모든 술어가 색인 스캔의 범위를 정하는 데 사용될 수 있습니다.

```
WHERE  NAME  = :hv1
  AND  DEPT  >= :hv2
  AND  DEPT  <= :hv3
  AND  MGR   <= :hv4
```

이 예를 좀더 자세히 설명하기 위해 `:hv2 = 404`, `:hv3 = 406` 및 `:hv4 = 12345`라고 가정해 보십시오. 데이터베이스 관리 프로그램은 404 및 405 부서에 대한 색인을 스캔합니다. 그러나 사원 번호(MGR 컬럼)가 12345보다 큰 첫 번째 관리자를 발견하면 406 부서의 스캔을 중지합니다.

자세한 내용은 195 페이지의 『범위 구분 술어 및 색인 SARGable 술어』를 참조하십시오.

데이터 순서화를 위한 색인 스캔

조회시 순서화를 해야 하는 경우, 순서화 컬럼이 첫 번째 색인 키 컬럼부터 시작하여 색인에 순서대로 나타나면 색인이 데이터를 순서화하는 데 사용될 수 있습니다. (순서화 또는 정렬은 ORDER BY, DISTINCT, GROUP BY, 『= ANY』

subquery, 『> ALL』 부속 조회, 『< ALL』 부속 조회, INTERSECT 또는 EXCEPT, UNION과 같은 조작에 의해 발생합니다.) 이에 대한 예외는 색인 키 컬럼이 『상수 값』과 같은지를 비교하는 경우(즉, 상수와 비교 평가되는 표현식)입니다. 이 경우에는 순서화 컬럼이 첫 번째 색인 키 컬럼이 아닌 다른 것일 수 있습니다. 예를 들어, 다음과 같은 조회에서

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

NAME 및 DEPT가 항상 같은 값이고 이렇게 순서화되므로, 색인이 행을 순서화하는 데 사용될 수 있습니다. 이를 다르게 표현하면 앞에 나오는 WHERE 및 ORDER BY절이 다음과 같다는 것입니다.

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

고유 색인을 사용하여 순서화에 대한 요구사항을 절단할 수 있습니다. 예를 들어, 다음 색인 정의와 ORDER BY절이 있는 경우,

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
   FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

IXO 색인이 PROJNO가 고유함을 보장하고 있으므로, PROJNAME 컬럼에 대한 추가 순서화 작업은 필요하지 않습니다. 이러한 고유성은 각각의 PROJNO 값에 대해 하나의 PROJNAME만 있음을 보장하는 것입니다.

색인 전용 액세스

일부 경우에는 필요한 모든 데이터를 테이블에 액세스하지 않고 색인으로부터 검색할 수 있습니다. 이를 *색인 전용* 액세스라고 합니다.

색인 전용 액세스를 설명하려면 다음 색인 정의를 고려해 보십시오.

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

다음 조회는 기본 테이블을 읽지 않고 색인에만 액세스하여 충족될 수 있습니다.

```
SELECT NAME, DEPT, MGR, SALARY
     FROM   EMPLOYEE
 WHERE NAME = 'SMITH'
```

다른 경우에는, 색인에 없는 컬럼이 있을 수 있습니다. 이러한 컬럼에 대한 데이터를 확보하려면 기본 테이블의 행을 읽어야 합니다. 예를 들어, IX1 색인이 있는 경우, PHONENO 및 HIREDATE 컬럼 데이터를 확보하려면 다음 조회는 기본 테이블에 액세스해야 합니다.

```
SELECT NAME, DEPT, MGR, SALARY, PHONENO, HIREDATE
     FROM   EMPLOYEE
 WHERE NAME = 'SMITH'
```

포함 컬럼이 있는 고유 색인을 작성하면 전적으로 색인에 기초한 액세스의 수를 증가시켜 데이터 검색 성능을 향상시킬 수 있습니다.

포함 컬럼의 사용을 설명하려면 다음 색인 정의를 고려해 보십시오.

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

이는 NAME 컬럼의 고유성을 강제 시행하는 반면 DEPT, MGR, SALARY 및 YEARS 컬럼에 대한 데이터를 저장하고 유지보수하는 고유 색인을 작성합니다.

다음 조회는 기본 테이블은 읽지 않고 색인에만 액세스하여 충족될 수 있습니다.

```
SELECT NAME, DEPT, MGR, SALARY
     FROM   EMPLOYEE
   WHERE NAME='SMITH'
```

다중 색인 액세스

위의 모든 예에서는 결과를 산출하는 데 단일 색인 스캔이 수행되었습니다. WHERE절의 술어를 충족시키기 위해 최적화 알고리즘은 다중 색인을 스캔하도록 선택할 수 있습니다. 예를 들어, 다음과 같은 두 개의 색인 정의가 있는 경우,

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

다음 술어는 이들 두 색인을 사용하여 해결될 수 있습니다.

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

이 예에서 *스캐닝 색인* IX2는 DEPT = :hv1 술어를 충족시키는 ID(RID) 행 목록을 생성합니다. *스캐닝 색인* IX3은 JOB = :hv2 AND YEARS >= :hv3 술어를 충족시키는 RID 목록을 생성합니다. 이러한 두 종류의 RID 목록이 테이블에 액세스하기 전에 결합되어 중복이 제거됩니다. 이를 *색인 OR 작업*이라고 합니다.

색인 OR 작업은 아래의 예에서처럼 IN 표현식을 사용하는 술어의 경우에도 사용될 수 있습니다.

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

색인 OR 작업의 목적은 중복 RID를 제거하는 것입니다. 그러나 *색인 AND 작업*의 목적은 공통 RID를 찾는 것입니다. 색인 AND 작업은 같은 테이블 내 해당 컬럼에 다중 색인이 있는 응용프로그램과, 배수 AND 술어를 사용하는 조회가 해당 테이블에 대해 수행되는 응용프로그램과 함께 발생할 수 있습니다. 이러한 조회에서 각 색인 컬럼에 대한 다중 색인 스캔은 비트맵을 작성하기 위해 해싱되는 값을 생성합니다. 두 번째 비트맵은 최종 데이터 세트를 작성하도록 페치된 규정 행을 생성하기 위해 첫 번째 비트맵을 조사하는 데 사용됩니다.

예를 들어, 다음과 같은 두 개의 색인 정의가 있는 경우,

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

다음 술어는 이들 두 색인을 사용하여 해결될 수 있습니다.

```
WHERE SALARY BETWEEN 20000 AND 30000
   AND COMM BETWEEN 1000 AND 3000
```

이 예에서 색인 IX4를 스캔하면 SALARY BETWEEN 20000 AND 30000 술어를 충족시키는 비트맵 색인을 생성합니다. IX5를 스캔하고 IX4용 비트맵을 조사하면 이 두 술어를 충족시키는 규정 RID 목록이 생깁니다. 『동적 비트맵 AND 작업』으로도 알려져 있습니다. 테이블에 기본 행 수(cardinality)가 충분히 있으며, 컬럼이 규정 범위에서 충분한 값이 있거나 또는 등호 술어가 사용될 때 충분한 중복이 있는 경우에만 발생합니다.

다중 색인을 스캔할 때 동적 비트맵의 성능 이점을 이해하려면, 힙 정렬 *크기*(*sortheap*) 데이터베이스 구성 매개변수의 값과 힙 정렬 임계값(*sheapthres*) 데이터베이스 관리 프로그램 구성 매개변수의 값을 변경해야 할 수 있습니다.

동적 비트맵이 액세스 플랜에 사용될 때 추가 힙 정렬 공간이 필요합니다. *sheapthres*가 상대적으로 *sortheap*(즉, 동시 조회당 2 - 3개인수보다 적음)에 근접하게 설정된 경우, 다중 색인 액세스를 사용하는 동적 비트맵은 최적화 알고리즘이 예상하는 것보다 메모리를 덜 사용하게 됩니다.

해결책은 *sortheap*에 상대적인 *sheapthres*의 값을 증가시키는 것입니다.

주: 모든 단일 테이블의 액세스에서 DB2는 색인 AND 작업과 색인 OR 작업을 결합하지 않습니다.

클러스터된 색인

액세스 플랜을 선택할 때 최적화 알고리즘은 디스크에서 버퍼 풀로 페이지를 페치하는 I/O 비용을 고려합니다. 계산 과정을 통해 최적화 알고리즘은 조회를 충족시키는 데 필요한 I/O 횟수를 추정합니다. 이미 버퍼 풀에 들어 있는 페이지의 행을 읽는 데는 I/O가 추가로 필요하지 않기 때문에, 이러한 추정에는 버퍼풀 사용에 대한 예상치도 포함됩니다.

색인 스캔의 경우, 최적화 알고리즘은 시스템 카탈로그 테이블(SYSCAT.INDEXES)의 정보를 이용하여 데이터 페이지를 버퍼 풀로 일어들이는 I/O 비용의 계산을 도와줍니다. SYSCAT.INDEXES 테이블의 다음 컬럼이 사용됩니다.

- 이 색인과 관련된 테이블 데이터가 클러스터된 등급을 나타내는 CLUSTERRATIO. 숫자가 높을수록 행이 데이터 페이지에서 색인 키 순서로 순서화됨을 의미합니다. 그러므로 데이터 페이지의 모든 행은 페이지가 버퍼에 있는 동안 읽을 수 있습니다. 이 컬럼의 값이 -1이면 최적화 알고리즘은 PAGE_FETCH_PAIRS 및 CLUSTERFACTOR를 사용하려고 합니다.

  또는

- CLUSTERFACTOR와 함께 다양한 *크기*의 버퍼 풀로 데이터 페이지를 읽어들이는 데 필요한 I/O 횟수를 모델링하는 여러 개의 숫자 쌍이 들어 있는 PAGE_FETCH_PAIRS. 색인에 대한 통계를 수집할 때 이 정보는 세부사항 통계로 간주됩니다.

통계를 사용할 수 없으면 최적화 알고리즘이 색인에 대한 데이터의 클러스터링을 잘못된 것으로 간주하고 기본값을 사용합니다. 자세한 내용은 125 페이지의 『제5장 시스템 카탈로그 통계』 및 127 페이지의 『RUNSTATS 유틸리티를 사용하여 통계 수집』을 참조하십시오.

테이블 재구성 중에 행을 클러스터하고 삽입 처리 중에 이러한 특성을 보존하는 데 사용될 클러스터링 색인을 지정할 수 있습니다. (테이블 재구성에 대한 자세한 내용은 304 페이지의 『카탈로그 및 사용자 테이블 재구성』을 참조하십시오.) 후속 갱신 및 삽입 작업으로 이러한 색인의 클러스터링을 줄일 수도 있으므로 (RUNSTATS로 수집된 통계에 의해 측정된 대로), 정기적으로 테이블을 재구성할 필요가 있습니다. INSERT, UPDATE 및 DELETES로 인해 자주 변경되는 테이블에서 재구성 빈도를 줄이려면 테이블 변경시 PCTFREE 매개변수를 사용하십시오. 그러면 기존 데이터에 대해 클러스터되도록 추가로 삽입할 수 있습니다.

데이터의 클러스터링 등급은 성능에 큰 영향을 미칠 수 있으므로, 테이블에 대한 색인 중 하나는 100%에 가까운 클러스터링 비율을 유지하도록 해야 합니다.

일반적으로, 하나의 색인만 100% 클러스터될 수 있습니다. 그러나 키가 클러스터링 색인의 키 수퍼세트인 경우 또는 두 색인의 키 컬럼간의 de facto 상관이 있는 경우는 제외됩니다.

클러스터링 색인 사용을 위한 성능상의 이유에 대한 자세한 내용은 115 페이지의 『색인 관리시 성능에 관련된 추가 정보』를 참조하십시오. 클러스터링 색인을 작성하는 방법에 대한 자세한 내용은 *SQL 참조서*, CREATE INDEX를 참조하십시오.

목록 프리페치를 사용한 페이지 읽기 클러스터링 최적화 알고리즘이 색인을 사용하여 행에 액세스할 경우, 색인에서 모든 RID(행 ID)가 확보될 때까지 데이터 페이지 읽기를 지연시킬 수 있습니다. 예를 들어, 이전에 정의된 색인 IX1이 있는 경우,

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

그리고 다음과 같은 검색 기준이 있는 경우,

```
WHERE NAME BETWEEN 'A' and 'I'
```

최적화 알고리즘은 검색할 행(및 데이터 페이지)을 결정하기 위해 IX1에 대한 색인 스캔을 수행할 수 있습니다. 데이터가 이 색인에 따라 클러스터되어 있지 않은 경우, 목록 프리페치는 색인 스캔에서 확보된 RID 목록을 정렬하는 단계를 포함합니다. 자세한 내용은 294 페이지의 『목록 프리페치의 이해』를 참조하십시오.

색인 페이지 프리페치

적절할 경우, 데이터베이스 관리 프로그램이 색인 페이지에 대한 순차 액세스를 검출하고 프리페치 요청을 생성합니다. 이는 색인의 중요한 부분에 액세스하는 선택적 색인 스캔과 비선택적 색인 스캔에 대한 경과 시간을 줄입니다.

최적화 알고리즘은 DENSITY 및 SEQUENTIAL_PAGES와 같은 색인 통계, 색인이 상주하는 테이블 공간의 특성, 범위 구분 술어의 영향을 사용하여 발생할 색인 페이지 프리페치 양을 추정합니다. 이러한 추정치는 특정 색인을 사용하기 위한 전반적인 비용 계산의 인수가 됩니다.

자세한 내용은 292 페이지의 『순차적 프리페치의 이해』를 참조하십시오.

관계 스캔 대 색인 스캔

최적화 알고리즘은 조회에 색인이 사용될 수 없거나 색인 스캔에 드는 비용이 더 큰 것으로 판별한 경우, 관계 스캔을 선택합니다. 다음과 같은 경우, 색인 스캔에 드는 비용이 더 큽니다.

- 테이블이 작은 경우
- 색인 클러스터링 비율이 낮은 경우
- 대부분의 테이블이 액세스한 경우

SQL Explain 기능을 사용하여 액세스 플랜이 관계 스캔을 사용할지 색인 스캔을 사용할지 여부를 결정하할 수 있습니다. 자세한 내용은 243 페이지의 『제7장 SQL Explain 기능』을 참조하십시오.

술어에 관련된 용어

사용자 응용프로그램은 술어 사용을 통해 특정 행을 규정하여 SQL문으로 데이터베이스에서 행 세트를 요청합니다. 최적화 알고리즘이 SQL문의 평가 방법을 결정할 때 각 술어는 다음의 네 가지 범주 중 하나에 속하게 됩니다. 범주는 평가 프로세스 중에 술어가 어떻게, 언제 사용되는지에 따라 결정됩니다. 이러한 범주가 성능이 가장 뛰어난 것부터 나쁜 것의 순서로 아래에 표시되어 있습니다.

1. 범위 구분 술어
2. 색인 SARGable 술어
3. 데이터 SARGable 술어
4. Residual 술어

*SARGable*은 검색 인수(search *arg*ument)로 사용될 수 있는 것을 의미합니다.

196 페이지의 『술어 사용 요약』에서는 다양한 술어 범주의 성능에 영향을 미치는 특성을 비교하여 보여줍니다.

범위 구분 술어 및 색인 SARGable 술어

범위 구분 술어는 색인 스캔의 범위를 제한하는 데 사용되는 것입니다. 이것은 색인 검색에 대한 시작 및 중지 키 값을 제공합니다. 색인 SARGable 술어는 검색 범위를 제한하는 데 사용되지는 않지만, 술어와 관련된 컬럼이 색인 키의 일부이기 때문에 색인으로부터 평가될 수 있습니다. 예를 들어, 이전에 정의된 색인 IX1(184 페이지의 『색인 스캔 개념』 절)과 다음의 WHERE절이 있는 경우,

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

처음 두 술어(NAME = :hv1, DEPT = :hv2)는 범위 구분 술어인 반면, YEARS > :hv5는 색인 SARGable 술어입니다.

데이터베이스 관리 프로그램은 기본 테이블을 읽는 것보다 이러한 술어를 평가하는 데 색인 데이터를 사용합니다. 이러한 *색인 SARGable* 술어는 테이블로부터 읽어야 할 행의 세트를 줄임으로써, 액세스되는 데이터 페이지의 수를 감소시킵니다. 이러한 유형의 술어는 액세스되는 색인 페이지의 수에 영향을 미치지 않습니다.

데이터 SARGable 술어

색인 관리 프로그램에 의해서는 평가될 수 없지만 데이터 관리 서비스에 의해서는 평가될 수 있는 술어를 *데이터 SARGable* 술어라고 합니다. 일반적으로, 이러한 술어는 기본 테이블의 행에 개별적으로 액세스해야 합니다. 필요한 경우, 데이터 관리 서비스는 색인에서 확보될 수 없는 SELECT 목록의 컬럼을 충족시키기 위한 다른 것뿐 아니라 술어를 평가하는 데 필요한 컬럼도 검색합니다.

예를 들어, 하나의 색인이 PROJECT 테이블에 정의되어 있는 경우,

```
INDEX IX0:  PROJNO ASC
```

그리고 다음 조회가 있는 경우, `DEPTNO = 'D11'` 술어는 데이터 SARGble로 간주됩니다.

```
SELECT PROJNO, PROJNAME, RESPEMP
  FROM PROJECT
WHERE DEPTNO = 'D11'
 ORDER BY PROJNO
```

Residual 술어

Residual 술어는 일반적으로 기본 테이블의 단순한 액세스 이상의 I/O이 필요한 술어를 말합니다. Residual 술어의 예에는 상관 부속 조회를 사용하는 것, 양적 부속 조회(ANY, ALL, SOME, IN 등을 사용하는 부속 조회)를 사용하는 것 또는 (테이블과 다른 파일에 저장되어 있는) LONG VARCHAR 또는 LOB 데이터를 읽는 것을 포함한 술어 등이 있습니다. 이러한 술어는 관계형 데이터 서비스에 의해 평가됩니다.

때때로 색인에만 적용된 술어는 데이터 페이지에 액세스할 때 재적용되어야 합니다. 예를 들어, 색인 OR 작업 또는 색인 AND 작업을 사용하는 액세스 플랜(190 페이지의 『다중 색인 액세스』 참조)은 데이터 페이지에 액세스할 때 항상 Residual 술어로서 술어를 재적용합니다.

술어 사용 요약

조회에서 술어를 사용하면 조회를 충족시키기 위해 읽어야 하는 데이터의 양을 감소시킬 수 있습니다. 서로 다른 범주의 술어는 조회의 성능에 다른 정도의 영향을 미치며, 최적화 알고리즘은 이러한 영향을 고려합니다. 다음 표에서는 서로 다른 종류의 술어의 순위 및 이들이 각각 어떻게 성능에 영향을 미치는지를 보여줍니다.

표 14. 술어의 유형별 특성 요약

특성	술어 유형			
	범위 구분	**색인 SARGable**	**데이터 SARGable**	**Residual**
색인 I/O 감소	예	아니오	아니오	아니오
데이터 페이지 I/O 감소	예	예	아니오	아니오
내부적으로 전달된 행 수 감소	예	예	예	아니오
규정 행 수 감소	예	예	예	예

조인 개념

조인은 어떤 테이블의 행이 하나 이상의 다른 테이블의 행과 병합되는 것을 말합니다. 예를 들어, 아래와 같이 두 개의 테이블이 있는 경우,

```
      TABLE1                    TABLE2
-----------------         -----------------
 PROJ     PROJ_ID         PROJ_ID     NAME
------    -------         -------    ------
  A          1              1        Sam
  B          2              3        Joe
  C          3              4        Mary
  D          4              1        Sue
                            2        Mike
```

ID 컬럼이 같은 조인 테이블 1과 테이블 2는 다음 SQL문에 의해 나타납니다.

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

그리고 이것은 다음과 같은 결과 행을 산출합니다.

```
 PROJ     PROJ_ID     NAME
------    -------    ------
  A          1        Sam
  A          1        Sue
  B          2        Mike
  C          3        Joe
  D          4        Mary
```

두 개의 테이블을 조인할 때 하나의 테이블은 외부 테이블로, 다른 하나는 내부 테이블로 선택됩니다. 먼저 외부 테이블에 액세스하고, 이 테이블은 단 한번 스캔됩니다. 내부 테이블이 여러 번 스캔되는지의 여부는 조인의 유형과 어떤 색인이 존재하는지에 따라 결정됩니다. 조회시 두 개의 테이블을 조인하든, 세 개 이상의 테이블을 조인하든지 상관없이, 최적화 알고리즘은 한 번에 두 개의 테이블만을 조인합니다. 필요한 경우, 임시의 중간 결과 테이블이 작성됩니다.

최적화 알고리즘은 (200 페이지의 『병합 조인』에 정의된) Join 술어의 존재 여부 및 테이블과 색인 통계에 의해 결정되는 다양한 비용 등에 따라 두 개의 조인 방법 중 하나를 선택합니다.

중첩된 루프 조인

중첩된 루프 조인은 다음의 두 방법 중 한 가지로 수행됩니다.

1. 각각의 액세스된 외부 테이블의 행에 대하여 내부 테이블을 스캔하여 수행

   예를 들어, 테이블 T1 및 T2의 컬럼 A가 다음과 같은 값을 가지고 있는 경우,

   ```
   외부 테이블 T1: 컬럼 A        내부 테이블 T2: 컬럼 A
   -----------------------      -----------------------
            2                            3
            3                            2
            3                            2
                                         3
                                         1
   ```

   중첩된 루프를 수행하는 단계는 다음과 같습니다.

   - T1에서 첫 번째 행을 읽습니다. A의 값은 『2』입니다.
   - 일치하는 값(『2』)이 발견될 때까지 T2를 스캔한 다음 두 행을 조인합니다.
   - 다음으로 일치하는 값(『2』)이 발견될 때까지 T2를 스캔한 다음 두 행을 조인합니다.
   - 테이블의 끝까지 T2를 스캔합니다.
   - T1으로 돌아간 후 다음 행(『3』)을 읽습니다.
   - 일치하는 값(『3』)이 발견될 때까지 T2를 첫 번째 행에서 시작하여 스캔한 다음 두 행을 조인합니다.

- 다음으로 일치하는 값(『3』)이 발견될 때까지 T2를 스캔한 다음 두 행을 조인합니다.
- 테이블의 끝까지 T2를 스캔합니다.
- T1으로 돌아간 후 다음 행(『3』)을 읽습니다.
- 전과 같이 T2를 스캔한 후 일치하는 값(『3』)을 가진 행을 모두 조인합니다.

2. 각각의 액세스된 외부 테이블의 행에 대해 내부 테이블의 색인을 찾음으로써 수행됩니다.

   다음 양식의 술어가 있는 경우, 지정된 술어에 대해 이 방법을 사용할 수 있습니다.

   ```
   expr(outer_table.column)  relop  inner_table.column
   ```

   여기서 relop는 관계 연산자(예: =, >, >=, < 또는 <=)이며 expr은 외부 테이블에서 유효한 표현식입니다. 예를 들면, 다음과 같습니다.

   ```
   OUTER.C1 + OUTER.C2 <= INNER.C1
   ```

   그리고

   ```
   OUTER.C4 < INNER.C3
   ```

   이 방법은 외부 테이블의 각 액세스에 대해 내부 테이블에서 액세스되는 행의 수를 상당히 감소시키는 방법이 될 수 있습니다. (그러나 이는 Join 술어의 선택성을 포함한 수많은 인수에 따라 달라질 수 있습니다.)

중첩된 루프 조인을 평가할 때 최적화 알고리즘은 조인을 수행하기 전에 외부 테이블을 정렬할 것인지의 여부도 역시 결정하게 됩니다. 조인 컬럼을 기준으로 외부 테이블을 순서화하면 내부 테이블에 대해 디스크로부터 페이지에 액세스하기 위한 읽기 조작 횟수가 줄어들 수 있는데, 이는 페이지가 이미 버퍼 풀에 들어 있을 가능성이 높기 때문입니다. 조인이 내부 테이블에 액세스하는 데 높은 비율의 클러스터된 색인을 사용하는 경우, 외부 테이블이 정렬되어 있으면 액세스되는 색인 페이지의 수는 최소화될 수 있습니다.

또한 조인이 나중에 정렬하는 데 더 큰 비용이 들게 만든다고 예상하는 경우, 최적화 알고리즘은 조인 전에 정렬을 수행하도록 선택할 수도 있습니다. 나중 정렬에 GROUP BY, DISTINCT, ORDER BY 또는 병합 조인을 지원해야 할 수도 있습니다.

병합 조인

병합 조인(병합 스캔 조인 또는 정렬 병합 조인이라고도 함)은 table1.column = table2.column 양식의 술어가 필요합니다. 이를 등호 *Join* 술어라고 합니다. 병합 조인에서는 색인 액세스를 통해, 또는 정렬을 통해 조인되는 컬럼순으로 순서화된 입력이 필요합니다. 병합 조인을 사용하려면 조인 컬럼이 LONG 필드의 컬럼이거나 대형 오브젝트(LOB) 컬럼이어서는 안됩니다.

조인된 테이블은 동시에 스캔됩니다. 병합 조인의 외부 테이블은 한 번만 스캔됩니다. 외부 테이블에 반복되는 값이 없으면 내부 테이블 역시 한 번만 스캔됩니다. 외부 테이블에 반복되는 값이 있으면 내부 테이블의 행 그룹이 다시 스캔될 수도 있습니다. 예를 들어, 테이블 T1 및 T2의 컬럼 A가 다음과 같은 값을 가지고 있는 경우,

```
외부 테이블 T1: 컬럼 A        내부 테이블 T2: 컬럼 A
-----------------------      -----------------------
          2                            1
          3                            2
          3                            2
                                       3
                                       3
```

병합 조인을 수행하기 위한 단계는 다음과 같습니다.

- T1에서 첫 번째 행을 읽습니다. A의 값은 『2』입니다.
- 일치하는 값(『2』)이 발견될 때까지 T2를 스캔한 다음 두 행을 조인합니다.
- 컬럼이 일치하는 동안 행을 조인하여 계속해서 T2를 스캔합니다.
- T2의 『3』이 읽혀지면 T1으로 돌아간 후 다음 행을 읽습니다.
- T1의 다음 값은 『3』이고, 이 값은 T2와 일치하므로 행을 조인합니다.
- 컬럼이 일치하는 동안 행을 조인하여 계속해서 T2를 스캔합니다.
- T2의 끝에 도달합니다.

- T1으로 돌아가서 다음 행을 가져오십시오. T1의 다음 값은 T1의 이전 값과 동일하므로, T2는 T2의 첫 번째 『3』에서 시작하여 다시 스캔됩니다(데이터베이스 관리 프로그램은 이 위치를 기억함).

해쉬 조인

해쉬 조인에서는 table1.columnX = table2.columnY의 양식으로 된 술어가 하나 이상 필요하고 컬럼 유형은 **같습니다**. CHAR 유형의 컬럼에 대해 길이는 같아야 합니다. DECIMAL 유형의 컬럼에 대해 정밀도와 스케일은 같아야 합니다. 컬럼 유형은 LONG 필드 컬럼 또는 대형 오브젝트(LOB) 컬럼일 수 없습니다.

먼저, 테이블 하나(INNER 테이블이라고 함)가 스캔되고 행은 정렬 힙 할당에서 파생된 메모리 버퍼로 복사됩니다(414 페이지의 『정렬 힙 *크기*(sortheap)』 데이터베이스 구성 매개변수 참조). 메모리 버퍼는 Join 술어(들)의 컬럼(들)에서 계산된 『해쉬 *코드*』를 기초로 파티션으로 나누어집니다. 첫 번째 테이블의 *크기*가 사용할 수 있는 정렬 힙 공간을 초과하면 선택된 파티션의 버퍼는 임시 테이블에 기록됩니다. INNER 테이블의처리를 완료한 후 두 번째 테이블(OUTER 테이블이라고 함)이 스캔됩니다. OUTER 테이블의 행은 먼저 Join 술어(들)의 컬럼에서 생성된 『해쉬 코드』를 비교하여 INNER 테이블의 행과 대응됩니다. 그런 다음 OUTER 행의 『해쉬 코드』가 INNER 행의 『해쉬 코드』와 대응되면 실제 Join 술어 컬럼이 비교됩니다.

임시 테이블에 기록되지 않은 파티션에 해당하는 OUTER 테이블 행은 즉시 메모리의 INNER 테이블 행과 대응됩니다. 또는 해당 INNER 테이블 파티션이 임시 테이블에 기록되면, OUTER 행도 임시 테이블에 기록됩니다. 결국, 임시 테이블에서 일치하는 파티션 쌍이 읽히고 해당 행의 『해쉬 코드』가 대응되며 Join 술어는 점검됩니다.

해쉬 조인의 성능상의 이점을 이해하려면 *sortheap* 데이터베이스 구성 매개변수와 *sheapthres* 데이터베이스 관리 프로그램 구성 매개변수의 값을 변경하십시오.

결정 지원 조회의 경우, 해쉬 조인 액세스 플랜은 비 해쉬 조인 플랜보다 더 많은 정렬 힙(heap) 공간을 사용합니다. *sheapthres*가 상대적으로 *sortheap*(즉, 동시 조회당 2-3개인수보다 적음)에 근접하게 설정된 경우, 해쉬 조인은 최적화 알고리즘이 예상하는 것보다 메모리를 덜 사용하게 됩니다. 제한된 메모리를 사용하여 실

행할 경우, 해쉬 조인은 느리게 수행될 수 있습니다. 여러 번의 정렬 및 해쉬 조인이 있는 조회에서 문제점이 발생하며, 이 때 정렬이나 해쉬 조인에서 대부분의 사용 가능한 메모리를 확보합니다.

해결책은 *sheapthres*를 (*sortheap*에 비해) 충분히 크게 구성하는 것입니다.

외부 및 내부 테이블 결정

조인할 때 어떻게 내부 테이블과 외부 테이블이 결정되는 것일까요? 다음은 최적화 알고리즘에서 어떤 테이블이 내부 테이블이 되고, 어떤 테이블이 외부 테이블이 될지를 결정하는 방식에 대한 일반적인 지침입니다.

해쉬 조인의 경우, 내부 테이블은 메모리 버퍼에 보존됩니다. 메모리 버퍼 수가 너무 적으면 해쉬 조인이 강제로 유출되어야 합니다. 최적화 알고리즘은 이를 피하려고 하므로 두 테이블 중 더 작은 것을 내부 테이블로 선택하고 큰 테이블을 외부 테이블로 선택합니다.

중첩된 루프 조인의 경우, 테이블에 액세스하는 순서는 특히 중요한데, 이는 외부 테이블에 한 번 액세스하는 데 대해, 외부 테이블의 각 행에 대해 각각 한번씩 내부 테이블에 액세스되기 때문입니다. 최적화 알고리즘이 비용 추정치에 근거하여 외부 테이블과 내부 테이블을 선택합니다. 이 비용 추정치는 다음 인수의 영향을 받습니다.

- *크기*

  내부 테이블에 다시 액세스하는 횟수를 줄이기 위하여 때때로 더 작은 테이블이 외부 테이블로 선택됩니다. 그러나 프리페치의 경우는 반대일 수 있습니다. 프리페치는 대형 테이블에 액세스하는 비용을 상당히 줄일 수 있습니다. 그러나 일반적으로 프리페치는 조인의 외부 테이블에 대해서만 효과적인 방법입니다. 그러므로 더 큰 테이블에 맨 먼저 액세스할 수 있습니다. 자세한 내용은 292 페이지의 『버퍼 풀로 데이터 프리페치』를 참조하십시오.

- 술어

  내부 테이블에 외부 테이블에 적용된 술어를 충족시키는 행에 대해서만 액세스하므로, 선택적 술어를 테이블에 적용할 수 있는 경우 테이블은 외부 테이블로서 선택될 가능성이 많습니다.

- 버퍼링

외부 테이블의 각각의 행에 대해 전체 내부 테이블이 스캔되어야 하는 경우(즉, 색인 찾아보기가 내부 테이블에서 수행될 수 없는 경우), 버퍼링을 이용하여 두 테이블 중 작은 테이블이 내부 테이블로 선택될 수 있습니다. 이것은 테이블 크기와 버퍼 풀의 크기에 영향을 받게 됩니다. 조인 결정이 버퍼 풀 크기에 의하여 영향을 받으므로, 버퍼 풀 크기를 변경한 후 응용프로그램을 데이터베이스에 리바인드한 경우에는 응용프로그램에 대한 액세스 플랜이 변경될 수 있습니다.

내부 및 외부 테이블에 버퍼링이 사용될 경우, 둘 이상의 버퍼 풀을 작성하고 해당 버퍼 풀의 크기를 변경하며 버퍼 풀을 사용하는 테이블 공간을 제어하는 사용자의 능력에 영향을 줍니다.

- 색인

  테이블 중 하나에 대하여 색인 찾아보기를 할 수 있는 경우, 해당 테이블이 내부 테이블로 사용될 수 있는 좋은 후보가 됩니다. 그러면 키 값 중 하나로 외부 테이블의 조인 키 술어를 사용하여 색인 키 찾아보기로 액세스할 수 있습니다. 테이블에 색인이 없으면 내부 테이블의 후보로서 적합치 않은데, 이는 외부 테이블의 모든 행에 대해 내부 테이블 전체가 스캔되어야 하기 때문입니다.

- 순서 요구사항

  요구되는 순서와 연관된 테이블에 먼저 액세스해야 합니다. 예를 들어, t1과 t2의 조인 결과가 t1.c를 기준으로 순서화되어야 하는 경우, t1.c의 색인으로 t1을 외부 테이블로 액세스하는 것이 좋은 선택일 수 있습니다. 조인의 결과는 순서화되어 있으며 정렬은 필요하지 않습니다.

  ```
  SELECT * FROM t1, t2
    WHERE t1.a = t2.b
    ORDER BY t1.c
  ```

병합 조인의 경우, 테이블에 액세스하는 순서는 별로 중요하지 않은데, 이는 내부 및 외부 테이블이 모두 한 번씩만 읽히기 때문입니다. 그러나 외부 테이블의 중복 조인 값에 대응되는 내부 테이블의 부분은 메모리 내 버퍼에 보존됩니다. 다음 외부 행이 이전 외부 행과 같은 경우 버퍼는 다시 읽혀지고, 그렇지 않은 경우 버퍼는 재설정됩니다. 중복 조인 값의 수가 메모리 내 버퍼 용량을 초과하는 경우, 중복이 모두 보존되는 것은 아닙니다. 모든 중복 값이 크고 값이 외부 테이블에 일치하는 값을 가진 경우에만 발생할 수 있습니다.

중복 값에 대한 모든 고려사항으로, 대부분 경우에 조인의 외부 테이블로 선택될 중복이 거의 없는 테이블입니다. 그러나 결국 최적화 알고리즘은 상세한 비용 추정치에 근거하여 외부 및 내부 테이블을 선택합니다.

최적의 조인을 선택하기 위한 검색 전략

최적화 알고리즘은 다양한 검색 전략을 사용하여 최적의 조인 방법을 결정할 수 있습니다. 사용할 검색 전략은 사용 중인 최적화 클래스에 의해 결정됩니다(74 페이지의 『최적화 클래스 조정』 참조). 검색 전략과 해당 특성은 다음과 같습니다.

- Greedy 조인 열거
  - 시간과 공간 고려시에 효과적입니다.
  - 단일 방향 열거, 즉 일단 두 개의 테이블에 대해 조인 방법이 선택되면 이후 최적화 동안에 변경되지 않습니다.
  - 많은 테이블을 조인할 때 최상의 액세스 플랜을 놓칠 수도 있습니다. 조회에서 두 개 또는 세 개의 테이블만을 조인하는 경우, Greedy 조인 열거로 선택된 액세스 플랜은 동적 프로그래밍 조인 열거로 선택된 액세스 플랜과 같은 것이 됩니다. 이는 특히 같은 컬럼에 Join 술어가 많은 조회의 경우에(이것이 명시적으로 지정되어 있든, 술어 전이 폐쇄에 의해 내재적으로 생성된 것이든) 해당됩니다.
- 동적 프로그래밍 조인 열거
  - 조인 중인 테이블 수가 증가함에 따라 공간 및 시간 요구사항도 기하급수적으로 증가합니다.
  - 최상의 액세스 플랜을 위한 효율적이고 철저한 검색
  - OS/390용 DB2에 사용된 전략과 유사

조인 열거 알고리즘이 최적화 알고리즘에 의해 탐색되는 플랜 조합의 수를 결정하는 주된 요소가 됩니다.

스타 조인의 검색 전략

일반적으로, 조회시 참조되는 테이블은 Join 술어로 연결되어 있어야 합니다. Join 술어 없이 두 개의 테이블이 조인될 경우, 두 테이블의 Cartesian 제품이 형성됩니다. 즉, 첫 번째 테이블의 모든 규정 행은 두 번째 테이블의 모든 규정 행과 조인되어, 보통 매우 큰 두 테이블의 크기로 된 cross 제품으로 구성된 결과 테이블

을 작성합니다. 이러한 플랜이 제대로 수행된다는 것은 불가능하므로, 최적화 알고리즘은 그러한 액세스 플랜의 비용을 결정하는 것조차 피합니다. 예외가 발생하는 경우에는 최적화 클래스가 9로 설정되거나, 뒤따르는 『스타 스키마』의 특수한 경우뿐입니다. 자세한 내용은 74 페이지의 『최적화 클래스 조정』을 참조하십시오.

Cartesian 제품을 포함하는 액세스 플랜이 제대로 수행되는 경우는 보통 스타 스키마 기술로 설계된 대형 결정 지원 데이터베이스의 경우입니다. 스타 스키마는 대량의 원시 데이터가 여러 컬럼으로 구성되며 보통 『사실(fact)』 테이블이라는 하나의 대형 테이블에 보존되는 데이터베이스 설계입니다. 많은 컬럼에 사실 테이블에 저장된 특정 기준의 차원 특성을 나타내는 인코딩 값이 들어 있습니다. 사실(fact) 테이블의 일부 부속 집합을 쉽게 분석할 수 있도록 인코딩 값을 해독하는 데 차원 테이블이 사용됩니다. 일반 조회는 차원 테이블의 해독된 값을 참조하는 복수의 지역 술어로 구성되며, 차원 테이블을 사실 테이블에 연결하는 Join 술어가 들어 있습니다. 이러한 종류의 조회인 경우, 대형의 사실 테이블에 액세스하기 전에 여러 개의 작은 차원 테이블을 갖는 Cartesian 제품을 수행하는 것이 유익합니다. 이 기법은 여러 Join 술어가 다중 컬럼 색인에 일치될 때 유용합니다.

DB2는 최소한 2차원 테이블을 가진 스타 스키마로 설계된 데이터베이스에 대한 조회를 인식할 수 있으며, 차원 테이블의 Cartesian 제품 형성 과정에 관련된 잠재적 플랜을 포함할 수 있도록 검색 공간을 늘릴 수도 있습니다. Cartesian 제품을 포함하는 플랜이 최저의 계산된 비용을 갖는 경우, 이 플랜이 최적화 알고리즘에 의해 선택됩니다.

위에서 논의된 스타 스키마 기술은 조인에 기본 키 색인이 사용된 것으로 가정한 것입니다. 또다른 시나리오로는 외부 키 색인과 관련된 것이 있습니다. 사실 테이블의 외부 키 컬럼이 단일 컬럼 색인이며, 모든 차원 테이블에 대해 상대적으로 높은 선택성이 있는 경우, 다음의 스타 조인 기술을 사용할 수 있습니다.

1. 각 차원 테이블은 다음 작업에 의해 처리됩니다.
   - 차원 테이블과 사실 테이블의 외부 키 사이에 세미 조인(semi-join) 수행
   - 행 ID(RID) 값을 해싱하여 동적으로 비트맵 작성
2. 각 비트맵은 이전 비트맵에 대해 AND 술어로 사용됩니다(190 페이지의 『다중 색인 액세스』 참조).
3. 마지막 비트맵을 처리한 후 남아 있는 RID 판별

4. 이 RID를 선택적으로 정렬
5. 기본 테이블 행 페치
6. SELECT절에 필요한 차원 테이블의 컬럼에 액세스하여 각 차원 테이블을 가진 사실 테이블 재조인
7. 술어(Residual 술어) 재적용

이 기술을 사용하면 다중 컬럼의 색인을 가져야 한다는 요구사항이 필요없습니다. 사실 테이블과 차원 테이블 사이의 명시적 참조 무결성 제한조건은 이 기술을 선택할 경우에 사실 테이블과 차원 테이블 사이의 관계가 이러한 특성을 가지고 있어야 하지만 반드시 필요한 것은 아닙니다.

동적 비트맵은 Star 조인 기술이 정렬 힙 메모리를 사용하는 부분으로 작성되고 사용됩니다. 정렬 힙 *크기*(*sortheap*) 데이터베이스 구성 매개변수에 대한 자세한 내용은 *관리 안내서: 성능* 매뉴얼의 13장, ″DB2 구성″을 참조하십시오.

복합 테이블

또다른 중요한 매개변수가 조회 내의 조인 순서 모양을 결정합니다. 한 쌍의 테이블을 조인한 결과 복합이라는 새 테이블이 만들어집니다. 일반적으로, 이러한 결과 복합 테이블은 또다른 내부 테이블을 갖는 조인의 외부 테이블이 됩니다. 이를 『복합 외부』라고 합니다. 일부 상황에서는 Greedy 조인 열거 기법 사용시 두 테이블의 조인 결과로 나중 조인의 내부 테이블을 만드는 것이 유용합니다. 조인 자체의 내부 테이블이 둘 이상의 테이블을 조인한 결과로 구성된 것이면 이 플랜에는 『복합 내부』가 포함됩니다. 예를 들어, 다음 조회의 경우,

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

이는 테이블 T1과 T2(T1xT2)를 조인한 다음 T3을 T4(T3xT4)에 조인하며, 마지막으로 첫 번째 조인 결과를 외부로, 두 번째 조인 결과를 내부로 선택하는 데 유리합니다. 최종 플랜((T1xT2) x (T3xT4))에서 조인 결과(T3xT4)는 복합 내부라고 합니다. 조회 최적화 클래스에 따라 최적화 알고리즘이 다른 제한조건을 조인의 내부 테이블이 될 수도 있는 최대수의 테이블에 놓습니다. 복합 내부는 최적화 클래스 5, 7 또는 9에서 허용됩니다.

복제 요약 테이블

파티션된 데이터베이스 환경에서 복제 요약 테이블을 사용하면 데이터베이스 관리에서 기본 테이블 데이터의 값을 사전에 계산하여 성능이 향상될 수 있습니다. 예를 들어, 아래의 조회는 아래의 복제 요약 테이블을 작성하는 이점을 얻게 됩니다. 다음 사항을 가정합니다.

- SALES 테이블은 다중 파티션 테이블 공간 REGIONTABLESPACE에 있으며, REGION 컬럼에서 파티션됩니다.
- EMPLOYEE 및 DEPARTMENT 테이블은 단일 파티션 노드 그룹에 있습니다.

그런 다음 EMPLOYEE 테이블의 정보에 기초하여 복제 요약 테이블을 작성합니다.

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM   EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

복제 요약 테이블이 작성되면 이 테이블의 내용은 다음 명령문을 수행하여 갱신됩니다.

```
REFRESH TABLE R_EMPLOYEE;
```

다음 예에서는 사원별 판매액, 부서별 총계 및 이를 합한 총계를 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;
```

하나의 데이터베이스 파티션에만 있는 EMPLOYEE 테이블을 사용하는 대신 데이터베이스 관리 프로그램은 R_EMPLOYEE 테이블을 사용하며, 이 테이블은

SALE 테이블이 있는 각 데이터베이스 파티션에서 복제됩니다. 사원 정보가 조인 계산을 위해 네트워크를 거쳐 각 데이터베이스 파티션에 이동될 필요가 없으므로 성능이 향상됩니다.

복제 요약 테이블은 조인의 조합을 지원하는 데 사용될 수 있습니다. 예를 들어, 20개의 노드에 퍼진 대형 사실 테이블이 있는 스타 스키마가 있으면 사실 테이블과 차원 테이블이 조합된 경우 이 두 테이블의 조인이 가장 효과적입니다.

동일한 노드 그룹에 모든 테이블을 놓으면 조합 조인에 적어도 하나의 올바르게 파티션된 차원 테이블이 있습니다. 기타 모든 차원 테이블은 사실 테이블의 조인 컬럼이 사실 테이블의 파티션 키와 상응하지 않기 때문에 조합 조인에 사용될 수 없습니다.

예를 들어, C1에 파티션된 FACT 테이블(C1, C2, C3, ...), C1에 파티션된 DIM1 테이블(C1, dim1a, dim1b, ...) 및 C2에 파티션된 DIM2 테이블(C2, dim2a, dim2b, ...) 등을 가질 수 있습니다.

이 예에서는 DIM1.C1 = FACT.C1 술어가 조합되기 때문에 FACT와 DIM1의 조인이 완벽합니다. 이들 두 테이블 모두 컬럼 C1에 파티션되어 있습니다.

FACT는 컬럼 C1에 파티션되어 있지만 컬럼 C2에는 파티션되어 있지 않으므로 WHERE DIM2.C2 = FACT.C2 술어를 사용하여 DIM2 간의 조인이 조합될 수 없습니다.

이런 경우에는 사실 테이블의 노드 그룹에 DIM2를 복제하는 것이 좋습니다. 이런 방식으로 각 파티션에서 논리적으로 조인을 수행할 수 있습니다.

주: 여기에서 설명하는 복제 요약 테이블은 데이터베이스 내 복제로 수행해야 합니다. 데이터베이스 내 복제는 복사 작업 내역, 제어 테이블 및 다른 데이터베이스 및 다른 운영 체제에 있는 데이터로 수행해야 합니다. 데이터베이스 내 복제에 관심이 있는 경우, 자세한 내용은 *복제 안내 및 참조서*를 참조하십시오.

복제 요약 테이블을 작성할 때 소스 테이블은 단일 노드의 노드 그룹 테이블 또는 다중 노드의 노드 그룹 테이블일 수 있습니다. 대부분의 경우 테이블은 작으며 단일 노드의 노드 그룹에 배치할 수 있습니다. 테이블에서 컬럼의 부속 집합만 지

정하거나 사용된 술어를 통해 행 수를 제한하거나 또는 복제 요약 테이블을 작성할 때 두 가지 방법을 모두 사용하여 복제할 데이터에 대한 한계를 배치할 수 있습니다.

주: 데이터 캡쳐 옵션은 복제 요약 테이블이 기능하는 데는 필요하지 않습니다.

또한 복제 요약 테이블이 다중 노드의 노드 그룹에 작성될 수 있습니다. 노드 그룹은 대형 테이블을 배치한 노드 그룹과 동일합니다. 이 경우 소스 테이블의 복사본은 노드 그룹의 모든 파티션에 작성됩니다. 대형 사실 테이블과 차원 테이블간의 조인은 소스 테이블을 모든 파티션으로 브로드캐스트해야 하는 것보다 오히려 이 환경에서 논리적으로 더 잘 수행될 수 있습니다.

복제 테이블의 색인은 자동으로 작성되지 않습니다. 색인은 작성되어 소스 테이블에서 식별된 것과 다를 수 있습니다.

주: 복제 테이블에 고유 색인을 작성(또는 제한조건에 배치)할 수 없습니다. 이것은 소스 테이블에 나타나지 않는 제한조건 위반을 방지합니다. 이러한 제한조건은 소스 테이블에 동일한 제한조건이 있는 경우에도 허용되지 않습니다.

REFRESH문을 사용한 후 기타 테이블을 수행할 때 복제 테이블에 RUNSTATS를 수행해야 합니다.

복제 테이블은 조회 내에서 직접 참조될 수 있습니다. 그러나 복제 테이블에서 NODENUMBER() 술어를 사용하여 특정 파티션의 테이블 데이터를 볼 수 없습니다.

작성된 복제 요약 테이블이 사용되었는지(소스 테이블을 참조한 조회를 부여했는지) 확인하기 위해 EXPLAIN 기능을 사용할 수 있습니다. 먼저 EXPLAIN 테이블이 있는지 확인합니다. 그런 다음 관심있는 SELECT문에 대한 Explain 플랜을 작성합니다. 마지막으로 db2exfmt 유틸리티를 사용하여 EXPLAIN 출력을 형식화합니다.

최적화 알고리즘에 의해 선택된 액세스 플랜은 조인해야 하는 정보에 따라 복제 요약 테이블을 사용할 수도 있고 사용하지 않을 수도 있습니다. 최적화 알고리즘이 원래 소스 테이블을 노드 그룹의 다른 파티션으로 브로드캐스트하는 것이 비용이 더 적게 든다고 판단한 경우, 복제 요약 테이블을 사용하지 않습니다.

파티션된 데이터베이스에서의 조인 전략

다음 절에서는 파티션된 데이터베이스 환경에서 가능한 조인 전략에 대해 설명합니다. DB2 최적화 알고리즘은 각 응용프로그램의 요구사항에 따라 최상의 조인 전략을 자동으로 선택합니다. 여기에 조인 전략을 소개하는 것은 각 전략에서 발생하는 사항에 대한 사용자의 이해를 돕기 위한 것입니다. 『테이블 대기행렬』은 데이터베이스 파티션 사이로, 또는 하나의 파티션된 데이터베이스의 프로세서 사이로 행을 전송하는 메커니즘입니다.

다음 설명에서 *경로지정* 테이블 대기행렬이란 행이 수신 데이터베이스 파티션 중 하나로 해쉬되는 대기행렬입니다. *브로드캐스트* 테이블 대기행렬은 행이 모든 수신 데이터베이스 파티션으로 전송되는(즉, 해쉬되지 않는) 대기행렬입니다. 이 q1 섹션에 대한 도표에서 q2와 q3는 예에 있는 테이블 대기행렬을 참조합니다. 또한 참조된 테이블은 이러한 시나리오의 목적에 맞추어 두 데이터베이스 파티션에 걸쳐 파티션됩니다. 화살표는 테이블 대기행렬이 전송되는 방향을 나타냅니다. 조정자(coordinator) 노드는 파티션 0입니다.

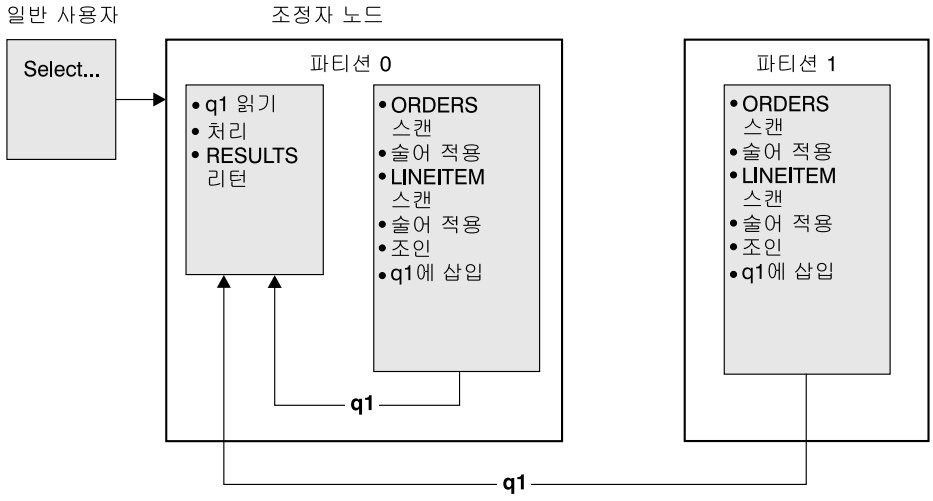
파티션된 데이터베이스의 잦은 조인과 관련된 해당 테이블에 대한 고려사항은 테이블 조합 고려사항이 됩니다. 테이블 조합은 파티션된 데이터베이스에서 데이터를 데이터를 가진 테이블로부터 같은 파티션 키에 기초한 같은 파티션의 다른 테이블로 위치지정하는 방법을 제공합니다. 일단 조합되면 조인될 데이터는 조회 활동의 일부로 다른 데이터베이스 파티션으로 이동하지 않고 조회에 참여할 수 있습니다. 조인에 대한 응답 세트만이 조정자(coordinator) 노드로 이동됩니다. 이 주제에 대한 자세한 내용은 *관리 안내서: 계획*의 『테이블 배치』를 참조하십시오.

조인 종속성에 대한 자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오.

조합 조인

최적화 알고리즘이 배치 조인을 고려하는 경우, 조인된 테이블이 조합되어 있어야 하며, 대응한 모든 파티션 키 쌍이 등호 join 술어에 참여하고 있어야 합니다. 예는 211 페이지의 그림14에 있습니다.

주: 복제 요약 테이블은 조합 조인의 가능성을 향상시킵니다. 자세한 내용은 207 페이지의 『복제 요약 테이블』을 참조하십시오.

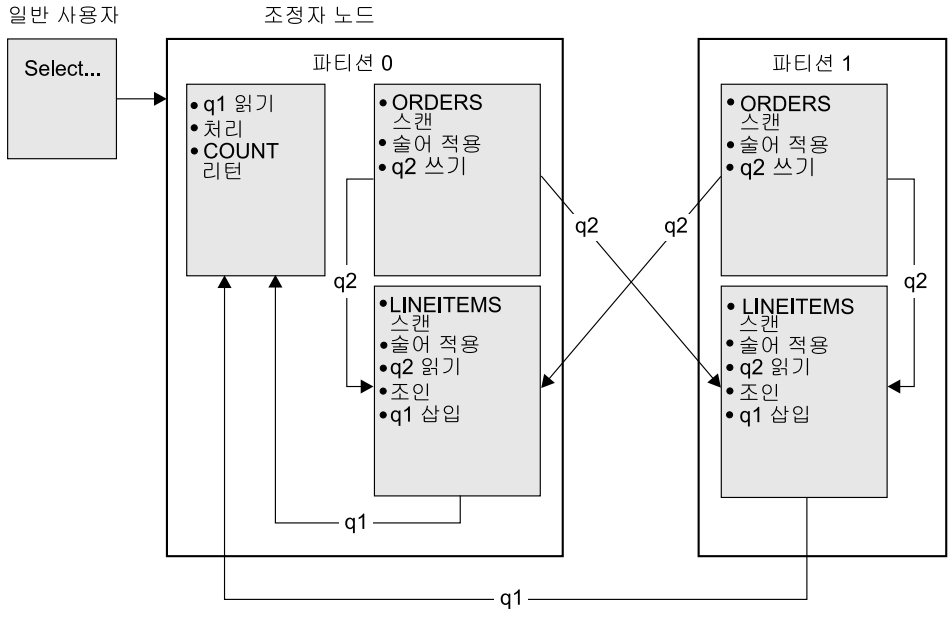


LINEITEM과 ORDERS 테이블이 ORDERKEY 컬럼에서 파티션됩니다.
조인은 각각의 데이터베이스 파티션에서 지역적으로 수행됩니다.
이 예에서 Join 술어는 다음으로 가정됩니다.
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

그림14. 조합 조인 예

외부 테이블 브로드캐스트 조인

조인된 테이블간에 등호 Join 술어가 없을 경우, 이러한 병렬 조인 전략을 사용할 수 있습니다. 또한 이 방법이 가장 저렴한 조인 방법이면 다른 상황에서도 이 방법을 사용할 수 있습니다. 일반적으로, 매우 큰 테이블과 아주 작은 테이블이 있는데, Join 술어 컬럼에서 파티션되지 않은 경우에 발생할 수 있습니다. 두 테이블을 파티션하는 것보다 더 작은 테이블을 더 큰 테이블로 브로드캐스트하는 것이 『비용이 적게』 듭니다. 예는 212 페이지의 그림15에 있습니다.



ORDERS 테이블이 LINEITEM 테이블이 있는 모든 데이터베이스 파티션으로 전송됩니다.
테이블 대기행렬 q2는 내부 테이블의 모든 데이터베이스 파티션으로 브로드캐스트됩니다.

그림 15. 외부 테이블 브로드캐스트 조인 예

외부 테이블 경로지정 조인

이 조인 전략에서 외부 테이블의 각 행은 내부 테이블의 한 데이터베이스 파티션으로 전송됩니다(내부 테이블의 파티션 속성에 기초하여). 이 데이터베이스 파티션에서 조인이 발생합니다. 예는 213 페이지의 그림16에 있습니다.

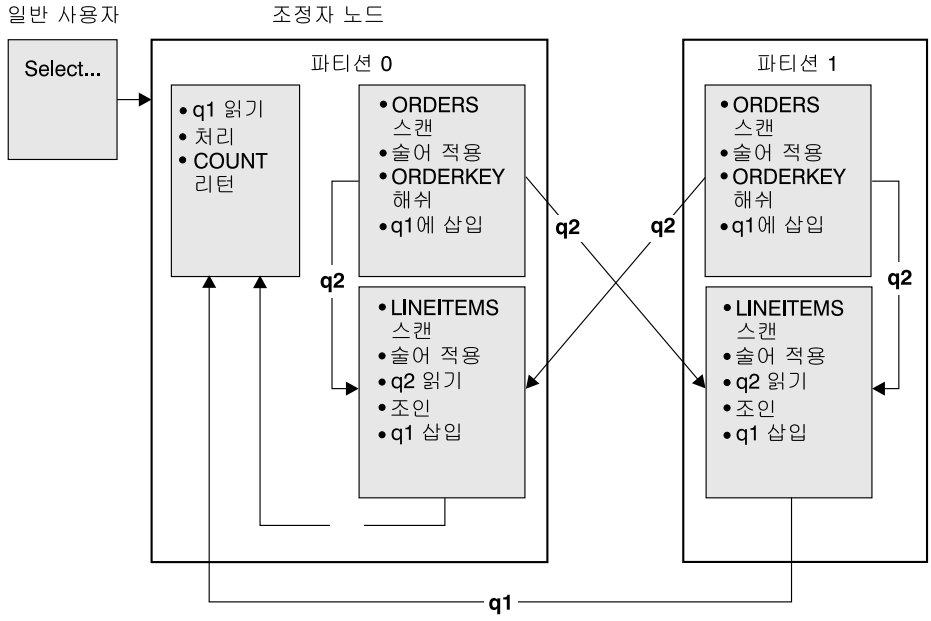
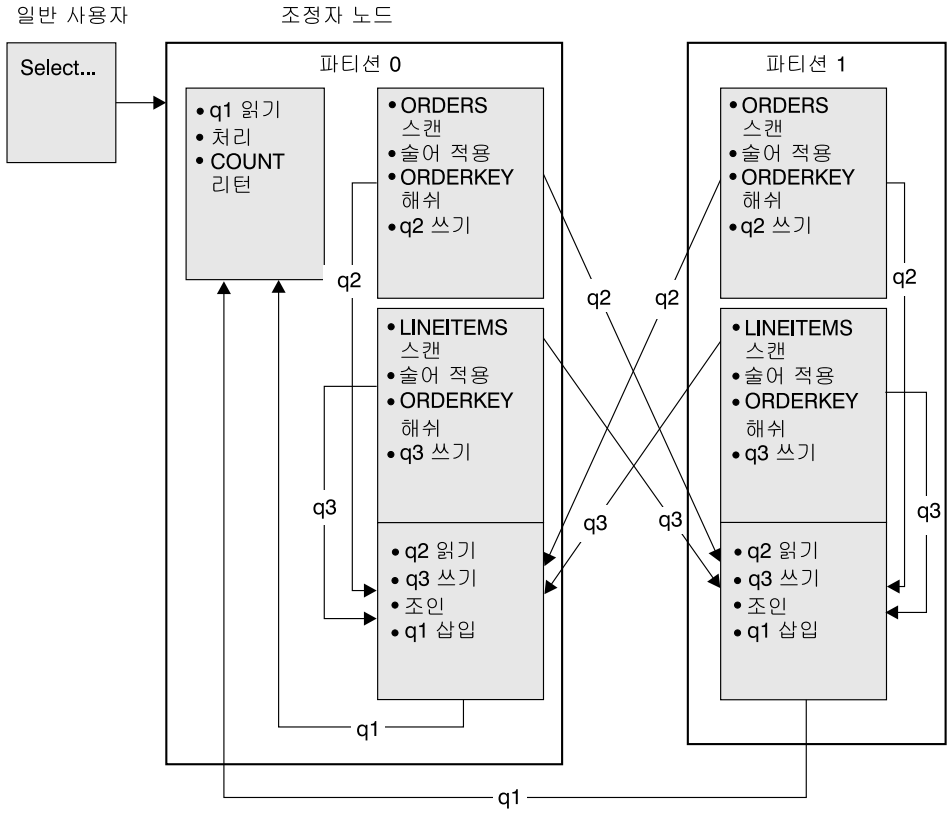


그림 16. 외부 테이블 경로지정 조인 예

내부 테이블 및 외부 테이블 경로지정 조인

이 전략에서 외부 및 내부 테이블 행은 조인 컬럼의 값에 따라 데이터베이스 파티션 세트로 경로지정됩니다. 이들 데이터베이스 파티션에서 조인이 발생합니다. 예는 214 페이지의 그림17에 있습니다.

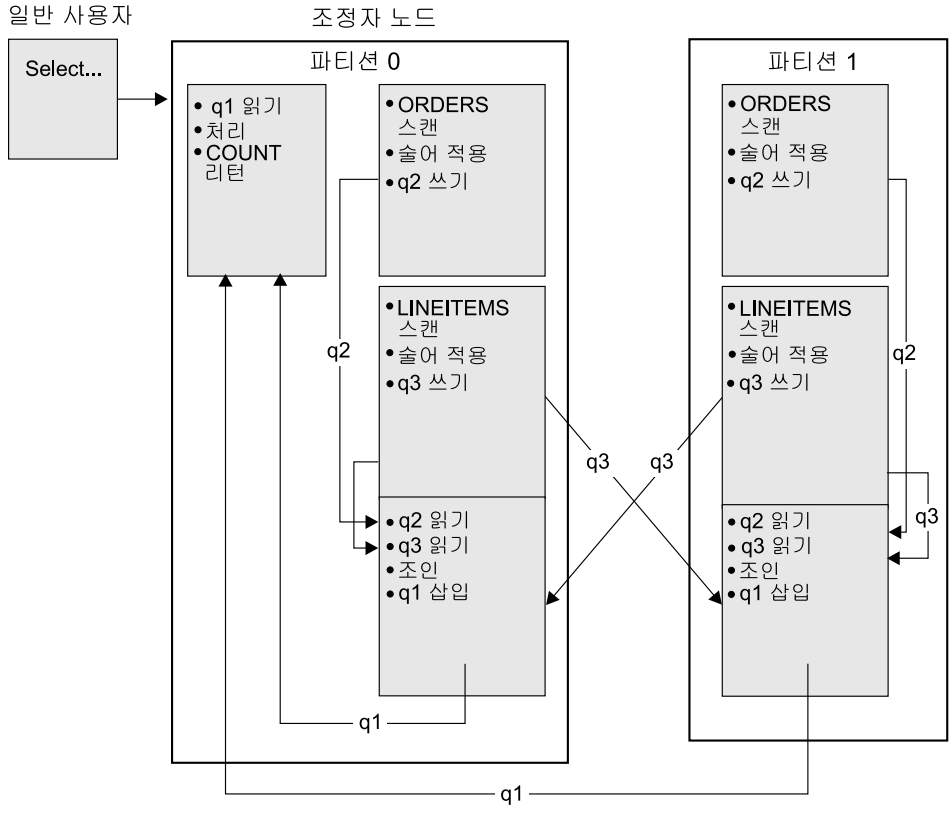


어느 테이블도 ORDERKEY 컬럼에서 파티션되지 않습니다.
두 테이블 모두 해쉬되어 조인되는 새 데이터베이스 파티션으로 전송됩니다.
두 테이블 대기행렬 q2와 q3가 경로지정됩니다.
이 예에서 Join 술어는 다음으로 가정됩니다.
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

그림 17. 내부 테이블 및 외부 테이블 경로지정 조인 예

내부 테이블 브로드캐스트 조인

이 경우, 내부 테이블은 외부 조인 테이블의 모든 데이터베이스 파티션으로 브로드캐스트됩니다. 예는 215 페이지의 그림18에 있습니다.

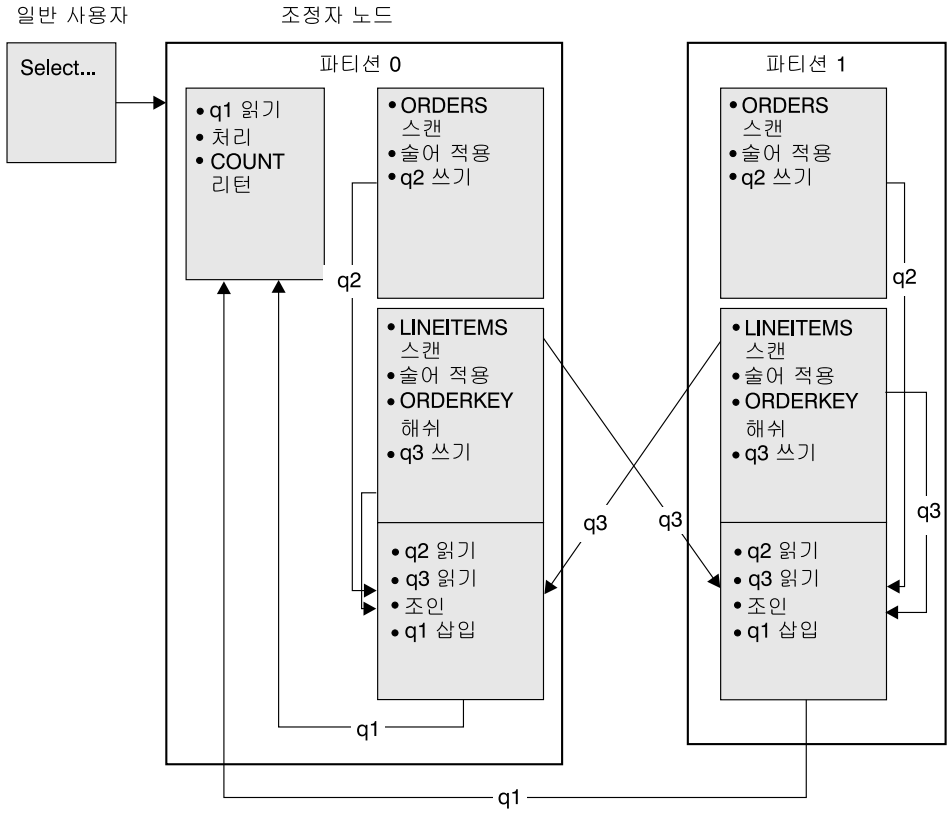


ORDERS 테이블이 LINEITEM 테이블에 있는 모든 데이터베이스 파티션으로 전송됩니다.
테이블 대기행렬 q3은 내부 테이블의 모든 데이터베이스 파티션으로 브로드캐스트됩니다.

그림 18. 내부 테이블 브로드캐스트 조인 예

내부 테이블 경로지정 조인

이 조인 전략에서 내부 테이블의 각 행은 외부 조인 테이블의 한 데이터베이스 파티션으로 전송됩니다(외부 테이블의 파티션 속성에 기초하여). 이 데이터베이스 파티션에서 조인이 발생합니다. 예는 216 페이지의 그림19에 있습니다.



ORDERS 테이블은 ORDERKEY 컬럼에서 파티션됩니다.
LINEITEM 테이블은 다른 컬럼에서 파티션됩니다.
LINEITEM 테이블은 해쉬되어 올바른 ORDERS 테이블 데이터베이스 파티션으로 전송됩니다.
이 예에서 Join 술어는 다음으로 가정됩니다.
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

그림 19. 내부 테이블 경로지정 조인 예

테이블 대기행렬

테이블 대기행렬은 다음과 같은 경우에 사용됩니다.

- 파티션간 병렬 처리를 사용할 때 테이블 데이터를 하나의 데이터베이스 파티션에서 다른 데이터베이스 파티션으로 전달하는 경우
- 파티션 내 병렬 처리를 사용할 때 데이터베이스 파티션 안에 테이블 데이터를 전달하는 경우

- 하나의 데이터베이스 파티션을 사용할 때 데이터베이스 파티션 안에 테이블 데이터를 전달하는 경우

각 테이블 대기행렬은 한 방향으로 데이터를 전달하는 데 사용됩니다.

컴파일러는 어디에 테이블 대기행렬이 필요한지 판별하고, 이를 플랜에 포함시킵니다. 플랜이 실행되면 데이터베이스 파티션간의 연결이 테이블 대기행렬을 시작합니다. 테이블 대기행렬은 프로세스가 종료될 때 닫힙니다.

다음과 같은 여러 테이블 대기행렬이 있습니다.

- *비동기 테이블 대기행렬*. 이 테이블 대기행렬을 비동기라고도 하며, 이는 응용프로그램이 FETCH를 발행하기 이전에 행을 읽기 때문입니다. FETCH가 발행되면 테이블 대기행렬에서 행이 검색됩니다.

  비동기 테이블 대기행렬은 SELECT문에 FOR FETCH ONLY절을 지정할 때 사용됩니다. 행만을 페치 중인 경우, 비동기 테이블 대기행렬은 보다 빠릅니다.
- *동기 테이블 대기행렬*. 이 테이블 대기행렬을 동기라고도 하며, 이는 응용프로그램이 각 FETCH를 발행할 때 한 행씩 읽기 때문입니다. 각 데이터베이스 파티션에서 커서는 해당 데이터베이스 파티션에서 읽게 될 행의 다음 행에 위치합니다.

  동기 테이블 대기행렬은 SELECT문에 FOR FETCH ONLY절을 지정할 때 사용됩니다. 파티션된 데이터베이스 환경에서 행을 갱신 중인 경우, 데이터베이스 관리 프로그램은 동기 테이블 대기행렬을 사용합니다.
- *테이블 대기행렬 병합*. 테이블 대기행렬은 순서가 있습니다.
- *테이블 대기행렬 병합하지 않음*. 또한 이 테이블 대기행렬을 『일반』 테이블 대기행렬이라고도 합니다. 이 대기행렬은 순서를 지정하지 않습니다.
- *리스너 테이블 대기행렬*. 테이블 대기행렬은 상관 부속 조회로 사용합니다. 상관 값은 부속 조회 아래로 전달되고, 결과는 이 테이블 대기행렬 유형을 사용하여 상위 조회 블록까지 뒤로 전달됩니다.

정렬이 최적화 알고리즘에 미치는 영향

최적화 알고리즘이 액세스 플랜을 선택할 때 데이터의 정렬이 성능에 미치게 될 영향을 고려합니다. 페치된 행의 요청된 순서화를 충족시키는 색인이 없을 경우 정렬이 발생합니다. 색인 스캔보다 비용을 덜 들이기 위해 최적화 알고리즘이 정렬

을 결정하면 정렬이 또한 발생할 수 있습니다. 최적화 알고리즘은 데이터를 정렬할 때 다음 조치 중 하나를 수행할 수 있습니다.

- 조회가 실행될 때 정렬의 『파이핑』 결과. 자세한 내용은 『파이프 정렬 및 비 파이프 정렬』 및 101 페이지의 『조회 최적화에 영향을 주는 구성 매개변수』를 참조하십시오.
- 데이터베이스 관리 프로그램 내에서의 정렬의 내부 처리. 자세한 내용은 『총계 및 정렬 푸시다운 연산자』를 참조하십시오.

파이프 정렬 및 비 파이프 정렬

정렬의 완료 시점에서 데이터가 최종적으로 정렬된 목록을 하나의 순차 패스에서 읽을 수 있는 경우, 결과는 *파이프화된* 것이라고 할 수 있습니다. 파이핑은 정렬 결과를 통신하는 데 다른(파이프화가 아닌) 방법을 사용하는 것보다 더 빠릅니다. 최적화 알고리즘은 가능하면 정렬의 결과 파이프화를 항상 선택합니다.

정렬이 파이프화되었는지의 여부와는 상관없이, 정렬하는 데 걸리는 시간은 정렬될 행 수, 키 *크기* 및 행 폭 등의 여러 인수에 따라 달라집니다. 정렬될 행이 정렬 힙에 사용 가능한 공간보다도 더 많이 차지하면 몇 차례의 정렬 패스가 수행되는데, 각각의 패스는 전체 행 세트의 부속 집합을 정렬하게 됩니다. 각각의 정렬 단계는 버퍼 풀의 임시 테이블에 저장됩니다. (버퍼 풀 관리의 일부로, 페이지를 이 임시 테이블로부터 디스크로 쓰는 것도 가능합니다.) 일단 모든 정렬 패스가 완료되면 이들 정렬된 부속 집합은 하나의 정렬된 행 세트로 병합되어야 합니다. 정렬이 파이프화된 경우, 행은 병합되면서 바로 관계형 데이터 서비스로 넘어갑니다.

자세한 내용은 302 페이지의 『정렬 성능 문제점의 표시기 검토』 또는 101 페이지의 『조회 최적화에 영향을 주는 구성 매개변수』에 있는 *sortheap* 구성 매개변수에 대한 논의를 참조하십시오.

총계 및 정렬 푸시다운 연산자

간혹, 최적화 알고리즘이 관계형 데이터 서비스 구성요소로부터 데이터 관리 서비스 구성요소로 정렬 또는 총계 조작을 푸시다운하도록 선택할 수 있습니다. 이러한 조작을 푸시다운하면 데이터 관리 서비스 구성요소가 데이터를 곧바로 정렬 또는 총계 루틴으로 전달할 수 있으므로 성능이 향상됩니다. 이렇게 푸시다운하지 않으면 데이터 관리 서비스는 먼저 이 데이터를 관계형 데이터 서비스로 전달하고 관

계형 데이터 서비스는 다시 정렬 또는 총계 루틴과 인터페이스를 하게 됩니다. 예를 들어, 다음 조회는 이런 최적화로부터 이점을 얻게 됩니다.

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
     FROM   EMPLOYEE
GROUP BY WORKDEPT
```

정렬에서의 총계

GROUP BY 조작에 대해 필요한 순서를 생성하는 데 정렬이 사용될 경우, 최적화 알고리즘은 정렬을 수행하는 동안 GROUP BY 총계의 일부나 전체를 수행하는 옵션을 수반합니다. 이것이 각 그룹의 행 수가 큰 경우의 이점입니다. 이는 정렬하는 동안 그룹화를 수행하여 디스크로 정렬을 유출해야 하는 필요성이 감소되거나 없어진 경우에 더욱 유리합니다.

정렬에서의 총계가 사용될 경우, 적절한 결과가 계산되도록 하기 위해 필요한 최대 세 개의 총계 단계가 있습니다. 총계의 첫 번째 단계인『부분 총계』는 정렬 힙이 채워질 때까지 총계 값을 계산합니다. 부분 총계는 총계되지 않은 데이터가 취해지고 부분 총계가 생성되는 프로세스입니다. 정렬 힙이 채워질 경우, 나머지 데이터는 디스크로 유출되고 정렬 힙의 현재 채우기에서 계산된 모든 부분 총계를 포함합니다. 정렬 힙의 재설정 이후에 새 총계가 시작됩니다.

총계의 두 번째 단계인『중간 총계』에서는 유출된 모든 정렬 수행을 취하고 그룹화 키에 대해 추가로 총계합니다. 그룹화 키 컬럼은 파티션 키 컬럼의 부속 집합이므로, 총계가 완료될 수 없습니다. 중간 총계에서는 기존의 부분 총계를 취하고 새 부분 총계를 생성합니다. 이 단계는 생략 가능한 단계로, 파티션 내 병렬 처리와 파티션간 병렬 처리에 모두 사용됩니다. 마지막 경우에 그룹화는 전역 그룹화 키가 사용 가능할 때 완료됩니다. 파티션간 병렬 처리에서 이는 그룹화 키가 파티션에서 그룹을 나누는 파티션 키의 부속 집합이므로 다시 파티션하며 총계를 완료해야 할 경우에 발생합니다. 총계를 완료하기 위해 단일 에이전트로 줄이기 전에 각 에이전트가 해당되는 유출된 정렬 수행의 병합을 완료할 때 유사한 경우가 파티션 내 병렬 처리에 존재합니다.

총계의 마지막 단계인『최종 총계』에서는 모든 부분 총계를 취하고 총계를 완료합니다. 최종 총계는 부분 총계에서 취해지며 최종 총계를 생성합니다. 이 단계는 항상 GROUP BY 연산자에서 발생합니다. 정렬이 유출되지 않도록 하는 확실한 방법이 없으므로, 정렬은 전체 총계를 수행할 수 없습니다. 전체 총계는 총계되지 않

은 데이터에서 취해지며 최종 총계를 생성합니다. 이러한 총계 방법은 보통 이미 올바른 순서로 되어 있는 데이터를 그룹화하고 파티션이 이 방법의 사용을 방해할 때 사용됩니다.

파티션 내 병렬 처리에 대한 최적화 전략

최적화 알고리즘은 액세스 플랜을 선택하여 SQL문이 컴파일될 때 병렬 처리 수준이 지정되는 경우, 데이터베이스 파티션 내에서 조회는 병렬로 실행됩니다.

실행시 『서브에이전트』인 다중 데이터베이스 에이전트는 조회를 실행하기 위해 작성됩니다. 서브에이전트 수는 SQL문이 컴파일된 경우에 결정된 병렬 처리 수준 이하입니다. SQL문에 대한 병렬 처리 수준을 설정에 대한 자세한 내용은 98 페이지의 『응용프로그램의 병렬 처리』를 참조하십시오. 에이전트 및 서브에이전트에 대한 자세한 내용은 311 페이지의 『데이터베이스 에이전트』를 참조하십시오.

파티션된 데이터베이스에서 병렬 처리 수준은 파티션에 적용합니다. 예를 들어, 주어진 데이터베이스 파티션에서 실행 중인 조회 부분은 해당 SQL문에 대한 데이터베이스 파티션에서 결정된 병렬 처리 수준에 기초하여 한층 병렬 처리됩니다.

액세스 플랜은 각 서브에이전트 및 조정 에이전트가 수행하는 부분으로 액세스 플랜을 분리하여 병렬 처리됩니다. 서브에이전트는 테이블 대기행렬을 통해 데이터를 조정 에이전트 또는 다른 서브에이전트로 전달합니다. 파티션된 데이터베이스에서 서브에이전트는 다른 데이터베이스 파티션의 서브에이전트로부터 테이블 대기행렬을 통해 데이터를 수신하거나 전송합니다.

이 절에서는 하나의 데이터베이스 파티션 내의 병렬 처리 전략에 대해 설명합니다.

병렬 스캔 전략

관계형 스캔 및 색인 스캔은 같은 테이블이나 색인에서 병렬로 수행될 수 있습니다. 병렬 관계형 스캔의 경우, 테이블은 행 또는 페이지 범위로 분리됩니다. 행 또는 페이지 범위는 서브에이전트에 지정됩니다. 서브에이전트는 지정된 범위를 스캔하고, 현재 범위에서 작업이 완료되면 다른 범위로 지정됩니다.

병렬 색인 스캔의 경우, 색인은 색인 키 값과, 키 값에 대한 색인 항목 수에 기초한 레코드 범위로 분리됩니다. 병렬 색인 스캔은 레코드 범위로 지정된 서브에이전

트를 가진 병렬 테이블 스캔 같이 계속 진행합니다. 서브에이전트는 현재 범위에서 작업이 완료되면 새 범위로 지정됩니다.

스캔 단위(페이지 또는 행)와 스캔 세분성은 최적화 알고리즘에 의해 결정됩니다.

병렬 스캔은 서브에이전트간의 작업 분산도 제공합니다. 병렬 스캔의 목적은 서브에이전트간의 로드 균형을 맞추고 로드를 똑같이 사용 중인 상태로 유지하는 것입니다. 사용 중인 서브에이전트 수가 사용 가능한 프로세서 수와 같고 디스크에서 I/O 요청으로 과도하게 작업하지 않은 경우, 머신 자원은 효율적으로 사용되고 있습니다.

다른 액세스 플랜 조작으로 조회가 실행될 때 데이터 불균형이 발생할 수 있습니다. 최적화 알고리즘은 병렬 전략을 선택하여 데이터 균형이 유지보수됩니다.

병렬 정렬 전략

최적화 알고리즘은 다음 병렬 정렬 전략 중 하나를 선택할 수 있습니다.

라운드 로빈 정렬

이 정렬을『재분산 정렬』이라고도 합니다. 모든 서브에이전트에서 가능하면 균등하게 데이터를 재분산하려고 하는 효율적인 공유 메모리 정렬입니다. 라운드 로빈 알고리즘을 사용하여 균등한 분산을 제공합니다. 먼저 각 서브에이전트에 대한 개별 정렬을 작성합니다. 삽입 단계 중에 서브에이전트는 라운드 로빈 방식으로 개별 정렬을 각각 삽입합니다. 데이터의 분산을 좀더 균등하게 할 수 있습니다.

파티션된 정렬

이 정렬은 각 서브에이전트에 대해 작성된 정렬의 라운드 로빈 정렬과 유사합니다. 서브에이전트는 해쉬 함수를 정렬 컬럼에 적용하여 행이 삽입되어야 하는 정렬을 결정합니다. 예를 들어, 내부 및 외부 병합 조인이 파티션된 정렬인 경우, 서브에이전트는 병합 조인을 사용하여 해당 파티션을 조인할 수 있습니다. 병합 조인이 병렬로 실행될 수도 있습니다.

복제된 정렬

이 정렬은 모든 서브에이전트에 모든 정렬 출력이 필요한 경우 사용됩니다. 하나의 정렬이 작성되고 서브에이전트는 정렬로의 삽입 과정 중 동기화됩니다. 정렬이

완료되면 각 서브에이전트는 전체 정렬을 읽습니다. 이 정렬을 사용하면 행 수가 적은 경우 데이터 스트림의 균형을 다시 맞출 수 있습니다.

공유 정렬

이 정렬은 서브에이전트가 정렬된 결과에서 병렬 스캔을 연다는 점을 제외하면 복제된 정렬과 같은 정렬입니다. 여기서는 라운드 로빈 정렬과 유사한 방식으로 서브에이전트간에 데이터를 분산합니다.

병렬 임시 테이블

같은 테이블에 행을 삽입하여 임시 테이블을 생성하기 위해 서브에이전트는 같이 작동할 수 있습니다. 이것을 공유 임시 테이블이라고 합니다. 서브에이전트는 데이터 스트림이 복제되었는지 아니면 파티션되었는지에 따라 공유 임시 테이블에서 전용 스캔 또는 병렬 스캔을 열 수 있습니다.

병렬 총계 전략

총계 조작은 서브에이전트에 의해 병렬로 수행될 수 있습니다. 총계 조작에서 데이터는 그룹화 컬럼에 순서화되어야 합니다. 서브에이전트가 그룹화 컬럼 값 세트의 모든 행을 확실히 수신할 수 있으면 전체 총계가 수행됩니다. 이는 이전에 파티션된 정렬로 인해 스트림이 그룹화 컬럼에 이미 파티션되어 있을 경우 발생합니다.

그렇지 않으면 서브에이전트가 부분 총계를 수행하고 다른 전략을 사용하여 총계를 완료할 수 있습니다. 이러한 전략 중 일부는 다음과 같습니다.

- 부분적으로 총계된 데이터를 테이블 대기행렬 병합을 통해 조정자 에이전트로 전송하십시오. 조정자(coordinator)는 총계를 완료합니다.
- 부분적으로 총계된 데이터를 파티션된 정렬에 삽입하십시오. 정렬은 그룹화 컬럼에서 파티션됩니다. 이는 그룹화 컬럼 세트의 모든 행이 하나의 정렬 파티션에 들어 있음을 보장합니다.
- 균형을 맞추기 위해 스트림이 복제되어야할 경우, 부분적으로 총계된 데이터는 복제된 정렬에 삽입될 수 있습니다. 각 서브에이전트는 복제된 정렬을 사용하여 총계를 완료하고 총계 결과와 동일한 사본을 수신합니다.

병렬 조인 전략

조인 조작은 서브에이전트에 의해 병렬로 수행될 수 있습니다. 병렬 조인 전략은 데이터 스트림의 특성에 따라 결정됩니다.

조인은 파티션이나 또는 조인의 내부 및 외부 테이블에서 이루어지는 데이터 스트림 복제에 의해 병렬 처리될 수 있습니다. 예를 들어, 해당 외부 스트림이 병렬 스캔으로 인해 파티션되고 내부 스트림이 각 서브에이전트와 상관없이 재평가된 경우, 중첩된 루프 조인은 병렬 처리될 수 있습니다. 해당 조인의 내부 및 외부 스트림이 파티션된 정렬로 인해 값이 파티션된 경우, 병합된 조인은 병렬 처리될 수 있습니다.

자동 요약 테이블

요약 테이블은 조회 응답 시간을 개선하기 위한 강력한 방법입니다. 일부 기본 조회 구조가 참여할 수 있는 많은 환경에서 요약 테이블을 사용하여 다음을 수행할 수 있습니다.

- 하나 이상의 차원에서의 데이터 총계
- 테이블 그룹에서의 데이터 조인 및 총계
- 공통으로 액세스되는 데이터 부속 집합 식별(즉, 『hot』 수평 또는 수직 파티션)
- 파티션된 데이터베이스 환경에서 테이블이나 테이블 부분 재파티션

요약 테이블에 대한 지식은 SQL 컴파일러에 통합됩니다. SQL 컴파일러 내에서 조회 재작성(171 페이지의 『SQL 컴파일러에 의한 조회 재작성』 참조) 및 최적화 알고리즘(183 페이지의 『데이터 액세스 개념 및 최적화』 참조)은 조회를 요약 테이블과 대응하여 기본 테이블에 액세스하는 조회용 요약 테이블을 대체할 것인지를 판별할 때 관련됩니다. 요약 테이블을 사용하여 조회에 응답할 때마다 EXPLAIN 기능(243 페이지의 『제7장 SQL Explain 기능』 참조)을 사용하여 선택된 요약 테이블을 판별할 수 있습니다. 요약 테이블은 여러 방식으로 일반 테이블과 같이 작동하므로, 테이블 공간 정의를 사용한 데이터 액세스 최적화, 색인 작성 및 RUNSTATS 발행에 대한 동일한 고려사항이 요약 테이블에 적용됩니다.

요약 테이블의 강력함을 쉽게 이해할 수 있도록 하기 위해 다음과 같은 다차원 분석 조회 예와 이 요약 테이블을 이용하는 방법을 보여줍니다.

이 예에서는 웨어하우스에 일련의 고객들과 일련의 신용 카드 계정이 있는 시나리오를 가정해 보십시오. 웨어하우스는 신용 카드로 이루어지는 일련의 거래를 기록합니다. 각 거래에는 함께 구입되는 항목 세트가 포함됩니다. 이는 거래 항목을 포함하는 하나의 테이블과 구입 거래를 식별하는 다른 테이블이 크고, 더불어 스타의 허브이기 때문에 이 환경을 멀티 스타로 범주화합니다.

거래를 설명하는 세 개의 계층 구조 차원인 제품, 위치 및 시간이 있습니다. 제품 계층 구조는 제품 그룹과 제품군을 나타내는 두 개의 표준화된 테이블에 기록됩니다. 위치 계층 구조에는 시, 도, 국가 정보가 포함되며, 이는 하나의 표준화되지 않은 테이블로 표시됩니다. 시간 계층 구조에는 년, 월, 일 정보가 포함되고 하나의 날짜 필드로 인코딩됩니다. 날짜 차원은 내장 함수를 사용하여 거래의 날짜 필드에서 추출됩니다. 고객 및 고객 정보에 대한 계정 정보를 나타내는 다른 테이블도 이 시나리오에 있습니다.

요약 테이블은 다음의 각 레벨에 대한 판매 합계와 판매 수로 작성됩니다.

- 제품 계층
- 위치 계층
- 시간 계층(년, 월, 일로 구성됨)

광범위한 조회에서 이 저장된 총계 데이터로부터의 응답을 선택할 수 있습니다. 다음 예에서는 제품 그룹 및 제품군 차원을 따라, 시, 도, 국가 차원을 따라 그리고 시간 차원을 따라 판매 합계와 판매 횟수를 계산합니다. GROUP BY절에 몇 개의 다른 컬럼도 포함됩니다.

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
             loc.country, loc.state, loc.city,
             l.name AS linename, pg.name AS pgname,
             YEAR(pdate) AS year, MONTH(pdate) AS month,
             t.status,
             SUM(ti.amount) AS amount,
             COUNT(*) AS count
      FROM   cube.transitem AS ti, cube.trans AS t,
             cube.loc AS loc, cube.pgroup AS pg,
             cube.prodline AS l
      WHERE  ti.transid = t.id
         AND ti.pgid = pg.id
         AND pg.lineid = l.id
```

```
         AND t.locid = loc.id
         AND YEAR(pdate) > 1990
       GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
                year(pdate), month(pdate), t.status, l.name, pg.name
     )
DATA INITIALLY DEFERRED REFRESH DEFERRED;
REFRESH TABLE dba.SALESCUBE;
```

요약 테이블은 보통 기본 사실 테이블보다 더 작습니다. 예에서 처럼 DEFERRED 옵션을 지정하여 요약 테이블을 새로 고칠 시기를 제어할 수 있습니다.

이러한 사전에 계산된 합계를 이용할 수 있는 조회는 다음과 같습니다.

- 월 및 제품 그룹별 판매
- 1990년 이후 연도에 대한 총 판매액
- 1995 또는 1996년의 판매
- 제품 그룹 또는 제품군에 대한 판매 합계
- 특정 제품 그룹 또는 제품군 그리고 1995, 1996년에 대한 판매 합계
- 특정 국가에 대한 판매 합계

정확한 응답이 이러한 조회 중 하나에 대한 요약 테이블에 포함되지 않는 반면, 응답의 부분이 이미 계산되어 있으므로 요약 테이블을 사용하여 응답을 계산하는 비용이 대행 기본 테이블을 사용하는 것보다 적게 듭니다. 비용이 많이 드는 기본 데이터의 조인, 정렬 및 총계는 요약 테이블을 통해 피하거나 감소됩니다.

다음은 중요한 성능 향상을 가져오는 샘플 조회입니다. 이들 조회는 이미 계산된 요약 테이블의 결과를 사용할 수 있기 때문입니다. 첫 번째 예에서는 1995년과 1996년의 총 판매액을 리턴합니다.

```
SET CURRENT REFRESH AGE=ANY
SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
      FROM   cube.transitem AS ti, cube.trans AS t,
             cube.loc AS loc, cube.pgroup AS pg,
             cube.prodline AS l
      WHERE  ti.transid = t.id
         AND ti.pgid = pg.id
         AND pg.lineid = l.id
         AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

두 번째 예에서는 1995년과 1996년의 제품 그룹별 총 판매액을 리턴합니다.

```
SET CURRENT REFRESH AGE=ANY
SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
      FROM   cube.transitem AS ti, cube.trans AS t,
             cube.loc AS loc, cube.pgroup AS pg,
             cube.prodline AS l
      WHERE  ti.transid = t.id
         AND ti.pgid = pg.id
         AND pg.lineid = l.id
         AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

이러한 조회에 대한 응답 시간을 더 개선하면 더 많은 데이터베이스에서 아카이브 될 수 있습니다. 이는 기본 테이블의 증가보다 요약 테이블의 증가가 더 느리기 때문에 발생합니다. 요약 테이블의 이점 중 하나는 요약 테이블을 빌드하고 아주 많은 조회에 대해 그 내용을 재사용할 때 한 번 계산을 수행하여 DB2 Universal Database가 이 테이블을 통해 조회 사이의 중첩되는 작업을 효율적으로 제거하는 것입니다.

연합 데이터베이스 조회 컴파일러 단계

이 절에서는 연합 데이터베이스 시스템 내의 추가 조회 처리 단계에 대해 설명합니다. 이는 연합 데이터베이스의 조회 성능을 향상시키기 위한 권장사항을 제공합니다. 주요 주제에는 다음 내용이 포함됩니다.

- 『분석 푸시다운』
- 235 페이지의 『원격 SQL 생성 및 전역 최적화』

분석 푸시다운

분석 푸시다운은 DB2 최적화 알고리즘에 원격 데이터 소스에서 조작을 수행할 수 있는지 여부를 알려줍니다. 조작은 관계형 연산자, 시스템 또는 사용자 함수 또는 SQL 연산자(GROUP BY, ORDER BY 등)와 같은 함수일 수 있습니다.

푸시다운될 수 없는 함수는 조회 성능에 많은 영향을 미칩니다. 데이터 소스에서가 아닌 지역적으로 평가되는 선택적 술어를 강제하는 영향을 고려해 보십시오. 이 접근 방법에는 원격 데이터 소스로부터 전체 테이블을 검색하고 술어에 대해 지역

적으로 필터하려면 DB2가 있어야 합니다. 네트워크에 제한조건이 있고 테이블이 큰 경우, 조회 성능이 저하될 수 있습니다.

푸시다운되지 않은 연산자는 조회 성능에 많은 영향을 미칠 수 있습니다. 예를 들어, 원격 데이터를 지역적으로 총계하는 GROUP BY 연산자가 있을 경우, 원격 데이터 소스로부터 전체 테이블을 검색하려면 DB2가 있어야 합니다.

예를 들어, 별칭 N1이 OS/390용 DB2 데이터 소스에 있는 EMPLOYEE 데이터 소스 테이블을 참조한다고 가정해 보십시오. 또한 테이블에 10,000행이 들어 있으며, 컬럼 중 하나에 사원의 성이 들어 있고, 다른 컬럼에는 월급이 들어 있다고 가정해 보십시오. 제공된 명령문은 다음과 같습니다.

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

여러 가능성을 고려해 볼 수 있습니다.

- DB2 및 OS/390용 DB2에서의 조합 순서가 동일한 경우, 조회 술어가 OS/390용 DB2로 푸시다운될 가능성이 있습니다. 일반적으로, 전체 테이블을 DB2로 복사하여 조작을 지역적으로 수행하는 것보다 데이터 소스에서 결과를 필터하고 그룹화하는 것이 훨씬 효율적입니다. 연합 시스템 내에서 분석 푸시다운은 데이터 소스에서 조작을 수행할 수 있는지 여부를 판별합니다. 이런 경우, 술어 및 GROUP BY 조작은 데이터 소스에서 발생할 수 있습니다.
- 조합 순서가 다를 경우, 분석 푸시다운은 데이터 소스에서 전체 술어를 평가할 수 없는지 여부를 판별합니다. 그러나 최적화 알고리즘은 술어의 `SALARY > 50000` 부분을 푸시다운하도록 결정할 수 있습니다. 범위 비교는 계속해서 DB2에서 수행되어야 합니다.
- 조합 순서가 동일하고 최적화 알고리즘이 지역 DB2 서버가 매우 빠르다는 것을 알고 있는 경우, 최적화 알고리즘은 DB2에서 `GROUP BY` 조작을 지역적으로 수행하는 것이 최상의 접근 방법(최소의 비용)임을 결정할 수 있습니다. 술어는 데이터 소스에서 평가됩니다. 이는 전역 최적화와 결합된 분석 푸시다운의 한 예입니다. DB2는 사용 가능한 경로를 고려한 다음 가장 효율적인 플랜을 선택합니다.

일반적으로, 목표는 함수 및 연산자가 최적화 알고리즘에 의해 데이터 소스에서 평가될 수 있도록 하는 것입니다. 함수 및 SQL 연산자가 원격 데이터 소스에서 평가될 수 있는지 여부를 결정하는 데 여러 인수가 영향을 미칩니다. 핵심 인수는 세 개의 그룹(서버 특성, 별칭 특성 및 조회 특성)에서 논의됩니다.

푸시다운 기회에 영향을 미치는 서버 특성

다음 절에는 푸시다운 기회에 영향을 미칠 수 있는 데이터 소스 특정 인수가 들어 있습니다. 일반적으로, DB2에서는 풍부한 SQL dialect를 사용하여 조회를 제출할 수 있기 때문에 이러한 인수가 존재합니다. 이 dialect는 DB2 조회 중에 액세스되는 서버가 지원하는 SQL dialect보다 더 많은 기능을 제공할 수 있습니다. DB2는 데이터 서버에서 부족한 함수를 보충할 수 있으나, 그렇게 하려면 조작이 DB2에서 발생해야 합니다.

SQL 기능: 각 데이터 소스는 변형된 SQL dialect 및 여러 기능 레벨을 지원합니다. 예를 들어, GROUP BY 목록을 고려해 보십시오. 대부분의 데이터 소스는 GROUP BY 연산자를 지원합니다. 일부는 GROUP BY 목록의 항목 수 및 GROUP BY 목록에 표현식이 허용되는지에 대한 제한사항이 있습니다. 원격 데이터 소스에 제한사항이 있으면 DB2는 지역적으로 GROUP BY 조작을 수행해야 할 수 있습니다.

SQL 제한사항: 각 데이터 소스는 각기 다른 SQL 제한사항을 가질 수 있습니다. 예를 들어, 일부 데이터 소스에는 값에서 원격 SQL문으로 바인드하기 위해 매개변수 표시문자가 있어야 합니다. 그러므로 각 데이터 소스가 그러한 바인드 메커니즘을 지원할 수 있으려면 매개변수 표시문자 제한사항이 선택되어 있어야 합니다. DB2가 함수에 대한 값에서 바인드할 좋은 방법을 결정하지 못하면 이 함수는 지역적으로 평가되어야 합니다.

SQL 한계: DB2은 원격 데이터 소스보다 큰 정수의 사용을 허용합니다. 그러나 한계를 초과하는 값은 명령문에 내포되어 데이터 소스로 전송될 수 없습니다. 따라서 이 상수에서 조작하는 함수 또는 연산자는 지역적으로 평가되어야 합니다.

서버 특정사항: 여러 인수가 이 범주에 해당합니다. 하나의 예로 NULL 값 정렬을 들 수 있습니다(최고, 최저 또는 순서화에 따라). 예를 들어, NULL 값이 데이터 소스와 DB2에서 다르게 정렬될 경우, 널(NULL) 입력 가능 표현식에서의 ORDER BY 조작은 원격으로 평가될 수 없습니다.

조합 순서: 데이터 소스가 사용하는 것과 동일한 조합 순서를 사용하도록 연합 데이터베이스를 구성한 다음 *collating_sequence* 서버 옵션을 'Y'로 설정하면 최적화 알고리즘은 푸시다운 문자 범위 비교 술어를 고려할 수 있습니다.

연합 서버의 조회에 정렬이 필요한 경우, 정렬이 처리되는 곳은 서버 인수에 따라 다릅니다. 연합 데이터베이스의 조합 순서가 조회 데이터가 저장되는 데이터 소스의 조합 순서와 다른 경우, 정렬은 데이터 소스에서 발생할 수 있습니다. 조합 순서가 동일하면 최적화 알고리즘은 지역 정렬과 데이터 소스에서의 정렬 중 어느 것이 조회를 완료하는 데 효율적인지를 판별할 수 있습니다. 마찬가지로, 조회에 문자 데이터의 비교가 필요하면 이 비교는 데이터 소스에서도 수행될 수 있습니다.

일반적으로, 숫자 비교는 조합 순서가 다르더라도 두 곳에서 수행될 수 있습니다. 그러나 널(NULL) 문자의 가중치가 연합 데이터베이스와 데이터 소스 사이에 다르면 다른 결과가 발생할 수도 있습니다. 마찬가지로, 비교문에서 대소문자 비 구분 데이터 소스로 명령문을 제출할 경우에는 주의를 기울여야 합니다. 대소문자 비 구분 데이터 소스에서 문자 ″I″와 ″i″에 지정된 가중치는 동일합니다. 기본적으로 DB2는 대소문자를 구별하며 문자에 다른 가중치를 지정합니다.

연합 데이터베이스 및 데이터 소스의 조합 순서가 다르면 DB2는 연합 데이터베이스에 대해 데이터를 검색하므로 지역적으로 정렬 및 비교를 수행할 수 있습니다. 이유는 연합 서버에 대해 정의된 조합 순서에 따라 순서화된 조회 결과를 보기를 원하기 때문입니다. 그런 다음 연합 서버는 데이터를 지역적으로 순서화하여 결과가 예상한 대로임을 확인합니다.

지역 정렬 및 비교를 위해 데이터를 검색하면 일반적으로 성능이 저하됩니다. 따라서 연합 데이터베이스를 구성하여 데이터 소스가 사용하는 것과 동일한 조합 순서를 사용하는 것을 고려해 보십시오. 이 방법을 사용하면 성능이 향상되는데, 이유는 연합 서버는 정렬 및 비교가 데이터 소스에서 발생하는 것을 허용하기 때문입니다. 예를 들어, OS/390용 DB2 UDB에서 ORDER BY절로 정의된 정렬은 EBCDIC 코드 페이지를 기본으로 하는 조합 순서로 구현됩니다. 연합 서버를 사용하여 ORDER BY절에 따라 정렬된 OS/390용 DB2 데이터를 검색하려면 EBCDIC 코드 페이지를 기본으로 하는 사전 정의된 조합 순서를 사용하도록 연합 데이터베이스를 구성하는 것이 좋습니다.

연합 데이터베이스 및 데이터 소스에서의 조합 순서가 다를 때 데이터 소스의 순서에 따라 순서화된 데이터를 보아야 할 경우, 통과 모드에서 조회를 제출하거나 데이터 소스 뷰에서 조회를 정의할 수 있습니다.

조합 순서 및 설정 방법에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오. *collating_sequence* 서버 옵션에 대한 자세한 내용은 120 페이지의 표8을 참조하십시오.

서버 옵션: 여러 서버 옵션이 푸시다운 기회에 영향을 미칠 수 있습니다. 특히, *collating_sequence*, *varchar_no_trailing_blanks* 및 *푸시다운*에 대한 설정을 검토하십시오. 이 옵션의 설정에 대한 자세한 내용은 119 페이지의 『연합 데이터베이스 조회에 영향을 미치는 서버 옵션』을 참조하십시오.

DB2 유형 맵핑 및 함수 맵핑 인수: 각 데이터 소스 데이터 유형에 충분한 버퍼 공간을 제공하도록(데이터 유실을 방지하도록) DB2가 제공하는 기본 지역 데이터 유형 맵핑(데이터 유형 테이블은 *응용프로그램 개발 안내서* 참조)이 설계되었습니다. 사용자는 특정 응용프로그램에 맞게 특정 데이터 소스의 유형 맵핑을 사용자 정의하도록 선택할 수 있습니다. 예를 들어, DATE 데이터 유형을 가진 Oracle 데이터 소스 컬럼에 액세스할 경우(기본적으로 이는 DB2 TIMESTAMP 데이터 유형에 맵핑됨), 지역 데이터 유형을 DB2 DATE 데이터 유형으로 변경할 수 있습니다.

DB2는 데이터 소스가 지원하지 않는 함수를 보충할 수 있습니다. 다음 세 가지 경우에 함수 보충이 발생합니다.

- 이 함수가 단순히 원격 데이터 소스에 존재하지 않습니다.
- 이 함수가 존재하지 않습니다. 그러나 피연산자의 특성이 함수 제한사항을 위반합니다. 이러한 상황의 예로는 IS NULL 관계형 연산자가 있습니다. 대부분의 데이터 소스가 이를 지원하나, 일부는 제한사항이 있습니다(예: IS NULL 연산자의 왼쪽에만 컬럼 이름을 허용).
- 원격으로 평가될 경우, 함수가 다른 결과를 리턴할 수 있습니다. 이러한 상황의 예로는 '>'(보다 큼) 연산자가 있습니다. 조합 순서가 다른 데이터 소스의 경우, '>' 연산자는 DB2에 의해 지역적으로 평가될 때와 다른 결과를 리턴할 수 있습니다.

푸시다운 기회에 영향을 미치는 별칭 특성

다음 절에는 푸시다운 기회에 영향을 미칠 수 있는 별칭 특정 인수가 들어 있습니다.

별칭 컬럼의 지역 데이터 유형: 컬럼의 지역 데이터 유형 때문에 데이터 소스에서 술어가 평가되지 못하는지 확인하십시오. 앞에서 언급했듯이, 오버플로우의 가능성을 막기 위해 기본 데이터 유형 맵핑이 제공됩니다. 그러나 DB2가 더 긴 컬럼에 바인드하는 방법에 따라 길이가 다른 두 컬럼간에 술어를 조인하는 것은 조인 컬럼이 더 짧은 데이터 소스에서 고려되지 않을 수도 있습니다. 이러한 상황은 DB2 최적화 알고리즘에 의해 평가되는 조인 순서의 확률 수에 영향을 미칠 수 있습니다. 예를 들어, INTEGER 또는 INT 데이터 유형을 사용하여 작성된 Oracle 데이터 소스 컬럼에는 NUMBER(38) 유형이 제공됩니다. 이 Oracle 데이터 유형에 대한 별칭 컬럼에는 FLOAT 지역 데이터 유형이 제공되는데, 이유는 DB2 정수의 범위가 NUMBER(9)에 거의 근접한 2**31 - (-2**31)이기 때문입니다. 이런 경우, DB2 데이터 소스(짧은 조인 컬럼)에서는 DB2 정수 컬럼 및 Oracle 정수 컬럼간의 조인이 발생하지 않습니다. 그러나 이 Oracle 정수 컬럼의 도메인은 DB2 INTEGER 데이터 유형에서는 수용될 수 있으며, ALTER NICKNAME문으로 지역 데이터 유형을 변경하여 DB2 데이터 소스에서 조인이 발생하게 할 수 있습니다.

컬럼 옵션: ALTER NICKNAME SQL문은 별칭에 대한 컬럼 옵션을 추가하거나 변경하는 데 사용될 수 있습니다.

이 옵션 중 하나는 ″varchar_no_trailing_blanks″입니다. 이는 뒤 공백이 없는 컬럼을 식별하는 데 사용됩니다. 컴파일러 분석 푸시다운 단계에서는 이렇게 표시된 컬럼에서 수행되는 모든 조작을 점검할 때 이 정보를 계정으로 취합니다. 이 표시에 따라 DB2는 데이터 소스로 전송될 원격 SQL문에 사용할 동등한 양식을 갖는 여러 가지 술어를 생성할 수 있습니다. 데이터 소스에서 여러 가지 술어가 평가되나 네트 결과는 동일함을 알 수 있습니다.

다른 컬럼 옵션은 numeric_string입니다. 해당 컬럼의 값이 항상 뒤 공백이 없는 숫자인지를 나타내려면 이 컬럼을 사용하십시오.

컬럼 옵션 값 및 기본에 대한 자세한 내용은 232 페이지의 표15를 참조하십시오.

표 15. 컬럼 옵션 및 설정값

옵션	유효한 설정값		기본 설정값
numeric_string	'Y'	예, 이 컬럼은 숫자 데이터로의 문자열을 포함합니다. 중요: 이 컬럼이 뒤 공백을 가진 숫자 문자열을 포함할 경우에는 'Y'를 지정하지 않는 것이 좋습니다.	'N'
	'N'	아니오, 이 컬럼은 숫자 데이터의 문자열에 제한되지 않습니다.	
	numeric_string을 컬럼에 대해 'Y'로 설정하면 최적화 알고리즘에 이 컬럼은 컬럼 데이터의 정렬과 인터페이스될 수 있는 공백을 포함하지 않음을 알려줍니다. 이 옵션은 데이터 소스의 조합 순서가 DB2와 다를 경우에 유용합니다. 이 옵션으로 표시된 컬럼은 다른 조합 순서로 인해 지역(데이터 소스) 평가에서 제외되지 않습니다.		
varchar_no_trailing_blanks	이 데이터 소스가 공백이 아닌 문자로 채워진 varchar 비교 의미를 사용할지 여부를 지정합니다. 뒤 공백이 없는 변수 길이 문자열의 경우, 일부 DBMS의 공백이 아닌 문자로 채워진 비교 의미는 DB2의 비교 의미와 동일한 결과를 리턴합니다. 데이터 소스에 있는 모든 VARCHAR 테이블/뷰 컬럼에 뒤 공백이 포함되어 있지 않으면 이 서버 옵션을 데이터 소스에 대해 'Y'로 설정하십시오. 이 옵션은 종종 Oracle** 데이터 소스에도 사용됩니다. 별칭(뷰를 포함하여)을 사용할 가능성이 있는 모든 오브젝트도 고려하도록 하십시오.		'N'
	'Y'	이 데이터 소스는 DB2와 유사한 공백이 아닌 문자로 채워진 비교 의미를 가집니다.	
	'N'	이 데이터 소스는 DB2와 동일한 공백이 아닌 문자로 채워진 비교 의미를 갖지 않습니다.	

푸시다운 기회에 영향을 미치는 조회 특성

조회는 여러 데이터 소스에 있는 별칭을 포함하는 SQL 연산자를 참조할 수 있습니다. DB2가 하나의 연산자를 사용하여 두 개의 참조 데이터 소스로부터 발생한 결과를 결합해야 하는 경우(예: UNION과 같은 집합 연산자), 조작은 DB2에서 발생해야 합니다. 연산자는 원격 데이터 소스에서는 직접 평가될 수 없습니다.

분석 푸시다운 결정 분석 및 이해

SQL문을 재작성하여 DB2 조회 처리에 추가 푸시다운 기회를 제공할 수 있습니다. 이 절에서는 조회가 평가될 위치를 결정하는 도구를 소개하고, 조회 분석과 연관된 일반적인 질문(그리고 연구하도록 제안되는 분야) 및 데이터 소스 업그레이드에 대해 간략히 설명합니다.

조회가 평가되는 위치 분석: DB2에는 조회가 평가되는 위치를 보여주는 두 개의 유틸리티가 제공됩니다.

- Visual Explain. **db2cc** 또는 **db2vexp** 명령으로 시작하십시오. 조회 액세스 플랜 그래프를 보려면 이 명령을 사용하십시오. 각 연산자의 실행 위치는 연산자에 대한 세부사항 화면에 포함되어 있습니다.

  조회가 완전히 푸시다운되면 RQUERY 연산자의 맨 위에 RETURN 연산자가 있어야 합니다. RETURN 연산자는 표준 DB2 연산자입니다. RQUERY 연산자는 연합 데이터베이스 조작에 대해 고유합니다. RQUERY는 SQL SELECT 문을 데이터 소스로 전송하여 조회 결과를 검색합니다. SELECT문은 데이터 소스에서 지원하는 SQL dialect를 사용하여 생성됩니다. 해당 데이터 소스에 대해 유효한 조회를 포함할 수 있습니다.

- SQL Explain. **db2expln** 또는 **dynexpln** 명령으로 시작하십시오. 액세스 플랜 전략을 텍스트로 보려면 이 명령을 사용하십시오.

조회가 데이터 소스 또는 DB2에서 평가되는 이유 이해: 이 절에는 일반적인 플랜 분석 질문과 푸시다운 기회를 늘리기 위해 연구해야 할 분야가 나와 있습니다. 핵심 질문은 다음과 같습니다.

- 이 술어를 원격으로 평가할 수 없는 이유는 무엇입니까?

  이 질문은 술어가 매우 선택적이어서 행을 필터하는 데, 그리고 네트워크 통신량을 줄이는 데 사용될 경우에 제기됩니다. 원격 술어 평가는 동일한 데이터 소스에 있는 두 테이블간의 조인이 원격으로 평가될 수 있는지 여부에도 영향을 미칩니다.

  조사해야 할 분야는 다음과 같습니다.

- 부속 조회 술어. 술어에 다른 데이터 소스에 관련된 부속 조회가 들어 있습니까? 이 술어에 이 데이터 소스에서 지원되지 않는 SQL 연산자를 포함하는 부속 조회가 들어 있습니까? 모든 데이터 소스가 술어 내에서 집합 연산자를 지원하는 것은 아닙니다.
- 술어 함수. 이 술어에 이 원격 데이터 소스에 의해 평가될 수 없는 함수가 들어 있습니까? 관계형 연산자는 함수로 분류됩니다.
- 술어 바인드 요구사항. 원격으로 평가할 경우, 이 술어가 어떤 값과 바인드해야 합니까? 그렇다면, 이 데이터 소스에서의 SQL 제한사항을 위반합니까?
- 전역 최적화. 최적화 알고리즘이 지역 처리가 비용면에서 좀더 효율적이라고 결론지을 수 있습니다. 자세한 내용은 235 페이지의 『원격 SQL 생성 및 전역 최적화』를 참조하십시오.

• GROUP BY 연산자를 지역적으로 평가할 수 없는 이유는 무엇입니까?

  다음과 같은 여러 분야를 점검해야 합니다.

  - GROUP BY 연산자에 대한 입력이 원격으로 평가됩니까? 평가되지 않으면 입력을 검토하십시오.
  - 데이터 소스에 이 연산자에 대한 제한사항이 있습니까? 예에서는 다음을 포함합니다.
    - 제한된 GROUP BY 항목 수
    - 결합된 GROUP BY 항목의 제한된 바이트 수
    - GROUP BY 목록에만 적용되는 컬럼 스펙
  - 데이터 소스가 이 SQL 연산자를 지원합니까?
  - 전역 최적화. 최적화 알고리즘이 지역 처리가 비용면에서 좀더 효율적이라고 결론지을 수 있습니다. 자세한 내용은 235 페이지의 『원격 SQL 생성 및 전역 최적화』를 참조하십시오.

• 집합 연산자를 원격으로 평가할 수 없는 이유는 무엇입니까?

  여러 분야를 점검해야 합니다.

  - 두 연산자 모두 동일한 원격 데이터 소스에서 완전히 평가됩니까? 완전히 평가가 되어야 하는데 그렇지 않은 경우, 각 피연산자를 검토하십시오.

- 데이터 소스에 이 집합 연산자에 대한 제한사항이 있습니까? 예를 들어, 대형 오브젝트(LOB)나 Long 필드가 이 특정 집합 연산자에 대해 유효한 입력입니까?

- ORDER BY 조작을 원격으로 평가할 수 없는 이유는 무엇입니까?

  다음을 고려해 보십시오.

  - ORDER BY 조작에 대한 입력이 원격으로 평가됩니까? 평가되지 않으면 입력을 검토하십시오.
  - ORDER BY절에 문자 표현식이 들어 있습니까? 그렇다면, 원격 데이터 소스가 DB2와 동일한 조합 순서를 갖고 있지 않습니까?
  - 데이터 소스에 이 연산자에 대한 제한사항이 있습니까? 예를 들어, ORDER BY 항목 수에 제한이 있습니까? 데이터 소스가 ORDER BY 목록에 대한 컬럼 스펙을 제한합니까?

데이터 소스 업그레이드 및 사용자 정의: DB2 SQL 컴파일러가 데이터 소스 SQL 지원에 대해 많은 정보를 갖고 있더라도, 데이터 소스는 업그레이드 및/또는 조정될 수 있기 때문에 데이터에는 항상 저정이 필요합니다. 이 경우에는 지역 카탈로그 정보를 변경하여 DB2를 향상시키십시오. 카탈로그를 갱신하려면 DB2 DDL 문을 사용하십시오(예: CREATE FUNCTION MAPPING 및 ALTER SERVER). 자세한 내용은 *SQL 참조서*를 참조하십시오.

원격 SQL 생성 및 전역 최적화

이 단계는 조회를 평가할 전역 최적 액세스 전략을 생성하는 데 도움이 됩니다. 연합 데이터베이스 조회의 경우, 액세스 전략에는 원래 조회를 원격 조회 단위 세트로 나눈 다음 결과를 결합하는 과정이 포함될 수 있습니다.

최적화 알고리즘은 권고된 분석 푸시다운 출력을 사용하여 각 조작을 DB2에서 지역적으로 평가할 것인지, 아니면 원격으로 데이터 소스에서 평가할 것인지를 결정합니다. 결정은 비용 모델(조작을 평가하는 데 드는 비용뿐 아니라, DB2와 데이터 소스간에 데이터 및 메시지를 전송하는 데 드는 비용도 포함)의 출력을 근거로 내려집니다.

목표는 최적화된 조회를 생성하는 것이지만, 여러 가지 인수가 전역 최적화의 출력에 영향을 미치며 조회 성능에도 영향을 미칩니다. 핵심 인수는 서버 특성 및 별칭 특성에서 다루어집니다.

서버 특성/전역 최적화에 영향을 미치는 옵션

전역 최적화에 영향을 미칠 수 있는 데이터 소스 서버 인수는 다음과 같습니다.

- CPU 속도에 대한 상대적 비율

  데이터 소스 CPU 속도가 DB2 CPU 속도에 비해 얼마나 빠른지 또는 느린지를 나타내려면 *cpu_ratio* 서버 옵션을 사용하십시오. 비율이 낮다는 것은 데이터 소스 워크스테이션 CPU가 DB2 워크스테이션 CPU보다 빠르다는 것을 나타냅니다. 비율이 낮으면 DB2 최적화 알고리즘은 CPU 집중 조작을 데이터 소스로 푸시다운하는 것으로 생각하면 됩니다. 이 비율에 대한 자세한 내용은 119 페이지의 『연합 데이터베이스 조회에 영향을 미치는 서버 옵션』을 참조하십시오.

- I/O 속도에 대한 상대적 비율

  데이터 소스 시스템 I/O 속도가 DB2 시스템 속도에 비해 얼마나 빠른지 또는 느린지를 나타내려면 *io_ratio* 서버 옵션을 사용하십시오. 비율이 낮다는 것은 데이터 소스 워크스테이션 I/O 속도가 DB2 워크스테이션 I/O 속도보다 빠르다는 것을 나타냅니다. 비율이 낮으면 DB2 최적화 알고리즘은 I/O 집중 조작을 데이터 소스로 푸시다운하는 것으로 생각하면 됩니다. 이 비율에 대한 자세한 내용은 119 페이지의 『연합 데이터베이스 조회에 영향을 미치는 서버 옵션』을 참조하십시오.

- DB2 및 데이터 소스간의 통신 비율

  네트워크의 용량을 나타내려면 *comm_rate* 서버 옵션을 사용하십시오. 낮은 비율(DB2와 데이터 소스간의 네트워크 통신 속도가 느림)은 DB2 최적화 알고리즘에 이 데이터 소스로 송수신되는 메시지의 수를 줄이도록 권장합니다. 비율을 0으로 설정하면 최적화 알고리즘은 최소한의 네트워크 통신량이 필요한 조회를 생성합니다. 이 비율에 대한 자세한 내용은 119 페이지의 『연합 데이터베이스 조회에 영향을 미치는 서버 옵션』을 참조하십시오.

- 데이터 소스 조합 순서

  데이터 소스 조합 순서가 지역 DB2 데이터베이스 조합 순서와 일치하는지를 나타내려면 *collating_sequence* 서버 옵션을 사용하십시오. 이 옵션을 'Y'로 설

정하지 않으면 최적화 알고리즘은 이 데이터 소스로부터 검색된 데이터를 순서화되지 않은 데이터로 간주합니다. 조합 순서 성능 관련 문제에 대한 자세한 내용은 229 페이지의 『조합 순서』를 참조하십시오.

- 원격 플랜 추가 정보

  데이터 소스에서 플랜 추가 정보가 지원되는지 여부를 나타내려면 *plan_hints* 서버 옵션을 사용하십시오. 플랜 추가 정보는 데이터 소스 최적화 알고리즘에 대한 추가 정보를 제공하는 명령문의 단편입니다. 특정 조회 유형의 경우, 이 정보는 조회 성능을 향상시킬 수 있습니다. 플랜 추가 정보는 데이터 소스 최적화 알고리즘이 색인 사용 여부, 사용할 색인 또는 사용할 테이블 조인 순서를 결정하는 데 도움이 됩니다.

  플랜 추가 정보가 사용 가능한 경우, 조회는 추가 정보를 포함하는 데이터 소스로 전송됩니다. 예를 들어, 플랜 추가 정보와 함께 Oracle 최적화 알고리즘으로 전송되는 명령문은 다음과 같을 수 있습니다.

  ```
  SELECT /*+ INDEX (table1, t1index)*/
     col1
     FROM table1
  ```

  플랜 추가 정보는 `/*+ INDEX (table1, t1index)*/` 문자열입니다.

- DB2 최적화 알고리즘 지식 기본 내의 정보

  DB2는 원시 데이터 소스에 대한 정보가 포함된 최적화 알고리즘 지식 기본을 갖고 있습니다. DB2 최적화 알고리즘은 특정 DBMS에서는 생성될 수 없는 원격 액세스 플랜은 생성하지 않습니다. 즉, DB2는 원격 데이터 소스에 있는 최적화 알고리즘이 이해하거나 수용하지 못하는 플랜은 생성하지 않습니다.

전역 최적화에 영향을 미치는 별칭 특성

다음 절에는 전역 최적화에 영향을 미칠 수 있는 별칭 특정 인수가 들어 있습니다.

색인 고려사항: DB2는 데이터 소스에 있는 색인 정보를 사용하여 조회를 최적화할 수 있습니다. 이러한 이유로, DB2에 사용 가능한 색인 정보는 최신 정보여야 합니다. 별칭에 대한 색인 정보는 처음에는 별칭 시간을 작성할 때 획득할 수 있습니다. 색인 정보는 별칭을 보는 중에는 수집되지 않습니다.

별칭에 대한 색인 스펙 작성: 별칭에 대한 색인 스펙을 작성할 수 있습니다. 색인 스펙은 DB2 최적화 알고리즘이 사용할 색인 정의(실제 색인이 아닌)를 카탈로그에 빌드합니다. 색인 스펙을 작성하려면 CREATE INDEX SPECIFICATION ONLY문을 사용하십시오. 별칭에 대한 색인 스펙을 작성하는 구문은 지역 테이블에서 색인을 작성하는 구문과 비슷합니다. 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

다음과 같은 경우에 색인 스펙을 작성하는 것을 고려해 보십시오.

- DB2는 별칭을 작성하는 동안에는 데이터 소스로부터 색인 정보를 검색할 수 없습니다.
- 별칭을 보는 데 사용할 색인이 필요합니다.
- DB2 최적화 알고리즘에 중첩된 루프 조인의 내부 테이블로서 특정 별칭을 사용하도록 권장합니다. 색인이 없을 경우, 사용자는 조인 컬럼에 색인을 작성할 수 있습니다.

별칭에 대해 CREATE INDEX문을 발행하기 전에 뷰를 위해 필요한 것이 무엇인가를 고려해 보십시오. 하나의 예로, 뷰가 색인을 가진 테이블에서의 단순한 SELECT인 경우 데이터 소스의 테이블에 있는 색인과 일치하는 별칭에 대한 색인을 작성하면(지역적으로) 조회 성능이 월등히 향상될 수 있습니다. 그러나 색인이 단순한 Select문이 아닌 뷰를 통해 지역적으로 작성될 경우(예를 들면, 두 테이블을 조인하여 작성된 뷰), 조회 성능은 저하됩니다. 예를 들어, 색인이 두 테이블을 조인한 뷰를 통해 작성되는 경우, 최적화 알고리즘은 해당 뷰를 중첩된 루프 조인 내의 내부 요소로 선택할 수 있습니다. 조인이 여러 번 평가되기 때문에 조회의 성능은 저하됩니다. 다른 방법은 데이터 소스 뷰에서 참조되는 각 테이블에 대한 별칭을 작성하는 것과 DB2에 두 별칭을 참조하는 지역 뷰를 작성하는 것입니다.

카탈로그 통계 고려사항: 카탈로그 통계에서는 연관된 컬럼 내의 전체적인 별칭 크기와 값의 범위에 대해 설명합니다. 통계는 최적화 알고리즘이 별칭을 포함하는 조회 처리에 필요한 최소 비용 경로를 계산할 때 사용됩니다. 별칭 통계는 테이블 통계와 동일한 카탈로그 뷰에 저장됩니다. 통계 유형 및 지역적으로 갱신하는 방법에 대한 자세한 내용은 125 페이지의 『제5장 시스템 카탈로그 통계』 및 152 페이지의 『테이블 및 별칭 통계 갱신 규칙』을 참조하십시오.

DB2는 데이터 소스에 보유된 통계 데이터를 검색할 수는 있으나, 데이터 소스에 있는 기존의 통계 데이터에 대한 변경사항을 자동으로 검출할 수 없습니다. 또한 DB2에는 오브젝트 정의 또는 데이터 소스에 있는 오브젝트에 대한 구조 변경사항(컬럼 추가) 처리에 필요한 메커니즘이 없습니다. 오브젝트에 대한 통계 데이터 또는 구조 데이터가 변경된 경우, 다음 두 가지를 선택할 수 있습니다.

- 데이터 소스에서 RUNSTATS를 수행하십시오. 그런 다음 현재 별칭을 삭제하십시오. 별명을 재작성하십시오. 구조 정보가 변경된 경우에는 이 접근 방법을 사용하십시오.
- SYSSTAT.TABLES 뷰의 통계를 수동으로 갱신하십시오. 이 접근 방법은 단계는 간단하나 구조 정보가 변경된 경우에는 작동하지 않습니다.

전역 최적화 결정의 분석 및 이해

이 절에서는 조회 최적화 분석에 필요한 도구를 소개하고 조회 최적화에 연관된 일반적인 질문(및 연구하도록 제안되는 분야)에 대해 설명합니다.

조회 최적화 분석: DB2에서는 전역 액세스 플랜을 보여주는 두 개의 유틸리티를 제공합니다.

- Visual Explain. **db2cc** 또는 **db2vexp** 명령으로 시작하십시오. 조회 액세스 플랜 그래프를 보려면 이 명령을 사용하십시오. 각 연산자의 실행 위치는 연산자에 대한 세부사항 화면에 포함되어 있습니다. RQUERY(select 조작) 연산자에서 각 데이터 소스에 대해 생성된 원격 SQL문도 찾을 수 있습니다. 각 연산자에 대한 세부사항을 검토하여 DB2 최적화 알고리즘이 각 연산자에 대한 입력으로 추정된 행 수를 알 수 있습니다. 통신 비용을 포함한 각 연산자를 실행하는 데 필요한 계산된 비용도 알 수 있습니다. 자세한 내용은 643 페이지의 『부록C. SQL Explain 도구』를 참조하십시오.
- SQL Explain. db2expln 또는 dynexpln 명령으로 시작하십시오. 액세스 플랜 전략을 텍스트로 보려면 이 명령을 사용하십시오. SQL Explain에서는 비용 정보를 제공하지 않습니다. 그러나 원격 Explain 함수에서 지원하는 데이터 소스에 대한 원격 최적화 알고리즘에 의해 생성된 액세스 플랜을 확보할 수 있습니다. 자세한 내용은 643 페이지의 『부록C. SQL Explain 도구』를 참조하십시오.

DB2 최적화 결정의 이해: 이 절에서는 최적화에 대한 질문과 성능 향상을 위해 연구할 핵심 분야를 나열합니다. 핵심 질문은 다음과 같습니다.

- 동일한 데이터 소스에 있는 두 별칭간의 조인이 원격으로 평가되지 못하는 이유는 무엇입니까?

  조사해야 할 분야는 다음과 같습니다.

  - 조인 조작. 데이터 소스가 이를 지원합니까?
  - Join 술어. Join 술어를 원격 데이터 소스에서 평가할 수 있습니까? 평가할 수 없으면 Join 술어를 검토하십시오. 자세한 내용은 233 페이지의 『조회가 데이터 소스 또는 DB2에서 평가되는 이유 이해』를 참조하십시오.
  - (Visual Explain을 사용한) 조인 결과 내의 행 수. 조인이 두 별칭을 결합한 것보다 훨씬 많은 행 세트를 생성합니까? 숫자가 의미가 있습니까? 아니라면, 별칭 통계를 수동으로 갱신하는 것을 고려 하십시오(SYSSTAT.TABLES).

- GROUP BY 연산자를 지역적으로 평가할 수 없는 이유는 무엇입니까?

  조사해야 할 분야는 다음과 같습니다.

  - 연산자 구문. 원격 데이터 소스에서 연산자를 평가할 수 있는지를 검증하십시오. 자세한 내용은 233 페이지의 『조회가 데이터 소스 또는 DB2에서 평가되는 이유 이해』를 참조하십시오.
  - 행 수. Visual Explain을 사용하여 GROUP BY 연산자 입력 및 출력에서 추정된 행 수를 점검하십시오. 이 두 수가 매우 근접합니까? 근접하면 DB2 최적화 알고리즘은 이 GROUP BY절을 지역적으로 평가하는 것이 더 효율적이라고 판단합니다. 또한 이 두 수가 의미가 있습니까? 의미가 없으면 수동으로 별칭 통계를 갱신하는 것을 고려하십시오(SYSSTAT.TABLES).

- 명령문이 원격 데이터 소스에서 완전히 평가되지 못하는 이유는 무엇입니까?

  DB2 최적화 알고리즘은 비용에 근거한 최적화를 수행합니다. 분석 푸시다운에 모든 연산자가 원격 데이터 소스에서 평가될 수 있는 것으로 나타나더라도, 최적화 알고리즘은 여전히 비용 측정치에 따라 전역 최적 플랜을 생성합니다. 해당 플랜에 기여할 수 있는 여러 가지 인수가 있습니다. 예를 들어, 원격 데이터 소스가 원래 조회 내의 모든 조작을 처리할 수는 있더라도, CPU 속도는 DB2의 속도보다 느리므로 DB2에서 조작을 수행하는 것이 훨씬 낫다고 판단될 수 있습니다. 결과가 만족스럽지 못하면 SYSCAT.
  SERVEROPTIONS의 서버 통계를 검증하십시오.

- 최적화 알고리즘에 의해 생성되고 원격 데이터 소스에서 완전히 평가된 플랜이 원격 데이터 소스에서 직접 실행되는 원래의 조회보다 성능이 떨어지는 이유는 무엇입니까?

  조사해야 할 분야는 다음과 같습니다.

  - DB2 최적화 알고리즘에 의해 생성된 원격 SQL문. 이것이 원래의 조회와 동일한지 확인하십시오. 술어 순서화 변경사항을 점검하십시오. 우수한 조회 최적화 알고리즘은 조회의 술어 순서화에 영향을 받아서는 안됩니다. 불행히도, 모든 DBMS 최적화 알고리즘이 동일하지는 않으므로, 원격 데이터 소스의 최적화 알고리즘이 입력 술어 순서화에 따라 다른 플랜을 생성할 수 있습니다. 다른 플랜이 생성될 경우, 이는 원격 최적화 알고리즘의 문제점입니다. DB2에 대한 입력에서 술어의 순서화를 수정하거나 원격 데이터 소스 서비스 센터에 지원을 요청하십시오.

    또한 술어 대체도 점검하십시오. 우수한 조회 최적화 알고리즘은 동등한 술어로 대체에 영향을 받아서는 안됩니다. 불행히도, 모든 DBMS 최적화 알고리즘이 동일하지는 않으므로, 원격 데이터 소스의 최적화 알고리즘이 입력 술어에 따라 다른 플랜을 생성할 수 있습니다. 예를 들어, 일부 최적화 알고리즘은 술어에 대한 전이 폐쇄 명령문을 생성할 수 없습니다.
  - 리턴된 행 수. Visual Explain으로부터 이 행 수를 확보할 수 있습니다. 조회가 대량의 행을 리턴하면 네트워크 통신량에 병목 현상이 발생할 수 있습니다.
  - 추가 함수. 원격 SQL문에 원래 조회에 비해 추가 함수가 들어 있습니까? 데이터 유형 변환을 위해 일부 추가 함수가 생성될 수 있습니다. 이들 함수가 반드시 필요한지 확인하십시오.

제7장 SQL Explain 기능

SQL Explain 기능은 정적 SQL문 또는 동적 SQL문이 컴파일된 환경에 대한 정보를 캡쳐하는 데 사용될 수 있는 SQL 컴파일러의 일부입니다. 캡쳐된 정보를 사용하면 다음을 비롯한 SQL문의 구조와 잠재적인 실행 성능을 이해할 수 있습니다.

- 조회 처리를 위한 조작 순서
- 비용 정보
- 술어 및 선택성 추정치
- Explain시 SQL문에서 참조된 모든 오브젝트에 대한 통계

이 정보는 다음을 수행하는 데 도움이 될 수 있습니다.

- 조회에 대해 선택된 실행 플랜의 이해
- 응용프로그램 설계 지원
- 응용프로그램이 리바인드되어야 할 시기 결정
- 데이터베이스 설계 지원

다음과 같은 주제가 제공됩니다.

Explain 출력은 관계형 테이블에 저장되는데, 옵션으로 Visual Explain 도구를 사용하여 그래픽으로 표시될 수 있는 형식으로도 저장될 수 있습니다. 관심있는

© Copyright IBM Corp. 1993, 2001

Explain 테이블에 대해 수행되는 조회를 찾으려면 Explain 테이블을 사용해야 합니다. Explain 기능에 의해 사용되는 테이블 및 이들 테이블 작성 방법에 대한 자세한 내용은 609 페이지의 『부록B. Explain 테이블 및 정의』를 참조하십시오.

Explain 도구 선택

DB2는 산업 분야에서 Explain된 SQL문에 선택된 액세스 플랜에 대한 상세한 최적화 알고리즘 정보와 함께 가장 포괄적인 Explain 기능을 제공합니다. Explain 정보를 캡쳐하고 액세스하는 데 필요한 융통성을 부여하는 여러 가지 방법이 제공됩니다.

액세스 플랜의 심도 깊은 분석을 가능하게 하는 상세 최적화 알고리즘 정보는 실제 액세스 플랜 자체와는 별도로 Explain 테이블에 보존됩니다. Explain 테이블에서 정보를 얻는 방법에는 세 가지가 있습니다.

1. 사용자의 조회 작성(609 페이지의 『부록B. Explain 테이블 및 정의』에 나와 있는 Explain 테이블 설명에 기초하여)
2. *db2exfmt* 도구 사용
3. Visual Explain 사용(Explain 스냅샷 정보를 보기 위해)

지원되는 모든 플랫폼상의 Explain 테이블에 액세스가 가능하며, 여기에는 정적 및 동적 SQL문 모두에 대한 정보가 들어 있습니다. 출력의 손쉬운 조작 및 다른 조회간의 비교 또는 같은 조회 오버타임의 비교를 위해 허용되는 SQL문을 사용하여 Explain 테이블에 액세스할 수 있습니다. Explain 테이블의 정보를 사전 정의된 형식으로 표시하려는 경우, *db2exfmt* 도구를 사용할 수 있습니다. 이 도구에 대한 자세한 내용은 695 페이지의 『부록D. db2exfmt - Explain 테이블 형식 도구』를 참조하십시오. 다른 방법으로, 테이블에 액세스하기 위해 사용자의 명령문을 작성할 수 있습니다.

주: 이 도구의 위치(및 *db2batch, dynexpln, db2vexp* 및 *db2_all*)는 *sqllib* 디렉토리의 *misc* 서브디렉토리에 있습니다. 이 도구가 이 경로로부터 이동되면 위에서 설명한 명령행 항목은 작업하지 않습니다.

Visual Explain에서는 그래픽 인터페이스를 통해 Explain 테이블의 액세스 플랜 및 최적화 알고리즘 정보 분석이 가능합니다. 이 도구를 사용하여 정적 및 동적

SQL문을 모두 분석할 수 있습니다. Visual Explain은 일반적으로 제어 센터 내에서 호출됩니다. 제어 센터는 명령행에서 *db2cc*를 입력하여 사용할 수 있습니다. 또한 Visual Explain은 하나의 SQL문인 경우에는 *db2vexp* 명령을 사용하여 명령행에서 직접 호출도될 수 있습니다. 일부 플랫폼에서 Visual Explain은 DB2 Universal Database 폴더 내에서 폴더를 사용하여 호출될 수도 있습니다. Visual Explain은 지원되는 모든 플랫폼에서 사용 가능한 것은 아닙니다. Visual Explain이 지원되는지를 알아보려면 *빠른 시작* 매뉴얼을 참조하십시오. Visual Explain에서는 다른 플랫폼에서 캡쳐되거나 취해진 스냅샷을 볼 수 없습니다. 예를 들어, Windows NT 클라이언트는 HP-UX용 DB2 서버에서 생성된 스냅샷을 그래프로 나타낼 수 있습니다. 이렇게 하려면 두 플랫폼 모두 버전 5 이상이어야 합니다. Visual Explain에서 생성된 출력은 심층 분석을 위해 쉽게 조작하거나 다른 응용프로그램에서 액세스할 수 있는 정보가 아닙니다. *db2vexp* 명령에 대해서는 명령행에 *db2vexp -h*를 입력하거나 *Command Reference* 매뉴얼을 참조하십시오. Visual Explain에 대한 일반적인 내용은 *db2cc*를 입력하여 제어 센터 내의 온라인 도움말을 참조하십시오.

정적 SQL문의 액세스 플랜에 관한 정보는 패키지의 일부로서 생성되어 시스템 카탈로그에 저장됩니다. 하나 이상의 패키지에 사용 가능한 액세스 플랜 정보를 보기 위해 명령행에서 *db2expln* 도구를 사용할 수 있습니다. *db2expln*은 선택된 액세스 플랜에 대한 실제 구현을 보여줍니다. 그러나 최적화 알고리즘에 대한 정보는 보여주지 않습니다.

dynexpln 도구는 도구 내에서 *db2expln*을 사용하며, 매개변수 표시문자를 포함하지 않는 동적 SQL문을 설명하는 신속한 방법을 제공합니다. *dynexpln* 내에서 *db2expln*을 사용하는 것은 의사 패키지 내에서 입력 SQL문을 정적 명령문으로 변환하여 수행됩니다. 이것이 실행되면 정보가 항상 완벽하게 정확하지는 않습니다. 완전한 정확성을 원하는 경우, 246 페이지의 『SQL Explain 기능 사용』에 설명된 대로 Explain 기능을 사용해야 합니다.

db2expln 도구는 생성되는 실제 액세스 플랜을 조사하여 런타임 시 발생할 조작에 대한 상대적으로 간결하고 영어와 유사한 개요를 제공합니다(코드 생성 방법에 대한 자세한 내용은 171 페이지를 참조하십시오). *db2expln* 사용 및 출력 해석에 대한 자세한 내용은 643 페이지의 『부록C. SQL Explain 도구』를 참조하십시오.

표16에는 DB2 Explain 기능에 사용 가능한 여러 가지 도구 및 각각의 특성이 요약되어 있습니다. 이 표를 사용하여 사용자의 환경과 요구사항에 가장 적합한 도구를 선택하십시오.

표 16. Explain 기능 도구

원하는 특성	**Visual Explain**	**db2vexp**	**Explain 테이블**	**db2exfmt**	**db2expln**	**dynexpln**
GUI 인터페이스	예	예				
텍스트 출력				예	예	예
『quick and dirty』 정적 SQL 분석					예	
지원되는 정적 SQL	예		예	예	예	
지원되는 동적 SQL	예	예	예	예		예*
지원되는 CLI 응용프로그램	예		예	예		
DRDA 응용프로그램 리퀘스터에 사용 가능함			예			
상세 최적화 알고리즘 정보	예	예	예	예		
다중 명령문 분석에 적합함			예	예	예	예
응용프로그램 내에서 액세스 가능한 정보			예			

주:

***** 간접적으로 db2expln 사용. 몇 가지 제한사항이 있습니다.

SQL Explain 기능 사용

Explain 정보를 캡쳐하는 방법은 다음과 같습니다.

1. EXPLAIN 및 EXPLSNAP BIND/PREP 옵션
2. CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터
3. EXPLAIN SQL문
4. *db2vexp* 도구(직접 Visual Explain을 호출하여 정보를 표시할 수 있음)

Explain 데이터를 수집하고 사용하려는 이유로는 세 가지가 있습니다.

1. 데이터베이스 관리 프로그램이 조회를 충족시키기 위해 수행해야 하는 단계(액세스 플랜)를 이해하기 위해서입니다. 183 페이지의 『데이터 액세스 개념 및 최적화』에서는 Explain 출력을 이해하려고 할 경우 참조해야 하는 정보를 제공합니다.
2. 성능 조정 여부를 평가하는 데 도움이 됩니다. 조회의 성능 향상을 위해 취할 수 있는 조치에는 여러 가지가 있습니다. 이러한 가능한 조치 중 대부분이 다음의 하위 주제에 설명되어 있습니다.
   - 45 페이지의 『제3장 응용프로그램 고려사항』
   - 101 페이지의 『제4장 환경상의 고려사항』
   - 125 페이지의 『제5장 시스템 카탈로그 통계』

   영역 중 어느 부분을 변경한 후에는 선택된 액세스 플랜에 대해 변경사항이 주는 영향을 결정하는 데 SQL Explain 기능을 사용할 수 있습니다. 예를 들어, 109 페이지의 『색인화가 조회 최적화에 미치는 영향』에서 제공된 권장사항에 기초하여 색인을 추가하는 경우, Explain 데이터는 색인이 생각대로 사용되는지 결정하는 데 도움을 줄 수 있습니다.

   Explain 출력이 선택된 액세스 플랜과 상대적 비용을 결정하도록 해주는 정보를 제공하는 반면에 조회에 대한 성능 향상을 정확하게 측정하는 유일한 방법은 363 페이지의 『제12장 벤치마크 테스팅』에 설명된 대로 벤치마크 테스트 기술을 사용하는 것입니다.
3. 조회 성능의 변경 원인을 이해하려면 영향을 분석할 수 있도록 변경 전후 모두에 Explain 정보를 가지고 있어야 합니다. 그러므로 데이터베이스로 SQL문을 컴파일할 때 다음을 수행해야 합니다.
   - Explain 기능을 사용하여 변경 전에 플랜 정보를 캡쳐하고, 그 결과 Explain 테이블을 저장합니다. 또는 `db2exfmt` Explain 도구로부터의 출력을 저장합니다.
   - Visual Explain에 액세스하기를 원하지 않거나 액세스할 수 없어 이 정보를 볼 수 없는 경우, 현재 카탈로그 통계를 저장 및/또는 인쇄합니다. (159 페이지의 『생산 데이터베이스 모델링』에 설명되어 있는 `db2look` 도구를 사용하여 이 타스크 수행을 도울 수 있습니다.)

- CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE TABLESPACE에 대한 명령문을 포함하여 DDL(Data Definition Language) 문을 저장 및/또는 인쇄합니다.

위에서 언급한 정보는 차후 분석을 위한 참조점으로 사용할 수 있는 *이전* 그림을 제공합니다. 또한 동적 SQL문의 경우, 응용프로그램을 처음으로 수행할 때 이 정보를 수집할 수 있습니다. 정적 SQL문의 경우도 바인드시 이 정보를 수집할 수 있습니다.

성능 변경에 대한 이유를 분석하려면 *이전* 데이터를 분석 시작시 조회와 환경에 대해 수집한 정보(*이후* 데이터)와 비교하십시오.

간단한 예로서, 색인이 액세스 경로의 부분으로서 더 이상 사용되지 않고 있다는 것을 분석으로 나타낼 수 있습니다. Visual Explain의 카탈로그 통계 정보를 사용하면 현재 색인 레벨(NLEVELS 컬럼) 수가 조회가 처음으로 데이터베이스에 바인드될 때보다 실질적으로 높음을 알 수 있습니다. 다음을 선택할 수도 있습니다.

- 색인 재구성
- 테이블과 색인에 대한 새 통계 수집
- 조회 리바인드시 Explain 정보 수집

이 조치를 따름으로써 색인이 다시 액세스 플랜에서 사용되고 있으며, 조회의 성능이 더 이상 문제가 되지 않음을 볼 수 있습니다.

Explain에 대한 개념 소개

Explain 정보를 사용하여 183 페이지의 『데이터 액세스 개념 및 최적화』에 설명된 선택사항에 따라 최적화 알고리즘이 선택되는 액세스 플랜을 분석할 수 있습니다. 예를 들어, Explain 정보는 최적화 알고리즘에 의해 색인 스캔(184 페이지의 『색인 스캔 개념』 참조)이 선택되었음을 나타낼 수 있습니다. 이 외에도, 다음을 결정할 수도 있습니다.

- 195 페이지의 『범위 구분 술어 및 색인 SARGable 술어』에 설명된 대로 검색 기준으로 사용된 색인의 컬럼 수
- 189 페이지의 『색인 전용 액세스』에 설명된 대로 색인 전용 액세스의 사용 여부

- 294 페이지의『목록 프리페치의 이해』에 설명된 대로 페이지를 읽는 데 목록 프리패치가 사용되는지 여부

또 다른 예로서, Explain 정보는 두 테이블이 조인되는 방법을 이해하는 데 도움을 줄 수 있습니다.

- 조인 방법
- 테이블이 조인되는 순서
- 정렬의 발생 및 유형

SELECT, SELECT INTO, UPDATE, INSERT, VALUES, VALUES INTO 및 DELETE SQL문에 Explain을 사용할 수 있지만, Explain의 기본 용도는 명령문의 SELECT 부분에 대한 액세스 경로를 관찰하는 것입니다.

SQL 조회를 충족시키기 위해 데이터베이스 관리 프로그램은 일반적으로 다음과 같은 작업을 수행합니다.

- 하나 이상의 데이터 오브젝트(테이블, 색인 또는 둘다) 사용
- 하나 이상의 조작(예: 테이블 스캔, 색인 스캔, 조인) 수행
- 호출 응용프로그램으로 결과 세트 리턴

다음과 같은 단순한 SQL 조회의 경우,

```
SELECT DEPTNO, DEPTNAME
         FROM DEPARTMENT
```

수행된 단계의 그래픽 표현은 Visual Explain에 의해 표시될 수 있습니다.

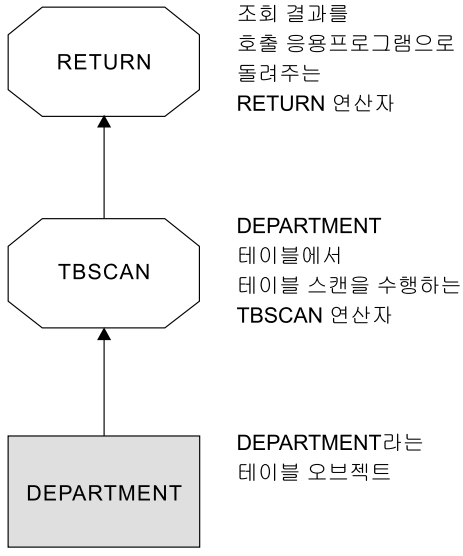


그림 20. Explain 출력의 그래픽 화면

다음 주제에서는 오브젝트와 연산자에 대해 볼 수 있는 세부사항의 유형에 대해 설명합니다.

- 『데이터 오브젝트에 대한 Explain 정보』
- 251 페이지의 『데이터 연산자에 대한 Explain 정보』

데이터 오브젝트에 대한 Explain 정보

단일 액세스 플랜은 하나 이상의 데이터 오브젝트를 사용하여 SQL문을 충족시킬 수 있습니다.

오브젝트 통계: Explain 기능은 다음과 같은 오브젝트에 대한 사실을 기록합니다.

- 작성 시간
- 오브젝트에 대한 통계가 수집된 마지막 시점(125 페이지의 『제5장 시스템 카탈로그 통계』 참조)
- 오브젝트에서의 데이터가 순서화되는지에 대한 표시(테이블 또는 색인 오브젝트만)
- 오브젝트의 컬럼 수(테이블 또는 색인 오브젝트만)
- 오브젝트의 계산된 행 수(테이블 또는 색인 오브젝트만)

- 버퍼 풀에서 오브젝트가 차지하는 페이지 수
- 이 오브젝트가 저장된 지정 테이블 공간으로의 단일 무작위(random) I/O에 대한 총 추정 오버헤드(밀리초 단위)
- 지정된 테이블 공간에서 4K 페이지를 읽기 위한 추정 전송률(밀리초 단위)
- 4K 페이지의 프리페치 및 Extent *크기*
- 해당 색인을 갖는 데이터 클러스터링의 등급
- 이 오브젝트 색인이 사용하는 리프 페이지의 수 및 트리의 레벨 수
- 이 색인 오브젝트의 구별 전체 키 값의 수
- 테이블의 오버플로우 레코드의 총 수

데이터 연산자에 대한 Explain 정보

단일 액세스 플랜이 데이터에 여러 조작을 수행하여 SQL문을 충족시키고 결과를 다시 제공할 수 있습니다. SQL 컴파일러는 필요한 조작(예: 테이블 스캔, 색인 스캔, 중첩된 루프 조인 또는 group-by 연산자)을 결정합니다. 이 연산자에 대한 자세한 내용은 183 페이지의 『데이터 액세스 개념 및 최적화』에 설명되어 있습니다.

Explain 정보는 액세스 플랜에 사용된 여러 연산자를 표시하며, 액세스 플랜의 누적 효과뿐 아니라, 각 연산자에 대해서도 Explain 정보를 사용할 수 있습니다.

계산된 비용 정보: 연산자에 대해 다음과 같은 계산된 누적 비용을 표시할 수 있습니다. 이 비용은 선택된 액세스 플랜에 대한 것이며, 정보가 캡쳐된 연산자까지 포함한 것입니다.

- 총 비용(timeron 단위)
- 페이지 I/O 수
- CPU 명령의 수
- 요구되는 최기의 모든 오버헤드를 포함하여 첫 번째 행을 페치하는 비용(timeron 단위)
- 통신 비용(프레임 단위)

*timeron*이란 인위적인 상대적 측정 단위입니다. timeron은 데이터베이스를 사용함에 따라 변경되는 통계와 같은 내부 값에 기반을 둔 최적화 알고리즘으로 결정됩

니다. 그 결과, timeron에서 SQL문이 매번 계산된 비용이 같도록 하는데 대해 timerons 수치가 결정되는 것을 보장할 수 없습니다.

연산자 등록 정보: 다음 정보는 각 연산자의 등록 정보을 설명하기 위해 Explain 기능에 의해 기록됩니다.

- 액세스된 테이블 세트
- 액세스된 컬럼 세트
- 데이터가 순서화된 컬럼 - 최적화 알고리즘이 이 순서화가 후속의 연산자에 의해 사용될 수 있다고 결정할 경우
- 적용된 술어 세트
- 리턴될 행의 추정된 수(기본 행 수)

Explain 정보 구성 방법

모든 Explain 정보는 Explain 인스턴스의 개념 위주로 구성됩니다. Explain 인스턴스는 하나 이상의 SQL문에 대한 Explain 기능의 한 가지 호출을 나타냅니다. Explain 인스턴스는 다음에 대한 Explain 정보를 나타냅니다.

- **정적** SQL문에 대한 한 패키지 내의 모든 적당한 SQL문
- 증분식 바인드 SQL문에 대한 하나의 특정 SQL문
- **동적** SQL문에 대한 하나의 특정 SQL문
- 각각의 EXPLAIN SQL문(동적인지 또는 정적인지 여부)

하나의 Explain 인스턴스 내에서 캡쳐된 Explain 정보에는 컴파일 중인 SQL문을 충족시키기 위해 선택된 액세스 플랜뿐만 아니라, SQL 컴파일 환경도 포함됩니다. Explain 정보는 세 가지 부속 집합으로 구성됩니다.

Explain 인스턴스 정보	각 Explain 인스턴스에 대해 캡쳐된 컴파일 환경 정보
Explain 스냅샷 정보	Visual Explain에서 사용하는 정보
Explain 테이블 정보	Explain 테이블 정보 요청시 수집된 정보

Explain 인스턴스 정보

Explain 인스턴스 정보는 EXPLAIN_INSTANCE 테이블에 저장됩니다. Explain 인스턴스 내에서 각각의 Explain된 SQL문에 관한 추가 특정 정보는 EXPLAIN_STATEMENT 테이블에 저장됩니다.

Explain 인스턴스 식별: 각 Explain 인스턴스를 고유하게 식별하고 주어진 기능 호출에 대한 SQL문 정보를 다음 정보와 상관시킬 수 있습니다.

- Explain 정보를 요청한 사용자
- Explain 요청이 시작된 시기
- Explain된 SQL문이 유래된 패키지 이름
- Explain된 SQL문이 유래된 패키지의 스키마
- 스냅샷이 Explain 요청의 일부인지 여부에 대한 표시

환경 설정: SQL 컴파일러가 조회를 최적화한 방법에 대한 환경 정보가 캡쳐됩니다. 환경 정보에는 다음이 포함됩니다.

- 사용 중인 DB2의 버전 및 릴리스 번호
- 조회를 컴파일하는 데 사용된 병렬 처리 수준. CURRENT DEGREE 특수 레지스터, DEGREE 바인드 옵션, SET RUNTIME DEGREE API 및 *dft_degree* 구성 매개변수는 특정 조회를 컴파일할 때 사용되는 병렬 처리 수준을 결정할 때 사용될 수 있습니다.
- SQL문이 동적인지 정적인지의 여부
- 조회를 컴파일하는 데 사용되는 조회 최적화 클래스. 자세한 내용은 74 페이지의 『최적화 클래스 조정』을 참조하십시오.
- 조회를 컴파일할 때 지정되는 커서 블로킹의 유형. 커서에 대한 자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오. 커서 블로킹에 대한 자세한 내용은 88 페이지의 『행 블로킹』을 참조하십시오.
- 조회를 컴파일할 때 사용되는 분리 레벨. 자세한 내용은 45 페이지의 『동시성』을 참조하십시오.

- 조회가 컴파일될 때 다양한 구성 매개변수의 값. Explain 스냅샷이 취해질 때 기록되는 다음 매개변수를 비롯하여 조회 최적화에 영향을 미치는 구성 매개변수에 대한 자세한 내용은 101 페이지의 『조회 최적화에 영향을 주는 구성 매개변수』를 참조하십시오.
  - 396 페이지의 『버퍼 풀 크기(buffpage)』
  - 414 페이지의 『정렬 힙 크기(sortheap)』
  - 458 페이지의 『사용 중인 응용프로그램의 평균 수(avg_appls)』
  - 400 페이지의 『데이터베이스 힙(dbheap)』
  - 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
  - 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』
  - 552 페이지의 『CPU 속도(cpuspeed)』
  - 551 페이지의 『통신 대역폭(comm_bandwidth)』

SQL문 식별: 각 Explain 인스턴스에 대해 여러 SQL문이 설명되었습니다. Explain 인스턴스를 유일하게 식별하는 정보와 같이 다음 정보는 각각의 SQL문을 식별하는 데 도움을 줍니다.

- 명령문 유형: SELECT, DELETE, INSERT, UPDATE, 위치지정 DELETE, 위치지정 UPDATE
- SYSCAT.STATEMENTS 카탈로그 뷰에 기록된 것과 같이 SQL문을 발행하는 명령문과 패키지의 섹션 번호

EXPLAIN_STATEMENT 테이블 내에서 QUERYTAG 및 QUERYNO 필드는 식별자를 포함하며 Explain 프로세스의 일부로서 설정되어 있습니다.

CLP 또는 CLI 세션 중에 제출된 동적 SQL문 Explain의 경우, EXPLAIN MODE 또는 EXPLAIN SNAPSHOT이 활성화될 때 QUERYTAG가 『CLP』 또는 『CLI』로 설정됩니다. 이와 같은 상황이 발생하면 QUERYNO의 기본값으로 각 명령문에 대해 1 이상이 증가된 숫자가 설정됩니다.

다른 모든 동적 SQL문 Explain의 경우(CLP, CLI에서 나오지 않거나 SQL문 EXPLAIN을 사용하는), QUERYTAG는 공백으로 설정되고 QUERYNO는 항상 『1』이 됩니다.

비용 계산: 설명된 각 명령문에 대해 선택된 액세스 플랜을 실행하는 상대적 비용의 추정치가 기록됩니다. 이 비용은 *timeron*이라고 하는 상대적인 측정 단위를 사용하여 제공됩니다. 다음 이유로 경과 시간에 대한 추정치가 제공되지 **않습니다**.

- SQL 최적화 알고리즘은 경과 시간을 추정하지 않고 자원 소비를 추정합니다.
- 최적화 알고리즘은 경과 시간에 영향을 줄 수 있는 모든 인수를 모델링하지 않고, 액세스 플랜의 효율성에 영향을 주지 않는 인수를 무시합니다. 경과 시간은 다음을 포함한 런타임 인수의 수에 의해 영향을 **받습니다**. 시스템 워크로드, 자원 경합의 양, 병렬 처리 및 I/O의 양, 행을 사용자에게 리턴하는 데 드는 비용 그리고 클라이언트와 서버간의 통신 시간 등입니다.

명령문 텍스트: 각각의 Explain된 각 명령문에 대해 두 버전의 SQL문 텍스트가 기록됩니다. 그 중 한 가지는 SQL 컴파일러에 의해 수신되는 텍스트이고, 다른 한 가지는 조회의 내부 컴파일러 표현으로부터 역변환되는 명령문 텍스트의 버전입니다. 이 변환은 다른 SQL문과 유사해 보이지만, 올바른 SQL 구문을 따르거나 전체적으로 내부 표현의 실제 내용을 반영할 필요는 **없습니다**. 이 변환은 단지 SQL 최적화 알고리즘이 액세스 플랜을 선택하는 SQL 문맥을 이해할 수 있게 하기 위해 제공되는 것일 뿐입니다. 사용자가 작성한 명령문 텍스트를 SQL문의 내부 표현과 비교하면 SQL 컴파일러가 더욱 최적화하기 위해 조회를 어떻게 재작성하는지를 이해하는 데 도움이 됩니다(171 페이지의 『SQL 컴파일러에 의한 조회 재작성』 참조). 또한 환경의 기타 요소가 트리거나 제한조건 같은 명령문에 영향을 주는지도 보여줍니다. 이 『최적화된』 텍스트에 의해 사용되는 일부 키워드는 다음과 같습니다.

$Cn	파생된 컬럼의 이름. 여기서 n은 정수 값을 나타냅니다.
$CONSTRAINT$	컴파일 중에 원래 SQL문에 추가된 제한조건의 이름을 나타내는 데 사용되는 태그. $WITH_CONTEXT$ 접두부와 함께 표시됩니다.
$DERIVED.Tn	파생된 테이블의 이름. 여기서 n은 정수 값을 나타냅니다.
$INTERNAL_FUNC$	범용으로는 사용되지는 않으나, Explain된 조회를 위해 SQL 컴파일러에 의해 사용되는 함수의 존재를 나타내는 데 사용되는 태그

$INTERNAL_PRED$	Explain된 조회의 컴파일 중에 SQL 컴파일러에 의해 추가된 술어의 존재를 나타내는 데 사용되는 태그. 이러한 술어를 다시 범용으로 사용할 수 없습니다. 내부 술어는 트리거와 제한조건의 결과로서 원래 SQL문에 추가된 추가 문맥을 충족시키기 위해 컴파일러에 의해 사용됩니다.
RID	특정 행에 대한 행 ID(RID) 컬럼을 식별하는 데 사용되는 태그.
$TRIGGER$	컴파일 중에 원래 SQL문에 추가된 트리거의 이름을 나타내는 데 사용되는 태그. $WITH_CONTEXT$ 접두부와 함께 표시됩니다.
$WITH_CONTEXT$(...)	이 접두부는 추가 트리거나 제한조건이 원래 SQL문에 추가될 때 텍스트의 시작 부분에 표시됩니다. 이 접두부 다음에는 SQL문의 컴파일과 해석에 영향을 주는 모든 트리거나 제한조건의 이름 목록이 표시됩니다.

Explain 스냅샷 정보

Explain 스냅샷이 요청되면 SQL 최적화 알고리즘에 의해 선택된 액세스 플랜을 설명하는 추가 Explain 정보가 기록됩니다. 이 정보는 EXPLAIN_STATEMENT 테이블의 SNAPSHOT 컬럼에 Visual Explain이 요구하는 형식으로 저장됩니다. 다른 응용프로그램이 이 형식을 사용할 수 없습니다.

Explain 스냅샷의 내용에 관한 추가 정보는 Visual Explain 자체와 다음에서 얻을 수 있습니다.

- 250 페이지의『데이터 오브젝트에 대한 Explain 정보』
- 251 페이지의『데이터 연산자에 대한 Explain 정보』

Explain 테이블 정보

Explain 테이블 정보가 요청되면 SQL 최적화 알고리즘에 의해 선택된 액세스 플랜을 설명하는 추가 Explain 정보가 기록됩니다. 이 정보는 다음 Explain 테이블에 저장됩니다.

- EXPLAIN_ARGUMENT. 이 테이블은 각각의 개별적인 연산자에 대한 고유 특성(만일 있는 경우)을 나타냅니다.
- EXPLAIN_INSTANCE. 이 테이블은 모든 Explain 정보에 대한 주 제어 테이블입니다. Explain 테이블 내에 있는 데이터의 각 행은 명시적으로 이 테이블에 있는 하나의 고유 행에 링크됩니다. Explain 중인 SQL문의 소스에 대한 기본 정보 및 환경 정보가 이 테이블에 보존됩니다.
- EXPLAIN_OBJECT. 이 테이블은 SQL문을 충족시키기 위해 생성된 액세스 플랜에 필요한 데이터 오브젝트를 식별합니다.
- EXPLAIN_OPERATOR. 이 테이블은 SQL 컴파일러가 SQL문을 충족시키기 위해 필요한 모든 연산자를 포함합니다.
- EXPLAIN_PREDICATE. 이 테이블은 어떤 술어가 특정 연산자에 의해 적용되는지를 식별합니다.
- EXPLAIN_STATEMENT. 이 테이블은 다른 레벨의 Explain 정보를 위해 존재할 때 SQL문 텍스트를 포함합니다. 사용자가 입력한 원래 SQL문은 최적화 알고리즘에 의해 SQL문을 충족시키기 위해 액세스 플랜을 선택하는 데 사용된 버전과 함께 이 테이블에 저장됩니다.
- EXPLAIN_STREAM. 이 테이블은 개별적인 연산자와 데이터 오브젝트간의 입력 및 출력 데이터 스트림을 나타냅니다. 데이터 오브젝트 자체가 EXPLAIN_OBJECT 테이블에 나타나 있습니다. 데이터 스트림에 포함되는 연산자는 EXPLAIN_OPERATOR 테이블에 나타나 있습니다.
- ADVISE_WORKLOAD. 이 테이블을 통해 데이터베이스에 대한 사용자의 워크로드를 설명할 수 있습니다. 워크로드의 각 행은 SQL문을 나타내며 연관된 빈도에 의해 설명됩니다. 이 테이블은 db2advis 도구 및 색인 마법사에 의해 작업 및 정보를 찾고 저장하는 데 사용됩니다.
- ADVISE_INDEX. 이 테이블은 권장 색인에 대한 정보를 저장합니다. 테이블은 SQL 컴파일러, db2advis 유틸리티, 색인 마법사 또는 사용자에 위해 채워집니다. 이 테이블은 두 가지 방법으로 사용됩니다.
  - 권장되는 색인을 얻기 위해
  - 제안되는 색인에 대한 입력을 기본으로 색인을 평가하기 위해

기본적으로는 위의 모든 테이블이 작성되지 않습니다. 이는 sqllib 서브디렉토리의 misc 서브디렉토리에 있는 EXPLAIN.DDL 스크립트를 수행하여 작성될 수 있습니다. Explain 및 Advise 테이블이 필요한 데이터베이스에 연결하십시오. 그런 다음 명령 db2 -tf EXPLAIN.DDL을 발행하면 이 테이블이 작성됩니다. 필요할 경우, 테이블은 색인 마법사에 의해 자동으로 작성될 수도 있습니다.

Visual Explain의 각 직사각형 *오브젝트* 노드는 EXPLAIN_OBJECT 테이블의 행에 해당합니다. Visual Explain의 각 팔각형『연산자』노드는 EXPLAIN_OPERATOR 테이블의 행에 해당합니다. 연산자와 연산자의 오브젝트 간 링크 각각은 EXPLAIN_STREAM 테이블의 한 행에 해당됩니다.

Explain 테이블 정보는 Explain 스냅샷에 대해 기록된 정보와 내용면에서 유사하지만, 이 정보는 표준 SQL문을 사용하여 액세스할 수 있는 일반 관계형 테이블에 저장됩니다.

Visual Explain 액세스 플랜 그래프처럼 Explain 테이블은 액세스 플랜 내에서 연산자와 데이터 오브젝트간의 관계를 반영하도록 설계되었습니다. 다음 그림에서는 이러한 테이블간의 관계를 보여줍니다.

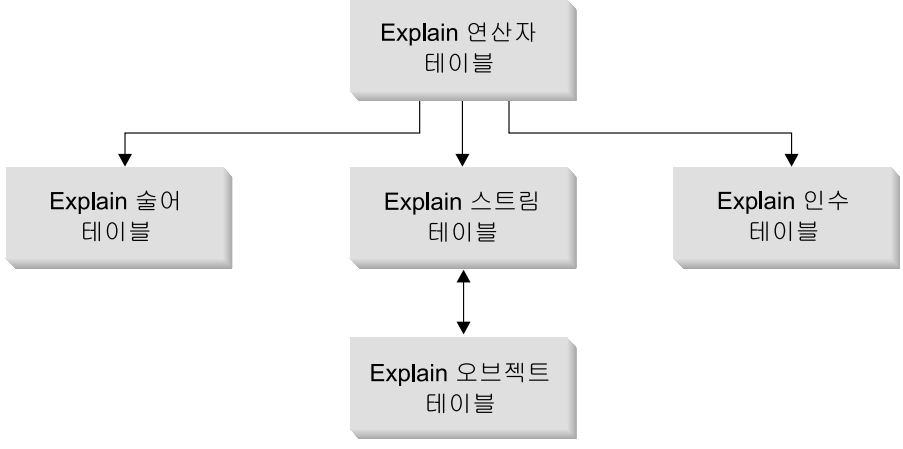


그림 21. Explain 테이블 관계(모든 테이블이 표시된 것은 아님) 개요

두 명 이상의 사용자가 공통된 Explain 테이블을 가질 수 있습니다. Explain 테이블은 한 명의 사용자에 대해 정의할 수 있습니다. 그리고 정의된 테이블을 가리키는 각각의 추가

사용자에 대해 동일한 이름을 사용하여 별명을 정의할 수 있습니다. 공통의 Explain 테이블을 공유하는 각 사용자는 이 테이블에 대한 삽입 사용권한을 가지고 있어야 합니다.

Explain 테이블 및 테이블 작성에 대한 자세한 내용은 643 페이지의 『부록C. SQL Explain 도구』를 참조하십시오. Explain 테이블 정보의 내용에 관한 추가 정보는 다음에서 얻을 수 있습니다.

- 250 페이지의 『데이터 오브젝트에 대한 Explain 정보』
- 251 페이지의 『데이터 연산자에 대한 Explain 정보』

`sqllib` 디렉토리하의 `misc` 서브디렉토리에 제공된 `db2exfmt` 도구는 Explain 테이블의 내용을 읽기 쉽게 구성된 출력으로 형식화하는 데 사용될 수 있습니다.

Explain 데이터의 획득

SQL문에 대한 Explian 데이터를 획득하기 전에 Explain 기능을 호출하는 권한 부여 ID와 동일한 스키마를 사용하여 정의된 한 세트의 Explain 테이블이 있어야 합니다. 테이블 작성 방법에 대한 자세한 내용은 632 페이지의 『Explain 테이블에 대한 테이블 정의』를 참조하십시오.

Explain 테이블 정보 캡쳐

일단 이들 테이블이 정의되면 SQL문이 컴파일되고 Explain 데이터가 요청될 때 Explain 데이터가 캡쳐됩니다.

- **정적** 또는 **증분식 바인드** SQL문의 경우, Explain 테이블 정보는 EXPLAIN ALL 또는 EXPLAIN YES 옵션이 BIND 또는 PREP 명령에 지정되거나 소스 프로그램에 정적 EXPLAIN SQL문이 사용될 경우에 캡쳐됩니다.

  주: 증분식 바인드 SQL문이 런타임 시 컴파일될 경우, 그 SQL문은 런타임 시 Explain 테이블에 배치되며 바인드시에는 배치되지 않습니다. 또한 Explain 테이블에 삽입하기 위해 사용되는 Explain 테이블 규정자와 권한 부여 ID는 패키지를 수행할 때의 사용자가 아닌 패키지 소유자의 것입니다.

- **동적** SQL의 경우, Explain 테이블 정보는 다음 상황에 대해 캡쳐됩니다.
  - EXPLAIN SQL문. FOR SNAPSHOT절이 사용되지 않으면 모든 Explain 정보가 캡쳐되어 Explain 테이블에 배치됩니다.

    다음은 EXPLAIN SQL문의 예입니다.

```
EXPLAIN PLAN FOR <any valid DELETE, INSERT, SELECT, SELECT INTO,
  UPDATE, VALUES, or VALUES INTO SQL statement>
```

- CURRENT EXPLAIN MODE 특수 레지스터가 YES로 설정됩니다. 이렇게 설정하면 SQL 컴파일러는 Explain 데이터를 캡쳐하며, SQL문이 실행되어 조회의 결과가 리턴됩니다.
- CURRENT EXPLAIN MODE 특수 레지스터가 EXPLAIN으로 설정됩니다. 이렇게 설정하면 SQL 컴파일러는 Explain 데이터를 캡쳐하지만, SQL 문을 실행하지는 않습니다.
- CURRENT EXPLAIN MODE 특수 레지스터는 RECOMMEND INDEXES로 설정됩니다. 이렇게 설정하면 SQL 컴파일러가 Explain 데이터 및 권장 색인을 캡쳐하여 ADVISE_INDEX 테이블에 배치합니다. 그러나 SQL문은 실행되지 않습니다.
- CURRENT EXPLAIN MODE 특수 레지스터는 EVALUATE INDEXES로 설정됩니다. 이렇게 설정하면 SQL 컴파일러는 사용자가 ADVISE_INDEX 테이블에 배치한 색인을 사용하게 됩니다. 사용자는 평가되어야 할 각 색인에 대한 새 행을 삽입합니다. 각 색인에 필요한 정보는 색인 이름, 테이블 이름, 평가되는 색인을 구성하는 컬럼 이름입니다. 일단 입력하면 특수 레지스터인 CURRENT EXPLAIN MODE가 EVALUATE INDEXES로 설정되어야 합니다. 그러면 SQL 컴파일러는 USE_INDEX 필드가 『Y』로 설정된 ADVISE_INDEX 테이블을 스캔합니다. 이를 가상 색인이라고 합니다. EVALUTE INDEXS 모드에서 실행되는 모든 동적문에는 이 가상 색인이 사용 가능한 것으로 설명되어 있습니다. 명령문의 성능이 향상될 경우, SQL 컴파일러는 가상 색인을 사용하도록 선택합니다. 그렇지 않으면 색인이 무시됩니다. EXPLAIN 결과를 검토하여 사용자가 제안한 색인이 SQL 컴파일러에 의해 사용되었음을 알 수 있습니다. 사용된 색인은 액세스 개선을 위해 구현될 것으로 간주되어야 합니다.
- EXPLSNAP ALL 옵션이 BIND나 PREP 명령에 지정되어 있습니다. 이렇게 설정하면 CURRENT EXPLAIN MODE 특수 레지스터의 설정이 NO이더라도, SQL 컴파일러가 런타임 시 동적 SQL의 Explain 데이터를 캡쳐합니다. 또한 SQL문도 실행되어 조회 결과를 리턴합니다.

주: Explain 정보는 SQL문이 컴파일될 때에만 캡쳐됩니다. 초기 컴파일을 따라 동적 SQL문은 환경에 대한 변경이 재컴파일될 명령문을 요구할 때에만

재컴파일됩니다. 동일한 PREPARE문이 동일한 SQL문에 대해 연속적으로 발행될 때에는 환경이 변경되지 않은 것으로 가정하고, 처음 PREPARE문이 발행되었을 때 SQL문은 컴파일되기만 하고 Explain 데이터는 캡쳐됩니다.

EXPLAIN SQL문 또는 CURRENT EXPLAIN MODE 레지스터 사용에 대한 자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오. BIND 및 PREP 명령에 대한 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

Explain 스냅샷 정보 캡쳐

Explain 스냅샷 데이터는 SQL문이 컴파일되고 Explain 데이터가 요청되면 캡쳐됩니다.

- **정적** 또는 **증분식 바인드** SQL문의 경우, Explain 스냅샷은 EXPLSNAP ALL 또는 EXPLSNAP YES절이 BIND 또는 PREP 명령에 지정되거나, 소스 프로그램에 FOR SNAPSHOT 또는 WITH SNAPSHOT절을 사용하는 정적 EXPLAIN SQL문이 사용될 경우에 캡쳐됩니다.

  주: 증분식 바인드 SQL문이 런타임 시 컴파일될 경우, 그 SQL문은 런타임 시 Explain 테이블에 배치되며 바인드시에는 배치되지 않습니다. 또한 Explain 테이블에 삽입하는 데 사용되는 Explain 테이블 규정자와 권한 부여 ID는 패키지를 수행할 때의 사용자가 아닌 패키지 소유자의 것입니다.

- **동적** SQL문의 경우, Explain 스냅샷은 다음 상황에 대해 캡쳐됩니다.
  - FOR SNAPSHOT 또는 WITH SNAPSHOT절을 사용하는 EXPLAIN SQL문. FOR SNAPSHOT절에는 Explain 스냅샷과 연관된 정보를 제외하면 캡쳐된 Explain 테이블 정보가 없습니다. WITH SNAPSHOT절에는 Explain 스냅샷과 연관된 정보를 비롯하여 캡쳐된 모든 Explain 테이블 정보가 있습니다.

    다음은 EXPLAIN SQL문을 사용하는 Explain 스냅샷의 예입니다.

    ```
    EXPLAIN PLAN FOR SNAPSHOT FOR <any valid DELETE, INSERT, SELECT,
       SELECT INTO, UPDATE, VALUES, or VALUES INTO SQL statement>
    ```

    Explain 스냅샷만 찍히고 캡쳐된 정보는 EXPLAIN_INSTANCE 및 EXPLAIN_STATEMENT 테이블에 배치됩니다.

- CURRENT EXPLAIN SNAPSHOT 특수 레지스터가 YES로 설정됩니다. 이렇게 설정하면 SQL 컴파일러가 Explain 데이터의 스냅샷을 찍으며 SQL 문이 실행되어 조회의 결과가 리턴됩니다.
- CURRENT EXPLAIN SNAPSHOT 특수 레지스터가 EXPLAIN으로 설정됩니다. 이렇게 설정하면 SQL 컴파일러는 Explain 데이터의 스냅샷을 찍지만 SQL문을 실행하지는 않습니다.
- EXPLSNAP ALL 옵션이 BIND나 PREP 명령에 지정되어 있습니다. 이렇게 설정하면 CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 설정이 NO이더라도, SQL 컴파일러는 런타임 시 Explain 데이터의 스냅샷을 찍습니다. 또한 SQL문도 실행되어 조회 결과가 리턴됩니다.

주: Explain 정보는 SQL문이 컴파일될 때에만 캡쳐됩니다. 초기 컴파일을 따라 동적 SQL문은 환경에 대한 변경이 재컴파일될 명령문을 요구할 때에만 재컴파일됩니다. 동일한 PREPARE문이 동일한 SQL문에 대해 연속적으로 발행될 때에는 환경이 변경되지 않은 것으로 가정하고, 처음 PREPARE문이 발행되었을 때 SQL문은 컴파일되기만 하고 Explain 데이터는 캡쳐됩니다.

EXPLAIN SQL문 및 FOR SNAPSHOT 또는 WITH SNAPSHOT절 사용과 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 사용에 대한 자세한 내용은 *SQL 참조서* 매뉴얼을 참조하십시오. BIND 및 PREP 명령에 대한 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

Explain 출력 사용에 대한 지침

Explain 데이터의 분석이 조회와 환경을 조정하는 데 도움을 줄 수 있는 방법에는 여러 가지가 있습니다. 예를 들면, 다음과 같습니다.

- **색인이 사용되고 있습니까?**

  109 페이지의 『색인화가 조회 최적화에 미치는 영향』에 설명된 대로 적합한 색인은 성능상 이점을 많이 얻을 수 있습니다. Explain 출력을 사용하여 특정 조회의 세트에 도움을 주도록 작성한 색인이 사용될 것인지를 결정할 수 있습니다. Explain 출력에서 다음 영역의 색인 사용법에 대해 알아보아야 합니다.

  - Join 술어

– 지역 술어
– GROUP BY절
– ORDER BY절
– 선택 목록

다른 색인이 기존 색인 대신에 사용될 수 있는지 또는 색인이 전혀 사용되지 않아도 되는지를 평가하는 데에도 Explain 기능을 사용할 수 있습니다. 새 색인을 작성한 후 해당 색인에 대한 통계를 수집하고(RUNSTATS 명령 사용) 조회를 재컴파일하십시오. Explain 데이터를 통해 색인 스캔 대신에 테이블 스캔이 사용되고 있는 것을 볼 수 있습니다. 테이블 데이터의 클러스터링 변경을 통해 이러한 결과가 생길 수 있습니다. 이전에 사용한 색인이 현재 낮은 클러스터 비율을 가지면 다음과 같이 수행할 수 있습니다.

– 테이블을 재구성하여 해당 색인에 따라 데이터 클러스터
– RUNSTATS 명령을 사용하여 테이블과 색인에 대한 카탈로그 통계 갱신
– 조회 재컴파일
– Explain 출력을 재조사하여 테이블의 재구성이 액세스 플랜을 향상시켰는지 판별

- **액세스 유형이 응용프로그램에 적합합니까?**

  Explain 출력을 분석하고 일반적으로 수행하고 있는 응용프로그램 유형에 대해 최적이 아닌 데이터에 대한 액세스의 유형을 찾을 수 있습니다. 예를 들면, 다음과 같습니다.

  – **온라인 트랜잭션 처리(OLTP) 조회**

    OLTP 응용프로그램은 색인 스캔을 범위 구분 술어와 함께 사용하는 핵심적인 분야인데, 이는 등호 술어를 키 컬럼에 대해 사용하여 규정된 몇몇의 행만을 리턴하는 경향이 있기 때문입니다. OLTP 조회가 테이블 스캔을 사용하는 경우, Explain 데이터를 분석하여 색인 스캔이 사용되지 않는 이유를 판별할 수 있습니다.

  – **브라우저 전용 조회**

    『찾아보기』 유형 조회에 대한 검색 기준은 매우 모호할 수 있습니다. 따라서 많은 행을 규정해야 할 필요가 있습니다. 사용자가 보통 출력 데이터의 몇 화면을 보기만 하는 경우, 어떤 결과가 리턴되기 전에 전체 대답 세트가

계산될 필요가 없도록 할 수 있습니다. 이 경우, 사용자의 목표는 데이터의 처음 몇 화면만이 아니라, 전체 조회에 대한 자원 소비를 최소화하려고 시도하는 최적화 알고리즘의 기본 작동 원칙과 다릅니다.

예를 들어, Explain 출력이 액세스 플랜에서 사용된 병합 스캔 조인과 정렬 연산자를 모두 보여주면, 전체 대답 세트는 어떤 행이 응용프로그램에 리턴되기 전에 임시 테이블에 구체화됩니다. 이 경우, SELECT문에 OPTIMIZE FOR절을 사용하여 액세스 플랜을 변경할 수 있습니다. (OPTIMIZE FOR 절에 대한 자세한 내용은 84 페이지의 『OPTIMIZE FOR n ROWS절』을 참조하십시오.) 최적화 알고리즘은 이러한 방식으로 응용프로그램에 첫 번째 행을 리턴하기 전에 임시 테이블에 있는 전체 대답 세트를 생성하지 않는 액세스 플랜을 선택할 수 있습니다.

- **어떤 유형의 조인 방법이 사용되고 있습니까?**

  조회시 두 개의 테이블이 조인될 경우, 사용 중인 조인 처리의 유형을 점검할 수 있습니다. 여러 행을 포함하는 조인(예: 결정 지원 조회)은 대개 병합 조인과 함께 더 빨리 수행됩니다. 몇 개의 행만을 포함하는 조인(예: OLTP 조회)의 경우에는 일반적으로 중첩된 루프 조인과 함께 더 빨리 수행됩니다. 그러나 이러한 일반 조인의 작업 방법을 변경하는 지역 술어 또는 색인의 사용과 같은 경우는 참작해야 할 사정이 있을 수 있습니다. (이 두가지 조인 방법이 작동하는 방법에 대한 자세한 내용은 198 페이지의 『중첩된 루프 조인』 및 200 페이지의 『병합 조인』을 참조하십시오.)

Visual Explain

Visual Explain은 다른 방법에 비해 특별히 좀더 복잡한 조작을 포함하는 조회에 대해 상세하게 학습하는 데 사용할 수 있습니다. Visual Explain은 지원되는 모든 플랫폼에서 사용 가능한 것은 아닙니다. Visual Explain이 지원되는지를 알아보려면 사용 중인 플랫폼에 대한 *빠른 시작* 매뉴얼을 참조하십시오.

Visual Explain을 사용하여 Explain된 SQL문에 대한 액세스 플랜을 그래프로 볼 수 있습니다. 그래프의 유용한 정보를 사용하여 보다 나은 성능이 발휘되도록 SQL 조회를 조정할 수 있습니다. Visual Explain을 사용하여 SQL문을 동적으로 설명하고 결과 액세스 플랜 그래프를 볼 수 있습니다.

최적화 알고리즘은 액세스 플랜을 선택하고, Visual Explain은 정보를 테이블 및 색인, 여기에서의 각 조작이 노드로서 표현되고 데이터의 흐름이 노드간의 링크로 표현되는 액세스 플랜 그래프로 표시합니다.

액세스 플랜 그래프를 표시하려면 Explain 스냅샷을 작성해야 합니다. 액세스 플랜 그래프에서 다음에 대한 세부사항을 볼 수 있습니다.

- 테이블 및 색인(및 연관된 컬럼)
- 연산자(예: 테이블 스캔, 정렬 및 조인)
- 테이블 공간 및 함수

Visual Explain을 사용하여 다음을 수행할 수 있습니다.

- 최적화시 사용되는 통계 뷰. 그런 다음 이러한 통계를 현재 카탈로그 통계와 비교하여 패키지 리바인딩이 성능을 향상시킬 수 있을지 여부를 판별하는 데 도움을 줄 수 있습니다.
- 테이블에 액세스하는 데 색인이 사용되는지 여부 결정. 색인이 사용되지 않을 경우, Visual Explain은 색인화하는 데 어떤 컬럼을 사용하는 것이 좋은지를 결정하는 데 도움이 될 수 있습니다.
- 하나의 조회에 대한 액세스 플랜 그래프의 이전 및 이후 버전을 비교하여 다양한 조정 기법 수행의 효과 보기.
- 해당 액세스 플랜에서 각 조작에 관한 정보(총 계산된 비용 및 수신된 행 수(기본 행 수) 포함) 획득.

Visual Explain에 대한 자세한 내용은 제어 센터를 통해 구할 수 있는 온라인 정보를 참조하십시오. 명령행에서 `db2cc`를 입력하여 제어 센터에 액세스할 수 있습니다.

SQL 조언 기능

색인 보조 프로그램은 사용자의 데이터에 적합한 색인을 설계 및 정의할 필요성을 줄여주는 관리 도구입니다.

색인 보조 프로그램은 다음과 같은 경우에 유용합니다.

- 문제점 조회에 대한 최상의 색인을 찾을 때

- 선택적으로 자원 한계에 적용될 경우, 조회(워크로드) 세트에 대한 최상의 색인을 찾을 때
- 색인을 작성하지 않고 워크로드에 대해 색인을 테스트할 때

SQL 조언 기능과 연관된 개념은 다음과 같습니다. 첫 번째는 *워크로드*입니다. 워크로드는 주어진 기간 동안에 데이터베이스 관리 프로그램이 처리해야 하는 SQL 문 세트입니다. SQL문에는 SELECT, INSERT, UPDATE 및 DELETE문이 포함될 수 있습니다. 예를 들어, 1개월 동안에 데이터베이스 관리 프로그램이 1 000건의 INSERT, 10 000건의 UPDATE, 10 000건의 SELECT 및 1 000건의 DELETE를 처리해야 할 수 있습니다. 워크로드 내의 정보는 주어진 기간 동안에 처리되는 SQL문의 유형 및 빈도와 관련이 있습니다. 조언 엔진은 이 워크로드 정보와 데이터베이스 정보를 함께 사용하여 색인을 권장합니다. 조언 엔진의 목적은 총 워크로드 비용을 최소화하는 것입니다.

두 번째는 *가상 색인* 개념입니다 가상 색인은 현재 데이터베이스 스키마에 존재하지 않는 색인입니다. 이 색인은 조언 기능이 사용자에게 권한 권장사항이거나 조언 기능에 사용자가 찾은 것을 평가해 주도록 요청한 색인입니다. 또한 이 색인은 조언 기능을 프로세스의 일부로 간주하여 권장할 만한 색인이 아니기 때문에 버리는 것일 수도 있습니다. 가상 색인은 ADVISE_INDEX 테이블을 사용하여 사용자로부터 조언 기능으로 전달되거나 되돌아올 수 있습니다.

조언 기능은 워크로드와 데이터베이스의 통계를 사용하여 권장 색인을 생성합니다.

조언 기능에서는 두 가지의 EXPLAIN 테이블을 사용합니다.

- ADVISE_WORKLOAD

  이 테이블은 고려되는 워크로드에 대해 설명하는 테이블입니다. 테이블의 각 행은 SQL문을 나타내며 연관된 빈도에 의해 설명됩니다. 여기에는 WORKLOAD_NAME이라는 테이블의 필드인 각 워크로드에 대한 식별자가 있습니다. 동일한 워크로드의 일부인 모든 SQL문은 동일한 WORKLOAD_NAME을 가져야 합니다.

  색인 마법사 및 `db2advis` 도구는 테이블을 사용하여 워크로드 정보를 찾고 저장합니다.
- ADVISE_INDEX

이 테이블은 권장 색인에 대한 정보를 저장합니다. 정보는 SQL 컴파일러, 색인 마법사, db2advis 도구 또는 사용자에 의해 이 테이블에 배치됩니다.

이 테이블은 두 가지 방법으로 사용됩니다.

- 조언 기능으로부터 권장 색인을 얻기 위해
- 색인을 평가하기 위해

주: 이 테이블을 작성하려면 sqllib 서브디렉토리의 misc 서브디렉토리에 있는 EXPLAIN.DDL 스크립트를 수행하십시오. 아직 작성되어 있지 않으면 색인 마법사로 테이블을 작성할 수도 있습니다.

색인 보조 프로그램 사용 프로세스에는 입력, 보조 프로그램의 호출, 출력 및 고려해야 하는 특수한 몇 가지 경우가 포함됩니다.

다음 세 가지 방법으로 색인 보조 프로그램에 대한 입력을 작성할 수 있습니다.

- 워크로드의 캡쳐

  평가될 SQL을 작성하는 다음 방법 중 하나를 사용하십시오.

  - 모니터를 사용하여 동적 SQL을 확보합니다.
  - SYSSTMT 카탈로그 뷰를 사용하여 정적 SQL을 확보합니다.
  - ADVISE_INDEX 테이블에 값을 잘라내기 및 붙여넣기를 수행하여 명령문 및 빈도를 추가합니다.
- 워크로드를 수정하여 조회의 중요도를 증가 또는 감소합니다.
- 데이터에 대한 제한조건(있을 경우)을 판별합니다.

다음 네 가지 방법으로 색인 보조 프로그램을 호출할 수 있습니다.

- 제어 센터의 사용

  바람직한 방법은 색인 보조 프로그램을 사용하는 것입니다. 제어 센터로부터 **색인** 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오. **색인** 폴더를 마우스 버튼 2로 누른 다음 팝업 메뉴에서 **작성->마법사를 사용한 색인**을 선택하십시오. 색인 마법사가 열립니다. 색인 마법사에는 사용하기 쉬운 확장 도움말이 있습니다. 이 마법사에는 최근 실행된 SQL을 찾거나, 최근 사용된 패키지를 통해 찾거나, 직접 SQL문을 추가하여 워크로드를 구성할 수 있는 기능도 들어 있습니다.

- 명령행 처리기 사용

  명령행에서 db2advis를 입력하십시오. db2advis는 다음 세 위치 중에서 워크로드를 읽어 시작합니다.

  – 명령행으로부터
  – 텍스트 파일의 명령문으로부터
  – 제안된 워크로드(SQL 및 빈도)에 행을 삽입한 후 ADVISE_WORKLOAD 테이블로부터

  도구는 CURRENT EXPLAIN MODE 레지스터를 사용하여 권장 색인을 확보한 다음 내부 최적화 알고리즘과 결합하여 최적의 색인을 찾아냅니다. 출력은 터미널 화면, ADVISE_INDEX 테이블 및 출력 파일로 출력됩니다(원할 경우).

  예를 들어, 도구에 간단한 조회인 『select count(*) from sales where region = 'Quebec'』에 대한 색인을 권장해줄 것을 요청할 수 있습니다.

```
$ db2advis -d sample \
 -s "select count(*) from sales where region = 'Quebec'" \
 -t 1
performing auto-bind
Bind is successful. Used bindfile: /home3/valentin/sqllib/bnd/db2advis.bnd
Calculating initial cost (without recommended indexes) [31.198040] timerons
Initial set of proposed indexes is ready.
Found maximum set of [1] recommended indexes
Cost of workload with all indexes included [2.177133] timerons
cost without index [0] is [31.198040] timerons.  Derived benefit is
[29.020907]
total disk space needed for initial set [1] MB
total disk space constrained to         [-1] MB
  1  indexes in current solution
 [31.198040] timerons  (without indexes)
 [2.177133] timerons  (with current solution)
 [%93.02] improvement
Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ689 ON VALENTIN.SALES (REGION DESC)
===========================
Index Advisor tool is finished.
```

  db2advis 도구는 워크로드에 대한 색인을 권장하는 데도 사용될 수 있습니다. 『sample.sql』이라는 입력 파일을 작성할 수 있습니다.

```
--#SET FREQUENCY 100
select count(*) from sales where region = ?;
--#SET FREQUENCY 3
select projno, sum(comm) tot_comm  from employee, emp_act
```

```
where employee.empno = emp_act.empno and
     employee.job='DESIGNER'
group by projno
order by tot_comm desc;
--#SET FREQUENCY 50
select * from sales where sales_date = ?;
```

그런 다음 다음 명령을 실행하십시오.

```
$ db2advis -d sample -i sample.sql -t 0
  found [3] SQL statements from the input file
Calculating initial cost (without recommmended indexes) [62.331280] timerons
Initial set of proposed indexes is ready.
Found maximum set of [2] recommended indexes
Cost of workload with all indexes included [29.795755] timerons
cost without index [0] is [58.816662] timerons.  Derived benefit is
[29.020907]
cost without index [1] is [33.310373] timerons.  Derived benefit is
[3.514618]
total disk space needed for initial set [2] MB
total disk space constrained to         [-1] MB
  2  indexes in current solution
 [62.331280] timerons  (without indexes)
 [29.795755] timerons  (with current solution)
 [%52.20] improvement
Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ119 ON VALENTIN.SALES (SALES_DATE DESC,
SALES_PERSON DESC)
index[2], 1MB CREATE INDEX WIZ63 ON VALENTIN.SALES (REGION DESC)
===========================
Index Advisor tool is finished.
```

- EXPLAIN 모드 및 PREP 명령 옵션을 포함한 자체 지시 방법 사용

  예를 들어, CURRENT EXPLAIN MODE 특수 레지스터는 RECOMMEND INDEXES로 설정됩니다. 이렇게 설정하면 SQL 컴파일러는 Explain 데이터 및 권장 색인을 캡쳐하여 ADVISE_INDEX 테이블에 배치합니다. 그러나 SQL문은 실행되지 않습니다.

  아니면, CURRENT EXPLAIN MODE 특수 레지스터는 EVALUATE INDEXES로 설정됩니다. 이렇게 설정하면 SQL 컴파일러는 사용자가 ADVISE_INDEX 테이블에 배치한 색인을 사용하게 됩니다. 사용자는 평가되어야 할 각 색인에 대한 새 행을 삽입합니다. 각 색인에 필요한 정보는 색인 이름, 테이블 이름, 평가되는 색인을 구성하는 컬럼 이름입니다. 일단 입력하면 특수 레지스터인 CURRENT EXPLAIN MODE가 EVALUATE INDEXES로 설정되어야 합니다. 그러면 SQL 컴파일러는 USE_INDEX 필드가 『Y』로 설

정된 ADVISE_INDEX 테이블을 스캔합니다. 이를 가상 색인이라고 합니다. EVALUTE INDEXS 모드에서 실행되는 모든 동적문에는 이 가상 색인이 사용 가능한 것으로 설명되어 있습니다. 명령문의 성능이 향상될 경우, SQL 컴파일러는 가상 색인을 사용하도록 선택합니다. 그렇지 않으면 색인이 무시됩니다. EXPLAIN 결과를 검토하여 사용자가 제안한 색인이 SQL 컴파일러에 의해 사용되었음을 알 수 있습니다. 사용된 색인은 액세스 개선을 위해 구현될 것으로 간주되어야 합니다.

- 콜 레벨 인터페이스(CLI) 사용

  응용프로그램을 작성할 때 이 인터페이스를 사용하고 있는 경우, 보조 프로그램을 사용할 수도 있습니다.

여러 가지 방법으로 보조 프로그램의 결과를 사용할 수 있습니다.

- 색인 보조 프로그램의 출력 해석

  조언 기능이 어떤 색인을 권장했는지를 알기 위해 다음 조회를 사용할 수 있습니다.

```
SELECT CAST(CREATION_TEXT as CHAR(200))
   FROM ADVISE_INDEX
```

- 색인 보조 프로그램의 권장사항 적용
- 색인 삭제 시기 알기

특정 조회에 대한 좀더 나은 권장사항을 확보하기 위해 사용자가 해당 조회 자체에 조언을 하도록 제안합니다. 색인 마법사를 사용하여 해당 조회만을 포함하는 워크로드를 빌드하여 하나의 조회에 대한 색인을 권장하도록 할 수 있습니다.

샘플 워크로드는 이벤트 모니터 출력을 통해 수집할 수 있습니다. 이벤트 모니터는 동적 SQL 실행을 수집하는 데 사용될 수 있습니다. 그런 다음 이 명령문은 조언 기능으로 피드백될 수 있습니다.

색인 마법사는 보조 프로그램 기능에 액세스하는 탁월한 방법을 제공하는, 단순하고 간단하며 사용하기 쉬운 시각적 인터페이스입니다.

제3부 시스템의 조정 및 구성

© Copyright IBM Corp. 1993, 2001

제8장 수행시의 성능

다음 주제에서는 런타임 시 SQL 조회의 성능에 영향을 미칠 수 있는 방법에 대한 정보를 제공합니다.

- DB2의 메모리 사용법
- 데이터베이스 버퍼 풀 관리
- 다중 데이터베이스 버퍼 풀 관리
- 버퍼 풀로 데이터 프리페치
- 프리페치 및 병렬 I/O를 위한 I/O 서버 구성
- 정렬
- 카탈로그 및 사용자 테이블 재구성
- DMS 장치를 위한 성능상의 고려사항
- 오버헤드 초기화 관리
- 데이터베이스 에이전트
- 데이터베이스 시스템 모니터 사용
- 메모리 확장

다음 장에서는 성능에 영향을 미치는 방법에 대한 정보도 제공합니다.

- 45 페이지의 『제3장 응용프로그램 고려사항』
- 101 페이지의 『제4장 환경상의 고려사항』
- 125 페이지의 『제5장 시스템 카탈로그 통계』

또한 *관리 안내서: 계획*에 있는 물리적 데이터베이스 설계 고려사항을 참조할 수도 있습니다.

© Copyright IBM Corp. 1993, 2001

DB2의 메모리 사용법

DB2에서 사용 가능한 많은 구성 매개변수는 시스템의 메모리 사용에 영향을 줍니다. 일부는 서버의 메모리에, 일부는 클라이언트의 메모리에, 어떤 것은 양쪽에 모두 영향을 미칩니다. 메모리는 시스템의 서로 다른 영역과 서로 다른 시간에 할당되기도 하고, 할당이 해제되기도 합니다.

시스템 관리자는 시스템상의 전반적인 메모리 사용이 편중되지 않고 균형을 이루도록 관리해야 합니다. 운영 체제에서 수행 중인 다양한 응용프로그램은 메모리를 다른 방식으로 사용합니다. 예를 들어, 일부 응용프로그램은 파일 시스템 캐쉬를 사용하는 반면, 데이터베이스 관리 프로그램은 데이터 캐쉬에 운영 체제 기능 대신 자체 버퍼 풀을 사용합니다. 자세한 내용은 281 페이지의 『메모리 사용에 영향을 미치는 매개변수 설정』을 참조하십시오.

그림22에서는 데이터베이스 관리 프로그램이 서로 다른 유형의 메모리를 사용함을 보여줍니다. 메모리 사용을 보여주는 이 그림은 Enterprise - Extended Edition도, 다중 논리 노드 환경도 아님을 가정한 것입니다. Enterprise - Extended Edition이나 다중 논리 노드 환경에서는 여러 개의 데이터베이스 관리 프로그램 공유 메모리가 설정되어 있습니다(노드마다 하나씩).

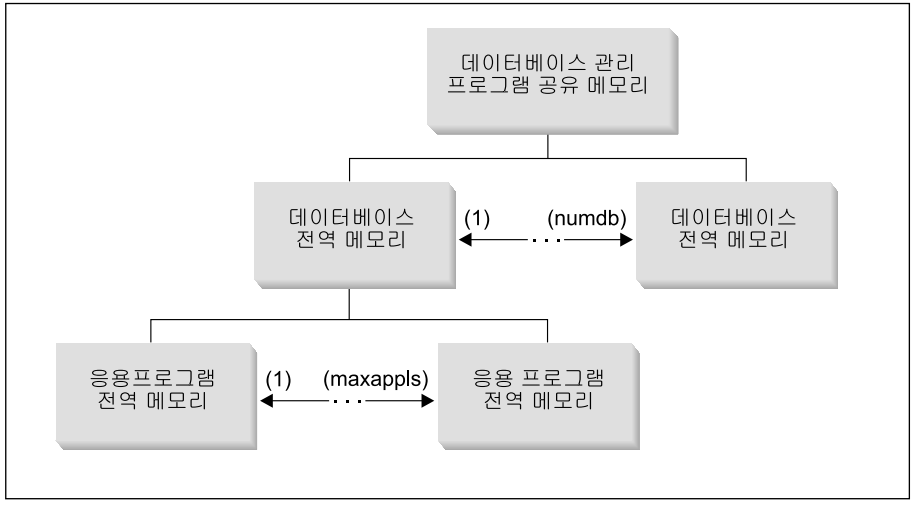


그림 22. 데이터베이스 관리 프로그램에 의해 사용되는 메모리 유형

메모리는 다음과 같은 시기에 데이터베이스 관리 프로그램 인스턴스에 할당됩니다.

- 데이터베이스 관리 프로그램이 시작될 경우(db2start), 『데이터베이스 관리 프로그램 공유 메모리』라고 표시된 영역이 할당되며, 이는 데이터베이스 관리 프로그램이 중단(db2stop)될 때까지 할당된 상태로 남아 있습니다. 이 영역에는 데이터베이스 관리 프로그램이 모든 데이터베이스 연결에 걸친 활동을 관리하는 데 필요한 정보가 들어 있습니다. 첫 번째 응용프로그램이 데이터베이스에 연결될 때 전역 메모리 영역 및 개인용 메모리 영역이 모두 할당됩니다.
- 데이터베이스가 처음으로 활성화되거나 연결될 경우, 『데이터베이스 전역 메모리』가 할당됩니다. 데이터베이스 전역 메모리는 데이터베이스에 연결될 수 있는 모든 응용프로그램에서 사용되고, 버퍼 풀, 잠금 목록, 데이터베이스 힙(heap) 및 유틸리티 힙(heap)과 같은 메모리 영역을 포함합니다.
- 응용프로그램이 데이터베이스에 연결되면 『응용프로그램 전역 메모리』가 할당됩니다. (이는 파티션된 데이터베이스 환경 또는 *intra_parallel* 구성 매개변수가 작동 가능한 경우에만 발생합니다.) 이 메모리는 데이터를 공유하고 에이전트 사이에 활동을 조정하기 위해 응용프로그램 대신 에이전트 작업에 사용됩니다.
- (이전 도표에는 표시되어 있지 않음) 에이전트가 특정 응용프로그램에 대한 작업에 지정될 경우(연결 요청의 결과로 또는 병렬 환경에서 새 SQL 요청의 결과로), 해당 에이전트에 대해 『에이전트 개인용 메모리』가 할당됩니다. 에이전트 개인 메모리 영역은 에이전트에 대해 할당되고, 정렬 힙과 응용프로그램 힙과 같은 특정 에이전트에 의해서만 사용되는 메모리 할당사항을 포함하고 있습니다.

  데이터베이스가 하나의 응용프로그램에 의해 이미 사용 중이면 연결된 후속 응용프로그램은 대신 할당된 에이전트 개인용 메모리와 응용프로그램 전역 공유 메모리만을 가지게 됩니다.

274 페이지의 그림22에서 구성 매개변수 설정이 메모리에 어떤 영향을 줄 수 있는지를 보여줍니다. 특히, 다음 목록의 매개변수는 특정 목적으로 할당된 메모리 양을 제한할 수 있습니다. (파티션된 데이터베이스 환경에서 이 메모리는 모든 데이터베이스 파티션에서 필요합니다.)

- *numdb*는 (서로 다른 응용프로그램에 의해 사용 중인) 동시에 활동할 수 있는 데이터베이스의 최대수를 정의합니다. 각각의 데이터베이스는 자신만의 전역 메모리 영역을 가지고 있으므로, 잠재적으로 할당된 메모리양은 이 매개변수의 값이 증가함에 따라 커집니다.
- *maxappls*는 하나의 데이터베이스에 연결될 수 있는 응용프로그램의 최대수를 정의합니다. 해당 데이터베이스에 대한 『에이전트 개인용 메모리』 및 『응용프로그램 전역 메모

리』용으로 할당될 수 있는 메모리 양에 영향을 줍니다. (이 매개변수는 각각의 데이터베이스 별로 다르게 설정할 수 있음에 유의하십시오.)

- (이전 도표에는 표시되어 있지 않음) *maxagents*(그리고 병렬 처리의 경우에는 *max_coordagents*)는 인스턴스 내의 모든 사용 중인 데이터베이스에서 동시에 존재할 수 있는 데이터베이스 관리 프로그램 에이전트의 수를 제한합니다. *maxappls*와 더불어, 이 매개변수는 『에이전트 개인용 메모리』 및 『응용프로그램 전역 메모리』용으로 할당된 메모리 양을 제한합니다. (에이전트에 대한 자세한 내용은 311 페이지의 『데이터베이스 에이전트』를 참조하십시오.)

277 페이지의 그림23에는 응용프로그램을 지원하는 데 사용되는 메모리 양이 요약되어 있습니다. 다음 구성 매개변수를 사용하면 "메모리 세그먼트"(논리 메모리 부분)의 수를 제한하여 이 메모리 *크기*를 제어할 수 있습니다.

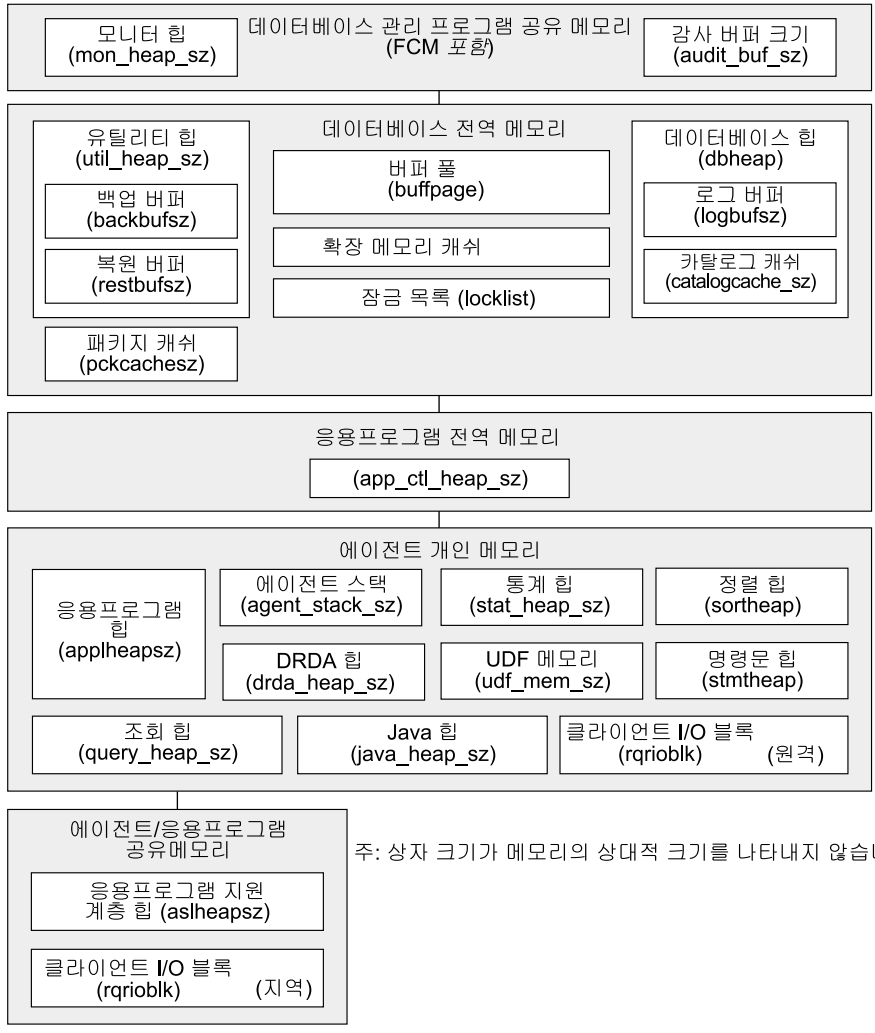


그림 23. 데이터베이스 관리 프로그램이 메모리를 사용하는 방법

데이터베이스 관리 프로그램 공유 메모리

데이터베이스 관리 프로그램이 수행하려면 메모리 공간이 필요합니다. 이 공간은 특히 파티션 내 및 파티션간 병렬 처리 환경에서는 매우 클 수 있습니다. 다음 절을 검토하여 이 공간의 *크기*를 추정하고 제어할 수 있습니다.

- 311 페이지의 『데이터베이스 에이전트』. 응용프로그램 대신에 수행하는 에이전트에는 특히 *maxagents* 값이 적합하지 않은 경우, 상당히 큰 메모리 공간이 필요합니다.

- 282 페이지의 『FCM 요구사항』. 파티션된 데이터베이스 시스템의 경우, FCM(Fast Communications Manager)에는 상당히 큰 메모리 공간이 필요합니다(특히, *fcm_num_buffers*가 적합하지 않은 경우).

  FCM 메모리 요구사항은 파티션된 데이터베이스 시스템이 다중 논리 노드를 사용하는지 여부에 따라 FCM 버퍼 풀로부터 할당되거나 데이터베이스 관리 프로그램 공유 메모리 및 FCM 버퍼 풀로부터 할당됩니다. 자세한 내용은 다음의 FCM 버퍼 풀에 대한 설명을 참조하십시오.

FCM 버퍼 풀

다중 논리 노드를 갖지 않는 파티션된 데이터베이스 시스템을 사용 중인 경우, 데이터베이스 관리 프로그램 공유 메모리 및 FCM 버퍼 풀이 그림24에 나와 있습니다.

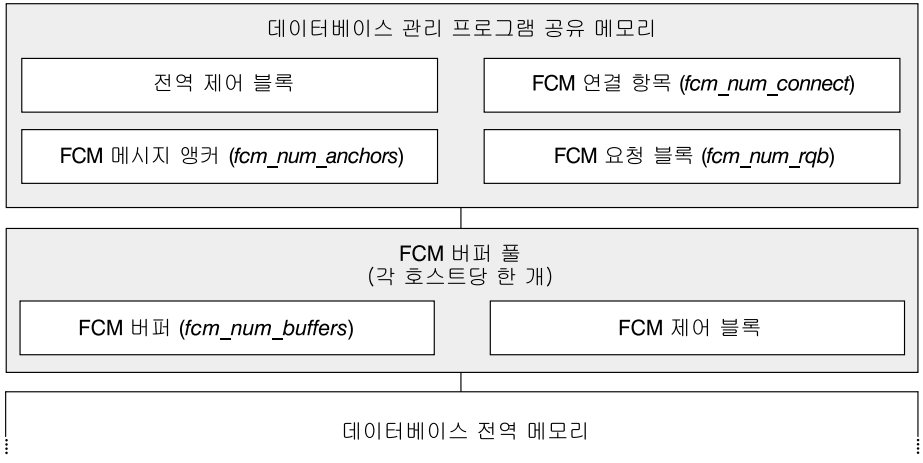


그림 24. 다중 논리 노드를 사용하지 않을 경우의 FCM 버퍼 풀

다중 논리 노드를 사용하는 파티션된 데이터베이스 시스템의 경우, 데이터베이스 관리 프로그램 공유 메모리 및 FCM 버퍼 풀이 279 페이지의 그림25에 나와 있습니다.

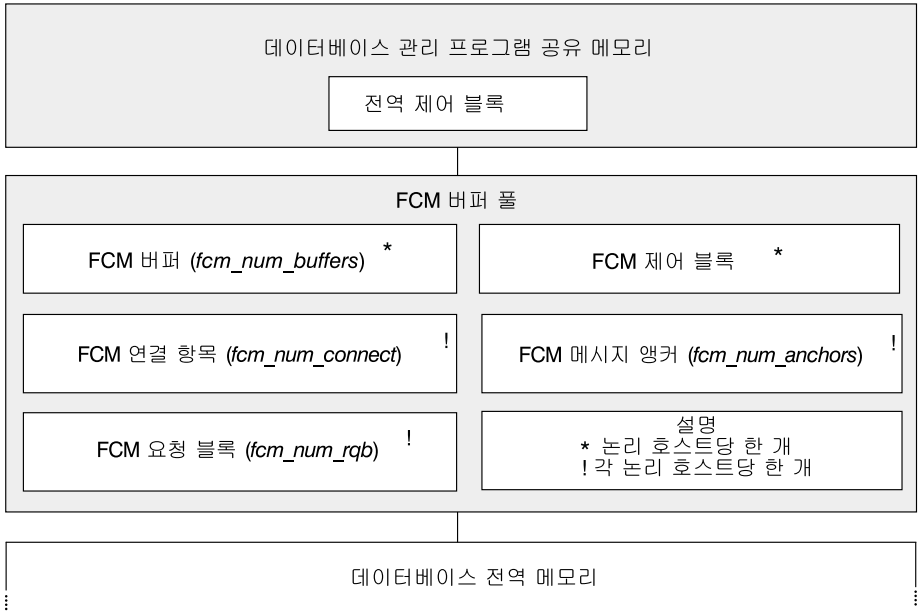


그림 25. 다중 논리 노드를 사용할 경우의 FCM 버퍼 풀

데이터베이스 전역 메모리

데이터베이스 전역 메모리는 다음 구성 매개변수에 의해 영향을 받습니다.

- 메모리 세그먼트 수는 *numdb*에 의해 제한됩니다(553 페이지의 『동시에 사용 중인 데이터베이스의 최대수(numdb)』 참조).
- 메모리 세그먼트의 최대 *크기*는 다음 매개변수의 값에 의해 결정됩니다.
  - 396 페이지의 『버퍼 풀 *크기*(buffpage)』(버퍼 풀 *크기*가 -1인 경우) 또는 버퍼 풀이 작성되거나 변경될 때 지정된 명시적 *크기*
  - 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
  - 400 페이지의 『데이터베이스 힙(dbheap)』
  - 404 페이지의 『유틸리티 힙 *크기*(util_heap_sz)』
  - 454 페이지의 『확장 저장 메모리 세그먼트 *크기*(estore_seg_sz)』
  - 454 페이지의 『확장 저장영역 메모리 세그먼트의 수(num_estore_segs)』
  - 410 페이지의 『패키지 캐쉬 *크기*(pckcachesz)』

응용프로그램 전역 메모리

응용프로그램 전역 메모리는 다음 구성 매개변수: 412 페이지의 『응용프로그램 제

어 힙 *크기*(app_ctl_heap_sz)』에 의해 영향을 받습니다. 병렬 처리 시스템에 대해 공간은 하나의 데이터베이스 파티션에서 같은 응용프로그램 대신 작업 중인 에이전트간에 공유되는 응용프로그램 제어 힙(heap)에서도 필요합니다. 첫 번째 에이전트가 응용프로그램의 요청을 수신하기 위해 연결을 요청하는 경우, 힙이 할당됩니다. 에이전트는 조정 에이전트 또는 서브에이전트가 될 수 있습니다. (자세한 내용은 311 페이지의 『데이터베이스 에이전트』를 참조하십시오.)

에이전트 개인용 메모리

- 메모리 세그먼트의 수는 다음 중 낮은 것으로 제한됩니다.
  - 모든 사용 중인 데이터베이스에 대한 *maxappls*의 총합(456 페이지의 『사용 중인 응용프로그램의 최대수(maxappls)』 참조)
  - *maxagents* 값(462 페이지의 『에이전트의 최대수(maxagents)』 참조).
- 메모리 세그먼트의 최대 *크기*는 다음 매개변수의 값에 의해 결정됩니다.
  - 418 페이지의 『응용프로그램 힙 *크기*(applheapsz)』
  - 414 페이지의 『정렬 힙 *크기*(sortheap)』
  - 417 페이지의 『명령문 힙 *크기*(stmtheap)』
  - 419 페이지의 『통계 힙 *크기*(stat_heap_sz)』
  - 420 페이지의 『조회 힙 *크기*(query_heap_sz)』
  - 421 페이지의 『DRDA 힙 *크기*(drda_heap_sz)』
  - 422 페이지의 『UDF 공유 메모리 세트 *크기*(udf_mem_sz)』
  - 424 페이지의 『에이전트 스택 *크기*(agent_stack_sz)』

에이전트/응용프로그램 공유 메모리

- 에이전트/응용프로그램 공유 메모리 세그먼트(지역 클라이언트에 대한)의 총 수는 다음 중 낮은 것으로 제한됩니다.
  - 모든 사용 중인 데이터베이스에 대한 *maxappls*의 총합(456 페이지의 『사용 중인 응용프로그램의 최대수(maxappls)』 참조)
  - *maxagents*의 값(462 페이지의 『에이전트의 최대수(maxagents)』 참조) 또는 (병렬 시스템에 대해) *max_coordagents*(465 페이지의 『조정 에이전트의 최대 수(max_coordagents)』 참조)
- 에이전트/응용프로그램 공유 메모리는 다음에 의해서도 영향을 받습니다.
  - 429 페이지의 『응용프로그램 지원 계층 힙 *크기*(aslheapsz)』
  - 432 페이지의 『클라이언트 I/O 블록 *크기*(rqrioblk)』

메모리 사용에 영향을 미치는 매개변수 설정

메모리를 할당하는 매개변수는 이 값을 사용할 근거가 확실하게 제시되지 않은 한 최대량의 메모리가 설치된 시스템의 경우에도 *절대로* 최대값으로 설정되면 안됩니다. 여러 매개변수를 사용하면 데이터베이스 관리 프로그램이 머신에서 사용 가능한 모든 메모리를 매우 쉽고 빠르게 확보할 수 있습니다. 또한 대량의 메모리를 관리하는 것은 데이터베이스 관리 프로그램의 입장에서는 상당히 부담이 되는 작업이 될 수 있으므로, 이는 더 많은 오버헤드를 발생시키게 됩니다.

일부 UNIX 기반 운영 체제는 프로세스가 메모리를 할당하고 프로세스 자체가 교체 공간으로 페이지아웃되지 않는 경우에 교체 공간을 할당합니다. 이 경우, 총 공유 메모리 *크기*가 페이징 공간의 동등한 양에 의해 지원되는지 확인해야 합니다.

대부분의 구성 매개변수의 경우, 메모리는 단지 필요할 때에만 확약됩니다. 이러한 매개변수는 특정 메모리 힙의 최대 *크기*를 반영합니다. 이 규칙의 중요한 예외는 메모리가 매개변수 값에 따라 완전히 확약되는 다음과 같은 매개변수입니다.

- 396 페이지의 『버퍼 풀 *크기*(buffpage)』(버퍼 풀 *크기*가 -1인 경우) 또는 버퍼 풀이 작성되거나 변경될 때 지정된 명시적 *크기*
- 415 페이지의 『정렬 힙 임계값(sheapthres)』
- 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
- 429 페이지의 『응용프로그램 지원 계층 힙 *크기*(aslheapsz)』
- 536 페이지의 『FCM 메시지 앵커의 수(fcm_num_anchors)』
- 537 페이지의 『FCM 버퍼 수(fcm_num_buffers)』
- 539 페이지의 『FCM 연결 항목의 수(fcm_num_connect)』
- 539 페이지의 『FCM 요청 블록의 수(fcm_num_rqb)』

이들 매개변수 유형에 적합한 값은 벤치마킹에 의해 가장 적절하게 결정될 수 있는데, 여기서 대표적이고 최악의 경우를 고려한 SQL문을 서버에 대해 수행하여 성능에 대한 리턴 값이 감소하는 지점에 이를 때까지 매개변수 값이 조정됩니다. 성능과 매개변수 값을 비교한 값이 그래프로 표시되는 경우, 곡선이 올라가거나 하강하는 시점에서 할당을 추가하면 값이 추가되지 않고 메모리만 낭비한다는 사실을 나타냅니다. (자세한 내용은 363 페이지의 『제12장 벤치마크 테스팅』를 참조하십시오.)

여러 매개변수에 대한 메모리 할당의 상한값은 기존의 하드웨어 및 운영 체제의 메모리 용량을 초과할 수도 있습니다. 이들 한계 선택은 앞으로 있을 증가에서 가능합니다.

유효한 매개변수 범위는 379 페이지의 『제13장 DB2 구성』의 매개변수 설명 부분을 참조하십시오.

FCM 요구사항

다음의 FCM(Fast Communications Manager) 구성 매개변수를 구성하는 경우, 기본값으로 시작하십시오.

- 537 페이지의 『FCM 버퍼 수(fcm_num_buffers)』
- 539 페이지의 『FCM 요청 블록의 수(fcm_num_rqb)』
- 539 페이지의 『FCM 연결 항목의 수(fcm_num_connect)』
- 536 페이지의 『FCM 메시지 앵커의 수(fcm_num_anchors)』

이 매개변수를 조정하려면, 데이터베이스 시스템 모니터를 사용하여 사용 가능한 버퍼, 사용 가능한 메시지 앵커, 사용 가능한 연결 항목, 사용 가능한 요청 블록에 대한 낮은 물결 마크를 모니터하십시오. 낮은 물결 마크가 해당하는 사용 가능한 데이터 항목 수의 10%보다 적은 경우, 해당하는 매개변수 값을 증가시키십시오. 데이터베이스 시스템 모니터에 대한 자세한 내용은 318 페이지의 『데이터베이스 시스템 모니터 사용』을 참조하십시오.

FCM 통신 사용에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

데이터베이스 버퍼 풀 관리

버퍼 풀은 데이터베이스 페이지(테이블 행 또는 색인 항목을 포함한)가 임시로 읽히고 변경되는 저장영역입니다. 버퍼 풀의 목적은 데이터베이스 시스템의 성능을 향상시키는 것입니다. 데이터를 디스크에서 액세스하는 것보다 메모리에서 액세스하는 것이 훨씬 빠릅니다. 그러므로 데이터베이스 관리 프로그램이 디스크로부터 읽거나 디스크에 쓰는 횟수가 적을수록 성능이 향상됩니다.

하나 이상의 버퍼 풀 구성은 가장 중요한 단일 조정 영역입니다. 그 이유는 대부분의 데이터 조작이 데이터베이스에 연결된 응용프로그램에서 발생하기 때문입니다(대형 오브젝트(LOB)와 Long 필드 데이터는 제외).

응용프로그램이 테이블 행에 처음으로 액세스할 때 데이터베이스 관리 프로그램은 행을 포함한 페이지를 버퍼 풀에 배치합니다. 응용프로그램이 다음으로 데이터를 요청할 때 버퍼 풀이 최초로 점검됩니다. 버퍼 풀에 보존된 페이지에 요청된 데이터가 있을 경우, 데이터베이스 관리 프로그램은 디스크 저장영역에 가서 요청된 데이터를 검색할 필요가 없습니다. 디스크 저장영역에서 데이터를 검색할 필요가 없어지면 성능이 빨라집니다.

데이터베이스가 활성화되거나 첫 번째 응용프로그램이 데이터베이스에 연결될 때 버퍼 풀에 연관된 저장영역이 할당됩니다. 버퍼 풀은 1차로 응용프로그램에 할당됩니다. 일단 응용프로그램이 모두 연결 해제되면 버퍼 풀에 연관된 저장영역은 할당 해제됩니다.

페이지는 데이터베이스가 종료될 때까지 또는 다른 페이지에 페이지가 차지한 공간이 필요할 때까지 남아 있습니다. 버퍼 풀에서 다른 페이지를 가져오기 위해 선택된 공간은 다음 기준을 사용하여 선택합니다.

- 페이지에 대한 최종 참조
- 페이지를 본 최종 에이전트에 의해 다시 참조되는 페이지 가능성
- 페이지의 데이터 유형
- 메모리에서 변경되지 않고 디스크에 기록되지 않은 페이지의 유무(변경된 페이지는 겹쳐쓰기 전에 디스크에 항상 기록됩니다.)

주: 변경된 페이지가 디스크에 기록된 후에 다른 페이지에 페이지가 차지한 공간이 필요한 경우가 아니면 버퍼 풀에서 제거되지 않습니다. 페이지를 겹쳐쓸 때까지 페이지의 데이터가 필요하면 다시 액세스될 수 있습니다.

버퍼 풀을 작성할 때, 기본적으로 페이지 크기는 4KB입니다. 버퍼 풀을 작성할 때 페이지 크기를 4KB, 8KB, 16KB 또는 32KB 중 하나로 설정하도록 선택할 수 있습니다. 하나의 페이지 크기를 사용하여 버퍼 풀을 작성할 경우, 동일한 페이지 크기를 사용하여 작성된 테이블 공간만이 버퍼 풀에 연관될 수 있습니다. 작성하

고 나면 버퍼 풀의 페이지 *크기*를 변경할 수 없습니다. 새로 작성한 버퍼 풀은 원하는 *크기*의 페이지로 작성되어야 합니다.

Windows 시스템에서 대용량 메모리 이용

Windows 2000으로 작업할 때 버퍼 풀 *크기*가 최대 64GB인 것은 DB2 및 운영 체제 *크기* 이하에서 지원됩니다. (이것은 DB2가 시스템의 기본 제품이라고 가정합니다.) 이 지원은 Microsoft AWE(Address Windowing Extensions)를 통해 사용할 수 있습니다.

AWE가 어떤 *크기*의 버퍼 풀도 사용할 수 있더라도 AWE를 더 큰 버퍼 풀에서 사용해야 하는 경우, 기타 권장되는 Windows 제품이 있습니다. Windows 2000 고급 서버는 최대 8GB의 메모리를 지원합니다. Windows 2000 Data Center 서버는 최대 64GB의 메모리를 지원합니다.

DB2 및 Windows 2000이 AWE 버퍼 풀을 지원하도록 올바르게 구성되어야 합니다. AWE를 사용하는 버퍼 풀이 데이터베이스에 있어야 합니다.

3GB의 사용자 공간을 할당하려면 /3GB Windows 2000 부트 옵션을 사용합니다. 이 옵션은 보다 큰 AWE 창 *크기* 사용을 허용합니다. AWE 메모리 인터페이스를 통해 4GB보다 큰 메모리에 액세스할 수 있게 하려면 /PAE Windows 2000 부트 옵션을 사용하십시오. 올바른 부트 옵션을 선택했는지 검증하려면 제어판 아래에서 시스템을 선택한 다음 『시작 및 복구』를 선택하십시오. 드롭 다운 목록에서 사용 가능한 부트 옵션을 볼 수 있습니다. 원하는 부트 옵션(/3GB 또는 /PAE)이 선택되었으면 AWE 지원을 설정할 다음 타스크로 진행할 준비가 된 것입니다. 원하는 옵션이 선택용으로 사용할 수 없는 것이면 해당 옵션을 시스템 드라이브의 boot.ini 파일에 추가해야 합니다. boot.ini 파일에는 운영 체제가 시작될 때 수행될 조치 목록이 들어 있습니다. 기존 매개변수의 목록 끝에 /3GB나 /PAE 또는 둘 다(공백으로 구분)를 추가하십시오. 변경된 이 파일을 일단 저장했으면 위에서 언급한 대로 올바른 부트 옵션을 검증 및 선택할 수 있습니다.

또한 Windows 2000은 DB2를 설치한 사용자와 『메모리의 잠금 페이지』를 직접 연관시키도록 수정해야 합니다. 『메모리의 잠금 페이지』 권한을 설정하려면 일단 DB2를 설치한 사용자로서 Windows 2000에 로그온하고 Windows 2000의 시작

메뉴 아래에서『관리 도구』폴더를 선택한 다음『로컬 보안 규정』프로그램을 선택하십시오. 지역 규정하에서『메모리의 잠금 페이지』권한에 대한 사용자 권한 지정을 선택할 수 있습니다.

DB2에는 DB2_AWE 레지스트리 변수의 설정이 필요합니다. 이 레지스트리 변수를 올바르게 설정하려면 AWE의 지원을 허용할 버퍼 풀의 버퍼 풀 ID를 알고 있어야 합니다. 버퍼 풀 ID는 SYSCAT.BUFFERPOOLS 시스템 카탈로그 뷰의 BUFFERPOOLID 컬럼에 있습니다. 또한 할당할 실제 페이지 수와 주소 창 페이지를 알고 있어야 합니다. 할당할 실제 페이지 수는 총 사용 가능한 실제 페이지보다 적은 값이어야 합니다. 선택된 실제 수는 작업 환경에 따라 다릅니다. 예를 들어, 시스템에 DB2 및 데이터베이스 응용프로그램만 사용된 환경에서는 DB2_AWE 변수와 함께 사용되는 값으로 총 실제 페이지 크기보다 0.5 - 1GB가 적게 선택할 수 있습니다. 시스템에 기타 비 데이터베이스 응용프로그램이 사용된 환경에서는 기타 응용프로그램을 위한 더 많은 실제 페이지를 허용하려면 총계에서 감하는 값을 증가시켜야 합니다. DB2_AWE 레지스트리 변수에 사용된 번호는 AWE의 지원에 사용되고 DB2가 사용할 실제 페이지 수입니다. /3GB Windows 2000 부트 옵션이 효과를 나타낼 때 주소 창 페이지의 상한값은 1.5GB 또는 2.5GB입니다.

DB2 레지스트리 변수 DB2_AWE 설정에 대한 자세한 내용은 이 책의 뒷부분에 있는『부록 A. DB2 레지스트리 및 환경 변수』의 레지스트리 변수를 참조하십시오.

버퍼 풀 페이지 작업

버퍼 풀의 페이지는 다음과 같이 서로 다른 속성을 가질 수 있습니다.

- 이 페이지는 현재 읽혀지고 있거나 갱신되고 있는 페이지입니다. 다른 에이전트가 읽을 수는 있지만, 갱신할 수는 없습니다.
- 『더티(Dirty)』페이지란 데이터는 변경되었지만 아직 디스크에 기록되지 않은 페이지입니다. 페이지가 디스크에 기록된 후 기록된 페이지는『클린(Clean)』으로 간주되며 버퍼 풀에 남아 있습니다. 클린 페이지가 차지한 공간은 새 페이지로 사용할 수 있으며, 연관된 확장 저장영역 캐쉬(정의된 경우)로의 이주에 사용 가능합니다.

버퍼 풀에서 변경된 페이지가 차지하는 공간의 백분율이 *chngpgs_thresh* 구성 매개변수에 의해 지정된 매개변수 값을 초과하는 경우, 페이지는 버퍼 풀에서 디스크로 기록될 수 있습니다. 또한 페이지 정리자 에이전트를 둘 이상 포함하도록 데이터베이스를 구성해야 합니다. 이 에이전트가 변경된 페이지를 디스크에 기록하면 데이터베이스 에이전트는 버퍼 풀에서 사용 가능한 페이지를 찾을 수 있습니다.

페이지 정리자 에이전트는 수행되지 않을 경우, 데이터베이스 에이전트에 의해 수행되어야 하는 I/O를 수행합니다. 결과적으로, 응용프로그램은 더 빨리 수행될 수 있는데, 트랜잭션은 해당 데이터베이스 에이전트가 페이지를 디스크에 기록하는 동안 대기할 필요가 없기 때문입니다. (페이지 정리자 에이전트는 데이터베이스 에이전트와 동시에 작업을 수행할 수 있기 때문에 때때로 *비동기 페이지 정리자* 또는 *비동기 버퍼 기록기*라고도 지칭됩니다.)

페이지 정리자 에이전트의 수를 변경하려면, *num_iocleaners* 구성 매개변수를 사용하십시오. (기본값은 하나의 페이지 정리자 에이전트를 작성하는 것입니다.) 자세한 내용은 448 페이지의 『비동기 페이지 정리자(cleaner)의 수(num_iocleaners)』를 참조하십시오. 이 매개변수를 1과 데이터베이스에 있는 실제 디스크 수 사이의 값으로 설정하십시오. 이 숫자가 클수록 갱신 중심의 워크로드를 처리할 때 성능이 좋아집니다. 이는 또한 비동기 데이터 또는 색인 페이지 기록 횟수에 대해 데이터 또는 색인 페이지 기록 횟수가 많을 경우에도 해당됩니다.

시스템에 문제가 발생한 경우, 디스크에 페이지를 기록하면 데이터베이스 복구가 더 빨리 되는데, 이유는 데이터베이스 관리 프로그램이 데이터베이스 로그 파일을 사용하는 것보다 디스크에서 더 많은 버퍼 풀을 재빌드할 수 있기 때문입니다. 결과적으로, 복구 중에 읽어야 하는 로그의 크기가 다음 최대값을 초과할 경우 페이지 정리가 요청됩니다.

```
logfilsiz * softmax
```

여기서

- *logfilsiz*는 로그 파일의 크기를 나타냅니다(475 페이지의 『로그 파일의 크기(logfilsiz)』 참조).
- *softmax*는 다음의 데이터베이스 손상이 복구되는 로그 파일 백분율을 나타냅니다(484 페이지의 『복구 범위 및 소프트 점검점 간격(softmax)』 참조).

예를 들어, softmax 값이 250인 경우, 2.5 로그 파일에는 손상 발생시 복구되어야 하는 변경사항이 들어 있습니다.

복구 중에 로그 읽기 시간을 최소화하기 위해 데이터베이스 시스템 모니터를 사용하여 페이지 정리가 요청된 횟수를 추적할 수 있습니다. 자세한 내용은 *시스템 모니터 안내 및 참조서* 매뉴얼의 *pool_lsn_gap_clns(트리거된 버퍼 풀 로그 공간 정리자)* 모니터 요소 설명을 참조하십시오.

복구 중에 읽어야 하는 로그의 크기는 로그에서의 다음 위치에 따라 다릅니다.

- 가장 최근에 기록된 로그 레코드
- 버퍼 풀의 데이터에 가장 오래된 변경사항을 기술하는 로그 레코드

다음 그림에서는 모든 I/O를 수행하는 데이터베이스 에이전트와 비교하여 페이지 정리자 에이전트 및 데이터베이스 에이전트간에 버퍼 풀 관리 작업을 어떻게 공유할 수 있는지를 보여줍니다.

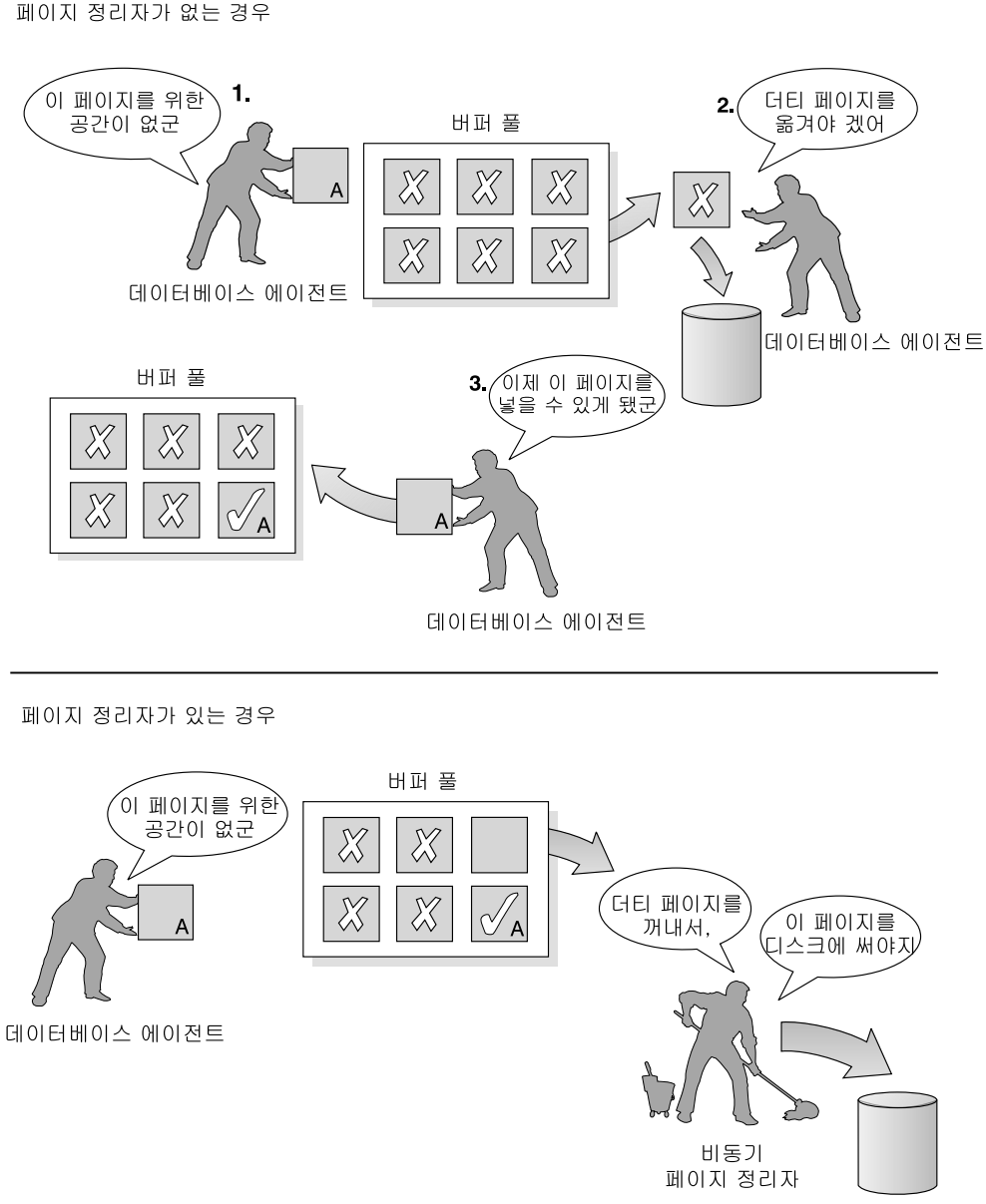


그림 26. 비동기 페이지 정리자. 『더티(dirty)』 페이지는 디스크에 기록됩니다.

다중 데이터베이스 버퍼 풀 관리

데이터베이스마다 최소한 하나의 버퍼 풀이 필요합니다. 그러나 필요에 따라 하나의 데이터베이스에서 여러 버퍼 풀과 다른 *크기*의 버퍼 풀 작성을 선택할 수 있습니다. CREATE, ALTER 및 DROP BUFFERPOOL문을 통해 버퍼 풀을 작성, 변경 또는 제거할 수 있습니다. CREATE TABLESPACE 및 ALTER TABLESPACE문으로 버퍼 풀에 캐쉬될 데이터를 지정할 수 있습니다.

buffpage 구성 매개변수는 버퍼 풀의 *크기*가 SYSCAT.BUFFERPOOLS 카탈로그 뷰에서 -1로 지정되는 경우, 버퍼 풀 *크기*를 지정합니다. (그렇지 않으면 이 매개변수는 무시됩니다.) 버퍼 풀 *크기*는 DDL문 ALTER BUFFERPOOL 또는 CREATE BUFFERPOOL로 설정될 수 있습니다.

새 데이터베이스는 IBMDEFAULTBP라는 기본 버퍼 풀을 가지며 플랫폼에 의해 *크기*가 결정됩니다. 데이터베이스가 작성되거나 이주되면 다른 버퍼 풀이 이에 대해 작성될 수 있습니다.

데이터베이스 설계에 대해 작업할 때 페이지 *크기*가 8KB인 테이블이 최상이라고 결정했을 수 있습니다. 결과적으로, 페이지 *크기*가 8KB인 버퍼 풀을 작성해야 합니다(같은 페이지 *크기*를 갖는 하나 이상의 테이블 공간과 함께).

파티션된 데이터베이스 환경에서 데이터베이스에 대한 각 버퍼 풀은 모든 데이터베이스 파티션에 대해 같은 기본 정의를 갖습니다. (그렇지 않으면, CREATE BUFFERPOOL문에서 지정되거나 버퍼 풀 *크기*가 특정 데이터베이스 파티션에서 ALTER BUFFERPOOL문을 통해 변경됩니다.)

페이지 *크기*가 4KB인 테이블 공간을 작성하고 특정 버퍼 풀에 이를 지정하지 않으면 테이블 공간은 기본 버퍼 풀에 지정됩니다. 페이지 *크기*가 4KB보다 큰 (8KB, 16KB, 32KB) 테이블 공간을 작성하면 이를 동일한 페이지 *크기*를 사용하는 버퍼 풀에 지정해야 합니다. 이 버퍼 풀이 현재 사용되고 있지 않으면 DB2는 테이블 공간을 동일한 페이지 *크기*(하나가 사용 가능한 경우)를 사용하며 현재 사용 중인 버퍼 풀에 지정하려고 합니다. 이러한 지정은 임시적인 것입니다. 데이터베이스가 다시 활성화되고 원래 지정된 버퍼 풀이 활성화되면 DB2는 테이블 공간을 해당 버퍼 풀에 지정합니다.

다른 페이지 *크기*를 사용하는 버퍼 풀에 테이블 공간을 추가할 경우에는 ALTER TABLESPACE문을 사용할 수 없습니다.

버퍼 풀을 작성하거나 변경하는 동안, 모든 버퍼 풀에 필요한 전체 메모리는 데이터베이스 관리 프로그램에서 사용 가능해야 합니다. 그러면 모든 버퍼 풀은 데이터베이스가 시작할 때 할당될 수 있습니다. 데이터베이스가 시작될 때 이 메모리가 사용 가능하지 않으면 데이터베이스 관리 프로그램은 기본 버퍼 풀(IBMDEFAULTBP) 및 다른 페이지 *크기*로 정의된 각 버퍼 풀 중 하나를 시작하지만 최소 *크기*(각 16페이지)로 시작합니다. 이 최소 버퍼 풀의 *크기*는 레지스트리 변수 DB2_OVERRIDE_BPF를 사용하여 대체할 수 있습니다. 레지스트리 및 환경 변수에 대한 자세한 내용은 575 페이지의 『부록A. DB2 레지스트리 및 환경 변수』를 참조하십시오. 버퍼 풀을 시작하려는 시도가 실패할 때마다 경고 메시지가 리턴됩니다. 데이터베이스는 구성이 변경되어 데이터베이스를 완벽하게 재시작할 수 있을 때까지, 이 조작 상태에서 계속 실행합니다.

데이터베이스 관리 프로그램이 최소 *크기* 값으로 시작할 수 있도록 하는 이유는 데이터베이스에 대한 연결을 허용하기 위해서입니다. 그런 다음 올바른 버퍼 풀 *크기*로 데이터베이스를 재시작할 목표로, 버퍼 풀 *크기*를 재구성하거나 기타 중요한 타스크를 수행할 수 있습니다. 이러한 상황에서는 장시간 동안 데이터베이스를 작동하지 마십시오.

주: 기본 버퍼 풀과 연관된 *크기* 및 속성을 변경할 수는 있지만 삭제할 수 없습니다. 또한 사용되는 플랫폼에 따라 각 버퍼 풀마다 최소 *크기*가 있습니다.

버퍼 풀에 할당된 메모리 양이 크면 이점이 많습니다. 예를 들어, 버퍼 풀 *크기*가 클수록 다음과 같은 이점이 있습니다.

- 자주 요청된 데이터 페이지가 버퍼 풀에서 유지될 수 있고 더 빠른 액세스를 허용합니다. I/O 조작이 적으면 I/O 경합이 줄어들어 응답 시간이 덜 걸리고 I/O 조작에 필요한 프로세서 자원도 줄어듭니다.
- 같은 응답 시간에 트랜잭션 비율을 늘릴 수 있는 기회를 제공합니다.
- 카탈로그 테이블, 자주 참조되는 사용자 테이블 및 색인과 같이 자주 사용된 디스크 저장 장치에 대한 I/O 경합을 막을 수 있습니다. 임시 테이블 공간이 들어 있는 디스크 저장 장치에서 I/O 경합이 줄어들어, 조회에 필요한 정렬에도 도움이 됩니다.

한 개 또는 여러 개의 버퍼 풀 선택

다음과 같은 조건이 시스템에 적용될 경우, 하나의 버퍼 풀만을 사용해야 합니다.

- 총 버퍼 공간이 10 000 4KB 페이지 미만인 경우
- 전문화된 조정을 수행할 만큼 응용프로그램 지식을 가진 사용자가 없는 경우
- 테스트 시스템에서 작업할 경우

시스템이 이들 조건에 제한을 받지 않으면 다음과 같은 잠재적 성능 향상을 위해 둘 이상의 버퍼 풀을 사용해 보십시오.

- 별도의 버퍼 풀에 임시 테이블 공간을 지정하여 임시 저장, 특히 정렬 중심 조회가 필요한 조회에 대해 더 나은 성능을 제공할 수 있습니다.
- 많은 짧은 갱신 트랜잭션 응용프로그램에 의해 반복적으로 신속하게 액세스해야 할 데이터가 있을 경우, 데이터가 들어 있는 테이블 공간을 별도의 버퍼 풀에 이동시키는 것을 고려해야 합니다. 이 버퍼 풀의 *크기*가 적합하면 응답 시간이 더 작고 트랜잭션 비용이 더 적기 때문에 버퍼 풀 페이지가 있을 가능성이 더 많습니다.
- 별도의 버퍼 풀에서 데이터를 분리하여 특정 응용프로그램, 데이터, 색인에 도움을 줄 수 있습니다. 예를 들어, 자주 갱신되는 테이블과 색인은, 자주 조회는 되지만 갱신은 거의 되지 않는 테이블 및 색인에서 분리한 버퍼 풀에 지정할 수 있습니다. 이렇게 변경하면 잦은 조회(테이블의 두 번째 세트)에서 빈번한 갱신(테이블의 첫 번째 세트에서)의 영향을 줄입니다.
- 응용프로그램이 매우 큰 테이블에 임의로 액세스해야 하는 경우와 같이, 거의 사용되지 않는 응용프로그램에 의해 액세스된 데이터에 더 작은 버퍼 풀을 사용할 수 있습니다. 이 경우, 버퍼 풀 메모리의 데이터는 한 번의 조회 동안만 보유하면 됩니다. 이 데이터에 대한 작은 버퍼 풀을 유지하는 것이 더 좋고, 기타 사용(예를 들어, 다른 버퍼 풀에 대해 사용)에 대한 여분의 메모리를 프리업(free up)하는 것이 더 좋습니다.
- 서로 다른 활동과 데이터를 별도의 버퍼 풀에 분리한 후에는 통계 및 계정 추적을 통해 우수하며, 비교적 비용이 적게 드는 성능 진단 데이터를 생성할 수 있습니다.

버퍼 풀로 데이터 프리페치

버퍼 풀로 색인과 데이터 페이지를 프리페치할 경우, I/O가 완료되기까지 기다려야 하는 시간이 줄어들어 성능이 향상됩니다. 페이지의 프리페치는 데이터의 사용을 예상하여 디스크에서 하나 이상의 페이지를 검색하는 것을 의미합니다. 프리페치에는 두 종류의 범주가 있습니다.

- *순차적 프리페치*는 응용프로그램에서 페이지를 요구하기 전에 계속되는 페이지를 버퍼 풀로부터 읽는 메커니즘입니다. (자세한 내용은 『순차적 프리페치의 이해』를 참조하십시오.)
- *목록 프리페치* 또는 목록 순차적 프리페치는 필요한 데이터 페이지가 연속적으로 존재하지 않더라도, 데이터 페이지에 효율적으로 액세스할 수 있는 방법입니다. (자세한 내용은 294 페이지의 『목록 프리페치의 이해』를 참조하십시오.)

이러한 두 가지 데이터 읽기 방법이 일반적인 읽기에 추가됩니다. 일반적인 읽기는 하나 또는 몇 페이지의 연속 페이지만이 검색될 경우 사용됩니다. 일반적인 읽기 중에는 데이터의 한 페이지가 전송됩니다.

프리페치를 가능하게 하는 것에 대한 자세한 내용은 295 페이지의 『프리페치 및 병렬 I/O를 위한 I/O 서버 구성』을 참조하십시오.

순차적 프리페치의 이해

단 한 번의 I/O 조작으로 여러 개의 연속된 페이지를 버퍼 풀로 읽어들이는 것은 응용프로그램 수행과 연관된 오버헤드를 크게 줄일 수 있습니다. 추가로, 여러 페이지 범위에서 동시에 읽어 병렬로 I/O 조작을 실행하면 I/O 조작을 완료하기 위해 응용프로그램이 기다리는 시간을 줄이는 데 도움이 될 수 있습니다.

데이터베이스 관리 프로그램이 순차적 I/O가 적당하고 프리페치가 성능 향상에 도움이 될 것이라고 판단할 경우, 프리페치가 시작됩니다. 테이블 스캔 및 테이블 정렬의 경우, 데이터베이스 관리 프로그램은 순차적 프리페치가 I/O 성능을 향상시킬 것이라고 쉽게 결정할 수 있습니다. 이 경우, 데이터베이스 관리 프로그램은 자동으로 순차적 프리페치를 시작합니다. 다음 예는 테이블 스캔이 필요한 작업이며, 순차적 프리페치가 적용될 수 있는 좋은 예입니다.

```
SELECT NAME FROM EMPLOYEE
```

데이터베이스 관리 프로그램이 프리페치할 수 있는 페이지 수는 CREATE TABLESPACE 또는 ALTER TABLESPACE문을 가진 PREFETCHSIZE절을 사용하여 각 테이블 공간에 대해 정의될 수 있습니다. 지정된 값은 SYSCAT.TABLESPACES 시스템 카탈로그 테이블의 PREFETCHSIZE 컬럼에서 유지보수됩니다.

테이블 공간 및 테이블 공간 컨테이너 수에 대한 EXTENTSIZE 값의 배수로 PREFETCHSIZE 값을 명시적으로 설정하는 것은 좋은 연습입니다. (다른 컨테이너를 사용하기 전에 데이터베이스 관리 프로그램이 컨테이너에 기록하는 페이지의 수를 Extent *크기*라고 합니다. 자세한 내용은 *관리 안내서: 계획*의 『테이블 공간 설계 및 선택』을 참조하십시오.) 예를 들어, Extent *크기*가 16페이지이고 테이블 공간에 두 컨테이너가 있으면 프리페치 양을 32페이지로 설정하도록 선택합니다.

다른 작업 단위(UOW)에 이 페이지가 필요한 할 경우, 데이터베이스 관리 프로그램은 데이터의 프리페치로 인해 버퍼 풀로부터 페이지가 제거되지 않도록 버퍼 풀의 사용을 모니터합니다. 문제점이 발생하지 않도록 하기 위해 데이터베이스 관리 프로그램은 프리페치되는 페이지의 수를 테이블 공간에 지정한 것보다 작은 양으로 제한하기도 합니다.

프리페치 *크기*를 설정하는 것은 성능에 있어서 중요한 의미를 갖는데, 특히 대형 테이블 스캔의 경우가 그렇습니다. 데이터베이스 시스템 모니터 및 다른 시스템 모니터 도구를 사용하면 테이블 공간에 대한 PREFETCHSIZE를 조정하는 데 도움이 됩니다. 예를 들어, 다음에 대한 정보를 수집할 수 있습니다.

- 운영 체제에서 사용 가능한 모니터링 도구를 사용하여 조회를 기다리는 I/O가 있습니다.
- 데이터베이스 시스템 모니터에 의해 제공된 데이터 요소인 *pool_async_data_reads(버퍼 풀 비동기 데이터 읽기)*를 보면 프리페치가 발생하고 있습니다. 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

I/O가 대기 상태이고 조회가 데이터를 프리페치 중인 경우, PREFETCHSIZE의 값을 증가시키려 할 수 있습니다. 프리페치가 I/O 대기의 원인이 아닌 경우도 있는데, 이 경우 PREFETCHSIZE 값을 증가시키는 것은 조회의 성능을 향상시킬 수 없게 됩니다.

모든 유형의 프리페치에서 프리페치 *크기*가 테이블 공간에 대한 Extent *크기*의 배수이며, 테이블 공간의 extent가 분리된 컨테이너인 경우, 다중 I/O 조작은 병렬로 수행됩니다. 더 좋은 성능을 위해 컨테이너는 별도의 물리 장치를 사용하도록 구성되어야 합니다. 병렬 프리페치에 대한 자세한 내용은 295 페이지의 『프리페치 및 병렬 I/O를 위한 I/O 서버 구성』을 참조하십시오.

순차적 검출에 대한 이해

때때로 순차적 프리페치가 성능을 향상시킬 수 있을지가 한 번에 뚜렷하게 판단되지 않는 경우가 있습니다. 이 경우, 데이터베이스 관리 프로그램은 I/O를 모니터하여 순차 페이지 읽기가 발생하고 있으면 데이터베이스 관리 프로그램이 프리페치를 활성화시킬 수 있습니다. 이런 경우, 적합하다고 여겨지면 데이터베이스 관리 프로그램에 의해 프리페치가 활성화되거나 비활성화될 수 있습니다. 이러한 유형의 순차적 프리페치를 *순차적 검출*이라 하며, 색인과 데이터 페이지에 모두 적용됩니다. *seqdetect* 구성 매개변수(451 페이지의 『순차적 검출 플래그(seqdetect)』 참조)를 사용하여 데이터베이스 관리프로그램이 순차적 검출을 수행해야 하는지 여부를 제어할 수 있습니다. 순차적 검출이 실행되는 경우, 다음의 SQL문이 순차적 프리페치부터 이점을 얻는지 결정할 수 있습니다.

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

이 예에서 최적화 알고리즘은 EMPNO 컬럼에 색인을 사용하여 테이블을 스캔하도록 선택할 수도 있습니다. 테이블이 이 색인으로 제대로 잘 클러스터링되어 있으면 데이터 페이지 읽기는 거의 순차적인 것이고, 프리페치로 성능을 향상시킬 수도 있습니다. 이 경우에는 데이터 페이지 프리페치가 발생합니다.

이 예에서는 색인 페이지 프리페치도 발생할 수 있습니다. 많은 수의 색인 페이지가 검사되어야 하고 데이터베이스 관리 프로그램이 색인 페이지를 읽는 순차 페이지 발생을 발견하게 되면 색인 페이지 프리페치가 발생합니다.

목록 프리페치의 이해

목록 프리페치 또는 목록 순차적 프리페치는 필요한 데이터 페이지가 인접해 있지 않더라도, 데이터 페이지에 효율적으로 액세스할 수 있는 방법입니다. 목록 프리페치는 하나 또는 여러 개의 색인 액세스와 함께 사용될 수 있습니다.

프리페치와 파티션 내 병렬 처리

프리페치는 파티션 내 병렬 처리 성능에 매우 중요하며, 색인이나 테이블을 스캔하는 경우 다중 서브에이전트를 사용합니다. 이러한 병렬 스캔으로 데이터 소비 비율이 더 커지게 되고, 프리페치 비율이 더 높아집니다.

적합하지 않은 프리페치 비용은 직렬 스캔의 경우보다 병렬 스캔이 더 높습니다. 직렬 스캔을 실행할 때 프리페치가 발생하지 않는 경우, 에이전트가 항상 I/O를 기다려야 하기 때문에 조회가 더 느려집니다. 병렬 스캔을 실행할 때 프리페치가 발생하지 않는 경우, 모든 서브에이전트는 I/O를 기다리는 하나의 서브에이전트를 기다려야 합니다.

프리페치의 중요성 때문에 파티션 내 병렬 처리로 프리페치는 더 적극적으로 수행됩니다. 순차적 검출 메커니즘은 인접한 페이지간에 더 큰 간격을 허용하여 페이지는 순차적인 것으로 간주될 수 있습니다. 이런 간격의 폭은 스캔과 관련된 서브에이전트 수와 함께 증가합니다.

프리페치 및 병렬 I/O를 위한 I/O 서버 구성

프리페치를 작동 가능하게 하기 위해 데이터베이스 관리 프로그램은 *I/O 서버*라는 별도의 제어 스레드를 시작하여 페이지 읽기를 수행합니다. 결과적으로, 조회 처리는 두 개의 병렬 활동(데이터 처리(CPU) 및 데이터 페이지 I/O)으로 나뉩니다. I/O 서버는 CPU 처리 활동으로부터 프리페치 요청을 기다립니다. 이러한 프리페치 요청에는 필요한 데이터 요구사항을 충족시키기 위한 I/O에 대한 설명이 포함됩니다. 프리페치를 하는 이유에 따라 데이터베이스 관리 프로그램이 프리페치 요청을 생성하는 시기 및 방법을 결정합니다. (자세한 내용은 292 페이지의 『순차적 프리페치의 이해』 및 294 페이지의 『목록 프리페치의 이해』를 참조하십시오.)

다음 그림에서는 I/O 서버를 사용하여 데이터를 버퍼 풀로 프리페치하는 방식을 나타냅니다.

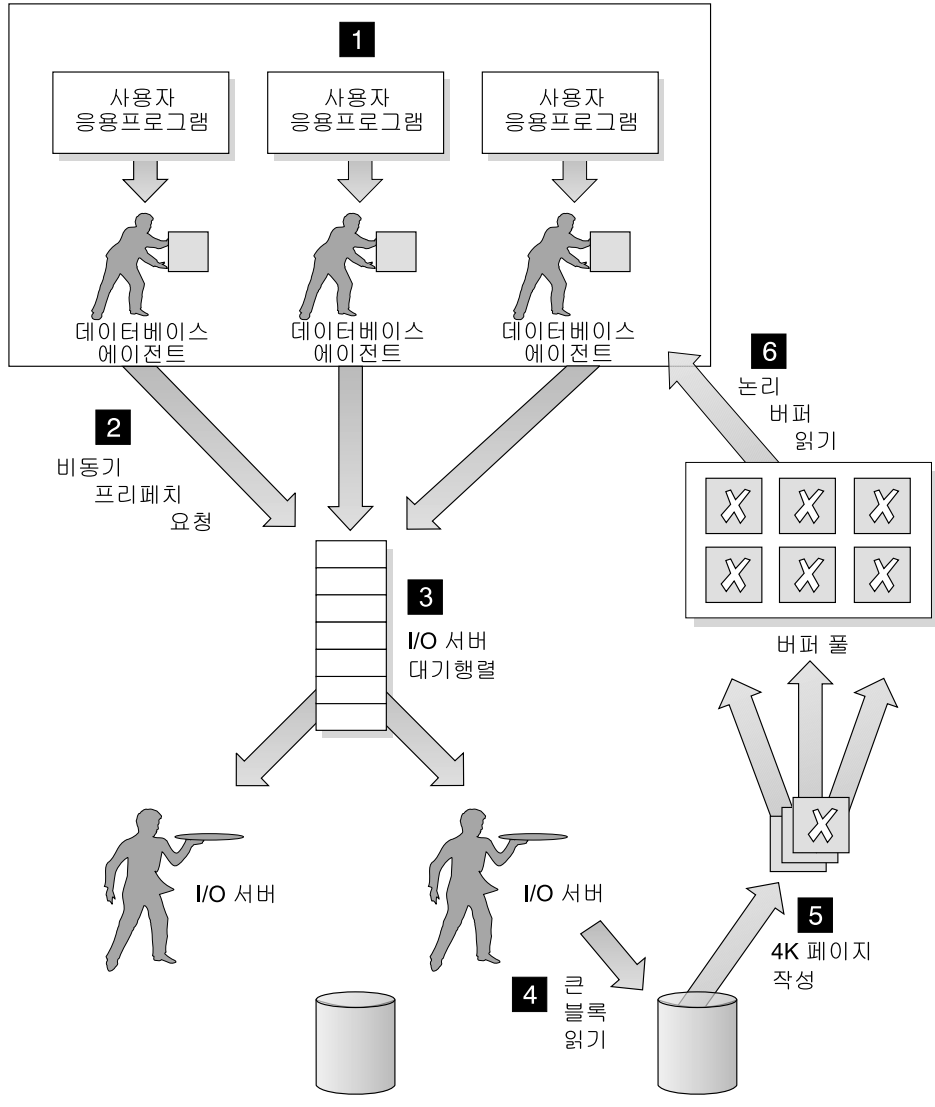


그림 27. I/O 서버를 사용한 데이터 프리페치

다음 단계는 그림27에 설명되어 있습니다.

1 사용자 응용프로그램은 SQL 요청을 데이터베이스 관리 프로그램에 의해 사용자 응용프로그램에 지정했던 데이터베이스 에이전트로 전달합니다.

2, **3**

데이터베이스 에이전트는 SQL 요청을 충족시키는 데 필요한 데이터를 확보하기 위해 프리페치를 사용할 것인지를 결정하고 I/O 서버 대기행렬에 프리페치 요청을 작성합니다.

4, **5**

맨처음 가능한 I/O 서버가 대기행렬에서 프리페치 요청을 읽고, 테이블 공간에서 데이터를 읽어 버퍼 풀에 넣습니다. 대기행렬의 프리페치 요청 수와 *num_ioservers* 구성 매개변수에 의한 I/O 서버 수에 따라 다중 I/O 서버는 테이블 공간으로부터 데이터를 동시에 페치할 수 있습니다.

6 데이터베이스 에이전트는 SQL 요청에 대한 결과를 사용자 응용프로그램으로 리턴하기 위해 버퍼 풀에 있는 데이터 페이지에 대해 필요한 조치를 수행합니다.

num_ioservers 구성 매개변수로 충분한 I/O 서버를 구성하면 데이터의 프리페치를 사용할 수 있는 조회 성능을 크게 향상시킬 수 있습니다. I/O 서버를 여유 있게 구성한다고 해서 성능에 나쁜 영향을 주는 것은 아닌데, 이는 여분의 I/O 서버는 사용되지 않으며 메모리 페이지가 페이지아웃되기 때문입니다. 각 I/O 서버 프로세스에는 번호가 매겨지고 데이터베이스 관리 프로그램은 항상 사용 가능한 것 중 가장 낮은 번호를 사용하므로, 결과적으로 번호가 높은 프로세스는 전혀 사용되지 않을 수도 있습니다.

구성해야 하는 I/O 서버의 수를 결정하려면 다음을 고려하십시오.

- 데이터베이스에서 동시에 일어나는 작업량. 즉, 해당 시간에 I/O 서버 대기행렬에 프리페치 요청을 쓸 수 있는 데이터베이스 에이전트의 수
- I/O 서버가 병렬로 작업을 수행할 수 있는 가장 높은 등급. 자세한 내용은 『병렬 I/O를 가능하게 하기』를 참조하십시오.

병렬 I/O에 대한 기회를 최대화하려면 *num_ioservers*를 최소한 데이터베이스의 실제 디스크 수로 설정하십시오.

병렬 I/O를 가능하게 하기

테이블 공간에 대해 여러 컨테이너가 있는 상황에서 데이터베이스 관리 프로그램은 *병렬 I/O*를 시작할 수 있습니다. 병렬 I/O는 데이터베이스 관리 프로그램이 여러 I/O 서버를 사용하여 단일 조회의 I/O 요구사항을 처리할 수 있는 기능을 말

합니다. 각 I/O 서버는 별도의 컨테이너에 대해 I/O 워크로드로 지정되고 병렬로 읽혀지는 여러 컨테이너를 허용합니다. I/O를 병렬로 수행하면 I/O 작업 처리량에 상당한 향상을 가져올 수 있습니다.

별도의 I/O 서버가 각각의 컨테이너에 대한 워크로드를 처리하는 반면, 병렬로 I/O를 수행할 수 있는 I/O 서버의 실제 수는 요청되는 데이터가 분산되어 있는 물리 장치의 수로 제한됩니다. 또한 이것은 물리 장치만큼의 I/O 서버가 필요하다는 의미입니다.

병렬 I/O를 시작하고 사용하는 방법은 I/O를 수행하는 이유에 따라 다릅니다.

- **순차적 프리페치**

  순차적 프리페치 수행의 경우, 프리페치 *크기*가 테이블 공간에 대한 Extent *크기*의 배수일 때, 병렬 I/O가 시작됩니다. 각각의 프리페치 요청은 extent 경계를 따라 다수의 더 작은 요청으로 나뉘어집니다. 그런 다음 더 작은 요청은 서로 다른 I/O 서버에 지정됩니다.

- **목록 프리페치**

  목록 프리페치 수행의 경우, 페이지의 각각의 목록은 데이터 페이지가 저장되어 있는 컨테이너에 따라서 더 작은 목록으로 나뉘어집니다. 그런 다음 더 작은 목록이 서로 다른 I/O 서버에 지정됩니다.

- **데이터베이스 또는 테이블 공간 백업 및 복원**

  데이터를 백업하거나 복원할 경우, 병렬 I/O 요청의 수는 백업 버퍼 *크기*를 Extent *크기*로 나눈 것과 같으며, 이는 컨테이너 수와 같은 최대값까지 가능합니다.

- **데이터베이스 또는 테이블 공간 복원**

  데이터 복원의 경우, 병렬 I/O 요청이 순차적 프리페치에 사용되는 것과 같은 방법으로 시작되고 분할됩니다. 데이터를 버퍼 풀로 복원하는 대신, 데이터가 직접 복원 버퍼에서 디스크로 이동됩니다.

- **로드**

  데이터를 로드하는 경우, LOAD 명령의 DISK_PARALLELISM 옵션을 사용하여 I/O 병렬 처리의 레벨을 지정할 수 있습니다. (지정되지 않은 경우, 사용되는 기본값은 테이블과 연관된 모든 테이블 공간에 대한 테이블 공간 컨테이너의 누적 수에 기초합니다.)

최적의 병렬 I/O 성능을 위해 다음을 확인하십시오.

- I/O 서버가 충분히 있어야 합니다. 데이터베이스 내에서 모든 테이블 공간에 사용되는 컨테이너의 수보다 조금 많은 수의 I/O 서버를 구성해야 합니다.
- Extent *크기*와 프리페치 *크기*가 테이블 공간에 대하여 적절한 것이어야 합니다. 프리페치 *크기*는 버퍼 풀을 과도하게 사용하는 것을 방지하기 위해 너무 크지 않아야 합니다. (이상적인 *크기*는 Extent *크기* 및 테이블 공간 컨테이너 수의 배수입니다.) 또한 Extent *크기*는 다소 작아야 하며, 8페이지에서 16페이지 범위의 값이 적당합니다.
- 컨테이너는 별도의 물리적 드라이브에 상주하도록 구성됩니다.
- 병렬 처리 수준이 일관성 있게 하려면, 모든 컨테이너의 *크기*가 같아야 합니다.

  하나 이상의 컨테이너가 다른 컨테이너보다 작은 경우, 잠재적인 최적화된 병렬 프리페치를 줄입니다. 예를 들면, 다음과 같습니다.

  – 더 작은 컨테이너가 가득 차게 되면, 추가 데이터는 나머지 컨테이너에 저장되고 컨테이너는 불균형하게 됩니다. 프리페치될 수 있는 데이터에서 컨테이너 수가 전체 컨테이너 수보다 작기 때문에 불균형 상태의 컨테이너는 병렬 프리페치의 성능을 줄입니다.

  – 더 작은 컨테이너가 나중에 추가되고 데이터가 다시 균형이 맞추어지면 더 작은 컨테이너에는 다른 컨테이너보다 적은 데이터가 들어 있습니다. 다른 컨테이너와 관련된 컨테이너의 데이터량은 병렬 프리페치를 최적화하지 않습니다.

  – 한 컨테이너가 더 크고 다른 모든 컨테이너가 가득 차면 이 컨테이너는 추가 데이터를 저장하는 단 하나의 컨테이너가 됩니다. 데이터베이스 관리 프로그램은 병렬 프리페치를 사용하여 이러한 추가 데이터에 액세스할 수 없습니다.
- 파티션 내 병렬 처리를 사용할 경우, 적합한 I/O 용량이 있습니다. 파티션 내 병렬 처리를 SMP 머신에서 사용하여 다중 프로세서에서 조회를 수행하는 동안의 조회 경과 시간을 줄일 수 있습니다. 각 프로세서를 사용 중으로 유지하려면 I/O 용량이 충분해야 하며, 보통 I/O 용량을 제공하는 추가 물리적 드라이브가 보통 필요합니다.

  프리페치는 I/O 용량을 효율적으로 사용하기 위해 더 높은 비율에서 발생해야 합니다. 프리페치 *크기*는 프리페치가 더 높은 비율에서 발생하도록 더 높아야

합니다. 프리페치 *크기*는 Extent *크기* 및 테이블 공간 컨테이너 수의 배수여야 합니다. 컨테이너는 별도의 물리적 드라이브에 상주하도록 구성하는 것이 이상적입니다.

필요한 물리적 드라이브 수는 드라이브 및 I/O 버스의 속도와 용량, 프로세서 속도에 따라 결정할 수 있습니다.

한 번에 다중 페이지 할당

SMS 테이블 공간은 요구에 따라 확장됩니다. 이 확장은 기본적으로 한 번에 한 페이지씩 수행됩니다. 그러나 특정 워크로드(예를 들어, 대량의 삽입 수행시)에서는 *db2empfa* 도구를 사용하여 DB2에 페이지 또는 extent 그룹의 테이블 공간을 확장하도록 지시하여 성능을 향상시킬 수 있습니다. *db2empfa* 도구는 *sqllib* 디렉토리의 *bin* 서브디렉토리에 위치합니다. 이를 수행하면 *multipage_alloc* 데이터베이스 구성 매개변수가 예로 설정됩니다. 이 도구에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

사용 가능한 메모리를 가장 효율적으로 이용할 수 있는 또 다른 방법이 321 페이지의 『메모리 확장』에 설명되어 있습니다.

정렬

정렬은 조회에 대해 자주 요구되고 적합한 정렬 힙(heap) 영역의 구성은 조회 성능에서 아주 중요한 문제입니다. 다음과 같은 경우에 정렬이 필요합니다.

- 요청된 순서화를 충족시키는 색인이 없습니다(예: ORDER BY절을 사용하는 SELECT문).
- 색인은 존재하지만 정렬이 색인을 사용하는 것보다 더 효율적입니다.
- 색인 작성(*indexsort* 구성 매개변수가 예로 설정된 경우)

여러 가지 정렬 유형

정렬 작업은 두 단계로 이루어집니다.

1. 정렬 단계
2. 정렬 단계의 결과 리턴

정렬이 두 단계에서 처리되는 방식은 정렬을 설명할 수 있는 다른 범주 또는 유형을 생기게 합니다. 정렬 단계 고려시, 정렬은『오버플로우』또는『비 오버플로우』로 나눌 수 있습니다. 정렬 단계의 결과 리턴 고려시, 정렬은『파이프』또는『비 파이프』로 나눌 수 있습니다.

오버플로우 및 비 오버플로우

정렬 중인 정보가 정렬 힙(정렬이 수행될 때마다 할당되는 메모리 블록)에 완전히 들어맞지 않을 경우, 임시 데이터베이스 테이블로 오버플로우됩니다. 오버플로우하지 않는 정렬은 그렇지 않은 정렬보다 항상 더 잘 수행됩니다.

파이프 및 비 파이프

정렬된 정보가 임시 테이블이 최종 정렬된 데이터 목록을 저장하지 않고 직접 리턴할 수 있는 경우,『파이프 정렬』로 간주됩니다. 정렬된 정보가 임시 테이블을 리턴해야 하는 경우,『비 파이프 정렬』로 간주됩니다. 파이프 정렬은 항상 비 파이프 정렬보다 더 잘 수행됩니다.

정렬에 영향을 미치는 매개변수 조정

다음 상황은 정렬 성능에 영향을 미칩니다.

- 다음 구성 매개변수에 대한 설정

  414 페이지의『정렬 힙 크기(sortheap)』

  각 정렬에 사용되는 메모리 양을 지정합니다.

  415 페이지의『정렬 힙 임계값(sheapthres)』

  모든 정렬에 대한 전체 인스턴스에 걸쳐 사용 가능한 정렬을 위한 전체 메모리 양을 제어합니다.

- 다량의 정렬을 포함하는 명령문
- 불필요한 정렬을 피하는 데 도움이 될 수 있는 색인 누락
- 정렬을 최소화하지 않는 응용프로그램 논리
- 병렬 정렬은 정렬의 성능을 향상시키지만, 명령문이 파티션 내 병렬 처리를 사용하는 경우에만 발생할 수 있습니다(297 페이지의『병렬 I/O를 가능하게 하기』참조).

정렬 성능 문제점의 표시기 검토

정렬의 전반적인 문제점에 대해 알려면정렬에 소요된 총 CPU 시간을 전체 응용 프로그램에 소요된 CPU 시간과 비교해 보십시오. 데이터베이스 시스템 모니터는 도움이 될 수 있습니다(318 페이지의 『데이터베이스 시스템 모니터 사용』 참조). 특히, 성능 모니터(『스냅샷 모니터』 및 『이벤트 모니터』로 구성되며 제어 센터에서 사용 가능함)는 기본적으로 기타 시간(예: *I/O* 및 *잠금 대기*)과 함께 *총 정렬 시간*도 나타냅니다.

총 정렬 시간이 다른 시간에 비해 큰 비율을 차지하면 기본적으로 표시되는 다른 값도 살펴 보십시오.

오버플로우된 정렬의 백분율

이 변수(스냅샷 모니터의 성능 세부사항 뷰에서)는 오버플로우된 정렬의 백분율을 나타냅니다. 오버플로우된 정렬의 백분율이 높으면 포스트 임계값 정렬이 수행될 경우 *sortheap* 및/또는 *sheapthres* 구성 매개변수의 값을 증가시키십시오. (다른 포스트 임계값 정렬이 있는지 알아보려면 데이터베이스 모니터를 사용하십시오.)

포스트 임계값 정렬

포스트 임계값 정렬이 높은 경우, *sheapthres*를 증가시키고/또는 *sortheap*을 줄이십시오.

일반적으로, 인스턴스에 걸쳐 사용 가능한 전체 정렬 메모리(*sheapthres*)를 과도한 페이징을 일으키지 않을 만큼 가능한 한 크게 만드십시오. 정렬 메모리 내에서 정렬이 전체적으로 수행될 수 있습니다. 그러나 이로 인해 운영 체제가 해당 정렬 메모리를 수용하기 위해 과도한 페이지 스와핑을 수행하게 될 경우, 대형 정렬 힙의 이점이 사라질 수 있습니다. 따라서 정렬 구성 매개변수를 조정할 때마다 운영 체제 모니터를 사용하여 시스템 페이징시 모든 변경사항을 추적하십시오.

주: DB2 부분 키 2진 정렬 기법에서의 향상을 위해 정수가 아닌 데이터 유형 키를 포함시킬 때, 긴 키를 정렬할 경우에 약간의 추가 메모리가 필요합니다. Long 키가 사용되고 있다고 확신하면 *sortheap* 구성 매개변수를 증가시키십시오.

또한 파이프 정렬시 응용프로그램이 해당 정렬과 연관되는 커서를 닫을 때까지 정렬 힙은 해제되지 않습니다. 그러므로 파이프 정렬은 커서가 닫힐 때까지 메모리를 사용할 수 있습니다.

정렬 성능 관리 기법

데이터베이스 시스템 모니터와 벤치마킹 기법 사용은 *sortheap* 및 *sheapthres* 구성 매개변수를 설정하는 데 도움이 될 수 있습니다. 각각의 데이터베이스 관리 프로그램과 해당 데이터베이스에 대해 다음을 수행하십시오.

- 대표 워크로드를 설정하고 수행하십시오.
- 각 적용 가능한 데이터베이스에 대해 벤치마크 워크로드 기간 중에 다음의 성능 변수에 대한 평균 값을 수집하십시오.
  - 사용 중인 총 정렬 힙(heap)
  - 사용 중인 정렬

  이러한 성능 변수는 스냅샷 모니터의 성능 세부사항 뷰에서 나타납니다.
- *sortheap*을 각 데이터베이스에 대한 *사용 중인 총 정렬 힙(heap)* 평균으로 설정하십시오.
- 다음을 수행하여 *sheapthres*를 설정하십시오.
  1. 가장 큰 *sortheap* 값을 갖는 인스턴스 내의 데이터베이스를 판별하십시오.
  2. 이 데이터베이스의 평균 정렬 힙(heap) 크기를 판별하십시오.

     판별하기가 너무 어려우면 최대 정렬 힙(heap)의 80%를 사용하십시오.
  3. *sheapthres*를 사용 중인 정렬의 평균 수에 위에서 계산된 평균 정렬 힙(heap) 크기를 곱한 값으로 설정하십시오.

     이것이 바람직한 초기 설정입니다. 그런 다음 벤치마크 기법을 사용하여 이 값을 조정할 수 있습니다.

정렬이 중요한 성능 문제점인 특정 응용프로그램과 명령문을 식별할 수도 있습니다.

- 응용프로그램과 명령문 레벨의 이벤트 모니터를 설정하여 가장 긴 총 정렬 시간을 갖는 응용프로그램을 식별할 수 있습니다.
- 이러한 각각의 응용프로그램 내에서 가장 긴 *총 정렬 시간*을 갖는 명령문을 찾으십시오.

- Visual Explain과 같은 도구를 사용하여 이러한 명령문을 조정하십시오.
- 적절한 색인이 있는지 확인하십시오. Visual Explain을 사용하여 주어진 명령문에 대한 모든 정렬 조작을 식별할 수 있습니다. 그런 다음 이 명령문에 의해 액세스되는 각각의 테이블에 대한 적절한 색인이 존재하는지 여부를 조사하십시오.

주: Explain 테이블을 통해 검색하여 조회에 정렬 조작이 있는지 식별할 수 있습니다. (자세한 내용은 643 페이지의 『부록C. SQL Explain 도구』를 참조하십시오.)

카탈로그 및 사용자 테이블 재구성

색인을 사용한 SQL문의 성능은 갱신, 삭제 또는 삽입이 많이 이루어지면 손상될 수 있습니다. 일반적으로, 새로 삽입된 행은 색인에 의하여 정의된 논리 순서와 똑같은 물리 순서에 배치될 수 없습니다(클러스터된 색인을 사용하지 않을 경우). 이는 논리적으로 순차적인 데이터가 순차적이지 않은 서로 다른 물리적 데이터 페이지에 있을 수도 있기 때문에, 데이터베이스 관리 프로그램이 추가로 읽기 조작을 수행해야 함을 의미합니다.

일반적으로, 테이블을 재구성하는 작업은 통계 작업보다 시간이 더 오래 걸립니다. 성능은 데이터에 대한 현재의 통계치를 구하거나 응용프로그램을 리바인드하여 현저하게 향상될 수 있으므로, 이를 실행해 보십시오. 이렇게 하여 성능이 향상되지 않으면, 테이블과 색인의 데이터가 효율적으로 배열되어 있지 않을 수도 있으므로 재구성이 도움이 됩니다. 이 절의 정보는 자체 테이블을 재인식에 적용할 뿐만 아니라, 재구성이 또한 필요한 시스템 카탈로그 테이블에 적용됩니다.

입력된 테이블의 경우, 지정되는 테이블 이름은 계층 구조의 루트 테이블 이름이어야 합니다.

REORGCHK 명령은 테이블의 물리적 특성에 대한 정보 및 이 테이블을 재구성하는 것이 도움이 될지 여부에 대한 정보를 리턴합니다. 이 명령은 명령행 처리기를 통해 사용될 수 있습니다. 명령 출력을 해석하는 방법을 비롯하여 자세한 내용은 *Command Reference*를 참조하십시오.

주: REORGCHK 명령은 확장 색인이나 선언된 임시 테이블의 데이터는 표시하지 않습니다.

REORG 유틸리티는 지정된 색인에 따른 물리적 순서로 데이터를 선택적으로 재배열합니다. REORG에는 색인과 함께 테이블의 데이터 순서를 지정하는 옵션이 있어서, 색인에 따라 테이블 데이터를 클러스터하고 RUNSTATS 유틸리티가 수집한 CLUSTERRATIO 또는 CLUSTERFACTOR 통계를 향상시킵니다. 결과적으로, 색인화된 순서에 따라 정렬된 행을 요구하는 SQL문은 더 효율적으로 처리될 수 있습니다. 또한 REORG는 사용하지 않는 빈 공간(ALTER TABLE을 사용할 때 PCTFREE를 지정한 경우에도)을 제거하여 테이블을 더욱 압축하여 저장합니다.

별칭과 함께 REORG 또는 REORGCHK 명령을 사용하지 마십시오.

REORG 유틸리티는 일반적으로 테이블 데이터 및 색인에 대해 작업할 모든 다른 응용프로그램이 오프라인이기를 요구합니다. 응용프로그램이 데이터에 대해 작업할 수 없는 시간을 제한하려는 작업 환경이 있을 수 있습니다. 이 환경에서는 온라인 색인 재구성 유틸리티의 사용을 고려해 보아야 합니다.

재구성 중에 발생하는 색인 재빌드에 필요한 로그 공간은 다음과 같이 계산됩니다.

```
2 * (10500 + ((색인 페이지의 수 / Extent 크기) * 110) +
(색인 페이지의 수 * 45) +
(색인 페이지의 수 / 16000) * 64 ))
```

계산의 다양한 부분은 색인이 재빌드됨에 따라 로그에 작성되고 기록되는 것과 연관된 오버헤드의 여러 가지 유형을 결정합니다.

테이블 데이터를 재구성해야 하는 시기를 결정하려면 다음과 같은 인수를 고려해야 합니다.

- 삽입, 갱신 및 삭제 활동의 양
- 높은 클러스터 비율의 색인을 사용하는 조회 성능에 대한 중요한 변경사항
- 통계 수행(RUNSTATS)은 조회 성능을 향상시키지 않습니다.
- REORGCHK 명령은 테이블을 재구성해야 할 필요가 있는지를 나타냅니다.

- 재구성이 완료될 때까지 테이블을 잠그는 CPU 시간, 경과 시간, REORG 유틸리티에서 일어난 감소된 동시성을 포함한 테이블을 재구성하는 비용

REORG 유틸리티를 실행하려면 테이블에 대한 SYSADM 권한, SYSMAINT 권한, SYSCTRL 권한 또는 CONTROL 특권이 있어야 합니다.

컬럼이 테이블에 추가되거나 테이블에 LOB 컬럼이 있는 경우, REORG 유틸리티는 원래의 테이블보다 훨씬 큰 임시 테이블을 사용합니다. 이러한 임시 테이블이 더 크다면 REORG 유틸리티에 의해 작성되는 결과의 영구 테이블도 역시 더 큽니다.

주: REORG 유틸리티로는 선언된 임시 테이블을 재구성할 수 없습니다.

REORG 유틸리티는 임시 테이블 공간을 지정하도록 하는데, 이것은 임시 REORG 테이블을 작성하는 데 사용됩니다. 임시 테이블 공간이 지정되지 않으면 REORG 유틸리티는 재구성될 테이블이 들어 있는 테이블 공간에 임시 REORG 테이블을 작성하게 됩니다. 다음 지침은 임시 테이블 공간을 사용할지 여부를 결정하는 데 도움이 될 수 있습니다.

- 임시 테이블 공간을 지정할 경우, 일반적으로 SMS 임시 테이블 공간을 지정하는 것이 좋습니다. DMS 임시 테이블 공간은 이러한 유형의 테이블 공간을 사용할 때 하나의 REORG만 진행될 수 있으므로 바람직하지 않습니다.
- 일반적으로 SMS 임시 테이블 공간을 지정하는 것이 좋습니다. 같은 테이블 공간을 사용하여 테이블을 재구성하는 것은 속도는 더 빠르지만 로깅 발생이 더 많아지고 재구성된 테이블을 위한 충분한 공간이 있어야 합니다. 임시 테이블 공간을 지정할 경우, 일반적으로 SMS 임시 테이블 공간을 지정하는 것이 좋습니다. DMS 임시 테이블 공간은 이러한 유형의 테이블 공간을 사용할 때 하나의 REORG만 진행될 수 있으므로 바람직하지 않습니다.

REORG 유틸리티는 열려 있는 모든 커서를 내재적으로 닫습니다.

4KB(8KB, 16KB 또는 32KB) 페이지 보다 큰 페이지를 사용하는 테이블 공간 내의 테이블을 재구성할 수 있음을 기억하십시오. 재구성시, 재구성 중에 사용되는 임시 테이블 공간의 크기는 기본 테이블 공간의 페이지 크기와 동일해야 합니다.

REORG 유틸리티가 성공적으로 완료되지 않은 경우, 임시 파일, 테이블 또는 테이블 공간을 삭제하지 **않습니다**. 이 파일과 테이블은 데이터베이스 관리 프로그램이 REORG 유틸리티에 의해 변경된 부분을 구간 복원하거나 실패하기 전에 재구성 진행 정도에 따라 재구성 작업을 완료하는 데 사용됩니다.

파티션된 데이터베이스에서 REORG 유틸리티는 파티션마다 데이터를 재구성합니다. 어떤 파티션에서 유틸리티가 실패할 경우, 실패한 파티션만이 구간 복원됩니다. 임시 테이블을 저장할 디렉토리 경로를 지정할 경우, 이 경로는 각 데이터베이스 파티션에서 데이터베이스 관리 프로그램에 의해 확장됩니다. 그러므로 다른 데이터베이스 파티션이 공유하는 경로를 지정할 경우, 임시 파일은 이 경로 아래의 다른 서브디렉토리(노드 이름으로 구분됨)에 저장됩니다.

온라인 색인 재구성

온라인 재구성은 사용자 정의 임계값에 색인 리프(leaf) 페이지에서 사용 가능한 최대 공간량을 제공함으로써 가능합니다. 리프 페이지에서 색인 키가 삭제될 경우 임계값이 교차되면, 이웃하는 색인 리프 페이지를 선택하여 두 리프 페이지가 병합될 수 있는지를 판별합니다. 페이지에 이웃하는 두 페이지의 병합이 발생할 만한 충분한 공백이 있으면 데이터베이스를 오프라인하지 않고 병합이 발생합니다.

이 색인의 온라인 재구성은 버전 6 이후에서 작성된 색인만 가능합니다. 색인 리프 페이지에 필수 내부 변경사항을 반영하려면 이러한 형식의 온라인 재구성 기능이 필요한 기존의 색인을 삭제한 다음 재작성해야 합니다. 특정 색인에 대한 온라인 색인 재구성을 조정하려면 색인이 작성될 때 MINPCTUSED 값을 지정하십시오. MINPCTUSED 값은 100 미만이어야 합니다. 이 값은 색인 페이지에서 사용된 공간의 백분율인 재구성 임계값이 됩니다. 이는 색인 리프 페이지와 이웃하는 색인 리프 페이지의 병합을 시도하기 전에 승인 가능한 최소값입니다. MINPCTUSED에 권장되는 값은 50% 미만인데, 이유는 목표가 두 개의 이웃하는 색인 리프 페이지를 병합하는 것이기 때문입니다. MINPCTUSED의 값이 0(기본값이기도 함)이면 온라인 재구성이 작동 불가능합니다.

다음과 같은 온라인 색인 재구성을 사용하기 위해 해제된 색인 리프 페이지는 다시 사용할 수 있습니다. 그러나 사용 가능한 페이지는 동일한 테이블 내의 다른 색인에 대해서만 사용 가능합니다. 테이블을 완전히 재구성하면 DMS 저장영역 모

델에 대해 작업할 때에는 다른 오브젝트에 대한 페이지가 해제되고 SMS 저장영역 모델에 대해 작업할 때에는 디스크 공간이 해제됩니다.

색인 비 리프 페이지는 다음과 같은 온라인 색인 재구성을 사용하기 위해 해제되지 않습니다. 그러나 테이블을 완전히 재구성하면 색인을 가능하면 작게 만듭니다. 색인 레벨뿐 아니라, 리프 및 비 리프 페이지의 숫자도 축소됩니다.

테이블 재구성의 필요성 제한

테이블 재구성의 필요성을 감소시키려면 테이블을 작성한 후 다음을 수행하십시오.

- PCTFREE를 추가하려면 테이블을 변경하십시오.
- 색인에서 PCTREE를 가진 클러스터링 색인을 작성하십시오.
- 데이터를 정렬하십시오.
- 데이터를 로드하십시오.

기존 테이블에 위 작업 중 하나를 수행한 후 클러스터링 색인으로 테이블을 채웁니다. 테이블의 PCTREE와 함께 클러스터링 색인은 원래의 정렬 순서를 보존합니다. 페이지에 공간이 충분하면 새 데이터를 정확한 페이지에 삽입할 수 있으므로, 클러스터링 색인의 클러스터링 특성을 유지보수할 수 있습니다. 더 많은 데이터가 삽입되고 테이블의 페이지가 가득 차면 레코드는 테이블 끝에 추가되어 점점 클러스터되지 않은 상태가 됩니다.

클러스터링 색인을 작성한 후 REORG 또는 정렬과 LOAD를 수행하는 것이 바람직합니다. 클러스터링 색인은 RUNSTATS 유틸리티에 의해 수집된 CLUSTERRATIO 또는 CLUSTERFACTORA 통계를 개선시키는 데이터의 특정 순서를 유지보수하려고 합니다.

REORG 중에 각 페이지에 남아 있어야 하는 여유 공간량은 테이블의 PCTREE 값에 의해 결정됩니다. 이 값이 설정되지 않으면 REORG는 데이터가 재구성될 때 페이지를 채웁니다.

DMS 장치를 위한 성능상의 고려사항

테이블 공간에 대해 데이터베이스 관리 공간(DMS) 장치 컨테이너를 사용 중인 경우, 다음 사항을 이해하여 환경을 효율적으로 관리해야 합니다.

- **파일 시스템 캐쉬**

  파일 시스템 캐쉬는 다음과 같이 수행됩니다.

  – DMS 파일 컨테이너(그리고 모든 SMS 컨테이너)의 경우, 운영 체제는 파일 시스템 캐쉬에서 페이지를 캐쉬할 수 있습니다.

  – DMS 장치 컨테이너 테이블 공간의 경우, 운영 체제는 파일 시스템 캐쉬에서 페이지를 캐쉬하지 않습니다.

  주: Windows NT에서 작업할 경우, 레지스트리 변수 DB2NTNOCACHE는 DB2가 NOCACHE 옵션을 사용하여 데이터베이스 파일을 열지 여부를 지정합니다. DB2NTNOCACHE=ON이면 파일 시스템 캐쉬이 제거됩니다. DB2NTNOCACHE=OFF이면 운영 체제는 DB2 파일을 캐쉬합니다. LONG FIELDS 또는 LOBS가 들어 있는 파일을 제외하고 모든 데이터에 적용됩니다. 시스템 캐쉬을 제거하면 더 많은 메모리를 데이터베이스에 사용 가능하게 할 수 있어서, 버퍼 풀 또는 정렬 힙을 증가시킬 수 있습니다.

- **데이터의 버퍼링**

  디스크로부터 읽혀지는 테이블 데이터는 보통 데이터베이스의 버퍼 풀에서 사용할 수 있습니다. (자세한 내용은 282 페이지의 『데이터베이스 버퍼 풀 관리』를 참조하십시오.) 일부 경우에는 응용프로그램이 실제 페이지를 사용하기 전에 데이터 페이지가 버퍼 풀로부터 해제될 수 있습니다. (버퍼 풀 공간에 다른 데이터 페이지가 필요할 때 이러한 상황이 일어날 수 있습니다.) 시스템 관리 공간(SMS) 또는 데이터베이스 관리 공간(DMS) 파일 컨테이너를 사용 중인 테이블 공간의 경우, 위의 파일 시스템 캐쉬에 관한 설명을 참조하십시오. 이렇게 하면 다른 경우에 요구되는 I/O를 제거할 수 있습니다.

  데이터베이스 관리 공간(DMS) 장치 컨테이너를 사용 중인 테이블 공간은 파일 시스템 또는 파일 시스템 캐쉬를 사용하지 **않습니다**. 결과적으로, 장치 컨테이

너를 사용한 DMS 테이블 공간을 가지고 이중 버퍼링을 수행할 수 없는 사실을 보완하기 위해 데이터베이스 버퍼 풀의 크기를 증가시키고 파일 시스템 캐쉬 크기를 줄이려고 할 수 있습니다.

시스템 레벨 모니터링 도구 사용을 통해 장치 컨테이너를 사용한 DMS 테이블 공간의 I/O가 같은 SMS 테이블 공간에 비해 더 높은 것을 알게 되는 경우, 이러한 차이는 위에서 논의된 이중 버퍼링으로 인한 것일 수 있습니다.

- **LOB 또는 LONG 데이터 사용**

  응용프로그램이 LOB 또는 LONG 데이터를 검색하는 경우, 데이터베이스 관리 프로그램은 해당 버퍼를 사용하여 데이터를 캐쉬하지 않습니다. 응용프로그램이 이 페이지 중 하나를 필요로 할 때마다 데이터베이스 관리 프로그램은 디스크로부터 이 페이지를 검색해야 합니다.

  그러나 LOB 또는 LONG 데이터를 SMS 또는 DMS 파일 컨테이너에 저장할 경우, 파일 시스템 캐쉉은 버퍼링을 제공하므로 결과적으로 성능이 향상됩니다.

  시스템 카탈로그에는 일부 LOB 컬럼이 들어 있기 때문에, SMS(또는 DMS 파일에) 테이블 공간에 해당 컬럼을 보존하는 것이 좋습니다.

오버헤드 초기화 관리

ACTIVATE DATABASE 명령은 선택된 데이터베이스를 시작합니다. 파티션된 데이터베이스에서 이 명령을 사용하면 모든 파티션에서 선택된, 파티션된 데이터베이스를 활성화하려는 시도가 일어납니다. 이 명령을 사용하면 데이터베이스 초기화 또는 시작시 응용프로그램 시간이 소요되지 않습니다.

ACTIVATE DATABASE 명령을 사용하여 초기화한 데이터베이스는 DEACTIVATE DATABASE 명령으로 종료되어야 합니다. 데이터베이스에서 연결 해제되는 최종 응용프로그램으로는 종료되지 않습니다. ACTIVATE 및 DEACTIVATE 명령에 대한 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

데이터베이스가 시작되지 않고 응용프로그램에 CONNECT TO(또는 내재된 연결)가 있을 경우, 응용프로그램은 데이터베이스 관리 프로그램이 해당 데이터베이스에 대한 작업을 수행하기 전에 필요한 데이터베이스를 시작하는 동안 기다려야 합니다. 이것은 첫 번째 응용프로그램이 특정 데이터베이스에 액세스하기 위해 창출

하는 시작 비용입니다. 파티션된 데이터베이스에서 시작 비용은 각 데이터베이스 파티션에서 발생합니다. 일단 데이터베이스가 시작되면 다른 모든 응용프로그램은 데이터베이스 시작과 연관된 시간 비용 없이 해당 데이터베이스에 연결하여 사용할 수 있습니다.

데이터베이스 에이전트

DB2 서버는 데이터베이스 관리 프로그램과 클라이언트와 지역 응용프로그램 사이에 통신을 도와야 합니다. UNIX 기반 환경은 *프로세스*에 근거한 아키텍처를 사용합니다. 예를 들어, DB2 통신 리스너는 프로세스로서 작성됩니다. OS/2 및 Windows NT와 같은 Intel 운영 체제는 성능을 최대화하기 위해 *스레드*에 근거한 아키텍처를 사용합니다. 예를 들어, DB2 통신 리스너는 DB2 서버의 시스템 제어기 프로세스 안에서 스레드로서 작성됩니다. 액세스 중인 데이터베이스마다 다양한 프로세스와 스레드는 다양한 데이터베이스 타스크(예: 프리페치, 통신, 로깅)를 처리하기 위해 시작됩니다.

중요한 프로세스와 스레드 중 하나는 데이터베이스 에이전트의 프로세스/스레드이고, 데이터베이스를 가진 응용프로그램 조작을 가능하게 합니다.

*논리 에이전트*는 데이터베이스 관리 프로그램에 대한 연결된 응용프로그램을 나타냅니다. 논리 에이전트에는 응용프로그램에 필요한 모든 정보와 제어 블록이 있습니다. 최대 논리 에이전트 수는 *max_logicagents* 데이터베이스 관리 프로그램 구성 매개변수에 의해 제어됩니다. 각 응용프로그램에는 하나의 논리 에이전트가 있어야 하므로, 이 매개변수는 인스턴스에 연결될 수 있는 최대 응용프로그램 수를 제어합니다.

*작업자 에이전트*는 응용프로그램 요청을 수행하지만, 특정 응용프로그램에 영구적으로 접속되지는 않습니다. 작업자 에이전트에는 응용프로그램에서 요청한 조치를 데이터베이스 관리 프로그램 내에서 완료하는 데 필요한 모든 정보와 제어 블록이 있습니다.

사용 중인 조정자 에이전트, *서브에이전트*, *비활동 에이전트* 및 *유휴 에이전트*와 같은 네 가지 유형의 작업자 에이전트가 있습니다.

유휴 에이전트는 작업자 에이전트의 가장 간단한 양식으로, 논리 에이전트에 연결되어 있지 않으며, 아웃바운드 연결도 없고, 지역 데이터베이스 연결이나 인스턴스 접속도 수반하지 않습니다.

비활동 에이전트는 사용 중인 트랜잭션에 없는 작업자 에이전트로, 논리 에이전트에 연결되어 있지 않으며, 아웃바운드 연결도 없고, 지역 데이터베이스 연결이나 인스턴스 접속도 수반하지 않습니다. 비활동 에이전트는 자유로이 다른 논리 에이전트에 연결되어 그 논리 에이전트에서 제시하는 응용프로그램에 서비스를 제공할 수 있습니다.

클라이언트 응용프로그램의 각 프로세스/스레드에는 데이터베이스에서 작동되는 하나의 *사용 중인 조정자 에이전트*가 있습니다. 일단 조정자(coordinator) 에이전트가 작성되면 자신의 응용프로그램 대신 모든 데이터베이스 요청을 수행하고 프로세스간 통신(IPC) 또는 원격 통신 프로토콜을 사용하여 다른 에이전트와 통신합니다. 각 에이전트는 자체의 고유한 개인용 메모리를 사용하여 작동하며 다른 에이전트와 함께 버퍼 풀과 같은 데이터베이스 관리 프로그램 및 데이터베이스 전역 자원을 공유합니다. 트랜잭션이 완료되면 사용 중인 조정자 에이전트는 논리 에이전트로부터 분리되어 비활동 에이전트가 될 수 있습니다.

파티션된 데이터베이스 환경 및 파티션 내 병렬 처리가 사용 가능한 환경에서 조정자 에이전트는 데이터베이스 요청을 *서브에이전트*로 분산하고, 이러한 에이전트는 응용프로그램에 대한 요청을 수행합니다. 조정자(coordinator) 에이전트가 작성되면 데이터베이스에서 요청을 수행하는 서브에이전트를 조정하여 해당 응용프로그램 대신에 모든 데이터베이스 요청을 처리합니다.

클라이언트가 데이터베이스로부터 연결 해제되거나 인스턴스와의 접속이 해제되면 조정 에이전트는 다음과 같이 됩니다.

- 사용 중인 에이전트. 다른 논리 에이전트가 기다리고 있으면, 작업자 에이전트는 사용 중인 조정자 에이전트가 됩니다.
- 기다리고 있는 다른 논리 에이전트가 없고 최대 풀 에이전트 수에 도달하지 않은 경우, 해제되어 유휴 상태로 표시됩니다.
- 기다리고 있는 다른 논리 에이전트가 없고 최대 풀 에이전트 수에 도달한 경우, 종료되고 해당되는 저장영역은 해제됩니다.

응용프로그램 대신 작업을 수행하지 않고 지정되기를 기다리는 에이전트는 유휴 에이전트로 간주되어 *에이전트 풀*에 상주합니다. 이들 에이전트는 클라이언트 프로그램 대신에 조작하는 조정자 에이전트의 요청에서 사용 가능하거나 기존의 조정자(coordinator) 에이전트를 대신하여 조작하는 서브에이전트에서 사용 가능합니다. 사용 가능한 에이전트 수는 데이터베이스 관리 프로그램 구성 매개변수인 *maxagents* 및 *num_poolagents*에 따라 결정됩니다.

에이전트 풀의 에이전트(*num_poolagents*)는 다음에 대해 조정자 에이전트로 다시 사용됩니다.

- 원격 TCP/IP 기반 응용프로그램
- UNIX 기반 운영 체제의 지역 응용프로그램
- Windows NT 및 OS/2 운영 체제의 지역 및 원격 응용프로그램

그렇지 않으면 원격 응용프로그램은 항상 새 에이전트를 작성합니다.

에이전트를 요구할 때 유휴 에이전트가 없는 경우, 새 에이전트가 동적으로 작성되어야 합니다. 새 에이전트를 생성하는 것은 일정량의 오버헤드가 되므로, 결과적으로 클라이언트를 활성화시킬 수 있는 유휴 상태의 에이전트가 있으면, CONNECT 및 ATTACH의 성능이 향상됨을 알 수 있습니다.

서브에이전트가 응용프로그램 대신에 작업하는 경우, 해당 응용프로그램과 *연관된* 것으로 간주됩니다. 서브에이전트가 지정된 작업을 완료한 후 에이전트 풀에 배치될 수는 있지만 여전히 원래의 응용프로그램과 연관된 상태로 남아 있습니다. 응용프로그램이 추가 작업을 요청하면 데이터베이스 관리 프로그램은 응용프로그램이 작업할 에이전트를 찾으면서 우선 유휴 풀에 연관된 에이전트가 있는지 점검합니다.

*max_logicagents*에 의해 정의된 논리 에이전트 수를 사용하여 연결된 응용프로그램 수를 별도로 제어할 수 있는 기능과, *max_coordagents*에 의해 정의된 사용 중인 조정자 에이전트 수를 사용하여 처리될 수 있는 응용프로그램 요청 수를 제어하는 기능으로, 데이터베이스에서 처리되는 워크로드에 융통성이 생깁니다. 연결된 응용프로그램 수와 처리될 수 있는 응용프로그램 요청 수 사이의 일 대 일 관계는 응용프로그램이 데이터베이스에서 작동될 일반적인 방법입니다. 그러나 연결된 응

용프로그램 수와 처리될 수 있는 응용프로그램 요청 수 사이의 다 대 일 관계를 요구하는 작업 환경에 있을 수도 있습니다.

데이터베이스 전역 자원 오버헤드가 활동 조정자 에이전트와 연관되므로, 이러한 에이전트 수가 커지면 이는 사용할 수 있는 데이터베이스 전역 자원의 상한에 도달할 경우가 많아진다는 것을 의미합니다. 사용 가능한 데이터베이스 전역 자료 상한에 도달하지 않도록 활동 중인 조정자 에이전트보다 연결된 응용프로그램이 많게 할 수도 있습니다. *max_logicagents* 값을 *max_coordagents* 값보다 크게 설정하면 데이터베이스 작업에 집중할 수 있습니다.

XA 트랜잭션 지원 집중기(concentrator)로 DB2 Connect를 사용하는 방법과 예에 대한 자세한 내용은 *DB2 Connect 사용자 안내서*를 참조하십시오.

원격 시스템에 연결하기 위해 DB2 Connect가 필요한 환경에서 작업할 경우, *아웃바운드 연결 풀*이 있습니다. 이 연결 풀은 호스트에 대한 연결 시간(첫 번째 연결 다음의)을 단축시킵니다. 호스트로부터 연결 해제해야 할 경우, DB2 Connect는 인바운드 연결을 삭제하나 풀 내의 호스트에 대한 아웃바운드 연결을 유지합니다. 호스트에 대한 새 연결 요청이 있는 경우, DB2 Connect는 풀에 있는 기존의 아웃바운드 연결을 다시 사용합니다(사용 가능한 경우).

주: 연결 풀을 사용할 경우, DB2 Connect는 인바운드 TCP/IP와 아웃바운드 TCP/IP 및 SNA 연결로 제한됩니다. SNA에 대한 작업을 할 경우, 풀에 놓일 연결의 보안 유형은 NONE이어야 합니다.

연결 풀에서 사용 중인 에이전트는 연결 해제 이후에 아웃바운드 연결을 닫지 않으나, 원격 호스트에 대해 사용 중인 연결과 함께 에이전트 풀에 놓입니다. 이러한 유형의 에이전트를 *비 사용 DRDA* 에이전트라고 합니다. 비 사용 DRDA 에이전트 풀은 아웃바운드 연결 풀과 동의어입니다. 『DRDA』는 『Distributed Relational Database Architecture』를 나타냅니다.

네 가지의 다른 사용법과 워크로드 요구사항에 따라 다음 예들을 고려할 수 있습니다.

1. 첫 번째 예에서 동시에 평균 40명의 사용자가 DB2 Connect를 통해 원격 호스트 데이터베이스에 연결합니다. 최대 동시 연결 수는 50에 달하기도 하지만, 절대 55를 초과하지는 않습니다. 트랜잭션은 지속기간이 짧으며 연결 및 연결 해제가 빈번합니다.

   이러한 조건에서 시스템 관리자는 동시에 DB2 Connect를 통해 연결을 시도하는 최대 사용자의 수가 55임을 알기 때문에 *max_coordagents*를 55로 구성해야 합니다. *num_poolagents*(에이전트 풀의 크기)는 40으로 설정해야 하는데, 이는 평균 숫자의 사용자가 연결되어 있거나 연결을 시도하기 때문입니다. 풀 크기를 이렇게 설정하면 워크로드가 최고에도 도달할 경우를 제외하고는 새 연결을 설정하지 않고도 기존의 원격 데이터베이스 연결로 모든 인바운드 클라이언트를 충족시킬 수 있습니다.

2. 두 번째 예에서 워크로드는 1 000 인바운드 클라이언트로 약간 높습니다. 연결 기간은 역시 단기간입니다. 시스템 관리자는 그 이상의 동시 연결을 원하지 않습니다. 그러므로 시스템 관리자는 *max_coordagents* 및 *num_poolagents*를 모두 1 000으로 설정합니다. 이는 동시에 원격데이터베이스에 연결될 수 있는 최대 인바운드 클라이언트 수가 1 000임을 의미합니다. 모든 클라이언트가 연결 해제되면 풀은 새 인바운드 클라이언트에 연결되기를 기다리는 정확히 1 000개의 연결된 에이전트를 포함하게 됩니다.

3. 세 번째 예는 단 하나의 원격 데이터베이스에 대해 DB2 Connect를 통한 하나의 응용프로그램 연결을 포함합니다. 응용프로그램은 장시간 동안 연결된 채 남아 있습니다. 이 시나리오에서 최상의 에이전트 및 연결 풀 구성은 *max_coordagents*를 1로 구성하는 것인데, 이는 하나의 클라이언트만이 연결되는 것을 알고 있기 때문입니다. 이 경우, 원격 호스트에 대한 빈번한 연결 및 연결 해제가 없으므로 *num_poolagents*를 0으로 설정할 수 있습니다. 원격 데이터베이스에 대해 사용 중인 연결을 가진 에이전트는 풀에 보유되지 않으므로, *num_poolagents*를 0으로 설정하면 연결을 풀에 넣는 작업이 효율적으로 작동 불가능하게 됩니다. 연결하는 모든 새 인바운드 클라이언트의 경우, 새 에이전트가 작성되며 새로 설정된 원격 연결이 에이전트에 제공됩니다.

4. 네 번째 예는 앞의 세 워크로드 시나리오를 기본으로 변형한 경우입니다. 이 예에서 시스템 관리자는 원격 데이터베이스에 대한 동시 액세스의 수를 100으로 제한하고자 합니다. 그러므로 *max_coordagents*는 100으로 설정되고, 연결 성능을 최대화하기 위해 *num_poolagents*는 100으로 설정됩니다. 그러나 나중

에 DB2 Connect가 설치된 시스템에서의 워크로드를 모니터하기 위해 지역적 연결이 필요할 수 있습니다. 한 번에 6개 이상의 동시 모니터 스냅샷이 발생하지는 않을 것으로 예상되므로 *max_coordagents*는 105로 설정됩니다. 이 새 구성 값은 최대 동시 응용프로그램 수가 이전의 상한인 100을 넘을 수 있도록 하여 간혹 발생하는 모니터 스냅샷이나 인스턴스 접속을 허용할 수 있게 합니다.

파티션된 데이터베이스 환경 및 파티션 내 병렬 처리가 작동 가능한 환경에서 각 파티션(즉, 각 데이터베이스 서버 또는 노드)에는 서브에이전트를 끌어낼 수 있는 자체의 에이전트 풀이 있습니다. 이 풀이 있으므로, 필요하거나 작업을 완료할 때마다 서브에이전트를 작성하고 폐기할 필요가 없습니다. 서브에이전트는 풀에서 연관된 에이전트로 남아 있을 수 있으며, 연관된 응용프로그램의 새 요청이 있을 때 데이터베이스 관리 프로그램이 이를 사용합니다.

다음 데이터베이스 관리 프로그램 구성 매개변수는 데이터베이스 에이전트 수에 영향을 미칩니다.

- 462 페이지의 『에이전트의 최대수(maxagents)』. 작업자 에이전트 수가 이 값에 도달하면 새 에이전트를 요구하는 모든 후속 요청은 에이전트 수가 이 값 아래로 떨어질 때까지 거부됩니다. 이 값은 모든 응용프로그램에서 작동하는 조정자 에이전트, 서브에이전트, 비활동 에이전트 및 유휴 에이전트를 포함하여 전체 에이전트 수에 적용됩니다.
- 467 페이지의 『에이전트 풀 크기(num_poolagents)』. 에이전트 풀의 비활동 에이전트, 유휴 에이전트 및 연관된 서브에이전트 수는 이 값을 초과할 수 없습니다.
- 469 페이지의 『풀의 초기 에이전트의 수(num_initagents)』. 데이터베이스 관리 프로그램이 시작되면 작업자 에이전트 풀이 이 값을 기초로 작성됩니다. 이렇게 하면 초기 조회 성능을 향상시킵니다. 작업자 에이전트는 모두 유휴 에이전트로 시작합니다.
- 466 페이지의 『논리 에이전트의 최대 수(max_logicagents)』. 최대 논리 에이전트 수. 각 응용프로그램에는 하나의 논리 에이전트가 있어야 하므로, 이 매개변수는 인스턴스에 연결될 수 있는 최대 응용프로그램 수를 제어합니다.

- 465 페이지의『조정 에이전트의 최대 수(max_coordagents)』. 파티션된 데이터베이스 환경 및 파티션 내 병렬 처리가 작동 가능한 환경에서 이 값은 조정 에이전트의 수를 제한합니다.
- 464 페이지의『동시 에이전트의 최대수(maxcagents)』. 이 값은 데이터베이스 관리 프로그램에 의해 허용되는 토큰 수를 제어합니다. 클라이언트가 데이터베이스에 연결될 때 발생하는 각 데이터베이스 트랜잭션(작업 단위)의 경우, 조정 에이전트는 데이터베이스 관리 프로그램으로부터 트랜잭션(처리 토큰이라고도 함)을 처리할 수 있는 사용권한을 확보해야 합니다. 데이터베이스 관리 프로그램은 처리 토큰을 가진 에이전트만이 데이터베이스에 대해 작업 단위를 실행할 수 있도록 허용합니다. 토큰을 사용할 수가 없는 경우, 사용할 수 있을 때까지 기다린 다음 요구된 작업 단위(UOW)가 처리됩니다.

  이 매개변수는 최대 사용 요구사항이 시스템 자원(메모리, CPU 및 디스크)을 초과하는 환경에서 유용할 수 있습니다. 그러한 환경에서 최대 로드는 페이징과 같은 원인으로 인해 과도한 성능 감소를 초래할 수도 있습니다. 이 매개변수가 동시성과 대기 시간에 영향을 줄 수 있더라도 이 매개변수를 사용하여 로드를 제어하고 성능 저하를 피할 수 있습니다.

파티션된 데이터베이스 환경 및 파티션 내 병렬 처리가 작동 가능한 환경에서 성능 및 메모리 비용에 미치는 영향은 에이전트 풀을 조정하는 방법과 밀접한 관계가 있습니다.

- 에이전트 풀 *크기*(*num_poolagents*)에서 데이터베이스 관리 프로그램 구성 매개변수는 파티션(즉, 노드)의 응용프로그램과 연관될 수 있는 서브에이전트 수에 영향을 미칩니다. 풀 *크기*가 너무 작은 경우(그리고 풀이 가득 찬 경우), 서브에이전트는 작업한 응용프로그램에서 스스로 연관성을 없애고 종료합니다. 이렇게 하면 성능이 저하되는데, 서브에이전트가 계속 작성되어 응용프로그램에 계속 다시 연관되어야 하기 때문입니다.

  추가로, *num_poolagents* 값이 너무 작으면 응용프로그램은 풀을 연관된 서브에이전트로 채웁니다. 그러므로 다른 응용프로그램이 새 서브에이전트를 요구하고 연관된 에이전트 풀에 서브에이전트가 없는 경우, 다른 응용프로그램의 에이전트 풀로부터 서브에이전트를『도용할』것입니다. 이렇게 되면 비용이 많이 들고 성능이 저하됩니다.

- 위의 상황에는 주어진 시간에 너무 많은 에이전트를 사용 중이도록 하는 자원 비용에 대해 적당한 조치가 필요합니다.

  예를 들어, *num_poolagents* 값이 너무 큰 경우, 연관된 서브에이전트는 오랜 시간 동안 풀에서 사용되지 않은 상태를 유지합니다. 이 서브에이전트는 다른 타스크에 대해 사용 가능하지 않은 데이터베이스 관리 프로그램 자원을 사용합니다.

데이터베이스 에이전트 외에, 다음을 비롯하여 데이터베이스 관리 프로그램에 의해 자신의 프로세스(또는 스레드)로 수행되는 다른 비동기 활동이 있습니다.

- 데이터베이스 I/O 서버(또는 I/O 프리페처)(292 페이지의 『버퍼 풀로 데이터 프리페치』 참조)
- 데이터베이스 비동기 페이지 정리자(282 페이지의 『데이터베이스 버퍼 풀 관리』 참조)
- 데이터베이스 로그 프로그램
- 데이터베이스 교착 상태(deadlock) 검출기
- 이벤트 모니터
- 통신 및 IPC 리스너
- 테이블 공간 컨테이너 재조정자

여러 DB2 프로세스 식별에 대한 자세한 내용은 *문제점 해결 안내서*를 참조하십시오.

데이터베이스 시스템 모니터 사용

DB2 데이터베이스 관리 프로그램은 이 프로그램의 조작 및 성능, 프로그램을 사용하는 응용프로그램에 대한 데이터를 유지보수합니다. 데이터베이스 관리 프로그램이 수행할 때 이 데이터는 유지보수되며 중요한 성능과 문제점 해결 정보를 제공할 수 있습니다. 예를 들어, 다음과 같은 정보를 찾을 수 있습니다.

- 데이터베이스에 연결된 응용프로그램 수, 응용프로그램 상태, 각 응용프로그램이 실행 중인 SQL문이 있는 경우, 이러한 명령문 정보
- 데이터베이스 관리 프로그램과 데이터베이스가 얼마나 구성이 잘 되었는지 보여주고, 조정하는 데 도움이 되는 정보

- 교착 상태가 지정된 데이터베이스에서 발생하는 시기, 관련된 응용프로그램, 경합 상태인 잠금
- 응용프로그램이나 데이터베이스에 의해 보유된 잠금 목록. 응용프로그램이 잠금 대기 중이어서 진행할 수 없는 경우, 잠금에는 정보를 보유한 응용프로그램을 포함한 추가 정보가 있습니다.

이 데이터를 수집하면 DB2 조작에서 오버헤드가 일어나므로, 수집된 정보를 제어하는 데 **모니터 스위치**를 사용할 수 있습니다. 모니터 스위치를 명시적으로 보려면 UPDATE MONITOR SWITCHES 명령 또는 sqlmon() API를 사용하십시오. (SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있어야 합니다.)

데이터베이스 관리 프로그램에 의해 유지보수된 데이터에 액세스하는 방법에는 두 가지가 있습니다.

- **스냅샷 찍기**

  스냅샷을 찍는 세 가지 방법이 있습니다. 명령행에서 GET SNAPSHOT 명령을 사용하거나, OS/2에서 제어 센터 또는 그래픽 인터페이스용 Windows 기반 운영 체제를 사용하거나, sqlmonss() API 호출을 사용하여 자신의 응용프로그램을 작성할 수 있습니다.

  DB2 폴더로부터 또는 `db2cc` 명령으로부터 사용 가능한 제어 센터는 스냅샷을 찍어 일정한 간격으로 샘플 모니터 데이터를 수집하는 성능 모니터 도구를 제공합니다. 이 그래픽 인터페이스는 세부사항과 요약사항에서 모두 스냅샷 데이터의 그래프 또는 텍스트 뷰를 제공합니다. 데이터베이스 모니터에서 리턴된 데이터 요소를 사용하여 성능 변수를 정의할 수 있습니다.

  제어 센터의 스냅샷 모니터 도구를 사용하여 성능 변수에 대한 임계값을 지정함으로써 예외 조건을 정의할 수 있습니다. 임계값에 도달하면 그 다음에 발생할 조치(창 또는 청취 가능한 알람을 통한 통지, 스크립트 또는 프로그램의 실행과 같이)를 사전 정의할 수 있습니다.

  제어 센터로부터 스냅샷을 찍을 경우, 해당 오브젝트 또는 하위 오브젝트에서 스냅샷 모니터링을 수행하는 동안에는 데이터베이스 오브젝트(예: 인스턴스 또는 데이터베이스)를 변경하거나 삭제하는 조치는 수행할 수 없습니다. (이 외에도, 파티션된 데이터베이스 시스템을 모니터할 때 파티션된 데이터베이스 오브젝트의 뷰를 새로 고칠 수 없습니다.) 예를 들어, 데이터베이스 A의 인스턴스를 제

거하려할 때 데이터베이스 A를 모니터할 수 없습니다. 그러나 인스턴스만을 모니터할 때에는 데이터베이스 A를 변경할 수 있습니다.

인스턴스(해당되는 하위 오브젝트를 포함하여)의 모든 모니터링을 중지시키려면 인스턴스에 대한 팝업 메뉴에서 **모든 모니터링 중지**를 선택하십시오. 항상 인스턴스에서 모니터링을 중지시켜야 합니다. 왜냐하면, 이로 인하여 성능 모니터에서 보유한 모든 잠금이 해제되기 때문입니다.

- **이벤트 모니터 사용**

  트랜잭션 종료, 명령문 종료 또는 교착 상태 검출과 같은 특정 이벤트가 발생한 후에 이벤트 모니터는 시스템 모니터 정보를 캡쳐합니다. 이 정보는 파일 또는 Named Pipe에 기록될 수 있습니다.

  이벤트 모니터를 사용하려면 다음을 수행하십시오.

  1. 제어 센터 또는 SQL문 CREATE EVENT MONITOR를 사용하여 해당 정의를 작성하십시오. 이 명령문은 데이터베이스 시스템 카탈로그에 정의를 저장합니다.
  2. 제어 센터 또는 SQL문을 사용하여 이벤트 모니터를 활성화하십시오.

     ```
     SET EVENT MONITOR evname STATE 1
     ```

     Named Pipe에 기록하려면 이벤트 모니터를 활성화하기 전에 Named Pipe로부터 읽는 응용프로그램을 시작하십시오. 사용자의 응용프로그램을 사용하거나 **db2evmon**을 사용하여 이를 수행할 수 있습니다. 이벤트 모니터가 사용 중이고 파이프에 기록하는 이벤트가 시작되면 **db2evmon**은 이벤트가 생성되어 표준 출력에 기록될 때 이벤트를 읽게 됩니다.
  3. 추적을 읽으십시오. 파일 이벤트 모니터를 사용하는 경우, 다음 방식으로 작성하는 2진 추적을 볼 수 있습니다.
     - **db2evmon** 도구를 사용하여 표준 출력으로 추적을 형식화하십시오.
     - 제어 센터(Windwos 기반 운영 체제 또는 OS/2 시스템)에서 **이벤트 분석기** 아이콘을 눌러 그래픽 인터페이스를 사용해서 추적을 보고 키워드를 검색하며 필요하지 않은 데이터를 필터하십시오.

       주: 모니터하는 데이터베이스 시스템이 제어 센터와 동일한 머신에서 수행 중이면 추적을 보기 전에 이벤트 모니터 파일을 제어 센터와 동

일한 머신으로 복사해야 합니다. 다른 방법으로는 파일을 두 머신에 모두 액세스할 수 있는 공유 파일 시스템에 두는 방법이 있습니다.

시스템 데이터베이스 모니터 및 이벤트 모니터에 대한 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

메모리 확장

머신에는 최대 가상 주소 지정 가능 메모리 양보다 더 많은 실제 주소 지정 가능 메모리가 있을 수 있습니다. 예를 들어, 대부분의 플랫폼에서 가상 주소 지정 가능 메모리는 보통 2GB - 4GB 사이입니다. 가상 주소 지정 가능 메모리를 초과하는 추가 실제 주소 지정 가능 메모리는 *확장 저장영역 캐쉬*로 구성할 수 있습니다. 정의된 모든 버퍼 풀은 이런 확장 저장영역 캐쉬를 사용할 수 있으며, 이 캐쉬는 데이터베이스 관리 프로그램의 성능을 향상시켜야 합니다. 확장 저장영역 캐쉬는 메모리 세그먼트에 의해 정의됩니다.

실제 주소 지정 가능 메모리의 일부를 확장 저장영역 캐쉬로 사용하려는 경우에는, 이 메모리가 머신에서 더 이상 JFS 캐쉬나 프로세스 개인 주소 공간과 같은 다른 목적에는 사용될 수 없다는 것에 주의해야 합니다. 추가 실제 주소 지정 가능 메모리를 확장 저장영역 캐쉬에 지정하면 시스템 페이징 성능을 높일 수 있습니다.

DB2는 버퍼 풀이 있는 머신에서 주소 지정 가능 메모리를 사용합니다. (자세한 내용은 282 페이지의 『데이터베이스 버퍼 풀 관리』를 참조하십시오.) 버퍼 풀이 사용한 확장 저장영역 캐쉬는 두 번째 캐쉬 레벨로 사용됩니다. (첫 번째 캐쉬 레벨을 실행하는 버퍼 풀이 있는 캐쉬 레벨을 말합니다.) 이상적으로 버퍼 풀은 가장 자주 액세스되는 데이터를 보유하는 반면에, 확장 저장영역 캐쉬는 액세스되지만 자주 액세스되지는 않는 데이터를 보유할 수 있습니다.

DB2_AWE 레지스트리 변수를 사용하여 Windows 2000 AWE(Address Windowing Extensions) 버퍼 풀을 할당할 때 확장 저장영역 캐쉬를 사용할 수 없습니다.

다음 데이터베이스 구성 매개변수는 확장 저장영역에 사용 가능한 메모리 양 및 크기에 영향을 미칩니다.

- *num_estore_segs*는 확장 저장영역 메모리 세그먼트의 수를 정의합니다. 이 구성 매개변수의 기본값은 0이며, 이 값은 확장 저장영역 캐쉬가 존재하지 않는다는 의미입니다. (자세한 내용은 454 페이지의 『확장 저장영역 메모리 세그먼트의 수(num_estore_segs)』를 참조하십시오.)
- *estore_seg_sz*는 각 확장 메모리 세그먼트의 *크기*를 정의합니다. 이 *크기*는 사용 중인 확장 저장영역 캐쉬가 있는 플랫폼에 의해 제한됩니다. (자세한 내용은 454 페이지의 『확장 저장 메모리 세그먼트 *크기*(estore_seg_sz)』를 참조하십시오.)

확장 저장영역 캐쉬는 버퍼 풀에 대한 확장이므로, 하나 이상의 특정 버퍼 풀과 항상 연관되어 있어야 합니다. 그러므로 캐쉬가 작성되면 캐쉬를 사용할 수 있는 버퍼 풀을 선언해야 합니다. CREATE 및 ALTER BUFFERPOOL문은 캐쉬 사용을 제어하는 NOT EXTENDED STORAGE와 EXTENDED STORAGE 속성을 갖습니다. 기본적으로 IBMDEFAULTBP 또는 새로 작성된 모든 버퍼 풀은 확장 저장영역을 사용하지 않습니다.

주: 다른 페이지 *크기*로 정의된 버퍼 풀을 사용할 수 있습니다. 이러한 버퍼 풀의 일부 또는 전체를 정의하여 확장 저장영역을 사용할 수 있습니다. 확장 저장영역 지원과 함께 사용되는 페이지 *크기*는 정의한 것 중 가장 큽니다.

데이터베이스 관리 프로그램은 확장 저장영역 캐쉬에 상주하는 데이터를 직접 조작할 수 없습니다. 그러나 디스크 저장영역에서 보다 훨씬 빠른 속도로 확장 저장영역 캐쉬에서 버퍼 풀로 데이터를 전송할 수 있습니다.

확장 저장영역 캐쉬의 페이지로부터 데이터 행이 필요할 경우, 전체 페이지가 해당 버퍼 풀로 읽어들입니다.

버퍼 풀 및 버퍼 풀과 연관된 확장 저장영역 캐쉬가 정의되면 데이터베이스가 활성화되거나 먼저 연결되는 경우에 할당됩니다.

제9장 조정자(governor) 사용

조정자(governor)를 사용하여 데이터베이스에 대해 수행되는 응용프로그램의 작동을 모니터하고 변경할 수 있습니다.

조정자(governor)는 다음의 두 부분으로 이루어집니다.
- 프론트엔드 유틸리티
- 디먼

조정자(governor)를 시작할 때 조정자(governor) 프론트엔드 유틸리티로부터 시작 명령을 발행하면, 조정자(governor) 디먼이 시작됩니다. 기본적으로 디먼은 파티션된 데이터베이스의 모든 파티션에서 시작되지만, 프론트엔드 유틸리티를 사용하여 특정 파티션에서 하나의 디먼을 시작하여 원래 데이터베이스 파티션에 대한 활동을 모니터할 수도 있습니다. 또는 디먼은 하나의 파티션된 데이터베이스에서의 활동을 모니터할 수 있습니다. 자세한 내용은 324 페이지의 『조정자(governor) 시작 및 중지』를 참조하십시오.

각 조정자(governor) 디먼은 데이터베이스에 대해 수행 중인 응용프로그램에 대한 통계를 수집합니다. 그런 다음 특정 데이터베이스에 적용되는 조정자(governor) 구성 파일에 지정된 규칙에 대한 이들 통계를 점검합니다(자세한 내용은 327 페이지의 『조정자(governor) 구성 파일 작성』 참조). 그리고 조정자(governor)는 이 규칙에 따라 작동합니다. 예를 들어, 규칙은 응용프로그램이 자원을 너무 많이 사용하고 있음을 지정해줍니다. 이 경우, 조정자(governor)는 사용자가 조정자(governor) 구성 파일에 지정한 지침에 따라 응용프로그램의 우선순위를 변경하거나 데이터베이스로부터 벗어나도록 강제합니다.

규칙과 연관된 조치가 응용프로그램의 우선순위를 변경하는 것이면 조정자(governor)는 자원 위반을 검출한 데이터베이스 파티션에서 에이전트의 우선순위를 변경합니다. 규칙과 연관된 조치가 응용프로그램을 강제하는 것이면 자원 위반을 검출한 조정자(governor)가 응용프로그램의 조정자 노드 또는 파티션된 데이터베이스 환경에서 수행 중이더라도 응용프로그램은 강제(force)됩니다.

© Copyright IBM Corp. 1993, 2001

또한 조정자(governor)는 취하는 모든 조치를 로그합니다. 로그 파일을 조회하여 조정자(governor)가 취한 조치를 검토할 수 있습니다. 자세한 내용은 337 페이지의 『조정자(governor) 로그 파일』 및 338 페이지의 『조정자(governor) 로그 파일 조회』를 참조하십시오.

조정자(governor) 시작 및 중지

db2gov 조정자(governor) 프론트엔드 유틸리티를 사용하여 조정자(governor)를 시작하거나 중지합니다(모든 데이터베이스 파티션 또는 하나의 데이터베이스 파티션에서). 유틸리티를 사용하려면 SYSADM 또는 SYSCTRL 권한이 필요합니다.

db2gov의 구문은 다음과 같습니다.

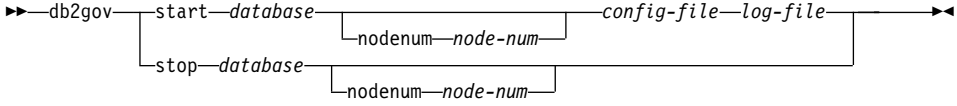


그림 28. db2gov용 구문

매개변수는 다음과 같습니다.

start **database**

조정자(governor) 디먼을 시작하여 지정된 데이터베이스를 모니터합니다. *데이터베이스*에 대해 데이터베이스 이름 또는 데이터베이스 별명을 지정할 수 있습니다.

지정한 데이터베이스 이름은 조정자(governor) 구성 파일에 지정된 이름과 같아야 합니다. 조정자(governor)는 이 두 이름을 점검하여 올바른 구성 파일이 사용되고 있는지 확인합니다. 하나의 별명으로 프론트엔드 유틸리티가 시작되고 조정자(governor) 구성 파일에 다른 별명이 들어 있을 경우 오류가 보고되는데, 조정자(governor)는 이름이 동일한 데이터베이스의 별명인지 판별할 수 없기 때문입니다.

파티션된 데이터베이스 환경에 있는 경우, 모든 파티션에서 조정자(governor)를 시작할 때, 프론트엔드 유틸리티는 먼저 오류가 없는 구성 파일을 점검합니다. 그러면 노드 구성 파일을 읽고 명령을 각 데이터베이스 파티션으로 전송하여 시작 옵션(차례로 각 데이터베이스 파티션에서 디먼을 시작하는)으로 각 데이터베이스 파티션에서 조정자(governor) 프론트엔드 유틸리티를 시작합니다.

주: 조정자(governor)는 데이터베이스 레벨에서 모니터하므로, 하나의 디먼이 모니터되는 각 데이터베이스에 대해 수행됩니다. (파티션된 데이터베이스 환경에서 하나의 디먼이 각 데이터베이스 파티션에 대해 수행됩니다.) 조정자(governor)가 여러 데이터베이스에 대해 수행되고 있을 경우, 해당되는 데이터베이스 서버에서 여러 개의 디먼이 수행됩니다.

nodenum **node-num**

조정자(governor) 디먼을 시작하는 데이터베이스 파티션을 지정합니다. 이 수는 노드 구성 파일에 지정된 수와 동일합니다.

하나의 데이터베이스 파티션에서 조정자(governor)를 시작하는 경우, 프론트엔드 유틸리티는 디먼을 작성하여 조정자(governor) 구성 파일의 유효성을 확인합니다. 조정자(governor) 디먼은 다른 디먼이 이 파티션에서 아직 수행 중이 아니라는 사실을 확인합니다.

config-file

데이터베이스를 모니터할 때 사용할 구성 파일을 지정합니다.

구성 파일의 기본 위치는 `sqllib` 디렉토리입니다. 지정한 파일이 없으면 프론트엔드는 지정된 이름이 파일의 전체 이름인 것으로 가정합니다.

log-file 조정자(governor)가 로그 레코드를 기록할 파일 이름을 지정합니다. 로그 파일은 `sqllib` 디렉토리의 `log` 서브디렉토리에 저장됩니다. Windows NT에서 `log` 서브디렉토리는 인스턴스 디렉토리 밑에 있습니다. 조정자(governor)가 수행 중인 데이터베이스 파티션의 수는 로그 파일(예: `mylog.0, mylog.1, mylog.2`)에 자동으로 추가됩니다.

stop **database**

지정된 데이터베이스를 모니터하는 조정자(governor) 디먼을 중지합니다.

파티션된 데이터베이스 환경에 있는 경우, 프론트엔드 유틸리티는 노드 구성 파일을 읽고, 각 데이터베이스 노드로 명령을 전송하여 중지 매개변수로 조정자(governor) 프론트엔드 유틸리티를 호출하여 모든 데이터베이스 파티션의 조정자(governor)를 중지시킵니다. 이렇게 하면 데이터베이스 파티션마다 디먼이 중지됩니다.

nodenum **node-num**

조정자(governor) 디먼을 중지시키는 데이터베이스 파티션을 지정합니다. 이 수는 노드 구성 파일에 지정된 수와 동일합니다.

프론트엔드 유틸리티가 하나의 데이터베이스 파티션에서 조정자(governor) 디먼을 중지시키는 경우, sqllib 디렉토리의 tmp 서브디렉토리에 있는 파일을 작성, 이동시키거나 삭제하여 해당 데이터베이스 파티션의 디먼과 통신합니다. 이 파일을 삭제하거나 수정하려고 하지 마십시오.

조정자(governor) 디먼

조정자(governor) 디먼이 시작될 때에는 db2gov 프론트엔드 유틸리티를 통해(또는 가동을 통해), 루프에서 수행됩니다. 첫 번째로 수행되는 타스크는 조정자(governor) 구성 파일이 변경되었는지 아니면 아직 읽히지 않았는지 여부를 점검하는 것입니다. 이러한 조건에 해당되는 경우, 디먼은 파일의 규칙을 읽습니다. 이렇게 하여 조정자(governor) 디먼이 수행 중일 때 조정자(governor) 디먼의 작동을 변경할 수 있습니다.

그런 다음 조정자(governor) 디먼은 스냅샷 요청을 발행하여 데이터베이스에서 작업 중인 각 응용프로그램 및 에이전트에 대한 통계를 확보합니다.

주: 일부 플랫폼에서는 DB2 모니터로부터 CPU 통계를 사용할 수 없습니다. 여기에 해당되는 경우, 계정 규칙과 CPU 한계를 사용할 수 없습니다.

그런 다음 조정자(governor)는 조정자(governor) 구성 파일의 규칙에 대해 각 응용프로그램의 통계를 점검합니다. 규칙이 응용프로그램에 적용되는 경우, 조정자(governor)는 응용프로그램을 강제로 실행하고 해당 데이터베이스 파티션에서 작업하는 에이전트와 서브에이전트의 모든 에이전트 우선순위를 간접적으로 변경하는 응용프로그램의 우선순위를 변경할 수 있습니다. 또는 규칙에 의해 지정된 조치에 따라 응용프로그램에서 작업하는 에이전트 우선순위를 간접적으로 변경하는 응용프로그램 스케줄을 변경할 수 있습니다. 조정자(governor)는 취하는 모든 조치의 레코드를 로그 파일에 기록합니다.

주: *agentpri* 데이터베이스 관리 프로그램 구성 매개변수가 시스템 기본값을 제외한 값인 경우, 대체 방법으로서 조정자(governor)를 사용하여 에이전트 우선순위를 조정할 수 없습니다. (이는 Windows NT 플랫폼에는 적용되지 않습니다.)

조정자(governor)는 모든 응용프로그램 점검을 완료할 경우, 구성 파일에 지정된 간격 동안 대기합니다. 이 시간이 경과되면 조정자(governor)는 다시 실행 루프를 시작합니다.

조정자(governor)가 오류나 중지 신호를 발견한 경우, 종료되기 전에 정리(clean-up) 처리를 수행합니다. 정리 처리는 모든 응용프로그램 에이전트 우선순위를 재설정합니다(우선순위가 설정된 응용프로그램 목록을 사용하여). 그런 다음 응용프로그램에서 더 이상 작업하지 않는 모든 에이전트의 우선순위를 재설정합니다. 이 경우, 조정자(governor)가 종료된 후 에이전트가 기본값이 아닌 우선순위로 실행되지 않아도 됩니다. 오류가 발생하는 경우, 메시지는 db2diag.log 파일에 기록되어 조정자(governor)가 비정상적으로 종료된 것을 나타냅니다.

주: 조정자(governor) 디먼은 데이터베이스 응용프로그램이 아니므로, 데이터베이스로의 연결을 유지보수하지 않습니다. (그러나 인스턴스 접속은 가지고 있습니다.) 스냅샷 요청을 발행하기 때문에 조정자(governor) 디먼은 데이터베이스 관리 프로그램이 종료되는 시점을 검출할 수 있습니다.

조정자(governor) 구성 파일 작성

사용자는 조정자(governor)를 시작할 때 데이터베이스에 대해 수행 중인 응용프로그램을 조정하는 데 사용될 규칙을 포함하고 있는 구성 파일 이름을 지정합니다. 조정자(governor)는 이 규칙에 따라 작동합니다.

데이터베이스 조정에 대한 요구사항이 변경될 경우, 조정자(governor)를 중지시키지 않고 구성 파일을 편집할 수 있습니다. 각 조정자(governor) 디먼은 파일이 변경되었는지 탐지한 다음 다시 읽습니다.

각 파티션의 조정자(governor) 디먼은 동일한 구성 파일을 읽을 수 있어야 하기 때문에, 모든 데이터베이스 파티션에 걸쳐 마운트된 디렉토리에서 구성 파일을 작성해야 합니다.

구성 파일은 규칙과 주석으로 구성됩니다. 대부분의 항목은 대문자, 소문자 또는 혼합 문자로 지정될 수 있습니다. 예외는 applname으로서 대소문자가 구별됩니다.

{ } 괄호 내에 있는 주석을 구분합니다. 이 규칙에는 다음 사항이 포함됩니다.

- 규칙이 적용될 데이터베이스
- 조정자(governor)가 응용프로그램을 점검하기 위해 가동하기 전 대기하는 시간
- 응용프로그램 조정을 지정하는 규칙. 이 규칙은 규칙 절이라고 하는 작은 구성 요소로 구성됩니다.

파일의 각 규칙 다음에는 세미콜론(;)이 와야 합니다.

다음 규칙은 모니터 중인 데이터베이스를 지정하고, 디먼이 활동 루프(326 페이지의 『조정자(governor) 디먼』 참조)를 통해 작동한 후 가동하는 간격을 지정합니다. 이들 각 규칙은 파일에서 한 번만 지정됩니다.

dbname

모니터될 데이터베이스의 이름 또는 별명

account *nnn*

지정된 시간(분 단위) 동안의 각 연결에 대한 CPU 사용 통계를 수록하고 있는 계정 레코드가 작성됩니다.

주: 이 옵션은 Windows NT 환경에서 사용할 수 없습니다.

계정 간격 내에서 짧은 연결 세션이 발생할 경우, 어떠한 로그 레코드도 기록되지 않습니다. 로그 레코드가 기록될 때 레코드에는 연결에 대한 이전 로그 레코드 이후의 CPU 사용을 반영하는 CPU 통계가 있습니다. 조정자(governor)가 중지한 다음 재시작하면 CPU 사용이 두 개의 로그 레코드에 반영될 수 있습니다. 이들 로그 레코드는 해당 로그 레코드의 응용프로그램 ID를 통해 식별될 수 있습니다. 조정자(governor) 로그 파일에 대한 자세한 내용은 337 페이지의 『조정자(governor) 로그 파일』을 참조하십시오.

interval

디먼이 가동하게 되는 시간 간격(초 단위). 간격이 지정되지 않을 경우, 120 초 간격이 사용됩니다.

사용자는 다음 규칙 절을 결합하여 규칙을 형성합니다. (즉, 각 절이 아닌 전체 규칙 다음에 세미콜론이 옵니다.) 절은 규칙이 적용되는 시간, 사용할 수 있는 자원의 한계를 지정하고, 특정 사용자나 응용프로그램, 규칙에 지정된 제한을 초과할 경

우 조정자(governor)가 취할 모든 조치를 선택적으로 지정합니다. 하나의 규칙에 단 한 번 절을 지정할 수 있지만, 둘 이상의 규칙에 지정할 수 있습니다. 절은 반드시 표시된 순서대로 지정해야 합니다. 다음의 설명에서 [ ]는 선택적 절을 나타냅니다.

[desc] 규칙에 대한 텍스트 설명을 지정합니다. 설명은 작은따옴표 또는 큰따옴표로 묶어야 합니다.

[time] 얼마 동안 규칙을 평가할 것인지 지정합니다.

시간 기간은 반드시 `time hh:mm hh:mm` 형식으로 지정되어야 합니다. 예를 들면, `time 8:00 18:00`과 같습니다. 이 절이 지정되어 있지 않을 경우, 24시간 내내 규칙이 유효합니다.

[authid]

응용프로그램을 실행할 하나 이상의 권한 부여 ID(authid)를 지정합니다. 다중 권한 부여 ID는 `authid gene, michael, james`처럼 쉼표(,)로 구분해야 합니다. 규칙에 이 절이 표시되지 않을 경우, 모든 권한 부여 ID에 규칙이 적용됩니다.

[applname]

데이터베이스에 연결하는 실행 파일(또는 오브젝트 파일)의 이름을 지정합니다.

다중 응용프로그램 이름은 `applname db2bp, batch, geneprog`처럼 쉼표(,)로 구분해야 합니다. 규칙에 이 절이 표시되지 않을 경우, 모든 응용프로그램 이름에 규칙이 적용됩니다.

주:

1. 응용프로그램 이름은 대소문자가 구별됩니다.
2. 데이터베이스 관리 프로그램은 모든 응용프로그램 이름을 20자에서 자릅니다. 조정하려는 응용프로그램이 해당 응용프로그램 이름의 처음 20자를 보고 고유하게 식별할 수 있는지 확인해야 합니다. 그렇지 않으면 본의아니게 다른 응용프로그램을 조정하게 될 수 있습니다.

   조정자(governor) 구성 파일에 지정된 응용프로그램 이름은 내부 표현에 일치하도록 20자에서 잘립니다.

setlimit

점검한 조정자(governor)에 대해 하나 이상의 한계를 설정합니다. 설정할 수 있는 한계는 -1, 아니면 0보다 큰 수입니다. (예: `cpu -1 locks 1000 rowssel 10000`) 적어도 한 가지의 한계(cpu, locks, rowsread, uowtime)를 지정해야 합니다. 규칙에 의해 지정되지 않은 한계는 해당되는 특정 한계에서 제한되지 않습니다. 조정자(governor)는 다음 한계를 점검할 수 있습니다.

cpu *nnn*

응용프로그램이 소비할 수 있는 CPU 초 수를 지정합니다. -1을 지정할 경우, 조정자(governor)는 응용프로그램의 CPU 사용을 제한하지 않습니다.

주: 이 옵션은 Windows NT 환경에서 사용할 수 없습니다.

locks *nnn*

응용프로그램이 보유할 수 있는 잠금 수를 지정합니다. -1을 지정할 경우, 조정자(governor)는 응용프로그램이 보유할 잠금 수를 제한하지 않습니다.

rowssel *nnn*

응용프로그램으로 리턴되는 행 수를 지정합니다. 이 값은 조정자 노드에서만 0이 아닌 값이 됩니다. -1을 지정할 경우, 조정자(governor)는 선택될 수 있는 행 수를 제한하지 않습니다.

uowtime *nnn*

작업 단위(UOW)가 처음으로 활성화된 시간 이후의 경과 시간을 초 수로 지정합니다. -1을 지정할 경우, 경과 시간을 제한되지 않습니다.

주: `sqlmon`(데이터베이스 시스템 모니터 스위치) API를 사용하여 작업 단위 스위치의 비활성화할 경우, 이것은 조정자(governor)가 작업 단위 경과 시간에 따라 응용프로그램을 조정하는 기능에 영향을 줍니다. 조정자(governor)는 모니터를 사용하여 시스템에 대한 정보를 수집합니다. 데이터베이스 관리 프로그램 구성 파일에서 스위치를 끌 경우, 구성 파일은 전체 인스

턴스에 대해 꺼지고 조정자(governor)는 더 이상 이 내용은 수신하지 않습니다. 자세한 내용은 549 페이지의 『데이터베이스 시스템 모니터 매개변수』를 참조하십시오.

idle *nnn*

지정된 조치가 취해지기 전에 연결에 허용되는 대기 시간을 초 수로 지정합니다. -1을 지정할 경우, 연결 대기 시간은 제한되지 않습니다.

rowsread *nnn*

응용프로그램이 선택할 수 있는 행 수를 지정합니다. -1을 지정할 경우, 응용프로그램이 선택할 수 있는 행 수에는 제한이 없습니다.

주: 이 제한은 rowssel의 경우와는 다릅니다. 차이점은 rowsread는 결과 세트를 리턴하기 위해 읽어야 할 행 수를 나타낸다는 점입니다. 행 읽기 수는 엔진에 의한 카탈로그 테이블의 읽기를 포함하며 색인이 사용되면 사라지게 됩니다.

[action]

하나 이상의 제한을 초과할 경우 취할 조치를 지정합니다. 다음 조치를 지정할 수 있습니다.

주: 한계를 초과하고 조치 절이 지정되지 않을 경우, 조정자(governor)는 응용프로그램에 대해 작동 중인 에이전트의 우선순위를 10씩 낮춥니다.

priority *nnn*

응용프로그램에 대해 작동 중인 에이전트의 우선순위를 변경하도록 지정합니다. 유효한 값은 -20에서 +20까지입니다.

이 매개변수가 유효하려면, 다음 조건이 충족되어야 합니다.

- UNIX 기반 플랫폼에서 *agentpri* 데이터베이스 관리 프로그램 매개변수는 기본값으로 설정되어야 하고, 그렇지 않으면 우선순위 절을 겹쳐씁니다.
- OS/2 및 Windows NT 플랫폼에서 *agentpri* 데이터베이스 관리 프로그램 매개변수와 *우선순위* 조치는 같이 사용됩니다.

force 응용프로그램을 처리하고 있는 에이전트를 강제(force)하도록 지정합니다. (FORCE APPLICATION을 발행하여 조정자 에이전트를 종료하십시오.)

schedule [class]

스케줄링은 모든 응용프로그램에 대해 균형을 유지보수하면서 평균 응답 시간을 최소화하려는 목적으로 응용프로그램에서 작동하고 있는 에이전트의 우선순위를 높입니다.

조정자(governor)는 응용프로그램에서 작업하는 에이전트에 대한 우선순위를 설정하여 해당 스케줄을 시행하며, 이 스케줄에서는 DB2 내부 조회 컴파일러로부터 조회 비용 추정치를 사용합니다. 클래스 옵션이 지정된 경우, 규칙에서 선택된 모든 응용프로그램은 응용프로그램간에만 스케줄됩니다. 이 옵션이 지정되지 않은 경우, 조정자(governor)는 각 클래스 안에서 수행된 스케줄링과 함께 하나 이상의 클래스를 사용합니다.

응용프로그램은 각 클래스에서 다음에 기초하여 우선순위가 산정됩니다.

- 클래스 내의 응용프로그램이 보유하고 있는 잠금 수(잠금으로 인해 다른 많은 응용프로그램을 보유하고 있는 응용프로그램에 높은 우선순위가 부여됩니다.)
- 응용프로그램의 유효 기간(오랫동안 시스템에 있는 응용프로그램에 높은 우선순위가 부여됩니다.)
- 응용프로그램의 나머지 수행 시간 추정치(완료 시간에 가까운 응용프로그램이 높은 우선순위를 갖습니다.)

스케줄 규칙이 적용되지 않는 응용프로그램이 높은 권한을 가지고 수행됩니다.

주: sqlmon(데이터베이스 시스템 모니터 스위치) API를 사용하여 명령문 스위치를 비활성화하면 조정자(governor)가 명령문 경과 시간에 근거한 응용프로그램을 조정하는 기능에 영향을 줍니다. 조정자(governor)는 모니터를 사용하여 시스템에 대한 정보를 수집합니다. 데이터베이스 관리 프로그램 구성 파일에

서 스위치를 끌 경우, 구성 파일은 전체 인스턴스에 대해 꺼지고 조정자(governor)는 더 이상 이 정보를 수신하지 않습니다.

스케줄 조치는 다음을 수행할 수 있습니다.

- 서로 다른 그룹에 있는 응용프로그램이 각각 모든 응용프로그램 시간을 공평하게 분할하지 않고 시간을 확보하도록 합니다.

  예를 들어, 12개의 응용프로그램(세 개는 짧은 응용프로그램, 5개는 중간, 6개는 긴 응용프로그램)이 동시에 수행되고 있을 경우, 응용프로그램은 CPU를 분할하므로 모두 응답 시간이 오래 걸릴 수 있습니다. 데이터베이스 관리 프로그램은 중간 길이 응용프로그램과 긴 길이 응용프로그램의 두 그룹을 설정할 수 있습니다. 조정자(governor)는 우선순위를 사용하여, 짧은 모든 응용프로그램이 수행되도록 허용하고, 최대 세 개의 중간 응용프로그램과 긴 응용프로그램이 동시에 수행되도록 합니다. 이렇게 하기 위해 조정자(governor) 구성 파일에는 중간 길이 응용프로그램에 대한 한 가지의 규칙과, 긴 응용프로그램에 대한 또 다른 규칙이 있습니다. 다음 예에서는 이 시점을 나타내는 조정자(governor) 구성 파일의 부분을 보여줍니다.

  ```
  desc "Group together medium applications in 1 schedule class"
  applname medq1, medq2, medq3, medq4, medq5
  setlimit cpu -1
  action schedule class;
  desc "Group together long applications in 1 schedule class"
  applname longq1, longq2, longq3, longq4, longq5, longq6
  setlimit cpu -1
  action schedule class;
  ```

- 몇 개의 사용자 그룹(예: 조직의 부서) 각각이 같은 우선순위를 확보하도록 합니다.

  하나의 그룹이 많은 응용프로그램을 수행하고 있으면 관리자는 다른 그룹이 계속해서 해당되는 응용프로그램에 대해 적당한 응답 시간을 확보할 수 있습니다. 예를 들어, 세 개의 부서(재정, 재고 및 기획)가 있을 경우, 모든 재정부 사용자를 한 그룹에 두고, 모든 재고부 사용자를 두 번째 그룹에 두며, 모든 기획부 사용자를 세 번째 그룹에 둘 수 있습니다. 처리 능력은 세

부서 사이에 더 공평하게 또는 덜 공평하게 분할됩니다. 다음 예에서는 이 시점을 나타내는 조정자(governor) 구성 파일의 부분을 보여줍니다.

```
desc "Group together Finance department users"
authid tom, dick, harry, mo, larry, curly
setlimit cpu -1
action schedule class;
desc "Group together Inventory department users"
authid pat, chris, jack, jill
setlimit cpu -1
action schedule class;
desc "Group together Planning department users"
authid tara, dianne, henrietta, maureen, linda, candy
setlimit cpu -1
action schedule class;
```

- 조정자(governor)가 모든 응용프로그램을 스케줄링하게 합니다.

  클래스 옵션이 조치와 함께 포함되지 않으면 조정자(governor)가 스케줄 조치하에 속하는 응용프로그램 수에 따라 해당되는 고유한 클래스를 작성하여, 응용프로그램이 수행 중인 조회에 대한 DB2 조회 컴파일러 비용 계산을 기초로 서로 다른 클래스에 응용프로그램을 둡니다. 관리자는 모든 응용프로그램이 선택되는 응용프로그램을 규정하지 않고 스케줄링되도록 선택할 수 있습니다. 즉, *applname* 또는 *authid*절은 전혀 제공되지 않으며, *setlimit*절은 어떠한 제한도 초래하지 않습니다.

주: 제한을 초과하고 조치 절이 지정되어 있지 않을 경우, 조정자(governor)는 응용프로그램에 대해 작동 중인 에이전트의 우선순위를 낮춥니다.

둘 이상의 규칙이 응용프로그램에 적용될 경우, 모든 규칙이 적용됩니다. 설정은 규칙과 한계에 따라 결정되는데, 첫 번째 발생하는 규칙 한계와 연관된 조치는 첫 번째로 적용되는 조치입니다. 규칙의 절에 -1을 지정할 경우, 예외가 발생합니다. 이 상황에서 후속 규칙의 절에 지정된 값은 같은 절에 지정된 앞의 값만을 대체할 수 있습니다. 이전 규칙의 기타 절은 계속 수행됩니다. 예를 들어, 경과 시간이 1시간 이상이거나 100 000행 이상(즉, `rowssel 100000 uowtime 3600`)을 넘게 선택할 경우, 하나의 규칙은 응용프로그램의 우선순위가 감소되었음을 나타냅니다. 후속 규칙은 동일한 응용프로그램의 경과 시간이 무제한일 수 있음을 나타냅니다(즉,

uowtime -1). 이 경우, 응용프로그램이 1시간 이상 수행될 경우, 우선순위는 바뀌지 않지만(즉, uowtime -1이 uowtime 3600을 겹쳐씀), 100 000행 이상 선택할 경우 우선순위는 낮아집니다(rowssel 100000은 계속 유효함).

336 페이지의 그림29에서는 구성 파일의 예를 보여줍니다.

```
{ Wake up once a second, the database name is ibmsampl
 do accounting every 30 minutes.  }
interval 1; dbname ibmsampl; account 30;
desc "CPU restrictions apply 24 hours a day to everyone"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;
desc "Allow no UOW to run for more than an hour"
setlimit uowtime 3600 action force;
desc 'Slow down a subset of applications'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;
desc "Have governor prioritize these 6 long apps in 1 class"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;
desc "Schedule all applications run by the planning dept"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;
desc "Schedule all CPU hogs in one class which will control consumption"
setlimit cpu 3600
action schedule class;
desc "Slow down the use of db2 CLP by the novice user"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;
desc "During day hours do not let anyone run for more than 10 seconds"
time 8:30 17:00 setlimit cpu 10 action force;
desc "Allow users doing performance tuning to run some of
     their applications during lunch hour"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;
desc "Some people should not be limited -- database administrator
  and a few others.  As this is the last specification in the
  file, it will override what came before."
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;
desc "Increase the priority of an important application so it always
      completes quickly"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

그림 29. 조정자(governor) 구성 파일 예

조정자(governor) 로그 파일

조정자(governor) 디먼은 응용프로그램을 강제 실행(force)하고 조정자(governor) 구성 파일을 읽으며 응용프로그램의 우선순위를 변경하고 오류나 경고, 시작이나 종료가 발생할 때, 레코드를 로그 파일에 기록합니다. 각 조정자(governor) 디먼에 대해 별도의 로그 파일이 존재합니다. 이렇게 하면 여러 조정자(governor) 디먼이 동시에 한 파일에 기록하려고 할 때 발생할 수 있는 파일 잠금 병목 현상을 막을 수 있습니다. db2govlg 유틸리티를 사용하여 로그 파일을 병합하고 조회할 수 있습니다. 이 유틸리티는 338 페이지의 『조정자(governor) 로그 파일 조회』에 설명되어 있습니다.

로그 파일은 sqllib 디렉토리의 log 서브디렉토리에 저장됩니다. Windows NT에선 log 서브디렉토리는 인스턴스 디렉토리 밑에 있습니다. db2gov 명령을 발행할 때에는 로그 파일의 기본 이름을 제공합니다. 조정 중인 각 데이터베이스 노드에 대해 로그 파일이 존재하기 때문에, 로그 파일 이름에 데이터베이스 이름이 들어 있는지 확인해야 합니다. 파티션된 데이터베이스 환경에서 조정자(governor)가 수행 중인 데이터베이스 파티션의 노드 수는 로그 파일 이름에 자동으로 추가되어 조정자(governor)에 대해 파일 이름이 고유한 사실을 확인합니다.

로그 파일의 각 레코드는 다음과 같은 형식으로 이루어집니다.

```
Date Time NodeNum RecType Message
```

Date 및 *Time* 필드는 yyyy-mm-dd hh.mm.ss 형식에 있어서, 이 필드를 정렬하여 데이터베이스 파티션마다 로그 파일을 병합할 수 있습니다.

NodeNum 필드는 조정자(governor)가 수행 중인 데이터베이스 파티션 수를 나타냅니다.

RecType 필드에는 로그에 작성되고 있는 로그 레코드 유형에 따라 다양한 값이 포함됩니다. 기록될 수 있는 값은 다음과 같습니다.

- START: 조정자(governor)가 시작되었음을 나타냄
- FORCE: 응용프로그램이 강제 실행(force)되었음을 나타냄
- PRIORITY: 응용프로그램의 우선순위가 변경되었음을 나타냄
- ERROR: 오류를 나타냄
- WARNING: 경고를 나타냄

- READCFG: 조정자(governor)가 구성 파일을 읽음을 나타냄
- STOP: 조정자(governor)가 중지되었음을 나타냄
- ACCOUNT: 응용프로그램의 계정 통계를 산출하고 있음을 나타냄

  필드는 다음과 같습니다.

  - authid
  - appl_id
  - written_usr_cpu
  - written_sys_cpu
  - appl_con_time
- SCHEDULE: 에이전트 우선순위가 변경되었음을 나타냄

표준값이 작성되었기 때문에, 다양한 유형의 조치에 대해 로그 파일을 조회할 수 있습니다. *Message* 필드에는 *RecType* 필드의 값에 따라 달라지는 표준이 아닌 값이 제공됩니다. 예를 들어, FORCE 또는 NICE 레코드는 *Message* 필드의 응용프로그램 정보를 나타내는 반면, ERROR 레코드는 오류 메시지를 포함합니다.

로그 파일의 예는 다음과 같습니다.

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

조정자(governor) 로그 파일 조회

각 조정자(governor) 디먼은 자체의 로그 파일에 작성됩니다. db2govlg 유틸리티를 사용하여 로그 파일을 조회할 수 있습니다. 하나의 파티션 또는 모든 데이터베이스 파티션에서 날짜와 시간으로 정렬된 로그 파일을 나열할 수 있습니다. 또한 *RecType* 로그 필드의 기초를 조회할 수 있습니다. db2govlg의 구문은 다음과 같습니다.

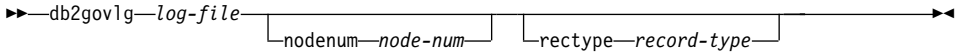


그림 30. db2govlg용 구문

매개변수는 다음과 같습니다.

log-file 조회할 로그 파일의 기본 이름

nodenum **node-num**

조정자(governor)가 수행 중인 데이터베이스 파티션의 노드 수

rectype **record-type**

조회할 레코드 유형. 레코드 유형은 다음과 같습니다.

- START
- READCFG
- STOP
- FORCE
- NICE
- ERROR
- WARNING
- ACCOUNT

이 유틸리티를 사용하는 경우에는 권한 부여에 제한이 없습니다. 이렇게 하면 모든 사용자는 조정자(governor)가 응용프로그램에 영향을 주는지 여부를 조회할 수 있습니다. 이 유틸리티에 대한 액세스를 제한하려는 경우, `db2govlg` 파일의 그룹 사용권한을 변경할 수 있습니다.

조정자(governor) 및 데이터베이스 관리 프로그램 성능 수행

조정자(governor)는 데이터베이스 관리 프로그램의 스냅샷을 요청하기 때문에 데이터베이스 관리 프로그램의 성능에 영향을 줄 수 있습니다. 조정자(governor)가 너무 많은 CPU를 사용할 경우, 가동 간격을 늘려서 CPU 사용을 줄일 수 있습니다.

제10장 프로세서 추가를 통해 구성 조정

구성 특성이 사용자의 현재 요구나 계획한 요구에 맞지 않는 경우가 있습니다. 결과적으로, 구성 용량, 성능 또는 둘 다를 증가시키는 조치를 고려해야 합니다. 예를 들어, 구성에 컨테이너를 추가하면 데이터를 저장할 용량이 증가될 뿐 아니라 유틸리티를 사용하는 동안(예를 들어, 데이터를 로드할 때) 성능이 향상될 수 있습니다. 용량이나 성능을 향상하는 다른 방법으로는, 메모리를 추가하는 방법과 대칭적 멀티프로세서 또는 파티션된 데이터베이스 환경에서 프로세서를 추가하는 것입니다.

이 장의 초점은 구성에서 프로세서 수를 증가하여 성능을 향상시키는 것입니다.

다음과 같은 경우, 이 장의 나머지 부분에서 논의되는 바와 같이 구성 스케일 조정을 고려해 보아야 합니다.

- 최대 용량으로 사용 중인 단일 프로세서를 가진 단일 파티션 구성이 있습니다. 따라서 구성을 변경하기로 결정하고 다음과 같은 판단을 내렸습니다.
  - 대칭적 멀티프로세서(SMP) 구성을 결정하는 것이 새 환경에 대한 최상의 선택입니다. 둘 이상의 프로세서의 처리 능력을 사용하기 위해 이 선택이 이루어졌습니다. 각 프로세서는 메모리와 저장 시스템 자원을 공유합니다. 모든 프로세서가 한 시스템 안에 있으므로, 시스템간의 통신 회선과 같은 추가 고려사항은 없으며, 새 시스템을 지원하기 위해 추가 관리 스탭도 없고 시스템간의 타스크 조정도 문제되지 않습니다. DB2 Universal Database는 이 환경을 지원합니다.
  - 파티션된 데이터베이스 구성을 결정하는 것이 새 환경에 대한 최상의 선택입니다. 첫 번째 프로세서와 물리적으로 분리된 둘 이상의 프로세서의 처리 능력을 사용하기 위해 이 선택이 이루어졌습니다. 각 프로세서는 다른 프로세서와의 공유 없이 자체 메모리와 저장 시스템 자원을 갖습니다. 위에서 설명된 추가 고려사항(통신, 스탭, 타스크 조정)을 생각해 보아야 하는 반면, 이 선택에는 둘 이상의 시스템에 걸쳐 데이터와 사용자 액세스의 균형을 맞추는 기능과 같은 이점이 있습니다. DB2 Universal Database는 이 환경을 지원합니다.

© Copyright IBM Corp. 1993, 2001

- 현재 SMP 구성 상태이고 여기에 하나 이상의 프로세서를 추가하기로 합니다. 이 경우, 이 환경 유형과 연관된 해당 고려사항은 이미 익숙합니다. 하나 이상의 추가 프로세서를 추가하면 새 고려사항을 추가하지 않고도 간단하게 컴퓨팅 능력을 환경에 추가할 수 있습니다. DB2 Universal Database는 이 환경을 지원합니다.
- 현재 파티션된 데이터베이스 구성 상태이고 하나 이상의 데이터베이스 파티션을 추가하기로 합니다. 이 경우, 이 환경 유형과 연관된 해당 고려사항은 이미 익숙합니다. 하나 이상의 추가 데이터베이스 파티션을 추가하면 많은 수의 파티션으로 이전하는 것 외에는 새 고려사항을 추가하지 않고도 간단하게 컴퓨팅 능력을 환경에 추가할 수 있습니다. DB2 Universal Database는 이 환경을 지원합니다.

  파티션된 데이터베이스 구성에서의 변동은 데이터베이스 파티션이 SMP 머신인 변동입니다. DB2 Universal Database는 이 환경을 지원합니다.

환경을 변경하여 시스템의 스케일을 조정할 경우에는 이러한 변경이 데이터베이스 백업 및 복원, 데이터 로드와 같은 데이터베이스 프로시듀어에 미치는 영향에 대해 알고 있어야 합니다.

새 데이터베이스 파티션을 추가할 경우, 프로시듀어가 완료될 때까지 새 파티션을 이용하는 데이터베이스를 삭제하거나 작성할 수 없으며, 새 서버는 성공적으로 시스템에 통합됩니다.

머신에 프로세서 추가

기존 프로세서가 오랜 시간 동안 전체적으로 이용될 경우, 머신에 하나 이상의 추가 프로세서를 설치할 것을 고려해 보십시오. DB2 데이터베이스 관리 프로그램이 새 프로세서를 활용할 수 있도록 하기 위해서 여러 구성 매개변수를 검토하고 갱신해야 합니다. (Solaris와 같은 몇 가지 운영 체제는 온라인 프로세서와 오프라인 프로세서를 동적으로 변경할 수 있습니다.) 사용된 프로세서의 수를 결정하고 갱신되어야 하는 매개변수에는 다음이 포함됩니다.

- 514 페이지의 『기본 등급(dft_degree)』
- 543 페이지의 『병렬의 처리 최대 조회 수준(max_querydegree)』
- 544 페이지의 『파티션 내 병렬 처리 작동(intra_parallel)』

갱신될 필요가 있는 응용프로그램과 연관된 매개변수도 고려해야 합니다. 자세한 내용은 98 페이지의 『응용프로그램의 병렬 처리』를 참조하십시오.

TCP/IP가 통신에 사용되는 환경에서 작업할 때 DB2TCPCONNMGRS 레지스트리 변수의 값을 고려해야 합니다. 이 변수에 대한 자세한 내용은 575 페이지의 『부록A. DB2 레지스트리 및 환경 변수』를 참조하십시오.

파티션된 데이터베이스 시스템에 데이터베이스 파티션 추가

시스템이 수행 중이거나 중지될 때 데이터베이스 파티션을 파티션된 데이터베이스에 추가할 수 있습니다. 다음 절에서는 이 타스크를 수행하는 방법에 대해 설명합니다. 새 서버를 추가하는 데 시간이 걸리기 때문에, 데이터베이스 관리 프로그램이 이미 수행 중일 때 추가해야 합니다. 프로시듀어는 344 페이지의 『수행 중인 시스템에 데이터베이스 파티션 추가』에 설명되어 있습니다.

ADD NODE 명령을 사용하여 데이터베이스 파티션을 시스템에 추가합니다. 이 명령은 다음에 의해 호출될 수 있습니다.

- `db2start` 옵션으로서
- 다음을 사용하십시오.
  - 명령행 처리기 ADD NODE 명령
  - `sqleaddn`
  - `sqlepstart`

명령을 호출하는 데 사용하는 방법은 시스템이 중지된 상태(`db2start` 사용)인지 수행(다른 선택사항 사용)중인지에 따라 결정됩니다.

ADD NODE 명령을 사용하여 새 데이터베이스 파티션이 시스템에 추가되는 경우, 인스턴스의 모든 기존 데이터베이스는 새 데이터베이스 파티션에서 작성됩니다. 작성되는 데이터베이스와 함께 사용될 임시 테이블 공간용 컨테이너를 지정할 수도 있습니다. 컨테이너는 다음과 같습니다.

- 데이터베이스마다 카탈로그 노드에서 정의된 컨테이너와 같습니다. (이 컨테이너는 기본값입니다.)
- 다른 데이터베이스 파티션에 정의된 것과 같습니다.

- 전혀 작성되지 않았습니다. ALTER TABLESPACE문을 사용하여 데이터베이스를 사용하기 전에 임시 테이블 공간 컨테이너를 데이터베이스마다 추가해야 합니다.

새 파티션의 데이터베이스는 하나 이상의 노드 그룹이 새 데이터베이스 파티션을 포함하도록 변경되기 전까지 데이터를 포함하는 데 사용할 수 없습니다. 노드 그룹 변경에 대한 자세한 내용은 357 페이지의 『데이터베이스 파티션의 추가 및 삭제』를 참조하십시오.

주: 시스템에 정의된 데이터베이스가 없고 UNIX 기반 시스템에서 DB2 Enterprise - Extended Edition을 수행 중일 경우, `db2nodes.cfg` 파일을 편집하여 새 데이터베이스 파티션 정의를 추가하십시오. 데이터베이스가 존재할 때에만 적용되므로, 다음 프로시듀어는 사용하지 마십시오. 노드 구성 파일을 갱신하는 방법에 대한 자세한 내용은 *관리 안내서: 계획*의 『노드 그룹 변경』을 참조하십시오.

Windows NT 고려사항: Windows NT에서 DB2 Enterprise - Extended Edition을 사용하고 있고 인스턴스에 데이터베이스가 없으면, DB2NCRT 명령을 사용하여 데이터베이스 시스템을 조정해야 합니다. 이 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오. 한편, 이미 데이터베이스가 있는 경우에는 DB2START ADDNODE 명령을 사용해야 합니다. 왜냐하면, 시스템을 스케일할 때 기존의 데이터베이스마다 데이터베이스 파티션이 작성되기 때문입니다. DB2START 명령 및 Windows NT에서 사용해야 하는 매개변수에 대한 자세한 내용은 *Command Reference*을 참조하십시오. Windows NT에서는 절대로 노드 구성 파일(`db2nodes.cfg`)을 수동으로 편집하지 말아야 합니다. 왜냐하면, 이로 인해 파일에 불일치가 생길 수 있기 때문입니다.

수행 중인 시스템에 데이터베이스 파티션 추가

파티션된 데이터베이스 시스템이 수행되고 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 서버를 사용할 수 있습니다.

다중 서버 시스템에 데이터베이스 파티션을 추가하려면 다음을 수행하십시오.

1. 데이터베이스 파티션이 이미 시스템에 존재하는 서버에서 작성될 경우, 다음 단계를 수행하십시오. 그렇지 않으면 다음을 수행하십시오.
   - UNIX 플랫폼에서
     a. 새 서버를 설치하십시오. 이 작업에는 실행 파일에 액세스할 수 있는 상태로 만들기(공유 파일 시스템 마운트 또는 지역 사본 사용), 기존 프로세서에 있는 파일과 운영 체제 파일 동기화하기, `sqllib` 디렉토리에 공유 파일 시스템으로서 액세스할 수 있도록 하기, 관련 운영 체제 매개변수(프로세스의 최대 가능 횟수 등)가 적절한 값에 설정되었는지 확인하기 등이 포함됩니다.
     b. 모든 데이터베이스 파티션에서 `etc` 디렉토리의 호스트 파일에 등록하거나 이름 서버를 가진 호스트 이름을 등록하십시오.
   - Windows NT 플랫폼에서
     a. 새 서버를 설치하십시오.
     b. 새 서버에서 ADD NODE 명령을 수행하십시오. 이 명령은 시스템에 이미 존재하는 모든 데이터베이스에 대해 데이터베이스 파티션이 지역적으로 작성됩니다. 새 데이터베이스 파티션의 데이터베이스 매개변수는 기본값으로 설정되며, 각 데이터베이스 파티션은 사용자가 데이터를 파티션에 이동시킬 때까지 비어 있습니다. 다른 데이터베이스 파티션에서 발견되는 것과 일치하도록 데이터베이스 구성 매개변수를 갱신해야 합니다.
     c. 시점 3으로 이동하십시오.
2. NODENUM, ADDNODE, HOSTNAME, PORT 및 NETNAME 매개변수를 위한 새 파티션 값을 지정하여 하나의 데이터베이스 파티션에서 DB2START 명령을 수행하십시오. Windows NT 플랫폼에서 COMPUTER, USER 및 PASSWORD 매개변수도 지정해야 합니다. DB2START 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

   데이터베이스와 함께 작성되어야 하는 임시 테이블 공간 컨테이너 정의에 대한 소스를 선택적으로 지정할 수도 있습니다. 제공된 테이블 공간 정보가 없는 경우, 임시 테이블 공간 컨테이너 정의가 각 데이터베이스의 카탈로그 노드로부터 검색됩니다.

   명령을 완료하면 새 서버는 중지됩니다. 노드 구성 파일은 DB2STOP이 실행될 때까지 새 서버 정보로 갱신되지 않습니다. 이것은 ADD NODE 명령

(ADDNODE 매개변수가 지정되면 호출됨)이 올바른 데이터베이스 파티션에서 수행되도록 하기 위함입니다. 유틸리티가 종료되면 새 서버는 중지됩니다.

3. DB2STOP 명령을 수행하여 데이터베이스 관리 프로그램을 시작하십시오.

   시스템의 모든 데이터베이스 파티션를 중지시키는 경우, 새 데이터베이스 파티션을 포함하도록 노드 구성 파일이 갱신됩니다.

4. DB2START 명령을 수행하여 데이터베이스 관리 프로그램을 중지시키십시오.

   새로 추가된 데이터베이스 파티션은 시스템의 나머지 파티션과 함께 이제 시작됩니다.

   시스템의 모든 데이터베이스 파티션이 수행 중일 때, 데이터베이스 작성이나 삭제와 같은 전체 시스템 활동을 수행할 수 있습니다.

   주: 새 `db2nodes.cfg` 파일에 액세스하기 위해 모든 데이터베이스 파티션 서버에 대해 DB2START 명령을 두 번 발행해야 될지도 모릅니다.

5. 선택적으로 새 데이터베이스 파티션의 모든 데이터베이스를 백업하십시오.
6. 선택적으로 데이터를 새 데이터베이스 파티션에 재분산하십시오. 자세한 내용은 355 페이지의 『제11장 데이터베이스 파티션에 걸친 데이터 재분산』을 참조하십시오.

중지된 시스템에 데이터베이스 파티션 추가

파티션된 데이터베이스 시스템이 중지될 때 새 데이터베이스 파티션을 파티션된 데이터베이스 시스템에 추가할 수 있습니다. 데이터베이스 관리 프로그램이 다시 시작되면 새로 추가된 데이터베이스 파티션을 모든 데이터베이스에서 사용할 수 있습니다. 추가하는 방법에는 두 가지가 있습니다. 데이터베이스 관리 프로그램이 노드 구성 파일을 갱신하게 하거나, 사용자가 직접 수행할 수 있습니다. 두 가지 방법에 대한 사전 작업 단계는 같습니다.

주: Windows NT에서 작업하는 중에는 노드 구성 파일을 수동으로 갱신하지 마십시오. 그 대신, 이 파일을 갱신하려면 아래에 설명된 대로 데이터베이스 관리 프로그램을 사용해야 합니다.

다중 서버 시스템에 새 데이터베이스 파티션을 추가하려면 다음을 수행하십시오.

1. DB2STOP을 발행하여 모든 데이터베이스 파티션을 중지시키십시오.

2. 서버가 시스템에 이미 존재하는 프로세서에 대해 작성될 경우, 다음 단계를 수행하십시오. 그렇지 않으면 다음을 수행하십시오.
   a. UNIX 플랫폼에서
      1) 새 서버를 설치하십시오. 이 작업에는 실행 파일에 액세스할 수 있는 상태로 만들기(공유 파일 시스템 마운트 또는 지역 사본 사용), 기존 프로세서에 있는 파일과 운영 체제 파일 동기화하기, `sqllib` 디렉토리에 공유 파일 시스템으로서 액세스할 수 있도록 하기, 관련 운영 체제 매개변수(프로세스의 최대 가능 횟수 등)가 적절한 값에 설정되었는지 확인하기 등이 포함됩니다.
      2) 모든 데이터베이스 파티션에서 `etc` 디렉토리의 호스트 파일에 등록하거나 이름 서버를 가진 호스트 이름을 등록하십시오.
   b. Windows NT 플랫폼에서
      1) 새 서버를 설치하십시오.
      2) 새 서버에서 ADD NODE 명령을 수행하십시오. 이 명령은 시스템에 이미 존재하는 모든 데이터베이스에 대해 데이터베이스 파티션이 지역적으로 작성됩니다. 새 데이터베이스 파티션의 데이터베이스 매개변수는 기본값으로 설정되며, 각 데이터베이스 파티션은 사용자가 데이터를 파티션에 이동시킬 때까지 비어 있습니다. 다른 데이터베이스 파티션에서 발견되는 것과 일치하도록 데이터베이스 구성 매개변수를 갱신해야 합니다.
      3) DB2START 명령을 수행하여 데이터베이스 시스템을 시작하십시오. 새 서버를 설치하는 중에 노드 구성 파일(`db2nodes.cfg`)이 이미 새 서버를 포함하도록 갱신되어 있어야 합니다.
      4) 선택적으로 데이터를 새 서버에 재분산하십시오. 이를 수행하는 방법에 대한 자세한 내용은 355 페이지의 『제11장 데이터베이스 파티션에 걸친 데이터 재분산』을 참조하십시오.
   c. 데이터베이스 관리 프로그램이 사용자를 대신하여 `db2nodes.cfg` 파일을 갱신하게 하려면 348 페이지의 『데이터베이스 관리 프로그램이 노드 구성 파일을 갱신하도록 하기』의 지시사항을 따르십시오.

주: Windows NT에서 db2nodes.cfg 파일을 수동으로 편집하지 말아야 합니다. 왜냐하면, 이로 인해 파일에 불일치가 생길 수 있기 때문입니다. 그 대신, 데이터베이스 관리 프로그램이 이 파일을 갱신해야 합니다.

db2nodes.cfg 파일을 직접 사용자가 갱신하려면 349 페이지의 『수동으로 노드 구성 파일 갱신』의 지시사항을 따르십시오.

데이터베이스 관리 프로그램이 노드 구성 파일을 갱신하도록 하기

하나 이상의 새 데이터베이스 파티션을 파티션된 데이터베이스 시스템에 추가한 다음 새 파티션을 사용 가능하게 만들려면 db2nodes.cfg 파일을 갱신해야 합니다. 데이터베이스 관리 프로그램이 노드 구성 파일을 갱신하도록 한 경우, 다음 정보는 수정해야 할 내용에 대한 세부사항을 제공합니다.

주: 수동으로 노드 구성 파일을 갱신하도록 한 경우, 이 절에 있는 나머지 정보를 무시하십시오.

다음과 같이 프로시듀어를 진행하십시오.

1. NODENUM, ADDNODE, HOSTNAME, PORT 및 NETNAME 매개변수를 지정하여 새 데이터베이스 파티션에서 DB2START 명령을 수행하십시오. Windows NT 플랫폼에서 COMPUTER, USER 및 PASSWORD 매개변수도 지정해야 합니다. DB2START 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오. 이 매개변수에 대해 사용자가 지정한 값을 사용하여 노드 구성 파일을 갱신합니다.

   명령을 완료하면 새 서버는 중지됩니다. 노드 구성 파일은 DB2STOP이 실행될 때까지 새 서버 정보로 갱신되지 않습니다. 이것은 ADD NODE 명령(ADDNODE 매개변수가 지정되면 호출됨)이 올바른 데이터베이스 파티션에서 수행되도록 하기 위함입니다. 유틸리티가 종료되면 새 데이터베이스 파티션은 중지됩니다.
2. DB2STOP 명령을 발행하십시오.

   새 데이터베이스 파티션을 포함하기 위해 DB2STOP 명령을 발행하면, 노드 구성 파일이 갱신됩니다.
3. DB2START 명령을 발행하여 데이터베이스 시스템을 시작하십시오.

주: 새 노드 구성 파일에 액세스하기 위해 모든 데이터베이스 파티션에 대해 DB2START 명령을 두 번 수행해야 될지도 모릅니다.

4. 선택적으로 새 데이터베이스 파티션의 모든 데이터베이스를 백업합니다.
5. 데이터를 새 서버에 선택적으로서 재분산하십시오. 자세한 내용은 355 페이지의 『제11장 데이터베이스 파티션에 걸친 데이터 재분산』을 참조하십시오.

수동으로 노드 구성 파일 갱신

하나 이상의 새 데이터베이스 파티션을 파티션된 데이터베이스 시스템에 추가한 다음 새 파티션을 사용 가능하게 만들려면 반드시 db2nodes.cfg 파일을 갱신해야 합니다. 수동으로 노드 구성 파일을 갱신하도록 한 경우, 다음 정보는 노드 구성 파일을 수동으로 갱신하는 데 필요한 세부사항을 제공합니다. (Windows NT에서 작업할 때는 노드 구성 파일를 수동으로 갱신하지 말아야 한다는 점을 상기하십시오.)

주: 데이터베이스 관리 프로그램이 노드 구성 파일을 갱신하도록 한 경우, 348 페이지의 『데이터베이스 관리 프로그램이 노드 구성 파일을 갱신하도록 하기』로 되돌아가십시오.

다음과 같이 프로시듀어를 진행하십시오.

1. db2nodes.cfg 파일을 편집하고, 여기에 새 데이터베이스 파티션을 추가하십시오.
2. 새 노드를 시작하려면 다음 명령을 발행하십시오.

   ```
   DB2START NODENUM nodenum
   ```

   새 데이터베이스 파티션에 할당할 수를 *nodenum* 값으로 지정하십시오.
3. 새 서버가 논리 데이터베이스 파티션이어야 하는 경우에(즉, 노드 0이 아닐 때), **db2set** 명령을 사용하여 추가 중인 데이터베이스 파티션의 수를 지정하면서 DB2NODE 레지스트리 변수를 갱신하십시오.
4. 새 데이터베이스 파티션에서 ADD NODE 명령을 수행하십시오.

   이 명령을 사용하면 시스템에 이미 존재하는 모든 데이터베이스에 대해 데이터베이스 파티션이 지역적으로 작성됩니다. 새 데이터베이스 파티션의 데이터베이스 매개변수는 기본값으로 설정되며, 각 데이터베이스 파티션은 사용자가 데

이터를 파티션에 이동시킬 때까지 비어 있습니다. 다른 데이터베이스 파티션에서 발견되는 것과 일치하도록 데이터베이스 구성 매개변수를 갱신해야 합니다.

5. ADD NODE 명령이 완료되면 DB2START 명령을 발행하여 시스템의 다른 데이터베이스 파티션을 시작하십시오.

   모든 데이터베이스 파티션이 성공적으로 시작될 때까지 데이터베이스 작성 또는 삭제와 같은 전체 시스템 활동을 시도해서는 안됩니다.
6. 선택적으로 새 서버에서 모든 새 데이터베이스 파티션을 백업하십시오.
7. 선택적으로 데이터를 새 데이터베이스 파티션에 재분산하십시오. 자세한 내용은 355 페이지의 『제11장 데이터베이스 파티션에 걸친 데이터 재분산』을 참조하십시오.

시스템에서 데이터베이스 파티션 삭제

DB2STOP 명령을 DROP NODENUM 매개변수와 함께 사용하거나 sqlepstp API를 사용하여 데이터베이스 파티션을 삭제할 수 있습니다. 이를 수행하기 전에 삭제할 데이터베이스 파티션을 다른 데이터베이스가 사용하지는 않는지 먼저 확인해야 합니다. 확인하려면 DROP NODE VERIFY 명령을 발행하십시오.

이 데이터베이스 파티션이 조정자(coordinator) 역할을 한 모든 트랜잭션이 성공적으로 확약되거나 구간 복원되었는지 확인해야 합니다. 이 작업 중 다른 서버에 발생되었을 수 있는 응급 복구 작업을 하게 될 수도 있습니다.

예를 들어, 조정자(coordinator) 데이터베이스 파티션(즉, 조정자(coordinator) 노드)을 삭제하고 조정자(coordinator) 노드가 삭제되기 전에 트랜잭션에 참여한 다른 데이터베이스 파티션이 손상되면 충돌된 데이터베이스 파티션은 2단계 확약 중 이상 실패 트랜잭션에서 발생한 사항에 대하여 조정자(coordinator) 노드를 조회할 수 없습니다.

파티션된 데이터베이스 시스템에서 데이터베이스 파티션을 삭제하려면 다음을 수행하십시오.

1. 이 노드에 상주하는 모든 데이터베이스에 대한 데이터를 재분산하십시오. 이는 삭제할 데이터베이스 파티션이 데이터베이스에서 사용되지 않아야 한다는 요구

사항을 만족시킵니다. 자세한 내용은 355 페이지의 『제11장 데이터베이스 파티션에 걸친 데이터 재분산』을 참조하십시오.

2. DROP NODE VERIFY 명령 또는 sqledrpn API를 발행하여 서버를 사용하고 있지 않음을 확인하십시오.

수신되는 메시지에 따라 3단계 또는 4단계를 진행하십시오.

3. SQL6034W 메시지(어떠한 데이터베이스에서도 사용되지 않은 노드)를 수신할 경우, 다음을 수행할 수 있습니다.
   a. DB2STOP 명령을 DROP NODENUM 매개변수와 함께 발행하여 데이터베이스 파티션을 삭제하십시오. 명령이 성공적으로 완료한 후에 시스템은 중지됩니다.
   b. 원할 경우, DB2START 명령으로 데이터베이스 관리 프로그램을 시작하십시오.
4. SQL6035W 메시지(데이터베이스가 사용하는 노드)를 수신하면 다음을 수행하십시오.
   a. 메시지 SQL6035W의 지시대로 REDISTRIBUTE NODEGROUP 명령을 사용하여 삭제 중인 데이터베이스 파티션에서 데이터베이스 별명을 가진 다른 데이터베이스 파티션으로 데이터를 재분산하십시오. 이 작업이 수행될 때까지 데이터베이스 파티션을 삭제할 수 없습니다.
   b. 데이터베이스 파티션에서 정의된 모든 이벤트 모니터를 삭제하십시오.
   c. 2단계로 리턴한 후 계속하십시오.

파티션된 데이터베이스에 노드 추가 시 문제점

기본 페이지 *크기*(4kb)와는 다른 페이지 *크기*를 갖는 하나 이상의 시스템 임시 테이블 공간을 포함하는 파티션된 데이터베이스에 노드를 추가하는 경우, 다음 오류 메시지가 표시될 수 있습니다. 『SQL6073N 노드 추가 조작에 실패했습니다.』 SQLCODE. 이 오류 메시지의 발생 이유는 노드가 작성될 때 IBMCEFAULTBP 버퍼 풀만 4KB *크기*의 페이지로 존재하기 때문입니다.

예를 들어, **db2start** 명령을 사용하여 현재 파티션된 데이터베이스에 노드를 추가할 수 있습니다.

```
DB2START NODENUM 2 ADDNODE HOSTNAME newhost PORT 2
```

파티션된 데이터베이스가 기본 크기의 시스템 임시 테이블 공간을 가지고 있는 경우, 다음 메시지가 리턴됩니다.

```
SQL6075W 데이터베이스 관리 프로그램 시작 조작을 노드에 추가했습니다.
이 노드는 모든 노드를 중지한 후 다시 시작할 때까지 사용되지 않습니다.
```

그러나 파티션된 데이터베이스가 기본 페이지 크기가 아닌 시스템 임시 테이블 공간을 가지고 있는 경우, 리턴된 메시지는 다음과 같습니다.

```
SQL6073N 노드 추가 조작에 실패했습니다. SQLCODE = "<-902>"
```

비슷한 예로, db2nodes.cfg 파일을 수동으로 갱신한 뒤 새 노드 설명과 함께 ADD NODE 명령을 사용할 수 있습니다. 파티션된 데이터베이스로 파일 편집과 ADD NODE 명령 수행 이후에 기본 페이지 크기의 시스템 임시 테이블 공간을 가지며, 다음과 같은 메시지가 리턴됩니다.

```
DB20000I ADD NODE 명령을 완료했습니다.
```

그러나 파티션된 데이터베이스가 기본 페이지 크기가 아닌 시스템 임시 테이블 공간을 가지고 있는 경우, 리턴된 메시지는 다음과 같습니다.

```
SQL6073N 노드 추가 조작에 실패했습니다. SQLCODE = "<-902>"
```

살펴 본 위의 문제를 방지하는 하나의 방법은 **db2start** 또는 ADD NODE 명령을 발행하기 전에

```
DB2SET DB2_HIDDENBP=16
```

명령을 수행하는 것입니다. 이 레지스트리 변수는 DB2가 기본값과 다른 페이지 크기를 사용하여 숨겨진 버퍼 풀 16페이지에 각각 할당할 수 있도록 합니다. 이는 ADD NODE 조작을 완료하도록 해 줍니다.

이러한 문제를 방지하는 또 다른 방법은 ADD NODE 또는 **db2start** 명령에 WITHOUT TABLESPACES 절을 지정하는 것입니다. 이렇게 한 후 CREATE BUFFERPOOL문을 사용하여 버퍼 풀을 작성하고 ALTER TABLESPACE문을 사용하여 시스템 임시 테이블 공간을 버퍼 풀과 연관시켜야 합니다.

기본 페이지 크기(4KB)와 다른 페이지 크기로 둘 이상의 테이블 공간을 차지하는 기존 노드 그룹에 노드를 추가할 때 오류 메시지: 『SQL0647N 버퍼 풀 ""이(가)

현재 사용되고 있지 않습니다』가 발생할 수 있습니다. 이것은 새 노드에 작성된 기본이 아닌 페이지 *크기* 버퍼 풀이 테이블 공간에 대해 활성화되지 않았기 때문에 발생합니다.

예를 들어, ALTER NODEGROUP문을 사용하여 노드 그룹에 노드를 추가할 수 있습니다.

```
DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2)
```

노드 그룹이 *기본* 페이지 *크기*의 테이블 공간을 갖는 *경우*, 다음 메시지가 리턴됩니다.

```
SQL1759W Redistribute nodegroup is required to change data positioning for
objects in nodegroup "<ng1>" to include some added nodes or exclude
some drop nodes.
```

그러나 노드 그룹이 *기본* 페이지 *크기*가 아닌 테이블 공간을 갖는 *경우*, 리턴된 메시지는 다음과 같습니다.

```
SQL0647N 버퍼 풀 ""이(가) 현재 사용되고 있지 않습니다.
```

이 문제를 방지하는 한 *가지* 방법은 ALTER NODEGROUP문을 발행하기 전에 각 페이지 *크기*에 대한 버퍼 풀을 작성한 다음 데이터베이스에 다시 연결하는 것입니다.

```
DB2START
CONNECT TO mpp1
CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
CONNECT RESET
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2)
```

이 문제를 방지하는 두 번째 방법은 **db2start** 명령과 CONNECT 및 ALTER NODEGROUP문을 발행하기 전에

```
DB2SET DB2_HIDDENBP=16
```

명령을 수행하는 것입니다.

테이블 공간을 노드에 추가할 경우, ALTER TABLESPACE문을 사용할 때 다른 문제가 발생할 수 있습니다. 예를 들면, 다음과 같습니다.

```
DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2) WITHOUT TABLESPACES
ALTER TABLESPACE ts1 ADD ('ts1') ON NODE (2)
```

일련의 이 명령 및 명령문은 오류 메시지 SQL0647N(예상 메시지 SQL1759W는 아님)을 생성합니다.

이 변경을 올바르게 완료하려면 ALTER NODEGROUP... WITHOUT TABLESPACES문을 실행한 후 데이터베이스에 다시 연결해야 합니다.

```
DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2) WITHOUT TABLESPACES
CONNECT RESET
CONNECT TO mpp1
ALTER TABLESPACE ts1 ADD ('ts1') ON NODE (2)
```

문제를 방지하는 또 다른 방법은 **db2start** 명령과 CONNECT, ALTER NODEGROUP 및 ALTER TABLESPACE문을 발행하기 전에

```
DB2SET DB2_HIDDENBP=16
```

명령을 수행하는 것입니다.

제11장 데이터베이스 파티션에 걸친 데이터 재분산

파티션된 데이터베이스 환경에서 작업하는 경우에만 데이터 재분산과 관련되어야 합니다. 단일 파티션된 데이터베이스 환경에 있는 경우, 여기에 있는 정보를 사용할 필요는 없습니다.

데이터 재분산 유틸리티를 사용하여 기존의 노드 그룹의 데이터베이스 파티션간에 데이터를 이동시킵니다. 다음을 수행하기 위해 이를 사용할 수 있습니다.

- 데이터베이스 파티션에 걸쳐 데이터 볼륨과 처리 로드의 균형을 맞춥니다.

  모든 데이터에 규칙적으로 액세스하는 데이터베이스 테이블의 경우 이 기능은 유용합니다.
- 데이터베이스 파티션에 걸쳐 데이터 재분산시에 데이터를 비대칭이 되게 할 수 있습니다.

  일부 데이터에만 규칙적으로 액세스하는 데이터베이스 테이블의 경우 유용합니다. 이 경우, 테이블을 재분산하여 자주 액세스되지 않는 데이터는 노드 그룹의 적은 수의 데이터베이스 파티션에 두고, 자주 액세스되는 데이터는 더 많은 파티션에 분산시킵니다. 이렇게 하면 자주 사용되는 수행 응용프로그램의 액세스 성능과 처리량을 늘릴 수 있습니다.

REDISTRIBUTE NODEGROUP 명령이 데이터 재분산 유틸리티를 호출하는 방법입니다. 이 명령의 구문에 대해서는 *Command Reference*에서 자세한 내용을 참조하십시오.

테이블 조합을 보존하기 위해 노드 그룹의 모든 테이블에 이 조작이 적용되며, 테이블 레벨이 아닌 노드 그룹 레벨에서 재분산이 이루어집니다.

원하는 데이터 분산을 수행하기 위해 유틸리티는 파티션 맵을 사용하여 노드 그룹의 데이터베이스 파티션간에 테이블 행을 이동시킵니다. 지정된 옵션에 따라 유틸리티는 목표 파티션 맵을 생성하거나 기존 파티션 맵을 입력으로 사용할 수 있습니다.

© Copyright IBM Corp. 1993, 2001

주:

1. 데이터 재분산 조작에 필요하다고 생각되는 로그 공간 요구사항에 의거하여 로그 파일 *크기*를 지정해야 합니다. 데이터가 재분산될 각 데이터베이스 파티션에서 INSERT 및 DELETE 조작을 수행할 수 있을 만큼 커야 한다는 것을 확인하십시오.
2. 복제 요약 테이블이 있는 노드 그룹에서 데이터를 재분산할 경우, 먼저 이 테이블을 삭제하고 노드 그룹을 재분산한 후 테이블을 재작성해야 합니다. 복제 요약 테이블을 포함하는 노드 그룹을 재분산할 수 없습니다.

데이터를 파티션하는 방식

기본적으로 데이터 재분산 유틸리티는 각 해쉬 파티션에 동일한 행 수가 해쉬되는 것으로 간주하기 때문에 노드 그룹의 모든 데이터베이스 파티션에 걸쳐 해쉬 파티션이 일정하게 파티션됩니다. 각 해쉬 파티션에 동일한 수의 행이 해쉬되지 않을 경우, *분산 파일*을 사용하여 현재 분산을 지정할 수 있습니다. 이 파일에는 각 4 096 해쉬 파티션에 대한 값이 들어 있습니다. 각 값은 해당되는 해쉬 파티션의 *가중치*로 사용됩니다. 데이터 재분산 유틸리티는 모든 데이터베이스 파티션이 거의 같은 가중치를 갖는 목표 파티션 맵을 생성합니다. 그러므로 데이터 분산이 비대칭적이라도 분산 파일을 사용하여 데이터를 균등하게 분산할 수 있습니다.

자동 로드 프로그램 유틸리티를 사용하여 ANALYZE 옵션으로 데이터 분산 파일을 작성할 수 있습니다. 이 파일을 데이터 재분산 유틸리티에 대한 입력으로 사용할 수 있습니다. 자동 로드 프로그램 유틸리티에 대한 자세한 내용은 *데이터 이동 유틸리티 안내 및 참조서*를 참조하십시오.

다른 방법으로는 PARTITION 및 NODENUMBER SQL 함수를 사용하여 해쉬 파티션이나 데이터베이스 파티션에 걸쳐 현재의 데이터 분산을 결정할 수 있습니다. (PARTITION 함수를 사용하여 해쉬 파티션에서의 분산을 결정합니다.) 이러한 정보를 사용하여 분산 파일 및 목표 파티션 맵을 이끌어낼 수 있습니다.

예를 들어, 일정하지 않은 데이터 분산으로 불규칙하게 많은 행을 가지고 있는 데이터베이스 파티션을 찾으려면, 다음을 수행하십시오.

```
SELECT PARTITION(column_name), COUNT(*) FROM table_name
   GROUP BY PARTITION(column_name)
   ORDER BY PARTITION(column_name) DESC
   FETCH FIRST 100 ROWS ONLY
```

table_name이 가장 큰 테이블이고 column_name이 그 테이블에서 적절한 컬럼인지 확인해야 합니다.

데이터베이스 파티션의 추가 및 삭제

ALTER NODEGROUP문을 사용하여 노드 그룹에서 데이터베이스 파티션을 추가하거나 삭제할 수 있습니다. 데이터베이스 파티션을 추가하는 경우, 파티션은 노드 구성 파일에 이미 정의되어 있어야 합니다.

ALTER NODEGROUP문을 사용하고 나면 새 파티션 맵이 작성됩니다. 이 새 파티션 맵은 데이터 재분산 유틸리티를 사용할 때 목표 파티션 맵이 됩니다. (목표 파티션 맵를 작성하는 또 다른 방법은 이를 사용자 스스로 작성하는 것입니다.)

ALTER NODEGROUP문에서 WITHOUT TABLESPACES절을 사용할 경우, 데이터를 재분산하기 전에 테이블 공간 컨테이너를 새 데이터베이스 파티션(또는 파티션)에 추가해야 합니다. ALTER NODEGROUP문에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

목표 파티션 맵 지정

데이터 재분산 유틸리티는 파티션 맵을 사용하여 데이터를 재분산합니다. 이 유틸리티가 자체의 목표 파티션 맵을 작성하기고 하고, 사용자가 이를 제공하기도 합니다. 하나를 작성하면 항목(들)이 데이터 재분산으로 인한 노드 그룹의 유형을 결정합니다.

- 단일 파티션 노드 그룹에 대한 1 항목
- 다중 파티션 노드 그룹에 대한 4 096 항목

목표 파티션 맵에 둘 이상의 데이터베이스 파티션이 있을 경우, 노드 그룹의 모든 테이블에 같은 파티션 키가 정의되어 있어야 합니다.

목표 파티션 맵에는 SYSCAT.NODEGROUPDEF 카탈로그 테이블에서 정의되고 'T'의 IN_USE 값의 것은 제외된 데이터베이스 파티션 번호만 있습니다. ('T'는 파티션이 목표 파티션 맵에 없다는 것을 의미합니다.) IN_USE 값 'D'(삭제)를 가지며 목표 파티션 맵에 표시되지 않는 모든 데이터베이스 파티션은 재분산 조작이 성공적으로 완료될 때 삭제됩니다.

데이터베이스 파티션에 걸친 데이터의 재분산 방식

데이터 재분산 조작은 데이터베이스의 지정된 노드 그룹의 지정된 테이블 세트에서 수행됩니다. (조작을 실행하기 전에 응용프로그램은 카탈로그 데이터베이스 파티션의 데이터베이스에 연결되어 있어야 합니다.) 유틸리티는 소스 파티션 맵 및 목표 파티션 맵을 모두 사용하여 새 위치(즉, 새 데이터베이스 파티션 번호)에 어떤 파티션이 지정되었는지 식별합니다. 새 위치의 파티션에 해당하는 모든 행은 소스 파티션 맵에서 지정된 데이터베이스 파티션으로부터 목표 파티션 맵에서 지정된 데이터베이스 파티션으로 이동됩니다.

데이터 재분산 유틸리티는 다음을 수행합니다.

1. 목표 파티션 맵에 대한 새 파티션 맵 ID를 확보하고, 이를 SYSCAT.PARTITIONMAPS 카탈로그 뷰에 삽입합니다.
2. 새 파티션 맵 ID의 노드 그룹에 대한 SYSCAT.NODEGROUPS 카탈로그 뷰의 REBALANCE_PMAP_ID 컬럼을 갱신합니다.
3. 모든 새 데이터베이스 파티션을 SYSCAT.NODEGROUPDEF 카탈로그 뷰에 추가합니다.
4. 삭제되어야 하는 모든 데이터베이스 파티션에 대해 SYSCAT.NODEGROUPDEF 카탈로그 뷰에서 IN_USE 컬럼을 'D'로 설정합니다.
5. 카탈로그 갱신사항에 대한 COMMIT를 수행합니다.
6. 모든 새 데이터베이스 파티션에 대한 데이터베이스 파일을 작성합니다.
7. 노드 그룹의 모든 테이블에 대해 각 테이블마다 데이터를 재분산합니다. 자세한 내용은 359 페이지의 『데이터가 테이블에 재분산되는 과정』에 설명되어 있습니다.
8. 데이터베이스 파일을 삭제하고, 이전에 삭제가 표시된 데이터베이스 파티션에 대한 SYSCAT.NODEGROUPDEF 카탈로그 뷰에서 항목을 삭제합니다.

9. PMAP_ID를 REBALANCE_PMAP_ID 값으로 설정하고 REBALANCE_PMAP_ID를 널(NULL)로 설정하기 위해 SYSCAT.NODEGROUPS 카탈로그 뷰에 있는 노드 그룹 레코드를 갱신합니다.
10. 이전의 파티션 맵을 SYSCAT.PARTITIONMAPS 카탈로그 뷰에서 삭제합니다.
11. 모든 변경사항에 대해 COMMIT를 수행합니다.

데이터가 테이블에 재분산되는 과정

테이블에서 데이터를 재분산할 때 유틸리티는 다음을 수행합니다.

1. SYSTABLES 카탈로그 테이블에 있는 이 테이블의 행을 잠급니다.
2. 이 테이블과 관련된 모든 패키지를 무효로 합니다. 테이블에서 재분산이 이루어지고 있기 때문에 이 테이블과 연관된 파티션 맵 ID는 변경됩니다. 이전 패키지는 무효로 되기 때문에, 컴파일러는 이 테이블에 대한 새 파티션 정보를 확보하여 이에 따라 패키지를 생성해야 합니다.
3. 테이블을 독점 모드로 잠급니다.
4. DELETE와 INSERT를 통해 테이블의 데이터를 재분산합니다.
5. 재분산 조작이 성공하면 다음을 수행합니다.
   a. 테이블에 대해 COMMIT를 발행합니다.
   b. 노드 그룹의 다음 테이블 작업을 계속합니다.

   테이블이 완전히 재분산되기 전에 조작이 실패하면 유틸리티는 다음을 수행합니다.
   a. 테이블의 갱신사항에 대해 ROLLBACK을 발행합니다.
   b. 전체 재분산 조작을 종료한 후 오류를 리턴합니다.

데이터를 분산할 때 로그 공간 요구사항을 측정하는 것은 중요합니다. 로그는 데이터가 재분산될 각 데이터베이스 파티션에서 INSERT 및 DELET 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, 새 데이터베이스 파티션 수에 대한 현재 데이터베이스 파티션 비율은 INSERT 및 DELETE 조작 횟수를 판별하는 데 도움이 됩니다.

예를 들어, 네 개의 데이터베이스 파티션에서 다섯 개의 데이터베이스 파티션으로 이동할 경우, 네 개의 원래 데이터베이스 파티션에서 대략 20%의 데이터가 새 데이터베이스 파티션으로 이동됩니다. 즉, 네 개의 원래 데이터베이스 파티션 각각에서는 각각의 데이터베이스 파티션에서 총 데이터 양을 기초로 20%의 DELETE 조작이 발생됩니다. 새 데이터베이스 파티션에서는 모든 INSERT 조작이 발생됩니다. 즉, 네 개의 원래 데이터베이스 파티션 모두에서의 DELETE 조작 수와 같습니다.

위의 예에서는 데이터가 일정하게 분산된다고 가정합니다. 파티션 키에 많은 널(NULL) 값이 있는 경우에서처럼 데이터가 일정하지 않게 분산되는 경우도 있을 수 있습니다. 이러한 경우, 이러한 모든 행은 이전 파티션 스킴 아래의 하나의 데이터베이스 파티션에서 그리고 새 파티션 스킴 아래의 다른 데이터베이스 파티션에서 종료됩니다. 결과적으로, 두 개의 데이터베이스 파티션에 필요로 로그 공간 양이 증가하여 일정한 분산을 가정할 때 계산되는 양을 초과하게 됩니다.

실제 계산을 수행할 경우, 변경 백분율(20%와 같은)에 가장 큰 테이블의 *크기*를 곱해야 합니다. 각 테이블의 재분산은 단일 트랜잭션으로 수행되므로 이렇게 해야 합니다.

주: 그러나 가장 큰 테이블은 일정하게 분산되지만, 두 번째로 큰 테이블(예를 들어)에는 현저하게 팽창된 하나 이상의 데이터베이스 파티션이 있을 수도 있습니다. 그러한 경우, 두 번째 테이블을 사용하고 가장 큰 테이블은 사용하지 않도록 해야 합니다.

일단 데이터베이스 파티션에서 삽입되거나 삭제될 최대 데이터 양을 계산하면 그 수치를 두 배로 하여 사용 중인 로그의 최대 *크기*를 결정하십시오. 이것이 사용 중인 로그 한계인 32GB를 초과할 경우, 데이터 재분산은 단계별로 수행해야 합니다. 『makepmap』이라고 하는 유틸리티가 있는데, 이를 사용하여 각 단계마다 하나씩, 일련의 목표 파티션 맵을 생성할 수 있습니다.

재분산 오류로부터 복구

재분산 조작 실행을 시작한 후에 하나의 파일이 sqllib 디렉토리의 redist 서브디렉토리에 기록됩니다. 이 상태 파일은 데이터베이스 파티션에서 수행되는 모든 조작, 재분산된 테이블 이름, 조작의 완료 상태를 나열합니다. 테이블에서 재분산할 수 없을 경우, 테이블 이름과 적용 가능한 SQLCODE가 파일에 나열됩니다. 잘못된 입력 매개변수 때문에 재분산 조작을 시작할 수 없는 경우, 이 파일은 기록되지 않고 SQLCODE는 리턴됩니다.

이 파일에는 다음과 같은 이름 지정 규칙이 있습니다.

databasename.nodegroupname.timestamp (UNIX 플랫폼용)
databasename\nodegroupname\date\time (비 UNIX 플랫폼용)

주: UNIX 이외의 플랫폼에서는 nodegroupname의 처음 8바이트만이 사용됩니다.

데이터 재분산 조작에 실패할 경우, 일부 테이블은 재분산되고 나머지 테이블은 재분산되지 않는 현상이 일어날 수 있습니다. 한 번에 한 테이블씩 데이터 재분산이 이루어지기 때문입니다. 복구 방법에는 두 가지 옵션이 있습니다.

- CONTINUE 옵션을 사용하여 나머지 테이블에 재분산 조작을 계속합니다.
- ROLLBACK 옵션을 사용하여 재분산의 실행 취소하고 재분산된 테이블을 원래의 상태로 복원합니다. 구간 복원 조작을 하는 데에는 원래의 재분산 조작에 걸린 만큼의 시간이 소요됩니다.

두 옵션 중 하나를 사용하려면 이전의 데이터 재분산 조작이 실패하여 SYSNODEGROUPS 카탈로그 테이블의 REBALANCE_PMID 컬럼이 널(NULL)이 아닌 값으로 설정되어야 합니다.

이 파일을 실수로 삭제한 경우에도 CONTINUE 조작을 계속 시도할 수 있습니다.

데이터 재분산 및 기타 조작

유틸리티가 수행 중인 동안 노드 그룹의 다른 오브젝트에서 다음 조작을 수행할 수 있습니다. 그러나 재분산 작업이 진행되고 있는 테이블에서 이 조작을 수행하지 마십시오. 다음을 수행할 수 있습니다.

- 다른 테이블의 색인을 작성합니다. CREATE INDEX문은 해당 테이블의 파티션 맵을 사용합니다.
- 다른 테이블을 삭제합니다. DROP TABLE문은 해당 테이블의 파티션 맵을 사용합니다.
- 다른 테이블의 색인을 삭제합니다. DROP INDEX문은 해당 테이블의 파티션 맵을 사용합니다.
- 다른 테이블을 조회합니다.
- 다른 테이블을 갱신합니다.
- 노드 그룹에 정의된 테이블 공간에서 새 테이블을 작성합니다. CREATE TABLE 문은 목표 파티션 맵을 사용합니다.
- 노드 그룹에 테이블 공간을 작성합니다.

유틸리티가 수행 중일 때에는 다음 조작을 수행할 수 없습니다.

- 노드 그룹에서의 또 다른 재분산 조작
- 노드 그룹의 테이블에 대한 ALTER TABLE문 사용
- 노드 그룹 삭제
- 노드 그룹 변경

데이터 재분산 후속 조치

노드 그룹에서 데이터 재분산을 완료한 후에는 RUNSTATS를 수행하여 재분산된 테이블과 연관된 통계를 갱신하는 것이 가장 좋습니다.

RUNSTATS에 대한 자세한 내용은 *Command Reference* 매뉴얼을 참조하십시오.

제12장 벤치마크 테스팅

벤치마킹은 일반적으로 응용프로그램 개발 주기에 포함되는 과정입니다. 이는 응용프로그램 개발자와 데이터베이스 관리자(DBA)의 협력하에 이루어지는 작업이며, 응용프로그램을 대상으로 품질을 결정하고 성능을 향상시키기 위하여 수행하는 작업입니다. 응용프로그램 코드가 최대한 효율적으로 작성되면 응용프로그램의 요구사항을 충족시키기 위해 데이터베이스와 데이터베이스 관리 프로그램 구성 매개변수를 조정함으로써 성능을 더욱 향상시킬 수 있습니다.

벤치마킹에는 몇 가지 종류가 있습니다. *초당 트랜잭션* 벤치마킹 기법은 일정한 제한된 테스트 조건하에서 데이터베이스 관리 프로그램의 처리 능력을 판별합니다. *응용프로그램* 벤치마킹 기법은 위와 같은 처리 능력을 테스트하나, 해당 응용프로그램이 수행될 환경과 좀더 유사한 조건에서 수행됩니다. 구성 매개변수를 조정하는 목적으로 수행되는 벤치마킹은 이러한 『실제의』 조건하에서, 해당 응용프로그램이 최대한 효율적으로 수행될 때까지, 다양한 매개변수의 값으로 응용프로그램으로부터의 SQL을 반복적으로 실행시킵니다.

이 절에서 설명된 벤치마킹 기법은 구성 매개변수의 조정을 중심으로 개발되었습니다. 그러나 성능에 영향을 미칠 수 있는 다음 요소를 조정하는 데에도 활용할 수 있습니다.

- SQL문
- 색인
- 테이블 공간 구성
- 응용프로그램 코드
- 하드웨어 구성

벤치마킹은 데이터베이스 관리 프로그램이 여러 가지 다양한 조건하에서 어떻게 반응하는지를 이해하는 데 도움이 됩니다. 교착 상태 처리 기능, 유틸리티의 성능, 데이터를 로드하는 데 사용되는 여러 가지 방법, 사용자가 증가로 인한 트랜잭션율의 특성 등을 테스트하고 새 릴리스의 제품을 사용하는 것이 응용프로그램에 어떠한 영향을 미치는지까지도 테스트할 수 있는 시나리오를 작성할 수 있습니다.

© Copyright IBM Corp. 1993, 2001

다음과 같은 주제가 제공됩니다.

- 『벤치마크 테스팅 방법론』
- 365 페이지의 『벤치마크 테스팅 준비』
- 367 페이지의 『벤치마크 프로그램 작성』
- 374 페이지의 『벤치마크 테스팅 실행』

벤치마크 테스팅 방법론

이 벤치마킹 기법은 과학적인 방법에 근거한 것입니다. 동일한 조건에서 동일한 테스트가 실행되어 비교 가능한 결과를 산출하는 반복 가능한 환경이 조성됩니다.

벤치마킹은 일반 환경에서 테스트 응용프로그램을 수행하여 시작할 수도 있습니다. 성능에 관련된 문제 영역이 좁혀지게 되면 테스트 및 검토할 대상이 되는 기능의 범위를 제한하기 위해 특수화된 테스트 케이스를 작성할 수 있습니다. 특수화된 테스트 케이스에서 좋은 테스트 결과를 얻기 위해 반드시 응용프로그램 전체를 에뮬레이트할 필요는 없습니다. 단순한 측정으로 시작하여 확인이 되면 복잡도를 높혀 가십시오.

바람직한 벤치마킹(또는 측정)은 다음과 같은 특성을 갖습니다.

- 모든 테스트는 반복 수행이 가능해야 합니다.
- 특정 테스트를 반복해서 수행하려고 할 때에는 반드시 동일한 시스템 상태에서 시작해야 합니다.
- (시나리오에서 시스템상에 기타 다른 수행 작업이 있다고 명시한 경우가 아니면) 시스템에 테스트하려고 하는 기능 이외의 다른 기능 또는 응용프로그램이 있어서는 안됩니다.

  주: 수행이 시작된 응용프로그램은 아무리 최소화되어 있거나 유휴 상태라고 하더라도 메모리를 사용합니다. 따라서 매번 적용할 때마다 메모리와 보조 기억장치간에 데이터를 이송할 때 벤치마킹 결과를 왜곡시키거나 반복성의 규칙을 위반할 가능성이 높아집니다.
- 벤치마킹에 사용되는 하드웨어와 소프트웨어는 제품 환경에서 사용되는 것과 같은 것이어야 합니다.

다른 모든 벤치마킹과 마찬가지로 시나리오가 작성된 다음 여러 번 수행해야 합니다. 수행할 때마다 따르는 핵심 정보를 수집하는 것은 응용프로그램과 데이터베이스의 성능을 모두 향상시키는 데 필요한 변경을 결정하는 데 있어서 매우 중요합니다.

벤치마크 테스팅 준비

성능에 대한 벤치마킹이 시작되기 전에 응용프로그램 데이터베이스의 논리 설계가 완료되어야 합니다. 테이블, 뷰 및 색인 등을 설정하고 구현해야 합니다. 테이블은 정규화되어야 하고, 응용프로그램 패키지는 바인드되어야 하며, 테이블에 실제 데이터가 들어 있어야 합니다.

데이터베이스의 최종적인 실제 설계를 결정해야 합니다. 데이터베이스 관리 프로그램 오브젝트가 최종적으로 결정된 디스크에 위치하고, 로그 파일의 크기가 조정되고, 작업 파일 및 백업 위치가 결정되며, 백업 프로시듀어가 테스트되어야 합니다. 또한 응용프로그램 패키지에 있어서는 성능에 영향을 미치는 행 블로킹 등의 옵션이 가능할 때 작동될 수 있는지 등을 확인해야 합니다.

벤치마크 프로그램을 작성할 만한 응용프로그램과 테스트를 해낼 수 있는 수준이면 된다는 것입니다(367 페이지의 『벤치마크 프로그램 작성』 참조). 응용프로그램의 실제적인 한계가 벤치마크 테스팅을 하는 동안에 드러날 수 있습니다. 그러나 여기에서 설명하고 있는 벤치마크의 목적은 성능을 평가하는 것이지, 결함이나 이상 종료 현상을 발견하고자 하는 것은 아닙니다.

벤치마킹 테스트 프로그램은 가능한 한 정확한 최종 제품 환경의 연출하에서, 동일한 메모리와 디스크 구성을 갖는 같은 서버 모델에 대해 수행되는 것이 이상적입니다. 특히, 해당 응용프로그램을 사용할 사용자가 많고 데이터의 양이 많은 경우는 더욱 중요합니다. 운영 체제 및 벤치마크에서 직접적으로 사용될 통신 설비와 파일 관련 설비도 역시 조정되어야 합니다.

또한 실제 수행될 데이터베이스 크기를 가지고 벤치마크를 수행해야 합니다. 각각의 SQL문은 실제 수행될 때와 같은 양의 데이터를 리턴해야 하고 실제와 유사한 만큼의 정렬 작업도 거쳐야 합니다. 이렇게 함으로써, 응용프로그램이 사용하는 일반적인 메모리의 요구사항도 파악할 수 있습니다.

벤치마킹 대상이 될 SQL문의 유형은 아래에서 설명하는 바와 같이 *대표적인* 명령문이거나 *최악의 경우를 고려한* 명령문으로 구성할 수 있습니다.

대표적인 SQL

대표적인 SQL은 벤치마크되는 응용프로그램에서 일반 작업에 사용될 명령문을 말합니다. 어떤 명령문을 선택할 것인지는 응용프로그램의 성격에 따라 달라집니다. 예를 들어, 데이터 입력 응용프로그램은 INSERT문을 테스트하고, 은행의 거래용 응용프로그램에서는 FETCH, UPDATE, 몇몇 INSERT를 테스트해 봅니다. 이렇게 선택된 명령문에 의해 처리될 데이터의 양과 수행 횟수는 평균치로 간주됩니다. 처리된 데이터의 양이 너무 많은 경우, 일반 SQL문이더라도 *최악의 경우를 고려한* SQL 범주에 속하게 됩니다.

최악의 경우를 고려한 SQL

이 범주에 속하는 명령문은 다음과 같습니다.

- 자주 실행되는 명령문
- 처리할 데이터의 양이 많은 명령문
- 처리 시간이 중요한 명령문

  예를 들어, 고객으로부터 전화가 걸려와서, 고객이 기다리고 있는 동안에 명령문이 수행되어 데이터를 검색하고, 고객 정보를 갱신해야 하는 응용프로그램의 경우
- 많은 수의 테이블이 조인되어야 하거나 응용프로그램의 성격상 복잡한 구조의 명령문

  예를 들어, 은행 응용프로그램에서 고객의 다양한 계좌에 대한 월별 데이터를 조합하여 검색하고자 하는 명령문을 들 수 있습니다. 공통 테이블로부터 고객의 주소와 계좌 번호를 검색할 수 있지만, 필요한 거래 내역을 처리하기 위해서는 여러 개의 테이블이 조인되어야 합니다. 하나의 계좌를 처리하기 위해 필요한 작업량에 같은 시간 내에 처리해야 할 수많은 계좌 수를 곱하면 잠재적으로 절약되는 시간이 계산되는데, 이것으로 성능에 대한 요구사항을 알 수 있을 것입니다.
- 액세스 경로가 좋지 못한 명령문. 예를 들어, 자주 실행되지 않는 명령문 또는 테이블에 대한 색인이 없이 수행되는 명령문
- 수행 경과 시간이 긴 명령문

- 응용프로그램을 초기화할 때에만 수행되지만 자원을 불균형적으로 필요로 하는 명령문

  예를 들면, 하루 동안에 처리되어야 하는 계좌의 목록을 생성하는 응용프로그램이 있습니다. 응용프로그램을 시작할 때 맨 처음 수행되는 주요 SQL은 7가지 방법으로 테이블을 조인하여 응용프로그램 사용자가 맡고 있는 모든 계좌의 긴 목록을 생성합니다. 이러한 명령문은 하루에 몇 번만 실행되지만, 제대로 조정되지 않은 경우 한 번 수행될 때마다 몇 분씩 걸리기도 합니다.

벤치마크 프로그램 작성

벤치마크 프로그램을 설계하고 작성할 때에는 고려해야 할 여러 가지 요소가 있습니다. 프로그램의 주된 목적이 사용자 응용프로그램을 모의 실험하는 것이므로, 전반적인 프로그램의 구조는 다양할 수 있습니다. 응용프로그램 전체를 벤치마크로서 사용하고, 단지 분석될 SQL문의 시간을 측정하는 수단만을 도입할 수도 있습니다. 크거나 복잡한 응용프로그램에 있어서, 중요한 명령문을 가지고 있는 블록을 포함하는 것이 더 실제적일 수도 있습니다.

특정 SQL문을 테스트하는 또 다른 방법은 이 명령문만을 필요한 CONNECT, PREPARE, OPEN 및 다른 명령문과 시간측정(timing) 메커니즘과 함께 벤치마크 프로그램에 포함시키는 것입니다.

고려해야 할 또 다른 인수는 사용할 벤치마크의 유형입니다. 한 가지 방법은 SQL문 세트를 일정 시간 간격을 두고 반복적으로 실행시키는 것입니다. 실행된 명령문의 수와 이러한 시간 간격의 비율이 응용프로그램에 대한 처리량을 제공합니다. 또 다른 방법은 개별적인 SQL문을 실행시키는 데 필요한 시간을 측정하는 것입니다.

벤치마크 프로그램의 유형에 관계없이 개별적인 SQL문이거나 응용프로그램 전체를 대상으로 하거나 경과 시간을 계산하기 위해서는 효율적인 시간 측정 시스템이 필요합니다. 각각의 SQL문이 개별적으로 실행될 수 있는 응용프로그램 환경을 만드는 경우에는 CONNECT, PREPARE 및 COMMIT문에 소요되는 시간을 고려하는 것이 중요합니다. 그러나 많은 수의 다양한 명령문을 테스트하는 프로그램의

경우에는 아마도 단 하나의 CONNECT나 COMMIT문이 필요할 것이므로, 각각의 명령문 실행에 소요되는 시간을 측정하는 것이 더욱 중요합니다.

각 조회의 경과 시간이 성능을 분석하는 데 중요한 요소인 반면에, 병목 현상을 반드시 밝혀낸다고 할 수는 없습니다. 예를 들어, CPU의 사용, 잠금 상태 및 버퍼 풀 I/O 등에 대한 정보를 통해 응용프로그램이 CPU의 전체 용량을 사용하기보다는 I/O 위주임을 알 수 있습니다. 벤치마크 프로그램은 필요한 경우 더 세부적인 분석을 위해 이러한 종류의 데이터를 얻을 수 있도록 해주어야 합니다.

모든 응용프로그램에서, 조회로부터 검색된 전체 행 세트를 출력 장치로 전송할 필요는 없습니다. 예를 들어, 어떤 응용프로그램에서는 전체 응답 세트를 다른 프로그램의 입력으로 사용하기도 합니다. (즉, 첫 번째 응용프로그램에서 보낸 어떠한 행도 출력되지 않습니다.) 화면으로 출력하기 위해 데이터를 형식화하는 것은 일반적으로 CPU의 비용이 많이 들기 때문에 사용자의 필요에 부합되지 않을 수도 있습니다. 정확한 시뮬레이션을 제공하려면 벤치마크 프로그램이 특정 응용프로그램을 어떻게 하는지 나타내어야 합니다. 행이 출력 장치로 전송되면 비효율적인 형식화에 CPU 시간 대부분을 사용하게 되므로, SQL문 자체를 처리하는 실제 성능이 제대로 나타나지 않을 수 있습니다.

db2batch 벤치마크 도구: 벤치마크 도구(db2batch)가 사용자의 인스턴스 sqllib 디렉토리의 bin 서브디렉토리에 제공됩니다. 이 도구는 벤치마크 프로그램을 작성하는 것에 대해서 위에서 언급한 많은 사항을 고려하고 있습니다. 이 도구는 SQL문을 일반 파일 또는 표준 입력 장치로부터 읽어들여 해당 명령문을 그때그때마다 설명하고 준비하여 결과를 리턴합니다. 출력 장치로 전송될 응답 세트의 행 수뿐만 아니라, 응답 세트 크기 자체를 조정할 수 있는 유연성도 제공합니다.

또한 경과 시간, CPU 및 버퍼 풀 사용 상황, 잠금 및 기타의 데이터베이스 모니터로부터 얻을 수 있는 성능에 관련된 통계 정보의 레벨을 지정할 수도 있습니다. SQL문 세트의 시간을 재는 중이라면, db2batch가 성능 결과를 요약하고, 산술 평균 및 기하 평균(geometric mean)도 제공합니다. 호출 구문 및 옵션에 대한 자세한 내용은 db2batch의 *Command Reference* 매뉴얼을 참조하십시오. 사용 가능한 구문 및 옵션에 대한 명령행에 db2batch -h를 입력할 수도 있습니다.

다음은 db2batch가 db2batch.sql 입력 파일과 함께 사용될 수 있는 방법에 대한 예입니다.

```
-- db2batch.sql
-- ------------
--#SET PERF_DETAIL 3 ROWS_OUT 5
-- This query lists employees, the name of their department
-- and the number of activities to which they are assigned for
-- employees who are assigned to more than one activity less than
-- full-time.
--#COMMENT Query 1
select lastname, firstnme,
       deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
      emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2;
--#SET PERF_DETAIL 1 ROWS_OUT 5
--#COMMENT Query 2
select lastname, firstnme,
       deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
      emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2;
```

그림 31. 샘플 벤치마크 입력 파일: db2batch.sql

벤치마크 도구를 다음과 같이 호출하면

```
db2batch -d sample -f db2batch.sql
```

다음과 같은 출력이 생성됩니다.

```
--#SET PERF_DETAIL 3 ROWS_OUT 5
Query 1
Statement number: 1
select lastname, firstnme,
       deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2
```

그림 32. db2batch로부터의 샘플 출력(파트 1)

```
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
-------------------------------------------------------------------------
JEFFERSON        JAMES         ADMINISTRATION SYSTEMS                3
JOHNSON          SYBIL         ADMINISTRATION SYSTEMS                4
NICHOLLS         HEATHER       INFORMATION CENTER                    4
PEREZ            MARIA         ADMINISTRATION SYSTEMS                4
SMITH            DANIEL        ADMINISTRATION SYSTEMS                7
Number of rows retrieved is:        5
Number of rows sent to output is:   5
Elapsed Time is:           0.074      seconds
Locks held currently                        = 0
Lock escalations                            = 0
Total sorts                                 = 5
Total sort time (ms)                        = 0
Sort overflows                              = 0
Buffer pool data logical reads              = 13
Buffer pool data physical reads             = 5
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 3
Buffer pool index physical reads            = 0
Buffer pool index writes                    = 0
Total buffer pool read time (ms)            = 23
Total buffer pool write time (ms)           = 0
Asynchronous pool data page reads           = 0
Asynchronous pool data page writes          = 0
Asynchronous pool index page reads          = 0
Asynchronous pool index page writes         = 0
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
LSN Gap cleaner triggers                    = 0
Dirty page steal cleaner triggers           = 0
Dirty page threshold cleaner triggers       = 0
Direct reads                                = 8
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct read elapsed time (ms)               = 0
Direct write elapsed time (ms)              = 0
Rows selected                               = 5
Log pages read                              = 0
Log pages written                           = 0
Catalog cache lookups                       = 3
Catalog cache inserts                       = 3
Buffer pool data pages copied to ext storage    = 0
Buffer pool index pages copied to ext storage   = 0
Buffer pool data pages copied from ext storage  = 0
Buffer pool index pages copied from ext storage = 0
Total Agent CPU Time (seconds)              = 0.02
Post threshold sorts                        = 0
Piped sorts requested                       = 5
Piped sorts accepted                        = 5
```

그림 33. db2batch로부터의 샘플 출력(파트 1)

```
--#SET PERF_DETAIL 1 ROWS_OUT 5
Query 2
Statement number: 2
select lastname, firstnme,
       deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
------------------------------------------------------------------------
GEYER            JOHN          SUPPORT SERVICES                      2
GOUNOT           JASON         SOFTWARE SUPPORT                      2
HAAS             CHRISTINE     SPIFFY COMPUTER SERVICE DIV.          2
JONES            WILLIAM       MANUFACTURING SYSTEMS                 2
KWAN             SALLY         INFORMATION CENTER                    2
Number of rows retrieved is:        8
Number of rows sent to output is:   5
Elapsed Time is:           0.037      seconds
Summary of Results
==================
                 Elapsed             Agent CPU         Rows      Rows
Statement #      Time (s)            Time (s)          Fetched   Printed
1                     0.074               0.020          5         5
2                     0.037       Not Collected          8         5
Arith. mean      0.055
Geom.  mean      0.052
```

그림 34. db2batch로부터의 샘플 출력(파트 2)

위의 샘플 출력에는 데이터베이스 시스템 모니터가 리턴한 특정 데이터 요소도 포함되어 있습니다. 이러한 사항 및 모니터 요소에 대한 자세한 내용은 *시스템 모니터 안내 및 참조서* 매뉴얼을 참조하십시오.

다음 예(UNIX에서)에서는 요약 테이블만 생성됩니다.

```
db2batch -d sample -f db2batch.sql -r /dev/null,
```

-r 옵션을 사용하면 outfile1이 /dev/null로 대체되고 outfile2(요약 테이블만 포함)는 비게 되므로 db2batch는 출력을 화면으로 전송합니다.

```
Summary of Results
==================
                Elapsed              Agent CPU          Rows       Rows
Statement #     Time (s)             Time (s)           Fetched    Printed
1                    0.074                0.020           5          5
2                    0.037       Not Collected            8          5
Arith. mean     0.055
Geom.  mean     0.052
```

그림35. db2batch로부터의 샘플 출력 -- 요약 테이블만

이 벤치마킹 도구에는 CLI 옵션도 있습니다. 이 옵션으로 캐쉬 크기를 지정할 수 있습니다. 다음 예에서 db2batch는 명령문 30의 캐쉬 크기를 가진 CLI 모드에서 수행됩니다.

```
db2batch -d sample -f db2batch.sql -cli 30
```

이는 db2batch를 원격으로 수행 가능하게 합니다. 벤치마킹 도구의

```
-f <filename>
```

또는

```
-o <options>
```

명령 매개변수를 사용하면

- 이 매개변수보다 크게 설정된

  ```
  perf_detail
  ```

  및

  ```
  -p <perf_detail>
  ```

  제어 옵션은 (리턴된 성능 정보의 레벨 지정) 원격으로 수행될 때 지원되지 않습니다.
- 제어 옵션

  ```
  perf_detail
  ```

  및

  ```
  -p <perf_detail>
  ```

  제어 옵션은 (리턴된 성능 정보의 레벨 지정) 원격으로 수행될 때 Windows 3.x 또는 DOS 플랫폼용 DB2 Universal Database에 대해 유효하지 않습니다.

이 두 항목 이외에

perf_detail

및

-p <perf_detail>

제어 옵션 값이 지원되고 모든 DB2 Universal Database 플랫폼에 유효합니다.

벤치마크 테스팅 실행

데이터베이스 벤치마킹의 한 유형으로서 구성 매개변수를 선택하여 최고 이익에 도달할 때까지, 해당 매개변수에 대해 여러 가지 다른 값으로 테스트를 수행하는 것이 있습니다. 한 번의 테스트에서 매개변수에 동일한 값을 주고, 응용프로그램을 여러 번 반복시켜서(예: 20 또는 30회 정도) 평균 값을 산출하게 되는데, 이렇게 함으로써 매개변수를 변경했을 때의 효과를 더 잘 알 수 있습니다.

벤치마크를 수행할 때 첫 번째 반복(웜업 수행이라고 함)은 후속 반복(정상 수행이라고 함)과는 구분되어야 합니다. 이는 웜업 수행의 결과에 버퍼 풀 초기화와 같은 일부 시작 활동이 포함되므로 필요한 조치입니다. 결국, 웜업 수행은 정상 수행보다 많은 시간을 소요하게 됩니다. 웜업 수행으로부터의 정보가 실제적으로는 유효한 정보가 *될 수 있겠지만* 통계적으로는 적합하지 않습니다. 그러므로 특정 매개변수 값 세트에 대한 소요 시간이나 CPU 평균 값을 측정할 경우, 정상 수행 결과를 사용하십시오.

성능 구성 마법사를 사용하여 벤치마크 웜업 수행을 작성하는 것을 고려할 수 있습니다. 성능 구성 마법사 실행 중 나오는 질문은 벤치마크 활동 동안 정상 수행에 대해 환경 구성을 조정할 경우에 고려하여야 될 일부 사항에 대한 식견을 제공합니다. 성능 구성 마법사를 사용하려면 db2cc를 입력하여 제어 센터에 액세스하여 계속하십시오.

개별적인 조회를 가지고 벤치마킹을 하는 경우, 이전 조회의 잠재적 영향을 최소화하도록 신중해야 합니다. 이는 (사용자의 조회와 관련이 없는) 얼마간의 페이지를 읽어들여 버퍼 풀을 채워, 버퍼 풀의 내용을 비움으로써 가능합니다.

하나의 매개변수 값 세트에 대한 반복 수행이 끝나면 매개변수 값을 변경할 수 있습니다. 그러나 각각의 반복 수행 주기 사이에는 벤치마크 테스트 환경을 원래의 상태로 복원하기 위해 다음과 같은 타스크를 수행해야 합니다.

- 응용프로그램 데이터와 데이터베이스 관리 프로그램 통계치를 원래의 상태로 리턴합니다. 테스트에 대한 카탈로그 통계 값이 갱신된 경우에는 같은 값이 반복 수행시마다 사용되도록 하십시오. 도중에 테스트에 사용되는 데이터가 갱신되는 경우, 데이터 일관성이 유지되어야 합니다. 이것은 다음과 같은 방법으로 가능합니다.
  - RESTORE 유틸리티를 사용하여 전체의 데이터베이스를 복원합니다. 데이터베이스의 백업 사본은 이전의 상태에 있으면서 다음 번 테스트를 위해 준비되어야 합니다.
  - IMPORT나 LOAD 유틸리티를 사용하여 내보내기된 데이터의 사본을 복원합니다. 이 방법은 테스트 중에 값이 변경된 데이터만을 복원하는 방법입니다. REORG 및 RUNSTATS 유틸리티는 이러한 데이터를 가지고 있는 테이블과 색인에 대하여 수행되어야 합니다.
- 데이터베이스에 리바인드하여 응용프로그램을 원래의 상태로 리턴합니다.

다음은 OS/2에서 벤치마킹할 때 추가로 고려해야 할 사항입니다.

- 시나리오 수행 중에 페이징이 발생하는 경우, 반드시 SWAPPER.DAT를 원래의 상태로 리턴하십시오.
- 필요한 경우, 다시 반복 수행하기 위해 시스템을 재부팅하십시오.

벤치마크 프로그램은 각각의 테스트, 테스트 프로그램 반복 수행 주기, 명령문 번호 및 각 실행에 있어서 측정된 시간에 대한 식별자를 출력으로 가지고 있어야 합니다. 일련의 측정 작업 후에 나온 벤치마크 결과 요약은 다음과 같습니다.

```
Test      Iter.    Stmt    Timing          SQL Statement
Numbr     Numbr    Numbr   (hh:mm:ss.ss)
 002       05       01     00:00:01.34     CONNECT TO SAMPLE
 002       05       10     00:02:08.15     OPEN cursor_01
 002       05       15     00:00:00.24     FETCH cursor_01
 002       05       15     00:00:00.23     FETCH cursor_01
 002       05       15     00:00:00.28     FETCH cursor_01
 002       05       15     00:00:00.21     FETCH cursor_01
 002       05       15     00:00:00.20     FETCH cursor_01
 002       05       15     00:00:00.22     FETCH cursor_01
 002       05       15     00:00:00.22     FETCH cursor_01
 002       05       20     00:00:00.84     CLOSE cursor_01
 002       05       99     00:00:00.03     CONNECT RESET
```

그림 36. 벤치마크 샘플 결과

주: 위 보고서의 데이터는 단지 예일뿐입니다. 측정된 결과가 **아닙니다.**

이 보고서를 살펴보면 CONNECT(명령문 01)는 1.34초가 소요되고, OPEN CURSOR(명령문 10)은 2분 8.15초가 소요되며, FETCHES(명령문 15)는 최장 지연시간을 0.28초로 하여 7행을 리턴하며, CLOSE CURSOR(명령문 20)는 .84초가 소요되고, CONNECT RESET(명령문 99)는 .03초가 소요되었습니다.

나중에 데이터베이스 테이블이나 스프레드시트로 가져오기하여 더 미세한 통계적 분석을 할 수 있도록 벤치마크 결과를 컬럼 식별자가 있는 ASCII 형식으로 출력하는 것이 더욱 유용합니다.

벤치마크 보고서의 샘플 출력은 다음과 같습니다.

```
      PARAMETER        VALUES FOR EACH BENCHMARK TEST
      TEST NUMBER      001      002      003      004      005
      locklist         63       63       63       63       63
>>    buffpage         1000     1175     1250     1325     1400      <<
      maxappls         8        8        8        8        8
      applheapsz       48       48       48       48       48
      dbheap           128      128      128      128      128
      sortheap         256      256      256      256      256
      maxlocks         22       22       22       22       22
      stmtheap         1024     1024     1024     1024     1024
      SQL STMT         AVERAGE TIMINGS (seconds)
        01             01.34    01.34    01.35    01.35    01.36
        10             02.15    02.00    01.55    01.24    01.00
        15             00.22    00.22    00.22    00.22    00.22
        20             00.84    00.84    00.84    00.84    00.84
        99             00.03    00.03    00.03    00.03    00.03
```

그림 37. 벤치마크 샘플 시간 측정 보고서

주: 위 보고서의 데이터는 단지 예일뿐입니다. 측정된 결과가 **아닙니다.**

위의 예를 살펴보면 *buffpage* 매개변수를 변경하면 OPEN CURSOR 수행시 소요 시간이 2.15초에서 1.00초까지 차례로 줄어들고 있음을 알 수 있습니다. (*크기*(NPAGES)가 -1로 설정된 하나의 버퍼 풀만을 가정합니다. 이것은 버퍼 풀 *크기*가 *buffpage* 매개변수에 의해 제어되는 것을 의미합니다.)

요약하면 데이터베이스 응용프로그램에 대한 벤치마크 테스트를 수행하기 위해서는 다음과 같은 단계/반복 작업이 있어야 합니다.

1단계 다음과 같은 경우를 제외하고는 데이터베이스 및 데이터베이스 관리 프로그램의 매개변수를 **기본값**으로 두십시오.

- 테스트의 작업량 및 목적에 중요한 의미를 갖는 매개변수(벤치마크 테스트를 실행하여 매개변수를 전부 조정하기에는 시간이 없는 경우가 대부분이므로, 매개변수 몇 개를 임의로 골라내어 시작하게 됩니다.)
- 응용프로그램의 단위 및 시스템 테스트시에 결정해야 할 로그 *크기*(475 페이지의 『로그 파일의 *크기*(logfilsiz)』 참조)
- 응용프로그램을 수행하기 위해 변경해야 할 매개변수(즉, 메모리 부족과 같은 이벤트로 SQL의 오류 리턴 코드가 돌아오는 것을 방지하기 위해 필요한 변경)

초기 상황에 대해 사용자 스스로의 반복 수행 세트를 수행하여 평균 소요 시간과 CPU를 계산하십시오.

2단계 테스트할 매개변수를 단 하나만 선택하여 값을 변경하십시오.

3단계 또 다른 반복 수행 세트를 가지고 수행하여 평균 소요 시간과 CPU를 계산하십시오.

4단계 벤치마크 테스트의 결과에 따라 아래의 작업 중 하나를 수행하십시오.

- 성능이 향상된 경우, 동일한 매개변수에 대한 값을 변경시키고 3단계로 가십시오. 최대의 효과를 볼 때까지 매개변수 값을 계속 변경하십시오.
- 성능이 오히려 저하되거나 별 변화가 없는 경우, 매개변수 값을 이전 값으로 리턴하고 2단계로 리턴한 후 새 매개변수를 선택하십시오. 매개변수가 모두 테스트될 때까지 이 과정을 반복하십시오.

  주: 성능 테스트 결과를 그래프로 나타내려 할 경우에는 그래프의 선이 평평해지거나 하강하기 시작하는 지점을 찾습니다.

벤치마크 테스트에 도움이 될 드라이버 프로그램을 작성할 수도 있습니다. 이 드라이버 프로그램은 REXX 또는 UNIX 시스템에서 사용하는 쉘 스크립트와 같은 언어로 작성할 수 있습니다.

이 드라이버 프로그램은 벤치마크 프로그램을 실행하며 적절한 매개변수를 전달하며 반복 수행을 하도록 하고, 환경을 일관성 있는 상태로 복원시켜 주고, 다음 테스트를 위해 새 매개변수 값을 할당하고 테스트 결과를 수집 및 조정하는 역할을 합니다. 이 드라이버 프로그램은 융통성이 있어서 벤치마크 테스트 전체 세트의 수행에 사용되어, 결과를 분석하고 최종적인 최적의 매개변수 값에 대한 보고서를 제공할 수 있습니다.

제13장 DB2 구성

구성 매개변수란 데이터베이스 또는 데이터베이스 관리 시스템의 조작 특성에 영향을 미치는 값을 말합니다.

데이터베이스 관리 프로그램 구성 매개변수는 서버 및 클라이언트에 존재합니다. 그러나 클라이언트에는 특정 데이터베이스 관리 프로그램 구성 매개변수만을 설정할 수 있습니다. 이 매개변수는 서버에 설정될 수 있는 데이터베이스 관리 구성 매개변수의 부속 집합입니다. 사용 중인 DB2 Universal Database 제품의 유형에 따라 구성 매개변수에 관련된 특정 문제가 있습니다. 예를 들어, DB2 Extended Enterprise Edition에서 하나의 데이터베이스 관리 프로그램 구성 파일을 인스턴스 내의 모든 데이터베이스 파티션 서버가 공유할 수 있습니다. 그리고 각각의 데이터베이스 파티션은 그 자체의 데이터베이스 구성 파일을 갖고 있습니다.

DB2는 광범위한 조정 및 구성 매개변수를 활용할 수 있도록 설계되었습니다. 이러한 매개변수는 두 범주로 나뉩니다.

- 381 페이지의 『데이터베이스 관리 프로그램 매개변수』
- 388 페이지의 『데이터베이스 매개변수』

개별적인 매개변수에 대한 설명뿐만 아니라 구성 매개변수에 심각한 영향을 주는 다음 주제에 대해 설명합니다.

- 380 페이지의 『구성 매개변수 조정』
- 394 페이지의 『매개변수의 기능별 세부사항』(각 함수 영역마다 자체의 구성 매개변수 목록이 있습니다.)
- 575 페이지의 『부록A. DB2 레지스트리 및 환경 변수』

  성능과 관련된 구성 매개변수 외에도 사용을 고려 중인 특정 플랫폼에 대한 성능 관련 환경 및 레지스트리 변수가 있을 수 있습니다.
- 273 페이지의 『제8장 수행시의 성능』
- 363 페이지의 『제12장 벤치마크 테스팅』

© Copyright IBM Corp. 1993, 2001

383 페이지의 표17과 390 페이지의 표19의 모든 매개변수 요약을 검토하고, 작업 환경에 가장 큰 혜택을 가져다줄 매개변수에 대한 내용과 조정에 초점을 두어야 합니다.

구성 매개변수 조정

데이터베이스 관리 프로그램이 매개변수의 기본값으로 할당한 디스크 공간 및 메모리는 사용자의 요구를 충족시키기에는 충분할 수 있습니다. 그러나 일부 경우에는 기본값을 사용해서는 최대의 성능 효과를 얻지 못할 수도 있습니다.

기본값은 상대적으로 메모리가 작은 시스템을 위주로 설정되고 데이터베이스 서버 전용이기 때문에, 다음과 같은 사용자 환경에서는 값을 수정해야 할 수도 있습니다.

- 규모가 큰 데이터베이스
- 연결 수가 많은 경우
- 특정 응용프로그램에 있어서 우수한 성능이 요구되는 경우
- 고유 조회, 트랜잭션 로드 또는 유형
- 다양한 시스템 구성 또는 용도

각 트랜잭션 처리 환경은 한 가지 이상의 고유 특성을 갖습니다. 이러한 차이점은 기본 구성 값을 사용하는 경우, 데이터베이스 관리 프로그램의 성능에 큰 영향을 미칠 수 있습니다. 이러한 이유로 사용자 자신의 환경에 적합한 구성으로 조정하는 것이 좋습니다.

다양한 종류의 응용프로그램과 사용자는 응답시간에 대한 다양한 요구사항과 기대치를 가지고 있기 마련입니다. 응용프로그램도 단순한 데이터 입력 화면으로부터, 하나의 작업에 수십 개의 테이블에 접근해야 하는 여러 개의 복잡한 SQL문을 사용하는 전략적인, 즉 계획을 세워서 접근해야 하는 응용프로그램까지 범위가 매우 넓습니다. 이를 테면, 고객 전화 응답 서비스 응용프로그램과 보고서 일괄 출력 응용프로그램의 응답시간 요구사항은 차이가 매우 큽니다.

기타 관련 주제는 구성 매개변수를 조정하기 위해 응용프로그램을 벤치마크하는 데 도움이 될 수 있습니다.

- 『데이터베이스 관리 프로그램 매개변수』
- 388 페이지의 『데이터베이스 매개변수』
- 394 페이지의 『매개변수의 기능별 세부사항』(함수 영역마다 자체적인 구성 매개변수 목록이 있습니다.)
- 273 페이지의 『제8장 수행시의 성능』
- 363 페이지의 『제12장 벤치마크 테스팅』
- *시스템 모니터 안내 및 참조서*의 데이터베이스 시스템 모니터 요소 설명

데이터베이스 관리 프로그램 매개변수

데이터베이스 관리 프로그램 매개변수는 db2systm이라는 파일에 저장되어 있습니다. 이 파일은 데이터베이스 관리 프로그램의 인스턴스가 작성될 때 작성됩니다. UNIX 기반 환경에서 이 파일은 데이터베이스 관리 프로그램 인스턴스에 대한 sqllib 서브디렉토리에 들어 있습니다. 다른 모든 환경에서 이 파일의 기본 위치는 sqllib 디렉토리의 인스턴스 서브디렉토리입니다. DB2INSTPROF 변수가 설정될 경우, 파일은 DB2INSTPROF 변수에 의해 지정된 디렉토리의 instance 서브디렉토리에 존재합니다.

파티션된 데이터베이스 환경에서 이 파일은 모든 데이터베이스 파티션 서버가 같은 파일에 대한 액세스 권한을 갖도록 공유 파일 시스템에 상주해야 합니다. 데이터베이스 관리 프로그램의 구성은 모든 데이터베이스 파티션 서버에서 동일합니다.

대부분의 매개변수는 하나의 데이터베이스 관리 프로그램 인스턴스에 할당될 시스템 자원의 양에 영향을 미치거나, 환경 요건에 따라 데이터베이스 관리 프로그램 설정과 다양한 통신 서브시스템을 구성합니다. 또한 단지 정보를 전달할 목적만으로 값이 변경되지 않는 다른 매개변수도 있습니다. 이들 매개변수는 데이터베이스 관리 프로그램 인스턴스하에 저장되어 있는 모든 데이터베이스와 별개로 전체적으로 적용할 수 있습니다.

db2systm 파일은 직접 편집할 수 없습니다. 제공되는 API를 사용하거나 API를 호출하는 도구를 사용하여 내용을 보거나 변경할 수 있습니다.

주의사항: 제품에 명시되지 않은 다른 방법을 사용하여 위의 파일을 편집할 경우, 시스템이 사용 불가능해질 수 있습니다. DB2에서 언급하고 지원하는 것이 아닌 다른 방법을 사용하여 이 파일을 변경하지 **않도록 해야** 합니다.

다음 방법 중 하나를 사용하여 데이터베이스 관리 프로그램 구성 매개변수를 재설정하고 갱신하며 볼 수 있습니다.

- DB2 제어 센터 사용. DB2 제어 센터는 구성 인스턴스 노트북을 제공하며, 이를 사용하여 클라이언트나 서버에서 데이터베이스 관리 프로그램 구성 매개변수를 설정할 수 있습니다. DB2 제어 센터는 서버에서 구성 매개변수의 값을 변경할 수 있도록 성능 구성 마법사를 제공합니다. 이 마법사는 질문에 대해 사용자가 제공한 워크로드 및 데이터베이스에 대해 수행되는 트랜잭션 유형과 같은 응답에 따라 매개변수에 값을 생성합니다. 이 인터페이스의 사용법에 대한 자세한 내용은 제어 센터에서 사용 가능한 온라인 도움말을 참조하십시오.
- 명령행 처리기 사용. 설정값을 변경하는 데 사용하는 명령을 빠르고 편리하게 입력할 수 있습니다. 다음 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.
  - GET DATABASE MANAGER CONFIGURATION(또는 GET DBM CFG)
  - UPDATE DATABASE MANAGER CONFIGURATION(또는 UPDATE DBM CFG)
  - RESET DATABASE MANAGER CONFIGURATION(또는 RESET DBM CFG)
- API(API) 사용. 응용프로그램에서 API가 쉽게 호출될 수 있습니다. 자세한 내용은 *Administrative API Reference*를 참조하십시오.
- 클라이언트 구성 지원 프로그램 사용. 클라이언트에 데이터베이스 관리 프로그램 구성 매개변수를 설정하는 데에는 클라이언트 구성 지원 프로그램만을 사용할 수 있습니다.

매개변수를 변경한 후에 데이터베이스 관리 프로그램은 새 매개변수 값이 적용되도록 중지(`db2stop`)된 후 재시작(`db2start`)되어야 합니다. 클라이언트의 경우, 데이터베이스 관리 프로그램 구성 매개변수의 변경사항은 클라이언트가 서버에 연결

된 다음에 효과가 나타납니다. 새로 변경된 매개변수 값이 즉시 효력을 발휘하지는 못하지만, 매개변수 설정은 항상 가장 최근에 갱신된 값으로 되어 있습니다.

주: *dft_monswitches* 매개변수의 값을 갱신할 경우, 데이터베이스 관리 프로그램을 다시 시작하지 않아도 됩니다. 이 매개변수는 값을 변경할 때 자동으로 갱신됩니다.

데이터베이스 관리 프로그램 구성 매개변수 요약

다음 표는 데이터베이스 서버에 대한 데이터베이스 관리 프로그램 구성 파일의 매개변수를 나열하고 있습니다. 데이터베이스 관리 프로그램 구성 매개변수를 변경할 경우, 각각의 매개변수에 대한 자세한 정보를 참고하십시오. 기본값 및 특정 운영 환경 정보가 각 매개변수에 대한 설명의 일부로서 주어집니다.

다음 표의 『성능 영향』 컬럼은 시스템 성능에 영향을 미치는 각각의 매개변수의 상대적 중요도를 나타냅니다. 이 컬럼은 모든 환경에 정확하게 적용할 수 없습니다. 이 정보는 일반 정보로 보아야 합니다.

- **높음** - 매개변수가 성능에 상당한 영향을 미칠 수 있다는 것을 나타냅니다. 이러한 값은 신중하게 결정해야 하며, 경우에 따라 기본값을 받아들여야 합니다.
- **중간** - 매개변수가 성능에 어느 정도의 영향을 미칠 수 있다는 것을 나타냅니다. 사용자의 특정 작업 환경과 요구사항에 따라 이 매개변수를 조정하는 데 얼마나 많은 노력을 기울여야 하는지가 결정됩니다.
- **낮음** - 매개변수가 성능에 일반적인 수준 이하의 영향 또는 미비한 영향을 미칠 수 있다는 것을 나타냅니다.
- **없음** - 매개변수가 성능에 직접적인 영향을 미치지 않는다는 것을 나타냅니다. 성능을 향상시키기 위해서는 이 매개변수를 조정할 필요는 없다 하더라도, 통신 지원 등 시스템 구성의 여러 다른 사항에 대해 매우 중요한 매개변수일 수 있습니다.

표 17. 구성 가능한 데이터베이스 관리 프로그램 구성 매개변수

매개변수	성능 영향	추가 정보
agentpri	높음	461 페이지의 『에이전트의 우선순위(agentpri)』
agent_stack_sz	낮음	424 페이지의 『에이전트 스택 크기(agent_stack_sz)』

표 17. 구성 가능한 데이터베이스 관리 프로그램 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
aslheapsz	높음	429 페이지의 『응용프로그램 지원 계층 힙 크기 (aslheapsz)』
audit_buf_sz	높음	440 페이지의 『감사 버퍼 크기(audit_buf_sz)』
authentication	낮음	565 페이지의 『인증 유형(authentication)』
backbufsz	중간	405 페이지의 『기본 백업 버퍼 크기(backbufsz)』
catalog_noauth	없음	567 페이지의 『권한 없이 허용되는 카탈로그화 (catalog_noauth)』
comm_bandwidth	중간	551 페이지의 『통신 대역폭(comm_bandwidth)』
conn_elapse	중간	536 페이지의 『연결 경과 시간(conn_elapse)』
cpuspeed	낮음(주 참조)	552 페이지의 『CPU 속도(cpuspeed)』
datalinks	낮음	509 페이지의 『데이터 링크 지원 사용(datalinks)』
dft_account_str	없음	558 페이지의 『기본 접미부 계정(dft_account_str)』
dft_client_adpt	없음	530 페이지의 『기본 클라이언트 어댑터 번호 (dft_client_adpt)』
dft_client_comm	없음	529 페이지의 『기본 클라이언트 통신 프로토콜 (dft_client_comm)』
dft_monswitches • *dft_mon_bufpool* • *dft_mon_lock* • *dft_mon_sort* • *dft_mon_stmt* • *dft_mon_table* • *dft_mon_uow*	중간	549 페이지의 『기본 데이터베이스 시스템 모니터 스위치 (dft_monswitches)』
dftdbpath	없음	568 페이지의 『기본 데이터베이스 경로(dftdbpath)』
diaglevel	낮음	545 페이지의 『진단 오류 캡쳐 레벨(diaglevel)』
diagpath	없음	546 페이지의 『진단 데이터 디렉토리 경로(diagpath)』
dir_cache	중간	438 페이지의 『디렉토리 캐쉬 지원(dir_cache)』
dir_obj_name	없음	527 페이지의 『DCE 이름 공간의 오브젝트 이름 (dir_obj_name)』
dir_path_name	없음	526 페이지의 『DCE 이름 공간의 디렉토리 경로 이름 (dir_path_name)』
dir_type	없음	525 페이지의 『디렉토리 서비스 유형(dir_type)』
discover	중간	531 페이지의 『발견 모드(discover)』

표 17. 구성 가능한 데이터베이스 관리 프로그램 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
discover_comm	낮음	533 페이지의 『검색 발견 통신 프로토콜 (discover_comm)』
discover_inst	낮음	534 페이지의 『발견 서버 인스턴스(discover_inst)』
dos_rqrioblk	높음	434 페이지의 『DOS 리퀘스터 I/O 블록 크기 (dos_rqrioblk)』
drda_heap_sz	낮음	421 페이지의 『DRDA 힙 크기(drda_heap_sz)』
fcm_num_anchors	높음	536 페이지의 『FCM 메시지 앵커의 수 (fcm_num_anchors)』
fcm_num_buffers	높음	537 페이지의 『FCM 버퍼 수(fcm_num_buffers)』
fcm_num_connect	높음	539 페이지의 『FCM 연결 항목의 수 (fcm_num_connect)』
fcm_num_rqb	높음	539 페이지의 『FCM 요청 블록의 수(fcm_num_rqb)』
federated	중간	560 페이지의 『연합 데이터베이스 시스템 지원(연합)』
fileserver	없음	522 페이지의 『IPX/SPX 파일 서버 이름(fileserver)』
indexrec	중간	489 페이지의 『색인 재작성 시간(indexrec)』
initdari_jvm	중간	472 페이지의 『JVM을 사용하여 DARI 프로세스 초기화(initdari_jvm)』
intra_parallel	높음	544 페이지의 『파티션 내 병렬 처리 작동 (intra_parallel)』
ipx_socket	없음	524 페이지의 『IPX/SPX 소켓 번호(ipx_socket)』
java_heap_sz	높음	441 페이지의 『Java 인터프리터의 최대 힙 크기 (java_heap_sz)』
jdk11_path	없음	559 페이지의 『JDK 1.1 설치 경로(jdk11_path)』
keepdari	중간	470 페이지의 『DARI 프로세스 유지 표시기(keepdari)』
maxagents	중간	462 페이지의 『에이전트의 최대수(maxagents)』
maxcagents	중간	464 페이지의 『동시 에이전트의 최대수(maxcagents)』
max_connretries	중간	541 페이지의 『노드 연결 재시도(max_connretries)』
max_coordagents	중간	465 페이지의 『조정 에이전트의 최대 수 (max_coordagents)』
maxdari	중간	471 페이지의 『DARI 프로세스의 최대수(maxdari)』
max_logicagents	중간	466 페이지의 『논리 에이전트의 최대 수 (max_logicagents)』
max_querydegree	높음	543 페이지의 『병렬의 처리 최대 조회 수준 (max_querydegree)』

표 17. 구성 가능한 데이터베이스 관리 프로그램 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
max_time_diff	중간	541 페이지의 『노드간의 최대 시차(max_time_diff)』
maxtotfilop	중간	460 페이지의 『열린 총 파일 수(maxtotfilop)』
min_priv_mem	중간	425 페이지의 『최소의 확약된 개인용 메모리 (min_priv_mem)』
mon_heap_sz	낮음	436 페이지의 『데이터베이스 시스템 모니터 힙 크기 (mon_heap_sz)』
nname	없음	519 페이지의 『NetBIOS 워크스테이션 이름(nname)』
notifylevel	낮음	548 페이지의 『레벨 통지(notifylevel)』
numdb	낮음	553 페이지의 『동시에 사용 중인 데이터베이스의 최대수 (numdb)』
num_initagents	중간	469 페이지의 『풀의 초기 에이전트의 수 (num_initagents)』
num_initdaris	중간	473 페이지의 『풀 내의 초기 분리(fenced) DARI 프로세스의 수(num_initdaris)』
num_poolagents	높음	467 페이지의 『에이전트 풀 크기(num_poolagents)』
objectname	없음	523 페이지의 『IPX/SPX DB2 서버 오브젝트 이름 (objectname)』
priv_mem_thresh	중간	426 페이지의 『개인용 메모리 임계값 (priv_mem_thresh)』
query_heap_sz	중간	420 페이지의 『조회 힙 크기(query_heap_sz)』
restbufsz	중간	405 페이지의 『기본 복원 버퍼 크기(restbufsz)』
resync_interval	없음	497 페이지의 『트랜잭션 재동기화 간격 (resync_interval)』
route_obj_name	없음	528 페이지의 『경로지정 정보 오브젝트 이름 (route_obj_name)』
rqrioblk	높음	432 페이지의 『클라이언트 I/O 블록 크기(rqrioblk)』
sheapthres	높음	415 페이지의 『정렬 힙 임계값(sheapthres)』
spm_log_file_sz	낮음	499 페이지의 『동기 지점 관리 프로그램 로그 파일 크기 (spm_log_file_sz)』
spm_log_path	중간	498 페이지의 『동기 지점 관리 프로그램 로그 파일 경로 (spm_log_path)』
spm_max_resync	낮음	501 페이지의 『동기 지점 관리 프로그램 재동기화 에이전트 한계(spm_max_resync)』
spm_name	없음	499 페이지의 『동기 지점 관리 프로그램 이름 (spm_name)』

표 17. 구성 가능한 데이터베이스 관리 프로그램 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
ss_logon	없음	569 페이지의 『DB2START/DB2STOP에 필요한 LOGON(ss_logon)』
start_stop_time	낮음	542 페이지의 『시작 및 중지 시간종료(start_stop_time)』
svcename	없음	520 페이지의 『TCP/IP 서비스 이름(svcename)』
sysadm_group	없음	561 페이지의 『시스템 관리 권한 그룹 이름 (sysadm_group)』
sysctrl_group	없음	562 페이지의 『시스템 제어 권한 그룹 이름 (sysctrl_group)』
sysmaint_group	없음	563 페이지의 『시스템 유지보수 권한 그룹 이름 (sysmaint_group)』
tm_database	없음	496 페이지의 『트랜잭션 관리 프로그램 데이터베이스 이름(tm_database)』
tp_mon_name	없음	555 페이지의 『트랜잭션 프로세서 모니터 이름 (tp_mon_name)』
tpname	없음	521 페이지의 『APPC 트랜잭션 프로그램 이름(tpname)』
trust_allclnts	없음	569 페이지의 『모든 클라이언트 신뢰(trust_allclnts)』
trust_clntauth	없음	571 페이지의 『신뢰성 있는 클라이언트 인증 (trust_clntauth)』
udf_mem_sz	낮음	422 페이지의 『UDF 공유 메모리 세트 크기 (udf_mem_sz)』
주: *cpuspeed* 매개변수는 성능에 큰 영향을 미칠 수 있지만, 매개변수 설명에 나와 있는 특정 환경을 제외하고는 기본값을 사용해야 합니다.		

표 18. 정보 전달용 데이터베이스 관리 프로그램 구성 매개변수

매개변수	추가 정보
nodetype	557 페이지의 『머신 노드 유형(nodetype)』
release	503 페이지의 『구성 파일 릴리스 레벨(release)』

데이터베이스 매개변수

각각의 데이터베이스에 대한 매개변수는 SQLDBCON이라는 이름의 구성 파일에 저장되어 있습니다. 이 파일은 SQLnnnnn 디렉토리 안에 데이터베이스에 대한 기타 제어 파일과 함께 들어 있는데, 여기서 nnnnn는 데이터베이스가 작성될 때 지정되는 번호입니다. 이 디렉토리의 위치에 대한 자세한 내용은 *관리 안내서: 계획*의 『데이터베이스 물리적 디렉토리』를 참조하십시오. 각 데이터베이스에는 고유한 구성 파일이 있으며, 파일에 있는 대부분의 매개변수는 해당 데이터베이스에 할당되는 자원의 양을 지정합니다. 파일에는 또한 데이터베이스의 상태를 나타내는 플래그와 설명 정보도 들어 있습니다.

SQLDBCON 파일은 직접 편집할 수 없으며, 제공되는 API 또는 해당 API를 호출하는 도구에 의해 변경되거나 볼 수 있습니다.

주의사항: DB2에 제공되지 않은 다른 방법을 사용하여 파일을 편집할 경우, 데이터베이스가 사용 불가능해질 수 있습니다. DB2에서 언급하고 지원하는 것이 아닌 다른 방법을 사용하여 이 파일을 변경하지 **않도록 해야** 합니다.

다음 세 가지 방법 중 하나를 사용하여 데이터베이스 구성 매개변수를 재설정, 갱신 및 볼 수 있습니다.

- 제어 센터 사용. DB2 제어 센터는 구성 매개변수 값을 변경하기 위해 데이터베이스 구성 노트북 및 성능 구성 마법사를 제공합니다. 이 마법사는 질문에 대해 사용자가 제공한 워크로드 및 데이터베이스에 대해 실행되는 트랜잭션 유형과 같은 응답에 따라 매개변수에 값을 생성합니다. 이 인터페이스의 사용법에 대한 자세한 내용은 제어 센터에서 사용 가능한 온라인 도움말을 참조하십시오.

  파티션된 데이터베이스 환경에서 SQLDBCON 파일은 각 데이터베이스 파티션에 대해 존재합니다. 제어 센터 데이터베이스 구성 노트북은 제어 센터 트리 뷰의 데이터베이스 오브젝트에서 노트북을 시작할 경우, 모든 파티션에서 값을 변경합니다. 데이터베이스 파티션 오브젝트에서 노트북을 시작할 경우에는 해당 파티션의 값만 변경합니다. (그러나 구성 매개변수 값은 모든 파티션에서 동일해야 합니다.)

  주: 성능 구성 마법사는 파티션된 데이터베이스 환경에서는 사용할 수 없습니다.

- 명령행 처리기 사용. 설정값을 변경하는 데 사용하는 명령을 빠르고 편리하게 입력할 수 있습니다. 다음 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.
  - GET DATABASE CONFIGURATION(또는 GET DB CFG)
  - UPDATE DATABASE CONFIGURATION(또는 UPDATE DB CFG)
  - RESET DATABASE CONFIGURATION(또는 RESET DB CFG)
- API(API) 사용. 호스트 언어로 된 프로그램으로부터 API가 쉽게 호출될 수 있습니다. 자세한 내용은 *Administrative API Reference*를 참조하십시오.

응용프로그램이 데이터베이스에 연결되어 있는 동안에는 대부분의 변경 가능한 매개변수에 대한 갱신사항이 적용되지 않습니다. 우선 데이터베이스로부터 모든 응용프로그램을 연결 해제해야 합니다. (데이터베이스가 활성화된 경우, 비활성화한 후 다시 활성화해야 합니다.) 그런 다음 데이터베이스에 새로 연결하면 그때부터 변경된 매개변수가 기능을 발휘합니다. *newlogpath* 및 *logfilsiz, logprimary* 등의 매개변수는 공간을 할당하는 작업과 관련된 오버헤드로 인해 시간이 오래 걸릴 수도 있습니다. 또한 데이터베이스에 테스트 연결을 하여 변경사항이 테스트 연결시에 기능을 보임으로써, 이에 관련된 오버헤드가 다른 사용자에게 영향을 미치지 않도록 할 수도 있습니다. 여기서 설명한 오버헤드에 대해 관심이 있으면 *Command Reference*에서 설명된 ACTIVATE DATABASE 명령을 사용해 보십시오.

주: *mincommit* 매개변수의 값을 갱신할 경우, 데이터베이스로부터 연결 해제하지 않아도 됩니다. 이 매개변수는 값을 변경할 때 자동으로 갱신됩니다.

몇몇 데이터베이스 구성 매개변수를 변경하는 것은 SQL 최적화 알고리즘에 의해 선택된 액세스 플랜에 영향을 미칠 수 있습니다. 이들 데이터베이스 매개변수는 101 페이지의 『조회 최적화에 영향을 주는 구성 매개변수』에 설명되어 있습니다. 논의된 매개변수 중 하나를 변경한 후에는 응용프로그램을 리바인드하여 해당 SQL 문에 대해 최상의 액세스 플랜이 사용되도록 해야 합니다. BIND 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

새 매개변수 값이 즉시 적용되지는 않지만, 매개변수 값 설정을 보면 항상 가장 최근에 변경된 값이 표시됩니다.

주: 많은 데이터베이스 구성 매개변수(예: *userexit*)가 도움말 또는 DB2 책에 『Yes』 또는 『No』, 『On』 또는 『Off』 값을 승인할 수 있다고 설명되어 있습니다. 혼란이 없도록 명확하게 하려면, 『Yes』는 『On』과, 『No』는 『Off』와 같다고 생각하면 됩니다.

데이터베이스 구성 매개변수 요약

다음 표는 데이터베이스 구성 파일의 매개변수를 나열하고 있습니다. 데이터베이스 구성 매개변수를 변경할 경우, 각 매개변수에 대한 자세한 정보를 참조하십시오.

다음 표의 『성능 영향』 컬럼은 시스템 성능에 영향을 미치는 각각의 매개변수의 상대적 중요도를 나타냅니다. 이 컬럼은 모든 환경에 정확하게 적용할 수 없습니다. 이 정보는 일반 정보로 보아야 합니다.

- **높음** - 매개변수가 성능에 상당한 영향을 미칠 수 있다는 것을 나타냅니다. 이러한 값은 신중하게 결정해야 하며, 경우에 따라 기본값을 받아들여야 합니다.
- **중간** - 매개변수가 성능에 어느 정도의 영향을 미칠 수 있다는 것을 나타냅니다. 사용자의 특정 작업 환경과 요구사항에 따라 이 매개변수를 조정하는 데 얼마나 많은 노력을 기울여야 하는지가 결정됩니다.
- **낮음** - 매개변수가 성능에 일반적인 수준 이하의 영향 또는 미비한 영향을 미칠 수 있다는 것을 나타냅니다.
- **없음** - 매개변수가 성능에 직접적인 영향을 미치지 않는다는 것을 나타냅니다. 성능을 향상시키기 위해서는 이 매개변수를 조정할 필요는 없다 하더라도, 통신 지원 등 시스템 구성의 여러 다른 사항에 대해 매우 중요한 매개변수일 수 있습니다.

표 19. 구성 가능한 데이터베이스 구성 매개변수

매개변수	성능 영향	추가 정보
app_ctl_heap_sz	중간	412 페이지의 『응용프로그램 제어 힙 크기 (app_ctl_heap_sz)』
applheapsz	중간	418 페이지의 『응용프로그램 힙 크기(applheapsz)』
autorestart	낮음	489 페이지의 『자동 재시작 가능(autorestart)』
avg_appls	높음	458 페이지의 『사용 중인 응용프로그램의 평균 수 (avg_appls)』

표 19. 구성 가능한 데이터베이스 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
buffpage	높음(사용 중일 경우)	396 페이지의 『버퍼 풀 크기(buffpage)』
catalogcache_sz	중간	401 페이지의 『카탈로그 캐쉬 크기(catalogcache_sz)』
chngpgs_thresh	높음	447 페이지의 『변경된 페이지 임계값(chngpgs_thresh)』
copyprotect	없음	506 페이지의 『복사 방지 가능(copyprotect)』
dbheap	중간	400 페이지의 『데이터베이스 힙(dbheap)』
dft_degree	높음	514 페이지의 『기본 등급(dft_degree)』
dft_extent_sz	중간	453 페이지의 『기본 테이블 공간 Extent 크기 (dft_extent_sz)』
dft_loadrec_ses	중간	491 페이지의 『로드 복구 세션의 기본 수 (dft_loadrec_ses)』
dft_prefetch_sz	중간	452 페이지의 『기본 프리페치 크기(dft_prefetch_sz)』
dft_queryopt	중간	515 페이지의 『기본 조회 최적화 클래스(dft_queryopt)』
dft_refresh_age	중간	516 페이지의 『기본 새로 고침 유효 기간 (dft_refresh_age)』
dft_sqlmathwarn	없음	512 페이지의 『산술 예외 시 계속(dft_sqlmathwarn)』
dir_obj_name	없음	527 페이지의 『DCE 이름 공간의 오브젝트 이름 (dir_obj_name)』
discover_db	중간	531 페이지의 『데이터베이스 발견(discover_db)』
dlchktime	중간	442 페이지의 『교착 상태 점검을 위한 시간 간격 (dlchktime)』
dl_expint	없음	506 페이지의 『데이터 링크 액세스 토큰 만기 간격 (dl_expint)』
dl_num_copies	없음	507 페이지의 『데이터 링크 사본 수(dl_num_copies)』
dl_time_drop	없음	507 페이지의 『삭제 후 데이터 링크 시간 (dl_time_drop)』
dl_token	낮음	508 페이지의 『데이터 링크 토큰 알고리즘(dl_token)』
dl_upper	없음	508 페이지의 『데이터 링크 토큰을 대문자로만 표시 (dl_upper)』
dyn_query_mgmt	낮음	502 페이지의 『동적 SQL 조회 관리 (dyn_query_mgmt)』
estore_seg_sz	중간	454 페이지의 『확장 저장 메모리 세그먼트 크기 (estore_seg_sz)』
indexrec	중간	489 페이지의 『색인 재작성 시간(indexrec)』
indexsort	낮음(393페이지 주 참조)	451 페이지의 『색인 정렬 플래그(indexsort)』

표 19. 구성 가능한 데이터베이스 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
locklist	효과가 상승하면 높음	406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
locktimeout	중간	445 페이지의 『잠금 시간종료(locktimeout)』
logbufsz	높음	402 페이지의 『로그 버퍼 크기(logbufsz)』
logfilsiz	중간	475 페이지의 『로그 파일의 크기(logfilsiz)』
logprimary	중간	477 페이지의 『1차 로그 파일의 수(logprimary)』
logretain	낮음	486 페이지의 『로그 유지 작동 가능(logretain)』
logsecond	중간	479 페이지의 『2차 로그 파일의 수(logsecond)』
maxappls	중간	456 페이지의 『사용 중인 응용프로그램의 최대수 (maxappls)』
maxfilop	중간	459 페이지의 『응용프로그램당 열린 최대 데이터베이스 파일 수(maxfilop)』
maxlocks	효과가 상승하면 높음	443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』
mincommit	높음	482 페이지의 『그룹 확약의 수(mincommit)』
min_dec_div_3	높음	431 페이지의 『DECDIV3(십진수 나누기 배율 3) (min_dec_div_3)』
newlogpath	낮음	480 페이지의 『데이터베이스 로그 경로 변경 (newlogpath)』
num_db_backups	없음	492 페이지의 『데이터베이스 백업의 수 (num_db_backups)』
num_estore_segs	중간	454 페이지의 『확장 저장영역 메모리 세그먼트의 수 (num_estore_segs)』
num_freqvalues	낮음	516 페이지의 『보유되어 자주 사용되는 값의 수 (num_freqvalues)』
num_iocleaners	높음	448 페이지의 『비동기 페이지 정리자(cleaner)의 수 (num_iocleaners)』
num_ioservers	높음	450 페이지의 『I/O 서버의 수(num_ioservers)』
num_quantiles	낮음	517 페이지의 『컬럼에 대한 quantile 수 (num_quantiles)』
pckcachesz	높음	410 페이지의 『패키지 캐쉬 크기(pckcachesz)』
rec_his_retentn	없음	492 페이지의 『복구 실행기록 보유 기간 (rec_his_retentn)』
seqdetect	높음	451 페이지의 『순차적 검출 플래그(seqdetect)』

표 19. 구성 가능한 데이터베이스 구성 매개변수 (계속)

매개변수	성능 영향	추가 정보
softmax	중간	484 페이지의 『복구 범위 및 소프트 점검점 간격 (softmax)』
sortheap	높음	414 페이지의 『정렬 힙 크기(sortheap)』
stat_heap_sz	낮음	419 페이지의 『통계 힙 크기(stat_heap_sz)』
stmtheap	중간	417 페이지의 『명령문 힙 크기(stmtheap)』
trackmod	낮음	493 페이지의 『수정된 페이지 추적 사용(trackmod)』
tsm_mgmtclass	없음	494 페이지의 『Tivoli Storage Manager 관리 클래스 (tsm_mgmtclass)』
tsm_nodename	없음	495 페이지의 『Tivoli Storage Manager 노드 이름 (tsm_nodename)』
tsm_owner	없음	495 페이지의 『Tivoli Storage Manager 소유자 이름 (tsm_owner)』
tsm_password	없음	494 페이지의 『Tivoli Storage Manager 암호 (tsm_password)』
userexit	낮음	487 페이지의 『User Exit 사용 가능(userexit)』
util_heap_sz	낮음	404 페이지의 『유틸리티 힙 크기(util_heap_sz)』
주: *indexsort* 매개변수를 기본값이 아닌 다른 값으로 변경하는 것은 색인 작성시의 성능에 부정적인 영향을 줄 수 있습니다. 이 매개변수에 대해서는 항상 기본값을 사용하도록 하는 것이 좋습니다.		

표 20. 정보 전달용 데이터베이스 구성 매개변수

매개변수	추가 정보
backup_pending	509 페이지의 『백업 보류 표시기(backup_pending)』
codepage	505 페이지의 『데이터베이스 코드 페이지(codepage)』
codeset	504 페이지의 『데이터베이스 코드 세트(codeset)』
collate_info	505 페이지의 『조합 정보(collate_info)』
country	504 페이지의 『데이터베이스 국가 코드(country)』
database_consistent	510 페이지의 『데이터베이스 일치(database_consistent)』
database_level	503 페이지의 『데이터베이스 릴리스 레벨 (database_level)』
log_retain_status	511 페이지의 『로그 유지 상태 표시기 (log_retain_status)』
loghead	482 페이지의 『사용 중인 첫 번째 로그 파일(loghead)』
logpath	481 페이지의 『로그 파일의 위치(logpath)』

표 20. 정보 전달용 데이터베이스 구성 매개변수 (계속)

매개변수	추가 정보
multipage_alloc	511 페이지의 『다중 페이지 파일 할당 사용 (multipage_alloc)』
numsegs	453 페이지의 『SMS 컨테이너의 기본 수(numsegs)』
release	503 페이지의 『구성 파일 릴리스 레벨(release)』
restore_pending	511 페이지의 『복원 보류(restore_pending)』
rollfwd_pending	510 페이지의 『롤 포워드 보류 표시기(rollfwd_pending)』
territory	504 페이지의 『데이터베이스 지역(territory)』
user_exit_status	511 페이지의 『User Exit 상태 표시기(user_exit_status)』

매개변수의 기능별 세부사항

다음에서는 다양한 구성 매개변수를 이해하고 조정하는 데 도움이 될 추가 설명을 제공하고 있습니다. 각 매개변수는 그 기능 및 목적에 따라 설명됩니다.

- 395 페이지의 『용량 관리』
- 474 페이지의 『로그 및 복구』
- 502 페이지의 『데이터베이스 관리』
- 519 페이지의 『통신』
- 535 페이지의 『파티션된 데이터베이스』
- 545 페이지의 『인스턴스 관리』

각 매개변수에 대한 설명에는 다음 정보가 포함됩니다.

구성 유형

구성 파일에 매개변수에 대한 설정이 들어 있음을 나타냅니다.

- 데이터베이스 관리 프로그램(데이터베이스 관리 프로그램의 인스턴스 및 해당 인스턴스 내에서 정의된 모든 데이터베이스에 영향을 미칩니다.)
- 데이터베이스(특정 데이터베이스에 영향을 미칩니다.)

매개변수 유형

매개변수 값을 변경할 수 있는지의 여부를 표시합니다.

- *구성 가능*

  범위 값이 있을 수 있으며, 데이터베이스 관리 프로그램의 응용프로그램 지식이나 벤치마킹 경험에 따라 매개변수를 조정해야 합니다.

- *정보용*

  이들 매개변수는 데이터베이스 관리 프로그램 자체에 의해서만 변경될 수 있으며, 데이터베이스가 작성된 DB2 릴리스 정보나 필요한 백업이 나올 것임을 나타내는 정보를 담고 있습니다.

용량 관리

데이터베이스 및 데이터베이스 관리 프로그램 레벨에서 시스템의 시간당 처리량에 영향을 미칠 수 있는 매개변수가 많이 있습니다. 이 매개변수는 다음과 같은 그룹으로 구분될 수 있습니다.

- 396 페이지의 『데이터베이스 공유 메모리』
- 412 페이지의 『응용프로그램 공유 메모리』
- 413 페이지의 『에이전트 개인용 메모리』
- 428 페이지의 『에이전트/응용프로그램 통신 메모리』
- 436 페이지의 『데이터베이스 관리 프로그램 인스턴스 메모리』
- 442 페이지의 『잠금』
- 446 페이지의 『I/O 및 저장』
- 455 페이지의 『에이전트』
- 469 페이지의 『저장 프로시듀어(DARI)』

DB2 메모리 관리에 대한 개요는 274 페이지의 『DB2의 메모리 사용법』을 참조하십시오.

데이터베이스 공유 메모리

다음 매개변수는 시스템에 할당된 데이터베이스 전역 메모리에 영향을 미칩니다.

- 『버퍼 풀 *크기*(buffpage)』.
- 400 페이지의 『데이터베이스 힙(dbheap)』.
- 401 페이지의 『카탈로그 캐쉬 *크기*(catalogcache_sz)』.
- 402 페이지의 『로그 버퍼 *크기*(logbufsz)』.
- 404 페이지의 『유틸리티 힙 *크기*(util_heap_sz)』.
- 405 페이지의 『기본 백업 버퍼 *크기*(backbufsz)』.
- 405 페이지의 『기본 복원 버퍼 *크기*(restbufsz)』.
- 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』.
- 410 페이지의 『패키지 캐쉬 *크기*(pckcachesz)』.

데이터베이스 전역 메모리가 데이터베이스 관리 프로그램에 의해 할당된 나머지 메모리와 어떤 연관성을 갖는지에 대한 자세한 내용은 274 페이지의 『DB2의 메모리 사용법』을 참조하십시오.

버퍼 풀 크기(buffpage)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 32비트 플랫폼
1 000 [2 -- 524 288]

UNIX 64비트 플랫폼
1 000 [2 -- 2 147 483 647]

OS/2 및 Windows NT
250 [2 -- 524 288]

측정 단위 페이지

할당 시기 첫 번째 응용프로그램이 데이터베이스에 연결될 때

해제 시기 마지막 응용프로그램이 데이터베이스로부터 연결 해제될 때

관련 매개변수

- 447 페이지의 『변경된 페이지 임계값 (chngpgs_thresh)』
- 400 페이지의 『데이터베이스 힙(dbheap)』
- 448 페이지의 『비동기 페이지 정리자(cleaner)의 수(num_iocleaners)』

각 데이터베이스는 적어도 하나의 버퍼 풀(데이터베이스가 작성될 때 작성되는 IBMDEFAULTBP)을 가지며, 더 이상 가질 수도 있습니다. 모든 버퍼 풀은 데이터베이스를 사용하는 모든 응용프로그램에 사용할 수 있는 전역 메모리에 상주합니다. 메모리는 데이터베이스가 위치한 머신에 할당됩니다. 버퍼 풀이 메모리의 필수 데이터를 수용할 만큼 충분히 클 경우, 디스크 활동은 덜 발생합니다. 반면에, 버퍼 풀이 충분히 크지 않으면 데이터베이스의 전반적인 성능이 현저하게 저하되고, 응용프로그램에 필요한 데이터를 처리하기 위해 디스크 활동(I/O)이 증가하여 데이터베이스 관리 프로그램이 I/O 바인드될 수 있습니다.

buffpage 매개변수는 CREATE BUFFERPOOL 또는 ALTER BUFFERPOOL 문이 NPAGES -1과 함께 수행될 때 버퍼 풀의 *크기*를 제어하며, 그렇지 않은 경우 *buffpage* 매개변수는 무시되고 NPAGES 매개변수에 의해 지정된 페이지 수로 버퍼 풀이 작성됩니다.

버퍼 풀에서 *buffpage* 매개변수가 사용 중인지 판별하려면 다음을 수행하십시오.

```
SELECT * from SYSCAT.BUFFERPOOLS.
```

NPAGES 값이 -1인 각 버퍼 풀은 *buffpage*를 사용합니다.

버퍼 풀 *크기*와 다른 시스템 사용자의 메모리 할당 사이에 교환이 있습니다. 데이터베이스 서버의 메모리 요구사항은 다중 사용자의 트랜잭션 비율이 높은 서버에서는 매우 중요한 것으로서, 데이터베이스 서버와 파일 또는 통신 서버는 종종 분리되어 서로 다른 머신에 놓이게 됩니다.

조회가 별칭에 액세스할 경우, 다음과 같은 경우에 버퍼 풀의 *크기*를 늘리도록 하십시오.

- 최적화 알고리즘이 대부분의 조작 또는 모든 조작이 지역적으로 완료된다고 판단할 경우. 조회가 처리될 때 최적화 알고리즘은 일반적으로 조작을 가능한 데이터 소스로 푸시다운됩니다. 예를 들어, GROUP BY 연산자는 대부분 데이터 소스에서 평가됩니다. 그러나 DB2에서 테이블을 처리하거나 지역적으로 조작을 수행하는 것은 비용이 가장 적게 드는 방법입니다. 이러한 상황은 DB2 서버 워크스테이션의 성능이 데이터 소스 워크스테이션보다 강력한 경우에 발생합니다.
- 정렬 조작은 지역적으로 완료되어야 합니다. 별칭을 포함하는 조회는 DB2 조합 순서에 따라 정렬됩니다. 데이터 소스가 동일한 조합 순서를 갖고 있지 않으면 모든 정렬 조작은 지역적으로 수행됩니다.

첫 번째 응용프로그램이 데이터베이스에 연결될 때 또는 데이터베이스가 가시적으로 활성화될 때 모든 버퍼 풀이 할당됩니다. 응용프로그램이 데이터베이스로부터 데이터를 요청함에 따라 데이터가 들어 있는 페이지가 디스크에서 버퍼 풀 중 하나로 전송됩니다. (데이터베이스 데이터는 디스크에 있는 테이블 내의 페이지에 저장되는 것을 주의하십시오.) 페이지가 변경되고 다음 중 한 상태가 발생할 때까지 페이지는 디스크에 다시 기록되지 않습니다.

- 모든 응용프로그램이 데이터베이스로부터 연결 해제됩니다.
- 데이터베이스가 가시적으로 비활성화됩니다.
- 데이터베이스 quiesce(즉, 연결된 모든 응용프로그램은 확약되었습니다.)
- 버퍼 풀로 읽어들여야 하는 또 다른 페이지에 대해 해당 공간이 필요합니다.
- 페이지 정리자(cleaner)를 사용할 수 있게 되어(*num_iocleaners*) 데이터베이스 관리 프로그램에 의해 활성화됩니다.

권장사항:

- *buffpage* 구성 매개변수 대신 CREATE BUFFERPOOL 및 ALTER BUFFERPOOL SQL문을 사용하여 버퍼 풀을 작성하고 해당 *크기*를 변경할 수 있습니다.
- 버퍼 풀 *크기*는 최적화 알고리즘이 액세스 플랜을 결정할 때 사용합니다. 이 매개변수를 변경하고 난 후에는 (REBIND PACKAGE 명령을 사용하여) 응용프로그램을 리바인드해야 합니다.
- 모든 버퍼 풀의 *크기*가 성능에 영향을 미치는 가장 주요한 요인이므로, 과도한 페이지 스왑핑이 발생하지 않도록 다음의 사항들을 고려해야 합니다.
  - 머신에 설치된 메모리의 양

  - 같은 머신에서 데이터베이스 관리 프로그램과 동시에 수행되는 다른 응용프로그램에 필요한 메모리

  *페이지 스와핑*은 액세스 중인 페이지를 보유할 메모리가 충분하지 않을 때 발생합니다. 그 결과, 페이지가 임시 디스크 저장에 기록되어(『스와핑』) 다른 페이지를 위한 공간을 만듭니다. 임시 디스크 저장영역에 있는 페이지가 필요하면 메모리로 『다시 스와핑』됩니다.
- 다음과 같은 경우, 시스템 메모리의 75%를 데이터베이스 버퍼 풀에 할당할 수 있습니다.
  - 다중 사용자
  - 데이터베이스 서버로만 사용되는 머신
  - 동일한 데이터와 색인 페이지에 대한 다량의 반복적인 액세스
  - 머신상에 하나의 데이터베이스
- 모든 버퍼 풀 페이지가 할당된 경우, 내부 제어 구조에 대해 데이터베이스 힙에서 일부 공간이 사용됩니다.

  버퍼 풀의 전체 크기가 증가될 경우에도 *dbheap*을 증가시켜야 합니다.
- 연합 환경에서 작업 중일 때 데이터 소스 조합 순서가 DB2 조합 순서와 일치하면 서버 옵션 collating_sequence도 일치하게 표시되도록 설정되어 있는지 확인하십시오. 즉, collating_sequence 옵션을 『Y』로 설정하십시오. 데이터 소스 조합 순서와 일치하는 특정 조합 순서를 사용하여 연합 데이터베이스를 작성할 수 있습니다. 이 접근 방법은 모든 데이터 소스가 동일한 조합 순서를 사용하거나 대부분 또는 모든 컬럼 함수가 동일한 조합 순서를 사용하는 데이터 소스로 지정될 경우, 성능이 향상됩니다. 데이터 소스가 DB2의 조합 순서와 다른 조합 순서를 사용할 경우, DB2의 조합 순서에 따라 대부분의 조작은 데이터 소스에서 원격으로 평가될 수 없습니다. 이 서버 옵션에 대한 자세한 내용은 *관리 안내서: 구현*을 참조하십시오.

데이터베이스 시스템 모니터를 사용하여 사용자의 버퍼 풀을 조정하는 데 도움이 되는 버퍼 풀 사용 비율(hit ratio)을 계산할 수 있습니다. 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

데이터베이스 힙(dbheap)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 1200 [32 - 524 288]

지역 및 원격 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
600 [32 - 524 288]

지역 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
300 [32 - 524 288]

측정 단위 페이지(4KB)

할당 시기 데이터베이스로의 첫 번째 연결

해제 시기 마지막 응용프로그램이 데이터베이스로부터 연결 해제될 때

관련 매개변수

- 401 페이지의 『카탈로그 캐쉬 크기(catalogcache_sz)』
- 402 페이지의 『로그 버퍼 크기(logbufsz)』

데이터베이스당 하나의 데이터베이스 힙이 있어, 데이터베이스 관리 프로그램이 데이터베이스에 연결된 모든 응용프로그램 대신에 이를 사용합니다. 여기에는 테이블, 색인, 테이블 공간 및 버퍼 풀에 대한 제어 블록 정보가 있습니다. 로그 버퍼(*logbufsz*) 및 카탈로그 캐쉬(*catalogcache_sz*)용 공간도 있습니다. 그러므로 힙 크기는 주어진 시간에 힙에 저장된 제어 블록 수에 따라 결정됩니다. 제어 블록 정보는 모든 응용프로그램이 데이터베이스로부터 연결 해제될 때까지 힙에 남아 있습니다.

첫 번째 연결시에 데이터베이스 관리 프로그램이 시작하는 데 필요한 최소량이 할당됩니다. 데이터 영역은 필요에 따라 *dbheap*이 지정하는 최대값까지 확장됩니다.

권장사항: 응용프로그램에 데이터베이스 힙에는 명령문을 처리할 만큼 충분한 저장영역이 없음을 나타내는 오류가 수신될 경우, 이 값을 증가시켜야 합니다.

데이터베이스 시스템 모니터를 사용하여 데이터베이스 힙에 사용된 가장 큰 메모리를 추적할 수도 있습니다. 자세한 내용은 *시스템 모니터 안내 및 참조서*의 *db_heap_top(할당된 최대 데이터베이스 힙)* 모니터 요소 설명을 참조하십시오.

이 매개변수를 설정할 때에는 다음 사항을 고려해야 합니다.

- 로그 버퍼는 데이터베이스 힙으로부터 할당되기 때문에 *logbufsz*의 값을 고려해야 합니다.
- 카탈로그 캐쉬는 데이터베이스 힙으로부터 할당되기 때문에 *catalogcache_sz*의 값을 고려해야 합니다.

카탈로그 캐쉬 크기(catalogcache_sz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 64 [1 - 60 000]

지역 및 원격 클라이언트가 있는 OS/2 및 Windows NT 대이터베이스 서버
32 [1 - 60 000]

지역 클아이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
16 [1 - 60 000]

측정 단위 페이지(4KB)

관련 매개변수

- 400 페이지의 『데이터베이스 힙(dbheap)』
- 402 페이지의 『로그 버퍼 크기(logbufsz)』
- 412 페이지의 『응용프로그램 제어 힙 크기(app_ctl_heap_sz)』

이 매개변수는 카탈로그 캐쉬가 데이터베이스 힙(*dbheap*)에서 사용할 수 있는 공간의 최대 *크기*를 나타냅니다. 카탈로그 캐쉬는 SQL문의 컴파일시에 테이블, 뷰 또는 별명을 참조할 때 사용되는 테이블 설명자 정보를 저장하는 데 사용됩니다.

이 캐쉬를 사용하면 이전의 명령문에서 동일한 테이블, 뷰 또는 별명을 참조한 경우 SQL문(동적 SQL 포함) 바인딩의 성능을 높이는 데 도움이 될 수 있습니다. 선언된 임시 테이블에 대한 설명자 정보는 카탈로그 캐쉬에 저장되지 않습니다. 그 대신, 응용프로그램 제어 힙(heap)이 사용됩니다.

테이블에 대해 DDL문을 수행하면 카탈로그 캐쉬에 있는 해당 테이블의 항목을 제거하게 됩니다. 그렇지 않으면 테이블 항목은 다른 테이블에 공간이 필요할 때까지 캐쉬에 유지되지만, 해당 테이블을 참조하는 작업 단위(UOW)가 완료될 때까지는 제거되지 않습니다.

권장사항: 기본값으로 시작하고 데이터베이스 시스템 모니터로 조정하십시오.

모니터 요소에 대한 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

- *cat_cache_lookups*(카탈로그 캐쉬 찾아보기)
- *cat_cache_inserts*(카탈로그 캐쉬 삽입)
- *cat_cache_overflows*(카탈로그 캐쉬 오버플로우)
- *cat_cache_heap_full*(카탈로그 캐쉬 힙(heap) 가득 참)

데이터베이스 시스템 모니터 요소는 이 구성 매개변수의 조정 여부를 결정하는 데 도움을 줄 수 있습니다. 이 매개변수를 조정할 때, 예를 들면, 한 번에 두 페이지씩 작은 증분으로 늘려야 합니다.

일반적으로 작업 단위(UOW)에 몇몇 동적 SQL문이 포함되어 있거나 많은 정적 SQL문을 포함한 패키지를 바인드하는 경우, 캐쉬 공간이 더 많이 필요합니다.

카탈로그 캐쉬의 *크기*를 설정할 때는 로그 파일의 *크기*(*logbufsz*)도 고려해야 하는데, *catalogcache_sz* 및 *logbufsz*가 모든 데이터베이스 힙(*dbheap*)으로부터 할당되기 때문입니다.

로그 버퍼 크기(logbufsz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 32비트 플랫폼

8 [4 -- 4 096]

UNIX 64비트 플랫폼

8 [4 -- 65 535]

OS/2 및 Windows NT

8 [4 -- 4 096]

측정 단위 페이지(4KB)

관련 매개변수

- 401 페이지의 『카탈로그 캐쉬 크기(catalogcache_sz)』
- 400 페이지의 『데이터베이스 힙(dbheap)』
- 482 페이지의 『그룹 확약의 수(mincommit)』

이 매개변수로 데이터베이스 힙의 양을 지정하여(*dbheap* 매개변수에 의해 정의됨) 디스크에 로그 레코드를 기록하기 전에 이들 레코드에 대한 버퍼로 사용할 수 있습니다. 로그 레코드는 다음 중 하나가 발생하면 디스크로 기록됩니다.

- *mincommit* 구성 매개변수로 정의된 트랜잭션 확약 또는 트랜잭션 그룹의 확약
- 로그 버퍼가 가득 참(full)
- 기타 내부 데이터베이스 관리 프로그램 이벤트의 결과로 기인하여

이 매개변수는 *dbheap* 매개변수 이하여야 합니다. 로그 레코드의 버퍼링은 파일 I/O를 기록하는 데 보다 효과적일 수 있습니다. 이는 로그 레코드가 디스크에 기록되는 빈도 수가 줄어들고 한 번에 더 많은 로그 레코드가 기록되기 때문입니다.

권장사항: 전용 로그 디스크에 상당한 읽기 활동이 있을 경우 또는 디스크 사용이 많을 경우, 이 버퍼 영역의 크기를 늘리십시오. 이 매개변수 값을 증가시키는 경우, 로그 버퍼 영역이 *dbheap* 매개변수에 의해 제어된 공간을 사용하기 때문에 *dbheap* 매개변수도 고려해야 합니다.

데이터베이스 시스템 모니터를 사용하여 어느 정도의 로그 버퍼 공간이 특정 트랜잭션(또는 작업 단위)에 사용되는지 판별할 수 있습니다.

자세한 내용은 *시스템 모니터 안내 및 참조서*의 *log_space_used*(사용된 작업 로그 공간 단위) 모니터 요소 설명을 참조하십시오.

로그 버퍼 *크기*를 설정할 때에는 카탈로그 캐쉬의 *크기*(*catalogcache_sz*)도 고려해야 하는데, *logbufsz_sz* 및 *catalogcache_sz*가 모두 데이터베이스 힙(*dbheap*)으로부터 할당되기 때문입니다.

유틸리티 힙 크기(util_heap_sz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 5000 [16 - 524 288]

측정 단위 페이지(4KB)

할당 시기 데이터베이스 관리 프로그램 유틸리티에 필요할 때

해제 시기 유틸리티에 더 이상 메모리가 필요하지 않을 때

관련 매개변수

- 405 페이지의 『기본 백업 버퍼 크기(backbufsz)』
- 405 페이지의 『기본 복원 버퍼 크기(restbufsz)』

이 매개변수는 BACKUP, RESTORE 및 LOAD(로드 복구 포함) 유틸리티가 동시에 사용할 수 있는 메모리의 최대량을 나타냅니다.

권장사항: 유틸리티의 공간이 부족한 경우가 아니라면 기본값을 사용하십시오. 부족한 경우, 이 값을 증가시켜야 합니다. 시스템 메모리가 제한되는 경우, 이 매개변수의 값을 더 낮추어 데이터베이스 유틸리티에 의해 사용된 메모리를 제한합니다. 이 매개변수를 너무 낮게 설정하면 유틸리티를 동시에 수행할 수 없습니다. 동시에 수행되는 유틸리티에 할당하려는 모든 버퍼를 수용할 만큼 충분히 크게 이 매개변수를 설정해야 합니다.

기본 백업 버퍼 크기(backbufsz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 1024 [8 -- 16 384]

측정 단위 페이지(4KB)

할당 시기 백업 유틸리티가 호출될 때

해제 시기 백업 유틸리티가 처리를 완료할 때

관련 매개변수

- 『기본 복원 버퍼 크기(restbufsz)』
- 404 페이지의 『유틸리티 힙 크기(util_heap_sz)』

이 매개변수는 백업 유틸리티 호출시 버퍼 크기가 명시적으로 지정되지 않은 경우, 데이터베이스 백업에 사용되는 버퍼 크기를 지정합니다. 백업 유틸리티에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

데이터베이스 백업시, 데이터는 먼저 내부 버퍼로 복사됩니다. 그런 다음 버퍼가 가득 차면 이 버퍼에서 백업 미디어로 기록됩니다.

이 버퍼 크기를 조정하면 백업 유틸리티의 성능을 향상시킬 뿐만 아니라, 동시에 수행되는 다른 데이터베이스 작업의 성능에 미치는 영향을 최소화하는 데에도 도움이 됩니다.

기본 복원 버퍼 크기(restbufsz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 1024 [8 -- 16 384]

측정 단위 페이지(4KB)

할당 시기 복원 유틸리티가 호출될 때

해제 시기 복원 유틸리티가 처리를 완료할 때

관련 매개변수

- 405 페이지의 『기본 백업 버퍼 *크기* (backbufsz)』
- 404 페이지의 『유틸리티 힙 *크기*(util_heap_sz)』

이 매개변수는 복원 데이터베이스 유틸리티 호출시 버퍼 *크기*가 명시적으로 지정되지 않은 경우, 데이터베이스 복원에 사용되는 버퍼 *크기*를 지정합니다. 복원 유틸리티에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

데이터베이스 복원시, 데이터는 먼저 백업 미디어에서 내부 버퍼로 복사됩니다. 그런 다음 버퍼가 가득 차면 이 버퍼에서 목표 데이터베이스 미디어로 기록됩니다.

이 버퍼 *크기*를 조정하면 데이터베이스 복원 유틸리티의 성능을 향상시킬 뿐만 아니라, 동시에 수행되는 다른 데이터베이스 작업의 성능에 미치는 영향을 최소화하는 데에도 도움이 됩니다.

잠금 목록용 최대 저장영역(locklist)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 100 [4 - 524 288]

지역 및 원격 클라이언트가 있는 OS/2 및 NT 데이터베이스 서버
50 [4 - 524 288]

지역 클라이언트가 있는 OS/2 및 NT 데이터베이스 서버 25 [4 - 60 000]

측정 단위 페이지(4KB)

할당 시기 첫 번째 응용프로그램이 데이터베이스에 연결될 때

해제 시기 마지막 응용프로그램이 데이터베이스로부터 연결 해제될 때

관련 매개변수

- 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』
- 456 페이지의 『사용 중인 응용프로그램의 최대 수(maxappls)』

이 매개변수는 잠금 목록에 할당되는 저장영역의 양을 나타냅니다. 데이터베이스당 하나의 잠금 목록이 있고, 여기에는 데이터베이스에 동시에 연결되어 있는 모든 응용프로그램이 보유한 잠금이 들어 있습니다. 잠금이란 여러 응용프로그램이 데이터베이스의 데이터에 동시에 액세스하는 것을 제어하기 위해 데이터베이스 관리 프로그램이 사용하는 메커니즘입니다. 행과 테이블 모두 잠길 수 있습니다. 데이터베이스 관리 프로그램은 내부 사용을 위해 잠금을 확보할 수도 있습니다.

잠금에 대한 자세한 내용은 55 페이지의 『잠금』을 참조하십시오.

오브젝트에 다른 잠금이 있는지에 따라 각 잠금에는 36 또는 72바이트의 잠금 목록이 필요합니다.

- 오브젝트에 다른 잠금이 없을 때 하나의 잠금을 보유하는 데 72바이트가 필요합니다.
- 오브젝트에 다른 잠금이 있을 때 하나의 잠금을 기록하는 데 36바이트가 필요합니다.

응용프로그램이 사용하는 잠금 목록의 비율이 *maxlocks*에 달하면 응용프로그램이 보유하는 잠금에 대해 데이터베이스 관리 프로그램이 행에서 테이블로의 잠금 레벨 자동 업그레이드를 수행합니다(아래에 설명됨). 레벨 자동 업그레이드(escalation) 프로세스 자체가 많은 시간이 걸리지는 않아도, 전체 테이블을 잠그는 것은 (개별적인 행에 비해) 동시성을 감소시키고, 영향받은 테이블에 대한 후속 액세스의 경우 전반적인 데이터베이스 성능이 저하될 수도 있습니다. 잠금 목록의 *크기*를 제어하는 방법으로는 다음이 있습니다.

- 자주 COMMIT를 수행하여 잠금을 해제하십시오.
- 많은 갱신을 수행할 때에는 그 전에 전체 테이블을 잠그십시오(SQL LOCK TABLE문 사용). 이는 다른 것이 갱신을 방해하지 않도록 하나의 잠금만을 사용하지만, 데이터의 동시성은 감소됩니다.

  또한 ALTER TABLE문의 LOCKSIZE 매개변수를 사용하여 특정 테이블에 대해 잠금을 수행하는 방법을 제어할 수도 있습니다. 자세한 내용은 *SQL 참조서*를 참조하십시오.

  반복 읽기(RR) 분리 레벨을 사용하면 자동 테이블 잠금이 발생할 수도 있습니다. 분리 레벨에 대한 자세한 내용은 45 페이지의 『제3장 응용프로그램 고려사항』을 참조하십시오.
- 보유된 공유 잠금의 수를 감소시킬 수 있는 경우에는 커서 안정성(CS) 분리 레벨을 사용하십시오. 응용프로그램 무결성 요구사항이 절충되지 않은 경우, 커서 안정성(CS) 대신에 미확약 읽기(UR)를 사용하여 잠금량을 더 줄이십시오.

일단 잠금 목록이 가득 차면 잠금 레벨 자동 업그레이드가 더 많은 테이블 잠금과 더 적은 행 잠금을 생성하여 데이터베이스 내의 공유 오브젝트에서 동시성을 감소시킴으로써 성능이 저하될 수 있습니다. 그 외에도, 응용프로그램 사이에 더 많은 교착 상태가 발생하여(모든 응용프로그램이 제한된 수의 테이블 잠금에 대해 대기하기 때문에) 트랜잭션이 구간 복원되는 결과가 발생하게 됩니다. 데이터베이스에 대한 잠금 요청 수가 최대수에 도달하게 되면 사용자의 응용프로그램이 SQLCODE -912를 수신하게 됩니다.

권장사항: 잠금 레벨 자동 업그레이드로 인해 성능 문제가 발생할 경우, 이 매개변수 또는 *maxlocks* 매개변수의 값을 증가시키십시오. 데이터베이스 시스템 모니터를 사용하여 잠금 레벨 자동 업그레이드가 발생하는지 결정할 수 있습니다.

자세한 내용은 *시스템 모니터 안내 및 참조서*의 *lock_escals(잠금 레벨 자동 업그레이드)* 모니터 요소 설명을 참조하십시오.

다음 단계는 잠금 목록에 필요한 페이지 수를 결정하는 데 도움이 될 수 있습니다.

1. 잠금 목록 *크기*의 하한 값을 계산하십시오.

   ```
   (512 * 36 * maxappls) / 4096
   ```

   여기서 512는 응용프로그램당 평균 잠금 수를 추정한 것이고, 36은 기존의 잠금이 있는 오브젝트에서 각 잠금에 필요한 바이트 수입니다.
2. 잠금 목록 *크기*의 상한 값을 계산하십시오.

   ```
   (512 * 72 * maxappls) / 4096
   ```

   여기서 72는 오브젝트에 대한 첫 번째 잠금에 필요한 바이트 수입니다.
3. 사용자의 데이터에 대해 갖게 될 동시성의 양을 추정하고, 예상에 따라 계산한 상한 값과 하한 값 사이에서 *locklist*의 초기 값을 선택하십시오.
4. 아래 설명처럼 데이터베이스 시스템 모니터를 사용하여 이 매개변수 값을 조정하십시오.

데이터베이스 시스템 모니터를 사용하여 주어진 트랜잭션이 보유하는 최대 잠금 수를 결정할 수 있습니다.

자세한 내용은 *시스템 모니터 안내 및 참조서*의 *locks_held_top(보유된 최소 잠금 수)* 모니터 요소 설명을 참조하십시오.

이 정보를 사용하면 응용프로그램마다 추정된 잠금 수를 유효하게 하거나 조정하는 데 도움이 될 수 있습니다. 이 유효성 검사를 수행하려면 몇 개의 응용프로그램(모니터 정보는 응용프로그램 레벨이 아닌 트랜잭션 레벨에서 제공됨에 주의)을 샘플로 해야 합니다.

*maxappls*가 증가된 경우 또는 수행되고 있는 응용프로그램이 확약을 자주 실행하지 않는 경우, *locklist*를 증가시키려 할 수도 있습니다.

이 매개변수를 변경한 후에는 (REBIND PACKAGE 명령을 사용하여) 응용프로그램을 리바인드해야 합니다.

응용프로그램 성능 및 조회 최적화에 미치는 영향에 대한 자세한 내용은 43 페이지의 『제2부 응용프로그램 성능 조정』을 참조하십시오.

패키지 캐쉬 크기(pckcachesz)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	
	UNIX 32비트 플랫폼 -1 [-1, 32 -- 128 000]
	UNIX 64비트 플랫폼 -1 [-1, 32 -- 524 288]
	OS/2 및 Windows NT -1 [-1, 32 -- 128 000]
측정 단위	페이지(4KB)
할당 시기	데이터베이스가 초기화될 때
해제 시기	데이터베이스가 종료될 때

이 매개변수는 데이터베이스 전역 메모리로부터 할당되며, 데이터베이스에 정적 및 동적 SQL문을 캐쉬화하는 데 사용됩니다. 파티션된 데이터베이스 시스템에는 각 데이터베이스 파티션에 대해 하나의 패키지 캐쉬가 있습니다.

데이터베이스 관리 프로그램은 패키지 캐쉬를 사용하여 패키지를 재로드할 때 시스템 카탈로그에 액세스할 필요가 없게 하고, 동적 SQL의 경우에는 컴파일할 필요가 없게 하여 내부 오버헤드 시간을 줄입니다. 다음 중 하나가 발생할 때까지 섹션은 패키지 캐쉬에 유지됩니다.

- 데이터베이스가 종료될 때
- 패키지나 동적 SQL문이 무효화될 때
- 캐쉬 공간이 부족할 때

데이터베이스에 연결된 응용프로그램이 동일한 명령문을 여러 번 사용할 때, 정적 또는 동적 SQL문에 대한 이러한 섹션 캐쉬는 특히 성능을 향상시킬 수 있습니다. 이 내용은 트랜잭션 처리 응용프로그램에서 특히 중요한 사항입니다.

기본값(-1)을 취하면 페이지 할당을 계산하는 데 사용되는 값은 *maxappls* 구성 매개변수에 지정된 값의 8배가 됩니다. *maxappls* 8배 값이 32보다 적을 경우 예외가 발생합니다. 이 경우, 기본값인 -1은 *pckcachesz*를 32로 설정합니다.

권장사항: 이 매개변수를 조정할 때 패키지 캐쉬용으로 예약된 여분의 메모리가 버퍼 풀과 같은 다른 목적으로 할당될 때 더 효율적일 수 있는지를 고려해야 합니다. 이러한 이유로, 이 매개변수를 조정할 때 벤치마킹 기법을 사용해야 합니다.

초기에 몇몇 섹션이 사용된 다음 일부만이 반복하여 수행될 때 이 매개변수를 조정하는 것이 매우 중요합니다. 캐쉬가 너무 클 경우, 초기 섹션의 사본을 보유하느라 메모리가 소모됩니다.

다음 모니터 요소에 대한 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

- *pkg_cache_lookups*(패키지 캐쉬 찾아보기)
- *pkg_cache_inserts*(패키지 캐쉬 삽입)
- *pkg_cache_size_top*(최대 패키지 캐쉬 크기)
- *pkg_cache_num_overflows*(패키지 캐쉬 오버플로우의 수)

데이터베이스 시스템 모니터 요소는 이 구성 매개변수의 조정 여부를 결정하는 데 도움을 줄 수 있습니다.

주: 패키지 캐쉬가 작업 캐쉬일 경우, 이 매개변수를 0으로 설정할 수 없습니다. 이 캐쉬에는 현재 실행 중일 SQL문의 모든 섹션을 보유할 만큼 충분한 메모리가 있어야 합니다. 현재 필요한 것보다 더 많은 공간이 할당될 경우, 섹션은 캐쉬됩니다. 이 절은 로드하거나 컴파일할 필요 없이 필요할 때 실행될 수 있습니다.

pckcachesz 매개변수에 의해 지정된 한계는 소프트 한계입니다. 메모리가 아직 데이터베이스 공유 세트에서 사용 가능한 경우, 필요하면 이 한계를 초과할 수 있습니다. *pkg_cache_size_top* 모니터 요소를 사용하여 패키지 캐쉬가 늘어날 최대 크기를 판별할 수 있으며, *pkg_cache_num_overflows* 모니터 요소를 사용하여 몇 번이나 *pckcachesz* 매개변수에 의해 지정된 한계를 초과했는지를 판별할 수 있습니다.

응용프로그램 공유 메모리

다음 매개변수는 응용프로그램에 대해 작동하는 모든 에이전트(조정 및 서브에이전트 둘 다)에 의해 사용되는 작업 영역을 지정합니다.

- 『응용프로그램 제어 힙 *크기*(app_ctl_heap_sz)』

응용프로그램 제어 힙 크기(app_ctl_heap_sz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

지역 및 원격 클라이언트가 있는 데이터베이스 서버 128 [1-64 000]

지역 클라이언트가 있는 데이터베이스 서버
64 [1-64 000](UNIX가 아닌 플랫폼의 경우)

128 [1-64 000](UNIX 기반 플랫폼의 경우)

지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 256 [1-64 000]

측정 단위 페이지(4KB)

할당 시기 응용프로그램이 시작될 때

해제 시기 응용프로그램이 완료될 때

관련 매개변수

- 401 페이지의 『카탈로그 캐쉬 크기(catalogcache_sz)』
- 418 페이지의 『응용프로그램 힙 크기(applheapsz)』
- 544 페이지의 『파티션 내 병렬 처리 작동(intra_parallel)』

내부 병렬 처리가 작동 가능한 상태에서(intra-parallel=ON) 파티션된 데이터베이스 및 파티션되지 않은 데이터베이스의 경우, 이것은 응용프로그램 제어 힙에 대해 할당된 공유 메모리 영역의 *크기*입니다. 내부 병렬 처리가 작동 가능하지 않은 상태에서 (intra_parallel=OFF) 파티션되지 않은 데이터베이스의 경우, 이것은 힙을 할당할 수 있는 최대 개인용 메모리입니다. 파티션마다 각 연결에 하나의 응용프로그램 제어 힙이 있습니다.

SQL문을 나타내는 실행 가능한 섹션을 저장할 경우 동일한 요청 대신 작업하는 에이전트와 파티션된 데이터베이스 환경 사이에 정보를 공유하기 위해 일차적으로 응용프로그램 제어 힙이 필요합니다. 이 힙의 사용은 1보다 작거나 같은 병렬 처리 수준으로 조회를 수행할 때 파티션되지 않은 데이터베이스에 대해 최소입니다.

이 힙은 선언된 임시 테이블에 대한 설명자 정보를 저장하는 데도 사용됩니다. 명시적으로 삭제되지 않은 선언된 모든 임시 테이블에 대한 설명자 정보는 이 힙(heap)의 메모리에 보존되므로 선언된 임시 테이블이 삭제될 때까지 삭제할 수 없습니다.

권장사항: 처음에는 기본값을 사용하여 시작하십시오. 복잡한 응용프로그램을 수행하거나 시스템에 많은 수의 데이터베이스 파티션이 있을 경우 또는 선언된 임시 테이블을 사용할 경우 값을 더 높게 설정할 수도 있습니다. 필요한 메모리 양은 동시에 사용 중인 선언된 임시 테이블 수에 따라 증가됩니다. 컬럼이 많은 선언된 임시 테이블에서는 컬럼이 적은 테이블보다 테이블 설명자 *크기*가 크므로, 응용프로그램의 선언된 임시 테이블에 컬럼 수가 많으면 응용프로그램 제어 힙에서의 수요도 증가됩니다.

에이전트 개인용 메모리

다음 매개변수는 각 데이터베이스 에이전트에 사용되는 메모리의 양에 영향을 미칩니다.

- 414 페이지의 『정렬 힙 *크기*(sortheap)』.
- 415 페이지의 『정렬 힙 임계값(sheapthres)』.
- 417 페이지의 『명령문 힙 *크기*(stmtheap)』.
- 418 페이지의 『응용프로그램 힙 *크기*(applheapsz)』.
- 419 페이지의 『통계 힙 *크기*(stat_heap_sz)』.

- 420 페이지의 『조회 힙 *크기*(query_heap_sz)』.
- 421 페이지의 『DRDA 힙 *크기*(drda_heap_sz)』.
- 422 페이지의 『UDF 공유 메모리 세트 *크기*(udf_mem_sz)』.
- 424 페이지의 『에이전트 스택 *크기*(agent_stack_sz)』.
- 425 페이지의 『최소의 확약된 개인용 메모리(min_priv_mem)』.
- 426 페이지의 『개인용 메모리 임계값(priv_mem_thresh)』.
- 441 페이지의 『Java 인터프리터의 최대 힙 *크기*(java_heap_sz)』. UNIX 기반 플랫폼에서는 *java_heap_sz*가 에이전트별로 할당됩니다.

개인용 에이전트 메모리가 데이터베이스 관리 프로그램에 의해 할당된 나머지 메모리와 어떤 관계를 가지고 있는지에 대한 재세한 내용은 274 페이지의 『DB2의 메모리 사용법』을 참조하십시오.

정렬 힙 크기(sortheap)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	256 [16 - 524 288]
측정 단위	페이지(4KB)
할당 시기	정렬을 수행해야 할 때
해제 시기	정렬이 완료될 때
관련 매개변수	415 페이지의 『정렬 힙 임계값(sheapthres)』

이 매개변수는 개별 정렬에 사용될 개인용 메모리 페이지의 최대수나, 공유 정렬에 사용될 공유 메모리 페이지의 최대수를 지정합니다. 개별 정렬의 경우, 이 매개변수는 에이전트 개인용 메모리에 영향을 줍니다. 공유 정렬의 경우, 이 매개변수는 데이터베이스 공유 메모리에 영향을 줍니다. 정렬마다 필요한 만큼 데이터베이스 관리 프로그램에 의해 할당되어 개별 정렬 힙이 있습니다. 정렬 힙은 데이터가 정렬되는 영역입니다. 최적화 알고리즘에 의해 지정될 경우, 매개변수에 의해 지정된 힙보다 더 작은 힙이 최적화 알고리즘에 의해 제공된 정보를 사용하여 할당됩니다.

권장사항:

정렬 힙(heap)으로 작업할 때 다음 사항을 고려하십시오.

- 적합한 색인으로 정렬 힙 사용을 최소화할 수 있습니다.
- 해쉬 조인 버퍼와 동적 비트맵은 (AND 작업 색인과 Star 조인을 사용하는) 정렬 힙(heap) 메모리를 사용하십시오. 이들 기술을 사용할 때 이 매개변수의 크기를 늘리십시오.
- 자주 많은 데이터를 정렬해야 하는 경우, 이 매개변수의 크기를 늘리십시오.
- 이 매개변수의 값을 증가시키면 데이터베이스 관리 프로그램 구성 파일에 있는 *sheapthres* 매개변수도 조정해야 하는지를 조사해야 합니다.
- 정렬 힙의 크기는 최적화 알고리즘이 액세스 경로를 결정할 때 사용됩니다. 이 매개변수를 변경한 후에는 (REBIND PACKAGE 명령을 사용하여) 응용프로그램을 리바인드해야 합니다.

정렬 힙 임계값(sheapthres)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

UNIX 32비트 플랫폼

20 000 [250 -- 2 097 152]

UNIX 64비트 플랫폼

20 000 [250 -- 2 147 483 647]

OS/2 및 Windows NT

10 000 [250 -- 2 097 152]

측정 단위 페이지(4KB)

관련 매개변수 414 페이지의 『정렬 힙 *크기*(sortheap)』

개별 및 공유 정렬에서는 두 개의 서로 다른 메모리 소스에서 메모리를 사용합니다. 공유 정렬 메모리 영역의 *크기*는 *sheapthres*의 값을 기초로 데이터베이스에 처음으로 연결될 때 통계적으로 사전에 결정됩니다. 개별 정렬 메모리 영역의 *크기*에는 제한이 없습니다.

sheapthres 매개변수는 개별 정렬과 공유 정렬에 대해 다르게 사용됩니다.

- 개별 정렬의 경우, 이 매개변수는 주어진 시간에 개별 정렬에 소요될 수 있는 총 메모리 양에서의 인스턴스 전반의 소프트 한계입니다. 인스턴스의 개별 정렬 메모리 소요량이 이 한계에 도달하면 들어오는 추가 개별 정렬 요청에 대해 할당되는 메모리는 감소될 것으로 간주됩니다.
- 공유 정렬의 경우, 이 매개변수는 주어진 시간에 공유 정렬에 소요될 수 있는 총 메모리 양에서의 데이터베이스 전반의 하드 한계입니다. 이 한계에 도달하면 어떠한 추가 공유 정렬 요청도 허용되지 않습니다(총 공유 정렬 메모리 소요량이 *sheapthres*에 지정된 한계치 이하가 될 때까지).

정렬 힙을 사용하는 이러한 조작의 예에는 정렬, 해쉬 조인, 동적 비트맵(색인 AND 작업 및 Star Join에 사용) 및 테이블이 메모리에 있는 조작이 있습니다.

임계값을 명시적으로 정의하면 데이터베이스 관리 프로그램이 많은 수의 정렬에 과다한 메모리를 사용하는 것을 방지할 수 있습니다.

단일 노드에서 다중 노드 환경으로 이동할 때 매개변수의 값이 증가하는 이유는 없습니다. 단일 노드(DB2 EE에서) 환경에서 데이터베이스와 데이터베이스 관리 프로그램 구성 매개변수로 변환하면 대부분의 동일한 다중 노드(DB2 EEE에서) 환경에서 제대로 작동합니다.

데이터베이스 관리 프로그램 구성 매개변수와 같은 정렬 힙 임계값 매개변수는 전체 DB2 인스턴스에 적용됩니다. 이 매개변수를 다른 노드 또는 파티션에서 다른 값으로 설정하는 방법은 둘 이상의 DB2 인스턴스를 작성하는 것입니다. 이러한 경

우에 다른 노드 그룹에서 다른 DB2 데이터베이스를 관리해야 합니다. 이러한 배열은 파티션된 데이터베이스 환경의 여러 가지 이점을 무효하게 만듭니다.

권장사항: 데이터베이스 관리 프로그램 인스턴스에 있는 가장 큰 *sortheap* 매개변수의 합당한 배수로 이 매개변수를 설정하는 것이 가장 좋습니다. 이 매개변수는 인스턴스 내의 데이터베이스에 정의되어 있는 최대 *sortheap*의 **최소** 두 배는 되어야 합니다.

개별 정렬을 수행하고 시스템 메모리가 제한되지 않을 경우에는 다음 단계를 통해 이 매개변수에 맞는 최적의 값을 계산할 수 있습니다.

1. 각 데이터베이스에 대해 일반적인 정렬 힙 사용 계산은 다음과 같습니다.

   ```
   (데이터베이스에 대해 수행되는 동시 에이전트의 일반적인 수)
   * (sortheap(해당 데이터베이스에서 정의됨))
   ```

2. 위 결과의 합계를 계산하면 인스턴스 내에서 모든 데이터베이스의 일반적인 환경에서 사용할 수 있는 총 정렬 힙이 제공됩니다.

SMP 환경에서 정렬에 대한 자세한 내용은 221 페이지의 『병렬 정렬 전략』을 참조하십시오.

정렬 성능과 메모리 사용 사이의 적절한 균형을 위해 이 매개변수를 조정하려면 벤치마킹 기법을 사용해야 합니다. 자세한 내용은 363 페이지의 『제12장 벤치마크 테스팅』을 참조하십시오.

정렬에 대한 자세한 내용은 300 페이지의 『정렬』을 참조하십시오.

데이터베이스 시스템 모니터를 사용하여 정렬 활동을 추적할 수도 있습니다.

자세한 내용은 *시스템 모니터 안내 및 참조서*의 포스트 임계값 정렬 (*post_threshold_sorts*) 모니터 요소 설명을 참조하십시오.

명령문 힙 크기(stmtheap)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	2048 [128 - 60 000]
측정 단위	페이지 (4KB)

할당 시기 각 명령문의 사전 처리 컴파일 또는 바인딩 도중

해제 시기 각 명령문의 사전 처리 컴파일 또는 바인딩이 완료될 때

명령문 힙은 SQL문의 컴파일 중 SQL 컴파일러의 작업 공간으로 사용됩니다. 이 매개변수는 이 작업 공간의 크기를 지정합니다.

이 영역은 영구적으로 할당된 상태에 있는 것은 아니지만, 처리된 모든 SQL문에 할당되고 해제됩니다. 동적 SQL문의 경우, 이 작업 영역은 프로그램 실행 중에 사용됩니다. 반면, 정적 SQL문의 경우에는 프로그램 실행 중이 아닌 바인드 프로세스 중에 사용됩니다.

권장사항: 대부분의 경우, 이 매개변수의 기본값을 사용하면 됩니다. 매우 큰 SQL문이 있는데 데이터베이스 관리 프로그램이 명령문의 최적화를 시도할 때 오류(명령문이 너무 복잡함)를 발행한 경우, 오류가 해결될 때까지 규칙적인 증분(256 또는 1024)으로 이 매개변수 값을 증가시켜야 합니다.

응용프로그램 힙 크기(applheapsz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 128 [16 - 60 000]

64 [16 - 60 000] (파티션된 데이터베이스 환경)

측정 단위 페이지(4KB)

할당 시기 응용프로그램에 대한 작업을 수행하기 위해 에이전트가 초기화될 때

해제 시기 에이전트가 응용프로그램에 대한 작업을 완료할 때

관련 매개변수 412 페이지의 『응용프로그램 제어 힙 크기 (app_ctl_heap_sz)』

이 매개변수는 특정 에이전트 또는 서브에이전트 대신 데이터베이스 관리 프로그램이 사용할 수 있는 개인용 메모리 페이지의 수를 정의합니다.

힙은 응용프로그램에 대해 에이전트나 서브에이전트가 초기화될 때 할당됩니다. 할당되는 양은 에이전트 또는 서브에이전트에 주어진 요청을 처리하는 데 필요한 최소량이 됩니다. 에이전트 또는 서브에이전트가 더 큰 SQL문을 처리하기 위해 더 많은 힙 공간을 필요로 하는 경우, 데이터베이스 관리 프로그램이 필요에 따라 메모리를 이 매개변수가 지정하는 최대량까지 할당합니다.

주: 파티션된 데이터베이스 환경에서 에이전트 및 서브에이전트에 대한 SQL문의 실행 섹션 사본을 저장하는 데 응용프로그램 제어 힙 *app_ctl_heap_sz*)이 사용됩니다. 그러나 다른 모든 환경의 에이전트처럼 SMP 서브에이전트도 *applheapsz*를 사용합니다.

권장사항: 응용프로그램 힙에 충분한 저장영역이 없다는 오류가 수신되면 이 매개변수를 값을 늘리십시오.

응용프로그램 힙(*applheapsz*)은 에이전트 개인용 메모리로부터 할당됩니다.

통계 힙 크기(stat_heap_sz)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	4384 [1096 - 524 288]
측정 단위	페이지(4KB)
할당 시기	RUNSTATS 유틸리티가 시작될 때
해제 시기	RUNSTATS 유틸리티가 완료될 때
관련 매개변수	• 516 페이지의 『보유되어 자주 사용되는 값의 수(num_freqvalues)』 • 517 페이지의 『컬럼에 대한 quantile 수(num_quantiles)』

이 매개변수는 RUNSTATS 명령으로 통계를 수집하는 데 사용된 힙의 **최대** *크기*를 나타냅니다.

권장사항: 분산 통계가 수집되지 않거나 상대적으로 좁은 테이블에 대해서만 분산 통계가 수집될 경우, 기본값을 사용하는 것이 적절합니다. 분산 통계가 수집 중일 때 최소값은 권장되지 **않습니다.** 왜냐하면, 1 컬럼 또는 2 컬럼이 들어 있는 테이블만이 힙에 적합하기 때문입니다.

이 매개변수 값을 통계가 수집되고 있는 컬럼 수에 따라 조정해야 합니다. 약간의 컬럼만 있는 작은 테이블은 분산 통계가 수집되는 데 더 적은 메모리가 필요합니다. 많은 컬럼이 있는 큰 테이블은 충분히 큰 메모리가 필요합니다. 매우 큰 테이블에 대한 분산 통계를 수집 중이어서 큰 통계 힙이 필요한 경우, 시스템 활동이 적은 기간에 통계를 수집하여 다른 사용자의 메모리 요구사항에 방해되지 않기를 원합니다.

조회 힙 크기(query_heap_sz)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	1000 [2 - 524 288]
측정 단위	페이지(4KB)
할당 시기	응용프로그램(지역 또는 원격)이 데이터베이스에 연결될 때
해제 시기	원격 응용프로그램이 데이터베이스로부터 연결 해제되거나 인스턴스에서 접속 해제될 때

관련 매개변수 429 페이지의『응용프로그램 지원 계층 힙 크기 (aslheapsz)』

이 매개변수는 조회 힙에 할당할 수 있는 메모리의 **최대** 양을 지정합니다. 조회 힙은 에이전트의 개인용 메모리에 각 조회를 저장하는 데 사용됩니다. 각 조회의 정보는 입력 및 출력 SQLDA, 명령문 텍스트, SQLCA, 패키지 이름, 작성자, 섹션 번호 및 일관성 토큰 등으로 이루어집니다. 이 매개변수는 응용프로그램이 에이전트 내의 가상 메모리를 불필요하게 많이 소비하지 않도록 하기 위해 제공됩니다.

조회 힙은 블로킹 커서에 할당된 메모리에도 사용됩니다. 이 메모리는 커서 제어 블록 및 완전 분석된 출력 SQLDA로 구성됩니다.

할당된 초기 조회 힙은 *aslheapsz* 매개변수에 의해 지정된 것처럼 응용프로그램 지원 계층 힙과 같은 크기입니다. 조회 힙 크기는 2 이상이어야 하며, *aslheapsz* 매개변수보다 커야 합니다. 이 조회 힙이 크지 않아 주어진 요청을 처리할 수 없는 경우, (*query_heap_sz*를 넘지 않는) 요청에 필요한 크기로 재할당됩니다. 이 새 조회 힙의 크기가 *aslheapsz*의 1.5배가 넘으면 조회가 종료할 때 *aslheapsz*의 크기로 조회 힙이 재할당됩니다.

권장사항: 대부분의 경우, 기본값을 사용하는 것이 좋습니다. 최소한, *query_heap_sz*를 *aslheapsz*보다 5배가 큰 크기로 설정해야 합니다. 이는 *aslheapsz*보다 큰 조회를 가능하게 하며, 주어진 시간에 열리는 세 개 또는 네 개의 블로킹 커서에 추가 메모리를 제공합니다.

매우 큰 LOB가 있을 경우, 조회 힙이 이 LOB를 수용할 수 있도록 이 매개변수의 값을 증가시켜야 합니다.

DRDA 힙 크기(drda_heap_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트

- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 128 [16 - 60 000]

측정 단위 페이지(4KB)

할당 시기

- DRDA 응용프로그램 서버(AS)는 DRDA 응용프로그램 리퀘스터(AR)가 DB2 데이터베이스에 연결될 때마다 DRDA 힙을 할당합니다.
- DB2 Connect는 DRDA AS에 연결될 때마다 DRDA 힙을 할당합니다.

해제 시기 DRDA AR이 데이터베이스로부터 연결 해제될 때

이 매개변수는 DB2 Connect 및 DRDA 응용프로그램 서버(AS) 지원 기능에 의해 사용되는 메모리에 할당할 페이지 수를 나타냅니다. 다음 항목은 이 힙으로부터 할당되는 메모리 양에 영향을 미칩니다.

- 응용프로그램이 여는 커서의 수
- 입력 호스트 변수의 수
- 선택 목록 항목의 수
- I/O 데이터의 크기
- 바인드되거나 준비 중인 SQL문의 길이

권장사항: DRDA 힙 메모리가 충분하지 않다는 오류 코드가 수신되지 않으면 기본값을 사용하십시오.

UDF 공유 메모리 세트 크기(udf_mem_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형	구성 가능
기본 [범위]	256 [128 - 60 000]
측정 단위	페이지(4KB)
할당 시기	UDF가 시작될 때
해제 시기	UDF가 완료될 때

이 매개변수는 분리(fenced) 및 비분리(unfenced) 사용자 정의 함수(UDF)에서 공통적인 매개변수입니다. 분리(fenced) UDF의 경우, 데이터베이스 프로세스와 UDF 사이에서 공유되는 메모리의 기본 할당을 지정합니다. 단일 파티션된 데이터베이스 환경에는 단 하나의 공유 메모리 세트만 있습니다. 파티션된 데이터베이스 환경에는 각 데이터베이스 파티션 서버에 대해 하나의 공유 메모리 세트가 존재하며, 해당 서버에서 실행되는 모든 응용프로그램 에이전트 및 서브에이전트가 동일한 공유 메모리 세트를 사용합니다.

비분리(unfenced) UDF의 경우, 매개변수가 개인용 메모리 세트의 크기를 지정합니다. 단일 파티션된 데이터베이스 환경에서는 개별 메모리로부터 힙이 할당됩니다. 파티션된 데이터베이스 환경에서는 각 데이터베이스 파티션 서버에 대한 응용프로그램 전역 메모리로부터 힙(heap)이 할당되며, 해당 데이터베이스 파티션 서버의 응용프로그램 대신에 수행 중인 모든 에이전트 및 서브에이전트가 동일한 공유 메모리 세트를 사용합니다.

분리 및 비분리 UDF의 경우 모두, 이 메모리는 데이터를 UDF로 전달하고 다시 데이터베이스로 전달하는 데 사용됩니다.

응용프로그램에서 사용된 UDF가 없는 경우, 메모리는 할당되지 않습니다. 분리 UDF와 비분리 UDF가 둘다 같은 응용프로그램에서 수행될 경우, 분리 UDF 메모리 할당과 비분리 UDF 메모리 할당이 일어납니다.

사용자 정의 함수(UDF)에 대한 자세한 내용은 *응용프로그램 개발 안내서* 및 *SQL 참조서*를 참조하십시오.

권장사항: 기본 설정은 UDF로 LOB 데이터를 전달하는 것을 제외하고 모든 경우에 적절해야 합니다. LOB 데이터를 UDF로 전달하는 경우, 할당된 메모리 양을 증가시킬 필요가 있습니다. 이 매개변수 값을 입력 인수 크기 및 외부 함수의 결과보다 최소 2페이지는 크게 설정해야 합니다.

주: UDF의 메모리 요구사항은 습관성이 있으므로, 응용프로그램에서 참조되는 UDF의 수는 이 매개변수의 최적 설정에 영향을 줍니다.

에이전트 스택 크기(agent_stack_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

OS/2 64 [8 - 1000]

Windows NT 16 [8 - 1000]

측정 단위 페이지(4KB)

할당 시기 응용프로그램에 대한 작업을 수행하기 위해 에이전트가 초기화될 때

해제 시기 에이전트가 응용프로그램에 대한 작업을 완료할 때

에이전트 스택은 DB2가 각 에이전트에 할당한 가상 메모리입니다. 이 메모리는 SQL문을 처리해야 할 때 확약됩니다. 이 매개변수를 사용하여 주어진 응용프로그램 세트용 서버의 메모리 사용을 최적화할 수 있습니다. 복합 조회일수록 단순한 조회에 비해 더 많은 스택 공간을 사용합니다.

이 매개변수는 UNIX 기반 플랫폼에는 적용되지 않습니다.

권장사항: 대부분의 경우, 기본 스택 크기를 사용할 수 있습니다. 환경에 매우 복합 조회가 많이 있는 경우에만 이 매개변수 값을 증가시킬 필요가 있습니다. 스택 크기가 SQL문을 처리할 만큼 충분히 크지 않을 경우, 오류가 db2diag.log 파일에 기록되고 SQL 코드가 발행됩니다. *agent_stack_sz*를 증가시키고 데이터베이스 인스턴스를 재시작해야 합니다.

사용자의 환경이 다음과 일치하면 기타 클라이언트에서 더 많은 주소 공간을 사용할 수 있도록 스택 크기를 줄일 수 있습니다.
- 복합 조회가 없는 단순한 응용프로그램(예: 간단한 OLTP)만 포함하는 경우
- 상대적으로 많은 수의 동시 클라이언트(예: 100 이상)가 필요한 경우

에이전트 스택 크기와 동시 클라이언트의 수는 역의 관계에 있습니다. 예를 들어, 스택 크기가 크면 수행할 수 있는 동시 클라이언트의 잠재적인 수는 줄게 됩니다. 이러한 상황은 주소 공간이 OS/2 및 Windows NT 플랫폼에서 제한되기 때문에 발생합니다. 예를 들어, OS/2에서 400MB의 주소 공간을 가지고 있다고 가정해 보십시오(`config.sys` 파일에 따라 용량은 달라짐). *agent_stack_sz* 값을 1MB로 설정할 경우, 400MB가 넘는 에이전트를 확보할 수는 없습니다. (사실상, 버퍼 풀과 같은 주소 공간에 대한 다른 요구사항으로 인해 더 적은 수의 에이전트가 확보될 수도 있습니다.) 즉, *maxagents*를 큰 값(예: 5000)으로 설정할 경우, 이 한계에 결코 도달할 수 없게 됩니다.

최소의 확약된 개인용 메모리(min_priv_mem)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 32 [32 - 112 000]

측정 단위 페이지(4KB)

할당 시기 데이터베이스 관리 프로그램이 시작될 때

해제 시기 데이터베이스 관리 프로그램이 중지될 때

관련 매개변수 『개인용 메모리 임계값(priv_mem_thresh)』

이 매개변수는 데이터베이스 관리 프로그램 인스턴스가 시작될 때(db2start) 데이터베이스 서버 프로세스가 개인용 가상 메모리로 확보할 페이지 수를 지정합니다. 서버에 더 많은 개인용 메모리가 필요한 경우, 그때마다 운영 체제로부터 더 많이 확보하려고 합니다.

이 매개변수는 UNIX 기반 시스템에는 적용되지 않습니다.

권장사항: 기본값을 사용하십시오.

데이터베이스 서버에 더 많은 메모리를 확약하려면 이 매개변수 값을 변경하기만 하면 됩니다. 이렇게 하면 할당 시간이 절약됩니다. 그러나 이 값은 DB2 이외의 응용프로그램 성능에 영향을 주기 때문에, 이 값을 너무 높게 설정하지 않도록 주의해야 합니다.

개인용 메모리 임계값(priv_mem_thresh)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 1296 [-1; 32 - 112 000]

지역 클라이언트가 있는 위성 데이터베이스 서버에서 32 [-1; 32 - 112 000]

측정 단위 페이지(4KB)

관련 매개변수 425 페이지의『최소의 확약된 개인용 메모리 (min_priv_mem)』

이 매개변수는 할당된 상태로 있으면서 시작되는 새 에이전트가 사용할 준비를 갖추고 있는 미사용 에이전트 개인용 메모리의 *크기*를 결정하는 데 사용됩니다. 이것은 UNIX 기반 플랫폼에는 적용되지 않습니다.

에이전트가 종료되면 해당 에이전트에서 사용한 모든 메모리를 자동으로 할당 해제하는 대신 데이터베이스 관리 프로그램이 다음 공식으로 결정되는 *여분의 메모리 할당*을 할당 해제할 뿐입니다.

```
할당된 개인용 메모리 -
(사용된 개인용 메모리 + priv_mem_thresh)
```

이 공식의 결과, 음수가 생성되면 아무런 조치도 취해지지 않습니다.

다음 표에서는 메모리 할당 및 할당 해제 시기를 보여주는 예를 제공합니다. 이 예는 *priv_mem_thresh*에 대해 임의의 설정으로 100을 사용합니다.

조치 설명	할당된 메모리	사용된 메모리
많은 에이전트가 수행 중이며 메모리를 할당했습니다.	1000	1000
새 에이전트가 시작되며 100페이지의 메모리를 사용합니다.	1100	1100
200페이지의 메모리를 사용하는 에이전트가 종료됩니다. (100페이지의 메모리가 해제되고, 100페이지의 메모리는 나중에 사용될 수 있도록 할당된 채 남아 있습니다.)	1000	900
50페이지의 메모리를 사용하는 에이전트가 종료됩니다. (50페이지의 메모리가 해제되며 여분의 100페이지는 기존 에이전트에 의해 사용 중인 것과 비교하여 할당된 채 남아 있습니다.)	950	850
새 에이전트가 시작되며 150페이지의 메모리가 필요합니다. (150페이지 중 100페이지는 이미 할당되어 있으며 데이터베이스 관리 프로그램은 이 에이전트용으로 50페이지의 추가 페이지만 할당하면 됩니다.)	1000	1000

값『-1』은 이 매개변수가 *min_priv_mem* 매개변수의 값을 사용하게 합니다.

권장사항: 이 매개변수를 설정할 때에는 동일한 시스템의 다른 프로세스의 메모리 요구사항뿐 아니라 클라이언트 연결/연결 해제 패턴을 고려해야 합니다.

많은 클라이언트가 동시에 데이터베이스에 연결되는 동안 짧은 주기만이 있는 경우, 상위 임계값을 통해 미사용 메모리가 확약되지 않은 상태로 있도록 하고 다른 프로세스에서도 사용 가능하게 합니다. 이 때 메모리 관리가 소홀하여 메모리를 필요로 하는 다른 프로세스에 영향을 줄 수 있습니다.

동시 클라이언트의 수가 좀더 일정하고 이 수가 자주 변경되는 경우, 상위 임계값은 메모리를 클라이언트 프로세스에 사용할 수 있도록 하고 메모리 할당 및 할당 해제의 오버헤드를 줄이는 데 도움이 됩니다.

에이전트/응용프로그램 통신 메모리

다음 매개변수는 응용프로그램과 에이전트 프로세스간의 데이터 전달을 위해 할당되는 메모리의 양에 영향을 미칩니다.

- 429 페이지의『응용프로그램 지원 계층 힙 크기(aslheapsz)』
- 431 페이지의『DECDIV3(십진수 나누기 배율 3) (min_dec_div_3)』

- 432 페이지의『클라이언트 I/O 블록 *크기*(rqrioblk)』
- 434 페이지의『DOS 리퀘스터 I/O 블록 *크기*(dos_rqrioblk)』

이 에이전트/응용프로그램 공유 메모리가 데이터베이스 관리 프로그램에 의해 할당된 나머지 메모리와 어떤 연관성을 갖는지에 대한 자세한 내용은 274 페이지의『DB2의 메모리 사용법』을 참조하십시오.

응용프로그램 지원 계층 힙 크기(aslheapsz)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	15 [1 - 524 288]
측정 단위	페이지(4KB)
할당 시기	지역 응용프로그램에 대해 데이터베이스 관리 프로그램 에이전트 프로스세스가 시작될 때
해제 시기	데이터베이스 관리 프로그램 에이전트 프로세스가 종료될 때
관련 매개변수	420 페이지의『조회 힙 *크기*(query_heap_sz)』

응용프로그램 지원 계층 힙은 지역 응용프로그램과 관련 에이전트 사이의 통신 버퍼를 나타냅니다. 이 버퍼는 시작된 각 데이터베이스 관리 프로그램 에이전트에 의해 공유 메모리로 할당됩니다.

데이터베이스 관리 프로그램에 대한 요청 또는 관련 응답이 버퍼에 맞지 않는 경우, 둘 이상의 송수신 쌍으로 분할됩니다. 이 버퍼의 *크기*는 단일 송수신 쌍을 사

용하여 대부분의 요청을 처리하도록 설정해야 합니다. 요청의 *크기*는 다음을 보유하는 데 필요한 저장영역에 따라 다릅니다.

- 입력 SQLDA
- SQLVAR 내의 모든 관련 데이터
- 출력 SQLDA
- 보통 259바이트를 넘지 않는 다른 필드

이 통신 버퍼 외에도, 이 매개변수는 블로킹 커서가 열려 있을 때 I/O 블록 *크기*를 결정하는 데에도 사용됩니다. 블록화된 커서의 메모리는 응용프로그램의 개인용 주소 공간 내에서 할당되기 때문에 각 응용프로그램에 할당되는 개인용 메모리의 최적 양을 판별해야 합니다. 데이터베이스 클라이언트가 응용프로그램의 개인용 메모리 내에서 블로킹 커서에 공간을 할당할 수 없는 경우, 비 블로킹 커서가 열립니다.

지역 응용프로그램에서 전송된 데이터는 데이터베이스 관리 프로그램에 의해 조회 힙으로부터 할당된 인접 메모리 세트로 수신됩니다. *aslheapsz* 매개변수는 조회 힙(지역 및 원격 클라이언트에 대해)의 *초기 크기*를 결정하는 데 사용됩니다. 조회 힙의 최대 *크기*는 *query_heap_sz* 매개변수에 의해 정의됩니다.

권장사항: 일반적으로 응용프로그램의 요청이 적고 메모리가 넉넉하지 않은 시스템에서 응용프로그램이 수행될 경우, 이 매개변수의 값을 줄일 수 있습니다. 일반적으로 조회가 매우 크며 둘 이상의 송수신 요청이 필요하고 시스템이 메모리 제한을 받지 않는 경우, 이 매개변수 값을 증가시킬 수 있습니다.

다음 공식을 사용하여 *aslheapsz*의 최소 페이지 수를 계산하십시오.

```
aslheapsz >= ( sizeof(input SQLDA)
             + sizeof(each input SQLVAR)
             + sizeof(output SQLDA)
             + 250 ) / 4096
```

여기서 sizeof(x)는 주어진 I/O 값의 페이지 값을 계산하는 x *크기*(바이트 단위)입니다.

또한 이 매개변수가 블로킹 커서의 수와 잠재적인 *크기*에 주는 영향도 고려해야 합니다. 큰 행 블록은 전송 중인 행의 수나 *크기*가 큰 경우(예를 들어, 데이터의 양

이 4 096바이트보다 큰 경우), 성능 향상을 가져올 수 있습니다. 그러나 레코드 블록이 커지면 각 연결의 작업용 메모리 크기를 증가시킨다는 단점이 있습니다.

더 큰 레코드 블록이 실제 응용프로그램에 필요한 것보다 더 많은 페치 요청을 유발하기도 합니다. 이 경우, 응용프로그램에서 SELECT문에 OPTIMIZE FOR절을 페치 요청의 수를 제어할 수 있습니다. OPTIMIZE FOR절에 대한 자세한 내용은 84 페이지의 『OPTIMIZE FOR n ROWS절』을 참조하십시오.

DECDIV3(십진수 나누기 배율 3) (min_dec_div_3)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	아니오 [예, 아니오]

min_dec_div_3 데이터베이스 구성 매개변수의 추가는 SQL에서 십진수 나누기에 대한 배율 계산에 대한 변경을 사용 가능하게 하는 빠른 정의 방법으로 제공됩니다. *min_dec_div_3*은 ″예″ 또는 ″아니오″로 설정될 수 있습니다. *min_dec_div_3*의 기본값은 ″아니오″입니다.

min_dec_div_3 데이터베이스 구성 매개변수는 나누기와 관련한 십진수 산술 연산의 결과 배율을 변경합니다. 값이 ″아니오″이면, 배율은 31-p+s-s'로 계산됩니다. *SQL 참조서*에 대한 자세한 내용은 3장 ″SQL에서 십진수 산술″을 참조하십시오. ″예″로 설정하는 경우 배율은 MAX(3, 31-p+s-s')로 계산됩니다. 이것은 십진수 배율이 항상 최소한 3이 되는 결과를 초래합니다. 정밀도는 항상 31입니다.

이 데이터베이스 구성 매개변수를 변경하면 기존 데이터베이스에 대한 응용프로그램을 변경할 수 있습니다. 이것은 십진수 나누기의 결과 배율이 이 데이터베이스 구성 매개변수를 변경하여 영향을 받을 때 일어날 수 있습니다. 아래의 목록은 응용프로그램에 영향을 줄 수 있는 몇 가지 가능성 있는 시나리오입니다. 이 시나리오는 기존 데이터베이스를 사용하는 데이터베이스 서버의 *min_dec_div_3*을 변경하기 전에 고려해야 합니다.

- 뷰 컬럼 중 하나의 결과 배율이 변경된 경우, 한 가지 설정으로 환경에 정의된 뷰는 데이터베이스 구성 매개변수가 변경된 후 참조될 때 SQLCODE -344로

실패할 수 있습니다. 메시지 SQL0344N은 순환 공통 테이블 표현식을 나타내지만, 오브젝트 이름(첫 번째 토큰)이 뷰이면 이 오류를 피하기 위해 뷰를 삭제한 다음 다시 뷰를 작성해야 합니다.

- 통계 패키지는 패키지가 내시적으로 또는 명시적으로 리바인드될 때까지 작동을 변경하지 않습니다. 예를 들어, 아니오에서 예로 값을 변경한 후, 추가 배율 숫자는 라바인드가 발생할 때까지 결과에 포함되지 않을 수 있습니다. 변경된 통계 패키지의 경우, 명시적 REBIND 명령을 사용하여 리바인드를 강제 실행할 수 있습니다.
- 십진수 나누기와 관련한 점검 제한조건은 이전에 승인된 일부 값을 제한할 수 있습니다. 이제 이 같은 행은 제한조건을 위반하지만 점검 제한조건 행에 포함된 컬럼 중 하나가 갱신되거나 IMMEDIATE CHECKED 옵션으로 SET INTEGRITY문이 처리될 때까지 검출되지 않습니다. 이 같은 제한조건의 점검을 강제 실행하려면 점검 제한조건을 삭제한 다음 ALTER TABLE문을 다시 제한조건에 추가하기 위해 TABLE ALTER문을 수행합니다.

주: 또한 DB2 버전 7은 다음과 같은 한계가 있습니다.

1. GET DB CFG FOR DBNAME 명령은 *min_dec_div_3* 설정을 표시하지 않습니다. 현재 설정을 판별하는 가장 좋은 방법은 십진수 나누기 결과의 효과 측면을 관찰하는 것입니다. 예를 들어, 다음 명령문을 고려해 보십시오.

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   이 명령문이 sqlcode SQL0419N을 리턴하면 데이터베이스는 *min_dec_div_3* 지원이 없거나 "아니오"로 설정된 것입니다. 명령문이 1.000으로 리턴되면 *min_dec_div_3*이 "예"로 설정된 것입니다.
2. *min_dec_div_3*은 ? UPDATE DB CFG 명령을 수행할 때 구성 키워드 목록에 나타나지 않습니다.

클라이언트 I/O 블록 크기(rqrioblk)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버

- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 32 767 [4 096 - 65 535]

측정 단위 바이트

할당 시기

- 원격 클라이언트 응용프로그램이 서버 데이터베이스에 대한 연결 요청을 발행할 때
- 블로킹 커서가 열리고, 클라이언트에서 추가로 블록이 열릴 때

해제 시기

- 원격 응용프로그램이 서버 데이터베이스로부터 연결 해제될 때
- 블로킹 커서가 닫힐 때

관련 매개변수 434 페이지의 『DOS 리퀘스터 I/O 블록 *크기* (dos_rqrioblk)』

이 매개변수는 데이터베이스 서버에서 원격 응용프로그램과 데이터베이스 에이전트 사이의 통신 버퍼 *크기*를 지정합니다. 데이터베이스 클라이언트가 원격 데이터베이스로 연결을 요청할 때 이 통신 버퍼가 클라이언트에 할당됩니다. 데이터베이스 서버에서 연결이 설정되고 서버가 클라이언트에서 *rqrioblk* 값을 판정할 수 있을 때까지 32 767바이트의 통신 버퍼가 *초기*에 할당됩니다. 일단 서버가 이 값을 알게 되면 클라이언트의 버퍼가 32 767바이트가 아닐 경우 서버는 통신 버퍼를 재할당합니다.

이 통신 버퍼 외에도, 이 매개변수는 블로킹 커서가 열려 있을 때 데이터베이스 클라이언트에서 I/O 블록 *크기*를 결정하는 데에도 사용됩니다. 블록화된 커서의 메모리는 응용프로그램의 개인용 주소 공간 내에서 할당되기 때문에 각 응용프로그

램에 할당되는 개인용 메모리의 최적 양을 판별해야 합니다. 데이터베이스 클라이언트가 응용프로그램의 개인용 메모리 내에서 블로킹 커서에 공간을 할당할 수 없는 경우, 비 블로킹 커서가 열립니다.

권장사항: 비 블로킹 커서의 경우, 이 매개변수의 값을 증가시키는 이유는 단일 SQL 문에 의해 전송되는 데이터(예: 대형 오브젝트(LOB) 데이터)가 너무 커서 기본값으로 부족하기 때문입니다.

또한 이 매개변수가 블로킹 커서의 수와 잠재적인 *크기*에 주는 영향도 고려해야 합니다. 큰 행 블록은 전송 중인 행의 수나 *크기*가 큰 경우(예를 들면, 데이터의 양이 4 096바이트보다 큰 경우), 성능 향상을 가져올 수 있습니다. 그러나 레코드 블록이 커지면 각 연결의 작업용 메모리 *크기*를 증가시킨다는 단점이 있습니다.

더 큰 레코드 블록이 실제 응용프로그램에 필요한 것보다 더 많은 페치 요청을 유발하기도 합니다. 이 경우, 응용프로그램에서 SELECT문에 OPTIMIZE FOR절을 사용하여 페치 요청의 수를 제어할 수 있습니다. OPTIMIZE FOR절에 대한 자세한 내용은 84 페이지의 『OPTIMIZE FOR n ROWS절』을 참조하십시오.

DOS 리퀘스터 I/O 블록 크기(dos_rqrioblk)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 4 096 [4 096 - 65 535]

측정 단위 바이트

할당 시기

- 원격 DOS 또는 Windows 3.1 클라이언트가 서버 데이터베이스로 연결 요청을 발행할 때
- 블로킹 커서가 열리고, 클라이언트에서 추가로 블록이 열릴 때

해제 시기

- 원격 응용프로그램이 데이터베이스로부터 연결 해제될 때
- 블로킹 커서가 닫힐 때

관련 매개변수 432 페이지의『클라이언트 I/O 블록 크기 (rqrioblk)』

이 매개변수는 데이터베이스 서버에서 DOS/Windows 3.1 응용프로그램과 데이터베이스 에이전트 사이의 통신 버퍼 크기를 지정합니다. 이 매개변수는 DOS/Windows 3.1 클라이언트에서 사용되는 블록에 다른 값을 설정할 수 있다는 점을 제외하고는 *rqrioblk* 매개변수와 유사합니다. DB2 구성 파일에서 Windows 32 비트, OS/2 및 UNIX 클라이언트에 대해 사용되는 *rqrioblk* 매개변수와, DOS 및 Windows 3.1 클라이언트에 대해 사용되는 *dos_rqrioblk* 매개변수를 모두 설정할 수 있습니다.

이 통신 버퍼 외에도, 이 매개변수는 블로킹 커서가 열려 있을 때 데이터베이스 클라이언트에서 I/O 블록 크기를 결정하는 데에도 사용됩니다. 블록화된 커서의 메모리는 응용프로그램의 개인용 주소 공간 내에서 할당되기 때문에 각 응용프로그램에 할당되는 개인용 메모리의 최적 양을 판별해야 합니다. 데이터베이스 클라이언트가 응용프로그램의 개인용 메모리 내에서 블로킹 커서에 공간을 할당할 수 없는 경우, 비 블로킹 커서가 열립니다.

권장사항: 비 블로킹 커서의 경우, 이 매개변수의 값을 증가시키는 이유는 단일 SQL 문에 의해 전송되는 데이터(예: 대형 오브젝트(LOB) 데이터)가 너무 커서 기본값으로 부족하기 때문입니다.

또한 이 매개변수가 블로킹 커서의 수와 잠재적인 크기에 주는 영향도 고려해야 합니다. 큰 행 블록은 전송 중인 행의 수나 크기가 큰 경우(예를 들면, 데이터의 양

이 4 096바이트보다 큰 경우), 성능 향상을 가져올 수 있습니다. 그러나 레코드 블록이 커지면 각 연결의 작업용 메모리 크기를 증가시킨다는 단점이 있습니다.

더 큰 레코드 블록이 실제 응용프로그램에 필요한 것보다 더 많은 페치 요청을 유발하기도 합니다. 이 경우, 응용프로그램에서 SELECT문에 OPTIMIZE FOR절을 사용하여 페치 요청의 수를 제어할 수 있습니다. OPTIMIZE FOR절에 대한 자세한 내용은 84 페이지의『OPTIMIZE FOR n ROWS절』을 참조하십시오.

데이터베이스 관리 프로그램 인스턴스 메모리

다음 매개변수는 인스턴스 레벨에서 할당 및 사용되는 메모리에 영향을 미칩니다.

- 『데이터베이스 시스템 모니터 힙 크기(mon_heap_sz)』
- 438 페이지의『디렉토리 캐쉬 지원(dir_cache)』
- 440 페이지의『감사 버퍼 크기(audit_buf_sz)』
- 441 페이지의『Java 인터프리터의 최대 힙 크기(java_heap_sz)』

데이터베이스 시스템 모니터 힙 크기(mon_heap_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

UNIX 56 [0 - 60 000]

지역 및 원격 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버 및 지역 클라

이언트가 있는 위성 데이터베이스 서버
32 [0 - 60 000]

지역 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
12 [0 - 60 000]

측정 단위 페이지(4KB)

할당 시기 데이터베이스 관리 프로그램이 *db2start* 명령으로 시작될 때

해제 시기 데이터베이스 관리 프로그램이 *db2stop* 명령으로 중지될 때

관련 매개변수 549 페이지의 『기본 데이터베이스 시스템 모니터 스위치(dft_monswitches)』

이 매개변수는 데이터베이스 시스템 모니터 데이터에 할당할 메모리의 양을 페이지 수로 결정합니다. 메모리는 스냅샷 확보, 모니터 스위치 작동 시작, 모니터 재설정 또는 이벤트 모니터 활성화와 같은 데이터베이스 모니터링 활동을 수행할 때 모니터 힙으로부터 할당됩니다.

0 값은 데이터베이스 관리 프로그램이 데이터베이스 시스템 모니터 데이터를 수집하는 것을 막습니다.

권장사항: 모니터링 활동에 필요한 메모리 양은 모니터링 응용프로그램의 수(스냅샷 또는 이벤트 모니터를 취하는 응용프로그램), 설정된 스위치, 데이터베이스 활동 레벨에 따라 달라집니다.

다음 공식은 모니터 힙(heap)에 필요한 페이지 수의 근사치를 제공합니다.

```
( 모니터링 응용프로그램의 수 + 1 ) *
( 데이터베이스의 수 *
  (800 + ( 액세스된 테이블의 수 * 20 )
   + ( ( 연결된 응용프로그램의 수 + 1) *
       (200 + (테이블 공간의 수 * 100) ) ) ) )
/ 4096
```

이 힙(heap)에서 사용할 수 있는 메모리가 없어지면 다음 중 하나가 발생합니다.

- 이 이벤트 모니터가 정의된 데이터베이스에 첫 번째 응용프로그램을 연결할 때 레벨 2 오류 메시지가 db2alert.log 및 db2diag.log 파일에 기록됩니다.
- SET EVENT MONITOR문을 사용하여 동적으로 시작되는 이벤트 모니터에 실패하면, 오류 코드가 응용프로그램으로 리턴됩니다.
- 모니터 명령 또는 API 서브루틴이 실패하면 오류 코드가 응용프로그램으로 리턴됩니다.

디렉토리 캐쉬 지원(dir_cache)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 예 [예; 아니오]

할당 시기

- 응용프로그램이 첫 번째 연결을 발행하면 개인용 캐쉬가 할당됩니다.
- 데이터베이스 관리 프로그램 인스턴스가 시작되면(db2start) 공유 캐쉬가 할당됩니다.

해제 시기

- 응용프로그램 프로세스가 종료하면 개인용 캐쉬가 해제됩니다.
- 데이터베이스 관리 프로그램 인스턴스가 중지되면(db2stop) 공유 캐쉬가 해제됩니다.

*dir_cache*를 예로 설정하면 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐쉬됩니다. 디렉토리 캐쉬를 사용하면 디렉토리 파일 I/O가 제거되고 디렉토리 정보를 검색하는 데 필요한 디렉토리 검색이 최소화되어 연결 비용이 감소합니다. 디렉토리 캐쉬에는 두 가지 유형이 있습니다.

- 응용프로그램이 수행 중인 머신에서 각 응용프로그램 프로세스에 할당되어 사용되는 개인용 캐쉬
- 내부 데이터베이스 관리 프로그램 프로세스에 할당되어 사용되는 공유 캐쉬

주: 개인용 캐쉬만이 Windows 환경에 적용될 수 있습니다.

개인용 캐쉬의 경우, 응용프로그램이 첫 번째 연결을 발행할 때 각 디렉토리 파일이 읽혀지며 이 응용프로그램의 개인용 메모리에 정보가 캐쉬됩니다. 후속 연결 요청시 응용프로그램 프로세스가 이 캐쉬를 사용하게 되며, 이 캐쉬는 응용프로그램 프로세스가 진행되는 동안 계속 유지보수됩니다. 데이터베이스가 개인용 캐쉬에서 발견되지 않을 경우, 디렉토리 파일에 정보는 있지만, 캐쉬는 갱신되지 않습니다. 응용프로그램이 디렉토리 항목을 수정면 해당 응용프로그램 내에 다음 연결이 발생할 때 이 응용프로그램의 캐쉬가 새로 고쳐집니다. 다른 응용프로그램의 개인용 캐쉬는 새로 고쳐지지 않습니다. 응용프로그램 프로세스가 종료하면 개인용 캐쉬가 해제됩니다. (명령행 처리기 세션에 의해 사용된 디렉토리 캐쉬를 새로 고치려면 `db2 terminate` 명령을 발행하십시오.)

공유 캐쉬의 경우, 데이터베이스 관리 프로그램 인스턴스가 시작될 때(`db2start`), 각 디렉토리 파일이 읽혀지며 공유 메모리에 정보가 캐쉬됩니다. 이 캐쉬는 일부 데이터베이스 관리 프로그램 프로세스에 사용되며 인스턴스가 중지될 때까지 (`db2stop`) 유지보수됩니다. 이 캐쉬에 디렉토리 항목이 없는 경우, 디렉토리 파일에 정보가 있는지 검색됩니다. 인스턴스가 수행되는 도중에는 공유 캐쉬가 새로 고쳐지지 않습니다.

권장사항: 디렉토리 파일이 자주 변경되지 않으며 성능이 중요할 경우, 디렉토리 캐슁 기능을 사용하십시오.

또한 원격 클라이언트에서 응용프로그램이 몇 가지 다른 연결 요청을 발행할 경우, 디렉토리 캐슁이 유용합니다. 이 경우, 캐슁으로 인해 하나의 응용프로그램이 디렉토리 파일을 읽어야 하는 횟수가 줄어들게 됩니다.

디렉토리 캐쉬는 또한 데이터베이스 시스템 모니터 스냅샷 찍기 기능도 증진시킵니다. 그리고 스냅샷 호출시에는 데이터베이스 별명을 사용하지 말고 데이터베이스 이름을 명시적으로 언급해 주어야 합니다.

주: 디렉토리 캐쉬이 활성화된 경우, 그리고 데이터베이스 관리 프로그램이 시작된 후 데이터베이스가 카탈로그화, 카탈로그화 해제, 작성, 또는 삭제될 경우 스냅샷 호출을 수행하면 오류가 발생할 수 있습니다.

감사 버퍼 크기(audit_buf_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 0 [0 - 65 000]

측정 단위 페이지(4KB)

할당 시기 DB2가 시작될 때

해제 시기 DB2가 중지될 때

이 매개변수는 데이터베이스 감사시 사용되는 버퍼의 *크기*를 지정합니다. 감사 기능에 대한 자세한 내용은 *관리 안내서: 구현*의 『DB2 활동 감사』를 참조하십시오.

이 매개변수에 대한 기본값은 0입니다. 값이 0이면 감사 버퍼가 사용되지 않습니다. 값이 0보다 크면 감사 기능이 생성될 때 감사 레코드가 위치하는 감사 버퍼에 공간이 할당됩니다. 값에 4KB 페이지를 곱하면 감사 버퍼에 할당된 공간량이 됩니다. 감사 버퍼는 동적으로 할당될 수 없습니다. DB2는 이 매개변수의 새 값이 적용되기 전에 중지된 다음 재시작되어야 합니다.

이 매개변수의 값을 기본값에서 0보다 큰 값으로 변경하면 감사 기능은 감사 레코드를 생성하는 명령문의 실행과 비교할 때 비동기적으로 디스크에 레코드를 기록합니다. 그러면 DB2 성능은 매개변수 값을 0으로 둘 때보다 향상됩니다. 값 0은 감사 기능이 감사 레코드를 생성하는 명령문의 실행과 비교할 때 동기적으로 (동시에) 디스크에 레코드를 기록한다는 것을 의미합니다. 감사 중 발생하는 동기 조작은 DB2에서 수행되는 응용프로그램의 성능을 저하시킵니다.

Java 인터프리터의 최대 힙 크기(java_heap_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 512 [0 - 4 096]

측정 단위 페이지(4KB)

할당 시기 Java 응용프로그램이 시작될 때

해제 시기 Java 응용프로그램이 완료될 때

관련 매개변수 559 페이지의 『JDK 1.1 설치 경로(jdk11_path)』

이 매개변수는 Java 해석기가 사용하는 최대 힙(heap) 크기를 결정합니다.

각 DB2 프로세스(UNIX 기반 플랫폼의 각 에이전트 또는 서브에이전트마다 하나씩, 다른 플랫폼의 각 인스턴스마다 하나씩)에는 하나의 힙(heap)만이 존재하고, 분리(fenced) UDF 및 분리(fenced) 저장 프로시듀어 프로세스에도 하나의 힙(heap)이 존재합니다. 어느 경우에나 Java UDF 또는 저장 프로시듀어를 수행하는 에이

전트 또는 프로세스만이 이 메모리를 할당합니다. 파티션된 데이터베이스 시스템에서 같은 값이 각 파티션에 사용됩니다.

잠금

다음 매개변수는 사용자 환경에서 잠금 기능이 관리되는 방법에 영향을 미칩니다.

- 『교착 상태 점검을 위한 시간 간격(dlchktime)』
- 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』
- 445 페이지의 『잠금 시간종료(locktimeout)』

또한 자세한 내용은 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』을 참조하십시오.

55 페이지의 『잠금』에서 데이터베이스 관리 프로그램이 잠금 기능을 사용하여 데이터의 무결성을 유지보수하는 방법에 대한 일반 개요를 제공합니다.

교착 상태 점검을 위한 시간 간격(dlchktime)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 10 000(10 초) [1 000 - 600 000]

측정 단위 밀리초

관련 매개변수

- 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
- 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』

교착 상태는 동일한 데이터베이스에 연결된 둘 이상의 응용프로그램이 자원을 무한정 기다릴 때 발생합니다. 각 응용프로그램이 다른 응용프로그램의 계속 수행에 필요한 자원을 보유하고 있기 때문에 기다리는 것은 해결되지 않습니다.

교착 상태 점검 간격은 데이터베이스 관리 프로그램이 데이터베이스에 연결된 모든 응용프로그램 사이에서 교착 상태를 점검하는 빈도를 정의합니다.

주:

1. 파티션된 데이터베이스 환경에서 이 매개변수는 카탈로그 노드에만 적용됩니다.
2. 파티션된 데이터베이스 환경에서는 두 번째 반복 수행 이후에야 교착 상태가 플래그됩니다.

권장사항: 이 매개변수를 늘리면 교착 상태의 점검 빈도 수가 줄어, 응용프로그램이 교착 상태가 해결되기까지 기다려야 하는 시간이 늘어납니다.

이 매개변수를 줄이면 교착 상태의 점검 빈도 수가 늘어, 응용프로그램이 교착 상태가 해결되기까지 기다려야 하는 시간은 줄지만 데이터베이스 관리 프로그램이 교착상태를 점검하기 위한 시간은 늘어납니다. 교착 상태 간격이 너무 작으면 데이터베이스 관리 프로그램이 교착 상태 점검을 자주 수행하기 때문에 런타임 성능이 감소됩니다. 이 매개변수가 동시성 향상을 위해 더 낮게 설정되어 있는 경우, 불필요한 잠금 레벨 자동 업그레이드를 피할 수 있도록 *maxlocks* 및 *locklist*가 적절히 설정되어 있는지 확인해야 합니다. 불필요한 잠금 레벨 자동 업그레이드는 더 많은 잠금 경합을 일으켜 더 많은 교착 상태 상황이 발생하게 합니다.

레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 10 [1 - 100]

OS/2 및 Windows NT
22 [1 - 100]

측정 단위 백분율

관련 매개변수

- 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
- 456 페이지의 『사용 중인 응용프로그램의 최대 수(maxappls)』

잠금 레벨 자동 업그레이드는 행 잠금을 테이블 잠금으로 바꾸어 목록 내의 잠금 수를 줄이는 프로세스입니다. 이 매개변수는 데이터베이스 관리 프로그램이 레벨 자동 업그레이드를 수행하기 전에 완수되어야 하는 응용프로그램이 보유하는 잠금 목록의 비율을 정의합니다. 임의의 하나의 응용프로그램이 보유한 잠금 수가 이 백분율 만큼의 총 잠금 목록 *크기*에 도달하면 그 응용프로그램이 보유한 잠금에 대해 잠금 레벨 자동 업그레이드가 발생합니다. 잠금 레벨 자동 업그레이드는 또한 잠금 목록의 공간이 부족할 때에도 발생합니다.

데이터베이스 관리 프로그램은 응용프로그램의 잠금 목록을 검색하여 대부분의 행이 잠긴 테이블을 찾아 레벨 자동 업그레이드할 잠금을 판별합니다. 이들을 단일 테이블 잠금으로 바꾼 이후 *maxlocks* 값이 더 이상 초과되지 않으면 잠금 레벨 자동 업그레이드가 중단됩니다. 그렇지 않으면 보유되어 있는 잠금 목록의 비율이 *maxlocks* 값 미만이 될 때까지 계속됩니다. *maxlocks* 매개변수에 *maxappls* 매개변수를 곱한 값은 100보다 작을 수 없습니다.

권장사항: 다음 공식을 사용하여 *maxlocks*를 설정하여 응용프로그램이 잠금 평균 값을 두 배로 할 수 있습니다.

```
maxlocks = 2 * 100 / maxappls
```

2는 평균 값의 두배를 얻기 위해 사용하는 것이고, 100은 허용된 최대 백분율을 나타냅니다. 동시에 수행하는 응용프로그램이 몇 개만 있으면 다음 공식을 첫 번째 공식에 대한 대안으로 사용하십시오.

```
maxlocks = 2 * 100 / (동시에 수행 중인 응용프로그램의 평균
수)
```

*maxlocks*를 설정할 때 한 가지 고려할 점은 잠금 목록(*locklist*)의 *크기*와 함께 사용하는 것입니다. 잠금 레벨 자동 업그레이드가 발생하기 전에 응용프로그램이 보유한 잠금 수의 실제 한계는 다음과 같습니다.

```
maxlocks * locklist * 4 096 / (100 * 36)
```

여기서 4 096은 페이지의 바이트 수이고, 100은 *maxlocks*에 대해 허용된 최대 백분율이며 36은 잠금당 바이트 수입니다. 응용프로그램 중 하나가 1 000개의 잠금을 필요로 하고 잠금 레벨 자동 업그레이드가 발생하지 않게 하려면 이 공식에서

maxlocks 및 *locklist*의 값을 선택하여 그 결과가 1 000보다 커지도록 해야 합니다. (*maxlocks*에는 10을, *locklist*에는 100을 사용하면 이 공식은 필요한 1 000개의 잠금보다 더 많은 결과가 나옵니다.)

*maxlocks*가 너무 적게 설정된 경우, 기타 동시 응용프로그램에 대한 잠금 공간이 충분할 때 잠금 레벨 자동 업그레이드가 발생합니다. *maxlocks*가 너무 높게 설정되면 몇 개의 응용프로그램이 대부분의 잠금 공간을 소비할 수 있으며, 기타 응용프로그램은 잠금 레벨 자동 업그레이드를 수행해야 합니다. 이 경우, 잠금 레벨 자동 업그레이드에 대한 요구는 동시성에 좋지 않습니다.

데이터베이스 시스템 모니터를 사용하면 이 구성 매개변수를 추적하고 조정하는 데 도움이 됩니다.

잠금 시간종료(locktimeout)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] -1 [-1; 0 - 30 000]

측정 단위 초

관련 매개변수

- 406 페이지의『잠금 목록용 최대 저장영역(locklist)』
- 443 페이지의『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』

이 매개변수는 응용프로그램이 잠금 확보를 위해 기다리는 시간(초)을 지정합니다. 이로써 응용프로그램의 전체적인 교착 상태를 피할 수 있습니다.

이 매개변수를 0으로 설정하면 잠금은 기다리지 않습니다. 이 상황에서 요청시 사용할 수 있는 잠금이 없으면 응용프로그램이 즉시 -911을 수신합니다.

이 매개변수를 -1로 설정하면 잠금 시간종료 검출 기능이 중단됩니다. 이 상황에서 잠금은 다음 중 하나가 발생할 때까지 기다립니다(요청시 잠금을 사용할 수 없을 때).

- 잠금 권한부여

- 교착 상태 발생

권장사항: 트랜잭션 처리(OLTP) 환경에서는 초기 시작 값을 30초로 할 수 있습니다. 조회 전용 환경에서는 더 높은 값으로 시작할 수도 있습니다. 두 경우 모두, 벤치마킹 기법을 사용하여 이 매개변수를 조정해야 합니다.

DataLinks 관리 프로그램에 대해 작업할 때 DataLinks 관리 프로그램(dlfm) 인스턴스의 db2diag.log에서 잠금 시간종료를 발견할 경우, *locktimeout*의 값을 증가시켜야 합니다. 또한 *locklist*의 값을 증가시키는 것도 고려해야 합니다.

이 값은 트랜잭션이 정지된 것(사용자가 워크스테이션을 떠남으로 인한 것 등)과 같은 비정상 상황으로 인해 기다리는 것을 빨리 검출할 수 있도록 설정되어야 합니다. 충분히 큰 값으로 설정해야 유효한 잠금 요청(잠금을 더 기다려야 하는)이 최대 워크로드로 인해 시간종료하지 않습니다.

데이터베이스 시스템 모니터를 사용하면 응용프로그램(연결)이 잠금 시간종료를 경험한 횟수나, 데이터베이스가 연결된 모든 응용프로그램에 대해 시간종료 상황을 발견한 횟수를 추적하는 데 도움이 됩니다. 자세한 내용은 *시스템 모니터 안내 및 참조서*의 *locks_timeouts*(잠금 시간종료의 수) 모니터 요소 설명을 참조하십시오.

lock_timeout 모니터 요소 값은 다음 이유로 높아질 수 있습니다.

- 이 구성 매개변수에 대해 값이 너무 낮습니다.
- 확장된 기간 동안 잠금을 보유하고 있는 응용프로그램(트랜잭션). 데이터베이스 시스템 모니터를 사용하여 이들 응용프로그램을 더 조사할 수 있습니다.
- 잠금 레벨 자동 업그레이드(행 레벨에서 테이블 레벨로 잠금)로 인해 발생할 수 있는 동시성 문제. 자세한 내용은 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』 및 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』을 참조하십시오.

매개변수의 사용에 대한 자세한 내용은 63 페이지의 『잠금 대기 및 시간종료』를 참조하십시오.

I/O 및 저장

다음 매개변수는 데이터베이스의 조작과 관련된 I/O 및 저장 비용에 영향을 미칠 수 있습니다.

- 『변경된 페이지 임계값(chngpgs_thresh)』
- 448 페이지의 『비동기 페이지 정리자(cleaner)의 수(num_iocleaners)』
- 450 페이지의 『I/O 서버의 수(num_ioservers)』
- 451 페이지의 『색인 정렬 플래그(indexsort)』
- 451 페이지의 『순차적 검출 플래그(seqdetect)』
- 452 페이지의 『기본 프리페치 크기(dft_prefetch_sz)』
- 453 페이지의 『SMS 컨테이너의 기본 수(numsegs)』
- 453 페이지의 『기본 테이블 공간 Extent 크기(dft_extent_sz)』
- 454 페이지의 『확장 저장 메모리 세그먼트 크기(estore_seg_sz)』
- 454 페이지의 『확장 저장영역 메모리 세그먼트의 수(num_estore_segs)』

변경된 페이지 임계값(chngpgs_thresh)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	60 [5 - 99]
측정 단위	백분율
관련 매개변수	448 페이지의 『비동기 페이지 정리자(cleaner)의 수(num_iocleaners)』

비동기 페이지 정리자(cleaner)는 데이터베이스 에이전트가 버퍼 풀 내 공간을 필요로 하기 전에 변경된 페이지를 버퍼 풀에서 디스크로 기록합니다. 그 결과, 데이터베이스 에이전트는 기록될 변경된 페이지를 기다릴 필요가 없으므로, 버퍼 풀의 공간을 사용할 수 있습니다. 이것은 데이터베이스 응용프로그램의 전체 성능을 향상시킵니다.

이 매개변수를 사용하여 비동기 페이지 정리자(cleaner)가 현재 사용되지 않는 경우 시작될 변경된 페이지의 레벨(백분율)을 지정할 수 있습니다. 페이지 정리자(cleaner)가 시작되면 디스크에 기록할 페이지 목록을 작성합니다. 디스크에 이들 페이지를 기록하는 것이 일단 완료되면 다시 비활동 상태가 되고 다음 트리거가 시작되기를 기다립니다.

읽기 전용(예: 조회) 환경에서는 페이지 정리자(cleaner)가 사용되지 않습니다.

권장사항: 갱신 트랜잭션 워크로드가 높은 데이터베이스의 경우, 일반적으로 매개변수 값을 기본값 이하로 설정하여 버퍼 풀에서 정리 페이지를 충분히 확보할 수 있습니다. 매우 큰 테이블에서 데이터베이스 수가 적으면 기본값보다 큰 백분율로 성능을 향상시킬 수 있습니다.

비동기 페이지 정리자(cleaner)의 수(num_iocleaners)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 1 [0 - 255]

측정 단위 카운터

관련 매개변수

- 396 페이지의 『버퍼 풀 크기(buffpage)』
- 447 페이지의 『변경된 페이지 임계값(chngpgs_thresh)』

이 매개변수로 데이터베이스에 대한 비동기 페이지 정리자(cleaner)의 수를 지정할 수 있습니다. 이들 페이지 정리자(cleaner)는 데이터베이스가 버퍼 풀 내 공간을 필요로 하기 전에 변경된 페이지를 버퍼 풀에서 디스크로 기록합니다. 그 결과, 데이터베이스 에이전트는 기록될, 변경된 페이지를 기다릴 필요가 없으므로, 버퍼 풀의 공간을 사용할 수 있습니다. 이것은 데이터베이스 응용프로그램의 전체 성능을 향상시킵니다.

매개변수를 0으로 설정하면 페이지 정리자(cleaner)가 시작되지 않으며, 따라서 데이터베이스 에이전트가 모든 페이지를 버퍼 풀에서 디스크로 기록합니다. 이 매개변수는 많은 실제 저장 장치에 걸쳐 저장된 데이터베이스의 성능에 중요한 영향을 미칠 수 있는데, 이러한 경우에 장치 중 하나가 사용되지 않는 기회가 많아지기 때문입니다. 페이지 정리자(cleaner)가 구성되지 않은 경우, 응용프로그램이 정기적인 로그 가득 참(full) 상태를 발견할 수 있습니다.

데이터베이스에 대한 응용프로그램이 우선적으로 먼저 데이터를 갱신하는 트랜잭션으로 이루어진 경우, 정리자(cleaner)의 수를 늘리면 성능이 향상됩니다. 페이지

정리자(cleaner)를 늘리면 전원 중단과 같은 일시적인 장애로부터의 복구 시간이 줄어듭니다. 이는 디스크에 있는 데이터베이스의 내용이 주어진 시간에서 최근 내용이기 때문입니다.

권장사항: 이 매개변수에 대한 값을 설정할 때 다음 인수를 고려하십시오.

- 응용프로그램 유형
  - 갱신하지 않는 조회 전용 데이터베이스인 경우, 이 매개변수를 0으로 설정하십시오. 조회 워크로드로 많은 TEMP 테이블이 작성될 경우, 예외가 발생합니다. Explain 유틸리티를 사용하여 이를 판별할 수 있습니다.
  - 트랜잭션이 데이터베이스에 대해 수행 중인 경우, 이 매개변수를 1과 데이터베이스에 사용되는 실제 저장 장치의 수 사이에서 설정하십시오.
- 워크로드

  높은 갱신 트랜잭션 비율을 가진 환경에서는 더 많은 페이지 정리자(cleaner)가 구성되어야 합니다.
- 버퍼 풀 *크기*(*buffpage*)

  큰 버퍼 풀을 가진 환경에서도 더 많은 페이지 정리자(cleaner)가 구성되어야 합니다.

버퍼 풀의 쓰기 활동에 대한 이벤트 모니터의 정보를 사용하여 이 구성 매개변수를 조정할 때 데이터베이스 시스템 모니터를 사용할 수 있습니다.

- 다음 조건이 모두 참일 경우, 매개변수를 줄일 수 있습니다.
  - *pool_data_writes*는 대략 *pool_async_data_writes*와 같음
  - *pool_index_writes*는 대략 *pool_async_index_writes*와 같음
- 다음 조건 중 어느 하나가 참일 경우, 매개변수를 증가해야 합니다.
  - *pool_data_writes*가 *pool_async_data_writes*보다 훨씬 큼
  - *pool_index_writes*가 *pool_async_index_writes*보다 훨씬 큼

자세한 내용은 *시스템 모니터 안내 및 참조서*에서 다음의 모니터 요소 설명을 참조하십시오.

- *pool_data_writes*(버퍼 풀 데이터 쓰기)
- *pool_index_writes*(버퍼 풀 색인 쓰기)
- *pool_async_data_writes*(버퍼 풀 비동기 데이터 쓰기)
- *pool_async_index_writes*(버퍼 풀 비동기 색인 쓰기)

I/O 서버의 수(num_ioservers)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	3 [1 - 255] 지역 클라이언트가 있는 위성 데이터베이스 서버에서 1 [1 - 255]
측정 단위	카운터
할당 시기	응용프로그램이 데이터베이스에 연결될 때
해제 시기	응용프로그램이 데이터베이스로부터 연결 해제될 때
관련 매개변수	• 452 페이지의 『기본 프리페치 크기(dft_prefetch_sz)』 • 451 페이지의 『순차적 검출 플래그(seqdetect)』

I/O 서버는 백업 및 복원과 같은 유틸리티에 의한 프리페치 I/O 및 비동기 I/O를 수행하는 데 데이터베이스 에이전트 대신 사용됩니다. 이 매개변수는 데이터베이스에 대한 I/O 서버의 수를 지정합니다. 프리페치 및 유틸리티에 대해 이보다 많은 I/O가 언제라도 데이터베이스에 대해 진행 상태에 있을 수는 없습니다. I/O 서버는 시작한 I/O 조작이 진행되는 동안 대기합니다. 프리페치가 안 된 I/O가 데이터베이스 에이전트에서 직접 스케줄되기로 되어 있어, 결과적으로 *num_ioservers*로 제한되지 않습니다.

권장사항: 시스템의 모든 I/O 장치를 완전 활용하려면 일반적으로 데이터베이스가 상주하는 실제 장치의 수보다 하나 또는 둘 정도 큰 값을 사용하는 것이 좋습니다. 각 I/O 서버와 연관된 최소한의 오버헤드가 있고 미사용 I/O 서버가 유휴 상태로 남아 있기 때문에 I/O 서버를 추가로 구성하는 것이 더 좋습니다.

자세한 내용은 292 페이지의 『버퍼 풀로 데이터 프리페치』 및 295 페이지의 『프리페치 및 병렬 I/O를 위한 I/O 서버 구성』을 참조하십시오.

색인 정렬 플래그(indexsort)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	예 [예; 아니오]

이 매개변수는 색인 작성 도중 색인 키 정렬이 발생하는지를 나타냅니다. 정렬을 먼저 수행하면 색인 작성의 성능이 향상됩니다(특히, 낮은 클러스터 비율 또는 클러스터 인수를 가진 색인의 경우). 색인이 정렬되어 작성되면 조회의 성능도 좋아집니다. 이 성능 향상의 대가는 정렬에 필요한 디스크 공간의 증가로서, 초기에 정렬을 수행하지 않고 색인을 작성할 때의 두 배의 공간이 필요합니다.

권장사항: 디스크 공간이 부족하지 않으면 기본 설정 (예)를 사용하십시오. 이 정렬에 필요한 디스크 공간은 색인 컬럼에 ORDER BY절이 있는 테이블에서 색인 컬럼 행을 SELECT하는 데 필요한 공간과 거의 같습니다.

대칭적 멀티프로세서(SMP) 환경이고 이 매개변수에 아니오를 지정하면, SMP 환경에서 가능한 다중 처리는 색인 작성 중에는 사용되지 않습니다.

순차적 검출 플래그(seqdetect)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	예 [예; 아니오]
관련 매개변수	452 페이지의 『기본 프리페치 크기 (dft_prefetch_sz)』

데이터베이스 관리 프로그램이 I/O를 모니터할 수 있으며, 순차적 페이지 읽기가 발생하는 경우 데이터베이스 관리 프로그램이 I/O 프리페치를 활성화할 수 있습니다. 이 유형의 순차적 프리페치는 *순차적 검출*로 알려져 있습니다. *seqdetect* 구성 매개변수를 사용하여 데이터베이스 관리 프로그램이 순차적 검출을 수행해야 하는지 여부를 제어할 수 있습니다.

이 매개변수가 아니오로 설정되면 데이터베이스 관리 프로그램이 유용함을 인식하는 경우에만 프리페치가 발생합니다(예: 테이블 정렬, 테이블 스캔 또는 목록 프리페치).

권장사항: 대부분의 경우, 이 매개변수의 기본값을 사용해야 합니다. 다른 조정으로는 심각한 조회 성능상의 문제점을 정정할 수 없을 때에만 순차적 검출을 중단시키십시오.

기본 프리페치 크기(dft_prefetch_sz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 32 [0 -- 32 767]

OS/2 및 Windows NT
16 [0 -- 32 767]

측정 단위 페이지

관련 매개변수

- 453 페이지의 『기본 테이블 공간 Extent 크기(dft_extent_sz)』
- 450 페이지의 『I/O 서버의 수(num_ioservers)』

테이블 공간이 작성될 때 PREFETCHSIZE n을 선택적으로 지정할 수 있습니다. (여기서 n은 프리페치가 수행될 때 데이터베이스 관리 프로그램이 읽을 페이지 수입니다.) CREATE TABLESPACE문에 프리페치 크기를 지정하지 않으면 데이터베이스 관리 프로그램은 이 매개변수가 제공하는 값을 사용합니다.

자세한 내용은 292 페이지의 『버퍼 풀로 데이터 프리페치』를 참조하십시오.

권장사항: 시스템 모니터 도구를 사용하여 시스템이 I/O를 기다리는 동안 CPU가 유휴 상태인지를 판별할 수 있습니다. 사용 중인 테이블 공간이 정의된 것과 동일한 프리페치 크기를 갖고 있지 않을 경우, 매개변수의 값을 늘리면 도움이 됩니다.

이 매개변수는 전체 데이터베이스에 기본값을 제공하는데, 데이터베이스 내의 모든 테이블 공간에는 적합하지 않을 수도 있습니다. 예를 들어, 값 32는 Extent 크기가 32페이지인 테이블 공간에는 적합하지만, Extent 크기가 25페이지인 테이블 공간에는 적합하지 않습니다. 각 테이블 공간에 대해 프리페치 크기를 명시적으로 설정하는 것이 이상적입니다.

기본 Extent 크기(*dft_extent_sz*)로 정의된 테이블 공간의 I/O를 최소화하려면 이 매개변수를 *dft_extent_sz* 매개변수 값의 한 인수나 완전한 배수로 설정해야 합니다. 예를 들어, *dft_extent_sz* 매개변수가 32이면 *dft_prefetch_sz*를 16(32의 인수) 또는 64(32의 배수)로 설정해야 합니다. 프리페치 크기가 Extent 크기의 배수이면 데이터베이스 관리 프로그램은 다음 조건이 참일 경우에 I/O를 병렬로 수행합니다.

- 프리페치되는 extent가 다른 실제 장치에 있음
- 다중 I/O 서버가 구성됨(*num_ioservers*)

SMS 컨테이너의 기본 수(numsegs)

구성 유형 데이터베이스

매개변수 유형 정보용

측정 단위 카운터

이 매개변수는 SMS 테이블 공간에만 적용되며, 기본 테이블 공간 내에서 작성될 컨테이너의 수를 나타냅니다. 이 매개변수는 사용자가 데이터베이스를 작성할 때 사용된 정보를 표시하여, 이것이 CREATE DATABASE 명령에서 명시적으로 지정되었는지 또는 내재적으로 지정되었는지 여부를 나타냅니다. CREATE TABLESPACE문은 어떠한 식으로도 이 매개변수를 사용하지 **않습니다**.

자세한 내용은 *관리 안내서: 계획*의 『데이터베이스 물리적 디렉토리』를 참조하십시오.

기본 테이블 공간 Extent 크기(dft_extent_sz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 32 [2 - 256]

측정 단위 페이지

관련 매개변수 452 페이지의 『기본 프리페치 크기 (dft_prefetch_sz)』

테이블 공간이 작성될 때 EXTENTSIZE n이 선택적으로 지정될 수 있습니다(여기서 n은 Extent *크기*). CREATE TABLESPACE문에서 Extent *크기*를 지정하지 않으면 데이터베이스 관리 프로그램이 이 매개변수 값을 사용합니다.

자세한 내용은 *관리 안내서: 계획*의 『테이블 공간 설계 및 선택』을 참조하십시오.

권장사항: 대부분의 경우, 테이블 공간을 작성할 때 Extent *크기*를 명시적으로 지정하려 합니다. 매개변수 값을 지정하기 전에 CREATE TABLESPACE문의 Extent *크기*를 명시적으로 선택하는 방법을 이해해야 합니다. 자세한 내용은 105 페이지의 『조회 최적화에 미치는 테이블 공간의 영향』에서 참조하십시오.

확장 저장 메모리 세그먼트 크기(estore_seg_sz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 16 000 [0 - 1 048 575]

측정 단위 페이지

관련 매개변수 『확장 저장영역 메모리 세그먼트의 수 (num_estore_segs)』

이 매개변수는 데이터베이스의 각 확장 영역 메모리 세그먼트의 페이지 수를 지정합니다. 이 매개변수는 머신이 최대 가상 주소 지정 가능 메모리보다 실제 주소 지정 가능 메모리가 더 클 경우에만 사용됩니다.

권장사항: 이 매개변수는 확장 저장영역을 사용할 수 있을 때만 효과가 있으며, *num_estore_segs* 매개변수에 의해 표시된 대로 사용됩니다. 또한 각 확장 메모리 세그먼트에 사용될 페이지 수를 지정할 때 *num_estore_segs* 매개변수를 검토하고 수정하여 확장 메모리 세그먼트의 수도 고려해야 합니다. 확장 저장에 대한 자세한 내용은 321 페이지의 『메모리 확장』을 참조하십시오.

확장 저장영역 메모리 세그먼트의 수(num_estore_segs)

구성 유형 데이터베이스

매개변수 유형	구성 가능
기본 [범위]	0 [0 - 214 7483 647]
관련 매개변수	454 페이지의『확장 저장 메모리 세그먼트 크기 (estore_seg_sz)』

이 매개변수는 데이터베이스가 사용할 수 있는 확장 저장영역 메모리 세그먼트의 수를 지정합니다.

기본값은 확장 저장영겨 메모리 세그먼트가 없는 상태를 나타냅니다.

권장사항: 플랫폼 환경에 최대 주소 공간보다 많은 메모리가 있으며 이 메모리를 사용할 의사가 있을 경우에 확장 저장영역 메모리 세그먼트를 설정할 때에만 이 매개변수를 사용하십시오. 세그먼트 수를 지정할 때 *estore_seg_sz* 매개변수를 검토하고 수정하여 각 세그먼트의 크기 또한 고려해야 합니다.

num_estore_segs 및 *estore_seg_sz* 구성 매개변수가 모두 설정될 경우, CREATE/ALTER BUFFERPOOL문을 통해 어떤 버퍼 풀이 확장 메모리를 사용하게 될 것인지 지정해 주어야 합니다. 확장 저장영역에 대한 자세한 내용은 321 페이지의『메모리 확장』에서 자세한 내용을 참조하십시오.

에이전트

다음 매개변수는 동시에 수행되어 최적의 성능을 달성할 수 있는 응용프로그램의 수에 영향을 미칩니다.

- 456 페이지의『사용 중인 응용프로그램의 최대수(maxappls)』
- 458 페이지의『사용 중인 응용프로그램의 평균 수(avg_appls)』
- 459 페이지의『응용프로그램당 열린 최대 데이터베이스 파일 수(maxfilop)』
- 460 페이지의『열린 총 파일 수(maxtotfilop)』
- 461 페이지의『에이전트의 우선순위(agentpri)』
- 462 페이지의『에이전트의 최대수(maxagents)』
- 464 페이지의『동시 에이전트의 최대수(maxcagents)』
- 465 페이지의『조정 에이전트의 최대 수(max_coordagents)』
- 466 페이지의『논리 에이전트의 최대 수(max_logicagents)』

- 467 페이지의 『에이전트 풀 크기(num_poolagents)』
- 469 페이지의 『풀의 초기 에이전트의 수(num_initagents)』

사용 중인 응용프로그램의 최대수(maxappls)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 40 [1 - 60 000]

지역 및 원격 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
20 [1 - 60 000]

지역 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
10 [1 - 60 000]

측정 단위 카운터

관련 매개변수

- 462 페이지의 『에이전트의 최대수(maxagents)』
- 465 페이지의 『조정 에이전트의 최대 수(max_coordagents)』
- 443 페이지의 『레벨 자동 업그레이드 전의 최대 잠금 목록 비율(maxlocks)』
- 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
- 458 페이지의 『사용 중인 응용프로그램의 평균 수(avg_appls)』

이 매개변수는 동시에 데이터베이스에 연결될 수 있는(지역 및 원격 모두) 응용프로그램의 최대수를 지정합니다. 데이터베이스에 연결되는 각 응용프로그램으로 인해 일부 개인용 메모리가 할당되므로, 동시 응용프로그램의 수를 크게 하면 더 큰 메모리를 사용할 수도 있습니다.

이 매개변수의 값은 연결된 응용프로그램의 합에 2단계 확약 또는 구간 복원 프로세스에서 동시에 수행될 수 있는 동일한 응용프로그램의 수를 합한 값보다 *크거나 같아야* 합니다. 그런 다음 이 합에 한 번에 존재할 수 있는 2단계 확약 중 이상 실패 트랜잭션의 예상 수를 더하십시오. 2단계 확약 중 이상 실패 트랜잭션에 대한 자세한 내용은 *관리 안내서: 계획*의 『2단계 확약 중의 문제점 복구』를 참조하십시오.

응용프로그램이 데이터베이스에 연결하려고 하지만, *maxappls*에 이미 도달되어 있어서 최대 응용프로그램 수만큼 데이터베이스에 연결되어 있음을 가리키는 오류가 응용프로그램으로 리턴됩니다.

더 많은 응용프로그램이 DataLinks 관리 프로그램을 사용하므로, *maxappls* 값을 증가해야 합니다. 다음 공식을 사용하여 필요한 값을 계산하십시오.

```
<maxappls> = 5 * (number of nodes) + (peak number of active applications
                 using DataLinks Manager)
```

DataLinks 관리 프로그램에 지원되는 최대값은 2 000입니다.

파티션된 데이터베이스 환경에서 이 값은 한 데이터베이스 파티션에 대해 동시에 사용할 수 있는 최대 응용프로그램의 수를 나타냅니다. 해당 서버가 응용프로그램에 대한 조정자(coordinator) 노드인지 여부에 상관없이 이 매개변수는 데이터베이스 파티션 서버의 데이터베이스 파티션에 대해 사용 중인 응용프로그램 수를 제한합니다. 파티션된 데이터베이스 환경에서 카탈로그 노드의 *maxappls* 값은 다른 환경의 경우에 비해 높아야 하는데, 이는 파티션된 데이터베이스 환경에서는 모든 응용프로그램이 카탈로그 노드로 연결되어야 하기 때문입니다.

권장사항: *maxlocks* 매개변수를 줄이거나 *locklist* 매개변수를 늘리지 않은 채 이 매개변수의 값을 늘리면 응용프로그램의 한계가 아닌 잠금상의 데이터베이스 한계(*locklist*)에 도달하여 과도한 잠금 레벨 자동 업그레이드 문제점이 발생할 수 있습니다.

어느 정도까지는 최대 응용프로그램의 수 역시 *maxagents*로 관리됩니다. 사용 가능한 연결(*maxappls*)이 하나이고 사용 가능한 에이전트(*maxagents*)도 하나인 경우, 하나의 응용프로그램만을 데이터베이스에 연결할 수 있습니다. 그리고 응용프

로그램의 최대수도 *max_coordagents* 구성 매개변수에 의해 제어되는데, 이는 *max_coordagents* 값에 도달할 경우 새 응용프로그램(즉, 조정자(coordinator) 에이전트)을 시작할 수 없기 때문입니다.

사용 중인 응용프로그램의 평균 수(avg_appls)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 1 [1 - maxappls]

측정 단위 카운터

관련 매개변수

- 456 페이지의 『사용 중인 응용프로그램의 최대 수(maxappls)』

이 매개변수는 SQL 최적화 알고리즘에 사용되어 선택된 액세스 플랜의 런타임 시 사용할 수 있는 버퍼 풀의 *크기*를 계산하는 데 도움이 됩니다.

권장사항: 다중 사용자 환경 특히 복합 조회 및 대형 버퍼 풀에서 DB2를 수행할 경우, 다중 조회 사용자가 시스템을 사용하는 중이므로 최적화 알고리즘이 버퍼 풀의 사용 가능성을 가정할 때 보다 신중하게 함을 SQL 최적화 알고리즘이 알도록 해야 합니다.

이 매개변수를 설정할 때 일반적인 데이터베이스 사용에 있어서 복합 조회 응용프로그램의 수를 추정해야 합니다. 이는 모든 단순 OLTP 응용프로그램을 제외한 수여야 합니다. 이 수를 계산하는 데 문제가 있으면, 다음을 곱하십시오.

- 데이터베이스에 대해 수행 중인 모든 응용프로그램의 평균 수. 데이터베이스 시스템 모니터가 주어진 시간에 응용프로그램 수에 관한 정보를 제공할 수 있으므로, 샘플링 기법을 사용하여 일정 기간 동안의 평균을 계산할 수 있습니다. 데이터베이스 시스템 모니터의 정보에는 OLTP 및 OLTP 이외의 응용프로그램이 모두 포함됩니다.
- 복합 조회 응용프로그램 비율의 추정치

최적화 알고리즘에 영향을 주는 다른 구성 매개변수를 조정함에 따라 이 매개변수도 약간 증가시켜야 합니다. 이로써 경로 선택의 차이를 최소화할 수 있습니다.

이 매개변수를 변경한 후에는 (REBIND PACKAGE 명령을 사용하여) 응용프로그램을 리바인드해야 합니다.

응용프로그램당 열린 최대 데이터베이스 파일 수(maxfilop)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX 64 [2 - 1950]

OS/2 및 Windows NT
64 [2 - 32 768]

측정 단위 카운터

관련 매개변수

- 460 페이지의『열린 총 파일 수(maxtotfilop)』
- 456 페이지의『사용 중인 응용프로그램의 최대 수(maxappls)』

이 매개변수는 각 데이터베이스 에이전트에 대해 열 수 있는 파일 핸들의 최대수를 지정합니다. 파일을 열어 이 값이 초과되는 경우, 이 에이전트가 사용 중인 일부 파일이 닫힙니다. *maxfilop*이 너무 작으면 이 한계를 넘지 않기 위해 여는 파일과 닫는 파일의 오버헤드가 과도하게 되어 성능을 저하시킬 수 있습니다.

SMS 테이블 공간 및 DMS 테이블 공간 파일 컨테이너가 모두 운영 체제와 상호작용하는 데이터베이스 관리 프로그램 내의 파일로 처리되므로, 파일 핸들이 필수입니다. SMS 테이블 공간이 사용하는 파일은 DMS 파일 테이블 공간에 사용되는 컨테이너의 수보다 일반적으로 많습니다. 그러므로 SMS 테이블 공간을 사용하는 경우, 이 매개변수의 값이 DMS 파일 테이블 공간에 필요한 값보다 더 커야 합니다.

이 매개변수를 사용하면 에이전트당 핸들의 수를 특정 수로 제한하여 데이터베이스 관리 프로그램이 사용하는 파일 핸들의 전반적인 총 수가 운영 체제 한계를 넘지 않도록 할 수 있습니다. 실제 수는 동시에 수행되는 에이전트의 수에 따라 다릅니다.

열린 총 파일 수(maxtotfilop)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	16 000 [100 - 32 768]
측정 단위	카운터
관련 매개변수	459 페이지의 『응용프로그램당 열린 최대 데이터베이스 파일 수(maxfilop)』

이 매개변수는 단일 데이터베이스 관리 프로그램 인스턴스에서 실행 중인 모든 에이전트와 기타 스레드가 열 수 있는 파일의 최대수를 정의합니다. 파일을 열어 이 값이 초과되는 경우, 오류가 응용프로그램으로 리턴됩니다.

주: 이 매개변수는 UNIX 기반 플랫폼에는 적용되지 않습니다.

권장사항: 이 매개변수를 설정할 때에는 데이터베이스 관리 프로그램 인스턴스의 각 데이터베이스에 사용할 수 있는 파일 핸들의 수를 고려해야 합니다. 이 매개변수의 상한 값을 추정하려면 다음을 수행하십시오.

1. 한 인스턴스에서 각 데이터베이스에 대해 열 수 있는 파일 핸들의 최대수를 계산하려면 다음 공식을 사용하십시오.

   ```
   maxappls * maxfilop
   ```

2. 위 결과의 합계를 계산하여 매개변수의 최대값을 넘지 않는지 확인하십시오.

새 데이터베이스가 작성되면 이 매개변수의 값을 다시 산출해야 합니다.

에이전트의 우선순위(agentpri)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

AIX -1 [41 - 125]

기타 UNIX
-1 [41 - 128]

Windows NT
-1 [0 - 6]

OS/2 -1 [200 - 231; 300 - 331; 400 - 431]

이 매개변수는 모든 에이전트와, 운영 체제 스케줄러에 의해 다른 데이터베이스 관리 프로그램 인스턴스 프로세스 및 스레드에 주어지는 우선순위를 제어합니다. 파티션된 데이터베이스 환경에서는 여기에 조정 및 서브에이전트, 병렬 시스템 제어기 및 FCM 디먼이 포함됩니다. 이 우선순위는 머신에서 수행 중인 다른 프로세스나 스레드에 비추어 DB2 프로세스, 에이전트 및 스레드에 CPU 시간이 어떻게 주어지는지 결정합니다. 매개변수가 -1로 설정될 때 특별한 조치가 취해지지 않고, 운영 체제가 프로세스와 스레드를 스케줄링하는 정상적인 방법으로 데이터베이스 관리 프로그램이 스케줄됩니다. 매개변수가 -1 이외의 값으로 설정되면 데이터베이스 관리 프로그램은 매개변수의 값으로 설정된 정적 우선순위를 갖는 프로세스와 스레드를 작성합니다. 따라서 이 매개변수로 사용자는 데이터베이스 관리 프로그램 프로세스와 스레드가 머신에서 실행될 우선순위를 제어할 수 있습니다.

이 매개변수를 사용하여 데이터베이스 관리 프로그램 처리량을 증가시킬 수 있습니다. 이 매개변수 설정값은 데이터베이스 관리 프로그램이 수행되고 있는 운영 체제에 따라 결정됩니다. 예를 들어, UNIX 기반 환경에서는 작은 숫자 값이 우선순위가 높습니다. 매개변수가 41에서 125 사이 값으로 설정되면 데이터베이스 관리 프로그램은 매개변수 값에 설정된 UNIX 정적 우선순위를 가지고 에이전트를 작성합니다. 이것은 UNIX 기반 환경에서 중요한데, 낮은 숫자 값의 데이터베이스 관리 프로그램 우선순위가 높은 반면 다른 프로세스(응용프로그램 및 사용자 포함)가 충분한 CPU 시간을 확보하지 못하여 지연이 발생할 수 있기 때문입니다. 이 매개변수의 설정과 머신에서 예상되는 기타 활동과의 균형을 맞추어야 합니다.

OS/2 환경에서는 큰 숫자 값이 더 높은 우선순위를 나타냅니다.

권장사항: 초기에는 기본값을 사용해야 합니다. 이 값은 다른 사용자/응용프로그램의 응답 시간과 데이터베이스 관리 프로그램 처리량 사이를 절충합니다.

데이터베이스 성능이 중요한 경우, 이 매개변수의 최적 설정을 결정하기 위해 벤치마킹 기법을 사용할 수 있습니다. 특히 CPU 사용도가 매우 높을 때에는 다른 사용자 프로세스의 성능이 극도로 저하될 수 있으므로, 데이터베이스 관리 프로그램의 우선순위를 증가시킬 때에는 주의해야 합니다. 데이터베이스 관리 프로그램 프로세스와 스레드의 우선순위를 증가시키면 상당한 성능상의 이점을 얻을 수 있습니다.

주: 이 매개변수를 UNIX 기반 플랫폼에서 기본값 이외의 다른 값으로 설정할 경우, 조정자(governor)를 사용하여 에이전트 우선순위를 변경할 수는 없습니다.

에이전트의 최대수(maxagents)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 200 [1 - 64 000]

지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버에서 400 [1 - 64 000]

지역 클라이언트가 있는 위성 데이터베이스 서버에서 10 [1 - 64 000]

측정 단위 카운터

관련 매개변수

- 456 페이지의 『사용 중인 응용프로그램의 최대수(maxappls)』
- 464 페이지의 『동시 에이전트의 최대수(maxcagents)』
- 465 페이지의 『조정 에이전트의 최대 수(max_coordagents)』
- 471 페이지의 『DARI 프로세스의 최대수(maxdari)』
- 425 페이지의 『최소의 확약된 개인용 메모리(min_priv_mem)』
- 467 페이지의 『에이전트 풀 크기(num_poolagents)』

이 매개변수는 주어진 시간에 응용프로그램 요청을 승인하기 위해 사용할 수 있는 데이터베이스 관리 프로그램 에이전트(조정자 에이전트 또는 서브에이전트) 수를 나타냅니다. 조정 에이전트 수를 제한하려면 *max_coordagents* 매개변수를 사용하십시오.

메모리가 한정된 환경에서 각 추가 에이전트에 추가 메모리가 필요하기 때문에 데이터베이스 관리 프로그램의 총 메모리 사용을 제한해야 하는 경우, 이 매개변수를 사용하는 것이 좋습니다.

권장사항: *maxappls*의 값은 적어도 동시 액세스가 허용되는 각 데이터베이스의 *maxappls* 값의 합이 되어야 합니다. 데이터베이스의 수가 *numdb* 매개변수보다 크면 *maxappls*의 가장 큰 값과 *numdb*를 곱한 값을 사용하는 것이 가장 안전한 방법입니다.

각 추가 에이전트에는 데이터베이스 관리 프로그램이 시작될 때 할당된 일부 자원의 오버헤드가 필요합니다.

동시 에이전트의 최대수(maxcagents)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 (*max_coordagents*) [-1; 1 - *max_coordagents*]

측정 단위 카운터

관련 매개변수

- 456 페이지의 『사용 중인 응용프로그램의 최대수(maxappls)』
- 462 페이지의 『에이전트의 최대수(maxagents)』
- 465 페이지의 『조정 에이전트의 최대 수(max_coordagents)』

데이터베이스 관리 프로그램 트랜잭션을 동시에 실행할 수 있는 데이터베이스 관리 프로그램 에이전트의 최대수. 이 매개변수는 응용프로그램의 동시 활동을 높일 동안 시스템의 로드를 제어하는 데 사용합니다. 예를 들어, 시스템이 대량의 연결

을 필요로 하지만 이 연결을 처리할 메모리가 제한되어 있는 경우가 있을 수 있습니다. 동시 활동이 많은 기간이 있어 운영 체제 페이징 과다를 유발시킬 수 있는 환경에서 이 매개변수를 조정하면 유용합니다.

이 매개변수는 데이터베이스에 대한 연결이 되는 응용프로그램의 수를 제한하지 않습니다. 단지 데이터베이스 관리 프로그램이 한 번에 동시 처리할 수 있는 데이터베이스 관리 프로그램 에이전트의 수를 제한하여 처리량이 많을 때의 시스템 자원 사용을 제한합니다.

값 -1은 한계치가 *max_coordagents*임을 나타냅니다.

권장사항: 대부분의 경우, 이 매개변수의 기본값을 사용하는 것이 좋습니다. 높은 응용프로그램의 동시성이 문제를 일으킬 경우에는, 벤치마크 테스팅으로 매개변수를 조정하여 데이터베이스의 성능을 최적화할 수 있습니다.

조정 에이전트의 최대 수(max_coordagents)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 (*maxagents - num_initagents*)

[-1, 0 - *maxagents*]

파티션된 데이터베이스 환경 및 *intra_parallel*이 예로 설정된 환경에서 기본값은 *maxagents - num_initagents*입니다. 그렇지 않으면 *maxagents* 입니다. 파티션되지 않은 데이터베이스 환경에서는

시스템이 파티션 내 병렬 처리용으로 구성되지 않으면 *max_coordagents*가 항상 *maxagents*와 같습니다.

파티션된 데이터베이스 환경에 있지 않고 *intra_parallel* 매개변수가 작동되지 않는 경우에는 *max_coordagents*가 *maxagents*와 같아야 합니다.

관련 매개변수

- 469 페이지의『풀의 초기 에이전트의 수(num_initagents)』
- 467 페이지의『에이전트 풀 크기(num_poolagents)』
- 462 페이지의『에이전트의 최대수(maxagents)』
- 544 페이지의『파티션 내 병렬 처리 작동(intra_parallel)』

이 매개변수는 파티션되거나 파티션되지 않은 데이터베이스 환경에서 한 서버에 동시에 존재할 수 있는 조정 에이전트의 최대수를 지정합니다.

데이터베이스에 연결되거나 인스턴스에 접속되는 각 지역 또는 원격 응용프로그램에 대해 하나의 조정 에이전트가 취득됩니다. 인스턴스 접속이 필요한 요청으로는 CREATE DATABASE, DROP DATABASE 및 데이터베이스 시스템 모니터 명령이 있습니다.

논리 에이전트의 최대 수(max_logicagents)

구성 유형	데이터베이스 관리 프로그램
매개변수 유형	구성 가능
기본 [범위]	-1 (*max_coordagents*) [-1; *max_coordagents* -- 64 000]

이 매개변수는 인스턴스에 연결될 수 있는 응용프로그램 최대 수를 제어합니다. 일반적으로, 각 응용프로그램에는 조정자 에이전트가 지정됩니다. 에이전트는 응용프로그램과 데이터베이스 사이에 조작을 시행합니다. 이 매개변수의 기본값이 사용

될 경우, 조정자 기능은 활성화되지 않습니다. 결과적으로, 각 에이전트는 자체의 고유한 개인용 메모리를 사용하여 작동하며 다른 에이전트와 함께 버퍼 풀과 같은 데이터베이스 관리 프로그램 및 데이터베이스 전역 자원을 공유합니다. 매개변수가 기본값보다 큰 값으로 설정될 경우, 집중기(concentrator) 기능이 활성화됩니다. 집중기(concentrator)는 DB2 Connect 게이트웨이가 10,000개 이상의 클라이언트 연결을 처리할 수 있는 지점까지 클라이언트 응용프로그램당 서버 자원 수를 줄이기 위한 것입니다.

DB2 Connect를 XA 트랜잭션 지원 집중기로 사용하는 방법과 예에 대한 자세한 내용은 *DB2 Connect 사용자 안내서*를 참조하십시오.

값 -1은 한계가 *max_coordagents*임을 나타냅니다.

에이전트 풀 크기(num_poolagents)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 [-1, 0 -- *maxagents*]

기본값을 사용하면 파티션되지 않은 데이터베이스와 지역 클라이언트가 있는 서버 값은 *maxagents*/50 또는 *max_querydegree* 중 큰 값입니다.

기본값을 사용하면 파티션되지 않은 데이터베이스와 지역 및 원격 클라이언트가 있는 서버 값이

maxagents/50 x *max_querydegree* 또는 *maxagents* - *max_coordagents* 중 큰 값입니다.

기본값을 사용하면 데이터베이스 파티션 서버의 값은 *maxagents*/10 x *max_querydegree* 또는 *maxagents* - *max_coordagents*입니다.

관련 매개변수

- 469 페이지의 『풀의 초기 에이전트의 수 (num_initagents)』
- 462 페이지의 『에이전트의 최대수(maxagents)』
- 543 페이지의 『병렬의 처리 최대 조회 수준 (max_querydegree)』
- 465 페이지의 『조정 에이전트의 최대 수 (max_coordagents)』

이 매개변수는 에이전트 풀의 *크기*를 지정하는 지침이 됩니다. (또한 DB2 버전 2에 사용된 *max_idleagents* 매개변수를 바꿉니다.)

에이전트 풀에는 서브에이전트와 유휴 에이전트가 들어 있습니다. 유휴 에이전트를 병렬 서브에이전트 또는 조정자 에이전트로 사용할 수 있습니다. 이 매개변수 값으로 지정한 수보다 많은 에이전트가 작성될 경우, 에이전트는 풀로 리턴되지 않고 현재의 요청 실행을 완료하면서 종료됩니다.

이 매개변수의 값이 0이면 에이전트는 필요한 만큼 작성되고, 현재의 요청 실행을 완료하면서 종료됩니다. 매개변수 값이 *maxagents*일 경우, 풀은 연관된 서브에이전트로 가득 차게 되며 서버는 조정자 노드로 사용될 수 없는데, 이는 새 조정 에이전트를 작성할 수 없기 때문입니다.

권장사항: 동시에 연결되는 응용프로그램이 거의 없는 결정 지원 환경을 실행하는 경우에 에이전트 풀이 유휴 에이전트로 가득 차는 일이 없도록 *num_poolagents*를 작은 값으로 설정하십시오.

많은 응용프로그램이 동시에 연결되는 트랜잭션 처리 환경을 수행하는 경우, 자주 일어나는 에이전트의 작성 및 종료와 연관된 비용을 피하기 위해 *num_poolagents* 값을 증가시키십시오.

풀의 초기 에이전트의 수(num_initagents)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 0 [0 -- *num_poolagents*]

관련 매개변수

- 462 페이지의 『에이전트의 최대수(maxagents)』
- 467 페이지의 『에이전트 풀 크기(num_poolagents)』
- 465 페이지의 『조정 에이전트의 최대 수(max_coordagents)』

이 매개변수는 DB2START시 에이전트 풀에서 작성되는 대기 에이전트의 초기 수를 결정합니다.

저장 프로시듀어(DARI)

다음 매개변수는 데이터베이스 응용프로그램 원격 인터페이스(DARI) 응용프로그램에 영향을 줄 수 있습니다.

- 470 페이지의 『DARI 프로세스 유지 표시기(keepdari)』
- 471 페이지의 『DARI 프로세스의 최대수(maxdari)』
- 472 페이지의 『JVM을 사용하여 DARI 프로세스 초기화(initdari_jvm)』
- 473 페이지의 『풀 내의 초기 분리(fenced) DARI 프로세스의 수(num_initdaris)』

주: DARI는 저장 프로시듀어를 말합니다.

DARI 프로세스 유지 표시기(keepdari)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	예 [예; 아니오]
관련 매개변수	471 페이지의 『DARI 프로세스의 최대수 (maxdari)』

이 매개변수는 DARI 호출이 완료된 후에 DARI 프로세스의 유지 여부를 나타냅니다. DARI 프로세스는 데이터베이스 관리 프로그램 에이전트 프로세스로부터 사용자가 작성한 DARI 코드를 분리하기 위해 별도의 시스템 항목으로 작성됩니다. 이 매개변수는 데이터베이스 서버에만 적용됩니다.

*keepdari*가 *아니오*로 설정되면 새 DARI 프로세스가 각 DARI 호출에 대해 작성되고 제거됩니다. *keepdari*가 *예*로 설정되면 한 DARI 프로세스가 후속 DARI 호출에 대해 재사용됩니다. 데이터베이스 관리 프로그램이 중단되면 모든 미해결 DARI 프로세스가 종료합니다.

이 매개변수를 *예*로 설정하면 활성화된 각 DARI 프로세스에 대한 데이터베이스 관리 프로그램이 시스템 자원을 *maxdari* 매개변수에 들어 있는 값까지 추가로 사용합니다. 이는 후속 DARI 호출을 처리하는 데 사용할 기존의 DARI 프로세스가 없는 경우에만 적용됩니다. 이 매개변수는 *maxdari*가 0으로 설정된 경우, 무시됩니다.

권장사항: 비 DARI 요청 수에 비해 DARI 요청 수가 많으며 시스템 자원이 한정되어 있지 않은 환경에서는, 이 매개변수를 *예*로 설정할 수 있습니다. 기존의

DARI 프로세스를 호출 처리에 사용할 수 있기 때문에 초기의 DARI 프로세스 작성 오버헤드를 피할 수 있어 DARI 성능을 향상시킬 수 있습니다.

예를 들어, OLTP 대-차변 은행 거래 응용프로그램에서 각 트랜잭션을 수행하는 코드가 DARI 프로세스에서 실행하는 저장 프로시듀어로 수행될 수 있습니다. 이 응용프로그램에서 주 워크로드는 DARI 프로세스 외부에서 수행됩니다. 이 매개변수가 *아니오*로 설정되면 각 트랜잭션이 새 DARI 프로세스 작성 오버헤드가 발생하여 심각한 성능 저하를 초래할 수 있습니다. 그러나 이 매개변수를 *예*로 설정하면 각 트랜잭션이 기존의 DARI 프로세스를 사용하려 하므로, 이러한 오버헤드를 피할 수 있습니다.

DARI 프로세스의 최대수(maxdari)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 (*max_coordagents*) [-1; 0 - *max_coordagents*]

측정 단위 카운터

관련 매개변수

- 462 페이지의 『에이전트의 최대수(maxagents)』
- 470 페이지의 『DARI 프로세스 유지 표시기(keepdari)』
- 473 페이지의 『풀 내의 초기 분리(fenced) DARI 프로세스의 수(num_initdaris)』

• 465 페이지의『조정 에이전트의 최대 수 (max_coordagents)』

이 매개변수는 데이터베이스 서버에 상주하는 DARI 프로세스의 최대수를 나타냅니다. 일단 이 한계에 도달하면 새 DARI 요청이 호출될 수 없습니다. 이 매개변수는 데이터베이스 서버에만 적용됩니다.

조정 에이전트당 DARI 프로세스를 둘 이상 사용할 수 없기 때문에 DARI 프로세스의 최대 수 또한 조정 에이전트의 최대 수(*max_coordagents*)에 따라 달라집니다.

권장사항: 사용자의 환경이 데이터베이스 관리 프로그램 내에서 DARI 기능을 사용하는 경우, 데이터베이스 관리 프로그램 내에서 만들어진 DARI 호출을 처리하는 데 적절한 수의 DARI 프로세스가 사용 가능하도록 이 매개변수를 사용할 수 있습니다.

매개변수가 -1로 설정되는 경우에 DARI 프로세스의 최대 수는 *max_coordagents* 매개변수에 설정된 값과 같아집니다.

데이터베이스 관리 프로그램의 성능에 영향을 미치는 DARI 프로세스에 제공되는 시스템 자원의 양이 적절하지 못하여 기본값이 사용자 환경에 부적합하다는 것을 알게 되면 다음 값은 이 매개변수를 조정하기 위한 시작점을 제공하는 데 유용할 수 있습니다.

```
maxdari = 한 번에 DARI를 호출할 수 있는 응용프로그램의 수
```

*keepdari*가 *예*로 설정되면 작성된 각 DARI 프로세스는 계속 존재하면서 DARI 호출이 처리되어 에이전트로 리턴된 후에도 시스템 자원을 사용합니다.

사용자의 환경이 극히 한정적이어서 DARI와 관련된 프로세스 자원에 여유가 없는 경우, 이 매개변수를 0으로 설정하여 DARI를 사용 불가능하게 할 수 있습니다.

JVM을 사용하여 DARI 프로세스 초기화(initdari_jvm)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 아니오 [예; 아니오]

관련 매개변수

- 471 페이지의 『DARI 프로세스의 최대수 (maxdari)』
- 『풀 내의 초기 분리(fenced) DARI 프로세스의 수(num_initdaris)』
- 470 페이지의 『DARI 프로세스 유지 표시기 (keepdari)』

이 매개변수는 각 분리(fenced) DARI 프로세스가 시작시 JVM을 로드할지 여부를 나타냅니다. 이 매개변수를 설정하면 특히, *num_initdaris* 매개변수와 함께 사용될 경우 분리된 Java 저장 프로시듀어에 대한 초기 시작 시간이 짧아집니다. 이 매개변수를 사용하면 JVM가 필요하지 않는 Java가 아닌 분리(fenced) 저장 프로시듀어에 대한 초기 로드 시간이 길어질 수 있습니다.

풀 내의 초기 분리(fenced) DARI 프로세스의 수(num_initdaris)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 0 [0 -- *maxdari*]

관련 매개변수

- 471 페이지의 『DARI 프로세스의 최대수 (maxdari)』
- 472 페이지의 『JVM을 사용하여 DARI 프로세스 초기화(initdari_jvm)』
- 470 페이지의 『DARI 프로세스 유지 표시기 (keepdari)』

이 매개변수는 DB2STAR시 DARI 풀에서 작성되는 초기 유휴 분리(fenced) DARI 프로세스 수를 나타냅니다. 이 매개변수를 설정하면 분리(fenced) 저장 프로시듀어의 초기 시작 시간이 줄어듭니다. 이 매개변수는 *keepdari*가 지정되지 않을 경우에는 무시됩니다.

로그 및 복구

사용자 환경을 복구하는 작업은 중요한 데이터 유실을 방지하는 데 있어서 매우 중요합니다. 사용자 환경을 적절하게 관리하고 데이터 및 트랜잭션을 올바르게 복구하는 데 도움이 되는 다수의 매개변수가 사용 가능합니다. 이 매개변수는 다음 범주로 그룹화됩니다.

- 『데이터베이스 로그 파일』
- 482 페이지의 『데이터베이스 로그 작업』
- 488 페이지의 『복구』
- 496 페이지의 『분산 작업 단위(DUOW) 복구』

데이터베이스 로그 파일

다음 매개변수는 데이터베이스 로그에 사용되는 파일의 수, 크기 및 상태에 대한 정보를 제공합니다.

- 475 페이지의 『로그 파일의 크기(logfilsiz)』

- 477 페이지의『1차 로그 파일의 수(logprimary)』
- 479 페이지의『2차 로그 파일의 수(logsecond)』
- 480 페이지의『데이터베이스 로그 경로 변경(newlogpath)』
- 481 페이지의『로그 파일의 위치(logpath)』
- 482 페이지의『사용 중인 첫 번째 로그 파일(loghead)』

로그 파일의 크기(logfilsiz)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위]

UNIX	1000 [4 -- 65 535]
Windows NT	250 [4 -- 65 535]
OS/2	250 [4 -- 65 535]

측정 단위 페이지(4KB)

관련 매개변수

- 477 페이지의『1차 로그 파일의 수(logprimary)』
- 479 페이지의『2차 로그 파일의 수(logsecond)』
- 484 페이지의『복구 범위 및 소프트 점검점 간격(softmax)』

이 매개변수는 1차 및 2차 로그 파일 각각의 *크기*를 정의합니다. 이들 로그 파일의 *크기*는 로그 파일이 가득 차서 새 로그 파일이 필요하게 되기 전에 기록할 수 있는 로그 레코드의 수를 제한합니다.

1차 및 2차 로그 파일의 사용뿐만 아니라, 로그 파일이 가득 찰 때 취해지는 조치는 수행되는 로그 유형에 따라 달라집니다.

- 순환 로그

  1차 로그 파일은 그 안에 기록된 변경사항이 확약되고 나면 재사용할 수 있습니다. 로그 파일 *크기*가 작은데 응용프로그램이 변경사항의 확약 없이 많은 수의 데이터베이스에 대한 변경사항을 처리하면 1차 로그 파일이 빨리 차게 됩니

다. 모든 1차 로그 파일이 가득 차게 되면 데이터베이스 관리 프로그램이 새 로그 레코드를 보유하기 위해 2차 로그 파일을 할당합니다.

- 로그 유지 로그

  1차 로그 파일이 가득 차면 로그는 아카이브되고 새 1차 로그 파일이 할당됩니다.

권장사항: 로그 파일의 *크기*와 1차 로그 파일의 수가 균형을 이루도록 합니다.

- 데이터베이스가 로그 파일을 매우 빨리 가득 차게 하는 로그 파일 값에 대해 수행하는 많은 양의 갱신, 삭제 및/또는 삽입 트랜잭션을 갖는 경우, *logfilsiz* 값을 증가시켜야 합니다.

  주: 총 로그 파일 *크기* 한계는 32GB입니다. 즉, 로그 파일의 수(*logprimary* + *logsecond*)에 각 로그 파일의 *크기*(바이트)를 곱한 값(*logfilsiz* * 4096)은 32GB를 초과해서는 안됩니다.

  로그 파일이 너무 작으면 오래된 로그 파일을 아카이브하고, 새 로그 파일을 할당하며, 사용할 수 있는 로그 파일을 기다리는 등의 작업 시간 때문에 시스템 성능에 영향을 줄 수 있습니다.
- 1차 로그가 이 *크기*로 사전에 할당되어 있으므로, 디스크 공간이 부족한 경우 *logfilsiz* 값을 줄여야 합니다.

  일부 미디어는 전체 로그 파일을 보유하는 데 사용할 수 없으므로, 로그 파일이 너무 크면 아카이브된 로그 파일과 로그 파일의 사본을 관리할 때 융통성을 감소시킬 수 있습니다.

로그 유지 기능을 사용할 경우, 마지막 응용프로그램이 데이터베이스로부터 연결 해제되면 현재 사용 중인 로그 파일은 닫히고 절단됩니다. 데이터베이스에 다음 연결이 있을 때 새 로그 파일이 사용됩니다. 그러므로 동시에 수행되는 응용프로그램의 로그 요구사항을 알고 있다면 낭비되는 많은 양의 공간을 할당하지 않도록 로그 파일 *크기*를 적절하게 결정할 수 있습니다.

이 매개변수에 대한 자세한 내용은 *관리 안내서: 구현*의 『데이터베이스 로그를 위한 구성 매개변수』를 참조하십시오.

1차 로그 파일의 수(logprimary)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	3 [2 - 128]
측정 단위	카운터
할당 시기	• 데이터베이스가 작성될 때 • 로그가 다른 위치로 이동될 때(*logpath* 매개변수가 갱신될 때) • 모든 사용자가 연결 해제된 후 다음 데이터베이스의 연결 중 이 매개변수(*logprimary*) 값의 증가 후에 • 로그 파일이 아카이브되고 새 파일이 할당될 때(*logretain* 또는 *userexit* 매개변수가 작동되어야 함) • *logfilsiz* 매개변수가 변경되면 사용 중인 로그 파일은 모든 사용자가 연결 해제된 후 다음 데이터베이스의 연결 중에 크기가 조정됩니다.
해제 시기	이 매개변수가 감소해야 해제됩니다. 감소되면 불필요한 로그 파일이 데이터베이스로의 다음 연결 중에 삭제됩니다.
관련 매개변수	• 475 페이지의 『로그 파일의 크기(logfilsiz)』 • 479 페이지의 『2차 로그 파일의 수(logsecond)』 • 486 페이지의 『로그 유지 작동 가능(logretain)』 • 487 페이지의 『User Exit 사용 가능(userexit)』

1차 로그 파일은 복구 로그 파일에 할당된 저장영역의 고정량을 설정합니다. 이 매개변수로 사전에 할당될 1차 로그 파일의 수를 지정할 수 있습니다.

순환 로그에서는 1차 로그가 순서대로 반복적으로 사용됩니다. 즉, 로그가 가득 차면 순서상 다음 1차 로그가 (사용 가능한 경우) 사용됩니다. 한 로그 내에서 로그 레코드에 대한 모든 작업 단위(UOW)가 확약 또는 구간 복원되면 로그는 사용 가능한 것으로 간주됩니다. 순서상 다음 1차 로그를 사용할 수 없는 경우, 2차 로그가 할당되어 사용됩니다. 순서상 다음 1차 로그를 사용할 수 있게 되거나 *logsecond* 매개변수에 의한 한계값에 도달할 때까지 추가 2차 로그가 할당되어 사용됩니다. 이들 2차 로그 파일은 데이터베이스 관리 프로그램에서 더 이상 필요하지 않게 되면 동적으로 할당 해제됩니다.

1차 및 2차 로그 파일의 수는 다음 공식을 따라야 합니다.

- (*logprimary* + *logsecond*) <= 128

권장사항: 이 매개변수는 사용 중인 로그의 유형, 로그 파일의 크기, 처리되고 있는 환경 유형(예: 트랜잭션의 길이와 확약의 빈도)과 함께 인수의 수에 따라 결정됩니다.

이 값을 늘리면 1차 로그 파일이 최초의 데이터베이스 연결 중에 사전 할당되므로 로그의 디스크 요구사항이 증가합니다.

2차 로그 파일이 자주 할당됨을 알게 되면 로그 파일 크기(*logfilsiz*)를 늘리거나 1차 로그 파일의 수를 늘려서 시스템 성능을 향상시킬 수도 있습니다.

자주 액세스하지 않는 데이터베이스의 경우, 디스크 저장영역을 절약할 수 있도록 매개변수를 2로 설정하십시오. 롤 포워드 복구에 사용되는 데이터베이스의 경우, 즉시 새 로그를 할당하는 오버헤드를 피하려면 매개변수를 더 큰 값으로 설정하십시오.

데이터베이스 시스템 모니터를 사용하면 1차 로그 파일의 크기를 조정하는 데 도움이 됩니다.

자세한 내용은 *시스템 모니터 안내 및 참조서*의 다음 모니터 요소 설명을 참조하십시오.

- *sec_log_used_top(사용된 2차 로그 공간)*
- *tot_log_used_top(사용된 최대 총 로그 공간)*
- *sec_logs_allocated(현재 할당된 2차 로그)*

이들 모니터 값을 일정 기간 동안 조사해 보면 더 나은 성능 조정을 결정하는 데 도움이 됩니다. 왜냐하면, 평균 값이 사용자가 작업 중에 계속 요구하는 값을 나타낼 수도 있기 때문입니다.

2차 로그 파일의 수(logsecond)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	2 [0 - 126]
측정 단위	카운터
할당 시기	*logprimary*가 불충분할 때 필요에 따라(세부사항은 아래 참조)
해제 시기	데이터베이스 관리 프로그램이 더 이상 필요하지 않다고 결정할 때마다
관련 매개변수	• 475 페이지의 『로그 파일의 크기(logfilsiz)』 • 477 페이지의 『1차 로그 파일의 수(logprimary)』 • 486 페이지의 『로그 유지 작동 가능(logretain)』 • 487 페이지의 『User Exit 사용 가능(userexit)』

이 매개변수는 복구 로그 파일에 대해 작성되고 사용되는 2차 로그 파일의 수를 지정합니다(필요한 경우에만). 1차 로그 파일이 가득 차게 되면 2차 로그 파일(크기 *logfilsiz*)이 필요할 때 한 번에 하나씩, 이 매개변수로 제어할 수 있는 최대수까지 할당됩니다. 이 매개변수가 허용하는 수보다 더 많은 수의 2차 로그 파일이 필요한 경우, 응용프로그램으로 오류 코드가 리턴되고 데이터베이스가 종료됩니다.

2차 로그의 사용 방법에 대한 자세한 내용은 477 페이지의 『1차 로그 파일의 수(logprimary)』를 참조하십시오.

권장사항: 주기적으로 많은 로그 공간이 필요한 데이터베이스에는 2차 로그 파일을 사용하십시오. 예를 들어, 한 달에 한 번 수행되는 응용프로그램에 1차 로그

파일로 제공된 로그 공간 이상이 필요할 수도 있습니다. 2차 로그 파일에는 영구적인 파일 공간이 필요하지 않으므로 이러한 상황에서 더 유리합니다.

데이터베이스 로그 경로 변경(newlogpath)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 널(NULL) [유효한 모든 경로 또는 장치]

관련 매개변수

- 481 페이지의 『로그 파일의 위치(logpath)』
- 510 페이지의 『데이터베이스 일치(database_consistent)』

이 매개변수로 로그 파일이 저장되는 위치를 변경하기 위해 문자열을 최대 242바이트까지 지정할 수 있습니다. 문자열은 경로 이름 또는 원시 장치를 가리킬 수 있습니다. 문자열이 경로 이름을 가리키면 이는 상대 경로 이름이 아닌 완전한 경로 이름이어야 합니다.

주: 파티션된 데이터베이스 환경에서는 자동으로 경로에 노드 번호가 첨부됩니다. 번호를 붙이는 것은 다양한 논리 노드 구성에서 각 경로를 구별하기 위해서입니다.

장치를 지정하려면 운영 체제에서 장치로 식별되는 문자열을 지정하십시오. 예를 들면, 다음과 같습니다.

- Windows NT에서 `\\.\d:` 또는 `\\.\PhysicalDisk5`

  주: 로그를 장치에 기록하려면 Service Pack 3 이상과 함께 Windows NT 4.0이 있어야 합니다.

- UNIX 기반 플랫폼에서 `/dev/rdblog8`

주: AIX, Windows 2000, Windows NT, Solaris, HP-UX, NUMA-Q 및 Linux 플랫폼에서만 장치를 지정할 수 있습니다.

새 설정은 다음 두 경우가 모두 발생할 때에만 *logpath* 값이 됩니다.

- 데이터베이스가 *database_consistent* 매개변수에 의해 표시된 대로 일관성 있는 상태에 있습니다.
- 모든 사용자가 데이터베이스로부터 연결 해제됩니다.

데이터베이스에 처음으로 새 연결이 이루어지면 데이터베이스 관리 프로그램은 로그를 *logpath*에 의해 지정된 새 위치로 이동시킵니다.

이전 로그 경로에 로그 파일이 있을 수 있습니다. 이 로그 파일은 아카이브되지 않았을 수도 있습니다. 이 로그 파일을 수동으로 아카이브해야 합니다. 또한 이 데이터베이스에서 복제를 수행할 경우, 로그 경로가 변경되기 이전의 로그 파일을 필요로 할 수 있습니다. User Exit 사용(*userexit*) 데이터베이스 구성 매개변수가 예로 설정되어 데이터베이스가 구성된 경우와 모든 로그 파일이 DB2에 의해 자동으로 또는 스스로 수동으로 아카이브된 경우, DB2는 로그 파일을 검색하여 복제 프로세스를 완료할 수 있게 됩니다. 그렇지 않으면 파일을 이전 로그 경로에서 새 로그 경로로 복사할 수 있습니다.

권장사항: 로그 파일은 I/O가 많지 **않은** 물리 디스크상에 두는 것이 좋습니다. 예를 들어, 운영 체제 또는 볼륨이 큰 데이터베이스가 있는 디스크에 로그를 두지 마십시오. 이렇게 하면 I/O 대기와 같은 오버헤드를 최소화하면서 효율적인 기록 활동이 가능해집니다.

데이터베이스 시스템 모니터를 사용하여 데이터베이스 로그와 관련된 I/O의 수를 추적할 수 있습니다.

자세한 내용은 *시스템 모니터 안내 및 참조서*에서 다음의 모니터 요소 설명을 참조하십시오.
- *log_reads*(읽혀진 로그 페이지의 수)
- *log_writes*(기록된 로그 페이지의 수)

이전의 데이터 요소는 데이터베이스 로그와 관련된 I/O 활동량을 리턴합니다. 운영 체제 모니터 도구를 사용하여 다른 디스크 I/O 활동 정보를 수집한 다음 두 개의 I/O 활동을 비교할 수 있습니다.

로그 파일의 위치(logpath)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 480 페이지의『데이터베이스 로그 경로 변경(newlogpath)』

이 매개변수에는 로그 목적으로 사용되는 현재 경로가 들어 있습니다. 이 매개변수는 *newlogpath* 매개변수에 대한 변경사항이 적용된 이후에 데이터베이스 관리 프로그램이 설정하므로 사용자가 직접 변경할 수 없습니다.

데이터베이스가 작성될 때 그에 대한 복구 로그 파일이 데이터베이스가 들어 있는 디렉토리의 서브디렉토리에 작성됩니다. 기본값은 데이터베이스에 대해 작성된 디렉토리 아래의 SQLOGDIR이라는 이름의 서브디렉토리입니다.

사용 중인 첫 번째 로그 파일(loghead)

구성 유형 데이터베이스

매개변수 유형 정보용

이 매개변수에는 현재 사용 중인 로그 파일 이름이 들어 있습니다.

데이터베이스 로그 작업

다음 매개변수는 데이터베이스 로그의 유형 및 성능에 영향을 미칠 수 있습니다.

- 『그룹 확약의 수(mincommit)』
- 484 페이지의『복구 범위 및 소프트 점검점 간격(softmax)』
- 486 페이지의『로그 유지 작동 가능(logretain)』
- 487 페이지의『User Exit 사용 가능(userexit)』

그룹 확약의 수(mincommit)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 1 [1 - 25]

측정 단위 카운터

이 매개변수로 최소 수만큼의 확약이 수행될 때까지 로그 레코드가 디스크에 기록되는 것을 지연시킬 수 있습니다. 이러한 지연으로 로그 레코드 기록과 관련된 데이터베이스 관리 프로그램의 오버헤드를 줄일 수 있습니다. 결과적으로, 데이터베이스에 대해 수행 중인 다중 응용프로그램이 매우 짧은 시간 내에 응용프로그램이 요청하는 많은 확약을 처리해야 할 때 성능이 향상됩니다.

이 확약의 그룹화는 이 매개변수의 값이 1보다 크고, 데이터베이스에 연결된 응용프로그램의 수가 이 매개변수 값보다 크거나 같을 때에만 발생합니다. 확약 그룹화가 수행되면 응용프로그램 확약 요청이 1초 동안 또는 확약 요청의 수가 이 매개변수 값과 같아질 때까지 보류됩니다.

이 매개변수에 지정된 값은 변경되는 즉시 적용됩니다. 그러므로 모든 응용프로그램이 데이터베이스에서 연결 해제될 때까지 기다릴 필요가 없습니다.

권장사항: 다중 읽기/쓰기 응용프로그램이 보통 동시 데이터베이스 확약을 요청할 경우, 이 매개변수의 값을 기본값에서 더 증가시키십시오. 값의 증가로 인해 로그 파일 I/O의 발생 빈도는 낮아지고 발생할 때마다 더 많은 로그 레코드를 로그하게 되어 기록 파일 I/O가 효과적으로 이루어집니다.

초당 트랜잭션 수의 샘플을 취하여 이 매개변수가 가장 효율적인 초당 트랜잭션의 최대 수를 제공하도록 조정할 수 있습니다. 최대 활동의 조정은 트랜잭션 집중 기간에 로그 레코드 기록의 오버헤드를 최소화할 수 있습니다.

*mincommit*를 늘리는 경우, *logbufsz* 매개변수도 늘려야 트랜잭션 집중 기간 동안 가득 찬 로그 버퍼의 기록이 강제 시행되는 것을 방지할 수 있습니다. 이 경우, *logbufsz*는 다음과 같아야 합니다.

```
mincommit * (트랜잭션별로 사용된 평균 로그 공간)
```

데이터베이스 시스템 모니터를 사용하면 다음 방법으로 이 매개변수를 조정하는 데 도움이 됩니다.

- 초당 트랜잭션의 최대수 계산

  하루 동안의 모니터 샘플을 취하여 트랜잭션 집중 기간을 결정할 수 있습니다. 다음 모니터 요소를 더하여 총 트랜잭션 수를 계산할 수 있습니다.
  - *commit_sql_stmts(시도된 확약문 수)*
  - *rollback_sql_stmts(시도된 구간 복원문 수)*

이 정보와 사용 가능한 시간소인을 사용하여 초당 트랜잭션 수를 계산할 수 있습니다.

- 트랜잭션당 사용된 로그 공간 계산

  일정 기간 동안 다수의 트랜잭션에 대해 샘플링 기법을 사용하면 다음 모니터 요소와 함께 사용되는 평균 로그 공간을 계산할 수 있습니다.

  – *log_space_used(사용된 작업 단위(UOW) 로그 공간)*

  데이터베이스 시스템 모니터에 대한 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

복구 범위 및 소프트 점검점 간격(softmax)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	100 [1 - 100 * *logprimary*]
측정 단위	1차 로그 파일 한 개 *크기*의 백분율
관련 매개변수	• 475 페이지의 『로그 파일의 *크기*(logfilsiz)』 • 477 페이지의 『1차 로그 파일의 수(logprimary)』

이 매개변수는 다음과 같이 사용됩니다.

- 시스템에 이상(전원 공급 중단 등)이 발생한 다음 복구되어야 할 로그의 수에 영향을 줍니다. 예를 들어, 기본값이 사용되면 데이터베이스 관리 프로그램은 1로 복구되어야 할 로그의 수를 유지하려고 합니다. 이 매개변수의 값으로 300을 지정할 경우, 데이터베이스 관리 프로그램은 3으로 복구되어야 할 로그의 수를 유지하려고 합니다.

  응급 복구시 필요한 로그의 수에 영향을 주기 위해 데이터베이스 관리 프로그램은 이 매개변수를 사용하여 페이지 정리자를 트리거하여 지정된 복구 창 보다 오래된 페이지가 이미 디스크에 기록되었는지 확인합니다.

- 소프트 점검점 빈도를 판별합니다.

전원 공급 중단 등으로 인해 데이터베이스 오류가 생기면 다음과 같은 데이터베이스에 대한 변경이 일어날 수 있습니다.

- 확약되어 있지 않으나, 버퍼 풀의 데이터를 갱신함
- 확약되어 있으나, 버퍼 풀에서 디스크로 기록되지 않음
- 확약되고, 버퍼 풀에서 디스크로 기록됨

데이터베이스가 재시작되면 로그 파일이 데이터베이스의 응급 복구를 수행하는 데 사용되어 데이터베이스가 일관성 있는 상태에 있도록 합니다(즉, 확약된 모든 트랜잭션은 데이터베이스에 적용되고, 모든 미확약 트랜잭션은 데이터베이스에 적용되지 않습니다.)

로그 파일에서 어느 레코드가 데이터베이스에 적용되어야 하는지를 판별하기 위해 데이터베이스 관리 프로그램은 로그 제어 파일을 사용합니다. 이 로그 제어 파일은 주기적으로 디스크에 기록되며, 이 이벤트의 빈도에 따라 데이터베이스 관리 프로그램은 확약된 트랜잭션의 로그 레코드를 적용하거나 이미 버퍼 풀에서 디스크로 기록된 변경사항을 설명하는 로그 레코드를 적용합니다. 이 로그 레코드는 데이터베이스에는 영향을 미치지 않지만, 이러한 로그 레코드를 적용할 경우 데이터베이스 재시작 프로세스에 얼마간의 오버헤드가 발생하기도 합니다.

로그 파일이 가득 찰 때와 소프트 점검점을 실시 중에는 로그 제어 파일이 항상 디스크에 기록됩니다. 이 구성 매개변수를 사용하여 추가 소프트 점검점을 트리거할 수 있습니다.

소프트 점검점의 시기는 *logfilsiz*의 백분율로 제공되어 『현재 상태』와 『기록된 상태』의 차이에 따라 정해집니다. 『기록된 상태』는 디스크에서 로그 제어 파일에 나타난 가장 오래된 유효 로그 레코드로 판별됩니다. 반면에 『현재 상태』는 메모리 내의 로그 제어 정보로 판별됩니다. (가장 오래된 유효 로그 레코드란 복구 프로세스가 읽게 되는 첫 번째 로그 레코드입니다.) 소프트 점검점은 다음 공식으로 계산된 값이 이 매개변수의 값보다 크거나 같으면 사용됩니다.

```
( (기록된 상태와 현재 상태간의 차이) /  logfilsiz ) * 100
```

권장사항: 승인 가능한 복구 창이 한 로그 파일보다 큰지 여부에 따라 이 매개변수의 값을 늘리거나 줄일 수 있습니다. 이 매개변수의 값을 줄이면 데이터베이스 관리 프로그램이 페이지 정리자를 더욱 자주 트리거하게 되고 소프트 점검점을 더

자주 사용하게 됩니다. 이와 같은 조치로 처리에 필요한 로그 레코드의 수와 응급 복구시 처리되는 중복 로그 레코드의 수가 감소됩니다.

그러나 페이지 정리자를 더 많이 트리거하고 소프트 점검점을 자주 사용하게 되면 데이터베이스 로그와 관련된 오버헤드를 증가시켜 데이터베이스 관리 프로그램의 성능에 영향을 줄 수 있습니다. 또한 다음의 경우에는 소프트 점검점을 더 자주 사용한다고 데이터베이스 재시작에 필요한 시간이 줄어들지는 않습니다.

- 확약 지점이 아주 적은 매우 긴 트랜잭션
- 매우 큰 버퍼 풀과 자주 디스크로 다시 기록되지 않는 확약된 트랜잭션이 들어 있는 페이지(비동기 페이지 정리자(cleaner)를 사용하면 이러한 상황을 피할 수 있습니다. 자세한 내용은 448 페이지의 『비동기 페이지 정리자(cleaner)의 수 (num_iocleaners)』를 참조하십시오.)

두 경우에서 모두 메모리에 유지되는 로그 제어 정보가 자주 변경되지 않으면 로그 제어 정보를 디스크에 기록할 때 아무런 이익을 얻을 수 없습니다.

로그 유지 작동 가능(logretain)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 아니오 [복구; 캡쳐; 아니오]

관련 매개변수

- 487 페이지의 『User Exit 사용 가능(userexit)』
- 511 페이지의 『로그 유지 상태 표시기 (log_retain_status)』
- 509 페이지의 『백업 보류 표시기 (backup_pending)』

값은 다음과 같습니다.

- 아니오 - 로그가 유지되지 않음을 나타냅니다.
- 복구 - 로그가 유지되며 포워드 복구에 사용될 수 있음을 나타냅니다. 또한 데이터 복제를 사용할 경우, Capture 프로그램은 로그에 기록된 갱신사항을 변경 테이블에 기록할 수 있습니다.

- 캡쳐 - 로그가 유지만 되어서 Capture 프로그램이 변경 테이블에 갱신사항을 기록할 수 있음을 나타냅니다. 데이터 복제 Capture 프로그램으로 사용된 것을 제거하지 않으면 이 로그는 포워드 복구로 사용될 수도 있습니다.

*logretain*을 복구로 설정하거나 *userexit*를 예로 설정한 경우, 사용 중인 로그 파일은 보유되며 롤 포워드 복구에 사용될 온라인 아카이브 로그 파일이 됩니다. 이를 로그 유지 로그라고 합니다.

*logretain*을 복구로 설정하거나 *userexit*를 예(또는 모두)로 설정한 후에는 데이터베이스 전체를 백업해야 합니다. 이 상태는 *backup_pending* 플래그 매개변수로 표시됩니다.

*logretain*을 아니오로 설정하거나 *userexit*를 아니오로 설정한 경우에는 데이터베이스에 대한 롤 포워드 복구가 사용 가능하지 않습니다.

*logretain*을 캡쳐로 설정할 때 데이터 복제 Capture 프로그램이 로그 프로그램의 사용을 완료하면 PRUNE LOGFILE 명령을 호출하여 로그 파일을 삭제합니다. 데이터베이스에서 롤 포워드 복구를 수행하려면 *logretain*을 캡쳐로 설정해서는 안됩니다.

*logretain*을 아니오로 설정하고 *userexit*를 아니오로 설정하면, 로그가 유지되지 않습니다. 이러한 상황에서 데이터베이스 관리 프로그램은 *logpath* 디렉토리 내의 모든 로그 파일(온라인 아카이브 로그 파일 포함)을 삭제하고, 사용 중인 새 로그를 할당하며, 순환 로그로 전환합니다.

User Exit 사용 가능(userexit)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	아니오 [예; 아니오]
관련 매개변수	• 486 페이지의 『로그 유지 작동 가능(logretain)』 • 511 페이지의 『User Exit 상태 표시기 (user_exit_status)』

- 509 페이지의 『백업 보류 표시기 (backup_pending)』

이 매개변수를 사용할 수 있으면 *logretain* 매개변수의 설정 방법과 상관없이 로그 유지 로그가 수행됩니다. 이 매개변수는 또한 User Exit 프로그램이 로그 파일을 보존하고 검색하는 데 사용되어야 함을 나타냅니다. 로그 파일은 데이터베이스 관리 프로그램이 로그 파일을 닫을 때 아카이브됩니다. 이들은 ROLLFORWARD 유틸리티가 데이터베이스 복원에 이를 사용해야 할 때 검색됩니다.

logretain 또는 *userexit* 매개변수 중 하나가 사용 가능하게 되면 데이터베이스 전체를 백업해야 합니다. 이 상태는 *backup_pending* 플래그 매개변수로 표시됩니다.

이 매개변수를 모두 선택 취소하면 로그가 더 이상 유지되지 않으므로 데이터베이스에 대한 롤 포워드 복구를 사용할 수 없게 됩니다. 이 경우, 데이터베이스 관리 프로그램이 *logpath* 디렉토리에 있는 모든 로그 파일(온라인 아카이브 로그 파일 포함)을 삭제하고, 사용 중인 로그 파일을 새로 할당하며, 순환 로그로 전환합니다.

User Exit 프로그램에 대한 자세한 내용은 *관리 안내서: 구현*의 『데이터베이스 복구를 위한 User Exit』를 참조하십시오.

복구

다음 매개변수는 데이터베이스 복구의 여러 측면에 영향을 미칩니다.

- 489 페이지의 『자동 재시작 가능(autorestart)』
- 489 페이지의 『색인 재작성 시간(indexrec)』
- 491 페이지의 『로드 복구 세션의 기본 수(dft_loadrec_ses)』
- 492 페이지의 『데이터베이스 백업의 수(num_db_backups)』
- 492 페이지의 『복구 실행기록 보유 기간(rec_his_retentn)』
- 493 페이지의 『수정된 페이지 추적 사용(trackmod)』

또한 자세한 내용은 496 페이지의 『분산 작업 단위(DUOW) 복구』를 참조하십시오.

다음 매개변수는 Tivoli Storage Manager(TSM)에 대해 작업할 때 사용됩니다.

- 494 페이지의『Tivoli Storage Manager 관리 클래스(tsm_mgmtclass)』
- 494 페이지의『Tivoli Storage Manager 암호(tsm_password)』
- 495 페이지의『Tivoli Storage Manager 노드 이름(tsm_nodename)』
- 495 페이지의『Tivoli Storage Manager 소유자 이름(tsm_owner)』

자동 재시작 가능(autorestart)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	On [On; Off]

이 매개변수가 on으로 설정되면 응용프로그램이 데이터베이스에 연결될 때 필요하면 데이터베이스 관리 프로그램이 자동으로 데이터베이스 재시작 유틸리티를 호출합니다. *응급 복구*는 데이터베이스 재시작 유틸리티가 수행하는 조작입니다. 이는 응용프로그램이 연결되어 있을 때 데이터베이스가 비정상 종료하는 경우 수행됩니다. 데이터베이스의 비정상 종료는 전원 공급 중단이나 시스템 소프트웨어 장애로 인해 발생할 수 있습니다. 이러한 종료는 실패시 데이터베이스 버퍼 풀에 있으나 디스크에 기록되지 않은 확약 트랜잭션에 적용됩니다. 이는 또한 디스크에 기록될 미확약 트랜잭션을 리턴시키기도 합니다.

*자동 재시작(autorestart)*이 사용 가능하지 않으면 응급 복구가 수행되어야 하는(재시작해야 하는) 데이터베이스에 연결을 시도하는 응용프로그램이 SQL1015N 오류를 수신하게 됩니다. 이 경우, 응용프로그램이 데이터베이스 재시작 유틸리티를 호출하거나, 사용자가 복구 도구 재시작 조작을 선택하여 데이터베이스를 재시작할 수 있습니다.

색인 재작성 시간(indexrec)

구성 유형	데이터베이스 및 데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버

- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

UNIX 데이터베이스 관리 프로그램
재시작 [재시작; 액세스]

OS/2 및 Windows NT 데이터베이스 관리 프로그램 액세스 [재시작; 액세스]

데이터베이스 시스템 설정값 사용 [시스템; 재시작; 액세스]

관련 매개변수 489 페이지의『자동 재시작 가능(autorestart)』

이 매개변수는 데이터베이스 관리 프로그램이 유효하지 않은 색인을 재빌드하게 되는 시기를 나타냅니다. 이 매개변수는 다음 세 가지로 설정될 수 있습니다.

SYSTEM *시스템 설정값 사용*으로 설정하면 데이터베이스 관리 프로그램 구성 파일에 지정될 때 유효하지 않은 색인을 재빌드합니다. (주: 이 설정값은 데이터베이스 구성에만 사용할 수 있습니다.)

ACCESS *색인 액세스* 동안으로 설정하면 색인에에 처음 액세스할 때 유효하지 않은 색인을 재빌드합니다.

RESTART *데이터베이스 재시작* 동안으로 설정하면 RESTART DATABASE 명령이 명시적으로 또는 내재적으로 발행될 때 유효하지 않은 색인이 재빌드됩니다. RESTART DATABASE 명령은 *autorestart* 매개변수가 사용 가능하게 된 경우 내재적으로 발행됩니다.

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

색인은 디스크에 심각한 문제점이 발생하면 유효하지 않게 됩니다. 데이터 자체에 이런 문제점이 발생하면 데이터를 유실할 수도 있습니다. 그러나 색인에 발생하면

색인을 재작성하여 이를 복구할 수 있습니다. 사용자가 데이터베이스에 연결되어 있을 때 색인이 재빌드되면 두 가지 문제점이 발생할 수 있습니다.

- 색인 파일이 재작성됨에 따라 응답 시간에 예상치 않은 성능 저하가 발생할 수 있습니다. 사용자가 테이블에 액세스하여 이 특정 색인을 사용하려면 색인이 재빌드되는 동안 기다려야 합니다.
- 색인 재작성 이후 예상치 않은 잠금이 보유될 수 있는데, 특히 색인 재작성의 원인이 된 사용자 트랜잭션이 COMMIT 또는 ROLLBACK을 전혀 수행하지 않은 경우 그러합니다.

권장사항: 고급 사용자 서버에서 재시작 시간이 중요한 관심사가 아닐 경우, 데이터베이스 재시작시 색인을 재빌드하는 프로세스를 시스템 이상 후에 데이터베이스를 복구하는 프로세스의 일부로서 구성하는 것이 이 옵션에 대한 최상의 선택입니다.

이 매개변수를 『ACCESS』로 설정하면 결과적으로 색인이 재작성될 동안, 데이터베이스 관리 프로그램 성능을 저하시키게 됩니다. 사용자가 특정 색인이나 테이블에 액세스하려면 색인이 재작성되는 동안 기다려야 합니다.

이 매개변수를 『RESTART』로 설정하면 색인 재작성으로 인해 데이터베이스를 재시작하는 데 시간이 더 많이 소요되지만, 일단 데이터베이스를 온라인 상태로 되돌리면 정상적인 처리는 영향을 받지 않습니다.

로드 복구 세션의 기본 수(dft_loadrec_ses)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	1 [1 - 30 000]
측정 단위	카운터

이 매개변수는 테이블 로드의 복구 중 사용되는 기본 세션 수를 지정합니다. 값은 로드 복사를 검색하는 데 사용될 최적의 I/O 세션의 수로 설정되어야 합니다. 로드 복사의 검색은 복원과 유사한 조작입니다. 환경 변수 DB2LOADREC로 지정되는 복사 위치 파일의 항목 전반에 걸쳐 이 매개변수를 대체할 수 있습니다.

로드 검색에 사용되는 기본 버퍼의 수는 이 매개변수 값보다 2가 큰 값입니다. 복사 위치 파일의 버퍼 수도 대체할 수 있습니다.

이 매개변수는 롤 포워드 복구 기능을 사용할 수 있을 때에만 적용됩니다.

로드 복구에 대한 자세한 내용은 *데이터 이동 유틸리티 안내 및 참조서*를 참조하십시오.

데이터베이스 백업의 수(num_db_backups)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	12 [1 -- 32 768]
관련 매개변수	『복구 실행기록 보유 기간(rec_his_retentn)』

이 매개변수는 데이터베이스에 대해 보유할 데이터베이스 백업 수를 지정합니다. 지정된 백업 숫자에 도달하면 기존 백업은 복구 실행기록 파일에 만기된 것으로 표시됩니다. 만기된 베이타베이스 백업에 관련된 테이블 공간 백업 및 로드 사본 백업에 대한 복구 실행기록 파일 항목도 만기된 것으로 표시됩니다. 백업이 만기된 것으로 표시되면 물리적 백업은 저장된 곳(예: 디스크, 테이프, ADSM)으로부터 제거될 수 있습니다. 다음 데이터베이스 백업은 복구 실행기록 파일로부터 만기된 항목을 제거합니다.

데이터베이스 백업이 실행기록 파일에 만기된 것으로 표시되면 DB2 Data Links Manager를 통해 링크된 해당 파일 백업은 아카이브 서버로부터 제거됩니다.

rec_his_retentn 구성 매개변수는 *num_db_backups* 값과 호환되는 값으로 설정되어야 합니다. 예를 들어, *num_db_backup*이 큰 값으로 설정되면 *rec_his_retentn*에 대한 값은 해당 백업 수를 지원할 수 있을 만큼 커야 합니다.

복구 실행기록 보유 기간(rec_his_retentn)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	366 [-1; 0 -- 30 000]

측정 단위 일

관련 매개변수 492 페이지의 『데이터베이스 백업의 수 (num_db_backups)』

이 매개변수는 백업의 실행기록 정보가 보유되어야 하는 일 수(days)를 지정하는 데 사용됩니다. 복구 실행기록 파일이 백업, 복원 및 로드를 추적하는 데 필요없는 경우, 이 매개변수를 작은 수로 설정할 수 있습니다.

이 매개변수 값이 -1이면 복구 실행기록 파일이 사용 가능한 명령 또는 API를 사용하여 명시적으로 제거될 수 있습니다. 값이 -1이 아니면 복구 실행기록 파일이 모든 전체 데이터베이스 백업 후에 제거됩니다.

이 매개변수 값은 *num_db_backups* 매개변수의 값을 대체하지만, *rec_his_retentn* 및 *num_db_backups*는 함께 사용됩니다. *num_db_backups*의 값이 크면 *rec_his_retentn*은 값이 백업 수를 지원할 수 있을 만큼 충분히 커야 합니다.

보유 기간이 짧아도, PRUNE 유틸리티를 FORCE 옵션과 함께 사용하지 않으면 최근의 전체 데이터베이스 백업과 복원 세트는 항상 유지됩니다. 이 유틸리티에 대한 자세한 내용은 *Command Reference* 를 참조하십시오.

수정된 페이지 추적 사용(trackmod)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 아니오 [예, 아니오]

이 매개변수가 "예"로 설정되면 데이터베이스 관리 프로그램은 데이베이스 수정사항을 추적하여 백업 유틸리티는 증분 백업에 의해 검사되고 잠재적으로 백업 이미지에 포함되어야 하는 데이터베이스 페이지의 부속 집합을 검출할 수 있습니다. 매개변수를 "예"로 설정한 후에 증분 백업이 얻을 수 있는 만큼 기준선을 유지하기 위해 데이터베이스 전체를 백어해야 합니다. 또한 매개변수가 사용 가능한 경우와 테이블 공간이 작성딘 경우에는 백업은 테이블 공간을 포함하여 얻을 수 있습니다. 이 백업은 데이터베이스 백업이나 테이블 공간 백업을 할 수 있습니다. 다음 백업은 증분 백업이 테이블 공간에 포함되도록 허용하게 됩니다.

Tivoli Storage Manager 관리 클래스(tsm_mgmtclass)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 널(NULL) [모든 문자열]

Tivoli Storage Manager 관리 클래스는 TSM 서버가 백업되고 있는 오브젝트의 백업 버전을 관리하는 방법을 알려줍니다.

기본값은 TSM 관리 클래스가 없는 상태입니다.

관리 클래스는 Tivoli Storage Manager 관리자로부터 지정됩니다 일단 지정되고 나면 이 매개변수를 관리 클래스 이름으로 설정해야 합니다. TSM 백업이 수행되고 있을 때 데이터베이스 관리 프로그램은 이 매개변수를 사용하여 관리 클래스를 TSM으로 전달합니다.

Tivoli Storage Manager에 대한 자세한 내용은 *Data Recovery and High Availability Guide and Reference*의 『Tivoli Storage Manager』를 참조하십시오.

Tivoli Storage Manager 암호(tsm_password)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 널(NULL) [임의의 문자열]

이 매개변수는 Tivoli Storage Manager(TSM) 제품과 연관되는 암호에 대한 기본 설정을 대체하는 데 사용됩니다. 암호는 또 다른 노드에서 TSM으로 백업된 데이터베이스를 복원할 수 있게 하기 위해 필요합니다.

주: DB2를 통해 백업을 수행하는 동안(예를 들어, BACKUP DATABASE 명령을 사용하여) *tsm_nodename*이 대체될 경우, *tsm_password*를 설정해야 할 수도 있습니다.

기본값은 백업을 수행한 동일한 노드의 TSM에서만 데이터베이스를 복원할 수 있음을 나타냅니다. DB2로 백업을 수행하는 동안 *tsm_nodename*이 대체될 수 있습니다.

Tivoli Storage Manager에 대한 자세한 내용은 *Data Recovery and High Availability Guide and Reference*의 『Tivoli Storage Manager』를 참조하십시오.

Tivoli Storage Manager 노드 이름(tsm_nodename)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	널(NULL) [모든 문자열]

이 매개변수는 Tivoli Storage Manager(TSM) 제품과 연관되는 노드 이름에 대한 기본 설정을 대체하는 데 사용됩니다. 노드 이름은 또 다른 노드에서 TSM으로 백업된 데이터베이스를 복원할 수 있게 하기 위해 필요합니다.

기본값은 백업을 수행한 동일한 노드의 TSM에서만 데이터베이스를 복원할 수 있음을 나타냅니다. DB2를 통해 백업을 수행하는 동안(예를 들어, BACKUP DATABASE 명령을 사용하여) *tsm_nodename*이 대체될 수 있습니다.

Tivoli Storage Manager에 대핸 자세한 내용은 *Data Recovery and High Availability Guide and Reference*의 『Tivoli Storage Manager』를 참조하십시오.

Tivoli Storage Manager 소유자 이름(tsm_owner)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	널(NULL) [임의의 문자열]

이 매개변수는 Tivoli Storage Manager(TSM) 제품 사용과 관련된 소유자가 기본 설정을 대체하는 데 사용됩니다. 소유자 이름은 또 다른 노드에서 ADSM으로 백업된 데이터베이스를 복원할 수 있게 하기 위해서 필요합니다. DB2를 통해 백업을 수행하는 동안(예를 들어, BACKUP DATABASE 명령을 사용하여) *tsm_owner*이 대체될 수 있습니다.

주: 소유자의 이름은 대소문자를 구별해야 합니다.

기본값은 백업을 수행한 동일한 노드의 TSM에서만 데이터베이스를 복원할 수 있음을 나타냅니다.

Tivoli Storage Manager에 대한 자세한 내용은 *Data Recovery and High Availability Guide and Reference*의 『Tivoli Storage Manager』를 참조하십시오.

분산 작업 단위(DUOW) 복구

다음 매개변수는 분산 작업 단위(DUOW) 트랜잭션의 복구에 영향을 미칩니다.

- 『트랜잭션 관리 프로그램 데이터베이스 이름(tm_database)』
- 497 페이지의 『트랜잭션 재동기화 간격(resync_interval)』
- 498 페이지의 『동기 지점 관리 프로그램 로그 파일 경로(spm_log_path)』
- 499 페이지의 『동기 지점 관리 프로그램 이름(spm_name)』
- 499 페이지의 『동기 지점 관리 프로그램 로그 파일 크기(spm_log_file_sz)』
- 501 페이지의 『동기 지점 관리 프로그램 재동기화 에이전트 한계(spm_max_resync)』

트랜잭션 관리 프로그램 데이터베이스 이름(tm_database)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 1ST_CONN [유효한 모든 데이터베이스 이름]

이 매개변수는 DB2 인스턴스 각각에 대한 트랜잭션 관리 프로그램(TM) 데이터베이스 이름을 식별합니다. TM 데이터베이스는 다음과 같습니다.

- 지역 DB2 Universal Database database

- 호스트나 AS/400 시스템에 상주하지 않는 원격 DB2 Universal Database 데이터베이스
- TCP/IP를 통해 액세스하고 동기 지점 관리 프로그램(SPM)은 사용하지 않을 경우 OS/390용 DB2 버전 5 데이터베이스

TM 데이터베이스는 로그 프로그램 및 조정자(coordinator)로 사용되며, 2단계 확약 중 이상 실패 트랜잭션을 복구하는 데 사용됩니다.

이 매개변수를 **1ST_CONN**으로 설정하면 TM 데이터베이스가 사용자가 연결하는 첫 번째 데이터베이스가 됩니다.

분산 작업 단위(DUOW)에 대한 자세한 내용은 *관리 안내서: 계획*의 『분산 데이터베이스』를 참조하십시오.

권장사항: 관리 및 조작을 단순화하기 위해 여러 인스턴스에 걸쳐 몇몇 데이터베이스를 작성하여 이 데이터베이스를 TM 데이터베이스에만 독점적으로 사용할 수 있습니다.

트랜잭션 재동기화 간격(resync_interval)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 180 [1 - 60 000]

측정 단위 초

이 매개변수는 트랜잭션 관리 프로그램(TM), 자원 관리자(RM) 또는 동기 지점 관리 프로그램(SPM)이 TM, RM 또는 SPM에서 미결 2단계 확약 중 이상 실패 트

랜잭션의 복구를 재시도해야 하는 시간 간격을 초 단위로 지정합니다. 이 매개변수는 분산 작업 단위(DUOW) 환경에서 수행 중인 트랜잭션이 있을 때 적용할 수 있습니다.

분산 작업 단위(DUOW)에 대한 자세한 내용은 *관리 안내서: 계획*의 『분산 데이터베이스』를 참조하십시오.

권장사항: 사용자 환경에서 2단계 확약 중 이상 실패 트랜잭션이 데이터베이스에 대한 다른 트랜잭션을 방해하지 않을 경우, 이 매개변수 값을 증가시킬 수도 있습니다. DB2 Connect 게이트웨이를 사용하여 DRDA2 응용프로그램 서버에 액세스할 경우, 지역 데이터 액세스와 방해가 없더라도 2단계 확약 중 이상 실패 트랜잭션이 응용프로그램 서버에 미치는 영향을 고려해야 합니다. 2단계 확약 중 이상 실패 트랜잭션이 없는 경우에는 성능의 영향이 최소화됩니다.

동기 지점 관리 프로그램 로그 파일 경로(spm_log_path)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본값 sqllib/spmlog [임의의 유효한 경로 또는 장치]

이 매개변수는 동기 지점 관리 프로그램(SPM) 로그가 기록되는 디렉토리를 지정합니다. 기본적으로 로그는 볼륨이 높은 트랜잭션 환경에서 I/O 병목 현상을 일으킬 수 있는 sqllib/spmlog 디렉토리에 기록됩니다. 이 매개변수를 사용하여 SPM 로그 파일을 현재 sqllib/spmlog 디렉토리보다 더 빠른 디스크에 놓으십시오. 이렇게 하면 SPM 에이전트간에 더 나은 동시성이 이루어집니다.

동기 지점 관리 프로그램에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

2단계 확약 중 이상 실패 DRDA 트랜잭션 복구에 대한 자세한 내용은 *관리 안내서: 계획*의 『호스트상의 2단계 확약 중 이상 실패 트랜잭션 복구』를 참조하십시오.

동기 지점 관리 프로그램 이름(spm_name)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본값 TCP/IP 호스트 이름에서 파생됨

이 매개변수는 동기 지점 관리 프로그램(SPM) 인스턴스 이름을 데이터베이스 관리 프로그램으로 식별합니다.

동기 지점 관리 프로그램에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

2단계 확약 중 이상 실패 DRDA 트랜잭션 복구에 대한 자세한 내용은 *관리 안내서: 계획*의 『호스트상의 2단계 확약 중 이상 실패 트랜잭션 복구』를 참조하십시오.

동기 지점 관리 프로그램 로그 파일 크기(spm_log_file_sz)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 256 [4 -- 1 000]

측정 단위 페이지

이 매개변수는 동기 지점 관리 프로그램(SPM) 로그 파일 *크기*를 4KB 페이지 단위로 식별합니다. 이 로그 파일은 `sqllib` 아래의 `spmlog` 서브디렉토리에 들어 있으며, SPM이 맨 처음 시작될 때 작성됩니다.

동기 지점 관리 프로그램에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

2단계 확약 중 이상 실패 DRDA 트랜잭션 복구에 대한 자세한 내용은 *관리 안내서: 계획*의 『호스트상의 2단계 확약 중 이상 실패 트랜잭션 복구』를 참조하십시오.

권장사항: 동기 지점 관리 프로그램 로그 파일 *크기*는 성능을 유지보수하기에 충분한 *크기*여야 하지만, 낭비되는 공간이 없도록 지나치게 *크*지 않아야 합니다. 필요한 *크기*는 보호 대화를 사용하는 트랜잭션의 수와, COMMIT 또는 ROLLBACK이 발행되는 횟수에 따라 결정됩니다.

SPM 로그 파일의 *크기*를 변경하려면 다음을 수행하십시오.

1. LIST DRDA INDOUBT TRANSACTIONS 명령을 사용하여 2단계 확약 중 이상 실패 트랜잭션이 남아 있지 않은지를 확인하십시오.
2. 없으면 데이터베이스 관리 프로그램을 중단시키십시오.
3. 데이터베이스 관리 프로그램 구성을 새 SPM 로그 파일 *크기*로 갱신하십시오.

4. $HOME/sqllib 디렉토리로 이동한 다음 rm -fr spmlog를 발행하여 현재의 SPM 로그를 삭제하십시오. (주: 이것은 AIX 명령입니다. 다른 시스템에서는 다른 삭제 명령이 필요합니다.)
5. 데이터베이스 관리 프로그램을 시작하십시오. 지정된 *크기*의 새 SPM *로그*가 데이터베이스 관리 프로그램 시작 중에 생성됩니다.

동기 지점 관리 프로그램 재동기화 에이전트 한계(spm_max_resync)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	20 [10 -- 256]

이 매개변수는 재동기화 조작을 동시적으로 수행하는 에이전트의 수를 식별합니다.

2단계 확약 중 이상 실패 DRDA 트랜잭션 복구에 대한 자세한 내용은 *관리 안내서: 계획*의 『호스트상의 2단계 확약 중 이상 실패 트랜잭션 복구』를 참조하십시오.

동기 지점 관리 프로그램에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

데이터베이스 관리

데이터베이스에 대한 정보를 제공하거나 데이터베이스 관리에 영향을 미치는 다수의 매개변수가 제공됩니다. 이는 다음과 같이 그룹화됩니다.

- 『Query Enable』
- 『속성』
- 506 페이지의 『DB2 Data Links Manager』
- 509 페이지의 『상태』
- 512 페이지의 『컴파일러 설정』

Query Enable

다음 매개변수는 Query Enable의 제어에 대한 정보를 제공합니다.

- 『동적 SQL 조회 관리(dyn_query_mgmt)』

동적 SQL 조회 관리(dyn_query_mgmt)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	0 (DISABLE) [1(ENABLE), 0 (DISABLE)]

이 매개변수는 DB2 Query Patroller가 설치된 곳과 관련됩니다. 데이터베이스 구성 매개변수인 *dyn_query_mgmt*가 『ENABLE』로 설정되고 동적 조회 비용이 사용자나 그룹에 대한 trap_threshold(DB2 Query Patroller 사용자 프로파일 테이블에서 지정한)를 초과할 경우, 이 조회는 DB2 Query Patroller에서 포착됩니다. trap_threshold는 사용자가 DB2 Query Patroller에서 설정한 조회 포착용 트리거로, 비용을 기반으로 합니다. 동적 조회가 포착되면 사용자가 런타임 매개변수를 지시할 수 있는 대화 상자가 나타납니다.

*dyn_query_mgmt*가 『DISABLE』로 설정되면 어떤 조회도 포착되지 않습니다.

속성

다음 매개변수는 데이터베이스에 대한 일반 정보를 제공합니다.

- 503 페이지의 『구성 파일 릴리스 레벨(release)』

- 『데이터베이스 릴리스 레벨(database_level)』
- 504 페이지의 『데이터베이스 지역(territory)』
- 504 페이지의 『데이터베이스 국가 코드(country)』
- 504 페이지의 『데이터베이스 코드 세트(codeset)』
- 505 페이지의 『데이터베이스 코드 페이지(codepage)』
- 505 페이지의 『조합 정보(collate_info)』
- 506 페이지의 『복사 방지 가능(copyprotect)』

*copyprotect*를 제외한 나머지 매개변수는 단지 정보 전달용으로 제공됩니다.

구성 파일 릴리스 레벨(release)

구성 유형 데이터베이스 관리 프로그램, 데이터베이스

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 정보용

관련 매개변수 『데이터베이스 릴리스 레벨(database_level)』

이 매개변수는 구성 파일의 릴리스 레벨을 지정합니다.

데이터베이스 릴리스 레벨(database_level)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 『구성 파일 릴리스 레벨(release)』

이 매개변수는 데이터베이스를 사용할 수 있는 데이터베이스 관리 프로그램의 릴리스 레벨을 나타냅니다. 미완료 또는 실패한 이주의 경우, 이 매개변수는 이주되

지 않은 데이터베이스의 릴리스 레벨을 반영하는 것으로 *release* 매개변수(데이터베이스 구성 파일의 릴리스 레벨)와는 다릅니다. 그렇지 않으면 *database_level*의 값은 *release* 매개변수의 값과 동일합니다.

데이터베이스 지역(territory)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 『데이터베이스 국가 코드(country)』

이 매개변수는 데이터베이스를 작성하는 데 사용된 지역을 표시합니다. 데이터베이스 관리 프로그램은 지역을 사용하여 *country* 매개변수 값을 판별합니다. 데이터베이스 관리 프로그램이 지역을 사용하는 방법에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

데이터베이스 국가 코드(country)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 『데이터베이스 지역(territory)』

이 매개변수는 데이터베이스를 작성하는 데 사용되는 국가 코드를 표시합니다. *country* 매개변수는 *territory* 매개변수에 따라 결정됩니다. 데이터베이스 관리 프로그램이 국가 코드를 사용하는 방법에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

데이터베이스 코드 세트(codeset)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 505 페이지의 『데이터베이스 코드 페이지(codepage)』

이 매개변수는 데이터베이스를 작성하는 데 사용된 코드 세트를 표시합니다. 데이터베이스 관리 프로그램은 코드 세트를 사용하여 *codepage* 매개변수 값을 결정합

니다. 데이터베이스 관리 프로그램이 코드 세트를 사용하는 방법에 대한 자세한 내용은 *관리 안내서: 계획*의 자국어 지원 부록을 참조하십시오.

데이터베이스 코드 페이지(codepage)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 504 페이지의 『데이터베이스 코드 세트(codeset)』

이 매개변수는 데이터베이스를 작성하는 데 사용된 코드 페이지를 표시합니다. *codepage* 매개변수는 *codeset* 매개변수에 따라 결정됩니다. 데이터베이스 관리 프로그램이 코드 페이지를 사용하는 방법에 대한 자세한 내용은 *관리 안내서: 계획*의 자국어 지원 부록을 참조하십시오.

조합 정보(collate_info)

이 매개변수는 GET DATABASE CONFIGURATION API를 사용하여 표시할 수만 있습니다. 이것은 명령행 처리기 또는 제어 센터를 통해서는 표시할 수 **없습니다**.

구성 유형 데이터베이스

매개변수 유형 정보용

이 매개변수는 260바이트의 데이터베이스 조합 정보를 제공합니다. 처음 256바이트는 데이터베이스 조합 순서를 지정하며, 여기서 바이트 『n』에는 데이터베이스의 코드 페이지에서 기반이 되는 10진 표현이 『n』인 코드 포인트의 정렬 가중치가 포함됩니다.

마지막 4바이트에는 조합 순서에 대한 내부 정보가 포함됩니다. 이것을 데이터베이스 플랫폼에 적용 가능한 정수로 처리할 수 있습니다. 다음 세 가지 값이 있습니다.

- **0** - 고유하지 않은 가중치를 포함하는 순서
- **1** - 고유한 모든 가중치를 포함하는 순서
- **2** - 문자열이 바이트 단위로 비교되는 식별 순서인 순서

위의 내부 유형 정보를 사용할 경우, 다른 플랫폼의 데이터베이스의 정보를 검색할 때 바이트 리버설을 고려해야 합니다.

데이터베이스 작성시에 조합 순서를 지정할 수 있습니다.

복사 방지 가능(copyprotect)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	아니오 [예; 아니오]

이 매개변수는 복사 방지 속성을 사용 가능하며, 기본적으로 사용 불가능합니다. 데이터베이스 관리 프로그램 버전 2 이전에는 기본값이 복사 방지 속성이 사용 가능한 것이었습니다.

이 매개변수는 UNIX 기반 환경에는 적용되지 않습니다.

백업 데이터베이스와 복원 데이터베이스 유틸리티는 *copyprotect* 매개변수에 영향을 받지 않습니다. 복사 방지된 데이터베이스를 백업하여 다른 워크스테이션에 복원한 다음 데이터베이스를 카탈로그화하여 데이터베이스에 액세스할 수 있습니다.

주의사항: 데이터베이스 관리 프로그램이나 운영 체제를 재설치하기 전에 모든 데이터베이스에서 복사 방지를 제거하십시오. 복사 방지를 제거하지 않을 경우, 데이터베이스에 액세스하려 할 때 오류가 생깁니다. 재설치한 후, 복사 방지를 사용할 수 있습니다.

DB2 Data Links Manager

다음은 DB2 Data Links Manager에 관련된 매개변수입니다.

- 『데이터 링크 액세스 토큰 만기 간격(dl_expint)』
- 507 페이지의 『데이터 링크 사본 수(dl_num_copies)』
- 507 페이지의 『삭제 후 데이터 링크 시간(dl_time_drop)』
- 508 페이지의 『데이터 링크 토큰 알고리즘(dl_token)』
- 508 페이지의 『데이터 링크 토큰을 대문자로만 표시(dl_upper)』
- 509 페이지의 『데이터 링크 지원 사용(datalinks)』

데이터 링크 액세스 토큰 만기 간격(dl_expint)

구성 유형	데이터베이스

매개변수 유형 구성 가능

기본 [범위] 60 [-1, 1 -- 31 536 000]

측정 단위 초

이 매개변수는 생성된 파일 액세스 제어 토큰이 유효한 시간 간격(초)을 지정합니다. 토큰 수(초)는 토큰이 생성되는 시점부터 유효합니다. 데이터 링크 파일시스템 필터는 토큰이 만기 시간에 대해 유효한지를 점검합니다.

파일 액세스 제어 토큰에 대한 자세한 내용은 *DB2 Data Links Manager 빠른 시작* 책을 참조하십시오.

이 매개변수의 기본값은 60초입니다. 이 매개변수가 ″-1″로 설정되면 액세스 제어 토큰이 만기됩니다. 일시적인 해결책은 이 매개변수를 31536000(초)과 같이 최대 값으로 설정하면 됩니다. 즉, 이 값은 모든 응용프로그램을 충족시킬 수 있는 만기 시간 1년에 해당됩니다.

이 매개변수는 『READ PERMISSION DB』를 지정하는 DATALINK 컬럼에 적용됩니다.

데이터 링크 사본 수(dl_num_copies)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 0 [0 - 15]

이 매개변수는 파일이 데이터베이스에 링크될 때 아카이브 서버(예: TSM 서버)에서 추가로 수행될 파일의 사본 수를 지정합니다.

이 매개변수에 대한 기본값은 0입니다.

이 매개변수는 『복구=예』를 지정하는 DATALINK 컬럼에 적용됩니다.

삭제 후 데이터 링크 시간(dl_time_drop)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 1 [0 -- 365]

측정 단위 일

이 매개변수는 DROP DATABASE가 발행된 후 아카이브 서버(예: TSM 서버)에 파일을 보유할 기간(일 수)을 지정합니다.

이 매개변수의 기본값은 1일입니다. 값 0은 파일이 DROP 명령이 발행된 후 즉시 아카이브 서버에서 삭제된다는 것을 의미합니다. (실제 파일은 DATALINK 컬럼에 대해 ON UNLINK DELETE 매개변수가 지정되지 않으면 삭제되지 않습니다.)

이 매개변수는 『복구=예』를 지정하는 DATALINK 컬럼에 적용됩니다.

데이터 링크 토큰 알고리즘(dl_token)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] MAC0 [MAC0; MAC1]

이 매개변수는 DATALINK 파일 액세스 제어 토큰의 생성에 사용되는 알고리즘을 지정합니다. MAC1(메시지 인증 코드)은 MAC0보다 보안이 강화된 메시지 인증 코드를 생성하지만, 성능 오버헤드가 증가됩니다.

파일 액세스 제어 토큰에 대한 자세한 내용은 *DB2 Data Links Manager 빠른 시작* 책을 참조하십시오.

이 매개변수는 『READ PERMISSION DB』를 지정하는 DATALINK 컬럼에 적용됩니다.

데이터 링크 토큰을 대문자로만 표시(dl_upper)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 아니오 [예; 아니오]

이 매개변수는 파일 액세스 제어 토큰이 대문자를 사용할 것인지 여부를 나타냅니다. 값이 『YES』이면 액세스 제어 토큰의 모든 문자가 대문자임을 나타냅니다. 값이 『NO』이면 토큰이 대문자 및 소문자를 모두 포함할 수 있음을 지정합니다.

파일 액세스 제어 토큰에 대한 자세한 내용은 *DB2 Data Links Manager 빠른 시작* 책을 참조하십시오.

이 매개변수는 『READ PERMISSION DB』를 지정하는 DATALINK 컬럼에 적용됩니다.

데이터 링크 지원 사용(datalinks)

구성 유형 데이터베이스 관리 프로그램

매개변수 유형 구성 가능

기본 [범위] 아니오 [예; 아니오]

이 매개변수는 데이터 링크 지원이 사용 가능한지 여부를 지정합니다. 『YES』는 데이터 링크 지원이 원시 파일 시스템(AIX의 JFS)에 저장된 데이터 링크 관리 프로그램 링크 파일에 대해 사용 가능하도록 지정합니다. 『NO』는 데이터 링크 지원이 사용 가능하지 않도록 지정합니다.

상태

다음 매개변수는 데이터베이스의 상태에 대한 정보를 제공합니다.

- 『백업 보류 표시기(backup_pending)』
- 510 페이지의 『데이터베이스 일치(database_consistent)』
- 510 페이지의 『롤 포워드 보류 표시기(rollfwd_pending)』
- 511 페이지의 『로그 유지 상태 표시기(log_retain_status)』
- 511 페이지의 『User Exit 상태 표시기(user_exit_status)』
- 511 페이지의 『복원 보류(restore_pending)』
- 511 페이지의 『다중 페이지 파일 할당 사용(multipage_alloc)』

백업 보류 표시기(backup_pending)

구성 유형 데이터베이스

매개변수 유형 정보용

이 매개변수가 on으로 설정되면 데이터베이스에 액세스하기 전에 데이터베이스 전체를 백업해야 합니다. 이 매개변수는 데이터베이스가 복구 불가 상태에서 복구 가능 상태로 이동할 때만 on으로 설정됩니다. (즉, 초기에 *logretain*과 *userexit* 매개변수는 둘다 NO로 설정되지만, 이 매개변수 중 하나 또는 둘다 YES로 설정되면 데이터베이스 구성을 갱신할 수 있게 됩니다.)

데이터베이스 일치(database_consistent)

구성 유형 데이터베이스

매개변수 유형 정보용

이 매개변수는 데이터베이스가 일치 상태에 있는지를 나타냅니다.

YES는 모든 트랜잭션이 확약 또는 구간 복원되어 데이터에 일관성이 있음을 나타냅니다. 데이터베이스에 일관성이 있을 때 시스템이 『파과』되는 경우, 데이터베이스를 사용 가능하도록 하기 위해 특수 조치를 취할 필요가 없습니다.

NO는 트랜잭션이나 일부 다른 타스크가 데이터베이스에서 보류 중이어서 데이터가 이 지점에서 일치하지 않음을 나타냅니다. 데이터베이스에 일관성이 없을 때 시스템이 『파괴』되면 데이터베이스를 사용 가능하게 하기 위해 RESTART DATABASE 명령을 사용하여 데이터베이스를 재시작해야 합니다. RESTART DATABASE 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

롤 포워드 보류 표시기(rollfwd_pending)

구성 유형 데이터베이스

매개변수 유형 정보용

이 매개변수는 다음 상태 중 하나를 나타냅니다.

- **DATABASE**: 이 데이터베이스에 롤 포워드 복구 프로시듀어가 필요합니다
- **TABLESPACE**: 하나 이상의 테이블 공간에 롤 포워드가 되어야 합니다
- **NO**: 데이터베이스를 사용할 수 있으며 롤 포워드 복구가 필요 없습니다

(ROLLFORWARD DATABASE를 사용한) 복구를 데이터베이스 또는 테이블 공간에 액세스하기 전에 완료해야 합니다. ROLLFORWARD DATABASE에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

로그 유지 상태 표시기(log_retain_status)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 486 페이지의 『로그 유지 작동 가능(logretain)』

이 매개변수가 설정되어 있으면 이것은 로그 파일이 롤 포워드 복구용으로 보유되어 있음을 나타냅니다.

이 매개변수는 *logretain* 매개변수 설정이 복구와 같을 때 설정됩니다.

User Exit 상태 표시기(user_exit_status)

구성 유형 데이터베이스

매개변수 유형 정보용

관련 매개변수 487 페이지의 『User Exit 사용 가능(userexit)』

이 매개변수가 예로 설정되면 이는 데이터베이스 관리 프로그램이 롤 포워드 복구에 사용될 수 있고, 데이터베이스 관리 프로그램이 호출할 때 User Exit 프로그램이 로그 파일을 아카이브하고 검색하는 데 사용됨을 나타냅니다.

복원 보류(restore_pending)

구성 유형 데이터베이스

매개변수 유형 정보용

이 매개변수는 데이터베이스에 RESTORE PENDING 상태가 존재하는지 여부를 지정합니다.

다중 페이지 파일 할당 사용(multipage_alloc)

구성 유형 데이터베이스

매개변수 유형 정보용

다중 페이지 파일 할당은 삽입 성능을 향상시키는 데 사용됩니다. 이 작업은 SMS 테이블 공간에만 적용됩니다. 사용 가능할 경우, 모든 SMS 테이블 공간에 영향을 줍니다. 개별 SMS 테이블 공간에 대한 선택은 불가능합니다.

이 매개변수의 기본값은 No로서, 다중 페이지 파일 할당이 사용 불가능한 상태입니다.

데이터베이스 작성 이후, 매개변수를 예로 설정하면 다중 페이지 파일 할당 이 상용 가능하게 됩니다. 이러한 설정은 *db2empfa* 도구를 사용하여 수행됩니다. 일단 매개변수를 예로 설정하면 다시 아니오로 변경할 수 없습니다.

해당 도구에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

컴파일러 설정

다음 매개변수는 컴파일러에 영향을 미치는 정보를 제공합니다.

- 『산술 예외 시 계속(dft_sqlmathwarn)』
- 514 페이지의 『기본 등급(dft_degree)』
- 515 페이지의 『기본 조회 최적화 클래스(dft_queryopt)』
- 516 페이지의 『기본 새로 고침 유효 기간(dft_refresh_age)』
- 516 페이지의 『보유되어 자주 사용되는 값의 수(num_freqvalues)』
- 517 페이지의 『컬럼에 대한 quantile 수(num_quantiles)』

산술 예외 시 계속(dft_sqlmathwarn)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	아니오[아니오, 예]

이 매개변수는 SQL문이 컴파일을 하는 동안 오류와 경고로 산술 오류와 검색 변환 오류의 처리를 판별하는 기본값을 설정합니다. 정적 SQL문의 경우, 바인드시 이 매개변수 값은 패키지와 연관됩니다. 동적 SQL DML문의 경우에는 명령문이 준비될 때 이 매개변수 값을 사용합니다.

주의: 데이터베이스에 대해 *dft_sqlmathwarn* 값을 변경하면 점검 제한조건, 트리거 그리고 산술 표현식을 포함하는 뷰의 작동이 변경될 수 있습니다. 위의 변경은 데이터베이스의 데이터 무결성에 영향을 줄 수도 있습니다. 새 산술 예외 처리 작동이 얼마나 점검 제한조건, 트리거 및 뷰에 영향을 미치는 지를 주의깊게 평가한 후, 데이터베이스에 대한 *dft_sqlmathwarn*의 설정만 변경해야 합니다. 일단 변경되면 후속 변경은 같은 평가를 요구합니다.

한 예로서, 나눗셈 산술 연산을 포함한 다음과 같은 점검 제한조건을 고려해 보십시오.

```
A/B > 0
```

*dft_sqlmathwarn*이 『아니오』이고 B=0인 INSERT를 시도하면 0으로 나누기는 산술 오류로 처리됩니다. 삽입 조작은 DB2가 제한조건을 검사할 수 없기 때문에 실패합니다. *dft_sqlmathwarn*을 『예』로 변경할 경우, 0으로 나누기는 NULL 결과로 산술 경고를 처리합니다. NULL 결과로 『>』 술어가 UNKNOWN으로 평가되어 삽입 조작은 성공합니다. *dft_sqlmathwarn*이 다시 『아니오』로 변경될 경우, 0으로 나누기 오류로 인해 DB2가 제한조건을 평가할 수 없기 때문에 같은 행에 삽입하려는 시도는 실패합니다. B=0으로 삽입된 행은 *dft_sqlmathwarn*이 『예』일 때 테이블에 남아 있을 수 있어 선택될 수 있습니다. 제한조건을 평가하는 행으로 갱신하는 것은 실패하고, 반면 제한조건 재평가가 필요하지 않는 행으로의 갱신은 성공합니다.

*dft_sqlmathwarn*을 『아니오』에서 『예』로 변경하기 전에 산술식의 널(NULL)을 명시적으로 처리하는 제한조건을 재작성해야 합니다. 예를 들면, 다음과 같습니다.

```
( A/B > 0 ) AND ( CASE
                    WHEN A IS NULL THEN 1
                    WHEN B IS NULL THEN 1
                    WHEN A/B IS NULL THEN 0
                    ELSE 1
                    END
                  = 1 )
```

위와 같이 A와 B가 널(NULL) 입력 가능하면 사용될 수 있습니다. 그러나 A와 B가 널(NULL)로 입력 가능하지 않으면 대응되는 IS NULL WHEN절은 제거될 수 있습니다.

*dft_sqlmathwarn*을 『Yes』에서 『No』로 변경하기 전에 다음과 같은 술어를 사용하여 불일치할 수 있는 데이터를 먼저 점검해야 합니다.

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

불일치 행이 분리되면 *dft_sqlmathwarn*을 변경하기 전에 불일치를 정정하는 적당한 조치를 취해야 합니다. 그리고 변경 후에 산술식으로 제한조건을 수동으로 재점검할 수 있습니다. 이를 수행하려면 점검 보류 상태에 영향 받은 테이블을 우선 배치시키고(SET CONSTRAINTS문의 OFF로) 난 후, 테이블 검사를 요청하십시오(SET CONSTRAINTS문의 IMMEDIATE CHECKED절로). 제한조건이 평가되지 못하게 하는 불일치 데이터는 산술 오류에 나타납니다.

권장사항: 특별히 산술 예외를 포함하는 조회를 처리해야 되는 경우가 아니면 기본 설정인 아니오를 사용하십시오. 그리고 값을 예로 지정하십시오. 다른 데이터베이스 관리 프로그램에서 발생하는 산술 예외와 상관없이 결과를 산출하는 SQL문을 처리할 경우 이러한 상황이 발생할 수 있습니다.

기본 등급(dft_degree)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	1 [-1, 1 - 32 767]
관련 매개변수	543 페이지의 『병렬의 처리 최대 조회 수준(max_querydegree)』

이 매개변수는 CURRENT DEGREE 특수 레지스터 및 DEGREE 바인드 옵션의 기본값을 지정합니다.

기본값은 1입니다.

값 1은 파티션 내 병렬 처리가 없다는 의미입니다. 값 -1은 최적화 알고리즘이 프로세서 수 및 조회 유형에 기초하여 파티션 내 병렬 처리 수준을 결정한다는 의미입니다.

SQL문의 파티션 내 병렬 처리 수준은 CURRENT DEGREE 특수 레지스터 또는 DEGREE 바인드 옵션을 사용하여 명령문 컴파일시 지정됩니다. 사용 중인 응

응용프로그램의 파티션 내 병렬 처리의 최대 런타임 수준은 SET RUNTIME DEGREE 명령을 사용하여 지정됩니다. 병렬 처리의 최대 조회 수준 (*max_querydegree*) 구성 매개변수는 모든 SQL 조회에 대해 파티션 내 병렬 처리의 최대 조회 수준을 지정합니다.

사용되는 실제 런타임 등급은 다음 중 가장 낮은 값입니다.

- *max_querydegree* 구성 매개변수
- 응용프로그램 런타임 등급
- SQL문 컴파일 등급

기본 조회 최적화 클래스(dft_queryopt)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	5 [0 -- 9]
측정 단위	조회 최적화 클래스(아래 참조)

조회 최적화 클래스는 SQL 조회를 컴파일할 때 다른 등급의 최적화를 사용하도록 최적화 알고리즘에 지시하는 데 사용됩니다. 이 매개변수는 SET CURRENT QUERY OPTIMIZATION문 또는 QUERYOPT 바인드 명령이 사용되지 않을 때 사용되는 기본 조회 최적화 클래스를 설정하여 추가 융통성을 제공합니다.

현재 정의되어 있는 조회 최적화 클래스는 다음과 같습니다.

0 - 최소 조회 최적화
1 - 대략 DB2 버전 1 정도
2 - 미세 최적화
3 - 중간 조회 최적화
5 - 액세스 플랜을 선택할 때 들어가는 노력을 제한하기 위해 발견 프로그램을 갖춘 주요 조회 최적화. 이는 기본값입니다.
7 - 주요 조회 최적화
9 - 최대 조회 최적화

권장사항: 적당한 조회 최적화 클래스 선택에 대한 자세한 내용은 74 페이지의 『최적화 클래스 조정』을 참조하십시오.

프로그램이 데이터베이스 구성 매개변수를 검색하고 수정하는 방법에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

기본 새로 고침 유효 기간(dft_refresh_age)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 0 [0, 99999999999999 (ANY)]

이 매개변수에는 CURRENT REFRESH AGE 특수 레지스터가 지정되지 않은 경우에 REFRESH AGE에 사용되는 기본값이 있습니다. 이 매개변수는 데이터 유형이 DECIMAL(20,6)인 시간소인 지속기간 값을 지정합니다. 특정 REFRESH DEFERRED 요약 테이블에서 REFRESH TABLE문이 처리되므로, 그 요약 테이블을 조회 처리 최적화에도 사용할 수 있는 동안 이 시간 지속기간은 최대 지속기간을 나타냅니다. CURRENT REFRESH AGE의 값이 99999999999999(ANY)이고, QUERY OPTIMIZATION 클래스가 5 이상일 경우, 동적 SQL 조회 처리를 최적화하기 위해 REFRESH DEFERRED 요약 테이블이 고려됩니다.

보유되어 자주 사용되는 값의 수(num_freqvalues)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 10 [0 -- 32 767]

측정 단위 카운터

관련 매개변수

- 517 페이지의 『컬럼에 대한 quantile 수 (num_quantiles)』
- 419 페이지의 『통계 힙 *크기*(stat_heap_sz)』

이 매개변수로 RUNSTATS 명령에 WITH DISTRIBUTION 옵션이 지정될 때 수집되는 『가장 자주 사용되는 값』의 수를 지정할 수 있습니다. 이 매개변수 값을 늘리면 통계 수집시 사용된 통계 힙(*stat_heap_sz*)의 양이 증가합니다.

『가장 자주 사용되는 값』 통계는 최적화 알고리즘이 컬럼 내의 데이터 값 분산을 이해하는 데 도움이 됩니다. 값이 클수록 더 많은 정보가 SQL 최적화 알고리즘에서 사용 가능하게 되지만, 추가 카탈로그 공간이 필요합니다. 0이 지정되면 분산 통계 수집을 요청해도 자주 사용되는 값 통계가 보유되지 않습니다.

이 매개변수를 갱신하면 최적화 알고리즘이 일관성 없이 분산된 데이터의 몇몇 술어(=, <, >, IS NULL, IS NOT NULL)에 대해 더 나은 선택성 추정을 확보하는 데 도움이 됩니다. 좀더 정확한 선택성 계산은 더 효율적인 액세스 플랜을 선택하는 것입니다.

이 매개변수의 값을 변경한 후 다음을 수행해야 합니다.
- 모든 사용자가 데이터베이스로부터 연결 해제되고 사용자가 데이터베이스로 다시 연결된 후에 RUNSTATS 명령 수행
- 정적 SQL이 들어 있는 패키지 리바인드

자세한 내용은 135 페이지의 『분산 통계 수집 및 사용』을 참조하십시오.

권장사항: 이 매개변수를 갱신하려면 일반적으로 선택 술어가 있는 주요 컬럼(가장 주요한 테이블에서)의 불균일 정도를 판별해야 합니다. 지시된 랭킹에 컬럼에서 발생한 값의 수를 제공하는 SQL SELECT문을 사용하여 이 판별을 할 수 있습니다. 고르게 분포되고 고유하며 긴 또는 LOB 컬럼을 고려하지는 마십시오. 이 매개변수의 타당한 실제 값은 10에서 100 사이입니다.

자주 사용되는 값 통계를 수집하는 프로세스에는 충분한 CPU 및 메모리 (*stat_heap_sz*) 자원이 필요함을 주의하십시오.

컬럼에 대한 quantile 수(num_quantiles)

구성 유형	데이터베이스
매개변수 유형	구성 가능
기본 [범위]	20 [0 - 32 767]
측정 단위	카운터
관련 매개변수	• 516 페이지의 『보유되어 자주 사용되는 값의 수(num_freqvalues)』

• 419 페이지의 『통계 힙 크기(stat_heap_sz)』

이 매개변수는 RUNSTATS 명령에 WITH DISTRIBUTION 옵션이 지정될 때 수집되는 quantile의 수를 제어합니다. 이 매개변수 값을 늘리면 통계 수집시 사용된 통계 힙(*stat_heap_sz*)의 양이 증가합니다.

『quantile』 통계는 최적화 알고리즘이 컬럼 내 데이터 값의 분산을 이해하는 데 도움이 됩니다. 값이 클수록 더 많은 정보가 SQL 최적화 알고리즘에서 사용 가능하게 되지만, 추가 카탈로그 공간이 필요합니다. 0 또는 1이 지정되면 분산 통계 수집을 요청해도 quantile 통계가 보유되지 않습니다.

이 매개변수를 갱신하면 일관성 없이 분산된 데이터의 범위 술어에 대해 더 나은 선택성 추정을 확보하는 데 도움이 됩니다. 다른 최적화 알고리즘의 결정사항 중에서도, 이 정보가 색인 스캔 또는 테이블 스탠의 선택 여부에 강력한 영향을 미칩니다. (자주 발생하는 값의 범위에 액세스할 때는 테이블 스캔을 사용하고, 자주 발생하지 않는 값의 범위에 액세스할 때는 색인 스캔을 사용하는 것이 더 효과적입니다.)

이 매개변수의 값을 변경한 후 다음을 수행해야 합니다.
- 모든 사용자가 데이터베이스로부터 연결 해제되고 사용자가 다시 연결된 후에 RUNSTATS 명령 수행
- 정적 SQL이 들어 있는 패키지 리바인드

자세한 내용은 135 페이지의 『분산 통계 수집 및 사용』을 참조하십시오.

권장사항: 이 매개변수의 기본값을 사용하면 단면 범위 술어(>, >=, < 또는 <=)의 최대 추정 오류는 약 2.5%로, BETWEEN 술어의 최대 추정 오류는 5%로 보장됩니다. quantile 수에 접근하는 단순한 방법은 다음과 같습니다.
- 범위 조회의 행 수를 추정할 때 허용할 만한 최대 오류를 백분율 P로 결정하십시오.
- quantile 수는 대부분의 술어가 BETWEEN 술어일 경우에는 약 100/P이고, 대부분의 술어가 범위 술어(<, <=, > 또는 >=) 외의 다른 유형일 경우에는 50/P입니다.

예를 들어, 25 quantile은 BETWEEN 술어에 대해 최대 4%와 ″>″ 술어에 대해 2%로 추정 오류를 일으킵니다. 이 매개변수에 대한 타당한 실제 값은 10에서 50 사이입니다.

통신

다음 매개변수 그룹은 클라이언트/서버 환경에서 DB2를 사용하는 것에 대한 정보를 제공합니다.

- 『통신 프로토콜 설정』
- 525 페이지의 『분산 서비스』
- 531 페이지의 『DB2 발견』

통신 프로토콜 설정

다음 매개변수를 사용하여 데이터베이스 클라이언트 및 데이터베이스 서버를 구성할 수 있습니다.

- 『NetBIOS 워크스테이션 이름(nname)』
- 520 페이지의 『TCP/IP 서비스 이름(svcename)』
- 521 페이지의 『APPC 트랜잭션 프로그램 이름(tpname)』
- 522 페이지의 『IPX/SPX 파일 서버 이름(fileserver)』
- 523 페이지의 『IPX/SPX DB2 서버 오브젝트 이름(objectname)』
- 524 페이지의 『IPX/SPX 소켓 번호(ipx_socket)』

NetBIOS 워크스테이션 이름(nname)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버

- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

이 매개변수를 통해 NetBIOS LAN 환경의 워크스테이션에 있는 데이터베이스 인스턴스에 고유 이름을 지정할 수 있습니다. 이 *nname*은 워크스테이션에 대한 NetBIOS로 등록될 실제 NetBIOS 이름에 대한 기본입니다.

NetBIOS 프로토콜은 이러한 NetBIOS 이름을 사용하여 통신 연결을 설정하기 때문에, 클라이언트와 서버 둘 다에 대해 *nname* 매개변수를 설정해야 합니다.

클라이언트 응용프로그램은 액세스할 데이터베이스를 가지고 있는 서버의 *nname*을 알아야 합니다. 서버의 *nname*은 CATALOG NETBIOS NODE 명령을 사용하여 클라이언트의 노드 디렉토리에서 『server-nname』 매개변수로서 카탈로그화되어야 합니다.

서버 노드의 *nname*이 새 이름으로 변경될 경우, 해당 서버에서 데이터베이스에 액세스하는 모든 클라이언트는 서버에 대해 이 새 이름을 카탈로그화해야 합니다.

TCP/IP 서비스 이름(svcename)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

이 매개변수에는 데이터베이스 서버가 원격 클라이언트 노드로부터 통신을 기다리는 데 사용할 TCP/IP 포트 이름이 들어 있습니다. 이 이름은 데이터베이스 관리 프로그램이 사용하도록 예약된 두 개의 연속된 포트 중 첫 번째 것이어야 합니다. 두 번째 포트는 다운 레벨 클라이언트로부터 인터럽트 요청을 처리하는 데 사용됩니다.

TCP/IP를 사용하여 데이터베이스 클라이언트로부터 연결 요청을 승인하려면 데이터베이스 서버는 반드시 서버가 지정되기로 한 포트에서 요청을 청취하고 있어야 합니다. 데이터베이스 서버의 시스템 관리자는 포트(번호 *n*)를 확보하여 서버의 서비스 파일에 관련 TCP/IP 서비스 이름을 정의해야 합니다. 데이터베이스 서버가 다운 레벨 클라이언트로부터 요청을 지원해야 하는 경우, 두 번째 포트(인터럽트 요청의 경우, *n+1*번)는 서버의 서비스 파일에 정의되어야 합니다.

데이터베이스 서버 포트(번호 *n*) 및 해당 TCP/IP 서비스 이름은 데이터베이스 클라이언트의 서비스 파일에 정의되어야 합니다. 다운 레벨 클라이언트도 인터럽트 포트(*n+1*번)를 클라이언트의 서버 파일에 정의해야 합니다.

서비스 파일의 위치는 운영 환경에 따라 다릅니다. 예를 들면, 다음과 같습니다.

- UNIX -- /etc/services
- OS/2 -- \tcpip\etc\services
- OS/2 Warp -- \mptn\etc\services.

데이터베이스 서버가 시작될 때 들어오는 연결 요청을 청취할 포트를 결정할 수 있도록 *svcename* 매개변수는 주 연결 포트와 연관된 서비스 이름으로 지정되야 합니다. 다운 레벨 클라이언트를 지원하거나 사용할 경우, 인터럽트 포트에 대한 서비스 이름은 구성 파일에 저장되지 않습니다. 인터럽트 포트 번호는 주 연결 포트 번호에 따라 결정할 수 있습니다(*인터럽트 포트 번호 = 주 연결 포트 + 1*).

데이터베이스 서버의 TCP/IP 설정에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

APPC 트랜잭션 프로그램 이름(tpname)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

이 매개변수는 APPC 통신 프로토콜 사용시 데이터베이스 클라이언트가 데이터베이스 서버에 할당 요청을 발행할 때 사용해야 하는 원격 트랜잭션 프로그램 이름을 정의합니다. 이 매개변수는 데이터베이스 서버의 구성 파일에 설정되어야 합니다.

이 매개변수는 SNA 트랜잭션 프로그램 정의에 구성되어 있는 트랜잭션 프로그램 이름과 같아야 합니다. DB2 제품의 APPC 설정에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

권장사항: 이 이름에 사용할 수 있는 문자는 다음과 같습니다.
- 영문자(A에서 Z 또는 z까지)
- 숫자(0 - 9)
- 달러 부호($), 숫자 부호(#), at 부호(@) 및 점(.)

IPX/SPX 파일 서버 이름(fileserver)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 『IPX/SPX DB2 서버 오브젝트 이름(objectname)』
- 524 페이지의 『IPX/SPX 소켓 번호(ipx_socket)』

이 매개변수는 데이터베이스 관리 프로그램의 인터네트워크 주소가 등록되어 있는 NetWare** 파일 서버 이름을 지정합니다. 데이터베이스 관리 프로그램의 인터네트워크 주소는 NetWare 파일 서버의 바인더리에 저장됩니다. 등록된 파일 서버 이름이 변경되면 서버 인스턴스에 액세스하는 모든 클라이언트는 다음을 수행해야 합니다.

- 서버 노드의 UNCATALOG
- 새 파일 서버 이름을 지정하여 서버 노드의 CATALOG

자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

IPX/SPX DB2 서버 오브젝트 이름(objectname)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 522 페이지의 『IPX/SPX 파일 서버 이름(fileserver)』
- 524 페이지의 『IPX/SPX 소켓 번호(ipx_socket)』

이 매개변수는 IPX/SPX 네트워크의 데이터베이스 관리 프로그램 인스턴스 이름을 제공합니다. NetWare 파일 서버에 등록된 각 서버 인스턴스는 고유 이름을 가져야 합니다. 이 이름이 데이터베이스 서버에서 변경되면 서버에 액세스하는 모든 클라이언트가 서버 노드를 카탈로그 해제한 다음 새 오브젝트 이름을 지정하여 다시 카탈로그화해야 합니다.

DB2 제품의 IPX/SPX 설정에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

IPX/SPX 소켓 번호(ipx_socket)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 879E [879E - 87A2] 충돌이 없도록 하기 위해 DB2가 사용하는 5개의 소켓 번호(879E에서 87A2까지)가 Novell에 고유하게 등록됩니다.

관련 매개변수

- 522 페이지의 『IPX/SPX 파일 서버 이름(fileserver)』
- 523 페이지의 『IPX/SPX DB2 서버 오브젝트 이름(objectname)』

이 매개변수는 『잘 알려진』 소켓 번호를 지정하여 DB2 서버의 인터네트워크 주소에서 연결 끝점을 나타냅니다. 소켓 번호는 제공된 머신에서 각 DB2 서버 인스턴스에 고유해야 하며, 동일한 머신에서 수행되는 모든 Novell** IPX/SPX 응용 프로그램 사이에서 고유해야 합니다. 이로써, DB2 서버가 이 소켓 번호를 사용하여 들어오는 IPX/SPX 연결을 청취할 수 있도록 보장됩니다.

DB2 제품의 IPX/SPX 설정에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

분산 서비스

다음 매개변수를 사용하여 데이터베이스 클라이언트 및 데이터베이스 서버를 구성함으로써 DCE 디렉토리 서비스를 사용할 수 있습니다.

- 『디렉토리 서비스 유형(dir_type)』
- 526 페이지의 『DCE 이름 공간의 디렉토리 경로 이름 (dir_path_name)』
- 527 페이지의 『DCE 이름 공간의 오브젝트 이름(dir_obj_name)』
- 528 페이지의 『경로지정 정보 오브젝트 이름(route_obj_name)』
- 529 페이지의 『기본 클라이언트 통신 프로토콜(dft_client_comm)』
- 530 페이지의 『기본 클라이언트 어댑터 번호(dft_client_adpt)』

DB2에서 DCE 디렉토리를 사용하는 방법에 대한 자세한 내용은 *관리 안내서: 구현*의 『DCE(Distributed Computing Environment) 디렉토리 서비스 사용』을 참조하십시오.

디렉토리 서비스 유형(dir_type)

구성 유형

데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- UNIX 및 OS/2 클라이언트
- 지역 클라이언트가 있는 UNIX 및 OS/2 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형

구성 가능

기본 [범위]

NONE [NONE; DCE]

관련 매개변수

- 527 페이지의『DCE 이름 공간의 오브젝트 이름(dir_obj_name)』
- 『DCE 이름 공간의 디렉토리 경로 이름(dir_path_name)』
- 528 페이지의『경로지정 정보 오브젝트 이름(route_obj_name)』
- 529 페이지의『기본 클라이언트 통신 프로토콜(dft_client_comm)』
- 530 페이지의『기본 클라이언트 어댑터 번호(dft_client_adpt)』

이 매개변수는 DCE 디렉토리 서비스가 사용되는지 여부를 나타냅니다.

이 매개변수가 NONE으로 설정될 경우, CONNECT 또는 ATTACH가 요청하는 지역 디렉토리 파일만이 검색됩니다. 그러나 데이터베이스 인스턴스 및 데이터베이스를 DCE 이름 공간에 기록하기 위해 *dir_path_name* 및 *dir_obj_name* 매개변수를 사용할 수 있습니다.

이 매개변수가 DCE로 설정될 경우, 이 데이터베이스 관리 프로그램 인스턴스 내에서 수행 중인 응용프로그램이 CONNECT 또는 ATTACH가 요청하는 것을 찾을 수 없을 때에는 DCE 디렉토리가 검색됩니다.

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

DCE 이름 공간의 디렉토리 경로 이름 (dir_path_name)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- UNIX 및 OS/2 클라이언트
- 지역 클라이언트가 있는 UNIX 및 OS/2 데이터베이스 서버

- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 /.:/subsys/database/

관련 매개변수

- 『DCE 이름 공간의 오브젝트 이름(dir_obj_name)』
- 525 페이지의 『디렉토리 서비스 유형(dir_type)』
- 528 페이지의 『경로지정 정보 오브젝트 이름(route_obj_name)』

전역 이름 공간의 데이터베이스 관리 프로그램 인스턴스의 고유 이름은 이 값과 *dir_obj_name* 매개변수 값으로 구성됩니다.

이 인스턴스 내에서 수행되는 모든 클라이언트 응용프로그램은 DB2DIRPATHNAME 환경 변수 값이 대체되지 않으면 이를 CONNECT 또는 ATTACH 요청의 기본 경로 이름으로 사용할 수도 있습니다.

권장사항: DCE 관리자가 제공하는 이름을 사용하십시오.

DCE 이름 공간의 오브젝트 이름(dir_obj_name)

구성 유형 데이터베이스 관리 프로그램, 데이터베이스

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- UNIX 및 OS/2 클라이언트
- 지역 클라이언트가 있는 UNIX 및 OS/2 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 525 페이지의 『디렉토리 서비스 유형(dir_type)』
- 526 페이지의 『DCE 이름 공간의 디렉토리 경로 이름 (dir_path_name)』

디렉토리에서 사용자의 데이터베이스 관리 프로그램 인스턴스(또는 사용자의 데이터베이스)를 나타내는 오브젝트 이름. 이 값과 *dir_path_name* 값을 합하면 *dir_type* 매개변수에 지정된 디렉토리 서비스가 제어하는 이름 공간의 데이터베이스 관리 프로그램 인스턴스 또는 데이터베이스를 나타내는 고유한 전역 이름이 산출됩니다.

이 매개변수는 *dir_path_name* 매개변수가 지정된 경우에만 의미가 있습니다.

구성 매개변수 *dir_path_name* 및 *dir_obj_name*의 총 길이는 255자를 넘으면 안 됩니다.

자세한 내용은 *관리 안내서: 구현*의 『DCE(Distributed Computing Environment) 디렉토리 서비스 사용』을 참조하십시오.

경로지정 정보 오브젝트 이름(route_obj_name)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 526 페이지의 『DCE 이름 공간의 디렉토리 경로 이름 (dir_path_name)』
- 525 페이지의 『디렉토리 서비스 유형(dir_type)』

이 매개변수는 DRDA 서버에 액세스하려고 시도하는 모든 클라이언트 응용프로그램이 사용할 기본 경로지정 정보 오브젝트 항목 이름을 지정합니다. 이것은 OS/2 및 UNIX 기반 환경에만 적용됩니다.

이 매개변수의 값이 /.:/나 /.../로 시작하면 값은 있는 그대로 사용됩니다. 그렇지 않으면 이 값은 *dir_path_name* 매개변수나 DB2DIRPATHNAME 환경 변수 값에 추가되어 완전한 경로지정 정보 오브젝트 이름을 형성합니다.

환경 변수 DB2ROUTE를 사용하여 이 기본값을 겹쳐쓸 수 있습니다.

이 매개변수는 *dir_type* 매개변수가 DCE로 설정될 경우에만 의미가 있습니다.

자세한 내용은 *관리 안내서: 구현*의 『DCE(Distributed Computing Environment) 디렉토리 서비스 사용』을 참조하십시오.

기본 클라이언트 통신 프로토콜(dft_client_comm)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] Null [Null; TCPIP; APPC; IPXSPX(OS/2 전용); NETBIOS(OS/2 전용)]

관련 매개변수 525 페이지의 『디렉토리 서비스 유형(dir_type)』

이 매개변수는 이 인스턴스의 클라이언트 응용프로그램이 원격 통신에 사용할 수 있는 통신 프로토콜을 나타냅니다. 내용은 하나 이상의 토큰으로 구성된 문자열입니다. 둘 이상의 토큰을 지정할 경우, 이들을 쉼표로 분리하십시오. 토큰 순서는 우선순위면에서 중요합니다.

이 매개변수는 DCE와 함께 사용해야 하며, OS/2 및 UNIX 기반 환경에만 적용됩니다.

DB2CLIENTCOMM 환경 변수를 설정하여 이 매개변수 값을 임시로 겹쳐쓸 수 있습니다.

이 매개변수 값이 NULL이고 환경 변수가 설정되어 있지 않을 경우, 서버의 전역 디렉토리 오브젝트에 지정되어 있는 첫 번째 프로토콜이 사용됩니다.

*dir_type*이 NONE으로 설정된 경우, 이 매개변수는 무시됩니다.

권장사항: 가장 자주 사용되는 프로토콜을 첫 번째로 지정해야 합니다.

기본 클라이언트 어댑터 번호(dft_client_adpt)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 0 [0-15]

관련 매개변수

- 529 페이지의 『기본 클라이언트 통신 프로토콜 (dft_client_comm)』.
- 525 페이지의 『디렉토리 서비스 유형 (dir_type)』. (dir_type이 DCE로 설정될 때)

이 매개변수는 서버 이름을 DCE CDS(Cell Directory Services)로부터 발췌한 NETBIOS 프로토콜에 대한 기본 클라이언트 어댑터 번호를 정의합니다. 이 매개변수는 OS/2 환경에만 적용됩니다.

이 매개변수는 DCE와 함께 사용해야 합니다.

DB2CLIENTADPT 환경 변수를 설정하여 이 매개변수 값을 임시로 겹쳐쓸 수 있습니다. 이 환경 변수에 숫자가 아닌 값이나 범위를 벗어난 숫자가 들어 있을 경우, 어댑터 번호는 0이 사용됩니다.

DB2 발견

다음 매개변수를 사용하여 DB2 발견을 설정하십시오.

- 『데이터베이스 발견(discover_db)』
- 『발견 모드(discover)』
- 533 페이지의 『검색 발견 통신 프로토콜(discover_comm)』
- 534 페이지의 『발견 서버 인스턴스(discover_inst)』

데이터베이스 발견(discover_db)

구성 유형 데이터베이스

매개변수 유형 구성 가능

기본 [범위] 사용 [사용 안함, 사용]

이 매개변수를 사용하여 서버에서 발견 요청이 수신될 때 데이터베이스에 대한 정보가 클라이언트로 리턴되는 것을 방지합니다.

이 매개변수의 기본값은 이 데이터베이스에 대한 발견이 사용 가능함을 나타냅니다.

이 매개변수 값을 『사용 안함』으로 변경하면 관련 데이터가 들어 있는 데이터베이스를 발견 프로세스로부터 숨길 수 있습니다. 이 작업은 데이터베이스의 기타 데이터베이스 보안 제어와 함께 수행됩니다.

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

발견 모드(discover)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] SEARCH [DISABLE, KNOWN, Search]

관련 매개변수 533 페이지의 『검색 발견 통신 프로토콜 (discover_comm)』

관리 서버 측면에서 이 구성 매개변수는 DB2ADMIN 시작시 시작되는 발견 모드의 유형을 판별합니다.

- DB2ADMIN 시작시 *discover* = SEARCH이면 *discover_comm*에 지정된 각 프로토콜에 대한 검색 발견 기능 연결 관리 프로그램이 시작됩니다. 또한 DB2COMM 레지스트리 변수에 지정된 각 프로토콜에 대한 연결 관리 프로그램이 시작됩니다. 이로써, 관리 서버는 클라이언트로부터의 검색 발견 기능 요청을 처리할 수 있습니다. SEARCH는 KNOWN 발견 기능에 의해 제공되는 큰 기능 세트를 제공합니다. *discover* = SEARCH일 경우, 관리 서버는 클라이언트로부터 알려진 발견 기능 요청과 검색 발견 기능 요청을 둘 다 처리합니다.
- DB2ADMIN 시작 시 *discover* = KNOWN이면 DB2COMM 레지스트리 변수에 지정된 연결 관리 프로그램이 시작됩니다. 이러한 연결 관리 프로그램은 KNOWN 발견 기능 요청을 처리합니다.
- DB2ADMIN 시작 시 *discover* = DISABLE이면 관리 서버는 어떤 유형의 발견 기능 요청도 처리하지 않습니다.

서버 인스턴스 측면에서 *discover* = DISABLE이면 이 서버 인스턴스에 대한 정보가 클라이언트로부터 숨겨집니다. 클라이언트가 알려진 발견 기능 요청을 이 시스템에 대해 발행할 때 관리 서버는 이 인스턴스에 대한 정보를 패키지화하지 않습니다.

클라이언트 측면에서 다음 중 하나가 발생합니다.

- *discover* = SEARCH이면 클라이언트는 검색 발견 기능 요청을 발행하여 네트워크에서 DB2 서버 시스템을 찾을 수 있습니다. 검색 발견 기능은 KNOWN 발견 기능에 의해 제공되는 큰 기능 세트를 제공합니다. *discover* = SEARCH 일 경우, 클라이언트가 알려진 발견 기능 요청과 검색 발견 기능 요청을 둘 다 발행할 수 있습니다.
- *discover* = KNOWN일 경우, 클라이언트로부터 알려진 발견 기능 요청만 발행할 수 있습니다. 특정 시스템에서 관리 서버에 대한 일부 연결 정보를 지정하면 DB2 시스템의 모든 인스턴스와 데이터베이스 정보가 클라이언트로 리턴됩니다.
- *discover* = DISABLE이면 발견 기능은 클라이언트에서 사용할 수 없습니다.

기본 발견 모드는 SEARCH입니다.

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

DB2 발견에 대한 자세한 내용은 사용자의 플랫폼에 해당하는 *빠른 시작* 매뉴얼을 참조하십시오.

검색 발견 통신 프로토콜(discover_comm)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버

- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 없음 [NETBIOS 및 TCPIP의 모든 조합]

관련 매개변수 531 페이지의 『발견 모드(discover)』

관리 서버 측면에서 이 매개변수는 DB2ADMIN 시작 시 시작되는 검색 발견 기능(discovery) 관리 프로그램을 정의합니다. 이 관리 프로그램은 클라이언트로부터 검색 발견 기능 요청에 대해 서비스를 제공합니다.

주: *discover_comm*에 정의된 프로토콜은 DB2COMM 레지스트리 변수에 지정되어야 합니다.

클라이언트 측면에서 이 매개변수는 클라이언트가 검색 발견 기능 요청을 발행하는 프로토콜을 정의합니다.

쉼표로 구분하여 둘 이상의 프로토콜을 지정하거나 매개변수를 공백으로 둘 수 있습니다.

이 매개변수에 대한 기본값은 검색 발견(discovery) 통신 프로토콜이 없음을 의미하는 "없음"입니다.

발견 서버 인스턴스(discover_inst)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] ENABLE [ENABLE, DISABLE]

이 매개변수는 인스턴스가 DB2 발견으로 검출될 수 있는지 여부를 지정합니다. 기본값인 사용은 인스턴스가 검출될 수 있도록 지정하고, 반면에 사용 안함은 인스턴스를 발견하지 못하게 합니다.

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

DB2 발견에 대한 자세한 내용은 사용자의 플랫폼에 해당하는 *빠른 시작* 매뉴얼을 참조하십시오.

파티션된 데이터베이스

다음 매개변수 그룹은 병렬 조작 및 파티션된 데이터베이스 환경에 대한 정보를 제공합니다.

- 『통신』
- 543 페이지의 『병렬 처리』.

통신

다음 매개변수는 파티션된 데이터베이스 환경에서의 통신에 대한 정보를 제공합니다.

- 536 페이지의 『연결 경과 시간(conn_elapse)』
- 536 페이지의 『FCM 메시지 앵커의 수(fcm_num_anchors)』
- 537 페이지의 『FCM 버퍼 수(fcm_num_buffers)』
- 539 페이지의 『FCM 연결 항목의 수(fcm_num_connect)』
- 539 페이지의 『FCM 요청 블록의 수(fcm_num_rqb)』
- 541 페이지의 『노드 연결 재시도(max_connretries)』
- 541 페이지의 『노드간의 최대 시차(max_time_diff)』
- 542 페이지의 『시작 및 중지 시간종료(start_stop_time)』.

연결 경과 시간(conn_elapse)

구성 유형 데이터베이스 관리 프로그램

적용 대상 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 10 [0-100]

측정 단위 초

관련 매개변수 541 페이지의 『노드 연결 재시도 (max_connretries)』

이 매개변수는 두 데이터베이스 파티션 서버간에 TCP/IP 연결을 설정할 때의 초 수를 지정합니다. 이 매개변수에 의해 지정된 시간 내에 시도가 완료될 경우, 통신이 설정됩니다. 시도가 실패할 경우, 통신을 설정하기 위한 다른 시도가 이루어집니다. 연결 시도가 *max_connretries* 매개변수에 의해 지정된 회수만큼 일어나고 시간이 항상 종료되면, 오류가 발생합니다.

FCM 메시지 앵커의 수(fcm_num_anchors)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 [-1, 128-*fcm_num_rqb*]

파티션되지 않은 데이터베이스 시스템에서 *intra_parallel* 매개변수는 이 매개변수가 사용되기 전에 사용 중이어야 합니다.

관련 매개변수

- 539 페이지의 『FCM 요청 블록의 수 (fcm_num_rqb)』
- 544 페이지의 『파티션 내 병렬 처리 작동 (intra_parallel)』

이 매개변수는 FCM *메시지 앵커*의 수를 지정합니다. 에이전트는 메시지 앵커를 사용하여 서로 메시지를 전송합니다. 기본값(-1)은 *fcm_num_rqb*에 대해 값의 75%가 지정되었음을 나타냅니다.

FCM 버퍼 수(fcm_num_buffers)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 512, 1 024 또는 4 096 [128 -- *fcm_num_rqb*]

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버: 기본값은 1 024입니다.
- 지역 클라이언트가 있는 데이터베이스 서버: 기본값은 512입니다.

- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버: 기본값은 4 096입니다.

  단일 파티션된 데이터베이스 시스템에서 *intra_parallel* 매개변수는 이 매개변수가 사용되기 전에 사용 중이어야 합니다.

이 매개변수는 데이터베이스 서버간에 그리고 데이터베이스 서버 내에서의 내부 통신(메시지)에 사용되는 4KB 버퍼 수를 지정합니다.

FCM에 대한 자세한 내용은 *관리 안내서: 구현*의 『FCM 통신 사용』을 참조하십시오.

프로세서에 다중 논리 노드가 있을 경우, 이 매개변수 값을 늘릴 필요가 있습니다. 시스템의 사용자 수, 시스템의 데이터베이스 파티션 서버 수 또는 응용프로그램의 복잡도 때문에 메시지 버퍼가 부족할 경우, 매개변수 값을 늘릴 필요가 있습니다.

다중 논리 노드를 사용하고 있는 경우, AIX가 아닌 머신에서는 같은 시스템에 있는 모든 다중 논리 노드가 하나의 *fcm_num_buffers* 버퍼 풀을 공유해야 하는 반면, AIX 시스템에서는 다음과 같습니다.

- 데이터베이스 관리 프로그램에서 사용하는 총 메모리에 충분한 공간이 있을 경우, FCM 버퍼 힙을 메모리 공간으로부터 할당하게 됩니다. 이 경우, 각 데이터베이스 파티션 서버는 자체 *fcm_num_buffers* 버퍼를 갖게 되며, 데이터베이스 파티션 서버는 FCM 버퍼의 풀을 공유하지 않습니다(이는 DB2 버전 5에서 새로 지원되는 기능임).
- 데이터베이스 관리 프로그램에서 사용하는 총 메모리에 충분한 공간이 없을 경우에는, 같은 머신의 모든 다중 논리 노드에 의해 공유되는 별도의 메모리 영역(AIX 공유 메모리 세트)에서 FCM 버퍼 힙(heap)을 할당하게 됩니다. 동일한 머신의 모든 다중 논리 노드가 하나의 *fcm_num_buffers* 풀을 공유합니다. 이는 AIX가 아닌 시스템과 AIX용 DB2 Parallel Edition 버전 1.2에서도 마찬가지입니다.

AIX의 기존 Parallel Edition 고객을 위한 권장사항: 다중 논리 노드를 사용하는 경우, Parallel Edition 버전 1.2에서 사용한 *fcm_num_buffers* 값 때문에 머

신당 사용된 저장영역보다 더 많은 저장영역이 사용됩니다. 예를 들어, 네 개의 노드 다중 논리 노드 구성은 이전보다 네 배의 FCM 버퍼를 산출합니다.

사용 중인 값을 다시 검토하십시오. 다중 논리 노드가 상주하는 머신(들)에 총 얼마만큼의 FCM 버퍼가 할당되는지 고려하십시오. 앞에서 설명한 작동을 알아보기 위해 *fcm_num_buffers*를 변경할 수 있습니다.

FCM 연결 항목의 수(fcm_num_connect)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 [-1, 128 -- *fcm_num_rqb*]

파티션되지 않은 데이터베이스 시스템에서 *intra_parallel* 매개변수는 이 매개변수가 사용되기 전에 사용 중이어야 합니다.

관련 매개변수 『FCM 요청 블록의 수(fcm_num_rqb)』

이 매개변수는 FCM 연결 항목의 수를 지정합니다. 에이전트는 연결 항목을 사용하여 서로 데이터를 전달합니다. 기본값(-1)은 *fcm_num_rqb*에 대해 값의 75%가 지정되었음을 나타냅니다.

FCM 요청 블록의 수(fcm_num_rqb)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형

구성 가능

기본 [범위]

UNIX 32비트 플랫폼

256, 512, 2 048 [128 -- 120 000]

UNIX 64비트 플랫폼

256, 512, 2 048 [128 -- 524 288]

OS/2 및 Windows NT

10 000 [250 -- 2 097 152]

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버: 기본값은 512입니다.
- 지역 클라이언트가 있는 데이터베이스 서버: 기본값은 256입니다.
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버: 기본값은 2 048입니다.

파티션되지 않은 데이터베이스 시스템에서 *intra_parallel* 매개변수는 이 매개변수가 사용되기 전에 사용 중이어야 합니다.

이 매개변수는 FCM *요청* 블록의 수를 지정합니다. 요청 블록은 FCM 디먼과 에이전트 사이에 또는 에이전트간에 정보가 전달되는 미디어입니다.

시스템의 사용자 수, 시스템의 데이터베이스 파티션 서버 수, 수행되는 조회의 복잡도에 따라 요청 블록의 요구사항이 달라집니다. 기초적으로, 기본 번호로 시작하고, 이 매개변수를 미세하게 조정할 때에는 데이터베이스 시스템 모니터의 결과를 이용하십시오.

노드 연결 재시도(max_connretries)

구성 유형	데이터베이스 관리 프로그램
적용 대상	지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	5 [0-100]
관련 매개변수	536 페이지의 『연결 경과 시간(conn_elapse)』

두 데이터베이스 파티션 서버간의 통신이 설정되지 않으면(예를 들어, *conn_elapse* 매개변수에 의해 지정된 값에 도달하면) *max_connretries*가 데이터베이스 파티션 서버에서 수행할 수 있는 연결 재시도의 수를 지정합니다. 이 매개변수에 지정된 값을 초과할 경우, 오류가 리턴됩니다.

노드간의 최대 시차(max_time_diff)

구성 유형	데이터베이스 관리 프로그램
적용 대상	지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	60 [1-1 440]
측정 단위	분

각 데이터베이스 파티션 서버는 시스템 자체의 클럭을 가지고 있습니다. 이 매개변수는 노드 구성 파일에서 나열된 데이터베이스 파티션 서버 사이에서 허용되는 최대 시차를 분으로 지정합니다.

둘 이상의 데이터베이스 파티션 서버가 트랜잭션과 연관되어 있으며 이 매개변수에 의해 지정된 시간 내에 클럭이 동기화되어 있지 않을 경우, 트랜잭션은 거부되

며 db2diag.log 파일에 경고 또는 오류 메시지가 로그됩니다. (트랜잭션은 데이터 수정과 연관되어 있을 경우에만 거부됩니다.)

DB2 Universal Database Enterprise - Extended Edition은 *세계 표준시*(UTC)를 사용하므로, 이 매개변수를 설정할 때는 다른 시간대가 문제가 되지 않습니다. 조정 표준시는 그리니치 표준시와 같습니다.

시작 및 중지 시간종료(start_stop_time)

구성 유형	데이터베이스 관리 프로그램
적용 대상	지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	10 [1 -- 1 440]
측정 단위	분

이 매개변수는 파티션된 데이터베이스 환경에만 적용됩니다. 이 매개변수는 모든 데이터베이스 파티션 서버가 DB2START 또는 DB2STOP 명령에 응답해야 하는 시간을 분 단위로 지정합니다. 또한 이 값은 ADDNODE 조작 중의 시간종료 값으로도 사용됩니다.

지정된 시간 내에 DB2START 명령에 응답하지 않는 데이터베이스 파티션 서버는 인스턴스에 대한 홈 디렉토리의 sqllib 서브디렉토리 아래에 있는 log 서브디렉토리에의 db2start 오류 로그로 메시지를 전송합니다. 재시작하기 전, 노드에 B2STOP을 발행하십시오.

지정된 시간 내에 DB2STOP 명령에 응답하지 않는 데이터베이스 파티션 서버는 인스턴스에 대한 홈 디렉토리의 sqllib 서브디렉토리 아래에 있는 log 서브디렉토리의 db2start 오류 로그로 메시지를 전송합니다. 응답하지 않는 각 데이터베이스 파티션 서버나 모든 서버에 DB2STOP을 발행할 수 있습니다. (이미 중지된 것은 중지되었다고 알리면서 리턴됩니다.)

병렬 처리

다음 매개변수는 병렬 처리에 대한 정보를 제공합니다.

- 『병렬의 처리 최대 조회 수준(max_querydegree)』
- 544 페이지의 『파티션 내 병렬 처리 작동(intra_parallel)』

병렬의 처리 최대 조회 수준(max_querydegree)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 (ANY) [ANY, 1 -- 32 767] (ANY는 시스템이 결정되었다는 의미임)

관련 매개변수

- 514 페이지의 『기본 등급(dft_degree)』
- 544 페이지의 『파티션 내 병렬 처리 작동(intra_parallel)』

이 매개변수는 데이터베이스 관리 프로그램의 이 인스턴스에서 실행되는 모든 SQL 문에 사용되는 파티션 내 병렬 처리의 최대 수준을 지정합니다. SQL문은 명령문이 실행될 때 파티션 내에서 설정된 수보다 많은 병렬 처리 조작은 수행하지 않습니다. *intra_parallel* 구성 매개변수를 『YES』로 설정해야만 데이터베이스 파티션이 작동되어 파티션 내 병렬 처리가 사용됩니다.

이 구성 매개변수의 기본값은 -1입니다. 이 값은 시스템이 최적화 프로그램에 의해 결정된 병렬 처리 수준을 사용함을 의미하며, 그렇지 않으면 사용자가 지정한 값이 사용됩니다.

주: SQL문의 병렬 처리 수준은 CURRENT DEGREE 특수 레지스터 또는 DEGREE 바인드 옵션을 사용하여 명령문을 컴파일할 때 지정될 수 있습니다.

사용 중인 응용프로그램의 파티션 내 병렬 처리의 최대 조회 수준은 SET RUNTIME DEGREE 명령을 사용하여 수정할 수 있습니다. 사용되는 실제 런타임 수준은 다음 중 가장 낮은 값입니다.

- *max_querydegree* 구성 매개변수
- 응용프로그램 런타임 수준
- SQL문 컴파일 수준

색인을 작성할 때에는 병렬 처리의 실제 조회 수준과 관련한 예외가 발생합니다. 이 경우, *intra_parallel*은 『YES』이고 테이블이 다중 프로세서를 활용할 수 있을 만큼 공간이 충분하면 색인 작성시 온라인 프로세서 수보다 하나 많은 수가 사용됩니다(최대값은 6). 위에서 언급된 기타 매개변수, 바인드 옵션 또는 특수 레지스터는 영향을 주지 않습니다.

파티션 내 병렬 처리 작동(intra_parallel)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] NO (0) [SYSTEM (-1), NO (0), YES (1)]

값 -1을 사용하면 매개변수 값은 데이터베이스 관리 프로그램이 수행 중인 하드웨어에 따라 『YES』 또는 『NO』로 설정됩니다.

관련 매개변수 543 페이지의 『병렬의 처리 최대 조회 수준 (max_querydegree)』

이 매개변수는 데이터베이스 관리 프로그램이 파티션 내 병렬 처리 기능을 사용할 수 있는지 여부를 지정합니다.

이 매개변수가 ″YES″로 설정될 때 병렬을 통한 성능 향상을 활용할 수 있는 일부 조작에는 데이터베이스 조회 및 색인 작성이 있습니다.

주: 이 매개변수 값을 변경할 경우, 패키지는 데이터베이스로 리바인드됩니다. 이 경우, 리바인드 중에 성능이 저하될 수 있습니다.

인스턴스 관리

데이터베이스 관리 프로그램 인스턴스를 관리하는 데 도움이 되는 다수의 매개변수가 제공됩니다. 이는 다음과 같은 범주로 그룹화됩니다.

- 『문제점 진단』
- 549 페이지의 『데이터베이스 시스템 모니터 매개변수』
- 550 페이지의 『시스템 관리』
- 560 페이지의 『인스턴스 관리』

문제점 진단

다음 매개변수를 사용하면 데이터베이스 관리 프로그램에서 사용 가능한 진단 정보를 제어할 수 있습니다.

- 『진단 오류 캡쳐 레벨(diaglevel)』
- 546 페이지의 『진단 데이터 디렉토리 경로(diagpath)』
- 548 페이지의 『레벨 통지(notifylevel)』

진단 오류 캡쳐 레벨(diaglevel)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 3 [0 -- 4]

관련 매개변수 『진단 데이터 디렉토리 경로(diagpath)』

db2diag.log 파일에 기록된 진단 오류 유형은 이 매개변수에 의해 판별됩니다. 유효한 값은 다음과 같습니다.

0 - 캡쳐된 진단 데이터가 없음

1 - 심각한 오류만 있음

2 - 모든 오류

3 - 모든 오류 및 경고

4 - 모든 오류, 경고 및 정보 메시지

이것은 오류 파일, 이벤트 로그 파일(Windows NT 전용), 경보 로그 파일, *diaglevel* 매개변수 값을 기초로 생성될 수 있는 덤프 파일을 포함할 디렉토리를 지정하는 데 사용되는 *diagpath* 구성 매개변수입니다.

권장사항: 문제점 해결을 돕기 위해 추가 문제점 판별 데이터를 수집하도록 이 매개변수 값을 증가시킬 수 있습니다.

진단 데이터 디렉토리 경로(diagpath)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 널(NULL)[임의의 유효 경로 이름]

관련 매개변수 545 페이지의 『진단 오류 캡쳐 레벨(diaglevel)』

이 매개변수로 완전한 DB2 진단 정보용 경로를 지정할 수 있습니다. 플랫폼에 따라 이 디렉토리에는 덤프 파일, 트랩 파일, 오류 로그 및 경보 로그 파일이 수록될 수 있습니다.

이 매개변수가 널(NULL)이면 진단 정보는 다음 디렉토리 또는 폴더 중 하나에 있는 파일에 기록됩니다.

- OS/2 및 지원되는 Windows 환경의 경우:
  - DB2INSTPROF 환경 변수 또는 키워드가 설정되어 있지 **않으면** 정보는 x:\SQLLIB\DB2INSTANCE에 기록됩니다. 여기서 x:\SQLLIB는 DB2PATH 레지스트리 변수 또는 환경 변수에 지정된 드라이브 참조 또는 디렉토리이고, DB2INSTANCE는 인스턴스 소유자의 이름입니다.

    주: 디렉토리는 SQLLIB로 이름 지정될 필요는 없습니다.
  - DB2INSTPROF 환경 변수 또는 키워드가 설정되면 정보는 x:\DB2INSTPROF\DB2INSTANCE에 기록됩니다. 여기서 DB2INSTPROF는 인스턴스 프로파일 디렉토리의 이름이고, DB2INSTANCE는 인스턴스 소유자의 이름입니다.
- UNIX 기반 환경의 경우: INSTHOME/sqllib/db2dump. 여기서 INSTHOME은 인스턴스 소유자의 홈 디렉토리입니다.
- Macintosh 환경의 경우: DB2 폴더

권장사항: 기본값을 사용하거나, 여러 인스턴스의 diagpath에 대해 중앙 위치를 보유하십시오.

파티션된 데이터베이스 환경에서는 지정된 경로가 공유 파일 시스템상에 있어야 합니다.

레벨 통지(notifylevel)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- Windows NT상의 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- Windows NT상의 클라이언트
- Windows NT상의 지역 클라이언트가 있는 데이터베이스 서버
- Windows NT상의 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- Windows 95, Windows 98 및 Windows NT상의 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 2 [0 -- 4]

이 매개변수는 파일에 기록되는 관리 통지 오류 메시지 유형을 지정합니다. 위성 노드 유형을 가진 서버의 경우, 오류는 *instance*.nfy라고 하는 통지 파일에 기록됩니다. 다른 모든 노드 유형의 경우, 이 매개변수는 Windows NT 플랫폼에서만 사용 가능하며, 오류는 Windows NT 이벤트 로그에 기록됩니다. 오류는 DB2, Capture 및 Apply 프로그램 사용자 응용프로그램에 의해 기록될 수 있습니다.

이 매개변수에 대해 유효한 값을 다음과 같습니다.

0 -- 캡쳐된 진단 데이터가 없음

1 -- 심각한 오류만 있음

2 -- 모든 오류

3 -- 모든 오류 및 경고

4 -- 모든 오류, 경고 및 정보 메시지

사용자 응용프로그램이 통지 파일 또는 Windows NT 이벤트 로그에 기록될 수 있게 하려면 db2AdminMsgWrite API를 호출해야 합니다. 이 API에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

권장사항: 문제점 해결을 돕기 위해 추가 문제점 판별 데이터를 수집하도록 이 매개변수 값을 증가시킬 수 있습니다.

데이터베이스 시스템 모니터 매개변수

다음 매개변수를 사용하면 데이터베이스 시스템 모니터의 여러 측면을 제어할 수 있습니다.

- 『기본 데이터베이스 시스템 모니터 스위치(dft_monswitches)』

기본 데이터베이스 시스템 모니터 스위치(dft_monswitches)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본값	모든 스위치가 off 상태일 때

이 매개변수를 고유하게 사용하여 각각 내부적으로 매개변수의 한 비트로 표현되는 많은 스위치를 설정할 수 있습니다. 데이터베이스 관리 프로그램 구성을 갱신하는 데 사용 중인 인터페이스에 따라 이 매개변수를 직접 갱신할 수도 있습니다. 다음 매개변수를 설정하여 이들 스위치마다 독립적으로 갱신할 수도 있습니다.

dft_mon_uow	스냅샷 모니터의 작업 단위(UOW) 스위치의 기본값
dft_mon_stmt	스냅샷 모니터의 명령문 스위치의 기본값
dft_mon_table	스냅샷 모니터의 테이블 스위치의 기본값
dft_mon_bufpool	스냅샷 모니터의 버퍼 풀 스위치의 기본값
dft_mon_lock	스냅샷 모니터의 잠금 스위치의 기본값
dft_mon_sort	스냅샷 모니터의 정렬 스위치의 기본값

이들 데이터베이스 시스템 모니터 스위치에 대한 변경사항은 즉시 적용됩니다. 즉, 데이터베이스 관리 프로그램을 중단시키고 재시작할 필요가 없습니다.

주: 기존의 모니터링 응용프로그램은 스위치의 새 기본값을 자동으로 사용하지 않습니다. 새 값(들)을 사용하려면 응용프로그램은 종료되고 인스턴스에 재접속되어야 합니다.

스냅샷 모니터 및 모니터 스위치 사용에 대한 자세한 내용은 *시스템 모니터 안내 및 참조서*를 참조하십시오.

권장사항: ON 상태인 스위치는 데이터베이스 관리 프로그램이 해당 스위치와 관련된 모니터 데이터를 수집하도록 지시합니다. 추가 모니터 데이터를 수집하여 시스템 성능 향상에 영향을 줄 수 있는 데이터베이스 관리 프로그램 오버헤드를 증가시킬 수 있습니다.

모든 모니터링 응용프로그램은 응용프로그램이 첫 번째 모니터링 요청을 발행할 때 이 기본 스위치 설정값을 계승합니다(예: 스위치 설정, 이벤트 모니터 활성화, 스냅샷 찍기). 데이터베이스 관리 프로그램이 시작될 때 데이터 수집을 시작하려는 경우에만 구성 파일의 스위치를 켜야 합니다. 그렇지 않으면 각 모니터링 응용프로그램이 자신의 스위치를 설정하여 수집한 데이터가 스위치의 설정 시간에 관련됩니다.

시스템 관리

다음 매개변수는 시스템 관리에 관련되어 있습니다.

• 551 페이지의『통신 대역폭(comm_bandwidth)』

- 552 페이지의『CPU 속도(cpuspeed)』
- 553 페이지의『동시에 사용 중인 데이터베이스의 최대수(numdb)』
- 555 페이지의『트랜잭션 프로세서 모니터 이름(tp_mon_name)』
- 557 페이지의『머신 노드 유형(nodetype)』
- 558 페이지의『기본 접미부 계정(dft_account_str)』
- 559 페이지의『JDK 1.1 설치 경로(jdk11_path)』
- 560 페이지의『연합 데이터베이스 시스템 지원(연합)』

통신 대역폭(comm_bandwidth)

구성 유형	데이터베이스 관리 프로그램
적용 대상	지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	-1 [.1 - 100 000] 값 -1을 사용하면 매개변수 값이 기본값으로 재설정됩니다. 기본값은 고속 스위치의 사용 여부에 따라 계산됩니다.
측정 단위	초당 메가바이트

SQL 최적화 알고리즘은 파티션된 데이터베이스 시스템의 데이터베이스 파티션 서버간에 특정 조작을 수행하기 위한 비용을 계산하는 데 초당 MB로 계산된 통신 대역폭의 값을 사용합니다. 최적화 알고리즘은 클라이언트와 서버간의 통신 비용을 모델링하지 않기 때문에, 이 매개변수는 데이터베이스 파티션 서버가 있는 경우 이 서버간의 근소한 대역폭만을 반영해야 합니다.

이 값을 명시적으로 설정하여 테스트 시스템에서의 제품 환경을 모델링하거나 하드웨어 업그레이드의 영향을 평가할 수 있습니다.

권장사항: 다른 환경을 모델링하려는 경우, 이 매개변수만을 조정해야 합니다.

통신 대역폭은 최적화 알고리즘이 액세스 경로를 결정하는 데 사용됩니다. 이 매개변수를 변경한 후에는 (REBIND PACKAGE 명령을 사용하여) 응용프로그램을 리바인드해야 합니다.

CPU 속도(cpuspeed)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] -1 [1^{-10} -- 1] 값 -1의 값을 사용하면 측정 프로그램의 수행에 따라 매개변수 값이 재설정됩니다.

측정 단위 초

SQL 최적화 알고리즘은 어떤 조작을 수행하는 데 드는 비용을 계산하는 데 명령어당 밀리초로 표시되는 CPU 속도를 사용합니다. 이 매개변수의 값은 CPU 속도를 측정하도록 설계된 프로그램의 출력을 근거로 데이터베이스 관리 프로그램을 설치할 때 자동으로 설정됩니다. 다음 이유로 벤치마크 결과를 사용할 수 없을 때에는 이 프로그램이 실행됩니다.

- 플랫폼에서 `db2spec.dat` 파일이 지원되지 않는 경우
- `db2spec.dat` 파일이 **없는** 경우
- IBM RISC System/6000 모델 530H 데이터가 파일에 없는 경우
- 사용하는 머신에 대한 데이터가 파일에 없는 경우

이 값을 명시적으로 설정하여 테스트 시스템에서의 제품 환경을 모델링하거나 하드웨어 업그레이드의 영향을 평가할 수 있습니다. 이 값을 -1로 설정하면, *CPU* 속도가 다시 계산됩니다.

권장사항: 다른 환경을 모델링하려는 경우, 이 매개변수만을 조정해야니다.

액세스 경로를 판별할 때 최적화 알고리즘은 CPU 속도를 사용합니다. 이 매개변수를 변경한 난 후에는 (REBIND PACKAGE 명령을 사용하여) 응용프로그램을 리바인드해야 합니다.

동시에 사용 중인 데이터베이스의 최대수(numdb)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

UNIX 8 [1 -- 256]

지역 및 원격 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버
8 [1 -- 256]

지역 클라이언트가 있는 OS/2 및 Windows NT 데이터베이스 서버 및 지역 클라이언트가 있는 위성 데이터베이스 서버
3 [1 -- 256]

측정 단위 카운터

이 매개변수는 동시에 사용 중일 수 있는(즉, 연결된 응용프로그램이 있는) 지역 데이터베이스의 수를 지정합니다. 파티션된 데이터베이스 환경에서 해당 서버가 응용프로그램에 대한 조정자(coordinator) 노드인지 여부에 상관없이 이 값은 데이터베이스 파티션 서버에서 사용 중인 최대 데이터베이스 파티션 수를 제한합니다.

각 데이터베이스가 저장영역을 차지하고 사용 중인 데이터베이스가 새 공유 메모리 세그먼트를 사용하기 때문에, 머신의 개별적인 데이터베이스의 수를 제한하여 시스템 자원 사용을 감소시킬 수 있습니다. 그러나 임의대로 데이터베이스의 수를 줄이는 것은 해답이 될 수 없습니다. 즉, 모든 데이터를 관련성과는 상관없이 하나의 데이터베이스에 놓으면 디스크 공간은 줄어들지만, 좋은 방법은 아닙니다. 기능적으로 관련된 정보를 같은 데이터베이스에 놓는 것이 일반적으로 좋은 방법입니다.

권장사항: 보통 이 값은 데이터베이스 관리 프로그램에 이미 정의된 실제 데이터베이스의 수로 설정하고, 향후 단기간(6개월 내지 1년)에 데이터베이스 수가 증가되면 계정에 합당한 증분을 추가하는 것이 가장 좋습니다. 실제로 증분이 지나치게 크지는 않지만, 이 매개변수를 자주 갱신하지 않고서도 새 데이터베이스를 추가할 수 있습니다.

numdb 매개변수를 변경하면 할당되는 메모리의 총량에 영향을 줄 수도 있습니다. 그러므로 이 매개변수를 자주 갱신하는 것은 좋지 않습니다. 이 매개변수를 갱신할 때에는 다음을 비롯한 데이터베이스나 해당 데이터베이스에 연결된 응용프로그램에 메모리를 할당할 수 있는 기타 구성 매개변수를 고려해야 합니다.

- 396 페이지의 『버퍼 풀 *크기*(buffpage)』
- 406 페이지의 『잠금 목록용 최대 저장영역(locklist)』
- 418 페이지의 『응용프로그램 힙 *크기*(applheapsz)』
- 412 페이지의 『응용프로그램 제어 힙 *크기*(app_ctl_heap_sz)』
- 414 페이지의 『정렬 힙 *크기*(sortheap)』
- 417 페이지의 『명령문 힙 *크기*(stmtheap)』
- 429 페이지의 『응용프로그램 지원 계층 힙 *크기*(aslheapsz)』
- 400 페이지의 『데이터베이스 힙(dbheap)』
- 436 페이지의 『데이터베이스 시스템 모니터 힙 *크기*(mon_heap_sz)』
- 419 페이지의 『통계 힙 *크기*(stat_heap_sz)』

트랜잭션 프로세서 모니터 이름(tp_mon_name)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 클라이언트 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본값	기본값 없음
유효한 값	• CICS • MQ • ENCINA • CB • SF • TUXEDO • TOPEND • 공백 또는 기타 값(UNIX, OS/2 및 Windows NT의 경우. Solaris 또는 SINIX의 경우에는 다른 가능한 값이 없음)

이 매개변수는 사용 중인 트랜잭션 처리(TP) 모니터 제품의 이름을 식별합니다.

- 응용프로그램이 WebSphere Enterprise Edition CICS 환경에서 수행될 경우, 이 매개변수를 『CICS』로 설정해야 합니다.
- 응용프로그램이 WebSphere Enterprise Edition Encina 환경에서 수행될 경우, 이 매개변수를 『ENCINA』로 설정해야 합니다.

- 응용프로그램이 WebSphere Enterprise Edition Component Broker 환경에서 수행될 경우, 이 매개변수를 『CB』로 설정해야 합니다.
- 응용프로그램이 IBM MQSeries 환경에서 수행될 경우, 이 매개변수를 『MQ』로 설정해야 합니다.
- 응용프로그램이 BEA Tuxedo 환경에서 수행될 경우, 이 매개변수를 『TUXEDO』로 설정해야 합니다.
- 응용프로그램이 IBM San Francisco 환경에서 수행될 경우, 이 매개변수를 『SF』로 설정해야 합니다.

IBM WebSphere EJB와 **Microsoft Transaction Server** 사용자는 이 매개변수에 대해 어떤 값도 구성할 필요가 없습니다.

위의 제품 중 어느 것도 사용하지 않을 경우, 이 매개변수는 구성하지 말고 공백으로 두어야 합니다.

OS/2 및 Windows NT 환경에 있는 DB2 Universal Database의 이전 버전에서 이 매개변수에는 XA 트랜잭션 관리 프로그램의 함수 *ax_reg* 및 *ax_unreg*를 포함하는 DLL의 경로와 이름이 들어 있었습니다. 이 형식은 계속 지원됩니다. 이 매개변수의 값이 위의 TP 모니터 이름 중 어느 것과도 일치하지 않을 경우, 그 값을 *ax_reg* 및 *ax_unreg* 함수를 포함하는 라이브러리 이름으로 가정합니다. 이는 UNIX, OS/2 및 Windows NT 환경에만 해당됩니다.

TXSeries CICS 및 Encina 사용자: OS/2 및 Windows NT상에 있는 이 제품의 이전 버전에서는 『libEncServer:C』 또는 『libEncServer:E』로 이 매개변수를 구성해야 했습니다. 이는 계속 지원되기는 하지만, 더 이상 반드시 필요한 것은 아닙니다. 매개변수를 『CICS』 또는 『ENCINA』로 구성하는 것으로 충분합니다.

MQSeries 사용자: OS/2 및 Windows NT상에 있는 이 제품의 이전 버전에서는 이 매개변수를 『mqmax』로 구성해야 했습니다. 이는 계속 지원되기는 하지만, 더 이상 반드시 필요한 것은 아닙니다. 매개변수를 『MQ』로 구성하는 것으로 충분합니다.

Component Broker 사용자: OS/2 및 Windows NT상에 있는 이 제품의 이전 버전에서는 이 매개변수를 『somtrx1i』로 구성해야 했습니다. 이는 계속 지원되기는 하지만, 더 이상 반드시 필요한 것은 아닙니다. 매개변수를 『CB』로 구성하는 것으로 충분합니다.

San Francisco 사용자: OS/2 및 Windows NT상에 있는 이 제품의 이전 버전에서는 이 매개변수를 『ibmsfDB2』로 구성해야 했습니다. 이는 계속 지원되기는 하지만, 더 이상 반드시 필요한 것은 아닙니다. 매개변수를 『SF』로 구성하는 것으로 충분합니다.

이 매개변수에 대해 지정할 수 있는 최대 문자열 길이는 19자입니다.

또한 DB2 Universal Database의 XA OPEN 문자열로 이 정보를 구성할 수도 있습니다. 여러 트랜잭션 처리 모니터에서 단일 DB2 인스턴스를 사용할 경우, 이 기능을 사용해야 합니다. XA OPEN 문자열 사용에 대한 자세한 내용은 *관리 안내서: 계획*을 참조하십시오.

머신 노드 유형(nodetype)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 정보용

이 매개변수는 머신에 설치한 DB2 제품에 대한 정보를 제공하며, 그 결과 데이터베이스 관리 프로그램 구성 유형에 대한 정보도 제공합니다. 다음은 이 매개변수 및 해당 노드 유형에 연관된 제품에 의해 리턴 가능한 값입니다.

- **지역 및 원격 클라이언트가 있는 데이터베이스 서버** - 지역 및 원격 데이터베이스 클라이언트를 지원하고 기타 원격 데이터베이스 서버에 액세스할 수 있는 DB2 서버 제품
- **클라이언트** - 원격 데이터베이스 서버에 액세스할 수 있는 데이터베이스 클라이언트
- **지역 클라이언트가 있는 데이터베이스 서버** - 지역 데이터베이스 클라이언트를 지원하고 원격 데이터베이스 서버에 액세스할 수 있는 DB2 관계형 데이터베이스 관리 시스템
- **지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버** - 지역 및 원격 데이터베이스 클라이언트를 지원하고 다른 원격 데이터베이스 서버에 액세스할 수 있으며 파티션 병렬 처리를 수행할 수 있는 DB2 서버 제품
- **지역 클라이언트가 있는 위성 데이터베이스 서버** - 지역 데이터베이스 클라이언트를 지원하고 다른 원격 데이터베이스 서버에 액세스할 수 있는 DB2 관계형 데이터베이스 관리 시스템

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

기본 접미부 계정(dft_account_str)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 널(NULL) [모든 유효한 문자열]

각 응용프로그램 연결 요청시, DB2 Connect 생성 접두부 및 사용자 제공 접미부로 구성되는 계정 식별자가 응용프로그램 요청자에서 DRDA 응용프로그램 서버로 전송됩니다. 이 계정 정보는 사용자 액세스와 자원 사용을 연관시키는 시스템 관리자에게 메커니즘을 제공합니다.

주: 이 매개변수는 DB2 Connect에서만 적용됩니다.

`sqlesact()` API를 호출하는 응용프로그램 또는 환경 변수 DB2ACCOUNT를 설정하는 사용자가 접미부를 제공합니다. API나 환경 변수가 접미부를 제공하지 않을 경우, DB2 Connect는 이 매개변수 값을 기본 접미부 값으로 사용합니다. 이 매개변수는 특히 계정 문자열을 DB2 Connect로 전달할 수 없는 다운 레벨 데이터베이스 클라이언트(버전 2 이전)에 유용합니다.

권장사항: 다음을 사용하여 이 계정 문자열을 설정하십시오.
- 영문자(A에서 Z)
- 숫자(0 - 9)
- 밑줄(_)

JDK 1.1 설치 경로(jdk11_path)

구성 유형 데이터베이스 관리 프로그램

적용 대상
- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 널(NULL) [유효한 경로]

관련 매개변수

- 441 페이지의『Java 인터프리터의 최대 힙 크기(java_heap_sz)』

이 매개변수는 JDK 1.1이 설치된 디렉토리를 지정합니다. Java 인터프리터가 사용하는 CLASSPATH 및 기타 환경 변수는 이 매개변수 값으로부터 계산됩니다.

이 매개변수에는 기본값이 없기 때문에, JDK를 설치할 때는 이 매개변수에 대한 값을 지정해야 합니다.

연합 데이터베이스 시스템 지원(연합)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] 아니오 [예; 아니오]

이 매개변수는 데이터 소스(예: DB2 계열 및 Oracle)에 의해 관리되는 데이터 분산 요청을 제출하는 응용프로그램에 대한 지원을 가능하게 하거나 불가능하게 합니다.

인스턴스 관리

다음 매개변수는 데이터베이스 관리 프로그램 인스턴스의 보안 및 관리에 관련되어 있습니다.

- 561 페이지의『시스템 관리 권한 그룹 이름(sysadm_group)』
- 562 페이지의『시스템 제어 권한 그룹 이름(sysctrl_group)』
- 563 페이지의『시스템 유지보수 권한 그룹 이름(sysmaint_group)』
- 565 페이지의『인증 유형(authentication)』
- 567 페이지의『권한 없이 허용되는 카탈로그화(catalog_noauth)』

- 568 페이지의『기본 데이터베이스 경로(dftdbpath)』
- 569 페이지의『DB2START/DB2STOP에 필요한 LOGON(ss_logon)』
- 569 페이지의『모든 클라이언트 신뢰(trust_allclnts)』
- 571 페이지의『신뢰성 있는 클라이언트 인증(trust_clntauth)』

시스템 관리 권한 그룹 이름(sysadm_group)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 562 페이지의『시스템 제어 권한 그룹 이름(sysctrl_group)』
- 563 페이지의『시스템 유지보수 권한 그룹 이름(sysmaint_group)』

시스템 관리(SYSADM) 권한은 데이터베이스 관리 프로그램 내에서 최상위 레벨의 권한으로서, 모든 데이터베이스 오브젝트를 제어합니다. 이 매개변수는 데이터베이스 관리 프로그램 인스턴스에 대한 SYSADM 권한을 사용하여 그룹 이름을 정의합니다.

SYSADM 권한은 특정 운영 체제에서 사용되는 보안 기능에 의해 판별됩니다.

- Windows 95 또는 Windows 98 운영 체제에서 SYSADM 그룹은 널(NULL)이어야 합니다.

이 매개변수는 시스템 보안이 사용된 경우 Windows 95 또는 Windows 98 클라이언트에 대해 『널(NULL)』이어야 합니다. 왜냐하면, Windows 95 또는 Windows 98 운영 체제는 그룹 정보를 저장하지 않으므로 사용자가 지정된 SYSADM 그룹의 구성원인지를 판별할 방법이 없기 때문입니다. 그룹 이름이 지정되면 어떠한 사용자도 구성원이 될 수 없습니다.

- Windows NT 및 Windows 2000 운영 체제의 경우, 이 매개변수는 이름이 8자 이하로 이루어진 지역 그룹으로 설정될 수 있으며, 이는 Windows NT 및 Windows 2000 보안 데이터베이스에 정의됩니다. 이 매개변수에 『NULL』이 지정될 경우, 관리 그룹의 모든 구성원이 SYSADM 권한을 갖습니다.
- UNIX 기반 시스템의 경우, 이 매개변수 값으로서 『NULL』이 지정되면 SYSADM 그룹의 기본값은 인스턴스 소유자의 1차 그룹이 됩니다.

  이 값이 『NULL』이 아닐 경우, 유효한 모든 UNIX 그룹 이름은 SYSADM 그룹이 될 수 있습니다.
- OS/2에서 이 매개변수에 지정된 값이 『NULL』일 경우, 사용자 프로파일 관리(UPM)에 관리자로 정의된 사용자는 SYSADM 권한을 갖습니다.

  이 매개변수에 그룹 이름이 지정될 경우, 그룹에 속하는 사용자만이 SYSADM 권한을 갖습니다. 지정된 그룹은 사용자 프로파일 관리(UPM) 그룹 중 하나일 수 있습니다.

DCE 보안이 사용되며 *sysadm_group*이 『NULL』일 경우, 기본 DCE 그룹 이름 DB2ADMIN이 사용됩니다. 권한 부여 ID 맵핑이 DB2ADMIN인 유효한 DCE 핵심부가 이미 존재해야 합니다. 다른 그룹 이름을 지정할 수 있습니다.

매개변수를 기본값(NULL)으로 복구하려면 UPDATE DBM CFG USING SYSADM_GROUP NULL을 사용하십시오. 키워드 『NULL』은 대문자로 지정해야 합니다. DB2 제어 센터의 인스턴스 구성 노트북을 사용할 수도 있습니다.

시스템 제어 권한 그룹 이름(sysctrl_group)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버

- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 561 페이지의『시스템 관리 권한 그룹 이름 (sysadm_group)』
- 『시스템 유지보수 권한 그룹 이름 (sysmaint_group)』

이 매개변수는 시스템 제어(SYSCTRL) 권한을 사용하여 그룹 이름을 정의합니다. SYSCTRL은 시스템 자원에 영향을 주는 조작을 허용하지만, 데이터에 대한 직접 액세스는 허용하지 않습니다.

주의사항: 시스템 보안이 사용되면 Windows 95 및 Windows 98 클라이언트에 대해 이 매개변수는 NULL이어야 합니다. 즉, 인증은 CLIENT, SERVER, DCS 또는 임의의 다른 유효한 인증입니다. 이는 Windows 95 및 Windows 98 운영 체제가 그룹 정보를 저장하지 않으므로 사용자가 지정된 SYSCTRL 그룹의 구성원인지를 판별할 방법이 없기 때문입니다. 그룹 이름이 지정되면 어떠한 사용자도 구성원이 될 수 없습니다. 그러나 DCE 인증이 사용될 때에는 해당되지 않습니다. 이 경우에는 그룹 이름을 지정할 수 있습니다.

매개변수를 기본값(NULL)으로 복구하려면 UPDATE DBM CFG USING SYSCTRL_GROUP NULL을 사용하십시오. 키워드『NULL』은 대문자로 지정해야 합니다. DB2 제어 센터의 인스턴스 구성 노트북을 사용할 수도 있습니다.

시스템 유지보수 권한 그룹 이름(sysmaint_group)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본값 널(NULL)

관련 매개변수

- 561 페이지의 『시스템 관리 권한 그룹 이름 (sysadm_group)』
- 562 페이지의 『시스템 제어 권한 그룹 이름 (sysctrl_group)』

이 매개변수는 시스템 유지보수(SYSMAINT) 권한을 사용하여 그룹 이름을 정의합니다. SYSMAINT는 데이터에 직접 액세스하지 않고 인스턴스와 연관된 모든 데이터베이스에 유지보수 조작을 수행할 특권을 갖습니다.

주의사항: 시스템 보안이 사용되면 Windows 95 및 Windows 98 클라이언트에 대해 이 매개변수는 NULL이어야 합니다. 즉, 인증은 CLIENT, SERVER, DCS 또는 임의의 다른 유효한 인증입니다. 이는 Windows 95 및 Windows 98 운영 체제가 그룹 정보를 저장하지 않으므로 사용자가 지정된 SYSMAINT 그룹의 구성원인지를 판별할 방법이 없기 때문입니다. 그룹 이름이 지정되면 어떠한 사용자도 구성원이 될 수 없습니다. 그러나 DCE 인증이 사용될 때에는 해당되지 않습니다. 이 경우에는 그룹 이름을 지정할 수 있습니다.

매개변수를 기본값(NULL)으로 복구하려면 UPDATE DBM CFG USING SYSCTRL_GROUP NULL을 사용하십시오. 키워드 『NULL』은 대문자로 지정해야 합니다. DB2 제어 센터의 인스턴스 구성 노트북을 사용할 수도 있습니다.

인증 유형(authentication)

구성 유형	데이터베이스 관리 프로그램
적용 대상	• 지역 및 원격 클라이언트가 있는 데이터베이스 서버 • 클라이언트 • 지역 클라이언트가 있는 데이터베이스 서버 • 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버 • 지역 클라이언트가 있는 위성 데이터베이스 서버
매개변수 유형	구성 가능
기본 [범위]	SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DCS; DCS_ENCRYPT; DCE; DCE_SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT]

이 매개변수는 사용자의 인증이 어디에서 어떻게 수행되는지를 판별합니다.

인증이 SERVER이면 사용자 ID와 암호가 클라이언트에서 서버로 전송되어 인증이 서버에서 수행될 수 있습니다. SERVER_ENCRYPT 값은 네트워크를 통해 전송되는 암호가 암호화된다는 것을 제외하고는 SERVER와 동일한 작동을 제공합니다.

CLIENT 값은 모든 인증이 클라이언트에서 발생하므로 서버에서는 인증을 수행할 필요가 없음을 나타냅니다.

DCS 값은 호스트나 AS/400 시스템에서 인증이 발생함을 나타냅니다. DCS_ENCRYPT 값은 네트워크를 통해 전송되는 암호가 암호화된다는 것을 제외하고는 DCS와 동일한 작동을 제공합니다. 클라이언트의 암호를 DB2 서버에 표시하지 않는 APPC 및 통신 제품을 사용할 경우, DCS를 지정하여 다음을 확보할 수 있습니다.

- 비 DRDA 클라이언트에 대한 SERVER 유형 인증
- DRDA 클라이언트에 대한 CLIENT 유형 인증

DCE 값은 DCE 보안 서비스를 사용하여 DCE 서버에서 인증이 수행됨을 의미합니다. DCE_SERVER_ENCRYPT 값은 네트워크를 통해 전송되는 암호가 암호화된다는 것을 제외하고는 DCE와 동일한 작동을 제공합니다.
DCE_SERVER_ENCRYPT 값은 서버에서만 사용됩니다. 이 값은 서버가 DCE 인증 또는 SERVER_ENCRYPT 인증을 승인할 수 있음을 나타냅니다.

KERBEROS 값은 인증에 대해 Kerberos 보안 프로토콜을 사용하는 Kerberos 서버에서 인증이 수행됨을 의미합니다. Kerberos 보안 시스템을 지원하는 서버 및 클라이언트에서 KRB_SERVER_ENCRYPT 유형의 인증을 사용할 경우, 효율적인 시스템 인증 유형은 KERBEROS입니다. 클라이언트에서 Kerberos 보안 시스템을 지원하지 않을 경우, 효율적인 시스템 인증 유형은 SERVER_ENCRYPT와 같습니다.

주: Kerberos 인증 유형은 Windows 2000을 수행하는 서버에서만 지원됩니다.

암호화를 지원하는 인증 값으로는 SERVER_ENCRYPT, DCS_ENCRYPT, DCE_SERVER_ENCRYPT, KRB_SERVER_ENCRYPT가 있습니다. 소스에 카탈로그화되어 있는 인증 유형에 지정된 대로, 이동하는 모든 암호는 소스에서 암호화되며 목표에서는 암호 해독이 필요하다는 것을 제외하고, 이 값은 인증 위치 관점에서 SERVER, DCS, DCE 및 KERBEROS와 동일한 기능을 제공합니다. 인증 위치와 일치하는 암호화 및 비 암호화 값은 인증이 발생하는 위치에 관계없이 클라이언트와 게이트웨이 또는 게이트웨이와 서버간에 다른 암호 인증 조합을 선택하는 데 사용될 수 있습니다.

이 값에 해당하는 숫자 및 API 상수에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

DCE 또는 DCS를 사용하는 시기와 이유, 연합 데이터베이스에 관련된 인증 문제에 대한 자세한 내용은 *관리 안내서: 구현*의 『데이터베이스 액세스 제어』 장을 참조하십시오.

권장사항: 일반적으로 기본값(SERVER)을 사용하는 것이 좋습니다. Kerberos, DB2 Connect 또는 DCE에 의해 들어오는 요청이 있을 경우에는 *관리 안내서: 구현*의 『데이터베이스 액세스 제어』를 참조하십시오.

권한 없이 허용되는 카탈로그화(catalog_noauth)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 클라이언트
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

지역 및 원격 클라이언트가 있는 데이터베이스 서버; 지역 및 원격 클라이언트가 있는 데이터베이스 서버

NO [NO (0) -- YES (1)]

클라이언트; 지역 클라이언트가 있는 데이터베이스 서버; 지역 클라이언트가 있는 위성 데이터베이스 서버

YES [NO (0) -- YES (1)]

이 매개변수는 사용자가 SYSADM 권한 없이 데이터베이스와 노드 또는 DCS와 ODBC 디렉토리를 카탈로그화하고 카탈로그 해제할 수 있는지를 지정합니다. 이 매개변수의 기본값(0)은 SYSADM 권한이 필요함을 나타냅니다. 이 매개변수를 1(예)로 설정하면 SYSADM 권한이 필요하지 않습니다.

기본 데이터베이스 경로(dftdbpath)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버
- 지역 클라이언트가 있는 위성 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위]

UNIX 인스턴스 소유자의 홈 디렉토리 [기존의 모든 경로]

OS/2 및 Windows NT
DB2가 설치된 드라이브 [기존의 모든 경로]

이 매개변수에는 데이터베이스 관리 프로그램하에서 데이터베이스를 작성하는 데 사용하는 기본 파일 경로가 들어 있습니다. 데이터베이스 작성시 경로를 지정하지 않으면 *dftdbpath* 매개변수가 지정하는 경로하에 데이터베이스가 작성됩니다.

파티션된 데이터베이스 환경에서 데이터베이스가 작성되고 있는 경로가 NFS 마운트 경로인지(UNIX 기반 플랫폼에서), 아니면 네트워크 드라이브(Windows NT에서)인지 확인하십시오. 지정된 경로는 각 데이터베이스 파티션 서버에 물리적으로 존재해 있어야 합니다. 혼동을 피하려면 각 데이터베이스 파티션 서버에 지역적으로 마운트된 경로를 지정하는 것이 가장 좋습니다. 최대 경로 길이는 205자입니다. 시스템이 경로의 끝에 노드 이름을 첨부합니다.

데이터베이스가 더 커질 수 있고 많은 사용자가 데이터베이스를 작성(사용자 환경과 목적에 따라)할 수 있도록 제공된 경우, 모든 데이터베이스를 지정된 위치에 작

성하여 저장하는 것이 때로 편리할 수 있습니다. 또한 데이터베이스를 무결성과 백업 및 복구 용이성의 이유로 다른 응용프로그램 및 데이터로부터 분리할 수 있도록 하는 것도 좋습니다.

UNIX 기반 환경의 경우 *dftdbpath* 이름의 길이는 215자를 초과할 수 없으며, 유효한 절대 경로 이름이어야 합니다. OS/2 및 Windows NT의 경우 *dftdbpath*는 드라이브 이름이 될 수 있으며, 뒤에 콜론을 쓸 수도 있습니다.

권장사항: 가능하면 볼륨이 큰 데이터베이스는 운영 체제 파일 및 데이터베이스 로그와 같이 자주 액세스하는 데이터가 있는 디스크가 아닌 다른 디스크에 넣어두십시오.

DB2START/DB2STOP에 필요한 LOGON(ss_logon)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] YES [NO (0), YES (1)]

이 매개변수는 OS/2 환경에만 적용됩니다. 이 매개변수의 기본값을 승인하면 DB2START 또는 DB2STOP을 발행하기 전에 LOGON 사용자 ID 및 암호를 사용해야 합니다.

모든 클라이언트 신뢰(trust_allclnts)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] YES [NO, YES, DRDAONLY]

관련 매개변수

- 565 페이지의 『인증 유형(authentication)』
- 571 페이지의 『신뢰성 있는 클라이언트 인증(trust_clntauth)』

이 매개변수는 *authentication* 매개변수가 CLIENT로 설정될 경우에만 활성화됩니다.

이 매개변수와 *trust_clntauth*는 사용자가 데이터베이스 환경에서 유효성이 확인되는 위치를 판별하는 데 사용됩니다.

이 매개변수에 대해 기본값 『YES』를 승인하면 모든 클라이언트가 신뢰성 있는 클라이언트로 간주됩니다. 즉, 서버는 클라이언트에서 보안 레벨을 사용할 수 있으며 사용자는 클라이언트에서 유효성이 확인될 수 있다고 가정합니다.

이 매개변수는 *authentication* 매개변수가 CLIENT로 설정될 경우에만 『NO』로 변경될 수 있습니다. 이 매개변수가 『NO』로 설정될 경우, 신뢰성 없는 클라이언트는 서버에 연결할 때 사용자 ID와 암호 조합을 제공해야 합니다. 신뢰성 없는 클라이언트란 사용자를 인증할 보안 서브시스템을 가지고 있지 않은 운영 체제 플랫폼을 의미합니다.

이 매개변수를 『DRDAONLY』로 설정하면 MVS용 OS/390용 DB2, VM용 DB2, VSE용 DB2 및 OS/400용 DB2의 DRDA 클라이언트를 제외한 모든 클라이언트를 보호합니다. 이 클라이언트가 클라이언트측 인증을 수행해야만 신뢰할 수 있습니다. 다른 모든 클라이언트는 서버에 의해 인증되려면 사용자 ID와 암호를 제공해야 합니다.

*trust_allclnts*를 『DRDAONLY』로 설정하면 *trust_clntauth* 매개변수는 클라이언트가 인증되는 위치를 판별하는 데 사용됩니다. *trust_clntauth*를 『CLIENT』로 설정하면 인증은 클라이언트에서 발생합니다. *trust_clntauth*를 『SERVER』로 설정하면 암호를 제공하지 않을 경우에 인증은 클라이언트에서 발생하고, 암호를 제공할 경우에는 서버에서 발생합니다.

신뢰성 있는 클라이언트에 대한 자세한 내용은 *관리 안내서: 구현*의 『서버에 따른 인증 방법 선택』을 참조하십시오.

신뢰성 있는 클라이언트 인증(trust_clntauth)

구성 유형 데이터베이스 관리 프로그램

적용 대상

- 지역 및 원격 클라이언트가 있는 데이터베이스 서버
- 지역 클라이언트가 있는 데이터베이스 서버
- 지역 및 원격 클라이언트가 있는 파티션된 데이터베이스 서버

매개변수 유형 구성 가능

기본 [범위] CLIENT [CLIENT, SERVER]

관련 매개변수

- 565 페이지의 『인증 유형(authentication)』
- 569 페이지의 『모든 클라이언트 신뢰(trust_allclnts)』

이 매개변수는 클라이언트가 연결을 위해 사용자 ID 및 암호의 조합을 제공할 때 서버 또는 클라이언트에서 신뢰성 있는 클라이언트가 인증되는지 여부를 지정합니다. 이 매개변수(및 *trust_allclnts*)는 *authentication* 매개변수가 CLIENT로 설정될 때에만 활성화됩니다. 사용자 ID 및 암호가 제공되지 않을 경우, 사용자가 클라이언트에서 유효성이 확인된 것으로 서버에서 더 이상 유효성 확인이 수행되지 않습니다.

이 매개변수가 CLIENT(기본값)로 설정될 경우, 신뢰성 있는 클라이언트는 사용자 ID와 암호 조합을 제공하지 않고 연결할 수 있으며, 운영 체제가 이미 사용자를 인증한 것으로 가정합니다. 이 값이 SERVER로 설정될 경우, 사용자 ID 및 암호는 서버에서 유효성이 확인됩니다.

CLIENT의 숫자 값은 0입니다. SERVER의 숫자 값은 1입니다.

신뢰성 있는 클라이언트에 대한 자세한 내용은 *관리 안내서: 구현*의 『서버에 따른 인증 방법 선택』을 참조하십시오.

제4부 부록 및 끝머리

© Copyright IBM Corp. 1993, 2001

부록A. DB2 레지스트리 및 환경 변수

다음은 조직화하여 수행하는 데 필요할 수도 있는 환경 변수와 DB2 레지스트리 변수에 대한 목록입니다. 각각 간단한 설명이 있는데, 사용자의 환경에 적용되지 않는 것도 있습니다.

다음을 사용하여 지원되는 모든 레지스트리 변수 목록을 볼 수 있습니다.

```
db2set -lr
```

다음을 사용하여 현재 또는 기본 인스턴스에 있는 변수 값을 변경할 수 있습니다.

```
db2set registry_variable_name=new_value
```

환경 변수를 갱신하려면 set 명령을 사용한 다음 시스템을 재부트해야 합니다.

변경된 레지스트리 변수 값은 DB2START 명령이 발행되기 전에 설정해야 합니다. 레지스트리 변수 변경 및 사용에 대한 자세한 내용은 *관리 안내서: 구현*을 참조하십시오.

2진 레지스트리 변수의 설명에 사용된 값에는 동등한 값이 있습니다. 값 YES, 1 및 ON은 모두 동등한 값입니다. 마찬가지로, 값 NO, 0 및 OFF도 모두 동등한 값입니다. 동등한 값은 사용시 서로 교환할 수 있습니다.

표 21. 일반 레지스트리 변수

변수 이름 설명	운영 체제	값
DB2ACCOUNT	모두	기본값=널(NULL)
원격 호스트에 전송된 계정 문자열. 자세한 내용은 *DB2 Connect 사용자 안내서*를 참조하십시오.		
DB2BIDI	모두	기본값=NO 값: YES 또는 NO
이 변수는 양방향 지원을 가능하게 하며, DB2CODEPAGE 변수는 사용될 코드 페이지를 선언하는 데 사용됩니다. 양방향 지원에 대한 자세한 내용은 자국어 지원 부록의 *관리 안내서: 계획*을 참조하십시오.		

© Copyright IBM Corp. 1993, 2001

표 21. 일반 레지스트리 변수 (계속)

변수 이름 설명	운영 체제	값
DB2_BLOCK_ON_LOG_DISK_FULL 이 DB2 레지스트리 변수는 사용 중인 로그 경로에서 DB2가 새 로그 파일을 생성할 수 없을 때 생성되는 "디스크 공간 없음" 이라는 오류를 방지하기 위해 설정할 수 있습니다. 또한 DB2는 로그 파일이 작성될 때까지 5분마다 로그 파일 작성을 시도합니다. 각각의 시도 이후에 DB2는 db2diag.log 파일에 메시지를 기록합니다. db2diag.log 파일을 모니터하는 것은 으로 로그 디스크 공간 없음 상태로 인해 응용프로그램이 정지되었는지 확인할 수 있는 유일한 메소드입니다. 로그 파일이 작성될 때까지 테이블 데이터를 갱신하려고 하는 모든 사용자 응용프로그램은 트랜잭션을 확약할 수 없습니다. 읽기 전용 조회에 직접 영향을 미칠 수도 있습니다. 갱신 요청에 의해 잠긴 데이터에 조회가 액세스할 필요가 있거나, 갱신 응용프로그램이 버퍼 풀의 데이터 페이지를 수행한 경우, 읽기 전용 조회가 정지된 것처럼 보이게 됩니다.	ALL	기본값=아니오 값: 예 또는 아니오
DB2CODEPAGE 데이터베이스 클라이언트 응용프로그램용으로 DB2 표시된 데이터의 코드 페이지를 지정합니다. DB2 문서에 언급되지 않았거나 DB2 서비스로 언급되기를 요청하지 않으면 DB2CODEPAGE를 설정하지 마십시오. 운영 체제로 지원되지 않은 값에 DB2CODEPAGE를 설정하면 예기치 않은 결과가 생길 수 있습니다. DB2가 자동으로 코드 페이지 정보를 운영 체제에서 얻기 때문에 정상적인 경우에는 DB2CODEPAGE를 설정할 필요가 없습니다.	모두	기본값: 운영 체제에서 지정된 대로 언어 ID에서 획득
DB2COUNTRY 날짜 및 시간 형식에 영향을 줄 수 있는 클라이언트 응용프로그램의 국가, 지역 또는 지역 코드를 지정합니다.	모두	기본값: 운영 체제에서 지정된 대로 언어 ID에서 획득
DB2DBDFT 내재된 연결에 사용될 데이터베이스의 데이터베이스 별명을 지정합니다. 응용프로그램에 데이터베이스 연결이 없는데 SQL문이 발행되면, DB2DBDFT 환경 변수에 기본 데이터베이스가 정의되어 있는 경우 내재된 연결이 이루어집니다.	모두	기본값=널(NULL)
DB2DBMSADDR 16진 형식의 기본 데이터베이스 관리 프로그램 공유 메모리 주소를 지정합니다. 공유 메모리 충돌로 인해 *db2start*가 실패한 경우, 데이터베이스 관리 프로그램 인스턴스가 자신의 공유 메모리를 다른 주소로 할당하도록 이 레지스트리 변수를 수정할 수 있습니다.	Windows 32비트 운영 체제	기본값=0x20000000(Windows NT의 경우), 0x90000000(Windows 95의 경우) 값: 0x20000000 - 0xB0000000(0x10000씩 증가)
DB2_DISABLE_FLUSH_LOG	모두	Default= OFF 값: ON 또는 OFF

표 21. 일반 레지스트리 변수 (계속)

변수 이름	운영 체제	값
설명		
온라인 백업이 완료되었을 때 사용 중인 로그 파일을 닫지 않도록 하는지를 지정합니다. 온라인 백업이 완료되면 마지막 사용 중인 로그 파일이 절단되고 닫혀지며 아카이브되도록 사용 가능해집니다. 이것은 온라인 백업이 복구에 사용할 수 있는 전체 아카이브 로그 세트를 가지도록 합니다. 로그 순차 번호(LSN) 공간의 부분을 소모하고 있는 것이 염려되면 마지막 사용 중인 로그 파일을 닫지 않을 수도 있습니다. 사용 중인 로그 파일이 절단될 때마다 LSN은 절단되는 공간에 비례하는 양만큼 증가됩니다. 매일 아주 많은 온라인 백업을 수행할 경우, 마지막 사용 중인 로그 파일을 닫지 않을 수도 있습니다. 온라인 백업 완료 직후에 로그가 가득 찼다는 메시지를 수신할 경우에도 마지막 사용 중인 로그 파일을 닫지 않을 수 있습니다. 로그 파일이 절단되면 절단된 로그의 크기에 비례하는 양만큼 예약된 사용 중인 로그 공간이 증가됩니다. 사용 중인 로그 공간은 절단된 로그 파일이 재생되면 사용할 수 있게 됩니다. 로그 파일이 비활동 상태가 되고 나면 곧장 재생이 발생합니다. 로그가 가득 찼다는 메시지(Log full message)를 수신할 수도 있는 짧은 간격 동안입니다.		
DB2DISCOVERYTIME	OS/2 및 Windows 32비트 운영 체제	기본값=40초, 최소값=20초
DB2 시스템을 SEARCH 발견 기능으로 검색하는 시간을 지정합니다.		
DB2INCLUDE	모두	기본값=현재 디렉토리
DB2 PREP 처리 동안 SQL INCLUDE 텍스트 파일문을 처리할 때 사용될 경로를 지정합니다. 이렇게 하여 INCLUDE 파일이 발견될 수도 있는 디렉토리의 목록을 제공합니다. 사전 컴파일된 언어에서 DB2INCLUDE를 사용하는 방법에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.		
DB2INSTDEF	OS/2 및 Windows 32비트 운영 체제	기본값=DB2
DB2INSTANCE가 정의되지 않은 경우에 사용될 값을 설정합니다.		
DB2INSTOWNER	Windows NT	기본값=널(NULL)
인스턴스가 처음 작성될 때 DB2 프로파일 레지스트리에 작성된 레지스트리 변수. 이 변수는 인스턴스 소유 머신의 이름으로 설정됩니다.		
DB2_LIC_STAT_SIZE	모두	기본값=널(NULL) 범위: 0 - 32 767
레지스트리 변수는 시스템의 사용권 통계를 포함하는 파일의 최대 크기(MB 단위)를 결정하는 데 사용됩니다. 값이 0이면 사용권 통계를 수집하지 않는 것(off)입니다. 변수가 인식되지 않거나 정의되지 않으면 변수는 기본값으로 무제한 설정됩니다. 통계는 사용권 센터를 사용하여 표시됩니다.		
DB2NBDISCOVERRCVBUFS	모두	기본값=16개의 버퍼 최대값=16개의 버퍼

표 21. 일반 레지스트리 변수 (계속)

변수 이름	운영 체제	값
설명		
이 변수는 NetBIOS 검색 발견 기능에 사용됩니다. 변수는 클라이언트에 의해 수신될 수 있는 동시 발견 기능 응답 수를 지정합니다. 클라이언트가 이 변수에 지정된 수보다 많은 동시 응답을 수신하면 NetBIOS 계층에 의해 초과 응답이 버려집니다. 기본값은 16개의 NetBIOS 수신 버퍼입니다. 기본값보다 작은 값을 선택하면 기본값이 사용됩니다.		
DB2OPTIONS	Windows 3.1 및 Macintosh를 제외한 모든 운영 체제	기본값=널(NULL)
명령행 처리기 옵션을 설정합니다.		
DB2SLOGON	Windows 3.x	기본값=널(NULL), 값: YES 또는 NO
Windows 3.x용 DB2에서 보안 로그인을 가능하게 합니다. DB2SLOGON=YES일 경우, DB2는 사용자 ID와 암호를 파일에는 기록하지 않으나, 대신 메모리 세그먼트를 사용하여 이를 유지보수합니다. DB2SLOGON이 작동 가능하면 Windows 3.x가 시작될 때마다 로그온해야 합니다.		
DB2TIMEOUT	Windows 3.x 및 Macintosh	기본값=(설정되지 않음)
긴 SQL 조회 동안 Windows 3.x와 Macintosh 클라이언트에 대한 시간종료 기간을 제어하는 데 사용합니다. 조회가 인터럽트되거나 계속될 수 있는 경우, 시간종료 기간은 대화 상자 팝업을 만기합니다. 변수의 최소값은 30초입니다. DB2TIMEOUT이 1 - 30으로 설정된 경우, 최소 기본값이 사용됩니다. DB2TIMEOUT이 0이나 음의 값으로 설정된 경우, 시간종료 기능을 사용할 수 없습니다. 시간종료 기능은 기본적으로는 사용할 수 없습니다.		
DB2TRACENAME	Windows 3.x 및 Macintosh	기본값=DB2WIN.TRC(Windows 3.x에서), DB2MAC.TRC(Macintosh에서)
Windows 3.x 및 Macintosh에서 추적 정보가 저장되는 파일의 이름을 지정합니다. 각 시스템의 기본값은 현재 인스턴스 디렉토리(예: \sqllib\db2)에 저장됩니다. 추적 파일 이름을 지정할 때 전체 경로 이름을 지정하는 것이 가장 좋습니다.		
DB2TRACEON	Windows 3.x 및 Macintosh	기본값=NO 값: YES 또는 NO
Windows 3.x 및 Macintosh에서 문제점이 발생할 때를 대비해 IBM에 정보를 제공할 수 있도록 추적을 설정(on)합니다. (해결할 수 없는 문제점이 발생하지 않으면 추적을 설정 하는 것은 바람직하지 않습니다. 클라이언트에서의 추적 기능 사용에 대한 자세한 내용은 *문제점 해결 안내서*를 참조하십시오.		
DB2TRCFLUSH	Windows 3.x 및 Macintosh	기본값=NO 값: YES 또는 NO

표 21. 일반 레지스트리 변수 (계속)

변수 이름 설명	운영 체제	값
Windows 3.x 및 Macintosh에서 DB2TRACEFLUSH는 DB2TRACEON=YES와 함께 사용할 수 있습니다. DB2TRACEFLUSH=YES로 설정하면 각 추적 레코드가 추적 파일에 즉시 기록됩니다. 이렇게 하면 DB2 시스템이 많이 느려져 기본값은 DB2TRACEFLUSH=NO로 설정됩니다. 이 설정은 응용프로그램이 시스템을 정지시켜 재부트해야 할 때 유용합니다. 키워드를 이렇게 설정하여 추적 파일과 추적 항목을 재부트시 확실하게 유실되지 않게 합니다.		
DB2TRCSYSERR	Windows 3.x 및 Macintosh	기본값=1 값: 1 - 32 767
클라이언트가 추적을 멈추기 전에 추적에 시스템 오류 번호를 지정합니다. 기본값은 추적이 멈춘 후의 시스템 오류를 추적합니다.		
DB2YIELD	Windows 3.x	기본값=NO 값: YES 또는 NO
원격 서버와 통신하는 동안 Windows 3.x 클라이언트의 작동을 지정합니다. NO로 설정되면 클라이언트는 CPU를 다른 Windows 3.x 응용프로그램에 양도하지 못하게 되고, Windows 환경은 클라이언트 응용프로그램이 원격 서버와 통신하는 동안 정지됩니다. 다른 타스크를 재개하기 전에 통신 조작이 완료될 때까지 기다려야 합니다. YES로 설정되면 시스템은 정상으로 기능합니다. DB2YIELD=YES로 응용프로그램을 수행하는 것이 바람직합니다. 시스템이 파괴될 경우, DB2YIELD=NO로 설정해야 합니다. 응용프로그램을 개발하기 위해 통신 조작이 완료되기를 기다리는 동안 Windows 메시지 승인과 처리에 응용프로그램이 기록되었는지 확인하십시오.		

표 22. 시스템 환경 변수

변수 이름 설명	운영 체제	값
DB2CONNECT_IN_APP_PROCESS	모두	기본값=YES 값: YES 또는 NO
이 변수를 NO로 설정하면 DB2 Connect Enterprise Edition의 지역 DB2 Connect 클라이언트는 강제로 에이전트 내에서 수행되게 됩니다. 에이전트 내에서 수행될 때의 장점은 지역 클라이언트를 모니터할 수 있으며 SYSPLEX 지원을 사용할 수 있다는 점입니다.		
DB2DOMAINLIST	Windows NT 서버 전용	기본값=널(NULL) 값: Windows NT 도메인 이름의 목록은 쉼표(『,』)로 구분됩니다.
하나 이상의 Windows NT 도메인을 정의합니다. 이 도메인에 속한 사용자만 연결 또는 접속 요청을 승인합니다. 이 레지스트리 변수는 오직 DB2 서버와 함께 Windows NT 도메인 환경, 그리고 DB2 Universal Database 버전 7.1(또는 그 이후 버전)을 실행하는 클라이언트에서만 사용됩니다.		
DB2ENVLIST	UNIX	기본값: 널(NULL)

표 22. 시스템 환경 변수 (계속)

변수 이름	운영 체제	값
설명		
저장 프로시듀어 또는 사용자 정의 함수(UDF)에 대해 특정 변수 이름을 나열합니다. 기본적으로 **db2start** 명령은 접두부가 **DB2** 또는 **db2**인 변수를 제외한 모든 사용자 환경 변수를 필터링합니다. 특정 레지스트리 변수를 저장 프로시듀어 또는 사용자 정의 함수(UDF)로 전달해야 할 경우, DB2ENVLIST 레지스트리 변수에 변수 이름을 나열할 수 있습니다. 각 변수 이름을 하나 이상의 공간으로 구분하십시오. DB2는 고유한 PATH 및 LIBPATH를 구성하므로, PATH 또는 LIBPATH를 DB2ENVLIST에 지정하면 변수 이름의 실제 값이 DB2가 구성하는 값의 끝에 추가됩니다.		
DB2INSTANCE	모두	OS/2 및 Windows 32비트 운영 체제에서 기본값=DB2INSTDEF
기본적으로 사용 중인 인스턴스를 지정할 때 사용되는 환경 변수. UNIX에서 사용자는 DB2INSTANCE에 대한 값을 지정해야 합니다.		
DB2INSTPROF	OS/2, Windows 3.x 및 Windows 32비트 운영 체제	기본값: 널(NULL)
DB2PATH와 다를 경우, OS/2, Windows 3.x 및 Window 32비트 운영 체제의 인스턴스 디렉토리 위치를 지정할 때 사용되는 환경 변수.		
DB2LIBPATH	UNIX	기본값: 널(NULL)
DB2LIBPATH 레지스트리 변수에 LIBPATH 값을 지정합니다. LIBPATH 값은 사용자 ID가 변경된 경우, 상위 및 하위 프로세스 사이에 계승될 수 없습니다. **db2start** 실행 파일은 루트 소유이기 때문에, DB2는 일반 사용자의 LIBPATH 설정을 계승할 수 없습니다. DB2ENVLIST 레지스트리 변수에 변수 이름 LIBPATH를 나열할 경우, DB2LIBPATH 레지스트리 변수에 LIBPATH 값도 지정해야 합니다. **db2start** 실행 파일은 DB2LIBPATH 값을 읽어 이 값을 DB2가 구성하는 LIBPATH의 끝에 추가합니다.		
DB2NODE	모두	기본값: 널(NULL) 값: 1 - 999
접속하려는 DB2 Extended Enterprise Edition 데이터베이스 파티션 서버의 목표 논리 노드를 지정하는 데 사용됩니다. 이 변수를 설정하지 않으면 목표 논리 노드의 기본값은 머신에서 포트 0에 정의된 논리 노드로 설정됩니다.		
DB2_PARALLEL_IO	모두	기본값: 널(NULL) 값: * (모든 테이블 공간을 의미함) 또는 하나로 정의된 테이블 공간 이외의 쉼표로 구분된 목록
테이블 공간 컨테이너에서 데이터를 읽거나 또는 쓰는 경우, DB2는 데이터베이스에 있는 컨테이너의 수가 1보다 클 경우에 병렬 I/O를 사용할 수 있습니다. 단일 컨테이너에 대해 병렬 I/O를 강제로 실행할 경우, 이 레지스트리 변수를 사용하십시오. 레지스트리 변수를 설정한 후에 DB2STOP을 발행하고 DB2START를 입력하여 변경사항이 적용되도록 하십시오.		

표 22. 시스템 환경 변수 (계속)

변수 이름 설명	운영 체제	값
DB2PATH	OS/2, Windows 3.x 및 Windows 32비트 운영 체제	기본값: (운영 체제에 따라 다름)
OS/2, Windows 3.x 및 Windows 32비트 운영 체제에서 제품이 설치된 디렉토리를 지정할 때 사용되는 환경 변수		
DB2_STRIPED_CONTAINERS	모두	기본값: 널(NULL) 값: 설정, 널(NULL)
테이블 공간 컨테이너에서 RAID 장치를 사용하는 경우, RAID 스트라이프 크기와 같거나 배수가 되는 Extent 크기로 테이블 공간을 작성하는 것이 바람직합니다. 그러나 하나의 페이지 컨테이너 태그로 인해 extent는 RAID 스트라이프로 정렬되지 않으며 I/O 요청 중에 최적의 수 이상의 물리 디스크에 액세스해야 합니다. DMS 테이블 공간 컨테이너를 사용할 경우, 자체의 고유한 전체 extent에 태그를 할당하여 이러한 문제점을 피할 수 있습니다. 이와 같이 하면 문제점은 피할 수 있지만 컨테이너에 여분의 Extent 오버헤드가 필요합니다. 레지스트리 변수를 설정한 후에 DB2STOP을 발행하고 DB2START를 입력하여 변경사항이 적용되도록 하십시오.		

표 23. 통신 변수

변수 이름 설명	운영 체제	값
DB2CHECKCLIENTINTERVAL	AIX, 서버 전용	기본값=0 값: 0보다 큰 숫자 값
APPC 클라이언트 연결 상태를 검증하는 데 사용됩니다. 조회가 완료될 때까지 기다리기보다 클라이언트 종료를 미리 검출할 수 있습니다. 0으로 설정되면 점검되지 않습니다. 0보다 큰 숫자 값으로 설정되면 값은 DB2 내부 작업 단위를 나타냅니다. 지침으로, 다음과 같은 점검 자주 사용되는 값이 제공됩니다. 낮은 빈도는 300을 사용하고 중간 빈도는 100을 사용하며 높은 빈도는 50을 사용합니다. 데이터베이스 요청을 실행하면서 클라이언트 상태를 더 자주 점검하면 조회를 완료하는 데 걸리는 시간이 길어집니다. DB2 워크로드가 많을 때(즉, 많은 내부 요청을 처리해야 되는 경우) DB2CHECKCLIENTINTERVAL을 낮은 값으로 설정하면 워크로드가 적고 DB2가 대부분의 시간 동안 대기 상태에 있을 때보다 성능에 더 많은 영향을 줍니다.		
DB2COMM	모두, 서버 전용	기본값=널(NULL) 값: APPC, IPXSPX, NETBIOS, NPIPE, TCPIP의 모든 조합
데이터베이스 관리 프로그램을 시작할 때 시작된 통신 관리 프로그램을 지정합니다. 이 변수가 설정되지 않으면 어떠한 DB2 통신 관리 프로그램도 서버에서 시작할 수 없습니다.		

표 23. 통신 변수 (계속)

변수 이름 설명	운영 체제	값
DB2_FORCE_NLS_CACHE	AIX, HP_UX, Solaris	기본값=FALSE 값: TRUE 또는 FALSE
다중 스레드 응용프로그램에서 잠금 경합의 가능성을 제거하는 데 사용됩니다. 이 레지스트리 변수가 『TRUE』이면 코드 페이지 및 국가 코드 정보는 스레드가 처음으로 액세스할 때 저장됩니다. 그 시점에서 캐쉬된 정보는 이 정보를 요청하는 다른 스레드에 사용됩니다. 이는 잠금 경합을 제거하고 특정 상황에서 성능 이득을 가져옵니다. 이 설정은 응용프로그램이 연결간의 로케일 설정을 변경할 때 사용되지 않아야 합니다. 다중 스레드 응용프로그램은 보통 로케일 설정을 변경하는 것이 『스레드 안전』하지 않기 때문에 이를 변경하지 않으므로, 이러한 상황에서는 필요하지 않습니다.		
DB2NBADAPTERS	OS/2 및 Windows NT	기본값=0 범위: 0-15, 쉼표로 다중 값을 분리해야 합니다.
DB2 NetBIOS LAN 통신에 사용할 지역 어댑터를 지정하는 데 사용됩니다. 지역 어댑터 번호를 사용하여 지역 어댑터를 지정합니다.		
DB2NBCHECKUPTIME	OS/2 및 Windows NT, 서버 전용	기본값=1분 값: 1-720
NetBIOS 프로토콜 점검 프로시듀어를 호출하는 시간 간격을 지정합니다. 점검 시간은 분으로 지정됩니다. 낮은 값은 기대하지 않은 에이전트/세션 종료가 발생할 때 남겨진 메모리와 다른 시스템 자원을 해제하여 NetBIOS 프로토콜 점검이 더욱 자주 수행되는지 확인합니다.		
DB2NBINTRLISTENS	OS/2 및 Windows NT, 서버 전용	기본값=1 값: 1-10 쉼표로 다중 값을 분리해야 합니다.
원격 클라이언트 인터럽트 준비에 비동기적으로 발행된 명령(NCB)을 전송하는 NetBIOS 청취 수를 지정합니다. 이러한 융통성은 "인터럽트 사용 중" 환경에 제공되어, 서버가 다른 원격 인터럽트 지원에 사용 중일 때 원격 클라이언트를 호출하는 인터럽트가 연결을 설정할 수 있게 보장합니다. DB2NBINTRLISTENS를 낮은 값으로 설정하여 서버의 NCB 및 NetBIOS 세션을 보존합니다. 그러나 클라이언트 인터럽트가 보편적인 환경에서는 인터럽트하고 있는 클라이언트에 응답하기 위해 DB2NBINTRLISTENS를 높은 값에 설정해야 할 수도 있습니다. **주:** 지정된 값은 위치에 따라 달라지고 DB2NBADAPTERS에 해당하는 값 위치와 관련이 있습니다.		
DB2NBRECVBUFFSIZE	OS/2 및 Windows NT, 서버 전용	기본값=4096바이트 범위: 4096-65536

표 23. 통신 변수 (계속)

변수 이름 설명	운영 체제	값
DB2 NetBIOS 프로토콜 수신 버퍼의 *크기*를 지정합니다. 이 버퍼는 NetBIOS 수신 NCB에 할당됩니다. 클라이언트 데이터 전송이 더욱 커질 때 더 높은 값이 필요한 반면, 더 낮은 값은 서버 메모리를 보존합니다.		
DB2NBBRECVNCBS	OS/2 및 Windows NT, 서버 전용	기본값=10 범위: 1-99
조작 동안 서버가 발행하고 유지보수하는 NetBIOS ″receive_any″ 명령(NCB)의 수를 지정합니다. 서버에 연결된 원격 클라이언트의 수에 따라 값이 조정될 수도 있습니다. 더 낮은 값이 서버 자원을 보존합니다. **주:** 사용 중인 각 어댑터에는 DB2NBBRECVNCBS로 지정된 고유 수신 NCB 값이 있습니다. 지정된 값은 위치에 따라 작동되고 DB2NBADAPTERS에 해당하는 값 위치와 관련이 있습니다.		
DB2NBRESOURCES	OS/2 및 Windows NT, 서버 전용	기본값=널(NULL)
다중 문맥 환경에서 사용 중인 DB2용으로 할당하기 위한 NetBIOS 자원의 수를 지정합니다. 이 값은 다중 문맥 클라이언트 조작에는 제한적으로 사용됩니다.		
DB2NBSENDNCBS	OS/2 및 Windows NT, 서버 전용	기본값=6 범위: 1-720
서버가 사용하도록 등록하는 NetBIOS 명령(NCB)의 수를 지정합니다. 서버에 연결된 원격 클라이언트의 수에 따라 값이 조정될 수도 있습니다. DB2NBSENDNCBS를 더 낮은 값에 설정하여 서버 자원을 보존합니다. 그러나 다른 모든 송신 명령을 사용 중일 때 더 높은 값으로 설정하여 서버가 원격 클라이언트에 전송하기 위해 대기하지 않도록 해야 할 필요가 있을 수 있습니다.		
DB2NBSESSIONS	OS/2 및 Windows NT, 서버 전용	기본값=널(NULL) 범위: 5-254
DB2용으로 예약하기 위해 DB2가 요청해야 하는 세션의 수를 지정합니다. DB2NBSESSIONS의 값은 DB2NBADAPTERS를 사용하여 지정한 각 어댑터에 대한 특정 세션을 요청하기 위해 설정될 수 있습니다. **주:** 지정된 값은 위치에 따라 달라지고 DB2NBADAPTERS에 해당하는 값 위치와 관련이 있습니다.		
DB2NBXTRANCBS	OS/2 및 Windows NT, 서버 전용	기본값=어댑터당 5 범위: 5-254
db2start 명령이 발행될 때 서버가 예약해야 하는 ″여분의″ NetBIOS 명령(NCB)의 수를 지정합니다. DB2NBXTRANCBS의 값은 DB2NBADAPTERS를 사용하여 지정한 각 어댑터에 대한 특정 세션을 요청하기 위해 설정될 수 있습니다.		
DB2NETREQ	Windows 3.x	기본값=3 범위: 0-25

표 23. 통신 변수 (계속)

변수 이름	운영 체제	값
설명		
Windows 3.x 클라이언트에서 현재 수행될 수 있는 NetBIOS 요청 수를 지정합니다. 이 값에 더 높은 값을 설정하면 할수록 1MB 레벨 내에서 더 많은 메모리가 사용됩니다. NetBIOS 서비스를 사용하려는 동시 요청의 수가 사용자가 설정한 수에 이르면 NetBIOS 서비스에 대해 들어오는 후속 요청은 대기행렬에 보유되어, 현재 요청이 완료될 때 활동하게 됩니다. DB2NETREQ에 대해 0을 입력할 경우, NetBIOS 대기 옵션을 사용하여 Windows 데이터베이스 클라이언트는 NetBIOS 호출을 동기 모드로 발행합니다. 동기 모드에서 데이터베이스 클라이언트는 현재 NetBIOS 요청이 활동할 수 있게 하고, 현재 요청이 완료되어야 비로소 다른 요청을 처리합니다. 그러나 다른 응용프로그램에 영향을 줄 수 있습니다. 역방향 호환에만 0을 제공합니다. 0을 사용하지 않는 것이 매우 좋습니다.		
DB2RETRY	OS/2 및 Windows NT	기본값=0 범위: 0-20 000
DB2가 APPC 리스너를 재시작하는 횟수. 서버/게이트웨이에서의 SNA 서브시스템이 정지되면 DB2RETRYTIME과 함께 이 프로파일 변수는 기타 프로토콜을 사용하는 클라이언트 통신을 중단시키지 않고 APPC 리스너를 자동으로 재시작하는 데 사용됩니다. 이러한 시나리오에서 APPC 클라이언트 통신을 회복시키기 위해 더 이상 DB2를 중지 및 재시작할 필요가 없습니다.		
DB2RETRYTIME	OS/2 및 Windows NT	기본값=1분 범위: 0-7 200분
증가분이 1분일 때, APPC 리스너를 시작하기 위해 DB2가 수행하는 연속적인 재시도간의 분의 수. 서버/게이트웨이에서 SNA 서브시스템이 정지되면 DB2RETRY와 함께 이 프로파일 변수는 기타 프로토콜을 사용하는 클라이언트 통신을 방해하지 않고 APPC 리스너를 자동으로 재시작하는 데 사용됩니다. 이러한 시나리오에서 APPC 클라이언트 통신을 회복시키기 위해 더 이상 DB2를 중지 및 재시작할 필요가 없습니다.		
DB2SERVICETPINSTANCE	OS/2, Windows NT, AIX 및 Sun Solaris	기본값=널(NULL)
다음으로 인한 문제점을 해결하는 데 사용됩니다. • 동일한 머신에서 수행하는 둘 이상의 인스턴스 • 동일한 TP 이름을 등록하려고 하는 동일한 머신에서 수행하는 버전 6 또는 버전 7 인스턴스 **db2start** 명령이 호출되면 지정된 인스턴스는 다음 TP 이름에 대한 APPC 리스너를 시작합니다. • DB2DRDA • x'07'6DB		
DB2SOSNDBUF	Windows 95 및 Windows NT	기본값=32767
Windows 95 및 Windows NT 운영 체제에 버퍼를 전송하는 TCP/IP 값을 지정합니다.		

표 23. 통신 변수 (계속)

변수 이름 설명	운영 체제	값
DB2SYSPLEX_SERVER	OS/2, Windows NT 및 UNIX	기본값=널(NULL)
OS/390용 DB2에 연결될 때 SYSPLEX 사용이 작동 가능한지를 지정합니다. 이 레지스트리 변수를 설정하지 않거나(기본값) 0이 아닌 값으로 설정하면 사용할 수 있게 됩니다. 이 레지스트리 값이 0으로 설정되면 사용할 수 없게 됩니다. 0으로 설정되면 DCS 데이터베이스 카탈로그 항목이 지정된 방법에 관계없이 게이트웨이에 대해 SYSPLEX를 사용할 수 없게 됩니다. 자세한 내용은 *Command Reference* 및 **CATALOG DCS DATABASE** 명령을 참조하십시오.		
DB2TCPCONNMGRS	모두	직렬 머신에서 기본값=1. 프로세서 수의 제곱근은 대칭적 멀티프로세서 머신에 있는 최대 8개의 연결 관리 프로그램까지 반올림됩니다. 값: 1 - 8
레지스트리 변수를 설정하지 않으면 기본 개수만큼의 연결 관리 프로그램이 작성됩니다. 레지스트리 변수가 설정되면 여기에 지정되는 값이 기본값을 대체합니다. 최대 8까지 지정되는 갯수만큼 TCP/IP 연결 관리 프로그램이 작성됩니다. 1보다 작게 지정하면 DB2TCPCONNMGRS는 1로 설정되고 값이 범위를 벗어났음을 알리는 경고가 로그됩니다. 8보다 크게 지정하면 DB2TCPCONNMGRS는 8로 설정되고 값이 범위를 벗어났음을 알리는 경고가 로그됩니다. 1 - 8 사이의 값이 제공되는 대로 사용됩니다. 두 개 이상의 연결 관리 프로그램이 작성되면 다중 클라이언트 연결이 동시에 수신될 때 연결 처리량을 향상시켜야 합니다. 사용자가 SMP 머신에서 수행 중이거나 DB2TCPCONNMGRS 레지스트리 변수를 수정하면 추가 TCP/IP 연결 관리 프로그램 프로세스(UNIX에서)나 스레드(OS/2 및 Windows 운영 체제에서)가 있을 수도 있습니다. 추가 프로세스나 스레드에는 추가 저장영역이 필요합니다. **주:** 연결 관리 프로그램의 수를 1로 설정하는 경우, 다수의 사용자를 가지거나 빈번하게 연결하고 연결 해제하는(또는 둘 다) 시스템에 있어서 원격 연결 성능이 저하될 수 있습니다.		
DB2_VI_ENABLE	Windows NT	기본값=OFF 값: ON 또는 OFF
가상 인터페이스(VI) 아키텍처 통신 프로토콜의 사용 여부를 지정합니다. 이 레지스트리 값이 『ON』이면 FCM은 노드간 통신에 VI를 사용하게 됩니다. 이 레지스트리 값이 『OFF』이면 FCM은 노드간 통신에 TCP/IP를 사용하게 됩니다. **주:** 이 레지스트리 변수의 값은 인스턴스의 모든 데이터베이스 파티션에서 같아야 합니다.		
DB2_VI_VIPL	Windows NT	기본값=`vipl.dll`
DB2가 사용하는 가상 인터페이스 제공업체 라이브러리(VIPL)의 이름을 지정합니다. 라이브러리를 성공적으로 로드하려면 이 레지스트리 변수에 사용되는 라이브러리 이름이 PATH 사용자 환경 변수에 있어야 합니다. 현재 지원되는 구현은 모두 동일한 라이브러리 이름을 사용합니다.		
DB2_VI_DEVICE	Windows NT	기본값=널(NULL) 값: `nic0` 또는 `VINIC`

표 23. 통신 변수 (계속)

변수 이름	운영 체제	값
설명		
네트워크 인터페이스 카드(NIC)에 관련된 장치 또는 가상 인터페이스 제공업체 인스턴스의 기호 이름을 지정합니다. 독립 하드웨어 벤더(IHV)는 각각 자체 NIC를 생산합니다. Windows NT 머신당 하나의 NIC만 허용됩니다. 같은 물리적 머신에 있는 다중 논리 노드는 같은 NIC를 공유합니다. 기호 장치 이름『VINIC』는 대문자여야 하며, Synfinity Interconnect와만 사용할 수 있습니다. 현재 지원되는 모든 구현은 기호 장치 이름으로『nic0』을 사용합니다.		

표 24. DCE 디렉토리 변수

변수 이름	운영 체제	값
설명		
DB2DIRPATHNAME	OS/2, UNIX 및 Windows 32비트 운영 체제	기본값=널(NULL)
데이터베이스 관리 프로그램 구성 파일에 DIR_PATH_NAME 매개변수 값의 임시 겹쳐쓰기를 지정합니다. 디렉토리 서버가 사용되고 CONNECT문 또는 ATTACH문이 명시적으로 카탈로그화되지 않을 경우, 완전한 DCE 이름을 형성하도록 DB2DIRPATHNAME(지정된 경우)와 목표를 병합해야 합니다. **주:** DB2DIRPATHNAME 변수는 인스턴스의 전역 이름에 영향을 주지는 않지만, 항상 데이터베이스 관리 프로그램 구성 매개변수 DIR_PATH_NAME 및 DIR_OBJ_NAME에 의해 식별됩니다.		
DB2CLIENTADPT	OS/2 및 Windows 32비트 운영 체제	기본값=널(NULL) 범위: 0-15
OS/2 및 Windows 32비트 운영 체제에서 NETBIOS 프로토콜에 대한 클라이언트 어댑터 번호를 사용합니다. DB2CLIENTADPT 값은 데이터베이스 관리 프로그램 구성 파일의 DFT_CLIENT_ADPT 매개변수 값을 겹쳐씁니다.		
DB2CLIENTCOMM	OS/2, UNIX 및 Windows 32비트 운영 체제	기본값=널(NULL)
데이터베이스 관리 프로그램 구성 파일에 DFT_CLIENT_COMM 매개변수 값의 임시 겹쳐쓰기를 지정합니다. DFT_CLIENT_COMM 및 DB2CLIENTCOMM 둘다 지정하지 않은 경우, 오브젝트에서 발견된 첫 번째 프로토콜이 사용됩니다. 둘 중 하나 또는 둘다를 지정한 경우, 일치하는 첫 번째 프로토콜만을 사용합니다. 어느쪽의 경우에도 첫 번째 연결에 실패할 경우, 재시도하지 않습니다.		
DB2ROUTE	OS/2, UNIX 및 Windows 32비트 운영 체제	기본값=널(NULL)
데이터베이스와 다른 데이터베이스 프로토콜을 연결할 때 클라이언트가 사용하는 경로지정 정보 오브젝트 이름을 지정합니다. DB2ROUTE 값은 데이터베이스 관리 프로그램 구성 파일의 ROUTE_OBJ_NAME 매개변수 값을 겹쳐씁니다.		

표 25. 명령행 변수

변수 이름 설명	운영 체제	값
DB2BQTIME	모두	기본값=1초 최대값: 1초
백 엔드 처리가 사용 중인지 연결을 하고 있는지 점검하기 전에 명령행 처리기 프론트 엔드가 쉬고 있는 시간을 지정합니다.		
DB2BQTRY	모두	기본값=60회 재시도 최소값: 0회 재시도
백 엔드 처리를 이미 사용 중인지 판별하기 위한 명령행 처리기 프론트 엔드 처리의 시도 횟수를 지정합니다. DB2BQTIME과 함께 작동합니다.		
DB2IQTIME	모두	기본값=5초 최소값: 1초
명령행 처리기 백 엔드 처리가 명령을 통과시키기 위해 프론트 엔드 처리에 대해 입력 대기행렬에서 기다리는 시간을 지정합니다.		
DB2RQTIME	모두	기본값=5초 최소값: 1초
명령행 처리기 백 엔드 처리가 프론트 엔드 처리에서부터의 요청을 기다리는 시간을 지정합니다.		

표 26. MPP 구성 변수

변수 이름	운영 체제	값
DB2ATLD_PORTS	AIX, Solaris 및 Windows NT에서의 DB2 UDB EEE	기본값= 6000:6063 값: num1:num2. 여기서 두 숫자 모두 1 - 65535 및 num1<=num2
자동 로드 프로그램 유틸리티의 내부 TCPIP 통신에 사용될 포트 번호의 범위를 지정합니다. 설정되지 않으면 자동 로드 프로그램은 내부 기본 포트 범위 6000:6063을 사용합니다. 자동 로드 프로그램 기본 포트 범위를 사용하는 다른 응용프로그램이 있을 경우, 이 변수를 사용하여 대체 포트 범위를 선택할 수 있습니다.		
DB2ATLD_PWFILE	AIX, Solaris 및 Windows NT에서의 DB2 UDB EEE	기본값=널(NULL) 값: 파일 경로 표현식
자동 로드 프로그램 인증시 사용되는 암호가 있는 파일에 대한 경로를 지정합니다. 설정하지 않으면 자동 로드 프로그램은 해당 구성 파일에서 암호를 발췌하고 대화식으로 프롬프트를 표시합니다. 이 변수를 사용하면, 주소 암호 보안이 관련되고 인증 정보로부터 자동 로드 프로그램 구성 정보가 구분될 수 있게 됩니다.		

표 26. MPP 구성 변수 (계속)

변수 이름	운영 체제	값
DB2CHGPWD_EEE	AIX 및 Windows NT에서의 DB2 UDB EEE	기본값=널(NULL) 값: YES 또는 NO
다른 사용자가 AIX 또는 Windows NT EEE 시스템에서 암호를 변경할 수 있도록 허용할 것인지 여부를 지정합니다. 모든 파티션 및 노드에 대한 암호는 Windows NT에서는 Windows NT 도메인 제어기, AIX에서는 NIS를 사용하여 중앙에서 유지보수되도록 해야 합니다. 중앙에서 유지보수되지 않을 경우, 암호는 모든 플랫폼 또는 노드간에 일치하지 않게 됩니다. 이렇게 되면 변경을 하기 위해 연결한 데이터베이스 파티션에서만 암호를 변경할 수 있게 됩니다. 이 전역 레지스트리 변수를 수정하려면 루트 디렉토리의 DAS 인스턴스에 있어야 합니다.		
DB2_FORCE_FCM_BP	AIX	기본값=아니오 값: 예 또는 아니오
이 레지스트리 변수는 다중 논리 파티션을 사용할 때 AIX용 DB2 UDB EEE에 적용될 수 있습니다. DB2START가 발행되면 DB2는 데이터베이스 전역 메모리에서 FCM 버퍼를 할당하거나 또는 메모리에 공간이 부족하면 같은 물리적 머신의 모든 FCM 디먼(해당 인스턴스에 대해)에 의해 사용되는 별도의 공유 메모리 세그먼트에서 할당합니다. 선택사항은 주로 작성될 FCM 버퍼 수에 따라 다릅니다. (이는 차례로 FCM_NUM_BUFFERS 데이터베이스 관리 프로그램 구성 매개변수에 의해 결정됩니다.) 레지스트리 변수가 예로 설정되면 FCM 버퍼는 항상 별도의 메모리 세그먼트에 작성됩니다. FCM 버퍼가 별도의 메모리 세그먼트에서 작성되는 경우, 통신은 같은 물리 노드의 서로 다른 논리적 파티션에 있는 FCM 디먼 사이에서 공유 메모리를 통해 발생합니다. 그렇지 않으면 같은 노드에 있는 FCM 디먼은 UNIX 소켓을 통해 통신합니다. 이러한 방법으로 공유 메모리를 통해 통신하는 것이 더 빠릅니다. 그러나 단점은 다른 용도, 특히 데이터베이스 버퍼 풀을 위해 사용할 수 있는 공유 메모리가 하나 적어진다는 것입니다. 이는 데이터베이스 버퍼 풀의 최대 크기를 감소시킵니다.		
DB2_NUM_FAILOVER_NODES	모두	기본값: 2 값: 0 - 논리 노드 수
고가용성 환경에서 failover 노드로 사용할 수 있는 노드 수를 지정합니다. 고가용성 환경에서 노드가 실패할 경우, 그 노드는 다른 호스트의 두 번째 논리 노드로 재시작될 수 있습니다. 이 변수에서 사용되는 숫자는 failover 노드의 FCM 자원에 대해 예약될 메모리 양을 결정합니다. 예를 들어, 호스트 A에 두 개의 논리 노드인 1과 2가 있고, 호스트 B에는 두 개의 논리 노드 3과 4가 있을 경우, DB2_NUM_FAILOVER_NODES를 2로 설정했다고 가정해 보십시오. DB2START 동안, 호스트 A와 호스트 B는 둘 다 FCM에 대해 충분한 메모리를 예약하므로 네 개까지 논리 노드를 관리할 수 있습니다. 이 때 하나의 호스트가 실패하면 실패한 호스트의 논리 노드는 다른 호스트에서 재시작할 수 있습니다.		
DB2PORTRANGE	Windows NT	값: nnnn:nnnn
이 값은 또 다른 머신에서 작성되는 추가 파티션도 같은 포트 범위를 갖도록 FCM에 의해 사용되는 TCP/IP 포트 범위로 설정됩니다.		
DB2_UPDATE_PART_KEY	ALL	기본값=예 값: 예 또는 아니오
수정팩 3 이상인 경우, 기본값은 "예"입니다. 이 레지스트리 변수는 파티션 키의 갱신이 허용되는지를 지정합니다.		

표 27. SQL 컴파일러 변수

변수 이름 설명	운영 체제	값
DB2_ANTIJOIN	모두	EEE 환경에서 기본값=아니오 비 EEE 환경에서 기본값=예 값: YES 또는 NO
DB2 Universal Database EEE 환경의 경우, "예"가 지정되면 최적화 알고리즘은『NOT EXISTS』부속 조회를 DB2가 더 효율적으로 처리할 수 있는 안티 조인으로 변환할 수 있는 기회를 찾습니다. 비 EEE 환경의 경우, "아니오"로 지정되면 최적화 알고리즘은『NOT EXISTS』부속 조회를 안티 조인으로 변환하는 기회를 제한합니다.		
DB2_CORRELATED_PREDICATES	모두	기본값=예 값: 예 또는 아니오
이 변수의 기본값은 "예"입니다. 조인의 상관 컬럼에 고유 색인이 있고 이 레지스트리 변수가 "예"인 경우, 최적화 알고리즘은 Join 술어의 상관을 검출하고 보상하려고 시도합니다. 이 레지스트리 변수가 "예"인 경우, 최적화 알고리즘은 고유 색인 통계의 KEYCARD 정보를 사용하여 상관 경우를 검출하고, 상관 술어의 조합된 선택성을 동적으로 조정하여 더 정확한 조인 *크기* 및 비용 계산 값을 얻습니다. 또한 C1 및 C2에 색인이 있는 경우, `WHERE C1=5 AND C2=10`과 같은 간단한 등호 술어의 상관에 대해 조정이 수행됩니다. 색인이 고유할 필요는 없지만 등호 술어 컬럼은 해당 색인의 모든 컬럼을 다루어야 합니다.		
DB2_HASH_JOIN	모두	기본값=NO 값: YES 또는 NO
액세스 플랜 컴파일시 가능한 조인 방법으로서 해쉬 조인을 지정합니다.		
DB2_LIKE_VARCHAR	모두	기본값=Y,N 값: Y, N, S 또는 0 - 6.2의 부동 소수점 상수

표 27. SQL 컴파일러 변수 (계속)

변수 이름	운영 체제	값 설명

		콜렉션 및 부속 요소의 사용 통계를 제어합니다. 이것은 데이터가 일련의 부속 필드 또는 공백으로 분리된 부속 요소 양식의 구조를 가질 때 컬럼 데이터의 내용에 대한 통계입니다.
		
		이 레지스트리 변수는 최적화 알고리즘이 양식의 술어를 처리하는 방식에 영향을 줍니다.
		
		`COLUMN LIKE '%xxxxxx%'`
		
		여기서 xxxxxx는 모든 문자열입니다.
		
		이 레지스트리 변수가 사용되는 방법을 보여주는 구문은 다음과 같습니다.
		
		`db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]`
		
		여기서
		• 쉼표 앞에 오는 조건 또는 술어의 오른쪽에 오는 조건은 양수의 부속 요소 통계를 갖지 않는 컬럼에 대한 것만 따름을 의미합니다.
		– S - 최적화 알고리즘이 %로 묶인 문자열의 길이에 따라 컬럼을 형성하도록 병합된 일련의 요소에 있는 각 요소의 길이를 추정합니다.
		– Y - 기본값. 알고리즘 매개변수의 기본값 1.9를 사용합니다. 알고리즘 매개변수와 함께 변수 길이 부속 요소 알고리즘을 사용합니다.
		– N - 고정 길이 부속 요소 알고리즘을 사용합니다.
		– num1 - 알고리즘 매개변수로 값 num1을 변수 길이 부속 요소 알고리즘과 함께 사용합니다.
		• 쉼표 다음의 조건의 의미는 다음과 같습니다.
		– N - 기본값. 부속 요소 통계를 수집하거나 사용하지 않습니다.
		– Y - 부속 요소 통계를 수집합니다. 양수의 부속 요소 통계가 있는 컬럼의 경우, 수집된 통계를 알고리즘 매개변수의 기본값 1.9와 함께 사용하는 변수 길이 부속 요소 알고리즘을 사용합니다.
		– num2 - 부속 요소 통계를 수집합니다. 양수의 부속 요소 통계가 있는 컬럼의 경우, 알고리즘 매개변수로 값 num2와 함께 수집된 통계를 사용하는 변수 길이 부속 요소 알고리즘을 사용합니다.

DB2_SELECTIVITY	ALL	기본값=아니오 값: 예 또는 아니오

표 27. SQL 컴파일러 변수 (계속)

변수 이름 설명	운영 체제	값
이 레지스트리 변수는 SELECTIVITY 절을 사용할 수 있는 곳을 제어합니다. SELECTIVITY 절에 대한 자세한 내용은 *SQL 참조서*, 언어 요소, 검색 조건을 참조하십시오. 이 레지스트리 변수가 "예"로 설정되면, 술어가 최소한 한 표현식에 호스트 변수가 있는 기본 술어일 때 SELECTIVITY 절이 지정될 수 있습니다.		
DB2_NEW_CORR_SQ_FF	모두	기본값=OFF 값: ON 또는 OFF

『ON』으로 설정되면, 특정 부속 조회 술어에 대해 SQL 최적화 알고리즘이 계산한 선택성 값에 영향을 미칩니다. 이는 부속 조회의 SELECT 목록에 MIN 또는 MAX 총계 함수를 사용하는 등호 부속 조회 술어의 선택성 값의 정확도를 높이기 위해 사용될 수 있습니다. 예를 들면, 다음과 같습니다.
`SELECT * FROM T WHERE`
`T.COL = (SELECT MIN(T.COL)`
`FROM T WHERE ...)`		

DB2_PRED_FACTORIZE	모두	기본값=NO 값: YES 또는 NO

표 27. SQL 컴파일러 변수 (계속)

변수 이름	운영 체제	값
설명		

최적화 알고리즘이 분리된 것들에서 추가 술어를 발췌할 수 있는 기회를 찾을지 여부를 지정합니다. 어떠한 상황에서는 추가 술어가 중간 및 최종 결과 세트에 대한 추정된 기본 행 수(cardinality)를 변경할 수 있습니다. 다음 조회에서,
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`
최적화 알고리즘은 다음과 같은 추가 술어를 생성할 수 있습니다.
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE n1.lastname IN`
`('SMITH', 'JONES')`
`AND n2.lastname IN`
`('SMITH', 'JONES')`
`AND`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`		

표 28. 성능 변수

변수 이름	운영 체제	값
설명		
DB2_AVOID_PREFETCH	모두	기본값=OFF 값: ON 또는 OFF
응급 복구 중에 프리페치 사용 여부를 지정합니다. DB2_AVOID_PREFETCH=ON이면 프리페치가 사용되지 않습니다.		

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		
DB2_AWE	Windows 2000	기본값=널(NULL) 값: <entry>[,<entry>,...] 여기서 <entry>=<버퍼 풀 ID>, <물리적 페이지 수>, <주소 창 번호>
Windows 2000에서 DB2 UDB가 최대 64GB의 메모리를 사용하는 버퍼 풀을 할당하도록 합니다. Windows 2000은 AWE(Address Windowing Extensions) 버퍼 풀을 지원하도록 올바르게 구성되어야 합니다. 이 구성 작업은 『메모리의 잠금 페이지』를 사용자와 연관시키고, 물리적 페이지 및 주소 창 페이지를 할당하며, 이 레지스트리 변수를 설정하는 것을 포함합니다. 이 변수 설정에서는 AWE 지원에 사용할 버퍼 풀의 버퍼 풀 ID를 알고 있어야 합니다. 버퍼 풀 ID는 SYSCAT.BUFFERPOOLS 시스템 카탈로그 뷰의 BUFFERPOOLS 컬럼에 표시될 수 있습니다. **주:** AWE 지원을 사용할 수 있는 경우, 확장 저장영역은 데이터베이스의 버퍼 풀에 사용할 수 없습니다. 또한 이 레지스트리 변수로 참조되는 버퍼 풀이 SYSCAT.SYSBUFFERPOOLS에 이미 있어야 합니다.		
DB2_BINSORT	모두	기본값=YES 값: YES 또는 NO
CPU 시간 및 정렬의 경과 시간을 줄이는 새 정렬 알고리즘이 사용 가능합니다. 이 새 알고리즘은 효율을 극대화한 DB2 UDB의 정수 정렬 기법을 BIGINT, CHAR, VARCHAR, FLOAT 및 DECIMAL과 같은 모든 정렬 데이터 유형뿐 아니라, 이러한 유형을 조합한 데이터 유형으로도 확대했습니다. 이 새 알고리즘을 사용 가능하게 하려면 다음 명령을 사용하십시오. `db2set DB2_BINSORT = yes`		
DB2BPVARS	Windows NT	기본값=경로

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		

버퍼 풀을 조정할 때 사용되는 매개변수를 포함하는 파일의 경로를 지정합니다. 현재 지원되는 매개변수는 NT_SCATTER_DMSFILE, NT_SCATTER_DMSDEVICE 및 NT_SCATTER_SMS입니다.
 	
이러한 각 매개변수의 기본값은 0(또는 OFF)이며, 가능한 값은 0(또는 OFF)과 1(또는 ON)입니다. 각 매개변수는 각 컨테이너 유형에 대해 분산 읽기를 ON으로 설정하는 데 사용됩니다. 레지스트리에서 DB2NTNOCACHE가 ON으로 설정된 경우, 각각은 사용 가능으로만 될 수 있습니다(ON으로 설정). DB2NTNOCACHE가 OFF로 설정된 경우(또는 설정되지 않은 경우), 경고 메시지가 db2diag.log에 기록됩니다. 분산 읽기는 사용 불가능한 상태로 남아 있습니다. 시스템에는 각각의 컨테이너 유형에 대해 대량의 순차 프리페치를 가진 매개변수가 권장되며, 이미 ON으로 설정된 DB2NTNOCACHE를 사용하기로 결정되었습니다.
주: DB2NTNOCACHE를 ON으로 설정할 때 Windows NT 파일 캐싱을 사용하지 않습니다.
 	
다음 예에는 파일에 대한 경로를 설정하는 방법입니다.
`db2set DB2BPVARS =`
`f:\BPVARSFILE`
 	
파일의 내용은 다음 양식을 가진 매개변수로 구성됩니다.
`parameter=value`		

DB2CHKPTR	모두	기본값=OFF 값: ON 또는 OFF
입력 점검을 하는 포인터가 필요한지 여부를 지정합니다.		
DB2_ENABLE_BUFPD	모두	기본값=OFF 값: ON 또는 OFF
조회 성능을 향상시키기 위해 DB2가 중간 버퍼링을 사용하는지 여부를 지정합니다. 버퍼링이 모든 환경에서 조회 성능을 향상시키지는 않습니다. 테스트를 수행하여 개별적인 조회 성능 향상사항을 판별해야 합니다.		
DB2_EXTENDED_OPTIMIZATION	모두	기본값=OFF 값: ON 또는 OFF
조회 최적화 알고리즘이 조회 성능 향상을 위해 최적화 확장을 사용하는지 여부를 지정합니다. 확장이 모든 환경에서 조회 성능을 향상시키지는 않습니다. 테스트를 수행하여 개별적인 조회 성능 향상사항을 판별해야 합니다.		
DB2MAXFSCRSEARCH	모두	기본값=5 값: -1, 1 - 33 554

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		
테이블에 레코드를 추가할 때 검색할 여유 공간 제어 레코드의 수를 지정합니다. 기본값은 5개의 여유 공간 제어 레코드를 검색하는 것입니다. 이 값을 수정하면 공간 재사용과 삽입 속도의 균형을 유지할 수 있습니다. 공간 재사용을 최적화려면 큰 값을 사용하십시오. 삽입 속도를 최적화하려면 작은 값을 사용합니다. 값을 -1로 설정하면 데이터베이스 관리 프로그램이 모든 여유 공간 제어 레코드를 검색하도록 강제합니다.		
DB2MEMDISCLAIM	AIX	기본값=YES 값: YES 또는 NO
AIX에서 DB2 프로세스에 의해 사용된 메모리는 몇 가지 연관된 페이징 공간을 가질 수 있습니다. 이 페이징 공간은 연관된 메모리가 사용 가능한 때에도 예약 상태를 유지할 수 있습니다. 이것의 가능 여부는 AIX 시스템의 (조정된) 가상 메모리 관리 할당 규정에 따릅니다. DB2MEMDISCLAIM 레지스트리 변수는 AIX가 예약된 페이징 공간과 여유 메모리를 분리할 것을 DB2 에이전트가 명시적으로 요청할지 여부를 제어합니다. DB2MEMDISCLAIM을 예로 설정하면 페이징 공간 요구사항이 적어지고 페이징으로부터 디스크 활동이 적어질 수 있습니다. DB2MEMDISCLAIM을 아니오로 설정하면, 페이징 공간 요구사항이 커지고 페이징으로부터 디스크 활동이 많아질 수 있습니다. 페이징이 절대 발생하지 않도록 페이징 공간과 실제 메모리가 충분한 경우, 아니오를 설정하면 성능이 조금 향상됩니다.		
DB2MEMMAXFREE	모두	기본값= 8 388 608바이트 값: 0 - 2^{32}-1바이트
DB2 프로세스에 의해 보유되는 미사용 메모리의 최대량(바이트)을 지정합니다.		
DB2_MMAP_READ	AIX	기본값=ON, 값: ON 또는 OFF
DB2가 대체 I/O 방법으로 mmap를 사용할 수 있도록 하기 위해 DB2_MMAP_WRITE와 함께 사용됩니다. 대부분의 환경에서 여러 프로세스가 동일한 파일의 다른 섹션에 기록할 때 운영 체제가 잠기는 것을 방지하기 위해서는 mmap를 사용해야 합니다. 그러나 기본값이 AIX가 DB2 데이터 캐쉬를 JFS 파일 시스템에서 메모리(버퍼 풀 외부)로 읽어들이게 하는 OFF인 Parallel Edition V1.2에서 이주했을 수도 있습니다. DB2 UDB와 비교할 만큼의 성능을 원하면 버퍼 풀의 크기를 증가시키거나 DB2_MMAP_READ 및 DB2_MMAP_WRITE를 OFF로 변경할 수 있습니다.		
DB2_MMAP_WRITE	AIX	기본값=ON 값: ON 또는 OFF
DB2가 대체 I/O 방법으로 mmap를 사용할 수 있도록 하기 위해 DB2_MMAP_READ와 함께 사용됩니다. 대부분의 환경에서 여러 프로세스가 동일한 파일의 다른 섹션에 기록할 때 운영 체제가 잠기는 것을 방지하기 위해서는 mmap를 사용해야 합니다. 그러나 기본값이 AIX가 DB2 데이터 캐쉬를 JFS 파일 시스템에서 메모리(버퍼 풀 외부)로 읽어들이게 하는 OFF인 Parallel Edition V1.2에서 이주했을 수도 있습니다. DB2 UDB와 비교할 만큼의 성능을 원하면 버퍼 풀의 크기를 증가시키거나 DB2_MMAP_READ 및 DB2_MMAP_WRITE를 OFF로 변경할 수 있습니다.		

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		
DB2_NO_PKG_LOCK	모두	기본값=OFF 값: ON 또는 OFF
전역 SQL 캐쉬가 캐쉬된 패키지 항목을 보호하는 패키지 잠금을 사용하지 않고 조작되도록 합니다. (패키지 잠금은 내부 시스템 잠금입니다.) 성능을 향상시키기 위해(잠금을 획득하고 해제하는 데 시간이 소요되므로) 지금은 『패키지 잠금 없음』 모드에서 작업하도록 선택할 수 있습니다. 이 모드에서는 특정 데이터베이스 조작이 허용되지 않습니다. 이러한 조작에는 패키지를 무효화하는 조작, 패키지가 작동되지 못하게 하는 조작, 직접 패키지를 변경하는 조작이 포함됩니다.		
DB2NTMEMSIZE	Windows NT	기본값=(메모리 세그먼트별로 다름)
Windows NT에서는 프로세스간의 주소 일치를 보장하기 위해 DLL 초기화 시간에 모든 공유 메모리 세그먼트가 예약되어 있어야 합니다. DB2NTMEMSIZE는 필요한 경우 Windows NT에서 사용자가 DB2 기본값을 대체할 수 있도록 도입되어 있습니다. 대부분의 경우, 기본값으로 충족되어야 합니다. 메모리 세그멘트, 기본 *크기* 및 겹쳐쓰기 옵션은 다음과 같습니다. 1) 데이터베이스 커널: 기본 *크기*는 16777216(16MB)이며 겹쳐쓰기 옵션은 DBMS:<바이트 수>입니다. 2) 병렬 FCM 버퍼: 기본 *크기*는 22020096(21MB)이며 겹쳐쓰기 옵션은 FCM:<바이트 수>입니다. 3) 데이터베이스 관리 GUI: 기본 *크기*는 33554432(32MB)이며 겹쳐쓰기 옵션은 DBAT:<바이트 수>입니다. 4) 분리(fenced) 저장 프로시듀어: 기본 *크기*는 16777216(16MB)이며 겹쳐쓰기 옵션은 APLD:<바이트 수>입니다. 둘 이상의 세그먼트는 겹쳐쓰기 옵션을 세미콜론(;)으로 분리함으로써 겹쳐쓸 수 있습니다. 예를 들어, 데이터베이스 커널을 약 256K로 제한하고 FCM 버퍼를 약 64MB로 제한하려면 다음 명령을 사용하십시오. `db2set DB2NTMEMSIZE=DBMS:256000;FCM:64000000`		
DB2NTNOCACHE	Windows NT	기본값=OFF 값: ON 또는 OFF
DB2가 NOCACHE 옵션으로 데이터베이스를 여는지 여부를 지정합니다. DB2NTNOCACHE=ON이면 파일 시스템 캐싱이 제거됩니다. DB2NTNOCACHE=OFF이면 운영 체제는 DB2 파일을 캐쉬합니다. long 필드 또는 LOBS가 들어 있는 파일을 제외하고 모든 데이터에 적용됩니다. 시스템 캐싱을 제거하면 더 많은 메모리를 데이터베이스에 사용 가능하게 할 수 있어서, 버퍼 풀 또는 정렬 힙을 증가시킬 수 있습니다.		
DB2NTPRICLASS	Windows NT	기본값=널(NULL) 값: R, H, (기타 모든 값)

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		
DB2 인스턴스의 우선순위 클래스를 설정합니다(프로그램 DB2SYSCS.EXE). 다음은 세 가지의 우선순위 클래스입니다. • NORMAL_PRIORITY_CLASS(기본 우선순위 클래스) • REALTIME_PRIORITY_CLASS(『R』을 사용하여 설정) • HIGH_PRIORITY_CLASS(『H』를 사용하여 설정) 이 변수는 시스템의 다른 스레드에 상대적인 DB2 스레드의 절대 우선순위를 판별하기 위해 개별 스레드 우선순위(DB2PRIORITIES를 사용하여 설정)와 함께 사용됩니다. **주:** 이 변수를 사용할 때에는 주의해야 합니다. 잘못 사용하면 전체 시스템 성능에 영향을 줄 수 있습니다. 자세한 정보는 Win32 문서의 *SetPriorityClass()* API를 참조하십시오.		
DB2NTWORKSET	Windows NT	기본값=1,1
DB2에서 사용 가능한 최소, 최대 작업 설정 크기를 변경하는 데 사용됩니다. 기본적으로 Windows NT가 페이징을 하고 있지 않으면 프로세스의 작업 설정을 필요한 만큼 늘릴 수 있습니다. 그러나 페이징이 발생하면 프로세스가 가질 수 있는 최대 작업 설정은 약 1MB입니다. DB2NTWORKSET를 사용하여 이 기본 잠동을 대체할 수 있습니다. DB2NTWORKSET=`min,max` 구문을 사용하여 DB2에 DB2NTWORKSET를 지정하십시오. 여기서 min 및 max는 MB 단위로 표시됩니다.		
DB2_OVERRIDE_BPF	모두	기본값=설정되지 않음 값: 양수 페이지 수
데이터베이스를 활성화하거나 처음 연결시 작성될 버퍼 풀의 크기를 페이지로 지정합니다. 메모리 제한조건으로 인해 데이터베이스를 활성화하거나 처음 연결시 장애가 발생한 경우에 유용합니다. 데이터베이스 관리 프로그램이 최소 버퍼 풀 크기인 16페이지를 할당하지 못하면 사용자는 이 환경 변수를 사용하여 좀더 작은 페이지 수를 지정한 후 다시 시도할 수 있습니다. 메모리 제한조건은 실제 메모리가 부족(드문 경우임)하거나 데이터베이스 관리 프로그램이 부정확하게 구성된 대형의 버퍼 풀을 할당하려고 할 때 발생할 수 있습니다. 이 값이 설정되면 현재 버퍼 풀 크기가 겹쳐쓰여집니다.		
DB2_PINNED_BP	AIX, HP-UX	기본값=NO 값: YES 또는 NO

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		

이 변수는 AIX 운영 체제의 주 메모리에 있는 데이터베이스와 연관된 데이터베이스 전역 메모리(버퍼 풀 포함)를 보유하는 데 사용됩니다. 시스템의 주 메모리에서 이 데이터베이스 전역 메모리를 유지하면 데이터베이스 성능을 더욱 일관성있게 할 수 있습니다.
 	
예를 들어, 버퍼 풀이 시스템의 주 메모리 밖으로 스와핑되었으면 데이터베이스 성능이 저하됩니다. 시스템 메모리에 버퍼 풀을 둠으로써 디스크 I/O를 감소시키면 데이터베이스 성능이 향상됩니다. 주 메모리를 더 많이 필요로 하는 다른 응용프로그램이 있으면 시스템의 주 메모리 요구사항에 따라 데이터베이스 전역 메모리를 주 메모리 밖으로 스와핑할 수 있도록 할 것입니다.
 	
64비트 환경의 HP-UX로 작업할 때 이 레지스트리 변수 수정 외에도, DB2 인스턴스 그룹에 MLOCK 특권을 제공해야 합니다. 이것은 루트 액세스 권한이 있는 사용자가 다음을 수행하면 됩니다.
1. DB2 인스턴스 그룹을 /etc/privgroup 파일에 추가하십시오. 예를 들어, DB2 인스턴스 그룹이 db2iadm1 그룹에 속하면 /etc/privgroup 파일에 다음 행을 추가해야 합니다.
`db2iadm1 MLOCK`
2. 다음 명령을 실행하십시오.
`setprivgrp -f /etc/privgroup`		

DB2PRIORITIES DB2 프로세스와 스레드의 우선순위를 제어합니다.	모두	값 설정은 플랫폼마다 다릅니다.
DB2_RR_TO_RS	모두	기본값=NO 값: YES 또는 NO
*다음 키 잠금*은 모든 INSERT 및 DELETE문에 대한 다음 키와 SELECT문의 경우, 결과 세트에서 더 높은 다음 키 값을 자동으로 잠궈 반복 읽기(RR) 분리 레벨을 보증합니다. 색인의 키 부분을 변경하는 UPDATE문의 경우, 원래 색인 키는 삭제되고 새 키 값이 삽입됩니다. 다음 키 잠금은 키 삽입과 키 삭제에 대해 모두 수행됩니다. 다음 키 잠금은 ANSI 및 SQL92 표준 RR을 보증하는 데 필요하며 DB2 기본값입니다. 응용프로그램이 중지 또는 정지를 나타내는 경우, 응용프로그램에 대한 스냅샷 정보를 조사하십시오. 문제가 다음 키 잠금에 있는 경우, 두 가지 조건에 기반하여 DB2_RR_TO_RS 레지스트리 변수를 설정할 수 있습니다. 응용프로그램이 반복 읽기(RR) 작동에 의존하지 않고 미확약 삭제에 대해서는 스캔을 승인하도록 할 경우, DB2_RR_TO_RS를 ON으로 설정할 수 있습니다. 생략 작동은 반복 읽기(RR), 읽기 안정성(RS) 및 커서 안정성(CS) 분리 레벨에 영향을 줍니다. (미확약 읽기(UR) 분리 레벨에 대한 행 잠금이 없습니다.) DB2_RR_TO_RS가 ON일 때 다음 키 잠금이 색인 키 삽입 및 삭제 도중에 수행되지 않았기 때문에 사용자 테이블의 스캔에 대해서는 RR 작동이 보장될 수 없습니다. 카탈로그 테이블은 이 옵션에 영향을 받지 않습니다. 기타 작동 변경은 DB2_RR_TO_RS가 ON인 상태에서 스캔에 대해 행이 규정될 수 있더라도 스캔이 삭제되었지만 확약되지 않은 행을 생략합니다.		

표 28. 성능 변수 (계속)

변수 이름	운영 체제	값
설명		
DB2_SORT_AFTER_TQ	모두	기본값=NO 값: YES 또는 NO
수신 종료시 데이터가 정렬되어야 하고 수신 노드 수가 송신 노드 수와 같아야 할 때 최적화 알고리즘이 파티션된 데이터베이스의 지시된 테이블 대기행렬과 함께 작동되는 방법을 지정합니다. DB2_SORT_AFTER_TQ= NO인 경우, 최적화 알고리즘은 송신이 종료되면 행을 정렬하고, 수신이 종료되면 행을 병합하려 합니다. DB2_SORT_AFTER_TQ= YES이면 최적화 알고리즘은 정렬되지 않은 행을 전송하고, 수신 종료시 행을 병합하지 않으며, 모든 행이 수신된 후 수신 종료시 행을 정렬하려고 합니다.		
DB2_STPROC_LOOKUP_FIRST	모두	기본값=OFF 값: ON 또는 OFF
DB2_DARI_LOOKUP_ALL 변수는 *sqllib* 서브디렉토리의 *function* 서브디렉토리, *sqllib* 서브디렉토리의 *function* 서브디렉토리에 있는 *unfenced* 서브디렉토리를 찾아보기 전에 UDB 서버가 모든 DARI 및 저장 프로시듀어에서 카탈로그 찾아보기를 수행할지 여부를 지정합니다. **주:** 위에서 언급한 디렉토리에 위치한 PARAMETER TYPE DB2DARI의 저장 프로시듀어의 경우, 이 값을 『ON』으로 설정하면 카탈로그 찾아보기가 함수 디렉토리에서 검색되기 전에 EEE 구성의 다른 노드에서 수행될 가능성이 있기 때문에 성능이 저하됩니다. 저장 프로시듀어를 호출할 때 DB2의 기본 작동은 시스템 카탈로그의 저장 프로시듀어에 대한 공유 라이브러리의 이름을 찾아보기 전에 *sqllib* 서브디렉토리의 *function* 서브디렉토리, *sqllib* 서브디렉토리의 *function* 서브디렉토리에 있는 *unfenced* 서브디렉토리를 검색하는 것입니다. PARAMETER TYPE DB2DARI의 저장 프로시듀어만 공유 라이브러리와 동일한 이름을 가질 수 있으므로, DB2DARI 저장 프로시듀어만 DB2의 기본 작동으로부터 이점을 얻습니다. 다른 PARAMETER TYPE으로 카탈로그화된 저장 프로시듀어를 사용하는 경우, DB2가 위의 디렉토리를 검색하는 데 걸리는 시간은 이러한 저장 프로시듀어의 성능을 저하시킵니다. PARAMETER TYPE DB2DARI로 카탈로그화되지 않은 저장 프로시듀어의 성능을 향상시키려면 DB2_STPROC_LOOKUP_FIRST 레지스트리 변수를 ON으로 설정합니다. 이 레지스트리 변수는 위의 디렉토리를 검색하기 전에 DB2가 시스템 카탈로그에 있는 저장 프로시듀어에 대한 공유 라이브러리의 이름을 검색하게 합니다.		

표 29. 데이터 링크 변수

변수 이름	운영 체제	값
설명		
DLFM_BACKUP_DIR_NAME	AIX, Windows NT	기본값: 널(NULL) 값: TSM 또는 모든 유효 경로

표 29. 데이터 링크 변수 (계속)

변수 이름	운영 체제	값
설명		
사용할 백업 장치를 지정합니다. 런타임 시 TSM과 경로 사이에 이 레지스트리 변수 설정을 변경하면 아카이브된 파일은 이동되지 않습니다. 새 백업만 새 위치에 놓입니다. 이전에 아카이브된 파일은 이동되지 않습니다.		
DLFM_BACKUP_LOCAL_MP	AIX, Windows NT	기본값: 널(NULL) 값: DFS 시스템에서 지역 마운트 지점에 대한 유효한 경로
DFS 시스템에서 마운트 지점에 대한 완전한 경로를 지정합니다. 경로를 제공하면 DLFM_BACKUP_DIR_NAME에서 제공한 경로 대신 그 경로가 사용됩니다.		
DLFM_BACKUP_TARGET	AIX, Windows NT	기본값: 널(NULL) 값: LOCAL, TSM, XBSA
사용된 백업의 유형을 지정합니다.		
DLFM_BACKUP_TARGET_LIBRARY	AIX, Windows NT	기본값: 널(NULL) 값: DLL 또는 공유 라이브러리 이름에 대한 유효한 경로
DLL 또는 공유 라이브러리에 대한 완전한 경로를 지정합니다. 이 라이브러리는 *libdfmxbsa.a* 라이브러리를 사용하여 로드됩니다.		
DLFM_ENABLE_STPROC	AIX, Windows NT	기본값: NO 값: YES 또는 NO
저장 프로시듀어가 파일 그룹을 링크하는 데 사용되는지 여부를 지정합니다.		
DLFM_FS_ENVIRONMENT	AIX, Windows NT	기본값: NATIVE 값: NATIVE 또는 DFS
데이터 링크 서버가 작동되는 환경을 지정합니다. NATIVE는 데이터 링크 서버가 자체의 고유한 머신에 있는 파일을 인계할 수 있는 단일 머신 상황에 있음을 나타냅니다. DFS는 데이터 링크 서버가 파일 시스템을 통해 파일을 인계할 수 있는 분산 파일 시스템(DFS) 환경에 있음을 나타냅니다. DFS 파일 세트와 원시 파일 시스템을 혼합할 수는 없습니다.		
DLFM_GC_MODE	AIX, Windows NT	기본값: PASSIVE 값: SLEEP, PASSIVE 또는 ACTIVE
데이터 링크 서버의 불필요 정보 파일 콜렉션의 제어를 제공합니다. SLEEP로 설정하면 불필요 정보 콜렉션은 발생하지 않습니다. PASSIVE로 설정하면 불필요 정보 콜렉션은 다른 트랜잭션이 수행되지 않을 경우에만 수행됩니다. ACTIVE로 설정하면 불필요 정보 콜렉션은 다른 트랜잭션이 수행 중일 경우에만 수행됩니다.		

표 29. 데이터 링크 변수 (계속)

변수 이름 설명	운영 체제	값
DLFM_INSTALL_PATH 데이터 링크 실행 파일이 설치되어 있는 경로를 지정합니다.	AIX, Windows NT	기본값 AIX의 경우: /usr/lpp/ db2_06_00 /adm NT의 경우: DB2PATH /bin 범위: 유효한 모든 정보
DLFM_LOG_LEVEL 기록될 진단 정보 레벨을 지정합니다.	AIX, Windows NT	기본값: LOG_INFO 값: LOG_CRIT, LOG_DEBUG, LOG_ERR, LOG_INFO, LOG_NOTICE, LOG_WARNING
DLFM_PORT DB2 Data Links Manager를 수행하는 데이터 링크 서버와 통신하는 데 사용되는 포트 번호를 지정합니다. 이 환경 변수는 테이블에『DATALINKS』컬럼이 포함되어 있을 경우에만 사용됩니다.	Windows 3.n을 제외한 모든 운영 체제	기본값: 50100 값: 유효한 모든 포트 번호
DLFM_TSM_MGMTCLASS 링크 파일을 아카이브하고 검색하는 데 사용할 TSM 관리 클래스를 지정합니다. 이 변수에 대해 설정된 값이 없으면 기본 TSM 관리 클래스가 사용됩니다.	AIX, Windows NT, Solaris	기본값: 기본 TSM 관리 클래스 값: 유효한 모든 TSM 관리 클래스

표 30. 기타 변수

변수 이름 설명	운영 체제	값
DB2ADMINSERVER DB2 인스턴스가 DB2 관리 서버(DAS)로 설정되도록 지정합니다.	OS/2, Windows 95, Windows NT 및 UNIX	기본값=널(NULL)
DB2CLIINIPATH DB2 CLI/ODBC 구성 파일(db2cli.ini)의 기본 경로를 겹쳐쓰고, 클라이언트의 다른 위치를 지정할 때 사용됩니다. 지정된 값은 반드시 클라이언트 시스템의 유효한 경로에 있어야 합니다.	모두	기본값=널(NULL)

표 30. 기타 변수 (계속)

변수 이름 설명	운영 체제	값
DB2DEFPREP	모두	기본값=NO 값: ALL, YES 또는 NO
이 옵션이 사용 가능해지기 전에 사전 처리 컴파일된 응용프로그램에 대한 DEFERRED_PREPARE 사전 컴파일 옵션의 런타임 조작을 시뮬레이트합니다. 예를 들어, DB2 v2.1.1 또는 이전 응용프로그램이 DB2 v2.1.2 또는 이후 환경에서 수행 중이면 바람직한 '연기된 준비' 작동을 가리키기 위해 DB2DEFPREP를 사용할 수 있습니다.		
DB2_DJ_COMM	모두	기본값=널(NULL) 값에는 libdrda.a, libsqlnet.a, libnet8.a, libdrda.dll, libsqlnet.dll, libnet8.dll 등이 포함됩니다.
데이터베이스 관리 프로그램이 시작될 때 로드된 랩퍼 라이브러리를 지정합니다. 이 변수를 지정하면 자주 사용되는 랩퍼의 런타임 로드 비용을 줄일 수 있습니다. 다른 운영 체제에는 다른 값이 지원됩니다. (.dll 확장자는 Windows NT 운영 체제에 사용되며, .a 확장자는 AIX 운영 체제에 사용됩니다.) 라이브러리 이름은 프로토콜 및 운영 체제에 따라 다릅니다. 이 변수는 데이터베이스 관리 프로그램 매개변수 *federated*가 YES로 설정되어 있지 않으면 사용 가능하지 않습니다.		
DB2DMNBCKCTLR	Windows NT	기본값=널(NULL) 값: ? 또는 도메인 이름
DB2 서버가 백업 도메인 제어기인 도메인의 이름을 알고 있는 경우, DB2DMNBCKCTLR=DOMAIN_NAME으로 설정합니다. DOMAIN_NAME은 대문자여야 합니다. DB2가 어떤 지역 머신에 대한 도메인이 백업 도메인 제어기인지를 판별하려면 DB2DMNBCKCTLR=?로 설정하십시오. DB2DMNBCKCTLR 프로파일 변수가 설정되지 않거나 또는 공백으로 설정되면 DB2가 1차 도메인 제어기에서 인증을 수행합니다. **주:** DB2는 기본적으로 기존의 백업 도메인 제어기를 사용하지 않습니다. 왜냐하면, 백업 도메인 제어기는 보안을 노출시키면서 1차 도메인 제어기와의 동기화에서 벗어날 수 있기 때문입니다. 1차 도메인 제어기의 보안 데이터베이스가 갱신될 때 동기화에서 벗어나게 되지만, 이 변경사항은 백업 도메인 제어기로 전달되지 않습니다. 이는 네트워크 대기가 있거나 또는 컴퓨터 브라우저 서비스가 작동하지 않을 때 발생합니다.		
DB2_ENABLE_LDAP	모두	기본값=NO 값: YES 또는 NO
LDAP(Lightweight Directory Access Protocol)의 사용 여부를 지정합니다. LDAP는 디렉토리 서비스에 대한 액세스 메소드입니다.		
DB2_FALLBACK	Windows NT	기본값=OFF 값: ON 또는 OFF

표 30. 기타 변수 (계속)

변수 이름	운영 체제	값
설명		
이 변수를 사용하여 모든 데이터베이스 연결이 폴백 처리 중에 강제로 연결이 해제되도록 할 수 있습니다. Microsoft Cluster Server(MSCS)가 있는 Windows NT 환경에서 failover 지원과 함께 사용됩니다. DB2_FALLBACK을 설정하지 않거나 OFF로 설정하고 데이터베이스 연결이 폴백 중에 존재할 경우, DB2 자원은 오프라인으로 가져올 수 없습니다. 다시 말해서, 폴백 처리에 실패합니다.		
DB2_FORCE_TRUNCATION	모두	기본값=NO 값: YES 또는 NO
복구 재시작 중에 사용됩니다. 『NO』로 설정하면 불량 페이지가 너무 빨리 복구 재시작을 중지시킨 경우 복구 재시작을 중단합니다(즉, 사용 중인 모든 로그가 읽혀지지 않음). 이러한 상황은 보통 로그 중 하나에 불량 페이지가 있을 경우에 발생합니다. 사용자는 이 변수를 『YES』로 설정하여 로그 끝에 도달한 것처럼 처리를 계속해야 하는 복구 재시작 신호를 보낼 수 있습니다. 변수를 『YES』로 설정한 후에는, 로그는 데이터베이스가 다시 사용 중 상태가 될 때 복구 재시작 중에 읽혀지지 않은 로그가 겹쳐쓰입니다. 기본값은 『NO』이며, 이는 불량 페이지를 찾을 수 없는 경우 계속되지 **않습니다**. IBM 서비스 요원의 지시하에서만 이 변수를 사용하십시오.		
DB2_GRP_LOOKUP	Windows NT	기본값=널(NULL) 값: LOCAL, DOMAIN
이 변수는 DB2에 사용자 계정의 유효성을 확인하고 그룹 구성원을 찾을 위치를 알려주는 데 사용됩니다. DB2가 항상 강제로 그룹을 나열하고 DB2 서버에 있는 사용자 계정의 유효성을 확인하도록 하려면 변수를 LOCAL로 설정하십시오. DB2가 항상 강제로 그룹을 나열하고 사용자 계정이 속하는 Windows NT 도메인에 있는 사용자 계정의 유효성을 확인하도록 하려면 변수를 DOMAIN으로 설정하십시오.		
DB2_INDEX_2BYTEVARLEN	모두	기본값=NO 값: YES 또는 NO
이 레지스트리 변수를 사용하여 색인 키의 부분으로 255바이트보다 긴 컬럼을 지정할 수 있습니다. 이 레지스트리 변수를 YES로 설정하기 전에 이미 작성된 색인은 계속 255바이트 제한을 받습니다. 이 레지스트리 변수를 ″YES″로 조정한 후 작성된 색인은 레지스트리 변수가 다시 ″NO″로 설정될 때도 2바이트 색인처럼 작동합니다. CREATE TABLE, CREATE INDEX 및 ALTER TABLE을 포함하여 이 레지스트리 변수를 변경하면 여러 가지 SQL문이 영향을 받습니다. 이들 명령문에 대한 자세한 내용은 *SQL 참조서*의 문서화된 변경사항을 참조하십시오.		
DB2LDAP_BASEDN	모두	기본값=널(NULL) 값: 유효한 기본 도메인 이름
LDAP 디렉토리의 기본 도메인 이름을 지정합니다.		
DB2LDAPCACHE	모두	기본값=YES 값: YES 또는 NO

표 30. 기타 변수 (계속)

변수 이름	운영 체제	값
설명		

LDAP 캐쉬가 사용되도록 지정합니다. 이 캐쉬는 데이터베이스, 노드 및 DCS 디렉토리를 지역 머신을 카탈로그화하는 데 사용됩니다.
 	
캐쉬의 최신 항목이 있는지 확인하려면 다음을 수행하십시오.
`REFRESH LDAP DB DIR`
`REFRESH LDAP NODE DIR`
 	
이들 명령은 데이터베이스 디렉토리와 노드 디렉토리에서 잘못된 항목을 갱신하고 제거합니다.		

DB2LDAP_CLIENT_PROVIDER	Windows 95/98/NT/2000 전용	기본값=널(NULL)(사용할 수 있으면 Microsoft가 사용되며, 그렇지 않으면 IBM이 사용됩니다.) 값: IBM 또는 Microsoft
Windows 환경에서 수행될 경우, DB2는 Microsoft LDAP 클라이언트나 IBM LDAP 클라이언트를 사용하여 LDAP 디렉토리에 액세스하는 것을 지원합니다. 이 레지스트리 변수는 DB2에서 사용될 LDAP 클라이언트를 명시적으로 선택하는 데 사용됩니다. **주:** 이 레지스트리 변수의 현재 값을 표시하려면 `db2set` 명령을 사용하십시오. `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	모두	기본값=널(NULL) 값: 유효한 모든 호스트 이름
LDAP 디렉토리의 위치 호스트 이름을 지정합니다.		
DB2LDAP_SEARCH_SCOPE	모두	기본값= DOMAIN 값: LOCAL, DOMAIN, GLOBAL
LDAP(Lightweight Directory Access Protocol)의 파티션이나 도메인에서 발견된 정보에 대한 검색 범위를 지정합니다. 『LOCAL』은 LDAP 디렉토리에서 탐색이 이루어지지 않게 합니다. 『DOMAIN』은 LDAP에서만 현재 디렉토리 파티션을 검색합니다. 『GLOBAL』은 오브젝트를 찾을 때까지 LDAP 내의 모든 디렉토리 파티션을 검색합니다.		
DB2LOADREC	모두	기본값=널(NULL)
롤 포워드 동안 로드 사본의 위치 겹쳐쓰기에 사용됩니다. 사용자가 로드 사본의 실제 위치를 변경한 경우, 롤 포워드를 발행하기 전에 DB2LOADREC를 설정해야 합니다.		
DB2LOCK_TO_RB	모두	기본값=널(NULL) 값: 명령문

표 30. 기타 변수 (계속)

변수 이름	운영 체제	값
설명		
잠금 시간종료가 전체 트랜잭션을 구간 복원시키는지, 아니면 현재 명령문만 구간 복원시키는지 지정합니다. DB2LOCK_TO_RB가 STATEMENT로 설정된 경우, 잠금 시간종료는 현재 명령문만을 구간 복원되게 합니다. 기타 설정은 결과적으로 트랜잭션을 구간 복원되게 합니다.		
DB2_NEWLOGPATH2	UNIX	기본값=0 값: 0 또는 1
이 매개변수를 사용하여 이중 로깅을 구현하는 데 사용해야 할 2차 경로를 지정할 수 있습니다. 사용된 경로는 *logpath* 데이터베이스 구성 매개변수의 현재 값에 『2』를 추가하여 생성됩니다.		
DB2NOEXITLIST	모두	기본값=OFF 값: ON 또는 OFF
정의된 경우, 이 변수는 DB2가 응용프로그램에 나감 목록 핸들러를 설치하지 않고 COMMIT를 수행하지 않음을 나타냅니다. 일반적으로, DB2는 응용프로그램에서 프로세스 나감 목록 핸들러를 설치하고 응용프로그램이 이상 종료될 경우 나감 목록 핸들러가 COMMIT 조작을 수행합니다. 동적으로 DB2 라이브러리를 로드하고 응용프로그램이 종료되기 전에 이를 언로드하는 응용프로그램의 경우, 나감 목록 핸들러 호출에 실패합니다. 핸들러 루틴이 더 이상 응용프로그램에서 로드되지 않기 때문입니다. 사용자의 응용프로그램이 이러한 방식으로 작동될 경우, DB2NOEXITLIST 변수를 설정하고 응용프로그램이 명시적으로 필요한 모든 COMMIT를 호출하도록 해야 합니다.		
DB2REMOTEPREG	Windows 95 및 Windows NT	기본값=널(NULL) 값: 유효한 모든 Windows 95 또는 Windows NT 시스템 이름
DB2 인스턴스 프로파일 및 DB2 인스턴스의 Win32 레지스트리 목록을 포함하는 원격 머신 이름을 지정합니다. DB2REMOTEPREG 값은 DB2가 설치된 후에만 설정해야 하고 수정해서는 안됩니다. 주의깊게 이 변수를 사용하십시오.		
DB2ROUTINE_DEBUG	AIX 및 Windows NT	기본값=OFF 값: ON, OFF
Java 저장 프로시듀어에 대한 디버그 기능을 작동 가능하게 할지 여부를 지정합니다. Java 저장 프로시듀어를 디버그하지 않을 경우에는 기본값(OFF)을 사용하십시오. 디버그를 작동 가능하게 하면 성능에 영향을 미칩니다. Java 저장 프로시듀어에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.		
DB2SORCVBUF	Windows 95 및 Windows NT	기본값=32767
Windows 95 및 Windows NT 운영 체제에 버퍼를 수신하는 TCP/IP 값을 지정합니다.		
DB2SORT	모두, 서버 전용	기본값=널(NULL)

표 30. 기타 변수 (계속)

변수 이름	운영 체제	값
설명		
LOAD 유틸리티에 의해 런타임 시 로드될 라이브러리의 위치를 지정합니다. 라이브러리는 색인화 데이터 정렬에 사용된 함수의 시작점을 포함합니다. DB2SORT를 사용하여 테이블 색인을 생성할 때 LOAD 유틸리티를 사용하는 벤더 제공 정렬 제품을 활용하십시오. 제공된 경로는 데이터베이스 서버에 상대적이어야 합니다.		
DB2SYSTEM	Windows NT, Windows 95, OS/2 및 UNIX	기본값=널(NULL)
사용자가 사용하는 이름과 DB2 서버 시스템을 식별하는 데이터베이스 관리자를 지정합니다. 가능하면 네트워크상에서 고유한 이름이어야 합니다. 제어 센터에서 관리되는 서버 시스템을 관리자가 식별하는 것을 돕기 위해, 제어 센터의 오브젝트 트리의 시스템 레벨로 이름을 표시합니다. 클라이언트 구성 지원 프로그램의 '네트워크 검색' 기능을 사용할 때 DB2 발견은 이 이름을 리턴하여 결과 오브젝트 트리에 시스템 레벨로 표시합니다. 이 이름은 사용자가 액세스하려는 데이터베이스가 들어 있는 시스템 식별을 도와줍니다. DB2SYSTEM 값은 다음과 같이 설치시에 설정됩니다. • Windows NT 또는 Windows 95에서 설치 프로그램은 Windows 시스템용으로 지정된 컴퓨터 이름과 동일하게 설정합니다. • OS/2에서는 설치 처리 중에 DB2SYSTEM 이름을 입력하도록 프롬프트를 표시합니다. • UNIX 시스템에서는 UNIX 시스템의 TCP/IP 호스트 이름과 동일하게 설정됩니다.		
DB2UPMPR	OS/2	기본값=ON 값: ON 또는 OFF
사용자가 틀린 사용자 ID 또는 암호를 OS/2에 입력할 때 UPM 로그온 화면이 표시되는지 여부를 지정합니다.		
DB2_VENDOR_INI	AIX, HP-UX, Sun Solaris 및 Windows NT	기본값=널(NULL) 값: 유효한 모든 경로 및 파일
모든 벤더별 환경 설정을 포함하는 파일을 가리킵니다. 데이터베이스 관리 프로그램이 시작할 때 값이 선택됩니다.		
DB2_XBSA_LIBRARY	AIX, HP-UX, Sun Solaris 및 Windows NT	기본값=널(NULL) 값: 유효한 모든 경로 및 파일

표 30. 기타 변수 (계속)

변수 이름	운영 체제	값
설명		

벤더 공급 XBSA 라이브러리를 가리킵니다. AIX에서 설정은 오브젝트의 이름이 shr.o가 아닌 경우 공유 오브젝트를 포함해야 합니다. HP-UX, Sun Solaris 및 Windows NT에는 공유 오브젝트 이름이 필요하지 않습니다. 예를 들어, DB2용 Legato의 NetWorker Business Suite Module을 사용하려면 레지스트리 변수를 다음과 같이 설정해야 합니다.
`db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"`
XBSA 인터페이스는 BACKUP DATABASE 또는 RESTORE DATABASE 명령을 통해 호출될 수 있습니다. 예를 들면, 다음과 같습니다.
`db2 backup db sample use XBSA`
`db2 restore db sample use XBSA`		

부록B. Explain 테이블 및 정의

Explain 테이블은 Explain 기능이 활성화될 때 액세스 플랜을 캡쳐합니다. 이 절에서는 다음 Explain 테이블 및 정의를 다룹니다.

Explain을 호출하려면 Explain 테이블을 먼저 작성해야 합니다. 이를 작성하려면 'sqllib' 디렉토리의 'misc' 서브디렉토리에 있는 EXPLAIN.DDL 파일에 제공된 샘플 명령행 처리기 입력 스크립트를 사용하십시오. Explain 테이블이 필요한 데이터베이스에 연결하십시오. 그런 다음 `db2 -tf EXPLAIN.DDL` 명령을 발행하면 이 테이블이 작성됩니다. 자세한 내용은 632 페이지의『Explain 테이블에 대한 테이블 정의』를 참조하십시오.

Explain 기능으로 Explain 테이블을 채운다고 해서 트리거가 활성화되거나 참조 또는 점검 제한조건이 활성화되지는 않습니다. 예를 들어, 삽입 트리거가 EXPLAIN_INSTANCE 테이블에 정의되어 있고 적절한 명령문이 설명된 경우, 트리거는 활성화되지 않습니다.

Explain 기능에 대한 자세한 내용은 243 페이지의『제7장 SQL Explain 기능』을 참조하십시오.

© Copyright IBM Corp. 1993, 2001

Explain 테이블 설명:

표제	설명
컬럼 이름	컬럼의 이름
데이터 유형	컬럼의 데이터 유형
널(NULL) 입력 가능 여부	예: 널(NULL)이 허용됨 아니오: 널(NULL)이 허용되지 않음
키?	PK: 컬럼이 기본 키의 일부임 FK: 컬럼이 외부 키의 일부임
설명	컬럼에 대한 설명

EXPLAIN_ARGUMENT 테이블

EXPLAIN_ARGUMENT 테이블이 있는 경우, 개별 연산자에 대한 고유 특성을 표시합니다.

표 31. EXPLAIN_ARGUMENT 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 한가?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	Explain 요청 개시 프로그램 권한부여 ID
EXPLAIN_TIME	TIMESTAMP	아니오	FK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 원시 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행에 적절한 설명 정보 레벨
STMTNO	INTEGER	아니오	FK	Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	INTEGER	아니오	FK	Explain 정보와 관련된 패키지 내의 절 번호
OPERATOR_ID	INTEGER	아니오	아니오	조회 내에서 연산자의 고유 ID
ARGUMENT_TYPE	CHAR(8)	아니오	아니오	연산자에 대한 인수 유형
ARGUMENT_VALUE	VARCHAR(1024)	예	아니오	연산자에 대한 인수 값. 값이 LONG_ARGUMENT_VALUE 내에 있을 경우, 널(NULL)이 적용됩니다.
LONG_ARGUMENT_VALUE	CLOB(1M)	예	아니오	이 연산자에 대한 인수 값으로, 텍스트가 ARGUMENT_VALUE에 맞지 않을 경우에 사용됩니다. 값이 ARGUMENT_VALUE 내에 있을 경우, 널(NULL)이 적용됩니다.

표 32. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
AGGMODE	완료 부분 중간 최종	부분 총계 표시기
BITFLTR	참 거짓	해쉬 조인은 비트 필터를 사용하여 성능을 향상시킵니다.
CSETEMP	참 거짓	일반 부속 표현 플래그를 통한 임시 테이블
DIRECT	참	직접 페치 표시기
DUPLWARN	참 거짓	중복 경고 플래그
EARLY OUT	참 거짓	이른 출력 표시기
ENVVAR	이 유형의 각 행에는 다음이 포함됩니다. • 환경 변수 이름 • 환경 변수 값	최적화 알고리즘에 영향을 미치는 환경 변수
FETCHMAX	무시 정수	FETCH 연산자에 MAXPAGES 인수로 겹쳐쓴 값
GROUPBYC	참 거짓	Group By 컬럼이 제공되었는지 여부
GROUPBYN	정수	비교 컬럼의 수
GROUPBYR	이 유형의 각 행에는 다음이 포함됩니다. • Group By 절에 있는 컬럼의 서수 값(콜론 및 공간이 뒤에 옴) • 컬럼 이름	Group By 요구조건
INNERCOL	이 유형의 각 행에는 다음이 포함됩니다. • 순서대로 이루어진 컬럼의 서수 값(콜론 및 공백이 뒤에 옴) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	내부 순서 컬럼
ISCANMAX	무시 정수	ISCAN 연산자에 MAXPAGES 인수로 겹쳐쓴 값
JN_INPUT	내부 외부	연산자가 조인의 내부 또는 외부를 공급하는 연산자인지 표시합니다.

표 32. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
LISTENER	참 거짓	리스너 테이블 대기행렬 표시기
MAXPAGES	모두 없음 정수	프리페치에 예상할 수 있는 최대 페이지
MAXRIDS	없음 정수	각 목록 프리페치 요청에 포함될 최대 행 ID
NUMROWS	정수	정렬이 예상된 행의 수
ONEFETCH	참 거짓	한 개의 페치 표시기
OUTERCOL	이 유형의 각 행에는 다음이 포함됩니다. • 순서대로 이루어진 컬럼의 서수 값(콜론 및 공백이 뒤에 옴) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	외부 순서 컬럼
OUTERJN	왼쪽 오른쪽	외부 조인 표시기
PARTCOLS	컬럼 이름	연산자의 파티션 컬럼
PREFETCH	목록 없음 순차	적당한 프리페치 유형
RMTQTEXT	조회 텍스트	원격 조회 텍스트
ROWLOCK	독점 없음 재사용 공유 짧은(순간) 공유 갱신	행 잠금 모드
ROWWIDTH	정수	정렬될 행의 너비
SCANDIR	앞으로 역방향	스캔 방향
SCANGRAN	정수	SCANUNIT로 표현된 파티션 내 병렬 처리, 파티션 내 병렬 스캔의 세분성
SCANTYPE	지역 병렬	파티션 내 병렬 처리, 색인 또는 테이블 스캔

표 32. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
SCANUNIT	행 페이지	파티션 내 병렬 처리, 스캔 세분성 단위
SERVER	원격 서버	원격 서버
SHARED	참	파티션 내 병렬 처리 공유, TEMP 표시기
SLOWMAT	참 거짓	느린 구체화 플래그
SNGLPROD	참 거짓	단일 에이전트에 의해 생성되는 파티션 내 병렬 처리 정렬 또는 temp
SORTKEY	이 유형의 각 행에는 다음이 포함됩니다. • 키에 있는 컬럼의 서수 값(콜론 및 공백이 뒤에 옴) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	정렬 키 컬럼
SORTTYPE	파티션됨 공유 라운드 로빈 복제됨	파티션 내 병렬 처리, 정렬 유형
TABLOCK	독점 부분 독점 잠금 없음 부분 공유 재사항 공유 부분 독점이 있는 공유 강한 독점 갱신	테이블 잠금 모드
TQDEGREE	정수	파티션 내 병렬 처리, 테이블 대기행렬에 액세스하는 서브에이전트의 수
TQMERGE	참 거짓	병합(정렬된) 테이블 대기행렬 표시기
TQREAD	선행 읽기 단계 지정 부속 조회 단계 지정	테이블 대기행렬의 읽기 등록 정보
TQSEND	브로드캐스트 방향지정 분산 부속 조회 방향지정	테이블 대기행렬 전송 등록 정보

표 32. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
TQTYPE	지역	파티션 내 병렬 처리, 테이블 대기행렬
TRUNCSRT	참	절단 정렬(생성되는 행의 수를 제한함)
UNIQUE	참 거짓	고유성 표시기
UNIQKEY	이 유형의 각 행에는 다음이 포함됩니다. • 키에 있는 컬럼의 서수 값(콜론 및 공백이 뒤에 옴) • 컬럼 이름	고유 키 컬럼
VOLATILE	참	소멸성 테이블

EXPLAIN_INSTANCE 테이블

EXPLAIN_INSTANCE 테이블은 모든 Explain 정보에 대한 주요 제어 테이블입니다. Explain 테이블 내 데이터의 각 행은 명시적으로 이 테이블에 있는 하나의 고유 행에 링크됩니다. EXPLAIN_INSTANCE는 설명이 수행되는 환경에 대한 정보와 함께 설명되고 있는 SQL문의 소스에 대한 기본 정보를 제공합니다.

이 테이블 정의에 대한 자세한 내용은 634 페이지의 『EXPLAIN_INSTANCE 테이블 정의』를 참조하십시오.

표 33. EXPLAIN_INSTANCE 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	PK	Explain 요청 개시 프로그램의 권한 부여 ID
EXPLAIN_TIME	시간소인	아니오	PK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	PK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	PK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_OPTION	CHAR(1)	아니오	아니오	이 요청에 대해 요청된 Explain 정보를 나타냅니다. 가능한 값은 다음과 같습니다. **P** 플랜 선택

표 33. EXPLAIN_INSTANCE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
SNAPSHOT_TAKEN	CHAR(1)	아니오	아니오	이 요청에 대해 Explain 스냅샷이 생성되었는지를 나타냅니다. 가능한 값은 다음과 같습니다. **Y** 예. Explain 스냅샷이 생성되어 EXPLAIN_STATEMENT 테이블에저장되었습니다. 또한 일반 Explain 정보도 캡쳐되었습니다. **N** Explain 스냅샷이 생성되지 않았습니다. 일반 Explain 정보가 캡쳐되었습니다. **O** Explain 스냅샷만 생성되었습니다. 일반 Explain 정보가 캡쳐되지 않았습니다.
DB2_VERSION	CHAR(7)	아니오	아니오	Explain 요청을 처리한 DB2 Universal Database에 대한 제품 릴리스 번호. 형식은 vv.rr.m이며, 다음이 적용됩니다. **vv** 버전 번호 **rr** 릴리스 번호 **m** 유지보수 릴리스 번호
SQL_TYPE	CHAR(1)	아니오	아니오	Explain 인스턴스가 정적 SQL인지, 동적 SQL인지를 나타냅니다. 가능한 값은 다음과 같습니다. **S** 정적 SQL **D** 동적 SQL
QUERYOPT	정수	아니오	아니오	Explain 호출시 SQL 컴파일러에서 사용된 조회 최적화 클래스를 나타냅니다. 이 값은 설명 중인 SQL문에 대해 SQL 컴파일러가 수행한 조회 최적화의 레벨을 나타냅니다.
BLOCK	CHAR(1)	아니오	아니오	SQL문 컴파일시 사용된 커서 블로킹의 유형을 나타냅니다. 자세한 내용은 SYSCAT.PACKAGES의 BLOCK 컬럼을 참조하십시오. 가능한 값은 다음과 같습니다. **N** 블로킹 없음 **U** 명확한 커서 블록화 **B** 전체 커서 블록화

표 33. EXPLAIN_INSTANCE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
ISOLATION	CHAR(2)	아니오	아니오	SQL문 컴파일시 사용된 분리 유형을 나타냅니다. 자세한 내용은 SYSCAT.PACKAGES의 ISOLATION 컬럼을 참조하십시오. 가능한 값은 다음과 같습니다. **RR** 반복 읽기 **RS** 읽기 안정성 **CS** 커서 안정성 **UR** 미확약 읽기
BUFFPAGE	정수	아니오	아니오	Explain 호출시 BUFFPAGE 데이터베이스 구성 설정의 값을 포함합니다.
AVG_APPLS	정수	아니오	아니오	Explain 호출시 AVG_APPLS 구성 매개변수의 값을 포함합니다.
SORTHEAP	정수	아니오	아니오	Explain 호출시 SORTHEAP 데이터베이스 구성 설정의 값을 포함합니다.
LOCKLIST	정수	아니오	아니오	Explain 호출시 LOCKLIST 데이터베이스 구성 설정의 값을 포함합니다.
MAXLOCKS	SMALLINT	아니오	아니오	Explain 호출시 MAXLOCKS 데이터베이스 구성 설정의 값을 포함합니다.
LOCKS_AVAIL	정수	아니오	아니오	사용자에 대해 최적화 알고리즘으로 사용 가능하도록 가정된 잠긴 번호를 포함하십시오(LOCKLIST 및 MAXLOCKS로부터 파생됨).
CPU_SPEED	DOUBLE	아니오	아니오	Explain 호출시 CPUSPEED 데이터베이스 관리 프로그램 구성 설정의 값을 포함합니다.
REMARKS	VARCHAR(254)	예	아니오	사용자 제공 주석
DBHEAP	정수	아니오	아니오	Explain 호출시 DBHEAP 데이터베이스 구성 설정의 값을 포함합니다.
COMM_SPEED	DOUBLE	아니오	아니오	Explain 호출시 COMM_BANDWIDTH 데이터베이스 구성 설정의 값을 포함합니다.
PARALLELISM	CHAR(2)	아니오	아니오	가능한 값은 다음과 같습니다. • N = 병렬 처리 없음 • P = 파티션 내 병렬 처리 • IP = 파티션간 병렬 처리 • BP = 파티션 내 병렬 처리 및 파티션간 병렬 처리

표 33. EXPLAIN_INSTANCE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
DATAJOINER	CHAR(1)	아니오	아니오	가능한 값은 다음과 같습니다. • N = 비연합 시스템 플랜 • Y = 연합 시스템 플랜

EXPLAIN_OBJECT 테이블

EXPLAIN_OBJECT 테이블은 SQL문을 충족시키기 위해 생성된 액세스 플랜에 필요한 데이터 오브젝트를 식별합니다.

표 34. EXPLAIN_OBJECT 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	Explain 요청 개시 프로그램 권한 부여 ID
EXPLAIN_TIME	시간소인	아니오	FK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행에 적절한 Explain 정보 레벨
STMTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 섹션 번호
OBJECT_SCHEMA	VARCHAR(128)	아니오	아니오	오브젝트가 속하는 스키마
OBJECT_NAME	VARCHAR(128)	아니오	아니오	오브젝트 이름
OBJECT_TYPE	CHAR(2)	아니오	아니오	해당 유형의 오브젝트에 대한 설명 레이블
CREATE_TIME	시간소인	예	아니오	오브젝트의 작성 시간. 테이블 함수의 경우, 널(NULL)이 적용됩니다.
STATISTICS_TIME	시간소인	예	아니오	오브젝트에 대한 통계가 갱신된 마지막 시간. 오브젝트에 대한 통계가 없을 경우, 널(NULL)이 적용됩니다.
COLUMN_COUNT	SMALLINT	아니오	아니오	오브젝트의 컬럼 수
ROW_COUNT	정수	아니오	아니오	오브젝트의 추정된 행 수
WIDTH	정수	아니오	아니오	오브젝트의 평균 너비(바이트 단위). 색인의 경우, -1로 설정됩니다.
PAGES	정수	아니오	아니오	오브젝트가 버퍼 풀에서 차지하는 추정된 페이지 수. 테이블 함수의 경우, -1로 설정됩니다.

표 34. EXPLAIN_OBJECT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
DISTINCT	CHAR(1)	아니오	아니오	오브젝트의 행이 구별되는지(중복이 없는지)를 나타냅니다. 가능한 값은 다음과 같습니다. **Y** 예 **N** 아니오
TABLESPACE_NAME	VARCHAR(128)	예	아니오	오브젝트가 저장된 테이블 공간 이름. 관련된 테이블 공간이 없을 경우, 널(NULL)이 적용됩니다.
OVERHEAD	DOUBLE	아니오	아니오	지정된 테이블 공간으로의 단일 무작위 I/O에 대해 추정된 총 오버헤드(밀리초 단위). 여기에는 제어기 오버헤드, 디스크 탐색 및 대기 시간이 포함됩니다. 관련된 테이블 공간이 없을 경우, -1로 설정됩니다.
TRANSFER_RATE	DOUBLE	아니오	아니오	지정된 테이블 공간으로부터 데이터 페이지를 읽는 데 걸리는 추정 시간(밀리초 단위). 관련된 테이블 공간이 없을 경우, -1로 설정됩니다.
PREFETCHSIZE	정수	아니오	아니오	프리페치가 수행될 때 기록될 데이터 페이지의 수. 테이블 함수의 경우, -1로 설정됩니다.
EXTENTSIZE	정수	아니오	아니오	데이터 페이지 단위의 Extent 크기. 다음 컨테이너로 바뀌기 전 테이블 공간의 하나의 컨테이너에 기록되는 많은 페이지가 있습니다. 테이블 함수의 경우, -1로 설정됩니다.
CLUSTER	DOUBLE	아니오	아니오	색인을 사용한 데이터 클러스터링 등급. >= 1이면 CLUSTERRATIO가 됩니다. >= 0이고 < 1이면 CLUSTERFACTOR가 됩니다. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
NLEAF	정수	아니오	아니오	색인 오브젝트의 값이 차지하는 리프 페이지의 수. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
NLEVELS	정수	아니오	아니오	색인 오브젝트 트리의 색인 레벨의 수. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
FULLKEYCARD	BIGINT	아니오	아니오	색인 오브젝트에 포함된 전체 구별 키 값의 수. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
OVERFLOW	정수	아니오	아니오	테이블의 전체 오버플로우 레코드의 수. 색인, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
FIRSTKEYCARD	BIGINT	아니오	아니오	첫 번째 구별 키 값의 수. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.

표 34. EXPLAIN_OBJECT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
FIRST2KEYCARD	BIGINT	아니오	아니오	색인의 첫 번째 {2,3,4} 컬럼을 사용하는 첫 번째 구별 키 값의 수. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
FIRST3KEYCARD	BIGINT	아니오	아니오	
FIRST4KEYCARD	BIGINT	아니오	아니오	
SEQUENTIAL_PAGES	정수	아니오	아니오	서로 간격이 거의 없는 색인 키순으로 디스크에 배치된 리프 페이지 수. 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.
DENSITY	정수	아니오	아니오	색인이 차지한 페이지 범위 내의 페이지 수에 대한 SEQUENTIAL_PAGES 비율로서, 백분율 단위로 표시됩니다(0에서 100 사이의 정수). 테이블, 테이블 함수이거나 통계가 사용 가능하지 않을 경우, -1로 설정됩니다.

표 35. 가능한 OBJECT_TYPE 값

값	설명
IX	색인
TA	테이블
TF	테이블 함수

EXPLAIN_OPERATOR 테이블

EXPLAIN_OPERATOR 테이블에는 SQL문을 충족시키기 위해 SQL 컴파일러에 필요한 모든 연산자가 들어 있습니다.

표 36. EXPLAIN_OPERATOR 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	Explain 요청 개시 프로그램 권한 부여 ID
EXPLAIN_TIME	시간소인	아니오	FK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행에 적절한 Explain 정보 레벨
STMTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 섹션 번호
OPERATOR_ID	정수	아니오	아니오	조회 내에서 연산자의 고유 ID

표 36. EXPLAIN_OPERATOR 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
OPERATOR_TYPE	CHAR(6)	아니오	아니오	연산자 유형에 대한 설명 레이블
TOTAL_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜을 실행하는 데 필요한 (timerons에서) 추정된 누적 총 비용
IO_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜을 실행하는 데 필요한 (데이터 페이지 I/O에서) 추정된 누적 I/O 비용
CPU_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜을 실행하는 데 필요한 (지침에서) 추정된 누적 CPU 비용
FIRST_ROW_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜의 첫 번째 행을 페치하는 데 필요한 (timerons에서) 추정된 누적 비용. 이 값에는 필요한 초기 오버헤드가 포함됩니다.
RE_TOTAL_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜의 다음 행을 페치하는 데 필요한 (timerons에서) 추정된 누적 비용
RE_IO_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜의 다음 행을 페치하는 데 필요한 (데이터 페이지 I/O에서) 추정된 누적 I/O 비용
RE_CPU_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜의 다음 행을 페치하는 데 필요한 (timerons에서) 추정된 누적 CPU 비용
COMM_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜을 실행하는 데 필요한 (TCP/IP 프레임에서) 추정된 누적 통신 비용
FIRST_COMM_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 액세스 플랜의 첫 번째 행을 페치하는 데 필요한 (TCP/IP 프레임에서) 추정된 누적 통신 비용. 이 값에는 필요한 초기 오버헤드가 포함됩니다.
BUFFERS	DOUBLE	아니오	아니오	이 연산자 및 해당 입력에 대한 추정된 버퍼 요구사항
REMOTE_TOTAL_COST	DOUBLE	아니오	아니오	원격 데이터베이스에서 조작을 수행하는 데 필요한 (timerons에서) 추정된 누적 총 비용
REMOTE_COMM_COST	DOUBLE	아니오	아니오	이 연산자를 포함하여 지금까지 선택한 원격 액세스 플랜을 실행하는 데 필요한 추정되는 누적 통신 비용

표 37. OPERATOR_TYPE 값

값	설명
DELETE	삭제
FETCH	페치
FILTER	행 필터

표 37. OPERATOR_TYPE 값 (계속)

값	설명
GENROW	행 생성
GRPBY	Group By
HSJOIN	해쉬 조인
INSERT	삽입
IXAND	동적 비트맵 색인 AND 작업
IXSCAN	색인 스캔
MSJOIN	병합 스캔 조인
NLJOIN	중첩 루트 조인
RETURN	결과
RIDSCN	행 ID(RID) 스캔
RQUERY	원격 조회
SORT	정렬
TBSCAN	테이블 스캔
TEMP	임시 테이블 구조
TQ	테이블 대기행렬
UNION	결합
UNIQUE	중복 제거
UPDATE	갱신

EXPLAIN_PREDICATE 테이블

EXPLAIN_PREDICATE 테이블은 어떤 술어가 특정 연산자에 의해 적용되는지를 식별합니다.

표 38. EXPLAIN_PREDICATE 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	Explain 요청 개시 프로그램 권한 부여 ID
EXPLAIN_TIME	시간소인	아니오	FK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행에 적절한 Explain 정보 레벨
STMTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 섹션 번호

표 38. EXPLAIN_PREDICATE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
OPERATOR_ID	정수	아니오	아니오	조회 내에서 연산자의 고유 ID
PREDICATE_ID	정수	아니오	아니오	지정된 연산자에 대한 술어의 고유 ID
HOW_APPLIED	CHAR(5)	아니오	아니오	술어가 지정된 연산자에 의해 사용되는 방식
WHEN_EVALUATED	CHAR(3)	아니오	아니오	술어에 사용되는 부속 조회가 평가되는 시기를 나타냅니다. 가능한 값은 다음과 같습니다. **blank** 술어에는 부속 조회가 없습니다. **EAA** 술어에 사용되는 부속 조회가 응용프로그램(EAA)에서 평가됩니다. 즉, 술어가 적용 중일 때 지정된 연산자가 처리하는 모든 행에 대해 재평가됩니다. **EAO** 술어에 사용되는 부속 조회는 열린 EAO에서 평가됩니다. 즉, 지정된 연산자에 대해 한 번씩만 재평가되며, 그 결과는 각 행에 대한 술어의 응용프로그램에서 재사용됩니다. **MUL** 술어에는 1바이트가 넘는 부속 조회가 있습니다.
RELOP_TYPE	CHAR(2)	아니오	아니오	술어에 사용되는 관계형 연산자의 유형
SUBQUERY	CHAR(1)	아니오	아니오	이 술어를 사용하기 위해 부속 조회로부터 데이터 스트림이 필요한지 여부. 다중 부속 조회 스트림이 필요할 수 있습니다. 가능한 값은 다음과 같습니다. **N** 필요한 부속 조회 스트림이 없음 **Y** 하나 이상의 부속 조회 스트림이 필요함
FILTER_FACTOR	DOUBLE	아니오	아니오	술어에 의해 규정된 행의 추정 부분
PREDICATE_TEXT	CLOB(1M)	예	아니오	SQL문의 내부 표시로부터 재작성된 술어의 텍스트 사용 불가능한 경우, 널(NULL)이 적용됩니다.

표 39. 가능한 HOW_APPLIED 값

값	설명
JOIN	테이블 조인에 사용됨
RESID	Residual 술어로 평가됨

표 39. 가능한 HOW_APPLIED 값 (계속)

값	설명
SARG	색인 또는 데이터 페이지에 대한 sargable 술어로 평가됨
시작	시작 조건으로 사용됨
종료	중지 조건으로 사용됨

표 40. 가능한 RELOP_TYPE 값

값	설명
공백	적용 불가능
EQ	같음
GE	크거나 같음
GT	보다 큼
IN	목록에서
LE	작거나 같음
LK	같음
LT	보다 작음
NE	같지 않음
NL	널(NULL)임
NN	널(NULL)이 아님

EXPLAIN_STATEMENT 테이블

EXPLAIN_STATEMENT 테이블은 다른 레벨의 Explain 정보에 포함되어 있을 때처럼 SQL문의 텍스트를 포함합니다. 사용자가 입력한 원시 SQL문은 SQL문을 충족시키기 위해 액세스 플랜을 선택하는 데 (최적화 알고리즘에 의해) 사용된 버전과 함께 이 테이블에 저장됩니다. 위에 나오는 버전 중 후자의 경우는 SQL 컴파일러에 의해 판별된 것처럼 추가 술어로 재작성되고 강화되었으므로, 원래 버전과 유사하지 않을 수 있습니다.

이 테이블 정의에 대한 자세한 내용은 638 페이지의 『EXPLAIN_STATEMENT 테이블 정의』를 참조하십시오.

표 41. EXPLAIN_STATEMENT 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	PK, FK	Explain 요청 개시 프로그램 권한 부여 ID

표 41. EXPLAIN_STATEMENT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_TIME	시간소인	아니오	PK, FK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	PK, FK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	PK, FK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	PK	이 행에 적절한 Explain 정보 레벨 유효한 값은 다음과 같습니다. **O** 원시 텍스트(사용자가 입력한 그대로임) **P** 플랜 선택
STMTNO	정수	아니오	PK	Explain 정보와 관련된 패키지 내의 명령문 번호. 동적 Explain SQL문의 경우, 1로 설정됩니다. 정적 SQL문의 경우, 이 값은 SYSCAT.STATEMENTS 카탈로그 뷰에 사용된 값과 같습니다.
SECTNO	정수	아니오	PK	이 SQL문을 포함하는 패키지 내의 섹션 번호. 동적 Explain SQL문의 경우, 이것은 런타임 시 명령문에 대한 섹션을 보유하는 데 사용되는 절 번호가 됩니다. 정적 SQL문의 경우, 이 값은 SYSCAT.STATEMENTS 카탈로그 뷰에 사용된 값과 같습니다.
QUERYNO	정수	아니오	아니오	Explain된 SQL문에 대한 숫자 식별자. CLP 또는 CLI를 통해 발행된 동적 SQL문(SQL문 EXPLAIN 제외)의 경우, 기본값은 순차적으로 증가되는 값이 됩니다. 그렇지 않으면 기본값은 정적 SQL문의 경우 값 STMTNO가 되고, 동적 SQL문의 경우 1이 됩니다.
QUERYTAG	CHAR(20)	아니오	아니오	Explain된 SQL문에 대한 식별자 태그. CLP를 통해 발행된 동적 SQL문(SQL문 EXPLAIN 제외)의 경우, 기본값은 'CLP'가 됩니다. CLI를 통해 발행된 동적 SQL문(SQL문 EXPLAIN 제외)의 경우, 기본값은 'CLI'가 됩니다. 그렇지 않으면 사용되는 기본값은 공백이 됩니다.

표 41. EXPLAIN_STATEMENT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
STATEMENT_TYPE	CHAR(2)	아니오	아니오	설명 중인 조회 유형에 대한 설명 레이블 가능한 값은 다음과 같습니다. **S** 선택 **D** 삭제 **DC** 커서의 현재 위치 삭제 **I** 삽입 **U** 갱신 **UC** 커서의 현재 위치 갱신
UPDATABLE	CHAR(1)	아니오	아니오	명령문이 갱신 가능한 것으로 간주되는지 여부를 나타냅니다. 이것은 갱신 가능한 것으로 판별될 수 있는 SELECT 문과 특히 관련이 있습니다. 가능한 값은 다음과 같습니다. **' '** 적용 불가능(공백) **N** 아니오 **Y** 예
삭제 가능	CHAR(1)	아니오	아니오	명령문이 삭제 가능한 것으로 간주되는지 여부를 나타냅니다. 이것은 삭제 가능한 것으로 판별될 수 있는 SELECT 문과 특히 관련이 있습니다. 가능한 값은 다음과 같습니다. **' '** 적용 불가능(공백) **N** 아니오 **Y** 예
TOTAL_COST	DOUBLE	아니오	아니오	이 명령문에 대해 선택된 액세스 플랜을 실행하는 데 필요한 추정되는 전체 비용(timerons에서). EXPLAIN_LEVEL이 *O*(원시 텍스트)일 경우, 이 때 선택된 액세스 플랜이 없으므로 0으로 설정됩니다.
STATEMENT_TEXT	CLOB(1M)	아니오	아니오	설명중인 SQL문의 텍스트나 텍스트 부분. 플랜 선택 레벨의 Explain에 대해 표시된 텍스트는 내부 표시로부터 재구성되고 원래 SQL과 유사합니다. 따라서 재구성된 이 명령문은 올바른 SQL 구문을 따른다고 확신할 수 없습니다.

표 41. EXPLAIN_STATEMENT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
SNAPSHOT	BLOB(10M)	예	아니오	표시된 Explain_Level에서 이 SQL문에 대한 내부 표시의 스냅샷 이 컬럼은 DB2 Visual Explain과 함께 사용되도록 고안되었습니다. 컬럼은 EXPLAIN_LEVEL이 *0*(원시 명령문)일 경우, 이 특정 버전의 명령문이 캡쳐될 때 선택된 액세스 플랜이 없으므로 널(NULL)로 설정됩니다.
QUERY_DEGREE	정수	아니오	아니오	Explain 호출시 파티션 내 병렬 처리 수준을 나타냅니다. 원시 명령문의 경우, 여기에는 지시된 파티션 내 병렬 처리 수준이 포함됩니다. PLAN SELECTION의 경우, 여기에는 사용할 플랜에 대해 생성된 파티션 내 병렬 처리 플랜이 포함됩니다.

EXPLAIN_STREAM 테이블

EXPLAIN_STREAM 테이블은 개별 연산자와 데이터 오브젝트간의 I/O 데이터 스트림을 나타냅니다. 데이터 오브젝트 자체가 EXPLAIN_OBJECT 테이블에 나타나 있습니다. 데이터 스트림에 포함된 연산자는 EXPLAIN_OPERATOR 테이블에 표시됩니다.

표 42. EXPLAIN_STREAM 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	Explain 요청 개시 프로그램 권한 부여 ID
EXPLAIN_TIME	시간소인	아니오	FK	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 소스 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행에 적절한 Explain 정보 레벨
STMTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	정수	아니오	FK	Explain 정보와 관련된 패키지 내의 섹션 번호
STREAM_ID	정수	아니오	아니오	지정된 연산자 내의 데이터 스트림에 대한 고유 ID

표 42. EXPLAIN_STREAM 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키	설명
SOURCE_TYPE	CHAR(1)	아니오	아니오	다음은 데이터 스트림의 소스를 나타냅니다. **O** 연산자 **D** 데이터 오브젝트
SOURCE_ID	SMALLINT	아니오	아니오	데이터 스트림의 소스가 되는 조회 내의 연산자에 대한 고유 ID. SOURCE_TYPE이 'D'일 경우, -1로 설정됩니다.
TARGET_TYPE	CHAR(1)	아니오	아니오	다음 데이터 스트림의 목표를 나타냅니다. **O** 연산자 **D** 데이터 오브젝트
TARGET_ID	SMALLINT	아니오	아니오	데이터 스트림의 목표가 되는 조회 내의 연산자에 대한 고유 ID. TARGET_TYPE이 'D'일 경우, -1로 설정됩니다.
OBJECT_SCHEMA	VARCHAR(128)	예	아니오	영향받은 데이터 주제가 속하는 스키마. SOURCE_TYPE과 TARGET_TYPE이 모두 'O'일 경우, 널(NULL)로 설정됩니다.
OBJECT_NAME	VARCHAR(128)	예	아니오	데이터 스트림의 오브젝트가 되는 오브젝트의 이름. SOURCE_TYPE과 TARGET_TYPE이 모두 'O'일 경우, 널(NULL)로 설정됩니다.
STREAM_COUNT	DOUBLE	아니오	아니오	데이터 스트림 추정 기본 행 수(cardinality)
COLUMN_COUNT	SMALLINT	아니오	아니오	데이터 스트림의 컬럼 수
PREDICATE_ID	정수	아니오	아니오	스트림이 술어에 대한 부속 조회의 일부일 경우 술어 ID가 반영되고, 그렇지 않으면 컬럼은 -1로 설정됩니다.
COLUMN_NAMES	CLOB(1M)	예	아니오	이 컬럼에는 스트림에 포함된 컬럼의 이름과 순서 정보가 들어 있습니다. 이들 이름의 형식은 다음과 같습니다. `NAME1(A)+NAME2(D)+NAME3+NAME4` 여기서 *(A)*는 오름차순으로 정렬된 컬럼을 나타내고, *(D)*는 내림차순으로 정렬된 컬럼을 나타내며, 정렬에 대한 정보가 없는 것은 컬럼이 정렬되어 있지 않거나 정렬 상태가 적절하지 않습니다.
PMID	SMALLINT	아니오	아니오	파티션 맵 ID

표 42. EXPLAIN_STREAM 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키	설명
SINGLE_NODE	CHAR(5)	예	아니오	다음과 같이 데이터 스트림이 단일 파티션에 있는지 다중 파티션에 있는지를 나타냅니다. **MULT** 다중 파티션 **COOR** 조정자(coordinator) 노드 **해쉬** 해쉬 사용이 지시됨 **RID** 행 ID 사용이 지시됨 **FUNC** 함수 사용이 지시됨(PARTITION() 또는 NODENUMBER()) **상관** 상관 값 사용이 지시됨 **숫자** 사전에 판별된 단일 노드로 지시됨
PARTITION_COLUMNS	CLOB(64K)	예	아니오	데이터 스트림이 파티션된 컬럼의 목록

ADVISE_INDEX 테이블

ADVISE_INDEX 테이블은 권장 색인을 나타냅니다.

표 43. ADVISE_INDEX 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	아니오	Explain 요청 개시 프로그램 권한 부여 ID
EXPLAIN_TIME	시간소인	아니오	아니오	Explain 요청을 개시하는 시간
SOURCE_NAME	VARCHAR(128)	아니오	아니오	동적문을 설명할 때 수행하는 패키지 이름 또는 정적 SQL을 설명할 때의 원시 파일 이름
SOURCE_SCHEMA	VARCHAR(128)	아니오	아니오	Explain 요청에 대한 스키마, 규정자, 소스
EXPLAIN_LEVEL	CHAR(1)	아니오	아니오	이 행에 적절한 Explain 정보 레벨
STMTNO	정수	아니오	아니오	Explain 정보와 관련된 패키지 내의 명령문 번호
SECTNO	정수	아니오	아니오	Explain 정보와 관련된 패키지 내의 섹션 번호

표 43. ADVISE_INDEX 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
QUERYNO	정수	아니오	아니오	Explain된 SQL문에 대한 숫자 식별자. CLP 또는 CLI를 통해 발행된 동적 SQL문(SQL문 EXPLAIN 제외)의 경우, 기본값은 순차적으로 증가되는 값이 됩니다. 그렇지 않으면 기본값은 정적 SQL문의 경우 값 STMTNO가 되고, 동적 SQL문의 경우 1이 됩니다.
QUERYTAG	CHAR(20)	아니오	아니오	Explain된 SQL문에 대한 식별자 태그. CLP를 통해 발행된 동적 SQL문(SQL문 EXPLAIN 제외)의 경우, 기본값은 'CLP'가 됩니다. CLI를 통해 발행된 동적 SQL문(SQL문 EXPLAIN 제외)의 경우, 기본값은 'CLI'가 됩니다. 그렇지 않으면 사용되는 기본값은 공백이 됩니다.
이름	VARCHAR(128)	아니오	아니오	색인 이름
생성기	VARCHAR(128)	아니오	아니오	색인 이름의 규정자
TBNAME	VARCHAR(128)	아니오	아니오	색인이 정의된 테이블 이름 또는 별칭
TBCREATOR	VARCHAR(128)	아니오	아니오	테이블 이름 규정자
COLNAMES	CLOB(64K)	아니오	아니오	컬럼 이름의 목록
UNIQUERULE	CHAR(1)	아니오	아니오	고유 규칙 D = 중복 허용 P = 1차 색인 U = 고유 항목만 허용
COLCOUNT	SMALLINT	아니오	아니오	키에 있는 컬럼 수와 include 컬럼(있는 경우) 수를 합한 것
IID	SMALLINT	아니오	아니오	내부 색인 ID
NLEAF	정수	아니오	아니오	통계가 수집되지 않은 경우, 리프 페이지의 수는 -1이 됩니다.
NLEVELS	SMALLINT	아니오	아니오	통계가 수집되지 않은 경우, 색인 레벨의 수는 -1이 됩니다.
FULLKEYCARD	BIGINT	아니오	아니오	통계가 수집되지 않은 경우, 전체 구별 키 값의 수는 -1이 됩니다.
FIRSTKEYCARD	BIGINT	아니오	아니오	통계가 수집되지 않은 경우, 첫 번째 구별 키 값은 -1이 됩니다.
CLUSTERRATIO	SMALLINT	아니오	아니오	색인을 사용한 데이터 클러스터링 등급. 통계가 수집되지 않거나 상세한 색인 통계가 수집된 경우(두 경우 모두 CLUSTERFACTOR가 대신 사용됨), -1이 적용됩니다.
CLUSTERFACTOR	DOUBLE	아니오	아니오	클러스터링 등급을 세분화하여 측정합니다. 상세한 색인 통계가 수집되지 않거나 색인이 별칭에 정의된 경우, -1이 적용됩니다.
USERDEFINED	SMALLINT	아니오	아니오	사용자에 의해 정의됨.

표 43. ADVISE_INDEX 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
SYSTEM_REQUIRED	SMALLINT	아니오	아니오	기본 키 또는 고유 키 제한조건에 이 색인이 필요하거나, 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 대한 색인인 경우, 1이 적용됩니다. 기본 키 또는 고유 키 제한조건에 이 색인이 필요하고, 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 대한 색인인 경우, 2가 적용됩니다. 그렇지 않으면 0이 적용됩니다.
CREATE_TIME	시간소인	아니오	아니오	색인이 작성된 시간
STATS_TIME	시간소인	예	아니오	이 색인에 대해 기록된 통계에 변경이 수행된 마지막 시간. 사용 가능한 통계가 없을 경우, 널(NULL)이 적용됩니다.
PAGE_FETCH_PAIRS	VARCHAR(254)	아니오	아니오	정수의 페어 목록으로서 문자 형식으로 표현합니다. 각 페어는 가상 버퍼의 페이지 수와 가상 버퍼를 사용하는 색인을 통해 테이블을 스캔하는 데 필요한 페이지 페치의 수를 나타냅니다. (사용 가능한 데이터가 없을 경우, 0 길이 문자열이 적용됩니다.)
REMARKS	VARCHAR(254)	예	아니오	사용자 제공 주석 또는 널(NULL)
DEFINER	VARCHAR(128)	아니오	아니오	색인을 작성한 사용자
CONVERTED	CHAR(1)	아니오	아니오	차후 사용을 위해 예약됩니다.
SEQUENTIAL_PAGES	정수	아니오	아니오	서로 간격이 거의 없는 색인 키순으로 디스크에 배치된 리프 페이지 수(사용 가능한 통계가 없을 경우, -1이 적용됩니다.)
DENSITY	정수	아니오	아니오	색인이 차지한 페이지 범위 내의 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로서, 백분율 단위로 표시됩니다. (사용 가능한 통계가 없을 경우, 0과 100 사이의 정수 또는 -1이 적용됩니다.)
FIRST2KEYCARD	BIGINT	아니오	아니오	색인의 처음 두 개의 컬럼을 사용하는 구별 키의 수(통계가 없거나 적용 불가능한 통계일 경우, -1이 적용됨)
FIRST3KEYCARD	BIGINT	아니오	아니오	색인의 처음 세 개의 컬럼을 사용하는 구별 키의 수(통계가 없거나 적용 불가능한 통계일 경우, -1이 적용됨)
FIRST4KEYCARD	BIGINT	아니오	아니오	색인의 처음 네 개의 컬럼을 사용하는 구별 키의 수(통계가 없거나 적용 불가능한 통계일 경우, -1이 적용됨)
PCTFREE	SMALLINT	아니오	아니오	초기 색인 빌드 중에 보유될 각 색인 리프 페이지의 백분율. 이 공간은 색인이 빌드된 후 삽입에 사용될 수 있습니다.

표 43. ADVISE_INDEX 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
UNIQUE_COLCOUNT	SMALLINT	아니오	아니오	고유 키에 필요한 컬럼 수. 항상 <=COLCOUNT입니다. incude 컬럼이 있을 경우에만 < COLCOUNT가 적용됩니다. 색인이 고유 키가 없을 경우, -1이 적용됩니다(중복 허용).
MINPCTUSED	SMALLINT	아니오	아니오	0이 아닐 경우, 온라인 색인 재구성이 가능하며 값은 페이지 병합 전 최소 사용 공간의 임계값입니다.
REVERSE_SCANS	CHAR(1)	아니오	아니오	Y = 색인이 역 스캔을 지원합니다. N = 색인이 역 스캔을 지원하지 않습니다.
USE_INDEX	CHAR(1)	예	아니오	Y = 색인이 권장되거나 평가됩니다. N = 색인이 권장되지 않습니다.
CREATION_TEXT	CLOB(1M)	아니오	아니오	색인 작성에 사용되는 SQL문
PACKED_DESC	BLOB(20M)	예	아니오	테이블의 내부 설명.

ADVISE_WORKLOAD 테이블

ADVISE_WORKLOAD 테이블은 워크로드를 구성하는 명령문을 나타냅니다. 워크로드에 대한 자세한 내용은 *관리 안내서: 성능*을 참조하십시오.

표 44. ADVISE_WORKLOAD 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능 여부	키?	설명
WORKLOAD_NAME	CHAR(128)	아니오	아니오	이 명령문이 속하는 SQL문(워크로드) 콜렉션의 이름
STATEMENT_NO	정수	아니오	아니오	이 Explain 정보가 관련되는 워크로드 내의 명령문 번호
STATEMENT_TEXT	CLOB(1M)	아니오	아니오	SQL문의 내용
STATEMENT_TAG	VARCHAR(256)	아니오	아니오	Explain된 SQL문에 대한 식별자 태그
FREQUENCY	정수	아니오	아니오	이 명령문이 워크로드 내에서 나타나는 횟수
IMPORTANCE	DOUBLE	아니오	아니오	명령문의 중요성
COST_BEFORE	DOUBLE	예	아니오	권장 색인이 작성되지 않을 경우의 조회 비용(timeron에서)
COST_AFTER	DOUBLE	예	아니오	권장 색인이 작성될 경우의 조회 비용(timeron 단위)

Explain 테이블에 대한 테이블 정의

Explain을 호출하려면 Explain 테이블을 먼저 작성해야 합니다. 다음 정의는 필요한 Explain 테이블 작성 방법을 지정합니다.

- 633 페이지의 『EXPLAIN_ARGUMENT 테이블 정의』
- 634 페이지의 『EXPLAIN_INSTANCE 테이블 정의』
- 635 페이지의 『EXPLAIN_OBJECT 테이블 정의』
- 636 페이지의 『EXPLAIN_OPERATOR 테이블 정의』
- 637 페이지의 『EXPLAIN_PREDICATE 테이블 정의』
- 638 페이지의 『EXPLAIN_STATEMENT 테이블 정의』
- 639 페이지의 『EXPLAIN_STREAM 테이블 정의』
- 640 페이지의 『ADVISE_INDEX 테이블 정의』
- 642 페이지의 『ADVISE_WORKLOAD 테이블 정의』

또는 'sqllib' 디렉토리의 'misc' 서브디렉토리에 있는 EXPLAIN.DDL 파일에 제공된 샘플 명령행 처리기 입력 스크립트를 사용하여 이를 작성하십시오. Explain 테이블이 필요한 데이터베이스에 연결하십시오. 그런 다음 `db2 -tf EXPLAIN.DDL` 명령을 발행하면 이 테이블이 작성됩니다.

EXPLAIN_ARGUMENT 테이블 정의

```
CREATE TABLE EXPLAIN_ARGUMENT ( EXPLAIN_REQUESTER   VARCHAR(128)  NOT NULL,
                                EXPLAIN_TIME        TIMESTAMP     NOT NULL,
                                SOURCE_NAME         VARCHAR(128)  NOT NULL,
                                SOURCE_SCHEMA       VARCHAR(128)  NOT NULL,
                                EXPLAIN_LEVEL       CHAR(1)       NOT NULL,
                                STMTNO              INTEGER       NOT NULL,
                                SECTNO              INTEGER       NOT NULL,
                                OPERATOR_ID         INTEGER       NOT NULL,
                                ARGUMENT_TYPE       CHAR(8)       NOT NULL,
                                ARGUMENT_VALUE      VARCHAR(1024) NOT NULL,
                                LONG_ARGUMENT_VALUE CLOB(1M)      NOT LOGGED,
                                   FOREIGN KEY (EXPLAIN_REQUESTER,
                                                EXPLAIN_TIME,
                                                SOURCE_NAME,
                                                SOURCE_SCHEMA,
                                                EXPLAIN_LEVEL,
                                                STMTNO,
                                                SECTNO)
                                   REFERENCES EXPLAIN_STATEMENT
                                   ON DELETE CASCADE )
```

EXPLAIN_INSTANCE 테이블 정의

```
CREATE TABLE EXPLAIN_INSTANCE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_OPTION    CHAR(1)      NOT NULL,
                                SNAPSHOT_TAKEN    CHAR(1)      NOT NULL,
                                DB2_VERSION       CHAR(7)      NOT NULL,
                                SQL_TYPE          CHAR(1)      NOT NULL,
                                QUERYOPT          INTEGER      NOT NULL,
                                BLOCK             CHAR(1)      NOT NULL,
                                ISOLATION         CHAR(2)      NOT NULL,
                                BUFFPAGE          INTEGER      NOT NULL,
                                AVG_APPLS         INTEGER      NOT NULL,
                                SORTHEAP          INTEGER      NOT NULL,
                                LOCKLIST          INTEGER      NOT NULL,
                                MAXLOCKS          SMALLINT     NOT NULL,
                                LOCKS_AVAIL       INTEGER      NOT NULL,
                                CPU_SPEED         DOUBLE       NOT NULL,
                                REMARKS           VARCHAR(254),
                                DBHEAP            INTEGER      NOT NULL,
                                COMM_SPEED        DOUBLE       NOT NULL,
                                PARALLELISM       CHAR(2)      NOT NULL,
                                DATAJOINER        CHAR(1)      NOT NULL,
                                    PRIMARY KEY (EXPLAIN_REQUESTER,
                                                 EXPLAIN_TIME,
                                                 SOURCE_NAME,
                                                 SOURCE_SCHEMA))
```

EXPLAIN_OBJECT 테이블 정의

```
CREATE TABLE EXPLAIN_OBJECT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128) NOT NULL,
                              OBJECT_NAME       VARCHAR(128) NOT NULL,
                              OBJECT_TYPE       CHAR(2)      NOT NULL,
                              CREATE_TIME       TIMESTAMP,
                              STATISTICS_TIME   TIMESTAMP,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              ROW_COUNT         INTEGER      NOT NULL,
                              WIDTH             INTEGER      NOT NULL,
                              PAGES             INTEGER      NOT NULL,
                              DISTINCT          CHAR(1)      NOT NULL,
                              TABLESPACE_NAME   VARCHAR(128),
                              OVERHEAD          DOUBLE       NOT NULL,
                              TRANSFER_RATE     DOUBLE       NOT NULL,
                              PREFETCHSIZE      INTEGER      NOT NULL,
                              EXTENTSIZE        INTEGER      NOT NULL,
                              CLUSTER           DOUBLE       NOT NULL,
                              NLEAF             INTEGER      NOT NULL,
                              NLEVELS           INTEGER      NOT NULL,
                              FULLKEYCARD       BIGINT       NOT NULL,
                              OVERFLOW          INTEGER      NOT NULL,
                              FIRSTKEYCARD      BIGINT       NOT NULL,
                              FIRST2KEYCARD     BIGINT       NOT NULL,
                              FIRST3KEYCARD     BIGINT       NOT NULL,
                              FIRST4KEYCARD     BIGINT       NOT NULL,
                              SEQUENTIAL_PAGES  INTEGER      NOT NULL,
                              DENSITY           INTEGER      NOT NULL,
                                      FOREIGN KEY (EXPLAIN_REQUESTER,
                                                   EXPLAIN_TIME,
                                                   SOURCE_NAME,
                                                   SOURCE_SCHEMA,
                                                   EXPLAIN_LEVEL,
                                                   STMTNO,
                                                   SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

EXPLAIN_OPERATOR 테이블 정의

```
CREATE TABLE EXPLAIN_OPERATOR ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                STMTNO            INTEGER      NOT NULL,
                                SECTNO            INTEGER      NOT NULL,
                                OPERATOR_ID       INTEGER      NOT NULL,
                                OPERATOR_TYPE     CHAR(6)      NOT NULL,
                                TOTAL_COST        DOUBLE       NOT NULL,
                                IO_COST           DOUBLE       NOT NULL,
                                CPU_COST          DOUBLE       NOT NULL,
                                FIRST_ROW_COST    DOUBLE       NOT NULL,
                                RE_TOTAL_COST     DOUBLE       NOT NULL,
                                RE_IO_COST        DOUBLE       NOT NULL,
                                RE_CPU_COST       DOUBLE       NOT NULL,
                                COMM_COST         DOUBLE       NOT NULL,
                                FIRST_COMM_COST   DOUBLE       NOT NULL,
                                REMOTE_TOTAL_COST DOUBLE       NOT NULL,
                                REMOTE_COMM_COST  DOUBLE       NOT NULL,
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                  SECTNO)
                                     REFERENCES EXPLAIN_STATEMENT
                                     ON DELETE CASCADE )
```

EXPLAIN_PREDICATE 테이블 정의

```
CREATE TABLE EXPLAIN_PREDICATE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 OPERATOR_ID       INTEGER      NOT NULL,
                                 PREDICATE_ID      INTEGER      NOT NULL,
                                 HOW_APPLIED       CHAR(5)      NOT NULL,
                                 WHEN_EVALUATED    CHAR(3)      NOT NULL,
                                 RELOP_TYPE        CHAR(2)      NOT NULL,
                                 SUBQUERY          CHAR(1)      NOT NULL,
                                 FILTER_FACTOR     DOUBLE       NOT NULL,
                                 PREDICATE_TEXT    CLOB(1M)     NOT LOGGED,
                                      FOREIGN KEY (EXPLAIN_REQUESTER,
                                                   EXPLAIN_TIME,
                                                   SOURCE_NAME,
                                                   SOURCE_SCHEMA,
                                                   EXPLAIN_LEVEL,
                                                   STMTNO,
                                                   SECTNO)
                                      REFERENCES EXPLAIN_STATEMENT
                                      ON DELETE CASCADE )
```

EXPLAIN_STATEMENT 테이블 정의

```
CREATE TABLE EXPLAIN_STATEMENT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 QUERYNO           INTEGER      NOT NULL,
                                 QUERYTAG          CHAR(20)     NOT NULL,
                                 STATEMENT_TYPE    CHAR(2)      NOT NULL,
                                 UPDATABLE         CHAR(1)      NOT NULL,
                                 DELETABLE         CHAR(1)      NOT NULL
                                 TOTAL_COST        DOUBLE       NOT NULL,
                                 STATEMENT_TEXT    CLOB(1M)     NOT NULL
                                                                NOT LOGGED,
                                 SNAPSHOT          BLOB(10M)    NOT LOGGED,
                                 QUERY_DEGREE      INTEGER      NOT NULL,
                                     PRIMARY KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                  SECTNO),
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA)
                                     REFERENCES EXPLAIN_INSTANCE
                                     ON DELETE CASCADE )
```

EXPLAIN_STREAM 테이블 정의

```
CREATE TABLE EXPLAIN_STREAM ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              STREAM_ID         INTEGER      NOT NULL,
                              SOURCE_TYPE       CHAR(1)      NOT NULL,
                              SOURCE_ID         SMALLINT     NOT NULL,
                              TARGET_TYPE       CHAR(1)      NOT NULL,
                              TARGET_ID         SMALLINT     NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128),
                              OBJECT_NAME       VARCHAR(128),
                              STREAM_COUNT      DOUBLE       NOT NULL,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              PREDICATE_ID      INTEGER      NOT NULL,
                              COLUMN_NAMES      CLOB(1M)     NOT LOGGED,
                              PMID              SMALLINT     NOT NULL,
                              SINGLE_NODE       CHAR(5),
                              PARTITION_COLUMNS CLOB(64K)    NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

ADVISE_INDEX 테이블 정의

```
CREATE TABLE ADVISE_INDEX (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           EXPLAIN_TIME      TIMESTAMP    NOT NULL
                             WITH DEFAULT CURRENT TIMESTAMP,
                           SOURCE_NAME       VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           SOURCE_SCHEMA     VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           EXPLAIN_LEVEL     CHAR(1)      NOT NULL
                             WITH DEFAULT '',
                           STMTNO            INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           SECTNO            INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           QUERYNO           INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           QUERYTAG          CHAR(20)     NOT NULL
                             WITH DEFAULT '',
                           NAME              VARCHAR(128) NOT NULL,
                           CREATOR           VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           TBNAME            VARCHAR(128) NOT NULL,
                           TBCREATOR         VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           COLNAMES          CLOB(64K)    NOT NULL,
                           UNIQUERULE        CHAR(1)      NOT NULL
                             WITH DEFAULT '',
                           COLCOUNT          SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           IID               SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           NLEAF             INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           NLEVELS           SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           FIRSTKEYCARD      BIGINT       NOT NULL
                             WITH DEFAULT 0,
                           FULLKEYCARD       BIGINT       NOT NULL
                             WITH DEFAULT 0,
                           CLUSTERRATIO      SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           CLUSTERFACTOR     DOUBLE       NOT NULL
                             WITH DEFAULT 0,
                           USERDEFINED       SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           SYSTEM_REQUIRED   SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           CREATE_TIME       TIMESTAMP    NOT NULL
                             WITH DEFAULT CURRENT TIMESTAMP,
                           STATS_TIME        TIMESTAMP
                             WITH DEFAULT CURRENT TIMESTAMP,
                           PAGE_FETCH_PAIRS  VARCHAR(254) NOT NULL
                             WITH DEFAULT '',
                           REMARKS           VARCHAR(254)
                             WITH DEFAULT '',
                           DEFINER           VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           CONVERTED          CHAR(1) NOT NULL
                             WITH DEFAULT '',
```

```
SEQUENTIAL_PAGES  INTEGER      NOT NULL
                     WITH DEFAULT 0,
DENSITY           INTEGER      NOT NULL
                     WITH DEFAULT 0,
FIRST2KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
FIRST3KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
FIRST4KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
PCTFREE           SMALLINT     NOT NULL
                     WITH DEFAULT -1,
UNIQUE_COLCOUNT   SMALLINT     NOT NULL
                     WITH DEFAULT -1,
MINPCTUSED        SMALLINT     NOT NULL
                     WITH DEFAULT 0,
REVERSE_SCANS     CHAR(1)      NOT NULL
                     WITH DEFAULT 'N',
USE_INDEX         CHAR(1),
CREATION_TEXT     CLOB(1M)     NOT NULL
                     NOT LOGGED WITH DEFAULT '',
PACKED_DESC       BLOB(1M)     NOT LOGGED)
```

ADVISE_WORKLOAD 테이블 정의

```
CREATE TABLE ADVISE_WORKLOAD (WORKLOAD_NAME  CHAR(128)    NOT NULL
                                WITH DEFAULT 'WK0',
                              STATEMENT_NO   INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              STATEMENT_TEXT CLOB(1M)     NOT NULL NOT LOGGED,
                              STATEMENT_TAG  VARCHAR(256) NOT NULL
                                 WITH DEFAULT '',
                              FREQUENCY      INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              IMPORTANCE     DOUBLE       NOT NULL
                                WITH DEFAULT 1,
                              COST_BEFORE    DOUBLE,
                              COST_AFTER     DOUBLE)
```

부록C. SQL Explain 도구

db2expln 도구에서는 시스템 카탈로그 테이블에 저장되어 있는 패키지 내의 정적 SQL문에 대해 선택된 액세스 플랜을 설명합니다. 이 도구는 바인드 시간에 Explain 데이터가 캡쳐되지 않은 패키지에 대한 액세스 플랜의 신속한 설명을 얻는 데 사용될 수 있습니다.

dynexpln 도구에서는 동적문에 대해 선택된 액세스 플랜을 설명합니다. 이 도구는 명령문에 대한 정적 패키지를 작성하고, **db2expln** 도구를 사용하여 이를 설명합니다.

이 Explain 도구를 사용하여 특정 SQL문에 대해 선택된 액세스 플랜을 이해할 수 있습니다. 또는 통합 Explain 기능(243 페이지의 『제7장 SQL Explain 기능』)을 Visual Explain와 함께 사용하여 특정 SQL문에 선택된 액세스 플랜을 이해할 수 있습니다. 동적 및 정적 SQL문 둘다 Explain 기능을 사용하여 설명될 수 있습니다. Explain 도구와 다른 점은 Visual Explain을 사용하면 Explain 정보가 그래픽 형식으로 표시된다는 점입니다. 아니면, 두 방법에 제공된 세부사항 레벨이 같습니다.

db2expln 및 dynexpln의 출력을 전체적으로 활용하려면 다음을 이해해야 합니다.

- 지원되는 각기 다른 SQL문과 해당 명령문에 관계된 용어(예: SELECT문에서의 술어)
- 패키지의 목적(액세스 플랜)(183 페이지의 『데이터 액세스 개념 및 최적화』 참조)
- 시스템 카탈로그 테이블의 목적 및 내용(*SQL 참조서* 참조)
- 43 페이지의 『제2부 응용프로그램 성능 조정』에 설명된 기타 개념

다음 주제에서는 db2expln 및 dynexpln에 대한 정보를 제공합니다.

- db2expln 및 dynexpln 수행
- db2expln 구문 및 매개변수

© Copyright IBM Corp. 1993, 2001

- db2expln 사용에 관한 유의사항
- dynexpln 구문 및 매개변수
- dynexpln 사용에 관한 유의사항
- db2expln 및 dynexpln 출력에 대한 설명
- db2expln 및 dynexpln 출력의 예

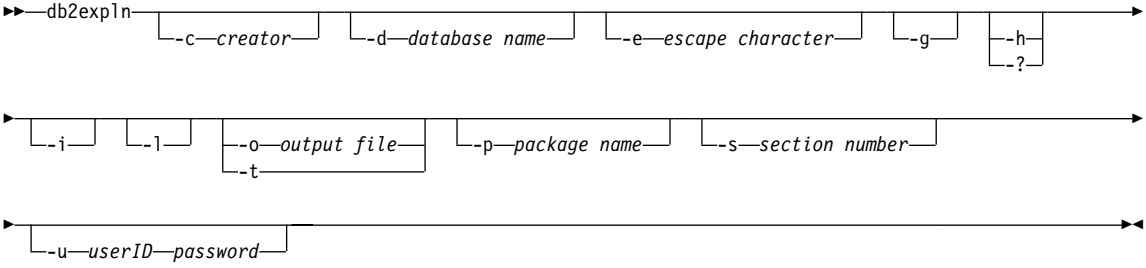
db2expln 및 dynexpln 수행

Explain 도구(db2expln 및 dynexpln)는 sqllib 디렉토리의 misc에 있습니다. db2expln 및 dynexpln이 현재 디렉토리에 있지 않을 경우, 이것은 PATH 환경 변수에 나타난 디렉토리에 있어야 합니다.

db2expln 프로그램은 데이터베이스에 처음으로 액세스할 때 db2expln.bnd 파일을 사용하여 스스로 데이터베이스에 연결하고 바인드니다. db2expln.bnd 파일은 sqllib 디렉토리의 bnd 서브디렉토리에 있습니다.

db2expln을 수행하려면 사용자에게 시스템 카탈로그 뷰에 대한 SELECT 특권이 있어야 하며, db2expln 패키지에 대한 EXECUTE 권한이 있어야 합니다. dynexpln을 수행하려면 데이터베이스에 대한 BINDADD 권한이 있어야 하고, 데이터베이스 연결에 사용 중인 스키마가 존재해야 하거나 데이터베이스에 대한 EXPLICIT_SCHEMA 권한이 있어야 하며, Explain 중인 SQL문에 필요한 모든 권한이 있어야 합니다. (사용자에게 SYSADM 또는 DBADM 권한이 있는 경우, 자동으로 이 모든 권한 레벨을 갖게 됩니다.)

db2expln 구문 및 매개변수



여기서

-c creator

패키지 작성자의 사용자 ID

이 옵션을 지정하지 않으면 입력하라는 메시지가 나타납니다.

LIKE 술어에 사용될 수 있는 패턴 일치 문자, 퍼센트 부호(%) 및 밑줄(_)을 사용하여 작성자 이름을 지정할 수 있습니다.

-d database name

Explain될 패키지를 포함하는 데이터베이스의 이름.

이 옵션을 지정하지 않으면, 입력하라는 메시지가 나타납니다.

-e escape character

패턴 일치 문자가 아닌 Escape 문자로 해석되는 문자를 지정하는 데 사용됩니다.

예를 들어, 패키지 TESTID.CALC%를 Explain한 `db2expln` 명령은 `db2expln -c TESTID -p CALC%`입니다. 그러나 이 명령은 `CALC`로 시작하는 다른 모든 플랜에 대해도 Explain합니다. TESTID.CALC% 패키지만을 Explain하려면 Escape 문자를 사용해야 합니다. 읽을 명령을 `db2expln -c TESTID -e ! -p CALC!%`로 변경하면 `!` 문자가 Escape 문자로 사용되며 `!%`는 `%` 문자로 해석됩니다.

-g 최적화 알고리즘 플랜 그래프를 표시합니다. 각 섹션이 검토되고 원래의 최적화 알고리즘 플랜 그래프(Visual Explain에 의해 표시됨)가 구성됩니다. 생성된 그래프가 원래 플랜과 일치하지 않을 수 있다는 점을 유의하십시오.

-h 또는 -?

입력 매개변수에 관한 도움말 정보를 얻습니다. 이 옵션의 지정은 다른 모든 옵션을 겹쳤습니다.

-i Explain된 플랜에 연산자 ID를 표시합니다. 연산자 ID로 db2expln의 출력을 Explanin 기능의 출력과 일치시킬 수 있습니다.

-l 이 옵션이 지정되면 패키지 이름이 소문자 또는 혼합 문자형으로 될 수 있습니다. **-l** 옵션이 지정되지 않으면 패키지 이름은 대문자로 변환됩니다.

-o output file

db2expln이 결과를 작성할 파일 이름.

사용자가 파일 이름 없이 **-o**를 지정하면 사용자에게 파일 이름에 대한 프롬프트를 표시됩니다. 기본 파일 이름은 db2expln.out입니다.

-t 출력이 터미널로 방향이 지정되었습니다.

사용자가 **-o** 또는 **-t**를 지정하지 않으면 기본 출력이 터미널에 표시됨과 동시에 사용자에게 파일 이름에 대한 프롬프트가 표시됩니다.

-p package name

Explain될 패키지 이름.

이 옵션을 지정하지 않을 경우, 사용자에게 옵션에 대한 프롬프트가 표시됩니다.

LIKE 술어에 사용될 수 있는 패턴 일치 문자, 퍼센트 부호(%) 및 밑줄(_)을 사용하여 패키지 이름을 지정할 수 있습니다.

-s section number

패키지 내에서 Explain하는 섹션 번호. 사용자가 패키지에 있는 모든 섹션을 설명하려고 하는 경우, 번호 0이 지정됩니다. 패키지 작성자(**-c**) 또는 패키지 이름(**-p**) 인수가 여러 개의 패키지 및 섹션이 Explain될 것을 의미할 경우, 섹션 값이 제공되면 값을 0으로 겹쳐씁니다.

이 옵션을 지정하지 않을 경우, 사용자에게 옵션을 제공하도록 프롬프트가 표시됩니다.

섹션 번호는 시스템 카탈로그 SYSCAT.STATEMENTS를 조회함으로써 알 수 있습니다. (시스템 카탈로그 테이블에 대한 설명은 *SQL 참조서*를 참조하십시오.)

-u userID password

데이터베이스에 연결할 때 제공된 사용자 ID와 암호를 사용하십시오.

사용자 ID와 암호는 DB2 이름 지정 규칙에 맞게 유효해야 하며, 데이터베이스가 인식할 수 있어야 합니다.

위 테이블의 일부 옵션 플래그는 운영 체제에 따라 특수 의미를 띄기 때문에 db2expln 명령행에서 정확하게 해석되지 않을 수 있습니다. 그러나 이 문자 앞에 Escape 문자를 입력할 수는 있습니다. 자세한 내용은 운영 체제 사용자 매뉴얼을 참조하십시오.

db2expln에 의해 생성된 도움말과 초기 상태 메시지는 표준 출력에 작성됩니다. Explain 도구에 의해 생성된 모든 프롬프트와 다른 상태 메시지는 표준 오류에 작성됩니다. 선택된 출력 옵션에 따라 Explain 텍스트는 표준 출력 장치 또는 파일로 작성됩니다.

-p 및 **-c** 옵션과 함께 다중 플랜은 LIKE 패턴으로 패키지와 작성자에 대한 문자열 상수를 지정하여 하나의 Explain 호출로 설명될 수 있습니다. 즉, 밑줄(_)은 단일 문자를 나타내는 데 사용할 수 있고, 퍼센트 부호(%)는 0개 이상의 문자가 사용되었음을 나타내는 데 사용할 수 있습니다.

예를 들어, SAMPLE이라는 이름의 데이터베이스에서 모든 플랜에 대한 모든 섹션을 Explain하고, 그 결과를 **my.exp** 파일에 작성하려면 다음과 같이 입력하십시오.

```
db2expln -d SAMPLE -p % -c % -s 0 -o my.exp
```

db2expln 사용에 관한 유의사항

다음은 db2expln에 의해 표시된 공통 메시지입니다.

- 데이터베이스에 패키지 없음<database>, 패키지 패턴: <작성자>.<패키지>

  지정된 패턴에 일치하는 데이터베이스에서 패키지가 발견되지 않는 경우, 이 메시지가 표시됩니다.

- 바인드 메시지는 db2expln.msg에서 찾을 수 있습니다.

  db2expln.bnd의 바인드가 성공적이지 않을 경우, 이 메시지가 표시됩니다. 발생한 문제점에 대한 정보는 현재 디렉토리의 db2expln.msg 파일에서 찾아볼 수 있습니다.

- 섹션 번호는 잠재적인 복합 패키지에 대해 0으로 겹쳐씁니다.

  다중 패키지가 db2expln에 의해 발견될 때 이 메시지가 표시됩니다. 패턴 일치 문자 중 하나가 패키지 또는 작성자 입력 인수에서 사용된 경우, 조치가 취해집니다.

- 정적 섹션을 패키지로 부터 규정해서는 안됩니다.

지정된 패키지가 동적 SQL문만을 가지고 있는 경우, 즉 정적 섹션이 없는 경우에 이 메시지가 표시됩니다.

- 데이터베이스<데이터베이스>, 패키지 <작성자>.<패키지>는 유효하지 않습니다. 리바인드된 후에 db2expln이 리턴됩니다.

  지정된 패키지가 현재 유효하지 않은 경우, 이 메시지가 표시됩니다. 지시된 대로 플랜에 대해 BIND 또는 REBIND 명령을 재발행하여 데이터베이스에서 유효한 패키지를 재작성한 다음 db2expln을 리턴하십시오.

- 섹션이 처리되지 않습니다. 지원되지 않는 릴리스로 생성됩니다.

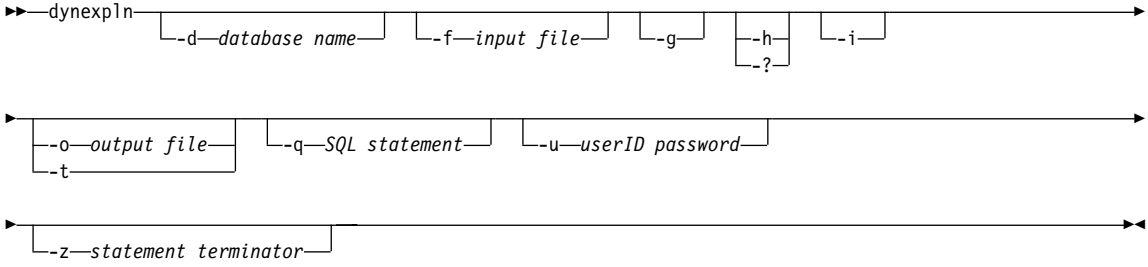
  현재 처리 중인 섹션이 db2expln 실행 파일을 제공한 릴리스가 아닌 다른 DB2 릴리스에 의해 생성된 것일 때 이 메시지는 나타납니다. 이 경우, 섹션을 생성한 DB2 릴리스에서 db2expln 사본을 사용하십시오.

제외되는 SQL문: 다음 명령문은 설명되지 않습니다.

- BEGIN/END DECLARE SECTION
- BEGIN/END COMPOUND
- INCLUDE
- WHENEVER
- COMMIT 및 ROLLBACK
- CONNECT
- OPEN cursor
- FETCH
- CLOSE cursor
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- DESCRIBE
- 동적 DECLARE CURSOR
- SQL 제어 명령문

복합 SQL문 내의 각 부속문은 각자의 섹션을 가지며, 이는 **db2expln**으로 설명될 수 있습니다.

dynexpln 구문 및 매개변수



여기서

-d database name

Explain될 패키지를 포함하는 데이터베이스의 이름.

이 옵션을 지정하지 않으면 입력하라는 메시지가 나타납니다.

-f input file

Explain되는 SQL문이 들어 있는 파일 이름.

명령문 종료문자(**-e**) 옵션을 사용하지 않을 경우, 파일의 각 행에는 SQL문만이 표시됩니다. SQL 주석이 파일에 입력됩니다. SQL 주석은 **--**로 시작하여 행의 끝까지 이어집니다.

-g 최적화 알고리즘 플랜 그래프를 표시합니다. 각 섹션이 검토되고 원래의 최적화 알고리즘 플랜 그래프(Visual Explain에 의해 표시됨)가 구성됩니다. 생성된 그래프가 원래 플랜과 일치하지 않을 수 있다는 점을 유의하십시오.

-h 또는 -?

입력 매개변수에 관한 도움말 정보를 얻습니다. 이 옵션의 지정은 다른 모든 옵션을 겹쳐씁니다.

-i Explain된 플랜에 연산자 ID를 표시합니다. 연산자 ID로 db2expln의 출력을 Explanin 기능의 출력과 일치시킬 수 있습니다.

-o output file

db2expln이 결과를 작성할 파일 이름.

-t 출력이 터미널로 방향이 지정되었습니다.

출력(**-o**) 및 **-t** 옵션이 모두 지정될 경우, 출력은 터미널 방향으로 지정됩니다.

출력 파일(**-o**) 또는 **-t** 옵션을 지정하지 않으면, 기본 출력이 터미널에 표시됨과 동시에 사용자에게 파일 이름에 대해 프롬프트가 표시됩니다.

-q SQL statement

Explain될 SQL문

이 옵션을 지정하지 않고 입력 파일(**-f**)의 선택적 매개변수를 지정하지 않을 경우, Explain될 SQL문을 제공하도록 프롬프트가 표시됩니다.

이 옵션과 입력 파일(**-f**)의 선택적 매개변수를 모두 지정할 경우, dynexpln은 우선 SQL문(**-s**) 옵션에 의해 제공되는 명령문을 설명한 다음, 입력 파일(**-f**)의 명령문을 설명합니다.

-u userID password

데이터베이스에 연결할 때 제공된 사용자 ID와 암호를 사용하십시오.

사용자 ID와 암호는 DB2 이름 지정 규칙에 맞게 유효해야 하며, 데이터베이스가 인식할 수 있어야 합니다.

-z statement terminator

이 문자는 SQL문의 끝에 도달했음을 나타내는 데 사용됩니다.

기본값은 명령문 종료문자가 없는 상태입니다. 이 옵션을 사용하지 않을 경우, 파일의 각 행은 별도의 SQL문인 것으로 가정됩니다. 이 옵션을 사용할 경우, dynexpln은 지정된 종료 문자를 사용하여 명령문을 구분합니다.

위 테이블의 일부 옵션 플래그는 운영 체제에 따라 특수 의미를 띄기 때문에 dynexpln 명령행에서 정확하게 해석되지 않을 수 있습니다. 그러나 이 문자 앞에 Escape 문자를 입력할 수는 있습니다. 자세한 내용은 운영 체제 사용자 매뉴얼을 참조하십시오.

명령문 종료 문자(**-z**) 옵션을 사용할 경우, SQL문(**-s**) 옵션을 사용하여 다중 명령문을 입력할 수 있습니다. 이렇게 할 경우, 명령문을 종료 문자로 구분해야 합니다.

dynexpln에 의해 생성된 도움말과 초기 상태 메시지는 표준 출력에 작성됩니다. Explain 도구에 의해 생성된 모든 프롬프트와 다른 상태 메시지는 표준 오류에 작성됩니다. 선택된 출력 옵션에 따라 Explain 텍스트는 표준 출력 장치 또는 파일로 작성됩니다.

예를 들어, SAMPLE이라는 이름의 데이터베이스에 연결하여 TRYIT 파일의 모든 명령문을 설명하고, 그 결과를 my.exp 파일에 작성하려면, 다음과 같이 입력하십시오.

```
dynexpln -d SAMPLE -f TRYIT -o my.exp
```

dynexpln 사용에 관한 유의사항

dynexpln은 동적문을 설명하기 위해 명령문에 대해 정적 응용프로그램을 작성한 다음 db2expln을 호출합니다. 정적문을 작성하기 위해 dynexpln은 명령문으로 간단한 C 프로그램을 생성한 다음 DB2 사전 처리 컴파일러를 호출하여 패키지를 작성합니다. (생성된 C 프로그램은 완료되지 않으며 컴파일될 수 없습니다. 이 프로그램에는 사전 처리 컴파일러가 패키지를 빌드하는 데 필요한 충분한 정보만이 들어 있습니다.)

다음은 dynexpln에 의해 표시된 공통 메시지입니다.

- db2expln에서 모든 오류 메시지

  dynexpln은 db2expln을 호출한 이후에는 대부분의 db2expln 오류 메시지를 볼 수 있습니다.

- 데이터베이스 연결에 오류가 발생하였습니다.

  데이터베이스에 연결하면서 오류가 발생할 경우, 이 메시지가 표시됩니다. 연결이 완료되지 않은 이유를 나타내는 CLI 오류 메시지도 표시됩니다. 오류 원인을 정정하고 dynexpln을 다시 수행하십시오.

- "<파일 이름>" 파일은 dynexpln가 수행되기 전에 제거해야 합니다.

  dynexpln이 수행될 때 주어진 파일이 존재할 경우, 이 메시지가 표시됩니다. 파일을 제거하거나 DYNEXPLN_PACKAGE 환경 변수 값을 변경하여 작성될 파일 이름을 변경하고 dynexpln을 다시 수행하십시오.

- "<작성자>.<패키지>" 패키지는 dynexpln가 수행되기 전에 제거해야 합니다.

  dynexpln이 수행될 때 주어진 패키지가 존재할 경우, 이 메시지가 표시됩니다. 패키지를 삭제하고 **DYNEXPLN_PACKAGE** 환경 변수 값을 수행 또는 변경하여, 작성될 패키지 이름을 변경하고 dynexpln을 다시 수행하십시오.

- "<파일 이름>" 파일 쓰기 오류가 발생하였습니다.

주어진 파일을 작성할 수 없을 때 이 메시지가 표시됩니다. dynexpln이 현재 디렉토리에 파일을 작성할 수 있는지 확인하고 다시 수행하십시오.

- "<파일 이름>" 입력 파일 읽기 오류가 발생하였습니다.

  -f 옵션으로 주어진 파일을 읽을 수 없을 때 이 메시지가 표시됩니다. 파일이 존재하며 dynexpln이 파일을 읽을 수 있는지 확인하십시오. 그런 다음 dynexpln을 다시 수행하십시오.

환경 변수: dynexpln과 함께 사용될 수 있는 환경 변수에는 다음의 두 가지가 있습니다.

- **DYNEXPLN_OPTIONS**은 명령문에 대해 패키지를 빌드할 때 사용할 SQL 사전 처리 컴파일러 옵션입니다. CLP를 통해 PREP 명령을 발행할 때에는 동일한 구문 변수를 사용하십시오.

  예: DYNEXPLN_OPTIONS="OPTLEVEL 5 BLOCKING ALL"

- **DYNEXPLN_PACKAGE**는 데이터베이스에서 작성되는 패키지 이름입니다. 설명될 명령문이 이 패키지에 배치됩니다. 이 변수가 정의되어 있지 않을 경우, 패키지에는 기본값인 **DYNEXPLN**이 제공됩니다. (이 환경 변수에서 이름 중 처음 8자만이 사용됩니다.)

  또한 이름은 dynexpln이 사용하는 중간 파일 이름을 작성하는 데에도 사용됩니다.

db2expln 및 dynexpln 출력에 대한 설명

출력에서 각 패키지에 대한 Explain 정보는 다음의 두 부분으로 구분됩니다.

- 바인드 날짜 및 바인드 옵션 등과 같은 패키지 정보
- Explain될 SQL문에 붙은 섹션 번호와 같은 섹션 정보. 섹션 정보 아래쪽은 표시된 SQL문에 대해 선택된 액세스 플랜의 Explain 출력이 됩니다.

액세스 플랜 또는 섹션의 단계는 데이터베이스 관리 프로그램이 실행한 순서대로 나타납니다. 각 주요 단계는 왼쪽 정렬로 표시되며, 해당 단계에 관한 정보가 바로 밑에 들여쓰기로 표시됩니다. 액세스 플랜에 대한 Explain 출력에는 출력물의 왼쪽 여백에 들여쓰기 막대가 표시됩니다. 이 막대는 또한 연산 "범위"를 나타내

기도 합니다. 같은 연산 내의 낮은 레벨의 연산(즉 오른쪽으로 갈수록 높은 수준의 연산)은 이전의 들여쓰기 레벨로 리턴되기 전에 처리됩니다.

선택된 액세스 플랜이 원래의 SQL문의 확대된 버전(출력에 나타난 것)임을 기억하십시오. 예를 들면, 원래의 명령문은 개수에 상관없이 트리거와 제한조건을 활성화시킬 수도 있습니다. 또한 SQL문이 SQL 컴파일러의 조회 재작성 구성요소에 의해 같은 내용의, 그러나 더 효율적인 형식으로 재작성될 수도 있습니다. 이러한 모든 요소는 최적화 알고리즘이 명령문을 충족시킬 가장 효율적인 액세스 플랜을 결정할 때, 최적화 알고리즘에 제공되는 정보에 포함되어 있습니다. 이렇게 Explain 출력에 나타나는 액세스 플랜은 원래의 SQL문에 대해 예상하는 액세스 플랜과는 근본적으로 다를 수도 있습니다. 통합 Explain 기능(243 페이지의 『제7장 SQL Explain 기능』 참조)은 실제 최적화에 사용되는 SQL문을 SQL과 유사한 명령문의 양식으로 보여주는데, 이는 내부적으로 표시되는 조회의 형태를 역변환(reverse-translating)시켜서 생성된 것입니다.

db2expln 또는 dynexpln의 출력을 Explain 기능의 출력과 비교할 때 연산자 ID 옵션(**-i**)을 사용하면 매우 유용합니다. db2expln 또는 dynexpln이 Explain 기능을 통해 새 연산자 처리를 시작할 때마다, Explain된 플랜의 왼쪽에 연산자 ID 번호가 인쇄됩니다. 액세스 플랜의 다양한 표현에서의 단계를 맞추는 데 연산자 ID를 사용할 수 있습니다. Explain 기능 출력의 연산자와 db2expln 및 dynexpln에 의해 표시되는 조작간에 항상 일 대 일 대응이 있는 것은 아님에 유의하십시오.

다음 주제에서는 db2expln 및 dynexpln에 의해 생성될 수 있는 Explain 텍스트에 대해 설명합니다.

- 테이블 액세스
- 임시 테이블
- 조인
- 데이터 스트림
- 삽입, 갱신 및 삭제
- 행 ID(RID) 준비
- 총계

- 병렬 처리
- 연합 명령문 처리
- 기타 명령문

테이블 액세스

이 명령문에서는 액세스되는 테이블 이름과 유형을 보여줍니다. 여기에는 두 가지 형식이 사용됩니다.

1. 세 가지 유형의 일반 테이블:
   - 액세스 테이블 이름

     ```
     Access Table Name = schema.name  ID = ts,n
     ```

     여기서
     - *schema.name*은 액세스되는 테이블의 완전한 이름입니다.
     - *ID*는 테이블의 SYSCAT.TABLES 카탈로그에 있는 TABLESPACEID 및 TABLEID에 해당합니다.
   - 액세스 계층 구조 테이블 이름

     ```
     Access Hierarchy Table Name = schema.name  ID = ts,n
     ```

     여기서
     - *schema.name*은 액세스되는 테이블의 완전한 이름입니다.
     - *ID*는 테이블의 SYSCAT.TABLES 카탈로그에 있는 TABLESPACEID 및 TABLEID에 해당합니다.
   - 액세스 요약 테이블 이름

     ```
     Access Summary Table Name = schema.name  ID = ts,n
     ```

     여기서
     - *schema.name*은 액세스되는 테이블의 완전한 이름입니다.
     - *ID*는 테이블의 SYSCAT.TABLES 카탈로그에 있는 TABLESPACEID 및 TABLEID에 해당합니다.
2. 두 가지 유형의 임시 테이블:
   - 액세스 임시 테이블 ID

```
Access Temp Table  ID = tn
```

여기서

– *ID*는 db2expln에 의해 지정되는 해당 식별자입니다.

- 선언된 액세스 전역 임시 테이블 ID

```
Access Global Temp Table  ID = ts,tn
```

여기서

– *ID*는 테이블의 SYSCAT.TABLES 카탈로그에 있는 TABLESPACEID 및 TABLEID에 해당되며(ts), 해당되는 식별자는 db2expln에 의해 지정됩니다(tn).

테이블 액세스문에 이어 액세스를 더 자세히 설명하기 위해 추가 명령문이 제공됩니다. 이 명령문은 테이블 액세스문 밑에 들여쓰기됩니다. 가능한 명령문은 다음과 같습니다.

- 컬럼의 수
- 병렬 스캔
- 스캔 방향
- 행 액세스 메소드
- 잠금 모드
- 술어
- 기타 테이블 명령문

컬럼의 수

다음 명령문은 각 테이블 행에서 사용되는 컬럼의 수를 나타냅니다.

```
#Columns = n
```

병렬 스캔

다음 명령문은 데이터베이스 관리 프로그램이 몇몇 서브에이전트를 사용하여 테이블로부터 병렬로 읽을 것임을 나타냅니다.

```
Parallel Scan
```

이 텍스트가 표시되지 않을 경우, 테이블은 한 에이전트(또는 서브에이전트)에 의해서만 읽혀질 수 있습니다.

스캔 방향

다음 명령문은 데이터베이스 관리 프로그램이 행을 역순으로 읽을 것임을 나타냅니다.

```
Scan Direction = Reverse
```

이 텍스트가 나타나지 않으면 스캔 방향은 진행 방향이며, 이는 기본값입니다.

행 액세스 메소드

다음 명령문 중 하나가 표시되어 테이블의 규정 행에 액세스하는 방식을 나타냅니다.

- Relation Scan문은 테이블이 규정 행을 찾기 위해 순차적으로 스캔 중임을 나타냅니다.
  - 다음 명령문은 데이터의 프리페치가 수행되지 않음을 나타냅니다.

    ```
    Relation Scan
    |  Prefetch: None
    ```

  - 다음 명령문은 최적화 알고리즘이 프리페치될 페이지의 수를 미리 결정했음을 나타냅니다.

    ```
    Relation Scan
    |  Prefetch: n Pages
    ```

  - 다음 명령문은 데이터가 프리페치되어야 함을 나타냅니다.

    ```
    Relation Scan
    |  Prefetch: Eligible
    ```

  - 다음 명령문은 규정 행이 색인을 통해 식별되고 액세스됨을 나타냅니다.

    ```
    Index Scan:  Name = schema.name  ID = xx
    |  Index Columns:
    ```

    여기서

    - *schema.name*은 스캔 중인 색인의 완전한 이름입니다.
    - *ID*는 SYSCAT.INDEXES 카탈로그 뷰에 있는 대응하는 ID 컬럼입니다.

색인의 각 컬럼의 뒤에 하나의 행이 와야 합니다. 색인의 각 컬럼은 다음 양식 중 하나로 나열됩니다.

```
n: column_name(Ascending)
n: column_name(Descending)
n: column_name(Include Colum)
```

다음 명령문은 색인 스캔의 유형을 명확히 보여주기 위해 제공됩니다.

- 다음에 표시된 색인에 대한 범위 구분 술어

  ```
  #Key Columns = n
  |  Start Key: xxxxx
  |  Stop Key: xxxxx
  ```

  여기서 xxxxx는 다음 중 하나입니다.

  - Start of Index
  - End of Index
  - Inclusive Value: 또는 Exclusive Value:

    포괄 키 값은 색인 스캔에 포함됩니다. 독점 키 값은 스캔에 포함되지 않습니다. 키의 각 부분에 대한 키 값은 다음 행 중 하나로 제공됩니다.

    ```
    n: 'string'
    n: nnn
    n: yyyy-mm-dd
    n: hh:mm:ss
    n: yyyy-mm-dd hh:mm:ss.uuuuuu
    n: NULL
    n: ?
    ```

    리터럴 문자열이 표시되는 경우, 처음 20자만이 표시됩니다. 문자열이 21자 이상인 경우에는 뒤에 ... 문자열이 표시됩니다. 일부 키는 섹션이 실행될 때까지 결정할 수 없습니다. 이러한 키는 값으로 ?가 표시됩니다.

- Index-Only Access

  필요한 모든 컬럼이 색인 키로부터 구해질 수 있을 경우, 이 명령문이 나타나고 테이블 데이터에 액세스하지 않습니다.

- 다음 명령문은 색인 페이지의 프리페치가 수행되지 않음을 나타냅니다.

```
Index Prefetch: None
```

- 다음 명령문은 색인 페이지가 프리페치되어야 함을 나타냅니다.

```
Index Prefetch: Eligible
```

- 다음 명령문은 데이터 페이지의 프리페치가 수행되지 않음을 나타냅니다.

```
Data Prefetch: None
```

- 다음 명령문은 데이터 페이지가 프리페치되어야 함을 나타냅니다.

```
Data Prefetch: Eligible
```

- 색인 항목 규정에 도움을 주기 위해 색인 관리 프로그램으로 전달될 수 있는 술어가 있는 경우, 다음 명령문이 술어의 수를 나타내는 데 사용됩니다.

```
Sargable Index Predicate(s)
|  #Predicates = n
```

– Fetch Direct문은 이전에 액세스 플랜에 준비된 행 ID(RID)를 사용하여 규정 행에 액세스하고 있음을 나타냅니다.

잠금 모드

각 테이블 액세스의 경우, 다음 명령문을 사용하면 테이블 및 행 레벨에서 확보되는 잠금의 유형이 표시됩니다.

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

테이블 잠금에 대한 가능한 값은 다음과 같습니다.

- 독점
- 부분 독점
- 잠금 없음
- 부분 공유
- 공유
- 부분 독점이 있는 공유
- 강한 독점
- 갱신

행 잠금에 대한 가능한 값은 다음과 같습니다.

- 독점
- 다음 키 독점(db2expln 출력에는 나타나지 않음)
- 없음
- 공유
- 다음 키 공유
- 갱신
- 다음 키 약한 독점
- 약한 독점

잠금 유형은 56 페이지의 『잠금 속성』에 설명되어 있습니다.

술어

액세스 플랜에 사용되는 술어에 대한 정보를 제공하는 명령문은 다음 두 가지입니다.

1. 다음 명령문은 일단 데이터가 리턴되면 평가될 술어의 수를 나타냅니다.

```
Residual Predicate(s)
|  #Predicates = n
```

2. 다음의 명령문은 일단 데이터에 액세스하면 평가될 술어의 수를 나타냅니다. 술어 계수는 총계 또는 정렬과 같은 푸시다운 조작은 포함하지 않습니다.

```
Sargable Predicate(s)
|  #Predicates = n
```

위 명령문에 나타난 술어에 다음 현상이 나타날 수 있으므로, SQL문에 제공된 술어의 수를 반영하지 않을 수 있습니다.

- 같은 조회 내에서 두 번 이상 적용됨
- 조회 최적화 프로세스가 진행되는 동안에 내재적 술어가 추가되어 변환 또는 확장됨
- 조회 최적화 프로세스가 진행되는 동안에 더 작은 개수의 술어로 변환 또는 압축됨

기타 테이블 명령문

- 다음 명령문은 하나의 행에만 액세스됨을 나타냅니다.

  ```
  Single Record
  ```

- 다음 명령문은 이 테이블 액세스에 사용되는 분리 레벨이 패키지와는 다른 분리 레벨을 사용할 때 나타납니다.

  ```
  Isolation Level: xxxx
  ```

  다음과 같은 여러 가지 이유로 다른 분리 레벨이 사용될 수 있습니다.

  – 패키지가 반복 읽기(RR)와 바인드되어 참조 무결성 제한조건에 영향을 주는 경우입니다. 참조 무결성 제한조건을 점검하기 위해 상위 테이블에 액세스하면 이 테이블에 대한 불필요한 잠금보유를 피하기 위해 커서 안정성(CS) 분리 레벨로 낮춰집니다.

  – 미확약 읽기(UR)와 바인드된 패키지가 DELETE 및 UPDATE문을 발행한 경우입니다. 실제 삭제에 대한 테이블 액세스가 커서 안정성(CS) 분리 레벨로 올라가게 됩니다.

- 다음 명령문은 사용 가능한 정렬 힙 메모리가 충분한 경우, 임시 테이블에서 읽은 행의 일부 또는 전체가 버퍼 풀 밖에서 캐쉬될 것임을 나타냅니다.

  ```
  Keep Rows In Private Memory
  ```

- 테이블이 소멸성 기본 행 수(cardinality) 속성 세트를 가질 경우, 다음과 같이 표시됩니다.

  ```
  Volatile Cardinality
  ```

임시 테이블

임시 테이블은 액세스 플랜이 실행되는 동안 전이 및 임시 작업 테이블에 데이터를 저장하기 위해 액세스 플랜에 의해 사용됩니다. 이 테이블은 액세스 플랜이 실행되는 동안에만 존재합니다. 일반적으로, 임시 테이블은 액세스 플랜 초기에 부속 조회가 평가되어야 할 필요가 있는 경우 또는 중간 결과가 사용 가능한 메모리에 맞지 않는 경우에 사용됩니다.

임시 테이블이 생성될 필요가 있을 경우, 두 가지 명령문 중 하나가 나타납니다. 이들 명령문은 임시 테이블이 작성되고 행이 테이블에 삽입되는 것을 나타냅니다.

ID는 임시 테이블을 언급할 때 편리하도록 db2expln이 지정한 식별자입니다. 이 ID는 앞에 ‘t’를 붙여 테이블이 임시 테이블임을 나타냅니다.

- 다음 명령문은 일반 임시 테이블이 작성됨을 나타냅니다.

```
Insert Into Temp Table  ID = tn
```

- 다음 명령문은 임시 테이블이 다중 서브에이전트에 의해 병렬로 작성됨을 나타냅니다.

```
Insert Into Shared Temp Table  ID = tn
```

- 다음 명령문은 정렬된 임시 테이블이 작성됨을 나타냅니다.

```
Insert Into Sorted Temp Table  ID = tn
```

- 다음 명령문은 정렬된 임시 테이블이 다중 서브에이전트에 의해 병렬로 작성됨을 나타냅니다.

```
Insert Into Sorted Shared Temp Table  ID = tn
```

- 다음 명령문은 선언된 전역 임시 테이블이 작성됨을 나타냅니다.

```
Insert Into Global Temp Table  ID = ts,tn
```

- 다음 명령문은 선언된 임시 테이블이 다중 서브에이전트에 의해 병렬로 작성됨을 나타냅니다.

```
Insert Into Shared Global Temp Table  ID = ts,tn
```

- 다음 명령문은 정렬되어 있는 선언된 전역 임시 테이블이 작성됨을 나타냅니다.

```
Insert Into Sorted Global Temp Table  ID = ts,tn
```

- 다음 명령문은 정렬되어 있는 선언된 임시 테이블이 다중 서브에이전트에 의해 병렬로 작성됨을 나타냅니다.

```
Insert Into Sorted Shared Global Temp Table  ID = ts,tn
```

위의 각 명령문 뒤에는 다음 항목이 나옵니다.

```
#Columns = n
```

이는 임시 테이블에 삽입되고 있는 각 행에 얼마나 많은 수의 컬럼이 있는지를 나타냅니다.

정렬된 임시 테이블

정렬된 임시 테이블은 다음 조작으로부터 작성될 수 있습니다.

- ORDER BY
- DISTINCT
- GROUP BY
- 병합 조인
- '= ANY' 부속 조회
- '<> ALL' 부속 조회
- INTERSECT 또는 EXCEPT
- UNION(ALL 키워드 없이)

여러 추가 명령문이 정렬된 임시 테이블에 대한 원래의 작성문 뒤에 따라 나올 수 있습니다.

- 다음 명령문은 정렬에서 사용되는 키 컬럼의 수를 나타냅니다.

  ```
  #Sort Key Columns = n
  ```

  정렬 키의 각 컬럼에 대해 다음 행 중 하나가 표시됩니다.

  ```
  Key n: column_name (Ascending)
  Key n: column_name (Descending)
  Key n: (Ascending)
  Key n: (Descending)
  ```

- 다음 명령문은 런타임 시 최적 정렬 힙이 할당될 수 있도록 행 수와 행 크기 추정치를 제공합니다.

  ```
  Sortheap Allocation Parameters:
  |  #Rows     = n
  |  Row Width = n
  ```

- 정렬된 결과의 첫 번째 행만 필요한 경우, 다음이 표시됩니다.

  ```
  Sort Limited To Estimated Row Count
  ```

- 대칭적 멀티프로세서(SMP) 환경의 정렬에서 수행되는 정렬의 유형은 다음 명령문 중 하나로 표시됩니다.

```
Use Partitioned Sort
Use Shared Sort
Use Replicated Sort
Use Round-Robin Sort
```

다른 정렬 기법에 대한 자세한 내용은 221 페이지의 『병렬 정렬 전략』을 참조하십시오.

- 다음 명령문은 정렬의 결과가 정렬 힙에 유지되는지 여부를 나타냅니다.

  ```
  Piped
  ```

  그리고

  ```
  Not Piped
  ```

  파이프 정렬을 표시할 경우, 데이터베이스 관리 프로그램은 정렬된 결과를 다른 임시 테이블에 놓기보다는 메모리의 정렬된 출력에 보존합니다. (파이프 정렬 대비 파이프된 정렬에 대한 자세한 내용은 217 페이지의 『정렬이 최적화 알고리즘에 미치는 영향』을 참조하십시오.)

- 다음 명령문은 중복 값이 정렬 중에 제거됨을 나타냅니다.

  ```
  Duplicate Elimination
  ```

- 정렬에서 총계가 수행되는 경우, 다음 명령문 중 하나로 표시됩니다.

  ```
  Partial Aggregation
  Intermediate Aggregation
  Buffered Partial Aggregation
  Buffered Intermediate Aggregation
  ```

임시 테이블 완료

임시 테이블을 생성하기 위한 푸시다운 조작을 포함한 테이블 액세스 후에(즉, 테이블 액세스의 범위 내에서 발생하는 임시 테이블 작성), ″완료(completion)″ 명령문이 나타나게 되는데, 이는 임시 테이블이 행을 제공하여 후속 임시 테이블 액세스에 대비하도록 함으로써 파일의 끝(end-of-file)을 처리하는 것입니다. 다음 행 중 하나가 표시됩니다.

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

테이블 함수

테이블 함수는 데이터를 테이블 형식으로 명령문에 리턴하는 사용자 정의 함수(UDF)입니다. 테이블 함수에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오. 테이블 함수는 다음과 같은 명령으로 표시됩니다.

```
Access User Defined Table Function
|   Name = schema.funcname
|   Language = xxxx
|   Fenced   Deterministic   NULL Call   Disallow Parallel
```

테이블 함수가 작성된 언어(C, OLE 또는 Java)는 테이블 함수의 속성과 함께 부여됩니다.

조인

조인에는 세 가지 유형이 있습니다. (이들 조인에 대한 자세한 내용은 197 페이지의 『조인 개념』을 참조하십시오.)

- 해쉬 조인
- 병합 조인
- 중첩된 루프 조인

섹션 실행에서 조인이 수행되는 데 걸리는 시간에 다음 명령문 중 하나가 표시됩니다.

```
Hash Join
```

또는

```
Merge Join
```

또는

```
Nested Loop Join
```

왼쪽 외부 조인이 수행될 수 있습니다. 다음 명령문 중 하나가 왼쪽의 외부 조인을 표시합니다.

```
Left Outer Hash Join
```

또는

```
Left Outer Merge Join
```

또는

```
Left Outer Nested Loop Join
```

병합 및 중첩된 루프 조인의 경우, 조인의 외부 테이블이 출력에 나타나는 이전 액세스 명령문에서 참조되는 테이블이 됩니다. 조인의 내부 테이블은 조인문의 범위 내에 포함되어 있는 액세스 명령문에서 참조된 테이블이 됩니다. 해쉬 조인의 경우, 액세스 명령문에서 조인의 범위 내에 포함된 외부 테이블 및 조인 앞에 나타나는 내부 테이블이 서로 바뀝니다.

해쉬 또는 병합 조인의 경우, 다음 명령문이 추가로 표시될 수 있습니다.

- 일부 환경에서 조인은 단지 내부 테이블이 외부에 있는 현재 행과 일치하는지 판별해야 할 때가 있습니다. 이것은 다음 명령문으로 표시됩니다.

  ```
  Early Out: Single Match Per Outer Row
  ```

- 조인이 완료된 다음 술어를 적용할 수 있습니다. 적용되고 있는 술어의 수는 다음과 같이 표시됩니다.

  ```
  Residual Predicate(s)
  #Predicates = n
  ```

해쉬 조인의 경우, 다음 명령문이 추가로 표시될 수 있습니다.

- 해쉬 테이블은 내부 테이블로부터 빌드됩니다. 해쉬 테이블 빌드가 내부 테이블 액세스의 술어로 푸시다운되면 내부 테이블의 액세스에 다음 명령문으로 표시됩니다.

  ```
  Process Hash Table For Join
  ```

- 외부 테이블에 액세스하는 중에 조인의 성능을 향상시키기 위해 조사(probe) 테이블이 빌드될 수 있습니다. 조사(probe) 테이블 빌드는 외부 테이블의 액세스에 다음 명령문으로 표시됩니다.

  ```
  Process Probe Table For Hash Join
  ```

- 해쉬 테이블을 빌드하는 데 필요한 추정 바이트 수는 다음으로 표시됩니다.

  ```
  Estimated Build Size: n
  ```

- 조사 테이블에 필요한 추정 바이트 수는 다음으로 표시됩니다.

  ```
  Estimated Probe Size: n
  ```

중첩된 루프 조인의 경우, 다음과 같은 추가 명령문이 조인 명령문 바로 다음에 나올 수 있습니다.

```
Piped Inner
```

이 명령문은 조인의 내부 테이블이 다른 일련의 연산의 결과임을 나타냅니다. 이는 또한 *복합 내부*로서 참조되기도 합니다.

조인에 세 개 이상의 테이블이 포함되는 경우, 맨 위에서부터 아래까지 Explain 단계를 전부 읽어야 합니다. 예를 들면, Explain 출력 흐름이 다음과 같다고 가정해 보십시오.

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

실행 단계는 다음과 같습니다.

1. W로부터 규정된 행을 가져옵니다.
2. W의 행을 X의 (다음) 행과 조인하고 결과 P1(부분 조인 결과 번호 1에 대한)을 호출합니다.
3. P1을 Y의 (다음) 행과 조인하여 P2를 작성합니다.
4. P2를 Z의 (다음) 행과 조인하여 하나의 완전한 결과 행을 구합니다.
5. Z에 행이 더 있으면 4단계로 가십시오.
6. Y에 행이 더 있으면 3단계로 가십시오.
7. X에 행이 더 있으면 3단계로 가십시오.
8. W에 행이 더 있으면 1단계로 가십시오.

데이터 스트림

액세스 플랜 내에서 때로 데이터의 작성 및 흐름을 하나의 연산에서 다른 연산으로 제어할 필요가 있는 경우가 있습니다. 데이터 스트림 개념은 액세스 플랜 내에서 연산 그룹이 하나의 단위로 제어될 수 있도록 해줍니다. 데이터 스트림의 시작은 다음 명령문으로 표시됩니다.

```
Data Stream n
```

여기서 n은 참조하기 쉽도록 하기 위해 db2expln이 지정한 고유 식별자입니다. 데이터 스트림의 끝은 다음으로 표시됩니다.

```
End of Data Stream n
```

이 명령문간의 모든 연산은 같은 데이터 스트림으로 간주됩니다.

데이터 스트림은 몇 가지 특성을 가지고 있으며 하나 이상의 명령문이 이러한 특성을 설명하기 위해 초기 데이터 스트림문 뒤에 따라올 수 있습니다.

- 데이터 스트림의 작동이 액세스 플랜에서 이전에 생성된 값에 따라 다를 경우, 데이터 스트림은 다음과 같이 표시됩니다.

  ```
  Correlated
  ```

- 정렬된 임시 테이블과 마찬가지로 다음 명령문은 데이터 스트림의 결과가 메모리에 보관되는지를 나타냅니다.

  ```
  Piped
  ```

  그리고

  ```
  Not Piped
  ```

  파이프 데이터 스트림은 임시 테이블에서처럼 실행시 메모리가 부족한 경우, 디스크에 작성될 수도 있습니다. 액세스 플랜은 두 가지 가능성을 모두 제공합니다.

- 다음 명령문은 단일 레코드만이 이러한 데이터 스트림으로부터 요구됨을 나타냅니다.

  ```
  Single Record
  ```

데이터 스트림에 액세스되면 다음 데이터 스트림이 출력에 나타납니다.

```
Access Data Stream n
```

삽입, 갱신 및 삭제

이들 SQL문에 대한 Explain 텍스트는 자체 설명적입니다. 이러한 SQL 연산에 대한 가능한 명령문 텍스트는 다음이 될 수 있습니다.

- `Insert: Table Name = schema.name ID = ts,n`

- Update: Table Name = schema.name ID = ts,n
- Delete: Table Name = schema.name ID = ts,n
- Insert: Hierarchy Table Name = schema.name ID = ts,n
- Update: Hierarchy Table Name = schema.name ID = ts,n
- Delete: Hierarchy Table Name = schema.name ID = ts,n
- Insert: Summary Table Name = schema.name ID = ts,n
- Update: Summary Table Name = schema.name ID = ts,n
- Delete: Summary Table Name = schema.name ID = ts,n
- Insert: Global Temporary Table ID = ts, tn
- Update: Global Temporary Table ID = ts, tn
- Delete: Global Temporary Table ID = ts, tn

행 ID(RID) 준비

일부 액세스 플랜의 경우, 규정 행 ID(RID)가 정렬되고 중복이 제거되거나(색인 OR 작업의 경우) 기술을 사용하여 실제 테이블 액세스가 수행되기 전에 액세스 중인 모든 색인에 표시되는 RID를 식별하는 (색인 AND 작업의 경우) 보다 유용합니다. Explain문에 의해 표시된 것처럼 다음과 같은 세 가지 경우에 주로 RID 준비가 사용됩니다.

- 다음 명령문은 『색인 OR 작업』이 규정 RID 목록을 준비하는 데 사용됨을 나타냅니다.

  ```
  Index ORing RID Preparation
  ```

  *색인 OR 작업*은 둘 이상의 색인 액세스를 만들고, 액세스된 모든 색인에 표시되는 구별 RID가 포함되도록 그 결과를 결합하는 기법을 말합니다. 최적화 알고리즘은 술어가 OR 키워드에 의해 연결되거나 IN 술어가 있을 때 색인 OR 작업을 고려합니다. 색인 액세스는 같거나 다른 색인들 위에서 가능합니다.
- 또한 RID 준비는 다음에 지정된 것처럼 목록 프리페치 중에 사용될 입력 데이터를 준비하는 데도 사용됩니다.

  ```
  List Prefetch RID Preparation
  ```

- *색인 AND 작업*은 하나 이상의 색인 액세스를 만들고, 액세스된 모든 색인에 나타나는 RID가 포함되도록 그 결과를 결합하는 기법을 말합니다. 색인 AND 작업 처리는 다음 명령문으로 시작됩니다.

  ```
  Index ANDing
  ```

  최적화 알고리즘이 결과 세트의 *크기*를 추정한 경우, 추정치는 다음 명령문으로 표시됩니다.

  ```
  Optimizer Estimate of Set Size: n
  ```

  색인 AND 작업 필터 조작은 RID를 처리하고 비트 필터 기술을 사용하여 액세스되는 모든 색인에 표시되는 RID를 판별합니다. 다음 명령문은 RID가 색인 AND 작업에 대해 처리됨을 나타냅니다.

  ```
  Index ANDing Bitmap Build
  Index ANDing Bitmap Probe
  Index ANDing Bitmap Build and Probe
  ```

  최적화 알고리즘이 비트맵의 결과 세트 *크기*를 추정한 경우, 추정치는 다음 명령문으로 표시됩니다.

  ```
  Optimizer Estimate of Set Size: n
  ```

어떠한 유형의 RID 준비의 경우에도 목록 프리페치를 수행할 수 있으면 다음 명령문으로 표시됩니다.

```
Prefetch: Enabled
```

총계

총계는 SQL문의 술어가 제공하는 특정 기준을 충족시키는 행에 대해 수행됩니다. 일부 총계 함수의 정렬이 수행되면 다음 명령문 중 하나가 나타납니다.

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

술어 총계는 데이터가 실제로 액세스될 때 술어로 처리되기 위해 푸시다운되는 총계 연산을 말합니다.

위의 총계문 중 하나의 총계문 아래에는 수행되는 해당 총계 함수 유형이 표시됩니다.

- `Group By`
- `Column Function(s)`
- `Single Record`

특정 컬럼 함수가 원래의 SQL문에서 파생될 수 있습니다. 단일 레코드는 MIN 또는 MAX 연산을 충족시키는 색인으로부터 페치될 수 있습니다.

술어 총계가 사용된 다음, 총계가 나타난 테이블 액세스문이 후속되는 경우 총계 "완료(completion)"가 나오는데, 이는 각 그룹의 완료 또는 파일의 끝(end-of-file)에 대한 필요한 처리를 수행합니다. 다음 행 중 하나가 표시됩니다.

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

병렬 처리

SQL문을 병렬로 실행하려면(파티션 내 또는 파티션간 병렬 처리 사용), 일부 특수 조작이 필요합니다. 병렬 플랜의 조작은 다음과 같이 이루어집니다.

- 파티션 내 병렬 플랜을 수행할 때 몇몇 서브에이전트를 사용하여 플랜의 일부가 동시가 실행됩니다. 서브에이전트의 작성은 다음 명령문으로 표시됩니다.

  ```
  Process Using n Subagents
  ```

- 파티션간 병렬 플랜을 수행할 때, 섹션은 여러 서브섹션으로 나누어집니다. 각 서브섹션은 수행될 하나 이상의 노드로 전송됩니다. 중요한 서브섹션은 *조정자 서브섹션*입니다. 조정자(coordinator) 서브섹션은 모든 플랜의 첫 번째 서브섹션으로서 제어를 확보하고 다른 서브섹션을 분산시키며, 결과를 호출 응용프로그램으로 리턴하는 책임이 있습니다.

  서브섹션의 분산은 다음 명령문으로 표시됩니다.

  ```
  Distribute Subsection #n
  ```

서브섹션을 수신하는 노드는 다음 8가지 방법 중 하나로 결정될 수 있습니다.

- 다음은 컬럼 값에 따라 서브섹션이 노드 그룹 내의 노드로 전송됨을 나타냅니다.

```
Directed by Hash
|  #Columns = n
|  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
```

- 다음은 서브섹션이 사전 결정된 노드로 전송됨을 나타냅니다. (명령문이 NODENUMBER() 함수를 사용할 경우에 자주 나타납니다.)

```
Directed by Node Number
```

- 다음은 서브섹션이 주어진 노드 그룹 내의 사전 결정된 파티션 번호에 해당하는 노드로 전송됨을 나타냅니다. (명령문이 PATITION() 함수를 사용할 경우에 자주 나타납니다.)

```
Directed by Partition Number
|  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
```

- 다음은 서브섹션이 응용프로그램 커서에 대해 현재 행을 제공한 노드로 전송됨을 나타냅니다.

```
Directed by Position
```

- 다음은 명령문이 컴파일될 때 결정된 하나의 노드만이 서브섹션을 수신함을 나타냅니다.

```
Directed to Single Node
|  Node Number = n
```

- 다음은 서브섹션이 조정자 노드에서 실행됨을 나타냅니다.

```
Directed to Coordinator Node
```

- 다음은 서브섹션이 나열된 모든 노드로 전송됨을 나타냅니다.

```
Broadcast to Node List
|  Nodes = n1, n2, n3, ...
```

- 다음은 명령문이 실행될 때 결정된 하나의 노드만이 서브섹션을 수신함을 나타냅니다.

```
Directed to Any Node
```

- 테이블 대기행렬은 파티션된 데이터베이스 환경의 서브섹션 간에 또는 대칭 멀티프로세서(SMP) 환경의 서브에이전트간에 데이터를 이동시키는 데 사용됩니다. 테이블 대기행렬은 다음과 같이 설명됩니다.

– 다음 명령문은 데이터가 대기행렬로 삽입되고 있음을 나타냅니다.

```
Insert Into Synchronous Table Queue  ID = qn
Insert Into Asynchronous Table Queue  ID = qn
Insert Into Synchronous Local Table Queue  ID = qn
Insert Into Asynchronous Local Table Queue  ID = qn
```

– 데이터베이스 파티션 테이블 대기행렬의 경우, 테이블 대기행렬로 삽입된 행의 목적지는 다음 중 하나에 의해 설명됩니다.

```
Broadcast to Coordinator Node
```

모든 행이 조정자 노드로 전송됩니다.

```
Broadcast to All Nodes of Subsection n
```

주어진 서브섹션이 실행 중인 모든 데이터베이스 파티션으로 모든 행이 전송됩니다.

```
Hash to Specific Node
```

각 행은 행의 값에 따라 데이터베이스 파티션으로 전송됩니다.

```
Send to Specific Node
```

각 행은 명령문이 실행되는 동안 결정된 데이터베이스 파티션으로 전송됩니다.

```
Send to Random Node
```

각 행은 임의의 데이터베이스 파티션으로 전송됩니다.

– 데이터베이스 파티션 테이블 대기행렬은 임시 테이블에 임시적인 오버플로우 행을 가져야 하는 경우도 있습니다. 다음 명령문으로 가능성 여부가 식별됩니다.

```
Rows Can Overflow to Temporary Table
```

– 행을 테이블 대기행렬에 삽입하기 위한 푸시다운 조작을 포함하는 테이블 액세스 이후에는 즉시 전송할 수 없는 행을 처리하는 "완료" 명령문이 나옵니다. 다음 행 중 하나가 표시됩니다.

```
Insert Into Synchronous Table Queue Completion  ID = qn
Insert Into Asynchronous Table Queue Completion  ID = qn
Insert Into Synchronous Local Table Queue Completion  ID = qn
Insert Into Asynchronous Local Table Queue Completion  ID = qn
```

– 다음 명령문은 데이터가 테이블 대기행렬로부터 검색 중임을 나타냅니다.

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

메시지 다음에는 항상 검색 중인 컬럼 수 표시가 뒤따릅니다.

```
#Columns = n
```

– 테이블 대기행렬이 수신 끝에서 행을 정렬하는 경우, 테이블 대기행렬 액세스도 다음 메시지 중 하나를 수신하게 됩니다.

```
Output Sorted
Output Sorted and Unique
```

메시지 다음에는 항상 정렬 조작에 사용되는 키 수 표시가 뒤따릅니다.

```
#Key Columns = n
```

정렬 키의 각 컬럼의 경우, 다음 중 하나가 표시됩니다.

```
Key n: (Ascending)
Key n: (Descending)
```

– 술어가 테이블 대기행렬의 수신 끝에 의해 행으로 적용되는 경우, 다음 메시지가 표시됩니다.

```
Residual Predicate(s)
|  #Predicates = n
```

- 파티션된 데이터베이스 환경의 일부 서브섹션은 다음 명령문을 통해 서브섹션의 시작 부분으로 루프백됩니다.

```
Jump Back to Start of Subsection
```

연합 명령문 처리

연합 데이터베이스에서 SQL문을 실행할 때는 다른 데이터 소스에서 그 명령문의 일부를 수행할 수 있는 기능이 있어야 합니다.

다음은 데이터 소스에 액세스함을 나타냅니다.

```
Distributed Subquery #n
|  #Columns = n
```

분산 부속 조회에서 리턴되는 데이터에 술어를 적용할 수 있습니다. 적용되고 있는 술어의 수는 다음과 같이 지정됩니다.

```
Residual Predicate(s)
|  #Predicates = n
```

각 분산 부속 조회에 대한 세부사항이 별도로 제공됩니다. 분산 부속 조회에 대한 옵션이 아래에 설명되어 있습니다.

- 부속 조회에 대한 데이터 소스는 다음 중 하나로 표시됩니다.

  ```
  Server: server_name (type, version)
  Server: server_name (type)
  Server: server_name
  ```

- 부속 조회에 대한 SQL문은 다음과 같이 표시됩니다.

  ```
  Subquery SQL Statement:
  statement
  ```

- 부속 조회에서 참조되는 별칭은 다음과 같이 나열됩니다.

  ```
  Nickname Referenced:
  Schema.nickname Base = baseschema.basetable
  ```

- 부속 조회를 실행하기 전에 연합 서버에서 데이터 소스로 값이 전달될 경우, 값 갯수는 다음으로 표시됩니다.

  ```
  #Input Columns: n
  ```

- 부속 조회를 실행한 후에 데이터 소스에서 연합 서버로 값이 전달될 경우, 값 갯수는 다음으로 표시됩니다.

  ```
  #Output Columns: n
  ```

기타 명령문

- DDL문에 대한 섹션은 다음 명령문으로 출력에 표시됩니다.

  ```
  DDL Statement
  ```

  DDL문에 대해 추가 Explain 출력이 제공되지 않습니다.

- **CURRENT EXPLAIN SNAPSHOT**과 같은 갱신 가능 특수 레지스터에 대한 SET문의 섹션은 다음 명령문으로 출력에 표시됩니다.

  ```
  SET Statement
  ```

  SET문에 대해 추가 Explain 출력이 제공되지 않습니다.

- SQL문에 DISTINCT절이 포함되어 있으면 다음 텍스트가 출력에 나타납니다.

```
Distinct Filter  #Columns = n
```

여기서 n 구별 행 확보에 관계된 컬럼 수입니다. 구별 행 값을 검색하기 위해 행이 중복을 건너뛸 수 있도록 정렬되어야 합니다. 이 명령문은 다음 경우와 같이 데이터베이스 관리 프로그램이 중복을 명시적으로 제거할 필요가 없는 경우에는 나타나지 않습니다.

- 고유 색인이 존재하고 색인 키의 모든 컬럼이 DISTINCT 조작의 일부인 경우
- 정렬 중에 제거될 수 있는 중복인 경우

- 다음 명령문은 다음 연산이 특정 레코드 ID에 따라 달라질 수 있는 경우 나타납니다.

  ```
  Positioned Operation
  ```

  이 명령문은 WHERE CURRENT OF 구문을 사용하는 SQL문에 대해서 나타납니다.

- 다음 명령문은 결과에 적용되어야 하지만 다른 연산의 일부로는 적용될 수 없는 술어가 있는 경우 나타납니다.

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```

- 다음 명령문은 SQL문에 UNION 연산자가 있을 경우 나타납니다.

  ```
  UNION
  ```

- 다음 명령문은 후속 연산에서 사용될 행 값을 생성하려는 목적만을 가진 연산이 액세스 플랜에 있는 경우 나타납니다.

  ```
  Table Constructor
  |  n-Row(s)
  ```

  테이블 구성자(constructors)는 값을 후속 조작에 전달되는 행으로 변환시키는 데 사용될 수 있습니다. 테이블 구성자에 다음 행에 대한 프롬프트가 표시되면, 다음 명령문이 나타납니다.

  ```
  Access Table Constructor
  ```

- 다음 명령문은 특정 조건하에서만 처리되는 연산이 있을 경우 나타납니다.

```
   Conditional Evaluation
수   |  Condition #n:
   |  |  #Predicates = n
   |  Action #n:
```

조건 평가는 SQL CASE문과 같은 활동이나 참조 무결성 제한조건 또는 트리거와 같은 내부 메커니즘을 구현하는 데 사용됩니다. 아무런 조치도 나타나지 않으면, 조건이 참일 때 데이터 조작만이 처리됩니다.

- ALL, ANY 또는 EXISTS 부속 조회가 액세스 플랜에서 처리되고 있는 경우, 다음 명령문 중 하나가 나타납니다.
  - ANY/ALL Subquery
  - EXISTS Subquery
  - EXISTS SINGLE Subquery
- 특정 UPDATE 및 DELETE 조작에 앞서 테이블 내에 특정 행의 위치를 설정해야 합니다. 이것은 다음 명령문으로 표시됩니다.

  ```
  Establish Row Position
  ```

- 다음 명령문은 응용프로그램으로 리턴된 행이 있을 경우 나타납니다.

  ```
  Return Data to Application
  |  #Columns = n
  ```

  조작이 테이블 액세스로 푸시다운될 경우에는 완료 구문이 필요합니다. 이 구문은 다음과 같습니다.

  ```
  Return Data Completion
  ```

db2expln 및 dynexpln 출력의 예

db2expln 및 dynexpln으로부터 제공된 출력의 레이아웃과 형식을 이해하는 데 도움이 되는 5가지의 예가 다음에 나옵니다. 이 예들은 DB2에서 제공되는 SAMPLE 데이터베이스에 대해 수행됩니다. 각 예에 대해 간단한 설명이 나옵니다. 각 예간의 차이점은 **굵은체**로 표시됩니다.

예 1: 병렬 처리가 없는 플랜

이 예는 사원의 이름, 업무, 부서 이름 및 위치와 이들이 일하고 있는 프로젝트 이름의 목록을 요청하는 단순한 것입니다. 이 액세스 플랜의 핵심은 병합 조인이

지정된 각 테이블에서 관련 데이터와 조인하는 데 사용된다는 것입니다. 색인을 사용할 수 없기 때문에 액세스 플랜은 각 테이블에 대한 관계 스캔을 수행하며, 각 테이블은 조인하기 전에 정렬되어야 합니다.

```
******************** PACKAGE ****************************************
Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:47:58
Bind Timestamp = 2000-01-03-15.47.58.607455
Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
Partition Parallel       = No
Intra-Partition Parallel = No
Function Path            = "SYSIBM", "SYSFUN", "DOOLE"
-------------------- SECTION -----------------------------------------
Section = 1
SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno
Estimated Cost        = 126
Estimated Cardinality = 153
Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Insert Into Sorted Temp Table  ID = t1
|  |  #Columns = 3
|  |  #Sort Key Columns = 1
|  |  |  Key 1: DEPTNO (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 40
|  |  |  Row Width = 48
|  |  Piped
Sorted Temp Table Completion  ID = t1
Access Temp Table  ID = t1
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
   Merge Join
|  Access Table Name = DOOLE.PROJECT  ID = 2,7
```

```
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t2
|  |  |  #Columns = 2
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 38
|  |  |  |  Row Width = 28
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t2
|  Access Temp Table  ID = t2
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t3
|  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: WORKDEPT (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 63
|  |  |  |  Row Width = 32
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t3
|  Access Temp Table  ID = t3
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Return Data to Application
|  #Columns = 5
End of section
Optimizer Plan:
              RETURN
                 (   1)
                 |
              MSJOIN
```

```
                ( 2)
                /  \
           MSJOIN   TBSCAN
           ( 3)      ( 12)
           /  \        |
      TBSCAN  TBSCAN  SORT
      ( 4)    ( 8)    ( 13)
        |       |       |
      SORT    SORT    TBSCAN
      ( 5)    ( 9)    ( 14)
        |       |       |
      TBSCAN  TBSCAN  Table:
      ( 6)    ( 10)   DOOLE
        |       |     EMPLOYEE
    Table:  Table:
    DOOLE   DOOLE
    DEPARTMENT  PROJECT
```

플랜의 첫 번째 부분은 DEPARTMENT 및 PROJECT 테이블에 액세스하며 병합 조인을 사용하여 조인합니다. 이 조인의 결과는 EMPLOYEE 테이블에 조인됩니다. 결과 행은 응용프로그램으로 리턴됩니다.

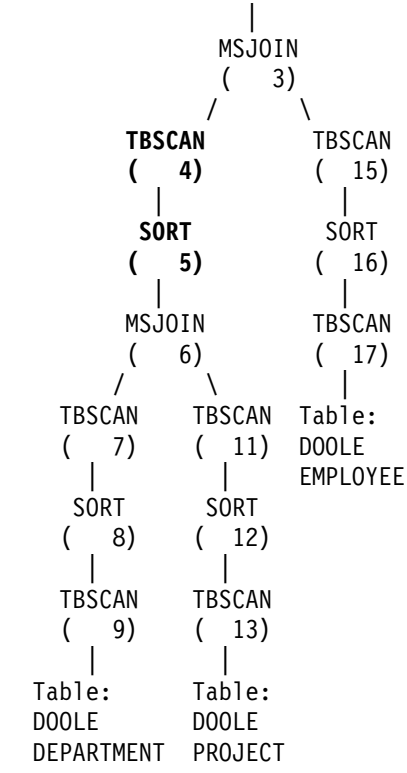
예 2: 파티션 내 병렬 처리를 사용한 단일 파티션된 데이터베이스 플랜

이 예에서는 676 페이지의 『예 1: 병렬 처리가 없는 플랜』과 동일한 SQL문을 보여주지만, 4 방향 SMP 머신에 의해 이 조회가 컴파일되어야 합니다.

```
******************** PACKAGE ****************************************
Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:48:51
Bind Timestamp = 2000-01-03-15.48.51.402403
Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
Partition Parallel       = No
Intra-Partition Parallel = Yes (Bind Degree = 4)
Function Path            = "SYSIBM", "SYSFUN", "DOOLE"
------------------- SECTION ----------------------------------------
Section = 1
SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno
```

```
Intra-Partition Parallelism Degree = 4
Estimated Cost        = 142
Estimated Cardinality = 153
Process Using 4 Subagents
|  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  |  #Columns = 3
|  |  Parallel Scan
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Shared Temp Table  ID = t1
|  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  Use Round-Robin Sort
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 40
|  |  |  |  Row Width = 48
|  |  |  Piped
|  Sorted Shared Temp Table Completion  ID = t1
|  Access Temp Table  ID = t1
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  Merge Join
|  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
|  |  |  #Columns = 2
|  |  |  Parallel Scan
|  |  |  Relation Scan
|  |  |  |  Prefetch: Eligible
|  |  |  Lock Intents
|  |  |  |  Table: Intent Share
|  |  |  |  Row  : Next Key Share
|  |  |  Insert Into Sorted Shared Temp Table  ID = t2
|  |  |  |  #Columns = 2
|  |  |  |  #Sort Key Columns = 1
|  |  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  |  Use Replicated Sort
|  |  |  |  Sortheap Allocation Parameters:
|  |  |  |  |  #Rows     = 38
|  |  |  |  |  Row Width = 28
|  |  |  |  Piped
|  |  Sorted Shared Temp Table Completion  ID = t2
|  |  Access Temp Table  ID = t2
|  |  |  #Columns = 2
|  |  |  Relation Scan
```

```
|  |  |  |  Prefetch: Eligible
|  Insert Into Sorted Shared Temp Table  ID = t3
|  |  #Columns = 5
|  |  #Sort Key Columns = 1
|  |  |  Key 1: (Ascending)
|  |  Use Partitioned Sort
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 61
|  |  |  Row Width = 72
|  |  Piped
|  Access Temp Table  ID = t3
|  |  #Columns = 5
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  Merge Join
|  |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|  |  |  #Columns = 3
|  |  |  Parallel Scan
|  |  |  Relation Scan
|  |  |  |  Prefetch: Eligible
|  |  |  Lock Intents
|  |  |  |  Table: Intent Share
|  |  |  |  Row  : Next Key Share
|  |  |  Insert Into Sorted Shared Temp Table  ID = t4
|  |  |  |  #Columns = 3
|  |  |  |  #Sort Key Columns = 1
|  |  |  |  |  Key 1: WORKDEPT (Ascending)
|  |  |  |  Use Partitioned Sort
|  |  |  |  Sortheap Allocation Parameters:
|  |  |  |  |  #Rows     = 63
|  |  |  |  |  Row Width = 32
|  |  |  |  Piped
|  |  Sorted Shared Temp Table Completion  ID = t4
|  |  Access Temp Table  ID = t4
|  |  |  #Columns = 3
|  |  |  Relation Scan
|  |  |  |  Prefetch: Eligible
|  Insert Into Asynchronous Local Table Queue  ID = q1
Access Local Table Queue  ID = q1  #Columns = 5
Return Data to Application
|  #Columns = 5
End of section
Optimizer Plan:
              RETURN
              (   1)
                |
               LTQ
              (   2)
```



이 플랜은 첫 번째 예의 플랜과 거의 동일합니다. 주요 차이점은 플랜이 처음으로 시작될 때 네 개의 서브에이전트가 작성되고, 플랜이 끝날 때 테이블 대기행렬이 각 서브에이전트의 작업 결과를 응용프로그램으로 리턴하기 전에 수집한다는 점입니다.

EMPLOYEE를 조인하기 전에 추가 정렬이 필요하다는 것도 알아두십시오. 이는 DEPARTMENT와 PROJECT간의 병합 조인을 처리하는 서브에이전트가 순서를 벗어난 조인된 행을 생성할 수 있기 때문에 필요합니다.

예 3: 파티션간 병렬 처리를 사용한 다중 파티션된 데이터베이스 플랜

이 예에서는 676 페이지의 『예 1: 병렬 처리가 없는 플랜』과 동일한 SQL문을 보여주지만, 이 조회는 세 개의 데이터베이스 파티션으로 이루어진 파티션된 데이터베이스에서 컴파일되었습니다.

```
******************** PACKAGE ****************************************
Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
```

```
Prep Time = 15:21:29
Bind Timestamp = 2000-01-03-15.21.29.990983
Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
Partition Parallel       = Yes
Intra-Partition Parallel = No
Function Path            = "SYSIBM", "SYSFUN", "DOOLE"
---------- SECTION ----------------------------------------
Section = 1
SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno
Estimated Cost        = 118
Estimated Cardinality = 263
Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5
Subsection #1:
   Access Table Queue  ID = q2  #Columns = 3
   |  Output Sorted
   |  |  #Key Columns = 1
   |  |  |  Key 1: (Ascending)
   Merge Join
   |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  |  Lock Intents
   |  |  |  Table: Intent Share
   |  |  |  Row  : Next Key Share
   |  |  Insert Into Sorted Temp Table  ID = t1
   |  |  |  #Columns = 3
   |  |  |  #Sort Key Columns = 1
   |  |  |  |  Key 1: DEPTNO (Ascending)
```

```
    |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 40
    |  |  |  |  Row Width = 48
    |  |  |  Piped
    |  Sorted Temp Table Completion  ID = t1
    |  Access Temp Table  ID = t1
    |  |  #Columns = 3
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
  Merge Join
    |  Access Table Queue  ID = q3  #Columns = 2
    |  |  Output Sorted
    |  |  |  #Key Columns = 1
    |  |  |  |  Key 1: (Ascending)
  Insert Into Asynchronous Table Queue  ID = q1
    |  Broadcast to Coordinator Node
    |  Rows Can Overflow to Temporary Table
Subsection #2:
  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
    |  #Columns = 3
    |  Relation Scan
    |  |  Prefetch: Eligible
    |  Lock Intents
    |  |  Table: Intent Share
    |  |  Row  : Next Key Share
    |  Insert Into Sorted Temp Table  ID = t2
    |  |  #Columns = 3
    |  |  #Sort Key Columns = 1
    |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  Sortheap Allocation Parameters:
    |  |  |  #Rows     = 27
    |  |  |  Row Width = 32
    |  |  Piped
  Sorted Temp Table Completion  ID = t2
  Access Temp Table  ID = t2
    |  #Columns = 3
    |  Relation Scan
    |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Table Queue  ID = q2
    |  |  Hash to Specific Node
    |  |  Rows Can Overflow to Temporary Tables
  Insert Into Asynchronous Table Queue Completion  ID = q2
Subsection #3:
  Access Table Name = DOOLE.PROJECT  ID = 2,7
    |  #Columns = 2
    |  Relation Scan
    |  |  Prefetch: Eligible
    |  Lock Intents
```

```
   |  |  Table: Intent Share
   |  |  Row  : Next Key Share
   |  Insert Into Sorted Temp Table  ID = t3
   |  |  #Columns = 2
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: DEPTNO (Ascending)
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 38
   |  |  |  Row Width = 28
   |  |  Piped
   Sorted Temp Table Completion  ID = t3
   Access Temp Table  ID = t3
   |  #Columns = 2
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Insert Into Asynchronous Table Queue  ID = q3
   |  |  Hash to Specific Node
   |  |  Rows Can Overflow to Temporary Tables
   Insert Into Asynchronous Table Queue Completion  ID = q3
End of section
Optimizer Plan:
              RETURN
                 (   1)
                  |
                 BTQ
                (   2)
                  |
                 MSJOIN
                 (   3)
                /      \
          MSJOIN          MDTQ
          (   4)         (  14)
         /      \          |
       MDTQ      TBSCAN    TBSCAN
      (   5)     (  10)    (  15)
        |          |         |
      TBSCAN      SORT      SORT
      (   6)     (  11)    (  16)
        |          |         |
       SORT      TBSCAN    TBSCAN
      (   7)     (  12)    (  17)
        |          |         |
      TBSCAN   Table:      Table:
      (   8)   DOOLE       DOOLE
        |      DEPARTMENT  PROJECT
     Table:
     DOOLE
     EMPLOYEE
```

이 플랜은 첫 번째 예의 플랜과 같지만, 섹션이 네 개의 서브섹션으로 나누어져 있습니다. 서브섹션은 다음 타스크를 포함합니다.

- **Coordinator Subsection**. 이 서브섹션은 다른 서브섹션을 조정합니다. 이 플랜에서 서브섹션은 다른 서브섹션이 분산되도록 하며, 테이블 대기행렬을 사용하여 응용프로그램으로 리턴되는 결과를 수집합니다.
- **Subsection #1**. 이 서브섹션은 테이블 대기행렬 q2를 스캔하고 병합 조인을 사용하여 이를 DEPARTMENT 테이블에 조인합니다. 두 번째 병합 조인은 테이블 대기행렬 q3의 데이터에 추가합니다. 조인된 행은 테이블 대기행렬 q1을 사용하여 조정자(coordinator) 서브섹션으로 전송됩니다.
- **Subsection #2**. 이 서브섹션은 EMPLOYEE 테이블을 스캔하고, 정렬하며, 결과를 특정 노드에 해쉬합니다. 이들 결과는 Subsection #1에 의해 읽혀집니다.
- **Subsection #3**. 이 서브섹션은 PROJECT 테이블을 스캔하고, 정렬하며, 결과를 특정 노드에 해쉬합니다. 이들 결과는 Subsection #1에 의해 읽혀집니다.

예 4: 파티션간 및 파티션 내 병렬 처리를 사용한 다중 파티션된 데이터베이스 플랜

이 예에서는 676 페이지의 『예 1: 병렬 처리가 없는 플랜』과 동일한 SQL문을 보여 주지만, 이 조회는 각각 4방향 SMP 머신에 있는 세 개의 데이터베이스 파티션으로 이루어진 파티션된 데이터베이스에서 컴파일되었습니다.

```
******************** PACKAGE ****************************************
Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:22:14
Bind Timestamp = 2000-01-03-15.22.14.659970
Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
Partition Parallel       = Yes
Intra-Partition Parallel = Yes (Bind Degree = 4)
Function Path            = "SYSIBM", "SYSFUN", "DOOLE"
-------------------- SECTION -----------------------------------------
Section = 1
SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
```

```
          = z.deptno
Intra-Partition Parallelism Degree = 4
Estimated Cost        = 140
Estimated Cardinality = 263
Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5
Subsection #1:
   Process Using 4 Subagents
   |  Access Table Queue  ID = q3  #Columns = 3
   |  Insert Into Sorted Shared Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: (Ascending)
   |  |  Use Partitioned Sort
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 27
   |  |  |  Row Width = 32
   |  |  Piped
   |  Access Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  Relation Scan
|  |  |  Prefetch: Eligible
   |  Merge Join
   |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
   |  |  |  #Columns = 3
   |  |  |  Parallel Scan
   |  |  |  Relation Scan
   |  |  |  |  Prefetch: Eligible
   |  |  |  Lock Intents
   |  |  |  |  Table: Intent Share
   |  |  |  |  Row  : Next Key Share
   |  |  |  Insert Into Sorted Shared Temp Table  ID = t2
   |  |  |  |  #Columns = 3
   |  |  |  |  #Sort Key Columns = 1
   |  |  |  |  |  Key 1: DEPTNO (Ascending)
   |  |  |  |  Use Partitioned Sort
   |  |  |  |  Sortheap Allocation Parameters:
```

```
        |  |  |  |  |  #Rows     = 40
        |  |  |  |  |  Row Width = 48
        |  |  |  |  Piped
        |  |  Sorted Shared Temp Table Completion  ID = t2
        |  |  Access Temp Table  ID = t2
        |  |  |  #Columns = 3
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  Insert Into Sorted Shared Temp Table  ID = t3
        |  |  #Columns = 6
        |  |  #Sort Key Columns = 1
        |  |  |  Key 1: (Ascending)
        |  |  Use Partitioned Sort
        |  |  Sortheap Allocation Parameters:
        |  |  |  #Rows     = 44
        |  |  |  Row Width = 76
        |  |  Piped
        |  Access Temp Table  ID = t3
        |  |  #Columns = 6
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  Merge Join
        |  |  Access Table Queue  ID = q5  #Columns = 2
        |  |  Insert Into Sorted Shared Temp Table  ID = t4
        |  |  |  #Columns = 2
        |  |  |  #Sort Key Columns = 1
        |  |  |  |  Key 1: (Ascending)
        |  |  |  Use Partitioned Sort
        |  |  |  Sortheap Allocation Parameters:
        |  |  |  |  #Rows     = 38
        |  |  |  |  Row Width = 28
        |  |  |  Piped
        |  |  Access Temp Table  ID = t4
        |  |  |  #Columns = 2
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  Insert Into Asynchronous Local Table Queue  ID = q2
     Access Local Table Queue  ID = q2  #Columns = 5
     Insert Into Asynchronous Table Queue  ID = q1
        |  Broadcast to Coordinator Node
        |  Rows Can Overflow to Temporary Table
  Subsection #2:
     Process Using 4 Subagents
        |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Parallel Scan
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
```

```
    |  |  Lock Intents
    |  |  |  Table: Intent Share
    |  |  |  Row  : Next Key Share
    |  |  Insert Into Sorted Shared Temp Table  ID = t5
    |  |  |  #Columns = 3
    |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  |  Use Round-Robin Sort
    |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 27
    |  |  |  |  Row Width = 32
    |  |  |  Piped
    |  Sorted Shared Temp Table Completion  ID = t5
    |  Access Temp Table  ID = t5
    |  |  #Columns = 3
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Local Table Queue  ID = q4
   Access Local Table Queue  ID = q4  #Columns = 3
   Insert Into Asynchronous Table Queue  ID = q3
    |  Hash to Specific Node
    |  Rows Can Overflow to Temporary Tables
Subsection #3:
   Process Using 4 Subagents
|  Access Table Name = DOOLE.PROJECT  ID = 2,7
    |  |  #Columns = 2
    |  |  Parallel Scan
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    |  |  Lock Intents
    |  |  |  Table: Intent Share
    |  |  |  Row  : Next Key Share
    |  |  Insert Into Sorted Shared Temp Table  ID = t6
    |  |  |  #Columns = 2
    |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: DEPTNO (Ascending)
    |  |  |  Use Round-Robin Sort
    |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 38
    |  |  |  |  Row Width = 28
    |  |  |  Piped
    |  Sorted Shared Temp Table Completion  ID = t6
    |  Access Temp Table  ID = t6
    |  |  #Columns = 2
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Local Table Queue  ID = q6
   Access Local Table Queue  ID = q6  #Columns = 2
```

```
   Insert Into Asynchronous Table Queue  ID = q5
   |  Hash to Specific Node
   |  Rows Can Overflow to Temporary Tables
End of section
Optimizer Plan:
                RETURN
                   (   1)
                    |
                   BTQ
                  (   2)
                    |
                   LTQ
                  (   3)
                    |
                  MSJOIN
                  (   4)
                 /      \
          TBSCAN          TBSCAN
         (   5)          (  20)
           |               |
          SORT            SORT
         (   6)          (  21)
           |               |
         MSJOIN           DTQ
         (   7)          (  22)
        /      \          |
   TBSCAN     TBSCAN     LTQ
   (   8)     (  16)     (  23)
     |          |         |
    SORT        SORT     TBSCAN
   (   9)     (  17)     (  24)
     |          |         |
    DTQ        TBSCAN     SORT
   (  10)     (  18)     (  25)
     |          |         |
    LTQ     Table:       TBSCAN
   (  11)   DOOLE        (  26)
     |      DEPARTMENT     |
   TBSCAN                Table:
   (  12)                DOOLE
     |                   PROJECT
    SORT
   (  13)
     |
   TBSCAN
   (  14)
```

```
   |
 Table:
 DOOLE
 EMPLOYEE
```

이 플랜은 다중 서브에이전트가 각 서브섹션을 실행하는 것을 제외하고, 682 페이지의 『예 3: 파티션간 병렬 처리를 사용한 다중 파티션된 데이터베이스 플랜』의 플랜과 유사합니다. 또한 각 서브섹션의 끝에서 지역 테이블 대기행렬은 규정 행이 특정 노드로 해쉬될 두 번째 테이블 대기행렬에 삽입되기 전에 모든 서브에이전트로부터 결과를 수집합니다.

예 5: 연합 데이터베이스 플랜

이 예에서는 676 페이지의 『예 1: 병렬 처리가 없는 플랜』과 동일한 SQL문을 보여 주지만, 이 조회는 DEPARTMENT 및 PROJECT 테이블이 데이터 소스에 있고 EMPLOYEE 테이블이 연합 서버에 있는 연합 데이터베이스에서 컴파일되었습니다.

```
******************** PACKAGE ****************************************
Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 16:29:01
Bind Timestamp = 2000-01-03-16.29.01.479230
Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
Partition Parallel       = No
Intra-Partition Parallel = No
Function Path            = "SYSIBM", "SYSFUN", "DOOLE"
------------------- SECTION ----------------------------------------
Section = 1
SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno
Estimated Cost        = 1954
Estimated Cardinality = 100800
Distribute Subquery #2
   |  #Columns = 3
   Insert Into Sorted Shared Temp Table  ID = t1
   |  #Columns = 3
   |  #Sort Key Columns = 1
```

```
|  |  Key 1: Remote Query #2, Output Column 1 (Ascending)
|  Sortheap Allocation Parameters:
|  |  #Rows     = 1000
|  |  Row Width = 56
|  PipedAccess Temp Table  ID = t1
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.DEPARTMENT  ID = 2,5
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t2   |  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: WORKDEPT (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 63
|  |  |  |  Row Width = 32
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t2
|  Access Temp Table  ID = t2
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Merge Join
|  Distribute Subquery #1
|  |  #Columns = 2
|  Insert Into Sorted Temp Table  ID = t3
|  |  #Columns = 2
|  |  |  Key 1: Remote Query #1, Output Column 1 (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 1000
|  |  |  Row Width = 36
|  |  Piped
|  Access Temp Table  ID = t3
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Return Data to Application
|  #Columns = 5

Distributed Subquery #1:
Server: REMOTE_SAMPLE  (DB2/CS 7.1)
Subquery SQL Statement:
```

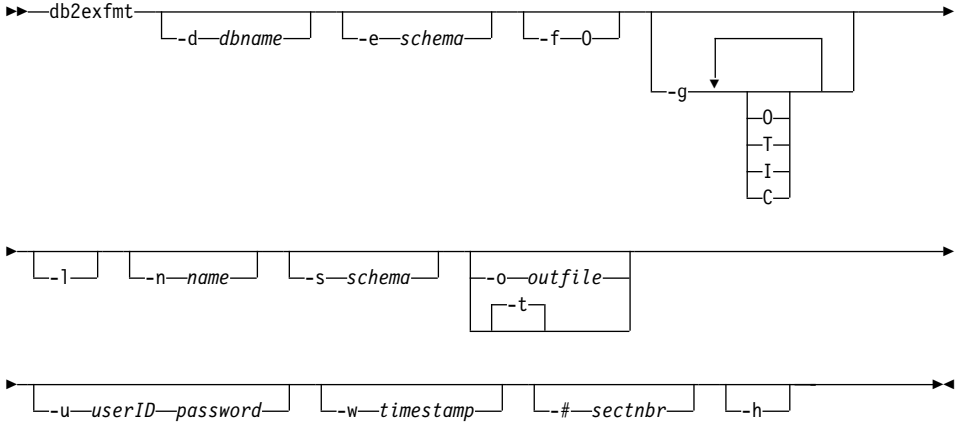
```
        SELECT A0."DEPTNO", A0."PROJNAME"
        FROM "DOOLE"."PROJECT" A0

   Nicknames Referenced:
      REMOTE.PROJECT  ID = 7  Base = DOOLE.PROJECT
   #Output Columns = 2

   Distributed Subquery #2:
   Server: REMOTE_SAMPLE  (DB2/CS 7.1)
   Subquery SQL Statement:

        SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
        FROM "DOOLE"."DEPARTMENT" A0

   Nicknames Referenced:
      REMOTE.DEPARTMENT  ID = 4  Base = DOOLE.DEPARTMENT
   #Output Columns = 3
End of section
Optimizer Plan:
                RETURN
                (   1)
                  |
                MSJOIN
                (   2)
               /      \
         MSJOIN        TBSCAN
         (   3)        (  13)
        /      \         |
    TBSCAN     TBSCAN    SORT
    (   4)     (   9)    (  14)
      |          |         |
     SORT       SORT     DSBQRY
    (   5)     (  10)    (  15)
      |          |         |
    DSBQRY     TBSCAN   Nickname:
    (   6)     (  11)   REMOTE
      |          |      PROJECT
  Nickname:    Table:
  REMOTE       DOOLE
  DEPARTMENT   EMPLOYEE
```

이 플랜은 테이블 중 두 테이블에 대한 데이터가 데이터 소스로부터 제공된다는 점은 제외하고, 첫 번째 예의 플랜과 같습니다. 두 개의 테이블에 분산 부속 조회를 통해 액세스하는데, 이 경우 간단하게 그 테이블에서 모든 행을 선택합니다. 데이터가 연합 서버로 리턴되면 그 데이터는 지역 테이블의 데이터에 조인됩니다.

부록D. db2exfmt - Explain 테이블 형식 도구

db2exfmt 도구를 사용하여 Explain 테이블 내용을 형식화할 수 있습니다. 이 도구는 인스턴스 sqllib 디렉토리의 misc 서브디렉토리에 있습니다.

이 도구는 사용하려면 형식화 중인 Explain 테이블에 대한 읽기 액세스 권한이 있어야 합니다.



-d dbname

패키지를 포함하는 데이터베이스의 이름

-e schema

Explain 테이블 스키마

-f 포맷팅 플래그. 이 릴리스에서 지원되는 유일한 값은 O입니다(연산자 요약).

-g 그래프 플랜. -g만 지정된 경우, 모든 테이블에 대한 형식화된 정보가 뒤에 따라오는 그래프가 생성됩니다. 그렇지 않으면 다음의 유효한 값이 조합되어 지정될 수 있습니다.

- **O** 그래프만 생성합니다. 테이블 내용을 형식화하지 않습니다.
- **T** 그래프의 각 연산자 아래에 전체 비용을 포함시킵니다.
- **I** 그래프의 각 연산자 아래에 I/O 비용을 포함시킵니다.
- **C** 그래프의 각 연산자의 예상 출력 기본 행 수(cardinality)(tuple의 수)를 포함시킵니다.

© Copyright IBM Corp. 1993, 2001

-l 패키지 이름을 처리할 때 대소문자를 구별합니다.

-n name

Explain 요청 소스의 이름(SOURCE_NAME)

-s schema

Explain 요청 소스의 스키마 또는 규정자(SOURCE_SCHEMA)

-o outfile

출력 파일 이름

-t 출력 경로를 터미널로 지정합니다.

-u user ID password

데이터베이스에 연결할 때 제공된 사용자 ID와 암호를 사용하십시오.

사용자 ID와 암호는 이름 지정 규칙에 맞게 유효해야 하며, 데이터베이스가 인식할 수 있어야 합니다.

-w timestamp

Explain 시간소인. 최신 Explain 요청을 확보하려면 -1을 지정하십시오.

-# sectnbr

소스의 섹션 번호. 모든 섹션을 요청하려면 0을 지정하십시오.

-h 도움말 정보를 표시합니다. 이 옵션이 지정되면 다른 옵션은 모두 무시되며 도움말 정보만 표시됩니다.

-h 및 -l 옵션의 경우를 제외하고, 제공되지 않거나 완전히 지정되지 않은 매개변수 값에 대해 프롬프트가 표시됩니다.

Explain 테이블 스키마가 제공되지 않을 경우, 환경 변수 값 **USER**가 기본값으로 사용됩니다. 이 변수를 찾을 수 없을 경우, Explain 테이블 스키마에 대해 프롬프트가 표시됩니다.

소스 이름, 소스 스키마 및 Explain 시간소인이 LIKE 술어 양식에 제공될 수 있으며, 이 양식에는 한 번의 호출로 여러 개의 소스를 선택하기 위해 퍼센트 부호(%) 및 밑줄(_)이 패턴 일치 문자로 사용될 수 있습니다. 최근에 Explain된 명령문의 경우, Explain 시간은 -1로 지정될 수 있습니다.

파일 이름 없이 -o가 지정되고 -t가 지정되지 않으면 파일 이름(기본 이름은 db2exfmt.out 임)에 대해 프롬프트가 표시됩니다. -o와 -t를 모두 지정하지 않아도 파일 이름(기본 옵션은 터미널 출력임)에 대해 프롬프트가 표시됩니다. -o와 -t가 모두 지정되면 출력 경로가 터미널로 지정됩니다.

부록E. DB2 라이브러리 사용

DB2 Universal Database 라이브러리는 온라인 도움말, 책(PDF 및 HTML) 및 샘플 프로그램이 HTML 형식으로 구성됩니다. 이 절에서는 제공되는 정보 및 액세스하는 방법에 대해 설명합니다.

제품 정보에 온라인으로 액세스하려면 정보 센터를 이용할 수 있습니다. 자세한 내용은 713 페이지의 『정보 센터를 사용하여 정보 액세스』를 참조하십시오. 웹에서 타스크 정보, DB2 책, 문제점 해결 정보, 샘플 프로그램 및 DB2 정보를 볼 수 있습니다.

DB2 PDF 파일 및 인쇄된 책

DB2 정보

다음의 표는 DB2 책을 네 개의 범주로 나눕니다.

DB2 안내 및 참조 정보

이들 책에는 모든 플랫폼에 대해 공통 DB2 정보가 수록되어 있습니다.

DB2 설치 및 구성 정보

이들 책은 특정 플랫폼에서의 DB2에 대한 것입니다. 예를 들어, OS/2, Windows 및 UNIX 기반 플랫폼에서의 DB2용으로 각각 다른 *빠른 시작* 책이 있습니다.

HTML 형식의 크로스 플랫폼 샘플 프로그램

이들 샘플은 응용프로그램 개발 클라이언트와 함께 설치된 샘플 프로그램의 HTML 버전입니다. 이들은 단지 정보용으로서 실제 프로그램을 바꾸지는 않습니다.

릴리스 정보

이러한 파일에는 DB2 책에 포함되지 않은 최신 정보가 포함되어 있습니다.

© Copyright IBM Corp. 1993, 2001

설치 매뉴얼, 릴리스 정보 및 자습서는 제품 CD-ROM의 HTML 디렉토리에서 볼 수 있습니다. 대부분의 책은 단지 보기용으로 제품 CD-ROM에서 HTML 형식으로 제공되고 보기와 인쇄용으로 DB2 책 CD-ROM에서 Adobe Acrobat(PDF) 형식으로 제공됩니다. 또한 IBM에서 인쇄된 책을 주문할 수 있습니다. 708 페이지의 『인쇄 책 주문』을 참조하십시오. 다음 표에는 주문할 수 있는 책을 보여줍니다.

OS/2 및 Windows 플랫폼에서는 `sqllib\doc\html` 디렉토리에 HTML 파일을 설치할 수 있습니다. DB2 정보는 여러 언어로 번역되었습니다. 그러나 모든 정보가 모든 언어로 번역된 것은 아닙니다. 정보를 특정 언어로 사용할 수 없을 경우에는 영문으로 제공됩니다.

UNIX 플랫폼에서는 `doc/%L/html` 디렉토리에 여러 나라 언어 버전의 HTML 파일을 설치할 수 있습니다. 여기서 *%L*은 해당 언어의 로케일을 나타냅니다. *빠른 시작* 책에서 보다 자세한 내용을 참조하십시오.

다음의 여러 가지 방법으로 DB2 책을 구하고 정보를 액세스할 수 있습니다.

- 712 페이지의 『온라인 정보 보기』
- 716 페이지의 『온라인 정보 검색』
- 708 페이지의 『인쇄 책 주문』
- 707 페이지의 『PDF 책 인쇄』

표 45. DB2 정보

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
	DB2 안내 및 참조 정보		
관리 안내서	*관리 안내서: 계획*에서는 데이터베이스의 개념에 대한 개요, 논리적 또는 물리적인 데이터베이스 설계와 같은 설계에 대한 정보 그리고 고가용성에 대한 정보를 제공합니다.	SA30-0990 db2d1x70	db2d0
	*관리 안내서: 구현*에서는 사용자의 설계, 데이터베이스 액세스, 감사, 백업 및 복구와 같은 구현에 대한 정보를 제공합니다.	SA30-0988 db2d2x70	
	*관리 안내서: 성능*에서는 데이터베이스의 환경, 응용프로그램 성능 평가 및 성능 조정에 대한 정보를 제공합니다.		
	사용자는 문서 번호 SBOF-8934를 사용하여 세 권으로 된 *관리 안내서* 책을 주문할 수 있습니다.	SA30-0989 db2d3x70	
Administrative API Reference	데이터베이스를 관리하는 데 사용할 수 있는 DB2 API와 데이터 구조에 대해 설명합니다. 또한 응용프로그램에서 API를 호출하는 방법을 설명합니다.	SC09-2947 db2b0x70	db2b0
응용프로그램 빌드 안내서	환경 설정 정보와 Windows, OS/2 및 UNIX 기반 플랫폼에서 DB2 응용프로그램을 컴파일하고, 링크하고, 수행하는 방법에 대한 단계별 지침을 제공합니다.	SA30-0991 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 Universal Database 제품을 사용할 때 만날 수 있는 APPC, CPI-C 및 SNA 감지 코드에 관한 일반 정보를 제공합니다. HTML 형식으로만 사용할 수 있습니다.	문서 번호 없음 db2apx70	db2ap

표 45. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
응용프로그램 개발 안내서	Embedded SQL 또는 Java(JDBC 및 SQLJ)를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램을 개발하는 방법에 대해 설명합니다. 저장 프로시듀어 작성, 사용자 정의 함수(UDF) 작성, 사용자 정의 유형 작성, 트리거 사용, 파티션된 환경 또는 연합 시스템에서 응용프로그램을 개발하는 등의 다양한 주제가 다루어집니다.	SA30-0992 db2a0x70	db2a0
CLI Guide and Reference	Microsoft ODBC 스펙과 호환 가능한 DB2 콜 레벨 인터페이스 및 호출 가능 SQL 인터페이스를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램을 개발하는 방법에 대해 설명합니다.	SC09-2950 db2l0x70	db2l0
Command Reference	명령행 처리기를 사용하는 방법 및 데이터베이스를 관리하기 위해 사용할 수 있는 DB2 명령에 대해 설명합니다.	SC09-2951 db2n0x70	db2n0
Connectivity Supplement	AS/400용 DB2, OS/390용 DB2, MVS용 DB2 또는 VM용 DB2를 DB2 Universal Database 서버와의 DRDA 응용프로그램 리퀘스터(AR)로 사용하는 방법에 대한 참조 정보 및 설치 정보를 제공합니다. 또한 DB2 Connect 응용프로그램 리퀘스터(AR)와 함께 DRDA 응용프로그램 서버를 사용하는 방법에 대해서도 상세히 설명합니다. HTML 및 PDF 형식으로만 사용할 수 있습니다.	문서 번호 없음 db2h1x70	db2h1
데이터 이동 유틸리티 안내 및 참조서	가져오기, 내보내기, 로드, 자동 로드 프로그램 및 DPROP와 같이 데이터 이동을 용이하게 해 주는 DB2 UDB 유틸리티의 사용 방법에 대해 설명합니다.	SA30-0994 db2dmx70	db2dm
Data Warehouse Center 관리 안내서	Data Warehouse Center를 사용하여 데이터 웨어하우스를 빌드 및 유지보수하는 방법에 대한 정보를 제공합니다.	SA30-1000 db2ddx70	db2dd
Data Warehouse Center 응용프로그램 통합 안내서	프로그래머들이 Data Warehouse Center 및 Information Catalog Manager를 응용프로그램과 통합하는 데 도움을 주는 정보를 제공합니다.	SA30-1001 db2adx70	db2ad

표 45. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
DB2 Connect 사용자 안내서	DB2 Connect 제품에 대한 개념, 프로그래밍 및 일반 사용 정보를 제공합니다.	SA30-0993 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	DB2 Query Patroller 시스템의 조작 개요, 특정 조작 및 관리 정보, 관리 그래픽 사용자 인터페이스 유틸리티에 대한 타스크 정보를 제공합니다.	SC09-2958 db2dwx70	db2dw
DB2 Query Patroller User's Guide	DB2 Query Patroller의 도구 및 함수를 사용하는 방법에 대해 설명합니다.	SC09-2960 db2wwx70	db2ww
용어집	DB2에서 사용되는 용어와 그 구성요소에 대한 정의를 제공합니다. HTML 형식과 *SQL 참조서*에서 사용할 수 있습니다.	문서 번호 없음 db2t0x70	db2t0
Image, Audio 및 Video Extenders 관리 및 프로그래밍	DB2 Extender에 대한 일반 정보와, 이미지, 오디오 및 비디오(IAV) Extenders 관리 및 구성에 대한 정보, 그리고 IAV Extenders를 사용한 프로그래밍에 대한 정보를 제공합니다. 여기에는 참조 정보, 진단 정보(메시지 포함) 및 샘플도 들어 있습니다.	SA30-1043 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	정보 카탈로그 관리에 대한 지침을 제공합니다.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Information Catalog Manager에 대한 아키텍처 인터페이스에 대한 정의를 제공합니다.	SC26-9997 db2bix70	db2bi
Information Catalog Manager 사용자 안내서	Information Catalog Manager 사용자 인터페이스 사용에 대한 정보를 제공합니다.	SA30-1002 db2aix70	db2ai
설치 및 구성 보충 설명서	플랫폼 특정 DB2 클라이언트의 플랜, 설치 및 설정에 대해 설명합니다. 또한 바인딩, 클라이언트 및 서버 통신의 설정, DB2 GUI 도구, DRDA AS, 분산 설치 및 분산 요청(DR)의 구성 및 이기종 데이터 소스에 액세스 등에 대한 정보가 들어 있습니다.	GA30-0975 db2iyx70	db2iy

표 45. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
메시지 참조서	DB2, Information Catalog Manager 및 Data Warehouse Center가 발행하는 메시지와 코드를 나열하고 수행해야 할 조치에 대해 설명합니다. 문서 번호(SBOF-8932)를 사용하여 두 권으로 된 메시지 참조서 책을 모두 주문할 수 있습니다.	볼륨 1 GA30-0986 db2m1x70 볼륨 2 GA30-0987 db2m2x70	db2m0
OLAP Integration Server Administration Guide	OLAP Integration Server의 관리 프로그램 구성요소를 사용하는 방법에 대해 설명합니다.	SC27-0782 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	표준 OLAP Metaoutline 인터페이스(Metaoutline Assistant가 아닌)를 사용하여 OLAP Metaoutlines를 작성하고 처리하는 방법에 대해 설명합니다.	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	표준 OLAP 모델 인터페이스(Model Assistant가 아닌)를 사용하여 OLAP 모델을 작성하는 방법에 대해 설명합니다.	SC27-0783 db2lpx70	n/a
OLAP 설치 및 사용자 안내서	OLAP Starter Kit에 대한 구성 및 설치 정보를 제공합니다.	SA30-1074 db2ipx70	db2ip
Excel용 OLAP Spreadsheet Add-in 사용자 안내서	Excel 스프레드시트 프로그램을 사용하여 OLAP 데이터를 분석하는 방법에 대해 설명합니다.	SA30-1094 db2epx70	db2ep
Lotus 1-2-3용 OLAP Spreadsheet Add-in 사용자 안내서	Lotus 1-2-3 스프레드시트 프로그램을 사용하여 OLAP 데이터를 분석하는 방법에 대해 설명합니다.	SA30-1093 db2tpx70	db2tp
복제 안내 및 참조서	DB2와 함께 제공되는 IBM 복제 도구에 대한 계획, 구성, 관리 및 사용법에 관한 정보를 제공합니다.	SA30-1003 db2e0x70	db2e0
Spatial Extender 사용자 안내 및 참조서	Spatial Extender 설치, 구성, 관리, 프로그래밍 및 문제점 해결에 대한 정보를 제공합니다. 또한 공간 데이터 개념에 대한 설명을 제공하고 Spatial Extender에 대한 특정 참조 정보(메시지 및 SQL)를 제공합니다.	SA30-1045 db2sbx70	db2sb

표 45. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
SQL 시작하기	SQL 개념을 소개하고 많은 구조와 타스크에 관한 예를 제공합니다.	SA30-0996 db2y0x70	db2y0
SQL 참조서, 볼륨 1 및 볼륨 2	SQL 구문, 의미 및 규칙에 대해 설명합니다. 또한 릴리스간 비호환성, 제품 제한사항 및 카탈로그 뷰에 대한 정보도 들어 있습니다. SBOF-8933 문서 번호를 사용하여 *SQL 참조서*를 주문할 수 있습니다.	볼륨 1 SA30-0997 db2s1x70 볼륨 2 SA30-0998 db2s2x70	db2s0
시스템 모니터 안내 및 참조서	데이터베이스와 데이터베이스 관리 프로그램에 대한 다른 종류의 정보를 수집하는 방법에 대해 설명합니다. 이 책은 데이터베이스 활동을 이해하고 성능을 향상시키며 문제점의 원인을 판별하기 위한 정보를 사용하는 방법에 대해 설명합니다.	SA30-0995 db2f0x70	db2f0
Text Extender 관리 및 프로그래밍	DB2 Extenders에 대한 일반 정보, Text Extenders 관리 및 구성에 관한 정보, Text Extenders를 사용한 프로그래밍에 대한 정보를 제공합니다. 여기에는 참조 정보, 진단 정보(메시지 포함) 및 샘플도 들어 있습니다.	SA30-1044 desu9x70	desu9
문제점 해결 안내서	오류의 소스를 판별하고 문제점으로부터 복구하며 DB2 고객 서비스와 상담하여 진단 도구를 사용하는 것을 도와줍니다.	GA30-0704 db2p0x70	db2p0
새로운 기능	DB2 Universal Database 버전 7의 새로운 특성, 기능 및 향상된 내용에 대해 설명합니다.	SA30-0999 db2q0x70	db2q0
	DB2 설치 및 구성 정보		
OS/2 및 Windows용 DB2 Connect Enterprise Edition 빠른 시작	OS/2 및 Windows 32비트 운영 체제에서 DB2 Connect Enterprise Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0974 db2c6x70	db2c6

표 45. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
UNIX용 DB2 Connect Enterprise Edition 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Connect Enterprise Edition에 대한 플랜, 이주, 설치, 구성 및 타스크 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0973 db2cyx70	db2cy
DB2 Connect Personal Edition 빠른 시작	OS/2 및 Windows 32비트 운영 체제에서 DB2 Connect Personal Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 모든 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0981 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	지원되는 모든 Linux 분산에서 DB2 Connect Personal Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GC09-2962 db2c4x70	db2c4
DB2 Data Links Manager 빠른 시작	AIX 및 Windows 32비트 운영 체제용 DB2 Data Links Manager에 대한 플랜, 설치, 구성 및 타스크 정보를 제공합니다.	GA30-0980 db2z6x70	db2z6
UNIX용 DB2 Enterprise - Extended Edition	UNIX 기반 플랫폼에서의 DB2 Enterprise - Extended Edition 플랜, 설치 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0978 db2v3x70	db2v3
Windows용 DB2 Enterprise - Extended Edition 빠른 시작	Windows 32비트 운영 체제용 DB2 Enterprise - Extended Edition에 대한 플랜, 설치 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0977 db2v6x70	db2v6
OS/2용 DB2 빠른 시작	OS/2 운영 체제에서의 DB2 Universal Database에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0982 db2i2x70	db2i2
UNIX용 DB2 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Universal Database에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0984 db2ixx70	db2ix

표 45. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
Windows용 DB2 빠른 시작	Windows 32비트 운영 체제에서의 DB2 Universal Database에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0985 db2i6x70	db2i6
DB2 Personal Edition 빠른 시작	OS/2 및 Windows 32비트 운영 체제에서의 DB2 Universal Database Personal Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GA30-0983 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	지원되는 모든 Linux 분산에서 DB2 Universal Database Personal Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller 설치 안내서	DB2 Query Patroller에 대한 설치 정보를 제공합니다.	GA30-0976 db2iwx70	db2iw
DB2 Warehouse Manager 설치 안내서	웨어하우스 에이전트, 웨어하우스 변환기 및 Information Catalog Manager에 대한 설치 정보를 제공합니다.	GA30-1027 db2idx70	db2id
	HTML 형식의 크로스 플랫폼 샘플 프로그램		
HTML 형식의 샘플 프로그램	DB2가 지원하는 모든 플랫폼에서 프로그래밍 언어에 대한 샘플 프로그램이 HTML 형식으로 제공됩니다. 이 샘플 프로그램은 정보용으로만 제공됩니다. 모든 샘플을 모든 프로그래밍 언어로 사용할 수 있는 것은 아닙니다. HTML 샘플은 DB2 응용프로그램 개발 클라이언트가 설치될 때만 사용할 수 있습니다. 프로그램에 대한 자세한 내용은 *응용프로그램 빌드 안내서*를 참조하십시오.	문서 번호 없음	db2hs
	릴리스 정보		
DB2 Connect 릴리스 정보	DB2 Connect 책에는 포함될 수 없었던 최신 정보를 제공합니다.	#2를 참조하십시오.	db2cr
DB2 설치 정보	DB2 책에는 포함될 수 없었던 최신 설치 특정 정보를 제공합니다.	제품 CD-ROM에서만 사용할 수 있습니다.	

표 45. DB2 정보 (계속)

이름	설명	문서 번호 PDF 파일 이름	HTML 디렉토리
DB2 릴리스 정보	DB2 책에는 포함될 수 없었던 모든 DB2 제품 및 기능에 대한 최신 정보를 제공합니다.	#2를 참조하십시오. db2ir	

주:

1. 파일 이름의 6번째 자리에 있는 문자 *x*는 책의 언어 버전을 나타냅니다. 예를 들어, 파일 이름 db2d0e70은 *관리 안내서* 책의 영문 버전을 나타내며 db2d0k70은 같은 책의 한글 버전을 나타냅니다. 다음 문자는 파일 이름의 6 번째 자리에 사용되어 언어 버전을 나타냅니다.

언어	식별자
브라질 포르투갈어	b
불가리아어	u
체코어	x
덴마크어	d
네덜란드어	q
영어	e
핀란드어	y
프랑스어	f
독일어	g
그리스어	a
헝가리어	h
이탈리아어	i
일본어	j
한국어	k
노르웨이어	n
폴란드어	p
포루투갈어	v
러시아어	r
중국어	c
슬로베니아어	l
스페인어	z
스웨덴어	s
대만어	t
터키어	m

2. DB2 책에 포함되어 있지 않을 수 있는 최신 정보는 릴리스 정보에서 HTML 형식과 ASCII 파일로 사용할 수 있습니다. HTML 버전은 정보 센터와 제품 CD-ROM에서 사용할 수 있습니다. ASCII 파일을 보려면,

- UNIX 기반 프랫폼에서, Release.Notes 파일을 참조하십시오. 이 파일은 DB2DIR/Readme/%L 디렉토리에 있으며, 여기서 %L은 로케일 이름을 나타내고 DB2DIR은 다음과 같습니다.
  - AIX에서는 /usr/lpp/db2_07_01
  - HP-UX, PTX, Solaris 및 Silicon Graphics IRIX에서는 /opt/IBMdb2/V7.1
  - Linux에서는 /usr/IBMdb2/V7.1
- 기타 플랫폼에서, RELEASE.TXT 파일을 참조하십시오. 이 파일은 제품이 설치된 디렉토리에 있습니다. OS/2 플랫폼에서는 **IBM DB2** 폴더를 더블 클릭한 다음 **릴리스 정보** 아이콘을 더블 클릭할 수 있습니다.

PDF 책 인쇄

인쇄된 책의 사본을 원하는 경우, DB2 책 CD-ROM에 있는 PDF 파일을 인쇄할 수 있습니다. Adobe Acrobat Reader를 사용하여 책 전체나 특정 페이지를 인쇄할 수 있습니다. 라이브러리에 있는 각 책의 파일 이름에 대한 자세한 내용은 699 페이지의 표45을 참조하십시오.

Adobe 웹 사이트 http://www.adobe.com에서 Adobe Acrobat Reader의 최신 버전을 얻을 수 있습니다.

PDF 파일은 파일 확장자가 PDF인 DB2 책 CD-ROM에 들어 있습니다. PDF 파일에 액세스하려면 다음을 수행하십시오.

1. DB2 책 CD-ROM을 삽입하십시오. UNIX 기반의 플랫폼에서는 DB2 책 CD-ROM을 마운트합니다. 마운트 프로시듀어에 대한 자세한 내용은 *빠른 시작* 책을 참조하십시오.
2. Acrobat Reader를 시작하십시오.
3. 다음 위치 중 하나에서 원하는 PDF 파일을 여십시오.
   - OS/2 및 Windows 플랫폼에서

     x:\doc*language* 디렉토리. 여기서 *x*는 CD-ROM 드라이브를 나타내며 *language*는 사용자 언어를 나타내는 2문자 국가 코드를 나타냅니다. 예를 들어, 영문인 경우에는 EN입니다.
   - UNIX 기반 플랫폼에서

/cdrom/doc/*%L* 디렉토리. 여기서 */cdrom*은 CD-ROM의 마운트 지점을 나타내고 *%L*은 원하는 로케일의 이름을 나타냅니다.

또한 PDF 파일을 CD-ROM에서 지역이나 네트워크 드라이브로 파일을 복사하고 거기서 읽을 수도 있습니다.

인쇄 책 주문

인쇄된 DB2 책은 책 주문 번호(SBOF)를 사용하여 세트나 낱권으로 주문할 수 있습니다. 인쇄본을 주문하려면 IBM 협력업체 또는 영업 대표에게 문의하십시오. 또한 웹 페이지 http://www.elink.ibmlink.ibm.com/pbl/pbl에서도 책을 주문할 수 있습니다.

두 종류의 책 세트를 사용할 수 있습니다. SBOF-8935는 DB2 Warehouse Manager에 대한 참조 및 사용에 관한 정보를 제공합니다. SBOF-8931은 다른 모든 DB2 Universal Database 제품과 기능에 대한 참조 및 사용 정보를 제공합니다. 각 SBOF의 내용은 다음 표에 나열되어 있습니다.

표 46. 인쇄된 책 주문

SBOF 번호	**포함된 책**	
SBOF-8931	• 관리 안내서: 계획 • 관리 안내서: 구현 • 관리 안내서: 성능 • Administrative API Reference • 응용프로그램 빌드 안내서 • 응용프로그램 개발 안내서 • CLI Guide and Reference • Command Reference • 데이터 이동 유틸리티 안내 및 참조서 • Data Warehouse Center 관리 안내서 • Data Warehouse Center 응용프로그램 통합 안내서 • DB2 Connect 사용자 안내서 • 설치 및 구성 보충 설명서 • Image, Audio, and Video Extenders 관리 및 프로그래밍 • 메시지 참조서, 볼륨 1 및 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP 설치 및 사용자 안내서 • Excel용 OLAP Spreadsheet Add-in 사용자 안내서 • Lotus 1-2-3용 OLAP Spreadsheet Add-in 사용자 안내서 • 복제 안내 및 참조서 • Spatial Extender Administration and Programming Guide • SQL 시작하기 • SQL 참조서, 볼륨 1 및 2 • 시스템 모니터 안내 및 참조서 • Text Extender 관리 및 프로그래밍 • 문제점 해결 안내서 • 새로운 기능
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager 사용자 안내서 • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2 온라인 문서

온라인 도움말 액세스

온라인 도움말은 모든 DB2 구성요소에서 사용 가능합니다. 다음의 표에서는 다양한 도움말 유형에 대해 설명합니다.

도움말 유형	내용	액세스하는 방법
명령 도움말	명령행 처리기의 명령 구문에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오. `? command` 여기서 *command*는 키워드이거나 전체 명령입니다. 예를 들어, `? catalog`는 모든 CATALOG 명령에 대한 도움말을 표시하고, `? catalog database`는 CATALOG DATABASE 명령에 대한 도움말을 표시합니다.
클라이언트 구성 지원 프로그램 도움말 *명령 센터 도움말* *제어 센터 도움말* *Data Warehouse Center 도움말* *이벤트 분석기 도움말* *Information Catalog Manager 도움말* *위성 관리 센터 도움말* *스크립트 센터 도움말*	창 또는 노트북에서 수행할 수 있는 타스크에 대해 설명합니다. 도움말은 알아야 할 개요와 전제조건 정보를 포함하고, 창 또는 노트북 제어를 사용하는 방법에 대해 설명합니다.	창이나 노트북에서 **도움말** 버튼을 누르거나 **F1** 키를 누르십시오.

도움말 유형	내용	액세스하는 방법

메시지 도움말	메시지의 원인과 사용자가 취해야 할 조치에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오.
		`? XXXnnnnn`
		여기서 *XXXnnnnn*은 유효한 메시지 식별자입니다.
		예를 들어, `? SQL30081`은 SQL30081 메시지에 대한 도움말을 표시합니다.
		한 번에 한 화면씩 메시지 도움말을 보려면 다음을 입력하십시오.
		`? XXXnnnnn | more`
		파일에 메시지 도움말을 저장하려면, 다음을 입력하십시오.
		`? XXXnnnnn > filename.ext`
		여기서 *filename.ext*는 메시지 도움말을 저장하려는 파일입니다.

SQL 도움말	SQL문의 구문에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오. `help statement` 여기서 *statement*는 SQL문입니다. 예를 들어, `help SELECT`는 SELECT문에 대한 도움말을 표시합니다. **주:** SQL 도움말은 UNIX 기반 플랫폼에서는 사용 가능하지 않습니다.
SQLSTATE 도움말	SQL 상태와 클래스 코드에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오. `? sqlstate` 또는 `? class code` 여기서 *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자입니다. 예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

온라인 정보 보기

이 제품에 들어 있는 책은 HTML(Hypertext Markup Language) 소프트카피 형식으로 제공됩니다. 소프트카피 형식은 정보를 검색할 수 있게 하고 관련된 정보로 링크하는 하이퍼텍스트를 제공합니다. 또한 사이트에서 라이브러리를 공유하는 것도 더 쉬워집니다.

HTML 버전 3.2 스펙을 따르는 브라우저로 온라인 책 또는 샘플 프로그램을 볼 수 있습니다.

온라인 책 또는 샘플 프로그램을 보려면 다음을 수행하십시오.

- DB2 관리 도구를 수행중이면, 정보 센터를 사용하십시오.
- 브라우저에서 **파일 —> 페이지 열기**를 누르십시오. 사용자가 여는 페이지에 DB2 정보에 대한 설명과 링크가 있습니다.
  - UNIX 기반 플랫폼의 경우, 다음 페이지를 여십시오.

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    여기서 *%L*은 로케일 이름입니다.
  - 기타 플랫폼에서는, 다음 페이지를 여십시오.

    sqllib\doc\html\index.htm

    이 경로는 DB2가 설치된 드라이브에 있습니다.

    정보 센터를 설치하지 않은 경우, **DB2 정보** 아이콘을 더블 클릭하여 페이지를 열 수 있습니다. 사용하는 시스템에 따라 주 제품 폴더나 Windows 시작 메뉴에 아이콘이 있습니다.

Netscape 브라우저 설치

웹 브라우저를 설치하지 않은 경우, 제품 상자에 있는 Netscape CD-ROM에서 Netscape를 설치할 수 있습니다. 설치하는 방법에 대한 자세한 지침을 보려면 다음을 수행하십시오.

1. Netscape CD-ROM을 삽입하십시오.
2. UNIX 기반 플랫폼에서는 CD-ROM을 마운트하십시오. 마운트 프로시듀어에 대한 자세한 내용은 *빠른 시작* 책을 참조하십시오.

3. 설치 지침의 경우에는 CDNAV*nn*.txt 파일을 참조하십시오. 여기서 *nn*은 2문자로 된 언어 식별자입니다. 파일은 CD-ROM의 루트 디렉토리에 있습니다.

정보 센터를 사용하여 정보 액세스

정보 센터는 DB2 제품 정보에 대한 빠른 액세스를 제공합니다. 정보 센터는 DB2 관리 도구를 사용할 수 있는 모든 플랫폼에서 사용 가능합니다.

정보 센터 아이콘을 더블 클릭하여 정보 센터를 열 수 있습니다. 사용하는 시스템에 따라 아이콘은 주 제품 폴더나 Windows **시작** 메뉴의 정보 폴더에 있습니다.

또한 DB2 Windows 플랫폼에서 도구 모음이나 **도움말** 메뉴를 사용하여 정보 센터를 액세스할 수 있습니다.

정보 센터는 6개 유형의 정보를 제공합니다. 적절한 탭을 눌러 해당 유형에 제공되는 주제를 보십시오.

타스크 DB2를 사용하여 수행할 수 있는 주요 타스크.

참조 키워드, 명령 및 API와 같은 DB2 참조 정보.

책 DB2 책.

문제점 해결 오류 메시지의 범주와 복구 조치.

샘플 프로그램 DB2 응용프로그램 개발 클라이언트와 함께 제공되는 샘플 프로그램. DB2 응용프로그램 개발 클라이언트를 설치하지 않은 경우, 이 탭은 표시되지 않습니다.

웹 월드 와이드 웹에서의 DB2 정보. 이 정보에 액세스하려면 시스템에서 웹으로의 연결을 갖고 있어야 합니다.

목록 중 하나에서 항목을 선택하면 정보 센터가 표시기를 시작하여 정보를 표시합니다. 표시기는 사용자가 선택하는 정보의 종류에 따라 시스템 도움말 표시기, 편집기 또는 웹브라우저가 될 수 있습니다.

정보 센터는 찾기 기능을 제공하므로 목록을 찾지 않고도 특정 주제를 찾을 수 있습니다.

전체 텍스트 검색의 경우, **DB2 온라인 정보 검색** 검색 양식으로 연결된 정보 센터의 하이퍼텍스트 링크를 따라 검색하십시오.

HTML 검색 서버는 보통 자동으로 시작됩니다. HTML 정보에서 검색 기능이 작동하지 않으면 다음 방법 중 하나를 사용하여 검색 서버를 시작할 수 있습니다.

Windows의 경우:

시작을 누르고 **프로그램 —> IBM DB2 —> 정보 —> HTML 검색 서버 시작**을 선택하십시오.

OS/2의 경우

OS/2용 DB2 폴더를 더블 클릭한 다음 **HTML 검색 서버 시작** 아이콘을 더블 클릭하십시오.

HTML 정보 검색시 그 외의 다른 문제가 발생한 경우에는 릴리스 정보를 참조하십시오.

주: 검색 기능은 Linux, PTX 및 Silicon Graphics IRIX 환경에서는 사용할 수 없습니다.

DB2 마법사 사용

마법사는 한 번에 한 단계씩 각 타스크를 수행하게 함으로써 특정 관리 타스크를 완료하는 데 도움을 줍니다. 마법사는 제어 센터 및 클라이언트 구성 지원 프로그램을 통해 사용할 수 있습니다. 다음 표에서는 마법사를 나열하고 해당 기능에 대해 설명합니다.

주: 데이터베이스 작성, 색인 작성, 다중 사이트 갱신 구성 및 성능 구성 마법사는 파티션된 데이터베이스 환경에서 사용할 수 있습니다.

마법사	도움 내용	액세스하는 방법
데이터베이스 추가	클라이언트 워크스테이션의 데이터베이스를 카탈로그화합니다.	클라이언트 구성 지원 프로그램에서 **추가**를 누르십시오.
데이터베이스 백업	백업 플랜의 결정, 작성 및 스케줄합니다.	제어 센터에서 백업하려는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 **백업 —> 마법사를 사용한 데이터베이스**를 선택하십시오.
다중 사이트 갱신 구성	다중 사이트 갱신, 분산 트랜잭션 또는 2 단계 확약을 구성합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스 오른쪽 단추로 누른 다음 **다중 사이트 갱신**을 선택하십시오.

마법사	도움 내용	액세스하는 방법
데이터베이스 작성	데이터베이스 작성 및 일부 기본 구성 타스크를 수행합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 데이터베이스**를 선택하십시오.
테이블 작성	기본 데이터 유형의 선택 및 테이블의 기본 키를 작성합니다.	제어 센터에서 **테이블** 아이콘을 마우스 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 테이블**을 선택하십시오.
테이블 공간 작성	새로운 테이블 공간을 작성합니다.	제어 센터에서 **테이블 공간** 아이콘을 마우스 오른쪽 단추로 선택한 다음 **작성 —> 마법사를 사용한 테이블 공간**을 선택하십시오.
색인 작성	사용자의 모든 조회를 작성하고 삭제하기 위해 색인화합니다.	제어 센터에서 **색인** 아이콘을 마우스 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 색인**을 선택하십시오.
성능 구성	비지니스 요구에 맞게 구성 매개변수를 갱신하여 데이터베이스의 성능을 조정합니다.	제어 센터에서 조정하려는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 **마법사를 사용한 성능 구성**을 선택하십시오. 파티션된 데이터베이스 환경에 대해 데이터베이스 파티션 뷰에서 조정하려는 첫 번째 데이터베이스 파티션을 마우스 오른쪽 단추로 누른 다음 **마법사를 사용한 성능 구성**을 선택하십시오.
데이터베이스 복원	실패 후에 데이터베이스를 복구합니다. 사용할 백업 종류 및 재작동할 로그를 이해하는 데 도움을 줍니다.	제어 센터에서 복원하려는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 **복원 —> 마법사를 사용한 데이터베이스**를 선택하십시오.

문서 서버 설정

기본값으로 DB2 정보는 지역 시스템에 설치됩니다. 이것은 DB2 정보에 대한 액세스를 필요로 하는 각 사용자가 같은 파일을 설치해야 함을 의미합니다. DB2 정보를 단일 위치에 저장하려면 다음 단계를 수행하십시오.

1. 모든 파일과 서브디렉토리를 지역 시스템의 \sqllib\doc\html에서 웹 서버로 복사하십시오. 각 책은 책을 구성하는 데 필요한 모든 HTML 및 GIF 파일이 들어 있는 서브디렉토리를 가집니다. 디렉토리 구조가 동일하게 유지되도록 해야 합니다.
2. 새 위치에서 파일을 찾을 수 있도록 웹 서버를 구성하십시오. 자세한 내용은 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.
3. Java 버전의 정보 센터를 사용 중인 경우, 모든 HTML 파일에 대한 기본 URL을 지정할 수 있습니다. 책 목록에 대한 URL을 사용해야 합니다.
4. 책 파일을 볼 수 있게 되면 다음과 같이 자주 보는 주제 항목에 대해서는 책갈피를 설정할 수 있습니다. 다음의 페이지들을 북마크로 설정해 두면 도움이 될 것입니다.
   - 책 목록
   - 자주 사용되는 책의 목차
   - ALTER TABLE 주제와 같은 자주 참조하는 항목
   - 검색 형식

DB2 Universal Database 온라인 문서 파일을 중앙 시스템에서 제공하는 방법에 대한 정보를 보려면 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.

온라인 정보 검색

HTML 파일에서 정보를 찾으려면 다음 방법 중 하나를 사용하십시오.

- 맨 위 프레임에서 **검색**을 누르십시오. 특정 주제를 찾으려면 검색 양식을 사용하십시오. 이 기능은 Linux, PTX 또는 Silicon Graphics IRIX 환경에서는 사용할 수 없습니다.
- 맨 위 프레임에서 **색인**을 누르십시오. 책에서 특정 주제를 찾으려면 색인을 사용하십시오.
- 책에서 특정 주제를 찾으려면 목차나 도움말의 색인 또는 HTML 책을 표시하고 웹 브라우저의 찾기 기능을 사용하십시오.
- 특정 주제로 빨리 리턴하려면 웹 브라우저의 책갈피 기능을 사용하십시오.

- 특정 주제를 찾으려면 정보 센터의 검색 기능을 사용하십시오. 자세한 내용은 713 페이지의 『정보 센터를 사용하여 정보 액세스』를 참조하십시오.

부록F. 주의사항

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 그러한 특허에 대한 사용권까지를 부여하는 것은 아닙니다. 사용권에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2 바이트(DBCS) 정보에 관한 사용권 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이

© Copyright IBM Corp. 1993, 2001

책을『현상태대로』제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통고 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트의 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

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

(i) 독립적으로 작성된 프로그램 및 기타 프로그램(이 프로그램 포함)간의 정보 교환 (ii) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 사용권자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건(예를 들면, 사용권 지불 포함)에 따라 사용할 수 있습니다.

이 정보에 기술된 사용권 프로그램 및 사용가능한 모든 사용권 자료는 IBM이 IBM 기본 계약, IBM 프로그램 사용권 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

여기에 있는 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, 이러한 측정치가 일반적으로 사용가능한 시스템

에서도 동일하다고는 보장하지 않습니다. 더우기, 일부 측정치는 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 본인의 특정 환경에 적용할 수 있는 데이터를 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 배상 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지없이 변경될 수 있습니다.

이 정보에는 일상의 업무에서 사용되는 자료와 보고의 예제가 포함되어 있을 수 있습니다. 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 예제에 들어 있습니다. 이들 이름은 모두 가공의 것이며, 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권:

이 정보에는 여러 가지 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 포함되어 있을 수 있습니다. 샘플 응용프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스에 부합하는 응용프로그램을 개발, 사용, 마케팅 또는 배포를 목적으로 이들 샘플 프로그램을 복사, 수정 및 배포할 수 있으며 IBM에 대한 지불 의무는 없습니다. 이들 예제 프로그램은 모든 조건에서 철저하게 검사된 것은 아닙니다. 따라서, IBM은 이들 프로그램의 신뢰성, 서비스 가능성 또는 기능에 대해 어떠한 보증도 하지 않습니다.

각 사본이나 이들 샘플 프로그램의 일부 또는 파생본에는 다음과 같은 저작권 주의사항을 포함시켜야 합니다.

© (귀하의 회사명) (연도). 이 코드 부분은 IBM 샘플 프로그램에 나와 있습니다. © Copyright IBM Corp. _연도 입력_. All rights reserved.

상표

별표(*)로 표시된 다음의 용어는 전세계에서 IBM의 상표입니다.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 ConnectDB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational
Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

다음 용어는 기타 회사의 상표 또는 등록상표입니다.

Microsoft, Windows 및 Windows NT는 Microsoft Corporation의 상표 또는 등록상표입니다.

Java 또는 모든 Java 관련 상표와 로고 및 Solaris는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.

Tivoli 및 NetView는 미국 또는 기타 국가에서 사용되는 Tivoli Systems Inc.의 상표입니다.

UNIX는 미국 또는 기타 국가에서 X/Open Company Limited가 독점적인 사용권을 가진 등록상표입니다.

두 개의 별표(**)가 붙은 기타 회사 이름, 제품 이름 또는 서비스 이름은 해당 회사의 상표이거나 서비스표입니다.

색인

[가]

© Copyright IBM Corp. 1993, 2001

[나]

[다]

[라]

[마]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

C

D

E

T

U

V

W

X

Z

IBM에 문의

기술적인 문제가 발생한 경우에는 DB2 고객만족센터에 문의하기 전에 *문제점 해결 안내서*에서 제안한 조치를 검토하고 실행해 보십시오. 이것은 DB2 고객 지원 부서로 하여금 사용자를 보다 더 잘 지원할 수 있도록 사용자가 모을 수 있는 정보를 제공합니다.

DB2 Universal Database 제품에 대한 정보나 주문은 지방 사무소의 IBM 담당자나 공인 IBM 소프트웨어 재판매업자에게 문의하십시오.

미국에 사시는 분은 다음 번호 중 하나를 선택하여 전화하십시오.

- 고객 지원을 받으려면, 1-800-237-5511.
- 사용 가능한 서비스 옵션을 알려면, 1-888-426-4343.

제품 정보

미국에 사시는 분은 다음 번호 중 하나를 선택하여 전화하십시오.

- 제품 주문이나 일반 정보를 얻으려면 1-800-IBM-CALL(1-800-426-2255) 또는 1-800-3IBM-OS2(1-800-342-6672).
- 책에 대한 주문은, 1-800-879-2755.

http://www.ibm.com/software/data/

DB2 WWW(World Wide Web) 페이지에서는 새 소식, 제품 설명, 교육 일정 등에 관한 현재 DB2 정보를 제공합니다.

http://www.ibm.com/software/data/db2/library/

DB2 제품 및 서비스 기술 라이브러리는 빈도 높은 질문(FAQ), 수정사항(fixes), 책 및 최신 DB2 기술 정보에 대한 액세스를 제공합니다.

주: 이러한 정보는 영어로만 제공됩니다.

http://www.elink.ibmlink.ibm.com/pbl/pbl/

여기에서는 책을 웹 사이트에서 주문할 수 있는 방법을 제공합니다.

© Copyright IBM Corp. 1993, 2001

http://www.ibm.com/education/certify/

IBM 웹 사이트에서 기술 전문 인증 프로그램은 DB2를 포함하여 다른 IBM 제품의 기술 전문 인증 테스트 정보를 제공합니다.

ftp.software.ibm.com

anonymous로 로그온하십시오. /ps/products/db2 디렉토리에서, DB2와 많은 관련 제품에 관한 데이터, 수정사항, 도구 등을 찾을 수 있습니다.

comp.databases.ibm-db2, bit.listserv.db2-l

이러한 인터넷 뉴스 그룹으로 사용자는 DB2 제품에 대한 자신의 사용 경험을 토론할 수 있습니다.

Compuserve에서, GO IBMDB2

이 명령을 입력하여 IBM DB2 계열 포럼을 액세스하십시오. 모든 DB2 제품은 이 포럼을 통해 지원됩니다.

미국 외 지역에서 IBM에 연락하는 방법에 관한 정보는 *IBM Software Support Handbook*의 Appendix A를 참조하십시오. 이 문서에 액세스하려면, 웹 사이트 http://www.ibm.com/support/로 가서 페이지 맨 밑에 있는 IBM Software Support Handbook 링크를 누르십시오.

주: 일부 국가에서는 IBM 공인 딜러는 IBM 지원 센터 대신 해당 딜러 지원 부서에 연락해야 합니다.

IBM 한글 지원에 관한 설문

FAX : (02) 3787-0123

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목: IBM® DB2® Universal Database
관리 안내서: 성능
버전 7

책 번호: SA30-0989-01

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☜ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

SA30-0989-01