IBM® DB2® Universal Database

데이터 복구 및 고가용성 안내 및 참조서

버전 7

IBM® DB2® Universal Database

데이터 복구 및 고가용성 안내 및 참조서

버전 7

이 책의 정보와 지원하는 제품을 사용하기 전에 반드시 555 페이지의 『부록K. 주의사항』을 읽으십시오.

목차

이 책의 정보

이 책은 IBM DB2 UDB(Universal Database) 백업, 복원 및 복구 유틸리티에 대한 자세한 정보를 제공하고 이를 사용하는 방법을 보여줍니다. 또한, 고가용성의 중요성에 대해 설명하고 여러 플랫폼에서의 DB2 장애 복구 지원에 대해 설명합니다.

이 책의 사용자

이 매뉴얼은 데이터베이스 관리자, 응용프로그램 프로그래머, 그리고 DB2 데이터베이스 시스템을 이해하고 백업, 복원 및 복구 조작에 대한 책임을 가지고 있거나 이러한 조작을 원하는 기타 DB2 UDB 사용자들을 위한 것입니다.

이 책에서는 사용자가 DB2 Universal Database, SQL(Structured Query Language), 그리고 DB2 UDB가 수행 중인 운영 체제 환경에 익숙한 것으로 가정합니다. DB2 UDB에 대한 일반 정보는 *관리 안내서*를 참조하십시오. SQL에 대한 정보는 *SQL 참조서*를 참조하십시오. DB2 UDB 명령행 처리기의 구성, 호출 및 사용에 대한 정보는 *Command Reference*를 참조하십시오. DB2 UDB API에 대한 정보는 *Administrative API Reference*를 참조하십시오. DB2 관리 API를 포함한 응용프로그램 작성에 대한 일반 정보는 *응용프로그램 빌드 안내서*를 참조하십시오. 이 매뉴얼에는 사용하는 운영 체제에 따른 DB2 설치에 대한 지시사항이 수록되어 있지는 않습니다. 설치 정보는 사용 중인 운영 체제에 적절한 *빠른 시작* 책에서 볼 수 있습니다.

이 책의 구성 방법

이 절에는 다음 주제를 다룹니다.

데이터 복구

제1장 좋은 백업 및 복구 전략 개발

데이터베이스 또는 테이블 공간의 백업 및 복구, 롤 포워드 복구 사용을 포함하여 데이터베이스 및 테이블 공간 복구 방법을 선택할 때 고려할 사항을 설명합니다.

제2장 데이터베이스 백업

데이터베이스 또는 테이블 공간의 백업 사본을 작성하기 위해 사용되는 DB2 백업 유틸리티에 대해 설명합니다.

제3장 데이터베이스 복원

손상되거나 훼손된 데이터베이스나 이전에 백업한 테이블 공간을 재빌드하기 위해 사용되는 DB2 복원 유틸리티에 대해 설명합니다.

제4장 롤 포워드 복구

데이터베이스 복구 로그 파일에 기록된 트랜잭션을 적용하여 데이터베이스를 복구하기 위해 사용되는 DB2 롤 포워드 유틸리티에 대해 설명합니다.

고가용성

제5장 고가용성 및 장애 복구 지원 소개

DB2가 제공하는 고가용성 장애 복구 지원에 대한 개요를 나타냅니다.

제6장 AIX의 고가용성

현재 AIX용 HACMP(High Availability Cluster Multi-processing)의 ES(Enhanced Scalability) 기능을 통해 구현되는 AIX의 고가용성 장애 복구에 대한 DB2 지원에 대해 설명합니다.

제7장 Windows 운영 환경의 고가용성

현재 MSCS(Microsoft Cluster Server)를 통해 구현되는 Windows NT의 고가용성 장애 복구에 대한 DB2 지원에 대해 설명합니다.

제8장 Solaris 운영 환경의 고가용성

현재 SC2.x(Sun Cluster 2.x), SC3.0(Sun Cluster 3.0) 또는 VCS(Veritas Cluster Server)를 통해 구현되는 Solaris 운영 환경의 고가용성 장애 복구에 대한 DB2 지원에 대해 설명합니다.

부록

부록A. 구문 다이어그램을 읽는 방법

구문 다이어그램에서 사용되는 규칙에 대해 설명합니다.

부록B. 경고, 오류 및 완료 메시지

경고 또는 오류 상태가 삭제되었을 때 데이터베이스 관리 프로그램에서 생성되는 메시지들의 해석에 대한 정보를 제공합니다.

부록C. 추가 DB2 명령

복구와 관련된 DB2 명령을 설명합니다.

부록D. 추가 API 및 연관 데이터 구조

복구와 관련된 API와 해당 데이터 구조를 설명합니다.

부록E. 샘플 프로그램 복구

복구 관련 DB2 API와 Embedded SQL 호출을 포함하는 샘플 프로그램에 대한 코드 목록과, 이를 사용하는 방법에 대한 정보를 제공합니다.

부록F. CLP 스크립트 복구

복구 관련 CLP 명령을 포함하는 DB2 명령 스크립트에 대한 코드 목록과, 이를 사용하는 방법에 대한 정보를 제공합니다.

부록G. Tivoli Storage Manager

데이터베이스나 테이블 공간 백업 조작을 관리하기 위해 사용할 수 있는 Tivoli Storage Manager(TSM, 이전에는 ADSM) 제품에 대한 정보를 제공합니다.

부록H. 데이터베이스 복구를 위한 User Exit

데이터베이스 로그 파일로 User Exit 프로그램을 사용하는 방법과 몇 개의 User Exit 샘플 프로그램을 제공합니다.

부록I. 벤더 제품에 대한 API 백업 및 복원

DB2가 다른 벤더 소프트웨어와 통합할 수 있도록 하는 API의 기능과 사용 방법에 대해 설명합니다.

제1부 데이터 복구

제1장 좋은 백업 및 복구 전략 개발

데이터베이스는 하드웨어 또는 소프트웨어 오류(아니면 둘다)로 인해 사용할 수 없게 됩니다. 간혹 저장 문제점, 전원 장애 및 응용프로그램 실패가 발생할 수 있으며, 서로 다른 실패 시나리오에서는 서로 다른 복구 조치를 요구합니다. 제대로 확인된 복구 전략을 세워 데이터를 유실 가능성으로부터 보호하십시오. 복구 전략을 전개할 때 응답해야 하는 몇 가지 질문은 다음과 같습니다. 데이터베이스가 복구 가능하게 되어 있습니까? 데이터베이스를 복구하는데 걸리는 시간은 얼마입니까? 백업 조작 사이에 걸리는 시간은 어느 정도입니까? 백업 사본 및 아카이브 로그 용으로 할당될 수 있는 저장영역 공간의 크기는 얼마입니까? 테이블 공간 레벨 백업이 충분합니까? 아니면 전체 데이터베이스 백업이 필요합니까?

데이터베이스 복구 전략은 모든 정보가 데이터베이스 복구에 필요할 경우 그 정보를 사용할 수 있도록 해야 합니다. 그 전략에는 데이터베이스 백업을 취하기 위한 정기 스케줄과, 파티션된 데이터베이스 시스템의 경우 시스템 스케일이 조정될 때 (데이터베이스 파티션 서버 또는 노드가 추가되거나 제거될 때) 백업이 포함되어야 합니다. 전체 전략에는 명령 스크립트, 응용프로그램, 사용자 정의 함수(UDF), 운영 체제 라이브러리의 저장 프로시듀어 코드 및 로드 사본을 복구하기 위한 프로시듀어도 포함되어야 합니다.

다음 절에서는 여러가지 복구 방법을 설명하며, 비지니스 환경에 가장 적합한 복구 방법을 판단하는 방법을 보여줍니다.

데이터베이스 *백업*의 개념은 다른 모든 데이터 백업과 동일합니다. 원본의 실패 또는 손상시 서로 다른 매체에 데이터 사본을 확보하고 저장합니다. 백업의 가장 간단한 경우는 더이상 트랜잭션이 발생하지 않도록 확인하기 위해 데이터베이스를 종료한 후 단순히 백업하는 것과 관련됩니다. 그러면 어떤 이유로 데이터베이스가 손상되거나 훼손될 경우, 그 데이터베이스를 재빌드할 수 있습니다.

데이터베이스의 재구축을 *복구*라고 합니다. *버전 복구*는 백업 조작 중 작성된 이미지를 사용하는 데이터베이스의 이전 버전의 복원입니다. *롤 포워드 복구*는 데이터베이스 또는 테이블 공간 백업 이미지가 복원된 후에 데이터베이스 로그 파일에 기록된 트랜잭션을 재적용하는 것입니다.

*응급 복구*는 하나 이상의 작업 단위(트랜잭션)에 대한 모든 변경사항이 완료되어 확약되기 전에 실패가 발생할 경우 데이터베이스를 자동으로 복구하는 것입니다. 완료되지 않은 트랜잭션을 구간 복원하고, 파손이 발생했을 때 메모리에 남아 있던 확약된 트랜잭션을 완료하여 이루어집니다.

서로 다른 복구 방법에 대한 자세한 정보는 26 페이지의 『버전 복구』, 27 페이지의 『롤 포워드 복구』 또는 11 페이지의 『시스템 손상 복구』를 참조하십시오.

복구 로그 파일 및 복구 실행기록 파일은 데이터베이스가 작성될 때 자동으로 작성됩니다(5 페이지의 그림1). 복구 로그 파일이나 복구 실행기록 파일을 직접 수정할 수 없습니다. 그러나 손실되거나 손상된 데이터를 복구하기 위해 데이터베이스 백업 이미지를 사용할 필요가 있는 경우 중요합니다.

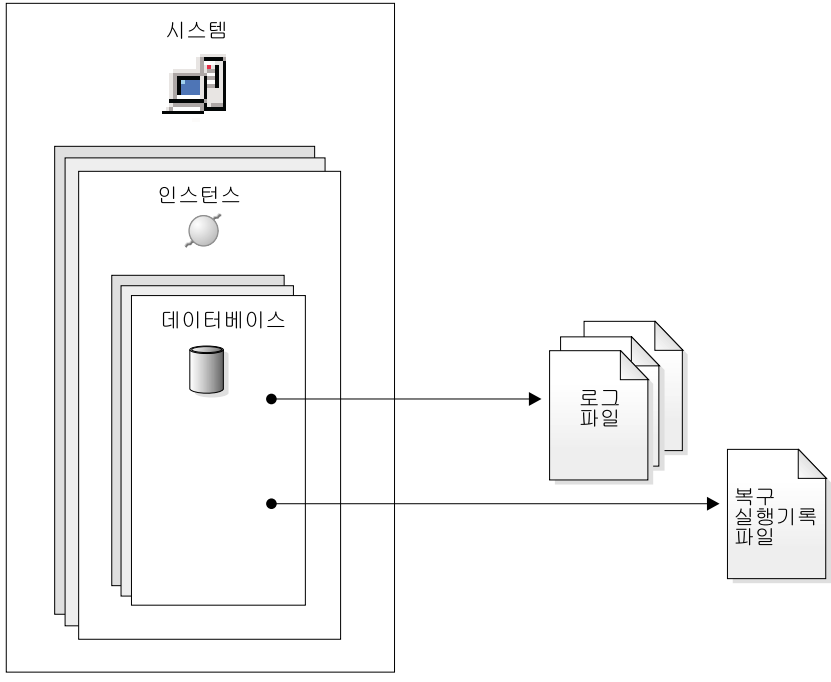


그림 1. 복구 로그 파일 및 복구 실행기록 파일

각 데이터베이스에는 응용프로그램 또는 시스템 오류에서 복구하기 위해 사용되는 *복구 로그*가 포함됩니다. 데이터베이스 백업과 결합되어, 오류가 발생했을 때의 시점까지 데이터베이스의 일관성을 복구하는데 사용됩니다,

*복구 실행기록 파일*에는 데이터베이스의 전부 또는 일부가 주어진 시점으로 복구되어야 하는 경우에 사용될 수 있는 백업 정보의 요약이 들어 있습니다. 이는 백업 및 복원 조작과 같은 복구 관련 이벤트를 추적하는데 사용됩니다.

쉽게 재작성되는 데이터는 복구 불가능한 데이터베이스에 저장할 수 있습니다. 여기에는 읽기 전용 응용프로그램에 사용되는 외부 소스와, 자주 갱신되지 않는 테이블(적은 양의 로깅으로는 복원 조작 후에 추가된 로그 파일 관리 및 롤 포워드의 복잡도를 입증하지 못하는)의 데이터가 포함됩니다. *복구 불가능한 데이터베이스*의 경우, ***logretain*** 및 ***userexit*** 데이터베이스 구성 매개변수가 모두 사용 불가능합니다. 즉, 손상 복구에 필요한 로그만이 보존됩니다. 이러한 로그를 *사용 중인 로그*라고 하며, 이러한 로그는 현재의 트랜잭션 데이

터를 수록합니다. *오프라인* 백업을 사용한 버전 복구는 복구 불가능한 데이터베이스를 복구하는 1차적 방법입니다. (오프라인 백업은 백업 조작이 진행 중일 때 다른 응용프로그램은 데이터베이스를 사용할 수 없음을 의미합니다.) 그러한 데이터베이스는 오프라인으로 복원만 될 수 있습니다. 백업 이미지를 얻을 때 있던 상태로 복원됩니다.

쉽게 재작성할 수 없는 데이터는 복구 가능한 데이터베이스에 저장해야 합니다. 여기에는 데이터가 로드된 후 소스가 소실되는 데이터와, 데이터베이스에 로드된 후에 응용프로그램이나 사용자에 의해 수정되는 데이터가 포함됩니다. *복구 가능한 데이터베이스*는 *logretain* 데이터베이스 구성 매개변수를 『RECOVERY』로 설정되게 하거나 *userexit* 데이터베이스 구성 매개변수가 사용 가능한 경우 또는 둘다입니다. 사용 중인 로그도 응급 복구용으로 사용할 수 있지만 또한 확약 트랜잭션 데이터가 들어 있는 *아카이브 로그*도 사용할 수 있습니다. 그러한 데이터베이스는 오프라인으로 복원만 될 수 있습니다. 백업 이미지를 얻을 때 있던 상태로 복원됩니다. 그러나 롤 포워드 복구의 경우, 사용 중인 로그 및 아카이브 로그를 특정 시점이나 사용 중인 로그의 끝까지 사용하여 데이터베이스를 롤 포워드할 수 있습니다. (즉, 백업 이미지를 얻은 시점을 지날 수 있습니다.)

복구 가능한 데이터베이스 백업 조작은 오프라인으로도 수행할 수 있고 *온라인*으로도 수행할 수 있습니다. (온라인은 백업 조작중에 다른 응용프로그램이 데이터베이스로 연결할 수 있음을 의미합니다.) 데이터베이스 복원 및 롤 포워드 조작은 항상 오프라인으로 수행해야 합니다. 온라인 백업 조작중 롤 포워드 복구 방법은 백업이 복원될 경우, 모든 테이블 변경사항이 캡쳐 및 재적용되도록 합니다.

복구 가능한 데이터베이스인 경우, 전체 데이터베이스가 아닌 개별적 테이블 공간의 백업, 복원 및 롤 포워드를 수행할 수 있습니다. 테이블 공간을 온라인으로 백업한 경우, 여전히 해당 테이블 공간을 사용할 수 있으며 동시 갱신이 로그에 기록됩니다. 테이블 공간에서 온라인 복원 또는 롤 포워드 조작을 수행하면, 조작이 완료될 때까지 테이블 공간 자체는 사용할 수 없지만 사용자는 다른 테이블 공간의 테이블에 액세스할 수 있습니다.

백업 빈도 결정

데이터베이스 백업에 시간 및 시스템 자원이 필요하기 때문에 복구 계획은 규칙적으로 스케줄된 백업 조작을 수행해야 합니다. 전체 데이터베이스 백업과 증분 백업 조작을 조합하여 플랜에 포함 시킬 수도 있습니다(30 페이지의 『증분 백업 및 복구』 참조).

로그를 아카이브하더라도(롤 포워드 복구 허용), 데이터베이스를 정기적으로 완전 백업해야 합니다. 전체 데이터베이스 백업 이미지에서 데이터베이스를 복구하는 것보다 테이블 공간 백업 이미지의 콜렉션에서 데이터베이스를 재빌드하는 것이 더 어렵습니다. 테이블 공간 백업 이미지는 판별된 디스크 실패나 응용프로그램 오류로부터 복구할 경우에 유용합니다.

또한, 백업 이미지와 로그를 겹쳐쓰지 않고, 추가 예방책으로서 둘 이상의 전체 데이터베이스 백업과 관련 로그를 저장하는 방안도 고려해야 합니다.

자주 사용 중인 데이터베이스를 복구하고 롤 포워드할 때 아카이브 로그를 적용하는데 필요한 시간이 상당한 경우, 데이터베이스의 잦은 백업으로 인한 경비도 고려해야 합니다. 이것은 롤 포워드할 때 적용하는데 필요한 아카이브 로그의 수를 감축시킵니다.

데이터베이스가 *온라인* 상태일 때와 *오프라인* 상태일 때 백업 조작을 시작할 수 있습니다. 온라인인 경우, 다른 응용프로그램 또는 프로세스가 데이터베이스에 연결될 수 있으며, 백업 조작이 수행중인 동안 데이터를 읽고 수정할 수 있습니다. 백업 조작이 오프라인에서 수행 중일 경우, 다른 응용프로그램은 데이터베이스에 연결할 수 *없습니다*.

데이터베이스를 사용할 수 없는 시간의 양을 줄이려면, 온라인 백업 조작 사용을 고려하십시오. 온라인 백업 조작은 롤 포워드 복구가 사용 가능할 경우에만 지원됩니다. 롤 포워드 복구가 작동되고 복구 로그 세트가 완료되면, 필요한 데이터베이스를 다시 작성할 수 있습니다. 백업 조작이 수행될 때 걸린 시간에 걸쳐 분포되는 로그가 있는 경우에만 복구에 온라인 백업 이미지를 사용할 수 있습니다.

오프라인 백업 조작은 온라인 백업 조작보다 더 빠릅니다.

데이터베이스에 대량의 long 필드와 대형 오브젝트(LOB) 데이터가 있으면 데이터베이스를 백업하는데 시간이 많이 걸릴 수 있습니다. 백업 유틸리티를 사용하면 선택한 테이블 공간을 백업할 수 있습니다. DMS 테이블 공간을 사용하면, 자체 테이블 공간에 다른 데이터 유형을 저장하여 백업 조작에 필요한 시간을 줄일 수 있습니다. 테이블 데이터를 하나의 테이블 공간에, long 필드 및 LOB 데이터를 다른 테이블 공간에, 색인을 다른 테이블 공간에서 보존할 수 있습니다. Long 필드와 LOB 데이터를 별도의 테이블 공간에 저장하고, 백업 조작을 완료하는데 필요

한 시간은 long 필드와 LOB 데이터를 포함하고 있는 테이블 공간을 백업하지 않도록 선택함으로써 줄일 수 있습니다. Long 필드와 LOB 데이터가 사용자의 비지니스에 중요하면, 이러한 테이블 공간에 대한 복원 조작 완료시 필요한 시간에 대해 해당 테이블 공간의 백업을 고려해야 합니다. 별도의 소스에서 LOB 데이터를 재생산할 수 있을 경우, 테이블이 LOB 컬럼을 포함하도록 테이블을 작성하거나 변경할 때에는 NOT LOGGED 옵션을 선택하십시오.

주: 다음은 long 필드 데이터, LOB 데이터 및 색인을 별도의 테이블 공간에 보존하지만 함께 백업하지는 않을 경우의 특수 고려사항입니다. 모든 테이블 데이터를 포함하지 않은 테이블 공간을 백업하는 경우, 해당 테이블 공간에서 특정 시점 롤 포워드 복구를 실행할 수 없습니다. 테이블에 대한 데이터 유형이 들어 있는 모든 테이블 공간은 특정 시점으로 동시에 롤 포워드되어야 합니다.

테이블을 다시 인식하는 경우, 조작 완료 후에 손상을 입은 테이블 공간을 백업해야 합니다. 테이블 공간을 복원해야 하는 경우, 데이터 재인식을 통해 롤 포워드할 필요는 없습니다.

데이터베이스를 복원하는데 필요한 시간은 두 부분으로 구성됩니다. 한 부분은 백업을 완전히 복원하는데 필요한 시간이고, 데이터베이스를 포워드 복구에 사용할 수 있는 경우 롤 포워드 조작 중에 로그를 적용하는데 필요한 시간입니다. 복구 플랜을 형식화할 때, 복구 비용 및 사용자 비지니스 조작에 대한 영향을 고려해야 합니다. 전반적인 복구 플랜을 테스트하면, 데이터베이스를 복구하는데 필요한 시간이 비지니스상의 요구사항에 비추어 합당한지 여부를 판별하는데 도움이 됩니다. 각 테스트에 따라, 백업할 빈도를 증가시킬 수 있습니다. 롤 포워드 복구가 전략의 일부라면, 이렇게 하여 백업 사이에 아카이브되는 로그의 수를 줄임으로써 복원 조작 후에 데이터베이스를 롤 포워드하는데 필요한 시간을 줄입니다.

저장영역 고려사항

사용할 복구 방법을 결정할 때에는 필요한 저장영역 공간을 고려하십시오.

버전 복구 방법을 사용하려면 데이터베이스의 백업 사본 및 복원된 데이터베이스를 보존할 공간이 필요합니다. 롤 포워드 복구 방법을 사용하려면 데이터베이스 또는 테이블 공간의 백업 사본, 복원된 데이터베이스 및 아카이브된 데이터베이스 로그를 보유할 공간이 필요합니다.

테이블에 long 필드나 대형 오브젝트(LOB) 컬럼이 포함된 경우, 이 데이터를 별도의 테이블 공간에 둘 것을 고려해야 합니다. 이것은 복구 플랜에 영향을 줄 뿐만 아니라, 저장영역 공간 고려사항에도 영향을 줍니다. Long 필드와 LOB 데이터용으로 별도의 테이블 공간이 있고 long 필드와 LOB 데이터를 백업하는데 필요한 시간을 알면, 이 테이블 공간의 백업을 가끔씩만 보관하는 복구 플랜의 사용을 결정할 수도 있습니다. LOB 컬럼을 포함할 테이블을 작성하거나 변경할 경우, 해당 컬럼의 변경사항을 로그하지 않도록 선택할 수도 있습니다. 이 경우, 필요한 로그 공간의 크기와 해당 로그 아카이브 공간의 크기가 줄어듭니다.

LOB이 들어 있는 SMS 테이블 공간의 백업은 원래의 테이블 공간 크기보다 클 수 있습니다. 백업은 테이블 공간 내에 있는 LOB 데이터 크기에 따라 최대 40퍼센트까지 더 클 수 있습니다. 예를 들어, 1GB SMS 테이블 공간(LOB가 있는)을 백업할 경우, 이를 복원할 때 1GB가 넘는 디스크 공간이 필요합니다. 이것은 산재하는 할당(예: UNIX 운영 체제)을 지원하는 파일 시스템에서만 발생합니다.

미디어 고장으로 데이터베이스와 데이터베이스를 재구축하는 기능을 손상시키지 않으려면, 다른 장치에서 데이터베이스 백업, 데이터베이스 로그, 데이터베이스 자체를 유지해야 합니다. 이러한 이유로 인해, 일단 데이터베이스가 작성되면, *newlogpath* 구성 매개변수를 사용하여 데이터베이스 로그를 별도의 장치에 두는 것이 매우 바람직합니다. (이 매개변수와 로깅에 관련된 다른 구성 매개변수에 대해서는 43 페이지의 『데이터베이스 로깅을 위한 구성 매개변수』에 설명되어 있습니다.)

데이터베이스 로그는 대량의 저장영역을 사용할 수 있습니다. 롤 포워드 복구 방법을 사용하려고 계획하는 경우, 아카이브 로그를 관리하는 방법을 결정해야 합니다. 다음을 선택할 수 있습니다.

- 로그를 보유하도록 데이터베이스 로그 경로에 충분한 공간을 전용하십시오.

- 해당 로그가 사용 중인 로그 세트에 더이상 없게 되면, 로그를 저장영역 장치 또는 데이터베이스 로그 경로 디렉토리 이외의 디렉토리에 수동으로 복사하십시오.
- User Exit 프로그램을 사용하여 이들 로그를 해당 환경 내 다른 저장영역 장치에 복사하십시오. 예를 들어, OS/2에서 DB2는 표준 및 비표준 장치에 있는 데이터베이스의 백업 이미지와 데이터베이스 로그 모두의 저장영역을 처리하는 User Exit 프로그램을 지원합니다. (487 페이지의 『부록H. 데이터베이스 복구를 위한 User Exit』에서 자세한 내용을 참조하십시오.)

관련된 데이터 보존

사용자는 데이터베이스 설계의 일부로서, 테이블 사이에 존재하는 관계를 알고 있을 것입니다. 이러한 관계는 트랜잭션이 둘 이상의 테이블을 갱신하는 응용프로그램 수준일 수도 있고, 테이블간에 참조 무결성이 존재하거나 하나의 테이블의 트리거가 다른 테이블에 영향을 주는 데이터베이스 레벨일 수도 있습니다. 복구 플랜을 개발할 때에는 이러한 관계에 대해 고려해야 합니다. 사용자는 관련된 데이터 세트를 같이 백업하려는 경우가 있습니다. 데이터 세트는 테이블 공간 또는 데이터베이스 레벨에서 설정될 수 있습니다. 관련된 데이터 세트로 한 곳에 보존함으로써, 모든 데이터가 일관되는 지점까지 복구할 수 있게 됩니다. 테이블 공간에서 특정 시점 롤 포워드 복구를 실행할 수 있으려면, 이것은 특히 중요한 사항입니다.

다른 운영 체제 사용

둘 이상의 운영 체제가 있는 환경에서 작업중인 경우, 대부분의 경우 백업 및 복구 플랜이 통합될 수 없음을 고려해야 합니다. 즉, 일반적으로 특정 운영 체제에서 백업한 데이터베이스를 다른 운영 체제에서 복원할 수 없습니다. 이 경우, 각 운영 체제에 대한 복구 플랜을 각각 개별적이고 독립적으로 보존해야 합니다.

그러나 Sun Solaris 및 HP 사이의 상호 플랫폼 백업 및 복원 조작은 지원됩니다. 시스템 사이에 백업 이미지를 전송할 경우, 2진 모드로 전송해야 합니다. 목표 시스템에서, 데이터베이스는 원래 데이터베이스가 작성된 시스템과 같은 코드 페이지와 지역을 사용하여 작성해야 합니다.

운영 체제 사이에 테이블을 이동시켜야 하는데 상호 플랫폼 백업 및 복원 지원을 환경에서 사용할 수 없는 경우, **db2move** 명령을 사용하거나 가져오기 또는 로드 유틸리티 다음의 내보내기 유틸리티를 사용할 수 있습니다. 이들 유틸리티에 대한 자세한 정보는 *데이터 이동 유틸리티 안내 및 참조서*를 참조하십시오.

시스템 손상 복구

데이터베이스에 대해 트랜잭션(또는 작업 단위(UOW))은 예기치 않게 인터럽트될 수 있습니다. 작업 단위의 일부인 모든 변경사항이 완료되어 확약되기 전에 실패가 발생할 경우, 데이터베이스는 불일치 또는 사용 불가능 상태로 남아 있게 됩니다. *응급 복구*는 데이터베이스를 다시 일치되고 사용 가능한 상태로 이동하는 프로세스입니다. 완료되지 않은 트랜잭션을 구간 복원하고, 파손이 발생했을 때 메모리에 남아 있던 확약된 트랜잭션을 완료하여 이루어집니다(그림2). 데이터베이스가 일관성 있고 사용 가능한 상태인 경우, "일관성 지점"이라고 하는 곳에 도달합니다.

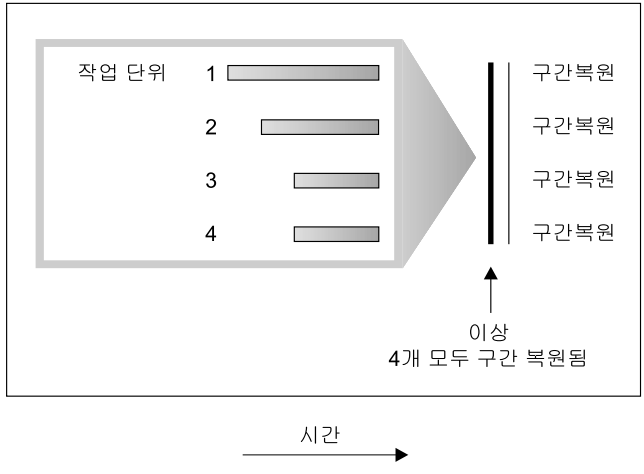


그림 2. 작업 단위(UOW) 구간 복원(응급 복구)

*트랜잭션 실패*가 발생하면 데이터베이스나 데이터베이스 관리 프로그램이 비정상적으로 종료되도록 하는 심각한 오류 또는 상태가 됩니다. 실패 발생시 디스크에 저장되지 않은, 부분적으로 완료된 작업 단위(UOW)는 데이터베이스를 불일치 및 사용 불가능 상태로 둡니다. 트랜잭션이 실패한 후에는 데이터베이스를 복구해야 합니다. 트랜잭션 실패를 야기할 수 있는 상태는 다음과 같습니다.

- 데이터베이스 관리 프로그램과 데이터베이스 파티션이 중단되도록 하는 머신의 전원 장애
- DB2가 중단되도록 하는 심각한 운영 체제 오류

불완전 작업 단위(UOW)의 구간 복원이 데이터베이스 관리 프로그램에 의해 자동으로 수행되도록 하려면, 자동 재시작(*autorestart*) 데이터베이스 구성 매개변수를 ON으로 설정하여 사용 가능하게 하십시오. (이는 기본값입니다. 이 매개변수에 대한 자세한 정보는 *관리 안내서: 성능*을 참조하십시오.) 자동 재시작 작동을 원하지 않을 경우에는 데이터베이스 실패가 발생할 때 RESTART DATABASE 명령을 발행해야 합니다. db2diag.log 파일은 데이터베이스 재시작 조작이 시작되는 시기를 기록합니다.

포워드 복구용으로 사용 가능한 데이터베이스에 응급 복구가 적용된 경우(즉, *logretain* 구성 매개변수가 RECOVERY로 설정되거나 *userexit* 구성 매개변수가 사용 가능한 경우), 응급 복구 동안 개별 테이블 공간에서 기인한 오류가 발생하면, 해당 테이블 공간은 오프라인되어야 하고 복구할 때까지 액세스할 수 없습니다. 응급 복구는 계속됩니다. 응급 복구 완료시 데이터베이스의 다른 테이블 공간은 보통 여전히 사용할 수 있으며, 데이터베이스로 연결을 설정할 수 있습니다.

손상된 테이블 공간 복구

손상된 테이블 공간에는 액세스할 수 없는 하나 이상의 컨테이너가 있습니다. 이는 영구(예: 불량 디스크) 또는 임시(예: 오프라인 디스크 또는 마운트되지 않은 파일 시스템) 미디어 문제점으로 인해 발생한 것일 수 있습니다.

손상된 테이블 공간이 시스템 카탈로그 테이블 공간이면, 데이터베이스는 다시 시작될 수 없습니다. 컨테이너 문제점을 수정할 수 없어 원래 데이터를 그대로 두는 경우, 다음 옵션만을 사용할 수 있습니다.

- 데이터베이스 복원
- 카탈로그 테이블 공간 복원(테이블 공간 복원은 데이터베이스가 롤 포워드되어야 하므로 복구 가능한 데이터베이스에 대해서만 유효합니다.)

손상된 테이블 공간이 시스템 카탈로그 테이블 공간이 *아니면*, DB2는 가능한 많은 데이터베이스를 사용할 수 있도록 작성합니다.

손상된 테이블 공간이 유일한 임시 테이블 공간인 경우, 데이터베이스에 연결되는 즉시 사용자는 새로운 임시 테이블 공간을 작성할 수 있습니다. 일단 작성되면, 새

로운 임시 테이블 공간이 사용되고, 임시 테이블 공간이 필요한 정상적인 데이터베이스 조작이 재개될 수 있습니다. 원한다면, 오프라인 임시 테이블 공간을 삭제할 수 있습니다. 시스템 임시 테이블 공간을 사용한 테이블 재구성에 대해 다음과 같은 특수사항을 고려해야 합니다.

- 데이터베이스 또는 데이터베이스 관리 프로그램 구성 매개변수 *indexrec*가 RESTART로 설정되면, 유효하지 않은 모든 색인을 데이터베이스 활성화 동안 재구성해야 합니다. 즉, 여기에는 작성 단계에서 손상된 재구성 색인도 포함됩니다.
- 손상된 임시 테이블 공간에서 재구성 요청이 불완전하면, *indexrec* 구성 매개변수를 ACCESS로 설정하여 재시작이 실패하지 않도록 해야 합니다.

복구 가능한 데이터베이스에 있는 테이블 공간 복구

손상된 테이블 공간은 응급 복구가 필요하므로, 오프라인 상태에 놓이며 액세스가 능하지 않으며, 롤 포워드 보류 상태에 있습니다. 추가 문제점이 없는 경우 재시작 조작이 성공합니다. 다음과 같이 손상된 테이블 공간을 다시 사용할 수 있습니다.

- 원래 데이터를 잃지 않고 손상된 컨테이너를 수정한 다음, 테이블 공간 롤 포워드 조작을 완료하십시오. (롤 포워드 조작은 우선 오프라인에서 보통 상태로 가져오려고 시도합니다.)
- 손상된 컨테이너를 수정한 후 테이블 공간 복원 조작을 수행한 다음(원래 데이터를 유실하거나 유실하지 않고), 롤 포워드 조작을 수행하십시오.

복구 불가능한 데이터베이스에 있는 테이블 공간 복구

응급 복구가 필요하고 로그가 무기한 보존되는 것이 아니므로, 사용자가 손상된 테이블 공간을 삭제하려는 경우에만 재시작 조작이 성공합니다. (복구를 완료했다는 것은 손상된 테이블 공간을 일관성있는 상태로 복구하는데 필요한 로그 레코드가 삭제되었음을 의미하므로, 이러한 테이블 공간에 대해 유효한 조치는 이를 삭제하는 것입니다.)

규정화되지 않은 데이터베이스 재시작 조작을 호출하여 이를 수행할 수 있습니다. 손상된 테이블 공간이 없으면 성공합니다. 실패하면(SQL0290N), db2diag.log 파일에서 현재 손상된 테이블 공간의 전체 목록을 조사할 수 있습니다.

- 데이터베이스 재시작 조작이 완료된 후 이 모든 테이블 공간을 삭제하려는 경우, 다른 데이터베이스 재시작 조작을 시작하여, DROP PENDING TABLESPACES 옵션 아래에 손상된 모든 테이블 공간을 나열할 수 있습니다. 손상된 테이블 공간이 DROP PENDING TABLESPACES 목록에 포함된 경우, 테이블 공간은 삭제 보류 상태가 되며, 복구 이후의 유일한 옵션은 테이블 공간을 삭제하는 것입니다. 이 테이블 공간을 복구하지 않고 재시작 조작을 계속합니다. 손상된 테이블 공간이 DROP PENDING TABLESPACES 목록에 포함되지 않는 경우, 데이터베이스 재시작 조작은 SQL0290N으로 실패합니다.
- 사용자가 이러한 테이블 공간을 삭제하지 않으려는 경우(데이터를 유실하지 않으려는 경우), 다음을 수행하는 옵션을 사용할 수 있습니다.
  - 기다린 후 손상된 컨테이너를 수정한 다음(원래 데이터를 잃지 않고), 다시 데이터베이스 재시작 조작을 시도하십시오.
  - 데이터베이스 복원 조작을 수행하십시오.

주: 테이블 공간 이름을 DROP PENDING TABLESPACES 목록에 삽입하는 것은 테이블 공간이 삭제 보류 상태가 됨을 의미하는 것은 아닙니다. 이것은 재시작 조작 중 테이블 공간이 손상된 경우에만 발생합니다. 재시작 조작에 성공하고 나면, DROP TABLESPACE 문을 발행하여 삭제 보류 상태에 있는 각 테이블 공간을 삭제해야 합니다. (테이블 공간이 이 상태에 있는지 알려면 LIST TABLESPACES 명령을 호출하십시오.) 이러한 방식으로 공간을 다시 요청하거나 테이블 공간을 다시 작성할 수 있습니다.

미디어 오류의 영향 줄이기

미디어 오류의 가능성을 줄이고, 이러한 유형의 오류에서 복구를 단순화하려면 다음을 수행하십시오.

- 중요한 데이터베이스에 대한 데이터와 로그를 담고 있는 디스크를 반영하거나 복제하십시오.
- RAID(Redundant Array of Independent Disks) 구성을 사용하십시오(예: RAID 레벨 5). RAID에 대한 자세한 정보는 15 페이지의 『디스크 오류 방지』를 참조하십시오.

- 파티션된 데이터베이스 환경에서, 카탈로그 노드의 데이터 및 로그를 처리할 엄밀한 프로시듀어를 설정하십시오. 이 노드는 데이터베이스를 유지보수하는데 중요하므로, 다음을 수행하십시오.
  - 이 노드가 신뢰 가능한 디스크에 상주하는지 확인하십시오.
  - 이 노드를 복사하십시오.
  - 자주 백업하십시오.
  - 이 노드에 사용자 데이터를 넣지 마십시오.

디스크 오류 방지

디스크 손상 때문에 손상된 데이터 또는 로그에 대해 걱정하는 경우, 디스크 오류 허용한계 형식 사용을 고려하십시오. 일반적으로 *디스크 배열*을 사용하여 가능합니다. 디스크 배열은 응용프로그램에 대해 하나의 큰 디스크 드라이브로 나타나는 디스크 드라이브 콜렉션으로 이루어져 있습니다.

디스크 배열에는 다중 디스크에 걸친 파일의 분산인 *디스크 스트라이핑(disk striping)*과 디스크 및 데이터 패러티 이중복사를 나타내는 것이 있습니다.

디스크 배열은 종종 RAID(Redundant Array of Independent Disks)로 참조되기도 합니다. 디스크 배열은 운영 체제 또는 응용프로그램 레벨에서 소프트웨어를 통해 제공될 수도 있습니다. 하드웨어 디스크 배열과 소프트웨어 디스크 배열간의 구별점은 입출력(I/O) 요청에 대해 CPU 처리가 다루어지는 형식입니다. 하드웨어 디스크 배열의 경우, 디스크 제어기는 입출력 활동을 관리하고 이에 비해 소프트웨어 디스크 배열 사용은 운영 체제 또는 응용프로그램에 의해 관리됩니다.

하드웨어 디스크 배열: 하드웨어 디스크 배열에서, 다중 디스크는 디스크 제어기에 의해 사용되고 관리되며, 자체 CPU로 완료합니다. 배열을 만드는 디스크를 관리하는데 필요한 모든 논리는 디스크 제어기에 포함되어 있고, 이 구현은 독립적인 운영 체제입니다.

기능과 성능에서 다른 몇 가지 유형의 RAID 구조가 있지만, 오늘날에는 일반적으로 RAID 레벨 1 및 레벨 5만 사용됩니다.

RAID 레벨 1은 또한 디스크 이중복사 또는 디스크 양방향이라고 합니다. *디스크 이중복사*는 하나의 디스크 제어기를 사용하여 특정 디스크에서 다른 디스크로 데

이터(완료 파일)를 복사합니다. *디스크 양방향*은 디스크가 다른 디스크 제어기(두 개의 SCSI 어댑터)에 접속된 경우를 제외하고 디스크 이중복사와 같습니다. 데이터 보호에 효과적입니다. 데이터에 오류가 발생해도 데이터를 다른 디스크에서 여전히 액세스할 수 있습니다. 디스크 양방향의 경우, 데이터 보호에 영향을 미치지 않고 디스크 제어기는 실패할 수 있습니다. 성능은 좋지만 이러한 구현을 위해서는 일반적인 디스크 수의 두 배가 필요합니다.

RAID 레벨 5에는 모든 디스크에 걸쳐 섹터에 의한 스트라이핑 데이터와 패리티가 있습니다. 패리티에는 전용 드라이브에 저장하기 보다는 데이터를 끼워 넣습니다. 데이터 보호에 효과적입니다. 디스크 오류가 발생하는 경우, 데이터는 스트라이프된 패리티 정보와 함께 다른 디스크의 정보를 사용하여 계속 액세스될 수도 있습니다. 읽기 성능은 좋지만 쓰기 성능은 좋지 않습니다. RAID 레벨 5 구성에는 최소한 세 개의 식별 디스크가 필요합니다. 오버헤드에 대해 필요한 디스크 공간은 배열의 디스크 수에 따라 달라집니다. 5개 디스크로 된 RAID 레벨 5 구성의 경우, 공간 오버헤드는 20퍼센트입니다.

RAID(RAID 레벨 0이 아닌) 디스크 배열을 사용할 경우, 실패한 디스크에서 사용자는 배열의 데이터에 액세스할 수 없습니다. hot-pluggable 또는 hot-swappable 디스크는 배열에서 사용되며, 대체 디스크는 배열 사용 중 실패한 디스크로 교체될 수 있습니다. RAID 레벨 5의 경우, 두 디스크에서 동시에 오류가 발생하면, 모든 데이터는 유실됩니다. (그러나 동시에 디스크에서 오류가 발생하는 경우는 거의 없습니다.)

로그에 대해 RAID 레벨 1 하드웨어 디스크 배열이나 소프트웨어 디스크 배열을 사용할 것을 고려할 수 있습니다(17 페이지의 『소프트웨어 디스크 배열』 참조). 이렇게 하면 실패 발생 시점까지 복구할 수 있으며, 로그에 중요한 양호한 쓰기 성능을 제공하기 때문입니다. 신뢰성이 중요하고(디스크 실패 후 데이터 복구하는데 시간을 지체할 수 없으므로) 쓰기 성능은 그렇게 중요하지 않은 경우에는 RAID 레벨 5의 하드웨어 디스크 배열을 사용하십시오. 또한, 쓰기 성능이 중요하고 추가 디스크 공간 비용은 중요하지 않은 경우, 로그뿐만 아니라 데이터에 대해 RAID 레벨 1의 하드웨어 디스크 배열을 사용하십시오.

사용 가능한 RAID 레벨에 대한 자세한 정보는 다음 주소를 방문하십시오.

```
http://www.acnc.com/04_01_00.html
```

소프트웨어 디스크 배열: 소프트웨어 디스크 배열은 하드웨어 디스크 배열처럼 수행하지만(15 페이지의 『하드웨어 디스크 배열』 참조), 디스크 트래픽은 서버에서 수행 중인 응용프로그램이나 운영 체제에 의해 관리됩니다. 다른 모든 프로그램처럼 소프트웨어 배열은 CPU 및 시스템 자원과 경합해야 합니다. CPU 제한 시스템에서는 좋은 옵션이 아니며, 전체 디스크 배열 성능은 서버의 CPU 로드와 용량에 따라 결정되는 것을 기억해야 합니다.

일반 소프트웨어 디스크 배열은 RAID-1과 함께 디스크 이중복사를 제공합니다(15 페이지의 『하드웨어 디스크 배열』 참조). 여유 디스크가 필요하지만, 소프트웨어 디스크 배열은 값비싼 디스크 제어기가 필요하지 않기 때문에 구현시 상대적으로 비용이 덜 듭니다.

주의:
디스크 배열에서 운영 체제 부트 드라이브가 있으면 드라이브 오류시 시스템을 시작할 수 없게 합니다. 디스크 배열 수행중에 드라이브에 오류가 발생하는 경우, 디스크 배열은 드라이브로의 액세스를 허용할 수 없습니다. 부트 드라이브는 디스크 배열에서 분리되어야 합니다.

트랜잭션 오류의 영향 줄이기

트랜잭션 오류의 영향을 줄이려면, 다음을 수행하십시오.

- 인터럽트되지 않는 전원
- 데이터베이스 로그에 적절한 디스크 공간 확보
- 파티션된 데이터베이스 환경에서 데이터베이스 파티션 서버간에 신뢰 가능한 통신 링크
- 파티션된 데이터베이스 환경에서 시스템 시계 동기화(166 페이지의 『파티션된 데이터베이스 시스템에서 시계 동기화』 참조)

파티션된 데이터베이스 환경의 트랜잭션 장애 복구

파티션된 데이터베이스 환경에서 트랜잭션 실패가 발생할 경우, 보통 실패한 데이터베이스 파티션 서버와 트랜잭션에 참여했던 다른 데이터베이스 파티션 서버 모두 데이터베이스 복구가 필요합니다.

- 응급 복구는 앞선 상태가 올바른 다음에 실패한 데이터베이스 파티션 서버에서 발생합니다.
- 아직 사용 중인 다른 데이터베이스 파티션 서버에서 *데이터베이스 파티션 장애 복구*는 실패가 감지되면 즉시 발생합니다.

파티션된 데이터베이스 환경에서, 응용프로그램이 제출된 데이터베이스 파티션 서버는 조정자 노드이며, 응용프로그램에 대해 작동하는 첫 번째 에이전트는 조정자 에이전트입니다. 조정자 에이전트는 다른 데이터베이스 파티션 서버로 작업을 분배해야 하며, 트랜잭션과 관련된 것이 어떤 것인지 추적합니다. 응용프로그램이 트랜잭션에 대해 COMMIT문을 나타내는 경우, 조정자 에이전트는 2단계 확약 프로토콜을 사용하여 트랜잭션을 확약합니다. 1단계 중에, 조정자 노드는 트랜잭션에 참여하고 있는 다른 모든 데이터베이스 파티션 서버에 PREPARE 요청을 분배합니다. 그런 다음, 이들 서버는 다음 중 하나로 응답합니다.

READ-ONLY 이 서버에서 데이터 변경이 발생하지 않았습니다.

YES 이 서버에서 데이터 변경이 발생했습니다.

NO 오류 때문에, 서버는 확약할 준비가 되어 있지 않습니다.

서버 중 하나가 NO로 응답할 경우, 트랜잭션은 구간 복원됩니다. 그렇지 않으면, 조정자 노드가 2단계를 시작합니다.

2단계 중에, 조정자 에이전트는 확약 로그 레코드를 기록한 다음, COMMIT 요청을 YES로 응답하는 모든 서버에 분배합니다. 다른 모든 데이터베이스 파티션 서버가 확약된 후, 이들은 조정자 노드로 COMMIT 확인을 전송합니다. 조정자 에이전트가 참여한 모든 서버로부터 COMMIT 확인을 수신하면 트랜잭션이 완료됩니다. 이때, 조정자 에이전트는 FORGET 로그 레코드를 기록합니다.

2단계 확약에 대한 자세한 정보는 *관리 안내서: 계획*을 참조하십시오.

사용 중인 데이터베이스 파티션 서버의 트랜잭션 장애 복구

어떤 데이터베이스 파티션 서버가 다른 서버가 다운되었음을 발견하면, 고장난 데이터베이스 파티션 서버와 관련된 모든 작업이 중단됩니다.

- 계속 사용 중인 데이터베이스 파티션 서버가 응용프로그램에 대한 조정자 노드이고 응용프로그램이 실패한 데이터베이스 파티션 서버(COMMIT 준비가 되지 않은)에서 실행 중일 경우, 조정자 에이전트는 오류 복구로 인터럽트됩니다. 조정자 에이전트가 COMMIT 처리의 두 번째 단계에 있을 경우, 응용프로그램에 SQL0279N이 리턴되고, 다음으로 데이터베이스 연결이 끊어집니다. 그렇지 않으면, 조정자 에이전트는 트랜잭션에 참여하고 있는 다른 서버에 ROLLBACK 요청을 분배하고, 응용프로그램에는 SQL1229N이 표시됩니다.
- 실패한 데이터베이스 파티션 서버가 응용프로그램에 대한 조정자 노드라면, 사용 중인 서버의 응용프로그램에 대해 작업중인 에이전트는 오류 복구를 하기 위해 인터럽트됩니다. 현재 트랜잭션이 준비되어 있으며 트랜잭션 결과를 기다리고 있지 않은 경우, 각 서버에서 지역으로 현재 트랜잭션이 구간 복원됩니다. 이 경우, 트랜잭션은 사용 중인 데이터베이스 파티션에서 불확실하게 남아 있는데, 조정자 노드는 사용할 수 없기 때문에 이에 대해 알지 못합니다.

  단계 확약중 이상 실패 트랜잭션의 해결 방법에 대한 자세한 정보는 *관리 안내서: 계획*을 참조하십시오.
- 실패한 데이터베이스 파티션 서버에 응용프로그램이 연결되어 있지만(실패하기 전에), 지역 데이터베이스 서버나 실패한 데이터베이스 파티션 서버는 모두 조정자 노드가 아닐 경우, 이 응용프로그램에 대해 작업중인 에이전트가 인터럽트됩니다. 조정자 노드는 다른 데이터베이스 파티션 서버에 ROLLBACK 또는 연결해제 메시지를 보냅니다. 조정자 노드가 SQL0279를 리턴할 경우, 트랜잭션은 데이터베이스 파티션 서버에서만 2단계 확약중 이상 실패 상태입니다.

실패한 서버로 요청을 전송하고자 하는 모든 프로세스(예: 에이전트 또는 교착상태 검출기)에는 요청을 전송할 수 없다는 정보가 전달됩니다.

오류 데이터베이스 파티션 서버에서의 트랜잭션 오류 복구

트랜잭션 실패로 데이터베이스 관리 프로그램이 비전상적으로 종료될 경우, RESTART 옵션을 사용하는 **db2start** 명령을 발행하여 프로세서가 재시작되고 나면 데이터베이스 관리 프로그램을 재시작할 수 있습니다. 프로세서를 재시작할 수 없으면, 다른 프로세서에서 **db2start** 명령을 발행하여 데이터베이스 관리 프로그램을 재시작할 수 있습니다. 자세한 정보는 *Command Reference*를 참조하십시오.

데이터베이스 관리 프로그램이 비정상적으로 종료되면, 서버의 데이터베이스 파티션이 불일치 상태로 남아 있을 수 있습니다. 이를 사용 가능한 상태로 만들려면, 다음과 같이 데이터베이스 파티션 서버에서 응급 복구를 트리거하면 됩니다.

- RESTART DATABASE 명령을 통해(명시적)
- *autorestart* 데이터베이스 구성 매개변수가 ON으로 설정된 경우 CONNECT 요청을 통해(암시적)

데이터베이스의 모든 완전한 트랜잭션 효과를 보증하기 위해, 사용 중인 로그 파일의 로그 레코드에 손상 복구가 다시 적용됩니다. 변경사항이 다시 적용되고 나면, 2단계 확약 중 이상 실패 트랜잭션을 *제외하고* 모든 미확약 트랜잭션이 지역적으로 구간 복원됩니다. 파티션된 데이터베이스 환경에는 두 가지 유형의 2단계 확약 중 이상 실패 트랜잭션이 있습니다.

- 조정자 노드가 아닌 데이터베이스 파티션 서버에서 트랜잭션이 준비되어 있으나 확약되어 있지는 않을 경우, 트랜잭션은 불확실한 상태입니다.
- 조정자 노드에서, 트랜잭션이 확약되어 있지만 완료된 것으로 기록되어 있지 않을 경우(즉, FORGET 레코드가 기록되어 있지 않을 경우), 트랜잭션은 2단계 확약중 이상 실패 상태입니다. 조정자 에이전트가 응용프로그램에 대해 작업하는 모든 서버로부터 모든 COMMIT 확인을 받지 않은 경우 이러한 상황이 발생합니다.

손상 복구는 다음 중 한 가지를 수행함으로써 모든 2단계 확약중 이상 실패 트랜잭션을 해결하고자 합니다. 수행된 조치는 데이터베이스 파티션 서버가 응용프로그램에 대한 조정자 노드였는지에 따라 달라집니다.

- 재시작된 서버가 응용프로그램에 대한 조정자 노드가 아닐 경우, 서버는 조정자 에이전트로 조회 메시지를 전송하여 트랜잭션의 결과를 알아냅니다.
- 재시작된 서버가 응용프로그램용 조정자 노드인 *경우*, 메시지를 조정자 에이전트가 여전히 COMMIT 승인을 기다리는 모든 다른 에이전트(종속 에이전트)로 송신합니다.

손상 복구를 통해 모든 2단계 확약중 이상 실패 트랜잭션을 해결할 수 없을 경우가 있습니다(예를 들면, 일부 데이터베이스 파티션 서버가 사용 가능하지 않을 경우). 이 경우, SQL 경고 메시지 SQL1061W가 리턴됩니다. 잠금 및 사용 중인 로그 공간과 같은 2단계 확약 중 이상 실패 트랜잭션이 자원들을 보유하고 있으므

로, 사용 중인 로그 공간을 2단계 확약 중 이상 실패 트랜잭션이 차지하고 있어서 데이터베이스에 대해 어떤 변경도 수행할 수 없는 시점에 이를 수 있습니다. 그렇기 때문에, 응급 복구 후 2단계 확약 중 이상 실패 트랜잭션이 남아 있는지 판별하고, 2단계 확약 중 이상 실패 트랜잭션을 가능한 한 신속하게 해결하는데 필요한 모든 데이터베이스 파티션 서버를 복구해야 합니다.

2단계 확약중 이상 실패 트랜잭션을 해결하는데 필요한 하나 이상의 서버를 적시에 복구할 수 없으며 다른 서버의 데이터베이스 파티션에 액세스가 필요할 경우, 경험적 결정을 내려 수동으로 2단계 확약중 이상 실패 트랜잭션을 해결할 수 있습니다. LIST INDOUBT TRANSACTIONS 명령(*Command Reference* 참조)을 사용하여 서버에서 2단계 확약중 이상 실패 트랜잭션을 조회, 확약, 구간 복원할 수 있습니다.

주: LIST INDOUBT TRANSACTIONS 명령은 분산 트랜잭션 환경에서도 사용됩니다. 두 가지 유형의 2단계 확약 중 이상 실패 트랜잭션을 구분하기 위해, LIST INDOUBT TRANSACTIONS 명령이 리턴한 출력의 *originator* 필드에 다음 중 하나가 표시됩니다.

- 파티션된 데이터베이스 환경에서 발원된 트랜잭션을 나타내는 DB2 Universal Database Enterprise - Extended Edition
- 트랜잭션이 분산 환경에서 발생했음을 나타내는 XA

분산된 환경에 대한 자세한 정보는 *관리 안내서: 계획*을 참조하십시오.

실패한 데이터베이스 파티션 서버 식별

데이터베이스 파티션 서버에 오류가 발생하면, 응용프로그램은 보통 다음 SQLCODE 중 하나를 수신합니다. 실패한 데이터베이스 관리 프로그램을 검출하는 방법은 수신된 SQLCODE에 따라 다릅니다.

SQL0279N

이 SQLCODE는 트랜잭션에 속한 데이터베이스 파티션 서버가 COMMIT 처리중에 종료되면 수신됩니다.

SQL1224N

이 SQLCODE는 실패한 데이터베이스 파티션 서버가 트랜잭션에 대한 조정자 노드인 경우 수신됩니다.

SQL1229N

이 SQLCODE는 실패한 데이터베이스 파티션 서버가 트랜잭션에 대한 조정자 노드가 아닌 경우 수신됩니다.

데이터베이스 파티션 서버가 실패한 것을 결정하는 것은 두 단계의 프로세스입니다. SQLCODE SQL1229N과 연관된 SQLCA에는 *sqlerrd* 필드의 6번째 배열 위치에서 오류를 검출한 서버의 노드 번호가 들어 있습니다. (서버용으로 기록된 노드 번호는 db2nodes.cfg 파일의 노드 번호와 일치합니다.) 오류를 검출한 데이터베이스 파티션 서버에서는 실패한 서버의 노드 번호를 나타내는 메시지가 db2diag.log 파일에 기록됩니다.

주: 다중 논리 노드가 프로세서에서 사용 중이면, 하나의 논리 노드의 오류는 동일한 프로세서에서 다른 논리 노드가 실패하도록 합니다.

데이터베이스 파티션 서버의 오류를 복구하는 방법

1. 오류를 일으킨 문제점을 정정하십시오.
2. 데이터베이스 파티션 서버에서 **db2start** 명령을 발행하여 데이터베이스 관리 프로그램을 재시작하십시오.
3. 실패한 데이터베이스 파티션 서버 또는 서버에서 RESTART DATABASE 명령을 발행하여 데이터베이스를 재시작하십시오.

호스트에서 2단계 확약중 이상 실패 트랜잭션 복구

응용프로그램이 트랜잭션 중 호스트 또는 AS/400 데이터베이스 서버에 액세스한 경우, 2단계 확약중 이상 실패 트랜잭션을 복구하는 방법에 약간의 차이가 있습니다.

호스트 또는 AS/400 데이터베이스 서버에 액세스하기 위해 DB2 Connect가 사용됩니다. DB2 Connect에 DB2 동기 지점 관리 프로그램이 구성된 경우 복구 단계는 다릅니다.

DB2 Connect에 DB2 동기 지점 관리 프로그램이 구성된 경우의 복구

일반적으로 호스트 또는 AS/400 서버에서의 2단계 확약중 이상 실패 트랜잭션 복구는 트랜잭션 관리자(TM) 및 DB2 동기 지점 관리 프로그램(SPM)에 의해 자동으로 수행됩니다. 호스트 또는 AS/400 서버에서의 2단계 확약중 이상 실패 트랜

잭션은 지역 DB2 위치에서는 자원을 보유하지 않고, 트랜잭션이 해당 위치에서 2단계 확약중 이상 실패 상태로 있는 한 호스트 또는 AS/400 서버에서 자원을 보유합니다. 호스트 또는 AS/400 관리자가 경험적 방법으로 결정을 내려야 한다고 판단한 경우, 관리자는 호스트 또는 AS/400 서버에서 트랜잭션을 확약할 것인지 구간 복원할 것인지를 판별하기 위해 현지 DB2 데이터베이스 관리자(전화 등을 통해)에게 문의할 것입니다. 이 경우, LIST DRDA INDOUBT TRANSACTIONS 명령을 사용하여 DB2 Connect 인스턴스에서의 트랜잭션 상태를 판별할 수 있습니다. SNA 통신 환경과 관련된 대부분의 상황에서 다음 단계를 지시사항으로 삼을 수 있습니다.

1. 아래와 같이 SPM에 연결하십시오.

```
db2 => connect to db2spm
 데이터베이스 연결 정보
 데이터베이스 제품       = SPM0500
 SQL 권한 부여 ID        = CRUS
 지역 데이터베이스 별명 = DB2SPM
```

2. LIST DRDA INDOUBT TRANSACTIONS 명령을 사용하여 SPM으로 알려진 2단계 확약중 이상 실패 트랜잭션을 표시하십시오. 다음의 예에서는 SPM으로 알려진 하나의 2단계 확약중 이상 실패 트랜잭션을 보여줍니다. db_name은 호스트 또는 AS/400 서버의 지역 별명입니다. partner_lu는 호스트 또는 AS/400 서버의 완전한 LU 이름입니다. 이 별명은 호스트 또는 AS/400 서버의 최상의 식별자로서, 호스트 또는 AS/400 서버에서 호출자가 제공합니다. luwid는 트랜잭션에 고유한 식별자를 제공하며, 모든 호스트 및 AS/400 서버에서 사용 가능합니다. 해당 트랜잭션이 표시되면, `uow_status` 필드를 사용하여 값이 C(확약)인지 R(구간 복원)인지 여부에 따라 트랜잭션 결과를 판별할 수 있습니다. LIST DRDA INDOUBT TRANSACTIONS 명령을 WITH PROMPTING 매개변수와 함께 실행하면, 해당 트랜잭션을 대화식으로 확약, 구간 복원 또는 무시할 수 있습니다. 자세한 정보는 *Command Reference*를 참조하십시오.

```
db2 => list drda indoubt transactions
 DRDA Indoubt Transactions:
 1.db_name: DBAS3    db_alias: DBAS3    role: AR
   uow_status: C  partner_status: I  partner_lu: USIBMSY.SY12DQA
 corr_tok: USIBMST.STB3327L
 luwid: USIBMST.STB3327.305DFDA5DC00.0001
 xid: 53514C2000000017 00000000544D4442 0000000000305DFD A63055E962000000
      00035F
```

3. partner_lu 및 luwid에 대한 2단계 확약중 이상 실패 트랜잭션이 표시되지 않거나 LIST DRDA INDOUBT TRANSACTIONS 명령이 다음과 같이 리턴될 경우,

   ```
   db2 => list drda indoubt transactions
   SQL1251W  경험적 조회로 어떠한 데이터도 리턴되지 않았습니다.
   ```

   트랜잭션이 구간 복원된 것입니다.

   흔하지는 않으나 또 하나의 상황이 발생할 수 있습니다. partner_lu에 대해 적절한 luwid를 가진 2단계 확약중 이상 실패 트랜잭션이 표시되었지만 uow_status가 "I"인 경우, SPM은 트랜잭션이 확약될 것인지 구간 복원될 것인지 여부를 알지 못합니다. 이 경우, DB2 Connect 워크스테이션에서 트랜잭션을 확약하거나 구간 복원하려면 WITH PROMPTING 매개변수를 사용해야 합니다. 그런 다음, DB2 Connect가 경험적 결정에 의거하여 호스트 또는 AS/400 서버와 재동기화되도록 하십시오.

DB2 Connect가 DB2 동기 지점 관리 프로그램을 사용하지 않는 경우의 복구

TCP/IP 연결을 사용하여 OS/390용 DB2를 DB2 Connect Personal Edition 또는 DB2 Connect Enterprise Edition으로부터 다중 사이트 갱신하며 DB2 동기 지점 관리 프로그램이 사용되지 않는 경우 이 절의 정보를 참조하십시오. 이 경우의 2단계 확약중 이상 실패 트랜잭션 복구는 DB2 동기 지점 관리 프로그램과 관련된 2단계 확약중 이상 실패 트랜잭션 복구와는 다릅니다. 이러한 환경에서 2단계 확약중 이상 실패 트랜잭션이 발생하는 경우, 누가 문제점을 탐지하느냐에 따라 클라이언트, 데이터베이스 서버, 트랜잭션 관리자(TM) 데이터베이스에서 경고 항목이 생성됩니다. 경고 항목은 db2alert.log 파일에 위치합니다. 경고에 대한 자세한 정보는 *문제점 해결 안내서*에서 참조하십시오.

TM, 해당 데이터베이스 및 연결을 모두 다시 사용할 수 있게 되면, 곧바로 모든 2단계 확약중 이상 실패 트랜잭션의 재동기화가 자동으로 이루어집니다. 데이터베이스 서버에서 경험적 방식으로 결정이 이루어지도록 강제하기 보다는 자동 재동기화가 이루어지도록 해야 합니다. 그러나 경험적 방식을 사용해야 한다면 다음 단계를 지시사항으로 사용하십시오.

주: DB2 동기 지점 관리 프로그램이 관련되기 때문에 LIST DRDA INDOUBT TRANSACTIONS 명령을 사용할 수 없습니다.

1. OS/390 호스트에서, DISPLAY THREAD TYPE(INDOUBT) 명령을 발행하십시오.

   경험적 방식으로 완료하려는 트랜잭션을 이 목록에서 확인합니다. DISPLAY 명령에 대한 세부사항은 *OS/390용 DB2 Command Reference*를 참조하십시오. 표시되는 LUWID는 트랜잭션 관리자 데이터베이스의 동일한 luwid에 대응될 수 있습니다.

2. 수행하려는 작업에 따라 RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) 명령을 발행하십시오.

   RECOVER 명령에 대한 세부사항은 *OS/390용 DB2 Command Reference*를 참조하십시오.

재해 복구

재해 복구 용어를 사용하여 화재, 지진 또는 기타 천재지변이 일어난 경우에 데이터베이스를 복원하는데 필요한 활동을 설명합니다. 재해 복구 플랜에는 다음 중 하나 이상이 포함될 수 있습니다.

- 비상시에 사용되는 사이트
- 데이터베이스를 복구하는 다른 머신
- 데이터베이스 백업 및 아카이브 로그의 오프사이트 저장영역

재해 복구 플랜에서 전체 데이터베이스를 다른 머신에서 복구하려면, 적어도 하나의 전체 데이터베이스 백업과 데이터베이스의 모든 아카이브 로그가 필요합니다. 로그를 아카이브되어 있는 상태로 대기중인 데이터베이스에 적용하여 이 데이터베이스를 최신 상태로 유지할 수 있습니다. 또는, 데이터베이스 백업 및 로그 아카이브를 대기 사이트에서 유지하는 것을 선택하고 재난 발생 후에만 복원 및 롤 포워드 조작을 실행할 수 있습니다. (이 경우, 최근 데이터베이스 백업이 바람직합니다.) 그러나 재난이 발생할 경우, 보통 재난 발생 시점까지의 모든 트랜잭션을 복구하기는 불가능합니다.

재해 복구를 위한 테이블 공간 백업의 유용성은 오류 범위에 따라 달라집니다. 일반적으로, 재해 복구를 하려면 전체 데이터베이스를 복원해야 하기 때문에 주요한 재해가 발생할 경우 예비 사이트에서 전체 데이터베이스 백업을 수행해야 합니다. 모든 테이블 공간에 대한 별도의 백업 이미지를 가지고 있더라도, 이를 사용하여

데이터베이스를 복원할 수는 없습니다. 디스크가 손상되는 재난이 발생한 경우, 해당 디스크에서의 각 테이블 공간의 테이블 공간 백업이 복구에 사용될 수 있습니다. 디스크 오류(또는 다른 이유로) 때문에 컨테이너로의 액세스할 수 없는 경우, 다른 위치에서 컨테이너를 복원할 수 있습니다. 추가 정보는 128 페이지의 『복원 조작 중 테이블 공간 컨테이너 재정의(경로 재지정 복원)』에서 자세한 내용을 참조하십시오.

테이블 공간 백업과 전체 데이터베이스 백업은 둘다 복구 플랜에 있어서 중요한 역할을 수행합니다. 데이터를 백업, 복원 및 롤 포워드할 수 있는 DB2 기능은 재해 복구 플랜의 기초를 제공합니다. 비지니스를 보호하려면, 복구 프로시듀어를 테스트해야 합니다.

버전 복구

*버전 복구*는 백업 조작 중 작성된 이미지를 사용하는 데이터베이스의 이전 버전의 복원입니다. 복구 불가능한 데이터베이스(즉, 보존 로그가 없는 데이터베이스)의 경우 이 복구 방법을 사용할 수 있습니다. RESTORE DATABASE 명령의 WITHOUT ROLLING FORWARD 옵션을 사용하여 복구 가능 데이터베이스에 이 방법을 사용할 수도 있습니다. 데이터베이스 복원 조작은 초기에 작성된 백업 이미지를 사용하여 전체 데이터베이스를 다시 빌드합니다. 데이터베이스 백업은 백업 작성시와 동일한 상태로 데이터베이스를 복원할 수 있도록 합니다. 그러나 백업시부터 실패시까지의 모든 작업 단위(UOW)가 유실됩니다(27 페이지의 그림3 참조).

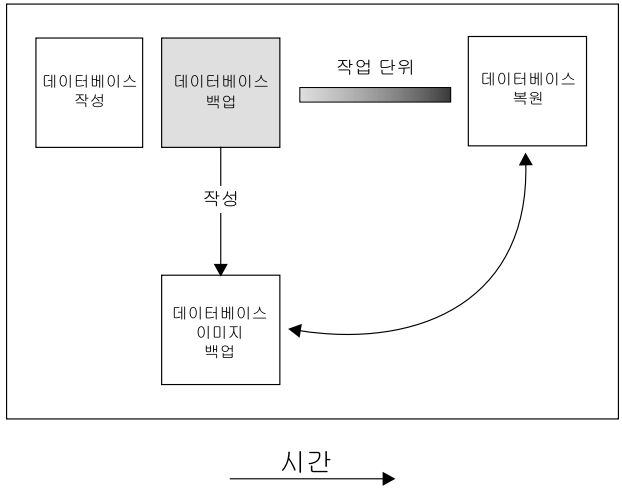


그림 3. 버전 복구. 데이터베이스는 최근 백업 이미지에서 복원되지만, 백업 및 실패 시간 사이에 처리된 모든 작업 단위(UOW)가 유실됩니다.

버전 복구 방법을 사용하여 정기적으로 데이터베이스 전체 백업 스케줄을 잡고 이를 수행해야 합니다.

파티션된 데이터베이스 환경에서, 데이터베이스는 수많은 데이터베이스 파티션 서버(또는 노드)에 걸쳐 위치지정됩니다. 모든 파티션을 복원해야 하며, 복원에 사용한 백업은 동시에 모두 선택되어야 합니다. (각 데이터베이스 파티션은 백업되어 개별적으로 복원됩니다.) 동시에 선택된 각 데이터베이스 파티션의 백업은 *버전 백업*이라고 합니다.

롤 포워드 복구

롤 포워드 복구 방법을 사용하려면, 데이터베이스를 백업하고, 로그를 아카이브해야 합니다. (*logretain* 또는 *userexit* 데이터베이스 구성 매개변수 중 하나 또는 둘 다를 작동시켜서 수행합니다. 사용하는 로그 프로시듀어와 관련하여 해야 할 결정에 대해서는 35 페이지의 『복구 로그 이해』를 참조하십시오.) 데이터베이스를 복원하고 WITHOUT ROLLING FORWARD 옵션을 지정하는 것은 버전 복구 방법을 사용하는 것과 같습니다. 오프라인 백업 이미지가 작성되는 시점과 동일한 상태로 복원됩니다. 복원 데이터베이스 조작에 대한 WITHOUT ROLLING

FORWARD 옵션을 지정하지 않은 상태에서 데이터베이스를 복원하는 경우, 해당 데이터베이스는 복원 종료시 롤 포워드 보류 상태가 됩니다. 그러면 롤 포워드 복구가 수행될 수 있게 됩니다.

다음 두 가지 유형의 롤 포워드 복구를 고려해야 합니다.

- *데이터베이스 롤 포워드 복구*. 이러한 유형의 롤 포워드 복구에서, 데이터베이스 로그에 기록되는 트랜잭션은 다음과 같은 데이터베이스 복원 조작에 적용됩니다(그림4 참조). 데이터베이스 로그는 데이터베이스의 모든 변경사항을 기록합니다. 이 방법은 데이터베이스를 특정 시점의 상태 또는 실패 바로 이전 상태(즉, 사용 중인 로그의 끝)로 복구합니다.

  파티션된 데이터베이스 환경에서, 데이터베이스는 여러 데이터베이스 파티션에 위치지정됩니다. 적절한 롤 포워드 복구를 실행하려면, 모든 데이터베이스 파티션은 롤 포워드되어 모든 파티션이 같은 레벨에 있는지 확인해야 합니다. 단일 데이터베이스 파티션을 복원해야 하는 경우, 로그 끝까지 롤 포워드 복구를 실행하여 데이터베이스의 다른 파티션과 같은 레벨로 만들 수 있습니다. 하나의 데이터베이스 파티션이 롤 포워드 중일 경우에는 로그 끝까지의 복구만 사용할 수 있습니다. 특정 시점 복구는 모든 데이터베이스 파티션에 적용합니다.

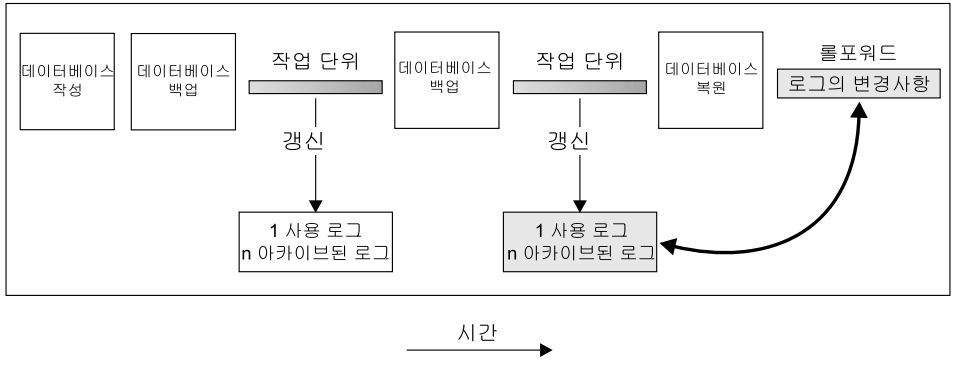


그림 4. 데이터베이스 롤 포워드 복구. 장기 수행 트랜잭션의 경우에는 사용 중 로그가 여러 개 있을 수 있습니다.

- *테이블 공간 롤 포워드 복구*. 데이터베이스에서 포워드 복구가 작동 가능한 경우, 테이블 공간을 백업, 복원 및 롤 포워드할 수도 있습니다. 테이블 공간 복원 및롤 포워드 조작을 수행하려면 전체 데이터베이스(즉, 모든 테이블 공간) 또는 하나 이상의 개별 테

이블 공간의 백업 이미지가 필요합니다. 또한 복구되는 테이블 공간에 영향을 미치는 로그 레코드를 필요로 합니다. 로그를 다음 두 지점 중 하나로 롤 포워드할 수 있습니다.

- 로그 끝
- 특정 시점(*특정 시점* 복구라고 함)

테이블 공간 롤 포워드 복구는 다음 두 상황에서 사용될 수 있습니다.

- 복원 조작 후에 테이블 공간은 항상 롤 포워드 보류 상태에 있으며, 반드시 롤 포워드되어야 합니다. ROLLFORWARD DATABASE 명령(168 페이지의 『ROLLFORWARD DATABASE 명령』 참조)을 호출하여 테이블 공간에 대한 로그를 특정 시점이나 로그의 끝에 적용할 수 있습니다.
- 하나 이상의 테이블 공간이 응급 복구 후 *롤 포워드 보류* 상태에 있으면, 먼저 테이블 공간 문제점을 정정하십시오. 이 경우, 테이블 공간 문제점의 정정 작업에는 복원 데이터베이스 조작이 포함되지 않습니다. 예를 들어, 정전이 되면 테이블 공간은 롤 포워드 보류 상태에 있게 됩니다. 복원 데이터베이스 조작은 이 경우 요청되지 않습니다. 테이블 공간에 대한 문제점이 정정되고 나면, ROLLFORWARD DATABASE 명령을 사용하여 테이블 공간에 반하는 로그를 로그의 끝으로 적용할 수 있습니다. 응급 복구 전 문제점이 정정되면, 응급 복구가 데이터베이스를 일관성있고 사용 가능한 상태로 만들 수 있습니다.

  주: 오류가 발생한 테이블 공간에 시스템 카탈로그 테이블이 들어 있으면, 데이터베이스를 시작할 수 없습니다. SYSCATSPACE 테이블 공간을 먼저 복원하고 로그 끝까지 롤 포워드 복구를 실행해야 합니다.

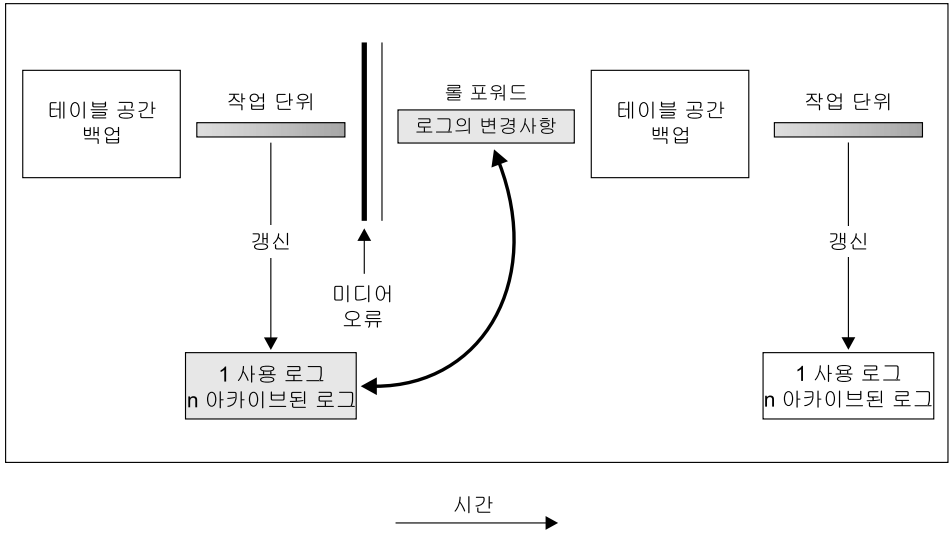


그림 5. 테이블 공간 롤 포워드 복구. 장기 수행 트랜잭션의 경우에는 사용 중 로그가 여러 개 있을 수 있습니다.

파티션된 데이터베이스 환경에서, 테이블 공간을 *특정 시점*으로 롤 포워드하면, 테이블 공간이 상주하는 노드 목록(데이터베이스 파티션)을 지원할 필요가 없습니다. DB2는 모든 파티션에 대한 롤 포워드 요청을 제출합니다. 이것은 테이블 공간이 상주하는 모든 데이터베이스 파티션에서 테이블 공간이 복원되어야 함을 의미합니다.

파티션된 데이터베이스 환경에서, 테이블 공간을 로그 *끝*으로 롤 포워드하는 경우, 모든 파티션에 테이블 공간을 롤 포워드하지 *않으려면* 데이터베이스 파티션의 목록을 지원할 필요가 없습니다. 롤 포워드 보류 상태인 모든 파티션의 테이블 공간을 로그 *끝*까지 롤 포워드할 경우, 데이터베이스 파티션 목록을 제공할 필요가 없습니다. 기본적으로, 데이터베이스 롤 포워드 요청은 모든 파티션으로 송신됩니다.

증분 백업 및 복구

데이터베이스, 특히 웨어하우스 *크기*가 테라바이트와 페타바이트 범위로 계속해서 확장되고 있으므로, 이러한 데이터베이스를 백업 및 복구하는데 필요한 하드웨어 자원과 시간도 확실히 증가하고 있습니다. 대형 데이터베이스를 다룰 경우에 전체 데이터베이스 및 테이블 공간 백업이 항상 최상의 방법은 아닙니다. 그러한 데이터베이스의 여러 사본에 대한 저장 요구사항이 엄청나기 때문입니다. 다음 사항을 고려하십시오.

- 웨어하우스에서 적은 양의 데이터만 변경할 경우, 전체 데이터베이스를 백업할 필요가 없습니다.
- 테이블 공간을 기존 데이터베이스에 추가한 다음 테이블 공간 백업만 취하는 것은 위험합니다. 백업한 테이블 공간 외에 어떤 것도 테이블 공간 백업 사이에 변경되지 않는다고 보장할 수 없기 때문입니다.

DB2는 증분 백업 및 복구를 지원합니다(long 필드나 대형 오브젝트(LOB) 데이터는 제외). *증분 백업*은 이전 백업이 취해진 이후로 갱신된 페이지만 포함하는 백업 이미지입니다. 갱신된 데이터 및 색인 페이지 외에도, 각 증분 백업 이미지에는 정상적으로 전체 백업 이미지에 저장되는 모든 초기 데이터베이스 메타데이터(데이터베이스 구성, 테이블 공간 정의, 데이터베이스 실행기록 등)도 포함됩니다.

두 가지 유형의 증분 백업이 지원됩니다.

- *증분*. 증분 백업 이미지는 최근의 성공한 전체 백업 조작 이후에 변경된 모든 데이터베이스 데이터의 사본입니다. 이것은 또한 시간을 거쳐 취해진 일련의 증분 백업 시리즈가 각각 이전 증분 백업 이미지를 가지고 있으므로 누적 백업 이미지라고도 합니다. 증분 백업 이미지의 선임자는 항상 최근에 성공한 동일 오브젝트의 전체 백업입니다.
- *델타*. 델타 또는 증분 델타 백업 이미지는 문제의 테이블 공간에서 최근의 성공한 백업(전체, 증분 또는 델타) 이후로 변경된 모든 데이터베이스 데이터의 사본입니다. 이는 차등 또는 비누적 백업 이미지라고도 합니다. 델타 백업 이미지의 선임자는 델타 백업 이미지에서 각 테이블 공간의 사본을 포함하는 최근의 성공한 백업입니다.

증분 및 델타 백업 이미지의 주요 차이점은 연속 백업에서 시간에 걸쳐 계속해서 변경되는 오브젝트가 사용될 경우의 작동에 있습니다. 각 연속 증분 이미지에는 이전의 증분 이미지에 대한 전체 내용과, 이전 백업이 생성된 이후로 변경되었거나 새 것인 데이터가 있습니다. 델타 백업 이미지에는 이전 이미지가 생성된 이후로 변경된 페이지만 포함됩니다.

데이터베이스 및 테이블 공간 증분 백업의 조합은 조작의 온라인 및 오프라인 모드 둘다에서 허용됩니다. 백업 전략을 계획할 때는 주의하십시오. 데이터베이스 및 테이블 공간 증분 백업을 결합한다는 것은 데이터베이스 백업(또는 여러 테이블 공

간의 테이블 공간 백업)의 선임자가 단일 이미지를 필요로 하지 않지만 다른 시간에 취해진 고유한 이전 데이터베이스 세트와 테이블 공간 백업일 수 있음을 내포하기 때문입니다.

데이터베이스나 테이블 공간을 일치된 상태로 재빌드하려면, 복원될 전체 오브젝트(데이터베이스 또는 테이블 공간)의 일치된 이미지로 복구 프로세스를 시작한 다음 아래에 설명된 순서로 적절한 증분 백업 이미지 각각을 적용해야 합니다(『증분 백업 이미지로부터 복원』 참조).

DB2는 데이터베이스 갱신사항을 추적할 수 있도록 새로운 데이터베이스 구성 매개변수인 *trackmod*를 지원합니다. 이 매개변수는 두 가지의 승인값 중 하나를 수반할 수 있습니다.

- NO. 이 구성에 대해 증분 백업이 허용되지 않습니다. 어떤 방법으로든지 데이터베이스 페이지 갱신사항을 추적하거나 기록하지 않습니다.
- YES. 이 구성에 대해 증분 백업이 허용됩니다. 갱신 추적이 가능할 경우, 변경사항은 인스턴스에서 임의 데이터베이스에 처음으로 연결될 때 적용됩니다. 증분 백업을 취하려면 먼저 전체 데이터베이스 백업이 필요합니다.

기존 데이터베이스의 경우 기본 *trackmod* 설정은 `NO`이며, 새 데이터베이스의 경우 `YES`입니다.

SMS 테이블 공간의 경우, 이 추적의 세분성은 테이블 공간 레벨에 있습니다. DMS 테이블 공간의 경우, 세분성은 데이터 및 색인 페이지에 대해서는 extent 레벨에, 다른 페이지 유형에 대해서는 테이블 공간 레벨에 있습니다.

최소한이기는 하지만, 데이터베이스에 대한 갱신사항을 추적하는 것은 데이터를 갱신하거나 삽입하는 트랜잭션의 런타임 성능에 영향을 미칠 수 있습니다.

증분 백업 이미지로부터 복원

증분 백업 이미지로부터의 복원 조작은 항상 다음 단계들로 구성됩니다.

1. 증분 목표 이미지 식별.

   DBA는 먼저 복원될 최종 이미지를 판별하고 DB2 복원 유틸리티를 통해 증분 복원 조작을 요청해야 합니다. 이 이미지는 복원될 마지막 이미지이므로 증분 복원의 목표 이미지라고 합니다. 이 이미지에 대한 증분 복원 명령은 이 목

표 이미지의 구성 및 공간 정의를 사용하여 새로운 데이터베이스의 작성을 초기화할 수 있습니다. 증분 목표 이미지는 RESTORE DATABASE 명령에서 TAKEN AT 매개변수를 사용하여 지정됩니다.

2. 후속 증분 백업 이미지 각각을 적용할 수 있는 기준을 설정한 최근의 전체 데이터베이스 또는 테이블 공간 이미지 복원.
3. 2단계에서 복원한 기준 이미지의 위에서, 생성된 순서대로 필요한 전체 또는 테이블 공간 증분 백업 이미지 각각에 대한 복원.
4. 1단계의 목표 이미지가 두 번 읽혀질 때까지 3단계 반복. 목표 이미지에는 완전한 증분 복원 조작 동안 두 번 액세스합니다. 첫 번째 액세스에서는 이미지에서 초기 데이터만 읽혀지며, 어떤 사용자 데이터도 읽혀지지 않습니다. 완전한 이미지는 두 번째 액세스에서만 읽혀지고 처리됩니다.

   증분 복원 조작의 목표 이미지는 복원 조작 중에 작성될 데이터베이스에 대한 올바른 실행기록, 데이터베이스 구성 및 테이블 공간 정의를 사용하여 초기에 구성되도록 두 번 액세스합니다. 초기 전체 데이터베이스 백업 이미지가 취해진 이후로 테이블 공간이 제거된 경우, 그 이미지에 대한 테이블 공간 데이터가 백업 이미지에서 읽혀지지만 증분 복원 처리 중에 무시됩니다.

증분 백업 이미지 세트를 복원하려면, RESTORE DATABASE 명령에 TAKEN AT *timestamp* 옵션을 지정하십시오. 복원할 마지막 이미지에 대한 시간소인을 지정하십시오. 예를 들면, 다음과 같습니다.

```
db2 restore db sample incremental automatic taken at 20001228152133
```

이는 DB2 복원 유틸리티가 자동으로 위에 설명된 각각의 단계를 수행하도록 합니다. 초기 처리 단계에서, 시간소인이 20001228152133인 백업 이미지가 읽혀지고 복원 유틸리티는 데이터베이스, 데이터베이스의 실행기록, 테이블 공간 정의가 존재하며 유효한지를 확인합니다.

두 번째 처리 단계에서는 요청된 복원 조작을 수행하는데 필요한 백업 이미지 체인을 빌드하기 위해 데이터베이스 실행기록이 조회됩니다. 어떤 이유로 이러한 상황이 가능하지 않고 필수 이미지의 완전한 체인을 빌드하는데 DB2를 사용할 수 없는 경우, 복원 조작은 종료하고 오류 메시지가 리턴됩니다. 이러한 경우, 자동 복원이 가능하지 않으므로 수동 복원 프로시듀어로 진행해야 합니다.

주: PRUNE HISTORY 명령에서는 FORCE 옵션을 사용하지 마십시오. 이 명령의 기본 조작은 최근의 전체 데이터베이스 백업 이미지에서 복구하는데 필요할 수 있는 실행기록 항목을 삭제하지 못하게 하지만, FORCE 옵션을 사용하면 자동 복원 조작에 필요한 항목을 삭제할 수 있습니다.

데이터베이스 실행기록을 사용할 수 없으면, 이 절의 시작 부분에 요약된 단계들을 수행하여 수동으로 증분 복원 조작을 수행할 수 있습니다. 예를 들면, 다음과 같습니다.

```
1. db2 restore database sample incremental taken at <ts>
   여기서,
     <ts>는 복원할 증분 백업 이미지를 가리킵니다.
2. db2 restore database sample incremental taken at <ts1>
   여기서,
     <ts1>은 초기 전체 데이터베이스(또는 테이블 공간) 이미지를 가리킵니다.
3. db2 restore database sample incremental taken at <tsX>
   여기서,
     <tsX>는 작성 순서에서 각 증분 백업 이미지를 가리킵니다.
4. 3단계를 반복하여, 이미지 <ts>까지 각각의 증분 백업 이미지를 복원합니다.
```

데이터베이스 복원 조작을 시도 중인데 테이블 공간 증분 백업 이미지가 생성된 경우, 테이블 공간 이미지는 백업 시간소인 순서로 복원해야 합니다.

자동 증분 복원 제한사항

1. 복원하려는 백업 이후로 테이블 공간의 이름을 바꾸고 새 이름을 사용하여 테이블 공간 레벨 복원을 발행할 경우, 데이터베이스 실행기록을 사용하는 필수 백업 이미지 체인이 올바르게 생성되지 않아서 오류가 발생합니다.

   예를 들면, 다음과 같습니다.

   ```
   db2 backup db sample -> <ts1>
   db2 backup db sample incremental -> <ts2>
   db2 rename tablespace from userspace1 to t1
   db2 restore db sample tablespace ('t1') incremental automatic taken at <ts2>
   ```

   일시적인 해결책: 수동 증분 복원을 사용하십시오.

2. 데이터베이스를 제거할 경우, 데이터베이스 실행기록이 삭제됩니다. 제거된 데이터베이스를 복원할 경우, 데이터베이스 실행기록은 복원된 백업 시간에 해당되는 상태로 복원되고 그 시간 이후의 모든 실행기록 항목은 유실됩니다. 그런 다음 이러한 유실된 실행기록 항목을 사용해야 하는 자동 증분 복원을 수행하려고 하면, RESTORE 유틸리티가 잘못된 백업 체인을 복원하려고 시도하여 "순서를 벗어남" 오류를 리턴합니다.

   예를 들면, 다음과 같습니다.

```
db2 backup db sample -> <ts1>
db2 backup db sample incremental -> <ts2>
db2 backup db sample incremental delta -> <ts3>
db2 backup db sample incremental delta -> <ts4>
db2 drop db sample
db2 restore db sample incremental automatic taken at <ts2>
db2 restore db sample incremental automatic taken at <ts4>
```

일시적인 해결책:

- 수동 증분 복원을 사용하십시오.
- 자동 증분 복원을 발행하기 전에 먼저 이미지 <ts4>에서 실행기록 파일을 복원하십시오.

복구 로그 이해

모든 데이터베이스는 이와 연관된 로그를 가지고 있습니다. 이들 로그는 데이터베이스 변경사항에 대한 레코드를 보유합니다. 데이터베이스가 마지막 전체, 오프라인 백업을 넘은 지점으로 복원될 필요가 있는 경우, 로그는 실패 지점으로 데이터를 롤 포워드하도록 요청됩니다.

순환, *캡쳐* 및 *아카이브*의 세 가지 DB2 로깅 유형이 있으며, 각각은 다른 레벨의 복구 기능을 제공합니다.

- *순환* 로그는 새로운 데이터베이스를 작성할 때의 기본 활동입니다. (*logretain* 데이터베이스 구성 매개변수 설정은 NO입니다.) 이러한 유형의 로그시 데이터베이스의 전체, 오프라인 백업만이 유효합니다. 이름이 제시하는 것처럼, 순환 로그는 온라인 로그의 『링』을 사용하여 트랜잭션 실패 및 시스템 고장으로부터의 복구를 제공합니다. 로그는 현재 트랜잭션의 무결성을 보장하는 지점에서만 사용되며 보유됩니다. 순환 로깅은 마지막 전체 백업 조작 이후에 수행된 트랜잭션을 통해 데이터베이스를 롤 포워드하도록 허용하지 않습니다. 마지막 백업 조작 이후의 모든 변경사항이 유실됩니다. 전체 백업이 이루어질 때 데이터베이스는 오프라인되어야 합니다(사용자에게 액세스 가능하지 않음). 이러한 유형의 복원 조작은 전체 백업이 취해진 특정 시점까지 데이터를 복구하므로, *버전 복구*라고 합니다.

그림6에서는 순환 로그가 사용 중일 때 사용 중인 로그가 로그 파일의 링을 사용함을 보여줍니다.

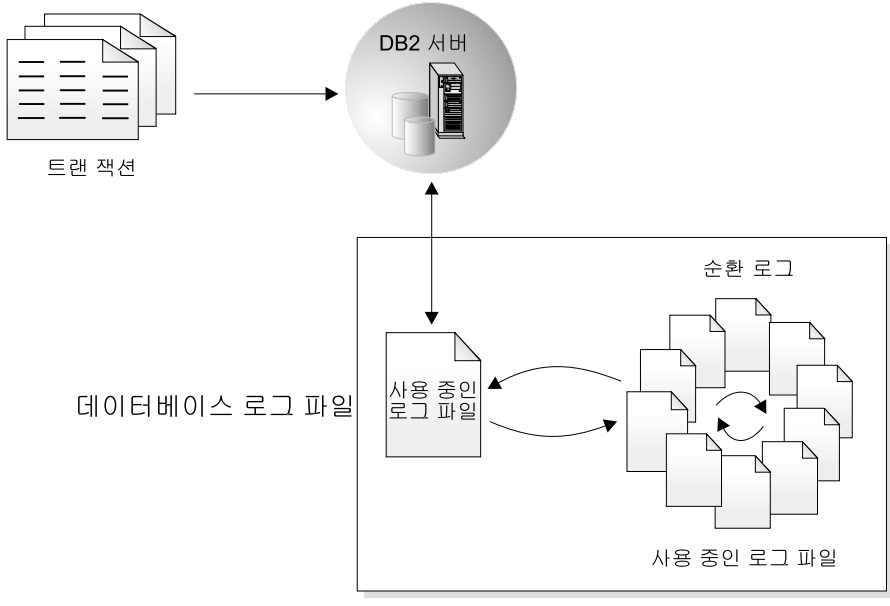


그림 6. 순환 로그

사용 중인 로그는 응급 복구시 오류(시스템 전원 또는 응용프로그램 오류)가 불일치 상태로 데이터베이스를 떠날 수 없게 합니다. RESTART DATABASE 명령은 필요한 경우 사용 중인 로그로 일관성 있고 사용할 수 있는 상태로 데이터베이스를 이동시킵니다. 응급 복구 시, 이 로그에 기록된 아직 확약되지 않은 변경사항은 구간 복원됩니다. 확약되었지만 아직 메모리(버퍼 풀)에서 디스크(데이터베이스 컨테이너)로 쓰여지지 않은 변경사항은 재실행됩니다. 이러한 조치로 데이터베이스의 무결성을 확인할 수 있습니다. 사용 중인 로그는 데이터베이스 로그 경로 디렉토리에 위치합니다.

- *캡쳐* 로깅은 *logretain* 데이터베이스 구성 매개변수를 CAPTURE로 설정하면 구성됩니다. 캡쳐 로그는 복제 처리를 위해 사용됩니다. 로그 파일은 복제 처리가 완료될 때(자동으로 삭제되는 시기)까지 보유됩니다. 모든 DB2 유틸리티는 순환 로깅을 조절한 것과 같은 방법으로 로깅 모드를 조절합니다. 즉, 온라인 백업 조작이나, 테이블 공간 백업 및 복원 조작이나, 또는 롤 포워드 조작이 허용되지 않습니다. RECOVERY NO 옵션을 지정하는 로드 조작은 테이블 공간을 백업 보류 상태에 놓지 않습니다.

- *아카이브* 로그는 롤 포워드 복구에 고유하게 사용됩니다. 이는 *logretain* 데이터베이스 구성 매개변수를 RECOVERY로 설정하면 구성됩니다. 아카이브된 로그는 다음일 수 있습니다.

  온라인 아카이브 로그

  사용 중인 로그의 변경사항이 정상적인 처리를 위해 더이상 필요하지 않을 때, 로그가 닫히고 아카이브 로그가 됩니다. 아카이브 로그는 데이터베이스 로그 경로 디렉토리에 저장될 때 *온라인*이라고 합니다(그림7 참조).

  오프라인 아카이브 로그

  아카이브 로그는 데이터베이스 로그 경로 디렉토리에 저장되지 않을 때 *오프라인*이라고 합니다(38 페이지의 그림8 참조). 또한 User Exit 프로그램을 사용함으로써 데이터베이스 로그 경로 디렉토리 이외의 위치에 아카이브 로그를 저장할 수 있습니다. (추가 정보는 487 페이지의 『부록H. 데이터베이스 복구를 위한 User Exit』를 참조하십시오.)

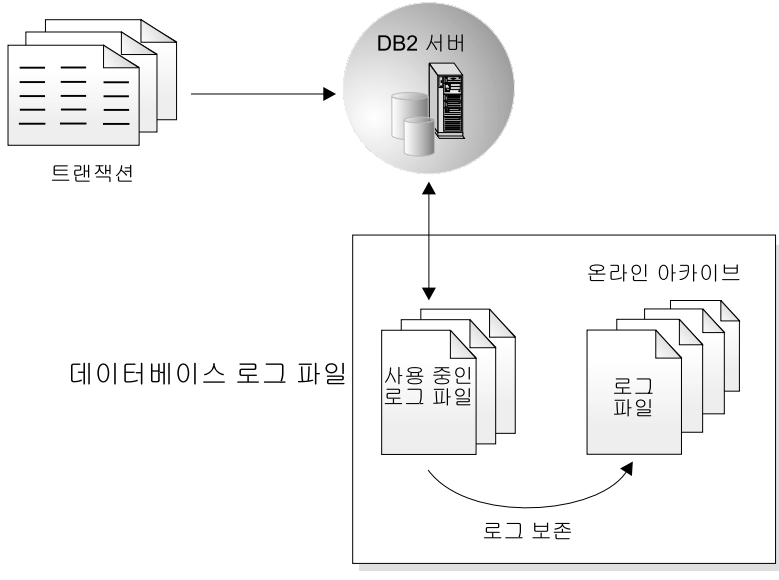


그림7. 온라인 아카이브 로그

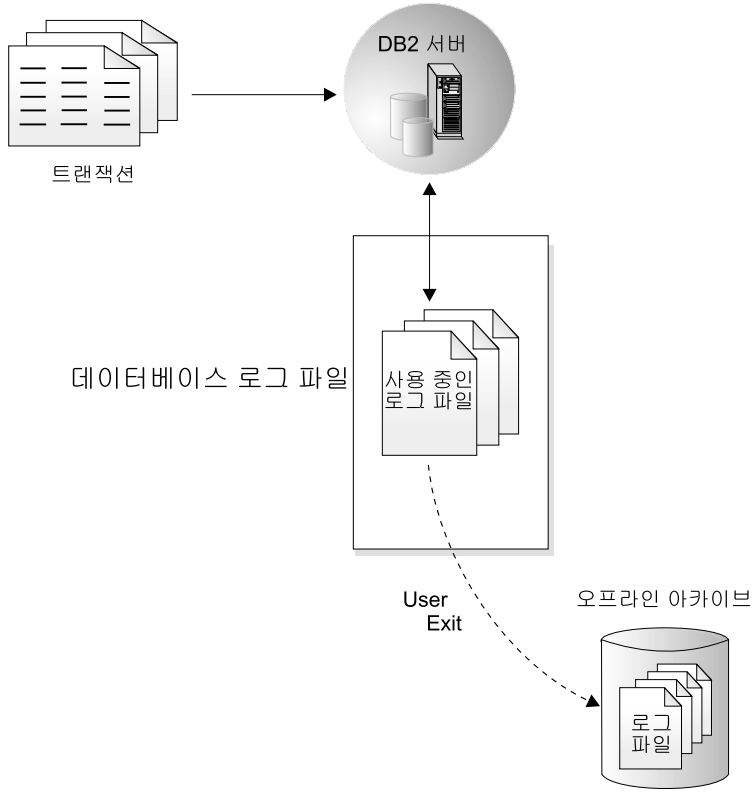


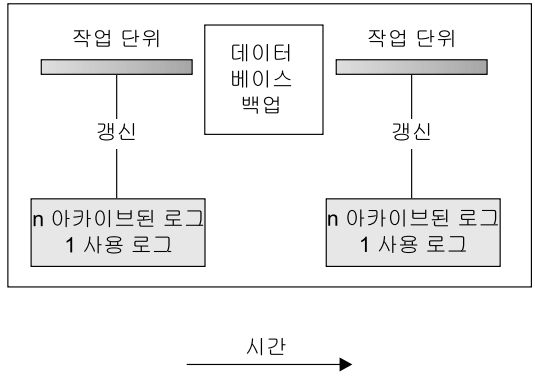
그림 8. 오프라인 아카이브 로그

롤 포워드 복구는 아카이브 로그와 사용 중인 로그를 모두 사용하여 데이터베이스를 로그 끝이나 특정 시점까지 재작성할 수 있습니다. 롤 포워드 유틸리티는 아카이브 로그 및 사용 중인 로그의 확약된 변경사항을 복원된 데이터베이스에 다시 적용하여 이 작업을 수행합니다.

롤 포워드 복구는 또한 로그를 사용하여 아카이브 로그 및 사용 중인 로그의 확약된 갱신사항을 재적용함으로써 테이블 공간을 재작성합니다. 테이블 공간을 로그 끝 또는 특정 시점에서 복구할 수 있습니다.

온라인 백업 조작 동안, 데이터베이스에 대해 모든 활동이 로그됩니다. 온라인 백업 이미지가 복원될 때, 최소한 백업 조작이 완료된 시점으로 로그가 롤 포워드되어야 합니다. 이렇게 하려면, 데이터베이스가 복원될 때 로그를 아카이브하여 사용 가능하도록 만들어야 합

니다. 온라인 백업이 완료되면, DB2는 현재 사용 중인 로그가 닫혀서 아카이브되도록 강제 실행합니다. 그러면 온라인 백업은 복구에 사용 가능한 완전한 아카이브 로그 세트를 갖게 됩니다.



로그는 변경사항을 데이터베이스로 추적하기 위해 백업 간에 사용됩니다.

그림 9. 롤 포워드 복구시 사용 중인 로그 또는 아카이브 데이터베이스 로그. 장기 수행 트랜잭션의 경우에는 사용 중 로그가 여러 개 있을 수 있습니다.

두 개의 데이터베이스 구성 매개변수(*newlogpath* 매개변수 및 *userexit* 매개변수)를 사용하면 아카이브 로그가 저장된 위치를 변경할 수 있습니다. *newlogpath* 매개변수를 변경하면 사용 중인 로그가 저장되는 곳에도 영향을 줍니다. 이들 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

데이터베이스 로그 경로 디렉토리에서 어떤 로그 *extents*가 로그를 아카이브했는지 판별하려면, *loghead* 데이터베이스 구성 매개변수의 값을 점검하십시오. 이 매개변수는 사용 중인 가장 낮은 번호의 로그를 나타냅니다. *loghead*보다 작은 순차 번호를 가진 로그들이 아카이브 로그이며 이동할 수 있습니다. 제어 센터를 사용하여 이 매개변수 값을 확인하거나, 명령행 처리기를 사용하고 GET DATABASE CONFIGURATION 명령을 사용하여 "첫 번째 사용 중인 로그 파일"을 볼 수 있습니다. 이 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

주:

1. 사용 중인 로그를 지우면 데이터베이스를 사용할 수 없게 되므로, 복원해야 다시 사용할 수 있습니다. 지워진 첫 번째 로그까지만 롤 포워드할 수 있습니다.

2. 사용 중인 로그가 손상될 수 있다는 점(디스크 파손으로 인해)이 염려되면, 운영 체제 레벨에서 로깅에 사용되는 디스크를 이중복사하거나, DB2 레벨에서 NEWLOGPATH2 레지스트리 변수를 사용한 이중복사를 고려해야 합니다.

로그 이중복사

DB2는 이제 데이터베이스 레벨에서의 로그 이중복사를 지원합니다. 로그 파일을 이중복사하면 다음으로부터 데이터베이스가 보호됩니다.

- 우연한 사용 중인 로그 삭제
- 하드웨어 장애로 인한 데이터 훼손

사용 중인 로그가 손상될 수 있다는 점(디스크 손상으로 인해)이 염려되면, 새로운 레지스트리 변수인 NEWLOGPATH2를 사용하여 사용 중인 로그의 사본을 관리할 데이터베이스에 대한 2차 경로를 지정하여, 로그가 저장되는 볼륨을 이중복사할 것을 고려해야 합니다.

NEWLOGPATH2 레지스트리 변수는 데이터베이스가 로그 파일의 동일한 두 번째 사본을 다른 경로에 기록할 수 있도록 합니다. 2차 로그 경로는 물리적으로 구분된 별도의 디스크(되도록 다른 디스크 제어기에도 있는 디스크)에 놓는 것이 좋습니다. 그러면 디스크 제어기가 단일 실패 지점이 될 수 없습니다.

주: Windows NT에서는 임시 경로 이름 아래에 장치를 『마운트』할 수 없으므로, 별도의 장치에 2차 경로를 지정할 수 없습니다(이 플랫폼에서는).

NEWLOGPATH2를 사용하거나(1로 설정) 사용하지 않을(0으로 설정) 수 있습니다. 기본값은 0입니다. 이 변수를 1로 설정하면, 2차 경로 이름은 문자 2로 연결된 LOGPATH 변수의 현재값입니다. 예를 들어, SMP 환경에서 LOGPATH가 `/u/dbuser/sqllogdir/logpath`이면, 2차 로그 경로는 `/u/dbuser/sqllogdir/logpath2`입니다. MPP 환경에서 LOGPATH가 `/u/dbuser/sqllogdir/logpath`이면, DB2는 노드 표시기를 경로에 추가하고 `/u/dbuser/sqllogdir/logpath/NODE0000`을 1차 로그 경로로 사용합니다. 이러한 경우, 2차 로그 경로는 `/u/dbuser/sqllogdir/logpath2/NODE0000`이 됩니다.

NEWLOGPATH2를 처음 사용할 경우, 이는 현재 로그 파일이 다음 데이터베이스 시동 시 완료될 때까지 실제로 사용되지 않습니다. 이는 LOGPATH 및 NEWLOGPATH가 현재 사용되는 방법과 유사합니다.

경로 1이나 경로 2에 쓰는 중 오류가 발생할 경우, 데이터베이스는 실패 경로를 『bad』로 표시하고 메시지를 db2diag.log 파일에 기록한 다음, 후속 로그 레코드를 나머지 『good』 로그 경로에만 기록합니다. DB2는 현재 로그 파일이 완료될 때까지 다시 『bad』 경로를 사용하려고 하지 않습니다. DB2가 다음 로그 파일을 열어야 할 경우, 이 경로가 유효한지 검증하고, 유효하면 이를 사용하기 시작합니다. 그렇지 않으면, DB2는 처음으로 다음 로그 파일에 액세스할 때까지 다시 그 경로를 사용하려고 하지 않습니다. 로그 경로를 동기화하려고 하지는 않지만, DB2는 발생하는 액세스 오류에 대한 정보를 보존하므로, 로그 파일을 아카이브할 때 올바른 경로를 사용할 수 있습니다. 나머지 『good』 경로에 기록하는 동안 실패할 경우, 데이터베이스가 이상종료합니다.

작업 테이블에서 로그 줄이기

응용프로그램이 마스터 테이블에서 작업 테이블을 작성하고 상주시키며, 해당 테이블이 마스터 테이블에서 쉽게 다시 작성될 수 있기 때문에 작업 테이블의 복구에 신경을 쓰지 않는 경우, CREATE TABLE문에서 NOT LOGGED INITIALLY 매개변수를 사용하여 작업 테이블을 작성할 수 있습니다. NOT LOGGED INITIALLY 매개변수를 사용하여 얻는 이점은 테이블이 작성된 작업 단위(UOW)의 테이블에서 이루어진 모든 변경사항(삽입, 삭제, 갱신 또는 색인 작성 조작을 포함한)이 기록되지 않는다는 것입니다. 실행된 기록을 감소시킬 뿐만 아니라 응용프로그램의 더 나은 성능을 가져올 수 있습니다. 또한 ALTER TABLE문을 NOT LOGGED INITIALLY 매개변수와 함께 사용하면 기존 테이블에 대해서도 동일한 동작을 할 수 있습니다. (이렇게 작동하려면, NOT LOGGED INITIALLY 옵션을 사용하여 테이블을 작성해야 합니다.)

주:

1. 같은 작업 단위(UOW)에서 NOT LOGGED INITIALLY 매개변수를 사용하여 둘 이상의 테이블을 작성할 수 있습니다.
2. 카탈로그 테이블 및 다른 사용자 테이블에 대한 변경사항은 여전히 기록됩니다.

테이블에 대한 변경사항은 로그되지 않으므로, NOT LOGGED INITIALLY 매개변수 사용 여부를 결정할 때 다음을 고려해야 합니다.

- 테이블에 대한 모든 변경사항은 확약시 디스크에 기록됩니다. 이로 인해 확약이 더 오래 걸릴 수 있습니다.
- 테이블이 작성된 작업 단위(UOW)의 모든 조작에 대해 리턴된 오류로 인해 전체 작업 단위(UOW)(SQLCODE -1476, SQLSTATE 40506)의 구간 복원이 발생합니다.
- 롤 포워드 중에 이 테이블을 복구할 수 없습니다. 롤 포워드 조작이 NOT LOGGED INITIALLY 옵션으로 작성된 테이블에서 일어나면, 테이블은 사용불가능으로 표시됩니다. 데이터베이스가 복구된 후에, 테이블에 액세스하려는 모든 시도는 SQL1477N을 리턴합니다.

  주: 테이블이 작성되는 경우, 행 잠금은 COMMIT가 완료될 때까지 카탈로그 테이블에 보유됩니다. 로그 작동이 되지 않게 하려면, 작성된 작업 단위(UOW)에서 테이블을 증가시켜야 합니다. 이 사항은 동시성을 포함하고 있습니다. 자세한 정보는 *관리 안내서: 성능* 책의 "동시성" 절을 참조하십시오.

테이블 작성에 대한 자세한 정보는 *SQL 참조서*에서 참조하십시오.

작업 테이블로서 선언된 임시 테이블을 사용하려는 경우, 다음을 기억하십시오.

- 선언된 임시 테이블은 카탈로그에서 작성되지 않으므로, 잠금이 유지되지 않습니다.
- 로그는 첫 번째 COMMIT 이후에 선언된 임시 테이블에 대해 수행되지 않습니다.
- ON COMMIT PRESERVE 옵션을 사용하여 COMMIT 이후에 테이블에 행을 유지하고, 그렇지 않으면 모든 행이 삭제됩니다.
- 선언된 임시 테이블을 작성한 응용프로그램만이 테이블의 인스턴스를 액세스할 수 있습니다.
- 데이터베이스에 대한 응용프로그램 연결이 삭제될 때 테이블이 내재적으로 삭제됩니다.

- 선언된 임시 테이블을 사용하는 작업 단위(UOW) 중 조작에서의 오류는 작업 단위(UOW)가 완전히 구간 복원되지 않게 합니다. 그러나 선언된 임시 테이블의 내용을 변경하는 명령문에서의 조작 오류는 해당 테이블의 모든 행을 삭제합니다. 작업 단위(UOW)(또는 저장 지점)의 구간 복원은 해당 작업 단위(UOW)(또는 저장 지점)에서 수정된 선언된 임시 테이블에서 모든 행을 삭제합니다.

선언된 임시 테이블 및 관련 제한사항에 대한 자세한 내용은 *SQL 참조서*에서 DECLARE GLOBAL TEMPORARY TABLE문을 참조하십시오.

데이터베이스 로깅을 위한 구성 매개변수

데이터베이스 구성 파일은 롤 포워드 복구와 관련된 매개변수를 가지고 있습니다. 매개변수 기본값은 이 복구 유형을 지원하지 않으므로, 이 복구를 사용할 계획이면 기본값 중 일부를 변경해야 합니다. DB2 Universal Database 구성에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

1차 로그(logprimary)

이 매개변수는 작성될 1차 로그의 수를 지정합니다.

비어 있든지 가득 차 있든지에 관계없이 1차 로그는 같은 크기의 디스크 공간이 필요합니다. 그러므로 필요한 수 이상의 로그를 구성하는 경우에는 디스크 공간을 불필요하게 사용하게 됩니다. 너무 적은 수의 로그를 구성하면 로그가 가득 찬 상태가 발생할 수 있습니다. 구성할 로그 수를 선택할 때 각 로그의 크기 및 응용프로그램이 로그 가득 참 상태를 처리할 수 있는지 여부를 고려해야 합니다.

기존 데이터베이스에 롤 포워드 복구를 사용 가능하게 하고자 할 경우, 1차 로그의 수를 1차 로그와 2차 로그의 수의 합계+1로 변경하십시오. 추가 정보는 롤 포워드 복구가 사용 가능한 데이터베이스의 LONG VARCHAR 및 LOB 필드에 로그됩니다.

총 로그 파일 크기 한계는 32GB입니다. 즉, 로그 파일 수(*logprimary* + *logsecond*)에 바이트 단위의 로그 파일 크기(*logfilsiz* * 4096)를 곱한 값이 32GB 미만이어야 합니다.

이 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

2차 로그(logsecond)

이 매개변수는 복구 로그 파일에 대해 작성되고 사용되는 2차 로그 파일의 수를 지정합니다(필요할 경우).

1차 로그 파일이 가득차게 되면, 2차 로그 파일(*logfilsiz*)이 필요할 때마다 한 번에 하나씩, 이 매개변수에 지정된 최대수까지 할당됩니다. 구성된 2차 로그 파일의 수보다 많은 파일이 필요할 경우, 오류가 리턴되고 데이터베이스에 대한 활동이 중지됩니다.

이 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

로그 크기(logfilsiz)

이 매개변수는 구성된 각 로그의 크기를 지정합니다(4KB 페이지 수 단위).

구성할 수 있는 총 사용 중 로그 공간은 논리적으로 32GB로 제한됩니다. 이 한계는 *logfilsiz*에 대한 상한(65535)과 (*logprimary* + *logsecond*)에 대한 상한(128)의 결과입니다. ((*logprimary* + *logsecond*) * *logfilsiz*) < (32GB / 4096)이 됩니다.

로그 파일의 크기는 성능과 직접 관련이 있습니다. 데이터베이스가 로그를 보유하도록 구성된 경우, 로그가 찰 때마다 새 로그의 할당과 초기화를 위한 요청이 발행됩니다. 로그의 크기를 늘리면, 새 로그 할당 및 초기화에 필요한 요청의 수가 줄어듭니다. (그러나 로그 크기가 커질수록 각각의 새로운 로그를 형식화하는데 더 많은 시간이 걸린다는 점에 유의하십시오.) 새로운 로그의 형식화는 형식화가 데이터베이스 성능에 영향을 주지 않도록, 데이터베이스에 연결된 응용프로그램에 대해 투명합니다.

데이터베이스를 여는 처리 시간을 최소화하기 위해 데이터베이스를 열린 상태로 유지하는 응용프로그램을 가지고 있다고 가정할 경우(63 페이지의 『복구 성능 향상』 참조), 로그 파일 크기는 오프라인에서 아카이브된 로그 사본을 만드는데 걸리는 시간으로 결정해야 합니다.

오프라인 아카이브 로그를 저장하기 위해 사용하는 장치의 데이터 전송 속도와 사본을 만드는데 사용되는 소프트웨어는 최소한 데이터베이스 관리 프로그램이 로그에 데이터를 기록하는 평균 속도와 일치해야 합니다. 전송 속도가 새 로그 데이터의 생성 속도를 따라잡지 못하고, 오랜 시간 동안

로깅 활동이 계속되는 경우, 빈 디스크 공간 크기에 의해 결정된 디스크 공간이 모두 소모될 수 있습니다. 이 상태가 발생하면, 데이터베이스 처리가 중단됩니다.

데이터 전송 속도는 테이프 또는 광학 미디어를 사용할 때 가장 중요합니다(로그 저장을 위해 서로 다른 미디어를 사용하는데 대한 정보는 487 페이지의 『부록H. 데이터베이스 복구를 위한 User Exit』를 참조하십시오.) 일부 테이프 장치는 파일의 크기에 상관없이 파일을 복사하는데 똑같은 시간이 걸립니다. 장치를 아카이브하는 기능을 결정해야 합니다.

테이프 장치에는 기타 고려사항이 있습니다. 아카이브 요청의 빈도가 중요합니다. 예를 들어, 어떤 복사 조작을 완료하는데 걸린 시간이 5분일 경우, 로그는 작업량이 최대치일 때 5분 동안의 로그 데이터를 수용할 만큼 충분히 커야 합니다. 테이프 장치는 하루당 조작의 수를 제한하는 설계상의 한계를 가질 수 있습니다. 이상의 요소가 로그 크기를 결정할 때 고려해야 할 사항입니다.

로그 파일 유실을 최소화하는 것도 로그 크기를 설정할 때 중요한 고려사항입니다. 아카이브는 전체 로그를 취합니다. 단일의 대형 로그를 사용할 경우, 아카이브 시간 간격을 증가시킵니다. 로그를 가진 미디어가 고장나면 일부 트랜잭션 정보가 유실될 수도 있습니다. 로그 크기를 줄이면 아카이브의 빈도는 늘어나지만, 미디어 고장인 경우, 유실된 로그 앞의 더 작은 로그를 사용할 수 있으므로 정보 유실량을 줄일 수 있습니다.

로그 버퍼(logbufsz)

이 매개변수로 로그 레코드를 디스크에 쓰기 전에 로그 레코드용 버퍼로서 사용할 데이터베이스 공유 메모리의 크기를 지정할 수 있습니다. 로그 레코드는 다음 이벤트 중 어느 하나가 발생할 때 디스크에 기록됩니다.

- 트랜잭션 확약
- 로그 버퍼가 가득 차게 됨
- 일부 기타 내부 데이터베이스 관리 프로그램 이벤트가 발생함

로그 버퍼 크기를 늘리면 로그 레코드가 디스크에 기록되는 횟수가 줄어들고 각 시간에 더 많은 레코드가 기록되기 때문에 로깅과 연관되는 입출력 활동이 더 효율적이 됩니다.

그룹 확약 수(mincommit)

이 매개변수로 최소수 만큼의 확약이 수행될 때까지 로그 레코드가 디스크에 기록되는 것을 지연시킬 수 있습니다. 이러한 지연으로 로그 레코드 기록과 관련된 데이터베이스 관리 프로그램의 오버헤드를 줄일 수 있고, 결과적으로 데이터베이스에 대해 수행중인 응용프로그램이 여러 개 있어 매우 짧은 시간 프레임 내에 응용프로그램이 요구하는 많은 확약을 처리해야 할 때 성능이 향상됩니다.

확약 그룹화는 이 매개변수의 값이 1 이상이고, 데이터베이스에 연결된 응용프로그램의 수가 이 매개변수 값보다 클 경우에만 발생합니다. 확약 그룹화가 적용되면, 응용프로그램 확약 요청은 1초 동안이나 확약 요청 수가 이 매개변수 값과 같아질 때까지 보류됩니다.

새 로그 경로(newlogpath)

데이터베이스 로그는 처음에는 데이터베이스 디렉토리의 서브디렉토리인 SQLOGDIR에 작성됩니다. 이 구성 매개변수의 값을 변경함으로써 사용중인 로그와 차후 아카이브 로그가 배치되는 위치를 다른 디렉토리 또는 장치로 변경할 수 있습니다. 데이터베이스가 롤 포워드 복구용으로 구성되어 있는 경우, 데이터베이스 로그 경로 디렉토리에 저장되는 보존 로그는 새 로그 경로 디렉토리로 이동되지 않습니다.

사용자가 로그 경로 디렉토리를 변경할 수 있기 때문에, 롤 포워드 복구에 필요한 로그는 다른 디렉토리 또는 다른 장치에 있을 수 있습니다. 여러 위치의 로그에 액세스를 허용하기 위해 롤 포워드 조작 중에 이 구성 매개변수의 값을 변경할 수 있습니다.

로그의 위치에 대한 기록을 가지고 있어야 합니다.

변경사항은 데이터베이스가 일치 상태가 될 때까지는 적용되지 않습니다. 구성 매개변수 *database_consistent*는 데이터베이스의 상태를 리턴합니다. 이 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오. 데이터베이스가 일치 상태가 아닐 경우, 데이터베이스 로그가 수행하는 역할에 대한 자세한 정보는 47 페이지의 『로그 파일 관리』를 참조하십시오.

로그 보유(logretain)

logretain을 RECOVERY로 설정하면, 아카이브된 로그는 데이터베이스 로그

경로 디렉토리에 보존되고 데이터베이스는 복구 가능한 것으로 간주됩니다. 이는 롤 포워드 복구를 사용할 수 있음을 의미합니다.

logretain을 CAPTURE로 설정하면, 복제 캡쳐 프로그램은 PRUNE LOGFILE 명령을 호출하여 캡쳐 프로그램이 완료될 때 로그 파일을 삭제합니다. 데이터베이스에서 롤 포워드 복구를 수행하려면 logretain을 CAPTURE로 설정해서는 안됩니다.

User Exit(userexit)

이 매개변수는 데이터베이스 관리 프로그램이 로그 아카이브 및 검색을 위해 User Exit 프로그램을 호출합니다. 로그 파일은 사용 중인 로그 경로와 다른 위치에서 아카이브됩니다. *userexit*가 ON으로 설정된 경우, 롤 포워드 복구를 사용할 수 있습니다. User Exit 프로그램에 대한 자세한 정보는 487 페이지의 『부록H. 데이터베이스 복구를 위한 User Exit』를 참조하십시오.

로그 파일 관리

데이터베이스 로그를 관리할 때 다음 사항을 고려하십시오.

- 아카이브된 로그용 번호 체계는 S0000000.LOG로 시작하여 S9999999.LOG까지이며, 최대 1000만 개의 로그 파일을 보유할 수 있습니다. 다음의 경우, 데이터베이스 관리 프로그램은 S0000000.LOG로 재설정됩니다.
  - 데이터베이스 구성 파일이 롤 포워드 복구가 사용 가능하도록 변경되었을 때
  - 데이터베이스 구성 파일이 롤 포워드 복구가 *사용 불가능*하도록 변경되었을 때
  - S9999999.LOG가 사용되었을 때

  DB2는 데이터베이스를 복원한 후 로그 이름을 재사용합니다(롤 포워드 복구와 무관). 데이터베이스 관리 프로그램은 롤 포워드 복구시 틀린 로그가 적용되지 않도록 하지만, 필요한 로그의 위치를 검출할 수 없습니다. 롤 포워드 복구에 정확한 로그를 사용할 수 있도록 해야 합니다.

  롤 포워드 조작이 완료되면 사용된 마지막 로그는 절단되고, 로깅은 다음 순서의 로그에서부터 시작됩니다. 롤 포워드 복구에 사용된 마지막 로그보다 순차 번호가 큰 로그 경로 디렉토리의 로그가 재사용됩니다. 절단 시점 이후에 절단 로그에 있는 항목에는 0이 겹쳐쓰기됩니다. 롤 포워드 유틸리티를 호출하기 전에

로그 사본을 만들도록 하십시오. (User Exit 프로그램을 호출하여 로그를 다른 위치로 복사할 수 있습니다.) User Exit 프로그램에 대한 자세한 정보는 487 페이지의 『부록H. 데이터베이스 복구를 위한 User Exit』를 참조하십시오.)

- 데이터베이스가 활성화되지 않았으면(ACTIVATE DATABASE 명령에 의해), DB2는 모든 응용프로그램이 데이터베이스에서 연결해제될 때 현재 로그 파일을 절단합니다. 그런 다음, 응용프로그램이 그 데이터베이스에 연결하면, DB2는 새 로그 파일에 로깅을 시작합니다. 시스템에서 여러 개의 작은 로그 파일을 생성할 경우에는 ACTIVATE DATABASE 명령을 사용할 것을 고려할 수 있습니다. 이렇게 하면, 응용프로그램이 연결할 때 데이터베이스를 초기화해야 하는 오버헤드가 줄어들 뿐만 아니라, 대형 로그 파일을 할당한 다음 새로운 대형 로그 파일을 할당해야 하는 오버헤드도 줄어듭니다.
- 로그 이름은 다시 사용되기 때문에, 아카이브된 로그가 데이터베이스에 대해 두 개 이상의 다른 *로그 순서*와 연관될 수 있습니다(49 페이지의 그림10 참조). 예를 들어, 백업 2를 복구하려면 사용할 수 있는 두 개의 가능한 로그 순서가 있어야 합니다. 전체 데이터베이스를 복구하는 동안 특정 시점까지를 롤 포워드하고 로그 끝에 이르기 전에 중지할 경우, 새로운 로그 순서를 작성하게 됩니다. 두 로그 순서는 결합될 수 없습니다. 첫 번째 로그 순서에 걸쳐 있는 온라인 백업 이미지가 있으면, 이 로그 순서를 사용하여 롤 포워드 복구를 완료해야 합니다.

  복구 후에 새로운 로그 순서를 작성한 경우, 기존 로그 순서의 모든 테이블 공간 백업 이미지는 유효하지 않습니다. 복원 유틸리티는 데이터베이스 복원 조작 바로 다음에 테이블 공간 복원 조작이 수행되므로 이전 로그 순서에서 테이블 공간 백업 이미지를 인식하지 못합니다. 데이터베이스가 실제로 롤 포워드될 때까지, 사용되는 로그 순서를 알 수 없습니다. 테이블 공간이 이전 로그 순서에 있을 경우, 테이블 공간 롤 포워드 조작에 의해 『파악』해야 합니다. 유효하지 않은 백업 이미지를 사용하는 복원 조작은 완료될 수는 있지만 테이블 공간 롤 포워드 조작이 실패하여 테이블 공간이 롤 포워드 보류 상태에 있게 됩니다.

  예를 들어, 테이블 공간 레벨의 백업 조작인 백업 3이 맨 위 로그 순서에서 S0000013.LOG와 S0000014.LOG 사이에 완료된다고 가정합니다(49 페이지의 그림10 참조). 데이터베이스 레벨의 백업 이미지인 백업 2를 사용하여 복원 및 롤 포워드할 경우, S0000012.LOG를 통해 롤 포워드해야 합니다. 이후로는 맨 위 로그 순서나 더 새로운 맨 아래 로그 순서를 통해 계속 롤 포워드할 수 있습니

다. 맨 아래 로그 순서를 통해 롤 포워드할 경우, 테이블 공간 레벨의 백업 이미지를 사용하여 테이블 공간 복원과 롤 포워드 복구를 수행할 수 없습니다.

테이블 공간 레벨의 백업 이미지인 백업 3을 사용하여 로그 끝까지 테이블 공간 롤 포워드 조작을 완료하려면, 데이터베이스 레벨의 백업 이미지인 백업 2를 복원한 다음 맨 위 로그 순서를 사용하여 완료해야 합니다. 테이블 공간 레벨의 백업 이미지인 백업 3이 복원되고 나면, 로그 끝까지 롤 포워드 조작을 초기화할 수 있습니다.

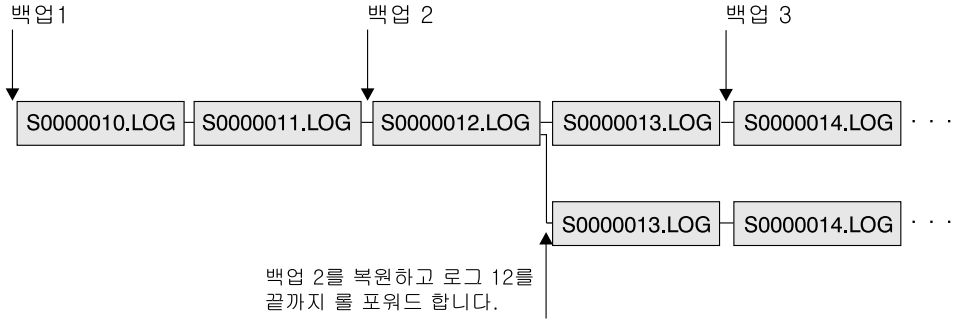


그림 10. 로그 파일 이름 재사용

User Exit 프로그램을 사용한 로그 파일 관리

다음 고려사항은 로그 파일을 아카이브하고 검색하기 위해 User Exit 프로그램을 호출할 때 적용됩니다.

- 데이터베이스 구성 파일 매개변수 *userexit*는 데이터베이스의 롤 포워드 복구 동안에 로그 파일을 아카이브하거나 로그 파일을 검색하기 위해 데이터베이스 관리 프로그램이 User Exit 프로그램을 호출하는지 여부를 지정합니다. 로그 파일을 검색하려는 요청은 롤 포워드 유틸리티가 로그 경로 디렉토리에 존재하지 않는 로그 파일을 필요로 할 때 만들어집니다.

  주: Windows NT에서, 로그를 아카이브하기 위해 REXX User Exit를 사용할 수 없습니다.

- 아카이브 작업시에 로그 파일이 아직 활동중이고 정상 처리에 필요하더라도 파일이 채워진 상태로 User Exit에 전달됩니다. 이것은 데이터의 사본이 가능한

한 빨리 휘발성 미디어로부터 이동되어 나오는 것을 허용합니다. User Exit에 넘겨진 로그 파일은 더이상 정상 처리에 필요하지 않게 될 때까지 로그 경로 디렉토리에 남겨져 있게 됩니다. 이 시점에서 디스크 공간은 재사용됩니다.

- DB2는 User Exit 프로그램을 시작하여 로그 파일을 시작할 때 읽기 모드에서 로그 파일을 엽니다. 일부 플랫폼에서 이것은 User Exit 프로그램이 로그 파일을 삭제할 수 없게 합니다. AIX와 같은 다른 플랫폼에서는 User Exit 프로그램과 같은 프로세스가 로그 파일을 삭제할 수 있습니다. User Exit 프로그램은 파일이 계속 사용 중 상태이며 응급 복구에 필요할 수 있으므로 아카이브한 후에 로그 파일을 삭제하지 않습니다. DB2는 로그 파일이 아카이브될 때 디스크 공간 재사용을 관리합니다.
- 로그 파일이 아카이브되었는데 그 파일이 비활동 상태인 경우, DB2는 파일을 삭제하지 않지만 이러한 파일이 필요할 때 다음 로그 파일로 이름을 변경합니다. 그러면 성능이 좋아지는데, 이는 새로운 로그 파일을 작성하면(파일 이름을 바꾸는 대신) 모든 페이지가 디스크 공간에 확실하게 기록되기 때문입니다. 디스크에서 필요한 페이지를 비운 후 다시 확보하는 것보다 재사용하는 것이 더 좋습니다.
- DB2는 손상 복구 또는 구간 복원 둘다에서 로그 파일을 검색하기 위해 User Exit 프로그램을 호출하지 *않습니다*.
- User Exit 프로그램은 롤 포워드 복구가 실패에 대한 것을 보증하지는 않지만 실패 가능성을 줄이기 위해 시도합니다. 로그 파일이 채워지면 User Exit 루틴을 위한 대기행렬에 있게 됩니다. 로그 파일이 채워지기 전에 로그를 포함하고 있는 디스크가 작업을 실패하게 되면, 로그 파일 안에 있는 데이터를 분실하게 됩니다. 또한, 파일은 아카이브 작업을 위해 대기행렬에서 대기하므로, 모든 파일들이 복사되기 전에 디스크가 실패하여 대기행렬에 있는 로그 파일이 유실될 수 있습니다.
- 각 개별 로그 파일의 구성된 크기는 User Exit에 직접 관련됩니다. 각 로그 파일이 매우 큰 경우, 아주 많은 양의 데이터가 디스크 작업 실패시 분실됩니다. 작은 로그 파일로 구성된 데이터베이스는 데이터가 User Exit에 더욱 자주 전달되도록 합니다.

  그러나 만약 사용자가 테이프와 같이 속도가 느린 장치로 데이터를 보낸다면, 사용자는 대기행렬이 최고로 올라가는 것을 방지하기 위해 더 큰 로그 파일을

갖기 원할 수도 있습니다. 대기행렬이 완전히 채워지게 되면, 아카이브 및 검색 요청은 처리되지 않습니다. 처리는 대기행렬에 들어갈 여유가 있을 때 다시 시작됩니다. 처리되지 않은 요청은 자동으로 다시 대기행렬에 넣어지지 않습니다.

- User Exit 프로그램에 대한 아카이브 요청은 *userexit*가 구성되고 사용 중인 로그 파일이 채워지는 각 시간마다 발생합니다. 데이터베이스에서의 연결을 마지막으로 해제하고, 더구나 User Exit 프로그램을 부분적으로 채워진 사용 로그 파일에 호출하면, 사용 중인 로그 파일이 채워지지 않을 수 있습니다.

  주: 사용되지 않은 로그 공간을 해제하려면 로그 파일을 아카이브하기 전에 절단하십시오.

- 로그 파일을 포함하는 장치에 미디어 실패가 발생한 경우, 로그 사본은 오프라인 로그 파일이 롤 포워드 복구에 의해 사용될 수 있도록 하기 위해 또 다른 장치에 만들어져야 합니다. 이 장치는 데이터베이스 데이터 파일을 포함하는 동일한 장치가 될 수 없습니다.
- 데이터베이스가 아카이브 요청에 대해 User Exit 프로그램으로부터 긍정적인 반응을 얻기 전에 폐쇄되어 버렸다면, 데이터베이스 관리 프로그램은 데이터베이스가 열릴 때 다른 요청을 보냅니다. 그래서 로그 파일은 두 번 이상 아카이브됩니다.
- User Exit 프로그램이 존재하지 않는 파일(다중 아카이브 요청이 있었기 때문이거나 첫 번째 아카이브 조작이 끝난 후에 파일이 삭제되었기 때문에)을 아카이브하는 요청을 받은 경우, 또는 존재하지 않는 파일(다른 디렉토리에 있거나 로그의 끝부분에 도달해 있기 때문에)을 검색하는 요청을 받은 경우에 User Exit 프로그램은 이 요청을 무시하고 제대로 실행되었다는 리턴 코드를 전달합니다.
- User Exit 프로그램은 각기 다른 로그 파일이 특정 시점 복구 후에 같은 이름으로 존재하는 것을 허용합니다. 그것은 두 개의 로그 파일을 아카이브하고, 정확한 복구 경로에 의해 로그 파일을 결합하도록 작성되어야 합니다(47 페이지의 『로그 파일 관리』 참조).
- 두 개 이상의 데이터베이스가 동시에 하나의 장치를 쓰고 있고 조작 중 하나는 롤 포워드 조작을 포함하고 있다면, 롤 포워드 복구를 위해 필요한 로그 파일은 현재 장치 내에 있는 미디어에 존재하지 않을 수도 있습니다. 다음의 두 상태가 발생할 수 있습니다.

- User Exit 프로그램이 데이터베이스 관리 프로그램에게 리턴 코드 0(성공)을 되돌리고 요청된 로그 파일이 검색되지 않으면, 데이터베이그 관리 프로그램은 롤 포워드 조작이 로그 파일의 끝까지 완전히 행해지고 롤 포워드 조작이 중단되었다고 판단합니다. 그러나 롤 포워드 처리는 로그 파일의 끝까지 도달하지 않을 수도 있습니다.
- 만약 0 이외의 리턴 코드가 리턴되면 데이터베이스 롤 포워드 보류 상태에 있게 되고, 사용자는 롤 포워드 처리를 다시 시작하거나 중단해야 합니다.

이 두 가지 상황의 발생을 방지하기 위해 사용자는 User Exit 프로그램을 호출하는 노드상의 다른 어떤 데이터베이스도 롤 포워드 조작중에는 열리지 않도록 하거나, 이러한 상황을 처리할 수 있도록 User Exit 프로그램을 작성할 수 있습니다.

로그 디렉토리가 가득 찬 경우의 트랜잭션 블로킹

새로운 DB2 레지스트리 변수인 DB2_BLOCK_ON_LOG_DISK_FULL을 설정하여, DB2가 사용 중인 로그 경로에서 새로운 로그 파일을 작성할 수 없을 때 ″디스크 가득 참″ 오류가 생성되지 않도록 할 수 있습니다.

그 대신, DB2는 성공할 때까지 5분마다 로그 파일을 작성하려고 합니다. *userexit* 매개변수를 ON으로 설정하여 데이터베이스를 구성한 경우, DB2는 로그 파일 아카이브 완료에 대해서도 확인합니다. 비활동 로그 파일이 아카이브되면, DB2는 비활동 로그 파일의 이름을 새 로그 파일 이름을 바꾼 후에 계속할 수 있습니다. 각각의 시도 다음에 DB2는 메시지를 db2diag.log 파일에 보냅니다. 로그 디스크가 가득 찬 상태 때문에 응용프로그램이 정지되는 것을 확인할 수 있는 유일한 방법은 db2diag.log 파일을 모니터하는 것입니다.

로그 파일이 작성될 때까지, 테이블 데이터를 갱신하려고 하는 사용자 응용프로그램은 트랜잭션을 확약할 수 없습니다. 읽기 전용 조회는 직접 영향을 받지 않을 수도 있지만, 조회에서 갱신 요청에 의해 잠긴 데이터나 갱신 응용프로그램에 의해 버퍼 풀에서 수정된 데이터 페이지에 액세스해야 할 경우, 읽기 전용 조회도 정지하는 것으로 나타납니다.

요구 시 로그 아카이브

DB2는 이제 언제든지 복구 가능한 데이터베이스에 대해 사용 중인 로그 닫기(User Exit 옵션이 사용될 경우, 아카이브)를 지원합니다. 이로써 알 수 있는 시점까지 안전한 로그 파일 세트를 수집한 다음에 이 로그 파일을 사용하여 대기 데이터베이스를 갱신할 수 있습니다.

ARCHIVE LOG 명령을 호출하거나 **db2ArchiveLog** API를 호출하여 요구 시 로그 아카이브를 초기화할 수 있습니다. 이에 대해서는 365 페이지의 『ARCHIVE LOG』 및 380 페이지의 『db2ArchiveLog - 아카이브 사용 중 로그 API』에 각각 설명되어 있습니다.

원시 로그 사용

데이터베이스 로그에 대해 원시 장치를 사용할 수 있습니다. 이를 수행하는 데에는 장점과 단점이 있습니다.

- 장점은 다음과 같습니다.
  - 27개 이상의 물리 드라이브를 시스템에 접속할 수 있습니다.
  - 파일 입출력 경로 길이가 짧아집니다. 이것은 시스템에서 성능을 개선시킬 수 있습니다. 작업 로드에 측정 가능한 이점이 있는지 여부를 평가할 벤치마크를 관리해야 합니다.
- 단점은 다음과 같습니다.
  - 장치는 다른 응용프로그램이 공유할 수 없으며, 전체 장치는 *반드시* DB2에 지정되어야 합니다.
  - 장치에서 백업하거나 복사하는 운영 체제 유틸리티 또는 제3의 도구로 장치를 조작할 수 없습니다.
  - 잘못된 물리 드라이브 번호를 지정하는 경우, 기존 드라이브에서 파일 시스템이 손쉽게 지워질 수 있습니다.

newlogpath 데이터베이스 구성 매개변수로 원시 로그를 구성할 수 있습니다. 원시 장치를 지정하는데 사용되는 구문의 예는 *관리 안내서: 구현* 책의 "원시 입출력" 절을 참조하십시오. 그러나 이를 수행하기 전에, 위에 나열된 장점과 단점, 아래 나열된 추가 고려사항을 고려하십시오.

- 하나의 장치만을 사용할 수 있습니다. 운영 체제 레벨에서 여러 디스크에 장치를 정의할 수 있습니다. DB2는 운영 체제 호출을 수행하여 4KB 페이지의 장치 *크기*를 판별합니다.

  여러 디스크를 사용하는 경우, 보다 큰 장치가 제공되고 결과로 인한 스트라이핑은 입출력 처리 속도를 높임으로써 성능을 향상시킵니다.

- DB2는 장치의 마지막 4KB 페이지에 기록하려고 시도합니다. 장치 *크기*가 2GB를 넘는 경우, 2GB를 넘는 장치를 지원하지 않는 운영 체제의 마지막 페이지에 기록하려는 시도는 실패하게 됩니다. 이 경우, DB2는 지원되는 모든 페이지를 최대한 사용하려고 시도하게 됩니다.

  장치 *크기*에 대한 정보는 운영 체제를 지원하는 DB2가 사용할 수 있는 장치 (4KB 페이지) *크기*를 지정하는데 사용됩니다. DB2가 기록할 수 있는 디스크 공간량을 *device-size-available*이라 합니다.

  DB2는 장치의 처음 4KB 페이지를 사용하지 않습니다. (일반적으로, 이 공간은 운영 체제가 다른 용도로 사용합니다.) 즉, DB2가 사용할 수 있는 총 공간은 *device-size = device-size-available - 1*입니다.

- *logsecond* 매개변수는 사용되지 않습니다. DB2는 2차 로그를 할당하지 않습니다. 사용 중인 로그 공간의 *크기*는 *logprimary* x *logfilsiz*를 통해 산출되는 4KB 페이지 수입니다.

- 로그 레코드는 각 4KB의 로그 파일 *크기*(*logfilsiz*)를 갖는 로그 extent로 그룹화됩니다. 각 extent는 원시 장치에 하나씩 배치됩니다. 또한 각 extent는 extent 헤더용의 추가적 두 페이지로 구성됩니다. 즉, 장치가 지원할 수 있는 *사용 가능한 로그 extent 수*는 *device-size* / (*logfilsiz*+ 2)입니다.

- 장치는 사용 중인 로그 공간을 지원할 만큼 충분한 *크기*여야 합니다. 즉, *사용 가능한 로그 extent 수*는 *logprimary* 구성 매개변수에 대해 지정된 값 이상이어야 합니다.

- 순환 로그를 사용하는 경우, *logprimary* 구성 매개변수는 장치에 기록된 로그 extent 수를 결정합니다. 그 결과, 장치에 미사용 공간이 생기게 됩니다.

- User Exit 프로그램 없이 로그 보유(*logretain*)를 사용하는 경우, *사용 가능한 로그 extent 수* 만큼 모두 사용하고 나면 갱신으로 인한 모든 조작은 로그 가득 참 오류를 수신하게 됩니다. 이 경우, 데이터베이스를 종료한 다음 오프라인 백업하여 복구할 수 있도록 해야 합니다. 데이터베이스 백업 조작이 수행되고

나면, 장치에 기록된 로그 레코드는 유실됩니다. 즉, 초기 데이터베이스 백업 이미지를 사용하여 데이터베이스를 복원한 다음 롤 포워드할 수 없습니다. *사용 가능한 로그 extent 수*까지 모두 사용하기 전에 데이터베이스 백업을 수행하면, 데이터베이스를 복원 및 롤 포워드할 수 있습니다.

- User Exit 프로그램과 함께 로그 보유(*logretain*)를 사용하는 경우, 각 로그가 로그 레코드로 가득 찰 때 각 로그 extent에 대해 User Exit 프로그램이 호출됩니다. User Exit 프로그램은 장치를 읽고 보존 로그를 파일로 저장할 수 있어야 합니다. DB2는 로그 파일을 원시 장치로 검색하기 위해 User Exit 프로그램을 호출하지 않습니다. 대신, 롤 포워드 복구 동안 DB2는 extent 헤더를 읽어 원시 장치에 사용될 필수 로그 파일이 들어 있는지 여부를 판별합니다. 원시 장치에 필수 로그 파일이 없는 경우, DB2는 오버플로우 로그 경로를 검색합니다. 여전히 로그 파일이 없는 경우, DB2는 User Exit 프로그램을 호출하여 오버플로우 로그 경로로 로그 파일을 검색합니다. 롤 포워드 조작에 대해 오버플로우 로그 경로를 지정하지 않은 경우, DB2는 로그 파일을 검색하기 위해 User Exit 프로그램을 호출하지 않습니다. User Exit 프로그램 호출에 대한 추가 정보는 490 페이지의 『호출 형식』을 참조하십시오.
- DPROP를 사용하고 있으며 로그를 원시 장치에 기록하는 경우, 읽기 로그 API는 로그 파일을 검색하기 위해 User Exit 프로그램을 호출하지 않습니다. 그러나 요청된 로그 레코드를 장치에서 사용할 수 있는 경우 여전히 이 레코드가 리턴됩니다. 장치에서 가장 오래된 로그보다 이전의 로그를 요청하는 경우, 해당 로그는 리턴되지 않습니다. (이 작동은 DB가 요청된 로그 레코드가 들어 있는 로그 파일을 찾을 수 없는 경우와 유사합니다.)

  주:

  1. 원시 장치를 로그에 사용하는 경우, DPROP을 사용하지 않는 것이 좋습니다.
  2. **sqlurlog** API를 사용하는 경우, 원시 장치를 로그에 사용할 수 없습니다.

로그 유실

데이터베이스를 삭제하면 현재 데이터베이스 로그 경로 디렉토리의 모든 로그가 지워집니다. 데이터베이스를 삭제하기 전에 로그 사본 작성을 고려하십시오.

특정 시점으로 데이터베이스를 롤 포워드하는 중이면, 사용된 마지막 로그와 그 다음에 오는 기존의 모든 로그가 다시 사용됩니다. 그러므로 특정 시점 복구 조작을 시작하기 *전에* 현재 데이터베이스 로그 경로 디렉토리의 모든 로그를 복사해야 합니다.

롤 포워드 조작이 완료될 때, 마지막으로 확약된 트랜잭션이 있는 로그 파일이 절단되고 그 다음 순서의 로그부터 기록이 시작됩니다. 절단되기 전에 로그의 사본과 순차 번호가 높은 로그 사본을 작성하지 않으면, 지정된 특정 시점을 지나간 데이터베이스는 복구할 수 없습니다. (일단 정상적인 데이터베이스 활동이 재개되면, 모든 후속 복구 조작에 사용될 수 있는 새로운 로그가 작성됩니다.)

로그 경로 디렉토리를 변경한 후 서브디렉토리를 제거하거나 로그 경로에서 호출한 서브디렉토리의 모든 로그를 지우면, 데이터베이스 관리 프로그램은 데이터베이스가 열릴 때 기본 로그 경로 SQLOGDIR에서 로그를 찾습니다. 로그가 없을 경우, 데이터베이스는 백업 보류 상태에 놓이므로 먼저 데이터베이스를 백업해야 데이터베이스를 사용할 수 있습니다. 이 백업은 서브디렉토리에 빈 로그가 들어 있더라도 수행되어야 합니다.

온라인 백업 끝의 특정 시점을 포함하고 있는 로그를 유실하고 복원된 해당 이미지를 롤 포워드하고 있는 경우, 데이터베이스를 사용할 수 없게 됩니다. 데이터베이스를 사용할 수 있으려면, 다른 백업 및 모든 연관된 로그로부터 데이터베이스를 복원해야 합니다.

다음과 비슷한 상황이 발생할 수도 있습니다. 전체 데이터베이스에 대해 특정 시점 복구를 수행하려 하지만 복구 프로세스 동안 로그를 유실할 수도 있음을 고려합니다. (이 시나리오는 최종 백업 데이터베이스 이미지의 시간과 데이터베이스를 복구하려는 시점 사이에 확장된 수의 아카이브 로그가 있는 경우 발생할 수 있습니다.)

우선, 적용 가능한 모든 로그를 『안전한』 위치에 복사해야 합니다. 그런 다음, RESTORE 명령을 수행하여 데이터베이스에 대해 원하는 특정 시점으로 롤 포워드 복구 방법을 사용할 수 있습니다. 필요로 하는 로그 중에서 이 프로세스 동안 손상되거나 유실되는 것이 있으면, 사용자는 모든 로그의 백업 사본을 다른 위치에 보관합니다.

복구 실행기록 파일 이해

복구 실행기록 파일은 각 데이터베이스마다 작성되며, 다음의 경우에 자동으로 갱신됩니다.

- 데이터베이스 또는 테이블 공간이 백업되었을 때
- 데이터베이스 또는 테이블 공간이 복원되었을 때
- 데이터베이스 또는 테이블 공간이 롤 포워드되었을 때
- 테이블 공간이 작성되었을 때
- 테이블 공간이 변경되었을 때
- 테이블 공간이 quiesce되었을 때
- 테이블 공간 이름을 바꾸었을 때
- 테이블 공간이 제거되었을 때
- 테이블이 로드되었을 때
- 테이블이 제거되었을 때
- 테이블이 재구성되었을 때
- 테이블 통계가 갱신되었을 때

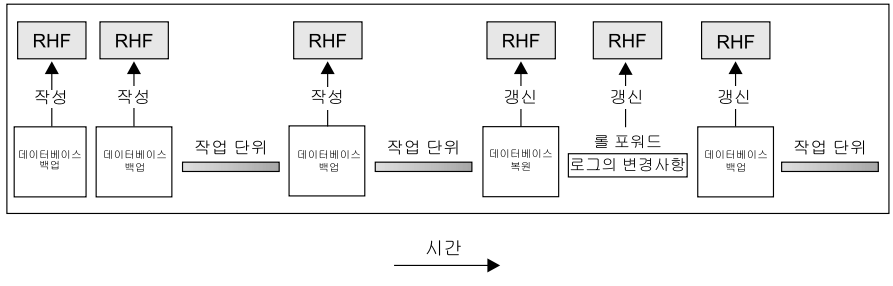


그림 11. 복구 실행기록 파일 작성 및 갱신

이 파일에서 요약된 백업 정보를 사용하여 주어진 시간 안에 데이터베이스의 전체 또는 부분을 복구할 수 있습니다. 이 파일의 정보에는 다음이 포함됩니다.

- 각 항목을 고유하게 식별하는 식별(ID) 필드
- 복사된 데이터베이스의 부분과 방법

- 복사된 시간
- 사본의 위치(장치 정보 및 사본에 액세스하는 논리적인 방법을 명시)
- 복원 조작이 수행된 최종 시간
- 테이블 공간의 이름이 바뀐 시간. 테이블 공간의 이전 및 현재 이름을 표시합니다.
- 백업 조작의 상태: 사용 중, 미사용 중, 만기됨 또는 삭제됨
- 데이터베이스 백업으로 저장되거나 롤 포워드 복구 조작 동안 처리된 마지막 로그 순차 번호

복구 실행기록 파일에서 항목을 보려면, LIST HISTORY 명령을 사용하십시오. 이 명령에 대한 자세한 정보는 370 페이지의 『LIST HISTORY』를 참조하십시오.

주: 복원 후 롤 포워드 조작이 로그 끝까지 수행된 경우, LIST HISTORY 명령의 다음 호출을 표시한 백업 ID는 시간의 끝을 나타냅니다. 즉, 백업 ID 값은 99991231235959 입니다. 백업 ID는 롤 포워드 조작이 수행될 때에만 이와 같이 변환됩니다.

모든 백업 조작(데이터베이스, 테이블 공간 또는 증분)에는 복구 실행기록 파일의 사본이 포함됩니다. 복구 실행기록 파일은 데이터베이스에 링크됩니다. 데이터베이스를 삭제하면 복구 실행기록 파일이 삭제됩니다. 데이터베이스를 새로운 위치로 복원하면, 복구 실행기록 파일이 복원됩니다. 복원은 기존의 실행기록 복구 파일을 겹쳐쓰지 않습니다.

현재 데이터베이스를 사용할 수 없거나 사용하지 않고, 연관된 복구 실행 기록 파일이 손상되거나 삭제되면 RESTORE 명령에 대한 옵션은 복구 실행기록 파일만 복원되도록 합니다. 그러면 복구 실행기록 파일은 데이터베이스를 복원하는데 사용하는 백업의 정보를 제공하여 검토될 수 있습니다.

파일 크기는 파일에 있는 항목에 대한 보유 기간(일 단위)을 지정하는 *rec_his_retentn* 구성 매개변수에 의해 제어됩니다. 이 매개변수의 수가 0으로 설정되더라도 맨 나중의 전체 데이터베이스 백업(자체 복원 세트 포함) 유지됩니다. (이 사본을 제거하는 유일한 방법은 PRUNE을 FORCE 옵션과 함께 사용하는 것입니다.) 보유 기간의 기본값은 366일입니다. 이 기간은 -1을 사용함으로써 무한정의 일수로 설정될 수 있습니다. 이 경우, 파일의 명시적 제거가 요구됩니다. 이 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

가비지 콜렉션

언제든지 PRUNE HISTORY 명령을 사용하여(373 페이지의 『PRUNE HISTORY/LOGFILE』 참조) 실행기록 파일에서 항목을 제거할 수 있지만, 그러

한 제거 작업은 DB2에 맡겨두는 것이 좋습니다. 복구 실행기록 파일에 기록된 DB2 데이터베이스 백업 수는 DB2 *가비지 콜렉션*에 의해 자동으로 모니터됩니다. DB2 가비지 콜렉션은 데이터베이스 백업 조작이 완료된 후에 호출됩니다. 또한, 데이터베이스 복원 조작이 완료된 후에도 호출됩니다. 구성 매개변수 *num_db_backups*는 보존되는 사용 중인 전체(증분이 아님) 데이터베이스 백업 이미지 수를 정의합니다 이 매개변수 값은 마지막 항목에서부터 시작하여 실행기록 파일을 스캔하기 위해 사용됩니다.

모든 데이터베이스 백업 조작 후, *rec_his_retentn* 구성 매개변수는 실행기록 파일에서 만기된 항목을 제거하기 위해 사용됩니다.

*사용 중인 데이터베이스 백업*은 데이터베이스의 현재 상태를 복구하기 위해 현재 로그를 사용하여 복원 및 롤 포워드될 수 있습니다. *사용 중이 아닌 데이터베이스 백업*은 복원된 경우에 데이터베이스를 이전 상태로 이동시키는 데이터베이스 백업입니다.

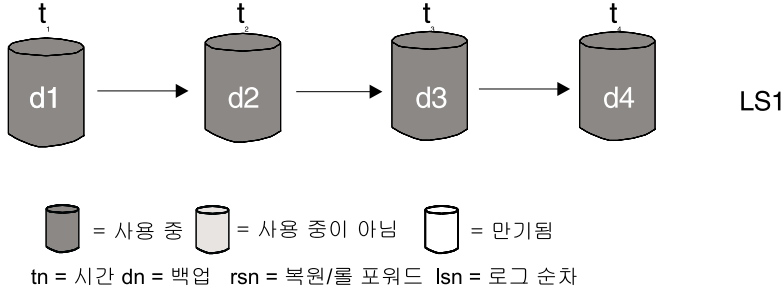


*그림 12. 사용 중인 데이터베이스 백업. num_db_backups*의 값이 4로 설정되었습니다.

더이상 필요하지 않은 모든 사용 중인 데이터베이스 백업 이미지는 『만기됨』으로 표시됩니다. 이러한 이미지는 더 최근의 백업 이미지가 사용 가능하므로, 불필요한 것으로 간주됩니다. 데이터베이스 백업 이미지가 만기되기 전에 수행된 모든 테이블 공간 백업 이미지 및 로드 백업 사본은 『만기됨』으로 표시됩니다.

『사용 중이 아님』으로 표시되고 만기된 데이터베이스 백업이 취해진 시점 이전에 취해진 모든 데이터베이스 백업도 『만기됨』으로 표시됩니다. 연관된 모든 미사용 테이블 공간 백업 이미지 및 로드 백업 사본도 『만기됨』으로 표시됩니다.

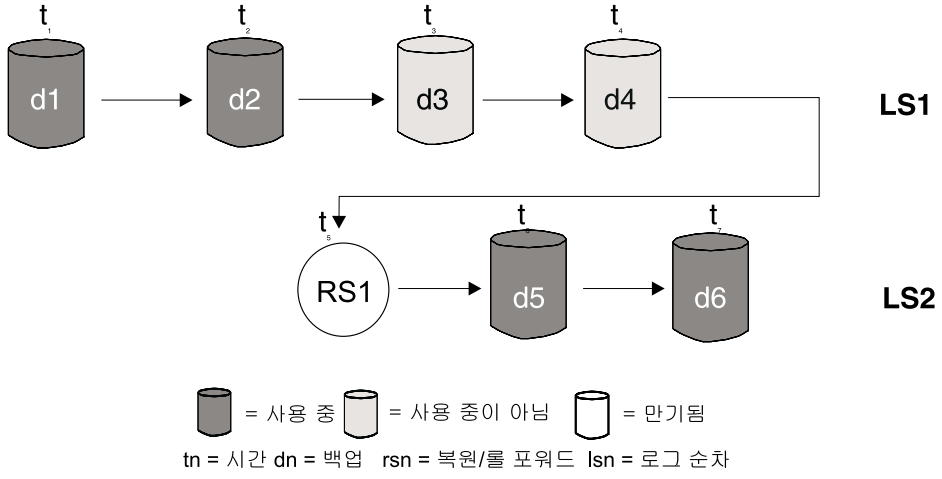


그림 13. 미사용 데이터베이스 백업

사용 중인 데이터베이스 백업 이미지가 복원되었지만 실행기록 파일에 기록된 가장 최근의 데이터베이스 백업 이미지가 아닌 경우, 같은 로그 순서에 속하는 후속 데이터베이스 백업은 『미사용』으로 표시됩니다.

미사용 데이터베이스 백업 이미지가 복원되면, 현재 로그 순서에 속하는 모든 미사용 데이터베이스 백업은 다시 『사용 중』으로 표시됩니다. 더이상 현재 로그 순서에 있지 않은 모든 사용 중인 데이터베이스 백업 이미지는 『사용 중이 아님』으로 표시됩니다.

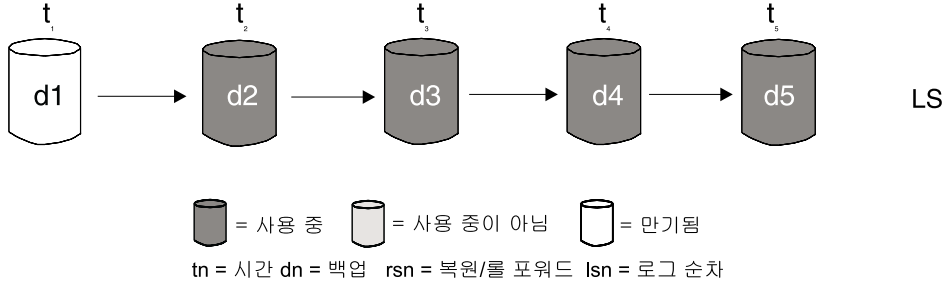


그림 14. 만기된 데이터베이스 백업

DB2 가비지 콜렉션도 해당 백업이 현재 로그 *체인*이라고도 하는 하는 로그 *순서*에 해당되지 않으면 DB2 데이터베이스 또는 테이블 공간 백업 이미지에 대한 실행기록 파일 항목을 『사용 중이 아님』으로 표시해야 합니다. 현재 로그 순서는 복원된 DB2 데이터베이스 백업 이미지와 처리된 로그 파일에 의해 결정됩니다. 일단 데이터베이스 백업 이미지가

복원되면, 복원된 이미지가 새로운 로그 체인을 시작하므로 모든 후속 데이터베이스 백업 이미지는『사용 중이 아님』이 됩니다. (이는 백업 이미지가 롤 포워드 없이 복원된 경우에 해당됩니다.) 롤 포워드 조작이 발생하였으면, 로그 체인에서 중단 후에 취해진 모든 데이터베이스 백업이『사용 중이 아님』으로 표시됩니다. 롤 포워드 유틸리티는 손상된 현재 백업 이미지를 포함하는 로그 순서를 통해 진행되었으므로, 더 오래된 데이터베이스 백업 이미지를 복원해야 합니다.

테이블 공간 레벨의 백업 이미지를 복원한 후에 현재 로그 순서를 적용하여 데이터베이스의 현재 상태에 도달할 수 없는 경우, 그 백업 이미지는『사용 중이 아님』이 됩니다.

백업 이미지에 DATALINK 컬럼이 있으면, DB2 Data Links Manager를 수행 중인 모든 데이터 링크 서버가 가비지 콜렉션을 요청하기 위해 접속합니다. 그러면 DB2 가비지 콜렉션은 만기된 백업이 포함되었지만 다음 데이터베이스 백업 조작 이전에 링크 해제된 연관되는 데이터 링크 서버 파일의 백업을 삭제합니다.

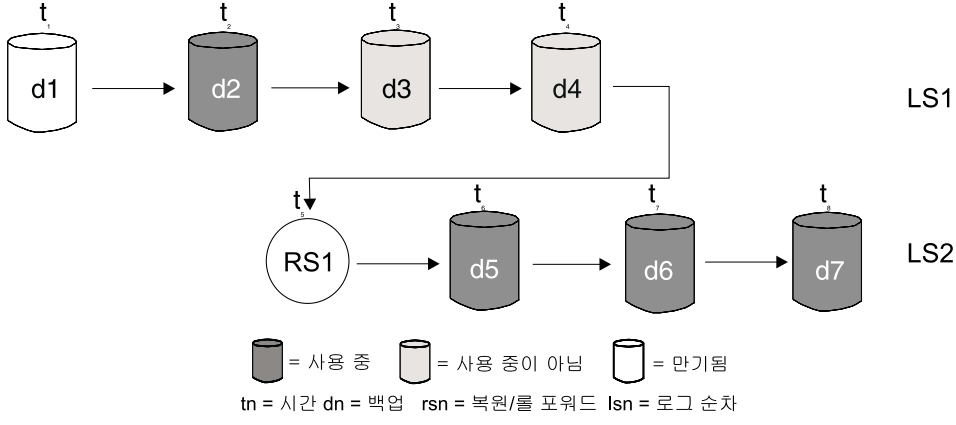


그림 15. 사용 중, 사용 중이 아님 및 만기된 데이터베이스 백업 혼합

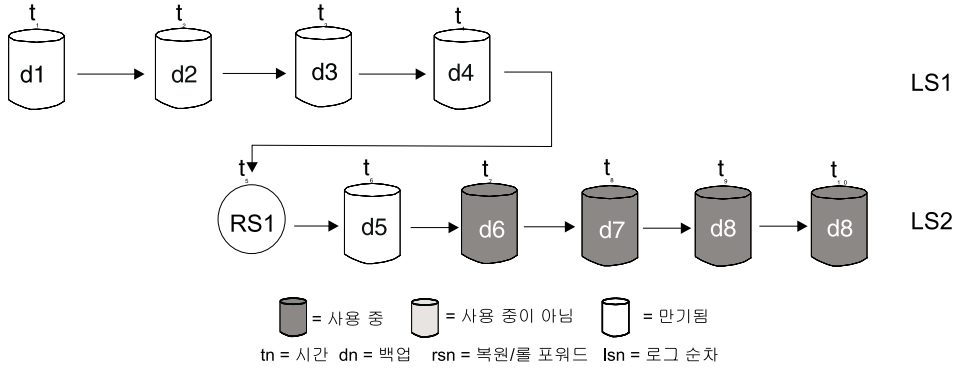


그림 16. 만기된 로그 순서

테이블 공간 상태 이해

테이블 공간의 현재 상태는 해당 *상태*별로 반영됩니다. 복구와 가장 일반적으로 연관되는 테이블 공간 상태는 다음과 같습니다.

- *롤 포워드 보류*. 테이블 공간은 복원된 후나 입출력 오류 후에 이 상태가 됩니다. 테이블 공간이 복원되고 나면, 로그 끝이나 특정 시점까지 테이블 공간을 롤 포워드할 수 있습니다. 입출력 오류 다음에는 로그 끝까지 테이블 공간을 롤 포워드해야 합니다.
- *롤 포워드 진행 중*. 테이블 공간에서의 롤 포워드 조작이 진행 중일 때 그 테이블 공간은 이 상태가 됩니다. 롤 포워드 조작이 완료되면, 테이블 공간은 더이상 롤 포워드 진행 중 상태에 있지 않습니다. 롤 포워드 조작이 취소되면 테이블 공간도 이 상태에서 벗어날 수 있습니다.
- *복원 보류*. 테이블 공간에서의 롤 포워드 조작이 취소되거나 그 조작에서 복구할 수 없는 오류가 발생한 경우(이 경우에는 테이블 공간을 복원하고 다시 롤 포워드해야 함), 그 테이블 공간은 이 상태가 됩니다.
- *백업 보류*. 테이블 공간은 특정 시점 롤 포워드 조작 후나 복사 옵션을 사용하지 않은 로드 조작 후에 이 상태가 됩니다. 테이블 공간을 사용하기 전에 백업해야 합니다. (백업되지 않은 경우, 테이블 공간을 갱신할 수 없지만 읽기 조작만은 허용됩니다.)

복구 성능 향상

다음 항목은 복구 성능에 대해 고려할 사항입니다.

- 로그를 별도의 장치에 배치함으로써, 자주 갱신되는 데이터베이스의 성능을 향상키실 수 있습니다. 온라인 트랜잭션 처리(OLTP) 환경에서는 데이터 행 저장 시보다 데이터를 기록하는데 더 많은 입출력이 필요합니다. 로그를 별도의 장치에 두면, 로그와 데이터베이스 파일 사이를 이동시키는데 필요한 디스크 암(arm) 이동이 최소화됩니다.

  디스크에 있는 다른 파일도 고려해야 합니다. 예를 들어, 실제 메모리가 부족한 시스템에서 시스템 페이징에 사용되는 디스크로 로그를 옮기면, 사용자의 조정 노력이 무산됩니다.

- 복원 조작을 완료하는데 필요한 시간을 줄이려면, 다음을 수행하십시오.
  - 복원 버퍼 *크기*를 조정하십시오. 버퍼 *크기*는 백업 조작중에 사용된 복수의 버퍼 *크기*여야 합니다.
  - 버퍼의 수를 *증가*시키십시오.

    다중 버퍼 및 입출력 채널을 사용하려면, 적어도 채널 수의 두 배의 버퍼를 사용하여 채널이 데이터를 대기할 필요가 없음을 확인해야 합니다. 사용된 버퍼의 *크기* 또한 복원 조작의 성능에 도움이 됩니다. 이상적인 복원 버퍼 *크기*는 테이블 공간의 다중 확장 *크기*가 되어야 합니다.

    다른 확장 *크기*를 가진 다중 테이블 공간이 있으면, 최대 확장 *크기*의 배수인 값을 지정하십시오.

    버퍼의 *최*소 권장 수는 PARALLELISM 옵션용으로 지정된 수를 미디어 장치 또는 컨테이너 수에 합한 것입니다.
  - 여러 소스 장치를 사용하십시오.
  - 복원 조작의 PARALLELISM 옵션을 소스 장치 수보다 1 이상 더 큰 수로 설정하십시오.

- 테이블에 대량의 long 필드와 LOB 데이터가 들어 있으면 데이터베이스를 복원하는데 시간이 많이 걸릴 수 있습니다. 데이터베이스에서 롤 포워드 복구가 작동 가능하면, RESTORE 명령으로 선택된 테이블 공간을 복원할 수 있습니다. long 필드와 LOB 데이터가 사용자의 비지니스에 중요하면, 이러한 테이블 공간에 대한 백업 타스크 완료시 필요한 시간에 대해 해당 테이블 공간을 복원하

는 것을 고려해야 합니다. long 필드와 LOB 데이터를 별도의 테이블 공간에 저장하면, 복원 조작을 완료하는데 필요한 시간은 long 필드와 LOB 데이터를 포함하는 테이블 공간을 복원하지 않도록 선택함으로써 줄일 수 있습니다. 별도의 소스에서 LOB 데이터를 재생산할 수 있을 경우, 테이블이 LOB 컬럼을 포함하도록 테이블을 작성하거나 변경할 때에는 NOT LOGGED 옵션을 선택하십시오. long 필드 및 LOB 데이터가 들어 있는 테이블 공간을 복원하지 않고 테이블이 들어 있는 테이블 공간을 복원해야 하는 경우, 로그 끝까지 롤 포워드하여 테이블 데이터가 들어 있는 모든 테이블 공간이 일관성을 갖도록 해야 합니다.

주: long 또는 LOB 필드 없이 테이블 데이터를 포함한 테이블 공간을 백업하는 경우, 해당 테이블 공간에 적절한 롤 포워드 복구를 실행할 수 없습니다. 테이블에 대한 모든 테이블 공간은 특정 시점으로 동시에 롤 포워드되어야 합니다.

- 다음은 백업 및 복원 조작 모두에 적용됩니다.
  - 다중 입출력 버퍼 및 장치를 사용합니다.
  - 사용 중인 장치 수보다 적어도 두 배 만큼의 버퍼를 할당하도록 합니다.
  - 입출력 장치 제어기 대역폭을 초과하지 않도록 합니다.
  - 몇몇 대형 버퍼를 사용하기 보다는 작은 *크기*의 버퍼를 여러 개 사용하십시오.
  - 시스템 자원에 따라 버퍼의 수와 *크기*를 조정하십시오.

병렬 복구

DB2는 이제 여러 에이전트를 사용하여 응급 복구와 데이터베이스 롤 포워드 복구를 모두 수행합니다. 이러한 조작 중에, 특히 SMP(Symmetric Multi-Processor) 머신에서는 더 나은 성능을 기대할 수 있습니다. 데이터베이스 복구 중에 여러 에이전트를 사용하면 SMP 머신에서 사용 가능한 여분의 CPU를 이용하게 됩니다.

이러한 개선에서 도입된 새로운 에이전트 유형이 db2agnsc입니다. DB2는 머신의 CPU 수를 기초로 데이터베이스 복구에 사용할 에이전트 수를 선택합니다. SMP 머신의 경우, 사용되는 에이전트 수는 (CPU 수 + 1)입니다. 단일 CPU를 사용하

는 머신에서는, 더 효율적인 로그 읽기, 로그 레코드 처리, 데이터 페이지의 프리페치를 위해 여러 에이전트가 사용됩니다.

DB2는 현재 재적용할 수 있도록 로그 레코드들을 에이전트에 분배합니다(적절할 경우). 예를 들어, 삽입, 삭제, 갱신, 키 추가, 키 삭제 조작과 연관되는 로그 레코드의 처리를 이러한 방식으로 병렬 처리할 수 있습니다. 로그 레코드는 페이지 레벨에서 병렬 처리되므로(같은 데이터 페이지의 로그 레코드들은 같은 에이전트에 의해 처리됨), 모든 작업이 하나의 테이블에서 수행되었어도 성능이 개선됩니다.

DB2 Data Links Manager 고려사항

다음 절에서는 DATALINK 컬럼이 들어 있는 테이블이 있는 경우 적용할 정보를 제공합니다. DATALINK 컬럼에 대한 전체 설명은 *SQL 참조서*에 있는 CREATE TABLE문을 참조하십시오.

시스템 손상 복구 고려사항

응용프로그램이 데이터베이스 관리 프로그램을 수행하는 Data Links 서버를 포함하여 SQL 요청을 발행하면(FILE LINK CONTROL 속성이 있는 DATALINK 컬럼을 사용하여), 데이터베이스 관리 프로그램은 작업을 Data Links 서버로 분배합니다. 또한, 트랜잭션에 포함된 Data Links 서버도 추적합니다. 응용프로그램이 트랜잭션에 대해 COMMIT를 발행하면, 데이터베이스 관리 프로그램은 2단계 확약 프로토콜을 사용하여 트랜잭션을 확약합니다. 1단계에서 데이터베이스 관리 프로그램은 PREPARE 로그 레코드를 기록하고 모든 Data Links 서버로 PREPARE 요청을 분배합니다. 그런 다음, 각각의 Data Links 서버는 다음 중 하나로 응답합니다.

- Yes: 데이터 링크 서버가 확약할 준비가 되었음을 나타냅니다.
- No: 오류로 인해 데이터 링크 서버가 확약할 준비가 되어 있지 않습니다.

1단계는 모든 데이터 링크 서버가 『YES』로 응답했으므로 성공한 것으로 간주됩니다.

2단계 처리는 1단계 결과에 따라 달라집니다. 적어도 데이터 링크 서버 중 하나가 『NO』를 응답하면, 데이터베이스 관리 프로그램은 포함된 모든 데이터 링크 서버로 ABORT 요청을 전달합니다. 트랜잭션은 구간 복원되고 이유 코드가 『03』인

오류 메시지 SQL0903N이 응용프로그램으로 리턴됩니다. 그렇지 않으면, 데이터베이스 관리 프로그램은 데이터 링크 서버가 포함되지 않은 것처럼 트랜잭션을 확약합니다. 이 처리 끝에서, COMMIT 요청을 트랜잭션에 포함된 모든 데이터 링크 서버로 분배합니다.

데이터 링크 서버에서 트랜잭션을 PREPARE 상태에 놓이게 하는 오류가 발생하면, 이러한 트랜잭션을 *2단계 확약중 이상 실패 트랜잭션*이라고 합니다 데이터베이스 관리 프로그램은 이러한 트랜잭션 결과를 추적하고, 결국 데이터 링크 서버에서 이를 분석할 책임이 있습니다. 데이터베이스 관리 프로그램이 오류로 인해 데이터 링크 서버에서 2단계 확약중 이상 실패 트랜잭션이 작성될 가능성이 있음을 결정할 때마다, 데이터 링크 서버의 상태를 시스템 손상 복구가 필요한 것으로 표시합니다. 이 상태에 있는 동안에는 데이터 링크 서버에 포함된 SQL 요청이 허용되지 않습니다. 이유 코드가 『03』인 SQL0357N이 SQL 요청을 작성한 응용프로그램으로 리턴됩니다.

RESTART, ACTIVATE DATABASE 또는 첫 번째 CONNECT 처리시, 데이터베이스 관리 프로그램은 각각의 구성된 데이터 링크 서버로 연결하여 2단계 확약중 이상 실패 트랜잭션을 중지시키거나 확약하여 해결하려고 합니다. 데이터 링크 서버의 상태는 데이터베이스 관리 프로그램에서도 2단계 확약중 이상 실패 상태인 트랜잭션을 제외하고 2단계 확약중 이상 실패 트랜잭션 모두가 해결되면 사용 가능한 것으로 표시됩니다. 사용 가능한 상태에서는 데이터 링크 서버를 포함한 SQL 요청이 허용됩니다. 2단계 확약중 이상 실패 트랜잭션을 해결하기 위한 이러한 시도 끝에, 데이터베이스 관리 프로그램이 데이터 링크 서버에 해결이 필요한 잠재적인 2단계 확약중 이상 실패 트랜잭션이 있을 것으로 결정하게 되면, 데이터 링크 서버의 상태를 시스템 손상 복구가 필요한 것으로 표시합니다. 예를 들어, 이것은 데이터 링크 서버를 RESTART, ACTIVATE DATABASE 또는 첫 번째 CONNECT 처리중에 사용할 수 없거나 데이터 링크 서버가 해당 처리 도중 오류를 발견하는 경우 발생할 수 있습니다.

데이터베이스로 구성된 데이터 링크 서버가 시스템 손상 복구가 필요한 상태에 있는 동안, 데이터베이스 관리 프로그램은 특별한 데이터 링크 서버가 포함된 SQL 요청을 허락하지 않습니다. 데이터베이스의 다른 데이터가 포함된 SQL 요청은 계속 허용됩니다. 데이터베이스 관리 프로그램은 비동기적으로 복구가 필요한 각각의 데이터 링크 서버에서 시스템 손상 복구를 시도하는 프로세스를 시작합니다. 프로

세스가 시스템 손상 복구를 완료하면, 데이터 링크 서버 상태가 사용 가능한 것으로 표시되므로, 이를 포함한 더이상의 SQL 요청이 허용됩니다.

백업 유틸리티 고려사항

DB2는 백업 조작이 완료되는 시간에 DB2 Data Links Manager를 수행중인 데이터 링크 서버에 있는 링크 파일도 백업도록 합니다. (백업 유틸리티는 온라인 또는 오프라인일 수 있으며, 백업 이미지는 데이터베이스 또는 테이블 공간일 수 있습니다.) 다음 설명은 RECOVERY 매개변수를 YES로 설정한 DATALINK 컬럼에 의해 링크된 파일에만 적용됩니다. (RECOVERY=NO가 지정된 DATALINK 컬럼으로 참조하는 파일은 백업되지 않습니다.)

파일이 링크되면, 데이터 링크 서버는 TSM과 같은 아카이브 서버로 또는 디스크로 이들이 비동기적으로 복사되도록 스케줄합니다. 백업 유틸리티가 수행되면, DB2는 복사하도록 스케줄된 모든 파일이 복사되었는지 확인합니다. 백업 처리를 시작할 때, DB2는 또한 DB2 구성 파일에서 지정된 모든 데이터 링크 서버에 접속합니다. 데이터베이스에 대해 하나 이상의 데이터 링크 서버가 구성된 경우, 데이터 링크 서버를 사용할 수 없는 경우에도 백업 조작이 성공합니다. 데이터 링크 서버가 재시작되면, 데이터베이스에 대해 다시 사용 가능하게 되기 전에 데이터 링크 서버에서 백업 처리가 완료됩니다. 데이터 링크 서버에 하나 이상의 링크된 파일이 있지만 수행되고 있지 않거나 백업 조작 중에 수행을 중지한 경우, 백업 이미지에는 완전한 DATALINK 정보가 포함되지 않습니다. 백업 조작을 성공적으로 완료합니다. 백업 조작에 대한 실행기록 파일의 주석 필드는 실패한 DLM 서버를 식별하거나, 실패한 DLM 서버가 5개 이상일 경우 실패한 DLM 서버 수를 식별합니다. DLM이 백업되고 나면, 주석 필드는 이전 값으로 재설정됩니다.

데이터 링크가 데이터베이스에 대해 다시 사용 가능한 것으로 표시되기 전에, 모든 미해결 백업에 대한 백업 처리를 완료해야 합니다. (이 작업은 비동기 프로세스에 의해 자동으로 완료됩니다.) 데이터 링크 서버에 이미 *num_db_backups*(아래 참조) 값의 두 배만큼 완료되기를 기다리는 미해결 백업이 있을 때 백업 조작을 초기화하면, 백업 조작이 실패합니다. 그 데이터 링크 서버를 재시작해야 하고, 미해결 백업이 완료되어야 추가 백업이 허용됩니다.

파일이 링크 해제되면, ON UNLINK 매개변수에 지정한 값에 따라 이전 사용권한으로 리턴되거나 삭제됩니다. 백업 조작에 성공하면 데이터 링크 서버가 아카이

브 서버(디스크 또는 TM 둘다. 58 페이지의『가비지 콜렉션』참조)에 있는 아카이브 버전의 파일을 제거할 수 있게 됩니다. *num_db_backups* 데이터베이스 구성 매개변수는 파일의 아카이브 버전(링크 해제된)이 제거되기 전에 DB2 데이터베이스 백업의 수를 지정합니다. 이 구성 매개변수에 대한 자세한 정보는 *관리 안내서 : 성능* 책을 참조하십시오.

링크 해제된 파일이 제거되면, 링크 해제된 파일에 대한 정보도 데이터 링크 서버 등록 테이블에서 제거됩니다.

AIX에서 DB2 Data Links Manager의 백업 방법 선택

디스크 복사와 XBSA 외에도, 데이터 링크 서버에 상주하는 파일을 백업하기 위해 TSM(Tivoli Storage Manager)을 사용할 수도 있습니다.

Tivoli Storage Manager를 아카이브 서버로 사용하는 방법

1. 데이터 링크 서버에 Tivoli Storage Manager를 설치하십시오. 자세한 정보는 Tivoli Storage Manager 제품 문서를 참조하십시오.
2. Tivoli Storage Manager 서버를 사용하여 데이터 링크 서버 클라이언트 응용프로그램을 등록하십시오. 자세한 정보는 Tivoli Storage Manager 제품 문서를 참조하십시오.
3. 다음의 환경 변수를 Data Links Manager 관리자의 `db2profile` 또는 `db2cshrc` 스크립트 파일에 추가하십시오.

   ```
   (Bash, Bourne 또는 Korn shell의 경우)
   export DSMI_DIR=/usr/tivoli/tsm/client/api/bin
   export DSMI_CONFIG=$HOME/tsm/dsm.opt
   export DSMI_LOG=$HOME/dldump
   export PATH=$PATH:$DSMI_DIR
   (C shell의 경우)
   setenv DSMI_DIR /usr/tivoli/tsm/client/api/bin
   setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
   setenv DSMI_LOG ${HOME}/dldump
   setenv PATH=${PATH}:$DSMI_DIR
   ```

4. `dsm.sys` TSM 시스템 옵션 파일이 `$DSMI_DIR` 디렉토리에 있는지 확인하십시오.
5. `dsm.opt` TSM 사용자 옵션이 파일이 *INSTHOME*`/tsm` 디렉토리에 있는지 확인하십시오. 여기서, *INSTHOME*은 Data Links Manager 관리자의 홈 디렉토리입니다.

6. Tivoli Storage Manager 시스템 옵션 파일의 /usr/tivoli/tsm/client/api/bin/dsm.sys에 있는 generate에 *PASSWORDACCESS* 옵션을 설정하십시오.
7. 처음으로 데이터 링크 파일 관리 프로그램을 시작하기 *전에* generate 옵션으로 TSM 암호를 등록하십시오. 이 방법에서는 데이터 링크 파일 관리 프로그램이 TSM 서버에 대한 연결을 초기화할 때 암호를 제공하지 않아도 됩니다. 자세한 정보는 TSM 제품 문서를 참조하십시오.
8. TSM에 DLFM_BACKUP_TARGET 레지스트리 변수를 설정하십시오. 이 경우 DLFM_BACKUP_DIR_NAME 레지스트리 변수의 값은 무시됩니다. 이 변수는 Tivoli Storage Manager 백업 옵션을 활성화합니다.

   주:

   a. 런타임에서 TSM과 디스크 사이에 DLFM_BACKUP_TARGET 레지스트리 변수의 설정을 변경할 경우, 아카이브된 파일이 새로 지정된 아카이브 위치로 이동되지 않는 것을 알아야 합니다. 예를 들어, DLFM_BACKUP_TARGET 레지스트리 값을 TSM으로 설정하여 데이터 링크 파일 관리 프로그램을 시작하고 레지스트리 값을 디스크 위치로 변경할 경우, 새로 아카이브된 모든 파일이 디스크에서 새 위치에 저장됩니다. 이전에 TSM에 아카이브된 파일들은 새로운 디스크 위치로 이동하지 않습니다.

   b. 기본 TSM 관리 클래스를 대체하기 위해 DLFM_TSM_MGMTCLASS라고 하는 새로운 레지스트리 변수가 있습니다. 이 레지스트리 변수를 설정하지 않은 상태로 두면, 기본 TSM 관리 클래스가 사용됩니다.
9. **dlfm stop** 명령을 입력하여 데이터 링크 파일 관리 프로그램을 중지시키십시오.
10. **dlfm start** 명령을 입력하여 데이터 링크 파일 관리 프로그램을 시작하십시오.

Solaris 운영 환경에서 DB2 Data Links Manager의 백업 방법 선택

디스크 복사와 XBSA 외에도, 데이터 링크 서버에 상주하는 파일을 백업하기 위해 TSM(Tivoli Storage Manager)을 사용할 수도 있습니다.

Tivoli Storage Manager를 아카이브 서버로 사용하는 방법

1. 데이터 링크 서버에 Tivoli Storage Manager를 설치하십시오. 자세한 정보는 Tivoli Storage Manager 제품 문서를 참조하십시오.
2. Tivoli Storage Manager 서버를 사용하여 데이터 링크 서버 클라이언트 응용프로그램을 등록하십시오. 자세한 정보는 Tivoli Storage Manager 제품 문서를 참조하십시오.
3. 다음의 환경 변수를 Data Links Manager 관리자의 `db2profile` 또는 `db2cshrc` 스크립트 파일에 추가하십시오.

   ```
   (Bash, Bourne 또는 Korn shell의 경우)
   export DSMI_DIR=/opt/tivoli/tsm/client/api/bin
   export DSMI_CONFIG=$HOME/tsm/dsm.opt
   export DSMI_LOG=$HOME/dldump
   export PATH=$PATH:/opt/tivoli/tsm/client/api/bin
   (C shell의 경우)
   setenv DSMI_DIR /opt/tivoli/tsm/client/api/bin
   setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
   setenv DSMI_LOG ${HOME}/dldump
   setenv PATH=${PATH}:/opt/tivoli/tsm/client/api/bin
   ```

4. `dsm.sys` TSM 시스템 옵션 파일이 `/opt/tivoli/tsm/client/api/bin` 디렉토리에 있는지 확인하십시오.
5. `dsm.opt` TSM 사용자 옵션이 파일이 *INSTHOME*`/tsm` 디렉토리에 있는지 확인하십시오. 여기서, *INSTHOME*은 Data Links Manager 관리자의 홈 디렉토리입니다.
6. Tivoli Storage Manager 시스템 옵션 파일의 `/opt/tivoli/tsm/client/api/bin/dsm.sys`에 있는 `generate`에
   PASSWORDACCESS 옵션을 설정하십시오.
7. 처음으로 데이터 링크 파일 관리 프로그램을 시작하기 *전에* `generate` 옵션으로 TSM 암호를 등록하십시오. 이 방법에서는 데이터 링크 파일 관리 프로그램이 TSM 서버에 대한 연결을 초기화할 때 암호를 제공하지 않아도 됩니다. 자세한 정보는 TSM 제품 문서를 참조하십시오.
8. TSM에 DLFM_BACKUP_TARGET 레지스트리 변수를 설정하십시오. 이 경우 DLFM_BACKUP_DIR_NAME 레지스트리 변수의 값은 무시됩니다. 이 변수는 Tivoli Storage Manager 백업 옵션을 활성화합니다.

주:

a. 런타임에서 TSM과 디스크 사이에 DLFM_BACKUP_TARGET 레지스트리 변수의 설정을 변경할 경우, 아카이브된 파일이 새로 지정된 아카이브 위치로 이동되지 않는다는 것을 명심하십시오. 예를 들어, DLFM_BACKUP_TARGET 레지스트리 값을 TSM으로 설정하여 데이터 링크 파일 관리 프로그램을 시작하고 레지스트리 값을 디스크 위치로 변경할 경우, 새로 아카이브된 모든 파일이 디스크에서 새 위치에 저장됩니다. 이전에 TSM에 아카이브된 파일들은 새로운 디스크 위치로 이동하지 않습니다.

b. 기본 TSM 관리 클래스를 대체하기 위해 DLFM_TSM_MGMTCLASS라고 하는 새로운 레지스트리 변수가 있습니다. 이 레지스트리 변수를 설정하지 않은 상태로 두면, 기본 TSM 관리 클래스가 사용됩니다.

9. **dlfm stop** 명령을 입력하여 데이터 링크 파일 관리 프로그램을 중지시키십시오.

10. **dlfm start** 명령을 입력하여 데이터 링크 파일 관리 프로그램을 시작하십시오.

Windows NT에서 DB2 Data Links Manager의 백업 방법 선택

복구에 대해 정의된 DATALINK 컬럼이 있는 테이블에 DATALINK 값을 삽입할 때마다, 데이터 링크 서버의 해당되는 DATALINK 파일이 아카이브 서버에 백업되도록 스케줄됩니다. 현재, 디스크 복사(기본 방법)와 Tivoli Storage Manager가 아카이브 서버로의 파일 백업에 대해 지원되는 두 가지 옵션입니다. Windows NT용 DB2 Data Links Manager의 차후 릴리스에서는 다른 벤더의 백업 미디어와 소프트웨어가 지원될 것입니다.

디스크 복사(기본 메소드)

DB2 서버에서 백업 유틸리티가 호출될 때, 그 유틸리티는 데이터베이스에 있는 링크된 파일이 데이터 링크 서버에서 DLFM_BACKUP_DIR_NAME 환경 변수에 의해 지정된 디렉토리로 백업되는지 확인합니다. 변수의 기본값은 `c:\dlfmbackup`입니다. 여기서, `c:\`는 Data Links Manager 백업 설치 드라이브를 표시합니다.

이 변수를 c:\dlfmbackup에 설정하려면 다음 명령을 입력하십시오.

```
db2set -g DLFM_BACKUP_DIR_NAME=c:\dlfmbackup
```

DLFM_BACKUP_DIR_NAME 환경 변수에 의해 지정된 위치는 데이터 링크 파일 시스템 필터를 사용하고 필수 공간이 백업 파일에 대해 지정한 디렉토리에서 사용 가능한 파일 시스템에 위치하지 *않아야* 합니다.

또한, 다음 명령을 입력하여 DLFM_BACKUP_TARGET 변수가 LOCAL에 설정되었는지 확인하십시오.

```
db2set -g DLFM_BACKUP_TARGET=LOCAL
```

이들 변수를 설정 및 변경한 후, **dlfm stop** 및 **dlfm start** 명령을 사용하여 데이터 링크 파일 관리 프로그램을 중지시키고 재시작하십시오.

Tivoli Storage Manager

Tivoli Storage Manager를 아카이브 서버로 사용하는 방법

1. 데이터 링크 서버에 Tivoli Storage Manager를 설치하십시오. 자세한 정보는 Tivoli Storage Manager 제품 문서를 참조하십시오.
2. Tivoli Storage Manager 서버를 사용하여 데이터 링크 서버 클라이언트 응용프로그램을 등록하십시오. 자세한 정보는 Tivoli Storage Manager 제품 문서를 참조하십시오.
3. **시작**을 눌러 **설정 --> 제어판 --> 시스템**을 선택하십시오. 시스템 등록 정보 창이 열립니다. **환경** 탭을 선택하여 다음 환경 변수 및 해당 값을 입력하십시오.

변수	값
DSMI_DIR	c:\tsm\baclient
DSMI_CONFIG	c:\tsm\baclient\dsm.opt
DSMI_LOG	c:\tsm\dldump

4. `dsm.sys` TSM 시스템 옵션 파일이 `c:\tsm\baclient` 디렉토리에 있는지 확인하십시오.
5. `dsm.opt` TSM 사용자 옵션 파일이 `c:\tsm\baclient` 디렉토리에 있는지 확인하십시오.

6. Tivoli Storage Manager 시스템 옵션 파일의 `c:\tsm\baclient\dsm.sys`에 있는 generate에 *PASSWORDACCESS* 옵션을 설정하십시오.
7. 처음으로 데이터 링크 파일 관리 프로그램을 시작하기 *전에* generate 옵션으로 TSM 암호를 등록하십시오. 이 방법에서는 데이터 링크 파일 관리 프로그램이 TSM 서버에 대한 연결을 초기화할 때 암호를 제공하지 않아도 됩니다. 자세한 정보는 TSM 제품 문서를 참조하십시오.
8. 다음 명령을 사용하여 DLFM_BACKUP_TARGET 환경 변수를 TSM에 설정하십시오.

   ```
   db2set -g DLFM_BACKUP_TARGET=TSM
   ```

   이 경우 DLFM_BACKUP_DIR_NAME 환경 변수의 값은 무시됩니다. 이 변수는 Tivoli Storage Manager 백업 옵션을 활성화합니다.

   주:

   a. 런타임에서 TSM과 LOCAL 사이에 DLFM_BACKUP_TARGET 환경 변수의 설정을 변경할 경우, 아카이브된 파일이 새로 지정된 아카이브 위치로 이동되지 않는다는 것을 명심하십시오. 예를 들어, DLFM_BACKUP_TARGET 환경 변수를 TSM으로 설정하여 데이터 링크 파일 관리 프로그램을 시작하고 그 값은 LOCAL로 변경할 경우, 새로 아카이브된 모든 파일이 디스크에서 새 위치에 저장됩니다. 이전에 TSM에 아카이브된 파일들은 새로운 디스크 위치로 이동하지 않습니다.

   b. 기본 TSM 관리 클래스를 대체하기 위해 DLFM_TSM_MGMTCLASS라고 하는 새로운 환경 변수가 있습니다. 이 변수를 설정하지 않은 상태로 두면, 기본 TSM 관리 클래스가 사용됩니다.
9. **dlfm stop** 명령을 입력하여 데이터 링크 파일 관리 프로그램을 중지시키십시오.

10. **dlfm start** 명령을 입력하여 데이터 링크 파일 관리 프로그램을 시작하십시오.

AIX에 JFS(Journaled File System) 백업

Data Links Manager를 중지시킨 후 AIX에서 JFS(Journaled File System)의 백업을 오프라인에서 수행할 수 있습니다. 더 고가용성을 요구하는 사용자의 경우, 다음과 같이 Data Links Manager를 중지시켜야 하는 요구사항을 제거하는 것이 바람직합니다.

1. CLI 소스 파일 quiesce.c 및 쉘 스크립트 online.sh를 액세스하십시오. 이들 파일은 /samples/dlfm 디렉토리에 위치합니다.
2. quiesce.c 컴파일:

```
xlC -o quiesce  -L$HOME/sqllib/lib -I$HOME/sqllib/include -c quiesce.c
```

3. 루트로서 DLFS 파일 시스템을 가지고 있는 노드에서 스크립트를 실행하십시오.

쉘 스크립트 online.sh에서는 Data Link Manager로 등록한 각 데이터베이스에 대해 Data Link Manager 노드에 카탈로그 항목을 가지고 있는 것으로 간주합니다. 또한, /etc/filesystems에 DLFS 파일 시스템에 대한 완전한 항목이 있는 것으로도 간주합니다. 쉘 스크립트는 다음 사항을 수행합니다.

- Data Links Manager로 등록한 데이터베이스의 모든 테이블을 quiesce합니다. 이 작업은 모든 새로운 활동을 중지시킵니다.
- 파일 시스템을 마운트 취소한 다음 읽기 전용 파일 시스템으로 다시 마운트합니다.
- 파일 시스템 백업 수행
- 파일 시스템을 마운트 취소한 다음 읽기-쓰기 파일 시스템으로 다시 마운트합니다.
- DB2 테이블을 재설정합니다. 즉, quiesce 상태 밖으로 가져옵니다.

스크립트는 다음과 같이 환경에 맞도록 수정해야 합니다.

1. 백업 명령을 선택한 다음 스크립트의 do_backup 함수에 놓으십시오.
2. 스크립트 내에서 다음 환경 변수를 설정하십시오.
   - DLFM_INST: DLFM 인스턴스 이름에 설정

- PATH_OF_EXEC: 이를 ″quiesce″ 실행 파일이 있는 경로로 설정하십시오.

스크립트를 다음과 같이 호출하십시오.

```
online.sh <filesystem_name>
```

복원 및 롤 포워드 유틸리티 고려사항

다음 정보는 테이블 공간에서 테이블에 대해 RECOVERY=YES가 정의된 DATALINK 컬럼이 있는 경우 적용됩니다. 테이블에 RECOVERY=NO 옵션이 정의된 DATALINK 컬럼이 있으면, 테이블은 복원 조작 끝에서 데이터 링크 조정 상태가 됩니다. 87 페이지의 『데이터 링크 조정』에서 자세한 내용을 참조하십시오.

복원 조작 동안 DATALINK 컬럼이 있는 테이블은 다음 두 상태 중 하나에 해당 됩니다.

- *데이터 링크 조정이 가능하지 않음*

  테이블이 데이터 링크 조정 불가능 상태에 있을 경우, 이는 DATALINK 컬럼이 아닌 컬럼에 대해 제한되지 않은 조작 조치에 사용할 수 있습니다. DATALINK 컬럼이 SELECT문에 포함되면, 경고가 리턴됩니다. UPDATE 호출을 DATALINK 컬럼에 대해 발행할 수 있습니다. (제한사항에 대해서는 86 페이지의 『데이터 링크 조정이 불가능한 상태에서 테이블 제거』에서 자세한 내용을 참조하십시오.) INSERT 및 DELETE문은 DATALINK 컬럼을 포함하므로 발행할 수 없습니다.

- *데이터 링크 조정 보류*

  테이블이 데이터 링크 조정 보류 상태에 있을 경우, 이는 DATALINK 컬럼이 아닌 컬럼에 대해 제한되지 않은 조치에 사용할 수 있습니다. DATALINK 컬럼이 SELECT문에 포함되면, 경고가 리턴됩니다. UPDATE, INSERT 또는 DELETE와 같은 DML문을 발행할 수 없습니다.

이 상태는 복원 또는 롤 포워드 유틸리티가 수행될 때 db2diag.log 파일에 보고됩니다. 또한, **db2dart** 명령을 사용하여 이 정보를 취득할 수도 있습니다.

데이터베이스 또는 테이블 공간을 복원할 때 복원 조작이 성공하려면, 다음 조건을 충족시켜야 합니다.

- 백업 파일에 기록된 데이터 링크 서버가 수행 중 상태가 아니어도, 복원 조작은 완료됩니다.

  누락된 데이터 링크 서버의 영향을 받는 DATALINK 컬럼 정보가 있는 테이블은 복원 조작(또는 롤 포워드 조작)이 완료된 후에 데이터 링크 조정 보류 상태에 놓입니다. 데이터 링크 서버가 데이터베이스에 대해 다시 사용 가능한 것

으로 표시되려면, 먼저 이 복원 처리를 완료해야 합니다. DLM이 사용 가능하게 되면(67 페이지의 『백업 유틸리티 고려사항』 참조) 한 번에 백업 처리를 완료하는 비동기 프로세스도 복원 처리를 완료합니다.

- 백업 파일에 기록된 데이터 링크 서버가 복원 조작 중에 수행을 중지할 경우, 복원 조작은 실패하게 됩니다. 복원은 데이터 링크 서버가 중단된 상태에서 재시작할 수 있습니다(위의 내용 참조).
- 이전 데이터베이스 복원 조작이 데이터 링크 서버에서 계속 완료되지 않을 경우, 해당되는 데이터 링크 서버가 재시작하고, 완료되지 않은 복원이 완료될 때까지 후속 데이터베이스 또는 테이블 공간 복원 조작은 실패합니다.
- 백업 파일에 기록된 모든 DATALINK 컬럼에 대한 정보가 해당 데이터 링크 서버의 등록 테이블에 있어야 합니다.

  DATALINK 컬럼에 대한 모든 정보가 등록 테이블에 기록되지 않으면, DATALINK 컬럼 정보가 누락된 테이블은 복원 조작(또는 롤 포워드 조작)이 완료된 후 데이터 링크 조정 불가능 상태가 됩니다.

  백업이 등록 테이블에 기록되지 않으면, 이는 제공된 백업 파일이 num_db_backups 값보다 이전 것이므로 이미 ″가비지 콜렉션 처리″되었음을 의미합니다. 이것은 이전 버전에서 백업의 아카이브 파일이 제거되었으므로 복원될 수 없음을 의미합니다. DATALINK 컬럼이 있는 모든 테이블은 데이터링크 조정 보류 상태에 놓입니다.

  백업이 등록 테이블에 기록되지 않으면, 이는 데이터 링크 서버가 수행되고 있지 않아서 아직 백업 처리가 완료되지 않았음을 의미합니다. DATALINK 컬럼이 있는 모든 테이블은 데이터 링크 조정 보류 상태에 놓입니다. 데이터 링크 서버가 재시작되면, 복원 처리 이전에 백업 처리가 완료됩니다.

  테이블은 사용자가 사용할 수 있는 상태로 남지만, DATALINK 컬럼에 있는 값은 정확히 파일을 참조하지 않을 수 있습니다. (예를 들어, DATALINK 컬럼 값에 대응되는 파일을 찾지 못할 수도 있습니다.) 이러한 작동을 원하지 않는다면, ″SET CONSTRAINTS for tablename TO DATALINK RECONCILE PENDING″문을 발행하여 테이블을 점검 보류 상태가 되도록 할 수 있습니다.

복원 조작 이후 테이블이 데이터 링크 조정이 가능하지 않음 상태가 될 경우, 86 페이지의 『데이터 링크 조정이 불가능한 상태에서 테이블 제거』 아래에 제안된 방법 중 하나를 사용하여 DATALINK 컬럼 데이터를 정정할 수 있습니다.

주: 링크 해제된 상태에서 링크된 상태로 파일을 표시하는 프로세스에서, 해당 파일은 파일 시스템에 대한 아카이브 서버에서 검색되어야 합니다. 이 프로세스 도중 오류가 발생하면(예를 들어, 중복된 파일 이름으로 인해 파일 시스템으로 파일을 복사할 수 없음), 해당 테이블은 데이터 링크 조정 보류 상태가 됩니다.

특정의 복구 경우를 포괄하도록 datalink.cfg 파일이 아카이브할 것을 강력히 권장합니다. 데이터베이스 백업 이미지의 datalink.cfg 파일은 백업 시간 상태의 datalink.cfg만 반영하기 때문입니다. 모든 복구 경우를 포괄하려면 최근의 datalink.cfg 파일을 가지고 있어야 합니다. 그러므로 datalink.cfg 파일은 모든 ADD DATALINKS MANAGER 또는 DROP DATALINKS MANAGER 명령 호출 후에 백업해야 합니다. 그러면 디스크에서 최근 datalink.cfg 파일을 사용할 수 없는 경우에 최근의 datalink.cfg 파일을 검색하는데 도움이 됩니다.

디스크에서 최근의 datalink.cfg 파일을 사용할 수 없으면, 기존의 datalink.cfg 파일(백업 이미지로부터 복원된)을 롤 포워드 조작 수행 이전에 아카이브한 최근 datalink.cfg 파일로 바꾸십시오. 데이터베이스가 복원된 후 이를 수행하십시오.

롤 포워드 없이 오프라인 백업에서 데이터베이스 복원

테이블 공간 레벨이 아닌 데이터베이스 레벨에서 롤 포워드 없이 복원만 가능합니다. 롤 포워드 없이 데이터베이스를 복원하려면, 복구 불가능한 데이터베이스(즉, 순환 로그를 사용하는 데이터베이스)를 복원하거나 RESTORE DATABASE 명령의 WITHOUT ROLLING FORWARD 매개변수를 지정합니다.

WITHOUT DATALINK 옵션과 함께 복원 유틸리티를 사용하는 경우, 데이터 링크 조정 보류(DRP) 상태에 있는 DATALINK 컬럼이 있는 모든 테이블은 복원 조작 동안 데이터 링크 서버와의 어떠한 조정도 수행되지 않습니다.

WITHOUT DATALINK 옵션을 사용하지 않고 백업 파일에 기록된 데이터 링크 서버가 더이상 데이터베이스에 대해 정의되지 않은 경우(즉, DROP DATALINKS MANAGER 명령을 사용하여 제거한), 제거된 데이터 링크 서버를 참조하는 DATALINK 데이터를 포함하는 테이블은 복원 유틸리티에 의해 DRP 상태에 놓입니다.

WITHOUT DATALINK 옵션을 사용하지 않고 모든 데이터 링크 서버를 사용할 수 있으며 DATALINK 컬럼에 대한 모든 정보가 등록 테이블에 모두 기록되면, 백업 파일에 기록된 각각의 데이터 링크 서버에 대해 다음이 발생합니다.

- 이들 파일은 백업 이미지에서 링크된 것으로 기록되지 않으므로, 데이터베이스 복원 조작에 사용된 백업 이미지 이후에 링크된 모든 파일은 링크 해제된 것으로 표시됩니다.
- 이들 파일은 백업 이미지에서 링크된 것으로 표시되므로, 백업 이미지 이후에 링크 해제되었으나 백업 이미지가 형성될 때 링크된 모든 파일은 링크된 것으로 표시됩니다. 파일이 차후에 다른 데이터베이스에 있는 다른 테이블로 링크되면, 복원된 테이블은 데이터 링크 조정 보류 상태가 됩니다.

주: 데이터베이스 복원 조작에 대해 사용된 백업 이미지가 최소한 하나의 데이터 링크 서버도 수행되고 있지 않을 때 취해진 경우, 백업의 DATALINK는 정보가 불완전하므로 위의 내용을 수행할 수 없습니다. 또한, 데이터베이스 복원 조작에 대해 사용된 백업 이미지가 데이터베이스 복원 후에 롤 포워드 복구를 사용하거나 사용하지 않고 취해진 경우에도 수행되지 않습니다. 두 경우 모두, 데이터 링크 조정 보류 상태의 DATALINK 컬럼이 있는 모든 테이블은 복원 조작 동안 데이터 링크 서버와의 어떠한 조정도 수행되지 않습니다.

데이터베이스와 테이블 공간 복원 및 로그 끝까지 롤 포워드

로그의 끝까지 데이터베이스 또는 테이블 공간을 복원한 다음 롤 포워드할 경우 (즉 모든 로그가 제공되는 경우), 백업 파일에 기록된 최소한 하나 이상의 데이터 링크 서버가 복원 중에 수행되고 있으면 조정 점검은 필요하지 않습니다. 롤 포워드 조작에 모든 로그가 제공되는지 여부가 확실하지 않거나, DATALINK 값을 조정해야 할 필요가 있다고 생각되는 경우, 다음을 수행하십시오.

1. 포함된 테이블에 대해 SQL문을 발행하십시오.

   ```
   SET CONSTRAINTS FOR tablename TO DATALINK RECONCILE PENDING
   ```

   그러면 테이블이 데이터 링크 조정 보류 상태와 점검 보류 상태에 놓입니다.

2. 테이블이 점검 보류 상태에 있지 않도록 하려면, 다음 SQL문을 발행하십시오.

   ```
   SET CONSTRAINTS FOR tablename IMMEDIATE CHECKED
   ```

이것으로 테이블은 점검 보류 상태에서 벗어나지만 데이터 링크 조정 보류 상태는 그대로 유지됩니다. 이 상태로부터 테이블을 가져오려면 조정 유틸리티를 사용해야 합니다.

백업 파일에 데이터베이스에서 제거된 DB2 Data Links Manager를 참조하는 DATALINK 데이터가 있을 수 있습니다. (즉, 백업을 취할 때 DB2 Data Links Manager가 데이터베이스에 등록되었습니다.) 제거된 DB2 Data Links Manager를 참조하는 DATALINK 데이터가 있는 최소한 하나 이상의 테이블을 포함하는 각 테이블 공간이 롤 포워드될 경우, DATALINK 컬럼이 있는 모든 테이블은 롤 포워드 유틸리티에 의해 DRP 상태에 놓입니다.

데이터베이스와 테이블 공간 복원 및 특정 시점까지 롤 포워드

데이터 링크 테이블에 대해 작업할 때에는 로그 끝 또는 특정 시점으로 롤 포워드할 수 있습니다.

특정 시점으로 롤 포워드된 테이블 공간의 테이블은 롤 포워드 조작 끝에서 데이터 링크 조정 보류 상태가 됩니다. 이 상태로부터 테이블을 제거하려면 조정 유틸리티를 사용해야 합니다. 87 페이지의 『데이터 링크 조정』에서 자세한 내용을 참조하십시오.

특정 시점 롤 포워드 예

다음은 백업 및 복구를 핸들하기 위해 보유될 필요가 있는 파일을 보여주는 간단한 시나리오입니다. 이 예에서는 복구를 지원하기 위해 DB2 Data Links Manager가 보유해야 하는 파일과 함께 DATALINK 유형의 컬럼에서 단일 행 값에 대한 변경사항을 보여줍니다. 이 예의 경우, 최종 백업보다 이전 파일의 특정 시점 복구를 지원하지 않는다고 가정합니다. DB2 Data Links Manager를 수행하는 데이터 링크 서버에는 이러한 제한사항이 없습니다. 시간 3까지 fileA가 존재하는지 관찰하십시오. 이때 시간 2에서는 fileA가 링크 해제되었으므로 삭제되며 이 예에 있는 데이터베이스에 대한 방침은 다음 백업이 수행될 때까지 링크 해제된 파일을 보존하는 것입니다. (즉, *num_db_backups* 데이터베이스 구성 매개변수가 1로 설정됩니다.)

시간	1 1	2	3	4	5	6	7
활동	작성	갱신	백업	갱신	갱신	삭제	5로 복원
컬럼 값	valueA	valueB	valueB	valueC	valueD	-	valueD
링크된 파일	fileA	fileB	fileB	fileC	fileD	-	fileD
데이터 링크 파일 관리 프로그램이 유지하는 기타 파일		fileA		fileB	fileB, fileC	fileB, fileC, fileD	fileB, fileC

주: 링크된 파일의 복구는 나머지 데이터베이스와 결합하여 수행됩니다.

DB2 Data Links Manager와 복구의 상호작용

다음 테이블에서는 사용자가 수행할 수 있는 서로 다른 유형의 복구, 복원 및 롤포워드 처리 도중 발생하는 DB2 Data Links Manager 처리, 그리고 복구가 완료된 후 조정 유틸리티를 호출해야 하는지 여부를 보여줍니다.

복구 유형	복원 중에 DB2 Data Links Manager 처리	롤 포워드 중에 DB2 Data Links Manager 처리	조정
복구 불가능한 데이터베이스			
완전 백업의 데이터베이스 복원, 모든 데이터 링크 서버 가동	빠른 조정이 수행됩니다.	N/A	파일 링크의 문제점이 의심스러우면 선택적으로 수행할 수 있습니다.
WITHOUT DATALINK 옵션을 사용하여 데이터베이스 복원	데이터 링크 조정 보류 상태의 테이블	N/A	필수
완전 백업의 데이터베이스 복원, 최소한 하나 이상의 데이터 링크 서버 중단	중단된 데이터 링크 서버에 링크되어 있지 않은 테이블 공간의 테이블에 대해서만 빠른 조정이 수행되며, 다른 테이블은 데이터 링크 조정 보류 상태에 놓임	NA	중단된 데이터 링크 서버에 링크되어 있는 테이블 공간의 테이블에 대해서만 필수
불완전 백업의 데이터베이스 복원, 모든 데이터 링크 서버 가동	빠른 조정은 수행되지 않고, DATALINK 컬럼이 있는 모든 테이블은 데이터 링크 조정 보류 상태에 놓임	NA	필수
복구 가능한 데이터베이스			
WITHOUT ROLLING FORWARD 옵션을 사용하여 데이터베이스 복원, 완전 백업 사용, 모든 데이터 링크 서버 가동	빠른 조정이 수행됩니다.	N/A	선택적

복구 유형	복원 중에 DB2 Data Links Manager 처리	롤 포워드 중에 DB2 Data Links Manager 처리	조정
WITHOUT ROLLING FORWARD 및 WITHOUT DATALINK 옵션을 사용하여 데이터베이스 복원, 완전 또는 불완전 백업 사용, 모든 데이터 링크 서버 가동 또는 중단	데이터 링크 조정 보류 상태의 테이블	N/A	필수
WITHOUT ROLLING FORWARD 옵션을 사용하여 데이터베이스 복원, 완전 백업 사용, 최소한 하나 이상의 데이터 링크 서버 중단	중단된 데이터 링크 서버에 링크되어 있지 않은 테이블 공간의 테이블에 대해서만 빠른 조정이 수행되며, 다른 테이블은 데이터 링크 조정 보류 상태에 놓임	N/A	중단된 데이터 링크 서버에 링크되어 있는 테이블 공간의 테이블에 대해서만 필수
WITHOUT ROLLING FORWARD 옵션을 사용하여 데이터베이스 복원, 불완전 백업 사용, 모든 데이터 링크 서버 가동 또는 중단	빠른 조정은 수행되지 않고, DATALINK 컬럼이 있는 모든 테이블은 데이터 링크 조정 보류 상태에 놓임	N/A	필수
로그 끝까지 데이터베이스 복원 및 롤 포워드, 완전 백업 사용, 모든 데이터 링크 서버 가동	조치 없음	조치 없음	선택적
로그 끝까지 데이터베이스 복원 및 롤 포워드, 완전 백업 사용, 롤 포워드 처리 동안 최소한 하나 이상의 데이터 링크 서버 중단	조치 없음	조치 없음	선택적

복구 유형	복원 중에 DB2 Data Links Manager 처리	롤 포워드 중에 DB2 Data Links Manager 처리	조정
로그 끝까지 데이터베이스 복원 및 롤 포워드, 완전 또는 불완전 백업 사용, 복원 동안 데이터 링크 서버 중단	조치 없음	DATALINK 컬럼이 있는 모든 테이블은 데이터 링크 조정 보류 상태에 놓임	DATALINK 컬럼이 있는 모든 테이블에 대해 필수
로그 끝까지 데이터베이스 복원 및 롤 포워드, 불완전 백업 사용, 복원 동안 모든 데이터 링크 서버 가동	조치 없음	조치 없음	선택적
로그 끝까지 데이터베이스 복원 및 롤 포워드, 완전 또는 불완전 백업 사용, 모든 데이터 링크 서버 가동, 어떤 데이터 링크 서버에서도 알 수 없는 백업	조치 없음	백업을 알 수 없는 데이터 링크 서버에 링크되어 있는 테이블 공간의 모든 테이블이 데이터 링크 조정 보류 상태에 놓임	필수
로그 끝까지 테이블 공간 복원 및 롤 포워드, 완전 백업 사용, 모든 데이터 링크 서버 가동	조치 없음	조치 없음	선택적
로그 끝까지 테이블 공간 복원 및 롤 포워드, 완전 백업 사용, 롤 포워드 처리 동안 최소한 하나 이상의 데이터 링크 서버 중단	조치 없음	조치 없음	선택적
로그 끝까지 테이블 공간 복원 및 롤 포워드, 완전 또는 불완전 백업 사용, 복원 처리 동안 데이터 링크 서버 중단	조치 없음	중단된 데이터 링크 서버에 링크되어 있는 테이블 공간의 모든 테이블이 데이터 링크 조정 보류 상태에 놓임	중단된 데이터 링크 서버에 링크되어 있는 테이블 공간의 테이블에 대해 필수
로그 끝까지 테이블 공간 복원 및 롤 포워드, 불완전 백업 사용, 모든 데이터 링크 서버 가동	조치 없음	조치 없음	선택적

복구 유형	복원 중에 DB2 Data Links Manager 처리	롤 포워드 중에 DB2 Data Links Manager 처리	조정
특정 시점으로 데이터베이스 복원 및 롤 포워드, 완전 또는 불완전 백업 사용, 복원 및 (또는) 롤 포워드 처리 동안 데이터 링크 서버 가동 또는 중단	조치 없음	데이터 링크 조정 보류 상태의 테이블	필수
특정 시점으로 테이블 공간 복원 및 롤 포워드, 완전 또는 불완전 백업 사용, 복원 및(또는) 롤 포워드 처리 동안 데이터 링크 서버 가동 또는 중단	조치 없음	데이터 링크 조정 보류 상태의 테이블	필수
롤 포워드 없이 다른 데이터베이스 이름, 별명, 호스트 이름 또는 인스턴스로 데이터베이스 복원(86 페이지의 1 주 참조)	데이터 링크 조정이 불가능한 상태의 테이블	N/A	선택적. 데이터 링크 조정이 불가능한 상태의 테이블은 수동으로 정정되어야 합니다.
다른 데이터베이스 이름, 별명, 호스트 이름 또는 인스턴스 및 롤 포워드로 데이터베이스 복원	조치 없음	데이터 링크 조정이 불가능한 상태의 테이블	선택적. 데이터 링크 조정이 불가능한 상태의 테이블은 수동으로 정정되어야 합니다.
WITHOUT DATALINK 옵션을 사용하거나 사용하지 않고, 롤 포워드 없이 사용 불가능한 백업(이미지가 데이터 링크 서버에서 가비지 콜렉션 처리됨)에서 데이터베이스 복원(86 페이지의 1 주 참조)	데이터 링크 조정 보류 상태의 테이블	조치 없음	필수

복구 유형	복원 중에 DB2 Data Links Manager 처리	롤 포워드 중에 DB2 Data Links Manager 처리	조정
WITHOUT DATALINK 옵션을 사용하거나 사용하지 않고, 사용 불가능한 백업(이미지가 데이터 링크 서버에서 가비지 콜렉션 처리됨) 및 롤 포워드에서 데이터베이스 복원	조치 없음	데이터 링크 조정 보류 상태의 테이블	필수
사용 불가능한 백업(이미지가 데이터 링크 서버에서 가비지 수집됨)에서의 테이블 공간 복원 및 롤 포워드	조치 없음	데이터 링크 조정 보류 상태의 테이블	필수

주:

1. 온라인 백업 이미지와 WITHOUT ROLLING FORWARD 옵션을 사용한 복원 조작이나, 오프라인 백업 이미지를 사용한 복원 조작
2. *완전* 백업은 모든 필수 데이터 링크 서버가 수행 중일 때 취해진 백업입니다. *불완전* 백업은 최소한 하나 이상의 필수 데이터 링크 서버가 수행되고 있지 않을 때 취해진 백업입니다.
3. 빠른 조정 처리는 데이터베이스 복원 조작에 대해 사용된 백업 이미지가 데이터베이스 복원 후에 롤 포워드를 사용하거나 사용하지 않고 취해진 경우에 수행되지 않습니다. 이 경우, DATALINK 컬럼이 있는 모든 테이블은 데이터 링크 조정 보류 상태에 놓입니다.

데이터 링크 조정이 불가능한 상태에서 테이블 제거

DATALINK 컬럼이 있는 복원된 테이블은 테이블 공간이 *num_db_backup* 데이터베이스 구성 매개변수에 대해 지정한 값보다 더 이전인 백업에서 복원되는 경우에 데이터 링크 조정 불가능 상태에 놓입니다. 이 구성 매개변수에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.

DATALINK 컬럼 값이 유효하지 않더라도 DB2는 테이블이 액세스되도록 허용합니다. 불일치 DATALINK 컬럼 값이 있는 테이블에 대한 액세스를 막으려면, *테이블 이름* TO DATALINK RECONCILE PENDING 명령에 대한 SET CONSTRAINT를 발행하십시오. 다음과 같이 DATALINK 값을 갱신할 수 있습니다.

- SQL UPDATE문을 사용하여 컬럼이 널(NULL) 입력 불가능하면 DATALINK 컬럼의 데이터 위치 부분을 길이가 0인 URL로 설정하거나, 컬럼이 널(NULL) 입력 가능하면 널(NULL)로 설정하십시오.
- 해당 데이터 링크 서버에서 파일을 복원하십시오. 그런 다음, DATALINK 컬럼 값을 읽기 위해 SELECT문을 발행하고 동일한 값으로 DATALINK 컬럼을 갱신하기 위해 UPDATE문을 발행하는 응용프로그램을 수행하십시오. 데이터 링크 조정 불가능 상태는 DATALINK 컬럼 값이 갱신되는 동안에 적용되어야 합니다. 갱신 조작이 완료된 후, 파일은 해당 데이터 링크 서버에서 링크된 것으로 표시됩니다.

그런 다음, 다음 명령을 발행하여 데이터 링크 조정이 불가능한 상태를 재설정합니다.

```
SET CONSTRAINTS FOR tablename DATALINK RECONCILE PENDING IMMEDIATE UNCHECKED
```

데이터 링크 조정

데이터 링크를 조정하기 위해 조정 유틸리티를 사용합니다. 유틸리티는 DB2에서 시작되며, DATALINK 컬럼 값이 참조하는 DB2 Data Links Manager를 수행하는 모든 데이터 링크 서버가 포함됩니다. 참조 파일이 데이터 링크 서버에 있는지, 아니면 링크가 재설정될 수 있는지 확인합니다. 다음 절에서는 데이터 링크를 조정할 필요가 있는지 여부를 DB2가 어떻게 검출하고 데이터 링크를 어떻게 조정하는가에 대해 설명합니다.

데이터 링크 서버 파일 참조가 없거나 재설정될 수 없는 경우, 조정 유틸리티는 이유와 함께 오류가 있는 행 사본을 각 예외 테이블에 위치시킨 후 잘못된 행을 수정합니다. 예외 테이블이 지정되지 않으면, 다시 구성될 수 없는 파일 참조에 대한 DATALINK 컬럼 값은 컬럼 ID 및 이유와 함께 예외 보고서 파일로 복사됩니다. 예외 테이블(지정된 경우) 정보 또는 보고서를 사용하여 필요한 정정을 수행하도록 행을 갱신할 수 있습니다. 조정 유틸리티와 함께 사용되는 예외 테이블은 로드 유

틸리티가 사용하는 예외 테이블과 같습니다. 로드 유틸리티에 대한 자세한 정보는 *데이터 이동 유틸리티 안내 및 참조서*에서 참조하십시오. 보고서는 *report*.exp 이름 지정 규약(조정 유틸리티는 .exp 확장자를 제공함)을 사용합니다. 예를 들어, 다음 명령문을 사용하여 조정 유틸리티를 호출할 수 있습니다.

```
db2 RECONCILE dept DLREPORT  /u/scottba/report FOR EXCEPTION excptab
```

이 명령은 사용자가 작성한 dept라는 테이블을 조정하고, 예외 테이블 report.exp에 예외를 작성합니다. 조정 도중 링크 해제된 파일에 대해서는 report.ulk 파일로 기록되며, 이것은 /u/scottba 디렉토리에 작성됩니다. *FOR EXCEPTION excptab*가 지정되지 않으면, 예외 정보는 /u/scottba 디렉토리에서 작성된 report.exp 파일로 기록됩니다. 조정 유틸리티에 대한 자세한 정보는 *Command Reference*를 참조하십시오.

조정이 필요한 상황 검출

조정 유틸리티를 수행할 필요가 있는 상황은 다음과 같습니다.

- 전체 데이터베이스가 복원되며 특정 시점까지 롤 포워드됩니다. 전체 데이터베이스는 확약된 트랜잭션까지 롤 포워드되므로, 참조 제한조건 또는 점검 제한조건에 따라 점검 보류 상태에는 테이블이 없습니다. 데이터베이스의 모든 데이터를 일관된 상태로 가져옵니다. 그러나 DATALINK 컬럼은 DB2 Data Links Manager의 메타 데이터로 동기화될 수 없으며, 조정이 필요합니다.

  이 상황에서,DATALINK 데이터가 있는 테이블은 이미 DRP 상태에 있습니다. 이들 각 테이블에 대해 조정 유틸리티를 호출해야 합니다.

- DB2 Data Links Manager를 수행하는 특별한 데이터 링크 서버는 해당 메타 데이터의 트랙을 유실합니다. 이것은 다른 이유로 발생할 수 있습니다. 예를 들면, 다음과 같습니다.
  - 데이터 링크 서버가 콜드 스타트(cold start)되었습니다.
  - 데이터 링크 서버 메타 데이터가 백 레벨(back-level) 상태로 복원되었습니다.

  SQL UPDATE 및 DELETE 수행중과 같은 일부 상황에서, DB2는 데이터 링크 서버에서 메타 데이터의 문제점을 검출할 수 있습니다. 이 상황에서, SQL문은 실패합니다. SET CONSTRAINTS문을 사용하여 테이블을 DRP 상태로 만든 다음, 해당 테이블에 대해 조정 유틸리티를 수행합니다.

- 파일 시스템은 사용 가능하지 않으며(예를 들어, 디스크 이상 때문에) 현재 상태로 복원되지 않습니다. 이 상황에서, 파일이 누락될 수 있습니다.
- DB2 Data Links Manager는 데이터베이스에서 제거되고, 해당 DB2 Data Links Manager를 참조하는 DATALINK FILE LINK CONTROL 값이 있습니다. 이러한 테이블에 대해 조정 유틸리티를 수행해야 합니다.

조정을 위한 프로시듀어 요약

특정 시점 복구로 인해 또는 DB2 Data Links Manager 및 DB2 제어 정보를 수행하는 데이터 링크 서버가 일치하지 않으므로 데이터 링크를 조정할 필요가 있는 경우, 다음을 수행하십시오.

1. ET CONSTRAINTS문을 발행하여 테이블을 데이터 링크 조정 보류 상태가 되도록 하십시오. (어떤 상황에서는 DB2가 이를 수행합니다.)
2. 조정 유틸리티를 사용하여 링크를 해결하고, 예외 테이블 또는 예외 보고서에 있는 예외에 대한 해당 조치를 취하십시오.

제2장 데이터베이스 백업

이 절에서는 데이터베이스 또는 테이블 공간의 백업 사본을 작성하기 위해 사용되는 DB2 UDB 백업 유틸리티리에 대해 설명합니다.

이 절에는 다음 주제를 다룹니다.

백업 개요

DB2 BACKUP DATABASE 명령의 가장 간단한 양식에서는 백업할 데이터베이스의 별명 이름만 지정하면 됩니다. 예를 들면, 다음과 같습니다.

```
db2 backup db sample
```

명령이 제대로 완료되면, 명령이 발행된 디렉토리나 경로에 위치하는 새 백업 이미지를 확보하게 됩니다. 이 예의 명령이 명시적으로 백업 이미지의 대상 위치를

지정하지 않으므로 이미지는 이 디렉토리에 위치됩니다. 예를 들어, Windows NT/2000에서 이 명령은(루트 디렉토리에서 발행한 경우) 다음과 같은 디렉토리 목록에 표시되는 이미지를 작성합니다.

```
 Directory of D:\SAMPLE.0\DB2\NODE0000\CATN0000\20010320
03/20/2001  12:26p      <DIR>          .
03/20/2001  12:26p      <DIR>          ..
03/20/2001  12:27p          12,615,680 122644.001
```

백업 이미지는 백업 유틸리티 호출 시기를 지정하기 위한 옵션을 가지고 있는 대상 위치에서 작성됩니다. 이 위치는 다음과 같을 수 있습니다.

- (디스크 또는 디스켓에 백업하기 위한) 디렉토리
- (테이프에 백업하기 위한) 장치
- TSM(Tivoli Storage Manager) 서버(479 페이지의 『부록G. Tivoli Storage Manager』 참조)
- 다른 벤더의 서버
- User Exit 프로그램을 통해 지정(OS/2 전용)

복구 실행기록 파일은 전체 데이터베이스의 백업 조작을 호출할 때마다 요약 정보를 통해 자동으로 갱신됩니다. 이 파일은 데이터베이스 구성 파일과 같은 디렉토리에 작성됩니다. 복구 실행기록 파일에 대한 자세한 정보는 57 페이지의 『복구 실행기록 파일 이해』를 참조하십시오.

UNIX 기반 시스템에서, 디스크에서 작성된 백업 이미지의 파일 이름은 마침표로 구분되는 몇 개 요소의 연결로 구성됩니다.

```
DB_alias.Type.Inst_name.NODEnnnn.CATNnnnn.timestamp.Seq_num
```

예를 들면, 다음과 같습니다.

```
STAFF.0.DB201.NODE0000.CATN0000.19950922120112.001
```

다른 플랫폼에서는 4레벨 서브디렉토리 트리가 사용됩니다.

```
DB_alias.Type\Inst_name.NODEnnnn\CATNnnnn\yyyymmdd\hhmmss.Seq_num
```

Windows NT/2000의 예:

```
SAMPLE.0\DB2\NODE0000\CATN0000\20010320\122644.001
```

데이터베이스 별명 백업 유틸리티를 호출할 때 지정한 1 - 8자의 데이터베이스 별명 이름

유형 백업 조작의 유형. 여기서 0은 전체 데이터베이스 레벨 백업을, 3은 테이블 공간 레벨 백업을, 4는 LOAD...COPY TO 명령으로 생성된 백업 이미지를 나타냅니다.

인스턴스 이름 **DB2INSTANCE** 환경 변수로부터 얻은 현재 인스턴스의 1 - 8자의 문자 이름

노드 번호 노드 번호. 파티션되지 않은 데이터베이스 시스템에서는 항상 NODE0000입니다. 파티션된 데이터베이스 시스템에서 이 번호는 NODE*xxxx*입니다. 여기서 *xxxx*는 db2nodes.cfg 파일에 있는 노드로 할당된 번호입니다.

카탈로그 노드 번호 데이터베이스에 대한 카탈로그 노드의 노드 번호. 파티션되지 않은 데이터베이스 시스템에서는 항상 CATN0000입니다. 파티션된 데이터베이스 시스템에서 이 번호는 CATN*xxxx*입니다. 여기서 *xxxx*는 db2nodes.cfg 파일에 있는 노드로 할당된 번호입니다.

시간소인 백업 조작이 수행된 날짜와 시간의 14자 문자 표시. 시간소인은 *yyyymmddhhnnss* 양식입니다.

- *yyyy*: 연도(1995 - 9999)
- *mm*: 월(01 - 12)
- *dd*: 월의 날짜(01 - 31)
- *hh*: 시간(00 - 23)
- *nn*: 분(00 - 59)
- *ss*: 초(00 - 59)

순차 번호 파일 확장자로 사용되는 3 자릿수 번호

백업 이미지가 테이프에 쓰여진 경우:

- 파일 이름은 작성되지 않지만, 위에 설명된 정보는 검증 목적으로 백업 헤더에 저장됩니다.
- 표준 운영 체제 인터페이스를 통해 테이프 장치를 사용할 수 있어야 합니다. 그러나 파티션된 대형 데이터베이스에서는 각 데이터베이스 파티션 서버 전용의 테이프 장치를 보유하는 것은 실용적이지 않습니다. 테이프 장치를 하나 이상의 TSM 서버에 연결하여 각 데이터베이스 파티션 서버가 이 테이프 장치에 액세스하도록 할 수 있습니다. TSM에 대한 자세한 정보는 479 페이지의 『부록G. Tivoli Storage Manager』을 참조하십시오.
- 파티션된 데이터베이스 시스템에서, REELlibrarian 4.2 또는 CLIO/S와 같이 가상 테이프 장치를 제공하는 제품을 사용할 수 있습니다. 이 제품을 사용하여 의사 테이프 장치를 통한 다른 노드(데이터베이스 파티션 서버)에 연결된 테이프 장치를 액세스할 수도 있습니다. 원격 테이프 장치로의 액세스는 투명하게 제공되며, 표준 운영 체제 인터페이스를 통해 의사 테이프 장치에 액세스할 수 있습니다.

백업을 사용하기 위해 필요한 특권, 권한 및 권한 부여

특권은 사용자가 데이터베이스 자원을 작성하거나 액세스할 수 있도록 합니다. 권한 레벨은 특권을 그룹화하는 방법을 제공하고, 더 높은 레벨에서 데이터베이스 관리 프로그램 유지보수와 유틸리티 조작을 제어합니다. 이들이 함께 활동하여 데이터베이스 관리 프로그램과 데이터베이스 오브젝트에 대한 액세스를 제어합니다. 사용자는 적합한 권한이 부여된, 즉 필수 특권 또는 권한을 가진 오브젝트에만 액세스할 수 있습니다.

백업 유틸리티를 사용하기 위해서는 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있어야 합니다.

백업 사용

백업하기 전에

백업할 데이터베이스에는 연결하지 마십시오. 백업 유틸리티는 지정된 데이터베이스에 대한 연결을 자동으로 설정하며 이 연결은 백업 조작 완료시 종료됩니다.

데이터베이스는 지역 또는 원격일 수 있습니다. TSM(Tivoli Storage Manager)과 같은 저장영역 관리 제품을 사용하지 않을 경우, 백업 이미지가 데이터베이스 서버에 남아 있습니다.

파티션된 데이터베이스 시스템에서 데이터베이스 파티션은 개별적으로 백업됩니다. 조작은 유틸리티를 호출하는 데이터베이스 파티션 서버에 대해 지역적 성격을 갖습니다. 그러나 인스턴스에서 데이터베이스 파티션 서버 중 하나로부터 **db2_all**을 발행하여 서버 목록에서 해당 노드 번호로 식별하는 백업 유틸리티를 호출할 수 있습니다. (사용자 테이블이 있는 노드나 데이터베이스 파티션 서버를 식별하려면 LIST NODES 명령을 사용하십시오. LIST NODES 명령에 대한 자세한 정보는 *Command Reference*를 참조하십시오.) 이렇게 할 경우, 카탈로그 노드를 먼저 백업하고 다른 데이터베이스 파티션을 백업해야 합니다. 또한 명령 센터를 사용하여 데이터베이스 파티션을 백업할 수도 있습니다. 이 방식은 롤 포워드 복구를 지원하지 않으므로, 정기적으로 노드에 상주하는 데이터베이스를 백업하십시오. 또한 백업 사본으로 `db2nodes.cfg` 파일의 사본을 보존하여 이 파일이 손상되는 경우에 대비해야 합니다.

분산 요청 시스템에서, 백업 조작은 데이터베이스 카탈로그에 저장된 메타데이터(랩퍼, 서버, 별명 등) 및 분산 요청 데이터베이스에 적용됩니다. 데이터 소스 오브젝트(테이블 및 뷰)는 이러한 오브젝트가 분산 요청 데이터베이스에 저장되지 않으면 백업되지 않습니다.

데이터베이스가 이전 릴리스의 데이터베이스 관리 프로그램으로 작성되고 데이터베이스가 이주되지 않은 경우, 데이터베이스를 이주시켜야 백업할 수 있습니다. 데이터베이스 이주에 대한 자세한 정보는 *관리 안내서: 계획* 책을 참조하십시오.

백업 호출

백업 유틸리티는 다음을 통해 호출할 수 있습니다.

- 명령행 처리기(CLP)

  다음은 CLP를 통해 발행된 BACKUP DATABASE 명령의 예입니다.

  ```
  db2 backup database sample to c:\DB2Backups
  ```

- 제어 센터에 백업 데이터베이스 노트북 또는 마법사가 있습니다. 백업 데이터베이스 노트북 또는 마법사를 열려면, 다음을 수행하십시오.

1. 제어 센터에서, 데이터베이스 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
2. 데이터베이스 폴더를 누르십시오. 기존 데이터베이스는 창의 오른쪽에 있는 분할창(내용 분할창)에 표시됩니다.
3. 내용 분할창에서 원하는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 데이터베이스 백업이나 마법사를 사용한 데이터베이스 백업을 선택하십시오. 백업 데이터베이스 노트북 또는 백업 데이터베이스 마법사가 열립니다.

제어 센터에 대한 일반 정보는 *관리 안내서*를 참조하십시오. 세부사항 정보는 제어 센터 내의 온라인 도움말 기능을 통해 제공됩니다.

- API, **sqlubkp**. API에 대한 정보는 105 페이지의 『데이터베이스 API 백업』을 참조하십시오. DB2 관리 API를 포함한 응용프로그램 작성에 대한 일반 정보는 *응용프로그램 빌드 안내서*에서 참조하십시오.

백업 정보 표시

db2ckbkp를 사용하여 기존 백업 이미지에 대한 정보를 표시할 수 있습니다. 이 유틸리티로 다음을 수행할 수 있습니다.

- 백업 이미지의 무결성을 테스트하며 복원할 수 있을지 여부를 판별합니다.
- 백업 헤더에 저장된 정보를 표시합니다.

이 유틸리티에 대한 자세한 정보는 355 페이지의 『db2ckbkp - 백업 점검』에서 참조하십시오.

테이프에 백업

데이터베이스 또는 테이블 공간을 백업할 때, 블록 *크기*와 버퍼 *크기*를 제대로 설정해야 합니다. 변수 블록 *크기*를 사용 중일 때에는 특히 그렇습니다(AIX의 예를 들면, 블록 *크기*가 0으로 설정되어 있을 경우).

백업할 때 사용될 수 있는 고정 블록 *크기*의 수에 제한사항이 있습니다. DB2는 4KB 블록으로서 백업 이미지 헤더를 작성하므로 이 제한사항이 있습니다. DB2가 지원하는 유일한 고정 블록 *크기*는 512, 1024, 2048 및 4096바이트입니다. 고

정된 블록 *크기*를 사용 중이라면 버퍼 *크기*를 지정할 수 있습니다. 그러나 고정된 블록 *크기*가 DB2가 지원하는 *크기*가 아니라면 백업이 정상적으로 완료되지 않습니다.

데이터베이스가 클 경우, 고정된 블록 *크기*를 사용하면 백업 조작에 시간이 많이 걸립니다. 변수 블록 *크기*의 사용을 고려할 수 있습니다.

주: 가변 블록 *크기*를 사용하는 것은 현재 지원되지 *않습니다*. 이 옵션을 사용해야 할 경우, 가변 블록 *크기*로 작성된 백업 이미지를 사용하여 제대로 복구할 수 있도록 하는 프로시듀어를 적절하게 테스트하였는지 확인하십시오.

변수 블록 *크기*를 사용할 때, 사용 중인 테이프 장치의 최대 한계 이하인 백업 버퍼 *크기*를 지정해야 합니다. 최적 성능의 경우, 버퍼 *크기*는 사용 중인 장치의 최대 블록 *크기* 한계와 같아야 합니다.

가변 블록 *크기*의 백업 이미지로부터 복원하면 오류가 리턴될 수도 있습니다. 이 오류가 발생하면, 적절한 블록 *크기*를 사용하여 이미지를 재작성할 필요가 있습니다. 다음은 AIX의 예입니다.

```
tcl -b 0 -Bn -f /dev/rmt0 read > backup_filename.file
dd if=backup_filename.file of=/dev/rmt0/ obs=4096 conv=sync
```

백업 이미지는 backup_filenam.file이라는 파일로 덤프됩니다. **dd** 명령은 4096의 블록 *크기*를 사용하여 테이프로 다시 이미지를 덤프합니다.

이미지가 파일로 덤프하기에 너무 큰 경우 이 방식에 문제점이 있습니다. 가능한 한 가지 솔루션은 하나의 테이프 장치에서 다른 테이프 장치로 이미지를 덤프하기 위해 **dd** 명령을 사용하는 것입니다. 이것은 이미지가 둘 이상의 테이프에 걸쳐 있지 않으면 작동합니다. 두 개의 테이프 장치를 사용할 때 **dd** 명령은 다음과 같습니다.

```
dd if=/dev/rmt1 of=/dev/rmt0 obs=4096
```

두 개의 테이프 장치 사용이 가능하지 않은 경우, **dd** 명령을 사용하여 원시 장치로 이미지를 덤프한 후 원시 장치에서 테이프로 이미지를 덤프할 수 있습니다. 이 접근의 문제점은 **dd** 명령이 *반드시* 원시 장치로 덤프되는 블록 수의 트랙을 유지해야 한다는 것입니다. 이 수는 테이프에 다시 이동할 때 지정해야 합니다. **dd** 명령이 원시 장치에서 테이프로 이미지를 덤프하기 위해 사용될 때, 명령은 원시 장

치의 전체 내용을 테이프로 덤프합니다. **dd** 유틸리티는 원시 장치가 이미지를 보유하기 위해 몇 개나 사용되는지를 판별할 수는 없습니다.

백업 유틸리티를 사용할 때 테이프 장치의 최대 블록 *크기* 한계를 알아야 합니다. 다음은 몇 가지 예입니다.

장치	접속	블록 *크기* 한계	**DB2** 버퍼 *크기* 한계 **(4KB** 페이지**)**
8mm	scsi	131,072	32
3420	s370	65,536	16
3480	s370	65,536	16
3490	s370	65,536	16
3490E	s370	65,536	16
7332(4mm)[1]	scsi	262,144	64
3490e	scsi	262,144	64
3590[2]	scsi	2,097,152	512
3570 (magstar MP)		262,144	64

주:

1. 7332는 블록 *크기* 한계를 구현하지 않습니다. 256KB는 단순히 제시된 값입니다. 블록 *크기* 한계는 상위 어댑터에 의해 부과됩니다.
2. 3590이 2MB 블록 *크기*를 지원하지만, 더 낮은 값을 가질 때(256K와 같이) 성능이 사용자 요구에 적합함을 경험할 수 있습니다.
3. 장치 한계에 대한 정보는 장치 문서를 참조하거나 장치 벤더에 문의하십시오.

Named Pipes에 백업

지금은 Unix 기반 시스템에서 지역 Named Pipes로의 데이터베이스 백업과 Named Pipes로부터의 데이터베이스 복원에 대한 지원을 사용할 수 있습니다. Named Pipe의 기록기와 판독기는 둘다 같은 머신에 있어야 합니다. 파이프가 있어야 하며 이 파이프는 지역 파일 시스템에 있어야 합니다. Named Pipe는 지역 장치로 취급되므로, 목표가 Named Pipe임을 지정할 필요는 없습니다. 다음은 AIX의 예입니다.

1. Named Pipe 작성

```
mkfifo /u/dmcinnis/mypipe
```

2. 데이터베이스 백업 조작에 대한 목표로 이 파이프를 사용하십시오.

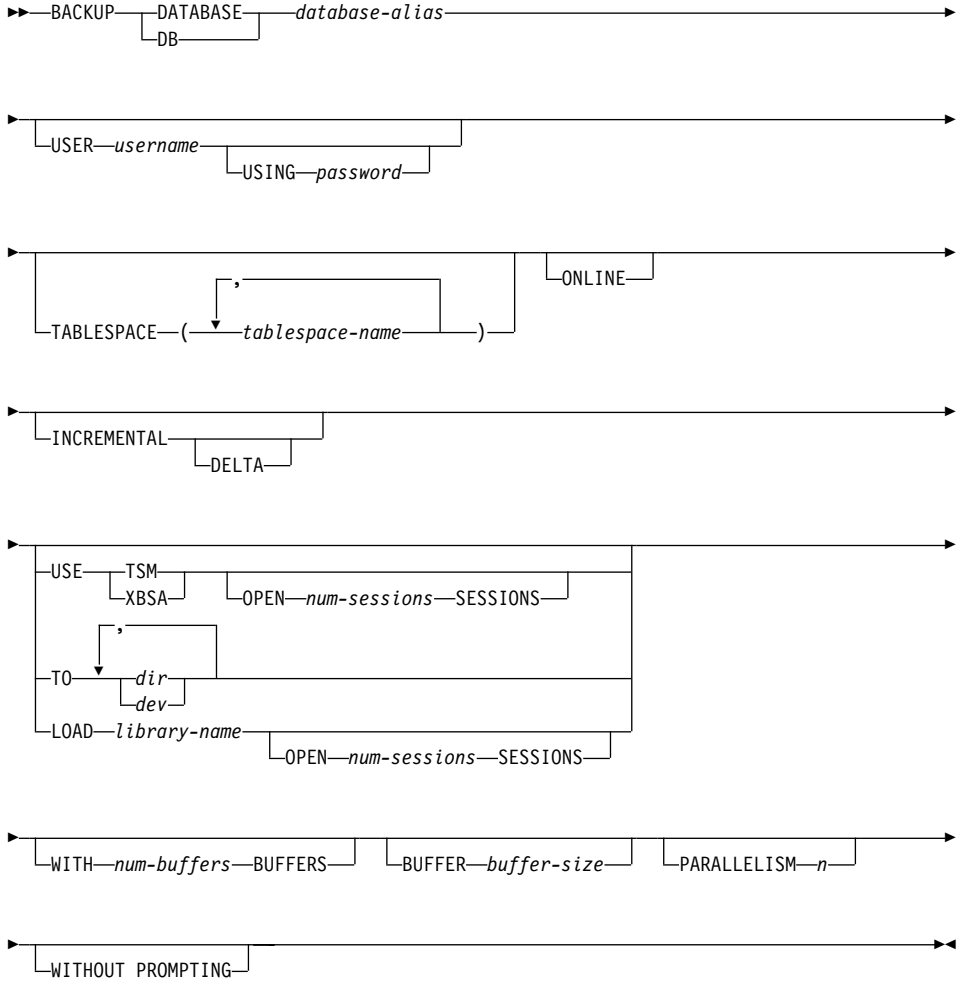
   ```
   db2 backup db sample to /u/dmcinnis/mypipe
   ```

3. 복원 유틸리티에서 이 백업 이미지를 사용할 경우, 복원 조작은 어떤 데이터도 놓치지 않도록 백업 조작 *이전에* 호출해야 합니다.

   ```
   db2 restore db sample into mynewdb from /u/dmcinnis/mypipe
   ```

BACKUP DATABASE 명령

명령 구문



명령 매개변수

DATABASE database-alias

백업할 데이터베이스의 별명을 지정합니다.

USER username

데이터베이스를 백업할 사용자 이름을 식별합니다.

USING password

사용자 이름을 인증하기 위해 사용되는 암호. 암호를 생략하면, 사용자에게 입력하라는 프롬프트가 표시됩니다.

TABLESPACE tablespace-name

백업할 테이블 공간을 지정하기 위해 사용되는 이름 목록

ONLINE

온라인 백업 지정. 기본값은 오프라인 백업입니다. *logretain* 또는 *userexit*가 사용되도록 구성된 데이터베이스에 대해서만 온라인 백업을 사용할 수 있습니다.

주: DB2 백업 유틸리티는 LOB를 포함하는 오브젝트에서 S 잠금을 요구하므로, sysibm.systables에 IX 잠금이 있을 경우 온라인 백업 조작에서 시간이 종료될 수 있습니다.

INCREMENTAL

누적(증분) 백업 이미지를 지정합니다. 증분 백업 이미지는 최근의 성공한 전체 백업 조작 이후에 변경된 모든 데이터베이스 데이터의 사본입니다.

DELTA

비누적(델타) 백업 이미지를 지정합니다. 델타 백업 이미지는 최근의 성공한 모든 유형의 백업 조작 이후에 변경된 모든 데이터베이스 데이터의 사본입니다.

USE TSM

백업에서 Tivoli Storage Manager(이전에 ADSM) 출력을 사용할 것을 지정합니다.

OPEN num-sessions SESSIONS

DB2 및 TSM 또는 다른 백업 벤더 제품 사이에 작성될 입출력 세션 수

주: 이 매개변수는 테이프, 디스크 또는 기타 지역 장치에 백업할 경우에는 효과가 없습니다.

USE XBSA

사용할 XBSA 인터페이스를 지정합니다. XBSA(Backup Services API)는 백업 및 아카이브 목적으로 데이터 저장영역 관리를 필요로 하는 응용

프로그램이나 기능에 대한 개방된 API입니다. Legato NetWorker는 현재 XBSA 인터페이스를 지원하는 저장영역 관리 프로그램입니다.

TO dir/dev

디렉토리의 목록 또는 테이프 장치 이름 디렉토리가 상주하는 전체 경로를 지정해야 합니다. 목표는 데이터베이스 서버에 상주해야 합니다. 이 매개변수는 백업 이미지를 분산시킬 수 있는 목표 디렉토리와 장치를 지정하기 위해 반복할 수 있습니다. 둘 이상의 목표를 지정할 경우(예: target1, target2, target3), target1이 첫 번째로 열립니다. 미디어 헤더 및 특수 파일(구성 파일, 테이블 공간 테이블 및 실행기록 파일)은 target1에 위치합니다. 나머지 모든 목표는 열려서 백업 조작 동안 병렬로 사용됩니다. OS/2 또는 Windows 운영 체제에서는 일반 테이프 지원이 없으므로, 각각의 테이프 장치 유형마다 고유한 장치 드라이버가 있어야 합니다. OS/2 또는 Windows 운영 체제에서 FAT 파일 시스템을 백업하려면, 사용자는 8.3 이름 지정 제한사항을 준수해야 합니다.

테이프 장치나 플로피 디스크를 사용하면 사용자 입력에 대한 프롬프트와 메시지가 생성될 수 있습니다. 유효한 응답 옵션은 다음과 같습니다.

c 계속. 경고 메시지를 생성한 장치를 계속 사용합니다(예를 들어, 새 테이프가 마운트된 경우).

d 장치 종료. 경고 메시지를 생성한 장치만 중지시킵니다(예를 들어, 테이프가 더이상 없을 경우).

t 종료. 백업 조작을 취소합니다.

테이프는 OS/2에서는 지원되지 않습니다. OS/2에서, `0` 또는 `0:`을 지정하여 백업 조작이 User Exit 프로그램을 호출하도록 할 수 있습니다(487 페이지의 『부록H. 데이터베이스 복구를 위한 User Exit』 참조). 데이터베이스는 User Exit 프로그램을 사용하는 온라인 데이터베이스 백업 조작이 시작되기 전에 Quiesce 상태가 됩니다. 백업 유틸리티는 모든 트랜잭션이 확약되거나 롤백될 때까지 기다립니다. 유틸리티가 수행 중인 동안, 새 모든 트랜잭션은 백업 조작이 완료될 때까지 기다립니다.

테이프 시스템이 백업 이미지를 고유하게 참조하는 기능을 지원하지 않을 경우, 같은 테이프에 같은 데이터베이스의 여러 백업 사본을 보존하지 않는 것이 바람직합니다.

LOAD library-name

사용될 벤더 백업 및 복원 입출력 함수를 포함하는 공유 라이브러리의 이름(OS/2 또는 Windows 운영 체제의 DLL). 전체 경로를 포함할 수 있습니다. 전체 경로를 제공하지 않으면, User Exit 프로그램이 상주하는 경로가 기본값이 됩니다.

WITH num-buffers BUFFERS

사용할 버퍼의 수. 기본값은 2입니다. 그러나 여러 위치에 대해 백업을 작성할 경우 성능을 향상시키기 위해 많은 수의 버퍼를 사용할 수도 있습니다.

BUFFER buffer-size

백업 이미지를 빌드할 때 사용되는 버퍼의 *크기*(4KB 페이지). 이 매개변수의 최소값은 8페이지이고, 기본값은 1024페이지입니다. 버퍼 *크기*로 0을 지정하면, 데이터베이스 관리 프로그램 구성 매개변수 *backbufsz*의 값이 버퍼 할당 *크기*로 사용됩니다.

블록 *크기*가 변하는 테이블을 사용할 경우, 테이프 장치에서 지원하는 범위로 버퍼 *크기*를 줄이십시오. 그렇지 않으면, 백업 조작은 성공할 수 있지만 결과 이미지가 복구 가능하지 않을 수 있습니다.

SCO UnixWare 7에서 테이프 장치를 사용 중이면, 버퍼 *크기*로 16을 지정하십시오.

대부분의 Linux 버전에서, SCSI 테이프 장치로의 백업 조작에 DB2의 기본 버퍼 *크기*를 사용할 경우, 오류 SQL2025N, 이유 코드 75가 발생합니다. Linux 내부 SCSI 버퍼의 오버플로우를 방지하려면, 다음 공식을 사용하십시오.

```
bufferpages <= ST_MAX_BUFFERS * ST_BUFFER_BLOCKS / 4
```

여기서 *bufferpages*는 *backbufsz* 또는 *restbufsz*의 값이고, ST_MAX_BUFFERS 및 ST_BUFFER_BLOCKS는 drivers/scsi 디렉토리의 Linux 커널에서 정의됩니다.

PARALLELISM n

백업 유틸리티가 병렬로 읽을 수 있는 테이블 공간 수를 판별합니다. 기본값은 1입니다.

WITHOUT PROMPTING

백업이 무인으로 수행되고, 수동으로 사용자가 개입해야 하는 조치는 오류 메시지를 리턴하도록 지정합니다.

데이터베이스 API 백업

C API 구문

```
/* File: sqlutil.h */
/* API: Backup Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlubkp (
    char * pDbAlias,
    sqluint32     BufferSize,
    sqluint32     BackupMode,
    sqluint32     BackupType,
    sqluint32     CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32     NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

일반 API 구문

```
/* File: sqlutil.h */
/* API: Backup Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgbkp (
    unsigned short DbAliasLen,
    unsigned short UserNameLen,
    unsigned short PasswordLen,
    unsigned short * pReserved1,
    char * pDbAlias,
    sqluint32 BufferSize,
    sqluint32 BackupMode,
    sqluint32 BackupType,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32 NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

API 매개변수

DbAliasLen

입력. 데이터베이스 별명의 길이(바이트)를 표시하는 부호 없는 2바이트 정수

UserNameLen

입력. 사용자 이름의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 사용자 이름이 제공되지 않았으면 0으로 설정하십시오.

PasswordLen

입력. 암호의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 암호가 제공되지 않았으면 0으로 설정하십시오.

pReserved1.

차후 사용을 위해 예약됩니다.

pDbAlias

입력. 백업할 데이터베이스의 데이터베이스 별명(시스템 데이터베이스 디렉토리에 카탈로그화된 대로)을 포함하는 문자열

BufferSize

입력. 4KB 할당 단위(페이지)의 백업 버퍼 크기. 최소값은 8단위입니다. 기본값은 1024단위입니다.

BackupMode

입력. 백업 모드를 지정합니다. 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLUB_OFFLINE

오프라인은 데이터베이스에 대한 독점적 연결을 제공합니다.

SQLUB_ONLINE

온라인은 백업 조작이 진행 중인 동안 다른 응용프로그램에서 데이터베이스 액세스를 허용합니다.

주: DB2 백업 유틸리티가 SMS LOB 오브젝트에서 S 잠금을 획득하고 다른 모든 오브젝트에서 IN 잠금을 획득하므로, sysibm.systables에 IX 잠금이 있을 경우 온라인 백업 조작에서 시간이 종료될 수 있습니다.

BackupType

입력. 취할 백업 유형을 지정합니다. 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLUB_FULL

전체(비증분) 데이터베이스 백업 조작을 지정합니다. 이 값은 차후에는 지원되지 않습니다. 전체 데이터베이스 백업 조작을 지정하려면 SQLUB_DB를 사용하십시오.

SQLUB_DB

데이터베이스에서 모든 테이블 공간의 백업을 지정합니다.

SQLUB_TABLESPACE

테이블 공간 레벨 백업 조작을 지정합니다. *pTablespaceList* 매개변수에서 백업할 테이블 공간 목록을 제공하십시오.

SQLUB_INCREMENTAL

누적(증분) 백업 이미지를 지정합니다. 증분 백업 이미지는 최근의 성공한 전체 백업 조작 이후에 변경된 모든 데이터베이스 데이터의 사본입니다.

SQLUB_DELTA

비누적(델타) 백업 이미지를 지정합니다. 델타 백업 이미지는 최근의 성공한 모든 유형의 백업 조작 이후에 변경된 모든 데이터베이스 데이터의 사본입니다.

CallerAction

입력. 취할 조치를 지정합니다. 올바른 값은 다음과 같습니다(`sqlutil`에 정의됨).

SQLUB_BACKUP

백업 조작 시작

SQLUB_NOINTERRUPT

백업 조작 시작. 백업 조작이 무인으로 수행될 것을 지정합니다. 보통 사용자 개입을 요구하는 시나리오는 먼저 호출자에게 리턴하지 않고 시도되거나 오류를 생성합니다. 예를 들어, 백업 조작에 필요한 모든 미디어가 마운트되었음을 알고 유틸리티 프롬프트를 원하지 않을 경우에 이 호출자 조치를 사용하십시오.

SQLUB_CONTINUE

사용자가 유틸리티에서 요청한 일부 조치를 수행한 후에 백업 조작을 계속합니다. (예를 들어, 새 테이프를 마운트한 후 계속합니다.)

SQLUB_TERMINATE

사용자가 유틸리티에서 요청한 일부 조치를 수행하는데 실패한 후에 백업 조작을 종료합니다.

SQLUB_DEVICE_TERMINATE

백업 유틸리티에서 사용 중인 장치 목록에서 특정의 장치를 제거합니다. 특정 매체가 가득 찰 경우, 백업 유틸리티는 호출자에게 경고를 리턴합니다(나머지 장치를 사용하여 처리를 계속하는 동안). 이 호출자 조치로 백업 유틸리티를 다시 호출하여 사용 중인 장치 목록에서 경고를 생성한 장치를 제거하십시오.

SQLUB_PARM_CHECK

백업 조작을 수행하지 않고 매개변수의 유효성을 확인하기 위해 사용됩니다. 이 옵션은 호출이 리턴된 후 데이터베이스 연결을 종료하지 않습니다. 이 호출을 제대로 리턴하고 나면, 사용자가 조치를 계속 진행하기 위해 SQLUB_CONTINUE로 호출을 발행할 것을 예상합니다.

SQLUB_PARM_CHECK_ONLY

백업 조작을 수행하지 않고 매개변수의 유효성을 확인하기 위해 사용됩니다. 이 호출이 리턴되기 전에, 이 호출이 설정한 데이터베이스 연결이 종료되고 어떤 후속 호출도 필요하지 않습니다.

pApplicationId

출력. 길이 SQLU_APPLID_LEN+1(sqlutil에 정의된)의 버퍼를 제공하십시오. API는 응용프로그램에 서비스를 제공하는 에이전트를 식별하는 문자열을 리턴합니다. 데이터베이스 모니터를 사용하여 백업 조작의 진행에 대한 정보를 확보하기 위해 사용할 수 있습니다.

pTimestamp

출력. 길이 SQLU_TIME_STAMP_LEN+1(sqlutil에 정의된)의 버퍼를 제공하십시오. API는 백업 이미지의 시간소인을 리턴합니다.

NumBuffers

입력. 사용할 백업 버퍼의 수를 지정합니다.

pTablespaceList

입력. 백업할 테이블 공간 목록. 테이블 공간 레벨 백업 조작에 대해서만 필수입니다. 120 페이지의 『데이터 구조: SQLU-TABLESPACE-BKRST-LIST』에서 자세한 내용을 참조하십시오.

pMediaTargetList

입력. 이 구조를 사용하여 호출자가 백업 조작에 대한 목적지를 지정할 수 있습니다. 제공되는 정보는 *media_type* 필드의 값에 따라 달라집니다. *media_type*에 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLU_LOCAL_MEDIA

지역 장치(테이프, 디스크 또는 디스켓의 조합). *sqlu_media_entry* 구조 목록을 제공합니다. OS/2 또는 Windows 운영 체제에서, 항목은 테이프 장치 이름이 아닌 디렉토리 경로만 될 수 있습니다.

SQLU_TSM_MEDIA

TSM. 백업 이미지에 대한 경로 지정에 *sqlu_media_entry* 구조를 사용하지 않을 경우, *sqlu_media_list_targets* 구조에서 *media* 포인터를 NULL로 초기화하십시오. DB2와 함께 제공되는 TSM 공유 라이브러리가 사용됩니다. 다른 버전의 TSM 공유 라이브러리를 원하면, SQLU_OTHER_MEDIA를 사용하여 공유 라이브러리 이름을 제공하십시오.

SQLU_OTHER_MEDIA

벤더 제품. *sqlu_vendor* 구조에서 공유 라이브러리 이름을 제공하십시오.

SQLU_USER_EXIT

User Exit. 추가 입력은 필요하지 않습니다(OS/2에서만 사용 가능).

115 페이지의 『데이터 구조: SQLU-MEDIA-LIST』에서 자세한 내용을 참조하십시오.

pUserName

입력. 연결을 시도할 때 사용할 사용자 이름을 포함하는 문자열

pPassword

입력. 사용자 이름과 함께 사용될 암호를 포함하는 문자열

pReserved2

차후 사용을 위해 예약됩니다.

VendorOptionsSize

입력. 65535바이트를 초과할 수 없는 *pVendorOptions* 필드 길이

pVendorOptions

입력. 응용프로그램에서 벤더 함수로 정보를 전달하기 위해 사용됩니다. 이 데이터 구조는 정확해야 합니다. 즉, 간접 레벨이 지원되지 않습니다. 바이트 리버설이 수행되지 않으며 이 데이터에 대한 코드 페이지도 확인되지 않습니다.

Parallelism

입력. 병렬 처리 수준(버퍼 조작자 수)

pBackupSize

출력. 백업 이미지의 크기(MB 단위). 널(NULL)로 설정할 수도 있습니다.

pReserved4

차후 사용을 위해 예약됩니다.

pReserved3

차후 사용을 위해 예약됩니다.

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

REXX API 구문

```
BACKUP DATABASE dbalias USING :value [USER username USING password]
[TABLESPACE :tablespacenames] [ONLINE]
[LOAD vendor-library [OPTIONS vendor-options] [OPEN num-sessions SESSIONS] |
 TO :target-area |
 USE TSM [OPEN num-sessions SESSIONS] |
 USER_EXIT]
[ACTION caller-action] [WITH num-buffers BUFFERS] [BUFFERSIZE buffer-size]
[PARALLELISM parallelism-degree]
```

REXX API 매개변수

dbalias

백업되는 데이터베이스의 별명

value 데이터베이스 백업 정보가 리턴되는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0 변수에 있는 요소의 수(항상 2)

XXX.1 백업 이미지의 시간소인

XXX.2 응용프로그램에 서비스를 제공하는 에이전트를 식별하는 응용프로그램 ID

username

데이터베이스를 백업할 사용자 이름을 식별합니다.

password

사용자 이름을 인증하기 위해 사용되는 암호

tablespacenames

백업할 테이블 공간 목록을 포함하는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름입니다.

XXX.0 백업할 테이블 공간 수

XXX.1 첫 번째 테이블 공간 이름

XXX.2 두 번째 테이블 공간 이름

XXX.3 기타

vendor-library

사용될 벤더 백업 및 복원 입출력 함수를 포함하는 공유 라이브러리의 이름(Windows 운영 체제 또는 OS/2의 DLL). 전체 경로가 들어 있습니다. 전체 경로를 제공하지 않으면, User Exit 프로그램이 상주하는 경로가 기본값이 됩니다.

vendor-options

벤더 함수에서 요구하는 정보

num-sessions

TSM 또는 벤더 제품에서 사용할 입출력 세션 수

target-area

지역 장치. 테이프, 디스크 또는 디스켓 조합을 허용합니다. 115 페이지의 『데이터 구조: SQLU-MEDIA-LIST』에 목록을 제공합니다. OS/2 또는 Windows 운영 체제에서, 항목은 테이프 장치 이름이 아닌 디렉토리 경로만 될 수 있습니다.

caller-action

취할 조치를 지정합니다. 유효한 값은 다음과 같습니다.

SQLUB_BACKUP

백업 조작 시작

SQLUB_NOINTERRUPT

백업 조작 시작. 백업 조작이 무인으로 수행될 것을 지정합니다. 보통 사용자 개입을 요구하는 시나리오는 먼저 호출자에게 리턴하지 않고 시도되거나 오류를 생성합니다. 예를 들어, 백업 조작에 필요한 모든 미디어가 마운트되었음을 알고 유틸리티 프롬프트를 원하지 않을 경우에 이 호출자 조치를 사용하십시오.

SQLUB_CONTINUE

사용자가 유틸리티에서 요청한 일부 조치를 수행한 후에 백업 조작을 계속합니다. (예를 들어, 새 테이프를 마운트한 후 계속합니다.)

SQLUB_TERMINATE

사용자가 유틸리티에서 요청한 일부 조치를 수행하는데 실패한 후에 백업 조작을 종료합니다.

SQLUB_DEVICE_TERMINATE

백업 유틸리티에서 사용 중인 장치 목록에서 특정 장치를 제거합니다. 특정 매체가 가득 찰 경우, 백업 유틸리티는 호출자에게 경고를 리턴합니다(나머지 장치를 사용하여 처리를 계속하는 동안). 이 호출자 조치로 백업 유틸리티를 다시 호출하여 사용 중인 장치 목록에서 경고를 생성한 장치를 제거하십시오.

SQLUB_PARM_CHECK

백업 조작을 수행하지 않고 매개변수의 유효성을 확인하기 위해 사용됩니다.

num-buffers

사용할 백업 버퍼의 수를 지정합니다.

buffer-size

4KB 할당 단위의 백업 버퍼 크기. 최소값은 8단위입니다.

parallelism-degree

병렬 처리 수준(버퍼 조작자 수)

데이터 구조: SQLU-MEDIA-LIST

이 구조는 다음과 같이 사용됩니다.

- 백업 이미지에 대한 목표 미디어 목록을 보류합니다(105 페이지의 『데이터베이스 API 백업』 참조).
- 백업 이미지에 대한 소스 미디어 목록을 보류합니다(138 페이지의 『데이터베이스 API 복원』 참조).
- 정보를 DB2 로드 유틸리티에 전달합니다.

표 1. SQLU-MEDIA-LIST 구조 내의 필드

필드 이름	데이터 유형	설명
MEDIA_TYPE	CHAR(1)	미디어 유형을 표시하는 문자
SESSIONS	INTEGER	배열에서 이 구조의 목표 필드에서 지시한 요소 수를 표시합니다.
TARGET	Union	이 필드는 세 가지 구조 유형 중 하나에 대한 포인터입니다. 지시하는 구조의 유형은 *media_type* 필드 값에 의해 판별됩니다. 이 필드에 제공할 값에 대한 자세한 정보는 해당되는 API를 참조하십시오.

표 2. SQLU-MEDIA-LIST-TARGETS 구조 내의 필드

필드 이름	데이터 유형	설명
MEDIA	포인터	*sqlu_media_entry* 구조에 대한 포인터
VENDOR	포인터	*sqlu_vendor* 구조에 대한 포인터
LOCATION	포인터	*sqlu_location_entry* 구조에 대한 포인터

표 3. SQLU-MEDIA-ENTRY 구조 내의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN	INTEGER	*media_entry* 필드의 길이. C 이외의 언어의 경우
MEDIA_ENTRY	CHAR(215)	백업 및 복원 유틸리티에서 사용되는 백업 이미지의 경로

표 4. SQLU-VENDOR 구조 내의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN1	INTEGER	*shr_lib* 필드의 길이. C 이외의 언어의 경우
SHR_LIB	CHAR(255)	데이터 저장 또는 검색을 위해 벤더에서 제공하는 공유 라이브러리의 이름
RESERVE_LEN2	INTEGER	*filename* 필드의 길이. C 이외의 언어의 경우

표 4. SQLU-VENDOR 구조 내의 필드 (계속)

필드 이름	데이터 유형	설명
FILENAME	CHAR(255)	공유 라이브러리를 사용할 경우 로드 입력 소스를 식별하기 위한 파일 이름

표 5. SQLU-LOCATION-ENTRY 구조 내의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN	INTEGER	*location_entry* 필드의 길이. C 이외의 언어의 경우
LOCATION_ENTRY	CHAR(256)	로드 유틸리티를 위한 입력 데이터 파일의 이름

*MEDIA_TYPE*을 위한 올바른 값은 다음과 같습니다(`sqlutil`에 정의됨).

SQLU_LOCAL_MEDIA
로컬 장치(테이프, 디스크 또는 디스켓)

SQLU_SERVER_LOCATION
서버 장치(테이프, 디스크 또는 디스켓: 로드 전용). *pDataFileList* 매개변수에 대해서만 지정할 수 있습니다.

SQLU_TSM_MEDIA
TSM

SQLU_OTHER_MEDIA
벤더 라이브러리

SQLU_USER_EXIT
User Exit(OS/2 전용)

SQLU_PIPE_MEDIA
Named pipe(벤더 API 전용)

SQLU_DISK_MEDIA
디스크(벤더 API 전용)

SQLU_DISKETTE_MEDIA
디스켓(벤더 API 전용)

SQLU_TAPE_MEDIA
테이프(벤더 API 전용)

언어 구문

C 구조

```
/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_media_list
{
  char            media_type;
  char            filler[3];
  sqlint32            sessions;
  union sqlu_media_list_targets target;
} sqlu_media_list;
/* ... */


/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-LIST-TARGETS */
/* ... */
union sqlu_media_list_targets
{
  struct sqlu_media_entry        *media;
  struct sqlu_vendor             *vendor;
  struct sqlu_location_entry     *location;
};
/* ... */


/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_media_entry
{
  sqluint32       reserve_len;
  char            media_entry[SQLU_DB_DIR_LEN+1];
} sqlu_media_entry;
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLU-VENDOR */
/* ... */
typedef SQL_STRUCTURE sqlu_vendor
{
  sqluint32       reserve_len1;
  char            shr_lib[SQLU_SHR_LIB_LEN+1];
  sqluint32       reserve_len2;
  char            filename[SQLU_SHR_LIB_LEN+1];
} sqlu_vendor;
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLU-LOCATION-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_location_entry
{
  sqluint32       reserve_len;
  char            location_entry[SQLU_MEDIA_LOCATION_LEN+1];
} sqlu_location_entry;
/* ... */
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQLU-MEDIA-LIST.
    05 SQL-MEDIA-TYPE         PIC X.
    05 SQL-FILLER             PIC X(3).
    05 SQL-SESSIONS           PIC S9(9) COMP-5.
    05 SQL-TARGET.
        10 SQL-MEDIA          USAGE IS POINTER.
        10 SQL-VENDOR         REDEFINES SQL-MEDIA
        10 SQL-LOCATION       REDEFINES SQL-MEDIA
        10 FILLER             REDEFINES SQL-MEDIA
*
```

```
* File: sqlutil.cbl
01 SQLU-MEDIA-ENTRY.
    05 SQL-MEDENT-LEN         PIC 9(9) COMP-5.
    05 SQL-MEDIA-ENTRY        PIC X(215).
    05 FILLER                 PIC X.
*
```

```
* File: sqlutil.cbl
01 SQLU-VENDOR.
    05 SQL-SHRLIB-LEN         PIC 9(9) COMP-5.
    05 SQL-SHR-LIB            PIC X(255).
    05 FILLER                 PIC X.
    05 SQL-FILENAME-LEN       PIC 9(9) COMP-5.
    05 SQL-FILENAME           PIC X(255).
    05 FILLER                 PIC X.
*


* File: sqlutil.cbl
01 SQLU-LOCATION-ENTRY.
    05 SQL-LOCATION-LEN       PIC 9(9) COMP-5.
    05 SQL-LOCATION-ENTRY     PIC X(255).
    05 FILLER                 PIC X.
*
```

데이터 구조: SQLU-TABLESPACE-BKRST-LIST

이 구조는 테이블 공간 이름의 목록을 제공하기 위해 사용됩니다.

표 6. SQLU-TABLESPACE-BKRST-LIST 구조 내의 필드

필드 이름	데이터 유형	설명
NUM_ENTRY	INTEGER	*tablespace* 필드에 의해 지시된 목록의 항목 수
TABLESPACE	포인터	*sqlu_tablespace_entry* 구조에 대한 포인터

표 7. SQLU-TABLESPACE-ENTRY 구조 내의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN	INTEGER	*tablespace_entry* 필드에 제공되는 문자열의 길이. C 이외의 언어의 경우
TABLESPACE_ENTRY	CHAR(19)	테이블 공간 이름

언어 구문

C 구조

```
/* File: sqlutil.h */
/* Structure: SQLU-TABLESPACE-BKRST-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long            num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLU-TABLESPACE-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32       reserve_len;
  char            tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char            filler[1];
} sqlu_tablespace_entry;
/* ... */
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQLU-TABLESPACE-BKRST-LIST.
    05 SQL-NUM-ENTRY          PIC S9(9) COMP-5.
    05 SQL-TABLESPACE         USAGE IS POINTER.
*


* File: sqlutil.cbl
01 SQLU-TABLESPACE-ENTRY.
    05 SQL-TBSP-LEN           PIC 9(9) COMP-5.
    05 SQL-TABLESPACE-ENTRY   PIC X(18).
    05 FILLER                 PIC X.
    05 SQL-FILLER             PIC X(1).
*
```

백업 세션 예

CLP 예

```
db2 backup database sample use tsm open 2 sessions with 4 buffers

db2 backup database payroll tablespace syscatspace, userspace1 to
   /dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

다음은 복구 가능한 데이터베이스에 대한 주별 증분 백업 전략의 샘플입니다. 여기에는 주별 전체 데이터베이스 백업 조작, 일별 비누적(델타) 백업 조작 및 주중 누적(증분) 백업 조작이 포함됩니다.

```
(Sun) db2 backup db kdr use tsm
(Mon) db2 backup db kdr online incremental delta use tsm
(Tue) db2 backup db kdr online incremental delta use tsm
(Wed) db2 backup db kdr online incremental use tsm
(Thu) db2 backup db kdr online incremental delta use tsm
(Fri) db2 backup db kdr online incremental delta use tsm
(Sat) db2 backup db kdr online incremental use tsm
```

DB2 명령 스크립트의 샘플과 이를 사용하는 방법에 대한 정보는 473 페이지의 『부록F. CLP 스크립트 복구』에 제공되어 있습니다.

API 예

DB2 API 및 Embedded SQL 호출을 포함하는 샘플 프로그램과 이를 사용하는 방법에 대한 정보는 421 페이지의 『부록E. 샘플 프로그램 복구』에 제공되어 있습니다.

백업 성능 최적화

백업 조작을 완료하는데 필요한 시간을 줄이려면, 다음을 수행하십시오.

- 테이블 공간 백업을 지정하십시오.

  BACKUP DATABASE 명령의 TABLESPACE 옵션을 사용하여 데이터베이스의 부분 백업(및 후속 복구)만 수행하도록 선택할 수 있습니다. 이 기능은 별도의 테이블 공간에서 테이블 데이터, 색인 및 long 필드나 대형 오브젝트(LOB) 데이터를 관리합니다.

- BACKUP DATABASE 명령의 PARALLELISM 매개변수의 값이 백업중인 테이블 공간의 수를 반영하도록 증가시키십시오.

  PARALLELISM 매개변수는 데이터베이스로부터 데이터를 읽을 때 시작된 프로세스 또는 스레드의 수를 정의합니다. 각 프로세스 또는 스레드는 특정 테이블 공간에 지정됩니다. 이 테이블 공간 백업을 완료하면, 이 매개변수는 다른 것을 요청합니다. 그러나 각 프로세스 또는 스레드에는 메모리와 CPU 오버헤드가 모두 필요하다는 점에 유의하십시오. 로드가 많은 시스템의 경우, PARALLELISM 매개변수를 기본값인 1로 두어야 합니다.

- 백업 버퍼 *크기*를 증가시키십시오.

  이상적인 백업 버퍼 *크기*는 테이블 공간 Extent *크기*의 배수입니다. 다른 확장 *크기*를 가진 다중 테이블 공간이 있으면, 최대 확장 *크기*의 배수인 값을 지정하십시오.

- 버퍼의 수를 증가시키십시오.

  다중 버퍼 및 입출력 채널을 사용하려면, 적어도 채널 수의 두 배의 버퍼를 사용하여 채널이 데이터를 대기할 필요가 없음을 확인해야 합니다.

- 다중 목표 장치를 사용하십시오.

백업 제한사항

백업 유틸리티에 다음과 같은 제한사항이 적용됩니다.

- 테이블 공간 백업 조작 및 테이블 공간 복원 조작은 비록 다른 테이블 공간에서 수행하더라도 동시에 수행할 수는 없습니다.
- 파티션된 데이터베이스 환경에서 롤 포워드 복구를 수행할 수 있으려면, 노드 목록에서 데이터베이스를 정기적으로 백업하고 시스템의 나머지 노드에 대해 적어도 하나의 백업 이미지를 가지고 있어야 합니다(노드가 해당 데이터베이스에 대한 사용자 데이터를 담고 있지 않은 경우에도). 데이터베이스에 대한 사용자 데이터가 없는 데이터베이스 파티션 서버에서 데이터베이스 파티션의 백업 이미지가 필요한 두 가지 상황이 있습니다.
  - 최종 백업을 수행한 후 데이터베이스 파티션 서버를 데이터베이스 시스템에 추가하고, 이 데이터베이스 파티션 서버에서 포워드 복구를 수행해야 합니다.

– 특정 시점 복구가 사용되어 시스템의 모든 데이터베이스 파티션이 롤 포워드 보류 상태에 있어야 합니다.

백업 문제점 해결

데이터베이스가 백업 보류 상태에 있는 경우를 제외하고, 사용 불가능 상태에 있는 데이터베이스는 백업할 수 없습니다. 모든 테이블 공간이 비정상 상태일 경우, 백업 보류 상태가 아닌 데이터베이스 또는 테이블 공간을 백업할 수 없습니다.

데이터베이스 또는 테이블 공간이 복원 조작시 시스템 손상이 발생하여 부분적으로 복원된 상태일 경우, 백업하기 전에 데이터베이스 또는 테이블 공간을 성공적으로 복원해야 합니다.

백업할 테이블 공간 목록에 임시 테이블 공간 이름이 있으면 백업 조작은 실패합니다.

백업 유틸리티는 서로 다른 데이터베이스의 백업 사본을 작성하는 여러 프로세스에 동시처리 제어를 제공합니다. 이 동시처리 제어는 모든 백업 조작이 종료할 때까지 백업 목표 장치를 열린 상태로 보존합니다. 백업 조작 중에 오류가 발생하고 열린 컨테이너를 닫을 수 없으면, 같은 드라이브를 목표로 하는 다른 백업 조작에서 액세스 오류를 받을 수 있습니다. 이와 같은 오류를 정정하려면, 오류가 발생한 백업 조작을 종료한 후 목표 장치에서 연결해제해야 합니다. 테이프로의 동시 백업 프로세스를 위해 백업 유틸리티를 사용 중인 경우, 프로세스가 동일한 테이프를 목표로 하지 않도록 해야 합니다.

제3장 데이터베이스 복원

이 절에서는 이전에 백업한 테이블 공간이나 손상 또는 훼손된 데이터베이스를 재빌드하기 위해 사용되는 DB2 UDB 복원 유틸리티에 대해 설명합니다.

이 절에는 다음 주제를 다룹니다.

복원 개요

DB2 RESTORE DATABASE 명령의 가장 간단한 양식에서는 복원할 데이터베이스의 별명 이름만 지정하면 됩니다. 예를 들면, 다음과 같습니다.

```
db2 restore db sample
```

이 예에서는 SAMPLE 데이터베이스가 존재하므로 다음과 같은 메시지가 리턴됩니다.

```
SQL2539W 경고! 백업 이미지 데이터베이스와 동일한 기존 데이터베이스에
복원중입니다.

데이터베이스 파일들이 삭제될 것입니다.
계속하시겠습니까? (y/n)
```

y를 지정하고 SAMPLE 데이터베이스에 대한 백업 이미지가 존재할 경우, 복원 조작이 제대로 완료되어야 합니다.

데이터베이스 복원 조작에는 독점적인 연결이 필요합니다. 즉, 조작이 시작될 때 데이터베이스에 대해 응용프로그램이 수행 중 상태가 될 수 있는데, 복원 유틸리티는 복원 조작이 완료할 때까지 다른 응용프로그램이 데이터베이스에 액세스할 수 없게 합니다. 테이블 공간 복원 조작은 온라인으로 수행될 수 있습니다.

복원 조작(다음으로 롤 포워드 복구 수행)이 완료될 때까지 테이블 공간은 사용할 수 없습니다.

테이블이 둘 이상의 테이블 공간에 걸쳐 있으면, 테이블 공간 세트를 함께 백업 및 복원해야 합니다.

부분 또는 부속 집합 복원 조작을 수행할 경우, 테이블 공간 레벨 백업 이미지나 전체 데이터베이스 레벨 백업 이미지를 사용하여 해당 이미지로부터 하나 이상의 테이블 공간을 선택할 수 있습니다. 백업 이미지가 작성된 시간 이후로 테이블 공간과 연관된 모든 로그 파일이 존재해야 합니다.

복원을 사용하기 위해 필요한 특권, 권한 및 권한 부여

특권은 사용자가 데이터베이스 자원을 작성하거나 액세스할 수 있도록 합니다. 권한 레벨은 특권을 그룹화하는 방법을 제공하고, 더 높은 레벨에서 데이터베이스 관리 프로그램 유지보수와 유틸리티 조작을 제어합니다. 이들이 함께 활동하여 데이터베이스 관리 프로그램과 데이터베이스 오브젝트에 대한 액세스를 제어합니다. 사용자는 적합한 권한이 부여된, 즉 필수 특권 또는 권한을 가진 오브젝트에만 액세스할 수 있습니다.

전체 데이터베이스 백업을 통해 기존 데이터베이스로 복원하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 *있어야* 합니다. *새로운* 데이터베이스로 복원하려면, SYSADM 또는 SYSCTRL 권한이 있어야 합니다.

복원 사용

복원하기 전에

기존 데이터베이스로 복원할 경우, 복원할 데이터베이스에는 연결하지 마십시오. 복원 유틸리티는 지정된 데이터베이스에 대한 연결을 자동으로 설정하며 이 연결은 복원 조작 완료시 종료됩니다. *새로운* 데이터베이스로 복원할 경우 데이터베이스를 작성하기 위해 인스턴스에 접속해야 합니다. *새로운 원격* 데이터베이스로 복원할 경우, 먼저 새로운 데이터베이스가 상주할 인스턴스에 접속해야 합니다. 그런 다음, 코드 페이지와 서버 지역을 지정하여 새로운 데이터베이스를 작성하십시오.

데이터베이스는 지역 또는 원격일 수 있습니다.

복원 호출

다음을 통해 복원 유틸리티를 호출할 수 있습니다.

- 명령행 처리기(CLP)

  다음은 CLP를 통해 발행된 RESTORE DATABASE 명령의 예입니다.

  ```
  db2 restore db sample from D:\DB2Backups taken at 20010320122644
  ```

- 제어 센터에 복원 데이터베이스 노트북 또는 마법사가 있습니다. 복원 데이터베이스 노트북 또는 마법사를 열려면, 다음을 수행하십시오.

  1. 제어 센터에서, 데이터베이스 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
  2. 데이터베이스 폴더를 누르십시오. 기존 데이터베이스는 창의 오른쪽에 있는 분할창(내용 분할창)에 표시됩니다.
  3. 내용 분할창에서 원하는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 데이터베이스 복원이나 마법사를 사용한 데이터베이스 복원을 선택하십시오. 복원 데이터베이스 노트북 또는 복원 데이터베이스 마법사가 열립니다.

  제어 센터에 대한 일반 정보는 *관리 안내서*를 참조하십시오. 세부사항 정보는 제어 센터 내의 온라인 도움말 기능을 통해 제공됩니다.

- API, **sqlrestore**. API에 대한 정보는 138 페이지의 『데이터베이스 API 복원』을 참조하십시오. DB2 관리 API를 포함한 응용프로그램 작성에 대한 일반 정보는 *응용프로그램 빌드 안내서*를 참조하십시오.

복원 조작 중 테이블 공간 컨테이너 재정의(경로 재지정 복원)

데이터베이스 백업 조작 중에, 레코드에는 백업되는 테이블 공간과 연관되는 모든 테이블 공간 컨테이너가 보존됩니다. 복원 조작 중, 백업 이미지에 나열되는 모든 컨테이너를 확인하여 존재 여부와 액세스 가능 여부를 판별합니다. 미디어 고장으로(또는 다른 이유로) 하나 이상의 컨테이너에 액세스할 수 없을 경우, 복원 조작은 실패합니다. 이 경우, 복원 조작을 성공하려면 다른 컨테이너로 경로를 재지정해야 합니다. DB2는 테이블 공간 컨테이너 추가, 변경 또는 제거를 지원합니다.

RESTORE DATABASE 명령을 호출하고 REDIRECT 매개변수를 지정하거나, 제어 센터에서 데이터베이스 복원 노트북의 컨테이너 페이지를 사용하여 테이블 공간 컨테이너를 재정의할 수 있습니다. 명령행 처리기, 명령 스크립트 또는 API를 사용하는 경로 재지정 복원 예에 대해서는 150 페이지의 『복원 세션 예』를 참조하십시오.

경로 재지정된 복원 조작 동안, 디렉토리 및 파일 컨테이너가 아직 존재하지 않는 경우 자동으로 작성됩니다. 데이터베이스 관리 프로그램은 장치 컨테이너를 자동으로 작성하지 않습니다.

컨테이너 경로 재지정은 테이블 공간 컨테이너 관리를 위한 상당한 융통성을 제공합니다. 예를 들어, SMS 테이블 공간에 대한 컨테이너 추가를 지원하지 않더라도, 경로 재지정된 복원 조작을 호출할 때 추가 컨테이너를 지정하여 이를 수행할 수 있습니다. 마찬가지로, 파일 컨테이너에서 장치 컨테이너로 DMS 테이블 공간을 이동시킬 수 있습니다.

기존 데이터베이스로 복원

전체 데이터베이스 백업 이미지를 기존의 데이터베이스로 복원할 수 있습니 백업 이미지는 별명, 데이터베이스 이름 또는 데이터베이스 시드(seed)에 있어서 기존의 데이터베이스와 다를 수 있습니다.

데이터베이스 *시드*는 데이터베이스의 주기 동안 변함없이 남아 있는 데이터베이스에 대한 고유 식별자입니다. 시드는 데이터베이스가 작성될 때 데이터베이스 관리 프로그램에 의해 지정됩니다. 백업 이미지에 다른 데이터베이스 시드가 있을 경우에도 복원 조작 다음에 변경되지 않은 상태로 남습니다. DB2는 항상 백업 이미지에서의 시드를 사용합니다.

기존 데이터베이스로 복원할 때 복원 유틸리티는 다음을 수행합니다.

- 테이블, 색인, 기존의 데이터베이스의 long 필드를 삭제하고 백업 이미지에서의 데이터로 바꿉니다.
- 복원중인 각 테이블 공간에 대해 테이블 항목을 바꿉니다.
- 손상되지 않으면 복구 실행기록 파일을 보유합니다. 복구 실행기록 파일이 손상되면, 데이터베이스 관리 프로그램이 백업 이미지로부터 파일을 복사합니다.
- 기존 데이터베이스에 대한 인증 유형을 보유합니다.
- 기존 데이터베이스에 대한 데이터베이스 디렉토리를 보유합니다. 디렉토리는 데이터베이스가 상주하는 위치와 카탈로그화되는 방법을 정의합니다.
- 데이터베이스 시드를 비교합니다. 시드가 서로 다르면 다음을 수행하십시오.
  - 기존 데이터베이스와 연관된 로그를 삭제하십시오.
  - 백업 이미지에서의 데이터베이스 구성 파일을 복사하십시오.
  - RESTORE DATABASE 명령에 NEWLOGPATH가 지정된 경우 NEWLOGPATH를 *logpath* 데이터베이스 구성 매개변수의 값으로 설정하십시오.

  데이터베이스 시드가 서로 같으면 다음을 수행하십시오.
  - 이미지가 복구 불가능한 데이터베이스 이미지이면 로그를 삭제하십시오.
  - 파일이 손상되지 않았으면, 백업 이미지에서 복사된 파일인 경우 현재 데이터베이스 구성 파일을 유지하십시오.
  - RESTORE DATABASE 명령에 NEWLOGPATH가 지정된 경우 NEWLOGPATH를 *logpath* 데이터베이스 구성 매개변수의 값으로 설정하십시오. 그렇지 않으면, 현재 로그 경로를 데이터베이스 구성 파일에 복사하십시오. 로그 경로를 검증하십시오. 데이터베이스에서 로그 경로를 사용할 수 없는 경우, 기본 로그 경로를 사용하도록 데이터베이스 구성을 변경하십시오.

새로운 데이터베이스로 복원

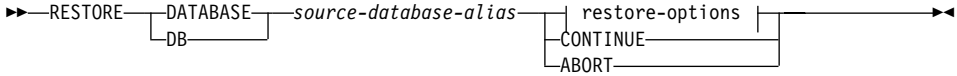
새로운 데이터베이스를 작성하여 전체 데이터베이스 백업 이미지를 이 데이터베이스로 복원할 수 있습니다. 백업 이미지의 코드 페이지와 목표 데이터베이스의 코드 페이지는 일치해야 합니다.

새로운 데이터베이스로 복원할 때 복원 유틸리티는 다음을 수행합니다.

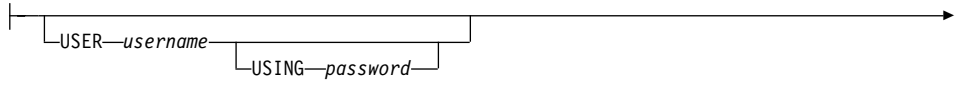
- 목표 데이터베이스 별명 매개변수에 의해 지정된 데이터베이스 별명 이름을 사용하여 새로운 데이터베이스를 작성합니다. (목표 데이터베이스 별명을 지정하지 않았으면, 복원 유틸리티는 소스 데이터베이스 별명 매개변수를 통해 지정한 것과 같은 별명이 있는 데이터베이스를 작성합니다.)
- 백업 이미지에서의 데이터베이스 구성 파일을 복원합니다.
- RESTORE DATABASE 명령에 NEWLOGPATH가 지정된 경우 NEWLOGPATH를 *logpath* 데이터베이스 구성 매개변수의 값으로 설정합니다. 로그 경로를 검증하십시오. 데이터베이스에서 로그 경로를 사용할 수 없는 경우, 기본 로그 경로를 사용하도록 데이터베이스 구성을 변경하십시오.
- 백업 이미지로부터 인증 유형을 복원합니다.
- 백업 이미지에서 데이터베이스 디렉토리로부터 주석을 복원합니다.
- 데이터베이스를 위한 복구 실행기록 파일을 복원합니다.

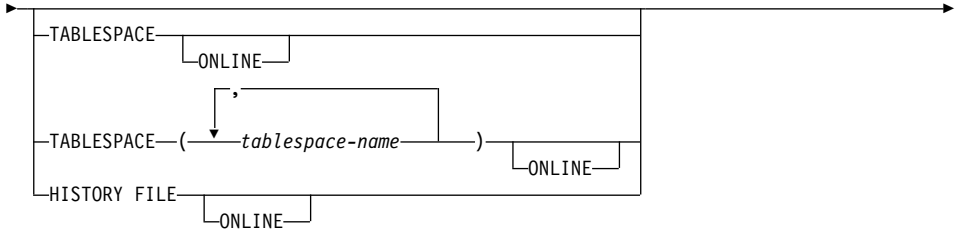
RESTORE DATABASE 명령

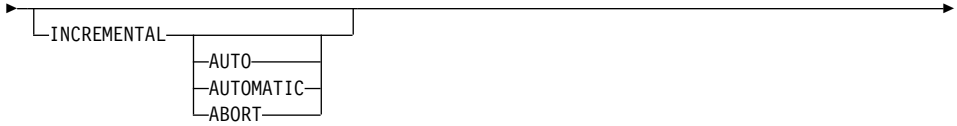
명령 구문

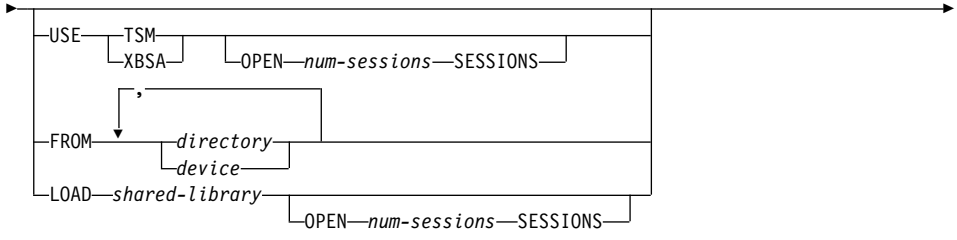


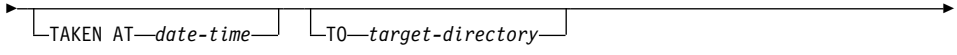
restore-options:

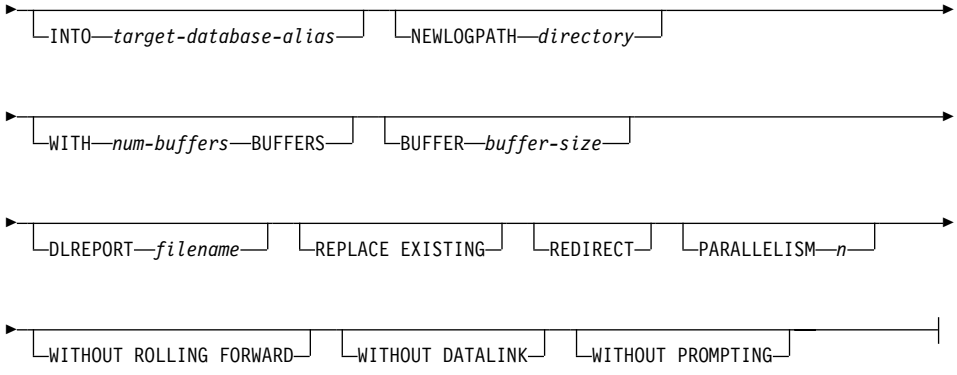












명령 매개변수

DATABASE source-database-alias

백업이 취해진 소스 데이터베이스의 별명

CONTINUE

컨테이너가 재정의되고, 경로 재지정 복원 조작의 최종 단계를 수행해야 함을 지정합니다.

ABORT

이 매개변수는

- 경로 재지정된 복원 조작을 중지시킵니다. 이는 하나 이상의 단계를 반복해야 하는 오류가 발생한 경우에 유용합니다. ABORT 옵션을 사용하여 RESTORE DATABASE를 발행하고 나면, REDIRECT 옵션을 사용하는 RESTORE DATABASE를 포함하여 경로 재지정 복원 조작의 각 단계를 반복해야 합니다.
- 완료 이전에 증분 복원 조작을 종료합니다.

USER username

데이터베이스를 복원할 사용자 이름을 식별합니다.

USING password

사용자 이름을 인증하기 위해 사용되는 암호. 암호를 생략하면, 사용자에게 입력하라는 프롬프트가 표시됩니다.

TABLESPACE tablespace-name

복원할 테이블 공간을 지정하기 위해 사용되는 이름 목록

ONLINE

테이블 공간 레벨 복원 조작을 수행할 경우에만 적용 가능한 이 키워드는 백업 이미지를 온라인에서 복원할 수 있도록 하기 위해 지정합니다. 이는 백업 이미지가 복원되는 동안 다른 에이전트가 데이터베이스에 연결할 수 있고, 지정된 테이블 공간이 복원되는 동안 다른 테이블 공간의 데이터를 사용할 수 있음을 의미합니다.

HISTORY FILE

이 키워드는 백업 이미지에서 실행기록 파일만 복원할 경우에 지정합니다.

INCREMENTAL

수동 누적 복원 조작을 지정합니다. 사용자가 각 복원 조작을 직접 발행해야 합니다.

AUTOMATIC/AUTO

자동 누적(증분) 복원 조작을 지정합니다.

USE TSM

TSM에서 관리하는 출력에서 데이터베이스를 복원할 것을 지정합니다.

OPEN num-sessions SESSIONS

TSM 또는 벤더 제품에서 사용할 입출력 세션 수를 지정합니다.

USE XBSA

사용할 XBSA 인터페이스를 지정합니다. XBSA(Backup Services API)는 백업 및 아카이브 목적으로 데이터 저장영역 관리를 필요로 하는 응용 프로그램이나 기능에 대한 개방된 API입니다. Legato NetWorker는 현재 XBSA 인터페이스를 지원하는 저장영역 관리 프로그램입니다.

FROM directory/device

백업 이미지가 상주하는 디렉토리 또는 장치. USE TSM, FROM 및 LOAD를 생략할 경우, 기본값은 현재 디렉토리입니다.

Windows 운영 체제 또는 OS/2에서, 지정된 디렉토리가 DB2 생성 디렉토리여서는 안 됩니다. 예를 들면, 다음 명령이 주어집니다.

```
db2 backup database sample to c:\backup
db2 restore database sample from c:\backup
```

DB2는 무시해야 하는 c:\backup 디렉토리 아래에 서브디렉토리를 생성합니다. 복원할 백업 이미지를 정확하게 지정하려면, TAKEN AT 매개변수를 사용하십시오. 같은 경로에 몇 개의 백업 이미지가 저장될 수도 있습니다.

몇 개의 항목을 지정하고 마지막 항목이 테이프 장치일 경우, 사용자에게 다른 테이프에 대한 프롬프트가 표시됩니다. 유효한 응답 옵션은 다음과 같습니다.

c 계속. 경고 메시지를 생성한 장치를 계속 사용합니다. (예를 들어, 새로운 테이프가 마운트된 경우 계속합니다.)

d 장치 종료. 경고 메시지를 생성한 장치만 중지시킵니다. (예를 들어, 테이프가 더이상 없을 경우에 종료합니다.)

t 종료. 사용자가 유틸리티에서 요청한 일부 조치를 수행하는데 실패한 후에 백업 조작을 취소합니다.

테이프는 OS/2에서는 지원되지 않습니다. OS/2에서, 0 또는 0:을 지정하여 복원 유틸리티가 User Exit 프로그램을 호출하도록 할 수 있습니다. (이는 User Exit 프로그램을 사용하여 데이터베이스를 백업한 경우에만 발생합니다.) User Exit 프로그램을 통해 복원할 경우, 데이터베이스 경로는 컨테이너를 찾기 위해 사용되는 단 하나의 참조사항입니다. 그러므로 데이터베이스에 대한 모든 컨테이너가 복원됩니다.

User Exit 프로그램을 사용할 때는 경로 재지정 복원이 허용되지 않습니다.

LOAD shared-library

사용될 벤더 백업 및 복원 입출력 함수를 포함하는 공유 라이브러리의 이름(Windows 운영 체제 또는 OS/2의 DLL). 이름은 전체 경로를 포함할 수 있습니다. 전체 경로를 제공하지 않으면, User Exit 프로그램이 상주하는 경로가 기본값이 됩니다.

TAKEN AT date-time

데이터베이스 백업 이미지의 시간소인. 백업 조작이 제대로 완료되면 시간소인이 표시됩니다. 이 시간소인은 백업 이미지에 대한 경로 이름의 일부입니다. *yyyymmddhhmmss* 양식으로 지정됩니다. 부분적인 시간소인을 지정할 수도 있습니다. 예를 들어, 시간소인인 19971001010101 및 19971002010101인 서로 다른 두 개의 백업 이미지가 존재할 경우, 19971002를 지정하면 시간소인이 19971002010101인 이미지가 사용됩니다. 이 매개변수의 값을 지정하지 않으면, 소스 미디어에는 하나의 백업 이미지만 있어야 합니다.

TO target-directory

목표 데이터베이스 디렉토리. 이 매개변수는 유틸리티가 기존 데이터베이스로 복원될 경우에는 무시됩니다.

주: Windows 운영 체제 또는 OS/2에서는 이 매개변수를 사용할 때 드라이브 이름만 지정하십시오. 더 긴 경로가 지정되면, 오류가 리턴됩니다.

INTO target-database-alias

목표 데이터베이스 별명. 목표 데이터베이스가 존재하지 않을 경우, 데이터베이스가 작성됩니다.

NEWLOGPATH directory

복원 조작 후에 사용 중인 로그 파일에 사용될 디렉토리의 완전한 이름. 이 매개변수의 기능은 효과가 지정하는 복원 조작으로 제한된다는 점을 제외하고는 *newlogpath* 데이터베이스 구성 매개변수와 같습니다. 백업 이미지의 로그 경로가 복원 조작 후 사용하기에는 적합하지 않을 경우(예를 들어, 경로가 더이상 유효하지 않거나 다른 데이터베이스에서 사용 중인 경우), 매개변수를 사용할 수 있습니다.

WITH num-buffers BUFFERS

사용할 버퍼의 수. 기본값은 2입니다. 그러나 여러 소스를 읽거나 PARALLELISM 값이 증가된 경우 성능을 향상시키기 위해 많은 버퍼를 사용할 수 있습니다.

BUFFER buffer-size

복원 조작에 사용되는 버퍼의 *크기*(페이지). 이 매개변수의 최소값은 8페이지이고, 기본값은 1024페이지입니다. 버퍼 *크기*로 0을 지정하면, 데이터베이스 관리 프로그램 구성 매개변수 *restbufsz*의 값이 버퍼 할당 *크기*로 사용됩니다.

복원 버퍼 *크기*는 백업 조작 중에 지정한 백업 버퍼 *크기*의 양의 정수 배수여야 합니다. 잘못된 버퍼 *크기*를 지정할 경우, 버퍼에는 승인 가능한 가장 작은 *크기*가 할당됩니다.

SCO UnixWare 7에서 테이프 장치를 사용 중이면, 버퍼 *크기*로 16을 지정하십시오.

DLREPORT filename

파일 이름. 지정할 경우에는 완전한 이름이어야 합니다. 복원 조작 중에 빠른 조정의 결과로 링크 해제되는 파일을 보고합니다. 이 옵션은 복원되는 테이블에 DATALINK 컬럼 유형과 링크된 파일이 있을 경우에만 사용됩니다.

REPLACE EXISTING

목표 데이터베이스 별명처럼 같은 별명을 가지고 있는 데이터베이스가 이미 존재할 경우, 매개변수는 복원 유틸리티가 기존 데이터베이스를 복원된 데이터베이스로 바꾸도록 지정합니다. 이는 복원 유틸리티를 호출하는 스크립트에 유용합니다. 명령행 처리기는 사용자에게 기존 데이터베이스의 삭제를 검증하도록 프롬프트를 표시하지 않기 때문입니다. WITHOUT PROMPTING 매개변수를 지정할 경우, REPLACE EXISTING을 지정할 필요는 없지만, 이 경우 정상적으로 사용자 개입을 요구하는 이벤트가 발생하면 조작이 실패하게 됩니다.

REDIRECT

경로 재지정된 복원 조작을 지정합니다. 경로 재지정된 복원 조작을 완료하려면, 이 명령 다음에 하나 이상의 SET TABLESPACE CONTAINERS 명령과, CONTINUE 옵션을 사용하는 RESTORE DATABASE 명령이 있어야 합니다.

주: 경로 재지정된 단일 복원 조작과 연관되는 모든 명령은 같은 창이나 CLP 세션에서 호출해야 합니다.

WITHOUT ROLLING FORWARD

데이터베이스를 복원한 후에 데이터베이스가 롤 포워드 보류 상태에 놓이도록 지정합니다.

복원 조작 다음에 데이터베이스가 롤 포워드 보류 상태가 되면, 먼저 168 페이지의 『ROLLFORWARD DATABASE 명령』을 호출해야 데이터베이스를 다시 사용할 수 있습니다.

WITHOUT DATALINK

DATALINK 컬럼이 있는 테이블을 DataLink_Reconcile_Pending(DRP) 상태에 놓을 것인지, 그리고 링크된 파일의 조정을 수행할 것인지를 지정합니다.

PARALLELISM n

복원 조작 동안 분산될 버퍼 조작자 수를 지정합니다. 기본값은 1입니다.

WITHOUT PROMPTING

복원 조작이 무인으로 수행됨을 지정합니다. 정상적으로 사용자 개입을 요구하는 조치가 오류 메시지를 표시합니다. 테이프나 디스켓처럼 제거 가능한 미디어 장치를 사용할 경우, 이 옵션을 지정하더라도 장치가 종료될 때 사용자에게 프롬프트가 표시됩니다.

데이터베이스 API 복원

C API 구문

```
/* File: sqlutil.h */
/* API: Restore Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurestore (
    char * pSourceDbAlias,
    char * pTargetDbAlias,
    sqluint32 BufferSize,
    sqluint32 RollforwardMode,
    sqluint32 DatalinkMode,
    sqluint32 RestoreType,
    sqluint32 RestoreMode,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    char * pTargetPath,
    sqluint32 NumBuffers,
    char * pReportFile,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaSourceList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    void * pRestoreInfo,
    void * pContainerPageList,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

일반 API 구문

```
/* File: sqlutil.h */
/* API: Restore Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgrestore (
   unsigned short SourceDbAliasLen,
   unsigned short TargetDbAliasLen,
   unsigned short TimestampLen,
   unsigned short TargetPathLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short ReportFileLen,
   unsigned short Reserved2Len,
   char * pSourceDbAlias,
   char * pTargetDbAlias,
   sqluint32 BufferSize,
   sqluint32 RollforwardMode,
   sqluint32 DatalinkMode,
   sqluint32 RestoreType,
   sqluint32 RestoreMode,
   sqluint32 CallerAction,
   char * pApplicationId,
   char * pTimestamp,
   char * pTargetPath,
   sqluint32 NumBuffers,
   char * pReportFile,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   struct sqlu_media_list * pMediaSourceList,
   char * pUserName,
   char * pPassword,
   void * pReserved2,
   sqluint32 VendorOptionsSize,
   void * pVendorOptions,
   sqluint32 Parallelism,
   unsigned short RestoreInfoSize,
   void * pRestoreInfo,
   unsigned short ContainerPageListSize,
   void * pContainerPageList,
   void * pReserved3,
   struct sqlca * pSqlca);
/* ... */
```

API 매개변수

SourceDbAliasLen

입력. 소스 데이터베이스 별명의 길이(바이트)를 표시하는 부호 없는 2바이트 정수

TargetDbAliasLen

입력. 목표 데이터베이스 별명의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 목표 데이터베이스 별명이 지정되지 않았으면 0으로 설정하십시오.

TimestampLen

입력. 시간소인의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 시간소인이 제공되지 않았으면 0으로 설정하십시오.

TargetPathLen

입력. 목표 디렉토리의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 목표 경로가 제공되지 않았으면 0으로 설정하십시오.

UserNameLen

입력. 사용자 이름의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 사용자 이름이 제공되지 않았으면 0으로 설정하십시오.

PasswordLen

입력. 암호의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 암호가 제공되지 않았으면 0으로 설정하십시오.

ReportFileLen

입력. 보고서 파일 이름의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 보고서 파일 이름이 제공되지 않았으면 0으로 설정하십시오.

Reserved2Len

입력. 예약 영역의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 0으로 설정하십시오.

pSourceDbAlias

입력. 소스 데이터베이스 백업 이미지의 데이터베이스 별명을 포함하는 문자열

pTargetDbAlias

입력. 목표 데이터베이스 별명을 포함하는 문자열. 이 매개변수의 값이 NULL이면, *pSourceDbAlias* 값이 사용됩니다.

BufferSize

입력. 4KB 할당 단위(페이지)의 백업 버퍼 크기. 최소값은 8단위입니다. 기본값은 1024단위입니다.

지정된 버퍼 크기는 백업 이미지를 생성하기 위해 사용되는 버퍼 크기와 같거나 정수 배수여야 합니다.

RollforwardMode

입력. 복원 조작 끝에서 데이터베이스를 롤 포워드 보류 상태에 놓을 것인지를 지정합니다. 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLUD_ROLLFWD

데이터베이스를 복원 후에 롤 포워드 보류 상태에 놓습니다.

SQLUD_NOROLLFWD

데이터베이스를 복원 후에 롤 포워드 보류 상태에 놓지 않습니다.

복원 조작 다음에 데이터베이스가 롤 포워드 보류 상태가 되면, 먼저 175 페이지의 『데이터베이스 API 롤 포워드』를 호출해야 데이터베이스를 다시 사용할 수 있습니다.

DatalinkMode

입력. DATALINK 컬럼이 있는 테이블을 DRP(DataLink_Reconcile _Pending) 상태에 놓을 것인지, 그리고 링크된 파일의 조정을 수행할 것인지를 지정합니다. 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLUD_DATALINK

조정 조작을 수행합니다. 정의된 DATALINK 컬럼이 있는 테이블에는 RECOVERY YES 옵션을 지정해야 합니다.

SQLUD_NODATALINK

조정 조작을 수행하지 않습니다. DATALINK 컬럼이 있는 테이블은 DRP 상태에 놓입니다. 정의된 DATALINK 컬럼이 있는 테이블에는 RECOVERY YES 옵션을 지정해야 합니다.

RestoreType

입력. 복원 조작의 유형을 지정하십시오. 올바른 값은 다음과 같습니다 (sqlutil에 정의됨).

SQLUD_FULL

백업 이미지로부터 모든 것을 복원합니다. 이는 오프라인에서 수행됩니다. 이 값은 차후에는 지원되지 않습니다. SQLUD_DB를 사용하여 전체 데이터베이스 복원 조작을 지정하십시오.

SQLUD_DB

데이터베이스에 있는 모든 테이블 공간을 복원합니다. 이는 오프라인에서 수행됩니다.

SQLUD_ONLINE_TABLESPACE

테이블 공간 레벨 백업 이미지만을 복원합니다. 이는 온라인에서 수행됩니다. 이 값은 차후에는 지원되지 않습니다. 온라인 테이블 공간 레벨 복원 조작을 지정하려면 SQLUD_TABLESPACE | SQLUD_ONLINE을 사용하십시오.

SQLUD_TABLESPACE

테이블 공간 레벨 백업 이미지만을 복원합니다. 이는 온라인이나 오프라인에서 수행될 수 있습니다.

SQLUD_HISTORY

복구 실행기록 파일만을 복원합니다.

SQLUD_INCREMENTAL

수동 누적 복원 조작을 수행합니다.

SQLUD_AUTOMATIC

자동 누적(증분) 복원 조작을 수행합니다. SQLUD_INCREMENTAL을 사용하여 지정해야 합니다(즉, SQLUD_INCREMENTAL | SQLUD_AUTOMATIC).

RestoreMode

입력. 복원 조작을 오프라인 또는 온라인에서 수행할 것인지 여부를 지정합니다. 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLUD_OFFLINE

오프라인 복원 조작을 수행합니다.

SQLUD_ONLINE

온라인 복원 조작을 수행합니다.

CallerAction

입력. 취할 조치 유형을 지정합니다. 올바른 값은 다음과 같습니다(sqlutil에 정의됨).

SQLUD_RESTORE

복원 조작을 시작합니다.

SQLUD_TERMINATE_INCREMENTAL

완료 이전에 증분 복원 조작을 종료합니다.

SQLUD_NOINTERRUPT

복원 조작을 시작합니다. 복원 조작이 무인으로 수행됨을 지정합니다. 보통 사용자 개입을 요구하는 시나리오는 먼저 호출자에게 리턴하지 않고 시도되거나 오류를 생성합니다. 예를 들어, 복원 조작에 필요한 모든 미디어가 마운트되었음을 알고 유틸리티 프롬프트를 원하지 않을 경우에 이 호출자 조치를 사용하십시오.

SQLUD_CONTINUE

경고 메시지를 생성한 장치를 계속 사용합니다. (예를 들어, 새로운 테이프가 마운트된 경우 계속합니다.)

SQLUD_TERMINATE

사용자가 유틸리티에서 요청한 일부 조치를 수행하는데 실패한 후에 백업 조작을 취소합니다.

SQLUD_DEVICE_TERMINATE

복원 유틸리티에서 사용되는 장치 목록에서 장치를 제거합니다. 장치에서 입력이 모두 소모되면, 복원 유틸리티가 호출자에게 경고를 리턴합니다. 이 호출자 조치를 사용하여 다시 복원 유틸리티를 호출하십시오. 경고를 생성한 장치는 사용 중인 장치 목록에서 제거됩니다.

SQLUD_PARM_CHECK

복원 조작을 수행하지 않고 매개변수의 유효성을 확인합니다.

SQLUD_RESTORE_STORDEF

초기 호출. 테이블 공간 컨테이너 재정의를 요청합니다.

*CallerAction*은 첫 번째 호출에서 SQLUD_RESTORE, SQLUD_NOINTERRUPT, SQLUD_RESTORE_STORDEF 또는 SQLUD_PARM_CHECK로 설정해야 합니다.

pApplicationId

출력. 길이 SQLU_APPLID_LEN+1(sqlutil에 정의된)의 버퍼를 제공하십시오. 복원 유틸리티는 응용프로그램에 서비스를 제공하는 에이전트를 식별하는 문자열을 리턴합니다. 데이터베이스 시스템 모니터 API와 함께 사용하여 응용프로그램을 모니터할 수 있습니다.

pTimestamp

입력. 문자열은 백업 이미지의 시간소인을 나타냅니다. 이 필드는 지정된 소스에 하나의 백업 이미지만 있을 경우에 선택적입니다.

pTargetPath

입력. 목표 데이터베이스 디렉토리의 완전한 이름이나 상대적 이름을 포함하는 문자열. 복원 조작 중에 새로운 데이터베이스가 작성될 경우에 사용됩니다.

NumBuffers

입력. 복원 조작시 사용되는 버퍼의 수입니다.

pReportFile

파일 이름. 지정할 경우에는 완전한 이름이어야 합니다. 복원 조작 중에 빠른 조정의 결과로 링크 해제되는 파일을 보고합니다. 이 옵션은 복원되는 테이블에 DATALINK 컬럼 유형과 링크된 파일이 있을 경우에만 사용됩니다.

pTablespaceList

복원될 하나 이상의 테이블 공간을 지정합니다. 테이블 공간 레벨 백업 이미지에서 테이블 공간이나 백업 이미지 부속 집합을 복원할 경우에 사용됩니다.

다음과 같은 제한사항이 적용됩니다.

- 데이터베이스는 복구 가능해야 합니다. 복구 가능한 데이터베이스는 *logretain* 데이터베이스 구성 매개변수를 『RECOVERY』로 설정하거나, *userexit* 데이터베이스 구성 매개변수가 사용 가능한 경우 또는 둘다입니다.
- 복원 중인 데이터베이스는 백업 이미지를 작성하기 위해 사용된 것과 같은 데이터베이스여야 합니다. 즉, 테이블 공간을 테이블 공간 복원 기능을 통해 데이터베이스에 추가할 수 없습니다.
- 이 기능은 OS/2에서 User Exit로부터 복원할 때 사용할 수 없습니다.
- 롤 포워드 유틸리티는 MPP 환경에서 복원된 테이블 공간이 같은 테이블 공간을 포함하는 다른 노드와 동기화되도록 합니다.

주: 백업된 이후에 이름이 바뀐 테이블 공간을 복원하면, 복원 유틸리티 명령을 호출할 때 새로운 테이블 공간 이름을 사용해야 합니다. 이전 이름을 사용할 경우, 테이블 공간을 찾을 수 없습니다.

pMediaSourceList

입력. 백업 이미지에 대한 소스 미디어. 115 페이지의 『데이터 구조: SQLU-MEDIA-LIST』에서 자세한 내용을 참조하십시오. 이 구조에서 호출자가 제공해야 하는 정보는 *media_type* 필드 값에 따라 다릅니다. 이 필드에 올바른 값은 다음과 같습니다(`sqlutil`에 정의됨).

SQLU_LOCAL_MEDIA

지역 장치(테이프, 디스크 또는 디스켓의 조합). *sqlu_media_entry* 구조 목록을 제공합니다. Windows 운영 체제 또는 OS/2에서, 항목은 테이프 장치 이름이 아닌 디렉토리 경로일 수 있습니다.

SQLU_TSM_MEDIA

TSM. 추가 입력이 필요없습니다. DB2와 함께 제공되는 TSM 공유 라이브러리가 사용됩니다. 다른 버전의 TSM을 원하면, `SQLU_OTHER_MEDIA`를 사용하여 공유 라이브러리 이름을 제공하십시오.

SQLU_OTHER_MEDIA

벤더 제품. *sqlu_vendor* 구조에서 공유 라이브러리 이름을 제공하십시오.

SQLU_USER_EXIT

User Exit. 추가 입력은 필요하지 않습니다(OS/2에서만 사용 가능).

pUserName

입력. 연결에 사용할 사용자 이름을 포함하는 문자열

pPassword

입력. 사용자 이름을 인증하기 위해 사용되는 암호를 포함하는 문자열

pReserved2

차후 사용을 위해 예약됩니다.

VendorOptionsSize

입력. 벤더 옵션 필드의 길이. 길이는 65535 바이트를 초과할 수 없습니다.

pVendorOptions

입력. 응용프로그램에서 벤더 함수로 정보를 전달하기 위해 벤더에서 사용됩니다. 이 데이터 구조는 정확해야 합니다. 즉, 간접 레벨이 지원되지 않습니다. 바이트 리버설이 수행되지 않으며 이 데이터에 대한 코드 페이지도 확인되지 않습니다.

Parallelism

입력. 파티션 내 병렬 처리 수준(버퍼 조작자 수)

RestoreInfoSize

차후 사용을 위해 예약됩니다.

pRestoreInfo

차후 사용을 위해 예약됩니다.

ContainerPageListSize

차후 사용을 위해 예약됩니다.

pContainerPageList

차후 사용을 위해 예약됩니다.

pReserved3

차후 사용을 위해 예약됩니다.

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

REXX API 구문

```
RESTORE DATABASE source-database-alias [USING :value] [USER username USING password]
[TABLESPACE :tablespacenames] [ONLINE | HISTORY FILE ]
[LOAD shared-library [OPTIONS vendor-options] [OPEN num-sessions SESSIONS] |
 FROM :source-area | USE TSM [OPEN num-sessions SESSIONS] | USER_EXIT]
[TAKEN AT timestamp] [TO target-directory] [INTO target-database-alias]
[ACTION caller-action] [WITH num-buffers BUFFERS] [BUFFERSIZE buffer-size]
[WITHOUT ROLLING FORWARD] [PARALLELISM parallelism-degree]
```

REXX API 매개변수

source-database-alias

데이터베이스 백업 이미지가 취해진 소스 데이터베이스의 별명

value 데이터베이스 복원 정보가 리턴되는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0 변수에 있는 요소의 수(항상 1)

XXX.1 응용프로그램에 서비스를 제공하는 에이전트를 식별하는 응용프로그램 ID

username

연결에 사용할 사용자 이름을 식별합니다.

password

사용자 이름을 인증하기 위해 사용되는 암호

tablespacenames

복원할 테이블 공간 목록을 포함하는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름입니다.

XXX.0 복원할 테이블 공간 수

XXX.1 첫 번째 테이블 공간 이름

XXX.2 두 번째 테이블 공간 이름

XXX.3 기타

HISTORY FILE

백업 이미지에서 실행기록 파일만 복원할 것을 지정합니다.

shared-library

사용될 벤더 복원 입출력 함수를 포함하는 공유 라이브러리의 이름 (Windows 운영 체제 또는 OS/2의 DLL). 전체 경로가 들어 있습니다. 전체 경로를 제공하지 않으면, User Exit 프로그램이 상주하는 경로가 기본값이 됩니다.

vendor-options

벤더 함수에서 요구하는 정보

num-sessions

TSM 또는 벤더 제품에서 사용할 입출력 세션 수

source-area

백업 이미지가 상주하는 디렉토리 또는 장치를 나타내는 복합 REXX 호스트 변수. 기본값은 현재 디렉토리입니다. Windows 운영 체제 또는 OS/2에서, 항목은 테이프 장치 이름이 아닌 디렉토리 경로일 수 있습니다.

timestamp

데이터베이스 백업 이미지의 시간소인

target-directory

목표 데이터베이스 디렉토리

target-database-alias

목표 데이터베이스 별명. 목표 데이터베이스가 존재하지 않을 경우, 데이터베이스가 작성됩니다.

caller-action

취할 조치를 지정합니다. 유효한 값은 다음과 같습니다.

SQLUD_RESTORE

복원 조작을 시작합니다.

SQLUD_NOINTERRUPT

복원 조작을 시작합니다. 복원 조작이 무인으로 수행됨을 지정합

니다. 보통 사용자 개입을 요구하는 시나리오는 먼저 호출자에게 리턴하지 않고 시도되거나 오류를 생성합니다. 예를 들어, 복원 조작에 필요한 모든 미디어가 마운트되었음을 알고 유틸리티 프롬프트를 원하지 않을 경우에 이 호출자 조치를 사용하십시오.

SQLUD_CONTINUE

경고 메시지를 생성한 장치를 계속 사용합니다. (예를 들어, 새로운 테이프가 마운트된 경우 계속합니다.)

SQLUD_TERMINATE

사용자가 유틸리티에서 요청한 일부 조치를 수행하는데 실패한 후에 백업 조작을 취소합니다.

SQLUD_DEVICE_TERMINATE

복원 유틸리티에서 사용되는 장치 목록에서 장치를 제거합니다. 장치에서 입력이 모두 소모되면, 복원 유틸리티가 호출자에게 경고를 리턴합니다. 이 호출자 조치를 사용하여 다시 복원 유틸리티를 호출하십시오. 경고를 생성한 장치는 사용 중인 장치 목록에서 제거됩니다.

SQLUD_PARM_CHECK

복원 조작을 수행하지 않고 매개변수의 유효성을 확인합니다.

SQLUD_RESTORE_STORDEF

초기 호출. 테이블 공간 컨테이너 재정의를 요청합니다.

num-buffers

사용할 백업 버퍼의 수

buffer-size

4KB 할당 단위의 백업 버퍼 크기. 최소값은 16단위입니다.

parallelism-degree

파티션 내 병렬 처리 수준(버퍼 조작자 수)

복원 세션 예

CLP 예

다음은 별명이 MYDB인 데이터베이스에 대한 일반적인 경로 재지정 복원 시나리오입니다.

1. RESTORE DATABASE 명령을 REDIRECT 옵션과 함께 발행합니다.

   ```
   db2 restore db mydb replace existing redirect
   ```

   1단계를 성공적으로 완료한 후 3단계를 완료하기 전에 다음을 발행하여 복원 조작을 중단시킬 수 있습니다.

   ```
   db2 restore db mydb abort
   ```

2. 컨테이너를 재정의해야 하는 각 테이블 공간에 대해 SET TABLESPACE CONTAINERS 명령을 발행하십시오. OS/2의 예:

   ```
   db2 set tablespace containers for 5 using
      (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

   복원된 데이터베이스의 컨테이너가 이 단계에서 지정한 컨테이너인지 검증하려면, LIST TABLESPACE CONTAINERS 명령을 발행하십시오.

3. 1 및 2단계를 성공적으로 완료한 후 다음을 발행하십시오.

   ```
   db2 restore db mydb continue
   ```

   이것은 경로 재지정된 복원 조작의 마지막 단계입니다.

4. 3단계가 실패하거나 복원 조작이 중단되었으면, 1단계에서 시작하여 경로 재지정된 복원을 재시작할 수 있습니다.

다음은 복구 가능한 데이터베이스에 대한 주별 증분 백업 전략의 샘플입니다. 여기에는 주별 전체 데이터베이스 백업 조작, 일별 비누적(델타) 백업 조작 및 주중 누적(증분) 백업 조작이 포함됩니다.

```
(Sun) backup db kdr use tsm
(Mon) backup db kdr online incremental delta use tsm
(Tue) backup db kdr online incremental delta use tsm
(Wed) backup db kdr online incremental use tsm
(Thu) backup db kdr online incremental delta use tsm
(Fri) backup db kdr online incremental delta use tsm
(Sat) backup db kdr online incremental use tsm
```

금요일 아침에 작성된 이미지의 자동 데이터베이스 백업의 경우, 다음을 발행하십시오.

```
restore db kdr incremental automatic taken at (Thu)
```

금요일 아침에 작성된 이미지의 수동 데이터베이스 백업의 경우, 다음을 발행하십시오.

```
restore db kdr incremental taken at (Thu)
restore db kdr incremental taken at (Sun)
restore db kdr incremental taken at (Wed)
restore db kdr incremental taken at (Thu)
```

DB2 명령 스크립트의 샘플과 이를 사용하는 방법에 대한 정보는 473 페이지의 『부록F. CLP 스크립트 복구』에 제공되어 있습니다.

API 예

DB2 API와 Embedded SQL 호출을 포함하는 샘플 프로그램과 사용 방법에 대한 정보는 421 페이지의 『부록E. 샘플 프로그램 복구』에 제공되어 있습니다.

복원 성능 최적화

복원 조작을 완료하는데 필요한 시간을 줄이려면, 다음을 수행하십시오.

- 복원 버퍼 *크기*를 증가시키십시오.

  복원 버퍼 *크기*는 백업 조작 중에 지정한 백업 버퍼 *크기*의 양의 정수 배수여야 합니다. 잘못된 버퍼 *크기*를 지정할 경우, 할당된 버퍼는 승인 가능한 가장 작은 *크기*가 됩니다.

- 버퍼의 수를 증가시키십시오.

  지정하는 값은 백업 버퍼용으로 지정한 다중 페이지 수이어야 합니다. 최소 페이지 수는 16입니다.

복원 제한사항

복원 유틸리티에 다음과 같은 제한사항이 적용됩니다.

- DB2 백업 유틸리티를 사용하여 이전에 데이터베이스를 백업한 경우에만 복원 유틸리티를 사용할 수 있습니다.

- DB2 제어 센터를 사용 중이면, 이전 DB2 버전으로 작성된 백업 이미지를 복원할 수 없습니다.
- 롤 포워드 프로세스가 실행 중일 때는 데이터베이스 복원 조작을 시작할 수 없습니다.
- 테이블 공간이 현재 존재하고 동일한 테이블 공간일 경우에만 테이블 공간을 복원할 수 있습니다. 『동일』은 테이블 공간이 삭제되지 않은 상태에서 백업 및 복원 조작 사이에 다시 작성되었음을 의미합니다.
- 테이블 공간 레벨 백업을 새로운 데이터베이스에 복원할 수 없습니다.
- 시스템 카탈로그 테이블과 연관한 온라인 테이블 공간 레벨 복원 조작을 수행할 수 있습니다.
- OS/2에서, User Exit 프로그램을 호출할 때에는 부분 또는 부속 집합 복원 조작이 가능하지 않습니다.
- 복원 중인 데이터베이스의 코드 페이지가 응용프로그램에 대해 사용할 수 있는 코드 페이지와 일치하지 않거나, 데이터베이스 관리 프로그램이 데이터베이스 코드 페이지에서 응용프로그램에 대해 사용할 수 있는 코드 페이지로의 코드 페이지 변환을 지원하지 않으면, 복원되는 데이터베이스는 사용할 수 없게 됩니다.

복원 문제점 해결

서로 다른 시스템에서 백업 이미지를 새로운 데이터베이스로 복원하려고 할 때 SQL0970N이 리턴되고, 원래 데이터베이스에 절대 경로를 사용하여 정의한 테이블 공간이 있을 경우, 경로 재지정 복원 조작을 사용하여 새로운 시스템에서 테이블 공간 컨테이너를 정의하십시오. 복원 유틸리티는 새로운 시스템에 존재하지 않는 컨테이너와 절대 경로를 사용하려고 합니다.

데이터베이스 복원 조작 동안 시스템 실패가 발생할 경우, 복원 유틸리티를 다시 호출하여 복원 조작을 제대로 완료할 때까지 데이터베이스에 연결할 수 없습니다. 테이블 공간 복원 조작 동안 시스템 실패가 발생하면, 복원 중인 테이블 공간만 사용할 수 없습니다. 데이터베이스의 다른 테이블 공간은 사용할 수 있습니다.

제4장 롤 포워드 복구

이 절에서는 데이터베이스 복구 로그 파일에 기록된 트랜잭션을 적용하여 데이터베이스를 복구하기 위해 사용되는 DB2 UDB 롤 포워드 유틸리티에 대해 설명합니다.

이 절에는 다음 주제를 다룹니다.

롤 포워드 개요

DB2 ROLLFORWARD DATABASE 명령의 가장 간단한 양식에서는 롤 포워드 복구할 데이터베이스의 별명 이름만 지정하면 됩니다. 예를 들면, 다음과 같습니다.

```
db2 rollforward db sample stop
```

예에서, 명령은 다음을 리턴합니다.

```
                                   롤 포워드 상태
 입력 데이터베이스 별명                = sample
 리턴된 노드 수                       = 1
 노드 번호                            = 0
 롤 포워드 상태                       = not pending
 읽을 다음 로그 파일                  =
 처리된 로그 파일                     =  -
 마지막 확약 트랜잭션                 = 2001-03-11-02.39.48.000000
DB20000I  ROLLFORWARD 명령이 정상적으로 완료되었습니다.
```

이들 필드에 대한 설명은 168 페이지의 『ROLLFORWARD DATABASE 명령』을 참조하십시오.

롤 포워드 복구에 대한 일반적인 접근 방식에는 다음이 포함됩니다.

1. STOP 옵션 없이 롤 포워드 유틸리티 호출
2. QUERY STATUS 옵션으로 롤 포워드 유틸리티 호출

   로그 끝까지의 복구를 지정할 경우, QUERY STATUS 옵션은 리턴된 특정 시점이 예상한 시기보다 이전일 경우에 하나 이상의 로그 파일이 누락되었음을 나타냅니다.

   특정 시점 복구를 지정할 경우, QUERY STATUS 옵션은 롤 포워드 조작이 해당되는 시점에 완료되었는지 확인하는데 도움이 됩니다.
3. STOP STATUS 옵션으로 롤 포워드 유틸리티를 호출합니다. 조작이 중지된 후에는 추가 변경사항을 롤 포워드할 수 없습니다.

데이터베이스는 롤 포워드되기 전에 복원되어야 하지만(복원 유틸리티를 사용하여), 테이블 공간은 그렇지 않습니다. 테이블 공간은 임시로 롤 포워드 보류 상태에 있게 되지만, 이를 실행 취소하기 위해 복원 조작이 필요하지는 않습니다(예를 들어, 정전된 후).

롤 포워드 유틸리티가 호출될 경우:

- 데이터베이스가 롤 포워드 보류 상태에 있을 경우, 데이터베이스는 롤 포워드됩니다. 테이블 공간이 롤 포워드 보류 상태에 있을 경우에는 데이터베이스 롤 포워드 조작이 테이블 공간의 롤 포워드를 완료한 후에 다시 롤 포워드 유틸리티를 호출해야 합니다.

- 데이터베이스가 롤 포워드 보류 상태에 있지 *않지만*, 데이터베이스의 테이블 공간이 롤 포워드 보류 상태에 *있을* 경우:
  - 테이블 공간 목록을 지정할 경우, 테이블 공간만 롤 포워드됩니다.
  - 테이블 공간 목록을 지정하지 않은 경우, 롤 포워드 보류 상태에 있는 모든 테이블 공간은 롤 포워드됩니다.

데이터베이스 롤 포워드 조작은 오프라인으로 수행됩니다. 데이터베이스는 롤 포워드 조작이 제대로 완료될 때까지 사용할 수 없으므로, 유틸리티를 호출할 때 STOP 옵션을 지정하지 않았으면 조작을 완료할 수 없습니다.

테이블 공간 롤 포워드 조작을 오프라인으로 수행할 수 있습니다. 롤 포워드 조작이 제대로 완료될 때까지 데이터베이스를 사용할 수 없습니다. 이는 로그 끝에 도달하거나 유틸리티를 호출할 때 STOP 옵션을 지정한 경우에 발생합니다.

SYSCATSPACE가 포함되지 않으면 테이블 공간에서 *온라인* 롤 포워드 조작을 수행할 수 있습니다. 테이블 공간에서 온라인 롤 포워드를 실행하는 경우, 사용할 수 없습니다. 그러나 데이터베이스의 다른 테이블 공간은 *그렇지 않습니다*.

처음으로 데이터베이스를 작성할 경우, 순환 로깅에 대해서만 가능합니다. 이는 로그가 저장되거나 아카이브되기 보다는 재사용됨을 의미합니다. 순환 로그가 활성화되면 롤 포워드 복구는 불가능합니다. 응급 복구 또는 버전 복구만이 수행할 수 있습니다(35 페이지의 『복구 로그 이해』 참조). 아카이브된 로그는 백업 후에 발생하는 데이터베이스에 대한 변경사항을 문서화합니다. *logretain* 데이터베이스 구성 매개변수를 RECOVERY로 설정하거나, *userexit* 데이터베이스 구성 매개변수를 YES로 설정하거나, 또는 둘다를 설정하여 로그 아카이브(및 롤 포워드 복구)를 사용합니다. 두 매개변수 모두 기본값은 NO입니다. 초기에는 데이터베이스를 복구하기 위해 사용할 수 있는 백업 이미지가 없기 때문입니다. 두 매개변수 값 중 하나 또는 둘다 변경할 경우, 데이터베이스는 백업 보류 상태가 되어, 다시 사용하려면 먼저 데이터베이스의 오프라인 백업을 수행해야 합니다.

로깅과 연관된 데이터베이스 구성 매개변수에 대한 자세한 정보는 43 페이지의 『데이터베이스 로깅을 위한 구성 매개변수』를 참조하십시오.

롤 포워드를 사용하기 위해 필요한 특권, 권한 및 권한 부여

특권은 사용자가 데이터베이스 자원을 작성하거나 액세스할 수 있도록 합니다. 권한 레벨은 특권을 그룹화하는 방법을 제공하고, 더 높은 레벨에서 데이터베이스 관리 프로그램 유지보수와 유틸리티 조작을 제어합니다. 이들이 함께 활동하여 데이터베이스 관리 프로그램과 데이터베이스 오브젝트에 대한 액세스를 제어합니다. 사용자는 적합한 권한이 부여된, 즉 필수 특권 또는 권한을 가진 오브젝트에만 액세스할 수 있습니다.

롤 포워드 유틸리티를 사용하기 위해서는 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있어야 합니다.

롤 포워드 사용

롤 포워드하기 전에

롤 포워드 복구할 데이터베이스에는 연결하지 마십시오. 롤 포워드 유틸리티는 지정된 데이터베이스에 대한 연결을 자동으로 설정하며 이 연결은 롤 포워드 조작 완료시 종료됩니다.

데이터베이스는 지역 또는 원격일 수 있습니다.

롤 포워드 호출

롤 포워드 유틸리티는 다음을 통해 호출할 수 있습니다.

- 명령행 처리기(CLP)

  다음은 CLP를 통해 발행된 ROLLFORWARD DATABASE 명령의 예입니다.

  ```
  db2 rollforward db sample to end of logs and stop
  ```

- 제어 센터에 롤 포워드 데이터베이스 노트북이 있습니다. 롤 포워드 데이터베이스 노트북을 열려면, 다음을 수행하십시오.

  1. 제어 센터에서, 데이터베이스 폴더를 찾을 때까지 오브젝트 트리를 확장하십시오.
  2. 데이터베이스 폴더를 누르십시오. 기존 데이터베이스는 창의 오른쪽에 있는 분할창(내용 분할창)에 표시됩니다.

3. 내용 분할창에서 원하는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 팝업 메뉴에서 롤 포워드를 선택하십시오. 롤 포워드 데이터베이스 노트북이 열립니다.

제어 센터에 대한 일반 정보는 *관리 안내서*를 참조하십시오. 세부사항 정보는 제어 센터 내의 온라인 도움말 기능을 통해 제공됩니다.

- API, **sqluroll**. API에 대한 정보는 175 페이지의 『데이터베이스 API 롤 포워드』를 참조하십시오. DB2 관리 API를 포함한 응용프로그램 작성에 대한 일반 정보는 *응용프로그램 빌드 안내서*를 참조하십시오.

파티션된 데이터베이스 환경에서 롤 포워드 유틸리티는 데이터베이스의 카탈로그 노드에서 호출되어야 합니다.

테이블 공간의 롤 포워드 변경사항

데이터베이스에서 포워드 복구가 작동 가능한 경우, 전체 데이터베이스 대신에 테이블 공간을 백업, 복원 및 롤 포워드하는 옵션이 있습니다. 이것으로 시간이 절약되므로 사용자는 각 테이블 공간에 대해 복구 방침을 구현하려고 합니다. 전체 데이터베이스를 복구하는 것보다 데이터베이스의 일부를 복구하는데 시간이 덜 소요됩니다. 예를 들어, 디스크가 불량인데 하나의 테이블 공간만 들어 있는 경우, 손상된 테이블 공간에 시스템 카탈로그 테이블이 없으면 그 테이블 공간은 전체 데이터베이스를 복구하지 않고, 나머지 데이터베이스에 대한 사용자 액세스에 영향을 주지 않으면서 복원 및 롤 포워드될 수 있습니다. 이러한 상황에서는 데이터베이스에 연결할 수 없습니다. (시스템 카탈로그 테이블 공간을 포함하는 테이블 공간 레벨 백업 이미지가 사용 가능할 경우, 시스템 카탈로그 테이블 공간을 독립적으로 복원할 수 있습니다.) 또한 테이블 공간 레벨 백업을 사용하면 데이터베이스의 중요 부분을 다른 부분보다 자주 백업할 수 있으므로, 전체 데이터베이스를 백업하는 것보다 시간이 적게 소요됩니다.

테이블 공간이 복원되고 나면 항상 롤 포워드 보류 상태에 놓입니다. 테이블 공간을 사용할 수 있으려면, 테이블 공간에서 롤 포워드 복구를 실행해야 합니다. 대부분의 경우, 로그 끝까지 롤 포워드하거나 특정 시점까지의 롤 포워드를 하는 옵션이 있습니다. 그러나 시스템 카탈로그 테이블을 포함하는 테이블 공간을 특정 시

점으로 롤 포워드할 수 없습니다. 이 테이블 공간은 로그 끝까지 롤 포워드되어야 하며 데이터베이스의 모든 테이블 공간이 일관성 있는 상태로 남아 있는지 확인합니다.

테이블 공간을 롤 포워드하기 전에 LIST TABLESPACES SHOW DETAIL 명령을 호출하십시오. 이 명령으로 *최소 복구 시간*을 리턴하고, 테이블 공간이 가장 빠른 특정 시점으로 롤 포워드되게 할 수 있습니다. DDL문이 테이블 공간에 대해 또는 테이블 공간의 테이블에 대해 수행되는 경우 최소 복구 시간이 갱신됩니다. 테이블 공간은 적어도 최소 복구 시간까지 롤 포워드되어 시스템 카탈로그 테이블의 정보와 동기화되어야 합니다. 여러 개의 테이블 공간을 복구할 경우, 테이블 공간은 최소한 복구되는 모든 테이블 공간의 최상위 최소 복구 시간까지 롤 포워드되어야 합니다. 파티션된 데이터베이스 환경에서는 모든 파티션에서 LIST TABLESPACES SHOW DETAIL 명령을 발행하십시오. 테이블 공간은 모든 파티션에서 최소한 모든 테이블 공간의 최상위 최소 복구 시간까지 롤 포워드되어야 합니다.

테이블 공간을 특정 시점으로 롤 포워드 중이며 테이블이 여러 테이블 공간에 포함되어 있을 경우, 이러한 모든 테이블 공간을 동시에 롤 포워드해야 합니다. 예를 들어, 테이블 데이터가 하나의 테이블 공간에 포함되고 테이블에 대한 색인이 다른 테이블 공간에 포함된 경우, 두 테이블 공간 모두 동일한 특정 시점으로 동시에 롤 포워드해야 합니다.

테이블에 있는 데이터 및 긴 오브젝트는 다른 테이블 공간에 있고 테이블이 재인식되면, 데이터 및 긴 오브젝트 모두에 대한 테이블 공간은 복원과 롤 포워드가 함께 이루어져야 합니다. 테이블이 재인식된 후 손상된 테이블 공간을 백업해야 합니다.

특정 시점으로 테이블 공간을 롤 포워드하고 테이블 공간의 테이블이 다음 중 하나에 속할 경우, 테이블 공간을 동일한 특정 시점으로 같이 롤 포워드해야 합니다.

- 다른 테이블 공간에 있는 요약 테이블에 대한 기본 테이블
- 다른 테이블 공간의 테이블에 대한 요약 테이블

이렇게 하지 않으면, 요약 테이블은 롤 포워드 조작 끝에서 점검 보류 상태가 됩니다.

특정 시점으로 테이블 공간을 롤 포워드하고 테이블 공간의 테이블이 참조 무결성 관계를 다른 테이블 공간에 포함된 테이블에서 간섭하는 경우, 특정 시점까지 두 테이블 공간을 동시에 롤 포워드해야 합니다. 그렇지 않은 경우, 두 테이블 공간은 특정 시점 롤 포워드 조작의 끝에서 점검 보류 상태에 있게 됩니다. 두 테이블 공간을 동시에 롤 포워드하는 경우, 제한조건은 특정 시점 롤 포워드 조작의 끝에서 사용 중으로 남게 됩니다.

특정 시점 테이블 공간 롤 포워드 조작으로 트랜잭션이 다른 테이블 공간에서 구간 복원되지 않고 확약되지 않는다는 것을 확인하십시오. 다음 경우에 발생할 수 있습니다.

- 특정 시점 롤 포워드는 트랜잭션에 의해 갱신된 테이블 공간의 부속 집합에서 실행되고, 시점은 트랜잭션이 확약되기 이전 시간입니다.
- 특정 시점으로 롤 포워드되는 테이블 공간에 포함된 테이블은 연관된 트리거를 가지고 있거나, 롤 포워드되는 테이블 공간을 제외한 테이블 공간에 영향을 주는 트리거에 의해 갱신됩니다.

해결책은 이러한 상황을 방지할 적절한 특정 시점을 찾는 것입니다.

QUIESCE TABLESPACES FOR TABLE 명령을 발행하여 테이블 공간을 롤 포워드하는 transaction-consistent 특정 시점을 작성할 수 있습니다. Quiesce 요청(공유, 갱신 가능 또는 독점 모드에서)은 테이블 공간에 대해 수행 중인 모든 트랜잭션이 완료되기를 기다리며(잠금을 통해) 새로운 요청을 블로킹합니다. quiesce 요청이 권한 부여되면, 테이블 공간은 일관성있는 상태가 됩니다. 롤 포워드 조작을 중지할 적절한 시간을 판별하려면, 복구 실행기록 파일에서 quiesce 지점을 찾은 다음 최소 복구 시간 후에 발생하는지 확인하십시오.

테이블 공간 특정 시점 롤 포워드 조작이 완료되고 나면, 테이블 공간은 백업 보류 상태에 놓입니다. 사용자가 롤 포워드한 시간과 현재 시간 사이의 모든 갱신사항이 제거되지 않았기 때문에 테이블 공간을 백업해야 합니다. 이전 데이터베이스 또는 테이블 공간 레벨 백업 이미지에서 현재 시간으로 테이블 공간을 롤 포워드할 수 없습니다. 다음 예에서 테이블 공간 레벨 백업 이미지가 필요한 이유와 사용 방법을 표시합니다. (테이블 공간을 사용 가능하게 하려면, 전체 데이터베이스,

백업 보류 상태의 테이블 공간 또는 백업 보류 상태의 테이블 공간을 포함하는 테이블 공간 세트를 백업할 수 있습니다.)

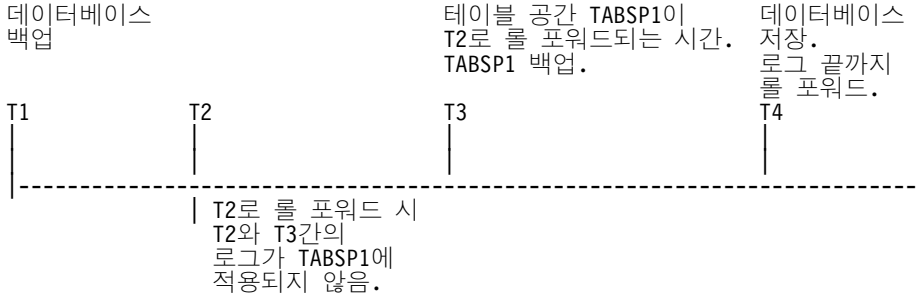


그림 17. 테이블 공간 백업 요구

위의 예에서, 데이터베이스는 T1에서 백업됩니다. 그런 다음 T3에서 테이블 공간 TABSP1이 특정 시점(T2)으로 롤 포워드됩니다. 테이블 공간은 T3 시간 이후에 백업됩니다. 테이블 공간이 백업 보류 상태이므로, 이 백업 조작은 필수입니다. 테이블 공간 백업 이미지의 시간소인은 T3 이후이지만, 테이블 공간은 T2에서입니다. T2와 T3간의 로그 레코드는 TABSP1에 적용되지 않습니다. T4에서 데이터베이스는 T1에서 작성한 백업 이미지를 사용하여 복원된 다음 로그 끝까지 롤 포워드됩니다. 데이터베이스 관리 프로그램은 T2와 T3 사이의 변경사항 로그를 테이블 공간에 적용하지 않고 T3과 T4 사이에 TABSP1에서 조작이 수행되었다고 가정하므로, TABSP1 테이블 공간은 T3시에 복원 보류 상태에 놓입니다. 변경사항 로그가 실제로 데이터베이스에 대해 롤 포워드 조작의 일부로 적용되었으면, 이러한 가정은 잘못된 것입니다. 테이블 공간이 특정 시점으로 롤 포워드된 후에 이루어져야 하는 테이블 공간 레벨 백업을 통해 이전의 특정 시점 롤 포워드 조작(예에서 T3)을 지나 테이블 공간을 롤 포워드할 수 있습니다.

테이블 공간 TABSP1을 T4로 복구한다고 가정하면, T3 이후에 이루어진 백업 이미지(필요한 백업 또는 다음 백업)로부터 테이블 공간을 복원한 다음 로그 끝까지 TABSP1을 롤 포워드합니다.

위의 예에서, 데이터베이스를 T4 시간으로 복원하는 가장 효과적인 방법은 다음 순서에 따라 필요한 단계를 실행하는 것입니다.

1. 데이터베이스 복원
2. 테이블 공간 복원

3. 데이터베이스 롤 포워드
4. 테이블 공간 롤 포워드

데이터베이스를 롤 포워드하기 전에 테이블 공간을 복원하므로, 데이터베이스를 롤 포워드할 때 로그 레코드를 테이블 공간에 적용하기 위해 자원을 사용할 수 없습니다.

T3 시간 이후인 TABSP1 백업 이미지를 찾을 수 없거나 TABSP1을 T3(또는 이전)으로 복원하려면, 다음을 수행할 수 있습니다.

- 테이블 공간을 T3으로 롤 포워드하십시오. 테이블 공간이 데이터베이스 백업 이미지에서 복원되었으므로, 테이블 공간을 다시 복원할 필요가 없습니다.
- T1 시간의 데이터베이스 백업을 사용하여 테이블 공간을 다시 복원한 다음 T3 시간 이전의 시간으로 테이블 공간을 롤 포워드하십시오.
- 테이블 공간을 삭제하십시오.

파티션된 데이터베이스 환경에서

- 동시에 같은 특정 시점으로 테이블 공간의 모든 부분을 롤 포워드해야 합니다. 이렇게 하여 테이블 공간을 데이터베이스 파티션 전반에 걸쳐 일관성있게 합니다.
- 일부 데이터베이스 파티션이 롤 포워드 보류 상태에 있고 다른 데이터베이스 파티션에서 일부 테이블 공간이 롤 포워드 보류 상태에 있는 경우(데이터베이스 파티션은 아님), 먼저 데이터베이스 파티션을 롤 포워드하고 테이블 공간을 롤 포워드해야 합니다.
- 테이블 공간을 로그 끝까지 롤 포워드하려면, 해당 테이블 공간을 각 데이터베이스 파티션에서 복원할 필요가 없습니다. 단지 복구해야 하는 데이터베이스 파티션에서만 복원하면 됩니다. 그러나 특정 시점으로 테이블 공간을 롤 포워드하려면, 각 데이터베이스 파티션에서 테이블 공간을 복원해야 합니다.

삭제된 테이블 복구

간혹 계속 필요한 데이터가 있는 테이블을 삭제할 경우가 있습니다. 그러한 경우, 중요한 테이블이 테이블 삭제 조작 다음에 복구 가능하도록 만들어야 합니다.

데이터베이스 복원 조작을 호출하여 테이블 데이터를 복구할 수 있습니다. 복원 조작 다음에는 테이블을 삭제하기 전에 특정 시점으로 데이터베이스 롤 포워드 조작을 수행합니다. 이렇게 하면 데이터베이스가 클 경우 시간을 낭비할 수 있고 복구하는 동안 데이터를 사용할 수 없습니다.

DB2의 삭제된 테이블 복구 기능은 테이블 공간 레벨 복원 및 롤 포워드 조작을 사용하여 삭제된 테이블 데이터를 복구할 수 있게 합니다. 이는 데이터베이스 레벨 복구보다 빨라서 사용자가 데이터베이스를 사용할 수 있는 상태로 유지됩니다.

삭제된 테이블을 복구 가능하게 하기 위해, 테이블이 있는 테이블 공간에서는 DROPPED TABLE RECOVERY 옵션이 활성화되어야 합니다. 이는 테이블 공간을 작성하는 동안, 또는 ALTER TABLESPACE문을 호출하여 수행할 수 있습니다(*SQL 참조서* 참조). DROPPED TABLE RECOVERY 옵션은 테이블 공간 고유의 것이며, 일반 테이블 공간에는 제한됩니다. 테이블 공간이 삭제된 테이블 복구에 대해 사용 가능한지 판별하려면, SYSCAT.TABLESPACES 카탈로그 테이블에서 DROP_RECOVERY 컬럼을 조회하면 됩니다.

테이블 공간이 삭제된 테이블 복구에 대해 사용 가능한 테이블에서 DROP TABLE 문을 수행할 경우, 로그 파일에 추가 항목이 항목(삭제된 테이블을 식별하는)이 작성됩니다. 항목은 또한 복구 실행기록 파일에서 작성되어 테이블을 다시 작성하는 데 사용할 수 있는 정보를 포함합니다.

한 번에 하나의 삭제된 테이블만 복구될 수 있습니다. 다음을 수행하여 삭제된 테이블을 복구할 수 있습니다.

1. LIST HISTORY DROPPED TABLE 명령을 호출하여 삭제된 테이블을 식별하십시오(370 페이지의 『LIST HISTORY』 참조). 삭제된 테이블 D는 백업 ID 컬럼에 나열되어 있습니다.
2. 테이블이 삭제되기 이전의 데이터베이스 또는 테이블 레벨 백업 이미지를 복원하십시오.
3. 테이블 데이터를 포함하는 파일이 쓰여질 작성 디렉토리를 작성하십시오. 이 디렉토리는 모든 데이터베이스 파티션에서 액세스 가능하거나 각 파티션에 존재해야 합니다. 내보내기 디렉토리 아래의 서브디렉토리는 각 데이터베이스 파티션에 의해 자동으로 작성됩니다. 서브디렉토리의 이름은 `NODE`*`nnnn`*이며, 여기서 *nnnn*은 데이터베이스 파티션 또는 노드 번호를 나타냅니다 삭제된 테이블 데이터가 각 데이터베이스 파티션에 존재하고 있어서 이러한 데이터를 포함하고 있는 데이터 파일은 `data`라고 하는 하위 서브디렉토리로 내보냅니다(예: `\export_directory\NODE0000\data`).

4. ROLLFORWARD DATABASE 명령의 RECOVERDROPPED TABLE 옵션을 사용하여 테이블이 삭제된 이후의 특정 시점으로 롤 포워드하십시오. 또는, 테이블 공간이나 데이터베이스에 있는 다른 테이블에 대한 갱신사항이 유실되지 않도록 로그 끝까지 롤 포워드하십시오.
5. 복구 실행기록 파일에서 CREATE TABLE문을 사용하여 테이블을 다시 작성하십시오.
6. 롤 포워드 조작 중에 테이블로 내보낸 테이블 데이터를 가져오십시오.

삭제된 테이블에서 복구할 수 있는 데이터 유형에 대해서는 몇 가지 제한사항이 있습니다. 다음을 복구할 수는 없습니다.

- 대형 오브젝트(LOB) 또는 Long 필드 데이터 DROPPED TABLE RECOVERY 옵션은 Long 테이블 공간에 대해 지원되지 않습니다. LOB 또는 LONG VARCHAR 컬럼이 있는 삭제된 테이블을 복원하려는 경우, 이들 컬럼은 생성된 내보내기 파일에서 NULL로 설정됩니다. DROPPED TABLE RECOVERY 옵션은 일반 테이블 공간에는 사용할 수 있으나, 임시 또는 Long 테이블 공간에는 사용할 수 없습니다.
- 행 유형과 연관된 메타데이터(데이터가 복구되지만 메타 데이터는 아님). 입력된 테이블의 계층 테이블에 있는 데이터가 복구됩니다. 이 데이터에는 입력된 테이블에 나타내는 삭제된 많은 정보가 들어 있을 수 있습니다.

DATALINK 컬럼과 연관된 링크 파일 이름은 *복구될 수 있습니다*. 테이블 데이터를 가져온 후, 테이블은 DB2 Data Links Manager와 조화되어야 합니다. 파일의 백업 이미지는 가비지 콜렉션이 이를 이미 삭제했는지 여부에 따라 DB2 Data Links Manager에 의해 복원될 수도 안될 수도 있습니다.

로드 사본 위치 파일 사용

DB2LOADREC 레지스트리 변수는 로드 사본 위치 정보로 파일을 식별하는데 사용됩니다. 이 파일은 롤 포워드 복구 동안에 로드 사본을 위치시키는데 사용됩니다. 이 파일에는 다음과 같은 정보가 있습니다.

- 미디어 유형
- 사용될 미디어 장치의 수
- 테이블 로드 조작시 생성된 로드 사본의 위치

- 적용 가능한 경우, 로드 사본의 파일 이름

위치 파일이 없거나 파일에 일치하는 항목이 없으면, 로그 레코드의 정보가 사용됩니다.

파일의 정보는 롤 포워드 복구가 발생하기 전에 겹쳐쓸 수 있습니다.

주:

1. 파티션된 데이터베이스 환경에서, DB2LOADREC 레지스트리 변수는 db2profile 파일에 있어야 합니다.
2. 파티션된 데이터베이스 환경에서, 로드 사본 파일이 각 데이터베이스 파티션 서버마다 있어야 하고 파일 이름(경로 포함)은 일치해야 합니다.
3. DB2LOADREC 레지스트리 변수로 식별된 파일의 항목이 유효하지 않으면, 이전 로드 사본 위치 파일이 유효하지 않은 항목을 대체할 정보를 제공하기 위해 사용됩니다.

다음 정보는 위치 파일에서 제공됩니다. 처음 5개의 매개변수는 유효한 값을 가져야 하며, 로드 사본을 식별하는데 사용됩니다. 전체 구조는 기록된 로드 사본 각각에 대해 반복됩니다. 예를 들면, 다음과 같습니다.

```
TIMestamp       19950725182542          * Timestamp generated at load time
SCHema          PAYROLL                 * Schema of table loaded
TABlename       EMPLOYEES               * Table name
DATabasename    DBT                     * Database name
DB2instance     TORONTO                 * DB2INSTANCE
BUFfernumber    NULL                    * Number of buffers to be used for recovery
SESsionnumber   NULL                    * Number of sessions to be used for recovery
TYPeofmedia     L                       * Type of media - L for local device
                                                          A for TSM
                                                          O for other vendors
LOCationnumber 3                        * Number of locations
   ENTry        /u/toronto/dbt.payroll.employes.001
   ENT          /u/toronto/dbt.payroll.employes.002
   ENT          /dev/rmt0
TIM             19950725192054
SCH             PAYROLL
TAB             DEPT
DAT             DBT
DB2             TORONTO
SES             NULL
BUF             NULL
TYP             A
TIM             19940325192054
SCH             PAYROLL
TAB             DEPT
DAT             DBT
DB2             TORONTO
```

```
SES             NULL
BUF             NULL
TYP             0
SHRlib          /@sys/lib/backup_vendor.a
```

주:

1. 각 키워드의 처음 3 문자가 중요합니다. 모든 키워드는 지정된 순서로 요구됩니다. 공백 행은 허용되지 않습니다.
2. 시간소인은 *yyyymmddhhmmss* 양식입니다.
3. 널(NULL)이 될 수 있는 BUF 및 SES를 제외한 모든 필드는 필수입니다. SES가 널(NULL)이면, *numloadrecses* 구성 매개변수가 지정한 값이 사용됩니다. BUF가 널(NULL)이면, 기본값은 SES+2입니다.
4. 위치 파일에 있는 항목 중 하나가 유효하지 않은 경우, 이전의 로드 사본 위치 파일을 사용하여 해당 값을 제공합니다.
5. 미디어 유형은 로컬 장치(테이프, 디스크 또는 디스켓에 대한 L), TSM(A) 또는 기타 벤더(0)가 될 수 있습니다. 유형이 L이면, 위치 수가 필요합니다. 위치 수 다음에는 위치 항목이 옵니다. 유형이 A이면, 더이상의 입력이 필요없습니다. 유형이 0이면, 공유 라이브러리 이름이 필요합니다. 백업 미디어로서 TSM 및 기타 벤더 제품 사용에 대해서는 479 페이지의 『부록G. Tivoli Storage Manager』에서 자세한 내용을 참조하십시오.
6. SHRlib 매개변수는 로드 사본 데이터를 저장하는 기능을 가진 라이브러리를 가리킵니다.
7. COPY NO 또는 NONRECOVERABLE 옵션을 지정하여 로드 조작을 호출하고 조작이 완료된 후에 데이터베이스나 영향을 받은 테이블 공간의 백업 사본을 사용하지 않을 경우, 그 다음에 로드 조작이 수행되는 특정 시점으로 데이터베이스나 테이블 공간을 복원할 수 없습니다. 즉, 롤 포워드 복구를 사용하여 데이터베이스 또는 테이블 공간을 로드 조작 이후의 상태로 다시 빌드할 수 없습니다. 로드 조작 이전의 특정 시점으로 데이터베이스 또는 테이블 공간을 복원할 수만 있습니다.

특정 로드 사본을 사용하려면, 데이터베이스에 대한 복구 실행기록 파일을 사용하여 특정 로드 조작에 대한 시간소인을 판별할 수 있습니다. 파티션된 데이터베이스 환경에서, 복구 실행기록 파일은 각 데이터베이스 파티션에 대해 지역입니다.

로드 유틸리티에 대한 자세한 정보는 *데이터 이동 유틸리티 안내 및 참조서*를 참조하십시오.

파티션된 데이터베이스 시스템에서 시계 동기화

데이터베이스 조작이 유연하게 이루어지고 포워드 복구 기능이 제한되지 않으려면, 데이터베이스 파티션 서버에 걸쳐 시스템 시계를 비교적 동기화된 상태로 유지보수해야 합니다. 데이터베이스 파티션 서버간의 시차와 트랜잭션에 발생할 수 있는 조작 및 통신상의 지연을 합한 시간은 *max_time_diff*(노드 간의 최대 시차) 데이터베이스 관리 프로그램 구성 매개변수에 지정한 값보다 작아야 합니다.

파티션된 데이터베이스 시스템에서 로그 레코드 시간소인이 트랜잭션 순서를 반영하게 하기 위해, DB2는 각 장치의 시스템 시계를 로그 레코드의 시간소인에 대한 기초로 사용합니다. 그러나 시스템 시계가 앞으로 설정될 경우, 로그 시계는 자동으로 시스템 시계와 함께 앞으로 설정됩니다. 시스템 시계는 뒤로 설정할 수도 있지만, 로그 시계는 뒤로 설정할 수 없으며, 시스템 시계가 이 시간에 일치할 때까지 같은 시간소인에 머물러 있습니다. 이 때 시계는 동기 상태에 있습니다. 이는 데이터베이스 노드의 단기 시스템 시계 오류가 데이터베이스 로그의 시간소인에 영향을 줄 수 있다는 의미가 됩니다.

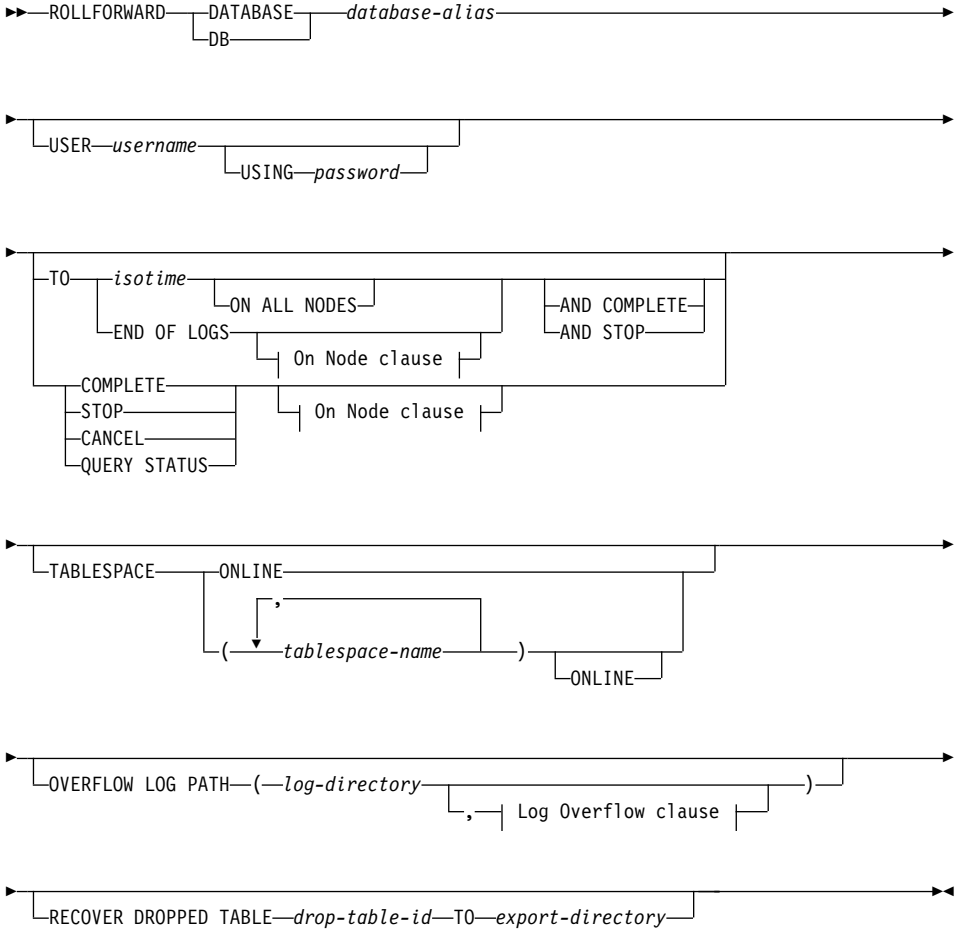
예를 들어, 연도는 1997년인데 데이터베이스 파티션 서버 A의 시스템 시계가 1999년 11월 7일로 잘못 설정되었다고 가정할 경우, 잘못 설정된 것은 *나중에* 정정되고, 갱신 트랜잭션은 데이터베이스 파티션 서버의 파티션에서 확약된다고 가정할 수 있습니다. 데이터베이스가 지속적으로 사용되며, 정기적으로 갱신될 경우, 롤 포워드 복구를 통해 1997년 11월 7일과 1999년 11월 7일 사이의 시점에는 도달할 수 없습니다. 데이터베이스 파티션 서버 A에서 확약이 완료되면 데이터베이스 로그의 시간소인은 1999로 설정되고, 로그의 시계는 시스템 시계가 이 시간과 일치할 때까지 1999년 11월 7일입니다. 이 시간 프레임 내의 특정 시점에 롤 포워드하려고 할 경우, 지정된 중지 시점(1997년 11월 7일)을 넘어서는 첫 번째 시간소인에서 조작이 멈추게 됩니다.

DB2는 시스템 시계의 갱신사항을 제어할 수 없지만, *max_time_diff* 데이터베이스 관리 프로그램 구성 매개변수는 이러한 유형의 문제점 발생 가능성을 줄입니다.

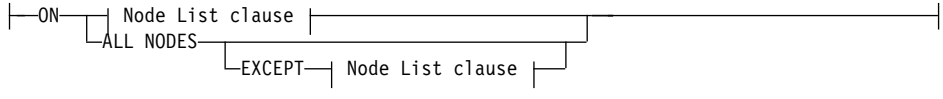
- 이 매개변수의 구성 가능 값은 1분에서 24시간 까지입니다. *max_time_diff* 설정에 대한 자세한 정보는 *관리 안내서: 성능* 책을 참조하십시오.
- 첫 번째 연결 요청이 카탈로그가 아닌 노드에서 이루어지는 경우, 데이터베이스 파티션 서버는 해당 시간을 데이터베이스용 카탈로그 노드로 보냅니다. 카탈로그 노드는 연결을 요청한 노드에서 시간을 확인하고, 해당 시간은 *max_time_diff* 매개변수에 의해 지정된 범위 내의 시간입니다. 이 범위가 초과되면, 연결이 거부됩니다.
- 데이터베이스에서 세 개 이상의 데이터베이스 파티션 서버를 포함한 갱신 트랜잭션은 관련된 데이터베이스 파티션 서버의 시계가 갱신 확약 전에 동기화된 것을 확인해야 합니다. 둘 이상의 데이터베이스 파티션 서버가 *max_time_diff*에 의해 허용된 시차를 초과하는 경우, 트랜잭션은 구간 복원되어 잘못된 시간이 다른 데이터베이스 파티션 서버로 전달되지 않도록 합니다.

ROLLFORWARD DATABASE 명령

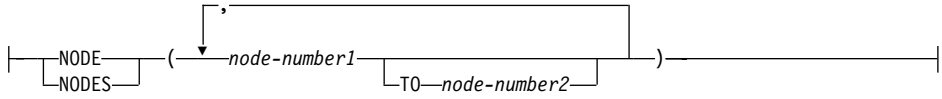
명령 구문



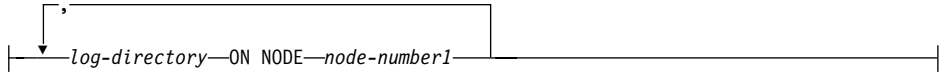
On Node clause:



Node List clause:



Log Overflow clause:



명령 매개변수

DATABASE database-alias

롤 포워드 복구할 데이터베이스의 별명

USER username

롤 포워드 복구할 데이터베이스의 사용자 이름

USING password

사용자 이름을 인증하기 위해 사용되는 암호. 암호를 생략하면, 사용자에게 입력하라는 프롬프트가 표시됩니다.

TO

isotime

확약된 모든 트랜잭션이 롤 포워드될 특정 시점(이전에 확약된 트랜잭션뿐 아니라, 그 시간에 정확하게 확약된 트랜잭션을 포함).

이 값은 시간소인(결합된 날짜 및 시간을 식별하는 7 부분 문자열)으로 지정합니다. 형식은 *yyyy-mm-dd-hh.mm.ss.nnnnnn*(연도, 월, 일, 시, 분, 초, 마이크로초)이며 UTC(Coordinated Universal Time)로 표시됩니다. UTC는 다른 로그와 연관된 동일한 시간소인(예를 들어, 일광 절약시간과 연관된 시간의 변경 때문에)을 갖지 않도록 하는데 도움이 됩니다. 백업 이미지의 시간소인은 백업 조작이 시작된 지역 시간을 기초로 합니다. CURRENT TIMEZONE 특수 레지스터는 응용프로그램 서버에서 UTC와 지역 시간 사이의 차이를 지정합니다. 차이는 시간 지속 기간으로 표시됩니다(처음 두 자리는 시간 수를, 다음 두 자리는 분 수를, 그

리고 마지막 두 자리는 초 수를 나타내는 십진수). 지역 시간에서 CURRENT TIMEZONE을 빼면 해당되는 지역 시간이 UTC로 변환됩니다.

END OF LOGS

데이터베이스 구성 매개변수 *logpath*에 나열된 모든 온라인 아카이브 로그 파일에서 확약된 모든 트랜잭션이 적용됨을 지정합니다.

ALL NODES

db2nodes.cfg 파일에 지정된 모든 노드에서 트랜잭션이 롤 포워드됨을 지정합니다. 이는 노드 절을 지정하지 않은 경우에 기본값입니다.

EXCEPT

노드 목록에 지정된 것을 제외하고, db2nodes.cfg 파일에 지정된 모든 노드에서 트랜잭션이 롤 포워드됨을 지정합니다.

ON NODE / ON NODES

노드 세트에서 데이터베이스를 롤 포워드합니다.

node-number1

노드 목록에 노드 번호를 지정합니다.

node-number2

*node-number1*에서 *node-number2*까지의 모든 노드가 노드 목록에 포함되도록 두 번째 노드 번호를 지정합니다.

COMPLETE / STOP

완료되지 않은 트랜잭션을 구간 복원한 다음 데이터베이스의 롤 포워드 보류 상태를 해제하여 롤 포워드 복구 프로세스를 완료하고 로그 레코드의 롤 포워드를 중지합니다. 이로써 롤 포워드되는 데이터베이스나 테이블 공간에 액세스할 수 있게 됩니다. 이 키워드들은 같으므로, 어느 하나만 지정하도록 하십시오. 키워드 AND는 한 번에 여러 조작을 지정할 수 있게 합니다(예: db2 rollforward db sample to end of logs and complete).

주: 특정 시점으로 테이블 공간을 롤 포워드할 때, 테이블 공간은 백업 보류 상태가 됩니다.

CANCEL

롤 포워드 복구 조작을 취소합니다. 그러면 포워드 복구가 복원 보류 상태에서 시작된 모든 노드에 데이터베이스 또는 하나 이상의 테이블 공간이 놓입니다.

- *데이터베이스* 롤 포워드 조작이 진행 중이지 않은 경우(즉, 데이터베이스가 롤 포워드 보류 상태에 있는 경우), 이 옵션은 데이터베이스를 복원 보류 상태에 놓습니다.
- *테이블 공간* 롤 포워드 조작이 진행중이 아닌 경우(즉, 테이블 공간이 롤 포워드 보류 상태에 있는 경우), 테이블 공간 목록을 지정해야 합니다. 목록의 모든 테이블 공간은 복원 보류 상태에 놓입니다.
- 테이블 공간 롤 포워드 조작이 진행 *중인* 경우(즉, 최소한 하나의 테이블 공간이 롤 포워드 진행 중 상태에 있는 경우), 롤 포워드 진행 중 상태에 있는 모든 테이블 공간이 복원 보류 상태가 됩니다. 테이블 공간 목록을 지정하면, 그 목록에 롤 포워드 진행 중 상태의 모든 테이블 공간이 포함되어야 합니다. 목록의 모든 테이블 공간은 복원 보류 상태에 놓입니다.
- 특정 시점으로 롤 포워드할 경우, 전달되는 테이블 공간 이름은 무시되고, 롤 포워드 진행 중 상태에 있는 모든 테이블 공간이 복원 보류 상태가 됩니다.
- 테이블 공간 목록에 따라 로그 *끝까*지 롤 포워드할 경우, 나열된 테이블 공간만 복원 보류 상태가 됩니다.

이 옵션은 *실제로 수행 중인* 롤 포워드 조작을 취소할 경우에 사용할 수 없습니다. 이것은 진행 중이지만 그 당시 실제로 수행 중인 상태가 아닌 롤 포워드 조작을 취소할 경우에만 사용할 수 있습니다. 롤 포워드 조작은 다음과 같은 경우에 진행 중이지만 수행 중 상태가 되지 않습니다.

- 이상 종료되었습니다.
- STOP 옵션을 지정하지 않았습니다.
- 오류로 인해 실패하였습니다. 복구할 수 없는 로드 조작을 통한 롤 포워드와 같은 일부 오류는 테이블 공간이 복원 보류 상태가 되도록 합니다.

주: 이 옵션은 주의해서 사용해야 하며, 진행 중인 롤 포워드 조작이 일부 테이블 공간 상태가 롤 포워드 보류 상태나 복원 보류 상태여서 완료할 수 없는 경우에만 사용하십시오. 확실하지 않으면, LIST TABLESPACES 명령을 사용하여 롤 포워드 진행 중 상태나 롤 포워드 보류 상태에 있는 테이블 공간을 식별하십시오.

QUERY STATUS

데이터베이스 관리 프로그램이 롤 포워드한 로그 파일, 필요한 다음 아카이브 파일 및 롤 포워드 처리가 시작된 이후에 마지막으로 확약된 트랜잭션의 시간소인(CUT로 표시)을 나열합니다. 파티션된 데이터베이스 환경에서는 이 상태 정보가 각 노드마다 리턴됩니다. 정보는 다음 필드를 포함하여 리턴됩니다.

노드 번호

롤 포워드 상태

가능한 상태: 데이터베이스 또는 테이블 공간 롤 포워드 보류, 데이터베이스 또는 테이블 공간 롤 포워드 진행 중, 데이터베이스 또는 테이블 공간 롤 포워드 처리 중지 또는 보류 중이 아님

읽을 다음 로그 파일

다음 필수 로그 파일의 이름을 포함하는 문자열. 파티션된 데이터베이스 환경에서는 롤 포워드 유틸리티가 로그 파일 누락 또는 로그 정보 불일치 발생을 나타내는 리턴 코드로 실패할 경우에 이 정보를 사용하십시오.

처리된 로그 파일

더이상 복구할 필요가 없어서 디렉토리에서 제거할 수 있는 처리된 로그 파일의 이름을 포함하는 문자열. 예를 들어, 가장 오래된 확약 트랜잭션이 로그 파일 *x*에서 시작할 경우, 사용하지 않는 로그 파일의 범위에는 *x*가 포함되지 않습니다. 범위는 *x* - 1에서 끝납니다.

마지막 확약 트랜잭션

ISO 형식(*yyyy-mm-dd-hh.mm.ss*)의 시간소인을 포함하는 문자열. 이 시간소인은 롤 포워드 복구 완료 후에 확약된 마지막 트랜잭션을 표시합니다. 시간소인을 데이터베이스로 적용합니다. 테이블

공간 롤 포워드 복구의 경우, 이는 데이터베이스에 대해 확약된 마지막 트랜잭션의 시간소인입니다.

주: QUERY STATUS는 TO, STOP, COMPLETE 또는 CANCEL 절을 생략할 경우에 기본값입니다. TO, STOP 또는 COMPLETE를 지정한 경우, 명령이 성공적으로 완료되면 상태 정보가 표시됩니다. 개별적 테이블 공간을 지정할 경우 그 테이블 공간은 무시되며, 상태 요청은 지정된 테이블 공간에만 적용되지 않습니다.

TABLESPACE

이 키워드는 테이블 공간 레벨 롤 포워드 복구에 대해서 지정됩니다.

tablespace-name

특정 시점으로의 테이블 공간 레벨 롤 포워드 복구에 대해 필수. 로그 끝까지의 롤 포워드 복구에 테이블 공간 부속 집합을 지정할 수 있게 합니다. 파티션된 데이터베이스 환경에서 목록에 있는 각 테이블 공간이 롤 포워드되는 각 노드에 존재할 필요는 없습니다. *존재할* 경우, 올바른 상태에 있어야 합니다.

ONLINE

이 키워드는 테이블 공간 레벨 롤 포워드 복구가 온라인에서 수행되도록 할 경우에 지정합니다. 이는 롤 포워드 복구가 진행 중일 때 다른 에이전트가 연결될 수 있음을 의미합니다.

OVERFLOW LOG PATH log-directory

복구 동안 아카이브된 로그에 대해 검색할 대체 로그 경로를 지정합니다. *logpath* 데이터베이스 구성 매개변수에 지정한 위치가 아닌 다른 위치로 로그 파일을 이동한 경우 이 매개변수를 사용하십시오. 파티션된 데이터베이스 환경에서, 이는 모든 노드*에 대해* (완전한) 기본 오버플로우 로그 경로입니다. 단일 파티션 데이터베이스에 대해 상대 오버플로우 로그 경로를 지정할 수 있습니다. 롤 포워드 유틸리티는 필요한 다음 로그를 찾을 수 없는 경우, 로그 이름이 SQLCA에서 리턴되고 롤 포워드 복구는 중지됩니다. 더이상 사용 가능한 로그가 없으면, STOP 옵션을 사용하여 롤 포워드 복구를 종료하십시오. 데이터베이스나 테이블 공간이 일관성 있는 상태에 있도록 불완전한 트랜잭션은 구간 복원됩니다.

log-directory ON NODE

파티션된 데이터베이스 환경에서는 다른 로그 경로가 특정 노드에 대해 기본 오버플로우 로그 경로 위에 겹쳐쓸 수 있습니다.

RECOVER DROPPED TABLE drop-table-id

롤 포워드 조작 중 삭제된 테이블을 복구합니다. 370 페이지의 『LIST HISTORY』를 사용하여 테이블 ID를 확보할 수 있습니다.

TO export-directory

테이블 데이터를 포함하는 파일을 쓸 디렉토리를 지정합니다. 디렉토리는 모든 노드에서 액세스 가능해야 합니다.

데이터베이스 API 롤 포워드

C API 구문

```
/* File: sqlutil.h */
/* API: Rollforward Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqluroll (
    struct rfwd_input * pRfwdInput,
    struct rfwd_output * pRfwdOuput,
    struct sqlca * pSqlca);
/* ... */
```

일반 API 구문

```
/* File: sqlutil.h */
/* API: Rollforward Database */
/* ... */
SQL_API_RC SQL_API_RN
  sqlgroll (
    struct grfwd_input * grfwdin,
    struct rfwd_output * rfwdout,
    struct sqlca * sqlca);

SQL_STRUCTURE grfwd_input
{
   unsigned short DbAliasLen,
   unsigned short StopTimeLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short OverflowLogPathLen,
   unsigned short ReportFileLen,      /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   sqluint32 Version,
   char * pDbAlias,
   unsigned short CallerAction,
   char * pStopTime,
   char * pUserName,
   char * pPassword,
   char * pOverflowLogPath,
   unsigned short NumChngLgOvrflw,
   struct sqlurf_newlogpath * pChngLogOvrflw,
   unsigned short ConnectMode,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   short AllNodeFlag,
   short NumNodes,
   SQL_PDB_NODE_TYPE * pNodeList,
   short NumNodeInfo,
   unsigned short DlMode,             /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   char * pReportFile,                /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   char * pDroppedTblID,
   char * pExportDir
}
/* ... */
```

API 매개변수

pRfwdInput

입력. *rfwd_input* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 185 페이지의 『데이터 구조: RFWD-INPUT』을 참조하십시오.

pRfwdOutput

출력. *rfwd_output* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 188 페이지의 『데이터 구조: RFWD-OUTPUT』을 참조하십시오.

DbAliasLen

입력. 데이터베이스 별명의 길이(바이트)를 표시하는 부호 없는 2바이트 정수

StopTimeLen

입력. 중지 시간의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 중지 시간이 제공되지 않았으면 0으로 설정하십시오.

UserNameLen

입력. 사용자 이름의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 사용자 이름이 제공되지 않았으면 0으로 설정하십시오.

PasswordLen

입력. 암호의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 암호가 제공되지 않았으면 0으로 설정하십시오.

OverflowLogPathLen

입력. 오버플로우 로그 경로의 길이(바이트)를 표시하는 부호 없는 2바이트 정수. 오버플로우 로그 경로가 제공되지 않았으면 0으로 설정하십시오.

ReportFileLen

입력. 이 매개변수는 현재 사용되지 않으며, 0으로 설정되어야 합니다.

Version

입력. 롤 포워드 매개변수의 버전 ID. SQLUM_RFWD_VERSION으로 정의됩니다.

pDbAlias

입력. 데이터베이스 별명을 포함한 문자열. 이는 시스템 데이터베이스 디렉토리에서 카탈로그화된 별명입니다.

CallerAction

입력. 취할 조치를 지정합니다. 올바른 값은 다음과 같습니다(`sqlutil`에 정의됨).

SQLUM_ROLLFWD

*pPointInTime*이 지정한 특정 시점으로 롤 포워드합니다. 데이터베이스 롤 포워드 복구의 경우, 데이터베이스는 롤 포워드 보류 상

태로 남습니다. 특정 시점으로 테이블 공간을 롤 포워드할 경우, 테이블 공간은 롤 포워드 진행 중 상태로 남습니다.

SQLUM_STOP
롤 포워드 복구의 끝. 새로운 로그 레코드는 처리되지 않으며 확약되지 않은 트랜잭션은 백아웃됩니다. 데이터베이스 또는 테이블 공간의 *롤 포워드 보류* 상태가 해제됩니다. SQLUM_COMPLETE가 동의어입니다.

SQLUM_ROLLFWD_STOP
*pPointInTime*이 지정한 특정 시점으로 롤 포워드하고 롤 포워드 복구를 종료합니다. 데이터베이스 또는 테이블 공간의 롤 포워드 보류 상태가 해제됩니다. SQLUM_ROLLFWD_COMPLETE가 동의어입니다.

SQLUM_QUERY
pNextArcFileName, pFirstDelArcFileName, pLastDelArcFileName 및 *pLastCommitTime*용 조회 값. 데이터베이스 상태 및 노드 번호가 리턴됩니다.

SQLUM_PARM_CHECK
롤 포워드 조작을 수행하지 않고 매개변수의 유효성을 확인합니다.

SQLUM_CANCEL
현재 수행중인 롤 포워드 조작 취소. 데이터베이스 또는 테이블 공간은 복구 보류 상태에 놓입니다.

주: 이 옵션은 롤 포워드 조작이 실제로 수행 중일 때 사용할 수 없습니다. 조작이 일시중지되거나(즉, STOP 대기 중) 롤 포워드 조작 중에 시스템 실패가 발생한 경우 사용할 수 있습니다. 사용할 경우 주의해야 합니다.

데이터베이스 롤 포워드에는 테이프 장치를 사용한 로드 복구 조작이 필요합니다. 장치에서의 사용 개입이 필요할 경우 경고 메시지와 함께 롤 포워드 API가 리턴됩니다. API는 다음과 같은 세 가지의 호출자 조치 중 하나를 사용하여 다시 호출할 수 있습니다.

SQLUM_LOADREC_CONTINUE

경고 메시지를 생성한 장치를 계속 사용합니다(예를 들어, 새로운 테이프가 마운트된 경우).

SQLUM_LOADREC_DEVICE_TERMINATE

경고 메시지를 생성한 장치의 사용을 중지합니다(예를 들어, 테이프가 더이상 없을 경우).

SQLUM_LOADREC_TERMINATE

로드 복구에 사용 중인 모든 장치를 종료합니다.

pStopTime

입력. ISO 형식의 시간소인을 포함하는 문자열. 데이터베이스 복구는 이 시간소인을 초과할 경우에 중지됩니다. 가능한 한 롤 포워드를 수행하려면 SQLUM_INFINITY_TIMESTAMP을 지정하십시오. SQLUM_QUERY, SQLUM_PARM_CHECK 및 로그 복구(SQLUM_LOADREC_*xxx*) 호출자 중 어느 하나에 대해 NULL일 수 있습니다.

pUserName

입력. 응용프로그램의 사용자 이름을 포함하는 문자열. 널(NULL)일 수 있습니다.

pPassword

입력. 제공된 사용자 이름(있는 경우)의 암호를 포함하는 문자열. 널(NULL)일 수 있습니다.

pOverflowLogPath

입력. 이 매개변수는 사용할 대체 로그 경로를 지정하기 위해 사용됩니다. 사용 중인 로그 파일 외에도, 아카이브된 로그 파일을 이 유틸리티에서 사용하기 전에 *logpath*로 이동해야 합니다(사용자에 의해)(*Administrative API Reference*의 『sqlfxdb - 데이터베이스 구성 가져오기』 참조). 이것은 사용자가 *logpath*에 충분한 공간을 가지고 있지 않은 경우에 문제점이 될 수 있습니다. 이러한 이유로 오버플로우 로그 경로가 제공됩니다. 롤 포워드 복구 동안, 첫 번째는 *logpath*, 그 다음은 오버플로우 로그 경로에서 필수 로그 파일을 검색합니다. 테이블 공간 롤 포워드 복구에 필요한 로그 파일은 *logpath*나 오버플로우 로그 경로로 가져올 수 있습니다. 호출자가 오버플로우 로그 경로를 지정하지 않을 경우, 기본값은 *logpath*입니다. 파티

션된 데이터베이스 환경에서, 오버플로우 로그 경로는 완전한 유효 경로여야 합니다. 기본 경로는 각 노드에 대한 기본 오버플로우 로그 경로입니다. 단일 파티션 환경에서는 서버가 지역 서버일 경우 오버플로우 로그 경로는 상대적입니다.

NumChngLgOvrflw

MPP 전용. 변경된 오버플로우 로그 경로의 수. 이러한 새로운 경로는 지정된 노드에 대해서만 기본 오버플로우 로그 경로를 대체합니다.

pChngLogOvrflw

MPP 전용. 변경된 오버플로우 로그 경로의 완전한 이름을 포함하는 구조에 대한 포인터. 이러한 새로운 경로는 지정된 노드에 대해서만 기본 오버플로우 로그 경로를 대체합니다.

ConnectMode

입력. 올바른 값은 다음과 같습니다(`sqlutil`에 정의됨).

SQLUM_OFFLINE

오프라인 롤 포워드. 이 값은 데이터베이스 롤 포워드 복구에 대해 지정해야 합니다.

SQLUM_ONLINE

온라인 롤 포워드

pTablespaceList

입력. 로그 끝이나 특정 시점으로 롤 포워드될 테이블 공간의 이름을 포함하는 구조에 대한 포인터. 지정하지 않을 경우, 롤 포워드를 필요로 하는 테이블 공간이 선택됩니다.

AllNodeFlag

MPP 전용. 입력. 롤 포워드 조작이 `db2nodes.cfg`에 정의된 모든 노드에 적용되는지 여부를 표시합니다. 유효한 값은 다음과 같습니다.

SQLURF_NODE_LIST

*pNodeList*에서 전달되는 노드 목록의 노드에 적용합니다.

SQLURF_ALL_NODES

모든 노드에 적용. *pNodeList*는 널(NULL)이어야 합니다. 이는 기본값입니다.

SQLURF_ALL_EXCEPT

*pNodeList*에서 전달되는 노드 목록에 있는 노드를 제외한 모든 노드에 적용합니다.

SQLURF_CAT_NODE_ONLY

카탈로그 노드에만 적용. *pNodeList*는 널(NULL)이어야 합니다.

NumNodes

입력. *pNodeList* 배열에 노드 수를 지정합니다.

pNodeList

입력. 롤 포워드 복구를 수행할 노드 번호 배열에 대한 포인터

NumNodeInfo

입력. 출력 매개변수 *pNodeInfo*의 크기를 정의합니다. 이 매개변수는 롤 포워드되는 각 노드에서 상태 정보를 보유할 만큼 충분히 커야 합니다. 단일 파티션 환경에서는 이 매개변수를 1로 설정해야 합니다. 이 매개변수의 값은 API가 호출되는 노드 수와 같아야 합니다.

DlMode

입력. 이 매개변수는 현재 사용되지 않으며, 0으로 설정되어야 합니다.

pReportFile

입력. 이 매개변수는 현재 사용되지 않으며, 널(NULL)로 설정되어야 합니다.

pDroppedTblID

입력. 복구가 시도되고 있는 삭제된 테이블의 ID를 포함하는 문자열

pExportDir

입력. 삭제된 테이블 데이터를 내보낼 디렉토리

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

REXX API 구문

```
ROLLFORWARD DATABASE database-alias [USING :value] [USER username USING password]
[rollforward_action_clause | load_recovery_action_clause]
where rollforward_action_clause stands for:
    { TO point-in-time [AND STOP] |
      {
        [TO END OF LOGS [AND STOP] | STOP | CANCEL | QUERY STATUS | PARM CHECK }
        [ON {:nodelist | ALL NODES [EXCEPT :nodelist]}]
      }
    }
    [TABLESPACE {ONLINE |:tablespacenames [ONLINE]} ]
    [OVERFLOW LOG PATH default-log-path [:logpaths]]
and load_recovery_action_clause stands for:
    LOAD RECOVERY { CONTINUE | DEVICE_TERMINATE | TERMINATE }
```

REXX API 매개변수

database-alias

롤 포워드되는 데이터베이스의 별명

value 출력값을 포함하는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0 변수의 요소 수

XXX.1 응용프로그램 ID

XXX.2 노드에서 받은 응답 수

XXX.2.1.1 첫 번째 노드 번호

XXX.2.1.2 첫 번째 상태 정보

XXX.2.1.3 필요한 첫 번째 다음 아카이브 파일

XXX.2.1.4 삭제할 첫 번째 최초 아카이브 파일

XXX.2.1.5 삭제할 첫 번째 최종 아카이브 파일

XXX.2.1.6 첫 번째 최종 확약 시간

XXX.2.2.1 두 번째 노드 번호

XXX.2.2.2 두 번째 상태 정보

XXX.2.2.3 필요한 두 번째 다음 아카이브 파일

XXX.2.2.4 삭제할 두 번째 최초 아카이브 파일

XXX.2.2.5 삭제할 두 번째 최종 아카이브 파일

XXX.2.2.6 두 번째 최종 확약 시간

XXX.2.3.x 기타

username

롤 포워드할 데이터베이스의 사용자 이름을 식별합니다.

password

사용자 이름을 인증하기 위해 사용되는 암호

point-in-time

ISO 형식 *yyyy-mm-dd-hh.mm.ss.nnnnnn*(연도, 월, 일, 시, 분, 초, 마이크로초)의 시간소인으로, UTC(Coordinated Universal Time)로 표시됩니다.

tablespacenames

롤 포워드할 테이블 공간 목록을 포함하는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름입니다.

XXX.0 롤 포워드된 테이블 공간 수

XXX.1 첫 번째 테이블 공간 이름

XXX.2 두 번째 테이블 공간 이름

XXX.x 기타

default-log-path

복구 동안 아카이브된 로그에 대해 검색할 기본 오버플로우 로그 경로

logpaths

복구 동안 아카이브된 로그에 대해 검색할 대체 로그 경로 목록을 포함하는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름입니다.

XXX.0 변경된 오버플로우 로그 경로 수

XXX.1.1 첫 번째 노드

XXX.1.2 첫 번째 오버플로우 로그 경로

XXX.2.1 두 번째 노드

XXX.2.2 두 번째 오버플로우 로그 경로

XXX.3.1 세 번째 노드

XXX.3.2 세 번째 오버플로우 로그 경로

XXX.x.1 기타

nodelist

노드 목록을 포함하는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름입니다.

XXX.0 노드 수

XXX.1 첫 번째 노드

XXX.2 두 번째 노드

XXX.x 기타

데이터 구조: RFWD-INPUT

이 구조는 정보를 175 페이지의 『데이터베이스 API 롤 포워드』에 전달하기 위해 사용됩니다.

표 8. RFWD-INPUT 구조 내의 필드

필드 이름	데이터 유형	설명
VERSION	sqluint32	롤 포워드 버전
PDBALIAS	포인터	데이터베이스 별명
CALLERACTION	UNSIGNED SHORT	조치
PSTOPTIME	포인터	중지 시간
PUSERNAME	포인터	사용자 이름
PPASSWORD	포인터	암호
POVERFLOWLOGPATH	포인터	오버플로우 로그 경로
NUMCHNGLGOVRFLW	UNSIGNED SHORT	변경된 오버플로우 로그 경로의 수(MPP 전용)
PCHNGLOGOVRFLW	구조	변경된 오버플로우 로그 경로(MPP 전용)
CONNECTMODE	UNSIGNED SHORT	연결 모드
PTABLESPACELIST	구조	테이블 공간 이름 목록에 대한 포인터. 이 구조에 대한 자세한 정보는 120 페이지의 『데이터 구조: SQLU-TABLESPACE-BKRST-LIST』를 참조하십시오.
ALLNODEFLAG	SHORT	모든 모드 플래그
NUMNODES	SHORT	노드 목록 크기
PNODELIST	포인터	노드 번호 목록
NUMNODEINFO	SHORT	188 페이지의 『데이터 구조: RFWD-OUTPUT』의 *pNodeInfo* 크기
DLMODE	UNSIGNED SHORT	이 매개변수는 현재 사용되지 않습니다.
PREPORTFILE	포인터	이 매개변수는 현재 사용되지 않습니다.
PDROPPEDTBLID	포인터	복구가 시도되고 있는 삭제된 테이블의 ID를 포함하는 문자열
PEXPORTDIR	포인터	삭제된 테이블 데이터를 내보낼 디렉토리
NODENUM	SQL_PDB_NODE_TYPE	노드 번호
PATHLEN	UNSIGNED SHORT	새 로그 경로의 길이
LOGPATH	CHAR(255)	새 오버플로우 로그 경로

언어 구문

C 구조

```
/* File: sqlutil.h */
/* Structure: RFWD-INPUT */
/* ... */
SQL_STRUCTURE rfwd_input
{
  sqluint32              version;
  char                   *pDbAlias;
  unsigned short         CallerAction;
  char                   *pStopTime;
  char                   *pUserName;
  char                   *pPassword;
  char                   *pOverflowLogPath;
  unsigned short         NumChngLgOvrflw;
  struct sqlurf_newlogpath *pChngLogOvrflw;
  unsigned short         ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  short                  AllNodeFlag;
  short                  NumNodes;
  SQL_PDB_NODE_TYPE      *pNodeList;
  short                  NumNodeInfo;
  unsigned short         DlMode;         /* This parameter is not */
                                         /*   currently used. */
  char                   *pReportFile;   /* This parameter is not */
                                         /*   currently used. */
  char                   *pDroppedTblID;
  char                   *pExportDir;
};
/* ... */


/* File: sqlutil.h */
/* Structure: SQLURF-NEWLOGPATH */
/* ... */
SQL_STRUCTURE sqlurf_newlogpath
{
  SQL_PDB_NODE_TYPE      nodenum;
  unsigned short         pathlen;
  char                   logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
};
/* ... */
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQL-RFWD-INPUT.
    05 SQL-VERSION            PIC 9(9) COMP-5.
    05 SQL-DBALIAS            USAGE IS POINTER.
    05 SQL-CALLERACTION       PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-STOPTIME           USAGE IS POINTER.
    05 SQL-USERNAME           USAGE IS POINTER.
    05 SQL-PASSWORD           USAGE IS POINTER.
    05 SQL-OVERFLOWLOGPATH    USAGE IS POINTER.
    05 SQL-NUMCHANGE          PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-CHNG-LOG-OVRFLW  USAGE IS POINTER.
    05 SQL-CONNECTMODE        PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-TABLESPACE-LIST  USAGE IS POINTER.
    05 SQL-ALLNODEFLAG        PIC S9(4) COMP-5.
    05 SQL-NUMNODES           PIC S9(4) COMP-5.
    05 SQL-NODELIST           USAGE IS POINTER.
    05 SQL-NUMNODEINFO        PIC S9(4) COMP-5.
    05 SQL-DLMODE             PIC 9(4) COMP-5.  * This parameter is not
                                                *   currently used.
    05 SQL-REPORTFILE         USAGE IS POINTER. * This parameter is not
                                                *   currently used.
    05 SQL-DROPPEDTBLID       USAGE IS POINTER.
    05 SQL-EXPORTDIR       USAGE IS POINTER.
*


* File: sqlutil.cbl
01 SQLURF-NEWLOGPATH.
    05 SQL-NODENUM            PIC S9(4) COMP-5.
    05 SQL-PATHLEN            PIC 9(4) COMP-5.
    05 SQL-LOGPATH            PIC X(254).
    05 FILLER                 PIC X.
    05 FILLER                 PIC X(1).
*
```

데이터 구조: RFWD-OUTPUT

이 구조는 175 페이지의『데이터베이스 API 롤 포워드』에서 정보를 전달하기 위해 사용됩니다.

표 9. RFWD-OUTPUT 구조 내의 필드

필드 이름	데이터 유형	설명
PAPPLICATIONID	포인터	API에서 리턴된 응용프로그램 식별자를 보유하기 위한 길이 SQLU_APPLID_LEN+1의 버퍼 주소(sqlutil에서 정의됨). 이 식별자는 데이터베이스 시스템 모니터 API와 함께 사용하여 응용프로그램을 모니터할 수 있습니다. 이 정보에 관심이 없으면, NULL 포인터를 지정하십시오. 파티션된 데이터베이스 환경에서는 카탈로그 노드에 대한 응용프로그램 식별자만 리턴합니다.
PNUMREPLIES	포인터	받은 노드 응답 수. 응답하는 각 노드는 *pNodeInfo*의 *sqlurf_info*를 채웁니다. 단일 파티션 환경에서는 이 매개변수의 값이 1입니다.
PNODEINFO	구조	노드 응답 정보. *NumNodeInfo sqlurf_info* 구조의 사용자 정의 배열

표 10. SQLURF-INFO 구조 내의 필드

필드 이름	데이터 유형	설명
NODENUM	SQL_PDB_NODE_TYPE	노드 번호
STATE	LONG	상태 정보
NEXTARCLOG	UNSIGNED CHAR(13)	다음 필수 아카이브 로그 파일의 리턴된 이름을 보유하기 위한 12 바이트 버퍼. SQLUM_QUERY 이외의 다른 호출자 조치를 지정할 경우, 이 필드에 리턴되는 값은 파일에 액세스하는 중 오류가 발생했음을 나타냅니다. 가능한 원인은 다음과 같습니다. • 데이터베이스 로그 디렉토리나 오버플로우 로그 경로 매개변수에 지정된 경로에서 파일을 찾을 수 없었습니다. • User Exit 프로그램이 아카이브 파일을 리턴하는데 실패합니다.

표 10. SQLURF-INFO 구조 내의 필드 (계속)

필드 이름	데이터 유형	설명
FIRSTARCDEL	UNSIGNED CHAR(13)	복구에 더이상 필요하지 않은 첫 번째 아카이브 로그 파일의 리턴된 이름을 보유하기 위한 12바이트 버퍼. 이 파일과 *lastarcdel*을 포함하여 모든 파일을 이동하여 디스크에 공간을 만들 수 있습니다. 예를 들어, *firstarcdel* 및 *lastarcdel*에서 리턴되는 값이 `S0000001.LOG` 및 `S0000005.LOG`일 경우, 다음 로그 파일을 이동할 수 있습니다. • `S0000001.LOG` • `S0000002.LOG` • `S0000003.LOG` • `S0000004.LOG` • `S0000005.LOG`
LASTARCDEL	UNSIGNED CHAR(13)	데이터베이스 로그 디렉토리에서 제거할 수 있는 마지막 아카이브 로그 파일의 리턴된 이름을 보유하기 위한 12바이트 버퍼
LASTCOMMIT	UNSIGNED CHAR(27)	ISO 형식의 시간소인을 포함하는 26자 문자열. 이 값은 롤 포워드 조작이 종료된 이후의 마지막 확약 트랜잭션에 대한 시간소인을 나타냅니다.

*STATE*에 가능한 값은 다음과 같습니다(`sqlutil`에 정의됨).

SQLURFQ_NOT_AVAILABLE
노드에 연결할 수 없습니다.

SQLURFQ_NOT_RFW_PENDING
데이터베이스가 롤 포워드 보류 상태가 아닙니다.

SQLURFQ_DB_RFW_PENDING
데이터베이스가 롤 포워드 보류 상태입니다.

SQLURFQ_TBL_RFW_PENDING
테이블 공간이 롤 포워드 보류 상태입니다.

SQLURFQ_DB_RFW_IN_PROGRESS
데이터베이스가 롤 포워드 진행 중 상태입니다.

SQLURFQ_TBL_RFW_IN_PROGRESS
테이블 공간이 롤 포워드 진행 중 상태입니다.

SQLURFQ_DB_RFW_STOPPING

STOP 요청을 처리하는 중에 데이터베이스 롤 포워드 조작이 인터럽트되었습니다.

SQLURFQ_TBL_RFW_STOPPING

STOP 요청을 처리하는 중에 테이블 공간 롤 포워드 조작이 인터럽트되었습니다.

언어 구문

C 구조

```
/* File: sqlutil.h */
/* Structure: RFWD-OUTPUT */
/* ... */
SQL_STRUCTURE rfwd_output
{
  char            *pApplicationId;
  long            *pNumReplies;
  struct sqlurf_info *pNodeInfo;
};
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLURF-INFO */
/* ... */
SQL_STRUCTURE sqlurf_info
{
  SQL_PDB_NODE_TYPE nodenum;
  long            state;
  unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
};
/* ... */
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQL-RFWD-OUTPUT.
    05 SQL-APPLID             USAGE IS POINTER.
    05 SQL-NUMREPLIES         USAGE IS POINTER.
    05 SQL-P-NODE-INFO        USAGE IS POINTER.
*


* File: sqlutil.cbl
01 SQLURF-INFO.
    05 SQL-NODENUM            PIC S9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-STATE              PIC S9(9) COMP-5.
    05 SQL-NEXTARCLOG         PIC X(12).
    05 FILLER                 PIC X.
    05 SQL-FIRSTARCDEL        PIC X(12).
    05 FILLER                 PIC X.
    05 SQL-LASTARCDEL         PIC X(12).
    05 FILLER                 PIC X.
    05 SQL-LASTCOMMIT         PIC X(26).
    05 FILLER                 PIC X.
    05 FILLER                 PIC X(2).
*
```

롤 포워드 세션 예

CLP 예

예 1

ROLLFORWARD DATABASE 명령은 한 번에 여러 조작을 지정하는 것을 허용합니다. 각각의 조작은 키워드 AND로 구분됩니다. 예를 들어, 로그 끝으로 롤 포워드하여 완료하려면, 다음과 같은 별도의 명령을 하나로 결합할 수 있습니다.

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete

db2 rollforward db sample to end of logs and complete
```

두 가지가 같다하더라도, 그러한 조작들을 두 단계로 수행하는 것이 좋습니다. 중지하기 전에 예상대로 롤 포워드 조작이 진행되었는지 검증하는 것이 중요합니다. 로그가 누락될 수도 있습니다. 롤 포워드 복구 동안 잘못된 로그가 발견될 경우에는 특히 중요합니다. 잘못된 로그는 『로그의 끝』을 의미하도록 해석됩니다. 그러한 경우, 해당 로그의 손상되지 않은 백업 사본을 사용하여 더 많은 로그를 통해 롤 포워드 조작을 계속할 수 있습니다.

예 2

로그 끝까지의 롤 포워드(두 개의 테이블 공간이 복원되었음)

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

위의 두 명령문은 동일합니다. 테이블 공간을 로그 끝까지 롤 포워드 복구할 경우에 AND STOP이나 AND COMPLETE는 필요하지 않습니다. 테이블 공간 이름은 필수는 아닙니다. 지정하지 않을 경우 롤 포워드 복구를 필요로 하는 모든 테이블 공간이 포함됩니다. 테이블 공간의 부속 집합만 롤 포워드될 경우, 해당되는 이름을 지정해야 합니다.

예 3

세 개의 테이블 공간을 복원한 후에 하나는 로그 끝으로 롤 포워드하고, 다른 두 개는 온라인에서 수행되도록 특정 시점으로 롤 포워드합니다.

```
db2 rollforward db sample to end of logs tablespace(TBS1) online
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS2, TBS3) online
```

두 개의 롤 포워드 조작은 동시에 수행할 수 없습니다. 두 번째 명령은 첫 번째 롤 포워드 조작이 성공적으로 완료된 후에만 호출할 수 있습니다.

예 4

데이터베이스를 복원한 후 OVERFLOW LOG PATH를 사용하여 User Exit이 아카이브된 로그를 저장하는 디렉토리를 지정함으로써 특정 시점으로 롤 포워드합니다.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

예 5(MPP)

0, 1 및 2 세 개의 노드가 있습니다. 테이블 공간 TBS1은 모든 노드에서 정의되고, 테이블 공간 TBS2는 노드 0 및 2에서 정의됩니다. 노드 1에서 데이터베이스를 복원하고, 노드 0과 2에서 TBS1을 복원한 후 노드 1에서 데이터베이스를 롤 포워드합니다.

```
db2 rollforward db sample to end of logs and stop
```

이 명령문은 경고 SQL1271(『데이터베이스가 복구되었으나 노드 0과 2에서 하나 이상의 테이블 공간이 오프라인 상태입니다.』)을 리턴합니다.

```
db2 rollforward db sample to end of logs
```

이는 노드 0과 2에서 TBS1을 롤 포워드합니다. TABLESPACE(TBS1) 절은 이 경우에 선택적입니다.

예 6(MPP)

노드 0과 2에서만 테이블 공간 TBS1을 복원한 후에 노드 0과 2에서 TBS1을 롤 포워드합니다.

```
db2 rollforward db sample to end of logs
```

노드 1은 무시됩니다.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

이 명령문은 TBS1이 노드 1에서 롤 포워드 복구를 준비하지 않으므로 실패합니다. SQL4906N을 보고합니다.

```
db2 rollforward db sample to end of logs on nodes (0, 2) tablespace(TBS1)
```

이 명령문은 성공적으로 완료됩니다.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

이 명령문은 TBS1이 노드 1에서 롤 포워드 복구를 준비하지 않으므로 실패합니다. 모든 조각이 함께 롤 포워드되어야 합니다.

주: 테이블 공간이 특정 시점으로 롤 포워드될 경우, 노드 절은 승인되지 않습니다. 롤 포워드 조작은 테이블 공간이 상주하는 모든 노드에서 발생해야 합니다.

노드 1에서 TBS1을 복원한 후,

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

이 명령문은 성공적으로 완료됩니다.

예 7(MPP)

모든 노드에서 테이블 공간을 복원한 후, PIT2로 롤 포워드하지만 AND STOP을 지정하지 않습니다. 롤 포워드 조작이 계속 진행됩니다. 취소하여 PIT1로 롤 포워드합니다.

```
   db2 rollforward db sample to pit2 tablespace(TBS1)
   db2 rollforward db sample cancel tablespace(TBS1)
** restore TBS1 on all nodes **
   db2 rollforward db sample to pit1 tablespace(TBS1)
   db2 rollforward db sample stop tablespace(TBS1)
```

예 8(MPP)

db2nodes.cfg 파일에 나열된 8개의 노드(3 - 10)에 상주하는 테이블 공간을 롤 포워드 복구합니다.

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

로그 끝까지(특정 시점이 아닌)의 이 조작은 성공적으로 완료됩니다. 테이블 공간이 상주하는 노드는 지정하지 않아도 됩니다. 유틸리티 기본값은 db2nodes.cfg 파일입니다.

예 9(MPP)

단일 노드의 노드 그룹(노드 6에 있는)에 상주하는 6개의 작은 테이블 공간을 롤 포워드 복구합니다.

```
db2 rollforward database dwtest to end of logs on node (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

로그 끝까지(특정 시점이 아닌)의 이 조작은 성공적으로 완료됩니다.

DB2 명령 스크립트의 샘플과 이를 사용하는 방법에 대한 정보는 473 페이지의 『부록F. CLP 스크립트 복구』에 제공되어 있습니다.

API 예

DB2 API 및 Embedded SQL 호출을 포함하는 샘플 프로그램과 이를 사용하는 방법에 대한 정보는 421 페이지의 『부록E. 샘플 프로그램 복구』에 제공되어 있습니다.

롤 포워드 제한사항

롤 포워드 유틸리티에 다음과 같은 제한사항이 적용됩니다.

- 한 번에 하나의 롤 포워드 조작만 호출할 수 있습니다. 복구할 테이블 공간이 많을 경우, 동일 조작에서 모두를 지정할 수 있습니다.
- 가장 최근의 백업 조작 다음에 테이블 공간의 이름을 바꾼 경우, 테이블 공간을 롤 포워드할 때 새 이름을 사용하는지 확인하십시오. 이전 테이블 공간 이름은 인지되지 않습니다.
- 수행 중인 롤 포워드 조작은 취소할 수 없습니다. 완료하였지만 STOP 옵션을 지정하지 않은 롤 포워드 조작이나 완료 이전에 실패한 롤 포워드 조작만 취소할 수 있습니다.

- 특정 시점으로 테이블 공간 롤 포워드 조작을 *계속*할 수 없습니다. 이전 시간소인보다 빠른 시간소인을 지정합니다. 특정 시점을 지정하지 않을 경우, 이전 시간소인이 사용됩니다. 단지 STOP을 지정하여 특정 시점으로의 롤 포워드를 초기화할 수 있지만, 이는 관련된 테이블 공간이 모두 동일한 오프라인 백업 이미지에서 복원된 경우에만 허용됩니다. 이 경우에 로그 처리는 필요없습니다. 진행 중인 롤 포워드 조작이 완료되거나 취소되기 전에 서로 다른 테이블 공간 목록을 사용하여 다른 롤 포워드 조작을 시작할 경우, 오류 메시지(SQL4908)가 리턴됩니다. 현재 롤 포워드 중인(롤 포워드 진행 중 상태) 테이블 공간과 롤 포워드 준비 상태(롤 포워드 보류 상태)의 테이블 공간을 판별하려면 모든 노드에서 LIST TABLESPACES 명령을 호출하십시오. 세 가지 옵션이 있습니다.
  - 모든 테이블 공간에서 진행 중인 롤 포워드 조작 완료
  - 테이블 공간의 부속 집합에서 진행 중인 롤 포워드 조작 완료. (롤 포워드 조작이 특정 시점으로 계속되어 모든 노드의 파티션이 필요할 경우에는 가능하지 않을 수 있습니다.)
  - 진행 중인 롤 포워드 조작 취소

롤 포워드 문제점 해결

진행 중인 롤 포워드 조작을 취소하지 않고서는 테이블 공간을 복원하지 마십시오. 그렇지 않으면, 일부 테이블 공간이 롤 포워드 진행 중 상태에 있고 일부 테이블 공간이 롤 포워드 보류 상태에 있는 테이블 공간 세트를 수반할 수 있습니다. 진행 중인 롤 포워드 조작은 롤 포워드 진행 중 상태에 있는 테이블 공간에서만 작동합니다.

롤 포워드 유틸리티가 복구할 수 없는 조작(예: 사본이 없는 로드)을 발견하면, 연관되는 테이블 공간은 복원 보류 상태에 놓입니다. 복원 보류 상태에서 테이블 공간을 제거하려면, 더 최근의 데이터베이스 레벨 또는 테이블 공간 레벨의 백업 이미지로부터 복원해야 합니다.

롤 포워드 보류 또는 복원 보류 상태에 있는 테이블 공간이 있음을 나타내는 경고로 SQL1271이 리턴됩니다(AND STOP을 아직 요청하지 않은 경우). 데이터베이스 롤 포워드 조작 중이면, 이는 테이블 공간 중 하나가 롤 포워드 유틸리티에 의해 오프라인으로 사용되었음을 의미합니다. *테이블 공간* 롤 포워드 조작 중이면,

이는 롤 포워드되는 테이블 공간 중 하나가 롤 포워드 유틸리티에 의해 오프라인에서 사용되었거나, 계속 롤 포워드해야 하는 다른(나열되지 않은) 테이블 공간이 있음을 의미할 수 있습니다.

SQL1272는 롤 포워드 중이던(목록에 지정되어 있거나 롤 포워드 보류 상태에 있어서) 모든 테이블 공간이 롤 포워드 유틸리티에 의해 오프라인으로 사용된 경우에 리턴됩니다. 이는 복구할 수 없는 로드 조작이 수행된 테이블이나 NOT LOGGED INITIALLY 처리 중인 테이블이 포함되어 있음을 의미할 수도 있습니다. 이 오류가 리턴되면, 테이블 공간 롤 포워드 조작을 중지해야 합니다. (즉, 더이상 진행 중 상태가 되지 않아야 합니다.)

제2부 고가용성

제5장 고가용성 및 장애 복구 지원 소개

성공적인 e-business는 인터럽트되지 않는 트랜잭션 처리 시스템들의 가용성에 달려 있으며, 이는 결국 일주일 내내 매일 24시간(『24 x 7』) 끊임없이 사용 가능해야 하는 DB2와 같은 데이터베이스 관리 시스템에 의해 구동됩니다.

고가용성

고가용성(HA)은 전체 시간 이상이나 이하로 수행되며 고객이 사용 가능한 시스템을 설명하기 위해 사용되는 용어입니다. 이러한 상태가 발생하도록 하려면 다음이 만족되어야 합니다.

- 최대 작동 기간 동안 현저한 성능 저하(또는 가용성 유실도) 없이 효율적으로 트랜잭션이 처리되어야 합니다. 파티션된 데이터베이스 환경에서, DB2는 트랜잭션을 효율적으로 처리하기 위해 파티션 내 및 파티션 간 병렬 처리를 둘다 이용할 수 있습니다. *파티션 내 병렬 처리*는 복합 SQL문의 다양한 구성요소를 동시에 처리하기 위해 SMP 환경에서 사용할 수 있습니다. 한편, 파티션된 데이터베이스에서의 *파티션 간 병렬 처리*는 관련된 모든 노드에서 동시에 조회 처리하는 것을 말합니다. 병렬 처리에 대한 자세한 정보는 *관리 안내서*를 참조하십시오.
- 시스템은 하드웨어 또는 소프트웨어 실패가 발생하거나 재난이 발생할 경우 신속하게 복구할 수 있어야 합니다. JFS(Journaled File System)를 사용하면 도움이 될 것입니다. *JFS*는 구조에 대한 변경사항을 자동으로 기록하는 파일 시스템입니다. 이로써 파일 시스템 구조에 대한 어떤 변경사항도 시스템 손상으로부터 보호받게 됩니다. 단, 저장 장치가 손실되거나 훼손되지 않아야 합니다. 시스템이 손상되면, 로그에 있는 변경사항은 초기에 기록된 순서대로 파일 시스템에 적용됩니다. JFS(Journaled File System)는 빠른 복구 시간이 특징입니다. 이는 대형 파일 시스템이 있는 환경이나 데이터 무결성이 특히 중요한 경우에 선호합니다.

빠르게 복구할 수 있는 능력은 판명된 백업 및 복구 전략이 적절한지에 전적으로 달려 있습니다. 복구 전략에 대한 자세한 정보는 3 페이지의 『제1장 좋은 백업 및 복구 전략 개발』을 참조하십시오.

- 엔터프라이즈 데이터베이스를 강화시키는 소프트웨어는 계속 수행 중이고 트랜잭션 처리에 사용 가능해야 합니다. 데이터베이스 관리 프로그램을 수행 중 상태로 보존하려면, 실패할 경우 다른 데이터베이스 관리 프로그램에 인계될 수 있도록 해야 합니다. 이를 장애 복구라고 합니다. *장애 복구* 기능을 사용하면 하드웨어 장애가 있을 경우 하나의 시스템에서 다른 시스템으로 워크로드를 자동으로 전송할 수 있습니다.

끊임없이 로그 파일을 롤 포워드하는 데이터베이스의 사본을 다른 머신에 보존하여 장애 복구를 보호할 수 있습니다. 로그 *제공*은 전체 로그 파일을 아카이브 장치로부터, 또는 1차 데이터베이스에 대해 수행 중인 User Exit 프로그램을 통해 대기 머신으로 복사하는 프로세스입니다. 이 방식을 사용할 경우, 1차 데이터베이스는 DB2 복원 유틸리티나 분할 이중복사 기능을 사용하여 대기 머신으로 복원됩니다. 새로운 일시중단 입출력 지원을 사용하여 새로운 데이터베이스를 신속하게 초기화할 수 있습니다(204 페이지의 『온라인 분할 이중복사 및 일시중단된 입출력 지원을 통한 고가용성』 참조). 대기 머신의 2차 데이터베이스는 로그 파일을 연속적으로 롤 포워드합니다. 1차 데이터베이스가 실패하면, 나머지 로그 파일이 대기 머신으로 복사됩니다. 로그 끝까지 롤 포워드하고 조작을 중지하고 나면, 모든 클라이언트가 대기 머신의 2차 데이터베이스에 다시 연결됩니다.

시스템에 추가할 수 있는 플랫폼 특정 소프트웨어를 통해서도 장애 복구 지원을 제공할 수 있습니다. 예를 들면, 다음과 같습니다.

- AIX용 HACMP/ES(High Availability Cluster Multi-Processing, Enhanced Scalability)

  HACMP/ES에 대한 자세한 정보는 209 페이지의 『제6장 AIX의 고가용성』이나 『IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES』 백서를 참조하십시오. 이는 『DB2 UDB and DB2 Connect Online Support』 웹 사이트(http://www.ibm.com/software/data/pubs/papers/)에서 구할 수 있습니다.

- Windows NT 또는 Windows 2000용 Microsoft Cluster Server

MSCS에 대한 자세한 정보는 257 페이지의 『제7장 Windows 운영 환경의 고가용성』을 참조하십시오.

- Solaris 운영 환경용 Sun Cluster 또는 VERITAS Cluster Server

  Sun Cluster 2.x에 대해서는 295 페이지의 『제8장 Solaris 운영 환경의 고가용성』을 참조하십시오. Sun Cluster 3.0에 대해서는 『DB2 and High Availability on Sun Cluster 3.0』 백서를 참조하십시오. 이는 『DB2 UDB and DB2 Connect Online Support』 웹 사이트(http://www.ibm.com/software/data/pubs/papers/)에서 구할 수 있습니다. VERITAS Cluster Server에 대해서는 『DB2 and High Availability on VERITAS Cluster Server』 백서를 참조하십시오. 이는 『DB2 UDB and DB2 Connect Online Support』 웹 사이트(http://www.ibm.com/software/data/pubs/papers/)에서 구할 수 있습니다.

- Hewlett-Packard용 Multi-Computer/ServiceGuard

  HP MC/ServiceGuard에 대한 자세한 정보는 『IBM DB2 EE v.7.1 Implementation and Certification With Hewlett-Packard's MC/ServiceGuard High Availability Software』 백서를 참조하십시오. 이는 『DB2 UDB and DB2 Connect Online Support』 웹 사이트(http://www.ibm.com/software/data/pubs/papers/)에서 구할 수 있습니다.

장애 복구 전략은 보통 시스템의 클러스터를 기초로 합니다. *클러스터*는 단일 시스템으로 함께 작동하는 연결된 시스템 그룹입니다. 각 프로세스는 클러스터 내의 노드라고 합니다. 클러스터링을 사용하면 실패할 경우 서버들이 실패한 서버의 워크로드를 선택하여 서로 백업할 수 있습니다.

IP 주소 인계(또는 IP 인계)는 서버가 중단될 경우에 머신 사이에 서버 IP 주소를 전송할 수 있는 기능입니다. 클라이언트 응용프로그램에는 두 개의 머신이 다른 시간에 같은 서버인 것으로 나타납니다.

장애 복구 소프트웨어는 시스템 사이에 *하트비트(heartbeat) 모니터링*이나 *보존(keepalive) 패킷*을 사용하여 가용성을 확인할 수 있습니다. 하트비트 모니터링에는 클러스터 내의 모든 노드 사이에 일정한 통신을 유지하는 시스템 서비스가 포함됩니다. 하트비트가 검출되지 않으면, 백업 시스템에 대한 장애 복구가 시작됩니다. 일반 사용자는 보통 시스템의 실패를 인식하지 못합니다.

시장에서 가장 일반적인 두 가지의 장애 복구 전략으로는 *유휴 대기*와 *상호 인계*가 있습니다. (이 용어들과 연관되는 구성이 벤더에 따라 다양한 다른 용어와 연관될 수도 있습니다.)

유휴 대기

이 구성에서, 하나의 시스템은 DB2 인스턴스를 수행하기 위해 사용되며, 두 번째 시스템은『유휴』상태에 있거나 대기 모드에 있어서, 첫 번째 시스템을 포함하는 운영 체제 또는 하드웨어 실패가 있을 경우를 대비하여 인스턴스를 받을 준비가 되어 있습니다. 대기 시스템은 필요할 때까지만 유휴 상태이므로 전체 시스템 성능에는 영향이 미치지 않습니다.

상호 인계

이 구성에서, 각 시스템은 다른 시스템을 위해 지시된 백업입니다. 백업 시스템은 장애 복구 다음에 여분의 작업을 수행해야 하므로 전체 시스템 성능에 영향이 미칠 수도 있습니다. 고유 작업과 실패한 시스템에서 수행 중이던 작업을 수행해야 합니다.

장애 복구 전략을 사용하여 인스턴스, 파티션 또는 여러 논리 노드를 장애 복구할 수 있습니다.

온라인 분할 이중복사 및 일시중단된 입출력 지원을 통한 고가용성

*일시중단된 입출력*은 온라인 분할 이중복사 조절에 대한 완전한 구현을 제공하여 연속되는 시스템 가용성을 지원합니다. 즉, 데이터베이스를 종료하지 않고 이중복사를 분할합니다. *분할 이중복사*는 데이터를 포함하는 디스크를 이중복사하고 사본이 필요할 때 이중복사를 분할하여 만들 수 있는 데이터베이스의『인스턴스화』사본입니다. *디스크 이중복사*는 모든 데이터를 두 개의 구분된 하드 디스크(하나는 다른 디스크의 이중복사임)에 쓰는 프로세스입니다. 이중복사 분할은 이중복사의 백업 사본을 만드는 프로세스입니다.

DB2 백업 유틸리티를 사용하여 대형 데이터베이스를 백업할 것이 아니면, 일시중단된 입출력 및 분할 이중복사 기능을 사용하여 이중복사된 이미지에서 사본을 만들 수 있습니다. 또한 이 방식은 다음과 같은 역할을 합니다.

- 생산 머신에서 백업 조작 오버헤드를 없애줍니다.
- 빠른 시스템 복제 방법을 나타냅니다.

- 유휴 대기 장애 복구의 빠른 구현을 나타냅니다. 초기 복원 조작이 없고, 롤 포워드 조작이 너무 느린 것으로 판명되거나 오류가 발생할 경우, 재초기화가 매우 빠릅니다.

db2inidb 명령은 다음 목적으로 사용할 수 있도록 분할 이중복사를 초기화합니다.

- 복제 데이터베이스 작성을 위해

  1차 데이터베이스의 읽기 전용 복제본은 보고서를 작성하는 등의 작업에 사용할 수 있습니다.
- 대기 데이터베이스로
- 백업 이미지로

이 명령은 분할 해제 이중복사에 대해서만 발행할 수 있으며, 분할 해제 이중복사는 사용되기 전에 먼저 **db2inidb**를 수행해야 합니다(362 페이지의 『db2inidb - 이중복사 데이터베이스 초기화』 참조).

파티션된 데이터베이스 환경에서, **db2inidb** 명령은 임의의 파티션으로부터 분할 이미지를 사용하기 전에 모든 파티션에서 수행해야 합니다. 모든 파티션에서 동시에 도구를 수행할 수 있습니다.

복제 데이터베이스 작성

데이터베이스 복제는 1차(활동 중인) 데이터베이스의 오프라인 『백업』을 나타낼 수 있습니다. 그러나 복제된 데이터베이스 백업, 원래 시스템에서의 이미지 복원, 그리고 원래 시스템에서 생성된 로그 파일을 통한 롤 포워드는 수행할 수 없습니다.

데이터베이스를 복제하려면 다음 단계를 수행하십시오.

1. 1차 데이터베이스에서 입출력을 일시중단하십시오.

   ```
   db2 set write suspend for database
   ```
2. 적절한 운영 체제 레벨의 명령을 사용하여 1차 데이터베이스에서 이중복사를 분할하십시오.
3. 1차 데이터베이스에서 입출력을 재개하십시오.

   ```
   db2 set write resume for database
   ```
4. 다른 머신에서 이중복사된 데이터베이스에 접속하십시오.
5. 데이터베이스 인스턴스를 시작하십시오.

```
db2start
```

6. 이중복사된 데이터베이스를 1차 데이터베이스의 복제본으로 초기화하십시오.

   ```
   db2inidb database_alias as snapshot
   ```

   주: 이 명령은 분할될 때 불안정한 트랜잭션을 구간 복원합니다.

대기 데이터베이스로 분할 이중복사 사용

이중복사된(대기) 데이터베이스는 로그를 통해 계속 롤 포워드하므로, 1차 데이터베이스에 의해 작성되는 새로운 로그가 1차 시스템에서 계속 페치됩니다. 분할 이중복사를 대기 데이터베이스로 사용하려면, 다음 단계를 수행하십시오.

1. 1차 데이터베이스에서 입출력을 일시중단하십시오.

   ```
   db2 set write suspend for database
   ```

2. 해당되는 운영 체제 레벨의 명령을 사용하여 1차 데이터베이스에서 이중복사를 분할하십시오.
3. 1차 데이터베이스에서 입출력을 재개하십시오.

   ```
   db2 set write resume for database
   ```

4. 이중복사된 데이터베이스를 다른 인스턴스에 접속하십시오.
5. 이중복사된 데이터베이스를 롤 포워드 보류 상태에 놓으십시오.

   ```
   db2inidb database_alias as standby
   ```

   DMS 테이블 공간(데이터베이스 관리 공간)이 있으면, 전체 데이터베이스 백업을 사용하여 생산 데이터베이스에서 백업 사용에 따른 오버헤드를 줄일 수 있습니다.

6. 1차 시스템에서 가장 최근 로그 파일을 검색하도록 User Exit 프로그램을 설정하십시오.
7. 로그의 끝으로 데이터베이스를 롤 포워드하십시오.
8. 계속해서 로그 파일을 검색하고 1차 데이터베이스가 중단될 때까지 로그 끝까지 데이터베이스를 롤 포워드하십시오.

백업 이미지로 분할 이중복사 사용

분할 이중복사를 『백업 이미지』로 사용하려면, 다음 단계를 수행하십시오.

1. 1차 데이터베이스에서 입출력을 일시중단하십시오.

   ```
   db2 set write suspend for database
   ```

2. 해당되는 운영 체제 레벨의 명령을 사용하여 1차 데이터베이스에서 이중복사를 분할하십시오.
3. 1차 데이터베이스에서 입출력을 재개하십시오.

   ```
   db2 set write resume for database
   ```

4. 운영 체제 레벨의 명령을 사용하여 이중복사된 데이터와 로그를 1차 시스템에서 복사하십시오.
5. 데이터베이스 인스턴스를 시작하십시오.

   ```
   db2start
   ```

6. 분할 해제 디스크의 데이터를 다시 원래 시스템의 디스크로 복사하기 위해 사용할 수 있는 『백업 이미지』로 이중복사된 데이터베이스를 초기화하십시오. (롤 포워드 프로세스 동안에 로그가 필요하게 되므로, 로그 파일을 포함하는 파일 시스템은 다시 가져오지 마십시오.)

   ```
   db2inidb database_alias as mirror
   ```

7. 로그의 끝으로 데이터베이스(원래 시스템에 있는)를 롤 포워드하십시오.

제6장 AIX의 고가용성

향상된 확장성(ES)은 High Availability Cluster Multi-Processing (HACMP) for AIX의 기능입니다. 이 기능은 HACMP와 같은 장애 복구를 제공하며 똑같은 이벤트 구조를 가지고 있습니다(*HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide* 참조). 또한 향상된 확장성은 다음을 제공합니다.

- 더 커진 HACMP 클러스터
- *사용자 정의 이벤트*를 통한 추가 오류 범위. 모니터된 영역은 프로세스의 중단 또는 용량에 가까워지는 페이징 공간과 같은 다양한 사용자 정의 이벤트를 트리거할 수 있습니다. 이러한 이벤트는 필요에 따라 장애 복구 프로세스에 추가될 수 있는 사전 및 사후 이벤트를 포함합니다. 다른 구현에서 고유한 여분의 함수는 HACMP 사전 및 사후 이벤트 스트림에 올 수도 있습니다.

  규칙 파일(`/usr/sbin/cluster/events/rules.hacmprd`)이 HACMP 이벤트에 있습니다. 사용자 정의 이벤트가 이 파일에 추가됩니다. 이벤트가 발생할 때 수행될 스크립트 파일은 이 정의의 일부입니다.

  231 페이지의 『HACMP ES 이벤트 모니터링 및 사용자 정의 이벤트』에서 사용자 정의 이벤트와 규칙 파일에 대한 자세한 내용을 참조하십시오.
- HACMP 클러스터 밖에 있는 AIX 물리적 노드에서 일어나는 상태 변경(하나 이상의 클러스터)을 모니터하고 감지하기 위한 HACMP 클라이언트 유틸리티

HACMP ES 클러스터의 노드는 다른 노드에 각자의 사용 가능성을 알리는 *heartbeats* 또는 *keepalive 패킷*이라는 메시지를 교환합니다. 응답을 중단한 노드 때문에 클러스터의 나머지 노드는 복구를 호출하게 됩니다. 복구 프로세스는 *node_down 이벤트*라고 하며 *장애 복구*라고도 합니다. 복구 프로세스는 노드를 클러스터에 다시 통합시키기 위해 작업하는 것으로 완료됩니다. 이것은 *node_up 이벤트*라고도 합니다.

이벤트에는 두 가지 유형이 있습니다. 하나는 표준 이벤트로 HACMP ES의 조작에 포함되고, 다른 하나는 사용자 정의 이벤트로 하드웨어 및 소프트웨어 구성요소의 매개변수 모니터 작업에 관여합니다.

표준 이벤트 중 하나는 node_down 이벤트입니다. 복구 프로세스의 일부가 되는 작업을 계획할 때 HACMP에서는 두 가지 장애 복구 옵션을 제공합니다. 하나는 긴급(또는 유휴) 대기이고 다른 하나는 상호 인계입니다.

클러스터 구성

긴급 대기 구성에서 인계 노드인 AIX 프로세서 노드는 다른 표준 워크로드에서 수행되지 *않습니다*. *상호 인계* 구성에서 인계 노드인 AIX 프로세서 노드는 다른 표준 워크로드에서 *수행됩니다*.

일반적으로, DB2 Universal Database Enterprise - Extended Edition(UDB EEE)은 노드마다 파티션이 있는 상호 인계 모드에서 수행됩니다. 카탈로그 노드가 긴급 대기 구성의 일부인 시나리오는 예외입니다.

HACMP ES를 사용하여 RS/6000에서 규모가 큰 DB2를 설치할 계획이라면, RS/6000 SP 프레임 내 또는 그 사이에서 클러스터의 노드를 나누는 방법을 고려해야 합니다. 노드와 백업이 서로 다른 SP 프레임에 있을 경우, 하나의 프레임이 정지(즉, 프레임 전원/스위치 보드가 고장남)하면, 인계가 일어납니다. 단, 이러한 고장은 각 SP 프레임에 *N*+1 전원 장치가 있고 각 SP 스위치마다 *N*+1 팬 및 전원 장치와 함께 중복되는 경로가 있기 때문에 아주 드물게 일어납니다. 프레임 고장의 경우, 나머지 프레임을 복구하기 위해 수동 개입이 필요하게 될 수 있습니다. 이 복구 프로시듀어는 SP 관리 안내서에 있습니다. HACMP ES는 SP 노드 고장을 복구합니다. 프레임 고장 복구는 SP 프레임에 있는 적절한 클러스터 배치와 관계가 있습니다.

계획시 다른 고려사항에는 큰 클러스터를 관리하는 방법이 포함됩니다. 큰 클러스터보다 작은 클러스터를 관리하는 것이 더 용이합니다. 그러나 작은 클러스터 여러 개보다 큰 클러스터 하나를 관리하는 것이 더 쉽습니다. 계획시 클러스터 환경에서 응용프로그램이 어떻게 사용되는지에 대해 고려해 보십시오. 만일 커다란 동종의 응용프로그램 하나가, 예를 들어 16 노드에서 수행 중이라면, 8개의 2 노드 클러스터보다는 하나의 클러스터로 관리하는 것이 더 쉬울 것입니다. 같은 16 노

드에 서로 다른 네트워크, 디스크, 그리고 노드 관계가 있는 서로 다른 응용프로그램이 여러 개 있다면, 노드를 작은 클러스터로 그룹 짓는 것이 더 낫습니다. 노드는 한 번에 하나씩 HACMP 클러스터에 통합된다는 것을 기억해 두십시오. 커다란 클러스터 하나의 구성보다는 여러 개의 클러스터의 구성을 시작하는 것이 더 빠릅니다. HACMP ES는 노드와 백업이 같은 클러스터에 있는 한 클러스터 하나 및 여러 개를 모두 지원합니다.

HACMP ES 장애 복구에서는 자원 그룹의 물리적 노드 지정을 사전에 정의할 수 있습니다(*연쇄*라고도 함). 장애 복구 프로시듀어는 또한 자원 그룹을 물리적 노드에 부동(또는 *순환*) 지정할 수 있습니다. 각 지원 그룹의 IP 주소, 외부 디스크 볼륨 그룹, 파일 시스템, NFS 파일 시스템 및 응용프로그램 서버는 장애 복구 또는 재통합에 의한 물리적 노드간의 HACMP ES에 의해 처리될 수 있는 응용프로그램 또는 응용프로그램 구성요소를 지정합니다. 장애 복구 또는 재통합 동작은 작성된 자원 그룹의 유형 및 자원 그룹에 있는 노드의 수에 의해 지정됩니다.

예를 들면, DB2 데이터베이스 파티션(논리 노드)을 고려해 보십시오. 로그 및 테이블 공간 컨테이너가 외부 디스크에 놓이고 기타 노드가 해당 디스크로 링크되면, 기타 노드는 이들 디스크에 액세스하고 데이터베이스 파티션을 재시작할 수 있게 됩니다(인계 노드에서). HACMP에 의해 자동화되는 조작 유형이 바로 이것입니다. HACMP ES는 또한 DB2 인스턴스 주 사용자 디렉토리에 의해 사용된 NFS 파일 시스템을 복구하기 위해서도 사용됩니다.

DB2 UDB EEE와의 복구 계획의 일부로 HACMP ES 문서를 자세히 읽어 보십시오. 개념, 계획, 설치 및 관리 안내서를 읽은 다음, 사용자 환경의 복구 아키텍처를 빌드해야 합니다. 식별된 장애 복구 시점을 근거로 복구를 위해 식별된 각 서브시스템에 필요한 HACMP 클러스터와 각각에 대한 복구 노드(긴급 대기 또는 상호 인계)를 식별하십시오. 이것은 책에 포함된 HACMP 워크쉬트를 완료하기 위한 시작점이 됩니다.

외부 디스크 구성에서 디스크와 어댑터를 둘다 이중복사되는 것이 매우 바람직합니다. HACMP에 대해 구성된 DB2 물리 노드에 대해 노드가 외부 공유 디스크의 볼륨 그룹에서 달라질 수 있도록 주의해야 합니다. 상호 인계 구성에서 이 배열은 쌍을 이룬 노드가 서로간의 볼륨 그룹을 충돌 없이 액세스할 수 있도록 추가

계획이 필요합니다. 이는 DB2 UDB EEE에서 모든 컨테이너 이름이 모든 데이터베이스에서 고유해야 한다는 것을 의미합니다.

고유성을 확보하는 한 가지 방법은 파티션 번호를 이름의 일부로 포함시키는 것입니다. SMS 또는 DMS 컨테이너를 작성할 때 컨테이너 문자열 구문에 대한 노드를 지정할 수 있습니다. 표현식을 지정할 때, 노드 번호는 컨테이너 이름의 일부가 될 수 있으며, 인수를 추가로 지정할 때에는 인수의 결과가 컨테이너 이름의 일부가 됩니다. 인수 ″ $N″ ([blank]$N)를 사용하여 노드 표현식을 나타내십시오. 인수는 컨테이너 문자열 끝에 와야 하고 다음 양식 중 하나에만 사용됩니다.

표 11. 컨테이너 작성 인수. 노드 번호는 5로 간주됩니다.

구문	예	값
[blank]$N	″ $N″	5
[blank]$N+[number]	″ $N+1011″	1016
[blank]$N%[number]	″ $N%3″	2
[blank]$N+[number]%[number]	″ $N+12%13″	4
[blank]$N%[number]+[number]	″ $N%3+20″	22
주: 1. %는 계수입니다. 2. 모든 경우, 연산자는 왼쪽에서 오른쪽으로 계산됩니다.		

다음은 이 특수 인수를 사용한 컨테이너 작성 방법의 몇 가지 예입니다.

- 2 노드 시스템에서 사용하기 위한 컨테이너 작성

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

  다음 컨테이너는 다음과 같이 사용됩니다.

```
/dev/rcont0   - on Node 0
/dev/rcont1   - on Node 1
```

- 4 노드 시스템에서 사용하기 위한 컨테이너 작성

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

  다음 컨테이너는 다음과 같이 사용됩니다.

```
/DB2/containers/TS2/container100   - on Node 0
/DB2/containers/TS2/container101   - on Node 1
/DB2/containers/TS2/container102   - on Node 2
/DB2/containers/TS2/container103   - on Node 3
```

- 2 노드 시스템에서 사용하기 위한 컨테이너 작성

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

다음 컨테이너는 다음과 같이 사용됩니다.

```
/TS3/cont0   - on Node 0
/TS3/cont2   - on Node 0
/TS3/cont1   - on Node 1
/TS3/cont3   - on Node 1
```

214 페이지의 그림18 및 215 페이지의 그림19에서는 DB2 SSA I/O 서브시스템 구성의 예와 고가용성의 외부 디스크 구성을 확인하는 계획의 일부와 충돌 없이 모든 볼륨 그룹에 액세스하는 능력을 보여줍니다.

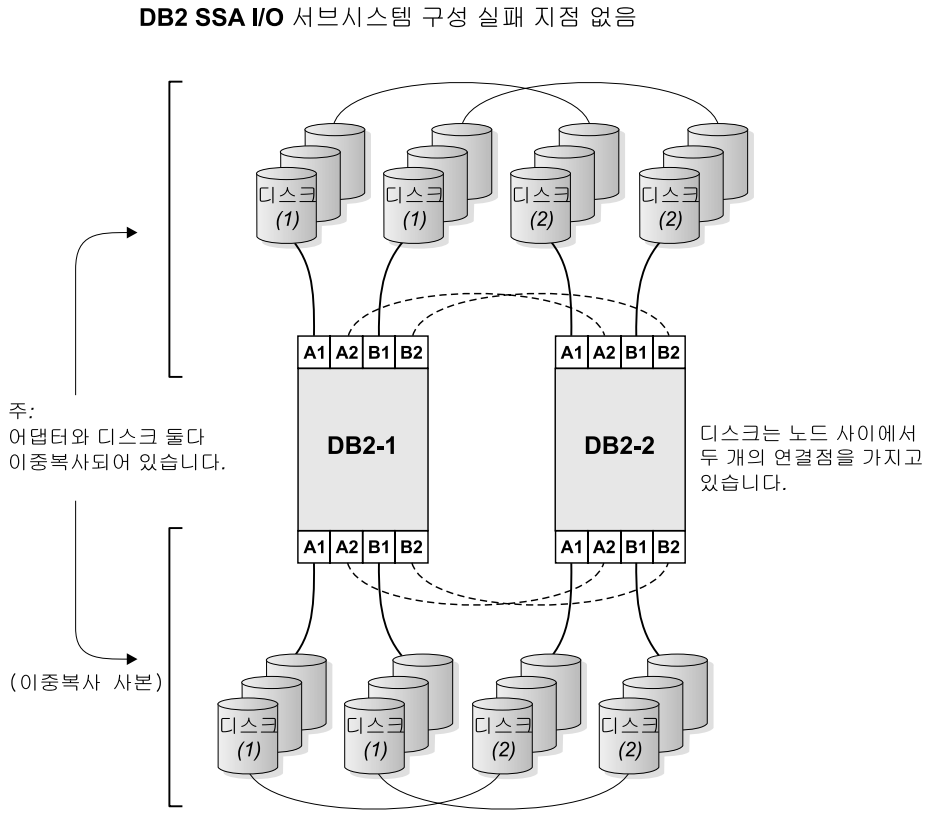


그림 18. 장애 복구 시점이 전혀 없는 경우

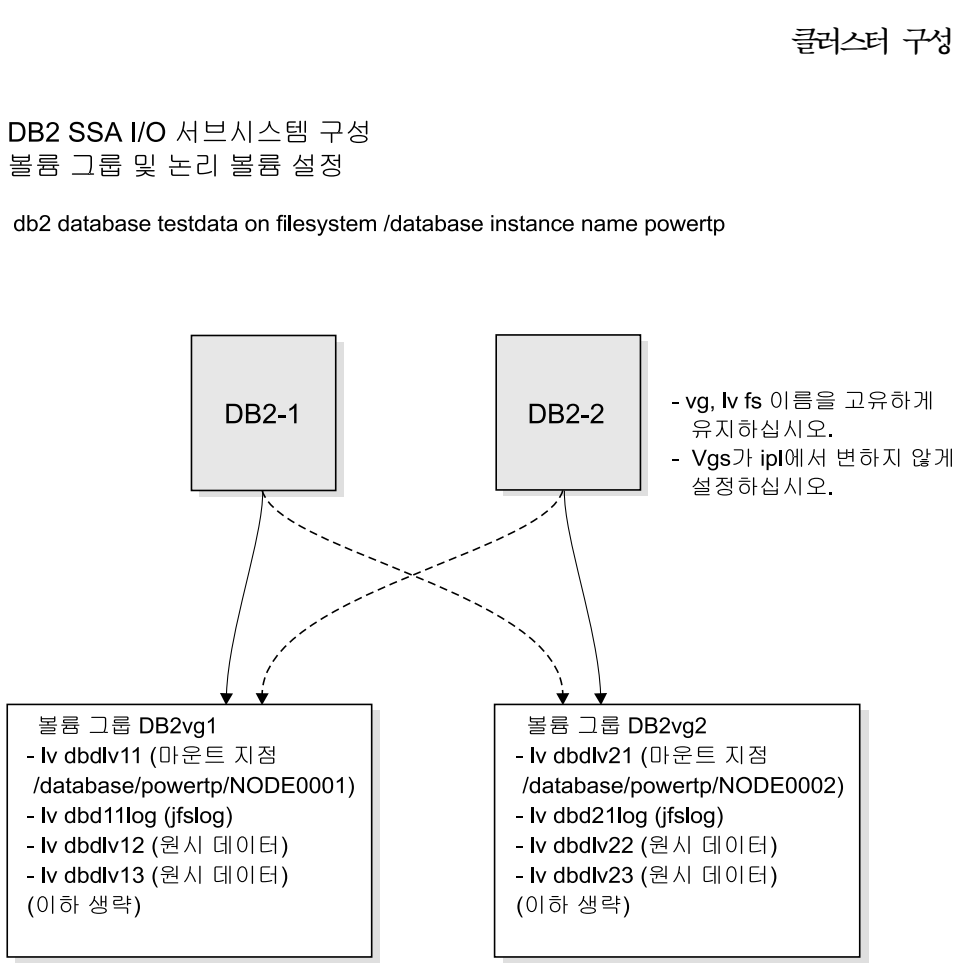


그림 19. 볼륨 그룹 및 논리 볼륨 설정

DB2 데이터베이스 파티션 구성하기

일단 구성되면, 인스턴스의 각 데이터베이스 파티션은 HACMP ES에 의해 한 번에 하나씩 물리적 노드가 시작됩니다. 4 노드보다 많은 병렬 DB2 구성을 시작할 때에는 다중 클러스터를 사용하는 것이 바람직합니다. 64-노드 병렬 DB2 구성에서는 32개의 2-노드 HACMP 클러스터를 병렬로 시작하는 것이 4개의 16-노드 클러스터를 시작하는 것보다 빠릅니다.

`rc.db2pe` 스크립트 파일은 긴급 대기 또는 상호 인계 노드에서 HACMP ES 장애 복구 또는 복구 구성을 지원하기 위해 DB2 UDB EEE와 함께 패키지화됩니다. (그리고 `/usr/bin`의 각 노드에 설치됩니다.) 이 외에도, DB2 버퍼 풀 크기

는 rc.db2pe의 상호 인계 구성에서 일어나는 장애 복구 중에 사용자 정의할 수 있습니다. (두 개의 데이터베이스 파티션이 하나의 물리적 노드에서 수행될 때 적절한 성능을 보장하기 위해 버퍼 풀 *크기*를 구성해야 합니다.)

DB2 데이터베이스 파티션의 HACMP 구성에서 응용프로그램 서버를 작성할 때, 다음과 같이 rc.db2pe를 시작 및 중단 스크립트로 지정하십시오.

```
   /usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> start <use switch>
   /usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> stop <use switch>
여기서,
   <instance>는 인스턴스 이름입니다.
   <dpn>은 데이터베이스 파티션 번호입니다.
   <secondary dpn>은 상호 인계 구성에서만 "연관" 데이터베이스 파티션 번호입니다.
      기급 대기 구성에서는 <dpn>과 동일합니다
   <use switch>는 일반적으로 공백입니다.
      db2nodes.cfg 파일에서 SP 교환 네트워크가 hostname으로 사용됨을 표시하며
      (모든 DB2 트래픽은 SP 교환으로 경로지정됨), 공백이 아닐 경우, 사용된
      이름은 사용될 SP 노드의 호스트 이름입니다.
```

DB2 명령인 LIST DATABASE DIRECTORY는 rc.db2pe에서 이 데이터베이스 파티션에 대해 구성된 데이터베이스를 모두 찾기 위해 사용됩니다. 스크립트 파일은 /usr/bin/reg.parms.*DATABASE* 파일 및 /usr/bin/failover.parms.*DATABASE* 파일을 찾는데, 여기서 *DATABASE*는 이 데이터베이스 파티션에 대해 구성된 각 데이터베이스입니다. 상호 인계 구성에서, 매개변수 파일 reg.parms.*xxx* 및 failover.parms.*xxx*를 작성하는 것이 바람직합니다. failover.parms.*xxx* 파일에서 버퍼 풀에 영향을 주는 BUFFPAGE, DBHEAP 및 기타에 대한 설정값은 버퍼 풀이 둘 이상일 수 있는 가능성에 대해 조정되어야 합니다. 사용자는 reg.parms.SAMPLE과 failover.parms.SAMPLE 샘플 파일을 사용할 수 있습니다.

이 환경에서 중요한 매개변수 중 하나는 10분의 기본값을 가진 *start_stop_time* 데이터베이스 관리 프로그램 매개변수입니다. 그러나 rc.db2pe는 이 매개변수를 2분으로 설정합니다. rc.db2pe를 통해 이 매개변수를 10분이나 그 보다 약간 더 큰 값으로 설정해야 합니다. 이 문맥에서, 지정된 지속기간은 파티션의 실패와 복구 간의 시간 간격입니다. 파티션에서 실행하는 응용프로그램이 빈번한 COMMIT를 발행하는 경우, 데이터베이스 파티션에서 실패 이후의 10분은 확약되지 않은 트랜잭션을 구간 복원하고 해당 파티션에서 데이터베이스의 일관성 지점에 도달하는 데 충분한 시간이어야 합니다. 워크로드가 많거나, 여러 파티션이 있는 경우, 구간 복원 작업이 완료되기 전에 발생하는 시간종료의 확률을 감소시키려면 지속기간을 증가시키는 것이 필요할 수 있습니다.

다음은 긴급 대기 구성 및 상호 인계 구성의 예입니다. 두 예에서 서비스 IP 스위치 별명 주소가 자원 그룹에 있습니다. 이 스위치 별명 주소는 다음의 경우에 사용됩니다.

- DB2 인스턴스 소유자 파일 시스템에 대한 파일 서버의 NFS 액세스
- 장애 복구, Tivoli Storage Manager(TSM, 이전의 ADSM) 연결 또는 다른 유사한 조작의 경우 유지보수되어야 하는 기타 클라이언트 액세스.

구현시 이들 별명이 필요하지 않을 때에는 이를 제거할 수 있습니다. 제거되면, 반드시 *MOUNT_NFS* 매개변수를 rc.db2pe 스크립트 파일에서 NO로 설정하십시오.

긴급 대기 구성의 예

이 예에서의 가정은 물리적 노드 1 및 2 간에 긴급 대기 구성이 있으며, DB2 인스턴스 이름은 POWERTP이라는 것입니다. 데이터베이스 파티션은 1이고, 데이터베이스는 /database 파일 시스템에서 TESTDATA입니다.

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< 이것은 스위치 별명 주소입니다.)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 1 stop
```

상호 인계 구성의 예

이 예에서의 가정은 물리적 노드 1 및 2 간에 상호 인계 구성이 있으며, DB2 인스턴스 이름은 POWERTP이라는 것입니다. 데이터베이스 파티션은 1 및 2이고, 데이터베이스는 /database 파일 시스템에서 TESTDATA입니다.

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< 이것은 스위치 별명 주소입니다.)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 2 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 2 stop
Resource group name: db2_pd_2
Node Relationship: cascading
Participating nodenames: node2_eth, node1_eth
Service_IP_label: nfs_switch_2     (<<< 이것은 스위치 별명 주소입니다.)
Filesystems: /database/powertp/NODE0002
Volume Groups: DB2vg2
```

```
Application Servers: db2_dp2_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 2 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 2 1 stop
```

NFS 서버 노드의 구성

rc.db2pe 스크립트는 DB2 병렬 처리 인스턴스 사용자 디렉토리의 NFS 마운트 디렉토리를 사용할 수 있도록 이용됩니다. 이는 rc.db2pe 스크립트 파일에서 *MOUNT_NFS* 매개변수를 YES로 설정하고 NFS 실패 서버 쌍을 다음과 같이 구성하여 이루어질 수 있습니다.

- 홈 디렉토리를 구성하고 이를 /etc/exports와 **exportfs** 명령을 사용하여 "루트"로 NFS 서버의 IP 조소와 같은 서브네트의 노드에 사용되는 IP 주소로 내보내십시오. HACMP 부트와 서비스 주소를 둘다 포함하십시오. NFS 서버의 IP 주소는 백업으로 인계될 수 있는 HACMP 서비스 주소와 같은 주소입니다. DB2 인스턴스 소유자의 홈 디렉토리는 직접 NFS 마운트되어야 합니다. (스크립트는 자동 마운터의 사용을 DB2 인스턴스 소유자 홈 디렉토리로 지원하지 않습니다.)
- SMIT 또는 현실적인 구성을 사용할 때에는 파일 서버를 포함한 DB2 병렬처리 그룹 짓기에서 모든 노드가 NFS 파일 시스템 명령을 사용하여 마운트될 수 있도록 이 파일 시스템에 대해 별도의 /etc/filesystems 항목을 작성하십시오.

  예를 들어, /nfshome JFS 파일 시스템은 /dbhome의 모든 노드로 내보낼 수 있습니다. 각 노드는 nfs_server:/nfshome이라고 하는 NFS 파일 시스템 /dbname을 작성합니다. 그러므로 인스턴스 이름이 "powertp"이면 DB2 인스턴스 소유자의 홈 디렉토리는 /dbhome/powertp가 됩니다.

  /etc/filesystems에서 마운트할 NFS 매개변수가 "hard", "bg", "intr" 및 "rw"인지 확인하십시오.
- /etc/passwd의 /dbhome/powertp 홈 디렉토리와 관련된 DB2 인스턴스 소유자 정의는 모든 노드에서 같아야 합니다.

  사용자 정의는 SP 환경의 제어 워크스테이션에 주로 작성되고 "supper" 또는 "pcp"는 /etc/passwd, /etc/security/passwd, /etc/security/user 및 /etc/security/group을 모든 노드에 분배하기 위해 사용됩니다.

- 내보내기되는 볼륨 그룹과 파일 시스템에 대한 HACMP 자원 그룹에서 "내보낼 nfs_filesystems"를 구성하지 *마십시오*. 대신, 이를 NFS에서 정상적으로 구성하십시오. NFS 서버에 대한 스크립트는 파일 시스템의 내보내기를 제어합니다.
- 파일 시스템이 상주하는 볼륨 그룹의 주요 번호가 기본 노드와 인계 노드에서 모두 같아야 합니다. 이는 `-V` 옵션이 있는 **importvg**를 사용하여 이루어집니다.
- *MOUNT_NFS* 매개변수가 `rc.db2pe` 스크립트 파일에서 `YES`로 설정되고 각 노드에 `/etc/filesystems`로 마운트될 NFS 파일 시스템이 있는지를 검증하십시오. 만일 그렇지 않으면, `rc.db2pe`는 파일 시스템을 마운트하지 못하고 DB2를 시작하지 못합니다.
- 만일 DB2 인스턴스 소유자가 이미 작성되어 있고 사용자의 디렉토리 구조를 작성 중인 파일 시스템에 복사하고 있다면, 반드시 디렉토리를 **tar**(`-cvf`)하십시오. 그러면 기호 링크를 보존할 수 있습니다.
- 논리 볼륨과 작성 중인 파일 시스템의 파일 시스템 로그에 대해 어댑터와 디스크 둘다 이중복사되어야 합니다.

NFS 서버 인계 구성의 예

이 예에서는 "nfs_server" IP 주소상의 `nfsvg` 볼륨 그룹에 `/nfshome` NFS 서버 파일 시스템이 있다고 가정하십시오. DB2 인스턴스 이름은 POWERTP이고, 홈 디렉토리는 `/dbhome/powertp`입니다.

```
Resource group name: nfs_server
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_server     (<<< 이것은 스위치 별명 주소입니다.)
Filesystems: /nfshome
Volume Groups: nfsvg
Application Servers: nfs_server_app
Application Server Start Script: /usr/bin/rc.db2pe powertp NFS SERVER start
Application Server Stop Script: /usr/bin/rc.db2pe powertp NFS SERVER stop
```

이 예에서,

- 모든 노드의 `/etc/filesystems`에는 `nfs_server:/nfshome` 마운트용으로 `/dbhome`에 대한 항목이 들어 있습니다. `nfs_server`는 서비스 IP 스위치 별명 주소입니다.

- nfs_server 노드 및 백업 노드에 있는 /etc/exports에는 부트 및 서비스 주소가 포함되고 /nfsfs -root=nfs_switch_1, nfs_switch_2, ...에 대한 항목도 들어 있습니다.

SP 스위치 구성시 고려사항

HACMP ES를 SP 스위치로 구현할 때, 다음을 고려하십시오.

- SP 스위치에 "기본" 및 "별명" 주소가 있습니다. 기본 주소는 SP 시스템 데이터 저장소(SDR)에 정의되고 시스템이 "부트"되면 rc.switch에 의해 구성됩니다. 별명 주소는 기본 주소 외에 별명 속성을 가진 ifconfig 명령을 통해 **css0** 인터페이스에 구성된 IP 주소입니다. 예를 들면, 다음과 같습니다.

  ```
  ifconfig css0 inet alias sw_alias_1 up
  ```

- DB2 db2nodes.cfg 파일 구성시, SP 스위치 "기본" IP 주소 이름은 "호스트 이름"과 "네트 이름" 필드 둘다에 사용되어야 합니다. 스위치 IP 주소 별명은 NFS 연결성을 유지보수하기 *위해서만* 사용됩니다. DB2 장애 복구는 **db2start**(RESTART) 명령(db2nodes.cfg 갱신)으로 DB2를 재시작하여 수행됩니다.
- 스위치 주소와 etc/hosts 별명을 혼동하지 마십시오. SP 스위치 주소와 SP 스위치 별명 주소는 etc/hosts 또는 DNS에서 실수입니다. 스위치 주소 별명은 SP 스위치 기본 주소의 다른 이름이 아닙니다. 각각 자체 주소를 가지고 있습니다.
- SP 스위치 기본 주소는 사용되지 않을 때 항상 노드에 표시됩니다. HACMP ES는 노드 사이에서 이들 주소를 구성하거나 이동시키지 않습니다.
- SP 스위치 별명 주소를 사용하려면, 이를 HACMP에서 부트로 구성하고 "heartbeating"과 IP 주소 인계에 대한 서비스 주소로 구성하십시오. SP 스위치 별명을 사용하지 않으려면, 기본 SP 스위치 주소를 "heartbeating"에 *대해서만* 서비스 주소로 HACMP에 구성하십시오(IP 주소 인계 없음). 어떠한 구성에서도 별명 주소 *및* 스위치 기본 주소를 구성하지 마십시오. HACMP ES는 이 구성을 지원하지 않습니다.
- SP 스위치 별명 주소만이(SP 스위치 기본 주소가 아닌) IP 인계 구성을 위한 노드 간에 이동됩니다.

- SP 스위치 별명은 한 노드당 SP 스위치 어댑터가 하나뿐이므로 더욱 필요합니다. 별명 주소를 사용하면, 한 노드가 다른 스위치 어댑터를 추가하지 않고도 다른 노드의 스위치 별명 IP 주소를 인계할 수 있습니다. 이는 "슬롯에 제한된" 노드에서 유용합니다. SP 스위치 어댑터 고장의 복구 조절에 관한 자세한 정보는 235 페이지의 『HACMP ES 스크립트 파일』의 네트워크 실패를 참조하십시오.
- IP 주소 인계에 대해 SP 스위치를 구성하는 경우, 한 노드당 여분의 별명 IP 주소를 두 개씩 작성해야 합니다. 하나는 부트 주소용이고 다른 하나는 서비스 주소용입니다.
- SP 스위치 기본 IP 주소 또는 SP 스위치 별명 IP 주소에 대한 HACMP ES 네트워크 이름 정의에서 반드시 "HPS"를 사용하십시오.
- HACMP의 `rc.cluster`는 HACMP가 시작되면 SP 스위치 부트 주소에서 자동으로 **ifconfig**합니다. IP 주소 및 이름 작성과 이를 HACMP에서 정의하는 것 외의 추가 구성은 필요없습니다.
- SP 스위치의 Eprimary 노드는 Estart, Efence 및 Eunfence 명령을 구현하는 서버입니다. HACMP 스크립트는 HACMP가 시작될 때 노드를 Eunfence 또는 Estart하려고 하고 스위치가 네트워크 중 하나로 정의된 경우 스위치를 사용 가능하게 만듭니다. 이러한 이유로, HACMP를 시작할 때 반드시 Eprimary 노드를 사용할 수 있어야 합니다. HACMP 코드는 오류와 함께 종료하기 전에 Eprimary 장애 복구가 완료할 때까지 12분을 기다립니다.
- SP 스위치의 Eprimary 노드는 HACMP가 아닌 SP 병렬 시스템 지원 프로그램(PSSP)에 의해 노드 사이에서 이동됩니다. Eprimary 노드가 오프 라인이 되면, PSSP에는 자동으로 Eprimary 노드로 동작하는 백업 노드가 있습니다. 스위치 네트워크는 이 변경으로 영향을 받지 않고 정지된 상태로 남습니다.

DB2 HACMP 구성 예

다음 예에서는 여러 가지 장애 복구 지원 구성과 장애가 일어났을 때 발생하는 현상을 보여줍니다.

DB2 HACMP 상호 인계 구성(223 페이지의 그림20, 224 페이지의 그림21 및 225 페이지의 그림22)의 경우,

- HACMP 어댑터는 이더넷에 대해 정의되고 기본 주소인 SP 스위치 별명 부트 및 서비스 별명은 그대로 있습니다. HACMP 네트워크 이름에서 반드시 ″HPS″ 문자열을 사용하십시오.
- NFS_server/nfshome은 스위치 별명을 통해 모든 노드에 /dbhome으로 마운트됩니다.
- db2nodes.cfg 파일에는 SP 스위치 기본 주소가 있습니다. db2nodes.cfg 파일은 DB2 데이터베이스 파티션(논리 노드) 장애 복구 이후 **db2start**(RESTART) 명령에 의해 변경됩니다.
- SP 스위치 별명 부트 주소는 표시되지 않습니다.
- 노드는 서로 다른 SP 프레임에 있을 수 있습니다.

NFS 장애 복구가 있는 DB2 HACMP 상호 인계 - 정상

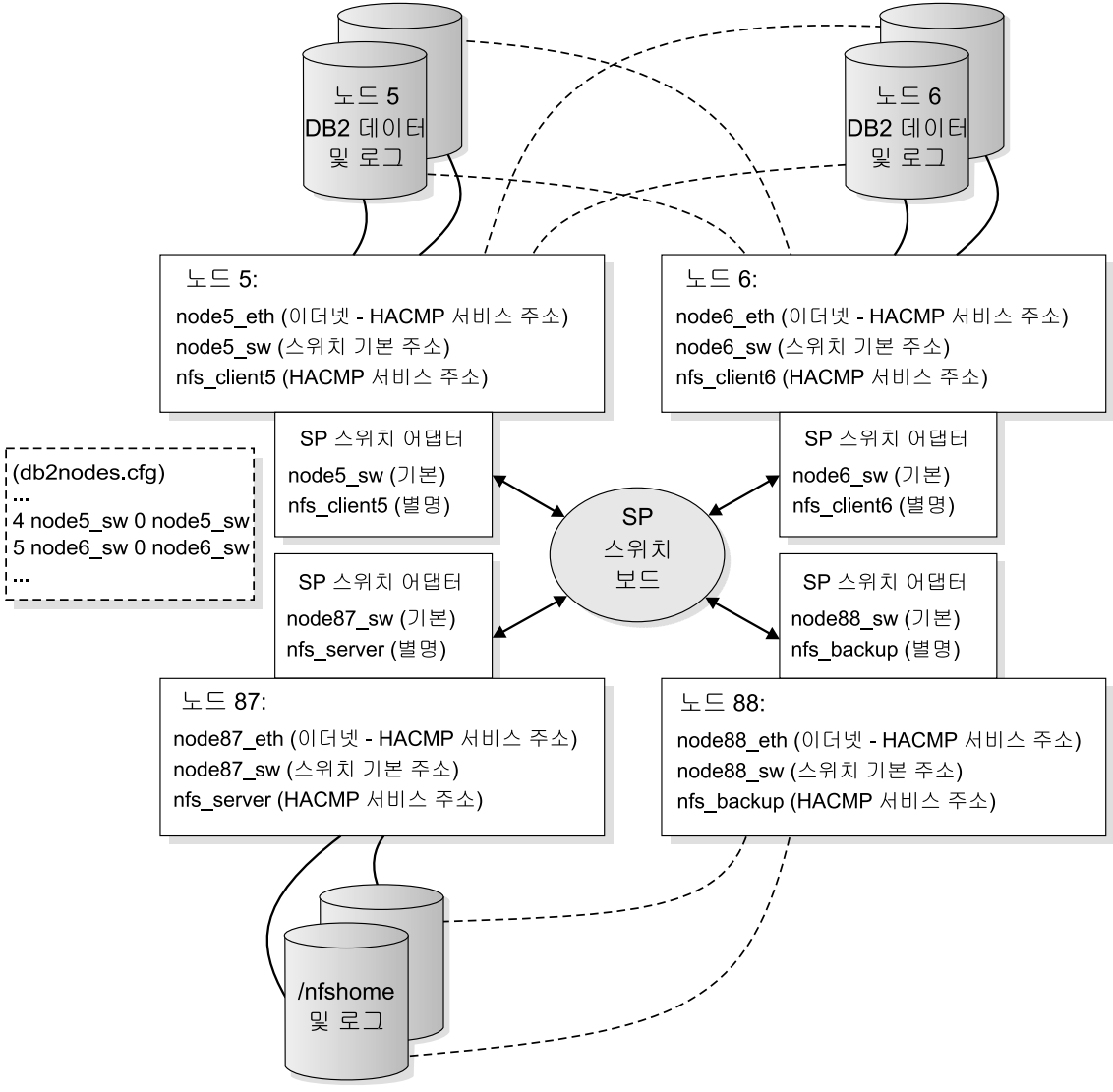


그림 20. NFS 장애 복구가 있는 상호 인계 - 정상

NFS 장애 복구가 있는 DB2 HACMP 상호인계 - NFS 장애 복구

- nfs_server SP 스위치 별명 IP 주소 및 nfs 마운트된/nfshome이 노드 87에서 88로 이동했습니다.
- SP 스위치 arp 코드는 이동시 모든 스위치 arp 캐쉬를 갱신하는 기능을 갖습니다.

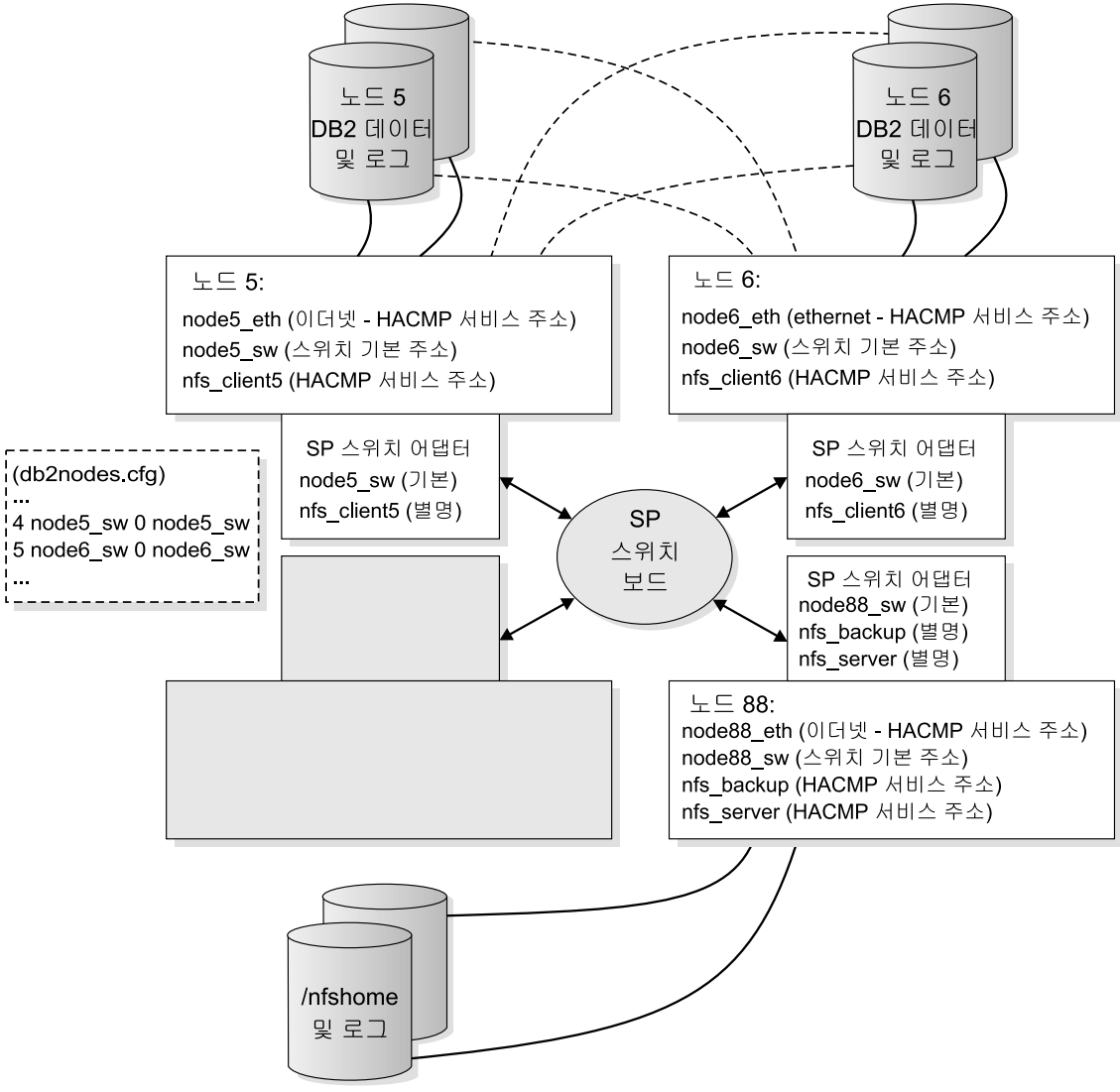


그림 21. NFS 장애 복구가 있는 상호 인계 - NFS 장애 복구

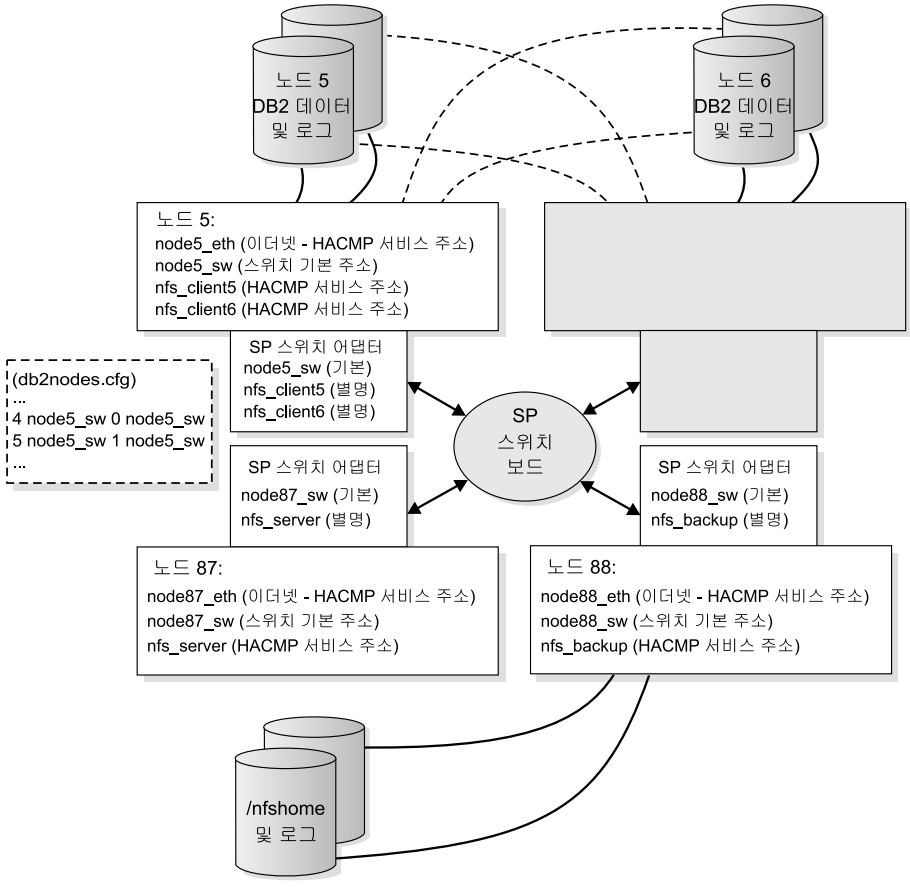


그림 22. NFS 장애 복구가 있는 상호 인계 - DB2 장애 복구

DB2 HACMP 긴급 대기 구성(227 페이지의 그림23 및 228 페이지의 그림24)의 경우,

- HACMP 어댑터는 이더넷에 대해 정의되고 기본 주소인 SP 스위치 별명 부트 및 서비스 별명은 그대로 있습니다. HACMP 네트워크 이름에서 반드시 "HPS" 문자열을 사용하십시오.
- `NFS_server/nfshome`은 스위치 별명을 통해 모든 노드에 `/dbhome`으로 마운트됩니다.

- db2nodes.cfg 파일에는 SP 스위치 기본 주소가 있습니다. db2nodes.cfg 파일은 DB2 데이터베이스 파티션(논리 노드) 장애 복구 이후 **db2start**(RESTART) 명령에 의해 변경됩니다.
- SP 스위치 별명 부트 주소는 표시되지 않습니다.

NFS 장애 복구가 있는 DB2 HACMP 긴급대기 - 정상

주: 긴급대기 노드는 디스크 케이블에 따라 둘 이상의 노드를 백업할 수 있습니다.

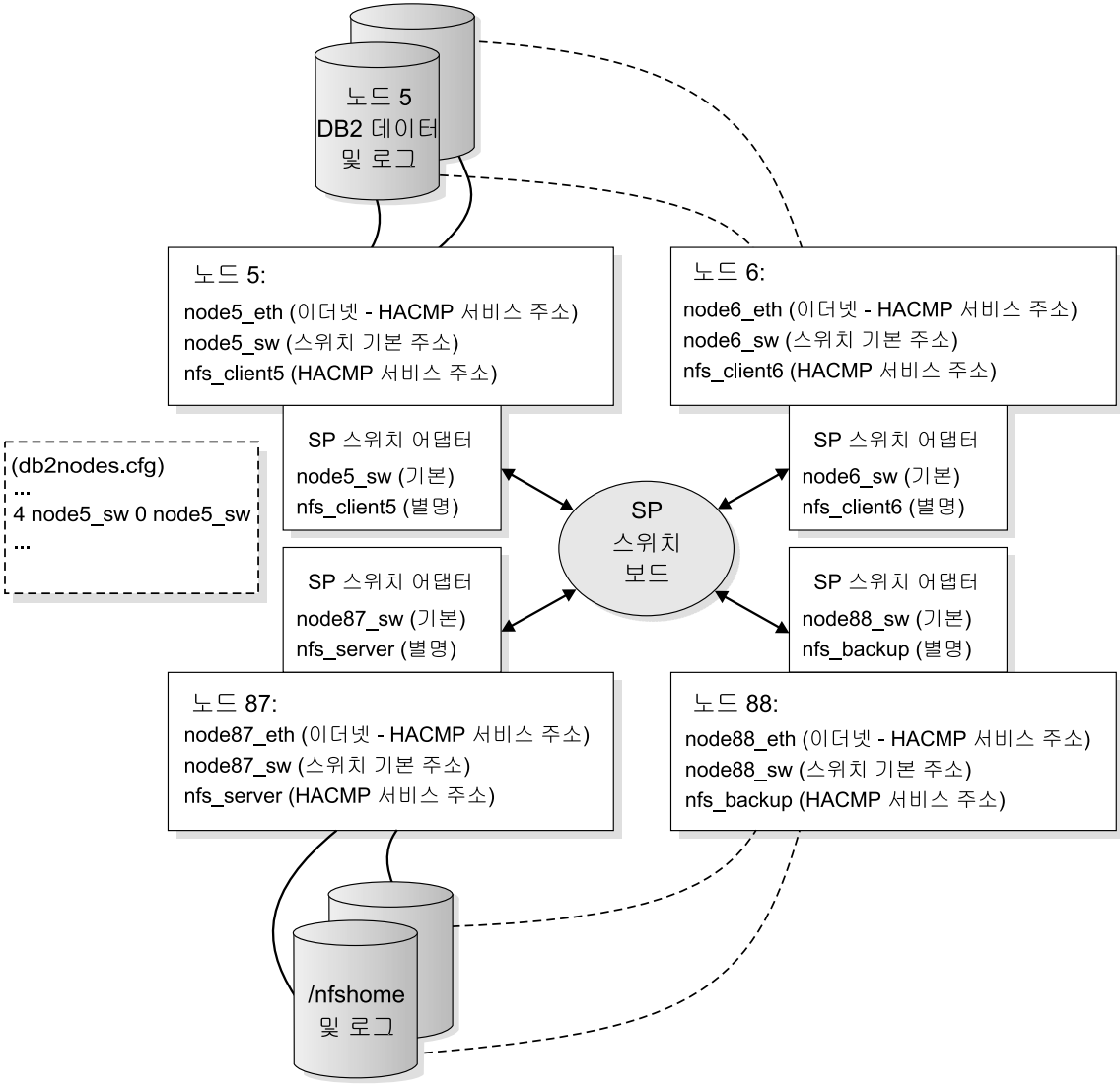


그림 23. NFS 장애 복구가 있는 긴급 대기 - 정상

NFS 장애 복구가 있는 DB2 HACMP 긴급대기 - DB2 장애 복구

주: 노드는 디스크 케이블에 따라 둘 이상의 노드를 백업할 수 있습니다.

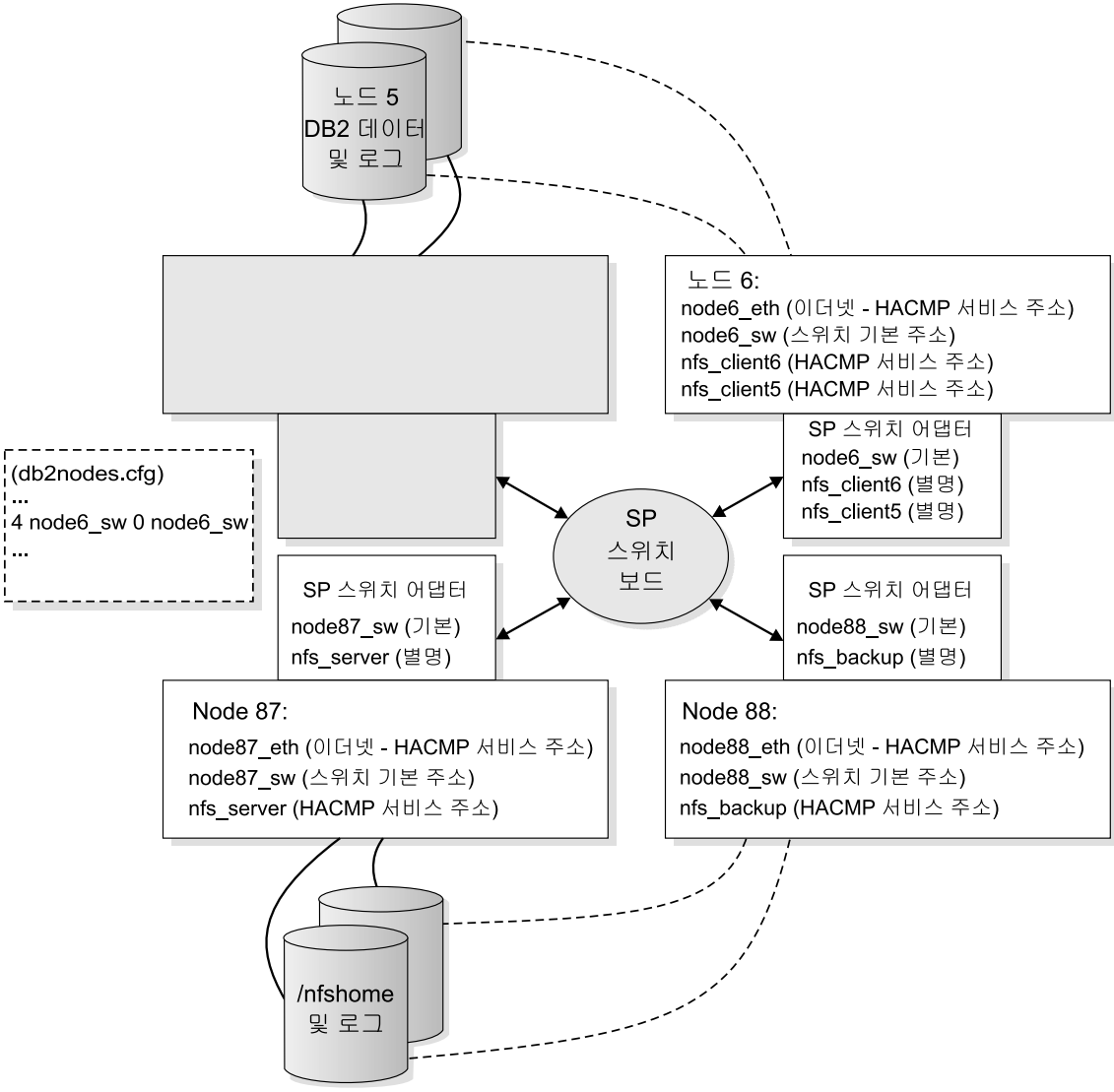


그림 24. NFS 장애 복구가 있는 긴급 대기 - DB2 장애 복구

NFS 장애 복구 구성이 없는 DB2 HACMP 상호 인계(229 페이지의 그림25 및 230 페이지의 그림26)의 경우,

- HACMP 어댑터는 이더넷과 SP 스위치 기본 주소에 대해 정의됩니다. 기본 주소가 HACMP에서 서비스 주소로 구성되면, 부트 주소가 없다는 점을 기억하십시오("heartbeat" 만 있음). SP 스위치에 대한 HACMP 네트워크 이름에서 반드시 "HPS" 문자열을 사용하십시오.
- db2nodes.cfg 파일에는 SP 스위치 기본 주소가 있습니다. db2nodes.cfg 파일은 DB2 데이터베이스 파티션(논리 노드) 장애 복구 이후 **db2start**(RESTART) 명령에 의해 변경됩니다.
- 표시된 NFS 장애 복구 기능이 없습니다.
- 노드는 서로 다른 SP 프레임에 있을 수 있습니다.

NFS 장애 복구가 없는 DB2 HACMP 상호 인계 - 정상

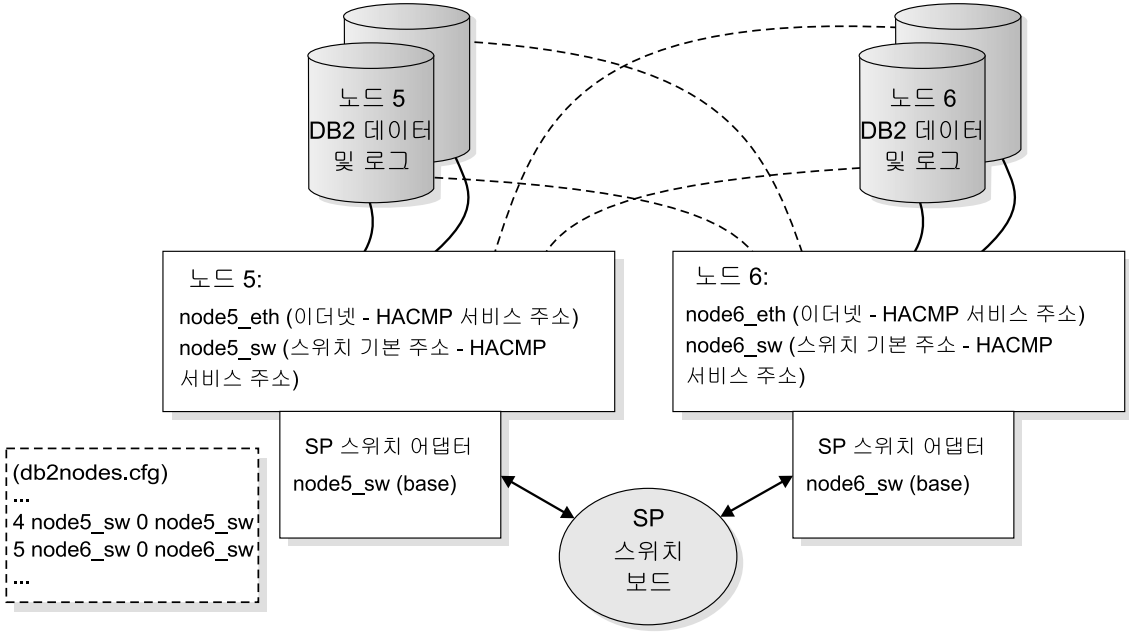


그림 25. NFS 장애 복구가 없는 상호 인계 - 정상

NFS 장애 복구가 없는 DB2 HACMP 상호인계 - DB2 장애 복구

- 노드 5는 2개의 DB2 논리 노드를 수행합니다.

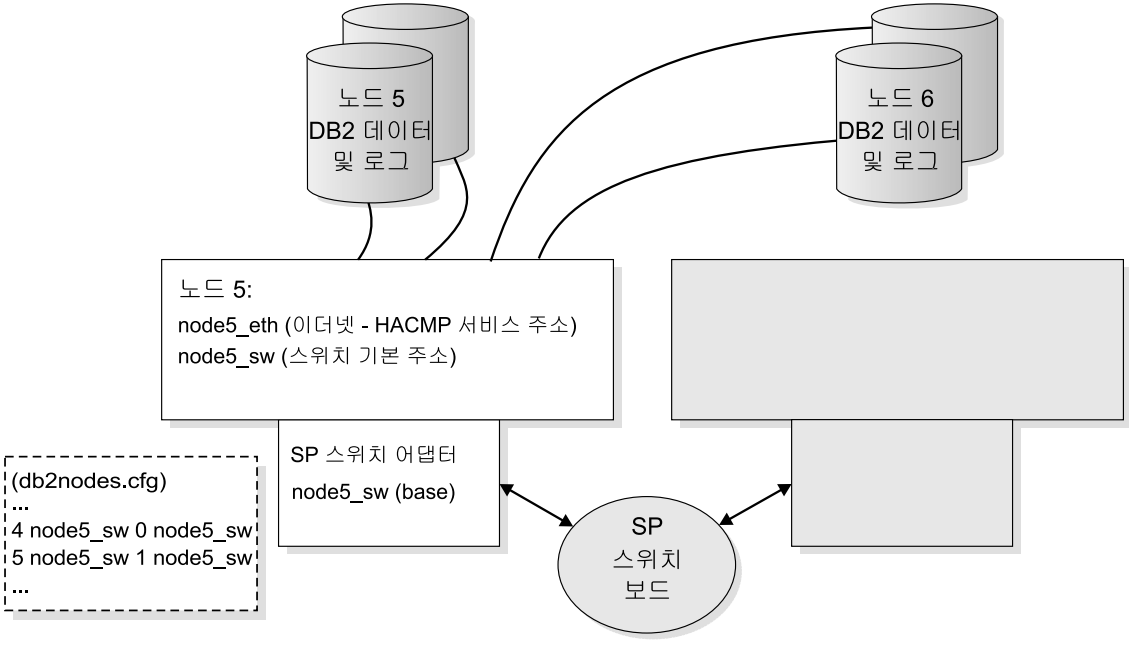


그림 26. NFS 장애 복구가 없는 상호 인계 - DB2 장애 복구

DB2 HACMP 시동시 권장사항

/etc/inittab에서 부트시 HACMP가 시작되지 않도록 지정하는 것이 바람직합니다. HACMP는 노드가 부트된 다음에 수동으로 시작되어야 합니다. 이렇게 하면, 장애를 일으킨 노드를 훼손하지 않고 유지보수할 수 있습니다.

"고장난 노드를 훼손하는 유지보수"의 예로서 노드에서 하드웨어 장애와 고장이 일어난 경우를 가정해 보십시오. 장애 복구는 HACMP에 의해 자동으로 시작되며, 복구가 정상적으로 완료합니다. 그러나 고장난 노드는 고쳐야 합니다. HACMP가 재부트시 시작하는 /etc/inittab로 구성되는 경우, 이 노드는 이 경우에 바람직하지 않은 부트 완료 이후에 재통합하도록 시도합니다.

"고장난 노드를 훼손하지 않는 유지보수"에 대해서는 노드마다 HACMP를 수동으로 시작하는 경우를 가정해 보십시오. 이렇게 하면, 고장난 노드는 다른 노드에 영향을 주지 않고 수리되고 재통합될 수 있습니다. ha_cmd 스크립트는 제어 워크스테이션에서 HACMP 시작 및 중단 명령을 제어하기 위해 제공됩니다.

주: 처음 DB2 인스턴스를 작성할 때, 다음과 같은 항목이 /etc/inittab 파일에 추가됩니다.

```
rcdb2:2:once:/etc/rc.db2 > /dev/console 2>&1 # Autostart DB2 Services
```

HACMP 또는 HACMP ES가 사용 가능하면, HACMP 항목 앞에 위 라인을 위치시켜 /etc/inittab 파일을 갱신하십시오. 다음은 /etc/inittab 파일의 샘플 HACMP 항목입니다.

```
clinit:a:wait:touch /usr/sbin/cluster/.telinit # HACMP for AIX
```

항목은 /etc/inittab 파일의 마지막 항목이어야 합니다.

HACMP ES 이벤트 모니터링 및 사용자 정의 이벤트

페이징 공간이 특정 비율 만큼 찼을 때 AIX 물리 노드에서 DB2 데이터베이스 파티션을 종료하고, DB2 데이터베이스 파티션을 재시작하거나 주어진 노드에서 프로세스가 종료는 경우 장애 복구 조작을 시작하는 것은 사용자 이벤트의 두가지 예입니다. 데이터베이스 파티션을 종료하고 페이징 공간을 해제시키기 위해 강제로 트랜잭션을 중단시키는 것과 같은 사용자 정의 이벤트를 나타내는 예가 samples 서브디렉토리에 있습니다.

HACMP 이벤트가 들어 있는 /user/sbin/cluster/events/rules.hacmprd 규칙 파일이 있습니다. 이 파일의 각 이벤트 설명은 다음 9개의 구성요소를 가집니다.

- 이벤트 이름. 이 이름은 고유해야 합니다.
- 이벤트에 대한 상태 또는 규정자. 이벤트 이름과 상태는 규칙 트리거입니다. HACMP ES 클러스터 관리 프로그램은 이벤트 이름 및 상태에 해당하는 트리거가 있는 규칙을 찾을 때에만 복구를 시작합니다.
- 자원 프로그램 경로. 복구 프로그램이 들어 있는 *xxx*.rp 파일의 전체 경로 스펙입니다.

- 복구 유형. 이는 앞으로 사용하기 위해 보존됩니다.
- 복구 레벨. 이는 앞으로 사용하기 위해 보존됩니다.
- 자원 변수 이름. 이벤트 관리 프로그램 이벤트에 사용됩니다.
- 인스턴스 벡터. 이벤트 관리 프로그램 이벤트에 사용됩니다. 이것은 "name=value"의 요소 집합입니다. 값은 시스템에서 자원의 사본을 고유하게 식별하고 확장자를 보고하여 자원 변수의 사본을 고유하게 식별합니다.
- 술어. 이벤트 관리 프로그램 이벤트에 사용됩니다. 이것은 자원 변수 및 다른 요소 간의 관계형 표현식입니다. 이 표현식이 참이면, 이벤트 관리 서브시스템이 클러스터 관리 프로그램과 해당 응용프로그램에 알리기 위한 이벤트를 생성합니다.
- Rearm 술어. 이벤트 관리 프로그램에 사용됩니다. 이는 기본 술어의 상태를 변경하는 이벤트를 생성하기 위해 사용되는 술어입니다. 이 술어는 보통 기본 술어의 역입니다. 이는 또한 관심있는 조건의 상한선과 하한선을 설정하기 위해 이벤트 술어와 함께 사용됩니다.

각 오브젝트는 이벤트 정의에서 행이 사용되지 않더라도 하나의 행은 있어야 합니다. 만일 이 행들이 제거되면, HACMP ES 클러스터 관리 프로그램은 이벤트 정의를 제대로 분석할 수 없게 되며, 시스템을 정지시킬 수 있습니다. "#"으로 시작되는 모든 행은 주석 행으로 처리됩니다.

주: 규칙 파일은 주석 행을 계산하지 않으며 각 이벤트 정의에 대해 정확하게 9 행이 필요합니다. 규칙 파일 맨 아래에 사용자 정의 이벤트를 추가할 때에는 파일 끝의 불필요한 빈 행을 제거하는 것이 중요합니다. 그렇지 않으면, 노드가 정지합니다.

다음은 node_up에 대한 이벤트 정의의 예입니다.

```
##### Beginning of the Event Definition: node_up
#
TE_JOIN_NODE
0
/usr/sbin/cluster/events/node_up.rp
2
0
# 6) Resource variable - only used for event management events
# 7) Instance vector - only used for event management events
```

```
# 8) Predicate  - only used for event management events
# 9) Rearm predicate - only used for event management events
###### End of the Event Definition: node_up
```

이 예는 rules.hacmprd 파일에서 찾을 수 있는 이벤트 정의 중 하나일 뿐입니다. 이 예에서, node_up 이벤트가 발생하면, /usr/sbin/cluster/events /node_up.rp 복구 프로그램이 실행됩니다. 값은 상태, 복구 유형 및 복구 레벨에 따라 지정됩니다. 자원 변수, 인스턴스 변수, 술어 및 rearm 술어에 대해 4개의 빈 행이 있습니다.

비표준 HACMP ES 이벤트에 반응하도록 다른 이벤트를 정의할 수 있습니다. 예를 들어, /tmp 파일 시스템이 90% 이상 차 있는 이벤트를 정의하려면, rules.hacmprd 파일이 수정되어야 합니다.

많은 이벤트는 IBM 병렬 시스템 지원 프로그램(PSSP)에 사전 정의되어 있습니다. 이러한 이벤트는 사용자 정의 이벤트에서 다음과 같이 사용될 수 있습니다.

1. 클러스터를 중단시키십시오.
2. rules.hacmprd 파일을 편집하십시오. 파일을 수정하기 전에 백업하십시오. 사전 정의된 PSSP 이벤트를 수동으로 추가하십시오. 클러스터의 모든 노드에 걸쳐 동기화 지점이 필요하면, 복구 프로그램에서 **barrier** 명령을 사용하십시오. (HACMP 개념, 설치 및 관리 안내서에 barrier 명령과 복구 프로그램의 동기화에 관한 자세한 설명이 있습니다.)
3. 클러스터를 재시작하십시오. rules.hacmprd 파일은 클러스터 관리 프로그램이 시작될 때 메모리에 저장됩니다. 변경사항을 더 정확하게 구현하기 위해 모든 클러스터를 재시작하십시오. 클러스터에는 어떠한 불일치도 있어서는 안 됩니다.
4. 클러스터 관리 프로그램은 rules.hacmprd 파일의 모든 이벤트를 사용합니다.

HACMP ES는 사용자 이벤트를 처리하기 위해 PSSP 이벤트 감지를 사용합니다. PSSP 이벤트 관리 서브시스템은 다양한 하드웨어 및 소프트웨어 자원을 모니터하여 광범위한 이벤트 감지를 제공합니다.

자원 상태는 자원 변수로 표시됩니다. 자원 조건은 술어라고 하는 표현식으로 나타납니다.

이벤트 관리는 자원 모니터에서 자원 변수를 받으며 특정 시스템 자원의 상태를 관찰하고 이 상태를 여러 자원 변수로 변환합니다. 이들 변수는 정기적으로 이벤트 관리로 전달됩니다. 이벤트 관리는 rules.hacmprd에서 HACMP ES 클러스터 관리 프로그램에 의해 지정된 술어를 각 자원 변수에 적용시킵니다. 술어가 참이면 이벤트가 생성되고 클러스터 관리 프로그램으로 보냅니다. 클러스터 관리 프로그램은 투표 프로토콜을 시작하고 복구 프로그램 파일(xxx.rp)은 복구 프로그램의 노드 세트에 의해 지정된 노드 세트에서 이벤트 우선순위에 따라 수행됩니다.

복구 프로그램 파일(*xxx*.rp)은 하나 이상의 복구 프로그램 행으로 이루어집니다. 각 행은 다음 형식으로 선언됩니다.

```
relationship     command_to_run     expected_status     NULL
```

행의 각 값 사이에 최소한 공백이 하나씩 있어야 합니다. "관계"는 노드의 유형별로 수행되어야 하는 프로그램을 결정하기 위해 사용됩니다. 지원되는 세 가지 유형의 관계는 다음과 같습니다.

- 모두. 지정된 명령 또는 프로그램은 현재 HACMP 클러스터의 모든 노드에서 수행됩니다.
- 이벤트. 지정된 명령 또는 프로그램은 이벤트가 발생한 노드에서만 수행됩니다.
- 기타. 지정된 명령 또는 프로그램은 이벤트가 발생하지 않은 모든 노드에서 수행됩니다.

"Command_to_run"은 실행가능한 프로그램에서 전체 경로 스펙이 있거나 없이, 따옴표로 구분된 문자열입니다. HACMP 전달 이벤트 스크립트만 상대 경로 정의를 사용할 수 있습니다. 다른 스크립트 또는 프로그램에서는 전체 경로 스펙이 사용되어야 합니다(비록 이러한 프로그램이 HACMP 이벤트 스크립트와 같은 디렉토리에 있어도).

"Expected_states"는 지정된 명령 또는 프로그램의 리턴 코드입니다. 정수값이나 "x" 중 하나입니다. "x"가 사용되는 경우에 클러스터 관리 프로그램은 리턴 코드에 대해 신경 쓰지 않습니다. 다른 모든 코드는 예상된 리턴 코드와 동일해야 하며, 그렇지 않으면 클러스터 관리 프로그램이 이벤트 실패를 검출합니다. 이 이벤트의 조절은 수동 개입을 통해 복구가 일어날 때까지 프로세스를 "정지"시킵니다. 노드는 수동 개입 없이 다른 노드와 동기화하지 않습니다. 모든 노드에 걸친 동기화는 클러스터 관리 프로그램이 모든 노드를 제어하기 위한 요구사항입니다.

"널(NULL)"은 앞으로 사용하기 위해 예약된 필드입니다. "널(NULL)"이라는 단어는 barrier line을 제외한 모든 행 끝에 나타나야 합니다. 만일 두 barrier 명령 사이에 또는 첫 번째 명령 전에 여러 복구 명령을 지정하는 경우, 복구 명령은 노드 자체와 노드 사이에서 병렬로 처리됩니다.

barrier 명령은 모든 클러스터 노드에 걸쳐 모든 명령을 동기화하기 위해 사용됩니다. 노드가 복구 프로그램에서 barrier문에 이르면, 클러스터 관리 프로그램이 이 노드에서 프로토콜을 시작합니다. barrier 프로토콜은 2단계 프로토콜이므로, 모든 노드가 복구 프로그램에서 barrier에 다다르고 프로토콜 승인을 선택했다면, 두 단계가 모두 완료되었다는 것을 모든 노드에게 알리게 됩니다.

프로세스는 다음과 같이 요약될 수 있습니다.

1. Group Services/ES(사전 정의된 이벤트) 또는 이벤트 관리(사용자 정의된 이벤트)는 클러스터 관리 프로그램에 이벤트의 발생을 알립니다.
2. 클러스터 관리 프로그램은 `rules.hacmprd` 파일을 읽고 이벤트에 맵된 복구 프로그램을 판별합니다.
3. 클러스터 관리 프로그램은 복구 명령 순서로 이루어진 복구 프로그램을 수행합니다.
4. 복구 프로그램은 쉘 스크립트 또는 2진 명령인 복구 명령을 실행합니다. (AIX용 HACMP에서, 복구 명령은 HACMP 이벤트 스크립트와 같습니다.)
5. 클러스터 관리 프로그램은 복구 명령에서 리턴 상태를 받습니다. 예기치 못한 상태는 수동 개입(`smit cm_rec_aids` 또는 `/usr/sbin/cluster /utilities/clruncmd` 명령을 사용)이 수행될 때까지 클러스터를 "정지"시킵니다.

HACMP ES 스크립트 파일

장애 복구 및 사용자 정의 이벤트에 대해 다음과 같은 샘플 스크립트가 DB2 UDB EEE에 포함됩니다. 스크립트 파일은 `$INSTNAME/sqllib/samples/hacmp/es` 디렉토리에 있습니다. 스크립트는 "있는 그대로" 작동하거나 복구 조치를 사용자 정의할 수 있습니다.

- DB2 데이터베이스 파티션 복구 스크립트 rc.db2pe. 이는 데이터베이스 파티션에서 HACMP 구성을 시작 및 중단하는데 사용되는 스크립트 파일입니다. 이는 또한 DB2 인스턴스 소유자의 NFS 서버에 대해 HACMP 시작 및 중단 스크립트로도 작동합니다.
- HACMP ES에 대한 DB2 고유 사용자 정의 이벤트. 6개의 이벤트가 포함됩니다. 하나는 프로세스 복구용, 두 개는 페이징 공간용, 그리고 세 개는 NFS 및 자동 마운터 복구용입니다.
- DB2 인스턴스 NFS 파일 서버 장애 복구. 이 스크립트는 DB2 인스턴스에 대한 파일 시스템의 서버 장애 복구를 백업에 제공합니다.
- 네트워크 장애 복구. 스크립트 network_up_complete, network_back, network_down_complete 및 network_down에서 SP DB2 데이터베이스 파티션은 SP 스위치 어댑터가 장애를 일으킬 때 장애 복구할 수 있습니다.
- SP GUI Perspectives에 대한 이벤트 모니터링을 정의하기 위한 스크립트. 장애 복구 및 사용자 정의 복구의 모니터 작업은 이벤트 및 하드웨어 Perspectives를 통해 수행될 수 있습니다. Perspectives에 대해 더 알고 싶으면, PSSP 관리 문서를 읽으십시오.
- HACMP ES 노드에서 코어 스크립트 및 이벤트를 설치하고 제거하기 위한 설치 스크립트
- HACMP 및 DB2 구성 모니터 작업을 위해 SP Perspectives 문제점 관리(pman) 자원을 작성하고 제거하기 위한 스크립트 파일

복구 스크립트는 복구 조작을 수행할 각 노드에 설치되어야 합니다. 스크립트 파일은 SP 제어 워크스테이션 또는 기타 지정된 SP 노드로부터 중앙에 설치될 수 있습니다.

1. $INSTNAME/sqllib/samples/hacmp/es 디렉토리에서 스크립트를 SP 제어 워크스테이션이나 **pcp**와 **pexec** 명령을 수행할 수 있는 다른 SP 노드 중 하나로 복사하십시오. 이 명령은 설치 조작에 필요합니다.
2. 장애 복구 구성을 위한 키 매개변수(BUFFPAGE와 같은)를 설정하여 사용자 환경에 대해 reg.parms.SAMPLE 및 failover.parms.SAMPLE 파일을 사용자 정의하십시오. 일반적으로, 상호 인계 구성에 대한 장애 설정값은 일반 설정값 크기의 반 또는 그 이하로 낮춰 조정됩니다. 또한, 사용자의 이름으로 재명명된 이들 파일의 사본을 사용할 수 있습니다("SAMPLE" 대신).

3. 필요하면 5개의 매개변수 NFS_RETRIES, START_RETRIES, MOUNT_NFS, STOP_RETRIES 및 FAILOVER를 rc.db2pe 파일에서 사용자 정의하십시오. 재시도 및 장애 복구 설정은 대부분의 구현에 적합해야 합니다.
   MOUNT_NFS 설정값은 NFS 서버 가용성에 대한 패키지의 사용 여부에 따라 구성되어야 합니다. rc.db2pe가 사용자를 위해 DB2 인스턴스 소유자의 NFS 홈 디렉토리를 마운트하고 검증하고 싶으면 이 설정값을 지정해야 합니다. FAILOVER 매개변수를 "YES"로 설정하면, db2_proc_restart가 실행되고 DB2 데이터베이스 파티션을 재시작하는 시작 프로그램을 호출합니다. 재시작 조작이 성공하지 않으면, HACMP는 장애 복구로 종료됩니다.

4. 이벤트 파일에서 db2_paging_action, db2_proc_recovery 및 nfs_auto_recovery를 사용자에 맞게 정의하십시오. pwq를 편집하여 이를 DB2 인스턴스 소유자로 변경하십시오. 페이징 공간이 90% 넘게 찼다면 db2_paging_action를 사용자 정의하여 어떤 조치를 취할지 지정하십시오. (이러한 상황에서는 DB2 데이터베이스 파티션이 중단됩니다.) 복구 조치가 추가로 필요할 때에는 스크립트를 수정하십시오.

5. 스크립트와 이벤트를 사용자가 지정하는 노드에 설치하려면, db2_inst_ha를 사용하십시오. (HACMP ES는 시작하기 전에 이들 노드에 사전 설치되어야 합니다.) db2_inst_ha 구문은 다음과 같습니다.

```
   db2_inst_ha $INSTNAME/sqllib/samples/hacmp/es <nodelist> <DATABASENAME>
여기서,
      $INSTNAME/sqllib/samples/hacmp/es is the directory in which the
         scripts and the event are located
      <nodelist> is the pcp or pexec style of the nodes; for example,
         1-16 or 1,2,3,4
      <DATABASENAME> is the name of the database for regular and
         failover parameter files.
```

   reg.parms.SAMPLE과 failover.parms.SAMPLE 파일은 각 노드에 복사되고 reg.parms.DATABASENAME으로 재명명됩니다. db2_inst_ha는 /usr/bin에 있는 각 노드로 파일을 복사하며, HACMP 이벤트 파일을 갱신합니다.

```
   /usr/sbin/cluster/events/rules.hacmprd
   /usr/sbin/cluster/events/network_up_complete
   /usr/sbin/cluster/events/network_down_complete
```

6. 시스템과 스크립트를 HACMP로 구성하십시오.

7. 문제점 관리 지원(pman)과 SP GUI Perspectives에 대해 모니터 작업 이벤트를 설치하려면, **create_db2_events** 명령을 사용하십시오. Perspectives에서는

구성 및 사용자 정의가 추가로 필요합니다. Perspectives에 관한 자세한 내용은 PSSP 관리 안내서를 참조하십시오.

8. HACMP ES 장애 복구를 하지 않고 데이터베이스 파티션을 종료하려면 ha_db2stop 명령을 사용하십시오. 이 명령을 사용하려면, 데이터베이스 사용자의 홈 디렉토리에 파일을 복사한 후 해당 사용자에 대해 사용권한 및 소유권이 설정되어 있는지 확인하십시오. 장애 복구를 하지 않고 데이터베이스를 중단하려면, 다음을 입력하십시오.

   ```
   ha_db2stop
   ```

   주: 명령이 리턴되도록 기다려야 합니다. ctrl-C 인터럽트를 사용하거나 프로세스를 강제 종료하여 나가면, 장애 복구를 너무 일찍 재작동하게 할 수도 있고, 일부 데이터베이스 파티션은 중지되지 않을 수도 있습니다.

HACMP ES에 의한 DB2 복구 스크립트 조작

HACMP ES는 다음 방법으로 DB2 복구 스크립트를 호출합니다.

- node_up_local(노드 시작)

  HACMP는 node_up 순서를 수행하면서, 이 노드에 속하거나(연쇄를 통해) 또는 지정된(회전을 통해) 자원 그룹에 지정된 볼륨 그룹, 논리 볼륨, 파일 시스템 및 IP 주소를 얻습니다.

  node_up_local_complete가 수행되면, rc.db2pe가 들어 있는 응용프로그램 서버 정의가 시작되어 이 물리적 노드의 응용프로그램 서버 정의에 지정된 데이터베이스 파티션이 시작됩니다.

  주: rc.db2pe가 시작 모드에서 수행될 때 매개변수(parms) 파일과 일치하는 데이터베이스 디렉토리의 각 DATABASE에 대한 reg.parms.DATABASE에 지정된 DB2 매개변수가 조정됩니다.

  각 노드는 시작할 때 다음 순서에 따릅니다. 다중 HACMP 클러스터가 병렬로 시작하면, 다중 노드가 한꺼번에 올라옵니다.

- node_down_remote(장애 복구)

  HACMP는 지정된 인계 노드상의 자원 그룹에 지정된 볼륨 그룹, 논리 볼륨, 파일 시스템 및 IP 주소를 얻습니다.

node_down_remote_complete가 수행되면, HACMP는 rc.db2pe를 이 데이터베이스 파티션에 대한 자원 그룹에 지정된 응용프로그램 서버 시작 스크립트로 수행합니다.

주: rc.db2pe가 상호 인계에서 수행될 때, 여기서 수행되는 DB2 데이터베이스 파티션이 중단되고, 매개변수(parms) 파일과 일치하는 데이터베이스 디렉토리의 각 DATABASE에 대한 failover.parms.DATABASE에 지정된 DB2 매개변수가 조정되며, 물리적 인계 노드의 데이터베이스 파티션이 모두 시작됩니다.

- node_up_remote(장애를 일으킨 노드의 재통합 - 연쇄 상호 인계 자원 그룹)

  node_up_remote가 이전 인계 노드에서 수행되면, 응용프로그램 서버 정의로서 rc.db2pe는 중단 모드에서 수행됩니다.

  주: rc.db2pe가 재통합 모드(상호 인계)에서 수행되면, 여기서 수행되는 양 데이터베이스 파티션이 모두 중단되고, 매개변수(parms) 파일과 일치하는 데이터베이스 디렉토리의 각 DATABASE에 대한 reg.parms.DATABASE에 지정된 DB2 매개변수가 조정되며, 이 물리적 인계 노드에 보관될 데이터베이스 파티션이 시작됩니다.

  이전 인계 노드는 재통합 노드에 속하는 자원 그룹에 지정된 볼륨 그룹, 논리 볼륨, 파일 시스템 및 IP 주소를 해제합니다.

  HACMP는 현재 재통합되는 노드에 속하는 자원 그룹에 지정된 볼륨 그룹, 논리 볼륨, 파일 시스템 및 IP 주소를 다시 얻습니다.

  node_up_local_complete가 수행되면, rc.db2pe가 들어 있는 응용프로그램 서버 정의가 시작되어 이 물리적 노드의 응용프로그램 서버 정의에 지정된 DB2 데이터베이스 파티션이 시작됩니다.

  주: rc.db2pe가 시작 모드에서 수행될 때 매개변수(parms) 파일과 일치하는 데이터베이스 디렉토리의 각 DATABASE에 대한 reg.parms.DATABASE에 지정된 DB2 매개변수가 조정됩니다.

- node_down_local(노드 중단 또는 인계를 갖춘 중단)

node_down_local이 중단 노드에서 수행되면, 응용프로그램 서버 정의로 rc.db2pe는 중단 모드에서 수행됩니다.

주: rc.db2pe가 중단 모드에서 수행되면, 매개변수(parms) 파일과 일치하는 데이터베이스 디렉토리의 각 DATABASE에 대한 failover.parms.DATABASE에 지정된 DB2 매개변수가 조정되고 DB2 데이터베이스 파티션(인계용)이 중단됩니다.

HACMP는 현재 노드에 속하는 자원 그룹에 지정된 볼륨 그룹, 논리 볼륨, 파일 시스템 및 IP 주소를 해제합니다.

- db2_proc_recovery(db2 프로세스 중단)

  모든 노드는 db2_proc_restart 스크립트를 수행합니다. 장애가 일어난 노드는 올바른 DB2 데이터베이스 파티션을 재시작합니다.

- db2_paging_recovery(페이징 공간 복구)

  모든 노드는 db2_paging_action 스크립트를 수행합니다. 만일 노드의 페이징 공간이 70% 넘게 찼다면, 월(wall) 명령이 생성됩니다. 만일 노드의 페이징 공간이 90% 넘게 찼다면, 이 물리 노드의 DB2 데이터베이스 파티션이 중단되고 재시작됩니다.

- nfs_auto_recovery(nfs 또는 자동 마운트 프로세스 장애)

  모든 노드는 rc.db2pe 스크립트를 NFS 모드에서 수행합니다. 만일 NFS 프로세스의 수행이 중단되면, 재시작됩니다. 마찬가지로, 자동 마운트 프로세스의 수행이 중단되면, 재시작됩니다.

- network_down_complete(네트워크 장애 - SP 스위치)

  net_down 스크립트가 호출됩니다. 이는 네트워크를 SP 스위치 네트워크로 검증하고 또한 이것이 중단되었음을 검증합니다. 만일 이러한 경우에, 이는 사용자 정의 시간 간격 동안 대기합니다. 기본 시간 간격은 100초입니다.

  SP 스위치 네트워크가 network_up_complete 이벤트에 의한 것처럼 되돌아오면, 어떠한 복구도 영향을 받지 않습니다.

  시간 한계에 도달하면, HACMP는 장애 복구와 함께 중단됩니다.

주: 모든 이벤트는 SP 문제점 관리 및 SP Perspectives GUI로 모니터될 수 있습니다.

기타 스크립트 유틸리티

다른 스크립트 유틸리티는 다음을 포함하여 사용자가 사용할 수 있습니다.

- ha_cmd. 제어 워크스테이션의 SP 노드에서 HACMP를 시작하기 위해 제공된 명령입니다. 구문은 다음과 같습니다.

```
   ha_cmd <noderange> <START|STOP|TAKE|FORCE>
여기서,
     <noderange>는 SP 노드 범위의 PCP 또는 Pexec 스타일입니다.
        예를 들어, "ha_cmd 3-6 START"은 노드 3,4,5,6에서 HACMP를 시작합니다.
                   "ha_cmd 5 TAKE"은 상호 인계를 위해 노드 5에서 HACMP를 종료합니다.
```

- ha_mon, SP 제어 워크스테이션에서 HACMP hacmp_out 파일을 모니터하기 위한 명령입니다. 구문은 다음과 같습니다.

```
   ha_mon <node>
여기서,
     <node>는 모니터되는 SP 노드입니다.
     ha_mon will "tail -f" the /tmp/hacmp.out file on the node you specify.
```

- db2_turnoff_recov, 일시적으로 모든 HACMP(비 장애 복구) 복구를 사용하지 않으며, 극히 드문 상황을 위해 디자인된 명령. 어떠한 DB2 프로세스, 페이징, NFS 또는 자동 마운터 복구도 시작되지 않습니다. 이 기능은 HACMP 규칙 파일에서 해당 복구에 대한 이벤트 스탠자를 제거합니다. HACMP는 중단된 후 시작되어야 합니다. 구문은 다음과 같습니다.

```
db2_turnoff_recov <nodelist>
```

- db2_turnon_recov, HACMP(장애 복구를 제외한) 복구를 재작동시키기 위한 명령입니다. 이 명령은 사용자 정의 이벤트 복구가 발생할 수 있도록 HACMP 규칙 파일을 복원하기 위해 db2_turnoff_recov 이후에 사용됩니다. HACMP는 중단된 후 시작되어야 합니다. 구문은 다음과 같습니다.

```
db2_turnon_recov <nodelist>
```

HACMP 클러스터 모니터링

스크립트는 이미 HACMP ES에 있는 모니터 작업 유틸리티 외에 DB2 HACMP ES 구성을 모니터하기 위한 SP 문제점 관리(pman) 이벤트를 작성하기 위해 제공됩니다. HACMP 상태를 SP 제어 워크스테이션에서 모니터하려면 다음과 같이 하십시오.

- 제어 워크스테이션에서 HACMP 클라이언트 코드를 설치하십시오.

- /usr/sbin/cluster/etc/clhosts 파일을 편집한 후 모니터할 노드의 SP 이더넷 IP 주소를 포함시키십시오.
- 클러스터 모니터를 시작하기 위해 startsrc -s clinfo 명령을 호출하십시오.

HACMP는 클러스터(/usr/sbin/cluster/clstat) 모니터에 대한 인터페이스를 제공합니다.

HACMP RS 및 사용자 정의 이벤트에 대해 SP Perspectives와 함께 문제점 관리 모니터 작업을 사용하려면, 다음을 수행하십시오.

1. create_db2_events <nodelist>를 호출하십시오. 여기서, *nodelist*에는 pcp 또는 pexec 스타일 노드가 들어 있습니다. 이 스크립트는 Perspectives에 의한 모니터 작업에 대해 5개의 pman 이벤트를 작성합니다.

   주: PSSP.pm.User_state12-16 자원 변수는 이러한 이벤트 작성시 사용됩니다. 이러한 자원 변수가 이미 몇 가지 다른 목적을 위해 사용되었다면, create_db2_events 및 update_db2_events는 다른 자원 변수를 사용하도록 갱신되어야 합니다.

2. 제어 워크스테이션에서 Perspectives를 시작하십시오. 런치 패드에서 이벤트 perspective를 선택하십시오. 5개의 이벤트인 db2_hacmp_recovery, db2_process_recovery, db2_paging_err, db2_nfs_err, Errlog_PERM_entry를 보아야 합니다.
3. 각 이벤트를 두 번 클릭하십시오. 나타나는 화면에서 이벤트에 대한 조건을 등록합니다(정의 테이블에). 아래 화살표 옆의 Name: "unnamed"를 클릭한 후 조건으로 지정한 이벤트와 같은 이름을 선택하십시오. "Response Options" 탭을 선택하십시오. 표시장치 맨 위에 있는 버튼을 클릭하십시오(″Perspectives 이벤트 세션으로 메시지 송신″). 명령, errlog 항목뿐만 아니라 SNMP 트랩도 지정할 수 있습니다. 이벤트 로그 표시장치는 Perspective 세션에서만 유지보수됩니다. 그러므로 각각에 대해 AIX 오류 로그 항목을 작성할 수 있습니다. **OK**를 선택하고 창을 닫으십시오.
4. Perspectives 런치 패드에서, 하드웨어 Perspective를 선택하십시오.
5. 하드웨어 프레임 GUI가 나타나면, ″뷰″를 선택한 다음 ″모니터″를 선택하십시오. SP에 대해 모니터될 수 있는 이벤트 목록이 제공됩니다. 목록 맨 아래로 화면이동하면, 두 개의 추가 이벤트를 볼 수 있습니다. 하나는 HACMP DB2

복구(db2_ha_ind)에 대한 것이고, 다른 하나는 SP 노드 PERM 오류(Errlog_PERM_mon)에 대한 것입니다. 모니터할 이벤트를 선택하십시오. (이벤트가 나타나면, 노드에 대한 표시장치는 빨간색 "X"입니다. 모니터된 모든 조건이 좋으면, 노드에 대한 표시장치는 초록색이 됩니다.) 일반적으로 host_responds, switch_responds 및 node_power_LED가 사용됩니다. 또한, 노드에서 DB2 HACMP 복구뿐만 아니라, PERM 오류도 모니터할 수 있습니다.

주: pman과 Perspectives에 대한 db2_hacmp_mon 및 db2_hacmp_recovery 변수는 HACMP 클러스터상태를 반영하지 않습니다. 그 보다, 이들 변수는 DB2를 시작하거나 정지시키기 위한 rc.db2pe 조작의 상태를 반영합니다. "실제" HACMP 상태는 HACMP clstat 모니터에 표시되고 HACMP 클러스터 상태를 반영합니다. db2_hacmp_ind가 HACMP 상태와 비슷한 모니터 작업을 반영하도록 만들려면, 다음 행을 /etc/inittab 파일에 추가하십시오.

```
haind:2:wait:/usr/bin/db2_update_events HAIND OFF 2>&1 >/dev/null
```

구현시 NetView를 사용할 계획이라면, 구성 모니터시 HAVIEW(HACMP의 일부) 사용을 고려해 보십시오. NetView 문서에서 제품 구성에 대한 자세한 내용을 참조하십시오.

DB2 SP HACMP ES 설치

DB2 Universal Database를 사용하여 HACMP ES 설치 계획을 도우려면, 다음에서 설치 및 이주 프로세스의 단계별 개요를 참조하십시오.

DB2 SP HACMP ES 새로 설치

HACMP ES를 설치하려면 다음과 같이 하십시오.

1. 각 SP 노드에서 AIX 운영 체제를 설치하십시오(SP 설치 및 관리 안내서 참조). 제어 워크스테이션과 각 SP 노드 양쪽에 사용할 수 있는 적합한 페이징 공간이 있는지 확인하십시오. 스위치 구성이 고려되고 다른 모든 수정 가능한 구성 매개변수와 함께 구현되어 있는지 확인하십시오. 사용하려는 SP 모니터링(Perspectives)을 위치시키십시오. SP dsh, pcp 및 pexec 명령이 작동하는지 확인하십시오.

2. 데이터베이스 배치를 설계하십시오. 이는 최소한 사용된 노드의 수, 물리적 노드에 대한 DB2 데이터베이스 파티션의 맵핑, 노드 또는 파티션당 디스크 요구량, 그리고 테이블 공간 고려사항을 포함해야 합니다. 누가 주 DB2 인스턴스 소유자가 될지와 이 사용자와 다른 사용자에게 필요한 액세스 권한 부여에 대해서도 고려해야 합니다.
3. 여분의 어댑터, 이중복사된 디스크 및 디스크의 트윈 테일링(twin-tailing)을 포함한 외부 SSA 구성을 계획하십시오.
4. 데이터베이스 배치와 SSA 구성을 사용하여 HACMP 계획, 설치 및 관리 안내서에 있는 HACMP 워크쉬트를 완성하십시오.
5. 외부 SSA 디스크 구성을 구현하십시오. 모든 드라이브에서 마이크로코드 레벨이 일정한지 확인하고 워크쉬트에 빈 곳이 있을 때에는 Maymap 유틸리티를 사용하여 유효성을 검사하고 채우십시오.
6. 각 SP 노드에 DB2 UDB EEE를 설치하십시오.
7. 각 SP 노드에 HACMP ES를 설치하십시오.
8. **db2_inst_ha** 명령을 사용하여 SP 패키지에 DB2 UDB EEE HACMP ES를 설치하십시오.
9. DB2 주 인스턴스 사용자를 작성하고 이것으로 모든 노드에 액세스할 수 있는지 확인하십시오. 이는 이 시점에서 주로 사용할 수 있는 사용자가 아닙니다. 이는 SP 제어 워크스테이션에서 일시적으로 SP 사용자가 될 수 있습니다.
10. DB2 인스턴스와 데이터베이스를 작성하십시오. 다음 단계로 진행하기 전에 **db2start**를 호출한 후 **db2stop**를 호출하여 조작 가능한지 확인하십시오.
11. HACMP를 추가하기 전에 데이터베이스를 로드하려면, 다음 단계를 수행하십시오.
12. HACMP 워크쉬트와 이 책의 정보에 따라 SP 노드 토폴로지 및 자원 그룹에 HACMP ES를 구성하십시오.
13. 주 DB2 인스턴스 사용자에 대해 NFS 서버 노드로 시작하면서, 이 책에 지정된 내용에 따라 모든 노드에서 이 사용자를 변경하십시오(`/etc/security/user`와 `/etc/passwd`를 수정하여). 이 사용자는 자주 사용할 수 있는 NFS 사용자가 되며, 이 노드 및 백업은 `/etc/exports`를 갱신합니다. 모든 노드

는 스위치 별명 IP 주소를 통해 NFS(각 노드의 /etc/filesystems에 항목이 있는)를 사용하여 이 디렉토리를 마운트할 수 있습니다.

14. 주 인스턴스 사용자의 홈 디렉토리를 "Tar"하고 홈 디렉토리를 새로운 위치에서 "un-tar"하십시오.
15. 새로운 주 인스턴스 홈 디렉토리를 마운트하기 위해 SP 노드마다 NFS 파일 시스템을 작성하십시오.
16. NFS 서버 노드에서 HACMP를 시작하십시오. /tmp/hacmp.out를 조사하여 제대로 시작되는지 검증하십시오. **ha_mon** 명령은 이 파일이 기록될 때 모니터하기 위해 사용될 수 있습니다.
17. 한 번에 다른 노드 하나씩 가져오십시오. /tmp/hacmp.out를 조사하여 각각 제대로 완료되었는지 검증하십시오. **ha_mon** 명령은 이 파일이 기록될 때 모니터하기 위해 사용될 수 있습니다.
18. Perspectives 및 문제점 관리를 통해 선택적 모니터 작업을 설정하십시오.
19. 각 노드에서 동시적인 유지보수 조치를 시뮬레이트하여 각 노드의 장애 복구 기능성에 대해 유효성 검사를 실시하십시오. **ha_cmd** 명령(TAKE 옵션 지정)을 사용하여 HACMP를 인계와 함께 자연스럽게 중단시킬 수 있습니다. /tmp/hacmp.out을 질문하고 모니터링 도구를 사용하여 인계 및 재통합이 정상적인지 검증하십시오.

DB2 SP HACMP ES 이주

HACMP가 없는 설치에서 HACMP가 있는 설치로 이주할 때에는 다음의 개요를 고려해야 합니다.

1. 기존의 외부 디스크를 자주 사용할 수 있는 트윈 테일, 이중복사 구성으로 변환하십시오. 서로 다른 노드에 있는 논리 볼륨은 트윈 테일될 때, 해당 이름이 *반드시* 고유해야 된다는 것을 염두에 두고 이 구성을 이루기 위해 여분의 하드웨어 및 디스크를 추가하십시오. 이는 볼륨 그룹, 논리 볼륨 및 파일 시스템에 적용됩니다.
2. 이 문서에 있는 워크쉬트를 포함하여, HACMP 계획 및 관련된 워크쉬트를 완료하십시오.

3. 외부 SSA 디스크 구성 변경사항을 구현하십시오. 모든 드라이브에서 마이크로코드 레벨이 일정한지 확인하고 워크쉬트에 빈 곳이 있을 때에는 Maymap 유틸리티를 사용하여 유효성을 검사하고 제거하십시오.

   주: RAID5의 SSA 디스크 구성이 지원됩니다. 동일한 RAID 루프에 두 개의 SSA 어댑터를 구성하는 것만 허용됩니다. RAID 디스크가 트윈 테일된 HACMP 구성의 경우, 노드당 한개의 어댑터만 지원됩니다. 이러한 구성의 경우, 어댑터는 디스크에 액세스할 때 단일 실패 지점이 되고, 어댑터 정지를 감지하여 이를 HACMP 장애 복구 이벤트로 보내기 위한 여분의 구성이 권장됩니다. AIX 오류 통지는 SSA 어댑터가 고장났을 때 장애 복구를 위한 노드를 구성하기 위한 가장 간단한 방법입니다. AIX 오류 통지에 대해서는 *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*에서 자세한 내용을 참조하십시오.

4. 각 SP 노드에 HACMP ES를 설치하십시오.
5. **db2_inst_ha** 명령을 사용하여 SP 패키지에 DB2 UDB EEE HACMP ES를 설치하십시오.
6. HACMP 워크쉬트와 이 책의 정보에 따라 SP 노드 토폴로지 및 자원 그룹에 HACMP ES를 구성하십시오.
7. 주 DB2 인스턴스 사용자에 대해 NFS 서버 노드로 시작하면서, 이 책에 지정된 내용에 따라 모든 노드에서 이 사용자를 변경하십시오(`/etc/security/user`와 `/etc/passwd`를 수정하여). 이 사용자는 자주 사용할 수 있는 NFS 사용자가 되며, 이 노드 및 백업은 `/etc/exports`를 갱신합니다. 모든 노드는 스위치 별명 IP 주소를 통해 NFS(각 노드의 `/etc/filesystems`에 항목이 있는)를 사용하여 이 디렉토리를 마운트할 수 있습니다.
8. 주 인스턴스 사용자의 홈 디렉토리를 ″Tar″하고 홈 디렉토리를 새로운 위치에서 ″un-tar″하십시오.
9. 새로운 주 인스턴스 홈 디렉토리를 마운트하기 위해 SP 노드마다 NFS 파일 시스템을 작성하십시오.
10. NFS 서버 노드에서 HACMP를 시작하십시오. `/tmp/hacmp.out`를 조사하여 제대로 시작되는지 검증하십시오. **ha_mon** 명령은 이 파일이 기록될 때 모니터하기 위해 사용될 수 있습니다.

11. 한 번에 다른 노드 하나씩 가져오십시오. /tmp/hacmp.out를 조사하여 각각 제대로 완료되었는지 검증하십시오. **ha_mon** 명령은 이 파일이 기록될 때 모니터하기 위해 사용될 수 있습니다.
12. Perspectives 및 문제점 관리를 통해 선택적 모니터 작업을 설정하십시오.
13. 각 노드에서 동시적인 유지보수 조치를 시뮬레이트하여 각 노드의 장애 복구 기능성에 대해 유효성 검사를 실시하십시오. **ha_cmd** 명령(TAKE 옵션 지정)을 사용하여 HACMP를 인계와 함께 자연스럽게 중단시킬 수 있습니다. /tmp/hacmp.out을 질문하고 모니터링 도구를 사용하여 인계 및 재통합이 정상적인지 검증하십시오.

DB2 SP HACMP ES 워크쉬트

다음 워크쉬트는 외부 SSA 디스크 구성을 준비하기 위해 완료되어야 하며 HACMP 계획, 설치 및 관리 안내서에 있는 HACMP 워크쉬트와 함께 사용하도록 디자인되어 있습니다. 각 경우에, 완료된 예 및 비어있는 워크쉬트 둘다가 제공됩니다.

첫 번째 샘플 워크쉬트에 문서화된 외부 디스크에서의 데이터베이스 구성은 다음 그림에 있습니다. 데이터베이스를 작성하기 위해 사용된 명령문은 다음과 같습니다.

```
db2 create database pwq on /newdata
```

SSA 외부 어댑터와 외부 SSA 디스크는 둘다 장애를 일으킨 적이 없는 논리 볼륨에 대해 이중복사되고 트윈 테일됩니다. 그림에서는 **maymap** 명령의 출력과 유사한 구성을 나타냅니다. Maymap은 외부 SSA 구성을 표시하는 유틸리티(AIXTOOLS를 통해 사용 가능)이며, 사용자 설정을 계획할 때 사용되어야 합니다.

샘플 DB2 4-노드 데이터베이스 외부 디스크 설정

- 고가용성을 위해 두 개의 노드 연결

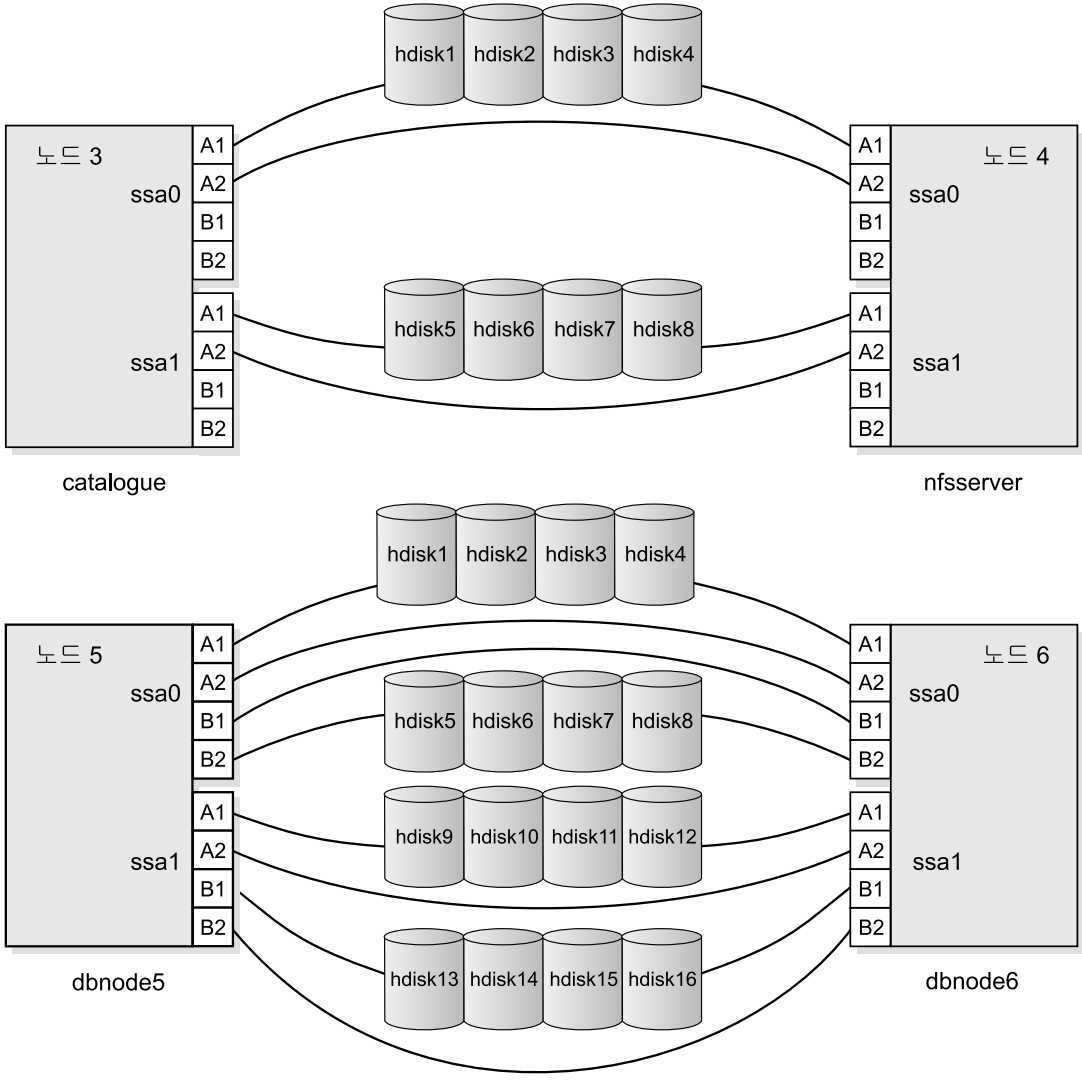


그림 27. 샘플 DB2 4 노드 데이터베이스 외부 디스크 설정

다음 테이블을 검토하기 전에 볼륨 그룹에서의 quorum 설정값과 논리 볼륨에서의 이중복사된 기록 일관성 설정값에 관한 HACMP 문서를 읽어야 합니다. 양쪽에 사용된 설정값

은 가용성과 성능에 직접적으로 영향을 미칩니다. 이들 설정값을 검토하고 해당 의미를 이해해야 합니다. ″quorum″ 및 ″이중복사된 쓰기 일관성″에 대한 일반적인 설정은 ″off″입니다.

표 12. HACMP 볼륨 그룹, 논리 볼륨 및 파일 시스템

SP 노드	볼륨 그룹 이름	PP 크기 (MB)	논리 볼륨 이름	PP의 #	사본	hdisk 목록	파일 시스템 마운트 지점	파일 시스템 로그 논리 볼륨	노드 설명 및 백업	/dev 논리 장치의 사용자 소유자
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	Catalognode 마운트 지점, 노드 4	루트 *
3	havg3	8	hlog301	1	2	hdisk1 hdisk5	N/A	N/A	Catalognode jfslog; 노드 4	루트 *
3	havg3	8	hlv301	10	2	hdisk2 hdisk6	N/A	N/A	Catalognode rawtemp 공간, 노드 4	pwq **
4	havg4	8	hlv400	10	2	hdisk3 hdisk7	/dbmnt	hlog401	nfsserver pwq 홈, 노드 3	루트 *
4	havg4	8	hlog401	1	2	hdisk3 hdisk7	N/A	N/A	nfsserver jfslog; 노드 3	루트 *
5	havg5	8	hlv500	10	2	hdisk1 hdisk9	/newdata/ pwq/ NODE0005	HLOG501	Dbnode5 마운트 지점, 노드 6	루트 *
5	havg5	8	hlog501	1	2	hdisk1 hdisk9	N/A	N/A	Dbnode5 jfslog; 노드 6	루트 *
5	havg5	8	hlv501	10	2	hdisk2 hdisk10	N/A	N/A	Dbnode5 raw temp 공간, 노드 6	pwq **
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	N/A	N/A	Dbnode5 원시 테이블 공간, 노드 6	pwq **
5	havg5	8	halv503	100	2	hdisk3 hdisk11	N/A	N/A	Dbnode5 원시 테이블 공간, 노드 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	N/A	N/A	Dbnode5 원시 테이블 공간, 노드 6	pwq **

표 12. HACMP 볼륨 그룹, 논리 볼륨 및 파일 시스템 (계속)

SP 노드	볼륨 그룹 이름	PP 크기 (MB)	논리 볼륨 이름	PP의 #	사본	hdisk 목록	파일 시스템 마운트 지점	파일 시스템 로그 논리 볼륨	노드 설명 및 백업	/dev 논리 장치의 사용자 소유자
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Dbnode6 시스템 테이블 공간, 노드 6	루트 *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Dbnode6 마운트 지점, 노드 5	루트 *
6	havg6	8	hlog601	1	2	hdisk5 hdisk13	N/A	N/A	Dbnode6 jfslog; 노드 5	루트 *
6	havg6	8	hlv601	10	2	hdisk6 hdisk14	N/A	N/A	Dbnode6 원시 임시 공간, 노드 5	pwq **
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	N/A	N/A	Dbnode6 원시 테이블 공간, 노드 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	N/A	N/A	Dbnode6 원시 테이블 공간, 노드 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	N/A	N/A	Dbnode6 원시 테이블 공간, 노드 5	pwq **
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Dbnode6 시스템 테이블 공간, 노드 5	루트 *

주:

1. * jfs 파일 시스템 논리 볼륨 및 로그는 루트 승인을 유지합니다.
2. ** 원시 데이터베이스 공간은 /dev 원시 파일 항목에서 데이터베이스 사용자 사용권한을 얻습니다(/dev/rxxxx).

표 13. HACMP 볼륨 그룹, 논리 볼륨 및 파일 시스템 - 공백

SP 노드	볼륨 그룹 이름	PP 크기 (MB)	논리 볼륨 이름	PP의 #	사본	hdisk 목록	파일 시스템 마운트 지점	파일 시스템 로그 논리 볼륨	노드 설명 및 백업	/dev 논리 장치의 사용자 소유자

표 13. HACMP 볼륨 그룹, 논리 볼륨 및 파일 시스템 - 공백 (계속)

SP 노드	볼륨 그룹 이름	PP 크기 (MB)	논리 볼륨 이름	PP의 #	사본	hdisk 목록	파일 시스템 마운트 지점	파일 시스템 로그 논리 볼륨	노드 설명 및 백업	/dev 논리 장치의 사용자 소유자

표 13. HACMP 볼륨 그룹, 논리 볼륨 및 파일 시스템 - 공백 (계속)

SP 노드	볼륨 그룹 이름	PP 크기 (MB)	논리 볼륨 이름	PP의 #	사본	hdisk 목록	파일 시스템 마운트 지점	파일 시스템 로그 논리 볼륨	노드 설명 및 백업	/dev 논리 장치의 사용자 소유자

표 13. HACMP 볼륨 그룹, 논리 볼륨 및 파일 시스템 - 공백 (계속)

SP 노드	볼륨 그룹 이름	PP 크기 (MB)	논리 볼륨 이름	PP의 #	사본	hdisk 목록	파일 시스템 마운트 지점	파일 시스템 로그 논리 볼륨	노드 설명 및 백업	/dev 논리 장치의 사용자 소유자

표 14. HACMP NFS 서버 계획

SP 노드	외부 파일 시스템	백업 노드	SP 스위치 부트 및 서비스 IP 별명 쌍	마운트할 파일 시스템 (/etc/filesystems)	데이터베이스 홈 디렉토리로 지정할 파일 시스템	파일 시스템이 내보내기 할 주소 (/etc/exports)
3	/dbmnt	4	nfs_boot_3 nfs_client_3	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
4	/dbmnt	3	nfs_server_boot nfs_server	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
5	N/A	N/A	nfs_boot_5 nfs_client_5	nfs_server:/ dbmnt as /dbi	/dbi/pwq	N/A
6	N/A	N/A	nfs_boot_6 nfs_client_6	nfs_server:/ dbmnt as /dbi	/dbi/pwq	N/A

표 14. HACMP NFS 서버 계획 (계속)

SP 노드	외부 파일 시스템	백업 노드	SP 스위치 부트 및 서비스 IP 별명 쌍	마운트할 파일 시스템 (/etc/filesystems)	데이터베이스 홈 디렉토리로 지정할 파일 시스템	파일 시스템이 내보내기 할 주소 (/etc/exports)
주: 1. /etc/passwd는 모든 노드에서 같아야 합니다. 이는 제어 워크스테이션에서 동기화될 수 있습니다. 2. 외부 파일 시스템에 데이터베이스 인스턴스 소유자의 사용권한이 있는지 확인하십시오. 3. /etc/filesystems에는 다음과 같은 마운트 매개변수가 있어야 합니다. hard, bg, intr 및 rw. 4. /etc/exports는 서버 및 백업에만 다음을 가집니다. -root=ip1:ip2:ip3						

표 15. HACMP NFS 서버 계획 - 공백

SP 노드	외부 파일 시스템	백업 노드	SP 스위치 부트 및 서비스 IP 별명 쌍	마운트할 파일 시스템(/etc/filesystems)	데이터베이스 홈 디렉토리로 지정할 파일 시스템	파일 시스템이 내보내기 할 주소 (/etc/exports)

표 15. HACMP NFS 서버 계획 - 공백 (계속)

SP 노드	외부 파일 시스템	백업 노드	SP 스위치 부트 및 서비스 IP 별명 쌍	마운트할 파일 시스템(/etc/filesystems)	데이터베이스 홈 디렉토리로 지정할 파일 시스템	파일 시스템이 내보내기 할 주소 (/etc/exports)

제7장 Windows 운영 환경의 고가용성

머신이 실패할 경우, 실패한 머신의 데이터베이스 서버가 다른 머신에서 수행될 수 있도록 데이터베이스 시스템을 설정할 수 있습니다. Windows NT에서, 파일복구 지원은 MSCS(Microsoft Cluster Server)로 구현될 수 있습니다. MSCS를 사용하려면, MSCS 기능이 있는 Windows NT 버전 4.0 Enterprise Edition이 설치되어 있어야 합니다.

MSCS는 클러스터된 환경에서 물리적 디스크 및 IP 주소에 대한 장애 복구 지원과 같은, 실패 검출과 자원 재시작 둘다를 수행할 수 있습니다. (실패한 머신이 다시 온라인 상태에 있을 때, 이전에 자원이 자동으로 실패한 머신으로 후진하도록 구성하지 않는 한 자원은 자동으로 후진되지 않습니다. 271 페이지의 『후진 고려사항』에서 자세한 내용을 참조하십시오.)

장애 복구 지원에 대해 DB2 인스턴스를 작동 가능하게 하기 전에, 다음의 계획 단계를 수행하십시오.

1. 데이터 저장영역에 사용할 디스크를 결정하십시오. 각 데이터베이스 서버에는 자체 사용을 위해 최소한 하나의 디스크가 지정되어야 합니다. 데이터 저장에 사용할 디스크는 공유 디스크 서브시스템에 접속되어야 하고 MSCS 디스크 자원으로 구성되어야 합니다.
2. 원격 요청을 지원하기 위해 사용할 하나의 IP 주소가 각 데이터베이스 서버에 대해 있는지 확인하십시오.

장애 복구 지원을 설정할 경우, 기존 인스턴스에 대해 설정하거나 장애 복구 지원을 구현할 때 새 인스턴스를 작성할 수 있습니다.

장애 복구 지원이 작동되도록 하려면, 다음 단계를 수행하십시오.

1. DB2MSCS 유틸리티에 대해 입력 파일을 작성하십시오.
2. **db2mscs** 명령을 호출하십시오.

3. 파티션된 데이터베이스 시스템을 사용 중인 경우, 상호 인계가 가능하도록 데이터베이스 드라이브 맵핑을 등록하십시오. 271 페이지의 『파티션된 데이터베이스 환경에서의 상호 인계 구성에 대한 데이터베이스 드라이브 맵핑 등록』에서 자세한 내용을 참조하십시오.

인스턴스의 장애 복구 지원을 작동 가능하게 하고 나면, 구성은 그림28과 같게 됩니다.

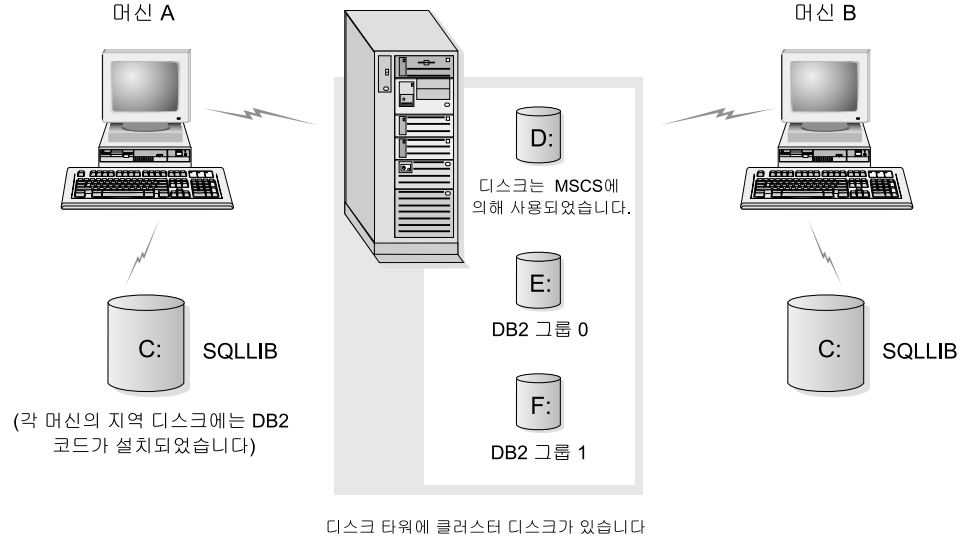


그림 28. MSCS 구성 예

다음 절에서는 서로 다른 유형의 장애 복구 지원 및 유형을 구현하는 방법에 대해 설명합니다. 다음에 설명된 단계를 수행하기 전에, MSCS 클러스터에서 사용할 MSCS 소프트웨어가 모든 머신에 설치되어 있어야 합니다. 또한, 모든 머신에 DB2도 설치해야 합니다.

장애 복구 구성

두 유형의 구성이 사용 가능합니다.

- 긴급 대기
- 상호 인계

현재, MSCS는 두 머신에 대한 클러스터를 지원합니다.

파티션된 데이터베이스 환경에서, 클러스터의 모두 구성 유형이 같을 필요는 없습니다. 일부 클러스터는 긴급 대기를 사용하도록 설정하고, 다른 클러스터는 상호 인계할 수 있도록 설정할 수 있습니다. 예를 들어, DB2 인스턴스가 5개의 워크스테이션으로 구성될 경우, 두 머신은 상호 인계 구성을 사용하도록 설정하고 두 개는 긴급 대기 구성을 사용하도록 또 하나는 장애 복구 지원에 대해 구성하지 않을 수 있습니다.

긴급 대기 구성

긴급 대기 구성에서, MSCS 클러스터의 하나의 머신은 전용 장애 복구 지원을 제공하고, 다른 머신은 데이터베이스 시스템에 관여합니다. 데이터베이스 시스템에 관여하는 머신이 실패할 경우, 그 머신의 데이터베이스 서버는 장애 복구 머신에서 시작됩니다. 파티션된 데이터베이스 시스템에서, 한 머신에서 다중 논리 노드를 수행하는데 그 머신이 실패할 경우, 논리 노드는 장애 복구 머신에서 시작됩니다. 그림29에서는 긴급 대기 구성의 예를 보여줍니다.

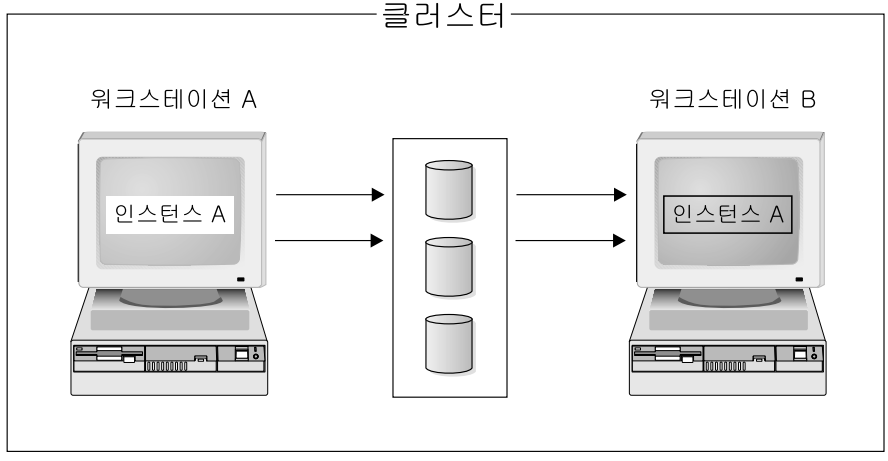


그림 29. 긴급 대기 구성

상호 인계 구성

상호 인계 구성에서, 두 워크스테이션은 모두 데이터베이스 시스템에 관여합니다. (즉, 각 머신에는 최소한 하나의 데이터베이스 서버가 수행됩니다.) MSCS 클러스

터에서 워크스테이션 중 하나가 실패할 경우, 실패하는 머신의 데이터베이스 서버는 다른 머신에서 수행되도록 시작됩니다. 상호 인계 구성에서는 한 머신의 데이터베이스 서버가 다른 머신에 있는 데이터베이스 서버와 독립적으로 실패할 수 있습니다. 어떠한 데이터베이스 서버라도 특정 지점에서 모든 머신에서 활성화될 수 있습니다. 그림30에서는 상호 인계 구성의 예를 보여줍니다.

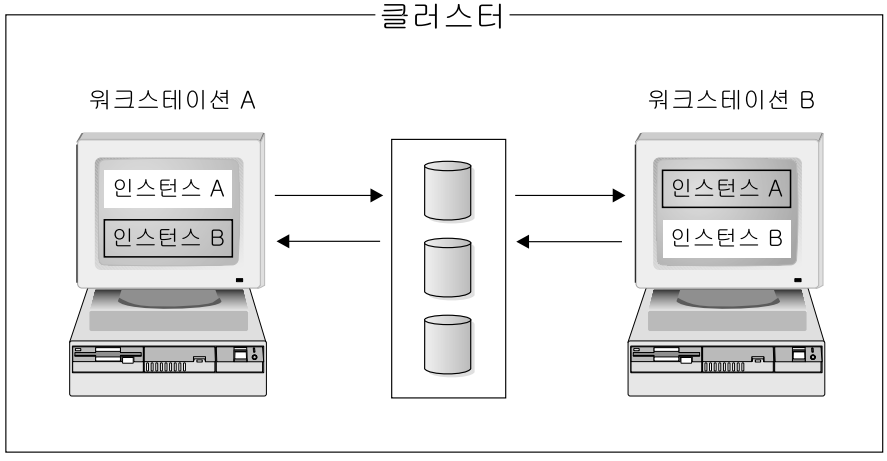


그림 30. 상호 인계 구성

DB2MSCS 유틸리티 사용

MSCS(Microsoft Cluster Service) 지원을 사용하는 Windows NT 환경에서 장애 복구를 지원하기 위해 DB2에 대해 하부구조를 작성하려면 DB2MSCS 유틸리티를 사용하십시오. 이 유틸리티를 사용하여 단일 파티션과 파티션된 데이터베이스 환경에서 장애 복구가 작동되도록 할 수 있습니다.

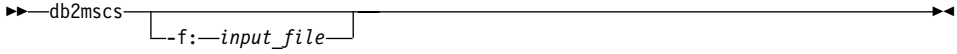
DB2MSCS 유틸리티가 제대로 실행되도록 하려면, Cluster Service가 `%ProgramFiles%\SQLLIB\bin` 디렉토리에 있는 자원 DLL `db2wolf.dll`을 찾을 수 있어야 합니다. DB2 UDB 버전 7 설치 프로그램은 PATH 시스템 환경 변수가 `%ProgramFiles%\SQLLIB\bin` 디렉토리를 지시하도록 설정합니다. 그러나 Windows 2000 운영 체제에서 수행 중이면 설치 후에 머신을 재부트하지 않아도 됩니다. DB2MSCS 유틸리티를 수행하려면, Cluster Service에 대해 PATH 환경 변수가 갱신되도록 머신을 재부트해야 합니다.

인스턴스를 소유하는 머신에서 각 인스턴스에 대해 한 번씩 **db2mscs** 명령을 발행하십시오. 하나의 DB2 인스턴스가 MSCS 클러스터의 한 머신에서 수행되고 있을 경우에만 긴급 대기 구성을 설정합니다. MSCS 클러스터의 각 머신에서 한 인스턴스가 수행되고 있을 경우, 각 인스턴스 소유 머신에서 한 번씩 DB2MSCS를 수행하여 상호 인계 구성을 설정합니다.

DB2MSCS 유틸리티

1. DB2MSCS.CFG라고 하는 입력 파일에서 필요한 MSCS 및 DB2 매개변수를 읽으십시오. 완전한 입력 매개변수 세트에 대한 정보는 262 페이지의 『DB2MSCS.CFG 파일 지정』에서 자세한 내용을 참조하십시오.
2. 입력 파일에서 매개변수의 유효성을 확인하십시오.
3. DB2 자원 유형을 등록하십시오.
4. MSCS와 DB2 자원을 포함하도록 MSCS 그룹을 작성하십시오.
5. IP 자원을 작성하십시오.
6. 네트워크 이름 자원을 작성하십시오.
7. MSCS 디스크를 그룹으로 이동시키십시오.
8. DB2 자원을 작성하십시오.
9. DB2 자원에 대해 필요한 모든 종속성을 추가하십시오.
10. 클러스터되지 않은 DB2 인스턴스를 클러스터된 인스턴스로 변환하십시오.
11. 모든 자원을 온라인 상태로 하십시오.

명령 구문은 다음과 같습니다.



여기서,

-f:*input_file*

MSCS 유틸리티에서 사용될 DB2MSCS.CFG 입력 파일을 지정합니다. 이 매개변수를 지정하지 않으면, DB2MSCS 유틸리티는 현재 디렉토리에 있는 DB2MSCS.CFG 파일을 읽습니다.

DB2MSCS.CFG 파일 지정

DB2MSCS.CFG 파일은 DB2MSCS 유틸리티로부터 읽혀지는 매개변수를 포함하는 ASCII 텍스트 파일입니다. PARAMETER_KEYWORD=*parameter_value* 형식을 사용하여 별도의 행에 각 입력 매개변수를 지정합니다. 예를 들면, 다음과 같습니다.

```
CLUSTER_NAME=WOLFPACK
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

두 가지의 구성 파일 예가 /SQLLIB 디렉토리의 /CFG 서브디렉토리에 있습니다. 첫 번째인 DB2MSCS.EE는 단일 파티션 데이터베이스 환경에 대한 예입니다. 두 번째인 DB2MSCS.EEE는 파티션된 데이터베이스 환경에 대한 예입니다.

DB2MSCS.CFG 파일에 대한 매개변수는 다음과 같습니다.

DB2_INSTANCE

DB2 인스턴스의 이름. 인스턴스 이름을 지정하지 않은 경우, 기본 인스턴스(DB2INSTANCE 환경 변수의 값)가 사용됩니다.

이 매개변수는 전역 범위를 가지며, 이를 DB2MSCS.CFG 파일에서 한 번만 지정하십시오.

이 매개변수는 선택적입니다.

예를 들면, 다음과 같습니다.

```
DB2_INSTANCE=DB2
```

인스턴스가 이미 존재하고 있어야 합니다. 인스턴스 작성에 대해서는 *Windows용 DB2 Enterprise - Extended Edition 빠른 시작* 책에서 자세한 내용을 참조하십시오.

DB2_LOGON_USERNAME

DB2 서비스에 대한 로그온 계정의 이름

이 매개변수는 전역 범위를 가지며, 이를 DB2MSCS.CFG 파일에서 한 번만 지정하십시오.

이 매개변수는 DB2 Enterprise - Extended Edition 인스턴스에 대해서만 필요합니다.

예를 들면, 다음과 같습니다.

```
DB2_LOGON_USERNAME=db2user
```

DB2_LOGON_PASSWORD

DB2 서비스에 대한 로그온 계정의 암호. DB2_LOGON_USERNAME 매개변수는 제공하고 DB2_LOGON_PASSWORD 매개변수는 제공하지 않은 경우, DB2MSCS 유틸리티는 암호를 입력하라는 프롬프트를 표시합니다. 암호를 명령행에 입력할 때 암호는 화면에 표시되지 않습니다.

이 매개변수는 전역 범위를 가지며, 이를 DB2MSCS.CFG 파일에서 한 번만 지정하십시오.

이 매개변수는 DB2 Enterprise - Extended Edition 인스턴스에 대해서만 필요합니다.

예를 들면, 다음과 같습니다.

```
DB2_LOGON_PASSWORD=xxxxxx
```

CLUSTER_NAME

MSCS 클러스터의 이름. 이 행 다음에 지정되는 모든 자원은 다른 CLUSTER_NAME 태크를 지정할 때까지 이 클러스터에서 작성됩니다.

각 클러스터에 대해 이 매개변수를 한 번만 지정하십시오.

이 매개변수는 선택적입니다. 지정하지 않으면, 지역 머신에 있는 MSCS 클러스터의 이름이 사용됩니다.

예를 들면, 다음과 같습니다.

```
CLUSTER_NAME=WOLFPACK
```

GROUP_NAME

MSCS 그룹의 이름. 이 매개변수를 지정할 경우, 새 MSCS 그룹이 존재하지 않으면 새로운 MSCS 그룹이 작성됩니다. 그룹이 이미 존재하면, 그룹은 목표 그룹으로 사용됩니다. 이 행 다음에 작성된 MSCS 자원들은 다른 GROUP_NAME 키워드가 지정될 때까지 이 그룹에서 작성됩니다.

각 그룹에 대해 이 매개변수를 한 번만 지정하십시오.

이 매개변수는 필수입니다.

예를 들면, 다음과 같습니다.

```
GROUP_NAME=DB2 Group
```

DB2_NODE

현재 MSCS 그룹에 포함될 데이터베이스 파티션 서버(노드)의 노드 번호. 다중 논리 노드가 같은 머신에 있을 경우, 각 노드에 대해 별도의 DB2_NODE 키워드가 필요합니다.

해당되는 MSCS 그룹에서 DB2 자원이 작성되도록 GROUP_NAME 매개변수 다음에 이 매개변수를 지정하십시오.

이 매개변수는 DB2 Enterprise - Extended Edition 인스턴스에 대해서만 필요합니다.

예를 들면, 다음과 같습니다.

```
DB2_NODE=0
```

IP_NAME

IP 주소 자원의 이름. IP_NAME의 값은 일정하지는 않지만, 고유해야 합니다. 이 매개변수를 지정할 경우, IP 주소 유형의 MSCS 자원이 작성됩니다.

원격 TCP/IP 연결에 대해 이 매개변수는 필수입니다. 파티션된 데이터베이스 환경에서는 인스턴스 소유 머신에 대해 이 매개변수를 반드시 지정해야 합니다. 이 매개변수는 단일 파티션 데이터베이스 환경에서 선택적입니다.

예를 들면, 다음과 같습니다.

```
IP_NAME=IP Address for DB2
```

주: DB2 클라이언트는 이 IP 자원의 TCP/IP 주소를 사용하여 TCP/IP 노드 항목을 카탈로그화해야 합니다. MSCS IP 주소를 사용하면, 데이터베이스 서버가 다른 머신에 대해 장애 복구할 때, DB2 클라이언트가 계속해서 데이터베이스 서버에 연결할 수 있습니다. 왜냐하면, 장애 복구 머신에서 IP 주소를 사용할 수 있기 때문입니다.

IP 자원의 속성은 다음과 같습니다.

IP_ADDRESS

IP 자원의 TCP/IP 주소. 이 키워드를 앞의 IP 자원에 대한 TCP/IP 주소를 설정하도록 지정하십시오.

이 매개변수는 IP_NAME 매개변수를 지정한 경우 필수입니다.

예를 들면, 다음과 같습니다.

```
IP_ADDRESS=9.21.22.34
```

IP_SUBNET

선행 IP 자원에 대한 서브넷 마스크

이 매개변수는 IP_NAME 매개변수를 지정한 경우 필수입니다.

예를 들면, 다음과 같습니다.

```
IP_SUBNET=255.255.255.0
```

IP_NETWORK

선행 IP 자원이 속하는 MSCS 네트워크의 이름. 이 매개변수를 지정하지 않으면, 시스템에서 검출되는 첫 번째 MSCS 네트워크가 사용됩니다.

이 매개변수는 선택적입니다.

예를 들면, 다음과 같습니다.

```
IP_NETWORK=Token Ring
```

NETNAME_NAME

네트워크 이름 자원의 이름. 네트워크 이름 자원을 작성하려면 이 매개변수를 지정하십시오.

이 매개변수는 단일 파티션 데이터베이스 환경에 대해 선택적입니다. 파티션된 데이터베이스 환경에 대해서는 필수입니다.

예를 들면, 다음과 같습니다.

```
NETNAME_NAME=Network name for DB2
```

네트워크 이름 자원의 속성은 다음과 같습니다.

NETNAME_VALUE

네트워크 이름의 값

이 매개변수는 NETNAME_NAME 매개변수를 지정한 경우 필수입니다.

예를 들면, 다음과 같습니다.

```
NETNAME_VALUE=DB2SRV
```

NETNAME_DEPENDENCY

네트워크 이름 자원에 대한 종속성 목록. 각 네트워크 이름 자원은 IP 주소 자원에서 종속성을 가지고 있어야 합니다. 이 매개변수를 지정하지 않은 경우, 네트워크 이름 자원은 그룹에서 첫 번째 IP 자원에 대한 종속성을 갖습니다.

이 매개변수는 선택적입니다.

예를 들면, 다음과 같습니다.

```
NETNAME_DEPENDENCY=IP Address for DB2
```

DISK_NAME

현재 그룹으로 이동될 물리적 디스크 자원의 이름. 필요한 만큼 디스크 자원을 지정하십시오.

주:

1. 디스크 자원이 이미 존재하고 있어야 합니다.
2. DB2MSCS 유틸리티가 MSCS 지원을 위한 DB2 인스턴스를 구성할 경우, 인스턴스 디렉토리는 그룹에서 *첫 번째* MSCS 디스크에 복사됩니다. 인스턴스 디렉토리에 대해 서로 다른 MSCS 디스크를 지정할 경우, INSTPROF_DISK 매개변수를 사용하십시오.

예를 들면, 다음과 같습니다.

```
DISK_NAME=Disk E:
DISK_NAME=Disk F:
```

INSTPROF_DISK

DB2 인스턴스 디렉토리를 포함할 MSCS 디스크를 지정하기 위한 선택적 매개변수. 이 매개변수를 지정하지 *않으면*, DB2MSCS 유틸리티는 인스턴스 디렉토리와 같은 그룹에 속하는 *첫 번째* MSCS 디스크를 사용합니다.

DB2 인스턴스 디렉토리는 `X:\DB2PROFS` 디렉토리하의 MSCS 디스크에서 작성됩니다. (X는 MSCS 디스크 드라이브명입니다.)

예를 들면, 다음과 같습니다.

```
INSTPROF_DISK=Disk E:
```

TARGET_DRVMAP_DISK

데이터베이스 드라이브 맵핑에 대해 목표 MSCS 디스크를 지정하는 선택적 매개변수. 노드로서 동일한 그룹에 속하지 않는 MSCS 디스크에 데이터베이스가 작성된 경우, 목표 드라이브 맵 디스크는 데이터베이스 파티션을 포함하기 위해 사용됩니다. 이 매개변수가 지정되지 않으면, 데이터베이스 드라이브 맵핑은 DB2DRVMP 유틸리티를 사용하여 수동으로 등록되어야 합니다.

예를 들면, 다음과 같습니다.

```
TARGET_DRVMAP_DISK = Disk E:
```

단일 파티션 데이터베이스 시스템에 대한 장애 복구 설정

단일 파티션 데이터베이스 시스템에 대해 DB2MSCS 유틸리티를 수행할 때, 하나의 MSCS 그룹에 DB2와 모든 종속되는 MSCS 자원(IP 주소, 네트워크 이름 및 디스크)이 있습니다. 예를 들어, 단일 파티션 데이터베이스 시스템에 대한 DB2MSCS.CFG 파일의 내용은 다음과 같습니다.

```
#
#  DB2MSCS.CFG for a single-partition database system
#
DB2_INSTANCE=DB2
CLUSTER_NAME=MSCS
GROUP_NAME=DB2 Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

두 개의 단일 파티션 데이터베이스 시스템에 대한 상호 인계 구성 설정

별도의 머신에 각각 두 개의 단일 파티션 데이터베이스 시스템을 설정할 수 있습니다. 그래서 하나의 머신에 있는 데이터베이스 시스템이 실패할 경우 다른 MSCS 노드에서 재시작될 수 있습니다.

이 구성에 대한 장애 복구 지원을 설정하려면, 각 인스턴스 소유 머신에서 DB2MSCS 유틸리티를 한 번 수행해야 합니다. 각 데이터베이스 시스템에 맞게 구성 파일을 조정해야 합니다.

DB2 인스턴스를 DB2A 및 DB2B라고 가정하십시오. DB2A 인스턴스의 DB2MSCS.CFG 파일은 다음과 같습니다.

```
#
#  DB2MSCS.CFG for first single-partition database system
#
DB2_INSTANCE=DB2A
CLUSTER_NAME=MSCS
GROUP_NAME=DB2A Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

DB2B 인스턴스의 DB2MSCS.CFG 파일은 다음과 같습니다.

```
#
#  DB2MSCS.CFG for second single-partition database system
#
DB2_INSTANCE=DB2B
CLUSTER_NAME=MSCS
GROUP_NAME=DB2B Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk F:
```

완전한 예는 275 페이지의 『예 - 상호 인계에 대한 두 개의 단일 파티션 인스턴스 설정』에서 자세한 내용을 참조하십시오.

파티션된 데이터베이스 시스템에 대한 다중 MSCS 클러스터 설정

다중 파티션 데이터베이스 시스템에 대해 DB2MSCS 유틸리티를 수행할 때, 시스템에 관여하는 각 물리적 머신에 대해 하나의 MSCS 그룹이 작성됩니다.

DB2MSCS.CFG 파일에는 여러 개의 섹션이 있어야 합니다. 각 섹션에는 GROUP_NAME 매개변수에 대한 서로 다른 값과 해당 그룹에 대한 모든 필수 종속 자원이 있어야 합니다.

또한, 각 MSCS 그룹에서 각 데이터베이스 파티션 서버에 대해 DB2_NODE 매개변수를 지정해야 합니다. 다중 논리 노드가 있을 경우, 각 논리 노드는 별도의 DB2_NODE 키워드를 요구합니다.

예를 들어, 네 개의 머신에서 네 개의 데이터베이스 파티션 서버로 이루어진 다중 파티션 데이터베이스 시스템이 있고, 상호 인계 구성을 사용하는 두 개의 MSCS 클러스터 구성을 원할 경우, 다음과 같이 DB2MSCS.CFG 구성 파일을 설정해야 합니다.

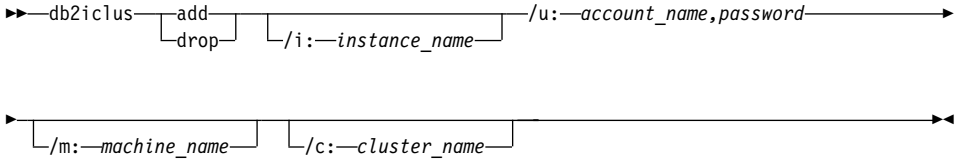
```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple clusters
DB2_INSTANCE=DB2MPP
DB2_LOGON_USERNAME=db2user
DB2_LOGON_PASSWORD=xxxxxx
CLUSTER_NAME=MSCS1
# Group 1
GROUP_NAME=DB2 Group 1
DB2_NODE=0
IP_NAME=...
...
# Group 2
GROUP_NAME=DB2 Group 2
DB2_NODE=1
IP_NAME=...
...
CLUSTER_NAME=MSCS2
# Group 3
GROUP_NAME=DB2 Group 3
DB2_NODE=2
IP_NAME=...
...
# Group 4
GROUP_NAME=DB2 Group 4
DB2_NODE=3
IP_NAME=...
...
```

완전한 예는 278 페이지의 『예 - 상호 인계에 대한 4 노드 파티션된 데이터베이스 시스템 설정』에서 자세한 내용을 참조하십시오.

MSCS 시스템 유지보수

DB2MSCS 유틸리티를 수행할 때, 그 유틸리티는 MSCS 클러스터에서 모든 머신에 대해 장애 복구 지원을 위한 하부구조를 작성합니다. 머신에서 지원을 제거하려면 **db2iclus** 명령에서 "drop" 옵션을 사용하십시오. 머신에 대해 지원이 다시 작동 가능하도록 하려면 "add" 옵션을 사용하십시오.

명령 구문은 다음과 같습니다.



여기서,

add MSCS 클러스터에 추가하여 머신에서 장애 복구 지원이 작동 가능하게 합니다. 그러면 DB2 자원(데이터베이스 서버)이 이 머신에 대해 장애 복구할 수 있습니다.

drop MSCS 클러스터에서 이를 삭제하여 머신으로부터 장애 복구 지원을 제거합니다.

/i: *instance_name* 인스턴스의 이름. (이 매개변수는 DB2INSTANCE 환경 변수의 설정을 대체합니다.)

/u: *account_name*, *password* DB2 서비스의 로그온 계정 이름으로 사용되는 도메인 계정 예를 들면, 다음과 같습니다.

```
/u:domainA\db2nt,password
```

이 매개변수는 "add" 옵션에 대해서만 필수입니다.

/m:*machine_name*	MSCS 클러스터에 추가하거나 제거할 머신의 컴퓨터 이름 장애 복구 지원을 수정하는 머신이 아닌 다른 머신으로부터 명령을 수행할 경우, 이 옵션을 지정해야 합니다.
/c: *cluster_name*	LAN에 알려진 대로의 MSCS 클러스터 이름. 이 이름은 MSCS 클러스터가 처음 작성될 때 지정됩니다.

후진 고려사항

기본값으로, 그룹은 원래(실패한) 머신으로 후진되지 않도록 설정됩니다. 장애 복구 후에 후진하도록 DB2 그룹을 수동으로 구성하지 않으면, DB2 그룹은 장애 복구 해결 후에도 계속해서 대체 MSCS 노드에서 수행됩니다.

DB2 그룹이 자동으로 원래의 머신으로 후진하도록 구성할 경우, DB2 자원을 포함한 DB2 그룹의 모든 자원은 원래 머신이 사용 가능하게 되면 바로 후진됩니다. 후진중에 데이터베이스 연결이 존재할 경우, DB2 자원은 오프라인 상태로 할 수 없어서 후진 처리에 실패합니다.

모든 데이터베이스 연결이 후진 처리중에 강제로 단절되도록 하려면, DB2_FALLBACK 레지스트리 변수를 ON으로 설정하십시오. 이변수는 다음과 같이 설정되어야 합니다.

```
db2set DB2_FALLBACK=ON
```

이 레지스트리 변수를 설정한 후에 클러스터 서비스를 재부트하거나 재시작할 필요가 없습니다.

파티션된 데이터베이스 환경에서의 상호 인계 구성에 대한 데이터베이스 드라이브 맵핑 등록

파티션된 데이터베이스 환경에서 데이터베이스를 작성할 때, 드라이브 이름을 지정하여 데이터베이스가 작성될 위치를 나타낼 수 있습니다.

주: 단일 파티션 데이터베이스 환경에 대해 데이터베이스 드라이브 맵핑을 설정하지 마십시오.

CREATE DATABASE 명령이 수행될 때, 이 명령은 사용자가 지정하는 드라이브가 인스턴스에 참여하는 모든 머신에 대해 동시에 사용 가능하게 되기를 기대합니다. 이것은 가능하지 않으므로, DB2는 데이터베이스 드라이브 맵핑을 사용하여 각 머신에 대해 서로 다른 이름을 같은 드라이브에 지정합니다.

예를 들어, 이름이 DB2인 DB2 인스턴스에 두 개의 데이터베이스 파티션 서버가 있다고 가정하십시오.

```
NODE0 is active on machine WOLF_NODE_0
NODE1 is active on machine WOLF_NODE_1
```

공유 디스크 E:가 NODE0과 같은 그룹에 속하고, 공유 디스크 F:가 NODE1과 같은 그룹에 속한다고 가정하십시오.

공유 디스크 E에서 데이터베이스를 작성하려면 다음과 같이 하십시오.

```
db2 create database mppdb on e:
```

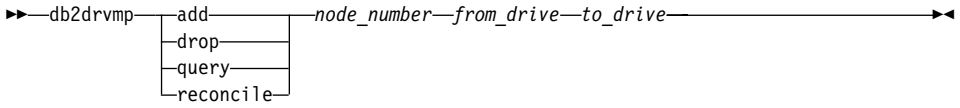
명령이 성공하려면, 드라이브 E:가 두 머신 모두에 대해 사용 가능해야 합니다. 상호 인계 구성에서, 각 데이터베이스 파티션 서버는 서로 다른 머신에서 활동 상태일 수 있으며, 클러스터 디스크 E:는 하나의 머신에 대해 유일하게 사용 가능한 것입니다. 이러한 상황에서는 CREATE DATABASE 명령이 항상 실패합니다.

이러한 문제점을 해결하려면, 데이터베이스 드라이브가 다음과 같이 맵핑되어야 합니다.

```
For NODE0, the mapping is from drive F: to drive E:
For NODE1, the mapping is from drive E: to drive F:
```

그러면 드라이브 F:에 대한 NODE0의 데이터베이스 액세스는 드라이브 E:에 맵핑되고, 드라이브 E:에 대한 NODE1의 데이터베이스 액세스는 드라이브 F:에 맵핑됩니다. 드라이브 맵핑을 사용할 경우, CREATE DATABASE 명령은 NODE0에 대해 드라이브 E:에서, NODE1에 대해 드라이브 F:에서 데이터베이스 파일을 작성합니다.

드라이브 맵핑을 설정하려면 **db2drvmp** 명령을 사용하십시오. 명령 구문은 다음과 같습니다.



매개변수는 다음과 같습니다.

add 새로운 데이터베이스 드라이브 맵을 지정합니다.

drop 기존의 데이터베이스 드라이브 맵을 제거합니다.

query 데이터베이스 맵핑 조회.

reconcile 레지스트리 내용이 손상될 경우 데이터베이스 맵 드라이브를 수리합니다. 274 페이지의 『데이터베이스 드라이브 맵핑 조정』에서 자세한 내용을 참조하십시오.

node_number 노드 번호. 이 매개변수는 추가 및 삭제 조작에 대해 필수입니다.

from_drive 맵핑이 시작될 드라이브 이름. 이 매개변수는 추가 및 삭제 조작에 대해 필수입니다.

to_drive 맵핑 대상이 될 드라이브 이름. 이 매개변수는 추가 조작에 대해 필수입니다. 다른 조작에는 적용할 수 없습니다.

NODE0에 대해 F:에서 E:로 데이터베이스 드라이브 맵핑 설정을 원하면, 다음 명령을 사용해야 합니다.

```
db2drvmp add 0 F E
```

주: 데이터베이스 드라이브 맵핑은 테이블 공간, 컨테이너 또는 다른 데이터베이스 저장영역 오브젝트에는 적용되지 않습니다.

마찬가지로, NODE1에 대해 드라이브 E에서 드라이브 F로 데이터베이스 드라이브 맵핑을 설정하려면, 다음 명령을 발행해야 합니다.

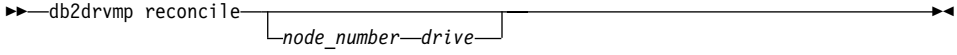
```
db2drvmp add 1 E F
```

주: 데이터베이스 드라이브 맵핑에 대한 설정 또는 변경사항은 즉시 적용되지 않습니다. 데이터베이스 드라이브 맵핑을 활성화하려면, 클러스터 관리자 도구를 사용하여 DB2 자원을 오프라인 상태로 한 후 다시 온라인 상태로 하십시오.

DB2MSCS.CFG 파일에서 TARGET_DRVMAP_DISK 키워드를 사용하면 드라이브 맵핑이 자동으로 완료될 수 있습니다.

데이터베이스 드라이브 맵핑 조정

데이터베이스 드라이브 맵핑이 적용되는 머신에서 데이터베이스를 작성할 경우, 그 맵은 해당 드라이브에서 숨겨진 파일에 저장됩니다. 이렇게 하면 데이터베이스 드라이브가 데이터베이스 작성 후에 제거되지 않게 됩니다. 예를 들어, 실수로 데이터베이스 드라이브 맵을 삭제한 경우 데이터베이스 드라이브 맵핑을 조정하려고 할 것입니다. 맵을 조정하려면, 데이터베이스를 포함하는 각 데이터베이스 파티션 서버에 대해 **db2drvmp reconcile** 명령을 수행하십시오. 명령 구문은 다음과 같습니다.



매개변수는 다음과 같습니다.

node_number 수리할 노드의 번호. *node_number*를 지정하지 않을 경우, 명령은 모든 노드에 대한 맵핑을 조정합니다.

drive 조정할 드라이브. 드라이브를 지정하지 않으면, 명령은 모든 드라이브에 대한 맵핑을 조정합니다.

db2drvmp 명령은 데이터베이스 파티션 서버에 의해 관리되는 머신에서 데이터베이스 파티션에 대한 모든 드라이브를 스캔하고, 데이터베이스 드라이브 맵핑을 필요에 따라 레지스트리에 다시 적용합니다.

예 - 상호 인계에 대한 두 개의 단일 파티션 인스턴스 설정

이 예의 목적은 상호 인계 구성에서 장애 복구 지원을 사용하여 두 개의 단일 파티션 데이터베이스 인스턴스를 설정하는 것입니다. 이 예에서, 네 개의 서버가 두 개의 MSCS 클러스터에 구성됩니다. 상호 인계 구성을 사용하면, 머신이 실패할 때 해당 머신에 대해 구성된 데이터베이스 서버가 MSCS 소프트웨어를 사용하여 구성된 것처럼 대체 머신에 대해 장애 복구하고 대체 머신에서 수행합니다.

결과 구성에는 두 개의 MSCS 클러스터가 있습니다. 각 클러스터에는 다음이 있습니다.

- 각각 메모리가 64MB이고 하나의 지역 SCSI 디스크가 2GB인 두 개의 서버
- 각각 2GB인 세 개의 공유 SCSI 디스크가 있는 하나의 SCSI 디스크 타워

또한, 각 머신에는 하나의 100X 이더넷 어댑터 카드가 설치되어 있습니다.

각 머신에는 다음과 같은 소프트웨어가 설치되어 있습니다.

- MSCS 기능이 있는 Windows NT 버전 4.0 Enterprise Edition
- DB2 Universal Database Enterprise Edition 버전 7

결과 네트워크 구성은 다음과 같습니다.

서버 1:	서버 2:
• 머신 이름: db2test1	• 머신 이름: db2test2
• TCP/IP 호스트 이름: db2test1	• TCP/IP 호스트 이름: db2test2
• IP 주소: 9.9.9.1 (서브넷 마스크: 255.255.255.0)	• IP 주소: 9.9.9.2 (서브넷 마스크: 255.255.255.0)
• MSCS 클러스터 이름: ClusterA	• MSCS 클러스터 이름: ClusterA

네트워크에서 두 머신 모두 TCP/IP에 대해 구성되고 이더넷 100 T-base Hub를 사용하여 전용 LAN에 연결됩니다. 도메인 이름 서버(DNS)가 없을 때, 모든 머신에는 다음과 같은 항목을 가진 지역 TCP/IP 호스트 파일이 있습니다.

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 ClusterA # for MSCS ClusterA
9.9.9.4 db2tcp1 # for DB2 remote client connection to Server 1
9.9.9.5 db2tcp2 # for DB2 remote client connection to Server 2
```

예비 타스크

다음 타스크를 수행하기 전에, 두 머신 모두 DB2NTD라고 하는 같은 도메인에 속해 있다고 가정하십시오.

1. DB2가 수행될 해당 머신에서 지역 관리자 그룹의 구성원인 도메인 계정을 DB2에 대해 작성하십시오. 모든 타스크를 수행하려면 계정을 사용하십시오.
   - 사용자 이름을 db2nt로 설정하십시오.
   - 암호를 db2nt로 설정하십시오.
2. db2test1 및 db2test2 머신에 MSCS 기능을 설치하십시오.
   - MSCS 클러스터에 ClusterA라고 이름을 붙이십시오.
   - 클러스터 IP 주소는 9.9.9.3입니다.
   - 공유 디스크 D:는 MSCS 소프트웨어가 사용합니다.
   - 공유 디스크 E:와 F:는 DB2가 사용합니다.
3. db2test1 머신에 DB2 Universal Database Enterprise Edition 버전 7을 설치하십시오. 지역 드라이브인 C:\SQLLIB에 소프트웨어를 설치하십시오.
4. db2test2 머신에 DB2 Universal Database Enterprise Edition 버전 7을 설치하십시오. 지역 드라이브인 C:\SQLLIB에 소프트웨어를 설치하십시오.

다음 단계는 각 인스턴스에 대해 DB2MSCS.CFG 파일을 설정한 후 각 인스턴스에 대해 DB2MSCS 유틸리티를 수행하는 것입니다.

DB2MSCS 유틸리티 수행

db2test1 머신을 설정하려면, 다음 타스크를 수행하십시오.

1. db2test1 머신에서, 사용자 db2nt로 로그온하십시오. 암호는 db2nt입니다.
2. DB2 인스턴스 DB2A를 작성하십시오(아직 존재하지 않을 경우). 인스턴스를 작성하기 위한 명령은 다음과 같습니다.

   ```
   db2icrt DB2A
   ```

3. db2test1 머신에서 DB2 인스턴스에 대해 DB2MSCS.CFG 파일을 설정하십시오.

```
#
#  DB2MSCS.CFG for database system
#  on machine db2test1
DB2_INSTANCE=DB2A
CLUSTER_NAME=ClusterA
#
# Group 1
GROUP_NAME=DB2A Group
IP_NAME=IP Address for DB2A
IP_ADDRESS=9.9.9.4
IP_SUBNET=255.255.255.0
IP_NETWORK=ClusterA
NETNAME_NAME=Network name for DB2A
NETNAME_VALUE=DB2SRV1
NETNAME_DEPENDENCY=IP Address for DB2A
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
```

4. 다음과 같이 DB2MSCS 유틸리티를 수행하십시오.

```
db2mscs -f:DB2MSCS.CFG
```

5. db2nt 계정에서 로그아웃하십시오.
6. 머신 db2test2에서, 지역 관리자 그룹에 속하는 사용자 db2nt로 로그온하십시오. 암호는 db2nt입니다.
7. DB2 인스턴스 DB2B를 작성하십시오(아직 존재하지 않을 경우). 인스턴스를 작성하기 위한 명령은 다음과 같습니다.

```
db2icrt DB2B
```

8. db2test2 머신에서 DB2 인스턴스에 대해 DB2MSCS.CFG 파일을 설정하십시오.

```
#
#  DB2MSCS.CFG for database system
#  on machine db2test2
DB2_INSTANCE=DB2B
CLUSTER_NAME=ClusterA
#
# Group 1
GROUP_NAME=DB2B Group
IP_NAME=IP Address for DB2B
IP_ADDRESS=9.9.9.5
IP_SUBNET=255.255.255.0
```

```
IP_NETWORK=ClusterA
NETNAME_NAME=Network name for DB2B
NETNAME_VALUE=DB2SRV2
NETNAME_DEPENDENCY=IP Address for DB2B
DISK_NAME=Disk F:
INSTPROF_DISK=Disk F:
```

9. 다음과 같이 DB2MSCS 유틸리티를 수행하십시오.

```
db2mscs -f:DB2MSCS.CFG
```

10. db2nt 계정에서 로그아웃하십시오.

예 - 상호 인계에 대한 4 노드 파티션된 데이터베이스 시스템 설정

이 예의 목적은 상호 인계 구성에서 장애 복구 지원을 사용하여 4 노드 파티션된 데이터베이스 시스템을 설정하는 것입니다. 이 예에서, 네 개의 서버가 두 개의 MSCS 클러스터에 구성됩니다. 상호 인계 구성을 사용하면, 머신이 실패할 때 해당 머신에 대해 구성된 데이터베이스 파티션 서버가 MSCS 소프트웨어를 사용하여 구성된 것처럼 대체 머신에 대해 장애 복구하고 대체 머신에서 논리 노드로서 수행합니다.

결과 구성에는 두 개의 MSCS 클러스터가 있습니다. 각 클러스터에는 다음이 있습니다.

- 각각 메모리가 64MB이고 하나의 지역 SCSI 디스크가 2GB인 두 개의 서버
- 각각 2GB인 세 개의 공유 SCSI 디스크가 있는 하나의 SCSI 디스크 타워

또한, 각 머신에는 하나의 100X 이더넷 어댑터 카드가 설치되어 있습니다.

각 머신에는 다음과 같은 소프트웨어가 설치되어 있습니다.

- MSCS 기능이 있는 Windows NT 버전 4.0 Enterprise Edition
- DB2 Universal Database Extended Enterprise Edition 버전 7

결과 네트워크 구성은 다음과 같습니다.

서버 1: • 머신 이름: db2test1 • TCP/IP 호스트 이름: db2test1 • IP 주소: 9.9.9.1 (서브넷 마스크: 255.255.255.0) • MSCS 클러스터 이름: ClusterA	서버 2: • 머신 이름: db2test2 • TCP/IP 호스트 이름: db2test2 • IP 주소: 9.9.9.2 (서브넷 마스크: 255.255.255.0) • MSCS 클러스터 이름: ClusterA
서버 3: • 머신 이름: db2test3 • TCP/IP 호스트 이름: db2test3 • IP 주소: 9.9.9.3 (서브넷 마스크: 255.255.255.0) • MSCS 클러스터 이름: ClusterB	서버 4: • 머신 이름: db2test4 • TCP/IP 호스트 이름: db2test4 • IP 주소: 9.9.9.4 (서브넷 마스크: 255.255.255.0) • MSCS 클러스터 이름: ClusterB

네트워크의 모든 머신은 TCP/IP에 대해 구성되고 이더넷 100 T-base Hub를 사용하여 전용 LAN에 연결됩니다. 도메인 이름 서버(DNS)가 없을 때, 모든 머신에는 다음과 같은 항목을 가진 지역 TCP/IP 호스트 파일이 있습니다.

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 db2test3 # for Server 3
9.9.9.4 db2test4 # for Server 4
9.9.9.5 ClusterA # for MSCS Cluster 1
9.9.9.6 ClusterB # for MSCS Cluster 2
9.9.9.7 db2tcp # for DB2 remote client connection
```

예비 타스크

다음 타스크를 수행하기 전에, 네 개의 머신 모두 DB2NTD라고 하는 같은 도메인에 속해 있다고 가정하십시오.

1. DB2가 수행될 해당 머신에서 지역 관리자 그룹의 구성원인 도메인 계정을 DB2에 대해 작성하십시오. 모든 타스크를 수행하려면 계정을 사용하십시오.
   - 사용자 이름을 db2nt로 설정하십시오.
   - 암호를 db2nt로 설정하십시오.

2. "*암호 만기 안됨*" 특성으로 두 번째 도메인 계정을 작성합니다. 이 계정은 DB2 서비스와 연관됩니다.
   - 사용자 이름을 db2mpp로 설정하십시오.
   - 암호를 db2mpp로 설정하십시오.
3. db2test1 및 db2test2 머신에 MSCS 기능을 설치하십시오.
   - MSCS 클러스터에 ClusterA라고 이름을 붙이십시오.
   - 클러스터 IP 주소는 9.9.9.5입니다.
   - 공유 디스크 D:는 MSCS 소프트웨어가 사용합니다.
   - 공유 디스크 E:와 F:는 DB2가 사용합니다.
4. db2test3 및 db2test4 머신에 MSCS 기능을 설치하십시오.
   - MSCS 클러스터에 ClusterB라고 이름을 붙이십시오.
   - 클러스터 IP 주소는 9.9.9.6입니다.
   - 공유 디스크 D:는 MSCS 소프트웨어가 사용합니다.
   - 공유 디스크 E:와 F:는 DB2가 사용합니다.
5. DB2 Enterprise - Extended Edition을 db2test1 머신에 설치하십시오.
   - "*이 머신은 인스턴스 소유 데이터베이스 파티션 서버임*" 옵션을 선택하십시오.
   - DB2 서비스의 계정은 db2mpp입니다. 암호는 db2mpp입니다.
   - 지역 드라이브인 C:\SQLLIB에 소프트웨어를 설치하십시오.
6. db2test2, db2test3 및 db2test4 머신에 DB2 Enterprise - Extended Edition을 설치하십시오.
   - "*이 머신은 기존의 파티션된 데이터베이스 시스템의 새 노드임*" 옵션을 선택하십시오.
   - 인스턴스 소유 머신으로 db2test1을 선택하십시오.
   - DB2 서비스의 계정은 db2mpp입니다. 암호는 db2mpp입니다.
   - 지역 드라이브인 C:\SQLLIB에 소프트웨어를 설치하십시오.

다음 단계는 DB2MSCS.CFG 파일을 설정한 후 DB2MSCS 유틸리티를 수행하는 것입니다.

DB2MSCS 유틸리티 수행

db2test1 머신을 설정하려면, 다음 타스크를 수행하십시오.

1. 지역 관리자 그룹에 속하는 사용자 db2nt로 로그온하십시오. 암호는 db2nt입니다.
2. DB2MSCS.CFG 파일을 설정하십시오.

```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple MSCS clusters
DB2_INSTANCE=DB2MPP
CLUSTER_NAME=ClusterA
DB2_LOGON_USERNAME=db2mpp
DB2_LOGON_PASSWORD=db2mpp
# Group 1
# for DB2 node 0
GROUP_NAME=DB2NODE0
DB2_NODE=0
IP_NAME=IP Address for DB2
IP_ADDRESS=9.9.9.7
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
NETNAME_NAME=Network name for DB2
NETNAME_VALUE=DB2WOLF
NETNAME_DEPENDENCY=IP Address for DB2
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
#

# Group 2
# for DB2 node 1
GROUP_NAME=DB2NODE1
DB2_NODE=1
DISK_NAME=Disk F:
#

CLUSTER_NAME=ClusterB
# Group 3
# for DB2 node 2
GROUP_NAME=DB2NODE2
DB2_NODE=2
DISK_NAME=Disk E:
```

```
#
# Group 4
# for DB2 node 3
GROUP_NAME=DB2NODE3
DB2_NODE=3
DISK_NAME=Disk F:
```

3. 다음과 같이 DB2MSCS 유틸리티를 수행하십시오.

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

4. db2nt 계정에서 로그아웃하십시오.

마지막 단계는 두 MSCS 클러스터에 대해 데이터베이스 드라이브 맵핑을 등록하는 것입니다.

ClusterA에 대한 데이터베이스 드라이브 맵핑 등록

MSCS 클러스터 ClusterA에 대해 데이터베이스 드라이브 맵핑을 등록하려면, 다음 타스크를 수행하십시오.

1. db2test1 머신에서, 사용자 db2mpp로 로그온하십시오. 이 사용자는 DB2 서비스와 연관되는 계정입니다. 암호는 db2mpp입니다.
2. 데이터베이스 드라이브 맵핑을 등록하려면, 다음 명령을 입력하십시오.

   ```
   db2drvmp add 0 F E
   db2drvmp add 1 E F
   ```

3. 모든 DB2 자원을 오프라인 상태로 한 다음, 다시 온라인 상태로 하십시오.

ClusterB에 대한 데이터베이스 드라이브 맵핑 등록

MSCS 클러스터 ClusterB에 대해 데이터베이스 드라이브 맵핑을 등록하려면, 다음 타스크를 수행하십시오.

1. db2test3 머신에서, 사용자 db2mpp로 로그온하십시오. 이 사용자는 DB2 서비스와 연관되는 계정입니다. 암호는 db2mpp입니다.
2. 데이터베이스 드라이브 맵핑을 등록하려면, 다음 명령을 입력하십시오.

   ```
   db2drvmp add 2 F E
   db2drvmp add 3 E F
   ```

3. 모든 DB2 자원을 오프라인 상태로 한 다음, 다시 온라인 상태로 하십시오.

MSCS 환경에서의 DB2 관리

MSCS 클러스터를 사용 중인 경우, DB2 인스턴스는 일상 조작, 데이터베이스 전개 및 데이터베이스 구성에 관한 추가 계획을 요구합니다. DB2가 MSCS 노드에서 투명하게 실행되도록 하려면, 추가 관리 타스크를 수행해야 합니다. 모든 DB2 종속 운영 체제 자원은 모든 MSCS 노드에서 사용 가능해야 합니다. 이 운영 체제 자원 중 일부는 MSCS 범위 밖에 있습니다. 즉, 일부 자원은 MSCS 자원으로 정의될 수 없습니다. 같은 운영 체제 자원이 모든 MSCS 노드에서 사용될 수 있도록 각 시스템을 구성해야 합니다. 다음 절에서 수행해야 하는 추가 작업에 대해 설명합니다.

DB2 자원 시작 및 중단

클러스터 관리자 도구에서 DB2 자원을 시작하고 중단해야 합니다. **db2start** 명령 및 제어판의 **서비스** 옵션과 같은 몇 가지의 메커니즘을 사용하여 DB2 인스턴스를 시작할 수 있습니다. 그러나 DB2가 클러스터 관리자로부터 시작되지 않은 경우, MSCS 소프트웨어는 DB2 인스턴스의 상태를 인식하지 못합니다. DB2 인스턴스가 클러스터 관리자를 사용하여 시작되고 **db2stop** 명령을 사용하여 중단된 경우, MSCS 소프트웨어는 **db2stop** 명령을 소프트웨어 실패로 분석하고 DB2를 재시작하려고 합니다. (현재 MSCS 인터페이스는 *자원 상태* 통지를 지원하지 않습니다.)

마찬가지로, **db2start**를 사용하여 DB2 인스턴스를 시작할 경우, MSCS는 자원이 온라인임을 검출할 수 없습니다. 데이터베이스 서버가 실패하면, MSCS는 클러스터 내의 파일복구 머신에서 DB2 자원을 온라인으로 가져오지 않습니다.

세 가지의 조작이 DB2 인스턴스에 적용될 수 있습니다.

온라인 이 조작은 **db2start** 명령을 사용하는 것과 같습니다. DB2가 이미 활동중이면, 이 조작은 간단히 DB2가 활동중임을 MSCS에 통지하면 사용할 수 있습니다. 이 조작중에 발생하는 오류는 Windows NT 이벤트 로그에 기록됩니다.

오프라인

이 조작은 **db2stop** 명령을 사용하는 것과 같습니다. 인스턴스에 대한 어

떠한 활동중인 접속도 없는 경우, 이 조작은 실패합니다. 이 조작은 **db2stop**의 동작과 동일하게 이루어집니다.

실패 자원

이 조작은 **db2stop** 명령에서 **force** 옵션을 지정하는 것과 같습니다. DB2는 모든 응용프로그램을 DB2 시스템에서 단절시켜 모든 데이터베이스 서버를 중단시킵니다.

스크립트 수행

DB2 지원이 온라인이 된 후와 그 이전에 스크립트를 수행합니다. 이들 스크립트는 DB2INSTPROF 환경 변수에 대해 지정된 인스턴스 프로파일 디렉토리에 상주*해야* 합니다. 이 디렉토리는 **db2icrt** 명령에서 ″-p″ 매개변수에 의해 지정된 디렉토리 경로입니다. 다음 명령을 발행하여 이 값을 확보할 수 있습니다.

```
db2set -i:instance_name DB2INSTPROF
```

이 파일 경로는 인스턴스 디렉토리가 모든 클러스터 노드에서 사용 가능하도록 클러스터된 디스크에 있어야 합니다.

이 스크립트 파일은 반드시 필요한 것은 아니며, 인스턴스 디렉토리에 있을 경우에만 수행됩니다. 파일은 백그라운드에서 MSCS 클러스터 서비스에 의해 시작됩니다. 스크립트 파일은 스크립트 파일 내의 명령에서 리턴된 모든 정보를 캡쳐하도록 표준 출력을 경로 재지정해야 합니다. 출력은 화면에 표시되지 않습니다.

기본적으로 파티션된 데이터베이스 환경에서는 동일한 스크립트가 인스턴스의 모든 데이터베이스 파티션 서버에서 사용됩니다. 인스턴스에서 서로 다른 데이터베이스 파티션 서버 사이에 구분이 필요할 경우, 특정 노드 번호를 목표로 하는 서로 다른 DB2NODE 환경 변수 지정을 사용하십시오. (예를 들어, `db2cpre.bat` 및 `db2cpost.bat` 파일에서 IF문을 사용합니다.)

DB2 자원을 온라인 상태로 하기 전에 스크립트 수행

DB2 자원을 온라인으로 하기 *전에* 스크립트를 수행하려면, 스크립트의 이름을 `db2cpre.bat`로 *지정해야 합니다*. DB2는 Windows NT 명령행 처리기(CLP)로부터 이 일괄처리 파일을 시작하고 DB2 자원이 온라인이 되기 전에 실행을 완료할 CLP를 기다리는 기능을 호출합니다. DB2 데이터베이스 관리 프로그램 구성 수정과 같은 타스크에 이 일괄처리 파일을 사용할 수 있습니다. 장애 복구 시스템

이 제한되는 상황에서 DB2에서 소모되는 시스템 자원을 줄여야 할 경우, 일부 데이터베이스 관리 프로그램 매개변수 값을 변경할 수 있습니다.

db2cpre.bat 스크립트에 있는 명령은 동기식으로 실행되어야 합니다. 그렇지 않으면, DB2 자원은 스크립트의 모든 타스크가 완료되기 전에 온라인 상태로 가서 예기치 않은 동작을 일으킬 수 있습니다. 특히, **db2cmd**는 db2cpre.bat 스크립트에서 호출되지 말아야 합니다. 그렇게 하면, 그 다음에 다른 명령 프로세서를 실행하고, 그 명령 프로세서는 **db2cmd** 프로그램에 대해 비동기식으로 DB2 명령을 실행합니다.

DB2 CLP 명령을 db2cpre.bat 스크립트에서 사용하려면, 명령은 파일에 위치되어 DB2 명령행 처리기에 대해 DB2 환경을 초기화하는 CLP 일괄처리 파일로서 프로그램 내에서 실행되어야 하며, 그런 다음 DB2 명령행 처리기의 완료를 기다려야 합니다. 예를 들면, 다음과 같습니다.

```
#include <windows.h>
int WINAPI DB2SetCLPEnv_api(DWORD pid);
void main ( int argc, char *argv [ ] )
{
      STARTUPINFO         startInfo   = {0};
      PROCESS_INFORMATION pidInfo     = {0};
      char     title  [32]   = "Run Synchronously";
      char     runCmd [64]  =
                             "DB2 -z c:\\run.out -tvf c:\\run.clp";
/* Invoke API to set up a CLP Environment */
      if ( DB2SetCLPEnv_api (GetCurrentProcessId ()) == 0 ) 1
      {
         startInfo.cb          = sizeof(STARTUPINFO);
         startInfo.lpReserved  = NULL;
         startInfo.lpTitle     = title;
         startInfo.lpDesktop   = NULL;
         startInfo.dwX         = 0;
         startInfo.dwY         = 0;
         startInfo.dwXSize     = 0;
         startInfo.dwYSize     = 0;
         startInfo.dwFlags     = 0L;
         startInfo.wShowWindow = SW_HIDE;
         startInfo.lpReserved2 = NULL;
         startInfo.cbReserved2 = 0;
               if ( CreateProcessA( NULL,
                              runCmd, 2
                              NULL,
                              NULL,
```

```
                                FALSE,
                                NORMAL_PRIORITY_CLASS CREATE_NEW_CONSOLE,
                                NULL,
                                NULL,
                                &startInfo,
                                &pidInfo ) )
            {
               WaitForSingleObject (pidInfo.hProcess, INFINITE);
               CloseHandle (pidInfo.hProcess);
               CloseHandle (pidInfo.hThread);
            }
         }
         return;
}
```

1 API DB2SetCLPEnv_api는 가져오기 라이브러리 DB2API.LIB에 의해분석됩니다. 이 API는 CLP 명령이 호출될 수 있도록 하는 환경을 설정합니다. 이 프로그램이 db2cpre.bat 스크립트로부터 호출될 경우, 명령 프로세서는 CLP 명령의 완료를 기다립니다.

2 runCmd는 DB2 CLP 명령을 포함하는 스크립트 파일의 이름입니다.

db2clpex.exe라고 하는 샘플 프로그램이 DB2 설치 경로의 MISC 서브디렉토리에 있습니다. 이 실행 가능한 파일은 제공된 예와 유사하지만, DB2 CLP 명령을 명령행 인수로 받아들입니다. 이 샘플 프로그램을 사용하려면, 이를 BIN 서브디렉토리에 복사하십시오. 다음과 같이, db2cpre.bat 스크립트에서 이 실행 가능 파일을 사용할 수 있습니다(INSTHOME은 인스턴스 디렉토리임).

```
db2clpex "DB2 -Z INSTHOME\pre.log -tvf INSTHOME\pre.clp"
```

All DB2 ATTACH 명령 또는 CONNECT문은 명시적으로 사용자를 지정해야 합니다. 그렇지 않으면, 클러스터 서비스와 연관되는 사용자 계정하에서 실행됩니다. CLP 스크립트는 또한 TERMINATE 명령으로 완료되어 CLP 백그라운드 프로세스를 종료해야 합니다.

다음은 db2cpre.bat 파일의 예입니다.

```
db2cpre.bat : 1
-----------------------
db2clpex "db2 -z INSTHOME\pre-%DB2NODE%.log -tvf INSTHOME\pre.clp" 2 - 5
-----------------------
PRE.CLP 6
-----------------------
```

```
update dbm cfg using MAXAGENTS 200;
get dbm cfg;
terminate;
------------------------
```

1 db2cpre.bat 스크립트는 클러스터 서비스와 연관되는 사용자 계정하에서 실행됩니다. DB2 조치가 필요하면, 클러스터 서비스와 연관되는 사용자 계정이 DB2에 정의된 유효한 SQL 식별자여야 합니다.

2 INSTHOME은 인스턴스 디렉토리입니다.

3 로그 파일의 이름은 두 논리 노드를 동시에 온라인 상태로 가져올 때 파일 결합을 피하기 위해 각 노드에 대해 달라야 합니다.

4 db2clpex.exe는 명령행 인수를 호출할 CLP 명령을 지정하기 위해 사용하는 샘플 프로그램입니다.

5 db2clpex.exe 샘플 프로그램은 모든 MSCS 클러스터 노드에서 사용가능하게 해야 합니다.

6 이 예의 CLP 명령은 에이전트 수에 대한 한계를 설정합니다.

DB2 자원을 온라인 상태로 한 후에 스크립트 수행

DB2 자원을 온라인으로 한 *후에* 스크립트를 수행하려면, 스크립트의 이름을 db2cpost.bat로 *지정해야 합니다*. 스크립트는 DB2 자원이 온라인 상태가 된 후에 MSCS에서 비동기식으로 수행됩니다. DB2 CLP 스크립트 파일을 실행하기 위해 이 스크립트에서 **db2cmd** 명령을 사용할 수 있습니다. 유틸리티가 타스크 완료시 모든 창을 닫도록 지정하려면 **db2cmd** 명령의 ″-c″ 매개변수를 사용하십시오. 예를 들면, 다음과 같습니다.

```
db2cmd -c db2 -tvf mycmds.clp
```

″-c″ 매개변수는 **db2cmd** 명령에 대해 첫 번째 인수여야 합니다. 이 매개변수는 백그라운드에서 Orphaned 명령 프로세서를 방지하기 때문입니다.

db2cpost.bat 스크립트는 DB2 자원이 장애 복구되어 활동 상태가 된 후에 즉시 데이터베이스 활동을 수행하고자 할 경우에 유용합니다. 예를 들어, 데이터베이스에 사용자가 액세스할 수 있도록 인스턴스에서 데이터베이스를 재시작하거나 활성화할 수 있습니다.

다음은 db2cpost.bat 스크립트의 예입니다.

```
db2cpost.bat 1
------------------------
db2cmd -c db2 -z INSTHOME\post-%DB2NODE%.log -tvf INSTHOME\post.clp 2 - 4
------------------------
POST.CLP 5
------------------------
restart database SAMPLE;
connect reset;
activate database SAMPLE;
terminate;
------------------------
```

1 db2cpost.bat 스크립트는 클러스터 서비스와 연관되는 사용자 계정하에서 수행됩니다. DB2 조치가 필요하면, 클러스터 서비스와 연관되는 사용자 계정이 DB2에 정의된 유효한 SQL 식별자여야 합니다.

2 INSTHOME은 인스턴스 디렉토리입니다.

3 로그 파일의 이름은 두 논리 노드를 동시에 온라인 상태로 가져올 때 파일 결합을 피하기 위해 각 노드에 대해 달라야 합니다.

4 db2cpost.bat 스크립트는 비동기적으로 실행할 수 있으므로 **db2cmd** 명령을 사용할 수 있습니다. "-c" 매개변수는 명령 프로세스를 종료할 때 사용해야 합니다.

5 이 예의 CLP 스크립트에는 데이터베이스를 재시작하고 활성화하기 위한 명령이 있습니다. 이 스크립트는 데이터베이스 관리 프로그램이 시작되고 나면 즉시 데이터베이스를 활동중인 상태로 리턴됩니다. 파티션된 데이터베이스 시스템에서, 여러 DB2 자원이 동시에 온라인 상태가 되므로 ACTIVATE DATABASE 명령을 제거해야 합니다. 다른 노드가 데이터베이스를 활성화하고 있으므로 RESTART DATABASE 명령이 실패할 수 있습니다. 이러한 상황이 발생할 경우, 데이터베이스가 올바르게 재시작되도록 스크립트를 다시 수행하십시오.

데이터베이스 고려사항

데이터베이스를 작성할 때, 데이터베이스 경로가 공유 디스크를 참조하는지 확인하십시오. 이렇게 하면, 데이터베이스를 모든 MSCS 노드에서 볼 수 있습니다. 모든 로그 및 다른 데이터베이스 파일도 DB2가 장애 복구하도록 클러스터된 디스크를 참조해야 합니다. 이들 단계를 수행하지 않으면, DB2에는 파일이 삭제되거나 사용할 수 없는 것으로 보여지므로 DB2 시스템 실패가 발생합니다.

또한, 데이터베이스 관리 프로그램 및 데이터베이스 구성 매개변수가 DB2에 의해 소모되는 시스템 자원 양이 MSCS 노드에서 지원되도록 설정되어 있는지 확인하십시오. *autorestart* 데이터베이스 구성 매개변수는 장애 복구에서의 첫 번째 데이터베이스 연결로 데이터베이스가 일관되는 상태가 되도록 ON으로 설정해야 합니다. (*autorestart*에 대한 기본 설정값은 ON입니다.) 데이터베이스는 또한 데이터베이스를 재시작하고 활성화하기 위해 db2cpost.bat 스크립트를 사용하여 준비 상태가 되도록 할 수 있습니다. 이 방법은 *autorestart*에 영향을 주지 않고, 데이터베이스가 사용자 연결 요청과는 독립적으로 준비 상태가 될 수 있어서 선호하는 방법입니다.

사용자 및 그룹 지원

DB2는 사용자 인증 및 그룹 지원에 대해 Windows NT에 의존합니다. 이음이 없는 형태에서 한 DB2 노드에서 다른 DB2 노드로 장애 복구하는 DB2 인스턴스에 대해, 각 MSCS 노드는 같은 Windows NT 보안 데이터베이스에 대한 액세스 권한을 가지고 있어야 합니다. Windows NT 도메인 보안을 사용하여 이를 아카이브할 수 있습니다.

도메인 보안 데이터베이스에서 모든 DB2 사용자 및 그룹을 정의하십시오. MSCS 노드는 이 도메인의 구성원이어야 하고 도메인은 신뢰된 도메인이어야 합니다. 그러면 DB2는 DB2가 실행되는 MSCS 노드와 독립적으로 인증 및 그룹 지원을 위해 도메인 보안 데이터베이스를 사용합니다.

지역 계정을 사용할 경우, 계정은 각 MSCS 노드에서 복제되어야 합니다. 이러한 접근 방식은 오류 경향이 있어서 이중 유지보수를 필요로 할 수 있으므로 권장되지 않습니다.

모든 MSCS 노드가 같은 DCE 셀의 클라이언트인 경우, DCE 보안도 지원되는 인증 모드입니다.

MSCS 서비스는 DB2 명명 규약을 따르는 사용자 계정에 연관시켜야 합니다. 그러면 MSCS 서비스가 db2cpre.bat 및 db2cpost.bat 스크립트에서 요구될 수도 있는 조치를 DB2에 대해 수행할 수 있습니다.

Windows NT 사용자 및 그룹 지원에 대한 자세한 정보는 *관리 안내서: 구현*에 있는 "Windows NT용 DB2 사용자 인증"을 참조하십시오.

통신 고려사항

DB2는 MSCS 환경에서 두 개의 LAN 프로토콜을 지원합니다.

- TCP/IP
- NetBIOS

TCP/IP는 지원되는 클러스터 자원 유형이므로 지원됩니다. DB2가 파티션된 데이터베이스 시스템에 대한 통신 프로토콜로 TCP/IP를 사용할 수 있도록 하려면, IP 주소 자원을 작성하고 이를 원격 응용프로그램에 대해 조정자 노드로 사용할 데이터베이스 파티션 서버를 나타내는 DB2 자원과 같은 그룹에 두십시오. 그런 다음, IP 자원이 DB2 자원 시작 이전에 온라인이 되도록 클러스터 관리자 도구를 사용하여 종속성을 작성하십시오. 그러면 DB2 클라이언트는 TCP/IP 노드 디렉토리 항목을 카탈로그화하여 이 TCP/IP 주소를 사용할 수 있게 됩니다.

svcename 데이터베이스 관리 프로그램 구성 매개변수와 연관되는 TCP/IP 포트는 인스턴스에 참여하는 모든 머신에서 DB2 인스턴스가 사용할 수 있도록 예약되어야 합니다. 또한 포트 번호와 연관되는 서비스 이름은 모든 머신에 있는 서비스 파일에서 같아야 합니다.

NetBIOS가 지원되는 클러스터 자원이 아니어도, 프로토콜은 LAN에서 NetBIOS 이름이 고유하도록 하므로 LAN 프로토콜로 NetBIOS를 사용할 수 있습니다. DB2가 NetBIOS 이름을 등록할 경우, NetBIOS는 이름이 LAN에서 사용 중이 아닌지 확인합니다. 장애 복구 시나리오에서, DB2가 시스템간에 이동되면, DB2에서 사용되는 *nname*은 MSCS 클러스터에 있는 하나의 상대 머신에서 등록이 취소되고 다른 머신에 등록됩니다.

DB2 NetBIOS 지원은 NetBIOS 프레임(NBF)을 사용합니다. 이 프로토콜 스택은 서로 다른 논리 어댑터 번호(LANA)와 연관될 수 있습니다. 서버에 대한 일관성 있는 NetBIOS 액세스가 가능하도록, NBF 프로토콜 스택과 연관되는 LANA가 모든 클러스터된 노드에서 같아야 합니다. 제어판에서 **네트워크** 옵션을 사용하면 이를 구성할 수 있습니다. NBF를 LANA 0과 연관시켜야 합니다. 이것이 DB2가 예상하는 기본 설정값이기 때문입니다.

시스템 시간 고려사항

DB2는 특정 조작의 시간소인 측정을 위해 시스템 시간을 사용합니다. DB2 장애 복구에 참여하는 모든 MSCS 노드는 DB2가 모든 머신에서 일관성 있게 동작하도록 시스템 시간대 및 시스템 시간이 동기화되어야 합니다.

제어판 대화 상자에서 **날짜/시간** 옵션을 사용하여 시스템 시간대를 설정하십시오. MSCS는 MSCS 노드가 클러스터를 형성하기 위해 조인될 때 날짜 및 시간을 동기화하는 시간 서비스를 수반합니다. 그러나 이 시간 서비스는 12시간마다 시간을 동기화하므로 한 시스템에서 시간이 변경되고, 그 시간이 동기화되기 전에 DB2가 실패할 경우 문제점이 발생할 수도 있습니다.

MSCS 클러스터 노드 중 하나에서 날짜가 변경되면, 시간은 다음 명령을 사용하여 다른 클러스터 노드에서 수동으로 동기화되어야 합니다.

```
net time /set /y \\remote_node
```

여기서, *remote_node*는 클러스터 노드의 머신 이름입니다.

파티션된 데이터베이스 환경의 관리 서버 및 제어 센터 고려사항

DB2 관리 서버는 DB2 Universal Database 설치중에 (선택적으로) 작성됩니다. 이것은 파티션된 데이터베이스 시스템이 아닙니다. 제어 센터는 DB2 인스턴스 및 데이터베이스를 관리하기 위해 관리 서버에 의해 제공되는 서비스를 사용합니다.

파티션된 데이터베이스 시스템에서, DB2 인스턴스는 다중 MSCS 노드에 상주할 수 있습니다. 즉, DB2 인스턴스는 DB2 인스턴스가 활동중인 MSCS 노드에 관계없이 액세스 가능한 상태로 유지되도록 제어 센터하의 다중 시스템에서 카탈로그화되어야 합니다.

관리 서버 인스턴스 디렉토리는 공유되지 않습니다. 관리 서버 디렉토리의 모든 사용자 정의 파일은 모든 MSCS 노드로 이중복사하여 모든 MSCS 노드에 대해 같은 레벨의 관리를 제공해야 합니다. 특히, 사용자 스크립트와 스케줄된 실행 가능 파일을 모든 노드에서 사용할 수 있게 만들어야 합니다. 또한, 스케줄된 활동이 MSCS 클러스터의 모든 머신에서 스케줄되는지 확인해야 합니다.

또한, 모든 머신에서 관리 서버를 중복시키는 대신 관리 서버 장애 복구를 할 수도 있습니다. 다음 예의 경우 클러스터에 두 개의 MSCS 노드가 있고, 그 노드는

각각 MACH0 및 MACH1이라고 합니다. MACH0은 관리 서버에서 사용될 클러스터 디스크에 대한 액세스 권한을 가지고 있습니다. MACH0 및 MACH1 둘 다에 관리 서버가 있다고 가정하십시오. 다음 단계를 수행하여 관리 서버를 사용 가능성이 높게 합니다.

1. 각 머신에서 **db2admin stop** 명령을 호출하여 두 머신 모두에서 관리 서버를 중단시키십시오.
2. 모든 관리 클라이언트 머신에서, UNCATALOG NODE 명령을 사용하여 MACH0 및 MACH1에서의 관리 서버에 대한 모든 참조사항을 카탈로그 해제하십시오. (클라이언트 머신에서 LIST NODE DIRECTORY 명령을 사용하여 관리 서버에 대한 참조가 있는지를 판별할 수 있습니다.)
3. MACH1으로부터 **db2admin 삭제** 명령을 호출하여 MACH1에서 관리 서버를 삭제하십시오. (두 머신 모두에 관리 서버가 있을 경우에만 이 단계를 수행하십시오.)
4. MACH0에서 **db2admin** 명령을 발행하여 관리 서버의 이름을 판별하십시오. (기본값은 DB2DAS00입니다.)
5. 관리 서버에 대한 장애 복구 지원을 설정하려면 DB2MSCS 유틸리티를 사용하십시오. 여기에는 IP 및 디스크 자원에 대한 종속성을 가지고 있는 DB2DAS00 MSCS에서 DB2 자원을 작성하는 작업이 수반됩니다. (상호 인계 구성을 가지고 있으면, MACH0에 대한 DB2 자원을 보유하는 그룹에 자원을 둡니다.) 이 자원은 관리 서버를 지원하는 MSCS 자원으로 사용됩니다. DB2MSCS.ADMIN 파일은 다음과 같습니다.

```
#
# db2mscs.admin for Administration Server
# run db2mscs -f:db2mscs.admin
#
DB2_INSTANCE=DB2DAS00
CLUSTER_NAME=CLUSTERA
DB2_LOGON_USERNAME=db2admin
DB2_LOGON_PASSWORD=db2admin
# put Administration server in the same group as DB2 Node 0
GROUP_NAME=DB2NODE0 1
DISK_NAME=DISK E:
INSTPROF_DISK=DISK E:
```

```
IP_NAME= IP Address for Administration Server
IP_ADDRESS=9.9.9.8
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
```

1 이 그룹은 기존 그룹과 같을 수 있습니다. 이때, 인스턴스 프로파일 디렉토리를 위한 추가 디스크는 필요하지 않습니다.

6. MACH1에서 다음 명령을 호출하여 DB2DAS00을 관리 서버로 설정하십시오.

   ```
   db2set -g db2adminserver=DB2DAS00
   ```

7. MACH0에서, 서비스 프로그램을 통해 DB2DAS00의 시동 등록 정보를 수정하여 자동이 아닌 수동으로 가져오도록 하십시오. 이제 DB2DAS00은 MSCS에 의해 제어되기 때문입니다.

관리 서버가 장애 복구에 대해 작동 가능하게 될 때, 모든 원격 액세스는 관리 서버와의 통신을 위해 MSCS IP 자원을 사용해야 합니다. 이제 관리 서버는 다음과 같은 특성을 갖습니다.

- 관리 서버 인스턴스 디렉토리는 관리 서버를 사용하여 장애 복구합니다.
- 클라이언트는 단일 노드만을 카탈로그에 등록하여 활동중인 MSCS 노드에 관계없이 관리 서버와 통신합니다.
- 관리 서버에 대해 한 번만 작업 스케줄을 정하면 됩니다.
- 지역 인스턴스는 관리 서버가 지역 인스턴스 같은 MSCS 노드에서 활동중일 때 관리 서버에 의해서만 제어됩니다.
- 관리 서버는 클러스터 서비스가 활동중이 아닐 때에는 액세스할 수 없습니다.

한계 및 제한사항

MSCS 환경에서 DB2를 수행할 때,

- 공유 디스크에 MSCS 노드 모두에 대해 동일한 물리적 디스크 번호가 없으면 공유 디스크에서 물리적 입출력을 사용할 수 없습니다. 디스크는 파티션 식별자를 사용하여 액세스하므로 논리 입출력을 사용할 수 있습니다.
- MSCS 지원에 대해 모든 DB2 자원을 구성해야 합니다. 그렇게 하지 않으면, DB2 런타임 시스템 오류가 발생할 수 있습니다. (DB2는 시스템 자원이 없을 때 제대로 작동하지 않을 수 있습니다.) 예를 들어, 데이터베이스 로그가 MSCS 공유 디스크에 없으면, DB2는 데이터베이스를 다시 시작할 수 없습니다.

- 클러스터 관리자 도구에서 DB2 인스턴스를 관리해야 합니다. MSCS는 소프트웨어 불일치처럼 데이터베이스 관리 프로그램을 시작하고 중단하는데 사용되는 다른 메커니즘을 보여줍니다. 예를 들어, MSCS를 사용하여 DB2를 시작하고 **db2stop** 명령을 사용하여 DB2를 중단시킬 경우, MSCS는 이를 소프트웨어 실패로 간주하고 인스턴스를 재시작합니다. 이것은 또한 DB2를 시작하고 중단시키는데 제어 센터를 사용하지 말아야 한다는 것을 의미합니다.
- DB2 설치를 제거하려면, 먼저 MSCS를 중단시켜야 합니다.

제8장 Solaris 운영 환경의 고가용성

Solaris 운영 환경에서의 고가용성은 SC2.x(Sun Cluster 2.x), SC3.0(Sun Cluster 3.0) 또는 VCS(Veritas Cluster Server)에서의 DB2 작동을 통해 이루어집니다. Sun Cluster 3.0에 대해서는『DB2 and High Availability on Sun Cluster 3.0』백서를 참조하십시오. 이는『DB2 UDB and DB2 Connect Online Support』웹 사이트(http://www.ibm.com/software/data/pubs/papers/)에서 구할 수 있습니다. VERITAS Cluster Server에 대해서는『DB2 and High Availability on VERITAS Cluster Server』백서를 참조하십시오. 이는『DB2 UDB and DB2 Connect Online Support』웹 사이트(http://www.ibm.com/software/data/pubs/papers/)에서 구할 수 있습니다.

이 장에서는 DB2가 고가용성을 위해 Sun Cluster 2.x(SC2.x)와 작업하는 방식을 자세히 설명하고 두 개의 소프트웨어 제품 간의 조정자로서 역할하는 고가용성에 대한 설명이 들어 있습니다(그림31 참조).

그림 31. DB2, Sun Cluster 2.x 및 고가용성

고가용성

호스트 데이터 서비스에 여러 가지 구별 구성요소가 들어 있는 컴퓨터 시스템과 연관된 각 구성요소는 "실패 이전의 평균 시간"(MTBF)을 갖습니다. MTBF는 구성요소가 사용 가능한 채로 남아 있는 평균 시간입니다. 양질의 하드 드라이브의 MTBF는 백만 시간(대략 114년) 정도입니다. 이것은 긴 시간인 것 같지만, 200개 디스크 중 하나 정도는 6개월 이내에 고장납니다.

데이터 서비스의 사용 가능성을 증가시키는 몇 가지 방법이 있지만, 가장 일반적인 것은 HA 클러스터입니다. 고가용성에 사용될 때의 클러스터는 두 개 이상의 머신, 개별 네트워크 인터페이스 세트, 하나 이상의 네트워크 인터페이스 및 일부 공유 디스크로 구성됩니다. 이 특수한 구성은 데이터 서비스가 한 머신에서 다른 머신으로 이동되게 합니다. 클러스터에서 데이터 서비스를 다른 머신으로 이동함으로써, 관련된 데이터에 대한 액세스를 계속 제공할 수 있어야 합니다. 한 머신에서 다른 머신으로 데이터 서비스를 이동하는 것은 297 페이지의 그림32에서 설명한 것처럼 *장애 복구*라고 합니다.

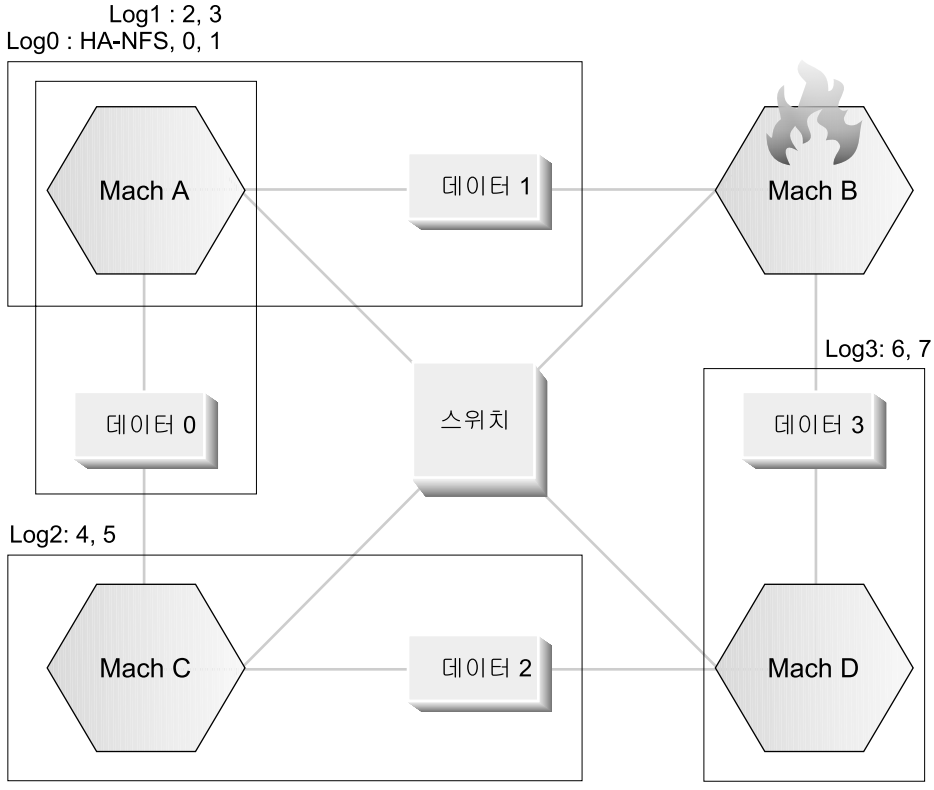


그림 32. 장애 복구

각 네트워크 인터페이스는 클러스터에서 머신간에 제어 메시지는 물론 *heartbeat* 메시지를 보내는데 사용됩니다. 공용 네트워크 인터페이스는 HA 클러스터의 클라이언트와 직접 통신하는데 사용됩니다. HA 클러스터에 있는 디스크는 클러스터에서 둘 이상의 머신으로 연결되어, 하나의 머신이 실패하면, 다른 머신이 액세스할 수 있습니다.

HA 클러스터에서 수행하는 데이터 서비스는 하나 이상의 논리 공용 네트워크 인터페이스 및 연관된 디스크 세트를 갖습니다. HA 데이터 서비스의 클라이언트는 TCP/IP를 통해 데이터 서비스 전용의 네트워크 인터페이스에 연결합니다. 장애 복구가 발생하면, 데이터 서비스는 연관된 논리 네트워크 인터페이스와 디스크 세트와 함께 다른 머신으로 이동됩니다.

HA 클러스터의 이점 중 하나는 데이터 서비스가 지원 담당자의 도움없이 복구할 수 있다는 것이며, 언제든지 수행할 수 있습니다. 또 다른 이점은 중복입니다. 클러스터 내의 모든 부분은 머신 자체를 포함하여 중복되어야 합니다. 클러스터는 어떤 실패 지점에서도 잔존해야 합니다.

매우 사용 가능한 데이터 서비스는 본래 매우 다를 수 있지만, 일부 공통적인 요구사항을 갖습니다. 매우 사용 가능한 데이터 서비스의 클라이언트는 네트워크 주소 및 데이터 서비스의 호스트 이름이 동일할 것으로 예상하며, 데이터 서비스가 설정되었는지와 무관하게 동일한 방식으로 요청을 작성할 수 있으리라 기대합니다.

매우 사용 가능한 웹 서버에 액세스하는 웹 브라우저를 고려하십시오. 호스트 이름과 웹 서버에서의 파일에 대한 경로 둘다가 들어 있는 URL(Uniform Resource Locator)로 요청을 발행합니다. 웹 서버의 장애 복구 이후에 호스트 이름과 경로가 둘다 동일하게 남아 있으리라 기대합니다. 브라우저가 웹 서버에서 파일을 다운로드하며 서버가 장애 복구되는 경우, 브라우저는 요청을 재발행할 필요가 있습니다.

데이터 서비스의 사용 가능성은 데이터 서비스가 관련 사용자에게 사용가능한 시간량에 의해 측정됩니다. 사용 가능성의 가장 일반적인 측정 단위는 "최대 시간"으로 이것은 보통 "9"의 숫자로 나타냅니다.

```
99.99% => service is down for (at most) 52.6 minutes / yr
99.999% => service is down for (at most) 5.26 minutes / yr
99.9999% => service is down for (at most) 31.5 seconds / yr
```

HA 클러스터를 디자인하고 테스트할 때 다음을 수행하십시오.

1. 클러스터의 관리자가 시스템과 친숙하고 장애 복구가 발생했을 때 어떤 일이 발생해야 하는가를 확인하십시오.
2. 클러스터의 각 부분이 정말로 중복되며 실패할 경우 신속하게 바꾸어질 수 있는지 확인하십시오.
3. 제어 환경에서 테스트 시스템이 실패하도록 강제 수행하고, 매번 정확하게 파일복구되는지 확인하십시오.
4. 각 파일복구의 이유의 트랙을 유지하십시오. 이것이 자주 발생하지는 않지만, 클러스터를 불안정하게 만드는 문제점을 처리하는 것이 중요합니다. 예를 들어, 클러스터의 한 부분이 한 달에 5번 장애 복구를 야기한 경우, 이유를 찾아 수정하십시오.
5. 장애 복구가 발생했을 때 클러스터의 지원 담당자에게 통지했는지 확인하십시오.

6. 클러스터를 초과적재하지 마십시오. 나머지 시스템이 장애 복구 이후에 수용가능한 레벨에서 워크로드를 아직 핸들할 수 있는지 확인하십시오.
7. 실패하기 쉬운 구성요소(디스크와 같은)를 자주 점검하여, 문제점이 발생하기 전에 바꿀 수 있게 하십시오.

결함 허용 한계 및 연속 사용 가능성

데이터 서비스의 사용 가능성을 증가시키는 다른 방법은 결함 허용 한계입니다. *결함 허용 한계* 머신은 모든 중복 내장 함수를 가지며, CPU 및 메모리를 포함하여 모든 부분의 단일 실패를 견딜 수 있어야 합니다. 결함 허용 한계 머신은 니치 마켓에서 가장 자주 사용되며, 보통 구현하는데 비용이 많이 듭니다. 여러가지 지리학적 위치에서 머신이 있는 HA 클러스터는 이 위치들의 부속 집합에만 영향을 미치는 재난에서 복구할 수 있는 추가된 이점을 갖습니다.

*연속 사용 가능성*은 고가용성 위의 단계입니다. 계획 및 비계획된 다운 시간 둘다에서 관련 클라이언트를 보호합니다. 연속 사용 가능성 구성을 사용하여, 데이터 서비스를 주관하는 머신 중 하나가 실패하거나 유지보수하도록 초래된 경우 클라이언트는 전혀 영향받지 않습니다. 연속 사용 가능성 구성은 복잡하며 구현하는데 비용이 많이 듭니다.

HA 클러스터는 확장가능하며, 사용하기 쉽고, 구현하는데 상대적으로 저렴하므로 사용 가능성을 증가시키는 가장 일반적인 솔루션입니다.

Sun Cluster 2.2

Sun Cluster 2.2(SC2.2)는 Sun Microsystems의 클러스터링 및 고가용성 제품입니다. SC2.2는 단일 클러스터에서 최대 4개의 머신까지 지원합니다. 4개의 Ultra Enterprise 10000을 사용하여, 클러스터는 최대 256개의 CPU와 256GB의 RAM을 가질 수 있습니다.

지원된 시스템

시스템	UltraSPARC	메모리 용량	입출력
Ultra Enterprise 1	1	64MB-1GB	3 SBus
Ultra Enterprise 2	1-2	64MB-2GB	4 SBus

Ultra Enterprise 450	1-4	32MB-4GB	10 PCI
Ultra Enterprise 3000	1-6	64MB-6GB	9 SBus
Ultra Enterprise 4000	1-14	64MB-14GB	21 SBus
Ultra Enterprise 5000	1-14	64MB-14GB	21 SBus
Ultra Enterprise 6000	1-30	64MB-30GB	45 SBus
Ultra Enterprise 10000	1-64	512MB-64GB	64 SBus

에이전트

Sun Cluster 소프트웨어는 지원되는 몇 개의 고가용성 에이전트를 포함하며 SC2.2 제품이 함께 제공됩니다. DB2 중 하나와 같이 다른 HA 에이전트는 Sun 외부에서 개발되어 Sun Cluster 소프트웨어가 제공되지 않습니다. DB2와 함께 제공된 DB2용 HA 에이전트는 IBM이 지원하며, DB2에 무료로 제공됩니다.

Sun Cluster 소프트웨어는 Sun Cluster 소프트웨어의 다양한 구성요소에 해당하는 메소드(스크립트 또는 프로그램)를 등록하는 기회를 제공하여 매우 사용 가능한 데이터 서비스로 작업합니다. 이 메소드를 활용하여, SC2.2 소프트웨어는 자세한 정보 없이 데이터 서비스를 제어할 수 있습니다. 이 메소드는 다음을 포함합니다.

START
: 논리 네트워크 인터페이스가 온라인이 되기 전에 데이터 서비스의 부분을 시작하기 위해 사용됩니다.

START_NET
: 논리 네트워크 인터페이스가 온라인이 된 후에 데이터 서비스의 부분을 시작하기 위해 사용됩니다.

STOP
: 논리 네트워크 인터페이스가 오프라인이 된 후에 데이터 서비스의 부분을 중지시키기 위해 사용됩니다.

STOP_NET
: 논리 네트워크 인터페이스가 오프라인이 되기 전에 데이터 서비스의 부분을 중지시키기 위해 사용됩니다.

ABORT
: STOP 메소드와 같으며, 머신이 클러스터 소프트웨어에 의해 다운되기 바

로 직전에 수행하는 것만 다릅니다. 이 경우, 머신의 "건전한 정도"는 알 수 없으며, 데이터 서비스는 머신이 다운되기 전의 "최종 희망" 요청을 실행하려고 할 수 있습니다. 논리 네트워크 인터페이스가 오프라인된 후에 실행하십시오.

ABORT_NET

ABORT 메소드와 같으나, 논리 네트워크 인터페이스가 오프라인되기 전에 수행하는 것만 다릅니다,

FM_INIT

결함 모니터를 초기화하는데 사용됩니다.

FM_START

결함 모니터를 시작하는데 사용됩니다.

FM_STOP

결함 모니터를 중지하는데 사용됩니다.

FM_CHECK

hactl 명령에 의해 호출됩니다. 해당 데이터 서비스의 현재 상태를 리턴합니다.

DB2 에이전트는 START_NET, STOP_NET, FM_START 및 FM_STOP과 같은 스크립트로 구성됩니다. ABORT, ABORT_NET 및 FM_CHECK 등의 스크립트는 클러스터 재구성 중에 수행하지 않습니다.

고가용성 에이전트는 하나 이상의 이 메소드들로 구성됩니다. 메소드는 **hareg** 명령을 통해 SC2.2로 등록됩니다. 등록되고 나면, Sun Cluster 소프트웨어가 해당 메소드를 호출하여 데이터 서비스를 제어합니다.

서비스의 ABORT 및 STOP 메소드는 호출되지 않을 수 있음을 기억하십시오. 이 메소드들은 데이터 서비스의 제어된 종료를 위해 사용하려는 것이며, 이 메소드들을 호출하지 않고 머신이 실패하는 경우 데이터 서비스가 복구될 수 있어야 합니다.

Sun Cluster 문서에서 자세한 내용을 참조하십시오.

논리 호스트

SC2.2 소프트웨어는 논리 호스트의 개념을 사용합니다. *논리 호스트*는 디스크 세트 및 하나 이상의 논리 공용 네트워크 인터페이스로 구성됩니다. 매우 사용 가능한 데이터 서비스는 논리 호스트와 연관되며, 논리 호스트의 디스크 그룹에 있는 디스크를 요구합니다. 논리 호스트는 클러스터에서 다른 머신에 의해 주관될 수 있으며, 수행 중인 머신의 CPU 및 메모리를 "빌립니다".

논리 네트워크 인터페이스

다른 UNIX 기반 운영 체제와 같이, Solaris는 네트워크 인터페이스에 대한 기본 IP 주소 이외에 여분의 IP 주소를 가지는 기능을 갖습니다. 여분의 IP 주소는 기본 IP 주소가 실제 네트워크 인터페이스에 상주하는 것과 같은 방식으로 논리 인터페이스에 상주합니다. 다음은 클러스터에서 두 개의 머신에 있는 논리 인터페이스의 예입니다. 두 개의 논리 호스트가 있으며, 둘다 현재 "thrash" 머신에 있습니다.

```
scadmin@crackle(202)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 289966 0 289966 0 0 0
hme0 1500 9.21.55.0 9.21.55.98 121657 6098 764122 0 0 0
scid0 16321 204.152.65.0 204.152.65.1 489307 0 476479 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.33 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.17 347317 0 348073 0 0 0
  1. lo0 is the loopback interface
  2. hme0 is the public network interface (ethernet)
  3. scid0 is the first private network interface (SCI or Scalable
     Coherent Interface)
  4. scid0:1 is a logical network interface that the Sun Cluster software
     uses internally
  5. scid1 is the second private network interface
scadmin@thrash(203)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 1128780 0 118780 0 0 0
hme0 1500 9.21.55.0 9.21.55.92 1741422 5692 757127 0 0 0
hme0:1 1500 9.21.55.0 9.21.55.109 0 0 0 0 0 0
hme0:2 1500 9.21.55.0 9.21.55.110 0 0 0 0 0 0
scid0 16321 204.152.65.0 204.152.65.2 476641 0 489476 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.34 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.18 348199 0 347444 0 0 0
  1. hme0:1 is a logical network interface for a logical host
  2. hme0:2 is a logical network interface for another logical host
```

논리 호스트는 연관된 하나 이상의 논리 인터페이스를 가질 수 있습니다. 이 논리 인터페이스는 머신에서 머신으로 논리 호스트와 함께 이동되며, 논리 호스트와 연

관된 데이터 서비스에 액세스하는데 사용됩니다. 이 논리 인터페이스는 논리 호스트와 함께 이동하므로, 클라이언트는 상주하는 머신과는 별개로 데이터 서비스에 액세스할 수 있습니다.

고가용성 데이터 서비스는 TCP/IP 주소 INADDR_ANY로 바인드해야 합니다. 이것은 시스템에 있는 각 IP 주소가 데이터 서비스에 대한 연결을 수용할 수 있음을 보장합니다. 데이터 서비스가 대신 특정 IP 주소에 바인드하는 경우, 데이터 서비스를 주관하는 논리 호스트와 연관된 논리 인터페이스를 바인드해야 합니다. INADDR_ANY로의 바인딩은 데이터 서비스에서 필요로 하는 시스템에 도달하면 새로운 IP 주소로 리바인드하는 필요성을 제거합니다.

주: HA 인스턴스의 클라이언트는 논리 호스트의 논리 IP 주소에 대한 호스트 이름을 사용하여 데이터베이스를 카탈로그해야 합니다. DB2가 해당 머신에서 수행 중일 것이라고 보장할 수 없으므로, 클라이언트는 머신의 기본 호스트 이름을 사용하지 말아야 합니다.

디스크 그룹 및 파일 시스템

데이터 서비스의 디스크는 그룹(또는 세트)에서 논리 호스트와 연관됩니다. 클러스터가 Sun StorEdge Volume Manager(Veritas)를 수행 중인 경우, Sun Cluster 소프트웨어는 Veritas ″vxdg″ 유틸리티를 사용하여 각 논리 호스트의 디스크 그룹을 가져오고 내보냅니다. 다음은 ″thrash″이라고 하는 머신에 의해 주관되는 두 개의 논리 호스트 ″log0″ 및 ″log1″의 디스크 그룹의 예입니다. ″crackle″이라고 하는 머신은 현재 임의의 논리 호스트를 주관하지 않습니다.

```
scadmin@crackle(206)# vxdg list
NAME STATE ID
rootdg enabled 899825206.1025.crackle
scadmin@thrash(205)# vxdg list
NAME STATE ID
rootdg enabled 924176206.1025.thrash
data0 enabled 925142028.1157.crackle=
data1 enabled 899826248.1108.crackle
```

디스크 그룹 ″data0″ 및 ″data1″은 논리 호스트 ″log0″ 및 ″log1″에 각각 해당합니다. 디스크 그룹 ″data0″은

```
vxdg deport data0
```

을 수행하여 ″thrash″에서 내보내지며 다음을 수행하여 ″crackle″로 가져옵니다.

```
vxdg import data1
```

이것은 Sun Cluster 소프트웨어에 의해 자동으로 수행되며, 활동 클러스터에서 수동으로 수행되지 말아야 합니다.

각 디스크 그룹에는 클러스터에서 두 개 이상의 머신 간에 공유될 수 있는 몇 개의 디스크가 들어 있습니다. 논리 호스트는 속한 디스크 그룹에 있는 디스크에 대한 실제 액세스를 갖는 다른 머신으로 이동만 될 수 있습니다.

각 논리 호스트에 대한 파일 시스템을 제어하는 두 개의 파일이 있습니다.

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
/etc/opt/SUNWcluster/conf/hanfs/dfstab.<logical_host>
```

여기서 *logical_host*는 연관된 논리 호스트 이름의 이름입니다.

vfstab 파일은 /etc/vfstab 파일과 유사하며, 단지 디스크 그룹이 논리 호스트용으로 가져온 후에 마운트되는 파일 시스템에 대한 항목이 들어 있는 것만 다릅니다. dfstab 파일은 /etc/dfs/dfstab 파일과 유사하며, 논리 호스트의 HA-NFS를 통해 내보내기 위해 파일 시스템에 대한 항목이 들어 있는 것만 다릅니다. 각 머신은 이 파일들에 대한 자체 사본을 가지며, 클러스터에 있는 각 머신에서 동일한 내용을 가지는지 확인하려면 주의해야 합니다.

주: 논리 호스트의 vfstab 및 dfstab 파일은 hanfs 디렉토리를 포함하므로 오해되기 쉽습니다. 논리 호스트의 dfstab 파일만이 HA-NFS용으로 사용됩니다. HA-NFS가 구성되지 않더라도 vfstab 파일이 사용됩니다.

다음은 상호 인계 구성에서 DB2 Universal Database EEE(Enterprise - Extended Edition)를 수행하는 클러스터의 예입니다.

```
scadmin@thrash(217)# ls -l /etc/opt/SUNWcluster/conf/hanfs
total 8
-rw-r--r-- 1 root build 173 Apr 14 15:01 dfstab.log0
-rw-r--r-- 1 root build 316 Apr 26 12:07 vfstab.log0
-rw-r--r-- 1 root build 389 Apr 13 21:04 vfstab.log1
scadmin@thrash(218)# cat dfstab.log0
share -F nfs -o root=crackle:thrash:\
jolt:bump:crackle.torolab.ibm.com:thrash.torolab.ibm.com:\
jolt.torolab.ibm.com:bump.torolab.ibm.com /log0/home
```

/log0/home 파일 시스템을 마운트하도록 사용권한이 제공된 호스트는 클러스터에 있는 각 머신의 모든 네트워크 인터페이스(논리 및 물리)에서 옵니다. 파일 시스템은 루트 사용권한으로 내보냅니다.

```
scadmin@thrash(220)# cat vfstab.log0
#device to mount             device  to fsck               mount
#                                                          point

/dev/vx/dsk/data0/data1-stat /dev/vx/rdsk/data0/data1-stat /log0
/dev/vx/dsk/data0/vol01      /dev/vx/rdsk/data0/vol01      /log0/home
/dev/vx/dsk/data0/vol02      /dev/vx/rdsk/data0/vol02      /log0/data

scadmin@thrash(221)# cat vfstab.log1
#device to mount             device  to fsck               mount
#                                                          point
/dev/vx/dsk/data1/data1-stat /dev/vx/rdsk/data1/data1-stat /log1
/dev/vx/dsk/data1/vol01      /dev/vx/rdsk/data1/vol01      /log1/home
/dev/vx/dsk/data1/vol02      /dev/vx/rdsk/data1/vol02      /log1/data
/dev/vx/dsk/data1/vol03      /dev/vx/rdsk/data1/vol03      /log1/data1
FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -


FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
```

vfstab.log0 파일에는 /log0 디렉토리 아래의 파일 시스템에 대한 세 가지 유효한 항목이 들어 있습니다. 논리 호스트 log0의 파일 시스템은 논리 호스트와 연관된 디스크 그룹 data0의 일부인 논리 볼륨 장치를 사용합니다.

vfstab 파일에 있는 파일 시스템은 맨 위에서 맨 아래의 순서로 마운트되므로, 파일 시스템이 올바른 순서로 나열되어 있음을 보장하는 것이 중요합니다. 특정 파일 시스템 아래에 마운트된 파일 시스템은 특정 파일 시스템 아래에 나열되어야 합니다. 논리 호스트용으로 필요한 실제 파일 시스템은 데이터 서비스의 필요성에 의해 좌우되며, 이 예와는 상당히 다릅니다.

장애 복구 중, SC2.2 소프트웨어는 논리 호스트와 연관된 디스크 그룹 및 논리 인터페이스가 머신에서 머신까지 클러스터 주변에서 뒤따르는지 확인하는데 관여합니다. 매우 사용 가능한 데이터 서비스는 장애 복구 이후에 새 시스템에서 최소한 이 자원들을 사용 가능하게 하려고 기대합니다. 사실, 많은 데이터 서비스는 매우 사용 가능한 지 인식할 수 조차 없으며 장애 복구 이후에 이 자원이 똑같이 "표시되게" 해야 합니다.

제어 방법

제어 방법은 다음을 사용하여 등록됩니다.

```
hareg(1m)
```

HA 서비스가 등록되고 나면, SC2.2는 클러스터 재구성 또는 장애 복구 중 적절한 시간에 HA 서비스용으로 등록된 메소드를 호출하는데 관여합니다.

다음 조치는 클러스터 재구성(제어된 장애 복구) 중 (제공된 순서로) 발생합니다. 머신에 장애가 있는 경우, 5단계를 선행하는 조치는 취해지지 않습니다. (클러스터 재구성에 대한 자세한 내용은, SC2.2 문서를 참조하십시오.)

```
 1. FM_STOP method is run.
 2. STOP_NET method is run.
 3. Logical interfaces for the logical host are brought offline.
      - ifconfig hme0:1 0.0.0.0 down
 4. STOP method is run.
 5. Disk groups and file systems are moved.
      a. Unmount logical host file systems.
      b. vxdg deport disk groups on one machine.
    - - Only the steps below are run if a machine crashes - -
      c. vxdg import disk groups on the other machine.
      d. fsck logical host file systems.
      e. Mount logical host file systems.
 6. START method is run.
 7. Logical interfaces for the logical host are brought online.
      - ifconfig hme0:1 <ip address> up
 8. START_NET method is run.
 9. FM_INIT method is run.
10. FM_START method is run.
```

제어 방법은 다음과 같은 명령행 인수로 수행됩니다.

```
METHOD <logical hosts being hosted> <logical hosts not being hosted> <time-out>
```

첫 번째 인수는 현재 주관하고 있는 논리 호스트를 쉼표로 구분한 목록이며, 두 번째는 주관하고 있지 않는 논리 호스트를 쉼표로 구분한 목록입니다. 마지막 인수는 메소드의 시간종료로, SC2.2 소프트웨어가 메소드를 취소하기 전에 수행하도록 허용된 시간량입니다.

디스크 및 파일 시스템 구성

SC2.2는 두 개의 볼륨 관리 프로그램, Sun StorEdge Volume Manager(Veritas) 및 Solstice Disk Suite를 지원합니다. 둘다 잘 작업하지만, StorEdge Volume Manager는 클러스터된 환경에서 이점을 갖습니다. 어떤 클러스터 구성에서, 디스크 격납장치의 제어기 번호는 클러스터에서 각 기계에 대해 다를 수 있습니다. 제어기 번호가 다른 경우, 제어기의 디스크 장치에 대한 경로는 다릅니다. Disk Suite는 디스크 장치 경로로 직접 작업하므로, 이 상황에서 잘 작업하지 않습니다. StorEdge Volume Manager는 제어기 번호와 무관하게 디스크 자체로 작업하며, 제어기 번호가 다른 경우 영향받지 않습니다.

HA의 목표는 데이터 서비스의 사용 가능성을 증가시키는 것으로, 모든 파일 시스템 및 디스크 장치가 이중복사되는지 또는 RAID 구성으로 되어 있는지 확인하는 것이 중요합니다. 이것은 실패한 디스크로 인한 장애 복구를 방지하며 클러스터의 안정성을 증가시킵니다.

HA-NFS

DB2 UDB EEE는 인스턴스가 여러 머신에서 구성될 때 공유된 파일 시스템을 요구합니다. 일반적인 DB2 UDB EEE 구성은 NFS로부터 하나의 머신에서 내보낸 홈 디렉토리를 가지며, EEE 인스턴스에 참여하는 모든 머신에 마운트됩니다. 상호 인계 구성의 경우, DB2 UDB EEE는 HA-NFS에 의해 좌우되어 공유된 고가용 파일 시스템을 제공합니다. 논리 호스트 중 하나는 HA-NFS를 통해 파일 시스템을 내보내며, 클러스터 내의 각 머신은 EEE 인스턴스의 홈 디렉토로서 파일 시스템을 마운트합니다. HA-NFS에 대해서는 Sun Cluster 문서에서 자세한 내용을 참조하십시오.

콘솔 및 ctelnet 유틸리티

SC2.2와 함께 제공되는 두가지 유용한 유틸리티는 `cconsole` 및 `ctelnet`입니다. 이 유틸리티는 클러스터에서 동시에 여러 머신에 단일 명령을 발행하는데 사용될

수 있습니다. 이 유틸리티로 구성 파일을 편집하면 클러스터에 있는 모든 머신에서 동일하도록 보장합니다. 또한 이 유틸리티는 각 머신에서 동일한 방법으로 소프트웨어를 설치하기 위해 사용될 수 있습니다. 이 유틸리티에 대해서는 Sun Cluster 문서에서 자세한 내용을 참조하십시오.

캠퍼스 클러스터링 및 대륙 클러스터링

클러스터의 머신이 동일한 건물에 없을 때의 클러스터를 *캠퍼스 클러스터*라고 합니다. 캠퍼스 클러스터는 단일 실패 지점으로서 건물 자체를 제거하는데 유용합니다. 예를 들어, 클러스터에 있는 머신이 모두 동일한 건물에 있으며 전소한 경우, 전체 클러스터가 영향을 받습니다. 그러나 머신이 다른 건물에 있으며 건물 중 하나가 전소하면 클러스터는 잔존합니다.

*대륙 클러스터*는 다른 도시간에 분배된 클러스터입니다. 이 경우, 목표는 단일 실패 지점으로서 지리학적 지역을 제거하는 것입니다. 이러한 유형의 클러스터는 지진과 해일과 같은 큰 재해로부터 보호합니다.

현재, Sun Cluster는 10km 또는 6 마일 가량 떨어진 머신을 지원할 수 있습니다. 이것은 두 개의 다른 사이트간의 고속 연결이 필요한 사람들에게 캠퍼스 클러스터링을 실행 가능한 옵션으로 만듭니다. 클러스터는 두 개의 개별 상호 연결과 몇 개의 공유 디스크용 광학 케이블을 요구합니다. 두 개의 사이트 간의 고속 연결의 비용은 이점을 상쇄시킬 수 있습니다.

일반적인 문제점

SC2.2 소프트웨어는 클러스터 구성 데이터베이스 또는 CCD(4)를 사용하여 클러스터 구성의 단일 클러스터에 걸친 저장소를 제공합니다. CCD는 개별 API를 가지며 `/etc/opt/SUNWcluster/conf` 디렉토리 아래에 저장됩니다. 드물게, CCD는 동시성을 벗어날 수 있으며, 정정될 필요가 있을 수 있습니다. 이 상황에서 CCD를 정정하는 가장 좋은 방법은 백업 사본에서 복원하는 것입니다.

CCD를 백업하려면, 클러스터 내의 모든 머신에서 클러스터 소프트웨어를 종료하고, `/etc/opt/SUNWcluster/conf` 디렉토리를 "tar"하고 tar 파일을 안전한 장소에 저장하십시오. 백업이 작성될 때 클러스터 소프트웨어가 종료되지 않으면, CCD를 복원시 문제점이 발생할 수 있습니다. 클러스터 구성이 변경되면 언제라도 새로 백업하여 최신 백업을 유지하십시오. CCD를 복원하려면, 클러스터 내의 모든

머신에서 클러스터 소프트웨어를 종료하고, conf 디렉토리를 conf.old로 이동하고, 백업 사본을 "untar"하십시오. 그러면 클러스터는 새로운 CCD로 시작될 수 있습니다.

DB2 고려사항

이 절에서는 다음 주제를 다룹니다.

- 『HA 인스턴스에 연결하는 응용프로그램』
- 311 페이지의 『EE 및 EEE 인스턴스에 대한 디스크 레이아웃』
- 312 페이지의 『EE 및 EEE 인스턴스에 대한 홈 디렉토리 레이아웃』
- 314 페이지의 『논리 호스트 및 DB2 UDB EEE』
- 315 페이지의 『DB2 설치 위치 및 옵션』
- 316 페이지의 『데이터베이스 및 데이터베이스 관리 프로그램 구성 매개변수』
- 316 페이지의 『시스템 손상 복구』
- 316 페이지의 『복제를 통한 고가용성』
- 316 페이지의 『Sun Cluster 2.2에서의 DB2 Connect 전제조건』

HA 인스턴스에 연결하는 응용프로그램

매우 사용 가능한 DB2 인스턴스에 좌우되는 응용프로그램은 장애 복구 이벤트에서 재연결할 수 있어야 합니다. 호스트 이름 및 논리 호스트의 IP 주소는 동일하게 남아 있으므로, 다른 호스트 이름에 연결하거나 데이터베이스를 다시 카탈로그할 필요가 없습니다.

두 개의 머신과 하나의 DB2 Universal Database EE(Enterprise Edition) 인스턴스가 있는 클러스터를 고려하십시오. EE 인스턴스는 클러스터 내의 머신 중 하나에 보통 상주합니다. HA 인스턴스의 클라이언트는 HA 인스턴스와 연관된 논리 호스트의 논리 IP 주소(또는 호스트 이름)에 연결합니다.

HA 클라이언트에 따라, 두 개 유형의 장애 복구가 있습니다. HA 인스턴스를 주관하는 머신이 파손되면 하나의 유형이 발생합니다. 적절하게 종료할 기회가 HA 인스턴스에 제공되면 다른 유형이 발생합니다.

머신이 파손되고 HA 인스턴스를 다운시키는 경우, 데이터베이스에 대한 기존 연결 및 새 연결 둘다 정지합니다. 클라이언트가 데이터베이스에 대해 사용 중이었던 IP 주소가 있는 네트워크에는 머신이 없으므로 연결이 정지합니다. 데이터베이스가 적절하게 종료하는 경우, `db2stop force`는 데이터베이스에 대한 기존 연결을 끊고 오류 메시지가 리턴됩니다.

장애 복구 중, 데이터베이스와 연관된 논리 IP 주소는 오프라인되는데, 그 이유는 SC2.2 소프트웨어가 그 논리 주소를 오프라인으로 하거나 논리 호스트가 있는 머신이 파손되었기 때문입니다. 여기에서 데이터베이스에 대한 임의의 새로운 연결은 짧은 기간 동안 정지합니다.

데이터베이스와 연관된 논리 IP 주소는 결국 DB2가 시작되기 전에 다른 머신에서 제공됩니다. 이 단계에서, 데이터베이스에 대한 연결은 정지되지 않지만, DB2가 아직 시스템에서 시작되지 않았으므로 통신 오류를 받습니다. 데이터베이스에 여전히 연결되어 있는 DB2 클라이언트는 통신 오류 받기를 시작합니다. DB2 클라이언트가 연결되어 있다고 여전히 믿기는 하지만, 논리 IP 주소를 주관하기 시작한 머신은 기존 연결에 대해 알지 못합니다. 연결은 단순히 재설정되며, DB2 클라이언트는 통신 오류를 받습니다. 짧은 시간 이후에 DB2는 머신에서 시작되며, 데이터베이스에 대한 성공적인 연결이 이루어질 수 있습니다. 여기에서 데이터베이스는 불일치할 수 있으며, 클라이언트는 데이터베이스를 복구하기를 기다려야 합니다.

HA 환경에 대한 응용프로그램을 설계할 때, 데이터베이스 연결이 정지한 단계에 대해 특수 코드를 작성할 필요가 없습니다. Sun Cluster 소프트웨어가 논리 IP 주소를 이동시키는 짧은 기간 동안에만 연결이 정지됩니다. Sun Cluster에서 수행 중인 모든 데이터 서비스는 이 단계 중 동일한 정지 연결을 경험합니다. 데이터베이스가 어떻게 다운되는가에 관계없이 클라이언트는 오류 메시지를 받고, 성공할 때까지 다시 연결을 시도해야 합니다. 클라이언트의 시각에서 HA 인스턴스는 마치 다운되어 동일한 머신에서 백업된 것과 같습니다. 제어된 장애 복구에서 HA 인스턴스는 강제로 해제된 클라이언트에 나타나며, 나중에 동일한 머신에서 데이터베이스에 다시 연결할 수 있습니다. 제어되지 않은 장애 복구에서 HA 인스턴스는 데이터베이스 서버가 파손된 클라이언트에 나타나며 동일한 머신에서 바로 백업되었습니다.

EE 및 EEE 인스턴스에 대한 디스크 레이아웃

DB2는 클러스터 내의 각 머신에 동일하게 나타나도록 요구하는 디스크 장치나 파일 시스템을 기대합니다. 이를 보장하려면, 필수 디스크나 파일 시스템이 HA 인스턴스와 연관된 논리 호스트에 따르는 방식으로 구성되어야 하며, 클러스터 내의 각 머신에서 동일한 경로 이름을 갖습니다.

DMS 및 SMS 테이블 공간 모두는 HA 환경에서 지원됩니다. DMS 테이블 공간의 장치 컨테이너는 이중복사되거나 RAID 구성 중 하나인 볼륨 관리 프로그램에 의해 작성된 원시 장치를 사용해야 합니다. `/dev/rdsk/c20t0d0s0`과 같은 일반 디스크 장치는 다음과 이유로 사용하지 말아야 합니다.

- 일반 디스크 장치는 둘 이상의 머신에서 동시에 작성될 수 있는 가능성을 증가시킵니다.
- 제어기 번호는 다른 머신에서 다를 수 있습니다.

DB2가 이 상황에서 장애 복구되면, 필요한 디스크 장치는 다른 머신에서 수행한 것과 동일하지 않으며 시작하지 않습니다. DMS 테이블 공간의 파일 컨테이너 및 SMS 테이블 공간의 컨테이너는 마운트된 파일 시스템에 상주해야 합니다. 논리 호스트의 파일 시스템은 논리 호스트의 `vfstab` 파일에 포함되었을 때 자동으로 마운트됩니다.

논리 호스트의 `vfstab` 파일은 다음 경로에 있습니다.

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
```

여기서 *logical_host*는 `vfstab` 파일과 연관되는 논리 호스트의 이름입니다.

각 논리 호스트는 자신의 `vfstab` 파일을 가지며, 논리 호스트의 디스크 그룹이 현재 머신으로 전송된 후, HA 서비스가 시작되기 전에 마운트되는 파일 시스템이 들어 있습니다. Sun Cluster 소프트웨어는 파일 시스템의 건재를 확인하기 위해 **fsck**(파일 시스템 점검)을 수행한 후에 적절하게 정의된 모든 파일 시스템을 마운트하려고 시도합니다. **fsck**가 실패하면, 파일 시스템은 마운트되지 않으며, 오류 메시지는 무시됩니다.

주: 프로세스가 열린 파일을 갖거나, 현재 작업 디렉토리가 마운트 지점 아래에 있으면, 마운트에 실패합니다. 이를 방지하기 위해, 어떠한 프로세스도 논리 호스트의 vfstab 파일에 포함된 마운트 지점 아래에 남아 있지 않도록 확인하십시오.

모든 규약은 SMS 테이블 공간 사용시 EEE 인스턴스의 파일 시스템 레이아웃에 사용될 수 있습니다. 다음은 hadb2_setup 유틸리티에서 사용되는 규약입니다.

```
scadmin@crackle(190)# pwd
/export/ha_home/db2eee/db2eee
scadmin@crackle(191)# ls -l
total 18
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0000 -> /log0/disks/db2eee/NODE0000
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0001 -> /log0/disks/db2eee/NODE0001
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0002 -> /log0/disks/db2eee/NODE0002
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0003 -> /log0/disks/db2eee/NODE0003
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0004 -> /log0/disks/db2eee/NODE0004
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0005 -> /log1/disks/db2eee/NODE0005
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0006 -> /log1/disks/db2eee/NODE0006
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0007 -> /log1/disks/db2eee/NODE0007
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0008 -> /log1/disks/db2eee/NODE0008
scadmin@crackle(192)#
```

인스턴스 소유자는 db2eee이며, db2eee 인스턴스의 기본 데이터베이스 디렉토리는 /export/ha_home/db2eee입니다. 논리 호스트 log0은 데이터베이스 파티션 0, 1, 2 및 3을 주관하며, 반면 논리 호스트 log1은 데이터베이스 파티션 4, 5, 6, 7 및 8을 주관합니다.

각 데이터베이스 파티션의 경우, 해당하는 NODE*xxxx* 디렉토리가 있습니다. 데이터베이스 파티션에 대한 노드 디렉토리는 연관된 논리 호스트 파일 시스템 아래의 디렉토리를 지시합니다.

경로 규약을 선택할 때, 다음을 확인하십시오.

1. 파일 시스템의 디스크는 디스크를 필요로 하는 데이터베이스 파티션에 관여하는 논리 호스트의 디스크 그룹에 있습니다.
2. 컨테이너를 보유하는 파일 시스템은 논리 호스트의 vfstab 파일을 통해 마운트됩니다.

EE 및 EEE 인스턴스에 대한 홈 디렉토리 레이아웃

EE 인스턴스의 경우, 홈 디렉토리는 논리 호스트의 vfstab 파일에 정의된 파일 시스템이어야 합니다. 이 디렉토리는 DB2가 시작되기 전에 사용 가능하며, 논리

호스트가 클러스터에서 이동되는 곳마다 DB2로 전송됩니다. 각 머신은 vfstab 파일의 자체 사본을 가지며, 각 머신에 동일한 내용이 있는지 확인하려면 주의해야 합니다. 다음은 EE 인스턴스의 홈 디렉토리 예입니다.

```
/log0/home/db2ee
```

여기서, /log0은 논리 호스트 log0의 논리 호스트 파일 시스템이며, db2ee는 DB2 인스턴스의 이름입니다. 이 홈 디렉토리 경로는 "db2ee" 인스턴스를 주관할 수 있는 클러스터 내의 각 머신에 있는 /etc/passwd 파일에 있어야 합니다.

EEE 인스턴스의 경우, 홈 디렉토리를 설정하는 두 가지 방법이 있습니다. 긴급 대기 구성의 경우, 홈 디렉토리는 EE 인스턴스에서와 동일한 방식으로 설정될 수 있습니다. 상호 인계 구성의 경우, HA-NFS는 홈 디렉토리용으로 사용되어야 하며, EEE 인스턴스를 설정하기 *전에* 적절하게 구성되어야 합니다.

클러스터 내의 머신 중 하나는 선택된 논리 호스트용 dfstab 파일을 사용하여 EEE 인스턴스용 파일 시스템을 내보내야 합니다. dfstab 파일에는 머신이 논리 호스트를 주관할 때 NFS를 통해 내보내야 하는 파일 시스템이 들어 있습니다. 각 머신은 dfstab 파일의 자체 사본을 가지며, 각 머신에 동일한 내용이 있는지 확인하려면 주의해야 합니다.

HA-NFS 파일 시스템에 대한 정보는 hadb2tab 파일에 위치됩니다(hadb2_setup 프로그램을 통해). HA 에이전트가 인스턴스에 대한 정보를 읽을 때, 자동으로 인스턴스에 대한 HA-NFS 파일 시스템을 마운트합니다(318 페이지의 『hadb2tab 파일』 참조).

HA-NFS 파일 시스템의 마운트 지점은 보통 /export/ha_home입니다. 클러스터 내의 각 머신에서, 이것은 HA-NFS 디렉토리를 내보내는 논리 호스트에서 마운트된 NFS입니다. EEE 인스턴스 소유자의 홈 디렉토리는 이 디렉토리 아래에 위치하며 다음과 같습니다.

```
/export/ha_home/<instance>
```

여기서, *instance*는 인스턴스 소유자의 이름입니다.

인스턴스를 마운트하거나 마운트해제하는 것을 피하기 위해 각 머신에서 인스턴스에 대한 홈 디렉토리를 가질 수 있습니다. 이것은 홈 디렉토리가 각 머신에서 동

일하게 남아 있음을 보장하는 특수한 관리 오버헤드를 요구합니다. 그렇지 못한 경우 DB2가 제대로 시작하지 못하거나, 다른 구성으로 DB2를 시작하게 할 수 있습니다. 이것은 지원되는 구성이 *아닙니다*.

논리 호스트 및 DB2 UDB EEE

논리 호스트는 보통 하나 이상의 데이터베이스 파티션을 주관하도록 선택되며, HA-NFS 파일 시스템을 내보냅니다. 예를 들어, 클러스터에 네 개의 데이터베이스 파티션과 두 개의 머신이 있는 경우, 각 머신에 대해 하나의 논리 호스트가 있어야 합니다(315 페이지의 그림33). 하나의 논리 호스트는 두 개의 데이터베이스 파티션을 주관하며 HA-NFS 파일 시스템을 내보내고, 다른 논리 호스트는 나머지 두 개의 데이터베이스 파티션을 주관할 수 있습니다.

기본값으로, DB2 UDB EEE 인스턴스는 이미 하나 이상의 활동 데이터베이스 파티션을 가진 머신에 최대 두 개의 데이터베이스 파티션을 정상적으로 추가하기 위해 충분한 자원을 할당합니다. 예를 들어, 클러스터에서 단일 인스턴스에 대해 네 개의 데이터베이스 파티션이 있는 경우, 이는 논리 호스트 당 하나의 데이터베이스 파티션이 있는 경우에만 관계되거나, 하나의 논리 호스트가 세 개의 데이터베이스 파티션을 주관합니다. 어느 경우나 동일한 인스턴스에 대해 데이터베이스 파티션을 이미 주관하는 머신에 대해 세 개의 데이터베이스 파티션 장애 복구를 가지는 것이 가능합니다.

DB2_NUM_FAILOVER_NODES 레지스트리 변수는 장애 복구된 데이터베이스 파티션용으로 예약된 자원량을 증가시키기 위해 사용할 수 있습니다.

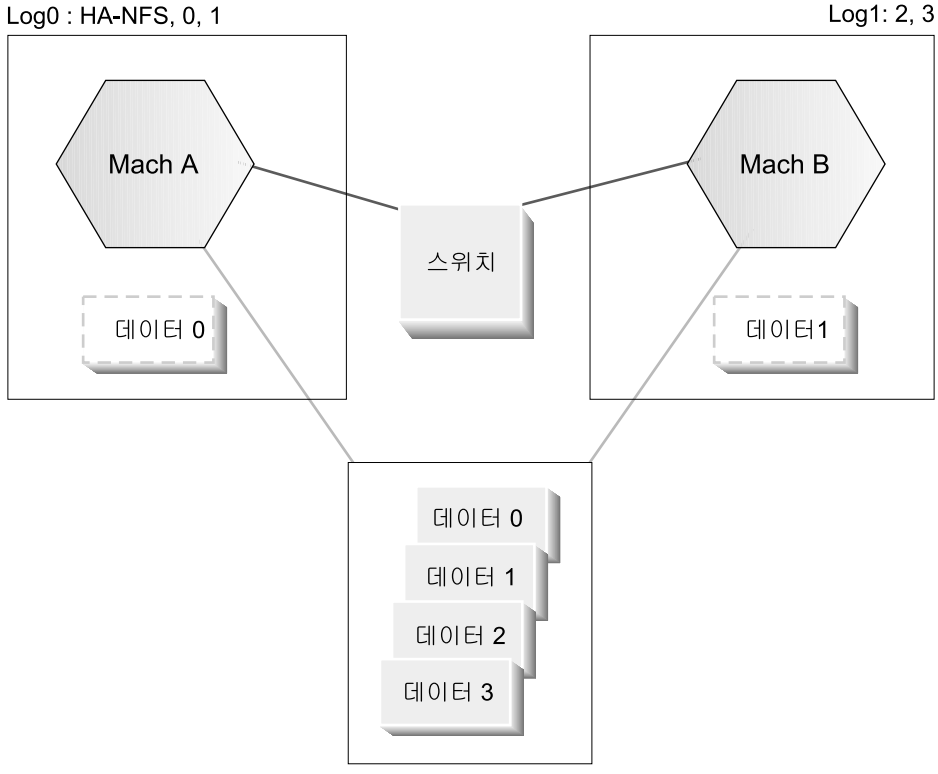


그림 33. 각 머신의 하나의 논리 호스트

DB2 설치 위치 및 옵션

DB2가 설치된 파일 시스템은 이중복사되어야 하거나, 최소한 RAID 구성으로 되어 있어야 합니다. DB2가 일반 디스크에 설치된 경우, 디스크 실패가 더 잘 발생하며, 그 결과 장애 복구는 예방가능한 것으로 고려되고 클러스터의 안정성을 감소시킵니다.

DB2는 논리 호스트의 디스크 그룹 내의 그룹에 설치될 수 없습니다. 왜냐하면 HA 에이전트는 항상 DB2 라이브러리에 액세스하는 것을 필요로 하기 때문입니다. HA 에이전트가 DB2 라이브러리에 대한 액세스를 갖지 않은 경우, 실패합니다. DB2는 클러스터 내의 각 머신에 정상적으로 설치되어야 합니다.

데이터베이스 및 데이터베이스 관리 프로그램 구성 매개변수

데이터베이스 관리 프로그램 구성 매개변수는 장애 복구 이후에 DB2가 시작되기 전에 pre_db2start 스크립트를 사용하여 변경할 수 있습니다(320 페이지의 『사용자 스크립트』 참조). 이 실행 스크립트가 있다면 인스턴스 소유자의 홈 디렉토리의 sqllib/ha 디렉토리 아래에서 수행됩니다. 이름이 제시하는 것처럼, **db2start** 이전에만 수행됩니다. 제어 방법에 전달되는 동일한 인수는 인스턴스가 EEE 인스턴스인 경우를 제외하고는 pre_db2start 스크립트로 전달됩니다. EEE 인스턴스의 경우, pre_db2start 스크립트에 **db2start** 명령에 대한 노드 번호가 전달됩니다.

시스템 손상 복구

HA 환경에서의 파손 복구는 일반 환경에서와 같습니다. HA 인스턴스가 파손된 하나의 머신에서 다른 머신으로 HA 인스턴스를 가져오더라도 인스턴스의 파일 및 디스크 장치는 동일하게 보이며, 데이터베이스를 복구하는데 필요한 조치는 다르지 않습니다. 응급 복구 및 기타 데이터베이스 복구 양식에 대한 자세한 정보는 3 페이지의 『제1장 좋은 백업 및 복구 전략 개발』을 참조하십시오.

데이터베이스는 수동으로 재시작될 수 있지만(또는 사용자 스크립트 중 하나를 통해), 특히 EEE 인스턴스의 경우 *autorestart* 데이터베이스 구성 매개변수를 ON으로 설정하는 것이 좋습니다. 이것은 데이터베이스가 불일치 상태에 있는 시간량을 최소화합니다.

복제를 통한 고가용성

데이터 사용 가능성은 복제를 통해 확장될 수 있습니다. 두 개의 서버 간에 데이터를 복제하여, 고가용성의 양식을 이룰 수 있습니다. 서버 중 하나가 종료되면, 다른 서버가 인수하여 데이터 서비스를 계속 제공할 수 있어야 합니다.

그러나 복제는 비동기적으로 수행되므로, 일부 변경사항은 해당 서버가 종료될 때 다른 서버로 전달될 수 없습니다.

Sun Cluster 2.2에서의 DB2 Connect 전제조건

DB2 Connect는 다음과 같은 경우에 Sun Cluster 2.2에서 지원됩니다.

- 호스트에 대한 프로토콜이 SNA가 아닌 TCP/IP입니다.

- 2단계 확약이 사용되지 않습니다. 이 제한사항은 공유 디스크에 SPM 로그를 위치시키고(이는 *spm_log_path* 데이터베이스 관리 프로그램 구성 매개변수를 통해 수행할 수 있음) 장애 복구 머신의 구성이 TCP/IP 구성과 같도록(같은 호스트 이름, IP 주소 등) 사용자가 구성할 경우에 어느 정도 완화됩니다.

DB2 고가용성 에이전트

DB2 고가용성 에이전트는 DB2 및 SC2.x 간의 중개자와 같이 역할합니다. DB2를 잘 알지 않고도 클러스터된 환경에서 Sun Cluster 2.2 소프트웨어가 DB2를 제어하게 하는 방식을 제공합니다. EE 및 EEE 인스턴스 둘다에 대한 하나의 에이전트가 있습니다. 에이전트는 관리 인스턴스 및 데이터베이스 인스턴스 둘다를 지원합니다.

hadb2 서비스 레지스터

SC2.2로 작업하려면, DB2 HA 에이전트가 등록되어야 합니다. 데이터 서비스를 등록하면 어떤 제어 방법이 사용가능하고 어떤 디렉토리에 상주하고 있는지 SC2.2에 알려줍니다. HA 에이전트와 함께 제공되는 hadb2_reg라고 하는 특수 스크립트는 EE 및 EEE 인스턴스 둘다에 대해 hadb2 서비스를 등록할 수 있습니다. hadb2_reg 스크립트는 전체 클러스터에 대해서 한 번만 수행하면 됩니다.

DB2 HA 에이전트에 대한 오직 하나의 제어 방법 세트가 있지만, 등록되는 방식은 EEE 인스턴스가 상호 인계 구성에서 사용되는지 여부에 의해 좌우됩니다. 긴급 대기 구성에서 EE 인스턴스 또는 EEE 인스턴스의 경우, HA-NFS는 사용되지 않습니다. 그러므로 hadb2 서비스가 HA-NFS에 의해 좌우됨을 SC2.2 소프트웨어에 알리는 "-d nfs" 스위치가 필요하지 않습니다.

hadb2_reg가 EEE 인스턴스에 대해 DB2 V7.1 제어 메소드를 등록하기 위해 사용하는 실제 명령은 다음과 같습니다.

```
hareg -r hadb2 -b /opt/IBMdb2/V7.1/ha -m
START=hadb2_start,START_NET=hadb2_startnet,STOP_NET=hadb2_stopnet,
FM_START=hadb2_fmstart,FM_STOP=hadb2_fmstop
-t START_NET=$TIMEOUT,STOP_NET=$TIMEOUT -d nfs
```

-b 스위치는 모든 제어 방법에 대한 opt/IBMdb2/V7.1/ha 디렉토리를 조사하도록 SC2.x에 알립니다. -m 스위치는 hadb2 서비스에 대한 실제 제어 방법을 정의

합니다. -t 스위치는 START_NET 및 STOP_NET 제어 방법에 대한 시간종료를 정의합니다. 각 제어 방법에 대한 자세한 설명은 Sun Cluster 문서를 참조하십시오.

hadb2_unreg 스크립트는 hadb2 서비스를 등록해제하기 위해 사용할 수 있으며, hadb2_reg와 같이 클러스터에 대해 한 번만 수행해야 합니다.

hadb2tab 파일

hadb2tab 파일은 DB2 HA 에이전트에 대한 기본 구성 파일입니다. 각 제어 방법은 어떤 인스턴스의 사용 가능성이 높은지 알아보기 위해 이 파일을 조사합니다. hadb2tab 파일은 DB2 UDB 버전 7.1의 /var/db2/v71/ 디렉토리 아래에 있습니다. 파일은 다중 인스턴스를 지원하며, 주석화되지 않은 각 라인은 다른 HA 인스턴스를 나타냅니다. 다음은 hadb2tab 파일의 예입니다.

```
<scadmin@thrash(203)# cat hadb2tab
EEE DATA db2eee jolt ON /export/ha_home /log0/home #Added by DB2 HA software
EE ADMIN db2ee log1 ON  -               -          #Added by DB2 HA software
```

첫 번째 필드는 인스턴스가 EE 인스턴스인지 아니면 EEE 인스턴스인지 여부를 DB2 HA 에이전트에 표시합니다. 두 번째 필드는 인스턴스가 데이터 인스턴스인지 아니면 관리 인스턴스인지 여부를 나타냅니다. 세 번째 필드에는 HA 인스턴스의 사용자 이름이 들어 있습니다. 네 번째 필드는 인스턴스가 EE 또는 EEE 인스턴스인지 여부에 따라 논리 호스트이거나 인스턴스의 HA-NFS 호스트입니다. 다섯 번째 필드는 인스턴스의 결함 모니터링이 설정되거나 해제되었는지 여부를 나타냅니다. 마지막 두 개의 필드는 각각 지역 마운트 지점과, 원격 HA-NFS 디렉토리입니다. 이 필드들은 사용하지 않는 경우 -(하이픈)으로 설정되어야 하며 EEE 상호 인계 구성에서만 사용되어야 합니다. "#" 표시문자 앞의 라인에 있는 정보가 0의 길이이거나 인스턴스의 유효한 정의인 경우 hadb2tab 파일에서 주석이 허용됩니다.

제어 방법

SC2.2 에이전트의 제어 방법은 스크립트 또는 프로그램 세트일 수 있습니다. Solaris에서의 DB2에 대한 에이전트는 다음과 같은 메소드를 포함하는 프로그램 세트입니다.

START_NET

hadb2_startnet: DB2를 시작하기 위해 사용됩니다.

STOP_NET

hadb2_stopnet: DB2를 중단하기 위해 사용됩니다.

FM_START

hadb2_fmstart: DB2에 대한 결함 모니터를 시작하기 위해 사용됩니다.

FM_STOP

hadb2_fmstop: DB2에 대한 결함 모니터를 중지하기 위해 사용됩니다.

이 제어 방법에 대해서는 Sun Cluster 문서에서 자세한 내용을 참조하십시오.

EE 인스턴스의 경우, 인스턴스와 연관된 논리 호스트는 hadb2tab 파일에서 오른쪽에 정의됩니다. 그러나 EEE 인스턴스의 경우, 제어 방법은 다음과 같아야 합니다.

```
~<instance>/sqllib/ha/hadb2-eee.cfg
```

여기서, ~<instance>는 인스턴스 소유자의 홈 디렉토리입니다. 이 파일에는 각 데이터베이스 파티션에 대해 한 행이 들어 있으며, 데이터베이스 파티션을 논리 호스트와 연관시키는데 사용됩니다. 유효한 hadb2-eee.cfg 파일의 예는 다음과 같습니다.

```
crackle % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

인스턴스 또는 데이터베이스 파티션은 클러스터 주위의 해당 논리 호스트를 따릅니다. 논리 호스트는 주요 하드웨어 및 SC2.2에 의해 지원되는 클러스터에서 임의의 기계로 이동할 수 있습니다. 구성이 제대로 설정된 경우, DB2는 SC2.2 소프트웨어가 지원하는 모든 토폴로지를 지원합니다.

인스턴스의 모든 정보를 읽은 후에 제어 방법은 어떤 논리 호스트가 인스턴스와 연관되어 있는지 압니다, 명령행 인수를 분석한 후에, 제어 방법은 어떤 논리 호스트가 주관되며, 현재 머신에서 주관되지 않는 것은 어떤 것인지 압니다.

다음 테이블은 어떤 메소드가 수행 중이며, 데이터베이스 파티션이나 인스턴스와 연관되는 논리 호스트가 현재 머신에서 주관되는지 여부에 따라 취해지는 조치를 나타냅니다.

제어 방법	연관된 논리 호스트가 주관됨	연관된 논리 호스트가 주관되지 않음
START_NET	DB2 인스턴스 또는 데이터베이스 파티션 시작	조치 없음
STOP_NET	조치 없음	DB2 인스턴스 또는 데이터베이스 파티션 중단
FM_START	인스턴스의 결함 모니터 시작	조치 없음
FM_STOP	조치 없음	인스턴스의 결함 모니터링 중지

시작 조치를 수행하는 제어 방법은 현재 주관하고 있는 논리 호스트에만 관계되며, 중지 조치를 수행하는 제어 방법은 현재 주관하고 있지 않은 논리 호스트에만 관련됩니다.

또한 제어 방법은 HA-NFS가 사용 중인 경우 특수한 방식으로 HA-NFS 디렉토리를 마운트할 필요가 있습니다. HA-NFS의 지역 마운트 지점 및 디렉토리는 -(하이픈)으로 정의되지 않으며, 제어 방법은 지역 마운트 지점에서 `statvfs(2)`를 수행합니다. 지역 마운트 지점의 파일 시스템 유형이 `nfs`가 아닌 경우, 에이전트는 `hadb2tab` 라인의 정보를 사용하여 파일 시스템을 마운트하려고 시도합니다. 마운트 지점 및 HA-NFS의 디렉토리가 -(하이픈)으로서 정의되는 경우, 해당 논리 호스트의 `vfstab` 파일은 인스턴스의 홈 디렉토리가 들어 있는 파일 시스템을 마운트하는 것이 필요합니다. 지역 마운트 지점 및 HA-NFS의 원격 디렉토리는 EE 및 EEE 긴급 대기 구성의 -(하이픈)으로서 정의되어야 합니다.

사용자 스크립트

이 스크립트는 추가 기능을 추가하기 위해 제어 방법에서 수행되며, 제어 방법이 전달될 때 동일한 명령행 인수가 전달되며 시스템 관리자나 데이터베이스 관리자가 작성합니다.

프로그램이 백그라운드에서 수행하지 않는 스크립트 내에서 수행되어야 하는 경우, `nohup(1)`으로 프로그램을 백그라운드하는 것을 고려하십시오. `nohup` 프로그램은

SIGHUP(또는 정지) 신호로부터 실행 프로그램을 보호합니다. nohup 없이, 스크립트로부터 백그라운드에서 수행되는 프로그램은 스크립트가 완료될 때 SIGHUP 신호 결과로서 종료될 수 있습니다.

제어 방법은 다음 스크립트에서 수행합니다.

- /var/db2/v61/failover
- ~<instance>/sqllib/ha/pre_db2start
- ~<instance>/sqllib/ha/post_db2start
- ~<instance>%s/sqllib/ha/post_failover
- ~<instance>/sqllib/ha/pre_db2stop
- ~<instance>/sqllib/ha/fm_warning

여기서, ~*instance*는 HA 인트스턴스의 홈 디렉토리입니다.

fm_warning 스크립트를 제외하고, 각 사용자 스크립트는 호출한 제어 방법과 동일한 인수를 사용하여 수행합니다. EEE 인스턴스를 사용할 때, 데이터베이스 파티션 번호는 사용자 스크립트에 전달됩니다(마지막 인수와 같이).

/var/db2/v71/failover 스크립트는 START_NET 메소드 시작 시 호출되어 백그라운드에서 수행됩니다. 예를 들어, 그러한 스크립트는 장애 복구의 이벤트에서 지원 담당자에게 전자우편을 보내는데 사용될 수 있습니다. 다음은 장애 복구 스크립트의 예입니다.

```
#!/bin/ksh
# E-mail or page support staff to notify them that a failover has occurred.
echo "Failover occurred on machine 'hostname':Running $0!"
|/bin/mail admin@sphere.torolab.ibm.com
```

정상적으로 스크립트에서 전자우편을 보내려면, sendmail(1m)이 시스템에서 적절하게 구성되어야 합니다.

이름에서 제시하는 것처럼, pre_db2start 스크립트는 **db2start**가 호출되기 바로 전에 수행됩니다. 이 스크립트는 데이터베이스 관리 프로그램 구성 매개변수를 변경하는 것과 같은 타스크에 사용될 수 있습니다. 이것을 완료하려면 최대 20초가 걸립니다. EEE 인스턴스의 경우, 이 스크립트는 각 데이터베이스 파티션에서 **db2start**가 호출되기 전에 수행됩니다. 이 스크립트는 데이터 인스턴스에만 적용 가능하며, 관리 인스턴스에는 적용할 수 없습니다.

마찬가지로 post_db2start 스크립트는 **db2start**가 스크립트 호출된 *이후*에만 수행됩니다. 이 스크립트는 데이터베이스를 재시작하는 것과 같은 타스크에 사용될 수 있습니다. 실행 시간이 다른 인스턴스와 충돌하지 않도록 보장하기 위해 백그라운드에서 수행됩니다. 이 스크립트는 데이터 인스턴스에만 적용가능하며, 관리 인스턴스에는 적용할 수 없습니다.

인스턴스 소유자의 홈 디렉토리에 있는 post_failover 스크립트는 인스턴스를 처리한 이후에 수행됩니다. 이 스크립트는 DB2가 이제 기능적인 클라이언트 응용프로그램에 통지하기 위해 또는 관리자에게 상태 파일을 보내기 위해 사용될 수 있습니다. 실행 시간이 다른 HA 인스턴스에 대한 조치를 지연시키지 않도록 보장하기 위해 백그라운드에서 수행됩니다. 다음은 사후 장애 복구 스크립트의 예입니다.

```
#!/bin/ksh
 #
 # Send the status file to the administrato-r.
 mail admin@sphere.torolab.ibm.com </tmp/HA.info.db2eee
```

DB2 HA 에이전트의 START_NET 및 STOP_NET 메소드 둘다 각 인스턴스를 처리한 후에 상태 파일을 작성합니다. 상태 파일의 이름은 다음과 같습니다.

```
/tmp/HA.info.<instance>
```

여기서, *instance*는 인스턴스 소유자의 사용자 이름입니다. 상태 파일에는 제어 방법을 수행하기 위해 걸리는 시간은 물론, 인스턴스의 시작 및 중지 보고서가 들어 있습니다. 다음은 상태 파일의 예입니다.

```
scadmin@crackle(173)# cat /tmp/HA.info.db2eee
-----  Elapsed Time: 00:00:18           -----
-----  Elapsed Time: 00:00:00  (HA-NFS) -----
NODE     ACTION     RESULT     TRIES     RC
----     ------     ------     -----     --
   4     stop       success        3    1064
   5     stop       success        1    1064
   6     stop       success        2    1064
   7     stop       success        2    1064
   8     stop       success        1    1064
-----------------------------------------------
```

pre_db2stop 스크립트는 **db2stop**이 호출되기 바로 전에 수행됩니다. 이 스크립트는 DB2가 중지하려고 함을 클라이언트 응용프로그램에 통지하기 위해 사용될 수

있습니다. 이것을 완료하려면 최대 20초가 걸립니다. 이 스크립트는 데이터 인스턴스에만 적용가능하며, 관리 인스턴스에는 적용할 수 없습니다.

예상치못한 종료 때문에 DB2가 재시작될 때 결함 모니터가 사용자 스크립트를 수행합니다. 이 스크립트는 다음과 같이 호출됩니다.

```
~<instance>/sqllib/ha/fm_warning
```

fm_warning 스크립트는 DB2가 결함 모니터에 의해 재시작되었음을 시스템 관리자에게 통지하기 위해 사용할 수 있습니다. 시스템 관리자는 DB2가 예상치못하게 종료된 이유를 알아내고, 이런 상황이 다시 발생하지 않도록 적절한 조치를 취해야 합니다. fm_warning 스크립트는 백그라운드에서 수행됩니다.

기타 고려사항

HA 데이터 서비스는 해제되며, 중지 메소드만이 장애 복구 또는 클러스터 재구성 중 수행되며, 다른 메소드는 HA 데이터 서비스가 적절하게 등록되고 설정된 경우에만 수행됩니다.

클러스터 내의 각 머신은 응답할 수 있는 모든 데이터 서비스를 수행할 수 있는 충분한 권한을 가지고 있는지 확인하십시오. CPU 로드, 메모리, 스왑 및 커널 매개변수와 같은 자원은 클러스터가 생산되기 전에 고려되어야 합니다. 예를 들어, 클러스터 내의 머신이 두 개의 DB2 인스턴스를 수행할 필요가 있는 경우, 해당 머신의 커널 매개변수 요구사항은 각 인스턴스에 필요한 요구사항을 합한 것입니다.

결함 모니터

결함 모니터링이 설정되면, 결함 모니터는 클러스터 재구성이나 장애 복구 중 시작됩니다. DB2가 START_NET 스크립트에 의해 시작되지 않는 경우, 결함 모니터 자체가 DB2를 시작합니다. DB2가 시작하지 않은 경우, 또는 알 수 없는 이유로 종료된 경우 결함 모니터가 검출할 수 있습니다. 이러한 이유로 결함 모니터가 설정되었을 때 수동으로 DB2를 종료하지 않는 것이 중요합니다. 결함 모니터는 예상치못한 종료로서 이를 파악하며 DB2를 재시작합니다. 이런 현상이 너무 여러번 발생하면, 해당 논리 호스트를 장애 복구합니다.

결함 모니터링이 인스턴스에 대해 사용 가능한 경우, 수동으로 인스턴스를 시작하거나 중지하는 올바른 방법은 우선 결함 모니터링 및 hadb2 서비스를 해제하십시오. 이 조치는 둘다 -f 및 -s 스위치를 사용하여 **hadb2_setup** 명령을 통해 시작할 수 있습니다(330 페이지의 『hadb2_setup 명령』 참조).

주: 동일한 논리 호스트에 대해 둘 이상의 인스턴스를 사용하지 마십시오. 둘 이상의 인스턴스가 논리 호스트와 연관되면, 건재한 인스턴스가 그렇지 못한 인스턴스와 함께 장애 복구될 수 있습니다.

EEE 고려사항

어떤 데이터베이스 파티션이 논리적 호스트와 연관되어 있는지 결정할 때, 장애 복구되는 방식을 고려하는 것이 중요합니다. 325 페이지의 그림34에서 표시한 것처럼 두 개의 머신 간의 네 개의 데이터베이스 파티션이 있는 두 개의 머신 클러스터를 고려하십시오.

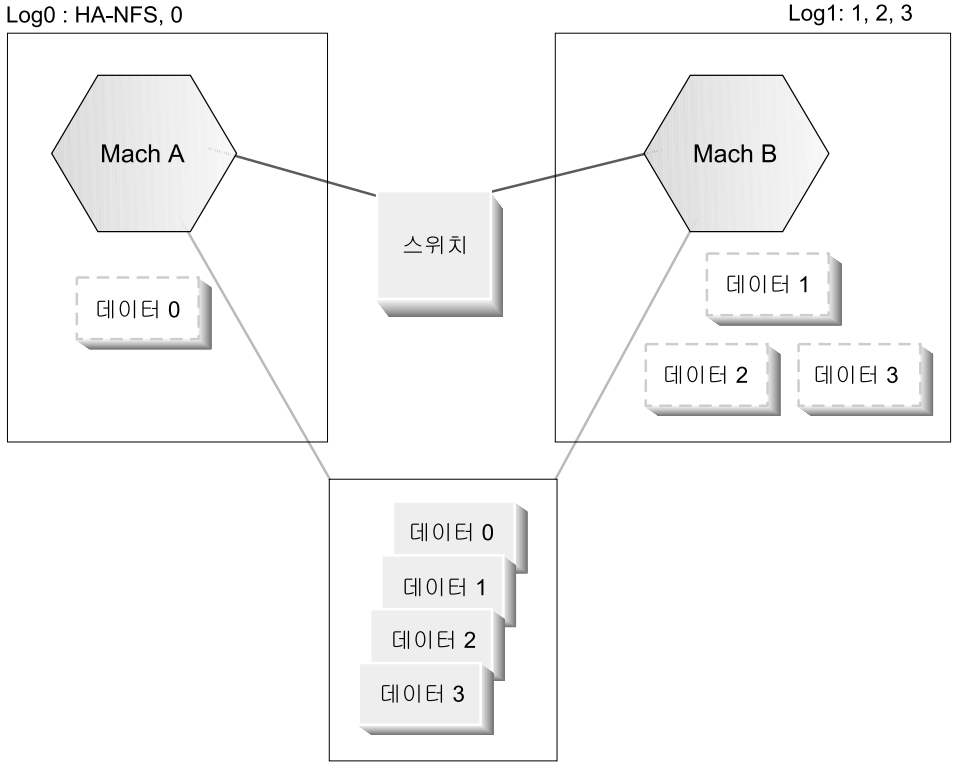


그림 34. 네 개의 데이터베이스 파티션이 있는 두 개의 머신 클러스터

각 데이터베이스 파티션을 하나의 논리 호스트로 연관시키고, HA-NFS에 대해 하나의 논리 호스트로 연관시킬 수 있습니다. 이 경우, 모든 논리 호스트가 하나의 시스템에 의해 주관되는 경우 문제점이 있을 수 있습니다. 해당 시스템이 실패하면, 모든 논리 호스트는 동시에 시스템에서 이동되어야 합니다. 불행하게도, Sun Cluster 소프트웨어는 예언할 수 있는 순서로 논리 호스트를 이동시키지 않으며, 연관된 데이터베이스 파티션을 가진 논리 호스트를 HA-NFS가 있는 논리 호스트 앞으로 이동시키는 것이 가능합니다. 단일 시스템에서 주관되는 것에 따라 데이터베이스 파티션을 함께 그룹화하는 것이 좋습니다. 이것은 보통 하나의 머신에 있는 두 개의 데이터베이스 파티션이 단일 논리 호스트와 연관되어야 함을 의미합니다.

EEE 인스턴스가 사용하는 `db2nodes.cfg` 파일은 데이터베이스 파티션이 거주하는 머신을 나타내기 위해 갱신됩니다. 예를 들어, 모든 데이터베이스 파티션은 "crackle"이라고 하는 머신에 있으며, `db2nodes.cfg` 파일은 다음과 유사합니다.

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 crackle 4 204.152.65.33
5 crackle 5 204.152.65.33
6 crackle 6 204.152.65.33
7 crackle 7 204.152.65.33
8 crackle 8 204.152.65.33
```

이 데이터베이스 파티션 중 일부가 ″thrash″라는 머신으로 이동되는 경우, db2nodes.cfg 파일은 다음과 같이 갱신됩니다.

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 thrash 0 204.152.65.34
5 thrash 1 204.152.65.34
6 thrash 2 204.152.65.34
7 thrash 3 204.152.65.34
8 thrash 4 204.152.65.34
```

호스트 이름 및 스위치 이름 둘다 머신 이름 ″thrash″를 반영하도록 변경되고, 포트 번호도 달라짐을 기억하십시오.

HA.config 파일

/etc/HA.config 파일이 있는 경우, 다음을 포함하여 몇 개의 구성 옵션이 들어 있을 수 있습니다.

```
scadmin@thrash(204)# cat /etc/HA.config
SYSLOG_FACILITY=LOG_LOCAL3
SYSLOG_LPRIORITY=LOG_INFO
SYSLOG_EPRIORITY=LOG_ERR
USE_INTERCONNECT=auto
SWITCH_NAME=204.152.65.18
DEBUG_LEVEL=2
FAILS_PER_HOUR=2
FAILS_PER_DAY=4
FAILS_PER_WEEK=10
```

```
FM_FAIL_SEV=soft
DB2START_TIMEOUT=60
DB2STOP_TIMEOUT=500
SCRIPT_USER=bin
```

주: HA.config 파일이 없는 경우, 기본값이 사용됩니다.

SYSLOG_FACILITY 변수는 메시지 및 오류 둘다를 기록하는 SYSLOG 기능을 설정합니다. SYSLOG_LPRIORITY 및 SYSLOG_EPRIORITY 변수는 각각 정보 메시지와 오류 메시지 로깅을 위해 SYSLOG 우선순위를 설정합니다.

DB2 HA 에이전트로부터의 정보를 기록하기 위해 SYSLOG 데몬을 사용하려면 변경이 필요할 수 있습니다. 예를 들어, /etc/syslog.conf 파일에 추가되는 다음 두 행 중 하나는 SYSLOG 데몬에 로그 파일에 대한 정보를 작성하도록 알립니다.

```
*.notice                                    /var/adm/SC.x
local3.info                                 /var/adm/SC.LOG_LOCAL3
```

Sun Cluster는 보통 고속 상호 연결을 갖습니다. DB2와의 고속 상호 연결을 사용하려면, USE_INTERCONNECT를 auto 또는 override로 설정하십시오. auto 설정(기본값)은 Sun 내부 논리 네트워크 인터페이스를 사용합니다. 이 인터페이스는 초기 인터페이스에 실패하면 다른 실제 인터페이스에 전송됩니다. USE_INTERCONNECT가 override로 설정하면, 스위치 이름은 SWITCH_NAME 변수에서 얻습니다. 또 다른 옵션은 USE_INTERCONNECT를 no로 설정하는 것으로, 이는 고속 상호연결이 사용되지 않도록 지정합니다.

DEBUG_LEVEL은 장애 복구 중 로깅되는 정보의 양을 지정합니다. 이것은 0과 10 사이의 숫자이며, 여기서 10은 최상의 디버그 레벨입니다. 정보는 지정된 SYSLOG 우선순위 및 기능에 기록됩니다. 문제점이 발생하면, 디버그 레벨을 최대 레벨로 설정하고, HA 에이전트에서 출력을 기록하도록 SYSLOG를 구성하고, SYSLOG 출력을 IBM 서비스로 보내십시오.

FAILS_PER_HOUR, FAILS_PER_DAY 및 FAILS_PER_WEEK의 세 변수는 DB2 결함 모니터가 논리 호스트를 장애 복구할 시기를 결정하는데 도움이 됩니다. 모든 HA 환경은 다르며, 얼마나 많은 DB2 실패를 허용할 수 있는지 결정해

야 합니다. 각 "수용가능한" 실패 이후에, DB2는 동일한 머신에서 재시작됩니다. 이 세가지 실패 임계값 중 하나가 초과되면, 인스턴스 및 데이터베이스 파티션과 연관된 논리 호스트는 장애 복구됩니다.

FM_FAIL_SEV 변수는 장애 복구가 "soft" 또는 "hard"인지 여부를 지정합니다. `hactl(1m)`의 Sun Cluster 문서에서 자세한 내용을 참조하십시오.

DB2START_TIMEOUT 및 DB2STOP_TIMEOUT 변수는 **db2start** 및 **db2stop** 수행을 허용하는 최대 시간(초)을 지정합니다. 지정된 간격이 지난 이후, HA 에이전트는 조작이 실패한 것으로 간주하고, 인스턴스를 재시작하려고 합니다.

임의의 특정 인스턴스와 연관되지 않은 일부 사용자 스크립트가 있습니다. 보통, 이 스크립트는 루트로서 수행되며, SCRIPT_USER 변수에 의해 겹쳐 쓸 수 있습니다. 스크립트를 수행할 수 있는 사용자 ID를 지정하도록 이 변수를 설정할 수 있습니다.

제어 방법이 DB2 명령을 수행하는 방식

DB2 HA 에이전트는 **su** 명령을 사용하여 인스턴스 소유자로서 명령을 수행합니다. 실제 명령은 다음과 유사합니다.

```
su - <instance> -c "db2stop"
```

여기서, *instance*는 인스턴스의 사용자 이름입니다.

인스턴스 소유자의 `.profile` 파일이 **su**-"friendly"인지 확인하는 것이 중요합니다. 그렇지 않은 경우, **su** 명령은 제대로 작업하지 않습니다. 수동으로 **su** 명령을 호출하거나, 스크립트에서 명령이 정상적으로 수행할 수 있는지 확인하십시오.

설정

이 절을 읽기 전에, SC2.2 소프트웨어에 익숙한지 확인하십시오. 이 절에서는 SC2.2 및 HA-NFS를 설정하는 방법을 알고, 사용자 볼륨 관리 프로그램을 사용하는 방법을 안다고 가정합니다. DB2의 다른 필수 패치와 함께, 다음 패치가 HA 에이전트에 필요합니다.

```
Solaris 2.6:
   105210-17 (or later)
   105786-05 (or later)
```

주: Solaris 7(Solaris 2.7)의 필수 패치가 없습니다.

일반적인 설치 단계

1. 클러스터 내의 모든 머신에 SC2.2를 설치하십시오. 설치 중, SC2.2는 어떤 에이전트를 설치할 지 묻습니다. DB2는 SC2.2와 함께 제공되지 않으므로, 에이전트 목록에 없습니다. DB2의 에이전트는 DB2와 함께 설치되며 **hadb2_reg** 명령을 통해 등록됩니다.
2. 디스크 그룹 및 논리 IP 주소로 논리 호스트를 구성하십시오.

DB2 UDB Enterprise Edition의 설정

1. 논리 호스트의 논리 호스트 파일 시스템 아래에 인스턴스의 홈 디렉토리를 작성하십시오.
2. 클러스터 내의 모든 머신에 DB2를 설치하십시오.
3. 현재 인스턴스에 대한 홈 디렉토리를 가진 클러스터 내의 머신에서 인스턴스를 작성하십시오.
4. 인스턴스의 사용자를 클러스터 내의 다른 머신에 추가하여, 숫자 사용자 ID가 동일한지 확인하십시오.
5. **hadb2_reg** 명령을 사용하여 `hadb2` 서비스를 등록하십시오.
6. **hadb2_setup** 명령을 수행하여 인스턴스의 HA를 설정하십시오.

DB2 UDB Enterprise - Extended Edition의 설정

1. HA 인스턴스 소유자의 홈 디렉토리를 작성하십시오.
   a. 긴급 대기의 경우, 논리 호스트의 논리 호스트 파일 시스템 아래에 인스턴스의 홈 디렉토리를 작성하십시오.
   b. 상호 인계의 경우, HA-NFS를 구성하고, 논리 호스트 중 하나에서 홈 디렉토리를 내보내십시오. 머신 중 하나에서, 선택한 마운트 포인트 아래에 HA-NFS 디렉토리를 마운트하십시오.
2. 클러스터 내의 모든 머신에 DB2를 설치하십시오.

3. HA-NFS 파일 시스템이 마운트된 머신에서 인스턴스를 작성하십시오.
4. 인스턴스의 사용자를 클러스터 내의 다른 머신에 추가하여, 숫자 사용자 ID가 동일한지 확인하십시오.
5. **hadb2_reg** 명령을 사용하여 hadb2 서비스를 등록하십시오.
6. **hadb2_setup** 명령을 수행하여 인스턴스의 HA를 설정하십시오.

주: HA 인스턴스의 정보를 정의하기 위해 NIS를 사용하는 것은 권장되지 않습니다. 왜냐하면 NIS가 실패의 단일 지점을 도입할 수 있기 때문입니다.

hadb2_setup 명령

hadb2_setup 명령은 DB2 HA 에이전트와 함께 제공되는 프로그램의 중심점입니다. 인스턴스를 설정하거나, 수정하거나 삭제하기 위해 사용될 수 있습니다. 또한 hadb2_setup 서비스를 설정하고 해제하기 위해서도 사용할 수 있습니다. 이 명령을 사용하여 hadb2tab 파일을 수동으로 편집할 필요는 없습니다.

주: **hadb2_setup** 명령은 수행하는 머신에서만 조치를 수행합니다. 하나의 머신에 작성된 변경사항은 클러스터 내의 다른 머신에 작성되어야 합니다.

다음 인수가 지원됩니다.

```
To add an EE instance:
----------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <logical_host> -p [DATA|ADMIN] -t EE
   For example:
      hadb2_setup -a -i db2ee -f off -h log1 -p DATA -t EE
To add an EEE instance:
----------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <nfs_host> -l <mount_point> \
      -r <ha-nfs_dir> -p [DATA|ADMIN] -t EEE -n "<node_info>"
   For example:
      hadb2_setup -a -i db2eee -f off -h ha-sun1 -l /export/ha_home \
         -r /log0/home -p DATA -t EEE -n "log0[0,10,20],log1[30,40,50]"
To delete an instance:
----------------------
   hadb2_setup -d -i <instance>
To modify an instance:
----------------------
   hadb2_setup -m -i <instance> [-f [on|off] | -l <mount_point> | \
      -h <host> | -p [DATA|ADMIN] | -r <ha-nfs_dir> | -t [EE|EEE] ]
Other options:
--------------
   -s   <on|off>             Bring hadb2 up or down (for all HA instances)
   -y                        Assume yes for safety checks
```

hadb2 서비스를 설정 또는 해제하려면, -s 스위치를 지정하십시오. 이것은 **hareg** 명령을 -n 및 -y 스위치와 함께 사용하고, hadb2 서비스를 지정하는 것과 같습니다. **hareg(1m)** 명령에 대해서는 Sun Cluster 문서에서 자세한 내용을 참조하십시오.

인스턴스의 결함 모니터는 -f 스위치를 사용하여 해제될 수 있습니다. 이것은 논리 기계에서 인스턴스에 대한 결함 모니터를 중지하는 것은 물론, hadb2tab 파일을 수정하여 결함 모니터링이 해제된다는 사실을 반영하는 효과를 갖습니다.

EE 인스턴스의 경우, 모든 머신에서 결함 모니터링을 해제하는 것은 인스턴스가 장애 복구하는 경우에 권장됩니다. EEE 인스턴스의 경우, 결함 모니터링은 수동으로 인스턴스를 종료하기 전에 인스턴스의 데이터베이스 파티션을 주관하는 모든 머신에서 해제되어야 합니다.

인스턴스를 삭제하려면, -d 스위치를 사용하십시오. 이것은 hadb2tab 파일에서 인스턴스를 제거하기만 하며, 다른 모든 파일이나 디렉토리는 제거하거나 수정하지 않습니다. hadb2tab 파일은 HA-DB2 에이전트의 기본 구성 파일이므로, 이 파일에서 인스턴스를 제거하면 제어 방법이 인스턴스의 존재를 인식하지 못합니다.

인스턴스를 수정하려면, -m 스위치를 사용하십시오. 이것은 hadb2tab 파일에서 정보를 변경하기만 하며, 다른 모든 파일이나 디렉토리는 제거하거나 수정하지 않습니다. -m 스위치는 hadb2tab 파일에 있는 정보에 부속한 모든 스위치와 함께 사용될 수 있습니다. db2nodes.cfg 파일 및 hadb2-eee.cfg 파일은 초기 설정 이후에 수동으로 변경해야 하는데, 이는 **hadb2_setup** 명령이 이러한 파일 수정을 지원하지 않기 때문입니다.

인스턴스 추가는 다소 더 복잡합니다.

EE 인스턴스의 경우, 다음 인수가 필요합니다.

```
hadb2_setup -a -i <instance> -f <fm> -h <logical_host> -t <EEE_or_EE>
   -p <purpose>
```

여기서 *instance*는 추가되는 인스턴스의 이름이며, *fm*은 결함 모니터링이 처음에 설정 또는 해제되는지 여부를 지정하고, *logical_host*는 연관된 논리 호스트입니다. *EEE_or_EE*는 EE로 설정되며, *purpose*는 DATA 또는 ADMIN 중 하나가 될 수 있습니다.

EEE 인스턴스의 경우, 다음 인수가 필요합니다.

```
hadb2_setup -a -i <instance> -f <fm> -h <nfs_host> -t <EEE_or_EE> -p
   <purpose> -l <mount_point> -r <HA-NFS_directory> -n <node_info>
```

여기서 *instance*는 추가되는 인스턴스의 이름이며, *fm*은 결함 모니터링이 처음에 설정되거나 해제되는지 여부를 지정하고, *nfs_host*는 HA-NFS 파일 시스템을 내보내는 논리 호스트의 호스트 이름입니다. *EEE_or_EE*는 EEE로 설정되며, *purpose*는 DATA 또는 ADMIN이 될 수 있고, *mount_point*는 HA-NFS 디렉토리의 지역 마운트 지점입니다. *HA-NFS_directory*는 HA-NFS 디렉토리이며, *node_info*는 데이터베이스 파티션을 논리 호스트와 연관시키는 정보입니다. 예를 들면, 다음과 같습니다.

```
hadb2_setup -a -i db2eee -f on -h jolt -l /export/ha_home -p DATA -t EEE -r
   /log1/home -n "log0[0,1],log1[2,3]"
```

EEE 인스턴스를 추가할 때, 노드 정보는 따옴표로 묶여야 합니다. 이 예에서, 인스턴스 ″db2eee″는 ″log0″ 및 ″log1″의 두 개의 논리 호스트와 연관됩니다. ″db2eee″ 인스턴스의 데이터베이스 파티션 ″0″ 및 ″1″은 논리 호스트 ″log0″와 연관되며, 데이터베이스 파티션 ″2″ 및 ″3″은 논리 호스트 ″log1″과 연관됩니다.

hadb2_setup 명령을 사용하여 클러스터 내의 모든 머신에 인스턴스를 추가하십시오. 그러면 인스턴스는 클러스터 재구성을 강제 수행하거나 hadb2 서비스를 해제한 후 설정하여 시작될 수 있습니다. 이는 **hareg** 명령을 통해서나, **hadb2_setup** 명령의 -s 스위치를 사용하여 수행할 수 있습니다. 인스턴스가 시작하지 않은 경우, 336 페이지의 『문제점 해결』을 참조하십시오.

hadb2_setup 명령이 EEE 인스턴스를 추가한 경우, 다음 조치가 투명하게 수행됩니다.

- 지정된 정보 점검. 이것은 사용자가 시스템에서 종료하며, HA-NFS가 수행 중인지 확인하는 것을 포함합니다.
- 'db2nodes.cfg 파일 작성.
- hadb2-eee.cfg 파일 작성.
- EEE 인스턴스에 대한 .rhosts 파일 작성.
- 기본 데이터베이스 경로에서 데이터 디렉토리를 주관하는 연관된 논리 호스트로 기호 링크 작성.

- hadb2tab 파일에서 행을 추가.

구성 오류를 방지하고 HA 인스턴스가 **hadb2_setup** 명령을 수행한 후에 시작할 수 있도록 보장하기 위해, 명령은 새로운 인스턴스가 추가되기 전에 상당히 많은 테스트를 수행합니다.

db2nodes.cfg 파일은 현재 클러스터 상태에 대응하는 정보로 작성되고 제공됩니다. 예를 들어, 논리 호스트 ″log0″이 머신 ″crackle″에 의해 주관되는 경우, ″log0″과 연관된 데이터베이스 파티션의 항목은 머신 이름 ″crackle″과 ″crackle″의 고속 상호 연결을 포함할 것입니다.

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 thrash 0 204.152.65.34
3 thrash 1 204.152.65.34
```

hadb2-eee.cfg 파일은 명령에 지정된 노드 정보에 근거하여서만 작성됩니다. 데이터베이스 파티션 당 하나의 행이 있습니다.

```
sphere % cat hadb2-eee.cfg
NODE: log0  0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

.rhost 파일은 DB2 UDB EEE용으로 필요하며, 클러스터 내의 각 머신에 대해 모든 호스트 이름(또는 IP 주소)을 포함해야 합니다. 예를 들면, 다음과 같습니다.

```
crackle db2eee
204.152.65.1 db2eee
204.152.65.17 db2eee
thrash db2eee
204.152.65.2 db2eee
204.152.65.18 db2eee
crackle db2eee
jolt db2eee
bump db2eee
thrash.torolab.ibm.com db2eee
crackle.torolab.ibm.com db2eee
```

SMS 테이블 공간의 파일 시스템 레이아웃에 따라, **hadb2_setup** 명령은 몇 개의 디렉토리 및 기호 링크를 설정합니다. 여기에는 다음이 포함됩니다.

- 각 논리 호스트의 논리 호스트 파일 시스템 아래에 있는 "data"라는 디렉토리.
- 논리 호스트와 연관된 각 데이터베이스 파티션의 노드 디렉토리(이 "data" 디렉토리 아래에 있는).
- ~<instance> 아래에 위치한 기본 데이터베이스 경로에 있는 기호 링크, 여기서 ~*instance*는 인스턴스의 홈 디렉토리입니다. 해당 노드 디렉토리를 지시하는 각 데이터베이스 파티션에 대한 하나의 기호 링크가 있습니다. 311 페이지의 『EE 및 EEE 인스턴스에 대한 디스크 레이아웃』에서 자세한 내용을 참조하십시오.

장애 복구 시간

장애 복구 시간은 데이터가 처음 사용 불가능할 때부터 다시 사용 가능하게 될 때까지 측정됩니다. 장애 복구 중 발생한 이벤트 수는 파일복구 시간에 상당히 중요합니다.

- 디스크 내보내기 및 가져오기.

  디스크 내보내기 및 가져오기는 보통 다른 이벤트와 비교할 때 전체 다운 시간에 사용되기는 하지만 그렇게 긴 시간은 걸리지 않습니다. 장애 복구 중 한 머신에서 다른 머신으로 이동되는데 필요한 디스크가 많을수록, 프로세스 시간이 더 깁니다. 결함이 있는 디스크가 있는 경우, 프로세스가 더 오래 걸릴 수 있습니다.

- 논리 호스트용으로 마운트된 파일 시스템의 fsck.

  논리 호스트의 파일 시스템이 마운트되기 전에, fsck를 전달하여 파일 시스템의 건재를 확인해야 합니다. 파일 시스템이 클수록, 이 프로세스에 걸리는 시간이 더 깁니다. 저널화된 파일 시스템을 사용하여, 이 시간은 현저하게 감소될 수 있습니다. 저널화된 파일 시스템은 보통 HA 환경에서 사용되므로, fsck 시간은 보통 문제가 되지 않습니다.

- HA 에이전트에서 호출된 사용자 스크립트.

  HA 에이전트는 사용자 스크립트가 있으며 실행가능한 경우 사용자 스크립트를 호출합니다. 이 스크립트 중 일부는 비동기적으로 수행하며, HA 인스턴스를 제공하는데 걸리는 시간을 추가할 수 있습니다. 가능한 한 빨리 스크립트를 수행하고, 백그라운드에서 이 스크립트에 의해 호출되는 모든 외부 프로그램을 수행하도록 고려하십시오.

- HA-NFS.

  상호 인계 구성에서의 단일 EEE 인스턴스의 경우, HA-NFS는 인스턴스 소유자의 홈 디렉토리용으로 사용되어야 합니다. HA-NFS는 *lockd*(HA-NFS의 HA 에이전트에 정의됨)의 유예 기간 때문에 HA-NFS를 수행할 때 90초인 장애 복구 시간을 추가합니다. 이것은 장애 복구 시간에 영향을 미치게 되는데, 이는 장애 복구 이후에 HA-NFS 파일 시스템에서 파일을 잠그는 모든 프로세스는 유예 기간이 끝날 때까지 기다려야 하기 때문입니다. DB2의 HA 에이전트는 장애 복구 이후의 인스턴스 소유자의 홈 디렉토리 아래에서 파일을 잠그는 첫 번째 프로세스이며, 첫 번째 잠금을 얻기 위해 취하는 시간을 기록합니다. 이 시간은 장애 복구 이후에 상태 보고서에 표시됩니다.

- DB2 시작.

  DB2 시작은 장애 복구 시간에 적은 시간만을 제공합니다. EE 인스턴스의 경우, 평균적으로 5-15초 정도 제공합니다. EEE 인스턴스의 경우, 약 10초에 장애 복구되고 있는 데이터베이스 파티션당 5초 정도를 추가하여 제공합니다. 예를 들어, 세 개의 데이터베이스 파티션이 장애 복구되고 있는 경우, 이 세 개의 데이터베이스 파티션을 시작하여 제공되는 장애 복구 시간은 대략 25초입니다. 이것은 인스턴스의 데이터베이스에 대한 응급 복구를 포함하지 *않습니다*.

- 데이터베이스 응급 복구.

  응급 복구는 보통 장애 복구와 연관된 대부분의 다운 시간에 제공됩니다. 데이터베이스를 복구하는데 걸리는 시간은 다음을 포함하는 인수 수에 좌우됩니다.

  – 클라이언트 워크로드. 데이터베이스에 대한 변경만이 트랜잭션 로그에 기록됩니다. 클라이언트 워크로드가 대개 읽기 전용 조작인 경우, 상대적으로 적은 트랜잭션이 응급 복구 중 데이터베이스에 적용되어야 합니다.

  – 디스크 및 머신 속도. HA 인스턴스를 주관하는 머신과 디스크의 속도도 데이터베이스를 복구하는데 걸리는 시간에 기여합니다. 시스템이 빠를수록, 응급 복구 시간이 적게 걸립니다.

  – *softmax* 데이터베이스 구성 매개변수의 값. *softmax*의 값은 소프트 점검점이 취해질 때의 로그 파일 크기 백분율이며, 로그 제어 파일이 작성됩니다. 어떤 로그 레코드가 데이터베이스를 일관된 상태로 복원하는데 필요한 지를 판별하기 위해 응급 복구 중 로그 제어 파일이 사용됩니다. 이 값을 줄이면

데이터베이스 관리 프로그램이 페이지 크리너를 더 자주 트리거하며, 더 자주 소프트 점검점을 얻으며, 성능은 저하되지만 데이터베이스 복구는 더 빠릅니다.

– 인스턴스가 EE 또는 EEE인지 여부. 인스턴스가 EEE 인스턴스이면, 데이터베이스 재시작 조작은 병렬로 수행됩니다. 각 데이터베이스 파티션은 데이터베이스 중 관련된 고유한 부분을 재시작하는데 관여합니다. 데이터베이스의 50GB 데이터가 있는 경우, 네 개의 데이터베이스 파티션이 있는 인스턴스는 EE 인스턴스보다 4배 정도 더 빠르게 데이터베이스를 복구할 수 있습니다.

문제점 해결

다음 테이블은 발생할 수 있는 문제점과, 그럴만한 이유 그리고 문제점을 해결하기 위해 취할 수 있는 조치를 식별합니다.

표 16. Sun Cluster 2.2의 고가용성 문제점 해결

Symptom	가능한 원인	조치
논리 호스트 파일 시스템을 마운트할 수 없음	논리 호스트 파일 시스템은 보통 논리 호스트의 장애 복구 중 마운트되고 마운트 해제됩니다. 장애 복구 중, 논리 호스트 파일 시스템 아래에서 활동중인 프로세스나 열린 파일이 없어야 합니다. 드물게, 종료될 수 없는 프로세스는 논리 호스트 파일 시스템 아래에 현재 작업 디렉토리를 갖습니다. 프로세스가 마운트 지점 아래에 있는지 파악하려면, `fuser(1m)`이나 `lsof`라고 하는 GNU 유틸리티를 사용하십시오. 오류 메시지는 논리 호스트 파일 시스템이 마운트될 수 없을 때 생성됩니다.[a]	시스템을 재부트하거나, 논리 호스트 파일 시스템을 다른 이름으로 이동시키고 재작성하십시오. 이것을 수행하면 동결 프로세스가 디렉토리 아래에 유지되게 하며(종료될 수 없으므로), 마운트가 발생하게 합니다.[b]
db2start 또는 db2stop 시간 종료가 작동하지 않음	SIGALRM 신호는 블로킹 시스템 호출로부터 끊어질 수 없습니다. 그 대신, 시스템 호출은 SA_RESTART 플래그가 `sigaction()`으로 설정된 것처럼 재시작합니다. 이것은 DB2 HA 에이전트가 무시되도록 시간종료를 야기하며, 에이전트 메소드는 정지된 **db2start** 또는 **db2stop** 명령에서 복구하는 대신 정지시킵니다.	Solaris 2.6의 경우 필수 패치, 105210-17(또는 이상)을 적용하십시오.

표 16. Sun Cluster 2.2의 고가용성 문제점 해결 (계속)

Symptom	가능한 원인	조치
인스턴스 정지로 로깅	발생할 수 있는 여러가지 이유가 있지만, 대부분의 일반적인 이유는 NFS 문제점 및 /usr/sbin/quota 프로그램을 포함합니다.	NFS 마운트를 점검하여 건재한지 확인하고 인스턴스 소유자가 소유하는 할당 프로세스를 살펴보십시오. 시스템 관리자의 재량으로, /bin/true에 대한 기호 링크로 할당 프로그램을 변경하면 문제점을 해결할 수 있습니다. 이것은 권장되는 솔루션은 아니지만 작동할 수 있습니다.
방금 EEE 인스턴스를 설정했으나 시작하지 않음	**hadb2_setup** 명령은 /etc/services 파일에 포트를 추가하지 않으며, 관리자가 수동으로 포트를 추가할 것으로 예상합니다. 오류 메시지가 리턴됩니다.[c]	적절한 포트 이름을 /etc/services 파일에서 명명했는지 확인하십시오.
START_NET 메소드가 DB2를 시작할 수 없음		인스턴스가 장애 복구되지 않는지 확인하려면 결함 모니터링을 해제하십시오. 인스턴스 소유자로서 로그인하고, 수동으로 DB2를 시작하십시오. 1. hadb2tab 구성 파일이 지정된 올바른 인스턴스 유형을 가지고 있는지 확인하십시오. 예를 들어, EE 관리 인스턴스의 db2nodes.cfg 파일을 문제점을 야기하며, HA 에이전트 메소드는 여기에서 복구될 수 없습니다. 2. .rhosts 파일이 있고, 이 파일에 유효한 항목이 있는지 확인하십시오. 3. HA-NFS 파일 시스템이 클러스터 내의 모든 머신의 루트 사용권한으로 공유되는지 확인하십시오. 4. 커널 매개변수를 점검하여 올바른지 확인하십시오. 5. /etc/services 파일에 인스턴스에 대한 항목이 들어 있는지 확인하십시오.

표 16. Sun Cluster 2.2의 고가용성 문제점 해결 (계속)

Symptom	가능한 원인	조치
인스턴스만이 하나의 머신에서 작업함	• 인스턴스의 숫자 *uid*는 클러스터 내의 각 머신에서 동일하지 않을 수 있습니다. • 커널 매개변수는 클러스터 내의 각 머신에서 유효하지 않을 수 있습니다. • hadb2tab 파일은 클러스터 내의 각 머신에서 동일하지 않을 수 있습니다. • 논리 호스트 vfstab 파일과 같이, 다른 구성 파일은 클러스터 내의 각 머신에서 동일하지 않을 수 있습니다.	이 원인들 중 어느 것도 적용되지 않는 경우, 인스턴스 소유자로서 로그인하고 수동으로 DB2를 시작하십시오. EE 인스턴스의 경우, 인스턴스를 주관하는 논리 호스트를 현재 머신에서 주관하고 있는 경우 DB2가 작동해야 합니다. EEE 인스턴스의 경우, 데이터베이스 파티션을 주관할 수 있는 클러스터 내의 모든 머신에서 DB2가 작동해야 합니다.
su - <instance> -c "db2start"가 작업하지 않음	• 인스턴스의 .profile은 **su**-"friendly"가 될 수 없습니다. • **su** 명령이 수동으로 작업하지만, HA 에이전트를 통해서는 작업할 수 없는 Bourne 쉘(/bin/sh)에 알려진 문제점이 있습니다.	• 루트로서, 이 명령을 수동으로 실행하고, HA 에이전트를 통해 다시 시도하기 전에 이 명령이 작업하는지 확인하십시오. • 필요한 경우 Korn shell(/bin/ksh)로 전환하십시오.
EEE 인스턴스는 시작할 수 없으나, 홈 디렉토리가 마운트됨	HA-NFS 디렉토리는 "루트" 사용권한"을 사용하여 클러스터 내의 머신으로 내보낼 수 없었습니다. DB2 및 HA 에이전트 둘 다 이것을 제대로 실행하도록 요구합니다.	이를 테스트하려면, 인스턴스 소유자 아래에서 파일을 작성하십시오(루트로서).
EEE 인스턴스 디렉토리가 "Stale NFS 파일 핸들" 오류를 리턴함	인스턴스 소유자의 홈 디렉토리 아래에서 여전히 처리될 수 있습니다.	인스턴스 소유자의 홈 디렉토리를 마운트 해제하고, HA 에이전트가 다시 마운트하게 하십시오. hadb2 서비스가 해제되고 다시 설정되면 HA 에이전트가 다시 마운트합니다. (330 페이지의 『hadb2_setup 명령』에 있는 **hadb2_setup** 명령에서 -s 스위치 설명 부분을 참조하십시오.)

표 16. Sun Cluster 2.2의 고가용성 문제점 해결 (계속)

Symptom	가능한 원인	조치
제어 방법이 SC2.2를 통해 정상적으로 수행하지 않음	hadb2 서비스는 Sun Cluster 소프트웨어로 등록될 수 없거나 설정될 수 없습니다.	제어 방법이 명령행에서 정상적으로 수행하도록 나타나는 경우, 문제점을 설명하도록 돕는 오류 메시지에 대해 SYSLOG 파일을 점검하십시오. hadb2 서비스가 Sun Cluster 소프트웨어로 등록되고 설정되는지 확인하십시오. 수동으로 메소드를 수행하면 문제점을 디버그하는데 유용합니다.[d] 메소드는 루트로서 수행되어야 하며 적절한 명령행 인수를 제공해야 합니다. 논리 호스트의 목록이 nil이면, 인수는 ""로서 제공되어야 합니다. 공백 구분자가 없는 따옴표는 공백 인수를 나타냅니다. 예를 들면, 다음과 같습니다. `hadb2_startnet log0,log1 "" 600` 첫 번째 인수 log0,log1은 논리 호스트 log0 및 log1이 현재 머신에 의해 주관되고 있음을 hadb2_startnet 메소드에 알립니다. 두 번째 인수가 nil이면, 클러스터 내의 다른 머신에서 주관되는 다른 논리 호스트가 없음(모든 논리 호스트는 현재 머신에 있음)을 hadb2_startnet 메소드에 알립니다. 세 번째 인수는 600초 이후에 SC2.2가 시간종료됨을 알립니다.
사용자 스크립트가 수행하지 않음	적절한 디렉토리가 있으며 실행 가능한 경우에만 사용자 스크립트가 수행될 수 있습니다.	파일 소유권 및 속성을 점검하십시오. 스크립트가 여전히 수행하는데 실패하면, IBM 서비스 담당자에게 문의하십시오. 수행하지 않는 스크립트의 디렉토리 목록과 클러스터를 수행해야 하는 파일복구 또는 클러스터 재구성의 SYSLOG 출력을 전달하십시오.

표 16. Sun Cluster 2.2의 고가용성 문제점 해결 (계속)

Symptom	가능한 원인	조치
정보가 /etc/syslog.conf에서 지정하는 파일로 기록되지 않음		touch(1)를 사용하여 /etc/syslog.conf 파일에서 지정하는 파일을 작성한 후 SYSLOG 데몬을 재시작하십시오.

[a] 논리 호스트 파일 시스템이 마운트될 수 없을 때 생성되는 오류 메시지는 다음과 같습니다.

```
Aug 17 11:14:01 rash ID[SUNWcluster.loghost.1170]: importing data1
Aug 17 11:14:06 rash ID[SUNWcluster.scnfs.3040]: mount -F ufs -o ""
     /dev/vx/dsk/data1/data1-stat /log1 failed.
Aug 17 11:14:07 rash ID[SUNWcluster.ccd.ccdd.5304]: error freeze cmd =
     /opt/SUNWcluster/bin/loghost_sync
CCDSYNC_POST_ADDU LOGHOST_CM:log1:rash /etc/opt/SUNWcluster/conf/ccd.database
     2 "0 1" 1 error code = 1
```

[b] 예:

```
scadmin@rash(218)# ps -fe | egrep db2
db2ee 1984 1 0 0:01 <defunct>

Solution:
   scadmin@rash(229)# cd /
   scadmin@rash(230)# mv /log1 /log1.bkp
   scadmin@rash(231)# mkdir /log1
```

[c] 오류 메시지는 다음과 유사합니다.

```
SQL6030N START 또는 STOP DATABASE MANAGER가 실패했습니다. 이유 코드 "13".
```

[d] 예를 들어, hadb2_startnet 메소드가 libdb2.so.1을 찾을 수 없으나, 보통 Sun Cluster 소프트웨어를 통해 수행하는 경우 오류가 보고되지 않습니다. 메소드를 수동으로 수행하면 다음과 결과를 갖습니다.

```
scadmin@crackle(213)# hadb2_startnet '''log0,log1' 600
ld.so.1: hadb2_startnet: fatal: libdb2.so.1: open failed:
   No such file or directory
Killed
```

제3부 부록 및 끝머리

부록A. 구문 다이어그램을 읽는 방법

구문 다이어그램은 운영 체제가 입력된 것을 올바르게 해석할 수 있도록 명령을 지정하는 방법을 보여줍니다.

왼쪽에서 오른쪽으로, 맨 위에서 맨 아래로 가로 줄(주 경로)을 따라 구문 다이어그램을 읽도록 하십시오. 가로 줄이 화살촉으로 끝날 경우, 명령 구문은 계속되며 다음 줄도 화살촉으로 시작합니다. 수직 막대는 명령 구문의 끝을 표시합니다.

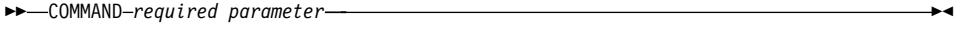
구문 다이어그램에서 정보를 입력할 때, 따옴표 및 등호와 같은 구두점을 포함하도록 하십시오.

매개변수는 키워드나 변수로 분류됩니다.

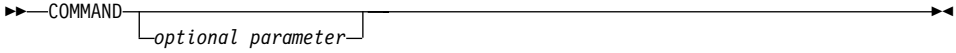
- 키워드는 상수를 나타내며 대문자로 표시됩니다. 그러나 명령 프롬프트에서 키워드는 대문자, 소문자 또는 대소문자 혼합하여 입력할 수 있습니다. 명령 이름은 키워드의 예입니다.
- 변수는 사용자가 제공하는 이름이나 값을 나타내며 소문자로 표시됩니다. 그러나 명령 프롬프트에서 변수는 명시적으로 대소문자 제한사항이 명시되지 않으면 대문자, 소문자 또는 대소문자 혼합하여 입력할 수 있습니다. 파일 이름은 변수의 예입니다.

매개변수는 키워드와 변수의 조합이 될 수 있습니다.

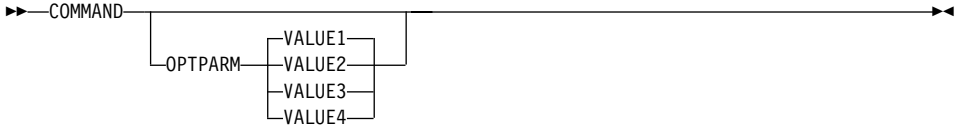
필수 매개변수는 주 경로에 표시됩니다.



선택적 매개변수는 주 경로 아래에 표시됩니다.



매개변수의 기본값은 경로 바로 위에 표시됩니다.



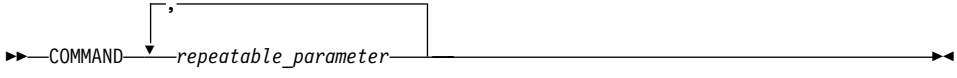
첫 번째 매개변수가 주 경로에 표시되는 매개변수 스택은 그 매개변수 중 하나를 선택해야 함을 나타냅니다.

첫 번째 매개변수가 주 경로 아래에 표시되는 매개변수 스택은 그 매개변수 중 하나를 선택할 수 있음을 나타냅니다.

경로 위에서 왼쪽으로 돌아가는 화살표는 다음 규칙에 따라 항목을 반복할 수 있음을 나타냅니다.

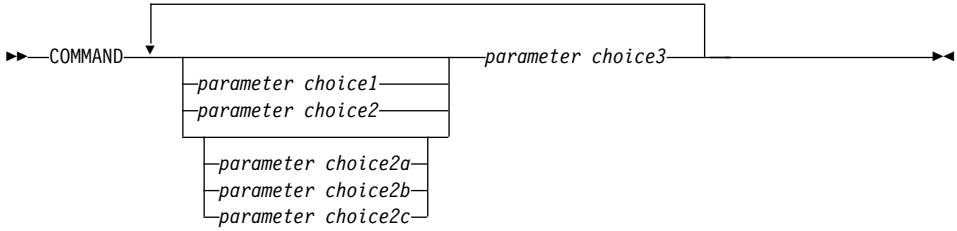
- 화살표가 연속되면, 항목들이 공백으로 구분되는 목록에서 항목을 반복할 수 있습니다.

- 화살표에 쉼표가 있으면, 항목들이 쉼표로 구분되는 목록에서 항목을 반복할 수 있습니다.

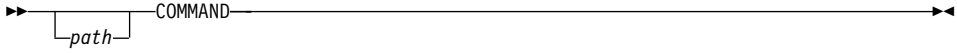


매개변수 스택의 항목들은 이전에 설명된 필수 및 선택적 매개변수에 대한 스택 규칙에 따라 반복할 수 있습니다.

일부 구문 다이어그램에는 다른 매개변수 스택 내의 매개변수 스택이 포함될 수 있습니다. 스택의 항목들은 이전에 설명된 규칙에 따라 반복할 수 있습니다. 즉, 내부 스택에 반복 화살표가 없지만 외부 스택에는 있을 경우, 내부 스택의 한 매개변수만 선택하여 외부 스택의 매개변수와 결합할 수 있으며, 그 결합을 반복할 수 있습니다. 예를 들어, 다음은 매개변수 *choice2a*를 매개변수 *choice2*와 결합한 다음 다시 조합(*choice2 + choice2a*)을 반복하는 다이어그램입니다.



일부 명령은 선택적 경로 매개변수에 의해 진행됩니다.



이 매개변수를 제공하지 않을 경우, 시스템은 명령에 대한 현재 디렉토리를 검색합니다. 명령을 찾을 수 없을 경우, 시스템은 `.profile`에 나열된 경로의 모든 디렉토리에서 명령을 계속 검색합니다.

일부 명령에는 기능적으로 같은 구문적 변형이 있습니다.

부록B. 경고, 오류 및 완료 메시지

다양한 백업 및 복원 유틸리티에 의해 생성된 메시지가 SQL 메시지 사이에 포함됩니다. 이 메시지는 경고 또는 오류 상태가 검출되었을 때 데이터베이스 관리 프로그램에서 생성됩니다. 각 메시지에는 접두부(SQL) 및 네 자리 또는 다섯 자리의 메시지 번호로 구성되는 메시지 식별자가 있습니다. 통지, 경고 및 중요와 같은 세 가지의 메시지 유형이 있습니다. N으로 끝나는 메시지 식별자는 오류 메시지입니다. W로 끝나는 메시지는 경고 또는 정보 메시지를 표시합니다. C로 끝나는 메시지 식별자는 중요한 시스템 오류를 표시합니다.

메시지 번호는 *SQLCODE*로도 언급됩니다. SQLCODE는 메시지 유형(N, W 또는 C)에 따라 양수 또는 음수로 응용프로그램에 전달됩니다. N 및 C는 음수 값을 생성하고 W는 양수 값을 생성합니다. DB2는 SQLCODE를 응용프로그램에 리턴하고, 응용프로그램은 그 SQLCODE와 연관되는 메시지를 가져옵니다. DB2는 또한 SQL문의 결과가 될 수 있는 상태에 대해 *SQLSTATE* 값을 리턴합니다. 일부 SQLCODE 값은 SQLSTATE 값과 연관됩니다.

모든 DB2 메시지에 대한 자세한 정보는 *메시지 참조서*를 참조하십시오. 이 책에 포함된 정보를 사용하여 오류나 문제점을 식별하고 해당되는 복구 조치를 사용하여 문제점을 해결할 수 있습니다. 이 정보는 또한 메시지가 생성되고 로그되는 곳을 알기 위해서도 사용할 수 있습니다.

SQL 메시지 및 SQLSTATE 값과 연관되는 메시지 텍스트도 운영 체제 명령행에서 액세스할 수 있습니다. 이 오류 메시지들에 대한 도움말에 액세스하려면, 운영 체제 명령 프롬프트에서 다음을 입력하십시오.

```
db2 ? SQLnnnnn
```

여기서, *nnnnn*은 메시지 번호를 나타냅니다. UNIX 기반 시스템에서는 큰따옴표 분리문자를 사용할 것을 권장합니다. 그러면 디렉토리에 단일 문자 파일 이름이 있어도 문제점이 발생하지 않게 됩니다.

```
db2 "? SQLnnnnn"
```

db2 명령에 대한 매개변수로 승인된 메시지 식별자는 대소문자가 구별되지 않으며, 종료 문자가 필요하지 않습니다. 그러므로 다음 명령들의 결과는 같습니다.

```
db2 ? SQL0000N
db2 ? sql0000
db2 ? SQL0000n
```

화면에 비해 메시지 텍스트가 너무 길면, 다음 명령을 사용하십시오(UNIX 기반 운영 체제와 ″more″ 파이프를 지원하는 다른 운영 체제에서).

```
db2 ? SQLnnnnn | more
```

출력을 찾아볼 수 있는 파일로 보낼 수도 있습니다.

대화식 입력 모드에서 도움말을 호출할 수도 있습니다. 이 모드에 액세스하려면, 운영 체제 명령 프롬프트에서 다음을 입력하십시오.

```
db2
```

이 모드에서 DB2 메시지 도움말을 얻으려면, 명령 프롬프트(db2 =>)에서 다음을 입력하십시오.

```
? SQLnnnnn
```

다음을 발행하여 SQLSTATE와 연관되는 메시지 텍스트를 검색할 수 있습니다.

```
db2 ? nnnnn
 또는
db2 ? nn
```

여기서 *nnnnn*은 5자 SQLSTATE 값(영숫자)이고 *nn*은 두 자리의 SQLSTATE 클래스 노드(SQLSTATE 값의 처음 두 자리)입니다.

부록C. 추가 DB2 명령

db2adutl - TSM 아카이브 이미지 작업

사용자가 Tivoli Storage Manager(이전의 ADSM)를 사용하여 저장한 로드 사본 이미지와 백업 이미지, 로그를 조회, 추출, 검증 및 삭제할 수 있도록 합니다.

UNIX 기반 시스템에서, 이 유틸리티는 INSTHOME/sqllib/misc 디렉토리에 위치됩니다. Windows 운영 체제 및 OS/2에서는 \sqllib\misc 디렉토리에 위치됩니다.

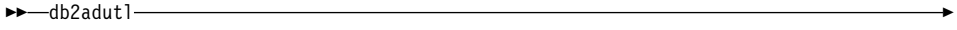
권한 부여

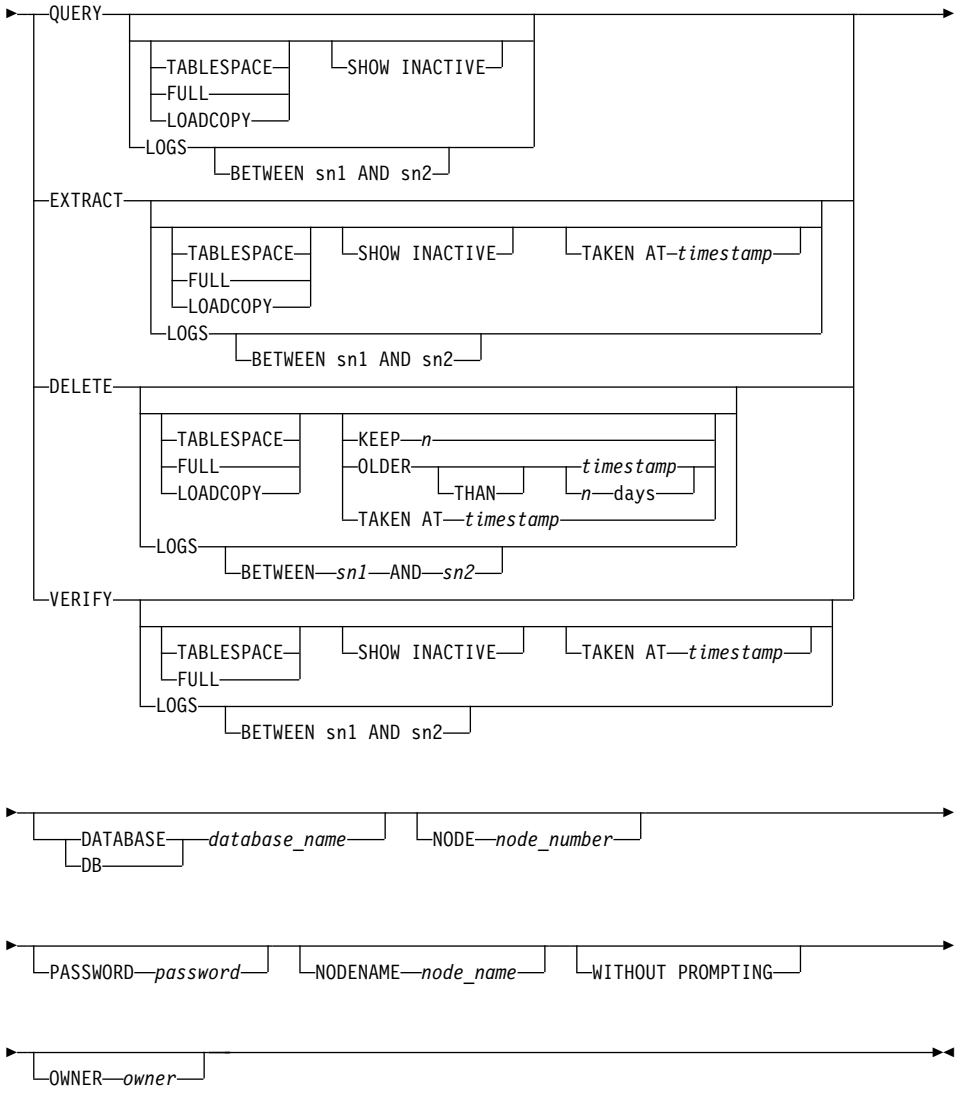
없음

필수 연결

없음

명령 구문





명령 매개변수

QUERY

DB2 오브젝트의 TSM 서버를 조회합니다.

EXTRACT

TSM 서버로부터 지역 머신의 현재 디렉토리로 DB2 오브젝트를 복사합니다.

DELETE

백업 오브젝트를 비활성화하거나 TSM 서버에서 로그 아카이브를 삭제합니다.

VERIFY

서버에 있는 백업 사본에 대해 일관성 점검을 수행합니다.

주: 이 매개변수는 전체 백업 이미지가 네트워크를 거쳐 전송되도록 합니다.

TABLESPACE

테이블 공간 백업 이미지만을 포함합니다.

FULL 전체 데이터베이스 백업 이미지만을 포함합니다.

LOADCOPY

로드 사본 이미지만을 포함합니다.

LOGS

로그 아카이브 이미지만 포함합니다.

BETWEEN *sn1* **AND** *sn2*

로그 순차 번호 1과 로그 순차 번호 2 사이의 로그가 사용됨을 지정합니다.

SHOW INACTIVE

비활성화된 백업 오브젝트를 포함합니다.

TAKEN AT *timestamp*

시간소인으로 백업 이미지를 지정합니다.

KEEP *n*

시간소인으로 최근의 *n*을 제외하고 지정 유형의 모든 오브젝트를 비활성화합니다.

OLDER THAN *timestamp* **또는** *n* **days**

timestamp 또는 *n*일 동안 보다 이른 시간소인을 가진 오브젝트가 비활성화됩니다.

DATABASE *database_name*

지정된 데이터베이스 이름과 연관되는 오브젝트만 고려합니다.

NODE *node_number*

지정된 노드 번호로 작성된 오브젝트만 고려합니다.

PASSWORD *password*

해당 노드에 대한 TSM 클라이언트 암호를 지정합니다(필수인 경우). 데이터베이스는 지정하고 암호는 제공하지 않을 경우, *tsm_password* 데이터베이스 구성 매개변수에 지정한 값이 TSM에 전달되며, 그렇지 않은 경우에는 암호가 사용되지 않습니다.

NODENAME *node_name*

특정 TSM 노드 이름과 연관되는 이미지만 고려합니다.

WITHOUT PROMPTING

오브젝트가 삭제되기 전에 검증하도록 프롬프트하지 않습니다.

OWNER *owner*

지정된 소유자가 작성한 오브젝트만 고려합니다.

예

다음은 db2 backup database rawsampl use tsm에서 출력되는 샘플입니다.

```
Backup successful. The timestamp for this backup is : 19970929130942

db2adutl query
Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 1, Time: 19970929130942,
                                  Oldest log: S0000053.LOG, Sessions used: 1
   full database backup image: 2, Time: 19970929142241,
                                  Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   table space backup image: 1, Time: 19970929094003,
                                Oldest log: S0000051.LOG, Sessions used: 1
   table space backup image: 2, Time: 19970929093043,
                                Oldest log: S0000050.LOG, Sessions used: 1
   table space backup image: 3, Time: 19970929105905,
                                Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

다음은 db2adutl delete full taken at 19950929130942 db rawsampl에서 출력되는 샘플입니다.

```
Query for database RAWSAMPL

Retrieving full database backup information.  Please wait.

  full database backup image: RAWSAMPL.0.db26000.0.19970929130942.001

  Do you want to deactivate this backup image (Y/N)? y

    Are you sure (Y/N)? y
db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 2, Time: 19950929142241,
                               Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   table space backup image: 1, Time: 19950929094003,
                             Oldest log: S0000051.LOG, Sessions used: 1
   table space backup image: 2, Time: 19950929093043,
                             Oldest log: S0000050.LOG, Sessions used: 1
   table space backup image: 3, Time: 19950929105905,
                             Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

db2ckbkp - 백업 점검

이 유틸리티는 백업 이미지의 무결성을 테스트하고 이미지를 복원할 수 있는지를 판별하기 위해 사용할 수 있습니다. 백업 헤더에 저장된 메타데이터를 표시하기 위해 사용할 수도 있습니다.

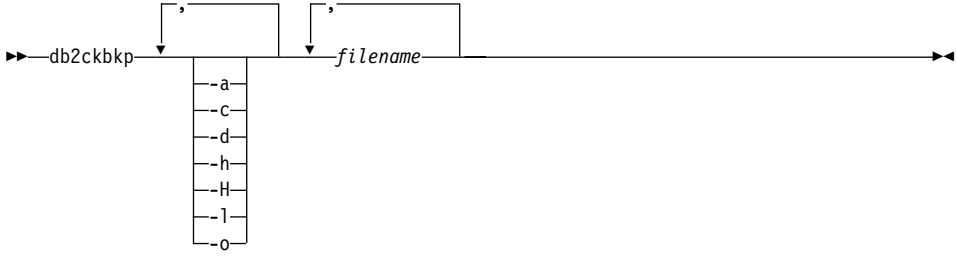
권한 부여

누구나 유틸리티에 액세스할 수는 있지만, 이미지 백업에 대해 이 유틸리티를 실행하려면 그 이미지 백업에 대한 읽기 사용권한을 가지고 있어야 합니다.

필수 연결

없음

명령 구문



명령 매개변수

-a 모든 사용 가능한 정보를 표시합니다.

-c 체크비트와 체크썸의 결과를 표시합니다.

-d DMS 테이블 공간 데이터 페이지의 헤더에서 정보를 표시합니다.

-h 복원 유틸리티에서 요구하는 이미지의 이름과 경로를 포함하는 미디어 헤더 정보를 표시합니다.

-H 미디어 헤더 정보만을 표시합니다.

주:

1. 이 옵션은 이미지의 유효성을 확인하지 않습니다. 이 옵션을 지정하지 않을 경우 유효성 확인은 전제 이미지에 대해 수행됩니다.
2. 이 옵션은 다른 옵션과 함께 사용할 수 없습니다.

-l 로그 파일 헤더 데이터를 표시합니다.

-o 오브젝트 헤더로부터 자세한 정보를 표시합니다.

filename

백업 이미지 파일의 이름. 하나 이상의 파일을 동시에 점검할 수 있습니다.

주:

1. 전체 백업이 여러 오브젝트로 구성될 경우, 유효성 확인은 **db2ckbkp**를 사용하여 동시에 모든 오브젝트의 유효성을 확인할 경우에만 성공합니다.
2. 이미지의 여러 파트를 확인할 경우, 첫 번째 백업 이미지 오브젝트(.001)를 먼저 지정해야 합니다.

예

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.*
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.
db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001
=====================
MEDIA HEADER REACHED:
=====================
        Server Database Name           -- SAMPLE2
        Server Database Alias          -- SAMPLE2
        Client Database Alias          -- SAMPLE2
        Timestamp                      -- 19990818122909
        Node                           -- 0
        Instance                       -- krodger
        Sequence Number                -- 1
        Release ID                     -- 900
        Database Seed                  -- 65E0B395
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
```

```
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Node Number            -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        Backup Buffer Size             -- 4194304
        Number of Sessions             -- 1
        Platform                       -- 0
The proper image file name would be:
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

[1] Buffers processed:  ####
Image Verification Complete - successful.
```

사용법 주

여러 세션을 사용하여 백업 이미지를 작성한 경우, **db2ckbkp**는 동시에 모든 파일을 조사할 수 있습니다. 순차 번호가 001인 세션이 지정되는 첫 번째 파일인지 확인하는 것은 사용자의 책임입니다.

이 유틸리티는 테이프에 저장된 백업 이미지를 검증할 수도 있습니다(가변 블록 크기로 작성된 이미지 제외). 이는 복원 조작에서처럼 테이프를 준비한 다음 테이프 장치 이름을 지정하여 유틸리티를 호출하면 수행됩니다. UNIX 기반 시스템의 예 :

```
db2ckbkp -h /dev/rmt0
```

Windows NT의 예:

```
db2ckbkp -d \\.\tape1
```

백업 이미지가 TSM에 상주할 경우, 350 페이지의 『db2adutl - TSM 아카이브 이미지 작업』을 참조하십시오.

db2ckrst - 증분 복원 이미지 순서 확인

데이터베이스 실행기록을 조회하고 증분 복원 조작에 대해 필요한 백업 이미지에 대해 시간소인 목록을 생성합니다. 수동 증분 복원에 대한 간단한 복원 구문도 생성됩니다.

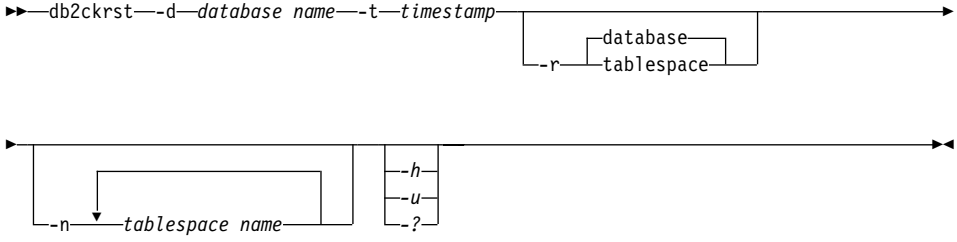
권한 부여

없음

필수 연결

없음

명령 구문



명령 매개변수

-d database namefile-name

복원할 데이터베이스에 대한 별명 이름을 지정합니다.

-t timestamp

점차적으로 복원될 백업 이미지에 대한 시간소인을 지정합니다.

-r 수행할 복원 유형을 지정합니다. 기본값은 데이터베이스입니다.

주: 테이블 공간을 선택하였는데 테이블 공간 이름을 제공하지 않을 경우, 유틸리티는 지정된 이미지의 실행기록 항목을 보고 나열된 공간 이름을 사용하여 복원을 수행합니다.

-n tablespace name

복원할 하나 이상의 테이블 공간 이름을 지정합니다.

주: 데이터베이스 복원 유형을 선택하고 테이블 공간 이름 목록을 지정할 경우, 유틸리티는 지정된 테이블 공간 이름을 사용하여 공간 복원 조작을 계속합니다.

-h/-u/-?

도움말 정보 표시. 이 옵션을 지정할 경우, 다른 모든 옵션이 무시되고 도움말 정보만 표시됩니다.

예

```
db2ckrst -d mr -t 20001015193455 -r database
db2ckrst -d mr -t 20001015193455 -r tablespace
db2ckrst -d mr -t 20001015193455 -r tablespace -n tbsp1 tbsp2
> db2 backup db mr
Backup successful. The timestamp for this backup image is : 20001016001426
> db2 backup db mr incremental
Backup successful. The timestamp for this backup image is : 20001016001445
> db2ckrst -d mr -t 20001016001445
Suggested restore order of images using timestamp 20001016001445 for database mr.
==========================================================================
  db2 restore db mr incremental taken at 20001016001445
  db2 restore db mr incremental taken at 20001016001426
  db2 restore db mr incremental taken at 20001016001445
==========================================================================
> db2ckrst -d mr -t 20001016001445 -r tablespace -n userspace1
Suggested restore order of images using timestamp 20001016001445 for database mr.
==========================================================================
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001445
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001426
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001445
==========================================================================
```

사용법 주

이 유틸리티를 사용하려면 데이터베이스 실행기록이 있어야 합니다. 데이터베이스 실행기록이 없으면, 이 유틸리티를 사용하기 전에 RESTORE DATABASE 명령에 HISTORY FILE 옵션을 지정하십시오.

PRUNE HISTORY 명령의 FORCE 옵션을 사용할 경우, 최근의 전체 데이터베이스 백업 이미지에서 복구하는데 필요한 항목을 삭제할 수 있습니다. PRUNE HISTORY 명령의 기본 조작은 필수 항목이 삭제되지 않도록 합니다. PRUNE HISTORY 명령의 FORCE 옵션은 사용하지 않는 것이 좋습니다.

백업 조작의 완전한 레코드를 보존하고 이 유틸리티를 지침으로 사용하는 것이 좋습니다.

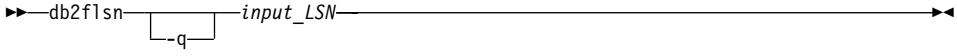
db2flsn - 로그 순차 번호 찾기

지정한 로그 순차 번호(LSN)에 의해 식별되는 로그 레코드를 포함하는 파일의 이름을 리턴합니다.

권한 부여

없음

명령 구문



명령 매개변수

-q 로그 파일 이름만 인쇄됨을 지정합니다. 오류 또는 경고 메시지가 인쇄되지 않으므로, 상태는 리턴 코드를 통해서만 판별할 수 있습니다. 유효한 코드는 다음과 같습니다.

- -100 올바르지 않은 입력
- -101 LFH 파일을 열 수 없음
- -102 LFH 파일 읽기에 실패함
- -103 올바르지 않은 LFH
- -104 데이터베이스를 복구할 수 없음
- -105 LSN이 너무 큼
- -500 논리적 오류

기타 유효한 리턴 코드는 다음과 같습니다.

- 0 성공적 실행
- 99 경고: 결과는 마지막으로 알려진 로그 파일 크기를 기초로 합니다.

input_LSN

선행 0이 사용되는 내부(6바이트) 16진수 값을 나타내는 12바이트 문자열.

예

```
db2flsn 000000BF0030
   Given LSN is contained in log file S0000002.LOG
db2flsn -q 000000BF0030
   S0000002.LOG
db2flsn 000000BE0030
   Warning: the result is based on the last known log file size.
   The last known log file size is 23 4K pages starting from log extent 2.
   Given LSN is contained in log file S0000001.LOG
db2flsn -q 000000BE0030
   S0000001.LOG
```

사용법 주

로그 헤더 제어 파일 sqlogctl.lfh는 현재 디렉토리에 있어야 합니다. 이 파일은 데이터베이스 디렉토리에 위치하므로 데이터베이스 디렉토리에서 도구를 수행할 수 있습니다. 또는 도구를 수행할 디렉토리에 제어 파일을 복사할 수 있습니다.

도구는 *logfilsiz* 데이터베이스 구성 매개변수를 사용합니다. DB2는 이 매개변수에 대한 최근 세 개의 값과 각각의 *logfilsiz* 값으로 작성된 첫 번째 로그 파일을 기록합니다. 그러면 *logfilsiz*가 변경될 때 도구가 올바르게 작동될 수 있습니다. 지정한 LSN이 기록된 최초 *logfilsiz* 값보다 이전이면, 도구는 이 값을 사용하며 경고를 리턴합니다. UDB 버전 5.2 이전의 데이터베이스 관리 프로그램에 대해 도구를 사용할 수 있습니다. 이 경우, 올바른 결과(*logfilsiz* 값이 변경되지 않은 경우에 확보되는)에 대해서도 경고가 리턴됩니다.

이 도구는 복구 가능한 데이터베이스에 대해서만 사용할 수 있습니다. 데이터베이스를 구성할 때 *logretain*을 RECOVERY로 설정하거나 *userexit*를 ON으로 설정할 경우 데이터베이스는 복구할 수 있습니다.

db2inidb - 이중복사 데이터베이스 초기화

분할 이중복사 환경에서, 이 명령은 다른 목적으로 이중복사된 데이터베이스를 초기화하는데 사용합니다.

권한 부여

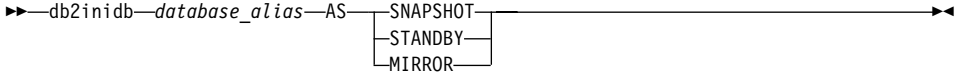
다음 중 하나를 수행하십시오.

- *sysadm*
- *sysctrl*
- *sysmaint*

필수 연결

없음

명령 구문



명령 매개변수

database_alias

초기화할 데이터베이스의 별명을 지정합니다.

SNAPSHOT

이중복사된 데이터베이스가 1차 데이터베이스의 복제본으로 초기화됨을 지정합니다. 이 데이터베이스는 읽기 전용입니다.

STANDBY

데이터베이스가 롤 포워드 보류 상태가 됨을 지정합니다. 1차 데이터베이스에서 새로운 로그는 페치하여 대기 데이터베이스에 적용할 수 있습니다. 다운될 경우, 대기 데이터베이스는 1차 데이터베이스 대신 사용할 수 있습니다.

MIRROR

이중복사된 데이터베이스가 1차 데이터베이스를 복원하기 위해 사용할 수 있는 백업 이미지로 사용됨을 지정합니다.

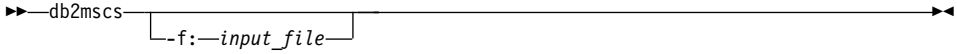
db2mscs - Windows NT 장애 복구 유틸리티 설정

MSCS(Microsoft Cluster Server)를 사용하여 Windows NT/2000에서 DB2 장애 복구 지원에 대한 내부 구조를 작성합니다. 이 유틸리티는 단일 파티션 데이터베이스 환경과 파티션된 데이터베이스 환경에서 장애 복구를 할 경우에 사용할 수 있습니다.

권한 부여

사용자는 MSCS 클러스터에 있는 각 머신의 관리자 그룹에 속하는 도메인 사용자 계정에 로그온해야 합니다.

명령 구문



명령 매개변수

-f:input_file

MSCS 유틸리티에서 사용될 DB2MSCS.CFG 입력 파일을 지정합니다. 이 매개변수를 지정하지 않으면, DB2MSCS 유틸리티는 현재 디렉토리에 있는 DB2MSCS.CFG 파일을 읽습니다.

ARCHIVE LOG

복구 가능한 데이터베이스에 대해 사용 중인 로그 파일을 닫아서 자릅니다. User Exit가 사용 가능하면 아카이브 요청을 발행합니다.

범위

MPP 환경에서 이 명령은 모든 노드에서 사용 중인 로그를 닫아서 자릅니다. 그러나 노드 부속 집합을 지정할 수 있습니다.

권한 부여

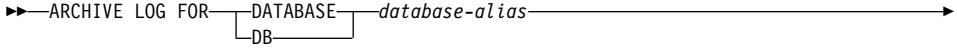
다음 중 하나를 수행하십시오.

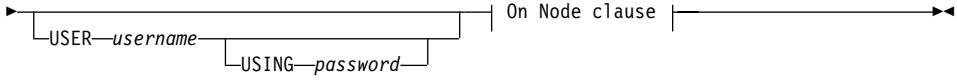
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

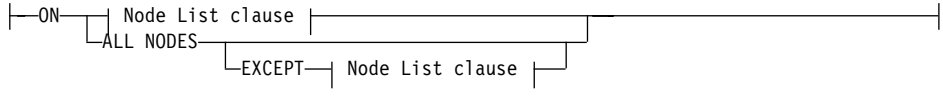
이 명령은 지정된 데이터베이스에 대한 연결을 자동으로 설정합니다. 이미 연결되어 있는 경우, 오류가 리턴됩니다.

명령 구문

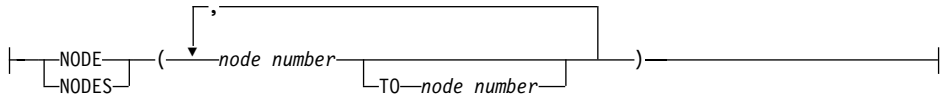




On Node clause:



Node List clause:



명령 매개변수

DATABASE database-alias

아카이브 로그가 아카이브될 데이터베이스의 별명을 지정합니다.

USER username

연결을 시도할 사용자 이름을 식별합니다.

USING password

사용자 이름을 인증할 암호를 지정합니다.

ON ALL NODES

명령이 db2nodes.cfg 파일의 모든 노드에서 발행되어야 함을 지정합니다. 이는 노드 절을 지정하지 않은 경우에 기본값입니다.

EXCEPT

노드 목록에 지정된 것을 제외하고, db2nodes.cfg 파일에 있는 모든 노드에서 명령을 발행해야 합니다.

ON NODE/ON NODES

노드 세트에서 지정된 데이터베이스에 대해 로그를 아카이브해야 함을 지정합니다.

node number

노드 목록에 노드 번호를 지정합니다.

끝 노드 번호

로그를 아카이브해야 하는 노드 범위를 지정할 때 사용됩니다. 지정된 첫 번째 노드 번호에서 지정된 두 번째 노드 번호까지의 모든 노드가 노드 목록에 포함됩니다.

사용법 주

이 명령은 알려진 지점까지의 완전한 로그 파일 세트를 수집하는데 사용할 수 있습니다. 그 로그 파일은 대기 데이터베이스를 갱신하는데 사용할 수 있습니다.

이 명령을 호출할 때 다른 응용프로그램에 진행 중인 트랜잭션이 있을 경우, 로그 버퍼의 내용을 디스크로 보내어 비울 때 성능이 떨어지는 것을 보게 될 것입니다. 로그 레코드를 버퍼에 쓰려고 하는 다른 트랜잭션은 비우기가 완료될 때까지 대기해야 합니다.

이 명령은 데이터베이스에서 LSN 공간의 부분이 유실되도록 하므로 유효한 LSN이 빨리 소모됩니다.

INITIALIZE TAPE

Windows NT용 DB2는 스트리밍 테이프 장치로의 백업 및 복원 조작을 지원합니다. 테이프를 초기화하려면 이 명령을 사용하십시오.

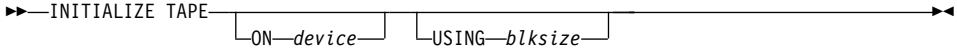
권한 부여

없음

필수 연결

없음

명령 구문



명령 매개변수

ON device

유효한 테이프 장치 이름을 지정하십시오. 기본값은 `\\.\TAPE0`입니다.

USING blksize

장치의 블록 *크기*를 바이트 단위로 지정하십시오. 값이 장치에 대해 지원되는 블록 *크기* 범위 내에 속할 경우, 장치는 지정된 블록 *크기*를 사용하도록 초기화됩니다.

주: 100 페이지의 『BACKUP DATABASE 명령』 및 131 페이지의 『RESTORE DATABASE 명령』에 지정된 버퍼 *크기*는 여기에 지정된 블록 *크기*에 의해 나눠질 수 있어야 합니다.

이 매개변수의 값을 지정하지 않으면, 장치를 기본 블록 *크기*를 사용하도록 초기화됩니다. 값으로 0을 지정할 경우, 장치는 가변 길이 블록 *크기*를 사용하도록 초기화됩니다. 장치가 가변 길이 블록 모드를 지원하지 않으면, 오류를 리턴합니다.

참조

375 페이지의『REWIND TAPE』

376 페이지의『SET TAPE POSITION』.

LIST HISTORY

실행기록 파일에 있는 항목을 나열합니다. 실행기록 파일에는 복구 및 관리 이벤트의 레코드가 있습니다. 복구 이벤트로는 전체 데이터베이스 및 테이블 공간 레벨 백업, 증분 백업, 복원 및 롤 포워드 조작이 있습니다. 로그된 추가 이벤트로는 테이블 공간 작성, 변경 또는 이름 변경, 통계 실행, 테이블 재구성, 테이블 삭제 및 로드가 있습니다.

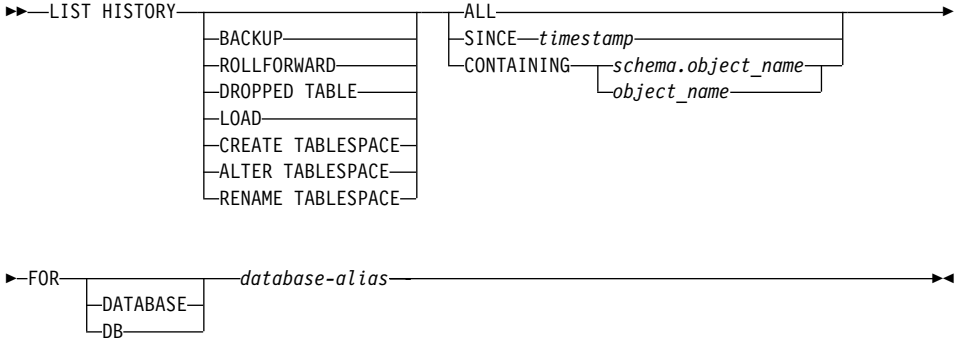
권한 부여

없음

필수 연결

인스턴스. 반드시 명시적으로 접속할 필요는 없습니다. 데이터베이스가 원격으로 나열되면, 원격 노드에 대한 인스턴스 접속이 명령 지속기간 동안 설정됩니다.

명령 구문



명령 매개변수

HISTORY

실행기록 파일에 현재 로그되어 있는 모든 이벤트를 나열합니다.

BACKUP

백업 및 복원 조작을 나열합니다.

ROLLFORWARD

롤 포워드 조작을 나열합니다.

DROPPED TABLE

삭제된 테이블 레코드를 나열합니다.

LOAD

로드 조작을 나열합니다.

CREATE TABLESPACE

테이블 공간 작성 및 제거 조작을 나열합니다.

RENAME TABLESPACE

테이블 공간 이름 바꾸기 조작을 나열합니다.

ALTER TABLESPACE

테이블 공간 변경 조작을 나열합니다.

ALL 실행기록 파일에 지정되어 있는 유형의 모든 항목을 나열합니다.

SINCE timestamp

완료 시간 소인(*yyyymmddhhnnss* 형식)이나 초기 접두부(최소 *yyyy*)를 지정할 수 있습니다. 시간소인이 제공된 시간소인과 같거나 그 이후인 모든 항목이 나열됩니다.

CONTAINING schema.object_name

이 규정된 이름은 테이블을 고유하게 식별합니다.

CONTAINING object_name

이 규정되지 않은 이름은 테이블 공간을 고유하게 식별합니다.

FOR DATABASE database-alias

복구 실행기록 파일이 나열되는 데이터베이스를 식별하는데 사용됩니다.

예

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

사용법 주

이 명령에 의해 생성되는 보고서에는 다음과 같은 기호가 포함됩니다.

조작
- A - 테이블 공간 작성
- B - 백업
- C - 로드 사본
- D - 삭제된 테이블
- F - 롤 포워드
- G - 테이블 재구성
- L - 로드
- N - 테이블 공간 이름 변경
- O - 테이블 공간 삭제
- Q - Quiesce
- R - 복원
- S - 통계 수행
- T - 테이블 공간 변경
- U - 언로드

유형

백업 유형
- F - 오프라인
- N - 온라인
- I - 증분 오프라인
- O - 증분 온라인
- D - 델타 오프라인
- E - 델타 온라인

롤 포워드 유형
- E - 로그 끝
- P - 특정 시점

로드 유형
- I - 삽입
- R - 바꾸기

테이블 공간 유형 변경
- C - 컨테이너 추가
- R - 재조정

Quiesce 유형
- S - Quiesce 상태 공유
- U - Quiesce 상태 갱신
- X - Quiesce 상태 독점
- Z - Quiesce 재설정

롤 포워드 조작이 모든 로그의 끝까지 수행된 경우, 백업 ID는 시간의 끝을 나타내며, 백업 ID 값은 99991231235959입니다.

PRUNE HISTORY/LOGFILE

복구 실행기록 파일에서 항목을 삭제하거나 사용 중인 로그 파일 경로에서 로그 파일을 삭제할 때 사용됩니다. 파일이 너무 커져서 보유 기간이 높아질 경우에는 복구 실행기록 파일에서 항목을 삭제해야 할 수도 있습니다. 로그를 수동으로(User Exit 프로그램을 통하지 않고) 아카이브할 경우에는 사용 중인 로그 파일 경로에서 로그 파일을 삭제해야 할 수도 있습니다.

권한 부여

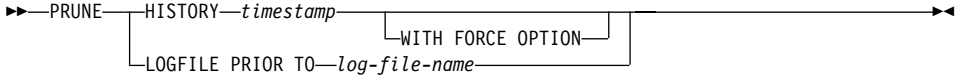
다음 중 하나를 수행하십시오.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

데이터베이스

명령 구문



명령 매개변수

HISTORY timestamp

복구 실행기록 파일에서 삭제할 항목 범위를 식별합니다. 완료 시간 소인(*yyyymmddhhmmss* 양식)이나 초기 접두부(최소 *yyyy*)를 지정할 수 있습니다. 시간소인이 제공된 시간소인과 같거나 그 이전인 모든 항목이 복구 실행기록 파일에서 삭제됩니다.

WITH FORCE OPTION

최근 복원 세트의 일부 항목이 파일에서 삭제된 경우에도 지정된 시간소인에 따라 항목이 제거되도록 지정합니다. 복원 세트는 해당되는 백업 이

미지의 복원을 포함하는 최근 전체 데이터베이스 백업입니다. 이 매개변수를 지정하지 않으면, 백업 이미지 포워드의 모든 항목이 실행기록에서 유지보수됩니다.

LOGFILE PRIOR TO log-file-name

로그 파일 이름에 대한 문자열을 지정합니다(예: S0000100.LOG). 지정된 로그 파일 이전의(포함하지 않고) 모든 로그 파일이 삭제됩니다. LOGRETAIN 데이터베이스 구성 매개변수는 RECOVERY 또는 CAPTURE로 설정해야 합니다.

예

복구 실행기록 파일에서 1994년 12월 1일을 포함하여 그 이전에 취해진 모든 복원, 로드, 테이블 공간 백업 및 전체 데이터베이스 백업을 제거하려면, 다음을 입력하십시오.

```
db2 prune history 199412
```

주: 199412는 19941201000000으로 해석됩니다.

사용법 주

실행기록 파일에서 백업 항목을 제거하면 DB2 Data Links Manager 서버에서 관련된 파일이 삭제됩니다.

REWIND TAPE

Windows NT용 DB2는 스트리밍 테이프 장치로의 백업 및 복원 조작을 지원합니다. 테이프를 되감기하려면 이 명령을 사용하십시오.

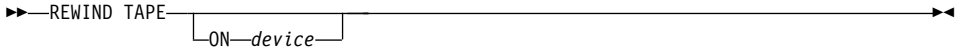
권한 부여

없음

필수 연결

없음

명령 구문



명령 매개변수

ON device

유효한 테이프 장치 이름을 지정하십시오. 기본값은 `\\.\TAPE0`입니다.

참조

368 페이지의『INITIALIZE TAPE』

376 페이지의『SET TAPE POSITION』

SET TAPE POSITION

Windows NT용 DB2는 스트리밍 테이프 장치로의 백업 및 복원 조작을 지원합니다. 테이프 위치 지정에 이 명령을 사용하십시오.

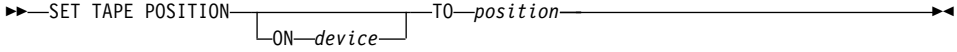
권한 부여

None

필수 연결

None

명령 구문



명령 매개변수

ON device

유효한 테이프 장치 이름을 지정하십시오. 기본값은 \\.\TAPE0입니다.

TO position

테이프가 위치될 표시를 지정합니다. Windows NT/2000용 DB2는 모든 백업 이미지 다음에 테이프 표시를 씁니다. 값 1은 첫 번째 위치를 지정하고, 2는 두 번째 위치를 지정합니다. 예를 들어, 테이프가 테이프 표시 1에 위치될 경우 아카이브 2가 복원되도록 위치됩니다.

참조

368 페이지의『INITIALIZE TAPE』

375 페이지의『REWIND TAPE』

UPDATE HISTORY FILE

실행기록 파일 항목에 있는 위치, 장치 유형 또는 주석을 갱신하십시오.

권한 부여

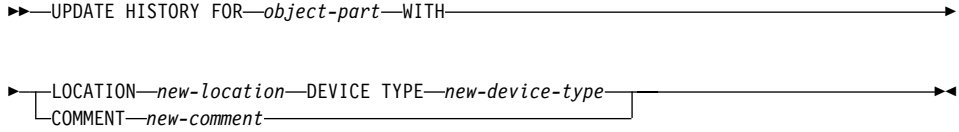
다음 중 하나를 수행하십시오.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

데이터베이스

명령 구문



명령 매개변수

FOR object-part

백업 및 복사 이미지에 대한 식별자를 지정합니다. 이것은 001 - 999 사이의 선택적 순차 번호가 있는 시간소인입니다.

LOCATION new-location

백업 이미지의 새로운 실제 위치를 지정합니다. 이 매개변수의 해석은 장치 유형에 따라 다릅니다.

DEVICE TYPE new-device-type

백업 이미지를 저장할 새로운 장치 유형을 지정합니다. 올바른 장치 유형은 다음과 같습니다.

D 디스크

K 디스켓

T 테이프

A TSM

U User Exit

O 기타

COMMENT new-comment

항목을 설명할 새로운 주석을 지정합니다.

예

1997년 4월 13일 오전 10:00에 취해진 전체 데이터베이스 백업에 대한 실행기록 파일 항목을 갱신하려면, 다음을 입력하십시오.

```
db2 update history for 19970413100000001 with
   location /backup/dbbackup.1 device type d
```

사용법 주

실행기록 파일은 레코드 보존을 위해 데이터베이스 관리자가 사용합니다. 이는 증분 백업의 자동 복구를 위해 내부적으로 DB2에서 사용됩니다.

참조

373 페이지의『PRUNE HISTORY/LOGFILE』

부록D. 추가 API 및 연관 데이터 구조

db2ArchiveLog - 아카이브 사용 중 로그 API

복구 가능한 데이터베이스에 대해 사용 중인 로그 파일을 닫아서 자릅니다. User Exit가 사용 가능하면 아카이브 요청을 발행합니다.

범위

MPP 환경에서 이 API는 모든 노드에서 사용 중인 로그를 닫아서 자릅니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

이 API는 지정된 데이터베이스에 대한 연결을 자동으로 설정합니다. 이미 연결되어 있는 경우, 오류가 리턴됩니다.

API 포함 파일

db2ApiDf.h

C API 구문

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2ArchiveLog (
      db2Uint32 version,
      void * pDB2ArchiveLogStruct,
      struct sqlca * pSqlca);
typedef struct
{
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct;
/* ... */
```

일반 API 구문

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2gArchiveLog (
      db2Uint32 version,
      void * pDB2gArchiveLogStruct,
      struct sqlca * pSqlca);
typedef struct
{
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2gArchiveLogStruct;
/* ... */
```

API 매개변수

버전 입력. 두 번째 매개변수 *pDB2ArchiveLogStruct*에서 전달되는 변수의 버전 및 릴리스 레벨을 지정합니다.

pDB2ArchiveLogStruct
입력. *db2ArchiveLogStruct* 구조에 대한 포인터

pSqlca
출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

iAliasLen
입력. 데이터베이스 별명의 길이를 바이트 단위로 표시하는 부호 없는 4 바이트 단위 정수

iUserNameLen

입력. 사용자 이름의 길이를 바이트 단위로 표시하는 부호 없는 4바이트 단위 정수. 사용자 이름이 사용되지 않았으면 0으로 설정하십시오.

iPasswordLen

입력. 암호의 길이를 바이트 단위로 표시하는 부호 없는 4바이트 단위 정수. 암호 사용되지 않았으면 0으로 설정하십시오.

piDatabaseAlias

입력. 사용 중인 로그를 아카이브할 데이터베이스의 데이터베이스 별명을 포함하는 문자열(시스템 데이터베이스 디렉토리에 카탈로그화된 것처럼)

piUserName

입력. 연결을 시도할 때 사용할 사용자 이름을 포함하는 문자열

piPassword

입력. 연결을 시도할 때 사용할 암호를 포함하는 문자열

iAllNodeFlag

입력. MPP 전용. `db2nodes.cfg` 파일에 나열된 모든 노드에 조작이 적용되어야 하는지를 표시하는 플래그. 유효한 값은 다음과 같습니다.

DB2ARCHIVELOG_ALL_NODES

모든 노드에 적용합니다. (piNodeList는 NULL이어야 합니다.) 이는 기본값입니다.

DB2ARCHIVELOG_NODE_LIST

*piNodeList*에서 전달되는 노드 목록에 지정된 모든 노드에 적용합니다.

DB2ARCHIVELOG_ALL_EXCEPT

*piNodeList*에서 전달되는 노드 목록에 지정된 노드를 *제외하고* 모든 노드에 적용합니다.

iNumNodes

입력. MPP 전용. *piNodeList* 배열에 노드 수를 지정합니다.

piNodeList

입력. MPP 전용. 아카이브 로그 조작을 적용할 노드 번호 배열에 대한 포인터

iOptions

입력. 차후 사용을 위해 예약됩니다.

사용법 주

이 API는 알려진 지점까지의 완전한 로그 파일 세트를 수집하는데 사용할 수 있습니다. 그 로그 파일은 대기 데이터베이스를 갱신하는데 사용할 수 있습니다.

이 API를 호출할 때 다른 응용프로그램에 진행 중인 트랜잭션이 있을 경우, 로그 버퍼의 내용을 디스크로 보내어 비울 때 성능이 떨어지는 것을 보게 될 것입니다. 로그 레코드를 버퍼에 쓰려고 하는 다른 트랜잭션은 비우기가 완료될 때까지 대기해야 합니다.

이 API는 데이터베이스에서 LSN 공간의 부분이 유실되도록 하므로 유효한 LSN이 빨리 소모됩니다.

db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기

복구 실행기록 파일 스캔을 종료하고 스캔에 필요한 DB2 자원을 해제합니다. 이 API 이전에 392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』에 대한 호출이 성공해야 합니다.

권한 부여

없음

필수 연결

인스턴스. 이 API를 호출하기 전에 **sqleatin**을 호출할 필요는 없습니다.

API 포함 파일

db2ApiDf.h

C API 구문

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

일반 API 구문

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2GenHistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

API 매개변수

버전 입력. 두 번째 매개변수 *piHandle*에서 전달되는 구조의 버전 및 릴리스 레벨을 지정합니다.

piHandle

입력. 392 페이지의 『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』에서 리턴된 스캔 액세스의 핸들에 대한 포인터를 지정합니다.

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

REXX API 구문

```
CLOSE RECOVERY HISTORY FILE :scanid
```

REXX API 매개변수

scanid OPEN RECOVERY HISTORY FILE SCAN에서 리턴된 스캔 식별자를 포함하는 호스트 변수

사용법 주

복구 실행기록 파일 API 사용에 대한 자세한 설명은
392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』

참조

388 페이지의『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』

392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』

404 페이지의『db2Prune API』

399 페이지의『db2HistoryUpdate - 복구 실행기록 파일 갱신』

db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기

복구 실행기록 파일에서 다음 항목을 가져옵니다. 이 API 이전에 392 페이지의 『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』에 대한 호출이 성공해야 합니다.

권한 부여

없음

필수 연결

인스턴스. 이 API를 호출하기 전에 **sqleatin**을 호출할 필요는 없습니다.

API 포함 파일

db2ApiDf.h

C API 구문

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void * pDB2HistoryGetEntryStruct,
    struct sqlca * pSqlca);
typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

일반 API 구문

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryGetEntry (
    db2Uint32 version,
    void * pDB2GenHistoryGetEntryStruct,
    struct sqlca * pSqlca);
typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2GenHistoryGetEntryStruct;
/* ... */
```

API 매개변수

버전 입력. 두 번째 매개변수 *pDB2HistoryGetEntryStruct*에서 전달되는 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2HistoryGetEntryStruct

입력. *db2HistoryGetEntryStruct* 구조에 대한 포인터

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

iHandle

입력. 392 페이지의 『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』에서 리턴된 스캔 액세스에 대한 핸들이 있습니다.

iCallerAction

입력. 취할 조치 유형을 지정합니다. 올바른 값은 다음과 같습니다 (db2ApiDf에 정의됨).

DB2HISTORY_GET_ENTRY

다음 항목을 가져오지만, 명령 데이터는 없습니다.

DB2HISTORY_GET_DDL
이전 페치에서 명령 데이터만 가져옵니다.

DB2HISTORY_GET_ALL
모든 데이터를 포함하여 다음 항목을 가져옵니다.

pioHistData
입력. *db2HistData* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 413 페이지의 『데이터 구조: db2HistData』를 참조하십시오.

REXX API 구문

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

REXX API 매개변수

scanid OPEN RECOVERY HISTORY FILE SCAN에서 리턴된 스캔 식별자를 포함하는 호스트 변수

value 복구 실행기록 파일 정보가 리턴되는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0 변수에 있는 첫 번째 레벨 요소 수(항상 15)

XXX.1 테이블 공간 요소 수

XXX.2 사용되는 테이블 공간 요소 수

XXX.3 OPERATION(수행된 조작 유형)

XXX.4 OBJECT(조작의 세분성)

XXX.5 OBJECT_PART(시간소인 및 순차 번호)

XXX.6 OPTYPE(조작의 규정자)

XXX.7 DEVICE_TYPE(사용된 장치 유형)

XXX.8 FIRST_LOG(맨 처음 로그 ID)

XXX.9 LAST_LOG(현재 로그 ID)

XXX.10 BACKUP_ID(백업을 위한 식별자)

XXX.11 SCHEMA(테이블 이름을 위한 식별자)

XXX.12 TABLE_NAME(로드된 테이블의 이름)

XXX.13.0 NUM_OF_TABLESPACES(백업 또는 복원에 연관된 테이블 공간의 수)

XXX.13.1 백업/복원한 첫 번째 테이블 공간의 이름

XXX.13.2 백업/복원한 두 번째 테이블 공간의 이름

XXX.13.3 기타

XXX.14 LOCATION(백업 또는 복사본이 저장되는 장소)

XXX.15 COMMENT(항목을 설명하는 텍스트)

사용법 주

리턴되는 레코드는 392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』에 대한 호출에 지정된 값을 사용하여 선택됩니다.

복구 실행기록 파일 API 사용에 대한 자세한 설명은
392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』를 참조하십시오.

참조

385 페이지의『db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기』

392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』

404 페이지의『db2Prune API』

399 페이지의『db2HistoryUpdate - 복구 실행기록 파일 갱신』

db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기

복구 실행기록 파일 스캔을 시작합니다.

권한 부여

없음

필수 연결

인스턴스. 이 API를 호출하기 전에 **sqleatin**을 호출할 필요는 없습니다. 데이터베이스가 원격으로 카탈로그화되면, 원격 노드에 대한 인스턴스 접속이 설정됩니다.

API 포함 파일

db2ApiDf.h

C API 구문

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void * pDB2HistoryOpenStruct,
    struct sqlca * pSqlca);
typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

일반 API 구문

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void * pDB2GenHistoryOpenStruct,
    struct sqlca * pSqlca);
typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2GenHistoryOpenStruct;
/* ... */
```

API 매개변수

버전 입력. 두 번째 매개변수 *pDB2HistoryOpenStruct*에서 전달되는 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2HistoryOpenStruct

입력. *db2HistoryOpenStruct* 구조에 대한 포인터

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

piDatabaseAlias

입력. 데이터베이스 별명을 포함하는 문자열에 대한 포인터

piTimestamp

입력. 레코드 선택에 사용할 시간소인을 지정하는 문자열에 대한 포인터 시

간소인이 이 값과 같거나 그 이후인 레코드들이 선택됩니다. 이 매개변수를 NULL로 설정하거나 0을 지시할 경우 시간소인을 사용한 항목 필터링이 금지됩니다.

piObjectName

입력. 레코드 선택에 사용할 오브젝트 이름을 지정하는 문자열에 대한 포인터 오브젝트는 테이블 또는 테이블 공간이 됩니다. 테이블이면 완전한 테이블 이름을 제공해야 합니다. 이 매개변수를 NULL로 설정하거나 0을 지시할 경우 오브젝트 이름을 사용한 항목 필터링이 금지됩니다.

oNumRows

출력. API에서 리턴되는 대로, 이 매개변수에는 일치하는 복구 실행기록 파일 항목 수가 있습니다.

iCallerAction

입력. 취할 조치 유형을 지정합니다. 올바른 값은 다음과 같습니다 (db2ApiDf에 정의됨).

DB2HISTORY_LIST_HISTORY

현재 실행기록 파일에 로깅된 모든 이벤트를 나열합니다.

DB2HISTORY_LIST_BACKUP

백업 및 복원 조작을 나열합니다.

DB2HISTORY_LIST_ROLLFORWARD

롤 포워드 조작을 나열합니다.

DB2HISTORY_LIST_DROPPED_TABLE

삭제된 테이블 레코드를 나열합니다. 항목과 연관되는 DDL 필드는 리턴되지 않습니다. 항목에 대한 DDL 정보를 검색하려면, 항목이 페치된 다음에 즉시 호출자 조치 DB2HISTORY_GET_DDL을 사용하여 388 페이지의 『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』를 호출해야 합니다.

DB2HISTORY_LIST_LOAD

로드 조작을 나열합니다.

DB2HISTORY_LIST_CRT_TABLESPACE

테이블 공간 작성 및 제거 조작을 나열합니다.

DB2HISTORY_LIST_REN_TABLESPACE
테이블 공간 이름 바꾸기 조작을 나열합니다.

DB2HISTORY_LIST_ALT_TABLESPACE
테이블 공간 변경 조작을 나열합니다. 항목과 연관되는 DDL 필드는 리턴되지 않습니다. 항목에 대한 DDL 정보를 검색하려면, 항목이 페치된 다음에 즉시 호출자 조치 DB2HISTORY_GET_DDL을 사용하여 388 페이지의 『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』를 호출해야 합니다.

DB2HISTORY_LIST_RUNSTATS
RUNSTATS 조작을 나열합니다. 이 값은 현재 지원되지 않습니다.

DB2HISTORY_LIST_REORG
REORGANIZE TABLE 조작을 나열합니다. 이 값은 현재 지원되지 않습니다.

oHandle
출력. API에서 리턴되는 대로, 이 매개변수에는 스캔 액세스에 대한 핸들이 있습니다. 이 값은 388 페이지의 『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』 및
385 페이지의 『db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기』에 뒤이어 사용됩니다.

REXX API 구문

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

REXX API 매개변수

database_alias
실행기록 파일이 나열되는 데이터베이스의 별명

objname

레코드 선택에 사용할 오브젝트 이름을 지정합니다. 오브젝트는 테이블 또는 테이블 공간이 됩니다. 테이블이면 완전한 테이블 이름을 제공해야 합니다. 이 매개변수를 NULL로 설정하면 *objname*을 사용한 항목 필터링이 금지됩니다.

timestamp

레코드 선택에 사용할 시간소인을 지정합니다. 시간소인이 이 값과 같거나 그 이후인 레코드들이 선택됩니다. 이 매개변수를 NULL로 설정하면 *timestamp*을 사용한 항목 필터링이 금지됩니다.

value 복구 실행기록 파일 정보가 리턴될 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0

변수에 있는 요소의 수(항상 2)

XXX.1

나중 스캔 액세스를 위한 식별자(핸들)

XXX.2

일치하는 복구 실행기록 파일 항목 수

사용법 주

레코드를 거르기 위해 시간소인, 오브젝트 이름 및 호출자 조치를 조합하여 사용할 수 있습니다. 지정된 모든 필터를 패스하는 레코드만 리턴됩니다.

오브젝트 이름의 필터링 효과는 지정된 값에 따라 다릅니다.

- 테이블을 지정하면 로드 조작에 대한 레코드를 리턴합니다. 이는 실행기록 파일에서 테이블에 대한 유일한 정보이기 때문입니다.
- 테이블 공간을 지정하면 테이블 공간의 백업, 복원 및 로드 조작에 대한 레코드를 리턴합니다.

주: 테이블에 대한 레코드를 리턴하려면, 테이블이 *schema.tablename*으로 지정되어야 합니다. *tablename*을 지정하면 테이블 공간에 대한 레코드만 리턴합니다.

프로세스당 최소 8개의 실행기록 파일 스캔이 허용됩니다.

실행기록 파일의 모든 항목을 나열하기 위해 일반 응용프로그램이 다음 단계를 수행합니다.

1. **db2HistoryOpenScan**을 호출하면, *oNumRows*를 리턴하게 됩니다.
2. *n*개의 *oTablespace* 필드에 대한 공간이 있는 *db2HistData* 구조를 할당합니다. 여기서 *n*은 임시 숫자입니다.
3. *db2HistData* 구조의 *iDB2NumTablespace* 필드를 *n*으로 설정합니다.
4. 루프에서 다음을 수행합니다.
   - **db2HistoryGetEntry**를 호출하여 실행기록 파일에서 페치합니다.
   - **db2HistoryGetEntry**가 SQLCODE SQL_RC_OK를 리턴할 경우, *db2HistData* 구조의 *sqld* 필드를 사용하여 리턴된 테이블 공간 수를 판별합니다.
   - **db2HistoryGetEntry**가 SQLCODE SQLUH_SQLUHINFO_VARS_WARNING을 리턴할 경우, DB2가 리턴하려고 하는 모든 테이블 공간에 대해 충분한 공간이 할당되지 않은 것입니다. *oDB2UsedTablespace* 테이블 공간 항목에 대해 충분한 공간이 있는 *db2HistData* 구조를 다시 할당하고 *iDB2NumTablespace*를 *oDB2UsedTablespace*로 설정합니다.
   - **db2HistoryGetEntry**가 SQLCODE SQLE_RC_NOMORE를 리턴할 경우, 모든 복구 실행기록 파일 항목이 검색된 것입니다.
   - 다른 SQLCODE는 문제점을 표시합니다.
5. 모든 정보가 페치되었으면, 385 페이지의 『db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기』를 호출하여 **db2HistoryOpenScan**에 대한 호출에서 할당된 자원들을 사용할 수 있도록 해제하십시오.

sqlutil에서 정의된 SQLUHINFOSIZE(*n*) 매크로는 *n oTablespace* 필드에 대한 공간이 있는 *db2HistData* 구조에 필요한 메모리의 양을 판별하는데 도움을 주기 위해 제공됩니다.

참조

385 페이지의 『db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기』

388 페이지의『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』

404 페이지의『db2Prune API』

399 페이지의『db2HistoryUpdate - 복구 실행기록 파일 갱신』

db2HistoryUpdate - 복구 실행기록 파일 갱신

실행기록 파일 항목에서 위치, 장치 유형 또는 주석을 갱신합니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

데이터베이스. 기본 데이터베이스가 아닌 다른 데이터베이스에 대해 실행기록 파일에서 항목을 갱신하려면, 이 API를 호출하기 전에 데이터베이스에 대한 연결을 설정해야 합니다.

API 포함 파일

db2ApiDf.h

C API 구문

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void * pDB2HistoryUpdateStruct,
    struct sqlca * pSqlca);
typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2HistoryUpdateStruct;
/* ... */
```

일반 API 구문

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void * pDB2GenHistoryUpdateStruct,
    struct sqlca * pSqlca);
typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2GenHistoryUpdateStruct;
/* ... */
```

API 매개변수

버전 입력. 두 번째 매개변수 *pDB2HistoryUpdateStruct*에서 전달되는 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2HistoryUpdateStruct
입력. *db2HistoryUpdateStruct* 구조에 대한 포인터

pSqlca
출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

piNewLocation
입력. 백업, 복원 또는 로드 사본 이미지의 새 위치를 지정하는 문자열에 대한 포인터. 이 매개변수를 NULL로 설정하거나 0을 지시할 경우, 값은 변경되지 않은 상태로 남습니다.

piNewDeviceType
입력. 백업, 복원 또는 로드 사본 이미지를 저장할 새로운 장치 유형을 지정하는 문자열에 대한 포인터. 이 매개변수를 NULL로 설정하거나 0을 지시할 경우 값은 변경되지 않은 상태로 남습니다.

piNewComment
입력. 항목을 설명할 새로운 주석을 지정하는 문자열에 대한 포인터. 이 매개변수를 NULL로 설정하거나 0을 지시할 경우 주석은 변경되지 않은 상태로 남습니다.

iEID 입력. 실행기록 파일에서 특정 항목을 갱신하기 위해 사용할 수 있는 고유 식별자

REXX API 구문

```
UPDATE RECOVERY HISTORY USING :value
```

REXX API 매개변수

value 복구 실행기록 파일 항목의 새로운 위치에 관한 정보를 포함하는 복합 REXX 호스트 변수. 다음의 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0

변수의 요소 수(1과 4 사이)

XXX.1

OBJECT_PART(순차 번호가 001 - 999인 시간소인)

XXX.2

백업 또는 복사 이미지에 대한 새로운 위치(이 매개변수는 선택적임)

XXX.3

백업 또는 복사 이미지를 저장할 새로운 장치(이 매개변수는 선택적임)

XXX.4

새로운 주석(이 매개변수는 선택적임)

사용법 주

이는 갱신 기능으로, 변경 이전의 모든 정보가 바뀌어서 재작성할 수 없습니다. 이 변경사항은 로그되지 않습니다.

실행기록 파일은 레코딩 목적으로만 사용됩니다. 복원 또는 롤 포워드 기능에서는 직접 사용되지 않습니다. 복원 조작 동안 백업 이미지 위치를 지정할 수 있으며, 실행기록 파일은 이 위치를 추적하는데 유용합니다. 정보는 나중에 백업 유틸리티에 제공될 수 있습니다(105 페이지의 『데이터베이스 API 백업』 참조). 마찬가지로, 로드 사본 이미지의 위치가 이동되면 새로운 위치와 저장 미디어 유형과 함께 롤 포워드 유틸리티를 제공해야 합니다. 추가 정보는 175 페이지의 『데이터베이스 API 롤 포워드』를 참조하십시오.

참조

385 페이지의 『db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기』

388 페이지의『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』

392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』

404 페이지의『db2Prune API』를 참조하십시오.

db2Prune API

사용 중인 로그 경로에서 복구 실행기록 파일이나 로그 파일을 삭제합니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

데이터베이스. 기본 데이터베이스가 아닌 다른 데이터베이스에 대해 복구 실행기록 파일에서 항목을 삭제하려면, 이 API를 호출하기 전에 데이터베이스에 대한 연결을 설정해야 합니다.

API 포함 파일

db2ApiDf.h

C API 구문

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2Prune (
    db2Uint32 version,
    void * pDB2PruneStruct,
    struct sqlca * pSqlca);
typedef struct
{
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2PruneStruct;
/* ... */
```

일반 API 구문

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenPrune (
    db2Uint32 version,
    void * pDB2GenPruneStruct,
    struct sqlca * pSqlca);
typedef struct
{
  db2Uint32 iStringLen;
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2GenPruneStruct;
/* ... */
```

API 매개변수

버전 입력. 두 번째 매개변수 *pDB2PruneStruct*에서 전달되는 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2PruneStruct
입력. *db2PruneStruct* 구조에 대한 포인터

pSqlca
출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

iStringLen
입력. *piString* 바이트 길이로 지정하십시오.

piString
입력. 시간소인이나 로그 순차 번호(LSN)를 지정하는 문자열에 대한 포인터. 시간소인이나 시간소인의 일부(최소 *yyyy*, 또는 연도)가 삭제할 레코드 선택에 사용됩니다. 시간소인이 같거나 그 이전에 해당되는 모든 항목이 삭제됩니다. 유효한 시간소인을 제공해야 합니다. NULL 매개변수에 대한 기본 작동은 없습니다.

이 매개변수를 사용하여 LSN을 전달할 수도 있습니다. 그러면 비활동 로그를 제거할 수 있습니다.

iEID 입력. 실행기록 파일에서 단일 항목을 제거하기 위해 사용할 수 있는 고유 식별자를 지정합니다.

iCallerAction
입력. 취할 조치 유형을 지정합니다. 올바른 값은 다음과 같습니다 (db2ApiDf에 정의됨).

DB2PRUNE_ACTION_HISTORY
실행기록 파일 항목을 삭제합니다.

DB2PRUNE_ACTION_LOG
사용 중인 로그 경로에서 로그 파일을 제거합니다.

iOptions
입력. 올바른 값은 다음과 같습니다(db2ApiDf에 정의됨).

DB2PRUNE_OPTION_FORCE

마지막 백업 제거를 강요합니다.

DB2PRUNE_OPTION_LSNSTRING

piString 값이 LSN임을 지정합니다. 이는 DB2PRUNE_ACTION_LOG의 호출자 조치를 지정할 때 사용됩니다.

REXX API 구문

```
PRUNE RECOVERY HISTORY BEFORE :timestamp [WITH FORCE OPTION]
```

REXX API 매개변수

timestamp

시간소인을 포함하는 호스트 변수. 시간소인이 제공된 시간소인과 같거나 그 이전인 모든 항목이 복구 실행기록 파일에서 삭제됩니다.

WITH FORCE OPTION

지정한 경우, 최근 복원 세트의 일부 항목이 파일에서 삭제된 경우에도 지정된 시간소인에 따라 복구 실행기록 파일이 제거됩니다. 지정하지 않으면, 시간소인이 입력으로 지정된 시간소인이나 그 이전이어도 최근 복원 세트가 보존됩니다.

사용법 주

실행기록 파일을 제거하여도 실제 백업 또는 로드 파일은 삭제되지 않습니다. 사용자는 이 파일들을 수동으로 삭제하여 저장 미디어에서 차지하고 있는 공간을 비워야 합니다.

주의:

(실행기록 파일에서 제거되는 것 외에도) 최근 전체 데이터베이스 백업이 매체에서 삭제된 경우, 사용자는 카탈로그 테이블 공간과 사용자 테이블 공간을 포함하여 모든 테이블 공간이 백업되도록 해야 합니다. 그렇게 하지 못하면 데이터베이스를 복구할 수 없거나, 데이터베이스에서 사용자 데이터의 일부분이 유실될 수 있습니다.

참조

385 페이지의『db2HistoryCloseScan - 복구 실행기록 파일 스캔 API 닫기』

388 페이지의『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』

392 페이지의『db2HistoryOpenScan - 복구 실행기록 파일 스캔 API 열기』

399 페이지의『db2HistoryUpdate - 복구 실행기록 파일 갱신』

sqlurlog - 비동기 읽기 로그 API

호출자에게 데이터베이스 로그로부터 특정의 로그 레코드를 추출하고, 현재 로그 상태 정보에 맞는 로그 관리 프로그램을 조회할 수 있는 기능을 제공합니다. 이 API는 복구 가능한 데이터베이스에 대해서만 사용할 수 있습니다. 데이터베이스를 구성할 때 *logretain*을 RECOVERY로 설정하거나, *userexit*를 ON으로 설정할 경우 데이터베이스는 복구할 수 있습니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *dbadm*

필수 연결

데이터베이스

API 포함 파일

sqlutil.h

C API 구문

```
/* File: sqlutil.h */
/* API: Asynchronous Read Log */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurlog (
    sqluint32 CallerAction,
    SQLU_LSN * pStartLsn,
    SQLU_LSN * pEndLsn,
    char * pLogBuffer,
    sqluint32 LogBufferSize,
    SQLU_RLOG_INFO * pReadLogInfo,
    struct sqlca * pSqlca);
/* ... */
```

API 매개변수

CallerAction

입력. 수행할 조치를 지정하십시오.

SQLU_RLOG_READ

시작 로그 순차 번호에서 끝 로그 순차 번호까지 데이터베이스 로그를 읽어서 이 범위 내의 모든 보급 가능한 로그 레코드를 리턴합니다.

SQLU_RLOG_READ_SINGLE

시작 로그 순차 번호에 의해 식별되는 단일 로그 레코드(보급 가능하거나 그렇지 않은)를 읽습니다.

SQLU_RLOG_QUERY

데이터베이스 로그 조회. 조회 결과는 SQLU_RLOG_INFO 구조를 통해 보냅니다(420 페이지의 『데이터 구조: SQLU-RLOG-INFO』 참조).

pStartLsn

입력. 시작 로그 순차 번호는 로그 읽기에 대한 시작 상대 바이트 주소를 지정합니다. 이 값은 실제 로그 레코드의 시작이어야 합니다.

pEndLsn

입력. 끝 로그 순차 번호는 로그 읽기에 대한 끝 상대 바이트 주소를 지정합니다. 이 값은 *startLsn*보다 커야 하며 실제 로그 레코드의 끝일 필요는 없습니다.

pLogBuffer

출력. 지정된 범위 내에서 읽혀진 모든 보급 가능한 로그 레코드가 순차적으로 저장되는 버퍼. 이 버퍼는 단일 로그 레코드를 보유할 수 있을만큼 충분히 커야 합니다. 지침에 따르면 이 버퍼는 최소 32바이트여야 합니다. 최대 *크기*는 요청 범위의 *크기*에 따라 결정됩니다. 버퍼의 각 로그 레코드 앞에는 다음 로그 레코드의 LSN을 나타내는 6바이트 로그 순차 번호(LSN)가 접두부로 추가됩니다.

LogBufferSize

출력. 로그 버퍼의 *크기*(바이트)를 지정합니다.

pReadLogInfo

출력. 호출 및 데이터베이스 로그에 대한 세부사항 정보를 보여주는 구조. 이 구조에 대한 자세한 정보는 420 페이지의 『데이터 구조: SQLU-RLOG-INFO』를 참조하십시오.

pSqlca

출력. *sqlca* 구조에 대한 포인터. 이 구조에 대한 자세한 정보는 *Administrative API Reference* 또는 *SQL 참조서*를 참조하십시오.

샘플 프로그램

C \sqllib\samples\c\asynrlog.sqc

사용법 주

요청한 조치가 로그를 읽는 것이면, 호출자는 로그 레코드를 보유할 버퍼와 로그 순차 번호 범위를 제공합니다. 이 API는 순차적으로 로그를 읽어서(요청된 LSN 범위를 경계로 하여) DATA CAPTURE 옵션이 CHANGES이고 SQLU_RLOG_INFO 구조에 현재 사용 중인 로그 정보가 있는 테이블과 연관되는 로그 레코드들을 리턴합니다. 요청한 조치가 조회일 경우, API는 현재 사용 중인 로그 정보가 있는 SQLU_RLOG_INFO 구조를 리턴합니다.

비동기 로그 판독기를 사용하려면, 먼저 유효한 시작 LSN에 대해 데이터베이스 로그를 조회하십시오. 조회 호출 후에는 읽기 로그 정보 구조(SQLU-RLOG-INFO)에는 유효한 시작 LSN(initialLSN 구성원에 있는)이 포함됩니다. 이는 읽기 호출에 사용됩니다. 현재 사용 중인 로그의 끝은 읽기 로그 정보 구조의 curActiveLSN 구성원에 위치됩니다. 읽기에서 끝 LSN으로 사용되는 값은 다음 중 하나가 될 수 있습니다.

- curActiveLSN의 값
- initialLSN보다 큰 값
- FFFF FFFF FFFF: 비동기 로그 판독기에 의해 현재 로그의 끝으로 해석됩니다.

읽기 로그 정보 구조에 대한 자세한 정보는 420 페이지의 『데이터 구조: SQLU-RLOG-INFO』를 참조하십시오.

시작 및 끝 LSN 범위 내에서 읽혀진 보급 가능한 로그 레코드가 로그 버퍼에서 리턴됩니다. 로그 레코드에는 LSN이 포함되지 않습니다. 이는 실제 로그 레코드 이전에 버퍼에 포함됩니다. **sqlurlog**에서 리턴되는 다양한 DB2 로그 레코드에 대한 설명은 *Administrative API Reference*를 참조하십시오.

초기 읽기 이후에 다음 순차 로그 레코드를 읽으려면, SQLU-RLOG-INFO에서 리턴된 마지막 읽기 LSN에 1을 추가하십시오. 이 새로운 시작 LSN과 유효한 끝 LSN을 사용하여 호출을 다시 제출하십시오. 그러면 다음 레코드 블록이 읽혀집니다. SQLU_RLOG_READ_TO_CURRENT의 *sqlca*는 로그 판독기가 현재 사용중인 로그의 끝까지 읽었음을 의미합니다.

데이터 구조: db2HistData

이 구조는 388 페이지의 『db2HistoryGetEntry - 다음 복구 실행기록 파일 항목 API 가져오기』 호출 후에 정보를 리턴하기 위해 사용됩니다.

표 17. db2HistData 구조

필드 이름	데이터 유형	설명
ioHistDataID	char(8)	저장영역 덤프를 위한 8바이트 구조 식별자 및 『eye-catcher』. 올바른 값은 『SQLUHINF』뿐입니다. 이 문자열에 대한 기호 링크는 없습니다.
oObjectPart	db2Char	처음 14자는 *yyyymmddhhnnss* 형식의 시간소인으로, 이는 조작을 시작한 시기를 나타냅니다. 다음 3 자는 순차 번호입니다. 각 백업 조작은 백업 이미지가 여러 파일이나 여러 테이프에 저장될 경우 이 파일에 여러 항목이 생성되도록 합니다. 순차 번호를 사용하여 여러 위치를 지정할 수 있습니다. 복원 및 로드 조작은 이 파일에서 해당 백업의 순차 번호 '001'에 해당되는 단일 항목만 가지고 있습니다. 순차 번호와 결합된 시간소인은 고유해야 합니다.
oEndTime	db2Char	조작이 완료된 시기를 나타내는 *yyyymmddhhnnss* 형식의 시간소인
oFirstLog	db2Char	맨 처음 로그 파일 ID(S0000000 - S9999999 범위) • 온라인 백업에 대한 롤 포워드 복구를 적용하기 위해 필요합니다. • 오프라인 백업에 대한 롤 포워드 복구를 적용하기 위해 필요합니다. • 로드를 시작하였을 당시에 있었던 전체 데이터베이스 또는 테이블 공간 레벨 백업을 복원한 후에 적용됩니다.
oLastLog	db2Char	맨 나중 로그 파일 ID(S0000000 - S9999999 범위) • 온라인 백업에 대한 롤 포워드 복구를 적용하기 위해 필요합니다. • 오프라인 백업에 대한 현재 특정 시점에 롤 포워드 복구를 적용하기 위해 필요합니다. • 로드 조작을 완료하였을 당시에 있었던 전체 데이터베이스 또는 테이블 공간 레벨 백업을 복원한 후에 적용됩니다. (롤 포워드 복구를 적용하지 않을 경우에는 *oFirstLog*와 같게 됩니다.)

표 17. db2HistData 구조 (계속)

필드 이름	데이터 유형	설명
oID	db2Char	고유한 백업 또는 테이블 식별자
oTableQualifier	db2Char	테이블 규정자
oTableName	db2Char	테이블 이름
oLocation	db2Char	백업 및 로드 사본의 경우, 이 필드는 데이터가 저장된 곳을 표시합니다. 파일에서 여러 항목을 필요로 하는 조작의 경우, *oObjectPart*에 위해 정의한 순차 번호는 지정된 위치에서 발견되는 백업 파트를 식별합니다. 복원 및 로드 조작의 경우, 위치는 항상 복원되거나 로드된 데이터의 첫 번째 파트(여러 파트 백업의 경우에는 순서 '001'에 해당되는)가 저장된 곳을 식별합니다. *oLocation*의 데이터는 *oDeviceType*에 따라 달리 해석됩니다. • 디스크 또는 디스켓용(D 또는 K), 완전한 파일 이름 • 테이프용(T), 볼륨 레이블 • TSM용(A), 서버 이름 • User Exit 또는 기타(U 또는 0), 자유 양식 텍스트
oComment	db2Char	자유 양식 텍스트 주석
oCommandText	db2Char	명령 텍스트 또는 DDL
oLastLSN	SQLU_LSN	마지막 로그 순차 번호
oEID	구조	고유한 항목 식별자
poEventSQLCA	구조	기록된 이벤트의 결과 *sqlca*. *sqlca* 구조에 대한 정보는 *Administrative API Reference* 및 *SQL 참조서*에서 참조하십시오.
poTablespace	db2Char	테이블 공간 이름의 목록
ioNumTablespaces	db2Uint32	*poTablespace* 목록의 항목 수. 각 테이블 공간 백업에는 하나 이상의 테이블 공간이 있습니다. 각 테이블 공간 복원 조작은 하나 이상의 테이블 공간을 바꿉니다. 이 필드가 0(테이블 공간 레벨 백업 또는 복원을 나타냄)이 아니면, 이 파일에서 다음 행에 백업 또는 복원된 테이블 공간의 이름이 18자 문자열로 표시되어 포함됩니다. 행마다 하나의 테이블 공간 이름이 표시됩니다.
oOperation	char	415 페이지의 표18에서 자세한 내용을 참조하십시오.
oObject	char	조작의 세분성: 전체 데이터베이스의 경우 D, 테이블 공간의 경우 P, 테이블의 경우 T
oOptype	char	416 페이지의 표19에서 자세한 내용을 참조하십시오.

표 17. db2HistData 구조 (계속)

필드 이름	데이터 유형	설명
oStatus	char	항목 상태: 삭제된 경우 D(나중 사용), 만기된 경우 E, 사용 중이지 않을 경우 I, 아직 확약되지 않은 경우 N, 확약되거나 사용 중일 경우 Y
oDeviceType	char	장치 유형. 이 필드는 *oLocation* 필드의 해석 방법을 판별합니다. TSM의 경우 A, 클라이언트의 경우 C, 디스크의 경우 D, 디스켓의 경우 K, 지역의 경우 L, 기타의 경우 O(기타 벤더 장치 지원의 경우), 파이프의 경우 P, 서버의 경우 S, 테이프의 경우 T, User Exit의 경우 U

표 18. db2HistData 구조 내의 올바른 oOperation 값

값	설명	C 정의	COBOL/FORTRAN 정의
A	테이블 공간 추가	DB2HISTORY_OP_ADD_TABLESPACE	DB2HIST_OP_ADD_TABLESPACE
B	백업	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	로드 사본	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	삭제된 테이블	DB2HISTORY_OP_DROPPED_TABLE	DB2HIST_OP_DROPPED_TABLE
F	롤 포워드	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	테이블 재구성	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
L	로드	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	테이블 공간 이름 변경	DB2HISTORY_OP_REN_TABLESPACE	DB2HIST_OP_REN_TABLESPACE
O	테이블 공간 삭제	DB2HISTORY_OP_DROP_TABLESPACE	DB2HIST_OP_DROP_TABLESPACE
Q	quiesce	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	복원	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
S	통계 수행	DB2HISTORY_OP_RUNSTATS	DB2HIST_OP_RUNSTATS
T	테이블 공간 변경	DB2HISTORY_OP_ALT_TABLESPACE	DB2HIST_OP_ALT_TBS
U	언로드	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

표 19. db2HistData 구조 내의 올바른 oOptype 값

oOperation	oOptype	설명	C/COBOL/FORTRAN 정의
B	F	오프라인	DB2HISTORY_OPTYPE_OFFLINE
	N	온라인	DB2HISTORY_OPTYPE_ONLINE
	I	증분 오프라인	DB2HISTORY_OPTYPE_INCR_OFFLINE
	O	증분 온라인	DB2HISTORY_OPTYPE_INCR_ONLINE
	D	델타 오프라인	DB2HISTORY_OPTYPE_DELTA_OFFLINE
	E	델타 온라인	DB2HISTORY_OPTYPE_DELTA_ONLINE
F	E	로그 끝	DB2HISTORY_OPTYPE_EOL
	P	특정 시점	DB2HISTORY_OPTYPE_PIT
L	I	삽입	DB2HISTORY_OPTYPE_INSERT
	R	바꾸기	DB2HISTORY_OPTYPE_REPLACE
Q	S	quiesce 상태 공유	DB2HISTORY_OPTYPE_SHARE
	U	quiesce 상태 갱신	DB2HISTORY_OPTYPE_UPDATE
	X	quiesce 상태 독점	DB2HISTORY_OPTYPE_EXCL
	Z	quiesce 재설정	DB2HISTORY_OPTYPE_RESET
R	F	오프라인	DB2HISTORY_OPTYPE_OFFLINE
	N	온라인	DB2HISTORY_OPTYPE_ONLINE
	I	증분 오프라인	DB2HISTORY_OPTYPE_INCR_OFFLINE
	O	증분 온라인	DB2HISTORY_OPTYPE_INCR_ONLINE
T	C	컨테이너 추가	DB2HISTORY_OPTYPE_ADD_CONT
	R	재조정	DB2HISTORY_OPTYPE_REB

표 20. db2Char 구조 내의 필드

필드 이름	데이터 유형	설명
pioData	char	문자 데이터 버퍼에 대한 포인터. NULL일 경우, 데이터가 리턴되지 않습니다.
iLength	db2Uint32	입력. *pioData* 버퍼 크기.
oLength	db2Uint32	출력. *pioData* 버퍼에 있는 데이터의 유효 문자 수

표 21. db2HistoryEID 구조 내의 필드

필드 이름	데이터 유형	설명
ioNode	SQL_PDB_NODE_TYPE	노드 번호
ioHID	db2Uint32	지역 실행기록 파일 항목 ID

언어 구문

C 구조

```
/* File: db2ApiDf.h */
/* ... */
typedef SQL_STRUCTURE db2HistoryData
{
  char ioHistDataID[8];
  db2Char oObjectPart;
  db2Char oEndTime;
  db2Char oFirstLog;
  db2Char oLastLog;
  db2Char oID;
  db2Char oTableQualifier;
  db2Char oTableName;
  db2Char oLocation;
  db2Char oComment;
  db2Char oCommandText;
  SQLU_LSN oLastLSN;
  db2HistoryEID oEID;
  struct sqlca * poEventSQLCA;
  db2Char * poTablespace;
  db2Uint32 ioNumTablespaces;
  char oOperation;
  char oObject;
  char oOptype;
  char oStatus;
  char oDeviceType
} db2HistoryData;
typedef SQL_STRUCTURE db2Char
{
  char * pioData;
  db2Uint32 ioLength
} db2Char;
typedef SQL_STRUCTURE db2HistoryEID
{
  SQL_PDB_NODE_TYPE ioNode;
  db2Uint32 ioHID
} db2HistoryEID;
/* ... */
```

데이터 구조: SQLU-LSN

409 페이지의 『sqlurlog - 비동기 읽기 로그 API』에서 사용되는 이 결합에는 로그 순차 번호의 정의가 포함됩니다. 로그 순차 번호(LSN)는 데이터베이스 로그 내에서의 상대적 바이트 주소를 나타냅니다. 모든 로그 레코드는 이 번호로 식별됩니다. 이는 데이터베이스 로그의 맨 앞에서부터 로그 레코드의 바이트 오프셋을 나타냅니다.

표 22. SQLU-LSN 구조 내의 필드

필드 이름	데이터 유형	설명
lsnChar	UNSIGNED CHAR 배열	6-구성원 문자 배열 로그 순차 번호를 지정합니다.
lsnWord	UNSIGNED SHORT 배열	3-구성원 축약 배열 로그 순차 번호를 지정합니다.

언어 구문

C 구조

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

데이터 구조: SQLU-RLOG-INFO

이 구조에는 409 페이지의 『sqlurlog - 비동기 읽기 로그 API』에 대한 호출의 상태에 대한 정보 및 데이터베이스 로그가 있습니다.

표 23. SQLU-RLOG-INFO 구조 내의 필드

필드 이름	데이터 유형	설명
initialLSN	SQLU_LSN	첫 번째 데이터베이스 CONNECT문이 발행된 이후에 쓰여지는 첫 번째 로그 레코드의 LSN 값을 지정합니다. 419 페이지의 『데이터 구조: SQLU-LSN』에서 자세한 내용을 참조하십시오.
firstReadLSN	SQLU_LSN	읽혀진 첫 번째 로그 레코드의 LSN 값을 지정합니다.
lastReadLSN	SQLU_LSN	읽혀진 마지막 로그 레코드의 LSN 값을 지정합니다.
curActiveLSN	SQLU_LSN	현재(사용 중인) 로그의 LSN 값을 지정합니다.
logRecsWritten	sqluint32	버퍼에 쓰여질 로그 레코드 수를 지정합니다.
logBytesWritten	sqluint32	버퍼에 쓰여진 바이트 수를 지정합니다.

언어 구문

C 구조

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
SQLU_LSN       initialLSN ;
SQLU_LSN       firstReadLSN ;
SQLU_LSN       lastReadLSN ;
SQLU_LSN       curActiveLSN ;
sqluint32      logRecsWritten ;
sqluint32      logBytesWritten ;
} SQLU_RLOG_INFO;
```

부록E. 샘플 프로그램 복구

Embedded SQL이 없는 샘플 프로그램(backrest.c)

다음 샘플 프로그램은 DB2 백업 및 복원 API를 사용하여 다음을 수행하는 방법을 보여줍니다.

- 데이터베이스 백업
- 데이터베이스 복원
- 데이터베이스 롤 포워드 복구

SAMPLE 데이터베이스에 대한 세부사항은 *SQL 참조서*를 참조하십시오.

이 샘플 프로그램에 대한 소스 파일(backrest.c)은 Windows 운영 체제 및 OS/2에서는 \sqllib\samples\c 디렉토리에서, UNIX 기반 시스템에서는 /sqllib/samples/c 디렉토리에서 볼 수 있습니다. 여기에는 DB2 API의 수가 포함됩니다. 같은 디렉토리에 있는 makefile에는 이 샘플 프로그램과 다른 샘플 프로그램을 빌드하기 위한 명령이 들어 있습니다. DB2 관리 API를 포함하는 응용프로그램의 작성에 대한 일반 정보와 컴파일 및 링크 옵션에 대한 자세한 정보는 *응용프로그램 빌드 안내서*를 참조하십시오. Windows NT 또는 Windows 2000의 소스 파일 backrest.c에서 샘플 프로그램 backrest를 빌드하려면, 다음을 수행하십시오.

1. 파일 backrest.c, makefile, utilapi.c 및 utilapi.h를 작업 디렉토리에 복사하십시오.
2. 데이터베이스 관리 프로그램이 수행 중이 아니면, DB2 명령 창에서 **db2start** 명령을 발행하십시오. CLP 사용 DB2 창을 열고 Windows 운영 체제에서 DB2 명령행 환경을 초기화하려면, 명령 프롬프트에서 **db2cmd**를 발행하십시오.
3. nmake backrest를 입력하십시오. 다음과 같은 파일이 생성됩니다.

```
backrest.exe
backrest.ilk
backrest.obj
backrest.pdb
utilapi.obj
```

샘플 프로그램(실행 파일)을 수행하려면 다음을 입력하십시오.

```
backrest database userID password
```

예를 들면, 다음과 같습니다.

```
backrest sample db2admin db2admin
```

다음은 이 프로그램이 리턴한 출력의 예입니다.

```
This is sample program : backrest.c
NOTE:   Ensure the database is not in use prior to running this program.
Updating the sample database configuration parameter LOGRETAIN to 'ON' to enable rollforward recovery.
Backing up the 'sample' database.
The database has been successfully backed up.
Updating the sample database configuration parameter LOGRETAIN to 'OFF'.
Restoring the database 'sample' as 'TESTBACK' (1st pass).
Should get returned value = SQLUD_INACCESSABLE_CONTAINER.
SQLUD_INACCESSABLE_CONTAINER is returned.
Need to SET TABLESPACES CONTAINERS
Tablespace container information for tablespace 1 obtained.
Tablespace containers have been set for tablespace 1.
Restoring the database 'sample' as 'TESTBACK' (2nd pass).
Database sample has been restored as 'TESTBACK'.
The database 'TESTBACK' has been successfully rolled forward.
The database 'TESTBACK' has been successfully dropped.
```

다음은 샘플 프로그램에 대해 나열되는 소스입니다.

```
/*****************************************************************************
**
** Source File Name = backrest.c
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 2000
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**    PURPOSE :
**       an example showing how to use BACKUP & RESTORE APIs in order to:
**          - backup a database
**          - restore a database
**          - roll forward database to return the database to a state it was
**            in prior to the damage...
**
**    APIs USED :
**       BACKUP DATABASE                  sqlubkp()
**       RESTORE DATABASE                 sqlurestore()
**       UPDATE DATABASE CONFIGURATION    sqlfudb()
**       GET TABLESPACE CONTAINER QUERY   sqlbtcq()
**       SET TABLESPACE CONTAINERS        sqlbstsc()
```

```
**       ROLLFORWARD DATABASE              sqluroll()
**
**    STRUCTURES USED :
**       sqlu_tablespace_bkrst_list
**       SQLB_TBSCONTQRY_DATA
**       sqlu_media_list
**       rfwd_input
**       rfwd_output
**       sqlurf_info
**       sqlca
**
**    OTHER FUNCTIONS DECLARED :
**       'C' COMPILER LIBRARY :
**          stdio.h  -  printf
**
**       internal :
**
**       external :
**          check_error :     Checks for SQLCODE error, and prints out any
**          [in UTIL.C]          related information available.
**                               This procedure is located in the UTIL.C file.
**
**    EXTERNAL DEPENDENCIES :
**       - Ensure existence of database for precompile purposes.
**       - Compile and link with the IBM Cset++ compiler (AIX and OS/2)
**         or the Microsoft Visual C++ compiler (Windows)
**         or the compiler supported on your platform.
**
** For more information about these samples see the README file.
**
** For more information on programming in C, see the:
**   -  "Programming in C and C++" section of the Application Development Guide
** For more information on building C applications, see the:
**   -  "Building C Applications" section of the Application Building Guide.
**
** For more information on the SQL language see the SQL Reference.
**
******************************************************************************/
   #include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlutil.h>
#include <sqlenv.h>
#include "utilapi.h"
/* You can change the following path to another directory for which */
/* you have permission. create the "backup" directory in the same path. */
   #define TEMPDIR "."
int main (int argc, char *argv[]) {
   /* Variables for the BACKUP API */
   sqluint32 buff_size;
   sqluint32 num_buff;
   struct sqlu_tablespace_bkrst_list *tablespace_list;
   struct SQLB_TBSCONTQRY_DATA *cont;
   struct sqlca sqlca;
   struct sqlu_media_entry media_entry;
   struct sqlu_media_list media_list;
   char applid[SQLU_APPLID_LEN+1];
   char timestamp[SQLU_TIME_STAMP_LEN+1];
   char *target_path;
   sqluint32 tsc_id;
   sqluint32 num_cont;
   /* variables for the Update Database Configuration API */
   struct sqlfupd itemList;
```

```
short log_retain;
/* variables for the ROLLFORWARD API */
struct rfwd_input rfwdInput;
char dbAlias[] = "TESTBACK";
char overFlowLogPath[SQL_PATH_SZ] = TEMPDIR;
struct rfwd_output rfwdOutput;
sqlint32   numReplies;
struct sqlurf_info nodeInfo;
printf ("\nThis is sample program : backrest.c\n\n");

printf ("NOTE:   Ensure the database is not in use prior to running
 this program.\n\n");
if (argc != 4) {
   printf ("\nUSAGE: backrest database userID password\n\n");
   return 1;
} /* endif */
/* Altering the default Database Configuration to enable rollforward
   recovery of the TESTBACK database. */
log_retain = 1;
itemList.token = SQLF_DBTN_LOG_RETAIN;

itemList.ptrvalue = (char *) &log_retain;
printf ("Updating the %s database configuration parameter
 LOGRETAIN \n", argv[1]);
printf ("to 'ON' to enable rollforward recovery.\n\n");
/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");
/* Initialize the BACKUP API variables */
buff_size = 16;
num_buff = 1;
tablespace_list = NULL;
media_list.media_type = 'L';
media_list.sessions = 1;
strcpy (media_entry.media_entry, TEMPDIR);
media_list.target.media = &media_entry;
printf ("Backing up the '%s' database.\n", argv[1]);
/*****************\
* BACKUP DATABASE *
\*****************/
sqlubkp (argv[1],
          buff_size,
          SQLUB_OFFLINE,
          SQLUB_FULL,
          SQLUB_BACKUP,
          applid,
          timestamp,
          num_buff,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
                          NULL,
          0,
                          NULL,
          1,
                          NULL,
                          NULL,
                          NULL,
          &sqlca);
API_SQL_CHECK("backing up the database");
```

```
printf ("The database has been successfully backed up.\n\n");

/* Altering the default Database Configuration to disable rollforward
   recovery of the TESTBACK database. */
log_retain = 0;
itemList.token = SQLF_DBTN_LOG_RETAIN;
itemList.ptrvalue = (char *) &log_retain;
printf ("Updating the %s database configuration parameter
 LOGRETAIN \n", argv[1]);
printf ("to 'OFF'.\n\n");
/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");

/* Initialize the variables for the RESTORE API */
buff_size = 1024;
target_path = NULL;
printf ("Restoring the database '%s' as 'TESTBACK' (1st pass).\n", argv[1]);
printf ("Should get returned value = SQLUD_INACCESSABLE_CONTAINER.\n");
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore(argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_RESTORE_STORDEF,
          applid,
          timestamp,
          target_path,
          num_buff,
                          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
                          NULL,
          0,
                          NULL,
          1,
                          NULL,
                          NULL,
                          NULL,
          &sqlca);
if (sqlca.sqlcode != SQLUD_INACCESSABLE_CONTAINER)
   API_SQL_CHECK("restoring database");
printf ("SQLUD_INACCESSABLE_CONTAINER is returned.\n\n");
printf ("Need to SET TABLESPACES CONTAINERS\n");
/* Set the containers structure to a new list of containers */
tsc_id = 1;
cont = (struct SQLB_TBSCONTQRY_DATA *) malloc
   (sizeof(struct SQLB_TBSCONTQRY_DATA));
/*********************************\
* GET TABLESPACE CONTAINER QUERY *
\*********************************/
sqlbtcq (&sqlca, tsc_id, &num_cont, &cont);
API_SQL_CHECK("GET TABLESPACE CONTAINER QUERY");
printf ("Tablespace container information for tablespace 1 obtained.\n");
```

```
/**************************\
* SET TABLESPACE CONTAINERS *
\**************************/
sqlbstsc (&sqlca,
          SQLB_SET_CONT_INIT_STATE,
          tsc_id,
          num_cont,
          cont);
API_SQL_CHECK("SET TABLESPACE CONTAINERS");
printf ("Tablespace containers have been set for tablespace 1.\n\n");
printf ("Restoring the database '%s' as 'TESTBACK' (2nd pass).\n", argv[1]);
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore (argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_CONTINUE,
          applid,
          timestamp,
          target_path,
          num_buff,
                          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
                          NULL,
          0,
                          NULL,
          1,
                          NULL,
                          NULL,
                          NULL,
          &sqlca);
API_SQL_CHECK("restoring database 2");
printf ("Database %s has been restored as 'TESTBACK'.\n\n", argv[1]);
/**********************\
* ROLLFORWARD DATABASE *
\**********************/
rfwdInput.version=SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias=dbAlias;
rfwdInput.CallerAction=SQLUM_ROLLFWD;
rfwdInput.pStopTime=SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName=argv[2];
rfwdInput.pPassword=argv[3];
rfwdInput.pOverflowLogPath=overFlowLogPath;
rfwdInput.NumChngLgOvrflw=0;
rfwdInput.pChngLogOvrflw=NULL;
rfwdInput.ConnectMode=SQLUM_OFFLINE;
rfwdInput.pTablespaceList=NULL;
rfwdInput.AllNodeFlag= SQLURF_ALL_NODES;
rfwdInput.NumNodes=0;
rfwdInput.pNodeList=NULL;
rfwdInput.NumNodeInfo=1;
rfwdInput.DlMode=0;
rfwdInput.pReportFile=NULL;
rfwdInput.pDroppedTblID=NULL;
rfwdInput.pExportDir=NULL;
```

```
   rfwdOutput.pApplicationId=applid;
   rfwdOutput.pNumReplies=&numReplies;
   rfwdOutput.pNodeInfo=&nodeInfo;
   sqluroll( &rfwdInput,
             &rfwdOutput,
             &sqlca);
   API_SQL_CHECK("rolling database forward");
   printf ("The database 'TESTBACK' has been successfully rolled
    forward.\n\n", argv[1]);
   /***************\
   * DROP DATABASE *
   \***************/
   sqledrpd ("TESTBACK",
             &sqlca);
   API_SQL_CHECK("DROP DATABASE");
   printf ("The database 'TESTBACK' has been successfully dropped.\n");
   return 0;
}
```

Embedded SQL이 있는 샘플 프로그램(dbrecov.sqc)

다음 샘플 프로그램은 DB2 백업 및 복원 API를 사용하여 다음을 수행하는 방법을 보여줍니다.

- 데이터베이스 백업
- 데이터베이스 복원
- 데이터베이스 롤 포워드 복구

SAMPLE 데이터베이스에 대한 세부사항은 *SQL 참조서*를 참조하십시오.

주: dbrecov 샘플 파일은 DB2 버전 7.2에 설치되어 있지 않습니다. 이 파일은 웹에서 다운로드해서만 사용 가능합니다.

dbrecov 샘플 프로그램의 필수 파일을 다운로드하는 방법

1. http://www.ibm.com/software/data/db2/udb/ad/v7/abg.html로 가십시오.
2. 『최신 갱신』 아래의 『플랫폼에 의한 dbrecov 샘플 파일』을 누르십시오.
3. 나타나는 페이지에서 플랫폼에 대한 파일을 다운로드할 링크를 선택하십시오. Windows 운영 체제 및 OS/2의 경우, 파일은 zip으로 압축된 실행 파일에 포함되고, UNIX 기반 시스템의 경우에는 파일이 tar.gz 파일에 포함됩니다.

주의:

기존의 DB2 UDB 버전 7 샘플 파일이 상주하는 디렉토리에서는 이 실행 파일을 실행하지 마십시오. 새로운 유틸리티 파일이 DB2 UDB 버전 7 유틸리티 파일의 기존 버전 위에 겹쳐쓰기 때문입니다. 새로운 유틸리티 파일은 다른 샘플 프로그램과 호환되지 않습니다.

이 샘플 프로그램(dbrecov.sqc)에 대한 소스 파일에는 DB2 API와 Embedded SQL 호출 둘다 있습니다. DB2 관리 API를 포함하는 응용프로그램의 작성에 대한 일반 정보와 컴파일 및 링크 옵션에 대한 자세한 정보는 *응용프로그램 빌드 안내서*를 참조하십시오.

다음은 Windows 운영 체제에서 dbrecov 프로그램을 빌드하고 수행하기 위한 지시사항입니다. 지원되는 다른 운영 체제에서 프로그램을 빌드하려면, 샘플 파일 사이에 포함되는 README 파일을 참조하십시오.

1. 다운로드한 실행 파일 dbrecov_win.exe를 작업 디렉토리에서 수행하십시오. 다음과 같은 파일이 압축해제됩니다.

```
dbrecov.sqc - Embedded SQL source file for dbrecov program
utilapi.c   - Error-checking utility file for DB2 API programs
utilapi.h   - Header file for utilapi.c
utilemb.sqc - Error-checking utility file for embedded SQL programs
utilemb.h   - Header file for utilemb.sqc
bldmapp.bat - Builds Microsoft Visual C++ application programs
bldvapp.bat - Builds VisualAge C++ application programs
embprep.bat - Precompiles and binds embedded SQL programs
README      - Contains information about the program files
```

2. 데이터베이스 관리 프로그램이 수행 중이 아니면, DB2 명령 창에서 **db2start** 명령을 발행하십시오. CLP 사용 DB2 창을 열고 Windows 운영 체제에서 DB2 명령행 환경을 초기화하려면, 명령 프롬프트에서 **db2cmd**를 발행하십시오.
3. 컴파일러에 따른 bldmapp dbrecov 또는 bldvapp dbrecov를 입력하십시오. 다음과 같은 파일이 생성됩니다.

```
dbrecov.exe
dbrecov.ilk
dbrecov.c
dbrecov.obj
dbrecov.bnd
dbrecov.pdb
utilemb.c
utilemb.obj
```

샘플 프로그램(실행 파일)을 수행하려면 다음을 입력하십시오.

```
dbrecov
```

출력은 운영 체제 환경에 따라 다양합니다. 다음은 Windows 시스템에서 이 프로그램이 리턴한 출력의 예입니다.

```
THIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.
USE THE DB2 API:
  sqlfxdb -- Get Database Configuration
TO GET THE DATABASE CONFIGURATION AND DETERMINE
THE SERVER WORKING PATH.
NOTE: Backup images will be created on the server
      in the directory D:\DB2,
      and will not be deleted by the program.
****************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
****************************************
USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT
  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.
  Connecting to 'sample' database...
  Connected to 'sample' database.
  Prune the recovery history file for 'sample' database.
  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.
****************************************
*** BACK UP AND RESTORE A DATABASE ***
****************************************
USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurestore -- Restore Database
TO BACK UP AND RESTORE A DATABASE.
  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO
  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162032
  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162032
    - target database        : sample
-- The following warning report is expected! --
---- warning report ---------------------------
application message = database restore -- start
line                = 506
file                = dbrecov.sqc
SQLCODE             = 2539
SQL2539W  Warning!  Restoring to an existing database
that is the same as the backup image database.
The database files will be deleted.
---- end warning report -----------------------
  Continuing the restore operation...
```

```
  Restore finished.
*************************
*** REDIRECTED RESTORE ***
*************************
USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqlbmtsq -- Tablespace Query
  sqlbtcq -- Tablespace Container Query
  sqlbstsc -- Set Tablespace Containers
  sqlefmem -- Free Memory
  sqledrpd -- Drop Database
TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.
  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO
  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162125
  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162125
    - target database        : RRDB
-- The following warning report is expected! --
---- warning report --------------------------
application message = database restore -- start
line                = 776
file                = dbrecov.sqc
SQLCODE             = 1277
SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
---- end warning report ----------------------
  Continuing the restore operation...
  Find and redefine inaccessible containers.
    Redefine inaccessible container:
      - table space name: SYSCATSPACE
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0000.0
      - container type: path
      - new container name: D:\DB2\SQLT0000.0
    Redefine inaccessible container:
      - table space name: TEMPSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0001.0
      - container type: path
      - new container name: D:\DB2\SQLT0001.0
    Redefine inaccessible container:
      - table space name: USERSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0002.0
      - container type: path
      - new container name: D:\DB2\SQLT0002.0
  Restore finished.
  Drop the 'RRDB' database.
****************************
*** ROLLFORWARD RECOVERY ***
****************************
USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
```

```
  sqluroll -- Rollforward Database
  sqledrpd -- Drop Database
TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD.
  Update 'sample' database configuration:
    - Enable the configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES
  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162223
  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162223
    - target database        : RFDB
  Restore finished.
  Rolling 'RFDB' database forward ...
  Rollforward finished.
  Drop the 'RFDB' database.
*****************************
*** ASYNCHRONOUS READ LOG ***
*****************************
USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurlog -- Asynchronous Read Log
AND THE SQL STATEMENTS:
  CONNECT
  ALTER TABLE
  COMMIT
  INSERT
  DELETE
  ROLLBACK
  CONNECT RESET
TO READ LOG RECORDS FOR THE CURRENT CONNECTION.
  Update 'sample' database configuration:
    - Enable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES
  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162247
  Connecting to 'sample' database...
  Connected to 'sample' database.
  Invoke the following SQL statements:
    ALTER TABLE emp_resume DATA CAPTURE CHANGES;
    COMMIT;
    INSERT INTO emp_resume
      VALUES('000777', 'ascii', 'This is a new resume.');
            ('777777', 'ascii', 'This is another new resume');
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000777';
    DELETE FROM emp_resume WHERE empno = '777777';
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000140';
    ROLLBACK;
    ALTER TABLE emp_resume DATA CAPTURE NONE;
    COMMIT;
  Start reading database log.
-- The following warning report is expected! --
---- warning report --------------------------
application message = database log info -- get
```

```
line                = 1457
file                = dbrecov.sqc
SQLCODE             = 2653
SQL2653W  A Restore, Forward or Crash Recovery may have reused log sequence
number ranges.  Reason code "1".
---- end warning report ----------------------
    Record type: Normal
      component ID: DMS log record
      function ID:  Alter Table Attribute
        Propagation attribute is changed to: ON
    Record type: Normal
      component ID: DMS log record
      function ID: Update Record
        oldRID: 2
        old subrecord length: 76
        old subrecord offset: 0
        subrecord type: Updatable, Internal control
        newRID: 2
        new subrecord length: 76
        new subrecord offset: 2916
        subrecord type: Updatable, Internal control
    Record type: Local pending list
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK
    Record type: Normal
      component ID: DMS log record
      function ID: Insert Record
        RID: 12
        subrecord length: 88
        subrecord offset: 486
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        30 30 30 37 37 37 0F 00 05 00 *000777....*
        14 00 3C 00 00 61 73 63 69 69 *.....ascii*
        49 06 01 00 00 00 00 00 15 00 *I.........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 3C 00 01 00 00 00 15 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
    Record type: Normal
      component ID: DMS log record
      function ID: Insert Record
        RID: 13
        subrecord length: 88
        subrecord offset: 398
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        37 37 37 37 37 37 0F 00 05 00 *777777....*
        14 00 3C 00 00 61 73 63 69 69 *.....ascii*
        49 06 01 00 00 00 00 00 1A 00 *I.........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 3C 00 01 00 00 00 1A 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 01 00 00 00 *..........*
    Record type: Normal commit
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK
    Record type: Normal
      component ID: DMS log record
```

```
  function ID: Delete Record
    RID: 12
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 13
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*
Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK
Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 6
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*
Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
```

```
          74 49 06 01 00 00 00 00 00 56 *tI.......V*
          07 00 00 00 00 00 00 00 00 00 *..........*
          00 00 01 3C 00 02 00 00 00 56 *.........V*
          07 00 00 00 00 00 00 00 00 00 *..........*
          00 00 00 00 00 00 00 00 00 00 *..........*
          00 00 00 00 00 00 00 16 00 00 *..........*
          00                            *.         *
      Record type: Compensation
        component ID: DMS log record
        function ID: Undo Delete Record
          RID: 7
          subrecord length: 89
          subrecord offset: 397
          subrecord type: Updatable, Formatted user data
          user data fixed length: 15
          user data:
          30 30 30 31 34 30 0F 00 06 00 *000140....*
          15 00 3C 00 00 73 63 72 69 70 *.....scrip*
          74 49 06 01 00 00 00 00 00 56 *tI.......V*
          07 00 00 00 00 00 00 00 00 00 *..........*
          00 00 01 3C 00 02 00 00 00 56 *.........V*
          07 00 00 00 00 00 00 00 00 00 *..........*
          00 00 00 00 00 00 00 00 00 00 *..........*
          00 00 00 00 00 00 00 16 00 00 *..........*
          00                            *.         *
      Record type: Compensation
        component ID: DMS log record
        function ID: Undo Delete Record
          RID: 6
          subrecord length: 88
          subrecord offset: 309
          subrecord type: Updatable, Formatted user data
          user data fixed length: 15
          user data:
          30 30 30 31 34 30 0F 00 05 00 *000140....*
          14 00 3C 00 00 61 73 63 69 69 *.....ascii*
          49 06 01 00 00 00 00 00 24 05 *I.........*
          00 00 00 00 00 00 00 00 00 00 *..........*
          00 01 3C 00 02 00 00 00 24 05 *..........*
          00 00 00 00 00 00 00 00 00 00 *..........*
          00 00 00 00 00 00 00 00 00 00 *..........*
          00 00 00 00 00 00 14 00 00 00 *..........*
      Record type: Normal abort
        authorization ID of the application: MELNYK
      Record type: Normal
        component ID: DMS log record
        function ID:  Alter Table Attribute
          Propagation attribute is changed to: OFF
      Record type: Local pending list
        UTC transaction committed (in seconds since 01/01/70): 989180591
        authorization ID of the application: MELNYK
    Disconnecting from 'sample' database...
    Disconnected from 'sample' database.
*********************************************
*** READ A DATABASE RECOVERY HISTORY FILE ***
*********************************************
USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryCloseScan -- Close Recovery History File Scan
TO READ A DATABASE RECOVERY HISTORY FILE.
  Open recovery history file for 'sample' database.
  Read entry number 0.
```

```
Display entry number 0.
  object part: 20010506162032001
  end time: 200105061620
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 48
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:
Read entry number 1.
Display entry number 1.
  object part: 20010506162058001
  end time: 200105061621
  first log: S0000000
  last log: S0000000
  ID: 20010506162032
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 RESTORE SAMPLE NO RF
  command text:
  history file entry ID: 49
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: R
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:
Read entry number 2.
Display entry number 2.
  object part: 20010506162125001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 50
  table spaces:
```

```
        SYSCATSPACE
        USERSPACE1
      type of operation: B
      granularity of the operation: D
      operation type: F
      entry status: A
      device type: D
      SQLCA:
        sqlcode: 0
        sqlstate:
        message:
    Read entry number 3.
    Display entry number 3.
      object part: 20010506162223001
      end time: 200105061622
      first log: S0000000
      last log: S0000000
      ID:
      table qualifier:
      table name:
      location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
      comment: DB2 BACKUP SAMPLE OFFLINE
      command text:
      history file entry ID: 51
      table spaces:
        SYSCATSPACE
        USERSPACE1
      type of operation: B
      granularity of the operation: D
      operation type: F
      entry status: A
      device type: D
      SQLCA:
        sqlcode: 0
        sqlstate:
        message:
    Read entry number 4.
    Display entry number 4.
      object part: 20010506162247001
      end time: 200105061623
      first log: S0000000
      last log: S0000000
      ID:
      table qualifier:
      table name:
      location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
      comment: DB2 BACKUP SAMPLE OFFLINE
      command text:
      history file entry ID: 52
      table spaces:
        SYSCATSPACE
        USERSPACE1
      type of operation: B
      granularity of the operation: D
      operation type: F
      entry status: A
      device type: D
      SQLCA:
        sqlcode: 0
        sqlstate:
        message:
    Close recovery history file for 'sample' database.
  ****************************************************
```

```
*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***
*****************************************************
USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryUpdate -- Update Recovery History File
  db2HistoryCloseScan -- Close Recovery History File Scan
TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.
  Open the recovery history file for 'sample' database.
  Read the first entry in the recovery history file.
  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:
  Connecting to 'sample' database...
  Connected to 'sample' database.
  Update the first entry in the history file:
    new location = 'this is the NEW LOCATION'
    new comment = 'this is the NEW COMMENT'
  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.
  Close recovery history file for 'sample' database.
  Read the first recovery history file entry.
  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: this is the NEW LOCATION
    comment: this is the NEW COMMENT
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
```

```
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:
  Close the recovery history file for 'sample' database.
****************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
****************************************
USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT
  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.
  Connecting to 'sample' database...
  Connected to 'sample' database.
  Prune the recovery history file for 'sample' database.
  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.
```

Following is the source listing for the sample program:

```
/*****************************************************************************
**
** Source File Name: dbrecov.sqc
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 2001
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**    PURPOSE:
**    This sample shows how to recover a database.
**
**    APIs USED:
**        db2HistoryCloseScan -- Close Recovery History File Scan
**        db2HistoryGetEntry -- Get Next Recovery History File Entry
**        db2HistoryOpenScan -- Open Recovery History File Scan
**        db2HistoryUpdate -- Update Recovery History File
**        db2Prune -- Prune Recovery History File
**        sqlbmtsq -- Tablespace Query
**        sqlbstsc -- Set Tablespace Containers
**        sqlbtcq -- Tablespace Container Query
**        sqledrpd -- Drop Database
**        sqlefmem -- Free Memory
**        sqlfudb -- Update Database Configuration
**        sqlfxdb -- Get Database Configuration
**        sqlubkp -- Backup Database
**        sqlurestore -- Restore Database
**        sqlurlog -- Asynchronous Read Log
**        sqluroll -- Rollforward Database
**
** For detailed information about database backup and recovery, see the
** "Data Recovery and High Availability Guide and Reference". This manual will
** help you to determine which database and table space recovery methods are
** best suited to your business environment.
**
** For more information about the sample programs, see the README file.
```

```
**
** For more information about programming in C, see the
** "Programming in C and C++" section of the "Application Development Guide".
**
** For more information about building C applications, see the
** section for your compiler in the "Building Applications" chapter
** for your platform in the "Application Building Guide".
**
** For more information about SQL, see the "SQL Reference".
**
** For more information about DB2 APIs, see the "Administrative API Reference".
**
** For the latest information on programming, compiling, and running DB2
** applications, visit the DB2 application development Web site at
**     http://www.software.ibm.com/data/db2/udb/ad
**
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <db2ApiDf.h>
#include "utilemb.h"
int DbRecoveryHistoryFilePrune(char *, char *, char *);
int DbBackupAndRestore(char *, char *, char *, char *, char *);
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);
/* support function called by the main() */
int ServerWorkingPathGet(char *, char *);
/* support function called by DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);
/* support function called by DbBackupAndRedirectedRestore() and
   DbBackupRestoreAndRollforward() */
int DbDrop(char *);
/* support function called by DbLogRecordsForCurrentConnectionRead() */
int LogBufferDisplay(char *, sqluint32);
int LogRecordDisplay(char *, sqluint32, sqluint16, sqluint16);
int SimpleLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32);
int ComplexLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32,
                            sqluint8, char *, sqluint32);
int LogSubRecordDisplay(char *, sqluint16);
int UserDataDisplay(char *, sqluint16);
/* support functions called by DbRecoveryHistoryFileRead() and
   DbFirstRecoveryHistoryFileEntryUpdate() */
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData);
int HistoryEntryDataFieldsFree(struct db2HistoryData *);
/* DbCreate will create a new database on the server with the server's
    code page.
   Use this function only if you want to restore a remote database.
   This support function is being called by DbBackupAndRedirectedRestore()
   and DbBackupRestoreAndRollforward().  */
int DbCreate(char *, char *);
int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1];
  char serverWorkingPath[SQL_PATH_SZ + 1];
  char restoredDbAlias[SQL_ALIAS_SZ + 1];
```

```
char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1];
char rolledForwardDbAlias[SQL_ALIAS_SZ + 1];
sqluint16 savedLogRetainValue;
char dbAlias[SQL_ALIAS_SZ + 1];
char user[USERID_SZ + 1];
char pswd[PSWD_SZ + 1];
/* check the command line arguments */
rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
if (rc != 0)
{
  return rc;
}
printf("\nTHIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.\n");
strcpy(restoredDbAlias, dbAlias);
strcpy(redirectedRestoredDbAlias, "RRDB");
strcpy(rolledForwardDbAlias, "RFDB");
/* attach to a local or remote instance */
rc = InstanceAttach(nodeName, user, pswd);
if (rc != 0)
{
  return rc;
}
printf("\nUSE THE DB2 API:\n");
printf("  sqlfxdb -- Get Database Configuration\n");
printf("TO GET THE DATABASE CONFIGURATION AND DETERMINE\n");
printf("THE SERVER WORKING PATH.\n");
/* get the server working path */
rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
if (rc != 0)
{
  return rc;
}
printf("\nNOTE: Backup images will be created on the server\n");
printf("      in the directory %s,\n", serverWorkingPath);
printf("      and will not be deleted by the program.\n");
/* call the sample functions */
rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
rc = DbBackupAndRestore(dbAlias,
                        restoredDbAlias,
                        user,
                        pswd,
                        serverWorkingPath);
rc = DbBackupAndRedirectedRestore(dbAlias,
                                  redirectedRestoredDbAlias,
                                  user,
                                  pswd,
                                  serverWorkingPath);
rc = DbBackupRestoreAndRollforward(dbAlias,
                                   rolledForwardDbAlias,
                                   user,
                                   pswd,
                                   serverWorkingPath);
rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                          user,
                                          pswd,
                                          serverWorkingPath);
rc = DbRecoveryHistoryFileRead(dbAlias);
rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);
rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
/* detach from the local or remote instance */
rc = InstanceDetach(nodeName);
if (rc != 0)
{
```

```
    return rc;
  }
  return 0;
} /* end main */
int ServerWorkingPathGet(char dbAlias[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbConfigFields[1];
  char serverLogPath[SQL_PATH_SZ + 1];
  int len;
  /* get the server log path */
  /* SQLF_DBTN_LOGPATH is a token of the non-updatable database configuration
     parameter 'logpath'; it is used to get the server log path */
  dbConfigFields[0].token = SQLF_DBTN_LOGPATH;
  dbConfigFields[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));
  /* get database configuration */
  /* the API sqlfxdb returns the values of individual entries in a
     database configuration file */
  sqlfxdb(dbAlias, 1, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("server log path -- get");
  strcpy(serverLogPath, dbConfigFields[0].ptrvalue);
  free(dbConfigFields[0].ptrvalue);
  /* choose the server working path; if, for example, serverLogPath =
     "C:\DB2\NODE0001\....", we'll keep "C:\DB2" for the serverWorkingPath
     variable; backup images created in this sample will be placed under
     the 'serverWorkingPath' directory */
  len = (int)(strstr(serverLogPath, "NODE") - serverLogPath - 1);
  memcpy(serverWorkingPath, serverLogPath, len);
  serverWorkingPath[len] = '\0';
  return 0;
} /* ServerWorkingPathGet */
int DbCreate(char existingDbAlias[], char newDbAlias[])
{
  struct sqlca sqlca;
  char dbName[SQL_DBNAME_SZ + 1];
  char dbLocalAlias[SQL_ALIAS_SZ + 1];
  char dbPath[SQL_PATH_SZ + 1];
  struct sqledbdesc dbDescriptor;
  struct sqledbcountryinfo countryInfo;
  struct sqlfupd dbConfigFields[2];
  printf("\n  Create '%s' empty database with the same code set as
   '%s' database.\n",
         newDbAlias, existingDbAlias);
  /* initialize dbConfigFields */
  dbConfigFields[0].token = SQLF_DBTN_TERRITORY;
  dbConfigFields[0].ptrvalue = (char *)malloc(10 * sizeof(char));
  memset(dbConfigFields[0].ptrvalue, '\0', 10);
  dbConfigFields[1].token = SQLF_DBTN_CODESET;
  dbConfigFields[1].ptrvalue = (char *)malloc(20 * sizeof(char));
  memset(dbConfigFields[1].ptrvalue, '\0', 20);
  /* get two database configuration fields */
  sqlfxdb(existingDbAlias, 2, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("DB Config. -- Get");
  /* create a new database */
  strcpy(dbName, newDbAlias);
  strcpy(dbLocalAlias, newDbAlias);
  strcpy(dbPath, "");
  strcpy(dbDescriptor.sqldbdid, SQLE_DBDESC_2);
  dbDescriptor.sqldbccp = 0;
  dbDescriptor.sqldbcss = SQL_CS_NONE;
  strcpy(dbDescriptor.sqldbcmt, "");
```

```
  dbDescriptor.sqldbsgp = 0;
  dbDescriptor.sqldbnsg = 10;
  dbDescriptor.sqltsext = -1;
  dbDescriptor.sqlcatts = NULL;
  dbDescriptor.sqlusrts = NULL;
  dbDescriptor.sqltmpts = NULL;
  strcpy(countryInfo.sqldbcodeset, (char *)dbConfigFields[1].ptrvalue);
  strcpy(countryInfo.sqldblocale, (char *)dbConfigFields[0].ptrvalue);
  /* create database */
  sqlecrea(dbName,
           dbLocalAlias,
           dbPath,
           &dbDescriptor,
           &countryInfo,
           '\0',
                              NULL,
           &sqlca);
  DB2_API_CHECK("Database -- Create");
  /* free the allocated memory */
  free(dbConfigFields[0].ptrvalue);
  free(dbConfigFields[1].ptrvalue);
  return 0;
} /* DbCreate */
int DbDrop(char dbAlias[])
{
  struct sqlca sqlca;
  printf("\n  Drop the '%s' database.\n", dbAlias);
  /* drop and uncatalog the database */
  sqledrpd(dbAlias, &sqlca);
  DB2_API_CHECK("Database -- Drop");
  return 0;
} /* DbDrop */
int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[],
                       char user[],
                       char pswd[],
                       char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
```

```
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;
printf("\n************************************\n");
printf("*** BACK UP AND RESTORE A DATABASE ***\n");
printf("************************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("TO BACK UP AND RESTORE A DATABASE.\n");
printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Disable the database configuration parameter LOGRETAIN\n");
printf("        i.e., set LOGRETAIN = OFF/NO\n");
/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;
/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");
/*******************************/
/*    BACK UP THE DATABASE     */
/*******************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);
backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;
backupCallerAction = SQLUB_BACKUP;
/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
```

```
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");
while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  /* depending on the sqlca.sqlcode value, user action may be */
  /* required, such as mounting a new tape */
  printf("\n  Continuing the backup operation...\n");
  backupCallerAction = SQLUB_CONTINUE;
  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}
printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);
/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_NOROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;
printf("\n  Restoring a database ...\n");
printf("    - source image alias     : %s\n", dbAlias);
```

```
  printf("    - source image time stamp: %s\n", restoreTimestamp);
  printf("    - target database        : %s\n", restoredDbAlias);
  restoreCallerAction = SQLUD_RESTORE;
  /* The API sqlurestore is used to restore a database that has been backed
     up using the API sqlubkp. */
  sqlurestore(dbAlias,
              restoredDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
              NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
              NULL,
              0,
              NULL,
              restoreParallelismDegree,
              NULL,
              NULL,
              NULL,
              &sqlca);
/*  DB2_API_CHECK("database restore -- start");  */
  EXPECTED_WARN_CHECK("database restore -- start");
  while (sqlca.sqlcode != 0)
  {
    /* continue the restore operation */
    printf("\n  Continuing the restore operation...\n");
    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape */
    restoreCallerAction = SQLUD_CONTINUE;
    /* restore the database */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
```

```
                NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }
  printf("\n  Restore finished.\n");
  return 0;
} /* DbBackupAndRestore */
int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                 char user[],
                                 char pswd[],
                                 char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;
  printf("\n**************************\n");
  printf("*** REDIRECTED RESTORE ***\n");
  printf("**************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  sqlfudb -- Update Database Configuration\n");
  printf("  sqlubkp -- Backup Database\n");
  printf("  sqlecrea -- Create Database\n");
  printf("  sqlurestore -- Restore Database\n");
  printf("  sqlbmtsq -- Tablespace Query\n");
  printf("  sqlbtcq -- Tablespace Container Query\n");
  printf("  sqlbstsc -- Set Tablespace Containers\n");
  printf("  sqlefmem -- Free Memory\n");
  printf("  sqledrpd -- Drop Database\n");
  printf("TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.\n");
  printf("\n  Update \'%s\' database configuration:\n", dbAlias);
  printf("    - Disable the database configuration parameter LOGRETAIN \n");
  printf("        i.e., set LOGRETAIN = OFF/NO\n");
  /* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
     parameter 'logretain'; it is used to update the database configuration
     file */
```

```
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;
/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;
/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE    */
/*******************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);
backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;
backupCallerAction = SQLUB_BACKUP;
/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");
while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape */
  printf("\n  Continuing the backup operation...\n");
  backupCallerAction = SQLUB_CONTINUE;
  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
```

```
                backupAppId,
                backupStartTimestamp,
                backupBuffersNb,
                &backupTablespaceList,
                &backupTargetMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                backupParallelismDegree,
                &backupImageSize,
                NULL,
                NULL,
                &sqlca);
      DB2_API_CHECK("Database -- Backup");
    }
    printf("  Backup finished.\n");
    printf("    - backup image size      : %d MB\n", backupImageSize);
    printf("    - backup image path      : %s\n",
           backupTargetMediaEntries[0].media_entry);
    printf("    - backup image time stamp: %s\n",
           backupStartTimestamp);
    /* To restore a remote database, you will first need to create an empty
       database if the client's code page is different from the server's
       code page.
       If this is the case, uncomment the call to DbCreate(). It will create
       an empty database on the server with the server's code page. */

  /*
    rc = DbCreate(dbAlias, restoredDbAlias);
    if (rc != 0)
    {
    return rc;
    }
  */
    /*****************************/
    /*    RESTORE THE DATABASE    */
    /*****************************/
    restoreBufferSize = 1024; /*  1024 x 4KB */
    rollforwardPendingState = SQLUD_NOROLLFWD;
    datalinkMode = SQLUD_NODATALINK;
    restoreType = SQLUD_FULL;
    restoreMode = SQLUD_OFFLINE;
    strcpy(restoreTimestamp, backupStartTimestamp);
    restoreTargetPath = NULL;
    restoreBuffersNb = 1;
    restoreTablespaceList.num_entry = 0; /* number of table spaces */
    restoreTablespaceList.tablespace = NULL;
    restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
    restoreSourceMediaList.sessions = 1; /* number of elements in the target */
    restoreSourceMediaList.target.media = restoreSourceMediaEntries;
    strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
    restoreParallelismDegree = 1;
    printf("\n  Restoring a database ...\n");
    printf("    - source image alias     : %s\n", dbAlias);
    printf("    - source image time stamp: %s\n", restoreTimestamp);
    printf("    - target database        : %s\n", restoredDbAlias);
    restoreCallerAction = SQLUD_RESTORE_STORDEF;
    /* The API sqlurestore is used to restore a database that has been backed
       up using the API sqlubkp. */
    sqlurestore(dbAlias,
                restoredDbAlias,
```

```
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
/*  DB2_API_CHECK("database restore -- start"); */
  EXPECTED_WARN_CHECK("database restore -- start");
  while (sqlca.sqlcode != 0)
  {
    /* continue the restore operation */
    printf("\n  Continuing the restore operation...\n");
    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape */
    if (sqlca.sqlcode == SQLUD_INACCESSABLE_CONTAINER)
    {
      /* redefine the table space container layout */
      printf("\n  Find and redefine inaccessable containers.\n");
      rc = InaccessableContainersRedefine(serverWorkingPath);
      if (rc != 0)
      {
        return rc;
      }
    }
    restoreCallerAction = SQLUD_CONTINUE;
    /* restore the database */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
```

```
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }
  printf("\n  Restore finished.\n");
  /* drop the restored database */
  rc = DbDrop(restoredDbAlias);
  return 0;
} /* DbBackupAndRedirectedRestore */
int InaccessableContainersRedefine(char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  sqluint32 numTablespaces;
  struct SQLB_TBSPQRY_DATA **ppTablespaces;
  sqluint32 numContainers;
  struct SQLB_TBSCONTQRY_DATA *pContainers;
  int tspNb;
  int contNb;
  char pathSep[2];
  /* The API sqlbmtsq provides a one-call interface to the table space query
     data. The query data for all table spaces in the database is returned
     in an array. */
  sqlbmtsq(&sqlca,
           &numTablespaces,
           &ppTablespaces,
           SQLB_RESERVED1,
           SQLB_RESERVED2);
  DB2_API_CHECK("tablespaces -- get");
  /* refedine the inaccessible containers */
  for (tspNb = 0; tspNb < numTablespaces; tspNb++)
  {
    /* The API sqlbtcq provides a one-call interface to the table space
       container query data. The query data for all the containers in a table
       space, or for all containers in all table spaces, is returned in an
       array. */
    sqlbtcq(&sqlca, ppTablespaces[tspNb]->id, &numContainers, &pContainers);
    DB2_API_CHECK("tablespace containers -- get");
    for (contNb = 0; contNb < numContainers; contNb++)
    {
      if (!pContainers[contNb].ok)
      {
        /* redefine inaccessible container */
        printf("\n    Redefine inaccessible container:\n");
        printf("      - table space name: %s\n",
               ppTablespaces[tspNb]->name);
        printf("      - default container name: %s\n",
               pContainers[contNb].name);
        if (strstr(pContainers[contNb].name, "/"))
        { /* UNIX */
          strcpy(pathSep, "/");
        }
        else
        { /* Intel */
          strcpy(pathSep, "\\");
        }
        switch (pContainers[contNb].contType)
        {
          case SQLB_CONT_PATH:
            printf("      - container type: path\n");
```

```
            sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                              serverWorkingPath, pathSep,
                                              ppTablespaces[tspNb]->id,
                                              pContainers[contNb].id);
            printf("      - new container name: %s\n",
             pContainers[contNb].name);
            break;
          case SQLB_CONT_DISK:
          case SQLB_CONT_FILE:
          default:
            printf("    Unknown container type.\n");
            break;
        }
      }
    }
    /* The API sqlbstsc is used to set or redefine table space containers
       while performing a 'redirected' restore of the database. */
    sqlbstsc(&sqlca,
             SQLB_SET_CONT_FINAL_STATE,
             ppTablespaces[tspNb]->id,
             numContainers,
             pContainers);
    DB2_API_CHECK("tablespace containers -- redefine");
    /* The API sqlefmem is used here to free memory allocated by DB2 for use
       with the API sqlbtcq (Tablespace Container Query). */
    sqlefmem(&sqlca, pContainers);
    DB2_API_CHECK("tablespace containers memory -- free");
  }
  /* The API sqlefmem is used here to free memory allocated by DB2 for
     use with the API sqlbmtsq (Tablespace Query). */
  sqlefmem(&sqlca, ppTablespaces);
  DB2_API_CHECK("tablespaces memory -- free");
  return 0;
} /* InaccessibleContainersRedefine */
int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[],
                                  char pswd[],
                                  char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
```

```
char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
char *restoreTargetPath;
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;
struct rfwd_input rfwdInput;
struct rfwd_output rfwdOutput;
char rollforwardAppId[SQLU_APPLID_LEN + 1];
sqlint32 numReplies;
struct sqlurf_info nodeInfo;
printf("\n**************************\n");
printf("*** ROLLFORWARD RECOVERY ***\n");
printf("**************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlecrea -- Create Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("  sqluroll -- Rollforward Database\n");
printf("  sqledrpd -- Drop Database\n");
printf("TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD. \n");
printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");

dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;
/* enable the configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;
/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");
/* start the backup operation */
printf("\n  Backing up the '%s' database...\n", dbAlias);
backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;
backupCallerAction = SQLUB_BACKUP;
/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
```

```
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
  while (sqlca.sqlcode != 0)
  {
    /* continue the backup operation */
    printf("\n  Continuing the backup operation...\n");
    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape. */
    backupCallerAction = SQLUB_CONTINUE;
    /* back up the database */
    sqlubkp(dbAlias,
            backupBufferSize,
            backupMode,
            backupType,
            backupCallerAction,
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            backupParallelismDegree,
            &backupImageSize,
            NULL,
            NULL,
            &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }
  printf("  Backup finished.\n");
  printf("    - backup image size      : %d MB\n", backupImageSize);
  printf("    - backup image path      : %s\n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s\n",
         backupStartTimestamp);
  /* To restore a remote database, you will first need to create an empty
     database if the client's code page is different from the server's
     code page.
     If this is the case, uncomment the call to DbCreate(). It will create
     an empty database on the server with the server's code page. */
/*
  rc = DbCreate(dbAlias, rolledForwardDbAlias);
  if (rc != 0)
  {
    return rc;
  }
*/
  /*****************************/
  /*    RESTORE THE DATABASE    */
  /*****************************/
  restoreBufferSize = 1024; /*  1024 x 4KB */
```

```
rollforwardPendingState = SQLUD_ROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;
printf("\n  Restoring a database ...\n");
printf("    - source image alias     : %s\n", dbAlias);
printf("    - source image time stamp: %s\n", restoreTimestamp);
printf("    - target database        : %s\n", rolledForwardDbAlias);
restoreCallerAction = SQLUD_RESTORE;
/* The API sqlurestore is used to restore a database that has been backed
   up using the API sqlubkp. */
sqlurestore(dbAlias,
            rolledForwardDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            restoreParallelismDegree,
            NULL,
            NULL,
            NULL,
            &sqlca);
DB2_API_CHECK("database restore -- start");
while (sqlca.sqlcode != 0)
{
  /* continue the restore operation */
  printf("\n  Continuing the restore operation...\n");
  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */
  restoreCallerAction = SQLUD_CONTINUE;
  /* restore the database */
  sqlurestore(dbAlias,
              rolledForwardDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
```

```
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }
  printf("\n  Restore finished.\n");

  /*****************************/
  /*    ROLLFORWARD RECOVERY    */
  /*****************************/

  printf("\n  Rolling '%s' database forward ...\n", rolledForwardDbAlias);
  rfwdInput.version = SQLUM_RFWD_VERSION;
  rfwdInput.pDbAlias = rolledForwardDbAlias;
  rfwdInput.CallerAction = SQLUM_ROLLFWD_STOP;
  rfwdInput.pStopTime = SQLUM_INFINITY_TIMESTAMP;
  rfwdInput.pUserName = user;
  rfwdInput.pPassword = pswd;
  rfwdInput.pOverflowLogPath = serverWorkingPath;
  rfwdInput.NumChngLgOvrflw = 0;
  rfwdInput.pChngLogOvrflw = NULL;
  rfwdInput.ConnectMode = SQLUM_OFFLINE;
  rfwdInput.pTablespaceList = NULL;
  rfwdInput.AllNodeFlag = SQLURF_ALL_NODES;
  rfwdInput.NumNodes = 0;
  rfwdInput.pNodeList = NULL;
  rfwdInput.NumNodeInfo = 1;
  rfwdInput.DlMode = 0;
  rfwdInput.pReportFile = NULL;
  rfwdInput.pDroppedTblID = NULL;
  rfwdInput.pExportDir = NULL;
  rfwdOutput.pApplicationId = rollforwardAppId;
  rfwdOutput.pNumReplies = &numReplies;
  rfwdOutput.pNodeInfo = &nodeInfo;
  /* rollforward database */
  /* The API sqluroll rollforward recovers a database by
     applying transactions recorded in the database log files. */
  sqluroll(&rfwdInput, &rfwdOutput, &sqlca);
  DB2_API_CHECK("rollforward -- start");
  printf("  Rollforward finished.\n");
  /* drop the restored database */
  rc = DbDrop(rolledForwardDbAlias);
  return 0;
} /* DbBackupRestoreAndRollforward */
int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                                         char pswd[],
                                         char serverWorkingPath[])
{
```

```
int rc = 0;
struct sqlca sqlca;
struct sqlfupd dbCfgParameters[1];
unsigned short logretain;
sqluint32 backupBufferSize;
sqluint32 backupMode;
sqluint32 backupType;
sqluint32 backupCallerAction;
char backupAppId[SQLU_APPLID_LEN + 1];
char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
sqluint32 backupBuffersNb;
struct sqlu_tablespace_bkrst_list backupTablespaceList;
struct sqlu_media_list backupTargetMediaList;
struct sqlu_media_entry backupTargetMediaEntries[1];
sqluint32 backupParallelismDegree;
sqluint32 backupImageSize;
SQLU_LSN startLSN;
SQLU_LSN endLSN;
char *logBuffer;
sqluint32 logBufferSize;
SQLU_RLOG_INFO readLogInfo;
int i;

printf("\n*****************************\n");
printf("*** ASYNCHRONOUS READ LOG ***\n");
printf("*****************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurlog -- Asynchronous Read Log\n");
printf("AND THE SQL STATEMENTS:\n");
printf("  CONNECT\n");
printf("  ALTER TABLE\n");
printf("  COMMIT\n");
printf("  INSERT\n");
printf("  DELETE\n");
printf("  ROLLBACK\n");
printf("  CONNECT RESET\n");
printf("TO READ LOG RECORDS FOR THE CURRENT CONNECTION.\n");
printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the database configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");

dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;
/* enable LOGRETAIN */
logretain = SQLF_LOGRETAIN_RECOVERY;
/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");
/* start the backup operation */
printf("\n  Backing up the '%s' database...\n", dbAlias);
backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
```

```
backupParallelismDegree = 1;
backupCallerAction = SQLUB_BACKUP;
/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");
while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  printf("\n  Continuing the backup operation...\n");
  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */

  backupCallerAction = SQLUB_CONTINUE;
  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}
printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
```

```
       backupStartTimestamp);
/* connect to the database */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}
/* invoke SQL statements to fill database log */
printf("\n  Invoke the following SQL statements:\n"
       "    ALTER TABLE emp_resume DATA CAPTURE CHANGES;\n"
       "    COMMIT;\n"
       "    INSERT INTO emp_resume\n"
       "      VALUES('000777', 'ascii', 'This is a new resume.');\n"
       "            ('777777', 'ascii', 'This is another new resume');\n"
       "    COMMIT;\n"
       "    DELETE FROM emp_resume WHERE empno = '000777';\n"
       "    DELETE FROM emp_resume WHERE empno = '777777';\n"
       "    COMMIT;\n"
       "    DELETE FROM emp_resume WHERE empno = '000140';\n"
       "    ROLLBACK;\n"
       "    ALTER TABLE emp_resume DATA CAPTURE NONE;\n"
       "    COMMIT;\n");
EXEC SQL ALTER TABLE emp_resume DATA CAPTURE CHANGES;
EMB_SQL_CHECK("SQL statement 1 -- invoke");
EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 2 -- invoke");
EXEC SQL INSERT INTO emp_resume
  VALUES('000777', 'ascii', 'This is a new resume.'),
        ('777777', 'ascii', 'This is another new resume');
EMB_SQL_CHECK("SQL statement 3 -- invoke");
EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 4 -- invoke");
EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
EMB_SQL_CHECK("SQL statement 5 -- invoke");
EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
EMB_SQL_CHECK("SQL statement 6 -- invoke");
EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 7 -- invoke");
EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
EMB_SQL_CHECK("SQL statement 8 -- invoke");
EXEC SQL ROLLBACK;
EMB_SQL_CHECK("SQL statement 9 -- invoke");
EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
EMB_SQL_CHECK("SQL statement 10 -- invoke");
EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 11 -- invoke");
printf("\n  Start reading database log.\n");
logBuffer = NULL;
logBufferSize = 0;
/* The API sqlurlog (Asynchronours Read Log) is used to extract records
   from the database logs, and to query the log manager for current
   log state information.
   This API can only be used on recoverable databases. */

/* Query the log manager for current log state information: */
rc = sqlurlog(SQLU_RLOG_QUERY,
              &startLSN,
              &endLSN,
              logBuffer,
              logBufferSize,
              &readLogInfo,
              &sqlca);
/*  DB2_API_CHECK("database log info -- get"); */
```

```
EXPECTED_WARN_CHECK("database log info -- get");
logBufferSize = 64 * 1024;
logBuffer = (char *)malloc(logBufferSize);
memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
memcpy(&endLSN, &(readLogInfo.curActiveLSN), sizeof(endLSN));
/* Extract a log record from the database logs, and
   read the first log sequence asynchronously: */
rc = sqlurlog(SQLU_RLOG_READ,
              &startLSN,
              &endLSN,
              logBuffer,
              logBufferSize,
              &readLogInfo,
              &sqlca);
if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  DB2_API_CHECK("database logs -- read");
}
else
{
  if (readLogInfo.logRecsWritten == 0)
  {
    printf("\n  Database log empty.\n");
  }
}
/* display log buffer */
rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  /* read the next log sequence */
  memcpy(&startLSN, &(readLogInfo.lastReadLSN), sizeof(startLSN));
  /* increase startLSN by 1 */
  startLSN.lsnChar[5] = startLSN.lsnChar[5] + 1;
  i = 5;
  while (startLSN.lsnChar[i] == 0 && i > 0)
  {
    startLSN.lsnChar[i - 1] = startLSN.lsnChar[i - 1] + 1;
    i = i - 1;
  }
  /* Extract a log record from the database logs, and
   read the next log sequence asynchronously: */
  rc = sqlurlog(SQLU_RLOG_READ,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
                &sqlca);
  if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    DB2_API_CHECK("database logs -- read");
  }
  /* display log buffer */
  rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
}
/* free the log buffer */
free(logBuffer);
/* disconnect from the database */
rc = DbDisconn(dbAlias);
if (rc != 0)
{
  return rc;
}
```

```
  return 0;
} /* DbLogRecordsForCurrentConnectionRead */
int LogBufferDisplay(char *logBuffer, sqluint32 numLogRecords)
{
  int rc = 0;
  sqluint32 logRecordNb;
  sqluint32 recordSize;
  sqluint16 recordType;
  sqluint16 recordFlag;
  char *recordBuffer;
  /* initialize recordBuffer */
  recordBuffer = logBuffer + sizeof(SQLU_LSN);
  for (logRecordNb = 0; logRecordNb < numLogRecords; logRecordNb++)
  {
    recordSize = *(sqluint32 *)(recordBuffer);
    recordType = *(sqluint16 *)(recordBuffer + 4);
    recordFlag = *(sqluint16 *)(recordBuffer + 6);
    rc = LogRecordDisplay(recordBuffer, recordSize, recordType, recordFlag);
    /* update recordBuffer */
    recordBuffer = recordBuffer + recordSize + sizeof(SQLU_LSN);
  }
  return 0;
} /* LogBufferDisplay */
int LogRecordDisplay(char *recordBuffer,
                     sqluint32 recordSize,
                     sqluint16 recordType,
                     sqluint16 recordFlag)
{
  int rc = 0;
  sqluint32 logManagerLogRecordHeaderSize;
  char *recordDataBuffer;
  sqluint32 recordDataSize;
  char *recordHeaderBuffer;
  sqluint8 componentIdentifier;
  sqluint32 recordHeaderSize;
  /* determine logManagerLogRecordHeaderSize */
  if ((char)recordType == 'C')
  { /* compensation */
    if (recordFlag & 0x0002)
    { /* propagatable */
      logManagerLogRecordHeaderSize = 32;
    }
    else
    {
      logManagerLogRecordHeaderSize = 26;
    }
  }
  else
  { /* non compensation */
    logManagerLogRecordHeaderSize = 20;
  }
  switch ((char)recordType)
  {
    case 'E':
    case 'M':
    case 'A':
      recordDataBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      recordDataSize = recordSize - logManagerLogRecordHeaderSize;
      rc = SimpleLogRecordDisplay(recordType,
                                  recordFlag,
                                  recordDataBuffer,
                                  recordDataSize);
      break;
```

```
    case 'N':
    case 'C':
      recordHeaderBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      componentIdentifier = *(sqluint8 *)recordHeaderBuffer;
      switch (componentIdentifier)
      {
        case 1:
          recordHeaderSize = 6;
          break;
        default:
          printf("    Unknown complex log record: %lu %c %u\n",
                 recordSize, recordType, componentIdentifier);
          return 1;
      }
      recordDataBuffer = recordBuffer +
                         logManagerLogRecordHeaderSize +
                         recordHeaderSize;
      recordDataSize = recordSize -
                       logManagerLogRecordHeaderSize -
                       recordHeaderSize;
      rc = ComplexLogRecordDisplay(recordType,
                                   recordFlag,
                                   recordHeaderBuffer,
                                   recordHeaderSize,
                                   componentIdentifier,
                                   recordDataBuffer,
                                   recordDataSize);
      break;
    default:
      printf("    Unknown log record: %lu %c\n",
             recordSize, (char)recordType);
      break;
  }
  return 0;
} /* LogRecordDisplay */
int SimpleLogRecordDisplay(sqluint16 recordType,
                           sqluint16 recordFlag,
                           char *recordDataBuffer,
                           sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint32 timeTransactionCommited;
  sqluint16 authIdLen;
  char authId[129];
  switch ((char)recordType)
  {
    case 'E':
      printf("\n    Record type: Local pending list\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = *(sqluint16 *)(recordDataBuffer + 4);
      memcpy(authId, (char *)(recordDataBuffer + 6), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'M':
      printf("\n    Record type: Normal commit\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = (sqluint16) (recordDataSize - 4);
      memcpy(authId, (char *)(recordDataBuffer + 4), authIdLen);
      authId[authIdLen] = '\0';
```

```
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'A':
      printf("\n    Record type: Normal abort\n");
      authIdLen = (sqluint16) (recordDataSize);
      memcpy(authId, (char *)(recordDataBuffer), authIdLen);
      authId[authIdLen] = '\0';
      printf("      authorization ID of the application: %s\n", authId);
      break;
    default:
      printf("    Unknown simple log record: %c %lu\n",
             (char)recordType, recordDataSize);
      break;
  }
  return 0;
} /* SimpleLogRecordDisplay */
int ComplexLogRecordDisplay(sqluint16 recordType,
                            sqluint16 recordFlag,
                            char *recordHeaderBuffer,
                            sqluint32 recordHeaderSize,
                            sqluint8 componentIdentifier,
                            char *recordDataBuffer,
                            sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint8 functionIdentifier;
  /* for insert, delete, undo delete */
  sqluint32 RID;
  sqluint16 subRecordLen;
  sqluint16 subRecordOffset;
  char *subRecordBuffer;
  /* for update */
  sqluint32 newRID;
  sqluint16 newSubRecordLen;
  sqluint16 newSubRecordOffset;
  char *newSubRecordBuffer;
  sqluint32 oldRID;
  sqluint16 oldSubRecordLen;
  sqluint16 oldSubRecordOffset;
  char *oldSubRecordBuffer;
  /* for alter table attributes */
  sqluint32 alterBitMask;
  sqluint32 alterBitValues;
  switch ((char)recordType)
  {
    case 'N':
      printf("\n    Record type: Normal\n");
      break;
    case 'C':
      printf("\n    Record type: Compensation\n");
      break;
    default:
      printf("\n    Unknown complex log record type: %c\n", recordType);
      break;
  }
  switch (componentIdentifier)
  {
    case 1:
      printf("      component ID: DMS log record\n");
      break;
```

```
  default:
    printf("      unknown component ID: %d\n", componentIdentifier);
    break;
}
functionIdentifier = *(sqluint8 *)(recordHeaderBuffer + 1);
switch (functionIdentifier)
{
  case 106:
    printf("      function ID: Delete Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 111:
    printf("      function ID: Undo Delete Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 118:
    printf("      function ID: Insert Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 120:
    printf("      function ID: Update Record\n");
    oldRID = *(sqluint32 *)(recordDataBuffer + 2);
    oldSubRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    oldSubRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    newRID = *(sqluint32 *)(recordDataBuffer +
                            12 + oldSubRecordLen + recordHeaderSize + 2);
    newSubRecordLen = *(sqluint16 *)(recordDataBuffer +
                                     12 + oldSubRecordLen +
                                     recordHeaderSize + 6);
    newSubRecordOffset =
      *(sqluint16 *)(recordDataBuffer + 12 + oldSubRecordLen +
                     recordHeaderSize + 10);
    printf("        oldRID: %lu\n", oldRID);
    printf("        old subrecord length: %u\n", oldSubRecordLen);
    printf("        old subrecord offset: %u\n",
           oldSubRecordOffset);
    oldSubRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(oldSubRecordBuffer, oldSubRecordLen);
    printf("        newRID: %lu\n", newRID);
    printf("        new subrecord length: %u\n", newSubRecordLen);
    printf("        new subrecord offset: %u\n",
           newSubRecordOffset);
```

```
  newSubRecordBuffer = recordDataBuffer +
    12 + oldSubRecordLen + recordHeaderSize + 12;
  rc = LogSubRecordDisplay(newSubRecordBuffer, newSubRecordLen);
  break;
case 124:
  printf("      function ID:  Alter Table Attribute\n");
  alterBitMask = *(sqluint32 *)(recordDataBuffer + 2);
  alterBitValues = *(sqluint32 *)(recordDataBuffer + 6);
  if (alterBitMask & 0x00000001)
  {
    /* Alter the value of the 'propagation' attribute: */
    printf("        Propagation attribute is changed to: ");
    if (alterBitValues & 0x00000001)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00000002)
  {
    /* Alter the value of the 'pending' attribute: */
    printf("        Pending attribute is changed to: ");
    if (alterBitValues & 0x00000002)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00010000)
  {
    /* Alter the value of the 'append mode' attribute: */
    printf("        Append Mode attribute is changed to: ");
    if (alterBitValues & 0x00010000)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00200000)
  {
    /* Alter the value of the 'LF Propagation' attribute: */
    printf("        LF Propagation attribute is changed to: ");
    if (alterBitValues & 0x00200000)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00400000)
  {
    /* Alter the value of the 'LOB Propagation' attribute: */
```

```
        printf("       LOB Propagation attribute is changed to: ");
        if (alterBitValues & 0x00400000)
        {
          printf("ON\n");
        }
        else
        {
          printf("OFF\n");
        }
      }
      break;
    default:
      printf("      unknown function identifier: %u\n",
             functionIdentifier);
      break;
  }
  return 0;
} /* ComplexLogRecordDisplay */
int LogSubRecordDisplay(char *recordBuffer, sqluint16 recordSize)
{
  int rc = 0;
  sqluint8 recordType;
  sqluint8 updatableRecordType;
  sqluint16 userDataFixedLength;
  char *userDataBuffer;
  sqluint16 userDataSize;
  recordType = *(sqluint8 *)(recordBuffer);
  if (recordType != 0 && recordType != 4)
  {
    printf("        Unknown subrecord type.\n");
  }
  else if (recordType == 4)
  {
    printf("        subrecord type: Special control\n");
  }
  else
  {
    /* recordType == 0 */
    printf("        subrecord type: Updatable, ");
    updatableRecordType = *(sqluint8 *)(recordBuffer + 4);
    if (updatableRecordType != 1)
    {
      printf("Internal control\n");
    }
    else
    {
      printf("Formatted user data\n");
      userDataFixedLength = *(sqluint16 *)(recordBuffer + 6);
      printf("        user data fixed length: %u\n",
             userDataFixedLength);
      userDataBuffer = recordBuffer + 8;
      userDataSize = recordSize - 8;
      rc = UserDataDisplay(userDataBuffer, userDataSize);
    }
  }
  return 0;
} /* LogSubRecordDisplay */
int UserDataDisplay(char *dataBuffer, sqluint16 dataSize)
{
  int rc = 0;
  sqluint16 line, col;
  printf("        user data:\n");
  for (line = 0; line * 10 < dataSize; line = line + 1)
```

```
    {
      printf("        ");
      for (col = 0; col < 10; col = col + 1)
      {
        if (line * 10 + col < dataSize)
        {
          printf("%02X ", dataBuffer[line * 10 + col]);
        }
        else
        {
          printf("   ");
        }
      }
      printf("*");
      for (col = 0; col < 10; col = col + 1)
      {
        if (line * 10 + col < dataSize)
        {
          if (isalpha(dataBuffer[line * 10 + col]) ||
              isdigit(dataBuffer[line * 10 + col]))
          {
            printf("%c", dataBuffer[line * 10 + col]);
          }
          else
          {
            printf(".");
          }
        }
        else
        {
          printf(" ");
        }
      }
      printf("*");
      printf("\n");
    }
    return 0;
  } /* UserDataDisplay */
  int DbRecoveryHistoryFileRead(char dbAlias[])
  {
    int rc = 0;
    struct sqlca sqlca;
    struct db2HistoryOpenStruct dbHistoryOpenParam;
    sqluint32 numEntries;
    sqluint16 recoveryHistoryFileHandle;
    sqluint32 entryNb;
    struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
    struct db2HistoryData histEntryData;
    printf("\n*****************************************\n");
    printf("*** READ A DATABASE RECOVERY HISTORY FILE ***\n");
    printf("*****************************************\n");
    printf("\nUSE THE DB2 APIs:\n");
    printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
    printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
    printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
    printf("TO READ A DATABASE RECOVERY HISTORY FILE.\n");
    /* initialize the data structures */
    dbHistoryOpenParam.piDatabaseAlias = dbAlias;
    dbHistoryOpenParam.piTimestamp = NULL;
    dbHistoryOpenParam.piObjectName = NULL;
    dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
    dbHistoryEntryGetParam.pioHistData = &histEntryData;
    dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
```

```
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  if (rc != 0)
  {
    return rc;
  }
  /*****************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("\n  Open recovery history file for '%s' database.\n", dbAlias);
  /* open the recovery history file to scan */
  db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");
  numEntries = dbHistoryOpenParam.oNumRows;
  /* dbHistoryOpenParam.oHandle returns the handle for scan access */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
  /*********************************************/
  /* READ AN ENTRY IN THE RECOVERY HISTORY FILE */
  /*********************************************/
  for (entryNb = 0; entryNb < numEntries; entryNb = entryNb + 1)
  {
    printf("\n  Read entry number %u.\n", entryNb);
    /* get the next entry from the recovery history file */
    db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
    DB2_API_CHECK("database recovery history file entry -- read")
    /* display the entries in the recovery history file */
    printf("\n  Display entry number %u.\n", entryNb);
    rc = HistoryEntryDisplay(histEntryData);
  }
  /*****************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("\n  Close recovery history file for '%s' database.\n", dbAlias);
  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");
  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);
  return 0;
} /* DbRecoveryHistoryFileRead */
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;
  strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");
  pHistEntryData->oObjectPart.pioData = malloc(17 + 1);
  pHistEntryData->oObjectPart.iLength = 17 + 1;
  pHistEntryData->oEndTime.pioData = malloc(12 + 1);
  pHistEntryData->oEndTime.iLength = 12 + 1;
  pHistEntryData->oFirstLog.pioData = malloc(8 + 1);
  pHistEntryData->oFirstLog.iLength = 8 + 1;
  pHistEntryData->oLastLog.pioData = malloc(8 + 1);
  pHistEntryData->oLastLog.iLength = 8 + 1;
  pHistEntryData->oID.pioData = malloc(128 + 1);
  pHistEntryData->oID.iLength = 128 + 1;
  pHistEntryData->oTableQualifier.pioData = malloc(128 + 1);
  pHistEntryData->oTableQualifier.iLength = 128 + 1;
  pHistEntryData->oTableName.pioData = malloc(128 + 1);
  pHistEntryData->oTableName.iLength = 128 + 1;
  pHistEntryData->oLocation.pioData = malloc(128 + 1);
  pHistEntryData->oLocation.iLength = 128 + 1;
  pHistEntryData->oComment.pioData = malloc(128 + 1);
```

```
  pHistEntryData->oComment.iLength = 128 + 1;
  pHistEntryData->oCommandText.pioData = malloc(128 + 1);
  pHistEntryData->oCommandText.iLength = 128 + 1;
  pHistEntryData->poEventSQLCA =
    (struct sqlca *)malloc(sizeof(struct sqlca));
  pHistEntryData->poTablespace = (db2Char *)malloc(3 * sizeof(db2Char));
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
    pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
  }
  pHistEntryData->iNumTablespaces = 3;
  return 0;
} /* HistoryEntryDataFieldsAlloc */
int HistoryEntryDisplay(struct db2HistoryData histEntryData)
{
  int rc = 0;
  char buf[129];
  sqluint32 tsNb;
  memcpy(buf, histEntryData.oObjectPart.pioData,
              histEntryData.oObjectPart.oLength);
  buf[histEntryData.oObjectPart.oLength] = '\0';
  printf("    object part: %s\n", buf);
  memcpy(buf, histEntryData.oEndTime.pioData,
              histEntryData.oEndTime.oLength);
  buf[histEntryData.oEndTime.oLength] = '\0';
  printf("    end time: %s\n", buf);
  memcpy(buf, histEntryData.oFirstLog.pioData,
              histEntryData.oFirstLog.oLength);
  buf[histEntryData.oFirstLog.oLength] = '\0';
  printf("    first log: %s\n", buf);
  memcpy(buf, histEntryData.oLastLog.pioData,
              histEntryData.oLastLog.oLength);
  buf[histEntryData.oLastLog.oLength] = '\0';
  printf("    last log: %s\n", buf);
  memcpy(buf, histEntryData.oID.pioData, histEntryData.oID.oLength);
  buf[histEntryData.oID.oLength] = '\0';
  printf("    ID: %s\n", buf);
  memcpy(buf, histEntryData.oTableQualifier.pioData,
              histEntryData.oTableQualifier.oLength);
  buf[histEntryData.oTableQualifier.oLength] = '\0';
  printf("    table qualifier: %s\n", buf);
  memcpy(buf, histEntryData.oTableName.pioData,
              histEntryData.oTableName.oLength);
  buf[histEntryData.oTableName.oLength] = '\0';
  printf("    table name: %s\n", buf);
  memcpy(buf, histEntryData.oLocation.pioData,
              histEntryData.oLocation.oLength);
  buf[histEntryData.oLocation.oLength] = '\0';
  printf("    location: %s\n", buf);
  memcpy(buf, histEntryData.oComment.pioData,
              histEntryData.oComment.oLength);
  buf[histEntryData.oComment.oLength] = '\0';
  printf("    comment: %s\n", buf);
  memcpy(buf, histEntryData.oCommandText.pioData,
              histEntryData.oCommandText.oLength);
  buf[histEntryData.oCommandText.oLength] = '\0';
  printf("    command text: %s\n", buf);
  printf("    history file entry ID: %u\n", histEntryData.oEID.ioHID);
  printf("    table spaces:\n");
  for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb = tsNb + 1)
  {
    memcpy(buf, histEntryData.poTablespace[tsNb].pioData,
```

```
                histEntryData.poTablespace[tsNb].oLength);
    buf[histEntryData.poTablespace[tsNb].oLength] = '\0';
    printf("      %s\n", buf);
  }
  printf("    type of operation: %c\n", histEntryData.oOperation);
  printf("    granularity of the operation: %c\n", histEntryData.oObject);
  printf("    operation type: %c\n", histEntryData.oOptype);
  printf("    entry status: %c\n", histEntryData.oStatus);
  printf("    device type: %c\n", histEntryData.oDeviceType);
  printf("    SQLCA:\n");
  printf("      sqlcode: %ld\n", histEntryData.poEventSQLCA->sqlcode);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, 5);
  buf[5] = '\0';
  printf("      sqlstate: %s\n", buf);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc,
              histEntryData.poEventSQLCA->sqlerrml);
  buf[histEntryData.poEventSQLCA->sqlerrml] = '\0';
  printf("      message: %s\n", buf);
  return 0;
} /* HistoryEntryDisplay */
int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;
  free(pHistEntryData->oObjectPart.pioData);
  free(pHistEntryData->oEndTime.pioData);
  free(pHistEntryData->oFirstLog.pioData);
  free(pHistEntryData->oLastLog.pioData);
  free(pHistEntryData->oID.pioData);
  free(pHistEntryData->oTableQualifier.pioData);
  free(pHistEntryData->oTableName.pioData);
  free(pHistEntryData->oLocation.pioData);
  free(pHistEntryData->oComment.pioData);
  free(pHistEntryData->oCommandText.pioData);
  free(pHistEntryData->poEventSQLCA);
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
  }
  free(pHistEntryData->poTablespace);
  return 0;
} /* HistoryEntryDataFieldsFree */
int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[],
                                          char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1];
  char newComment[DB2HISTORY_COMMENT_SZ + 1];
  struct db2HistoryUpdateStruct dbHistoryUpdateParam;
  printf("\n*****************************************************\n");
  printf("*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***\n");
  printf("*****************************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
  printf("  db2HistoryUpdate -- Update Recovery History File\n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
```

```
printf("TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.\n");
/* initialize data structures */
dbHistoryOpenParam.piDatabaseAlias = dbAlias;
dbHistoryOpenParam.piTimestamp = NULL;
dbHistoryOpenParam.piObjectName = NULL;
dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
dbHistoryEntryGetParam.pioHistData = &histEntryData;
dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
rc = HistoryEntryDataFieldsAlloc(&histEntryData);
if (rc != 0)
{
  return rc;
}
/*****************************************/
/* OPEN THE DATABASE RECOVERY HISTORY FILE */
/*****************************************/
printf("\n  Open the recovery history file for '%s' database.\n", dbAlias);
/* The API db2HistoryOpenScan starts a recovery history file scan */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");
/* dbHistoryOpenParam.oHandle returns the handle for scan access */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
/*************************************************/
/* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE */
/*************************************************/
printf("\n  Read the first entry in the recovery history file.\n");
/* The API db2HistoryGetEntry gets the next entry from the recovery
   history file. */
db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("first recovery history file entry -- read");
printf("\n  Display the first entry.\n");
/* HistoryEntryDisplay is a support function used to display the entries
   in the recovery history file. */
rc = HistoryEntryDisplay(histEntryData);
/* update the first history file entry */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}
strcpy(newLocation, "this is the NEW LOCATION");
strcpy(newComment, "this is the NEW COMMENT");
printf("\n  Update the first entry in the history file:\n");
printf("    new location = '%s'\n", newLocation);
printf("    new comment = '%s'\n", newComment);
dbHistoryUpdateParam.piNewLocation = newLocation;
dbHistoryUpdateParam.piNewDeviceType = NULL;
dbHistoryUpdateParam.piNewComment = newComment;
dbHistoryUpdateParam.iEID.ioNode = histEntryData.oEID.ioNode;
dbHistoryUpdateParam.iEID.ioHID = histEntryData.oEID.ioHID;
/* The API db2HistoryUpdate can be used to update the location,
   device type, or comment in a history file entry. */
/* Call this API to update the location and comment of the first
   entry in the history file: */
db2HistoryUpdate(db2Version710, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("first history file entry -- update");
rc = DbDisconn(dbAlias);
if (rc != 0)
{
  return rc;
}
```

```
  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("\n  Close recovery history file for '%s' database.\n", dbAlias);
  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");
  /*******************************************/
  /* RE-OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************
  printf("\n  Open the recovery history file for '%s' database.\n", dbAlias);
  /* starts a recovery history file scan */
  db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
  printf("\n  Read the first recovery history file entry.\n");

  /*************************************************************************/
  /* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE AFTER MODIFICATION */
  /*************************************************************************/
  db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
  DB2_API_CHECK("first recovery history file entry -- read");
  printf("\n  Display the first entry.\n");
  rc = HistoryEntryDisplay(histEntryData);

  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("\n  Close the recovery history file for '%s' database.\n",
         dbAlias);
  /* ends the recovery history file scan */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");
  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);
  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */
int DbRecoveryHistoryFilePrune(char dbAlias[], char user[], char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2PruneStruct histPruneParam;
  char timeStampPart[14 + 1];
  printf("\n*************************************\n");
  printf("*** PRUNE THE RECOVERY HISTORY FILE ***\n");
  printf("*************************************\n");
  printf("\nUSE THE DB2 API:\n");
  printf("  db2Prune -- Prune Recovery History File\n");
  printf("AND THE SQL STATEMENTS:\n");
  printf("  CONNECT\n");
  printf("  CONNECT RESET\n");
  printf("TO PRUNE THE RECOVERY HISTORY FILE.\n");
  /* Connect to the database: */
  rc = DbConn(dbAlias, user, pswd);
  if (rc != 0)
  {
    return rc;
  }
  /* Prune the recovery history file: */
  printf("\n  Prune the recovery history file for '%s' database.\n",
         dbAlias);
```

```
  /* timeStampPart is a pointer to a string specifying a time stamp or
     log sequence number. Time stamp is used here to select records for
     deletion. All entries equal to or less than the time stamp will be
     deleted. */
  histPruneParam.piString = timeStampPart;
  strcpy(timeStampPart, "2010"); /* year 2010 */
  /* The action DB2PRUNE_ACTION_HISTORY removes history file entries: */
  histPruneParam.iAction = DB2PRUNE_ACTION_HISTORY;
  /* The option DB2PRUNE_OPTION_FORCE forces the removal of the last backup: */
  histPruneParam.iOptions = DB2PRUNE_OPTION_FORCE;
  /* db2Prune can be called to delete entries from the recovery history file
     or log files from the active log path. Here we call it to delete
     entries from the recovery history file.
     You must have SYSADM, SYSCTRL, SYSMAINT, or DBADM authority to prune
     the recovery history file. */
  db2Prune(db2Version710, &histPruneParam, &sqlca);
  DB2_API_CHECK("recovery history file -- prune");
  /* Disconnect from the database: */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }
  return 0;
} /* DbRecoveryHistoryFilePrune */
```

부록F. CLP 스크립트 복구

다음의 DB2 명령 스크립트는 CLP 명령을 사용하여 다음을 수행하는 방법을 보여줍니다.

- 데이터베이스 백업
- 데이터베이스 복원
- 데이터베이스 롤 포워드 복구

SAMPLE 데이터베이스가 존재하지만 사용 중이 아닌지 확인하십시오. SAMPLE 데이터베이스에 대한 자세한 정보는 *SQL 참조서*를 참조하십시오. DB2 명령행 처리기에 대한 일반 정보는 *Command Reference*를 참조하십시오.

스크립트의 Windows 호환 및 UNIX 호환 버전에 대해 설명합니다.

Windows 운영 체제를 위한 명령 스크립트 샘플

Windows NT 또는 Windows 2000에서 스크립트를 수행하려면 다음과 같이 하십시오.

1. 스크립트를 이름이 지정된 파일로 저장하십시오(예: `backrest.db2`).
2. 데이터베이스 관리 프로그램이 수행 중이 아니면, DB2 명령 창에서 **db2start** 명령을 발행하십시오. CLP 사용 DB2 창을 열고 Windows 운영 체제에서 DB2 명령행 환경을 초기화하려면, 명령 프롬프트에서 **db2cmd**를 발행하십시오.
3. `db2 -f backrest.db2 -t`를 입력하십시오.

다음은 이 스크립트가 리턴한 출력의 예입니다.

```
D:\>db2 -f backrest.db2 -t
This is CLP script: backrest.db2


Deleting old SAMPLE database backup images...

process SAMPLE.0\DB2\NODE0000\CATN0000
process 20010403

Updating the database configuration parameter LOGRETAIN to 'ON'...
```

```
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.


Backing up the SAMPLE database...


Backup successful. The timestamp for this backup image is : 20010403131027


Restoring the SAMPLE database as TESTBACK (1st pass)...

SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
DB20000I  The RESTORE DATABASE command completed successfully.

Listing the table spaces for the TESTBACK database...


           Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined


Defining new table space containers for Tablespace 2...

DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

Listing table space containers for Tablespace 2 (TESTBACK database)...
```

```
            Tablespace Containers for Tablespace 2

 Container ID                         = 0
 Name                                 = c:\ts2con1
 Type                                 = Path


Restoring the SAMPLE database as TESTBACK (2nd pass)...

DB20000I  The RESTORE DATABASE command completed successfully.

Rolling the TESTBACK database forward...


                                 Rollforward Status

 Input database alias                 = testback
 Number of nodes have returned status = 1

 Node number                          = 0
 Rollforward status                   = not pending
 Next log file to be read             =
 Log files processed                  =  -
 Last committed transaction           = 2001-04-03-03.16.07.000000

DB20000I  The ROLLFORWARD command completed successfully.

Dropping the TESTBACK database...

DB20000I  The DROP DATABASE command completed successfully.

Terminating the command line processor's back-end process...

DB20000I  The TERMINATE command completed successfully.


D:\>
```

다음은 스크립트에 대해 나열되는 소스입니다.

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--               from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--              as the statement termination character.

!ECHO This is CLP script: backrest.db2;

-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;
```

```
!ECHO Deleting old SAMPLE database backup images...;
!rd! SAMPLE.0\DB2\NODE0000\CATN0000;

!ECHO Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

!ECHO Backing up the SAMPLE database...;
backup db sample;

!ECHO Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

!ECHO Listing the table spaces for the TESTBACK database...;
list tablespaces;

!ECHO Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "c:\ts2con1");

!ECHO Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

!ECHO Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

!ECHO Rolling the TESTBACK database forward...;
rollforward db testback stop;

!ECHO Dropping the TESTBACK database...;
drop db testback;

!ECHO Terminating the command line processor's back-end process...;
terminate;

-- End file
```

UNIX 기반 시스템에 대한 명령 스크립트 샘플

UNIX 기반 시스템에서 스크립트를 수행하려면 다음과 같이 하십시오.

1. 스크립트를 이름이 지정된 파일로 저장하십시오(예: backrest.db2).
2. 데이터베이스 관리 프로그램이 수행 중이 아니면, 명령 프롬프트에서 **db2start** 명령을 발행하십시오.
3. db2 -f backrest.db2 -t를 입력하십시오.

다음은 이 스크립트가 리턴한 출력의 예입니다.

```
sunfish /export/home2/falexand/samples/clp>db2 -f backrest.db2 -t
This is CLP script: backrest.db2
Deleting old SAMPLE database backup images...
Updating the database configuration parameter LOGRETAIN to 'ON'...
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.
```

```
Backing up the SAMPLE database...
Backup successful. The timestamp for this backup image is : 20010531172525
Restoring the SAMPLE database as TESTBACK (1st pass)...
SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
DB20000I  The RESTORE DATABASE command completed successfully.
Listing the table spaces for the TESTBACK database...
          Tablespaces for Current Database
 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined
 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined
 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined
Defining new table space containers for Tablespace 2...
DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.
Listing table space containers for Tablespace 2 (TESTBACK database)...
           Tablespace Containers for Tablespace 2
 Container ID                         = 0
 Name                                 = /export/home2/falexand/falexand...
                                          .../NODE0000/SQL00020/ts2con1
 Type                                 = Path
Restoring the SAMPLE database as TESTBACK (2nd pass)...
DB20000I  The RESTORE DATABASE command completed successfully.
Rolling the TESTBACK database forward...
                              Rollforward Status
 Input database alias                 = testback
 Number of nodes have returned status = 1
 Node number                          = 0
 Rollforward status                   = not pending
 Next log file to be read             =
 Log files processed                  =  -
 Last committed transaction           = 2001-05-29-21.20.16.000000
DB20000I  The ROLLFORWARD command completed successfully.
Dropping the TESTBACK database...
DB20000I  The DROP DATABASE command completed successfully.
Terminating the command line processor's back-end process...
DB20000I  The TERMINATE command completed successfully.
```

Following is the source listing for the script:

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input --
from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;) --
as the statement termination character.

echo This is CLP script: backrest.db2;

-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP --
is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;

echo Deleting old SAMPLE database backup images...;
!rm -f ./SAMPLE.0.*;

echo Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

echo Backing up the SAMPLE database...;
backup db sample;

echo Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

echo Listing the table spaces for the TESTBACK database...;
list tablespaces;

echo Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "ts2con1");

echo Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

echo Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

echo Rolling the TESTBACK database forward...;
rollforward db testback stop;

echo Dropping the TESTBACK database...;
drop db testback;

echo Terminating the command line processor's back-end process...;
terminate;

-- End file
```

부록G. Tivoli Storage Manager

DB2 백업 또는 복원 유틸리티를 호출할 경우, 백업 또는 복원 조작을 관리하기 위해 TSM(Tivoli Storage Manager, 이전의 ADSM) 제품을 사용할 것을 지정할 수 있습니다. DB2와 함께 TSM 클라이언트 버전 3.1.x.3 이상을 사용할 수 있습니다.

UNIX 기반 플랫폼에서 Tivoli Storage Manager Client 설정

데이터베이스 관리 프로그램에서 옵션을 사용하기 위해서는 먼저 다음과 같은 설정 활동을 수행해야 합니다.

1. SunOS 및 Solaris 환경에서 다음 단계를 수행하십시오. (다른 UNIX 기반 플랫폼의 경우, 2단계에서 시작하십시오.)
   a. SunOS 5.5.1 또는 Solaris 2.5.1과 같이 필수 운영 체제 레벨이 설치되어 있는지 확인하십시오.
   b. TSM 클라이언트 버전 3.1.x.3 이상을 설치하십시오. 이 버전의 클라이언트를 설치하기 전에 이전 TSM 패키지 모두를 제거하십시오.
   c. TSM이 /opt/IBMDSMap5, /opt/IBMDSMba5 및 /opt/IBMDSMsa5 디렉토리에 설치되어 있는지 확인하십시오.
   d. 없을 경우, /usr/lib 디렉토리에서 다음과 같은 기호 링크를 작성하십시오.

      ```
      libApiDS.so -> libApiDS.so.1
      libApiDS.so.1 -> /opt/IBMDSMap5/api/libApiDS.so.2
      ```

2. TSM 사용자 구성 옵션 파일 /usr/sbin/dsm.opt 및 TSM 시스템 구성 옵션 파일 /usr/sbin/dsm.sys를 작성하거나 수정하여 사용자 환경에 맞추십시오.
3. SunOS 및 Solaris 환경에서 다음 단계를 수행하십시오. (다른 UNIX 기반 플랫폼의 경우, 480 페이지의 4단계에서 계속하십시오.)

a. /usr/sbin/dsm.opt 및 /usr/sbin/dsm.sys를 디렉토리 /opt/IBMDSMap5로 복사하십시오.

b. /opt/IBMDSMap5/solaris/dsmaptica를 디렉토리 /opt/IBMDSMap5로 복사하십시오.

4. TSM이 사용하는 환경 변수를 설정하십시오.

DSMI_DIR API 신뢰 에이전트 파일(dsmapicta 또는 dsmtca)이 위치한 사용자 정의 디렉토리 경로를 식별합니다. SunOS 및 Solaris 환경에서는 /opt/IBMDSMap5로 설정되어야 합니다.

DSMI_CONFIG

dsm.opt 파일에 대해 사용자 정의 디렉토리 경로를 식별합니다. 이 경로에는 TSM 사용자 옵션이 들어 있습니다. 다른 두 개의 변수와 달리, 이 변수는 완전한 경로 및 파일 이름을 포함해야 합니다. SunOS 및 Solaris 환경에서는 /opt/IBMDSMap5/dsm.opt로 설정되어야 합니다.

DSMI_LOG 오류 로그(dsierror.log)가 작성될 사용자 정의 디렉토리 경로를 식별합니다.

5. TSM 암호를 설정하십시오.

Tivoli 클라이언트가 TSM 서버와 인터페이스할 수 있으려면, 서버에 대한 암호가 있어야 합니다. 실행 가능 파일인 dsmapipw는 인스턴스 소유자의 INSTHOME/sqllib/adsm 디렉토리에 설치되어 있습니다. 이 실행 파일로 사용자는 TSM 암호를 설정 및 재설정할 수 있습니다.

dsmapipw 명령을 실행하려면 『루트』 사용자로서 로그인되어야 합니다. 이 명령이 실행될 때, 다음과 같은 정보가 프롬프트됩니다.

- TSM 서버에 의해 인식되는 TSM 노드의 현재 암호인 *이전 암호*. 이 명령을 처음 실행할 때, 암호는 사용자의 노드가 TSM 서버에 등록될 때 TSM 관리자가 제공하는 암호입니다.
- TSM 서버에 저장될 TSM 노드의 새로운 암호인 *새 암호*. (입력 오류를 점검하기 위해 새 암호를 두 번 입력하도록 프롬프트됩니다.)

주: 백업 또는 복원 유틸리티를 호출하는 사용자는 이 암호를 알 필요가 없습니다. 초기 연결을 위해 암호를 설정하고 TSM 서버에 암호가 재설정된 이후에만 dsmapipw 명령을 수행해야 합니다.

6. 데이터베이스 관리 프로그램을 수행 중인 경우, 다음을 수행해야 합니다.
   - **db2stop** 명령을 사용하여 데이터베이스 관리 프로그램 중지
   - **db2start** 명령을 사용하여 데이터베이스 관리 프로그램 시작

기타 플랫폼에서 Tivoli Storage Manager Client 설정

데이터베이스 관리 프로그램에서 옵션을 사용하기 위해서는 먼저 다음과 같은 설정 활동을 수행해야 합니다.

1. TSM이 사용하는 환경 변수를 설정하십시오.

   DSMI_DIR API 신뢰 에이전트 파일(dsmapicta 또는 dsmtca)이 위치한 사용자 정의 디렉토리 경로를 식별합니다.

   DSMI_CONFIG
   dsm.opt 파일에 대해 사용자 정의 디렉토리 경로를 식별합니다. 이 경로에는 TSM 사용자 옵션이 들어 있습니다. 다른 두 개의 변수와 달리, 이 변수는 완전한 경로 및 파일 이름을 포함해야 합니다.

   DSMI_LOG 오류 로그(dsierror.log)가 작성될 사용자 정의 디렉토리 경로를 식별합니다.

2. 사용자의 운영 체제에 적용 가능하면, TSM 시스템 구성 옵션 파일(dsm.sys)을 작성(또는 변경)하십시오.
3. dsm.opt TSM 사용자 구성 옵션 파일을 작성(또는 변경)하십시오. 환경 변수 DSMI_CONFIG는 이 파일을 카리킵니다.
4. TSM 암호를 설정하십시오.

   Tivoli 클라이언트가 TSM 서버와 인터페이스할 수 있으려면, 서버에 대한 암호가 있어야 합니다. 실행 가능 파일인 dsmapipw는 인스턴스 소유자의 sqllib\adsm 디렉토리에 설치되어 있습니다. 이 실행 파일로 사용자는 TSM 암호를 설정 및 재설정할 수 있습니다.

dsmapipw 명령을 실행하려면, 지역 관리자로서 로그인되어야 합니다. 이 명령이 수행될 때, 다음과 같은 정보가 프롬프트됩니다.

- TSM 서버에 의해 인식되는 TSM 노드의 현재 암호인 *이전 암호*. 이 명령을 처음 수행할 때, 암호는 사용자의 노드가 TSM 서버에 등록될 때 TSM 관리자가 제공하는 암호입니다.
- TSM 서버에 저장될 TSM 노드의 새로운 암호인 *새 암호*. (입력 오류를 점검하기 위해 새 암호를 두 번 입력하도록 프롬프트됩니다.)

주: 백업 또는 복원 유틸리티를 호출하는 사용자는 이 암호를 알 필요가 없습니다. 초기 연결을 위해 암호를 설정하고 TSM 서버에 암호가 재설정된 이후에만 dsmapipw 명령을 수행해야 합니다.

5. 데이터베이스 관리 프로그램을 수행 중인 경우, 다음을 수행해야 합니다.
   - **db2stop** 명령을 사용하여 데이터베이스 관리 프로그램 중지
   - **db2start** 명령을 사용하여 데이터베이스 관리 프로그램 시작

Tivoli Storage Manager 사용시 고려사항

TSM 안에서 특정 기능을 사용하려면, 기능을 사용하고 있는 오브젝트의 완전한 경로 이름을 제공해야 합니다. (Windows NT 운영 체제 및 OS/2에서 \은 / 대신 사용된다는 점을 기억하십시오.) 완전한 경로 이름은 다음과 같습니다.

- 전체 데이터베이스 백업 오브젝트는 /<database>/NODEnnnn/FULL_BACKUP.timestamp.seq_no입니다.
- 테이블 공간 백업 오브젝트는 /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no입니다.
- 로드 사본 오브젝트는 /<database>/NODEnnnn/LOAD_COPY.timestamp.seq_no입니다.

여기서, <database>는 데이터베이스 별명 이름이고, NODEnnnn은 노드 번호입니다. 대문자로 표시된 이름은 표시된 대로 입력해야 합니다.

- 같은 데이터베이스 별명 이름을 사용하여 다중 백업 이미지가 있는 경우, 시간소인 및 순차 번호는 완전한 이름에서 구별되는 부분이 됩니다. 사용할 백업 버전을 결정하기 위해서는 TSM을 조회해야 합니다.

- 개별 백업 이미지는 TSM 그래픽 사용자 인터페이스에 알려지지 않았습니다. 백업 이미지는 TSM이 관리하는 파일 공간으로 풀됩니다. 개별 백업 이미지는 TSM API를 통하거나 이 API를 사용하는 **db2adutl**을 통해서만 조작될 수 있습니다(350 페이지의『db2adutl - TSM 아카이브 이미지 작업』참조).
- TSM 서버는 Tivoli 클라이언트가 서버의 구성 파일에서 COMMTIMEOUT 매개변수로 지정된 시간 내에 응답하지 않을 경우, 세션을 종료합니다. 다음과 같은 세 가지의 요인으로 시간종료 문제점이 발생할 수 있습니다.
  - COMMTIMEOUT 매개변수가 TSM 서버에서 너무 낮게 설정됩니다. 예를 들어, 복원 조작 동안 대형 DMS 테이블 공간을 작성하면 시간종료될 수 있습니다. 이 매개변수에 권장된 값은 6000초입니다.
  - DB2 백업 또는 복원 버퍼가 너무 커질 수 있습니다.
  - 온라인 백업 조작 동안 데이터베이스 활동이 너무 많을 수 있습니다.
- 데이터베이스 관리 프로그램은 TSM 전체 백업 옵션을 사용합니다. TSM 증분 백업 조작은 지원되지 않습니다.
- 처리량을 증가시키기 위해서는 다중 세션을 사용하십시오.
- UNIX 기반이 아닌 플랫폼에서, DB2 백업 및 복원 유틸리티는 하나 이상의 TSM 세션을 허용하지 않습니다.

Windows 운영 체제 및 OS/2에 있는 현재 Tivoli 클라이언트는 재입력이 가능하므로 단일 머신에서 백업, 복원 또는 로드 유틸리티를 사용하여 여러 입출력 세션을 안전하게 작성할 수 있습니다. 그러나 사용자는 설치된 TSM 클라이언트의 버전이 이 기능을 지원하도록 확인해야 합니다.

단일 노드 구성에서, 사용자가 다음과 같은 BACKUP DATABASE 명령을 발행할 경우,

```
db2 backup db sample use tsm open 3 sessions
```

DB2는 TSM이 다중 세션을 지원하지 않음을 검출하고 오류 메시지를 리턴합니다. (이는 또한 TSM을 사용하여 COPY YES 옵션으로 호출한 로드 조작에도 적용됩니다.)

그러나 Windows NT상의 MLN(Multiple Logical Node) 구성에서, DB2는 각 논리 노드가 하나의 세션만을 작성하려고 하면 단일 머신에서 다중 세션이 사용됨

을 발견못할 수도 있음에 주의하십시오. 이러한 이유로 관련 TSM 클라이언트가 재입력을 지원함을 MLN 구성이 검증하는 것은 매우 중요합니다. 다중 논리 노드가 TSM을 사용하여 병렬로 백업, 복원 또는 로드중이면, DB2는 논리 노드가 실제로 같은 하드웨어에 상주하는 경우라도 각 노드가 단일 세션을 사용하려고 하면 조작을 진행하도록 합니다. 이것은 백업 시도가 실패하게 하며, 로드 프로세스를 정지시키므로, 가장 최근의 TSM 클라이언트 없이 시도해서는 안됩니다.

TSM에서 백업 및 로그 아카이브 관리

db2adutl 유틸리티를 통해 백업 이미지, 로그, TSM을 사용하여 저장된 로드 사본 이미지를 조회, 발췌 및 삭제할 수 있습니다. 유틸리티는 UNIX 플랫폼의 `INSTHOME/sqllib/misc` 디렉토리에 설치되며, Windows 운영 체제 및 OS/2의 `\sqllib\misc` 디렉토리에 설치됩니다. 이 유틸리티에 대한 자세한 정보는 350 페이지의 『db2adutl - TSM 아카이브 이미지 작업』을 참조하십시오.

QUERY 옵션을 사용하면 백업 이미지, 로드 사본 이미지 또는 로그를 나열할 수 있습니다. 나열할 로그 범위를 선택할 수 있습니다. 사용 중이 아닌 백업 이미지를 보기 위해 요청할 수도 있습니다.

EXTRACT 옵션을 사용하면 백업 이미지나 로그를 TSM에서 현재 디렉토리로 복사할 수 있습니다. 추출할 백업 이미지나 로그를 선택할 수 있습니다.

DELETE 옵션을 사용하면 백업 이미지를 비활성화하거나 TSM에서 로그를 삭제할 수 있습니다. 처리할 백업 이미지나 로그를 선택할 수 있습니다. KEEP *n* 옵션을 사용하여 최근 *n*개의 백업 이미지를 보존할 수 있습니다. OLDER THAN *timestamp* 또는 *n* DAYS 옵션을 사용하여 DELETE 요청을 조정할 수도 있습니다.

데이터 링크를 사용한 Tivoli Space Manager 통합

DB2 Data Links Manager는 TSM(Tivoli Space Manager)과, 원시 JFS(Journaled File System)의 맨 위 층에 있는 가상 파일 시스템(FSM)을 사용할 수 있습니다. FSM은 JFS와 같은 방법으로 액세스하여 구성할 수 있습니다.

이 기능은 정기적으로 제3의 저장영역으로 이동해야 하는 많은 파일이 있는 파일 시스템을 가지고 있는 사용자에게 유익합니다. 이러한 파일 시스템에 대한 공간은

정기적으로 관리해야 합니다. TSM의 DB2 Data Links Manager 지원은 DATALINK 파일에 대한 공간을 관리할 때 더 많은 융통성을 제공합니다. DB2 Data Links Manager 파일 시스템에서 저장할 수 있는 모든 파일에 대해 충분한 저장영역을 사전에 할당하기 보다는, TSM은 정상적인 사용 중에 갑자기 파일 시스템이 다 차는 위험이 없도록, 일정 시간이 지나서 Data Link에서 파일 시스템의 할당을 조정할 수 있도록 허용합니다.

제한사항

- 이 기능은 현재 AIX에서만 지원됩니다.
- FC(읽기 사용권한 데이터베이스) 링크 파일의 선택적 이주(**dsmmigrate**) 및 취소는 루트 사용자만 수행해야 합니다.

  선택적 이주는 읽기 사용권한 데이터베이스 파일의 경우 DataLink 관리 프로그램 관리자(dlfm)인 파일 소유자만 수행할 수 있습니다. 그러한 파일에 액세스하려면, 호스트 데이터베이스로부터의 토큰이 필요합니다. 토큰을 필요로 하지 않는 유일한 사용자는『루트』사용자입니다.『루트』사용자가 그러한 파일에서 선택적 이주와 취소를 수행하는 것은 쉽습니다. dlfm 사용자는 처음에 유효한 토큰을 사용해야만 FC 파일을 이주할 수 있습니다. 두 번째로 이주를 시도하면(취소 후에),『ANS1028S 내부 프로그램 오류. 서비스 담당자에게 문의하십시오.』라는 오류 메시지와 함께 조작이 실패합니다. 루트가 아닌 사용자가 FC 파일에서 **dsmmigrate**를 수행하려고 하면, 조작은 실패합니다. 이러한 제한사항은 보통 파일 서버에서 파일에 액세스하는 관리자들에 대한 것이므로 부수적인 제한사항입니다.
- **stat** 및 **statfs** 시스템 호출이 dlfs가 fsm에 거쳐 마운트되어도 *Vfs-type*을 `dlfs`가 아닌 `fsm`을 표시합니다.

  이러한 작동은 *Vfs-type*이 `fsm`인지 여부를 판별하기 위해 파일 시스템에서 **statfs**를 수행하는 **dsmrecalld** 디먼의 정상적인 기능을 지원합니다.
- **dsmls** 명령은 최소 *inode* 번호를 가지고 있는 파일이 FC(읽기 사용권한 데이터베이스)에 링크된 경우 출력을 표시하지 않습니다.

  dsmls 명령은 **ls** 명령과 유사하여, TSM에서 관리하는 파일들을 나열합니다. 사용자 조치가 필요없습니다.

부록H. 데이터베이스 복구를 위한 User Exit

User Exit 프로그램을 개발하여 로그 파일 아카이브 및 검색을 자동화할 수 있습니다. (OS/2에서는 백업 및 복원 조작에 User Exit 프로그램을 사용할 수도 있습니다.) 로그 파일 아카이브 또는 검색을 위해 User Exit 프로그램을 호출하기 전에, *userexit* 데이터베이스 구성 매개변수가 YES로 설정되었는지 확인하십시오. 이 경우에도 롤 포워드 복구에 데이터베이스를 사용할 수 있게 됩니다.

User Exit 프로그램이 호출되면, 데이터베이스 관리 프로그램은 제어를 실행 파일인 db2uext2에 전달합니다. (OS/2에서 백업 및 복원 조작은 먼저 db2uexit.cmd를 호출한 다음 db2uext2를 호출합니다.) 데이터베이스 관리 프로그램은 매개변수를 db2uext2에 전달하고, 완료되면 프로그램이 리턴 코드를 다시 데이터베이스 관리 프로그램에 전달합니다. 데이터베이스 관리 프로그램은 리턴 상태의 한정된 세트만 처리하므로 User Exit 프로그램이 오류 상태를 처리할 수 있어야 합니다(494 페이지의 『오류 조절』 참조). 그리고 데이터베이스 관리 프로그램 인스턴스 내에서 하나의 User Exit 프로그램만 호출할 수 있으므로, 수행을 요청할 수 있는 각 조작에 대해 섹션을 가지고 있어야 합니다.

이 절에는 다음 주제를 다룹니다.

- 『User Exit 프로그램 샘플』
- 490 페이지의 『호출 형식』
- 492 페이지의 『백업 및 복원에 대한 고려사항(OS/2용 DB2 전용)』
- 494 페이지의 『오류 조절』

User Exit 프로그램 샘플

샘플 User Exit 프로그램은 지원되는 모든 플랫폼에 대해 제공됩니다. 사용자의 요구사항에 맞도록 이 프로그램을 수정할 수 있습니다. 샘플 프로그램에서는 가장 효율적인 사용을 위해 도움이 될 정보를 주석으로 첨부하고 있습니다.

User Exit 프로그램이 사용 중인 로그 경로에서 아카이브 로그 경로로 로그 파일을 *복사*해야 한다는 점에 유의해야 합니다. 사용 중인 로그 경로에서 로그 파일을 제거하지 마십시오. (데이터베이스를 복구할 때 문제점이 발생할 수 있습니다.) DB2는 더이상 복구에 로그 파일이 필요하지 않을 때 사용 중인 로그 경로에서 아카이브 로그 파일을 제거합니다.

다음은 DB2와 함께 제공되는 샘플 User Exit 프로그램에 대한 설명입니다.

- **UNIX 기반 시스템**

  UNIX 기반 시스템용 DB2에 대한 User Exit 샘플 프로그램은 `sqllib/samples/c` 서브디렉토리에 있습니다. 제공된 샘플이 C로 코딩되어 있는 상태라도, User Exit 프로그램은 다른 프로그램밍 언어로 쓰여질 수 있습니다.

  User Exit 프로그램은 실행 가능한 파일 이름인 `db2uext2`여야 합니다.

  UNIX 기반 시스템에 대해 네 개의 샘플 User Exit 프로그램이 있습니다.

  – `db2uext2.cadsm`

    이 샘플에서는 Tivoli Storage Manager를 사용하여 데이터베이스 로그 파일을 아카이브하고 검색합니다.

  – `db2uext2.ctape`

    이 샘플에서는 테이프 미디어를 사용하여 데이터베이스 로그 파일을 아카이브하고 검색합니다.

  – `db2uext2.cdisk`

    이 샘플에서는 운영 체제 COPY 명령과 디스크 미디어를 사용하여 데이터베이스 로그 파일을 아카이브하고 검색합니다.

  – `db2uxt2.cxbsa`

    이 샘플에서는 Legato** Systems, Inc.의 Legato NetWorker** 버전 4.2.5 프로그램을 사용하여 데이터베이스 로그 파일을 아카이브하고 검색합니다. 이 샘플은 AIX에서만 지원됩니다.

- **Windows 운영 체제**

  Windows 운영 체제용 DB2에 대한 User Exit 샘플 프로그램은 sqllib\samples\c 서브디렉토리에 있습니다. 제공된 샘플이 C로 코딩되어 있는 상태라도, User Exit 프로그램은 다른 프로그래밍 언어로 쓰여질 수 있습니다.

  User Exit 프로그램은 실행 가능한 파일 이름인 db2uext2여야 합니다.

  Windows 운영 체제에 대해 두 개의 샘플 User Exit 프로그램이 있습니다.

  – db2uext2.cadsm

    이 샘플에서는 Tivoli Storage Manager를 사용하여 데이터베이스 로그 파일을 아카이브하고 검색합니다.

  – db2uext2.cdisk

    이 샘플에서는 운영 체제 COPY 명령과 디스크 미디어를 사용하여 데이터베이스 로그 파일을 아카이브하고 검색합니다.

- **OS/2**

  S/2용 DB2에 대한 User Exit 샘플 프로그램은 \sqllib\samples\rexx 디렉토리의 인스턴스 서브디렉토리에 있습니다. (dbuexit.CAD 프로그램은 예외로, 이는 \sqllib\samples\c 디렉토리의 인스턴스 서브디렉토리에 있습니다.) 제공된 샘플이 대부분 REXX 명령 파일인 반면, 사용자의 User Exit 프로그램은 다른 프로그램 언어로 기록될 수 있습니다.

  구현하려는 샘플은 확장자 이름이 .cmd 또는 .exe인 db2uexit로 이름을 바꾸어야 합니다. 이름이 바뀐 파일을 \sqllib\bin 디렉토리로 이동시키십시오.

  다음 다섯 가지의 OS/2 샘플 User Exit 프로그램이 제공됩니다.

  – db2uexit.ex1

    이 샘플에서는 Seagate** Software Inc.의 Sytos Premium** Version 2.2 프로그램을 사용하여 IBM 외부 테이프 장치에서 데이터를 저장하고 검색합니다. Sytos Premium의 Version 2.2 제품만 현재 지원됩니다. (이 제품을 사용하려면 OS/2 FixPak 26이 필요합니다.) 다른 요구사항에 대해서는 샘플 프로그램 목록을 검토하십시오.

  – db2uexit.ex2

이 샘플에서는 Mountain** Corporation의 Filesafe** 프로그램을 사용합니다. 이는 Mountain 테이프 장치에서 데이터를 저장하고 검색하는데 사용됩니다. 고유한 볼륨 레이블이 데이터베이스의 각 백업 사본에 지정되어 있으므로 하나 이상의 데이터베이스의 여러 백업 이미지를 같은 테이프에 저장할 수 있습니다.

– `db2uexit.ex3`

이 샘플에서는 Maynard** Corporation의 MaynStream** 프로그램을 사용합니다. 이는 Maynard 테이프 장치에서 데이터를 저장하고 검색하는데 사용됩니다. MaynStream은 데이터베이스가 백업된 드라이브가 아닌 다른 드라이브로 데이터베이스 복원 조작 경로를 재지정하는 것을 지원하지 않습니다.

– `db2uexit.ex4`

이 샘플은 OS/2 XCOPY 명령을 사용합니다. 이 저장영역은 하드 디스크, 디스켓 또는 광 카트리지와 같이 OS/2에서 지원하는 어떠한 장치도 될 수 있습니다. 이 장치는 워크스테이션이 적절하게 구성된 경우에 LAN 경로 재지정 드라이브가 될 수 있습니다.

XCOPY는 데이터베이스를 백업하고 복원하는데 사용할 수 없습니다.

– `db2uexit.CAD`

C로 작성된 이 샘플은 TSM(Tivoli Storage Manager) 샘플 프로그램과 같습니다. 이 샘플은 데이터베이스 로그 파일을 아카이브하고 검색하는데 사용합니다.

호출 형식

데이터베이스 관리 프로그램이 User Exit 프로그램을 호출할 때, 일련의 매개변수 세트(CHAR 데이터 유형의)를 프로그램에 전달합니다. 호출 형식은 사용자의 운영 체제에 따라 달라집니다.

- UNIX 기반 운영 체제 또는 Windows NT/2000 호출 형식:

```
db2uext2 -OS<os> -RL<db2rel> -RQ<request> -DB<dbname>
-NN<nodenum> -LP<logpath> -LN<logname> -AP<tsmpasswd>
-SP<startpage> -LS<logsize>
```

os 인스턴스가 수행 중인 플랫폼을 지정합니다. 올바른 값은 AIX, Solaris, HP-UX, SCO, Linux, Dynix/ptx, SGI 및 NT입니다.

db2rel DB2 릴리스 레벨을 지정하십시오(예: SQL07020).

request 요청 유형을 지정하십시오. 올바른 값은 ARCHIVE 및 RETRIEVE입니다.

dbname 데이터베이스 이름을 지정하십시오.

nodenum 지역 노드 번호를 지정하십시오(예: 5).

logpath 로그 파일에 완전한 경로를 지정하십시오. 경로에는 반드시 뒤 경로 분리 문자가 있어야 합니다(예: /u/database/log/path/ 또는 d:\logpath\).

logname 아카이브하거나 검색할 로그 파일의 이름을 지정합니다(예: S0000123.LOG).

tsmpasswd TSM 암호를 지정하십시오. (데이터베이스 구성 매개변수 *tsm_password*의 값을 이전에 지정한 경우, 그 값은 User Exit 프로그램에 전달됩니다.)

startpage 로그 extent가 시작하는 장치의 4KB 오프셋 페이지 수를 지정합니다.

logsize 로그 extent 크기를 4KB 페이지 내로 지정하십시오. 이 프로그램은 로깅에 원시 장치가 사용될 경우에만 유효합니다.

- OS/2 호출 형식:

```
action drive db_alias log_path log_file indicator
```

action 올바른 값은 BACKUP, RESTORE, ARCHIVE 및 RETRIEVE입니다.

drive 백업 조작의 경우, 백업하는 데이터베이스가 있는 드라이브를 지정합니다. 복원 조작의 경우, 복원하는 데이터베이스가 있는 드라이브를 지정합니다. 아카이브 또는 검색 조작의 경우, 데이터베이스가 있는 드라이브를 지정합니다. 어느 경우에서든지 드라이브 이름과 콜론을 지정하십시오(예: C:).

db_alias 데이터베이스 별명(또는 별명이 없는 경우에는 데이터베이스 이름)을 지정합니다.

log_path 백업 또는 복원 조작의 경우, 백업하거나 복원할 파일 목록을 포함하는 응답 파일의 완전한 이름을 지정합니다. 목록의 각 이름은 와일드카드 문자를 포함할 수도 있는 완전한 파일 이름입니다. 복원 조작의 경우, 드라이브 이름과 경로는 백업되었을 때의 소스 드라이브와 데이터베이스 파일 경로를 나타냅니다. 예를 들어, 응답 파일에 C:\SQLUTIL\dbname.MH1이 있을 경우, 이는 dbname.MH1 파일이 C:\SQLUTIL로부터 백업되었음을 의미합니다.

아카이브 또는 검색 조작의 경우, 로그 경로 디렉토리를 지정합니다(예: C:\SQL00001\SQLOGDIR\).

log_file 백업 조작의 경우, 백업 유틸리티에 의해 생성된 미디어 레이블을 지정합니다. 이 레이블은 데이터베이스 별명 및 시간소인으로 구성됩니다. 복원 조작의 경우, 파일이 복원될 데이터베이스 서브디렉토리의 경로 이름을 지정합니다. 드라이브 이름은 *drive* 매개변수에 지정되기 때문에 포함되지 않습니다. 형식은 \SQL*nnnnn*\입니다.

아카이브 또는 검색 조작의 경우, 로그 파일 이름을 지정합니다(예: S0000001.LOG).

indicator 백업 또는 복원 조작 동안에 여러 호출을 지원하기 위해 사용할 수 있는 표시기를 지정합니다. 첫 번째 호출의 문자값은 1이고, 후속 호출의 문자값은 2입니다.

User Exit 프로그램은 백업 또는 복원 조작 동안 여러 번 호출됩니다. 첫 번째 호출은 미디어 헤더(.MH*n*) 파일을 백업 또는 복원하고, 두 번째 호출은 전체 데이터베이스 파일 세트를 백업 또는 복원합니다.

이 매개변수는 아카이브 또는 검색 조작에 사용되지 않습니다.

백업 및 복원에 대한 고려사항(OS/2용 DB2 전용)

다음 고려사항은 백업 또는 복원 유틸리티로부터 호출되는 User Exit 프로그램을 작성 중일 경우에 적용됩니다.

- User Exit 프로그램에 의해 리턴되는 0 이외의 리턴 코드는 유틸리티의 실패를 일으키고 재시도되지 않습니다.
- 완전한 파일 이름에는 지원되는 와일드카드 문자만 사용하십시오. 예를 들어, C:\SQL00001*.* 및 C:*.MH*는 승인 가능한 검색 표준입니다.
- User Exit 프로그램은 각각의 행이 캐리지 리턴 및 줄 바꾸기 문자에 의해 종료되는 행 별로 완전한 하나의 파일 이름을 가지는 응답 파일 형식을 처리해야만 합니다. 파일에 파일 끝 문자가 없습니다.
- 같은 데이터베이스의 여러 백업 이미지가 하나의 미디어에 위치될 경우, User Exit 프로그램은 복원 조작 동안 올바른 이미지를 선택할 수 있어야 합니다. (설명은 487 페이지의 『User Exit 프로그램 샘플』에 있는 db2uexit.ex2를 참조하십시오.)
- 하나의 백업 장치를 공유하여 동시에 수행되는 두 개의 백업 조작은 직렬이어야 합니다.
- 백업 이미지가 둘 이상의 미디어에 걸쳐 있게 되면, 미디어에 대한 프롬프트는 그 이미지가 호출하는 User Exit 프로그램이나 응용프로그램에 의해 처리해야 합니다. 이러한 상태를 지원하기 위해 백업 및 복원 유틸리티는 User Exit 프로그램을 호출하는 운영 체제 포그라운드 세션을 시작합니다.
- User Exit 프로그램은 데이터베이스 디렉토리 내의 어떠한 서브디렉토리도 백업해서는 안됩니다.
- User Exit 프로그램을 사용하여 데이터베이스를 복원할 때, 복원 유틸리티는 데이터베이스에 대한 완전한 제어를 필요로 합니다. 그러나 워크스테이션은 복원되는 데이터베이스보다는 다른 데이터베이스의 활동 중인 연결을 할 수 있습니다.
- 데이터베이스가 User Exit 프로그램으로 백업 또는 복원되고 다른 조작이 동일한 테이프 장치를 사용하고 있을 경우, 백업 또는 복원 조작에 실패하여 재시작해야 합니다. 이러한 상황을 피하려면 백업 또는 복원 조작이 진행 중일 때 로깅을 위해 User Exit 프로그램을 호출하는 다른 데이터베이스가 사용되고 있지는 않은지 확인하거나, 아직 장치가 준비되지 않은 경우에는 나중에 User Exit 프로그램이 백업 또는 복원 조작을 재시도하도록 할 수 있습니다.

- 복원 조작의 드라이브 이름 및 경로는 백업 조작 동안에 지정된 것과는 다를 수 있습니다. 예를 들어, dbname.MH1 파일이 C:\SQLUTIL로부터 백업될 경우, 이를 D:\SQLUTIL2로 백업할 수 있습니다.

오류 조절

User Exit 프로그램은 고유하고 의미있는 리턴 코드를 제공하도록 설계해야 합니다. 그러면 데이터베이스 관리 프로그램이 이 리턴 코드를 올바르게 해석할 수 있습니다. User Exit 프로그램은 기반이 되는 운영 체제 명령 프로세서에 의해 호출되므로, 운영 체제 자체적으로 오류 코드를 리턴할 수 있습니다. 그리고 이러한 오류 코드는 다시 맵핑되지 않으므로, 운영 체제 메시지 도움말 유틸리티를 사용하여 이 코드에 대한 정보를 확보하십시오.

OS/2에서는 User Exit 프로그램에서 0이 아닌 리턴 코드를 리턴하면 백업 또는 복원 유틸리티가 실패하여 재시도되지 않습니다. 유틸리티는 일반 SQLCODE -2029를 보고하는데, 이 코드의 메시지 텍스트에는 User Exit 프로그램이나 운영 체제에서 리턴하는 코드가 표시됩니다.

표24에서는 User Exit 프로그램에서 리턴될 수 있는 코드와 이 코드들이 데이터베이스 관리 프로그램에서 해석되는 방법에 대한 설명을 보여줍니다. 만약 리턴 코드가 표에 기록되어 있지 않으면, 값이 32인 것으로 취급됩니다.

표 24. User Exit 프로그램의 리턴 코드. 아카이브 및 검색 조작에만 적용됩니다.

리턴 코드	설명
0	성공적
4	임시 자원 오류를 만났습니다.[a]
8	운영자 개입이 필요합니다.[a]
12	하드웨어 오류[b]
16	User Exit 프로그램에 의한 오류 또는 프로그램에 의해 사용된 소프트웨어 함수에 의한 오류[b]
20	User Exit로 넘겨진 하나 이상의 매개변수에 의한 오류. User Exit 프로그램이 지정된 매개변수를 정확하게 처리했는지 검증하십시오.[b]
24	User Exit 프로그램을 찾을 수 없습니다. OS/2의 경우, 이 오류 메세지는 복원 조작을 완료하기 위해 필요한 파일을 현재 백업 미디어에서 찾을 수 없음을 의미합니다.[b]
28	입/출력(I/O) 실패나 운영 체제에 의해 발생한 오류[b]

표 24. User Exit 프로그램의 리턴 코드 (계속). 아카이브 및 검색 조작에만 적용됩니다.

리턴 코드	설명
32	User Exit 프로그램이 사용자에 의해 종료되었습니다.[b]
255	실행 파일에 대해 라이브러리 파일을 로드할 수 없는 User Exit 프로그램에 의해 발생한 오류[c]

[a]아카이브 및 검색 요청의 경우, 리턴 코드 4 또는 8은 5분 내에 재시도시킵니다. User Exit 프로그램이 같은 로그 파일에 대한 검색 요청에 대해 4 또는 8을 계속 리턴할 경우, DB2는 중단됩니다. (이는 롤 포워드 조작이나 복제 유틸리티에서 사용되는 **sqlurlog** API 호출에 적용됩니다.)

[b]User Exit 요청은 5분 동안 일시중단됩니다. 이 시간 동안에는 오류 조건의 원인이 된 요청을 포함한 모든 요청들이 무시됩니다. 5분 정도 일시중단된 후에 그 다음 요청이 진행됩니다. 이 요청이 오류 없이 진행되면, 새로운 User Exit 요청 처리가 계속되고 DB2는 실패하거나 이전에 일시중단된 아카이브 요청을 다시 발행합니다. 재시도를 하는 동안 8보다 큰 리턴코드가 작성될 경우, 요청은 5분간 더 일시중단됩니다. 5분간의 일시정지는 문제점을 정정하거나 데이터베이스를 중지시키고 재시작할 때까지 계속됩니다. 모든 응용프로그램이 데이터베이스로부터 연결해제되고 나면, DB2는 이전에 제대로 아카이브되지 않았을 수도 있는 로그 파일에 대해 아카이브 요청을 발행합니다. User Exit 프로그램이 로그 파일 아카이브에 실패할 경우, 디스크가 로그 파일로 채워져서 성능이 떨어질 수 있습니다. 한 번 디스크가 완전히 채워지면, 데이터베이스 관리 프로그램은 데이터베이스 갱신을 위한 더이상의 응용프로그램 요청을 받아들이지 않습니다. 로그 파일을 검색하기 위해 User Exit 프로그램이 호출되면, ROLLFORWARD STOP 옵션이 지정되어 있지 않으면 롤 포워드 복구는 중지되지 않고 일시중단됩니다. STOP 옵션이 지정되지 않았으면, 사용자는 문제점을 수정하고 복구를 재개할 수 있습니다.

[c] User Exit 프로그램이 오류 코드 255를 리턴할 경우, 프로그램이 실행 파일에 대해 라이브러리 파일을 로드할 수 없는 경우입니다. 이를 검증하려면 직접 User Exit 프로그램을 호출하십시오. 자세한 정보가 표시됩니다.

주: 아카이브 및 검색 조작 중에 0, 4 및 24를 제외한 모든 리턴 코드에 대해 경고 메시지가 발행됩니다. 경고 메시지는 User Exit 프로그램으로부터 오는 리턴 코드 및 User Exit 프로그램에 제공된 입력 매개변수의 사본을 포함합니다.

부록I. 벤더 제품에 대한 API 백업 및 복원

DB2는 백업 및 복원 조작에 대해 데이터를 저장하고 검색하기 위해 써드 파티 미디어 관리 제품에서 사용할 수 있는 인터페이스를 제공합니다. 이 기능은 DB2의 표준 포트로 지원되는 디스켓, 디스크, 테이프 및 Tivoli Storage Manager의 백업 및 복원 데이터 목표를 증가시키기 위해 설계됩니다.

이 써드 파티 미디어 관리 제품은 이 부록의 나머지에서 벤더 제품으로 언급됩니다.

DB2는 많은 벤더에서 사용할 수 있는 백업 및 복원에 대한 일반 용도의 인터페이스를 제공하는 일련의 함수 프로토타입 세트를 정의합니다. 이 함수들은 UNIX 기반 시스템의 공유 라이브러리나 OS/2 또는 Windows 운영 체제의 DLL에서 벤더에 의해 제공됩니다. DB2에서 함수를 호출할 경우, 호출하는 백업 또는 복원 루틴에 지정된 공유 라이브러리나 DLL이 로드되고 벤더에서 제공하는 함수가 호출되어 필수 타스크를 수행합니다.

이 부록은 네 개의 파트로 나누어 집니다.

- 벤더 제품과의 DB2 상호작용에 대한 조작상의 개요
- DB2의 벤더 API에 대한 자세한 설명
- API 호출에서 사용되는 데이터 구조의 정보
- 벤더 제품을 사용한 백업 및 복원 호출에 대한 세부사항

조작 개요

DB2 및 벤더 제품을 인터페이스하기 위해 다섯 개의 함수가 정의되어 있습니다.

- sqluvint - 초기화 및 장치에 링크
- sqluvget - 장치로부터 데이터 읽기
- sqluvput - 장치에 데이터 기록
- sqluvend - 장치 링크 해제

- sqluvdel - 확약된 세션 삭제

DB2는 이러한 함수를 호출하므로, UNIX 기반 시스템의 공유 라이브러리나 OS/2 또는 Windows 운영 체제의 DLL에서 벤더 제품에 의해 제공해야 합니다.

주: 공유 라이브러리나 DLL 코드는 데이터베이스 엔진 코드의 일부로 수행됩니다. 그러므로 재입력하여 전체적으로 디버그해야 합니다. 잘못된 함수는 데이터베이스의 데이터 무결성을 방해할 수 있습니다.

특정의 백업 또는 복원 조작 동안 DB2가 호출할 일련의 함수들은 다음 사항에 따라 다릅니다.

- 초기화할 세션 수
- 백업 또는 복원 조작의 여부
- 백업 또는 복원 조작에 지정되는 PROMPTING 모드
- 데이터가 저장될 장치의 특성
- 조작 중에 발견할 수 있는 오류

세션 수

DB2는 하나 이상의 데이터 스트림이나 세션을 사용하는 데이터베이스 오브젝트의 백업 및 복원을 지원합니다. 세 개의 세션을 사용하는 백업 또는 복원에는 세 개의 실제 또는 논리 장치를 사용할 수 있어야 합니다. 벤더 장치 지원을 사용 중일 경우, 각 실제 또는 논리 장치에 대한 인터페이스를 관리하는 것은 벤더 함수의 책임입니다. DB2는 단지 벤더에서 제공하는 함수로부터 데이터를 받거나 그 함수로 데이터 버퍼를 보냅니다.

사용되는 세션 수는 백업 또는 복원 데이터베이스 함수를 호출하는 응용프로그램에 의해 매개변수로 지정됩니다. 이 값은 **sqluvint**에서 사용되는 INIT-INPUT 구조에서 제공됩니다(507 페이지의 『sqluvint - 초기화 및 장치에 링크』 참조).

DB2는 지정된 수에 도달할 때까지 세션들을 계속 초기화하거나, **sqluvint** 호출로부터 SQLUV_MAX_LINK_GRANT 경고 리턴 코드를 받습니다. 지원할 수 있는 최대 세션 수에 도달하였음을 DB2에 경고하기 위해 벤더 제품에서는 사용 중

인 세션 수를 추적할 코드를 요구할 것입니다. DB2에 경고하는데 실패하면 DB2 초기화 세션 요청이 실패하여, 모든 세션이 종료하고 전체 백업 또는 복원 조작이 실패하게 됩니다.

조작이 백업일 경우, DB2는 각 세션의 시작 부분에 미디어 헤더 레코드를 씁니다. 레코드에는 DB2가 복원 조작 동안 세션을 식별하기 위해 사용하는 정보가 있습니다. DB2는 백업 이미지의 이름에 순차 번호를 추가하여 각 세션을 식별합니다. 번호는 첫 번째 세션에 대해 1로 시작하여, 백업 또는 복원 조작에 대해 **sqluvint** 호출을 사용하여 다른 세션을 초기화할 때마다 1씩 증가합니다. 자세한 정보는 527 페이지의 『INIT-INPUT』을 참조하십시오.

백업 조작이 제대로 완료되면, DB2는 닫은 마지막 세션에 미디어 바닥글을 씁니다. 이 바닥글에는 백업 조작을 수행하기 위해 사용한 세션 수를 DB2에 알리는 정보가 포함되어 있습니다. 복원 조작 동안, 이 정보는 모든 세션이나 데이터 스트림이 복원되었는지 확인하는 데 사용됩니다.

오류, 경고 또는 프롬프트 없이 조작

백업의 경우, DB2는 *각* 세션마다 다음과 같은 일련의 호출을 발행합니다.

```
sqluvint, action = SQLUV_WRITE
```

1부터 n만큼 다음을 수행합니다.

```
sqluvput
```

다음을 1회 수행합니다.

```
sqluvend, action = SQLUV_COMMIT
```

DB2가 **sqluvend** 호출(조치 SQLUV_COMMIT)을 발행할 경우, DB2는 벤더 제품이 출력 데이터를 적절하게 저장할 것으로 예상합니다. DB2에 리턴되는 SQLUV_OK 리턴 코드는 성공을 나타냅니다.

sqluvint 호출에서 사용되는 DB2-INFO 구조에는 백업을 식별하기 위해 필요한 정보가 있습니다(522 페이지의 『DB2-INFO』 참조). 순차 번호가 제공됩니다. 벤더 제품은 이 정보를 저장할 것을 선택할 수 있습니다. DB2는 복원 동안 이를 사용하여 복원할 백업을 식별합니다.

복원의 경우, 각 세션에 대한 일련의 호출은 다음과 같습니다.

```
sqluvint, action = SQLUV_READ
```

다음을 1부터 n만큼 수행합니다.

```
sqluvget
```

다음을 1회 수행합니다.

```
sqluvend, action = SQLUV_COMMIT
```

sqluvint 호출에서 사용되는 DB2-INFO 구조에 있는 정보에는 백업을 식별하기 위해 필요한 정보가 포함될 것입니다. 순차 번호가 제공되지 않습니다. DB2는 모든 백업 오브젝트(백업 동안 확약된 세션 출력)가 리턴될 것으로 예상합니다. 리턴되는 첫 번째 백업 오브젝트는 순차 번호 1로 생성된 오브젝트이고, 다른 모든 오브젝트는 특정 순서 없이 복원됩니다. DB2는 모든 오브젝트가 처리되었는지 확인하기 위해 미디어 바닥글을 확인합니다.

주: 모든 벤더 제품이 백업 오브젝트의 이름 레코드를 보존하는 것은 아닙니다. 테이프나 제한된 용량의 다른 미디어에 백업될 경우에 거의 그렇습니다. 복원 세션의 초기화 동안, 식별 정보를 초기화하여 필요한 백업 정보가 요구될 때 사용할 수 있도록 계획할 수 있습니다. 이는 백업을 저장하기 위해 주크 박스나 로봇 시스템을 사용할 경우에 가장 유용할 수 있습니다. DB2는 항상 미디어 헤더(각 세션 출력의 첫 번째 레코드)를 확인하여 올바른 데이터가 복원되는지 확인합니다.

PROMPTING 모드

백업 또는 복원 조작이 초기화될 경우, 두 개의 프롬프팅 모드가 가능합니다.

- WITHOUT PROMPTING 또는 NOINTERRUPT: 벤더 제품이 메시지를 사용자에게 제공하거나 사용자가 그 메시지에 응답할 수 있는 기회가 없습니다.
- PROMPTING 또는 INTERRUPT: 사용자가 벤더 제품으로부터 메시지를 받아서 응답할 수 있습니다.

PROMPTING 모드의 경우, 백업 및 복원은 세 가지의 가능한 사용자 조치를 정의합니다.

- 계속

데이터를 읽어서 장치에 쓰는 조작이 재개됩니다.

- 장치 종료

  장치는 추가 데이터를 받지 않으며 세션은 종료합니다.

- 종료

  전체 백업 및 복원 조작이 종료합니다.

PROMPTING 및 WITHOUT PROMPTING 모드 사용에 대해서는 다음 절을 참조하십시오.

장치 특성

벤더 장치 지원 API의 목적을 위해 장치의 두 가지 일반 유형이 정의됩니다.

- 미디어를 변경하려면 사용자 조치가 필요한 제한 용량의 장치(예: 테이프 드라이브, 디스켓 또는 CDROM 드라이브)
- 정상적인 조작에서 사용자가 미디어를 조절하지 않아도 되는 대용량의 장치 (예: 주크 박스 또는 지능형 로봇 미디어 조절 장치)

제한된 용량의 장치를 사용할 경우에는 백업 또는 복원 조작 동안 사용자에게 추가 미디어를 로드하라는 프롬프트를 표시해야 할 수도 있습니다. 일반적으로 DB2는 백업 또는 복원 조작에 대해 미디어가 로드되는 순서에 민감하지 않습니다. DB2는 또한 벤더 미디어 조절 메시지를 사용자에게 전달할 수 있는 기능도 제공합니다. 이 프롬프팅에서는 백업 또는 복원 조작이 PROMPTING을 설정한 상태에서 초기화하도록 요구합니다. 미디어 조절 메시지 텍스트는 리턴 코드 구조의 설명 필드에 지정됩니다.

PROMPTING을 설정하고, DB2가 SQLUV_ENDOFMEDIA 또는 SQLUV_ENDOFMEDIA_NO_DATA 리턴 코드를 **sqluvput**(쓰기) 또는 **sqluvget**(읽기) 호출로부터 받으면, DB2는 다음을 수행합니다.

- 호출이 **sqluvput**인 경우, 세션에 보내진 마지막 버퍼를 다시 보내도록 표시합니다. 이는 나중에 세션에 놓입니다.
- **sqluvend**(조치 = SQLUV_COMMIT)를 사용하여 세션을 호출합니다. 성공할 경우(SQLUV_OK return code), DB2는 다음을 수행합니다.

  - 미디어 끝 상태 신호를 보낸 리턴 코드 구조에서 벤더 미디어 조절 메시지를 사용자에게 보냅니다.
  - continue, device terminate 또는 terminate 응답에 대해 사용자에게 프롬프트를 표시합니다.

- 응답이 *continue*이면, DB2는 **sqluvint** 호출을 사용하여 다른 세션을 초기화하고, 성공하면 세션에 데이터를 쓰거나 세션에서 데이터를 읽기 시작합니다. 쓸 때 세션을 고유하게 식별하기 위해, DB2는 순차 번호를 증가합니다. 순차 번호는 **sqluvint**에서 사용되는 DB2-INFO 구조에서 사용 가능하며, 세션에 보내진 첫 번째 데이터 레코드인 미디어 헤더 레코드에 있습니다.

  DB2는 백업 또는 복원 조작을 시작할 때 요청한 것이나, **sqluvint** 호출의 SQLUV_MAX_LINK_GRANT 경고에서 벤더 제품에 의해 표시된 것보다 더 많은 세션을 시작하지는 않습니다.
- 응답이 *device terminate*일 경우, DB2는 다른 세션을 초기화하려고 하지 않으며 사용 중인 세션 수도 1씩 감소합니다. DB2는 모든 세션이 device terminate 응답에 의해 종료되도록 허용하지 않습니다. 백업 또는 복원 조작이 완료될 때까지 최소한 하나의 세션이 사용 중 상태로 보존되어야 합니다.
- 응답이 *terminate*일 경우, DB2는 백업 또는 복원 조작을 종료합니다. DB2가 정확하게 세션을 종료하는 방법에 대한 자세한 정보는 503 페이지의 『DB2에 오류 상태가 리턴될 경우』를 참조하십시오.

백업 또는 복원 성능은 종종 사용하는 장치 수에 좌우되기 때문에, 병렬 처리 방식으로 유지보수하는 것이 중요합니다. 백업 조작의 경우, 사용자는 사용 중인 나머지 세션이 계속 쓰여지는 데이터를 보유할 것을 알지 못할 경우 `continue`로 응답하려고 합니다. 복원 조작의 경우에도 사용자는 모든 미디어가 처리될 때까지 `continue`로 응답하려고 합니다.

백업 또는 복원 모드가 WITHOUT PROMPTING이고, DB2가 SQLUV_ENDOFMEDIA 또는 SQLUV_ENDOFMEDIA_NO_DATA 리턴 코드를 세션으로부터 받을 경우, DB2는 세션을 종료하고 다른 세션은 열려고 하지 않습니다. 백업 또는 복원 조작이 완료되기 전에 모든 세션이 DB2에 미디어 끝 상태를 리턴할 경우, 조작은 실패합니다. 이로 인해, WITHOUT PROMPTING은 용량이 제한되는 장치에서 주의하여 사용해야 합니다. 그러나 용량이 아주 큰 장치에 대해서는 이 모드에서 작동되도록 합니다.

벤더 제품에 대해서는 DB2로부터 미디어 마운팅 및 전환 조치를 숨길 수 있으므로, 장치는 무한한 용량을 가지고 있는 것으로 표시됩니다. 일부 대용량 장치가 이 모드에서 작동합니다. 이러한 경우, 백업한 데이터가 복원 조작이 진행될 때와 같은 순서로 DB2에 리턴되는 것은 중요합니다. 그렇게 되지 않을 경우 데이터가 누락될 수 있지만, DB2는 누락된 데이터를 검출할 수 있는 방법이 없으므로 복원 조작이 성공한 것으로 가정합니다.

DB2는 각 버퍼가 단 하나의 미디어(예: 테이프)에 포함될 것으로 가정하고 벤더 제품에 데이터를 씁니다. 벤더 제품에 대해서는 DB2가 인식하지 못하는 상태에서 여러 미디어에 버퍼를 분할할 수 있습니다. 이러한 경우, 다중 미디어에서 DB2로 재구성된 버퍼를 리턴하는 것은 벤더 제품의 책임이므로 복원 조작 동안 미디어가 처리되는 순서는 중요합니다. 이렇게 할 수 없으면 복원 조작이 실패합니다.

DB2에 오류 상태가 리턴될 경우

백업 또는 복원 조작을 수행할 때, DB2는 모든 세션이 성공적으로 완료할 것으로 예상합니다. 그렇지 않으면 전체 백업 또는 복원 조작이 실패합니다. 세션은 **sqluvend** 호출, 조치 = SQLUV_COMMIT에서 SQLUV_OK 리턴 코드를 사용하여 DB2에 성공적인 완료 신호를 보냅니다.

복구할 수 없는 오류가 발견되면, 세션은 DB2에 의해 종료됩니다. 이는 DB2 오류나 벤더 제품에서 DB2로 리턴된 오류가 될 수 있습니다. 모든 세션은 백업 또는 복원 조작이 완전하도록 제대로 확약해야 하므로, 실패하면 해당 조작과 연관되는 다른 세션이 종료하게 됩니다.

벤더 제품은 복구할 수 없는 리턴 코드를 사용하여 DB2의 호출에 응답할 경우, 벤더 제품은 RETURN-CODE 구조의 설명 필드에 위치된 메시지 텍스트를 사용하여 선택적으로 추가 정보를 제공할 수 있습니다. 이 메시지 텍스트는 정정 조치를 취할 수 있도록 DB2 정보와 함께 사용자에게 표시됩니다.

세션이 제대로 확약된 백업 시나리오가 있으며, 백업 조작과 연관되는 다른 세션에서는 복구할 수 없는 오류가 발생합니다. 모든 세션은 백업 조작이 성공한 것으로 간주되기 전에 제대로 완료해야 하므로, DB2는 확약된 세션에서 출력 데이터를 삭제해야 합니다. DB2는 **sqluvdel** 호출을 발행하여 오브젝트 삭제를 요청합

니다. 이 호출은 입출력 세션으로 간주되지 않으며, 백업 오브젝트를 삭제하기 위해 필요할 수 있는 연결을 초기화하고 종료하는 책임을 가지고 있습니다.

DB2-INFO 구조는 순차 번호를 포함하지 않습니다. **sqluvdel**은 DB2-INFO 구조에서 나머지 매개변수와 일치하는 모든 백업 오브젝트를 삭제합니다.

경고 조건

DB2는 벤더 제품으로부터 경고 리턴 코드를 받을 수 있습니다. 예를 들어, 장치가 준비되어 있지 않거나 일부 다른 정정 가능한 상태가 발생하였습니다. 이는 읽기 및 쓰기 조작 둘다에 해당됩니다.

sqluvput 및 **sqluvget** 호출에서 벤더는 리턴 코드를 SQLUV_WARNING으로 설정하고, RETURN-CODE 구조의 설명 필드에 위치된 메시지 텍스트를 사용하여 선택적으로 추가 정보를 제공할 수 있습니다. 이 메시지 텍스트는 정정 조치를 취할 수 있도록 사용자에게 표시됩니다. 사용자는 continue, device terminate 또는 terminate 중 하나로 응답할 수 있습니다.

- 응답이 *continue*일 경우, DB2는 백업 조작 동안 **sqluvput**을 사용하여 버퍼를 다시 쓰려고 합니다. 복원 조작 동안, DB2는 **sqluvget** 호출을 발행하여 다음 버퍼를 읽습니다.
- 응답이 *device terminate* 또는 *terminate*일 경우, DB2는 복구할 수 없는 오류가 발생한 후에 응답하는 것과 같은 방법으로 전체 백업 또는 복원 조작을 종료합니다. (예를 들어, 사용 중인 세션을 종료하고 확약된 세션을 삭제합니다.)

조작 힌트 및 추가 정보

이 절에서는 벤더 제품 빌드에 대한 일부 힌트 및 추가 정보를 제공합니다.

복구 실행기록 파일

데이터베이스 복구 조작에서처럼 복구 실행기록 파일을 사용하여 도움을 받을 수 있습니다. 이 파일은 각 데이터베이스와 연관되며, 자동으로 각 백업 또는 복원 조작으로 갱신됩니다. 복구 실행기록 파일에 대한 자세한 정보는 57 페이지의 『복구 실행기록 파일 이해』를 참조하십시오. 파일의 정보는 다음 기능을 통해 보거나, 갱신하거나 제거할 수 있습니다.

- 제어 센터
- 명령행 처리기(CLP)
  - LIST HISTORY 명령
  - UPDATE HISTORY FILE 명령
  - PRUNE HISTORY 명령
- API
  - db2HistoryOpenScan
  - db2HistoryGetEntry
  - db2HistoryCloseScan
  - db2HistoryUpdate
  - db2Prune

파일의 레이아웃에 대한 자세한 정보는 413 페이지의 『데이터 구조: db2HistData』를 참조하십시오.

백업 조작이 완료되면, 하나 이상의 레코드가 파일에 쓰여집니다. 백업 조작의 출력이 벤더 장치에 보내진 경우, 실행기록 레코드의 DEVICE 필드에는 0이 있고 LOCATION 필드에는 다음 중 하나가 있습니다.

- 백업 조작이 호출된 경우에 지정된 벤더 파일 이름
- 벤더 파일 이름을 지정하지 않은 경우, 공유 라이브러리의 이름

옵션 지정에 대한 자세한 정보는 532 페이지의 『벤더 제품을 사용한 백업 또는 복원 조작 호출』을 참조하십시오.

LOCATION 필드는 제어 센터, CLP 또는 API를 사용하여 갱신할 수 있습니다. 백업 이미지를 보유하기 위해 용량이 제한되는 장치(예: 제거 가능한 미디어)를 사용한 경우에는 백업 정보 위치를 갱신할 수 있으며, 미디어는 실제로 다른(아마도 오프 사이트) 저장 위치로 이동됩니다. 지금이 그러한 경우라면, 복구 조작이 필요할 경우에 복구 실행기록 파일을 사용하여 백업 이미지를 찾는데 도움을 받을 수 있습니다.

함수 및 데이터 구조

다음 절에서는 벤더 제품에서 사용할 수 있는 일반 함수와 데이터 구조에 대해 설명합니다.

벤더 제품에 대한 API는 다음과 같습니다.

- 507 페이지의『sqluvint - 초기화 및 장치에 링크』
- 511 페이지의『sqluvget - 장치로부터 데이터 읽기』
- 514 페이지의『sqluvput - 장치에 데이터 기록』
- 517 페이지의『sqluvend - 장치 링크 해제 및 자원 해제』
- 520 페이지의『sqluvdel - 확약된 세션 삭제』

벤더 API가 사용하는 데이터 구조는 다음과 같습니다.

522 페이지의『DB2-INFO』
벤더 장치에 대해 DB2를 식별하는 정보가 있습니다.

525 페이지의『VENDOR-INFO』
벤더 및 장치 버전을 식별하는 정보가 있습니다.

527 페이지의『INIT-INPUT』
DB2 및 벤더 장치 사이의 논리 링크를 설정합니다.

529 페이지의『INIT-OUTPUT』
장치로부터의 출력이 있습니다.

530 페이지의『DATA』
DB2와 벤더 장치 사이에 전송된 데이터가 있습니다.

531 페이지의『RETURN-CODE』
리턴 코드와 오류 설명이 있습니다.

sqluvint - 초기화 및 장치에 링크

이 함수는 DB2와 벤더 장치 사이의 논리 링크 초기화 및 설정에 대한 정보를 제공하기 위해 호출됩니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *dbadm*

필수 연결

데이터베이스

API 포함 파일

sql.h

C API 구문

```
/* File: sqluvend.h */
/* API: Initialize and Link to Device */
/* ... */
int sqluvint (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API 매개변수

Init_input

입력. 벤더 장치와의 논리 링크를 설정하기 위해 DB2에서 제공하는 정보를 포함하는 구조

Init_output

출력. 벤더 장치에서 리턴되는 출력을 포함하는 구조

Return_code

출력. DB2에 전달될 리턴 코드와 간단한 텍스트 설명을 포함하는 구조

사용법 주

각 미디어 입출력(I/O) 세션마다 DB2는 이 함수를 호출하여 장치 핸들을 확보합니다. 어떤 이유로 인해 벤더 함수가 초기화 중에 오류를 발견할 경우, 리턴 코드를 통해 이를 표시합니다. 리턴 코드가 오류를 표시할 경우, DB2는 **sqluvend** 함수를 호출하여 조작을 종료할 것을 선택할 수 있습니다. 가능한 리턴 코드에 대한 세부사항과 각각의 리턴 코드에 대한 DB2 반응이 리턴 코드 테이블에 포함됩니다(509 페이지의 표25 참조).

INIT-INPUT 구조에는 백업 또는 복원을 진행할 수 있는지 판별하기 위해 벤더 제품에서 사용할 수 있는 요소들이 있습니다.

- size_HI_order 및 size_LOW_order

  이것은 백업의 예상 크기입니다. 벤더 장치가 백업 이미지의 크기를 처리할 수 있는지 판별하기 위해 사용할 수 있습니다. 백업을 보유할 경우에 필요할 제거 가능 미디어 수량을 측정할 경우에도 사용할 수 있습니다. 이는 문제점이 있을 경우에 첫 번째 **sqluvint** 호출에서의 실패에 유익할 수 있습니다.

- req_sessions

  계산된 크기와 프롬프팅 레벨과 함께 사용자 요청 세션 수를 사용하여 백업 또는 복원 조작이 가능한지 판별할 수 있습니다.

- prompt_lvl

  프롬프팅 레벨은 제거 가능 미디어 변경(예를 들어, 다른 테이프를 테이프 드라이브에 놓음)과 같은 조치에 대해 프롬프트가 가능한지 벤더에 표시합니다. 사용자에게 프롬프트를 표시할 방법이 없어서 조작을 진행할 수 없음을 표시할 수도 있습니다.

  프롬프팅 레벨이 WITHOUT PROMPTING이고 제거 가능 미디어의 수량이 요청한 세션 수보다 많으면, DB2는 조작을 제대로 완료할 수 없습니다. (500 페이지의 『PROMPTING 모드』 및 501 페이지의 『장치 특성』에서 자세한 정보를 참조하십시오.)

DB2는 DB2-INFO 구조의 필드들을 통해 읽혀질 복원이나 씌여질 백업의 이름을 지정합니다. 조치가 SQLUV_READ일 경우, 벤더 제품은 이름이 지정된 오브젝트가 있는지 확인해야 합니다. 이를 찾을 수 없으면, DB2가 적절한 조치를 취하도록 리턴 코드를 SQLUV_OBJ_NOT_FOUND로 설정해야 합니다.

초기화가 제대로 완료되고 나면, DB2는 다른 데이터 전송 함수를 발행하여 계속하지만 언제든지 **sqluvend** 호출로 세션을 종료할 수 있습니다.

리턴 코드

표 25. sqluvint 및 DB2 조치 결과에 대해 유효한 리턴 코드

헤더 파일 내의 리터럴	설명	유망한 다음 호출	기타 주석
SQLUV_OK	조작 성공	sqluvput, sqluvget (주석 참조)	조치가 SQLUV_WRITE이면, 다음 호출은 sqluvput(데이터를 백업하기 위한)입니다. 조치가 SQLUV_READ이면, SQLUV_OK를 리턴하기 전에 이름이 지정된 오브젝트의 존재를 검증하십시오. 다음 호출은 데이터를 복원하기 위한 sqluvget이 됩니다.
SQLUV_LINK_EXIST	이전에 활성화된 세션	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_COMM_ERROR	장치로 인한 통신 오류	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_INV_VERSION	DB2와 벤더 제품이 호환되지 않습니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_INV_ACTION	올바르지 않은 조치가 요청되었습니다. 매개변수 조합으로 가능하지 않은 조작이 야기되었음을 표시하기 위해 사용되었을 수 있습니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_NO_DEV_AVAIL	현재 사용 가능한 장치가 없습니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_OBJ_NOT_FOUND	지정된 오브젝트를 찾을 수 없습니다. 이는 sqluvint 호출에서의 조치가 'R'(읽기)이고 DB2-INFO 구조에 지정된 기준에 따라 요청한 오브젝트 이름을 찾을 수 없는 경우에 사용해야 합니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.

표 25. sqluvint 및 DB2 조치 결과에 대해 유효한 리턴 코드 (계속)

헤더 파일 내의 리터럴	설명	유망한 다음 호출	기타 주석
SQLUV_OBJS_FOUND	두 개 이상의 오브젝트가 지정된 기준과 일치하지 않습니다. 이는 sqluvint 호출에서의 조치가 'R'(읽기)이고, 두 개 이상의 오브젝트가 DB2-INFO 구조에 지정된 기준과 일치할 경우에 발생합니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_INV_USERID	올바르지 않은 사용자 ID가 지정되었습니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_INV_ PASSWORD	올바르지 않은 암호가 제공되었습니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_INV_OPTIONS	벤더 옵션 필드에서 유효하지 않은 옵션이 발견되었습니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_INIT_FAILED	초기화에 실패하여 세션이 종료됩니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_DEV_ERROR	장치 오류	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_MAX_LINK_ GRANT	설정된 링크의 최대 수	sqluvput, sqluvget (주석 참조)	이는 DB2에 의한 경고로 처리됩니다. 경고는 DB2가 지원할 수 있는 최대 세션 수에 도달했기 때문에 벤더 제품에서 추가 세션을 열지 않도록 지시합니다. (주: 이는 장치 가용성 때문입니다.) 조치가 SQLUV_WRITE (BACKUP)이면, 다음 호출은 sqluvput이 됩니다. 조치가 SQLUV_READ이면, SQLUV_MAX_LINK_GRANT를 리턴하기 전에 이름이 지정된 오브젝트의 존재를 검증하십시오. 다음 호출은 데이터를 복원하기 위한 sqluvget이 됩니다.
SQLUV_IO_ERROR	입출력 오류	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.
SQLUV_NOT_ ENOUGH_SPACE	전체 백업 이미지를 저장하기에는 공간이 충분하지 않습니다. *크기* 계산은 바이트 단위의 64비트 값으로 제공됩니다.	추가 호출 없음	세션 초기화 실패. 해당 세션에 할당된 메모리를 해제하고 종료합니다. 세션이 전혀 설정되지 않았으므로 sqluvend 호출이 수신되지 않습니다.

sqluvget - 장치로부터 데이터 읽기

초기화 이후에 장치에서 데이터를 읽기 위해 이 함수를 호출할 수 있습니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *dbadm*

필수 연결

데이터베이스

API 포함 파일

sqluvend.h

C API 구문

```
/* File: sqluvend.h */
/* API: Reading Data from Device */
/* ... */
int sqluvget (
  void * pVendorCB,
  struct Data         *,
  struct Return_code  *);
/* ... */
typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

API 매개변수

pVendorCB

입력. DATA 구조(데이터 버퍼를 포함하여) 및 Return_code에 대해 할당된 공간에 대한 포인터

Data 입출력. *data* 구조에 대한 포인터

Return_code

출력. API 호출로부터의 리턴 코드

obj_num

검색해야 하는 백업 오브젝트를 지정합니다.

buff_size

사용할 버퍼 *크기*를 지정합니다.

actual_buff_size

읽기 또는 쓰기된 실제 바이트를 지정합니다. 이 값은 출력에 실제로 읽혀진 데이터 바이트 수가 표시되도록 설정해야 합니다.

dataptr

데이터 버퍼에 대한 포인터

reserve

나중에 사용하도록 예약되어 있습니다.

사용법 주

이 함수는 복원 유틸리티에서 사용됩니다.

리턴 코드

표 26. sqluvget 및 DB2 조치 결과에 대해 유효한 리턴 코드

헤더 파일 내의 리터럴	설명	유망한 다음 호출	기타 주석
SQLUV_OK	조작 성공	sqluvget	DB2는 데이터를 처리합니다.
SQLUV_COMM_ERROR	장치로 인한 통신 오류	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_INV_ACTION	올바르지 않은 조치가 요청되었습니다.	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_INV_DEV_HANDLE	올바르지 않은 장치 핸들	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_INV_BUFF_SIZE	올바르지 않은 버퍼 크기가 지정되었습니다.	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_DEV_ERROR	장치 오류	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_WARNING	경고. 이는 DB2에 대한 미디어 끝을 나타내기 위해서는 사용할 수 없습니다. 이러한 목적이 있을 경우에는 SQLUV_ENDOFMEDIA 또는 SQLUV_ENDOFMEDIA_NO_DATA를 사용하십시오. 그러나 이 리턴 코드를 사용하여 장치 준비되지 않음 상태를 표시할 수 있습니다.	sqluvget 또는 sqluvend, action = SQLU_ABORT	504 페이지의 『경고 조건』에서 경고에 대한 DB2의 조절 설명을 참조하십시오.
SQLUV_LINK_NOT_EXIST	현재 링크가 없습니다.	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_MORE_DATA	조작 성공. 추가 데이터를 사용할 수 있습니다.	sqluvget	
SQLUV_ENDOFMEDIA_NO_DATA	미디어 끝 및 0바이트 읽기 (예: 테이프의 끝).	sqluvend	500 페이지의 『PROMPTING 모드』 및 501 페이지의 『장치 특성』에서 미디어 끝 상태에 대한 DB2의 조절 설명을 참조하십시오.
SQLUV_ENDOFMEDIA	미디어 끝 및 0바이트 초과 읽기 (예: 테이프의 끝).	sqluvend	DB2는 500 페이지의 『PROMPTING 모드』 및 501 페이지의 『장치 특성』에 설명된 대로 데이터를 처리한 다음 미디어 끝 상태를 조절합니다.
SQLUV_IO_ERROR	입출력 오류	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
다음 호출: [a] 다음 호출이 sqluvend이고 조치가 SQLU_ABORT일 경우, 이 세션과 다른 모든 사용 중인 세션이 종료됩니다.			

sqluvput - 장치에 데이터 기록

초기화 이후에 장치에 데이터를 쓰기 위해 이 함수를 사용할 수 있습니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *dbadm*

필수 연결

데이터베이스

API 포함 파일

sqluvend.h

C API 구문

```
/* File: sqluvend.h */
/* API: Writing Data to Device */
/* ... */
int sqluvput (
  void * pVendorCB,
  struct Data  *,
  struct Return_code  *);
/* ... */
typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

API 매개변수

pVendorCB

입력. DATA 구조(데이터 버퍼를 포함하여) 및 Return_code에 대해 할당된 공간에 대한 포인터

Data 출력. 쓰려는 데이터로 채워지는 데이터 버퍼

Return_code

출력. API 호출로부터의 리턴 코드

obj_num

검색해야 하는 백업 오브젝트를 지정합니다.

buff_size

사용할 버퍼 크기를 지정합니다.

actual_buff_size

읽기 또는 쓰기된 실제 바이트를 지정합니다. 이 값은 출력에 실제로 읽혀진 데이터 바이트 수가 표시되도록 설정해야 합니다.

dataptr

데이터 버퍼에 대한 포인터

reserve

차후 사용을 위해 예약됩니다.

사용법 주

이 함수는 복원 유틸리티에서 사용됩니다.

리턴 코드

표 27. sqluvput 및 DB2 조치 결과에 대해 유효한 리턴 코드

헤더 파일 내의 리터럴	설명	유망한 다음 호출	기타 주석
SQLUV_OK	조작 성공	완료할 경우(예를 들어, DB2에 데이터가 더이상 없을 경우), sqluvput 또는 sqluvend	성공적인 조작의 다른 프로세스를 알립니다.
SQLUV_COMM_ERROR	장치로 인한 통신 오류	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_INV_ACTION	올바르지 않은 조치가 요청되었습니다.	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_INV_DEV_HANDLE	올바르지 않은 장치 핸들	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_INV_BUFF_SIZE	올바르지 않은 버퍼 크기가 지정되었습니다.	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_ENDOFMEDIA	미디어의 끝(예: 테이프의 끝)에 도달했습니다.	sqluvend	500 페이지의 『PROMPTING 모드』 및 501 페이지의 『장치 특성』에서 미디어 끝 상태에 대한 DB2의 조절 설명을 참조하십시오.
SQLUV_DATA_RESEND	장치에서 버퍼를 다시 보낼 것을 요청하였습니다.	sqluvput	DB2는 마지막 버퍼를 재전송합니다. 이는 한 번만 수행됩니다.
SQLUV_DEV_ERROR	장치 오류	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_WARNING	경고. 이는 DB2에 대한 미디어 끝을 나타내기 위해서는 사용할 수 없습니다. 이러한 목적이 있을 경우에는 SQLUV_ENDOFMEDIA를 사용하십시오. 그러나 이 리턴 코드를 사용하여 장치 준비되지 않음 상태를 표시할 수 있습니다.	sqluvput	504 페이지의 『경고 조건』에서 경고에 대한 DB2의 조절 설명을 참조하십시오.
SQLUV_LINK_NOT_EXIST	현재 링크가 없습니다.	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.
SQLUV_IO_ERROR	입출력 오류	sqluvend, action = SQLU_ABORT[a]	세션이 종료됩니다.

다음 호출:

[a] 다음 호출이 sqluvend이고 조치가 SQLU_ABORT일 경우, 이 세션과 다른 모든 사용 중인 세션이 종료됩니다. 확약된 세션은 일련의 sqluvint, sqluvdel, sqluvend 호출 순서를 사용하여 삭제됩니다(503 페이지의 『DB2에 오류 상태가 리턴될 경우』 참조).

sqluvend - 장치 링크 해제 및 자원 해제

장치를 종료하거나 링크 해제한 다음 관련된 모든 자원을 해제합니다. 벤더는 DB2로 돌아가기 전에 사용하지 않는 자원(예: 할당된 공간과 파일 핸들)을 해제해야 합니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *dbadm*

필수 연결

데이터베이스

API 포함 파일

sql.h

C API 구문

```
/* File: sqluvend.h */
/* API: Unlink the Device and Release its Resources */
/* ... */
int sqluvend (
  sqlint32 action,
  void * pVendorCB,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API 매개변수

action 입력. 세션을 확약하거나 취소할 경우에 사용됩니다.

- SQLUV_COMMIT(0 = 확약)
- SQLUV_ABORT(1 = 취소)

pVendorCB

입력. Init_output 구조에 대한 포인터

Init_output

출력. 할당해제된 Init_output에 대한 공간. 조치가 확약하는 것일 경우 데이터는 백업을 위해 안전한 저장영역으로 확약됩니다. 조치가 취소하는 것일 경우에는 데이터는 백업을 위해 제거됩니다.

Return code

출력. API 호출로부터의 리턴 코드

사용법 주

이 함수는 열린 각 세션마다 호출됩니다. 두 가지 가능한 조치 코드가 있습니다.

- 확약

  이 세션에 대한 데이터 출력이나 세션으로부터의 데이터 읽기가 완료되었습니다.

  쓰기(백업) 세션의 경우, 벤더가 리턴 코드 SQLUV_OK와 함께 DB2에 리턴하면, DB2는 출력 데이터가 벤더 제품에 의해 적절하게 저장되어 나중에 **sqluvint** 호출에서 참조될 경우에 액세스할 수 있는 것으로 가정합니다.

  읽기(복원) 세션의 경우, 벤더가 리턴 코드 SQLUV_OK와 함께 DB2에 리턴하면, 데이터가 다시 필요할 수도 있으므로 데이터를 삭제해서는 안됩니다.

  벤더가 SQLUV_COMMIT_FAILED를 리턴할 경우, DB2는 전체 백업 또는 복원 조작에 문제점이 있는 것으로 가정합니다. 사용 중인 모든 세션은 조치가 SQLUV_ABORT인 **sqluvend** 호출에 의해 종료됩니다. 백업 조작의 경우, 확약된 세션은 일련의 **sqluvint**, **sqluvdel** 및 **sqluvend** 호출 순서를 받습니다 (503 페이지의 『DB2에 오류 상태가 리턴될 경우』 참조).

- 취소

  DB2에서 문제점이 발견되었으며, 더이상 세션에 대한 데이터 읽기 또는 쓰기가 수행되지 않습니다.

  쓰기(백업) 세션의 경우, 벤더는 부분 출력 데이터 세트를 삭제하고 그 부분 출력이 삭제되면 SQLUV_OK 리턴 코드를 사용해야 합니다. DB2는 전체 백업에 문제점이 있는 것으로 가정합니다. 사용 중인 모든 세션은 조치가

SQLUV_ABORT인 **sqluvend** 호출에 의해 종료되며, 확약된 세션은 일련의 **sqluvint**, **sqluvdel** 및 **sqluvend** 호출 순서를 받습니다(503 페이지의『DB2에 오류 상태가 리턴될 경우』참조).

읽기(복원) 세션의 경우, 벤더는 데이터를 삭제하지 말아야 하지만(다시 필요할 수 있으므로) 정리하여 리턴 코드 SQLUV_OK와 함께 DB2에 리턴해야 합니다. DB2는 조치가 SQLUV_ABORT인 **sqluvend** 호출에 의해 모든 복원 세션을 종료합니다. 벤더가 SQLUV_ABORT_FAILED를 DB2에 리턴할 경우, 호출자에게는 이 오류가 통지되지 않습니다. DB2가 첫 번째 심각한 실패를 리턴하고 그 뒤의 실패는 무시하기 때문입니다. 이 경우, 조치가 SQLUV_ABORT인 **sqluvend**를 DB2가 호출하도록 하려면 초기 심각한 오류가 발생해야 합니다.

리턴 코드

표 28. sqluvend 및 DB2 조치 결과에 대해 유효한 리턴 코드

헤더 파일 내의 리터럴	설명	유망한 다음 호출	기타 주석
SQLUV_OK	조작 성공	추가 호출 없음	해당 세션에 할당된 모든 메모리를 해제하고 종료합니다.
SQLUV_COMMIT_FAILED	확약 요청 실패	추가 호출 없음	해당 세션에 할당된 모든 메모리를 해제하고 종료합니다.
SQLUV_ABORT_FAILED	취소 요청 실패	추가 호출 없음	

sqluvdel - 확약된 세션 삭제

확약된 세션을 삭제합니다.

권한 부여

다음 중 하나를 수행하십시오.

- *sysadm*
- *dbadm*

필수 연결

데이터베이스

API 포함 파일

sqluvend.h

C API 구문

```
/* File: sqluvend.h */
/* API: Delete Committed Session */
/* ... */
int sqluvdel (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API 매개변수

Init_input

입력. Init_input 및 Return_code에 대해 할당된 공간

Return_code

출력. API 호출로부터의 리턴 코드. Init_input 구조에 의해 지시된 오브젝트가 삭제됩니다.

사용법 주

여러 세션이 열리고, 일부 세션은 확약되었지만 그 중 하나가 실패할 경우, 이 함수는 확약된 세션을 삭제하기 위해 호출됩니다. 순차 번호를 지정하지 않습니다. **sqluvdel**은 특정의 백업 조작 동안 작성된 모든 오브젝트를 찾아서 삭제하는 책임을 가지고 있습니다. INIT-INPUT 구조의 정보는 삭제할 출력 데이터를 식별하기 위해 사용됩니다. **sqluvdel**에 대한 호출은 벤더 장치에서 백업 오브젝트를 삭제하기 위해 필요한 세션이나 연결을 설정하는 책임을 가지고 있습니다. 이 호출의 리턴 코드가 SQLUV_DELETE_FAILED일 경우, DB2는 첫 번째 심각한 실패를 리턴하고 그 뒤의 실패는 무시하므로 호출자에게 통지하지 않습니다. 이 경우, DB2가 **sqluvdel**를 호출하도록 하려면 초기 심각한 오류가 발생해야 합니다.

리턴 코드

표 29. sqluvdel 및 DB2 조치 결과에 대해 유효한 리턴 코드

헤더 파일 내의 리터럴	설명	유망한 다음 호출	기타 주석
SQLUV_OK	조작 성공	추가 호출 없음	
SQLUV_DELETE_FAILED	삭제 요청 실패	추가 호출 없음	

DB2-INFO

이 구조에는 벤더 장치에 대해 DB2를 식별하는 정보가 있습니다.

표 30. DB2-INFO 구조 내의 필드. 모든 필드는 NULL 종료되는 문자열입니다.

필드 이름	데이터 유형	설명
DB2_id	char	DB2 제품 식별자. 지시하는 문자열의 최대 길이는 8자입니다.
version	char	DB2 제품의 현재 버전. 지시하는 문자열의 최대 길이는 8자입니다.
release	char	DB2 제품의 현재 릴리스. 중요하지 않으면 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
level	char	DB2 제품의 현재 레벨. 중요하지 않으면 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
action	char	취할 조치 지정. 지시하는 문자열의 최대 길이는 1자입니다.
filename	char	백업 이미지를 식별하기 위해 사용되는 파일 이름. NULL일 경우, *server_id*, *db2instance*, *dbname* 및 *timestamp*는 백업 이미지를 고유하게 식별합니다. 지시하는 문자열의 최대 길이는 255자입니다.
server_id	char	데이터베이스가 상주하는 서버를 식별하는 고유한 이름. 지시하는 문자열의 최대 길이는 8자입니다.
db2instance	char	db2instance ID. 이는 명령을 호출하는 사용자 ID입니다. 지시하는 문자열의 최대 길이는 8자입니다.

type	char	사용할 백업 유형이나 수행할 복원 유형을 지정합니다. 가능한 값은 다음과 같습니다.
		
		SQLUV_WRITE 조치의 경우:
		`0 - 전체 데이터베이스 백업`
		`3 - 테이블 공간 레벨 백업`
		
		SQLUV_READ 조치의 경우:
		`0 - 전체 복원`
		`3 - 온라인 테이블 공간 복원`
		`4 - 테이블 공간 복원`
		`5 - 실행기록 파일 복원`

dbname	char	백업하거나 복원할 데이터베이스의 이름. 지시하는 문자열의 최대 길이는 8자입니다.
alias	char	백업하거나 복원할 데이터베이스의 별명. 지시하는 문자열의 최대 길이는 8자입니다.

표 30. DB2-INFO 구조 내의 필드 (계속). 모든 필드는 NULL 종료되는 문자열입니다.

필드 이름	데이터 유형	설명
timestamp	char	백업 이미지를 식별하기 위해 사용되는 시간소인. 지시하는 문자열의 최대 길이는 26자입니다.
sequence	char	백업 이미지에 대한 파일 확장자를 지정합니다. 쓰기 조작의 경우, 첫 번째 세션의 값은 1이고 다른 세션이 sqluvint 호출로 시작될 때마다 값은 1씩 증가합니다. 읽기 조작의 경우, 값은 항상 0입니다. 지시하는 문자열의 최대 길이는 3자입니다.
obj_list	struct sqlu_gen_list	백업 이미지 내의 오브젝트 목록. 이는 정보용으로 벤더에 제공됩니다.
max_bytes_per_txn	sqlint32	사용자가 지정한 전송 버퍼 *크기*를 바이트 단위로 벤더에 지정합니다.
image_filename	char	차후 사용을 위해 예약됨
reserve	void	차후 사용을 위해 예약됨
nodename	char	백업이 생성된 노드의 이름
password	char	백업이 생성된 노드에 대한 암호
owner	char	백업 초기자의 ID
mcNameP	char	관리 클래스
nodeNum	SQL_PDB_NODE_TYPE	노드 번호. 255를 초과하는 숫자는 벤더 인터페이스에 의해 지원됩니다.

*filename*나, *server_id*, *db2instance*, *type*, *dbname* 및 *timestamp*는 백업 이미지를 고유하게 식별합니다. *sequence*에 의해 지정된 순차 번호는 파일 확장자를 식별합니다. 백업 이미지가 복원될 경우, 백업 이미지를 검색하려면 같은 값을 지정해야 합니다. 벤더 제품에 따라 *filename*이 사용되면 다른 매개변수가 NULL로 설정되거나, 그 반대일 수 있습니다.

언어 구문

C 구조

```
/* File: sqluvend.h */
/* ... */
typedef struct DB2_info
{
  char                  *DB2_id;
  char                  *version;
  char                  *release;
  char                  *level;
  char                  *action;
  char                  *filename;
  char                  *server_id;
  char                  *db2instance;
  char                  *type;
  char                  *dbname;
  char                  *alias;
  char                  *timestamp;
  char                  *sequence;
  struct sqlu_gen_list  *obj_list;
  long                  max_bytes_per_txn;
  char                  *image_filename;
  void                  *reserve;
  char                  *nodename;
  char                  *password;
  char                  *owner;
  char                  *mcNameP;
  SQL_PDB_NODE_TYPE     nodeNum;
} DB2_info;
/* ... */
```

VENDOR-INFO

이 구조에는 벤더와 장치 버전을 식별하는 정보가 있습니다.

표 31. VENDOR-INFO 구조 내의 필드. 모든 필드는 NULL 종료되는 문자열입니다.

필드 이름	데이터 유형	설명
vendor_id	char	벤더 식별자. 지시하는 문자열의 최대 길이는 64자입니다.
version	char	벤더 제품의 현재 버전. 지시하는 문자열의 최대 길이는 8자입니다.
release	char	벤더 제품의 현재 릴리스. 중요하지 않으면 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
level	char	벤더 제품의 현재 레벨. 중요하지 않으면 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
server_id	char	데이터베이스가 상주하는 서버를 식별하는 고유한 이름. 지시하는 문자열의 최대 길이는 8자입니다.
max_bytes_per_txn	sqlint32	지원되는 최대 전송 버퍼 크기. 벤더가 바이트 단위로 지정합니다. 이는 벤더 초기화 함수의 리턴 코드가 SQLUV_BUFF_SIZE(유효하지 않은 버퍼 크기를 지정했음을 나타냄)일 경우에만 사용됩니다.
num_objects_in_backup	sqlint32	완전한 백업을 수행하기 위해 사용된 세션 수. 이는 복원 조작 동안 모든 백업 이미지가 처리된 시기를 판별할 때 사용됩니다.
reserve	void	차후 사용을 위해 예약됨

언어 구문

C 구조

```
typedef struct Vendor_info
{
  char       *vendor_id;
  char       *version;
  char       *release;
  char       *level;
  char       *server_id;
  sqlint32   max_bytes_per_txn;
  sqlint32   num_objects_in_backup;
  void       *reserve;
} Vendor_info;
```

INIT-INPUT

이 구조에는 벤더 장치와의 논리 링크를 설정하기 위해 DB2에서 제공되는 정보가 있습니다.

표 32. INIT-INPUT 구조 내의 필드. 모든 필드는 NULL 종료되는 문자열입니다.

필드 이름	데이터 유형	설명
DB2_session	struct DB2_info	DB2 측면에서 세션의 설명
size_options	unsigned short	옵션 필드의 길이. DB2 백업 또는 복원 기능을 사용할 때, 이 필드의 데이터는 *VendorOptionsSize* 매개변수로부터 직접 전달됩니다.
size_HI_order	sqluint32	바이트 단위의 DB *크기* 계산에서 상위 32비트. 총 *크기*는 64비트입니다.
size_LOW_order	sqluint32	바이트 단위의 DB *크기* 계산에서 하위 32비트. 총 *크기*는 64비트입니다.
options	void	이 정보는 백업 또는 복원 기능이 호출될 때 응용프로그램에서 전달됩니다. 이 데이터 구조는 정확해야 합니다. 즉, 간접 레벨이 지원되지 않습니다. 바이트 리버설이 수행되지 않으며, 이 데이터에 대한 코드 페이지도 확인되지 않습니다. DB2 백업 또는 복원 기능을 사용할 때, 이 필드의 데이터는 *pVendorOptions* 매개변수로부터 직접 전달됩니다.
reserve	void	차후 사용을 위해 예약됨
prompt_lvl	char	백업 또는 복원 조작이 호출되었을 때 사용자가 요청한 프롬프팅 레벨. 지시하는 문자열의 최대 길이는 1자입니다.
num_sessions	unsigned short	백업 또는 복원 조작이 호출되었을 때 사용자가 요청한 세션 수

언어 구문

C 구조

```
typedef struct Init_input
{
  struct DB2_info  *DB2_session;
  unsigned short   size_options;
  sqluint32        size_HI_order;
  sqluint32        size_LOW_order;
  void             *options;
  void             *reserve;
  char             *prompt_lvl;
  unsigned short   num_sessions;
} Init_input;
```

INIT-OUTPUT

이 구조에는 벤더 장치에서 리턴한 출력이 있습니다.

표 33. INIT-OUTPUT 구조 내의 필드

필드 이름	데이터 유형	설명
vendor_session	struct Vendor_info	DB2에 대해 벤더를 식별하기 위한 정보가 있습니다.
pVendorCB	void	벤더 제어 블록
reserve	void	차후 사용을 위해 예약됨

언어 구문

C 구조

```
typedef struct Init_output
{
  struct Vendor_info  *vendor_session;
  void                *pVendorCB;
  void                *reserve;
} Init_output;
```

DATA

이 구조에는 DB2와 벤더 장치 사이에 전송된 데이터가 있습니다.

표 34. DATA 구조 내의 필드

필드 이름	데이터 유형	설명
obj_num	sqlint32	백업 조작 동안 DB2에서 지정한 순차 번호
buff_size	sqlint32	버퍼 크기
actual_buf_size	sqlint32	보내거나 받은 실제 바이트 수. *buff_size*를 초과할 수 없습니다.
dataptr	void	데이터 버퍼에 대한 포인터. DB2는 버퍼에 대한 공간을 할당합니다.
reserve	void	차후 사용을 위해 예약됨

언어 구문

C 구조

```
typedef struct Data
{
  sqlint32  obj_num;
  sqlint32  buff_size;
  sqlint32  actual_buff_size;
  void      *dataptr;
  void      *reserve;
} Data;
```

RETURN-CODE

이 구조에는 리턴 코드와 DB2에 리턴되는 오류에 대한 간단한 설명이 있습니다.

표 35. RETURN-CODE 구조 내의 필드

필드 이름	데이터 유형	설명
return_code[a]	sqlint32	벤더 함수에서 코드를 리턴합니다.
description	char	리턴 코드의 간단한 설명
reserve	void	차후 사용을 위해 예약됨
[a] 이는 다양한 DB2 API에서 리턴된 값과 같지 않은 벤더 고유의 리턴 코드입니다. 벤더 제품에서 승인된 리턴 코드에 대한 개별적인 API 설명을 참조하십시오.		

언어 구문

C 구조

```
typedef struct Return_code
{
  sqlint32  return_code,
  char      description[60],
  void      *reserve,
} Return_code;
```

벤더 제품을 사용한 백업 또는 복원 조작 호출

벤더 제품은 다음에서 DB2 백업 또는 DB2 복원 유틸리티를 호출할 때 지정할 수 있습니다.

- 제어 센터
- 명령행 처리기(CLP)
- API(Application Programming Interface)

제어 센터

제어 센터는 DB2와 함께 제공되는 데이터베이스 관리에 대한 그래픽 사용자 인터페이스입니다.

지정하는 방법	백업 또는 복원 조작을 위한 제어 센터 입력 변수
벤더 장치 및 라이브러리 이름 사용	*라이브러리를 사용하십시오.*. 라이브러리 이름 (UNIX 기반 시스템에서)이나 DLL 이름 (Windows 운영 체제 또는 OS/2에서)을 지정하십시오.
세션 수	*세션*
벤더 옵션	지원되지 않음
벤더 파일 이름	지원되지 않음
전송 버퍼 *크기*	백업의 *경우에는 각 버퍼의 크기*이고, 복원의 *경우에는* 적용할 수 없습니다.

명령행 처리기(CLP)

명령행 처리기(CLP)는 DB2 BACKUP DATABASE 또는 RESTORE DATABASE 명령을 호출하는데 사용될 수 있습니다.

지정하는 방법	명령행 처리기 매개변수	
	백업용	복원용
벤더 장치 및 라이브러리 이름 사용	*library-name*	*shared-library*
세션 수	*num-sessions*	*num-sessions*
벤더 옵션	지원되지 않음	지원되지 않음
벤더 파일 이름	지원되지 않음	지원되지 않음

지정하는 방법	명령행 처리기 매개변수	
	백업용	복원용
전송 버퍼 크기	*buffer-size*	*buffer-size*

API

두 가지의 API 함수 호출이 백업 및 복원 조작을 지원합니다. 백업을 위한 **sqlubkp**(105 페이지의 『데이터베이스 API 백업』 참조)와 복원을 위한 **sqlurestore** (138 페이지의 『데이터베이스 API 복원』 참조)가 있습니다.

지정하는 방법	**API 매개변수(sqlubkp 및 sqlurestore 둘다)**
벤더 장치 및 라이브러리 이름 사용	방법: 구조 *sqlu_media_list*에서 미디어 유형 SQLU_OTHER_MEDIA를 지정한 다음 구조 *sqlu_vendor*에서 *shr_lib*에 공유 라이브러리나 DLL을 지정하십시오.
세션 수	방법: 구조 *sqlu_media_list*에서 *sessions*를 지정하십시오.
벤더 옵션	*PVendorOptions*
벤더 파일 이름	방법: 구조 *sqlu_media_list*에서 미디어 유형 SQLU_OTHER_MEDIA를 지정한 다음 구조 *sqlu_vendor*에서 *filename*에 파일 이름을 지정하십시오.
전송 버퍼 크기	*BufferSize*

벤더 제품을 사용한 백업 또는 복원 조작 호출

부록J. 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 551 페이지의 『정보 센터로 정보에 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

이 책에는 모든 플랫폼에 공통적인 DB2 정보가 들어 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 플랫폼 공통 샘플 프로그램

이들 샘플은 Application Development Client와 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.

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. 546 페이지의 『인쇄된 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 언어 버전의 HTML 파일을 설치할 수 있습니다. 여기서 *%L*은 해당 언어의 로케일을 나타냅니다. 자세한 내용은 *빠른 시작* 책을 참조하십시오.

다음의 여러 가지 방법으로 DB2 책을 구하고 정보에 액세스할 수 있습니다.

- 550 페이지의 『온라인 정보 보기』
- 554 페이지의 『온라인 정보 검색』
- 546 페이지의 『인쇄된 책 주문』
- 545 페이지의 『PDF 책 인쇄』

표 36. 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. *관리 안내서: 구현*에서는 사용자의 설계 구현, 데이터베이스 액세스, 감사, 백업 및 복구와 같은 구현에 대한 정보를 제공합니다. *관리 안내서: 성능*에서는 데이터베이스의 환경, 응용프로그램 성능 평가 및 조정에 대한 정보를 제공합니다. 사용자는 문서 번호 SBOF-8934를 사용하여 세 권으로 된 *관리 안내서* 책을 주문할 수 있습니다.	SA30-0990 db2d1x70 SA30-0988 db2d2x70 SA30-0989 db2d3x70	db2d0
Administrative API Reference	데이터베이스를 관리하는 데 사용할 수 있는 DB2 API와 데이터 구조에 대해 설명합니다. 또한 응용프로그램에서 API를 호출하는 방법에 대해 설명합니다.	SC09-2947 db2b0x70	db2b0
응용프로그램 빌드 안내서	환경 설정 정보와 Windows, OS/2 및 UNIX 기반 플랫폼에서 DB2 응용프로그램을 컴파일, 링크 및 수행하는 방법에 대한 단계별 지침을 제공합니다.	SA30-0991 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 Universal Database 제품을 사용할 때 발생할 수 있는 APPC, CPI-C 및 SNA 감지 코드에 대한 일반 정보를 제공합니다. HTML 형식으로만 사용할 수 있습니다.	문서 번호 없음 db2apx70	db2ap

표 36. 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
연결성 보충 설명서	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 36. 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 36. 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 1-2-3 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 제공된 IBM 복제 도구에 대한 플랜, 구성, 관리 및 사용 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 36. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
SQL 시작하기	SQL 개념을 소개하고 많은 구조와 타스크에 대한 예를 제공합니다.	SA30-0996 db2y0x70	db2y0
SQL 참조서, 볼륨 1 및 볼륨 2	SQL 구문, 의미 및 언어 규칙에 대해 설명합니다. 또한 릴리스 간 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 대한 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 36. 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 대한 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 빠른 시작	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 36. 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 플랫폼 공통 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 Application Development Client가 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 36. 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 6번째 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
포르투갈어	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 다음을 수행하십시오.
   - UNIX 기반 플랫폼의 경우에는 Release.Notes 파일을 참조하십시오. 이 파일은 DB2DIR/Readme/*%L* 디렉토리에 있으며, 여기서 *%L*은 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
   - 다른 플랫폼의 경우에는 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 537 페이지의 표36을 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.

인쇄된 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 대한 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 37. 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 및 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** 단추를 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 플랫폼에서 사용할 수 없습니다.
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 수행할 경우, 정보 센터를 사용하십시오.
- 브라우저에서 **파일 —> 페이지 열기**를 누르십시오. 열린 페이지에 DB2 정보에 대한 설명과 링크가 들어 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 센터로 정보에 액세스

정보 센터는 DB2 제품 정보에 대한 빠른 액세스를 제공합니다. 정보 센터는 DB2 관리 도구를 사용할 수 있는 모든 플랫폼에서 사용할 수 있습니다.

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 센터에 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 Application Development Client와 함께 제공되는 샘플 프로그램. DB2 Application Development Client를 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 마우스의 오른쪽 단추로 누른 다음 **백업 —> 마법사를 사용한 데이터베이스**를 선택하십시오.
다중 사이트 갱신 구성	다중 사이트 갱신, 분산 트랜잭션 또는 2단계 확약을 구성합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스의 오른쪽 단추로 누른 다음 **다중 사이트 갱신**을 선택하십시오.

마법사	도움 내용	액세스하는 방법
데이터베이스 작성	데이터베이스 작성 및 일부 기본 구성 작업의 수행	제어 센터에서 **데이터베이스** 폴더를 마우스의 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 마우스의 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 테이블**을 선택하십시오.
테이블 공간 작성	새로운 테이블 공간을 작성합니다.	제어 센터에서 **테이블 공간** 아이콘을 마우스의 오른쪽 단추로 선택한 다음 **작성 —> 마법사를 사용한 테이블 공간**을 선택하십시오.
색인 작성	사용자의 모든 조회를 작성하고 삭제하기 위해 색인화합니다.	제어 센터에서 **색인** 아이콘을 마우스의 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 색인**을 선택하십시오.
성능 구성	비지니스 요구사항에 맞게 구성 매개변수를 갱신하여 데이터베이스의 성능을 조정합니다.	제어 센터에서 조정하려는 데이터베이스를 마우스의 오른쪽 단추로 누른 다음 **마법사를 사용한 성능 구성**을 선택하십시오. 파티션된 데이터베이스 환경에 대해 데이터베이스 파티션 뷰에서 성능을 조정하려는 첫 번째 파티션을 마우스의 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 마우스의 오른쪽 단추로 누른 다음 **복원 —> 마법사를 사용한 데이터베이스**를 선택하십시오.

문서 서버 설정

기본값으로 DB2 정보는 지역 시스템에 설치됩니다. 이는 DB2 정보에 액세스해야 하는 모든 사람이 동일한 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 변경되지 않게 하십시오.
2. 새로운 위치에 있는 파일을 찾도록 웹 서버를 구성하십시오. 자세한 내용은 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.
3. Java 버전의 정보 센터를 사용하면 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. 다음 페이지를 책갈피로 설정해 두면 도움이 됩니다.
   - 책 목록
   - 자주 사용되는 책의 목차
   - ALTER TABLE 주제와 같은 자주 참조하는 항목
   - 검색 양식

DB2 Universal Database 온라인 문서 파일을 중앙 머신에서 제공하는 방법에 대한 자세한 내용은 *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 5 51 페이지의 『정보 센터로 정보에 액세스』를 참조하십시오.

부록K. 주의사항

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 사용권까지 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

2바이트(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 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 시스템에서도 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

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

[숫자]

A

B

C

D

E

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

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 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 새로운 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.

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 제품이 이러한 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
데이터 복구 및 고가용성
안내 및 참조서
버전 7

책 번호: SA30-1389-00

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.

Spine information:

IBM® DB2® Universal Database

데이터 복구 및 고가용성 안내 및 참조서

버전 7