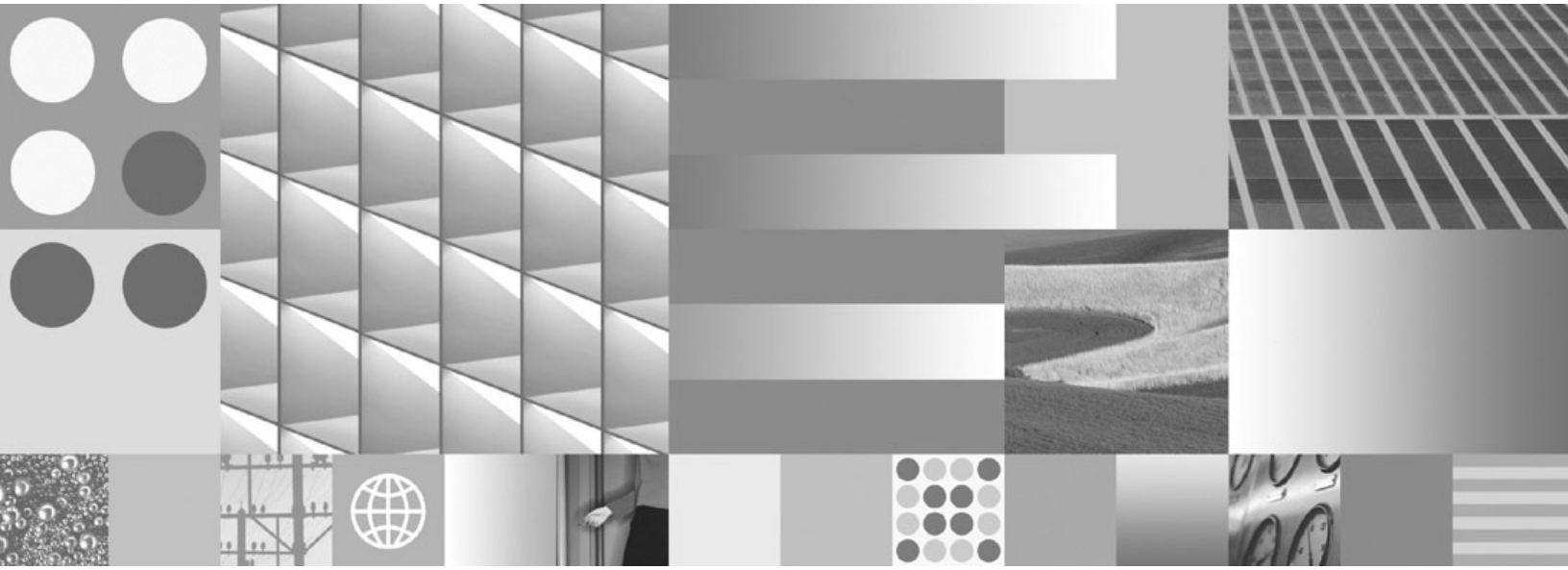
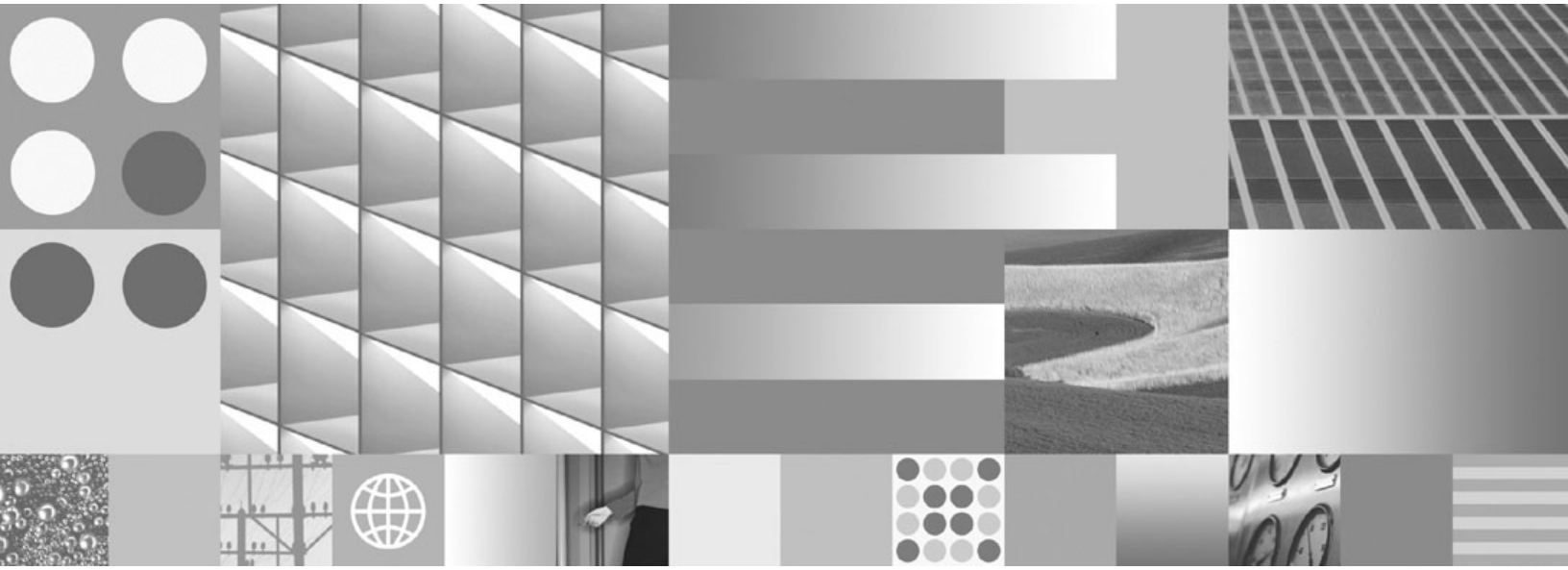
Linux, UNIX 및 Windows용
IBM DB2 9.7



데이터 복구 및 고가용성 안내서 및 참조서

SA30-3970-00

Linux, UNIX 및 Windows용
IBM DB2 9.7



데이터 복구 및 고가용성 안내서 및 참조서

SA30-3970-00

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 417 페이지의 부록 B『주의사항』에서 일반 정보를 읽어 보십시오.

개정판 주의사항

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

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

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

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

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

목차

이 책에 대한 정보

데이터 복구 및 고가용성 안내서 및 참조에서는 Linux®, UNIX® 및 Windows®용 DB2® 데이터베이스 데이터베이스 솔루션의 고가용성을 유지하는 방법과 데이터의 유실을 막는 방법에 대해 설명합니다.

데이터 복구 및 고가용성 안내서 및 참조는 다음과 같은 두 개의 내용으로 나뉘어져 있습니다.

- 제 1부 고가용성에서는 데이터베이스 솔루션의 고가용성을 유지하기 위한 전략과 DB2 데이터베이스 기능에 대해 설명합니다.
- 제 2부 데이터 복구에서는 데이터 유실을 막기 위한 DB2 백업 및 복구 기능에 대해 설명합니다.

제 1 부 고가용성

데이터베이스 솔루션의 사용 가능성은 성공적인 사용자 응용프로그램이 필요한 데이터베이스 태스크 수행 정도에 대한 측정입니다. 사용자 응용프로그램이 데이터베이스에 연결할 수 없거나 해당 트랜잭션이 오류로 실패하거나 시스템에서의 로드로 제한시간을 초과하는 경우 데이터베이스 솔루션은 사용 가능성이 높지 않습니다. 사용자 응용프로그램이 성공적으로 데이터베이스에 연결하여 해당 작업을 수행 중인 경우 데이터베이스 솔루션은 사용 가능성이 높습니다.

사용 가능성이 높은 데이터베이스 솔루션을 설계하거나 기존 솔루션의 사용 가능성을 높이려면 데이터베이스에 액세스하는 응용프로그램의 필요성을 이해해야 합니다. 추가 스토리지 스페이스, 더 빠른 프로세서 또는 더 많은 소프트웨어 라이센스로부터 최상의 이득을 얻으려면, 응용프로그램이 가장 필요로 할 때 사용자 비즈니스에 가장 중요한 응용프로그램에 대해 필요한 만큼 데이터베이스 솔루션을 사용 가능하도록 만드는 데 초점을 맞추십시오.

계획되지 않은 가동 중단

사용자에 대한 데이터베이스 솔루션 사용 가능성에 영향을 줄 수 있는 예기치 않은 시스템 오류로는 전원 인터럽트, 네트워크 정지, 하드웨어 장애, 운영 체제 또는 기타 소프트웨어 오류, 재해 발생 시 전체 시스템 장애 등이 있습니다. 사용자들이 데이터베이스에 대해 작업할 수 있을 것으로 예상할 때 이와 같은 실패가 발생하는 경우 사용 가능성이 높은 데이터베이스 솔루션이 다음을 수행해야 합니다.

- 사용자 응용프로그램을 실패로부터 보호하여, 사용자 응용프로그램이 실패를 인식하지 못하도록 합니다. 예를 들어, DB2 Data Server는 데이터베이스 서버가 실패하는 경우 데이터베이스 클라이언트 연결을 대체 데이터베이스 서버로 리라우트합니다.
- 해당 결과를 포함하도록 실패에 응답합니다. 예를 들어, 클러스터에 있는 하나의 머신에서 실패가 발생하면 클러스터 관리 프로그램은 더 이상 트랜잭션이 실패한 머신에서 처리하기 위해 라우트되지 않도록 클러스터에서 해당 머신을 제거할 수 있습니다.
- 실패에서 복구하여 시스템을 정상 조작으로 되돌립니다. 예를 들어, 대기 데이터베이스가 실패한 기본 데이터베이스의 데이터베이스 조작을 인계하는 경우 실패한 데이터베이스는 재시작하고 복구하여 다시 기본 데이터베이스로서 인계할 수 있습니다.

이 세 가지 태스크는 사용자 응용프로그램에 대한 솔루션 사용 가능성에 최소한의 영향을 주고 수행되어야 합니다.

계획된 가동 중단

사용 가능성이 높은 데이터베이스 솔루션에서, 사용자 응용프로그램에 대한 데이터베이스 사용 가능성의 유지보수 활동 영향도 최소화해야 합니다.

예를 들어, 데이터베이스 솔루션이 오전 9시에서 오후 5시 사이의 영업을 위해 열려 있는 상점 프런트에 서비스를 제공하는 경우, 유지보수 활동은 사용자 응용프로그램에 대한 데이터베이스의 사용 가능성에 영향을 주지 않고 해당 영업 시간을 벗어나서 오프라인에서 발생할 수 있습니다. 데이터베이스 솔루션이 매일 24시간 인터넷을 통해 고객이 액세스할 수 있을 것으로 예상되는 온라인 은행 거래에 서비스를 제공하는 경우, 유지보수 활동은 온라인에서 실행되거나, 고객에 대한 데이터베이스의 사용 가능성에 최소한의 영향을 주도록 한가한 활동 기간 동안 수행되도록 스케줄링해야 합니다.

데이터베이스 솔루션의 사용 가능성에 대한 비즈니스 결정 및 설계 선택을 수행할 때 다음의 두 가지 요인을 중요시해야 합니다.

- 고객이 사용할 수 없게 되는 데이터베이스 비즈니스에 대한 비용
- 특정 정도의 사용 가능성을 구현하는 비용

예를 들어, 특정 금액의 수익(X)을 내는 인터넷 기반 비즈니스를 고려해 보십시오. 매 시간 데이터베이스 솔루션은 고객에서 서비스를 제공합니다. 연간 10시간의 중단 시간을 줄이는 고가용성 전략은 연간 비즈니스 10X 추가 수익을 벌어들입니다. 이 고가용성 전략을 구현하는 비용이 예상되는 추가 수익 미만일 경우 구현할 가치가 있습니다.

제 1 장 가동 중단

가동 중단은 사용자 응용프로그램에 서비스를 제공하는 데이터베이스 솔루션 기능이 손상을 입었음을 의미합니다. 가동 중단은 계획되지 않은 가동 중단과 계획된 가동 중단의 두 그룹으로 분류할 수 있습니다.

계획되지 않은 가동 중단

계획되지 않은 가동 중단의 예는 다음과 같습니다.

- 하드웨어 또는 소프트웨어 실패를 포함한 시스템의 한 구성요소의 실패.
- 비즈니스에 중요한 트랜잭션에 필요한 테이블을 우발적으로 삭제하는 것과 같은 유효하지 않은 관리 또는 사용자 응용프로그램 조치.
- 차선 구성 또는 부적절한 하드웨어 또는 소프트웨어로 인한 성능 저하.

계획된 가동 중단

계획된 가동 중단의 예는 다음과 같습니다.

- 유지보수. 일부 유지보수 활동은 완전한 가동 중단이 필요합니다. 기타 유지보수 활동은 데이터베이스를 중지하지 않고 수행할 수 있지만 성능에 부정적인 영향을 줄 수 있습니다. 나중에 계획된 가동 중간의 가장 일반적인 유형입니다.
- 업그레이드. 소프트웨어 또는 하드웨어 업그레이드가 가끔 부분 또는 전체 가동 중단을 요구할 수 있습니다.

사용 가능성에 대한 논의에서 보통 손상 시나리오 또는 구성요소 실패에 초점을 둡니다.그러나 강력한 고가용성 솔루션을 설계하려면 이러한 모든 유형의 가동 중단을 다루어야 합니다.

가동 중단 시그니처

가동 중단 시그니처는 가동 중단의 특징인 증상 및 동작의 콜렉션입니다. 가동 중단의 시그니처는 일반 사용자에 대한 응답 속도 저하를 가져오는 임시 성능 문제부터 모든 시스템 중단까지 다양할 수 있습니다. 가동 중단 회피, 최소화 및 복구에 대한 전략을 고안할 때 이러한 변형이 비즈니스에 어떤 영향을 주는지 고려하십시오.

기능 정지(Blackout)

기능 정지 가동 중단 유형은 일반 사용자가 시스템을 완전히 사용할 수 없을 때 경험합니다. 이 유형의 가동 중단은 하드웨어, 운영 체제 또는 데이터베이스 레벨의 문제점이 원인일 수 있습니다. 기능 정지가 발생할 때 가동 중단 범위

가 즉시 식별되는 것이 필수적입니다. 가동 중단이 순전히 데이터베이스 레벨에서 발생합니까? 가동 중단이 인스턴스 레벨에 있습니까? 또는 운영 체제나 하드웨어 레벨에 있습니까?

등화관제(Brownout)

등화관제 유형의 가동 중단은 일반 사용자가 자신의 작업을 효과적으로 수행할 수 없는 수준까지 시스템 성능이 느려질 때 경험합니다. 시스템이 전체적으로는 가동되고 실행 중일 수 있지만, 본질적으로 일반 사용자의 눈에는 예상한 대로 작동 중이 아닙니다. 이 유형의 가동 중단은 시스템 유지보수 기간 및 최대 사용 기간 중에 발생할 수 있습니다. 일반적으로 CPU 및 메모리 사용량이 그러한 가동 중단 중에는 용량에 근접합니다. 빈약하게 조정되거나 과도하게 이용된 서버는 종종 절전에 도움이 됩니다.

가동 중단의 빈도와 지속 기간

데이터베이스 사용 가능성에 대한 대화에서, 초점은 보통 주어진 기간 동안 작동 중지 시간(또는 반대로 데이터베이스 시스템이 사용 가능한 시간)의 전체 양이나 백분율에 있습니다. 그러나 계획 또는 계획되지 않은 가동 중단의 빈도와 지속 기간에 따라 가동 중단이 비즈니스에 미치는 영향에 큰 차이가 있습니다.

수행하는 데 7시간이 걸리는 몇 가지 데이터베이스 시스템 업그레이드를 수행해야 하고, 낮은 사용자 활동 기간 중에 매일 1시간 동안 데이터베이스 시스템을 오프라인으로 하는 것과 가장 바쁜 요일의 가장 바쁜 파트 중에 7시간 동안 데이터베이스를 오프라인으로 하는 것 중에서 선택할 수 있는 상황을 고려하십시오. 여러 번의 작은 가동 중단이 한 번의 7시간 가동 중단보다 비용이 덜 들고 비즈니스 활동에 덜 유해합니다. 이제, 매주 총 몇 분 정도의 간헐적인 네트워크 실패가 있어서 소수의 트랜잭션이 정기적인 빈도로 실패한 상황을 고려하십시오. 이런 매우 짧은 가동 중단은 결국 많은 양의 수익을 대가로 지불하고 비즈니스 고객의 신뢰도에 돌이킬 수 없는 손상을 입혀서 더 큰 미래의 수익을 잃게 만들 수 있습니다.

전체 가동 중단(또는 사용 가능) 시간에만 집중하지는 마십시오. 유지보수 활동에 대해 결정할 때나 계획되지 않은 가동 중단에 반응할 때 여러 번의 더 작은 가동 중단의 비용에 대해 소수의 더 긴 가동 중단의 비용을 비교하십시오. 가동 중단 중에는 그러한 판단을 하기가 어려울 수 있습니다. 따라서 이러한 가동 중단 시그니처를 사용하여 비즈니스에 대한 비용을 계산하여 최상의 선택을 할 수 있도록 하십시오.

여러 번의 연쇄 실패

가동 중단 회피, 최소화 및 복구를 위한 데이터베이스 솔루션을 설계할 때, 여러 개의 구성요소가 동시에 실패하거나 한 구성요소의 실패가 다른 구성요소가 실패하도록 만들 가능성을 염두에 두십시오.

가동 중단 비용

가동 중단 비용은 비즈니스에 따라 다릅니다. 각 비즈니스는 우수 사례로서 직무에 중요한 비즈니스 프로세스에 대한 가동 중단 비용을 분석해야 합니다. 이 분석의 결과가 복원 계획을 공식화하는 데 사용됩니다. 이 계획에는 둘 이상의 프로세스가 식별되는 경우 복원 활동 중에 우선순위 순서화가 포함됩니다.

가동 중단 비용

고객 트랜잭션에 사용할 수 없는 데이터베이스 시스템을 대하는 고객의 비즈니스에 대한 비용을 계산할 수 있습니다. 예를 들어, 데이터베이스 시스템이 사용 불가능한 시간 및 분에 대해 영업 수익 손실의 평균 비용을 계산할 수 있습니다 고객 신뢰도 저하로 인한 예상 수익 손실을 계산하는 것을 훨씬 더 어렵지만, 비즈니스의 사용 가능성 요구사항을 평가할 때 이 비용을 고려해야 합니다.

비즈니스 프로세스에 사용 불가능한 내부 데이터베이스 시스템의 비용도 고려하십시오. 전자 우편 또는 달력 소프트웨어만큼 단순한 구성요소가 한 시간 동안 사용 불가능해도 비즈니스가 정지하게 만들 수 있습니다. 직원이 작업할 수 없기 때문입니다.

가동 중단 허용 한계

가동 중단의 허용 한계는 비즈니스에 따라 다릅니다. 각 비즈니스는 우수 사례로서 직무에 중요한 비즈니스 프로세스에 대한 가동 중단의 영향을 분석해야 합니다. 이 분석의 결과가 복원 계획을 공식화하는 데 사용됩니다. 이 계획에는 둘 이상의 프로세스가 식별되는 경우 복원에 대한 우선순위 순서가 포함됩니다.

가동 중단 허용 한계

사용 가능성 필요성 판별에서 중요한 인수는 비즈니스 또는 비즈니스의 특정 시스템이 가동 중단 어커런스에서 얼마나 허용 가능한지 묻는 것입니다. 예를 들어 메뉴 정보를 발행하기 위해 주로 웹 사이트를 조작하는 레스토랑은 우발적인 서버 가동 중단 때문에 많은 수익을 잃지는 않습니다. 다른 한편으로, 거래를 기록하는 주식 매매 서버의 가동 중단은 파멸적일 수 있습니다. 따라서 레스토랑 서버의 사용 가능성이 99.99%임을 보장하기 위해 많은 자원을 사용하는 것은 비용 효과가 없는 반면, 주식 매매의 경우에는 확실합니다.

허용 한계를 논의할 때 복구 시간과 복구 지점의 두 개념을 고려해야 합니다.

복구 시간은 비즈니스 프로세스 또는 시스템을 다시 온라인으로 만드는 데 필요한 시간입니다.

복구 지점은 비즈니스 프로세스 또는 시스템이 복원되는 실행기록 위치입니다. 데이터베이스 용어에서 플랜은 수행하는 데 더 오래 걸리는 점 이외에는 모든 트랜잭션을 잃

지 않는 완전한 복원에 대해 일부 트랜잭션을 잃는 빠른 복원의 이점을 비교 검토합니다.

복구 및 회피 전략

사용 가능성에 대한 시스템 설계 선택 및 구입을 고려할 경우, 고가용성 기능과 기술의 긴 목록으로 뛰어드는 것은 매력적입니다. 그러나 시스템을 고가용성으로 만들고 유지하는 것과 관련된 우수 사례는 기술을 구매하는 것만큼 좋은 설계 및 구성을 선택하고 견고한 관리 프로시저 및 비상 계획을 설계하고 실행하는 것입니다.

먼저 비즈니스 요구에 가장 잘 맞는 고가용성 전략을 식별함으로써 투자에 대한 가장 포괄적인 사용 가능성을 얻게 됩니다. 그런 다음 가장 적합한 기술을 선택하여 전략을 구현할 수 있습니다.

고가용성의 데이터베이스 솔루션을 설계하거나 구성할 때 가동 중단의 회피, 영향의 최소화 및 시스템 신속 복구 방법을 고려하십시오.

가동 중단 회피

가능하면 항상 가동 중단을 피하십시오. 예를 들어 단일 실패점을 제거하여 계획되지 않은 가동 중단을 피하거나, 계획된 가동 중단을 피하기 위해 유지보수 활동을 온라인으로 수행하는 방법을 연구하십시오. 데이터베이스 시스템을 모니터하여 문제점을 표시하는 시스템 동작의 경향을 식별하고 문제점이 가동 중단을 유발하기 전에 문제점을 해결하십시오.

가동 중단 영향 최소화

계획 및 계획되지 않은 가동 중단의 영향을 최소화하기 위해 데이터베이스 솔루션을 설계하고 구성할 수 있습니다. 예를 들어 구성요소 및 기능이 국지화되도록 데이터베이스 솔루션을 분산하여 일부 사용자 응용프로그램은 하나의 구성요소가 오프라인일 때도 트랜잭션 처리를 계속할 수 있도록 하십시오.

계획되지 않은 가동 중단에서 신속한 복구

복구 계획을 작성하십시오. 관리자가 계획되지 않은 가동 중단 발생 시 쉽고 신속하게 따를 수 있는 명확하고 잘 형식화된 프로시저를 작성하십시오. 관련된 시스템의 모든 구성요소를 설명하는 명확하게 구조화된 문서를 작성하십시오. 서비스 계약 및 연락 정보가 잘 구성되도록 하고 가까이 두십시오. 신속하게 복구하는 것이 극히 중요하지만, 미래에 가동 중단을 피하기 위해 가동 중단의 근본 원인을 식별하기 위해 수집할 진단 정보도 알아야 합니다.

제 2 장 고가용성 전략

자신의 데이터베이스 요청이 실패한 이유는 사용자에게 문제가 되지 않습니다. 낮은 성능으로 트랜잭션 시간종료가 발생했거나, 솔루션 구성요소가 실패했거나, 관리자가 유지보수를 수행하기 위해 데이터베이스를 오프라인으로 가져왔는지 여부에 관계없이, 결과는 사용자에게 같으며 데이터베이스는 요청을 처리할 수 없습니다.

데이터베이스 솔루션을 개선하는 전략은 다음과 같습니다.

중복성 실패 시 워크로드를 인계할 수 있는 각 솔루션 구성요소의 보조 사본을 갖습니다.

시스템 모니터링
쉽게 구성요소 밸런스를 유지하거나 구성요소가 실패했음을 발견하도록 솔루션 구성요소에 대한 통계를 수집합니다.

로드 밸런싱
오버로드된 솔루션 구성요소에서 로드가 가벼운 다른 솔루션 구성요소로 일부 워크로드를 전송합니다.

장애 복구
실패한 솔루션 구성요소에서 보조 구성요소로 모든 워크로드를 전송합니다.

성능 최대화
트랜잭션이 완료하거나 시간종료하는 데 너무 많은 시간이 소요되는 경우를 줄입니다.

유지보수 영향 최소화
가능한 한 사용자 응용프로그램에 미치는 영향이 적도록 자동 유지보수 활동 및 수동 유지보수 활동을 스케줄링합니다.

중복성을 통한 고가용성

데이터베이스 솔루션의 한 요소 또는 구성요소가 실패하는 경우 전원 공급장치나 네트워크를 연결하는 케이블에서 운영 체제 또는 응용프로그램 소프트웨어까지 최종 결과는 같습니다. 사용자 응용프로그램이 데이터베이스 솔루션을 사용할 수 없게 되는 것입니다. 고가용성 유지보수의 중요한 전략은 하나의 구성요소가 실패하는 경우 해당 구성요소의 보조 또는 백업 사본이 실패한 구성요소 대신 인계할 수 있도록 중복 시스템을 가지고 있어서 사용자 응용프로그램이 데이터베이스를 계속 사용 할 수 있도록 하는 것입니다.

중복성은 시스템 설계에서 일반적입니다.

- 무정전 또는 백업 전원 장치

- 각 구성요소 사이에 여러 네트워크 파이버 실행
- 네트워크 카드 본딩 또는 로드 밸런싱
- 중복 배열에서 여러 하드 드라이브 사용
- CPU 클러스터 사용

시스템의 이와 같은 구성요소 중 하나가 중복되지 않는 경우 그 구성요소는 전체 시스템에 대한 단일 실패 지점이 될 수 있습니다.

두 개의 데이터베이스, 즉 보통 모든 또는 대부분의 응용프로그램 워크로드를 처리하는 기본 데이터베이스와, 기본 데이터베이스가 실패하는 경우 워크로드를 인계받을 수 있는 보조 데이터베이스를 가지고 있으면 데이터베이스 레벨에서 중복성을 작성할 수 있습니다. DB2 고가용성 재해 복구(HADR) 환경에서는 이 보조 데이터베이스를 대기 데이터베이스라고 합니다.

장애 복구를 통한 고가용성

장애 복구는 기본 시스템에서 실패 발생 시 기본 시스템에서 보조 시스템으로 워크로드를 전송하는 것입니다. 이와 같이 워크로드가 전송된 경우, 보조 시스템은 실패한 기본 시스템의 워크로드를 인계했다고 합니다.

예 1

클러스터된 환경에서, 클러스터에 있는 하나의 머신이 실패하면 클러스터 관리 소프트웨어가 실패한 머신에서 실행 중이었던 프로세스를 클러스터의 다른 머신으로 이동할 수 있습니다.

예 2

여러 개의 IBM® Data Server가 사용되는 데이터베이스 솔루션에서, 데이터베이스 관리 프로그램은 더 이상 보조 데이터베이스 서버에 사용할 수 없는 데이터베이스 서버에 연결된 데이터베이스 응용프로그램을 리라우트할 수 있습니다.

시장에서 가장 일반적인 두 가지의 장애 복구 전략은 유휴 대기 및 상호 인계로 알려져 있습니다.

유휴 대기

이 구성에서는 보조 또는 대기 시스템이 유휴 상태에 있거나 대기 모드에 있고 기본 시스템에서 실패 발생 시 워크로드를 인계할 준비가 완료된 상태에 있는 동안 기본 시스템이 모든 워크로드를 처리합니다. HADR 설정에서, 대기 시스템이 읽기 전용 워크로드를 허용하도록 구성할 수도 있습니다.

상호 인계

이 구성에서는 여러 개의 시스템이 있고 각 시스템은 다른 시스템에 대해 지정된 보조 시스템입니다. 시스템이 실패하면 전체 성능에 부정적인 영향을 미칩니다. 실패한 시스템의 보조 시스템이 자체의 워크로드와 실패한 시스템의 워크로드를 계속 처리하기 때문입니다.

클러스터링을 통한 고가용성

클러스터는 단일 시스템으로 함께 작동하는 연결된 머신의 그룹입니다. 클러스터에 있는 하나의 머신이 실패하면 클러스터 관리 소프트웨어가 실패한 머신의 워크로드를 다른 머신으로 전송합니다.

하트비트 모니터링

클러스터에 있는 하나의 머신에서 실패를 발견하기 위해, 장애 복구 소프트웨어는 머신 사이에 하트비트 모니터링이나 킵얼라이브 패킷을 사용하여 사용 가능성을 확인할 수 있습니다. 하트비트 모니터링에는 클러스터에 있는 모든 머신 사이의 지속적인 통신을 유지보수하는 시스템 서비스가 관련됩니다. 하트비트가 발견되지 않으면 백업 머신에 대한 장애 복구가 시작됩니다.

IP 주소 인계

클러스터에 있는 하나의 머신에 실패가 있는 경우 클러스터 관리 프로그램은 머신 사이에 IP 주소를 전송하여 하나의 머신에서 다른 머신으로 워크로드를 전송할 수 있습니다. 이를 IP 주소 인계 또는 IP 인계라고 합니다. 이 전송은 클라이언트 응용프로그램에 표시되지 않으므로, 이 응용프로그램은 계속 원래 IP 주소를 사용하여 IP 주소가 맵핑되는 실제 머신이 변경된 것을 인식하지 못합니다.

DB2 고가용성(HA) 기능을 사용하여 IBM Data Server와 클러스터 관리 소프트웨어 사이에 통합할 수 있습니다.

데이터베이스 로깅

모든 데이터베이스에는 연관된 *로그*가 있습니다. 이 로그에서는 데이터베이스 변경 사항 레코드를 보존합니다. 마지막 전체 오프라인 백업 이후 지점으로 데이터베이스를 리스토어해야 하는 경우 실패 지점으로 데이터를 롤 포워드하려면 로그가 필요합니다. 데이터베이스 로깅은 고가용성 데이터베이스 솔루션 디자인의 핵심입니다. 데이터베이스 로그가 있으면 실패 지점으로 복구하고 기본 및 보조 데이터베이스를 동기화할 수 있기 때문입니다.

IBM 데이터 서버에서는 *순환* 및 *아카이브*와 같은 두 가지 로깅 유형을 지원합니다. 각각 여러 레벨의 복구 기능을 제공합니다.

- 10 페이지의『순환 로깅』
- 11 페이지의『아카이브 로깅』

아카이브 로깅을 선택하면 롤 포워드 복구에서 아카이브된 로그 및 사용 중인 로그 모두를 사용하여 로그 끝 또는 특정 시점으로 데이터베이스를 리스토어할 수 있다는 장점이 있습니다. 아카이브된 로그 파일을 사용하면 백업을 수행한 후 변경된 사항을 복구할 수 있습니다. 백업 시점까지만 복구할 수 있고 백업 이후의 모든 변경사항은 손실되는 순환 로그와는 다릅니다.

순환 로깅

순환 로깅은 데이터베이스가 새로 작성될 때의 디폴트 동작입니다. (*logarchmeth1* 및 *logarchmeth2* 데이터베이스 구성 매개변수가 OFF로 설정됩니다.) 이 유형의 로깅을 사용하는 경우 데이터베이스의 오프라인 백업이 허용됩니다. 전체 백업을 수행할 때 데이터베이스는 오프라인(사용자가 액세스할 수 없음)이어야 합니다.

이름에서 암시하듯이, 순환 로깅은 트랜잭션 실패 및 시스템 손상에서 복구하기 위해 온라인 로드 『링』을 사용합니다. 로그는 현재 트랜잭션의 무결성을 위해서만 사용되고 보유됩니다. 순환 로깅은 마지막 전체 백업 작업 후에 수행되는 트랜잭션을 통한 데이터베이스 롤 포워드는 허용하지 않습니다. 마지막 백업 작업 이후에 발생하는 모든 변경사항은 손실됩니다. 이 유형의 리스토어 작업은 데이터를 전체 백업이 수행된 시간의 특정 지점까지 복구하므로, *버전 복구*라고 합니다.

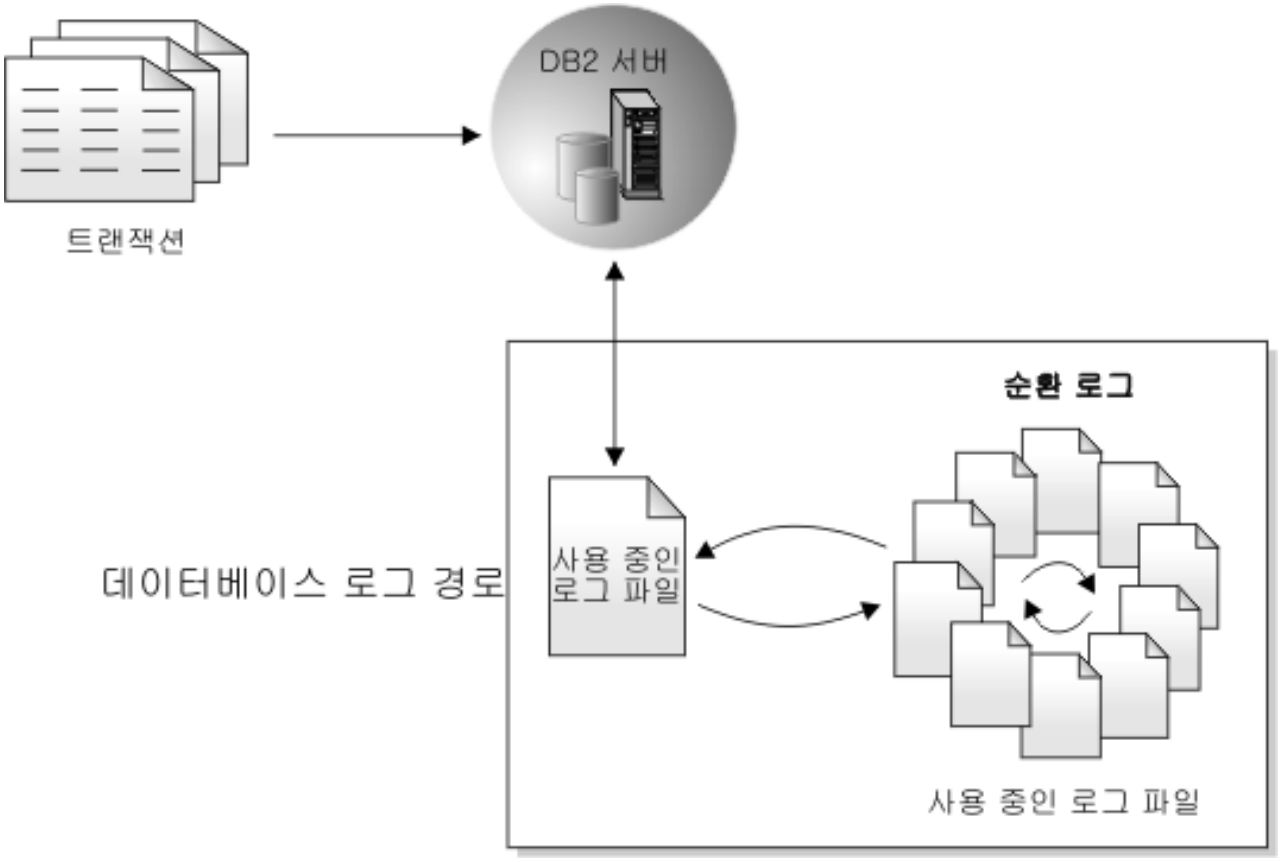


그림 1. 순환 로깅

사용 중인 로그는 데이터베이스가 불일치 상태에 있을 때 실패(시스템 전원 또는 응용프로그램 오류)를 방지하기 위한 응급 복구 중에 사용됩니다. 사용 중인 로그는 데이터베이스 로그 경로 디렉토리에 위치됩니다.

아카이브 로깅

아카이브 로깅은 특히 롤 포워드 복구에 사용됩니다. 아카이브된 로그는 활성화되었지만 더 이상 응급 복구에 필요하지 않은 로그입니다. 아카이브 로깅을 사용하려면 *logarchmeth1* 데이터베이스 구성 매개변수를 사용하십시오.

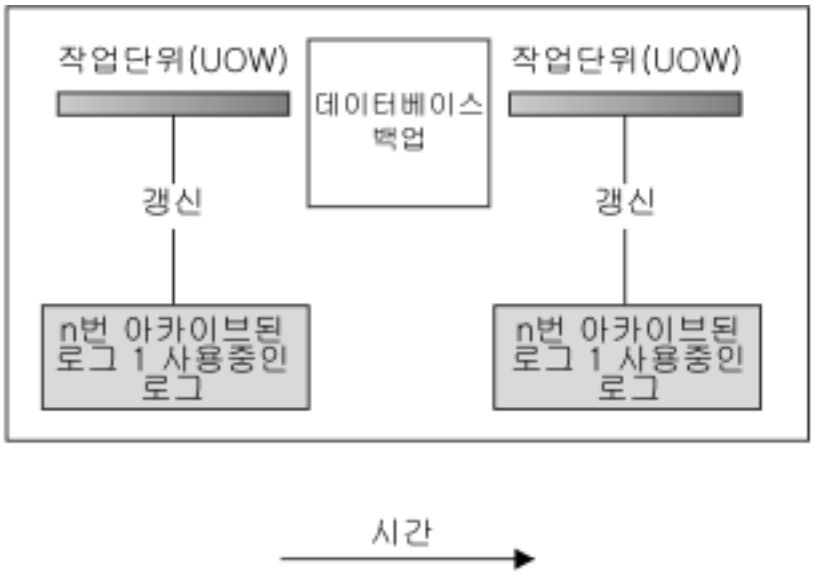


그림 2. 롤 포워드 복구에서 사용 중인 데이터베이스 로그와 아카이브된 데이터베이스 로그. 장기 실행 중인 트랜잭션이 있으면 사용 중인 로그가 둘 이상일 수 있습니다.

온라인 백업을 수행하는 것은 아카이브 로깅에 대해 데이터베이스가 구성된 경우에만 지원됩니다. 온라인 백업 작업 중, 데이터베이스에 대한 모든 활동이 로그됩니다. 온라인 백업 이미지가 리스토어되는 경우 로그는 최소한 백업 작업이 완료된 시점까지 롤 포워드되어야 합니다. 이러한 상황이 발생하도록 하려면 로그를 아카이브하고 데이터베이스 리스토어 시 사용 가능하도록 만들어야 합니다. 온라인 백업이 완료되면 DB2 데이터베이스 관리 프로그램은 현재 사용 중인 로그가 닫히도록 강요하고, 그 결과 아카이브됩니다. 그러면 온라인 백업에 복구에 사용 가능한 전체 아카이브 로그 세트가 생깁니다.

데이터베이스 구성 매개변수인 *newlogpath* 매개변수와, *logarchmeth1* 및 *logarchmeth2* 매개변수를 사용하여, 아카이브된 로그가 저장되는 위치를 변경할 수 있습니다. *newlogpath* 매개변수를 변경하면 사용 중인 로그가 저장되는 위치도 영향을 받습니다.

데이터베이스 로그 경로 디렉토리에서 아카이브된 로그인 로그 *Extent*를 판별하려면 *loghead* 데이터베이스 구성 매개변수의 값을 점검하십시오. 이 매개변수는 가장 낮은 번호가 매겨진 사용 중인 로그를 표시합니다. 시퀀스 번호가 *loghead*보다 작은 로그는 아카이브된 로그이므로 이동할 수 있습니다. 제어 센터를 사용하거나, 명령행 처리기 및 GET DATABASE CONFIGURATION 명령을 사용하여 이 매개변수의 값을 점검하여 "처음에 사용되는 로그 파일"을 볼 수 있습니다. 이 구성 매개변수에 대한 자세한 정보는 *Performance Guide* 책을 참조하십시오.

로그 제어 파일

데이터베이스가 실패 후 재시작하는 경우, 데이터베이스 관리 프로그램은 로그 파일에 저장된 트랜잭션 정보를 적용하여 데이터베이스를 일관성 있는 상태로 리턴합니다. 데이터베이스에 적용해야 하는 로그 파일의 레코드를 판별하기 위해 데이터베이스 관리 프로그램은 로그 제어 파일에 기록된 정보를 사용합니다.

데이터베이스 장애 허용성을 위한 중복성

데이터베이스 관리 프로그램은 두 개의 로그 제어 파일 사본 SQLOGCTL.LFH.1 및 SQLOGCTL.LFH.2를 유지보수하므로 한 사본이 손상되어도 데이터베이스 관리 프로그램이 계속 다른 사본을 사용할 수 있습니다.

성능 고려사항

로그 제어 파일에 포함된 트랜잭션 정보를 적용하면 실패 후 데이터베이스 재시작 오버헤드에 영향을 줍니다. 데이터베이스 관리 프로그램이 *Data Servers, Databases, and Database Objects Guide*에 있는 『softmax - 복구 범위 소프트 체크포인트 간격 구성 매개변수』를 사용하여 응급 복구 중에 처리해야 하는 로그 레코드 수를 줄이기 위해 디스크에 트랜잭션을 쓰는 빈도를 구성할 수 있습니다.

제 3 장 IBM Data Server에 대한 고가용성

IBM Data Server에는 많은 고가용성 전략을 지원하는 기능이 포함되어 있습니다.

자동 클라이언트 리라우트 로드맵

자동 클라이언트 리라우트는 실패한 서버의 클라이언트 응용프로그램을 대체 서버로 경로 재지정하여 응용프로그램이 인터럽트를 최소화하면서 작업을 계속할 수 있도록 하는 IBM Data Server 기능입니다. 자동 클라이언트 리라우트는 실패 전에 대체 서버가 지정된 경우에만 수행될 수 있습니다.

표 1에 각 범주의 관련 주제가 나열되어 있습니다.

표 1. 자동 클라이언트 리라우트 정보에 대한 로드맵

범주	관련 항목
일반 정보	• 29 페이지의 『자동 클라이언트 리라우트 제한사항』 • 24 페이지의 『자동 클라이언트 리라우트 설명 및 설정』 • *Quick Beginnings for DB2 Connect Servers*의 『자동 클라이언트 리라우트 설명 및 설정(DB2® Connect™)』
구성	• 28 페이지의 『자동 클라이언트 리라우트를 위한 대체 서버 식별』 • 26 페이지의 『레지스트리 변수를 사용하여 자동 클라이언트 리라우트 재시도 동작 구성』 • 29 페이지의 『JDBC 및 SQLJ용 IBM Data Server Driver 사용 시 클라이언트 리라우트 설정』
예	• 190 페이지의 『자동 클라이언트 리라우트 예』
기타 DB2 기능과의 상호작용	• 38 페이지의 『자동 클라이언트 리라우트 및 고가용성 재해 복구(HADR) 구성』 • 27 페이지의 『자동 클라이언트 리라우트와 함께 클라이언트 연결 시간종료 사용』 • *Developing Java Applications*의 『IBM Data Server Driver for JDBC and SQLJ 클라이언트 리라우트 지원』
문제점 해결	• 27 페이지의 『클라이언트 연결 분배기 기술에 대한 자동 클라이언트 리라우트 구성』

주: z/OS® Sysplex용 DB2에 대한 자동 클라이언트 리라우트는 IBM 데이터 서버 클라이언트 및 비Java IBM 데이터 서버 드라이버에서 사용 가능합니다. 이 지원을 사용하여 z/OS Sysplex용 DB2에 액세스하는 응용프로그램은 DB2 Connect 서버를 사용하지 않고 클라이언트가 제공하는 자동 클라이언트 리라우트 기능을 사용할 수 있습니

다. 이 기능에 대한 자세한 정보는 DB2 정보 센터에 있는 자동 클라이언트 리라우트(클라이언트 측)에 대한 주제를 참조하십시오.

Linux 및 UNIX용 DB2 결함 모니터 기능

UNIX 기반 시스템에서만 사용 가능한 DB2 결함 모니터 기능은 DB2 데이터베이스 관리 프로그램 인스턴스를 모니터하고 너무 일찍 종료하는 인스턴스를 다시 시작하여 DB2 Data Server 데이터베이스가 가동되어 실행 중인 상태를 유지합니다.

FMC(Fault Monitor Coordinator)는 UNIX 시동 시퀀스에서 시작되는 결함 모니터 기능 프로세스입니다. *init* 디먼은 FMC를 시작하고 FMC가 비정상적으로 종료되는 경우 다시 시작합니다. FMC는 DB2 인스턴스마다 하나의 결함 모니터를 시작합니다. 각 결함 모니터는 디먼 프로세스로 실행되며 DB2 인스턴스와 동일한 사용자 특권을 갖습니다.

결함 모니터가 시작되면, 너무 일찍 종료하지 않도록 모니터를 받습니다. 결함 모니터가 실패하면 FMC에 의해 다시 시작됩니다. 각각의 결함 모니터는 차례로 하나의 DB2 인스턴스를 모니터해야 합니다. DB2 인스턴스가 너무 일찍 종료하면 결함 모니터가 그 인스턴스를 다시 시작합니다. 결함 모니터는 db2stop 명령이 발행되는 경우에 비활성 상태가 됩니다. DB2 인스턴스가 다른 방식으로 종료하면 결함 모니터는 다시 시작합니다.

DB2 결함 모니터 제한사항

고가용성 클러스터링 제품(예: HACMP™, MSCS 또는 IBM Tivoli® System Automation for Multiplatforms)을 사용 중인 경우 결함 모니터 기능을 해제해야 합니다. 인스턴스 시작 및 종료가 클러스터링 제품에 의해 제어되기 때문입니다.

DB2 결함 모니터 및 DB2 Health Monitor 사이의 다른 점

Health Monitor와 결함 모니터는 단일 데이터베이스 인스턴스에서 작동하는 도구입니다. Health Monitor는 *Health 표시기*를 사용하여 데이터베이스 관리 프로그램 성능이나 데이터베이스 성능의 특정 측면 상태(health)를 평가합니다. Health 표시기는 테이블 스페이스와 같은 특정 데이터베이스 오브젝트 클래스의 일부 측면에 대한 상태를 측정합니다. Health 표시기는 해당되는 데이터베이스 오브젝트 클래스의 상태를 판별하기 위해 특정 기준에 대해 평가할 수 있습니다. 또한 Health 표시기는 표시기가 임계값을 초과하거나 데이터베이스 오브젝트가 비정상 상태에 있음을 표시하는 경우 통지하기 위해 경보를 생성할 수 있습니다.

비교해 보면, 결함 모니터만 모니터 중인 인스턴스가 가동 및 실행 중 상태로 유지되도록 하는 책임을 가지고 있습니다. 모니터 중인 DB2 인스턴스가 예상치 않게 종료하는 경우 결함 모니터는 인스턴스를 다시 시작합니다. 결함 모니터는 Windows에서 사용할 수 없습니다.

고가용성 재해 복구(HADR)

DB2 Data Server 고가용성 재해 복구(HADR) 기능은 부분 및 전체 사이트 실패에 대한 고가용성 솔루션을 제공하는 데이터베이스 복제 기능입니다. HADR은 기본 데이터베이스라고 하는 소스 데이터베이스에서 대기 데이터베이스라고도 하는 목표 데이터베이스로 데이터 변경사항을 복제하여 데이터 손실을 보호합니다.

모든 또는 대부분의 데이터베이스에 보호가 필요하거나, 대기 데이터베이스에서 자동으로 복제해야 하는 DDL 조작을 수행하는 경우 HADR은 최상의 옵션이 될 수 있습니다.

응용프로그램은 현재 기본 데이터베이스에만 액세스할 수 있습니다. 대기 데이터베이스는 기본 데이터베이스에서 생성되어 대기 데이터베이스로 제공된 로그 데이터를 롤 포워드함으로써 갱신됩니다.

부분 사이트 실패는 하드웨어, 네트워크 또는 소프트웨어(DB2 데이터베이스 시스템 또는 운영 체제) 실패 때문에 발생될 수 있습니다. HADR이 없는 경우, 부분 사이트 실패가 발생하면 데이터베이스를 포함하는 데이터베이스 관리 시스템(DBMS) 서버를 재시작해야 합니다. 데이터베이스와 데이터베이스가 있는 서버를 재시작하는 데 소요되는 시간은 예상할 수 없습니다. 데이터베이스가 다시 일관성 있는 상태로 돌아와서 사용 가능하게 되려면 몇 분이 소요될 수 있습니다. HADR을 사용하여 대기 데이터베이스를 초 단위로 인계할 수 있습니다. 또한, 응용프로그램에서 자동 클라이언트 리라우트 또는 재시도 논리를 사용하여 원래의 기본 데이터베이스를 사용 중이었던 클라이언트를 대기 데이터베이스(새 기본 데이터베이스)로 경로 재지정할 수 있습니다.

화재와 같은 재해로 인해 전체 사이트가 파괴되는 경우 전체 사이트 실패가 발생할 수 있습니다. HADR은 기본 및 대기 데이터베이스 간의 통신에 TCP/IP를 사용하므로, 기본 및 대기 데이터베이스는 서로 다른 위치에 존재할 수 있습니다. 예를 들어, 기본 데이터베이스는 한 도시에 있는 본부에 위치하고 대기 데이터베이스는 다른 도시에 있는 영업부에 위치할 수 있습니다. 기본 사이트에서 재해가 발생하는 경우, 리모트 대기 데이터베이스가 전체 DB2 기능을 가지고 있는 기본 데이터베이스로서 인계받도록 하여 데이터 사용 가능성이 유지됩니다. 인계 조작 후에 원래의 기본 데이터베이스 백업을 가져온 다음 이것을 기본 데이터베이스 상태로 되돌릴 수 있습니다. 이를 장애 회복이라고 합니다.

HADR을 사용하여 세 가지 동기화 모드(동기, 근접 동기 또는 비동기) 중 하나를 지정하여 가능한 데이터 손실로부터 원하는 보호 레벨을 선택할 수 있습니다.

실패한 원래의 기본 서버가 수리되면, 데이터베이스의 두 사본이 일관성 있는 상태가 될 수 있는 경우에 대기 데이터베이스로서 HADR 쌍을 다시 조인할 수 있습니다. 원래

기본 데이터베이스가 다시 대기 데이터베이스로서 HADR 쌍에 통합되면, 원래의 기본 데이터베이스가 다시 기본 데이터베이스가 될 수 있도록 데이터베이스의 역할을 전환할 수 있습니다.

HADR은 단지 DB2 제품군에서 제공되는 몇 가지 복제 솔루션 중 하나입니다. WebSphere® Federation Server 및 DB2 데이터베이스 시스템에는 고가용성을 제공하기 위해 일부 구성에서 사용할 수도 있는 SQL 복제 및 Q 복제 솔루션이 포함되어 있습니다. 이 기능은 여러 위치에서 일관성 있는 데이터베이스 테이블 사본을 논리적으로 유지보수합니다. 또한 컬럼 및 행 필터링, 데이터 변환, 테이블 사본 갱신사항과 같은 복잡한 기능과 유연성을 제공하므로, 파티션된 데이터베이스 환경에서 사용할 수 있습니다.

DB2 고가용성(HA) 기능

DB2 고가용성(HA) 기능을 사용하여 IBM Data Server와 클러스터 관리 소프트웨어 사이에 통합할 수 있습니다.

클러스터된 환경에서 데이터베이스 관리 프로그램 인스턴스를 중지할 때 인스턴스가 중지된 것을 클러스터 관리 프로그램이 인식하도록 해야 합니다. 클러스터 관리 프로그램이 인스턴스가 중지된 것을 인식하지 못하면, 클러스터 관리 프로그램은 중지된 인스턴스에 장애 복구와 같은 작업을 시도할 수 있습니다. DB2 고가용성(HA) 기능은 인스턴스 구성 변경사항(예: 데이터베이스 관리 프로그램 인스턴스 중지)으로 클러스터가 변경되어야 할 때 데이터베이스 관리 프로그램이 사용자의 클러스터 관리 프로그램과 통신할 수 있도록 하는 인프라스트럭처(infrastructure)를 제공합니다.

인스턴스 변경사항으로 인해 클러스터가 변경되어야 할 때마다 데이터베이스 관리 프로그램이 클러스터 관리 프로그램과 통신하는 경우, 사용자는 인스턴스 구성 변경사항을 수행한 후 별도의 클러스터 작업을 수행하지 않아도 됩니다.

DB2 HA 기능은 다음 요소로 구성됩니다.

- IBM Tivoli SA MP(System Automation for Multiplatforms)는 DB2 고가용성(HA) 기능의 일부로 AIX® 및 Linux의 IBM Data Server와 함께 번들로 제공되며 DB2 설치 프로그램으로 통합됩니다. DB2 설치 프로그램이나, IBM Data Server 설치 미디어에 포함된 installSAM 및 uninstallSAM 스크립트를 사용하여 SA MP를 설치, 업그레이드 또는 설치 제거할 수 있습니다.
- 클러스터 환경에서 일부 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작은 관련 클러스터 구성 변경이 필요합니다. DB2 고가용성(HA) 기능은 사용자가 특정 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작을 수행할 때마다 데이터베이스 관리 프로그램이 자동으로 클러스터 관리 프로그램 구성 변경을 요청할 수 있도록 합니다. 참조: 83 페이지의 『DB2 고가용성(HA) 기능을 사용하도록 자동으로 클러스터 구성』

- DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)는 클러스터된 환경에서 고가용성 데이터베이스를 구성 및 관리하기 위해 사용할 수 있는 텍스트 기반 유틸리티입니다. db2haicu는 사용자 시스템을 쿼리하여 데이터베이스 인스턴스, 사용자의 클러스터 환경 및 사용자의 클러스터 관리 프로그램에 대한 정보를 수집합니다. db2haicu 프롬프트에 정보를 제공하여 db2haicu 호출, 입력 파일에, 또는 런타임에서 매개변수를 통해 자세한 정보를 제공합니다. 참조: 92 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)』
- The DB2 클러스터 관리 프로그램 API는 데이터베이스 관리 프로그램이 클러스터 관리 프로그램의 구성 변경사항을 통신할 수 있도록 하는 기능 세트를 정의합니다. 참조: 131 페이지의 『DB2 클러스터 관리 프로그램 API』

로그 제공을 통한 고가용성

로그 제공은 전체 로그 파일을 아카이브 디바이스에서, 또는 기본 데이터베이스에 대해 실행 중인 User Exit 프로그램을 통해 대기 머신에 복사하는 프로세스입니다.

대기 데이터베이스는 프로덕션 머신에서 생성된 로그 파일을 통해 연속적으로 롤 포워드합니다. 프로덕션 머신이 실행하는 경우 장애 복구가 발생하고 다음 상황이 발생합니다.

- 남아 있는 로그가 대기 머신으로 전송됩니다.
- 대기 데이터베이스가 로그 끝까지 롤 포워드한 후 중지합니다.
- 클라이언트가 대기 데이터베이스에 다시 연결한 후 조작을 다시 시작합니다.

대기 머신은 자체 소유 자원(예: 디스크)을 가지고 있지만 프로덕션 데이터베이스와 동일한 실제 및 논리 자원을 가지고 있어야 합니다. 이 접근방식을 사용하는 경우 기본 데이터베이스는 리스토어 유틸리티나 미러 분할 기능을 사용하여 대기 머신으로 리스토어됩니다.

재해 복구 상황에서 데이터베이스를 복구할 수 있도록 하려면 다음을 고려하십시오.

- 아카이브 위치는 지리적으로 기본 사이트와 구분해야 합니다.
- 대기 데이터베이스 사이트에서 로그를 리모트로 미러링하십시오.
- 지원 손실이 없도록 동기식 미러를 사용하십시오. ESS 및 EMC와 같은 모던 디스크 서브시스템을 사용하거나 다른 리모트 미러링 기술을 사용하여 수행할 수 있습니다. NVRAM 캐시(로컬 및 리모트 모두) 역시 재해 복구 상황의 성능 영향을 최소화하기 위해 권장됩니다.

주:

1. 기본 데이터베이스에서 인덱스 재빌드가 발생했음을 표시하는 로그 레코드를 대기 데이터베이스가 처리하는 경우, 대기 서버의 인덱스는 자동으로 재빌드되지 않습니다. 인덱스는 처음 데이터베이스에 연결하거나 대기 서버가 롤 포워드 보류 상태를

벗어나서 취해진 후 처음 인덱스에 액세스하려고 할 때 대기 서버에서 재빌드됩니다. 기본 서버의 인덱스가 재빌드되는 경우 대기 서버를 기본 서버와 재동기화할 것을 권장합니다. LOGINDEXBUILD 데이터베이스 구성 매개변수를 설정한 경우 롤포워드 작업 중에 인덱스를 재빌드할 수 있습니다.

2. 로드 유틸리티가 COPY YES 옵션을 지정하여 기본 데이터베이스에 대해 실행되는 경우, 대기 데이터베이스는 사본 이미지에 대한 액세스 권한을 가지고 있어야 합니다.
3. 로드 유틸리티가 COPY NO 옵션을 지정하여 기본 데이터베이스에 대해 실행되는 경우, 대기 데이터베이스를 재동기화해야 합니다. 그렇지 않으면 테이블 스페이스가 리스토어 보류 상태에 놓입니다.
4. 대기 머신을 초기화하는 방법은 두 가지입니다.
   a. 백업 이미지에서 리스토어합니다.
   b. 프로덕션 시스템의 분할 미러를 작성하고 STANDBY 옵션을 사용하여 db2inidb 명령을 발행합니다.

   대기 머신이 초기화된 후에만 대기 시스템에서 ROLLFORWARD 명령을 발행할 수 있습니다.
5. 로그되지 않은 작업은 대기 데이터베이스에서 재생되지 않습니다. 결과적으로, 이와 같은 작업 후에는 대기 데이터베이스를 재동기화할 것을 권장합니다. 온라인 미러 분할 및 일시중단된 입출력 지원을 통해 재동기화할 수 있습니다.

로그 미러링

IBM 데이터 서버에서는 데이터베이스 레벨에서 로그 미러링을 지원합니다. 로그 파일을 미러링하면 사용 중인 로그의 우연한 삭제 및 하드웨어 장애로 인한 데이터 손상으로부터 데이터베이스를 보호할 수 있습니다.

사용 중인 로그가 손상될 수 있다는 점(디스크 손상으로 인해)이 염려되면, 사용 중인 로그의 사본을 관리할 데이터베이스에 대한 2차 경로를 지정하는 MIRRORLOGPATH 구성 매개변수를 사용하여 로그를 저장하는 볼륨 미러링을 고려해야 합니다.

MIRRORLOGPATH 구성 매개변수를 사용하면 데이터베이스에서 동일한 2차 로그 파일 사본을 다른 경로에 작성할 수 있습니다. 실제로 분리된 디스크(다른 디스크 제어기에 있는 디스크가 선호됨)에 2차 로그 경로를 배치하는 것이 좋습니다. 그러면 디스크 제어기가 단일 오류 지점이 될 수 없습니다.

MIRRORLOGPATH를 처음 사용하는 경우 다음 데이터베이스 시작 때까지 실제로 사용되지 않습니다. 이는 NEWLOGPATH 구성 매개변수와 비슷합니다.

미러링 로그 경로 또는 사용 중인 로그 경로에 작성하는 중 오류가 발생하면 데이터베이스는 장애가 발생한 경로를 『잘못된(bad)』 항목으로 표시하고 관리 통지 로그에 메

시지를 작성한 다음, 나머지 『올바른(good)』 로그 경로에만 다음 로그 레코드를 작성합니다. DB2에서는 현재 로그 파일이 가득 차거나 절단될 때까지 『잘못된(bad)』 경로를 다시 사용하지 않습니다. DB2에서 다음 로그 파일을 열어야 하는 경우 이 경로가 유효한지 검증하고 유효하면 해당 경로를 사용하기 시작합니다. 그렇지 않으면 처음으로 다음 로그 파일에 액세스할 때까지 DB2는 다시 경로를 사용하지 않습니다. 로그 경로를 동기화하지는 않지만 DB2는 로그 파일을 아카이브할 때 올바른 경로를 사용하도록 발생하는 액세스 오류에 대한 정보를 보존합니다. 나머지 『올바른(good)』 경로에 쓰는 중 오류가 발생할 경우, 데이터베이스가 종료됩니다.

일시중단된 입출력 및 온라인 분할 미러 지원을 통한 고가용성

IBM Data Server 일시중단 입출력 지원을 사용하면 데이터베이스가 오프라인이 아니어도 기본 데이터베이스의 미러링된 사본을 분할할 수 있습니다. 이를 사용하여 기본 데이터베이스가 실패하는 경우 인계받을 대기 데이터베이스를 매우 신속하게 작성하여 수 있습니다.

디스크 미러링은 데이터를 두 개의 분리된 하드 디스크에 동시에 쓰는 프로세스입니다. 하나의 데이터 사본을 다른 데이터 사본의 미러라고 합니다. 미러를 분할하는 것은 두 사본을 분리하는 프로세스입니다.

디스크 미러링을 사용하여 기본 데이터베이스의 보조 사본을 유지보수할 수 있습니다. IBM Data Server 일시중단 입출력 기능을 사용하면 데이터베이스가 오프라인이 아니어도 데이터베이스의 기본 및 보조 미러된 사본을 분할할 수 있습니다. 기본 및 보조 데이터베이스 사본이 분할되면, 기본 데이터베이스가 실패하는 경우 보조 데이터베이스가 작업을 인계받을 수 있습니다.

IBM Data Server 백업 유틸리티를 사용하여 대형 데이터베이스를 백업하지 않으려는 경우, 일시중단된 입출력 및 분할 미러 기능을 사용하여 미러된 이미지에서 사본을 작성할 수 있습니다. 이 접근방식은 또한 다음을 수행합니다.

- 프로덕션 머신에서 백업 작업 오버헤드를 제거합니다.
- 시스템을 클론할 빠른 방법을 나타냅니다.
- 유휴 대기 장애 복구의 급속 구현을 나타냅니다. 초기 리스토어 작업이 없는데 롤포워드 작업이 너무 느린 것으로 증명되거나 오류가 발생하면 다시 초기화가 아주 빠릅니다.

db2inidb 명령은 분할 미러를 초기화하여 다음과 같이 사용 가능하도록 합니다.

- 클론 데이터베이스로서
- 대기 데이터베이스로서
- 백업 이미지로서

이 명령은 분할 미러에 대해서만 발행할 수 있으며 분할 미러를 사용하기 전에 실행해야 합니다.

파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션에 대해 동시에 입출력 쓰기를 일시중단하지 않아도 됩니다. 하나 이상의 데이터베이스 파티션으로 구성된 서브세트를 일시중단하여 오프라인 백업을 수행하기 위한 분할 미러를 작성할 수 있습니다. 카탈로그 파티션이 서브세트에 포함되는 경우, 일시중단할 마지막 데이터베이스 파티션이어야 합니다.

파티션된 데이터베이스 환경의 경우 데이터베이스 파티션의 분할 이미지를 사용하려면 모든 데이터베이스 파티션에서 db2inidb 명령을 실행해야 합니다. 도구는 db2_all 명령을 사용하여 모든 데이터베이스 파티션에서 동시에 실행할 수 있습니다. 그러나 RELOCATE USING 옵션을 사용하는 경우에는 db2_all 명령을 사용하여 모든 파티션에서 동시에 db2inidb를 실행할 수 없습니다. 데이터베이스 파티션마다 변경될 데이터베이스 파티션의 NODENUM 값을 포함하는 개별 구성 파일을 제공해야 합니다. 예를 들어, 데이터베이스 이름이 변경되는 경우 모든 데이터베이스 파티션이 영향을 받으며 각 데이터베이스 파티션의 개별 구성 파일과 함께 db2relocatedb 명령을 실행해야 합니다. 단일 데이터베이스 파티션에 속하는 컨테이너를 이동하는 경우 db2relocatedb 명령은 해당 데이터베이스 파티션에서 한 번만 실행해야 합니다.

주: 분할 미러에 볼륨 디렉토리를 포함하여 데이터베이스를 구성하는 모든 컨테이너 및 디렉토리가 포함되어 있는지 확인하십시오. 이 정보를 수집하려면 분할해야 하는 데이터베이스의 모든 파일 및 디렉토리를 표시하는 DBPATHS 관리 뷰를 참조하십시오.

제 4 장 고가용성 구성

고가용성을 위해 DB2 데이터베이스 솔루션을 구성하려면, 데이터베이스 유지보수 활동을 스케줄하고, 실패 시 서로에 대해 알고 관련 역할을 알도록 기본 및 대기 데이터베이스를 구성하고, 실패한 클러스터 노드에서 워크로드를 전송하도록 모든 클러스터 관리 소프트웨어를 구성해야 합니다.

데이터베이스 솔루션을 구성하기 전에 다음을 수행하십시오.

- 솔루션을 구성하는 기본 하드웨어 및 소프트웨어 구성요소를 어셈블하고 설치하십시오. 이러한 기본 구성요소에는 전원 공급 장치, 네트워크 연결, 네트워크 카드, 디스크나 기타 스토리지 디바이스, 운영 체제 및 클러스터 관리 소프트웨어가 포함될 수 있습니다.
- 데이터베이스 워크로드 없이 이러한 기본 구성요소를 테스트하여 데이터베이스 로드 밸런싱, 장애 복구 또는 복구 작업에서 사용하기 전에 해당 구성요소가 제대로 기능하고 있는지 확인하십시오.

중복은 고가용성 솔루션의 중요한 부분입니다. 그러나 유지보수를 현명하게 스케줄하지 않는 경우, 필요한 복구 로그용 스토리지 공간이 부족한 경우 또는 클러스터 관리 소프트웨어가 올바르게 구성되지 않은 경우, 사용자가 데이터베이스를 사용하여 중요한 작업을 수행해야 할 때 솔루션을 사용하지 못할 수 있습니다.

고가용성 구성에는 다음이 포함됩니다.

- 클라이언트 리라우트 구성

  24 페이지의『자동 클라이언트 리라우트 설명 및 설정』
- 결함 모니터 구성

  32 페이지의『DB2 결함 모니터 레지스트리 파일』
- Configure DB2 고가용성 재해 복구 구성

  35 페이지의『고가용성 재해 복구(HADR) 초기화』
- 유지보수 활동 스케줄

  62 페이지의『고가용성을 위한 유지보수 스케줄링』
- 로깅 구성

  65 페이지의『데이터베이스 로깅 옵션 구성』
- 클러스터 관리 소프트웨어 구성

  80 페이지의『고가용성을 위한 클러스터링 환경 구성』

자동 클라이언트 리라우트 설명 및 설정

자동 클라이언트 리라우트 기능의 주 목적은 IBM Data Server Client 응용프로그램이 최소한의 인터럽트로 계속 작업할 수 있도록 통신 실패로부터 복구할 수 있도록 하는 것입니다. 이름에 제시된 것처럼 리라우트의 중심은 연속 작업 지원입니다. 그러나 리라우트는 클라이언트 연결에 식별된 대체 위치가 있는 경우에만 가능합니다.

자동 클라이언트 리라우트 기능은 서버가 Linux, UNIX 및 Windows의 DB2인 경우 다음 구성 가능 환경 내에서 사용할 수 있습니다.

1. 데이터베이스 파티셔닝 기능(DPF)이 있는 ESE(Enterprise Server Edition)
2. WebSphere Replication Server
3. 고가용성 클러스터 멀티프로세서(HACMP)
4. 고가용성 재해 복구(HADR)

   자동 클라이언트 리라우트는 액세스하는 데이터베이스의 장애 복구 후에 최소한의 인터럽트로 클라이언트 응용프로그램이 계속 작업을 수행할 수 있도록 HADR과 함께 작동합니다.

자동 클라이언트 리라우트 기능은 또한 데이터베이스 서버가 System i® 또는 System z®에 있는 경우 다음 구성에서도 사용할 수 있습니다.

1. IBM Data Server Client는 대체 서버를 가지고 있는 DB2 Connect 서버를 통해 z/OS 또는 i5/OS® 시스템에 연결합니다. 자동 클라이언트 리라우트는 IBM Data Server Client와 두 개의 DB2 Connect 서버 사이에 사용됩니다.
2. z/OS Sysplex용 DB2 데이터 공유 환경에 액세스하는 DB2 Connect 개인용 또는 서버 제품. 자동 클라이언트 리라우트는 DB2 Connect 및 z/OS Sysplex 시스템 사이에 사용됩니다. 자동 클라이언트 리라우트 기능은 DB2 Connect 라이센스 부여 클라이언트 및 Sysplex 사이의 원활한 장애 복구를 지원합니다. 원활한 장애 복구에 대한 자세한 정보는 DB2 정보 센터에서 자동 클라이언트 리라우트(클라이언트 측)에 대한 주제를 참조하십시오.

DB2 Connect 서버 및 해당되는 대체 서버의 경우, 로컬 데이터베이스에 대한 요구사항이 없으므로 원래 및 대체 DB2 Connect 서버 둘 다 동일한 데이터베이스 별명을 사용하여 액세스할 수 있는 방법으로 카탈로그된 목표 호스트 또는 System i 데이터베이스를 가지고 있는지 확인만 하면 됩니다.

DB2 데이터베이스 시스템이 통신 실패로부터 복구할 수 있는 기능을 갖도록 하려면, 통신 실패가 발생하기 전에 대체 서버 위치를 지정해야 합니다. UPDATE ALTERNATE SERVER FOR DATABASE 명령은 특정 데이터베이스에서 대체 서버 위치를 정의하기 위해 사용됩니다.

서버 인스턴스의 특정 데이터베이스에서 대체 서버 위치를 지정한 후, 대체 서버 위치 정보는 연결 프로세스의 일부로 IBM Data Server Client에 리턴됩니다. DB2 Connect 개인용 또는 서버 제품과 호스트 또는 System i 데이터베이스 서버 사이에 자동 클라이언트 리라우트를 사용하는 경우 리모트 서버는 자체적으로 하나 이상의 대체 주소를 제공해야 합니다. z/OS용 DB2의 경우, 데이터베이스가 Sysplex 데이터 공유 환경인 경우 여러 주소가 알려집니다. 따라서 대체 서버는 DB2 Connect에서 카탈로그해야 합니다. 클라이언트 및 서버 사이의 통신이 어떤 이유로 실패하면, IBM Data Server Client는 대체 서버 정보를 사용하여 연결을 다시 설정하려고 합니다. IBM Data Server Client는 원래 서버일 수 있는 데이터베이스 서버와, 서버에 있는 데이터베이스 디렉토리 파일에 나열된 대체 서버나 z/OS Sysplex 시스템에 의해 리턴된 서버 목록에 있는 대체 서버와의 재연결을 다시 시도합니다. 이와 같은 연결 재설정 시도의 시간 제어는 처음에는 아주 빠르게 시도하다가 점차적으로 시도 사이의 간격이 길어지면서 다양하게 됩니다.

연결되면 통신 실패 다음에 데이터베이스 연결이 재설정되었음을 표시하기 위해 SQLCODE -30108이 리턴됩니다. 호스트 이름 또는 IP 주소와 서비스 이름 또는 포트 번호가 리턴됩니다. IBM Data Server Client는 클라이언트 통신의 재설정이 원래 서버나 대체 서버에 대해 불가능한 경우 응용프로그램에 원래의 통신 실패에 대한 오류만 리턴합니다.

DB2 Connect 서버 환경에서의 대체 서버 연결성을 포함하는 다음 고려사항에도 주의해야 합니다.

- 리모트 및 로컬 클라이언트 대신 호스트 또는 System i 데이터베이스에 대한 액세스를 제공하기 위해 DB2 Connect 서버를 사용할 때 시스템 데이터베이스 디렉토리 항목에서 대체 서버 연결성 정보에 관한 혼동이 발생할 수 있습니다. 이러한 혼동을 최소화하려면 시스템 데이터베이스 디렉토리에서 동일한 호스트 또는 System i 데이터베이스를 표시하기 위한 두 항목을 카탈로그할 것을 고려하십시오. 리모트 클라이언트에 대해 하나의 항목을 카탈로그하고 로컬 클라이언트에 대해 다른 항목을 카탈로그하십시오.
- 목표 z/OS용 DB2 서버로부터 리턴되는 SYSPLEX 정보는 DB2 Connect 서버에 있는 캐시에서만 보존됩니다. 하나의 대체 서버만 디스크에 기록됩니다. 여러 개의 대체 또는 여러 개의 활성 서버가 존재하는 경우 정보는 메모리에서만 유지보수되고 프로세스가 종료될 때 손실됩니다.

일반적으로, 대체 서버를 지정한 경우 자동 클라이언트 리라우트는 통신 오류(sqlcode -30081) 또는 sqlcode -1224가 발견되는 경우에 사용 가능하게 됩니다. 그러나 고가용성 재해 복구(HADR) 환경에서는 HADR 대기 서버로부터 다시 sqlcode -1776이 리턴되는 경우에도 사용 가능하게 됩니다.

레지스트리 변수를 사용하여 자동 클라이언트 리라우트 재시도 동작 구성

기본 데이터베이스 서버가 실패할 때 DB2 자동 클라이언트 리라우트는 클라이언트 응용프로그램 요청을 보조 데이터베이스 서버로 전송합니다. 그런 다음 DB2 클라이언트 리라우트가 반복적으로 기본 데이터베이스에 다시 연결하려고 시도합니다. DB2 클라이언트 리라우트가 작성하는 최대 재연결 시도 수 및 연속적인 재연결 시도 사이의 휴면 시간을 구성할 수 있습니다.

디폴트로 자동 클라이언트 리라우트 기능은 최고 10분 동안 데이터베이스 연결을 반복적으로 재시도합니다. 그러나 다음 두 레지스트리 변수 중 하나 또는 둘 다를 사용하여 정확한 재시도 동작을 구성할 수 있습니다.

- DB2_MAX_CLIENT_CONNRETRIES: 자동 클라이언트 리라우트가 시도하는 최대 연결 재시도 수.
- DB2_CONNRETRIES_INTERVAL: 연속적인 연결 재시도 사이의 휴면 시간(초).

DB2_MAX_CLIENT_CONNRETRIES가 설정되지만
DB2_CONNRETRIES_INTERVAL이 설정되지 않은 경우,
DB2_CONNRETRIES_INTERVAL은 디폴트인 30입니다.

DB2_MAX_CLIENT_CONNRETRIES가 설정되지 않지만
DB2_CONNRETRIES_INTERVAL이 설정되는 경우,
DB2_MAX_CLIENT_CONNRETRIES는 디폴트인 10입니다.

DB2_MAX_CLIENT_CONNRETRIES 또는 DB2_CONNRETRIES_INTERVAL이 설정되지 않은 경우, 자동 클라이언트 리라우트 기능이 이전에 설명된 디폴트 동작으로 되돌아갑니다.

주:

- DB2 Universal JDBC 드라이버를 사용한 유형 4 연결의 사용자는 다음 두 데이터 소스 등록 정보를 사용하여 자동 클라이언트 리라우트를 구성해야 합니다.
  - maxRetriesForClientReroute: 서버에 대한 기본 연결이 실패하는 경우 재시도 횟수를 제한하려면 이 등록 정보를 사용하십시오. 이 등록 정보는 retryIntervalClientReroute 등록 정보도 설정되는 경우에만 사용됩니다.
  - retryIntervalForClientReroute: 다시 재시도하기 전에 휴면할 시간(초)을 지정하려면 이 등록 정보를 사용하십시오. 이 등록 정보는 maxRetriesForClientReroute 등록 정보도 설정되는 경우에만 사용됩니다.
- z/OS Sysplex용 DB2에 액세스하는 클라이언트에 대해 자동 클라이언트 리라우트가 사용 가능할 때(enableAcr이 db2dsdriver 구성 파일에서 True일 때) DB2_MAX_CLIENT_CONNRETRIES 및 DB2_CONNRETRIES_INTERVAL 설정값은 효력을 갖지 않습니다. 자세한 정보는 DB2 정보 센터에서 자동 클라이언트 리라우트(클라이언트 측)에 대한 주제를 참조하십시오.

자동 클라이언트 리라우트와 함께 클라이언트 연결 시간종료 사용

표준 DB2 데이터베이스 솔루션 구현에서, 클라이언트 응용프로그램은 성공할 때까지 또는 구성된 연결 시간종료 시간이 지날 때까지 데이터베이스 서버에 연결하려고 시도합니다. 연결이 시간종료하는 경우 통신 오류가 응용프로그램으로 리턴됩니다. DB2 자동 클라이언트 리라우트를 사용 중인 경우, DB2는 통신 오류를 생성하는 대신 해당 클라이언트 응용프로그램 연결을 대체 데이터베이스 서버로 경로 재지정합니다.

CLI/ODBC, OLE DB 및 ADO.NET 응용프로그램의 경우, 연결 시도를 종료하고 통신 시간종료를 생성하기 전에 연결 시간종료 값을 설정하여 서버에 연결을 시도할 때 클라이언트 응용프로그램이 응답에 대기하는 시간(초 단위)을 지정할 수 있습니다.

클라이언트 리라우트가 사용 가능한 경우, 연결 시간종료 값을 서버에 연결하는 데 걸리는 최대 시간보다 크거나 같은 값으로 설정해야 합니다. 그렇지 않으면 연결이 시간종료하고 클라이언트 리라우트에 의해 대체 서버로 리라우트될 수 있습니다. 예를 들어 일반적인 날에는 서버에 연결하는 데 약 10초가 걸리고 바쁜 날에는 약 20초가 걸리는 경우 연결 시간종료 값을 최소한 20초로 설정해야 합니다.

클라이언트 연결 분배기 기술에 대한 자동 클라이언트 리라우트 구성

WebSphere EdgeServer 같은 분배기 또는 디스패처 기술은 기본 데이터베이스 서버가 실패하는 경우 클라이언트 응용프로그램 재연결 요청을 정의된 시스템 세트로 분배합니다. DB2 자동 클라이언트 리라우트와 함께 분배기 기술을 사용 중인 경우 분배기 자체를 DB2 자동 클라이언트 리라우트에 대한 대체 서버로 식별해야 합니다.

다음과 비슷한 환경에서 분배기 기술을 사용 중일 수 있습니다.

클라이언트 —> 분배기 기술 —> (DB2 Connect 서버 1 또는 DB2 Connect 서버 2) —> DB2 z/OS

각 부분에 대한 설명은 다음과 같습니다.

- 분배기 기술 구성요소는 DThostname의 TCP/IP 호스트 이름을 갖습니다.
- DB2 Connect 서버 1은 GWYhostname1의 TCP/IP 호스트 이름을 갖습니다.
- DB2 Connect 서버 2는 GWYhostname2의 TCP/IP 호스트 이름을 갖습니다.
- DB2 z/OS 서버는 zOShostname의 TCP/IP 호스트 이름을 갖습니다.

클라이언트는 분배기 기술을 이용하여 DB2 Connect 서버 중 하나에 액세스하기 위해 **DThostname**을 사용하여 카탈로그됩니다. 중간 분배기 기술이 **GWYhostname1** 또는 **GWYhostname2**를 사용할 것을 결정합니다. 결정한 후에 클라이언트는 이러한 두 DB2 Connect 게이트웨이 중 하나에 직접 소켓 연결됩니다. 소켓 연결성이 선택된 DB2 Connect 서버에 설정된 후에, 일반 클라이언트 대 DB2 Connect 서버 대 DB2 z/OS 연결성을 갖습니다.

예를 들어 분배기가 **GWYhostname2**를 선택한다고 가정하십시오. 이것은 다음 환경을 생성합니다.

클라이언트 —> DB2 Connect 서버 2 —> DB2 z/OS

분배기는 통신 실패가 있는 경우 어떤 연결도 재시도하지 않습니다. 그러한 환경에 있는 데이터베이스에 대해 자동 클라이언트 리라우트 기능을 사용하려는 경우, DB2 Connect 서버에 있는 연관된 데이터베이스에 대한 대체 서버(DB2 Connect 서버 1 또는 DB2 Connect 서버 2)가 분배기(DThostname)가 되도록 설정되어야 합니다. 그런 다음 DB2 Connect 서버 1이 어떤 이유로 잠기는 경우, 자동 클라이언트 리라우트가 트리거되고 클라이언트 연결은 기본 및 대체 서버 둘 다로서의 분배기를 사용하여 재시도됩니다. 이 옵션을 사용하면 분배기 기능을 DB2 자동 클라이언트 리라우트 기능과 결합하고 유지보수할 수 있습니다. 대체 서버를 분배기 호스트 이름이 아닌 호스트로 설정해도 클라이언트에 자동 클라이언트 리라우트 기능을 제공합니다. 그러나 클라이언트는 정의된 대체 서버로의 직접 연결을 설정하고 분배기 기술을 생략하며, 이는 분배기 및 분배기로 인한 값을 제거합니다.

자동 클라이언트 리라우트 기능이 다음 SQL 코드를 가로챕니다.

- sqlcode -20157
- sqlcode -1768(이유 코드 = 7)

주: ″TCP Keepalive″ 운영 체제 구성 매개변수의 설정이 너무 높게 설정되면 소켓 실패가 클라이언트 리라우트에 시기 적절하게 알려지지 않을 수 있습니다. (이 구성 매개변수의 이름은 플랫폼에 따라 다름을 주의하십시오.)

자동 클라이언트 리라우트를 위한 대체 서버 식별

DB2 서버나 DB2 Connect 서버가 손상될 때마다, 해당 서버에 연결되는 각 클라이언트는 응용프로그램 오류를 생성하는 연결 종료 통신 오류를 수신합니다. 사용 가능성이 중요한 경우에는 중복 설정이나 서버를 대기 노드로 장애 복구하는 기능을 구현해야 합니다. 어느 경우에나 DB2 클라이언트 코드는 장애 복구 노드에서 실행 중일 수 있는 원래 서버(IP 주소도 장애 복구함) 또는 새 서버에 연결을 재설정하려고 시도합니다.

신규 또는 대체 서버를 정의하려면 다음을 수행하십시오.

UPDATE ALTERNATE SERVER FOR DATABASE 또는 **UPDATE ALTERNATE SERVER FOR LDAP DATABASE** 명령을 사용하십시오.
이 명령은 시스템 데이터베이스 디렉토리에서 데이터베이스 별명에 대한 대체 서버 정보를 갱신합니다.

JDBC 및 SQLJ용 IBM Data Server Driver 사용 시 클라이언트 리라우트 설정

서버가 손상될 때마다 해당 서버에 연결된 각 클라이언트는 통신 오류를 수신하고 이로 인해 연결이 종료되고 결국 응용프로그램 오류가 발생됩니다. 사용 가능성이 중요한 경우 중복 설정이나 장애 복구 지원이 있어야 합니다. 장애 복구는 다른 서버가 실패하는 경우 작업을 인계할 서버의 기능입니다. 연결이 종료되면, IBM Data Server Driver for JDBC and SQLJ 클라이언트는 원래 서버(장애 복구 노드에서 실행 중일 수 있는)나 새 서버와의 연결을 재설정하려고 합니다. 연결이 재설정되면, 응용프로그램은 트랜잭션 실패를 알리는 SQLException을 수신하지만 응용프로그램은 다음 트랜잭션으로 계속할 수 있습니다.

세부사항 정보는 아래의 관련 개념 절에서 IBM Data Server Driver for JDBC and SQLJ에 대한 클라이언트 리라우트 지원에 대한 주제를 참조하십시오.

자동 클라이언트 리라우트 제한사항

고가용성 DB2 데이터베이스 솔루션을 설계할 때 DB2 데이터베이스 클라이언트 리라우트 제한사항을 고려하십시오.

다음은 DB2 데이터베이스 자동 클라이언트 리라우트 기능의 제한사항 목록입니다.

- 자동 클라이언트 리라우트는 DB2 데이터베이스 서버 또는 DB2 Connect 서버와의 연결에 사용된 통신 프로토콜이 TCP/IP인 경우에만 지원됩니다. 이는 연결에서 TCP/IP가 아닌 다른 프로토콜을 사용하는 경우 자동 클라이언트 리라우트 기능을 사용할 수 없음을 의미합니다. DB2 데이터베이스가 루프백되도록 설정되어도, 자동 클라이언트 리라우트 기능을 수용하려면 TCP/IP 통신 프로토콜을 사용해야 합니다.
- DB2 Connect 개인용 또는 서버 제품과 호스트 또는 System i 데이터베이스 서버 사이에 자동 리라우트를 사용하는 경우, 사용자가 다음 상황에 있으면 연관된 내포 사항이 수반됩니다.
  - 리모트 및 로컬 클라이언트 대신 호스트 또는 System i 데이터베이스에 대한 액세스를 제공하기 위해 DB2 Connect 서버를 사용할 때 시스템 데이터베이스 디렉토리 항목에서 대체 서버 연결성 정보에 관한 혼동이 발생할 수 있습니다. 이러한 혼동을 최소화하려면 시스템 데이터베이스 디렉토리에서 동일한 호스트 또는 System i 데이터베이스를 표시하기 위한 두 항목을 카탈로그할 것을 고려하십시오. 리모트 클라이언트에 대해 하나의 항목을 카탈로그하고 로컬 클라이언트에 대해 다른 항목을 카탈로그하십시오.
  - 목표 z/OS용 DB2 서버로부터 리턴되는 SYSPLEX 정보는 DB2 Connect 서버에 있는 캐시에서만 보존됩니다. 하나의 대체 서버만 디스크에 기록됩니다. 여러 개의 대체 또는 여러 개의 활성 서버가 존재하는 경우 정보는 메모리에서만 유지보수되고 프로세스가 종료될 때 손실됩니다.

- 대체 서버 위치와의 연결이 재설정되면, 동일한 데이터베이스 별명에 대한 새 연결이 대체 서버 위치에 연결됩니다. 원래 위치의 문제점이 수정된 경우 원래 위치에 대해 새 연결을 설정하려는 경우, 선택할 한 쌍의 옵션이 있습니다.
  - 대체 서버를 오프라인으로 가져와서 연결이 원래 서버로 장애 복구되도록 해야 합니다. (이때 원래 서버는 대체 서버의 대체 위치로 설정되도록 UPDATE ALTERNATE SERVER 명령을 사용하여 카탈로그된 것으로 가정합니다.)
  - 새 연결에 사용될 새 데이터베이스 별명을 카탈로그할 수 있습니다.
  - 데이터베이스 항목을 카탈로그 해제하고 다시 카탈로그할 수 있습니다.
- Linux, UNIX 및 Windows용 DB2 데이터베이스는 클라이언트와 서버 둘 다 자동 클라이언트 리라우트 기능을 지원하는 경우 클라이언트와 서버 둘 다에 대해 이 기능을 지원합니다. 다른 DB2 데이터베이스 제품군은 현재 이 기능을 지원하지 않습니다.
- 자동 클라이언트 리라우트 기능의 동작과 z/OS용 DB2 sysplex 환경에서의 자동 클라이언트 리라우트 동작은 어느 정도 다릅니다. 특히 다음 사항이 다릅니다.
  - 자동 클라이언트 리라우트 기능에서는 기본 서버에 단일 대체 서버를 지정해야 합니다. 이는 기본 서버에서 발행되는 UPDATE ALTERNATE SERVER FOR DATABASE 또는 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 명령을 사용하여 수행됩니다. 이 명령은 같은 클라이언트에 있는 다른 응용프로그램이 이 정보에 액세스할 수 있도록 로컬 데이터베이스 디렉토리를 대체 서버 정보로 갱신합니다. 반대로, z/OS용 DB2에 사용되는 데이터 공유 sysplex는 클라이언트가 연결할 수 있는 하나 이상의 서버 목록을 메모리에서 유지합니다. 통신 실패가 발생하면 클라이언트는 이 서버 목록을 사용하여 적절한 대체 서버의 위치를 판별합니다.
  - 자동 클라이언트 리라우트 기능의 경우, 서버는 특수 레지스터 설정이 변경될 때마다 최근의 특수 레지스터 설정을 클라이언트에 알립니다. 이로서 클라이언트는 가능한 한 리라우트 발생 후 런타임 환경을 재설정할 수 있습니다. 반대로, z/OS용 DB2에 사용되는 Sysplex는 커미트 경계에서 클라이언트에 특수 레지스터 설정을 리턴하므로, 리라우트된 작업 단위(UOW) 내에서 변경된 특수 레지스터를 재생해야 합니다. 다른 것은 모두 자동으로 재생됩니다.

  DB2® Universal Database™ 버전 8 FixPak 7 현재, 전체 자동 클라이언트 리라우트 지원은 Linux, UNIX 또는 Windows 클라이언트와 Linux, UNIX 또는 Windows 서버 사이에서만 사용 가능합니다. Linux, UNIX 또는 Windows 클라이언트와 z/OS용 DB2 Sysplex 서버(지원되는 버전) 사이에는 사용할 수 없습니다. 리라우트 성능만 지원됩니다.
- 대체 호스트 서버에 설치된 DB2 데이터베이스 서버는 원래 호스트 서버에 설치된 DB2 데이터베이스 인스턴스와 비교될 때 동일한 버전(상위 FixPak을 가지고 있을 수 있음)에 있어야 합니다.

- 클라이언트 머신에서 데이터베이스 디렉토리를 갱신할 수 있는 권한을 가지고 있는지 여부에 관계없이, 대체 서버 정보는 항상 메모리에 보존됩니다. 즉, 데이터베이스 디렉토리를 갱신할 수 있는 권한이 없는 경우(또는 읽기 전용 데이터베이스 디렉토리여서), 다른 응용프로그램은 대체 서버를 판별하여 사용할 수 없습니다. 응용프로그램 사이에 메모리를 공유하지 않기 때문입니다.
- 모든 대체 위치에 동일한 인증이 적용됩니다. 이는 대체 위치에 원래 위치와 다른 인증 유형이 있는 경우 클라이언트가 데이터베이스 연결을 재설정할 수 없음을 의미합니다.
- 통신 실패가 있으면, 페더레이티드 처리 및 특수 레지스터에 대한 모든 세션 자원(예: 전역 임시 테이블, 신원(ID), 시퀀스, 커서, 서버 옵션(SET SERVER OPTION))이 손실됩니다. 응용프로그램은 작업 처리를 계속하려면 세션 자원을 다시 설정해야 합니다. 연결이 재설정된 후에는 특수 레지스터 명령문을 실행하지 않아도 됩니다. DB2 데이터베이스가 통신 오류 이전에 발행된 특수 레지스터 명령문을 재생하기 때문입니다. 그러나 일부 특수 레지스터는 재생되지 않습니다. 이러한 특수 레지스터는 다음과 같습니다.
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE
  - SET SESSION AUTHORIZATION
  - SET TRANSFORM GROUP

  DB2 Connect에 문제점이 있을 때, 데이터 서버의 DB2 Connect 제품에 특정한 제한된 특수 레지스터 목록을 참조해야 합니다.
- 통신 실패 후 연결이 재설정되고 클라이언트가 CLI, JCC Type 2 또는 Type 4 드라이버를 사용하는 경우, 원래 서버에 대해 준비된 SQL 및 XQuery 문은 내재적으로 새 서버에 대해 다시 준비됩니다. 그러나 임베디드(embedded) SQL 루틴(예: SQC 또는 SQX 응용프로그램)은 새 서버에 대해 다시 준비되지 않습니다.
- 클라이언트 리라우트가 사용 가능한 데이터베이스 별명에 대해 고가용성 재해 복구(HADR) 명령을 실행하지 마십시오. HADR 명령은 데이터베이스 별명을 사용하여 목표 데이터베이스를 식별하기 위해 구현됩니다. 결국, 목표 데이터베이스에 정의된 대체 데이터베이스가 있는 경우 HADR 명령이 실제로 작동하는 데이터베이스를 판별하는 것은 어렵습니다. 클라이언트가 클라이언트 리라우트 사용 가능 별명을 사용하여 연결해야 할 수 있지만, 특정 데이터베이스에서는 HADR 명령을 적용해야 합니다. 이를 조정하기 위해, 기본 및 대기 데이터베이스에 특정한 별명을 정의하고 해당 별명에 대해서만 HADR 명령을 실행할 수 있습니다.

자동 클라이언트 리라우트를 구현하기 위한 대체 방법은 DNS 항목을 사용하여 DNS 항목에 대한 대체 IP 주소를 지정하는 것입니다. DNS 항목에서 두 번째 IP 주소(대체 서버 위치)를 지정합니다. 클라이언트는 대체 서버에 대해 알지 못하지만 연결 시 DB2 데이터베이스 시스템이 DNS 항목의 IP 주소 사이에 교체합니다.

DB2 결함 모니터 레지스트리 파일

결함 모니터 레지스트리 파일은 결함 모니터 디먼이 시작될 때 각각의 실제 머신에서 모든 DB2 데이터베이스 관리 프로그램 인스턴스에 대해 작성됩니다. 이 파일의 키워드와 값은 결함 모니터의 동작을 지정합니다.

결함 모니터 레지스트리 파일은 /sqllib/ 디렉토리에서 찾을 수 있으며 fm.<machine_name>.reg라고 합니다. 이 파일은 db2fm 명령을 사용하여 변경할 수 있습니다.

결함 모니터 레지스트리 파일이 없는 경우 디폴트값이 사용됩니다.

다음은 결함 모니터 레지스트리 파일 컨텐츠의 예입니다.

```
FM_ON = no
FM_ACTIVE = yes
START_TIMEOUT = 600
STOP_TIMEOUT = 600
STATUS_TIMEOUT = 20
STATUS_INTERVAL = 20
RESTART_RETRIES = 3
ACTION_RETRIES = 3
NOTIFY_ADDRESS = <instance_name>@<machine_name>
```

결함 모니터 레지스트리 파일 키워드

FM_ON

결함 모니터를 시작해야 하는지 여부를 지정합니다. 값을 NO로 설정하면, 결함 모니터 디먼이 시작되지 않거나 이미 시작된 경우에는 해제됩니다. 디폴트값은 NO입니다.

FM_ACTIVE

결함 모니터가 활성 상태인지 여부를 지정합니다. 결함 모니터는 FM_ON 및 FM_ACTIVE 둘 다가 YES로 설정된 경우에만 조치를 수행합니다. FM_ON이 YES로 설정되고 FM_ACTIVE가 NO로 설정된 경우 결함 모니터 디먼은 시작되지만 활성 상태가 되지 않습니다. 즉, DB2가 종료된 경우 다시 온라인으로 가져오려고 하지 않습니다. 디폴트값은 YES입니다.

START_TIMEOUT

결함 모니터가 모니터 중인 서비스를 시작해야 하는 시간 양을 지정합니다. 디폴트값은 600초입니다.

STOP_TIMEOUT

결함 모니터가 모니터 중인 서비스를 중단시켜야 하는 시간 양을 지정합니다. 디폴트값은 600초입니다.

STATUS_TIMEOUT

결함 모니터가 모니터 중인 서비스 상태를 가져와야 하는 시간 양을 지정합니다. 디폴트값은 20초입니다.

STATUS_INTERVAL

모니터 중인 서비스의 상태를 확보하기 위한 두 연속 호출 사이의 최소 시간을 지정합니다. 디폴트값은 20초입니다.

RESTART_RETRIES

결함 모니터가 시도 실패 후 모니터 중인 서비스의 상태를 확보하기 위해 시도할 횟수를 지정합니다. 이 횟수에 도달하면 결함 모니터는 서비스를 다시 온라인으로 가져오기 위한 조치를 수행합니다. 디폴트값은 3입니다.

ACTION_RETRIES

결함 모니터가 서비스를 다시 온라인으로 가져오기 위해 시도할 횟수를 지정합니다. 디폴트값은 3입니다.

NOTIFY_ADDRESS

결함 모니터가 통지 메시지를 보낼 전자 우편 주소를 지정합니다. 디폴트는 <instance_name>@<machine_name>입니다.

db2fm 명령을 사용하여 DB2 결함 모니터 구성

db2fm 명령을 사용하여 DB2 결함 모니터 레지스트리 파일을 변경할 수 있습니다.

다음은 db2fm 명령을 사용하여 결함 모니터 레지스트리 파일을 갱신하는 예입니다.

예 1: START_TIMEOUT 갱신

인스턴스 DB2INST1에 대해 START_TIMEOUT 값을 100초로 갱신하려면 DB2 데이터베이스 명령 창에서 다음 명령을 입력하십시오.

```
db2fm -i db2inst1 -T 100
```

예 2: STOP_TIMEOUT 갱신

인스턴스 DB2INST1에 대해 STOP_TIMEOUT 값을 200초로 갱신하려면 다음 명령을 입력하십시오.

```
db2fm -i db2inst1 -T /200
```

예 3: START_TIMEOUT 및 STOP_TIMEOUT 갱신

인스턴스 DB2INST1에 대해 START_TIMEOUT 값을 100초로,
STOP_TIMEOUT 값을 200초로 갱신하려면 다음 명령을 입력하십시오.

```
db2fm -i db2inst1 -T 100/200
```

예 4: 결함 모니터링 설정

인스턴스 DB2INST1에 대해 결함 모니터링을 설정하려면 다음 명령을 입력하십시오.

```
db2fm -i db2inst1 -f yes
```

예 5: 결함 모니터링 해제

인스턴스 DB2INST1에 대해 결함 모니터링을 해제하려면 다음 명령을 입력하십시오.

```
db2fm -i db2inst1 -f no
```

결함 모니터가 더 이상 DB2INST1에 대해 실행되지 않음을 확인하려면, UNIX 시스템에서 다음 명령을 입력하십시오.

```
ps -ef|grep -i fm
```

Linux에서는 다음 명령을 입력하십시오.

```
ps auxw|grep -i fm
```

db2fmd 및 DB2INST1을 표시하는 항목은 결함 모니터가 계속 해당 인스턴스에서 실행 중임을 표시합니다. 결함 모니터를 해제하려면 인스턴스 소유자로서 다음 명령을 입력하십시오.

```
db2fm -i db2inst1 -D
```

db2fmc 및 시스템 명령을 사용하여 DB2 결함 모니터 구성

DB2 결함 모니터는 DB2 FMCU(Fault Monitor Controller Utility) 명령 db2fmcu 또는 시스템 명령을 사용하여 구성할 수 있습니다.

다음은 db2fmcu 및 시스템 명령을 사용하여 결함 모니터를 구성하는 예입니다.

예 1: FMC가 시작되지 않도록 금지

DB2 FMCU(Fault Monitor Controller Utility)를 사용하여 FMC가 시작되지 않도록 금지할 수 있습니다. FMCU는 시스템의 inittab 파일에 액세스하므로 루트로 실행해야 합니다. FMC가 실행되지 않도록 차단하려면 다음 명령을 루트로 입력하십시오.

```
db2fmcu -d
```

주: DB2 Data Server Fixpack을 적용하는 경우 FMC를 포함하도록 inittab를 다시 구성하기 위해 재설정됩니다. Fixpack을 적용한 후 FMC가 시작되지 않도록 하려면 위의 명령을 다시 발행해야 합니다.

예 2: 시작할 FMC 포함

`db2fmcu -d` 명령을 되돌리고 FMC를 포함하도록 inittab를 재구성하려면 다음 명령을 입력하십시오.

```
db2fmcu -u -p  <fullpath>
```

여기서 <fullpath>는 db2fmcd 오브젝트의 전체 경로입니다(예: /opt/IBM/db2/bin/db2fmcd).

예 3: DB2 데이터베이스 관리 프로그램 인스턴스 자동 시작

또한 시스템이 처음 시동될 때 FMC가 자동으로 인스턴스를 시작하도록 할 수도 있습니다. 인스턴스 DB2INST1에 대해 이 기능이 가능하도록 하려면 다음 명령을 입력하십시오.

```
db2iauto -on db2inst1
```

예 4: 인스턴스 자동 시작 사용 안함

자동 시작 동작을 해제하려면 다음 명령을 입력하십시오.

```
db2iauto -off db2inst1
```

예 5: 결함 모니터 프로세스가 시작되지 않도록 금지

인스턴스의 전역 레지스트리 레코드에서 필드를 변경하여 결함 모니터 프로세스가 시스템에서 특정 인스턴스에 대해 시작하지 않도록 금지할 수도 있습니다. 인스턴스 DB2INST1에 대해 결함 모니터가 사용되지 않도록 전역 레지스트리 필드를 변경하려면 루트로 다음 명령을 입력하십시오.

```
db2greg -updinstrec instancename=db2inst1!startatboot=0
```

이 명령을 되돌리고 인스턴스 DB2INST1에 대해 결함 모니터가 다시 사용 가능하도록 하려면 루트로 다음 명령을 입력하십시오.

```
db2greg -updinstrec instancename=db2inst1!startatboot=1
```

고가용성 재해 복구(HADR) 초기화

DB2 고가용성 재해 복구(HADR)에 대해 기본 및 대기 데이터베이스를 설정하고 초기화하려면 다음 프로시저를 사용하십시오.

HADR은 명령행 처리기(CLP), 제어 센터의 고가용성 재해 복구(HADR) 설정 마법사를 통해 또는 db2HADRStart API를 호출하여 초기화될 수 있습니다.

CLP를 사용하여 처음으로 시스템에 대한 HADR을 초기화하려면 다음을 수행하십시오.

1. 각 HADR 데이터베이스에 대해 호스트 이름, 호스트 IP 주소 및 서비스 이름이나 포트 번호를 판별하십시오.

   호스트에 다중 네트워크 인터페이스가 있는 경우, HADR 호스트 이름이나 IP 주소가 의도한 것에 맵핑되는지 확인하십시오. 각 보호 데이터베이스에 대한 /etc/services에서 개별 HADR 포트를 할당해야 합니다. 이들은 인스턴스에 할당된 포트와 동일해야 합니다. 호스트 이름은 하나의 IP 주소에만 맵핑할 수 있습니다.

   주: 기본 및 대기 데이터베이스에 대한 인스턴스 이름이 동일할 필요는 없습니다.
2. 기본이 될 기존 데이터베이스에 따라서 백업 이미지를 리스토어하거나 분할 미러를 초기화하여 대기 데이터베이스를 작성하십시오.

다음 예에서 BACKUP DATABASE 및 RESTORE DATABASE 명령이 데이터베이스 SOCKS를 대기 데이터베이스로 초기화하는 데 사용됩니다. 이 경우, NFS 마운트 파일 시스템은 두 사이트 모두에서 액세스 가능합니다.

기본 데이터베이스에서 다음 명령을 실행하십시오.

```
backup db socks to /nfs1/backups/db2/socks
```

대기 데이터베이스에서 다음 명령을 실행하십시오.

```
restore db socks from /nfs1/backups/db2/socks replace history file
```

다음 예는 db2inidb 유틸리티를 사용하여 기본 데이터베이스의 분할 미러로 대기 데이터베이스를 초기화하는 방법을 보여줍니다. 이 프로시저는 위에서 보여주는 백업 및 리스토어 프로시저의 대안입니다.

대기 데이터베이스에서 다음 명령을 실행하십시오.

```
db2inidb socks as standby
```

주:

a. 기본 및 대기 데이터베이스에 대한 데이터베이스 이름은 같아야 합니다.

b. 리스토어 작업 또는 분할 미러 초기화 후에 대기 데이터베이스에서 ROLLFORWARD DATABASE 명령을 실행하지 않는 것이 좋습니다. 롤 포워드 조작 사용의 결과는 대기 데이터베이스에서 HADR을 사용한 로그 재생과는 약간 다를 수 있습니다. 데이터베이스가 동일하지 않은 경우 AS STANDBY 옵션을 사용한 START HADR 명령 실행은 실패합니다.

c. RESTORE DATABASE 명령을 사용할 때는 REPLACE HISTORY FILE 옵션을 사용하는 것이 바람직합니다.

d. RESTORE DATABASE 명령을 사용하여 대기 데이터베이스를 작성할 때 대기가 롤 포워드 모드에 남아있는지 확인해야 합니다. 이것은 COMPLETE 옵션이나 STOP 옵션을 사용하여 ROLLFORWARD DATABASE 명령을 실행할 수 없음을 의미합니다. AS STANDBY 옵션을 사용한 START HADR 명령이 롤 포워드가 중지된 후 데이터베이스에 대해 시도되는 경우 오류가 리턴됩니다.

e. 대기 데이터베이스를 설정할 때 RESTORE DATABASE 명령 옵션 TABLESPACE, INTO, REDIRECT 및 WITHOUT ROLLING FORWARD는 피해야 합니다.

f. db2inidb 유틸리티를 사용하여 대기 데이터베이스를 설정할 때 SNAPSHOT 또는 MIRROR 옵션을 사용하지 마십시오. RELOCATE USING 옵션을 지정하여 인스턴스 이름, 로그 경로 및 데이터베이스 경로의 구성 속성 중 하나 이상을 변경할 수 있습니다. 그러나 데이터베이스 이름이나 테이블 스페이스 컨테이너 경로는 변경해서는 안됩니다.

3. 기본 및 대기 데이터베이스에서 HADR 구성 매개변수를 설정하십시오.

   주: 대기 데이터베이스가 작성된 후 다음 구성 매개변수를 설정하는 것이 매우 중요합니다.

   - HADR_LOCAL_HOST
   - HADR_LOCAL_SVC
   - HADR_REMOTE_HOST
   - HADR_REMOTE_SVC
   - HADR_REMOTE_INST

   대기 데이터베이스를 작성하기 전에 설정되는 경우 대기 데이터베이스의 설정값이 기본 데이터베이스에서 설정된 값을 반영합니다.

4. 다음 예에서와 같이 대기 인스턴스에 연결하고 대기 데이터베이스에서 HADR을 시작하십시오.

   ```
   START HADR ON DB SOCKS AS STANDBY
   ```

   주: 대개 대기 데이터베이스가 첫 번째로 시작됩니다. 기본 데이터베이스를 처음 시작하는 경우, 이 시작 프로시저는 대기 데이터베이스가 HADR_TIMEOUT 데이터베이스 구성 매개변수에 의해 지정되는 시간 안에 시작되지 않는 경우 실패합니다.

5. 다음 예에서와 같이 기본 인스턴스에 연결하고 기본 데이터베이스에서 HADR을 시작하십시오.

   ```
   START HADR ON DB SOCKS AS PRIMARY
   ```

6. HADR이 이제 기본 및 대기 데이터베이스에서 시작됩니다.

   고가용성 재해 복구(HADR) 데이터베이스 설정 마법사를 열려면 다음을 수행하십시오.

   a. 제어 센터에서 HADR을 구성하려는 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 고가용성 재해 복구 → 설정을 누르십시오. 고가용성 재해 복구 데이터베이스 설정 마법사가 열립니다.

   제어 센터의 문맥 도움말 기능을 통해 추가 정보가 제공됩니다.

   주: 고가용성 재해 복구 데이터베이스 설정 마법사에서 HADR을 시작하거나, 단지 마법사를 사용하여 HADR을 초기화한 후 다른 시간에 시작할 수 있습니다. HADR 시작 창을 열려면 다음을 수행하십시오.

a. 제어 센터에서 HADR을 관리하려는 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 고가용성 재해 복구 → 관리를 누르십시오. 고가용성 재해 복구 관리 창이 열립니다.

b. HADR 시작을 누르십시오. HADR 시작 창이 열립니다.

자동 클라이언트 리라우트 및 고가용성 재해 복구(HADR) 구성

고가용성 재해 복구(HADR) 기능과 함께 자동 클라이언트 리라우트 기능을 사용하여 클라이언트 응용프로그램 요청을 실패한 데이터베이스 서버에서 대기 데이터베이스 서버로 전송할 수 있습니다.

제한사항

- 리라우팅은 대체 데이터베이스 위치가 서버에 지정된 경우에만 가능합니다.
- 자동 클라이언트 리라우트는 TCP/IP 프로토콜에만 지원됩니다.

구성 세부사항

- 자동 클라이언트 리라우트가 사용 가능하도록 하려면 UPDATE ALTERNATE SERVER FOR DATABASE 명령을 사용하십시오.
- 클라이언트 리라우트는 제어 센터에서 고가용성 재해 복구(HADR) 설치 마법사를 사용하여 HADR을 설치할 경우 디폴트로 사용 가능합니다.
- 자동 클라이언트 리라우트는 HADR_REMOTE_HOST 및 HADR_REMOTE_SVC 데이터베이스 구성 매개변수를 사용하지 않습니다.
- 대체 호스트 위치는 서버의 시스템 데이터베이스 디렉토리 파일에 저장되어 있습니다.
- 자동 클라이언트 리라우트를 사용할 수 없는 경우, 클라이언트 응용프로그램이 SQL30081 오류 메시지를 수신하게 되고 서버와 연결하려는 시도가 더 이상 이루어지지 않습니다.

UPDATE ALTERNATE SERVER FOR DATABASE 명령을 사용하여 HADR를 사용하는 자동 클라이언트 리라우트 설정

사용자 시스템이 다음과 같이 설정되어 있습니다.

- 데이터베이스 MUSIC이 호스트 HORNET에 위치되는 것으로 카탈로그된 클라이언트가 있습니다.
- 데이터베이스 MUSIC이 기본 데이터베이스이고 해당되는 대기 데이터베이스(역시 MUSIC)가 포트 번호 456의 호스트 MONTERO에 있습니다. 이는 SVCENAME 구성 매개변수에 의해 지정됩니다.

자동 클라이언트 리라우트를 사용 가능하도록 하려면, 호스트 HORNET에서 데이터베이스 MUSIC에 대하 대체 서버를 갱신하십시오.

```
db2 update alternate server for database music using hostname montero port 456
```

이 명령이 발행되고 나면, 클라이언트는 HORNET에 연결되어 대체 서버 정보를 확보해야 합니다. 그러면 호스트 HORNET에서 클라이언트와 데이터베이스 사이에 오류가 발생하는 경우 클라이언트는 먼저 호스트 HORNET에서 데이터베이스 MUSIC에 다시 연결하려고 합니다. 실패하면 클라이언트는 호스트 MONTERO에서 대기 데이터베이스 MUSIC과의 연결을 설정하려고 합니다.

인덱스 로깅 및 고가용성 재해 복구(HADR)

DB2 고가용성 재해 복구(HADR) 데이터베이스에 대해 데이터베이스 구성 매개변수 LOGINDEXBUILD 및 INDEXREC를 설정할 것을 고려해야 합니다.

LOGINDEXBUILD 데이터베이스 구성 매개변수 사용

권장사항: HADR 데이터베이스의 경우, LOGINDEXBUILD 데이터베이스 구성 매개변수를 ON으로 설정하여 전체 정보가 인덱스 작성, 재작성 및 재구성에 대해 로그되도록 하십시오. 이는 인덱스 빌드가 기본 시스템에서 더 오래 걸릴 수 있고 더 많은 로그 스페이스가 필요함을 의미하지만, 인덱스는 HADR 로그 재생 중에 대기 시스템에서 재빌드되고 장애 복구가 발생할 때 사용 가능하게 됩니다. 기본 시스템에서의 인덱스 빌드가 로그되지 않고 장애 복구가 발생하는 경우, 장애 복구 완료 후에 남아 있는 유효하지 않은 인덱스가 완료되고 액세스하기 전에 재빌드됩니다. 인덱스가 재작성되는 동안 응용프로그램이 인덱스에 액세스할 수 없습니다.

주: LOG INDEX BUILD 테이블 속성이 디폴트값 NULL로 설정되는 경우, DB2는 LOGINDEXBUILD 데이터베이스 구성 매개변수에 대해 지정된 값을 사용합니다. LOG INDEX BUILD 테이블 속성이 ON 또는 OFF로 설정되는 경우, LOGINDEXBUILD 데이터베이스 구성 매개변수에 대해 지정된 값은 무시됩니다.

다음 이유 중 하나로 하나 이상의 테이블에서 LOG INDEX BUILD 테이블 속성을 OFF로 설정할 것을 선택할 수 있습니다.

- 인덱스 빌드 로깅을 지원하기에는 사용 중인 로그 스페이스가 충분하지 않습니다.
- 인덱스 데이터가 아주 크고 테이블에 자주 액세스하지 않습니다. 따라서 인계 조작 끝에서 인덱스를 재작성할 수 있습니다. 이 경우 INDEXREC 구성 매개변수를 RESTART로 설정하십시오. 테이블에 자주 액세스하지 않으므로, 이와 같이 설정하면 시스템이 인계 조작 후에 처음 테이블에 액세스할 때까지 기다리는 대신 인계 조작 끝에서 인덱스를 재작성합니다.

하나 이상의 테이블에서 LOG INDEX BUILD 테이블 속성을 OFF로 설정하는 경우, 해당 테이블에 대한 인덱스 빌드 조작은 인계 조작 발생 시 인덱스가 재작성되도록 할 수 있습니다. 마찬가지로, LOG INDEX BUILD 테이블 속성을 해당 디폴트값 NULL로 설정하고 LOGINDEXBUILD 데이터베이스 구성 매개변수를 OFF로 설정하는 경

우, 테이블에 대한 인덱스 빌드 조작은 인계 조작 발생 시 해당 테이블의 인덱스가 재작성되도록 할 수 있습니다. 다음 조치 중 하나를 수행하여 인덱스가 재작성되지 않도록 할 수 있습니다.

- 모든 유효하지 않은 인덱스가 새 기본 데이터베이스에서 재작성되면 데이터베이스 백업을 취하여 대기 데이터베이스에 적용하십시오. 그 결과, 대기 데이터베이스는 기본 데이터베이스에서 유효하지 않은 인덱스 재작성에 사용되는 로그를 적용하지 않아도 됩니다. 그 인덱스는 대기 데이터베이스에서 재빌드가 필요한 것으로 표시합니다.
- LOG INDEX BUILD 테이블 속성을 ON으로 설정하거나, LOG INDEX BUILD 테이블 속성을 NULL로 설정하고 대기 데이터베이스에서 LOGINDEXBUILD 구성 매개변수를 ON으로 설정하여 인덱스 재작성이 로그되도록 하십시오.

INDEXREC 데이터베이스 구성 매개변수 사용

권장사항: 기본 및 대기 데이터베이스 둘 다에서 INDEXREC 데이터베이스 구성 매개변수를 RESTART(디폴트)로 설정하십시오. 그러면 인계 조작 완료 후에 유효하지 않은 인덱스가 재빌드됩니다. 인덱스 재빌드가 로그되지 않은 경우 이 설정을 사용하면 DB2가 유효하지 않은 인덱스를 점검하여 재빌드할 수 있습니다. 이 프로세스는 백그라운드에서 발생하며 데이터베이스는 인계 조작이 성공적으로 완료된 후에 액세스할 수 있습니다.

인덱스가 인덱스 백그라운드 재작성 프로세스에 의해 재빌드되기 전에 유효하지 않은 인덱스를 가지고 있는 테이블에 트랜잭션이 액세스하는 경우, 액세스하는 첫 번째 트랜잭션에 의해 유효하지 않은 인덱스가 재빌드됩니다.

고가용성 재해 복구(HADR)에 대한 데이터베이스 구성

데이터베이스 구성 매개변수를 사용하여 DB2 고가용성 재해 복구(HADR)에 대해 최적의 성능을 달성할 수 있습니다.

DB2 고가용성 재해 복구(HADR)에 대해 최적의 성능을 달성하려면, 데이터베이스 구성이 다음 요구사항을 충족하는지 확인하십시오.

권장사항: 가능하면, 데이터베이스 구성 매개변수와 데이터베이스 관리 프로그램 구성 매개변수는 기본 및 대기 데이터베이스가 있는 시스템에 동일해야 합니다. 구성 매개변수가 대기 데이터베이스에서 적절하게 설정되지 않으면 다음과 같은 문제점이 발생할 수 있습니다.

- 기본 데이터베이스에서 제공된 로그 파일을 재생하는 동안 대기 데이터베이스에 대해 오류 메시지가 리턴될 수 있습니다.
- 인계 작업 후, 새 기본 데이터베이스는 워크로드를 처리할 수 없으므로 원래 기본 데이터베이스에 연결될 때 수신하지 못했던 오류 메시지를 응용프로그램이 수신하거나 성능 문제점이 발생합니다.

기본 데이터베이스에서 작성하는 구성 매개변수의 변경사항은 자동으로 대기 데이터베이스에 전달되지 않으므로 대기 데이터베이스에서 수동으로 작성해야 합니다. 동적 구성 매개변수의 경우 변경사항은 데이터베이스 관리 시스템(DBMS) 또는 데이터베이스를 종료하고 재시작하지 않아도 적용됩니다. 비동적 구성 매개변수의 경우 변경사항은 대기 데이터베이스가 재시작된 후 적용됩니다.

대기 데이터베이스에서 로그 파일 구성 매개변수의 크기

위에 설명된 구성 매개변수 동작에 대한 한 가지 예외는 LOGFILSIZ 데이터베이스 구성 매개변수입니다. 이 매개변수가 대기 데이터베이스에 복제되지 않아도 두 데이터베이스 모두에서 로그 파일이 동일하도록 하기 위해, 대기 데이터베이스는 로컬 LOGFILSIZ 구성을 무시하고 기본 데이터베이스의 로그 파일 크기와 일치하는 로컬 로그 파일을 작성합니다.

인계 후, 원래의 대기(새 기본) 데이터베이스는 데이터베이스가 재시작할 때까지 원래 기본 데이터베이스에 설정된 값을 계속 사용합니다. 이 때, 새 기본 데이터베이스는 로컬로 구성된 값으로 되돌립니다. 또한 새 기본 데이터베이스는 현재 로그 파일을 절단하고 이전에 작성된 로그 파일의 크기를 조정합니다.

데이터베이스가 강요되지 않은 인계를 통해 역할 전환을 유지하고 어느 데이터베이스도 비활성화되지 않은 경우, 사용되는 로그 파일 크기는 항상 처음 기본 데이터베이스에서 설정된 크기가 됩니다. 그러나 원래 대기(새 기본) 데이터베이스에서 비활성 후 재시작이 발생한 경우 로컬로 구성된 로그 파일을 사용합니다. 이 로그 파일 크기는 원래 기본 데이터베이스가 다시 인계하는 경우에 계속 사용됩니다. 원래 기본 데이터베이스에서 비활성 및 재시작이 발생한 후에만 로그 파일 크기는 원래 기본의 설정으로 되돌아갑니다.

대기 데이터베이스의 로그 수신 버퍼 크기

디폴트로, 대기 데이터베이스의 로그 수신 버퍼 크기는 기본 데이터베이스의 LOGBUFSZ 기본 데이터베이스에 지정된 값의 두 배가 됩니다. 이 크기가 충분하지 않을 경우가 있습니다. 예를 들어, HADR 동기화 모드가 비동기이고 기본 및 대기 데이터베이스가 피어 상태인 경우, 기본 데이터베이스에서 트랜잭션 로드가 높으면 대기 데이터베이스의 로그 수신 버퍼 용량이 다 채워져서 기본 데이터베이스로부터의 로그 제공 작업이 중단될 수 있습니다. 이와 같은 임시 최대 상황을 관리하기 위해, DB2_HADR_BUF_SIZE 레지스트리 변수를 수정하여 대기 데이터베이스에서 로그 수신 버퍼의 크기를 늘릴 수 있습니다.

로드 조작 및 HADR

로드 조작이 COPY YES 옵션을 사용하여 기본 데이터베이스에 대해 실행되는 경우, 명령은 기본 데이터베이스에 대해 실행되고 데이터는 LOAD 명령에 지정된 경로 또는 디바이스를 통해 기본에 액세스할 수 있는 한 대기 데이터베이스로 복제됩니다. 대기 데

이터베이스가 데이터에 액세스할 수 없으면 테이블이 저장된 테이블 스페이스가 대기 데이터베이스에서 유효하지 않은 것으로 표시됩니다. 대기 데이터베이스는 이 테이블 스페이스에 관련되는 추가 로그 레코드를 건너뜁니다. 로드 조작이 대기 데이터베이스에서 사본에 액세스할 수 있도록 하려면 COPY YES 옵션을 통해 출력 파일에 공유 위치를 사용하도록 하십시오. 또는 로드 조작이 수행되는 동안 대기 데이터베이스를 비활성화하고, 기본 데이터베이스에 대해 로드를 수행한 후 대기 경로에 출력 파일의 사본을 배치하고 대기 데이터베이스를 활성화하십시오.

로드 조작이 NONRECOVERABLE YES 옵션을 사용하여 기본 데이터베이스에 대해 실행되는 경우, 명령은 기본 데이터베이스에 대해 실행되고 대기 데이터베이스의 테이블은 유효하지 않은 것으로 표시됩니다. 대기 데이터베이스는 이 테이블에 관련되는 추가 로그 레코드를 건너뜁니다. 테이블을 다시 가져오기 위해 COPY YES 및 REPLACE 옵션이 지정된 LOAD 명령을 발행할 것을 선택하거나, 스페이스를 복구하기 위해 테이블을 삭제할 수 있습니다.

COPY NO 옵션을 사용하여 로드 조작을 실행하는 것은 HADR에서 지원되지 않으므로, 명령은 자동으로 NONRECOVERABLE 옵션을 사용하는 로드 조작으로 변환됩니다. COPY NO 옵션을 사용하여 로드 조작이 COPY YES 옵션을 사용하는 로드 조작으로 변환될 수 있도록 하려면 기본 데이터베이스에서 DB2_LOAD_COPY_NO_OVERRIDE 레지스트리 변수를 설정하십시오. 이 레지스트리 변수는 대기 데이터베이스에서 무시됩니다. 대기 데이터베이스가 동일한 경로, 디바이스 또는 로드 라이브러리를 사용하여 기본 데이터베이스에 지정된 디바이스 또는 디렉토리에 액세스할 수 있는지 확인하십시오.

TSM(Tivoli Storage Manager)을 사용하여 COPY YES 옵션으로 로드 조작을 수행하는 경우, 기본 및 대기 데이터베이스에서 VENDOROPT 구성 매개변수를 설정해야 할 수도 있습니다. TSM의 구성 방법에 따라 기본 및 대기 데이터베이스의 값이 같지 않을 수도 있습니다. 또한 TSM을 사용하여 COPY YES 옵션으로 로드 조작을 수행할 때, GRANT 옵션과 함께 db2adutl 명령을 발행하여 로드된 파일에 대한 읽기 액세스를 대기 데이터베이스에 부여해야 합니다.

COPY YES 옵션을 사용하는 로드 조작으로 테이블 데이터가 복제되면, 인덱스도 복제됩니다.

- 인덱스 모드가 REBUILD로 설정되고 테이블 속성이 LOG INDEX BUILD로 설정되거나, 테이블 속성이 DEFAULT로 설정되고 LOGINDEXBUILD 데이터베이스 구성 매개변수가 ON으로 설정된 경우, 기본 데이터베이스는 대기 데이터베이스가 인덱스 오브젝트를 복제할 수 있도록 사본 파일에 재빌드된 인덱스 오브젝트를 포함합니다. 대기 데이터베이스의 인덱스 오브젝트가 로드 조작 이전에 유효하지 않은 것으로 표시되면, 인덱스 재빌드 결과로 로드 조작 이후에 다시 사용 가능하게 됩니다.
- 인덱스 모드가 INCREMENTAL로 설정되고 테이블 속성이 LOG INDEX BUILD로 설정되거나, 테이블 속성이 NULL로 설정되고 LOGINDEXBUILD 데이터베이스 구

성 매개변수가 기본 데이터베이스에서 ON으로 설정된 경우, 대기 데이터베이스의 인덱스 오브젝트는 로드 조작 이전에 유효하지 않은 것으로 표시되지 않은 경우에만 갱신됩니다. 그렇지 않으면 인덱스는 대기 데이터베이스에서 유효하지 않은 것으로 표시됩니다.

레지스트리 변수 DB2_HADR_PEER_WAIT_LIMIT

레지스트리 변수 *DB2_HADR_PEER_WAIT_LIMIT*가 설정되면, HADR 기본 데이터베이스는 대기 데이터베이스로의 로그 복제로 인해 지정된 시간(초) 동안 기본 데이터베이스기본 데이터베이스 로그온이 차단된 경우 피어 상태에서 벗어납니다. 이 한계에 도달하면 기본 데이터베이스는 대기 데이터베이스와의 연결을 중단합니다. 피어 창을 사용하지 않는 경우, 기본 데이터베이스는 연결이 끊어진 상태가 되고 로깅이 다시 시작됩니다. 피어 창을 사용하는 경우, 기본 데이터베이스는 연결이 끊어진 피어 상태가 되고 로깅은 계속 차단됩니다. 기본 데이터베이스는 재연결 또는 피어 창 만기 시 연결이 끊어진 피어 상태에서 나갑니다. 기본 데이터베이스가 연결이 끊어진 피어 상태에서 나가면 로깅이 다시 시작됩니다.

피어 상태를 벗어날 때 피어 창 전이를 이행하면 모든 경우에서 피어 창 시맨틱이 안전하게 인계됩니다. 기본 데이터베이스가 전이 중에 실패하면 정상적인 피어 창 보호는 계속 적용됩니다(계속 연결이 끊어진 피어 상태에 있는 한 대기로부터 안전하게 인계됨).

대기 데이터베이스 측에서, 연결이 끊어진 후 데이터베이스는 이미 수신된 로그를 계속 재생합니다. 수신된 로그가 재생되면, 대기 데이터베이스는 기본 데이터베이스에 다시 연결합니다. 다시 연결되면, 일반 상태 전이가 진행됩니다(먼저 리모트 만회 상태, 그 다음 피어 상태).

HADR_TIMEOUT과의 관계:

HADR_TIMEOUT 데이터베이스 구성 매개변수는 기본 데이터베이스가 차단되어 있는 동안 대기 데이터베이스로부터 하트비트 메시지를 계속 수신하는 경우 피어 상태를 벗어나지 않도록 합니다. **HADR_TIMEOUT**은 HADR 네트워크 계층에 대한 시간종료입니다. HADR 데이터베이스는 **HADR_TIMEOUT** 기간 동안 해당되는 상대로부터 어떤 메시지도 수신하지 않은 경우 상대 데이터베이스와의 연결을 중단합니다. 로그 제공 및 수신확인과 같은 상위 계층 조작에 대해서는 시간종료를 제어하지 않습니다. 대기 데이터베이스에서의 로그 재생이 로드 또는 재구성과 같은 대형 조작에 집착하는 경우, HADR 구성요소는 정상적인 스케줄에 따라 하트비트 메시지를 기본 데이터베이스로 계속 보냅니다. 이와 같은 시나리오에서, *DB2_HADR_PEER_WAIT_LIMIT*가 설정되지 않으면 대기 데이터베이스 재생이 차단되는 한 기본 데이터베이스도 차단됩니다.

*DB2_HADR_PEER_WAIT_LIMIT*는 연결 상태에 관계없이 기본 로깅 차단을 해제합니다. *DB2_HADR_PEER_WAIT_LIMIT*가 설정되지 않은 경우에도 기본 데이터베이스

는 네트워크 오류가 발견되거나 연결이 닫힐 때(**HADR_TIMEOUT**의 결과로) 항상 피어 상태를 벗어납니다.

HADR 구성 매개변수

HADR을 지원하기 위해 몇 가지의 새 데이터베이스 구성 매개변수를 사용할 수 있습니다. 이 매개변수를 설정하면 데이터베이스의 역할이 변경되지 않습니다. START HADR 또는 STOP HADR 명령을 발행하여 데이터베이스 역할을 변경해야 합니다.

HADR 구성 매개변수는 동적이 아닙니다. HADR 구성 매개변수의 변경사항은 데이터베이스가 종료되고 재시작할 때까지 적용되지 않습니다. 파티션된 데이터베이스 환경에서, HADR 구성 매개변수는 볼 수 있고 변경 가능하지만 무시됩니다.

기본 데이터베이스의 로컬 호스트 이름은 대기 데이터베이스의 리모트 호스트 이름과 같아야 하며, 대기 데이터베이스의 로컬 호스트 이름은 기본 데이터베이스의 리모트 호스트 이름과 같아야 합니다. HADR_LOCAL_HOST 및 HADR_REMOTE_HOST 구성 매개변수를 사용하여 각 데이터베이스에 로컬 및 리모트 호스트를 설정하십시오. 연결이 설정될 때 로컬 및 리모트 호스트 이름에 대한 구성 일관성을 점검하여 지정된 리모트 호스트가 예상된 데이터베이스인지 확인합니다.

HADR 데이터베이스가 IPv4 또는 IPv6를 사용하여 해당되는 상대 데이터베이스를 찾도록 구성할 수 있습니다. 호스트 서버가 IPv6를 지원하지 않으면, 데이터베이스는 IPv4를 사용합니다. 서버가 IPv6를 지원하지 않은 경우 데이터베이스가 IPv4 또는 IPv6를 사용하는지 여부는 HADR_LOCAL_HOST 및 HADR_REMOTE_HOST 구성 매개변수에 지정된 주소의 형식에 따라 결정됩니다. 데이터베이스는 두 개의 매개변수를 동일한 IP 형식으로 분석하려고 합니다. 다음 표는 IPv6 사용 가능 서버에 대해 IP 모드가 판별되는 방법을 보여줍니다.

HADR_LOCAL_HOST에 대해 사용되는 IP 모드	**HADR_REMOTE_HOST에 대해 사용되는 IP 모드**	**HADR 통신에 대해 사용되는 IP 모드**
IPv4 주소	IPv4 주소	IPv4
IPv4 주소	IPv6 주소	오류
IPv4 주소	호스트 이름(v4에만 맵핑됨)	IPv4
IPv4 주소	호스트 이름(v6에만 맵핑됨)	오류
IPv4 주소	호스트 이름(v4 및 v6에 맵핑됨)	IPv4
IPv6 주소	IPv4 주소	오류
IPv6 주소	IPv6 주소	IPv6
IPv6 주소	호스트 이름(v4에만 맵핑됨)	오류
IPv6 주소	호스트 이름(v6에만 맵핑됨)	IPv6
IPv6 주소	호스트 이름(v4 및 v6에 맵핑됨)	IPv6
호스트 이름(v4에만 맵핑됨)	IPv4 주소	IPv4
호스트 이름(v4에만 맵핑됨)	IPv6 주소	오류
호스트 이름(v4에만 맵핑됨)	호스트 이름(v4에만 맵핑됨)	IPv4

HADR_LOCAL_HOST에 대해 사용되는 IP 모드	HADR_REMOTE_HOST에 대해 사용되는 IP 모드	HADR 통신에 대해 사용되는 IP 모드
호스트 이름(v4에만 맵핑됨)	호스트 이름(v6에만 맵핑됨)	오류
호스트 이름(v4에만 맵핑됨)	호스트 이름(v4 및 v6에 맵핑됨)	IPv4
호스트 이름(v6에만 맵핑됨)	IPv4 주소	오류
호스트 이름(v6에만 맵핑됨)	IPv6 주소	IPv6
호스트 이름(v6에만 맵핑됨)	호스트 이름(v4에만 맵핑됨)	오류
호스트 이름(v6에만 맵핑됨)	호스트 이름(v6에만 맵핑됨)	IPv6
호스트 이름(v6에만 맵핑됨)	호스트 이름(v4 및 v6에 맵핑됨)	IPv6
호스트 이름(v4 및 v6에 맵핑됨)	IPv4 주소	IPv4
호스트 이름(v4 및 v6에 맵핑됨)	IPv6 주소	IPv6
호스트 이름(v4 및 v6에 맵핑됨)	호스트 이름(v4에만 맵핑됨)	IPv4
호스트 이름(v4 및 v6에 맵핑됨)	호스트 이름(v6에만 맵핑됨)	IPv6
호스트 이름(v4 및 v6에 맵핑됨)	호스트 이름(v4 및 v6에 맵핑됨)	IPv6

기본 및 대기 데이터베이스는 동일한 형식을 사용하는 경우에만 HADR에 연결합니다. 하나의 서버가 IPv6 사용 가능하고(IPv4도 지원함) 다른 서버는 IPv4만 지원하는 경우, HADR_LOCAL_HOST 또는 HADR_REMOTE_HOST 매개변수 중 하나 이상에서 IPv4 주소를 지정해야 합니다. 이는 서버가 IPv6를 지원해도 데이터베이스가 IPv4를 사용하도록 지시합니다.

update database configuration 명령을 준비하는 동안 고가용성 재해 복구(HADR) 로컬 서비스 및 리모트 서비스 매개변수(HADR_LOCAL_SVC 및 HADR_REMOTE_SVC)에 값을 지정하는 경우, 지정하는 값은 다른 DB2 구성요소나 다른 HADR 데이터베이스를 비롯하여 다른 서비스에 사용하지 않는 포트여야 합니다. 특히, 하나의 매개변수 값을 리모트 클라이언트로부터의 통신을 기다리기 위해 서버에서 사용되는 TCP/IP 포트(SVCENAME 데이터베이스 관리 프로그램 구성 매개변수)나 다음 포트(SVCENAME + 1)로 설정할 수 없습니다.

기본 및 대기 데이터베이스가 다른 머신에 있는 경우에는 동일한 포트 번호나 서비스 이름을 사용할 수 있습니다. 그렇지 않으면 다른 값을 사용해야 합니다. HADR_LOCAL_SVC 및 HADR_REMOTE_SVC 매개변수를 포트 번호나 서비스 이름으로 설정할 수 있습니다.

동기화 모드(HADR_SYNCMODE) 및 시간종료 기간(HADR_TIMEOUT)은 기본 및 대기 데이터베이스 둘 다에서 동일해야 합니다. HADR 쌍이 연결을 설정할 때 이 구성 매개변수의 일관성을 점검합니다.

기본 및 대기 데이터베이스 사이의 통신에 TCP 연결이 사용됩니다. 대기 데이터베이스에 연결되지 않은 기본 데이터베이스(시작 중이거나 연결이 끊어져서)는 로컬 포트에서

새 연결을 대기(listen)합니다.
기본 데이터베이스에 연결되지 않은 대기 데이터베이스는 연결 요청을 리모트 호스트에 계속 발행합니다.

로컬 호스트 및 로컬 서비스 매개변수(HADR_LOCAL_HOST, HADR_LOCAL_SVC)가 기본 데이터베이스에서만 사용되어도 대기 데이터베이스에서 계속 설정하여, 대기 데이터베이스가 대기 데이터베이스로서 인계해야 하는 경우 준비하도록 해야 합니다.

기본 데이터베이스가 시작할 때 대기 데이터베이스가 연결하도록 최소 30초 또는 HADR_TIMEOUT 데이터베이스 구성 매개변수에 지정된 시간(초) 동안(어느 것이든지 더 긴 시간) 기다립니다. 대기 데이터베이스가 지정된 시간 안에 연결되지 않으면 시작되지 않습니다. (한 가지 예외는 BY FORCE 옵션과 함께 START HADR 명령이 발행되는 경우입니다.)

HADR 쌍이 연결을 설정한 후에는 하트비트 메시지를 교환합니다. 하트비트 간격은 HADR_TIMEOUT 데이터베이스 구성 매개변수 값의 1/4 또는 30초(어느 것이든지 더 짧은 시간)입니다. HADR_HEARTBEAT 모니터 요소는 데이터베이스가 수신할 것으로 예상했지만 다른 데이터베이스에서 수신하지 못한 하트비트 수를 표시합니다. 하나의 데이터베이스가 HADR_TIMEOUT에 지정된 시간(초) 내에 다른 데이터베이스로부터 메시지를 수신하지 못하면 연결 끊기를 초기화합니다. 이는 기본 데이터베이스가 대기 데이터베이스나 중개 네트워크의 실패를 발견하도록 하는 데 많아야 HADR_TIMEOUT에 지정된 시간(초)을 사용한다는 것을 의미합니다.
HADR_TIMEOUT 구성 매개변수를 너무 낮은 값으로 설정하면 거짓 알람 및 잦은 연결 끊어짐을 수신합니다.

HADR_PEER_WINDOW 데이터베이스 구성 매개변수가 0으로 설정되면 기본 및 대기 데이터베이스가 피어 상태에 있을 때 대기 데이터베이스 또는 네트워크에 문제점이 발생하면 HADR_TIMEOUT 구성 매개변수에 지정된 시간(초) 동안 기본 트랜잭션 처리만 차단됩니다. HADR_PEER_WINDOW를 0이 아닌 값으로 설정한 경우, 기본 데이터베이스는 대기 데이터베이스와의 연결이 복원될 때까지 또는 HADR_PEER_WINDOW 시간 값이 경과할 때까지(어느 것이든지 먼저 발생하는 것) 트랜잭션을 커미트하지 않습니다.

주: 최대 사용 가능성을 위해, HADR_PEER_WINDOW 데이터베이스 구성 매개변수의 디폴트값은 0입니다. HADR_PEER_WINDOW가 0으로 설정되면, 기본 및 대기 데이터베이스 사이의 연결이 닫히는 대로(대기 데이터베이스가 연결을 닫았거나, 네트워크 오류가 발견되었거나, 시간종료 값에 도달하여) 즉시 기본 데이터베이스가 트랜잭션 차단을 피하기 위해 피어 상태를 벗어납니다. 데이터 일관성은 높이고 사용 가능성은 줄이기 위해, HADR_PEER_WINDOW 데이터베이스 구성 매개변수를 0이 아닌 값으로 설정할 수 있습니다. 그러면 기본 데이터베이스가 HADR_PEER_WINDOW 값에 지정된 시간 동안 연결이 끊어진 피어 상태로 유지됩니다.

다음 샘플 구성은 기본 및 대기 데이터베이스의 구성입니다.

기본 데이터베이스:

```
HADR_LOCAL_HOST           host1.ibm.com
HADR_LOCAL_SVC            hadr_service
HADR_REMOTE_HOST          host2.ibm.com
HADR_REMOTE_SVC           hadr_service
HADR_REMOTE_INST          dbinst2
HADR_TIMEOUT              120
HADR_SYNCMODE             NEARSYNC
HADR_PEER_WINDOW          120
```

대기 데이터베이스:

```
HADR_LOCAL_HOST           host2.ibm.com
HADR_LOCAL_SVC            hadr_service
HADR_REMOTE_HOST          host1.ibm.com
HADR_REMOTE_SVC           hadr_service
HADR_REMOTE_INST          dbinst1
HADR_TIMEOUT              120
HADR_SYNCMODE             NEARSYNC
HADR_PEER_WINDOW          120
```

DB2 고가용성 재해 복구(HADR)에 대한 로그 아카이브 구성

DB2 고가용성 재해 복구(HADR)에서 로그 아카이브를 사용하려면, 모든 로그 아카이브 위치에서 자동 로그 검색 기능에 대해 기본 데이터베이스와 대기 데이터베이스 둘 다를 구성하십시오.

대기 데이터베이스나 기본 데이터베이스가 모든 로그 아카이브 위치에 액세스할 수 없는 경우, 수동으로 로그 아카이브에서 다음 위치로 로그 파일을 복사해야 합니다.

- 대기 데이터베이스 로그 경로 또는 로컬 만회를 위한 아카이브 위치
- 기본 데이터베이스 로그 경로 또는 리모트 만회를 위한 아카이브 위치

현재 기본 데이터베이스만 로그 아카이브를 수행할 수 있습니다. 기본 및 대기 데이터베이스가 별도의 아카이브 위치에서 설정된 경우, 로그는 기본 데이터베이스의 아카이브 위치에만 아카이브됩니다. 인계 발생 시, 대기 데이터베이스는 새 기본 데이터베이스가 되고 해당 시점에서부터 아카이브된 로그는 원래 대기 데이터베이스의 아카이브 위치에 저장됩니다. 이와 같은 구성에서, 로그는 하나의 위치 또는 다른 위치에 아카이브되지만 두 위치 모두에 아카이브되지 않습니다(인계 이후에는 제외). 새 기본 데이터베이스는 원래 기본 데이터베이스가 이미 아카이브한 몇 개의 로그를 아카이브할 수도 있습니다.

인계 후, 기본 데이터베이스(원래 대기 데이터베이스)에서 미디어 실패가 발생하여 리스토어 및 롤 포워드를 수행해야 하는 경우, 원래 기본 데이터베이스 아카이브 위치에만 존재하는 로그에 액세스해야 할 수 있습니다.

대기 데이터베이스는 기본 데이터베이스가 로그 파일을 아카이브했음을 통지할 때까지 로컬 로그 경로에서 로그 파일을 삭제하지 않습니다. 이 동작은 로그 파일이 손실되지 않도록 부가 보호 기능을 제공합니다. 특정 로그 파일이 기본 데이터베이스에서 아카이브되기 전에 기본 데이터베이스가 실패하고 해당 로그 디스크가 손상되는 경우, 대기 데이터베이스는 소유하고 있는 디스크에서 해당 로그 파일을 삭제하지 않습니다. 기본 데이터베이스가 로그 파일을 성공적으로 아카이브했음을 알리는 통지가 수신되지 않았기 때문입니다. 대기 데이터베이스가 새 기본 데이터베이스로서 인계하는 경우, 해당 로그 파일을 재사용하기 전에 아카이브합니다. *logarchmeth1* 및 *logarchmeth2* 구성 매개변수 둘 다 사용 중인 경우, 대기 데이터베이스는 기본 데이터베이스가 두 방법을 사용하여 로그 파일을 아카이브할 때까지 그 로그 파일을 재사용하지 않습니다.

고가용성 재해 복구(HADR) 성능

네트워크 대역폭, CPU 성능 및 버퍼 크기를 비롯하여 데이터베이스 시스템의 여러 측면을 구성하면 DB2 고가용성 재해 복구(HADR) 데이터베이스의 성능을 개선할 수 있습니다.

최적의 HADR 성능을 위해서는, 시스템 관리를 위해 다음 권장사항을 고려하십시오.

- 네트워크 대역폭은 데이터베이스 로그 생성 비율보다 커야 합니다.
- 네트워크 대기 시간은 SYNC 및 NEARSYNC 모드에서만 기본 데이터베이스에 영향을 줍니다.
- SYNC 모드를 사용한 결과로 발생하는 시스템 성능 저하는 다른 동기화 모드보다 상당히 클 수 있습니다. SYNC 모드에서, 기본 데이터베이스는 로그 페이지가 기본 데이터베이스 로그 디스크에 성공적으로 기록된 후에만 대기 데이터베이스로 로그 페이지를 보냅니다. 시스템 무결성을 보호하기 위해, 기본 데이터베이스는 트랜잭션이 준비 완료되거나 커미트되었음을 응용프로그램에 통지하기 전에 대기 데이터베이스의 수신확인을 기다립니다. 대기 데이터베이스는 수신된 로그 페이지를 대기 데이터베이스 디스크에 쓴 후에만 수신확인을 보냅니다. 결과 오버헤드는 대기 데이터베이스에 대한 로그 쓰기에 라운드 트립 메시징을 더한 것입니다.
- NEARSYNC 모드에서, 기본 데이터베이스는 로그 페이지를 병렬로 쓰고 보냅니다. 그런 다음 기본 데이터베이스는 대기 데이터베이스의 수신확인을 기다립니다. 대기 데이터베이스는 로그 페이지가 메모리에 수신되는 즉시 수신확인합니다. 급속 네트워크에서는, 기본 데이터베이스에 대한 오버헤드가 최소화됩니다. 수신확인은 기본 데이터베이스가 로컬 로그 쓰기를 완료할 때 이미 도착했을 수 있습니다.
- ASYNC 모드의 경우에도 로그 쓰기 및 보내기가 병렬이지만, 이 모드에서는 기본 데이터베이스가 대기의 수신확인을 기다리지 않습니다. 따라서 네트워크 대기 시간은 문제가 되지 않습니다. 성능 오버헤드도 NEARSYNC 모드보다 ASYNC 모드에서 더 적습니다.
- 기본 데이터베이스에서의 로그 쓰기마다, 동일한 로그 페이지를 대기 데이터베이스로 보냅니다. 각각의 쓰기 작업을 *플러시*라고 합니다. 플러시 크기는 기본 데이터베이스

의 로그 버퍼 *크기*(*logbufsz* 데이터베이스 구성 매개변수로 제어되는)로 제한됩니다. 각 플러시의 정확한 *크기*는 판별할 수 없습니다. 로그 버퍼가 클수록 반드시 플러시 *크기*가 커지는 것은 아닙니다.

- 대기 데이터베이스는 기본 데이터베이스에서 생성되는 것처럼 빠르게 데이터베이스의 로그된 작업을 재생할 수 있도록 충분히 강력해야 합니다. 동일한 기본 및 대기 하드웨어를 권장합니다.
- 대부분의 시스템에서, 로깅 성능은 한계까지 구동하지 않습니다. SYNC 모드에서도, 기본 데이터베이스에서의 주목할 만한 감속은 없을 수 있습니다. 예를 들어, 로깅 제한이 HADR 사용 가능 상태에서 초당 40Mb이지만, 시스템은 HADR이 사용 가능하기 전에 초당 30Mb로 실행 중이었을 것이므로, 전체 시스템 성능에서 차이를 인지하지 못할 수 있습니다.
- 만회 속도를 높이기 위해, 공유 로그 아카이브 디바이스를 사용할 수 있습니다. 그러나 공유 디바이스가 테이프 드라이브와 같은 직렬 디바이스인 경우 읽기 및 쓰기의 혼합 작업으로 인해 기본 및 대기 데이터베이스 둘 다에서 성능 저하가 발생할 수 있습니다.

네트워크 과다 전송

대기 데이터베이스가 로그 페이지를 너무 느리게 재생하면 로그 수신 버퍼가 가득 차서 버퍼가 추가 로그 페이지를 수신하지 못할 수 있습니다. SYNC 및 NEARSYNC 모드에서, 기본 데이터베이스가 로그 버퍼를 두 번 이상 플러시한 경우, 데이터는 기본 머신, 네트워크 및 대기 데이터베이스로 구성되는 네트워크 파이프라인에서 버퍼링될 수 있습니다. 대기 데이터베이스는 데이터를 수신할 여유 공간을 가지고 있지 않아서 수신 확인할 수 없으므로, 기본 데이터베이스는 대기 데이터베이스의 수신확인을 기다리는 동안 차단됩니다.

ASYNC 모드에서, 기본 데이터베이스는 파이프라인이 가득 차서 추가 로그 페이지를 보낼 수 없을 때까지 로그 페이지를 계속 보냅니다. 이 조건을 *과다 전송*이라고 합니다. 과다 전송은 **hadr_connect_status** 모니터 요소에 의해 보고됩니다. SYNC 및 NEARSYNC 모드의 경우, 파이프라인은 보통 단일 플러시를 병합하므로 과다 전송이 발생하지 않습니다. 그러나 기본 데이터베이스는 플러시 작업에 대한 대기 데이터베이스의 수신확인을 기다리면서 차단된 상태로 유지합니다.

또한 과다 전송은 대기 데이터베이스가 재생하는 데 많은 시간을 소모하는 로그 레코드(예: 데이터베이스 또는 테이블 재구성 로그 레코드)를 재생하는 경우에도 발생합니다.

대기 데이터베이스 로그 수신 버퍼의 *크기*를 늘리면 과다 전송의 모든 원인을 제거할 수는 없지만 과다 전송을 줄이는 데 도움이 될 수 있습니다. 디폴트로, 대기 데이터베이스 로그 수신 버퍼의 *크기*는 기본 데이터베이스 로그 쓰기 버퍼 *크기*의 두 배입니다. *logbufsz* 데이터베이스 구성 매개변수는 기본 데이터베이스 로그 쓰기 버퍼의 *크기*를 지

정합니다. DB2 레지스트리 변수 **DB2_HADR_BUF_SIZE**를 사용하여 대기 데이터베이스 로그 수신 버퍼의 *크기*를 조정할 수 있습니다.

레지스트리 변수 *DB2_HADR_PEER_WAIT_LIMIT*

레지스트리 변수 **DB2_HADR_PEER_WAIT_LIMIT**가 설정된 경우, HADR 기본 데이터베이스는 대기 데이터베이스로의 로그 복제로 인해 지정된 시간(초) 동안 기본 데이터베이스의 로깅이 차단된 경우 피어 상태에서 벗어납니다. 이 한계에 도달하면 기본 데이터베이스는 대기 데이터베이스와의 연결을 중단합니다. 피어 창을 사용하지 않는 경우, 기본 데이터베이스는 연결이 끊어진 상태가 되고 로깅이 다시 시작됩니다. 피어 창을 사용하는 경우, 기본 데이터베이스는 연결이 끊어진 피어 상태가 되고 로깅은 계속 차단됩니다. 기본 데이터베이스는 재연결 또는 피어 창 만기 시 연결이 끊어진 피어 상태에서 나갑니다. 기본 데이터베이스가 연결이 끊어진 피어 상태에서 나가면 로깅이 다시 시작됩니다.

피어 상태를 벗어날 때 피어 창 전이를 이행하면 모든 경우에서 피어 창 시맨틱이 안전하게 인계됩니다. 기본 데이터베이스가 전이 중에 실패하면 정상적인 피어 창 보호는 계속 적용됩니다(계속 연결이 끊어진 피어 상태에 있는 한 대기로부터 안전하게 인계됨).

레지스트리 변수 *DB2_HADR_SOSNDBUF* 및 *DB2_HADR_SORCVBUF*

네트워크와 HADR 성능을 최대화하려면 TCP 소켓 버퍼 *크기*를 조정해야 할 수도 있습니다. HADR 로그 제공 워크로드, 네트워크 대역폭 및 전송 대기 시간은 TCP 소켓 버퍼 *크기*를 조정할 때 고려할 중요한 요소입니다. 시스템 레벨에서 TCP 소켓 버퍼 크기를 변경하는 경우 머신에서 모든 TCP 연결에 설정을 적용할 수 있습니다. 대형 시스템 레벨 소켓 버퍼 *크기* 설정에는 많은 양의 메모리가 소모됩니다.

두 개의 레지스트리 변수인 **DB2_HADR_SOSNDBUF** 및 **DB2_HADR_SORCVBUF**를 사용하여 HADR 연결에 대한 TCP 소켓 송신 및 수신 버퍼 *크기*를 조정할 수 있습니다. 이 두 가지 변수의 값 범위는 1024 - 4294967295이며 디폴트는 운영 체제의 소켓 버퍼 *크기*로, 운영 체제에 따라 다릅니다. 일부 운영 체제는 자동으로 사용자 지정 값에 상한을 지정하거나 자동으로 반올림합니다.

클러스터 관리 프로그램 및 고가용성 재해 복구(HADR)

클러스터 노드에서 DB2 고가용성 재해 복구(HADR) 데이터베이스를 구현하고 클러스터 관리 프로그램을 사용하여 데이터베이스 솔루션의 사용 가능성을 개선할 수 있습니다. 동일한 클러스터 관리 프로그램이 기본 데이터베이스와 대기 데이터베이스 둘 다를 관리하도록 하거나, 다른 클러스터 관리 프로그램이 기본 데이터베이스와 대기 데이터베이스를 관리하도록 할 수 있습니다.

동일한 클러스터 관리 프로그램이 기본 및 대기 데이터베이스에 서비스를 제공하는 HADR 쌍 설정

이 구성은 기본 및 대기 데이터베이스가 동일한 사이트에 위치하고 가장 빠른 장애 복구가 필요한 환경에 가장 적합합니다. 이 환경에서는 응급 복구나 다른 복구 방법을 사용하는 것보다 HADR을 사용하여 DBMS 사용 가능성을 유지하는 것이 좋습니다.

클러스터 관리 프로그램을 사용하여 문제점을 신속하게 발견하고 인계 작업을 초기화할 수 있습니다. HADR에는 DBMS에 대한 별도의 스토리지가 필요하므로, 별도의 볼륨 제어를 사용하여 클러스터 관리 프로그램을 구성해야 합니다. 이 구성에서는 클러스터 관리 프로그램이 대기 시스템에서 DBMS를 사용하기 전에 볼륨에 대해 장애 복구가 발생할 때까지 기다리지 않아도 됩니다. 자동 클라이언트 리라우트 기능을 사용하여 클라이언트 응용프로그램 경로를 새 기본 데이터베이스로 재지정할 수 있습니다.

동일한 클러스터 관리 프로그램이 기본 및 대기 데이터베이스에 서비스를 제공하지 않는 HADR 쌍 설정

이 구성은 기본 및 대기 데이터베이스가 다른 사이트에 위치하고 전체 사이트 실패 발생 시 재해 복구를 위해 고가용성이 필요한 환경에 가장 적합합니다. 이 구성을 구현할 수 있는 몇 가지 방법이 있습니다. HADR 1차 또는 대기 데이터베이스가 클러스터의 일부일 경우 가능한 두 가지 장애 복구 시나리오가 있습니다.

- 부분 사이트 실패가 발생하고 DBMS가 장애 복구할 수 있는 노드가 사용 가능 상태로 유지되는 경우, 클러스터 장애 복구를 수행할 것을 선택할 수 있습니다. 이 경우, 클러스터 관리 프로그램을 사용하여 IP 주소 및 볼륨 장애 복구가 수행됩니다. HADR은 영향을 받지 않습니다.
- 기본 데이터베이스가 있는 전체 사이트 실패가 발생하면 HADR을 사용하여 인계 조작을 시작하여 DBMS 사용 가능성을 유지하십시오. 대기 데이터베이스가 있는 전체 사이트 실패가 발생하면 사이트를 수리하거나 대기 데이터베이스를 다른 사이트로 이동할 수 있습니다.

대기 데이터베이스 초기화

데이터베이스 솔루션을 고가용성으로 만들기 위한 하나의 전략은 사용자 응용프로그램 요청에 응답할 기본 데이터베이스 및 기본 데이터베이스가 실패하는 경우 기본 데이터베이스에 대한 데이터베이스 조작을 인계할 수 있는 보조 또는 대기 데이터베이스를 유지하는 것입니다. 대기 데이터베이스 초기화는 기본 데이터베이스를 대기 데이터베이스에 복사를 수반합니다.

대기 데이터베이스를 초기화하는 여러 가지 방법이 있습니다. 예를 들어, 다음과 같습니다.

- 디스크 미러링을 사용하여 기본 데이터베이스를 복사하고 DB2 데이터베이스 일시중단 입출력 지원을 사용하여 미러를 분할하여 보조 데이터베이스를 작성하십시오.

- 기본 데이터베이스의 백업 이미지를 작성하고 해당 이미지를 대기 데이터베이스에 복구하십시오.
- SQL 복제를 사용하여 기본 데이터베이스에서 데이터를 캡처하고 해당 데이터를 대기 데이터베이스에 적용하십시오.

대기 데이터베이스를 초기화한 후, 기본 데이터베이스와 대기 데이터베이스를 동기화하도록 데이터베이스 솔루션을 구성하여 기본 데이터베이스가 실패하는 경우 대기 데이터베이스가 기본 데이터베이스를 인계할 수 있도록 해야 합니다.

분할 미러를 대기 데이터베이스로 사용

대기 데이터베이스로 사용하기 위해 데이터베이스의 분할 미러를 작성하려면 다음 프로시저를 사용하십시오. 기본 데이터베이스에서 실패가 발생하고 응급 복구가 필요한 경우 대기 데이터베이스를 사용하여 기본 데이터베이스를 인계할 수 있습니다.

분할 미러를 대기 데이터베이스로 사용하려면 다음 단계를 수행하십시오.

1. 기본 데이터베이스의 입출력을 일시중단하십시오.

   ```
   db2 set write suspend for database
   ```

   데이터베이스가 일시중단된 상태에서 다른 유틸리티나 도구를 실행 중이 아니어야 합니다. 오직 데이터베이스의 사본을 작성 중이어야 합니다.
2. 적합한 운영 체제 레벨 명령을 사용하여 기본 데이터베이스에서 미러를 분할하십시오.

   주: 볼륨 디렉토리를 포함한 전체 데이터베이스 디렉토리를 복사해야 합니다. 또한 로그 디렉토리 및 데이터베이스 디렉토리 밖에 존재하는 모든 컨테이너 디렉토리를 복사해야 합니다. 이 정보를 알려면 분할되어야 하는 데이터베이스의 모든 파일과 디렉토리를 표시하는 DBPATHS 관리 뷰를 참조하십시오.
3. 기본 데이터베이스에서 입출력을 재개하십시오.

   ```
   db2 set write resume for database
   ```
4. 보조 시스템에서 미러된 데이터베이스를 카탈로그하십시오.

   주: 디폴트로 미러된 데이터베이스는 기본 데이터베이스와 동일한 시스템에 존재할 수 없습니다. 동일한 디렉토리 구조를 갖고 기본 데이터베이스와 동일한 인스턴스 이름을 사용하는 보조 시스템에 위치해야 합니다. 미러된 데이터베이스가 기본 데이터베이스와 동일한 시스템에 존재해야 하는 경우 db2relocatedb 유틸리티 또는 db2inidb 명령의 RELOCATE USING 옵션을 사용하여 이를 수행할 수 있습니다.
5. 보조 시스템에서 데이터베이스 인스턴스를 시작하십시오.

   ```
   db2start
   ```

6. 보조 시스템의 미러된 데이터베이스를 롤 포워드 보류 상태에 두어서 초기화하십시오.

   ```
   db2inidb database_alias as standby
   ```

   필요한 경우 db2inidb 명령의 RELOCATE USING 옵션을 지정하여 대기 데이터베이스를 재배치하십시오.

   ```
   db2inidb database_alias as standby relocate using relocatedbcfg.txt
   ```

   여기서 relocatedbcfg.txt 파일에 데이터베이스를 재배치하기 위해 필요한 정보가 들어 있습니다.

   주:

   a. DMS 테이블 스페이스(데이터베이스 관리 스페이스) 또는 자동 스토리지 테이블 스페이스가 있는 경우 분할 미러를 사용하여 전체 데이터베이스 백업을 만들 수 있습니다. 분할 미러를 사용한 백업 작성은 프로덕션 데이터베이스에서 백업을 작성할 때의 오버헤드를 로드 해제합니다.

   b. 데이터베이스 디렉토리(볼륨 디렉토리 포함), 로그 디렉토리 및 컨테이너 디렉토리가 RELOCATE USING 옵션을 사용하기 전에 원하는 위치로 이동되어야 합니다.

7. User Exit 프로그램을 설정하여 기본 시스템에서 로그 파일을 검색하십시오.
8. 로그 끝이나 특정 시점까지 데이터베이스를 롤 포워드하십시오.
9. 로그 끝이나 대기 데이터베이스에 필요한 특정 시점에 도달할 때까지 계속 로그 파일을 검색하고 로그를 통해 데이터베이스를 롤 포워드하십시오.
10. 대기 데이터베이스를 온라인으로 만들려면 STOP 옵션을 지정하여 ROLLFORWARD 명령을 실행하십시오.

    주: 기본 데이터베이스의 로그는 롤 포워드 보류 상태에서 벗어난 후에는 미러된 데이터베이스에 적용될 수 없습니다.

DB2 고가용성 재해 복구(HADR) 동기화 모드 구성

HADR_SYNCMODE 구성 매개변수는 DB2 고가용성 재해 복구(HADR) 데이터베이스 솔루션이 트랜잭션 손실에 대해 갖는 보호 정도를 판별합니다. 동기화 모드는 기본 데이터베이스가 대기 데이터베이스에서의 로깅 상태를 기초로 트랜잭션이 완료된 것으로 간주할 때 판별합니다. 동기 모드 구성 매개변수 값이 정밀할 수록, 트랜잭션 데이터 손실에 대해 데이터베이스 솔루션이 갖는 보호는 더 커지지만, 트랜잭션 처리 성능이 느려집니다. 성능에 대한 필요성과 트랜잭션 손실에 대한 보호 필요성의 밸런스를 조정해야 합니다.

이 모드는 기본 및 대기 데이터베이스가 피어 또는 연결이 끊어진 피어 상태에 있는 경우에만 적용됩니다.

동기화 모드를 설정하려면 HADR_SYNCMODE 구성 매개변수를 사용하십시오. 가능한 값은 다음과 같습니다.

SYNC(동기)

이 모드는 트랜잭션 손실을 최대로 보호하지만 세 가지 모드 중에서 트랜잭션 응답 시간이 가장 오래 걸립니다.

이 모드에서 로그 쓰기는 로그가 기본 데이터베이스의 로그 파일에 기록되고 기본 데이터베이스가 대기 데이터베이스로부터 대기 데이터베이스의 로그 파일에도 로그를 기록했다는 수신확인을 받은 경우에만 성공으로 간주됩니다. 로그 데이터가 두 사이트 모두에 저장되었음이 보증됩니다.

대기 데이터베이스가 로그 레코드를 재생하기 전에 손상된 경우 다음에 시작하면 로컬 로그 파일에서 레코드를 검색하여 재생합니다. 기본 데이터베이스가 실패하면 대기 데이터베이스로 장애 복구하여 기본 데이터베이스에서 커미트된 트랜잭션이 대기 데이터베이스에서도 커미트되도록 합니다. 장애 복구 작업 후, 클라이언트가 새 기본 데이터베이스에 재연결하면 원래 기본 데이터베이스에서 커미트된 것으로 보고되지 않은 트랜잭션이 새 기본 데이터베이스에서 커미트될 수 있습니다. 이러한 상황은 기본 데이터베이스가 대기 데이터베이스의 수신확인 메시지를 처리하기 전에 실패하는 경우에 발생합니다. 클라이언트 응용프로그램은 이와 같은 트랜잭션이 존재하는지 판별하기 위해 데이터베이스를 조회할 것을 고려해야 합니다.

기본 데이터베이스가 대기 데이터베이스와의 연결이 끊어진 경우 다음에 발생하는 상황은 hadr_peer_window 데이터베이스 구성 매개변수에 따라 다릅니다. hadr_peer_window가 0이 아닌 시간 값으로 설정되면 대기 데이터베이스와의 연결이 끊어질 때 기본 데이터베이스는 연결이 끊어진 피어 상태로 이동하고 대기 데이터베이스의 수신확인을 계속 기다린 후에 트랜잭션을 커미트합니다. hadr_peer_window 데이터베이스 구성 매개변수가 0으로 설정되면 기본 및 대기 데이터베이스는 더 이상 피어 상태에 있는 것으로 간주되지 않으므로 트랜잭션은 대기 데이터베이스의 수신확인을 기다리면서 보유되지 않습니다. 데이터베이스가 피어 또는 연결 끊어진 피어 상태가 아닐 때 장애 복구 작업이 수행되면 기본 데이터베이스에서 커미트된 모든 트랜잭션이 대기 데이터베이스에 나타난다고 보장할 수 없습니다.

데이터베이스가 피어 또는 연결 끊어진 피어 상태에 있을 때 기본 데이터베이스가 실패하면 장애 복구 작업 후에 HADR 쌍을 대기 데이터베이스로 다시 조인할 수 있습니다. 로그가 대기 데이터베이스의 로그 파일에도 기록되었다는 확인을 기본 데이터베이스가 대기 데이터베이스로부터 수신할 때까지 트랜잭션이 커미트된 것으로 간주하지 않으므로, 기본 데이터베이스에서의 로그 시퀀스는 대기 데이터베이스의 로그 시퀀스와 같습니다. 원래의 기본 데이터베이스(지금은 대기 데이터베이스)는 장애 복구 작업 이후에 새 기본 데이터베이스에서 생성된 새 로그 레코드를 재생하여 만회해야 합니다.

기본 데이터베이스가 실패할 때 피어 상태에 있지 않으면, 해당 로그 시퀀스는 대기 데이터베이스의 로그 시퀀스와 다를 수도 있습니다. 장애 복구 작업이 수행되어야 하는 경우, 기본 및 대기 데이터베이스의 로그 시퀀스가 다를 수 있습니다. 대기 데이터베이스는 장애 복구 후에 자체의 로그 시퀀스를 시작하기 때문입니다. 일부 작업은 실행 취소할 수 없으므로(예: 테이블 삭제), 새 로그 시퀀스가 작성된 시점까지 기본 데이터베이스를 되돌릴 수 없습니다. 로그 시퀀스가 다르고 원래 기본 데이터베이스에 대해 STANDBY 옵션과 함께 START HADR 명령을 발행하는 경우, 명령이 성공했다는 메시지를 수신합니다. 그러나 이 메시지는 재통합 시도 이전에 발행됩니다. 재통합이 실패하면 기본 및 대기 데이터베이스 둘 다의 관리 로그 및 진단 로그에 쌍 확인 메시지가 발행됩니다. 재통합된 대기 데이터베이스는 대기로 유지되지만 기본 데이터베이스는 쌍 유효성 확인 중에 대기 데이터베이스를 거부하여 대기가 종료되도록 합니다. 원래의 기본 데이터베이스가 HADR 쌍을 성공적으로 다시 조인하면 BY FORCE 옵션을 지정하지 않고 TAKEOVER HADR 명령을 발행하여 데이터베이스 롤백을 성취할 수 있습니다. 원래의 기본 데이터베이스가 HADR 쌍을 다시 조인할 수 없으면 새 기본 데이터베이스의 백업 이미지를 리스토어하여 대기 데이터베이스로 다시 초기화할 수 있습니다.

NEARSYNC(근접 동기)

이 모드는 동기 모드보다 트랜잭션 응답 시간이 짧지만 트랜잭션 손실에 대해 제공되는 보호 정도가 덜합니다.

이 모드에서 로그 쓰기는 로그 레코드가 기본 데이터베이스의 로그 파일에 기록되고 기본 데이터베이스가 대기 시스템으로부터 대기 시스템의 기본 메모리에 로그가 기록되었다는 수신확인을 받은 경우에만 성공으로 간주됩니다. 두 사이트가 동시에 실패하고 목표 사이트가 수신한 로그 데이터를 비휘발성 스토리지로 모두 전송하지 않은 경우에만 데이터 손실이 발생합니다.

대기 데이터베이스가 로그 레코드를 메모리에서 디스크로 복사하기 전에 손상된 경우 로그 레코드는 대기 데이터베이스에서 손실됩니다. 보통 대기 데이터베이스는 재시작할 때 기본 데이터베이스로부터 누락된 로그 레코드를 가져올 수 있습니다. 그러나 기본 데이터베이스 또는 네트워크에서 실패로 검색이 불가능하여 장애 복구가 필요한 경우, 로그 레코드는 대기 데이터베이스에 전혀 나타나지 않으므로 이 로그 레코드와 연관되는 트랜잭션도 대기 데이터베이스에 전혀 나타나지 않습니다.

트랜잭션이 손실되면 새 기본 데이터베이스는 장애 복구 작업 후에 원래의 기본 데이터베이스와 동일하지 않습니다. 클라이언트 응용프로그램은 이러한 트랜잭션을 다시 제출하여 응용프로그램 상태를 최근 상태로 만들 것을 고려해야 합니다.

기본 및 대기 데이터베이스가 피어 상태에 있을 때 기본 데이터베이스가 실패한 경우, 전체 리스토어 작업을 사용하여 다시 초기화하지 않으면 원래의 기본

데이터베이스가 대기 데이터베이스로서 HADR 쌍을 다시 조인할 수 없습니다. 장애 복구에 손실된 로그 레코드가 관련되는 경우(기본 및 대기 데이터베이스가 모두 실패하여), 기본 및 대기 데이터베이스의 로그 시퀀스는 다르게 되므로, 먼저 리스토어 작업을 수행하지 않고 대기 데이터베이스로서 원래의 기본 데이터베이스를 재시작하려고 하면 실패합니다. 원래의 기본 데이터베이스가 HADR 쌍을 성공적으로 다시 조인하면 BY FORCE 옵션을 지정하지 않고 TAKEOVER HADR 명령을 발행하여 데이터베이스 롤백을 성취할 수 있습니다. 원래의 기본 데이터베이스가 HADR 쌍을 다시 조인할 수 없으면 새 기본 데이터베이스의 백업 이미지를 리스토어하여 대기 데이터베이스로 다시 초기화할 수 있습니다.

ASYNC(비동기)

이 모드는 기본 시스템이 실패하는 경우 트랜잭션 손실 가능성이 가장 높습니다. 또한 세 모드 사이에 트랜잭션 응답 시간이 가장 짧습니다.

이 모드에서 로그 쓰기는 로그 레코드가 기본 데이터베이스의 로그 파일에 기록되고 기본 시스템 호스트 머신의 TCP 계층으로 전달된 경우에만 성공으로 간주됩니다. 기본 시스템이 대기 시스템의 수신확인을 기다리지 않으므로 트랜잭션이 대기로 전달되는 중에도 커미트로 간주될 수 있습니다.

기본 데이터베이스 호스트 머신, 네트워크 또는 대기 데이터베이스에서 실패하면 로그 레코드가 전이 중에 손실될 수 있습니다. 기본 데이터베이스가 사용 가능하면 쌍 연결이 재설정될 때 누락된 로그 레코드를 대기 데이터베이스로 다시 보낼 수 있습니다. 그러나 누락된 로그 레코드가 있을 때 장애 복구 작업이 필요한 경우 그 로그 레코드는 대기 데이터베이스에 도달하지 못하므로 연관되는 트랜잭션이 장애 복구에서 손실됩니다.

트랜잭션이 손실되면 새 기본 데이터베이스는 장애 복구 작업 후에 원래의 기본 데이터베이스와 정확히 동일하지 않습니다. 클라이언트 응용프로그램은 이러한 트랜잭션을 다시 제출하여 응용프로그램 상태를 최근 상태로 만들 것을 고려해야 합니다.

기본 및 대기 데이터베이스가 피어 상태에 있을 때 기본 데이터베이스가 실패한 경우, 전체 리스토어 작업을 사용하여 다시 초기화하지 않으면 원래의 기본 데이터베이스가 대기 데이터베이스로서 HADR 쌍을 다시 조인할 수 없습니다. 장애 복구에 손실된 로그 레코드가 관련되는 경우, 기본 및 대기 데이터베이스의 로그 시퀀스는 다르게 되므로, 대기 데이터베이스로서 원래의 기본 데이터베이스를 재시작하려고 하면 실패합니다. 비동기 모드에서 장애 복구가 발생하면 로그 레코드가 손실될 가능성이 크므로, 기본 데이터베이스가 HADR 쌍을 다시 조인할 수 없게 될 가능성도 큽니다. 원래의 기본 데이터베이스가 HADR 쌍을 성공적으로 다시 조인하면 BY FORCE 옵션을 지정하지 않고 TAKEOVER HADR 명령을 발행하여 데이터베이스 롤백을 성취할 수 있습니

다. 원래의 기본 데이터베이스가 HADR 쌍을 다시 조인할 수 없으면 새 기본 데이터베이스의 백업 이미지를 리스토어하여 대기 데이터베이스로 다시 초기화할 수 있습니다.

고가용성 재해 복구(HADR) 지원

DB2 데이터베이스 고가용성 재해 복구(HADR) 기능을 최대한 이용하려면, 고가용성 데이터베이스 솔루션을 설계할 때 시스템 요구사항 및 기능 제한사항을 고려하십시오.

고가용성 재해 복구(HADR)에 대한 시스템 요구사항

고가용성 재해 복구(HADR)에 대해 최적의 성능을 달성하려면, 시스템이 다음과 같은 하드웨어, 운영 체제 및 DB2 데이터베이스 시스템의 요구사항을 충족하는지 확인하십시오.

권장사항: 더 나은 성능을 위해, 기본 데이터베이스가 있는 시스템과 대기 데이터베이스가 있는 시스템에 동일한 하드웨어 및 소프트웨어를 사용하십시오. 대기 데이터베이스가 있는 시스템에 기본 데이터베이스가 있는 시스템보다 적은 자원이 있는 경우 대기 데이터베이스는 기본 데이터베이스에 의해 생성된 트랜잭션 로드로 유지될 수 없습니다. 이와 같은 경우 대기 데이터베이스가 뒤떨어지거나 기본 데이터베이스 성능이 저하될 수 있습니다. 장애 복구 상황에서, 새 기본 데이터베이스에는 클라이언트 응용프로그램에 적절하게 서비스를 제공할 자원이 있어야 합니다.

하드웨어 및 운영 체제 요구사항

권장사항: HADR 기본 및 대기 데이터베이스에 대해 동일한 호스트 컴퓨터를 사용하십시오. 즉, 동일한 벤더에서 제공되고 동일한 아키텍처를 가지고 있어야 합니다.

기본 및 대기 데이터베이스의 운영 체제는 패치를 포함하여 동일한 버전이어야 합니다. 롤 업그레이드하는 동안 잠시 이 규칙을 위반할 수 있지만 매우 주의해야 합니다.

TCP/IP 인터페이스는 HADR 호스트 머신 사이에 사용 가능해야 하므로 고속, 고용량 네트워크가 권장됩니다.

DB2 데이터베이스 요구사항

기본 및 대기 데이터베이스에 대한 데이터베이스 시스템 버전은 동일해야 합니다. 예를 들어, 둘 다 버전 8이거나 버전 9여야 합니다. 롤 업그레이드 중에 대기 데이터베이스에 대한 데이터베이스 시스템의 수정 레벨(예: Fixpack 레벨)은 새 레벨을 테스트하기 위해 잠시 동안 기본 데이터베이스보다 더 늦을 수 있습니다. 그러나 확장된 기간 동안 이 구성을 보존하면 안됩니다. 기본 데이터베이스의 데이터베이스 시스템 수정 레벨이 대기 데이터베이스보다 늦은 경우 기본 및 대기 데이터베이스는 서로 연결하지 않습니다.

기본 및 대기 데이터베이스의 DB2 데이터베이스 소프트웨어의 비트 *크기*는 동일해야 합니다(32 또는 64비트). 테이블 스페이스와 해당되는 컨테이너는 기본 및 대기 데이터베이스에서 동일해야 합니다. 동일해야 하는 등록 정보로는 테이블 스페이스 유형(DMS 또는 SMS), 테이블 스페이스 *크기*, 컨테이너 *경로*, 컨테이너 *크기* 및 컨테이너 파일 유형(원시 디바이스 또는 파일 시스템)이 있습니다. 로그 파일에 할당된 스페이스 양도 기본 및 대기 데이터베이스에서 동일해야 합니다.

기본 데이터베이스에서 테이블 스페이스 명령문을 발행하는 *경우*(예: CREATE TABLESPACE, ALTER TABLESPACE 또는 DROP TABLESPACE) 그 명령문은 대기 데이터베이스에서 재생됩니다. 기본 데이터베이스에서 테이블 스페이스 명령문을 발행하기 전에 관련된 디바이스가 두 데이터베이스 모두에 대해 설정되어 있는지 확인해야 합니다.

기본 데이터베이스에서 테이블 스페이스를 작성하고 컨테이너를 사용할 수 없어서 대기 데이터베이스에서 로그 재생에 실패하는 경우, 기본 데이터베이스는 로그 재생이 실패했음을 알리는 오류 메시지를 수신하지 않습니다.

로그 재생 오류를 점검하려면 새 테이블 스페이스를 작성할 때 대기 데이터베이스에서 관리 통지 로드와 db2diag를 모니터해야 합니다.

인계 작업이 발생하는 경우, 사용자가 작성한 새 테이블 스페이스는 새 기본 데이터베이스에서 사용할 수 없습니다. 이 상황에서 복구하려면 백업 이미지를 통해 새 기본 데이터베이스에서 테이블 스페이스를 리스토어하십시오.

다음 예에서, 테이블 스페이스 MY_TABLESPACE는 새 기본 데이터베이스에서 사용되기 전에 데이터베이스 MY_DATABASE에서 리스토어됩니다.

1. `db2 connect to my_database`
2. `db2 list tablespaces show detail`

   주: 모든 테이블 스페이스의 상태를 표시하고 5단계에 필요한 테이블 스페이스 ID 번호를 확보하려면 db2 list tablespaces show detail 명령을 실행하십시오.
3. `db2 stop hadr on database my_database`
4. `db2 "restore database my_database tablespace (my_tablespace) online redirect"`
5. `db2 "set tablespace containers for my_tablespace_ID_# ignore rollforward container operations using (path '/my_new_container_path/')"`
6. `db2 "restore database my_database continue"`
7. `db2 rollforward database my_database to end of logs and stop tablespace "(my_tablespace)"`
8. `db2 start hadr on database my_database as primary`

기본 및 대기 데이터베이스의 데이터베이스 경로는 동일하지 않아도 됩니다. 상대 컨테이너 경로를 사용하는 경우 기본 및 대기 데이터베이스의 여러 절대 컨테이너 경로에 동일한 상대 경로가 맵핑될 수 있습니다.

자동 스토리지 데이터베이스는 ADD STORAGE ON 절이 있는 ALTER DATABASE 문의 복제를 포함하여 HADR에 의해 완전히 지원됩니다. 테이블 스페이스 컨테이너와 마찬가지로, 스토리지 경로는 기본 및 대기 데이터베이스 둘 다에 존재해야 합니다.

기본 및 대기 데이터베이스의 데이터베이스 이름은 동일해야 합니다. 이는 다른 인스턴스에 있어야 함을 의미합니다.

경로 재지정된 리스토어는 지원되지 않습니다. 즉, HADR은 테이블 스페이스 컨테이너 경로 재지정을 지원하지 않습니다. 그러나 데이터베이스 디렉토리 및 로그 디렉토리 변경사항은 지원됩니다. 상대 경로에 의해 작성된 테이블 스페이스 컨테이너는 새 데이터베이스 디렉토리에 상대적인 경로로 리스토어됩니다.

버퍼 풀 요구사항

버퍼 풀 작업은 대기 데이터베이스에서도 재생되므로, 기본 및 대기 데이터베이스가 동일한 양의 메모리를 갖는 것은 중요합니다.

고가용성 재해 복구(HADR)에 대한 설치 및 스토리지 요구사항

고가용성 재해 복구(HADR)에 대해 최적의 성능을 달성하려면, 시스템이 다음의 설치 및 스토리지 요구사항을 충족하는지 확인하십시오.

설치 요구사항

HADR의 경우, 인스턴스 경로는 기본 및 대기 데이터베이스에서 동일해야 합니다. 일부 상황에서 다른 인스턴스 경로를 사용하면 문제가 발생할 수 있습니다. 예를 들어, SQL 스토어드 프로시저에서 사용자 정의 함수(UDF)를 호출하는데 UDF 오브젝트 코드의 경로가 1차 및 대기 서버에 대한 디렉토리와 동일할 경우에 문제가 발생합니다.

스토리지 요구사항

자동 스토리지 데이터베이스는 ADD STORAGE ON 절이 있는 ALTER DATABASE 문의 복제를 포함하여 HADR에 의해 완전히 지원됩니다. 테이블 스페이스 컨테이너와 마찬가지로, 스토리지 경로는 기본 및 대기 데이터베이스 둘 다에 존재해야 합니다. 기호 링크를 사용하여 동일한 경로를 작성할 수 있습니다. 기본 및 대기 데이터베이스는 동일한 컴퓨터에 있을 수 있습니다. 이 데이터베이스의 데이터베이스 스토리지가 동일한 경로에서 시작되어도, 사용되는 실제 디렉토리는 임베드(embed)된 인스턴스 이름을 가지고 있으므로 충돌하지 않습니다(기본 및 대기 데이터베이스는 동일한 데이터베이스 이름을 가지고 있어야 하므로 다른 인스턴스에 있어야 함). 스토리지 경로는 `storage_path_name/inst_name/dbpart_name/db_name/tbsp_name/container_name`으로 공식화됩니다.

테이블 스페이스와 해당되는 컨테이너는 기본 및 대기 데이터베이스에서 동일해야 합니다. 동일해야 하는 등록 정보로는 테이블 스페이스 유형(DMS 또는 SMS), 테이블 스페이스 크기, 컨테이너 경로, 컨테이너 크기 및 컨테이너 파일 유형(원시 디바이스 또는 파일 시스템)이 있습니다. 자동 스토리지에 대해 데이터베이스가 사용 가능한 경우, 스토리지 경로가 동일해야 합니다. 여기에는 경로 이름과 각 경로에서 데이터베이스만을 위한 스페이스의 양이 포함됩니다. 로그 파일에 할당된 스페이스 양도 기본 및 대기 데이터베이스에서 동일해야 합니다.

기본 데이터베이스에서 테이블 스페이스 명령문을 발행하는 경우(예: CREATE TABLESPACE, ALTER TABLESPACE 또는 DROP TABLESPACE) 그 명령문은 대기 데이터베이스에서 재생됩니다. 기본 데이터베이스에서 테이블 스페이스 명령문을 발행하기 전에 관련된 디바이스가 두 데이터베이스 모두에 대해 설정되어 있는지 확인해야 합니다.

테이블 스페이스 설정이 기본 및 대기 데이터베이스에서 동일하지 않으면, 대기 데이터베이스에서 로그 재생 시 OUT OF SPACE 또는 TABLE SPACE CONTAINER NOT FOUND 오류가 발생할 수 있습니다. 마찬가지로, 데이터베이스가 자동 스토리지에 대해 사용 가능하고 스토리지 경로가 동일하지 않은 경우 ALTER DATABASE 문의 ADD STORAGE ON 절과 연관되는 로그 레코드는 재생되지 않습니다. 결과적으로, 기존 스토리지 경로는 대기 시스템에서 스페이스를 너무 일찍 소모할 수 있으므로 자동 스토리지 테이블 스페이스가 크기를 늘릴 수 없습니다. 이와 같은 상황이 발생하면 영향을 받는 테이블 스페이스가 롤 포워드 보류 상태에 놓이고 연속 로그 재생에서 무시됩니다. 인계 작업이 발생하는 경우, 응용프로그램은 이 테이블 스페이스를 사용할 수 없습니다.

인계 이전에 대기 시스템에서 문제점이 통지된 경우 해결책은 스토리지 문제를 처리하는 동안 대기 데이터베이스를 다시 설정하는 것입니다. 단계는 다음과 같습니다.

- 대기 데이터베이스 비활성화
- 대기 데이터베이스 삭제
- 연속 리스토어 및 롤 포워드를 위해 충분한 여유 공간을 가지고 있는 필요한 파일 시스템이 존재하는지 확인
- 기본 데이터베이스의 최근 백업을 사용하여 대기 시스템에서 데이터베이스 리스토어(또는 db2inidb 명령과 함께 분할 미러 또는 플래시 사본을 사용하여 다시 초기화). 자동 스토리지에 대해 기본 데이터베이스가 사용 가능한 경우, 리스토어 중에 스토리지 경로를 재정의하지 마십시오. 또한 테이블 스페이스 컨테이너는 리스토어의 일부로 경로 재지정하면 안됩니다.
- 대기 시스템에서 HADR 재시작

그러나 인계 발생 후에 대기 데이터베이스에 대해 문제점이 통지된 경우(또는 이때까지 스토리지 문제를 처리하지 않을 것을 선택한 경우) 해결책은 발생된 문제점의 유형에 따라 결정됩니다.

자동 스토리지에 대해 데이터베이스가 사용 가능하고 스페이스가 대기 데이터베이스와 연관되는 스토리지 경로에서 사용 불가능한 경우, 다음 단계를 따르십시오.

1. 파일 시스템을 확장하거나 파일 시스템에서 불필요한 비DB2 파일을 제거하여 스토리지 경로에서 스페이스가 사용 가능하도록 만드십시오.
2. 로그 끝까지 테이블 스페이스 롤 포워드를 수행하십시오.

로그 재생의 일부로 컨테이너 확장 또는 추가가 발생할 수 없는 경우, 필요한 백업 이미지 및 로그 파일 아카이브를 사용할 수 있으면 먼저 IGNORE ROLLFORWARD CONTAINER OPERATIONS 옵션과 함께 SET TABLESPACE CONTAINERS 문을 발행한 후 ROLLFORWARD 명령을 발행하여 테이블 스페이스를 복구할 수 있습니다.

기본 및 대기 데이터베이스의 데이터베이스 경로는 동일하지 않아도 됩니다. 상대 컨테이너 경로를 사용하는 경우 기본 및 대기 데이터베이스의 여러 절대 컨테이너 경로에 동일한 상대 경로가 맵핑될 수 있습니다. 따라서, 기본 및 대기 데이터베이스가 동일한 컴퓨터에 놓이면 모든 테이블 스페이스 컨테이너는 기본 및 대기에 대해 다른 경로에 맵핑되도록 상대 경로를 사용하여 정의해야 합니다.

HADR 및 네트워크 주소 변환(NAT) 지원

NAT는 사용 가능한 IPv4 주소의 부득이한 부족을 경감시키는 데 도움이 됩니다. IPv6 솔루션(구현하는 데 10년이 걸릴 수 있음)에 대한 빠른 수정으로 보입니다. NAT는 HADR 환경에서 지원됩니다.

HADR 및 NAT 지원

HADR은 항상 기본 및 대기 노드의 로컬 및 리모트 호스트 구성을 교차 점검합니다.

NAT 환경에서 호스트는 하나의 IP 주소에 의해 스스로에게 알려지지만 다른 호스트에는 다른 IP 주소로 알려집니다. 이 동작으로 인해 HADR 호스트 교차 점검이 실패합니다.

NAT 환경에서 이 상황을 피하기 위해 레지스트리 변수 **DB2_HADR_NO_IP_CHECK**를 ON으로 설정할 수 있습니다. 이것은 호스트 교차 점검을 생략하며, 기본 및 대기가 NAT 환경에서 연결할 수 있게 합니다.

NAT 환경에서 실행 중이 아닌 경우에는 **DB2_HADR_NO_IP_CHECK** 레지스트리 변수를 디폴트 설정인 OFF로 두는 것이 좋습니다. 교차 점검을 사용하지 않으면 HADR의 구성 유효성 확인이 약화됩니다.

고가용성 재해 복구(HADR)에 대한 제한사항

고가용성 재해 복구(HADR)를 사용하여 최적의 성능을 달성하려면, 고가용성 DB2 데이터베이스 솔루션을 설계할 때 HADR 제한사항을 고려하십시오.

다음 목록은 고가용성 재해 복구(HADR) 제한사항을 요약한 것입니다.

- HADR은 파티션된 데이터베이스 환경에서 지원되지 않습니다.
- 기본 및 대기 데이터베이스는 롤링 업그레이드 시 잠깐 동안을 제외하고 동일한 운영 체제 버전과 동일한 DB2 데이터베이스 시스템 버전을 가지고 있어야 합니다.
- 기본 및 대기 데이터베이스의 DB2 데이터베이스 시스템 소프트웨어 비트 크기는 동일해야 합니다(32 또는 64비트).
- 대기 데이터베이스에서의 읽기가 지원되지 않습니다. 클라이언트는 대기 데이터베이스에 연결할 수 없습니다.
- 로그 아카이브는 현재 기본 데이터베이스에서만 수행될 수 있습니다.
- 자체 성능 조정 메모리 관리자는 현재 기본 데이터베이스에서만 실행될 수 있습니다. 기본 데이터베이스가 시작되거나 대기 데이터베이스가 인계에 의해 기본 데이터베이스로 변환된 후, 첫 번째 클라이언트가 연결될 때까지 STMM EDU가 시작되지 않을 수도 있습니다.
- 백업 작업은 대기 데이터베이스에서 지원되지 않습니다.
- 데이터베이스 구성 매개변수와 복구 실행기록 파일에 대한 변경사항과 같은 로그되지 않는 작업은 대기 데이터베이스에 복제되지 않습니다.
- COPY NO 옵션이 지정된 로드 조작은 지원되지 않습니다.
- HADR은 데이터베이스 로그 파일에 대해 원시 입출력(직접 디스크 액세스)의 사용을 지원하지 않습니다. HADR이 START HADR 명령을 통해 시작되거나, 데이터베이스가 HADR이 구성된 상태로 활성화(재시작)되고 원시 로그가 발견되는 경우 연관된 명령이 실패합니다.
- 1단계 커미트의 경우, HADR 데이터베이스는 페더레이티드 서버(트랜잭션 관리 프로그램)나 데이터 소스(자원 관리자)로서 작동할 수 있습니다. 2단계 커미트의 경우 HADR 데이터베이스는 데이터 소스로서만 작동할 수 있습니다.

고가용성을 위한 유지보수 스케줄링

DB2 데이터베이스 솔루션은 일반 유지보수가 필요합니다. 소프트웨어 또는 하드웨어 업그레이드, 데이터베이스 성능 조정, 데이터베이스 백업, 비즈니스 목적을 위한 통계 수집 및 모니터링 같은 유지보수를 수행해야 합니다. 데이터베이스 솔루션의 사용 가능성에 대한 이러한 유지보수 활동의 영향을 최소화해야 합니다.

유지보수 활동을 스케줄할 수 있기 전에 데이터베이스 솔루션에 수행해야 하는 유지보수 활동을 식별해야 합니다.

유지보수를 스케줄하려면 다음 단계를 수행하십시오.

1. 낮은 데이터베이스 활동 기간을 식별하십시오.

   사용량이 작은 시간(가장 작은 사용자 응용프로그램이 데이터베이스 시스템에 대한 요청을 작성 중인 기간)에 유지보수 활동을 스케줄하는 것이 가장 좋습니다. 작성 중인 비즈니스 응용프로그램의 유형에 따라서는 데이터베이스 시스템에 액세스하는 사용자 응용프로그램이 없는 기간일 수도 있습니다.
2. 다음에 따라서 사용자가 수행해야 하는 유지보수 활동을 구분하십시오.
   - 유지보수를 자동화할 수 있음
   - 유지보수를 수행하는 동안 데이터베이스 솔루션을 오프라인으로 해야 함
   - 데이터베이스 솔루션이 온라인인 중에 유지보수를 수행할 수 있음
3. 자동화할 수 있는 유지보수 활동의 경우, 다음 메소드 중 하나를 사용하여 자동화된 유지보수를 구성하십시오.
   - auto_maint 구성 매개변수 사용
   - 자동 유지보수 구성 마법사 사용
   - AUTOMAINT_SET_POLICY 및 AUTOMAINT_SET_POLICYFILE이라는 시스템 스토어드 프로시저 중 하나 사용
4. 사용자가 수행해야 하는 유지보수 활동에서 데이터베이스 서버가 오프라인이어야 하는 경우, 사용량이 낮은 시간에 오프라인 유지보수 활동을 스케줄하십시오.
5. 데이터베이스 서버가 온라인인 중에 수행할 수 있는 유지보수 활동의 경우,
   - 온라인 유지보수 활동 실행의 사용 가능성 영향을 식별하십시오.
   - 데이터베이스 시스템의 사용 가능성에 대한 유지보수 활동 실행의 영향을 최소화하도록 온라인 유지보수 활동을 스케줄하십시오.

   예를 들어, 사용량이 낮을 때 온라인 유지보수 활동을 스케줄하고 조절 메커니즘을 사용하여 유지보수 활동이 사용하는 시스템 자원의 양을 분산하십시오.

SYSPROC.AUTOMAINT_GET_POLICY 또는 SYSPROC.AUTOMAINT_GET_POLICYFILE을 사용한 자동화된 유지보수 규정 정보 수집

시스템 스토어드 프로시저 AUTOMAINT_GET_POLICY 및 AUTOMAINT_GET_POLICYFILE을 사용하여 데이터베이스에 대해 구성된 자동화된 유지보수 규정을 검색할 수 있습니다.

데이터베이스에 대한 자동화된 유지보수 규정을 검색하려면 다음 단계를 수행하십시오.

1. 데이터베이스에 연결
2. AUTOMAINT_GET_POLICY 또는 AUTOMAINT_GET_POLICYFILE 호출
   - AUTOMAINT_GET_POLICY에 필요한 매개변수는 다음과 같습니다.

a. 정보를 리턴할 자동화된 유지보수 활동의 유형을 지정하는 유지보수 유형.
b. 프로시저가 XML 형식으로 자동화된 유지보수 규정 정보를 리턴하는 BLOB에 대한 포인터.

- AUTOMAINT_GET_POLICYFILE에 필요한 매개변수는 다음과 같습니다.
  a. 정보를 리턴할 자동화된 유지보수 활동의 유형을 지정하는 유지보수 유형.
  b. 프로시저가 자동화된 유지보수 규정 정보를 인쇄할 파일의 이름.

유효한 유지보수 유형은 다음과 같습니다.

- AUTO_BACKUP - 자동 백업
- AUTO_REORG - 자동 테이블 및 인덱스 재구성
- AUTO_RUNSTATS - 자동 테이블 Runstats 조작
- MAINTENANCE_WINDOW - 유지보수 기간

SYSPROC.AUTOMAINT_SET_POLICY 또는 SYSPROC.AUTOMAINT_SET_POLICYFILE을 사용한 자동화된 유지보수 규정 구성

시스템 스토어드 프로시저 AUTOMAINT_SET_POLICY 및 AUTOMAINT_SET_POLICYFILE을 사용하여 데이터베이스에 대한 자동화된 유지보수 규정을 구성할 수 있습니다.

데이터베이스에 대한 자동화된 유지보수 규정을 구성하려면 다음 단계를 수행하십시오.

1. 데이터베이스에 연결
2. AUTOMAINT_SET_POLICY 또는 AUTOMAINT_SET_POLICYFILE 호출
   - AUTOMAINT_SET_POLICY에 필요한 매개변수는 다음과 같습니다.
     a. 구성할 자동화된 유지보수 활동의 유형을 지정하는 유지보수 유형.
     b. XML 형식으로 자동화된 유지보수 규정을 지정하는 BLOB에 대한 포인터.
   - AUTOMAINT_SET_POLICYFILE에 필요한 매개변수는 다음과 같습니다.
     a. 구성할 자동화된 유지보수 활동의 유형을 지정하는 유지보수 유형.
     b. 자동화된 유지보수 규정을 지정하는 XML 파일의 이름.

   유효한 유지보수 유형은 다음과 같습니다.

   - AUTO_BACKUP - 자동 백업
   - AUTO_REORG - 자동 테이블 및 인덱스 재구성
   - AUTO_RUNSTATS - 자동 테이블 Runstats 조작
   - MAINTENANCE_WINDOW - 유지보수 기간

AUTOMAINT_SET_POLICY 또는 AUTOMAINT_SET_POLICYFILE에 대한 자동화 유지보수 규정 스펙 XML 샘플

자동화된 유지보수 규정을 지정하기 위해 AUTOMAINT_SET_POLICY 또는 AUTOMAINT_SET_POLICYFILE 중 어느 것을 사용하는지 여부에 관계없이, XML을 사용하여 규정을 지정해야 합니다. SQLLIB/samples/automaintcfg에는 XML로 자동화된 유지보수 규정을 지정하는 방법을 설명하는 샘플 파일이 있습니다.

시스템 스토어드 프로시저 AUTOMAINT_SET_POLICY로 전달하는 두 번째 매개변수는 원하는 자동화된 유지보수 규정을 지정하는 XML을 포함하는 BLOB입니다. 시스템 스토어드 프로시저 AUTOMAINT_SET_POLICYFILE로 전달하는 두 번째 매개변수는 원하는 자동화된 유지보수 규정을 지정하는 XML 파일의 이름입니다. AUTOMAINT_SET_POLICY로 전달하는 BLOB에서 유효한 XML 요소는 AUTOMAINT_SET_POLICYFILE로 전달하는 XML 파일에서 유효한 요소입니다.

샘플 디렉토리 SQLLIB/samples/automaintcfg에는 자동화된 유지보수 규정 스펙 예를 포함하는 네 개의 XML 파일이 있습니다.

DB2MaintenanceWindowPolicySample.xml

유지보수 기간 지정에 대해 보여줍니다. 이 창에서 데이터베이스 관리 프로그램이 자동화된 유지보수를 스케줄링해야 합니다.

DB2AutoBackupPolicySample.xml

데이터베이스 관리 프로그램이 자동 백업을 수행해야 하는 방법을 지정하는 것을 보여줍니다.

DB2AutoReorgPolicySample.xml

데이터베이스 관리 프로그램이 자동 테이블 및 색인 재구성을 수행해야 하는 방법을 지정하는 것을 보여줍니다(다차원적으로 클러스터된(MDC) 테이블에서 Extent를 재개하는 것을 포함하여).

DB2DefaultAutoRunstatsPolicySample.xml

데이터베이스 관리 프로그램이 자동 테이블 runstat 작업을 수행해야 하는 방법을 지정하는 것을 보여줍니다.

이 파일에서 XML을 복사하고 시스템의 요구사항에 따라 XML을 수정하여 사용자 자신의 자동화된 유지보수 규정 스펙 XML을 작성할 수 있습니다.

데이터베이스 로깅 옵션 구성

사용할 로깅의 유형, 로그 파일 크기 및 로그 파일이 저장되어야 하는 위치 같은 데이터베이스에 대한 데이터 로깅 옵션을 지정하려면 데이터베이스 로깅 구성 매개변수를 사용하십시오.

데이터베이스 로깅 옵션을 구성하려면 SYSADM, SYSCTRL 및 SYSMAINT 권한이 있어야 합니다.

명령행 처리기(CLP)에서 UPDATE DATABASE CONFIGURATION 명령을 사용하거나 제어 센터의 데이터베이스 로깅 구성 마법사 GUI를 통하거나 db2CfgSet API를 호출하여 데이터베이스 로깅 옵션을 구성할 수 있습니다.

- 명령행 처리기에서 UPDATE DATABASE CONFIGURATION 명령을 사용하여 데이터베이스 로깅 옵션을 구성하려면 다음을 수행하십시오.
  1. 순환 로깅 또는 아카이브 로깅을 사용하려는지 여부를 지정하십시오. 순환 로깅을 사용하려는 경우 LOGARCHMETH1 및 LOGARCHMETH2 데이터베이스 구성 매개변수가 OFF로 설정되어야 합니다. 이것이 디폴트 설정입니다. 아카이브 로깅을 사용하려면 이들 데이터베이스 구성 매개변수 중 최소한 하나를 OFF가 아닌 값으로 설정해야 합니다. 예를 들어 아카이브 로깅을 사용하기 원하고 아카이브된 로그를 디스크에 저장하려는 경우 다음을 발행하십시오.

     ```
     db2 update db configuration for mydb using logarchmeth1
        disk:/u/dbuser/archived_logs
     ```

     아카이브된 로그는 /u/dbuser/archived_logs 디렉토리에 위치합니다.
  2. 필요한 대로 기타 데이터베이스 로그 구성 매개변수에 대한 값을 지정하십시오. 다음은 데이터베이스 로깅에 대한 추가 구성 매개변수입니다.
     - ARCHRETRYDELAY
     - BLK_LOG_DSK_FUL
     - FAILARCHPATH
     - LOGARCHOPT1
     - LOGARCHOPT2
     - LOGBUFSZ
     - LOGFILSIZ
     - LOGPRIMARY
     - LOGRETAIN
     - LOGSECOND
     - MAX_LOG
     - MIRRORLOGPATH
     - NEWLOGPATH
     - MINCOMMIT
     - NUMARCHRETRY
     - NUM_LOG_SPAN
     - OVERFLOWLOGPATH

– USEREXIT

이러한 데이터베이스 로깅 구성 매개변수에 대한 자세한 정보는 『데이터베이스 로깅을 위한 구성 매개변수』를 참조하십시오.

- 데이터베이스 로깅 구성 마법사를 열려면 다음을 수행하십시오.
  1. 제어 센터에서 로깅을 설정하려는 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오.
  2. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 데이터베이스 로깅 구성을 선택하십시오. 데이터베이스 로깅 구성 마법사가 열립니다.
- 제어 센터에 있는 온라인 도움말 기능을 통해 세부사항 정보가 제공됩니다.

데이터베이스 로깅을 위한 구성 매개변수

고가용성 전략의 핵심 요소는 데이터베이스 로깅입니다. 데이터베이스 로그를 사용하여 트랜잭션 정보를 기록하고, 기본 및 보조 또는 대기 데이터베이스를 동기화하고, 실패한 기본 데이터베이스에 대해 인계된 보조 데이터베이스를 롤 포워드할 수 있습니다. 이러한 데이터베이스 로깅 활동을 사용자 요구에 따라 구성하려면 다양한 데이터베이스 구성 매개변수를 설정해야 합니다.

아카이브 재시도 대기 시간(archretrydelay)

이전 시도가 실패한 후 로그 파일을 아카이브하려는 시도 사이에 기다릴 시간(초)을 지정합니다. 디폴트값은 20입니다.

로그 디스크의 블록 가득참(blk_log_dsk_ful)

이 구성 매개변수는 DB2가 사용 중인 로그 경로에 새 로그 파일을 작성할 수 없는 경우 디스크 가득참 오류가 발생하지 않도록 예방하기 위해 설정할 수 있습니다. 대신, DB2는 성공할 때까지 5분마다 로그 파일을 작성하려고 시도합니다. 각 시도 후, DB2는 관리 통지 로그에 메시지를 기록합니다. 로그 디스크 가득참 조건 때문에 응용프로그램이 정지 중인지 확인하는 유일한 방법은 관리 통지 로그를 모니터하는 것입니다. 테이블 데이터를 갱신하려고 시도하는 사용자 응용프로그램은 로그 파일이 작성될 때까지 트랜잭션을 커미트할 수 없습니다. 읽기 전용 쿼리는 직접 영향을 받지 않을 수 있습니다. 그러나 쿼리가 갱신 요청이 잠근 데이터 또는 갱신 응용프로그램이 버퍼 풀에서 고정한 데이터 페이지에 액세스해야 하는 경우, 읽기 전용 쿼리도 정지한 것처럼 보일 수 있습니다.

*blk_log_dsk_ful*을 YES로 설정하면 DB2에서 로그 디스크 가득참 오류가 발생할 때 응용프로그램이 정지하게 됩니다. 그러면 오류를 해결할 수 있으며 트랜잭션이 계속될 수 있습니다. 이전 로그 파일을 다른 파일 시스템으로 이동하거나 파일 시스템의 크기를 늘려서 디스크 가득참 상황을 해결하면 응용프로그램 정지를 완료할 수 있습니다.

blk_log_dsk_ful이 NO로 설정된 경우 로그 디스크 가득참 오류를 수신하는 트랜잭션이 실패하고 롤백됩니다. 일부 경우에는 트랜잭션이 로그 디스크 가득참 오류를 유발하는 경우 데이터베이스가 중지됩니다.

장애 복구 아카이브 경로(failarchpath)

지정된 로그 아카이브 메소드가 실패하는 경우 아카이브 로그 파일에 대한 대체 디렉토리를 지정합니다. 이 디렉토리는 로그 파일이 이 디렉토리에서 로그 아카이브 메소드로 이동될 시간에 실패한 로그 아카이브 메소드가 다시 사용 가능하게 될 때까지 로그 파일에 대한 임시 스토리지 영역입니다. 로그 파일을 이 임시 위치로 이동하여 로그 디렉토리 가득참 상황을 피할 수 있습니다. 이 매개변수는 완전한 기존 디렉토리여야 합니다.

로그 아카이브 메소드 1(logarchmeth1), 로그 아카이브 메소드 2(logarchmeth2)

이들 매개변수는 데이터베이스 관리 프로그램이 사용 중인 로그 경로가 아닌 위치에 로그 파일을 아카이브하게 만듭니다. 이들 매개변수가 둘 다 지정되는 경우 각 로그 파일이 두 번 아카이브됩니다. 이것은 두 개의 다른 위치에 아카이브된 로그 파일의 두 사본을 갖게 됨을 의미합니다.

이들 매개변수에 대한 올바른 값에는 미디어 유형 및 일부 경우에는 대상 필드가 포함됩니다. 값을 구분하려면 콜론(:)을 사용하십시오. 가능한 값은 다음과 같습니다.

OFF 로그 아카이브 메소드가 사용되지 않도록 지정합니다. *logarchmeth1*과 *logarchmeth2*가 둘 다 OFF로 설정되는 경우, 데이터베이스는 순환 로그를 사용하는 것으로 간주되며 롤 포워드 복구 가능하지 않습니다. 이것이 디폴트입니다.

LOGRETAIN

이 값은 *logarchmeth1*에만 사용할 수 있으며 *logretain* 구성 매개변수를 RECOVERY로 설정하는 것과 같습니다. 이 값을 지정하면 *logretain* 구성 매개변수는 자동으로 갱신됩니다.

USEREXIT

이 값은 *logarchmeth1*에만 유효하며 *userexit* 구성 매개변수를 ON으로 설정하는 것과 같습니다. 이 값을 지정하면 *userexit* 구성 매개변수는 자동으로 갱신됩니다.

DISK 이 값은 뒤에 콜론(:)이 오고, 로그 파일이 아카이브된 기존의 완전한 경로 이름이 나와야 합니다. 예를 들어, *logarchmeth1*을 DISK:/u/dbuser/archived_logs로 설정한 경우 아카이브 로그 파일은 /u/dbuser/archived_logs라는 디렉토리에 놓이게 됩니다.

주: 테이프에 아카이브하는 경우에는 db2tapemgr 유틸리티를 사용하여 로그 파일을 저장하고 검색할 수 있습니다.

TSM 추가적인 구성 매개변수를 사용하지 않고 이 값을 지정하면 로그 파일

이 디폴트 관리 클래스를 사용하여 로컬 TSM 서버에 아카이브되어야 함을 표시합니다. 다음에 콜론(:)과 TSM 관리 클래스가 있는 경우, 로그 파일은 지정된 관리 클래스를 사용하여 아카이브됩니다.

VENDOR

로그 파일을 아카이브하는 데 사용할 벤더 라이브러리를 지정합니다. 이 값 다음에는 콜론(:)과 라이브러리 이름이 있어야 합니다. 라이브러리에서 제공된 API는 벤더 제품에 대해 백업 및 리스토어 API를 사용해야 합니다.

주:

1. *logarchmeth1* 또는 *logarchmeth2*가 OFF가 아닌 값으로 설정되는 경우, 데이터베이스는 롤 포워드 복구에 대해 구성됩니다.
2. *userexit* 또는 *logretain* 구성 매개변수를 갱신하는 경우 *logarchmeth1*도 자동으로 갱신되며, 그 반대도 마찬가지입니다. 그러나 *userexit* 또는 *logretain*을 사용 중인 경우 *logarchmeth2*를 OFF로 설정해야 합니다.

로그 아카이브 옵션 1(logarchopt1), 로그 아카이브 옵션 2(logarchopt2)

TSM 서버나 벤더 API로 전달되는 문자열을 지정합니다. TSM의 경우 이 필드는 데이터베이스가 다른 TSM 노드에서 또는 다른 TSM 사용자에 의해 생성된 로그를 검색하도록 허용하는 데 사용됩니다. 문자열은 다음 형식으로 제공되어야 합니다.

```
"-fromnode=nodename -fromowner=ownername"
```

여기서 nodename은 원래 로그 파일을 아카이브한 TSM 노드의 이름이고, ownername은 원래 로그 파일을 아카이브한 TSM 사용자의 이름입니다. 각 로그 아카이브 옵션 필드는 로그 아카이브 메소드 중 하나에 해당합니다. *logarchopt1*은 *logarchmeth1*과 함께 사용되고, *logarchopt2*는 *logarchmeth2*와 함께 사용됩니다.

로그 버퍼(logbufsz)

이 매개변수를 사용하여 이러한 레코드를 디스크에 기록하기 전에 로그 레코드에 대한 버퍼로 사용할 메모리량을 지정할 수 있습니다. 로그 레코드는 다음 이벤트 중 하나가 발생할 때 디스크에 기록됩니다.

- 트랜잭션 커미트
- 로그 버퍼가 가득참
- 일부 다른 내부 데이터베이스 관리 프로그램 이벤트가 발생합니다.

로그 레코드가 덜 자주 디스크에 기록되고 매번 더 많은 레코드가 기록되기 때문에, 로그 버퍼 크기를 늘리면 로깅과 연관된 입출력(I/O) 활동이 더욱 효율적이 되게 합니다. 그러나 로그 버퍼 크기 값이 크면 복구가 더 오래 걸릴 수 있습니다.

로그 파일 *크기*(logfilsiz)

이 매개변수는 각 구성된 로그의 *크기*를 4KB 페이지 수로 지정합니다.

사용자가 구성할 수 있는 전체 사용 중인 로그 스페이스에 대한 1024GB 논리적 한계가 있습니다. 이 한계는 각 로그 파일에 대한 상한(4GB) 및 1차 및 2차 로그 파일을 합친 최대 수(256)의 결과입니다.

로그 파일의 *크기*는 성능에 직접적인 관계를 갖습니다. 한 로그에서 다른 로그로 전환하기 위한 성능 비용이 있습니다. 따라서 순수한 성능 perspective에서 로그 파일이 클수록 더 좋습니다. 이 매개변수는 또한 아카이브를 위한 로그 파일 *크기*를 표시합니다. 이 경우에 더 큰 로그 파일 *크기*는 실패할 가능성을 크게 하거나 로그 제공 시나리오에서 대기 시간을 유발할 수 있으므로 더 큰 로그 파일 *크기*가 반드시 더 좋은 것은 아닙니다. 사용 중인 로그 스페이스를 고려할 때 더 작은 로그 파일을 더 많이 갖는 것이 좋을 수 있습니다. 예를 들어 2개의 매우 큰 로그 파일이 있고 트랜잭션이 로그 파일 하나의 끝에 가까워지기 시작하는 경우 로그 스페이스의 절반만 사용 가능한 상태로 남습니다.

데이터베이스가 비활성화(데이터베이스에 대한 모든 연결이 종료됨)될 때마다, 현재 기록되고 있는 로그 파일이 절단됩니다. 그러므로 데이터베이스가 자주 비활성화되고 있는 경우에는 큰 로그 파일 *크기*를 선택하지 않는 것이 더 좋습니다. DB2가 큰 파일이 절단되도록 하기 위해서만 큰 파일을 작성하기 때문입니다. ACTIVATE DATABASE 명령을 사용하여 이 비용을 회피할 수 있으며, 버퍼 풀이 준비되도록 하는 것도 성능에 도움이 됩니다.

데이터베이스를 열 때 처리 시간을 최소화하기 위해 데이터베이스를 계속 열어두는 응용프로그램이 있다고 가정할 때, 로그 파일 *크기*는 아카이브된 로그 사본을 오프라인으로 만드는 데 걸리는 시간에 의해 판별되어야 합니다.

로그 파일 손실을 최소화하는 것도 로그 *크기*를 설정할 때 중요한 고려사항입니다. 아카이브는 전체 로그를 취합니다. 하나의 큰 로그를 사용하는 경우 아카이브 사이의 시간이 늘어납니다. 로그를 포함하는 매체가 실패하는 경우 일부 트랜잭션 정보가 유실될 수 있습니다. 로그 *크기*를 줄이면 아카이브 빈도는 늘어나지만 한 번의 유실 전에 더 작은 로그가 사용될 수 있으므로 매체 실패의 경우에 정보 손실의 양을 줄일 수 있습니다.

로그 보유(logretain)

이 구성 매개변수는 *logarchmeth1*으로 교체되었습니다. DB2의 이전 버전과의 호환성을 위해 아직 지원됩니다.

logretain이 RECOVERY로 설정되면 아카이브된 로그는 데이터베이스 로그 경로 디렉토리에 보존되며 데이터베이스는 복구 가능한 것으로 간주되므로 롤 포워드 복구가 사용 가능함을 의미합니다.

주: *logretain* 데이터베이스 구성 매개변수의 디폴트값은 롤 포워드 복구를 지원하지 않습니다. 롤 포워드 복구를 사용하려는 경우 이 매개변수의 값을 변경해야 합니다.

트랜잭션당 최대 로그(max_log)

이 매개변수는 한 트랜잭션에서 사용할 수 있는 1차 로그 스페이스의 백분율을 표시합니다. 값은 logprimary 구성 매개변수에 대해 지정된 값의 백분율입니다.

값이 0으로 설정되면 트랜잭션이 사용할 수 있는 전체 1차 로그 스페이스의 백분율에 대한 제한은 없습니다. 응용프로그램이 *max_log* 구성을 위반하는 경우 응용프로그램은 강제로 데이터베이스에서 연결이 끊어지며 트랜잭션은 롤백되고 오류 SQL1224N이 리턴됩니다.

DB2_FORCE_APP_ON_MAX_LOG 레지스트리 변수를 *FALSE*로 설정하여 이 동작을 겹쳐쓸 수 있습니다. 이것은 *max_log* 구성을 위반하는 트랜잭션이 실패하고 오류 SQL0964N을 리턴하게 합니다. 응용프로그램은 여전히 작업 단위(UOW)의 이전 명령문에 의해 완료된 작업을 커미트할 수 있거나 완료된 작업을 롤백하여 작업 단위를 실행 취소할 수 있습니다.

이 매개변수는 num_log_span 구성 매개변수와 함께 무한 활성 로그 스페이스가 사용 가능할 때 유용할 수 있습니다. 무한 로깅이 사용되는 경우(즉, logsecond가 -1) 트랜잭션은 로그 파일 수의 상한(logprimary + logsecond)으로 제한되지 않습니다. logprimary의 값에 도달할 때 DB2는 트랜잭션을 실패하는 대신 사용 중인 로그를 아카이브하기 시작합니다. 이것은 예를 들어 커미트되지 않은 채로 있는 장기 실행 중인 트랜잭션(아마도 잘못된 응용프로그램에 의해)이 있는 경우 문제점을 유발할 수 있습니다. 이 경우 활성 로그 스페이스는 계속 성장하며, 그러면 응급 복구 성능이 열악해질 수 있습니다. 이를 예방하기 위해 max_log 또는 num_log_span 구성 매개변수 중 하나 또는 둘 다에 대한 값을 지정할 수 있습니다.

주: 다음 DB2 명령은 *max_log* 구성 매개변수에 의해 부과되는 한계로부터 제외됩니다. ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE 및 ROLLFORWARD DATABASE.

미러 로그 경로(mirrorlogpath)

1차 로그 경로의 로그를 디스크 오류 또는 우발적인 삭제로부터 보호하기 위해 동일한 로그 세트가 2차(미러) 로그 경로에 유지보수되도록 지정할 수 있습니다. 이렇게 하려면 이 구성 매개변수의 값을 변경하여 다른 디렉토리를 가리키십시오. 데이터베이스가 롤 포워드 복구에 대해 구성되는 경우 현재 미러 로그 경로 디렉토리에 저장되는 사용 중인 로그는 새 위치로 이동되지 않습니다.

로그 경로 위치를 변경할 수 있기 때문에, 롤 포워드 복구를 위해 필요한 로그가 다른 디렉토리에 존재할 수 있습니다. 롤 포워드 조작 중에 이 구성 매개변수의 값을 변경하여 여러 위치에 있는 로그에 액세스할 수 있습니다.

로그 위치를 추적해야 합니다.

데이터베이스가 일관성 있는 상태에 있을 때까지 변경이 적용되지 않습니다. 구성 매개변수 *database_consistent*는 데이터베이스의 상태를 리턴합니다.

이 구성 매개변수를 끄려면 값을 DEFAULT로 설정하십시오.

주:

1. 이 구성 매개변수는 1차 로그 경로가 원시 디바이스인 경우에는 지원되지 않습니다.
2. 이 매개변수에 지정되는 값은 원시 디바이스일 수 없습니다.

새 로그 경로(newlogpath)

데이터베이스 로그는 처음에 데이터베이스 디렉토리의 서브디렉토리인 SQLOGDIR에 작성됩니다. 이 구성 매개변수의 값을 다른 디렉토리나 디바이스를 가리키도록 변경하여 사용 중인 로그 또는 미래의 아카이브 로그가 위치하는 위치를 변경할 수 있습니다. 데이터베이스가 롤 포워드 복구에 대해 구성되는 경우 현재 데이터베이스 로그 경로 디렉토리에 저장되는 사용 중인 로그는 새 위치로 이동되지 않습니다.

로그 경로 위치를 변경할 수 있기 때문에, 롤 포워드 복구를 위해 필요한 로그가 다른 디렉토리 또는 다른 디바이스에 존재할 수 있습니다. 롤 포워드 조작 중에 이 구성 매개변수의 값을 변경하여 여러 위치에 있는 로그에 액세스할 수 있습니다.

로그 위치를 추적해야 합니다.

데이터베이스가 일관성 있는 상태에 있을 때까지 변경이 적용되지 않습니다. 구성 매개변수 *database_consistent*는 데이터베이스의 상태를 리턴합니다.

그룹화할 커미트 수(mincommit)

이 매개변수를 사용하면 최소 커미트 수를 수행할 때까지 디스크에 로그 레코드 기록을 지연시킬 수 있습니다. 이 지연은 로그 레코드 쓰기와 연관된 데이터베이스 관리 프로그램 오버헤드를 줄이는 데 도움이 되며, 결과적으로 데이터베이스에 대해 실행 중인 응용프로그램이 여러 개이고 매우 짧은 시간 안에 응용프로그램이 많은 커미트를 요청할 때 성능을 향상시킬 수 있습니다.

커미트 그룹화는 이 매개변수의 값이 1보다 큰 경우와 데이터베이스에 연결된 응용프로그램 수가 이 매개변수 값보다 큰 경우에만 발생합니다. 커미트 그룹화가 사용 중인 경우, 1초가 경과되거나 커미트 요청 수가 이 매개변수 값과 동일할 때까지 응용프로그램 커미트 요청이 보류됩니다.

오류 시 아카이브 재시도 수(numarchretry)

로그 파일이 *failarchpath* 구성 매개변수에 의해 지정되는 경로에 아카이브되기 전에 지정된 로그 아카이브 메소드를 사용하여 로그 파일을 아카이브하려는 시도 수를 지정합니다. 이 매개변수는 *failarchpath* 구성 매개변수가 설정되는 경우에만 사용할 수 있습니다. 디폴트값은 5입니다.

로그 스팬 수(num_log_span)

이 매개변수는 활성 트랜잭션이 걸칠 수 있는 사용 중인 로그 파일의 수를 표시합니다. 값이 0으로 설정된 경우에는 단일 트랜잭션이 걸칠 수 있는 로그 파일의 수에 제한이 없습니다.

응용프로그램이 *num_log_span* 구성을 위반하는 경우 응용프로그램은 강제로 데이터베이스로부터 연결이 끊어지고 오류 SQL1224N이 리턴됩니다.

이 매개변수는 max_log 구성 매개변수와 함께 무한 활성 로그 스페이스가 사용 가능할 때 유용할 수 있습니다. 무한 로깅이 사용되는 경우(즉, logsecond가 -1) 트랜잭션은 로그 파일 수의 상한(logprimary + logsecond)으로 제한되지 않습니다. logprimary의 값에 도달할 때 DB2는 트랜잭션을 실패하는 대신 사용 중인 로그를 아카이브하기 시작합니다. 이것은 예를 들어 커미트되지 않은 채로 있는 장기 실행 중인 트랜잭션(아마도 잘못된 응용프로그램에 의해)이 있는 경우 문제점을 유발할 수 있습니다. 이 경우 활성 로그 스페이스는 계속 성장하며, 그러면 응급 복구 성능이 열악해질 수 있습니다. 이를 예방하기 위해 max_log 또는 num_log_span 구성 매개변수 중 하나 또는 둘 다에 대한 값을 지정할 수 있습니다.

주: 다음 DB2 명령은 *num_log_span* 구성 매개변수에 의해 부과되는 한계로부터 제외됩니다. ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE 및 ROLLFORWARD DATABASE.

오버플로우 로그 경로(overflowlogpath)

이 매개변수는 로깅 요구사항에 따라 여러 기능에서 사용됩니다. DB2가 롤 포워드 조작을 위해 필요한 로그 파일을 찾을 위치를 지정할 수 있습니다. ROLLFORWARD 명령의 OVERFLOW LOG PATH 옵션과 비슷합니다. 그러나 모든 ROLLFORWARD 명령에 대해 OVERFLOW LOG PATH 옵션을 지정하는 대신 이 구성 매개변수를 한 번 설정할 수 있습니다. 둘 다 사용되는 경우 해당 롤 포워드 조작에 대해 OVERFLOW LOG PATH 옵션이 *overflowlogpath* 구성 매개변수를 겹쳐씁니다.

*logsecond*가 -1로 설정되는 경우, DB2가 아카이브에서 검색되는 사용 중인 로그 파일을 저장할 디렉토리를 지정할 수 있습니다. (사용 중인 로그 파일이 더 이상 사용 중인 로그 경로에 있지 않은 경우, 롤백 조작을 위해 사용 중인 로그 파일을 검색해야 합니다.)

*overflowlogpath*가 지정되지 않은 경우 DB2는 로그 파일을 사용 중인 로그 경로에서 검색합니다. 이 매개변수를 지정하면 DB2가 검색된 로그 파일을 저장하기 위한 추가 자원을 제공할 수 있습니다. 다른 디스크로 입출력 비용을 전개하고 사용 중인 로그 경로에 더 많은 로그 파일이 저장되도록 허용할 수 있다는 이점이 있습니다.

예를 들어 복제를 위해 db2ReadLog API를 사용 중인 경우 *overflowlogpath*를 사용하여 DB2가 이 API에 필요한 로그 파일을 검색할 위치를 지정할 수 있습니다. 로그 파일을 찾을 수 없는 경우(사용 중인 로그 경로 또는 오버플로우 로그 경로에서) 및 데이터베이스가 *userexit* 사용 가능으로 구성된 경우, DB2는 로그 파일을 검색합니다. 또한 이 매개변수를 사용하여 DB2가 검색된 로그 파일을 저장할 디렉토리를 지정할 수도 있습니다. 사용 중인 로그 경로에서 입출력 비용을 줄이고 사용 중인 로그 경로에 더 많은 로그 파일을 저장할 수 있는 이점이 있습니다.

사용 중인 로그 경로에 대해 원시 디바이스를 구성한 경우, *logsecond*를 -1로 설정하려는 경우 또는 db2ReadLog API를 사용하려는 경우 *overflowlogpath*를 구성해야 합니다.

*overflowlogpath*를 설정하려면 최대 242바이트의 문자열을 지정하십시오. 이 문자열은 경로 이름을 가리켜야 하며, 상대 경로 이름이 아니라 완전한 경로여야 합니다. 경로 이름은 원시 디바이스가 아니라 디렉토리여야 합니다.

주: 파티션된 데이터베이스 환경에서 데이터베이스 파티션 번호가 자동으로 경로에 추가됩니다. 여러 논리 노드 구성에서 경로의 고유성을 유지하기 위해 이러한 작업이 수행됩니다.

1차 로그(logprimary)

이 매개변수는 작성될 *logfilsiz* 크기의 1차 로그 수를 지정합니다.

1차 로그는 비어 있든 가득 찼든 동일한 양의 디스크 스페이스가 필요합니다. 따라서, 필요한 것보다 많은 로그를 구성하면 디스크 스페이스를 불필요하게 사용합니다. 너무 적은 로그를 구성하면 로그 가득참 조건이 발생할 수 있습니다. 구성할 로그 수를 선택할 때 각 로그를 작성하는 크기 및 응용프로그램이 로그 가득참 조건을 처리할 수 있는지 여부를 고려해야 합니다. 사용 중인 로그 스페이스에 대한 전체 로그 파일 크기 한계는 256GB입니다.

롤 포워드 복구에 대해 기존 데이터베이스를 사용 중인 경우, 1차 로그 수를 1차 및 2차 로그 수의 합에 1을 더한 값으로 변경하십시오. 추가 정보가 롤 포워드 복구에 사용 가능한 데이터베이스의 LONG VARCHAR 및 LOB 필드에 대해 로그됩니다.

2차 로그(logsecond)

이 매개변수는 필요한 경우 복구를 위해 작성되고 사용되는 2차 로그 파일 수를 지정합니다.

1차 로그 파일이 가득 차면, 2차 로그 파일(*크기*는 *logfilsiz*)이 필요할 때마다 하나씩, 이 매개변수로 지정되는 최대 숫자까지 할당됩니다. 이 매개변수가 -1로 설정되는 경우 데이터베이스는 무한 활성 로그 스페이스를 사용하여 구성됩니다. 데이터베이스에서 실행 중인 인플라이트(inflight) 트랜잭션의 *크기* 또는 수에는 제한이 없습니다. 무한 활성 로깅은 일반적으로 1차 로그에 할당하는 것보다 많은 로그 스페이스가 필요한 큰 작업을 수용해야 하는 환경에서 유용합니다.

주:

1. *logsecond*를 -1로 설정하려면 로그 아카이브가 사용 가능해야 합니다.
2. 이 매개변수가 -1로 설정되는 경우, DB2가 아카이브된 로그 파일을 검색해야 할 수 있으므로 응급 복구 시간이 늘어날 수 있습니다.

User exit(userexit)

이 구성 매개변수는 *logarchmeth1*으로 교체되었습니다. DB2의 이전 버전과의 호환성을 위해 아직 지원됩니다.

이 매개변수는 데이터베이스 관리 프로그램이 로그 아카이브 및 검색을 위해 User Exit 프로그램을 호출하게 합니다. 로그 파일은 사용 중인 로그 경로와는 다른 위치에 아카이브됩니다. *userexit*가 ON으로 설정되는 경우 롤 포워드 복구가 사용 가능합니다.

오프라인 아카이브된 로그를 저장하는 데 사용하는 디바이스의 데이터 전송 속도와, 사본을 작성하는 데 사용되는 소프트웨어가 최소한 데이터베이스 관리 프로그램이 로그에 데이터를 기록하는 평균 속도와 일치해야 합니다. 전송 속도가 생성되는 새 로그 데이터와 계속 접속을 유지할 수 없는 경우, 로깅 활동이 충분히 긴 시간 동안 계속되면 디스크 스페이스가 부족할 수 있습니다. 디스크 스페이스가 부족해지는 데 걸리는 시간은 사용 가능한 디스크 스페이스의 양에 의해 판별됩니다. 디스크 스페이스 부족하면 데이터베이스 처리가 중지됩니다.

데이터 전송 속도는 테이프 또는 광학 매체를 사용할 때 가장 중요합니다. 일부 테이프 디바이스는 *크기*와 상관 없이 파일을 복사하는 데 동일한 시간이 필요합니다. 아카이브 디바이스의 성능을 판별해야 합니다.

테이프 디바이스는 다른 고려사항을 갖습니다. 아카이브 요청의 빈도가 중요합니다. 예를 들어 복사 조작을 완료하는 데 걸린 시간이 5분인 경우 로그는 최대 작업 로드 중에 5분의 로그 데이터를 보유하기 충분히 커야 합니다. 테이프 디바이스는 하루당 조작 수를 제한하는 설계 한계를 가질 수 있습니다. 로그 크기를 판별할 때 이러한 인수를 고려해야 합니다.

주:

1. 무한 활성 로그 스페이스를 사용하려면 이 값을 ON으로 설정해야 합니다.

2. *userexit* 데이터베이스 구성 매개변수의 디폴트값은 롤 포워드 복구를 지원하지 않으며, 이를 사용하려는 경우 변경되어야 합니다.

NOT LOGGED INITIALLY 매개변수로 로깅 줄이기

응용프로그램이 마스터 테이블에서 작업 테이블을 작성하여 채우는 경우 마스터 테이블에서 쉽게 재작성할 수 있으므로 이러한 작업 테이블의 복구 가능성이 문제되지 않으면 CREATE TABLE문에서 NOT LOGGED INITIALLY 매개변수를 지정하여 작업 테이블을 작성할 수 있습니다. 그러면 로깅이 줄어들고 성능이 향상됩니다.

NOT LOGGED INITIALLY 매개변수를 사용하면 테이블을 작성하는 동일한 작업 단위(UOW)에서는 테이블의 변경 사항(삽입, 삭제, 갱신 또는 인덱스 작성 조작 포함)을 로그하지 않는다는 장점이 있습니다. 수행되는 로깅을 줄일 뿐만 아니라 응용프로그램 성능을 향상시킬 수도 있습니다. NOT LOGGED INITIALLY 매개변수와 함께 ALTER TABLE문을 사용하여 기존 테이블에서도 동일한 결과를 얻을 수 있습니다.

주:

1. 동일한 작업 단위(UOW)로 NOT LOGGED INITIALLY 매개변수를 사용하여 둘 이상의 테이블을 작성할 수 있습니다.
2. 카탈로그 테이블 및 기타 사용자 테이블에 대한 변경 사항이 계속 로그됩니다.

테이블에 대한 변경 사항은 로그되지 않으므로 NOT LOGGED INITIALLY 테이블 속성 사용을 결정할 때 먼저 다음을 고려합니다.

- 테이블에 대한 모든 변경 사항은 커미트할 때 디스크로 플러시됩니다. 즉, 커미트하는 시간이 더 오래 걸립니다.
- NOT LOGGED INITIALLY 속성이 활성화되고 로그되지 않는 활동이 나타나는 경우 명령문에 실패하거나 ROLLBACK TO SAVEPOINT가 실행되면(SQL1476N) 전체 작업 단위(UOW)가 롤백됩니다.
- 고가용성 재해 복구(HADR)를 사용하는 경우 NOT LOGGED INITIALLY 테이블 속성을 사용할 수 없습니다. NOT LOGGED INITIALLY 옵션이 지정된 기본 데이터베이스에서 작성된 테이블은 대기 데이터베이스로 복제되지 않습니다. HADR 대기 데이터베이스가 기본 데이터베이스로 설정된 후 이러한 테이블에 액세스하려고 하면 오류가 발생합니다.
- 롤 포워드하는 경우 이 테이블은 복구할 수 없습니다. 롤 포워드 조작 중 NOT LOGGED INITIALLY 옵션으로 변경되거나 작성된 테이블에 도달하면 테이블은 사용 불가능한 항목으로 표시됩니다. 데이터베이스를 복구한 후 테이블에 액세스하려고 하면 SQL1477N이 리턴됩니다.

  주: 테이블을 작성하면 COMMIT가 완료될 때까지 카탈로그 테이블에 행 잠금이 보유됩니다. 로그하지 않는 동작을 활용하려면 작성한 동일한 작업 단위(UOW)에서 테이블을 채워야 합니다. 이는 동시성을 함축합니다.

선언된 임시 테이블에서 로깅 줄이기

선언된 임시 테이블을 작업 테이블로 사용하려는 경우 다음을 참고하십시오.

- 선언된 임시 테이블은 카탈로그에서 작성되지 않으므로 잠금은 보유되지 않습니다.
- 첫 번째 COMMIT 이후에도 선언된 임시 테이블에서 로깅은 수행되지 않습니다.
- ON COMMIT PRESERVE 옵션을 사용하여 COMMIT 이후 테이블에서 행을 보존합니다. 그렇지 않으면 모든 행이 삭제됩니다.
- 선언된 임시 테이블을 작성한 응용프로그램만 테이블의 해당 인스턴스에 액세스할 수 있습니다.
- 데이터베이스에 대한 응용프로그램 연결이 삭제되면 테이블은 내재적으로 삭제됩니다.
- 선언된 임시 테이블을 사용하는 작업 단위(UOW)에서 작업 중 오류가 발생한 경우 작업 단위(UOW)가 완전히 롤백되지 않습니다. 그러나 선언된 임시 테이블의 컨텐츠를 변경하는 명령문에서 작업 중 오류가 발생하면 해당 테이블의 모든 행이 삭제됩니다. 작업 단위(UOW)를 롤백하면 해당 작업 단위(UOW) 또는 세이브포인트에서 수정된 선언된 임시 테이블의 모든 행이 삭제됩니다.

로그 디렉토리가 가득 차면 트랜잭션 차단

DB2 데이터베이스 관리 프로그램은 사용 중인 로그 경로에서 새 로그 파일을 작성할 수 없습니다. 새 파일을 작성할 공간이 부족하기 때문입니다. 따라서 디스크가 가득 찼음을 표시하는 오류가 발생합니다. blk_log_dsk_ful 데이터베이스 구성 매개변수를 설정하면『디스크 가득참』오류를 리턴하는 대신 파일을 성공적으로 작성할 때까지 반복해서 DB2 데이터베이스 관리 프로그램에서 새 로그 파일을 작성합니다.

blk_log_dsk_ful 데이터베이스 구성 매개변수를 설정하면 성공할 때까지 5분 간격으로 DB2 데이터베이스 관리 프로그램에서 로그 파일을 작성합니다. 로그 아카이브 방법이 지정되면 DB2 데이터베이스 관리 프로그램에서는 로그 파일 아카이브 완료도 검사합니다. 아카이브된 로그 파일이 성공적으로 아카이브되면 DB2 데이터베이스 관리 프로그램은 사용하지 않는 로그 파일 이름을 새 로그 파일 이름으로 지정하고 계속 진행할 수 있습니다. 각 시도 후 DB2 데이터베이스 관리 프로그램은 관리 통지 로그에 메시지를 기록합니다. 로그 디스크 가득참 조건 때문에 응용프로그램이 정지되었는지 확인하는 유일한 방법은 관리 통지 로그를 모니터하는 것입니다.

테이블 데이터를 갱신하려고 시도하는 사용자 응용프로그램은 로그 파일이 작성될 때까지 트랜잭션을 커미트할 수 없습니다. 읽기 전용 쿼리는 직접 영향을 받지 않을 수 있습니다. 그러나 쿼리가 갱신 요청이 잠근 데이터 또는 갱신 응용프로그램이 버퍼 풀에서 고정한 데이터 페이지에 액세스해야 하는 경우, 읽기 전용 쿼리도 정지한 것처럼 보일 수 있습니다.

로그 아카이브를 통한 로그 파일 관리

DB2 데이터 서버 로그 파일 아카이브는 다양한 운영 체제 파일 처리 및 스케줄링 문제점으로 복잡해질 수 있습니다. 예를 들어 특정 조건에서 DB2 데이터베이스 관리 프로그램은 파일을 아카이브하는 동안 로그 파일을 검색하려고 합니다. 또한 DB2 데이터베이스 관리 프로그램이 로그 파일 큐를 아카이브할 때 디스크에 실패하면 해당 로그 파일 및 포함된 트랜잭션 데이터가 유실될 수 있습니다. 데이터베이스 로깅을 올바르게 구성하면 사용 가능성 및 복구 전략을 위협하는 이러한 잠재적인 문제를 예방할 수 있습니다.

다음은 모든 로그 아카이브 방법에 적용되는 일반 고려사항입니다.

- 데이터베이스 구성 매개변수 *logarchmeth1*에 대한 값을 지정하면 지정된 방법을 사용하여 데이터베이스의 롤 포워드 복구를 수행하는 중 데이터베이스 관리 프로그램에서 파일을 아카이브하거나 로그 파일을 검색하려고 함을 나타냅니다. 로그 파일을 검색하는 요청은 롤 포워드 유틸리티가 로그 경로 디렉토리에 없는 로그 파일이 필요한 경우 요구됩니다.
- 다음 중 하나를 사용하는 경우 로컬로 연결된 테이프 드라이브를 사용하여 로그 파일을 저장해서는 안됩니다.
  - 무한 로깅
  - 온라인 테이블 스페이스 레벨 복구
  - 복제
  - 비동기 읽기 로그 API(db2ReadLog)
  - 고가용성 재해 복구(HADR)

  이러한 이벤트로 로그 파일을 검색하여 로그 아카이브 조작과 충돌할 수 있습니다.
- 로그 아카이브를 사용하는 경우 로그 관리 프로그램은 로그가 채워지면 사용 중인 로그를 아카이브하려고 합니다. 일부 경우 로그 관리 프로그램이 아카이브를 성공한 항목으로 기록하기 전에 데이터베이스가 비활성화되면 로그 관리 프로그램은 데이터베이스가 활성화될 때 로그를 다시 아카이브하려고 합니다. 따라서 로그 파일은 두 번 이상 아카이브될 수 있습니다.
- 아카이브하면 로그 파일이 여전히 사용 중이고 정상 처리를 위해 필요해도 가득 차면 로그 관리 프로그램으로 해당 로그 파일이 전달됩니다. 이를 통해 가능한 한 신속하게 휘발성 미디어로부터 데이터의 사본을 이동할 수 있습니다. 로그 관리 프로그램에 전달된 로그 파일은 더 이상 일반 처리에 필요하지 않을 때까지 로그 경로 디렉토리에 보유됩니다. 이때 디스크 스페이스가 재사용됩니다.
- 로그 파일이 아카이브되고 미해결 트랜잭션을 포함하지 않는 경우 DB2 데이터베이스 관리 프로그램은 파일을 삭제하지 않지만 이 파일이 필요할 때 다음 로그 파일로 이름을 바꿉니다. 파일 이름을 바꾸는 대신 새 로그 파일을 작성하면 디스크 스페이

스를 보장하기 위해 모든 페이지가 작성되므로 성능이 향상됩니다. 여유 공간을 확보하고 디스크에서 필요한 페이지를 다시 확보하는 것보다 재사용하는 것이 보다 효율적입니다.

- DB2 데이터베이스 관리 프로그램은 *logsecond* 데이터베이스 구성 매개변수가 -1로 설정되지 않는 한, 응급 복구 또는 롤백 중 로그 파일을 검색하지 *않습니다*.
- 로그 아카이브 구성은 실패 시점으로의 롤 포워드 복구를 보장하지 않지만 장애가 지속되는 기간을 더 짧게 설정하는 시도만 수행합니다. 로그 파일이 가득 차면 로그 관리 프로그램은 비동기적으로 로그를 아카이브합니다. 로그 파일이 가득 차기 전에 로그를 포함하는 디스크에서 장애가 발생하면 해당 로그 파일의 데이터가 유실됩니다. 또한 아카이브를 위해 파일은 큐에 대기하므로 모든 파일을 복사하기 전에 디스크에서 장애가 발생하면 큐에 있는 로그 파일이 유실될 수 있습니다.

  로그 경로가 있는 디스크 또는 디바이스에서 장애가 발생한 경우 미러 로그 경로가 있는 디스크 또는 디바이스에서도 장애가 발생하지 않는 한, MIRRORLOGPATH 데이터베이스 구성 매개변수를 사용하여 로그를 2차 경로에 작성할 수 있어야 합니다.
- 각 개별 로그 파일의 구성된 *크기*는 로그 아카이브와 직접 관련이 있습니다. 각 로그 파일이 매우 크면 디스크 장애가 발생한 경우 많은 데이터가 유실될 수 있습니다. 작은 로그 파일을 사용하도록 데이터베이스를 구성하면 로그 관리 프로그램이 로그를 더 자주 아카이브합니다.

  그러나 테이프와 같이 속도가 느린 디바이스로 데이터를 이동하는 경우 큐에서 항목이 쌓이지 않도록 더 큰 로그 파일을 사용하려는 경우가 있습니다. 큰 로그 파일은 각 파일 아카이브에서 상당한 오버헤드가 요구되는 경우(예: 테이프 디바이스 되감기 또는 아카이브 미디어에 대한 연결 설정)에도 권장됩니다.
- 로그 아카이브를 사용하는 경우 로그 관리 프로그램은 로그가 채워지면 사용 중인 로그를 아카이브하려고 합니다. 일부 경우 가득 차기 전에 로그 관리 프로그램이 로그를 아카이브하기도 합니다. SET WRITE SUSPEND 명령을 실행하거나 온라인 백업 끝에서 ARCHIVE LOG 명령을 실행하여 로그 파일이 절단되면 데이터베이스 비활성화로 인해 이 상황이 나타납니다.

  주: 사용하지 않는 로그 스페이스를 해제하기 위해 아카이브하기 전에 로그 파일이 절단됩니다.
- 로그 및 백업 이미지를 위해 스토리지 디바이스로 테이프 드라이브에 로그 및 백업 이미지를 아카이브하는 경우 아카이브된 로그 및 백업 이미지의 대상은 동일한 테이프 드라이브가 아니어야 합니다. 일부 로그 아카이브는 백업 작업을 진행 중인 동안에도 수행할 수 있으므로 두 프로세스가 동시에 동일한 테이프 드라이브에 작성하려고 하면 오류가 발생할 수 있습니다.

다음 고려사항은 로그 파일을 아카이브 및 검색하는 경우 User Exit 프로그램 또는 벤더 프로그램을 호출할 때 적용됩니다.

- DB2 데이터베이스 관리 프로그램은 User Exit 프로그램을 시작하여 파일을 아카이브할 때 읽기 모드로 로그 파일을 엽니다. 일부 플랫폼에서 이러한 방법으로 User Exit 프로그램이 로그 파일을 삭제할 수 없도록 합니다. AIX와 같은 기타 플랫폼에서는 User Exit 프로그램을 포함하는 프로세스에서 로그 파일을 삭제할 수 있습니다. 아카이브한 후에는 파일이 여전히 사용 중이고 응급 복구에 필요하므로 User Exit 프로그램은 로그 파일을 삭제할 수 없습니다. DB2 데이터베이스 관리 프로그램은 로그 파일을 아카이브할 때 디스크 스페이스 재사용을 관리합니다.
- User Exit 또는 벤더 프로그램이 현재 없는 파일을 아카이브하라는 요청을 수신하거나(아카이브 요청이 여러 개 있고 첫 번째 아카이브 조작 후 파일이 삭제된 경우) 없는 파일을 검색하라는 요청을 수신한 경우(다른 디렉토리에 있거나 로그 끝에 도달한 경우) 이 요청을 무시하고 성공적인 리턴 코드를 전달해야 합니다.
- Windows 운영 체제의 경우 REXX™ User Exit를 사용하여 로그를 아카이브할 수 없습니다.
- User Exit 또는 벤더 프로그램은 특정 시점 복구 이후 이름이 동일한 다른 로그 파일 존재를 허용해야 합니다. 두 로그 파일 모두를 보존하고 해당 로그 파일을 올바른 복구 경로와 연관하도록 작성되어야 합니다.
- 동일한 테이프 디바이스를 사용하여 로그 파일을 아카이브하는 둘 이상의 데이터베이스에 대해 User Exit 또는 벤더 프로그램이 사용 가능하고 이 중 한 데이터베이스에서 롤 포워드 조작을 수행하는 경우 다른 데이터베이스는 사용 중이 아니어야 합니다. 롤 포워드 조작을 진행하는 중에 다른 데이터베이스가 로그 파일을 아카이브하려고 하면 롤 포워드 조작에 필요한 로그를 찾을 수 없거나 테이프 디바이스에 아카이브된 새 로그 파일이 이전에 해당 테이프 디바이스에 저장된 로그 파일을 겹쳐쓸 수 있습니다.

  이를 방지하기 위해 롤 포워드 조작 중 User Exit 프로그램을 호출하는 데이터베이스 파티션에서 다른 데이터베이스를 열지 않도록 합니다. 또는 User Exit 프로그램을 작성하여 이 상황을 처리할 수 있습니다.

고가용성을 위한 클러스터링 환경 구성

머신의 클러스터를 작성하고 클러스터 관리 소프트웨어를 사용하여 머신에서 작업 로드를 분산하는 것이 고가용성 솔루션을 설계하는 한 가지 전략입니다. 클러스터에 있는 머신 중 하나 또는 여러 머신에 IBM Data Server를 설치하는 경우 데이터베이스에 영향을 주는 실패에 적절하게 반응하도록 클러스터 관리 프로그램을 구성해야 합니다. 또한 클러스터 환경에서 적절하게 작업하도록 데이터베이스 관리 프로그램 인스턴스를 구성해야 합니다.

이 태스크에 대한 정보

데이터베이스 인스턴스 및 클러스터 관리 프로그램을 수동으로 구성 및 관리하는 것은 복잡하고 시간이 걸리며 오류가 발생하기 쉽습니다. DB2 고가용성(HA) 기능은 인스턴스 구성 변경사항(예: 데이터베이스 관리 프로그램 인스턴스 중지)으로 클러스터가 변경되어야 할 때 데이터베이스 관리 프로그램이 사용자의 클러스터 관리 프로그램과 통신할 수 있도록 하는 인프라스트럭처(infrastructure)를 제공합니다.

프로시저

1. 클러스터 관리 소프트웨어를 설치하십시오.

   SA MP는 AIX, Linux 및 Solaris SPARC 운영 체제에서 DB2 Enterprise Server Edition, DB2 Workgroup Server Edition, DB2 Connect Enterprise Server Edition 및 DB2 Connect Application Server Edition과 통합됩니다. 또한 Linux 운영 체제에서는 DB2 Express-C FTL(Fixed Term License) 및 DB2 High Availability Feature for Express™ Edition과 통합합니다. Windows 운영 체제에서, SA MP는 모든 DB2 데이터베이스 제품 및 기능과 함께 번들로 묶여 있지만, DB2 설치 프로그램과 통합되지는 않습니다.
2. 클러스터 관리 프로그램에 대해 IBM Data Server 데이터베이스 관리 프로그램 인스턴스를 구성하고 IBM Data Server에 대해 클러스터 관리 프로그램을 구성하십시오.

   DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)는 클러스터된 환경에서 고가용성 데이터베이스를 구성 및 관리하기 위해 사용할 수 있는 텍스트 기반 유틸리티입니다. db2haicu는 사용자 시스템을 쿼리하여 데이터베이스 인스턴스, 사용자의 클러스터 환경 및 사용자의 클러스터 관리 프로그램에 대한 정보를 수집합니다. db2haicu 프롬프트에 정보를 제공하여 db2haicu 호출, 입력 파일에, 또는 런타임에서 매개변수를 통해 자세한 정보를 제공합니다.
3. 시간이 지남에 따라서 데이터베이스에 변경이 필요하고 클러스터 환경 안에서 데이터베이스 구성을 수정해야 하므로, 데이터베이스 관리 프로그램 인스턴스 구성 및 클러스터 관리 프로그램 구성이 동기화되도록 계속 유지하십시오.

   DB2 고가용성(HA) 기능은 인스턴스 구성 변경사항(예: 데이터베이스 관리 프로그램 인스턴스 중지)으로 클러스터가 변경되어야 할 때 데이터베이스 관리 프로그램이 사용자의 클러스터 관리 프로그램과 통신할 수 있도록 하는 인프라스트럭처(infrastructure)를 제공합니다.

   SA MP에서 db2haicu를 사용하거나 DB2 클러스터 관리 프로그램 API를 지원하는 다른 클러스터 관리 프로그램을 사용하는지 여부와 상관 없이, DB2 HA Feature를 사용하여 클러스터 환경을 관리하는 것이 데이터베이스 관리 프로그램 구성 및 클러스터 구성을 개별적으로 유지보수하는 것보다 더 쉽습니다.

DB2 고가용성(HA) 기능과의 클러스터 관리 프로그램 통합

DB2 고가용성(HA) 기능을 사용하여 IBM Data Server와 클러스터 관리 소프트웨어 사이에 통합할 수 있습니다.

클러스터된 환경에서 데이터베이스 관리 프로그램 인스턴스를 중지할 때 인스턴스가 중지된 것을 클러스터 관리 프로그램이 인식하도록 해야 합니다. 클러스터 관리 프로그램이 인스턴스가 중지된 것을 인식하지 못하면, 클러스터 관리 프로그램은 중지된 인스턴스에 장애 복구와 같은 작업을 시도할 수 있습니다. DB2 고가용성(HA) 기능은 인스턴스 구성 변경사항(예: 데이터베이스 관리 프로그램 인스턴스 중지)으로 클러스터가 변경되어야 할 때 데이터베이스 관리 프로그램이 사용자의 클러스터 관리 프로그램과 통신할 수 있도록 하는 인프라스트럭처(infrastructure)를 제공합니다.

DB2 HA 기능은 다음 요소로 구성됩니다.

- IBM Tivoli SA MP(System Automation for Multiplatforms)는 DB2 고가용성(HA) 기능의 일부로 AIX 및 Linux의 IBM Data Server와 함께 번들로 제공되며 DB2 설치 프로그램으로 통합됩니다. DB2 설치 프로그램이나, IBM Data Server 설치 미디어에 포함된 installSAM 및 uninstallSAM 스크립트를 사용하여 SA MP를 설치, 업그레이드 또는 설치 제거할 수 있습니다.
- 클러스터 환경에서 일부 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작은 관련 클러스터 구성 변경이 필요합니다. DB2 고가용성(HA) 기능은 사용자가 특정 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작을 수행할 때마다 데이터베이스 관리 프로그램이 자동으로 클러스터 관리 프로그램 구성 변경을 요청할 수 있도록 합니다. 참조: 83 페이지의 『DB2 고가용성(HA) 기능을 사용하도록 자동으로 클러스터 구성』
- DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)는 클러스터된 환경에서 고가용성 데이터베이스를 구성 및 관리하기 위해 사용할 수 있는 텍스트 기반 유틸리티입니다. db2haicu는 사용자 시스템을 쿼리하여 데이터베이스 인스턴스, 사용자의 클러스터 환경 및 사용자의 클러스터 관리 프로그램에 대한 정보를 수집합니다. db2haicu 프롬프트에 정보를 제공하여 db2haicu 호출, 입력 파일에, 또는 런타임에서 매개변수를 통해 자세한 정보를 제공합니다. 참조: 92 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)』
- The DB2 클러스터 관리 프로그램 API는 데이터베이스 관리 프로그램이 클러스터 관리 프로그램의 구성 변경사항을 통신할 수 있도록 하는 기능 세트를 정의합니다. 참조: 131 페이지의 『DB2 클러스터 관리 프로그램 API』

IBM Tivoli SA MP(System Automation for Multiplatforms)

IBM Tivoli SA MP(System Automation for Multiplatforms)는 AIX, Linux, Solaris SPARC 및 Windows용 고가용성 및 재해 복구 기능을 제공합니다.

SA MP는 AIX, Linux 및 Solaris SPARC 운영 체제의 DB2 Enterprise Server Edition, DB2 Workgroup Server Edition, DB2 Connect Enterprise Server Edition 및 DB2 Connect Application Server Edition에 통합됩니다. 또한, Linux 운영 체제의 DB2 Express-C FTL(Fixed Term License) 및 Express Edition용 DB2 High Availability 기능과도 통합됩니다. Windows 운영 체제에서, SA MP는 모든 DB2 데이터베이스 제품 및 기능과 함께 번들로 묶여 있지만, DB2 설치 프로그램과 통합되지는 않습니다.

이 SA MP 사본을 사용하여 DB2 데이터베이스 시스템의 고가용성을 관리할 수 있습니다. SA MP 라이센스를 위한 업그레이드를 구매하지 않은 경우, 이를 사용하여 클러스터를 관리할 수 없습니다.

SA MP는 AIX 및 Linux에서 IBM Data Server 클러스터 환경의 디폴트 클러스터 관리 프로그램입니다.

SA MP에 관한 자세한 정보는 http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html을 참조하십시오. 지원되는 운영 체제 목록은 http://www.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html 웹 사이트를 참조하십시오.

DB2 고가용성(HA) 기능을 사용하도록 자동으로 클러스터 구성

클러스터 환경에서 일부 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작은 관련 클러스터 구성 변경이 필요합니다. DB2 고가용성(HA) 기능은 사용자가 특정 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작을 수행할 때마다 데이터베이스 관리 프로그램이 자동으로 클러스터 관리 프로그램 구성 변경을 요청할 수 있도록 합니다.

시작하기 전에

데이터베이스 관리 프로그램이 데이터베이스 관리 태스크에 대해 필요한 클러스터 구성을 수행할 수 있도록 하려면 db2haicu를 사용하여 인스턴스에 대한 *클러스터 도메인*을 작성함으로써 *고가용성을 위한 인스턴스를 구성*해야 합니다. 자세한 정보는 85 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 환경 구성』을 참조하십시오.

프로시저

다음 데이터베이스 관리 프로그램 인스턴스 구성 및 관리 조작을 수행할 때, 데이터베이스 관리 프로그램이 자동으로 관련 클러스터 관리 프로그램 구성을 수행합니다.

- START DATABASE 또는 db2start를 사용하여 데이터베이스 시작
- STOP DATABASE 또는 db2stop을 사용하여 데이터베이스 중지
- CREATE DATABASE를 사용한 데이터베이스 작성

- CREATE TABLESPACE를 사용한 스토리지 추가
- ALTER TABLESPACE DROP 또는 DROP TABLESPACE를 사용한 스토리지 제거
- ALTER DATABASE를 사용하여 스토리지 경로 추가 또는 제거
- DROP TABLESPACE를 사용하여 데이터베이스 삭제
- RESTORE DATABASE 또는 db2Restore를 사용하여 데이터베이스 리스토어
- SET TABLESPACE CONTAINERS를 사용하여 경로 재지정된 리스토어에 대한 테이블 스페이스 컨테이너 지정
- ROLLFORWARD DATABASE 또는 db2Rollforward를 사용하여 데이터베이스 롤포워드
- RECOVER DATABASE 또는 db2Recover를 사용하여 데이터베이스 복구
- CREATE EVENT MONITOR를 사용한 이벤트 모니터 작성
- DROP EVENT MONITOR를 사용한 이벤트 모니터 삭제
- 다음을 사용하여 외부 루틴 작성 및 변경:
  - CREATE PROCEDURE
  - CREATE FUNCTION
  - CREATE FUNCTION
  - CREATE METHOD
  - ALTER PROCEDURE
  - ALTER FUNCTION
  - ALTER METHOD
- 다음을 사용하여 외부 루틴 삭제:
  - DROP PROCEDURE
  - DROP FUNCTION
  - DROP METHOD
- START HADR을 사용하여 데이터베이스에 대한 DB2 고가용성 재해 복구(HADR) 조작 시작
- STOP HADR을 사용하여 데이터베이스에 대한 HADR 조작 중지
- TAKEOVER HADR을 사용하여 HADR 대기 데이터베이스가 HADR 기본 데이터베이스로서 인계하도록 함
- 데이터베이스 관리 프로그램 구성 매개변수 **DIAGPATH** 또는 **SPM_LOG_PATH** 설정
- 데이터베이스 구성 매개변수 **NEWLOGPATH**, **OVERFLOWLOGPATH**, **MIRRORLOGPATH** 또는 **FAILARCHPATH** 설정
- db2idrop을 사용하여 데이터베이스 관리 프로그램 인스턴스 삭제

결과

데이터베이스 관리 프로그램이 나열된 데이터베이스 관리 태스크에 대한 클러스터 구성 변경을 조정할 때, 사용자가 별도의 클러스터 관리 프로그램 조작을 수행할 필요가 없습니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 환경 구성

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 환경에서 데이터베이스를 구성 및 관리할 수 있습니다. 데이터베이스 관리 프로그램 인스턴스 구성 세부사항을 db2haicu에 지정할 때 db2haicu는 필요한 클러스터 구성 세부사항을 클러스터 관리 소프트웨어에 통신합니다.

시작하기 전에

- SLES(SUSE Linux Enterprise Server) 11에서 IBM Tivoli SA MP(System Automation for Multiplatforms) 버전 3.1에 DB2 고가용성을 사용하려면, db2haicu 도구를 실행하여 HA 환경을 작성하기 전에 먼저 SA MP 버전 3.1 Fixpack 4를 다운로드하여 설치해야 합니다. 필수 Fixpack을 다운로드하려면 http://www.ibm.com/software/tivoli/support/sys-auto-multi을 참조하십시오.
- DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하기 전에 수행해야 하는 태스크 세트가 있습니다. 자세한 정보는 126 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 전제조건』을 참조하십시오.

제한사항

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 사용에 대한 몇 가지 제한사항이 있습니다. 자세한 정보는 129 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 전제조건』을 참조하십시오.

이 태스크에 대한 정보

db2haicu를 대화식으로 또는 XML 입력 파일을 사용하여 실행할 수 있습니다.

대화식 모드

-f 매개변수를 사용하여 XML 입력 파일을 지정하지 않고 db2haicu 명령을 실행하여 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 호출할 때, 유틸리티는 대화식 모드에서 실행됩니다. 대화식 모드에서 db2haicu는 정보를 표시하고 텍스트 기반 형식으로 사용자에게 정보를 쿼리합니다. 자세한 정보는 95 페이지의 『대화식 모드에서 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 실행』을 참조하십시오.

XML 입력 파일을 사용하는 일괄처리 모드

`-f <input-file-name>` 매개변수를 db2haicu 명령과 함께 사용하여 구성 세

부사항을 지정한 XML 입력 파일과 함께 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행할 수 있습니다. XML 입력 파일과 함께 db2haicu를 실행하는 것은 고가용성을 위해 구성되어야 하는 데이터베이스 파티션이 여러 개 있을 때처럼 구성 태스크를 여러 번 수행해야 할 때 유용합니다. 자세한 정보는 96 페이지의 『XML 입력 파일과 함께 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 실행』을 참조하십시오.

프로시저

각 데이터베이스 관리 프로그램 인스턴스에 대해 다음 단계를 수행하십시오.

1. 새 클러스터 도메인 작성

   처음으로 데이터베이스 관리 프로그램 인스턴스에 대해 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행할 때 db2haicu가 *클러스터 도메인*이라는 사용자 클러스터의 모델을 작성합니다. 자세한 정보는 127 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인 작성』을 참조하십시오.
2. 계속해서 클러스터 도메인 구성 세부 정의 및 클러스터 도메인 관리 및 유지보수

   db2haicu를 사용하여 클러스터 환경의 클러스터 도메인 모델을 수정할 때, 데이터베이스 관리 프로그램은 관련 변경을 데이터베이스 관리 프로그램 인스턴스 및 클러스터 구성으로 전파시킵니다. 자세한 정보는 128 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인 유지보수』를 참조하십시오.

다음 단계

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)에는 별도의 진단 로그가 없습니다. 데이터베이스 관리 프로그램 진단 로그, db2diag 로그 파일 및 db2pd 도구를 사용하여 db2haicu 오류를 조사하고 진단할 수 있습니다. 자세한 정보는 129 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 문제점 해결』을 참조하십시오.

클러스터 도메인

클러스터 도메인은 클러스터 요소(예: 데이터베이스, 마운트 지점 및 장애 복구 규정)에 대한 정보를 포함하는 모델입니다. DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인을 작성합니다. db2haicu는 클러스터 도메인에 있는 정보를 사용하여 구성 및 유지보수 클러스터 관리 태스크가 가능하도록 합니다. 또한 DB2 고가용성(HA) 기능의 일부로, 데이터베이스 관리 프로그램이 클러스터 도메인의 정보를 사용하여 자동화된 클러스터 관리 태스크를 수행합니다.

클러스터 도메인에 클러스터 요소를 추가하면, 그 요소는 모든 연속 db2haicu 구성 작업이나, DB2 HA 기능의 일부로 데이터베이스 관리 프로그램이 수행하는 클러스터 관리 작업에 포함됩니다. 클러스터 도메인에서 클러스터 요소를 제거하면, 그 요소는 더

이상 db2haicu 구성 작업이나 데이터베이스 관리 프로그램 자동화 클러스터 관리 작업에 포함되지 않습니다. db2haicu 및 데이터베이스 관리 프로그램은 db2haicu를 사용하여 작성하는 클러스터 도메인에 있는 클러스터 요소에 대해서만 클러스터 관리 프로그램과 함께 조정할 수 있습니다.

db2haicu를 사용하여 다음 클러스터 도메인 요소를 작성하고 구성할 수 있습니다.

- 컴퓨터 또는 머신(클러스터 도메인 컨텍스트에서는 이를 *클러스터 도메인* 노드라고 함)
- 네트워크 인터페이스 카드 또는 NIC(db2haicu에서는 *네트워크 인터페이스*, *인터페이스*, *네트워크 어댑터* 또는 *어댑터*로 언급됨)
- IP 주소
- 고가용성 재해 복구(HADR) 기본 및 대기 데이터베이스 쌍을 포함한, 데이터베이스
- 데이터베이스 파티션
- 실패 발생 시 장애 복구에 중요하지 않은 경로를 포함한, 마운트 지점 및 경로
- 장애 복구 규정
- Quorum 디바이스

클러스터 관리 소프트웨어:

클러스터 관리 소프트웨어는 컴퓨터 클러스터가 수행할 수 있는 작업을 최적화합니다. 클러스터 관리 프로그램은 워크로드 밸런스를 유지하고, 클러스터 요소의 상태(health)를 모니터하며, 요소 실패 시 장애 복구를 관리합니다. 클러스터 관리 프로그램은 또한 시스템 관리자가 클러스터에 있는 요소에 대해 관리 태스크를 수행할 때 도움을 제공할 수 있습니다(예를 들어, 서비스가 필요한 컴퓨터로부터 워크로드를 리라우트하여).

클러스터의 요소

적절하게 작동하도록 하려면, 클러스터 관리 프로그램이 클러스터 요소에 관련된 많은 세부사항을 인식하고, 클러스터 요소 사이의 관계를 인식해야 합니다.

다음은 클러스터 관리 프로그램이 인식해야 하는 클러스터 요소의 예입니다.

- 클러스터에 있는 실제 또는 가상 컴퓨터, 머신 또는 디바이스(클러스터 컨텍스트에서는 이를 *클러스터* 노드라고 함)
- 클러스터 노드를 연결하는 네트워크
- 클러스터 노드를 네트워크에 연결하는 네트워크 인터페이스
- 클러스터 노드의 IP 주소
- 가상 또는 서비스 IP 주소

다음은 클러스터 관리 프로그램이 인식해야 하는 관계의 예입니다.

- 동일한 소프트웨어가 설치되고 서로 장애 복구할 수 있는 클러스터 노드 쌍
- 등록 정보가 동일하고 서로 장애 복구에 사용될 수 있는 네트워크

- 가상 IP 주소가 현재 연관되어 있는 클러스터 노드

클러스터 요소 추가 또는 수정

클러스터 관리 프로그램이 클러스터의 요소와 요소 사이의 관계를 인식하도록 하려면 시스템 관리자가 클러스터 관리 프로그램에 대해 요소를 등록해야 합니다. 시스템 관리자가 클러스터 요소를 변경하는 경우 관리자는 변경사항을 클러스터 관리 프로그램과 통신해야 합니다. 클러스터 관리 프로그램에는 이러한 태스크에 도움이 될 수 있는 인터페이스가 있습니다.

클러스터 관리는 가능한 클러스터 요소가 다양하므로 어렵습니다. 관리자는 클러스터 노드의 하드웨어 및 운영 체제, 네트워킹 프로토콜 및 구성과, 클러스터 노드에 설치된 소프트웨어(예: 데이터베이스 소프트웨어)에 대해 전문가여야 합니다. 클러스터 관리 소프트웨어에 대해 클러스터 요소를 등록하거나 시스템 변경 후 클러스터 관리 프로그램을 갱신하는 것은 복잡하고 많은 시간이 소비될 수 있습니다.

db2haicu를 사용하여 클러스터의 요소 추가 또는 수정

DB2 데이터베이스 솔루션에서, DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 관리 프로그램에 대해 클러스터 요소를 등록하고, 클러스터에 대한 관리 변경사항 작성 후 클러스터 관리 프로그램을 갱신할 수 있습니다. db2haicu는 사용하면 이러한 태스크가 단순하게 됩니다. db2haicu가 클러스터의 요소를 캡슐화하기 위해 사용하는 모델과 요소 사이의 관계를 알면 태스크를 수행하기 위해 하드웨어, 운영 체제 및 클러스터 관리 프로그램 인터페이스의 특징에 대해 전문가가 아니어도 됩니다.

자원 및 자원 그룹:

*자원*은 클러스터 관리 프로그램에 대해 등록된 인터페이스 클러스터 노드, 데이터베이스, 마운트 지점 또는 네트워크 인터페이스 카드와 같은 클러스터 요소입니다. 요소가 클러스터 관리 프로그램에 대해 등록되어 있지 않으면 클러스터 관리 프로그램이 요소를 인식하지 못하고 클러스터 관리 작업에 해당 요소를 포함시키지 않습니다. *자원 그룹*은 논리적 자원 콜렉션입니다. 자원 그룹은 해당 그룹에 있는 자원에 대한 복잡한 관리 태스크 수행을 단순화하는 관계 및 제한조건이 자원 그룹에 정의될 수 있으므로 아주 강력한 구성체입니다.

클러스터 관리 프로그램이 자원을 그룹으로 수집하는 경우, 클러스터 관리 프로그램은 해당되는 모든 자원에 대해 총체적으로 작동할 수 있습니다. 예를 들어, 두 개의 데이터베이스 `database-1` 및 `database-2`가 `resource-group-A` 자원 그룹에 속하는 경우 클러스터 관리 프로그램이 `resource-group-A`에 대해 시작 조작을 수행하면 `database-1` 및 `database-2`는 하나의 클러스터 관리 프로그램 조작에 의해 시작됩니다.

제한사항

- 자원 그룹은 *equivalency*를 포함할 수 없고 equivalency는 자원 그룹을 포함할 수 없습니다(*Equivalency*는 서로의 동일한 기능을 제공하는 자원 세트로, 서로 장애 복구할 수 있습니다.).
- 자원은 하나의 자원 그룹에만 있을 수 있습니다.
- 자원은 자원 그룹 및 equivalency에 있을 수 없습니다.
- 자원 그룹은 다른 자원 그룹을 포함할 수 있지만 최대 중첩 레벨은 50입니다.
- 자원 그룹에서 수집할 수 있는 최대 자원 수는 100입니다.

Quorum 디바이스:

*Quorum 디바이스*는 클러스터 관리 프로그램이 정상적인 결정 프로세스에서 명확한 선택을 하지 못한 경우 클러스터 관리 결정을 하는 데 도움을 제공합니다. 클러스터 관리 프로그램이 가능한 여러 조치 사이에서 선택해야 하는 경우 클러스터 관리 프로그램은 각각의 가능한 조치를 지원하는 클러스터 도메인 노드 수를 계산한 후 대다수의 클러스터 도메인 노드에서 지원되는 조치를 선택합니다. 정확하게 동일한 클러스터 도메인 노드 수가 두 개 이상의 선택사항을 지원하는 경우, 클러스터 관리 프로그램은 quorum 디바이스를 참조하여 선택합니다.

db2haicu는 다음 표에 나열된 quorum 디바이스를 지원합니다.

표 2. db2haicu에서 지원되는 quorum 디바이스의 유형

Quorum 디바이스	설명
네트워크	네트워크 quorum 디바이스는 모든 클러스터 도메인 노드가 언제나 연결할 수 있는 IP 주소입니다.

클러스터 도메인의 네트워크:

네트워크에 관련되는 클러스터 도메인 요소를 구성하려면, DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 *실제 네트워크*를 클러스터 도메인에 추가하면 됩니다. 실제 네트워크는 네트워크 인터페이스 카드, IP 주소 및 서브네트워크 마스크로 구성됩니다.

네트워크 인터페이스 카드

네트워크 인터페이스 카드(NIC)는 컴퓨터(*클러스터 노드*라고도 함)를 네트워크에 연결하는 하드웨어입니다. NIC는 간혹 *인터페이스*, *네트워크 어댑터* 또는 *어댑터*로 언급됩니다. db2haicu를 사용하여 실제 네트워크를 클러스터 도메인에 추가할 때 NIC가 속하는 컴퓨터의 호스트 이름, 호스트 컴퓨터의 NIC 이름, NIC의 IP 주소를 포함하여 최소 하나의 NIC를 지정합니다.

IP 주소

인터넷 프로토콜 주소(IP 주소)는 네트워크에서 고유한 주소입니다. IP 버전 4에서, IP 주소는 32비트 대형이고, 보통 129.30.180.16과 같이 점 10진 표기로 표현됩니다. IP 주소는 네트워크 부분과 호스트 컴퓨터 부분으로 구성됩니다.

db2haicu는 IP 버전 6을 지원하지 않습니다.

서브네트워크 마스크

네트워크는 *서브네트워크 마스크*를 사용하여 여러 개의 논리 서브네트워크로 파티션됩니다. 서브네트워크 마스크는 IP 주소의 호스트 부분 중 일부 비트를 IP 주소의 네트워크 부분으로 이동하기 위한 메커니즘입니다. db2haicu를 사용하여 IP 주소를 클러스터 도메인에 추가할 때 간혹 IP 주소에 대한 서브네트워크 마스크를 지정해야 합니다. 예를 들어, db2haicu를 사용하여 NIC를 추가할 때 NIC의 IP 주소에 대한 서브네트워크 마스크를 지정해야 합니다.

네트워크 equivalency

*Equivalency*는 서로의 동일한 기능을 제공하는 자원 세트로, 서로 장애 복구할 수 있습니다. db2haicu를 사용하여 네트워크를 작성할 때, 해당 네트워크의 NIC는 서로 장애 복구할 수 있습니다. 이와 같은 네트워크를 *네트워크 equivalency*라고도 합니다.

네트워크 프로토콜

db2haicu를 사용하여 네트워크를 클러스터 도메인에 추가할 때 사용하는 네트워크 프로토콜의 유형을 지정해야 합니다. 현재는 TCP/IP 네트워크 프로토콜만 지원됩니다.

클러스터 도메인에서 장애 복구 규정:

*장애 복구 규정*은 네트워크 인터페이스 카드 또는 데이터베이스 서버와 같은 클러스터 요소가 실패할 때 클러스터 관리 프로그램이 응답해야 하는 방법을 지정합니다. 일반적으로, 클러스터 관리 프로그램은 이전에 클러스터 관리 프로그램에 대해 실패한 요소에 적절한 교체로 식별된 대체 요소로 실패한 요소의 워크로드를 전송합니다. 실패한 요소에서 보조 요소로의 이러한 워크로드 전송을 *장애 복구*라고 합니다.

라운드 로빈 장애 복구 규정

*라운드 로빈 장애 복구 규정*을 사용 중일 때, 클러스터 도메인에 있는 한 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함)와 연관되는 실패가 있는 경우 데이터베이스 관리 프로그램은 클러스터 도메인에 있는 다른 노드의 실패한 클러스터 도메인 노드에서 작업을 재시작합니다.

상호 장애 복구 규정

*상호 장애 복구 규정*을 구성하려면 클러스터 도메인에 있는 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함) 쌍을 시스템 쌍으로 연관시킵니다. 이 쌍에 있는 노드 중 하나에서 실패가 발생하는 경우 실패한 노드의 데이터베이스 파티션은 쌍의 다른 노드로 장애 복구합니다. 상호 장애 복구는 여러 개의 데이터베이스 파티션이 있는 경우에만 사용 가능합니다.

N Plus M 장애 복구 규정

*N Plus M 장애 복구 규정*을 사용 중일 때, 클러스터 도메인에 있는 한 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함)와 연관되는 실패가 있는 경우 실패한 노드의 데이터베이스 파티션은 클러스터 도메인에 있는 다른 노드로 장애 복구합니다. N Plus M 장애 복구는 여러 개의 데이터베이스 파티션이 있는 경우에만 사용 가능합니다.

로컬 재시작 장애 복구 규정

*로컬 재시작 장애 복구 규정*을 사용 중일 때, 클러스터 도메인에 있는 한 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함)와 연관되는 실패가 있는 경우 데이터베이스 관리 프로그램은 실패한 동일 노드의 제 위치에서(또는 로컬로) 데이터베이스를 재시작합니다.

HADR 장애 복구 규정

*HADR 장애 복구 규정*을 구성하는 경우, DB2 고가용성 재해 복구(HADR) 기능이 장애 복구를 관리하도록 할 수 있습니다. HADR 기본 데이터베이스가 실패하면, 데이터베이스 관리 프로그램은 실패한 데이터베이스에서 HADR 대기 데이터베이스로 워크로드를 이동합니다.

사용자 정의 장애 복구 규정

*사용자 정의 장애 복구 규정*을 구성한 경우 클러스터 도메인에서 데이터베이스 관리 프로그램이 장애 복구할 수 있는 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함) 목록을 작성합니다. 클러스터 도메인의 노드가 실패하면 데이터베이스 관리 프로그램은 실패한 노드에서, 사용자가 지정한 목록에 있는 노드 중 하나로 워크로드를 이동합니다.

클러스터 도메인에서의 마운트 지점:

파일 시스템을 마운트하면 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 마운트 지점을 클러스터 도메인에 추가할 수 있습니다.

마운트 지점

UNIX, Linux 및 AIX 운영 체제에서, 파일 시스템을 *마운트*한다는 것은 운영 체제가 파일 시스템을 사용 가능하도록 만든다는 의미입니다. 마운트 작업 중, 운영 체제는 인덱스 읽기 또는 데이터 구조 탐색과 같은 태스크를 수행하고 디렉토리 경로를 마운트된 파일 시스템과 연관시킵니다. 마운트된 파일 시스템에 액세스하기 위해 사용할 수 있는 이 연관된 디렉토리 경로를 *마운트 지점*이라고 합니다.

중요하지 않은 마운트 지점 또는 경로

클러스터에 실패 발생 시 장애 복구하지 않아도 되는 마운트 지점이나 경로가 있을 수 있습니다. db2haicu를 사용하여 이와 같은 중요하지 않은 마운트 지점 또는 경로 목록을 클러스터 도메인에 추가할 수 있습니다. 클러스터 관리 프로그램은 중요하지 않은 목록에 있는 마운트 지점 또는 경로를 장애 복구 작업에 포함시키지 않습니다.

예를 들어, 클러스터의 node1 컴퓨터에서 하드 드라이브를 /mnt/driveA에 마운트한 경우를 고려해 보십시오. /mnt/driveA를 사용 가능하도록 하는 것이 중요하다고 결정하면, 클러스터 관리 프로그램은 node1이 실패하는 경우 /mnt/driveA를 사용 가능하도록 유지하기 위해 장애 복구합니다. 그러나 node1이 실패하는 경우 /mnt/driveA가 사용 불가능하게 되는 것을 승인할 것을 결정하면, 중요하지 않은 경로 목록에 /mnt/driveA를 추가하여 /mnt/driveA가 장애 복구에 중요하지 않음을 클러스터 관리 프로그램에 표시할 수 있습니다. /mnt/driveA가 복구에 중요하지 않은 것으로 식별되면 node1이 실패하는 경우 해당 드라이브를 사용할 수 없습니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)는 클러스터된 환경에서 고가용성 데이터베이스를 구성 및 관리하기 위해 사용할 수 있는 텍스트 기반 유틸리티입니다. db2haicu는 사용자 시스템을 쿼리하여 데이터베이스 인스턴스, 사용자의 클러스터 환경 및 사용자의 클러스터 관리 프로그램에 대한 정보를 수집합니다. db2haicu 프롬프트에 정보를 제공하여 db2haicu 호출, 입력 파일에, 또는 런타임에서 매개변수를 통해 자세한 정보를 제공합니다.

구문

```
db2haicu [ -f <XML-input-file-name> ]
         [ -disable ]
         [ -delete [ dbpartitionnum <db-partition-list> |
                     hadrdb <database-name> ] ]
```

매개변수

db2haicu 명령으로 전달하는 매개변수는 대소문자가 구분되며 소문자여야 합니다.

-f *<XML-input-file-name>*

-f 매개변수를 사용하여 XML 입력 파일 *<XML-input-file-name>*에서 클러

스터 도메인 세부사항을 지정할 수 있습니다. 자세한 정보는 96 페이지의 『XML 입력 파일과 함께 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 실행』을 참조하십시오.

-disable

사용자가 데이터베이스 관리 프로그램 인스턴스에 대한 클러스터 도메인을 작성하기 위해 db2haicu를 사용하면 이 인스턴스는 *고가용성으로 구성된 것으로* 고려됩니다. 데이터베이스 관리 프로그램 인스턴스가 고가용성으로 구성된 경우, 관련된 클러스터 구성 변경사항이 필요한 데이터베이스 관리 프로그램 관리 조작을 수행할 때마다 데이터베이스 관리 프로그램은 클러스터 관리 프로그램과 해당되는 클러스터 구성 변경사항을 통신합니다. 데이터베이스 관리 프로그램이 사용자 대신 클러스터 관리 프로그램과 함께 이 클러스터 관리 태스크를 조정하는 경우, 사용자는 해당되는 관리 태스크에 대해 별도의 클러스터 관리 프로그램 조작을 수행하지 않아도 됩니다. 데이터베이스 관리 프로그램과 클러스터 관리 프로그램 사이의 이와 같은 통합은 DB2 고가용성(HA) 기능입니다.

-disable 매개변수를 사용하여 데이터베이스 관리 프로그램 인스턴스가 고가용성으로 구성되는 것이 중지되도록 할 수 있습니다. 데이터베이스 관리 프로그램 인스턴스가 더 이상 고가용성으로 구성되지 않은 경우, 사용자가 관련된 클러스터 구성 변경사항이 필요한 데이터베이스 관리 프로그램 관리 조작을 수행해도 데이터베이스 관리 프로그램은 클러스터 관리 프로그램과 함께 조정하지 않습니다.

데이터베이스 관리 프로그램 인스턴스를 고가용성으로 재구성하려면 다시 db2haicu을 실행하면 됩니다.

-delete

-delete 매개변수를 사용하여 현재 데이터베이스 관리 프로그램 인스턴스에 대한 자원 그룹을 삭제할 수 있습니다.

dbpartitionnum 매개변수나 **hadrdb** 매개변수를 사용하지 않는 경우 db2haicu는 현재 데이터베이스 관리 프로그램 인스턴스와 연관되는 모든 자원 그룹을 제거합니다.

dbpartitionnum **<***db-partition-list***>**

dbpartitionnum 매개변수를 사용하여 <*db-partition-list*>에 나열된 데이터베이스 파티션과 연관되는 자원 그룹을 삭제할 수 있습니다. <*db-partition-list*>는 데이터베이스 파티션을 식별하는 번호 목록으로, 쉼표로 구분됩니다.

hadrdb **<***database-name***>**

hadrdb 매개변수를 사용하여 DB2 고가용성 재해 복구(HADR) 데이터베이스 <*database-name*>과 연관되는 자원 그룹을 삭제할 수 있습니다.

db2haicu가 자원 그룹을 제거한 후 클러스터 도메인에서 자원 그룹이 남아 있지 않으면 db2haicu는 클러스터 도메인도 제거합니다.

-delete 매개변수를 사용하여 db2haicu를 실행하면 현재 데이터베이스 관리 프로그램 인스턴스가 고가용성으로 구성되는 것이 중지됩니다. 데이터베이스 관리 프로그램 인스턴스가 더 이상 고가용성으로 구성되지 않은 경우, 사용자가 관련된 클러스터 구성 변경사항이 필요한 데이터베이스 관리 프로그램 관리 조작을 수행해도 데이터베이스 관리 프로그램은 클러스터 관리 프로그램과 함께 조정하지 않습니다.

데이터베이스 관리 프로그램 인스턴스를 고가용성으로 재구성하려면 다시 db2haicu을 실행하면 됩니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 시작 모드:

제공된 데이터베이스 관리 프로그램 인스턴스에 대해 처음 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행하는 경우 db2haicu는 시작 모드에서 작동합니다.

db2haicu를 실행할 때, db2haicu는 데이터베이스 관리 프로그램 인스턴스와 사용자의 시스템 구성을 조사하고 기존 *클러스터 도메인*을 검색합니다. 클러스터 도메인은 클러스터 요소(예: 데이터베이스, 마운트 지점 및 장애 복구 규정)에 대한 정보를 포함하는 모델입니다. DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인을 작성합니다. db2haicu는 클러스터 도메인에 있는 정보를 사용하여 구성 및 유지보수 클러스터 관리 태스크가 가능하도록 합니다. 또한 DB2 고가용성(HA) 기능의 일부로, 데이터베이스 관리 프로그램이 클러스터 도메인의 정보를 사용하여 자동화된 클러스터 관리 태스크를 수행합니다.

제공된 데이터베이스 관리 프로그램 인스턴스에 대해 db2haicu를 실행하는데 해당 인스턴스에 대해 이미 작성되고 구성된 클러스터 도메인이 없는 경우 db2haicu는 즉시 새 클러스터 도메인을 작성하고 구성하는 프로세스를 시작합니다. db2haicu는 새 클러스터 도메인의 이름과 현재 머신의 호스트 이름과 같은 정보를 사용자에게 프롬프트하여 새 클러스터 도메인을 작성합니다.

클러스터 도메인을 작성하지만 클러스터 도메인 구성 태스크가 완료되지 않은 경우, 다음에 db2haicu를 실행할 때 db2haicu는 클러스터 도메인 구성 태스크를 재개합니다.

데이터베이스 관리 프로그램 인스턴스에 대한 클러스터 도메인을 작성 및 구성하고 나면, db2haicu는 유지보수 모드에서 실행됩니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 유지보수 모드:

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행하고 현재 데이터베이스 관리 프로그램 인스턴스에 대해 작성된 클러스터 도메인이 이미 있는 경우, db2haicu는 유지보수 모드에서 작동합니다.

db2haicu가 유지보수 모드에서 실행 중인 경우 db2haicu는 사용자가 수행할 수 있는 구성 및 관리 태스크 목록을 제시합니다.

db2haicu 유지보수 태스크에는 데이터베이스 또는 클러스터 노드 같은 클러스터 요소를 클러스터 도메인에 추가 또는 클러스터 도메인에서 요소 제거가 포함됩니다. db2haicu 유지보수 태스크에는 데이터베이스 관리 프로그램 인스턴스에 대한 장애 복구 규정 같은 클러스터 도메인 요소의 세부사항 수정도 포함됩니다.

db2haicu를 유지보수 모드에서 실행할 때 db2haicu는 클러스터 도메인에 대해 수행할 수 있는 조작 목록을 제공합니다.

- 클러스터 노드(호스트 이름으로 식별되는 머신) 추가 또는 제거
- 네트워크 인터페이스(네트워크 인터페이스 카드) 추가 또는 제거
- 데이터베이스 파티션 추가 또는 제거(파티션된 데이터베이스 환경만 해당)
- DB2 고가용성 재해 복구(HADR) 데이터베이스 추가 또는 제거
- 고가용성 데이터베이스 추가 또는 제거
- 마운트 지점 추가 또는 제거
- IP 주소 추가 또는 제거
- 중요하지 않은 경로 추가 또는 제거
- 스케줄된 유지보수를 위한 데이터베이스 파티션 및 HADR 데이터베이스 이동
- 현재 인스턴스에 대한 장애 복구 규정 변경
- 클러스터 도메인에 대한 새 quorum 디바이스 작성
- 클러스터 도메인 삭제

대화식 모드에서 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 실행:

-f 매개변수를 사용하여 XML 입력 파일을 지정하지 않고 db2haicu 명령을 실행하여 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 호출할 때, 유틸리티는 대화식 모드에서 실행됩니다. 대화식 모드에서 db2haicu는 정보를 표시하고 텍스트 기반 형식으로 사용자에게 정보를 쿼리합니다.

시작하기 전에

- SLES(SUSE Linux Enterprise Server) 11에서 IBM Tivoli SA MP(System Automation for Multiplatforms) 버전 3.1에 DB2 고가용성을 사용하려면, db2haicu 도구를 실행하여 HA 환경을 작성하기 전에 먼저 SA MP 버전 3.1 Fixpack 4를 다운로드하여 설치해야 합니다. 필수 Fixpack을 다운로드하려면 http://www.ibm.com/software/tivoli/support/sys-auto-multi을 참조하십시오.
- DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하기 전에 수행해야 하는 태스크 세트가 있습니다. 자세한 정보는 126 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 전제조건』을 참조하십시오.

이 태스크에 대한 정보

db2haicu를 대화식 모드로 실행할 때 화면에 정보와 질문이 텍스트 형식으로 제공됩니다. 화면 맨 아래에 있는 프롬프트에 db2haicu가 요청하는 정보를 입력할 수 있습니다.

프로시저

db2haicu를 대화식 모드에서 실행하려면 `-f <input-file-name>` 없이 db2haicu 명령을 호출하십시오.

다음 단계

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)에는 별도의 진단 로그가 없습니다. 데이터베이스 관리 프로그램 진단 로그, db2diag 로그 파일 및 db2pd 도구를 사용하여 db2haicu 오류를 조사하고 진단할 수 있습니다. 자세한 정보는 129 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 문제점 해결』을 참조하십시오.

XML 입력 파일과 함께 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 실행:

`-f <input-file-name>` 매개변수를 db2haicu 명령과 함께 사용하여 구성 세부사항을 지정한 XML 입력 파일과 함께 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행할 수 있습니다. XML 입력 파일과 함께 db2haicu를 실행하는 것은 고가용성을 위해 구성되어야 하는 데이터베이스 파티션이 여러 개 있을 때처럼 구성 태스크를 여러 번 수행해야 할 때 유용합니다.

시작하기 전에

- SLES(SUSE Linux Enterprise Server) 11에서 IBM Tivoli SA MP(System Automation for Multiplatforms) 버전 3.1에 DB2 고가용성을 사용하려면, db2haicu 도구를 실행하여 HA 환경을 작성하기 전에 먼저 SA MP 버전 3.1 Fixpack 4를 다운로드하여 설치해야 합니다. 필수 Fixpack을 다운로드하려면 http://www.ibm.com/software/tivoli/support/sys-auto-multi을 참조하십시오.
- DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하기 전에 수행해야 하는 태스크 세트가 있습니다. 자세한 정보는 126 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 전제조건』을 참조하십시오.

이 태스크에 대한 정보

`sqllib` 디렉토리의 `samples` 서브디렉토리에는 수정하고 db2haicu와 함께 사용하여 클러스터 환경을 구성할 수 있는 샘플 XML 입력 파일 세트가 있습니다. 자세한 정보는 117 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 위한 샘플 XML 입력 파일』을 참조하십시오.

프로시저

1. XML 입력 파일을 작성하십시오.
2. `-f <input-file-name>`과 함께 db2haicu를 호출하십시오.

   db2haicu 및 사용자가 작성하는 db2haicu-input.xml이라는 입력 파일을 사용하여 현재 데이터베이스 관리 프로그램 인스턴스에 대해 클러스터 환경을 구성하려면 다음 명령을 사용하십시오.

   ```
   db2haicu -f db2haicu-input.xml
   ```

다음 단계

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)에는 별도의 진단 로그가 없습니다. 데이터베이스 관리 프로그램 진단 로그, db2diag 로그 파일 및 db2pd 도구를 사용하여 db2haicu 오류를 조사하고 진단할 수 있습니다. 자세한 정보는 129 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 문제점 해결』을 참조하십시오.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일 XML 스키마 정의:

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일 XML 스키마 정의(XSD)는 db2haicu XML 입력 파일에서 지정할 수 있는 클러스터 도메인 오브젝트를 정의합니다. 이 db2haicu XSD는 sqllib/samples/ha/xml 디렉토리의 db2ha.xsd라는 파일에 있습니다.

DB2ClusterType

db2haicu XML 스키마 정의(XSD)의 루트 요소는 유형이 DB2ClusterType인 DB2Cluster입니다. db2haicu XML 입력 파일은 DB2Cluster 요소로 시작해야 합니다.

『XML 스키마 정의』
『하위 요소』
99 페이지의 『속성』
99 페이지의 『사용 시 참고사항』

XML 스키마 정의

```
<xs:complexType name='DB2ClusterType'>
  <xs:sequence>
    <xs:element name='DB2ClusterTemplate' type='DB2ClusterTemplateType' minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='ClusterDomain'      type='ClusterDomainType'                  maxOccurs='unbounded'/>
<xs:element name='FailoverPolicy'
               type='FailoverPolicyType'
               minOccurs='0'/>
    <xs:element name='DB2PartitionSet'    type='DB2PartitionSetType'    minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='HADRDBSet'          type='HADRDBType'             minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='HADBSet'            type='HADBType'               minOccurs='0' maxOccurs='unbounded'/>
</xs:sequence>
  <xs:attribute name='clusterManagerName' type='xs:string' use='optional'/>
</xs:complexType>
```

하위 요소

DB2ClusterTemplate

유형: DB2ClusterTemplateType

사용법 참고:

db2haicu XML 입력 파일에 DB2ClusterTemplateType 요소를 포함하지 마십시오. DB2ClusterTemplateType 요소는 현재 나중에 사용하기 위해 예약됩니다.

ClusterDomain

유형: ClusterDomainType

ClusterDomainType 요소에는 클러스터 도메인(*클러스터 도메인* 노드라고도 함)의 머신 또는 컴퓨터, *네트워크 등가성*(서로에 대해 장애 복구할 수 있는 네트워크 그룹) 및 *Quorum 디바이스*(균형을 깨는 메커니즘)에 대한 권장 스펙이 들어 있습니다.

어커런스 규칙:

DB2ClusterType 요소에 하나 이상의 ClusterDomain 요소를 포함해야 합니다.

FailoverPolicy

유형: FailoverPolicyType

FailoverPolicyType 요소는 클러스터 관리 프로그램이 클러스터 도메인과 함께 사용해야 하는 *장애 복구 규정*을 지정합니다.

어커런스 규칙:

DB2ClusterType 요소에 0 또는 하나의 FailoverPolicy 요소를 포함할 수 있습니다.

DB2PartitionSet

유형: DB2PartitionSetType

DB2PartitionSetType 요소는 데이터베이스 파티션에 관한 정보를 포함합니다. DB2PartitionSetType 요소는 파티션된 데이터베이스 환경에서만 적용할 수 있습니다.

어커런스 규칙:

db2haicu db2haicu XML 스키마 정의에 따라서 DB2ClusterType 요소에 0개 이상의 DB2PartitionSet 요소를 포함할 수 있습니다.

HADRDBSet

유형: HADRDBType

HADRDBType 요소에는 고가용성 재해 복구(HADR) 기본 및 대기 데이터베이스 쌍의 목록이 들어 있습니다.

어커런스 규칙:

db2haicu db2haicu XML 스키마 정의에 따라서 DB2ClusterType 요소에 0개 이상의 HADRDBSet 요소를 포함할 수 있습니다.

사용법 참고:

- 파티션된 데이터베이스 환경에서는 HADRDBSet를 포함하지 않아야 합니다.
- HADRDBSet를 포함하는 경우, FailoverPolicy 요소에서 HADRFailover의 장애 복구 규정을 지정해야 합니다.

HADBSet

유형: HADBType

HADBType 요소는 클러스터 도메인에 포함하고 고가용성으로 만들 데이터베이스 목록을 포함합니다.

어커런스 규칙:

db2haicu db2haicu XML 스키마 정의에 따라서 DB2ClusterType 요소에 0개 이상의 HADBSet 요소를 포함할 수 있습니다.

속성

clusterManagerName(선택적)

clusterManagerName 속성은 클러스터 관리 프로그램을 지정합니다.

이 속성에 대해 유효한 값은 다음 표에서 지정됩니다.

표 3. clusterManager 속성에 대해 유효한 값

clusterManagerName value	클러스터 관리 프로그램 제품
TSA	IBM Tivoli SA MP(System Automation for Multiplatforms)

사용 시 참고사항

단일 파티션 데이터베이스 환경에서는 대개 각 데이터베이스 관리 프로그램 인스턴스에 대해 하나의 클러스터 도메인만을 작성합니다.

다중 파티션 데이터베이스 환경에 대해 가능한 구성은 다음과 같습니다.

- FailoverPolicy 요소를 Mutual로 설정하십시오.
- DB2PartitionSet의 DB2Partition 하위 요소에서 MutualPair 요소를 사용하여 단일 클러스터 도메인에 있는 두 클러스터 도메인 노드를 지정하십시오.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 ClusterDomainType XML 스키마 정의:

ClusterDomainType 요소에는 클러스터 도메인(*클러스터 도메인* 노드라고도 함)의 머신 또는 컴퓨터, *네트워크 등가성*(서로에 대해 장애 복구할 수 있는 네트워크 그룹) 및 *Quorum 디바이스*(균형을 깨는 메커니즘)에 대한 권장 스펙이 들어 있습니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
101 페이지의 『속성』

상위 요소

다음 유형의 요소가 ClusterDomainType 하위 요소를 포함합니다.

- DB2ClusterType

XML 스키마 정의

```
<xs:complexType name='ClusterDomainType'>
  <xs:sequence>
    <xs:element name='Quorum'
                type='QuorumType'
                minOccurs='0'/>
    <xs:element name='PhysicalNetwork' type='PhysicalNetworkType' minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='ClusterNode'     type='ClusterNodeType'                  maxOccurs='unbounded'/>
</xs:sequence>
  <xs:attribute name='domainName'      type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

Quorum

유형: QuorumType

QuorumType 요소는 클러스터 도메인에 대한 *Quorum 디바이스(quorum device)*를 지정합니다.

어커런스 규칙:

ClusterDomainType 요소에 0 또는 하나의 Quorum 요소를 포함할 수 있습니다.

PhysicalNetwork

유형: PhysicalNetworkType

PhysicalNetworkType 요소는 서로에 대해 장애 복구할 수 있는 네트워크 인터페이스 카드를 포함합니다. 이 종류의 네트워크를 *네트워크 등가성*이라고도 합니다.

어커런스 규칙:

ClusterDomainType 요소에 0개 이상의 PhysicalNetwork 요소를 포함할 수 있습니다.

ClusterNode

유형: ClusterNodeType

ClusterNodeType 요소에는 클러스터의 특정 컴퓨터나 머신(*클러스터 도메인 노드*라고도 하는)에 관한 정보가 들어 있습니다.

어커런스 규칙:

ClusterDomainType 요소에 최소한 하나의 ClusterNode 요소를 지정해야 합니다.

사용 시 참고사항

IBM Tivoli SA MP(System Automation for Multiplatforms)는 최대 32개
클러스터 도메인 노드를 지원합니다. 클러스터 관리 프로그램이 SA MP인 경우, ClusterDomainType 요소에 최대 32개 ClusterNode 요소를 포함할 수 있습니다.

속성

domainName(필수)

ClusterDomainType 요소에 대해 고유한 이름을 지정해야 합니다.

RSCT(Reliable Scalable Cluster Technology)를 사용하여 클러스터를 관리 중인 경우 다음 제한사항이 domainName에 적용됩니다.

- domainName은 문자 A - Z, a - z, 숫자 0 - 9, 마침표(.) 및 밑줄(_)만 포함할 수 있습니다.
- domainName은 "IW"일 수 없습니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 QuorumType XML 스키마 정의:

QuorumType 요소는 클러스터 도메인에 대한 *Quorum 디바이스(quorum device)*를 지정합니다.

『상위 요소』
『XML 스키마 정의』
102 페이지의 『하위 요소』
102 페이지의 『속성』

상위 요소

다음 유형의 요소가 QuorumType 하위 요소를 포함합니다.

- ClusterDomainType

XML 스키마 정의

```
<xs:complexType name='QuorumType'>
  <xs:attribute name='quorumDeviceProtocol' type='QuorumDeviceProtocolType' use='required'/>
<xs:attribute name='quorumDeviceName'     type='xs:string'               use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

quorumDeviceProtocol(필수)

quorumDeviceProtocol은 사용할 정족수의 유형을 지정합니다.

*Quorum 디바이스*는 클러스터 관리 프로그램이 정상적인 결정 프로세스에서 명확한 선택을 하지 못한 경우 클러스터 관리 결정을 하는 데 도움을 제공합니다. 클러스터 관리 프로그램이 가능한 여러 조치 사이에서 선택해야 하는 경우 클러스터 관리 프로그램은 각각의 가능한 조치를 지원하는 클러스터 도메인 노드 수를 계산한 후 대다수의 클러스터 도메인 노드에서 지원되는 조치를 선택합니다. 정확하게 동일한 클러스터 도메인 노드 수가 두 개 이상의 선택사항을 지원하는 경우, 클러스터 관리 프로그램은 quorum 디바이스를 참조하여 선택합니다.

quorumDeviceProtocol 속성의 유형은 QuorumDeviceProtocolType입니다.

다음은 QuorumDeviceProtocolType의 XML 스키마 정의입니다.

```
<xs:simpleType name='QuorumDeviceProtocolType'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='disk'/>
    <xs:enumeration value='scsi'/>
    <xs:enumeration value='network'/>
    <xs:enumeration value='eckd'/>
    <xs:enumeration value='mns'/>
  </xs:restriction>
</xs:simpleType>
```

이 속성에 대해 현재 지원되는 값은 다음 표에서 지정됩니다.

표 4. quorumDeviceProtocol 속성에 대해 유효한 값

quorumDeviceProtocol 값	**의미**
network	네트워크 quorum 디바이스는 모든 클러스터 도메인 노드가 언제나 연결할 수 있는 IP 주소입니다.

quorumDeviceName(필수)

quorumDeviceName의 값이 quorumDeviceProtocol에 지정된 정족수 유형에 의존합니다.

이 속성에 대해 유효한 값은 다음 표에서 지정됩니다.

표 5. quorumDeviceName 속성에 대해 유효한 값

quorumDeviceProtocol의 값	**quorumDeviceName에 대해 유효한 값**
network	적절하게 형식화된 IP 주소를 포함하는 문자열입니다. 예를 들어, 다음과 같습니다. 12.126.4.5 네트워크 Quorum 디바이스로서 유효하다고 지정하는 IP 주소에 대해, 각 클러스터 도메인 노드가 이 IP 주소에 액세스할 수 있어야 합니다(예를 들어 ping 유틸리티를 사용하여).

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 PhysicalNetworkType XML 스키마 정의:

PhysicalNetworkType 요소는 서로에 대해 장애 복구할 수 있는 네트워크 인터페이스 카드를 포함합니다. 이 종류의 네트워크를 *네트워크 등가성*이라고도 합니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
104 페이지의 『속성』

상위 요소

다음 유형의 요소가 PhysicalNetworkType 하위 요소를 포함합니다.

- ClusterDomainType

XML 스키마 정의

```
<xs:complexType name='PhysicalNetworkType'>
  <xs:sequence>
    <xs:element name='Interface'              type='InterfaceType' minOccurs='1' maxOccurs='unbounded'/>
<xs:element name='LogicalSubnet'          type='IPAddressType' minOccurs='0' maxOccurs='unbounded'/>
</xs:sequence>
  <xs:attribute name='physicalNetworkName'     type='xs:string'                   use='required'/>
<xs:attribute name='physicalNetworkProtocol' type='PhysicalNetworkProtocolType' use='required'/>
</xs:complexType>
```

하위 요소

Interface

유형: InterfaceType

InterfaceType 요소는 IP 주소, 네트워크에 있는 컴퓨터나 머신(*클러스터 도메인* 노드라고도 함)의 이름 및 해당 클러스터 도메인 노드의 *네트워크 인터페이스 카드*(NIC) 이름으로 구성됩니다.

어커런스 규칙:

PhysicalNetworkType 요소에 하나 이상의 Interface 요소를 지정해야 합니다.

LogicalSubnet

유형: IPAddressType

IPAddressType 요소에는 *기본 주소*, *서브넷 마스크* 및 IP 주소가 속하는 네트워크의 이름 같이 IP 주소의 모든 세부사항이 들어 있습니다.

어커런스 규칙:

PhysicalNetworkType 요소에 0개 이상의 LogicalSubnet 요소를 포함할 수 있습니다.

속성

physicalNetworkName(필수)

각 PhysicalNetworkType 요소에 대해 고유한 physicalNetworkName을 지정해야 합니다.

physicalNetworkProtocol(필수)

physicalNetworkProtocol 속성의 유형은 PhysicalNetworkProtocolType입니다.

다음은 PhysicalNetworkProtocolType 요소에 대한 XML 스키마 정의입니다.

```
<xs:simpleType name='PhysicalNetworkProtocolType'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='ip'/>
    <xs:enumeration value='rs232'/>
    <xs:enumeration value='scsi'/>
    <xs:enumeration value='ssa'/>
    <xs:enumeration value='disk'/>
  </xs:restriction>
</xs:simpleType>
```

이 속성에 대해 현재 지원되는 값은 다음 표에서 지정됩니다.

표 6. physicalNetworkProtocol 속성에 대한 유효한 값

physicalNetworkProtocol 값	**의미**
ip	TCP/IP 프로토콜

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 InterfaceType XML 스키마 정의:

InterfaceType 요소는 IP 주소, 네트워크에 있는 컴퓨터나 머신(*클러스터 도메인 노드*라고도 함)의 이름 및 해당 클러스터 도메인 노드의 *네트워크 인터페이스 카드*(NIC) 이름으로 구성됩니다.

105 페이지의 『상위 요소』
105 페이지의 『XML 스키마 정의』
105 페이지의 『하위 요소』
105 페이지의 『속성』

상위 요소

다음 유형의 요소가 InterfaceType 하위 요소를 갖습니다.

- PhysicalNetworkType

XML 스키마 정의

```
<xs:complexType name='InterfaceType'>
  <xs:sequence>
    <xs:element name='IPAddress'       type='IPAddressType'/>
  </xs:sequence>
  <xs:attribute name='interfaceName'   type='xs:string' use='required'/>
  <xs:attribute name='clusterNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

IPAddress

유형: IPAddressType

IPAddressType 요소에는 *기본 주소*, *서브넷 마스크* 및 IP 주소가 속하는 네트워크의 이름 같이 IP 주소의 모든 세부사항이 들어 있습니다.

어커런스 규칙:

InterfaceType 요소에 IPAddress를 하나만 지정해야 합니다.

속성

interfaceName(필수)

interfaceName 속성에 NIC의 이름을 지정해야 합니다. 사용자가 interfaceName에 지정하는 NIC가 clusterNodeName 속성에 지정하는 클러스터 도메인 노드에 존재해야 합니다.

clusterNodeName(필수)

IPAddress 요소에 지정하는 IP 주소에 위치하는 클러스터 도메인 노드의 이름을 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 IPAddressType XML 스키마 요소:

IPAddressType 요소에는 *기본 주소*, *서브넷 마스크* 및 IP 주소가 속하는 네트워크의 이름 같이 IP 주소의 모든 세부사항이 들어 있습니다.

106 페이지의『상위 요소』
106 페이지의『XML 스키마 정의』
106 페이지의『하위 요소』
106 페이지의『속성』

상위 요소

다음 유형의 요소는 IPAddressType 하위 요소를 갖습니다.

- PhysicalNetworkType
- InterfaceType
- DB2PartitionType

XML 스키마 정의

```
<xs:complexType name='IPAddressType'>
  <xs:attribute name='baseAddress' type='xs:string' use='required'/>
  <xs:attribute name='subnetMask'  type='xs:string' use='required'/>
  <xs:attribute name='networkName' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

baseAddress(필수)

유효한 IP 주소 형식(마침표로 구분되는 0 - 255 범위에 있는 숫자의 4 세트)이 있는 문자열을 사용하여 기본 IP 주소를 지정해야 합니다. 예를 들어, 다음과 같습니다.

```
162.148.31.101
```

subnetMask(필수)

유효한 IP 주소 형식이 있는 문자열을 사용하여 기본 IP 주소를 지정해야 합니다.

networkName(필수)

이 IPAddress 요소를 포함하는 PhysicalNetworkType 요소의 physicalNetworkName 속성에 지정한 대로 여기에 networkName에 대해 동일한 값을 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 ClusterNodeType XML 스키마 정의:

ClusterNodeType 요소에는 클러스터의 특정 컴퓨터나 머신(*클러스터 도메인* 노드라고도 하는)에 관한 정보가 들어 있습니다.

107 페이지의 『상위 요소』
107 페이지의 『XML 스키마 정의』
107 페이지의 『하위 요소』
107 페이지의 『속성』

상위 요소

다음 유형의 요소가 ClusterNodeType 요소를 갖습니다.

- ClusterDomainType

XML 스키마 정의

```
<xs:complexType name='ClusterNodeType'>
  <xs:attribute name='clusterNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

clusterNodeName(필수)

클러스터 도메인 노드의 이름을 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 FailoverPolicyType XML 스키마 정의:

FailoverPolicyType 요소는 클러스터 관리 프로그램이 클러스터 도메인과 함께 사용해야 하는 *장애 복구 규정*을 지정합니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
108 페이지의 『가능한 값』

상위 요소

다음 유형의 요소가 InterfaceType 하위 요소를 포함합니다.

- DB2ClusterType

XML 스키마 정의

```
<xs:complexType name='FailoverPolicyType'>
  <xs:choice>
    <xs:element name='RoundRobin'   type='xs:string' minOccurs='0' />
<xs:element name='Mutual'       type='xs:string' minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='NPlusM'       type='xs:string' minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='LocalRestart' type='xs:string' fixed=''/>
<xs:element name='HADRFailover' type='xs:string' fixed=''/>
<xs:element name='Custom'       type='xs:string' minOccurs='0' />
</xs:choice>
</xs:complexType>
```

하위 요소

없음.

가능한 값

클러스터 도메인 중 임의의 위치에서 실패가 있는 경우 사용할 장애 복구 규정의 유형을 클러스터 관리 프로그램에 지정하려면 다음 선택사항 중 하나를 선택하십시오.

*장애 복구 규정*은 네트워크 인터페이스 카드 또는 데이터베이스 서버와 같은 클러스터 요소가 실패할 때 클러스터 관리 프로그램이 응답해야 하는 방법을 지정합니다. 일반적으로, 클러스터 관리 프로그램은 이전에 클러스터 관리 프로그램에 대해 실패한 요소에 적절한 교체로 식별된 대체 요소로 실패한 요소의 워크로드를 전송합니다. 실패한 요소에서 보조 요소로의 이러한 워크로드 전송을 *장애 복구*라고 합니다.

RoundRobin

*라운드 로빈 장애 복구 규정*을 사용 중일 때, 클러스터 도메인에 있는 한 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함)와 연관되는 실패가 있는 경우 데이터베이스 관리 프로그램은 클러스터 도메인에 있는 다른 노드의 실패한 클러스터 도메인 노드에서 작업을 재시작합니다.

Mutual

*상호 장애 복구 규정*을 구성하려면 클러스터 도메인에 있는 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함) 쌍을 시스템 쌍으로 연관시킵니다. 이 쌍에 있는 노드 중 하나에서 실패가 발생하는 경우 실패한 노드의 데이터베이스 파티션은 쌍의 다른 노드로 장애 복구합니다. 상호 장애 복구는 여러 개의 데이터베이스 파티션이 있는 경우에만 사용 가능합니다.

NPlusM

*N Plus M 장애 복구 규정*을 사용 중일 때, 클러스터 도메인에 있는 한 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함)와 연관되는 실패가 있는 경우 실패한 노드의 데이터베이스 파티션은 클러스터 도메인에 있는 다른 노드로 장애 복구합니다. N Plus M 장애 복구는 여러 개의 데이터베이스 파티션이 있는 경우에만 사용 가능합니다.

LocalRestart

*로컬 재시작 장애 복구 규정*을 사용 중일 때, 클러스터 도메인에 있는 한 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함)와 연관되는 실패가 있는 경우 데이터베이스 관리 프로그램은 실패한 동일 노드의 제 위치에서(또는 로컬로) 데이터베이스를 재시작합니다.

HADRFailover

*HADR 장애 복구 규정*을 구성하는 경우, DB2 고가용성 재해 복구(HADR) 기능이 장애 복구를 관리하도록 할 수 있습니다. HADR 기본 데이터베이스가 실패하면, 데이터베이스 관리 프로그램은 실패한 데이터베이스에서 HADR 대기 데이터베이스로 워크로드를 이동합니다.

Custom

*사용자 정의 장애 복구 규정*을 구성한 경우 클러스터 도메인에서 데이터베이스

관리 프로그램이 장애 복구할 수 있는 컴퓨터(*클러스터 도메인* 노드 또는 단순히 노드라고도 함) 목록을 작성합니다. 클러스터 도메인의 노드가 실패하면 데이터베이스 관리 프로그램은 실패한 노드에서, 사용자가 지정한 목록에 있는 노드 중 하나로 워크로드를 이동합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 DB2PartitionSetType XML 스키마 정의:

DB2PartitionSetType 요소는 데이터베이스 파티션에 관한 정보를 포함합니다. DB2PartitionSetType 요소는 파티션된 데이터베이스 환경에서만 적용할 수 있습니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
『속성』

상위 요소

InterfaceType은 다음의 하위 요소입니다.

- PhysicalNetworkType

XML 스키마 정의

```
<xs:complexType name='DB2PartitionSetType'>
  <xs:sequence>
    <xs:element name='DB2Partition'
    type='DB2PartitionType' maxOccurs='unbounded'/>
</xs:sequence>
</xs:complexType>
```

하위 요소

DB2Partition

유형: DB2PartitionType

DB2PartitionType 요소는 데이터베이스 파티션이 속하는 DB2 데이터베이스 관리 프로그램 인스턴스를 포함하는 데이터베이스 파티션 및 데이터베이스 파티션 번호를 지정합니다.

어커런스 규칙:

DB2PartitionSetType 요소에 하나 이상의 DB2Partition 요소를 지정해야 합니다.

속성

없음.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 DB2PartitionType XML 스키마 요소:

DB2PartitionType 요소는 데이터베이스 파티션이 속하는 DB2 데이터베이스 관리 프로그램 인스턴스를 포함하는 데이터베이스 파티션 및 데이터베이스 파티션 번호를 지정합니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
111 페이지의 『속성』

상위 요소

InterfaceType은 다음의 하위 요소입니다.

- DB2PartitionSetType

XML 스키마 정의

```
<xs:complexType name='DB2PartitionType'>
  <xs:sequence>
    <xs:element name='VirtualIPAddress' t
    ype='IPAddressType'    minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='Mount'
type='MountType'        minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='HADRDB'
type='HADRDBType'       minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='MutualPair'
type='MutualPolicyType' minOccurs='0' maxOccurs='1'/>
<xs:element name='NPlusMNode'
type='NPlusMPolicyType' minOccurs='0' maxOccurs='unbounded'/>
</xs:sequence>
  <xs:attribute name='instanceName'
  type='xs:string'  use='required'/>
  <xs:attribute name='dbpartitionnum'
  type='xs:integer' use='required'/>
</xs:complexType>
```

하위 요소

VirtualIPAddress

유형: IPAddressType

IPAddressType 요소에는 *기본 주소*, *서브넷 마스크* 및 IP 주소가 속하는 네트워크의 이름 같이 IP 주소의 모든 세부사항이 들어 있습니다.

VirtualIPAddress 포함을 생략할 수 있거나, DB2PartitionType 요소에 한정되지 않은 수의 VirtualIPAddress 요소를 포함할 수 있습니다.

Mount

유형: MountType

MountType 요소에는 마운트된 파일의 위치를 식별하는 파일 경로 같이 *마운트 지점*에 관한 정보가 들어 있습니다.

Mount 포함을 생략할 수 있거나, DB2PartitionType 요소에 한정되지 않은 수의 Mount 요소를 포함할 수 있습니다.

HADRDB

유형: HADRDBType

HADRDBType 요소에는 고가용성 재해 복구(HADR) 기본 및 대기 데이터베이스 쌍의 목록이 들어 있습니다.

HADRDB 포함을 생략할 수 있거나, DB2PartitionType 요소에 한정되지 않은 수의 HADRDB 요소를 포함할 수 있습니다.

MutualPair

유형: MutualPolicyType

MutualPolicyType 요소에는 서로에 대해 장애 복구할 수 있는 클러스터 도메인 노드의 쌍에 대한 정보가 들어 있습니다.

MutualPair 포함을 생략할 수 있거나, DB2PartitionType 요소에 한정되지 않은 수의 MutualPair 요소를 포함할 수 있습니다.

NPlusMNode

유형: NPlusMPolicyType

NPlusMNode 포함을 생략할 수 있거나, DB2PartitionType 요소에 한정되지 않은 수의 NPlusMNode 요소를 포함할 수 있습니다.

속성

instanceName(필수)

instanceName 속성에서 이 DB2PartitionType 요소가 연관되는 DB2 데이터베이스 관리 프로그램 인스턴스를 지정해야 합니다.

dbpartitionnum(필수)

dbpartitionnum 속성에서 데이터베이스 파티션을 고유하게 식별하는 데이터베이스 파티션 번호(예를 들어 db2nodes.cfg 파일에서 지정된 dbpartitionnum 번호)를 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 MountType XML 스키마 정의:

MountType 요소에는 마운트된 파일의 위치를 식별하는 파일 경로 같이 *마운트 지점*에 관한 정보가 들어 있습니다.

112 페이지의 『상위 요소』
112 페이지의 『XML 스키마 정의』
112 페이지의 『하위 요소』
112 페이지의 『속성』

상위 요소

다음 유형의 요소가 MountType 하위 요소를 포함합니다.

- DB2PartitionType

XML 스키마 정의

```
<xs:complexType name='MountType'>
  <xs:attribute name='filesystemPath' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

filesystemPath(필수)

파일 시스템이 마운트될 때 마운트 지점과 연관된 경로를 지정하십시오.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 MutualPolicyType XML 스키마 정의:

MutualPolicyType 요소에는 서로에 대해 장애 복구할 수 있는 클러스터 도메인 노드의 쌍에 대한 정보가 들어 있습니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
113 페이지의 『속성』

상위 요소

다음 유형의 요소가 MutualPolicyType 하위 요소를 포함합니다.

- DB2PartitionType

XML 스키마 정의

```
<xs:complexType name='MutualPolicyType'>
  <xs:attribute name='systemPairNode1' type='xs:string' use='required'/>
  <xs:attribute name='systemPairNode2' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

systemPairNode1(필수)

systemPairNode1에서, systemPairNode2에서 사용자가 지정하는 클러스터 도메인 노드에 대해 장애 복구할 수 있는 클러스터 도메인 노드의 이름을 지정해야 합니다.

systemPairNode2(필수)

systemPairNode2에서, systemPairNode1에서 지정하는 클러스터 도메인 노드에 대해 장애 복구할 수 있는 클러스터 도메인 노드의 이름을 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 NPlusMPolicyType XML 스키마 정의:

『상위 요소』
『XML 스키마 정의』
『하위 요소』
『속성』

상위 요소

다음 유형의 요소가 NPlusMPolicyType 하위 요소를 포함합니다.

- DB2PartitionType

XML 스키마 정의

```
<xs:complexType name='NPlusMPolicyType'>
  <xs:attribute name='standbyNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

standbyNodeName(필수)

standbyNodeName 요소에서, 이 NPlusMPolicyType 요소를 포함하는 파티션이 장애 복구할 수 있는 클러스터 도메인 노드의 이름을 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 HADRDBType XML 스키마 정의:

HADRDBType 요소에는 고가용성 재해 복구(HADR) 기본 및 대기 데이터베이스 쌍의 목록이 들어 있습니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
『속성』
115 페이지의 『사용 시 참고사항』
115 페이지의 『제한사항』

상위 요소

다음 유형의 요소가 HADRDBType 하위 요소를 포함합니다.

- DB2ClusterType
- DB2PartitionType

XML 스키마 정의

```
<xs:complexType name='HADRDBType'>
  <xs:sequence>
    <xs:element name='VirtualIPAddress' type='IPAddressType'
    minOccurs='0' maxOccurs='unbounded'/>
<xs:element name='HADRDB'           type='HADRDBDefn'
maxOccurs='unbounded'/>
</xs:sequence>
</xs:complexType>
```

하위 요소

VirtualIPAddress

유형: IPAddressType

IPAddressType 요소에는 *기본 주소*, *서브넷 마스크* 및 IP 주소가 속하는 네트워크의 이름 같이 IP 주소의 모든 세부사항이 들어 있습니다.

어커런스 규칙:

HADRDBType 요소에 0개 이상의 VirtualIPAddress 요소를 포함할 수 있습니다.

HADRDB

유형: HADRDBDefn

HADRDBDefn 요소는 고가용성 재해 복구(HADR) 기본 및 대기 데이터베이스 쌍에 관한 정보를 포함합니다.

어커런스 규칙:

HADRDBType 요소에 하나 이상의 VirtualIPAddress 요소를 포함할 수 있습니다.

속성

없음.

사용 시 참고사항

주어진 클러스터 도메인에 대한 스펙에 HADRDBType 요소를 포함하는 경우 동일한 클러스터 도메인 스펙에 HADRFailover를 지정하는 FailoverPolicy 요소도 포함해야 합니다.

제한사항

파티션된 데이터베이스 환경에서는 HADRDBType 요소를 사용할 수 없습니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 HADRDBDefn XML 스키마 정의:

HADRDBDefn 요소는 고가용성 재해 복구(HADR) 기본 및 대기 데이터베이스 쌍에 관한 정보를 포함합니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
『속성』

상위 요소

다음 유형의 요소가 HADRDBDefn 하위 요소를 포함합니다.

- HADRDBType

XML 스키마 정의

```
<xs:complexType name='HADRDBDefn'>
  <xs:attribute name='databaseName'      type='xs:string' use='required'/>
  <xs:attribute name='localInstance'     type='xs:string' use='required'/>
  <xs:attribute name='remoteInstance'    type='xs:string' use='required'/>
  <xs:attribute name='localHost'         type='xs:string' use='required'/>
  <xs:attribute name='remoteHost'        type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

databaseName(필수)

HADR 데이터베이스의 이름을 입력하십시오.

localInstance(필수)

localInstance는 HADR 기본 데이터베이스의 데이터베이스 관리 프로그램 인스턴스입니다.

remoteInstance(필수)

remoteInstance는 HADR 대기 데이터베이스의 데이터베이스 관리 프로그램 인스턴스입니다.

localHost(필수)

localHost는 HADR 기본 데이터베이스가 위치하는 클러스터 도메인 노드의 호스트 이름입니다.

remoteHost(필수)

remoteHost는 HADR 대기 데이터베이스가 위치하는 클러스터 도메인 노드의 호스트 이름입니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 HADBType XML 스키마 정의:

HADBType 요소는 클러스터 도메인에 포함하고 고가용성으로 만들 데이터베이스 목록을 포함합니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
117 페이지의 『속성』

상위 요소

다음 유형의 요소가 HADBType 하위 요소를 포함합니다.

- DB2ClusterType

XML 스키마 정의

```
<xs:complexType name='HADBType'>
  <xs:sequence>
    <xs:element name='HADB'
    type='HADBDefn' maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='instanceName'
  type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

HADB

유형: HADBDefn

HADBDefn 요소는 클러스터 도메인에 포함되고 고가용성이 될 데이터베이스를 설명합니다.

어커런스 규칙:

HADBType 요소에 하나 이상의 HADB 요소를 포함해야 합니다.

속성

instanceName(필수)

instanceName 속성에서 HADB 속성에 지정되는 데이터베이스가 속하는 DB2 데이터베이스 관리 프로그램 인스턴스를 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 입력 파일에 대한 HADBDefn XML 스키마 요소:

HADBDefn 요소는 클러스터 도메인에 포함되고 고가용성이 될 데이터베이스를 설명합니다.

『상위 요소』
『XML 스키마 정의』
『하위 요소』
『속성』

상위 요소

HADBDefn은 다음의 하위 요소입니다.

- HADRDBType

XML 스키마 정의

```
<xs:complexType name='HADBDefn'>
  <xs:attribute name='databaseName' type='xs:string' use='required'/>
</xs:complexType>
```

하위 요소

없음.

속성

databaseName(필수)

databaseName 속성에 정확하게 하나의 데이터베이스 이름을 지정해야 합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 위한 샘플 XML 입력 파일:

sqllib 디렉토리의 samples 서브디렉토리에는 수정하고 db2haicu와 함께 사용하여 클러스터 환경을 구성할 수 있는 샘플 XML 입력 파일 세트가 있습니다.

db2ha_sample_sharedstorage_mutual.xml:

샘플 파일 db2ha_sample_sharedstorage_mutual.xml은 새 *클러스터 도메인*을 지정하기 위해 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)로 전달하는 XML 입력 파일의 예입니다. db2ha_sample_sharedstorage_mutual.xml은 sqllib/samples/ha/xml 디렉토리에 있습니다.

기능

db2ha_sample_sharedstorage_mutual.xml 샘플은 XML 입력 파일과 함께 db2haicu 를 사용하여 다음 세부사항으로 클러스터 도메인을 정의하는 방법을 보여줍니다.

- Quorum 디바이스: 네트워크
- 클러스터의 컴퓨터(클러스터 도메인 노드): 2
- 장애 복구 규정: mutual
- 데이터베이스 파티션: 1
- 가상(서비스) IP 주소: 1
- 장애 복구를 위한 공유 마운트 지점: 1

XML 소스

```
<!-- ================================================================ -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================ -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================ -->
  <!-- = Create a cluster domain named db2HAdomain.                     = -->
  <!-- ================================================================ -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================ -->
    <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
    <!-- = The IP must be pingable at all times by each of the cluster = -->
    <!-- = domain nodes.                                                = -->
    <!-- ================================================================ -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================ -->
    <!-- = Create a network named db2_public_network_0 with an IP      = -->
    <!-- = network protocol.                                            = -->
    <!-- = This network contains two computers: hasys01 and hasys02.   = -->
    <!-- = Each computer has one network interface card (NIC) called   = -->
    <!-- = eth0.                                                        = -->
    <!-- = The IP address of the NIC on hasys01 is 19.126.52.139       = -->
    <!-- = The IP address of the NIC on hasys02 is 19.126.52.140       = -->
    <!-- ================================================================ -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                     physicalNetworkProtocol="ip">

      <Interface interfaceName="eth0" clusterNodeName="hasys01">
        <IPAddress baseAddress="19.126.52.139"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

      <Interface interfaceName="eth0" clusterNodeName="hasys02">
        <IPAddress baseAddress="19.126.52.140"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

    </PhysicalNetwork>

    <!-- ================================================================ -->
    <!-- = List the computers (cluster nodes) in the cluster domain.   = -->
    <!-- ================================================================ -->
    <ClusterNode clusterNodeName="hasys01"/>
```

```
    <ClusterNode clusterNodeName="hasys02"/>

  </ClusterDomain>


  <!-- ================================================================= -->
  <!-- = The failover policy specifies the order in which the cluster  = -->
  <!-- = domain nodes should fail over.                                = -->
  <!-- ================================================================= -->
  <FailoverPolicy>
     <Mutual />
  </FailoverPolicy>


  <!-- ================================================================= -->
  <!-- = Specify all the details of the database partition             = -->
  <!-- ================================================================= -->
  <DB2PartitionSet>

    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
       <VirtualIPAddress baseAddress="19.126.52.222"
                         subnetMask="255.255.255.0"
                         networkName="db2_public_network_0"/>
       <Mount filesystemPath="/home/db2inst1"/>
       <MutualPair systemPairNode1="hasys01" systemPairNode2="hasys02" />
    </DB2Partition>

  </DB2PartitionSet>

</DB2Cluster>
```

db2ha_sample_DPF_mutual.xml:

샘플 파일 db2ha_sample_DPF_mutual.xml은 새 *클러스터 도메인*을 지정하기 위해 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)로 전달하는 XML 입력 파일의 예입니다. db2ha_sample_DPF_mutual.xml은 sqllib/samples/ha/xml 디렉토리에 있습니다.

기능

db2ha_sample_DPF_mutual.xml 샘플은 XML 입력 파일과 함께 db2haicu를 사용하여 다음 세부사항으로 클러스터 도메인을 정의하는 방법을 보여줍니다.

- Quorum 디바이스: 네트워크
- 클러스터의 컴퓨터(클러스터 도메인 노드): 4
- 장애 복구 규정: mutual
- 데이터베이스 파티션: 2
- 가상(서비스) IP 주소: 1
- 장애 복구를 위한 공유 마운트 지점: 2
- 고가용성을 위해 구성된 데이터베이스: 2

XML 소스

```
<!-- ================================================================= -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================= -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
```

```
         version="1.0">

<!-- ============================================================= -->
<!-- = Create a cluster domain named db2HAdomain.                  = -->
<!-- ============================================================= -->
<ClusterDomain domainName="db2HAdomain">

  <!-- ========================================================== -->
  <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
  <!-- = The IP must be pingable at all times by each of the cluster = -->
  <!-- = domain nodes.                                            = -->
  <!-- ========================================================== -->
  <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

  <!-- ========================================================== -->
  <!-- = Create a network named db2_public_network_0 with an IP      = -->
  <!-- = network protocol.                                        = -->
  <!-- = This network contains four computers: hasys01, hasys02,     = -->
  <!-- = hasys03, and hasys04.                                    = -->
  <!-- = Each computer has a network interface card called eth0.     = -->
  <!-- = The IP address of eth0 on hasys01 is 19.126.124.30          = -->
  <!-- = The IP address of eth0 on hasys02 is 19.126.124.31          = -->
  <!-- = The IP address of eth0 on hasys03 is 19.126.124.32          = -->
  <!-- = The IP address of eth0 on hasys04 is 19.126.124.33          = -->
  <!-- ========================================================== -->
  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth0" clusterNodeName="hasys01">
      <IPAddress baseAddress="19.126.124.30"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys02">
      <IPAddress baseAddress="19.126.124.31"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys03">
      <IPAddress baseAddress="19.126.124.32"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys04">
      <IPAddress baseAddress="19.126.124.33"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ========================================================== -->
  <!-- = Create a network named db2_private_network_0 with an IP     = -->
  <!-- = network protocol.                                        = -->
  <!-- = This network contains four computers: hasys01, hasys02,     = -->
  <!-- = hasys03, and hasys04 (same as db2_public_network_0.)        = -->
  <!-- = In addition to eth0, each computer has a network interface  = -->
  <!-- = card called eth1.                                        = -->
  <!-- = The IP address of eth1 on hasys01 is 192.168.23.101         = -->
  <!-- = The IP address of eth1 on hasys02 is 192.168.23.102         = -->
  <!-- = The IP address of eth1 on hasys03 is 192.168.23.103         = -->
  <!-- = The IP address of eth1 on hasys04 is 192.168.23.104         = -->
  <!-- ========================================================== -->
  <PhysicalNetwork physicalNetworkName="db2_private_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth1" clusterNodeName="hasys01">
      <IPAddress baseAddress="192.168.23.101"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys02">
      <IPAddress baseAddress="192.168.23.102"
```

```
                subnetMask="255.255.255.0"
                networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys03">
      <IPAddress baseAddress="192.168.23.103"
                subnetMask="255.255.255.0"
                networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys04">
      <IPAddress baseAddress="192.168.23.104"
                subnetMask="255.255.255.0"
                networkName="db2_private_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================= -->
  <!-- = List the computers (cluster nodes) in the cluster domain.   = -->
  <!-- ================================================================= -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>
  <ClusterNode clusterNodeName="hasys03"/>
  <ClusterNode clusterNodeName="hasys04"/>

</ClusterDomain>


<!-- ================================================================= -->
<!-- = The failover policy specifies the order in which the cluster  = -->
<!-- = domain nodes should fail over.                                = -->
<!-- ================================================================= -->
<FailoverPolicy>
   <Mutual />
</FailoverPolicy>


<!-- ================================================================= -->
<!-- = Specify all the details of the database partitions.           = -->
<!-- ================================================================= -->
<DB2PartitionSet>

  <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
     <VirtualIPAddress baseAddress="19.126.124.251"
                       subnetMask="255.255.255.0"
                       networkName="db2_public_network_0"/>
     <Mount filesystemPath="/hafs/db2inst1/NODE0000"/>
     <MutualPair systemPairNode1="hasys01" systemPairNode2="hasys02" />
  </DB2Partition>

  <DB2Partition dbpartitionnum="1" instanceName="db2inst1">
     <Mount filesystemPath="/hafs/db2inst1/NODE0001"/>
     <MutualPair systemPairNode1="hasys02" systemPairNode2="hasys01" />
  </DB2Partition>

  <DB2Partition dbpartitionnum="2" instanceName="db2inst1">
     <Mount filesystemPath="/hafs/db2inst1/NODE0002"/>
     <MutualPair systemPairNode1="hasys03" systemPairNode2="hasys04" />
  </DB2Partition>

  <DB2Partition dbpartitionnum="3" instanceName="db2inst1">
     <Mount filesystemPath="/hafs/db2inst1/NODE0003"/>
     <MutualPair systemPairNode1="hasys04" systemPairNode2="hasys03" />
  </DB2Partition>

</DB2PartitionSet>


<!-- ================================================================= -->
<!-- = List of databases to be configured for High Availability      = -->
<!-- ================================================================= -->
<HADBSet instanceName="db2inst1">
  <HADB databaseName = "SAMPLE" />
  <HADB databaseName = "MYDB" />
```

```
  </HADBSet>

</DB2Cluster>
```

db2ha_sample_DPF_NPlusM.xml:

샘플 파일 db2ha_sample_DPF_NPlusM.xml은 새 *클러스터 도메인*을 지정하기 위해 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)로 전달하는 XML 입력 파일의 예입니다. db2ha_sample_DPF_NPlusM.xml은 sqllib/samples/ha/xml 디렉토리에 있습니다.

기능

db2ha_sample_DPF_NPlusM.xml 샘플은 XML 입력 파일과 함께 db2haicu를 사용하여 다음 세부사항으로 클러스터 도메인을 정의하는 방법을 보여줍니다.

- Quorum 디바이스: 네트워크
- 클러스터의 컴퓨터(클러스터 도메인 노드): 4
- 장애 복구 규정: N Plus M
- 데이터베이스 파티션: 2
- 가상(서비스) IP 주소: 1
- 장애 복구를 위한 공유 마운트 지점: 4

XML 소스

```
<!-- ================================================================= -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================= -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================= -->
  <!-- = Create a cluster domain named db2HAdomain.                    = -->
  <!-- ================================================================= -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================= -->
    <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
    <!-- = The IP must be pingable at all times by each of the cluster = -->
    <!-- = domain nodes.                                               = -->
    <!-- ================================================================= -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================= -->
    <!-- = Create a network named db2_public_network_0 with an IP      = -->
    <!-- = network protocol.                                           = -->
    <!-- = This network contains four computers: hasys01, hasys02,     = -->
    <!-- = hasys03, and hasys04.                                       = -->
    <!-- = Each computer has a network interface card called eth0.     = -->
    <!-- = The IP address of eth0 on hasys01 is 19.126.124.30          = -->
    <!-- = The IP address of eth0 on hasys02 is 19.126.124.31          = -->
    <!-- = The IP address of eth0 on hasys03 is 19.126.124.32          = -->
    <!-- = The IP address of eth0 on hasys04 is 19.126.124.33          = -->
    <!-- ================================================================= -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                     physicalNetworkProtocol="ip">
```

```
    <Interface interfaceName="eth0" clusterNodeName="hasys01">
      <IPAddress baseAddress="19.126.124.30"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys02">
      <IPAddress baseAddress="19.126.124.31"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys03">
      <IPAddress baseAddress="19.126.124.32"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys04">
      <IPAddress baseAddress="19.126.124.33"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ============================================================ -->
  <!-- = Create a network named db2_private_network_0 with an IP     = -->
  <!-- = network protocol.                                          = -->
  <!-- = This network contains four computers: hasys01, hasys02,    = -->
  <!-- = hasys03, and hasys04 (same as db2_public_network_0.)       = -->
  <!-- = In addition to eth0, each computer has a network interface = -->
  <!-- = card called eth1.                                          = -->
  <!-- = The IP address of eth1 on hasys01 is 192.168.23.101        = -->
  <!-- = The IP address of eth1 on hasys02 is 192.168.23.102        = -->
  <!-- = The IP address of eth1 on hasys03 is 192.168.23.103        = -->
  <!-- = The IP address of eth1 on hasys04 is 192.168.23.104        = -->
  <!-- ============================================================ -->
  <PhysicalNetwork physicalNetworkName="db2_private_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth1" clusterNodeName="hasys01">
      <IPAddress baseAddress="192.168.23.101"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys02">
      <IPAddress baseAddress="192.168.23.102"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys03">
      <IPAddress baseAddress="192.168.23.103"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys04">
      <IPAddress baseAddress="192.168.23.104"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ============================================================ -->
  <!-- = List the computers (cluster nodes) in the cluster domain.  = -->
  <!-- ============================================================ -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>
  <ClusterNode clusterNodeName="hasys03"/>
  <ClusterNode clusterNodeName="hasys04"/>

</ClusterDomain>
```

```
  <!-- ================================================================ -->
  <!-- = The failover policy specifies the order in which the cluster  = -->
  <!-- = domain nodes should fail over.                                = -->
  <!-- ================================================================ -->
  <FailoverPolicy>
     <NPlusM />
  </FailoverPolicy>


  <!-- ================================================================ -->
  <!-- = Specify all the details of the database partitions            = -->
  <!-- ================================================================ -->
  <DB2PartitionSet>

    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
       <VirtualIPAddress baseAddress="19.126.124.250"
                         subnetMask="255.255.255.0"
                         networkName="db2_public_network_0"/>
       <Mount filesystemPath="/ha_dpf1/db2inst1/NODE0000"/>
       <Mount filesystemPath="/hafs/NODE0000"/>
       <NPlusMNode standbyNodeName="hasys03" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="1" instanceName="db2inst1">
       <Mount filesystemPath="/ha_dpf1/db2inst1/NODE0001"/>
       <Mount filesystemPath="/hafs/NODE0001"/>
       <NPlusMNode standbyNodeName="hasys04" />
    </DB2Partition>

  </DB2PartitionSet>

</DB2Cluster>
```

db2ha_sample_HADR.xml:

샘플 파일 db2ha_sample_DPF_HADR.xml은 새 *클러스터 도메인*을 지정하기 위해 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)로 전달하는 XML 입력 파일의 예입니다. db2ha_sample_HADR.xml은 sqllib/samples/ha/xml 디렉토리에 있습니다.

기능

db2ha_sample_HADR.xml 샘플은 XML 입력 파일과 함께 db2haicu를 사용하여 다음 세부사항으로 클러스터 도메인을 정의하는 방법을 보여줍니다.

- Quorum 디바이스: 네트워크
- 클러스터의 컴퓨터(클러스터 도메인 노드): 2
- 장애 복구 규정: HADR
- 데이터베이스 파티션: 1
- 가상(서비스) IP 주소: 없음
- 장애 복구를 위한 공유 마운트 지점: 없음

XML 소스

```
<!-- ================================================================ -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================ -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
```

```
          clusterManagerName="TSA"
          version="1.0">

<!-- ================================================================ -->
<!-- = Create a cluster domain named db2HAdomain.                     = -->
<!-- ================================================================ -->
<ClusterDomain domainName="db2HAdomain">

  <!-- ================================================================ -->
  <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
  <!-- = The IP must be pingable at all times by each of the cluster = -->
  <!-- = domain nodes.                                                = -->
  <!-- ================================================================ -->
  <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

  <!-- ================================================================ -->
  <!-- = Create a network named db2_public_network_0 with an IP      = -->
  <!-- = network protocol.                                            = -->
  <!-- = This network contains two computers: hasys01 and hasys02.   = -->
  <!-- = Each computer has a network interface card called eth0.     = -->
  <!-- = The IP address of eth0 on hasys01 is 19.126.52.139          = -->
  <!-- = The IP address of eth0 on hasys01 is 19.126.52.140          = -->
  <!-- ================================================================ -->
  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth0" clusterNodeName="hasys01">
      <IPAddress baseAddress="19.126.52.139"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys02">
      <IPAddress baseAddress="19.126.52.140"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================ -->
  <!-- = Create a network named db2_private_network_0 with an IP     = -->
  <!-- = network protocol.                                            = -->
  <!-- = This network contains two computers: hasys01 and hasys02.   = -->
  <!-- = In addition to eth0, each computer has a network interface  = -->
  <!-- = card called eth1.                                            = -->
  <!-- = The IP address of eth1 on hasys01 is 192.168.23.101         = -->
  <!-- = The IP address of eth1 on hasys02 is 192.168.23.102         = -->
  <!-- ================================================================ -->
  <PhysicalNetwork physicalNetworkName="db2_private_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth1" clusterNodeName="hasys01">
      <IPAddress baseAddress="192.168.23.101"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys02">
      <IPAddress baseAddress="192.168.23.102"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================ -->
  <!-- = List the computers (cluster nodes) in the cluster domain.   = -->
  <!-- ================================================================ -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>

</ClusterDomain>


<!-- ================================================================ -->
<!-- = The failover policy specifies the order in which the cluster  = -->
```

```
  <!-- = domain nodes should fail over.                                 = -->
  <!-- ================================================================ -->
  <FailoverPolicy>
     <HADRFailover />
  </FailoverPolicy>


  <!-- ================================================================ -->
  <!-- = Specify all the details of the database partitions             = -->
  <!-- ================================================================ -->
  <DB2PartitionSet>
    <DB2Partition dbpartitionnum="0" instanceName="db2inst1" />
  </DB2PartitionSet>


  <!-- ================================================================ -->
  <!-- = List of HADR databases                                         = -->
  <!-- ================================================================ -->
  <HADRDBSet>
    <HADRDB databaseName="HADRDB"
            localInstance="db2inst1"
            remoteInstance="db2inst1"
            localHost="hasys01"
            remoteHost="hasys02" />
  </HADRDBSet>

</DB2Cluster>
```

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 전제조건

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하기 전에 수행해야 하는 태스크 세트가 있습니다.

일반

데이터베이스 관리 프로그램 인스턴스 소유자가 db2haicu를 실행할 수 있기 전에, 루트 권한이 있는 사용자가 preprpnode 명령을 실행해야 합니다.

preprpnode는 AIX에 대한 RSCT(Reliable Scalable Cluster Technology) 파일 세트 및 Linux에 대한 RSCT 패키지의 파트입니다. preprpnode는 클러스터 내부 통신을 위한 노드 초기화를 처리합니다. preprpnode 명령은 클러스터 설정의 파트로서 실행됩니다. preprpnode에 대한 자세한 정보는 다음을 참조하십시오.

- preprpnode 명령 (AIX)
- Linux용 RSCT 기술 참조 - preprpnode

RSCT에 대한 자세한 정보는 RSCT Administration Guide - What is RSCT?를 참조하십시오.

db2haicu를 실행하기 전에, 데이터베이스 관리 프로그램 인스턴스 소유자가 다음 태스크를 수행해야 합니다.

- 클러스터에 추가될 모든 머신의 서비스 파일을 동기화하십시오.
- 클러스터 도메인을 작성하는 데 사용될 데이터베이스 관리 프로그램 인스턴스에 대해 db2profile 스크립트를 실행하십시오.
- db2start 명령을 사용하여 데이터베이스 관리 프로그램을 시작하십시오.

DB2 고가용성 재해 복구(HADR)

HADR 기능을 사용 중인 경우 다음 태스크를 수행하십시오.

- 모든 DB2 고가용성 재해 복구(HADR) 데이터베이스가 해당 기본 및 대기 데이터베이스 역할에서 시작되며 모든 HADR 기본-대기 데이터베이스 쌍이 피어 상태에 있는지 확인하십시오.
- 모든 HADR 데이터베이스에 대한 **hadr_peer_window**를 최소한 120초의 값으로 구성하십시오.
- DB2 결함 모니터를 사용하지 마십시오.

파티션된 데이터베이스 환경

고가용성을 위해 구성할 다중 데이터베이스 파티션이 있는 경우 다음 단계를 수행하십시오.

- 클러스터 도메인에 추가될 모든 머신에서 DB2_NUM_FAILOVER_NODES 레지스트리 변수를 구성하십시오.
- (선택적) db2haicu를 실행하기 전에 데이터베이스를 활성화하십시오.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인 작성

처음으로 데이터베이스 관리 프로그램 인스턴스에 대해 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행할 때 db2haicu가 *클러스터 도메인*이라는 사용자 클러스터의 모델을 작성합니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)가 자동으로 발견하는 데이터베이스 경로:

처음 DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 실행할 때 db2haicu는 클러스터 구성에 관련되는 데이터베이스 구성 정보를 찾기 위해 데이터베이스 시스템을 검색합니다.

단일 데이터베이스 파티션 환경

단일 데이터베이스 파티션 환경에서, db2haicu는 자동으로 다음 경로를 발견합니다.

- 인스턴스 홈 디렉토리 경로
- 감사 로그 경로
- 감사 아카이브 로그 경로
- 동기점 관리 프로그램(SPM) 로그 경로
- DB2 진단 로그(db2diag 로그 파일) 경로
- 데이터베이스 관련 경로:
  - 데이터베이스 로그 경로

– 데이터베이스 테이블 스페이스 컨테이너 경로
– 데이터베이스 테이블 스페이스 디렉토리 경로
– 로컬 데이터베이스 디렉토리

다중 데이터베이스 파티션 환경

다중 데이터베이스 파티션 환경에서, db2haicu는 자동으로 다음 경로만 발견합니다.

- 데이터베이스 로그 경로
- 데이터베이스 테이블 스페이스 컨테이너 경로
- 데이터베이스 테이블 스페이스 디렉토리 경로
- 로컬 데이터베이스 디렉토리

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인 유지보수

db2haicu를 사용하여 클러스터 환경의 클러스터 도메인 모델을 수정할 때, 데이터베이스 관리 프로그램은 관련 변경을 데이터베이스 관리 프로그램 인스턴스 및 클러스터 구성으로 전파시킵니다.

시작하기 전에

db2haicu를 사용하여 클러스터 환경을 구성할 수 있기 전에 클러스터 도메인을 작성 및 구성해야 합니다. 자세한 정보는 127 페이지의 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)를 사용하여 클러스터 도메인 작성』을 참조하십시오.

이 태스크에 대한 정보

db2haicu 유지보수 태스크에는 데이터베이스 또는 클러스터 노드 같은 클러스터 요소를 클러스터 도메인에 추가 또는 클러스터 도메인에서 요소 제거가 포함됩니다. db2haicu 유지보수 태스크에는 데이터베이스 관리 프로그램 인스턴스에 대한 장애 복구 규정 같은 클러스터 도메인 요소의 세부사항 수정도 포함됩니다.

프로시저

1. db2haicu를 실행하십시오.

   db2haicu를 유지보수 모드에서 실행할 때 db2haicu는 클러스터 도메인에 대해 수행할 수 있는 조작 목록을 제공합니다.
   - 클러스터 노드(호스트 이름으로 식별되는 머신) 추가 또는 제거
   - 네트워크 인터페이스(네트워크 인터페이스 카드) 추가 또는 제거
   - 데이터베이스 파티션 추가 또는 제거(파티션된 데이터베이스 환경만 해당)
   - DB2 고가용성 재해 복구(HADR) 데이터베이스 추가 또는 제거
   - 고가용성 데이터베이스 추가 또는 제거

- 마운트 지점 추가 또는 제거
- IP 주소 추가 또는 제거
- 중요하지 않은 경로 추가 또는 제거
- 스케줄된 유지보수를 위한 데이터베이스 파티션 및 HADR 데이터베이스 이동
- 현재 인스턴스에 대한 장애 복구 규정 변경
- 클러스터 도메인에 대한 새 quorum 디바이스 작성
- 클러스터 도메인 삭제

2. 수행할 태스크를 선택하고, db2haicu가 표시하는 후속 질문에 응답하십시오.

결과

데이터베이스 관리 프로그램은 클러스터 도메인의 정보를 사용하여 클러스터 관리 프로그램과 조정합니다. db2haicu를 사용하여 데이터베이스 및 클러스터 요소를 구성하면 해당 요소가 DB2 고가용성(HA) 기능이 제공하는 통합 및 자동화 클러스터 구성 및 관리에 포함됩니다. db2haicu를 사용하여 데이터베이스 관리 프로그램 인스턴스 구성을 변경할 때, 데이터베이스 관리 프로그램이 필요한 클러스터 관리 프로그램 구성 변경을 작성하므로 클러스터 관리 프로그램에 대한 후속 호출을 작성할 필요가 없습니다.

다음 단계

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)에는 별도의 진단 로그가 없습니다. 데이터베이스 관리 프로그램 진단 로그, db2diag 로그 파일 및 db2pd 도구를 사용하여 db2haicu 오류를 조사하고 진단할 수 있습니다. 자세한 정보는 『DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 문제점 해결』을 참조하십시오.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 문제점 해결

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu)에는 별도의 진단 로그가 없습니다. 데이터베이스 관리 프로그램 진단 로그, db2diag 로그 파일 및 db2pd 도구를 사용하여 db2haicu 오류를 조사하고 진단할 수 있습니다.

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 전제조건

DB2 고가용성 인스턴스 구성 유틸리티(db2haicu) 사용에 대한 몇 가지 제한사항이 있습니다.

- 130 페이지의 『소프트웨어 및 하드웨어』
- 130 페이지의 『구성 태스크』
- 130 페이지의 『사용 시 참고사항』
- 131 페이지의 『권장사항』

소프트웨어 및 하드웨어

- 현재, IBM Tivoli SA MP(System Automation for Multiplatforms) 버전 2.2, Fixpack 3이 db2haicu가 지원하는 유일한 클러스터 관리 프로그램입니다.
- RSCT(Reliable Scalable Cluster Technology) 버전 2.4.7.3도 필요합니다. RSCT에 대한 자세한 정보는 RSCT Administration Guide - What is RSCT?를 참조하십시오.
- db2haicu는 IP 버전 6을 지원하지 않습니다.

구성 태스크

db2haicu를 사용하여 다음 태스크를 수행할 수 없습니다.

- db2haicu를 사용하여 자동 클라이언트 리라우트를 구성할 수 없습니다.
- Linux, UNIX 및 Windows용 DB2 데이터베이스 버전 9에서 IBM Data Server 버전 9.5로, 또는 버전 9.5에서 나중 버전으로 업그레이드할 때, db2haicu를 사용하여 클러스터 구성을 이주할 수 없습니다. 클러스터 구성을 이주하려면 다음 단계를 수행해야 합니다.
  1. 기존 클러스터 도메인(하나가 있는 경우) 삭제
  2. 데이터베이스 서버 업그레이드
  3. db2haicu를 사용한 새 클러스터 도메인 작성

사용 시 참고사항

클러스터 구성 및 관리 활동을 계획할 때 다음 db2haicu 사용시 참고를 고려하십시오.

- db2haicu가 정상적으로 루트 권한이 필요한 일부 관리 작업을 수행하는 경우에도 db2haicu는 데이터베이스 관리 프로그램 인스턴스 소유자의 권한과 함께 실행됩니다. 루트 사용자가 수행하는 db2haicu 초기화를 통해 db2haicu가 인스턴스 소유자 특권만을 가지고 있어도 필수 구성 변경을 수행할 수 있습니다.
- 새 클러스터 도메인을 작성할 때 db2haicu는 사용자가 새 클러스터 도메인에 대해 지정하는 이름이 유효한지 검증하지 않습니다. 예를 들어 db2haicu는 이름의 길이가 유효하거나 이름이 유효한 문자를 포함하는지 또는 기존 클러스터 도메인과 동일한 이름이 아닌지 확인하지 않습니다.
- db2haicu는 사용자가 지정하는 정보 및 클러스터 관리 프로그램으로 전달되는 정보를 검증하거나 유효성 확인하지 않습니다. db2haicu가 클러스터 오브젝트 이름과 관련된 모든 클러스터 관리 프로그램 제한사항을 인식할 수 없기 때문에, 예를 들어 db2haicu는 유효한 문자나 길이 같은 사항에 대해 유효성 확인하지 않고 텍스트를 클러스터 관리 프로그램으로 전달합니다.
- 오류가 발생하고 새 클러스터 도메인을 작성 및 구성하는 중에 db2haicu가 실패하는 경우 다음 단계를 수행해야 합니다.

1. **-delete** 매개변수를 사용하여 db2haicu를 실행하여 부분적으로 작성된 클러스터 도메인의 자원 그룹을 제거하십시오.
2. db2haicu를 다시 실행하여 새 클러스터 도메인을 다시 작성하십시오.

- **-delete** 매개변수와 함께 db2haicu를 실행할 때, db2haicu는 해당 자원 그룹이 잠겼는지 확인하지 않고 현재 데이터베이스 관리 프로그램 인스턴스와 연관된 자원 그룹을 즉시 삭제합니다.
- DB2 고가용성 재해 복구(HADR) 기본 데이터베이스, 대기 데이터베이스 쌍의 데이터베이스 관리 프로그램 인스턴스와 연관된 자원 그룹을 제거하려면 다음 단계를 수행하십시오.
  1. 첫 번째로 HADR 대기 데이터베이스의 데이터베이스 관리 프로그램 인스턴스에 대해 **-delete** 매개변수와 함께 db2haicu를 실행하십시오.
  2. 또한 HADR 기본 데이터베이스의 데이터베이스 관리 프로그램 인스턴스에 대해 **-delete** 매개변수와 함께 db2haicu를 실행하십시오.
- db2haicu를 사용하여 수행하려는 클러스터 조작이 시간 종료하는 경우 db2haicu는 사용자에게 오류를 리턴하지 않습니다. 클러스터 조작이 시간종료할 때 db2haicu를 호출한 후 진단 로그를 검토하지 않는 경우 또는 후속 클러스터 조작이 실패하지 않는 경우 조작이 시간종료했는지 알 수 없습니다. 또한 해당 후속 실패를 조사하는 동안 원래 클러스터 조작이 시간종료했음을 판별합니다.
- 주어진 데이터베이스 인스턴스에 대한 장애 복구 규정을 활성-수동으로 변경하려는 경우, 해당 구성 조작이 실패하지만 db2haicu가 오류를 리턴하지 않을 한 가지 조건이 있습니다. *활성* 머신이 될 현재 오프라인인 머신을 지정하는 경우 db2haicu는 해당 머신을 활성 머신으로 만들지 않지만 db2haicu가 변경이 실패했음을 표시하는 오류를 리턴하지 않습니다.

권장사항

다음은 db2haicu를 사용할 때 클러스터 및 데이터베이스 관리 프로그램 인스턴스 구성을 위한 권장사항 목록입니다.

- `/etc/fstab`에 항목을 추가하여 클러스터에 대한 새 마운트 지점을 추가할 때, **noauto** 옵션을 사용하여 마운트 포인트가 클러스터의 둘 이상의 머신에서 자동으로 마운트되지 못하게 하십시오. 예를 들어, 다음과 같습니다.

```
dev/vpatha1      /db/svtpdb/NODE0010        ext3  noauto 0 0
```

DB2 클러스터 관리 프로그램 API

The DB2 클러스터 관리 프로그램 API는 데이터베이스 관리 프로그램이 클러스터 관리 프로그램의 구성 변경사항을 통신할 수 있도록 하는 기능 세트를 정의합니다.

지원되는 클러스터 관리 소프트웨어

클러스터 관리 소프트웨어는 클러스터의 한 노드에 있는 실패한 기본 데이터베이스에서 클러스터의 다른 노드에 있는 보조 데이터베이스로 DB2 데이터베이스 조작을 전송할 수 있습니다.

DB2 데이터베이스는 다음 클러스터 관리 소프트웨어를 지원합니다.

- AIX에 대한 HACMP(High Availability Cluster Multi-Processing)

  HACMP/ES에 대한 세부사항 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 사용 가능한 『AIX 및 HACMP/ES용 IBM DB2 Universal Database Enterprise Edition』이라는 백서를 참조하십시오.

- Linux용 Tivoli System Automation.

  Tivoli System Automation에 대한 세부사항 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 사용 가능한 『Linux용 Tivoli System Automation을 사용한 고가용성 DB2 Universal Database』라는 백서를 참조하십시오.

- Windows 운영 체제를 위한 Microsoft® Cluster Server

  Microsoft Cluster Server에 대한 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 사용 가능한 백서 『Microsoft Cluster Server를 사용하여 IBM DB2 Universal Database V8.1 Enterprise Server Edition 구현』을 참조하십시오.

- Solaris 운영 체제를 위한 Sun Cluster 또는 VERITAS Cluster Server.

  Sun Cluster에 대한 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 사용 가능한 『DB2 Universal Database and High Availability on Sun Cluster 3.X』 백서를 참조하십시오. VERITAS Cluster Server에 대한 자세한 정보는 『DB2 UDB and High Availability with VERITAS Cluster Server』라는 제목의 백서를 참조하십시오. 이 백서는 『IBM Support and downloads』 웹 사이트(http://www.ibm.com/support/docview.wss?uid=swg21045033)에서 사용 가능합니다.

- Hewlett-Packard에 대한 Multi-Computer/ServiceGuard

AIX용 HACMP(High Availability Cluster Multi-Processing)

AIX용 HACMP(High Availability Cluster Multi-Processing)는 클러스터 관리 소프트웨어입니다. HACMP 클러스터의 노드는 하트비트 또는 킵얼라이브 패킷이라고 하는 메시지를 교환합니다. 노드가 이 메시지 송신을 중지하면 HACMP는 클러스터에서 다른 노드 사이에 장애 복구를 호출합니다. 그리고 실패한 노드가 수리되면 HACMP는 다시 클러스터로 통합합니다.

두 가지 유형의 이벤트가 있습니다. HACMP 작업에 참여하는 표준 이벤트와, 하드웨어 및 소프트웨어 구성요소에서 매개변수 모니터링과 연관되는 사용자 정의 이벤트입니다.

표준 이벤트 중 하나는 node_down 이벤트입니다. 이 이벤트는 클러스터의 노드가 실패하고 HACMP가 클러스터의 다른 노드 사이에 장애 복구를 초기화한 경우입니다. 장애 복구 프로세스의 일부로 수행해야 할 사항을 계획할 때 HACMP에서 두 가지의 장애 복구 옵션인 긴급(또는 유휴) 대기와 상호 인계를 사용할 수 있습니다.

주: HACMP를 사용할 때, 다음과 같이 DB2 인스턴스가 db2iauto 유틸리티를 사용하여 시동 시 시작되지 않았는지 확인하십시오.

```
db2iauto -off InstName
```

여기서 *InstName*은 인스턴스의 로그인 이름입니다.

클러스터 구성

긴급 대기 구성에서, 인계 노드인 AIX 프로세서 노드는 다른 워크로드를 실행하지 않습니다. *상호 인계* 구성에서는 인계 노드인 AIX 프로세서 노드가 다른 워크로드를 실행합니다.

일반적으로, 파티션된 데이터베이스 환경에서 DB2 데이터베이스는 각 노드에서 데이터베이스 파티션에 대해 가상 인계 모드에서 실행됩니다. 한 가지 예외는 카탈로그 파티션이 긴급 대시 구성의 일부인 시나리오입니다.

계획할 때 주의할 한 가지 사항은 큰 클러스터를 관리하는 방법입니다. 큰 클러스터보다 작은 클러스터를 관리하는 것이 더 쉽지만, 작은 많은 클러스터보다 하나의 큰 클러스터를 관리하는 것이 더 쉽습니다. 계획할 때 클러스터 환경에서 응용프로그램이 사용될 방법을 고려하십시오. 예를 들어 16개의 노드에서 하나의 큰 이종 응용프로그램이 실행 중인 경우, 8개의 2 노드 클러스터보다 단일 클러스터로 구성을 관리하는 것이 더 쉽습니다. 동일한 16개의 노드가 다른 네트워크, 디스크 및 노드 관계를 사용하여 다른 많은 응용프로그램을 포함하는 경우, 노드를 작은 클러스터로 그룹화하는 것이 조 좋습니다. 노드는 한 번에 하나씩 HACMP 클러스터에 통합된다는 점에 유의하십시오. 하나의 큰 클러스터보다 여러 클러스터의 구성을 시작하는 것이 더 빠릅니다. HACMP는 노드 및 해당 백업이 동일한 클러스터에 있는 한 단일 및 복수의 클러스터 모두를 지원합니다.

HACMP 장애 복구에서는 실제 노드에 대한 자원 그룹의 사전 정의된(*연쇄*라고도 함) 지정이 허용됩니다. 장애 복구 프로시저에서는 또한 실제 노드에 대한 자원 그룹의 유동적(*회전*이라고도 함) 지정이 허용됩니다. 각 자원 그룹 내의 IP 주소와 외부 디스크 볼륨 그룹, 또는 파일 시스템, 또는 NFS 파일 시스템, 그리고 응용프로그램 서버(AS)는 응용프로그램 또는 응용프로그램 구성요소를 지정합니다. 응용프로그램 또는 응용프

로그램 구성요소는 장애 복구 및 재통합에 의해 실제 노드 사이에서 HACMP에 의해 조작할 수 있습니다. 장애 복구 및 재통합 동작은 작성되는 자원 그룹의 유형과, 자원 그룹에 위치되는 노드의 수로 지정됩니다.

예를 들어, DB2 데이터베이스 파티션(논리 노드)을 고려해 보십시오. 해당 로그 및 테이블 스페이스 컨테이너가 외부 디스크에 위치되고 다른 노드는 해당 디스크에 링크된 경우, 다른 노드는 이 디스크에 액세스하여 데이터베이스 파티션을 재시작할 수 있습니다(인계 노드에서). HACMP에 의해 자동화되는 조작은 이러한 유형입니다. HACMP는 또한 DB2 인스턴스 기본 사용자 디렉토리에서 사용되는 NFS 파일 시스템을 복구하기 위해서도 사용될 수 있습니다.

파티션된 데이터베이스 환경에서 DB2 데이터베이스에 대한 복구 계획의 일부로 HACMP 문서를 철저히 읽도록 하십시오. 개념, 계획, 설치 및 관리 안내서를 읽은 후 환경에 맞는 복구 아키텍처를 빌드해야 합니다. 알려진 실패 시점을 기초로 복구를 위해 식별한 서브시스템마다, 필요한 HACMP 클러스터와 복구 노드(긴급 대기 또는 상호 인계)를 식별하십시오.

디스크와 어댑터 둘 다를 외부 디스크 구성에서 미러할 것을 강력히 권장합니다. HACMP로 구성된 DB2 실제 노드의 경우, 볼륨 그룹의 노드가 공유 외부 디스크와 다를 수 있는지 확인하기 위해 주의해야 합니다. 상호 인계 구성에서는, 이 배열에 쌍을 이룬 노드가 충돌 없이 서로의 볼륨 그룹에 액세스할 수 있도록 추가 계획이 필요합니다. 파티션된 데이터베이스 환경에서, 이는 모든 컨테이너 이름이 모든 데이터베이스 사이에 고유해야 함을 의미합니다.

고유성을 성취하기 위한 한 가지 방법은 이름의 일부로 데이터베이스 파티션 번호를 포함시키는 것입니다. SMS 또는 DMS 컨테이너 작성 시 컨테이너 문자열 구문에 대한 노드 표현식을 지정할 수 있습니다. 표현식을 지정할 때 노드 번호는 컨테이너 이름의 일부이거나, 추가 인수를 지정할 경우 인수의 결과가 컨테이너 이름의 일부가 될 수 있습니다. 노드 표현식을 표시하려면 인수 ″ $N″(blank]$N)을 사용하십시오. 인수는 컨테이너 문자열의 끝에서 발생해야 하며 다음 형식 중 한가지 형식으로만 사용할 수 있습니다.

표 7. 컨테이너 작성에 필요한 인수. 노드 번호는 5로 가정합니다.

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 모듈러스입니다. 2. 모두 경우에, 연산자는 왼쪽에서 오른쪽으로 평가됩니다.		

다음은 이 특수 인수를 사용하여 컨테이너를 작성하기 위한 방법의 예입니다.

- 2-노드 시스템에서 사용할 컨테이너를 작성합니다.

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

  다음 컨테이너가 사용됩니다.

```
/dev/rcont0   - on Node 0
/dev/rcont1   - on Node 1
```

- 4-노드 시스템에서 사용할 컨테이너를 작성합니다.

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

  다음 컨테이너가 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 사용할 컨테이너를 작성합니다.

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

  다음 컨테이너가 사용됩니다.

```
/TS3/cont0   - on Node 0
/TS3/cont2   - on Node 0
/TS3/cont1   - on Node 1
/TS3/cont3   - on Node 1
```

HACMP에 대한 DB2 데이터베이스 파티션 구성

구성되면, 인스턴스의 각 데이터베이스 파티션은 한 번에 하나의 실제 노드에서 HACMP에 의해 시작됩니다. 네 개의 노드보다 큰 병렬 DB2 구성을 시작하려면 복수의 클러스터를 권장합니다. 64-노드 병렬 DB2 구성에서, 네 개의 16-노드 클러스터보다 32개의 2-노드 HACMP 클러스터를 병렬로 시작하는 것이 더 빠릅니다.

스크립트 파일은 긴급 대기 또는 상호 인계 노드에서 HACMP 장애 복구 또는 복구에 대해 구성할 때 보조하기 위해 DB2 Enterprise Server Edition과 패키징됩니다. 스크립트 파일은 단일 노드의 경우 rc.db2pe.ee이고 복구 노드의 경우 rc.db2pe.eee입니다. 이 파일들은 sqllib/samples/hacmp/es 디렉토리에 위치됩니다. HACMP 클러스터에 있는 각 시스템에서 /usr/bin으로 해당 파일을 복사한 후 이름을 rc.db2pe로 바꾸십시오.

또한 DB2 버퍼 풀 *크기*는 상호 인계 구성에서 장애 복구하는 중에 rc.db2pe에서 사용자 정의할 수 있습니다. (버퍼 풀 *크기*는 두 개의 데이터베이스 파티션이 하나의 실제 노드에서 실행될 때 적절히 자원 할당되도록 구성할 수 있습니다.)

HACMP 이벤트 모니터링 및 사용자 정의 이벤트

제공된 노드에서 프로세스가 중단된 경우 장애 복구 작업을 시작하는 것은 사용자 정의 이벤트의 예입니다. 이벤트는 클러스터 설정의 일부로서 사용자 정의 이벤트로 수동 구성해야 합니다.

고가용성 IBM DB2 데이터베이스 환경 구현 및 설계에 대한 세부사항 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)를 참조하십시오.

IBM Tivoli System Automation for Multiplatforms(Linux 및 AIX)

IBM Tivoli SAM(Tivoli System Automation for Multiplatforms)은 클러스터에서 데이터베이스 시스템 사이의 사용자, 응용프로그램 및 데이터 자동 전환을 용이하게 하는 클러스터 관리 소프트웨어입니다. Tivoli SAM은 IT 자원(예: 프로세스, 파일 시스템 및 IP 주소) 제어를 자동화합니다.

Tivoli SAM은 자원으로 알려진 것의 사용 가능성을 자동으로 관리하기 위한 Framework를 제공합니다. 자원 예는 다음과 같습니다.

- 제어하기 위해 시작, 모니터 및 중지 스크립트를 작성할 수 있는 소프트웨어의 조각
- Tivoli SAM에 액세스 권한이 부여된 네트워크 인터페이스 카드(NIC). 즉, Tivoli SAM은 액세스 권한이 부여된 NIC 사이에 IP 주소를 유동시켜 사용자가 사용하려는 IP 주소의 사용 가능성을 관리합니다.

예를 들어, DB2 인스턴스와 고가용성 재해 복구 기능에는 둘 다 시작, 중지 및 모니터 명령이 있습니다. 따라서 Tivoli SAM 스크립트를 작성하여 자동으로 이와 같은 자원을 관리할 수 있습니다. 밀접하게 관련되는 자원(예: 동시에 동일한 노드에서 집합적으로 실행되는 자원)을 *자원 그룹*이라고 합니다.

DB2 자원

단일 파티션 DB2 환경에서는 단일 DB2 인스턴스가 서버에서 실행됩니다. 이 DB2 인스턴스는 데이터(자체의 실행 가능 이미지와 인스턴스가 소유하는 데이터베이스)에 대한 로컬 액세스 권한을 갖습니다. 이 DB2 인스턴스가 리모트 클라이언트에 액세스 가능하면 사용되지 않는 IP 주소가 이 DB2 인스턴스에 지정되어야 합니다.

DB2 인스턴스 로컬 데이터 및 IP 주소는 모든 Tivoli SAM이 자동화해야 하는 자원으로 간주됩니다. 이 자원은 밀접하게 관련되므로(예: 동시에 동일한 노드에서 집합적으로 실행됨) 자원 그룹이라고 합니다.

전체 자원 그룹은 클러스터의 한 노드에서 수집됩니다. 장애 복구의 경우, 전체 자원 그룹은 다른 노드에서 시작됩니다.

그룹에 있는 자원 사이에는 다음과 같은 종속성이 존재합니다.

- DB2 인스턴스는 로컬 디스크 다음에 시작해야 합니다.
- DB2 인스턴스는 로컬 디스크 이전에 중지해야 합니다.
- HA IP 주소는 인스턴스와 나란히 배치해야 합니다.

디스크 스토리지

DB2 데이터베이스는 로컬 데이터 스토리지에 대해 이 자원을 이용합니다.

- 원시 디스크(예: `/dev/sda1`)
- LVM(Logical Volume Manager)이 관리하는 논리적 볼륨
- 파일 시스템(예: ext3, jfs)

DB2 데이터는 하나 이상의 원시 디스크에 전체적으로, 논리적 볼륨에 전체적으로, 파일 시스템에 전체적으로, 또는 세 가지 모두 혼합하여 저장할 수 있습니다. 실행 파일은 일종의 파일 시스템에 있어야 합니다.

HA IP 주소에 대한 DB2 데이터베이스 요구사항

DB2 데이터베이스에는 IP 주소에 대한 특수한 요구사항이 없습니다. 인스턴스가 가용성이 높은 것으로 간주되도록 하기 위해 가용성이 높은 IP 주소를 정의할 필요가 없습니다. 그러나 보호하는 IP 주소(있는 경우)는 데이터에 대한 클라이언트의 액세스점이므로, 이 주소가 모든 클라이언트에 의해 알려져야 합니다. 실제로, 이 IP 주소는 CATALOG TCPIP NODE 명령에서 클라이언트가 사용하는 IP 주소입니다.

Tivoli SAM 자원 그룹

IBM Tivoli SAM(System Automation for Multiplatforms)은 Linux 기반 클러스터의 프로세스, 응용프로그램, IP 주소 등과 같은 자원을 자동화하여 고가용성을 제공하는 제품입니다. IT 자원(예: IP 주소)을 자동화하려면 자원을 Tivoli SAM에 정의해야 합니다. 또한 이 자원은 모두 최소 하나의 자원 그룹에 포함되어야 합니다. 이 자원이 항상 동일한 머신에 호스트해야 하는 경우, 자원은 모두 동일한 자원 그룹에 위치되어야 합니다.

모든 응용프로그램이 Tivoli SAM으로 관리 및 자동화되도록 하려면 자원으로 정의해야 합니다. 응용프로그램 자원은 보통 일반 자원 클래스 IBM.Application에서 정의됩니다. 이 자원 클래스에는 자원을 정의하는 몇 개의 속성이 있는데 이 속성 중 최소한 다음의 세 속성이 응용프로그램 특정 속성입니다.

- StartCommand
- StopCommand
- MonitorCommand

이 명령은 스크립트나 실행 파일이 될 수 있습니다.

DB2 환경에 대해 Tivoli SAM 설정

Tivoli SAM이 DB2 환경에 대해 작동하도록 설정하는 데 도움이 될 자세한 구성 정보를 보려면 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 ″Tivoli System Automation″을 검색하십시오.

Microsoft Failover Clustering 지원(Windows)

Microsoft 장애 복구 클러스터링은 Windows 운영 체제에서 서버의 클러스터를 지원합니다. 자동으로 서버 또는 응용프로그램 장애를 감지하여 응답하고 서버 워크로드를 밸런싱할 수 있습니다.

소개

Microsoft 장애 복구 클러스터링은 Windows Server 운영 체제의 기능입니다. MSCS는 서버 두 대(DataCenter Server에서는 서버 네 대까지 가능)를 클러스터에 연결하여 데이터와 응용프로그램의 가용성을 높이고 쉽게 관리할 수 있도록 하는 소프트웨어입니다. 뿐만 아니라 장애 복구 클러스터링은 서버나 응용프로그램의 오류를 자동으로 발견하고 복구할 수 있습니다. 또한 서버의 워크로드를 다른 서버로 옮겨 머신 사용의 균형을 유지하고 시스템 작동 중지 시간 없이 계획된 유지보수 작업을 수행하는데 사용할 수 있습니다.

다음 DB2 제품은 다음과 같은 장애 복구 클러스터링을 지원합니다.

- DB2 Workgroup Server Edition
- DB2 Enterprise Server Edition(DB2 ESE)
- DB2 Connect Enterprise Edition(DB2 CEE)

DB2 장애 복구 클러스터링 구성요소

클러스터는 두 개 이상의 노드를 구성한 것이며, 각 노드는 독립된 컴퓨터 시스템입니다. 클러스터는 네트워크 클라이언트에 한 대의 서버로 나타납니다.

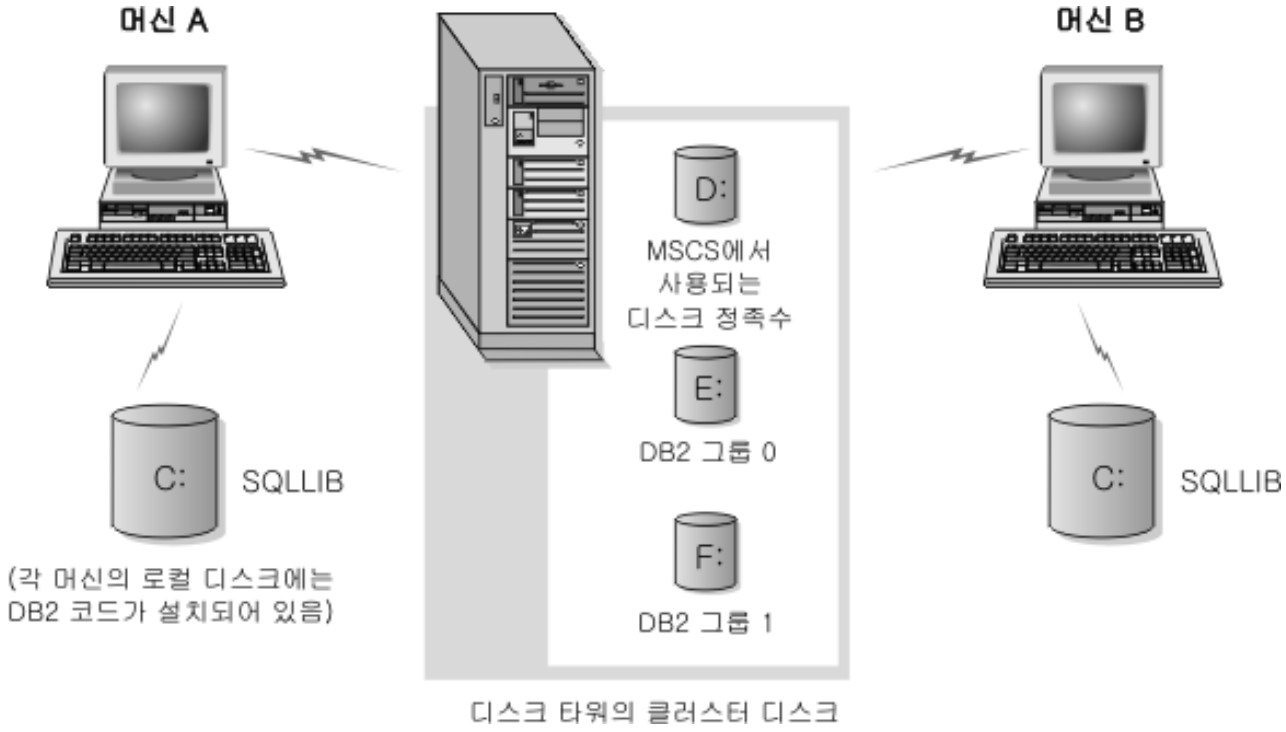


그림 3. 예제 장애 복구 클러스터링 구성

장애 복구 클러스터링 클러스터의 노드는 하나 이상의 공유 스토리지 버스와 하나 이상의 물리적으로 독립된 네트워크를 사용하여 연결됩니다. 서버만 연결하고 클라이언트는 클러스터에 연결하지 않는 네트워크를 *사설* 네트워크라고 합니다. 클라이언트 연결을 지원하는 네트워크를 *공용* 네트워크라고 합니다. 각 노드에는 하나 이상의 로컬 디스크가 있습니다. 각 공유 스토리지 버스는 하나 이상의 디스크에 접속됩니다. 클러스터의 노드는 공유 버스의 각 디스크를 한 번에 한 개만 소유합니다. DB2 소프트웨어는 로컬 디스크에 있습니다. DB2 데이터베이스 파일(예: 테이블, 인덱스, 로그 파일). 장애 복구 클러스터링에서 원시 파티션을 사용하지 않으므로 장애 복구 클러스터링 환경에서 원시 디바이스를 사용하도록 DB2를 구성할 수 없습니다.

DB2 자원

장애 복구 클러스터링 환경에서 자원은 클러스터링 소프트웨어가 관리하는 엔티티입니다. 예를 들어, 디스크나 IP 주소 또는 일반 서비스를 자원으로 관리할 수 있습니다. DB2는 DB2 서버라는 고유 자원 유형을 작성하여 장애 복구 클러스터링에 통합됩니다. 각 DB2 서버 자원이 DB2 인스턴스를 관리하고, 파티션된 데이터베이스 환경에서 실행될 때 각 DB2 서버 자원은 데이터베이스 파티션을 관리합니다. DB2 서버 자원의 이름은 파티션된 데이터베이스 환경에서 DB2 서버 자원 이름이 인스턴스 이름과 데이터베이스 파티션(또는 노드) 번호로 구성되는 경우에도 인스턴스 이름입니다.

온라인 이전 스크립트 및 온라인 이후 스크립트

DB2 자원이 온라인이 되기 이전과 이후에 모두 스크립트를 실행할 수 있습니다. 이러한 스크립트를 각각 온라인 이전 스크립트와 온라인 이후 스크립트라고 합니다. 온라인 이전 및 온라인 이후 스크립트는 DB2 및 시스템 명령을 실행할 수 있는 .BAT 파일입니다.

동일한 머신에서 여러 개의 DB2 인스턴스가 실행될 수 있는 상황에서는 온라인 이전 및 온라인 이후 스크립트를 사용하여 두 인스턴스 모두 시작할 수 있도록 구성을 조정할 수 있습니다. 장애 복구 시에는 온라인 이후 스크립트를 사용하여 수동 데이터베이스 복구를 수행할 수 있습니다. 온라인 이후 스크립트를 사용하여 DB2에 종속된 응용프로그램이나 서비스를 시작할 수도 있습니다.

DB2 그룹

관련 자원이나 종속 자원은 자원 그룹으로 구성됩니다. 클러스터 노드 간에 이동할 때 그룹의 모든 자원이 함께 이동합니다. 예를 들어, 일반적인 DB2 단일 파티션 클러스터 환경에는 다음 자원을 포함하는 DB2 그룹이 있습니다.

1. DB2 자원. DB2 자원은 DB2 인스턴스(또는 노드)를 관리합니다.
2. IP 주소 자원. IP 주소 자원을 사용하여 클라이언트 응용프로그램은 DB2 서버에 연결할 수 있습니다.
3. 네트워크 이름 자원. 네트워크 이름 자원을 사용하여 클라이언트 응용프로그램은 IP 주소를 사용하지 않고 이름을 사용하여 DB2 서버에 연결할 수 있습니다. 네트워크 이름 자원은 IP 주소 자원에 종속됩니다. 네트워크 이름 자원은 선택적 요소입니다. (네트워크 이름 자원을 구성하면 장애 복구 성능에 영향을 줄 수 있습니다.)
4. 하나 이상의 물리적 디스크 자원. 각 물리적 디스크 자원은 클러스터의 공유 디스크를 관리합니다.

주: DB2 자원은 동일한 그룹에 있는 다른 모든 자원에 의존하도록 구성되어 다른 모든 자원이 온라인이어야만 DB2 서버를 시작할 수 있습니다.

장애 복구 구성

두 가지 유형의 구성이 사용 가능합니다.

- 상시 대기
- 상호 인계

파티션된 데이터베이스 환경에서, 클러스터의 모두 구성 유형이 같을 필요는 없습니다. 일부 클러스터는 상시 대기를 사용하도록 설정하고, 다른 클러스터는 상호 인계를 사용하도록 설정할 수 있습니다. 예를 들어, DB2 인스턴스가 5개의 워크스테이션으로 구성되면, 두 머신은 상호 인계 구성을 사용하도록 설정되고, 두 개는 상시 대기 구성을 사

용하도록 설정되며, 한 머신은 장애 복구 지원에 대해 구성되지 않도록 설정할 수 있습니다.

상시 대기 구성

상시 대기 구성에서 장애 복구 클러스터링의 한 머신은 장애 복구 지원을 전담하고, 다른 머신은 데이터베이스 시스템에 참여합니다. 데이터베이스 시스템에 참여하는 머신에 장애가 발생할 경우, 그 머신의 데이터베이스 서버는 장애 복구 머신에서 시작됩니다. 파티션된 데이터베이스 환경의 한 머신에서 다중 논리 노드를 실행했는데 실패했을 경우, 장애 복구 머신에서 논리 노드가 시작됩니다. 그림 4는 상시 대기 구성의 예를 보여 줍니다.

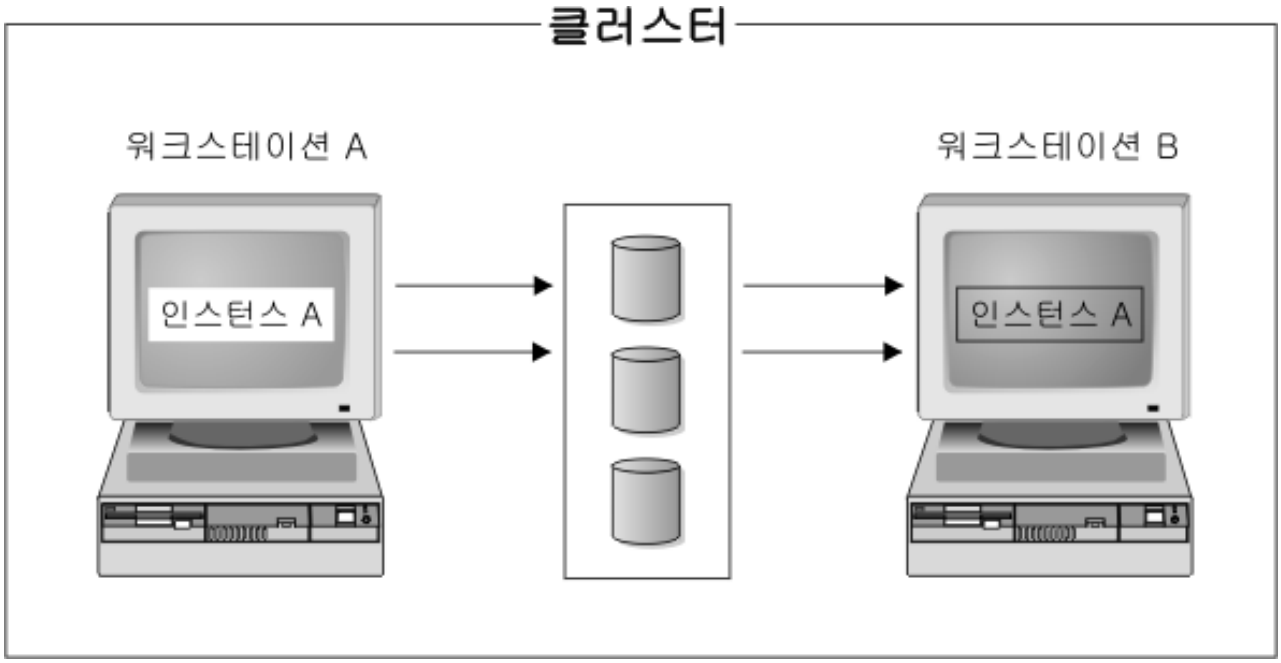


그림 4. 상시 대기 구성

상호 인계 구성

상호 인계 구성에서, 두 워크스테이션은 모두 데이터베이스 시스템에 참여합니다. (즉, 각 머신에는 최소한 하나의 데이터베이스 서버가 실행됩니다.) 장애 복구 클러스터링에서 워크스테이션 중 하나에 장애가 발생할 경우, 장애가 발생한 머신의 데이터베이스 서버는 다른 머신에서 실행되도록 시작됩니다. 상호 인계 구성에서는 한 머신의 데이터베이스 서버에 다른 머신에 있는 데이터베이스 서버와 별개로 장애가 발생할 수 있습니다. 어떠한 데이터베이스 서버라도 특정 시점에서 원하는 머신에서 활성화될 수 있습니다. 142 페이지의 그림 5는 상호 인계 구성의 예를 보여 줍니다.

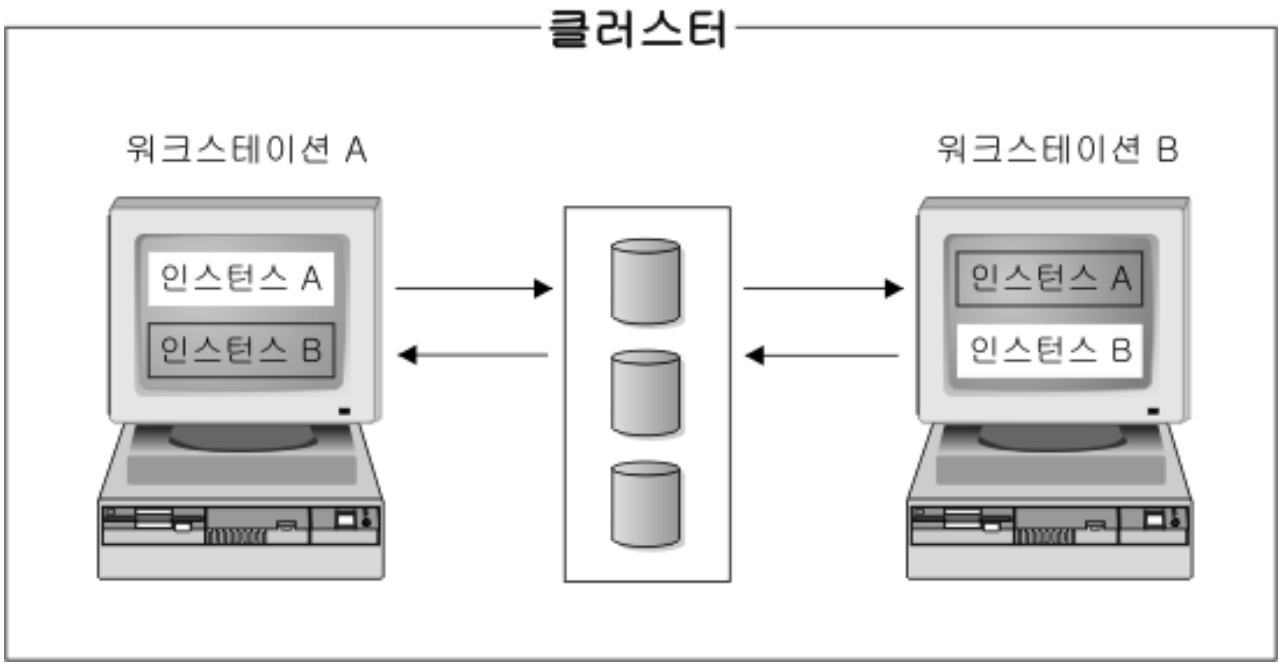


그림 5. 상호 인계 구성

Windows 운영 체제에서 고가용성 IBM DB2 데이터베이스 환경 구현 및 설계에 대한 세부사항 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)를 참조하십시오.

Solaris 운영 체제 클러스터 지원

DB2는 Solaris 운영 체제에 사용 가능한 두 가지의 클러스터 관리 프로그램 Sun Cluster와 VCS(Veritas Cluster Server)를 지원합니다.

Sun Cluster에 대한 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 사용 가능한 『DB2 Universal Database and High Availability on Sun Cluster 3.X』 백서를 참조하십시오.

주: Sun Cluster 3.0 또는 Veritas Cluster Server를 사용할 때, 다음과 같이 db2iauto 유틸리티를 사용하여 시동 시간에 DB2 인스턴스가 시작되지 않는지 확인하십시오.

```
db2iauto -off InstName
```

여기서 *InstName*은 인스턴스의 로그인 이름입니다.

고가용성

데이터 서비스를 호스트하는 컴퓨터 시스템은 다양한 많은 구성요소를 포함하며 각 구성요소에는 연관되는 "MTBF(mean time before failure)"가 있습니다. MTBF는 구성요소가 사용 가능 상태로 유지되는 평균 시간입니다. 양질의 하드 드라이브에 대한 MTBF는 약 100만 시간(대략 114년)입니다. 긴 시간인 것 같지만 200개의 디스크 중 하나는 거의 6개월 내에 실패합니다.

데이터 서비스에 대한 사용 가능성을 높이는 다양한 방법이 있지만 가장 일반적인 방법은 HA 클러스터입니다. 클러스터는 고가용성에 사용될 경우 두 개 이상의 머신, 사설 네트워크 인터페이스 세트, 하나 이상의 공용 네트워크 인터페이스 및 일부 공유 디스크로 구성됩니다. 이 특수 구성을 사용하여 데이터 서비스를 머신 사이에 이동할 수 있습니다. 데이터 서비스를 클러스터의 다른 머신으로 이동해도 계속 해당 데이터에 대한 액세스를 제공할 수 있어야 합니다. 머신 사이에 데이터 서비스를 이동하는 것을 *장애 복구*라고 하며 그림 6에 나와 있습니다.

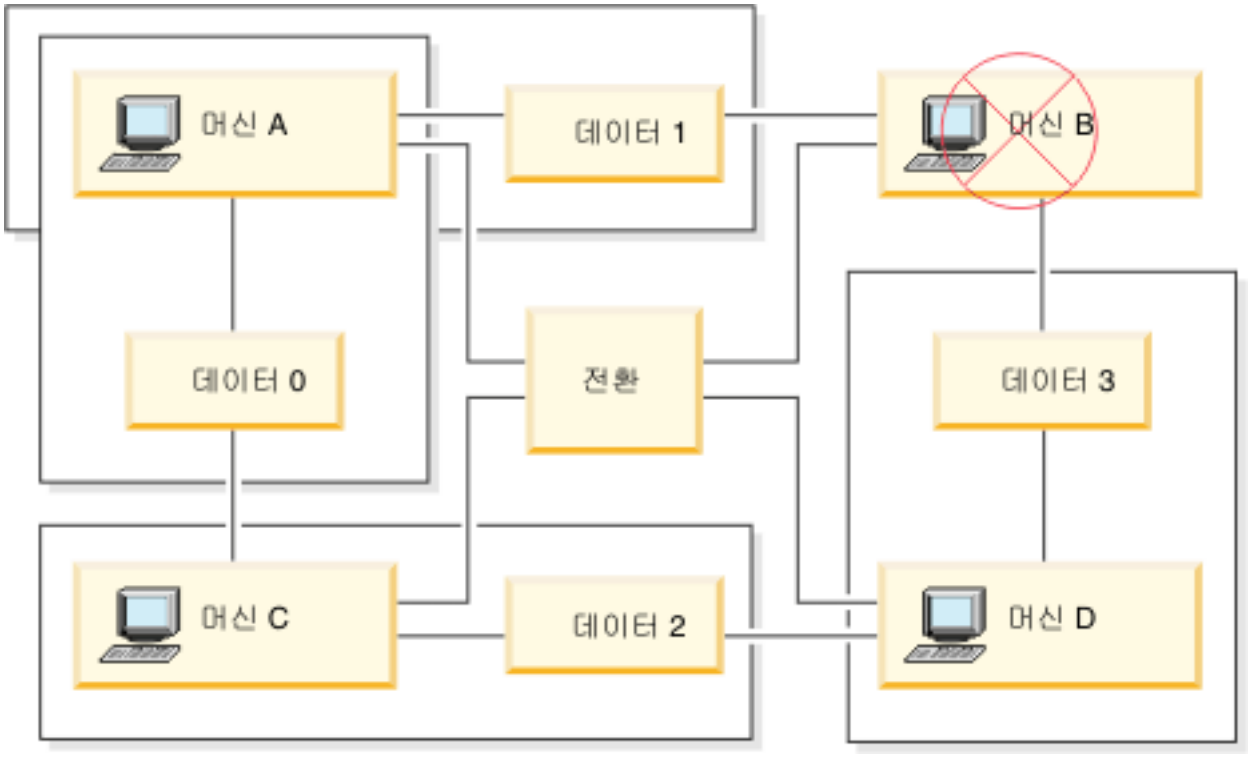


그림 6. 장애 복구. 머신 B가 실패하는 경우 데이터에 계속 액세스할 수 있도록 해당 데이터 서비스가 다른 머신으로 이동됩니다.

사설 네트워크 인터페이스는 클러스터에서 머신 사이에 *하트비트* 메시지와 제어 메시지를 보내기 위해 사용됩니다. 공용 네트워크 인터페이스는 HA 클러스터의 클라이언트와 직접 통신하기 위해 사용됩니다. HA 클러스터의 디스크는 클러스터에 있는 두 개 이상의 머신에 연결되므로, 하나의 머신이 실패하면 다른 머신이 디스크에 대한 액세스 권한을 갖습니다.

HA 클러스터에서 실행 중인 데이터 서비스에는 하나 이상의 논리적 공용 네트워크 인터페이스와 이에 연관되는 디스크 세트가 있습니다. HA 데이터 서비스의 클라이언트는 TCP/IP를 통해 데이터 서비스의 논리적 네트워크 인터페이스에만 연결합니다. 장애 복구가 발생하면, 데이터 서비스는 해당되는 논리적 네트워크 인터페이스 및 디스크 세트와 함께 다른 머신으로 이동됩니다.

HA 클러스터의 이점 중 하나는 데이터 서비스가 지원 스태프의 보조 없이 복구할 수 있고 언제든지 이와 같이 수행할 수 있다는 것입니다. 다른 이점은 중복성입니다. 머신 자체를 포함하여 클러스터에 있는 모든 파트는 중복되어야 합니다. 클러스터는 단일 실패점을 극복할 수 있어야 합니다.

사용 가능성이 높은 데이터 서비스는 본질적으로 아주 다를 수 있지만, 일부 공통적인 요구사항을 가지고 있습니다. 사용 가능성이 높은 데이터 서비스의 클라이언트는 데이터 서비스의 네트워크 주소 및 호스트 이름이 동일하게 유지될 것으로 예상하고, 데이터 서비스가 설정된 머신에 관계없이 동일한 방법으로 요청을 작성할 수 있을 것으로 예상합니다.

사용 가능성이 높은 웹 서버에 액세스 중인 웹 브라우저를 고려해 보십시오. 요청은 호스트 이름과 웹 서버에서의 파일 경로를 포함하는 URL(Uniform Resource Locator)로 발행됩니다. 브라우저는 호스트 이름 및 경로가 웹 서버 장애 복구 후에도 동일하게 유지될 것으로 예상합니다. 브라우저가 웹 서버에서 파일을 다운로드하는 중인데 서버가 장애 복구되는 경우 브라우저는 요청을 다시 발행해야 합니다.

데이터 서비스의 사용 가능성은 데이터 서비스를 사용자가 사용할 수 있는 시간으로 측정됩니다. 사용 가능성의 가장 일반적인 측정 단위는 "가동 시간"의 백분율입니다. 이는 종종 "9"의 개수로 언급됩니다.

```
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 설계 및 테스트할 때:

1. 클러스터 관리자가 시스템과 장애 복구 발생 시 발생해야 하는 상황에 익숙한지 확인하십시오.
2. 클러스터의 각 파트가 중복되어 있고 실패하는 경우 신속하게 교체할 수 있는지 확인하십시오.
3. 테스트 시스템이 제어 환경에서 강제로 실패하도록 하고 매번 올바르게 장애 복구하는지 확인하십시오.
4. 장애 복구마다 이유를 추적하십시오. 자주 발생하지 않아야 하지만 클러스터를 불안정한 상태로 만드는 문제를 처리하는 것은 중요합니다. 예를 들어, 클러스터의 한 부분에 의해 장애 복구가 한 달에 5번 발생한 경우 이유를 찾아서 수정하십시오.
5. 장애 복구가 발생할 때 클러스터의 지원 스태프에 통지되는지 확인하십시오.
6. 클러스터를 오버로드하지 마십시오. 나머지 시스템은 장애 복구 후 승인할 수 있는 레벨에서 계속 워크로드를 처리할 수 있는지 확인하십시오.
7. 문제점 발생 이전에 교체될 수 있도록 실패하기 쉬운 구성요소(예: 디스크)를 점검하십시오.

오류 허용

데이터 서비스의 사용 가능성을 증가시키는 또 다른 방법은 오류 허용입니다. *오류 허용* 머신에는 해당되는 모든 부품이 중복되어 내장되어 있으므로 CPU 및 메모리와 같은 어떤 부품의 단일 장애도 견딜 수 있어야 합니다. 오류 허용 머신은 틈새 시장에서 가장 자주 사용되며, 보통 구현에 많은 비용이 소비됩니다. 다른 지리적 위치에 머신에 있는 HA에는 해당 위치 서브세트에만 영향을 주는 재해로부터 복구할 수 있는 장점이 추가됩니다.

HA 클러스터는 확장 가능하고, 사용하기 쉬우며 상대적으로 구현 비용이 저렴하므로 사용 가능성을 증가시키기 위한 가장 일반적인 솔루션입니다.

Sun Cluster 3.0(이상) 지원:

DB2 데이터베이스 솔루션을 Solaris 운영 체제 클러스터에서 실행할 계획이면, Sun Cluster 3.0을 사용하여 클러스터를 관리할 수 있습니다. 고가용성 에이전트는 DB2 데이터베이스와 Sun Cluster 3.0 사이에서 중개자로 작동합니다.

이 주제에 있는 Sun Cluster 3.0 지원에 대한 문장은 Sun Cluster 3.0 이상 버전에도 적용됩니다.

그림 7. DB2 데이터베이스, Sun Cluster 3.0 및 고가용성. DB2 데이터베이스, Sun Cluster 3.0 및 고가용성 에이전트 사이의 관계.

장애 복구

Sun Cluster 3.0은 응용프로그램 장애 복구를 사용 가능하도록 설정하여 고가용성을 제공합니다. 각각의 노드는 정기적으로 모니터되고 클러스터 소프트웨어는 자동으로 실패한 기본 노드에서 지정된 보조 노드로 클러스터 인식 응용프로그램을 재조정합니다. 장애 복구가 발생할 경우 클라이언트는 서비스에서 잠시 인터럽트가 발생하고 서버에 다시 연결해야 할 수도 있습니다. 그러나 응용프로그램과 데이터에 액세스 중인 실제 서버를 인식하지 못합니다. 기본 노드가 실패할 때 클러스터에 있는 다른 노드가 자동으로 워크로드를 호스트하도록 하여, Sun Cluster 3.0은 중단시간을 상당히 줄이고 생산성을 높입니다.

멀티호스트 디스크

Sun Cluster 3.0에는 멀티호스트 디스크 스토리지가 필요합니다. 이는 디스크를 한 번에 여러 개의 노드에 연결할 수 있음을 의미합니다. Sun Cluster 3.0 환경에서, 멀티호스트 스토리지는 디스크 디바이스의 사용 가능성을 높입니다. 멀티호스트 스토리지에 있는 디스크 디바이스는 단일 노드 실패를 허용할 수 있습니다. 대체 서버 노드를 통한 데이터로의 실제 경로는 여전히 있기 때문입니다. 멀티호스트 디스크는 기본 노드를 통해 전역으로 액세스할 수 있습니다. 클라이언트 요청이 하나의 노드를 통해 데이터에 액세스 중인데 해당 노드가 실패하는 경우 요청은 동일 디스크와 직접 연결되어 있는 다른 노드로 전환됩니다. 볼륨 관리 프로그램은 멀티호스트 디스크의 데이터 중복성을 위해 미러된 구성이나 RAID 5 구성에 대해 제공합니다. 현재, Sun Cluster 3.0은 볼륨 관리 프로그램으로 Solstice DiskSuite 및 VERITAS Volume Manager를 지원합니다. 멀티호스트 디스크를 디스크 미러링 및 스트라이핑과 결합하면 노드 오류 및 개별적 디스크 오류 둘 다로부터 보호됩니다.

전역 디바이스

전역 디바이스는 디바이스의 물리적 위치에 관계없이 임의 노드로부터 클러스터에 있는 임의 디바이스로의 클러스터 전반의 고가용성 액세스를 제공하기 위해 사용됩니다. 모든 디스크는 디바이스 ID(DID)가 지정된 전역 이름 스페이스에 포함되고 전역 디바이스로 구성됩니다. 따라서 디스크 자체는 모든 클러스터 노드에서 볼 수 있습니다.

파일 시스템/전역 파일 시스템

클러스터 또는 전역 파일 시스템은 커널(하나의 노드에 있는)과 기본적인 파일 시스템 볼륨 관리 프로그램(하나 이상의 디스크에 실제로 연결되어 있는 노드에 있는) 사이의 프록시입니다. 클러스터 파일 시스템은 하나 이상의 노드에 실제로 연결되어 있는 전역 디바이스에 종속됩니다. 이 파일 시스템은 기본적인 파일 시스템 및 볼륨 관리 프로그램과는 관계가 없습니다. 현재 클러스터 파일 시스템은 Solstice DiskSuite 또는 VERITAS Volume Manager를 사용하여 UFS에서 빌드될 수 있습니다. 데이터는 디스크의 파일 시스템이 클러스터 파일 시스템으로 전역으로 마운트된 경우에만 모든 노드에 사용 가능하게 됩니다.

디바이스 그룹

모든 멀티호스트 디스크는 Sun Cluster Framework에서 제어해야 합니다. Solstice DiskSuite 또는 VERITAS Volume Manager에 의해 관리되는 디스크 그룹은 먼저 멀티호스트 디스크에서 작성됩니다. 그런 다음 Sun Cluster 디스크 디바이스 그룹으로 등록됩니다. 디스크 디바이스 그룹은 전역 디바이스의 유형입니다. 멀티호스트 디바이스 그룹은 사용 가능성이 높습니다. 디스크는 현재 디바이스 그룹을 마스터 중인 노드가 실패하는 경우 대체 경로를 통해 액세스할 수 있습니다. 디바이스 그룹을 마스터 중인 노드의 실패는 복구 및 일관성 검사 수행에 필요한 시간을 제외하고 디바이스 그룹에 대한 액세스에 영향을 주지 않습니다. 이 시간 동안, 시스템이 디바이스 그룹을 사용 가능 상태로 만들 때까지 모든 요청은 차단됩니다(응용프로그램에 투명하게).

RGM(Resource Group Manager)

RGM은 고가용성에 대한 메커니즘을 제공하고 각 클러스터 노드에서 디먼으로 실행됩니다. 미리 구성된 규정에 따라 선택된 노드에서 자원을 자동으로 시작하고 중지합니다. RGM을 사용하면 자원이 노드 실패 이벤트 발생 시 사용 가능이 높아지거나 영향을 받는 노드의 자원을 중지하고 다른 자원을 시작하여 재부트할 수 있습니다. RGM은 또한 자원 실패를 발견하고 실패한 자원을 다른 노드로 재배치할 수 있는 자원 특정 모니터를 자동으로 시작 및 중지합니다.

데이터 서비스

데이터 서비스라는 용어는 단일 서버가 아니라 클러스터에서 실행하도록 구성된 써드 파티 응용프로그램을 설정하기 위해 사용됩니다. 데이터 서비스에는 응용프로그램을 시작, 중지 및 모니터하는 Sun Cluster 3.0 소프트웨어와 응용프로그램 소프트웨어가 포함됩니다. Sun Cluster 3.0은 클러스터 내에서 응용프로그램을 제어하고 모니터하기 위해 사용되는 데이터 서비스 메소드를 제공합니다. 이 메소드는 클러스터 노드에서 응용프로그램을 시작, 중지 및 모니터하기 위해 메소드를 사용하는 RGM(Resource Group Manager) 제어 하에 실행됩니다. 이 메소드는 클러스터 Framework 소프트웨어 및 멀티호스트 디스크와 함께 응용프로그램이 고가용성 데이터 서비스가 될 수 있도록 합니다. 고가용성 데이터 서비스로서, 실패가 노드, 인터페이스 구성요소 또는 응용프로그램 자체에서 발생하는지 여부에 관계없이 클러스터 내에서 단일 실패 후에 중요한 응용프로그램 인터럽트를 방지할 수 있습니다. RGM은 또한 네트워크 자원(논리 호스트 이름 및 공유 주소)과 응용프로그램 인스턴스를 비롯하여 클러스터에 있는 자원을 관리합니다.

자원 유형, 자원 및 자원 그룹

자원 유형은 다음으로 구성됩니다.

1. 클러스터에서 실행할 소프트웨어 응용프로그램
2. 응용프로그램을 클러스터 자원으로 관리하기 위해 RGM에 의해 콜백 메소드로 사용되는 제어 프로그램
3. 클러스터의 정적 구성 파트를 형성하는 등록 정보 세트

RGM은 자원 유형 등록 정보를 사용하여 특정 유형의 자원을 관리합니다.

자원은 해당 자원 유형의 등록 정보 및 값을 상속합니다. 이 자원은 클러스터에서 실행 중인 기본적인 응용프로그램의 인스턴스입니다. 각 인스턴스에는 클러스터 내에서 고유한 이름이 필요합니다. 각 자원은 자원 그룹에서 구성해야 합니다. RGM은 그룹에 있는 모든 자원을 함께 동일한 노드에서 온라인 및 오프라인으로 가져옵니다. RGM이 자원 그룹을 온라인 또는 오프라인으로 가져올 때 그룹에 있는 개별 자원에 대해 콜백 메소드를 호출합니다.

자원 그룹이 현재 온라인에 있는 노드를 기본 노드 또는 기본이라고 합니다. 자원 그룹은 해당되는 기본 노드 각각에 의해 마스터됩니다. 각 자원 그룹에는 자원 그룹의 가능한 모든 기본 노드 또는 마스터 노드를 식별하기 위해 클러스터 관리자가 설정하는 연관된 Nodelist 등록 정보가 있습니다.

Sun Cluster 플랫폼에서의 고가용성 IBM DB2 데이터베이스 환경 구현 및 설계에 대한 세부사항 정보는 IBM Software Library 웹 사이트(http://www.ibm.com/software/sw-library/)에서 사용 가능한 ″DB2 and High Availability on Sun Cluster 3.X″ 백서를 참조하십시오.

VERITAS Cluster Server 지원:

DB2 데이터베이스 솔루션을 Solaris 운영 체제 클러스터에서 실행할 계획이면, VERITAS Cluster Server를 사용하여 클러스터를 관리할 수 있습니다. VERITAS Cluster Server는 이종 환경에서 광범위한 응용프로그램을 관리할 수 있으며, SAN(Storage Area Network) 및 일반적인 클라이언트/서버 환경에서 32개까지의 노드 클러스터를 지원합니다.

하드웨어 요구사항

다음은 VERITAS Cluster Server가 현재 지원하는 하드웨어의 목록입니다.

- 서버 노드의 경우:
  - 최소 128MB RAM이 있고 Solaris 2.6 이상을 실행 중인 Sun Microsystems의 SPARC/Solaris 서버
- 디스크 스토리지의 경우:
  - EMC Symmetrix, IBM Enterprise Storage Server®, HDS 7700 및 9xxx, Sun T3, Sun A5000, Sun A1000, Sun D1000, 그리고 by VCS 2.0 이상에서 지원되는 기타 디스크 스토리지. VERITAS 영업대표가 지원되는 디스크 서브시스템을 확인하거나 사용자가 VCS 문서를 참조할 수 있습니다.
  - 일반적인 환경에서는 DB2 실행 파일에 대한 미러된 개인용 디스크(각각의 클러스터 노드에 있는)와 DB2 데이터에 대한 노드 사이의 공유 디스크가 필요합니다.
- 네트워크 상호 연결의 경우:
  - 공용 네트워크 연결의 경우 IP 기반 주소 지정을 지원하는 모든 네트워크 연결.
  - 하트비트 연결(클러스터에 내부적)의 경우 중복 하트비트 연결이 필요합니다. 이 요구사항은 서버마다 두 개의 추가적인 이더넷 제어기를 사용하거나 클러스터마다 하나의 공유 GABdisk를 사용하여 충족할 수 있습니다.

소프트웨어 요구사항

다음 VERITAS 소프트웨어 구성요소는 규정된 구성입니다.

- VERITAS Volume Manager 3.2 이상, VERITAS File System 3.4 이상, VERITAS Cluster Server 2.0 이상.
- Solaris 1.0 이상을 위한 DB2의 DB 개정판.

VERITAS Cluster Server에는 볼륨 관리 프로그램이 필요하지 않지만, 설치, 구성 및 관리를 쉽게 하려면 VERITAS Volume Manager를 사용할 것을 강력히 권장합니다.

장애 복구

VERITAS Cluster Server는 응용프로그램 장애 복구를 사용 가능하도록 하여 DB2 데이터베이스와 같은 응용프로그램 서비스의 사용 가능성을 관리하는 사용 가능성 클러스터링 솔루션입니다. 각 개별적 클러스터 노드 및 연관된 소프트웨어 서비스의 상태를 정기적으로 모니터합니다. 응용프로그램 서비스(이 경우, DB2 데이터베이스 서비스)를 손상시키는 실패가 발생할 때, VERITAS Cluster Server나 VCS HA-DB2 Agent, 또는 둘 다는 실패를 발견하고 자동으로 서비스를 리스토어할 단계를 수행합니다. 서비스를 리스토어하기 위해 수행하는 단계에는 동일 노드에서 DB2 데이터베이스 시스템을 재시작하거나, 클러스터에 있는 다른 노드로 DB2 데이터베이스 시스템을 이동하고 그 노드에서 재시작하는 단계가 포함됩니다. 응용프로그램을 새 노드로 이주해야 하는 경우, VERITAS Cluster Server는 응용프로그램과 연관되는 모든 것(즉, 네트워크 IP 주소, 기본적인 스토리지의 소유권)을 새 노드로 이동하여, 서비스가 실제로 다른 노드에서 실행 중임을 사용자들이 인식하지 못하도록 합니다. 사용자들은 동일한 IP 주소를 사용하여 계속 서비스에 액세스하지만 이 주소는 이제 다른 클러스터 노드를 지시합니다.

VERITAS Cluster Server에서 장애 복구가 발생할 때, 사용자는 서비스에서 손상을 볼 수도, 보지 못할 수도 있습니다. 이는 클라이언트가 응용프로그램 서비스에 대해 가지고 있는 연결 유형(stateful 또는 stateless)을 근거로 합니다. Stateful 연결을 가지고 있는 응용프로그램 환경(예: DB2 데이터베이스)에서, 사용자는 서비스에서 잠깐의 인터럽트를 볼 수 있으므로 장애 복구 완료 후에 다시 연결해야 할 수도 있습니다. Stateless 연결을 가지고 있는 응용프로그램 환경(예: NFS)에서는, 사용자가 서비스에서 잠깐의 대기 시간을 볼 수도 있지만 일반적으로 손상을 보지는 못하므로 다시 로그온하지 않아도 됩니다.

클러스터 노드 사이에 자동으로 이주될 수 있는 서비스로 응용프로그램을 지원하면, VERITAS Cluster Server는 계획하지 않은 중단 시간을 감소시킬 뿐만 아니라 계획된 중단 시간(유지보수 및 업그레이드를 위한)과 연관되는 정지 지속기간도 줄일 수 있습니다. 장애 복구는 수동으로 시작할 수도 있습니다. 하드웨어 또는 운영 체제 업그레이드를 특정 노드에서 수행해야 하는 경우, DB2 데이터베이스 시스템은 클러스터의 다른 노드로 이주될 수 있고, 업그레이드를 수행할 수 있으므로 DB2 데이터베이스 시스템을 원래의 노드로 다시 이주할 수 있습니다.

이러한 유형의 클러스터링 환경에서 사용하도록 권장되는 응용프로그램은 손상을 견뎌야 합니다. 손상 인내 응용프로그램은 커미트된 데이터의 무결성을 계속 유지하면서 예기치 않은 손상에서 복구할 수 있습니다. 손상 인내 응용프로그램은 간혹 *클러스터 친숙* 응용프로그램이라고 합니다. DB2 데이터베이스 시스템은 손상 인내 응용프로그램입니다.

VERITAS CFS, CVM 및 VCS 솔루션을 사용하여 장애 복구를 수행하는 데 소요되는 시간을 줄이기 위한 방법에 대한 정보는 IBM Software Library 웹 사이트 (http://www.ibm.com/software/sw-library/)에서 사용 가능한 『DB2 Universal Database Version 8 and VERITAS Database Edition/HA for DB2』 백서를 참조하십시오.

공유 스토리지

VCS HA-DB2 Agent와 함께 사용하는 경우, Veritas Cluster Server에는 공유 스토리지가 필요합니다. 공유 스토리지는 클러스터에 있는 여러 개의 노드와 실제로 연결되어 있는 스토리지입니다. 공유 스토리지에 있는 디스크 디바이스는 노드 실패를 허용할 수 있습니다. 디스크 디바이스에 대한 실제 경로가 하나 이상의 대체 클러스터 노드를 통해 계속 존재하기 때문입니다.

VERITAS Cluster Server 제어를 통해, 클러스터 노드는 ″디스크 그룹″이라고 하는 논리적 구성을 통해 공유 스토리지에 액세스할 수 있습니다. 디스크 그룹은 소유권이 클러스터의 노드 사이에 자동으로 이주될 수 있는, 논리적으로 정의된 스토리지 디바이스의 콜렉션을 나타냅니다. 디스크 그룹은 지정된 시간에 단일 노드로만 임포트할 수 있습니다. 예를 들어, 디스크 그룹 A가 노드 1에 임포트되는 데 노드 1이 실패하는 경우, 디스크 그룹 A는 실패한 노드에서 익스포트하여 클러스터의 새 노드로 임포트될 수 있습니다. VERITAS Cluster Server는 단일 클러스터에서 여러 개의 디스크 그룹을 동시에 제어할 수 있습니다.

디스크 그룹 정의를 허용하는 것 외에도, 공유 스토리지에서 미러링 또는 RAID 5를 사용하는 중복 데이터 구성을 위해 볼륨 관리 프로그램을 제공할 수 있습니다. VERITAS Cluster Server는 논리적 볼륨 관리 프로그램으로 VERITAS Volume Manager 및 Solstice DiskSuite를 지원합니다. 공유 스토리지를 디스크 미러링 및 스트라이핑과 결합하면 노드 오류 및 개별적 디스크 또는 제어기 실패 둘 다로부터 보호할 수 있습니다.

VERITAS Cluster Server GAB(Global Atomic Broadcast) 및 LLT(Low Latency Transport)

노드간 통신 메커니즘은 노드가 하드웨어 및 소프트웨어 관련 정보를 교환하여 클러스터 멤버쉽 트랙을 유지하고 모든 클러스터 노드 사이에 이 정보를 동기화하여 보존할 수 있도록, 클러스터 구성에서 필요합니다. LLT(Low Latency Transport) 사이에 실행 중인 GAB(Global Atomic Broadcast) 기능은 VERITAS Cluster Server가 이를

수행하기 위해 사용하는 지연성이 낮은 고속 메커니즘을 제공합니다. GAB는 각 클러스터 노드에서 커널 모듈로 로드되고 모든 노드가 동시에 상태 갱신 정보를 가져오도록 하는 자동 브로드캐스트 메커니즘을 제공합니다.

커널 대 커널 통신 성능을 지지하여, LLT는 클러스터 노드 사이에 교환하고 동기화해야 하는 모든 정보에 대해 지연성이 낮은 고속 전송을 제공합니다. GAB는 LLT의 맨 위에서 실행됩니다. VERITAS Cluster Server는 하트비트 메커니즘으로 IP를 사용하지 않지만 두 가지의 신뢰할 수 있는 다른 추가 옵션을 제공합니다. 하트비트 메커니즘으로 작동하도록 구성될 수 있는 LLT가 있는 GAB나, 디스크 기반 하트비트로 구성될 수 있는 GABdisk입니다. 하트비트는 중복 연결을 통해 실행해야 합니다. 이 연결은 클러스터 노드 사이에 있는 두 개의 개인용 이더넷 연결이거나, 하나는 개인용 이더넷 연결이고 다른 하나는 GABdisk 연결이 될 수 있습니다. 두 GABdisk를 사용하는 것은 지원되는 구성이 아닙니다. 노드 사이에 클러스터 상태를 교환하려면 개인용 이더넷 연결이 필요하기 때문입니다.

GAB 또는 LLT에 대한 자세한 정보나, VERITAS Cluster Server 구성에서 이를 구성하는 방법에 대한 자세한 정보는 Solaris용 VERITAS Cluster Server 2.0 User's Guide를 참조하십시오.

번들로 묶인 에이전트와 엔터프라이즈 에이전트

에이전트는 특정 자원 또는 응용프로그램의 사용 가능성을 관리하기 위해 설계되는 프로그램입니다. 에이전트가 시작될 때, 에이전트는 VCS로부터 필요한 구성 정보를 확보한 후 자원 또는 응용프로그램을 정기적으로 모니터하고 VCS를 해당 상태로 갱신합니다. 일반적으로, 에이전트는 자원을 온라인으로 가져오거나, 자원을 오프라인으로 가져오거나, 자원을 모니터하기 위해 사용되며 네 가지 유형의 서비스(시작, 중지, 모니터 및 정리)를 제공합니다. 시작 및 중지는 자원을 온라인 또는 오프라인으로 가져오기 위해 사용되고, 모니터는 특정 자원 또는 응용프로그램을 해당 상태에 대해 테스트하기 위해 사용되며, 정리는 복구 프로세스에서 사용됩니다.

번들로 묶인 다양한 에이전트가 VERITAS Cluster Server의 일부로 포함되며 VERITAS Cluster Server가 설치될 때 설치됩니다. 번들로 묶인 에이전트는 보통 클러스터 구성에서 발견되는 사전 정의된 자원 유형(즉 IP, 마운트, 프로세스 및 공유)을 관리하는 VCS 프로세스로, 클러스터 설치 및 구성을 상당히 단순화하는 데 도움을 제공합니다. VERITAS Cluster Server에는 20개가 넘는 번들 에이전트가 있습니다.

엔터프라이즈 에이전트는 DB2 데이터베이스 응용프로그램과 같이 특정 응용프로그램에 초점을 맞추는 경향이 있습니다. VCS HA-DB2 Agent는 Enterprise Agent로 간주될 수 있으며, VCS Agent Framework를 통해 VCS와 인터페이스합니다.

VCS 자원, 유형 유형 및 자원 그룹

자원 유형은 모니터할 VCS 클러스터 내에서 자원을 정의하기 위해 사용되는 오브젝트 정의입니다. 자원 유형에는 자원 유형 이름과, 해당 자원과 연관되는 등록 정보(고가용성 관점에서 두드러진) 세트가 포함됩니다. 자원은 해당 자원 유형의 등록 정보 및 값을 상속하므로, 자원 이름은 클러스터 전반을 기준으로 고유해야 합니다.

두 가지 유형의 자원이 있습니다. 지속적 자원과 표준(비지속적) 자원입니다. 지속적 자원은 모니터하지만 VCS에 의해 온라인이나 오프라인으로 가져오지 않는 자원입니다 (예: 네트워크 인터페이스 제어기(NIC)). 표준 자원은 온라인 및 오프라인 상태가 VCS에 의해 제어되는 자원입니다.

모니터하는 가장 낮은 레벨의 오브젝트가 자원이며, 다양한 자원 유형이 있습니다 (즉, 공유, 마운트). 각 자원은 자원 그룹으로 구성되어야 하며, VCS는 특정 자원 그룹에 있는 모든 자원을 한꺼번에 온라인 및 오프라인으로 가져옵니다. 자원 그룹을 온라인 또는 오프라인으로 가져오기 위해, VCS는 그룹에 있는 각 자원에 대해 시작 또는 중지 메소드를 호출합니다. 두 가지 유형의 자원 그룹인 장애 복구와 병렬이 있습니다. 사용 가능성이 높은 DB2 데이터베이스 구성은 파티션된 데이터베이스 환경 여부에 관계없이 장애 복구 자원 그룹을 사용합니다.

″기본″ 또는 ″마스터″ 노드는 자원을 호스트할 수 있는 노드입니다. systemlist라고 하는 자원 그룹 속성을 사용하여 특정 자원 그룹의 기본이 될 수 있는 클러스터 내 노드를 지정합니다. 2-노드 클러스터에서, 보통 두 노드 모두 systemlist에 포함되지만, 몇 개의 고가용성 응용프로그램을 호스트 중일 수 있는 더 큰 다중 노드 클러스터에서는 특정 응용프로그램 서비스(가장 낮은 레벨에서 해당 자원에 의해 정의된)가 특정 노드로 장애 복구될 수 없도록 하기 위한 요구사항이 있을 수 있습니다.

종속성은 자원 그룹 사이에 정의될 수 있고, VERITAS Cluster Server는 다양한 자원 실패 영향을 평가하고 복구를 관리할 때 이 자원 그룹 종속성 계층 구조에 의존합니다. 예를 들어, 자원 그룹 ClientApp1은 자원 그룹 DB2가 이미 성공적으로 시작되지 않으면 온라인으로 가져올 수 없으므로, 자원 그룹 ClientApp1은 자원 그룹 DB2에 종속된 것으로 간주됩니다.

VERITAS Cluster Server에서의 고가용성 IBM DB2 데이터베이스 환경 구현 및 설계에 대한 세부사항 정보는 ″DB2 UDB and High Availability with VERITAS Cluster Server″(http://www.ibm.com/support/docview.wss?rs=71&uid;=swg21045033) technote를 참조하십시오.

파티션된 데이터베이스 환경에서 클럭 동기화

데이터베이스 파티션 서버 사이에 상대적으로 동기화된 시스템 클럭이 유지되도록 하여 원활한 데이터베이스 조작 및 무제한 포워드 복구성이 가능하도록 해야 합니다. 데이터베이스 파티션 서버 사이의 시간 차이와, 트랜잭션에 대해 가능한 작동 및 통신 지연은 *max_time_diff*(노드 사이의 최대 시간 차이) 데이터베이스 관리 프로그램 구성 매개변수에 지정된 값보다 작아야 합니다.

로그 레코드 시간소인에 파티션된 데이터베이스 환경에서 트랜잭션의 시퀀스를 반영하기 위해 DB2는 각 머신의 SQLOGCTL.LFH 파일에 저장된 시스템 클럭 시간소인을 로그 레코드 시간소인의 기준으로 사용합니다. 그러나 시스템 클럭이 우선 설정된 경우에는 로그 클럭이 시스템 클럭에 따라 자동으로 설정됩니다. 시스템 클럭을 다시 설정할 수 있어도, 로그 클럭은 다시 설정할 수 없으므로 시스템 클럭이 이 시간과 일치할 때까지 동일한 진행 시간으로 유지됩니다. 그런 다음 클럭이 동기화됩니다. 이는 데이터베이스 노드에서의 단기 시스템 클럭 오류가 데이터베이스 로그 시간소인에는 오래 지속되는 영향을 줄 수 있다는 것을 의미합니다.

예를 들어, 연도가 2003년일 때 데이터베이스 파티션 서버 A에서의 시스템 클럭이 잘못하여 2005년 11월 7일로 설정되고, 그 실수가 해당 데이터베이스 파티션 서버에 있는 데이터베이스 파티션에서 갱신 트랜잭션이 커미트된 *후에* 정정되었다고 가정합니다. 데이터베이스가 계속 사용 중이고 시간이 경과하면서 정기적으로 갱신되는 경우, 2003년 11월 7일과 2005년 11월 7일 사이의 임의 시점은 가상으로 롤 포워드 복구를 통해 도달할 수 없습니다. 데이터베이스 파티션 서버 A에서의 COMMIT가 완료되면, 데이터베이스 로그의 시간 소인은 2005로 설정되고, 로그 클럭은 시스템 클럭이 2005년 11월 7일 시간과 일치할 때까지 2005년 11월 7일로 유지됩니다. 이 시간 프레임 내의 특정 시점으로 롤 포워드하려고 하면 조작은 지정된 중지점(2003년 11월 7일)을 넘은 첫 번째 시간소인에서 중지합니다.

DB2가 시스템 클럭에 대한 갱신사항을 제어할 수 없어도, *max_time_diff* 데이터베이스 관리 프로그램 구성 매개변수는 이 유형의 문제점이 발생할 수 있는 기회를 감소시킵니다.

- 이 매개변수의 구성 가능 값 범위는 1분 - 24시간입니다.
- 비카탈로그 파티션에 대해 첫 번째 연결 요청이 작성될 때 데이터베이스 파티션 서버는 해당 시간을 데이터베이스의 카탈로그 파티션으로 보냅니다. 그러면 카탈로그 파티션은 연결을 요청하는 데이터베이스 파티션의 시간과 자체의 시간이 *max_time_diff* 매개변수에 지정된 범위 내에 있는지 점검합니다. 이 범위를 초과하면 연결이 거부됩니다.
- 데이터베이스에서 세 개 이상의 데이터베이스 파티션 서버와 관련되는 갱신 트랜잭션은 갱신이 커미트되기 전에 참여하는 데이터베이스 파티션 서버의 클럭이 동기화된 상태인지 검증해야 합니다. 두 개 이상의 데이터베이스 파티션 서버가 *max_time_diff*

에서 허용하는 한계를 초과하는 시간 차이를 수반하는 경우, 트랜잭션은 올바르지 않은 시간이 다른 데이터베이스 파티션 서버로 전달되지 않도록 롤백됩니다.

클라이언트/서버 시간소인 변환

이 절에서는 클라이언트/서버 환경에서의 시간소인 생성에 대해 설명합니다.

- 롤 포워드 작업의 로컬 시간을 지정하는 경우 모든 메시지도 로컬 시간입니다.

  주: 모든 시간은 카탈로그 데이터베이스 파티션의 서버와, 파티션된 데이터베이스 환경에서 변환됩니다.
- 시간소인 문자열은 서버에서 GMT로 변환되기 때문에 시간은 클라이언트가 아닌 서버의 시간대입니다. 클라이언트가 서버와 다른 시간대에 있으면 서버의 로컬 시간을 사용해야 합니다.
- 시간소인 문자열이 일광 절약 시간으로 인해 시간 변경에 가까운 경우 중지 시간이 시간 변경 이전 또는 이후인지 알고 올바르게 지정하는 것이 중요합니다.

제 5 장 고가용성 솔루션 관리 및 유지보수

DB2 데이터베이스 고가용성 솔루션을 작성, 구성 및 시작한 후에 사용자가 수행해야 하는 진행 중 활동이 있습니다. 데이터베이스 솔루션을 모니터, 유지보수 및 수리하여 클라이언트 응용프로그램이 계속 사용할 수 있도록 유지해야 합니다.

데이터베이스 시스템이 실행할 때 다음 종류의 사건을 모니터하고 응답해야 합니다.

1. 로그 파일을 관리하십시오.

   로그 파일이 더 커져서 아카이브가 필요하며, 일부 로그 파일은 리스토어 작업에 사용할 수 있도록 복사 또는 이동해야 합니다.
2. 다음 유지보수 활동을 수행하십시오.
   - 소프트웨어 설치
   - 하드웨어 업그레이드
   - 데이터베이스 테이블 재구성
   - 데이터베이스 성능 조정
   - 데이터베이스 백업
3. 원활하게 작업 장애 복구하도록 기본 및 보조 또는 대기 데이터베이스를 동기화하십시오.
4. 하드웨어 또는 소프트웨어에서의 예기치 않은 실패를 식별하고 응답하십시오.

로그 파일 관리

DB2 데이터베이스 관리 프로그램은 숫자 스킴을 사용하여 로그 파일 이름을 지정합니다. 이름 지정 전략은 로그 파일 재사용 및 로그 시퀀스에 대한 함축된 의미를 갖습니다. 또한 다음 클라이언트 응용프로그램이 해당 데이터베이스 서버에 연결할 때 클라이언트 응용프로그램 연결이 없는 DB2 데이터베이스는 새 로그 파일을 사용합니다. DB2 데이터 서버의 데이터베이스 로깅 동작에 나타나는 이러한 두 측면은 로그 파일 관리 선택사항에 영향을 줍니다.

데이터베이스 로그 관리 시 다음을 고려하십시오.

- 아카이브된 로그의 번호 지정 스킴은 S0000000.LOG로 시작되며 S9999999.LOG를 통해 계속됩니다. 최대 1천만 개 로그 파일을 수용할 수 있습니다. 데이터베이스 관리 프로그램은 다음 경우에 S0000000.LOG로 재설정됩니다.
  - 데이터베이스 구성 파일이 롤 포워드 복구가 가능하도록 변경됨
  - 데이터베이스 구성 파일이 롤 포워드 복구가 불가능하도록 변경됨
  - S9999999.LOG가 사용됨

DB2 데이터베이스 관리 프로그램은 롤 포워드 복구 수행 여부와 상관없이 데이터베이스를 리스토어한 후 로그 파일 이름을 재사용합니다. 데이터베이스 관리 프로그램은 롤 포워드 복구 중 올바르지 않은 로그가 적용되지 않도록 합니다. DB2 데이터베이스 관리 프로그램이 리스토어 작업 이후 로그 파일 이름을 재사용하는 경우 동일한 이름의 여러 로그 파일을 아카이브할 수 있도록 새 로그 파일은 별도의 디렉토리에 아카이브됩니다. 롤 포워드 복구 중 적용할 수 있도록 복구 실행기록 파일에 로그 파일 위치가 기록됩니다. 롤 포워드 복구에서 올바른 로그를 사용할 수 있어야 합니다.

롤 포워드 조작이 성공적으로 완료되면 사용된 마지막 로그가 절단되고 다음 순차 로그부터 로깅이 시작됩니다. 순차 번호가 롤 포워드 복구 시 사용한 마지막 로그보다 큰 로그 경로 디렉토리의 모든 로그가 재사용됩니다. 절단 시점 이후의 절단된 로그의 항목은 영(0)으로 겹쳐씁니다. 롤 포워드 유틸리티를 호출하기 전에 로그를 복사해야 합니다. User Exit 프로그램을 사용하여 로그를 다른 위치로 복사할 수도 있습니다.

- ACTIVATE DATABASE 명령을 사용하여 데이터베이스가 활성화되지 않은 경우 모든 응용프로그램이 데이터베이스와 연결이 끊어지면 DB2 데이터베이스 관리 프로그램은 현재 로그 파일을 절단합니다. 다음에 응용프로그램이 데이터베이스에 연결할 때 DB2 데이터베이스 관리 프로그램은 새 로그 파일에 로깅하기 시작합니다. 시스템에 크기가 작은 로그 파일이 많이 생성된 경우 ACTIVATE DATABASE 명령 사용을 고려할 수 있습니다. 그러면 응용프로그램을 연결할 때 시작해야 하는 오버헤드가 줄고 대형 로그 파일을 할당하고 절단한 다음, 새 대형 로그 파일을 할당해야 하는 작업의 오버헤드도 줄어듭니다.
- 아카이브된 로그는 데이터베이스에 대한 두 개 이상의 서로 다른 로그 *시퀀스*에 연결될 수 있습니다. 로그 파일 이름은 재사용 가능하기 때문입니다(157 페이지의 그림 8 참조). 예를 들어 백업 2를 복구하려는 경우 두 가지 로그 시퀀스를 사용할 수 있습니다. 전체 데이터베이스 복구 중 특정 시점으로 롤 포워드하고 로그 끝에 도달하기 전에 중지하면 새 로그 시퀀스가 작성됩니다. 두 개의 로그 시퀀스를 조합할 수 없습니다. 첫 번째 로그 시퀀스 범위에 걸친 온라인 백업 이미지를 보유한 경우 이 로그 시퀀스를 사용하여 롤 포워드 복구를 완료해야 합니다.

  복구 후 새 로그 시퀀스를 작성한 경우 이전 로그 시퀀스의 모든 테이블 스페이스 백업 이미지는 유효하지 않습니다. 일반적으로 리스토어에서 인식되지만 데이터베이스 리스토어 작업 바로 다음에 테이블 스페이스 리스토어 작업이 수행되면 리스토어 유틸리티는 이전 로그 시퀀스에서 테이블 스페이스 백업 이미지를 인식하는 데 실패합니다. 실제로 데이터베이스가 롤 포워드될 때까지 사용할 로그 시퀀스는 알 수 없습니다. 테이블 스페이스가 이전 로그 시퀀스인 경우 테이블 스페이스 롤 포워드 조작으로 『가져와야』 합니다. 유효하지 않은 백업 이미지를 사용하는 리스토어 작업이 성공적으로 완료될 수 있지만 해당 테이블 스페이스에서 테이블 스페이스 롤 포워드 조작은 실패하고 테이블 스페이스는 리스토어 보류 상태가 됩니다.

예를 들어 테이블 스페이스 레벨의 백업 작업, 백업 3이 맨 위 로그 시퀀스의 S0000013.LOG 및 S0000014.LOG 사이에서 완료된다고 가정합니다(그림 8 참조). 데이터베이스 레벨 백업 이미지, 백업 2를 사용하여 리스토어 및 롤 포워드하려는 경우 S0000012.LOG를 통해 롤 포워드해야 합니다. 그 다음, 맨 위 로그 시퀀스 또는 최신 맨 아래 로그 시퀀스를 통해 계속 롤 포워드할 수 있습니다. 맨 아래 로그 시퀀스를 통해 롤 포워드하는 경우 테이블 스페이스 레벨 백업 이미지, 백업 3을 사용하여 테이블 스페이스 리스토어 및 롤 포워드 복구를 수행할 수 없습니다.

테이블 스페이스 레벨 백업 이미지, 백업 3을 사용하여 로그 끝에서 테이블 스페이스 롤 포워드 조작을 완료하려면 데이터베이스 레벨 백업 이미지, 백업 2를 리스토어하고 맨 위 로그 시퀀스를 사용하여 롤 포워드해야 합니다. 테이블 스페이스 레벨 백업 이미지, 백업 3이 리스토어되면 로그 끝으로의 롤 포워드 조작을 시작할 수 있습니다.

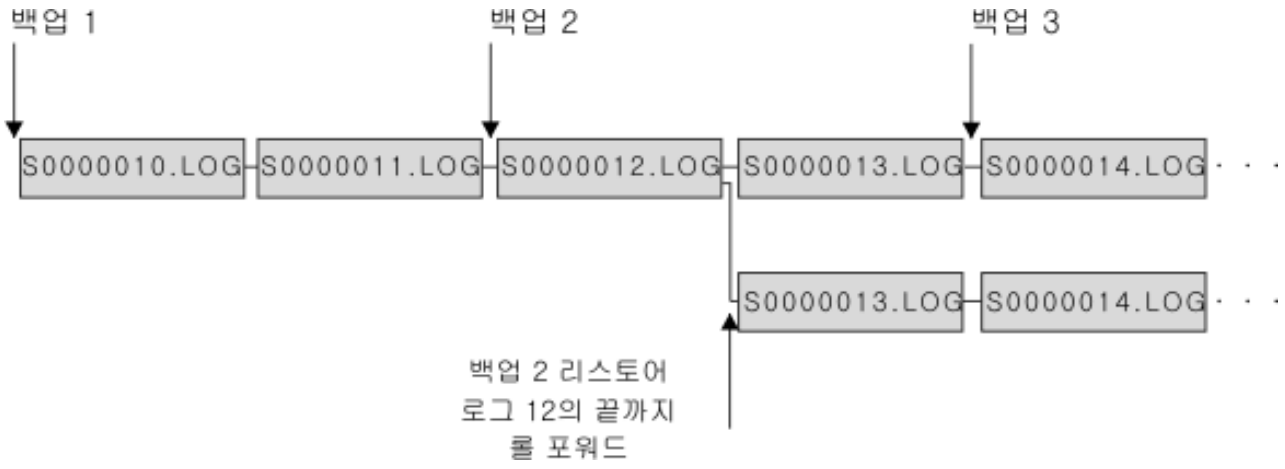


그림 8. 로그 파일 이름 재사용

온 디맨드 로그 아카이브

IBM 데이터 서버에서는 언제라도 복구 가능한 데이터베이스의 사용 중인 로그 닫기 및 아카이브(가능한 경우)를 지원합니다. 이를 통해 전체 로그 파일 세트를 알려진 지점까지 수집한 후 이 로그 파일을 사용하여 대기 데이터베이스를 갱신할 수 있습니다.

ARCHIVE LOG 명령 또는 db2ArchiveLog API를 호출하여 온 디맨드 로그 아카이브를 시작할 수 있습니다.

db2tapemgr을 사용하여 로그 아카이브

db2tapemgr 유틸리티를 사용하여 아카이브된 로그 파일을 테이프 디바이스에 저장할 수 있습니다. db2tapemgr 유틸리티는 디스크에서 지정된 테이프 디바이스로 로그 파일을 복사하고 복사된 로그 파일의 새 위치로 복구 실행기록 파일을 갱신합니다.

구성

데이터베이스 구성 매개변수 LOGARCHMETH1을 테이프에 복사할 로그 파일의 디스크 상의 위치로 설정하십시오. db2tapemgr 유틸리티는 이 LOGARCHMETH1 값을 읽고 복사할 로그 파일을 찾습니다. 파티션된 데이터베이스 환경에서는 복사될 로그 파일을 포함하는 데이터베이스 파티션마다 LOGARCHMETH1 구성 매개변수를 설정해야 합니다.

db2tapemgr 유틸리티는 LOGARCHMETH2 데이터베이스 구성 매개변수를 사용하지 않습니다.

STORE 및 DOUBLE STORE 옵션

아카이브된 로그를 디스크에서 테이프로 전송하려면 STORE 또는 DOUBLE STORE 옵션과 함께 DB2TAPEMGR 명령을 발행하십시오.

- STORE 옵션은 로그 아카이브 디렉토리의 모든 또는 한 범위의 로그 파일을 지정된 테이프 디바이스에 저장하고 디스크에서 파일을 삭제합니다.
- DOUBLE STORE 옵션은 실행기록 파일을 스캔하여 로그가 이전에 테이프에 저장된 적이 있는지 확인합니다.
  - 로그가 이전에 저장된 적이 없으면 DB2TAPEMGR은 로그 파일을 테이프에 저장하지만 디스크에서 삭제하지는 않습니다.
  - 로그가 이전에 저장된 적이 있으면 DB2TAPEMGR은 로그 파일을 테이프에 저장하고 디스크에서 삭제합니다.

  테이프 및 디스크 모두에 아카이브 로그의 사본을 이중으로 저장하거나 동일한 로그를 두 개의 다른 테이프에 저장하려면 DOUBLE STORE를 사용하십시오.

STORE 또는 DOUBLE STORE 옵션과 함께 DB2TAPEMGR 명령을 발행하는 경우, db2tapemgr 유틸리티는 먼저 LOGARCHMETH1 구성 매개변수가 디스크로 설정된 항목을 찾기 위해 실행기록 파일을 스캔합니다. 디스크에 있을 것으로 추정했지만 디스크에 없는 파일을 찾으면 경고를 발행합니다. db2tapemgr 유틸리티가 저장할 로그 파일을 찾지 못하면 조작을 중지하고 메시지를 발행하여 수행할 작업이 없음을 알립니다.

RETRIEVE 옵션

파일을 테이프에서 디스크로 전송하려면 RETRIEVE 옵션과 함께 DB2TAPEMGR 명령을 발행하십시오.

- 사용자의 지정된 기준을 충족하는 모든 아카이브된 로그를 검색하여 디스크에 복사하려면 RETRIEVE ALL LOGS 또는 LOGS n TO n 옵션을 사용하십시오.
- 롤 포워드 작업을 수행하기 위해 필요한 모든 아카이브된 로그를 검색하여 디스크에 복사하려면 RETRIEVE FOR ROLLFORWARD TO POINT-IN-TIME 옵션을 사용하십시오.
- 테이프에서 실행기록 파일을 검색하여 디스크에 복사하려면 RETRIEVE HISTORY FILE 옵션을 사용하십시오.

동작

- db2tapemgr 유틸리티가 디스크에서 로그 파일을 찾으면 테이프 헤더를 읽어서 로그 파일을 테이프에 쓸 수 있는지 확인합니다. 또한 현재 테이프에 있는 파일의 실행기록을 갱신합니다. 갱신에 실패하면 조작이 중지되고 오류 메시지가 표시됩니다.
- 테이프가 쓰기 가능하면 db2tapemgr 유틸리티는 로그를 테이프에 복사합니다. 파일이 복사된 후에는 디스크에서 로그 파일이 삭제됩니다. 마지막으로 db2tapemgr 유틸리티는 실행기록 파일을 테이프로 복사한 후 디스크에서 파일을 삭제합니다.
- db2tapemgr 유틸리티는 로그 파일을 테이프에 추가하지 않습니다. 저장 조작이 전체 테이프를 채우지 못할 경우 미사용 스페이스는 낭비됩니다.
- db2tapemgr 유틸리티는 제공된 테이프에 한 번만 로그 파일을 저장합니다. 이러한 제한사항은 테이프 미디어에 쓸 때 생기는 문제점(예: 테이프의 늘어짐)을 방지하기 위해 존재합니다.
- 파티션된 데이터베이스 환경에서, db2tapemgr 유틸리티는 한 번에 하나의 데이터베이스 파티션에 대해서만 실행됩니다. DB2TAPEMGR 명령의 ON DBPARTITIONNUM 옵션을 사용하여 데이터베이스 파티션 번호를 지정하여, 데이터베이스 파티션마다 적절한 명령을 실행해야 합니다. 또한 각각의 데이터베이스 파티션이 테이프 디바이스에 대한 액세스 권한을 가지고 있는지 확인해야 합니다.

예

다음 예는 DB2TAPEMGR 명령을 사용하여 데이터베이스 파티션 번호 0의 데이터베이스 샘플에 대한 기본 아카이브 로그 경로의 모든 로그 파일을 테이프 디바이스에 저장한 후 아카이브 로그 경로에서 이 파일을 제거하는 방법을 보여줍니다.

```
db2tapemgr db sample on dbpartitionnum 0 store on /dev/rmt0.1 all logs
```

다음 예는 기본 아카이브 로그 경로에서 처음 10개의 로그 파일을 테이프 디바이스에 저장한 후 아카이브 로그 경로에서 이 파일을 제거하는 방법을 보여줍니다.

```
db2tapemgr db sample on dbpartitionnum store on /dev/rmt0.1 10 logs
```

다음 예는 기본 아카이브 로그 경로에서 처음 10개의 로그 파일을 테이프 디바이스에 저장한 후 동일한 로그 파일을 두 번째 테이프에 저장하고 아카이브 로그 경로에서 이 파일을 제거하는 방법을 보여줍니다.

```
db2tapemgr db sample on dbpartitionnum double store on /dev/rmt0.1 10 logs
db2tapemgr db sample on dbpartitionnum double store on /dev/rmt1.1 10 logs
```

다음 예는 테이프의 모든 로그 파일을 디렉토리로 검색하는 방법을 보여줍니다.

```
db2tapemgr db sample on dbpartitionnum retrieve all logs from /dev/rmt1.1
      to /home/dbuser/archived_logs
```

User Exit 프로그램을 사용하여 로그 파일 아카이브 및 검색 자동화

아카이브 또는 검색 작업을 수행하기 위해 DB2 데이터베이스 관리 프로그램이 호출하는 User Exit 프로그램을 작성하여 로그 파일 아카이브 및 검색을 자동화할 수 있습니다.

DB2 데이터베이스 관리 프로그램이 사용자의 User Exit 프로그램을 호출할 때 다음 상황이 발생합니다.

- 데이터베이스 관리 프로그램이 제어를 User Exit 프로그램에 전달합니다.
- 데이터베이스 관리 프로그램이 매개변수를 User Exit 프로그램에 전달합니다.
- 완료되면 User Exit 프로그램이 리턴 코드를 다시 데이터베이스 관리 프로그램으로 전달합니다.

구성

로그 파일 아카이브 및 검색을 위해 User Exit 프로그램을 호출하기 전에, logarchmeth1 데이터베이스 구성 매개변수가 USEREXIT로 설정되었는지 확인하십시오. 이렇게 하면 롤 포워드 복구에도 데이터베이스를 사용할 수 있게 됩니다.

User Exit 프로그램 요구사항

- User Exit 프로그램의 실행할 수 있는 파일을 db2uext2라고 해야 합니다.
- User Exit 프로그램이 사용 중인 로그 경로에서 아카이브 로그 경로로 로그 파일을 이동하지 않고 복사해야 합니다. 사용 중인 로그 경로에서 로그 파일을 제거하면 안 됩니다. 사용 중인 로그 경로에서 로그 파일을 제거하면 DB2 데이터베이스가 실패 시 성공적으로 복구하지 못할 수도 있습니다.

  DB2 데이터베이스의 복구 중에는 사용 중인 로그 경로에 로그 파일이 있어야 합니다. DB2 데이터베이스 서버는 아카이브된 로그 파일이 더 이상 복구에 필요하지 않을 때 사용 중인 로그 경로에서 이 로그 파일을 제거합니다.
- User Exit 프로그램이 오류 조건을 처리해야 합니다. DB2 데이터베이스 관리 프로그램은 제한된 리턴 조건 세트만 처리할 수 있으므로 User Exit 프로그램이 오류를 처리해야 합니다.

163 페이지의 『User Exit 오류 조절』의 내용을 참조하십시오.

- 각 DB2 데이터베이스 관리 프로그램 인스턴스는 하나의 User Exit 프로그램만 호출할 수 있습니다. 데이터베이스 관리 프로그램 인스턴스는 하나의 User Exit 프로그램만 호출할 수 있으므로, 수행해야 하는 각 작업에 대한 섹션이 있는 User Exit 프로그램을 설계해야 합니다.

User Exit 프로그램 샘플

지원되는 모든 플랫폼에 대해 User Exit 프로그램 샘플이 제공됩니다. 사용자의 특수한 요구사항에 맞게 이 프로그램을 수정할 수 있습니다. 샘플 프로그램에는 사용자가 가장 효율적으로 사용할 수 있도록 도와주는 정보가 있는 주석이 표시되어 있습니다.

User Exit 프로그램이 사용 중인 로그 경로에서 아카이브 로그 경로로 로그 파일을 복사해야 한다는 점에 유의해야 합니다. 사용 중인 로그 경로에서 로그 파일을 제거하면 안됩니다. (이와 같이 하면 데이터베이스 복구 중에 문제점이 발생할 수 있습니다.) DB2는 아카이브된 로그 파일이 더 이상 복구에 필요하지 않을 때 사용 중인 로그 경로에서 이 로그 파일을 제거합니다.

다음은 DB2 Data Server와 함께 제공되는 샘플 User Exit 프로그램의 설명입니다.

- **UNIX 기반 시스템**

  UNIX 기반 시스템용 DB2 Data Serve의 User Exit 샘플 프로그램은 `sqllib/samples/c` 서브디렉토리에 있습니다. 제공되는 샘플이 C로 코딩되어도, User Exit 프로그램은 다른 프로그램 언어로 작성할 수 있습니다.

  User Exit 프로그램은 이름이 `db2uext2`인 실행할 수 있는 파일이어야 합니다.

  UNIX 기반 시스템에 대해 네 가지의 샘플 User Exit 프로그램이 있습니다.

  – `db2uext2.ctsm`

    이 샘플은 Tivoli Storage Manager를 사용하여 데이터베이스 로그 파일을 아카이브 및 검색합니다.

  – `db2uext2.ctape`

    이 샘플은 테이프 미디어를 사용하여 데이터베이스 로그 파일을 아카이브 및 검색합니다.

  – `db2uext2.cdisk`

    이 샘플은 운영 체제 COPY 명령과 디스크 미디어를 사용하여 데이터베이스 로그 파일을 아카이브 및 검색합니다.

  – `db2uxt2.cxbsa`

이 샘플은 X/Open 그룹에서 발행한 XBSA Draft 0.8에서 작동합니다. 데이터베이스 로그 파일을 아카이브하고 검색하는 데 사용할 수 있습니다. 이 샘플은 AIX에서만 지원됩니다.

- **Windows 운영 체제**

  Windows 운영 체제용 DB2 Data Server의 User Exit 프로그램은 sqllib¥samples¥c 서브디렉토리에 있습니다. 제공되는 샘플이 C로 코딩되어도, User Exit 프로그램은 다른 프로그램 언어로 작성할 수 있습니다.

  User Exit 프로그램은 이름이 db2uext2인 실행할 수 있는 파일이어야 합니다.

  Windows 운영 체제에 대해 두 가지의 샘플 User Exit 프로그램이 있습니다.

  – db2uext2.ctsm

    이 샘플은 Tivoli Storage Manager를 사용하여 데이터베이스 로그 파일을 아카이브 및 검색합니다.

  – db2uext2.cdisk

    이 샘플은 운영 체제 COPY 명령과 디스크 미디어를 사용하여 데이터베이스 로그 파일을 아카이브 및 검색합니다.

User Exit 프로그램 호출 형식

DB2 데이터베이스 관리 프로그램이 User Exit 프로그램을 호출할 때 매개변수 세트(데이터 유형 CHAR)를 프로그램에 전달합니다.

명령 구문

```
db2uext2 -OS<os> -RL<db2rel> -RQ<request> -DB<dbname>
-NN<nodenum> -LP<logpath> -LN<logname> -AP<tsmpasswd>
-SP<startpage> -LS<logsize>
```

os
인스턴스가 실행 중인 플랫폼을 지정합니다. 유효한 값은 AIX, Solaris, HP-UX, SCO, Linux 및 NT입니다.

db2rel
DB2 릴리스 레벨을 지정합니다(예를 들어, SQL07020).

request
요청 유형을 지정합니다. 유효한 값은 ARCHIVE 및 RETRIEVE입니다.

dbname
데이터베이스 이름을 지정합니다.

nodenum
로컬 노드 번호(예: 5)를 지정합니다.

logpath

로그 파일의 완전한 경로를 지정합니다. 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 지정합니다. (데이터베이스 구성 매개변수 *tsm_password*의 값을 이전에 지정한 경우에는 그 값이 User Exit 프로그램으로 전달됩니다.)

startpage

로그 Extent가 시작하는 디바이스의 4KB 오프셋 페이지 수를 지정합니다.

logsize

로그 Extent 크기를 4KB 페이지 수 단위로 지정합니다. 이 매개변수는 로깅에 원시 디바이스가 사용되는 경우에만 유효합니다.

User Exit 오류 조절

로그 파일 아카이브 및 검색을 자동화하기 위한 User Exit 프로그램을 작성하는 경우, User Exit 프로그램은 User Exit 프로그램을 호출한 DB2 데이터베이스 관리 프로그램에 리턴 코드를 전달합니다. DB2 데이터베이스 관리 프로그램은 제한된 특정 오류 코드 목록만 조절할 수 있습니다. 그러나 User Exit 프로그램은 다른 종류의 많은 오류 조건을 발견할 수 있습니다(예: 운영 체제 오류). User Exit 프로그램은 발견하는 오류 조건을 데이터베이스 관리 프로그램이 조절할 수 있는 오류 코드에 맵핑해야 합니다.

표 8은 User Exit 프로그램이 리턴할 수 있는 코드를 표시하고 이 코드를 데이터베이스 관리 프로그램이 해석하는 방법을 설명합니다. 리턴 코드가 표에 나열되어 있지 않으면 해당 값이 32인 것으로 처리됩니다.

표 8. User Exit 프로그램 리턴 코드. 아카이브 및 검색 작업에만 적용됩니다.

리턴 코드	설명
0	성공.
4	임시 자원 오류가 발견되었습니다.[a]
8	운영자 개입이 필요합니다.[a]
12	하드웨어 오류.[b]
16	User Exit 프로그램이나, 프로그램에 사용되는 소프트웨어 기능의 오류.[b]
20	User Exit 프로그램에 전달된 하나 이상의 매개변수에 대한 오류. User Exit 프로그램이 지정된 매개변수를 올바르게 처리하는지 검증하십시오.[b]
24	User Exit 프로그램을 찾을 수 없습니다.[b]
28	입/출력(I/O) 실패 또는 운영 체제로 인해 발생한 오류.[b]
32	User Exit 프로그램이 사용자에 의해 종료되었습니다.[b]
255	User Exit 프로그램에 의해 발생한 오류로, 실행 파일에 대해 라이브러리 파일을 로드할 수 없습니다.[c]

표 8. User Exit 프로그램 리턴 코드 (계속). 아카이브 및 검색 작업에만 적용됩니다.

리턴 코드	설명
[a] 아카이브 또는 검색 요청에 대해, 4 또는 8 리턴 코드는 5분 안에 재시도하도록 합니다. User Exit 프로그램이 계속 동일한 로그 파일에 대한 검색 요청에 4 또는 8을 리턴하면, DB2는 성공할 때까지 계속 재시도합니다. (이는 롤 포워드 작업이나 db2ReadLog API 호출(복제 유틸리티에서 사용되는)에 적용됩니다.) [b] User Exit 요청은 5분 동안 일시중단됩니다. 이 시간 동안, 오류 조건을 야기한 요청을 포함하여 모든 요청이 무시됩니다. 이 5분 동안 일시중단 후에는 다음 요청이 처리됩니다. 이 요청이 오류 없이 처리되는 경우, 새 User Exit 요청 처리가 계속되고 DB2는 이전에 실패하거나 일시중단된 아카이브 요청을 다시 발행합니다. 8을 초과하는 리턴 코드가 재시도 중에 생성되면 요청은 추가 5분 동안 일시중단됩니다. 문제점이 정정되거나 데이터베이스가 중지 또는 재시작할 때까지 5분 일시중단은 계속됩니다. 모든 응용프로그램이 데이터베이스에서 연결이 끊어진 경우 DB2는 이전에 성공적으로 아카이브되지 않았을 수 있는 로그 파일에 대한 아카이브 요청을 발행합니다. User Exit 프로그램이 로그 파일 아카이브에 실패하는 경우, 디스크가 로그 파일로 채워져서 성능이 저하될 수 있습니다. 디스크가 가득 차면, 데이터베이스 관리 프로그램은 데이터베이스 갱신사항에 대한 추가 응용프로그램 요청을 승인하지 않습니다. User Exit 프로그램이 로그 파일 검색을 위해 호출된 경우, 롤 포워드 복구가 일시중단되지만 ROLLFORWARD STOP 옵션을 지정하지 않은 경우 중지되지는 않습니다. STOP 옵션을 지정하지 않은 경우 문제점을 정정하고 복구를 다시 시작할 수 있습니다. [c] User Exit 프로그램이 오류 코드 255를 리턴하는 경우, 실행 파일에 대해 라이브러리 파일을 로드할 수 없습니다. 이를 검증하려면, 수동으로 User Exit 프로그램을 호출하십시오. 자세한 정보가 표시됩니다. **주:** 아카이브 및 검색 작업 중에, 0 및 4를 제외한 모든 리턴 코드에 대해 경보 메시지가 발행됩니다. 경보 메시지에는 User Exit 프로그램의 리턴 코드와, User Exit 프로그램에 제공된 입력 매개변수의 사본이 포함됩니다.	

로그 파일 할당 및 제거

데이터베이스 로그 디렉토리의 로그 파일은 응급 복구에 필요할 수 있는 경우 제거되지 않습니다. 응급 복구에 필요한 로그 파일을 사용 중인 로그라고 합니다. 응급 복구에 필요하지 않은 로그 파일을 아카이브된 로그라고 합니다.

무한 로깅이 사용 가능하도록 설정한 경우, 로그 파일이 성공적으로 아카이브되면 삭제됩니다. logarchmeth1 데이터베이스 구성 매개변수가 OFF로 설정되지 않으면, 전체 로그 파일은 더 이상 응급 복구에 필요하지 않게 된 후에만 제거 후보가 됩니다.

새 로그 파일 할당 및 이전 로그 파일 제거 프로세스는 *logarchmeth1* 데이터베이스 구성 매개변수의 설정에 따라 다릅니다.

Logarchmeth1 및 *Logarchmeth2*가 **OFF로 설정되어 있습니다.**

순환 로깅이 사용됩니다. 롤 포워드 복구는 순환 로깅에 대해 지원되지 않지만 응급 복구는 지원됩니다.

순환 로깅 중, 2차 로그가 아닌 새 로그 파일은 생성되지 않으며 이전 로그 파일은 삭제되지 않습니다. 로그 파일은 순환 형태로 처리됩니다. 즉, 마지막 로그 파일이 가득 차면 DB2는 첫 번째 로그 파일에 쓰기 시작합니다.

모든 로그 파일이 사용 중이고 순환 로깅 프로세스가 첫 번째 로그 파일로 맵핑될 수 없는 경우 로그가 가득 차는 상황이 발생할 수 있습니다. 2차 로그 파일은 모든 1차 로그 파일이 사용 중이고 가득 찼을 때 작성됩니다. 데이터베이스가 비활성화되거나 2차 로그 파일이 사용 중인 스페이스가 사용 중인 로그 파일에 필요한 경우 2차 로그 파일은 삭제됩니다.

*Logarchmeth1*이 **LOGRETAIN**으로 설정됩니다.

아카이브 로깅이 사용됩니다. 데이터베이스는 복구 가능한 데이터베이스입니다. 롤 포워드 복구 및 응급 복구 모두 사용 가능합니다. 로그 파일을 아카이브하면, 새 로그 파일에 디스크 스페이스를 재사용할 수 있도록 사용 중인 로그 경로에서 로그 파일을 삭제해야 합니다. 로그 파일이 가득 찰 때마다, DB2는 다른 로그 파일에 레코드를 쓰는 것을 시작하고 (최대 1차 및 2차 로그 수에 도달하지 않은 경우) 새 로그 파일을 작성합니다.

*Logarchmeth1*은 **OFF** 또는 **LOGRETAIN**이 아닌 다른 값으로 설정됩니다.

아카이브 로깅이 사용됩니다. 데이터베이스는 복구 가능한 데이터베이스입니다. 롤 포워드 복구 및 응급 복구 모두 사용 가능합니다. 로그 파일이 가득 차면 자동으로 아카이브됩니다.

로그 파일은 보통 삭제되지 않습니다. 대신, 새 로그 파일이 필요한데 사용할 수 없는 경우 아카이브된 로그 파일의 이름이 바뀌고 다시 사용됩니다. 아카이브된 로그 파일은 닫히고 로그 아카이브 디렉토리로 복사되고 나면 삭제하거나 이름을 바꿀 수 없습니다. DB2는 새 로그 파일이 필요할 때까지 기다린 후 가장 오래된 아카이브 로그의 이름을 바꿉니다. 복구 중에 데이터베이스 디렉토리로 이동된 로그 파일은 더 이상 필요하지 않을 때 복구 프로세스에서 제거됩니다. DB2가 로그 스페이스를 모두 소모할 때까지 데이터베이스 디렉토리에서 이전 로그 파일이 표시됩니다.

로그 파일이 아카이브될 때 오류가 발생하는 경우, 아카이브는 ARCHRETRYDELAY 데이터베이스 구성 매개변수에 지정된 시간 동안 일시 중단됩니다. 또한 NUMARCHRETRY 데이터베이스 구성 매개변수를 사용하여 DB2가 로그 파일을 장애 복구 디렉토리(FAILARCHPATH 데이터베이스 구성 매개변수에 지정된)로 아카이브하려고 시도하기 전에 기본 또는 보조 아카이브 디렉토리로 로그 파일을 아카이브하려고 합니다. NUMARCHRETRY는 FAILARCHPATH 데이터베이스 구성 매개변수가 설정된 경우에만 사용됩니다. NUMARCHRETRY가 0으로 설정된 경우 DB2는 1차 또는 2차 로그 경로에서 연속으로 아카이브하려고 시도합니다.

이전 로그 파일을 제거하기 위한 가장 쉬운 방법은 데이터베이스를 재시작하는 것입니다. 데이터베이스가 재시작되면 새 로그 파일과 User Exit 프로그램이 아카이브에 실패한 로그 파일만 데이터베이스 디렉토리에서 발견됩니다.

데이터베이스가 재시작되면, 데이터베이스 로그 디렉토리에 있는 최소 로그 수가 *logprimary* 데이터베이스 구성 매개변수를 사용하여 구성할 수 있는 1차 로

그 수와 같게 됩니다. 로그 디렉토리에서 1차 로그 수보다 더 많이 발견될 수 있습니다. 이러한 상황은 데이터베이스가 종료되었을 때 로그 디렉토리에 있는 비어 있는 로그 수가 데이터베이스 재시작 시 *logprimary* 구성 매개변수의 값보다 긴 경우에 발생할 수 있습니다. 이는 *logprimary* 구성 매개변수가 종료되는 데이터베이스와 재시작되는 데이터베이스 사이에 변경되는 경우, 또는 2차 로그가 할당되었지만 전혀 사용되지 않는 경우에 발생합니다.

데이터베이스가 재시작될 때 비어 있는 로그 수가 *logprimary* 구성 매개변수에 지정된 1차 로그 수보다 적은 경우 추가 로그 파일이 할당되어 차이를 채웁니다. 데이터베이스 디렉토리에서 사용 가능한 1차 로그 수보다 많은 로그가 비어 있는 경우 데이터베이스는 데이터베이스 디렉토리에서 발견할 수 있는 만큼의 사용 가능한 빈 로그로 재시작할 수 있습니다. 데이터베이스가 종료된 후, 작성된 2차 로그 파일은 데이터베이스가 재시작될 때 사용 중인 로그 경로에 유지됩니다.

백업 이미지와 함께 로그 파일 포함

온라인 백업 작업을 수행할 때 데이터베이스를 리스토어하고 복구하는 데 필요한 로그 파일이 백업 이미지에 포함됨을 지정할 수 있습니다. 이는 재해 복구 사이트에 백업 이미지를 제공해야 하는 경우 별도로 로그 파일을 보내거나 로그 파일을 함께 패키징하지 않아도 됨을 의미합니다. 또한 온라인 백업의 일관성을 보증하기 위해 필요한 로그 파일을 결정하지 않아도 됩니다. 이로서 성공적인 복구를 위해 필요한 로그 파일이 삭제되지 않도록 보호합니다.

이 기능을 사용하려면 BACKUP DATABASE 명령의 INCLUDE LOGS 옵션을 지정하십시오. 이 옵션을 지정할 때 백업 유틸리티는 현재 사용 중인 로그 파일을 절단하고 필요한 로그 Extent 세트를 백업 이미지에 복사합니다.

백업 이미지에서 로그 파일을 리스토어하려면 RESTORE DATABASE 명령의 LOGTARGET 옵션을 사용하고 DB2 서버에 존재하는 완전한 경로를 지정하십시오. 리스토어 데이터베이스 유틸리티는 로그 파일의 이미지를 목표 경로에 씁니다. 동일한 이름의 로그 파일이 이미 목표 경로에 있는 경우 리스토어 작업이 실패하고 오류가 리턴됩니다. LOGTARGET 옵션을 지정하지 않으면 백업 이미지에서 로그 파일이 리스토어되지 않습니다.

LOGTARGET 옵션이 지정되고 백업 이미지에 로그 파일이 포함되지 않는 경우, 테이블 스페이스 데이터 리스토어를 시도하기 전에 오류가 리턴됩니다. 유효하지 않거나 읽기 전용 경로를 지정한 경우에도 리스토어 작업이 실패합니다. LOGTARGET 옵션이 지정된 데이터베이스 또는 테이블 스페이스 리스토어 중에 하나 이상의 로그 파일을 추출할 수 없으면 리스토어 작업이 실패하고 오류가 리턴됩니다.

또한 백업 이미지에 저장된 로그 파일만 리스토어할 것을 선택할 수 있습니다. 이와 같이 하려면, RESTORE DATABASE 명령의 LOGTARGET 옵션과 함께 LOGS 옵션

을 지정하십시오. 이 모드에서 로그 파일을 리스토어할 때 리스토어 작업에서 문제점이 발생하면 리스토어 작업이 실패하고 오류가 리턴됩니다.

자동 증분 리스토어 작업 중에, 리스토어 작업의 목표 이미지에 포함된 로그만 백업 이미지에서 검색됩니다. 증분 리스토어 프로세스 중에 참조되는 중간 이미지에 포함된 로그는 중간 백업 이미지에서 추출되지 않습니다. 수동 증분 리스토어 중에, 로그 파일을 포함하는 백업 이미지를 리스토어할 때 로그 대상 디렉토리를 지정하는 경우 해당 백업 이미지의 로그 파일이 리스토어됩니다.

로그 파일을 포함하는 온라인 백업 이미지에서 리스토어된 데이터베이스를 롤 포워드하는 경우 SQL1268N 오류가 발생할 수 있습니다. 이 오류는 로그 검색 시 수신된 오류로 인해 롤 포워드 복구가 중지되었음을 표시합니다. 이 오류는 백업 이미지를 리스토어하려고 하는 목표 시스템이 트랜잭션 로그를 아카이브하기 위해 소스 시스템이 사용하는 기능에 액세스할 수 없을 때 생성됩니다.

데이터베이스를 백업할 때 BACKUP DATABASE 명령의 INCLUDE LOGS 옵션을 지정한 후 해당 백업 이미지를 사용하는 리스토어 작업 및 롤 포워드 작업을 연속으로 수행하면, DB2는 백업 이미지에 로그가 포함된 경우에도 데이터베이스를 롤 포워드할 때 추가 트랜잭션 로그를 계속 검색합니다. 이는 더 이상 로그가 발견되지 않을 때까지 추가 트랜잭션 로그를 계속 검색하기 위한 표준 롤 포워드 동작입니다. 시간소인이 같은 여러 개의 로그 파일이 있을 수 있습니다. 결국 DB2는 사용자가 데이터베이스를 롤 포워드하는 특정 시점과 일치하는 첫 번째 시간소인을 발견하는 즉시 중지하지 않습니다. 해당 시간소인을 가지고 있는 다른 로그 파일이 있을 수 있기 때문입니다. 대신 DB2는 지정된 특정 시점보다 큰 시간소인을 찾을 때까지 트랜잭션 로그를 계속 찾습니다.

더 이상 로그를 찾을 수 없는 경우 롤 포워드는 성공적으로 종료합니다. 그러나 추가 트랜잭션 로그 파일을 검색하는 동안 오류가 발생하는 경우 오류 SQL1268N이 리턴됩니다. 오류 SQL1268N이 발생할 수 있습니다. 초기 리스토어 중에 특정 데이터베이스 구성 매개변수가 재설정되거나 겹쳐서 쓰여졌기 때문입니다. 이러한 데이터베이스 구성 매개변수 중 세 가지는 TSM 매개변수, TSM_NODENAME, TSM_OWNER 및 TSM_PASSWORD입니다. 이 매개변수는 모두 NULL로 재설정됩니다. 로그 끝으로 롤 포워드하려면, 이 데이터베이스 구성 매개변수를 롤 포워드 작업 이전의 소스 시스템에 따라 재설정해야 합니다. 또는 ROLLFORWARD DATABASE 명령을 발행할 때 NORETRIEVE 옵션을 지정할 수 있습니다. 그러면 DB2 데이터베이스 시스템이 누락 가능성이 있는 트랜잭션 로그를 다른 곳에서 확보하려고 시도하지 않습니다.

주:

1. 이 기능은 오프라인 백업의 경우 지원되지 않습니다.
2. 로그가 온라인 백업 이미지에 포함되는 경우, 결과 이미지는 버전 8.2 이전의 DB2 데이터베이스 릴리스에서 리스토어할 수 없습니다.

로그 파일의 우연한 유실 예방

데이터베이스를 삭제해야 하거나 특정 시점 롤 포워드 복구를 수행해야 하는 경우 향후 복구 조작이 필요한 로그 파일을 유실할 수 있습니다. 이때 현재 데이터베이스 로그 경로 디렉토리에 있는 모든 로그를 복사하는 것이 중요합니다.

다음 시나리오를 고려해 보십시오.

- 리스토어 작업 전에 데이터베이스를 삭제하려는 경우 DROP DATABASE 명령을 실행하기 전에 사용 중인 로그 경로에 로그 파일을 저장해야 합니다. 데이터베이스를 리스토어한 후 롤 포워드 복구에 이 로그 파일이 필요할 수 있습니다. 일부 로그 파일은 데이터베이스를 삭제하기 전에 아카이브될 수 없기 때문입니다. 정상적으로는 RESTORE 명령을 실행하기 전에 데이터베이스를 삭제하지 않아도 됩니다. 그러나 RESTORE 명령에 실패한 Extent로 손상되었으므로 데이터베이스를 삭제해야 할 수도 있습니다. 또는 DROP DATABASE 명령의 AT® NODE 옵션을 지정하여 한 데이터베이스 파티션에서 데이터베이스를 삭제해야 할 수 있습니다. 또한 리스토어 작업으로 처음부터 시작하기 전에 데이터베이스를 삭제하려는 경우도 있습니다.
- 특정 시점으로 데이터베이스를 롤 포워드하는 경우 지정한 시간소인 이후의 로그 데이터를 겹쳐씁니다. 특정 시점 롤 포워드 조작을 완료하고 데이터베이스에 다시 연결한 후 나중의 특정 시점으로 데이터베이스를 실제로 롤 포워드해야 하는지 판별하는 경우 로그를 이미 겹쳐썼으므로 이를 수행할 수 없습니다. 로그 파일의 원래 세트가 아카이브될 수 있습니다. 그러나 DB2는 User Exit 프로그램을 호출하여 새로 생성된 로그 파일을 자동으로 아카이브할 수 있습니다. User Exit 프로그램이 작성된 방법에 따라 아카이브 로그 디렉토리에 있는 로그 파일의 원래 세트를 겹쳐쓸 수 있습니다. 로그 파일의 원래 세트 및 새 세트 모두 아카이브 로그 디렉토리에 있는 경우에도(동일한 파일의 다른 버전인 경우) 향후 복구 조작에 사용해야 하는 로그 세트를 판별해야 합니다.

사용 가능성에 대한 유지보수 영향 최소화

DB2 데이터베이스 솔루션에 대해 소프트웨어 또는 하드웨어 업그레이드, 데이터베이스 성능 조정, 데이터베이스 백업, 통계 수집 및 비즈니스 목적을 위한 모니터링 같은 유지보수를 수행해야 합니다. 솔루션의 사용 가능성에 대해 유지보수 수행이 갖는 영향의 최소화에는 오프라인 유지보수의 주의 깊은 스케줄링 및 온라인 유지보수의 사용 가능성 영향을 줄이는 DB2 기능 사용이 포함됩니다.

다음 단계를 사용하여 DB2 데이터베이스 솔루션의 사용 가능성에 대한 유지보수 영향을 최소화하기 전에 다음을 수행해야 합니다.

- 자동 유지보수 구성 및
- 고가용성 재해 복구(HADR) 기능 설치.

1. 자동 유지보수가 사용자 대신 유지보수를 수행하도록 허용하십시오.

DB2 데이터베이스는 많은 데이터베이스 유지보수 활동을 자동화할 수 있습니다. 자동 유지보수가 구성된 후에는 사용자가 해당 유지보수를 수행하기 위한 추가 단계를 수행하지 않고 유지보수가 발생합니다.

2. DB2 고가용성 재해 복구(HADR) 롤링 업그레이드를 사용하여 기타 유지보수 활동의 영향을 최소화하십시오.

   소프트웨어 또는 하드웨어를 업그레이드 중인 경우 또는 일부 데이터베이스 관리 프로그램 구성 매개변수를 수정 중인 경우 HADR 기능을 사용하여 사용 가능성의 인터럽트를 최소화하면서 필요한 변경을 수행할 수 있습니다. HADR에 의해 사용 가능한 이 유연한 변경을 롤링 업그레이드라고 합니다.

   일부 유지보수 활동은 HADR 환경에서도 유지보수를 수행하기 전에 데이터베이스를 종료해야 합니다. 일부 조건에서는 HADR 데이터베이스 종료 프로시저가 표준 데이터베이스 종료 프로시저보다 약간 어렵습니다. 즉, HADR 데이터베이스가 연결된 클라이언트 응용프로그램에 의해 시작되는 경우 DEACTIVATE DATABASE 명령을 사용해야 합니다.

DB2 고가용성 재해 복구(HADR) 중지

DB2 고가용성 재해 복구(HADR) 기능을 사용 중인 경우, 두 DB2 데이터베이스 시스템인 기본 및 대기에서 유지보수를 수행하는 것이 하나의 독립형 데이터베이스 서버에 유지보수를 수행하는 것보다 더 복잡할 수 있습니다. 유지보수를 수행하기 위해 HADR을 중지해야 하는 경우 STOP HADR 명령을 사용하여 HADR 기능을 중지하십시오. 대기 데이터베이스 시스템에서만 유지보수를 수행 중인 경우 대기 데이터베이스에서만 HADR을중지해야 합니다. HADR 사용을 완전히 중지하려면 두 데이터베이스 모두에서 HADR을 중지할 수 있습니다.

경고: 지정된 데이터베이스를 중지하기 원하지만 아직 HADR 기본 또는 대기 데이터베이스로서 역할을 유지하려는 경우 STOP HADR 명령을 실행하지 마십시오. STOP HADR 명령을 실행하는 경우 데이터베이스는 표준 데이터베이스가 되며 HADR 데이터베이스로서 조작을 재개하기 위해 재초기화가 필요할 수 있습니다. 대신 DEACTIVATE DATABASE 명령을 실행하십시오.

기본 또는 대기 데이터베이스에 대해서만 STOP HADR 명령을 발행할 수 있습니다. 이 명령을 표준 데이터베이스에 대해 발행하는 경우 오류가 리턴됩니다.

명령행 처리기(CLP), 제어 센터에서 고가용성 재해 복구(HADR) 관리 창, 또는 db2HADRStop API을 사용하여 HADR을 중지할 수 있습니다.

CLP를 사용하여 기본 또는 대기 데이터베이스에서 HADR 조작을 중지하려면, HADR 조작을 중지하려는 데이터베이스에서 STOP HADR 명령을 발행하십시오.

다음 예에서 HADR 조작이 SOCKS 데이터베이스에서 중지됩니다.

```
STOP HADR ON DATABASE SOCKS
```

이 명령을 비활성 기본 데이터베이스에 대해 발행하는 경우 데이터베이스는 표준 데이터베이스로 전환하고 오프라인으로 남아있습니다.

이 명령을 비활성 대기 데이터베이스에 대해 발행하는 경우 데이터베이스는 표준 데이터베이스로 전환하고 롤 포워드 보류 상태에 놓이며 오프라인으로 남아있습니다.

이 명령을 활성 기본 데이터베이스에 대해 발행하는 경우 로그는 대기 데이터베이스로 출시되기를 중지하고 모든 HADR 엔진 디스패치 가능 장치(EDU)가 기본 데이터베이스에서 종료됩니다. 데이터베이스는 표준 데이터베이스로 전환되고 오프라인으로 남습니다. 트랜잭션 처리는 계속될 수 있습니다. AS PRIMARY 옵션을 사용한 START HADR 명령을 발행하여 데이터베이스의 역할을 다시 기본 데이터베이스로 전환할 수 있습니다.

이 명령을 활성 대기 데이터베이스에 대해 발행하는 경우, 표준 데이터베이스로 변환하려고 시도하기 전에 대기 데이터베이스를 비활성화해야 함을 표시하는 오류 메시지가 리턴됩니다.

HADR 중지 창을 열려면 다음을 수행하십시오.

1. 제어 센터에서 HADR을 관리하려는 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 고가용성 재해 복구 → 관리를 누르십시오. 고가용성 재해 복구 관리 창이 열립니다.
2. HADR 중지를 누르십시오. HADR 중지 창이 열립니다.
3. 한 데이터베이스에서만 HADR을 중지하려는 경우 다른 데이터베이스에 대한 선택란을 지우십시오.
4. 단 하나의 데이터베이스만(기본 데이터베이스 또는 대기 데이터베이스) 시작되는 경우, 해당 데이터베이스의 이름이 HADR 중지 창에 표시됩니다.
5. 확인을 누르십시오. 창이 닫힙니다. 진행 표시기가 열려서 명령이 실행 중인 시기를 표시할 수 있습니다. 완료되면 성공 여부를 표시하는 통지가 수신됩니다.

   제어 센터의 문맥 도움말 기능을 통해 추가 정보가 제공됩니다.

DB2 고가용성 재해 복구(HADR) 환경에서 데이터베이스 활성화 및 비활성화

표준 데이터베이스가 클라이언트 연결에 의해 시작된 경우 데이터베이스는 마지막 클라이언트가 연결을 끊을 때 종료됩니다. HADR 기본 데이터베이스가 클라이언트 연결에 의해 시작된 경우 이는 ACTIVATE DATABASE 명령을 사용하여 데이터베이스를 시작하는 것과 같습니다. 클라이언트 연결에 의해 시작된 HADR 기본 데이터베이스를 종료하려면 명시적으로 DEACTIVATE DATABASE 명령을 발행해야 합니다.

롤 포워드 보류 상태의 표준 데이터베이스에서, ACTIVATE DATABASE 및 DEACTIVATE DATABASE 명령은 적용 가능하지 않습니다. 단지 계속해서 롤 포

워드 또는 롤 포워드 중지하거나 START HADR을 사용하여 데이터베이스를 HADR 대기 데이터베이스로 시작할 수 있습니다. 데이터베이스가 HADR 대기로 시작되면, ACTIVATE DATABASE 및 DEACTIVATE DATABASE 명령을 사용하여 데이터베이스를 시작 및 중지할 수 있습니다.

다음 방법을 사용하여 기본 데이터베이스를 활성화하십시오.

- 클라이언트 연결
- ACTIVATE DATABASE 명령
- AS PRIMARY 옵션과 함께 START HADR 명령

다음 방법을 사용하여 기본 데이터베이스를 비활성화하십시오.

- DEACTIVATE DATABASE 명령
- FORCE 옵션과 함께 db2stop 명령

다음 방법을 사용하여 대기 데이터베이스를 활성화하십시오.

- ACTIVATE DATABASE 명령
- AS STANDBY 옵션과 함께 START HADR 명령

다음 방법을 사용하여 대기 데이터베이스를 비활성화하십시오.

- DEACTIVATE DATABASE 명령
- FORCE 옵션과 함께 db2stop 명령

DB2 고가용성 재해 복구(HADR) 환경에서 롤링 업그레이드 수행

소프트웨어 또는 하드웨어를 업그레이드하거나 DB2 데이터베이스 시스템을 갱신할 때 또는 데이터베이스 구성 매개변수를 변경할 때 고가용성 재해 복구(HADR) 환경에서 이 프로시저를 사용하십시오. 이 프로시저는 처리가 한 데이터베이스에서 다른 데이터베이스로 전환될 때 순간적인 서비스 인터럽트만을 발생시키고 업그레이드 프로세스 동안 데이터베이스 서비스를 사용 가능하게 유지합니다. HADR은 기본 및 대기 데이터베이스 둘 다 동일한 시스템에 있을 때 최적으로 수행하기 때문에 가능한 빨리 두 시스템 모두에 변경을 적용해야 합니다.

주: 모든 DB2 데이터베이스 시스템 Fixpack 및 업그레이드는 프로덕션 시스템에 적용되기 전에 테스트 환경에서 구현되어야 합니다.

롤링 업그레이드를 시작하기 전에 HADR 쌍이 피어 상태에 있어야 합니다.

이 프로시저는 DB2 데이터베이스 시스템의 이전 버전에서 나중 버전으로 업그레이드하기 위해 작동하지 않습니다. 예를 들어 이 프로시저를 사용하여 버전 8에서 버전 9 데이터베이스 시스템으로 업그레이드할 수 없습니다. 이 프로시저를 사용하여 데이터베이스 시스템의 한 수정 레벨에서 다른 레벨로만, 예를 들어 Fixpack을 적용하여 롤링 갱신을 수행할 수 있습니다.

이 프로시저는 DB2 HADR 구성 매개변수를 갱신하는 경우 작동하지 않습니다. HADR 구성 매개변수에 대한 갱신은 별도로 수행되어야 합니다. HADR은 기본 및 대기의 매개변수가 동일해야 하기 때문에, 기본 및 대기 데이터베이스가 모두 동시에 비활성화되고 갱신되어야 할 수 있습니다.

HADR 환경에서 롤링 업그레이드를 수행하려면 다음을 수행하십시오.

1. 대기 데이터베이스가 있는 시스템을 업그레이드하십시오.
   a. DEACTIVATE DATABASE 명령을 사용하여 대기 데이터베이스를 종료하십시오.
   b. 필요한 경우 대기 데이터베이스의 인스턴스를 종료하십시오.
   c. 소프트웨어, 하드웨어 또는 DB2 구성 매개변수 중 하나 이상을 변경하십시오.

      주: 롤링 업그레이드를 수행할 때 HADR 구성 매개변수를 변경할 수 없습니다.
   d. 필요한 경우 대기 데이터베이스의 인스턴스를 재시작하십시오.
   e. ACTIVATE DATABASE 명령을 사용하여 대기 데이터베이스를 재시작하십시오.
   f. 대기 데이터베이스가 피어 상태로 되는지 확인하십시오. GET SNAPSHOT 명령을 사용하여 이를 점검하십시오.
2. 기본 및 대기 데이터베이스의 역할을 전환하십시오.
   a. 대기 데이터베이스에서 TAKEOVER HADR 명령을 발행하십시오.
   b. 클라이언트를 새 기본 데이터베이스로 경로 지정하십시오. 이는 자동 클라이언트 리라우트를 사용하여 수행될 수 있습니다.

      주: 대기 데이터베이스가 기본 데이터베이스로서 인계하기 때문에 새 기본 데이터베이스가 이제 업그레이드됩니다. DB2 데이터베이스 시스템 Fixpack을 적용하려는 경우, TAKEOVER HADR 명령이 원래 기본 데이터베이스의 역할을 대기 데이터베이스로 변경합니다. 그러나 명령은 새 대기 데이터베이스가 새로 갱신된 기본 데이터베이스에 연결하도록 허용하지 않습니다. 새 대기 데이터베이스가 DB2 데이터베이스 시스템의 이전 버전을 사용하기 때문에 갱신된 기본 데이터베이스에 의해 생성되는 새 로그 레코드를 이해하지 못할 수 있으며 종료됩니다. 새 대기 데이터베이스가 새 기본 데이터베이스에 재연결하려면(즉, HADR 쌍을 재형성하려면) 새 대기 데이터베이스도 갱신되어야 합니다.
3. 위의 1단계와 동일한 프로시저를 사용하여 원래 기본 데이터베이스(이제는 대기 데이터베이스임)를 업그레이드하십시오. 이를 수행했을 때 두 데이터베이스 모두가 업그레이드되고 HADR 피어 상태에서 서로에게 연결됩니다. HADR 시스템이 전체 데이터베이스 서비스 및 전체 고가용성 보호를 제공합니다.
4. 선택사항. 원래 구성으로 리턴하려면 2단계에서와 같이 기본 및 대기 데이터베이스의 역할을 전환하십시오.

분할 미러를 사용한 데이터베이스 클론

클론 데이터베이스를 작성하려면 다음 프로시저를 사용하십시오. 클론 데이터베이스에 쓸 수 있지만, 클론 데이터베이스는 일반적으로 보고서 실행 같은 읽기 전용 활동에 사용됩니다.

클론된 데이터베이스를 백업하거나, 백업 이미지를 원래 시스템에 리스토어하거나 원래 시스템에서 생성된 로그 파일을 통해 롤 포워드할 수 없습니다. AS SNAPSHOT 옵션을 사용할 수 있지만, 이것은 입출력이 일시중단된 시간에서 데이터베이스의 순간 사본만을 제공합니다. 다른 모든 미해결 언커미트 작업은 db2inidb 명령이 클론에 대해 실행된 후에 롤백됩니다.

데이터베이스를 클론하려면 다음 단계를 수행하십시오.

1. 기본 데이터베이스의 입출력을 일시중단하십시오.

   ```
   db2 set write suspend for database
   ```

   데이터베이스가 일시중단된 상태에서 다른 유틸리티나 도구를 실행 중이 아니어야 합니다. 오직 데이터베이스의 사본을 작성 중이어야 합니다.
2. 적합한 운영 체제 레벨 명령을 사용하여 기본 데이터베이스에서 미러를 분할하십시오.

   메모: 볼륨 디렉토리를 포함한 전체 데이터베이스 디렉토리를 복사해야 합니다. 또한 로그 디렉토리 및 데이터베이스 디렉토리 밖에 존재하는 모든 컨테이너 디렉토리를 복사해야 합니다. 이 정보를 알려면 분할되어야 하는 데이터베이스의 모든 파일과 디렉토리를 표시하는 DBPATHS 관리 뷰를 참조하십시오.
3. 기본 데이터베이스에서 입출력을 재개하십시오.

   ```
   db2 set write resume for database
   ```
4. 보조 시스템에서 미러된 데이터베이스를 카탈로그하십시오.

   메모: 디폴트로 미러된 데이터베이스는 기본 데이터베이스와 동일한 시스템에 존재할 수 없습니다. 동일한 디렉토리 구조를 갖고 기본 데이터베이스와 동일한 인스턴스 이름을 사용하는 보조 시스템에 위치해야 합니다. 미러된 데이터베이스가 기본 데이터베이스와 동일한 시스템에 존재해야 하는 경우 db2relocatedb 유틸리티 또는 db2inidb 명령의 RELOCATE USING 옵션을 사용하여 이를 수행할 수 있습니다.
5. 보조 시스템에서 데이터베이스 인스턴스를 시작하십시오.

   ```
   db2start
   ```
6. 보조 시스템에서 미러된 데이터베이스를 초기화하십시오.

   ```
   db2inidb database_alias as snapshot
   ```

   필요한 경우 db2inidb 명령의 RELOCATE USING 옵션을 지정하여 클론 데이터베이스를 재배치하십시오.

```
db2inidb database_alias as snapshot relocate using relocatedbcfg.txt
```

여기서 relocatedbcfg.txt 파일에 데이터베이스를 재배치하기 위해 필요한 정보가 들어 있습니다.

참고:

a. 이 명령은 분할이 발생할 때 진행 중인 트랜잭션을 롤백하며 새 로그 체인 순서를 시작하므로 기본 데이터베이스의 모든 로그가 클론된 데이터베이스에서 재생될 수 없게 됩니다.

b. 데이터베이스 디렉토리(볼륨 디렉토리 포함), 로그 디렉토리 및 컨테이너 디렉토리가 RELOCATE USING 옵션을 사용하기 전에 원하는 위치로 이동되어야 합니다.

기본 및 대기 데이터베이스 동기

한 가지 고가용성 전략은 기본 데이터베이스가 실패할 때 기본 데이터베이스와 보조 또는 대기 데이터베이스가 조작을 인계하도록 하는 것입니다. 대기 데이터베이스가 실패한 기본 데이터베이스에 대한 데이터베이스 조작을 인계해야 하는 경우, 대기 데이터베이스는 정확하게 동일한 데이터를 포함하고 모든 인플라이트(inflight) 트랜잭션에 대해 알아야 하며, 그렇지 않으면 기본데이터베이스 서버가 실패하지 않은 경우 수행하는 것과 정확하게 동일한 방법으로 데이터베이스 처리를 계속해야 합니다. 대기 데이터베이스를 갱신하여 기본 데이터베이스의 사본이 되도록 하는 진행 중 프로세스를 동기화라고 합니다.

기본 및 대기 데이터베이스를 동기화할 수 있기 전에 다음을 수행해야 합니다.

- 기본 및 대기 데이터베이스를 작성 및 구성하십시오.
- 기본 및 대기 데이터베이스 간의 통신을 구성하십시오.
- 동기화 전략(예: 로그 제공, 로그 미러링, 일시중단된 입출력 및 디스크 미러링 또는 HADR)을 선택하십시오.

  기본 데이터베이스 서버 및 대기 데이터베이스 서버가 동기화된 상태를 유지하도록 하는 여러 가지 전략이 있습니다.

  – 기본 데이터베이스에서 대기 데이터베이스로 로그를 제공한 후 대기 데이터베이스에서 롤 포워드

  – 기본 및 대기 데이터베이스 모두에 데이터베이스 로그를 동시에 쓰기(로그 미러링이라고 함)

  – 주기적으로 기본 데이터베이스의 사본을 작성하고 미러를 분할하고 사본을 새 대기 데이터베이스 서버로 초기화하는 디스크 미러링과 함께 일시중단된 입출력 지원 사용

– DB2 고가용성 재해 복구(HADR) 기능 같은 사용 가능성 기능을 사용하여 기본 및 대기 데이터베이스를 동기화된 상태로 유지.

1. 로그를 사용하여 기본 데이터베이스와 보조 또는 대기 데이터베이스를 동기화하려는 경우, 필요한 로그 관리를 수행하도록 DB2 데이터베이스를 구성하십시오. 예를 들어 DB2 데이터베이스가 로그를 미러링하기 원하는 경우, MIRRORLOGPATH 구성 매개변수를 로그의 2차 사본이 저장되기 원하는 위치로 설정하십시오.
2. DB2 데이터베이스 일시중단 입출력 기능을 사용하여 기본 데이터베이스의 디스크 미러를 분할하려는 경우 다음을 수행해야 합니다.
   a. 기본 데이터베이스에 대한 디스크 미러링을 초기화하십시오.
   b. 기본 데이터베이스의 미러를 분할해야 할 때 『분할 미러를 대기 데이터베이스로 사용』주제의 지시사항을 따르십시오.
3. HADR 기능을 사용하여 기본 및 대기 데이터베이스 동기화를 관리하는 경우, HADR에 대해 DB2 데이터베이스를 구성하고 DB2 데이터베이스가 사용자 대신 기본 및 대기 데이터베이스를 동기화하도록 허용하십시오.

DB2 고가용성 재해 복구(HADR) 복제 작업

DB2 고가용성 재해 복구(HADR)는 데이터베이스 로그를 사용하여 기본 데이터베이스에서 대기 데이터베이스로 데이터를 복제합니다. 일부 활동으로 인해 대기 데이터베이스가 기본 데이터베이스보다 뒤떨어질 수도 있습니다. 로그가 대기 데이터베이스에 대해 재생되기 때문입니다. 일부 활동은 너무 과도하게 로그되어, 활동에서 생성되는 많은 양의 로그 파일로 인해 스토리지 문제점이 발생할 수 있습니다. 로그를 사용하여 대기 데이터베이스로 데이터를 복제하는 것이 사용 가능성 전략의 핵심이지만, 로깅 자체는 사용자 솔루션의 사용 가능성에 부정적 영향을 줄 수 있습니다. 현명하게 유지보수 전략을 설계하고, 로깅의 부정적 영향이 최소화되도록 시스템을 구성하여, 로깅으로 트랜잭션 데이터가 보호될 수 있도록 하십시오.

고가용성 재해 복구(HADR)에서, 다음 작업은 기본 데이터베이스에서 대기 데이터베이스로 복제됩니다.

- 데이터 정의 언어(DDL)
- 데이터 처리 언어(DML)
- 버퍼 풀 작업
- 테이블 스페이스 작업
- 온라인 재구성
- 오프라인 재구성
- 스토어드 프로시저 및 UDF(User Defined Function)의 메타데이터(관련 오브젝트 또는 라이브러리 파일은 아님)

온라인 재구성 중에는 모든 작업이 자세히 로그됩니다. 결과적으로, HADR은 더 일반적인 데이터베이스 갱신사항의 경우보다 대기 데이터베이스가 뒤떨어지지 않고도 작업을 복제할 수 있습니다. 그러나 이 동작은 많은 로그 레코드가 생성되므로 시스템에 많은 영향을 줄 수 있습니다.

오프라인 재구성이 온라인 재구성만큼 광범위하게 로그되지 않지만, 작업은 보통 영향받는 수백 또는 수천 개의 행마다 로그됩니다. 이는 대기 데이터베이스가 로그 레코드마다 기다린 후 한 번에 많은 갱신사항을 재생하므로 뒤떨어질 수 있음을 의미합니다. 오프라인 재구성이 클러스터되지 않는 경우, 전체 재구성 작업 후에 단일 로그 레코드가 생성됩니다. 이 모드는 기본 데이터베이스에 뒤떨어지지 않도록 하는 대기 데이터베이스의 기능에 가장 많은 영향을 미칩니다. 대기 데이터베이스는 기본 데이터베이스에서 로그 레코드를 수신한 후 전체 재구성을 수행합니다.

HADR은 스토어드 프로시저 및 UDF 오브젝트와 라이브러리 파일은 복제하지 않습니다. 기본 및 대기 데이터베이스 둘 다에서 동일한 경로에 파일을 작성해야 합니다. 대기 데이터베이스가 참조된 오브젝트 또는 라이브러리 파일을 찾을 수 없으면 스토어드 프로시저 또는 UDF 호출이 대기 데이터베이스에서 실패합니다.

DB2 고가용성 재해 복구(HADR) 비복제 작업

DB2 고가용성 재해 복구(HADR)는 데이터베이스 로그를 사용하여 기본 데이터베이스에서 대기 데이터베이스로 데이터를 복제합니다. 로그되지 않는 작업은 기본 데이터베이스에서 허용되지만 대기 데이터베이스로 경로 재지정되지 않습니다. 로그되지 않은 작업(예: 실행기록 파일 갱신사항)이 대기 데이터베이스에 반영되도록 하려면 추가 단계를 수행해야 합니다.

다음은 기본 데이터베이스에 있는 작업이 대기 데이터베이스에 복제되지 않는 경우의 예입니다.

- NOT LOGGED INITIALLY 옵션을 지정하여 작성한 테이블은 복제되지 않습니다. HADR 대기 데이터베이스가 기본 데이터베이스로 설정된 후 이러한 테이블에 액세스하려고 하면 오류가 발생합니다.
- 1GB보다 큰 BLOB 및 CLOB는 로그할 수 없으므로 복제할 수 없습니다. 로그되지 않은 BLOB 및 CLOB는 복제되지 않습니다. 그러나 이 BLOB 및 CLOB에 대한 스페이스는 대기 데이터베이스에서 할당됩니다. LOB 컬럼의 데이터는 2진 영(0)이 됩니다. 로그된 모든 BLOB 및 CLOB는 복제됩니다.
- UPDATE DATABASE CONFIGURATION 및 UPDATE DATABASE MANAGER CONFIGURATION 명령을 사용한 데이터베이스 구성 갱신사항은 복제되지 않습니다.
- 데이터베이스 구성 및 데이터베이스 관리 프로그램 구성 매개변수는 복제되지 않습니다.

- 사용자 정의 함수(UDF)의 경우 데이터베이스 외부 오브젝트(예: 관련 오브젝트 및 라이브러리 파일) 변경사항은 복제되지 않습니다. 이 변경사항은 다른 방법으로 대기에서 설정해야 합니다.
- 복구 실행기록 파일(db2rhist.asc)과, 이 파일의 변경사항은 기본 데이터베이스에서 대기 데이터베이스로 자동 제공되지 않습니다.

  REPLACE HISTORY FILE 옵션과 함께 RESTORE DATABASE 명령을 발행하여 실행기록 파일의 초기 사본(기본의 백업 이미지에서 얻은)을 대기 데이터베이스에 놓을 수 있습니다.

  ```
  RESTORE DB KELLY REPLACE HISTORY FILE
  ```

  HADR이 초기화되고 기본 데이터베이스에서 연속 백업 활동이 발생하는 경우 대기 데이터베이스의 실행기록 파일은 최신의 파일이 아닙니다. 그러나 실행기록 파일 사본은 각 백업 이미지에서 저장됩니다. 다음 명령을 사용하여 백업 이미지로부터 실행기록 파일을 추출하여 대기 데이터베이스의 실행기록 파일을 갱신할 수 있습니다.

  ```
  RESTORE DB KELLY HISTORY FILE
  ```

  데이터베이스 디렉토리의 실행기록 파일을 기본 데이터베이스에서 대기 데이터베이스로 복사하기 위해 일반 운영 체제 명령을 사용하면 안됩니다. 복사할 때 기본 데이터베이스가 파일을 갱신하는 경우 실행기록 파일이 손상될 수 있습니다.

  인계 조작이 발생하고 대기 데이터베이스에 최근의 실행기록 파일이 있는 경우, 새 기본 데이터베이스에 대한 백업 및 리스토어 작업은 실행기록 파일에서 새 레코드를 생성하고 원래 기본 데이터베이스에서 생성된 레코드와 균일하게 혼합합니다. 실행기록 파일이 최신의 파일이 아니거나 항목이 누락된 경우, 자동 증분 리스토어가 불가능할 수 있습니다. 대신 수동 증분 리스토어 작업이 필요합니다.

DB2 고가용성 재해 복구(HADR) 대기 데이터베이스 상태

언제든지, 대기 데이터베이스 상태는 5가지 상태 중 하나입니다. 5가지 상태는 로컬 만회, 리모트 만회 보류, 리모트 만회, 피어 및 연결이 끊어진 피어입니다. 대기 데이터베이스의 상태에 따라 수행할 수 있는 작업이 판별됩니다. GET SNAPSHOT 명령을 사용하여 대기 데이터베이스의 상태를 볼 수 있습니다.

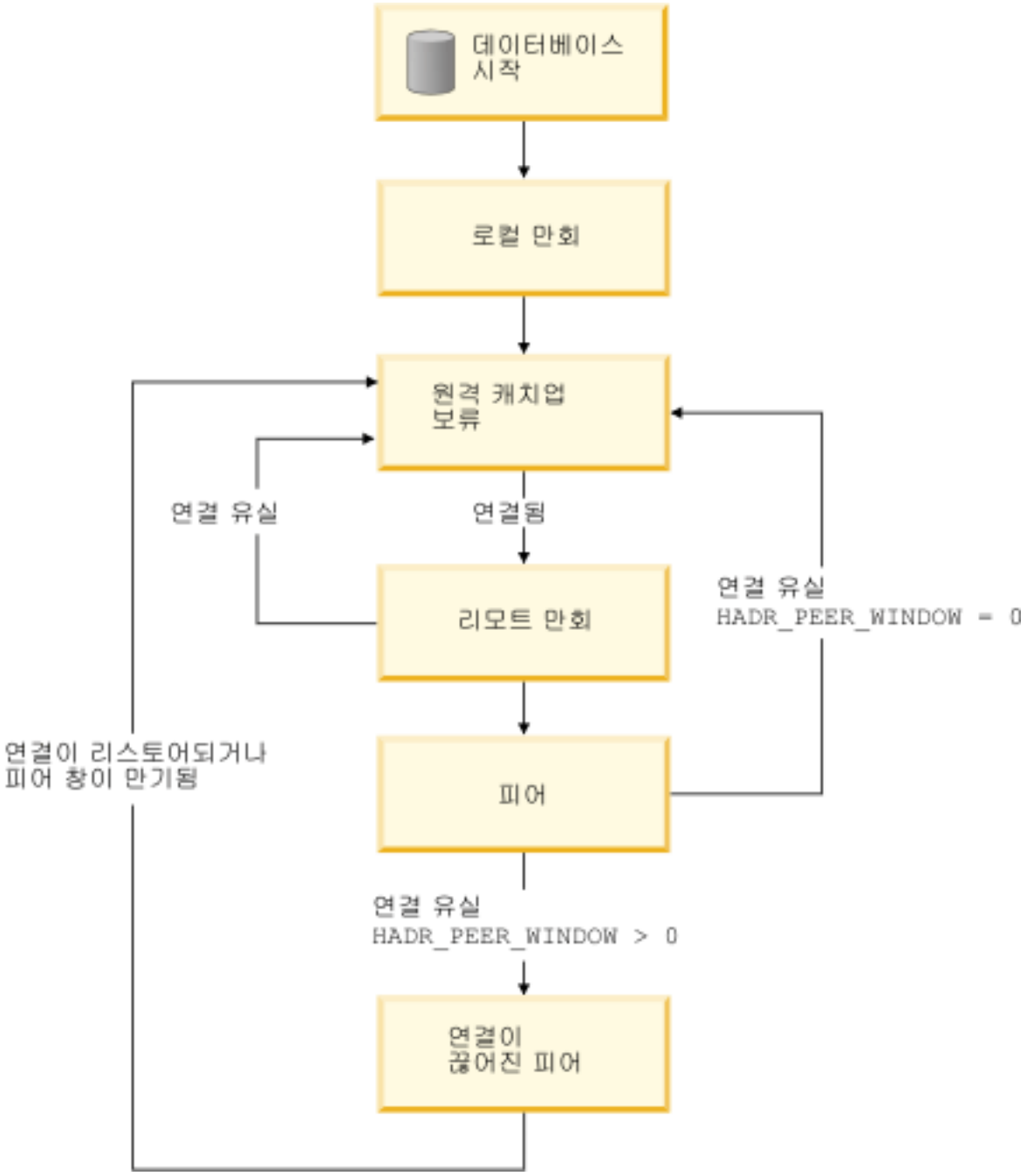


그림 9. 대기 데이터베이스 상태

데이터베이스 시작, 로컬 만회 및 리모트 만회 보류

고가용성 재해 복구(HADR) 기능을 사용할 경우, 대기 데이터베이스가 시작되면 로컬 만회 상태가 되고 로컬 로그 경로에서 로그 파일을 읽으려고 합니다. 로컬 로그 경로에서 로그 파일을 찾지 못했는데 로그 아카이브 방법은 지정된 경우, 지정된 방법을 사용하여 로그 파일을 검색합니다. 로그 파일을 읽으면, 대기 데이터베이스에서 재생됩니다. 이 시간 동안 기본 데이터베이스와의 연결은 필요하지 않습니다. 그러나 연결이 존재하

지 않으면 대기 데이터베이스가 기본 데이터베이스에 연결하려고 합니다. 로컬 로그 파일의 끝에 도달하면 대기 데이터베이스는 리모트 캐치업 보류 상태에 진입합니다.

대기 데이터베이스가 리모트 만회 보류 상태가 된 후 로컬 로그 파일이 사용 가능하게 되면, 대기 데이터베이스를 종료하고 재시작하여 다시 로컬 만회 상태가 되도록 할 수 있습니다. 대기 데이터베이스에서 이와 같은 로그 파일에 로컬 액세스하는 것이, HADR에서 네트워크를 통해 기본 데이터베이스에서 파일을 복사할 수 있도록 하는 것보다 더 효율적인 경우 이와 같이 수행할 수 있습니다.

리모트 만회 보류, 리모트 만회, 피어

대기 데이터베이스는 기본 데이터베이스와의 연결이 설정될 때까지(이때 대기 데이터베이스는 리모트 만회 상태가 됨) 리모트 만회 보류 상태로 유지됩니다. 이 시간 동안 기본 데이터베이스는 해당 로그 경로에서, 또는 로그 아카이브 방법으로 로그 데이터를 읽고 로그 파일을 대기 데이터베이스로 보냅니다. 대기 데이터베이스는 로그 데이터를 수신하고 재생합니다. 기본 및 대기 데이터베이스는 대기 데이터베이스가 기본 데이터베이스 머신의 디스크에 있는 모든 로그 파일을 수신할 때 피어 상태가 됩니다.

피어 상태에 있을 때, 로그 페이지는 기본 데이터베이스가 로그 페이지를 디스크로 플러시할 때마다 대기 데이터베이스로 보냅니다. 로그 페이지는 기본 및 대기 데이터베이스가 동일한 로그 파일 시퀀스를 갖도록 대기 데이터베이스의 로컬 로그 파일에 기록됩니다. 그런 다음 로그 페이지는 대기 데이터베이스에서 재생될 수 있습니다.

데이터베이스가 리모트 만회 상태에 있을 때 및 기본 대기 데이터베이스 사이의 연결이 끊어지면 대기 데이터베이스는 리모트 만회 보류 상태가 됩니다. 데이터베이스가 피어 상태에 있을 때 기본 및 대기 데이터베이스 사이의 연결이 끊어지고 HADR_PEER_WINDOW 데이터베이스 구성 매개변수가 설정되지 않은 경우(또는 0으로 설정된 경우) 대기 데이터베이스는 리모트 만회 보류 상태가 됩니다. 그러나 데이터베이스가 피어 상태에 있을 때 기본 및 대기 데이터베이스 사이의 연결이 끊어지고 HADR_PEER_WINDOW 데이터베이스 구성 매개변수가 0이 아닌 값으로 설정된 경우 대기 데이터베이스는 연결이 끊어진 피어 상태가 됩니다.

연결이 끊어진 피어

HADR_PEER_WINDOW 데이터베이스 구성 매개변수를 0보다 큰 시간 값으로 구성한 경우, 기본 데이터베이스에서 대기 데이터베이스와의 연결이 끊어지면 기본 데이터베이스는 기본 및 대기 데이터베이스가 구성된 시간 동안 피어 상태에 있는 것처럼 계속 작동합니다. 기본 데이터베이스 및 대기 데이터베이스의 연결이 끊어졌지만 피어 상태처럼 작동하는 경우 이 상태를 연결이 끊어진 피어라고 합니다. 기본 데이터베이스가 대기 데이터베이스와의 연결이 끊어진 후 연결 끊어진 피어 상태로 유지되는 시간을 피어 창이라고 합니다. 대기 데이터베이스와의 연결이 리스토어되거나 피어 창이 만기되면 대기 데이터베이스는 연결이 끊어진 피어 상태에서 벗어납니다.

피어 창 구성의 장점은 여러 실패 또는 연쇄 실패 중에 트랜잭션이 손실되는 위험이 낮아진다는 것입니다. 피어 창이 없는 경우, 기본 데이터베이스에서 대기 데이터베이스와의 연결이 끊어지면 기본 데이터베이스는 피어 상태 밖으로 이동합니다. 기본 데이터베이스의 연결이 끊어진 경우 기본 데이터베이스는 대기 데이터베이스와 독립적으로 트랜잭션을 처리합니다. 이와 같이 피어 상태에 있지 않을 때 기본 데이터베이스에서 실패가 발생하면 트랜잭션이 대기 데이터베이스에서 복제되지 않았을 수 있으므로 트랜잭션이 손실될 수 있습니다. 피어 창이 구성되면, 기본 데이터베이스는 로그가 대기 데이터베이스의 기본 메모리에 기록되거나 로그가 대기 데이터베이스의 로그 파일에 기록되었다는(HADR 동기화 모드에 따라) 확인을 대기 데이터베이스에서 수신할 때까지 트랜잭션이 커미트된 것으로 간주하지 않습니다.

피어 창 구성의 단점은 기본 데이터베이스의 트랜잭션이 기본 데이터베이스가 피어 창에서 대기 데이터베이스와의 연결이 리스토어되거나 피어 창이 만기되기를 기다리는 동안 시간종료가 발생하거나 많은 시간이 소비되는 것입니다.

GET SNAPSHOT 명령이나 -hadr 매개변수가 있는 db2pd 유틸리티를 사용하여 HADR_PEER_WINDOW 데이터베이스 구성 매개변수의 값인 피어 창 *크기*를 판별할 수 있습니다.

기본 및 대기 데이터베이스 동기화를 위한 대기 데이터베이스 상태의 내포사항 및 제한사항

기본 및 대기 데이터베이스를 동기화하는 한 가지 방법은 로컬 만회에 사용되도록 기본 데이터베이스 로그 파일을 대기 데이터베이스 로그 경로로 직접 복사하는 것입니다. 기본 및 대기 데이터베이스를 기본 데이터베이스 로그에서 대기 데이터베이스 로그 경로로 직접 복사하여 동기화하는 경우 다음과 같은 이유로 대기 데이터베이스를 시작하기 전에 1차 로그 파일을 복사해야 합니다.

1. 로컬 로그 파일 끝에 도달하면 대기 데이터베이스는 리모트 만회 보류 상태가 되고 대기 데이터베이스가 재시작할 때까지 다시 로컬 로그 파일에 액세스하려고 하지 않습니다.
2. 대기 데이터베이스가 리모트 만회 상태가 되면, 로그 파일을 해당되는 로그 경로로 복사하는 것이 대기 데이터베이스에 의한 로컬 로그 파일 *쓰기*에 방해가 될 수 있습니다.

GET SNAPSHOT 명령을 사용하여 HADR 대기 데이터베이스 상태 판별

DATABASE ON 옵션과 함께 GET SNAPSHOT 명령을 발행하여 DB2 고가용성 재해 복구(HADR) 대기 데이터베이스의 상태를 판별할 수 있습니다.

기본-대기 HADR 데이터베이스 쌍에서 HADR 대기 데이터베이스의 상태를 판별하기 위해 기본 데이터베이스 또는 대기 데이터베이스에서 GET SNAPSHOT 명령을 발행할 수 있습니다.

- 대기 데이터베이스에서 GET SNAPSHOT 명령을 발행하는 경우, 대기 데이터베이스의 상태가 출력의 State 필드에 리턴됩니다.
- 대기 데이터베이스에 연결되는 기본 데이터베이스에서 GET SNAPSHOT 명령을 발행하는 경우, 대기 데이터베이스의 상태는 출력의 State 필드에서 리턴됩니다.
- 대기 데이터베이스에 연결되지 않는 기본 데이터베이스에서 GET SNAPSHOT 명령을 발행하는 경우, disconnected가 출력의 State 필드에 리턴됩니다.

예를 들어 대기 데이터베이스 MUSIC이 있으면 다음 명령을 발행하여 상태를 확인할 수 있습니다.

```
get snapshot for database on music
```

다음 출력은 GET SNAPSHOT 명령에 의해 리턴되는 HADR 상태 섹션을 표시합니다.

```
HADR status

  Role                    = Primary
  State                   = Peer
  Synchronization mode    = Sync
  Connection status       = Connected, 11-03-2002 12:23:09.35092
  Heartbeat missed        = 0
  Local host              = host1.ibm.com
  Local service           = hadr_service
  Remote host             = host2.ibm.com
  Remote service          = hadr_service
  Remote instance         = dbinst2
  timeout(seconds)        = 120
  Primary log position(file, page, LSN) = S0001234.LOG, 12, 0000000000BB800C
  Standby log position(file, page, LSN) = S0001234.LOG, 12, 0000000000BB800C
  Log gap running average(bytes) = 8723
```

GET SNAPSHOT 명령의 출력을 검토할 때 로그 갭을 주목할 수 있습니다. 로그 갭은 로그 파일이 명시적 로그 절단의 결과로서 또는 기본 데이터베이스 중지 및 재시작의 결과로서 절단될 때 발생할 수 있으며, 기본이 다음 로그 파일의 시작으로 이동합니다. 그러나 대기는 마지막 로그 파일의 끝에 남아있습니다. 기본이 로그를 쓰자마자, 로그가 복제되고 대기가 로그 위치를 갱신합니다.

DB2 고가용성 재해 복구(HADR) 관리

DB2 고가용성 재해 복구(HADR) 관리에는 HADR 시스템의 상태 구성 및 유지보수가 포함됩니다.

HADR 관리에는 다음과 같은 태스크가 포함됩니다.

- 35 페이지의 『고가용성 재해 복구(HADR) 초기화』
- 169 페이지의 『DB2 고가용성 재해 복구(HADR) 중지』
- 195 페이지의 『고가용성 재해 복구(HADR)에서 데이터베이스 역할 전환』
- 192 페이지의 『HADR 장애 복구 조작 수행』
- 188 페이지의 『고가용성 재해 복구(HADR) 모니터링』

- HADR에 관련된 데이터베이스 구성 매개변수 검사 및 변경
- HADR 데이터베이스 카탈로깅(필요한 경우)

다음 방법을 사용하여 HADR을 관리할 수 있습니다.

- 명령행 처리기
- 제어 센터 GUI 도구
- DB2 관리 API

DB2 고가용성 재해 복구(HADR) 명령

DB2 고가용성 재해 복구(HADR) 기능은 DB2 고가용성 데이터베이스 솔루션에 대한 복합적 로깅, 장애 복구 및 복구 기능을 제공합니다. HADR이 제공하는 기능의 복잡도에도 불구하고, 수행할 명령 HADR에 직접적으로 필요한 조치는 단 몇 가지입니다(HADR 시작, HADR 중지, 대기 데이터베이스에 의한 기본 데이터베이스 인계).

HADR을 관리하기 위해 사용되는 고가용성 재해 복구(HADR) 명령은 세 가지입니다.

- HADR 시작
- HADR 중지
- HADR 인계

이 명령을 호출하려면 명령행 처리기나 관리 API를 사용하십시오. 제어 센터의 고가용성 재해 복구 관리 창에서 사용 가능한 GUI를 사용하여 이 명령을 호출할 수도 있습니다. 제어 센터에서 고가용성 재해 복구 관리 창을 열려면 데이터베이스를 마우스 오른쪽 단추로 누르고 고가용성 재해 복구-—>관리를 누르십시오.

AS PRIMARY 또는 AS STANDBY 옵션과 함께 START HADR 명령을 발행하면 데이터베이스 역할이 지정된 역할로 변경됩니다(데이터베이스가 아직 해당 역할에 있지 않은 경우). 이 명령은 또한 데이터베이스를 활성화합니다(아직 활성화되지 않은 경우).

STOP HADR 명령은 HADR 데이터베이스(기본 또는 대기)를 표준 데이터베이스로 변경합니다. HADR에 관련되는 데이터베이스 구성 매개변수는 데이터베이스가 HADR 데이터베이스로서 쉽게 다시 활성화될 수 있도록 변경되지 않은 상태로 유지됩니다.

대기 데이터베이스에서만 발행할 수 있는 TAKEOVER HADR 명령은 대기 데이터베이스를 기본 데이터베이스로 변경합니다. BY FORCE 옵션을 지정하지 않는 경우 기본 및 대기 데이터베이스는 역할을 전환합니다. BY FORCE 옵션을 지정하는 경우 대기 데이터베이스는 일방적으로 기본 데이터베이스가 되도록 전환됩니다. 이 경우, 대기 데이터베이스는 이전 기본 데이터베이스에서 처리 중인 트랜잭션을 중지하려고 합니다. 그러나 트랜잭션 처리가 중지된다는 보장은 없습니다. 장애 복구 조건의 경우에만 BY FORCE 옵션을 사용하여 강제로 인계 작업을 실행하십시오. 가능하면, BY FORCE 옵션과 함께 TAKEOVER HADR 명령을 발행하기 전에 현재 기본 데이터베이스가 명확히 실패했는지 확인하거나 사용자 스스로 종료하십시오.

HADR 데이터베이스 역할 전환

데이터베이스는 기본 및 표준 역할 사이에서 동적으로, 반복적으로 전환될 수 있습니다. 데이터베이스가 온라인 또는 오프라인인 경우 AS PRIMARY 옵션을 사용한 START HADR 명령과 STOP HADR 명령을 발행할 수 있습니다.

대기 및 표준 역할 사이에 정적으로 데이터베이스를 전환할 수 있습니다. 데이터베이스가 롤 포워드 보류 상태에서 유지되는 경우에만 반복적으로 이와 같이 수행할 수 있습니다. AS STANDBY 옵션과 함께 START HADR 명령을 발행하여 데이터베이스가 오프라인이고 롤 포워드 보류 상태에 있는 동안 표준 데이터베이스를 대기로 변경할 수 있습니다. 데이터베이스가 오프라인일 때 대기 데이터베이스를 표준 데이터베이스로 변경하려면 STOP HADR 명령을 사용하십시오. 데이터베이스는 STOP HADR 명령을 발행한 후 롤 포워드 보류 상태에서 유지됩니다. AS STANDBY 옵션과 함께 연속 START HADR 명령을 발행하면 데이터베이스가 대기로 리턴됩니다. 대기 데이터베이스에서 HADR을 중지한 후 STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하면 다시 대기로 가져올 수 없습니다. 데이터베이스는 롤 포워드 보류 상태에서 벗어났으므로 표준 데이터베이스로 사용할 수 있습니다. 이를 대기 데이터베이스 스냅샷 가져오기라고 합니다. 기존 대기 데이터베이스를 표준 데이터베이스로 변경하면 고가용성 목적을 위해 새 대기 데이터베이스를 작성할 것을 고려하십시오.

기본 및 대기 데이터베이스의 역할을 전환하려면 BY FORCE 옵션을 사용하지 않고 인계 작업을 수행하십시오.

대기를 일방적으로 기본으로 변경하려면(기본을 대기로 변경하지 않고) 강제 인계를 사용하십시오. 그러면 이전 기본을 새 대기로 다시 통합할 수 있습니다.

HADR 역할은 지속적입니다. HADR 역할이 설정되면, DB2 인스턴스의 반복되는 중지 및 재시작이나 DB2 데이터베이스의 비활성화 및 활성화에도 데이터베이스에 대해 유지됩니다.

대기 시작은 비동기입니다.

AS STANDBY 옵션과 함께 START HADR 명령을 발행할 때 명령은 관련된 EDU(engine dispatchable unit)가 시작되면 즉시 리턴합니다. 명령은 대기 데이터베이스가 기본 데이터베이스에 연결하는 것을 기다리지 않습니다. 반대로, 기본 데이터베이스는 대기 데이터베이스에 연결할 때까지 시작된 것으로 간주되지 않습니다(기본 데이터베이스에서 BY FORCE 옵션과 함께 START HADR 명령이 발행되는 경우를 제외하고). 대기 데이터베이스가 오류를 발견하면(예: 기본 데이터베이스의 연결 거부) AS STANDBY 옵션을 사용하는 START HADR 명령이 이미 리턴되었을 수 있습니다. 결과적으로, HADR이 오류 표시를 리턴할 수 있는 사용자 프롬프트는 없습니다. HADR 대기는 메시지를 DB2 진단 로그에 기록하고 스스로 종료합니다. HADR 기본 데이터베이스와 연결되는지 확인하기 위해 HADR 대기의 상태를 모니터해야 합니다.

재생 오류(로그 레코드를 재생하는 동안 대기 데이터베이스가 발견하는 오류)도 대기 데이터베이스를 중단시킬 수 있습니다. 이와 같은 오류는 예를 들어, 버퍼 풀을 작성하기에 충분한 메모리가 없거나 테이블 스페이스를 작성하는 동안 경로를 찾을 수 없는 경우에 발생할 수 있습니다. 대기 데이터베이스의 상태는 연속적으로 모니터해야 합니다.

클라이언트 리라우트에 대해 사용 가능한 데이터베이스 별명을 사용하여 클라이언트에서 HADR 명령을 실행하면 안됩니다.

자동 클라이언트 리라우트가 설정된 경우, 데이터베이스 서버는 클라이언트 응용프로그램이 최소한의 작업 인터럽트만으로 원래의 데이터베이스 서버나 대체 서버에 대한 작업 사이에 전환할 수 있도록 사전 정의된 대체 서버를 가지고 있습니다. 이와 같은 환경에서, 클라이언트가 TCP를 통해 데이터베이스에 연결할 때 실제 연결은 원래의 데이터베이스나 대체 데이터베이스로 이동할 수 있습니다. HADR 명령은 일반 클라이언트 연결 논리를 통해 목표 데이터베이스를 식별하기 위해 구현됩니다. 결국, 목표 데이터베이스에 정의된 대체 데이터베이스가 있는 경우 명령이 실제로 작동하는 데이터베이스를 판별하는 것은 어렵습니다. SQL 클라이언트는 자신이 연결하는 데이터베이스를 알 필요는 없지만, 특정 데이터베이스에서 HADR 명령을 적용해야 합니다. 이 제한사항을 조정하기 위해, 클라이언트 리라우트가 생략되도록(클라이언트 리라우트는 TCP/IP 연결에만 영향을 줌) 서버 머신에서 로컬로 HADR 명령을 발행해야 합니다.

HADR 명령은 유효한 라이센스를 사용하여 서버에서 실행해야 합니다.

START HADR, STOP HADR 및 TAKEOVER HADR 명령을 사용하려면 명령이 실행되는 서버에 유효한 HADR 라이센스가 설치되어 있어야 합니다. 라이센스가 없으면 이 명령은 실패하고 명령 특정 오류 코드(각각 SQL1767N, SQL1769N 또는 SQL1770N)를 98 이유 코드와 함께 리턴합니다. 문제점을 정정하려면 db2licm을 사용하여 유효한 HADR 라이센스를 설치하거나 유효한 HADR 라이센스가 분산의 일부로 포함되는 서버의 버전을 설치하십시오.

제 6 장 고가용성 솔루션에서 시스템 가동 중단 감지 및 응답

고가용성 솔루션 구현이 하드웨어 또는 소프트웨어 실패를 막지는 않습니다. 그러나 중복 시스템과 장애 복구 메커니즘이 있으므로 솔루션은 실패를 감지하고 반응하며 워크로드를 리라우트하여 사용자 응용프로그램이 계속 작업할 수 있도록 합니다.

실패가 발생할 때 데이터베이스 솔루션은 다음을 수행해야 합니다.

1. 실패를 감지합니다.

   장애 복구 소프트웨어가 하트비트 모니터링을 사용하여 시스템 구성요소의 사용 가능성을 확인할 수 있습니다. 하트비트 모니터는 시스템의 모든 구성요소로부터 정기적인 통신을 대기(listen)합니다. 하트비트 모니터가 구성요소에서 청취를 중지하는 경우 하트비트 모니터는 시스템에게 구성요소가 실패했다고 알립니다.

2. 실패에 응답합니다(장애 복구).
   a. 실패한 구성요소에 대한 조작을 인계할 보조 구성요소를 식별하고 온라인으로 만든 후 초기화합니다.
   b. 워크로드를 보조 구성요소로 리라우트합니다.
   c. 시스템에서 실패한 구성요소를 제거합니다.

3. 실패로부터 복구합니다.

   기본 데이터베이스 서버가 실패하면 첫 번째 우선순위는 클라이언트를 대체 서버로 경로 재지정하거나 대기 데이터베이스로 장애 복구하여 클라이언트 응용프로그램이 가능한 인터럽트를 최소화하고 작업을 계속할 수 있도록 하는 것입니다. 장애 복구가 성공한 후에는 실패한 데이터베이스 서버에서 잘못된 사항을 수리하여 해당 구성요소를 다시 솔루션에 재통합할 수 있도록 해야 합니다. 실패한 데이터베이스 서버 수리는 단순히 재시작을 의미할 수도 있습니다.

4. 정상 조작으로 리턴합니다.

   실패한 데이터베이스 시스템이 수리되면 다시 데이터베이스 솔루션에 통합해야 합니다. 실패한 기본 데이터베이스를 실패가 발생했을 때 기본 데이터베이스로서 인계한 데이터베이스에 대한 대기 데이터베이스로서 재통합할 수 있습니다. 또한 수리된 데이터베이스 서버가 다시 기본 데이터베이스 서버로서 인계하도록 강제할 수도 있습니다.

DB2 데이터베이스가 이러한 단계의 일부를 사용자 대신 수행할 수 있습니다. 예를 들어, 다음과 같습니다.

- DB2 고가용성 재해 복구(HADR) 하트비트 모니터 요소인 hadr_heartbeat가 기본 데이터베이스가 실패했을 때를 감지할 수 있습니다.

- DB2 클라이언트 리라우트가 실패한 데이터베이스 서버에서 보조 데이터베이스 서버로 워크로드를 전송할 수 있습니다.
- DB2 결함 모니터가 예상치 않게 종료하는 데이터베이스 인스턴스를 재시작할 수 있습니다.

관리 통지 로그

관리 통지 로그(*instance_name*.nfy)는 다양한 데이터베이스 관리 및 유지보수 활동에 대한 정보를 얻을 수 있는 저장소입니다. 데이터베이스 관리자는 이 정보를 사용하여 문제점을 진단하거나, 데이터베이스를 조정하거나, 데이터베이스 모니터를 단순하게 만들 수 있습니다.

DB2 데이터베이스 관리 프로그램은 UNIX and Linux 운영 체제 플랫폼의 관리 통지 로그에 다음과 같은 종류의 정보를 기록합니다(Windows 운영 체제 플랫폼에서는 관리 통지 이벤트를 기록하기 위해 이벤트 로그가 사용됩니다).

- DB2 유틸리티(예: REORG 및 BACKUP)의 상태
- 클라이언트 응용프로그램 오류
- 서비스 클래스 변경사항
- 라이센스 부여 활동
- 파일 경로
- 스토리지 문제점
- 모니터링 활동
- 인덱싱 활동
- 테이블 스페이스 문제점

관리 통지 로그 메시지는 표준화된 메시지 형식을 사용하는 db2diag 로그 파일에도 로그됩니다.

통지 메시지는 제공되는 SQLCODE를 보충하기 위한 추가 정보를 제공합니다.

관리 통지 로그 파일은 다른 두 가지 양식으로 존재할 수 있습니다.

단일 관리 통지 로그 파일

크기가 무한정 증가하는 하나의 활성 관리 통지 로그 파일 *instance_name*.nfy. 이 파일은 디폴트 폼이며 **diagsize** 데이터베이스 관리 프로그램 구성 매개변수의 값이 0(이 매개변수의 디폴트값은 0임)일 때마다 존재합니다.

회전하는 관리 통지 로그 파일

사용 중인 단일 로그 파일(*instance_name*.*N*.nfy. 여기서 *N*은 0에서 시작하여 계속 증가하는 번호인 파일 이름 인덱스임). **diagpath** 구성 매개변수에 정의된 위치에서 일련의 관리 통지 로그 파일을 찾을 수 있지만, 각 로그 파일은 제한

된 *크기*에 도달할 때까지(이때 로그 파일이 닫히고 증가된 파일 이름 인덱스 (*instance_name.N+1*.nfy)를 사용하여 로깅을 위해 새 로그 파일이 작성되어 열림) 증가합니다. 이 파일은 **diagsize** 데이터베이스 관리 프로그램 구성 매개변수가 0이 아닌 값을 가지고 있을 때마다 존재합니다.

주: 단일 또는 순환 관리 통지 로그 파일은 Windows 운영 체제 플랫폼에서 사용할 수 없습니다.

diagsize 데이터베이스 관리 프로그램 구성 매개변수를 적절하게 설정하여 시스템에 존재하는 이 두 가지 양식 중에서 하나를 선택할 수 있습니다.

구성

관리 통지 로그 파일은 다음의 데이터베이스 관리 프로그램 구성 매개변수를 설정하여 기록된 세부사항 레벨 및 이벤트 유형과, *크기*, 위치로 구성될 수 있습니다.

diagsize

diagsize의 값은 채택될 관리 통지 로그 파일 양식을 결정합니다. 값이 0인 경우 단일 관리 통지 로그 파일이 채택됩니다. 값이 0이 아니면 회전하는 관리 통지 로그 파일이 채택되며, 이 0이 아닌 값은 회전하는 모든 진단 로그 파일과 회전하는 모든 관리 통지 로그 파일의 총 *크기*도 지정합니다. **diagsize** 매개변수의 새 값이 적용되도록 하려면 인스턴스를 재시작해야 합니다. 자세한 세부사항은 ″diagsize - 진단 로그 파일 *크기* 구성 매개변수″ 주제를 참조하십시오.

diagpath

진단 정보는 **diagpath** 구성 매개변수에 정의된 위치에 있는 관리 통지 로그 파일에 기록되도록 지정할 수 있습니다. 자세한 세부사항은 ″diagsize - 진단 데이터 디렉토리 경로 구성 매개변수″ 주제를 참조하십시오.

notifylevel

관리 통지 로그 파일에 기록되는 세부사항의 레벨 및 이벤트 유형은 **notifylevel** 구성 매개변수를 사용하여 지정할 수 있습니다. 자세한 세부사항은 ″diagsize - 통지 레벨 구성 매개변수″ 주제를 참조하십시오.

계획되지 않은 가동 중단 감지

구성요소 실패에 응답할 수 있으려면 먼저 구성요소가 실패했음을 감지해야 합니다. DB2 Data Server에는 데이터베이스의 성능 상태를 모니터링하거나 데이터베이스가 실패했음을 감지하는 여러 가지 도구가 있습니다. 이러한 도구가 실패를 감지할 때 사용자에게 통지하거나 사전 정의된 조치를 취하도록 도구를 구성할 수 있습니다.

다음 도구를 사용하여 DB2 데이터베이스 솔루션의 일부 파트에서 실패가 발생했을 때를 감지할 수 있습니다.

DB2 결함 모니터 기능

DB2 결함 모니터 기능은 DB2 데이터베이스 인스턴스를 가동하고 실행 중 상태로 유지합니다. DB2 결함 모니터가 접속되는 DB2 데이터베이스 인스턴스가 예상치 않게 종료할 때 DB2 결함 모니터가해당 인스턴스를 재시작합니다. 데이터베이스 솔루션이 클러스터에서 구현되는 경우, DB2 결함 모니터 대신 실패한 데이터베이스 인스턴스를 재시작하도록 클러스터 관리 소프트웨어를 구성해야 합니다.

클러스터 환경에서 하트비트 모니터링

클러스터 관리 소프트웨어는 클러스터의 노드 사이의 하트비트 메시지를 사용하여 노드의 성능 상태를 모니터합니다. 클러스터 관리 프로그램은 노드가 응답 또는 메시지 전송을 중지할 때 노드가 실패했음을 감지합니다.

DB2 고가용성 재해 복구(HADR) 데이터베이스 모니터링

HADR 기능은 고유한 하트비트 모니터를 갖고 있습니다. 기본 데이터베이스와 대기 데이터베이스가 각각 정기적으로 상대로부터의 하트비트 메시지를 예상합니다.

고가용성 재해 복구(HADR) 모니터링

다음 방법을 사용하여 HADR 데이터베이스의 상태를 모니터할 수 있습니다.

db2pd 유틸리티

이 유틸리티는 DB2 메모리 세트에서 정보를 검색합니다. 예를 들어, 데이터베이스 MYDB에 대한 고가용성 재해 복구에 대한 정보를 보려면 다음 명령을 발행하십시오.

```
db2pd -db mydb -hadr
```

GET SNAPSHOT FOR DATABASE 명령

이 명령은 상태 정보를 수집하고 출력을 형식화합니다. 리턴된 정보는 명령이 발행된 시간에서의 데이터베이스 관리 프로그램 작동 상태 스냅샷을 나타냅니다. HADR 정보는 *HADR 상태* 표제 바로 아래에 있는 출력에 표시됩니다.

db2GetSnapshot API

이 API는 데이터베이스 관리 프로그램 모니터 정보를 수집하여 이를 사용자 할당 데이터 버퍼에 리턴합니다. 리턴된 정보는 API가 호출된 시점에서의 데이터베이스 관리 프로그램 작동 상태 스냅샷을 나타냅니다.

HADR 구성 매개변수는 동적 매개변수가 아닙니다.

HADR 데이터베이스가 온라인 상태일 때 매개변수를 변경한 경우, 데이터베이스에 대해 db2 get db cfg를 발행할 때 변경사항이 표시됩니다. 그러나 데이터베이스를 중지하고 재시작할 때까지 변경사항이 적용되지 않습니다. 현재 적용되는 매개변수를 검색

하려면 GET SNAPSHOT 명령, db2pd 도구 또는 스냅샷 모니터 API를 사용하십시오.

HADR 데이터베이스 역할

데이터베이스의 현재 역할은 데이터베이스 구성 매개변수 *hadr_db_role*에 의해 표시됩니다. 이 구성 매개변수의 유효한 값은 PRIMARY, STANDBY 또는 STANDARD입니다(나중 값은 데이터베이스가 HADR 데이터베이스가 아님을 표시함).

대기 데이터베이스의 상태

데이터베이스가 대기 역할에 있을 때, 롤 포워드 보류 상태이기도 합니다. 결국 대기 데이터베이스 구성은 다음을 표시합니다.

```
Rollforward pending
            = DATABASE
Restore pending                                         = YES
```

계획되지 않은 가동 중단에 응답

데이터베이스 관리 소프트웨어 또는 클러스터 관리 소프트웨어가 데이터베이스 서버가 실패했음을 감지하는 경우, 데이터베이스 솔루션이 가능한 빨리 그리고 유연하게 실패에 응답해야 합니다. 데이터베이스 솔루션은 가능한 경우 워크로드를 리라우트하여 실패로부터 사용자 응용프로그램을 차단하고 하나가 사용 가능한 경우 보조 또는 대기 데이터베이스로 장애 복구하려고 시도해야 합니다.

데이터베이스 또는 클러스터 관리 소프트웨어가 데이터베이스 서버가 실패했음을 감지하는 경우, 사용자 또는 데이터베이스나 클러스터 관리 소프트웨어가 다음을 수행해야 합니다.

1. 실패한 데이터베이스 서버에 대한 조작을 인계할 보조 데이터베이스 서버를 식별하고 온라인으로 한 후 초기화합니다.

   DB2 고가용성 재해 복구(HADR)를 사용하여 기본 및 대기 데이터베이스 서버를 관리 중인 경우, HADR이 대기 데이터베이스가 기본 데이터베이스와 동기화된 상태를 유지하도록 관리하고, HADR이 대기 데이터베이스에 의한 기본 데이터베이스의 인계를 관리합니다.

2. 사용자 응용프로그램 워크로드를 보조 데이터베이스 서버로 리라우트합니다.

   DB2 클라이언트 리라우트는 클라이언트 응용프로그램을 실패한 데이터베이스 서버에서 이 목적으로 이전에 식별되고 구성된 보조 데이터베이스 서버로 자동으로 리라우트할 수 있습니다.

3. 실패한 데이터베이스 서버를 시스템에서 제거하여 수리합니다.

사용자 응용프로그램이 보조 또는 대기 데이터베이스 서버로 리라우트된 후, 실패한 데이터베이스 서버는 재시작되거나 수리될 때까지 어떤 클라이언트 응용프로그램 요청도 처리할 수 없습니다. 예를 들어 기본 데이터베이스의 실패 원인이 데이터베이스 인스턴스가 예상치 않게 종료했기 때문인 경우, DB2 결함 모니터 기능이 자동으로 인스턴스를 재시작합니다.

자동 클라이언트 리라우트 예

DB2 Data Server 클라이언트 리라우트는 클라이언트 응용프로그램을 실패한 데이터베이스 서버에서 이 목적으로 이전에 식별되고 구성된 보조 데이터베이스 서버로 자동으로 리라우트할 수 있습니다. 이 DB2 Data Server 기능을 테스트하고 시연하는 클라이언트 응용프로그램을 쉽게 작성할 수 있습니다.

다음은 클라이언트 응용프로그램(의사 코드만을 사용하여 표시됨)에 대한 자동 클라이언트 리라우트 예입니다.

```
int checkpoint = 0;

check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
   if (sqlca->sqlcode == -30081)
   {
      // as communication is lost, terminate the application right away
      exit(1);
   }
   else

      // print out the error
      printf(...);

 if (sqlca->sqlcode == -30108)
 {
    // connection is re-established, re-execute the failed transaction
         if (checkpoint == 0)
         {
             goto checkpt0;
         }
    else if (checkpoint == 1)
         {
            goto checkpt1;
         }
         else if (checkpoint == 2)
         {
             goto checkpt2;
         }
         ....
         exit;
 }
 }
}

main()
{
   connect to mydb;
   check_sqlca("connect failed", &sqlca);

checkpt0:
   EXEC SQL set current schema XXX;
   check_sqlca("set current schema XXX failed", &sqlca);

   EXEC SQL create table t1...;
```

```
   check_sqlca("create table t1 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 1;
   }

checkpt1:
   EXEC SQL set current schema YYY;
   check_sqlca("set current schema YYY failed", &sqlca);

   EXEC SQL create table t2...;
   check_sqlca("create table t2 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 2;
   }
...
}
```

클라이언트 머신에서 『hornet』 노드를 참조하는 『mydb』라는 데이터베이스가 카탈로그됩니다. 『hornet』은 또한 노드 디렉토리(포트 번호가 456인 호스트 이름 『hornet』)에서 카탈로그됩니다.

예 1(비HADR 데이터베이스 포함)

『hornet』 서버(호스트 이름이 포트 번호가 있는 hornet과 동일함)에서, 『mydb』 데이터베이스가 작성됩니다. 또한 『mydb』 데이터베이스가 대체 서버(포트 번호가 456인 호스트 이름 『montero』)에서도 작성됩니다. 또한 다음과 같이 『hornet』 서버에서 『mydb』 데이터베이스에 대한 대체 서버를 갱신해야 합니다.

```
db2 update alternate server for database mydb using hostname montero port 456
```

위의 샘플 응용프로그램에서, 그리고 자동 클라이언트 리라우트 기능 설정이 없어도 `create table t1`문에서 통신 오류가 있는 경우 응용프로그램이 종료됩니다. 자동 클라이언트 리라우트 기능이 설정되면 DB2 데이터베이스 관리 프로그램은 『hornet』 호스트에(포트 456을 사용하여) 대한 연결을 다시 설정하려고 시도합니다. 여전히 작동하지 않는 경우 DB2 데이터베이스 관리 프로그램은 대체 서버 위치(포트 456을 사용하는 『montero』 호스트)를 시도합니다. 대체 서버 위치에 대한 연결에 통신 오류가 없다고 가정할 때, 응용프로그램이 계속 후속 명령문을 실행(및 실패한 트랜잭션을 재실행)할 수 있습니다.

예 2(HADR 데이터베이스 포함)

『hornet』 서버(호스트 이름이 포트 번호가 있는 hornet과 동일함)에서, 기본 데이터베이스 『mydb』가 작성됩니다. 대기 데이터베이스도 포트 456을 사용하는 『montero』 호

스트에 작성됩니다. 기본 및 대기 데이터베이스 둘 다에 대해 HADR을 설정하는 방법에 대한 정보는 *데이터 복구 및 고가용성 안내서 및 참조서*에 있습니다. 또한 다음과 같이 『mydb』 데이터베이스에 대한 대체 서버를 갱신해야 합니다.

```
db2 update alternate server for database mydb using hostname montero port 456
```

위의 샘플 응용프로그램에서, 그리고 자동 클라이언트 리라우트 기능 설정이 없어도 `create table t1`문에서 통신 오류가 있는 경우 응용프로그램이 종료됩니다. 자동 클라이언트 리라우트 기능이 설정되면 DB2 데이터베이스 시스템은 『hornet』 호스트에(포트 456을 사용하여) 대한 연결을 다시 설정하려고 시도합니다. 여전히 작동하지 않는 경우 DB2 데이터베이스 시스템은 대체 서버 위치(포트 456을 사용하는 『montero』 호스트)를 시도합니다. 대체 서버 위치에 대한 연결에 통신 오류가 없다고 가정할 때, 응용프로그램이 계속 후속 명령문을 실행(및 실패한 트랜잭션을 재실행)할 수 있습니다.

예 3(SSL 포함)

또한 연결에 SSL도 사용 중인 경우 클라이언트 리라우트를 사용할 수도 있습니다. 설정은 **예 2**에 표시된 것과 비슷합니다.

클라이언트 머신에서 『hornet_ssl』 노드를 참조하는 『mydb』 데이터베이스에 대한 데이터베이스 별명 『mydb_ssl』이 카탈로그됩니다. 『hornet_ssl』은 노드 디렉토리에서 카탈로그됩니다(호스트 이름은 『hornet』, SSL 포트 번호는 45678, 보안 매개변수는 SSL로 설정됨).

데이터베이스 별명은 또한 대체 서버에서 카탈로그됩니다(호스트 이름은 『montero』, SSL 포트 번호는 45678, 보안 매개변수는 SSL로 설정됨). 또한 표시된 것처럼 『hornet』 서버에서 별명 『mydb_ssl』에 대한 대체 서버를 갱신해야 합니다.

```
db2 update alternate server for database mydb_ssl using hostname montero port 45678
```

위의 샘플 응용프로그램에서 연결 명령문을 `connect to mydb_ssl`로 변경하십시오. 자동 클라이언트 리라우트 기능이 설정되지 않은 상태에서, `create table t1`문에서 통신 오류가 있는 경우 응용프로그램이 종료됩니다. 자동 클라이언트 리라우트 기능이 설정된 경우 DB2 데이터베이스 관리 프로그램이 다시 SSL을 사용하여 『hornet』 호스트(포트 45678)에 대한 연결을 설정하려고 합니다. 여전히 작동하지 않는 경우 DB2 데이터베이스 관리 프로그램은 SSL을 사용하여 대체 서버 위치(포트 45678의 『montero』 호스트)를 시도합니다. 대체 서버 위치에 대한 연결에 통신 오류가 없다고 가정할 때, 응용프로그램이 계속 후속 명령문을 실행(및 실패한 트랜잭션을 재실행)할 수 있습니다.

HADR 장애 복구 조작 수행

현재 기본 데이터베이스가 사용 불가능하기 때문에 현재 대기 데이터베이스가 새 기본 데이터베이스가 되기 원할 때 장애 복구를 수행할 수 있습니다.

경고:

이 프로시저로 인해 데이터를 유실할 수 있습니다. 이 비상 프로시저를 수행하기 전에 다음 정보를 검토하십시오.

- 기본 데이터베이스가 더 이상 데이터베이스 트랜잭션을 처리 중이 아닌지 확인하십시오. 기본 데이터베이스가 여전히 실행 중이지만 대기 데이터베이스와 통신할 수 없는 경우, 강제 인계 조작을 실행(BY FORCE 옵션을 갖는 TAKEOVER HADR 명령 실행)하면 기본 데이터베이스가 두 개가 될 수 있습니다. 두 기본 데이터베이스가 있을 때 각 데이터베이스는 서로 다른 데이터를 갖고, 두 데이터베이스는 더 이상 자동으로 동기화될 수 없습니다.
  - 기본 데이터베이스를 비활성화하거나 가능한 경우 인스턴스를 중지하십시오. (기본 시스템이 정지, 손상되었거나 그렇지 않으면 액세스 불가능한 경우 이것이 불가능할 수 있습니다.) 인계 조작이 수행된 후 실패한 데이터베이스가 나중에 재시작되는 경우 자동으로 기본 데이터베이스의 역할을 가정하지 않습니다.
- 트랜잭션 유실의 가능성과 Extent는 사용자의 특정 구성 및 상황에 따라 다릅니다.
  - 기본 데이터베이스가 피어 상태에 있거나 연결이 끊어진 피어 상태에 있고 동기화 모드가 동기(SYNC)인 동안 실패하면, 대기 데이터베이스는 기본 데이터베이스가 실패하기 전에 응용프로그램에 커미트된 것으로 보고된 트랜잭션을 유실하지 않습니다.
  - 기본 데이터베이스가 피어 상태에 있거나 연결이 끊어진 피어 상태에 있고 동기화 모드가 거의 동기(NEARSYNC)인 동안 실패하는 경우, 대기 데이터베이스는 기본 및 대기 데이터베이스 모두가 동시에 실패하는 경우에 기본 데이터베이스가 커미트한 트랜잭션만 유실할 수 있습니다.
  - 기본 데이터베이스가 피어 상태에 있거나 연결이 끊어진 피어 상태에 있고 동기화 모드가 비동기(ASYNC)인 동안 실패하는 경우, 대기 데이터베이스는 인계 조작이 수행되기 전에 대기 데이터베이스가 트랜잭션에 대한 모든 로그 레코드를 수신하지 않은 경우 기본 데이터베이스가 커미트한 트랜잭션을 유실할 수 있습니다. 대기 데이터베이스는 또한 기본 및 대기 데이터베이스 모두가 동시에 실패하는 경우에 기본 데이터베이스에 의해 커미트된 트랜잭션을 유실할 수 있습니다.
  - 기본 데이터베이스가 리모트 만회 보류 상태에 있는 중에 실패하는 경우, 대기 데이터베이스가 수신하고 처리하지 않은 트랜잭션이 유실됩니다.

    주: 데이터베이스 스냅샷에 표시되는 모든 로그 갭은 기본 및 대기 데이터베이스가 서로 통신 중인 마지막 시간의 갭을 나타냅니다. 기본 데이터베이스가 해당 시간 이후 아주 많은 수의 트랜잭션을 처리했을 수 있습니다.
- 새 기본에 연결하는(또는 클라이언트 리라우트에 의해 새 기본으로 리라우트되는) 모든 응용프로그램이 다음을 처리할 준비가 되었는지 확인하십시오.
  - 장애 복구 중에 데이터 유실이 있습니다. 새 기본이 이전 기본에서 커미트된 모든 트랜잭션을 갖지는 않습니다. HADR_SYNCMODE 구성 매개변수가 SYNC로 설정될 때도 이 상황이 발생할 수 있습니다. HADR 대기가 로그를 순차적으

로 적용하기 때문에, SQL 세션의 트랜잭션이 새 기본에서 커미트되는 경우 동일한 세션의 모든 이전 트랜잭션도 새 기본에서 커미트되었다고 가정할 수 있습니다. 다중 세션 사이에서 트랜잭션의 커미트 시퀀스는 로그 스트림의 상세한 분석을 사용해서만 판별될 수 있습니다.

  - 트랜잭션이 원래 기본에 발행되고, 원래 기본에서 커미트되고 새 기본(원래 대기)에 복제될 수 있지만, 원래 기본이 트랜잭션이 커미트된 클라이언트에 보고할 수 있기 전에 손상되기 때문에 커미트된 것으로 보고될 수 없습니다. 사용자가 작성하는 모든 응용프로그램은 트랜잭션이 원래 기본에 발행했지만 원래 기본에서 커미트된 것으로 보고되지 않았고 새 기본(원래 대기)에서 커미트되는 것을 처리할 수 있어야 합니다.
  - 데이터베이스 구성 및 외부 UDF 오브젝트에 대한 변경 같은 일부 조작은 복제되지 않습니다.

- TAKEOVER HADR 명령은 대기 데이터베이스에 대해서만 발행할 수 있습니다.
- HADR 은 실패한 데이터베이스를 자동으로 재시작하는 데 사용할 수 있는 DB2 결함 모니터(db2fm)와 인터페이스하지 않습니다. 결함 모니터가 사용 가능한 경우 아마 실패한 기본 데이터베이스에 대한 가능한 결함 모니터 조치를 인식해야 합니다.
- 인계 조작은 기본 및 대기 데이터베이스가 피어 상태에 있거나 대기 데이터베이스가 리모트 만회 보류 상태에 있는 경우에만 발생할 수 있습니다. 대기 데이터베이스가 다른 상태에 있으면 오류 메시지가 리턴됩니다.

  주: 로컬 만회 상태에 있는 대기 데이터베이스를 표준 데이터베이스로 변환하여 일반 사용에 사용 가능하게 만들 수 있습니다. 이를 수행하려면 DEACTIVATE DATABASE 명령을 발행하여 데이터베이스를 종료한 후 STOP HADR 명령을 발행하십시오. HADR이 중지된 후에는 이전 대기 데이터베이스에 대해 롤 포워드 조작을 완료해야 사용 가능하게 됩니다. 데이터베이스는 대기 데이터베이스에서 표준 데이터베이스로 변환된 후에 HADR 쌍을 다시 조인할 수 없습니다. 두 서버에서 HADR을 재시작하려면 HADR 초기화 프로시저를 수행하십시오.
  피어 창을 구성한 경우, 임의의 관련 장애 복구에서 잠재적인 트랜잭션 유실을 피하기 위해 창이 만기하기 전에 기본을 종료하십시오.

장애 복구 시나리오에서 인계 조작은 명령행 처리기(CLP), 제어 센터의 고가용성 재해 복구 관리 창 또는 db2HADRTakeover API를 통해 수행될 수 있습니다.

다음 프로시저는 CLP를 사용하여 기본 또는 대기 데이터베이스에서 장애 복구를 시작하는 방법을 보여줍니다.

1. 실패한 기본 데이터베이스를 완전히 사용 불가능하게 하십시오. 데이터베이스에서 내부 오류가 발생할 때 일반 종료 명령이 데이터베이스를 완전히 종료하지 않을 수 있습니다. 운영 체제 명령을 사용하여 프로세스, 공유 메모리 또는 네트워크 연결 같은 자원을 제거해야 할 수 있습니다.

2. 대기 데이터베이스에서 BY FORCE 옵션을 갖는 TAKEOVER HADR 명령을 발행하십시오. 다음 예에서는 LEAFS 데이터베이스에서 장애 복구가 발생합니다.

   ```
   TAKEOVER HADR ON DB LEAFS BY FORCE
   ```

   기본이 오프라인일 것으로 예상되기 때문에 BY FORCE 옵션이 필요합니다.

   기본 데이터베이스가 완전히 사용 불가능하지 않은 경우 대기 데이터베이스는 여전히 기본에 연결되어 있으며 기본 데이터베이스에 종료할 것을 요청하는 메시지를 전송합니다. 대기 데이터베이스는 기본 데이터베이스가 종료되었다는 확인을 수신하는지 여부와 상관 없이 기본 데이터베이스의 역할을 전환합니다.

HADR 인계 창을 열려면 다음을 수행하십시오.

1. 제어 센터에서 HADR을 관리하려는 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 고가용성 재해 복구 → 관리를 누르십시오. 고가용성 재해 복구 관리 창이 열립니다.
2. HADR 인계를 누르십시오. HADR 인계 창이 열립니다.
3. 장애 복구 조작을 실행하기 원함을 선택하십시오.
4. HADR 쌍의 두 데이터베이스가 모두 대기 데이터베이스로서 시작된 경우 기본 데이터베이스로서 인계할 데이터베이스 중 하나를 선택하십시오.
5. 확인을 누르십시오. 창이 닫힙니다. 진행 표시기가 열려서 명령이 실행 중인 시기를 표시할 수 있습니다. 완료되면 성공 여부를 표시하는 통지가 수신됩니다.
6. 고가용성 재해 복구 관리 창을 새로 고쳐서 대기 데이터베이스가 새 기본으로 인계했는지 확인하십시오.
7. 자동 클라이언트 리라우트 기능을 사용 중이 아닌 경우 클라이언트 응용프로그램을 새 기본 데이터베이스로 경로 재지정하십시오.

제어 센터에 있는 온라인 도움말 기능을 통해 세부사항 정보가 제공됩니다.

고가용성 재해 복구(HADR)에서 데이터베이스 역할 전환

고가용성 재해 복구(HADR) 중에 TAKEOVER HADR 명령을 사용하여 기본 및 대기 데이터베이스의 역할을 전환하십시오.

- TAKEOVER HADR 명령은 대기 데이터베이스에 대해서만 발행할 수 있습니다. 명령이 발행될 때 기본 데이터베이스가 대기 데이터베이스에 연결되지 않은 경우 인계 조작이 실패합니다.
- TAKEOVER HADR 명령은 데이터베이스가 피어 상태에 있는 경우 기본 및 대기 데이터베이스의 역할을 전환하기 위해서만 사용될 수 있습니다. 대기 데이터베이스가 다른 상태에 있으면 오류 메시지가 리턴됩니다.

명령행 처리기(CLP), 제어 센터의 고가용성 재해 복구(HADR) 관리 창 또는 db2HADRTakeover API를 사용하여 HADR 데이터베이스 역할을 전환할 수 있습니다.

CLP를 사용하여 대기 데이터베이스에서 인계 조작을 시작하려면 대기 데이터베이스에서 BY FORCE 옵션 없이 TAKEOVER HADR 명령을 발행하십시오.

다음 예에서는 대기 데이터베이스 LEAFS에서 인계 조작이 발생합니다.

```
TAKEOVER HADR ON DB LEAFS
```

로그 가득참 오류가 인계 조작 직후에 발생할 가능성이 좀 더 큽니다. 그런 오류의 가능성을 제한하기 위해, 각 인계의 종료 시에 비동기 버퍼 풀 플러시가 자동으로 시작됩니다. 로그 가득참 오류의 가능성은 비동기 버퍼 풀 플러시가 진행함에 따라서 줄어듭니다. 또한 구성이 충분한 양의 사용 중인 로그 스페이스를 제공하는 경우 로그 가득참 오류는 훨씬 더 가능성이 작습니다. 로그 가득참 오류가 발생하는 경우 트랜잭션이 중단되고 롤백됩니다.

주: BY FORCE 옵션이 없는 TAKEOVER HADR 명령을 발행하면 현재 HADR 기본 데이터베이스에 연결된 모든 응용프로그램이 강제로 해제됩니다. 이 조치는 자동 클라이언트 리라우트와 함께 작업하여 역할 전환 후 새 HADR 기본 데이터베이스로 클라이언트 리라우트를 돕도록 설계되었습니다. 그러나 응용프로그램을 기본에서 강제로 해제하는 것이 사용자 환경에서 파괴적인 경우, 사용자 고유의 프로시저를 구현하여 역할 전환을 수행하기 전에 그러한 응용프로그램을 종료한 후 역할 전환이 완료된 후 새 HADR 기본 데이터베이스를 대상으로 하여 응용프로그램을 재시작할 수 있습니다.

HADR 인계 창을 열려면 다음을 수행하십시오.

1. 제어 센터에서 HADR을 관리하려는 데이터베이스를 찾을 때까지 오브젝트 트리를 펼치십시오. 데이터베이스를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 고가용성 재해 복구 → 관리를 누르십시오. 고가용성 재해 복구 관리 창이 열립니다.
2. 데이터베이스가 피어 상태에 있는지 확인하십시오.
3. HADR 인계를 누르십시오. HADR 인계 창이 열립니다.
4. 데이터베이스 역할을 전환하기 원함을 선택하십시오.
5. HADR 쌍의 두 데이터베이스가 모두 대기 데이터베이스로서 시작된 경우 기본 데이터베이스로서 인계할 데이터베이스 중 하나를 선택하십시오.
6. 확인을 누르십시오. 창이 닫힙니다. 진행 표시기가 열려서 명령이 실행 중인 시기를 표시할 수 있습니다. 완료되면 성공 여부를 표시하는 통지가 수신됩니다.
7. 고가용성 재해 복구 관리 창을 새로 고쳐서 데이터베이스가 역할을 전환했는지 확인하십시오.
8. 자동 클라이언트 리라우트 기능을 사용 중이 아닌 경우 클라이언트 응용프로그램을 새 기본 데이터베이스로 경로 재지정하십시오.

제어 센터의 문맥 도움말 기능을 통해 추가 정보가 제공됩니다.

인계 조작 후 데이터베이스 재통합

기본 데이터베이스가 실패했기 때문에 고가용성 재해 복구(HADR) 환경에서 인계 조작을 실행한 경우, 실패한 데이터베이스를 다시 온라인으로 만들고 대기 데이터베이스로 사용하거나 상태를 기본 데이터베이스로 리턴할 수 있습니다.

HADR 쌍의 실패한 기본 데이터베이스를 새 대기 데이터베이스로서 재통합하려면 다음을 수행하십시오.

1. 원래 기본 데이터베이스가 있던 시스템을 수리하십시오. 이것은 실패한 하드웨어 수리 또는 손상된 운영 체제 재부팅을 포함할 수 있습니다.
2. 실패한 기본 데이터베이스를 대기 데이터베이스로 재시작하십시오. 다음 예에서 LEAFS 데이터베이스가 대기 데이터베이스로서 시작됩니다.

   ```
   START HADR ON DB LEAFS AS STANDBY
   ```

   주: 데이터베이스의 두 사본에 호환되지 않는 로그 스트림이 있는 경우 재통합은 실패합니다. 특히 HADR은 원래 기본 데이터베이스가 새 기본 데이터베이스로서 인계하기 전에 원래 대기 데이터베이스에서 반영되지 않은 모든 로그된 조작을 적용하지 않았을 것을 요구합니다. 이러한 상황이 발생한 경우, 새 기본 데이터베이스의 백업 이미지를 리스토어하거나 분할 미러를 초기화하여 원래 기본 데이터베이스를 대기 데이터베이스로서 재시작할 수 있습니다.

   이 명령의 성공적인 리턴이 재통합이 성공했음을 표시하지는 않습니다. 단지 데이터베이스가 시작되었음을 의미합니다. 재통합은 여전히 진행 중입니다. 재통합이 나중에 실패하는 경우 데이터베이스는 스스로 종료됩니다. GET SNAPSHOT FOR DATABASE 명령이나 db2pd 도구를 사용하여 대기 상태를 모니터하여 대기 데이터베이스가 온라인 상태로 머물고 정상 상태 전이를 계속하는지 확인해야 합니다. 필요한 경우 관리 통지 로그 파일 및 db2diag 로그 파일을 점검하여 데이터베이스의 상태를 확인할 수 있습니다.

원래 기본 데이터베이스가 대기 데이터베이스로서 HADR 쌍을 재결합한 후, 장애 회복 조작을 수행하여 데이터베이스의 역할을 전환하여 원래 기본 데이터베이스가 다시 기본 데이터베이스가 될 수 있도록 선택할 수 있습니다. 이 장애 회복 조작을 수행하려면 대기 데이터베이스에서 다음 명령을 발행하십시오.

```
TAKEOVER HADR ON DB LEAFS
```

주:

1. HADR 데이터베이스가 피어 상태에 있지 않거나 쌍이 연결되지 않은 경우 이 명령은 실패합니다.

2. 기본 데이터베이스에서 열린 세션은 강제로 닫히고 인플라이트(inflight) 트랜잭션은 롤백됩니다.
3. 기본 및 대기 데이터베이스의 역할을 전환할 때 TAKEOVER HADR 명령의 BY FORCE 옵션을 지정할 수 없습니다.

제 2 부 데이터 복구

복구는 미디어 또는 스토리지 오류, 전원 인터럽트 또는 응용프로그램 실패와 같은 문제점이 발생한 후 데이터베이스 또는 테이블 스페이스를 재빌드하는 것입니다. 데이터베이스나 개별적 테이블 스페이스를 백업한 경우 어떤 방식으로든지 손상되거나 훼손되면 재빌드할 수 있습니다.

세 가지 유형의 복구가 있습니다.

- 응급 복구는 데이터베이스가 트랜잭션(작업 단위라고도 함)이 예상치 않게 인터럽트될 때 불일치 또는 사용 불가능 상태가 되지 않도록 보호합니다.
- 버전 복구는 백업 작업 중에 작성된 이미지를 사용하여 데이터베이스의 이전 버전을 리스토어합니다.
- 롤 포워드 복구는 백업이 수행된 후 커미트된 트랜잭션에서 작성된 변경사항을 다시 적용하기 위해 사용할 수 있습니다.

DB2 데이터베이스 관리 프로그램은 전원 인터럽트 후 데이터베이스를 복구하기 위해 자동으로 응급 복구를 시작합니다. 버전 복구 또는 롤 포워드 복구를 사용하여 손상된 데이터베이스를 복구할 수 있습니다.

제 7 장 백업 및 복구 전략 개발

하드웨어 또는 소프트웨어 실패(또는 둘 다) 때문에 데이터베이스가 사용 불가능해질 수 있습니다. 한 번에 또는 다른 경우에 스토리지 문제점, 전력 인터럽트 또는 응용프로그램 실패가 발생할 수 있으며 실패 시나리오마다 서로 다른 복구 조치가 필요합니다. 안전하게 테스트된 복구 전략을 세워 데이터가 유실되지 않도록 데이터를 보호합니다. 복구 전략을 세울 때 다음과 같은 몇 가지 질문에 대답할 수 있어야 합니다.

- 데이터베이스가 복구 가능합니까?
- 데이터베이스를 복구하는 데 어느 정도의 시간이 걸립니까?
- 백업 작업 간격은 어느 정도입니까?
- 백업 사본 및 아카이브된 로그에 어느 정도의 스토리지 스페이스를 할당할 수 있습니까?
- 테이블 스페이스 레벨 백업 크기가 충분합니까? 또는 전체 데이터베이스 백업이 필요합니까?
- 고가용성 재해 복구(HADR)를 통해 또는 수동으로 대기 시스템을 구성해야 합니까?

데이터베이스 복구 전략에서는 데이터베이스 복구에 필요한 경우 모든 정보가 사용 가능한지 확인해야 합니다. 정기적인 데이터베이스 백업 스케줄을 포함해야 하며 파티션된 데이터베이스 환경의 경우 시스템 크기를 조정할 때(데이터베이스 파티션 서버 또는 노드가 추가되거나 삭제됨) 백업을 포함해야 합니다. 전반적인 전략에는 명령 스크립트, 응용프로그램, 사용자 정의 함수(UDF), 운영 체제 라이브러리의 스토어드 프로시저 코드 및 로드 사본을 복구하는 프로시저도 포함되어야 합니다.

다음에 나오는 절에서 다른 복구 방법을 설명합니다. 이를 통해 사용자의 비즈니스 환경에 가장 적합한 복구 방법을 찾을 수 있습니다.

데이터베이스 *백업*이란 개념은 기타 데이터 백업과 동일합니다. 데이터를 복사하고 원래 매체가 손상되거나 장애가 발생한 경우 다른 매체에 저장하는 작업을 말합니다. 가장 단순한 백업 방법은 추가 트랜잭션이 발생하지 않도록 데이터베이스를 종료하고 백업하는 것입니다. 그러면 손상되거나 파손된 경우 데이터베이스를 다시 작성할 수 있습니다.

데이터베이스를 다시 작성하는 작업을 *복구*라고 합니다. *버전 복구*는 백업 작업 중 작성된 이미지를 사용하여 데이터베이스의 이전 버전을 리스토어하는 작업입니다. *롤 포워드 복구*는 데이터베이스 또는 테이블 스페이스 백업 이미지를 리스토어한 후 데이터베이스 로그 파일에 기록된 트랜잭션을 다시 적용하는 것입니다.

*응급 복구*는 하나 이상의 작업 단위(UOW)에 포함된 모든 변경 사항을 완료 및 커미트하기 전에 장애가 발생한 경우 수행되는 데이터베이스의 자동 복구를 말합니다. 불완전한 트랜잭션을 롤백하고 손상되었을 때 메모리에 남아 있는 커미트된 트랜잭션을 완료하면 됩니다.

데이터베이스를 작성하면 복구 로그 파일 및 복구 실행기록 파일은 자동으로 작성됩니다(203 페이지의 그림 10 참조). 유실되거나 손상된 데이터를 복구해야 하는 경우 이러한 로그 파일이 중요합니다.

각 데이터베이스에는 응용프로그램 또는 시스템 오류로부터 복구하는 데 사용되는 *복구 로그*가 있습니다. 데이터베이스 백업과 함께 사용하여 최대 오류가 발생한 특정 시점까지 데이터베이스 일관성을 복구합니다.

*복구 실행기록 파일*에는 데이터베이스 전체 또는 일부를 지정된 특정 시점으로 복구해야 하는 경우 복구 옵션을 판별할 때 사용할 수 있는 백업 정보 요약이 포함됩니다. 이 파일은 특히, 백업 및 리스토어 작업과 같은 복구 관련 이벤트를 추적하는 데 사용합니다. 이 파일은 데이터베이스 디렉토리에 있습니다.

테이블 스페이스 변경 실행기록 파일 역시 데이터베이스 디렉토리에 있으며 특정 테이블 스페이스의 복구에 필요한 로그 파일을 판별하는 데 사용할 수 있는 정보를 포함합니다.

복구 실행기록 파일 또는 테이블 스페이스 변경 실행기록 파일은 직접 수정할 수 없습니다. 그러나 PRUNE HISTORY 명령을 사용하여 파일에서 항목을 삭제할 수 있습니다. 또한 *rec_his_retentn* 데이터베이스 구성 매개변수를 사용하여 이러한 실행기록 파일을 보유하는 기간(일)을 지정할 수도 있습니다.

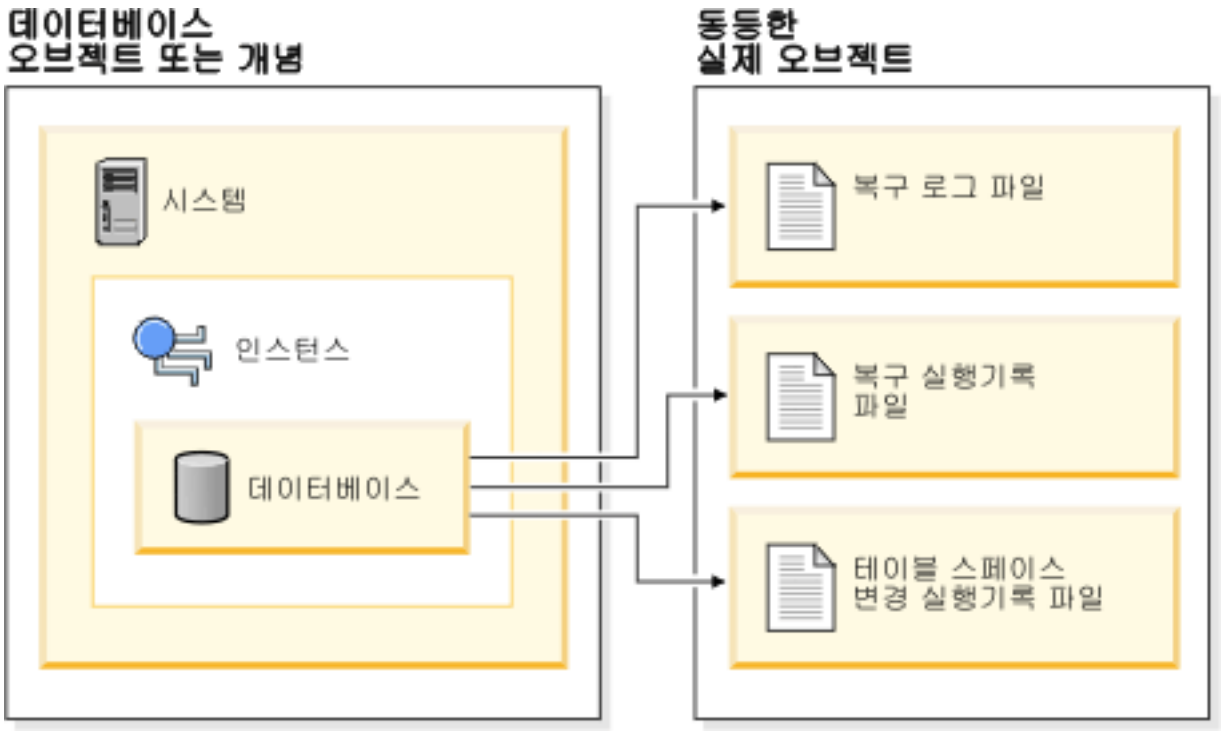


그림 10. 데이터베이스 복구 파일

쉽게 재작성된 데이터는 복구 불가능한 데이터베이스에 저장될 수 있습니다. 자주 갱신되지 않는 테이블 및 읽기 전용 응용프로그램에서 사용되는 외부 소스의 데이터를 포함합니다. 이때 작은 규모의 로깅은 리스토어 작업 후 로그 파일 관리 및 롤 포워드의 추가된 복잡도를 정당화하지 않습니다. *logarchmeth1* 및 *logarchmeth2* 데이터베이스 구성 매개변수 모두 『OFF』로 설정되면 데이터베이스는 *복구 불가능*합니다. 즉, 보존된 로그만 응급 복구에 필요한 로그임을 의미합니다. 이러한 로그는 *사용 중인 로그*라고 하며 여기에는 현재 트랜잭션 데이터가 포함됩니다. *오프라인* 백업을 사용하는 버전 복구는 복구 불가능 데이터베이스의 기본 복구 방법입니다. 이때 오프라인 백업은 백업 작업을 진행 중일 때 다른 응용프로그램에서 데이터베이스를 사용할 수 없음을 나타냅니다. 이러한 데이터베이스는 오프라인으로만 리스토어될 수 있습니다. 백업 이미지를 취한 당시의 상태로 리스토어되고 롤 포워드 복구는 지원되지 않습니다.

쉽게 재작성할 수 없는 데이터는 복구 가능한 데이터베이스에 저장해야 합니다. 여기에는 데이터가 로드된 후 소스가 파괴된 데이터, 수동으로 테이블에 입력된 데이터 및 데이터베이스로 로드된 후 응용프로그램 또는 사용자가 수정한 데이터가 포함됩니다. *복구 가능한 데이터베이스*에서 *logarchmeth1* 또는 *logarchmeth2* 데이터베이스 구성 매개변수는 『OFF』 이외의 값으로 설정되어 있습니다. 사용 중인 로그는 응급 복구에서 여전히 사용 가능하지만 커미트된 트랜잭션 데이터를 포함하는 *아카이브된 로그*도 보유하고 있습니다. 이러한 데이터베이스는 오프라인으로만 리스토어될 수 있습니다. 백업 이미지를 취한 당시의 상태로 리스토어됩니다. 그러나 롤 포워드 복구의 경우 사용 중인

로그 및 아카이브된 로그를 사용하여 사용 중인 로그의 끝으로 또는 특정 시점으로 데이터베이스를 롤 *포워드*(즉, 백업 이미지를 취한 과거의 시점)할 수 있습니다.

복구 가능한 데이터베이스 백업 작업은 오프라인 또는 *온라인*에서 모두 수행할 수 있습니다. 이때 온라인은 백업 작업 중 다른 응용프로그램이 데이터베이스에 연결할 수 있음을 나타냅니다. 온라인 테이블 스페이스 리스토어 및 롤 포워드 조작은 복구 가능한 데이터베이스에서만 지원됩니다. 데이터베이스가 복구 불가능한 경우 데이터베이스 리스토어 및 롤 포워드 조작은 오프라인으로 수행해야 합니다. 온라인 백업 작업 중 롤 포워드 복구를 수행하면 백업이 리스토어될 때 모든 테이블 변경을 캡처하고 다시 적용합니다.

복구 가능한 데이터베이스인 경우 전체 데이터베이스가 아닌 개별 테이블을 백업, 리스토어 및 롤 포워드할 수 있습니다. 온라인 상태에서 테이블 스페이스를 백업하면 계속 사용할 수 있으며 동시 갱신사항이 로그에 기록됩니다. 테이블 스페이스에서 온라인 리스토어 또는 롤 포워드 조작을 수행하는 경우 작업을 완료할 때까지 테이블 스페이스 자체는 사용 불가능하지만 다른 테이블 스페이스의 테이블에는 액세스할 수 있습니다.

자동화된 백업 조작

백업 작업과 같은 유지보수 활동을 실행하는지 여부와 실행하는 시기를 판별하는 데는 시간이 많이 걸릴 수 있으므로 자동 유지보수 구성 마법사를 사용하면 자동으로 이 작업을 수행할 수 있습니다. 자동 유지보수를 사용하는 경우 자동 유지보수를 실행할 수 있는 시점을 포함하여 유지보수 목표를 지정합니다. 그러면 DB2에서 이 목표를 사용하여 유지보수 활동을 수행해야 하는지 여부를 판별하고 다음에 사용 가능한 유지보수 기간(자동 유지보수 활동을 실행하는 사용자 정의된 기간) 중에 필요한 유지보수 활동만 실행합니다.

주: 자동 유지보수가 구성되면 계속 수동 백업 작업을 수행할 수 있습니다. 필요한 경우 DB2는 자동 백업 조작만 수행합니다.

백업 빈도 결정

데이터베이스 백업에는 시간 및 시스템 자원이 필요하기 때문에 복구 계획은 정기적으로 스케줄된 백업 조작을 고려해야 합니다. 전체 데이터베이스 백업과 증분 백업 조작을 조합하여 계획에 포함시킬 수 있습니다.

로그를 아카이브하더라도, 전체 데이터베이스를 정기적으로 백업해야 합니다(롤 포워드 복구 고려). 데이터베이스를 복구하기 위해 모든 테이블 스페이스 백업 이미지가 들어 있는 전체 데이터베이스 백업 이미지를 사용할 수 있습니다. 아니면 선택된 테이블 스페이스 이미지를 사용하여 데이터베이스를 재빌드할 수 있습니다. 테이블 스페이스 백업 이미지는 고립된 디스크 오류나 응용프로그램 오류를 복구할 경우에 유용합니다. 파

티션된 데이터베이스 환경에서는 실패한 데이터베이스 파티션에 있는 테이블 스페이스만 리스토어하면 됩니다. 모든 테이블 스페이스 또는 모든 데이터베이스 파티션을 리스토어할 필요는 없습니다.

더 이상 데이터베이스 복구 시 전체 데이터베이스 백업이 필요하지는 않더라도 테이블 스페이스 이미지로부터 데이터베이스를 재빌드할 수 있습니다. 이는 간혹 데이터베이스 전체 백업을 수행해야 하는 좋은 방법입니다.

또한 백업 이미지와 로그를 겹쳐쓰지 않고 둘 이상의 전체 데이터베이스 백업과 연관 로그를 저장하는 방안도 추가적인 예방책으로 고려해야 합니다.

자주 사용되는 데이터베이스를 복구하고 롤 포워드할 때 아카이브 로그를 적용하는데 필요한 시간이 문제가 될 경우, 데이터베이스를 더 자주 백업하는 것을 고려하십시오. 그러면 롤 포워드할 때 적용해야 하는 아카이브 로그 수가 줄어듭니다.

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

데이터베이스를 사용할 수 없는 시간을 줄이려면, 온라인 백업 조작 사용을 고려하십시오. 온라인 백업 조작은 롤 포워드 복구가 사용 가능할 경우에만 지원됩니다. 롤 포워드 복구가 사용 가능하고 전체 복구 로그 세트가 있으면, 필요한 경우 데이터베이스를 리스토어할 수 있습니다. 백업 조작이 실행되던 시간이 로그에 포함되어 있는 경우에만 복구에 온라인 백업 이미지를 사용할 수 있습니다.

오프라인 백업 조작은 데이터 파일 경합이 일어나지 않기 때문에 온라인 백업 조작보다 빠릅니다.

백업 유틸리티를 사용하면, 선택한 테이블 스페이스를 백업할 수 있습니다. DMS 테이블 스페이스를 사용하면, 자체 테이블 스페이스에 여러 데이터 유형을 저장하여 백업 조작에 필요한 시간을 줄일 수 있습니다. 테이블 데이터를 하나의 테이블 스페이스에, Long 필드 및 LOB 데이터를 다른 테이블 스페이스에, 인덱스를 또 다른 테이블 스페이스에 보존할 수 있습니다. 이렇게 하면 디스크 장애가 발생할 경우 테이블 스페이스 중 하나만 영향을 받습니다. 이들 테이블 스페이스 중 하나를 리스토어하거나 롤 포워드하는 것은 모든 데이터가 들어 있는 단일 테이블 스페이스를 리스토어할 때보다 시간이 적게 걸립니다.

또한 각 테이블 스페이스에 대한 변경사항이 동일하지만 않다면 각 테이블 스페이스를 서로 다른 시간에 백업하여 시간을 절약할 수 있습니다. 따라서 Long 필드나 LOB 데이터가 다른 데이터처럼 자주 변경되지 않을 경우에는 이러한 테이블 스페이스를 자주 백업하지 않아도 됩니다. Long 필드나 LOB 데이터가 복구에 필요하지 않을 경우에는 이 데이터가 들어 있는 테이블 스페이스를 복구하지 않아도 됩니다. 별도의 소스에서

LOB 데이터를 재생성할 수 있는 경우, 테이블이 LOB 컬럼을 포함하도록 테이블을 작성하거나 변경할 때에는 NOT LOGGED 옵션을 선택하십시오.

주: 다음은 Long 필드 데이터, LOB 데이터 및 인덱스를 별도의 테이블 스페이스에 보존하지만 함께 백업하지 않을 경우의 고려사항입니다. 모든 테이블 데이터를 포함하지 않은 테이블 스페이스를 백업하는 경우, 해당 테이블 스페이스에서 특정 시점 롤 포워드 복구를 수행할 수 없습니다. 테이블에 대한 데이터 유형이 들어 있는 모든 테이블 스페이스는 동일한 시점으로 동시에 롤 포워드되어야 합니다.

테이블을 재구성하는 경우, 조작이 완료된 후에 영향을 받은 테이블 스페이스를 백업해야 합니다. 테이블 스페이스를 리스토어해야 하는 경우, 데이터 재구성을 통해 롤 포워드할 필요는 없습니다.

데이터베이스를 복구하는데 필요한 시간은 두 부분으로 구성됩니다. 백업을 완전히 리스토어하는 데 필요한 시간과 데이터베이스를 포워드 복구에 사용할 수 있는 경우 롤 포워드 조작 중에 로그를 적용하는데 필요한 시간입니다. 복구 계획을 구체화할 때, 복구 비용 및 사용자 비즈니스 조작에 대한 영향을 고려해야 합니다. 전반적인 복구 계획을 테스트하면, 데이터베이스를 복구하는데 필요한 시간이 비즈니스 요구사항에 비추어 합당한지 여부를 판별하는 데 도움이 됩니다. 각 테스트 다음에 백업 수행 빈도를 늘리고자 할 수 있습니다. 롤 포워드 복구가 전략의 일부라면, 백업 도중에 아카이브되는 로그의 수를 줄임으로써 리스토어 조작 후에 데이터베이스를 롤 포워드하는 데 필요한 시간을 줄일 수 있습니다.

복구 시 스토리지 고려사항

사용할 복구 방법을 결정할 때 필요한 스토리지 공간을 고려합니다.

버전 복구 방법을 사용하려면 리스토어된 데이터베이스 및 데이터베이스의 백업 사본을 보유할 스페이스가 필요합니다. 롤 포워드 복구 방법을 사용하려면 데이터베이스 또는 테이블 스페이스, 리스토어된 데이터베이스 및 아카이브된 데이터베이스 로그의 백업 사본을 보유할 스페이스가 필요합니다.

테이블에 long 필드 또는 대형 오브젝트(LOB) 컬럼이 있는 경우 별도의 테이블 스페이스에 이 데이터를 배치하는 것이 좋습니다. 이는 스토리지 스페이스 고려사항 및 복구 계획에도 영향을 줍니다. long 필드 및 LOB 데이터를 백업하는 데 필요한 시간을 알면 long 필드 및 LOB 데이터에 대한 별도의 테이블 스페이스를 사용할 때 이 테이블 스페이스의 백업만 저장하는 복구 계획을 사용할 수 있습니다. 또한 LOB 컬럼에 대한 변경을 로그하지 않고 LOB 컬럼을 포함하도록 테이블을 작성하거나 변경하는 방법도 선택할 수 있습니다. 그러면 필요한 로그 스페이스 및 해당되는 로그 아카이브 스페이스의 *크기*가 줄어듭니다.

데이터베이스를 삭제하지 않고 미디어 장애를 예방하고 복구 기능을 보존하려면 데이터베이스 백업, 데이터베이스 로그 및 데이터베이스 자체를 다른 디바이스에 보존합니다. 따라서 데이터베이스를 작성할 때 별도의 디바이스에 데이터베이스 로그를 배치하려면 *newlogpath* 구성 매개변수를 사용하는 것이 좋습니다.

데이터베이스 로그는 최대 스토리지 크기를 사용할 수 있습니다. 롤 포워드 복구 방법을 사용하려는 경우 아카이브된 로그를 관리하는 방법을 결정해야 합니다. 다음 중에서 선택할 수 있습니다.

- LOGARCHMETH1 또는 LOGARCHMETH2 구성 매개변수를 사용하여 로그 아카이브 방법을 지정합니다.
- 더 이상 사용 중인 로그 세트에 포함되지 않는 경우 데이터베이스 로그 경로 디렉토리 이외의 다른 스토리지 디바이스 또는 디렉토리에 로그를 수동으로 복사합니다.
- User Exit 프로그램을 사용하여 이 로그를 환경의 다른 스토리지 디바이스에 저장합니다.

관련 데이터 함께 보존

데이터베이스 디자인 프로세스에서 테이블 간 관계를 더 잘 이해할 수 있습니다. 이 관계는 다음과 같이 표시할 수 있습니다.

- 응용프로그램 레벨, 트랜잭션에서 둘 이상의 테이블을 갱신하는 경우
- 데이터베이스 레벨, 테이블 간 참조 무결성이 존재하거나 한 테이블의 트리거가 다른 테이블에 영향을 주는 경우

복구 플랜을 개발하는 경우 이 관계를 고려해야 합니다. 데이터의 관련된 세트를 함께 백업하려는 경우가 있습니다. 이러한 세트는 테이블 스페이스 또는 데이터베이스 레벨로 설정할 수 있습니다. 관련 데이터 세트를 함께 보존하면 모든 데이터가 일관된 지점으로 복구할 수 있습니다. 특히 테이블 스페이스에서 특정 시점 롤 포워드 복구를 수행하도록 설정하려는 경우에 중요합니다.

서로 다른 운영 체제 및 하드웨어 플랫폼 간 백업 및 리스토어 작업

DB2 데이터베이스 시스템은 서로 다른 운영 체제 및 하드웨어 플랫폼 간 백업 및 리스토어 작업을 지원합니다.

DB2 백업 및 리스토어 작업에 대해 지원되는 플랫폼은 다음과 같이 세 계열 중 하나로 그룹화될 수 있습니다.

- 빅 엔디안(big-endian) Linux 및 UNIX
- 리틀 엔디안(little-endian) Linux 및 UNIX
- Windows

한 플랫폼 계열의 데이터베이스 백업은 동일한 플랫폼 계열의 모든 시스템에서 리스토어될 수 있습니다. Windows 운영 체제의 경우 DB2 버전 9 데이터베이스 시스템에서 DB2 UDB(Universal Database) V8에서 작성된 데이터베이스를 리스토어할 수 있습니다. Linux 및 UNIX 운영 체제의 경우 백업 및 리스토어 플랫폼의 엔디안(빅 엔디안 또는 리틀 엔디안)이 동일한 경우에 한해, DB2 UDB V8에서 생성된 백업을 DB2 버전 9에서 리스토어할 수 있습니다.

다음 표는 DB2에서 지원하는 각 Linux 및 UNIX 플랫폼을 보여주고 플랫폼 유형(빅 엔디안 또는 리틀 엔디안)을 표시합니다.

표 9. DB2가 지원하는 지원되는 Linux 및 UNIX 운영 체제의 엔디안

플랫폼	엔디안
AIX	빅 엔디안
HP on IA64	빅 엔디안
Solaris x64	리틀 엔디안
Solaris SPARC	빅 엔디안
Linux on zSeries®	빅 엔디안
Linux on pSeries®	빅 엔디안
Linux on IA-64	리틀 엔디안
Linux on AMD64 및 Intel® EM64T	리틀 엔디안
32비트 Linux on x86	리틀 엔디안

목표 시스템에 소스 시스템과 동일한 버전(또는 이상)의 DB2 데이터베이스 제품이 있어야 합니다. 데이터베이스 제품의 한 버전에서 작성된 백업은 데이터베이스 제품의 이전 버전을 실행하는 시스템으로 리스토어할 수 없습니다. 예를 들어 DB2 UDB V8 백업을 DB2 V9 데이터베이스 시스템에서 리스토어할 수 있지만 DB2 V9 백업을 DB2 UDB V8 데이터베이스 시스템에서 리스토어할 수는 없습니다.

주: 32비트 레벨에서 가져온 백업 이미지로부터 데이터베이스를 64비트 레벨로 리스토어할 수는 있지만 그 반대는 불가능합니다. 데이터베이스를 백업 및 리스토어하려면 DB2 백업 및 리스토어 유틸리티를 사용해야 합니다. 한 머신에서 다른 머신으로 파일 세트를 이동하지 않는 것이 좋습니다. 데이터베이스 무결성을 위협할 수 있기 때문입니다.

특정 백업 및 리스토어 조합이 허용되지 않는 경우 다음과 같은 다른 방법을 사용하여 DB2 데이터베이스 사이에서 테이블을 이동할 수 있습니다.

- db2move 명령
- 임포트 또는 로드 유틸리티 이전에 수행되는 익스포트 유틸리티

제 8 장 복구 실행기록 파일

복구 실행기록 파일은 각 데이터베이스에서 작성되며 다음 경우에 항상 자동으로 갱신됩니다.

- 데이터베이스 또는 테이블 스페이스가 백업됨
- 데이터베이스 또는 테이블 스페이스가 리스토어됨
- 데이터베이스 또는 테이블 스페이스가 롤 포워드됨
- 데이터베이스가 자동으로 재빌드되고 둘 이상의 이미지가 리스토어됨
- 테이블 스페이스가 작성됨
- 테이블 스페이스가 변경됨
- 테이블 스페이스가 Quiesce 상태임
- 테이블 스페이스 이름이 바뀜
- 테이블 스페이스가 삭제됨
- 테이블이 로드됨
- 테이블이 삭제됨(삭제된 테이블 복구가 가능한 경우)
- 테이블이 재구성됨
- 온 디맨드 로그 아카이브가 호출됨
- 새 로그 파일에 작성됨(복구 가능한 로그를 사용하는 경우)
- 로그 파일이 아카이브됨(복구 가능한 로그를 사용하는 경우)
- 데이터베이스가 복구됨

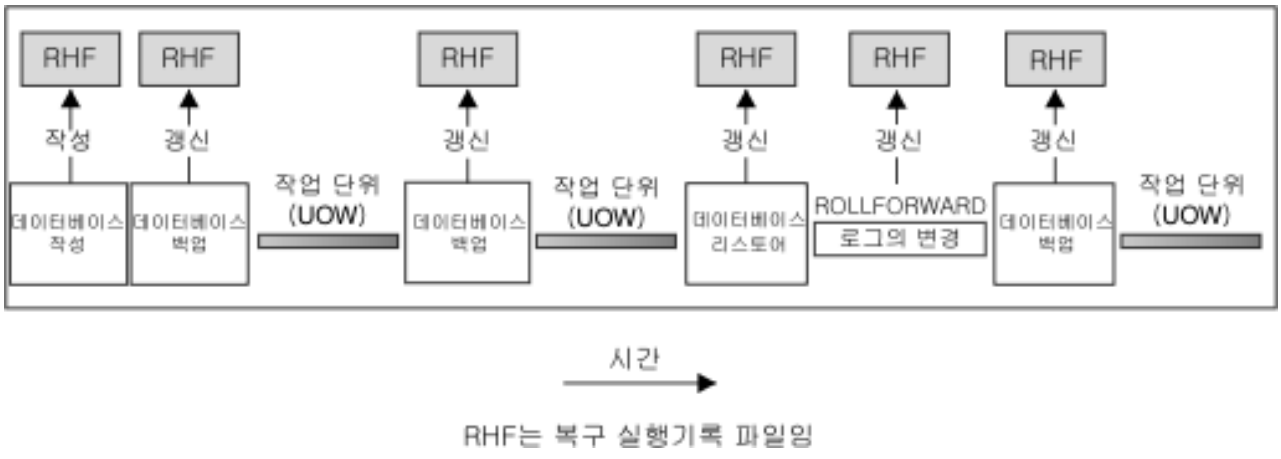


그림 11. 복구 실행기록 파일 작성 및 갱신

이 파일에 요약된 백업 정보를 사용하여 지정된 특정 시점으로 데이터베이스 전체 또는 일부를 복구할 수 있습니다. 다음은 파일의 정보입니다.

- 각 항목을 고유하게 식별하는 ID 필드
- 복사한 데이터베이스 파트 및 방법
- 복사한 시간
- 사본 위치(사본에 액세스할 논리적 방법 및 디바이스 정보 모두를 언급함)
- 리스토어 작업을 완료한 마지막 시간
- 테이블 스페이스 이름이 바뀐 시간(테이블 스페이스의 이전 및 현재 이름 표시)
- 백업 작업 상태(사용 중, 비활성, 만기됨 또는 삭제됨)
- 롤 포워드 복구 조작 중 처리되거나 데이터베이스 백업에 의해 저장된 마지막 로그 시퀀스 번호

복구 실행기록 파일에서 항목을 보려면 LIST HISTORY 명령을 사용합니다.

모든 백업 작업(데이터베이스, 테이블 스페이스 또는 증분)에는 복구 실행기록 파일 사본이 포함됩니다. 복구 실행기록 파일은 데이터베이스와 연관됩니다. 데이터베이스를 삭제하면 복구 실행기록 파일도 삭제됩니다. 데이터베이스를 새 위치로 리스토어하면 복구 실행기록 파일도 리스토어됩니다. 리스토어는 디스크에 있는 파일에 항목이 있는 한, 기존 복구 실행기록 파일을 겹쳐쓰지 않습니다. 이 경우 데이터베이스 실행기록이 백업 이미지에서 리스토어됩니다.

현재 데이터베이스가 사용 불가능하고 연관된 복구 실행기록 파일이 손상되거나 삭제된 경우 RESTORE 명령의 옵션을 사용하여 복구 실행기록 파일만 리스토어할 수 있습니다. 그러면 복구 실행기록 파일을 검토하여 데이터베이스를 리스토어하는 데 사용할 백업에 대한 정보를 제공할 수 있습니다.

파일 크기는 파일에서 항목의 보존 기간(일)을 지정하는 *rec_his_retentn* 구성 매개변수로 제어됩니다. 이 매개변수의 값이 영(0)으로 설정되어도 최근 전체 데이터베이스 백업(및 해당 리스토어 세트)이 보존됩니다. 이 사본을 제거하는 유일한 방법은 FORCE 옵션과 함께 PRUNE을 사용하는 것입니다. 보존 기간의 디폴트값은 366일입니다. 기간은 -1을 사용하여 무한 기간으로 설정할 수도 있습니다. 이 경우 파일을 명시적으로 프룬(prune)해야 합니다.

실행기록 파일 항목 상태 복구

데이터베이스 관리 프로그램은 백업 작업, 리스토어 작업, 테이블 스페이스 작성 및 기타와 같은 이벤트에 대한 항목을 복구 실행기록 파일에 작성합니다. 복구 실행기록 파일의 각 항목에는 연관된 상태(활성, 비활성, 만기됨, 삭제됨 또는 do_not_delete)가 있습니다.

데이터베이스 관리 프로그램은 복구 실행기록 파일 항목의 상태를 사용하여 해당 항목과 연관되는 실제 파일이 데이터베이스 복구에 필요한지 여부를 판별합니다. 자동화된 프룬(prune)의 일부로, 데이터베이스 관리 프로그램은 실행기록 파일 항목의 상태를 갱신합니다.

활성 데이터베이스 백업

활성 데이터베이스 백업은 데이터베이스의 현재 상태를 복구하기 위해 현재 로그를 사용하여 리스토어 및 롤 포워드할 수 있는 백업입니다.

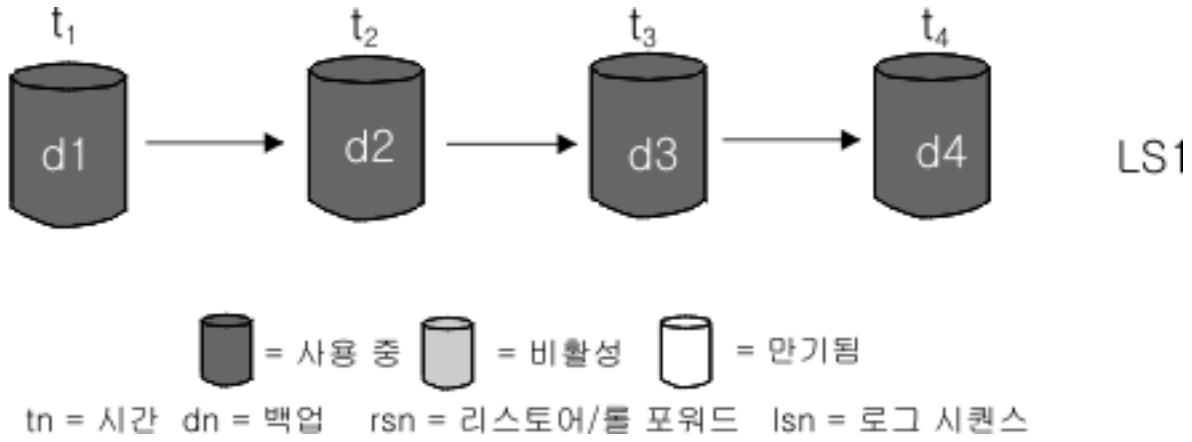


*그림 12. 활성 데이터베이스 백업. num_db_backups*의 값이 4로 설정되었습니다.

비활성 데이터베이스 백업

비활성 데이터베이스 백업은 리스토어된 경우 데이터베이스가 다시 이전 상태로 이동하는 백업입니다.

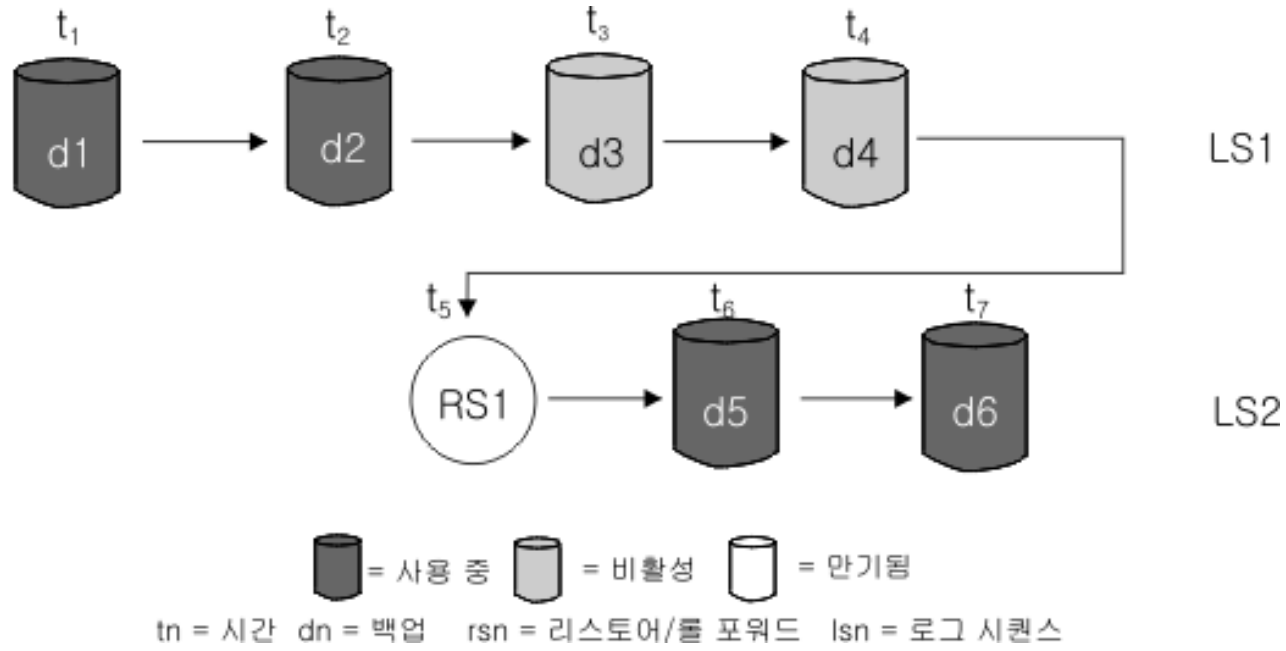


그림 13. 비활성 데이터베이스 백업

만기된 데이터베이스 백업

만기된 데이터베이스 백업은 더 이상 필요하지 않은 백업 이미지입니다. 최근 백업 이미지를 사용할 수 있기 때문입니다.

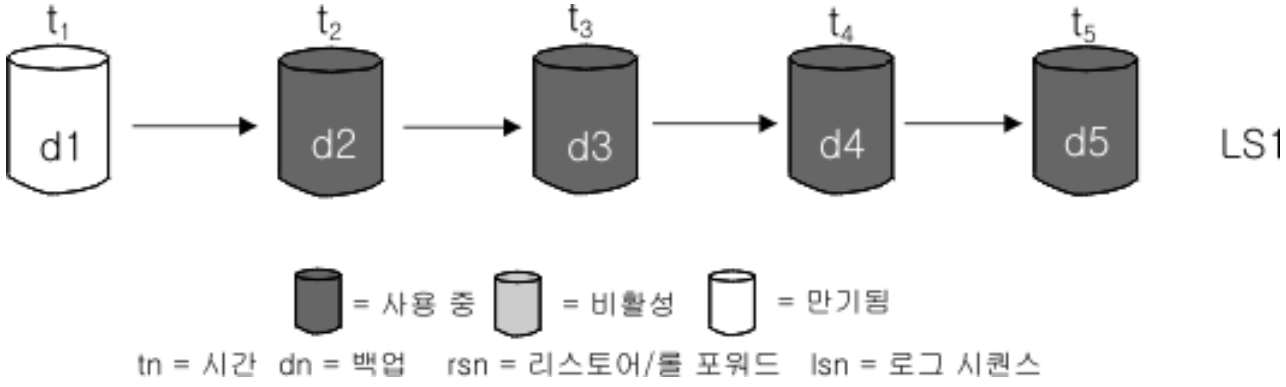


그림 14. 만기된 데이터베이스 백업

do_not_delete로 표시된 항목

PRUNE HISTORY 명령이나 db2Prune API를 사용하여 복구 실행기록 파일 항목을 제거(프룬)할 수 있습니다. 데이터베이스 관리 프로그램은 또한 자동화된 프룬(prune)의 일부로 복구 실행기록 파일 항목을 프룬(prune)합니다.

do_not_delete로 표시된 항목을 프룬할 수 있는 방법은 세 가지입니다.

- WITH FORCE 옵션으로 PRUNE HISTORY 명령 호출

- PRUNE HISTORY 및 WITH FORCE 옵션으로 ADMIN_CMD 프로시저 호출
- DB2_PRUNE_OPTION_FORCE 옵션으로 db2Prune API 호출

do_not_delete로 표시된 항목은 세 가지 조치 중 하나를 수행하지 않는 경우 복구 실행기록 파일에서 프룬되지 않습니다.

데이터베이스 관리 프로그램은 복구 실행기록 파일 항목을 do_not_delete로 설정하지 않습니다. UPDATE HISTORY 명령을 사용하여 복구 실행기록 파일 항목 상태를 do_not_delete로 설정할 수 있습니다.

다음은 복구 실행기록 파일 항목의 상태에 대한 추가 예입니다.

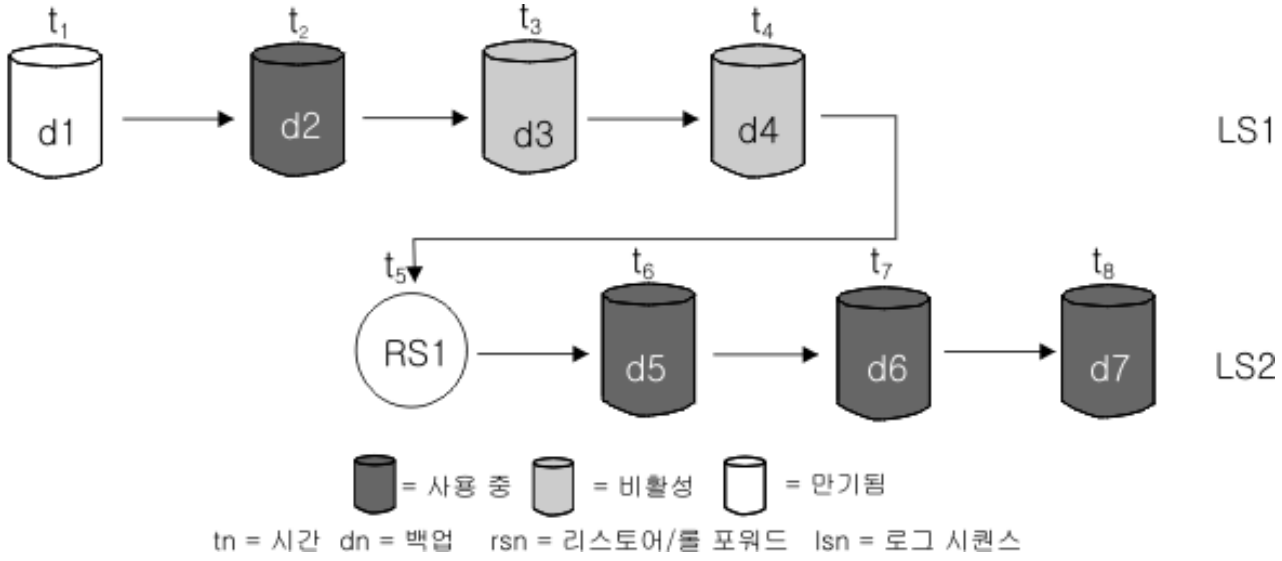


그림 15. 혼합된 활성, 비활성 및 만기된 데이터베이스 백업

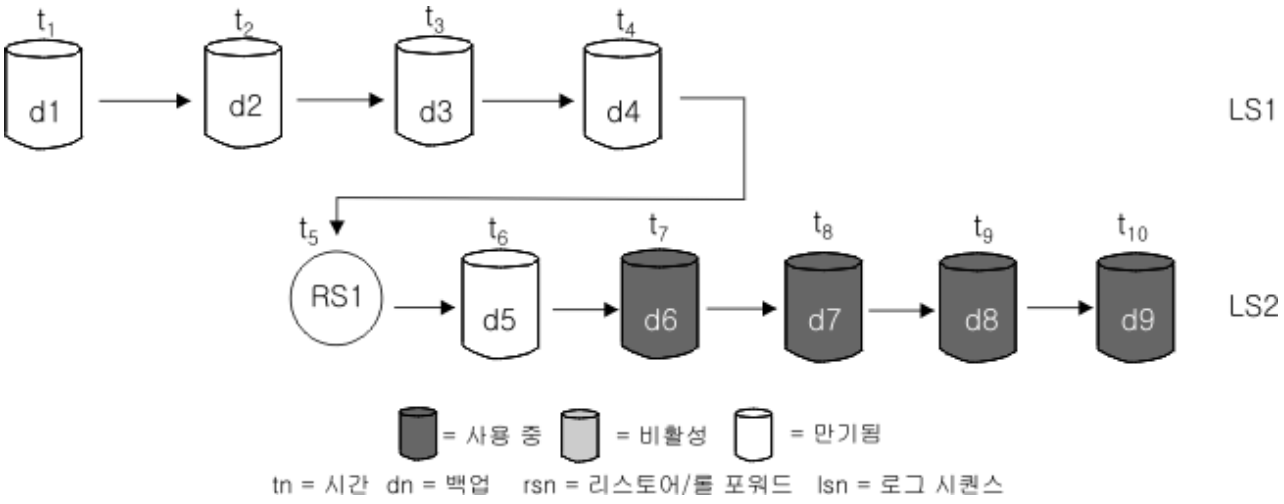


그림 16. 만기된 로그 시퀀스

DB_HISTORY 관리 뷰를 사용한 복구 실행기록 파일 보기

DB_HISTORY() 관리 뷰를 사용하여 데이터베이스 실행기록 파일의 컨텐츠에 액세스할 수 있습니다. 이 메소드는 LIST HISTORY CLP 명령 또는 C 실행기록 API 사용의 대안입니다.

이 함수를 사용하려면 데이터베이스 연결이 필요합니다.

데이터베이스 실행기록 파일에 대한 삭제 및 갱신은 PRUNE 또는 UPDATE HISTORY 명령을 통해서만 수행할 수 있습니다.

이 관리 뷰는 DB2 Universal Database 버전 8.2 이하를 사용하여 작성된 데이터베이스에는 사용할 수 없습니다.

실행기록 파일에 액세스하려면 다음을 수행하십시오.

1. 데이터베이스에 연결되었는지 확인하십시오.
2. SQL SELECT문에서 DB_HISTORY() 관리 뷰를 사용하여 사용자가 연결된 데이터베이스나 DB2NODE 환경에 의해 지정되는 데이터베이스 파티션에 대한 데이터베이스 실행기록 파일에 액세스하십시오. 예를 들어 실행기록 파일의 컨텐츠를 보려면 다음을 사용하십시오.

```
SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

관리 뷰의 구문을 숨기려면 다음과 같이 뷰를 작성할 수 있습니다.

```
CREATE VIEW LIST_HISTORY AS
  SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

이 뷰를 작성한 후 뷰에 대해 쿼리를 실행할 수 있습니다. 예를 들어, 다음과 같습니다.

```
SELECT * FROM LIST_HISTORY
```

또는

```
SELECT dbpartitionnum FROM LIST_HISTORY
```

또는

```
SELECT dbpartitionnum, start_time, seqnum, tabname, sqlstate
  FROM LIST_HISTORY
```

표 10 표는 LIST_HISTORY 테이블 함수에 의해 리턴되는 컬럼 및 컬럼 데이터 유형의 목록입니다.

표 10. 실행기록 테이블의 컨텐츠

컬럼 이름	데이터 유형
dbpartitionnum smallint	smallint
EID	bigint
start_time	char(14)
seqnum	smallint
end_time	varchar(14)
firstlog	varchar(254)
lastlog	varchar(254)
backup_id	varchar(24)
tabschema	varchar(128)
tabname	varchar(128)
comment	varchar(254)
cmd_text	clob(2M)
num_tbsps	integer
tbspnames	clob(5M)
operation	char(1)
operationtype	char(1)
objecttype	varchar(255)
location	char(1)
devicetype	char(1)
entry_status	varchar(8)
sqlcaid	varchar(8)
sqlcabc	integer
sqlcode	integer
sqlerrml	smallint
sqlerrmc	varchar(70)
sqlerrp	varchar(8)
sqlerrd1	integer

표 10. 실행기록 테이블의 컨텐츠 (계속)

컬럼 이름	데이터 유형
sqlerrd2	integer
sqlerrd3	integer
sqlerrd4	integer
sqlerrd5	integer
sqlwarn	varchar(11)
sqlstate	varchar(5)

복구 실행기록 파일 프룬(prune)

데이터베이스 관리 프로그램은 백업 작업, 리스토어 작업, 테이블 스페이스 작성 및 기타와 같은 이벤트에 대한 항목을 복구 실행기록 파일에 작성합니다. 연관된 복구 오브젝트가 더 이상 데이터베이스를 복구하기 위해 필요하지 않기 때문에 복구 실행기록 파일의 항목이 더 이상 관련이 없을 때, 해당 항목을 복구 실행기록 파일에서 제거 또는 프룬*(prune)*할 수 있습니다.

다음 메소드를 사용하여 복구 실행기록 파일의 항목을 프룬(prune)할 수 있습니다.

- PRUNE HISTORY 명령 호출
- db2Prune API 호출
- PRUNE_HISTORY 매개변수와 함께 ADMIN_CMD 프로시저 호출

이러한 메소드 중 하나를 사용하여 복구 실행기록 파일을 프룬(prune)할 때, 데이터베이스 관리 프로그램이 복구 실행기록 파일에서 사용자가 지정하는 시간소인보다 오래된 항목을 제거(프룬)합니다.

복구 실행기록 파일 항목이 프룬(prune) 대상으로 지정하는 기준과 일치하지만 해당 항목이 여전히 데이터베이스 복구를 위해 필요한 경우, 데이터베이스 관리 프로그램은 WITH FORCE 매개변수나 DB2PRUNE_OPTION_FORCE 플래그를 사용하지 않는 한 항목을 프룬하지 않습니다.

AND DELETE 매개변수나 DB2PRUNE_OPTION_DELETE 플래그를 사용하는 경우, 프룬된 항목과 연관된 로그 파일도 삭제됩니다.

AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수를 ON으로 설정하고 AND DELETE 매개변수나 DB2PRUNE_OPTION_DELETE 플래그를 사용하는 경우, 프룬된 항목과 연관된 로그 파일, 백업 이미지 및 로드 사본 이미지가 삭제됩니다.

복구 실행기록 파일 프룬(prune) 자동화

복구 실행기록 파일 항목의 상태를 자동으로 프룬(prune) 및 갱신하도록 데이터베이스 관리 프로그램을 구성할 수 있습니다.

UPDATE HISTORY 명령, db2HistoryUpdate API 또는 ″UPDATE_HISTORY″ 매개변수가 있는 ADMIN_CMD 프로시저를 사용하여 복구 실행기록 파일에서 항목의 상태를 수동으로 갱신할 수 있습니다. PRUNE HISTORY 명령, db2Prune API 또는 ″UPDATE_HISTORY″ 매개변수가 있는 ADMIN_CMD 프로시저를 사용하여 복구 실행기록 파일에서 항목을 수동으로 제거 또는 프룬(prune)할 수 있습니다. 그러나 복구 실행기록 파일을 수동으로 갱신 및 프룬(prune)하는 대신 복구 실행기록 파일을 관리하도록 데이터베이스 관리 프로그램을 구성하는 것이 좋습니다.

데이터베이스 관리 프로그램은 다음 시기에 복구 실행기록 파일 항목을 자동으로 갱신하고 프룬(prune)합니다.

- 전체(비증분) 데이터베이스 백업 작업 또는 전체(비증분) 테이블 스페이스 작업이 성공적으로 완료된 후
- 데이터베이스 리스토어 작업(롤 포워드 작업이 필요하지 않은)이 성공적으로 완료된 후
- 데이터베이스 롤 포워드 작업이 성공적으로 완료된 후

자동화된 프룬(prune) 중에는 데이터베이스 관리 프로그램이 다음의 두 작업을 수행합니다.

1. 복구 실행기록 파일 항목의 상태를 갱신합니다.
2. 만기된 복구 실행기록 파일 항목을 프룬(prune)합니다.

데이터베이스 관리 프로그램은 다음 방법으로 복구 실행기록 파일 항목을 갱신합니다.

- 더 이상 필요하지 않은 모든 활성 데이터베이스 백업 이미지는 만기된 것으로 표시됩니다.
- 비활성 상태로 표시되고, 만기된 데이터베이스 백업을 취한 시점 이전에 취한 모든 데이터베이스 백업 이미지도 만기된 것으로 표시됩니다. 연관된 모든 비활성 테이블 스페이스 백업 이미지와 로드 백업 사본도 만기된 것으로 표시됩니다.
- 활성 데이터베이스 백업 이미지가 리스토어되었지만 실행기록 파일에 기록된 최근 데이터베이스 백업이 아닌 경우, 동일한 로그 시퀀스에 속하는 연속 데이터베이스 백업 이미지는 비활성 상태로 표시됩니다.
- 비활성 데이터베이스 백업 이미지가 리스토어되는 경우, 현재 로그 시퀀스에 속하는 비활성 데이터베이스 백업은 다시 활성 상태로 표시됩니다. 더 이상 현재 로그 시퀀스에 없는 모든 활성 데이터베이스 백업 이미지는 비활성 상태로 표시됩니다.
- 현재 로그 시퀀스(현재 로그 체인이라고도 함)에 해당하지 않는 데이터베이스 또는 테이블 스페이스 백업 이미지는 비활성 상태로 표시됩니다.

  현재 로그 시퀀스는 리스토어된 데이터베이스 백업 이미지와, 처리된 로그 파일로 결정됩니다. 데이터베이스 백업 이미지가 리스토어되면, 리스토어된 이미지가 새 로그 체인을 시작하므로 모든 연속 데이터베이스 백업 이미지는 비활성 상태가 됩니다. (이

는 백업 이미지가 롤 포워드 없이 리스토어된 경우에 적용됩니다. 롤 포워드 작업이 발생한 경우, 로그 체인에서 중단 이후에 취한 모든 데이터베이스 백업은 비활성 상태로 표시됩니다. 롤 포워드 유틸리티는 손상된 현재 백업 이미지를 포함하는 로그 시퀀스를 통해 진행되므로 오래된 데이터베이스 백업 이미지를 리스토어해야 할 것으로 생각할 수 있습니다.)

- 테이블 스페이스 레벨 백업 이미지가 리스토어된 후 현재 로그 시퀀스를 적용해도 데이터베이스의 현재 상태에 도달할 수 없는 경우 테이블 스페이스 백업 이미지는 비활성 상태가 됩니다.
- do_not_delete 상태의 항목은 프룬되지 않으며, 해당되는 연관 로그 파일, 백업 이미지 및 로드 사본 이미지는 삭제되지 않습니다.
- 데이터베이스가 업그레이드될 때 모든 온라인 데이터베이스 백업 항목과, 실행기록 파일에 있는 모든 온라인 또는 오프라인 테이블 스페이스 백업 항목은 자동 재빌드에서 재빌드에 필요한 이미지로 선택되지 않도록 만기된 것으로 표시됩니다. 로드 사본 이미지와 로그 아카이브 항목도 만기된 것으로 표시됩니다. 이러한 유형의 항목은 복구 목적에 사용할 수 없기 때문입니다.

다음 데이터베이스 구성 매개변수는 데이터베이스 관리 프로그램이 프룬(prune)하는 항목을 제어합니다.

num_db_backups
데이터베이스에 대해 보유할 데이터베이스 백업 수를 지정합니다.

rec_his_retentn
백업에서 실행기록 정보를 보유할 일 수를 지정합니다.

auto_del_rec_obj
프룬하는 복구 실행기록 파일 항목과 연관되는 로그 파일, 백업 이미지 및 로드 사본 이미지를 데이터베이스 관리 프로그램이 삭제해야 하는지 여부를 지정합니다.

복구 실행기록 파일을 데이터베이스 관리 프로그램이 자동으로 관리하도록 구성하려면 다음 구성 매개변수를 설정하십시오.

- num_db_backups
- rec_his_retentn
- auto_del_rec_obj

auto_del_rec_obj가 ON으로 설정되고 num_db_backups 구성 매개변수보다 더 성공적인 데이터베이스 백업 항목이 있을 때마다, 데이터베이스 관리 프로그램은 rec_his_retentn보다 오래된 복구 실행기록 파일 항목을 자동으로 프룬(prune)합니다.

복구 실행기록 파일 항목이 프룬되지 않도록 보호

복구 실행기록 파일 항목의 상태를 do_not_delete로 설정하여 키 복구 실행기록 파일 항목이 프룬되지 않고 연관된 복구 오브젝트가 삭제되지 않도록 막을 수 있습니다.

PRUNE HISTORY 명령, PRUNE_HISTORY를 사용하는 ADMIN_CMD 프로시저 또는 db2Prune API를 사용하여 복구 실행기록 파일 항목을 제거(프룬)할 수 있습니다. PRUNE HISTORY와 함께 AND DELETE 매개변수를 사용하거나 db2Prune과 함께 DB2PRUNE_OPTION_DELETE 플래그를 사용하고 AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수가 ON으로 설정된 경우, 연관된 복구 오브젝트는 실제로도 삭제됩니다.

데이터베이스 관리 프로그램은 또한 자동화된 프룬(prune)의 일부로 복구 실행기록 파일 항목을 프룬(prune)합니다. AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수가 ON으로 설정되면 데이터베이스 관리 프로그램은 프룬된 모든 항목과 연관된 복구 오브젝트를 삭제합니다.

키 복구 실행기록 파일 항목 및 연관된 복구 오브젝트를 보호하려면 다음을 수행하십시오.

UPDATE HISTORY 명령, db2HistoryUpdate API 또는 ″UPDATE_HISTORY″와 함께 ADMIN_CMD 프로시저를 사용하여 키 복구 파일 항목의 상태를 do_no_delete로 설정하십시오.
do_not_delete로 표시된 항목을 프룬하는 세 가지 방법이 있습니다.

- WITH FORCE 옵션으로 PRUNE HISTORY 명령 호출
- PRUNE HISTORY 및 WITH FORCE 옵션으로 ADMIN_CMD 프로시저 호출
- DB2_PRUNE_OPTION_FORCE로 db2Prune API 호출

사용자가 이러한 세 프로시저 중 하나를 수행하지 않으면 do_not_delete로 표시되는 항목은 복구 실행기록 파일에서 프룬되지 않습니다.
제한사항:

- 백업 이미지, 로드 사본 이미지 및 로그 파일만의 상태를 do_not_delete로 설정할 수 있습니다.
- 백업 항목의 상태는 로드 사본 이미지, 증분이 아닌 백업 또는 해당 백업 작업과 관련된 로그 파일에 전파되지 않습니다. 특정 데이터베이스 백업 항목 및 관련된 로그 파일 항목을 저장하려는 경우 데이터베이스 백업 항목 및 각 관련된 로그 파일에 대한 항목의 상태를 설정해야 합니다.

제 9 장 복구 오브젝트 관리

정기적으로 데이터베이스를 백업할 때 매우 많은 데이터베이스 백업 이미지와 많은 데이터베이스 로그 및 로드 사본 이미지를 축적할 수 있습니다. IBM Data Server 데이터베이스 관리 프로그램이 이러한 복구 오브젝트 관리를 단순화할 수 있습니다.

복구 오브젝트 저장은 많은 양의 스토리지 스페이스를 소비할 수 있습니다. 후속 백업 작업이 실행된 후에는 이전 복구 오브젝트를 삭제할 수 있습니다. 이들은 더 이상 데이터베이스를 리스토어하기 위해 필요하지 않기 때문입니다. 그러나 이전 복구 오브젝트 제거에 시간이 걸릴 수 있습니다. 또한 이전 복구 오브젝트를 삭제하는 중에 우발적으로 아직 필요한 복구 오브젝트를 손상시킬 수 있습니다.

데이터베이스 관리 프로그램을 사용하여 더 이상 데이터베이스를 리스토어할 때 필요하지 않은 복구 오브젝트를 삭제하는 두 가지 방법이 있습니다.

- AND DELETE 매개변수와 함께 PRUNE HISTORY 명령을 호출하거나, DB2PRUNE_OPTION_DELETE 플래그를 사용하여 db2Prune API를 호출할 수 있습니다.
- 불필요한 복구 오브젝트를 자동으로 삭제하도록 데이터베이스 관리 프로그램을 구성할 수 있습니다.

PRUNE HISTORY 명령이나 db2Prune API를 사용하여 데이터베이스 복구 오브젝트 삭제

AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수와 PRUNE HISTORY 명령이나 db2Prune API를 사용하여 복구 오브젝트를 삭제할 수 있습니다.

PRUNE HISTORY 명령을 호출하거나 db2Prune API를 호출할 때 IBM Data Server 데이터베이스 관리 프로그램이 다음을 수행합니다.

- 복구 실행기록 파일에서 DB2HISTORY_STATUS_DO_NOT_DEL 상태를 갖지 않는 항목 프룬(prune)

AND DELETE 매개변수와 함께 PRUNE HISTORY 명령을 호출할 때 또는 DB2PRUNE_OPTION_DELETE 플래그를 사용하여 db2Prune API를 호출할 때 데이터베이스 관리 프로그램이 다음을 수행합니다.

- 복구 실행기록 파일에서 사용자가 지정하는 시간소인보다 오래되고 DB2HISTORY_STATUS_DO_NOT_DEL 상태를 갖지 않는 항목 프룬(prune)
- 프룬된 항목과 연관된 물리 로그 파일 삭제

AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수를 ON으로 설정하는 경우, AND DELETE 매개변수와 함께 PRUNE HISTORY 명령을 호출할 때 또는 DB2PRUNE_OPTION_DELETE 플래그를 사용하여 db2Prune API를 호출할 때 데이터베이스 관리 프로그램이 다음을 수행합니다.

- 복구 실행기록 파일에서 DB2HISTORY_STATUS_DO_NOT_DEL 상태를 갖지 않는 항목 프룬(prune)
- 프룬된 항목과 연관된 물리 로그 파일 삭제
- 프룬된 항목과 연관된 백업 이미지 삭제
- 프룬된 항목과 연관된 로드 사본 이미지 삭제

불필요한 복구 오브젝트를 삭제하려면 다음을 수행하십시오.

1. AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수를 ON으로 설정하십시오.
2. AND DELETE 매개변수와 함께 PRUNE HISTORY 명령을 호출하거나, DB2PRUNE_OPTION_DELETE 플래그를 사용하여 db2Prune API를 호출하십시오.

데이터베이스 복구 오브젝트 관리 자동화

AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수 및 자동화된 복구 실행기록 파일 프룬(prune)을 사용하여 모든 전체 데이터베이스 백업 작업 후에 불필요한 복구 오브젝트를 자동으로 삭제하도록 IBM Data Server 데이터베이스 관리 프로그램을 구성할 수 있습니다.

모든 성공적인 전체(증분이 아닌) 데이터베이스 백업 작업 후에, 데이터베이스 관리 프로그램은 num_db_backup 및 rec_his_retentn 구성 매개변수의 값에 따라서 복구 실행기록 파일을 프룬(prune)합니다.

- 복구 실행기록 파일에 num_db_backup 구성 매개변수의 값보다 많은 데이터베이스 백업 항목이 있는 경우, 데이터베이스 관리 프로그램은 복구 실행기록 파일에서 rec_his_retentn 구성 매개변수의 값보다 오래되었고 DB2HISTORY_STATUS_DO_NOT_DEL 상태를 갖지 않는 항목을 프룬(prune)합니다.

AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수를 ON으로 설정하는 경우, 데이터베이스 관리 프로그램은 복구 실행기록 파일에서 항목을 프룬(prune)하는 것 외에 다음을 수행합니다.

- 프룬된 항목과 연관된 물리 로그 파일 삭제
- 프룬된 항목과 연관된 백업 이미지 삭제
- 프룬된 항목과 연관된 로드 사본 이미지 삭제

현재 복구 실행기록에서 고려할 수 있는 전체 데이터베이스 백업 이미지가 없는 경우(작업된 백업이 없을 수도 있음), REC_HIS_RETENTN으로 지정되는 시간 범위보다 오래된 이미지가 삭제됩니다.

파일이 더 이상 복구 실행기록 파일에 나열되는 위치에 없기 때문에 데이터베이스 관리 프로그램이 파일을 삭제할 수 없는 경우 데이터베이스 관리 프로그램은 해당 실행기록 항목을 프룬(prune)합니다.

데이터베이스 관리 프로그램과 스토리지 관리 프로그램 또는 디바이스 간의 통신 오류 때문에 데이터베이스 관리 프로그램이 파일을 삭제할 수 없는 경우. 데이터베이스 관리 프로그램은 실행기록 파일 항목을 프룬(prune)하지 않습니다. 오류가 해결될 때 다음 자동화 프룬(prune) 중에 해당 파일을 삭제할 수 있습니다.

불필요한 복구 오브젝트를 자동으로 삭제하도록 데이터베이스 관리 프로그램을 구성하려면 다음을 수행하십시오.

1. AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수를 ON으로 설정하십시오.
2. rec_his_retentn 및 num_db_backups 구성 매개변수를 설정하여 자동 복구 실행기록 파일 프룬(prune)을 사용하십시오.

복구 오브젝트가 삭제되지 않도록 보호

자동화된 복구 오브젝트 관리는 관리 시간과 스토리지 스페이스를 절약합니다. 그러나 특정 복구 오브젝트가 자동으로 삭제되지 못하도록 막기 원할 수 있습니다. 연관된 복구 실행기록 파일 항목의 상태를 do_not_delete로 설정하여 키 복구 오브젝트가 삭제되지 않도록 막을 수 있습니다.

이 태스크에 대한 정보

AUTO_DEL_REC_OBJ 데이터베이스 구성 매개변수를 ON으로 설정하면 복구 오브젝트는 연관된 복구 실행기록 파일 항목이 프룬될 때 삭제됩니다. 복구 실행기록 파일 항목은 다음 중 하나가 발생할 때 프룬됩니다.

- AND DELETE 매개변수와 함께 PRUNE HISTORY 명령을 호출합니다.
- DB2PRUNE_OPTION_DELETE 플래그와 함께 db2Prune API를 호출합니다.
- 데이터베이스 관리 프로그램이 자동으로 복구 실행기록 파일을 프룬하며, 이것은 각 성공적인 테이블 스페이스 또는 데이터베이스 전체 백업 후에 발생합니다.

PRUNE HISTORY 명령을 호출하거나 db2Prune API를 호출하거나, 복구 실행기록 파일에서 자동으로 항목을 프룬하도록 데이터베이스 관리 프로그램을 구성하는지 여부와 상관없이, do_not_delete로 표시되는 항목은 프룬되지 않으며 연관된 복구 오브젝트는 삭제되지 않습니다.

프로시저

연관된 복구 파일 항목의 상태를 do_no_delete로 설정하려면 UPDATE HISTORY 명령을 사용하십시오.
제한사항:

- 백업 이미지, 로드 사본 이미지 및 로그 파일만의 상태를 do_not_delete로 설정할 수 있습니다.
- 백업 항목의 상태는 로그 파일, 로드 사본 이미지, 해당 백업 작업과 관련된 증분이 아닌 백업에 전파되지 않습니다. 특정 데이터베이스 백업 항목 및 관련된 로그 파일 항목을 저장하려는 경우 데이터베이스 백업 항목 및 각 관련된 로그 파일에 대한 항목의 상태를 설정해야 합니다.

스냅샷 백업 오브젝트 관리

스냅샷 백업 오브젝트를 관리하려면 db2acsutil 명령을 사용해야 합니다. 파일 시스템 유틸리티를 사용하여 스냅샷 백업 오브젝트를 이동하거나 삭제하지 마십시오.

시작하기 전에

스냅샷 백업 및 리스토어 작업을 수행하려면 스토리지 디바이스에 대해 DB2 ACS API 드라이버가 필요합니다. IBM Data Server에 통합된 드라이버는 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage® SAN Volume Controller
- IBM System Storage™ DS6000™
- IBM System Storage DS8000®
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS 시리즈

스냅샷 백업 오브젝트를 관리하려면 DB2 ACS(Advanced Copy Services)를 사용 가능하게 해야 합니다. 345 페이지의 『DB2 ACS(Advanced Copy Services) 사용』을 참조하십시오.

제한사항

db2acsutil 명령은 현재 AIX 및 Linux에서만 지원됩니다.

프로시저

1. 사용 가능한 스냅샷 백업 오브젝트를 나열하려면 **QUERY** 매개변수를 사용하십시오.

예를 들어 데이터베이스 관리 프로그램 인스턴스 dbminst1에 대한 사용 가능한 스냅샷 백업 오브젝트를 나열하려면 다음 구문을 사용하십시오.

```
db2acsutil query instance dbminst1
```

2. 주어진 스냅샷 백업 조작의 진행을 점검하려면 **STATUS** 매개변수를 사용하십시오.

   예를 들어 현재 database1 데이터베이스에서 실행 중일 수 있는 스냅샷 백업 조작의 진행을 확인하려면 다음 구문을 사용하십시오.

   ```
   db2acsutil query status db database1
   ```

3. 특정 스냅샷 백업 오브젝트를 삭제하려면 **DELETE** 매개변수를 사용하십시오.

   예를 들어 10일이 넘은 database1 데이터베이스에 대한 모든 스냅샷 백업 오브젝트를 삭제하려면 다음 구문을 사용하십시오.

   ```
   db2acsutil delete older than 10 days ago db database1
   ```

제 10 장 백업, 리스토어 및 복구 작업의 진행 모니터링

LIST UTILITIES 명령을 사용하여 데이터베이스에 대한 백업, 리스토어 및 롤 포워드 작업을 모니터할 수 있습니다.

백업, 리스토어 및 복구 작업에 대한 진행 모니터링을 사용하려면 다음을 수행하십시오.

LIST UTILITIES 명령을 실행하고 SHOW DETAIL 옵션을 지정하십시오.

```
list utilities show detail
```

다음은 오프라인 데이터베이스 백업 작업의 성능 모니터링 출력의 예입니다.

```
LIST UTILITIES SHOW DETAIL

ID                                   = 2
Type                         = BACKUP
Database Name                = SAMPLE
Description                  = offline db
Start Time                   = 10/30/2003 12:55:31.786115
Throttling:
Priority                     = Unthrottled
Progress Monitoring:
Estimated Percentage Complete    = 41
      Total Work Units     = 20232453 bytes
      Completed Work Units = 230637 bytes
      Start Time           = 10/30/2003 12:55:31.786115
```

백업 작업의 경우, 처리될 바이트 수의 초기 추정이 지정됩니다. 백업 작업이 진행됨에 따라서 처리될 바이트 수가 갱신됩니다. 표시되는 바이트는 이미지의 *크기*에 대응하지 않으며 백업 이미지 *크기*에 대한 추정으로 사용해서는 안됩니다. 실제 이미지는 증분 또는 압축 백업인지 여부에 따라서 훨씬 더 작을 수 있습니다.
리스토어 작업의 경우 초기 추정이 제공되지 않습니다. 대신 UNKNOWN이 지정됩니다. 각 버퍼를 이미지에서 읽을 때, 읽은 바이트의 실제 양이 갱신됩니다. 다중 이미지가 리스토어될 자동 증분 리스토어 작업의 경우, 진행은 단계를 사용하여 추적됩니다. 각 단계는 증분 체인으로부터 리스토어될 이미지를 나타냅니다. 처음에는 단 한 단계만 표시됩니다. 첫 번째 이미지가 리스토어된 후, 전체 단계 수가 표시됩니다. 각 이미지가 리스토어될 때 완료된 단계 수가 갱신되며 처리된 바이트 수도 갱신됩니다.
응급 복구 및 롤 포워드 복구의 경우, FORWARD 및 BACKWARD의 두 진행 모니터링 단계가 있습니다. FORWARD 단계 동안, 로그 파일을 읽고 로그 레코드가 데이터베이스에 적용됩니다. 응급 복구의 경우, 전체 작업량은 마지막 로그 파일의 끝까지 시작 로그 시퀀스 번호를 사용하여 계산됩니다. 롤 포워드 복구의 경우, 이 단계가 시작될 때 UNKNOWN이 전체 작업 추정에 대해 지정됩니다. 프로세스가 계속될 때 바이트 단위의 처리된 작업량이 갱신됩니다.

BACKWARD 단계 중에, FORWARD 단계 중에 적용된 모든 언커미트된 변경이 롤백됩니다. 바이트 단위로 처리될 로그 데이터량에 대한 추정이 제공됩니다. 프로세스가 계속될 때 바이트 단위의 처리된 작업량이 갱신됩니다.

테이블 스페이스 상태

테이블 스페이스의 현재 상태는 해당 *상태*에 의해 반영됩니다. 복구와 가장 일반적으로 연관되는 테이블 스페이스 상태는 다음과 같습니다.

- *백업 보류*. 테이블 스페이스가 특정 시점 롤 포워드 작업 후, 또는 no copy 옵션을 사용하는 로드 작업 후에 이 상태가 됩니다. 테이블 스페이스는 사용하기 전에 백업해야 합니다. (백업하지 않으면 테이블 스페이스를 갱신할 수 없지만 읽기 전용 작업은 가능합니다.)
- *리스토어 보류*. 테이블 스페이스에 대한 롤 포워드 작업이 취소되거나 테이블 스페이스에 대한 롤 포워드 작업에서 복구할 수 없는 오류가 발생하여 테이블 스페이스를 리스토어하고 다시 롤 포워드해야 하는 경우 테이블 스페이스는 이 상태가 됩니다. 또한 리스토어 작업 중에 테이블 스페이스를 리스토어할 수 없는 경우에도 테이블 스페이스는 이 상태가 됩니다.
- *롤 포워드 진행 중*. 테이블 스페이스에 대한 롤 포워드 작업이 진행 중인 경우 테이블 스페이스는 이 상태가 됩니다. 롤 포워드 작업이 완료되면 테이블 스페이스는 더 이상 롤 포워드 진행 중 상태가 되지 않습니다. 테이블 스페이스는 또한 롤 포워드 작업이 취소되는 경우 이 상태를 벗어날 수 있습니다.
- *롤 포워드 보류*. 테이블 스페이스는 리스토어된 후, 또는 입출력 오류 다음에 이 상태가 됩니다. 리스토어된 후, 테이블 스페이스는 로그의 끝으로 또는 특정 시점으로 롤 포워드될 수 있습니다. 입출력 오류 다음에, 테이블 스페이스는 로그 끝으로 롤 포워드해야 합니다.

제 11 장 백업 개요

DB2 BACKUP DATABASE 명령의 가장 단순한 양식에서는 백업하려는 데이터베이스의 별명 이름만 지정하면 됩니다. 예를 들어, 다음과 같습니다.

```
db2 backup db sample
```

명령이 완료되면, 명령이 발행된 디렉토리나 경로에 위치된 새 백업 이미지가 획득됩니다. 이 이미지는 이 디렉토리에 위치됩니다. 이 예에 있는 명령은 백업 이미지의 대상 위치를 명시적으로 지정하지 않기 때문입니다. DB2 버전 9.5 이상에서 작성된 백업 이미지는 파일 모드 600에서 생성됩니다. 즉, UNIX에서 인스턴스 소유자만 읽기 쓰기 특권을 가지고 Windows에서는 DB2ADMNS(및 관리자) 그룹 구성원만 백업 이미지에 대한 액세스 권한을 갖습니다.

주: DB2 클라이언트 및 서버가 동일한 시스템에 있지 않으면 DB2는 클라이언트 머신에서 현재 작업 디렉토리인 디렉토리를 판별하고 그 디렉토리를 서버에서 백업 대상 디렉토리로 사용합니다. 이와 같은 이유로, 백업 이미지의 대상 디렉토리를 지정할 것을 권장합니다.

백업 이미지는 백업 유틸리티를 호출할 때 지정한 대상 위치에서 작성됩니다. 이 위치는 다음이 될 수 있습니다.

- 디렉토리(디스크 또는 디스켓으로 백업하는 경우)
- 디바이스(테이프로 백업하는 경우)
- TSM(Tivoli Storage Manager) 서버
- 다른 벤더의 서버

복구 실행기록 파일은 사용자가 데이터베이스 백업 작업을 호출할 때마다 자동으로 요약 정보로 갱신됩니다. 이 파일은 데이터베이스 구성 파일과 같은 디렉토리에서 작성됩니다.

더 이상 필요하지 않은 이전 백업 이미지를 삭제하려면 백업이 파일로 저장된 경우 파일을 제거하면 됩니다. 그 뒤로 BACKUP 옵션과 함께 LIST HISTORY 명령을 실행하면 삭제된 백업 이미지에 대한 정보도 리턴됩니다. 복구 실행기록 파일에서 해당 항목을 제거하려면 PRUNE 명령을 사용해야 합니다.

복구 오브젝트가 TSM(Tivoli Storage Manager)을 사용하여 저장된 경우, db2adutl 유틸리티를 사용하여 복구 오브젝트를 조회, 추출, 검증 및 삭제할 수 있습니다. Linux 및 UNIX에서, 이 유틸리티는 sqllib/adsm 디렉토리에 있고 Windows 운영 체제에서는 sqllib₩bin에 있습니다.

모든 운영 체제에서, 디스크에서 작성된 백업 이미지의 파일 이름은 다음과 같은 몇 가지 요소의 병합(마침표로 분리됨)으로 구성됩니다.

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

예를 들어, 다음과 같습니다.

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

주: DB2 Universal Database, 버전 8.2.2 및 이전 버전에서는 Windows 운영 체제에서 백업 이미지를 저장할 때 4 레벨 서브디렉토리 트리가 사용되었습니다.

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

예를 들어, 다음과 같습니다.

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

데이터베이스 별명
백업 유틸리티가 호출될 때 지정된 1 - 8자의 데이터베이스 별명 이름.

유형 백업 작업의 유형. 0은 전체 데이터베이스 레벨 백업을 나타내고, 3은 테이블스페이스 레벨 백업을, 4는 LOAD...COPY TO 명령에 의해 생성된 백업 이미지를 나타냅니다.

인스턴스 이름
DB2INSTANCE 환경 변수에서 가져온 현재 인스턴스의 1 - 8자 이름.

노드 번호
데이터베이스 파티션 번호. 단일 파티션 데이터베이스 환경에서는 항상 NODE0000입니다. 파티션된 데이터베이스 환경에서는 NODE*xxxx*입니다. 여기서 *xxxx*는 db2nodes.cfg 파일에서 데이터베이스 파티션에 지정된 번호입니다.

카탈로그 파티션 번호
데이터베이스에 대한 카탈로그 파티션의 데이터베이스 파티션 번호. 단일 파티션 데이터베이스 환경에서는 항상 CATN0000입니다. 파티션된 데이터베이스 환경에서는 CATN*xxxx*입니다. 여기서 *xxxx*는 db2nodes.cfg 파일에서 데이터베이스 파티션에 지정된 번호입니다.

시간소인
백업 작업이 수행된 날짜 및 시간의 14자 표시입니다. 시간소인 양식은 *yyyymmddhhnnss*입니다.

- *yyyy*는 연도를 나타냅니다(1995 - 9999).
- *mm*은 월을 나타냅니다(01 - 12).
- *dd*는 일을 나타냅니다(01 - 31).
- *hh*는 시를 나타냅니다(00 - 23).
- *nn*은 분을 나타냅니다(00 - 59).
- *ss*는 초를 나타냅니다(00 - 59).

시퀀스 번호

파일 확장자로 사용되는 3자리 숫자입니다.

백업 이미지가 테이프에 기록되는 경우

- 파일 이름은 작성되지 않지만 위에 설명된 정보가 검증을 위해 백업 헤더에 저장됩니다.
- 테이프 디바이스는 표준 운영 체제 인터페이스를 통해 사용 가능해야 합니다. 그러나 대형 파티션된 데이터베이스 환경에서는 테이프 디바이스가 각 데이터베이스 파티션 서버 전용이 되도록 하는 것은 실용적이지 않습니다. 테이프 디바이스를 하나 이상의 TSM 서버에 연결하여, 이 테이프 디바이스에 대한 액세스가 각 데이터베이스 파티션 서버에 제공되도록 할 수 있습니다.
- 파티션된 데이터베이스 환경에서는 또한 REELlibrarian 4.2 또는 CLIO/S와 같은 가상 테이프 디바이스 기능을 제공하는 제품을 사용할 수도 있습니다. 이 제품을 사용하여 의사(pseudo) 테이프 디바이스를 통해 다른 노드(데이터베이스 파티션 서버)에 연결된 테이프 디바이스에 액세스할 수 있습니다. 리모트 테이프 디바이스에 대한 액세스는 투명하게 제공되므로, 의사 테이프 디바이스는 표준 운영 체제 인터페이스를 통해 액세스할 수 있습니다.

데이터베이스가 백업 보류 상태에 있는 경우를 제외하고, 사용할 수 없는 상태에 있는 데이터베이스를 백업할 수 없습니다. 테이블 스페이스가 비정상 상태에 있는 경우, 백업 보류 상태가 아니면 데이터베이스나 해당되는 테이블 스페이스를 백업할 수 없습니다.

동일한 테이블 스페이스에서의 동시 백업 작업은 허용되지 않습니다. 테이블 스페이스에서 백업 작업이 초기화되면 연속 시도는 실패합니다(SQL2048).

리스토어 작업 중에 시스템 손상이 발생하여 데이터베이스나 테이블 스페이스가 부분적으로 리스토어된 경우, 백업하기 전에 데이터베이스나 테이블 스페이스를 리스토어해야 합니다.

백업 작업은 백업할 테이블 스페이스 목록에 임시 테이블 스페이스의 이름이 포함된 경우 실패합니다.

백업 유틸리티는 서로 다른 데이터베이스의 백업 사본을 작성하는 여러 프로세스에 대해 동시처리 제어를 제공합니다. 이 동시처리 제어는 모든 백업 작업이 종료할 때까지 백업 목표 디바이스가 열려 있도록 합니다. 백업 작업 중에 오류가 발생하고 열린 컨테이너를 닫을 수 없는 경우, 동일한 드라이브를 목표로 하는 다른 백업 작업은 액세스 오류를 수신할 수 있습니다. 이와 같은 액세스 오류를 정정하려면 오류를 발생한 백업 작업을 종료하고 목표 디바이스로부터의 연결을 끊어야 합니다. 테이프로의 동시 백업 작업을 위해 백업 유틸리티를 사용하는 경우, 프로세스가 동일한 테이프를 목표로 하고 있지 않은지 확인하십시오.

백업 정보 표시

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

- 백업 이미지의 무결성을 테스트하고 이미지를 리스토어할 수 있는지 여부를 판별할 수 있습니다.
- 백업 헤더에 저장된 정보를 표시할 수 있습니다.
- 백업 이미지에 있는 로그 파일 헤더와 오브젝트에 대한 정보를 표시할 수 있습니다.

백업 사용

데이터베이스 데이터를 복사하고 실패 또는 원본에 대한 손상의 경우에 다른 매체에 저장하려면 BACKUP DATABASE 명령을 사용하십시오. 전체 데이터베이스, 데이터베이스 파티션 또는 선택된 테이블 스페이스만 백업할 수 있습니다.

시작하기 전에

백업될 데이터베이스에 연결될 필요는 없습니다. 데이터베이스 백업 유틸리티가 자동으로 지정된 데이터베이스에 연결하고 이 연결은 백업 작업이 완료될 때 종료됩니다. 백업될 데이터베이스에 연결된 경우 BACKUP DATABASE 명령이 실행될 때 연결이 끊어지며 백업 작업은 계속됩니다.

데이터베이스는 로컬 또는 리모트일 수 있습니다. Tivoli Storage Manager(TSM) 또는 DB2 ACS(Advanced Copy Services) 같은 스토리지 관리 제품을 사용 중이 아니면 백업 이미지는 데이터베이스 서버에 남아 있습니다.

오프라인 백업을 수행 중인 경우 및 ACTIVATE DATABASE문을 사용하여 데이터베이스를 활성화한 경우, 오프라인 백업을 실행하기 전에 데이터베이스를 비활성화해야 합니다. 데이터베이스에 대한 사용 중인 연결이 있는 경우 데이터베이스를 비활성화하려면 SYSADM 권한이 있는 사용자가 데이터베이스에 연결하고 다음 명령을 실행해야 합니다.

```
CONNECT TO database-alias
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE database-alias
```

파티션된 데이터베이스 환경에서는 BACKUP DATABASE 명령을 사용하여 데이터베이스 파티션을 개별적으로 백업하거나, ON DBPARTITIONNUM 명령 매개변수를 사용하여 한 번에 여러 데이터베이스 파티션을 백업하거나 ALL DBPARTITIONNUMS 매개변수를 사용하여 모든 데이터베이스 파티션을 동시에 백업할 수 있습니다. LIST NODES 명령을 사용하여 백업하려는 파티션에 사용자 테이블이 있는 데이터베이스 파티션을 식별할 수 있습니다.

단일 시스템 뷰(SSV) 백업을 사용 중이 아니면, 파티션된 데이터베이스 환경에서 *오프라인* 백업을 수행하려는 경우 다른 모든 데이터베이스 파티션에서 카탈로그 파티션을 별도로 백업해야 합니다. 예를 들어 카탈로그 파티션을 먼저 백업한 후 다른 데이터베이스 파티션을 백업할 수 있습니다. 이는 백업 작업에서 카탈로그 파티션에 대한 독점 데이터베이스 연결이 필요하여 이 동안에는 다른 데이터베이스 파티션이 연결할 수 없기 때문에 필요합니다. *온라인* 백업을 수행 중인 경우 모든 데이터베이스 파티션(카탈로그 파티션 포함)을 동시에 또는 임의의 순서로 백업할 수 있습니다.

또한 명령 편집기를 사용하여 데이터베이스 파티션을 백업할 수도 있습니다. 이 방식은 롤 포워드 복구를 지원하지 않기 때문에 이러한 노드에 있는 데이터베이스를 정기적으로 백업해야 합니다. db2nodes.cfg 파일에 대한 가능한 손상에 대한 보호 장치로서, 작성하는 모든 백업 사본과 함께 이 파일의 사본을 보존해야 합니다.

분산 요청(DR) 시스템에서 백업 작업은 분산 요청(DR) 데이터베이스 및 데이터베이스 카탈로그에 저장되는 메타데이터(랩퍼, 서버, 별명 등)에 적용됩니다. 데이터 소스 오브젝트(테이블 및 뷰)는 분산 요청(DR) 데이터베이스에 저장되지 않으면 백업되지 않습니다.

데이터베이스가 데이터베이스 관리 프로그램의 이전 릴리스를 사용하여 작성되었고 데이터베이스가 업그레이드되지 않은 경우 데이터베이스를 백업하기 전에 업그레이드해야 합니다.

이 태스크에 대한 정보

다음 제한사항이 백업 유틸리티에 적용됩니다.

- 테이블 스페이스 백업 작업 및 테이블 스페이스 리스토어 작업은 서로 다른 테이블 스페이스가 관련되는 경우에도 동시에 실행될 수 없습니다.
- 파티션된 데이터베이스 환경에서 롤 포워드 복구를 수행할 수 있기 원하는 경우, 노드 목록에 있는 데이터베이스를 정기적으로 백업해야 하며 시스템에 있는 나머지 노드(해당 데이터베이스에 대한 사용자 데이터를 포함하지 않는 것까지도)의 백업 이미지가 최소한 하나 있어야 합니다. 두 상황은 데이터베이스에 대한 사용자 데이터를 포함하지 않는 데이터베이스 파티션 서버에서 데이터베이스 파티션의 백업 이미지가 필요합니다.
  - 마지막 백업을 작성한 후 데이터베이스 시스템에 데이터베이스 파티션 서버를 추가했고, 이 데이터베이스 파티션 서버에서 포워드 복구를 수행해야 합니다.
  - 특정 시점 복구가 사용되는데, 이 복구는 시스템의 모든 데이터베이스 파티션이 롤 포워드 보류 상태에 있어야 합니다.
- DMS 테이블 스페이스의 온라인 백업 작업은 다음 작업과 호환되지 않습니다.
  - 로드
  - 재구성(온라인 및 오프라인)

  - 테이블 스페이스 삭제
  - 테이블 절단
  - 인덱스 작성
  - 처음에 로그되지 않음(CREATE TABLE 및 ALTER TABLE문과 함께 사용됨)
- 현재 사용 중인 데이터베이스의 오프라인 백업을 수행하려고 시도하면 오류가 수신됩니다. 오프라인 백업을 실행하기 전에 DEACTIVATE DATABASE 명령을 실행하여 데이터베이스가 활성이 아닌지 확인할 수 있습니다.

명령행 처리기(CLP), 제어 센터의 데이터베이스 백업 마법사를 통해서, BACKUP DATABASE 매개변수가 있는 ADMIN_CMD 프로시저나 *db2Backup* API를 실행하여 백업 유틸리티를 호출할 수 있습니다.

다음은 CLP를 통해 실행되는 BACKUP DATABASE 명령의 예입니다.

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

프로시저

데이터베이스 백업 마법사를 열려면 다음을 수행하십시오.

1. 제어 센터에서 백업하려는 데이터베이스나 테이블 스페이스 오브젝트를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. 오브젝트를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 백업을 선택하십시오. 데이터베이스 백업 마법사가 열립니다. .

다음 단계

제어 센터에 있는 문맥 도움말 기능을 통해 세부사항 정보가 제공됩니다.

오프라인 백업을 수행한 경우, 백업이 완료된 후 데이터베이스를 재활성화해야 합니다.

```
ACTIVATE DATABASE sample
```

스냅샷 백업 수행

스냅샷 백업 조작은 스토리지 디바이스의 빠른 복사 기술을 사용하여 백업의 데이터 복사 부분을 수행합니다.

시작하기 전에

스냅샷 백업 및 리스토어 작업을 수행하려면 스토리지 디바이스에 대해 DB2 ACS API 드라이버가 필요합니다. IBM Data Server에 통합된 드라이버는 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage SAN
  Volume Controller
- IBM System Storage DS6000

- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS 시리즈

스냅샷 백업을 수행하려면 DB2 ACS(Advanced Copy Services)를 사용 가능하게 해야 합니다. 345 페이지의 『DB2 ACS(Advanced Copy Services) 사용』을 참조하십시오.

프로시저

USE SNAPSHOT 매개변수와 함께 BACKUP DATABASE 명령, SQLU_SNAPSHOT_MEDIA 미디어 유형과 함께 db2Backup API, 또는 **BACKUP DATABASE** 및 **USE SNAPSHOT** 매개변수와 함께 ADMIN_CMD 프로시저를 사용하여 스냅샷 백업을 수행할 수 있습니다.

BACKUP DATABASE 명령:

```
db2 backup db sample use snapshot
```

BACKUP DATABASE 매개변수를 사용한 ADMIN_CMD 프로시저:

```
CALL SYSPROC.ADMIN_CMD
   ('backup db sample use snapshot')
```

db2Backup API

```
int sampleBackupFunction( char dbAlias[],
                          char user[],
                          char pswd[],
                          char workingPath[] )
{
  db2MediaListStruct mediaListStruct = { 0 };

  mediaListStruct.locations = &workingPath;
  mediaListStruct.numLocations = 1;
  mediaListStruct.locationType = SQLU_SNAPSHOT_MEDIA;


  db2BackupStruct backupStruct = { 0 };

  backupStruct.piDBAlias = dbAlias;
  backupStruct.piUsername = user;
  backupStruct.piPassword = pswd;
  backupStruct.piVendorOptions = NULL;
  backupStruct.piMediaList = &mediaListStruct;

  db2Backup(db2Version950, &backupStruct, &sqlca);

  return 0;
}
```

결과

스냅샷 백업을 수행한 후 스냅샷 백업에서 리스토어할 수 있습니다.

분할 미러를 백업 이미지로 사용

백업 이미지로 사용하기 위해 기본 데이터베이스의 분할 미러를 작성하려면 다음 프로시저를 사용하십시오. 이 프로시저는 기본 데이터베이스에 대해 데이터베이스 백업 조작을 수행하는 대신 사용될 수 있습니다.

분할 미러를 『백업 이미지』로 사용하려면 다음 단계를 수행하십시오.

1. 기본 데이터베이스의 입출력을 일시중단하십시오.

   ```
   db2 set write suspend for database
   ```

   데이터베이스가 일시중단된 상태에서 다른 유틸리티나 도구를 실행 중이 아니어야 합니다. 오직 데이터베이스의 사본을 작성 중이어야 합니다.
2. 적합한 운영 체제 레벨 명령을 사용하여 기본 데이터베이스에서 미러를 분할하십시오.

   주: 볼륨 디렉토리를 포함한 전체 데이터베이스 디렉토리를 복사해야 합니다. 또한 로그 디렉토리 및 데이터베이스 디렉토리 밖에 존재하는 모든 컨테이너 디렉토리를 복사해야 합니다. 이 정보를 알려면 분할되어야 하는 데이터베이스의 모든 파일과 디렉토리를 표시하는 DBPATHS 관리 뷰를 참조하십시오.
3. 기본 데이터베이스에서 입출력을 재개하십시오.

   ```
   db2 set write resume for database
   ```
4. 기본 시스템에서 실패가 발생하고, 백업에서의 리스토어가 필요합니다.
5. 기본 데이터베이스 인스턴스를 중지하십시오.

   ```
   db2stop
   ```
6. 운영 체제 레벨 명령을 사용하여 분할 데이터를 기본 시스템에 복사하십시오. 1차 로그가 롤 포워드 복구를 위해 필요하기 때문에 **분할 로그 파일을 복사하지 마십시오.**
7. 기본 데이터베이스 인스턴스를 시작하십시오.

   ```
   db2start
   ```
8. 기본 데이터베이스를 초기화하십시오.

   ```
   db2inidb database_alias as mirror
   ```
9. 로그의 끝까지 또는 특정 시점으로 기본 데이터베이스를 롤 포워드하고 중지하십시오.

테이프로 백업

데이터베이스나 테이블 스페이스를 백업할 때 블록 크기와 버퍼 크기를 올바르게 설정해야 합니다. 가변 블록 크기를 사용하는 경우에 특히 그러합니다(예를 들어, AIX에서 블록 크기가 0으로 설정된 경우)

백업할 때 사용할 수 있는 고정 블록 *크기* 수에는 제한이 있습니다. 이 제한은 DB2가 백업 이미지 헤더를 4-KB 블록으로 작성하기 때문입니다. DB2가 지원하는 고정 블록 *크기*는 512, 1024, 2048 및 4096바이트입니다. 고정 블록 *크기*를 사용 중인 경우 백업 버퍼 *크기*를 지정할 수 있습니다. 그러나 고정 블록 *크기*가 DB2에서 지원되는 *크기* 중 하나가 아니면 백업 작업이 성공적으로 완료되지 않을 것을 알 수 있습니다.

데이터베이스가 큰 경우 고정 블록 *크기*를 사용하는 것은 백업 작업이 완료하는 데 많은 시간이 소요될 수 있음을 의미합니다. 가변 블록 *크기*를 사용할 것을 고려할 수도 있습니다.

주: 가변 블록 *크기*의 사용은 현재 지원되지 *않습니다*. 이 옵션을 사용해야 하는 경우, 가변 블록 *크기*로 작성된 백업 이미지를 사용하여 성공적으로 복구할 수 있는 제대로 테스트된 프로시저가 있는지 확인하십시오.

가변 블록 *크기* 사용 시 사용 중인 테이프 디바이스에 대해 최대 한계 이하의 백업 버퍼 *크기*를 지정해야 합니다. 최적 성능을 위해서는 버퍼 *크기*를 사용 중인 디바이스의 최대 블록 *크기* 한계와 동일하게 지정해야 합니다.

Windows 운영 체제에서 테이프 디바이스를 사용하려면 먼저 다음 명령을 발행해야 합니다.

```
db2 initialize tape on <device> using <blksize>
```

여기서,

<device>

유효한 테이프 디바이스 이름입니다. Windows 운영 체제의 디폴트는 \\.\TAPE0입니다.

<blksize>

테이프의 블로킹 인수입니다. 하나의 인수이거나 4096의 배수여야 합니다. 디폴트값은 디바이스의 디폴트 블록 *크기*입니다.

가변 블록 *크기*의 백업 이미지에서 리스토어하면 오류가 리턴될 수 있습니다. 이러한 상황이 발생하면 적절한 블록 *크기*를 사용하여 이미지를 다시 작성해야 할 수도 있습니다. 다음은 AIX에서의 예입니다.

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

백업 이미지가 backup_filename.file 파일로 덤프됩니다. dd 명령은 4096 블록 *크기*를 사용하여 이미지를 다시 테이프로 덤프합니다.

파일로 덤프하기에는 이미지가 너무 큰 경우 이 접근방식에는 문제점이 있습니다. 가능한 한 가지 솔루션은 dd 명령을 사용하여 하나의 테이프 디바이스에서 다른 테이프 디

바이스로 덤프하는 것입니다. 이 방법은 이미지가 여러 테이프에 분산되어 있지 않으면 작동합니다. 두 개의 테이프 디바이스를 사용할 때 dd 명령은 다음과 같습니다.

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

두 개의 테이프 디바이스를 사용하는 것이 불가능하면 dd 명령을 사용하여 원시 디바이스로 이미지를 덤프한 후 원시 디바이스에서 테이프로 이미지를 덤프할 수 있습니다. 이 접근방식의 문제점은 dd 명령이 원시 디바이스로 덤프되는 블록 수를 *반드시* 추적해야 하는 것입니다. 이 수는 이미지가 다시 테이프로 이동될 때 지정해야 합니다. dd 명령을 사용하여 원시 디바이스에서 테이프로 이미지를 덤프하는 경우 명령은 원시 디바이스의 전체 컨텐츠를 테이프로 덤프합니다. dd 유틸리티는 이미지를 보유하기 위해 사용되는 원시 디바이스 양을 판별할 수 없습니다.

백업 유틸리티를 사용할 때 사용하는 테이프 디바이스의 최대 블록 *크기*를 알아야 합니다. 다음은 일부 예입니다.

디바이스	첨부	블록 *크기* 한계	**DB2** 버퍼 *크기* 한계 **(4KB** 페이지 단위**)**
8mm	scsi	131,072	32
3420	s370	65,536	16
3480	s370	61 440	15
3490	s370	61 440	15
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 단지 제안 값입니다. 블록 *크기* 한계는 상위 어댑터에 의해 부과됩니다.
2. 3590이 2MB 블록 *크기*를 지원하지만 성능이 사용자 필요에 적합하면 낮은 값(256KB와 같은)으로 시도할 수 있습니다.
3. 디바이스 한계에 대한 정보는 디바이스 문서를 점검하거나 디바이스 벤더에 문의하십시오.

테이프 디바이스의 호환성 검증

UNIX, Linux 및 AIX 운영 체제에서만, 테이프 디바이스가 DB2 데이터베이스 백업에 지원되는지 판별하려면 다음 프로시저를 수행하십시오.

데이터베이스 관리 프로그램 인스턴스 소유자로서, 운영 체제 명령 dd를 실행하여 테이프 디바이스에서 읽거나 테이프 디바이스에 쓰십시오. dd 명령이 성공하면 테이프 디바

이스를 사용하여 DB2 데이터베이스를 백업할 수 있습니다.

Named Pipes에 백업

UNIX 기반 시스템의 로컬 Named Pipes로의 데이터베이스 백업(및 Named Pipes로부터의 데이터베이스 리스토어)에 대한 지원을 사용할 수 있습니다.

Named Pipe의 기록기와 판독기 모두가 동일한 머신에 있어야 합니다. 파이프가 존재하고 로컬 파일 시스템에 위치해야 합니다. Named Pipe가 로컬 디바이스로 취급되기 때문에 목표가 Named Pipe임을 지정할 필요는 없습니다.

다음은 AIX 예입니다.

1. Named Pipe를 작성하십시오.

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

2. 이 백업 이미지가 리스토어 유틸리티에 의해 사용될 경우 리스토어 작업은 어떤 데이터도 누락되지 않도록 백업 작업 *전에* 호출되어야 합니다.

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

3. 이 파이프를 데이터베이스 백업 작업의 목표로 사용하십시오.

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

파티션된 데이터베이스 백업

파티션된 데이터베이스 환경에서의 데이터베이스 백업은 각 데이터베이스 파티션의 백업 성공 여부 추적, 다중 로그 파일 및 백업 이미지 관리, 모든 데이터베이스 파티션에 대한 로그 파일 및 백업 이미지가 데이터베이스를 리스토어하기 위해 필요한 최소 복구 시간 동안 유지되도록 확인 등과 같은 어려움을 내포할 수 있습니다. 단일 시스템 뷰(SSV) 백업 사용이 파티션된 데이터베이스를 백업하는 가장 쉬운 방법입니다.

파티션된 데이터베이스 환경에서 데이터베이스를 백업하는 세 가지 방법이 있습니다.

- BACKUP DATABASE 명령, ADMIN_CMD 프로시저와 함께 BACKUP DATABASE 명령 또는 db2Backup API 함수를 사용하여 한 번에 각 데이터베이스 파티션 백업
- BACKUP DATABASE 명령과 함께 db2_all 명령을 사용하여 사용자가 지정하는 모든 데이터베이스 파티션 백업
- 단일 시스템 뷰(SSV) 백업을 실행하여 데이터베이스 파티션의 전부 또는 일부를 동시에 백업

한 번에 하나씩 각 데이터베이스 파티션을 백업하는 것은 시간이 걸리고 오류가 발생하기 쉽습니다. 일반적으로 명령 호출을 한 번만 하면 되기 때문에 db2_all 명령을 사용하여 모든 파티션을 백업하는 것이 각 데이터베이스 파티션을 백업하는 것보다 쉽습니다. 그러나 db2_all을 사용하여 파티션된 데이터베이스를 백업할 때, 카탈로그를 포

함하는 데이터베이스 파티션은 비카탈로그 데이터베이스 파티션과 동시에 백업할 수 없기 때문에 때로는 db2_all을 여러 번 호출해야 합니다. 한 번에 하나씩 각 데이터베이스 파티션을 백업하거나 db2_all을 사용하거나 상관없이, 각 데이터베이스 파티션의 백업 이미지의 시간소인이 다르기 때문에 이들 방법 중 하나를 사용하여 작성된 백업 이미지 관리가 어려우며 데이터베이스 파티션의 백업 이미지 사이에 최소 복구 시간을 조정하는 것도 어렵습니다.

위에서 언급한 이유 때문에, 파티션된 데이터베이스 환경에서 데이터베이스를 백업하는 권장 방법은 SSV 백업을 사용하는 것입니다.

SSV 백업을 사용하여 파티션된 데이터베이스의 모든 데이터베이스 파티션 또는 일부를 백업하려면 다음을 수행하십시오.

1. 선택사항: 데이터베이스를 온라인 상태로 유지하거나, 데이터베이스를 오프라인으로 만드십시오.

   데이터베이스가 온라인 또는 오프라인인 중에 파티션된 데이터베이스를 백업할 수 있습니다. 데이터베이스가 온라인인 경우 백업 유틸리티는 다른 데이터베이스 파티션에 대한 공유 연결을 획득하므로, 데이터베이스 파티션이 백업되고 있는 중에 사용자 응용프로그램이 해당 파티션에 연결할 수 있습니다.
2. 데이터베이스 카탈로그를 포함하는 데이터베이스 파티션에서 파티션된 데이터베이스에 대한 적합한 매개변수와 함께 백업을 호출하십시오.
   - ON DBPARTITIONNUMS 매개변수와 함께 BACKUP DATABASE 명령을 호출할 수 있습니다.
   - ADMIN_CMD 프로시저를 사용하여 ON DBPARTITIONNUMS 매개변수와 함께 BACKUP DATABASE 명령을 호출할 수 있습니다.
   - iAllNodeFlag 매개변수와 함께 db2Backup API를 호출할 수 있습니다.
3. 선택사항: 백업 이미지에 복구에 필요한 로그 파일을 포함하십시오.

   디폴트로, SSV 백업을 수행 중인 경우(즉, ON DBPARTITIONNUM 매개변수를 지정하는 경우) 로그 파일이 백업 이미지에 포함됩니다. 로그 파일이 백업 이미지에 포함되기를 원하지 않는 경우 백업을 실행할 때 EXCLUDE LOGS 명령 매개변수를 사용하십시오. 비SSV 백업의 경우 디폴트로 로그 파일이 백업 이미지에서 제외됩니다.

   자세한 정보는 166 페이지의 『백업 이미지와 함께 로그 파일 포함』 주제를 참조하십시오.
4. 선택사항: 이전 백업 이미지를 삭제하십시오. 이전 백업 이미지를 삭제하는 데 사용하는 메소드는 백업 이미지를 저장하는 방법에 따라 다릅니다. 예를 들어 백업 이미지를 디스크에 저장하는 경우 파일을 삭제할 수 있습니다. Tivoli storage manager를 사용하여 백업 이미지를 저장하는 경우 db2adutl 유틸리티를 사용하여 백업 이

미지를 삭제할 수 있습니다. DB2 ACS(Advanced Copy Services)를 사용 중인 경우에는 db2acsutil을 사용하여 스냅샷 백업 오브젝트를 삭제할 수 있습니다.

IBM Tivoli Space Manager 계층 스토리지 관리를 사용하여 파티션된 테이블 백업

Tivoli Space Manager HSM(Hierarchical Storage Manager) 클라이언트 프로그램은 자동으로 적합한 파일을 보조 스토리지로 이주하여 로컬 파일 시스템에서 특정 레벨의 여유 공간을 유지보수합니다.

테이블 파티셔닝으로, 테이블 데이터는 데이터 파티션이라고 하는 여러 스토리지 오브젝트에 나눠집니다. HSM은 개별 데이터 파티션을 보조 스토리지로 백업하는 것을 지원합니다.

SMS 테이블 스페이스를 사용할 때, 각 데이터 파티션 범위는 해당되는 디렉토리에서 하나의 파일로 표시됩니다. 따라서 데이터의 개별 범위(데이터 파티션)를 보조 스토리지로 이주하는 것은 아주 쉽습니다.

DMS 테이블 스페이스를 사용할 때, 각 컨테이너는 하나의 파일로 표시됩니다. 이 경우, 자주 액세스하지 않는 범위는 자체에 소유된 테이블 스페이스에 저장해야 합니다. EVERY 절을 사용하여 CREATE TABLE 문을 발행할 때, NO CYCLE 절을 사용하여 테이블 레벨 IN 절에 나열된 테이블 스페이스 수가 작성되는 데이터 파티션 수와 일치하는지 확인하십시오. 다음 예에 나와 있습니다.

예 1

```
CREATE TABLE t1 (c INT) IN tbsp1, tbsp2, tbsp3 NO CYCLE
  PARTITION BY RANGE(c)
  (STARTING FROM 2 ENDING ( 6 EVERY 2);
```

자동 백업 사용

데이터베이스는 다양한 하드웨어 또는 소프트웨어 실패로 인해 사용 불가능하게 될 수 있습니다. 데이터베이스에 대한 최신 전체 백업이 있는지 확인하는 것이 시스템에 대한 재해 복구 전략 계획 및 구현의 필수적인 부분입니다. DB2가 데이터베이스를 적절하면서도 정기적으로 백업할 수 있도록 하려면 재해 복구 전략의 일부로 자동 데이터베이스 백업을 사용하십시오.

그래프 형식 사용자 인터페이스 도구, 명령 인터페이스 또는 AUTOMAINT_SET_POLICY 시스템 스토어드 프로시저를 사용하여 자동 백업을 구성할 수 있습니다.

- 그래프 형식 사용자 인터페이스 도구를 사용하여 자동 백업을 구성하려면 다음을 수행하십시오.

1. 제어 센터에서 데이터베이스 오브젝트를 마우스 오른쪽 단추로 누르거나 Health Center에서 자동 백업을 구성하려는 데이터베이스 인스턴스를 마우스 오른쪽 단추로 눌러서 자동 유지보수 구성 마법사를 여십시오. 팝업 창에서 **자동 유지보수 구성**을 선택하십시오.
2. 이 마법사에서 자동 백업을 사용 가능하게 하고 BACKUP 유틸리티의 실행을 위한 유지보수 기간을 지정할 수 있습니다.

- 명령 인터페이스를 사용하여 자동 백업을 구성하려면 다음의 각 구성 매개변수를 ON으로 설정하십시오.
  - AUTO_MAINT
  - AUTO_DB_BACKUP
- AUTOMAINT_SET_POLICY 시스템 스토어드 프로시저를 사용하여 자동 백업을 구성하려면 다음을 수행하십시오.
  1. 백업 미디어, 백업이 온라인 또는 오프라인인지 여부 및 백업 빈도 같은 세부사항을 지정하는 구성 XML 입력을 작성하십시오.

     SQLLIB/samples/automaintcfg 디렉토리에 있는 DB2DefaultAutoBackupPolicy.xml이라는 샘플 파일의 컨텐츠를 복사하고 구성 요구사항을 만족하도록 XML을 수정하십시오.
  2. 선택사항: 구성 XML 입력을 포함하는 XML 입력 파일을 작성하십시오.
  3. 다음 매개변수를 사용하여 AUTOMAINT_SET_POLICY를 호출하십시오.
     - 유지보수 유형: AutoBackup
     - 구성 XML 입력: 구성 XML 입력 텍스트를 포함하는 BLOB 또는 구성 XML 입력을 포함하는 파일의 이름.

  AUTOMAINT_SET_POLICY 시스템 스토어드 프로시저 사용에 대한 자세한 정보는 64 페이지의 『SYSPROC.AUTOMAINT_SET_POLICY 또는 SYSPROC.AUTOMAINT_SET_POLICYFILE을 사용한 자동화된 유지보수 규정 구성』 주제를 참조하십시오.

자동 데이터베이스 백업

데이터베이스는 광범위하고 다양한 하드웨어 또는 소프트웨어 실패로 인해 사용 불가능하게 될 수 있습니다. 자동 데이터베이스 백업은 항상 최근의 전체 데이터베이스 백업이 필요에 따라 수행되도록 하여 DBA에 대한 데이터베이스 백업 관리 태스크를 단순화합니다. 다음 측정 중 하나 이상을 기초로 백업 작업을 수행해야 하는지 판별합니다.

- 전체 데이터베이스 백업을 완료한 적이 없습니다.
- 마지막 전체 백업 이후 경과 시간이 지정된 시간 수보다 깁니다.

- 마지막 백업 이후 소비된 트랜잭션 로그 공간이 지정된 4KB 페이지 수보다 큽니다 (아카이브 로깅 모드에서만).

사용자 시스템에 맞는 재해 복구 전략을 계획하고 구현하여 데이터를 보호하십시오. 요구사항에 맞으면 자동 데이터베이스 백업 기능을 백업 및 복구 전략의 일부로서 통합할 수 있습니다.

롤 포워드 복구에 대해 데이터베이스가 사용 가능하면(아카이브 로깅), 자동 데이터베이스 백업을 온라인 또는 오프라인 백업에 사용할 수 있습니다. 그렇지 않으면 오프라인 백업만 사용 가능합니다. 자동 데이터베이스 백업은 디스크, 테이프, TSM(Tivoli Storage Manager) 및 벤더 DLL 미디어 유형을 지원합니다.

제어 센터나 Health Center의 자동 유지보수 구성 마법사를 통해 다음을 구성할 수 있습니다.

- 백업 사이에 요청된 시간 또는 로그 페이지 수
- 백업 미디어
- 온라인 또는 오프라인 백업 여부

디스크로의 백업을 선택하면, 자동 백업 기능은 자동 유지보수 구성 마법사에 지정된 디렉토리에서 정기적으로 백업 이미지를 삭제합니다. 가장 최근의 백업 이미지만 제공된 시간에 사용 가능합니다. 이 디렉토리는 자동 백업 기능을 위해 별도로 유지하고 다른 백업 이미지를 저장하는데 사용하지 않는 것이 좋습니다.

자동 데이터베이스 백업 기능은 auto_db_backup 및 auto_maint 데이터베이스 구성 매개변수를 사용하여 사용 가능 또는 사용 불가능하도록 설정할 수 있습니다. 파티션된 데이터베이스 환경의 경우, 데이터베이스 구성 매개변수가 해당 데이터베이스 파티션에서 사용 가능하면 각 데이터베이스 파티션에서 자동 데이터베이스 백업이 실행됩니다.

또한 AUTOMAINT_SET_POLICY 및 AUTOMAINT_SET_POLICYFILE이라고 하는 시스템 스토어드 프로시저 중 하나를 사용하여 자동 백업을 구성할 수도 있습니다.

백업 성능 최적화

백업 작업을 수행할 때, DB2는 버퍼 수, 버퍼 크기 및 병렬 처리 설정에 최적의 값을 자동으로 선택합니다. 값은 사용 가능한 유틸리티 힙 메모리 양, 사용 가능한 프로세서 수 및 데이터베이스 구성을 기초로 합니다. 따라서 시스템에서 사용 가능한 스토리지 양에 따라 UTIL_HEAP_SZ 구성 매개변수를 늘려서 추가 메모리를 할당할 것을 고려해야 합니다. 목적은 백업 작업을 완료하는 데 소비되는 시간을 최소화하는 것입니다. 다음 BACKUP DATABASE 명령 매개변수의 값을 명시적으로 입력하지 않으면 DB2가 값을 선택합니다.

- WITH num-buffers BUFFERS
- PARALLELISM n

- BUFFER buffer-size

버퍼 수와 버퍼 *크기*를 지정하지 않아서 DB2에서 값을 설정하는 경우, 대형 데이터베이스에서는 값이 최소한의 영향을 주어야 합니다. 그러나 작은 데이터베이스에서는 백업 이미지 *크기*에서 백분율 증가가 클 수 있습니다. 디스크에 기록된 마지막 데이터 버퍼에 약간의 데이터가 포함되어 있어도, 전체 버퍼가 이미지에 기록됩니다. 작은 데이터베이스에서는 이미지 *크기*의 상당히 많은 백분율이 비어 있을 수 있음을 의미합니다.

또한 백업 작업을 완료하는 데 필요한 시간을 줄이려면 다음 중 하나를 수행할 것을 선택할 수 있습니다.

- 테이블 스페이스 백업을 지정하십시오.

  BACKUP DATABASE 명령에서 TABLESPACE 옵션을 사용하여 데이터베이스의 일부를 백업(및 연속으로 복구)할 수 있습니다. 이를 통해 별도의 테이블 스페이스에서 테이블 데이터, 인덱스 및 긴 필드나 대형 오브젝트(LOB) 데이터 관리를 용이하게 할 수 있습니다.
- 백업하는 테이블 스페이스 수를 반영하도록 BACKUP DATABASE 명령에서 PARALLELISM 매개변수의 값을 늘리십시오.

  PARALLELISM 매개변수는 데이터베이스에서 데이터를 읽고 압축된 백업 작업 중에 데이터를 압축하기 위해 시작된 스레드 또는 프로세스 수를 정의합니다. 각 프로세스 또는 스레드는 특정 테이블 스페이스에 지정되므로, PARALLELISM 매개변수에 대해 백업되는 테이블 스페이스 수보다 큰 값을 지정하면 이득이 없습니다. 해당 테이블 스페이스의 백업을 완료하면 다른 것을 요청합니다. 그러나 각 프로세스 또는 스레드에는 메모리 및 CPU 오버헤드 모두 필요합니다.
- 백업 버퍼 *크기*를 늘리십시오.

  이상적인 백업 버퍼 *크기*는 테이블 스페이스 Extent *크기* 배수에 1페이지를 더한 것입니다. Extent *크기*가 다른 여러 개의 테이블 스페이스를 가지고 있으면 일반적인 Extent *크기* 배수에 1페이지를 더한 값을 지정하십시오.
- 버퍼 수를 늘리십시오.

  백업 목표(또는 세션) 수만큼의 버퍼 수의 최소 두 배를 사용하여 백업 목표 디바이스가 데이터를 기다리지 않도록 하십시오.
- 여러 개의 목표 디바이스를 사용하십시오.

백업을 사용하는 데 필요한 특권, 권한 및 권한 부여

사용자는 특권을 사용하여 데이터베이스 자원을 작성하거나 자원에 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작과 특권을 그룹화하는 방법을 제공합니다. 이를 통해 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다. 사용자는 적절한 권한(즉, 필요한 특권 또는 권한)을 가지고 있는 오브젝트에만 액세스할 수 있습니다.

백업 유틸리티를 사용하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한을 가지고 있어야 합니다.

온라인 백업 및 기타 유틸리티의 호환성

일부 유틸리티는 온라인 백업과 동시에 실행할 수 있지만 다른 유틸리티는 동시에 실행할 수 없습니다.

다음 유틸리티는 온라인 백업과 호환 가능합니다.

- EXPORT
- ONLINE INSPECT

다음 유틸리티는 특정 상황에서만 온라인 백업과 호환 가능합니다.

- ONLINE CREATE INDEX

  SMS 모드에서, 온라인 인덱스 작성 및 온라인 백업은 ALTER TABLE 잠금으로 인해 호환 가능하지 않습니다. 온라인 인덱스 작성은 독점 모드에서 잠금을 획득하고 온라인 백업은 공유 모드에서 획득합니다.

  DMS 모드에서, 온라인 인덱스 작성 및 온라인 백업은 대부분의 경우에 동시에 실행할 수 있습니다. 인덱스 수가 많은 경우, 온라인 인덱스 작성에서 동시 온라인 백업과 충돌할 온라인 백업 잠금을 내부적으로 획득하게 될 가능성이 있습니다.

- ONLINE INDEX REORG

  온라인 인덱스 작성에서와 같이, SMS 모드에서 온라인 인덱스 재구성은 ALTER TABLE 잠금으로 인해 온라인 백업과 호환 가능하지 않습니다. 온라인 인덱스 재구성은 독점 모드에서 잠금을 획득하고 온라인 백업은 공유 모드에서 획득합니다. 또한 온라인 인덱스 재구성 작업은 전환 단계 이전에 테이블을 Quiesce하고 온라인 백업을 금지하는 Z 잠금을 획득합니다. 그러나 ALTER TABLE 잠금은 Z 테이블 잠금이 획득되기 전에 온라인 백업이 동시에 실행되지 않도록 방지해야 합니다.

  DMS 모드에서, 온라인 인덱스 재구성 및 온라인 백업은 동시에 실행할 수 있습니다.

또한 온라인 인덱스 재구성은 전환 단계 이전에 테이블을 Quiesce하고 온라인 백업을 금지하는 Z 잠금을 획득합니다.

- REBALANCE

  온라인 백업 및 재조정 프로그램이 동시에 실행 중인 경우 온라인 백업은 재조정 프로그램을 일시정지하고 완료하기를 기다리지 않습니다.

- IMPORT

  임포트 유틸리티는 IMPORT 명령을 REPLACE 옵션과 함께 발행하는 경우를 제외하고 온라인 백업과 호환 가능합니다. 이 경우, 임포트는 테이블에서 Z 잠금을 확보하고 온라인 백업이 동시에 실행되지 않도록 합니다.

- ALLOW READ ACCESS LOAD

  ALLOW READ ACCESS 로드 작업은 LOAD 명령이 COPY NO 옵션과 함께 발행될 때 온라인 백업과 호환 가능하지 않습니다. 이 모드에서 두 유틸리티는 모두 테이블 스페이스 상태를 수정하므로, 유틸리티 중 하나가 오류를 보고합니다.

  ALLOW READ ACCESS 로드 작업은 LOAD 명령이 COPY YES 옵션과 함께 발행될 때 일부 호환성 문제가 있지만 온라인 백업과 호환 가능합니다. SMS 모드에서, 유틸리티는 동시에 실행할 수 있지만 호환되지 않는 테이블 잠금 모드를 보유하므로 결국 테이블 잠금 대기 대상이 될 수 있습니다. DMS 모드에서, 두 유틸리티는 모두 호환되지 않는 ″Internal-B″(OLB) 잠금 모두를 보유하므로 해당 잠금에 대해 대기 대상이 될 수 있습니다. 유틸리티가 동일한 테이블 스페이스에서 동시에 실행되면, 로드 유틸리티는 진행하기 전에 백업 유틸리티가 테이블 스페이스 처리를 완료하기를 강제로 기다리게 될 수도 있습니다.

- ONLINE TABLE REORG

  온라인 테이블 재구성의 정리 단계는 온라인 백업이 실행 중일 때 시작할 수 없습니다. 필요하면 테이블 재구성을 일시정지하여 온라인 테이블 재구성을 다시 시작하기 전에 온라인 백업이 완료되도록 할 수 있습니다.

  동일한 테이블 스페이스 내의 테이블이 온라인에서 재구성되는 경우 DMS 테이블 스페이스의 온라인 백업을 시작할 수 있습니다. 절단 단계에서 재구성 작업과 연관되는 잠금 대기가 있을 수 있습니다.

  동일한 테이블 스페이스 내의 테이블이 온라인에서 재구성되는 경우 SMS 테이블 스페이스의 온라인 백업을 시작할 수 없습니다. 두 작업 모두에서 배타적 잠금이 필요합니다.

- Z 잠금이 필요한 DDL(예: ALTER TABLE, DROP TABLE 및 DROP INDEX)

  온라인 DMS 테이블 스페이스 백업은 Z 잠금이 필요한 DDL과 호환 가능합니다.

온라인 SMS 테이블 스페이스 백업은 Z 잠금이 해제되기를 기다려야 합니다.

- RUNSTATS(쓰기 허용 및 읽기 허용)

  Runstats는 시스템 카탈로그 테이블 스페이스가 SMS 테이블 스페이스인 경우를 제외하고 온라인 백업과 호환 가능합니다. 이는 SMS 테이블 스페이스에 SYSIBM.SYSTABLES가 있는 경우 runstats 및 온라인 백업이 해당 테이블에 대해 호환되지 않는 테이블 잠금을 보유하게 되어, 잠금 대기가 발생하도록 합니다.

다음 유틸리티는 온라인 백업과 호환 가능하지 않습니다.

- REORG TABLE
- RESTORE
- ROLLFORWARD
- ONLINE BACKUP
- ALLOW NO ACCESS LOAD
- SET WRITE

백업 예

예 1

다음 예에서 데이터베이스 SAMPLE은 두 개의 동시 TSM 클라이언트 세션을 사용하여 TSM 서버로 백업됩니다. 백업 유틸리티는 최적의 버퍼 수를 계산합니다. 버퍼의 최적 크기(4KB 페이지 수)는 사용 가능한 목표 디바이스 수와 메모리 양을 기초로 계산됩니다. 병렬 처리 설정 역시 자동으로 계산되며 백업할 테이블 스페이스 수와 사용 가능한 프로세서 수를 기초로 합니다.

```
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
```

예 2

다음은 복구 가능한 데이터베이스의 주별 증분 백업 전략 샘플입니다. 주별 전체 데이터베이스 백업 조작과 일별 비누적(델타) 백업 조작 및 주중 누적(증분) 백업 조작이 포함됩니다.

```
(Sun) 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
```

예 3

Windows 환경에서 테이프 디바이스로의 백업 작업을 시작하려면 다음을 발행하십시오.

```
db2 backup database sample to \\.\tape0
```

제 12 장 복구 개요

복구 유틸리티는 필요한 리스토어 및 롤 포워드 작업을 사용하여 to recover a database to a specified time, based on information found in 복구 실행기록 파일에서 발견되는 정보를 기초로 지정된 시간까지 데이터베이스를 복구합니다. 이 유틸리티를 사용할 때 데이터베이스를 특정 시점이나 로그 파일 끝까지 복구할 것을 지정합니다. 유틸리티는 가장 적합한 백업 이미지를 선택하여 복구 작업을 수행합니다.

복구 유틸리티는 다음의 RESTORE DATABASE 명령 옵션을 지원하지 않습니다.

- TABLESPACE tablespace-name. 테이블 스페이스 리스토어 작업은 지원되지 않습니다.
- INCREMENTAL. 증분 리스토어 작업은 지원되지 않습니다.
- OPEN num-sessions SESSIONS. TSM 또는 다른 벤더 제품과 함께 사용될 입출력 세션 수를 표시할 수 없습니다.
- BUFFER buffer-size. 리스토어 작업에 사용되는 버퍼의 크기를 설정할 수 없습니다.
- DLREPORT filename. 링크 해제되는 보고 파일의 파일 이름을 지정할 수 없습니다.
- WITHOUT ROLLING FORWARD. 데이터베이스가 성공적인 리스토어 작업 후에 롤 포워드 보류 상태가 되지 않음을 지정할 수 없습니다.
- PARALLELISM n. 리스토어 작업에 대한 병렬 처리 수준을 표시할 수 없습니다.
- WITHOUT PROMPTING. 리스토어 작업이 의도하지 않게 실행됨을 지정할 수 없습니다.

또한 복구 유틸리티에서 어떤 REBUILD 옵션도 지정할 수 없습니다. 그러나 복구 유틸리티는 복구 실행기록 파일에 있는 정보를 기초로 데이터베이스 백업 이미지를 찾을 수 없는 경우 적절한 REBUILD 옵션을 자동으로 사용합니다.

데이터 복구

복구 실행기록 파일에서 발견되는 정보를 사용하여 데이터베이스를 지정된 시간으로 복구하려면 RECOVER DATABASE 명령을 사용하십시오.

롤 포워드 단계 중에 종료한 불완전한 복구 조작 후에 RECOVER DATABASE 명령을 발행하는 경우, 복구 유틸리티는 리스토어 단계를 재실행하지 않고 이전 복구 조작을 계속하려고 시도합니다. 복구 유틸리티가 리스토어 단계를 재실행하도록 강제하려는 경우, RESTART 옵션과 함께 RECOVER DATABASE 명령을 발행하여 강제로 복

구 유틸리티가 완료하지 못한 모든 이전 복구 조작을 무시하도록 하십시오. API를 사용 중인 경우 iRecoverAction 필드에 대한 호출자 조치 DB2RECOVER_RESTART를 지정하여 강제로 복구 유틸리티가 리스토어 단계를 재실행하도록 하십시오.

RECOVER DATABASE 명령이 리스토어 단계 중에 인터럽트되는 경우 계속될 수 없습니다. RECOVER DATABASE 명령을 재발행해야 합니다.

복구될 데이터베이스에 연결되지 않았어야 합니다. 데이터베이스 복구 유틸리티가 자동으로 지정된 데이터베이스에 연결하고 이 연결은 복구 조작이 완료될 때 종료됩니다.

데이터베이스는 로컬 또는 리모트일 수 있습니다.

명령행 처리기(CLP) 또는 db2Recover API를 통해 복구 유틸리티를 호출할 수 있습니다.

다음 예는 CLP를 통해 RECOVER DATABASE 명령을 사용하는 방법을 보여줍니다.

```
db2 recover db sample
```

주: 파티션된 데이터베이스 환경에서 복구 유틸리티는 데이터베이스의 카탈로그 파티션에서 호출되어야 합니다.

db2adutl을 사용한 데이터 복구

다음 예는 db2adutl 명령과 *logarchopt1* 및 *vendoropt* 데이터베이스 구성 매개변수를 사용하여 상호 노드 복구를 수행하는 방법을 보여줍니다.

다음 예의 경우 컴퓨터 1은 이름이 bar이고 AIX를 실행 중입니다. 이 머신의 소유자는 roecken입니다. bar의 데이터베이스 이름은 zample입니다. 컴퓨터 2의 이름은 dps입니다. 이 머신 역시 AIX를 실행 중이며 regress9가 소유하고 있습니다.

PASSWORDACCESS = generate

컴퓨터 1

1. 로그가 TSM으로 아카이브되도록 데이터베이스를 설정하십시오. zample 데이터베이스의 데이터베이스 구성 매개변수 *logarchmeth1*을 갱신하십시오.

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   다음 정보가 리턴됩니다.

   ```
   DB20000I  UPDATE DATABASE CONFIGURATION 명령이 완료되었습니다.
   ```

   주: 데이터베이스 구성을 갱신하기 전에, 데이터베이스의 오프라인 백업을 수행해야 할 수도 있습니다.
2. 응용프로그램을 강제로 해제하십시오.

```
db2 force applications all
```

3. 모든 응용프로그램이 강제로 해제되었는지 검증하십시오.

   ```
   db2 list applications
   ```

   어떤 데이터도 리턴되지 않았음을 알리는 메시지가 수신되어야 합니다.

   주: 파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션에 대해 이 단계를 수행해야 합니다.

4. 데이터베이스 백업을 수행하십시오.

   ```
   db2 backup db zample use tsm
   ```

   다음과 유사한 정보가 리턴됩니다.

   ```
   백업이 완료되었습니다.
이 백업 이미지에 대한 시간소인은 20040216151025입니다.
   ```

   주: 파티션된 데이터베이스 환경에서는 모든 데이터베이스 파티션에 대해 이 단계를 수행해야 합니다. 데이터베이스 파티션에서 이 단계를 수행하는 순서는 온라인 백업 또는 오프라인 백업 중 어느 백업을 수행하는지 여부에 따라 다릅니다. 자세한 정보는 232 페이지의 『백업 사용』을 참조하십시오.

5. zample 데이터베이스에 연결한 후 데이터베이스 안에서 테이블을 작성하십시오.
6. 데이터를 새 테이블로 로드하십시오. 이 예에서는 a 테이블이 있고, 데이터는 mr이라고 하는 컬럼 식별자가 있는 ASCII 파일에서 로드됩니다. COPY YES 옵션은 로드되는 데이터의 사본을 작성하기 위해 지정하고 USE TSM 옵션은 데이터 사본이 Tivoli Storage Manager에 저장됨을 지정합니다.

   주: 데이터베이스가 롤 포워드 복구에 사용 가능한 경우에만 COPY YES 옵션을 지정할 수 있습니다. 즉, *logarchmeth1* 데이터베이스 구성 매개변수는 USEREXIT 또는 LOGRETAIN으로 설정해야 합니다.

   ```
bar:/home/roecken> db2 load from mr of del modified by noheader replace
      into a copy yes use tsm
   ```

   유틸리티는 해당 프로세스를 표시하기 위해 일련의 메시지를 리턴합니다.

   ```
SQL3109N  SQL3109N  유틸리티가 파일 파일 "/home/roecken/mr"에서 데이터를 로드하기
          시작합니다.

SQL3500W  유틸리티가 "02/16/2004 15:12:13.392633"에 "
          LOAD" 단계를 시작 중입니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "0".

SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3110N  유틸리티가 처리를 완료했습니다. 입력 파일에서
          "1"개의 행을 읽었습니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "1".
   ```

```
SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3515W  유틸리티가 "02/16/2004 15:12:13.445718"에
          "LOAD" 단계를 시작 중입니다.


읽은 행 수                = 1
건너뛴 행 수              = 0
로드된 행 수              = 1
거부된 행 수              = 0
삭제된 행 수              = 0
커미트된 행 수            = 1
```

이제 하나의 백업 이미지, 하나의 로드 사본 이미지 및 하나의 로그 파일이 TSM에 있어야 합니다. zample 데이터베이스에 대한 쿼리는 다음과 같이 실행할 수 있습니다.

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
```

다음 정보가 리턴됩니다.

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

7. 상호 노드 복구를 사용하려면 다른 노드와 어카운트에 bar 컴퓨터에 있는 오브젝트에 대한 액세스 권한이 부여되어야 합니다. 이 예에서는 노드 dps와 사용자 regress9에 액세스가 부여됩니다.

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
      on nodename dps for db zample
   ```

   다음 정보가 리턴됩니다.

   ```
   Successfully added permissions for regress9 to access ZAMPLE on node dps.
   ```

db2adutl grant 조작의 결과를 쿼리하려면 다음 명령을 실행하십시오.

```
bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
```

다음 정보가 리턴됩니다.

```
Node                 Username             Database Name   Type
-----------------------------------------------------------------
DPS                  regress9             ZAMPLE          A
-----------------------------------------------------------------
 Access Types:  B - backup images   L - logs   A - both
```

PASSWORDACCESS = 환경 생성

컴퓨터 2

컴퓨터 2 dps가 아직 설정되지 않았습니다. zample 데이터베이스에 대한 dps에서의 db2adutl 쿼리는 다음 결과를 리턴합니다.

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
--- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.


dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
    bar owner roecken
--- Database directory is empty ---

Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

zample 데이터베이스가 아직 dps 컴퓨터에 존재하지 않습니다.

1. zample 데이터베이스를 dps 컴퓨터로 리스토어하십시오.

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   다음 정보가 리턴됩니다.

   ```
   DB20000I  RESTORE DATABASE 명령이 완료되었습니다.
   ```

   주: zample 데이터베이스가 이미 dps에 존재하면, OPTIONS 매개변수를 생략하고 데이터베이스 구성 매개변수 *vendoropt*가 사용됩니다. 이 구성 매개변수는 백업 또는 리스토어 작업에 대한 OPTIONS 매개변수보다 우선합니다.

   zample 데이터베이스에 대한 롤 포워드 작업은 롤 포워드 유틸리티가 로그 파일을 찾을 수 없어서 실패합니다. 다음과 같은 롤 포워드 작업은

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   다음 오류를 리턴합니다.

   ```
   SQL4970N  "0" 노드에서 로그 파일이 누락되어 "ZAMPLE" 데이터베이스의
   롤 포워드 복구가 지정된 중지점(로그의 끝 또는 특정 시점)에 도달할 수
   없습니다.
   ```

2. 롤 포워드 유틸리티가 다른 머신에서 로그 파일을 강제로 찾도록 하려면 적절한 *logarchopt* 값을 구성해야 합니다(이 상황에서는 *logarchopt1* 데이터베이스 구성 매개변수).

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

3. 롤 포워드 유틸리티가 로드 사본 이미지를 사용할 수 있도록, *vendoropt* 데이터베이스 구성 매개변수도 설정해야 합니다.

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

4. zample 데이터베이스는 이제 롤 포워드될 수 있습니다.

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   다음 정보가 리턴됩니다.

   ```
                                  롤 포워드 상태

   입력 데이터베이스 별명                = zample
   상태를 리턴한 노드 수                 = 1

   노드 번호                            = 0
   롤 포워드 상태                       = not pending
   다음에 읽을 로그 파일                 =
   처리된 로그 파일                      = S0000000.LOG - S0000000.LOG
   최종 커미트된 트랜잭션                = 2004-02-16-20.10.38.000000 UTC

  DB20000I  ROLLFORWARD 명령이 완료되었습니다.
   ```

PASSWORDACCESS = 프롬프트 환경

PROMPT 환경에서는 추가 정보가 필요합니다(특히, 오브젝트가 작성된 머신의 TSM 노드 이름 및 암호).

db2adutl의 경우, dsm.sys 파일(Windows 기반 플랫폼에서 *dsm.opt* 파일)을 갱신하고 NODENAME bar(bar은 소스 컴퓨터의 이름이므로)을 서버 절에 추가하십시오.

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
owner roecken password *******
```

다음 정보가 리턴됩니다.

```
Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

1. 데이터베이스가 없으면 비어 있는 zample 데이터베이스를 작성하십시오. zample 데이터베이스가 이미 존재하면 이 단계와 데이터베이스 구성을 갱신하는 다음 단계를 건너뛸 수 있습니다.

   ```
   dps:/home/regress9> db2 create db zample
   ```

2. zample 데이터베이스의 데이터베이스 구성 매개변수 *tsm_nodename*을 갱신하십시오.

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

3. zample 데이터베이스의 데이터베이스 구성 매개변수 *tsm_password*를 갱신하십시오.

```
dps:/home/regress9> db2 update db cfg for zample using
tsm_password ********
```

4. zample d데이터베이스를 리스토어하십시오.

```
dps:/home/regress9> db2 restore db zample use tsm options
"'-fromnode=bar -fromowner=roecken'" without prompting
```

리스토어 작업이 성공적으로 완료되지만 경고가 발행됩니다.

```
SQL2540W  리스토어는 성공적이었으나, 인터럽트 없음 모드에서 처리하는 동안
  데이터베이스 리스토어 중에 경고 "2539"이(가)
  발견되었습니다.
```

다시, 이 때 롤 포워드 유틸리티는 올바른 로그 파일을 찾을 수 없습니다.

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

다음 오류 메시지가 리턴됩니다.

```
SQL1268N  "0" 노드에서 "ZAMPLE" 데이터베이스에 대해 로그 파일 "S0000000.LOG"를
  검색하는 동안 -2112880618" 오류가 발생하여 롤 포워드 복구가 중지되었습니다.
```

5. 데이터베이스 리스토어 작업이 데이터베이스 구성 파일을 교체하므로, TSM 데이터베이스 구성 값을 올바른 값으로 설정해야 합니다. 먼저 *tsm_nodename* 구성 매개변수를 재설정해야 합니다.

```
dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
```

6. *tsm_password* 데이터베이스 구성 매개변수를 재설정해야 합니다.

```
dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
```

7. *logarchopt1* 데이터베이스 구성 매개변수를 재설정하여, 롤 포워드 유틸리티가 올바른 로그 파일을 찾을 수 있도록 해야 합니다.

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

8. 로드 복구 파일도 사용할 수 있도록 *vendoropt* 데이터베이스 구성 매개변수도 재설정해야 합니다.

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

9. 데이터베이스 구성 매개변수가 설정되면, 데이터베이스는 롤 포워드될 수 있습니다.

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

zample 데이터베이스의 ROLLFORWARD QUERY STATUS 명령은 다음을 표시합니다.

```
                                   롤 포워드 상태

입력 데이터베이스 별명                = zample
상태를 리턴한 노드 수                 = 1

노드 번호                           = 0
```

```
롤 포워드 상태                       = not pending
다음에 읽을 로그 파일                = 
처리된 로그 파일                     = S0000000.LOG - S0000000.LOG
최종 커미트된 트랜잭션               = 2004-02-16-20.10.38.000000 UTC

DB20000I  ROLLFORWARD 명령이 완료되었습니다.
```

삭제된 테이블 복구

때로는 여전히 필요한 데이터를 포함하는 테이블을 삭제할 수 있습니다. 이 경우에는 중요한 테이블을 테이블 삭제 조작 뒤에 복구 가능하게 만들 것을 고려해야 합니다. 데이터베이스 리스토어 작업을 호출할 후 테이블이 삭제되기 전의 특정 시점으로 데이터베이스 롤 포워드 작업을 호출하여 테이블 데이터를 복구할 수 있습니다. 데이터베이스가 큰 경우 이 작업은 시간이 많이 걸릴 수 있으며 복구 중에는 데이터가 사용 불가능합니다. 삭제된 테이블 복구 기능으로 테이블 스페이스 레벨 리스토어 및 롤 포워드 조작을 사용하여 삭제된 테이블 데이터를 복구할 수 있습니다. 이것이 데이터베이스 레벨 복구보다 더 빠르며 데이터베이스는 계속 사용 가능합니다.

삭제된 테이블이 복구 가능하려면 테이블이 있는 테이블 스페이스에서 DROPPED TABLE RECOVERY 옵션이 켜져 있어야 합니다. 이것은 테이블 스페이스 작성 중이나 ALTER TABLESPACE문을 호출하여 수행될 수 있습니다. DROPPED TABLE RECOVERY 옵션은 테이블 스페이스마다 다르며 일반 테이블 스페이스로 제한됩니다. 테이블 스페이스에서 삭제된 테이블 복구가 가능한지 판별하려면 SYSCAT.TABLESPACES 카탈로그 테이블의 DROP_RECOVERY 컬럼을 쿼리할 수 있습니다.

삭제된 테이블 복구 옵션은 테이블 스페이스를 작성할 때 디폴트로 설정됩니다. 테이블 스페이스에서 삭제된 테이블 복구를 사용하지 않으려는 경우, CREATE TABLESPACE 문을 실행할 때 DROPPED TABLE RECOVERY 옵션을 명시적으로 OFF로 설정하거나 ALTER TABLESPACE문을 사용하여 기존 테이블 스페이스에 대한 삭제된 테이블 복구를 사용 안할 수 있습니다. 삭제된 테이블 복구 기능은 복구할 테이블 삭제 조작이 많거나 실행기록 파일이 큰 경우 포워드 복구에 성능 영향을 미칠 수 있습니다.

DROP TABLE문이 테이블 스페이스가 삭제된 테이블 복구가 사용 가능한 테이블에 대해 실행될 때 추가 항목(삭제된 테이블 식별)가 로그 파일에 작성됩니다. 또한 복구 실행기록 파일에 테이블을 재작성하는 데 사용할 수 있는 정보를 포함하는 항목도 작성됩니다.

파티션된 테이블의 경우, 하나 이상의 테이블 스페이스의 파티션되지 않은 테이블에 대해 삭제된 테이블 복구 기능이 꺼져 있는 경우에도 삭제된 테이블 복구가 항상 켜져 있습니다. 파티션된 테이블에 대해 하나의 삭제된 테이블 로그 레코드만 작성됩니다. 이 로그 레코드가 테이블의 모든 데이터 파티션을 복구하기에 충분합니다.

삭제된 테이블에서 복구 가능한 데이터 유형에 대해 몇 가지 제한사항이 있습니다. 다음의 경우는 복구할 수 없습니다.

- DROPPED TABLE RECOVERY 옵션은 임시 테이블에 사용할 수 없습니다.
- 행 유형과 연관된 메타데이터. (데이터가 복구되지만 메타데이터는 복구되지 않습니다.) 유형이 지정된 테이블의 계층 구조 테이블에 있는 데이터는 복구됩니다. 이 데이터는 삭제된 유형이 지정된 테이블에 나타나는 것보다 많은 정보를 포함할 수 있습니다.
- XML 데이터. XML 데이터를 포함하는 삭제된 테이블을 복구하려고 시도하면 대응하는 열 데이터는 비어 있습니다.

테이블이 삭제될 때 reorg 보류 상태에 있었던 경우, 실행기록 파일의 CREATE TABLE DDL이 임포트 파일의 DDL과 정확하게 일치하지 않습니다. 임포트 파일은 첫 번째 REORG 권장 ALTER가 수행되기 전의 테이블의 형식이지만, 실행기록 파일의 CREATE TABLE문은 모든 ALTER TABLE문의 결과를 포함하는 테이블의 상태와 일치합니다.

복구될 데이터가 GRAPHIC 또는 VARGRAPHIC 데이터 유형인 경우 둘 이상의 코드 페이지를 포함할 수 있습니다. 이 데이터를 복구하려면 IMPORT 또는 LOAD 명령의 `usegraphiccodepage` 파일 유형 지정자를 지정해야 합니다. 이 경우에 LOAD 명령을 사용하여 데이터를 복구하면 복구 작업의 성능이 증가됩니다.

한 번에 하나의 삭제된 테이블만 복구할 수 있습니다. 다음을 수행하여 삭제된 테이블을 복구할 수 있습니다.

1. LIST HISTORY DROPPED TABLE 명령을 호출하여 삭제된 테이블을 식별하십시오. 삭제된 테이블 ID가 백업 ID 컬럼에 나열됩니다.
2. 테이블이 삭제되기 전에 작성된 데이터베이스 또는 테이블 스페이스 레벨 백업 이미지를 리스토어하십시오.
3. 테이블 데이터를 포함하는 파일이 기록될 익스포트 디렉토리를 작성하십시오. 이 디렉토리는 모든 데이터베이스 파티션에서 액세스 가능하거나 각 데이터베이스 파티션에 존재해야 합니다. 이 익스포트 디렉토리 아래의 서브디렉토리가 각 데이터베이스 파티션에 의해 자동으로 작성됩니다. 이러한 서브디렉토리는 NODEnnnn으로 이름이 지정되는데, nnnn은 데이터베이스 파티션 또는 노드 번호를 나타냅니다. 각 데이터베이스 파티션에 존재했던 그대로의 삭제된 테이블 데이터가 들어 있는 데이터 파일이 data라는 하위 서브디렉토리로 익스포트됩니다.
   예:

   ```
   ₩export_directory₩NODE0000₩data.
   ```

4. ROLLFORWARD DATABASE 명령에서 RECOVER DROPPED TABLE 옵션을 사용하여 테이블이 삭제된 후의 특정 시점으로 롤 포워드하십시오. 또는 로그의 끝까지 롤 포워드하여 테이블 스페이스 또는 데이터베이스의 기타 테이블에 대한 갱신이 유실되지 않게 하십시오.
5. 복구 실행기록 파일에서 CREATE TABLE문을 사용하여 테이블을 재작성하십시오.

6. 롤 포워드 작업 중에 익스포트된 테이블 데이터를 테이블로 임포트하십시오. 삭제가 발생할 때 테이블이 reorg 보류 상태에 있었던 경우, CREATE TABLE DDL의 컨텐츠가 데이터 파일의 컨텐츠와 일치하도록 변경되어야할 수 있습니다.

응급 복구

데이터베이스에서의 트랜잭션 또는 작업 단위(UOW)는 예상치 않게 인터럽트될 수 있습니다. 작업 단위(UOW)에 속한 변경 모두가 완료 및 커미트되기 전에 실패한 경우 데이터베이스는 불일치한 상태 및 사용 불가능한 상태로 남아 있습니다. *응급 복구*는 데이터베이스가 일관성 있는 사용 가능한 상태로 다시 이동되는 프로세스입니다. 불완전한 트랜잭션을 롤백하고 손상되었을 때 메모리에 남아 있는 커미트된 트랜잭션을 완료하면 됩니다(그림 17). 데이터베이스가 일관성 있고 사용 가능한 상태이면 "일관성 지점"이라고 하는 지점에 도달합니다.

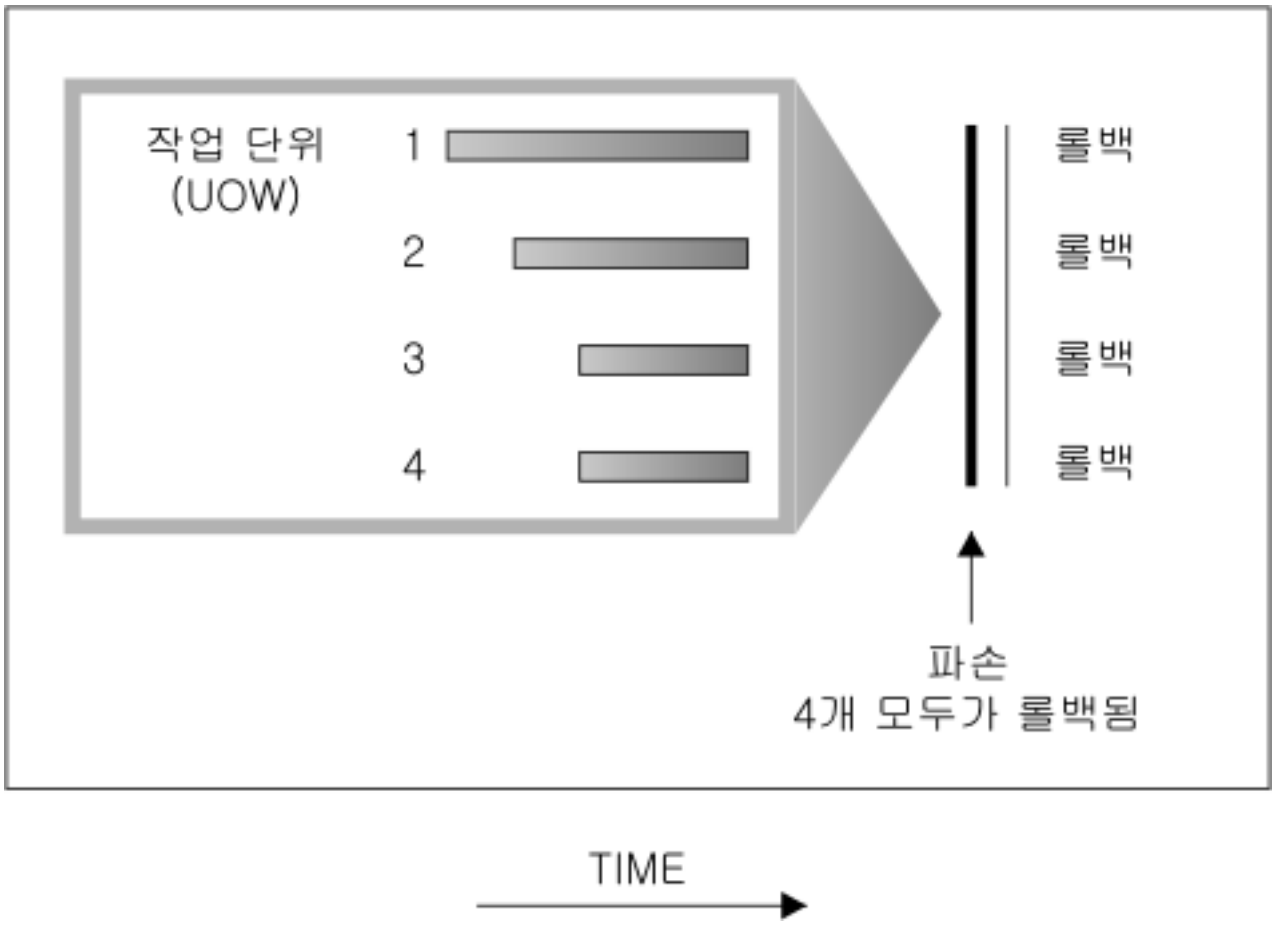


그림 17. 작업 단위(UOW) 롤백(응급 복구)

*트랜잭션 실패*는 데이터베이스 또는 데이터베이스 관리 프로그램이 비정상적으로 종료되는 심각한 오류 또는 조건으로 인해 발생합니다. 실패 시 디스크에 플러시되지 않은

부분적으로 완료된 작업 단위(UOW)는 데이터베이스에서 불일치 상태로 남게 됩니다. 트랜잭션 실패 이후에는 데이터베이스를 복구해야 합니다. 트랜잭션 실패를 야기하는 조건으로는 다음이 있습니다.

- 머신에서 전력 장애가 발생한 경우. 이로 인해 이 머신의 데이터베이스 관리 프로그램 및 데이터베이스 파티션이 중지됩니다.
- 메모리 손상 또는 디스크, CPU 또는 네트워크 장애와 같은 하드웨어 장애.
- DB2가 중단되는 심각한 운영 체제 오류.
- 응용프로그램의 비정상적 종료.

데이터베이스 관리 프로그램이 미완료된 작업 단위(UOW)를 자동으로 롤백하도록 하려면 자동 재시작(*autorestart*) 데이터베이스 구성 매개변수를 ON으로 설정하여 사용합니다. 이는 디폴트값입니다. 자동 재시작 동작을 사용하지 않으려면 *autorestart* 데이터베이스 구성 매개변수를 OFF로 설정합니다. 데이터베이스 오류가 발생한 경우 RESTART DATABASE 명령을 실행해야 합니다. 손상되기 전에 데이터베이스 입출력이 일시중단된 경우 응급 복구를 계속하려면 RESTART DATABASE 명령의 WRITE RESUME 옵션을 지정해야 합니다. 데이터베이스 재시작 조작을 시작하면 관리 통지 로그가 기록됩니다.

응급 복구가 포워드 복구가 가능한 데이터베이스(즉, *logarchmeth1* 구성 매개변수가 OFF로 설정되지 않음)에 적용되는 경우 개별 테이블 스페이스에 기인하는 응급 복구 중 오류가 발생하면 해당 테이블 스페이스가 오프라인 상태가 되고 수리할 때까지 액세스할 수 없습니다. 응급 복구가 계속됩니다. 응급 복구를 완료하면 데이터베이스의 다른 테이블 스페이스에 액세스할 수 있으며 데이터베이스에 대한 연결을 설정할 수 있습니다. 그러나 오프라인 상태의 테이블 스페이스가 시스템 카탈로그를 포함하는 테이블 스페이스인 경우 연결을 허용하기 전에 먼저 수리해야 합니다.

손상된 테이블 스페이스 복구

손상된 테이블 스페이스에는 액세스할 수 없는 하나 이상의 컨테이너가 있습니다. 종종 영구(예: 잘못된 디스크) 또는 임시(예: 오프라인 디스크 또는 언마운트된 파일 시스템) 상태인 미디어 문제점이 원인일 수 있습니다.

손상된 테이블 스페이스가 시스템 카탈로그 테이블이면 데이터베이스를 재시작할 수 없습니다. 컨테이너 문제점을 수정할 수 없고 원래 데이터를 그대로 두는 경우 다음 옵션만 사용 가능합니다.

- 데이터베이스 리스토어
- 카탈로그 테이블 스페이스 리스토어

주:

1. 데이터베이스를 롤 포워드해야 하므로 복구 가능한 데이터베이스에서는 테이블 스페이스 리스토어만 유효합니다.

2. 카탈로그 테이블 스페이스를 리스토어하는 경우 로그 끝에서 롤 포워드 조작을 수행해야 합니다.

손상된 테이블 스페이스가 시스템 카탈로그 테이블이 *아닌* 경우 DB2에서는 가능한 한 많은 데이터베이스를 사용 가능하게 설정하려고 합니다.

손상된 테이블 스페이스가 오직 임시 테이블 스페이스인 경우 데이터베이스에 연결하는 대로 새 임시 테이블 스페이스를 작성해야 합니다. 작성하면 새 임시 테이블 스페이스를 사용할 수 있으며 임시 테이블 스페이스가 필요한 일반 데이터베이스 조작을 다시 시작할 수 있습니다. 원하는 경우 오프라인 임시 테이블 스페이스를 삭제할 수 있습니다. 시스템 임시 테이블 스페이스를 사용하는 테이블을 재구성하는 경우 다음과 같은 특별 고려사항이 있습니다.

- 데이터베이스 또는 데이터베이스 관리 프로그램 구성 매개변수 *indexrec*가 RESTART로 설정된 경우 데이터베이스 활성화 중 유효하지 않은 모든 인덱스를 재빌드해야 합니다. 빌드 단계에서 손상된 인덱스도 재구성해야 합니다.
- 손상된 임시 테이블 스페이스에서 불완전한 재구성 요청이 있으면 *indexrec* 구성 매개변수를 ACCESS로 설정하여 재시작 실패를 방지해야 합니다.

복구 가능한 데이터베이스에서 테이블 스페이스 복구

응급 복구가 필요할 때 손상된 테이블 스페이스는 오프라인이 되고 액세스할 수 없습니다. 롤 포워드 보류 상태에 있습니다. 추가 문제점이 없는 경우 재시작 조작은 손상된 테이블 스페이스를 사용해도 데이터베이스를 온라인으로 만드는 데 성공합니다. 일단 온라인이 되면 손상된 테이블 스페이스를 사용할 수 없지만 나머지 데이터베이스는 사용 가능합니다. 손상된 테이블 스페이스를 수정하고 사용 가능하게 하려면 아래 프로시저를 수행하십시오.

손상된 테이블 스페이스를 사용 가능하게 하려면 아래 프로시저 중 하나를 사용하십시오.

- 메소드 1
  1. 원래 데이터를 잃지 않고 손상된 컨테이너를 수정하십시오.
  2. 로그 끝까지 테이블 스페이스 롤 포워드 조작을 완료하십시오.

     주: 롤 포워드 조작은 먼저 테이블 스페이스를 오프라인에서 정상 상태로 만들려고 시도합니다.
- 메소드 2
  1. 원래 데이터를 잃거나 잃지 않고 손상된 컨테이너를 수정하십시오.
  2. 테이블 스페이스 리스토어 작업을 수행하십시오.
  3. 로그 끝이나 특정 시점까지 테이블 스페이스 롤 포워드 조작을 완료하십시오.

복구 불가능한 데이터베이스에서 테이블 스페이스 복구

응급 복구가 필요하지만 손상된 테이블 스페이스가 있을 때 손상된 테이블 스페이스가 삭제되는 경우에만 데이터베이스를 성공적으로 재시작할 수 있습니다. 복구 불가능한 데이터베이스에서는 손상된 테이블 스페이스를 복구하기 위해 필요한 로그가 보유되지 않습니다. 그러므로 그런 테이블 스페이스에 대해 유일하게 유효한 조치는 삭제하는 것입니다.

손상된 테이블 스페이스가 있는 데이터베이스를 재시작하려면 다음을 수행하십시오.

1. 규정되지 않은 데이터베이스 재시작 조작을 호출하십시오. 손상된 테이블 스페이스가 없는 경우에는 성공합니다. 실패하는 경우(SQL0290N), 관리 통지 로그 파일에서 현재 손상된 테이블 스페이스의 전체 목록을 찾으십시오.
2. 모든 손상된 테이블 스페이스를 삭제하려는 경우, 다른 데이터베이스 재시작 조작을 시작하여 DROP PENDING TABLESPACES 옵션에서 모든 손상된 테이블 스페이스를 나열하십시오. 손상된 테이블 스페이스가 DROP PENDING TABLESPACES 목록에 포함되는 경우, 테이블 스페이스는 삭제 보류 상태에 들어가며 복구 작업이 완료된 후 테이블 스페이스를 삭제해야 합니다.

   재시작 조작은 지정된 테이블 스페이스를 복구하지 않고 계속됩니다. 손상된 테이블 스페이스가 DROP PENDING TABLESPACES 목록에 포함되지 않는 경우, 데이터베이스 재시작 작업은 SQL0290N과 함께 실패합니다.

   주: DROP PENDING TABLESPACES 목록에 테이블 스페이스 이름을 포함하는 것이 테이블 스페이스가 삭제 보류 상태에 있을 것을 의미하지는 않습니다. 테이블 스페이스가 재시작 조작 중에 손상된 것으로 발견되는 경우에만 이 상태에 들어갑니다.
3. 데이터베이스 재시작 조작이 성공하는 경우, LIST TABLESPACES 명령을 호출하여 삭제 보류 상태에 있는 테이블 스페이스를 찾으십시오.
4. DROP TABLESPACE문을 실행하여 삭제 보류 상태에 있는 각 테이블 스페이스를 삭제하십시오. 이를 수행한 후에는 손상된 테이블 스페이스가 사용 중인 스페이스를 재개하거나 테이블 스페이스를 재작성할 수 있습니다.
5. 손상된 테이블 스페이스의 데이터를 삭제하고 잃지 않으려는 경우 다음을 수행할 수 있습니다.
   - 손상된 컨테이너를(원래 데이터를 잃지 않고) 수정하십시오.
   - RESTART DATABASE 명령을 재실행하십시오.
   - 데이터베이스 리스토어 작업을 수행하십시오.

미디어 장애 영향 줄이기

미디어 장애 확률을 줄이고 이러한 장애 유형으로부터의 복구를 단순화하려면 다음을 수행하십시오.

- 중요한 데이터베이스의 로그 및 데이터를 보유하는 디스크를 미러링하거나 복제하십시오.
- RAID(Redundant Array of Independent Disks) 구성(예: RAID 레벨 5)을 사용하십시오.
- 파티션된 데이터베이스 환경에서, 카탈로그 파티션의 데이터 및 로그를 처리하는 정확한 프로시저를 설정하십시오. 이 데이터베이스 파티션은 데이터베이스를 유지보수하는 데 매우 중요하기 때문에 다음 작업이 요구됩니다.
  - 신뢰 가능한 디스크에 위치
  - 복제
  - 빈번한 백업
  - 여기에 사용자 데이터를 입력하지 않음

디스크 오류로부터 보호

디스크 손상으로 인한 데이터 또는 로그 손상이 우려되는 경우 일부 양식의 디스크 오류 허용을 사용할 수 있습니다. 일반적으로 디스크 세트인 *디스크 배열*을 사용하여 달성할 수 있습니다.

디스크 배열은 때때로 단순히 RAID(Redundant Array of Independent Disks)라고 하기도 합니다. 디스크 배열은 운영 체제 또는 응용프로그램 레벨의 소프트웨어를 통해 제공될 수도 있습니다. CPU가 입출력 요청을 처리하는 방식을 통해 하드웨어 및 소프트웨어 디스크 배열을 구별할 수 있습니다. 하드웨어 디스크 배열의 경우 입출력 활동은 디스크 제어기가 관리하고 소프트웨어 디스크 배열의 경우 운영 체제 또는 응용프로그램이 관리합니다.

하드웨어 디스크 배열

하드웨어 디스크 배열의 경우 고유한 CPU를 탑재한 디스크 제어기에서 다중 디스크를 사용 및 관리합니다. 이 배열을 구성하는 디스크를 관리하는 데 필요한 모든 로직은 디스크 제어기에 포함되어 있으므로 이 구현은 운영 체제와는 독립적입니다.

기능 및 성능에서 구별되는 여러 유형의 RAID 아키텍처가 있지만 일반적으로 RAID 레벨 1 및 레벨 5만 널리 사용됩니다.

RAID 레벨 1은 디스크 미러링 또는 이중화라고도 합니다. *디스크 미러링*은 단일 디스크 제어기를 사용하여 한 디스크에서 두 번째 디스크로 데이터(전체 파일)를 복사합니다. *디스크 이중화*는 디스크 미러링과 유사하지만 이 경우 디스크가 두 번째 디스크 제어기(2개의 SCSI 어댑터와 같음)에 접속됩니다. 데이터 보호 기능은 양호합니다. 둘 중 하나의 디스크에서 장애가 발생할 수 있으며 이때 다른 디스크의 데이터에는 계속 액

세스할 수 있습니다. 디스크 이중화의 경우 디스크 제어기도 데이터 보호를 위협하지 않고 장애가 발생할 수 있습니다. 성능도 양호합니다. 그러나 이 구현에는 기본 디스크 수의 2배가 필요합니다.

RAID 레벨 5는 모든 디스크에서 섹터별로 스트라이핑된 데이터 및 패리티를 포함합니다. 패리티는 전용 드라이브에 저장되지 않고 데이터와 인터리브됩니다. 데이터 보호 기능은 양호합니다. 디스크에서 장애가 발생한 경우 스트라이핑된 패리티 정보와 함께 다른 디스크의 정보를 사용하여 데이터에 계속 액세스할 수 있습니다. 읽기 성능은 좋지만 쓰기 성능은 그렇지 않습니다. RAID 레벨 5 구성에는 최소 3개의 동일한 디스크가 필요합니다. 오버헤드에 필요한 디스크 스페이스 크기는 배열에 있는 디스크 수에 따라 달라집니다. 디스크 5개의 RAID 레벨 5 구성의 경우 스페이스 오버헤드는 20%입니다.

RAID(RAID 레벨 0 제외) 디스크 배열을 사용하는 경우 하나의 디스크에 장애가 발생해도 사용자가 배열의 데이터에 액세스할 수 있습니다. 핫 플러그 연결 또는 핫스왑 가능한 디스크가 배열에 사용되는 경우 배열을 사용하면서 장애가 발생한 디스크를 교체 디스크로 스왑할 수 있습니다. RAID 레벨 5의 경우 두 개의 디스크에서 동시에 장애가 발생한 경우 모든 데이터가 유실됩니다. 그러나 동시에 디스크 장애가 발생할 가능성은 매우 낮습니다.

로그용으로 소프트웨어 디스크 배열 또는 RAID 레벨 1 하드웨어 디스크 배열 사용을 고려할 수 있습니다. 이 방법을 사용하면 장애 시점으로의 복구가 가능하며 로깅에 중요한 양호한 쓰기 성능을 제공할 수 있습니다. 이를 수행하려면 *mirrorlogpath* 구성 매개변수를 사용하여 RAID 레벨 1 파일 시스템에서 미러 로그 경로를 지정하십시오. 안정성이 중요하고(디스크 오류 시 데이터 복구에 필요한 시간을 낭비할 수 없으므로) 이에 비해 쓰기 성능은 그다지 중요하지 않은 경우 RAID 레벨 5 하드웨어 디스크 배열 사용을 고려할 수 있습니다. 또는 쓰기 성능이 중요하고 추가 디스크 스페이스 비용은 그다지 중요하지 않은 경우 로그 및 데이터에 대해 RAID 레벨 1 하드웨어 디스크 배열을 고려할 수 있습니다.

사용 가능한 RAID 레벨에 대한 세부사항 정보는 다음 웹 사이트를 참조하십시오. http://www.acnc.com/04_01_00.html

소프트웨어 디스크 배열

소프트웨어 디스크 배열은 대부분 하드웨어 디스크 배열과 동일하게 구성되지만 디스크 트래픽은 서버에서 실행하는 응용프로그램 또는 운영 체제에 의해 관리됩니다. 다른 프로그램과 같이 소프트웨어 배열은 CPU 및 시스템 자원을 차지합니다. CPU 성능이 좋지 않은 시스템에 권장하는 옵션은 아닙니다. 전반적인 디스크 배열 성능은 서버의 CPU 로드 및 성능에 좌우된다는 점을 염두에 두어야 합니다.

일반적인 소프트웨어 디스크 배열에서는 디스크 미러링을 제공합니다. 동일한 디스크가 더 필요하지만 값비싼 디스크 제어기가 필요하지 않으므로 소프트웨어 디스크 배열은 비교적 저렴하게 구현할 수 있습니다.

주의:
디스크 배열에 운영 체제 시동 드라이브가 있는 경우 해당 드라이브에서 장애가 발생하면 시스템이 시작되지 않습니다. 디스크 배열을 실행하기 전에 드라이브에서 장애가 발생하면 디스크 배열 드라이브에 액세스할 수 없습니다. 시동 드라이브는 디스크 배열과 분리되어야 합니다.

트랜잭션 실패 영향 줄이기

트랜잭션 실패 영향을 줄이려면 다음을 보장하십시오.

- 각 DB2 서버에서 중단되지 않는 전력 공급
- 모든 데이터베이스 파티션에서 데이터베이스 로그를 저장할 적절한 디스크 스페이스
- 파티션된 데이터베이스 환경에서 데이터베이스 파티션 서버 간 신뢰할 수 있는 통신 링크
- 파티션된 데이터베이스 환경에서 시스템 클럭 동기화

파티션된 데이터베이스 환경에서 트랜잭션 장애 복구

파티션된 데이터베이스 환경에서 트랜잭션에 실패하면 일반적으로 실패한 데이터베이스 파티션 서버 및 트랜잭션에 참여한 기타 데이터베이스 파티션 서버 모두에서 데이터베이스 복구를 수행해야 합니다.

- 실패한 조건을 정정한 후 실패한 데이터베이스 파티션 서버에서 응답 복구가 수행됩니다.
- 실패를 발견한 후 사용 중인 다른 데이터베이스 파티션 서버에서 즉시 *데이터베이스 파티션 장애 복구*가 수행됩니다.

파티션된 데이터베이스 환경에서 트랜잭션이 제출된 데이터베이스 파티션 서버가 코디네이터 파티션이며 트랜잭션에서 처리하는 첫 번째 에이전트가 코디네이터 에이전트입니다. 코디네이터 에이전트는 다른 데이터베이스 파티션 서버에서 작업을 분산하고 트랜잭션에 사용되는 항목을 추적합니다. 응용프로그램이 트랜잭션의 COMMIT문을 실행하면 코디네이터 에이전트는 2단계로 구성된 커미트 프로토콜을 사용하여 트랜잭션을 커미트합니다. 첫 번째 단계 중 코디네이터 파티션은 트랜잭션에 참여하는 모든 다른 데이터베이스 파티션 서버에 PREPARE 요청을 분산합니다. 그러면 서버는 다음 중 하나로 응답합니다.

READ-ONLY
이 서버에서 데이터가 변경되지 않습니다.

YES 이 서버에서 데이터가 변경되었습니다.

NO 오류 때문에 서버에서 커미트 준비가 되지 않음

서버 중 하나가 NO로 응답하면 트랜잭션이 롤백됩니다. 그렇지 않으면 코디네이터 파티션은 두 번째 단계를 시작합니다.

두 번째 단계 중 코디네이터 파티션은 COMMIT 로그 레코드를 작성하고 YES로 응답한 모든 서버에 COMMIT 요청을 분산합니다. 다른 모든 데이터베이스 파티션 서버가 커미트된 후 코디네이터 파티션에 COMMIT 수신확인을 전송합니다. 코디네이터 에이전트가 모든 참여 서버에서 모든 COMMIT 수신확인을 받으면 트랜잭션이 완료됩니다. 이때 포인트 코디네이터는 FORGET 로그 레코드를 작성합니다.

활성 데이터베이스 파티션 서버에서 트랜잭션 장애 복구

데이터베이스 파티션 서버가 다른 서버가 중지되었음을 감지한 경우 실패한 데이터베이스 파티션 서버와 연관된 모든 작업이 중지됩니다.

- 여전히 사용 중인 데이터베이스 파티션 서버가 응용프로그램의 코디네이터 파티션이고 장애가 발생한 데이터베이스 파티션 서버에서 응용프로그램이 실행된 경우(COMMIT는 준비되지 않음) 장애 복구를 수행하기 위해 코디네이터 에이전트가 인터럽트됩니다. 코디네이터 에이전트가 COMMIT 처리의 두 번째 단계를 진행하는 경우 응용프로그램에 SQL0279N이 리턴되고 데이터베이스 연결이 끊어집니다. 그렇지 않으면 코디네이터 에이전트는 트랜잭션에 참여하는 다른 모든 서버에 ROLLBACK 요청을 분산시키고 응용프로그램에 SQL1229N이 리턴됩니다.
- 장애가 발생한 데이터베이스 파티션 서버가 응용프로그램의 코디네이터 파티션인 경우 장애 복구를 수행하기 위해 사용 중인 서버의 응용프로그램에서 여전히 작동하는 에이전트가 인터럽트됩니다. 트랜잭션이 아직 준비된 상태가 아닌 각 데이터베이스 파티션에서 트랜잭션이 로컬로 롤백됩니다. 트랜잭션이 준비 상태인 해당 데이터베이스 파티션에서 트랜잭션은 인다우트(Indoubt) 상태가 됩니다. 코디네이터 데이터베이스 파티션은 일부 데이터베이스 파티션에서 트랜잭션이 인다우트(Indoubt)임을 인식하지 못합니다. 코디네이터 데이터베이스 파티션이 사용 불가능하기 때문입니다.
- 장애가 발생한 데이터베이스 파티션 서버에 응용프로그램이 연결되었지만(장애 전에 연결됨) 로컬 데이터베이스 파티션 서버 또는 장애가 발생한 데이터베이스 파티션 서버가 코디네이터 파티션이 아니면 이 응용프로그램에서 작동하는 에이전트가 인터럽트됩니다. 코디네이터 파티션은 다른 데이터베이스 파티션 서버에 ROLLBACK 또는 DISCONNECT 메시지를 보냅니다. 코디네이터 파티션이 SQL0279를 리턴하면 여전히 사용 중인 데이터베이스 파티션 서버에서 트랜잭션은 인다우트(Indoubt) 상태만 가능합니다.

장애가 발생한 서버로 요청을 보내는 모든 프로세스(예: 에이전트 또는 교착 상태 검출기)에 요청을 보낼 수 없음을 알립니다.

실패한 데이터베이스 파티션 서버에서 트랜잭션 장애 복구

트랜잭션 실패로 데이터베이스 관리 프로그램이 비정상적으로 종료되면 RESTART 옵션과 함께 db2start 명령을 실행하여 데이터베이스 파티션이 재시작되면 데이터베이스 관리 프로그램을 재시작할 수 있습니다. 데이터베이스 파티션을 재시작할 수 없으면 db2start를 실행하여 다른 데이터베이스 파티션에서 데이터베이스 관리 프로그램을 재시작할 수 있습니다.

데이터베이스 관리 프로그램이 비정상적으로 종료되면 서버의 데이터베이스 파티션은 불일치 상태로 남을 수 있습니다. 사용 가능하게 하려는 경우 다음과 같이 데이터베이스 파티션 서버에서 응급 복구를 트리거할 수 있습니다.

- 명시적으로 RESTART DATABASE 명령 사용
- *autorestart* 데이터베이스 구성 매개변수가 ON으로 설정된 경우 명시적으로 CONNECT 요청 사용

응급 복구는 완료된 모든 트랜잭션의 효과가 데이터베이스 안에 적용되도록 사용 중인 로그 파일의 로그 레코드를 다시 적용합니다. 변경 사항이 다시 적용된 후 커미트되지 않은 모든 트랜잭션은 로컬로 롤백됩니다(단, 인다우트(Indoubt) 트랜잭션은 *제외*됨). 파티션된 데이터베이스 환경에서 다음과 같은 두 가지 인다우트(Indoubt) 트랜잭션 유형이 있습니다.

- 코디네이터 파티션이 아닌 데이터베이스 파티션 서버에서 준비되었지만 아직 커미트되지 않은 트랜잭션은 인다우트(in doubt) 상태가 됩니다.
- 코디네이터 파티션에서 커미트되었지만 아직 완료된 것으로 로그되지 않은 트랜잭션(즉, FORGET 레코드가 아직 작성되지 않음)은 인다우트(in doubt) 상태가 됩니다. 코디네이터 에이전트가 응용프로그램에서 작동하는 모든 서버에서 모든 COMMIT 수신확인을 받지 못한 경우 이 상황이 나타납니다.

응급 복구에서는 다음 중 하나를 수행하여 인다우트(Indoubt) 트랜잭션을 모두 해결하려고 합니다. 수행하는 조치는 데이터베이스 파티션 서버가 응용프로그램의 코디네이터 파티션인지에 따라 달라집니다.

- 재시작된 서버가 응용프로그램의 코디네이터 파티션이 아니면 코디네이터 파티션으로 조회 메시지를 보내 트랜잭션 결과를 찾습니다.
- 재시작된 서버가 응용프로그램의 코디네이터 파티션*이면* 코디네이터 에이전트가 COMMIT 수신확인을 계속 기다리고 있음을 나타내는 메시지가 다른 모든 에이전트(종속 에이전트)로 전송됩니다.

응급 복구는 모든 인다우트(Indoubt) 트랜잭션을 해결할 수는 없습니다. 예를 들어 일부 데이터베이스 파티션 서버가 사용 불가능할 수 있습니다. 트랜잭션에서 다른 데이터베이스 파티션을 사용하기 전에 코디네이터 파티션이 응급 복구를 완료하면 응급 복구로 인다우트(Indoubt) 트랜잭션을 해결할 수 없습니다. 각 데이터베이스 파티션이 독립적으로 응급 복구를 수행했기 때문입니다. 이 경우 SQL 경고 메시지 SQL1061W가 리

턴됩니다. 인다우트(Indoubt) 트랜잭션이 자원을 보유하는 경우(예: 잠금 및 사용 중인 로그 스페이스) 데이터베이스를 변경할 수 없는 지점으로 이동할 수 있습니다. 인다우트(Indoubt) 트랜잭션이 사용 중인 로그를 보유하고 있기 때문입니다. 따라서 응급 복구 이후 인다우트(Indoubt) 트랜잭션이 남아 있는지 판별하고 가능한 한 신속하게 인다우트(Indoubt) 트랜잭션을 해결하는 데 필요한 모든 데이터베이스 파티션 서버를 복구해야 합니다.

주: 파티션된 데이터베이스 서버 환경에서 RESTART 데이터베이스 명령은 노드별로 실행됩니다. 모든 노드에서 데이터베이스를 재시작하도록 하려면 다음의 권장 명령을 사용합니다.

```
db2_all "db2 restart database <database_name>"
```

인다우트(Indoubt) 트랜잭션을 해결해야 하는 하나 이상의 서버를 제시간에 복구할 수 없고 다른 서버의 데이터베이스 파티션에 액세스해야 하는 경우 경험을 바탕으로 인다우트(Indoubt) 트랜잭션을 수동으로 해결해야 합니다. LIST INDOUBT TRANSACTIONS 명령을 사용하여 서버에서 인다우트(Indoubt) 트랜잭션을 쿼리, 커미트 및 롤백할 수 있습니다.

주: LIST INDOUBT TRANSACTIONS 명령은 분산 트랜잭션 환경에서도 사용됩니다. 두 가지 유형의 인다우트(Indoubt) 트랜잭션을 서로 구별하기 위해 LIST INDOUBT TRANSACTIONS 명령에서 리턴된 출력의 *originator* 필드는 다음 중 하나를 표시합니다.

- DB2 Enterprise Server Edition. 이는 파티션된 데이터베이스 환경에서 트랜잭션이 시작되었음을 나타냅니다.
- XA. 이는 분산 환경에서 트랜잭션이 시작되었음을 나타냅니다.

장애가 발생한 데이터베이스 파티션 서버 식별

데이터베이스 파티션 서버에서 장애가 발생한 경우 일반적으로 응용프로그램은 다음 SQLCODE 중 하나를 수신합니다. 장애가 발생한 데이터베이스 관리 프로그램을 감지하는 방법은 수신된 SQLCODE에 따라 달라집니다.

SQL0279N

이 SQLCODE는 COMMIT 처리 중 트랜잭션에서 사용하는 데이터베이스 파티션 서버가 종료되는 경우 수신됩니다.

SQL1224N

이 SQLCODE는 장애가 발생한 데이터베이스 파티션이 트랜잭션에 대한 코디네이터 파티션인 경우 수신됩니다.

SQL1229N

이 SQLCODE는 장애가 발생한 데이터베이스 파티션이 트랜잭션에 대한 코디네이터 파티션이 아닌 경우 수신됩니다.

장애가 발생한 데이터베이스 파티션 서버를 판별하는 작업은 2단계 프로세스로 구성됩니다.

1. SQLCA를 검사하여 장애를 발견한 파티션 서버를 찾으십시오. SQLCODE SQL1229N과 연관된 SQLCA는 *sqlerrd* 필드의 6번째 배열 위치에서 오류를 발견한 서버의 노드 번호를 포함합니다. 이때 서버에서 작성된 노드 번호는 `db2nodes.cfg` 파일의 노드 번호에 대응합니다.
2. 실패한 서버의 노드 수에 대해 1단계에서 찾은 서버의 관리 통지 로그를 검사하십시오.

주: 프로세서에서 다중 논리 노드를 사용하는 경우 한 논리 노드에서 장애가 발생하면 동일한 프로세서의 다른 논리 노드에서도 장애가 발생합니다.

데이터베이스 파티션 서버의 실패로부터 복구

데이터베이스 파티션 서버의 실패로부터 복구하려면 다음 단계를 수행하십시오.

1. 실패를 유발한 문제점을 정정하십시오.
2. 임의의 데이터베이스 파티션 서버에서 db2start 명령을 실행하여 데이터베이스 관리 프로그램을 재시작하십시오.
3. 실패한 데이터베이스 파티션 서버에서 RESTART DATABASE 명령을 실행하여 데이터베이스를 재시작하십시오.

메인프레임 또는 중형 서버에서 인다우트(Indoubt) 트랜잭션 복구

DB2 Connect가 DB2 Syncpoint Manager를 구성했을 때 호스트에서 인다우트(Indoubt) 트랜잭션 복구

응용프로그램이 트랜잭션 중에 호스트 또는 System i 데이터베이스 서버에 액세스한 경우, 인다우트(Indoubt) 트랜잭션이 복구되는 방법에 일부 차이가 있습니다. 호스트 또는 System i 데이터베이스 서버에 액세스하기 위해 DB2 Connect가 사용됩니다. DB2 Connect에서 DB2 Syncpoint Manager가 구성된 경우 복구 단계가 다릅니다.

호스트 또는 System i 서버에 있는 인다우트(Indoubt) 트랜잭션의 복구는 일반적으로 트랜잭션 관리 프로그램(TM) 및 DB2 Syncpoint Manager(SPM)에 의해 자동으로 수행됩니다. 호스트 또는 System i 서버의 인다우트(Indoubt) 트랜잭션은 로컬 DB2 위치에서 어떤 자원도 보유하지 않지만, 트랜잭션이 해당 위치에서 인다우트(Indoubt)인 동안에는 호스트 또는 System i 서버에서 자원을 보유합니다. 호스트 또는 System i 서버의 관리자가 경험적 결정을 수행해야 한다고 판별하는 경우, 관리자는 로컬 DB2 데이터베이스 관리자에게 문의하여(예를 들어 전화를 통해) 호스트 또는 System i 서버에서 트랜잭션을 커미트 또는 롤백할지 여부를 판별할 수 있습니다. 이러한 상황이 발생하는 경우 LIST DRDA® INDOUBT TRANSACTIONS 명령을 사용하여 DB2 Connect 인스턴스에 있는 트랜잭션의 상태를 판별할 수 있습니다. 다음 단계를 사용하여 SNA 통신 환경과 관련된 대부분의 상황에 대한 지침으로 사용할 수 있습니다.

1. 아래 표시된 것처럼 SPM에 연결하십시오.

```
db2 => connect to db2spm

데이터베이스 연결 정보

데이터베이스 제품       = SPM0500
SQL 권한 부여 ID        = CRUS
로컬 데이터베이스 별명 = DB2SPM
```

2. LIST DRDA INDOUBT TRANSACTIONS 명령을 사용하여 SPM에 알려진 인다우트 트랜잭션을 표시하십시오. 아래 예는 SPM에 알려진 하나의 인다우트(Indoubt) 트랜잭션을 표시합니다. db_name은 호스트 또는 System i 서버에 대한 로컬 별명입니다. partner_lu는 호스트 또는 System i 서버의 완전한 LU 이름입니다. 이것은 호스트 또는 System i 서버의 최상의 식별을 제공하며 호스트 또는 System i 서버로부터 호출자에 의해 제공되어야 합니다. luwid는 트랜잭션에 대한 고유 ID를 제공하며 모든 호스트 및 System i 서버에서 사용할 수 있습니다. 의심이 가는 트랜잭션이 표시되는 경우, uow_status 필드를 사용하여 값이 C(커미트) 또는 R(롤백)인 경우 트랜잭션의 결과를 판별할 수 있습니다. WITH PROMPTING 매개변수를 갖는 LIST DRDA INDOUBT TRANSACTIONS 명령을 실행하는 경우 트랜잭션을 대화식으로 커미트, 롤백 또는 무시할 수 있습니다.

```
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 인다우트(Indoubt) 트랜잭션이 표시되지 않는 경우 또는 LIST DRDA INDOUBT TRANSACTIONS 명령이 다음과 같이 리턴하는 경우,

```
db2 => list drda indoubt transactions
SQL1251W  경험적 쿼리에 대해 리턴된 데이터가 없습니다.

트랜잭션은 롤백되었습니다.
```

발생할 수 있는 또 다른 가망 없지만 가능한 상황이 있습니다. partner_lu에 대한 적절한 luwid의 인다우트(Indoubt) 트랜잭션이 표시되지만 uow_status가 ″I″인 경우, SPM은 트랜잭션이 커미트 또는 롤백될지 여부를 알지 못합니다. 이 상황에서, WITH PROMPTING 매개변수를 사용하여 DB2 Connect 워크스테이션에서 트랜잭션을 커미트 또는 롤백해야 합니다. 그런 다음 DB2 Connect가 경험적 결정을 기초로 호스트 또는 System i 서버와 다시 동기화할 수 있도록 하십시오.

DB2 Connect가 DB2 동기점 관리 프로그램을 사용하지 않을 때 호스트에 인다우트(Indoubt) 트랜잭션 복구

응용프로그램이 트랜잭션 중에 호스트 또는 System i 데이터베이스 서버에 액세스한 경우, 인다우트(Indoubt) 트랜잭션이 복구되는 방법에 일부 차이가 있습니다. 호스트 또는 System i 데이터베이스 서버에 액세스하기 위해 DB2 Connect가 사용됩니다. DB2 Connect에 DB2 동기점 관리 프로그램이 구성된 경우 복구 단계가 다릅니다.

TCP/IP 연결성을 사용하여 DB2 Connect Personal Edition 또는 DB2 Connect Enterprise Edition에서의 다중 사이트 갱신에서 z/OS용 DB2를 갱신하고 DB2 Syncpoint Manager가 사용되지 않을 때 이 절의 정보를 사용하십시오. 이 상황의 인다우트(Indoubt) 트랜잭션의 복구는 DB2 Syncpoint Manager과 관련된 인다우트(Indoubt) 트랜잭션의 경우와는 다릅니다. 이 환경에서 인다우트(Indoubt) 트랜잭션이 발생할 때, 문제점을 발견한 사람에 따라서 클라이언트, 데이터베이스 서버 및(또는) 트랜잭션 관리 프로그램(TM) 데이터베이스에서 경보 항목이 생성됩니다. 경보 항목은 `db2alert.log` 파일에 있습니다.

모든 인다우트(Indoubt) 트랜잭션의 재동기화는 TM 및 참여 데이터베이스와 해당 연결이 모두 다시 사용 가능한 경우 바로 자동으로 발생합니다. 데이터베이스 서버에서 경험적으로 결정을 강제하기 보다 자동 재동기화가 발생할 수 있도록 해야 합니다. 그러나 사용자가 이를 수행해야 하는 경우 다음 단계를 지침으로 사용하십시오.

주: DB2 Syncpoint Manager은 관련되지 않기 때문에 LIST DRDA INDOUBT TRANSACTIONS 명령을 사용할 수 없습니다.

1. z/OS 호스트에서 DISPLAY THREAD TYPE(INDOUBT) 명령을 실행하십시오.

   이 목록에서 경험적으로 완료하기 원하는 트랜잭션을 식별하십시오. DISPLAY 명령에 대한 자세한 내용은 *DB2 for z/OS Command Reference*를 참조하십시오. 표시되는 LUWID는 트랜잭션 관리 프로그램 데이터베이스의 동일한 luwid와 일치할 수 있습니다.
2. 수행하려는 작업에 따라서 RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) 명령을 실행하십시오.

   RECOVER THREAD 명령에 대한 자세한 내용은 *DB2 for z/OS Command Reference*를 참조하십시오.

재해 복구

*재해 복구*란 용어는 화재, 지진, 파괴 또는 자연 재해가 발생한 경우 데이터베이스를 리스토어하기 위해 수행해야 하는 활동을 설명하는 데 사용됩니다. 재해 복구 플랜에는 다음 중 하나 이상이 포함될 수 있습니다.

- 긴급 상황에서 사용할 사이트

- 데이터베이스를 복구할 다른 머신
- 데이터베이스 백업, 테이블 스페이스 백업 또는 둘 다 및 아카이브된 로그의 오프사이트 스토리지

재해 복구 계획이 다른 머신에서 전체 데이터베이스를 리스토어하는 것이면 데이터베이스의 모든 아카이브된 로그 및 하나 이상의 전체 데이터베이스 백업을 보유하는 것이 좋습니다. 데이터베이스에서 각 테이블 스페이스의 전체 테이블 스페이스 백업을 사용하는 경우 데이터베이스를 재빌드할 수 있어도 이 방법은 많은 백업 이미지를 다루며 전체 데이터베이스 백업을 사용하는 복구보다 시간이 더 오래 걸립니다.

아카이브할 때 로그를 이에 적용하여 대기 데이터베이스를 최신 상태로 보존하도록 선택할 수 있습니다. 또는 대기 사이트에 데이터베이스 또는 테이블 스페이스 백업 및 로그 아카이브를 보존하고 재해가 발생한 후에만 리스토어 및 롤 포워드 조작을 수행하도록 선택할 수 있습니다. 후자의 경우 최신 백업 이미지가 더 바람직합니다. 그러나 재해가 발생한 경우 일반적으로 재해 시점까지 모든 트랜잭션을 복구하지 못할 수 있습니다.

재해 복구에서 테이블 스페이스 백업 유용성은 실패 범위에 따라 달라집니다. 일반적으로 재해 복구는 전체 데이터베이스를 리스토어하는 경우 다소 덜 복잡하고 시간이 오래 걸리지 않습니다. 손상된 디스크가 재해에 해당하는 경우 해당 디스크에서 각 테이블 스페이스의 테이블 스페이스 백업을 사용하여 복구할 수 있습니다. 디스크 장애 또는 기타 이유로 컨테이너에 액세스하지 못하게 되면 컨테이너를 다른 위치로 리스토어할 수 있습니다.

부분 또는 완전한 사이트 장애로부터 데이터를 보호하는 또 다른 방법은 DB2 고가용성 재해 복구(HADR) 기능을 구현하는 것입니다. 일단 설정하면 HADR은 소스 데이터베이스(1차)에서 목표 데이터베이스(대기)로 데이터 변경사항을 복제하여 데이터가 유실되지 않도록 보호합니다.

복제를 사용하여 부분 또는 완전한 사이트 장애로부터 데이터를 보호할 수도 있습니다. 복제를 사용하면 정기적으로 데이터를 여러 리모트 데이터베이스에 복사할 수 있습니다. DB2 데이터베이스에서는 여러 복제 도구를 제공합니다. 이 도구를 사용하면 복사해야 하는 데이터, 데이터를 복사할 데이터베이스 테이블 및 갱신사항을 복사하는 횟수를 지정할 수 있습니다.

피어 투 피어(Peer-to-peer) 리모트 복사(PPRC)와 같은 스토리지 미러링을 사용하여 데이터를 보호할 수도 있습니다. PPRC에서는 재해로부터 보호하기 위해 볼륨 또는 디스크의 동기 사본을 제공합니다.

DB2에서는 재해 복구 계획 시 여러 옵션을 제공합니다. 비즈니스 요구에 기반하여 데이터 유실을 방지하는 보호 방법으로 테이블 스페이스 또는 전체 데이터베이스 백업 사용을 결정할 수 있습니다. 또는 HADR과 같은 솔루션에 보다 적합하도록 환경을 조정

할 수도 있습니다. 어떤 방법을 선택하든, 프로덕션 환경에서 구현하기 전에 테스트 환경에서 복구 프로시저를 테스트해야 합니다.

버전 복구

*버전 복구*는 백업 작업 중 작성된 이미지를 사용하여 데이터베이스의 이전 버전을 리스토어하는 작업입니다. 복구 불가능한 데이터베이스(즉, 로그를 아카이브하지 않은 데이터베이스)에서 이 복구 방법을 사용합니다. 또한 RESTORE DATABASE 명령에서 WITHOUT ROLLING FORWARD 옵션을 사용하여 복구 가능한 데이터베이스에서 이 방법을 사용할 수도 있습니다. 데이터베이스 리스토어 작업은 이전에 작성된 백업 이미지를 사용하여 전체 데이터베이스를 리스토어합니다. 데이터베이스 백업에서는 백업한 당시 상태와 동일하게 데이터베이스를 리스토어할 수 있습니다. 그러나 백업 시점부터 실패 시점까지 모든 작업 단위(UOW)는 손실됩니다(그림 18 참조).

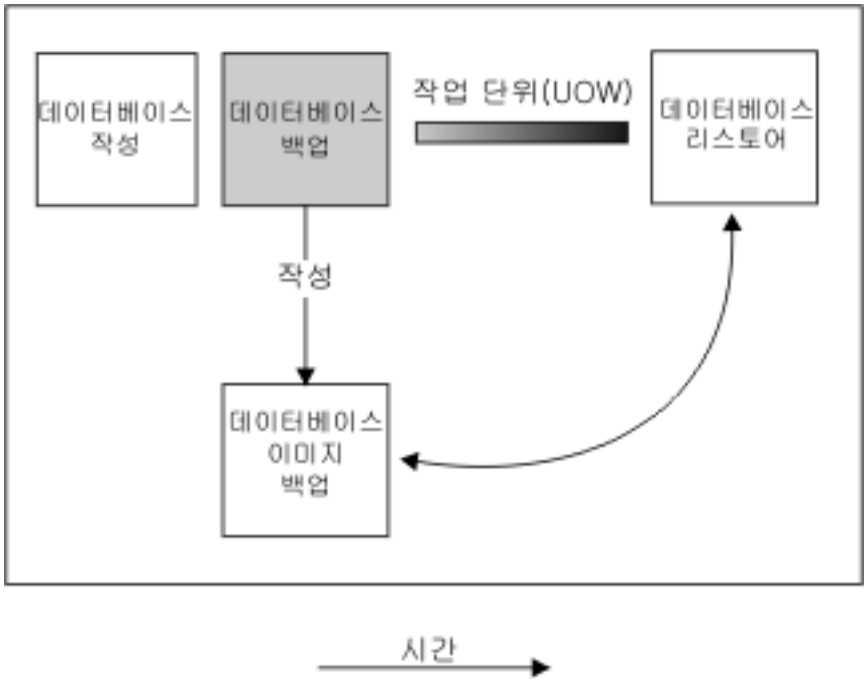


그림 18. 버전 복구. 백업 시점부터 실패 시점까지 작업 단위(UOW)가 손실됨을 표시합니다.

버전 복구 방법을 사용하는 경우 정기적으로 데이터베이스의 전체 백업을 스케줄 및 수행해야 합니다.

파티션된 데이터베이스 환경에서 데이터베이스는 여러 데이터베이스 파티션 서버 또는 노드에 있습니다. 모든 데이터베이스 파티션을 리스토어하고 데이터베이스 리스토어 조작에서 사용하는 백업 이미지는 모두 동시에 가져와야 합니다. 각 데이터베이스 파티션은 별도로 백업 및 리스토어됩니다. 동시에 수행된 각 데이터베이스 파티션 백업은 *버전 백업*이라고 합니다.

롤 포워드 복구

롤 포워드 복구 방법을 사용하려면 *logarchmeth1* 및 *logarchmeth2* 구성 매개변수를 OFF 이외의 값으로 설정하여 데이터베이스 및 아카이브된 로그를 백업해야 합니다. 데이터베이스를 리스토어하고 WITHOUT ROLLING FORWARD 옵션을 지정하는 방법은 버전 복구 방법을 사용하는 것과 같습니다. 데이터베이스는 오프라인 백업 이미지를 작성한 시점과 동일한 상태로 리스토어됩니다. 데이터베이스를 리스토어하고 데이터베이스 리스토어 조작에 대해 WITHOUT ROLLING FORWARD 옵션을 지정하지 않은 경우 데이터베이스는 리스토어 작업 종료 시 롤 포워드 보류 상태가 됩니다. 그러면 롤 포워드 복구를 수행할 수 있습니다.

주: 다음 경우에는 WITHOUT ROLLING FORWARD 옵션을 사용할 수 없습니다.

- 온라인 백업 이미지에서 리스토어하는 경우
- 테이블 스페이스 레벨 리스토어를 실행하는 경우

다음은 고려할 롤 포워드 복구의 두 가지 유형입니다.

- *데이터베이스 롤 포워드 복구*. 이러한 유형의 롤 포워드 복구의 경우 데이터베이스 리스토어 작업 이후에 데이터베이스 로그에 기록된 트랜잭션이 적용됩니다(275 페이지의 그림 19 참조). 데이터베이스 로그는 데이터베이스에서 변경한 모든 사항을 기록합니다. 이 방법은 실패 바로 전 상태(즉, 사용 중인 로그의 끝) 또는 특정 시점의 상태로 데이터베이스를 복구합니다.

  파티션된 데이터베이스 환경에서 데이터베이스는 여러 데이터베이스 파티션에 있으며 데이터베이스의 카탈로그 테이블이 있는 데이터베이스 파티션(카탈로그 파티션)에서 ROLLFORWARD DATABASE 명령을 실행해야 합니다. 특정 시점 롤 포워드 복구를 수행하는 경우 모든 데이터베이스 파티션 레벨이 동일하도록 하려면 모든 데이터베이스 파티션을 롤 포워드해야 합니다. 단일 데이터베이스 파티션을 리스토어해야 하는 경우 로그 끝으로 롤 포워드 복구를 수행하여 데이터베이스의 다른 데이터베이스 파티션과 동일한 레벨로 설정할 수 있습니다. 데이터베이스 파티션이 롤 포워드되는 경우 로그 끝으로 복구하는 작업만 사용할 수 있습니다. 특정 시점 복구는 모든 데이터베이스 파티션에 적용됩니다.

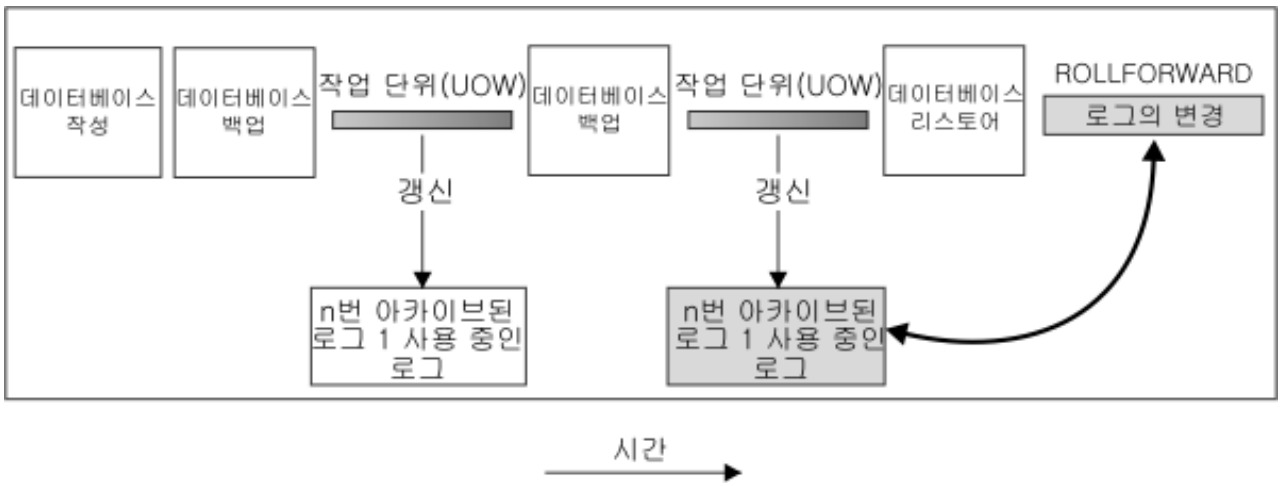


그림 19. 데이터베이스 롤 포워드 복구. 장기 실행 중인 트랜잭션이 있으면 사용 중인 로그가 둘 이상일 수 있습니다.

- *테이블 스페이스 롤 포워드 복구.* 데이터베이스에서 포워드 복구를 수행할 수 있으면 테이블 스페이스를 백업, 리스토어 및 롤 포워드할 수 있습니다(276 페이지의 그림 20 참조). 테이블 스페이스 리스토어 및 롤 포워드 조작을 수행하려면 전체 데이터베이스(즉, 모든 테이블 스페이스) 또는 하나 이상의 개별 테이블 스페이스의 백업 이미지가 필요합니다. 또한 복구할 테이블 스페이스에 영향을 주는 로그 레코드도 필요합니다. 로그를 통해 다음 두 지점 중 하나로 롤 포워드할 수 있습니다.
  - 로그 끝 또는
  - 특정 시점(*특정 시점* 복구라고 함)

  테이블 스페이스 롤 포워드 복구는 다음 두 상황에서 사용할 수 있습니다.
  - 테이블 스페이스 리스토어 작업 후, 테이블 스페이스는 항상 롤 포워드 보류 상태가 되므로 이를 롤 포워드해야 합니다. ROLLFORWARD DATABASE 명령을 호출하여 특정 시점 또는 로그 끝으로 테이블 스페이스에 대한 로그를 적용합니다.
  - 응급 복구 후 하나 이상의 테이블 스페이스가 *롤 포워드 보류* 상태이면 먼저 테이블 스페이스 문제점을 정정합니다. 일부 경우 테이블 스페이스 문제점 정정은 데이터베이스 리스토어 조작과 아무 상관이 없습니다. 예를 들어 전력 손실이 발생하면 테이블 스페이스는 롤 포워드 보류 상태로 남아 있을 수 있습니다. 이 경우 데이터베이스 리스토어 조작은 필요하지 않습니다. 테이블 스페이스 문제점을 정정하면 ROLLFORWARD DATABASE 명령을 사용하여 테이블 스페이스의 로그를 로그 끝에 적용할 수 있습니다. 응급 복구 전에 문제점을 정정하면 응급 복구에서 데이터베이스를 일관성 있는 사용 가능한 상태로 설정할 수 있습니다.

    주: 오류가 발생한 테이블 스페이스에 시스템 카탈로그 테이블이 있으면 데이터베이스를 시작할 수 없습니다. SYSCATSPACE 테이블 스페이스를 리스토어하고 로그 끝으로의 롤 포워드 복구를 수행해야 합니다.

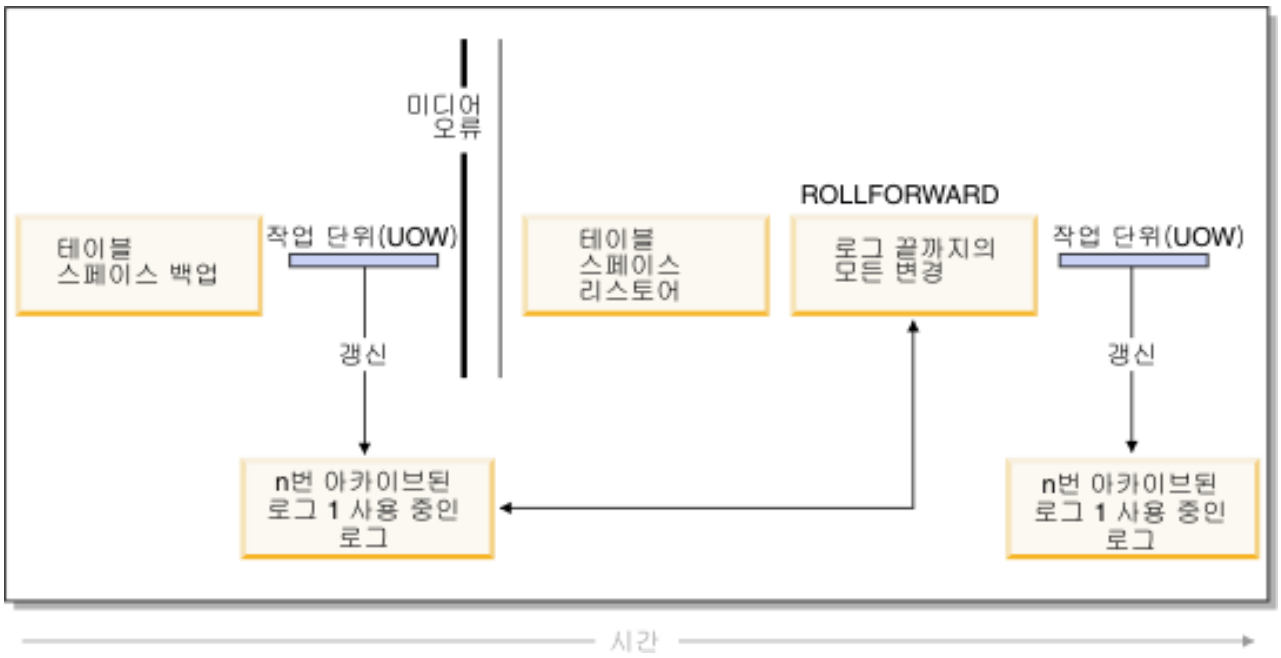


그림 20. 테이블 스페이스 롤 포워드 복구. 장기 실행 중인 트랜잭션이 있으면 사용 중인 로그가 둘 이상일 수 있습니다.

파티션된 데이터베이스 환경에서 테이블 스페이스를 *특정 시점*으로 롤 포워드하는 경우 테이블 스페이스가 있는 데이터베이스 파티션 목록을 제공하지 않아도 됩니다. DB2에서는 모든 데이터베이스 파티션에 롤 포워드 요청을 제출합니다. 즉, 테이블 스페이스가 있는 모든 데이터베이스 파티션에서 테이블 스페이스를 리스토어해야 함을 의미합니다.

파티션된 데이터베이스 환경에서 테이블 스페이스를 로그 끝으로 롤 포워드하는 경우 모든 데이터베이스 파티션에서 테이블 스페이스를 롤 포워드하지 *않으려면* 데이터베이스 파티션 목록을 제공해야 합니다. 모든 데이터베이스 파티션에서 롤 포워드 보류 상태의 모든 테이블 스페이스를 로그 끝으로 롤 포워드하려는 경우 데이터베이스 파티션 목록을 제공하지 않아도 됩니다. 디폴트로 데이터베이스 롤 포워드 요청은 모든 데이터베이스 파티션에 보냅니다.

파티션된 테이블 일부를 포함하는 테이블 스페이스를 롤 포워드하고 특정 시점으로 롤 포워드하는 경우 해당 테이블이 있는 다른 모든 테이블 스페이스도 동일한 특정 시점으로 롤 포워드해야 합니다. 그러나 파티션된 테이블 부분을 포함하는 단일 테이블 스페이스를 로그 끝으로 롤 포워드할 수 있습니다.

주: 파티션된 테이블에 접속되었거나, 분리되었거나 또는 삭제된 데이터 파티션이 있는 경우 특정 시점 롤 포워드는 이러한 데이터 파티션의 모든 테이블 스페이스를 포함해야 합니다. 파티션된 테이블에 접속되었거나, 분리되었거나 또는 삭제된 데이터 파티션이 있는지 판별하려면 SYSDATAPARTITIONS 카탈로그 테이블을 쿼리합니다.

증분식 백업 및 복구

특히 웨어하우스 같은 데이터베이스는 *크기*가 테라바이트와 페타바이트 범위로 계속 확장되고 있으므로, 이러한 데이터베이스를 백업 및 복구하는데 필요한 하드웨어 자원과 시간도 상당히 증가하고 있습니다. 대용량 데이터베이스를 다룰 때 전체 데이터베이스 및 테이블 스페이스 백업이 항상 최상의 방법은 아닙니다. 그러한 데이터베이스의 여러 사본에 대한 스토리지 요구사항이 엄청나기 때문입니다. 다음 사항을 고려하십시오.

- 웨어하우스에서 적은 양의 데이터만 변경될 경우, 전체 데이터베이스를 백업할 필요가 없습니다.
- 테이블 스페이스를 기존 데이터베이스에 추가한 다음 테이블 스페이스 백업만 수행하는 것은 위험합니다. 백업된 테이블 스페이스 외에 어떤 것도 테이블 스페이스 백업 도중에 변경되지 않는다고 보장할 수 없기 때문입니다.

이러한 문제를 해결하기 위해서 DB2는 증분 백업 및 복구를 제공합니다. *증분 백업*은 이전 백업이 수행된 이후에 갱신된 페이지만 포함하는 백업 이미지입니다. 갱신된 데이터 및 인덱스 페이지 외에도, 각 증분 백업 이미지에는 일반적으로 전체 백업 이미지에 저장되는 모든 초기 데이터베이스 메타데이터(데이터베이스 구성, 테이블 스페이스 정의, 데이터베이스 실행기록 등)도 포함됩니다.

주:

1. 테이블 스페이스에 Long 필드 또는 대형 오브젝트(LOB) 데이터가 있고 증분 백업이 작성된 경우, 이전 백업 이후에 테이블 스페이스에 있는 페이지 중에 수정된 것이 있으면 모든 Long 필드 또는 대형 오브젝트(LOB)가 백업 이미지에 복사됩니다.
2. 더티(dirty) 페이지(변경되었으나 아직 디스크에 기록되지 않은 데이터를 포함하는 페이지)를 포함하는 테이블 스페이스의 증분 백업을 수행하면, 모든 대형 오브젝트 데이터가 백업됩니다. 일반 데이터는 변경된 경우에만 백업됩니다.

두 가지 유형의 증분 백업이 지원됩니다.

- *증분*. 증분 백업 이미지는 최근의 성공한 전체 백업 조작 이후에 변경된 모든 데이터베이스 데이터의 사본입니다. 이것은 또한 시간에 걸쳐 수행된 일련의 증분 백업이 각각 이전 증분 백업 이미지의 내용을 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 스페이스에서 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 스페이스의 사본을 포함하는 최근의 성공한 백업입니다.

증분과 델타 백업 이미지의 주요 차이점은 시간 경과에 따라 계속 변경되는 오브젝트의 연속 백업이 수행될 때의 동작에 있습니다. 각 연속 증분 이미지에는 이전의 증분 이미지에 대한 전체 내용과 이전 전체 백업이 생성된 이후로 변경되었거나 새 데이터가 들어 있습니다. 델타 백업 이미지에는 이전 이미지 유형이 생성된 이후로 변경된 페이지만 포함됩니다.

데이터베이스 및 테이블 스페이스 증분 백업의 결합은 온라인 및 오프라인 모드 모두에서 조작이 허용됩니다. 백업 전략을 계획할 때는 주의하십시오. 데이터베이스 및 테이블 스페이스 증분 백업을 결합한다는 것은 데이터베이스 백업(또는 여러 테이블 스페이스의 테이블 스페이스 백업)의 선임자가 단일 이미지일 필요는 없지만 다른 시간에 수행된 이전 데이터베이스와 테이블 스페이스 백업의 고유한 세트일 수 있음을 내포하기 때문입니다.

데이터베이스 또는 테이블 스페이스를 일관성 있는 상태로 리스토어하려면, 리스토어될 전체 오브젝트(데이터베이스 또는 테이블 스페이스)의 일관성 있는 이미지로 복구 프로세스를 시작한 다음 아래에 설명된 순서로 적절한 증분 백업 이미지 각각을 적용해야 합니다.

DB2는 데이터베이스 갱신사항을 추적할 수 있도록 새 데이터베이스 구성 매개변수인 *trackmod*를 지원합니다. 이 매개변수는 두 가지의 승인된 값 중 하나를 가질 수 있습니다.

- NO. 이 구성을 사용할 경우 증분 백업이 허용되지 않습니다. 데이터베이스 페이지 갱신사항을 추적하거나 기록하지 않습니다. 이는 디폴트값입니다.
- YES. 이 구성을 사용할 경우 증분 백업이 허용됩니다. 갱신사항 추적이 사용 가능한 경우, 데이터베이스로 처음 연결될 때 변경사항이 적용됩니다. 특정 테이블 스페이스에 대해 증분 백업을 수행하려면 먼저 해당 테이블 스페이스의 전체 백업을 수행해야 합니다.

SMS 및 DMS 테이블 스페이스의 경우, 이 추적의 세분화도는 테이블 스페이스 레벨에 있습니다. 테이블 스페이스 레벨 추적에서 각 테이블 스페이스에 대한 플래그는 해당 테이블 스페이스에 백업할 페이지가 있는지 여부를 나타냅니다. 테이블 스페이스에 백업할 페이지가 없으면 해당 테이블 스페이스에 대해서는 백업 조작이 생략됩니다.

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

증분식 백업 이미지에서 리스토어

- 증분 백업 이미지로부터의 리스토어 조작은 항상 다음 단계로 구성됩니다.
  1. 증분 목표 이미지를 식별합니다.

리스토어될 최종 이미지를 판별하고 DB2 리스토어 유틸리티를 통해 증분 리스토어 조작을 요청합니다. 이 이미지는 리스토어될 최종 이미지이므로 증분 리스토어의 목표 이미지라고 합니다. 증분 목표 이미지는 RESTORE DATABASE 명령에서 TAKEN AT 매개변수를 사용하여 지정됩니다.

2. 후속 증분 백업 이미지 각각을 적용할 수 있는 기준이 될 최근의 전체 데이터베이스 또는 테이블 스페이스 이미지를 리스토어합니다.
3. 2단계에서 리스토어된 기준 이미지의 위에서, 생성된 순서대로 필요한 전체 또는 테이블 스페이스 증분 백업 이미지 각각을 리스토어합니다.
4. 1단계의 목표 이미지를 두 번 읽을 때까지 3단계 반복합니다. 목표 이미지에는 완전한 증분 리스토어 조작 동안 두 번 액세스됩니다. 첫 번째 액세스 동안 이미지에서 초기 데이터만 읽고, 어떤 사용자 데이터도 읽지 않습니다. 전체 이미지는 두 번째 액세스 동안에만 읽히고 처리됩니다.

   증분 리스토어 조작의 목표 이미지는 리스토어 조작 중에 작성될 데이터베이스에 대한 올바른 실행기록, 데이터베이스 구성 및 테이블 스페이스 정의를 사용하여 데이터베이스가 초기에 구성되도록 두 번 액세스되어야 합니다. 초기 전체 데이터베이스 백업 이미지가 작성된 이후로 테이블 스페이스가 삭제된 경우, 그 이미지에 대한 테이블 스페이스 데이터가 백업 이미지에서 읽혀지지만 증분 리스토어 처리 중에는 무시됩니다.

- 증분 백업 이미지를 리스토어하는 방법에는 두 가지가 있습니다.
  - 자동 증분 리스토어의 경우, 사용할 목표 이미지를 지정하여 RESTORE 명령을 한 번만 발행합니다. 그러면 DB2가 데이터베이스 실행기록을 사용하여 필요한 나머지 백업 이미지를 판별한 후 리스토어합니다.
  - 수동 증분 리스토어의 경우, 리스토어할 각 백업 이미지에 대해 위에서 설명한 단계에 따라 RESTORE 명령을 한 번씩 발행해야 합니다.
- **자동 증분 리스토어 예**

  자동 증분 리스토어를 사용하여 증분 백업 이미지 세트를 리스토어하려면 RESTORE DATABASE 명령에 TAKEN AT 시간소인 옵션을 지정하십시오. 리스토어할 마지막 이미지에 대한 시간소인을 사용하십시오. 예를 들어, 다음과 같습니다.

  ```
  db2 restore db sample incremental automatic taken at 20031228152133
  ```

  이렇게 하면 DB2 리스토어 유틸리티가 이 절의 시작 부분에서 설명한 각 단계를 수행합니다. 초기 처리 단계에서, 시간소인이 20001228152133인 백업 이미지가 읽히고 리스토어 유틸리티는 데이터베이스, 데이터베이스의 실행기록, 테이블 스페이스 정의가 존재하며 유효한지를 확인합니다.

  두 번째 처리 단계에서는 요청된 리스토어 조작을 수행하는데 필요한 백업 이미지 체인을 빌드하기 위해 데이터베이스 실행기록을 쿼리합니다. 어떤 이유로 이 작업을 수행할 수 없고 DB2가 필요한 이미지의 전체 체인을 빌드할 수 없는 경우, 리스토어

조작은 종료되고 오류 메시지가 리턴됩니다. 이 경우에는 자동 증분 리스토어를 할 수 없으므로 INCREMENTAL ABORT 옵션과 함께 RESTORE DATABASE 명령을 발행해야 합니다. 그렇게 하면 나머지 자원이 정리되므로 수동 증분 리스토어를 계속할 수 있습니다.

참고: PRUNE HISTORY 명령의 FORCE 옵션을 사용하지 않는 것이 좋습니다. 이 명령의 디폴트 조작은 최신의 전체 데이터베이스 백업 이미지에서 복구에 필요한 실행기록 항목을 삭제하지 못하게 하는 것이지만, FORCE 옵션을 사용하면 자동 리스토어 조작에 필요한 항목을 삭제할 수 있습니다.

세 번째 처리 단계에서 DB2는 생성된 체인의 나머지 각 백업 이미지를 리스토어합니다. 이 단계에서 오류가 발생하면 INCREMENTAL ABORT 옵션과 함께 RESTORE DATABASE 명령을 발행하여 나머지 자원을 정리해야 합니다. 그런 다음 오류를 해결할 수 있는지 판별한 후 RESTORE 명령을 다시 발행하거나 수동 증분 리스토어를 재시도해야 합니다.

- **수동 증분 리스토어 예**

  수동 증분 리스토어를 사용하여 증분 백업 이미지 세트를 리스토어하려면 RESTORE DATABASE 명령에 TAKEN AT 시간소인 옵션을 사용하여 목표 이미지를 지정한 다음 위에서 설명한 단계를 따르십시오. 예를 들어, 다음과 같습니다.

  1.
     ```
     db2 restore database sample incremental taken at <ts>
     ```
     여기서 <*ts*>는 리스토어할 최종 증분식 백업 이미지(목표 이미지)를 나타냅니다.
  2.
     ```
     db2 restore database sample incremental taken at <ts1>
     ```
     여기서 <*ts1*>은 초기 전체 데이터베이스(또는 테이블 스페이스) 이미지를 나타냅니다.
  3.
     ```
     db2 restore database sample incremental taken at  <tsX>
     ```
     여기서 <*tsX*>는 작성 시퀀스에 있는 각 증분 백업 이미지를 나타냅니다.
  4. 3단계를 반복하여, 이미지 <*ts*>를 포함할 때까지 각 증분 백업 이미지를 리스토어합니다.

  데이터베이스 리스토어 조작을 수행하고 테이블 스페이스 백업 이미지가 생성된 경우에는 테이블 스페이스 이미지의 백업 시간소인의 시간 순으로 리스토어해야 합니다.

db2ckrst 유틸리티를 사용하여 데이터베이스 실행기록을 쿼리하고 증분 리스토어에 필요한 백업 이미지 시간소인 목록을 생성할 수 있습니다. 수동 증분 리스토어를 위한 단순 리스토어 구문 또한 생성됩니다.전체 백업 레코드를 보존하고 이 유틸리티는 참조용으로만 사용하는 것이 좋습니다.

자동 증분 리스토어에 대한 제한사항

1. 리스토어하려는 백업 조작 이후에 테이블 스페이스의 이름이 변경되었고 새 이름을 사용하여 테이블 스페이스 레벨의 리스토어 조작을 발행할 경우, 데이터베이스 실행기록으로부터 필수 백업 이미지 체인이 올바르게 생성되지 않아 오류가 발생합니다(SQL2571N).

   예:

   ```
   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>

   SQL2571N 자동 리스토어를 계속할 수 없습니다.
   이유 코드: "3".
   ```

   제안되는 일시적인 해결책: 수동 증분 리스토어를 사용하십시오.

2. 데이터베이스를 삭제할 경우, 데이터베이스 실행기록이 삭제됩니다. 삭제된 데이터베이스를 리스토어할 경우, 데이터베이스 실행기록은 리스토어된 백업 시간에 해당되는 상태로 리스토어되고 그 시간 이후의 모든 실행기록 항목은 유실됩니다. 그런 다음 이러한 유실된 실행기록 항목을 사용해야 하는 자동 증분 리스토어를 수행하려고 하면, RESTORE 유틸리티가 올바르지 않은 백업 체인을 리스토어하려고 시도하여 ″시퀀스를 벗어남″ 오류가 리턴됩니다(SQL2572N).

   예:

   ```
   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>
   ```

   제안되는 일시적인 해결책:

   - 수동 증분 리스토어를 사용하십시오.
   - 자동 증분 리스토어를 발행하기 전에 먼저 <ts4> 이미지에서 실행기록 파일을 리스토어하십시오.

3. 한 데이터베이스에서 다른 데이터베이스로 백업 이미지를 리스토어한 다음 증분(델타) 백업을 수행한 경우에는 더 이상 자동 증분 리스토어를 사용하여 이 백업 이미지를 리스토어할 수 없습니다.

예:

```
db2 create db a
db2 create db b

db2 update db cfg for a using trackmod on

db2 backup db a -> ts1
db2 restore db a taken at ts1 into b

db2 backup db b incremental -> ts2

db2 restore db b incremental automatic taken at ts2

SQL2542N 제공된 소스 데이터베이스 별명 "(B)"과(와)
시간소인 "ts1"에 기초하는 데이터베이스 이미지 파일이 없습니다.
```

제안되는 일시적인 해결책:

- 다음과 같이 수동 증분 리스토어를 사용하십시오.

  ```
  db2 restore db b incremental taken at ts2
  db2 restore db a incremental taken at ts1 into b
  db2 restore db b incremental taken at ts2
  ```

- 데이터베이스 B로 수동 리스토어 조작을 수행한 후 전체 데이터베이스 백업을 발행하여 새 증분 체인을 시작하십시오.

복구 성능 최적화

다음은 복구 성능에 대해 생각할 때 고려해야 합니다.

- 로그를 별도의 디바이스에 위치시켜서 자주 갱신되는 데이터베이스의 성능을 개선할 수 있습니다. 온라인 트랜잭션 처리(OLTP) 환경의 경우, 종종 데이터 행을 저장하는 것보다 로그에 데이터를 쓰는 데 더 많은 입출력이 필요합니다. 별도의 디바이스에 로그를 위치시키면 로그 및 데이터베이스 파일 사이에 이동하는 데 필요한 디스크 암 이동이 최소화됩니다.

  또한 디스크에 어떤 다른 파일이 있는지도 고려해야 합니다. 예를 들어, 실제 메모리가 부족한 시스템에서 시스템 페이징에 사용되는 디스크로 로그를 이동하면 조정 노력이 허사가 됩니다.

  DB2는 버퍼 수, 버퍼 크기 및 병렬 처리 설정에 최적의 값을 선택하여 백업 또는 리스토어를 완료하는 데 소비되는 시간을 자동으로 최소화하려고 합니다. 값은 사용 가능한 유틸리티 힙 메모리 양, 사용 가능한 프로세서 수 및 데이터베이스 구성을 기초로 합니다.
- 리스토어 작업을 완료하는 데 필요한 시간을 줄이려면 여러 소스 디바이스를 사용하십시오.
- 테이블에 많은 양의 긴 필드 및 LOB 데이터가 포함되어 있으면 리스토어에 아주 많은 시간이 소비됩니다. 롤 포워드 복구에 데이터베이스를 사용할 수 있는 경우,

RESTORE 명령은 선택된 테이블 스페이스를 리스토어할 수 있는 기능을 제공합니다. 긴 필드와 LOB 데이터가 비즈니스에 중요한 경우, 이 테이블 스페이스에 대한 백업 태스크를 완료하는 데 필요한 시간에 대해 이 테이블 스페이스를 리스토어하는 것을 고려해야 합니다. 긴 필드와 LOB 데이터를 독립된 테이블 스페이스에 저장하면, 긴 필드와 LOB 데이터를 포함하는 테이블 스페이스를 리스토어하지 않을 것을 선택하여 리스토어 작업을 완료하는 데 필요한 시간을 줄일 수 있습니다. 별도의 소스에서 LOB 데이터를 재생성할 수 있는 경우, 테이블이 LOB 컬럼을 포함하도록 테이블을 작성하거나 변경할 때에는 NOT LOGGED 옵션을 선택하십시오. 긴 필드와 LOB 데이터를 포함하는 테이블 스페이스를 리스토어하지 않을 것을 선택하지만 테이블을 포함하는 테이블 스페이스를 리스토어해야 하는 경우, 테이블 데이터를 포함하는 모든 테이블 스페이스가 일관성 있도록 로그 끝까지 롤 포워드해야 합니다.

주: 연관된 긴 필드나 LOB 필드 없이 테이블 데이터를 포함하는 테이블 스페이스를 백업하는 경우 해당 테이블 스페이스에서 특정 시점 롤 포워드 복구를 수행할 수 없습니다. 테이블의 모든 테이블 스페이스는 동시에 같은 특정 시점까지 롤 포워드해야 합니다.

- 다음은 백업 및 리스토어 작업 둘 다에 적용됩니다.
  - 여러 개의 디바이스를 사용해야 합니다.
  - 입출력 디바이스 제어기 대역폭을 오버로드하지 마십시오.
- DB2는 여러 에이전트를 사용하여 응급 복구 및 데이터베이스 롤 포워드 복구를 수행합니다. 이 작업 중에 더 나은 성능을 예상할 수 있습니다(특히 SMP(symmetric multi-processor) 머신에서). 데이터베이스 복구 중에 여러 에이전트를 사용하면 SMP 머신에서 사용 가능한 추가 CPU를 이용할 수 있습니다.

  병렬 복구에 의해 도입되는 에이전트 유형은 db2agnsc입니다. DB2는 머신의 CPU 수를 기초로 데이터베이스 복구에 사용할 에이전트 수를 선택합니다.

  DB2는 적절할 때 동시에 적용할 수 있도록 로그 레코드를 에이전트에 분배합니다. 예를 들어, 삽입, 삭제, 갱신, 키 추가 및 키 삭제 조작과 연관되는 로그 레코드의 처리는 이 방식으로 병렬 처리될 수 있습니다. 로그 레코드는 페이지 레벨에서 병렬 처리되므로(동일한 데이터 페이지의 로그 레코드는 동일한 에이전트에 의해 처리됨) 모든 작업이 하나의 테이블에서 수행되었어도 성능이 향상됩니다.
- 복구 작업을 수행할 때, DB2는 버퍼 수, 버퍼 크기 및 병렬 처리 설정에 최적의 값을 자동으로 선택합니다. 값은 사용 가능한 유틸리티 힙 메모리 양, 사용 가능한 프로세서 수 및 데이터베이스 구성을 기초로 합니다. 따라서 시스템에서 사용 가능한 스토리지 양에 따라 UTIL_HEAP_SZ 구성 매개변수를 늘려서 추가 메모리를 할당할 것을 고려해야 합니다.

복구를 사용하는 데 필요한 특권, 권한 및 권한 부여

사용자는 특권을 사용하여 데이터베이스 자원을 작성하거나 자원에 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작과 특권을 그룹화하는 방법을 제공합니다. 이를 통해 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다. 사용자는 적절한 권한(즉, 필요한 특권 또는 권한)을 가지고 있는 오브젝트에만 액세스할 수 있습니다.

복구 유틸리티를 사용하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한을 가지고 있어야 합니다.

제 13 장 리스토어 개요

DB2 RESTORE DATABASE 명령의 가장 단순한 양식에서는 리스토어하려는 데이터베이스의 별명 이름만 지정하면 됩니다. 예를 들어, 다음과 같습니다.

```
db2 restore db sample
```

이 예에서, SAMPLE 데이터베이스가 존재하는데 RESTORE DATABASE 명령이 발행될 때 교체될 것이므로 다음 메시지가 리턴됩니다.

```
SQL2539W  경고! 백업 이미지 데이터베이스와 동일한 기존 데이터베이스로
리스토어 중입니다. 데이터베이스 파일이 삭제됩니다.
계속하시겠습니까? (y/n)
```

y를 지정하면 리스토어 작업이 성공적으로 완료되어야 합니다.

데이터베이스 리스토어 작업에서는 독점 연결이 필요합니다. 즉, 작업이 시작되면 데이터베이스에 대해 어떤 응용프로그램도 실행할 수 없으며 리스토어 유틸리티는 리스토어 작업이 성공적으로 완료될 때까지 다른 응용프로그램이 데이터베이스에 액세스할 수 없도록 합니다. 그러나 테이블 스페이스 리스토어 작업은 온라인에서 수행할 수 있습니다.

테이블 스페이스는 리스토어 작업(그 다음의 롤 포워드 복구)이 성공적으로 완료될 때까지 테이블 스페이스를 사용할 수 없습니다.

여러 테이블 스페이스에 걸쳐 있는 테이블이 있는 경우, 테이블 스페이스 세트를 함께 백업 및 리스토어해야 합니다.

부분 또는 서브세트 리스토어 작업을 수행 중인 경우, 테이블 스페이스 레벨 백업 이미지나 전체 데이터베이스 레벨 백업 이미지를 사용하여 해당 이미지에서 하나 이상의 테이블 스페이스를 선택할 수 있습니다. 백업 이미지가 작성된 시간으로부터 이 테이블 스페이스와 연관되는 모든 로그 파일이 존재해야 합니다.

32비트 레벨에서 가져온 백업 이미지로부터 데이터베이스를 64비트 레벨로 리스토어할 수는 있지만 그 반대는 불가능합니다.

데이터베이스를 백업 및 리스토어하려면 DB2 백업 및 리스토어 유틸리티를 사용해야 합니다. 한 머신에서 다른 머신으로 파일 세트를 이동하지 않는 것이 좋습니다. 데이터베이스 무결성을 위협할 수 있기 때문입니다.

리스토어 사용

미디어 또는 스토리지 오류 또는 응용프로그램 실패와 같은 문제점이 발생한 후 데이터베이스 또는 테이프 스페이스를 복구하려면 RESTORE DATABASE 명령을 사용하십시오. 데이터베이스 또는 개별 테이블 스페이스를 백업한 경우, 데이터베이스나 테이블 스페이스가 어떤 방법으로 손상되거나 훼손된 경우에 이들을 다시 작성할 수 있습니다.

기존 데이터베이스로 리스토어할 때, 리스토어될 데이터베이스에 연결되지 않아야 합니다. 리스토어 유틸리티가 지정된 데이터베이스에 자동으로 연결하며, 이 연결은 리스토어 작업의 완료 사에 종료됩니다. *새* 데이터베이스로 리스토어할 때 데이터베이스를 작성하기 위해 인스턴스 접속이 필요합니다. *새 리모트* 데이터베이스로 리스토어할 때, 먼저 새 데이터베이스가 있을 인스턴스에 접속해야 합니다. 그런 다음 서버의 코드 페이지와 지역을 지정하여 새 데이터베이스를 작성하십시오. 리스토어는 백업 이미지의 코드 페이지로 대상 데이터베이스의 코드 페이지를 겹쳐씁니다.

데이터베이스는 로컬 또는 리모트일 수 있습니다.

다음 제한사항이 리스토어 유틸리티에 적용됩니다.

- 데이터베이스가 이전에 DB2 백업 유틸리티를 사용하여 백업된 경우에만 리스토어 유틸리티를 사용할 수 있습니다.
- 인스턴스 소유자(UNIX에서) 또는 DB2ADMNS 또는 관리자 그룹의 멤버(Windows에서) 이외의 사용자가 백업 이미지를 리스토어하려고 시도하는 경우 오류가 발생합니다(SQL2061N). 다른 사용자가 백업 이미지에 액세스해야 하는 경우, 백업이 생성된 후 파일 권한을 변경해야 합니다.
- 롤 포워드 프로세스가 실행 중인 동안에는 데이터베이스 리스토어 작업을 시작할 수 없습니다.
- 테이블 스페이스가 현재 존재하는 경우 및 동일한 테이블 스페이스인 경우에만 테이블 스페이스를 기존 데이터베이스에 리스토어할 수 있습니다. 『동일』이란 테이블 스페이스가 백업 및 리스토어 작업 사이에 삭제된 후 재작성되지 않았음을 의미합니다. 디스크 및 백업 이미지의 데이터베이스가 동일해야 합니다.
- 테이블 스페이스 레벨 백업의 테이블 스페이스 레벨 리스토어를 새 데이터베이스에 실행할 수 없습니다.
- 시스템 카탈로그 테이블과 관련된 테이블 스페이스 레벨 리스토어 작업을 수행할 수 없습니다.
- 단일 데이터베이스 파티션 환경에서 작성된 백업을 기존의 파티션된 데이터베이스 환경으로 리스토어할 수 없습니다. 대신 백업을 단일 데이터베이스 파티션 환경으로 리스토어한 후 필요한 대로 데이터베이스 파티션을 추가해야 합니다.
- 하나의 코드 페이지를 갖는 백업 이미지를 다른 코드 페이지를 갖는 시스템에 리스토어할 때, 시스템 코드 페이지가 백업 이미지의 코드 페이지로 겹쳐써집니다.

- RESTORE DATABASE 명령을 사용하여 비자동 스토리지 사용 테이블 스페이스를 자동 스토리지 사용 테이블 스페이스로 변환할 수 없습니다.

리스토어 유틸리티는 명령행 처리기(CLP), 제어 센터의 데이터베이스 리스토어 노트북이나 마법사 또는 db2Restore API를 통해 호출될 수 있습니다.

다음은 CLP를 통해 실행되는 RESTORE DATABASE 명령의 예입니다.

```
db2 restore db sample from D:WDB2Backups taken at 20010320122644
```

리스토어 마법사를 열려면 다음을 수행하십시오.

1. 제어 센터에서 리스토어하려는 데이터베이스나 테이블 스페이스 오브젝트를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. 오브젝트를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 리스토어를 선택하십시오. 리스토어 마법사가 열립니다.

제어 센터에 있는 문맥 도움말 기능을 통해 세부사항 정보가 제공됩니다.

스냅샷 백업 이미지에서 리스토어

스냅샷 백업에서의 리스토어는 스토리지 디바이스의 빠른 복사 기술을 사용하여 리스토어의 데이터 복사 부분을 수행합니다.

시작하기 전에

스냅샷 백업 및 리스토어 작업을 수행하려면 스토리지 디바이스에 대해 DB2 ACS API 드라이버가 필요합니다. IBM Data Server에 통합된 드라이버는 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage SAN Volume Controller
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS 시리즈

스냅샷 백업에서 리스토어할 수 있기 전에 스냅샷 백업을 수행해야 합니다. 234 페이지의 『스냅샷 백업 수행』을 참조하십시오.

프로시저

USE SNAPSHOT 매개변수와 함께 RESTORE DATABASE 명령을 사용하거나 SQLU_SNAPSHOT_MEDIA 미디어 유형과 함께 db2Restore API를 사용하여 스냅샷 백업에서 리스토어할 수 있습니다.

RESTORE DATABASE 명령:

```
db2 restore db sample use snapshot
```

db2Restore API

```
int sampleRestoreFunction( char dbAlias[],
                           char restoredDbAlias[],
                           char user[],
                           char pswd[],
                           char workingPath[] )
{
  db2MediaListStruct mediaListStruct = { 0 };

  rmediaListStruct.locations = &workingPath;
  rmediaListStruct.numLocations = 1;
  rmediaListStruct.locationType = SQLU_SNAPSHOT_MEDIA;


  db2RestoreStruct restoreStruct = { 0 };

  restoreStruct.piSourceDBAlias = dbAlias;
  restoreStruct.piTargetDBAlias = restoredDbAlias;
  restoreStruct.piMediaList = &mediaListStruct;
  restoreStruct.piUsername = user;
  restoreStruct.piPassword = pswd;
  restoreStruct.iCallerAction = DB2RESTORE_STORDEF_NOINTERRUPT;


  struct sqlca sqlca = { 0 };


  db2Restore(db2Version900, &restoreStruct, &sqlca);

  return 0;
}
```

기존 데이터베이스로 리스토어

데이터베이스 또는 테이블 스페이스 백업 이미지를 기존 데이터베이스로 리스토어할 수 있습니다. 데이터베이스 레벨 리스토어의 경우, 백업 이미지는 별명 이름, 데이터베이스 이름 또는 데이터베이스 시드(seed)에서 기존 데이터베이스와 다를 수 있습니다. 테이블 스페이스 레벨 리스토어의 경우에는 테이블 스페이스가 현재 존재하고 동일한 테이블 스페이스인 경우에만 기존 데이터베이스로 테이블 스페이스를 리스토어할 수 있습니다. "동일"은 백업 및 리스토어 작업 사이에 테이블 스페이스가 삭제된 후 다시 작성되지 않았음을 의미합니다. 디스크 및 백업 이미지의 데이터베이스가 동일해야 합니다.

데이터베이스 *시드(seed)*는 데이터베이스 수명 동안 변경되지 않는 데이터베이스의 고유 ID입니다. 시드는 데이터베이스가 작성될 때 데이터베이스 관리 프로그램이 지정합니다. DB2는 항상 백업 이미지의 시드(seed)를 사용합니다.

기존 데이터베이스로 리스토어할 때 리스토어 유틸리티는 다음을 수행합니다.

- 기존 데이터베이스에서 테이블, 인덱스 및 긴 필드 데이터를 삭제하고 백업 이미지의 데이터로 교체합니다.
- 리스토어되는 각 테이블 스페이스의 테이블 항목을 교체합니다.
- 복구 실행기록 파일이 손상되거나 항목이 없지 않으면 파일을 보유합니다. 복구 실행기록 파일이 손상되거나 항목을 포함하지 않는 경우, 데이터베이스 관리 프로그램은 백업 이미지에서 파일을 복사합니다. 복구 실행기록 파일을 교체하려면 REPLACE HISTORY FILE 옵션과 함께 RESTORE 명령을 발행할 수 있습니다.
- 기존 데이터베이스의 인증 유형을 보유합니다.
- 기존 데이터베이스의 데이터베이스 디렉토리를 보유합니다. 디렉토리는 데이터베이스가 있는 위치와 카탈로그되는 방법을 정의합니다.
- 데이터베이스 시드(seed)를 비교합니다. 시드가 다르면 다음을 수행합니다.
  - 기존 데이터베이스와 연관되는 로그를 삭제합니다.
  - 백업 이미지에서 데이터베이스 구성 파일을 복사합니다.
  - RESTORE DATABASE 명령에 NEWLOGPATH가 지정된 경우 NEWLOGPATH를 *logpath* 데이터베이스 구성 매개변수의 값으로 설정합니다.

  데이터베이스 시드가 같으면 다음을 수행합니다.
  - 이미지가 복구할 수 없는 데이터베이스의 이미지이면 로그를 삭제합니다.
  - 현재 데이터베이스 파티션 구성 파일을 보유합니다.
  - RESTORE DATABASE 명령에 NEWLOGPATH가 지정된 경우 NEWLOGPATH를 *logpath* 데이터베이스 구성 매개변수의 값으로 설정합니다. 그렇지 않으면 현재 로그 경로를 데이터베이스 구성 파일에 복사합니다. 로그 경로의 유효성을 확인합니다. 데이터베이스에서 경로를 사용할 수 없는 경우 디폴트 로그 경로를 사용하도록 데이터베이스 구성을 변경합니다.

새 데이터베이스로 리스토어

데이터베이스를 새로 작성한 후 전체 데이터베이스 백업 이미지를 데이터베이스로 리스토어할 수 있습니다. 데이터베이스를 새로 작성하지 않으면 리스토어 유틸리티가 작성합니다.

새 데이터베이스로 리스토어하는 경우 리스토어 유틸리티는 다음을 수행합니다.

- 목표 데이터베이스 별명 매개변수를 통해 지정된 데이터베이스 별명 이름을 사용하여 데이터베이스를 새로 작성합니다. (목표 데이터베이스 별명을 지정하지 않은 경우 리스토어 유틸리티는 소스 데이터베이스 별명 매개변수를 통해 지정된 것과 같은 별명으로 데이터베이스를 작성합니다.)
- 백업 이미지에서 데이터베이스 구성 파일을 리스토어합니다.

- RESTORE DATABASE 명령에 NEWLOGPATH가 지정된 경우 NEWLOGPATH를 *logpath* 데이터베이스 구성 매개변수의 값으로 설정합니다. 로그 경로의 유효성을 확인합니다. 데이터베이스에서 경로를 사용할 수 없는 경우 디폴트 로그 경로를 사용하도록 데이터베이스 구성을 변경합니다.
- 백업 이미지에서 인증 유형을 리스토어합니다.
- 백업 이미지의 데이터베이스 디렉토리에서 주석을 리스토어합니다.
- 데이터베이스에 대한 복구 실행기록 파일을 리스토어합니다.
- 데이터베이스의 코드 페이지 위에 백업 이미지의 코드 페이지를 겹쳐씁니다.

테스트 및 프로덕션 환경에서 증분 리스토어 사용

증분 백업 및 복구에 대해 프로덕션 데이터베이스가 사용 가능하도록 설정되면, 증분 또는 델타 백업 이미지를 사용하여 테스트 데이터베이스를 작성하거나 새로 고칠 수 있습니다. 수동 또는 자동 증분 리스토어를 사용하여 이를 수행할 수 있습니다. 프로덕션 데이터베이스에서 테스트 데이터베이스로 백업 이미지를 리스토어하려면 RESTORE DATABASE 명령에서 INTO *target-database-alias* 옵션을 사용하십시오. 예를 들어 다음 백업 이미지가 있는 프로덕션 데이터베이스에서,

```
backup db prod
백업이 완료되었습니다. 이 백업 이미지에 대한 시간소인은 <ts1>입니다.

backup db prod incremental
백업이 완료되었습니다. 이 백업 이미지에 대한 시간소인은 <ts2>입니다.
```

수동 증분 리스토어의 예는 다음과 같습니다.

```
restore db prod incremental taken at <ts2> into test without
prompting
DB20000I  RESTORE DATABASE 명령이 완료되었습니다.

restore db prod incremental taken at <ts1> into test without
prompting
DB20000I  RESTORE DATABASE 명령이 완료되었습니다.

restore db prod incremental taken at <ts2> into test without
prompting
DB20000I  RESTORE DATABASE 명령이 완료되었습니다.
```

데이터베이스 TEST가 이미 존재하는 경우 리스토어 작업은 이미 있는 데이터 위에 겹쳐씁니다. 데이터베이스 TEST가 존재하지 않는 경우 리스토어 유틸리티가 작성하여 백업 이미지의 데이터로 채웁니다.

자동 증분 리스토어 작업은 데이터베이스 실행기록에 종속되므로, 리스토어 단계는 테스트 데이터베이스의 존재 여부에 따라 약간 변경됩니다. 데이터베이스 TEST에 대해 자동 증분 리스토어를 수행하려면 해당되는 실행기록에 데이터베이스 PROD에 대한 백

업 이미지 실행기록이 포함되어야 합니다. 백업 이미지에 대한 데이터베이스 실행기록은 다음의 경우 데이터베이스 TEST에 대해 이미 존재하는 데이터베이스 실행기록을 바꿉니다.

- RESTORE DATABASE 명령이 발행될 때 데이터베이스 TEST가 존재하지 않는 경우, 또는
- RESTORE DATABASE 명령이 발행될 때 데이터베이스 TEST가 존재하지만 데이터베이스 TEST 실행기록에 레코드가 없는 경우

다음 예는 존재하지 않는 데이터베이스 TEST로의 자동 증분 리스토어를 보여줍니다.

```
restore db prod incremental automatic taken at <ts2> into test without
prompting
DB20000I  RESTORE DATABASE 명령이 완료되었습니다.
```

리스토어 유틸리티는 TEST 데이터베이스를 작성하여 데이터를 채웁니다.

데이터베이스 TEST가 존재하고 데이터베이스 실행기록이 비어 있지 않은 경우, 다음과 같이 자동 증분 리스토어 작업 이전에 데이터베이스를 삭제(drop)해야 합니다.

```
drop db test
DB20000I  DROP DATABASE 명령이 완료되었습니다.

restore db prod incremental automatic taken at <ts2> into test without
prompting
DB20000I  RESTORE DATABASE 명령이 완료되었습니다.
```

데이터베이스를 삭제하지 않으려면 RESTORE DATABASE 명령을 발행하기 전에 나중 시간소인과 WITH FORCE OPTION 매개변수를 사용하여 PRUNE HISTORY 명령을 발행하십시오.

```
connect to test
데이터베이스 연결 정보

데이터베이스 서버          = <서버 ID>
SQL 권한 부여 ID           = <ID>
로컬 데이터베이스 별명     = TEST

prune history 9999 with force option
DB20000I  PRUNE 명령이 완료되었습니다.

connect reset
DB20000I  SQL 명령이 완료되었습니다.
restore db prod incremental automatic taken at <ts2> into test without
prompting
SQL2540W  리스토어는 성공적이었으나, 인터럽트 없음 모드에서 처리하는 동안
데이터베이스 리스토어 중에 경고 "2539"이(가)
발견되었습니다.
```

이 경우, RESTORE DATABASE COMMAND는 데이터베이스 TEST가 존재하지 않는 것과 같은 방식으로 작동합니다.

데이터베이스 TEST가 존재하고 데이터베이스 실행기록이 비어 있는 경우, 자동 증분 리스토어 작업 이전에 데이터베이스 TEST를 삭제(drop)하지 않아도 됩니다.

```
restore db prod incremental automatic taken at <ts2> into test without
prompting
SQL2540W  리스토어는 성공적이었으나, 인터럽트 없음 모드에서 처리하는

동안
데이터베이스 리스토어 중에 경고 "2539"이(가)
발견되었습니다.
```

먼저 전체 데이터베이스 백업을 취하지 않아도 테스트 데이터베이스의 증분 또는 델타 백업을 계속 취할 수 있습니다. 그러나 증분 또는 델타 이미지 중 하나를 리스토어해야 하는 경우 수동 증분 리스토어를 수행해야 합니다. 이는 자동 증분 리스토어 작업에서 동일한 데이터베이스 별명을 통해 작성되는 자동 증분 리스토어에서 각 백업 이미지가 리스토어되어야 하기 때문입니다.

프로덕션 백업 이미지를 사용하여 리스토어 작업을 완료한 후 테스트 데이터베이스의 전체 데이터베이스 백업을 작성하는 경우, 증분 또는 델타 백업을 취하고 수동 또는 자동 모드를 사용하여 리스토어할 수 있습니다.

경로 재지정된 리스토어 작업 수행

경로 재지정된 리스토어 작업은 다음 상황 중 하나가 발생할 때 수행됩니다.

- 소스 머신과 다른 목표 머신으로 백업 이미지를 리스토어하려고 합니다.
- 다른 물리적 위치로 테이블 스페이스 컨테이너를 리스토어하려고 합니다.
- 하나 이상의 컨테이너에 액세스할 수 없어서 리스토어 작업이 실패했습니다.

주: 경로 재지정된 리스토어는 운영 체제 사이에 데이터를 이동하기 위해 사용할 수 없습니다.

경로 재지정된 리스토어 작업 중에, 디렉토리 및 파일 컨테이너가 아직 존재하지 않으면 자동으로 작성됩니다. 데이터베이스 관리 프로그램은 디바이스 컨테이너를 자동으로 작성하지 않습니다.

DB2는 DMS 테이블 스페이스의 테이블 스페이스 컨테이너 추가, 변경 또는 제거만 지원합니다. SMS 테이블 스페이스의 경우, 경로 재지정된 리스토어는 테이블 스페이스 컨테이너 구성을 수정하기 위한 유일한 방법입니다.

RESTORE DATABASE 명령을 호출하고 REDIRECT 매개변수를 지정하거나, 제어 센터에서 데이터베이스 리스토어 마법사를 사용하여 테이블 스페이스 컨테이너를 재정의할 수 있습니다. 증분 백업 이미지의 경로 재지정된 리스토어를 호출하기 위한 프로세스는 비증분 백업 이미지의 경로 재지정된 리스토어를 호출하는 프로세스와 유사합니다. REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행하고 데이터베이스의 증분 리스토어에 사용해야 하는 백업 이미지를 지정하십시오. 또는 백업 이미지에

서 경로 재지정된 리스토어 스크립트를 생성한 후 필요에 따라 스크립트를 수정할 수 있습니다. 296 페이지의 『자동으로 생성된 스크립트를 사용하여 경로 재지정된 리스토어 수행』의 내용을 참조하십시오.

컨테이너 경로 재지정은 테이블 스페이스 컨테이너 관리를 위한 상당한 유연성을 제공합니다. 예를 들어, 컨테이너를 SMS 테이블 스페이스에 추가하는 것이 지원되지 않아도, 경로 재지정된 리스토어 작업을 호출할 때 추가 컨테이너를 지정하여 이를 수행할 수 있습니다.

예

경로 재지정된 리스토어 작업이 중개 테이블 스페이스 컨테이너 정의 단계가 있는 2단계 데이터베이스 리스토어 프로세스로 구성됩니다.

1. REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.
2. 리스토어된 데이터베이스에 대해 테이블 스페이스 컨테이너를 정의하려면 SET TABLESPACE CONTAINERS 명령을 사용하십시오(목표 시스템에서 테이블 스페이스 위치를 지정함).
3. RESTORE DATABASE 명령을 다시 발행하십시오. 이때 CONTINUE 옵션을 지정합니다.

다음 예는 데이터베이스 SAMPLE에서 경로 재지정된 리스토어를 수행하는 방법을 보여줍니다.

```
db2 restore db sample redirect without prompting
SQL1277W 경로 재지정된 리스토어 조작을 수행 중입니다.
이제 테이블 스페이스 구성을 볼 수 있으며 자동 스토리지를 사용하지 않는
테이블 스페이스에는 컨테이너를 다시 구성할 수 있습니다.

DB20000I RESTORE DATABASE 명령이 완료되었습니다.

db2 set tablespace containers for 2 using (path 'userspace1.0', path
'userspace1.1')
DB20000I SET TABLESPACE CONTAINERS 명령이 완료되었습니다

db2 restore db sample continue
DB20000I RESTORE DATABASE 명령이 완료되었습니다.
```

자동으로 생성된 스크립트를 사용하여 데이터베이스를 리스토어하여 테이블 스페이스 컨테이너 재정의

데이터베이스를 리스토어할 때, 리스토어 유틸리티는 실제 컨테이너 레이아웃이 백업되었을 때의 데이터베이스 레이아웃과 동일할 것으로 가정합니다. 실제 컨테이너의 *크기* 나 위치를 변경해야 하는 경우 REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행해야 합니다. 이 옵션을 사용하려면 백업 이미지에 저장된 실제 컨테이너의 위치를 지정하고 변경할 비자동 테이블 스페이스마다 전체 컨테이너 세트를 제공해야 합니다. 백업할 때 컨테이너 정보를 캡처할 수 있지만 이는 번거로울 수 있습니다.

경로 재지정된 리스토어를 쉽게 수행하기 위해, 리스토어 유틸리티에서 REDIRECT 옵션과 GENERATE SCRIPT 옵션을 사용하여 RESTORE DATABASE 명령을 발행하여 기존 백업 이미지에서 경로 재지정된 리스토어 스크립트를 생성할 수 있습니다. 리스토어 유틸리티는 백업 이미지를 조사하고 백업 이미지에서 컨테이너 정보를 추출한 후 자세한 모든 컨테이너 정보를 포함하는 CLP 스크립트를 생성합니다. 그러면 사용자는 스크립트에서 경로나 컨테이너 크기를 수정한 후 CLP 스크립트를 실행하여 새 컨테이너 세트로 데이터베이스를 재작성할 수 있습니다. 사용자가 생성하는 스크립트는 사용자가 백업 이미지만 가지고 있고 컨테이너 레이아웃을 모르는 경우에도 데이터베이스를 리스토어하는 데 사용할 수 있습니다. 스크립트는 클라이언트에서 작성됩니다. 스크립트를 기준으로 사용하여, 리스토어된 데이터베이스에서 로그 파일 및 컨테이너에 필요한 스페이스를 결정하고 이에 따라 로그 파일 및 컨테이너 경로를 변경할 수 있습니다.

생성된 스크립트는 네 개의 섹션으로 구성됩니다.

초기화 첫 번째 섹션은 명령 옵션을 설정하고 명령이 실행될 데이터베이스 파티션을 지정합니다. 다음은 첫 번째 섹션의 예입니다.

```
UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM 0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
```

where

- S ON은 명령 오류가 발생하는 경우 명령 실행이 중지되어야 함을 지정합니다.
- Z ON SAMPLE_NODE0000.out는 출력 방향을 <dbalias>_NODE<dbpartitionnum>.out 파일로 보내야 함을 지정합니다.
- V ON은 현재 명령이 표준 출력에 인쇄되어야 함을 지정합니다.

  파티션된 데이터베이스 환경에서 스크립트를 실행할 때 스크립트 명령이 실행될 데이터베이스 파티션을 지정하는 것은 중요합니다.

REDIRECT 옵션이 있는 RESTORE 명령

두 번째 섹션은 RESTORE 명령을 시작하고 REDIRECT 옵션을 사용합니다. 이 섹션은 REDIRECT 옵션과 함께 사용할 수 없는 옵션을 제외하고 모든 RESTORE 명령 옵션을 사용할 수 있습니다. 다음은 두 번째 섹션의 예입니다.

```
RESTORE DATABASE SAMPLE
-- USER '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050906194027
-- DBPATH ON '<target-directory>'
INTO SAMPLE
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/SQLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
```

```
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
```

테이블 스페이스 정의

이 섹션에는 명령행에 지정되거나 백업 이미지에 있는 각 테이블 스페이스의 테이블 스페이스 정의가 포함되어 있습니다. 테이블 스페이스마다 테이블 스페이스의 이름, 유형 및 크기에 대한 정보를 포함하는 주석 블록으로 구성된 섹션이 있습니다. 정보는 테이블 스페이스 스냅샷과 같은 형식으로 제공됩니다. 제공되는 정보를 사용하여 테이블 스페이스에 필요한 크기를 판별할 수 있습니다. 자동 스토리지를 사용하여 작성된 테이블 스페이스의 출력을 보는 경우 SET TABLESPACE CONTAINERS 절은 표시되지 않습니다. 다음은 테이블 스페이스 정의 섹션의 예입니다.

```
-- *****************************************************************************
-- ** 테이블 스페이스 이름                        = SYSCATSPACE
-- **   테이블 스페이스 ID                        = 0
-- **   테이블 스페이스 유형                      = 시스템 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형               = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)       = 4096
-- **   테이블 스페이스 Extent 크기(페이지)       = 32
-- **   자동 스토리지 사용                        = 아니오
-- **   총 페이지 수                              = 5572
-- *****************************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
);
-- *****************************************************************************
-- ** 테이블 스페이스 이름                        = TEMPSPACE1
-- **   테이블 스페이스 ID                        = 1
-- **   테이블 스페이스 유형                      = 시스템 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형               = 시스템 임시 데이터
-- **   테이블 스페이스 페이지 크기(바이트)       = 4096
-- **   테이블 스페이스 Extent 크기(페이지)       = 32
-- **   자동 스토리지 사용                        = 아니오
-- **   총 페이지 수                              = 0
-- *****************************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- *****************************************************************************
-- ** 테이블 스페이스 이름                        = DMS
-- **   테이블 스페이스 ID                        = 2
-- **   테이블 스페이스 유형                      = 데이터베이스 관리 스페이스
-- **   테이블 스페이스 컨텐츠 유형               = 모든 데이터
-- **   테이블 스페이스 페이지 크기(바이트)       = 4096
-- **   테이블 스페이스 Extent 크기(페이지)       = 32
-- **   자동 스토리지 사용                        = 아니오
-- **   자동 크기 조정 사용 가능                  = 아니오
-- **   총 페이지 수                              = 2000
-- **   사용 가능한 페이지 수                     = 1960
-- **   상위 워터 마크(페이지)                    = 96
-- *****************************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
```

```
USING (
  FILE   '/tmp/dms1'                                    1000
, FILE   '/tmp/dms2'                                    1000
);
```

CONTINUE 옵션이 있는 RESTORE 명령

최종 섹션은 CONTINUE 옵션과 함께 RESTORE 명령을 발행하여 경로 재지정된 리스토어를 완료합니다. 다음은 최종 섹션의 예입니다.

```
RESTORE DATABASE SAMPLE CONTINUE;
```

자동으로 생성된 스크립트를 사용하여 경로 재지정된 리스토어 수행

경로 재지정된 리스토어 작업을 수행할 때 백업 이미지에 저장된 실제 컨테이너의 위치를 지정하고 변경될 각 테이블 스페이스에 대한 컨테이너의 전체 세트를 제공해야 합니다. 다음 프로시저를 사용하여 기존 백업 이미지를 기초로 경로 재지정된 리스토어 스크립트를 생성하고, 생성된 스크립트를 수정한 후 스크립트를 실행하여 경로 재지정된 리스토어를 수행하십시오.

데이터베이스가 이전에 DB2 백업 유틸리티를 사용하여 백업된 경우에만 경로 재지정 리스토어를 수행할 수 있습니다.

- 데이터베이스가 존재하는 경우 스크립트를 생성하려면 데이터베이스에 연결할 수 있어야 합니다. 그러므로 데이터베이스가 업그레이드 또는 응급 복구가 필요한 경우 이것은 경로 재지정 리스토어 스크립트를 생성하려고 시도하기 전에 수행되어야 합니다.
- 파티션된 데이터베이스 환경에서 작업 중이고 목표 데이터베이스가 존재하지 않는 경우, 모든 데이터베이스 파티션에서 동시에 경로 재지정 리스토어 스크립트를 생성하는 명령을 실행할 수 없습니다. 대신 경로 재지정 리스토어 스크립트를 생성하는 명령은 카탈로그 파티션에서 시작하여 한 번에 하나의 데이터베이스 파티션에서 실행되어야 합니다.

  또는 먼저 목표 데이터베이스와 동일한 이름이 있는 더미 데이터베이스를 작성할 수 있습니다. 더미 데이터베이스가 작성된 후 모든 데이터베이스 파티션에서 동시에 경로 재지정 리스토어 스크립트를 생성할 수 있습니다.
- RESTORE 명령을 사용하여 스크립트를 생성할 때 REPLACE EXISTING 옵션을 지정하는 경우에도 주석 처리된 스크립트에 REPLACE EXISTING 옵션이 나타납니다.
- 보안상의 이유 때문에 암호가 생성된 스크립트에 나타나지 않습니다. 암호를 수동으로 채워야 합니다.
- 제어 센터의 리스토어 마법사를 사용하여 경로 재지정된 리스토어에 대한 스크립트를 생성할 수 없습니다.

스크립트를 사용하여 경로 재지정 리스토어를 수행하려면 다음을 수행하십시오.

1. 리스토어 유틸리티를 사용하여 경로 재지정 리스토어 스크립트를 생성하십시오. 리스토어 유틸리티는 명령행 처리기(CLP) 또는 db2Restore API를 통해 호출할 수 있습니다. 다음은 REDIRECT 옵션 및 GENERATE SCRIPT 옵션을 사용하는 RESTORE DATABASE 명령의 예입니다.

   ```
   db2 restore db test from /home/jseifert/backups taken at 20050304090733
            redirect generate script test_node0000.clp
   ```

   이것은 test_node0000.clp라는 클라이언트에 경로 재지정 리스토어 스크립트를 작성합니다.
2. 텍스트 편집기에서 경로 재지정 리스토어 스크립트를 열어서 필요한 수정을 수행하십시오. 다음을 수정할 수 있습니다.
   - 리스토어 옵션
   - 자동 스토리지 경로
   - 컨테이너 레이아웃 및 경로
3. 수정된 경로 재지정 리스토어 스크립트를 실행하십시오. 예를 들어, 다음과 같습니다.

   ```
   db2 -tvf test_node0000.clp
   ```

데이터베이스 재빌드

데이터베이스 재빌드는 리스토어 작업 세트를 사용하여 데이터베이스 또는 해당 테이블 스페이스 서브세트를 리스토어하는 프로세스입니다. 데이터베이스 재빌드로 제공되는 기능은 DB2를 더 강력하고 많은 기능을 갖도록 만들고 사용자에게 더 완벽한 복구 솔루션을 제공합니다.

테이블 스페이스 백업 이미지를 통해 데이터베이스를 재빌드할 수 있는 능력은 더 이상 많은 전체 데이터베이스 백업을 수행하지 않아도 된다는 것을 의미합니다. 데이터베이스 *크기*가 증가함에 따라, 전체 데이터베이스 백업을 수행할 기회는 제한됩니다. 대안으로 테이블 스페이스 백업을 사용하면, 더 이상 전체 데이터베이스 백업을 자주 수행하지 않아도 됩니다. 대신, 더 자주 테이블 스페이스 백업을 수행하고 재해가 발생하는 경우 로그 파일과 함께 이 백업을 사용하도록 계획할 수 있습니다.

복구 상황에서, 테이블 스페이스 서브세트를 다른 것보다 빨리 온라인으로 가져와야 하는 경우 재빌드를 사용하여 이를 수행할 수 있습니다. 테이블 스페이스 서브세트만 온라인으로 가져오는 기능은 특히 테스트 및 프로덕션 환경에서 유용합니다.

데이터베이스 재빌드에는 잠재적으로 많은 일련의 리스토어 작업이 관련됩니다. 재빌드 조작은 데이터베이스 이미지나 테이블 스페이스 이미지, 또는 둘 다를 사용할 수 있습니다. 전체 백업이나 증분 백업, 또는 둘 다를 사용할 수 있습니다. 초기 리스토어 작업은 목표 이미지를 리스토어합니다. 이 이미지는 리스토어할 수 있는 데이터베이스의

구조를 정의합니다(예: 테이블 스페이스 세트 및 데이터베이스 구성). 복구 가능한 데이터베이스의 경우 재빌드하면 나중에 오프라인에서 복구될 수 있는 테이블 스페이스를 보존하면서, 연결 가능하고, 온라인에서 가지고 있어야 하는 테이블 스페이스 서브세트를 포함하는 데이터베이스를 빌드할 수 있습니다.

데이터베이스를 재빌드하기 위해 사용하는 방법은 복구 가능 여부에 따라 결정됩니다.

- 데이터베이스가 복구 가능하면 다음 방법 중 하나를 사용하십시오.
  - 전체 또는 증분 데이터베이스나 테이블 스페이스 백업 이미지를 목표로 사용하여, REBUILD 옵션으로 목표 이미지에서만 다른 테이블 스페이스와 SYSCATSPACE를 리스토어하여 데이터베이스를 재빌드합니다. 그런 다음 특정 시점까지 데이터베이스를 롤 포워드할 수 있습니다.
  - 전체 또는 증분 데이터베이스나 테이블 스페이스 백업 이미지를 목표로 사용하여, REBUILD 옵션으로 목표 이미지가 리스토어될 때 데이터베이스에 정의된 테이블 스페이스 세트를 지정하여 데이터베이스를 재빌드합니다. SYSCATSPACE는 이 세트의 일부여야 합니다. 이 조작은 목표 이미지에 정의된, 지정된 테이블 스페이스를 리스토어한 후 복구 실행기록 파일을 사용하여 자동으로 목표 이미지에는 없는 남아 있는 테이블 스페이스에 대한 다른 필요한 백업 이미지를 찾아서 리스토어합니다. 리스토어가 완료되면 특정 시점까지 데이터베이스를 롤 포워드하십시오.
- 데이터베이스가 복구 불가능한 경우
  - 전체 또는 증분 데이터베이스 백업 이미지를 목표로 사용하여, 적절한 REBUILD 구문으로 목표 이미지에서 다른 테이블 스페이스와 SYSCATSPACE를 리스토어하여 데이터베이스를 재빌드합니다. 리스토어가 완료되면 데이터베이스에 연결할 수 있습니다.

목표 이미지 지정

데이터베이스 재빌드를 수행하려면 RESTORE 명령을 발행하고 리스토어 작업의 목표로 사용하는 최근의 백업 이미지를 지정하여 시작합니다. 이 이미지는 리스토어할 수 있는 테이블 스페이스, 데이터베이스 구성 및 로그 시퀀스를 비롯하여 리스토어할 데이터베이스의 구조를 정의하므로 재빌드 작업의 목표 이미지라고 합니다. 재빌드 목표 이미지는 RESTORE DATABASE 명령에서 TAKEN AT 매개변수를 사용하여 지정합니다. 목표 이미지는 어떤 유형의 백업도 가능합니다(전체, 테이블 스페이스, 증분, 온라인 및 오프라인). 목표 이미지가 작성될 때 정의된 테이블 스페이스는 데이터베이스 재빌드에 사용 가능한 테이블 스페이스가 됩니다.

다음 방법 중 하나를 사용하여 리스토어할 테이블 스페이스를 지정해야 합니다.

- 데이터베이스에 정의된 모든 테이블 스페이스를 리스토어할 것을 지정하고 제외할 테이블 스페이스가 있으면 예외 목록을 제공하십시오.

- 목표 이미지에 있는 사용자 데이터를 가지고 있는 모든 테이블 스페이스를 리스토어할 것을 지정하고 제외할 테이블 스페이스가 있으면 예외 목록을 제공하십시오.
- 리스토어할 데이터베이스에 정의된 테이블 스페이스 목록을 지정하십시오.

재빌드된 데이터베이스가 포함할 테이블 스페이스를 알면, 적절한 REBUILD 옵션과 함께 RESTORE 명령을 발행하여 사용할 목표 이미지를 지정하십시오.

재빌드 단계

적절한 REBUILD 옵션과 함께 RESTORE 명령을 발행하고 목표 이미지가 성공적으로 리스토어된 경우, 데이터베이스는 재빌드 단계에 있는 것으로 간주됩니다. 목표 이미지가 리스토어된 후, 발생하는 추가 테이블 스페이스 리스토어는 재빌드된 데이터베이스에 정의된 대로 기존 테이블 스페이스로 데이터를 리스토어합니다. 이 테이블 스페이스는 재빌드 조작 완료 시 데이터베이스에 대해 롤 포워드됩니다.

적절한 REBUILD 옵션과 함께 RESTORE 명령을 발행하는데 데이터베이스가 존재하지 않는 경우, 목표 이미지의 속성을 기초로 새 데이터베이스가 작성됩니다. 데이터베이스가 존재하지 않으면 재빌드 단계가 시작됨을 알리는 경고 메시지를 수신합니다. 사용자에게 재빌드 조작을 계속할 것인지 묻습니다.

재빌드 조작은 목표 이미지로부터 모든 초기 메타데이터를 리스토어합니다. 여기에는 데이터베이스에 속하고 테이블 스페이스 데이터나 로그 파일에는 속하지 않는 모든 데이터가 포함됩니다. 초기 메타데이터의 예는 다음과 같습니다.

- 테이블 스페이스 정의
- 실행기록 파일(관리 조작을 기록하는 데이터베이스 파일)

재빌드 조작은 데이터베이스 구성도 리스토어합니다. 목표 이미지는 재빌드 단계에서 남아 있는 리스토어에 사용할 수 있는 이미지를 판별하는 로그 체인을 설정합니다. 동일한 로그 체인에 있는 이미지만 사용할 수 있습니다.

데이터베이스가 이미 디스크에 존재하고 실행기록 파일이 목표 이미지에서 제공되도록 하려면, REPLACE HISTORY FILE 옵션을 지정해야 합니다. 이 때 디스크의 실행기록 파일은 데이터베이스 재빌드에 필요한 남아 있는 이미지를 찾기 위한 자동 논리에 사용됩니다.

목표 이미지가 리스토어되면

- 데이터베이스가 복구 가능한 경우, 데이터베이스는 롤 포워드 보류 상태에 놓이고 사용자가 리스토어하는 모든 테이블 스페이스도 롤 포워드 보류 상태에 놓입니다. 데이터베이스에 정의되었지만 리스토어되지 않은 모든 테이블 스페이스는 리스토어 보류 상태에 놓입니다.

- 데이터베이스가 복구 불가능한 경우, 데이터베이스와 리스토어된 테이블 스페이스는 정상적인 상태가 됩니다. 리스토어되지 않은 테이블 스페이스는 더 이상 복구될 수 없으므로 삭제 보류 상태에 놓입니다. 이 유형의 데이터베이스는 재빌드 단계가 완료된 것입니다.

복구 가능한 데이터베이스의 경우, 재빌드 단계는 첫 번째 ROLLFORWARD DATABASE 명령이 발행되고 롤 포워드 유틸리티가 로그 레코드 처리를 시작할 때 종료됩니다. 로그 레코드 처리를 시작한 후 롤 포워드 작업이 실패하고 그 다음으로 리스토어 작업이 발행된 경우, 리스토어는 재빌드 단계의 일부로 간주되지 않습니다. 이와 같은 리스토어는 재빌드 단계의 일부가 아닌 일반 테이블 스페이스 리스토어로 간주해야 합니다.

자동 처리

목표 이미지가 리스토어된 후, 리스토어 유틸리티는 리스토어해야 하는 남아 있는 테이블 스페이스가 있는지 판별합니다. 있으면 REBUILD 옵션과 함께 RESTORE DATABASE 명령을 실행하는 데 사용된 것과 같은 연결을 사용하여 리스토어됩니다. 유틸리티는 디스크의 실행기록 파일을 사용하여, 리스토어해야 하는 남아 있는 테이블 스페이스 각각을 포함하는 목표 이미지 이전에 취한 최근 백업 이미지를 찾습니다. 리스토어 유틸리티는 실행기록 파일에 지정된 백업 이미지 위치 데이터를 사용하여 자동으로 이 이미지를 각각 리스토어합니다. 테이블 스페이스 레벨 리스토어인 이러한 연속 리스토어는 오프라인에서만 수행할 수 있습니다. 선택된 이미지가 현재 로그 체인에 속해 있지 않으면 오류가 리턴됩니다. 이 이미지에서 리스토어된 각 테이블 스페이스는 롤 포워드 보류 상태에 놓입니다.

리스토어 유틸리티는 자동으로 필요한 모든 테이블 스페이스테이블 스페이스를 리스토어하려고 합니다. 어떤 경우에는, 실행기록 파일의 문제점으로 인해, 또는 필요한 이미지 중 하나를 리스토어하는 중에 오류가 발생하여 일부 테이블 스페이스를 리스토어할 수 없습니다. 이와 같은 경우, 수동으로 재빌드를 완료하거나 문제점을 정정하고 재빌드를 다시 발행할 수 있습니다.

자동 재빌드를 성공적으로 완료할 수 없는 경우, 리스토어 유틸리티는 남아 있는 리스토어 단계에 대해 수집된 정보를 진단 로그(db2diag 로그 파일)에 기록합니다. 이 정보를 사용하여 수동으로 재빌드를 완료할 수 있습니다.

데이터베이스가 재빌드되는 경우 재빌드 프로세스의 일부인 테이블 스페이스에 속하는 컨테이너만 획득됩니다.

경로 재지정된 리스토어를 통해 컨테이너를 재정의해야 하는 경우, 남아 있는 리스토어 및 연속 롤 포워드 작업에 대해 새 컨테이너의 크기와 새 경로를 설정해야 합니다.

이 남아 있는 이미지 중 하나에서 리스토어되는 테이블 스페이스의 데이터를 새 컨테이너 정의에 맞출 수 없는 경우, 테이블 스페이스는 리스토어 보류 상태가 되고 리스토

어 끝에서 경고 메시지가 리턴됩니다. 진단 로그에서 문제점에 대한 추가 정보를 찾을 수 있습니다.

재빌드 단계 완료

의도한 모든 테이블 스페이스가 리스토어되면 데이터베이스 구성에 따라 두 가지 옵션이 제공됩니다. 데이터베이스가 복구 불가능한 경우, 데이터베이스는 연결 가능하게 되고 리스토어된 모든 테이블 스페이스는 온라인이 됩니다. 삭제 보류 상태에 있는 테이블 스페이스는 더 이상 복구될 수 없으므로 데이터베이스에 대해 추가 백업이 수행될 경우 삭제해야 합니다.

데이터베이스가 복구 가능한 경우, 롤 포워드 명령을 발행하여 리스토어된 테이블 스페이스를 온라인으로 가져올 수 있습니다. SYSCATSPACE가 리스토어되지 않았으면, 롤 포워드가 실패하고 해당 테이블 스페이스는 롤 포워드 작업을 시작하기 전에 먼저 리스토어해야 합니다. 이는 재빌드 단계 중에 SYSCATSPACE를 리스토어해야 함을 의미합니다.

주: 파티션된 데이터베이스 환경에서, SYSCATSPACE는 비카탈로그 파티션에 존재하지 않으므로 그 곳에서 재빌드할 수 없습니다. 그러나 카탈로그 파티션에서, SYSCATSPACE는 재빌드된 테이블 스페이스 중 하나여야 합니다. 그렇지 않으면 롤 포워드 작업이 실패합니다.

데이터베이스를 롤 포워드하면 데이터베이스가 롤 포워드 보류 상태를 벗어나서 롤 포워드 보류 상태에 있는 테이블 스페이스를 롤 포워드합니다. 롤 포워드 유틸리티는 리스토어 보류 상태에 있는 테이블 스페이스에 대해 작동하지 않습니다.

롤 포워드 작업의 중지 시간은 재빌드 단계 중에 리스토어된 최근 백업 이미지의 종료 시간보다 늦은 시간이어야 합니다. 다른 시간이 제공되는 오류가 발생합니다. 롤 포워드 작업이 리스토어된 가장 오래된 이미지의 백업 시간에 도달할 수 없는 경우, 롤 포워드 유틸리티는 데이터베이스를 일관성 있는 시점까지 가져올 수 없어서 롤 포워드가 실패하게 됩니다.

사용할 롤 포워드 유틸리티에 사용 가능한 가장 빠른 백업 이미지와 가장 최근의 백업 이미지 사이의 시간 틀에 해당되는 모든 로그 파일을 가지고 있어야 합니다. 필요한 로그는 가장 빠른 백업 이미지에서, 목표 이미지의 절단 배열에 의해 정의된 목표 백업 이미지로의 로그 체인 뒤에 오는 로그들입니다. 그렇지 않으면 롤 포워드 작업은 실패합니다. 목표 이미지보다 더 최근의 백업 이미지기 재빌드 단계 중에 리스토어된 경우, 목표 이미지에서 리스토어된 가장 최근의 백업 이미지까지의 추가 로그가 필요합니다. 로그가 사용 가능하지 않으면, 롤 포워드 작업은 로그에 의해 도달되지 않은 해당 테이블 스페이스를 리스토어 보류 상태에 놓습니다. LIST HISTORY 명령을 발행하여 롤 포워드에 필요할 로그 범위를 가지고 있는 리스토어 재빌드 항목을 표시합니다.

올바른 로그 파일이 사용 가능해야 합니다. 롤 포워드 유틸리티에 의존하여 로그를 검색하는 경우, DB2 Log Manager가 로그 파일이 검색될 수 있는 위치를 표시하도록 구성되어 있는지 확인해야 합니다. 로그 경로 또는 아카이브 경로가 변경된 경우 ROLLFORWARD DATABASE 명령의 OVERFLOW LOG PATH 옵션을 사용해야 합니다.

롤 포워드 명령이 성공적으로 완료될 때 데이터베이스를 사용 가능하도록 만들려면 ROLLFORWARD DATABASE 명령의 AND STOP 옵션을 사용하십시오. 이 때 데이터베이스는 더 이상 롤 포워드 보류 상태가 아닙니다. 롤 포워드 작업이 시작되었지만 성공적으로 완료되기 전에 오류가 발생하면, 롤 포워드 작업은 실패 시점에서 중지되고 오류가 리턴됩니다. 데이터베이스는 롤 포워드 보류 상태에서 유지됩니다. 문제점을 정정하기 위한 단계를 수행한 후(예를 들어, 로그 파일 수정) 다른 롤 포워드 작업을 발행하여 처리를 계속하십시오.

오류를 수정할 수 없으면, ROLLFORWARD STOP 명령을 발행하여 데이터베이스를 실패 시점에서 정지할 수 있습니다. STOP 옵션이 사용되면 로그에서 이 시점을 넘어선 데이터는 더 이상 사용할 수 없게 됩니다. 데이터베이스는 이 시점에서 가동되고 복구된 테이블 스페이스는 온라인이 됩니다. 아직 복구되지 않은 테이블 스페이스는 리스토어 보류 상태가 됩니다. 데이터베이스는 일반 상태가 됩니다.

리스토어 보류 상태에 있는 남아 있는 테이블 스페이스를 복구할 최상의 방법을 결정해야 합니다. 이는 테이블 스페이스의 새 리스토어 및 롤 포워드를 수행하거나 전체 재빌드 조작을 다시 발행하여 수행할 수 있습니다. 이는 발견된 문제점 유형에 따라 결정됩니다. SYSCATSPACE가 리스토어 보류 상태에 있는 테이블 스페이스 중 하나인 상황에서는 데이터베이스가 연결 가능하지 않습니다.

재빌드 및 테이블 스페이스 컨테이너

재빌드 중에, 재빌드 프로세스의 일부인 테이블 스페이스만 컨테이너를 획득합니다. 각 테이블 스페이스에 속하는 컨테이너는 테이블 스페이스 사용자 데이터가 이미지 외부에서 리스토어될 때 획득됩니다.

목표 이미지가 리스토어될 때, 백업 시점에 데이터베이스에 알려진 각 테이블 스페이스는 해당되는 정의만 리스토어됩니다. 이는 재빌드로 작성된 데이터베이스가 백업했을 때와 동일한 테이블 스페이스의 지식을 갖게 됨을 의미합니다. 목표 이미지에서 리스토어되는 사용자 데이터도 가지고 있어야 하는 테이블 스페이스의 경우, 해당 컨테이너도 이 때 획득됩니다.

중간 테이블 스페이스 리스토어를 통해 리스토어되는 남아 있는 테이블 스페이스는 테이블 스페이스 데이터를 포함하는 이미지가 리스토어될 때 획득된 해당 컨테이너를 갖게 됩니다.

경로 재지정된 리스토어로 재빌드

경로가 재지정된 리스토어에서, 모든 테이블 스페이스 컨테이너는 목표 이미지의 리스토어 중에 정의해야 합니다. REDIRECT 옵션을 지정하는 경우, 제어는 테이블 스페이스 컨테이너를 재정의할 수 있도록 다시 사용자에게 부여됩니다. SET TABLESPACE CONTAINERS 명령을 사용하여 테이블 스페이스 컨테이너를 재정의한 경우에는 이 때 새 컨테이너가 획득됩니다. 재정의하지 않은 테이블 스페이스 컨테이너는 테이블 스페이스 사용자 데이터가 이미지 외부에서 리스토어될 때 정상적으로 획득됩니다.

리스토어되는 테이블 스페이스의 데이터를 새 컨테이너 정의에 맞출 수 없는 경우, 테이블 스페이스는 리스토어 보류 상태가 되고 리스토어 끝에서 경고(SQL2563W)가 사용자에게 리턴됩니다. DB2 진단 로그에 문제점을 자세히 설명하는 메시지가 있을 것입니다.

재빌드 및 임시 테이블 스페이스

일반적으로, DB2 백업 이미지는 다음 구성요소로 구성됩니다.

- 초기 데이터베이스 메타데이터(예: 테이블 스페이스 정의, 데이터베이스 구성 파일 및 실행기록 파일)
- BACKUP 유틸리티에 지정된 임시가 아닌 테이블 스페이스에 대한 데이터
- 최종 데이터베이스 메타데이터(예: 로그 파일 헤더
- 로그 파일(LNCLUDE LOGS 옵션이 지정되지 않은 경우)

모든 백업 이미지에서, 데이터베이스 또는 테이블 스페이스 백업, 전체 또는 증분(델타) 백업 여부에 관계없이 이 코어 구성요소들은 항상 발견됩니다.

데이터베이스 백업 이미지는 위의 구성요소 모두와, 백업 시 데이터베이스에 정의된 모든 테이블 스페이스에 대한 데이터를 포함합니다.

테이블 스페이스 백업 이미지는 항상 위에 나열된 데이터베이스 메타데이터를 포함하지만, 백업 유틸리티에 지정되는 테이블 스페이스에 대한 데이터만 포함합니다.

임시 테이블 스페이스는 임시가 아닌 테이블 스페이스와 다르게 처리됩니다. 임시 테이블 스페이스 데이터는 백업되지 않지만 존재 자체는 데이터베이스 Framework에 중요합니다. 임시 테이블 스페이스 데이터가 백업되지 않아도, 임시 테이블 스페이스는 데이터베이스의 일부로 간주되므로 특히 백업 이미지와 함께 저장되는 메타데이터에서 표시됩니다. 이로서 백업 이미지에 있는 것처럼 보이게 됩니다. 또한 테이블 스페이스 정의는 임시 테이블 스페이스의 존재에 대한 정보는 보유합니다.

백업 이미지에 임시 테이블 스페이스의 데이터가 포함된 적이 없어도, 데이터베이스 재빌드 작업 중 목표 이미지가 리스토어될 때(이미지 유형에 관계없이) 임시 테이블 스페이스도 리스토어됩니다(해당 컨테이너가 획득되고 할당된 것으로 감지되는 경우에만). 컨

테이너 획득 및 할당은 재빌드 프로세스의 일부로, 자동으로 수행됩니다. 결과적으로, 데이터베이스를 재빌드할 때 임시 테이블 스페이스를 제외할 수 없습니다.

데이터베이스 재빌드에 대한 목표 이미지 선택

재빌드 목표 이미지는 리스토어 작업의 시작점으로 사용하려는 최근 백업 이미지여야 합니다. 이 이미지는 리스토어할 수 있는 테이블 스페이스, 데이터베이스 구성 및 로그 시퀀스를 비롯하여 리스토어할 데이터베이스의 구조를 정의하므로 재빌드 작업의 목표 이미지라고 합니다. 어떤 유형의 백업도 가능합니다(전체, 테이블 스페이스, 증분, 온라인 및 오프라인).

목표 이미지는 재빌드 단계에서 남아 있는 리스토어에 사용할 수 있는 이미지를 판별하는 로그 시퀀스(또는 로그 체인)을 설정합니다. 동일한 로그 체인에 있는 이미지만 사용할 수 있습니다.

다음 예는 재빌드 작업의 목표 이미지로 사용해야 하는 이미지를 선택하는 방법을 보여줍니다.

데이터베이스 SAMPLE이 있고 그 안에는 다음과 같은 테이블 스페이스가 있다고 가정하십시오.

- SYSCATSPACE(시스템 카탈로그)
- USERSP1(사용자 데이터 테이블 스페이스)
- USERSP2(사용자 데이터 테이블 스페이스)
- USERSP3(사용자 데이터 테이블 스페이스)

305 페이지의 그림 21은 취한 다음의 데이터베이스 레벨 백업과 테이블 스페이스 레벨 백업을 연대순으로 보여줍니다.

1. 전체 데이터베이스 백업 DB1
2. 전체 테이블 스페이스 백업 TS1
3. 전체 테이블 스페이스 백업 TS2
4. 전체 테이블 스페이스 백업 TS3
5. TS1 및 TS2 사이의 시점까지 데이터베이스 리스토어 및 롤 포워드
6. 전체 테이블 스페이스 백업 TS4
7. 전체 테이블 스페이스 백업 TS5

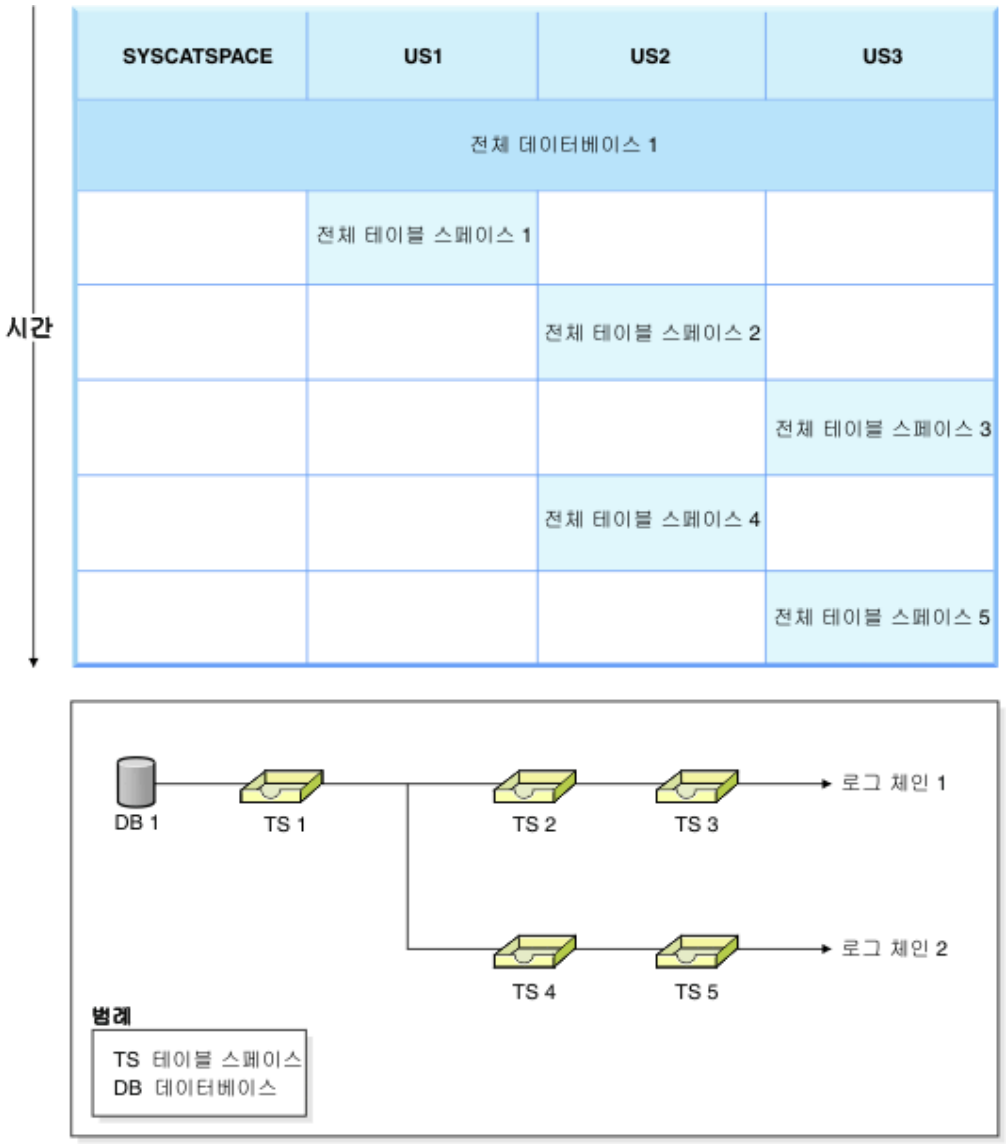


그림 21. 데이터베이스 SAMPLE의 데이터베이스 및 테이블 스페이스 레벨 백업

예 1

다음 예는 데이터베이스 SAMPLE을 현재 시점까지 재빌드하기 위해 발행해야 하는 CLP 명령을 보여줍니다. 먼저 재빌드할 테이블 스페이스를 선택해야 합니다. 목표는 현재 시점까지 데이터베이스를 재빌드하는 것이므로 목표 이미지로 최근의 백업 이미지를 선택해야 합니다. 최근의 백업 이미지는 로그 체인 2에 있는 이미지 TS5입니다.

```
db2 restore db sample rebuild with all tablespaces in database taken at
   TS5 without prompting
db2 rollforward db sample to end of logs
db2 rollforward db sample stop
```

이 명령은 백업 이미지 TS5, TS4, TS1 및 DB1을 자동으로 리스토어한 후 로그 체인 2의 끝까지 데이터베이스를 롤 포워드합니다.

주: 롤 포워드 작업의 완료를 위해 로그 체인 2에 속하는 모든 로그에 액세스할 수 있어야 합니다.

예 2

이 두 번째 예는 데이터베이스 SAMPLE을 로그 체인 1의 끝까지 재빌드하기 위해 발행해야 하는 CLP 명령을 보여줍니다. 선택하는 목표 이미지는 로그 체인 1에서 최근의 백업 이미지 TS3여야 합니다.

```
db2 restore db sample rebuild with all tablespaces in database
   taken at TS3 without prompting
db2 rollforward db sample to end of logs
db2 rollforward db sample stop
```

이 명령은 백업 이미지 TS3, TS2, TS1 및 DB1을 자동으로 리스토어한 후 로그 체인 1의 끝까지 데이터베이스를 롤 포워드합니다.

주: 롤 포워드 작업의 완료를 위해 로그 체인 1에 속하는 모든 로그에 액세스할 수 있어야 합니다.

재빌드에 대한 잘못된 목표 이미지 선택

데이터베이스 SAMPLE2이 있고 그 안에는 다음과 같은 테이블 스페이스가 있다고 가정하십시오.

- SYSCATSPACE(시스템 카탈로그)
- USERSP1(사용자 데이터 테이블 스페이스)
- USERSP2(사용자 데이터 테이블 스페이스)

307 페이지의 그림 22는 다음의 백업으로 구성되는 SAMPLE2의 백업 로그 체인을 보여줍니다.

1. BK1은 모든 테이블 스페이스를 포함하는 전체 데이터베이스 백업입니다.
2. BK2는 USERSP1의 전체 테이블 스페이스 백업입니다.
3. BK3는 USERSP2의 전체 테이블 스페이스 백업입니다.

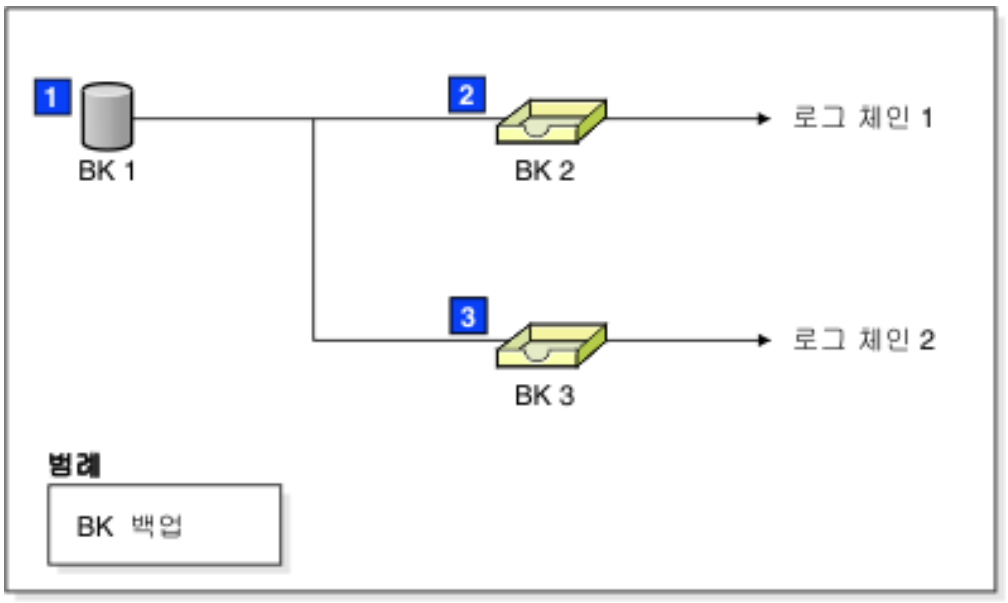


그림 22. 데이터베이스 SAMPLE2에 대한 백업 로그 체인

다음 예는 테이블 스페이스 SYSCATSPACE 및 USERSP2를 사용하여 BK3에서 데이터베이스를 재빌드하기 위해 발행해야 하는 CLP 명령을 보여줍니다.

```
db2 restore db sample2 rebuild with tablespace (SYSCATSPACE,
   USERSP2) taken at BK3 without prompting
```

이제, 이 리스토어가 완료된 후 USERSP1도 리스토어할 것을 결정한 경우 다음 명령을 발행합니다.

```
db2 restore db sample2 tablespace (USERSP1) taken at BK2
```

이 리스토어는 실패하여 BK2가 잘못된 로그 체인(SQL2154N)의 백업임을 알리는 메시지를 제공합니다. 그림 22에서 볼 수 있는 것처럼, USERSP1을 리스토어하기 위해 사용할 수 있는 유일한 이미지는 BK1입니다. 따라서 다음 명령을 입력해야 합니다.

```
db2 restore db sample2 tablespace (USERSP1) taken at BK1
```

이는 데이터베이스가 이에 따라 롤 포워드될 수 있도록 성공합니다.

선택된 테이블 스페이스 재빌드

데이터베이스를 재빌드하여 원래 데이터베이스를 구성하는 테이블 스페이스의 서브세트를 포함하는 데이터베이스를 빌드할 수 있습니다. 데이터베이스에 있는 테이블 스페이스의 서브세트만 재빌드하는 것이 다음 상황에서 유용할 수 있습니다.

- 테이블 스페이스의 서브세트에 대해서만 작업하려는 테스트 및 개발 환경에서.
- 더 중요한 테이블 스페이스를 다른 스페이스보다 빨리 온라인이 되도록 해야 하는 복구 상황에서는, 먼저 테이블 스페이스의 서브세트를 리스토어한 후 나중에 기타 테이블 스페이스를 리스토어할 수 있습니다.

원래 데이터베이스를 구성하는 테이블 스페이스의 서브세트를 포함하는 데이터베이스를 재빌드하려면 다음 예를 고려하십시오.

이 예에서 다음 테이블 스페이스가 있는 SAMPLE이라는 데이터베이스가 있습니다.

- SYSCATSPACE(시스템 카탈로그)
- USERSP1(사용자 데이터 테이블 스페이스)
- USERSP2(사용자 데이터 테이블 스페이스)
- USERSP3(사용자 데이터 테이블 스페이스)

다음 백업이 수행되었습니다.

- BK1은 SYSCATSPACE 및 USERSP1의 백업입니다.
- BK2는 USERSP2 및 USERSP3의 백업입니다.
- BK3는 USERSP3의 백업입니다.

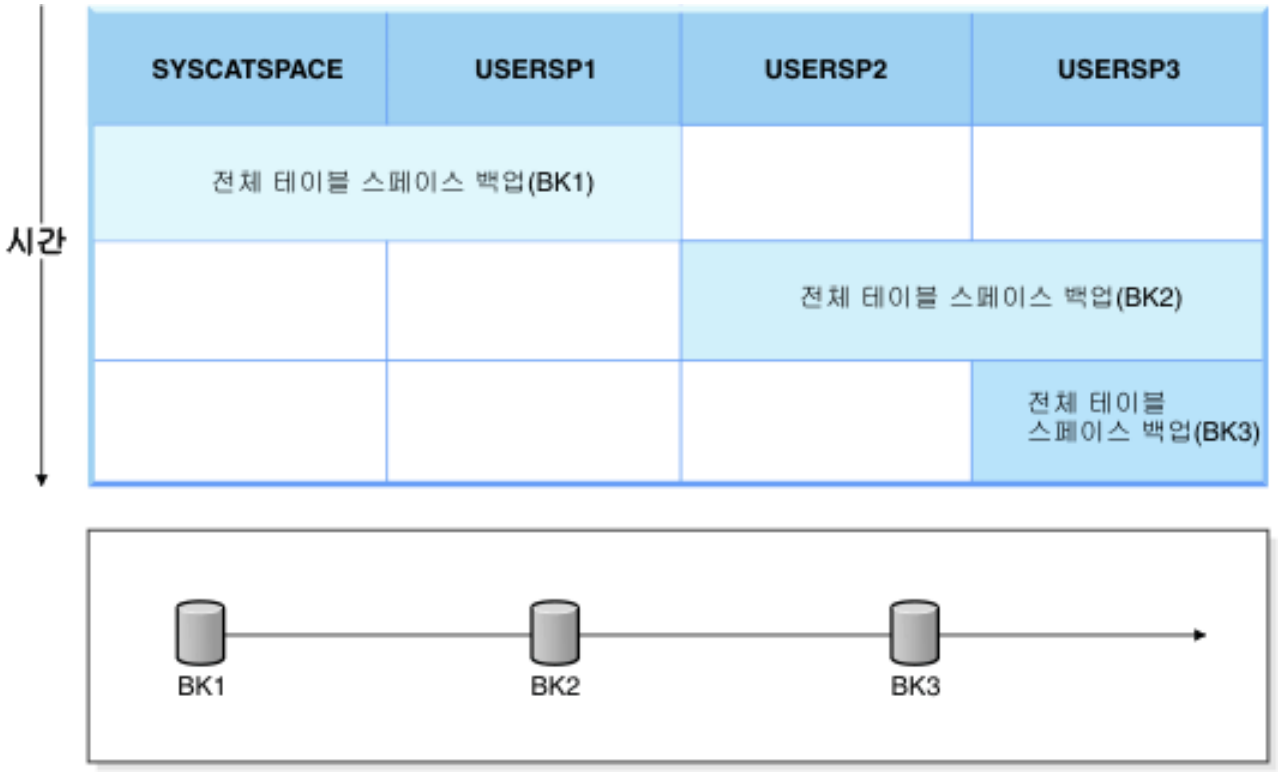


그림 23. SAMPLE 데이터베이스에 사용 가능한 백업 이미지

다음 프로시저는 CLP를 통해 발행되는 RESTORE DATABASE 및 ROLLFORWARD DATABASE 명령을 사용하여 SYSCATSPACE 및 USERSP1만 로그 끝까지 재빌드하는 것을 보여줍니다.

```
db2 restore db mydb rebuild with all tablespaces in image
        taken at BK1 without prompting
db2 rollforward db mydb to end of logs
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 SYSCATSPACE 및 USERSP1만 NORMAL 상태에 있습니다. USERSP2 및 USERSP3은 리스토어 보류 상태에 있습니다. 여전히 나중에 USERSP2 및 USERSP3을 리스토어할 수 있습니다.

재빌드 및 증분 백업 이미지

증분 이미지를 사용하여 데이터베이스를 재빌드할 수 있습니다. 디폴트로, 리스토어 유틸리티는 모든 증분 이미지에 대해 자동 증분 리스토어를 사용하려고 합니다. 즉, RESTORE DATABASE 명령의 INCREMENTAL 옵션을 사용하지 않지만 목표 이미지가 증분 백업 이미지인 경우 리스토어 유틸리티가 자동 증분 리스토어를 사용하여 재빌드 작업을 발행합니다. 목표 이미지가 증분 백업 이미지가 아니지만 다른 필수 이미지가 증분 이미지인 경우, 리스토어 유틸리티는 해당 증분 이미지가 자동 증분 리스토어를 사용하여 리스토어되는지 확인합니다. 리스토어 유틸리티는 AUTOMATIC 옵션을 사용하거나 사용하지 않고 INCREMENTAL 옵션을 지정하는지 여부에 관계없이 동일한 방법으로 작동합니다.

INCREMENTAL 옵션을 지정하지만 AUTOMATIC 옵션은 지정하지 않는 경우 전체 재빌드 프로세스를 수동으로 수행해야 합니다. 리스토어 유틸리티는 일반적인 수동 증분 리스토어에서와 같이 목표 이미지의 초기 메타데이터만 리스토어합니다. 그러면 필요한 증분 리스토어 체인을 사용하여 목표 이미지의 리스토어를 완료해야 합니다. 그런 다음 나머지 이미지를 리스토어하여 데이터베이스를 재빌드해야 합니다.

자동 증분 리스토어를 사용하여 데이터베이스를 재빌드할 것을 권장합니다. 리스토어가 실패하는 경우에만 수동 방법을 사용하여 데이터베이스 재빌드를 시도해야 합니다.

파티션된 데이터베이스 재빌드

파티션된 데이터베이스를 재빌드하려면 각 데이터베이스 파티션을 개별적으로 재빌드하십시오. 각 데이터베이스 파티션의 경우 카탈로그 파티션에서 시작하여 먼저 필요한 모든 테이블 스페이스를 리스토어하십시오. 리스토어되지 않는 모든 테이블 스페이스는 리스토어 보류 상태에 놓입니다. 모든 데이터베이스 파티션이 리스토어된 후, 카탈로그 파티션에서 ROLLFORWARD DATABASE 명령을 발행하여 모든 데이터베이스 파티션을 롤 포워드합니다.

주: 나중에 원래 재빌드 단계에 포함되지 않았던 모든 테이블 스페이스를 리스토어해야 하는 경우, 나중에 테이블 스페이스를 롤 포워드할 때 롤 포워드 유틸리티가 데이터베이스 파티션 사이의 모든 데이터를 동기화된 상태로 유지하는지 확인해야 합니다. 테이블 스페이스가 원래 리스토어 및 롤 포워드 조작 중에 누락되는 경우, 이는 데이터에

액세스하려는 시도가 있고 데이터 액세스 오류가 발생할 때까지 발견되지 않을 수 있습니다. 그런 다음 누락된 테이블 스페이스를 리스토어하고 롤 포워드하여 나머지 파티션과 다시 동기화되도록 해야 합니다.

테이블 스페이스 레벨 백업 이미지를 사용하여 파티션된 데이터베이스를 재빌드하려면 다음 예를 고려하십시오.

이 예에서 세 개의 데이터베이스 파티션이 있는 SAMPLE이라는 복구 가능한 데이터베이스가 있습니다.

- 데이터베이스 파티션 1은 테이블 스페이스 SYSCATSPACE, USERSP1 및 USERSP2를 포함하며 카탈로그 파티션입니다.
- 데이터베이스 파티션 2는 테이블 스페이스 USERSP1 및 USERSP3를 포함합니다.
- 데이터베이스 파티션 3은 테이블 스페이스 USERSP1, USERSP2 및 USERSP3를 포함합니다.

다음 백업이 작성되었으며, BKxy는 파티션 y의 백업 번호 x를 나타냅니다.

- BK11은 SYSCATSPACE, USERSP1 및 USERSP2의 백업입니다.
- BK12는 USERSP2 및 USERSP3의 백업입니다.
- BK13은 USERSP1, USERSP2 및 USERSP3의 백업입니다.
- BK21은 USERSP1의 백업입니다.
- BK22는 USERSP1의 백업입니다.
- BK23은 USERSP1의 백업입니다.
- BK31은 USERSP2의 백업입니다.
- BK33은 USERSP2의 백업입니다.
- BK42는 USERSP3의 백업입니다.
- BK43은 USERSP3의 백업입니다.

다음 프로시저는 CLP를 통해 발행되는 RESTORE DATABASE 및 ROLLFORWARD DATABASE 명령을 사용하여 로그 끝까지 전체 데이터베이스 재빌드를 보여줍니다.

1. 데이터베이스 파티션 1에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db sample rebuild with all tablespaces in database
      taken at BK31 without prompting
   ```

2. 데이터베이스 파티션 2에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db sample rebuild with tablespaces in database
      taken at BK42 without prompting
   ```

3. 데이터베이스 파티션 3에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db sample rebuild with all tablespaces in database
   taken at BK43 without prompting
```

4. 카탈로그 파티션에서, TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

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

5. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

   ```
   db2 rollforward db sample stop
   ```

이 때 데이터베이스는 모든 데이터베이스 파티션에서 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

데이터베이스 재빌드 제한사항

다음 목록은 데이터베이스 재빌드 제한사항을 요약한 것입니다.

- 재빌드하는 테이블 스페이스 중 하나는 카탈로그 파티션의 SYSCATSPACE여야 합니다.
- 제어 센터 GUI 도구로는 재빌드 작업을 수행할 수 없습니다. 명령행 처리기(CLP)를 사용하여 명령을 발행하거나 해당되는 API를 사용해야 합니다.
- REBUILD 옵션은 이미지가 오프라인 데이터베이스 백업의 이미지가 아닌 경우 버전 9.1 이전 목표 이미지에 대해 사용할 수 없습니다. 목표 이미지가 오프라인 데이터베이스 백업인 경우에만, 이 이미지의 테이블 스페이스를 재빌드에 사용할 수 있습니다. 데이터베이스는 재빌드 작업이 성공적으로 완료된 후에 이주해야 합니다. 다른 유형의 버전 9.1 이전 목표 이미지를 사용하여 재빌드하려고 하면 오류가 발생합니다.
- REBUILD 옵션은 목표 이미지가 전체 데이터베이스 백업이 아닌 경우 리스토어되는 것과 다른 운영 체제에서 목표 이미지에 대해 발행할 수 없습니다. 목표 이미지가 전체 데이터베이스 백업인 경우에만, 이 이미지의 테이블 스페이스를 재빌드에 사용할 수 있습니다. 다른 유형의 목표 이미지를 사용하여 리스토어되는 것과 다른 운영 체제에서 재빌드하려고 하면 오류가 발생합니다.

리스토어 성능 최적화

리스토어 작업을 수행할 때, DB2는 버퍼 수, 버퍼 *크기* 및 병렬 처리 설정에 최적의 값을 자동으로 선택합니다. 값은 사용 가능한 유틸리티 힙 메모리 양, 사용 가능한 프로세서 수 및 데이터베이스 구성을 기초로 합니다. 따라서 시스템에서 사용 가능한 스토리지 양에 따라 UTIL_HEAP_SZ 구성 매개변수를 늘려서 추가 메모리를 할당할 것을 고려해야 합니다. 목적은 리스토어 작업을 완료하는 데 소비되는 시간을 최소화하는 것입니다. 다음 RESTORE DATABASE 명령 매개변수의 값을 명시적으로 입력하지 않으면 DB2가 값을 선택합니다.

- WITH num-buffers BUFFERS

- PARALLELISM n
- BUFFER buffer-size

리스토어 작업의 경우 백업 작업에 사용되는 버퍼 *크기*의 배수가 항상 사용됩니다. RESTORE DATABASE 명령을 발행하지만 백업 버퍼 *크기*의 배수인지 확인해야 하는 경우에 버퍼 *크기*를 지정할 수 있습니다.

또한 리스토어 작업을 완료하는 데 필요한 시간을 줄이려면 다음 중 하나를 수행할 것을 선택할 수 있습니다.

- 리스토어 버퍼 *크기*를 늘리십시오.

  리스토어 버퍼 *크기*는 백업 작업 중에 지정된 백업 버퍼 *크기*의 양수의 배수입니다. 올바르지 않은 버퍼 *크기*를 지정하면 할당되는 버퍼는 승인할 수 있는 가장 작은 크기가 됩니다.
- 버퍼 수를 늘리십시오.

  지정하는 값은 백업 버퍼에 지정한 페이지 수의 배수여야 합니다. 최소 페이지 수는 8입니다.
- PARALLELISM 매개변수의 값을 늘리십시오.

  그러면 복원 작업 중에 데이터베이스에 *쓰기* 위해 사용할 BM(buffer manipulator) 수가 증가합니다.
- 유틸리티 힙 *크기*를 늘리십시오.

  그러면 다른 유틸리티가 동시에 사용할 수 있는 메모리가 증가합니다.

리스토어를 사용하는 데 필요한 특권, 권한 및 권한 부여

사용자는 특권을 사용하여 데이터베이스 자원을 작성하거나 자원에 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작과 특권을 그룹화하는 방법을 제공합니다. 이를 통해 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다. 사용자는 적절한 권한(즉, 필요한 특권 또는 권한)을 가지고 있는 오브젝트에만 액세스할 수 있습니다.

전체 데이터베이스 백업에서 *기존* 데이터베이스로 리스토어하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한을 가지고 있어야 합니다. *새* 데이터베이스로 리스토어하려면 SYSADM 또는 SYSCTRL 권한을 가지고 있어야 합니다.

리스토어 예

경로 재지정된 리스토어 세션 - CLP 예

예 1

다음은 별명이 MYDB인 데이터베이스에 대한 일반적인 비증분 경로 재지정된 리스토어 시나리오입니다.

1. REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

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

2. 컨테이너를 재정의해야 하는 테이블 스페이스마다 SET TABLESPACE CONTAINERS 명령을 발행하십시오. 예를 들어 Windows 환경에서는 다음과 같습니다.

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

   리스토어된 데이터베이스의 컨테이너가 이 단계에서 지정된 컨테이너인지 검증하려면, 컨테이너 위치가 재정의되는 모든 테이블 스페이스에 대해 LIST TABLESPACE CONTAINERS 명령을 발행하십시오.

3. 1 및 2 단계를 성공적으로 완료한 후 다음을 발행하십시오.

   ```
   db2 restore db mydb continue
   ```

   이는 경로 재지정된 리스토어 작업의 최종 단계입니다.

4. 3단계가 실패하거나 리스토어 조작이 중단된 경우, 경로 재지정된 리스토어를 1단계부터 재시작할 수 있습니다.

주:

1. 1단계가 성공적으로 완료되면, 3단계 완료 이전에 다음을 발행하여 리스토어 작업을 중단할 수 있습니다.

   ```
   db2 restore db mydb abort
   ```

2. 3단계가 실패하거나 리스토어 조작이 중단된 경우, 경로 재지정된 리스토어를 1단계부터 재시작할 수 있습니다.

예 2

다음은 별명이 MYDB인 데이터베이스에 대한 일반적인 수동 증분 경로 재지정된 리스토어 시나리오이며 다음 백업 이미지를 가지고 있습니다.

```
backup db mydb
백업이 완료되었습니다. 이 백업 이미지에 대한 시간소인은 <ts1>입니다.

backup db mydb incremental
백업이 완료되었습니다. 이 백업 이미지에 대한 시간소인은 <ts2>입니다.
```

1. INCREMENTAL 및 REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db mydb incremental taken at <ts2> replace existing redirect
   ```

2. 컨테이너를 재정의해야 하는 테이블 스페이스마다 SET TABLESPACE CONTAINERS 명령을 발행하십시오. 예를 들어 Windows 환경에서는 다음과 같습니다.

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

   리스토어된 데이터베이스의 컨테이너가 이 단계에서 지정된 컨테이너인지 검증하려면, TABLESPACE CONTAINERS 명령을 발행하십시오.

3. 1 및 2 단계를 성공적으로 완료한 후 다음을 발행하십시오.

```
db2 restore db mydb continue
```

4. 나머지 증분 리스토어 명령은 이제 다음과 같이 발행할 수 있습니다.

```
db2 restore db mydb incremental taken at <ts1>
db2 restore db mydb incremental taken at <ts2>
```

   이는 경로 재지정된 리스토어 작업의 최종 단계입니다.

주:

1. 1단계가 성공적으로 완료되면, 3단계 완료 이전에 다음을 발행하여 리스토어 작업을 중단할 수 있습니다.

```
db2 restore db mydb abort
```

2. 3단계가 성공적으로 완료되면, 4단계에서 필요한 모든 명령을 발행하기 전에 다음을 발행하여 리스토어 작업을 중단할 수 있습니다.

```
db2 restore db mydb incremental abort
```

3. 3단계가 실패하거나 리스토어 조작이 중단된 경우, 경로 재지정된 리스토어를 1단계부터 재시작할 수 있습니다.
4. 4단계에서 어느 한 리스토어 명령이 실패하면 실패하는 명령을 다시 발행하여 리스토어 프로세스를 계속할 수 있습니다.

예 3

다음은 동일한 데이터베이스에 대한 일반적인 자동 증분 경로 재지정된 리스토어 시나리오입니다.

1. INCREMENTAL AUTOMATIC 및 REDIRECT 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb incremental automatic taken at <ts2>
       replace existing redirect
```

2. 컨테이너를 재정의해야 하는 테이블 스페이스마다 SET TABLESPACE CONTAINERS 명령을 발행하십시오. 예를 들어 Windows 환경에서는 다음과 같습니다.

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

리스토어된 데이터베이스의 컨테이너가 이 단계에서 지정된 컨테이너인지 검증하려면, TABLESPACE CONTAINERS 명령을 발행하십시오.

3. 1 및 2 단계를 성공적으로 완료한 후 다음을 발행하십시오.

```
db2 restore db mydb continue
```

이는 경로 재지정된 리스토어 작업의 최종 단계입니다.

주:

1. 1단계가 성공적으로 완료되면, 3단계 완료 이전에 다음을 발행하여 리스토어 작업을 중단할 수 있습니다.

```
db2 restore db mydb abort
```

2. 3단계가 실패하거나 리스토어 작업이 중단된 경우, 경로 재지정된 리스토어를 발행 후 1단계부터 재시작할 수 있습니다.

```
db2 restore db mydb incremental abort
```

재빌드 세션 - CLP 예

시나리오 1

다음 예에서, 복구 가능한 데이터베이스 MYDB가 있고 그 안에는 다음과 같은 테이블 스페이스가 있습니다.

- SYSCATSPACE(시스템 카탈로그)
- USERSP1(사용자 데이터 테이블 스페이스)
- USERSP2(사용자 데이터 테이블 스페이스)
- USERSP3(사용자 데이터 테이블 스페이스)

다음 백업이 수행되었습니다.

- BK1은 SYSCATSPACE 및 USERSP1의 백업입니다.
- BK2는 USERSP2 및 USERSP3의 백업입니다.
- BK3는 USERSP3의 백업입니다.

예 1

다음은 전체 데이터베이스를 가장 최근 특정 시점까지 재빌드합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in database
   taken at BK3 without prompting
```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

예 2

다음은 단지 SYSCATSPACE 및 USERSP2를 특정 시점까지 재빌드합니다(BK3 끝이 특정 시점보다 최근이 아니고, 특정 시점은 로그 끝보다 최근이 아님).

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하고 포함하려는 테이블 스페이스를 지정하십시오.

   ```
   db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
      taken at BK2 without prompting
   ```

2. TO PIT OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

   ```
   db2 rollforward db mydb to PIT
   ```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

   ```
   db2 rollforward db mydb stop
   ```

이 때 데이터베이스는 연결 가능하고 SYSCATSPACE 및 USERSP2만 NORMAL 상태에 있습니다. USERSP1 및 USERSP3는 RESTORE_PENDING 상태에 있습니다.

나중에 USERSP1 및 USERSP3을 리스토어하려면 정상 테이블 스페이스를 사용하여 (REBUILD 옵션 없이) 다음을 수행하십시오.

1. REBUILD 옵션 *없이* RESTORE DATABASE 명령을 발행하고 리스토어하려는 테이블 스페이스를 지정하십시오. 먼저 USERSPI을 리스토어하십시오.

   ```
   db2 restore db mydb tablespace (USERSP1) taken at BK1 without prompting
   ```

2. 그런 다음 USERSP3를 리스토어하십시오.

   ```
   db2 restore db mydb tablespace taken at BK3 without prompting
   ```

3. END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하고 리스토어할 테이블 스페이스를 지정하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

   ```
   db2 rollforward db mydb to end of logs tablespace (USERSP1, USERSP3)
   ```

   롤 포워드는 모든 로그를 PIT까지 재생한 후 이 두 테이블 스페이스에 대해 중지합니다. 첫 번째 롤 포워드 이후에 작업이 수행되지 않았기 때문입니다.

4. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

   ```
   db2 rollforward db mydb stop
   ```

예 3

다음은 SYSCATSPACE 및 USERSP1만 로그 끝까지 재빌드합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db mydb rebuild with all tablespaces in image
      taken at BK1 without prompting
   ```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

   ```
   db2 rollforward db mydb to end of logs
   ```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

   ```
   db2 rollforward db mydb stop
   ```

이 때 데이터베이스는 연결 가능하고 SYSCATSPACE 및 USERSP1만 NORMAL 상태에 있습니다. USERSP2 및 USERSP3는 RESTORE_PENDING 상태에 있습니다.

예 4

다음 예에서, 백업 BK1 및 BK2는 더 이상 실행기록 파일에 언급된 것처럼 동일한 위치에 있지 않지만 재빌드가 발행될 때 이 사실이 알려지지 않습니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하고 최근 특정 시점까지 전체 데이터베이스를 재빌드할 것을 지정하십시오.

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK3 without prompting
   ```

   이 때, 목표 이미지는 성공적으로 리스토어되지만 필요한 이미지를 찾을 수 없음을 알리는 오류가 리스토어 유틸리티에서 리턴됩니다.

2. 이제 수동으로 재빌드를 완료해야 합니다. 데이터베이스는 재빌드 단계에 있으므로 다음과 같이 수행될 수 있습니다.

   a. RESTORE DATABASE 명령을 발행하고 BK1 백업 이미지의 위치를 지정하십시오.

      ```
      db2 restore db mydb tablespace taken at BK1 from <location>
         without prompting
      ```

   b. RESTORE DATABASE 명령을 발행하고 BK2 백업 이미지의 위치를 지정하십시오.

      ```
      db2 restore db mydb tablespace (USERSP2) taken at BK2 from
         <location> without prompting
      ```

   c. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

      ```
      db2 rollforward db mydb to end of logs
      ```

   d. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

      ```
      db2 rollforward db mydb stop
      ```

이 때 데이터베이스는 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

예 5

이 때, 테이블 스페이스 USERSP3에는 특정 보고서를 생성하는 데 필요한 독립 데이터가 있지만 사용자는 원래 데이터베이스와 충돌할 보고서 생성을 원하지 않습니다. 데이터에 대한 액세스를 얻지만 원래 데이터베이스에는 영향을 주지 않으려면, REBUILD를 사용하여 이 테이블 스페이스와 SYSCATSPACE로 새 데이터베이스를 생성할 수 있습니다. SYSCATSPACE는 리스토어 및 롤 포워드 작업 후에 데이터베이스가 연결 가능하도록 하기 위해서도 필요합니다.

SYSCATSPACE 및 USERSP3에서 가장 최근의 데이터로 새 데이터베이스를 빌드하려면 다음을 수행하십시오.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하고 SYSCATSPACE 및 USERSP3를 새 데이터베이스 NEWDB로 리스토어할 것을 지정하십시오.

   ```
   db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP3)
      taken at BK3 into newdb without prompting
   ```

2. TO END OF LOGS 옵션과 함께 NEWDB에서 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

   ```
   db2 rollforward db newdb to end of logs
   ```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

   ```
   db2 rollforward db newdb stop
   ```

이 때 새 데이터베이스가 연결 가능하고 SYSCATSPACE 및 USERSP3만 NORMAL 상태에 있습니다. USERSP1 및 USERSP2는 RESTORE_PENDING 상태에 있습니다.

주: 컨테이너 경로가 현재 데이터베이스와 세 데이터베이스 사이에 문제가 되는 경우(예를 들어, 파일 시스템이 존재하지 않아서 원래 데이터베이스의 컨테이너를 변경해야 하는 경우, 또는 컨테이너가 이미 원래 데이터베이스에서 사용 중인 경우) 경로 재지정 리스토어를 수행해야 합니다. 위의 예에서는 디폴트 자동 스토리지 데이터베이스 경로가 테이블 스페이스에 사용됩니다.

시나리오 2

다음 예에서는 SYSCATSPACE와 1000개의 사용자 테이블 스페이스 Txxxx를 가지고 있는 복구 가능한 데이터베이스 MYDB가 있습니다. 여기서 x는 테이블 스페이스 번호를 나타냅니다(예: T0001). 하나의 데이터베이스 백업 이미지(BK1)가 있습니다.

예 6

다음은 T0999 및 T1000을 제외하고 모든 테이블 스페이스를 리스토어합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in image except
   tablespace (T0999, T1000) taken at BK1 without prompting
```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 T0999 및 T1000을 제외한 모든 테이블 스페이스는 NORMAL 상태에 있습니다. T0999 및 T1000은 RESTORE_PENDING 상태에 있습니다.

시나리오 3

이 시나리오의 예제는 증분 백업을 사용하여 복구 가능한 데이터베이스를 재빌드하는 방법을 보여줍니다. 다음 예에서, 데이터베이스 MYDB가 있고 그 안에는 다음과 같은 테이블 스페이스가 있습니다.

- SYSCATSPACE(시스템 카탈로그)
- USERSP1(데이터 테이블 스페이스)
- USERSP2(사용자 데이터 테이블 스페이스)
- USERSP3(사용자 데이터 테이블 스페이스)

다음 백업이 수행되었습니다.

- FULL1은 SYSCATSPACE, USERSP1, USERSP2 및 USERSP3의 전체 백업입니다.
- DELTA1은 SYSCATSPACE 및 USERSP1의 델타 백업입니다.
- INCR1은 USERSP2 및 USERSP3의 증분 백업입니다.
- DELTA2는 SYSCATSPACE, USERSP1, USERSP2 및 USERSP3의 델타 백업입니다.
- DELTA3은 USERSP2의 델타 백업입니다.
- FULL2는 USERSP1의 전체 백업입니다.

예 7

다음은 증분 자동 리스토어를 사용하여 SYSCATSPACE 및 USERSP2를 가장 최근의 특정 시점까지 재빌드합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.
   INCREMENTAL AUTO 옵션은 선택적입니다. 리스토어 유틸리티는 이미지의 세분화도(granularity)를 발견하고 필요에 따라 자동 증분 리스토어를 사용합니다.

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
   incremental auto taken at DELTA3 without prompting
```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 SYSCATSPACE 및 USERSP2만 NORMAL 상태에 있습니다. USERSP1 및 USERSP3는 RESTORE_PENDING 상태에 있습니다.

예 8

다음은 증분 자동 리스토어를 사용하여 전체 데이터베이스를 가장 최근의 특정 시점까지 재빌드합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오. INCREMENTAL AUTO 옵션은 선택적입니다. 리스토어 유틸리티는 이미지의 세분화도(granularity)를 발견하고 필요에 따라 자동 증분 리스토어를 사용합니다.

```
db2 restore db mydb rebuild with all tablespaces in database
   incremental auto taken at DELTA3 without prompting
```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

예 9

다음은 USERSP3를 제외하고 전체 데이터베이스를 가장 최근 특정 시점까지 재빌드합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오. 목표 이미지가 비증분 이미지여도, 리스토어 유틸리티는 필요한 재빌드 체인에 증분 이미지가 포함되어 있음을 발견하므로 자동으로 해당 이미지를 증분식으로 리스토어합니다.

```
db2 restore db mydb rebuild with all tablespaces in database except
   tablespace (USERSP3) taken at FULL2 without prompting
```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

시나리오 4

이 시나리오의 예는 로그 파일을 포함하는 백업 이미지를 사용하여 복구 가능한 데이터베이스를 재빌드하는 방법을 보여줍니다. 다음 예에서, 데이터베이스 MYDB가 있고 그 안에는 다음과 같은 테이블 스페이스가 있습니다.

- SYSCATSPACE(시스템 카탈로그)
- USERSP1(사용자 데이터 테이블 스페이스)
- USERSP2(사용자 데이터 테이블 스페이스)

예 10

다음은 SYSCATSPACE 및 USERSP2가 있는 데이터베이스를 가장 최근의 특정 시점까지 재빌드합니다. 로그 파일을 포함하는 전체 온라인 데이터베이스 백업 이미지(BK1)가 있습니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
   taken at BK1 logtarget /logs without prompting
```

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 BK1 이후의 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs overflow log path (/logs)
```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 SYSCATSPACE 및 USERSP2만 NORMAL 상태에 있습니다. USERSP1은 RESTORE_PENDING 상태에 있습니다.

예 11

다음은 데이터베이스를 가장 최근 특정 시점까지 재빌드합니다. 로그 파일을 포함하는 두 개의 전체 온라인 테이블 스페이스 백업 이미지가 있습니다.

- BK1은 SYSCATSPACE의 백업이며 로그 파일 10 - 45를 사용합니다.
- BK2는 USERSP1 및 USERSP2의 백업이며 로그 파일 64 - 80을 사용합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in database
   taken at BK2 logtarget /logs without prompting
```

롤 포워드 작업은 로그 파일 10에서 시작하며, 1차 로그 파일 경로에 없으면 오버플로우 로그 경로에서 항상 찾습니다. 로그 범위 46 - 63은 어떤 백업 이미지에도 포함되지 않으므로 롤 포워드에 사용 가능하도록 만들어야 합니다.

2. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오. 로그 파일 64 - 80에 대해 오버플로우 로그 경로를 사용합니다.

   ```
   db2 rollforward db mydb to end of logs overflow log path (/logs)
   ```

3. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

   ```
   db2 rollforward db mydb stop
   ```

이 때 데이터베이스는 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

시나리오 5

다음 예에서, 복구 가능한 데이터베이스 MYDB가 있고 그 안에는 다음과 같은 테이블 스페이스가 있습니다.

- SYSCATSPACE(0), SMS 시스템 카탈로그(상대적 컨테이너)
- USERSP1(1) SMS 사용자 데이터 테이블 스페이스(상대적 컨테이너)
- USERSP2(2) DMS 사용자 데이터 테이블 스페이스(절대 컨테이너 /usersp2)
- USERSP3(3) DMS 사용자 데이터 테이블 스페이스(절대 컨테이너 /usersp3)

다음 백업이 수행되었습니다.

- BK1은 SYSCATSPACE 및 USERSP1의 백업입니다.
- BK2는 USERSP2 및 USERSP3의 백업입니다.
- BK3는 USERSP3의 백업입니다.

예 12

다음은 경로 재지정된 리스토어를 사용하여 전체 데이터베이스를 가장 최근의 특정 시점까지 재빌드합니다.

1. REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK3 redirect without prompting
   ```

2. 컨테이너를 재정의해야 하는 테이블 스페이스마다 SET TABLESPACE CONTAINERS 명령을 발행하십시오. 예를 들어, 다음과 같습니다.

   ```
   db2 set tablespace containers for 3 using (file '/newusersp2' 10000)
   ```

3. `db2 set tablespace containers for 4 using (file '/newusersp3' 15000)`
4. CONTINUE 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

   ```
   db2 restore db mydb continue
   ```

5. TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

6. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

시나리오 6

다음 예에서는 세 개의 데이터베이스 파티션이 있는 데이터베이스 MYDB가 있습니다.

- 데이터베이스 파티션 1은 테이블 스페이스 SYSCATSPACE, USERSP1 및 USERSP2를 포함하며 카탈로그 파티션입니다.
- 데이터베이스 파티션 2는 테이블 스페이스 USERSP1 및 USERSP3를 포함합니다.
- 데이터베이스 파티션 3은 테이블 스페이스 USERSP1, USERSP2 및 USERSP3를 포함합니다.

다음 백업이 작성되었으며, BKxy는 파티션 y의 백업 번호 x를 나타냅니다.

- BK11은 SYSCATSPACE, USERSP1 및 USERSP2의 백업입니다.
- BK12는 USERSP2 및 USERSP3의 백업입니다.
- BK13은 USERSP1, USERSP2 및 USERSP3의 백업입니다.
- BK21은 USERSP1의 백업입니다.
- BK22는 USERSP1의 백업입니다.
- BK23은 USERSP1의 백업입니다.
- BK31은 USERSP2의 백업입니다.
- BK33은 USERSP2의 백업입니다.
- BK42는 USERSP3의 백업입니다.
- BK43은 USERSP3의 백업입니다.

예 13

다음은 전체 데이터베이스를 로그 끝까지 재빌드합니다.

1. 데이터베이스 파티션 1에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in database
   taken at BK31 without prompting
```

2. 데이터베이스 파티션 2에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with tablespaces in database taken at
   BK42 without prompting
```

3. 데이터베이스 파티션 3에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in database
   taken at BK43 without prompting
```

4. 카탈로그 파티션에서, TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오(이 때 모든 로그가 저장되었고 데이터베이스 파티션에서 액세스 가능한 것으로 가정함).

```
db2 rollforward db mydb to end of logs
```

5. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

이 때 데이터베이스는 모든 데이터베이스 파티션에서 연결 가능하고 모든 테이블 스페이스는 NORMAL 상태에 있습니다.

예 14

다음은 SYSCATSPACE, USERSP1 및 USERSP2를 가장 최근의 특정 시점까지 재빌드합니다.

1. 데이터베이스 파티션 1에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in database
   taken at BK31 without prompting
```

2. 데이터베이스 파티션 2에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in image taken at
   BK22 without prompting
```

3. 데이터베이스 파티션 3에서, REBUILD 옵션과 함께 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb rebuild with all tablespaces in image taken at
   BK33 without prompting
```

   메모: 이 명령은 재빌드 작업을 완료하는 데 필요한 USERSP1을 생략했습니다.

4. 카탈로그 파티션에서, TO END OF LOGS 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb to end of logs
```

5. STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb stop
```

롤 포워드가 성공하고 데이터베이스는 모든 데이터베이스 파티션에서 연결 가능합니다. USERSP3 및 USERSP1을 제외한 모든 테이블 스페이스는 NORMAL 상태입니다. USERSP3는 존재하는 모든 데이터베이스 파티션에서 RESTORE PENDING 상태에 있고, USERSP1은 데이터베이스 파티션 3에서 RESTORE PENDING 상태에 있습니다.

데이터베이스 파티션 3의 USERSP1에 있는 데이터에 액세스하려고 할 때 데이터 액세스 오류가 발생합니다. 이 오류를 수정하려면 USERSP1을 복구해야 합니다.

a. 데이터베이스 파티션 3에서, USERSP1을 포함하는 백업 이미지를 지정하여 RESTORE DATABASE 명령을 발행하십시오.

```
db2 restore db mydb tablespace taken at BK23 without prompting
```

b. 카탈로그 파티션에서, TO END OF LOGS 옵션 및 AND STOP 옵션과 함께 ROLLFORWARD DATABASE 명령을 발행하십시오.

```
db2 rollforward db mydb to end of logs on dbpartitionnum (3) and stop
```

이 때 데이터베이스 파티션 3의 USERSP1은 NORMAL 상태에 있으므로 액세스한 데이터를 가지고 있을 수 있습니다.

시나리오 7

다음 예에서는 *복구할 수 없는* 데이터베이스 MYDB가 있고 그 안에는 다음과 같은 테이블 스페이스가 있습니다.

- SYSCATSPACE(0), SMS 시스템 카탈로그
- USERSP1(1) SMS 사용자 데이터 테이블 스페이스
- USERSP2(2) DMS 사용자 데이터 테이블 스페이스
- USERSP3(3) DMS 사용자 데이터 테이블 스페이스

데이터베이스의 백업이 하나만 있습니다(BK1).

예 15

다음은 복구할 수 없는 데이터베이스에 대한 재빌드 사용 방법을 보여줍니다.

SYSCATSPACE 및 USERSP1만 사용하여 데이터베이스를 재빌드하십시오.

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP1)
   taken at BK1 without prompting
```

리스토어 후에 데이터베이스는 연결 가능하게 됩니다. LIST TABLESPACES 명령을 발행하는 경우 USERSP2 및 USERSP3는 DELETE_PENDING/OFFLINE 상태에 있지만 SYSCATSPACE 및 USERSP1은 NORMAL 상태에 있음을 알 수 있습니다. 이제는 NORMAL 상태에 있는 두 개의 테이블 스페이스에 대해 작업할 수 있습니다.

데이터베이스 백업을 수행하려면 먼저 DROP TABLESPACE 명령을 사용하여 USERSP2 및 USERSP3를 삭제해야 합니다. 그렇지 않으면 백업이 실패합니다.

나중에 USERSP2 및 USERSP3를 리스토어하려면 BK1에서 데이터베이스 리스토어를 다시 발행해야 합니다.

제 14 장 롤 포워드 개요

DB2 ROLLFORWARD DATABASE 명령의 가장 단순한 양식에서는 롤 포워드 복구하려는 데이터베이스의 별명 이름만 지정하면 됩니다. 예를 들어, 다음과 같습니다.

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

이 예에서 명령은 다음을 리턴합니다.

```
                                 롤 포워드 상태

 입력 데이터베이스 별명                = sample
 상태를 리턴한 노드 수                 = 1

 노드 번호                             = 0
 롤 포워드 상태                        = not pending
 다음에 읽을 로그 파일                 =
 처리된 로그 파일                      =  -
 최종 커미트된 트랜잭션                = 2001-03-11-02.39.48.000000

DB20000I  ROLLFORWARD 명령이 완료되었습니다.
```

다음은 롤 포워드 복구를 수행하기 위해 사용할 수 있는 하나의 접근방식입니다.

1. STOP 옵션 없이 롤 포워드 유틸리티를 호출합니다.
2. QUERY STATUS 옵션을 사용하여 롤 포워드 유틸리티를 호출합니다.

   로그 끝까지의 복구를 지정하는 경우, QUERY STATUS 옵션은 리턴된 특정 시점이 예상한 것보다 이전인 경우 하나 이상의 로그 파일이 누락되었음을 표시할 수 있습니다.

   특정 시점 복구를 지정하는 경우, QUERY STATUS 옵션은 롤 포워드 작업이 올바른 시점에서 완료되었는지 확인하는 데 도움이 됩니다.
3. STOP 옵션을 사용하여 롤 포워드 유틸리티를 호출합니다. 작업이 중지되면 추가 변경사항을 롤 포워드할 수 없습니다.

롤 포워드 복구를 수행하기 위해 사용할 수 있는 대체 접근방식은 다음과 같습니다.

1. AND STOP 옵션을 사용하여 롤 포워드 유틸리티를 호출합니다.
2. 추가 단계 수행 필요성은 롤 포워드 작업의 결과에 따라 결정됩니다.
   - 성공하면 롤 포워드가 완료되고 데이터베이스는 연결 가능하며 사용 가능하게 됩니다. 이때 추가 변경사항을 롤 포워드할 수 없습니다.

- 오류가 리턴되면 문제점을 수정하는 데 필요한 조치를 수행하십시오(예를 들어, 로그 파일이 누락된 경우 로그 파일을 찾거나, 검색 오류가 있는 경우 로그 아카이브가 작동 중인지 확인). 그런 다음 AND STOP 옵션을 사용하여 롤 포워드 유틸리티를 다시 발행하십시오.

데이터베이스를 롤 포워드할 수 있으려면 먼저 데이터베이스를 성공적으로 리스토어(리스토어 유틸리티를 사용하여)해야 하지만, 테이블 스페이스는 그렇지 않습니다. 테이블 스페이스는 임시로 롤 포워드 보류 상태에 놓을 수 있지만, 실행 취소하기 위해(예를 들어, 전원 인터럽트 다음에) 리스토어 작업이 필요하지 않습니다.

롤 포워드 유틸리티가 호출될 때:

- 데이터베이스가 롤 포워드 보류 상태인 경우 데이터베이스는 롤 포워드됩니다. 테이블 스페이스도 롤 포워드 보류 상태에 있으면 데이터베이스 롤 포워드 작업이 완료된 후 다시 롤 포워드 유틸리티를 호출하여 테이블 스페이스를 롤 포워드해야 합니다.
- 데이터베이스가 롤 포워드 보류 상태가 *아니지만* 데이터베이스의 테이블 스페이스가 롤 포워드 보류 상태에 *있는* 경우:
  - 테이블 스페이스 목록을 지정하면 해당 테이블 스페이스만 롤 포워드됩니다.
  - 테이블 스페이스 목록을 지정하지 않으면 롤 포워드 보류 상태에 있는 모든 테이블 스페이스가 롤 포워드됩니다.

데이터베이스 롤 포워드 작업은 오프라인에서 실행됩니다. 데이터베이스는 롤 포워드 작업이 성공적으로 완료될 때까지 사용 가능하지 않고 작업은 유틸리티가 호출될 때 STOP 옵션이 지정되지 않은 경우 완료할 수 없습니다.

테이블 스페이스 롤 포워드 작업은 오프라인에서 실행됩니다. 데이터베이스는 롤 포워드 작업이 성공적으로 완료될 때까지 사용 가능하지 않습니다. 이는 로그 끝에 도달하거나 유틸리티가 호출될 때 STOP 옵션이 지정된 경우에 발생합니다.

SYSCATSPACE가 포함되지 않으면 테이블 스페이스에 대한 *온라인* 롤 포워드 작업을 수행할 수 있습니다. 테이블 스페이스에 대해 온라인 롤 포워드 작업을 수행하는 경우 테이블 스페이스는 사용할 수 없지만 데이터베이스의 다른 테이블 스페이스는 사용 *가능합니다*.

먼저 데이터베이스를 작성하는 경우 그 데이터베이스는 순환 로깅에 대해서만 사용 가능합니다. 이는 로그가 저장되거나 아카이브되기 보다는 재사용됨을 의미합니다. 순환 로깅을 사용하는 경우, 롤 포워드 복구는 불가능합니다. 응급 복구나 버전 복구만 수행할 수 있습니다. 아카이브된 로그는 백업 수행 후 발생하는 데이터베이스 변경사항을 문서화합니다. *logarchmeth1* 데이터베이스 구성 매개변수를 디폴트 OFF가 아닌 다른 값으로 설정하여 로그 아카이브(및 롤 포워드 복구)를 사용 가능하도록 설정할 수 있습

니다. *logarchmeth1*을 OFF가 아닌 다른 값으로 설정하는 경우 데이터베이스는 백업 보류 상태에 놓이므로 다시 사용하려면 데이터베이스의 오프라인 백업을 수행해야 합니다.

주: 롤 포워드 작업에 사용되는 로그 파일마다 복구 실행기록 파일에서 항목이 작성됩니다.

롤 포워드 사용

데이터베이스 로그 파일에 기록된 트랜잭션을 리스토어된 데이터베이스 백업 이미지나 테이블 스페이스 백업 이미지에 적용하려면 ROLLFORWARD DATABASE 명령을 사용하십시오.

롤 포워드 복구될 데이터베이스에 연결되지 않았어야 합니다. 롤 포워드 유틸리티가 자동으로 지정된 데이터베이스에 연결하고 이 연결은 롤 포워드 작업이 완료될 때 종료됩니다.

진행 중인 롤 포워드 작업을 취소하지 않고 테이블 스페이스를 리스토어하지 마십시오. 그렇지 않으면 일부 테이블 스페이스는 롤 포워드 진행 중 상태에 있고 일부 테이블 스페이스는 롤 포워드 보류 상태에 있는 테이블 스페이스 세트를 가질 수 있습니다. 진행 중인 롤 포워드 작업은 롤 포워드 진행 중 상태에 있는 테이블 스페이스에만 작용합니다.

데이터베이스는 로컬 또는 리모트일 수 있습니다.

다음 제한사항이 롤 포워드 유틸리티에 적용됩니다.

- 한 번에 하나의 롤 포워드 작업만 호출할 수 있습니다. 복구할 테이블 스페이스가 많은 경우 동일한 조작에서 모두 지정할 수 있습니다.
- 가장 최근의 백업 작업 뒤에 테이블 스페이스 이름을 바꾼 경우 테이블 스페이스를 롤 포워드할 때 새 이름을 사용해야 합니다. 이전 테이블 스페이스 이름은 인식되지 않습니다.
- 실행 중인 롤 포워드 조작을 취소할 수 없습니다. 완료되었지만 STOP 옵션이 지정되지 않은 롤 포워드 작업이나 완료되기 전에 실패한 롤 포워드 작업만 취소할 수 있습니다.
- 테이블 스페이스 롤 포워드 작업을 이전 시간소인보다 작은 시간소인을 지정하여 특정 시점까지 *계속*할 수 없습니다. 특정 시점이 지정되지 않으면 이전 시점이 사용됩니다. STOP만 지정하여 지정된 특정 시점에서 종료하는 롤 포워드 작업을 실행할 수 있지만, 이것은 관련된 테이블 스페이스가 모두 동일한 오프라인 백업 이미지로부터 리스토어된 경우에만 허용됩니다. 이 경우에 로그 처리가 필요하지 않습니다. 진행 중인 롤 포워드 작업이 완료되거나 취소되기 전에 다른 테이블 스페이스 목록이 있는 다른 롤 포워드 작업을 시작하는 경우 오류 메시지(SQL4908)가 리턴됩니다.

모든 데이터베이스 파티션에서 LIST TABLESPACES 명령을 호출하여 현재 롤 포워드되고 있는 테이블 스페이스(롤 포워드 진행 중 상태) 및 롤 포워드될 준비가 된 테이블 스페이스(롤 포워드 보류 상태)를 판별하십시오. 다음 세 가지 옵션이 있습니다.

  - 모든 테이블 스페이스의 진행 중 롤 포워드 작업을 완료하십시오.
  - 테이블 스페이스의 서브세트에 있는 진행 중 롤 포워드 작업을 완료하십시오. (롤 포워드 작업이 특정 시점까지 계속할 경우(모든 데이터베이스 파티션의 참여가 필요), 이것이 불가능할 수 있습니다.)
  - 진행 중 롤 포워드 작업을 취소하십시오.

- 파티션된 데이터베이스 환경에서 롤 포워드 유틸리티는 데이터베이스의 카탈로그 파티션에서 호출되어야 합니다.
- 테이블 스페이스의 특정 시점 롤 포워드는 DB2 버전 9 클라이언트에서만 사용 가능합니다. 테이블 스페이스를 특정 시점으로 롤 포워드하려면 데이터베이스 제품의 이전 버전을 실행하는 모든 클라이언트를 버전 9로 업그레이드해야 합니다.
- 이전 릴리스 버전의 로그를 롤 포워드할 수 없습니다.

롤 포워드 유틸리티는 명령행 처리기(CLP), 제어 센터의 리스토어 마법사 또는 db2Rollforward API를 통해 호출할 수 있습니다.

다음은 CLP를 통해 실행되는 ROLLFORWARD DATABASE 명령의 예입니다.

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

리스토어 마법사를 열려면 다음을 수행하십시오.

1. 제어 센터에서 리스토어하려는 데이터베이스나 테이블 스페이스 오브젝트를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. 오브젝트를 마우스 오른쪽 단추로 누르고 팝업 메뉴에서 롤 포워드를 선택하십시오. 롤 포워드 마법사가 열립니다.

제어 센터에 있는 문맥 도움말 기능을 통해 세부사항 정보가 제공됩니다.

테이블 스페이스에서 변경사항 롤 포워드

데이터베이스가 포워드 복구에 사용 가능하면 전체 데이터베이스 대신 테이블 스페이스를 백업, 리스토어 및 롤 포워드하는 옵션이 제공됩니다. 개별적 테이블 스페이스에 대해 복구 전략을 구현할 것을 원할 수 있습니다. 시간이 절약되기 때문입니다. 전체 데이터베이스를 복구하는 것보다 데이터베이스의 부분을 복구하는 데 걸리는 시간이 적습니다. 예를 들어, 디스크가 불량이고 하나의 테이블 스페이스만 포함하는 경우, 손상된 테이블 스페이스가 시스템 카탈로그 테이블을 포함하지 않는 한(이 상황에서는 데이터베이스에 연결할 수 없음) 전체 데이터베이스를 복구하지 않고, 데이터베이스의 나머지에 대한 사용자 액세스에 영향을 주지 않으면서 해당 테이블 스페이스를 리스토어 및 롤 포워드할 수 있습니다. (시스템 카탈로그 테이블 스페이스를 포함하는 테이블 스페

이스 레벨 백업 이미지를 사용할 수 있으면 시스템 카탈로그 테이블 스페이스를 독립적으로 리스토어할 수 있습니다.) 테이블 스페이스 레벨 백업은 또한 사용자가 다른 파트보다 더 자주 데이터베이스의 중요 파트를 백업할 수 있도록 하므로, 전체 데이터베이스를 백업하는 것보다 필요한 시간이 적습니다.

테이블 스페이스가 리스토어되면 항상 롤 포워드 보류 상태가 됩니다. 테이블 스페이스를 사용 가능하도록 만들려면 롤 포워드 복구를 수행해야 합니다. 대부분의 경우, 로그의 끝까지 롤 포워드하거나 특정 시점까지 롤 포워드하는 옵션이 제공됩니다. 그러나 시스템 카탈로그 테이블을 포함하는 테이블 스페이스는 특정 시점까지 롤 포워드할 수 없습니다. 이러한 테이블 스페이스는 로그 끝까지 롤 포워드하여 데이터베이스의 모든 테이블 스페이스가 일관성 있게 유지되도록 해야 합니다.

테이블 스페이스가 롤 포워드되는 경우, DB2는 해당 테이블 스페이스에 영향을 주는 로그 레코드를 포함하지 않는 경우에도 모든 로그 파일을 처리합니다. 테이블 스페이스에 영향을 주는 로그 레코드를 포함하지 않는 것으로 알려진 로그 파일을 건너뛰려면 DB2_COLLECT_TS_REC_INFO 레지스트리 변수를 *ON*으로 설정하십시오. 이는 디폴트값입니다. 로그 파일을 건너뛰기 위해 필요한 정보를 수집하려면 로그 파일을 작성하고 사용하기 전에 레지스트리 변수를 설정해야 합니다.

데이터베이스 디렉토리에 있는 테이블 스페이스 변경 실행기록 파일(DB2TSCHG.HIS)은 각 테이블 스페이스에 대해 로그가 처리되어야 하는 트랙을 유지합니다. db2logsForRfwd 유틸리티를 사용하여 이 파일의 컨텐츠를 보고 PRUNE HISTORY 명령을 사용하여 파일에서 항목을 제거할 수 있습니다. 데이터베이스 리스토어 작업 중에, DB2TSCHG.HIS는 백업 이미지로부터 리스토어된 후 데이터베이스 롤 포워드 작업 중에 최근 상태로 가져옵니다. 로그 파일에 대해 사용 가능한 정보가 없는 경우, 모든 테이블 스페이스의 복구에 필요한 것으로 처리됩니다.

각 로그 파일의 정보는 로그가 비활성화된 후 디스크로 플러시되므로, 이 정보는 손상 결과로 손실될 수 있습니다. 이를 보완하기 위해, 로그 파일 중간에 복구 작업이 시작되면 전체 로그가 시스템에 있는 모든 테이블 스페이스의 수정사항을 포함하는 것처럼 처리됩니다. 이후로, 사용 중인 로그가 처리되고 이 로그에 대한 정보가 재빌드됩니다. 오래되거나 아카이브된 로그 파일에 대한 정보가 손상된 상황에서 손실되어 어떤 정보도 데이터 파일에 존재하지 않는 경우, 이 로그 파일은 테이블 스페이스 복구 작업 중에 모든 테이블 스페이스에 대한 수정사항을 포함하는 것처럼 처리됩니다.

테이블 스페이스를 롤 포워드하기 전에 LIST TABLESPACES SHOW DETAIL 명령을 호출하십시오. 이 명령은 테이블 스페이스가 롤 포워드할 수 있는 가장 빠른 특정 시점인 *최소 복구 시간*을 리턴합니다. 최소 복구 시간은 데이터 정의 언어(DDL)문이 테이블 스페이스에 대해, 또는 테이블 스페이스의 테이블에 대해 실행될 때 갱신됩니다. 테이블 스페이스는 최소한 최소 복구 시간까지 롤 포워드하여, 시스템 카탈로그 테이블의 정보와 동기화되도록 해야 합니다. 두 개 이상의 테이블 스페이스를 복구하는 경우 테이블 스페이스는 최소한 복구하는 모든 테이블 스페이스의 가장 높은 최소 복구 시

간까지 롤 포워드해야 합니다. 파티션된 데이터베이스 환경에서 모든 데이터베이스 파티션에 대해 LIST TABLESPACES SHOW DETAIL 명령을 발행하십시오. 테이블 스페이스는 최소한 모든 데이터베이스 파티션에 있는 모든 테이블 스페이스의 가장 높은 최소 복구 시간까지 롤 포워드해야 합니다.

특정 시점까지 테이블 스페이스를 롤 포워드하는데 테이블이 여러 테이블 스페이스에 포함되는 경우, 이 모든 테이블 스페이스를 동시에 롤 포워드해야 합니다. 예를 들어, 테이블 데이터가 하나의 테이블 스페이스에 포함되고 테이블의 인덱스가 다른 테이블 스페이스에 포함되는 경우, 두 테이블 스페이스 모두를 동일한 특정 시점까지 동시에 롤 포워드해야 합니다.

데이터와 테이블에 있는 긴 오브젝트가 별도의 테이블 스페이스에 있는데 긴 오브젝트 데이터가 재구성된 경우, 데이터와 긴 오브젝트 둘 다에 대한 테이블 스페이스를 함께 리스토어하고 롤 포워드해야 합니다. 테이블이 재구성된 후 영향을 받는 테이블 스페이스의 백업을 취해야 합니다.

테이블 스페이스를 특정 시점까지 롤 포워드하고 테이블 스페이스의 테이블이 다음 중 하나인 경우,

- 다른 테이블 스페이스에 있는 구체화된 쿼리 또는 스테이징 테이블에 대한 기본적인 테이블
- 다른 테이블 스페이스에 있는 테이블에 대한 구체화된 쿼리 또는 스테이징 테이블

두 테이블 스페이스 모두를 동일한 특정 시점까지 롤 포워드해야 합니다. 그렇게 하지 않으면, 구체화된 쿼리 또는 스테이징 테이블이 롤 포워드 작업 끝에서 무결성 보류 상태에 놓입니다. 구체화된 쿼리 테이블은 완전히 새로 고쳐져야 하고 스테이징 테이블은 미완료로 표시됩니다.

테이블 스페이스를 특정 시점까지 롤 포워드하려고 하는데 테이블 스페이스의 테이블이 다른 테이블 스페이스에 포함된 다른 테이블과의 참조 무결성 관계에 참여하는 경우, 두 테이블 스페이스 모두 동일한 특정 시점까지 동시에 롤 포워드해야 합니다. 그렇게 하지 않으면, 참조 무결성 관계에 있는 하위 테이블이 롤 포워드 작업 끝에서 무결성 설정 보류 상태에 놓입니다. 나중에 제한조건 위반에 대해 하위 테이블을 점검할 때 전체 테이블에 대한 점검이 필요합니다. 다음 테이블이 존재하면 이 테이블도 하위 테이블과 함께 무결성 설정 보류 상태에 놓입니다.

- 하위 테이블에 대한 하위 구체화된 쿼리 테이블
- 하위 테이블에 대한 하위 스테이징 테이블
- 하위 테이블의 하위 외부 키 테이블

이 테이블들은 무결성 설정 보류 상태에서 벗어나게 하려면 전체 무결성 처리가 필요합니다. 두 테이블 스페이스 모두 동시에 롤 포워드하는 경우, 특정 시점 롤 포워드 작업 끝에서 제한조건이 활성 상태로 유지됩니다.

특정 시점 테이블 스페이스 롤 포워드 작업으로 인해 트랜잭션이 일부 테이블 스페이스에서 롤백되고 다른 테이블 스페이스에서 커미트되지 않도록 하십시오. 다음과 같은 경우에 이러한 상황이 발생할 수 있습니다.

- 트랜잭션에 의해 갱신된 테이블 스페이스 서브세트에 대해 특정 시점 롤 포워드 작업이 수행되는데, 그 특정 시점이 트랜잭션이 커미트된 시간 이전입니다.
- 특정 시점으로 롤 포워드되는 테이블 스페이스에 포함된 테이블이 트리거와 연관되거나, 롤 포워드되는 테이블 스페이스가 아닌 다른 테이블 스페이스에 영향을 주는 트리거에 의해 갱신됩니다.

솔루션은 이와 같은 상황이 발생하지 않도록 할 적합한 특정 시점을 찾는 것입니다.

테이블 스페이스를 롤 포워드할 트랜잭션 일치 특정 시점을 작성하기 위해 QUIESCE TABLESPACES FOR TABLE 명령을 발행할 수 있습니다. Quiesce 요청(공유, 갱신 가능 또는 독점 모드에서)은 해당되는 테이블 스페이스에 대해 실행 중인 모든 트랜잭션이 완료되기를 기다리고(잠금을 통해) 새 요청을 차단합니다. Quiesce 요청이 부여되면, 테이블 스페이스는 일관성 있는 상태가 됩니다. 롤 포워드 작업을 중지할 적합한 시간을 판별하기 위해, 복구 실행기록 파일에서 Quiesce 시점을 찾고 이 시점이 최소 복구 시간 이후에 발생하는지 점검할 수 있습니다.

테이블 스페이스 특정 시점 롤 포워드 작업이 완료되면, 테이블 스페이스는 백업 보류 상태에 놓입니다. 테이블 스페이스의 백업을 취해야 합니다. 롤 포워드한 특정 시점과 현재 시간 사이에 작성된 모든 갱신사항이 제거되었기 때문입니다. 더 이상 이전 데이터베이스 레벨 또는 테이블 스페이스 레벨 백업 이미지에서 현재 시간으로 테이블 스페이스를 롤 포워드할 수 없습니다. 다음 예는 테이블 스페이스 레벨 백업 이미지가 필요한 이유와 사용 방법을 보여줍니다. (테이블 스페이스를 사용 가능하도록 만들려면, 전체 데이터베이스, 백업 보류 상태에 있는 테이블 스페이스, 또는 백업 보류 상태에 있는 테이블 스페이스를 포함하는 테이블 스페이스 세트를 백업하면 됩니다.)

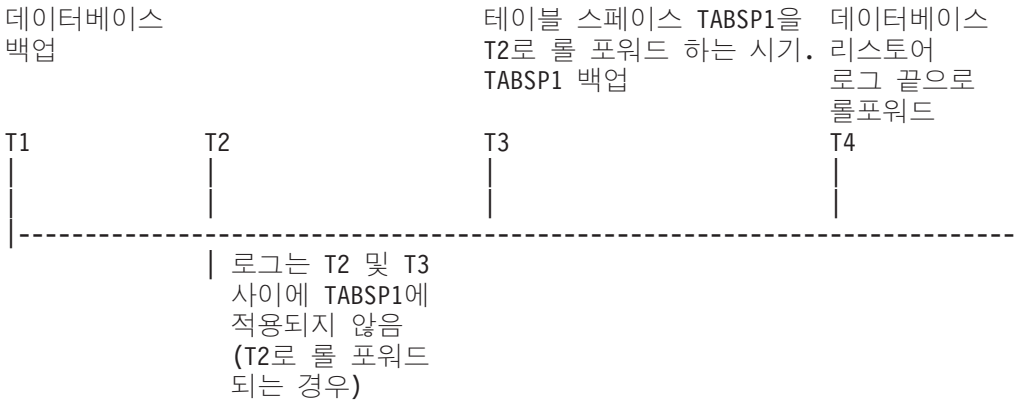


그림 24. 테이블 스페이스 백업 요구사항

이전 예에서, 데이터베이스는 시간 T1에서 백업됩니다. 그런 다음, 시간 T3에서 테이블 스페이스 TABSP1이 특정 시점(T2)으로 롤 포워드되고, 테이블 스페이스는 시간 T3

이후에 백업됩니다. 테이블 스페이스는 백업 보류 상태에 있으므로, 이 백업 작업은 필수입니다. 테이블 스페이스 백업 이미지의 시간소인은 시간 T3 이후이지만, 테이블 스페이스는 시간 T2에 있습니다. T2 및 T3 사이의 로그 레코드는 TABSP1에 적용되지 않습니다. 시간 T4에서, 데이터베이스는 T1에서 작성된 백업 이미지를 사용하여 백업되고 로그 끝까지 롤 포워드됩니다. 테이블 스페이스 TABSP1은 시간 T3에서 리스토어 보류 상태에 놓입니다. 데이터베이스 관리 프로그램은 T2 및 T3 사이의 로그 변경사항이 테이블 스페이스에 적용되지 않고 T3 및 T4 사이에서 TABSP1에 작업이 수행된 것으로 가정합니다. 이 로그 변경사항이 실제로 데이터베이스에 대해 롤 포워드의 일부로 적용된 경우, 이 가정은 올바르지 않습니다. 테이블 스페이스가 특정 시점으로 롤 포워드된 후 취해야 하는 테이블 스페이스 레벨 백업에서는 이전의 특정 시점 롤 포워드 작업(예에서 T3)을 지나 테이블 스페이스를 롤 포워드할 수 있습니다.

테이블 스페이스 TABSP1을 T4까지 복구하려는 경우, T3 이후에 취한 백업 이미지에서 테이블 스페이스를 리스토어하고(필수 백업 또는 나중 백업) TABSP1을 로그 끝까지 롤 포워드합니다.

이전 예에서, 데이터베이스를 시간 T4까지 리스토어하는 가장 효율적인 방법은 다음 순서로 필수 단계를 수행하는 것입니다.

1. 데이터베이스 리스토어
2. 테이블 스페이스 리스토어
3. 데이터베이스 롤 포워드
4. 테이블 스페이스 롤 포워드

데이터베이스를 롤 포워드하기 전에 테이블 스페이스를 리스토어하므로, 자원은 데이터베이스가 롤 포워드될 때 테이블 스페이스에 로그 레코드를 적용하는 데 사용되지 않습니다.

시간 T3 이후의 TABSP1 백업 이미지를 찾을 수 없거나, TABSP1을 T3(또는 이전)로 리스토어하려면 다음을 수행하면 됩니다.

- 테이블 스페이스를 T3로 롤 포워드합니다. 테이블 스페이스는 데이터베이스 백업 이미지로부터 리스토어되었으므로 다시 리스토어하지 않아도 됩니다.
- 시간 T1에서 취한 데이터베이스 백업을 사용하여 테이블 스페이스를 리스토어한 후 시간 T3 이전 시간까지 테이블 스페이스를 롤 포워드합니다.
- 테이블 스페이스를 삭제합니다.

파티션된 데이터베이스 환경에서는,

- 테이블 스페이스의 모든 파트를 동시에 동일한 특정 시점까지 롤 포워드해야 합니다. 그러면 테이블 스페이스가 데이터베이스 파티션 사이에 일관성을 유지합니다.

- 일부 데이터베이스 파티션이 롤 포워드 보류 상태에 있고, 다른 데이터베이스 파티션에서 일부 테이블 스페이스가 롤 포워드 보류 상태에 있는 경우(데이터베이스 파티션은 아님), 먼저 데이터베이스를 롤 포워드한 후 테이블 스페이스를 롤 포워드해야 합니다.
- 테이블 스페이스를 로그 끝까지 롤 포워드하려는 경우, 각각의 데이터베이스 파티션에서 테이블 스페이스를 리스토어하지 않아도 됩니다. 복구가 필요한 데이터베이스 파티션에서만 리스토어하면 됩니다. 그러나 테이블 스페이스를 특정 시점까지 롤 포워드하려는 경우에는 각 데이터베이스 파티션에서 리스토어해야 합니다.

파티션된 테이블이 있는 데이터베이스에서,

- 파티션된 테이블의 조각을 포함하는 테이블 스페이스를 특정 시점까지 롤 포워드하는 경우, 테이블이 있는 다른 모든 테이블 스페이스도 동일한 특정 시점으로 롤 포워드해야 합니다. 그러나 파티션된 테이블의 조각을 포함하는 단일 테이블 스페이스를 로그 끝까지 롤 포워드하는 것은 허용됩니다. 파티션된 테이블에 접속되었거나, 접속 해제되었거나 삭제된 데이터 파티션이 있는 경우 특정 시점 롤 포워드는 이러한 데이터 파티션의 모든 테이블 스페이스를 포함해야 합니다. 파티션된 테이블에 접속, 접속 해제 또는 삭제된 데이터 파티션이 있는지 판별하려면 YSCAT.DATAPARTITIONS 카탈로그 뷰를 쿼리하십시오.

롤 포워드에 필요한 권한 부여

사용자는 특권을 사용하여 데이터베이스 자원을 작성하거나 자원에 액세스할 수 있습니다. 권한 레벨에서는 상위 레벨 데이터베이스 관리 프로그램 유지보수 및 유틸리티 조작과 특권을 그룹화하는 방법을 제공합니다. 이를 통해 데이터베이스 관리 프로그램 및 해당 데이터베이스 오브젝트에 대한 액세스를 함께 제어합니다. 사용자는 적절한 권한(즉, 필요한 특권 또는 권한)을 가지고 있는 오브젝트에만 액세스할 수 있습니다.

롤 포워드 유틸리티를 사용하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한을 가지고 있어야 합니다.

롤 포워드 세션 - CLP 예

예 1

ROLLFORWARD DATABASE 명령은 각각 키워드 AND로 구분되는 다중 조작을 한 번에 지정할 수 있도록 합니다. 예를 들어 로그 끝까지 롤 포워드하고 완료하려면 개별 명령은 다음과 같습니다.

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

다음과 같이 조합할 수 있습니다.

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

둘이 동등하지만 이런 조작은 2단계로 수행하는 것이 좋습니다. 조작을 중지하기 전에 로그를 누락하지 않도록 롤 포워드 작업이 예상대로 진행했는지 검증하는 것이 중요합니다.

롤 포워드 명령이 오류를 발견하면 롤 포워드 작업은 완료되지 않습니다. 오류가 리턴되고, 사용자는 그 오류를 수정한 후 명령을 다시 발행할 수 있습니다. 그러나 오류를 수정할 수 없는 경우 다음을 발행하여 강제로 롤 포워드가 완료되도록 할 수 있습니다.

```
db2 rollforward db sample complete
```

이 명령은 데이터베이스를 온라인에서 실패 이전의 로그 지점으로 가져옵니다.

예 2

로그 끝까지 데이터베이스를 롤 포워드하십시오(두 테이블 스페이스가 리스토어되었음).

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

이들 두 명령문은 동등합니다. 로그 끝까지 테이블 스페이스 롤 포워드 복구에는 AND STOP 또는 AND COMPLETE가 필요하지 않습니다. 테이블 스페이스 이름은 필수가 아닙니다. 지정되지 않는 경우 롤 포워드 복구가 필요한 모든 테이블 스페이스가 포함됩니다. 이들 테이블 스페이스의 서브세트만 롤 포워드될 경우 해당 이름을 지정해야 합니다.

예 3

3개의 테이블 스페이스가 리스토어된 후 하나를 로그 끝까지 롤 포워드하고 다른 둘은 특정 시점으로 롤 포워드하고 둘 다 온라인이 되게 하십시오.

```
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 롤 포워드 조작을 동시에 실행할 수 없습니다. 두 번째 명령은 첫 번째 롤 포워드 조작이 완료된 후에만 호출할 수 있습니다.

예 4

데이터베이스를 리스토어한 후 OVERFLOW LOG PATH를 사용하여 User Exit가 아카이브된 로그를 저장하는 디렉토리를 지정하여 특정 시점으로 롤 포워드하십시오.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

예 5

다음 예에서는 데이터베이스 sample이 있습니다. 데이터베이스가 백업되고 복구 로그는 백업 이미지에 포함됩니다. 데이터베이스는 리스토어된 후 백업 시간소인 끝까지 롤 포워드됩니다.

백업 이미지에 있는 복구 로그를 포함하여 데이터베이스를 백업하십시오.

```
db2 backup db sample online include logs
```

백업 이미지를 사용하여 데이터베이스를 리스토어하십시오.

```
db2 restore db sample
```

백업 시간소인 끝까지 데이터베이스를 롤 포워드하십시오.

```
db2 rollforward db sample to end of backup
```

예 6(파티션된 데이터베이스 환경)

0, 1 및 2의 세 데이터베이스 파티션이 있습니다. 테이블 스페이스 TBS1은 모든 데이터베이스 파티션에서 정의되고, 테이블 스페이스 TBS2는 데이터베이스 파티션 0과 2에서 정의됩니다. 데이터베이스 파티션 1의 데이터베이스를 리스토어하고 데이터베이스 파티션 0 및 2의 TBS1을 리스토어한 후, 데이터베이스 파티션 1에서 데이터베이스를 롤 포워드하십시오.

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

경고 SQL1271(『데이터베이스가 복구되지만 하나 이상의 테이블 스페이스가 데이터베이스 파티션 0 및 2에서 오프라인입니다.』)을 리턴합니다.

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

이 명령은 데이터베이스 파티션 0과 2에서 TBS1을 롤 포워드합니다. 이 경우에 TABLESPACE(TBS1) 절은 선택적입니다.

예 7(파티션된 데이터베이스 환경)

다음 예에서는 파티션된 데이터베이스 sample이 있습니다. 모든 데이터베이스 파티션이 단일 시스템 뷰 백업으로 백업됩니다. 데이터베이스는 모든 데이터베이스 파티션에서 리스토어된 후 백업 시간소인 끝까지 롤 포워드됩니다.

단일 시스템 뷰(SSV) 백업을 수행하십시오.

```
db2 backup db sample on all nodes online include logs
```

모든 데이터베이스 파티션에서 데이터베이스를 리스토어하십시오.

```
db2_all "db2 restore db sample taken at 1998-04-03-14.21.56.245378"
```

백업 시간소인 끝까지 데이터베이스를 롤 포워드하십시오.

```
db2 rollforward db sample to end of backup on all nodes
```

예 8(파티션된 데이터베이스 환경)

다음 예에서는 파티션된 데이터베이스 sample이 있습니다. 모든 데이터베이스 파티션이 db2_all을 사용하여 하나의 명령으로 백업됩니다. 데이터베이스는 모든 데이터베이스 파티션에서 리스토어된 후 백업 시간소인 끝까지 롤 포워드됩니다.

db2_all을 사용하여 하나의 명령으로 모든 데이터베이스 파티션을 백업하십시오.

```
db2_all "db2 backup db sample include logs to /shared/dir/"
```

모든 데이터베이스 파티션에서 데이터베이스를 리스토어하십시오.

```
db2_all "db2 restore db sample from /shared/dir/"
```

백업 시간소인 끝까지 데이터베이스를 롤 포워드하십시오.

```
db2 rollforward db sample to end of backup on all nodes
```

예 9(파티션된 데이터베이스 환경)

데이터베이스 파티션 0과 2만의 테이블 스페이스 TBS1을 리스토어한 후 데이터베이스 파티션 0과 2의 TBS1을 롤 포워드하십시오.

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

데이터베이스 파티션 1은 무시됩니다.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

이 명령은 TBS1이 데이터베이스 파티션 1에서 롤 포워드 복구가 준비되지 않았기 때문에 실패합니다. SQL4906N을 보고합니다.

```
db2 rollforward db sample to end of logs on
   dbpartitionnums (0, 2) tablespace(TBS1)
```

이 명령은 성공적으로 완료됩니다.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

이 명령은 TBS1이 데이터베이스 파티션 1에서 롤 포워드 복구가 준비되지 않았기 때문에 실패합니다. 모든 조각이 함께 롤 포워드되어야 합니다.

주: 테이블 스페이스가 특정 시점으로 롤 포워드될 때 dbpartitionnum 절은 승인되지 않습니다. 롤 포워드 조작은 테이블 스페이스가 있는 모든 데이터베이스 파티션에서 발생해야 합니다.

데이터베이스 파티션 1의 TBS1을 리스토어한 후,

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

이 명령은 성공적으로 완료됩니다.

예 10(파티션된 데이터베이스 환경)

모든 데이터베이스 파티션의 테이블 스페이스를 리스토어한 후, PIT2로 롤 포워드하지만 AND STOP을 지정하지 마십시오. 롤 포워드 작업이 여전히 진행 중입니다. 취소하고 PIT1로 롤 포워드하십시오.

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** restore TBS1 on all dbpartitionnums **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

예 11(파티션된 데이터베이스 환경)

db2nodes.cfg 파일에 나열되는 8개 데이터베이스 파티션(3 - 10)에 있는 테이블 스페이스를 롤 포워드 복구하십시오.

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

로그 끝(특정 시점이 아닌)까지의 이 조작은 성공적으로 완료됩니다. 테이블 스페이스가 있는 데이터베이스 파티션을 지정할 필요는 없습니다. 유틸리티는 db2nodes.cfg 파일이 디폴트입니다.

예 12(파티션된 데이터베이스 환경)

단일 데이터베이스 파티션 데이터베이스 파티션 그룹(데이터베이스 파티션 6의)에 있는 6개의 작은 테이블 스페이스를 롤 포워드 복구하십시오.

```
db2 rollforward database dwtest to end of logs on dbpartitionnum (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

로그 끝(특정 시점이 아닌)까지의 이 조작은 성공적으로 완료됩니다.

예 13(파티션된 테이블 - 모든 데이터 파티션에서 로그 끝까지 롤 포워드)

tbsp0에서 인덱스와 함께 테이블 스페이스 tbsp1, tbsp2, tbsp3을 사용하여 파티션된 테이블이 작성됩니다. 나중에, 사용자는 tbsp4의 테이블에 파티션을 추가하고 tbsp5의 테이블에서 데이터 파티션을 접속합니다. 모든 테이블 스페이스는 END OF LOGS로 롤 포워드될 수 있습니다.

```
db2 rollforward db PBARDB to END OF LOGS and stop
   tablespace(tbsp0, tbsp1, tbsp2, tbsp3, tbsp4, tbsp5)
```

이 명령은 성공적으로 완료됩니다.

예 14(파티션된 테이블 - 하나의 테이블 스페이스에서 로그 끝까지 롤 포워드)

초기에 tbsp0에서 인덱스와 함께 테이블 스페이스 tbsp1, tbsp2, tbsp3을 사용하여 파티션된 테이블이 작성됩니다. 나중에, 사용자는 tbsp4의 테이블에 파티션을 추가하고 tbsp5의 테이블에서 데이터 파티션을 접속합니다. 테이블 스페이스 tbsp4는 손상되어 로그 끝까지 리스토어 및 롤 포워드해야 합니다.

```
db2 rollforward db PBARDB to END OF LOGS and stop tablespace(tbsp4)
```

이 명령은 성공적으로 완료됩니다.

예 15(파티션된 테이블 - 추가, 접속, 접속 해제된 파티션을 포함하여 또는 인덱스와 함께 모든 데이터 파티션의 PIT까지 롤 포워드)

tbsp0에서 인덱스와 함께 테이블 스페이스 tbsp1, tbsp2, tbsp3을 사용하여 파티션된 테이블이 작성됩니다. 나중에, 사용자는 tbsp4의 테이블에 파티션을 추가하고 tbsp5의 테이블에서 데이터 파티션을 접속합니다. 사용자는 INDEX IN 절에 지정된 테이블 스페이스를 포함하여 파티션된 테이블에서 사용되는 모든 테이블 스페이스에 대해 PIT까지 롤 포워드를 수행합니다.

```
db2 rollforward db PBARDB to 2005-08-05-05.58.53.000000 and stop
   tablespace(tbsp0, tbsp1, tbsp2, tbsp3, tbsp4, tbsp5)
```

이 명령은 성공적으로 완료됩니다.

예 16(파티션된 테이블 - 테이블 스페이스 서브세트에 대해 PIT까지 롤 포워드)

세 개의 테이블 스페이스(tbsp1, tbsp2, tbsp3)를 사용하여 파티션된 테이블이 작성됩니다. 나중에, 사용자는 모든 데이터 파티션을 tbsp3에서 접속 해제합니다. PIT까지의 롤 포워드는 tbsp1 및 tbsp2에서만 허용됩니다.

```
db2 rollforward db PBARDB to 2005-08-05-06.02.42.000000 and stop
   tablespace( tbsp1, tbsp2)
```

이 명령은 성공적으로 완료됩니다.

제 15 장 IBM Tivoli Storage Manager(TSM)를 사용한 데이터 복구

BACKUP DATABASE 또는 RESTORE DATABASE 명령을 호출할 때, IBM TSM(Tivoli Storage Manager) 제품을 사용하여 데이터베이스나 테이블 스페이스의 백업 또는 리스토어 작업을 관리할 것을 지정할 수 있습니다. TSM 클라이언트 API의 최소 필수 레벨은 다음의 경우를 제외하고 버전 4.2.0입니다.

- TSM 클라이언트 API 버전 4.2.1이 필요한 64비트 Solaris 시스템.
- TSM 클라이언트 API 버전 5.1이 필요한 64비트 Windows 운영 체제.
- TSM 클라이언트 API 버전 5.3.2가 필요한 모든 Windows X64 시스템.
- TSM 클라이언트 API 버전 5.1.5 이상이 필요한 32비트 Linux for iSeries® 및 pSeries.
- TSM 클라이언트 API 버전 5.2.2 이상이 필요한 64비트 Linux for System i 및 pSeries.
- TSM 클라이언트 API 버전 5.2.0 이상이 필요한 64비트 Linux on AMD Opteron 시스템.
- TSM 클라이언트 API 버전 5.2.2 이상이 필요한 64비트 Linux for zSeries.

Tivoli Storage Manager 클라이언트 구성

데이터베이스 관리 프로그램이 TSM을 사용할 수 있기 전에 TSM 환경을 구성하기 위해 다음 단계가 필요할 수 있습니다.

1. 기능하는 TSM 클라이언트 및 서버가 설치 및 구성되어야 합니다. 또한 TSM 클라이언트 API가 각 DB2 서버에 설치되어야 합니다.
2. TSM 클라이언트 API가 사용하는 환경 변수를 설정하십시오.

   DSMI_DIR

   API 트러스트된 에이전트 파일(dsmtca)이 위치하는 사용자 정의 디렉토리 경로를 식별합니다.

   DSMI_CONFIG

   TSM 사용자 옵션을 포함하는 dsm.opt 파일에 대한 사용자 정의 디렉토리 경로를 식별합니다. 다른 두 변수와는 달리, 이 변수는 완전한 경로 및 파일 이름을 포함해야 합니다.

   DSMI_LOG

   오류 로그(dsierror.log)가 작성될 사용자 정의 디렉토리 경로를 식별합니다.

주: 다중 파티션 데이터베이스 환경에서 이러한 설정값이 sqllib/userprofile 디렉토리에서 지정되어야 합니다.

3. 이러한 환경 변수에 변경이 작성되고 데이터베이스 관리 프로그램이 실행 중인 경우 다음을 수행해야 합니다.
   - db2stop 명령을 사용하여 데이터베이스 관리 프로그램을 중지하십시오.
   - db2start 명령을 사용하여 데이터베이스 관리 프로그램을 시작하십시오.
4. 서버의 구성에 따라서 Tivoli 클라이언트가 TSM 서버와 인터페이스하기 위해 암호가 필요할 수 있습니다. TSM 환경이 PASSWORDACCESS=generate를 사용하도록 구성되는 경우 Tivoli 클라이언트는 암호가 설정되어야 합니다.

   실행할 수 있는 파일 dsmapipw가 인스턴스 소유자의 sqllib/adsm 디렉토리에 설치됩니다. 이 실행 파일을 사용하여 TSM 암호를 설정하고 재설정할 수 있습니다.

   dsmapipw 명령을 실행하려면 로컬 관리자 또는 『루트』 사용자로 로그인해야 합니다. 이 명령이 실행될 때 다음 정보가 사용자에게 프롬프트됩니다.

   - *이전 암호* - TSM 서버가 인식하는 TSM 노드에 대한 현재 암호입니다. 처음 이 명령을 실행할 때, 이 암호는 사용자 노드가 TSM 서버에 등록될 때 TSM 관리자가 제공하는 암호입니다.
   - *새 암호* - TSM 서버에 저장되는 TSM 노드에 대한 새 암호입니다. (입력 오류를 점검하기 위해 새 암호에 대해 두 번 프롬프트됩니다.)

   주: BACKUP DATABASE 또는 RESTORE DATABASE 명령을 호출하는 사용자는 이 암호를 알 필요가 없습니다. 초기 연결을 위해 그리고 암호가 TSM 서버에서 재설정된 후에 암호를 설정하기 위해서만 dsmapipw 명령을 사용해야 합니다.

Tivoli Storage Manager 사용 고려사항

TSM 내에서 특정 기능을 사용하려면, 그 기능을 사용하는 오브젝트의 완전한 경로 이름을 제공해야 할 수도 있습니다. (Windows 운영 체제에서는 ₩가 / 대신 사용됩니다.) 완전한 경로 이름은 다음과 같습니다.

- 전체 데이터베이스 복구 오브젝트:
  /<database>/NODEnnnn/FULL_BACKUP.timestamp.seq_no
- 증분 데이터베이스 복구 오브젝트:
  /<database>/NODEnnnn/DB_INCR_BACKUP.timestamp.seq_no
- 증분 델타 데이터베이스 복구 오브젝트:
  /<database>/NODEnnnn/DB_DELTA_BACKUP.timestamp.seq_no
- 전체 테이블 스페이스 복구 오브젝트:
  /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no

- 증분 테이블 스페이스 복구 오브젝트:
  /<database>/NODEnnnn/TSP_INCR_BACKUP.timestamp.seq_no
- 증분 델타 테이블 스페이스 복구 오브젝트:
  /<database>/NODEnnnn/TSP_DELTA_BACKUP.timestamp.seq_no

여기서 <database>는 데이터베이스 별명 이름이고 NODEnnnn은 노드 번호입니다. 대문자로 표시된 이름은 표시된 대로 입력해야 합니다.

- 동일한 데이터베이스 별명 이름을 사용한 백업 이미지가 여러 개인 경우, 시간소인 및 시퀀스 번호가 완전한 이름의 구별 파트가 됩니다. 사용할 백업 버전을 판별하려면 TSM을 쿼리해야 합니다.
- 온라인 백업 작업을 수행하고 USE TSM 옵션과 INCLUDE LOGS 옵션을 지정하는 경우, 두 프로세스가 동시에 동일한 테이프 드라이브에 쓰려고 하면 교착 상태가 발생할 수 있습니다. 테이프 드라이브를 로그 및 백업 이미지의 스토리지 디바이스로 사용 중인 경우, TSM에 대해 두 개의 독립 테이프 풀을 정의해야 합니다(백업 이미지용 하나와 아카이브된 로그용 하나).

제 16 장 DB2 ACS(Advanced Copy Services)

DB2 ACS(Advanced Copy Services)를 사용하면 스토리지 디바이스의 급속 복사 기술을 사용하여 백업 및 리스토어 작업의 데이터 복사 파트를 수행할 수 있습니다.

전형적인 백업 또는 리스토어 작업에서, 데이터베이스 관리 프로그램은 운영 체제 호출을 사용하여 디스크 또는 스토리지 디바이스로(부터) 데이터를 복사합니다. 스토리지 디바이스를 사용하여 데이터 복사를 수행할 수 있으면 백업 및 리스토어 작업이 한층 빨라집니다. DB2 ACS를 사용하는 백업 작업을 스냅샷 백업이라고 합니다.

스냅샷 백업 및 리스토어 작업을 수행하려면 스토리지 디바이스에 대해 DB2 ACS API 드라이버가 필요합니다. IBM Data Server에 통합된 드라이버는 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage SAN Volume Controller
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS 시리즈

DB2 ACS(Advanced Copy Services) 설정 및 사용에 대한 자세한 지시사항은 http://publib.boulder.ibm.com/tividd/td/IBMTivoliStorageManagerforAdvancedCopyServices5.3.3.html 에 있는 ACS(Advanced Copy Services)에 대한 Tivoli 문서를 참조하십시오.

DB2 ACS(Advanced Copy Services) 사용

DB2 ACS(Advanced Copy Services)를 사용하거나 스냅샷 백업 조작을 수행하려면 DB2 ACS를 설치, 활성화 및 구성해야 합니다.

시작하기 전에

DB2 ACS는 IBM DB2 고가용성(HA) 기능의 파트입니다. DB2 ACS를 사용하려면 DB2 HA 기능에 대한 라이센스가 있어야 합니다.

스냅샷 백업 및 리스토어 작업을 수행하려면 스토리지 디바이스에 대해 DB2 ACS API 드라이버가 필요합니다. IBM Data Server에 통합된 드라이버는 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage SAN Volume Controller
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS 시리즈

프로시저

1. DB2 ACS를 설치하십시오. 『DB2 ACS(Advanced Copy Services) 설치』를 참조하십시오.
2. DB2 ACS를 사용할 데이터베이스 관리 프로그램 인스턴스를 작성하십시오.

   새 데이터베이스 관리 프로그램 인스턴스를 작성할 때 새 인스턴스 `sqllib` 디렉토리에 `acs`라는 디렉토리가 작성됩니다. 각 데이터베이스 관리 프로그램 인스턴스에 `acs` 디렉토리가 있기 때문에 각 데이터베이스 관리 프로그램 인스턴스를 상이하게 구성할 수 있습니다.
3. DB2 ACS를 사용할 각 데이터베이스 관리 프로그램 인스턴스에 대해 다음 단계를 수행하십시오.
   a. DB2 ACS를 활성화하십시오. 347 페이지의 『DB2 ACS(Advanced Copy Services) 활성화』를 참조하십시오.
   b. DB2 ACS를 구성하십시오. 348 페이지의 『DB2 ACS(Advanced Copy Services) 구성』을 참조하십시오.

결과

DB2 ACS를 사용 가능하게 한 후 스냅샷 백업 조작을 수행할 수 있습니다.

DB2 ACS(Advanced Copy Services) 설정 및 사용에 대한 자세한 지시사항은 http://publib.boulder.ibm.com/tividd/td/IBMTivoliStorageManagerforAdvancedCopyServices5.3.3.html

DB2 ACS(Advanced Copy Services) 설치

DB2 ACS(Advanced Copy Services)에 필요한 파일과 라이브러리는 IBM Data Server 설치 프로그램에 의해 설치됩니다.

제한사항

DB2 ACS는 IBM Data Server가 지원하는 하드웨어 및 운영 체제의 서브세트를 지원합니다. DB2 ACS가 지원하는 하드웨어 및 운영 체제의 목록은 401 페이지의 『DB2 ACS(Advanced Copy Services) 지원 운영 체제 및 하드웨어』를 참조하십시오.

시작하기 전에

ACS를 설치하기 전에 다음 라이브러리가 설치되어 있어야 합니다.

AIX:

- ln -s /opt/freeware/lib/powerpc-ibm-aix5.3.0//libgcc_s.a /usr/lib/libgcc_s.a

Red Hat Enterprise Linux:

- ln -s libssl.so.0.9.7xxx libssl.so.0.9.7
- ln -s libcrypto.so.0.9.7xxx libcrypto.so.0.9.7
- ln -s libssl.so.0.9.7xxx libssl.so
- ln -s libssl.so.0.9.7xxx libssl.so.0

프로시저

1. IBM Data Server를 설치하십시오.
2. TCP/IP 서비스 파일에 DB2 ACS 에이전트에 대한 포트를 추가하십시오. 예를 들어, 다음과 같습니다.

```
db2acs  5400/tcp # DB2 ACS service port
```

다음 단계

DB2 ACS를 설치한 후 DB2 ACS 활성화 및 DB2 ACS 구성 작업을 수행해야 스냅샷 백업 조작을 수행할 수 있습니다.

DB2 ACS(Advanced Copy Services) 설정 및 사용에 대한 자세한 지시사항은 http://publib.boulder.ibm.com/tividd/td/IBMTivoliStorageManagerforAdvancedCopyServices5.3.3.html

DB2 ACS(Advanced Copy Services) 활성화

DB2 ACS(Advanced Copy Services)를 사용하여 주어진 데이터베이스 관리 프로그램 인스턴스에 대한 스냅샷 백업을 수행할 수 있기 전에 해당 인스턴스에서 DB2 ACS 기능을 활성화해야 합니다. 스크립트를 실행하여 DB2 ACS를 활성화합니다.

시작하기 전에

DB2 ACS를 활성화할 수 있기 전에 다음 태스크를 수행해야 합니다.

1. DB2 ACS 설치
2. DB2 ACS를 사용할 데이터베이스 관리 프로그램 인스턴스를 작성하십시오.

이 태스크에 대한 정보

데이터베이스 관리 프로그램은 데이터베이스 관리 프로그램 인스턴스 작성 중에, 그리고 IBM Data Server를 업그레이드할 때 자동으로 setup.sh를 호출하여 DB2 ACS 기능을 활성화합니다.

또한 수동으로 DB2 ACS를 활성화할 수도 있습니다.

프로시저

DB2 ACS를 수동으로 활성화하려면 루트 권한이 있는 사용자로서 DB2 ACS를 활성화하기 위해 적합한 매개변수를 사용하여 setup.sh 스크립트를 실행하십시오. setup.sh에 대한 자세한 정보는 350 페이지의 『DB2 ACS(Advanced Copy Services) 설정 스크립트 setup.sh』를 참조하십시오.

결과

setup.sh 스크립트 실행의 중요한 한 가지 결과는 sqllib/acs 디렉토리에 있는 DB2 ACS 실행 가능 파일의 소유권 및 사용 권한이 검증되는 것입니다.

다음 단계

DB2 ACS를 활성화한 후, DB2 ACS 구성 작업을 수행해야 스냅샷 백업 조작을 수행할 수 있습니다.

DB2 ACS(Advanced Copy Services) 설정 및 사용에 대한 자세한 지시사항은 http://publib.boulder.ibm.com/tividd/td/IBMTivoliStorageManagerforAdvancedCopyServices5.3.3.html

DB2 ACS(Advanced Copy Services) 구성

DB2 ACS(Advanced Copy Services)를 사용하여 스냅샷 백업을 수행할 수 있으려면 먼저 DB2 ACS를 구성해야 합니다. 구성 파일을 사용하여 DB2 ACS를 구성할 수 있습니다.

시작하기 전에

DB2 ACS를 구성할 수 있기 전에 다음 태스크를 수행해야 합니다.

1. DB2 ACS 설치
2. DB2 ACS를 사용할 데이터베이스 관리 프로그램 인스턴스를 작성하십시오.
3. DB2 ACS 활성화

프로시저

sqllib/acs 디렉토리에서 매개변수 없이 setup.sh 스크립트를 실행하십시오. 그러면 DB2 ACS를 구성할 대화식 텍스트 기반 마법사가 표시됩니다. 이 마법사는 구성 프로파일 파일을 작성하고 머신에서 /etc/initab를 수정하여 DB2 ACS 디먼의 시작을 트리거

합니다.

다음은 setup.sh 마법사의 샘플 출력입니다.

```
./setup.sh
Do you have a full TSM license to enable all features of TSM for ACS ?[y/n]

n

****** Profile parameters for section GLOBAL: ******
ACS_DIR [/home/krodger/sqllib/acs ]
ACSD [localhost 57328 ]
TRACE [NO ]

****** Profile parameters for section ACSD: ******
ACS_REPOSITORY *mandatory parameter* /home/krodger/acsrepository

****** Profile parameters for section CLIENT: ******
MAX_VERSIONS [ADAPTIVE ] 2
LVM_FREEZE_THAW [YES ]
DEVICE_CLASS [STANDARD ]

****** Profile parameters for section STANDARD: ******
COPYSERVICES_HARDWARE_TYPE *mandatory parameter*
NAS_NSERIES COPYSERVICES_PRIMARY_SERVERNAME *mandatory parameter* fas960a
COPYSERVICES_USERNAME [superuser ] root

=====================================================================

The profile has beeen successfully created.
Do you want to continue by specifying passwords for the defined devices? [y/n]

y

Please specify the passwords for the following profile sections:
STANDARD
master

Creating password file at /home/krodger/sqllib/acs/shared/pwd.acsd.
A copy of this file needs to be available to all components that connect to acsd.

BKI1555I: Profile successfully created. Performing additional checks.
Make sure to restart all ACS components to reload the profile.
```

결과

DB2 ACS를 구성한 후 스냅샷 백업 조작을 수행할 수 있습니다.

DB2 ACS(Advanced Copy Services) 설정 및 사용에 대한 자세한 지시사항은 http://publib.boulder.ibm.com/tividd/td/IBMTivoliStorageManagerforAdvancedCopyServices5.3.3.html

DB2 ACS(Advanced Copy Services) 디렉토리 구성

새 데이터베이스 관리 프로그램 인스턴스를 작성할 때 새 인스턴스 sqllib 디렉토리에 acs라는 디렉토리가 작성됩니다. DB2 ACS(Advanced Copy Services)가 이 acs 디렉토리를 사용하여 대상 볼륨 제어 파일 및 복구 오브젝트에 대한 공유 저장소 같은 구성 파일을 저장합니다. 이 acs 디렉토리를 변경하거나 구성할 수 있는 방법에 대한 제한사항이 있습니다.

1. acs 디렉토리는 어떤 DB2 ACS 또는 스냅샷 백업 조작에도 관련되지 않아야 합니다.
2. acs 디렉토리는 IBM TSM(Tivoli Storage Manager)을 사용하여 모든 데이터베이스 파티션 및 스냅샷 백업에 대한 백업 시스템에서 NFS 익스포트 및 NFS 공유될 수 있습니다.

DB2 ACS(Advanced Copy Services) 설정 스크립트 setup.sh

setup.sh 스크립트는 DB2 ACS(Advanced Copy Services)를 활성화하고 구성합니다.

위치

스크립트 setup.sh는 sqllib/acs 디렉토리에 위치됩니다.

구문

다음은 setup.sh의 구문입니다.

```
usage: setup.sh -a <action>
                -d <DB2_Instance_Directory>
                -u <Instance_user_ID_name>
                -g <Instance_primary_group_name>
```

여기서 action은 다음 중 하나가 될 수 있습니다.

- start
- stop
- query
- enable
- disable

사용법

데이터베이스 관리 프로그램은 데이터베이스 관리 프로그램 인스턴스 작성 중에, 그리고 IBM Data Server를 업그레이드할 때 자동으로 setup.sh를 호출하여 DB2 ACS 기능을 활성화합니다.

다음과 같이 setup.sh 스크립트를 수동으로 호출할 수도 있습니다.

DB2 ACS 활성화

루트 권한이 있는 사용자로서 위에 설명된 매개변수를 사용하여 setup.sh를 실행하여 DB2 ACS를 활성화할 수 있습니다.

DB2 ACS 구성

어떤 매개변수도 사용하지 않고 setup.sh를 실행하여 DB2 ACS를 구성할 수 있습니다. 매개변수 없이 setup.sh를 실행하면 마법사는 DB2 ACS 구성을 통해 사용자를 안내합니다.

setup.sh 스크립트 실행의 중요한 한 가지 결과는 sqllib/acs 디렉토리에 있는 DB2 ACS 실행 가능 파일의 소유권 및 사용 권한이 검증되는 것입니다.

DB2 ACS(Advanced Copy Services) API

DB2 ACS(Advanced Copy Services) API는 데이터베이스 관리 프로그램이 스냅샷 백업 작업을 수행하기 위해 스토리지 하드웨어와 통신하는 데 사용하는 기능 세트를 정의합니다.

스냅샷 백업 및 리스토어 작업을 수행하려면 스토리지 디바이스에 대해 DB2 ACS API 드라이버가 필요합니다. IBM Data Server에 통합된 드라이버는 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage SAN Volume Controller
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS 시리즈

DB2 ACS(Advanced Copy Services) API 함수

데이터베이스 관리 프로그램은 DB2 ACS API 함수를 통해 DB2 ACS 요청을 스토리지 하드웨어로 통신합니다.

db2ACSQueryApiVersion - DB2 ACS(Advanced Copy Services) API의 현재 버전 리턴

DB2 ACS(Advanced Copy Services) API의 현재 버전을 리턴합니다.

API 내장 파일

```
db2ACSApi.h
```

API 및 데이터 구조 구문

```
db2ACS_Version db2ACSQueryApiVersion();
```

매개변수

없음.

사용 시 참고사항

가능한 리턴 값:

- DB2ACS_API_VERSION1
- DB2ACS_API_VERSION_UNKNOWN

db2ACSInitialize - DB2 ACS(Advanced Copy Services) 세션 초기화

새 DB2 ACS(Advanced Copy Services) 세션을 초기화합니다. 이 호출은 데이터베이스 관리 프로그램의 DB2 ACS 라이브러리와 스토리지 하드웨어에 대한 DB2 ACS API 드라이버 사이의 통신을 설정합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
/* ============================================================================
 * Session Initialization
 * ============================================================================ */
db2ACS_RC db2ACSInitialize(
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

db2ACSInitialize()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->session
pControlBlock->options

DB2 ACS API 드라이버가 리턴하기 전에 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 11. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INIT_FAILED	데이터베이스 관리 프로그램이 DB2 ACS 세션을 초기화하려고 했지만 초기화에 실패했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_COMM_ERROR	테이프 드라이브 같은 스토리지 디바이스에 통신 오류가 있었습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_NO_DEV_AVAIL	현재 사용할 수 있는 테이프 드라이브 같은 스토리지 디바이스가 없습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.

- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램이 db2ACSQueryAPIVersion() 호출을 제외한 DB2 ACS API를 호출할 수 있기 전에, 데이터베이스 관리 프로그램이 db2ACSInitialize()를 호출해야 합니다. 데이터베이스 관리 프로그램이 db2ACSInitialize()를 호출하여 DB2 ACS 세션을 설정한 후에는 데이터베이스 관리 프로그램이 DB2 ACS 쿼리, 읽기, 쓰기 또는 삭제 조작의 모든 조합을 수행할 수 있습니다. 데이터베이스 관리 프로그램은 db2ACSTerminate()를 호출하여 DB2 ACS 세션을 종료할 수 있습니다.

db2ACSTerminate - DB2 ACS(Advanced Copy Services) 세션 종료

DB2 ACS(Advanced Copy Services) 세션을 종료합니다.

내장 파일

```
db2ACSApi.h
```

구문 및 데이터 구조

```
/* ============================================================================
 * Session Termination
 * ============================================================================ */
db2ACS_RC db2ACSTerminate(
              db2ACS_CB              * pControlBlock,
              db2ACS_ReturnCode      * pRC );
```

매개변수

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSInitialize()를 호출하기 전에 데이터베이스 관리 프로그램이 이 매개변수에 대한 메모리를 할당했습니다. 데이터베이스 관리 프로그램이 db2ACSTerminate() 후 이 메모리를 비워야 합니다.

db2ACSTerminate()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->options

DB2 ACS API 드라이버가 pControlBlock->vendorInfo.vendorCB의 메모리를 무효화하고 비울 수 있습니다.

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 12. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	이 세션에 할당된 모든 메모리를 사용 가능화하고 종료합니다.
DB2ACS_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

DB2 ACS API 드라이버가 db2ACSTerminate()에서 DB2 ACS 세션에 대해 할당한 모든 메모리를 비워야 합니다.

db2ACSTerminate()가 오류 없이 완료하는지 여부와 상관없이, 데이터베이스 관리 프로그램은 먼저 db2ACSInitialize()를 호출하지 않으면 이 DB2 ACS 세션에서 어떤 DB2 ACS 함수도 호출할 수 없습니다.

db2ACSPrepare - 스냅샷 백업 작업 수행 준비

스냅샷 백업이 수행될 때 데이터베이스 관리 프로그램이 데이터베이스를 일시중단합니다. db2ACSPrepare()는 데이터베이스 관리 프로그램이 데이터베이스를 일시중단하는 시점까지(포함하지는 않음) 스냅샷 백업 작업을 수행하기 위해 준비할 모든 단계를 수행합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
/* ============================================================================
 * Prepare
 * ============================================================================ */
db2ACS_RC db2ACSPrepare(
               db2ACS_GroupList        * pGroupList,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

pGroupList

데이터 유형: db2ACS_GroupList *

db2ACS_GroupList에는 스냅샷 백업 작업에 포함될 그룹의 목록이 들어 있습니다.

pGroupList가 NULL이면 모든 그룹(경로)이 스냅샷 백업 작업에 포함됩니다.

pGroupList가 NULL이 아닌 경우,

- **pGroupList**가 스냅샷 백업 작업에 포함될 그룹(경로)의 목록을 포함합니다.
- 데이터베이스 관리 프로그램이 **pGroupList**에 대한 메모리를 할당하고 비워야 합니다.
- 데이터베이스 관리 프로그램이 **pGroupList**를 db2ACSPrepare()로 전달하기 전에 다음 필드를 채웁니다.

  pGroupList->numGroupID
  pGroupList->id

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSPrepare()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode`에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 `db2ACS_ReturnCode` 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 13. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.

- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

db2ACSPrepare()가 성공하면 데이터베이스 관리 프로그램은 db2ACSSnapshot()을 호출하기 전에 데이터베이스를 일시중단합니다.

db2ACSBeginOperation - DB2 ACS(Advanced Copy Services) 조작 시작

DB2 ACS(Advanced Copy Services) 조작을 시작합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
/* ****************************************************************************
 * Operation Begin
 *
 * A valid ACS operation is specified by passing an ObjectType OR'd with one of
 * the following Operations, such as:
 *
 *    (DB2ACS_OP_CREATE | DB2ACS_OBJTYPE_SNAPSHOT)
 * ************************************************************************** */
db2ACS_RC db2ACSBeginOperation(
               db2ACS_Operation           operation,
               db2ACS_CB                * pControlBlock,
               db2ACS_ReturnCode        * pRC );
```

매개변수

operation

데이터 유형: db2ACS_Operation.

operation은 시작할 DB2 ACS 조작과 관련된 오브젝트의 유형을 표시하는 비트 마스크입니다.

조작 유형:

DB2ACS_OP_CREATE
DB2ACS_OP_READ
DB2ACS_OP_DELETE

오브젝트 유형:

DB2ACS_OBJTYPE_BACKUP
DB2ACS_OBJTYPE_LOG

DB2ACS_OBJTYPE_LOADCOPY
DB2ACS_OBJTYPE_SNAPSHOT

예: (DB2ACS_OP_CREATE | DB2ACS_OBJTYPE_SNAPSHOT) 또는
(DB2ACS_OP_DELETE | DB2ACS_OBJTYPE_LOADCOPY).

데이터베이스 관리 프로그램이 **operation**을 db2ACSBeginOperation() 함수 호출로 전달합니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSBeginOperation()을 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

operation이 DB2ACS_OP_CREATE 또는 DB2ACS_OP_READ인 경우 데이터베이스 관리 프로그램은 다음 필드로 채웁니다.

pControlBlock->operation

pControlBlock->operation에 포함된 정보는 특정 DB2 ACS 조작의 컨텍스트에서만 유효합니다. pControlBlock->operation은 db2ACSBeginOperation() 중에 설정되며, db2ACSEndOperation()이 리턴할 때까지 변경되지 않습니다. 데이터베이스 관리 프로그램이나 DB2 ACS API 드라이버가 DB2 ACS 조작의 범위를 벗어나서 pControlBlock->operation을 참조해서는 안됩니다.

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 14. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_OPTIONS	데이터베이스 관리 프로그램이 유효하지 않은 옵션을 지정했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

없음.

db2ACSEndOperation - DB2 ACS(Advanced Copy Services) 조작 종료

DB2 ACS(Advanced Copy Services) 조작을 종료합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
/* ============================================================================
 * Operation End
 * ============================================================================ */
db2ACS_RC db2ACSEndOperation(
               db2ACS_EndAction          endAction,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

endAction

데이터 유형: db2ACS_EndAction.

endAction은 DB2 ACS API 드라이버가 DB2 ACS 조작을 종료해야 하는 방법을 표시하는 비트 마스크입니다.

값:

DB2ACS_END_COMMIT
DB2ACS_END_ABORT

데이터베이스 관리 프로그램이 **endAction**을 db2ACSEndOperation() 함수 호출로 전달합니다.

pControlBlock

데이터 유형: db2ACS_CB

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSEndOperation()을 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 15. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	

표 15. 리턴 코드 (계속)

리턴 코드	설명	주
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_COMMIT_FAILED	DB2 ACS API 드라이버가 트랜잭션을 커미트할 수 없습니다.	
DB2ACS_RC_ABORT_FAILED	데이터베이스 관리 프로그램이 DB2 ACS 조작을 중단하려고 했지만 중단 시도가 실패했습니다.	

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램이 DB2ACS_END_ABORT를 **endAction** 매개변수로서 전달하는 경우 결과는 스냅샷 백업 오브젝트가 삭제되는 것이어야 합니다.

db2ACSBeginQuery - 스냅샷 백업 오브젝트에 관한 쿼리 시작

리스토어 작업에 사용할 수 있는 스냅샷 백업 오브젝트에 관한 DB2 ACS(Advanced Copy Services) 쿼리 조작을 시작합니다.

내장 파일

```
db2ACSApi.h
```

구문 및 데이터 구조

```
db2ACS_RC db2ACSBeginQuery(
              db2ACS_QueryInput       * pQueryInput,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

매개변수

pQueryInput

데이터 유형: db2ACS_QueryInput *

db2ACS_QueryInput은 db2ACS_ObjectInfo와 동일한 필드를 갖습니다. db2ACS_ObjectInfo에는 DB2 ACS(Advanced Copy Services) API를 사용하여 작성된 오브젝트에 관한 정보가 들어 있습니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

db2ACSBeginQuery()를 호출하기 전에 데이터베이스 관리 프로그램이 **pQueryInput**의 필드를 채웁니다.

DB2 ACS API 드라이버는 쿼리에서 다음 와일드카드의 사용을 지원해야 합니다.

- 문자열 필드에서 DB2ACS_WILDCARD
- 데이터베이스 파티션 필드에 대한 DB2ACS_ANY_PARTITIONNUM
- 32비트 부호 없는 정수(Uint32) 필드에 대한 DB2ACS_ANY_UINT32

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSBeginQuery()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 16. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

db2ACSBeginQuery()는 어떤 쿼리 데이터도 리턴하지 않습니다.

db2ACSGetNextObject - 리스토어에 사용할 수 있는 다음 스냅샷 백업 오브젝트 나열

리스토어 작업에 사용할 수 있는 스냅샷 백업 오브젝트의 목록에 있는 다음 항목을 리턴합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
db2ACS_RC db2ACSGetNextObject(
               db2ACS_QueryOutput      * pQueryOutput,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

pQueryOutput

데이터 유형: db2ACS_QueryOutput *

db2ACS_QueryOutput에는 스냅샷 백업 오브젝트에 관한 쿼리 결과 정보가 들어 있습니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버는 리턴하기 전에 **pQueryOutput**의 필드를 채웁니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSGetNextObject()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 17. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램이 지정하는 스냅샷 백업 오브젝트를 찾을 수 없습니다.	함수 호출은 실패하지 않았지만, db2ACSBeginQuery()에 전달된 기준과 일치하는 스냅샷 백업 오브젝트가 없습니다.
DB2ACS_RC_END_OF_DATA	DB2 ACS API 드라이버가 추가 스냅샷 백업 오브젝트를 찾을 수 없습니다.	함수 호출은 실패하지 않았지만, db2ACSBeginQuery()에 전달된 기준과 일치하는 추가 스냅샷 백업 오브젝트가 없습니다.
DB2ACS_RC_MORE_DATA	스토리지 위치에서 데이터베이스 관리 프로그램으로 전송될 추가 데이터가 있습니다.	db2ACSBeginQuery()에 전달된 기준과 일치하는 스냅샷 백업 오브젝트에 관한 정보가 리턴되며, db2ACSBeginQuery()에 전달된 기준과 일치하는 추가 스냅샷 백업 오브젝트가 있습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램은 db2ACSGetNextObject()를 호출하기 전에 db2ACSBeginQuery()를 호출해야 합니다. 데이터베이스 관리 프로그램이 db2ACSBeginQuery()에 전달된 db2ACS_QueryInput 매개변수에서 검색 기준을 지정합니다.

db2ACSGetNextObject()는 db2ACSBeginQuery()에 전달된 검색 기준과 일치하는 하나의 스냅샷 백업 오브젝트에 관한 정보를 리턴합니다. db2ACSGetNextObject()가 DB2ACS_RC_MORE_DATA를 리턴하는 경우, 데이터베이스 관리 프로그램은 db2ACSGetNextObject()를 다시 호출하여 검색 기준과 일치하는 다른 스냅샷 백업 오브젝트에 관한 정보를 수신할 수 있습니다. db2ACSGetNextObject()가 DB2ACS_RC_END_OF_DATA를 리턴하는 경우, 검색 기준과 일치하는 추가 스냅샷 백업 오브젝트가 없습니다.

db2ACSEndQuery - 스냅샷 백업 오브젝트에 관한 쿼리 종료

데이터베이스 관리 프로그램은 DB2 ACS(Advanced Copy Services) API 함수 db2ACSBeginQuery() 및 db2ACSGetNextObject()를 사용하여 리스토어 작업에 사용할 수 있는 스냅샷 백업 오브젝트에 관해 쿼리합니다. db2ACSEndQuery()가 해당 DB2 ACS 쿼리 세션을 종료합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
db2ACS_RC db2ACSEndQuery(
              db2ACS_CB              * pControlBlock,
              db2ACS_ReturnCode      * pRC );
```

매개변수

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSEndQuery()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 18. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램은 먼저 db2ACSBeginQuery()를 다시 호출하지 않으면 이 DB2 ACS 세션에서 db2ACSGetNextObject()를 다시 호출할 수 없습니다.

db2ACSSnapshot - DB2 ACS(Advanced Copy Services) 조작 수행

DB2 ACS(Advanced Copy Services) 조작을 수행합니다.

내장 파일

```
db2ACSApi.h
```

구문 및 데이터 구조

```
typedef union db2ACS_ReadList
{
   db2ACS_GroupList           group;
} db2ACS_ReadList;


db2ACS_RC db2ACSSnapshot(
               db2ACS_Action             action,
               db2ACS_ObjectID           objectID,
               db2ACS_ReadList         * pReadList,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

action 데이터 유형: db2ACS_Action

수행할 DB2 ACS 조치의 유형입니다. 값:

DB2ACS_ACTION_WRITE
DB2ACS_ACTION_READ_BY_OBJECT
DB2ACS_ACTION_READ_BY_GROUP

데이터베이스 관리 프로그램이 **action**을 db2ACSSnapshot()으로 전달합니다.

objectID

데이터 유형: db2ACS_ObjectID

db2ACS_ObjectID는 스토리지 저장소에 대한 쿼리가 리턴하는 각 저장된 오브젝트에 대한 고유 ID입니다. db2ACS_ObjectID는 단일 DB2 ACS 세션의 시간 프레임 안에서만 고유하고 지속되는 것으로 보장됩니다.

데이터베이스 관리 프로그램이 db2ACSBeginOperation()에 대한 호출에서 DB2ACS_OP_READ 또는 DB2ACS_OP_DELETE를 **operation**으로 지정한 경우, 데이터베이스 관리 프로그램은 **objectID**에 대한 값을 db2ACSSnapshot()으로 전달합니다.

pReadList

데이터 유형: db2ACS_ReadList *

db2ACS_ReadList에는 그룹 목록이 들어 있습니다.

pReadList는 **action**이 DB2ACS_ACTION_READ_BY_GROUP인 경우에만 사용됩니다.

action이 DB2ACS_ACTION_READ_BY_GROUP인 경우, 데이터베이스 관리 프로그램이 db2ACSSnapshot()을 호출하기 전에 **pReadLIst**의 필드를 채우고 그 이후에 **pReadList**에 대한 메모리를 비워야 합니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSSnapshot()을 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 19. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

표 19. 리턴 코드 (계속)

리턴 코드	설명	주
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램은 db2ACSPartition(), db2ACSPrepare() 및 db2ACSSnapshot()을 호출하기 전에 db2ACSBeginOperation()을 호출합니다. 데이터베이스 관리 프로그램이 db2ACSBeginOperation()에 대한 호출의 **operation** 매개변수에 DB2 ACS API 드라이버가 수행해야 하는 DB2 ACS 조작의 유형을 지정합니다.

db2ACSPartition - 데이터베이스 파티션에 대한 목표 데이터를 함께 그룹화

데이터베이스 관리 프로그램이 데이터베이스 파티션에 속하는 것으로 나열하는 각 경로를 그룹 ID와 연관시킵니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
/* ============================================================================
 * Partition
 * ============================================================================ */
db2ACS_RC db2ACSPartition(
              db2ACS_PathList         * pPathList,
```

```
db2ACS_CreateObjectInfo * pCreateObjInfo,
db2ACS_CB               * PControlBlock,
db2ACS_ReturnCode       * pRC );
```

매개변수

pPathList

데이터 유형: db2ACS_PathList

db2ACS_PathList에는 DB2 ACS 조작에 특정한 각 경로에 관한 일부 추가 정보를 포함하여 데이터베이스 경로의 목록이 들어 있습니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

db2ACS_PathList 구조의 **entry** 필드는 db2ACS_PathEntry 유형 요소의 배열입니다. db2ACS_PathEntry에는 데이터베이스 경로에 관한 정보가 들어 있습니다.

db2ACSPartition을 호출하기 전에 데이터베이스 관리 프로그램이 **pPathList**의 각 db2ACS_PathEntry 항목의 다음 필드를 채웁니다.

- **path**
- **type**
- **toBeExcluded**

데이터베이스 관리 프로그램이 이 데이터베이스 파티션에 속하는 것으로 식별하는 모든 경로에 DB2 ACS API 드라이버에 의해 식별되는 그룹이 제공됩니다. DB2 ACS API 드라이버는 리턴하기 전에 **pPathList**의 각 db2ACS_PathEntry의 **groupID** 필드를 채웁니다.

pCreateObjInfo

데이터 유형: db2ACS_CreateObjectInfo

db2ACS_CreateObjectInfo에는 DB2 ACS 백업 오브젝트 작성에 관한 정보가 들어 있습니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

데이터베이스 관리 프로그램이 db2ACSPartition을 호출하기 전에 **pCreateObjInfo**의 필드를 채웁니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSPartition()을 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode`에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 `db2ACS_ReturnCode` 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 20. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INIT_FAILED	데이터베이스 관리 프로그램이 DB2 ACS 세션을 초기화하려고 했지만 초기화에 실패했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_OBJ_OUT_OF_SCOPE	데이터베이스 관리 프로그램이 DB2 ACS API 드라이버에 의해 관리되지 않는 복구 오브젝트에 대해 DB2 ACS 조작을 수행하려고 시도했습니다.	

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

DB2 ACS(Advanced Copy Services)가 단일 데이터베이스 파티션의 데이터를 극소적으로 처리합니다. 즉, 한 데이터베이스 파티션에 대한 데이터가, 조치가 다중 데이터베이스 파티션과 연관된 조작의 파트일 때도 다른 데이터베이스 파티션과 무관하게 함께 백업되거나 리스토어됩니다. db2ACSPartition은 단일 데이터베이스 파티션에 대한 데이터베이스 경로 정보를 함께 그룹화합니다.

데이터베이스 관리 프로그램은 db2ACSSnapshot을 호출하기 전에 db2ACSPartition을 호출합니다. 데이터베이스 관리 프로그램은 **pPathList** 매개변수에서 이 데이터베이스 파티션과 연관된 모든 경로를 나열합니다. 데이터베이스 관리 프로그램은 db2ACSSnapshot으로 전달되는 **pReadList** 매개변수에 경로의 해당 서브세트를 지정하여 **pPathList**에 나열되는 경로의 서브세트에 대해 DB2 ACS 조작을 수행할 수 있습니다.

db2ACSVerify - DB2 ACS(Advanced Copy Services) 조작이 성공적으로 완료했는지 검증

DB2 ACS(Advanced Copy Services) 조작이 성공했는지 검증합니다.

내장 파일

```
db2ACSApi.h
```

구문 및 데이터 구조

```
/* ============================================================================
 * Verify
 * ============================================================================ */
db2ACS_RC db2ACSVerify(
               db2ACS_PostObjectInfo   * pPostObjInfo,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

pPostObjInfo

데이터 유형: db2ACS_PostObjectInfo

db2ACS_DB2ID는 스냅샷 백업 오브젝트 작성 시간에는 알려질 수 없지만 오브젝트 저장소에 유지보수되어야 하는 데이터 세트입니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

데이터베이스 관리 프로그램은 db2ACSVerify를 호출하기 전에 **pPostObjInfo**의 필드를 채웁니다. **pPostObjInfo**에 DB2 ACS 조작 후에 관련되는 정보가 들어 있습니다. 예를 들어 성공적인 스냅샷 백업 후에 **pPostObjInfo**는 처음에 사용되는 로그 파일을 포함할 수 있습니다. DB2 ACS 조작 이후에 대해 관련된 데이터가 없는 경우 데이터베이스 관리 프로그램은 **pPostObjInfo**를 NULL로 설정합니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSVerify()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 21. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

db2ACSVerify가 스냅샷 백업 작업이 성공했음을 리턴하는 경우 스냅샷 백업에 의해 생성되는 복구 오브젝트를 리스토어 작업에 사용할 수 있음을 의미합니다.

db2ACSDelete - DB2 ACS(Advanced Copy Services)를 사용하여 작성된 복구 오브젝트 삭제

DB2 ACS(Advanced Copy Services)를 사용하여 작성된 복구 오브젝트를 삭제합니다.

내장 파일

db2ACSApi.h

구문 및 데이터 구조

```
/* ============================================================================
 * Delete
 * ============================================================================ */
db2ACS_RC db2ACSDelete(
               db2ACS_ObjectID           objectID,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

objectID

데이터 유형: db2ACS_ObjectID

db2ACS_ObjectID는 스토리지 저장소에 대한 쿼리가 리턴하는 각 저장된 오브젝트에 대한 고유 ID입니다. db2ACS_ObjectID는 단일 DB2 ACS 세션의 시간 프레임 안에서만 고유하고 지속되는 것으로 보장됩니다.

데이터베이스 관리 프로그램은 db2ACSQuery()를 사용하여 db2ACSDelete()에 전달할 유효한 **objectID**를 얻을 수 있습니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSDelete()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 22. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	지정된 오브젝트가 삭제됩니다. 해당 오브젝트에 대해 추가 DB2 ACS 조작을 수행할 수 없습니다.
DB2ACS_RC_DELETE_FAILED	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램에 의해 지정된 스냅샷 백업 오브젝트를 성공적으로 삭제할 수 없습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램이 지정하는 스냅샷 백업 오브젝트를 찾을 수 없습니다.	

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램이 db2ACSDelete를 호출할 때, DB2 ACS API 드라이버가 **objectID**로 식별되는 복구 오브젝트를 삭제합니다.

사용자가 DELETE 매개변수와 함께 db2acsutil을 호출할 때 데이터베이스 관리 프로그램이 db2ACSDelete를 호출합니다.

db2ACSStoreMetaData - DB2 ACS(Advanced Copy Services)를 사용하여 생성된 복구 오브젝트에 대한 메타데이터 저장

DB2 ACS(Advanced Copy Services)를 사용하여 작성된 복구 오브젝트에 관한 메타데이터를 저장합니다.

내장 파일

```
db2ACSApi.h
```

구문 및 데이터 구조

```
db2ACS_RC db2ACSStoreMetaData(
               db2ACS_MetaData         * pMetaData,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

pMetaData

데이터 유형: db2ACS_MetaData

db2ACS_MetaData는 스냅샷 백업 메타 데이터를 저장합니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

pMetaData의 **data** 필드에 저장되는 메타데이터는 데이터베이스 관리 프로그램 내부이며 시간에 따라 변경될 수 있으므로 DB2 ACS API 드라이버는 이 데이터를 2진 스트림으로만 취급합니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSStoreMetaData()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: db2ACS_ReturnCode *

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 23. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.
- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

스냅샷 백업 작업은 db2ACSInitialize, db2ACSBeginOperation, db2ACSPrepare 및 db2ACSSnapshot 같은 여러 DB2 ACS API 함수 호출로 구성됩니다.
db2ACSStoreMetaData는 또한 전체 조작의 파트입니다. db2ACSStoreMetaData를 포함한 이러한 모든 API 호출은 스냅샷 백업 작업이 성공하기 위해 성공해야 합니다.

db2ACSStoreMetaData가 실패하는 경우 DB2 ACS 백업 작업에 의해 생성된 복구 오브젝트는 사용 불가능합니다.

db2ACSRetrieveMetaData - DB2 ACS(Advanced Copy Services)를 사용하여 생성된 복구 오브젝트에 관한 메타데이터 검색

DB2 ACS(Advanced Copy Services)를 사용하여 작성된 복구 오브젝트에 관한 메타데이터를 검색합니다.

내장 파일

```
db2ACSApi.h
```

구문 및 데이터 구조

```
db2ACS_RC db2ACSRetrieveMetaData(
               db2ACS_MetaData         * pMetaData,
               db2ACS_ObjectID           objectID,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

매개변수

pMetaData

데이터 유형: db2ACS_MetaData

db2ACS_MetaData는 스냅샷 백업 메타 데이터를 저장합니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

pMetaData의 **data** 필드에 저장되는 메타데이터는 데이터베이스 관리 프로그램 내부이며 시간에 따라 변경될 수 있으므로 DB2 ACS API 드라이버는 이 데이터를 2진 스트림으로만 취급합니다.

objectID

데이터 유형: db2ACS_ObjectID

db2ACS_ObjectID는 스토리지 저장소에 대한 쿼리가 리턴하는 각 저장된 오브젝트에 대한 고유 ID입니다. db2ACS_ObjectID는 단일 DB2 ACS 세션의 시간 프레임 안에서만 고유하고 지속되는 것으로 보장됩니다.

데이터베이스 관리 프로그램은 db2ACSQuery()를 사용하여 db2ACSRetrieveMetaData()에 전달할 유효한 **objectID**를 얻을 수 있습니다.

pControlBlock

데이터 유형: db2ACS_CB *

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

db2ACSRetrieveMetaData()를 호출하기 전에 데이터베이스 관리 프로그램이 다음 필드를 채웁니다.

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC 데이터 유형: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode`에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 `db2ACS_ReturnCode` 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

데이터베이스 관리 프로그램이 이 매개변수에 대해 메모리를 할당하고, 해당 인스턴스화된 오브젝트에 대한 포인터를 함수로 전달합니다. 데이터베이스 관리 프로그램이 이 메모리를 비웁니다.

DB2 ACS API 드라이버가 리턴하기 전에 **pRC** 필드를 채웁니다.

리턴 코드

표 24. 리턴 코드

리턴 코드	설명	주
DB2ACS_RC_OK	조작이 성공했습니다.	
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램이 지정하는 스냅샷 백업 오브젝트를 찾을 수 없습니다.	DB2 ACS API 드라이버에서 오류가 발생했습니다. 데이터베이스 관리 프로그램이 DB2 ACS API 세션을 사용할 수 없습니다.

DB2 ACS API 드라이버에서 오류가 발생하는 경우 드라이버가 DB2 ACS 조작을 중단할 수 있습니다. 다음을 제외한 모든 조치에 대해 DB2 ACS 세션을 사용할 수 없습니다.

- db2ACSBeginQuery() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndQuery()를 호출할 수 있습니다.

- db2ACSBeginOperation() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSEndOperation()을 호출할 수 있습니다.
- db2ACSInitialize() 호출이 이전에 성공한 경우 데이터베이스 관리 프로그램이 db2ACSTerminate()를 호출할 수 있습니다.

DB2 ACS API 리턴 코드에 대한 자세한 정보는 400 페이지의 『DB2 ACS(Advanced Copy Services) API 리턴 코드』 주제를 참조하십시오.

사용 시 참고사항

없음.

DB2 ACS(Advanced Copy Services) API 데이터 구조

DB2 ACS(Advanced Copy Services) API 함수를 호출하려면 DB2 ACS API 데이터 구조를 사용해야 합니다.

db2ACS_BackupDetails DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_BackupDetails에는 스냅샷 백업 작업에 관한 정보가 들어 있습니다.

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_BackupDetails
{
   /* A traditional DB2 backup can consist of multiple objects (logical tapes),
    * where each object is uniquely numbered with a non-zero natural number.
    * ------------------------------------------------------------------------ */
   db2Uint32                  sequenceNum;

   char                       imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_BackupDetails;
```

sequenceNum

데이터 유형: db2Uint32.

고유한 번호로 백업 오브젝트를 식별합니다.

imageTimestamp

데이터 유형: char[].

길이 SQLU_TIME_STAMP_LEN + 1의 문자열.

db2ACS_CB DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_CB에는 DB2 ACS 세션을 초기화 및 종료하기 위해 필요한 기본 정보가 들어 있습니다.

```
/* ============================================================================
 * DB2 Backup Adapter Control Block
 * ============================================================================ */
typedef struct db2ACS_CB
{
   /* Output:  Handle value for this session.
```

```
   * ------------------------------------------------------------------------ */
   db2Uint32                 handle;
   db2ACS_VendorInfo         vendorInfo;

   /* Input fields and parameters.
   * ------------------------------------------------------------------------ */
   db2ACS_SessionInfo        session;
   db2ACS_Options            options;

   /* Operation info is optional, possibly NULL, and is only ever valid
   * within the context of an operation (from call to BeginOperation() until
   * the EndOperation() call returns).
   *
   * The operation info will be present during creation or read operations
   * of snapshot and backup objects.
   * ------------------------------------------------------------------------ */
   db2ACS_OperationInfo     * operation;
} db2ACS_CB;
```

handle

데이터 유형: db2Uint32.

DB2 ACS 세션을 참조하는 핸들.

vendorInfo

데이터 유형: db2ACS_VendorInfo.

db2ACS_VendorInfo에는 DB2 ACS API 드라이버에 대한 정보가 들어 있습니다.

session

데이터 유형: db2ACS_SessionInfo.

db2ACS_SessionInfo에는 DB2 ACS 세션에 관한 모든 정보가 들어 있습니다.

options

데이터 유형: db2ACS_Options.

db2ACS_Options는 DB2 ACS 조작에 사용될 옵션을 지정합니다. 이 문자열의 컨텐츠는 DB2 ACS API 드라이버에 특정합니다.

operation

데이터 유형: db2ACS_OperationInfo *.

db2ACS_OperationInfo에는 스냅샷 백업 작업에 관한 정보가 들어 있습니다.

db2ACS_CreateObjectInfo DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_CreateObjectInfo에는 DB2 ACS 백업 오브젝트 작성에 관한 정보가 들어 있습니다.

```
/* ============================================================================
 * Object Creation Parameters.
 * ============================================================================ */
typedef struct db2ACS_CreateObjectInfo
{
```

```
   db2ACS_ObjectInfo          object;
   db2ACS_DB2ID               db2ID;

   /* ------------------------------------------------------------------------
    * The following fields are optional information for the database manager
    * to use as it sees fit.
    * ------------------------------------------------------------------------ */

   /* Historically both the size estimate and management
    * class parameters have been used by the TSM client API for traditional
    * backup objects, log archives, and load copies, but not for snapshot
    * backups.
    * ------------------------------------------------------------------------ */
   db2Uint64 sizeEstimate;
   char                       mgmtClass[DB2ACS_MAX_MGMTCLASS_SZ + 1];

   /* The appOptions is a copy of the iOptions field of flags passed to DB2's
    * db2Backup() API when this execution was initiated.  This field will
    * only contain valid data when creating a backup or snapshot object.
    * ------------------------------------------------------------------------ */
   db2Uint32                  appOptions;
} db2ACS_CreateObjectInfo;
```

object 데이터 유형: db2ACS_ObjectInfo

db2ACS_ObjectInfo에는 DB2 ACS(Advanced Copy Services) API를 사용하여 작성된 오브젝트에 관한 정보가 들어 있습니다.

db2ID 데이터 유형: db2ACS_DB2ID

db2ACS_DB2ID는 IBM Data Server를 식별합니다.

sizeEstimate

데이터 유형: db2Uint64.

작성되는 백업 오브젝트의 크기 추정입니다. 이 추정은 로그 아카이브, 로드 사본 또는 스냅샷 백업 오브젝트에 적용되지 않습니다.

mgmtClass

데이터 유형: db2ACS_MgmtClass.

길이 db2ACS_MAX_MGMTCLASS_SZ + 1의 문자열.

이것은 스냅샷 백업 오브젝트에 적용되지 않습니다.

appOptions

데이터 유형: db2Uint32.

스냅샷 백업을 시작한 백업 명령에 전달된 백업 옵션의 사본.

db2ACS_DB2ID DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_DB2ID는 IBM Data Server를 식별합니다.

```
/* ============================================================================
 * DB2 Data Server Identifier
 * ============================================================================ */
typedef struct db2ACS_DB2ID
{
   db2Uint32 version;
```

```
   db2Uint32                 release;
   db2Uint32                 level;
   char                      signature[DB2ACS_SIGNATURE_SZ + 1];
} db2ACS_DB2ID;
```

버전 데이터 유형: db2Uint32.

IBM Data Server의 버전입니다. 예: 9

release

데이터 유형: db2Uint32.

IBM Data Server의 릴리스 레벨입니다. 예: 5

level 데이터 유형: db2Uint32.

IBM Data Server의 레벨 ID입니다. 예: 0

signature

데이터 유형: char[].

길이 DB2ACS_SIGNATURE_SZ + 1의 문자열입니다. 예: ″SQL09050″

db2ACS_GroupList DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_GroupList에는 스냅샷 백업 작업에 포함될 그룹의 목록이 들어 있습니다.

```
/* ============================================================================
 * Snapshot Group List
 *
 * This is an array of size 'numGroupIDs', indicating the set of groups that
 * are to be included in the snapshot operation.
 * ============================================================================ */
typedef struct db2ACS_GroupList
{
   db2Uint32                 numGroupIDs;
   db2Uint32               * id;
} db2ACS_GroupList;
```

numGroupIDs

데이터 유형: db2Uint32.

배열 **id**에 있는 그룹 수.

id 데이터 유형: db2Uint32 *.

그룹 ID의 배열입니다. 식별되는 그룹은 스냅샷 백업 작업에 포함될 그룹(또는 그룹 목록)입니다.

db2ACS_LoadcopyDetails DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_LoadcopyDetails에는 로드 사본 조작에 대한 정보가 들어 있습니다.

```
/* ---------------------------------------------------------------------- */
typedef struct db2ACS_LoadcopyDetails
{
   /* Just like the BackupDetails, a DB2 load copy can consist of multiple
    * objects (logical tapes), where each object is uniquely numbered with a
```

```
    * non-zero natural number.
    * ------------------------------------------------------------------------ */
   db2Uint32                  sequenceNum;

   char                       imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_LoadcopyDetails;
```

sequenceNum

데이터 유형: db2Uint32.

고유한 번호로 백업 오브젝트를 식별합니다.

imageTimestamp

데이터 유형: char[].

길이 SQLU_TIME_STAMP_LEN + 1의 문자열

db2ACS_LogDetails DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_LogDetails에는 특정 데이터베이스 로그 파일을 식별하는 정보가 들어 있습니다.

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_LogDetails
{
   db2Uint32                  fileID;
   db2Uint32                  chainID;
} db2ACS_LogDetails;
```

fileID 데이터 유형: db2Uint32.

데이터베이스 로그 파일의 파일 이름인 번호입니다.

chainID

데이터 유형: db2Uint32.

데이터베이스 로그 파일 **fileID**가 속하는 데이터베이스 로그 파일 체인을 식별하는 번호.

db2ACS_ObjectInfo DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_ObjectInfo에는 DB2 ACS(Advanced Copy Services) API를 사용하여 작성된 오브젝트에 관한 정보가 들어 있습니다.

```
/* ========================================================================
 * Object Description and Associated Information.
 *
 * This structure is used for both input and output, and its contents define
 * the minimum information that must be recorded about any object created
 * through this interface.
 * ======================================================================== */
typedef struct db2ACS_ObjectInfo
{
   db2ACS_ObjectType          type;
   SQL_PDB_NODE_TYPE dbPartitionNum;

   char                       db[SQL_DBNAME_SZ + 1];
```

```
   char                      instance[DB2ACS_MAX_OWNER_SZ + 1];
   char                      host[SQL_HOSTNAME_SZ + 1];
   char                      owner[DB2ACS_MAX_OWNER_SZ + 1];

   union
   {
      db2ACS_BackupDetails    backup;
      db2ACS_LogDetails       log;
      db2ACS_LoadcopyDetails  loadcopy;
      db2ACS_SnapshotDetails  snapshot;
   } details;
} db2ACS_ObjectInfo;
```

type 데이터 유형: db2ACS_ObjectType.

스냅샷 백업 오브젝트 유형을 지정합니다. 값:

DB2ACS_OBJTYPE_ALL
DB2ACS_OBJTYPE_BACKUP
DB2ACS_OBJTYPE_LOG
DB2ACS_OBJTYPE_LOADCOPY
DB2ACS_OBJTYPE_SNAPSHOT

DB2ACS_OBJTYPE_ALL은 쿼리에 대한 필터로서만 사용할 수 있습니다. 유형 0의 오브젝트는 없습니다.

dbPartitionNum

데이터 유형: SQL_PDB_NODE_TYPE.

이 데이터베이스 파티션에 대한 ID.

db 데이터 유형: char[].

길이 SQL_DBNAME_SZ + 1의 문자열.

instance

데이터 유형: char[].

길이 DB2ACS_MAX_OWNER_SZ + 1의 문자열.

host 데이터 유형: char[].

길이 SQL_HOSTNAME_SZ + 1의 문자열.

owner 데이터 유형: char[].

길이 DB2ACS_MAX_OWNER_SZ + 1의 문자열.

details

backup

데이터 유형: db2ACS_BackupDetails

db2ACS_BackupDetails에는 스냅샷 백업 작업에 관한 정보가 들어 있습니다.

log 데이터 유형: db2ACS_LogDetails

db2ACS_LogDetails에는 특정 데이터베이스 로그 파일을 식별하는 정보가 들어 있습니다.

loadcopy

데이터 유형: db2ACS_LoadcopyDetails

db2ACS_LoadcopyDetails에는 로드 사본 조작에 대한 정보가 들어 있습니다.

snapshot

데이터 유형: db2ACS_SnapshotDetails

db2ACS_SnapshotDetails에는 스냅샷 백업 작업에 관한 정보가 들어 있습니다.

db2ACS_ObjectStatus DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_ObjectStatus에는 스냅샷 백업 작업의 상태나 진행 또는 스냅샷 백업 오브젝트의 상태나 사용 편리성에 관한 정보가 들어 있습니다.

```
typedef struct db2ACS_ObjectStatus
{
   /* The total and completed bytes refer only to the ACS snapshot backup
    * itself, not to the progress of any offloaded tape backup.
    *
    * A bytesTotal of 0 indicates that the progress could not be determined.
    * ------------------------------------------------------------------------ */
   db2Uint64                  bytesCompleted;
   db2Uint64                  bytesTotal;
   db2ACS_ProgressState       progressState;
   db2ACS_UsabilityState      usabilityState;
} db2ACS_ObjectStatus;
```

bytesCompleted

데이터 유형: db2Uint64.

완료된 스냅샷 백업의 양(바이트)입니다.

bytesTotal

데이터 유형: db2Uint64.

완료된 스냅샷 백업의 크기(바이트)입니다.

progressState

데이터 유형: db2ACS_ProgressState.

스냅샷 백업 작업의 상태. 값:

DB2ACS_PSTATE_UNKNOWN
DB2ACS_PSTATE_IN_PROGRESS
DB2ACS_PSTATE_SUCCESSFUL
DB2ACS_PSTATE_FAILED

usabilityState

데이터 유형: db2ACS_UsabilityState.

스냅샷 백업 오브젝트의 상태, 스냅샷 백업 오브젝트를 사용할 수 있는 방법.
값:

DB2ACS_USTATE_UNKNOWN
DB2ACS_USTATE_LOCALLY_MOUNTABLE
DB2ACS_USTATE_REMOTELY_MOUNTABLE
DB2ACS_USTATE_REPETITIVELY_RESTORABLE
DB2ACS_USTATE_DESTRUCTIVELY_RESTORABLE
DB2ACS_USTATE_SWAP_RESTORABLE
DB2ACS_USTATE_PHYSICAL_PROTECTION
DB2ACS_USTATE_FULL_COPY
DB2ACS_USTATE_DELETED
DB2ACS_USTATE_FORCED_MOUNT
DB2ACS_USTATE_BACKGROUND_MONITOR_PENDING
DB2ACS_USTATE_TAPE_BACKUP_PENDING
DB2ACS_USTATE_TAPE_BACKUP_IN_PROGRESS
DB2ACS_USTATE_TAPE_BACKUP_COMPLETE

db2ACS_OperationInfo DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_OperationInfo에는 스냅샷 백업 작업에 관한 정보가 들어 있습니다.

```
/* ========================================================================
 * Operation Info
 *
 * The information contained within this structure is only valid within the
 * context of a particular operation.  It will be valid at the time
 * BeginOperation() is called, and will remain unchanged until EndOperation()
 * returns, but must not be referenced outside the scope of an operation.
 * ======================================================================== */
typedef struct db2ACS_OperationInfo
{
   db2ACS_SyncMode            syncMode;

   /* List of database and backup operation partitions.
    *
    * For details, refer to the db2ACS_PartitionList definition.
    * ------------------------------------------------------------------- */
   db2ACS_PartitionList     * dbPartitionList;
} db2ACS_OperationInfo;
```

syncMode

데이터 유형: db2ACS_SyncMode.

개별 데이터베이스 파티션에 대한 백업 작업 사이의 동기화 레벨.

값:

DB2ACS_SYNC_NONE

다중 데이터베이스 파티션에 대한 관련 조작 사이에 동기화가 없습니다. 다중 데이터베이스 파티션 사이에 어떤 동기화도 사용하지 않는 조작 중에 사용됩니다.

DB2ACS_SYNC_SERIAL

다중 데이터베이스 파티션에 대한 동시 스냅샷 백업 작업을 수행할 때 사용됩니다. 각 데이터베이스 파티션은 스냅샷 백업 작업이 실행될 때 입력 및 출력(IO)을 일시중단하며, 데이터베이스 파티션에 대한 IO는 동시가 아니라 일렬로 재개됩니다.

SYNC_PARALLEL

다중 파티션에 대해 스냅샷 조작을 동시에 수행합니다. 스냅샷 백업 작업에 포함되는 모든 데이터베이스 파티션이 스냅샷 백업 작업에 대한 준비를 완료한 후에 입력 및 출력(IO)이 모든 데이터베이스 파티션에서 일시중단됩니다. 나머지 스냅샷 백업 단계는 관련된 모든 데이터베이스 파티션에서 동시에 발생합니다.

dbPartitionList

데이터 유형: db2ACS_PartitionList *.

db2ACS_PartitionList에는 데이터베이스에 있는 데이터베이스 파티션 및 DB2 ACS 조작에 관련되는 데이터베이스 파티션에 관한 정보가 들어 있습니다.

db2ACS_Options DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_Options는 DB2 ACS 조작에 사용될 옵션을 지정합니다. 이 문자열의 컨텐츠는 DB2 ACS API 드라이버에 특정합니다.

```
/* ****************************************************************************
 * DB2 Backup Adapter User Options
 * ************************************************************************** */
typedef struct db2ACS_Options
{
   db2Uint32 size;
   void                    * data;
} db2ACS_Options;
```

size 데이터 유형: db2Uint32.

data의 크기(바이트).

data 데이터 유형: void *.

옵션을 포함하는 메모리 블록에 대한 포인터입니다.

db2ACS_PartitionEntry DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_PartitionEntry는 db2ACS_PartitionList의 요소입니다.

```
typedef struct db2ACS_PartitionEntry
{
   SQL_PDB_NODE_TYPE          num;
   char                       host[SQL_HOSTNAME_SZ + 1];
} db2ACS_PartitionEntry;
```

num 데이터 유형: SQL_PDB_NODE_TYPE.

데이터베이스 파티션 항목에 대한 ID.

host 데이터 유형: char[].

길이 SQL_HOSTNAME_SZ + 1의 문자열.

db2ACS_PartitionList DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_PartitionList에는 데이터베이스에 있는 데이터베이스 파티션 및 DB2 ACS 조작에 관련되는 데이터베이스 파티션에 관한 정보가 들어 있습니다.

```
typedef struct db2ACS_PartitionList
{
   db2Uint64                  numPartsInDB;
   db2Uint64                  numPartsInOperation;
   db2ACS_PartitionEntry    * partition;
} db2ACS_PartitionList;
```

numPartsInDB

데이터 유형: db2Uint64.

데이터베이스에 있는 데이터베이스 파티션 수.

numPartsInOperation

데이터 유형: db2Uint64.

DB2 ACS 조작에 관련되는 데이터베이스 파티션 수.

partition

데이터 유형: db2ACS_PartitionEntry *.

db2ACS_PartitionEntry는 db2ACS_PartitionList의 요소입니다.

db2ACS_PathEntry DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_PathEntry에는 데이터베이스 경로에 관한 정보가 들어 있습니다.

```
typedef struct db2ACS_PathEntry
{
   /* INPUT: The path and type will be provided by the database server, as well
    *        as a flag indicating if the path is to be excluded from the backup.
    * ------------------------------------------------------------------------ */
   char                       path[DB2ACS_MAX_PATH_SZ + 1];
   db2ACS_PathType            type;
   db2Uint32                  toBeExcluded;

   /* OUTPUT: The group ID is to be provided by the backup adapter for use by
    *         the DB2 server.  The group ID will be used during with snapshot
```

```
    *         operations as an indication of which paths are dependent and must
    *         be included together in any snapshot operation.  Unique group IDs
    *         indicate that the paths in those groups are independent for the
    *         purposes of snapshot operations.
    * ------------------------------------------------------------------------ */
   db2Uint32                 groupID;
} db2ACS_PathEntry;
```

path 데이터 유형: char[].

길이 DB2ACS_MAX_PATH_SZ + 1의 문자열.

type 데이터 유형: db2ACS_PathType.

경로의 유형입니다. 값:

DB2ACS_PATH_TYPE_UNKNOWN
DB2ACS_PATH_TYPE_LOCAL_DB_DIRECTORY
DB2ACS_PATH_TYPE_DBPATH
DB2ACS_PATH_TYPE_DB_STORAGE_PATH
DB2ACS_PATH_TYPE_TBSP_CONTAINER
DB2ACS_PATH_TYPE_TBSP_DIRECTORY
DB2ACS_PATH_TYPE_TBSP_DEVICE
DB2ACS_PATH_TYPE_LOGPATH
DB2ACS_PATH_TYPE_MIRRORLOGPATH

toBeExcluded

데이터 유형: db2Uint32.

주어진 경로를 스냅샷 백업에 포함할지 여부를 표시하는 플래그. 값:

- 0 - 스냅샷 백업에 경로 포함
- 1 - 스냅샷 백업에 경로를 포함하지 않음

groupID

데이터 유형: db2Uint32.

그룹 ID.

db2ACS_PathList DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_PathList에는 DB2 ACS 조작에 특정한 각 경로에 관한 일부 추가 정보를 포함하여 데이터베이스 경로의 목록이 들어 있습니다.

```
/* ============================================================================
 * Snapshot File List
 *
 * This is an array of 'numEntries' db2ACS_PathEntry's, where each path entry is
 * a path to some storage on the DB2 server which is in use by the current
 * database.
 * ========================================================================== */

typedef struct db2ACS_PathList
```

```
{
   db2Uint32                 numEntries;
   db2ACS_PathEntry         * entry;
} db2ACS_PathList;
```

numEntries

데이터 유형: db2Uint32.

entry 배열에 있는 경로 항목 수.

entry 데이터 유형: db2ACS_PathEntry.

db2ACS_PathEntry에는 데이터베이스 경로에 관한 정보가 들어 있습니다.

db2ACS_PostObjectInfo DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_DB2ID는 스냅샷 백업 오브젝트 작성 시간에는 알려질 수 없지만 오브젝트 저장소에 유지보수되어야 하는 데이터 세트입니다.

```
/* ============================================================================
 * The PostObjectInfo is a set of data that can not be known at object
 * creation time, but which must be maintained in the object repository.  This
 * is an optional field on the Verify() call, which may be NULL if there are
 * no post-operation updates to be made.
 * ============================================================================ */
typedef struct db2ACS_PostObjectInfo
{
   /* The first active log will only be valid when creating a backup or
    * snapshot object.  It will indicate the file number and chain id of the
    * first log required for recovery using this object.
    * ------------------------------------------------------------------------ */
   db2ACS_LogDetails          firstActiveLog;
} db2ACS_PostObjectInfo;
```

firstActiveLog

데이터 유형: db2ACS_LogDetails.

db2ACS_LogDetails에는 특정 데이터베이스 로그 파일을 식별하는 정보가 들어 있습니다.

db2ACS_QueryInput 및 db2ACS_QueryOutput DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_QueryInput에는 쿼리 중인 오브젝트에 대한 식별 정보가 들어 있습니다. db2ACS_QueryOutput에는 스냅샷 백업 오브젝트에 관한 쿼리 결과 정보가 들어 있습니다.

```
/* ============================================================================
 * Unique Querying.
 *
 * When using this structure as query input, to indicate the
 * intention to supply a 'wildcard' search criteria, DB2 will supply:
 *
 *    -- character strings as "*".
 *    -- numeric values as (-1), cast as the appropriate signed or unsigned
 *       type.
 * ============================================================================ */

typedef struct db2ACS_ObjectInfo db2ACS_QueryInput;
```

```
typedef struct db2ACS_QueryOutput
{
   db2ACS_ObjectID            objectID;
   db2ACS_ObjectInfo          object;
   db2ACS_PostObjectInfo      postInfo;
   db2ACS_DB2ID               db2ID;
   db2ACS_ObjectStatus        status;

   /* Size of the object in bytes.
    * ------------------------------------------------------------------------  */
   db2Uint64                  objectSize;

   /* Size of the metadata associated with the object, if any, in bytes.
    * ------------------------------------------------------------------------  */
   db2Uint64                  metaDataSize;

   /* The creation time of the object is a 64bit value with a definition
    * equivalent to an ANSI C time_t value (seconds since the epoch, GMT).
    *
    * This field is equivalent to the file creation or modification time in
    * a traditional filesystem.  This should be created and stored
    * automatically by the BA subsystem, and a valid time value should be
    * returned with object query results, for all object types.
    * ------------------------------------------------------------------------  */
   db2Uint64                  createTime;
} db2ACS_QueryOutput;
```

objectID

데이터 유형: db2ACS_ObjectID.

db2ACS_ObjectID는 스토리지 저장소에 대한 쿼리가 리턴하는 각 저장된 오브젝트에 대한 고유 ID입니다. db2ACS_ObjectID는 단일 DB2 ACS 세션의 시간 프레임 안에서만 고유하고 지속되는 것으로 보장됩니다.

object 데이터 유형: db2ACS_ObjectInfo

db2ACS_ObjectInfo에는 DB2 ACS(Advanced Copy Services) API를 사용하여 작성된 오브젝트에 관한 정보가 들어 있습니다.

postInfo

데이터 유형: db2ACS_PostObjectInfo.

db2ACS_DB2ID는 스냅샷 백업 오브젝트 작성 시간에는 알려질 수 없지만 오브젝트 저장소에 유지보수되어야 하는 데이터 세트입니다.

db2ID 데이터 유형: db2ACS_DB2ID.

db2ACS_DB2ID는 IBM Data Server를 식별합니다.

status 데이터 유형: db2ACS_ObjectStatus.

db2ACS_ObjectStatus에는 스냅샷 백업 작업의 상태나 진행 또는 스냅샷 백업 오브젝트의 상태나 사용 편리성에 관한 정보가 들어 있습니다.

objectSize

데이터 유형: db2Uint64.

오브젝트의 크기(바이트).

metaDataSize

데이터 유형: db2Uint64.

오브젝트와 연관된 메타데이터(있는 경우)의 크기(바이트).

createTime

데이터 유형: db2Uint64.

오브젝트의 작성 시간입니다. **createTime**의 값은 ANSI C time_t 값과 동등합니다.

db2ACS_ReadList DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_ReadList에는 그룹 목록이 들어 있습니다.

```
/* The ReadList will only be used for snapshots where the action is READ, and
 * where one of the granularity modifiers other than BY_OBJ has been specified.
 * In the typical usage scenario of ( READ | BY_OBJ ) the ReadList parameter
 * should be ignored.
 *
 * When the action is DB2ACS_ACTION_BY_GROUP the union is to be interpreted
 * as a group list.
 * ------------------------------------------------------------------------- */
typedef union db2ACS_ReadList
{
   db2ACS_GroupList           group;
} db2ACS_ReadList;
```

group 데이터 유형: db2ACS_GroupList.

db2ACS_GroupList에는 스냅샷 백업 작업에 포함될 그룹의 목록이 들어 있습니다.

db2ACS_ReturnCode DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_ReturnCode에는 스토리지 하드웨어에 특정한 메시지 텍스트 및 오류 코드를 포함한 진단 정보가 들어 있습니다. DB2 ACS API 함수에 대한 db2ACS_ReturnCode 매개변수의 컨텐츠가 데이터베이스 관리 프로그램 진단 로그에 기록됩니다.

```
/* ========================================================================
 * Storage Adapter Return Code and Diagnostic Data.
 *
 * These will be recorded in the DB2 diagnostic logs, but are intended to be
 * internal return and reason codes from the storage layers which can be used
 * in conjunction with the DB2ACS_RC to provide more detailed diagnostic info.
 * ======================================================================== */
typedef struct db2ACS_ReturnCode
{
   int                        returnCode;
   int                        reasonCode;
   char                       description[DB2ACS_MAX_COMMENT_SZ + 1];
} db2ACS_ReturnCode;
```

returnCode

데이터 유형: int.

스토리지 하드웨어에 특정한 리턴 코드.

reasonCode

데이터 유형: int.

스토리지 하드웨어에 특정한 이유 코드.

description

데이터 유형: char[].

길이 DB2ACS_MAX_COMMENT_SZ + 1의 문자열.

db2ACS_SessionInfo DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_SessionInfo에는 DB2 ACS 세션에 관한 모든 정보가 들어 있습니다.

```
/* ============================================================================
 * Session Info
 * ============================================================================ */
typedef struct db2ACS_SessionInfo
{
   db2ACS_DB2ID               db2ID;

   /* Fields identifying the backup session originator.
    * ------------------------------------------------------------------------ */
   SQL_PDB_NODE_TYPE dbPartitionNum;
   char                       db[SQL_DBNAME_SZ + 1];
   char                       instance[DB2ACS_MAX_OWNER_SZ + 1];
   char                       host[SQL_HOSTNAME_SZ + 1];
   char                       user[DB2ACS_MAX_OWNER_SZ + 1];
   char                       password[DB2ACS_MAX_PASSWORD_SZ + 1];

   /* The fully qualified ACS vendor library name to be used.
    * ------------------------------------------------------------------------ */
   char                       libraryName[DB2ACS_MAX_PATH_SZ + 1];
} db2ACS_SessionInfo;
```

db2ID 데이터 유형: db2ACS_DB2ID

db2ACS_DB2ID는 IBM Data Server를 식별합니다.

dbPartitionNum

데이터 유형: SQL_PDB_NODE_TYPE

데이터베이스 파티션에 대한 고유한 숫자 ID.

db 데이터 유형: char[].

길이 SQL_DBNAME_SZ + 1의 문자열.

instance

데이터 유형: char[].

길이 DB2ACS_MAX_OWNER_SZ + 1의 문자열.

host 데이터 유형: char[].

길이 SQL_HOSTNAME_SZ + 1의 문자열.

user 데이터 유형: char[].

길이 DB2ACS_MAX_OWNER_SZ + 1의 문자열.

password

데이터 유형: char[].

길이 DB2ACS_MAX_PASSWORD_SZ + 1의 문자열.

libraryName

데이터 유형: char[].

길이 DB2ACS_MAX_PATH_SZ + 1의 문자열.

db2ACS_SnapshotDetails DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_SnapshotDetails에는 스냅샷 백업 작업에 관한 정보가 들어 있습니다.

```
typedef struct db2ACS_SnapshotDetails
{
   char                         imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_SnapshotDetails;
```

imageTimestamp

데이터 유형: char[].

길이 SQLU_TIME_STAMP_LEN + 1의 문자열.

db2ACS_MetaData DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_MetaData는 스냅샷 백업 메타 데이터를 저장합니다.

```
/* ****************************************************************************
 * The metadata structure itself is internal to DB2 and is to be treated by
 * the storage interface as an unstructured block of data of the given size.
 * **************************************************************************** */
typedef struct db2ACS_MetaData
{
   db2Uint64                  size;
   void                     * data;
} db2ACS_MetaData;
```

size 데이터 유형: db2Uint32.

data의 *크기*(바이트).

data 데이터 유형: void *.

데이터베이스 관리 프로그램이 스냅샷 백업 메타데이터를 저장하는 데 사용하는 메모리 블록에 대한 포인터.

db2ACS_VendorInfo DB2 ACS(Advanced Copy Services) API 데이터 구조

db2ACS_VendorInfo에는 DB2 ACS API 드라이버에 대한 정보가 들어 있습니다.

```
/* ============================================================================
 * Storage Vendor Identifier
 * ============================================================================ */
typedef struct db2ACS_VendorInfo
{
   void                    * vendorCB;             /* Vendor control block  */
   db2Uint32                 version;              /* Current version       */
   db2Uint32                 release;              /* Current release       */
   db2Uint32                 level;                /* Current level         */
   char                      signature[DB2ACS_MAX_VENDORID_SZ + 1];
} db2ACS_VendorInfo;
```

vendorCB

데이터 유형: void *.

DB2 ACS API 드라이버에 특정한 제어 블록에 대한 포인터입니다.

버전 데이터 유형: db2Uint32.

DB2 ACS API 드라이버의 버전.

release

데이터 유형: db2Uint32.

DB2 ACS API 드라이버의 릴리스 레벨.

level 데이터 유형: db2Uint32.

DB2 ACS API 드라이버의 레벨 ID.

signature

데이터 유형: db2ACS_VendorSignature.

길이 DB2ACS_MAX_VENDORID_SZ + 1의 문자열.

DB2 ACS(Advanced Copy Services) 우수 사례

DB2 ACS(Advanced Copy Services)를 설치 및 구성할 때 다음 우수 사례를 고려하십시오.

로그 경로로 전용 볼륨 그룹 지정

데이터베이스 디렉토리 및 데이터베이스 컨테이너와 독립적인 자체 소유 스냅샷 볼륨 내에 로그 경로를 포함할 것을 권장합니다.

데이터베이스 파티션마다 하나의 볼륨 그룹 지정

데이터베이스 파티셔닝 기능(DPF) 환경에서, 각 데이터베이스 파티션은 다른 데이터베이스 파티션과 독립적인 스냅샷 볼륨 세트에 있어야 합니다.

DB2 ACS(Advanced Copy Services) 제한사항

DB2 ACS(Advanced Copy Services)를 설치 및 구성할 때 고려할 제한사항이 있습니다.

볼륨 공유는 지원되지 않습니다. 데이터베이스 파티션이 다른 데이터베이스 파티션과 같은 저장 볼륨에 있는 경우 스냅샷 조작은 허용되지 않습니다.

DB2 ACS(Advanced Copy Services) API 리턴 코드

DB2 ACS(Advanced Copy Services) API 함수는 가능한 리턴 코드의 정의된 세트를 리턴합니다.

표 25. DB2 ACS(Advanced Copy Services) API 리턴 코드

리턴 코드	설명
DB2ACS_RC_OK	조작이 성공했습니다.
DB2ACS_RC_LINK_EXIST	세션이 이전에 활성화되었습니다.
DB2ACS_RC_COMM_ERROR	테이프 드라이브 같은 스토리지 디바이스에 통신 오류가 있었습니다.
DB2ACS_RC_INV_VERSION	데이터베이스 관리 프로그램의 DB2 ACS 라이브러리의 버전과 DB2 ACS API 드라이버의 버전이 호환 가능하지 않습니다.
DB2ACS_RC_INV_ACTION	데이터베이스 관리 프로그램이 유효하지 않은 DB2 ACS API 드라이버로부터 조치를 요청했습니다.
DB2ACS_RC_NO_DEV_AVAIL	현재 사용할 수 있는 테이프 드라이브 같은 스토리지 디바이스가 없습니다.
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램이 지정하는 스냅샷 백업 오브젝트를 찾을 수 없습니다.
DB2ACS_RC_OBJS_FOUND	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램이 제공하는 스펙과 일치하는 둘 이상의 스냅샷 백업 오브젝트를 발견했습니다.
DB2ACS_RC_INV_USERID	데이터베이스 관리 프로그램이 DB2 ACS API 드라이버로 유효하지 않은 사용자 ID를 전달했습니다.
DB2ACS_RC_INV_PASSWORD	데이터베이스 관리 프로그램이 DB2 ACS API 드라이버로 유효하지 않은 암호를 전달했습니다.
DB2ACS_RC_INV_OPTIONS	데이터베이스 관리 프로그램이 유효하지 않은 옵션을 지정했습니다.
DB2ACS_RC_INIT_FAILED	데이터베이스 관리 프로그램이 DB2 ACS 세션을 초기화하려고 했지만 초기화에 실패했습니다.
DB2ACS_RC_INV_DEV_HANDLE	데이터베이스 관리 프로그램이 유효하지 않은 스토리지 디바이스 핸들을 전달했습니다.
DB2ACS_RC_BUFF_SIZE	데이터베이스 관리 프로그램이 유효하지 않은 버퍼 크기를 지정했습니다.
DB2ACS_RC_END_OF_DATA	DB2 ACS API 드라이버가 추가 스냅샷 백업 오브젝트를 찾을 수 없습니다.
DB2ACS_RC_END_OF_TAPE	스토리지 디바이스가 예상치 않게 테이프 백업 미디어의 끝에 도달했습니다.
DB2ACS_RC_DATA_RESEND	테이프 드라이브 같은 스토리지 디바이스가 데이터베이스 관리 프로그램이 가장 최근의 데이터 버퍼를 재전송하도록 요청했습니다.
DB2ACS_RC_COMMIT_FAILED	DB2 ACS API 드라이버가 트랜잭션을 커미트할 수 없습니다.
DB2ACS_RC_DEV_ERROR	테이프 드라이브 같은 스토리지 디바이스에서 오류가 발생했습니다.
DB2ACS_RC_WARNING	스토리지 하드웨어가 경고를 리턴했습니다. 데이터베이스 관리 프로그램 진단 로그에서 자세한 정보를 찾으십시오.
DB2ACS_RC_LINK_NOT_EXIST	세션이 이전에 활성화되지 않았습니다.
DB2ACS_RC_MORE_DATA	스토리지 위치에서 데이터베이스 관리 프로그램으로 전송될 추가 데이터가 있습니다.

표 25. DB2 ACS(Advanced Copy Services) API 리턴 코드 (계속)

리턴 코드	설명
DB2ACS_RC_ENDOFMEDIA_NO_DATA	스토리지 디바이스가 어떤 데이터도 찾지 않고 스토리지 미디어의 끝에 도달했습니다.
DB2ACS_RC_ENDOFMEDIA	스토리지 디바이스가 스토리지 미디어의 끝에 도달했습니다.
DB2ACS_RC_MAX_LINK_GRANT	최대 링크 수가 설정되었습니다. 데이터베이스 관리 프로그램이 추가 링크를 설정할 수 없습니다.
DB2ACS_RC_IO_ERROR	DB2 ACS API 드라이버에서 입력 또는 출력 조작에서 생성된 오류가 발생했습니다.
DB2ACS_RC_DELETE_FAILED	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램에 의해 지정된 스냅샷 백업 오브젝트를 성공적으로 삭제할 수 없습니다.
DB2ACS_RC_INV_BKUP_FNAME	데이터베이스 관리 프로그램이 스냅샷 백업 오브젝트에 대해 유효하지 않은 파일 이름을 지정했습니다.
DB2ACS_RC_NOT_ENOUGH_SPACE	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램에 의해 지정되는 데이터베이스의 스냅샷 백업을 수행하기에 충분한 스토리지 스페이스가 없다고 추정했습니다.
DB2ACS_RC_ABORT_FAILED	데이터베이스 관리 프로그램이 DB2 ACS 조작을 중단하려고 했지만 중단 시도가 실패했습니다.
DB2ACS_RC_UNEXPECTED_ERROR	DB2 ACS API 드라이버에서 심각하고 알 수 없는 오류가 발생했습니다.
DB2ACS_RC_NO_DATA	DB2 ACS API 드라이버가 데이터베이스 관리 프로그램으로 데이터를 리턴하지 않았습니다.
DB2ACS_RC_OBJ_OUT_OF_SCOPE	데이터베이스 관리 프로그램이 DB2 ACS API 드라이버에 의해 관리되지 않는 복구 오브젝트에 대해 DB2 ACS 조작을 수행하려고 시도했습니다.
DB2ACS_RC_INV_CALL_SEQUENCE	데이터베이스 관리 프로그램이 유효하지 않은 순서로 DB2 ACS API 함수를 호출했습니다. 예를 들어 데이터베이스 관리 프로그램은 db2ACSQueryAPIVersion을 제외한 다른 모든 DB2 ACS API 함수를 호출하기 전에 db2ACSInitialize를 호출해야 합니다.
DB2ACS_RC_SHARED_STORAGE_GROUP	데이터베이스 관리 프로그램이 다른 데이터베이스나 응용프로그램이 사용 중인 스토리지 오브젝트에 대해 스냅샷 조작을 수행하려고 시도했습니다.

DB2 ACS(Advanced Copy Services) 지원 운영 체제 및 하드웨어

IBM Data Server가 지원하는 운영 체제 및 하드웨어의 서브세트를 지원하는 DB2 ACS API 드라이버가 IBM Data Server에 통합됩니다.

표 26. DB2 ACS(Advanced Copy Services) API 지원 운영 체제 및 하드웨어

하드웨어	**SAN(Storage Area Network)** 스토리지가 있는 **AIX** 운영 체제 [2]	**NFS(Network File System)** 스토리지가 있는 **AIX** 운영 체제 [2]	**NFS(Network File System)** 스토리지가 있는 **Linux** 운영 체제 [1]
IBM TotalStorage SAN Volume Controller	전체 지원.	지원되지 않음.	지원되지 않음.
IBM System Storage DS6000	다음을 제외한 전체 지원: • 증분식 복사는 지원되지 않음	지원되지 않음.	지원되지 않음.

표 26. DB2 ACS(Advanced Copy Services) API 지원 운영 체제 및 하드웨어 (계속)

하드웨어	**SAN(Storage Area Network) 스토리지가 있는 AIX 운영 체제** [2]	**NFS(Network File System) 스토리지가 있는 AIX 운영 체제** [2] [1]	**NFS(Network File System) 스토리지가 있는 Linux 운영 체제** [1]
IBM System Storage DS8000	다음을 제외한 전체 지원: • 증분식 복사는 지원되지 않음	지원되지 않음.	지원되지 않음.
IBM System Storage N Series	전체 지원.	전체 지원.	전체 지원.
NetApp V-series	전체 지원.	전체 지원.	전체 지원.
NetApp FAS 시리즈	전체 지원	전체 지원	전체 지원

[1] 다음 시스템만 DB2 ACS 및 Linux에서 지원됩니다.

- x86(Intel Pentium®, Intel Xeon® 및 AMD) 프로세서에서 64비트만
- POWER®(IBM eServer™ OpenPower®, System i 또는 Linux를 지원하는 pSeries 시스템)

[2] DB2 ACS on AIX에는 AIX 5.3만 지원됩니다. . AIX 6.1 운영 체제의 경우 수동 복사를 수행할 수 있습니다. 이에 대한 한 예는 쓰기 입출력을 일시중단하고 236 페이지의 『분할 미러를 백업 이미지로 사용』에 설명된 대로 분할 미러 기능을 사용하는 것입니다. 또한 AIX 6.1용 Tivoli Storage Manager for Advanced Copy Services V6.1의 전체 버전을 구매하여 설치할 수 있습니다.

제 3 부 부록

부록 A. DB2 기술 정보 개요

DB2 기술 정보는 다음 도구 및 메소드를 통해 사용할 수 있습니다.

- DB2 정보 센터
  - 주제 항목(태스크, 개념 및 참조 항목)
  - DB2 도구에 대한 도움말
  - 샘플 프로그램
  - 자습서
- DB2 서적
  - PDF 파일(다운로드)
  - PDF 파일(DB2 PDF DVD)
  - 인쇄된 서적
- 명령행 도움말
  - 명령 도움말
  - 메시지 도움말

주: DB2 정보 센터 주제는 PDF 또는 하드카피 서적보다 자주 갱신됩니다. 최신 정보를 보려면 사용 가능한 문서 갱신사항을 설치하거나 ibm.com에서 DB2 정보 센터를 참조하십시오.

ibm.com에서 추가 DB2 기술 정보(예: 기술 노트, 백서 및 IBM Redbooks® 서적)를 온라인으로 액세스할 수 있습니다. DB2 정보 관리 라이브러리 소프트웨어 사이트 http://www.ibm.com/software/data/sw-library/에 액세스하십시오.

문서 피드백

DB2 문서에 대한 피드백을 환영합니다. DB2 문서를 향상시키는 방법에 대해서 제안사항이 있는 경우 db2docs@ca.ibm.com으로 전자 우편을 보내십시오. DB2 문서 팀에서는 고객의 모든 피드백을 읽지만 직접 응답할 수는 없습니다. 고객의 문제를 더 잘 이해할 수 있도록 가능한 위치에 특정 예를 제공해주십시오. 특정 주제 또는 도움말 파일에 대한 피드백을 보내실 경우, 제목 및 URL을 알려주십시오.

DB2 고객 지원에 문의할 때 이 전자 우편 주소를 사용하지 마십시오. 문서에서 해결할 수 없는 DB2 기술 문제점이 있는 경우, 해당 지역의 IBM 서비스 센터에 도움을 요청하십시오.

DB2 기술 라이브러리(하드카피 또는 PDF 형식)

다음 표는 IBM Publications Center(www.ibm.com/shop/publications/order)에서 사용할 수 있는 DB2 라이브러리에 대해 설명합니다. PDF 형식의 영문 DB2 버전 9.7 매뉴얼 및 번역된 버전은 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947에서 다운로드할 수 있습니다.

표에 인쇄할 수 있는 책이 나와 있는 경우에도, 사용 국가 또는 지역에서 해당 책을 사용할 수 없을 수도 있습니다.

매뉴얼이 갱신될 때마다 문서 번호가 증가합니다. 다음 사항을 참조하여 읽고 있는 매뉴얼이 최신 버전인지 확인하십시오.

주: *DB2 정보 센터*는 PDF 또는 하드카피 서적보다 자주 갱신됩니다.

표 27. DB2 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
관리 API 참조서	SA30-3958-00	예	2009년 8월
관리 루틴 및 뷰	SA30-3955-00	아니오	2009년 8월
Call Level Interface Guide and Reference, Volume 1	SC27-2437-00	예	2009년 8월
Call Level Interface Guide and Reference, Volume 2	SC27-2438-00	예	2009년 8월
명령어 참조서	SA30-3959-00	예	2009년 8월
데이터 이동 유틸리티 안내서 및 참조서	SA30-3969-00	예	2009년 8월
데이터 복구 및 고가용성 안내서 및 참조서	SA30-3970-00	예	2009년 8월
데이터베이스 관리 개념 및 구성 참조서	SA30-3951-00	예	2009년 8월
데이터베이스 모니터링 안내서 및 참조서	SA30-3953-00	예	2009년 8월
데이터베이스 보안 안내서	SA30-3971-00	예	2009년 8월
DB2 Text Search Guide	SC27-2459-00	예	2009년 8월
Developing ADO.NET and OLE DB Applications	SC27-2444-00	예	2009년 8월
Developing Embedded SQL Applications	SC27-2445-00	예	2009년 8월
Developing Java Applications	SC27-2446-00	예	2009년 8월
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-2447-00	아니오	2009년 8월

표 27. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Developing User-defined Routines(SQL and External)	SC27-2448-00	예	2009년 8월
Getting Started with Database Application Development	GI11-9410-00	예	2009년 8월
Linux 및 Windows에서 DB2 설치 및 관리 시작하기	GA30-3960-00	예	2009년 8월
자국어 안내서	SA30-3972-00	예	2009년 8월
DB2 Server 설치	GA30-3962-00	예	2009년 8월
IBM Data Server Client 설치	GA30-3963-00	아니오	2009년 8월
Message Reference Volume 1	SC27-2450-00	아니오	2009년 8월
Message Reference Volume 2	SC27-2451-00	아니오	2009년 8월
Net Search Extender Administration and User's Guide	SC27-2469-00	아니오	2009년 8월
파티셔닝 및 클러스터링 안내서	SA30-3973-00	예	2009년 8월
pureXML Guide	SC27-2465-00	예	2009년 8월
Query Patroller 관리 및 사용자 안내서	SA30-3974-00	아니오	2009년 8월
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC27-2468-00	아니오	2009년 8월
SQL Procedural Languages: Application Enablement and Support	SC27-2470-00	예	2009년 8월
SQL 참조서, 볼륨 1	SA30-3956-00	예	2009년 8월
SQL 참조서, 볼륨 2	SA30-3957-00	예	2009년 8월
문제점 해결 및 데이터베이스 성능 조정	SA30-3952-00	예	2009년 8월
DB2 버전 9.7로 업그레이드	SA30-3961-00	예	2009년 8월
Visual Explain 자습서	SA30-3968-00	아니오	2009년 8월
DB2 버전 9.7의 새로운 내용	SA30-3967-00	예	2009년 8월
Workload Manager Guide and Reference	SC27-2464-00	예	2009년 8월

표 27. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
XQuery Reference	SC27-2466-00	아니오	2009년 8월

표 28. DB2 Connect 특정 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
DB2 Connect Personal Edition 설치 및 구성	SA30-3965-00	예	2009년 8월
DB2 Connect Server 설치 및 구성	SA30-3966-00	예	2009년 8월
DB2 Connect 사용자 안내서	SA30-3964-00	예	2009년 8월

표 29. Information Integration 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	예	2009년 8월
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	예	2009년 8월
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	아니오	2009년 8월
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	예	2009년 8월
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	예	2009년 8월

인쇄된 DB2 서적 주문

인쇄된 DB2 서적이 필요한 경우, 대부분 온라인으로 구매할 수 있으나 모든 국가 또는 지역에 해당되지는 않습니다. 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. *DB2 PDF 문서* DVD의 일부 소프트카피 서적은 인쇄할 수 없다는 점에 유의하십시오. 예를 들면, *DB2 메시지 참조서*의 어떤 볼륨도 인쇄된 서적으로 사용할 수 없습니다.

DB2 PDF 문서 DVD에서 사용할 수 있는 다수의 DB2 서적의 인쇄된 버전은 IBM에서 유료로 주문할 수 있습니다. 주문하는 위치에 따라 IBM Publications Center에서 온라인으로 서적을 주문할 수도 있습니다. 해당 국가 또는 지역에서 온라인 주문이

불가능하면, 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. DB2 PDF 문서 DVD의 모든 서적을 인쇄할 수는 없다는 점에 유의하십시오.

주: 가장 최신 및 완료된 DB2 문서는 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7의 DB2 정보 센터에서 유지보수됩니다.

인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.

- 해당 국가 또는 지역에서 인쇄된 DB2 서적을 온라인으로 주문할 수 있는지 여부를 확인하려면 http://www.ibm.com/shop/publications/order의 IBM Publications Center를 확인하십시오. 서적 주문 정보에 액세스하려면 국가/지역/언어를 선택한 다음 해당 위치에서 주문 지시사항을 따르십시오.
- 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.
  1. 다음 웹 사이트 중 하나에서 해당 지역 담당자에 대한 문의처 정보를 찾으십시오.
     - www.ibm.com/planetwide에 있는 IBM 월드 와이드 문의처 디렉토리
     - http://www.ibm.com/shop/publications/order의 IBM Publications 웹 사이트. 사용 지역의 해당 서적 홈 페이지에 액세스하려면 해당 국가, 지역 또는 언어를 선택해야 합니다. 이 페이지에서 "이 제품의 정보" 링크를 수행하십시오.
  2. 전화로 주문할 경우, 주문할 DB2 서적을 지정하십시오.
  3. 담당자에게 주문하려는 서적의 제목 및 문서 번호를 제공하십시오. 서적의 제목 및 문서 번호는 406 페이지의 『DB2 기술 라이브러리(하드카피 또는 PDF 형식)』를 참조하십시오.

명령행 처리기에서 SQL 상태 도움말 표시

DB2 제품은 SQL문의 결과로 나타나는 상태에 대한 SQLSTATE 값을 리턴합니다. SQLSTATE 도움말은 SQL 상태 및 SQL 상태 클래스 코드의 의미를 설명합니다.

SQL 상태 도움말을 시작하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? sqlstate or ? class code
```

여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자를 나타냅니다.
예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

DB2 정보 센터의 다른 버전에 액세스

DB2 버전 9.7 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/입니다.

DB2 버전 9.5 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/입니다.

DB2 버전 9 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9/입니다.

DB2 버전 8 주제에 대한 버전 8 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v8/입니다.

DB2 정보 센터에서 원하는 언어로 항목 표시

DB2 정보 센터는 브라우저 환경 설정에 지정된 언어로 주제 항목을 표시합니다. 주제가 원하는 언어로 변환되지 않은 경우, DB2 정보 센터는 영어로 주제 항목을 표시합니다.

- Internet Explorer 브라우저에서 원하는 언어로 항목을 표시하려면 다음을 수행하십시오.
  1. Internet Explorer에서 **도구 —> 인터넷 옵션 —> 언어...** 단추를 누르십시오. 언어 환경 설정 창이 열립니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 누르십시오.

       주: 언어를 추가했다고 컴퓨터에 원하는 언어로 항목을 표시하는 데 필요한 글꼴이 설치되는 것은 아닙니다.
     - 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
  3. 브라우저 캐시를 지운 후 페이지를 새로 고쳐 원하는 언어로 DB2 정보 센터를 표시하십시오.
- Firefox 또는 Mozilla 브라우저에서 원하는 언어로 주제 항목을 표시하려면 다음을 수행하십시오.
  1. **도구 —> 옵션 —> 고급** 대화 상자의 **언어** 섹션에서 단추를 선택하십시오. 환경 설정 창에 언어 패널이 표시됩니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 눌러 언어 추가 창에서 언어를 선택합니다.

– 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.

3. 브라우저 캐시를 지운 후 페이지를 새로 고쳐 원하는 언어로 DB2 정보 센터를 표시하십시오.

일부 브라우저 및 운영 체제 조합에서 운영 체제의 국가별 설정을 선택한 로케일 및 언어로 변경해야 합니다.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신

로컬로 설치된 DB2 정보 센터는 주기적으로 갱신해야 합니다.

시작하기 전에

DB2 버전 9.7 정보 센터는 등록된 상태여야 합니다. 자세한 내용은 *DB2 Server 설치*의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치』 주제를 참조하십시오. 정보 센터 설치에 적용되는 모든 전제조건 및 제한사항은 정보 센터 갱신에도 적용됩니다.

이 태스크에 대한 정보

기존의 DB2 정보 센터는 자동 또는 수동으로 갱신할 수 있습니다.

- 자동 갱신 - 기존 정보 센터 기능 및 언어를 갱신합니다. 자동 갱신의 또 다른 이점으로는 갱신 동안 정보 센터를 아주 잠시 동안만 사용 불가능하다는 점입니다. 또한 자동 갱신은 주기적으로 실행되는 기타 일괄처리 작업의 일부로 실행되도록 설정 가능합니다.
- 수동 갱신 - 갱신 프로세스 중에 기능이나 언어를 추가하려는 경우 사용하십시오. 예를 들어, 로컬 정보 센터는 기본적으로 영어와 프랑스어로 설치되어 있으며 기존 정보 센터의 기능 및 언어 갱신 외에도 수동 갱신으로 독어도 설치할 수 있습니다. 단, 수동 갱신을 수행하려면 정보 센터를 중지한 다음 갱신하고 재시작해야 합니다. 정보 센터는 갱신 프로세스 동안에는 사용할 수 없습니다.

프로시저

이 주제는 자동 갱신 프로세스에 대한 설명입니다. 수동 갱신에 대한 지시사항은 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신』 주제를 참조하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 자동으로 갱신하려면 다음을 수행하십시오.

1. Linux 운영 체제에서,
   a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.
   b. 설치 디렉토리에서 `doc/bin` 디렉토리까지 탐색하십시오.

c. 다음과 같이 ic-update 스크립트를 실행하십시오.

```
ic-update
```

2. Windows 운영 체제에서,
   a. 명령 창을 여십시오.
   b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.
   c. 설치 디렉토리에서 doc₩bin 디렉토리까지 탐색하십시오.
   d. 다음과 같이 ic-update.bat 파일을 실행하십시오.

```
ic-update.bat
```

결과

DB2 정보 센터가 자동으로 재시작됩니다. 갱신사항이 사용 가능한 경우, 정보 센터에는 새로 갱신된 주제가 표시됩니다. 정보 센터 갱신을 사용할 수 없는 경우, 메시지가 로그에 추가됩니다. 로그 파일은 doc₩eclipse₩configuration 디렉토리에 있습니다. 이 로그 파일 이름은 임의로 생성된 번호입니다. (예: 1239053440785.log).

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신

DB2 정보 센터를 로컬로 설치한 경우, IBM으로부터 문서 갱신사항을 받아 설치할 수 있습니다.

로컬로 설치된 DB2 정보 센터를 수동으로 갱신하려면 다음을 수행하십시오.

1. 컴퓨터에서 DB2 정보 센터를 중지한 후 독립형 모드에서 다시 시작하십시오. 독립형 모드에서 정보 센터를 실행하면 사용자의 네트워크와 연결된 다른 사용자는 정보 센터에 액세스할 수 없으므로 갱신사항을 적용할 수 있습니다. DB2 정보 센터의 워크스테이션 버전은 항상 독립형 모드에서 실행됩니다. .
2. 어떤 갱신사항이 사용 가능한지 확인하려면 갱신 기능을 사용하십시오. 설치해야 할 갱신사항이 있는 경우, 갱신 기능을 사용하여 이를 가져온 후 설치할 수 있습니다.

   주: 인터넷에 연결되지 않은 머신에 DB2 정보 센터 갱신사항을 설치해야 할 경우, 인터넷에 연결된 머신을 사용하여 갱신 사이트를 로컬 파일 시스템에 미러해야 DB2 정보 센터가 설치됩니다. 네트워크 상에 문서 갱신사항을 설치하려는 사용자가 많을 경우에는 갱신 사이트를 로컬로 미러링하거나 갱신 사이트의 프록시를 작성하여 갱신을 수행하면 각 개인에게 필요한 시간을 줄일 수 있습니다.
   갱신 패키지가 사용 가능하면 갱신 기능을 사용하여 패키지를 가져오십시오. 그러나 갱신 기능은 독립형 모드에서만 사용할 수 있습니다.
3. 독립형 정보 센터를 중지한 후 컴퓨터에서 DB2 정보 센터를 재시작하십시오.

주: Windows 2008, Windows Vista 이상의 경우 이 섹션 다음에 나오는 명령은 관리자로 실행해야 합니다. 전체 관리자 권한으로 명령 프롬프트 또는 그래픽 도구를 열려면 단축키를 마우스 오른쪽 단추로 누른 후 **관리자로 실행**을 선택하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 갱신하려면 다음을 수행하십시오.

1. DB2 정보 센터를 중지하십시오.
   - Windows에서는 **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 정보 센터** 서비스를 마우스 오른쪽 단추로 누른 후 **중지**를 선택하십시오.
   - Linux에서는 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 독립형 모드에서 정보 센터를 시작하십시오.
   - Windows 사용자:
     a. 명령 창을 여십시오.
     b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.
     c. 설치 디렉토리에서 doc₩bin 디렉토리까지 탐색하십시오.
     d. 다음과 같이 help_start.bat 파일을 실행하십시오.

        ```
        help_start.bat
        ```
   - Linux 사용자:
     a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 /opt/ibm/db2ic/V9.7 디렉토리에 디폴트로 설치됩니다.
     b. 설치 디렉토리에서 doc/bin 디렉토리까지 탐색하십시오.
     c. 다음과 같이 help_start 스크립트를 실행하십시오.

        ```
        help_start
        ```

   독립형 정보 센터를 표시하기 위해 시스템의 기본 웹 브라우저가 열립니다.
3. **갱신** 단추( )를 누르십시오. (JavaScript™가 브라우저에서 사용 가능해야 합니다.) 정보 센터의 오른쪽 패널에서 **갱신사항 찾기**를 누르십시오. 기존 문서의 갱신사항 목록이 표시됩니다.
4. 설치 프로세스를 시작하려면 설치할 선택란을 체크한 후 **갱신사항 설치**를 누르십시오.
5. 설치 프로세스가 완료되면 **완료**를 누르십시오.
6. 독립형 정보 센터를 중지하십시오.

- Windows에서 설치 디렉토리의 doc¥bin 디렉토리를 탐색한 후 다음과 같이 help_end.bat 파일을 실행하십시오.

  ```
  help_end.bat
  ```

  주: help_end 일괄처리 파일에는 help_start 일괄처리 파일로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start.bat를 중지하는 데 Ctrl-C 또는 다른 메소드를 사용하지 마십시오.
- Linux에서 설치 디렉토리의 doc/bin 디렉토리를 탐색한 후 다음과 같이 help_end 스크립트를 실행하십시오.

  ```
  help_end
  ```

  주: help_end 스크립트에는 help_start 스크립트로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start 스크립트를 중지하는 데 다른 메소드를 사용하지 마십시오.

7. DB2 정보 센터를 재시작하십시오.
   - Windows에서는 **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 정보 센터** 서비스를 마우스 오른쪽 단추로 누른 후 **시작**을 선택하십시오.
   - Linux에서는 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 start
     ```

갱신된 DB2 정보 센터에는 새로 갱신된 주제가 표시됩니다.

DB2 자습서

DB2 자습서는 DB2 제품의 다양한 측면에 대해 학습하는 데 유용합니다. 각 레슨은 단계별 지시사항을 제공합니다.

시작하기 전에

http://publib.boulder.ibm.com/infocenter/db2help/의 정보 센터에서 XHTML 버전의 자습서를 볼 수 있습니다.

일부 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크에 대한 전제조건 설명은 자습서를 참조하십시오.

DB2 자습서

자습서를 보려면 제목을 누르십시오.

『**pureXML®**』(*pureXML Guide*)

XML 데이터를 저장하고 원시 XML 데이터 스토어로 기본 조작을 수행하려면 DB2 데이터베이스를 설정하십시오.

*Visual Explain 자습서*의 『**Visual Explain**』

Visual Explain을 사용하여 성능을 향상시킬 수 있도록 SQL문을 분석, 최적화 및 조정합니다.

DB2 문제점 해결 정보

DB2 데이터베이스 제품을 사용하는 데 도움이 되는 광범위한 문제를 해결하고 판별할 수 있는 정보가 있습니다.

DB2 문서

문제점 해결 정보는 *DB2 문제점 해결 안내서* 또는 *DB2 정보 센터*의 데이터베이스 기본 섹션을 참조하십시오. DB2 진단 도구 및 유틸리티를 사용하여 문제점을 찾아내고 식별하는 방법, 가장 일반적인 문제점에 대한 솔루션 및 DB2 데이터베이스 제품에서 발생할 수 있는 문제점을 해결하는 방법 등의 정보가 있습니다.

DB2 기술 지원 웹 사이트

문제점이 있는 경우 원인 및 솔루션을 찾으려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 서적, 기술 노트, APAR(Authorized Program Analysis Report 또는 버그 수정), FixPack 및 기타 자원에 대한 링크가 있습니다. 이러한 기술 자료를 검색하여 문제에 대한 가능한 솔루션을 찾을 수 있습니다.

http://www.ibm.com/software/data/db2/support/db2_9/에서 DB2 기술 지원 웹 사이트에 액세스하십시오.

이용약관

다음 조건에 따라 이 책을 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 귀하 사업장 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서의 2차적 저작물을 만들거나 본 문서 또는 그 일부를 복제, 배포 또는 전시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 이 책이나 이 책에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대한 어떠한 허가나 라이센스 또는 권한도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 본 문서의 사용이 IBM의 이익을 해친다고 판단되거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 이 사이트에서 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 경우에만 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 본 문서의 내용에 대해 어떠한 보증도 제공하지 않습니다. 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 현 상태대로 제공합니다.

부록 B. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. 비IBM 제품에 대한 정보는 이 책을 처음 발행할 때의 정보에 기초하고 있으며 변경될 수 있습니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트 문자 세트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

지적 자산 라이센스
법정 및 지적 자산 법률
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

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

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠. 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등) 하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 다른 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 되는 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 샘플 프로그램은 어떠한 보증없이 ″있는 그대로″ 제공됩니다. IBM은 샘플 프로그램의 사용으로 인해 발생하는 모든 손해에 대해 책임을 지지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. *_enter* 연도_. All rights reserved.

상표

IBM, IBM 로고 및 ibm.com®은 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 기타 회사의 상표입니다. 현재 IBM 상표 목록은 웹 “저작권 및 상표 정보”(www.ibm.com/legal/kr/copytrade.shtml)에 있습니다.

다음 표장은 기타 회사의 상표 또는 등록상표입니다.

- Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.
- Java™ 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.
- UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.
- Intel, Intel 로고, Intel Inside®, Intel Inside 로고, Intel® Centrino®, Intel Centrino 로고, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium® 및 Pentium은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표 또는 등록상표입니다.
- Microsoft, Windows, Windows NT® 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

[숫자]

A

B

C

D

G

H

I

L

M

N

O

Q

R

S

T

U

V

W

[특수 문자]

SA30-3970-00

Spine information:

Linux, UNIX 및 Windows용 IBM DB2 9.7

데이터 복구 및 고가용성 안내서 및 참조서