
Microsoft
SQL Server 2005
고가용성 구축 및 운영 가이드
Microsoft
SQL Server 2005

저자 약력

이종인 / 다우교육원 SQL Server 전임강사 / 온디멘드 수석 컨설턴트

- FMG 수석 컨설턴트
- 이룬인포텍 DB팀장
- 산업은행 여신지원시스템 DBA
- 동서증권 대리
- 산업은행 / LG-Philips LCD / 한국 문화 진흥 등 다수의 SQL Server 최적화 / 재난대비 컨설팅 및 Oracle 마이그레이션 컨설팅
- MCSE+Internet, MCT, MCDBA (SQL Server 7.0/2000), MCITP (SQL Server 2005), OCP (8/8i/9i/10g), HPCP- Master ASE, HPCP-ASE+StorageWorks

본 가이드가 제공하는 SQL Server 2005 Enterprise Edition의 고가용성 기능을 포함하여 SQL Server 2005가 제공하는 재해 복구 및 고가용 솔루션과 관련해서는 필자가 집필한 도서출판 대림의 [SQL Server 2005 재해 복구 및 고가용 솔루션]에서 보다 자세하게 살펴볼 수 있습니다. SQL Server 2005의 백업 및 복원, 장애 조치 클러스터링, 로그 전달, 데이터베이스 미러링, 피어 투 피어 복제, 온라인 작업, 빠른 복구, Hot-Add 메모리 등의 기능을 습득할 수 있으며 가용성의 확장을 위한 고가용성 통합 솔루션에 대해서도 살펴봅니다. 뿐만 아니라 Log Explorer, Log Rescue, LiteSpeed, SQL Backup Pro 등 ISV가 제공하는 백업 및 복구 제품과 Windows Server가 제공하는 Bare-Metal 복구 솔루션인 자동 복구 마법사의 기능을 살펴봅니다. 또한 SQL Server 2005 장애 조치 클러스터링을 보다 쉽게 이해하기 위해서 Microsoft Virtual Server 2005 R2를 소개하고 Virtual Server 2005 R2에서 Windows Server 2003의 장애 조치 클러스터와 SQL Server 2005 장애 조치 클러스터를 구축하고 관리하는 방법을 다루고 있습니다.

저자의 글

SQL Server 2005 Enterprise Edition은 고가용성 기술에서 비약적인 발전을 실현하고 있습니다. SQL Server 2005 Enterprise Edition은 사용자 데이터베이스에 대한 즉각적인 장애조치(Failover)가 가능하고 클라이언트가 장애 조치를 자동으로 인식하는 데이터베이스 미러링의 모든 기능과 사이트 재해에 대비하고 쓰기 작업이 가능한 상태에서 읽기 작업의 부하를 분산할 수 있는 피어 투 피어 복제, 향상된 장애조치 클러스터링과 로그 전달, 데이터베이스 스냅숏, 빠른 복구, 즉시 파일 초기화, Hot Add 메모리, 온라인 복원 및 온라인 인덱스 작업 등 기업의 핵심 데이터에 대한 고가용성을 유지할 수 있는 필수적인 기능을 제공합니다. 본 가이드에서는 이 가운데에서 SQL Server 2005가 새롭게 제공하는 고가용성 기능에 대해서 가이드를 제공합니다. 본 가이드를 통해서 기업의 핵심 데이터의 가용성을 극대화할 수 있는 방법에 접근해 보기 바랍니다. 지면 관계상 SQL Server 2005 Enterprise Edition이 제공하는 고가용 솔루션의 모든 기능을 설명하지 못하여 아쉬움이 큽니다. 보다 자세한 사항은 도서출판 대림의 [SQL Server 2005 재해 복구 및 고가용 솔루션]을 참조하기 바랍니다.

(2007년 4월)

컨텐츠 관련 문의처

jilee@daoudata.co.kr

목차

1. SQL Server 2005 Enterprise Edition 고가용 솔루션 소개

SQL Server 2005 Enterprise Edition은 하드웨어 또는 소프트웨어 오류 발생 시 응용 프로그램의 가용성을 유지하여 사용자가 인식하는 서비스 중단을 최소화 할뿐만 아니라 사이트의 재해 발생 시 최단 시간 내에 서비스를 다시 가동할 수 있으며 서비스 팩의 적용이나 인덱스의 다시 작성과 같은 유지 보수 업무 작업 중에도 가용성을 유지할 수 있도록 다음과 같은 다양한 고가용 솔루션을 제공합니다.

1) 장애 조치 클러스터링

장애 조치 클러스터링은 SQL Server 인스턴스 단위의 고가용성 기능을 제공합니다. 장애 조치 클러스터는 하나 이상의 노드(서버)와 둘 이상의 공유 디스크(Shared Disk)로 구성됩니다. SQL Server 장애 조치 클러스터는 Windows Server의 MSCS(Microsoft Cluster Service) 클러스터 리소스 그룹에 설치되며 SQL Server 리소스 그룹을 소유한 노드에서 SQL Server 서비스를 수행합니다.

현재 노드를 사용할 수 없는 장애가 발생하게 되면 SQL Server는 다른 노드로 장애 조치를 수행하여 장애가 발생하지 않은 노드에서 서비스를 다시 시작합니다. 이러한 기능은 노드 이름과는 무관해야 하고 노드의 IP주소와도 무관해야 가능합니다. 따라서 특정 노드로부터 독립된 가상 이름과 IP주소를 가지고 있어야 하며 이것을 [가상 서버]라고 부릅니다. 응용 프로그램은 어떤 노드가 가상 서버를 호스트하는지 알 필요 없이 가상 서버 이름이나 IP 주소를 참조하여 가상 서버에 연결하여 작업하게 됩니다.

SQL Server 가상 서버는 공유 디스크 장애를 제외한 하드웨어 오류, 운영 체제 오류 또는 계획된 운영 체제 업그레이드 작업 중에 장애 조치 클러스터의 특정 노드에서 다른 노드로 대해 장애 조치를 수행할 수 있습니다. 따라서, 장애 조치 클러스터링을 사용하여 시스템의 작동 중단 시간을 줄여서 보다 높은 가용성을 제공할 수 있습니다.

그러나 장애 조치 클러스터는 마이크로소프트가 장애 조치 클러스터를 위해서 인증한 하드웨어를 사용해야만 합니다. 또한 공유 디스크 장애에 대한 대비책이 될 수 없습니다. 예를 들어서 RAID5로 구성된 공유 디스크의 경우 디스크 2개가 동시에 손상을 입게 되는 단일 지점 오류(Single-Point of Failure)가 발생하면 치명적인 상태가 되며 특정 사이트 전체에 장애가 발생한 경우에도 마찬가지 입니다. 이것을 극복하기 위해서는 지리적으로 분산된 클러스터(Geo-Cluster)를 구축해야 합니다. 지리적으로 분산된 클러스터링에 대한 하드웨어 인증은 더욱 까다롭고 비용이 더욱 많이 소요될 뿐만 아니라 하드웨어 공급 업체와도 충분히 상담해야 합니다.

장애 조치 클러스터링은 SQL Server 2005 Standard Edition에서는 2노드(서버) 클러스터를 지원하며 Enterprise Edition에서는 최대 8노드 클러스터를 지원합니다.

2) 데이터베이스 미러링

데이터베이스 미러링은 SQL Server 2005에서 새롭게 제공하는 기능으로 로그 전달과 마찬가지로 사용자 데이터베이스 수준에서 동작하며 데이터베이스 단위로 장애 조치를 수행함으로써 데이터베이스 가용성을 높일 뿐만 아니라 데이터베이스의 복사본을 유지하므로 사이트 재해를 대비할 수 있는 소프트웨어 솔루션입니다.

데이터베이스 미러링은 평상시 서비스를 수행하는 주 데이터베이스를 포함하는 주 서버와 즉시 장애 조치할 수 있는 미러 데이터베이스를 포함하는 미러 서버, 선택적으로 자동 장애

조치를 가능하게 하는 모니터(Witness) 서버로 구성됩니다. 미러 데이터베이스는 복구(Recovery) 작업 없이 주 데이터베이스의 전체 백업을 복원(Restore)하는 방법으로 생성되어 평상시 클라이언트는 액세스 할 수 없습니다. 하지만 보고서 작성의 용도로 미러 데이터베이스에 데이터베이스 스냅숏을 만들면 간접적으로 사용할 수 있습니다. 데이터베이스 스냅숏은 스냅숏을 만든 시점의 데이터베이스 내 데이터를 클라이언트가 읽기 전용으로 액세스할 수 있도록 하는 SQL Server 2005 Enterprise Edition이 제공하는 새로운 고가용성 기능입니다.

데이터베이스 미러링은 이전 버전의 SQL Server를 사용하던 것보다 가용성 수준을 훨씬 향상시킬 수 있어서 보다 복잡한 구성과 관리 및 인증된 하드웨어를 요구하는 장애 조치 클러스터링을 대체 할 수 있는 솔루션을 제공합니다. 뿐만 아니라 로그 전달보다 데이터베이스 미러링이 우수한 점은 간단한 장애 조치 전략으로 동기식으로 구성한 경우 데이터의 손실 방지를 지원한 다는 것과 자동 장애 조치가 가능하다는 것, 클라이언트가 자동으로 장애 조치를 인식할 수 있다는 것입니다.

만약 사이트 재해를 대비하기 위해서 지리적으로 분산된 클러스터(Geo-Cluster) 구축을 고려하고 있다면 데이터베이스 미러링을 독립적으로 구성하거나 데이터베이스 미러링과 장애 조치 클러스터링을 통합하여 구성하는 것이 더욱 적은 관리 비용을 투자하여 효율적으로 가용성을 확장할 수 있습니다. 데이터베이스 미러링은 SQL Server 2005 Enterprise Edition에서 지원하며 일부 제한은 있지만 Standard Edition에서도 지원합니다.

3) 로그 전달

로그 전달은 데이터베이스 수준에서 가용성을 확보하기 위한 기능을 수행하며 서비스를 수행하는 주 데이터베이스와 주 데이터베이스의 복사본으로 구성되는 하나 이상의 보조 데이터베이스로 구성됩니다. 로그 전달은 데이터베이스의 복사본을 유지하므로 장애 조치 클러스터링에서 조치할 수 없는 공유 디스크의 손상이나 사이트의 재해 등을 대비하는 솔루션이 될 수 있습니다.

로그 전달은 단일 주 서버(주 데이터베이스 포함)와 하나 이상의 보조 서버(보조 데이터베이스 포함) 및 선택적으로 구성할 수 있는 모니터 서버로 구성됩니다. 주 서버가 정기적으로 트랜잭션 로그 백업을 수행하고 각 보조 서버는 주 데이터베이스의 로그 백업을 정기적으로 복사하고 복원합니다.

이러한 주 데이터베이스의 로그를 백업하는 시점과 보조 서버가 로그 백업을 복원해야 할 시점 사이에 지연 시간을 설정할 수 있으며 다중 대기 데이터베이스를 지원할 수 있는 유연성을 제공합니다. 또한 보조 서버에서 로그를 복원할 때 대기 모드로 복원하여 읽기 작업이 가능하게 구성하면 주 서버에 대해서 부하가 없는 상태에서 읽기 전용 데이터베이스 복사본을 유지할 수 있습니다. 로그 전달은 SQL Server 2005 Enterprise Edition, Standard Edition 및 Workgroup Edition에서 지원합니다.

4) 피어 투 피어 복제

SQL Server 복제는 주 서버(게시자)가 하나 이상의 보조 서버(구독자)에 데이터를 배포할 수 있도록 허용하는 게시자-구독자 모델을 사용합니다. 복제는 이들 서버 전체에 걸쳐 가용성과 확장성을 제공합니다. 복제는 구독자에게 필요한 데이터만 제공하기 위한 필터링 기능을 지원하고 분할 업데이트를 허용합니다. 구독자는 온라인 상태이며 로그 전달과 같이 쿼리 복구 과정이 없이도 보고 또는 다른 기능에 사용할 수 있습니다.

SQL Server는 스냅숏, 트랜잭션 및 병합 등 세 가지 기본 유형의 복제와 다양한 옵션을 제공합니다. 트랜잭션 복제는 대기 시간이 가장 짧기 때문에 고가용성을 위해 가장 일반적으로 사용되며 SQL Server 2005 Enterprise Edition에서 새롭게 제공하는 피어 투 피어(Peer-To-Peer) 트랜잭션 복제를 통해서 가용성과 확장성도 제공합니다. 피어 투 피어 트랜잭션 복제의 모든 노드는 피어라고 하며 각 노드가 게시자인 동시에 구독자 역할을 합니다. 각 노드는 동일한 스키마와 데이터를 게시하고 구독합니다. 모든 노드에서 변경(삽입, 업데이트 및 삭제) 작업과 조회(검색)를 수행할 수 있습니다. 변경 내용이 두 번 이 상 노드 전체에서 순환되는

것을 방지하기 위해 변경 내용이 지정된 노드에 적용되면 복제가 이를 자동 인식합니다. 그렇지만, 복제는 데이터의 손실 방지를 보장하지 않습니다. 피어 투 피어 복제는 SQL Server 2005 Enterprise Edition에서만 지원합니다.

5) SQL Server 2005 Enterprise Edition이 제공하는 추가적인 고가용성 기능

SQL Server 2005 Enterprise Edition은 고가용성을 위한 다양한 추가적인 기능을 제공하고 있습니다. 데이터베이스 스냅숏은 특정 시점의 읽기 전용 이미지를 보관하여 사용자 실수로부터 데이터를 복구할 수 있는 솔루션을 제공하며 사용자의 액세스가 불가능한 미러 데이터베이스에 대한 읽기 작업을 가능하게 합니다. 빠른 복구 기능은 복구 프로세스의 시간을 단축하여 빠른 시간 내에 데이터베이스를 사용 가능한 상태로 유지하고, 온라인 인덱스 작업은 인덱스를 생성하거나 수정할 때 온라인으로 작업을 수행하여 사용자의 데이터 액세스를 방해하지 않으며 온라인 복원 기능을 통해서는 데이터베이스가 온라인인 상태에서 해당 데이터베이스의 일부분을 복원할 수 있습니다. 뿐만 아니라 Hot Add 메모리 기능을 통해서 시스템을 종료하지 않고 즉시 메모리를 추가하여 사용할 수 있습니다. SQL Server는 이 이외에도 Edition에 상관 없이 즉시 파일 초기화 기능을 추가하여 복원 작업 및 파일 크기 증가, 파일 추가 작업 등의 성능을 향상하였습니다.

6) 하드웨어 업체와 소프트웨어 업체 기능 지원을 통한 고가용성 강화

가용성 향상을 위해서 SQL Server 2005의 자체 기능뿐만 아니라 하드웨어 업체와 소프트웨어 업체가 제공하는 다양한 기능의 제품들도 고려합니다. 많은 스토리지 업체가 자체적인 하드웨어 또는 소프트웨어 복제 솔루션을 가지고 있으며 또한 많은 백업 소프트웨어 업체가 보다 빠르게 백업과 복원을 수행하거나 백업 세트를 압축하는 소프트웨어 제품을

개발하여 판매하고 있습니다. SQL Server가 제공하는 고가용성 기능과 이와 같은 마이크로소프트 파트너 회사들이 제공하는 제품들을 통해서 SQL Server 2005의 가용성을 향상시킬 수 있으므로 예산과 업무의 중요성을 고려하여 SQL Server 2005 Enterprise Edition과 함께 검토합니다.

본 가이드에서는 이와 같은 고가용 솔루션 가운데 데이터베이스 미러링과 피어 투 피어 복제, 빠른 복구 및 온라인 작업, Hot-Add 메모리 등 SQL Server 2005에서 새롭게 제공하는 기능에 대해서 살펴봅니다.

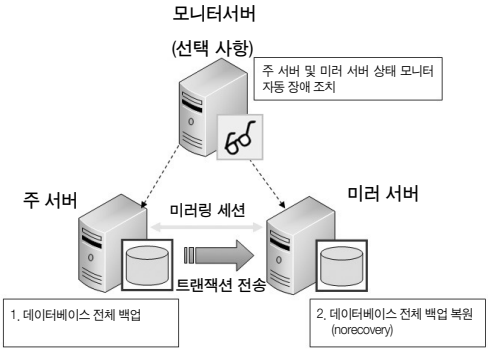
2. 데이터베이스 미러링

1) 데이터베이스 미러링 개요

데이터베이스 미러링은 데이터베이스 가용성 향상을 위해 검토할 수 있는 SQL Server 2005가 새롭게 제공하는 기능입니다. 데이터베이스 미러링은 서버 간에 직접 전송하여 데이터의 손실 방지를 보장하며 대기 서버로 장애 조치를 빠르게 수행할 수 있습니다. 뿐만 아니라 클라이언트의 연결 정보를 자동으로 리디렉션하고 장애 조치(Failover)시 대기 서버와 데이터베이스에 자동으로 연결되도록 클라이언트 응용 프로그램 코드를 작성할 수 있으며 데이터베이스 복사본을 유지하므로 사이트 재해 대비를 위한 솔루션을 제공합니다. 일반적으로 데이터 손상을 최소화하면서 신속한 장애 조치를 수행하려면 하드웨어 비용이 많이 들고 소프트웨어가 복잡해집니다. 그러나 데이터베이스 미러링은 커밋된 데이터의 손실 없이 신속하게 장애 조치를 수행할 수 있으며 전용 하드웨어가 필요 없고 설정 및 관리가 용이한 새로운 기술입니다

[주의]
데이터베이스 미러링은 SQL Server 2005 서비스 팩1부터 지원하는 기능입니다. 데이터베이스 미러링을 운영 시스템에 적용하기 위해서는 반드시 서비스 팩1 이상을 적용하도록 합니다.

데이터베이스 미러링은 주 서버와 미러 서버, 그리고 선택적인 모니터 서버로 구성됩니다. 주(운영) 서버는 데이터베이스를 백업하고 미러 서버는 해당 백업으로 NORECOVERY 옵션으로 복원합니다. 그리고, 미러를 설정하면 주 데이터베이스에서 수행한 모든 변경 사항의 트랜잭션 로그는 미러 데이터베이스에 반영됩니다.



〈 데이터베이스 미러링 아키텍처 〉

주 데이터베이스와 미러 데이터베이스는 별개의 서버 인스턴스(즉 Microsoft SQL Server 데이터베이스 엔진의 인스턴스)에 있어야 합니다. 두 서버 인스턴스는 서로 세션을 유지하는 데 이것을 미러링 세션이라고 합니다. 미러링 세션을 주고 받는 각각의 서버를 미러링 파트너라고 하며 주 서버와 미러 서버외에 데이터베이스 미러링 세션은 모니터라고 하는 선택적인 구성 요소인 모니터 서버를 가질 수 있습니다. 모니터 서버는 주 데이터베이스와 미러 데이터베이스의 상태를 모니터하다가 주 데이터베이스에 장애가 발생되면 자동으로 즉시 장애 조치(Failover) 합니다.

2) 데이터베이스 미러링의 기능

- 주 서버의 장애 시 즉시 미러 서버의 데이터베이스가 서비스를 수행합니다
- 클라이언트의 연결 정보를 자동으로 리디렉션하고 장애 조치 시 미러 데이터베이스에 자동으로 연결되도록 간단하게 클라이언트를 구성할 수 있습니다.

- 주 서버의 유지 관리를 위한 다운타임을 최소화합니다. .
- 데이터베이스 스냅숏과 함께 사용하면 사용자나 관리자의 실수로부터 빠른 복구가 가능합니다.
- 데이터베이스 스냅숏과 함께 사용하면 DBCC와 같은 관리 작업을 효율적으로 수행할 수 있습니다.
- 데이터베이스 스냅숏과 함께 사용하면 보고서 작업의 부하를 분산할 수 있습니다.

3) 데이터베이스 미러링을 구성시 고려 사항

SQL Server 2005 데이터베이스 미러링을 구성하기 위해서 고려해야 하는 사항은 다음과 같습니다.

- 데이터베이스 미러링의 대상은 사용자 데이터베이스만 지원합니다.
- 주 서버의 로그인 계정을 미러 서버에 동일하게 생성해야 합니다.
- 데이터베이스 미러링의 대상은 데이터베이스에는 전체 복구 모델을 사용해야 합니다. 대량 로그 작업의 로그 레코드는 미러 데이터베이스로 보낼 수 없습니다.
- 미러 데이터베이스는 NORECOVERY 옵션을 사용해서 복원한 뒤 순차적인 트랜잭션 로그 백업의 복원으로 초기화해야 합니다.
- 미러 데이터베이스 이름은 주 데이터베이스 이름과 같아야 합니다.
- 미러 데이터베이스가 복원 중인 상태에 있기 때문에 직접 액세스할 수 없습니다. 사용자가 미러 데이터베이스에 액세스하기 위해서는 데이터베이스 스냅숏을 만들어야 합니다.
- 데이터베이스의 미러링은 SQL Server 2005 Enterprise Edition에서 모든 기능을 수행할 수 있고, SQL Server 2005 Standard Edition은 보안(안전성)이 ON인 동기 모드만을 지원합니다. Workgroup Edition과 Express Edition은 모니터 서버 기능만 수행할 수 있습니다.

4) 미러링 운영 모드

데이터베이스 미러링의 모드는 다양한 수준의 가용성, 데이터 보호 및 성능을 제공하며 크게는 동기(보호 우선) 모드와 비동기(성능 우선) 모드로 구분하며 동기(보호 우선) 모드는 다시 자동 장애 조치가 있는 보호 우선 모드와 자동 장애 조치가 없는 보호 우선 모드로 구분합니다. 각 운영 모드는 다음과 같습니다.

용어] 가용성 우선모드

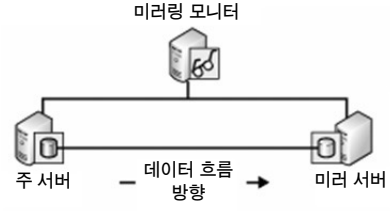
SQL Server 2005의 출시 당시에 제공된 온라인 설명서에서는 데이터베이스의 운영 모드를 가용성 우선 모드, 보호 우선 모드, 성능 우선 모드 세 가지로 구분하고 동기 모드 가운데 자동 장애 조치(Failover)가 있는 보호 우선 모드를 가용성 우선 모드라고 하고 자동 장애 조치(Failover)가 없는 보호 우선 모드를 보호 우선 모드라고 정의하였으나 2006년 7월 출시된 온라인 설명서에서부터는 본 가이드와 같이 동기 모드를 모니터 서버의 유무에 따라서 자동 장애 조치(Failover)가 있는 보호 우선 모드와 자동 장애 조치가 없는 보호 우선 모드로 구분하고 있습니다. 비동기 (성능 우선) 모드 용어는 변동 사항이 없으며 **더 이상 가용성 우선 모드라는 용어는 사용하지 않습니다.**

① 동기 (보호 우선) 모드

트랜잭션 보안(안전성)이 FULL로 설정된 경우는 동기 운영입니다. 동기 운영에서는 커밋된 트랜잭션이 두 파트너에서 모두 커밋되어야 합니다. 따라서, 트랜잭션 대기 시간이 길어질 위험이 있습니다. 동기 모드(보호 우선 모드)는 미러링 모니터라는 세 번째 서버가 있는지 여부에 따라 자동 장애 조치(Failover)가 있는 보호 우선 모드와 자동 장애 조치가 없는 보호 우선 도므로 구분됩니다.

• **자동 장애 조치(Failover)가 있는 보호 우선 모드** : 서버 한 대가 손실되어도 데이터베이스는 여전히 작동되므로 고가용성이 지원됩니다. 이를 위해서는 반드시 모니터 서버가 있어야 합니다. 미러링 모니터는 두 파트너와는 달리 데이터베이스를 제공하지 않습니다. 미러링 모니터는 주 서버와 미러 서버의 상태를 모니터 합니다. 주 서버와 모니터 서버는 쿼럼을 구성하고 주 서버에서 데이터베이스 서비스를 유지하고 있다가 주 서버를 사용할 수 없게 되면 모니터 서버는 미러 서버와 쿼럼을 구성하고 자동으로 미러 서버로 장애 조치됩니다.

용어] 쿼럼(Quorum)
데이터베이스 미러링에서 쿼럼은 데이터베이스 미러링 세션에서 둘 이상의 서버 인스턴스가 서로 연결될 때 존재하는 관계입니다. 일반적으로 쿼럼은 3개의 상호 연결된 서버 인스턴스를 포함합니다. 미러링 모니터 서버가 설정된 경우 쿼럼이 있어야만 데이터베이스를 사용 할 수 있습니다. 쿼럼은 자동 장애 조치(Failover)를 지원하는 보호 우선 모드를 위해 디자인되었으므로 한 번에 하나의 파트너만 데이터베이스를 소유할 수 있습니다.



트랜잭션 보안이 FULL입니다.

〈 자동 장애 조치(Failover)가 있는 보호 우선 모드 〉

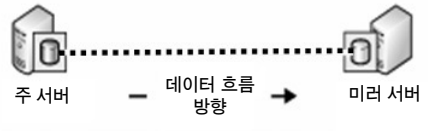
• **자동 장애 조치(Failover)가 없는 보호 우선 모드** : 동기적으로 운영되나 미러링 모니터가 없으므로 수동 장애 조치만 가능합니다.

트랜잭션 보안이 FULL입니다.

〈 자동 장애 조치(Failover)가 없는 보호 우선 모드 〉

② 비동기 (성능 우선) 모드

트랜잭션 보안이 OFF로 설정된 경우는 비동기 운영입니다. 트랜잭션은 미러 서버의 로그 확정을 기다리지 않고 커밋하며 트랜잭션 대기 시간을 최소화합니다. 성능은 강화하지만 고가용성은 저화됩니다. 성능 우선 모드는 비동기 방식으로 트랜잭션이 전송되므로 주 서버의 장애시 잠재적인 데이터 손실이 발생할 수 있습니다. 성능 우선 모드는 원거리의 사이트로 재해 대비를 하거나 잠재적인 일부의 데이터 손실을 허용하는 경우에 사용할 수 있습니다.



트랜잭션 보안이 OFF입니다.

〈 성능 우선 (비동기) 모드 〉

5) 데이터베이스 미러링 세션에 클라이언트 연결

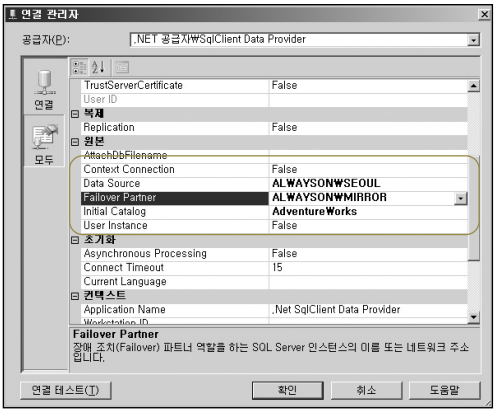
데이터베이스 미러링 세션에 연결하기 위해 클라이언트는 SQL 네이티브 클라이언트(SNAC) 또는 .NET Framework Data Provider for SQL Server를 사용할 수 있습니다. 연결 문자열에서 초기 주 서버 파트너와 데이터베이스, 그리고 장애 조치 파트너 서버를 지정하면 주 데이터베이스에서 장애가 발생해서 미러 데이터베이스로 장애 조치가 발생했을 때 자동으로 클라이언트 연결을 변경합니다.

" Data Source=〈주 서버〉; Failover Partner=〈미러 서버〉;Initial Catalog=AdventureWorks; Network=dbmssocn;" --TCP/IP를 사용하는 경우

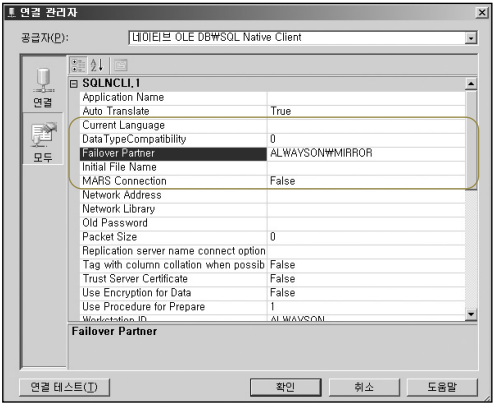
연결을 시도할 때 데이터 액세스 공급자는 최초 주 서버 파트너 이름부터 사용하기 시작합니다. 연결 시 최초 파트너와의 연결을 완료하면 데이터 액세스 공급자는 장애 조치 파트너인 현재 미러 서버의 서버 인스턴스 이름을 다운로드합니다. 이 이름은 캐시에 저장되며 클라이언트에서 제공한 장애 조치 파트너 이름(있는 경우)을 덮어씁니다. 그런 후에는 .NET Framework Data Provider for SQL Server에서 장애 조치 파트너 이름을 업데이트하지 않습니다. 반면 SQL 네이티브 클라이언트는 후속 연결이나 연결 다시 설정에서 다른 파트너 이름이 반환될 때마다 캐시를 업데이트합니다. 그리고, 이후에 초기 파트너 서버와의 연결 시도가 실패하면 데이터 액세스 공급자는 장애 조치 파트너 이름으로 연결을 시도하게 됩니다.

또한, 장애 조치 파트너 이름이 연결 문자열에 제공되는 경우 데이터 액세스 공급자의 동작은 클라이언트의 네트워크 프로토콜과 운영 체제에 따라 다음과 같이 달라집니다. TCP/IP의 경우 클라이언트에서 Microsoft Windows XP 이상을 실행 중이면 데이터베이스 미러링과 관련된 연결 다시 시도 알고리즘에 의해 연결 시도가 조정됩니다. [연결 다시 시도 알고리즘]에 관한 자세한 사항은 온라인 설명서를 참조하기 바랍니다. Microsoft Windows XP 이상을 실행하지 않는 클라이언트 및 다른 네트워크 프로토콜의 경우 초기 파트너를 사용할 수

없으면 데이터 액세스 공급자에서 네트워크 연결 제한 시간이 만료되거나 로그인 제한 시간이 만료될 때까지 초기 연결 시도가 대기합니다. 일반적으로 이 대기 시간은 20초에서 30초 정도입니다. 그런 후에는 장애 조치 파트너에 연결을 시도합니다. 연결에 성공하기 전에 연결 제한 시간이 만료되거나 장애 조치 파트너를 사용할 수 없으면 연결 시도가 실패합니다. 로그인 제한 시간 내에 장애 조치 파트너를 사용할 수 있으며 현재 주 서버인 경우 일반적으로 연결 시도에 성공합니다.



〈 ADO.NET 2.0 공급자에서 장애 조치 파트너 지정〉



〈 SNAC에서 장애 조치 파트너 지정〉

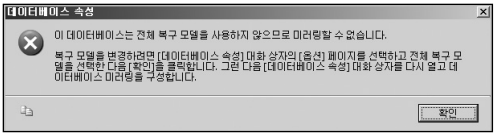
6) 데이터베이스 미러링 설정 및 관리

데이터베이스 미러링 설정은 T-SQL 구문을 사용하거나 SQL Server Management Studio (SSMS)에서 구성할 수 있습니다. 주 데이터베이스를 별도로 백업하거나 복원하는 작업을 제외하고 일반적인 구성은 SSMS를 사용하는 것이 처음에는 용이합니다. 따라서, 먼저 SSMS를 사용해서 데이터베이스 미러링을 설정하면서 그때그때 해당되는 T-SQL 구문도 함께 살펴보겠습니다. 그리고, T-SQL 구문은 별도로 정리하도록 하겠습니다. 일단 T-SQL 구문을 사용해서 미리 주 데이터베이스를 백업하고 NORECOVERY 옵션으로 복원한 뒤에 데이터베이스 미러링 설정을 시작하겠습니다.

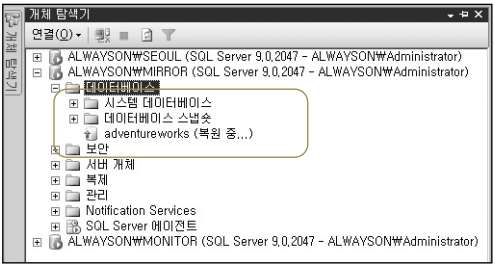
서버 이름	서버 역할	끝점 TCP 포트	끝점 이름
AlwaysOn\SEOUL	주 서버	5022	AW_Mirroring
AlwaysOn\MIRROR	미러 서버	5023	AW_Mirroring
AlwaysOn\MONITOR	모니터(Wintness) 서버	5024	AW_Mirroring

① 미러 데이터베이스 생성

주 데이터베이스에서 데이터베이스 미러링을 설정하기 전에 반드시 전체 복구 모델로 변경하고 전체 백업을 수행해야 합니다. 미러 데이터베이스는 최근 전체 데이터베이스 백업을 NORECOVERY로 복원하여 미러 데이터베이스를 만듭니다. 전체 복구 모델이 아닌 경우에는 다음 그림과 같은 오류가 발생합니다. 또한 미러 데이터베이스의 이름은 주 데이터베이스와 같아야 하며 데이터베이스 미러링 세션 중에도 이름을 바꿀 수 없습니다.



다음과 같이 미러 서버에서 미러 데이터베이스를 NORECOVERY 모델로 [AdventureWorks] 데이터베이스를 복원합니다.

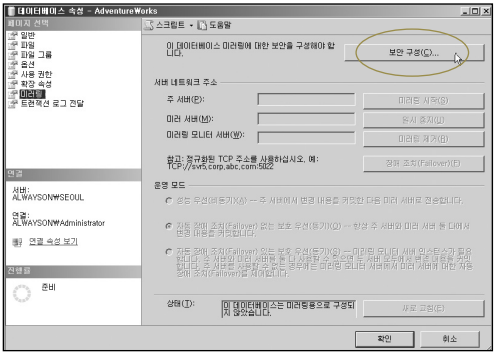


② 데이터베이스 미러링 구성

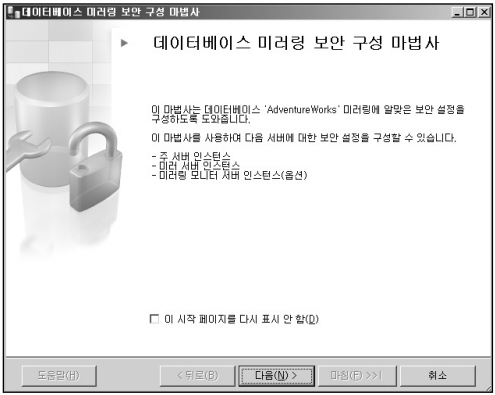
Microsoft SQL Server 2005의 연결 관리는 끝점(Endpoint)을 기반으로 합니다. 끝점은 SQL Server가 네트워크를 통해 통신할 수 있게 하는 SQL Server 개체입니다. 데이터베이스 미러링의 경우 서버 인스턴스에 자체의 전용 데이터베이스 미리링 끝점이 있어야 합니다. 서버 인스턴스의 모든 미러링 연결은 하나의 데이터베이스 미러링 끝점을 사용하며 데이터베이스 미러링 끝점은 다른 서버 인스턴스로부터 데이터베이스 미러링 연결을 받는 데만 사용되는 특별한 용도입니다.

기본적으로 SQL Server 인스턴스에는 데이터베이스 미러링 끝점이 없으므로 데이터베이스 미러링 세션을 설정하는 과정에서 데이터베이스 미러링 끝점을 직접 만들어야 합니다. 시스템 관리자는 데이터베이스 미러링에 참여할 각 서버 인스턴스에 별도의 끝점을 만들어야 하며 끝점에 대한 인증과 암호화를 구성해야 합니다. 데이터베이스 미러링 끝점은 CREATE ENDPOINT T-SQL 구문을 사용하거나 **[데이터베이스 미러링 보안 구성 마법사]**를 사용해서 생성하게 됩니다.

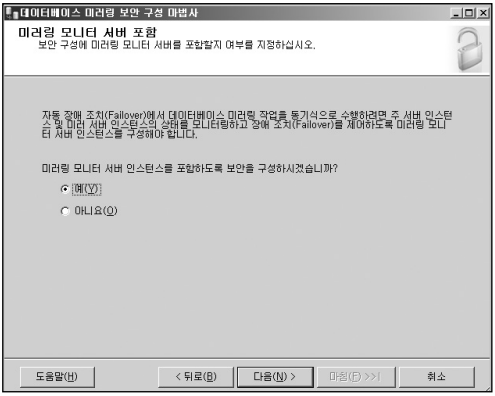
• SSMA 개체 탐색기를 통한 데이터베이스 미러링 구성



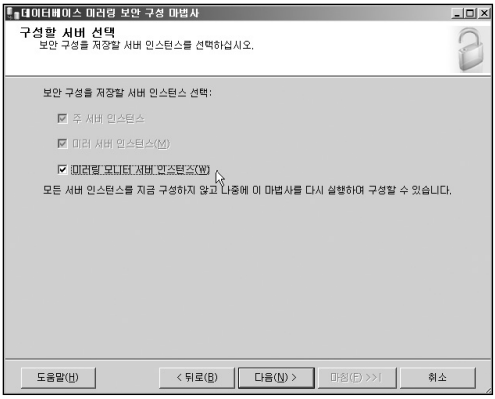
주 서버의 주 데이터베이스의 [속성]에서 [미러링] 페이지를 선택합니다. 다음 그림과 같이 [보안 구성(C)] 버튼을 누릅니다.



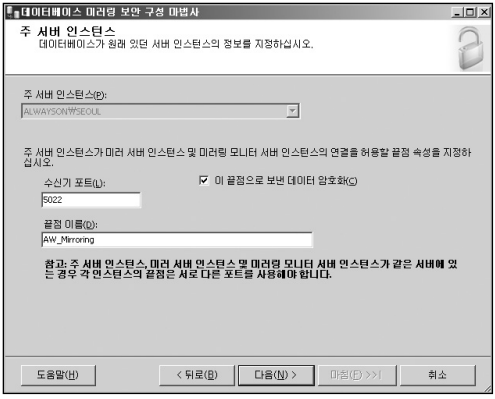
[데이터베이스 미러링 보안 구성 마법사]를 시작되면 [다음 (N)>] 버튼을 누릅니다.



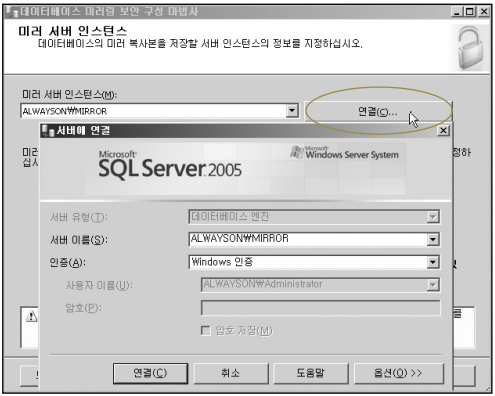
[미러링 모니터 서버 포함] 페이지에서는 미러링 모니터를 포함하여 동기식으로 구성할지 여부를 결정합니다. **[미러링 모니터 서버 인스턴스를 포함하도록 보안을 구성하시겠습니까?]** 질문에 [예(Y)]가 선택되어 있음을 확인하고 **[다음 (N)>]** 버튼을 누릅니다.



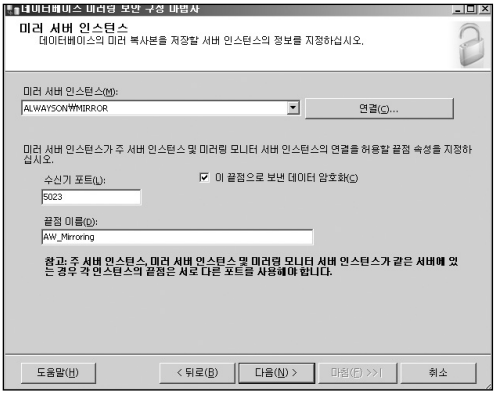
[구성할 서버 선택] 페이지에서 다음과 같이 [미러링 모니터 서버 인스턴스(W)]를 선택하고 [다음(N)>] 버튼을 누릅니다.



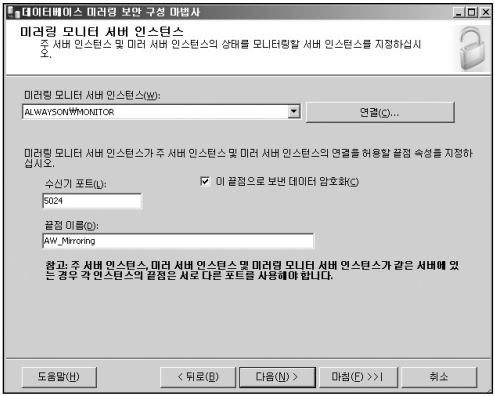
[주 서버 인스턴스] 페이지에서는 주서버에서 미러링 끝점(Endpoint)를 구성합니다. 주 서버의 [수신기 포트(L)]에 포트 번호를 입력하고 [이 끝점으로 데이터 암호화]가 기본으로 선택되어 있음을 확인합니다. [끝점 이름]에 끝점의 이름을 입력하고 [다음 (N)>] 버튼을 누릅니다.



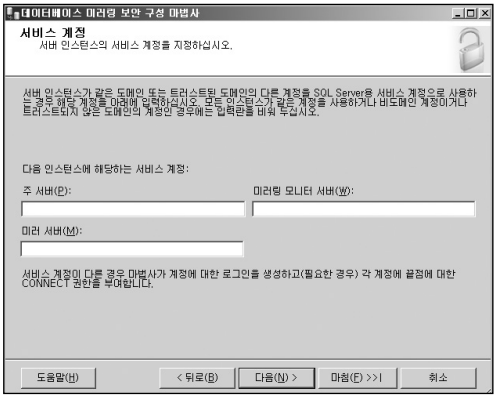
[미러 서버 인스턴스] 페이지에서는 미러 서버의 끝점을 구성합니다. [연결 (C)] 버튼을 눌러서 [서버에 연결] 창을 띄우고 미러 서버 인스턴스 [AlwaysOn\MIRROR]에 연결합니다.



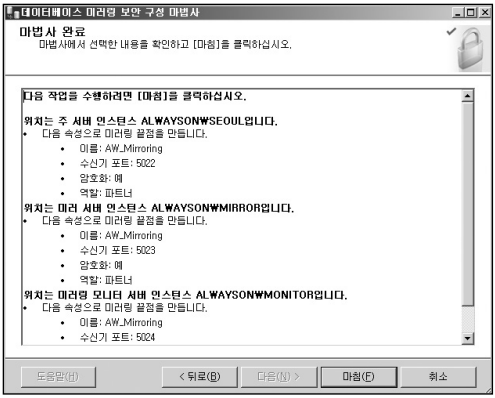
미러 서버 인스턴스에 연결한 뒤 그림과 같이 주 서버 인스턴스의 끝점을 설정한 것과 같이 미러 서버의 인스턴스에 끝점을 설정합니다.



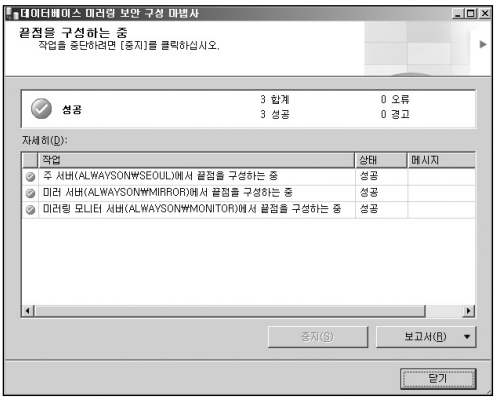
이어서 그림과 같이 모니터 서버 인스턴스에 연결한 뒤 앞에서 주 서버 및 보조 서버 인스턴스에서 끝점을 설정한 것과 같이 모니터 서버의 인스턴스에 끝점을 설정합니다.

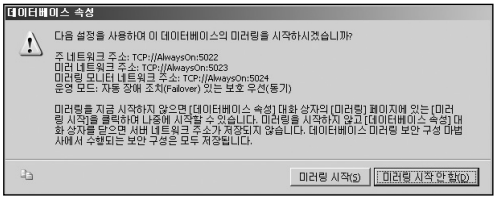


[서비스 계정] 페이지는 끝점에 대한 인증을 구성합니다. 모든 인스턴스가 같은 계정을 사용하는 경우에는 그림과 같이 계정 입력 부분을 비워둡니다.

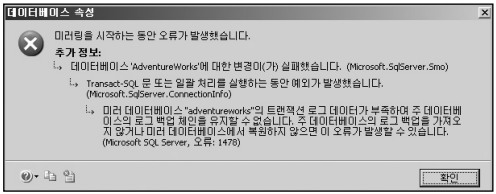


[마법사 완료] 페이지에서 마법사에서 설정한 사항을 확인한 뒤에 [마침(F)] 버튼을 누르면 다음 그림 과 같이 주 서버와 미러 서버 그리고 미러링 모니터 서버에서 끝점을 구성합니다.





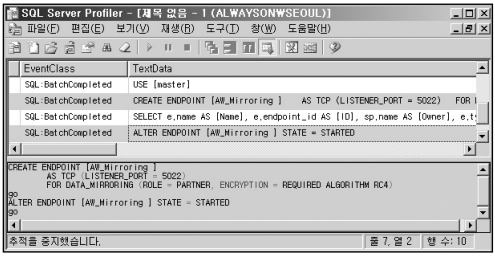
각 서버에 끝점이 생성되고 나면 미러링을 시작할지 여부를 묻는 **[대화창]**이 나타납니다. **[미러링 시작(S)]** 버튼을 누르면 **[대화창]**의 정보대로 각 서버간에 세션을 생성하고 **[자동 장애 조치(Failover)있는 보호 우선(동기)]** 모드로 미러링을 시작합니다. **[미러링 시작 안 함(D)]** 버튼을 누른 뒤 **[데이터베이스 속성]**의 **[미러링]** 페이지에서 **[미러링 시작(S)]** 버튼을 눌러서도 미러링을 시작할 수 있습니다.



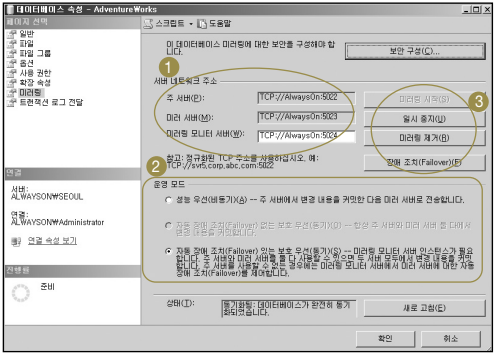
만약, 데이터베이스 미러링을 설정하는 데 최종 수정 사항이 미러 데이터베이스에 복원되지 않으면 앞에서 살펴본 그림과 같은 오류가 발생합니다. 이런 경우에는 각 서버에 끝점만 생성되고 세션은 설정되지 않습니다.

전체 백업 후에 생성된 모든 로그 백업 또한 WITH NORECOVERY를 사용하여 적용한 뒤 다시 **[데이터베이스 미러링 보안 구성 마법사]**를 통해서 설정을 완료합니다.

앞에서와 같이 오류가 발생한 경우에 수행되었던 T-SQL 구문은 다음과 프로파일러로 캡처한 구문과 같이 CREATE ENDPOINT 구문으로 끝점을 생성하고 ALTER ENDPOINT 구문을 수행해서 끝점을 시작 상태로 수정하기만 합니다.



전체 백업 후에 생성된 모든 로그 백업은 WITH NORECOVERY를 사용하여 적용하였으면 다시 **[데이터베이스 미러링 보안 구성 마법사]**를 통해서 설정을 완료하며 이때 **[데이터베이스 속성]** 창의 **[미러링]** 페이지는 다음과 같습니다.



[미러링] 페이지는 다음과 같이 주요 **[보안 구성(C)]** 부분 외에 세 부분으로 구성됩니다.

① **[서버 네트워크 주소]** 부분은 주 서버와 미러 서버, 그리고 미러링 모니터 서버의 네트워크 주소를 확인하는 부분입니다.

② **[운영 모드]** 부분은 미러링 운영 모드를 변경하는 부분입니다. **[성능 우선(비동기)]** 모드와 **[자동 장애 조치(Failover) 없는 보호 우선(동기)]** 모드, **[자동 장애 조치(Failover) 있는 보호 우선(동기)]** 모드 가운데 하나를 설정할 수 있습니다. 미러링 모니터 서버를 구성한 경우에는 **[자동 장애 조치(Failover) 없는 보호 우선(동기)]** 모드는 지원하지 않는 다는 것은 이미 설명하였습니다.

③ **[미러 관리]** 부분은 **[미러 시작(S)]**, **[일시 중지(U)]**, **[미러링 제거(R)]**, **[장애 조치(Failover)(F)]** 버튼을 통해서 쉽게 미러를 관리할 수 있습니다.

■ T-SQL을 통한 데이터베이스 미러링 구성

• 끝점 생성 T-SQL 구문

```
CREATE ENDPOINT 〈끝점 이름 지정〉 [ AUTHORIZATION 〈소유자 로그인 지정〉]
STATE = { STARTED | STOPPED | DISABLED }
        AS TCP (LISTENER_PORT = 〈TCP 포트 지정〉)
FOR DATABASE_MIRRORING
(
  [ AUTHENTICATION =
    { WINDOWS [ { NTLM | KERBEROS | NEGOTIATE } ]
    | CERTIFICATE certificate_name
    | WINDOWS [ { NTLM | KERBEROS | NEGOTIATE } ] CERTIFICATE
    certificate_name
```

```
    | CERTIFICATE certificate_name WINDOWS [ { NTLM | KERBEROS |
    NEGOTIATE } ]
  [ [ [ , ] ] ENCRYPTION = { DISABLED | { { SUPPORTED | REQUIRED }
    [ ALGORITHM { RC4 | AES | AES RC4 | RC4 AES } ] } }
  ]
  [ , ] ROLE = { WITNESS | PARTNER | ALL }
)
```

STATE = { STARTED | STOPPED | DISABLED }

끝점이 생성될 때의 끝점 상태입니다. 끝점을 만들 때 상태를 지정하지 않으면 STOPPED가 기본값이 됩니다.

STARTED

끝점이 시작되어 연결 수신 대기 중입니다.

STOPPED

끝점이 해제되었습니다. 이 상태에서 서버는 끝점 포트로의 수신을 대기하거나 끝점을 사용하기 위해 시도된 요청에 응답하지 않습니다.

DISABLED

끝점이 중지되었습니다. 이 상태일 경우 서버는 포트 요청을 수신하지만 오류를 클라이언트에 반환합니다.

LISTENER_PORT

데이터베이스 미러링 끝점이 수신하는 TCP 포트를 지정합니다.

FOR DATABASE_MIRRORING

페이로드 유형을 데이터베이스 미러링으로 지정합니다.

AUTHENTICATION

이 끝점에 대한 연결의 TCP/IP 인증 요구 사항을 지정합니다. 기본값은 WINDOWS입니다.

WINDOWS [{ NTLM | KERBEROS | NEGOTIATE }]

끝점이 Windows 인증을 사용합니다. 기본 설정입니다. 인증 방법(NTLM 또는 KERBEROS)을 지정한 경우 항상 해당 방법이 인증 프로토콜로 사용됩니다. 기본값인 NEGOTIATE를 적용하면 끝점이 Windows 협상 프로토콜을 사용하여 NTLM이나 Kerberos를 선택합니다.

CERTIFICATE 〈인증서 이름 지정〉

끝점이 〈인증서 이름 지정〉에 지정된 인증서를 사용하여 인증용 ID를 설정하고 연결을 인증하도록 지정합니다. 먼 끝점에는 지정된 인증서의 개인 키와 일치하는 공개 키를 가진 인증서가 있어야 합니다.

WINDOWS [{ NTLM | KERBEROS | NEGOTIATE }] CERTIFICATE 〈인증서 이름 지정〉

Windows 인증을 사용하여 끝점이 연결을 시도하고 이 시도가 실패하면 지정한 인증서를 사용하여 연결을 시도하도록 지정합니다.

CERTIFICATE 〈인증서 이름 지정〉 WINDOWS [{ NTLM | KERBEROS | NEGOTIATE }]

지정한 인증서를 사용하여 끝점이 연결을 시도하고 이 시도가 실패하면 Windows 인증을 사용하여 연결을 시도하도록 지정합니다.

ENCRYPTION=

{ DISABLED | SUPPORTED | REQUIRED } [ALGORITHM { RC4 | AES | AES RC4 | RC4 AES }]
프로세스에서 암호화를 사용할지 여부를 지정합니다. 기본값은 REQUIRED입니다.

DISABLED
연결을 통해 전송되는 데이터를 암호화하지 않도록 지정합니다.

SUPPORTED
반대쪽 끝점이 SUPPORTED나 REQUIRED로 지정된 경우에만 데이터를 암호화하도록 지정합니다.

REQUIRED
이 끝점에 대한 연결이 암화화를 사용하도록 지정합니다. 따라서, 이 끝점에 연결하려고 하는 다른 끝점의 ENCRYPION 이 SUPPOERTED 또는 REQUIRED로 설정되어 있어야 합니다.

[ALGORITHM { RC4 | AES | AES RC4 | RC4 AES }]
필요에 따라서 ALORITHM 인수를 사용하여 암호화 알고리즘을 지정할 수 있습니다. 기본 값은 RC4입니다.

ROLE= { WITNESS | PARTNER | ALL }
데이터베이스 미러링 역할 또는 끝점이 지원하는 역할을 지정합니다.

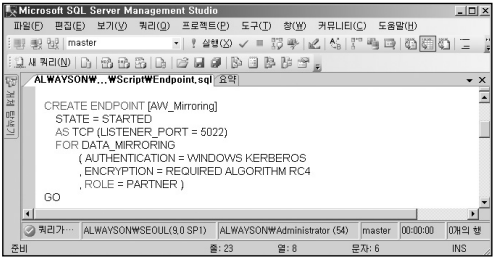
WITNESS
미러링 프로세스에서 끝점이 미러링 모니터의 역할을 수행하도록 합니다.

PARTNER

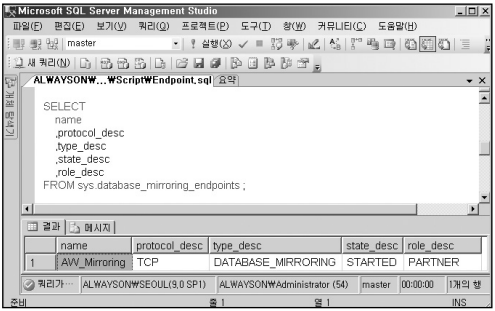
미러링 프로세스에서 끝점이 파트너의 역할을 수행하도록 합니다.

ALL

미러링 프로세스에서 끝점이 미러링 모니터 및 파트너 역할을 모두 수행하도록 지정합니다.



주 서버에서 끝점을 생성하는 구문입니다. 보조 서버 및 모니터 서버에서도 끝점을 생성하는 구문을 수행하여 각각 끝점을 생성합니다.



데이터베이스 미러링 끝점은 sys.database_mirroring_endpoints 카탈로그 뷰에서 확인할 수 있습니다.

■ 데이터베이스 미러링 파트너 및 모니터 서버 관련 T-SQL 구문

끝점을 생성하고 나면 각 파트너 서버간에 미러링 세션을 설정합니다. 다음은 T-SQL로 미러링 세션을 설정할 때 사용되는 구문입니다. ALTER DATABASE 구문에 파트너 또는 모니터 서버를 지정하고 수행하며 옵션은 한번에 하나만 지정 가능합니다.

• 파트너(PARTNER) 구문

```
ALTER DATABASE 〈데이터베이스 이름 지정〉
SET PARTNER =' TCP://〈파트너 IP 주소〉:〈파트너 미러링 포트 번호〉'
        | FAILOVER
        | FORCE_SERVICE_ALLOW_DATA_LOSS
        | OFF
        | RESUME
        | SAFETY { FULL | OFF }
        | SUSPEND
        | TIMEOUT integer
```

' TCP://〈파트너 IP 주소〉:〈모니터 서버 미러링 포트 번호〉'

구성하고자 하는 미러링 파트너 또는 설정을 변경하고자 하는 미러링 파트너 서버의 네트워크 주소를 지정합니다. 서버의 네트워크 주소는 IP 주소로 분해 가능한 주소를 지정합니다. 포트 번호는 미리 끝점으로 구성되어 있어야 합니다. SET PARTNER 문을 실행하려면 두 파트너의 끝점에 대한 STATE가 STARTED로 설정되어 있어야 합니다. 또한 각 파트너 서버 인스턴스에 대한 데이터베이스 미러링 끝점의 ROLE이 PARTNER 또는 ALL로 설정되어 있어야 합니다.

FAILOVER

수동으로 주 서버에서 미러 서버로 장애 조치를 수행합니다. 주 서버에서만 지정할 수 있으며 이 옵션은 SAFETY의 설정 값이 기본값인 FULL로 설정되어 있는 경우에만 유효합니다. 또한, 반드시 master 데이터베이스를 데이터베이스 컨텍스트로 지정하고 수행합니다.

FORCE_SERVICE_ALLOW_DATA_LOSS

자동 또는 수동 장애 조치가 불가능 한 경우에 수행합니다. 주 서버가 다운된 경우에만 가능합니다. 미러 데이터베이스에서만 실행할 수 있습니다.

OFF

데이터베이스 미러링 세션을 제거하고 데이터베이스에서 미러링을 제거합니다. 어느 파트너에나 지정할 수 있습니다.

RESUME

SUSPEND로 일시 중지된 데이터베이스 미러링 세션을 재개합니다. 주 서버에서만 지정할 수 있습니다.

SAFETY { FULL | OFF }

트랜잭션 보안의 수준을 설정합니다. 동기 모드 또는 비동기 모드를 지정할 때 사용하며 자동 장애 조치 여부는 WITNESS 의 유무에 따라서 설정됩니다. 주 서버에서만 지정할 수 있습니다.

SUSPEND

데이터베이스 미러링 세션을 일시 중지합니다. 어느 파트너에서나 지정할 수 있습니다.

TIMEOUT *integer*

제한 시간(초)을 지정합니다. 제한 시간은 서버 인스턴스가 미러링 세션 내의 다른 인스턴스로부터 PING 메시지를 받기까지 기다리는 최대 시간으로 이 시간이 경과하면 해당 인스턴스의 연결이 끊어진 것으로 간주합니다. 이 옵션을 지정하지 않으면 기본적으로 제한 시간은 10초로 설정됩니다. 주 서버에서만 지정할 수 있습니다.

• 미러 모니터(WITNESS) 구문

```
ALTER DATABASE 〈데이터베이스 이름 지정〉
SET WITNESS=' TCP://〈모니터 서버 IP 주소〉:〈모니터 서버 미러링 포트 번호〉'
        | OFF
```

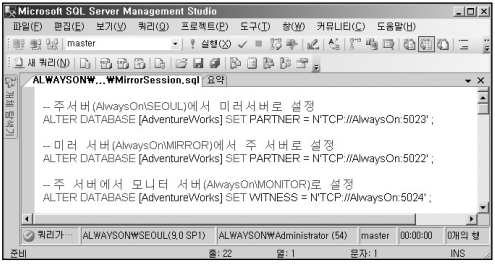
SET WITNESS=' TCP://**〈모니터 서버 IP 주소〉:〈모니터 서버 미러링 포트 번호〉**'

구성하고자 하는 모니터 서버 또는 설정을 변경하고자 하는 모니터 서버의 네트워크 주소를 지정합니다. 서버의 네트워크 주소는 IP 주소로 분해 가능한 주소를 지정합니다. 포트 번호는 미리 끝점으로 구성되어 있어야 합니다. SET WITNESS 문을 실행하려면 주 서버 인스턴스와 미러링 모니터 서버 인스턴스의 끝점에 대한 STATE가 STARTED로 설정되어 있어야 합니다. 또한 미러링 모니터 서버 인스턴스의 데이터베이스 미러링 끝점 ROLE이 WITNESS 또는 ALL로 설정되어 있어야 합니다.

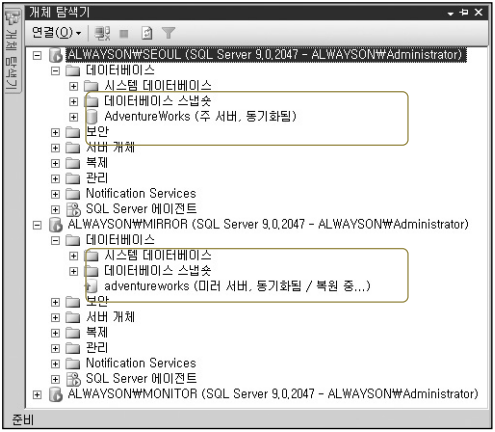
OFF

데이터베이스 미러링 세션에서 미러링 모니터를 제거합니다. 미러링 모니터를 OFF로 설정하면 자동 장애 조치가 해제됩니다.

■ 미러링 세션 설정 구문



데이터베이스 설정이 완료되었으면 **[개체 탐색기]**에서 주 데이터베이스와 미러 데이터베이스의 상태를 확인합니다. 현재 주 데이터베이스와 미러 데이터베이스가 동기화 되어 있음을 확인합니다.

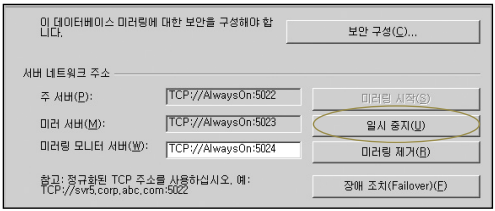


③ 데이터베이스 미러링 관리

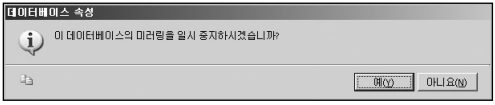
■ 일시 중지 또는 재개

데이터베이스 미러링 세션은 언제든지 일시 중지하고 나중에 재개할 수 있습니다. 일시 중지하더라도 세션 상태가 유지됩니다. 주 서버에서 다량의 트랜잭션으로 병목 상태가 발생하는 경우 주 서버의 성능을 높이기 위해서 세션을 일지 중지하는 것도 효과적입니다. 당연히, 세션을 일시 중지한 상태에서도 주 데이터베이스는 사용할 수 있으며 세션의 상태가 **[일시 중지됨(SUSPENDED)]**으로 설정됩니다.

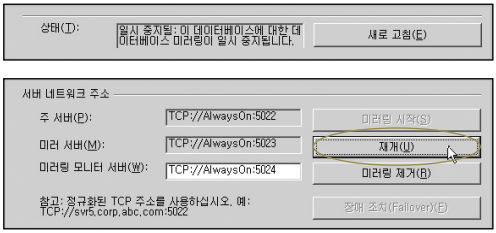
데이터베이스 미러링 세션이 일시 중지된 동안에는 트랜잭션 로그를 자를 수 없으므로 데이터베이스 미러링 세션을 너무 오래 일시 중지하면 트랜잭션 로그가 꽉 차서 데이터베이스를 사용할 수 없게 될 수 있으므로 주의해야 합니다.



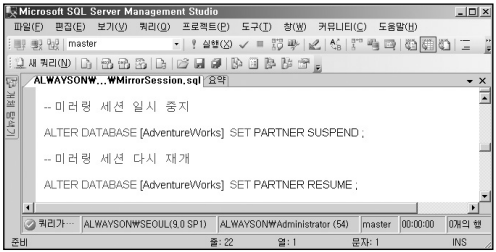
[데이터베이스 속성]의 **[미러링]** 페이지에서 **[일시 중지(U)]** 버튼을 누른뒤 다시한번 **[확인]** 버튼을 눌러서 미러링 세션을 일시 중지 합니다.



미러링 세션의 일시 중지 상태는 **[미러링]** 페이지의 하단 **[상태]** 부분에서 상태 정보를 확인할 수 있습니다.

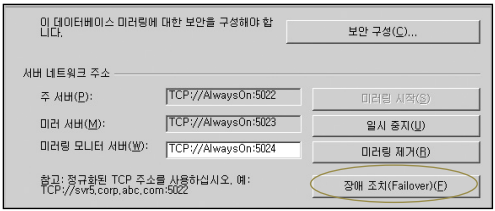


미러링 세션을 다시 재개하고자 하면 **[재개(U)]** 버튼을 누릅니다. 다음은 미러링 세션을 일시 중지하고 재개하는 T-SQL 구문입니다.

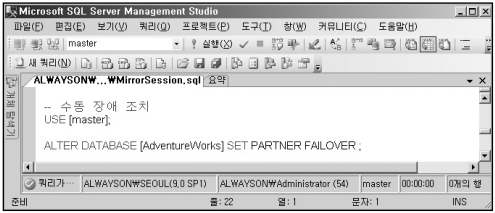


■ 수동 장애 조치 및 역할 변경

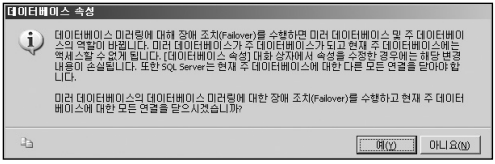
주 서버에서 하드웨어의 업그레이드나 서비스 팩의 적용 등 관리 작업이 필요한 경우에는 수종으로 장애 조치를 수행합니다. 장애 조치는 주 데이터베이스와 미러 데이터베이스의 역할을 변경하게 됩니다.

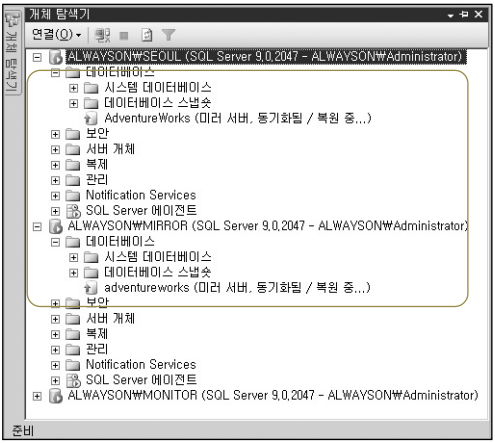


수동으로 장애 조치를 수행하기 위해서는 SSMS에서 **[데이터베이스 속성]**의 **[미러링]** 페이지에서 **[장애 조치(Failover)(F)]** 버튼을 누르거나 다음과 같이 ALTER DATABASE ~ SET PARTNER FAILOVER 구문을 수행합니다.

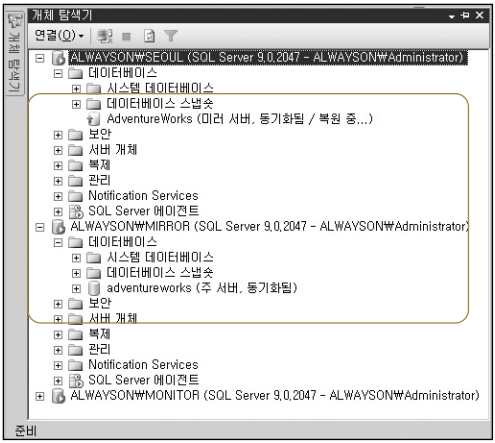


[데이터베이스 속성]의 **[미러링]** 페이지에서 **[장애 조치(Failover)(F)]** 버튼을 누르는 경우에는 경고 창을 통해서 장애 조치 작업과 관련해서 경고 문구를 보여줍니다.





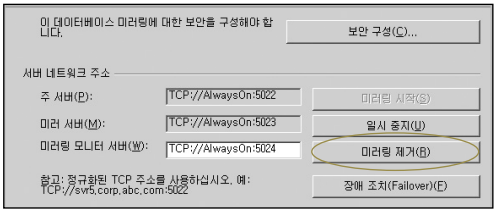
주 데이터베이스와 미러 데이터베이스는 장애 조치 작업이 진행되면 잠깐 동안 모두 **[복원 중]**으로 사용할 수 없게 되며 곧 이어서 주 데이터베이스와 미러 데이터베이스의 역할을 변경하게 됩니다.



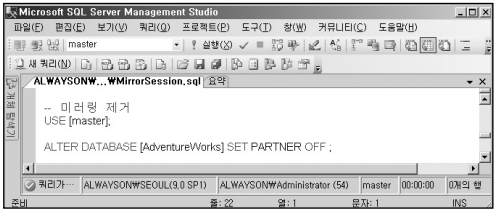
역할이 변경된 상태에서 다시 장애 복구(Failback)를 하기 위해서는 현재 주 데이터베이스에서 개체 탐색기를 사용하든지 ALTER DATABASE ~ SET PARTNER FAILOVER 구문을 수행해야 합니다.

■ 미러링 제거

미러링을 제거하게 되면 파트너 간의 관계 및 각 파트너와 미러링 모니터 서버 간의 관계가 영구적으로 해제됩니다. 또한, 세션을 일시 중지하는 경우와 달리 미러링 세션에 대한 정보가 삭제됩니다. 미러링은 주 데이터베이스와 미러 데이터베이스에서 모두 제거됩니다. 그렇지만 주 데이터베이스와 미러 데이터베이스는 제거 당시의 상태 그대로 존재하게 됩니다. **[복원 중]** 상태의 데이터베이스는 삭제하거나 RECOVERY 옵션을 사용하는 **[복원하지 않고 복구하기]** 방법으로 복구할 수 있습니다.



[데이터베이스 속성]의 **[미러링]** 페이지에서 **[미러링 제거(R)]** 버튼을 누르거나 ALTER DATABASE ~ SET PARTNER OFF 구문을 수행합니다.



미러링을 제거한 뒤 다시 설정하고자 하는 경우에 미러링을 중지한 후에 로그 백업을 수행했다면 반드시 이 로그 백업을 미러 데이터베이스에 NORECOVERY 옵션으로 복원해야 합니다. 만약, 트랜잭션 로그가 백업되지 않고 잘려서 미러 서버에 중지된 기간 동안의 트랜잭션을 적용할 수 없으면 미러 데이터베이스 전체 백업 복원부터 새로 구성해야 합니다.

④ 주 서버의 로그인 계정을 미러 서버로 전송 하기

데이터베이스 미러링의 단위는 사용자 데이터베이스라는 것은 **[3) 데이터베이스 미러링 구성시 고려 사항]** 부분에서 이미 살펴 보았습니다. 이것은 시스템 데이터베이스는 데이터베이스 미러링을 구성할 수 없다는 얘기이므로 데이터베이스 미러링을 마무리 하기 위해서는 주 서버의 시스템 데이터베이스에서 필요한 설정을 보조 서버로 가져와야 합니다.

가장 중요한 것이 로그인 계정입니다. SQL Sever는 서버에 연결하기 위해서 로그인 계정이 필요하고 각 데이터베이스에 연결하기 위해서는 데이터베이스 사용자 계정이 별도로 필요하다는 것은 SQL Server의 상식이므로 잘 알고 있을 겁니다. 이러한 로그인 계정과 각 데이터베이스 사용자 계정은 SID라고 하는 것으로 연결되어 있습니다. 따라서 단순히 데이터베이스 사용자 계정과 동일한 이름으로 로그인 계정을 생성한다고 해서 두 계정이 연결되는 것은 아님을 반드시 알아두기 바랍니다. 로그인 계정의 정보는 sys.server_principals 시스템 카탈로그 뷰에서, 데이터베이스 사용자 계정은 각 데이터베이스의 sys.database_principals 시스템 카탈로그 뷰에서 확인할 수 있습니다. 이전 버전에서는 이러한 정보는 시스템 테이블에 저장되어 있고, 시스템 테이블은 관리자가 수정이 가능하였으므로 조인 구문 등을 통해서 수정할 수 있었지만 SQL Server 2005에서는 msdb의 일부 시스템 테이블을 제외하고는 관리자가 수정할 수 없습니다.

[주의]
SQL Server 2005 인스턴스 간에 로그인 및 암호를 전송하는 것은 인스턴스 단위의 고가용성 솔루션인 장애 조치 클러스터링을 제외하고 데이터베이스 미러링, 로그 전달, 피어 투 피어 복제에서 모두 구현해야 합니다.

다음에서 소개하는 sp_hexadecimal 시스템 저장 프로시저와 sp_help_revlogin 시스템 저장 프로시저를 주 서버에 생성하고 sp_help_revlogin 시스템 저장 프로시저를 실행한 뒤 그 결과물로 보조 서버에서 로그인 계정을 생성하는 방법이 가장 효율적입니다.

[참고]
SQL Server 2005 인스턴스 간에 로그인 및 암호를 전송하는 방법은 다음 마이크로소프트 문서에 확인하기 바랍니다.
http://www.support.microsoft.com/kb/918992/

• sp_hexadecimal 시스템 저장 프로시저 생성 구문

```
USE master
GO
IF OBJECT_ID ('sp_hexadecimal') IS NOT NULL
  DROP PROCEDURE sp_hexadecimal
GO
CREATE PROCEDURE sp_hexadecimal
    @binvalue varbinary(256),
    @hexvalue varchar (514) OUTPUT
AS
DECLARE @charvalue varchar (514)
DECLARE @i int
DECLARE @length int
DECLARE @hexstring char(16)
SELECT @charvalue = '0x'
SELECT @i = 1
SELECT @length = DATALENGTH (@binvalue)
SELECT @hexstring = '0123456789ABCDEF'
WHILE (@i <= @length)
BEGIN
  DECLARE @tempint int
  DECLARE @firstint int
  DECLARE @secondint int
  SELECT @tempint = CONVERT(int, SUBSTRING(@binvalue,@i,1))
  SELECT @firstint = FLOOR(@tempint/16)
  SELECT @secondint = @tempint - (@firstint*16)
  SELECT @charvalue = @charvalue +
```

```
    SUBSTRING(@hexstring, @firstint+1, 1) +
    SUBSTRING(@hexstring, @secondint+1, 1)
  SELECT @i = @i + 1
END
SELECT @hexvalue = @charvalue
GO
```

- sp_help_revlogin 시스템 저장 프로시저 생성 구문

```
IF OBJECT_ID ('sp_help_revlogin') IS NOT NULL
  DROP PROCEDURE sp_help_revlogin
GO
CREATE PROCEDURE sp_help_revlogin @login_name sysname = NULL AS
DECLARE @name sysname
DECLARE @type varchar (1)
DECLARE @hasaccess int
DECLARE @denylogin int
DECLARE @is_disabled int
DECLARE @PWD_varbinary  varbinary (256)
DECLARE @PWD_string  varchar (514)
DECLARE @SID_varbinary varbinary (85)
DECLARE @SID_string varchar (514)
DECLARE @tmpstr  varchar (1024)
DECLARE @is_policy_checked varchar (3)
DECLARE @is_expiration_checked varchar (3)
DECLARE @defaultdb sysname
```

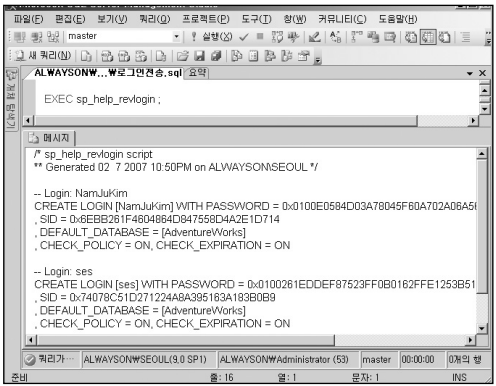
```
IF (@login_name IS NULL)
DECLARE login_curs CURSOR FOR
SELECT p.sid, p.name, p.type, p.is_disabled, p.default_database_name,
l.hasaccess, l.denylogin FROM
sys.server_principals p LEFT JOIN sys.syslogins l
    ON ( l.name = p.name ) WHERE p.type IN ( 'S', 'G', 'U' ) AND p.name <> 'sa'
ELSE
DECLARE login_curs CURSOR FOR
SELECT p.sid, p.name, p.type, p.is_disabled, p.default_database_name,
l.hasaccess, l.denylogin FROM sys.server_principals p LEFT JOIN sys.syslogins l
      ON ( l.name = p.name ) WHERE p.type IN ( 'S', 'G', 'U' ) AND p.name =
@login_name
OPEN login_curs

FETCH NEXT FROM login_curs INTO
@SID_varbinary, @name, @type, @is_disabled, @defaultdb, @hasaccess,
@denylogin
IF (@@fetch_status = -1)
BEGIN
  PRINT 'No login(s) found.'
  CLOSE login_curs
  DEALLOCATE login_curs
  RETURN -1
END
SET @tmpstr = '/* sp_help_revlogin script '
PRINT @tmpstr
SET @tmpstr = '** Generated ' + CONVERT (varchar, GETDATE()) + ' on ' +
```

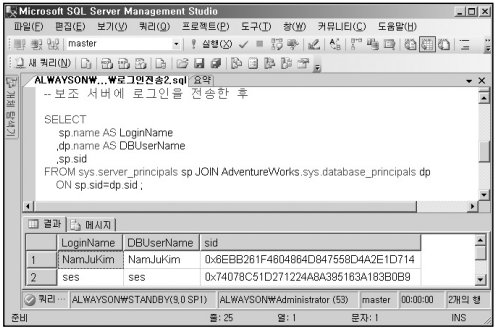
```
@@SERVERNAME + ' */'
PRINT @tmpstr
PRINT ''
WHILE (@@fetch_status <> -1)
BEGIN
 IF (@@fetch_status <> -2)
 BEGIN
  PRINT ''
  SET @tmpstr = '-- Login: ' + @name
  PRINT @tmpstr
  IF (@type IN ( 'G', 'U'))
  BEGIN -- NT authenticated account/group
   SET @tmpstr = 'CREATE LOGIN ' + QUOTENAME( @name ) + ' FROM
WINDOWS WITH DEFAULT_DATABASE = [' + @defaultdb + ']'
 END
ELSE BEGIN -- SQL Server authentication
  -- obtain password and sid
  SET @PWD_varbinary = CAST( LOGINPROPERTY( @name, 'PasswordHash' )
AS varbinary (256) )
  EXEC sp_hexadecimal @PWD_varbinary, @PWD_string OUT
  EXEC sp_hexadecimal @SID_varbinary,@SID_string OUT
   -- obtain password policy state
  SELECT
@is_policy_checked = CASE is_policy_checked WHEN 1 THEN 'ON' WHEN 0
THEN 'OFF' ELSE NULL END
FROM sys.sql_logins WHERE name = @name
```

```
SELECT @is_expiration_checked = CASE is_expiration_checked WHEN 1 THEN
'ON'
WHEN 0 THEN 'OFF' ELSE NULL END
FROM sys.sql_logins WHERE name = @name
  SET @tmpstr = 'CREATE LOGIN ' + QUOTENAME( @name ) + ' WITH
PASSWORD = ' + @PWD_string + ' HASHED, SID = ' + @SID_string + ',
DEFAULT_DATABASE = [' + @defaultdb + ']'
    IF ( @is_policy_checked IS NOT NULL )
    BEGIN
      SET @tmpstr = @tmpstr + ', CHECK_POLICY = ' + @is_policy_checked
    END
    IF ( @is_expiration_checked IS NOT NULL )
    BEGIN
          SET @tmpstr = @tmpstr + ', CHECK_EXPIRATION = ' +
@is_expiration_checked
    END
  END
  IF (@denylogin = 1)
  BEGIN -- login is denied access
     SET @tmpstr = @tmpstr + '; DENY CONNECT SQL TO ' + QUOTENAME(
@name )
  END
  ELSE IF (@hasaccess = 0)
  BEGIN -- login exists but does not have access
    SET @tmpstr = @tmpstr + '; REVOKE CONNECT SQL TO ' + QUOTENAME(
@name )
  END
```

```
    IF (@is_disabled = 1)
    BEGIN -- login is disabled
        SET @tmpstr = @tmpstr + '; ALTER LOGIN ' + QUOTENAME( @name ) + '
DISABLE'
    END
    PRINT @tmpstr
  END
FETCH NEXT FROM login_curs INTO
@SID_varbinary, @name, @type, @is_disabled, @defaultdb, @hasaccess,
@denylogin
  END
CLOSE login_curs
DEALLOCATE login_curs
RETURN 0
GO
```



주 서버에서 sp_hexadecimal 시스템 저장 프로시저와 sp_help_revlogin 시스템 저장 프로시저를 생성한 뒤 sp_help_revlogin 시스템 저장 프로시저를 수행하면 다음과 같이 주 서버의 로그인 계정의 SID와 암호를 16진수 값으로 표시하여 추출합니다.



sp_helpl_revlogin 시스템 저장 프로시저의 결과 스크립트를 미러 서버에서 수행하고 로그인 계정이 올바르게 생성된 것을 다음 그림과 같이 확인합니다.

7) 데이터베이스 미러링 모니터 작업

미러링 세션 중에 미러된 데이터베이스를 모니터링하여 데이터 흐름 여부와 상태를 확인할 수 있습니다. 서버 인스턴스에 있는 하나 이상의 미러된 데이터베이스에 대한 모니터링을 설정하고 관리하려면 데이터베이스 미러링 모니터나 dbmmonitor 시스템 저장 프로시저를 사용합니다. 이 이외에도 Windows 시스템 성능 모니터의 카운터를 통해서는 미러링과 관련한 성능 상태를 확인할 수 있습니다.

데이터베이스 미러링 모니터링 작업은 데이터베이스 미러링 모니터와 별개로 백그라운드에서 작동합니다. SQL Server 에이전트는 기본적으로 1분마다 데이터베이스 미러링 모니터 작업을 호출하고 작업에서는 미러링 상태를 업데이트하는 저장 프로시저를 호출합니다. SSMS(SQL Server Management Studio)를 사용하여 미러링 세션을 시작하는 경우 데이터베이스 미러링 모니터 작업은 자동으로 생성됩니다. 그러나 ALTER DATABASE~ SET PARTNER 구문만 사용해서 미러링을 시작한 경우에는 데이터베이스 미러링 모니터 작업을 별도로 생성해야 합니다.

데이터베이스 미러링 모니터나 sp_dbmmonitorresults 시스템 저장 프로시저를 사용해서 다음과 같은 작업을 수행할 수 있습니다.

- 미러링이 동작하고 있는 지 확인합니다.
- 미러 데이터베이스가 주 데이터베이스와 동기화되고 있는 지 여부를 확인합니다.
- 성능 우선(비동기) 모드에서 주 서버 인스턴스를 사용할 수 없는 경우 손실된 데이터 양을 확인합니다.
- 현재 성능을 이전 성능과 비교합니다.
- 미러링 파트너 간의 데이터 흐름 감소 원인을 찾아서 문제를 해결합니다.
- 주요 성능 메트릭에 경고 임계값을 설정합니다.

① 데이터베이스 미러링 모니터 작업

SSMS(SQL Server Management Studio)를 사용하여 미러링 세션을 시작하는 경우 데이터베이스 미러링 모니터 작업은 자동으로 생성되지만 ALTER DATABASE~ SET PARTNER 구문만 사용해서 미러링을 시작한 경우에는 자동으로 생성되지 않으므로 sp_dbmmonitor addmonitoring 시스템 저장 프로시저를 사용해서 데이터베이스 미러링 모니터 작업을 별도로 생성합니다.

```
sp_dbmmonitoraddmonitoring [ 〈업데이트 기간(분)〉 ]
```

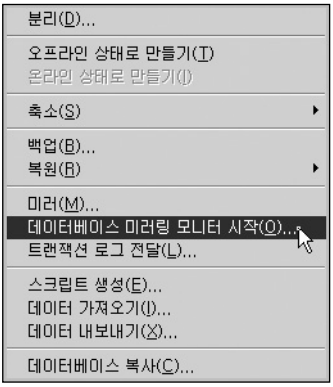
이미 생성된 미러링 모니터 작업의 일정을 변경하고자 하는 경우에는 SQL Server 에이전트에서 해당 작업의 일정을 변경하거나 sp_dbmmonitorchangemonitoring 시스템 저장 프로시저를 사용합니다.

```
sp_dbmmonitorchangemonitoring 1 , 〈새 업데이트 기간 (분)〉
```

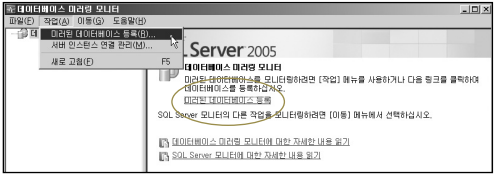
생성된 데이터베이스 미러링 모니터 작업을 삭제하고자 하는 경우에는 sp_dbmmonitor dropmonitoring 시스템 저장 프로시저를 사용하거나 SQL Server 에이전트의 작업에서 삭제합니다.

② 데이터베이스 미러링 모니터

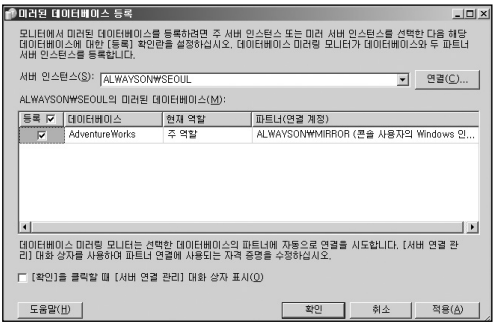
데이터베이스 미러링 모니터는 시스템 관리자가 데이터베이스 미러링 상태를 확인하고 여러 주요 성능 메트릭에 경고 임계값을 구성할 수 있도록 하는 그래픽 사용자 인터페이스 도구입니다. 단, SQL Server 2005 서비스 팩1 이상을 설치해야 사용할 수 있습니다.

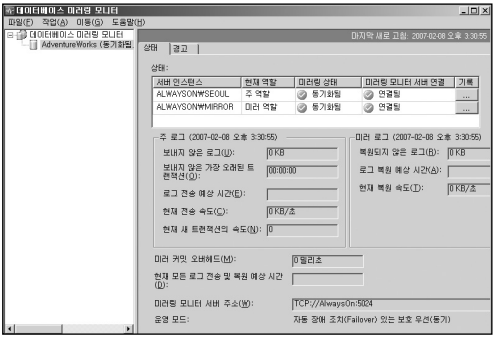


데이터베이스 미러링 모니터는 데이터베이스를 선택하고 오른 쪽 버튼을 누른 뒤 **[작업(T)]**에서 **[데이터베이스 미러링 모니터 시작(O)]**을 선택합니다.



이어서 **[데이터베이스 미러링 모니터]**의 **[작업(A)]** 메뉴나 오른쪽 창에서 **[미러된 데이터베이스 등록]**을 선택한 뒤 미러된 데이터베이스를 모니터에 등록합니다.

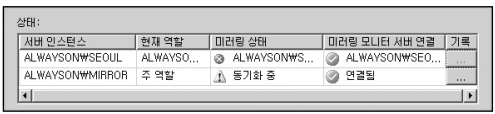


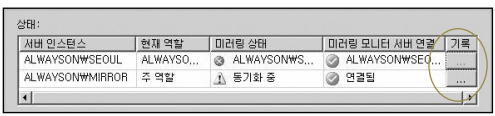


[데이터베이스 미러링 모니터]의 오른쪽은 **[상태]** 페이지와 **[경고]** 페이지로 구성되며 **[상태]** 페이지에서는 서버의 역할과 연결 및 미러링 상태 정보, 트랜잭션 로그의 전송과 관련된 정보가 표시되며, **[경고]** 페이지에서는 주요 성능 메트릭에 경고 임계값을 설정합니다.

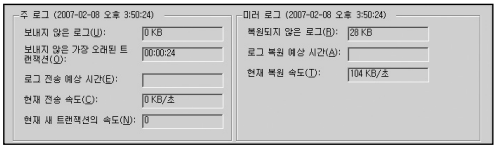
[상태] 페이지

데이터베이스 미러링 모니터의 상태 페이지에는 파트너에 대한 설명과 미러링 세션의 상태가 표시되어 서버의 현재 역할 및 미러링 상태 정보, 미러 모니터와 각 파트너와의 연결 상태를 확인할 수 있습니다.





오른 쪽 끝 부분의 **[기록]** 버튼을 누르면 주 서버와 미러 서버간의 트랜잭션 로그의 전송과 관련된 기록이 표시됩니다.



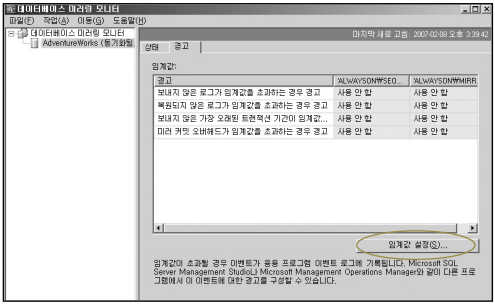
[주 로그] 부분에는 주 서버의 미러 서버로 보내지 못한 Send Queue에서 대기 중인 로그의 양(KB)과 아직 전송되지 않은 로그를 전송하는 데 예상되는 시간 등을 표시하고 **[미러 로그]** 부분에는 미러 서버에서 아직 복원하지 못한 Redo Queue에서 대기 중인 로그의 양(KB)과 복원에 예상되는 시간 등을 표시합니다

[상태] 페이지 밑 부분에서는 주 서버와 미러 서버가 동기화 되지 못한 경우 트랜잭션 로그 전송와 복원에 소요되는 총 예상 시간 등을 표시합니다.

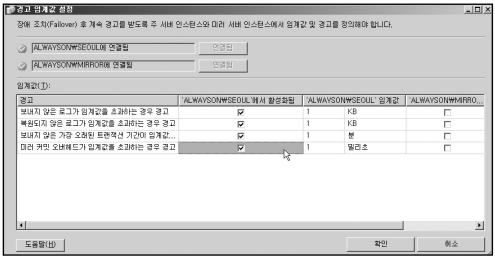
[경고] 페이지

[경고] 페이지에서 성능 메트릭에 대해 경고 임계값을 정의하면 상태 테이블이 업데이트될 때마다 최신 값을 평가합니다. 이 값이 임계값에 도달하면 sp_dbmmonitorupdate 시스템 저장 프로시저가 메트릭에 대해 경고 이벤트를 생성하고 해당 이벤트를 Microsoft Windows 이벤트 로그에 기록합니다. SSMS나 Microsoft Management Operations Manager와 같은 다른 프로그램에서 이 이벤트에 대해서 경고를 구성할 수 도 있습니다. 임계값을 지정하는 경고는 다음과 같습니다.

- 보내지 않은 로그가 임계값을 초과하는 경우 (KB)
- 복원되지 않은 로그가 임계값을 초과하는 경우 (KB)
- 보내지 않은 가장 오래된 트랜잭션 기간이 임계값을 초과하는 경우 (분)
- 미러 커밋 오버헤드가 임계값을 초과하는 경우 (밀리 초)

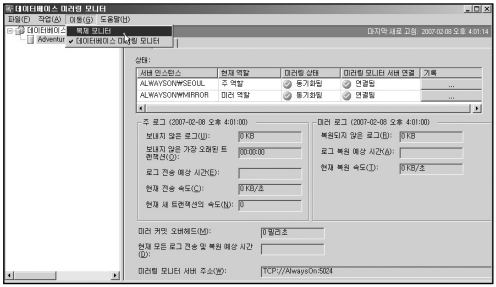


[임계값] 설정을 위해서는 **[경고]** 페이지 하단의 **[임계값 설정(S)]** 버튼을 누릅니다.



[경고 임계값 설정] 페이지에서는 사용하고자 하는 경고 별로 임계값을 지정합니다.

• 데이터베이스 미러링 모니터는 복제 모니터와 상호간에 이동이 가능합니다.

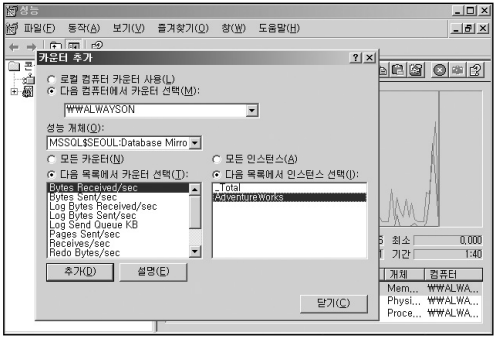


데이터베이스 미러링 모니터는 **[이동(G)]** 메뉴를 통해서 복제 모니터로 전환하여 사용할 수도 있습니다.

③ 데이터베이스 미러링 성능 카운터

Windows 시스템 성능 모니터에는 데이터베이스 미러링과 관련한 SQLServer:Database Mirroring 성능 모니터 개체를 제공합니다.

SQLServer:Database Mirroring 성능 개체는 다음과 같은 성능 카운터가 있습니다.



성능 카운터	설 명
Bytes Received/Sec	초당 받은 바이트 수입니다.
Bytes Sent/Sec	초당 보낸 바이트 수입니다.
Log Bytes Received/Sec	초당 받은 로그 바이트 수입니다.
Log Bytes Sent/Sec	초당 보낸 로그 바이트 수입니다.
Log Send Queue KB	미러 서버에 아직 보내지 않은 총 로그 킬로 바이트 수입니다.
Redo Queue KB	현재 미러 데이터베이스의 다시 실행 항목 중 확정된 로그의 총 킬로 바이트 수입니다.
Page Sent/Sec	초당 보낸 페이지 수입니다.
Sends/Sec	초당 보낸 미러링 페이지 수입니다.
Receives/Sec	초당 받은 미러링 페이지 수입니다.
Redo Bytes/Sec	초당 미러 데이터베이스에서 다시 실행한 로그 바이트 수입니다.
Transaction Delay	종료되지 않은 커밋 승인을 기다리는 지연 시간입니다.

〈 데이터베이스 미러링 성능 모니터 카운터〉

Transaction Delay 카운터를 사용하면 데이터베이스 미러링이 주 서버의 성능에 영향을 주는지 여부를 알 수 있고 Redo Queue와 Log Send Queue 카운터를 사용하면 미러 데이터베이스가 주 데이터베이스와 동기화가 제대로 이루어지는지 알 수 있습니다. Log Bytes Sent/sec 카운터를 사용하면 초당 보낸 로그의 양을 모니터링할 수 있습니다.

8) 데이터베이스 미러링 성능 고려 사항

데이터베이스 미러링은 가용성을 향상시키는 훌륭한 솔루션이지만 성능 문제에 대해서 충분히 고려해야 합니다. 특히 동기 모드를 구성하고자 하는 경우에는 성능에 대한 고려는 필수적입니다.

먼저 자동 장애 조치(Failover) 프로세스를 살펴보고 성능에 대해서 고려해야 하는 사항을 점검해 보겠습니다.

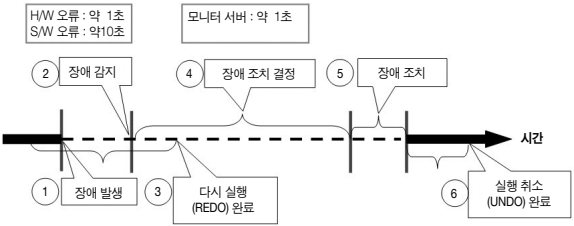
① 데이터베이스 미러링 자동 장애조치(Failover) 프로세스

데이터베이스 미러링의 자동 장애조치(Failover)는 반드시 모니터(witness) 서버가 구성되어 있어야 하며 프로세스는 다음과 같은 단계로 동작합니다.

1 단계 : 장애 발생
2 단계 : 미러 모니터서버에서 장애를 감지하고 주 서버가 실행 중이면 주 데이터베이스의 상태를 DISCONNECTED로 변경 및 주 데이터베이스로부터 모든 클라이언트의 연결 해제
3 단계 : 미러 데이터베이스의 다시 실행(REDO) 수행 완료
4 단계 : 장애조치 여부 결정
5 단계 : 장애조치를 수행하고 장애 조치가 완료되면 기존 미러 데이터베이스가 가용한 상태로 유지
6 단계 : 실행 취소(UNDO) 수행

> ***[주의]***
> *5 단계인 장애 조치(Failover)를 완료한 뒤 미러 데이터베이스가 주 데이터베이스 역할을 즉시 수행할 수 있는 것은 SQL Server 2005 Enterprise Edition에서 빠른 복구 기능을 사용해서만 가능합니다. Enterprise Edition이 아닌 경우에는 장애 조치(Failover)를 완료하고 실행 취소(Rollback) 단계를 모두 완료해야 데이터베이스가 사용 가능한 상태가 됩니다.*

각 단계를 자세히 살펴보면 다음과 같습니다.



- 1 단계 : 주 서버의 데이터베이스에 장애가 발생합니다. 장애는 SQL Server 인스턴스의 문제이거나 주 데이터베이스의 장애 또는 하드 디스크와 네트워크 등 하드웨어 장애가 모두 포함됩니다.

- 2 단계 : 장애가 발생하면 모니터(witness) 서버와 미러 서버가 장애를 감지합니다. 하드웨어 장애는 약 1초 내외로 즉시 감지되며 소프트웨어 장애는 기본 설정인 10초 동안 주 데이터베이스가 PING 메시지에 응답이 없으면 감지합니다. 10 초의 시간은 ALTER DATABASE SET PARTNER TIMEOUT 구문으로 변경이 가능하지만 기본 설정인 10 초보다 작게 잡는 것은 성능상의 문제로 권장하지 않습니다.

- 3 단계 : 장애 조치(Failover)와 관계 없이 미러 데이터베이스에서 주 서버로부터 전송된 트랜잭션을 반영하는 REDO 단계를 수행하여 완료하여야 합니다. 이 단계가 완료되어야 모니터 서버가 장애조치 여부를 결정합니다.

- 4 단계 : 미러 데이터베이스에서 다시 실행 단계가 완료되면 장애 조치(Failover) 여부를 결정합니다. 통상 약 1초 내외의 시간이 소요되며 만약 이 시점 이전에 주 데이터베이스가 가용한 상태가 되면 장애 조치(Failover)를 수행하지 않습니다.

- 5 단계 : 모니터 서버가 자동 장애 조치(Failover)를 결정하면 장애 조치(Failover)를 수행하고 장애 조치(Failover)가 완료되는 즉시 기존의 미러 데이터베이스는 읽기/쓰기 작업이 모두 가능한 상태의 주 데이터베이스 기능을 수행합니다.

- 6 단계 : 기존 주 서버에서 커밋(Commit)하지 못한 트랜잭션에 대해서 실행 취소(Rollback) 작업을 수행합니다.

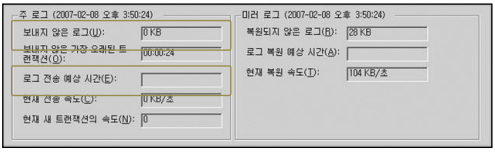
② 성능 고려 사항

본 가이드의 [7] **데이터베이스 미러링 모니터 작업]** 부분에서 데이터베이스 미러링을 모니터하는 방법과 도구에 대해서 살펴보았습니다. 성능과 관련해서 주의해서 살펴보아야 하는 부분은 주 서버에서 미러 서버로 보내지 못한 트랜잭션 량과 미러 서버에서 아직 반영하지 못한 트랜잭션 량 그리고 동기 모드에서 주 서버에서 발생한 트랜잭션이 커밋되는데 소요되는 시간입니다.

■ 주 서버에서 미러 서버로 보내지 못한 트랜잭션 량

동기 모드에서 주 서버에서 미러 서버로 보내지 못한 트랜잭션의 량은 성능상 매우 중요합니다. 이것은 동기 모드에서 주 서버에서 발생한 트랜잭션이 커밋되는 데 소요되는 시간을 좌우하기 때문입니다.

이러한 사항을 확인하기 위해서는 미러 모니터의 **[상태]** 페이지의 **[주 로그]** 부분을 확인하거나 성능 모니터의 Log Send Queue KB, Transaction Delay 성능 카운터를 참조합니다.



미러 모니터의 **[보내지 않은 로그(U)]**는 성능 카운터의 Log Send Queue KB의 값과 동일합니다. 이 값과 **[로그 전송 예상 시간(E)]**의 값은 동기 모드에서 성능 카운터의 Transaction Delay의 값과 함께 주 서버에서 트랜잭션이 커밋하는 데 소요되는 지연 시간을 확인할 수 있습니다. **[로그 전송 예상 시간(E)]**의 값은 장애 조치(Failover) 시 소요되는 시간에도 영향을 주게 됩니다. 자동으로 장애 조치(Failover)를 수행하기 위해서는 반드시 주 데이터베이스의 트랜잭션이 모두 미러 데이터베이스에 반영되어야만 합니다.

비동기 모드에서 성능 카운터 Transaction Delay의 값은 항상 0이 됩니다. 비동기 모드에서 주 서버에서 미러 서버로 보내지 못한 트랜잭션의 량은 장애 조치(Failover) 하는 시간에 영향을 주게 됩니다.

■ 미러 데이터베이스에서 반영하지 못한 트랜잭션 량

동기 모드에서 미러 데이터베이스에서 반영하지 못하는 트랜잭션의 량은 앞에서 살펴본 바와 같이 주 서버에서 수행된 트랜잭션이 커밋하는 시간을 지연하게 되며 비동기 모드에서는 주 서버에서 보내지 못한 트랜잭션 량과 같이 장애 조치(Failover)하는 시간을 지연하게 됩니다.

따라서 미러 모니터나 성능 모니터를 통해서 수시로 확인하고 조치해야 합니다.

■ 성능 고려 사항

데이터베이스 미러링의 성능 최적화를 위해서 고려해야 하는 사항은 다음과 같습니다.

- 미러링 모드
- 네트워크 라운드 트립 시간(RTT) :
- 주 서버의 트랜잭션 처리량(Database:Log Bytes Fushed /sec)

먼저 비동기 모드는 트랜잭션 로그의 전송이 주 데이터베이스의 트랜잭션 커밋과 관련이 없이 독립적으로 수행되므로 주 서버의 성능에 큰 영향을 미치지 않습니다. 다만 주 서버에서 Log Send Queue를 유지해야 하므로 트랜잭션 로그가 포함된 디스크의 IO가 많이 발생하고 주 서버의 CPU 사용량이 다소 증가하게 됩니다.

동기 모드의 경우는 네트워크의 RTT(Round-Trip Time)와 트랜잭션의 처리량에 따라서 주 서버에 성능상의 문제를 유발할 수 있습니다. 따라서 WAN 환경이나 인덱스 다시 작성이나 대량의 데이터 로드 작업 등 트랜잭션 처리량이 많은 작업이 빈번하게 수행되는 시스템에서는 데이터베이스 미러링이 주 데이터베이스의 성능에 미치는 영향을 면밀히 검토해야 합니다. 따라서 미러링 모드를 결정하기 전에 데이터의 중요성과 네트워크의 상태, 대량 작업의 빈도와 크기 등을 고려합니다.

9) 데이터베이스 스냅숏을 통한 미러 데이터베이스 사용

데이터베이스 스냅숏은 SQL Server 2005의 새로운 기능으로 Enterprise Edition에서만 사용 가능한 기능입니다. 데이터베이스 스냅숏은 원본 데이터베이스에 대한 읽기 전용 정적 뷰라고 할 수 있습니다. 데이터베이스 스냅숏은 생성된 시점의 이미지를 보유하게 되는 데 수정이 발생하지 않은 데이터는 원본에 유지하고 변경이 발생한 데이터 페이지는 데이터베이스 스냅숏으로 이동해서 저장됩니다. 즉 원본에서 변경이 발생하지 않았다면 데이터베이스 스냅숏이 사용하는 물리적인 저장 공간은 불과 수백 KB 수준에 지나지 않습니다. 데이터베이스 스냅숏은 하나의 원본 데이터베이스에 대해서 여러 개를 생성할 수 있으며, 원본 데이터베이스와 동일한 서버 인스턴스에 생성해야 합니다.

데이터베이스 미러링으로 설정된 미러 데이터베이스는 복원 중인 상태를 유지하므로 데이터에 액세스할 수 없습니다. 그렇지만 미러 데이터베이스 기반의 데이터베이스 스냅숏을 생성하면 읽기 전용 작업이 가능해 집니다. 미러 데이터베이스를 보고 용도로 사용하려면 미러 데이터베이스에서 데이터베이스 스냅숏을 만들고 클라이언트 연결 요청을 가장 최근의

스냅숏으로 지정할 수 있습니다. 데이터베이스 스냅숏은 스냅숏이 만들어진 시점에 존재하던 원본 데이터베이스의 정적, 읽기 전용, 트랜잭션 일치 스냅숏입니다. 따라서 주 데이터베이스의 읽기 작업에 대한 부하를 분산하는 용도로 사용할 수 있습니다. 단, 미러 데이터베이스에서 데이터베이스 스냅숏을 만들려면 데이터베이스가 동기화된 미러링 상태여야 합니다.

> ***[주의]***
> *데이터베이스 미러링은 주 서버의 부하를 고려하여야 합니다. 동기(보호 우선) 모드인 경우에 미러 데이터베이스를 기반으로 데이터베이스 스냅숏을 생성하는 것은 추가적인 부하를 야기합니다. 부하에 대한 충분한 검증을 한 뒤 사용 여부를 결정합니다.*

또한, 미러 서버와 주 서버의 구성에 따라 미러 데이터베이스에 있는 데이터베이스 스냅숏의 수가 너무 많으면 주 데이터베이스의 성능이 느려질 수 있으므로 주 데이터베이스의 성능을 고려하여 데이터베이스 스냅숏을 만들고 유지해야 합니다.

미러 데이터베이스에 생성된 데이터베이스 스냅숏을 사용하던 중 역할이 전환되면 일시적으로 사용자와의 연결을 끊지만 사용자는 장애 조치(Failover) 후에도 스냅숏을 계속 사용할 수 있습니다.

① 데이터베이스 스냅숏의 사용 목적

이와 같은 데이터베이스 스냅숏은 다음과 같은 용도로 사용하게 됩니다.

- 보고서 생성을 위해서 특정 시점 데이터를 유지 관리합니다.
- 가용성 목적으로 유지 관리 중인 미러 데이터베이스를 활용하여 보고 작업 부하를 분산합니다.
- 사용자의 실수나 관리자의 실수 등으로부터 데이터 보호합니다.

다음에서는 좀 더 자세하게 데이터베이스 스냅숏의 사용 목적에 대해서 좀 더 자세하게 살펴봅니다.

먼저, 보고서 생성을 위해서 특정 시점 데이터를 유지 관리하는 기능입니다. 데이터베이스 스냅숏은 데이터베이스의 정적인 보기를 제공하므로 특정 시점의 데이터베 대한 액세스 기간을 연장할 수 있습니다. 예를 들어서 회계 분기와 같은 지정된 기간이 끝나는 시점의 데이터베이스 스냅숏을 만들어 두면 향후 보고 시 유용하게 사용할 수 있습니다. 또한 이를 토대로 만든 반기 또는 기말 보고서를 실행할 수 있습니다.

또한, 데이터베이스 스냅숏은 미러 데이터베이스에 액세스 할 수 있는 기능을 활용하여 가용성 목적으로 유지 관리 중인 미러 데이터베이스를 활용하여 보고 작업 부하를 분산할 수 있습니다. 이러한 경우 원본인 주 데이터베이스의 리소스를 절약할 수 있어서 주 서버의 부하를 분산할 수 있습니다.

사용자 오류로부터 데이터를 보호하는 기능은 정기적으로 데이터베이스 스냅숏을 만들어 두면 테이블 삭제나 WHERE 절이 없는 잘못된 삭제나 수정으로부터 오류를 효율적으로 해결할 수 있습니다. 또한 관리자 오류로부터 데이터를 보호하는 기능은 대량의 업데이트와 같은 주요 작업을 수행하기 전에 데이터베이스 스냅숏을 만들어 두면 오류가 발생해서 작업 이전으로 복구하고자 할 때 유용하게 복구할 수 있습니다.

② 데이터베이스 스냅숏의 동작 구조

데이터베이스 스냅숏을 사용하기 위해서 가능한 데이터베이스의 동작 구조를 이해하고 작업하는 것이 효율적입니다. 데이터베이스 스냅숏은 데이터 페이지와 익스텐트 수준에서 동작합니다.원본 페이지를 수정하게 되면 해당 트랜잭션이 원본을 수정하기 전에 데이터베이스 스냅숏으로 해당 페이지를 복사합니다. 이와 같은 작업을 **[쓰기 시 복사 작업(Copy on Write)]** 작업이라고 합니다. 즉, 원본 데이터베이스에서 변경이 발생하더라도 데이터베이스 스냅숏 쪽에서는 수정되기 전의 페이지를 복사했기 때문에 항상 데이터베이스 스냅숏을 만들었을 때 시점의 데이터에 액세스할 수 있는 것입니다. 복사된 페이지를 저장하기 위해서는 하나 이상의 스파스(Sparse) 파일이라고 하는 파일을 사용합니다.

데이터베이스 스냅숏을 생성할 때의 스파스 파일은 기본적으로 사용자 데이터가 없는 빈 파일이며 실제 디스크에 공간이 할당되지 않습니다. 그렇지만 원본 데이터베이스에 수정 사항이 발생하게 되면 수정되기 전의 데이터 페이지가 이 스파스 파일에 저장되므로 최대 데이터베이스 스냅숏을 생성할 당시의 크기의 디스크를 할당 받게 됩니다.

③ 데이터베이스 스냅숏 생성

데이터베이스 스냅숏을 만들기 전에 먼저, 만들고자 하는 디스크에 충분한 공간이 있는 지 확인해야 합니다. 이어서 원본 데이터베이스의 데이터 파일의 이름과 크기 등 정보를 확인합니다.

• 데이터베이스 스냅숏 생성 구문

```
CREATE DATABASE 〈데이터베이스 스냅숏 이름 지정〉
  ON
    (
    NAME = 〈원본 데이터베이스의 논리적 데이터 파일 이름 지정〉,
    FILENAME = '〈데이터베이스 스냅숏의 새로운 스파스 파일 위치 지정〉'
    ) [ ,...n ]
  AS SNAPSHOT OF 〈원본 데이터베이스 이름 지정〉 [;]
```

〈데이터베이스 스냅숏 이름 지정〉

데이터베이스 스냅숏의 이름을 지정합니다. 일반적으로 데이터베이스 스냅숏을 생성한 시점을 표시하는 것을 권장합니다.

NAME

원본 데이터베이스의 논리적이 데이터 파일 이름을 지정합니다.

FILENAME

데이터베이스 스냅숏의 새로운 스파스 파일의 물리적인 위치와 파일 이름을 지정합니다. 이 때의 파일 이름은 확장자를 갖는 물리적인 파일 이름입니다.

[,...*n*]

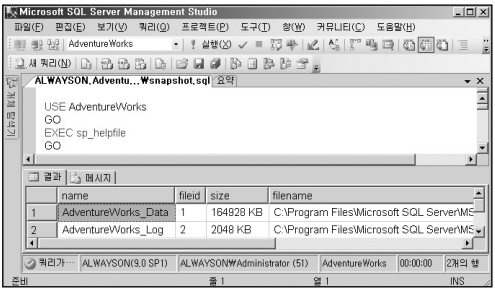
일반적으로 데이터베이스는 하나의 .mdf 확장자를 갖는 주 파일 하나와 .ndf 확장자를 갖는 보조 데이터 파일로 구성됩니다. 사용자 데이터는 사용자 파일 그룹에 저장하는 것을 원칙으로 함으로 사용자 데이터베이스는 여러 개의 데이터 파일로 구성되는 것이 일반적입니다. 이런 경우 각 데이터 파일을 각 파일 별로 괄호안에 NAME, FILENAME을 지정해야 합니다.

AS SNAPSHOT OF 〈원본 데이터베이스 이름 지정〉

생성하는 데이터베이스 스냅숏의 원본 데이터베이스 이름을 지정합니다. 원본 데이터베이스는 데이터베이스 스냅숏과 동일한 서버 인스턴스에 있어야 합니다.

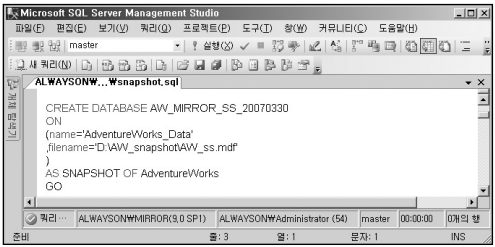
• 원본 데이터베이스의 데이터 파일 확인

데이터베이스 스냅숏을 생성하기 위해서는 원본 데이터베이스의 로그 파일을 제외한 모든 데이터 파일에 대해서 새로운 스파스 파일의 위치를 지정해야 합니다.



따라서 sp_helpfile과 같은 시스템 저장 프로시저를 사용해서 원본 데이터베이스의 데이터 파일 정보를 정확하게 확인합니다. 그렇지만 미러 데이터베이스는 사용자가 액세스할 수 없으므로 주 서버에서 데이터베이스 정보를 확인합니다. 또한 데이터 파일의 크기를 고려해서 데이터베이스 스냅숏을 만들고자 하는 위치에 충분한 공간이 있는 지도 확인합니다.

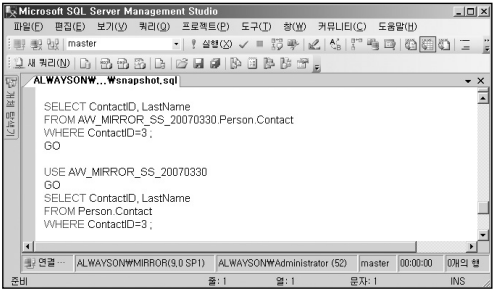
• 데이터베이스 스냅숏 생성



[CREATE DATABASE ~ AS SNAPSHOT OF~] T-SQL 구문을 사용해서 미러 데이터베이스에 대한 데이터베이스 스냅숏을 생성합니다.

• 데이터베이스 스냅숏에 대한 액세스

SQL Server가 일반적으로 데이터에 액세스하기 위해서 사용하는 Four-Part 이름을 사용해서 데이터베이스 스냅숏 통해서 데이터를 조회합니다. Four-Part 이름은 **[인스턴스_이름].[데이터베이스_이름].[스키마_이름].[개체_이름]**으로 구성되면 **[데이터베이스_이름]** 부분에 데이터베이스 스냅숏 이름을 지정합니다. 일반적으로 **[인스턴스_이름]** 부분은 생략하며 **[데이터베이스_이름]** 부분도 USE 구문을 사용하거나 연결 문자열에 지정하므로 생략합니다.



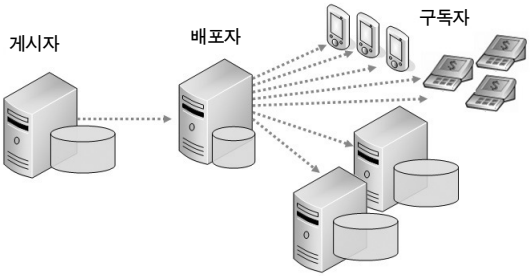
[주의]

*데이터베이스 스냅숏은 SQL Server 2005 Enterprise Edition에서만 지원하는 기능으로 보다 자세한 사항은 필자가 집필한 도서출판 대림의 **[SQL Server 2005 재해 복구 및 고가용 솔루션]**을 참조하기 바랍니다.*

3. 피어 투 피어 복제

1) 복제 개요

복제는 운영중인 데이터베이스의 데이터와 데이터베이스 개체를 다른 데이터베이스로 복사 및 배포한 후 데이터베이스간에 일관성 있게 시차를 두고 동기화하는 과정입니다. 복제를 사용하면 데이터를 여러 다른 위치에 배포하거나 LAN이나 WAN, 무선 연결 및 인터넷을 사용하여 원격의 사용자나 이동이 잦은 사용자에게 배포할 수 있습니다. 또한, 복제를 사용해서 운영 데이터베이스의 부하를 분산하고 OLTP 데이터베이스와 보고 및 의사 결정 지원 데이터베이스를 물리적으로 분리할 수 있으며 원격에 데이터베이스의 복사본을 유지할 수 있으므로 가용성을 향상할 수도 있습니다. 복제는 이와 같이 가용성과 확장성을 모두 향상할 수 있으므로 재해 복구 및 고가용 솔루션을 고려할 때 반드시 고려해야 하는 기술입니다.



〈복제 토폴로지 〉

SQL Server 복제 아키텍처를 이해하기 위해서는 먼저 SQL Server 복제에서 사용되는 용어에 친숙해야 합니다. 그래서 일단 용어부터 설명하겠습니다. SQL Server 복제는 게시자, 배포자, 구독자 세가지 구성 요소로 구성됩니다. 게시자는 배포하고자 하는 원본 데이터를 가지고 있는 서버 인스턴스입니다. 구독자는 게시자가 배포한 게시 데이터를 배달 받는 대상 서버 인스턴스입니다. 배포자는 게시자가 게시한 게시를 구독자에게 전달하는 역할을 하는 서버 인스턴스입니다. 복제에서 사용하는 용어의 이해를 돕기 위해서 온라인 설명서는 잡지사와 비교하고 있지만 필자는 다음과 같이 신문사에 비교해서 설명하겠습니다.

게시자 : 신문사입니다. 요즘 유명한 조 · 중 · 동 가운데 하나라고 보면 됩니다.
게시자는 구독자에게 배달할 신문을 생산하는 신문사와 같이 구독자에게 배달할 데이터베이스의 개체 및 데이터를 게시합니다.

배포자 : 보급소입니다. 신문사로부터 신문을 받아와 배달의 기수를 고용해서 구독자에게 배달하고 구독 내역 등의 정보를 유지관리 하는 역할을 수행합니다.

구독자 : 바로 우리와 같이 신문을 구입해서 읽는 구독자 입니다.

게 시 : 게시자가 구독자에게 전달하는 단위로 신문사의 발행 단위인 신문으로 이해하면 됩니다.

아티클 : 신문에 포함된 개별 기사입니다. [김남주 결혼기사] 같은 것입니다.

복제 에이전트 : 복제의 작업을 실질적으로 수행하는 일꾼들입니다.

이와 같은 복제는 다음과 같은 기능을 제공합니다. 이 부분에서 현장의 요구 사항과 맞는다면 복제 도입을 검토해야 할 것입니다.

① 확장성 및 가용성 향상

여러 사이트가 같은 데이터 복사본을 유지합니다. 이것은 여러 사이트가 같은 데이터를 읽어야 하거나 배치 작업을 위해 별도의 서버가 필요로 하는 경우에 부하를 분산하고 가용성을 향상합니다. 이 부분이 우리의 주제와 맞습니다.

② 보고서 및 의사 결정 지원 시스템 분리

OLTP 응용 프로그램을 데이터 웨어하우스, 온라인 분석 처리(OLAP) 데이터베이스와 같이 읽기 의존적인 응용 프로그램과 구분합니다.

③ 여러 사이트의 데이터 통합

영업 사원, 택배 사원 등을 비롯한 모바일 사용자와 POS 응용 프로그램과 같은 다양한 데이터를 통합합니다.

④ 다른 유형의 데이터 통합

Oracle과 같은 이기종 (비-SQL Server)환경과 데이터를 통합합니다.

2) 복제의 유형

SQL Server는 기본적으로 다음 세 가지 유형의 복제를 제공하며 또한 각 복제 유형이 갖는 단점을 극복하기 위해서 다양한 고급 옵션을 제공하고 있습니다.

- 스냅숏 복제

스냅숏 복제는 특정 시점에 데이터를 구독자에게 배달합니다. 또한, 배포한 데이터에 업데이트를 모니터링하지 않습니다. 다시 동기화가 일어나면 전체 스냅숏이 생성되어 구독자에게 배달됩니다. 다음과 같은 경우에 스냅숏 복제가 적합합니다.

① 데이터가 자주 변경되지 않는 경우
② 짧은 기간 동안 수정이 많이 발생하는 경우
③ 데이터의 크기가 크지 않은 경우
④ 구독자 측이 최신의 데이터가 필요하지 않은 경우

• 트랜잭션 복제

트랜잭션 복제는 일반적으로 초기화를 위해서 게시 데이터베이스 개체 및 데이터의 스냅숏으로 시작합니다. 초기 스냅숏이 배달되면 게시자에서 게시 데이터 변경 내용 및 스키마 수정 내용이 구독자로 계속 반영됩니다. 이러한 작업은 거의 실시간으로 수행됩니다.
데이터 변경 내용은 게시자에서 발생한 것과 같은 순서 및 같은 트랜잭션 경계 내에서 구독자에게 적용되므로 게시 내에서는 트랜잭션 일관성이 보장됩니다. 다음과 같은 경우에 트랜잭션 복제가 적합합니다.

① 증분 변경 내용을 발생과 동시에 구독자로 전파하려는 경우
② 게시자에서 많은 양의 입력, 수정 및 삭제 작업이 수행되는 경우
③ 데이터가 변경의 이력이 모두 반영되어야 하는 경우
(예를 들어 한 행이 특정 기간 동안 여러 번 변경된 경우 해당 변경 사항이 발생할 때마다 구독자에 반영되어야 하는 경우)
④ 게시자 또는 구독자가 Oracle과 같은 비-SQL Server(Non-SQL Server) 데이터베이스인 경우

트랜잭션 복제는 다음과 같은 네 가지의 게시 유형을 제공합니다.

① 표준 트랜잭션 게시
② 구독자가 구독 데이터를 업데이트할 수 있는 게시
③ 피어 투 피어 토폴로지 게시
④ Oracle과 같은 비-SQL Server(Non-SQL Server) 데이터베이스 게시

• 병합 복제

병합 복제는 트랜잭션 복제와 마찬가지로 일반적으로 게시 데이터베이스 개체 및 데이터를 스냅숏으로 시작합니다. 게시자 및 구독자에서 발생한 후속 데이터 변경 및 스키마 수정은 트리거로 추적합니다. 구독자는 네트워크에 연결될 때 게시자와 동기화하여 마지막 동기화 이후 게시자와 구독자 간에 변경된 모든 행을 병합합니다. 다음과 같은 경우에 병합 복제에 적합합니다.

① 여러 구독자가 다양한 시간에 동일한 데이터를 수정하고 변경 내용을 게시자 및 다른 구독자에 전파해야 하는 경우

② 구독자는 데이터를 받아 게시자와 연결되지 않은 상태에서 독립적으로 데이터를 변경할 수 있고 나중에 게시자 및 다른 구독자와 변경 내용을 동기화해야 하는 경우

③ 동일한 데이터를 게시자와 여러 구독자가 수정하는 충돌이 발생할 수 있으며 충돌이 발생하면 충돌을 감지하고 해결할 수 있어야 하는 경우

④ 데이터가 변경의 이력이 필요 없는 경우

(예를 들어 한 행이 특정 기간 동안 여러 번 변경된 경우 해당 변경 사항의 최종 변경만 구독자로 반영되어도 되는 경우)

[주의]

일반적인 경우에 복제의 초기화를 위해서 스냅숏 복제를 많이 이용합니다. 그렇지만, SQL Server 2005에서는 트랜잭션 복제를 구성하는 경우에 전체 백업을 통해서 초기화하는 기능도 새롭게 제공하고 있습니다. 많은 테이블을 초기화하거나 대용량의 테이블을 초기화하는 경우에는 백업을 통한 초기화가 효율적일 수 있습니다.

3) 복제의 작동 방법

여기는 약간 어려운 부분입니다. 피어 투 피어 복제가 주제인데 거기까지 가기 위해 참 길이 험난합니다. 이미 되돌아가기엔 너무 멀리 왔으니 계속 가도록 하겠습니다. 단순히 알고 넘어가는 것이 아니고 복제에 대해서 충분히 이해한 상태에서 구축하고 관리해야 하기 때문에 반드시 알아야 하는 개념이니까 즐거운 마음으로 계속 따라오기 바랍니다. 복제 프로세스는 snapshot.exe, logread.exe, distrib.exe, replmerge.exe, qrdrsvc.exe라는 각각의 복제 실행 프로그램이 SQL Server 에이전트의 작업으로 수행됩니다. 그래서 복제의 각 단계를 처리하는 독립된 프로세스들도 에이전트라고 부릅니다. 복제의 에이전트는 다음과 같이 다섯 개의 에이전트가 있습니다.

① 스냅숏 (Snapshot) 에이전트

스냅숏 에이전트는 스냅숏 복제에 사용되며 게시된 테이블 및 개체의 스키마 및 초기 데이터 파일을 준비하고, 스냅숏 파일을 저장하고, 배포 데이터베이스의 동기화에 대한 정보를 기록합니다. 배포자 서버가 스냅숏 에이전트를 실행합니다. 복제 실행 파일은 snapshot.exe 입니다.

② 로그 판독기 (Log Reader) 에이전트

로그 판독기 에이전트는 트랜잭션 복제에 사용되며 복제 표시된 트랜잭션을 게시자의 트랜잭션 로그에서 배포 데이터베이스로 가져와 저장합니다. 트랜잭션 복제를 사용하여 게시한 각 데이터베이스에는 배포자에서 실행되고 게시자로 연결되는 자체 로그 판독기 에이전트가 있습니다. 복제 실행 파일은 logread.exe 입니다.

[주의]
로그 판독기 에이전트가 게시자에서 실행된 트랜잭션 로그를 배포 데이터베이스로 이동하기 전에는 로그 백업을 수행하거나 로그 잘라내기(Truncate)를 수행하여도 트랜잭션 로그를 잘라낼 수 없습니다. 따라서, 데이터베이스 관리자는 배포자 서버가 정상적으로 작업을 수행하고 있는 지를 주의 깊게 모니터해야 합니다.

③ 배포 (Distributor) 에이전트

배포 에이전트는 스냅숏 복제 및 트랜잭션 복제와 함께 사용되며 초기 스냅숏을 구독자에 적용하고 로그 판독기 에이전트가 가져온 트랜잭션을 구독자로 배달합니다. 배포 에이전트는 밀어넣기 구독을 지정하면 배포자에서, 끌어오기 구독을 지정하면 구독자에서 실행됩니다. 복제 실행 파일은 distrib.exe 입니다.

④ 병합 (Merge) 에이전트

병합 에이전트는 병합 복제와 함께 사용되며 초기 스냅숏을 구독자에 적용하고 발생한 증분 데이터 변경 내용을 이동하거나 조정합니다. 각 병합 구독에는 게시자 및 구독자 모두에 연결되고 모두를 업데이트하는 자체 병합 에이전트가 있습니다. 병합 에이전트는 밀어넣기 구독을 지정하면 배포자 에서, 끌어오기 구독을 지정하면 구독자에서 실행됩니다. 기본적으로 병합 에이전트는 변경 내용을 구독자에서 게시자로 업로드한 다음 게시자에서 구독자로 변경 내용을 다운로드 합니다. 복제 실행 파일은 replmerge.exe 입니다.

⑤ 큐 판독기 (Queue Reader) 에이전트

큐 판독기 에이전트는 트랜잭션 복제 가운데 구독자의 지연 업데이트 옵션이 있는 경우에 사용됩니다. 구독자가 오프라인 상태에서 업데이트를 하고 해당 업데이트를 자신의 큐에 저장합니다. 구독자가 온라인이 되면 이 에이전트는 배포자에서 실행되어 구독자가 적용한 변경 내용을 읽어서 게시자로 보내는 역할을 합니다. 배포 에이전트 및 병합 에이전트와는 달리 지정한 배포 데이터베이스에 대한 모든 게시자 및 게시에 대해 하나의 큐 판독기 인스턴스만 사용됩니다. 복제 실행 파일은 qrdrsvc.exe 입니다.

이와 같은 복제 에이전트는 프로필이라고 하는 에이전트가 실행될 때마다 사용할 매개 변수 집합을 가지고 수행됩니다. 각 에이전트는 기본 프로필과 로그 판독기 에이전트, 배포 에이전트 및 병합 에이전트에 대해서는 미리 정의된 추가 프로필을 제공합니다. 뿐만 아니라 응용 프로그램 요구 사항에 적합한 사용자 정의 프로필을 만들어 사용할 수 있습니다. 사용자 정의 에이전트 프로필을 사용하면 해당 프로필에 연결된 모든 에이전트에 대해 매개 변수를 쉽게 변경할 수 있고 각 에이전트의 여러 인스턴스는 각각의 프로필을 사용할 수 있습니다.

그럼 이제 보험회사 영업 사원 시나리오를 통해서 **[구독자에서 업데이트할 수 있는 트랜잭션 복제]**를 예로 들어서 복제 프로세스가 어떻게 수행되는 지 정리를 해보겠습니다.

시나리오

보험회사 영업팀장 김태석은 자신의 고객 정보 데이터를 회사 데이터베이스 서버에서 자신의 노트북으로 내려 받아 외근을 나가야 합니다. 고객을 방문하여 고객 정보를 수정하거나 새로운 고객을 등록하기도 하고 새로운 보험상품에 가입도 시킬 예정입니다. 사내에서 근무할 때는 회사의 데이터베이스에서 변경된 사항이 가능한 빨리 자신의 노트북에 반영되어야 합니다. 실시간일 필요는 없습니다. 사내에서는 노트북에서의 변경사항이 회사 서버에 실시간으로 즉시 반영되어야 합니다. 외근을 할 때는 노트북을 지참하고 오프라인 상태에서 업무가 가능해야 합니다. 회사로 돌아와 온라인이 되면 노트북의 변경 사항이 자동으로 회사 서버에 반영되어야 합니다. 김태석의 노트북에는 SQL Server 2005 Express Edition 이 설치되어 있습니다.

① 최초 초기화를 위해서 복제할 대상 테이블과 저장 프로시저, 뷰 등을 선택하고 게시합니다. 이때 필요한 테이블 가운데 고객 테이블은 영업사원 김태석이 필요한 컬럼만 지정하고 김태석의 고객 데이터만 포함하여 원본 테이블을 필터링한 게시를 만듭니다.

② 김태석의 노트북의 SQL Server 2005 Express Edition으로 밀어넣기 구독을 사용한 지연 업데이트 및 즉시 업데이트 구독을 생성합니다.

③ 배포자 서버의 스냅숏 에이전트가 게시된 테이블 및 개체를 생성할 수 있는 스키마 파일과 데이터가 저장된 파일을 만들어서 스냅숏 폴더에 저장하고 복제 정보를 유지하기 위해서 배포자 서버가 가지고 있는 배포데이터베이스에 작업 상태를 저장합니다. 이 때 실행되는 파일은 snapshot.exe이며 기본 에이전트 프로필에 설정된 매개 변수 설정에 따라서 작업을 수행합니다.

④ 배포자 서버의 배포 에이전트가 복제 데이터베이스의 정보를 확인하고 distrib.exe 실행 파일을 기본 에이전트 프로필에 따라서 수행합니다. 수행하는 작업은 스냅숏 폴더에서 스냅숏 파일을 가져다가 구독자 서버에 실행해서 게시된 개체를 생성하고 데이터를 로드하는 작업입니다. 이제 김태석의 노트북에는 업무에 필요한 테이블과 데이터가 복사되었습니다.

⑤ 게시 데이터베이스에서 새로운 보험 상품이 추가되는 변경 사항이 발생하면 배포자 서버의 로그 판독기 에이전트는 logread.exe 실행 파일을 기본 프로필에 따라서 수행하여 게시자에서 트랜잭션 로그를 읽어서 배포자 서버에 배달합니다.

⑥ 배포 에이전트는 게시 서버에서 수행된 변경 사항을 구독자 서버에서 반영합니다.

⑦ 사내에서 김태석은 전화로 영업을 합니다. 새로운 고객을 섭외하고 등록합니다. 이때 구독자의 변경 사항은 게시자 서버에 즉시 반영됩니다. 즉시 구독 업데이트 옵션은 분산 트랜잭션으로 처리 되므로 MSDTC 서비스를 사용합니다.

⑧ 김태석이 상담을 위해서 노트북을 가지고 고객의 회사를 방문합니다. 고객 현지혜는 새로운 상품에 가입을 합니다.

⑨ 김태석은 오프라인 상태에서 작업을 해야 합니다. 즉시 구독 업데이트 옵션은 분산 트랜잭션을 사용하므로 서버와 연결이 끊어진 오프라인 상태에서는 작업이 불가능합니다. 따라서, 구독 업데이트 옵션을 지연 업데이트로 변경합니다.

⑩ 구독 옵션이 지연 업데이트이므로 구독자의 변경사항은 자체적으로 반영되고, 나중에 서버와 동기화하기 위해서 큐 테이블에 따로 저장합니다.

⑪ 김태석이 회사로 돌아와서 노트북을 켜고 LAN 케이블을 꽂아 온라인 상태가 되면 배포자 서버의 큐 판독기 에이전트가 qrdrsvc.exe 실행 파일을 기본 프로필에 따라서 수행을 하며 이 때 구독자의 큐 테이블에서 구독자의 변경 사항을 게시 서버에 반영합니다. 김태석은 퇴근시간에 오늘도 한 건 했다는 기쁨에 술집으로 향합니다.

이제 여러분은 SQL Server의 복제의 개념에 대해서 이해되었을 겁니다. 왜 복제라는 기능이 필요한지, 복제의 유형은 어떤 것들이 있으며 복제 프로세스는 어떻게 동작하는 지도 감을 잡았을 겁니다. 또한 여러분의 요구 사항을 충족하기 위해서 어떤 유형의 복제를 구성해야 할 지도 이해했을 겁니다. 이것을 바탕으로 이제 SQL Server 2005에서 달라진 기능에 대해서 살펴보도록 하겠습니다.

4) SQL Server 2005에서 달라진 복제 기능

SQL Server 2005 복제는 다음과 같은 여러 새로운 기능과 향상된 기능을 추가하였습니다.

DDL 구문 지원	향상된 복제 보안
스냅숏 배달 재개 기능	향상된 복제 모니터
복제 에이전트 지원 향상	향상된 Identity 범위 관리
논리적 레코드 복제	병렬 스냅숏 준비
RMO(복제 관리 개체) 지원	트랜잭션 게시를 위한 추적 프로그램 토큰
병합 복제의 웹(HTTPS) 동기화 지원	트랜잭션 구독을 백업에서 초기화
Oracle 게시자 복제 지원	재초기화 없이 트랜잭션 아티클의 호출 형식 수정 가능
피어 투 피어 복제 지원	트랜잭션 게시에서 동시 스냅숏 기본 사용
병합 에이전트 및 배포 에이전트에 대한 병렬 처리	트랜잭션 게시에서 허용되는 열 개수 증가

필터링된 병합 게시에 대한 파티션 미리 계산	병합 구독에 대한 통계 모니터링 기능 향상
필터링된 병합 게시에 대한 새로운 분할 옵션	매개 변수가 있는 필터의 병합 게시에 대한 향상된 스냅숏
병합 게시에 대한 새로운 구독자 업로드 옵션	병합게시의 아티클에 대한 선언적 정렬
병합 복제의 향상된 BLOB 배달 기능	병합 게시의 아티클에 대한 조건적 삭제 처리
업데이트 할 수 있는 구독 향상	향상된 오류 메시지

〈 SQL Server 2005 향상된 복제 기능 〉

- 스키마 변경 DDL 구문 복제 지원: SQL Server 2000에서는 sp_repladdcolumn (Transact-SQL) 및 sp_repldropcolumn(Transact-SQL) 프로시저를 사용하여 게시된 테이블에서 열을 추가하거나 삭제할 수 있었지만 SQL Server 2005에서는 특별한 저장 프로시저를 사용하지 않고 다음과 같은 스키마를 변경하는 DDL 구문 내용을 복제할 수 있습니다.

  - ALTER TABLE
  - ALTER VIEW
  - ALTER PROCEDURE
  - ALTER FUNCTION
  - ALTER TRIGGER (DML 트리거만 지원됨)

[주의]
테이블에 대한 스키마 변경은 Transact-SQL 또는 SMO(SQL Server Management Objects)를 사용하여 수행해야 합니다. SQL Server Management Studio 에서 스키마를 변경하면 일부 작업은 해당 테이블을 삭제하고 다시 만들려고 합니다. 그러나 게시된 개체는 삭제할 수 없으므로 스키마 변경에 실패합니다.

- 스냅숏 배달 재개 기능: 배달 중에 중단된 스냅숏의 자동 재개를 비롯하여 스냅숏 생성 및 적용 기능이 향상되었습니다. 특정 시점에서 중단된 스냅숏 배달은 자동으로 재개되며 배달이 이미 완료된 파일은 다시 보내지 않습니다. 이러한 기능을 활용하기 위한 특수한 옵션은 필요하지 않습니다.

- 복제 에이전트 지원 향상: 복제 에이전트의 안정성과 오류 복구 성능이 향상되었습니다. 에이전트 및 작업 간 충돌이 감소되었으며 네트워크 오류, 교착 상태 조건 및 쿼리 시간 종료 등의 경우에 에이전트는 자동으로 연결을 다시 시도합니다.

- 논리적 레코드 복제: 기본적으로 병합 복제는 행 단위로 변경 내용을 처리합니다. 논리적 레코드 기능은 병합 복제에서 주문 마스터 테이블의 부모 행과 주문 상세 테이블의 해당 자식 행과 같은 일련의 관련된 행을 하나의 단위로 취급할 수 있도록 합니다. 게시 속성의 조인 추가 대화상자에서 논리적 레코드를 정의하면 네트워크 안정성이나 기타 다른 요소에 관계없이 관련된 레코드 집합이 항상 전체 단위로 한 번에 구독자에서 처리되도록 보장합니다.

- RMO(복제 관리 개체) 지원: RMO는 복제 구성, 관리 및 스크립팅과 구독자 동기화를 위한 일련의 공용 언어 런타임 클래스를 제공하는 Microsoft .NET Framework 라이브러리입니다. 이 라이브러리를 사용하면 최상위 클래스로부터의 순회 검색 없이도 게시 또는 구독 클래스와 같은 개별 클래스를 프로그램에서 사용할 수 있습니다. 관련되지 않는 개체나 더 이상 사용되지 않는 개체는 캐시되지 않습니다. 따라서 가비지 수집기에서 사용되지 않은 메모리를 수집하고 확장성을 향상시킬 수 있습니다.

- 병합 복제의 웹(HTTPS)을 통한 동기화 지원: 병합 복제를 위한 웹 동기화 옵션은 HTTPS를 통해 데이터를 복제할 수 있는 기능을 제공합니다. HTTPS를 통한 동기화는 인터넷을 통한 복제와 Microsoft Windows Mobile 장치와 같은 모바일 구독자를 포함하는 토폴로지에 적합합니다.

- Oracle 게시자 복제 지원: SQL Server 2000에서도 데이터를 DB2나 Oracle과 같은 다른 데이터베이스에 게시하는 기능이 지원되었지만, 사용자 지정 프로그래밍 없이 다른 데이터베이스의 데이터를 게시하는 기능은 지원되지 않았습니다. SQL Server 2005에서는 표준 SQL Server 복제와 동일한 방식으로 Oracle 데이터베이스를 SQL Server에 직접 복제할 수 있습니다. 스냅숏 및 트랜잭션 복제가 지원됩니다.

- 피어 투 피어 복제: SQL Server 2000에서는 구독자에 복제된 데이터를 게시자가 소유하는 계층적 토폴로지가 트랜잭션 복제에서 지원되었습니다. 업데이트 구독이 포함된 트랜잭션 복제에서 구독자 측 업데이트가 지원되었지만 구독자는 복제에서 게시자와 다른 유형의 참여자로 분류되었습니다. SQL Server 2005에서는 토폴로지에서 동일한 참여자 간의 복제가 허용되는 새로운 피어 투 피어 모델이 제공됩니다. 이 새로운 지원 기능은 유지 관리 또는 장애 관리를 위해 복제된 노드 간의 동적 역할 이동이 필요한 서버 간 구성을 실행 중인 고객을 위해 디자인되었습니다.

- 병합 에이전트 및 배포 에이전트에 대한 병렬 처리: 병합 에이전트 및 배포 에이전트에 대해 병렬 처리를 허용하는 새로운 매개 변수를 제공합니다. 병합 에이전트 매개 변수는 -ParallelUploadDownload이며, 이 옵션은 병합 에이전트가 게시자에 업로드된 변경 내용과 구독자에 다운로드된 변경 내용을 병렬로 처리할 수 있도록 허용하여 네트워크 대역폭이 높은 대규모 환경에서 유용하며 배포 에이전트 매개 변수는 -SubscriptionStreams로 이 옵션은 배포 에이전트에 따른 여러 연결에서 구독자에 대한 변경 내용 일괄 처리를 병렬로 적용하고 단일 스레드를 사용할 때 나타나는 여러 트랜잭션 특성을 유지할 수 있도록 허용하여 집계 복제의 처리량을 크게 향상시켜 줍니다.

- 필터링된 병합 게시에 대한 파티션 미리 계산: 파티션 미리 계산은 매개 변수가 있는 필터(이전 버전에서는 "동적 필터"라고 함)를 사용하는 병합 게시에 대한 새로운 성능 최적화 기능입니다. 구독자가 게시자와 동기화되면 게시자는 구독자의 필터를 평가하여 그 구독자의 분할(파티션)이나 데이터 집합에 속하는 행을 확인해야 합니다. 게시자에서 필터링된 데이터 집합을 수신하는 각 구독자에 대한 변경 내용의 파티션 멤버 자격을

결정하는 이 과정을 파티션 평가라고 합니다. 미리 계산된 파티션을 사용하지 않을 경우 특정 구독자에 대해 병합 에이전트를 마지막으로 실행한 이후에 게시자에서 필터링된 열의 각 변경 내용에 대해 파티션 평가를 수행해야 합니다. 그러나 게시자와 구독자가 Microsoft SQL Server 2005에서 실행 중이고 미리 계산된 파티션을 사용할 경우 게시자의 모든 변경 내용에 대해 파티션 멤버 자격이 미리 계산되고 변경 시 지속됩니다. 결과적으로 구독자가 게시자와 동기화할 때 파티션 평가 과정을 거치지 않고도 파티션과 관련된 변경 내용을 즉시 다운로드할 수 있습니다. 이 기능을 통해 게시에 변경 내용, 구독자 또는 아티클이 많을 경우 성능이 크게 향상될 수 있는 것입니다.

- 필터링된 병합 게시에 대한 새로운 분할 옵션: 매개 변수가 있는 필터를 통해서 생성되고 병합 복제에서 성능을 최적화하는 데 사용할 수 있는 분할(파티션) 유형을 지정할 수 있습니다. 예를 들어 단일 구독자에서 데이터 분할(파티션)이 복제 및 업데이트되도록 지정하면(병합 복제의 일반적인 시나리오) 복제에서 추적 및 처리되는 메타데이터가 줄어들기 때문에 최적의 성능을 제공할 수 있습니다.

- 병합 게시에 대한 새로운 구독자 업로드 옵션: 병합 게시에서 구독자의 변경 내용을 게시자에 업로드할지 여부를 지정할 수 있습니다. 여러 응용 프로그램에서 일부 테이블(예: 제품 데이터파 포함된 테이블)은 게시자에서만 변경되어야 합니다. 이러한 유형의 아티클을 "다운로드 전용"으로 지정하면 성능을 향상시킬 수 있습니다.

- 병합 복제의 향상된 BLOB 배달 기능: 매우 큰 BLOB 열에 대한 메모리 활용도를 높일 수 있도록 BLOB 데이터의 처리 및 배달 기능이 향상되었습니다.

- 업데이트 할 수 있는 구독 향상: SQL Server 2000에서 구독자에 대한 즉시 업데이트 및 지연 업데이트에서는 큰 데이터 형식(text, ntext 및 image)에 대한 업데이트가 허용되지 않았지만 SQL Server 2005에서는 varchar(max) 및 varbinary(max) 데이터 형식이 추가되었으며 데이터 형식이 트리거에서 조작되기 때문에 트랜잭션 업데이트 구독자에서 큰 데이터 형식에 대한 업데이트를 지원할 수 있습니다. 이러한 기능을 활용하기 위한 특수한 옵션은 필요하지 않습니다.

- 향상된 복제 보안: 에이전트 간 보안 세분성이 향상되었습니다. 이전 버전의 SQL Server 에이전트는 기본적으로 SQL Server 에이전트의 컨텍스트에서 실행되었습니다. SQL Server 2005에서는 복제 에이전트가 실행되고 연결을 만드는 계정을 세부적으로 제어할 수 있습니다. 즉, 각 에이전트에 대해 다른 계정을 지정할 수 있습니다.

- 향상된 복제 모니터: 복제 모니터가 SQL Server 2005에서 완전히 다시 디자인되었습니다. 복제 토폴로지의 전반적 상태를 모니터링하여 게시 및 구독의 상태와 성능에 대한 자세한 정보를 제공합니다.

- 향상된 Identity 범위 관리: 다양한 ID 범위 관리 옵션과 할당 옵션이 추가되었습니다. 또한 병합 복제의 경우 각 노드에는 ID의 주 범위 및 보조 범위를 할당할 수 있습니다.

- 병렬 스냅숏 준비: 병렬 스냅숏 준비에는 스냅숏 에이전트 내에서 스키마를 스크립팅하거나 데이터를 대량 복사하는 동안 여러 아티클을 처리하는 과정이 포함되기 때문에 스냅숏 준비의 속도 및 효율성을 크게 향상시킬 수 있습니다. 이러한 기능을 활용하기 위한 특수한 옵션은 필요하지 않습니다.

- 트랜잭션 게시를 위한 추적 프로그램 토큰: SQL Server 2005는 이를 위한 새로운 추적 프로그램 토큰 기능을 제공합니다. 토큰(소량의 데이터)은 게시자에 삽입되고 구독자로 복제됩니다. 추적 프로그램 토큰이 시스템에서 이동할 때 통계가 수집되고 시스템 테이블에서 이러한 통계를 쿼리할 수 있습니다.

- 트랜잭션 복제에서 백업으로부터 복제 초기화: 처음부터 대량의 데이터가 포함된 데이터베이스 간 복제를 설정하는 작업은 시간이 많이 들고 저장소가 많이 필요합니다. SQL Server 2005는 구독을 시작하기 위해 스냅숏을 사용하는 대신 트랜잭션 게시를 만든 후에 가져온 백업을 구독자에서 복원할 수 있는 새로운 게시 옵션을 제공합니다.

- 다시 초기화 없이 트랜잭션 아티클의 호출 형식 수정 가능: 기본적으로 트랜잭션 게시의 아티클에 대해 복제는 일련의 저장 프로시저를 사용하여 구독자에 변경 내용을 전파합니다. 각 프로시저에서 사용되는 호출 구문은 프로시저에 제공되는 매개 변수의 구조 및 각 데이터 변경 시 구독자로 보낼 정보의 양을 결정합니다. 구독을 다시 초기화할 필요 없이 게시에 대해 호출 구문을 변경할 수 있습니다.

- 트랜잭션 게시에서 동시 스냅숏 기본 사용: SQL Server 2000에서는 동시 스냅숏을 트랜잭션 게시에서 사용할 수 있지만 기본적으로 사용되지는 않았습니다. 동시 스냅숏은 스냅숏 생성 중에 잠금이 유지되는 시간을 줄여 주므로 스냅숏 파일을 만드는 동안에도 사용자가 방해 받지 않고 작업을 계속할 수 있습니다. SQL Server 2005에서 동시 스냅숏은 기본적으로 사용됩니다.

- 트랜잭션 게시에서 허용되는 열 개수 증가: 트랜잭션 게시에는 각 게시된 테이블에 최대 1,000개까지의 열이 허용됩니다.

- 병합 구독에 대한 통계 모니터링 기능 향상: 병합 구독에서 제공되는 아티클 수준의 통계를 통해 병합 단계가 완료되기까지 남은 시간, 지정된 아티클 처리에 소요된 시간, 구독자가 사용 중인 연결 유형 및 기타 중요한 정보를 확인할 수 있습니다. 이 통계는 복제 모니터의 동기화 기록 탭에 표시됩니다.

- 매개 변수가 있는 필터의 병합 게시에 대한 향상된 스냅숏: 매개 변수가 있는 필터의 게시에 대한 스냅숏(이전 버전에서는 "동적 스냅숏" 이라고 함)은 구독자에게 올바른 데이터 파티션을 제공합니다. 이러한 스냅숏은 높은 성능을 제공하지만 수 백 또는 수 천의 구독자가 있는 경우에는 관리하기 어렵습니다. 병합 복제를 사용하면 각 구독자에 대한 스냅숏을 사전에 생성하거나 구독자가 처음으로 동기화하는 중에 스냅숏 생성을 시작할 수 있도록 지정할 수 있습니다.

- 병합게시의 아티클에 대한 선언적 정렬: 병합 동기화 중에 처리하는 아티클의 순서를 명시적으로 정의할 수 있으며, 트리거로 참조 무결성을 정의하거나 트리거의 특정 실행 순서에 의존하는 경우에 특히 유용합니다.

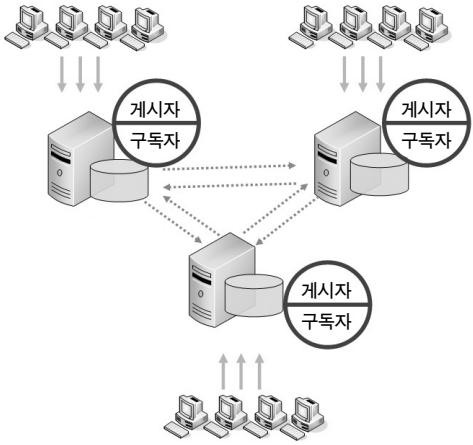
- 병합 게시의 아티클에 대한 조건적 삭제 처리: 일부 아티클의 경우에는 삽입 및 업데이트 작업이 구독자에서 게시자로 복제되어야 하며, 삭제 작업은 복제되지 않아야 하는 경우에 조건적 삭제 처리 작업을 사용하면 구독자의 테이블에 대한 삭제 작업이 게시자로 복제되지 않도록 지정할 수 있습니다.

- 향상된 오류 메시지: 상당히 많은 오류 메시지가 다시 작성되었으며 원인 및 해결 방법에 대해 더욱 자세한 정보가 제공됩니다. 또한 문제 해결 설명서에서 일부 오류에 대한 추가 정보가 제공됩니다.

5) 피어 투 피어 (Peer TO Peer) 트랜잭션 복제 개요

재해를 대비하고 고가용성을 유지하기 위해서 앞에서 살펴본 다양한 유형의 복제를 적용할 수 있습니다. 그렇지만 SQL Server 2005 Enterprise Edition이 새롭게 제공하는 피어 투 피어 트랜잭션 복제가 고가용 솔루션으로 가장 적합하다고 판단하여 여기에서는 피어 투 피어 트랜잭션 복제에 대해서 다루겠습니다.

피어 투 피어 복제는 복사 데이터베이스를 가진 모든 노드가 게시자이면서 구독자입니다. 따라서, 동일한 스키마와 데이터를 게시하고 구독하며 모든 노드에서 변경(수정, 입력 및 삭제) 작업을 수행할 수 있습니다. 변경 내용이 두 번 이 상 노드 전체에 순환되는 것을 방지하기 위해 변경 내용이 지정된 노드에 적용되면 복제가 이를 인식하여 중복 실행이나 오류를 방지합니다.



피어 투 피어 트랜잭션 복제는 복제에 참여하는 어느 데이터베이스에서나 데이터를 읽거나 수정할 수 있는 응용 프로그램용을 위해서 디자인되었습니다. 데이터를 검색하는 쿼리를 여러 데이터베이스로 분산시켜 응용 프로램 부하를 줄이고 성능을 향상시킬 수 있으며 변경 내용은 모든 노드로 전파되어 한번씩만 실행되므로 수정, 입력 및 삭제 성능이 단일 노드 상황과 유사합니다. 특히 각 피어 노드가 지리적으로 업무 시간대가 다른 지역에 분산되어 있으면 다른 사이트가 사용 중이지 않을 때 읽기 작업의 부하는 분산 처리가 가능합니다. 그렇지만 수정 작업에 대해서는 가능한 하나의 사이트를 사용하는 것이 업무 처리 데이터의 무결성을 유지하기 위해서 효율적입니다.

즉 피어 투 피어 복제는 트랜잭션 작업보다 읽기 작업이 많은 경우에 유용하게 사용할 수 있습니다. 한번의 주문을 위해서 많은 조회를 하는 온라인 쇼핑몰 같은 경우가 좋은 예입니다. 쇼핑몰의 멤버는 분산된 서버에 접속해서 상품을 조회합니다. 하나의 상품을 구입하기 위해서 유사한 상품을 비교하기도 하고 상품 평을 읽기도 합니다. 최종적으로 구매하고자 하는 상품을 장바구니에 넣고 결제를 진행합니다.

이때 대금을 결제하는 프로세스만 특정 서버를 사용하도록 합니다. 그리고 한번 결제한 사용자는 계속 해당 서버에서 작업을 수행하도록 응용 프로그램을 디자인합니다. 수정, 입력 및 삭제와 같은 트랜잭션은 특정 서버에서만 발생하지만 다른 노드로 전파됩니다.

피어 투 피어 복제는 이렇게 단 한번의 주문을 위해서 접속한 많은 고객의 읽기 작업 비중이 트랜잭션 작업보다 높은 경우에 읽기 작업에 대한 부하를 분산하여 성능을 향상합니다. 더구나 여러 노드에 데이터베이스의 복사본을 유지하므로 재해를 대비할 수 있고 가용성을 향상시킵니다. 따라서 스냅숏 복제, 트랜잭션 복제, 병합 복제 등 다양한 복제 유형 가운데에서 확장성과 가용성 향상에 가장 효율적인 복제가 바로 이 피어 투 피어 트랜잭션 복제입니다.

6) 피어 투 피어 트랜잭션 복제 구성 시 고려 사항

- 복제의 대상은 사용자 데이터베이스만 지원합니다.
- 피어 투 피어 복제를 통한 가용성을 디자인 하기 위해서는 각 피어 노드의 로그인 계정이 동일하도록 로그인을 전송해야 합니다.
- 피어 투 피어 복제에 참여하는 모든 데이터베이스에는 동일한 스키마와 데이터를 포함해야 합니다.
- 피어 투 피어 복제에 참여하는 모든 데이터베이스의 개체 이름, 개체 스키마 및 게시 이름이 서로 동일해야 합니다.
- 게시에서 스키마 변경 내용을 복제 할 수 있어야 합니다. 게시 속성 replicate_ddl의 값이 반드시 1로 설정되어야 합니다.
- 행 및 열의 필터링은 사용할 수 없습니다.
- 각 노드가 별도의 배포 서버를 사용하는 것을 권장합니다. 단일 지점 실패(Single-Point of Failure)를 대비해야 합니다.
- 테이블 및 기타 개체는 단일 게시 데이터베이스 내의 여러 피어 투 피어 게시에 포함될 수 없습니다.

- 구독을 만들기 전에 먼저 게시에 피어 투 피어 복제를 사용할 수 있도록 설정해야 합니다.
- 백업이나 'replication support only' 옵션을 사용하여 구독을 초기화해야 합니다.
- 충돌 감지 및 해결 기능은 제공되지 않습니다. 수정, 입력 및 삭제와 같은 트랜잭션은 가능한 하나의 데이터베이스에서만 수행하도록 디자인합니다.
- Identity 속성의 열은 사용하지 않는 것이 좋습니다. Identity 속성의 열을 사용하고자 하는 경우에는 복제에 참여한 각 데이터베이스별로 별도의 범위를 수동으로 관리해야 합니다.
- 다음 작업을 수행하고자 하는 경우 시스템을 정지해야 합니다. 즉, 게시된 테이블의 모든 노드에서 작업을 정지하고 각 노드에서 다른 모드 노드의 변경 내용을 모두 받았는지 확인해야 합니다.
  - 기존 토폴로지에 노드 추가
  - 기존 게시에 아티클 추가
  - 스키마 변경
  - 백업에서 노드 복원

7) 피어 투 피어 복제 구성 및 관리

여기에서는 먼저 두 개의 노드로 피어 투 피어 복제를 구성하고 운영중인 피어 투 피어 토폴로지에 노드를 추가하는 방법을 포함하는 관리 방법을 살펴보도록 하겠습니다.

■ 피어 투 피어 복제 구성 및 관리 단계

1 단계 : 피어 투 피어 복제에 참여한 멤버 서버를 각각 배포자로 구성합니다. 배포자를 별도로 두어도 상관없지만 단일 지점 실패(Single-Point of Failure)에 대비 해서 가용성을 유지해야 합니다.

2 단계 : 새 게시 마법사를 사용해서 데이터베이스를 게시합니다.

3 단계 : 게시 속성 페이지에서 피어 투 피어 복제용 게시를 설정합니다.

4 단계 : 게시용 데이터베이스를 전체 백업하여 다른 멤버에서 복원하는 방법 등으로 초기화를 수행합니다.

5 단계 : 피어 투 피어 토폴로지 구성 마법사를 사용하여 피어 투 피어 복제를 구성합니다.
6 단계 : 피어 투 피어 복제 구성을 확인합니다.
7 단계 : 가용성과 확장성의 향상을 위해서 새 피어 노드를 추가합니다.
8 단계 : 기준 노드 장애 시 남은 노드간 트랜잭션 배달을 확인합니다.
9 단계 : 내결함성 향상을 위해서 새로 추가된 피어 노드에 각각의 구독 추가합니다.
10 단계 : DDL 구문 배달을 확인합니다.

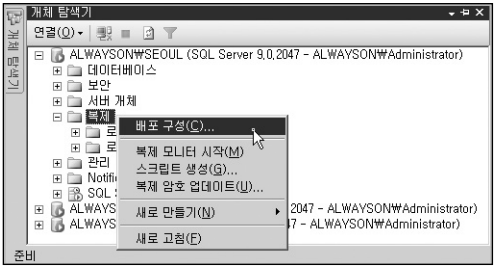
1단계 : 배포자 구성

다음 표를 참조하여 피어 투 피어 복제를 구성합니다.

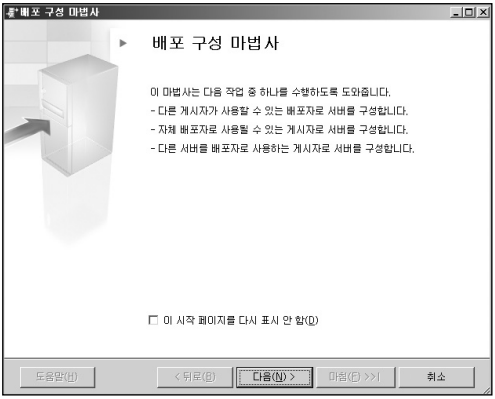
첫 번째 피어 노드	ALWAYSON\SEOUL
두 번째 피어 노드	ALWAYSON\SEATTLE
세 번째 피어 노드	ALWAYSON\CAIRO
게시 데이터베이스	AdventureWorks

〈 피어 투 피어 토폴로지 구성 정보 〉

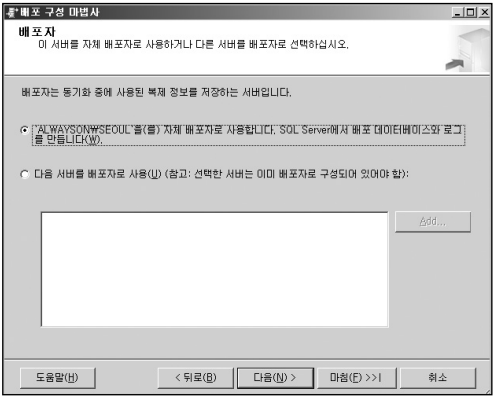
피어 투 피어 복제에 참여하는 [ALWAYSON\SEOUL] 서버 인스턴스와 [ALWAYSON\SEATTLE] 서버 인스턴스를 각각 배포자 서버로 구성합니다.



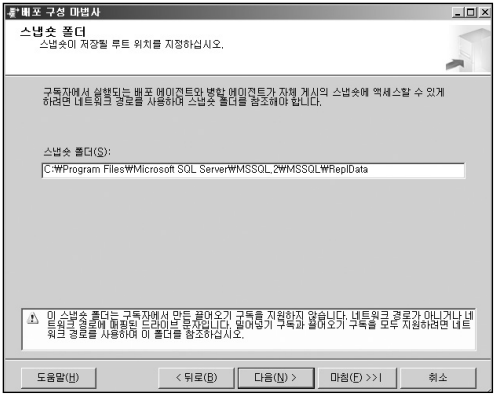
[개체 탐색기]의 [복제]에서 오른쪽 버튼을 누른 뒤 [배포 구성(C)]을 선택합니다.



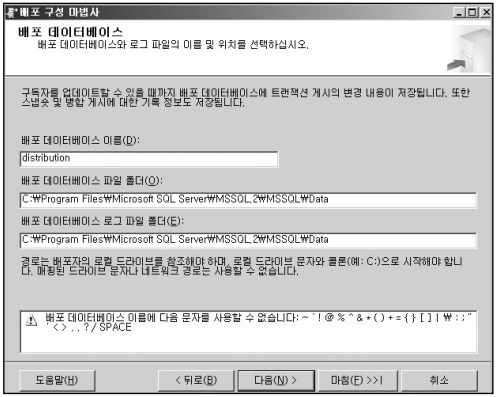
[배포 구성 마법사]가 시작되면 [다음(N) >] 버튼을 누릅니다.



[배포자] 페이지는 배포자 서버를 선택하는 페이지입니다. **['ALWAYSON\SEOUL' 을(를) 자체 배포자로 사용합니다.]** 부분의 선택을 확인하고 **[다음(N) >]** 버튼을 누릅니다.

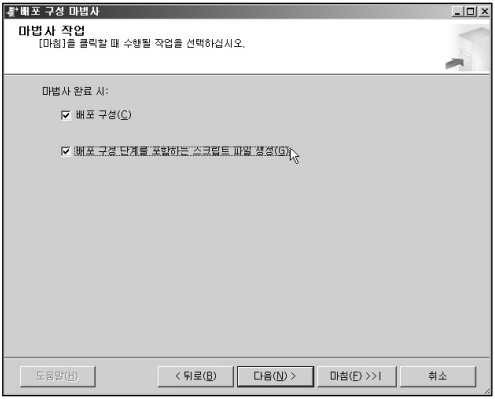


[스냅숏 폴더] 페이지는 스냅숏 파일이 저장되는 위치를 지정하는 페이지입니다. 피어 투 피어 복제는 스냅숏을 사용하지 않고 백업을 사용해서 초기화를 수행할 것이므로 그대로 두고 **[다음(N) >]** 버튼을 누릅니다.



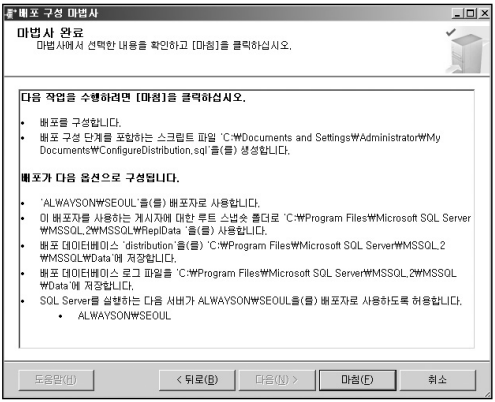
[배포 데이터베이스] 페이지는 배포자 서버에서 복제관련 정보를 유지하는 데이터베이스의 이름과 파일의 위치를 지정합니다. **[배포 데이터베이스 이름(D):]** 부분을 [distribution]으로 변경하고 **[다음(N) >]** 버튼을 누릅니다.

[게시자] 페이지는 게시자 서버를 지정하고 배포 정보를 저장할 배포 데이터베이스를 지정합니다. [ALWAYSON\SEOUL]이 게시자로 선택되어있음을 확인하고 [다음(N) >] 버튼을 누릅니다.



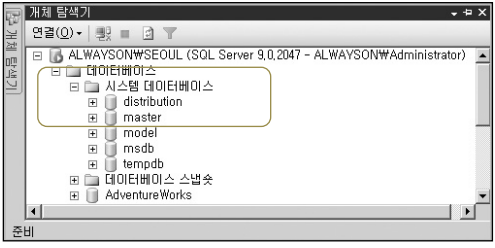
[마법사 작업] 페이지에서는 마법사 완료 시 수행하는 작업을 지정하는 페이지입니다. 복제 구성 및 설정 스크립팅은 복제에 문제가 발생하거나 하는 경우 복제 구성을 위해서 필요하며 복제를 모두 구성한 뒤에 전체 구성에 대해서 하나의 스크립트로 생성하는 것도 가능합니다만 스크립트 확인을 위해서 **[배포 구성 단계를 포함하는 스크립트 파일 생성(G)]**을 선택한 뒤 **[다음(N) >]** 버튼을 누릅니다.

[스크립트 파일 속성] 페이지에서는 배포 구성 단계를 포함하는 스크립트 파일 생성을 위한 파일 경로와 파일 형식 등을 지정한 뒤 **[다음(N) >]** 버튼을 누릅니다.

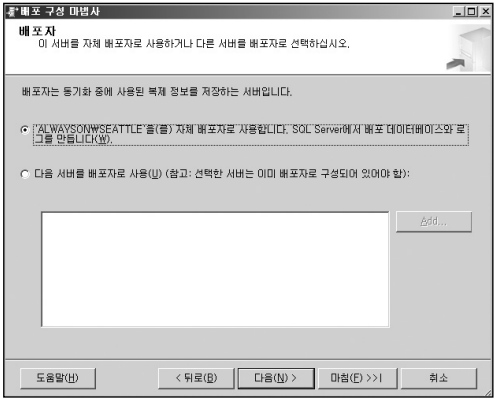


[마법사 완료] 페이지는 마법사에서 설정한 사항을 확인하는 페이지입니다. 설정 사항을 확인 한 뒤 **[다음(N))]** 버튼을 누릅니다.

[구성하는 중] 페이지에서는 작업의 성공 여부를 확인합니다. 오류가 발생하는 경우 메시지 부분에 오류 메시지가 표시됩니다.

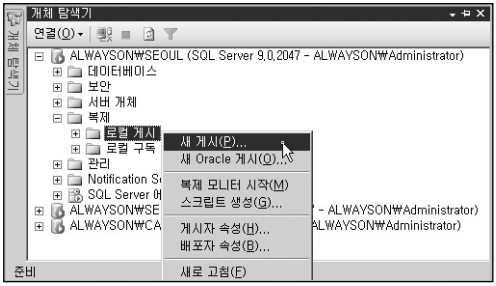


[개체 탐색기]에서 distribution 배포 데이터베이스의 생성을 확인합니다.

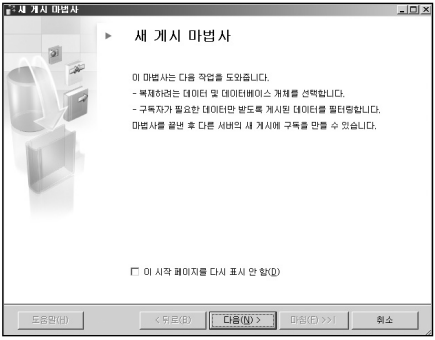


이어서 [ALWAYSON₩SEOUL]에서 배포자 서버를 구성한 방식으로 나머지 [ALWAYSON₩SEATTLE] 서버 인스턴스를 배포자 서버로 구성합니다. 두 멤버 모두 배포자 구성이 완료되면 다음 단계인 [데이터베이스 게시하기]를 진행합니다.

2 단계 : 데이터베이스 게시

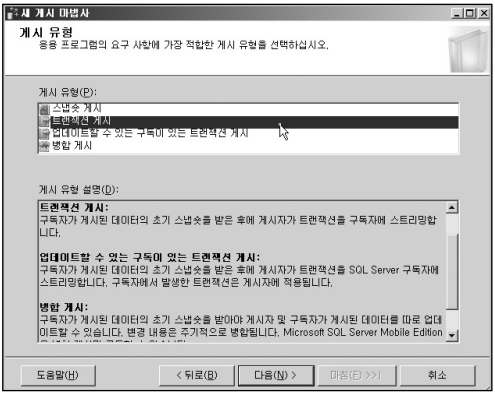


게시자 서버인 [ALWAYSON₩SEOUL]에서 [**개체 탐색기**]의 [**복제**], [**로컬 게시**]를 차례로 확장하고 [**로컬 게시**]에서 오른쪽 버튼을 누른 뒤 [**새 게시(P)**]을 선택합니다.

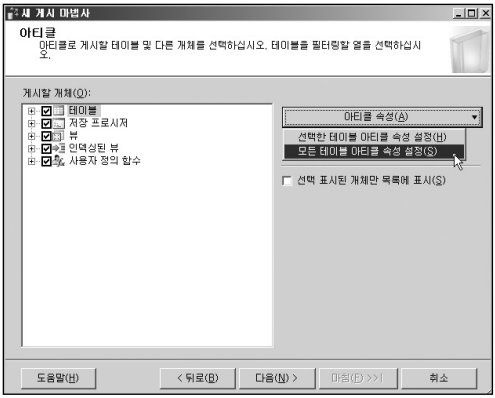


새 게시 마법사가 실행되면 [**다음(N) >**] 버튼을 눌러서 게시 마법사를 진행합니다.

[**게시 데이터베이스**] 페이지에서는 게시하려는 데이터 또는 개체가 있는 데이터베이스를 선택합니다. 여기에서는 AdventureWorks 데이터베이스를 선택하고 [**다음(N) >**] 버튼을 누릅니다.

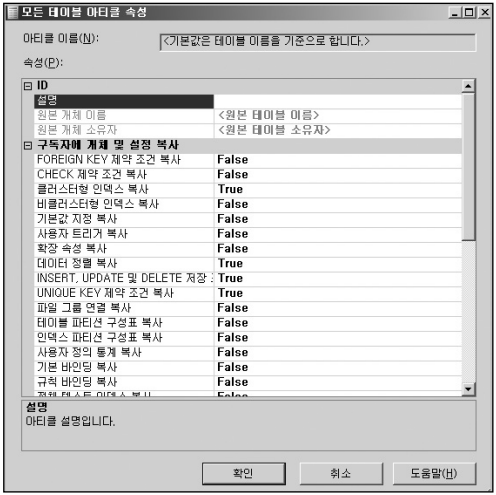


[게시 유형] 페이지에서는 스냅숏 게시, 트랜잭션 게시, 업데이트할 수 있는 구독이 있는 트랜잭션 게시, 병합 게시 가운데에서 게시 유형을 선택하는 페이지입니다. 여기에서는 **[트랜잭션 게시]**를 선택하고 **[다음(N))]** 버튼을 누릅니다.



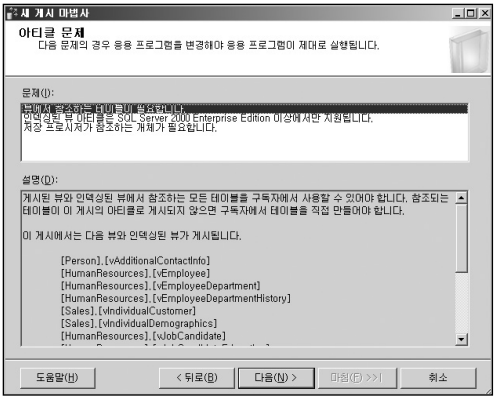
[아티클] 페이지에서는 게시할 테이블 및 개체를 선택하고 테이블을 필터링할 열을 지정하며 선택한 개체의 아티클 속성을 설정합니다. 특정 테이블을 선택하고 **[아티클 속성(A)]** 버튼을 눌러서 **[선택한 테이블 아티클 속성 설정(H)]**을 선택해서 개별적인 테이블 별로 아티클 속성을 설정하거나 전체 테이블을 선택하고 **[모든 테이블 아티클 속성 설정(S)]**을 선택하여 모든 테이블의 아티클 속성을 한번에 설정할 수 있습니다.

> ***[주의]***
> *[아티클] 페이지에서는 각 테이블 별로 원하는 컬럼만 복제하도록 컬럼 필터를 지정할 수 있습니다. 또한 **[테이블 행 필터]** 페이지에서는 조건을 지정하여 지정된 조건에 만족하는 행만 복제하도록 행 필터를 지정할 수 있습니다. 그렇지만 **피어 투 피어 복제**에서는 컬럼 필터와 행 필터를 모두 지원하지 않습니다.*

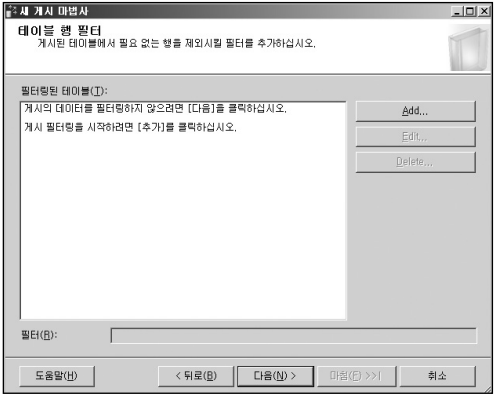


[아티클 속성] 페이지에서는 다음과 같은 속성을 복사하도록 지정할 수 있습니다.

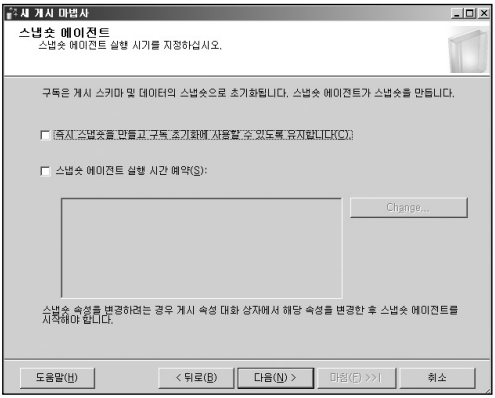
- Identity 속성값의 범위 지정
- 참조 키 (FOREIGN KEY), 기본값(DEFAULT), CHECK 제약, UNIQUE KEY 제약 조건
- 클러스터형 인덱스, 비클러스터형 인덱스, 사용자 트리거, 확장 속성
- 데이터 정렬 및 INSERT, UPDATE, DELETE 저장 프로시저
- 파일 그룹 연결, 테이블 파티션 구성표, 인덱스 파티션 구성표
- 사용자 정의 통계, 전체 텍스트 인덱스, XML 스키마, XML 인덱스, 권한
- 기존 개체를 삭제하고 새 개체를 만들지 여부와 기존 데이터 잘라낼지 여부 등



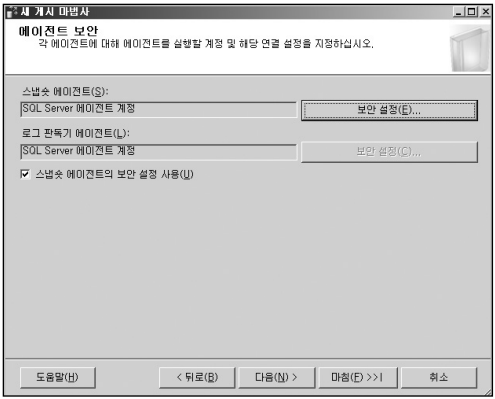
[아티클 문제] 페이지는 게시된 아티클이 참조하는 뷰와 저장 프로시저 리스트를 보여주고 참조 여부를 확인하는 페이지입니다. 참조 개체들의 게시 여부를 확인한 뒤 [다음(N) >] 버튼을 누릅니다.



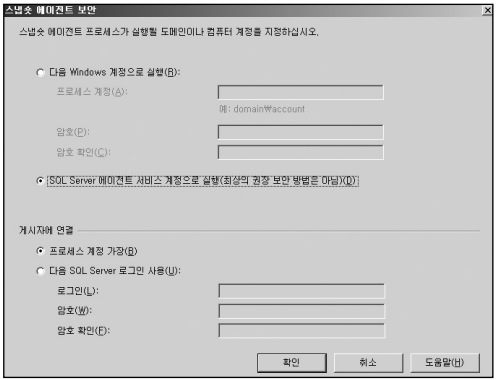
[테이블 행 필터] 페이지는 게시될 테이블에서 필요 없는 행을 제외하는 행 필터를 지정하는 페이지입니다. **[피어 투 피어]** 복제는 컬럼 필터와 행 필터를 지원하지 않으므로 행 필터를 지정하지 않고 **[다음(N) >]** 버튼을 누릅니다.



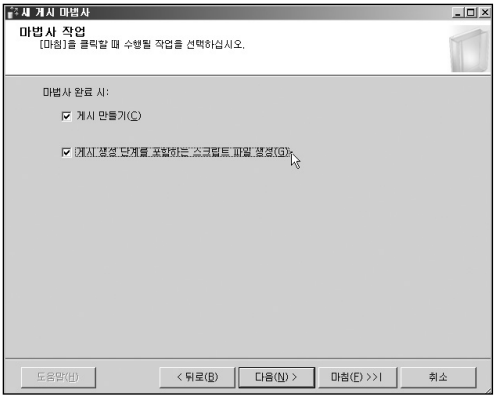
[스냅숏 에이전트] 페이지는 스냅숏을 만들어서 구독을 초기화할 지 여부와 스냅숏 에이전트의 실행 시간을 예약하는 페이지입니다. 백업과 복원으로 초기화하는 경우에는 지정할 필요가 없습니다. **[다음(N) >]** 버튼을 누릅니다.



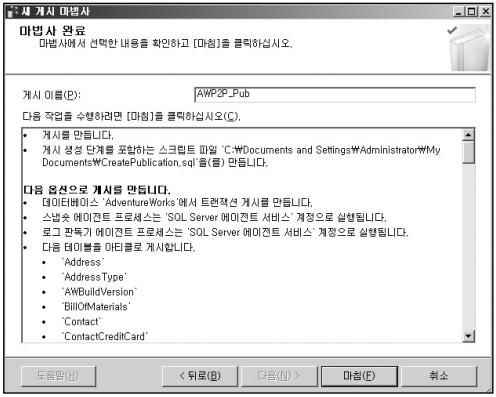
[에이전트 보안] 페이지는 스냅숏 에이전트와 로그 에이전트를 실행하는 계정을 지정하는 페이지입니다. 보안상 에이전트 계정을 지정하지 않고 필수 권한만 부여한 에이전트 프록시를 사용할 것을 권장합니다. **[프록시]**는 뒷부분에서 별도로 설명합니다. 피어 투 피어 복제는 스냅숏 에이전트를 사용하지 않지만 스냅숏 에이전트와 로그 판독기 에이전트의 보안을 모두 구성해야만 다음 단계로 진행할 수 있습니다.



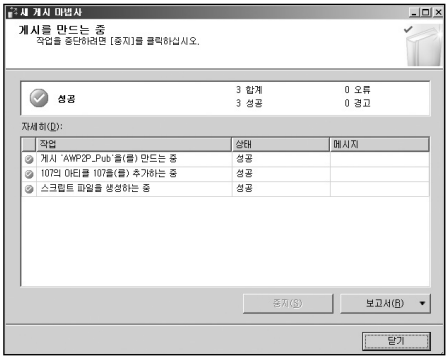
따라서 여기에서는 스냅숏 에이전트 보안을 **[SQL Server 에이전트 서비스 계정으로 실행(최상의 권장 보안 방법은 아님)(D)]**을 선택하고 **[확인]** 버튼을 누른 뒤 **[로그 판독기 에이전트 계정]** 밑의 **[스냅숏 에이전트의 보안 설정 사용(U)]**을 선택해서 스냅숏 에이전트에서 지정한 설정이 로그 판독기 에이전트 보안 설정에도 동일하게 적용되도록 합니다.



[마법사 작업] 페이지는 게시 만들기와 게시 생성 단계를 포함하는 스크립트 파일 생성 등 마법사 완료 시 작업 내용을 지정합니다. **[게시 만들기]**와 **[게시 생성 단계를 포함하는 스크립트 파일 생성(G)]**을 모두 선택하고 **[다음(N))]** 버튼을 누릅니다. **[스크립트 파일 속성]** 페이지에서는 배포 구성의 경우와 마찬가지로 스크립트 파일의 이름, 경로, 파일 형식 등을 지정하고 **[다음(N))]** 버튼을 누릅니다.

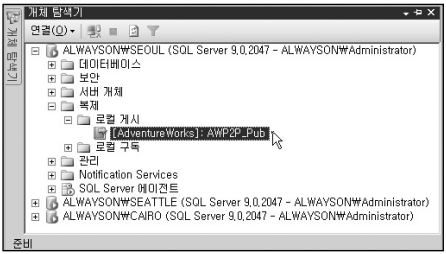


[마법사 완료] 페이지는 **[게시 이름(P)]**을 지정하고 **[새 게시 마법사]**에서 지정한 작업 내용을 확인합니다. **[게시 이름(P)]**에 **[AWP2P_Pub]**를 지정하고 **[마침(F)]** 버튼을 누릅니다.

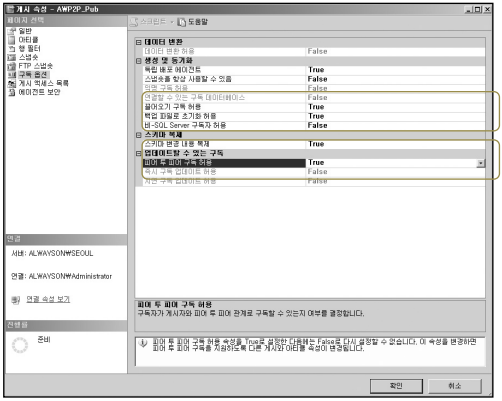


[게시를 만드는 중] 페이지에서는 마법사의 작업 수행 상태를 모니터하고 성공 및 실패 여부를 확인합니다. 경고 또는 오류가 발생하는 경우에 [메시지] 컬럼 부분에서 메시지를 확인할 수 있습니다.

3 단계 : 피어 투 피어 복제용 게시 설정



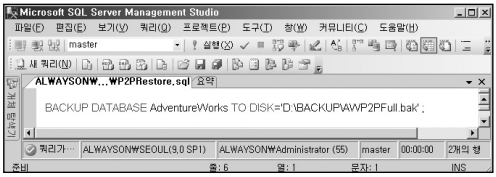
그림과 같이 [새 게시 마법사]를 통해서 오류 없이 게시를 생성한 경우에는 [개체 탐색기]에서 게시의 생성 여부를 확인하고 해당 게시를 선택하고 오른쪽 버튼을 눌러서[속성]을 선택해서 속성 창을 실행합니다.



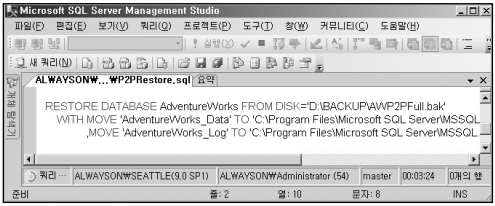
[구독 옵션] 페이지에서 [백업 파일로 초기화 허용] 및 [피어 투 피어 구독 허용] 속성 값을 [True]로 설정하고 [확인] 버튼을 누릅니다.

4 단계 : 게시용 데이터베이스 전체 백업으로 스키마 및 데이터 초기화

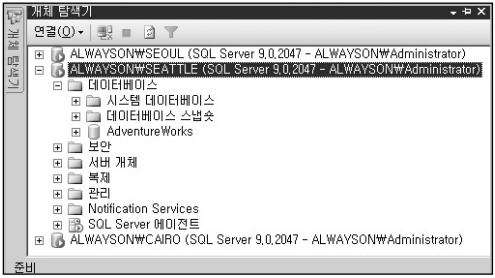
[ALWAYSON₩SEOUL] 서버 인스턴스에서 AdventureWorks 데이터베이스 전체 백업을 수행합니다.



[ALWAYSON₩SEATTLE] 서버 인스턴스에서 앞에서 수행한 백업 세트로부터 Adventure Works 데이터베이스를 복원합니다.



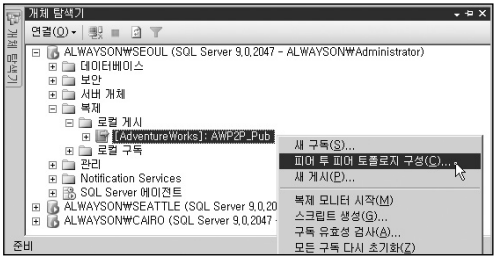
여기에서는 동일한 플랫폼에서 여러 개의 인스턴스로 작업하므로 복원 시 MOVE~TO 옵션을 사용해서[ALWAYSON₩SEATTLE] 인스턴스의 기본 데이터베이스 파일 경로로 지정합니다. 주의할 사항은 T-SQL로 데이터베이스를 복원할 때 [KEEP_REPLICATION] 옵션이나 SSMS에서 [복제 설정 유지] 옵션은 지정하지 않습니다.

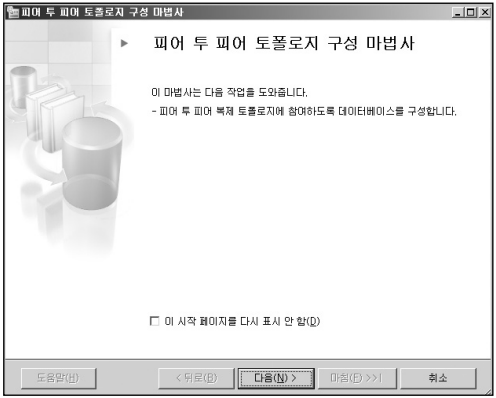


[개체 탐색기]에서 [ALWAYSON₩SEATTLE],[데이터베이스]를 차례로 확장하고 Adventure Works 데이터베이스가 초기화되었는지 확인합니다.

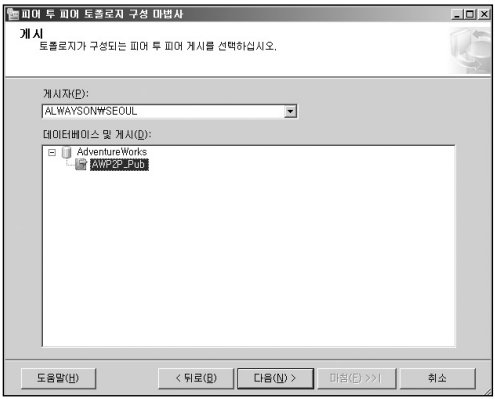
5 단계 : 피어 투 피어 토폴로지 구성

[ALWAYSON₩SEOUL] 서버 인스턴스에서 [개체 탐색기]의 [복제], [로컬 게시]를 차례로 확장하고 다음 그림과 같이 [AdventureWorks]:AWp2P_Pub 게시를 선택하고 오른쪽 버튼을 눌러서 [피어 투 피어 토폴로지 구성(C)]을 선택합니다.

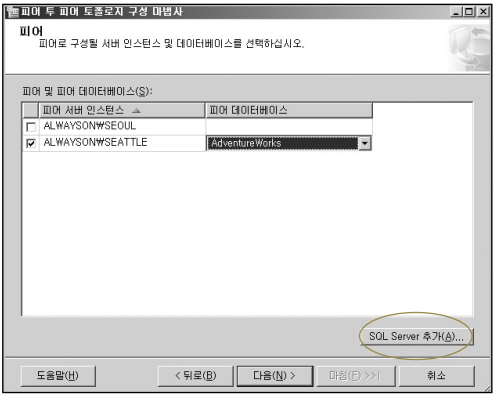




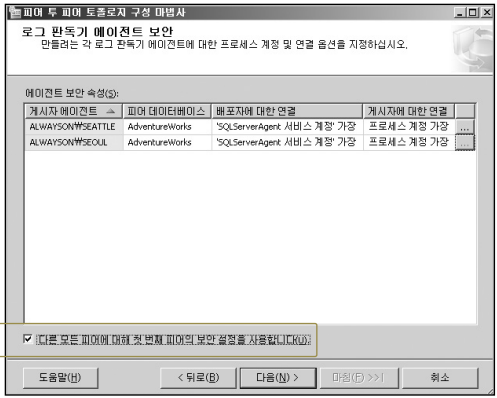
피어 투 피어 토폴로지 구성 마법사가 실행되면 [다음(N) >] 버튼을 눌러서 피어 투 피어 토폴로지 구성을 시작합니다.



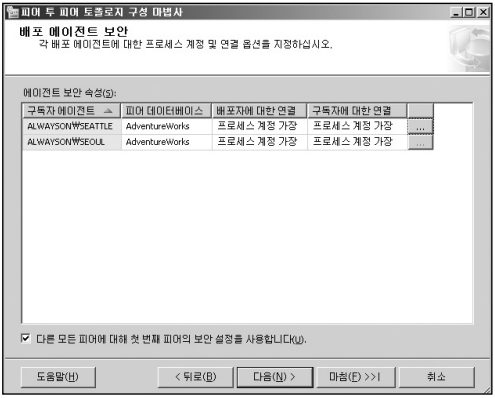
[게시] 페이지에서는 피어 투 피어 토폴로지가 구성되는 피어 투 피어 게시를 선택합니다. [게시자(P)]에서 [ALWAYSON₩SEOUL] 서버 인스턴스의 선택을 확인하고 [데이터베이스 및 게시(D)] 부분에서 [AWP2P_Pub]게시를 선택하고 [다음(N) >] 버튼을 누릅니다.



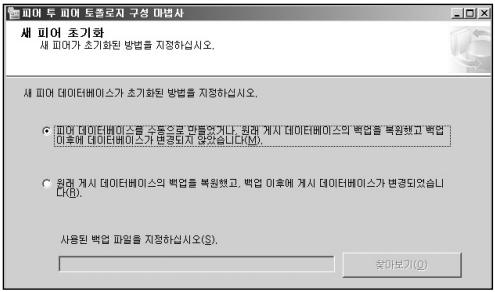
[피어] 페이지에서는 피어로 구성될 서버 인스턴스 및 데이터베이스를 선택합니다. 페이지 하단의 [SQL Server 추가(A)]을 눌러서 피어로 구성할 추가 서버 인스턴스 [ALWAYSON ₩SEATTLE]를 등록하고 [다음(N) >] 버튼을 누릅니다.



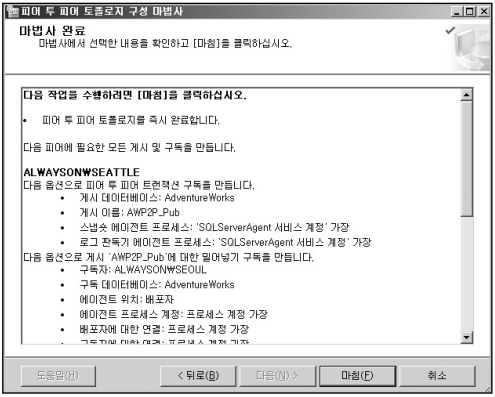
[로그 판독기 에이전트 보안] 페이지에서는 각 로그 판독기 에이전트에 대한 프로세스 계정 및 연결 옵션을 지정합니다. 첫 번째 피어에 대한 에이전트 보안 속성을 설정한 뒤 페이지 하단의 **[다른 모든 피어에 대해 첫 번째 피어의 보안 설정을 사용합니다(U).]**를 선택하면 나머지 피어에 동일한 설정이 적용됩니다.



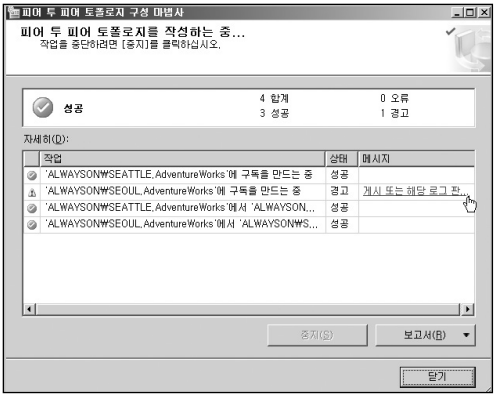
[배포 에이전트 보안] 페이지에서는 각 배포 에이전트에 대한 프로세스 계정 및 연결 옵션을 지정합니다. **[로그 판독기 에이전트 보안]** 페이지에서와 마찬가지로 배포 에이전트의 보안을 지정합니다.



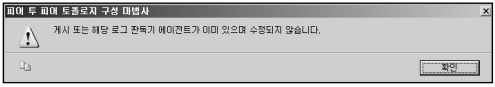
[새 피어 초기화] 페이지에서는 새 피어 데이터베이스의 초기화 방법을 지정합니다. 초기화를 위한 백업을 복원했고 그 이후에 변경 사항이 없는 경우에는 **[피어 데이터베이스를 수동으로 만들거나 원래 게시 데이터베이스의 백업을 복원했고 백업 이후에 데이터베이스가 변경되지 않았습니다(M).]**를 선택하고 **[다음(N) >]** 버튼을 누릅니다. 복원 이후에 변경이 발생한 경우는 **[원래 게시 데이터베이스의 백업을 복원했고, 백업 이후에 게시 데이터베이스가 변경되었습니다.]**를 선택하면 변경 여부를 추적하여 백업 이후의 트랜잭션을 구독 데이터베이스에 반영합니다.



[마법사 완료] 페이지는 **[피어 투 피어 토폴로지 구성 마법사]**에서 지정한 작업 내용을 확인합니다. 작업 내용을 확인하고 **[마침(F)]** 버튼을 누릅니다



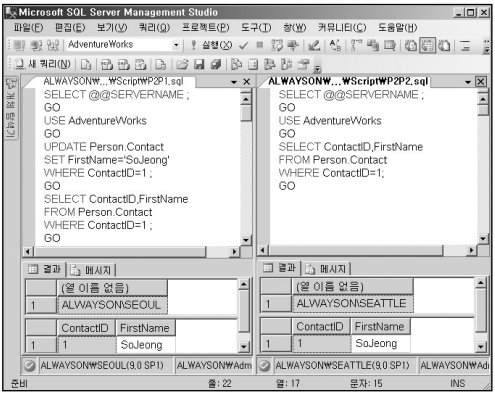
[구성하는 중] 페이지에서는 작업의 성공 여부를 확인합니다. 오류가 발생하는 경우 메시지 부분에 오류 메시지가 표시됩니다. 여기에서는 **[ALWAYSON₩SEOUL.AdventureWorks에 구독을 만드는 중]** 단계에서 경고가 발생하였습니다. **[메시지]** 컬럼에서 경고 메시지를 확인합니다. 경고 메시지를 누르면 다음과 같이 경고를 메시지 창을 통해서 확인할 수 있습니다.



여기에서는 게시자 서버 인스턴스에서 이미 게시 구성 단계에서 로그 판독기 에이전트를 구성하였으므로 **[피어 투 피어 토폴로지 구성 마법사]**에서 해당 구성을 수정하지 않았다는 경고 메시지입니다.

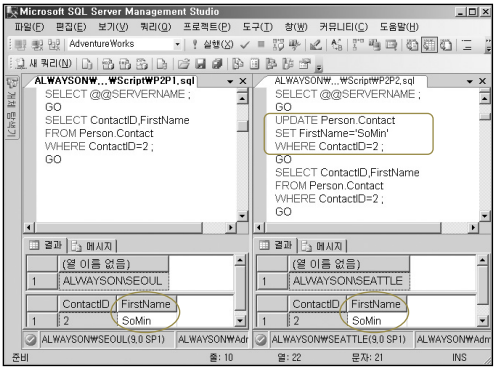
6 단계 : 피어 투 피어 복제에서 트랜잭션 배달 확인

피어 투 피어 토폴로지 구성을 완료하고 나면 SSMS(SQL Server Management Studio)에서 각 피어 데이터베이스에 연결한 뒤 트랜잭션을 실행해서 다른 멤버 노드로 트랜잭션이 이상 없이 배달되는지 확인합니다.



[ALWAYSON₩SEOUL]에 접속한 뒤 그림의 좌측 창과 같이 AdventureWorks 데이터베이스의 Person.Conatact 테이블의 데이터를 수정합니다. 이어서 [ALWAYSON₩SEATTLE]에 접속한 뒤 [ALWAYSON₩SEOUL]에서 수정한 트랜잭션이 배달되어 반영되었는지 여부를 확인합니다. 그림의 우측 창의 결과와 같이 잘 성공하였음을 확인합니다.

피어 투 피어 복제는 구성된 모든 멤버 노드가 피어이므로 이번에는 다음 그림과 같이 [ALWAYSON₩SEATTLE]에서 Person.Conatact 테이블의 데이터를 수정해서 [ALWAYSON₩SEOUL]로 트랜잭션이 배달되어 반영되는지 여부를 확인합니다.

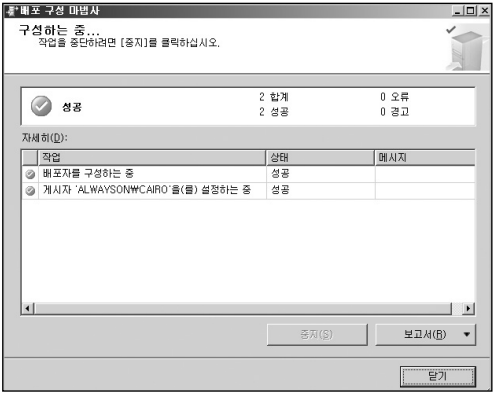


7단계 : 피어 투 피어 복제 노드 추가

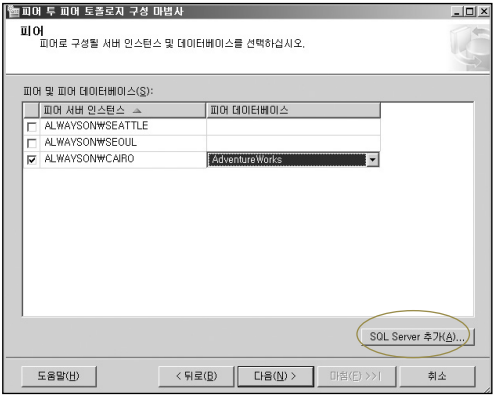
앞 단계에서 피어 투 피어 복제를 성공적으로 구성하였다면 이제 기존의 피어 투 피어 토폴로지에 노드를 추가하는 작업을 수행해봅니다. 여기서 기억해야 할 것은 하나 있습니다. **[피어 투 피어 트랜잭션 복제 구성 시 고려 사항]**에서 기존 피어 투 피어 토폴로지에 노드를 추가하는 경우 시스템을 정지해야 한다고 하였습니다. 이 사항을 유념하고 다음 단계를 통해서 새 노드를 추가합니다.

① 새 노드에 배포자 서버를 구성합니다.
② 기존 피어 투 피어 토폴로지의 첫 번째 멤버 노드에서 게시 데이터베이스를 백업하고 새 노드에서 복원합니다. 성능을 고려해서 백업 이후에 변경 사항이 많이 발생하지 않은 경우에는 이전의 백업을 사용할 수 있습니다.
③ 첫 번째 멤버 노드에서 **[피어 투 피어 토폴로지 구성 마법사]**에서 새 노드를 추가합니다.

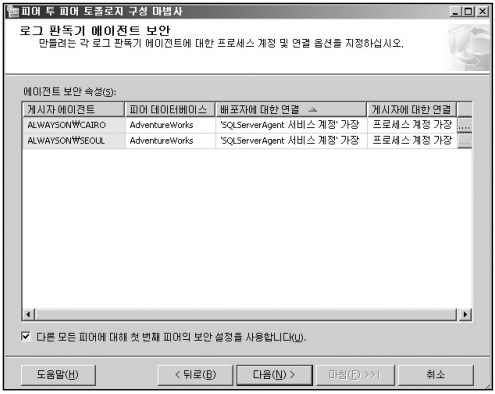
먼저 첫 번째 노드인 [ALWAYSON₩SEOUL]에서 게시 데이터베이스인 [AdventureWorks]를 전체 백업하고 새 노드인 [ALWAYSON₩CAIRO] 노드에서 복원합니다.



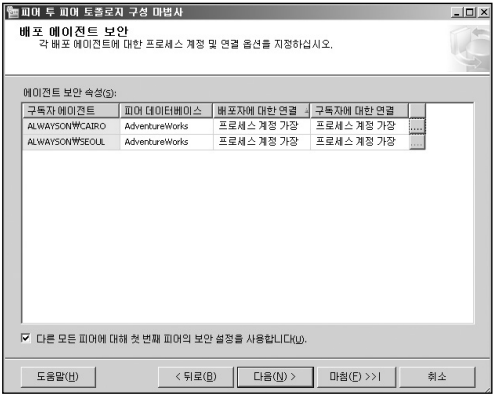
이어서 추가하는 새 노드를 배포자 서버로 구성합니다. 구성 방법은 **[단계 1: 배포자 구성]**를 참조합니다.



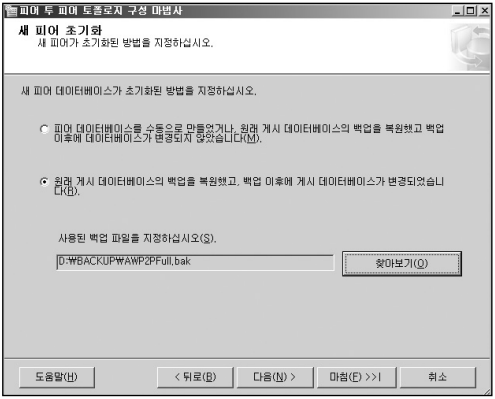
피어 투 피어 토폴로지의 첫 번째 노드인 [ALWAYSON₩SEOUL] 서버 인스턴스에서 **[개체 탐색기]**의 **[복제]**, **[로컬 게시]**를 차례로 확장하고 [AdventureWorks]:AWP2P_Pub 게시를 선택하고 오른쪽 버튼을 눌러서 **[피어 투 피어 토폴로지 구성(C)]**을 선택해서 **[피어 투 피어 토폴로지 구성 마법사]**를 실행합니다. **[피어]** 페이지의 하단 부분에 **[SQL Server 추가(A)]** 버튼을 눌러서 새 노드인 [ALWAYSON₩CAIRO]노드를 추가하고 AdventureWorks 데이터베이스를 추가합니다.



[로그 판독기 에이전트 보안] 페이지를 [ALWAYSON₩SEALTTLE]을 피어 노드로 구성할 때와 마찬가지로 설정합니다.



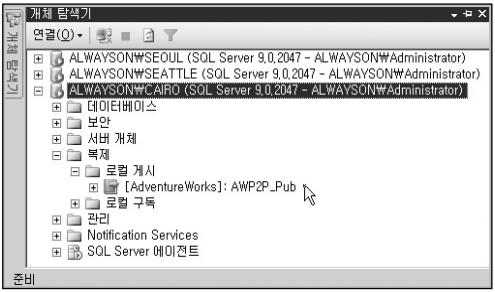
[배포 에이전트 보안] 페이지도 [ALWAYSON₩SEALTTLE]을 피어 노드로 구성할 때와 마찬가지로 설정합니다.

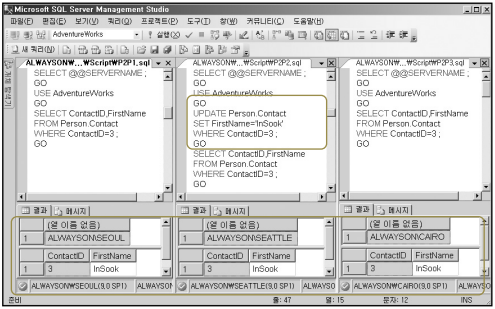


[새 피어 초기화] 페이지에서 [원래 게시 데이터베이스의 백업을 복원했고, 백업 이후에 게시 데이터베이스가 변경되었습니다(R).]를 선택하고 [찾아보기(O)] 버튼을 눌러서 백업 파일을 지정합니다. 이 옵션을 선택하면 백업 파일에 포함되지 않은 변경내용이 새 노드에 배달됩니다. 피어 투 피어 복제용 게시 설정 시 [백업 파일로 초기화 허용] 게시 속성을 [True]로 설정하였으므로 게시 데이터베이스의 변경 내용을 추적하여 새 노드에 복원 후에 최신 상태가 될 수 있도록 반영되지 않은 트랜잭션을 배달합니다. 전체 백업 이후 많은 트랜잭션이 발생한 경우에는 가능한 추가로 트랜잭션 로그 백업을 수행하여 차례로 복원합니다.

처음 피어 투 피어 토폴로지를 구성할 때와 마찬가지로 [ALWAYSON₩SEOUL] 노드에는 이미 게시와 로그 판독기 에이전트가 구성되어 있으므로 경고가 발생합니다.

새 노드의 구성이 완료되면 [개체 탐색기]에서 서버 인스턴스, [복제], [로컬 게시]를 차례로 확장하여 [AdventureWorks]:AWP2P_Pub 게시가 생성된 것을 확인합니다.

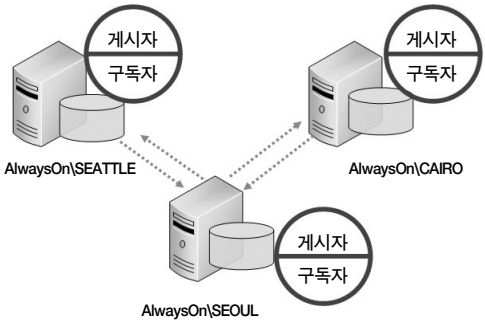




이어서 두 번째 노드인 [ALWAYSON₩SEATTLE]에서 트랜잭션을 실행하여 첫 번째 노드인 [ALWAYSON₩SEOUL]와 새 노드인 [ALWAYSON₩CAIRO] 노드에 트랜잭션이 배달되어 성공적으로 반영되었는지 확인합니다.

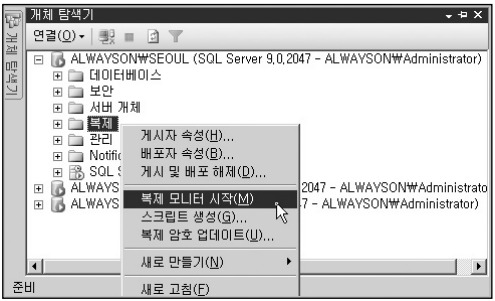
8 단계 : 기준 노드 장애 시 남은 노드간 트랜잭션 배달 확인

현재까지 설정은 첫 번째 노드의 게시 데이터베이스에 대해서 두 번째 노드인 [ALWAYSON₩SEATTLE]와 세 번째 노드인 [ALWAYSON₩CAIRO]가 구독 설정을 각각 첫 번째 노드와 설정한 것입니다.

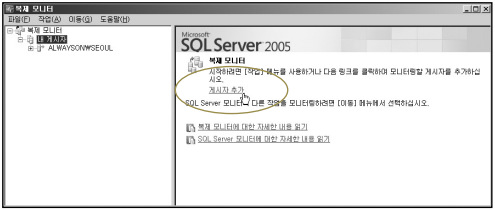


즉 그림과 같이 첫 번째 노드인 [ALWAYSON₩SEOUL]을 연결 고리로 하여 세 노드가 서로 간에 트랜잭션을 배달하고 있습니다. 따라서 이런 구성이라면 첫 번째 노드가 장애를 입게 되면 각 노드는 서로 트랜잭션을 배달하지 못하게 됩니다.

[개체 탐색기]에서 [ALWAYSON₩SEOUL] 인스턴스의 [복제]를 오른쪽 버튼으로 선택한 뒤 [복제 모니터 시작(M)]을 누릅니다.

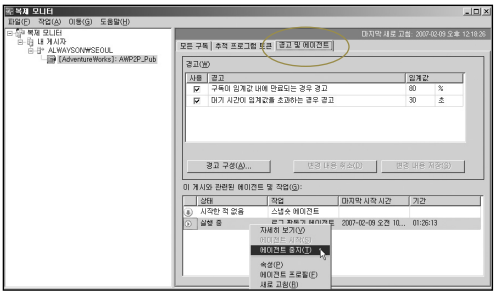


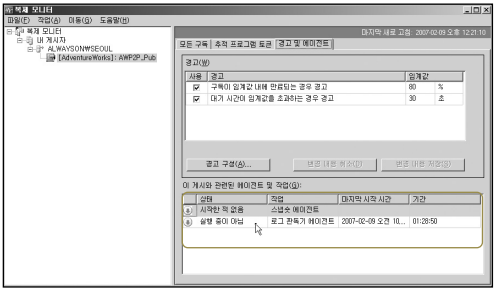
복제 모니터에는 기본적으로 [ALWAYSON₩SEOUL] 게시자가 등록되어 있습니다. 등록되어 있지 않은 경우에는 우측창의 [게시자 추가] 링크를 눌러서 [ALWAYSON₩SEOUL]을 추가합니다.



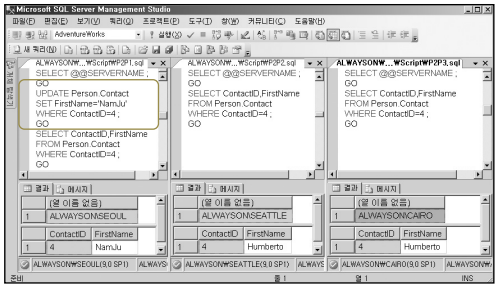
SQL Server 2005에서는 각 복제 에이전트의 중지 및 실행을 복제 모니터를 통해서 수행합니다. 복제 모니터에 자세한 사항은 [8) 복제 모니터링]에서 자세히 살펴보겠습니다.

피어 투 피어 토폴로지의 연결 고리 역할을 수행하는 [ALWAYSON₩SEOUL] 노드의 로그 판독기 에이전트가 정상적인 기능을 수행하지 않도록 [AdventureWorks]:AWP2P_Pub 게시를 선택한 뒤 **[경고 및 에이전트]** 탭에서 **[로그 판독기 에이전트]**를 중지합니다.



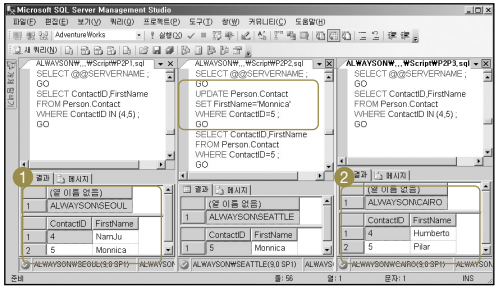


첫 번째 노드 [ALWASYSON₩SEOUL]의 [로그 판독기 에이전트]가 중지 된 것을 확인하고 첫 번째 노드에서 트랜잭션을 수행하고 배달 여부를 각 노드에서 확인합니다.



〈첫 번째 노드의 로그 판독기 에이전트가 중지된 경우 첫 번째 노드의 트랜잭션 배달 확인〉

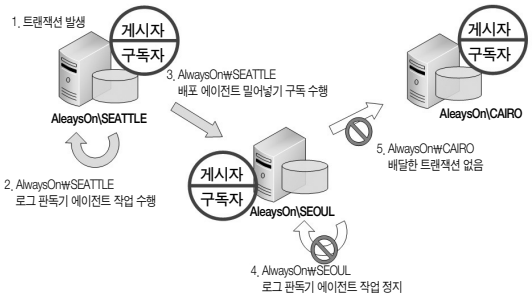
첫 번째 노드의 [로그 판독기 에이전트]가 중지되었으므로 첫 번째 노드에서 실행한 업데이트는 다른 노드로 배달되지 않았음을 확인할 수 있습니다.



〈첫 번째 노드의 로그 판독기 에이전트가 중지된 경우 두 번째 노드의 트랜잭션 배달 확인〉

이어서 두 번째 노드인 [ALWAYSON₩SEATTLE] 노드에서 트랜잭션을 실행해서 해당 트랜잭션이 첫번째 노드인 [ALWAYSON₩SEOUL]과 세 번째 노드인 [ALWAYSON₩CAIRO]로 배달되는지도 확인합니다. 두 번째 노드의 변경 사항은 그림 〈**첫 번째 노드의 로그 판독기 에이전트가 중지된 경우 두 번째 노드의 트랜잭션 배달 확인**〉의 ①번과 같이 첫 번째 노드인 [ALWAYSON₩SEOUL]에는 배달되었으나 ②번과 같이 세번째 노드인 [ALWAYSON₩CAIRO]에는 역시 배달되지 않았습니다.

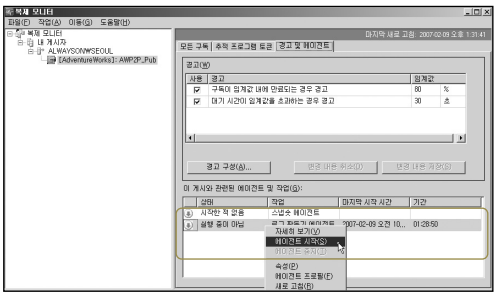
다음 그림을 통해서 수행 단계를 추적하여 원인을 분석해봅니다



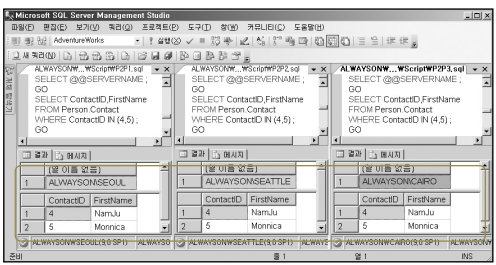
〈피어 투 피어 토폴로지 트랜잭션 배달 프로세스〉

먼저 트랜잭션이 발생한 두 번째 노드 [ALWAYSON₩SEATTLE]에서는 로그 판독기 에이전트가 해당 정보를 읽어와 자신의 배포 데이터베이스인 distribution 데이터베이스의 MSRepl_commands 테이블에 저장합니다. 그러면 해당 노드의 배포 에이전트가 밀어넣기 구독을 통해서 첫 번째 노드인 [ALWAYSON₩SEOUL]에게 트랜잭션을 배달합니다. 따라서 위의 그림과 같이 첫 번째 노드 [ALWAYSON₩SEOUL]에는 트랜잭션이 반영이 됩니다.

이어서 첫 번째 노드 [ALWAYSON₩SEOUL]의 로그 판독기 에이전트가 발생한 트랜잭션을 자신의 배포 데이터베이스 distribution 데이터베이스로 배달해야 하지만 로그 판독기 에이전트가 정지된 상태이므로 자신의 distribution 데이터베이스로 트랜잭션을 배달하지 못하여 나머지 세 번째 노드인 [ALWAYSON₩CAIRO]에는 두 번째 노드의 트랜잭션이 배달되지 않게 됩니다.



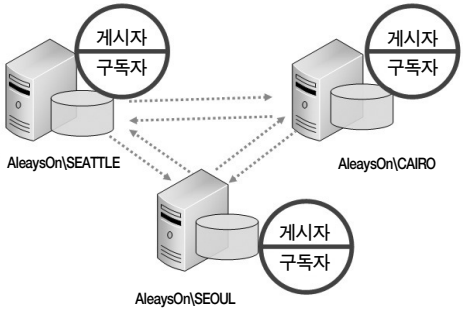
이제 첫 번째 노드인 [ALWAYSON₩SEOUL]에서 **[로그 판독기 에이전트]**를 다시 시작해서 지금까지 배달되지 않은 트랜잭션이 각 멤버 노드간에 상호 배달되는지 확인합니다.



첫 번째 노드에서 **[로그 판독기 에이전트]**를 중지한 뒤 실행한 첫 번째 노드의 트랜잭션과 두 번째 노드에서 수행한 트랜잭션이 상호간에 모두 잘 전파되었음을 확인할 수 있습니다.

9 단계 : 내 결함성 향상을 위한 구독 추가

앞에서 살펴본 바와 같이 세 개의 멤버 노드로 구성된 피어 투 피어 트랜잭션 복제는 연결 고리 역할을 하는 첫 번째 노드가 정상적으로 작업을 수행하지 못하는 경우에 첫 번째 노드에서 발생한 트랜잭션뿐만 아니라 다른 노드에서 발생한 트랜잭션까지도 상호간에 배달되지 않음을 확인할 수 있었습니다.



이와 같은 장애 상황을 극복하고 내 결함성을 향상하기 위해서는 그림과 같이 각 멤버 피어 노드 간에 모두 추가로 구독을 설정해야 합니다.

> ***[주의]***
> *[6) 피어 투 피어 트랜잭션 복제 구성 시 고려 사항]에서 살펴본 바와 같이 기존 토폴로지에 노드를 추가하거나 기존 토폴로지에 아티클 추가 또는 스키마 변경, 백업에서 노드 복원 등의 작업을 수행하는 경우에 시스템을 정지하고 기존의 모든 작업이 각 피어 투 피어 토폴로지 멤버 노드에 변경 사항이 반영되었는지 확인하는 작업을 수행해야 합니다.*

먼저, 피어 투 피어 토폴로지에 추가 구독을 설정하고자 하는 경우에 게시된 테이블에 대한 모든 작업을 중지합니다. 이어서 다음 작업을 수행합니다.

① 피어 투 피어 토폴로지의 모든 멤버 노드의 데이터베이스에서 sp_requestpeerresponse 시스템 저장 프로시저를 실행하고 출력 매개 변수 @request_id를 검색합니다.

• sp_requestpeerresponse 시스템 저장 프로시저 구문

```
DECLARE @request_id int
EXEC sp_requestpeerresponse  '〈게시 이름〉', '〈배달 메시지〉', @request_id
OUTPUT
SELECT @request_id
GO
```

② 기본적으로 배포 에이전트는 연속적으로 실행되도록 설정되므로 토큰이 모든 노드로 자동 배달됩니다.
③ sp_helppeerresponse 시스템 저장 프로시저를 실행한 다음 이전 단계에서 검색된 @request_id 값을 제공하여 모든 노드가 피어 요청을 받았음을 확인할 때까지 기다립니다.

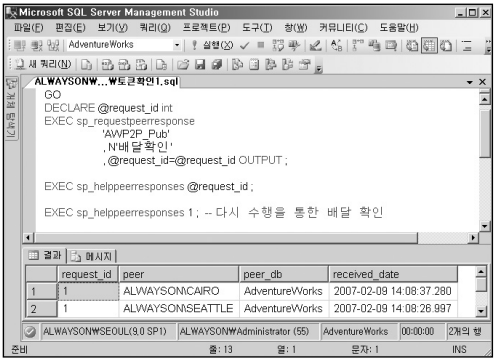
• sp_helppeerresponse 시스템 저장 프로시저 구문

```
EXEC sp_helppeerresponse @request_id
```

* 이 때 @request_id는 sp_requestpeerresponse 시스템 저장 프로시저를 사용한 뒤 반환 받은 출력 매개 변수의 결과임

④ 첫 번째 노드에서 만든 게시를 오른쪽 버튼으로 클릭한 다음 **[피어 투 피어 토폴로지 구성]**을 선택합니다.
⑤ 피어 투 피어 토폴로지 구성 마법사의 게시 페이지에서 첫 번째 노드에서 만든 게시를 선택합니다.

⑥ **[피어]** 페이지에서 두 번째 노드를 추가하고 게시 데이터베이스를 선택합니다.
⑦ 세 번째 노드도 마찬가지로 추가하고 게시 데이터베이스를 선택합니다.
⑧ **[로그 판독기 에이전트 보안]** 및 **[배포 에이전트 보안]** 페이지에서 자격 증명을 지정합니다.
⑨ **[새 피어 초기화]** 페이지에서 **[피어 데이터베이스를 수동으로 만들었거나 원래 게시 데이터베이스의 백업을 복원했고 백업 이후에 데이터베이스가 변경되지 않았습니다.]**를 선택합니다. 모든 노드에 이미 데이터가 있지만 이 옵션을 지정하면 각 노드 간에 적절한 구독 관계가 설정됩니다.
⑩ 마법사를 완료합니다.

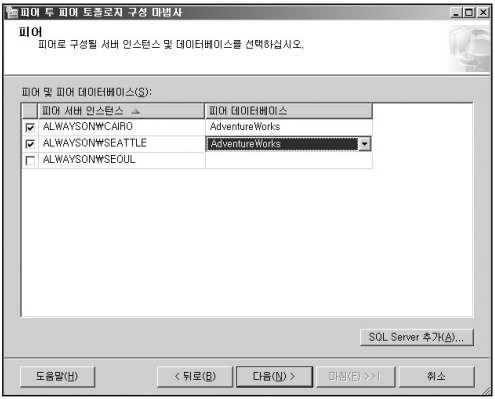


그림과 같이 sp_requestpeerresponse 시스템 저장 프로시저를 수행하고 이어서 sp_helppeerresponses 시스템 저장 프로시저를 실행하여 모두 이상 없이 처리되었는지 확인합니다.

sp_helppeerresponses 시스템 저장 프로시저의 실행 결과의 [received_date] 컬럼에 응답을 받은 시간이 기록되야 합니다. 어느 하나의 노드라도 sp_requestpeerresponses 시스템 저장 프로시저의 응답을 받지 않은 경우에 받을 때까지 기다려야 합니다. 혹시 **[배포 에이전트]**가 중지 중이면 시작해야 합니다.

다음 그림과 같이 첫 번째 노드인 [ALWAYSON₩SEOUL] 서버 인스턴스에서 **[개체 탐색기]**의 **[복제]**, **[로컬 게시]**를 차례로 확장하고 [AdventureWorks]:AWp2P_Pub 게시를 선택하고 오른쪽 버튼을 눌러서 **[피어 투 피어 토폴로지 구성(C)]**을 선택하여 **[피어 투 피어 토폴로지 마법사]**를 실행합니다.

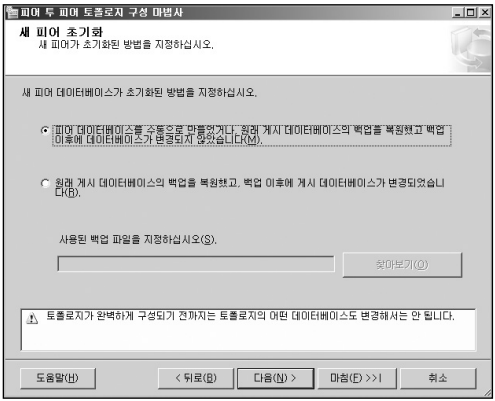
[게시] 페이지에서 게시자를 [ALWAYSON₩SEOUL]로 선택하고 AdventureWorks 데이터베이스의[AWp2P_Pub] 게시를 선택하고 **[다음(N) >]** 버튼을 누릅니다.



[피어] 페이지에서 첫 번째 노드인 [ALWAYSON₩SEOUL]를 제외한 두 번째 노드인 [ALWAYSON₩SEATTLE]와 세 번째 노드인 [ALWAYSON₩CAIRO]를 모두 선택하고 연결합니다. **[피어 데이터베이스]** 컬럼에 AdventureWorks 데이터베이스가 자동으로 지정되는 것을 확인하고 **[다음(N) >]** 버튼을 누릅니다.

> ***[주의]***
> *피어 투 피어 토폴로지의 내 결함성을 위해서는 [피어 투 피어 토폴로지 마법사]의 [피어] 피이지에서 첫 번째 노드인 [ALWAYSON₩SEOUL]를 제외한 모든 노드를 선택하여 지정해야 합니다. 첫 번째 노드인 [ALWAYSON₩SEOUL]는 지정하지 않습니다.*

[로그 판독기 에이전트 보안] 및 [배포 에이전트 보안] 페이지에서 자격 증명을 지정합니다.



[새 피어 초기화] 페이지에서 [피어 데이터베이스를 수동으로 만들었거나 원래 게시 데이터베이스의 백업을 복원했고 백업 이후에 데이터베이스가 변경되지 않았습니다.]를 선택하고 [다음(N) >] 버튼을 누릅니다.

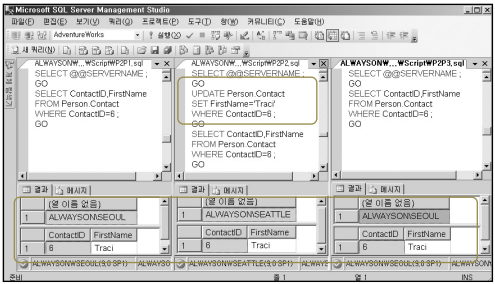
[마법사 완료 페이지]에서 [피어 투 피어 토폴로지 마법사]에서 설정한 사항을 확인합니다. 첫 번째 노드를 제외한 나머지 멤버 노드에 필요한 게시와 구독을 모두 만들도록 지정했는지를 다시 한번 확인합니다.

그림에서와 같이 총 9개의 작업 가운데 2개의 작업만 성공하고 나머지는 이미 만들어져 있으므로 변경하지 못했다는 경고가 생성되었습니다. 성공한 작업 2가지는 다음과 같습니다.

- 세 번째 피어 노드인 [ALWAYSON₩CAIRO]에서 [AdvnetureWorks] 데이터베이스에서 두 번째 피어 노드인 [ALWAYSON₩SEATTLE]의 AdventureWorks **데이터베이스**에 대한 구독 생성

- 두 번째 피어 노드인 [ALWAYSON₩SEATTLE]에서 AdventureWorks **데이터베이스**에서 세 번째 피어 노드인 [ALWAYSON₩CAIRO]에서 [AdvnetureWorks] 데이터베이스에 대한 구독 생성

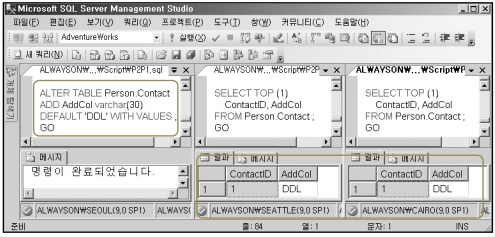
이제 각 피어 노드간에 게시와 구독이 만들어졌으므로 내 결함성을 갖는 피어 투 피어 토폴로지가 완성되었습니다. 내 결함성 여부를 확인하기 위해서 앞에서 수행한 것과 같이 첫 번째 피어 노드인 [ALWAYSON₩SEOUL]의 **[로그 판독기 에이전트]** 중료합니다.



첫 번째 피어 노드인 [ALWAYSON₩SEOUL]의 [로그 판독기 에이전트]가 종료된 상태에서 두 번째 피어 노드인 [ALWAYSON₩SEATTLE]에서 발생한 트랜잭션이 첫 번째 노드인 [ALWAYSON₩SEOUL]과 세 번째 피어 노드인 [ALWAYSON₩CAIRO]에 배달되는지 확인합니다. 그림과 같이 이상 없이 배달 되었다면 내 결함성을 갖는 피어 투 피어 토폴로지가 완성된 것입니다. 이 때 만약 [ALWAYSON₩SEOUL] 노드에서 트랜잭션이 발생한다면 해당 트랜잭션은 다른 피어 노드로 배달되지 않습니다.

10 단계 : DDL 구문 배달 확인

SQL Server 2005 복제는 DDL 구문을 배달합니다. 다음과 같이 게시된 아티클인 [Person.Contact] 테이블에 컬럼을 추가하는 DDL 구문을 실행하고 배달 여부를 확인합니다.



첫 번째 피어 노드에서 [AddCol이란 컬럼을 'DDL' 이라는 기본값과 함께 컬럼을 추가하는 DDL구문을 실행하고 다른 멤버 노드에서 컬럼이 추가 되었는지 확인합니다.

> ***[주의]***
> *게시자에서 수행한 스키마 변경 사항인 DDL 구문이 구독자로 전달되기 위해서는 게시 속성의 **[구독 옵션]**페이지에서 **[스키마 변경 내용 복제]**옵션의 설정 값을 [True]로 선택합니다.*

8) 복제 모니터링

① 복제 모니터

복제 모니터는 복제 토폴로지의 전체적인 상태와 성능을 모니터링할 수 있는 그래픽 도구로 모든 복제 작업의 게시자를 중심으로 두 개 창의 형식으로 보기를 제공합니다. 좌측 창에서 복제 모니터에 게시자를 추가하면 우측 창에 게시자, 해당 게시, 게시에 대한 구독 및 다양한 복제 에이전트 정보가 표시됩니다. 복제 토폴로지에 대한 정보를 제공하는 것 외에도 복제 모니터를 사용하여 에이전트 시작 및 중지, 데이터 유효성 검사 등 다양한 작업을 수행할 수 있습니다. 복제 모니터를 통해서는 다음과 같은 사항을 확인하고 수행할 수 있습니다.

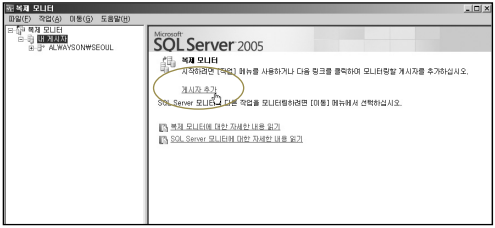
■ 전체 토폴로지 정보
- 복제 시스템이 정상적으로 작동 중인지 여부 확인
- 복제 에이전트가 실행되지 않는 이유 확인

■ 게시자 관련 정보
- 복제 시스템이 정상적으로 작동하는 지 여부
- 느린 구독 성능 상태 확인

■ 게시 관련 정보 보기 및 작업 수행
- 게시와 연결된 모든 에이전트에 대한 상태 확인 및 경고 지정
- 트랜잭션 복제 시 커밋된 트랜잭션이 구독자에게 배달되는 데 소요되는 시간 추정
- 게시에 대한 모든 구독 초기화 및 게시에 대한 구독 유효성 검사 등

■ 구독 관련 정보
- 게시자에서 배포자, 배포자에서 구독자로 연결 기록
- 배포되지 않은 명령, 동기화 기록 정보 등

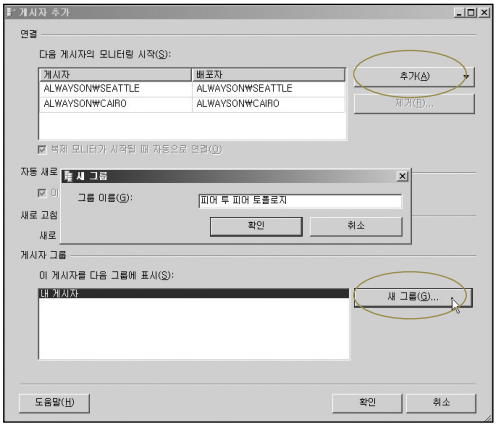
■ 에이전트 프로필 관련 정보 보기 및 작업 수행
- 프로필 속성 변경, 프로필 만들기 및 삭제, 사용하고자 하는 프로필 지정



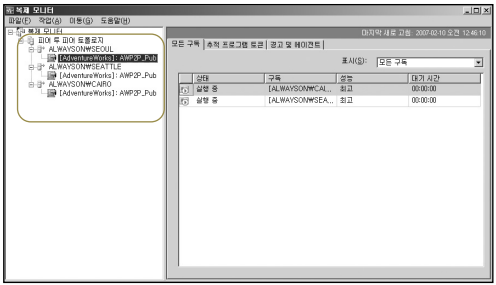
■ 게시자 추가

복제 모니터를 실행하는 서버가 게시자인 경우 자동으로 모니터에 추가됩니다. 복제 모니터의 오른쪽창이나 **[작업(A)]** 메뉴의 **[게시자 추가]** 대화 상자를 통해 게시자를 추가할 수 있습니다.

게시자를 추가하면 게시자는 모니터의 좌측 창에 있는 그룹에 표시됩니다. 기본적으로는 이미 만들어져 있는 **[내 게시자]** 그룹에 추가하는 게시자가 포함되지만 **[새 그룹(G)]** 버튼을 눌러서 새 그룹을 만들어 하나 이상의 복제 토폴로지를 관리하도록 합니다

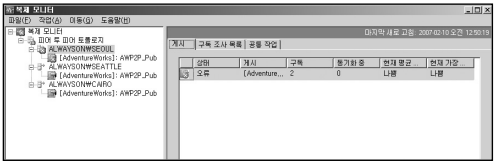


[추가] 버튼을 눌러서 모니터링할 게시자를 추가한 뒤 [새 그룹(G)] 버튼을 눌러서 [피어 투 피어 토폴로지]라는 이름으로 새 그룹을 생성하고 선택한 뒤 [확인] 버튼을 누릅니다. 이미 추가되어 있던 [내 게시자] 그룹의 [ALWAYSON₩SEOUL] 게시자를 제거하고 새로 만든 그룹 [피어 투 피어 토폴로지]에 추가합니다.

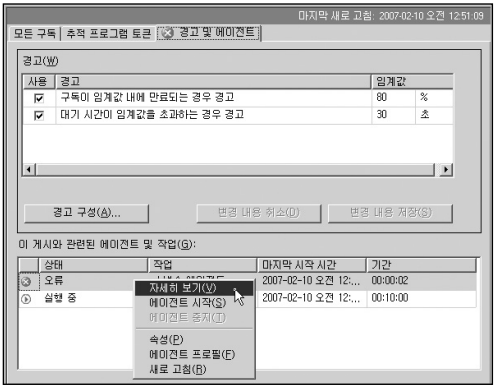


■ 전체 토폴로지 상태 확인

좌측 창의 노드에 오류가 발생하면 다음과 같이 해당 게시와 노드에 오류 아이콘이 표시됩니다.

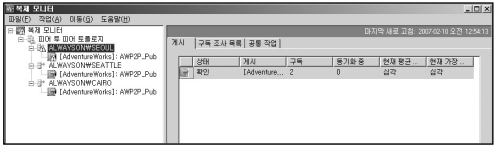


시스템 상태를 더 자세히 보려면 주의가 필요한 구독 정보가 표시되는 **[구독 조사 목록]** 탭도 확인해야 합니다



에이전트가 실행되도록 예약하지 않았거나 오류가 발생하여 특정 시간에 에이전트가 중지된 경우에도 오류가 발생해서 좌측 창의 해당 노드에 오류 아이콘이 표시됩니다. 그림과 같이 **[스냅숏 에이전트]**가 오류로 인해 중지되면 게시자 그룹, 게시자 및 게시 노드에 오류 아이콘이

표시되며 해당 게시를 선택한 뒤 우측 창에서 **[경고 및 에이전트]** 탭의 오류가 발생한 에이전트를 더블 클릭하거나 오른쪽 버튼을 누르고 **[자세히 보기(V)]**를 선택하면 자세한 오류 정보를 확인할 수 있습니다.



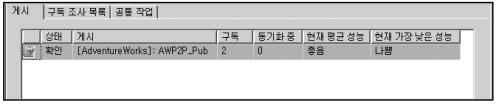
또한, 좌측 창과 **[구독 조사 목록]** 탭 등에서 성능 상태가 심각한 경우에 경고 아이콘이 표시됩니다.

■ 게시자 관련 정보 보기 및 작업 수행

복제 모니터의 좌측 창에서 게시자를 선택하면 **[게시]**, **[구독 조사 목록]**, **[공통 작업]** 3개의 탭에 게시자 정보를 표시합니다.

• **[게시]** 탭

게시자에 있는 모든 게시의 요약 정보를 제공합니다.



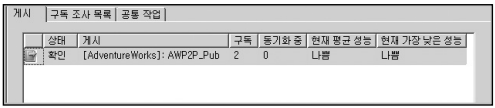
게시에 대해서는 게시자의 **[게시]** 탭의 **[현재 평균 성능]** 및 **[현재 가장 낮은 성능]** 컬럼에, 구독에 대해서는 게시자의 **[구독 조사 목록]** 탭이나 게시의 **[모든 구독 목록]** 탭의 **[성능]** 컬럼에 트랜잭션 복제 및 병합 복제에 대한 성능 상태를 표시합니다. 성능 상태는 최고, 좋음, 보통, 나쁨, 심각(트랜잭션 복제)으로 나누어 지면 이 값들은 다음과 같이 결정됩니다.

〈트랜잭션 복제〉

게시의 **[경고 및 에이전트]** 탭에서 **[대기 시간이 임계값을 초과하는 경우 경고]**의 임계값을 설정하는 경우에 표시되며 이 대기 시간을 기준으로 다음과 같이 결정됩니다.

최고	좋음	보통	나쁨	심각
0~34%	35~59%	60~84%	85~99%	100%+

예를 들어 이 임계값이 30초인 경우 게시의 경우 로그 판독기 에이전트 대기 시간이 구독의 경우 배포 에이전트 대기 시간이 약 10초 이내인 경우에 [최고]로 약 25.5초~30초 미만인 경우에는 다음 그림과 같이 나쁨으로 표시됩니다.



〈병합 복제〉

병합 복제의 경우 성능 품질은 임계값과 관련이 없으며 동일한 유형의 연결을 통해 각각 50개 이상 변경 내용이 포함된 동기화가 5회 수행된 후에 값을 표시합니다. 50개 이상 변경 내용이 포함된 동기화가 5회 미만이거나 최신 동기화의 변경 내용 수가 50개 미만이면 복제 모니터에서 값을 표시하지 않습니다. **[성능]** 열에 표시되는 값은 구독자가 초당 동일한 유형의 구독이 행을 처리하는 평균 속도와 개별 구독이 행을 처리하는 속도를 비교해서 결정합니다.

최고	좋음	보통	나쁨
151%+	76~150%	26~75%	0~25%

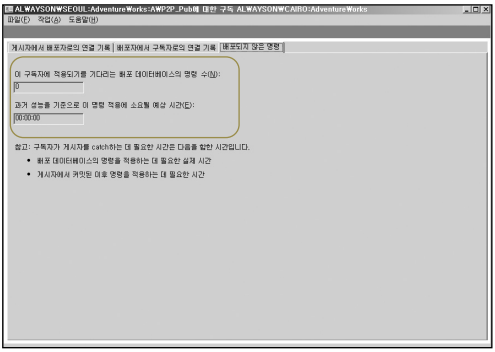
예를 들어 평균 속도가 초당 100개의 행을 처리하고 특정 구독이 초당 160개를 처리한다면 해당 구독자의 동기화 성능은 평균 160%가 되어 값은 최고가 됩니다.

• **[구독 조사 목록]** 탭

선택한 게시자에서 사용 가능한 모든 게시의 구독 정보가 표시됩니다. **[트랜잭션 구독 표시]** 부분에서는 **[스냅숏 구독]**, **[트랜잭션 구독]**, **[병합 구독]** 등 구독 유형별로 지정하고, **[표시(S)]** 부분에서는 구독 목록을 필터링하여 볼 수 있으며 에이전트의 실행 여부나 게시의 성능 상태와 마찬 가지로 구독의 성능 상태, 오류, 경고 등을 확인할 수 있습니다.

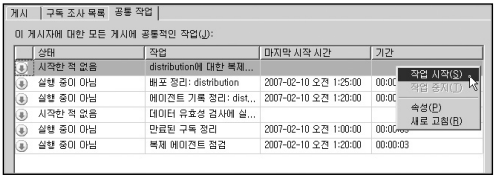
또한 구독 속성이나 구독과 연결된 **[배포 에이전트 프로필]**에 대한 자세한 정보에 액세스하거나 수정할 수 있으며 구독을 다시 초기화하는 등의 작업을 수행할 할 수 있습니다.

[자세히 보기(V)]를 통해서는 **[게시자에서 배포자로의 연결 기록]** 탭과 **[배포자에서 구독자로의 연결기록]** 탭에서 게시자와 배포자간의 연결 기록 및 배포자와 구독자간의 연결 기록을 확인할 수 있고, 트랜잭션 복제의 경우에 **[배포되지 않은 명령]** 탭에서 배포되지 않은 명령의 수와 배포 되지 않은 명령이 배포되는 데 걸리는 예상 소요 시간 등을 확인할 수 있습니다.



• **[공통 작업]** 탭

배포자 서버에서 수행되는 공통 작업 정보가 표시됩니다. 각 작업에 대한 자세한 작업 정보에 액세스하거나 각 작업을 시작하고 중지할 수 있습니다.



■ 게시관련 정보 보기 및 작업 수행

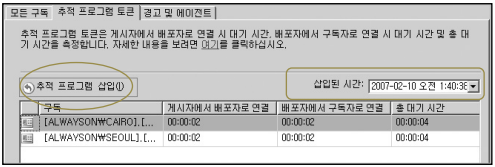
• **[모든 구독]** 탭

선택한 게시에 대한 모든 구독 정보가 표시되며 기본적으로 우선 순위 순서대로 정렬됩니다. 오류, 경고가 표시된 후 성능의 오름차순으로 성능이 저조한 구독이 맨 위에 표시됩니다. **[표시(S)]** 부분에서는 구독 목록을 모든 구독 및 오류, 경고 및 구독의 성능 상태, 실행 중이지 않은 구독 등 다양하게 필터링하여 을 볼 수 있습니다.

또한, 게시자의 **[구독 조사 목록]** 탭과 같이 구독 속성이나 구독과 연결된 **[배포 에이전트 프로필]**에 대한 자세한 정보에 액세스하거나 수정할 수 있으며 구독을 다시 초기화하는 등의 작업을 수행할 할 수 있습니다. **[자세히 보기(V)]**를 통해서는 게시자와 배포자간의 연결 기록 및 배포자와 구독자간의 연결 기록을 확인할 수 있고, 배포되지 않은 명령의 수와 배포 되지 않은 명령이 배포되는 데 걸리는 예상 소요 시간 등을 확인할 수 있습니다.

• **[추적 프로그램 토큰]** 탭

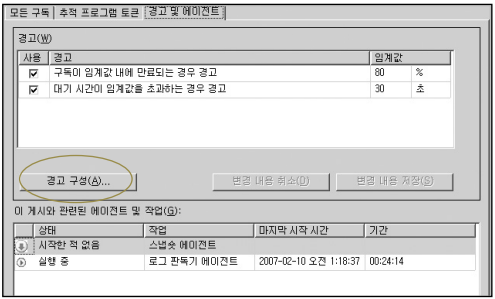
SQL Server 2005는 추적 프로그램 토큰을 통해서 트랜잭션 복제 수행 시 게시자에서 발생한 트랜잭션이 배포자를 통해서 다시 구독자로 배달될 때까지 걸린 대기 시간 및 총 대기 시간을 측정하고 각 연결의 유효성을 검사하는 기능을 제공합니다.



[추적 프로그램 삽입(I)] 버튼을 눌러서 추적 프로그램 토큰을 삽입하고 **[게시자에서 배포자로 연결]** 컬럼과 **[배포자에서 구독자로 연결]** 컬럼을 통해서 소요 시간을 통해서 대기 시간을 모니터링합니다. **[추적 프로그램 삽입(I)]** 버튼을 누를 때마다 새로운 토큰이 삽입되어 작업을 수행하고 최근 삽입된 시간은 **[삽입된 시간]** 부분에 표시됩니다.

• **[경고 및 에이전트]** 탭

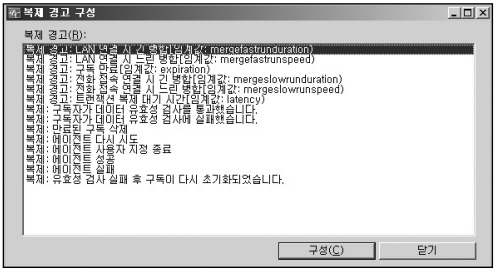
[경고 및 에이전트] 탭에서 경고를 지정하면 지정된 경고가 발생하면 해당 기록을 SQL Server 오류 로그에 기록하며 담당자에게 Net Send 메시지나 전자 메일 등으로 통보하고 지정된 작업을 수행할 수 있습니다.



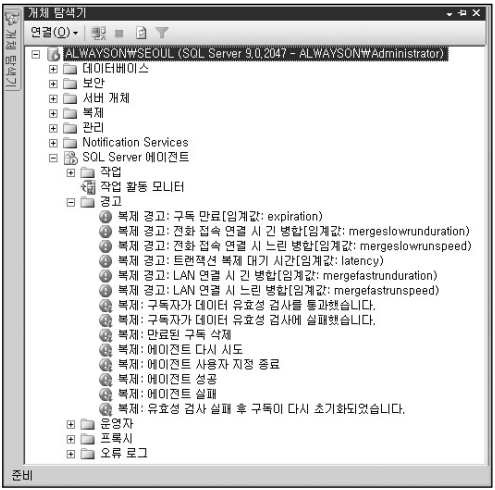
기본적으로는 **[구독이 임계값 내에 만료되는 경우 경고]**에 대해서만 경고를 표시하지만, 트랜잭션 복제의 경우 **[대기 시간이 임계값을 초과하는 경우 경고]**에 대해서도 기본적으로 표시하여 경고 임계값을 지정하고 사용할 수 있으며 이 이외의 다른 조건의 경고도 지정할 수 있습니다.

[대기 시간이 임계값을 초과하는 경우 경고]에 대한 임계값에 따라서 게시자의 **[게시]** 탭의 **[현재 평균 성능]** 및 **[현재 가장 낮은 성능]** 컬럼에, 구독에 대해서는 **[구독 조사 목록]** 탭이나 게시의 **[모든 구독 목록]** 탭의 **[성능]** 열에 성능 상태를 표시합니다. 지정한 임계값에 대한 경고를 지정하기 위해서는 **[경고(W)]** 부분의 하단부의 **[경고 구성(A)]** 버튼을 눌러서 **[복제 경고 구성]** 대화 상자를 실행하고 구성하고자 하는 경고를 선택합니다.

> ***[참고]***
> *경고 발생 시 운영자에게 전자 메일을 통한 통보를 구성하는 사항은 필자가 집필한 도서출판 대림의 [SQL Server 2005 재해 복구 및 고가용 솔루션] 본문 [2장 백업 6) 백업의 자동화]에서 [데이터베이스 메일 설정]부분을 참조하기 바랍니다.*



이와는 별도로 **[개체 탐색기]**에서 SQL Server 에이전트의 **[경고]**에서도 동일한 작업을 수행할 수 있습니다.



■ 구독 관련 정보 보기 및 작업 수행

구독관련 정보를 확인하고 관련 작업을 수행하기 위해서는 좌측 창에서 게시자를 선택하고 우측 창의 **[구독 조사 목록]** 탭을 사용하거나 좌측 창에서 게시를 선택하고 우측 창의 **[모든 구독]** 탭을 사용합니다. 조사하기 위한 구독을 선택하고 **[자세히 보기(V)]**를 선택하면 다음과 같은 정보를 확인할 수 있습니다.

- **[게시자에서 배포자로의 연결 기록]** 탭은 트랜잭션 복제에서만 표시되며 로그 판독기 에이전트 정보가 표시됩니다.

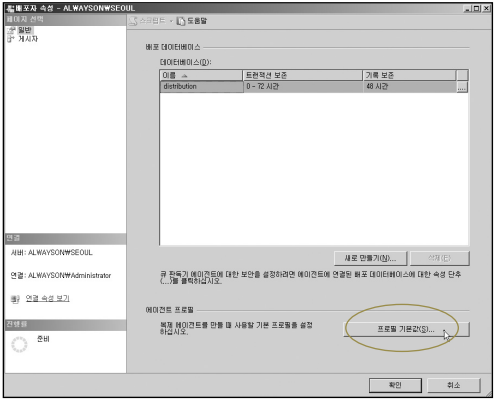
- **[배포자에서 구독자로의 연결 기록]** 탭은 스냅숏 복제 및 트랜잭션 복제에서 확인할 수 있으며 구독에 대한 배포 에이전트 정보가 표시됩니다.

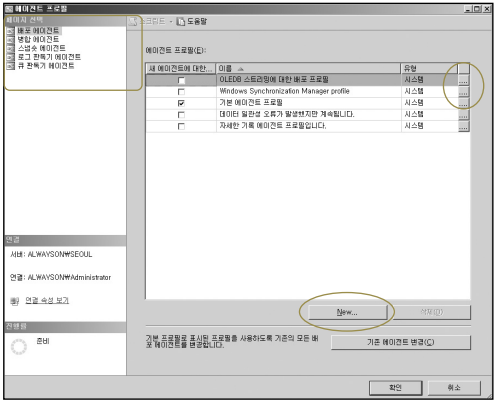
- **[배포되지 않은 명령]** 탭은 트랜잭션 복제에서만 표시되며 선택한 구독자에게 배달되지 않은 데이터베이스 명령의 수와 해당 명령의 예상 배달 시간에 대한 정보가 표시됩니다.

- **[동기화 기록]** 탭은 병합 복제에서만 표시되며 병합 복제의 각 처리 단계에 소요된 시간을 포함하여 동기화 중에 처리된 각 아티클에 대한 자세한 통계를 제공합니다.

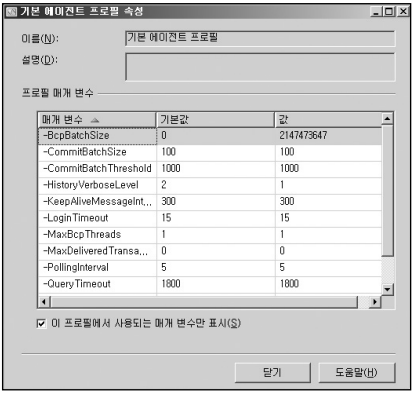
■ 에이전트 프로필 관련 정보 보기 및 작업 수행

에이전트 프로필은 에이전트 동작을 결정하는 에이전트에 대한 매개 변수 집합으로 복제 모니터에는 에이전트 프로필을 관리하기 위해 다양한 경로를 제공하고 있습니다만 가장 편리한 방법은 **[개체 탐색기]**의 **[복제]**에서 오른쪽 버튼을 누르고 **[배포자 속성(B)]**를 선택하여 **[배포자 속성]** 창을 실행하고 **[일반]** 페이지 하단 부의 **[프로필 기본값(S)]** 버튼을 누릅니다.





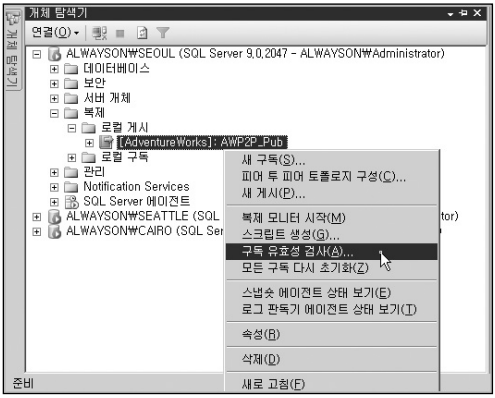
[에이전트 프로필] 창이 실행되면 좌측 창에서 각 에이전트를 선택하여 해당 에이전트가 사용중인 프로필의 정보를 확인할 수 있으며 [· · ·] 버튼을 눌러서 해당 에이전트 프로필 속성 창을 실행하여 에이전트 프로필의 매개 변수를 확인할 수 있습니다.



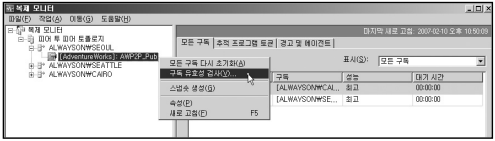
만약, 기본 에이전트 프로필의 매개 변수 값을 변경하고자 하는 경우에는 **[에이전트 프로필]** 창 하단 부분의 **[New]** 버튼을 눌러서 별도의 사용자 프로필을 만들어서 설정합니다.

② 복제 유효성 검사

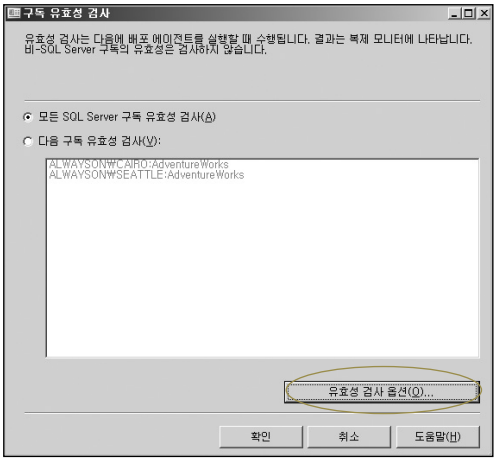
복제 에이전트가 일시 중단되었다 다시 실행되는 경우 구독의 유효성을 점검해 볼 필요가 있습니다. 물론 **[복제 모니터]**에서 해당 게시자를 선택하고 **[구독 조사 목록]**에서 **[자세히 보기(V)]**를 선택한 뒤 **[배포자에서 구독자로의 연결 기록]** 탭이나 **[배포되지 않은 명령]** 탭에서 구독 상태를 확인 할 수 있지만 다음과 같은 방법으로 구독의 유효성을 검사해서 자세한 상황을 확인할 수 있습니다.



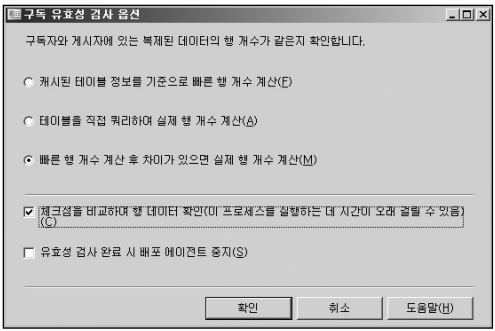
[개체 탐색기]에서 **[서버], [복제], [로컬 게시]**를 차례로 확장하고 게시를 오른쪽 버튼으로 누른 뒤 **[구독 유효성 검사(A)]**를 선택합니다.



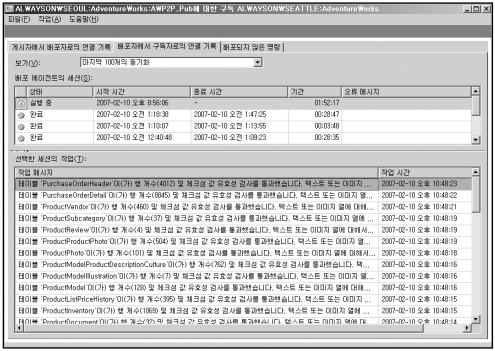
또는 **[복제 모니터]**에서 해당 구독을 오른쪽 버튼으로 선택한 뒤 **[구독 유효성 검사(V)]**를 선택합니다.



[구독 유효성 검사] 창에서 **[모든 SQL Server 구독 유효성 검사(A)]**를 선택하거나 **[다음 구독 유효성 검사(V)]**를 선택하고 바로 밑에 표시된 구독 가운데에서 특정 구독을 지정한 뒤 하단의 **[유효성 검사 옵션(O)]** 버튼을 누릅니다. 유효성 검사 옵션은 캐시된 정보를 기준으로 행 개수만 비교할 것인지 테이블에 직접 쿼리하여 실제 행 개수를 비교할 것인지 또는 먼저, 캐시된 정보를 기준으로 행 개수를 비교하고 일치하지 않는 경우에 실제 행 개수를 비교할 것인지를 선택하고 행 개수뿐만 아니라 행 데이터도 비교할 것인지도 지정할 수 있습니다.



구독 유효성 검사 옵션을 지정한 뒤 **[확인]** 버튼을 누르고 다음과 같이 **[복제 모니터]**에서 **[배포자에서 구독자로의 연결 기록]** 탭에서 유효성 검사 결과를 확인합니다.



③ TABLEDIFF 유틸리티를 통한 게시와 구독 데이터베이스 비교 및 동기화

만약, 앞에서 실시한 유효성 검사에서 특정 테이블이 유효성 검사를 통화하지 못했다면 [tablediff] 유틸리티를 통해서 테이블 별로 불일치 세부 사항을 조사할 수 있으며 동기화를 위한 스크립트도 생성할 수 있습니다. [tablediff] 유틸리티는 두 테이블간에 데이터를 비교하는데 사용하는 명령 프롬프트 유틸리티로 특히 복제 토폴로지에서 데이터 불일치 문제를 해결하는 데 유용하며 다음과 같은 기능을 합니다.

- 복제에서 원본 아티클인 테이블과 구독자의 아티클인 테이블을 행 단위로 비교합니다.
- **-q 옵션**을 사용하면 행 개수와 스키마만 비교하여 비교 작업을 빠르게 수행할 수 있습니다.
- **-c 옵션**을 사용하면 컬럼 수준에서 비교 작업을 수행할 수 있습니다.
- **-f [파일 이름]** 옵션을 사용하면 원본 테이블과 대상 테이블을 일치 시킬 수 있는 T-SQL 스크립트를 생성합니다.
- 수행 결과를 출력 파일이나 대상 데이터베이스의 테이블에 기록할 수 있습니다.

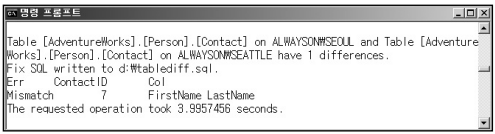
따라서 피어 투 피어 토폴로지가 오류 등으로 일부 동기화가 되지 않는 경우가 생기면 다음과 같이 [tablediff] 유틸리티를 통해서 불일치하는 사항을 비교하고 동기화 스크립트를 통해서 수동으로 동기화할 수 있습니다.

■ 원본 아티클과 대상 아티클 비교하고 동기화 스크립트 생성하기

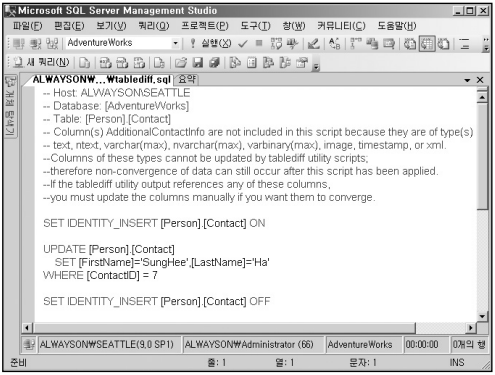
다음 tablediff 구문과 같이 명령 프롬프트에서 [C:₩Program Files₩Microsoft SQL Server ₩90₩COM] 경로에서 실행합니다. [ALWAYSON₩SEOUL] 피어 노드와 [ALWAYSON ₩SEATTLE] 피어 노드의 Person.Contact 테이블을 비교하고 불일치 하는 사항을 [D: ₩tablediffscript.sql] 파일에 저장합니다.

```
tablediff -sourceserver ALWAYSON₩SEOUL -sourcedatabase AdventureWorks
          -sourcetable Contact
          -sourceschema Person -sourcepassword Pa$$w0rd -sourceuser sa
          -destinationserver ALWAYSON₩SEATTLE -destinationdatabase
           AdventureWorks
          -destinationtable Contact
          -destinationschema Person -destinationpassword Pa$$w0rd
          -destinationuser sa
          -f D:₩tablediffscript.sql
```

굵은 글씨로 표시한 부분은 반드시 지정해야 하는 매개 변수입니다. 데이터베이스별로 또는 스키마 별로 지정할 수 있는 옵션은 제공되지 않습니다. 안타깝습니다.



[tablediff] 유틸리티의 실행 결과는 ALWAYSON₩SEATTLE 피어 노드에서 [Adventure Works.Person.Contact] 테이블의 ContactID가 7인 행의 [FistName] 컬럼과 [LastName]컬럼에서 하나의 불일치가 발견되었다고 보고하고 있습니다.



[D:₩tablediffscript.sql] 파일은 다음과 같이 불일치하는 행을 동기화할 수 있는 내용의 T-SQL 구문을 포함합니다.

④ 피어 투 피어 복제 성능 카운터

Windows 시스템 성능 모니터에서 [SQLServer:Replication Agents], [SQLServer:Dist.], [SQLServer:LogReader] 성능 개체를 통해서 피어 투 피어 복제와 관련한 성능을 모니터링 할 수 있습니다. 다음은 각 에이전트 별 성능 개체 및 카운터입니다.

■ [SQLServer:Replication Agents]

성능 카운터	설 명
Runing	현재 실행 중인 복제 에이전트 수입니다. Distribution, LogReader 인스턴스를 모니터합니다.

■ [SQLServer:Dist.]

성능 카운터	설 명
Dist:Delivered Cms/Sec	구독자로 배달된 초당 명령 수입니다.
Dist:Delivered Trans/Sec	구독자로 배달된 초당 트랜잭션 수입니다.
Dist:Delivery Latency	트랜잭션이 배포자로 배달된 시간부터 구독자에 적용된 시간까지 경과된 시간(밀리초)입니다.

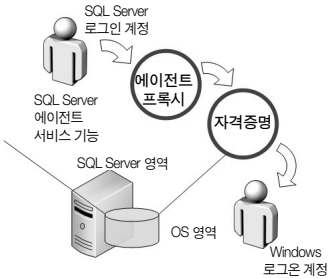
■ [SQLServer:LogReader]

성능 카운터	설 명
LogReader:Delivered Cms/Sec	배포자로 배달된 초당 명령 수입니다.
LogReader:Delivered Trans/Sec	배포자로 배달된 초당 트랜잭션 수입니다.
LogReader:Delivery Latency	트랜잭션이 게시자에 적용된 시간부터 배포자로 배달된 시간까지 경과된 시간(밀리초)입니다.

9) 복제 에이전트 계정 보안 강화

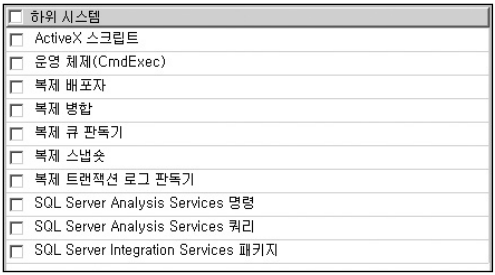
SQL Server 2005는 이전 버전에 비해서 엄청나게 보안을 강화하였습니다. 특히 자동화 작업을 위해서 각 작업 마다 프록시를 설정할 수 있습니다. 프록시란 SQL Server 에이전트 작업 단계의 보안 컨텍스트에 대한 정의로 Windows 서버와 상호 작용을 하는 경우에 지정하게 됩니다. 이전 버전에서는 sysadmin 고정 서버 역할의 멤버는 SQL Server 에이전트 서비스의 로그온 계정을 통해서 sysadmin 고정 서버 역할의 멤버가 아닌 계정은 별도로 프록시 계정을 하나만 지정하여 운영 체제와 상호 작용하는 작업을 수행하였지만 SQL Server 2005는 수행할 작업을 위해서 필요한 운영 체제 권한에 따라서 각각의 작업 별로 적절한 Windows 계정과 연결된 **[자격 증명(credential)]**을 생성하고, 이 자격 증명과 연결된 **[프록시]**를 생성하여 보안을 강화합니다.

즉 특정 작업에 프록시를 지정하는 경우에 다음 그림과 같이 SQL Server 영역에서 수행되는 작업은 프록시와 연결된 로그인 계정을 통해서 수행되며, 운영 체제 영역에서 수행되는 작업을 수행하기 위해서는 프록시와 연결된 자격증명을 통해서 자격증명과 연결된 운영 체제의 로그온 계정의 권한을 가지고 작업을 수행하게 됩니다.



[주의]
Windows 로그인 이라고 할지라도 SQL Server에 로그인 한 상태에서 SQL Server 영역을 넘어 운영 체제와 상호 작용을 하기 위해서는 반드시 자격 증명과 프록시를 지정해야 합니다.

프록시는 작업의 성격에 따라서 ActiveX 스크립트, 운영 체제(CmdExec), 복제 배포자, 복제 병합, 복제 큐 판독기, 복제 스냅숏, 복제 트랜잭션 로그 판독기, SQL Server Analysis Services 명령, SQL Server Analysis Services 쿼리, SQL Server Integration Services 패키지 하위 시스템에 할당할 수 있습니다.



SQL Server 에이전트 작업에서 최소 권한으로 운영 체제와의 상호 작업을 위해서 프록시를 구성하는 경우에는 다음 단계를 따릅니다.

1 단계 : 운영 체제와 상호 작업에 필요한 Windows 로그온 계정을 생성합니다. 또한 SQL Server와의 상호 작업을 위해서 로그인 계정을 생성합니다.
2 단계 : 해당 Windows 로그온 계정에게 필요한 운영 체제 권한을 부여합니다. 로그인 계정에게는 SQL Server에서 필요한 권한을 부여합니다.
3 단계 : 해당 Windows 로그온 계정과 연결된 자격 증명(Credential)을 생성합니다.
4 단계 : 해당 자격 증명과 연결된 프록시(Proxy)를 생성합니다.
5 단계 : SQL Server 에이전트 작업에서 각 단계의 실행 계정을 프록시로 지정합니다.

■ 복제를 위한 프록시 구성

복제에서는 Windows 로그온 계정을 생성한 뒤 이와 연결된 Windows 로그인을 생성한 뒤 적절한 권한을 부여한 뒤 **[배포 에이전트]**의 계정을 해당 Windows 로그온 계정으로 변경하면 이와 연결된 각 에이전트 별로 자격 증명과 프록시를 자동으로 생성합니다. 복제를 위한 프록시 구성은 일반적인 SQL Server 에이전트 작업을 위한 프록시 구성의 3 단계의 자격 증명 생성과 4 단계 프록시 생성과정이 생략됩니다.

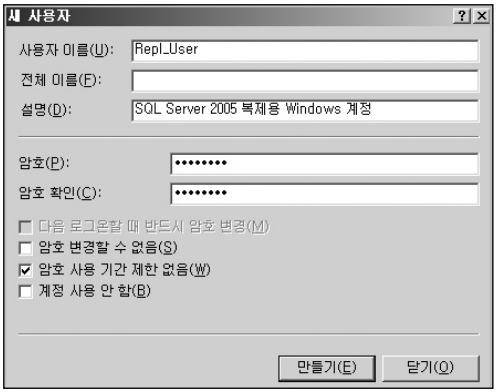
1 단계 : 운영 체제와 상호 작업에 필요한 Windows 로그온 계정을 생성합니다. 또한 SQL Server와의 상호 작업을 위해서 로그인 계정을 생성합니다.

2 단계 : 해당 Windows 로그온 계정에게 필요한 운영 체제 권한을 부여합니다. 로그인 계정에게는 SQL Server에서 필요한 권한을 부여합니다.

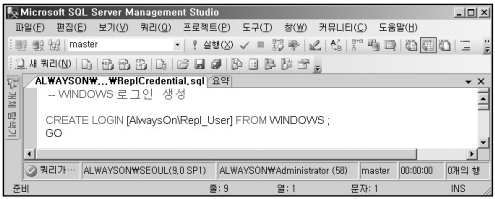
3 단계 : 각 복제 에이전트 보안 설정에서 1 단계에서 생성한 Windows 로그온 계정을 지정합니다.

1 단계 : 운영 체제와 상호 작업에 필요한 Windows 로그온 계정 및 로그인 계정 생성

워크 그룹 서버인 경우에는 관리도구의 **[컴퓨터 관리]** 도구에서 Active Directory 멤버 서버인 경우에는 **[Active Directory 사용자 및 컴퓨터]** 도구를 통해서 복제 전용 Windows 로그온 계정을 생성합니다.



이어서 해당 Windows 로그온 계정에 대해서 **[ALWAYSON₩SEOUL]** 노드를 포함한 각 피어 노드(**[ALWAYSON₩SEATTLE]**, **[ALWAYSON₩CAIRO]**)에서 SQL Server 로그인 계정을 생성합니다.



2 단계 : Windows 로그온 계정 및 로그인 계정에게 필요한 권한 부여

Windows 로그온 계정과 SQL Server 로그인 계정에게 다음과 같은 권한을 부여 합니다.

에이전트	사용 권한
스냅숏 에이전트	• 에이전트 계정 : 배포 데이터베이스에 db_owner 고정 역할에 멤버 스냅숏 공유에 대한 쓰기 권한 및 해당 폴더 NTFS 쓰기 권한
	• 게시자 연결에 사용되는 계정: 게시 데이터베이스의 db_owner 고정 역할에 멤버
로그 판독기 에이전트	• 에이전트 계정 : 배포 데이터베이스에 db_owner 고정 역할에 멤버 • 게시자 연결에 사용되는 계정: 게시 데이터베이스의 db_owner 고정 역할에 멤버
밀어넣기 구독에 대한 배포 에이전트	• 에이전트 계정 : 배포 데이터베이스에 db_owner 고정 역할에 멤버 게시 액세스 목록(PAL)의 멤버 스냅숏 공유에 대한 읽기 권한 및 해당 폴더 NTFS 읽기 권한 • 구독자 연결에 사용되는 계정: 구독 데이터베이스의 db_owner 고정 역할에 멤버

에이전트	사용 권한
끌어오기 구독에 대한 배포 에이전트	• 에이전트 계정 : 구독 데이터베이스에 db_owner 고정 역할에 멤버
	• 배포자 연결에 사용되는 계정: 게시 액세스 목록(PAL)의 멤버 스냅숏 공유에 대한 읽기 권한 및 해당 폴더 NTFS 읽기 권한
밀어넣기 구독에 대한 병합 에이전트	• 에이전트 계정 : 배포 데이터베이스에 db_owner 고정 역할에 멤버 게시 액세스 목록(PAL)의 멤버 스냅숏 공유에 대한 읽기 권한 및 해당 폴더 NTFS 읽기 권한 게시 데이터베이스에 사용자와 연결된 로그인이어야 합니다. • 구독자 연결에 사용되는 계정: 게시 데이터베이스의 db_owner 고정 역할에 멤버
끌어오기 구독에 대한 병합 에이전트	• 에이전트 계정 : 구독 데이터베이스에 db_owner 고정 역할에 멤버 • 게시자 및 배포자 연결에 사용되는 계정: 게시 액세스 목록(PAL)의 멤버 스냅숏 공유에 대한 읽기 권한 및 해당 폴더 NTFS 읽기 권한 게시 데이터베이스에 사용자와 연결된 로그인이어야 합니다. 배포 데이터베이스에 사용자와 연결된 로그인이어야 합니다.
큐 판독기 에이전트	• 에이전트 계정 : 배포 데이터베이스에 db_owner 고정 역할에 멤버 • 게시자 연결에 사용되는 계정: 게시 데이터베이스의 db_owner 고정 역할에 멤버 • 구독자 연결에 사용되는 계정 구독 데이터베이스의 db_owner 고정 역할에 멤버

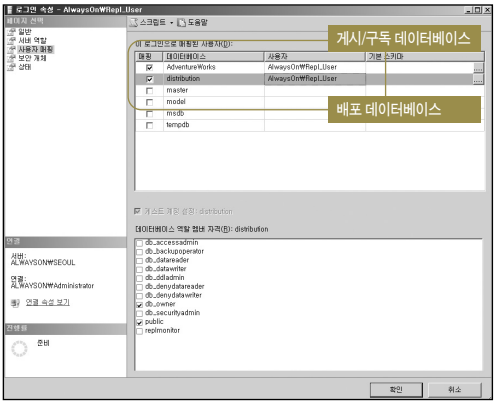
〈 복제용 프록시 또는 Windows 로그온 계정 및 SQL 로그인 필요 권한 리스트 〉

■ 피어 투 피어 복제 작업 수행을 위한 보안 구현

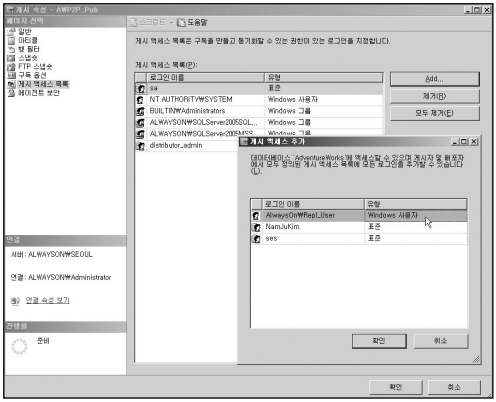
피어 투 피어 복제에서 에이전트에 대한 강화된 보안을 구성하기 위해서는 다음 작업을 수행합니다.

- **[로그 판독기 에이전트]**를 위한 권한을 설정합니다. 각 피어 노드에서 앞에서 생성한 로그인 계정과 연결된 데이터베이스 사용자 계정을 배포 데이터베이스와 게시데이터베이스(구독 데이터베이스와 동일)에 db_owner 데이터베이스 고정 역할에 멤버로 포함시킵니다
- 밀어 넣기 구독에 대한 **[배포 에이전트]**를 위한 권한을 설정합니다. 게시 액세스 목록(PAL)의 멤버로 추가합니다.
- 각 에이전트에 계정을 변경합니다.

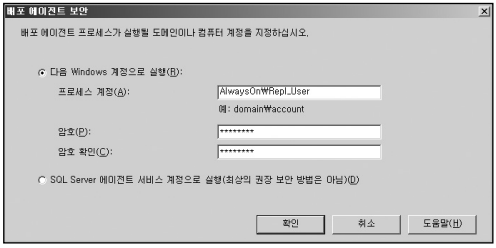
① **[로그 판독기 에이전트]**를 위한 권한을 설정합니다. 각 피어 노드에서 **[ALWAYSON₩Repl_User]** 로그인 계정에 대해서 **[distribution]** 배포 데이터베이스와 AdventureWorks 게시 및 구독 데이터베이스에 사용자 계정을 생성하고 각각 db_owner 데이터베이스 고정 역할에 포함시킵니다.



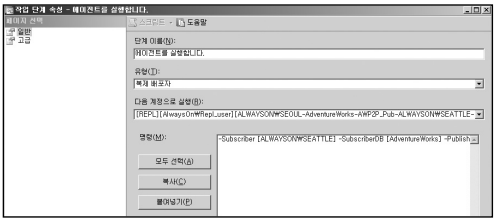
② **[배포 에이전트]**에 관한 보안을 설정합니다. 각 피어 노드에서 **[ALWAYSON\Repl_User]** 로그인 계정에 대해서 게시 **[AWP2P_Pub]**의 **[게시 액세스 목록(PAL)]**에 추가합니다. 각 피어 노드에서 **[개체 탐색기]**의 **[복제]**를 확장하고 **[AWP2P_Pub]** 게시를 오른쪽 버튼으로 눌러서 **[속성]**을 선택합니다. 이어서 **[게시 액세스 목록]** 페이지에서 **[Add…]** 버튼을 누른 뒤 **[ALWAYSON₩Repl_User]** 데이터베이스 계정을 추가합니다.



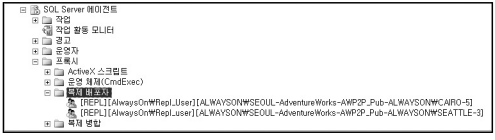
③ 복제 모니터에서 각 피어 노드의 **[배포 에이전트]**의 계정을 **[ALWAYSON₩Repl_User]**로 변경합니다.



④ 각 SQL Server 에이전트 작업에서 각 복제 작업의 실행 계정이 [ALWAYSON₩Repl_User]에 대해서 자동으로 생성된 프록시로 변경되었는지 확인합니다.



이어서 [개체 탐색기]에서 자동으로 자격 증명과 프록시가 생성되었는지 확인합니다.



10) 피어 투 피어 복제 백업 및 복원

복제된 데이터베이스 많은 주의를 기울여서 관리해야 합니다. 특히, 백업과 복원을 위해서는 특별한 주의를 필요로 합니다. 또한 게시 유형에 따라 백업 및 복원 전략이 달라 질 수 있습니다. 일반 트랜잭션 복제와 피어 투 피어 복제의 백업 및 복구 전략도 달라집니다.

다음은 일반 트랜잭션 복제의 백업 및 복구 전략을 위한 고려 사항입니다.

① 다음 데이터베이스를 정기적으로 백업합니다.

- 게시자의 게시 데이터베이스
- 배포자의 배포 데이터베이스
- 각 구독자의 구독 데이터베이스
- 게시자, 배포자 및 모든 구독자의 master 및 msdb 시스템 데이터베이스

② 단순 복구 모델의 시스템 데이터베이스는 다음 작업을 수행하게 되면 해당 데이터베이스 백업을 수행합니다.

- master 데이터베이스
  복제 활성화 또는 비활성화
  배포자에서 배포 데이터베이스 추가 또는 삭제
  게시자에서 데이터베이스 게시 활성화 또는 비활성화
  배포 게시자에서 게시자 활성화 및 비활성화
  게시자의 데이터베이스에 첫 번째 게시 추가 또는 마지막 게시 삭제
  구독자의 데이터베이스에 첫 번째 구독 추가 또는 마지막 구독 삭제

- msdb 데이터베이스
  복제 활성화 또는 비활성화
  배포자에서 배포 데이터베이스 추가 또는 삭제
  게시자에서 데이터베이스에 게시 활성화 또는 비활성화

배포자에서 복제 에이전트 프로필 만들기 또는 수정
배포자에서 복제 에이전트 프로필 매개 변수 수정
배포자에서 밀어넣기 구독의 일정을 비롯한 복제 에이전트 속성 변경
구독자에서 밀어넣기 구독의 일정을 비롯한 복제 에이전트 속성 변경

- 배포 데이터베이스는 기본적으로 단순 복구 모델로 설정됩니다. 배포자가 구성된 뒤 전체 모델로 변경하면 데이터베이스 백업을 자주 수행하지 않고 트랜잭션 로그 백업으로 대체할 수 있습니다.

③ 트랜잭션 복제에 대한 백업 설정

구독을 다시 초기화하거나 복제를 다시 구성하지 않고 게시 데이터베이스 또는 배포 데이터베이스가 손상된 경우에 트랜잭션 복제를 복구하려면, 게시 데이터베이스 및 배포 데이터베이스를 일치하는 시점으로 복원해야 합니다. 이를 위해서 트랜잭션 복제에는 배포 데이터베이스 및 게시 데이터베이스에 설정할 수 있는 sync with backup 옵션을 제공합니다.

- 게시 데이터베이스에서 sync with backup 옵션을 설정하면 게시자에서 백업되지 않은 트랜잭션을 배포 데이터베이스에 전파하지 않습니다. 따라서, 비상로그 백업에 실패하든지 성공하든지에 상관없이 게시자와 모든 구독자는 동기화된 데이터베이스를 유지할 수 있습니다.

- 배포 데이터베이스에 sync with backup 옵션을 설정하면, 배포 데이터베이스에서 트랜잭션 로그 백업을 수행하지 않은 게시 데이터베이스 트랜잭션 로그는 잘리지 않습니다. 즉, 배포 데이터베이스가 손상을 입고 비상 로그 백업이 실패하는 경우라도 마지막 트랜잭션 로그 백업으로 배포 데이터베이스를 복원하면 배달 되지 않은 게시자의 트랜잭션이 배포자에 배달되어 게시자와 구독자가는 동기화를 유지할 수 있습니다.

[주의]
배포 데이터베이스에 sync with backup을 설정하기 위해서는 복구 모델을 기본 설정인 단순 복구 모델에서 전체 모델로 변경합니다.

④ 복제 데이터베이스 백업관련 주의 사항

데이터베이스에 하나 이상의 트랜잭션 게시가 포함되어 있는 경우 게시와 관련된 트랜잭션이 모두 배포 데이터베이스로 배달될 때까지 로그가 잘리지 않습니다. 따라서, 데이터베이스 관리자는 항상 배포 데이터베이스의 상태를 점검해야 합니다. 배포 데이터베이스의 트랜잭션 로그가 가득 차거나 손상을 입는 경우에는 게시자 데이터베이스도 트랜잭션 로그가 잘리지 않게 됩니다.

어떤 백업 및 복구 전략을 사용하는지에 관계없이 항상 현재 복제 설정 스크립트를 안전한 위치에 보관해야 합니다. 서버에 오류가 발생하거나 테스트 환경을 설정해야 할 때는 서버 이름 참조를 변경하여 스크립트를 수정하고 이를 사용하여 복제 설정을 다시 만들 수 있습니다. 복제 스크립팅은 뒤에서 자세히 설명합니다.

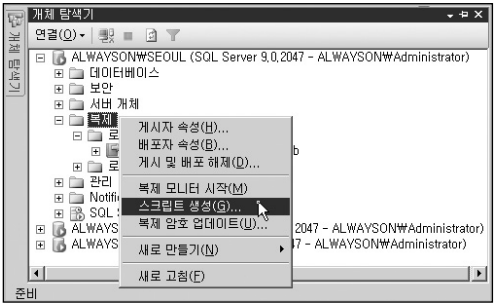
11) 피어 투 피어 복제 구성 스크립팅

복제 작업을 스크립트로 저장하게 되면 복제 작업을 재구성할 때 효율적으로 작업이 가능합니다. 따라서 복제를 구성하고 정상적으로 동작하는 것을 확인하면 즉시 복제 구성을 스크립트로 저장해야 합니다. 복제 스크립트 생성 작업은 배포자 및 게시자를 구성하는 배포 구성 단계 및 게시 구성 단계, 구독 구성 단계를 각각 스크립트로 별도로 저장할 수 있으며 작업 완료 뒤에 전체 작업을 하나의 스크립트로도 작성할 수 있습니다. 또한 현재 복제 설정을 스크립팅할 뿐만 아니라 복제의 활성화 및 비활성화도 스크립팅해야 합니다.

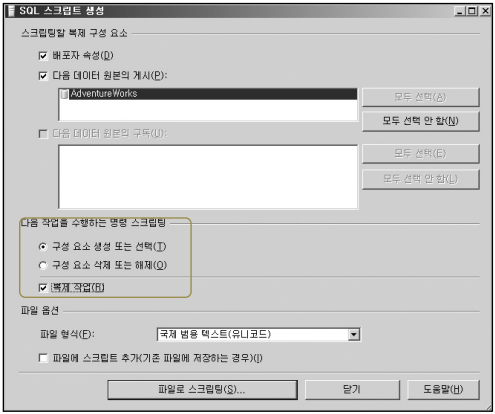
복제 전체 스크립트는 다음과 같이 구성되어 있습니다.

- 배포자 구성 스크립트
- 에이전트 프로필 생성 스크립트
- 배포 데이터베이스 추가 스크립트
- 게시자 추가 스크립트
- 복제 데이터베이스 설정 스크립트
- 트랜잭션 게시 추가 스크립트

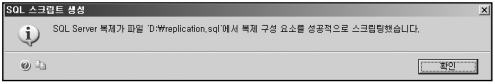
- 게시 액세스 목록 추가 스크립트
- 트랜잭션 아티클 추가 스크립트
- 트랜잭션 구독 추가 스크립트



[개체 탐색기]에서 서버 인스턴스와 복제를 차례로 확장하고 오른쪽 버튼을 눌러서 [스크립트 생성(G)]을 누릅니다.



[SQL 스크립트 생성] 창에서 [배포자 속정(D)], [다음 데이터 원본의 게시(P)], [다음 작업을 수행하는 명령 스크립팅], [복제 작업(R)]을 선택하고 [파일로 스크립팅(S)] 버튼을 누릅니다.



이렇게 복제를 설정하는 스크립트 뿐만 아니라 구성 요소를 삭제하거나 해제하는 스크립트도 생성해 놓아야 합니다. [다음 작업을 수행하는 명령 스크립팅] 부분에서 먼저 [구성 요소 생성 또는 선택(T)]옵션을 지정해서 스크립트를 생성한 뒤 다시 한번 [구성 요소 삭제 또는 해제(O)]를 선택해서 복제를 해제하는 스크립트도 작성합니다. 다음 스크립트는 은 복제 생성 스크립트의 각 요소를 설명하고 있습니다.

- 피어 투 피어 복제 생성 스크립트

```
/****** 서버 ALWAYSON\SEOUL의 복제 구성을 스크립팅하고 있습니다. 스크립트 날짜: ******/
/****** 참고: 보안을 위해 모든 암호 매개 변수는 NULL 또는 빈 문자열로 스크립팅되었습니다. ******/

/****** 시작: 게시자 ALWAYSON\SEOUL에서 실행할 스크립트 ******/

/****** 서버 ALWAYSON\SEOUL을(를) 배포자로 설치하고 있습니다. 스크립트 날짜: ******/
use master
exec sp_adddistributor @distributor = N'ALWAYSON₩SEOUL', @password = N'',
@from_scripting = 1
GO
```

```
-- 에이전트 프로필을 추가하는 중
-- 에이전트 프로필 기본값을 업데이트하는 중
exec sp_MSupdate_agenttype_default @profile_id = 1
GO
exec sp_MSupdate_agenttype_default @profile_id = 2
GO
exec sp_MSupdate_agenttype_default @profile_id = 4
GO
exec sp_MSupdate_agenttype_default @profile_id = 6
GO
exec sp_MSupdate_agenttype_default @profile_id = 11
GO

-- 배포 데이터베이스를 추가하는 중
use master
exec sp_adddistributiondb @database = N'distribution', @data_folder = N'D:
₩Program Files₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩Data', @data_file =
N'distribution.MDF', @data_file_size = 5, @log_folder = N'D:₩Program Files
₩ Microsoft SQL Server₩ MSSQL.1₩ MSSQL₩ Data', @log_file =
N'distribution.LDF', @log_file_size = 2, @min_distretention = 0, @max_distretention
= 72, @history_retention = 48, @security_mode = 1, @from_scripting = 1
GO

------ 개체: 작업    이름: 복제 에이전트 점검    스크립트 날짜: 2007-01-14 오전
11:20:42 ------
------ 개체: 작업    이름: distribution에 대한 복제 모니터링 리프레셔입니다.    스크
립트 날짜: 2007-01-14 오전 11:20:42 ------
```

--- 중간 생략 ---

-- 배포 게시자를 추가하는 중

exec sp_adddistpublisher @publisher = N'ALWAYSON₩SEOUL', @distribution_db = N'distribution', @security_mode = 1, @working_directory = N'D:₩Program Files₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩ReplData', @trusted = N'false', @thirdparty_flag = 0, @publisher_type = N'MSSQLSERVER'
GO
------ 개체: 작업 이름: ALWAYSON₩SEOUL-AdventureWorks-AWP2P_Pub-1 스크립트 날짜: 2007-01-14 오전 11:20:43 ------
------ 개체: 작업 이름: ALWAYSON₩SEOUL-AdventureWorks-1 스크립트 날짜: 2007-01-14 오전 11:20:43 ------
------ 개체: 작업 이름: ALWAYSON₩SEOUL-AdventureWorks-AWP2P_Pub-ALWAYSON₩SEATTLE-3 스크립트 날짜: 2007-01-14 오전 11:20:43 ------

--- 중간 생략 ---

-- 복제 데이터베이스를 설정하는 중

use master
exec sp_replicationdboption @dbname = N'AdventureWorks', @optname = N'publish', @value = N'true'
GO
exec [AdventureWorks].sys.sp_addlogreader_agent @publisher_security_mode = 1, @job_name = N'ALWAYSON\SEOUL-AdventureWorks-1', @job_login = null, @job_password = null

```
GO
exec [AdventureWorks].sys.sp_addqreader_agent @job_name = null,
@frompublisher = 1, @job_login = null, @job_password = null
GO

-- 트랜잭션 게시를 추가하는 중
use [AdventureWorks]
exec sp_addpublication @publication = N'AWP2P_Pub', @description = N'게시자
''ALWAYSON₩SEOUL''의 데이터베이스 ''AdventureWorks''에 대한 트랜잭션 게시
입니다.', @sync_method = N'native', @retention = 0, @allow_push = N'true',
@allow_pull = N'true', @allow_anonymous = N'false', @enabled_for_internet =
N'false', @snapshot_in_defaultfolder = N'true', @compress_snapshot = N'false',
@ftp_port = 21, @ftp_login = N'anonymous', @allow_subscription_copy = N'false',
@add_to_active_directory = N'false', @repl_freq = N'continuous', @status =
N'active', @independent_agent = N'true', @immediate_sync = N'true',
@allow_sync_tran = N'false', @autogen_sync_procs = N'false',
@allow_queued_tran = N'false', @allow_dts = N'false', @replicate_ddl = 1,
@allow_initialize_from_backup = N'true', @enabled_for_p2p = N'true',
@enabled_for_het_sub = N'false'
GO
exec sp_addpublication_snapshot @publication = N'AWP2P_Pub',
@snapshot_job_name = N'ALWAYSON₩SEOUL-AdventureWorks-AWP2P_Pub-
1', @job_login = null, @job_password = null, @publisher_security_mode = 1
GO
```

```
-- 게시 액세스 목록 추가

exec sp_grant_publication_access @publication = N'AWP2P_Pub', @login = N'sa'
GO
exec sp_grant_publication_access @publication = N'AWP2P_Pub', @login = N'NT
AUTHORITY₩SYSTEM'
GO
exec sp_grant_publication_access @publication = N'AWP2P_Pub', @login =
N'BUILTIN₩Administrators'
GO
exec sp_grant_publication_access @publication = N'AWP2P_Pub', @login =
N'ALWAYSON₩SQLServer2005SQLAgentUser$ALWAYSON$SEOUL'
GO
exec sp_grant_publication_access @publication = N'AWP2P_Pub', @login =
N'ALWAYSON₩SQLServer2005MSSQLUser$ALWAYSON$SEOUL'
GO
exec sp_grant_publication_access @publication = N'AWP2P_Pub', @login =
N'distributor_admin'
GO

-- 트랜잭션 아티클을 추가하는 중

use [AdventureWorks]
exec sp_addarticle @publication = N'AWP2P_Pub', @article = N'Address',
@source_owner = N'Person', @source_object = N'Address', @type =
N'logbased', @description = N'', @creation_script = N'', @pre_creation_cmd =
N'drop', @schema_option = 0x000000000803509F,
```

```
@identityrangemanagementoption = N'manual', @destination_table = N'Address',
@destination_owner = N'Person', @status = 24, @vertical_partition = N'false',
@ins_cmd = N'CALL [sp_MSins_PersonAddress]', @del_cmd = N'CALL
[sp_MSdel_PersonAddress]', @upd_cmd = N'SCALL
[sp_MSupd_PersonAddress]'
GO

--- 중간 생략 ---

-- 트랜잭션 구독을 추가하는 중

use [AdventureWorks]
exec sp_addsubscription @publication = N'AWP2P_Pub', @subscriber =
N'ALWAYSON₩CAIRO', @destination_db = N'AdventureWorks', @subscription_
type = N'Push', @sync_type = N'replication support only', @article = N'all',
@update_mode = N'read only', @subscriber_type = 0
exec sp_addpushsubscription_agent @publication = N'AWP2P_Pub', @subscriber
= N'ALWAYSON₩CAIRO', @subscriber_db = N'AdventureWorks', @job_login =
null, @job_password = null, @subscriber_security_mode = 1, @job_name =
N'ALWAYSON₩SEOUL-AdventureWorks-AWP2P_Pub-ALWAYSON₩CAIRO-4',
@dts_package_location = N'Distributor'
GO
use [AdventureWorks]
exec sp_addsubscription @publication = N'AWP2P_Pub', @subscriber =
N'ALWAYSON\SEATTLE', @destination_db = N'AdventureWorks',
@subscription_type = N'Push', @sync_type = N'replication support only', @article
= N'all', @update_mode = N'read only', @subscriber_type = 0
```

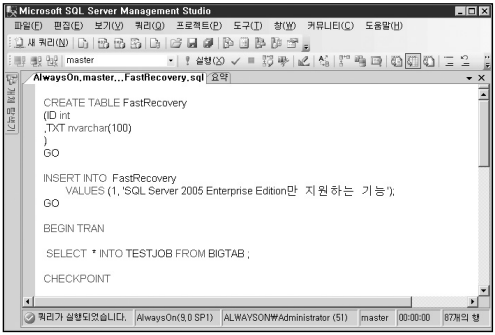
```
exec sp_addpushsubscription_agent @publication = N'AWP2P_Pub', @subscriber
= N'ALWAYSON₩SEATTLE', @subscriber_db = N'AdventureWorks', @job_login
= null, @job_password = null, @subscriber_security_mode = 1, @job_name =
N'ALWAYSON₩SEOUL-AdventureWorks-AWP2P_Pub-ALWAYSON₩SEATTLE-
3', @dts_package_location = N'Distributor'
GO
```

4. 고가용성을 위한 SQL Server 2005 추가 기능

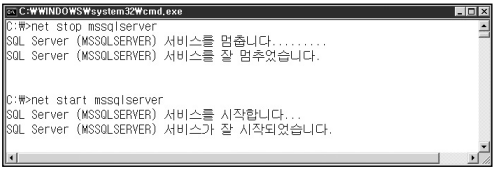
1) 빠른 복구 (Fast Recovery)

SQL Server 2005 Enterprise Edition은 트랜잭션 수행 중에 장애가 발생해서 복구 프로세스를 진행하거나 데이터베이스 미러링에서 장애 조치를 수행하기 위해서 실행 취소 단계를 수행하는 동안 데이터베이스를 빠르게 복구할 수 있어서 다운타임을 최소로 줄일 수 있습니다. 이것은 빠른 복구 (Fast Recovery)라는 기능을 수행하기 때문이며 빠른 복구는 복구 프로세스를 진행하는 과정에서 롤 포워드를 작업이 종료되면 즉시 데이터베이스를 사용할 수 있도록 개방합니다. 롤백 작업은 이 이후에 진행하게 됩니다. SQL Server 2005의 다른 Edition이나 이전 버전에서는 롤 포워드 작업과 롤백 작업이 모두 완료되어야 사용자가 데이터베이스에 액세스할 수 있습니다.

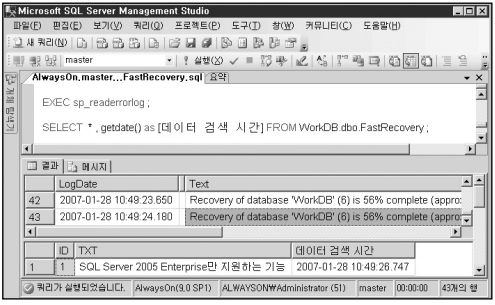
먼저 다음과 같이 커밋되지 않은 트랜잭션이 있는 상태에서 SQL Server 서비스를 다시 시작하여 데이터베이스 복구 프로세스가 다소 시간이 걸리도록 유도합니다.



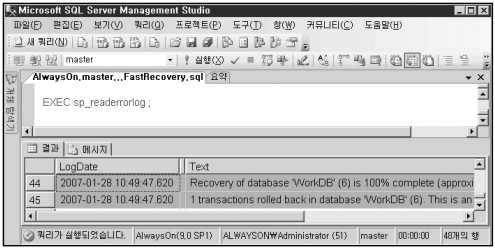
SQL Server 서비스가 정상 종료되는 경우에 검사점(CHECKPOINT) 프로세스가 자동으로 수행되지만 이해를 돕기 위해 명시적으로 수행하였습니다. 즉 [SELECT * INTO TESTJOB FROM BIGTAB]이라는 구문으로 생성된 TESTJOB 테이블과 해당 테이블에 입력된 행은 트랜잭션이 커밋되지 않았으나 검사점 프로세스에 의해서 데이터 파일에 기록되었습니다.



SQL Server 서비스를 다시 시작하여 데이터베이스 복구 프로세스 진행 상태와 롤백 프로세스를 진행하는 테이블이 아닌 다른 테이블이 사용 가능한지 확인합니다.



SQL Server 오류 로그에서 [WorkDB] 데이터베이스의 의 복구 프로세스가 약 56% 진행된 상태에서 [FastRecovery] 테이블을 검색하였습니다. [WorkDB] 데이터베이스의 복구 완료 시간을 확인합니다.



[WorkDB] 데이터베이스에서 하나의 트랜잭션을 롤백하는 복구 프로세스는 약 20여 초가 수행되었으나 사용자는 롤백 프로세스가 진행되는 테이블을 제외하고 롤 포워드가 진행된 모든 테이블에 대해서 작업이 가능합니다. SQL Server 2005 Enterprise Edition이 아니라면 이 상황에서 약 20여 초간 [WorkDB] 사용자 데이터베이스 전체를 사용할 수 없게 됩니다.

2) 즉시 파일 초기화

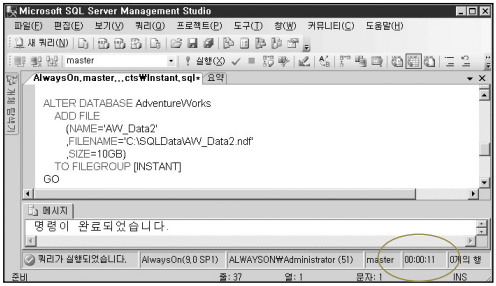
SQL Server는 데이터 파일을 생성하거나 크기를 증가시킬 때 그리고, 복원 작업을 수행할 때 파일을 초기화합니다. 이 초기화 작업은 생성하는 파일이나 증가된 파일 부분의 비트 값을 전부 0으로 채우는 작업입니다. 어찌 보면 RAID 5에서 0으로 초기화하는 것과 유사합니다. RAID는 성능 향상을 위해서 0으로 초기화 하지만 SQL Server는 보안 목적으로 0으로 초기화 합니다. 즉 디스크에 있는 기존 데이터를 지우는 작업을 하는 것입니다. 따라서 SQL Server가 사용하는 파일을 0으로 채우지 않는 즉시 파일 초기화 기능을 사용하면 파일의 생성이나 크기의 증가, 데이터베이스 복구 작업의 성능을 향상할 수 있습니다. 성능의 저하를 가져오면서 굳이 이전에 디스크에 기록되었던 데이터를 지울 필요가 없다면 디스크의 보안은 NTFS 보안에 의해서 관리하고 즉시 파일 초기화를 사용하는 것이 좋다고 판단됩니다.

SQL Server 2005에서는 다음 조건에 만족하면 파일 초기화 시에 0으로 채우지 않고 빠르게 파일을 즉시 초기화합니다.

- SQL Server 2005
- Windows Server 2003 또는 Windows XP
- SQL Server 서비스 계정에게 **[볼륨 관리 작업을 수행]** 권한 부여

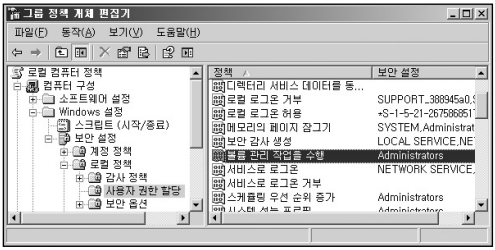
다음에서 즉시 파일 초기화 기능을 사용하여 데이터 파일을 추가하거나 데이터베이스를 복원 작업을 하는 경우와 즉시 파일 초기화 기능을 사용하지 않았을 때의 성능을 비교합니다.

① 즉시 파일 초기화 기능을 사용해서 10GB 크기의 데이터 파일을 추가하는 경우

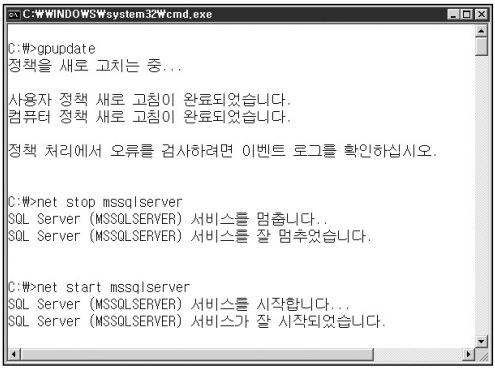


즉시 파일 초기화를 위한 조건이 만족한 경우에 10GB 크기의 데이터 파일을 추가하는 데 약 11초가 소요되었습니다. 이 시간은 디스크의 성능과 관련됩니다.

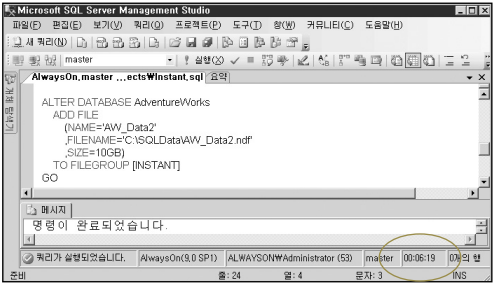
② 즉시 파일 초기화 기능을 사용하지 않고 10GB 크기의 데이터 파일을 추가하는 경우



즉시 파일 초기화를 사용하지 않기 위해서 그룹 정책 개체 편집기에서 **[볼륨 관리 작업을 수행]** 정책에서 SQL Server 서비스 계정이 포함된 도메인 그룹을 제거한 뒤 다음과 같이 변경된 그룹 정책을 즉시 적용하는 **[gpupdate]** 구문을 명령 프롬프트에서 수행한 뒤 SQL Server 서비스를 다시 시작합니다.

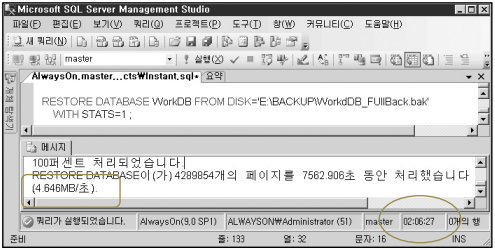


이어서 기존에 추가한 파일을 삭제하고 동일한 구문을 수행하여 10GB 크기의 데이터 파일을 추가하여 시간을 측정합니다.



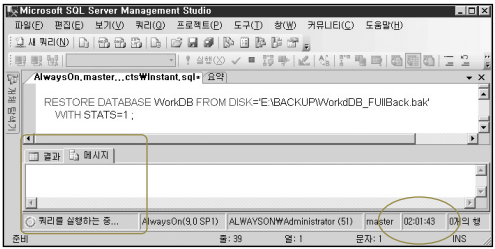
즉시 파일 초기화 기능을 사용하지 않은 경우 약 6분 19초가 소요되었습니다. 이와 같은 상황을 확인하였다면 당연히 즉시 파일 초기화 기능을 사용해야 한다고 판단하였을 것입니다.

③ 즉시 파일 초기화 기능을 사용하여 약 36.5GB의 데이터베이스를 복구하는 경우

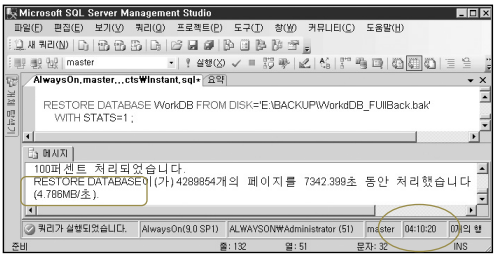


즉시 파일 초기화 기능을 사용해서 데이터베이스를 복구하는 경우에 약 2시간 6분 가량이 소요된 것을 확인할 수 있습니다.

④ 즉시 파일 초기화 기능을 사용하지 않고 약 36.5GB의 데이터베이스를 복구하는 경우



즉시 파일 초기화 기능을 사용하지 않은 경우는 복원 작업을 시작한지 약 2시간 이상이 진행되어도 채 1%도 진행되지 않고 있습니다. 아직까지도 파일 초기화 작업을 수행하고 있는 중이기 때문입니다.



데이터베이스가 복구되는 데까지 총 4시간 여가 소요되었습니다. 이 때 복원 처리 속도는 초당 4,786MB로 즉시 파일 초기화를 사용하여 복구한 상태와 거의 비슷합니다. 그런데 처리 시간은 2시간 가까이 차이가 발생하였습니다. 이 시간이 파일 초기화에 소요된 시간인 것입니다. 만약 이 상황이 실제 상황이라면 즉시 파일 초기화 기능을 사용하면 다운 타임을 2시간 정도 줄일 수 있게 된다는 것을 확인할 수 있습니다.

[주의]
파일 초기화에 소요되는 시간은 디스크의 쓰기 성능이 가장 큰 영향을 미칩니다. 데이터 복원과 복구에 걸리는 시간도 대상 디스크의 쓰기 성능과 백업 세트를 읽는 성능이 가장 큰 요소가 됩니다. 이에 따라서 테스트의 결과는 다양하게 발생할 수 있습니다.

3) 온라인 작업

SQL Server 2005 Enterprise Edition은 복원 작업과 인덱스 작업에 온라인 작업을 지원합니다. 온라인 복원 작업은 데이터베이스가 가용한 상태에서 손상을 입은 페이지나 데이터 파일, 파일 그룹을 복원하는 기능으로 복원 작업중의 가용성을 향상할 수 있으며 온라인 인덱스 작업은 해당 작업 시 ONLINE 옵션을 사용하며 인덱스 작업 동안 여러 사용자가 기본 테이블이나 클러스터형 인덱스 데이터 및 해당 테이블의 비클러스터형 인덱스에 동시에 액세스하거나 수정 또는 삭제 작업을 할 수 있어서 24시간 가동 장비와 같이 유지 관리 업무를 수행하는 가운데에서 가용성을 향상할 수 있습니다.

■ 온라인 복원

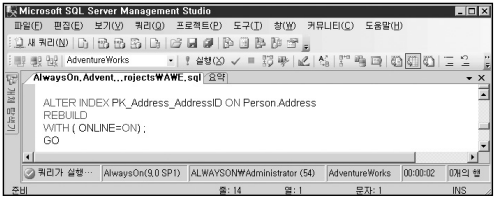
SQL Server 2005 Enterprise Edition은 증분 복원, 페이지 복원, 파일 및 파일 그룹 복원에 대해서 별도의 옵션 지정 없이 자동으로 온라인 복원 기능을 지원합니다

[참고]
*SQL Server 2005의 백업과 복원에 대해서는 온라인 설명서나 필자가 집필한 도서 출판 대림의 **[SQL Server 2005 재해 복구 및 고가용 솔루션]**을 참조하기 바랍니다.*

■ 온라인 인덱스

SQL Server 2005에서 ONLINE 옵션을 사용하지 않는 기본 설정 상태에서는 클러스터형 인덱스 작성 또는 다시 작성 등의 DDL 작업을 수행하면 기본 데이터와 관련 인덱스에 대해 배타적 잠금을 보유하게 되어 해당 인덱스 작업이 완료될 때까지 기본 데이터를 수정하거나 쿼리할 수 없으며 비 클러스터형 인덱스에 대한 DDL 작업은 기본 데이터에 대한 쿼리만 허용합니다.

따라서 24시간 가동 시스템인 경우에는 필수적인 기능이 바로 SQL Server 2005 Enterprise Edition에서 제공하는 온라인 인덱스 작업 기능입니다. 단 text, ntext, image, varchar(max), nvarchar(max), varbinary(max) 또는XML 데이터 형식이 있는 경우나 분할된 인덱스의 경우에는 온라인 인덱스 작업을 수행할 수 없습니다.



4) 관리자 전용 연결

조금 어려운 얘기일 수 있습니다. SQL Server는 Windows Server의 스레드 스케줄링을 담당하는 커널의 구성 요소인 디스패처(Dispatcher)와 통신하여 SQL Server의 작업자 스레드를 처리하는 자체 스레드 스케줄러를 가지고 있습니다. 이것을 사용자 모드 스케줄러라고 하는 데 얘는 선점형이 아니라 협동형 스케줄러입니다. 그래서 작업자 스레드가 뻗는 경우에 이 사용자 모드 스케줄러도 뻗게 됩니다. 이런 경우에 사용자의 연결 요청이나 작업을 처리하지 못하고 행이 걸리게 됩니다. 또한 할당된 스레드의 개수가 서버 구성 옵션 가운데 [Max Worker Thread]의 임계값에 도달하게 되면 더 이상 스레드를 할당하지 못하게 됩니다. 이전 버전에서는 이런 경우 부득이하게 SQL Server 서비스를 다시 시작해야만 했습니다. 결국 SQL Server의 다운 타임이 발생하게 되고 데이터베이스의 복구 프로세스도 수행해야 하므로 연쇄적인 문제를 유발할 수 있었습니다.

그렇지만 SQL Server 2005에서는 관리자 전용 연결(Dedicated Admin Connection)이라고 하는 별도의 스케줄러를 사용하는 연결을 통해서 관리자가 접속하여 문제를 유발하는 스레드를 종료하거나 상황을 분석 및 진단할 수 있습니다. sysadmin 고정 서버 역할의 멤버는 SQL Server 2005에서 새롭게 제공하는 sqlcmd 유틸리티나 SQL Server Management Studio를 통해서 관리자 전용 연결(DAC)을 통해서 SQL Server 2005에 연결할 수 있습니다.

sqlcmd 유틸리티를 통해서 DAC으로 연결하고자 하는 경우에는 [A] 옵션과 함께 연결합니다.

SSMS를 사용해서 DAC으로 연결하고자 하는 경우 [서버 이름(S)] 부분에 접속하고자 하는 서버 인스턴스 이름 앞에 [admin:]을 입력하고 연결합니다. 단, 관리자 전용 연결은 하나의 인스턴스에 하나의 관리자 전용 연결 세션만 허용합니다.

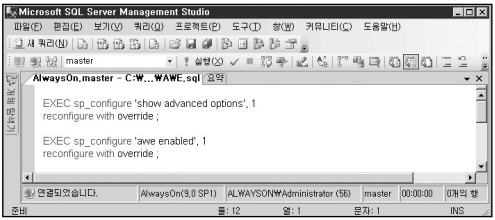
5) 동적 AWE 메모리 관리

SQL Server는 4GB 이상의 대용량 메모리를 사용하기 위해서는 AWE 메모리를 사용합니다. 32비트의SQL Server는 이 AWE 메모리를 사용할 때 메모리의 구성 요소가운데 버퍼 캐시만이 이 AWE 메모리 영역을 사용할 수 있습니다. 64비트 SQL Server는 메모리의 모든 구성 요소가 AWE 영역을 사용합니다. 이전 버전에서는 이 AWE 영역은 인스턴스가 시작될 때 고정된 메모리 양이 할당되어서 변화하는 작업에 맞게 메모리 사용을 조정할 수 없어서 불필요한 페이징이 발생할 수 있었습니다.

SQL Server 2005에서는 현재 작업을 기반으로 사용되는 AWE 메모리 양을 동적으로 조정하게 됩니다. 따라서 운영 체제가 메모리를 필요로 하는 경우 이미 할당된 AWE 메모리를 다시 운영체제에게 반납할 수 있으며 필요한 경우에는 다시 운영 체제로부터 메모리를 할당받을 수 있게 되었습니다.

■ AWE 메모리 활성화

AWE 메모리를 활성화 하기 위해서는 다음 그림과 같이 [awe enabled] 서버 구성 옵션을 활성화해야 합니다.



[awe enabled] 서버 구성 옵션은 고급 옵션이므로 [show advanced options] 서버 구성 옵션을 먼저 활성화해야 확인하고 변경할 수 있습니다. 또한 이와 같은 AWE 메모리를 사용하기 위해서는 32비트 및 64비트 버전의 SQL Server에서 모두 SQL Server 서비스 계정에게 **[메모리 페이지 잠그기]** 권한을 할당해야 합니다.

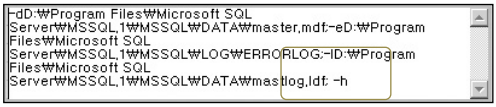
6) Hot Add 메모리

하드웨어를 업그레이드 하는 과정에서 어쩔 수 없는 서비스의 중단을 경험하곤 합니다. CPU의 추가나 메모리의 추가가 그렇습니다. 하드 디스크의 추가는 대부분의 어레이 컨트롤러가 디스크의 핫 플러깅 (Hot-Plugging) 기능을 지원하므로 시스템을 종료한 뒤 장착할 필요가 없습니다. 또한 Windows Server가 이러한 디스크의 핫 플러깅 기능을 지원하기 때문에 추가된 디스크를 디스크 관리자에서 초기화하여 즉시 사용할 수 있습니다.

SQL Server 2005 Enterprise Edition은 Hot Add 메모리라는 기술을 통해서 디스크의 핫 플러깅 기능과 같이 시스템의 종료하지 않고 메모리를 추가하여 즉시 사용할 수 있습니다. 단 32비트인 경우에는 [awe enabled] 서버 구성 옵션이 설정되어 있어야 사용할 수 있습니다.

디스크 핫 플러깅과 마찬가지로 이 기능은 하드웨어와 운영 체제가 기능을 지원해야 합니다. 따라서 운영 체제는 Windows Server 2003 Enterprise Edition 또는 Datacenter Edition이라야 하고 서버의 경우에는 해당 서버 공급 업체를 통해서 기능을 지원하는 지 반드시 확인해야 합니다. 이 기능을 지원하는 서버는 메모리 모듈이 외부에 노출되어 있습니다.

SQL Server 2005에서 Hot Add 메모리 기능을 사용하기 위해서는 SQL Server 서비스 시작 옵션 h를 지정한 뒤 서비스를 시작해야 합니다. SQL Server Configuration Manager를 실행하여 **[SQL Server 2005 서비스]**에서 해당 인스턴스의 **[등록 정보]** 창을 열고 시작 매개 변수 부분에 **-h**를 지정합니다. 이어서 반드시 SQL Server 서비스를 다시 시작해야 합니다.



7) 각 기능을 지원하는 Edition

고가용성을 위한 SQL Server 2005의 기능을 지원하는 Edition은 다음과 같습니다. 이 때 운영 체제의 요구 사항도 반드시 확인하여야 합니다

기 능	지원하는 Edition	운영 체제 요구 사항 등
장애 조치 클러스터링	SQL Server 2005 Enterprise Edition (8 노드) SQL Server 2005 Standard Edition (2 노드)	Windows Server 2000 Advanced Server 또는 Windows Server 2003 Enterprise Edition 이상
데이터베이스 미러링	SQL Server 2005 Enterprise Edition (모든 기능) SQL Server 2005 Standard Edition (동기 모드)	
로그 전달	SQL Server 2005 Workgroup Edition 이상	
피어 투 피어 복제	SQL Server 2005 Enterprise Edition	
데이터베이스 스냅숏	SQL Server 2005 Enterprise Edition	
빠른 복구	SQL Server 2005 Enterprise Edition	
파일 즉시 초기화	SQL Server 2005 모든 Edition	Windows Server 2003 / XP

기 능	지원하는 Edition	운영 체제 요구 사항 등
온라인 작업 (온라인 인덱스 온라인 복원)	SQL Server 2005 Enterprise Edition	
관리자 전용 연결	SQL Server 2005 모든 Edition	
동적 AWE 메모리 관리	SQL Server 2005 Standard Edition 이상	Windows Server 2003 모든 Edition
Hot Add 메모리	SQL Server 2005 Enterprise Edition	Windows Server 2003 Enterprise Edition 이상

비매품

SQL Server 2005 고가용성 구축 및 운영 가이드

- 저 자: 이 종 인
- 기 획: 임 무 호
- Contents 관련문의: jilee@daoudata.co.kr

- 발 행: 한국마이크로소프트(유)
- 제 작: (주)다우데이타시스템
- 초판 발행일: 2007년 4월

Microsoft
SQL Server 2005

SQL Server 2005 고가용성 구축 및 운영 가이드

Microsoft
한국마이크로소프트(유)

서울특별시 강남구 대치동 892번지 포스코센터 서관 5층
고객지원센터 : 1577-9700
인터넷 : http://www.microsoft.com/korea/sql

SQL-200707-01