Microsoft SQL Server 2005

Microsoft SQL Server 2005 관리자 가이드

Microsoft

저자 약력

김정선 (주)필라넷 수석 컨설턴트
MVP, MCDBA, 멀티캠퍼스 전임강사, 전 삼성 중공업
컨설팅: 삼성 반도체, 무림제지, 대우 중공업, 롯데 제과, Yes24, 대정 화금 외 다수

성대중 (주)필라넷 책임 컨설턴트
전, 영림원, SQL Server 매거진 전문 번역
컨설팅: 롯데 칠성, 삼성 반도체, LG 상사, Yes24, 두산그룹, 홍진 크라운 외 다수

현중균 (주)필라넷 주임 컨설턴트
전 한양여대 겸임교수, MCSE, MCDBA, SQL Server 매거진 전문 번역
컨설팅: CyberMBA, Yes24, Quest Tech Engineer

감수자 약력

정원혁 (주)필라넷 상무 이사 , DB사업부장, MCDBA, MCT
전문가로 가는 지름길, MS SQL Server 2000외 7권 저술, SQL Server 매거진 전문 번역
전, 마이크로소프트, 이랜드
컨설팅: 두산, 롯데 카드, 삼성 반도체, KT, CJ CGV, 대한 생명, 동양 증권, 동양 생명 , 신한 은행, 제일 은행 외

서문

첫 아이를 봤을 때, 그 아이가 세상에 나오기만을 고대하는 마음을 기억합니다. 물론 어찌 비유의 대상이 되겠습니까만, 2005년 11월 7일 샌프란시스코에서 SQL Server 2000이 나온 지 5년 만에 드디어 SQL Server 2005가 출시되었습니다.

베타 테스터를 포함한 전문가들은 이미 오래 전부터 SQL Server 2005의 모습을 보고 있었으며, 출시될 날 만을 기다리고 있습니다. 그래서인지 출시가 계속 지연될 때마다 아쉬움 마음이 가시질 않았습니다.

SQL Server 2005는 우리들의 기대를 져버리리 않을 만큼 훌륭한 기술들과 서비스로 변신을 했습니다. 관리자와 개발자 그리고 BI 분석가에 이르기까지 그 향상된 기술의 넓이와 폭은 상당하다고 말할 수 있습니다.

SQL Server 6.5에서 7.0으로 넘어갈 때의 그 놀라움을 넘어선 새로운 변화와 시도가 SQL Server 2005를 통해서 나타납니다.

SQL Server 2005 New Features 가이드 북은 이러한 기술들과 서비스들의 핵심을 요약하고 정리해서 작성되었습니다. SQL Server 2005를 알고자 하는 모든 사용자들에게 좋은 참고 자료가 될 수 있기를 기대하면서, 많은 IT 비즈니스 현장에서 SQL Server 2005를 통해 가치를 창조할 수 있었다는 행복한 이야기들을 들을 수 있게 되기를 기원합니다.

2006. 3. ㈜ 필라넷 DB 사업부

목차

SQL Server 2005 소개 ····· 6
- SQL Server 2005 도입효과 ····· 7
- SQL Server 2005 구성요소 ····· 8
- SQL Server 2005의 새로운 기능 ····· 11
- SQL Server 2005 에디션별 특징 ····· 12
- SQL Server 2005 관련자료 ····· 15

SQL Server Management Studio ····· 17
- SQL Server Management Studio ····· 17
- 등록된 서버 ····· 19
- 개체 탐색기 ····· 20
- 솔루션 탐색기 ····· 21
- SQL Server Management Studio로 쿼리 작성 ····· 23
- 비연결 편집 ····· 24
- 실행계획보기개선 ····· 24

SQL Server 2005 구성도구 ····· 27
- SQL Server 구성 관리자 ····· 27
- 서비스 관리 ····· 29
- 서버 네트워크 구성 ····· 31
- 클라이언트 네트워크 구성 ····· 34
- SQL Server 노출 영역 구성도구 ····· 35

SQL 서버 2005의 성능 도구 ····· 38
- 데이터베이스 엔진 튜닝 관리자(Database Engine Tuning Advisor) ····· 38
- SQL Server 프로필러 ····· 45

SQLCMD 유틸리티와 관리자전용연결(DAC)·····47
SQLCMD 유틸리티 ·····47

SQL Server 2005 데이터베이스 로그 크기 ·····58
인덱스 관리 개선기능 ·····60
SQL Server 2005의 새로운 인덱스 관리 기능 ·····60
인덱스 생성 ·····63
인덱스 수정 ·····67
인덱스 삭제 ·····72
인덱스 조각화 관리 ·····74

분할된 테이블과 인덱스(Partitioned Tables and Indexes) ·····77
대용량 테이블의 유지 관리 문제 ·····77
분할 함수 (Partition Function)·····79
분할 구성표 (Partition Scheme) ·····83
분할된 테이블 및 인덱스 (Partitioned Table and Indexes)·····84
분할된 인덱스(Partitioned Index) ·····87
카탈로그 뷰를 사용한 메타데이터 검색 ·····89
분할된 테이블에 대한 쿼리·····90
분할 유지관리 ·····91
슬라이딩 윈도우(Sliding Window) ·····95
백업과 복원 및 유지 관리 ·····99

복제(Replication) 향상 기능 ·····101
새로운 기능·····101
단순하고 편리해진 복제 구성 마법사·····103
복제 구성 따라하기 ·····104
향상된 복제 모니터 ·····133
피어 투 피어(Peer-to-Peer) 복제 ·····137
추적 프로그램 토큰 ·····138

DDL 구문 지원 ····· 139
Oracle도 이제 복제 게시자 역할 수행 ····· 140
기타 추가 향상 기능 ····· 141
웹을 통한 동기화 ····· 142
SQL Server 2005에서 사라지는 기능들 ····· 142
SQL Server 2005 성능 및 확장성 ····· 143

데이터베이스 스냅숏 ····· 145
백업의 새로운 점 ····· 151
복사전용(COPY ONLY)백업 ····· 151
부분백업 (Partial Backup) ····· 152

백업미디어에 대한 신뢰성 향상 ····· 154
체크섬(Checksum) 이용한 백업과 복원의 데이터 신뢰성 향상 ····· 154
미러백업 ····· 156
에러와 독립적으로 복원하기 ····· 158
다중인스턴스의 변화 ····· 159

복원의 새로운 점 ····· 160
온라인 복원 ····· 160
Piecemeal 복원 ····· 163
응급데이터베이스 상태 ····· 168
ATTACH_REBUILD_LOG 절 ····· 171

데이터베이스 미러링 ····· 173
데이터베이스 미러링이 가능하도록 서비스 시작하기 ····· 173
데이터베이스 미러링의 장점 ····· 174

SQL Server 2005 소개

Microsoft SQL Server 2005는 엔터프라이즈 환경에 사용할 수 있는 데이터 관리 및 분석 어플리케이션에 강력한 보안, 확장성 및 가용성을 제공하며, 어플리케이션을 더욱 쉽게 구축, 배치 및 관리할 수 있도록 돕는 차세대 데이터 관리 및 분석 솔루션입니다.

SQL Server 2005를 통해, 각 기업에서는 데이터를 기반으로 신속하게 의사결정을 내릴 수 있고, 개발 인력의 생산성과 유연성을 향상시키며, IT 부문의 총소유비용을 절감하고, 지속적으로 증가하는 비즈니스 요구사항을 충족시킬 수 있습니다.
SQL Server 2000의 강점을 기반으로 구축된 SQL Server 2005는 모든 규모의 기업들에게 다음과 같은 이점을 제공하는 통합 데이터 관리 및 분석 솔루션입니다.

- 보안, 확장성 및 안정성이 더욱 강화된 엔터프라이즈 어플리케이션의 구축, 배치 및 관리
- 데이터베이스 어플리케이션 구축, 배치 및 관리의 복잡성을 해소함으로써 IT 생산성 극대화
- 다수의 플랫폼, 어플리케이션 및 장치 간 데이터 공유 통해 내부 및 외부 시스템에 더욱 쉽게 연결
- 성능, 가용성, 확장성, 보안의 침해없이 총소유비용 통제

SQL Server 2005 도입효과

SQL Server 2005 데이터 플랫폼은 모든 규모의 조직에 다음과 같은 도입효과를 제공합니다.

도입효과	설명
데이터 자산 활용	SQL Server 2005는 업무용 및 분석 어플리케이션을 위한 안전하고 신뢰할 수 있는 데이터베이스를 제공합니다. 또한, 고객이 보유하고 있는 데이터의 가치를 최대한 활용할 수 있도록 하기 위해, 강력한 리포팅, 분석 및 데이터 마이닝 등과 같은 기능을 제공합니다.
생산성 향상	SQL Server 2005는 비즈니스 인텔리전스 기능과 Microsoft Office System 과 같은 친숙한 도구에 대한 통합을 통해, 조직 전반의 정보 근로자가 필요로 하는 최신 비즈니스 정보를 제공합니다. Microsoft의 목표는 조직 내 모든 정보 사용자에게 확장된 비즈니스 인텔리전스 기능을 제공하고, 조직내 가장 중요한 자산인 데이터를 기초로 보다 나은 비즈니스 의사결정을 내릴 수 있도록 하는 것에 있습니다.
IT 복잡성 해소	SQL Server 2005는 개발자를 위해, 유연한 개발 환경을 제공하고, 데이터베이스 관리자을 위해 자동화된 통합 관리 도구를 지원하기 때문에 업무용 및 분석 어플리케이션의 개발, 구축 및 관리 작업을 보다 용이하게 수행할 수 있습니다.
저렴한 총소유비용(TCO)	사용자 편이성 및 배포 용이성에 초점을 맞춘 통합적인 접근 방법을 통해, 기초투자, 구현 및 유지 보수 비용을 업계 최저 수준으로 줄일 수 있으며, 데이터베이스 투자에 대한 ROI를 신속하게 회수할 수 있습니다.

SQL Server 2005 구성요소

SQL Server 2005는 엔터프라이즈 데이터관리 및 BI(Business Intelligence) 어플리케이션을 위해 뛰어난 보안, 안정성 및 생산성을 갖춘 플랫폼을 제공합니다. SQL Server 2005는 정보 근로자 뿐만 아니라 IT 전문가들에게 강력하고 친숙한 도구를 제공하며, 모바일 장치에서 엔터프라이즈 데이터 시스템에 이르기까지, 다양한 플랫폼에서 엔터프라이즈 데이터관리 및 분석 어플리케이션을 구축, 배포, 관리, 운영하기 위한 업무적인 복잡성을 줄여 줄 수 있습니다. SQL Server 2005에서 제공하는 광범위하고 다양한 기능과, 기존 시스템과의 상호 운용성, 일상 업무의 자동화를 통해, 모든 규모의 기업에서 완벽한 데이터 솔루션으로 사용될 수 있습니다.

구성요소	설명
관계형 데이터베이스 엔진 (Relational Database Engine) 	SQL Server 관계형 데이터베이스 엔진은 SQL Server 2005의 핵심으로 관계형 또는 XML형식의 데이터를 저장하고,추출하고,수정하는데 있어 탁월한 성능과 확장가능하고 안전한 환경을 제공하는 강력한 관계형 데이터 베이스 엔진입니다.
분석 서비스 (SQL Server Analysis Services) 	온라인 분석 처리 어플리케이션(OLAP) 과 데이터 마이닝을 지원하는 강력한 비즈니스 인텔리전스 솔루션입니다.
통합서비스 (Integration Services, or SSIS) 	데이터를 가져오고 내보내는 솔루션으로서 데이터의 이동이 이루어질 때 변환과정을 수행합니다

알림서비스 (Notification Services) 	이벤트와 요청된 데이터에 근거하여 이메일,텍스트 메시지 기타 다른 방식으로 알림(notifications)을 발생시킬 수 있습니다
리포팅 서비스 (Reporting Services) 	데이터 원본으로부터 데이터를 추출하여 보고서를 만들고 브라우저로 볼 수 있게 하거나 파일로 내보내거나 이메일로 보낼 수 있습니다
서비스 브로커 (Service Broker) 	소프트웨어 서비스간의 메시지기반 통신에 관한 서비스 입니다
네이티브 http 서비스 (Native HTTP Service) 	Microsoft Windows Server™ 2003에 설치되어 있을시 SQL Server 2005는 HTTP (Hypertext Transfer Protocol)로 이루어진 요구에 응답할 수 있습니다. Native HTTP Service 는 SQL Server 2005가 IIS (Microsoft Internet Information Services)없이도 웹서비스 인터페이스를 만들수 있도록 하여 줍니다.
SQL Server 에이전트 (SQL Server Agent) 	데이터베이스 유지 및 작업,이벤트,경고 관리를 자동화 하도록 하는 예약 관리 업무 엔진입니다.

닷넷 CLR 기반 서비스 .NET Common Language Runtime 	CLR(Common Language Runtime)이 SQL Server 에 내재되어 있어서 데이터베이스 솔루션이 Microsoft Visual C#® .NET 또는 Microsoft Visual Basic® .NET 과 같은 언어에서 생성된 관리 코드 (managed code)를 이용할 수 있도록 합니다.
복제 (Replication) 	한쪽 데이터베이스에서 다른 데이베이스로 데이터 및 데이터베이스 객체들을 복사 이동시키고난 후 데이터 베이스간의 일관성이 동기화되도록 하여줍니다.
전체 텍스트 검색 (Full Text Search) 	SQL Server 데이터베이스에 있는 텍스트로 저장된 키워드 기반 쿼리에 대한 빠르고 유연한 인덱스 검사를 가능하게 하여 줍니다

SQL Server 2005의 새로운 기능

■ 데이터베이스 관리측면

주요특징	설명
데이터베이스 미러링	새로운 데이터베이스 미러링 솔루션을 이용해 로그 전달 기능을 확장할 수 있습니다. 데이터베이스 미러링을 사용하면 대기 서버에 자동 장애 조치를 설정하여 SQL Server 시스템의 가용성을 향상시킬 수 있습니다.
온라인 복원	데이터베이스 관리자들은 SQL Server 인스턴스를 실행하는 동시에 복원 작업을 수행할 수 있습니다. 온라인 복원 기능을 실행하는 경우, 복원 중인 데이터만 액세스할 수 없으며 온라인 상태를 유지하고 있는 나머지 데이터들은 액세스할 수 있기 때문에 SQL Server 의 가용성을 높일 수 있습니다.
온라인 인덱싱 작업	온라인 인덱스 옵션을 사용하여 인덱스 DDL(Data Definition Language) 실행 중에 기본 테이블이나 클러스터링된 인덱스 데이터 및 모든 관련 인덱스를 동시에 수정(업데이트, 삭제 및 삽입)할 수 있습니다. 예를 들어, 클러스터링된 인덱스가 다시 작성되는 동안 관리자들은 지속적으로 기본 데이터를 업데이트하고 이 데이터에 대해 쿼리를 수행할 수 있습니다.
고속 복구	새로운 고속 복구 옵션은 SQL Server 데이터베이스의 가용성을 향상시킵니다. 관리자들은 트랜잭션 로그가 롤 포워딩된 후에 복구 데이터베이스에 다시 연결할 수 있습니다.
향상된 보안 기능	데이터베이스 암호화, 안전한 기본값 설정, 암호 정책 적용, 세부적인 권한 제어, 강화된 보안 모델 등과 같은 보안 기능을 추가했습니다.

새로운 SQL Server Management Studio	새로운 통합 관리 도구 세트인 SQL Server Management Studio를 추가했습니다. 이를 통해 SQL Server 데이터베이스의 개발, 배포 및 문제 해결을 위한 새로운 기능이 추가된 것은 물론, 기존 기능을 한층 개선했습니다.
관리자 전용 연결	서버가 잠금 상태이거나 사용할 수 없는 경우에도 실행 서버에 액세스할 수 있는 관리자 전용 연결 기능을 새롭게 선보였습니다. 이 기능을 이용하면 관리자가 진단 함수나 Transact-SQL 문을 실행하여 서버의 문제를 해결할 수 있습니다.

SQL Server 2005 에디션별 특징

SQL Server 2005는 다양한 규모의 기업의 요구 사항에 가장 부합하는 솔루션을 선택할 수 있도록 유연한 라이센스 전략을 제공하며, 즉시 사용가능한 통합 데이터관리 플랫폼을 구축하기 위해 필요한, 데이터 저장소, 관리, 분석, 리포팅 기능이 단일 제품에 포함되어 있습니다.

초소형 기업에서 초대형 기업까지 확장할 수 있도록 설계된 SQL Server 2005는 모든 고객에게 동일한 성능, 보안, 안정성 및 비즈니스 가치를 제공합니다. SQL Server 2005는 멀티 테라바이트 데이터웨어하우스에서부터 SQL Server CE에디션 기반 포켓 PC(Pocket PC)에 이르는 다양한 구현을 지원합니다.

■ 에디션별 특징

에디션	이점	크기	주요 기능
Express (무료)	간단한 데이터 중심 어플리케이션을 배우고 구축 및 배포하는 가장 빠른 방법	1개의 CPU 1-GB RAM 4-GB DB 크기	**간단한 관리 도구 간단한 보고** 복제 및 SSB 클라이언트
Workgroup	소규모 부서와 성장하는 기업을 위한 가장 합리적인 가격대의 사용하기 쉬운 데이터베이스 솔루션	1-2개의 CPU 3-GB RAM	Management Studio 가져오기/내보내기 제한된 복제 게시 클러스터링 **백업 로그 전달**
Standard	중견 기업 및 대규모 부서를 위한 완벽한 데이터 관리 및 분석 플랫폼	1-4개의 CPU 무제한 RAM	**데이터베이스 미러링** 기본적인 ETL Analysis Services가 지원되는 표준 OLAP 서버 Reporting Services가 지원되는 표준 보고 Data Mining 완전 복제 및 SSB 게시 원시 32 및 64비트 에디션에서 사용 가능 Itanium2 및 x64 지원

Enterprise	비즈니스 크리티컬 엔터프라이즈 어플리케이션을 위한 완전히 통합된 데이터 관리 및 분석 플랫폼	**무제한 확장 및 파티셔닝**	**향상된 데이터 베이스 미러링, 완벽한 온라인 및 병렬 조작, 데이터베이스 스냅샷** 전체 OLAP 및 데이터 마이닝을 비롯한 향상된 분석 도구 **사용자 정의되고 확장성이 뛰어난 임의(ad hoc) 보고 기능을 통한 향상된 보고 기능** 복잡한 데이터 라우팅 및 변환 기능을 통한 향상된 ETL 원시 32 및 64비트 에디션에서 사용가능 Itanium2 및 x64 지원

[참고]
굵은 글꼴은 Microsoft SQL Server 2005에 새로 추가된 기능을 나타냅니다. 각각의 상위 에디션에는 하위 에디션과 동일한 기능이 포함되어 있습니다.

SQL Server 2005 관련자료

- Microsoft SQL Server 홈 페이지
  http://www.microsoft.com/sql

- Microsoft SQL Server TechCenter
  http://www.microsoft.com/technet/prodtechnol/sql/default.mspx

- Microsoft SQL Server Developer Center
  http://msdn.microsoft.com/SQL

- Microsoft SQL Server 신제품발표회 홈페이지
  http://www.ready2005.com/

- SQL Server 2005 에디션별 기능 비교
  http://www.microsoft.com/sql/2005/productinfo/sql2005features.asp

- 가상 Hands-On Labs
  http://msdn.microsoft.com/SQL/2005/default.aspx

- SQL Server 2005 E-Learning
  https://www.microsoftelearning.com/sqlserver2005/

- Microsoft SQL Server TechNet 교육용 웹캐스트
  http://www.microsoft.com/events/series/technetsqlserver2005.mspx

- Microsoft SQL Server MSDN 교육용 웹캐스트
  http://msdn.microsoft.com/SQL/2005Webcasts/

- 비즈니스 인텔리전스 정보
  http://msdn.microsoft.com/SQL/sqlwarehouse/SSIS/default.aspx

- SQL Server 2005 Express 와 XM
  http://lab.msdn.microsoft.com/express/sql/

- SQL Server 2005 관련 백서
  http://www.microsoft.com/sql/2005/techinfo/default.asp

- SQL Server 2005 가격정책 및 라이센스 정책
  http://www.microsoft.com/sql/howtobuy/default.asp

SQL Server Management Studio

SQL Server 2005는 관리자가 수행하는 일상적인 관리업무를 빠르고 쉽게 처리할 수 있도록 하기 위해 그래픽 기반 관리도구를 제공합니다. SQL Server Management Studio는 가장 자주 사용하게 될 그래픽기반 관리도구입니다.

SQL Server Management Studio

SQL Server Management Studio를 사용하여, 대부분의 SQL Server 2005 데이터베이스 관리 업무를 수행할 수 있습니다. SQL Server 2005 시스템을 관리하기 위해서는, SQL Server Management Studio에 대해 잘 알아두어야 합니다.

■ 정의

SQL Server Management Studio는 SQL Server 2005의 모든 구성요소에 대해 접근, 구성, 관리,유지보수하기 위해서 사용하는 통합환경입니다. 이전 버전 SQL Server에서 제공하였던, 엔터프라이즈 관리자, 쿼리 분석기, 분석 관리자와 같은 관리도구의 기능을 통합한 통합 관리도구입니다.

■ 기능

SQL Server Management Studio는 다음과 같은 기능을 관리자에게 제공합니다.

- 관계형 데이터베이스, 분석 서비스 데이터베이스, 리포팅 서비스, SQL Server 통합 서비스, SQL 모바일 데이터베이스에 대한 통합관리기능
- T-SQL , XMLA, MDX, XQuery를 생성하기 위한 그래픽 기반 도구기능

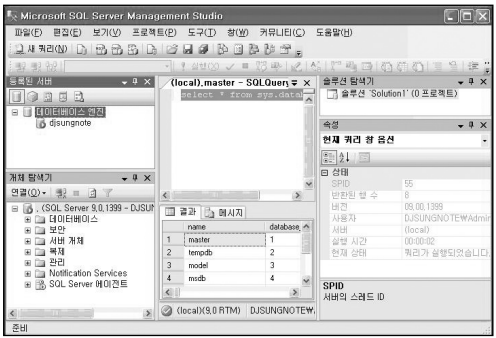
SQL Server Management Studio는 Visual Studio 프레임워크를 기반으로 하며, 쿼리나 스크립트 생성, 작성한 스크립트를 저장 및 유지보수하기 위한 소스 컨트롤 지원, 인터렉티브한 도움말 지원 등과 같은, Visual Studio의 기능이 포함되어 있습니다. 서버와 데이터베이스 개체에 대한 탐색기능을 지원하는 개체 탐색기와, 저장된 SQL Server 솔루션에 대한 관리 기능을 지원하는 솔루션 탐색기도 포함되어 있습니다.

알림

이전 버전 SQL Server에서 지원되었던 도구와는 달리, SQL Server Management Studio는 쿼리나 스크립트를 작성할 때, 데이터베이스에 대한 연결을 항상 활성상태로 유지할 필요가 없습니다.

■ 위치

시작-모든프로그램-Microsoft SQL Server 2005에서 SQL Server Management Studio를 실행시킬 수 있습니다.



〈그림, SQL Server Management Studio〉

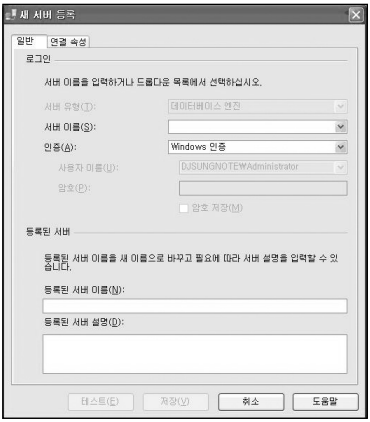
등록된 서버

서버를 등록하여, 서버에 연결하기 위한 연결정보를 저장할 수 있습니다. 서버를 등록하기 위해서는 다음과 같은 사전준비사항이 필요합니다.

- 등록하고자 하는 SQL Server 인스턴스명
- SQL Server 인스턴스에 접근하기 위한 인증 방법-Windows 인증 또는 SQL Server 인증
- SQL Server 인증을 사용하는 경우, SQL Server 인스턴스에 연결할 수 있는 유효한 사용자계정과 비밀번호

[따라하기] SQL Server 등록하기

1. 등록된 서버 창의 데이터베이스 엔진을 오른쪽 클릭한 다음, 새로만들기-서버등록을 클릭합니다.
2. 새 서버 등록 대화상자에서,
   - 서버이름 리스트 박스에서 등록하고자 하는 SQL Server 인스턴스를 찾거나, 특정 SQL Server 인스턴스를 선택
   - 인증 리스트 박스에서 SQL Server 인스턴스에서 사용할 인증 방법을 선택
   - SQL Server 인증을 선택한 경우, 유효한 사용자 계정과 비밀번호를 사용자이름과 암호 텍스트박스에 입력
   - 등록된 서버 이름과 등록된 서버 설명 텍스트 박스에 서버에 대한 이름과 설명을 입력
3. 테스트를 클릭하여, 입력한 상세정보에 대한 유효성검사를 수행합니다.
   - 연결을 테스트 했습니다 라는 메시지가 나타나면, 확인을 클릭하고, 저장을 클릭합니다.
   - 오류메시지가 나타나면, 서버 인스턴스, 인증 방법, 사용자 이름과 암호를 정확하게 입력하였는지 확인하고 재시도합니다.
4. 등록된 서버 창에 새로 추가한 서버가 나타나는지 확인합니다.



〈그림. 새 서버 등록〉

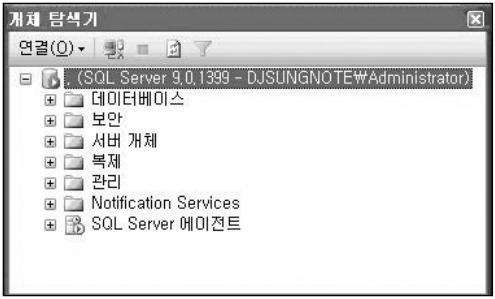
개체 탐색기

개체 탐색기를 사용하여, SQL Server 데이터베이스에 있는 개체를 관리할 수 있습니다. 개체 탐색기는 개체유형별로 그룹화한 트리뷰를 사용하여, 개체를 계층구조 형식으로 표시해 줍니다.

개체 탐색기 창을 사용하여 개체를 조회하고 관리할 수 있습니다. 개체 탐색기를 사용하여 수행할 수 있는 작업에는 다음과 같은 것이 있습니다.

- 데이터베이스와 개체를 생성하고 관리
- 데이터베이스 속성에 대한 조회 및 수정
- 데이터베이스 컨텐츠를 재생성하기 위한 스크립트 생성
- 데이터 원본 생성
- 접근권한 및 사용권한 통제

- 복제 구성
- DTS 패키지 생성
- SQL Server 로그 파일 조회
- SQL Server 에이전트 관리



〈그림. 개체 탐색기〉

솔루션 탐색기

작성한 쿼리와 연결정보를 함께 그룹화하여 저장하면, 향후 쉽게 접근이 가능하고, 함께 실행할 수 있습니다. SQL Server Management Studio 솔루션은 하나 또는 그 이상의 SQL Server Management Studio 프로젝트의 집합입니다. 관련된 연결정보와 쿼리를 편리하게 그룹화하기 위해, 연결정보와 쿼리의 집합체인 SQL Server Management Studio 프로젝트를 사용합니다. SQL Server Management Studio 프로젝트에 소스코드컨트롤 기능을 부여할 수 있습니다. SQL Server Management Studio 프로젝트의 개체는 폴더에 저장된 파일과 같은 구조로 표시됩니다.

■ SQL Server 프로젝트 템플릿

SQL Server Management Studio는 다음과 같은 프로젝트 템플릿을 제공합니다.

- SQL Server Scripts
  T-SQL 쿼리와 관련 SQL Server 연결정보를 생성, 편집, 저장하기 위한 템플릿

- Analysis Server Scripts
  MDX, DMX, XMLA 쿼리와 관련 분석 서버 연결정보를 생성, 편집, 저장하기 위한 템플릿.

- SQL Mobile Scripts
  T-SQL 쿼리와 관련 SQL Server 모바일 연결정보를 생성, 편집, 저장하기 위한 템플릿.

[따라하기] 솔루션 탐색기로 솔루션 생성

1. 시작-모든프로그램-Microsoft SQL Server 2005-SQL Server Management Studio를 클릭합니다. 서버에 연결할 것인지 여부를 확인하는 창이 나타나면, 취소를 클릭합니다.
2. 데이터베이스 엔진에 연결 대화상자에서, Windows 인증을 사용하여, Localhost에 연결합니다.
3. SQL Server Management Studio에서, 파일-새로만들기-프로젝트를 클릭합니다.
4. 템플릿 목록에서, SQL Server 스크립트를 선택하고, 확인을 클릭합니다.(프로젝트 이름과 경로는 기본값을 그대로 사용합니다.)
5. 솔루션 탐색기의 연결 폴더에서 오른쪽 클릭하고, 새 연결을 클릭합니다.
6. 솔루션 탐색기의 연결에서 오른쪽 클릭하고, 제거를 클릭하고, 확인을 클릭합니다. 프로젝트 메뉴에서 새 연결을 클릭합니다.
7. 데이터베이스 엔진에 연결 대화상자가 나타나면, 옵션 버튼을 클릭하여 연결할 데이터베이스, 네트워크 프로토콜과 같은 연결 속성을 조회합니다. 그 다음, localhost 서버에 Windows 인증으로 기본 데이터베이스 연결을 생성하기 위해 확인을 클릭합니다.
8. 솔루션 탐색기의 새로 추가된 연결에서 오른쪽 클릭한 다음, 새 쿼리를 클릭합니다.

9. 솔루션 탐색기의 쿼리 폴더에서 오른쪽 클릭한 다음, 새 쿼리를 클릭합니다. 데이터베이스 엔진에 연결 대화상자가 나타나면, 취소를 클릭합니다.
10. 프로젝트 메뉴에서 새 쿼리를 클릭합니다. 데이터베이스 엔진에 연결 대화상자가 나타나면, 취소를 클릭합니다.
11. 솔루션 탐색기의 현재 솔루션에서 오른쪽 클릭한 다음, 추가-새 프로젝트를 클릭하고, 확인을 클릭합니다.
12. 파일-추가-새프로젝트를 클릭한 다음, 확인을 클릭합니다.
13. 파일-새로만들기-프로젝트를 클릭합니다. 새 프로젝트 추가 대화상자에서 솔루션 드롭다운리스트박스에서 솔루션에 추가 옵션을 지정하고, 확인을 클릭합니다.

SQL Server Management Studio로 쿼리 작성

SQL Server Management Studio에서는 데이터베이스에 연결을 생성하지 않은 상태에서도 T-SQL 쿼리를 생성할 수 있습니다. 예를 들어, AdventureWorks 데이터베이스의 Product 테이블의 항목을 조회하기 위한 쿼리를 작성할 수 있습니다. 또한, 결과 영역에서 쿼리의 결과를 확인할 수 있습니다. 연결을 생성한 다음, SQL Server는 사용자가 작성한 쿼리를 실행하고, 조회된 결과를 결과영역에 표시합니다.

[따라하기] SQL Server Management Studio로 쿼리 작성

1. 시작-모든프로그램-Microsoft SQL Server 2005-SQL Server Management Studio를 클릭합니다. 서버에 연결할 것인지 여부를 확인하는 창이 나타나면, 취소를 클릭합니다.
2. 표준 도구바에서, 새 데이터베이스 엔진 쿼리를 클릭합니다. 서버에 연결할 것인지 여부를 확인하는 창이 나타나면, 취소를 클릭합니다.
3. SQLQuery1 페이지에 다음과 같은 T-SQL 코드를 입력합니다.

```
USE AdventureWorks; SELECT * FROM Production.Product
```

4. 도구바에서 텍스트로 결과 표시를 클릭합니다.
5. 도구바에서 실행을 클릭합니다.
6. 데이터베이스 엔진에 연결 대화상자가 나타나면, Windows 인증을 사용하여 localhost에 연결합니다.
7. 결과영역에 결과값을 확인합니다.

비연결 편집

Management Studio에서는 다양한 방법으로 새 쿼리를 실행할 수 있습니다. 파일 메뉴의 새로 만들기 옵션에서, T-SQL 쿼리, 분석서비스 MDX, DMS, XMLA 쿼리, SQL Server 모바일 쿼리를 유형에 따라 생성할 수 있습니다. 개체 탐색기, 솔루션 탐색기를 통해서도 새 쿼리를 시작할 수 있습니다.

Management Studio의 쿼리 편집 기능은 데이터베이스 연결과 별도로 독립적으로 구현되어 있습니다. 서버에 연결된 상태로 새 쿼리를 시작할 수도 있고, 비연결 모드로 새 쿼리편집을 시작한 다음, 나중에 연결할 수도 있고, 동일한 쿼리에 대해서 실행할 연결 위치를 변경할 수도 있습니다. 스크립트 파일 또는 텍스트 파일로부터 쿼리를 편집할 때, 비연결모드를 사용하는 방법은 매우 도움이 됩니다. 편집모드를 지정하기 위해서는, 새로운 쿼리 편집 아이콘을 익혀 둘 필요가 있는데, 쿼리 편집 아이콘에는 연결 모드, 비연결 모드, 연결 변경의 유형이 있습니다. Management Studio에서 쿼리를 편집할 때는, 편집화면의 공간을 확보하기 위해서, 개체 브라우저의 자동 숨김 기능을 활성화할 수 있습니다.

실행계획보기개선

Management Studio에서는 크게 두 가지의 쿼리실행계획 보기 관련 개선기능을 제공합니다. 기존 쿼리 분석기에서 제공하였던 것과 동일하게 그래픽 기반 실행계획 보기 기능을 제공하지만, Management Studio에서는 각 실행계획 아이콘과 색상이 좀 더 개선되었습니다.

또한, SQL Server가 어떤 작업을 수행하고 있는지에 대해서 좀 더 자세하게 표현할 수 있도록 여러 개의 아이콘이 추가되었으며, 각 아이콘은 SQL Server가 수행하는 작업에 대해서 좀 더 세밀하게 표현할 수 있습니다. 또한 그래픽 기반 쿼리 실행계획 보기 창의 오른쪽 하단에 있는 (+) 표시를 사용하여 대량의 쿼리 실행계획의 전체를 탐색할 수 있습니다. 또한, 쿼리 실행계획의 특정 노드를 선택하면, 좀 더 상세한 정보를 조회하기 위한 속성 대화상자가 표시됩니다.

Management Studio에서 그래픽 기반 실행계획 보기기능과 관련된 가장 큰 개선사항은 쿼리 실행계획을 다른 곳으로 쉽게 이동시킬 수 있다는 것입니다. SET SHOWPLAN_XML ON 옵션을 사용하여 쿼리실행계획을 XML 형식으로 직접 출력할 수도 있습니다. 쿼리를 표 형식으로 실행시킨 경우라면, 결과집합이 실행계획 정보가 포함된 XML 문서에 대한 링크를 포함한 형식으로 반환됩니다. 실행계획 정보가 포함된 XML 문서에 대한 링크를 클릭하면, Management Studio의 XML 편집기상에서 쿼리실행계획 결과를 조회해 볼 수 있습니다.

그래픽 기반 실행계획을 XML 형식으로 내보내기할 수 있습니다. 결과집합에 포함되어 있는 XML 링크에서 오른쪽 클릭한 다음, 해당 결과를 .sqlplan 확장자로 저장합니다. 그 다음, 파일 메뉴의 열기 옵션을 사용하여 저장된 쿼리실행계획 파일을 열면, 저장된 그래픽 기반 쿼리실행계획 XML 문서가 표시됩니다.

[쿼리실행계획 저장]

[그래픽 기반 쿼리실행계획 XML 파일]

쿼리실행계획에 대한 추가정보를 표시하기 위해서는 속성 대화상자를 사용해야 합니다. 쿼리실행계획을 저장하는 방법을 사용하면, 쿼리실행계획을 동료나 지원부서에 Email을 사용하여 보낼 수도 있고, 해당 담당자는 다시 데이터베이스 연결하지 않고서도, 해당 쿼리 실행계획을 확인해 볼 수 있습니다.

SQL Server 2005 구성도구

SQL Server 2005를 설치하면 세 가지 구성관리도구를 사용할 수 있게 됩니다. 보안 강화를 위해, SQL Server 2005는 대부분의 서비스와 외부 자원에 대한 연결 기능이 기본값으로 OFF 상태로 설정되어 있기 때문에, 설치작업 중에 필요에 따라 각 옵션을 적절하게 설정하여, 필요한 도구를 설치하여야 합니다. SQL Server 구성 관리자에서는 SQL Server의 각 서비스를 관리할 수 있고, SQL Server 노출 영역 구성에서는 SQL Server에 발생할 수 있는 의도되지 않는 접근이나 악의적인 접근을 통제할 수 있습니다.

SQL Server 구성 관리자

SQL Server 2005는 여러 개의 하위 서비스로 구성되어 있습니다. 관리자는 SQL Server 2005에 포함되어 있는 각 서비스에 대한 구성정보를 관리하며, 서버 자원을 효율적으로 사용할 수 있도록 관리해야 합니다. 또한, 클라이언트 어플리케이션에서 각 서비스에 연결할 수 있도록 설정해 주어야 합니다. SQL Server 2005의 하위 서비스에 대한 관리를 위해 SQL Server 2005에서는 SQL Server Configuration Manager를 제공합니다.

SQL Server 구성 관리자 사용자 인터페이스는 세 가지 구성요소로 구성됩니다.

1. SQL Server 2005 서비스 노드. 서비스 노드에서 SQL Server 2005 서비스를 시작, 중단, 일시중지할 수 있으며, 각 서비스에 대한 속성을 설정할 수 있습니다.
2. SQL Server 2005 네트워크 구성 노드. 네트워크 구성노드에서 SQL Server 2005 인스턴스가 클라이언트 요청을 수신하기 위해 사용할 네트워크 프로토콜을 구성할 수 있습니다.
3. SQL Native Client 구성 노드. SQL 네이티브 클라이언트 구성 노드는 현재 컴퓨터에서 실행 중인 어플리케이션이 SQL Server 2005 서비스와 통신하기 위해서 사용할 클라이언트 프로토콜을 구성할 수 있습니다.

■ SQL Server Configuration Manager 사용법

시작-모든프로그램-Microsoft SQL Server 2005-구성도구에서 SQL Server Configuration Manager를 실행할 수 있습니다. SQL Server Configuration Manager를 사용하여 SQL Server 서비스를 조회할 수 있습니다. SQL Server에 여러 개의 인스턴스가 설치되어 있는 경우에는, 각 인스턴스별 상태가 조회됩니다. 또한, 서버 네트워크 구성 설정 노드를 조회할 수 있습니다. SQL 네이티브 클라이언트 노드도 조회할 수 있습니다. SQL 네이티브 클라이언트 노드를 확장하고, 클라이언트 프로토콜을 클릭하면, 모든 클라이언트 프로토콜을 상태를 조회할 수 있습니다.

[따라하기] SQL Server Configuration Manager 사용법

1. 시작-모든프로그램-Microsoft SQL Server 2005-구성도구를 선택하고, SQL Server Configuration Manager를 클릭합니다.
2. SQL Server 2005 서비스 노드를 확장한 다음, SQL Server 서비스를 클릭합니다.
3. SQL Server 2005 네트워크 구성 노드를 확장한 다음, MSSQLSERVER에 대한 프로토콜을 선택합니다.
4. 오른쪽 영역에 기본 인스터스에 구성된 프로토콜이 표시됩니다.
5. SQL Native Client 구성 노드를 확장한 다음, 클라이언트 프로토콜을 클릭합니다.
6. 오른쪽 영역에 현재 컴퓨터에서 실행중인 클라이언트 어플리케이션에서 사용하는 프로토콜이 표시됩니다.

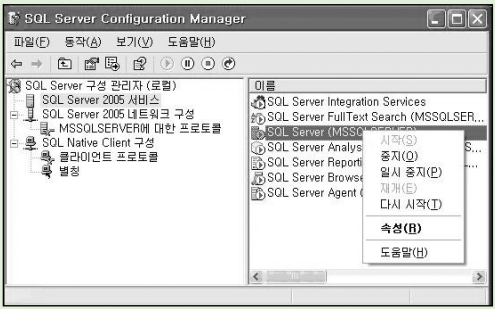
〈그림 SQL Server Configuration Manager〉

서비스 관리

SQL Server 2005를 설치할 때, 각 서비스를 구성할 수 있습니다. 하지만, 필요한 경우, 언제라도 각 서비스를 수작업으로 관리할 수 있어야 합니다. SQL Server Configuration Manager를 사용하여, 각 서비스를 시작, 중지, 일시중지, 재개, 다시시작 할 수 있습니다.

[따라하기] SQL Server 2005 서비스 관리하기

1. 시작-모든프로그램-Microsoft SQL Server 2005-구성도구를 선택하고, SQL Server Configuration Manager를 클릭합니다.
2. SQL Server 2005 서비스 노드를 선택합니다.
3. 오른쪽 영역에서, 특정 서비스 인스턴스를 클릭합니다.
4. 서비스 인스턴스에서 오른쪽 클릭하거나, 동작 메뉴에서 적절한 작업을 클릭합니다.
   - 시작 서비스를 시작합니다.
   - 중지 서비스를 중지합니다.
   - 일시중지 현재 서비스를 유지한 상태로, 신규 연결을 추가하지 않도록 설정합니다.
   - 재개 일시중지된 서비스를 다시 재개합니다.
   - 다시시작 서비스를 중단하였다가 다시 시작합니다.



〈그림 서비스 수동 관리〉

■ 서비스 속성값 조회 및 변경

SQL Server 2005 서비스에는 필요에 따라 조회하고 변경할 수 있는 속성이 포함되어 있습니다. 서비스의 속성 창은 세 가지 페이지로 구성되어 있습니다.

- 로그온. 서비스의 보안 시작계정에 대한 정보가 포함되어 있습니다.
- 서비스. 서비스의 시작 모드와 같은 서비스 일반정보가 포함되어 있습니다.
- 고급. SQL Server의 버전, 설치 경로, 사용언어와 같은 SQL Server 전용 정보가 포함되어 있습니다.

속성 중 대부분은 수정할 수 없는 상태입니다.

[따라하기] 서비스 속성값 변경하기

1. SQL Server 2005 서비스 노드를 선택하고, 오른쪽 영역에서 특정 서비스 인스턴스를 클릭합니다.
2. 서비스에서 오른쪽 클릭하거나 동작 메뉴에서 속성을 클릭합니다.
3. 해당 서비스의 등록정보 창에서 필요에 따라 특정 탭을 선택합니다.
   - 사용자계정을 지정하기 위해서는 로그온 탭을 클릭합니다.
   - 서비스 일반정보를 조회하기 위해서는 서비스 탭을 클릭합니다.
   - SQL Server 2005에 특화된 정보를 조회하기 위해서는 고급 탭을 클릭합니다.
4. 수정하고자 하는 속성을 변경한 다음, 확인을 클릭합니다.

〈그림 서비스 속성 조회 및 변경〉

서버 네트워크 구성

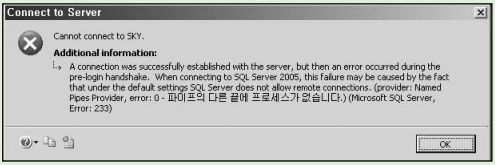
SQL Server 2005 네트워크 구성 노드에서는 SQL Server 2005 인스턴스가 클라이언트 요청을 받기 위해서 사용하는 서버측 프로토콜을 지정할 수 있습니다. 또한, 각 프로토콜별로 서로 다른 주소 매개변수를 지정하기 위해서, 주소 정보를 지정할 수 있습니다.

■ 서버 프로토콜 활성화/비활성화

SQL Server 2005는 최초 설치시 서버 프로토콜의 기본설정값은 에디션별로 차이가 있습니다. 엔터프라이즈 에디션(Enterprise Edition)과 달리 개발자 에디션(Developer Edition)의 경우 TCP/IP 프로토콜의 기본 설정값이 사용안함 상태입니다.

[따라하기] 원격 서버 연결

■ 적절한 서버 프로토콜이 사용안함으로 되어 있는 경우 다음과 같은 오류가 발생합니다

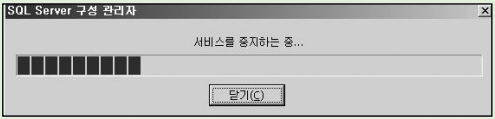


〈그림 오류발생〉

■ 서버 구성 마법사의 서버 프로토콜 상태를 체크하여 봅니다. TCP/IP,Named Pipe 설정이 사용안함으로 되어 있다면 사용으로 바꾸어 봅니다

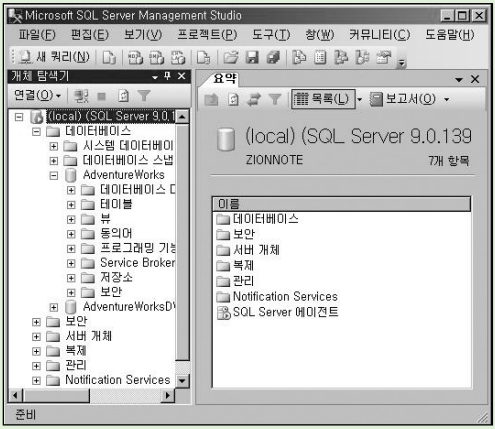
〈그림 서버 프로토콜 상태〉

■ SQL Server 서비스 재시작을 해야 적용됩니다.



〈그림 재시작〉

■ 다시 접속을 시도하여 봅니다



〈그림 연결성공〉

■ 서버 프로토콜 주소 매개변수 수정하기

[따라하기] 서버 프로토콜 주소 매개변수 수정하기

1. SQL Server 2005 인스턴스를 클릭하고, 오른쪽 영역에서 특정 프로토콜을 선택합니다.
2. 동작 메뉴에서 속성을 클릭합니다.
3. 프로토콜에 대한 설정 정보를 수정하고, 확인을 클릭합니다.

정보

네트워크 프로토콜 설정에 대한 좀 더 자세한 정보는 SQL Server 2005 온라인 도움말의 "네트워크 라이브러리와 네트워크 프로토콜" 항목을 참조하십시오.

클라이언트 네트워크 구성

클라이언트 컴퓨터에는 SQL Server 2005 서비스에 연결하기 위해서 사용할 프로토콜과 주소 정보가 구성되어야 합니다. SQL Server Configuration Manager의 SQL Native Client 구성 노드에서 클라이언트 프로토콜에 대한 속성값을 구성할 수 있습니다.

■ 클라이언트 프로토콜 활성화/비활성화

[따라하기] 클라이언트 프로토콜 활성화/비활성화

1. SQL Native Client 구성 노드를 확장합니다.
2. 클라이언트 프로토콜을 클릭한 다음, 오른쪽 영역에서 특정 프로토콜을 선택합니다.
3. 동작 메뉴에서, 사용을 클릭하여, 선택된 프로토콜을 사용하여, 클라이언트가 요청을 보낼 수 있도록 설정할 수도 있고, 사용안함을 클릭하여, 선택된 프로토콜을 사용하여, 클라이언트가 요청을 보낼 수 없도록 설정할 수도 있습니다.

■ 클라이언트 프로토콜 매개변수 변경

[따라하기] 클라이언트 프로토콜 매개변수 변경

1. 클라이언트 프로토콜을 클릭한 다음, 오른쪽 영역에서 특정 프로토콜을 선택합니다.
2. 동작 메뉴에서 속성을 클릭합니다.
3. 클라이언트 프로토콜에 대한 설정을 변경하고, 확인을 클릭합니다.

알림

클라이언트 프로토콜 설정에는 비활성 연결 타임아웃 설정과 같은, 서버에 적용할 수 없는 추가 매개변수가 포함되어 있습니다.

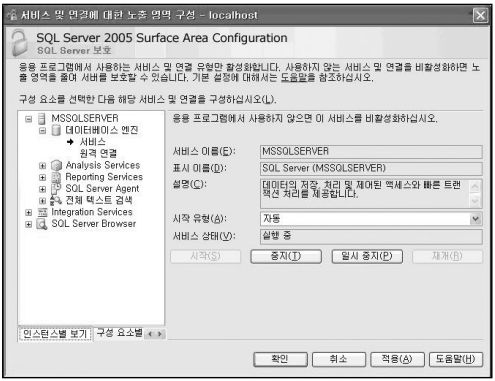
SQL Server 노출 영역 구성도구

SQL Server 2005에서는 기본 구성인 SQL Server 데이터베이스 엔진, SQL 에이전트, 분석 서비스에 추가하여, 통합서비스(과거 데이터변환서비스), 리포트 서비스, 전체 텍스트 검색, SQL 브라우저와 같은 서비스를 추가로 구성할 수 있습니다. 이러한 추가된 서비스가 잠재적인 보안 공격의 접점이 될 수 있습니다. SQL Server 2005에서는 잠재적인 보안 공격의 접점을 최소화하기 위해, 두 가지 새로운 SQL Server 노출 영역 구성 도구를 제공합니다. 서비스 및 연결에 대한 노출 영역 구성도구는 각 서비스를 관리하는 도구이고, 기능에 대한 노출 영역구성도구는 추가기능을 관리하기 위한 도구입니다. SQL Server 2005의 설치가 완료되면, SQL Server 노출 영역 구성 도구에 대한 링크가 나타나고, Windows 시작 메뉴를 통해서도 SQL Server 노출 영역 구성 도구를 실행할 수 있습니다.

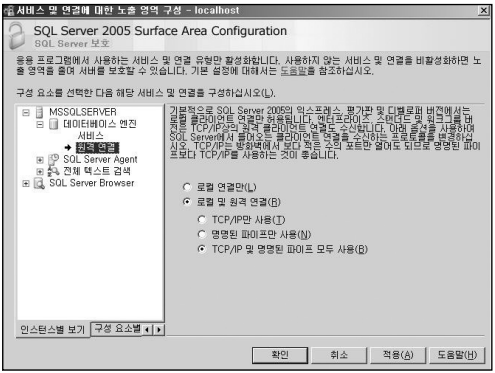
SQL Server 노출 영역 구성 도구를 사용하여 잠재적인 보안 공격에 노출되어 있는 SQL Server의 개별 서비스 및 기능을 통합하여 관리할 수 있으며, 각각의 서비스를 활성화하거나 비활성화할 수 있습니다.

■ 서비스 및 연결에 대한 노출 영역 구성

서비스 및 연결에 대한 노출 영역 구성 도구를 사용하여 다양한 서비스를 활성화하거나 비활성활 수 있습니다. 또한, UDP 포트 1434 에서 수신 중인 SQL 브라우저 서비스를 활성화하거나 비활성화할 수 있고, 명명된 인스턴스에 대한 연결을 관리할 수 있습니다. 원격연결 부분의 기본값은 에디션별로 차이가 있습니다. 개발자 에디션(Developer Edition)의 경우 "로컬 연결만" 입니다. 원격 연결을 허용하고자 한다면 바꾸어 주어야합니다.



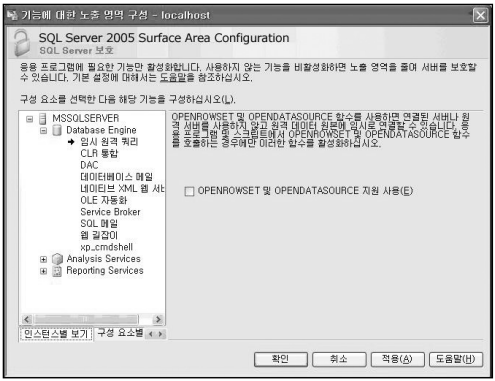
〈그림 서비스 및 연결에 대한 노출영역구성 도구〉



〈그림 원격 연결 - 로컬 영역만〉

■ 기능에 대한 노출영역 구성 도구

기능에 대한 노출 영역 구성도구에서는 임의 원격 쿼리, CLR 통합 기능, 데이터베이스 메일(SMTP), 저장 프로시저, 원격 관리자 전용 연결, 네이티브 웹 서비스(SOAP) 엔드포인트, OLE 자동화 확장 저장 프로시저, 서비스 브로커 엔드포인트, SQL 메일(MAPI) 저장 프로시저, xp_cmdshell, 웹 어시스턴트 등을 포함한 다양한 연결관련 기능을 활성화/비활성화할 수 있습니다. 기본값으로, 모든 연결관련 속성은 비활성화되어 있기 때문에, 사용하기 위해서는 반드시 활성화해 주어야 합니다. 또한, sp_configure 시스템 저장 프로시저를 사용하여 각 기능을 활성화시켜 줄 수도 있고, 관련된 T-SQL 구문을 사용하여 각 엔드포인트를 활성화시킬 수도 있습니다.

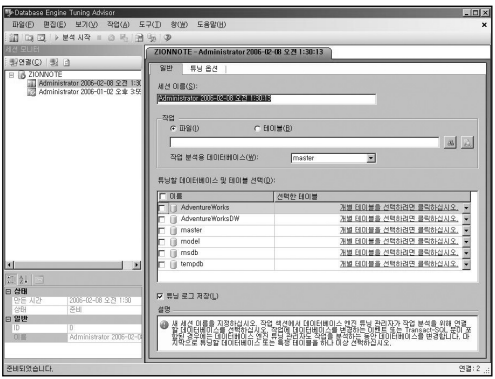


〈그림 기능에 대한 노출영역구성도구〉

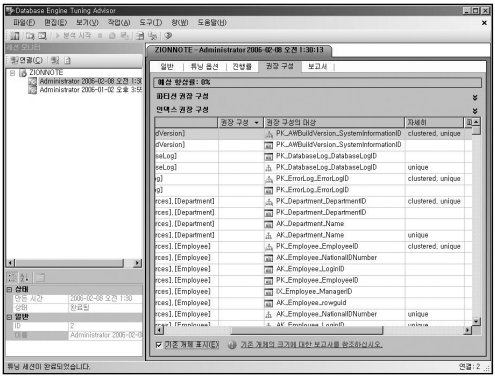
SQL 서버 2005의 성능 도구

데이터베이스 엔진 튜닝 관리자(Database Engine Tuning Advisor)

데이터베이스 엔진 튜닝 관리자는 SQL Server 2005에서 제공하는 도구이며, 이전 버전의 인덱스 튜닝 마법사를 대체하는 기능입니다. 데이터베이스를 튜닝하기 위해서, 데이터베이스 엔진 튜닝 관리자에 분석 대상이 되는 작업부하를 지정해야 합니다. 작업부하를 지정하기 위해서, SQL 프로필러 추적결과, T-SQL 스크립트 파일이 사용됩니다. 또한, 분석 대상이 되는 데이터베이스 및 데이터베이스 개체를 지정할 수 있습니다. 데이터베이스 엔진 튜닝 관리자에서는 지정된 작업부하를 분석하여 선택된 데이터베이스의 인덱스 사용현황을 분석합니다. 작업부하에 대한 분석결과, 데이터베이스 엔진 튜닝 관리자에서는 인덱스에 대한 권고사항을 생성하고, 각 인덱스에 대한 사용현황과 성능 지수를 표시합니다. 데이터베이스 엔진 튜닝 관리자를 사용하면, 데이터베이스 및 SQL Server 내부구조에 대한 이해가 부족한 비전문가라도 인덱스의 사용현황을 분석할 수 있고, 이를 기초로 하여 새로운 인덱스 전략을 수립하고, 데이터를 분할(partition)하기 위한 기준을 판단할 수 있습니다.



〈그림. 데이터베이스 엔진 튜닝 관리자〉



〈그림. 데이터베이스 엔진 튜닝 관리자 결과〉

데이터베이스 엔진 튜닝 관리자 사용하기

■ 새로운 세션 만들기

데이터베이스 엔진 튜닝 관리자에서 분석하기 위해 사용할 세션을 생성합니다. 세션에는 작업부하 원본에 대한 정보가 포함되며, 분석과정에서 사용할 튜닝 옵션과 분석결과물에 대한 정보가 포함됩니다. 세션이 생성되면, 데이터베이스 엔진 튜닝 관리자의 세션 모니터 창에 나타납니다. 세션 모니터 창을 통해 이전에 생성한 세션에 대한 상세정보를 확인할 수 있고, 기존 세션을 삭제할 수도 있습니다.

새로운 세션을 생성하기 위해서는, 파일메뉴의 새 세션 옵션을 선택합니다. 그 다음, 세션에 의미를 부여한 명칭을 지정합니다.

■ 작업부하 지정

데이터베이스 엔진 튜닝 관리자의 일반 탭에서, 데이터베이스 엔진 튜닝 관리자에서 사용할 작업부하 정보를 지정하고, 분석할 데이터베이스를 지정합니다.

데이터베이스 엔진 튜닝 관리자의 일반 탭의 파일 옵션에는 작업부하 원본으로 사용될 SQL 프로필러에서 수집한 추적 파일(.trc)이나 T-SQL 명령이 저장된 SQL 스크립트를 지정합니다. 지정된 파일안의 각 명령은 선택된 데이터베이스에서 실행되며, 해당 작업부하를 최적화하기 위한 권고사항을 생성하는데 판단기준으로 사용됩니다.

일반 탭의 테이블 옵션은 작업부하의 원본으로 사용할 추적 테이블을 지정하기 위해서 사용합니다. 추적 테이블은 SQL 프로필러에서 작업부하로 사용할 추적정보를 수집한 다음, 추적 테이블로 저장하기 옵션을 사용하여 생성합니다.

■ 튜닝옵션 지정

지정한 작업부하를 기준으로 데이터베이스 엔진 튜닝 관리자의 분석작업을 시작하기 전에, 분석과정에서 사용할 튜닝 옵션을 지정해야 합니다. 지정한 튜닝 옵션은, 데이터베이스 엔진 튜닝 관리자가 분석을 수행하고, 권고사항을 생성하는 방법을 지정하기 위해서 사용됩니다. 튜닝 옵션 탭에서 다음과 같은 튜닝 옵션을 지정할 수 있습니다.

- 분석 작업을 종료할 시점을 지정할 수 있습니다.(기본값은 현재 시간을 기준으로 1 시간 이후에 종료하는 것입니다.)
- 데이터베이스에서 사용할 물리적 디자인 구조를 지정합니다. 기존 물리적 디자인 구조를 분석한 다음, 어떤 구조를 유지하고, 어떤 구조를 삭제할 것인지에 대한 권고사항을 생성합니다. 또한, 새로 추가해야 할 인덱스와, 인덱스된 뷰를 제시해 주며, 불필요한 인덱스와 인덱스된 뷰를 삭제할 것을 권고합니다.
- 분할(partition) 기능을 사용할 것인지 여부를 판단하고, 분할(partition)의 수준을 권고합니다.
- 데이터베이스 내에서 어떤 물리적 디자인 구조를 유지할 것인지 판단합니다. 데이터베이스 엔진 튜닝 관리자에서는 기존 물리적 디자인 구조를 그대로 유지한 상태에서 새로운 항목만을 권고하도록 설정할 수도 있고, 기존 물리적 디자인 구조를 모두 제거한 다음, 최적의 구조를 다시 생성하도록 설정할 수도 있으며, 양쪽의 조합도 설정할 수 있습니다.
- 또한, 고급 튜닝 옵션 대화상자를 통해, 데이터베이스 엔진 튜닝 마법사에서 생성하는 스크립트를 온라인 작업기준으로 생성할 수도 있고, 오프라인 작업기준으로 생성할 수도 있습니다.

■ 분석작업수행

데이터베이스 엔진 튜닝 관리자 툴바의 분석시작 버튼을 클릭하면 분석작업을 실행할 수 있다. 진척도 체크 화면이 나타나며, 분석의 각 단계가 진행되는 과정을 살펴볼 수 있습니다. 분석작업과정에서 발생하는 오류나 경고는 모두 진척도 체크 화면에 표시됩니다. 각 단계 작업의 진척도 관련 정보는 튜닝 로그에 기록되게 할 수 있습니다.

■ 작업결과 조사

분석작업이 완료되면, 데이터베이스 엔진 튜닝 관리자에 두 개의 새로운 탭이 추가됩니다. 권고사항 탭에서는 새로 추가해야 할 개체의 목록과 삭제되어야 할 개체의 목록이 나타납니다. 선택된 데이터베이스나 개체에 대한 권고사항을 조회하기 위해 각 목록에 필터를 적용할 수 있습니다. 정의 컬럼의 링크를 클릭하면, 각 권고사항을 적용하기 위한 T-SQL 구문이 표시됩니다. 선택된 튜닝 옵션에 따라, 분할(partition)관련 권고사항과 인덱스 관련 권고사항이 표시됩니다.

보고서 탭에서는 분석과정을 요약한 다음과 같은 사전정의된 보고서가 표시됩니다.

• 문장 비용 보고서

작업부하 파일 또는 테이블에 포함된 각 SQL 구문의 비용관점에서의 성능개선 예측치에 대한 보고서입니다. 구문 비용 보고서를 통해, 권고사항을 반영했을 때 어떤 SQL 구문이 가장 크게 성능이 개선되는지를 판단할 수 있습니다.

• 문장 상세 보고서

작업부하에 포함된 전체 SQL 문장의 목록과 각 SQL 문장의 현재 비용, 권고사항 반영 후 예측 비용을 비교하는 보고서입니다.

• 문장-인덱스 관계 보고서(현재기준)

현재 상태에서 각 쿼리가 어떤 테이블이나 뷰의 어떤 인덱스를 사용하고 있는지를 나타내는 보고서입니다.

• 문장-인덱스 관계 보고서(권고사항)

권고사항을 반영한 이후에 각 쿼리가 어떤 인덱스를 사용하게 될 것인지를 나타내는 보고서입니다.

• 인덱스 사용량 보고서(현재기준)

작업부하에 포함된 쿼리에서 사용하는 인덱스의 목록과 사용량을 나타내는 보고서입니다.

• 인덱스 사용량 보고서(권고사항)

작업부하를 기준으로 권고되는 인덱스의 목록과 예상 사용량을 나타내는 보고서입니다.

• 인덱스 상세 보고서(현재기준)

현재 인덱스의 목록과 상세정보를 나타내는 보고서입니다.

• 뷰-테이블 관계 보고서

작업부하에 포함된 뷰에서 참조하고 있는 테이블에 대한 정보를 나타내는 보고서입니다.

• 작업부하 분석 보고서

분석 결과를 요약하는 보고서로, 각 문장의 유형별(SELECT, INSERT, UPDATE, DELETE)로 그룹화되어 나타나며, 권고사항을 반영했을 때, 어떤 문장이 성능 향상될 것인지, 어떤 문장이 성능상 아무런 영향을 미치게 되는지, 어떤 문장이 오히려 성능이 저하되는지에 대한 정보를 나타내 줍니다.

• 데이터베이스 액세스 보고서

• 테이블 액세스 보고서

작업부하에 포함된 쿼리에서 참조하는 데이터베이스 목록과 참조되는 회수를 나타내는 보고서입니다. 또한 작업부하에 포함된 쿼리에서 참조하는 테이블 목록과 참조 회수도 포함됩니다.

• 컬럼 액세스 보고서

작업부하에 포함된 쿼리에서 참조하는 컬럼 목록과 참조되는 회수를 나타내는 보고서입니다.
보고서 탭의 튜닝 요약 섹션에는 전체적인 튜닝 작업의 통계와 권고사항을 적용했을 때 예상되는 성능 향상 수준이 표시됩니다.

■ 권고사항 적용

동작 메뉴의 권고사항 적용을 클릭하면, 권고사항을 실제로 반영할 수 있습니다. 변경사항을 즉시 반영할 수도 있고, 일정계획을 수립하여 특정 시점에 실행되도록 지정할 수도 있으며, 변경사항 SQL 스크립트를 저장할 수도 있습니다.

■ 분석결과를 XML 형식으로 보기

데이터베이스 엔진 튜닝 관리자에서는 권고사항과 결과보고서를 XML 파일로 내보내기할 수 있습니다. 내보내기 한 XML 파일은 향후 XML 파싱 기능을 지원하는 3rd 파티 도구를 사용하여 분석작업할 수 있습니다.

■ 권고사항을 XML 형식으로 저장

권고사항을 XML 형식으로 저장하기 위해서는, 데이터베이스 엔진 튜닝 관리자의 파일 메뉴에서 세션 정의 내보내기 옵션을 선택하고, 저장할 XML 파일의 명칭을 부여합니다. Microsoft 인터넷 익스플로러와 같은, XML 파싱 기능을 지원하는 도구를 사용하여, 저장된 XML 파일을 열어볼 수 있으며, 저장된 XML 파일에 저장된 정보를 XML 기반 사용자정의 어플리케이션에서 사용할 수 있습니다.
저장된 XML 파일안에는 모든 권고사항이 Configuration 엘리먼트에 포함되어 있습니다.

다음은 Configuration 엘리먼트의 일부를 정리한 것입니다.

```
〈Configuration〉
    〈Server〉
        〈Name〉localhost〈/Name〉
        〈Database〉
            〈Name〉AdventureWorks〈/Name〉
            〈Schema〉
                〈Name〉Production〈/Name〉
                〈View〉
                    〈Name〉vProductAndDescription〈/Name〉
                        〈Recommendation〉
                            〈Drop〉
                            〈Index Clustered=" true" 〉
```

```
                    〈Name〉
                        IX_vProductAndDescription

                    〈/Name〉
                〈/Index〉
                〈/Drop〉
            〈/Recommendation〉
        〈/View〉
            〈Table〉

            〈Name〉TransactionHistory〈/Name〉
            〈Recommendation〉
                〈Drop〉
                    〈Index〉
                        〈Name〉
            IX_TransactionHistory_RefOrdID_RefOrdLineNumber
                        〈/Name〉
                    〈/Index〉
                〈/Drop〉
                〈Drop〉
                    〈Index〉
                        〈Name〉
            IX_TransactionHistory_ProductID
                        〈/Name〉
                    〈/Index〉
                〈/Drop〉
            〈/Recommendation〉
        〈/Table〉
    . . .
〈/Configuration〉
```

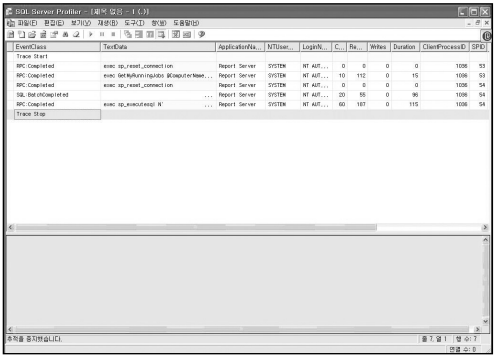
■ 결과보고서를 XML 형식으로 저장

결과보고서를 XML 형식으로 저장하기 위해서는, 보고서를 선택하고 오른쪽 버튼을 클릭한 다음, 파일로 내보내기 옵션을 선택하고, 저장할 XML 파일의 명칭을 입력합니다.

SQL Server 프로필러

SQL Server 2005에서는 SQL Server 프로필러의 개선기능을 통해, 좀 더 발전된 성능을 보장할 수 있고, SQL Server의 상태에 대해서 좀 더 깊은 수준의 정보를 제공해 줍니다.

[SQL Server 2005의 프로필러]



■ 새로운 기능과 개선기능

분석서비스 추적가능

SQL Server 2005의 분석 서비스에서 실행되는 이벤트에 대해서도 추적할 수 있습니다.

쿼리실행계획 및 데드락 추적기능

쿼리실행계획보기와 데드락 이벤트관련에 대해서 개선된 추적기능을 제공합니다.

- 데드락의 발생원인을 분석하기 위해, 데드락 사이클에 대한 그래픽 기반 분석을 지원합니다.

- 쿼리실행계획보기 결과를 XML 형식으로 저장한 다음, XML 파싱기능을 제공하는 다른 도구에서 해당 저장결과를 사용할 수 있습니다.

■ XML로 결과 저장

SQL 프로필러 추적결과를 표준 형식인 ANSI, 유니코드, OEM 형식 뿐만 아니라 XML 형식으로도 저장할 수 있습니다.

■ 데이터 집계

선택된 키 값을 기준으로 SQL 프로필러 데이터를 집계할 수 있습니다. 집계기능을 통해, 특정 이벤트가 발생한 건수를 간단하게 집계할 수 있습니다.

■ 성능모니터와 상관관계분석

특정 시간대에 SQL Server 및 분석 서비스 쿼리추적을 실행합니다. 또한, 사전정의된 성능 모니터 카운터 집합을 선택하고, 해당 시간대의 성능 카운터 정보를 수집합니다. 동일한 시간대에 추적한 쿼리추적결과과 성능모니터 카운터 수집결과를 비교하면, 성능카운터와 쿼리추적결과간의 상관관계를 시각적으로 비교할 수 있습니다.

■ 결과 XML 로 저장하기

XML은 매우 편리한 데이터 형식으로, 점점 지원하는 도구가 많아지고 있습니다. 쿼리추적결과를 XML 파일로 저장하면, 기존 SQL 프로필러 전용 형식으로 저장하는 것에 비해서, 더 많은 도구에서 해당 데이터를 분석할 수 있습니다. 예를 들어, Microsoft 엑셀 또는 사용자정의 어플리케이션에서 XML로 저장된 쿼리추적결과를 활용할 수 있습니다.

■ 추적결과 XML 형식으로 저장

1. SQL 프로필러에서 새로운 추적을 생성하고, 추적을 실행합니다.
2. 추적이 종료되면, SQL 프로필러의 파일 메뉴에서 다른 이름으로 저장의 추적 XML 파일 옵션을 선택합니다.
3. 파일명과 저장할 위치를 지정하고, 저장을 클릭합니다.

SQLCMD 유틸리티와 관리자전용연결(DAC)

SQLCMD 유틸리티

T-SQL 문장을 실행할때 항상 SQL Server Management Studio와 같은 그래픽 기반 도구가 필요한 것은 아닙니다. 예를 들어, 자동 실행되어야 하는 예약된 배치 스크립트를 실행해야 하는 경우에는 그래픽 기반 도구를 사용할 필요가 없습니다. SQL Server 2005에서는 명령줄 기반 T-SQL 문장을 실행하도록 하기 위해, SQLCMD 유틸리티를 새로 지원합니다.

■ SQLCMD 유틸리티란

SQLCMD 유틸리티는 명령줄이나 예약된 배치 작업에서 T-SQL 문장과 스크립트를 실행하도록 하는 유틸리티입니다. SQLCMD 유틸리티는 T-SQL 배치 스크립트를 실행하기 위해 OLE DB를 사용합니다. 이전 버전 명령줄 유틸리티(osql 또는 isql)에서는 ODBC나 DB 라이브러리를 사용하였습니다.

> ***[참고]***
> *SQL Server 2005 환경에서는 SQLCMD 유틸리티와 OSQL 유틸리티를 병행해서 사용할 수 있지만, OSQL 유틸리티는 SQL Server 2005의 차기 버전에서는 더 이상 지원되지 않을 것입니다. 새로운 개발 프로젝트에서는 SQLCMD 유틸리티를 사용할 것을 권고합니다.*

■ SQLCMD 유틸리티의 개선기능

SQLCMD 유틸리티는 이전 버전에서 지원하는 명령행 기반 OSQL 유틸리티의 기능을 확장하여, 변수를 사용할 수 있고, 서버를 동적으로 연결할 수 있으며, 서버정보를 쿼리하고, 호출자 환경에 오류정보를 전달하는 기능을 제공합니다. 또한, 관리자전용연결(DAC) 기능을 통해, 데이터베이스 관리자가 비상상황에서 서버에 연결하고, 문제를 해결할 수 있도록 지원합니다.

SQLCMD 유틸리티에서 T-SQL 문장외에 지원하는 명령은 다음과 같습니다.

명령	설명
GO [count]	지정된 회수만큼 T-SQL 명령을 실행합니다.(기본값은 1번 반복)
RESET	해당 문장의 캐시정보를 삭제합니다.
ED	가장 최근에 수행한 배치에 대한 텍스트 편집기를 실행합니다. SQLCMDEDITOR 환경변수는 사용할 텍스트 편집기를 지정하기 위해서 사용합니다.(기본값은 edit.com)
!! cmd	지정된 운영체제 명령을 실행하고, 결과값을 SQLCMD 유틸리티로 반환해 줍니다.
QUIT	SQLCMD 유틸리티를 종료합니다.
EXIT (statement)	지정된 T-SQL 문장에서 반환하는 첫번째 값을 EXIT 코드를 호출한 호출자에게 전달하고, SQLCMD 유틸리티를 종료합니다.

다음 명령은 항상 콜론(:)이 접두사로 지정되어야 합니다.

명령	설명
:r filename	지정된 파일안의 항목을 문장 버퍼로 읽어 들입니다.
:ServerList	네트워크 상에 존재하는 SQL Server 목록을 표시합니다.
:List	현재 문장 캐시에 항목을 표시합니다.
:Listvar	현재 지정된 스크립트 변수와 변수에 할당된 값의 목록을 표시합니다.
:Error filename | STDOUT | STDERR	모든 오류 출력물을 각각 지정된 옵션에 따라 지정된 파일, 표준 출력 스트림, 표준 에러 스트림으로 전달합니다.
:Out filename | STDOUT | STDERR	모든 쿼리 결과를 각각 지정된 옵션에 따라 지정된 파일, 표준 출력 스트림, 표준 오류 스트림으로 전달합니다.

:Perftrace filename | STDOUT | STDERR	모든 성능 추적 정보를 각각 지정된 옵션에 따라 지정된 파일, 표준 출력 스트림, 표준 오류 스트림으로 전달합니다.
:Connect server [\instance] [timeout] [user_name[password]]	지정된 사용자명과 비밀번호를 사용하여 지정된 SQL Server의 인스턴스에 연결합니다. SQL Server에 연결할 때, 지정된 타임아웃 경과초수를 초과하면 연결시도를 종료합니다. 연결관련 매개변수의 기본값은 SQLCMDLOGINTIMEOUT, SQLCMDSERVER, SQLCMDUSER, SQLCMDPASSWORD 환경변수에서도 지정할 수 있습니다.
:On Error [exit | ignore]	명령을 실행하는 동안 오류가 발생하면 지정된 동작이 수행되도록 지정합니다. EXIT 옵션은 오류가 발생한 경우, SQLCMD 유틸리티가 종료되도록 지정합니다. IGNORE 옵션은 오류메시지를 표시하고 스크립틀 계속 실행되도록 지정합니다.
:SetVar variable value	스크립트 변수를 지정하고, 변수에 값을 할당합니다.
:Help	SQLCMD 명령어에 대한 간략한 설명을 표시합니다.

■ SQLCMD 유틸리티의 명령줄 옵션

SQLCMD 유틸리티는 명령 프롬프트상에서 실행됩니다. SQLCMD 유틸리티는 여러 개의 명령줄 옵션을 지원합니다. 대부분의 옵션은 이전 버전 OSQL 유틸리티에서 지원하였던 옵션과 그 기능이 유사합니다.

명령줄 옵션

SQLCMD 유틸리티에서는 30 개의 명령줄 옵션을 지원합니다. 대부분의 옵션은 주로 SQLCMD 유틸리티를 사용하여, 별도의 사용자의 개입이 필요없는 배치 작업을 수행하기 위해 사용됩니다.

옵션	설명
-?	SQLCMD 유틸리티의 명령줄 옵션에 대한 구문사용법을 표시합니다.
-L[c]	네트워크 상에서 존재하는 SQL Server 목록을 표시합니다. c 옵션을 지정하면, 헤더없이 서버 목록만 표시됩니다. 이 옵션은 다른 옵션과 조합해서 사용할 수 없습니다.
-U login_id [-P password]	지정된 SQL Server 2005에 연결하기 위해서 사용할 로그인 사용자이름과 비밀번호를 지정합니다. -U 옵션과 -P 옵션이 지정되지 않으면, SQLCMD 유틸리티는 현재 SQLCMD 유틸리티를 실행하는 Windows 인증을 사용하여 SQL Server에 연결을 시도합니다. SQLCMDUSER , SQLCMDPASSWORD 환경변수에서도 지정할 수 있습니다.
-E	Windows 인증을 사용하여 연결합니다.(-U 옵션과 -P 옵션을 생략하면 기본값으로 -E 옵션이 지정됩니다.)
-S server_name [\instance_name]	연결할 SQL Server와 인스턴스를 지정합니다. SQLCMDSERVER 환경변수에서도 지정할 수 있습니다.
-H wksta_name	sp_who 저장 프로시저와 SQL Server Management Studio의 활동 모니터에 표시될 워크스테이션 이름을 표시합니다. 기본값은 SQLCMD 유틸리티를 실행하는 컴퓨터이름입니다. SQLCMDWORKSTATION 환경변수에서도 지정할 수 있습니다.
-d db_name	연결할 데이터베이스를 지정합니다. SQLCMDDBNAME 환경변수에서도 지정할 수 있습니다.
-l time_out	OLE DB 공급자에서 로그인에 대한 타임아웃 경과 초수를 지정합니다. 기본값은 8초입니다. SQLCMDLOGINTIMEOUT 환경변수에서도 지정할 수 있습니다.

-t time_out	명령이나 SQL 문장에 대한 타임아웃 경과 초수를 지정합니다. 기본값은 타임아웃이 없이 무제한으로 대기하는 것입니다. SQLCMDTIMEOUT 환경변수에서도 지정할 수 있습니다.
-h headers	컬럼 헤더로 사용할 행 수를 지정합니다. 기본값은 쿼리반환값이 시작하는 시점에 컬럼 헤더가 한 번만 표시되는 것입니다. -h -1 옵션을 사용하면 컬럼헤더가 표시되지 않습니다. SQLCMDHEADERS 환경변수에서도 지정할 수 있습니다.
-s col_separator	사용할 컬럼구분자를 지정합니다. 기본값은 빈칸을 컬럼구분자로 사용합니다. SQLCMDCOLSEP 환경변수에서도 지정할 수 있습니다.
-w column_width	출력화면의 너비를 지정합니다. 기본값은 80입니다. SQLCMDWIDTH 환경변수에서도 지정할 수 있습니다.
-a packet_size	SQL Server와 통신할 때 사용할 패킷 크기를 지정합니다. SQLCMDPACKETSIZE 환경변수에서도 설정할 수 있습니다.
-e	에코 입력
-I	QUOTED_IDENTIFIER 연결 옵션을 설정합니다.
-c cmd_end	사용할 배치 종결자를 지정합니다. 기본값은, GO 키워드를 입력한 라인입니다. T-SQL 예약어는 배치 종결자로 지정할 수 없습니다.
-q "query"	지정된 쿼리를 실행하고, 쿼리의 종료한 다음 계속 연결을 유지합니다. 지정된 쿼리안에는 GO 명령을 입력하지 않아야 합니다.
-Q "query"	지정된 쿼리를 실행한 후 SQLCMD 유틸리티의 연결을 해제하고 종료합니다. 지정된 쿼리안에는 GO 명령을 입력하지 않아야 합니다.
-m error_level	지정된 심각도 수준 이상의 메시지만 오류 메시지 번호, 메시지, 오류 레벨을 표시합니다. 표시할 오류 메시지 레벨은 SQLCMDERRORLEVEL 환경변수에 지정합니다.

-r [0 | 1]	발생한 오류 메시지를 표준에러장치(기본값은 화면)로 출력합니다. -r 0 옵션은 심각도 수준 17이상인 메시지만 전달합니다. -r 1 옵션은 모든 오류메시지를 전달합니다.(PRINT 명령의 결과 포함)
-i input_file[,file2…]	실행할 명령을 키보드로 입력하는 대신 지정된 파일목록에서 입력합니다.
-o output_file	반환값을 화면으로 출력하는 대신 지정된 파일로 저장합니다.
-p[1]	결과값에 성능관련정보를 표시하도록 설정합니다. 1 매개변수가 지정되면, 성능정보가 결과값 마지막 부분에 콜론으로 구분된 원시 데이터값으로 표시됩니다.
-b	오류가 발생한 경우, DOS ERRORLEVEL 값을 반환하고, SQL 유틸리티를 종료하도록 지정합니다.
-u	출력 파일을 유니코드 형식으로 저장하도록 지정합니다.
-R	통화, 날짜를 문자열형식으로 변환할 때 국가별 설정을 사용하도록 지정합니다.
-v var=" value" [var=" value" …]	특정 명칭을 가진 변수를 정의하고, 해당 변수에 값을 할 당합니다.
-A	관리자전용연결을 사용합니다.
-X[1]	시스템 보안에 문제를 일으킬 수 있는 명령(ED, !! 명령 등)을 비활성화시키거나, 경고를 표시한 다음 결과에 따라 실행하도록 설정합니다. -X1이 옵션이 사용되면, 시스템 보안에 문제를 일으킬 수 있는 명령이 실행된 경우, SQL 유틸리티는 오류메시지를 반환하고, 종료됩니다.
-V severitylevel	SQLCMD 유틸리티에서 반환하는 최하위 오류 심각도 수준을 지정합니다..

■ SQLCMD 유틸리티 사용방법

SQL Server 2005의 유효한 계정을 보유하고 있는 사용자는 SQLCMD유틸리티를 사용하여, 명령을 실행하고, 결과값을 반환할 수 있습니다.

명령실행하기

SQLCMD 유틸리티를 사용하여 T-SQL 명령문을 실행하기 위해서는 다음의 절차를 따라하면 됩니다.

1. 명령줄을 실행합니다.
2. 적절한 옵션과 함께 sqlcmd 유틸리티를 실행합니다. Sqlcmd 라는 명령을 입력하면 Windows 인증으로 SQL Server에 연결합니다.
3. 실행하고자 하는 T-SQL 명령이나 SQLCMD 유틸리티 명령을 입력합니다. 여러 개의 명령을 동시에 실행하기 위해서는 여러 행을 입력하면 되고, 여러 행에 걸쳐 한 명령을 입력할 수도 있습니다.
4. GO 명령을 입력하여 명령문 배치를 실행합니다.
5. QUIT 명령을 입력하여 SQLCMD 유틸리티를 종료합니다.

〈그림. SQLCMD 유틸리티로 T-SQL 명령 실행〉

스크립트 실행하기

SQLCMD 유틸리티에서는 T-SQL 명령이나 SQLCMD 명령을 저장한 스크립트 파일을 입력받아 실행할 수 있습니다. 또한, SQLCMD 유틸리티는 다양한 명령줄 옵션을 지원합니다.

SQLCMD 유틸리티를 사용하여 스크립트를 실행하기 위해서는 다음의 절차를 따라하면 됩니다.

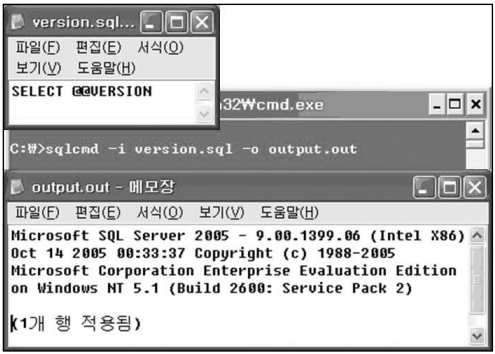
1. 텍스트 편집기를 사용하여, 실행하고자 하는 명령과 T-SQL 문장을 포함한 텍스트 스크립트 파일을 생성합니다.

주의
명령을 실행하기 위해서는 반드시 마지막에 GO 명령을 입력해야 합니다.

2. SQLCMDUSER, SQLCMDPASSWORD, SQLCMDSERVER와 같은, SQL Server에 접속하기 위한 접속정보를 지정하기 위해 환경변수를 설정합니다.

추가정보
SQLCMD 유틸리티에서 명령줄 매개변수로 로그인명, 비밀번호, 서버명을 지정할 수 있다고 하더라도, 해당 접속정보를 배치 스크립트내부에 하드코딩하지 않기 위해, 환경변수를 사용하는 방법을 권고합니다.

3. 명령 프롬프트에서, -i 옵션을 사용하여 입력 스크립트 파일 명칭을 지정하고, -o 옵션을 사용하여 출력파일을 지정합니다.
4. ERRORLEVEL 변수를 점검하여, 스크립트가 정상적으로 실행되었는지를 검사합니다.
5. 출력된 출력파일의 내용을 점검하여, 예상된 결과값이 반환되었는지를 확인합니다.



〈그림. SQLCMD 유틸리티로 스크립트 실행〉

■ 변수사용하기

SQLCMD 유틸리티에서 제공하는 변수사용 기능을 사용하면, 여러 데이터베이스나 서버에서 공통적으로 사용할 수 있는 일반화된 스크립트를 생성할 수 있습니다. SQLCMD 유틸리티에서 제공하는 환경변수와는 별도로, 스크립트 내부에서 :setvar 명령을 사용하여 변수를 정의할 수 있습니다. SQLCMD 환경에서 스크립트로 변수값을 전달할 수도 있으며, -v 옵션을 사용하여 명령줄에서 변수를 전달할 수도 있습니다.

SQLCMD 유틸리티에서 변수를 사용하기 위해서는 다음의 절차를 따라하면 됩니다.

1. SQLCMD 스크립트를 생성합니다. 다음과 같은 구문을 사용하여, 스크립트내에서 변수를 사용할 수 있습니다.

2. $(변수명)

3. 예를 들어:

```
4. SELECT $(colname)
FROM $(tabname)
GO
```

5. 변수의 값을 지정하여 스크립트를 실행하기 위해 -v 옵션을 사용합니다. 예를 들어:

```
6. sqlcmd -i MyScript.sql -o MyScript.out -v colname="name" tabname="sys.databases"
```

〈그림.SQLCMD 유틸리티에서 변수사용〉

■ 관리자전용연결(DAC) 사용방법

이전 버전 SQL Server어의 경우, SQL Server가 무응답상태가 된 상태에서 SQL Server로 연결이 불가능하였습니다. 하지만, SQL Server 2005에서 제공하는 관리자전용연결(DAC)을 사용하면, 서버가 무응답상태에 있다고 하더라도 관리목적의 연결을 생성할 수 있습니다.

관리자전용연결은 별도의 개별 SQL Server 스케줄러를 사용합니다. 그러므로, SQL Server 가 중단되어 있거나 일시중지 상태에 있는 경우를 제외한 모든 상황에서 서버로 관리자전용 연결을 생성할 수 있습니다. 관리자전용연결은 서버당 하나만 지원됩니다. 현재 하나의 관리자전용연결이 되어 있는 상태에서, 제 2 의 관리자전용연결을 생성하고자 하는 경우에는 해당 시도가 거부됩니다.

무응답상태의 서버에 연결을 생성한 다음, 문제의 원인을 진단하기 위한 명령을 수행할 수 있고, 문제를 일으키고 있는 연결을 중단시킬 수 있으며, 필요하다면 서버를 중단시키거나 재시작할 수 있습니다.

관리자전용연결 사용하기

관리자전용연결(DAC)을 연결하기 위해서는 다음의 절차를 따라하면 됩니다.

1 SQLCMD 유틸리티를 -A 옵션을 사용하여 실행합니다.

```
SQLCMD -A
```

2 관리자전용연결을 통해, 문제의 원인을 분석하고, 문제를 해결할 수 있고, 문제의 원인이 되는 연결을 종료시킬 수도 있고, 서버를 종료시킬 수도 있습니다.

〈그림. 관리자전용연결사용〉

추가정보

응답없음 상태의 사용자연결은 장기간 실행되는 트랜잭션으로 인해 특정 프로세스가 잠금획득을 위해 대기하는 경우에 자주 발생하게 됩니다. sp_lock 시스템 저장 프로시저를 사용하여, 특정 자원을 차단(blocking)하고 있는 배타적인 잠금을 설정한 연결을 찾아낼 수 있습니다. 일정 기간내에 종료되지 않은 연결이 있다면, KILL 명령을 사용하여 차단주체가 되는 연결을 강제로 종료시킬 수 있습니다.

SQL Server 2005 데이터베이스 로그 크기

SQL Server 2005는 이전의 SQL Server 2000보다 로그의 크기가 더 크게 증가하는 것으로 나타나고 있습니다. 데이터베이스 디자인시 고려해야 할 점입니다
다음과 같은 쿼리를 SQL Server2000과 2005에서 수행한 후 결과를 비교합니다

```
CREATE DATABASE [test] ON  PRIMARY
( NAME = N' test' , FILENAME = N' C:₩temp₩test.mdf'  , SIZE = 51200KB ,
FILEGROWTH = 10240KB )
 LOG ON
( NAME = N' test_log' , FILENAME = N' C:₩temp₩test_log.ldf'  , SIZE =
102400KB , FILEGROWTH = 51200KB )
GO
ALTER DATABASE [test] SET RECOVERY SIMPLE
GO
USE [test]
GO

Checkpoint

backup log test with no_log
go

dbcc sqlperf(logspace) --로그의 크기를 기록합니다
go

select * into test..c1
from credit..charge
```

```
select * into test..c2
from credit..charge


ALTER TABLE c1
add
        test  varchar(50) null
          default  replicate ( 'a' , 45) with values

ALTER TABLE c2
     add
        test  char(50) null
          default  replicate ( 'a' , 45)     with values

dbcc sqlperf(logspace) --로그의 크기를 기록합니다
go
```

SQL Server 2000

	Database Name	Log Size(MB)	Log Space Used(%)
테스트전 로그	test	99,992188	0,5967263
테스트후 로그	test	99,992188	26,17099

SQL Server 2005

	Database Name	Log Size(MB)	Log Space Used(%)
테스트전 로그	test	99,992188	0,6123526
테스트후 로그	test	149,9922	32,03455

인덱스 관리 개선기능

인덱스는 데이터베이스에 저장된 데이터에 좀 더 빠른 접근을 가능하게 하는 핵심 기능입니다. 하지만, 데이터베이스에 저장된 데이터가 변경됨에 따라 적절한 인덱스를 유지 보수하는 작업은 상당한 시간과 자원을 필요로 합니다. 데이터베이스 관리자가 인덱스를 생성, 변경, 삭제하는 방법을 이해하고 있어야 할 뿐만 아니라, 인덱스가 적절하게 사용될 수 있도록 조각화에 대한 대처 방법을 알고 있어야 합니다.

SQL Server 2005에서는 인덱스 관리측면에서 뛰어난 개선기능을 제공하며, 새로운 DDL 문장과 인덱스 관련 기능이 제공됩니다.

SQL Server 2005의 새로운 인덱스 관리 기능

SQL Server 2005에서도, SQL Server 2000과 마찬가지로 클러스터형 인덱스와 비클러스터형 인덱스를 제공합니다. 또한, 계산된 컬럼이나 뷰에 대해서도 인덱스를 생성할 수 있습니다. 하지만, SQL Server 2005에서는 인덱스를 생성하고 처리하는 방법에 대한 개선된 기능을 제공하고, 새로운 형식의 인덱스를 생성할 수 있습니다.

■ ALTER INDEX 구문

SQL Server 2005의 새로운 인덱스 관련 기능 중 하나는 ALTER INDEX T-SQL 구문에 대한 지원으로, 인덱스를 재생성하고, 재조직하고, 비활성화하고, 인덱스 관련 옵션을 조정하기 위해서 사용합니다. ALTER INDEX 구문을 사용하면, 인덱스의 조각모음을 위해 인덱스를 삭제하고 재 생성할 필요가 없어집니다. 또한, 이전 버전 SQL Server에서 사용하였던 DBCC 명령의 일부를 대체하게 됩니다.

■ 온라인 인덱스 작업

SQL Server 2005에서는 인덱스가 사용중인 상태에서 인덱스에 대한 관리작업을 수행할 수 있기 때문에, 관리작업을 위한 부하가 최소화될 수 있습니다. 이전 버전 SQL Server에서는

인덱스에 대한 작업을 하는 동안 테이블과 인덱스에 대해서 배타적 잠금을 설정하였기 때문에, 인덱스 작업이 완료될 때까지는 해당 테이블에 차단현상이 발생하였습니다.
온라인 인덱스 작업을 수행하기 위해서는 해당 작업을 지원하기 위한 추가적인 디스크 공간이 필요합니다. CREATE INDEX, ALTER INDEX, DROP INDEX 명령을 실행할 때는, 해당 작업을 온라인으로 작업할 것인지 여부를 지정해야 합니다.

■ 병렬인덱스 작업

인덱스를 생성, 변경, 삭제하는 작업에는 정렬 및 스캔을 작업을 위해 여러 개의 CPU를 사용하여 병렬처리 작업을 수행하는 것이 효과적입니다. SQL Server 2005에서는 인덱스 작업에 대한 병렬처리가 자동으로 수행되며, 별도의 MAXDOP 옵션을 지정하여 병렬처리에 사용할 CPU 개수를 지정할 수 있습니다. CREATE INDEX, ATLER INDEX, DROP INDEX T-SQL 명령을 수행할 때 MAXDOP 쿼리 힌트를 사용하여 해당 작업에 대한 병렬처리를 위해 사용할 CPU 개수를 지정할 수 있습니다.

■ 잠금옵션

SQL Server 2005에서는 인덱스를 사용할 때, 새로운 두 가지 잠금 옵션을 제공합니다. 두 가지 잠금옵션은 동시에 활성화될 수 있습니다.

- ALLOW_PAGE_LOCKS
  ALLOW_PAGE_LOCK 옵션이 ON으로 설정되면, 테이블 잠금이나 페이지 잠금을 사용하여 인덱스에 접근하도록 설정합니다. OFF로 설정되면, 페이지 잠금이 사용되지 않습니다.
- ALLOW_ROW_LOCKS
  ALLOW_ROW_LOCKS 옵션이 ON으로 설정되면, 테이블 잠금이나 행 단위 잠금을 사용하여 인덱스에 접근하도록 설정합니다. OFF로 설정되면, 행 단위 잠금이 사용되지 않습니다.

■ 인덱스에 포함되는 컬럼 지정

SQL Server 2005에 비클러스터형 인덱스에는 리프 노드에 인덱스 키가 아닌 데이터도 포함될 수 있습니다. 인덱스에 포함된 컬럼에 대한 접근이 필요한 쿼리가 커버된 쿼리로 실행될 수 있기 때문에 성능을 향상시킬 수 있지만, 해당 데이터를 관리하기 위한 별도의 추가적

인 디스크 공간이 필요합니다. 인덱스에 포함된 컬럼은 이전 버전 SQL Server에서 사용하였던 커버된 인덱스의 기능을 대체하기 위해 사용됩니다.

인덱스에 포함되는 컬럼을 지정하는 기능의 부가효과로 인덱스 키에 대한 크기의 제약이 확장됩니다. SQL Server 2000에서는 인덱스 키값의 최대 크기가 900 바이트로 제한되어 있었습니다. SQL Server 2005의 인덱스에 포함된 컬럼기능을 사용하면, 인덱스의 리프 노드에, 최대 행 크기과 동일하게, 8060 바이트까지 저장할 수 있습니다.

■ 분할 인덱스 (Partition)

분할 인덱스는 SQL Server 2005의 인덱스 관련 주요 개선 중 하나입니다. 분할 인덱스이란, 인덱스를 하나 이상으로 구성된 파일그룹에 분산하여 저장하는 기능입니다. 인덱스 분할은 주로 분할된 테이블에 생성됩니다. 분할 인덱스는 데이터를 여러 개의 파일에 분산시키기 때문에 성능을 향상시킬 수 있고, 경합현상을 줄여줄 수 있으며, IO 작업이 병렬로 처리될 수 있는 가능성을 높여주는 역할을 합니다.
분할 인덱스에 대한 좀 더 자세한 정보는 SQL Server 2005 온라인 도움말의 "분할 테이블 및 인덱스의 개념" 부분을 참조하십시오.

■ XML 인덱스

SQL Server 2005 에서는 XML 데이터형을 지원합니다. CREATE PRIMARY XML INDEX와 CREATE XML INDEX T-SQL 명령을 사용하여 XML 데이터형 컬럼에 XML 인덱스를 생성할 수 있습니다. XML 인덱스는 XML 데이터형 컬럼에 데이터가 변경되었을 때 부가적인 비용을 발생시키기는 하지만, XML 데이터에 대한 XQuery 작업의 성능을 극적으로 향상시켜 줍니다.

XML 인덱스에 대해서는 몇 가지 제약사항이 있습니다. XML 인덱스에 대한 좀 더 자세한 정보는 SQL Server 2005 온란인 도움말의 "XML 데이터 유형 열의 인덱스" 부분을 참조하십시오.

인덱스 생성

CREATE INDEX 명령에 대한 설명과 기능에 대해서 살펴보고, 각 기능을 설명하는 예제 구문과 코드 예제를 소개합니다.

■ Create Index 구문

CREATE INDEX 명령은 인덱스를 생성하는 역할을 합니다. SQL Server 2005에서는 새로운 인덱스 관련 기능을 지원하기 위해 CREATE INDEX 구문에 대한 기능이 확장되었습니다. 새로운 인덱스 관련 기능을 사용하기 위해서, SQL Server Management Studio의 개체 탐색기의 새 인덱스 생성 대화상자를 사용할 수 있습니다.

> ***[참고]***
> *CREATE INDEX 명령에서 유지관리 마법사에서 사용하는 PAD_INDEX, SORT_IN_TEMPdb, IGNORE_DUP_KEY, ONLINE 옵션도 지정하여 사용할 수 있습니다.*

```
CREATE [UNIQUE][CLUSTERED | NONCLUSTERED] INDEX index_name
ON [{database_name.[schema_name]. | schema_name.}]
{table_or_view_name}(column [ASC | DESC][,...n])
[INCLUDE (column_name[,...n])]
[WITH(〈relational_index_option〉[,...n])]
[ON {partition_scheme_name(column_name[,...n])
| filegroup_name | DEFAULT}]
〈relation_index_option〉::=
{ PAD_INDEX = {ON | OFF}
| FILLFACTOR = fillfactor
| SORT_IN_TEMPDB = {ON | OFF}
| IGNORE_DUP_KEY = {ON | OFF}
| STATISTICS_NO_RECOMPUTE = {ON | OFF}
```

```
| DROP_EXISTING = {ON | OFF}
| ONLINE = {ON | OFF}
| ALLOW_ROW_LOCKS = {ON | OFF}
| ALLOW_PAGE_LOCKS = {ON | OFF}
| MAXDOP = number_of_processors }
```

■ 온라인 인덱스 작업

CREATE INDEX 구문의 WITH 절에 ONLINE=ON 옵션을 지정하면, 인덱스 생성작업을 다른 DML 작업과 병행해서 수행할 수 있습니다.

[참고]
xml, text, ntext, image, varchar(max), nvarchar(max), varbinary(max), filestream 데이터형이 사용된 테이블의 경우에는 온라인 인덱스 작업을 수행할 수 없습니다.

```
CREATE INDEX IX_Employee_ManagerID
ON HumanResources.Employee (ManagerID)
WITH (ONLINE=ON)
```

■ 병렬처리수준 지정

기본값으로, 데이터베이스 엔진에서는 최대병렬처리수준(MAXDOP) 구성옵션의 값을 기준으로 인덱스를 생성하는 작업에서 사용할 최대 CPU 수를 결정하게 됩니다. 필요에 따라, 인덱스 작업과 병행하여 처리되는 다른 작업에 영향을 최소화하기 위해, 별도의 MAXDOP 옵션을 지정하여, 인덱스 작업에서 사용하는 CPU를 제한할 수 있습니다.

MAXDOP 옵션을 지정하면, 서버에 구성되어 있는 최대병렬처리수준(MAXDOP) 옵션값보다 우선하여 적용됩니다. MAXDOP =0이라고 지정하면, 현재 데이터베이스 엔진의 작업부하를 고려한 상태에서, 사용할 수 있는 전체 CPU를 인덱스 작업에 사용합니다.

[참고]
ONLINE 옵션과 함께, 인덱스 생성, 변경, 삭제 작업에만 적용되는 별도의 MAXDOP 옵션을 설정할 수 있습니다. 쿼리 최적화기에서는 해당 인덱스를 참조하는 쿼리를 최적화하기 위해서 기존에 설정된 최대병렬처리수준 (MAXDOP) 구성을 사용합니다.

```
CREATE INDEX IX_Employee_ManagerID
ON HumanResources.Employee (ManagerID)
WITH (ONLINE=ON, MAXDOP = 3)
```

■ 인덱스에 포함할 컬럼 지정

인덱스 리프(leaf) 노드에 포함될 컬럼을 추가로 지정할 수 있습니다.
커버된 인덱스를 생성하기 위해, 예제와 같이, AddressLine1 컬럼과, AddressLine2 컬럼을 AddressID컬럼에 대한 인덱스에 추가할 수 있습니다. 하지만, 해당 테이블에 대한 INSERT, UPDATE, DELETE 작업을 수행할 때는, 인덱스에 중복된 데이터를 유지하기 위해 추가 오버헤드가 발생하게 됩니다.

text, ntext, image 데이터형 컬럼에 대해서는 사용할 수 없습니다.

```
CREATE INDEX IX_AddressDetails
ON Person.Address (AddressID)
INCLUDE (AddressLine1, AddressLine2)
```

■ 인덱스 분할(Partition)

CREATE INDEX 명령에 파일그룹과 파티션 스키마를 지정하여 분할 인덱스를 정의할 수 있습니다.

인덱스가 설정된 테이블은 분할된 상태이고, 인덱스 분할을 지정하지 않은 상태라면, 해당 인덱스는 테이블이 사용하고 있는 파티션 스키마와 동일한 위치에 생성됩니다.

```
CREATE INDEX IX_CustomerDetails
ON Sales.Customer(CustomerID)
ON Sales.CustomerScheme(CustomerID)
```

■ XML 인덱스 생성

CREATE [PRIMARY] XML INDEX 명령을 사용하여 XML 인덱스를 생성할 수 있습니다. 테이블에 XML 인덱스를 생성하기 전에 해당 테이블에 반드시 클러스터형 인덱스를 생성해야 합니다. 부가(secondary) XML 인덱스를 생성하기 전에, 해당 XML 컬럼에 기본(primary) XML 인덱스가 생성되어 있어야 합니다. XML 인덱스를 생성할 때는, ONLINE 옵션을 사용할 수 없습니다.

```
CREATE [PRIMARY] XML INDEX index_name
ON [{database_name.[schema_name]. | schema_name.}]table_name
(xml_column_name)
[ USING XML INDEX xml_index_name
[ FOR { VALUE | PATH } ]
[WITH(〈xml_index_option〉[,...n])]
〈xml_index_option〉::=
{ PAD_INDEX = {ON | OFF}
| FILLFACTOR = fillfactor
```

```
| SORT_IN_TEMPDB = {ON | OFF}
| STATISTICS_NO_RECOMPUTE = {ON | OFF}
| DROP_EXISTING = {ON | OFF}
| ALLOW_ROW_LOCKS = {ON | OFF}
| ALLOW_PAGE_LOCKS = {ON | OFF}
| MAXDOP = number_of_processors }
```

다음과 같은 구문으로 XML 인덱스를 생성할 수 있습니다.

```
CREATE PRIMARY XML INDEX IXML_ContactAddContact
ON Person.Contact(AdditionalContactInfo)
```

인덱스 수정

ALTER INDEX 명령에 대한 설명과 기능에 대해서 살펴보고, 각 기능을 설명하는 예제 구문과 코드 예제를 소개합니다.

■ ALTER INDEX 구문

ALTER INDEX 구문은 인덱스를 재생성하고, 재조직화하고, 비활성화하고, 인덱스 관련 옵션을 변경하기 위해서 사용합니다. SQL Server Management Studio의 개체 탐색기의 인덱스 속성 대화상자를 통해 인덱스에 대한 변경작업을 수행할 수 있습니다.

다음 표는 ALTER INDEX 명령을 사용하여 수행할 수 있는 각 작업에 대한 간략한 설명입니다.

키워드	설명
REBUILD	인덱스를 재생성하면, 테이블에 저장된 데이터의 내부구조에 따라 인덱스를 재정렬하게 되고, 인덱스가 사용하고 있는 디스크 공간을 축소하기 위하여, 인덱스를 삭제했다가 다시 생성합니다.
REORGANIZE	테이블에 저장된 데이터의 논리적인 순서에 따라 인덱스의 리프 페이지를 재정렬합니다.
DISABLE	해당 인덱스를 사용하지 못하도록 하기 위해서 특정 인덱스(클러스터형 인덱스도 포함)를 비활성화합니다.
SET	인덱스를 유지관리하고 접근방법을 지정하기 위해 사용되는 인덱스 옵션을 조정합니다.

ALTER INDEX 명령은 인덱스 분할 정보를 변경하거나, 해당 인덱스에 컬럼을 추가하거나 삭제하기 위해서는 사용할 수 없습니다. 이러한 작업을 수행하기 위해서는 CREATE INDEX 명령을 WITH DROP_EXISTING 옵션과 함께 사용해야 합니다.

```
ALTER INDEX {index_name | ALL}
ON [{database_name.[schema_name]. | schema_name.}]
{table_or_view_name}
{ REBUILD [WITH( <rebuild_index_option> [,...n])]
| REORGANIZE [ WITH( LOB_COMPACTION = {ON | OFF})]
| DISABLE
| SET (<set_index_option> [,...n]) }

<rebuild_index_option> ::=
{ PAD_INDEX = {ON | OFF}
| FILLFACTOR = fillfactor
| SORT_IN_TEMPDB = {ON | OFF}
```

```
| IGNORE_DUP_KEY = {ON | OFF}
| STATISTICS_NO_RECOMPUTE = {ON | OFF}
| ONLINE = {ON | OFF}
| ALLOW_ROW_LOCKS = {ON | OFF}
| ALLOW_PAGE_LOCKS = {ON | OFF}
| MAXDOP = number_of_processors }

〈set_index_option〉::=
{ IGNORE_DUP_KEY = {ON | OFF}
| STATISTICS_NO_RECOMPUTE = {ON | OFF}
| ALLOW_ROW_LOCKS = {ON | OFF}
```

■ 인덱스 재생성

REBUILD 절을 통해, FILLFACTOR, PAD_INDEX, SORT_IN_TEMPDB, IGNORE_DUP_KEY, STATISTICS_NORECOMPUTE 등의 옵션을 지정하여 인덱스의 구조를 변경할 수 있습니다.

온라인 작업으로 크기가 큰 인덱스(128 익스텐트 이상의 인덱스)를 재생성하게 되면, 데이터베이스 엔진은 해당 인덱스가 사용하고 있던 기존 디스크 공간을 할당해제하기 전에 새로운 디스크 공간을 할당한 다음, 인덱스관련 데이터를 복사한 다음 재정렬작업을 수행합니다. 반드시 인덱스 재생성 작업을 수행하기 전에는 해당 작업을 수행할 수 있을 만한 충분한 디스크 공간을 확보해 두어야 합니다.

[참고]
클러스터형 인덱스를 재성성한다고 해서 자동으로 모든 비클러스터형 인덱스가 재생성되지는 않습니다. 하지만, ALTER INDEX ALL … REBUILD … 명령을 사용하여 단일 작업으로 특정 테이블에 존재하는 모든 인덱스를 삭제하고 다시 생성할 수 있습니다.

REBUILD 절에 ONLINE=ON 옵션을 지정하면, 온라인 작업으로 인덱스에 대한 변경작업을 수행할 수 있고, MAXDOP 옵션을 지정하면, 별도의 최대병렬처리정도를 지정할 수 있습니다.

```
ALTER INDEX IX_Employee_ManagerID
ON HumanResources.Employee
REBUILD WITH (FILLFACTOR = 80)
```

■ 인덱스 재조직화

인덱스를 재조직화하면, 인덱스를 사용하여 순차적인 행을 대량으로 스캔하는 작업을 수행하는 쿼리의 성능을 향상시킬 수 있습니다. 재조직화된 인덱스는 이미 할당되어 있는 동일한 인덱스 페이지를 재사용하게 되며, 인덱스 재조직화의 결과로 완전히 빈 상태가 된 페이지만 할당해제하게 됩니다. 또한, 인덱스 재조직화작업은 Tempdb에 추가적인 작업공간을 필요로 하지 않게 되며, 모든 데이터는 기존 인덱스 페이지내에서 재분배됩니다.

REORGANIZE 절에 지정할 수 있는 유일한 옵션은 LOB_COMPACTION으로, 대량개체데이터(image, text, ntext, varchar(max), nvarchar(max), varbinary(max)형 데이터)도 재조직화하도록 설정합니다.

인덱스 재조직화작업은 항상 온라인으로 수행되기 때문에, 병행해서 수행되는 동시성 쿼리나 데이터 변경작업을 차단하지 않습니다. MAXDOP 옵션을 별도로 지정할 수 없으며, 서버에 설정된 최대병렬처리정도(MAXDOP) 옵션도 무시됩니다.

```
ALTER INDEX IX_Employee_ManagerID
ON HumanResources.Employee
REORGANIZE WITH (LOB_COMPACTION = ON)
```

■ 인덱스 비활성화

쿼리 최적화기는 SQL 문장에 대한 실행계획을 수립할 때, 비활성화되어 있는 인덱스는 고려의 대상으로 포함시키지 않습니다. 비활성화되어 있는 인덱스에 대한 참조 힌트가 설정된 경우에는 쿼리 최적화기에서 오류를 발생시킵니다. 인덱스가 비활성화되면, 해당 인덱스에 연결된 모든 제약조건과 기본키 및 외래키 제약조건도 함께 비활성화됩니다.

데이터베이스 엔진은 비활성화된 인덱스를 유지보수하지 않습니다. ALTER INDEX… REBUILD… 명령이나 CREATE INDEX… WITH DROP_EXISTING 명령을 사용하여, 인덱스를 다시 활성화할 수 있습니다. ALTER TABLE 명령을 사용하여 비활성화된 제약조건도 반드시 활성화시켜 주어야 합니다.

> ***[참고]***
> *테이블에 대해 클러스터형 인덱스를 비활성화한 경우에는, 테이블이 OFFLINE 상태로 표시됩니다. 모든 어플리케이션에서는 해당 클러스터형 인덱스가 활성화될 때까지는 해당 테이블을 사용할 수 없습니다. 물론, SELECT 쿼리를 수행할 수도 없습니다.*

```
ALTER INDEX IX_Employee_ManagerID
ON HumanResources.Employee
DISABLE
```

■ 인덱스 옵션 변경

SET 절을 사용하여 인덱스에 사용하는 ALLOW_ROW_LOCKS, ALLOW_PAGE_LOCKS, IGNORE_DUP_KEY, STATISTICS_NORECOMPUTE 옵션을 변경할 수 있습니다.

```
ALTER INDEX IX_Employee_ManagerID
ON HumanResources.Employee
SET (ALLOW_PAGE_LOCKS = ON)
```

인덱스 삭제

DROP INDEX 명령을 사용하여 테이블에 설정된 인덱스를 삭제할 수 있습니다. DROP INDEX 명령은 XML 인덱스를 포함한 전체 인덱스에 대해서 사용될 수 있습니다. SQL Server Management Stuoio의 개체 편집기의 해당 인덱스에 대한 단축메뉴에서 삭제 옵션을 사용하여, 인덱스를 삭제할 수도 있습니다. DROP INDEX 명령에 대한 설명과 기능에 대해서 살펴보고, 각 기능을 설명하는 예제 구문과 코드 예제를 소개합니다.

■ DROP INDEX 구문

DROP INDEX 명령에 대한 옵션은 클러스터형 인덱스를 삭제하는 경우에만 사용할 수 있습니다. MOVETO 옵션은 인덱스를 삭제한 다음, 데이터를 저장할 위치를 지정하기 위해서 사용합니다.

```
DROP INDEX index_name
ON [schema_name.]{table_or_view_name}
[ WITH (〈drop_index_option〉 [,...n]) ]
〈drop_index_option〉 ::=
{ ONLINE = {ON | OFF}
| MAXDOP = number_of_processors
| MOVETO { partition_scheme_name(column_name[,...n])] |
filegroup_name | DEFAULT }
```

■ 인덱스 삭제

SQL Server 2005에서는 DROP INDEX 명령에 대한 구문이 변경되었습니다. 이전 버전 SQL Server에서 사용하던 마침표(.) 형식의 구문대신, ON 절을 사용하여 삭제할 인덱스가 설정된 테이블을 지정합니다.

비활성화되어 있는 비클러스터형 인덱스도 삭제할 수 있습니다.

```
DROP INDEX IX_Employee_ManagerID
ON HumanResources.Employee
```

[참고]
이전 버전에서 사용하던 구문이 대부분 그대로 사용할 수 있기는 하지만(물론, XML 인덱스에 대해서는 예외), 신규 개발 프로젝트에서는 이전 버전에서 사용하던 구문을 사용하지 않는 것이 바람직합니다.

■ 클러스터형 인덱스를 위한 옵션지정

DROP INDEX 명령은 클러스터형 인덱스를 삭제하는 경우에 대해서만, MAXDOP 옵션과 ONLINE 옵션을 지원합니다.

비활성화되어 있는 클러스터형 인덱스는 삭제할 수 없으며, 대형개체데이터(LOB)가 포함된 클러스터형 인덱스의 경우에는 ONLINE 옵션을 사용할 수 없습니다.

```
DROP INDEX IX_Cluster
ON Sales.Customer
WITH (ONLINE = ON, MAXDOP = 3)
```

■ 클러스터형 인덱스 데이터 위치재조정

클러스터형 인덱스를 삭제하는 경우에는, MOVETO 옵션을 사용하여 리프 페이지에 저장된 데이터 행을 저장할 위치를 지정할 수 있습니다. 클러스터형 인덱스가 삭제되면, 데이터는 지정된 위치에 테이블로 다시 생성됩니다. 데이터를 저장할 대상 위치로 파티션 스키마나 파일그룹을 지정할 수도 있습니다.

```
DROP INDEX IX_Cluster
ON Sales.Customer
WITH (MOVE TO NewCustomerScheme)
```

인덱스 조각화 관리

초기에는, 인덱스에 포함된 데이터는 기존 테이블의 행 순서에 따라 잘 정렬되어 저장됩니다. SQL Server 2005 데이터베이스 엔진에서는 테이블에 행이 삽입, 변경, 삭제될 때마다 자동적으로 인덱스를 변경합니다. 하지만, 테이블의 데이터가 변경되는 것처럼, 데이터베이스 파일에 저장된 물리적인 저장위치가 변경되는 것은 아닙니다. 테이블의 데이터가 변경되면, 인덱스가 실제 테이블의 물리적인 행 순서와 일치하지 않으며, 결국 성능저하를 일으키는 원인이 될 수 있습니다. 이러한 현상을 조각화라고 합니다. 결과적으로, 조각화를 제거하기 위해 인덱스에 대해 재조직화 또는 재생성작업을 수행해야 합니다.

■ 재조직화(defrag) vs 재생성(rebuild)

인덱스의 조각화정도에 따라 인덱스를 재조직화할 것인지, 재생성할 것인지를 결정하게 됩니다. 클러스터형 인덱스 또는 비클러스터형 인덱스를 재조직화하게 되면, 인덱스의 리프 노드의 조각화를 제거하고, 인덱스에 설정된 채우기비율(Fillfactor)에 따라 내부노드의 정보를 정리하게 됩니다. 이러한 조각모음 작업결과, 그동안 행 삭제 및 변경으로 인해 발생한 빈 공간을 제거하기 때문에, 인덱스는 그만큼 더 작게 축소되며, 실제 테이블과 동일한 순서로 리프 노드가 재정렬됩니다. 인덱스 재조직화작업은 테이블에 포함된 대량개체컬럼(LOB)도 정리하게 됩니다.

인덱스의 조각화정도가 매우 극심한 경우에는 인덱스를 재생성하는 것이 바람직합니다. 인덱스 재생성작업은 인덱스를 삭제했다가 다시 생성합니다. 인덱스를 재생성하는 경우, 채우기비율과 같은 인덱스 관련 설정을 변경할 수 있습니다. 인덱스 재조직화의 경우는, 이러한 인덱스 관련 설정을 변경할 수 없습니다.

■ 조각화 정보 수집

인덱스를 재생성하거나 재조직화하는 작업은 상당한 비용과 자원을 소모하는 작업입니다. 인덱스 재생성작업이 인덱스 재조직화 작업에 비해 더 높은 비용을 소모합니다 그러므로, 인덱스의 조각화정보를 조사하여 반드시 필요한 경우에만 인덱스 재생성작업을 수행하는 것이 바람직합니다. sys.dm_db_index_physical_stats 함수를 사용하여, 인덱스의 조각화정도를 수집할 수 있습니다.

다음 예제는 현재 데이터베이스내의 모든 테이블에 대해서 인덱스 조각화정보를 수집하기 위해, sys.dm_db_index_physical_stats 함수를 사용하는 방법을 나타내고 있습니다.

```
SELECT TableName, IndexName, AvgFragmentation
FROM sys.dm_db_index_physical_stats(DEFAULT, '*' , DEFAULT, 'DETAILED' )
```

sys.dm_db_index_physical_stats 함수에 전달되는 매개변수는 순서에 따라 다음과 같습니다.

- **@TableName.** 정보를 수집할 테이블을 지정합니다. 특정 테이블의 이름을 지정할 수도 있고, DEFAULT, NULL 을 지정할 수도 있습니다. DEFAULT 또는 NULL이 지정되면, sys.dm_db_index_physical_stats 함수는 현재 데이터베이스의 모든 테이블에 대한 정보를 반환합니다.

- **@IndexName.** 정보를 수집할 인덱스를 지정합니다. 특정 인덱스의 이름을 지정할 수도 있고, DEFAULT, NULL, '*' 를 지정할 수도 있습니다. DEFAULT와 NULL이 지정되면, 기본 테이블에 대한 정보(또는 클러스터형 인덱스에 대한 정보)만 반환합니다. '*' 이 지정되면, 테이블에 존재하는 모든 인덱스에 대한 정보가 반환됩니다.

- **@PartitionId.** 인덱스의 분할파티션 ID 번호를 지정합니다. DEFAULE, NULL,0 으로 지정되면 모든 분할파티션에 대한 정보가 반환됩니다.

- **@Mode.** 요청된 정보를 수집하기 위한 스캔 수준을 지정합니다. LIMITED 옵션을 사용할 것을 권고하며, 부모-수준의 페이지만 읽기 때문에 매우 빠르게 작업을 수행할 수 있습니다. SAMPLED 옵션이 지정되면, 부모-수준의 페이지와 샘플링된 리프 페이지를 읽게 되며, DETAILED 옵션이 지정되면, 부모-수준의 페이지와 모든 리프 페이지를 읽게 됩니다.

sys.dm_db_index_physical_stats 함수에 대한 좀 더 자세한 정보는, SQL Server 2005 온라인 도움말의 T-SQL 참조 부분을 참조하십시오.

분할된 테이블과 인덱스 (Partitioned Tables and Indexes)

분할(Partitioning)은 대용량 테이블 또는 인덱스를 보다 편리하게 운영 관리할 수 있는 구현 방안을 제공할 뿐만 아니라 한 테이블 내에서 분할된 하위 집합(행 단위)을 빠르고 효율적으로 관리 및 액세스할 수 있는 성능 이득 또한 제공됩니다. 이전에도 다양한 형태로 제공되던 분할 기능이 SQL Server 2005에서는 어떻게 달라졌는지 그 특징과 구현 방안을 예제를 통해서 살펴보겠습니다.

대용량 테이블의 유지 관리 문제

대용량 테이블의 기준은 무엇일까요? 전문가마다 조금씩은 다른 기준을 제시합니다. 쿼리 최적화에 관련된 한 서적에서는 인덱스 구조 상의 구성 수준이 4, 5 수준을 넘어가는 경우에 해당한다고 표현하기도 합니다. 또 한 사람 "Microsoft SQL Server 2000 High Availability"의 공동 저자인 Kimberly의 경우엔 아래와 같이 충분히 공감할만한 재미있는 내용들로 규정하고 있습니다.

■ 대용량 테이블은:
- 원하는 대로 동작을 하지 않는 경우
- 생각보다 많은 유지 보수 시간과 비용을 요구하는 경우
- 일반적이지 않은 작업이 수행되거나 엄청난 차단(Blocking)을 유발하는 경우
- 유지 보수 (인덱스 재생성 등)에 소요되는 시간이 다른 사용자의 작업을 방해할 정도로 장시간 수행되는 경우

결국 대용량 테이블은 한 테이블의 동일 데이터에 대해서 사용자의 요구 사항이 다양하고 그로 인한 쿼리의 접근 방법이 각각 달라지므로 그 많은 경우의 성능을 보장하기가 힘들어지며 또한 유지 보수에 들어가는 시간과 비용이 그러한 사용자의 다양한 데이터 처리 요청에 응답하는데 필요한 가용성(Availability)를 제한할 정도로 상당한 경우에 해당합니다.

예를 들어, 매출 테이블의 경우 마감 전의 데이터는 읽기/쓰기가 자주 일어나는 반면 마감 후의 데이터는 주로 읽기 작업으로 통계 분석 혹은 보고서 작업 등에 사용이 됩니다. 쓰기 작업이 수행되는 데이터를 위해서 인덱스 재 생성과 같은 테이블 관련 유지 보수 작업을 수행 시, 마감 이후의 정적인 데이터까지 모두 그 작업의 범위에 포함되게 되고 이는 유지 보수 작업의 시간과 비용을 초과 발생시킬 뿐만 아니라 실질적으로 불 필요한 작업에 해당이 되는 것입니다. 이것은 관리 비용의 손실이라도 볼 수 있는 것이죠. 더구나 유지 보수 작업을 수행하는 동안에 발생하는 성능 문제, 차단 문제, 백업에 들어가는 크기와 시간, 더 많은 용량의 백업 장비 요구 등을 포함한 추가 비용 등 운영 시스템 전반에 나쁜 영향을 미칠 수가 있는 것입니다.

그렇다면 이러한 문제점들을 기존엔 어떻게 해결해 왔을까요?

■ 분할(Partitioning)의 접근 방법들

수작업으로 처리-어플리케이션을 통한 분할

가장 보편화된 접근 방법으로, 테이블을 여러 개의 물리적인 테이블로 분할(비정규화모델의 하나인 "수평 분할"에 해당) 해서 생성하고, 저장 프로시저나 뷰 또는 어플리케이션을 사용하여 필요한 테이블을 처리하도록 조정합니다. 상대적으로 설계가 복잡하고 어플리케이션을 포함한 유지 관리의 어려움이 발생할 가능성이 있습니다. 뷰의 경우, 분할된 테이블을 UNION으로 결합하므로, 쿼리할 때, 모든 테이블을 액세스하는 성능 상의 문제가 발생할 수 있습니다.

수작업으로 처리-이력 테이블을 통한 분할

이력 테이블을 별도로 두고 주 별, 월 별, 년도 별 혹은 특정 기간 단위로 데이터를 이동시키는 방법입니다. 마감 작업을 수행하는 경우, 마감 데이터에 대해서 집계된 형태로 이력 관리도 가능합니다.

SQL Server 7.0의 분할된 뷰 (Partitioned View)

수평 분할에 의해서 분할된 테이블을 UNION으로 결합한 뷰의 경우, 근본적인 성능 상의 문제를 가지고 있었습니다. SQL Server 7.0에서는 분할 테이블의 CHECK 제약조건을 설정하고, 이를 활용하여 불필요한 테이블에 접근이 일어나지 않도록 쿼리 계획을 생성할 수 있었습니다. 하지만, CHECK 제약조건을 활용하는 기능도 SELECT 작업에서만 제한적으로 지원하였습니다.

SQL Server 2000의 분할된 뷰

이제 INSERT, UPDATE, DELETE 구문에 대해서도 적용이 될 뿐만 아니라, Inline 테이블-값 함수와 연동한 실행 시점의 조건 별 쿼리 계획 산출 기능, Instead-Of 트리거를 활용한 다양한 업무 로직 구현 그리고 분산 파티션 뷰(Distributed Partitioned View) 기능까지 확장되었습니다. 그러나 여전히 많은 제약사항을 가지고 있었습니다

SQL Server 2005의 분할된 테이블과 인덱스

SQL Server 2005에서 분할된 테이블 하위 집합의 설계, 구현, 개발, 관리가 단순화되었습니다. 또한 분할 인덱스 지원, 조인 테이블과의 정렬 기능 등 다양한 성능 이득을 얻을 수 있도록 지원됩니다.

이제 SQL Server 2005 달라진 분할 기능들을 하나씩 구현하고 경험해 봅시다.

SQL Server 2005의 분할 테이블은 두 가지 구성 요소를 통해서 분할 구조를 결정합니다. 바로 분할 함수와 분할 스키마입니다.

분할 함수 (Partition Function)

분할 함수는, 테이블 전체 데이터를 어떤 기준에 따라서 분할할 것인지를 결정하는 논리를 포함합니다. 현재 SQL Server는 "범위 분할(Range Partition)" 방법을 사용해서 분할에 기준으로 설정합니다. 즉 특정 데이터 범위에 의해서 분할 기준이 정의되는 것입니다. 또한, 그 범위의 구분은 분할 경계(LEFT, RIGHT 기준)를 지정하고 조정함으로써 변경될 수 있습니다. 일반적으로 이러한 범위는 데이터의 일정한 순서를 기준으로 하거나 별도의 데이터 그룹을

기준으로 할 수 있습니다. 예를 들어 매출 테이블의 매출 일자 열에 대해서 년(혹은 월) 별로 분할하는 방식을 생각해 볼 수 있습니다.

[따라하기] 분할 함수 만들기

```
/*
  연습용 DB 생성
*/
CREATE DATABASE Sample
ON PRIMARY (
   NAME = SampleSystem
, FILENAME =  'C:₩Temp₩SampleSystem.mdf'
)
LOG ON (
   NAME = SampleLog
, FILENAME =  'C:₩Temp₩SampleLog.ldf'
)
GO

/*
  분할 함수 생성
*/
USE Sample
GO
CREATE PARTITION FUNCTION Annual_Range (DATETIME)
AS RANGE RIGHT
FOR VALUES (
                    -- 분할-1. 2001년과 그 이전 범위 분할(RIGHT 경계 기준)
  '2002-01-01'   -- 분할-2. 2002년 범위에 대한 분할 영역
, '2003-01-01'   -- 분할-3. 2003년 범위에 대한 분할 영역
```

```
, '2004-01-01'  -- 분할-4, 2004년과 그 이후 범위에 대한 분할 영역
)
```

■ 중요 인수에 대한 설명

[LEFT | RIGHT]
VALUES 절에 경계 값(Boundary Value)내의 포함된 값들이, 왼쪽에서 오른쪽으로 정렬되었을 때, 경계 값을 기준으로 왼쪽에 속하는지 오른쪽에 속하는지(경계 값을 포함해서) 결정합니다. 디폴트는 LEFT 입니다.

경계 값(Boundary Value)
분할 테이블 또는 인덱스의 각 분할에 대한 경계 값을 지정합니다. 999를 초과하지 않아야 하며, 실제 생성되는 분할(Partition) 수는 [경계 값+1] 입니다. +1 에 해당하는 분할 영역은 [LEFT | RIGHT] 에 따라서 그 기준이 달라집니다.

위 예제를 정리하자면, RIGHT 범위을 지정했으므로, 202년부터 2005년까지의 경계 값(1월 1일)을 기준으로 오른쪽에 분할 영역이 생성됩니다(ex. 〉= 2002-01-01). 마지막으로 가장 왼쪽의 경계 값인 2002-01-01 이전 범위에 대해서는 빈 분할이 발생되는 것이며 이는 LEFT 범위의 경우에 반대가 됩니다. 결국 RIGHT 범위에서는 범위 영역에 시작 값을 경계 값으로 지정하고, LEFT 범위에서는 범위 영역의 마지막 값(ex. 〈= 2002-12-31) 을 경계 값으로 지정할 수 있는 것입니다.

■ $PARTITION 함수

$PARTITION 함수를 사용하여 다음과 같은 메타데이터를 액세스할 수 있습니다.

- 분할 테이블의 분할 하위 집합에 있는 모든 행에 액세스
- 각 분할에 행 수
- 특정 분할 키 값이 포함된 행이 있는 분할이나 행이 입력된 위치 확인

[따라하기] $PARTITION 함수 사용하기

```
/*
   분할 값이 포함된 분할 번호 확인
*/
SELECT $partition.annual_range( '2001-01-01' )
------------
1
(1 row(s) affected)

SELECT $partition.annual_range( '2002-01-01' )
------------
2
(1 row(s) affected)

SELECT $partition.annual_range( '2003-12-31 23:59:59.996' )
------------
3
(1 row(s) affected)

-- 그럼, 아래는 몇 번째 분할에 포함될까요?
-- DateTime 데이터 형의 반올림 규칙을 생각해보세요.
SELECT $partition.annual_range( '2003-12-31 23:59:59.999' )

/*
  OrderDate 칼럼 값을 기준으로 4번째 분할에 포함되는 행 집합 반환
*/
SELECT * FROM AdventureWorks.Sales.SalesOrderHeader
WHERE $partition.annual_range(OrderDate) = 4  -- 2004년 데이터

/*
```

```
  각 분할 영역 별로 그룹핑하고 집계 작업 수행 (ex. 분할 영역 내의 행수 집계)
*/
SELECT count(*) cnt, $partition.annual_range(OrderDate) ptn
FROM  AdventureWorks.Sales.SalesOrderHeader
GROUP BY $partition.annual_range(OrderDate)
ORDER BY ptn
```

분할 구성표 (Partition Scheme)

분할 함수에 의해서 테이블 및 인덱스의 분할 영역에 대한 정의가 완료되었습니다. 이제 각 분할 영역을 어떤 저장소 위치에 배치할 것인지 결정을 해야 합니다. SQL Server에서 테이블 및 인덱스에 대해서 저장소 배치를 결정짓기 위해 제공되는 것이 바로 파일그룹(FileGroup)입니다. 분할 구성표는 분할 영역을 어떤 파일 그룹에 저장할 것인지에 대한 매핑 정보로 구성됩니다.

[따라하기] 분할 구성표 만들기

```
USE Sample
GO
CREATE PARTITION SCHEME annual_scheme_1
AS PARTITION annual_range TO (
annual_2001  -- 2002년 이전 분할 영역을 위한 파일 그룹
, annual_2002  -- 2002년 분할 영역을 위한 파일 그룹
, annual_2003  -- 2003년 분할 영역을 위한 파일 그룹
, annual_2004  -- 2004년과 그 이후 분할 영역을 위한 파일 그룹
)
```

■ 중요 인수에 대한 설명

- Partition_function_name

  분할 구성표가 사용할 분할 함수의 이름입니다. 반드시 미리 존재해야 합니다.

- File_group_name | [PRIMARY] [,…n]

Partition_function_name 으로 지정한 분할을 저장할 파일 그룹의 이름을 지정합니다. 파일 그룹은 데이터베이스 내의 미리 존재해야 합니다.

[PRIMARY]를 지정하면 주 파일 그룹에 저장됩니다. 구문에서 ALL을 지정하면 하나의 파일 그룹에 모두 저장하도록 지정할 수도 있습니다. [,…n] 에 나열된 파일 그룹의 순서대로 1번 분할부터 할당됩니다. 물론 [,…n] 에서 같은 파일 그룹을 두 번 이상 지정할 수 있습니다. 만일 분할 함수보다 많은 파일 그룹이 지정되면 첫 번째 파일 그룹이 NEXT USED로 표시되며 나중에 ALTER PARTITION FUNCTION 문에서 새로운 분할을 추가하는 경우 NEXT USED 파일 그룹이 추가 분할을 저장하도록 구성할 수 있습니다. 물론 이 작업은 ALTER PARTITION SCHEME 문으로 기본적으로 지원됩니다.

만일 단일 그룹에 모든 분할을 저장하고자 하는 경우, ALL 인수를 사용할 수 있습니다.

[따라하기] 단일 그룹의 분할 구성표 만들기

```
-- 모든 분할 영역을 [PRIMARY] 그룹에 할당
CREATE PARTITION SCHEME annual_scheme_2
AS PARTITION annual_range ALL TO ( [PRIMARY] )
```

분할된 테이블 및 인덱스 (Partitioned Table and Indexes)

이제 분할 함수와 분할 구성표를 기준으로 분할 테이블 및 인덱스를 생성합니다. 작업 방법에 있어서 다음과 같은 특징들을 살펴볼 수 있습니다.

■ 분할된 테이블 및 인덱스 구성

- 분할 키로 하나의 칼럼을 지정할 수 있습니다.
- 분할된 테이블 및 인덱스는 파일 그룹이 아니라 "분할 구성표(Partition Scheme)" 기준으로 생성됩니다.
- 테이블 및 인덱스에 대한 모든 쿼리 작업은 분할에 대해서 투명하게 동작합니다.
- 서로 다른 테이블 및 인덱스가 같은 분할 함수 및 구성표를 공유할 수 있습니다.
- 결과적으로, 분할 함수(1) = 분할 구성표(n) = 테이블 혹은 인덱스(n)의 관계를 가질 수 있음을 의미합니다.

[따라하기] 분할된 테이블 만들기

```
/*
  준비 작업. 필요한 파일 그룹의 생성
 주의. 간단한 연습을 위해서 파일 크기 지정 등은 생략했습니다. 실제로는 반드시
    일정 크기를 지정하시기를 권장합니다.
*/
-- 1. 2002년 이전 파일그룹
ALTER DATABASE Sample
  ADD FILEGROUP annual_2001
GO
ALTER DATABASE Sample
  ADD FILE (
    NAME=annual_2001
  , FILENAME=' C:\Temp\annual_2001.ndf'
  ) TO FILEGROUP annual_2001
GO
-- 2. 2002년 파일그룹
ALTER DATABASE Sample
  ADD FILEGROUP annual_2002
GO
```

```
ALTER DATABASE Sample
   ADD FILE (
      NAME=annual_2002
   , FILENAME=' C:\Temp\annual_2002.ndf'
   ) TO FILEGROUP annual_2002
GO
-- 3-6. 위와 같은 방법으로 2006년 파일그룹까지 파일 추가 작업 반복 수행


/*
  분할된 테이블 생성
*/
-- annual_scheme_1 구성표를 기준으로 테이블 생성
CREATE TABLE SalesOrderHeaderHistory (
   SalesOrderID int NOT NULL
, RevisionNumber tinyint NOT NULL
, OrderDate datetime NOT NULL
, DueDate datetime NOT NULL
) ON annual_scheme_1 (OrderDate)

-- 아래는 분할된 테이블에 데이터 로드
INSERT INTO SalesOrderHeaderHistory
SELECT SalesOrderID, RevisionNumber, OrderDate, DueDate
FROM AdventureWorks.Sales.SalesOrderHeader
WHERE OrderDate >= '20010101'   -- 2001년 이전 데이터는 제외
```

분할된 인덱스(Partitioned Index)

테이블 데이터를 분할할 수 있을 뿐만 아니라 인덱스 또한 분할할 수 있습니다. 분할된 인덱스는 테이블과 독립적으로 구현할 수도 있지만 일반적으로 분할된 테이블을 구현한 다음 이 테이블에 인덱스를 만드는 것이 일반적입니다. 이미 분할된 테이블에 인덱스를 만드는 경우 명시적으로 다르게 분할하지 않는 한 SQL Server가 자동으로 인덱스를 테이블과 동일한 구성표와 분할 열을 사용해서 자동으로 분할 시킵니다. 그 결과 인덱스는 테이블과 동일한 방식으로 분할되게 되면, 이를 정렬된 인덱스(Aligned Index) 라고 합니다.

테이블 및 인덱스가 정렬되면 모든 관련 데이터 및 인덱스가 같은 알고리즘으로 나누어지므로 SQL Server에서 보다 빠르고 효율적으로 분할 영역 추가 및 전환이 가능합니다.

■ 인덱스 생성 시 분할 열의 처리 방법
- 고유한 인덱스를 분할하는 경우 인덱스 키에 분할 열이 포함되어야 합니다.
- 고유하지 않은 인덱스를 분할하는 경우, 인덱스 키 목록에 분할 열이 명시적으로 포함되지 않은 경우 SQL Server가 분할 열을 인덱스 키 목록에 추가합니다. 그러나 일반적인 인덱스 키와는 다른 속성으로 추가가 됩니다.

[따라하기] 분할된 인덱스 만들기

```
-- 테이블과 동일한 분할 구성표에 기반해서 인덱스 생성
CREATE UNIQUE CLUSTERED INDEX CL_SOH_SalesOrderID
ON SalesOrderHeaderHistory (OrderDate, SalesOrderID)
```

[참고]
데이터베이스 엔진 튜닝 관리자(Database Engine Tuning Advisor)의 튜닝 옵션(Tuning Option) 탭에서도 분할 전략(Partitioning Strategy) 항목을 통해서 권장 인덱스에 대해 정렬된 분할 여부에 대한 지정 및 기존 인덱스 조정이 가능합니다.

반드시 정렬된 인덱스를 사용해야 하는 것은 아닙니다. 다음의 경우에 분할된 인덱스를 테이블과 독립적으로 분할되는 것이 유용할 수 있습니다.

■ 비 정렬 인덱스가 유용할 수 있는 경우
- 테이블이 분할되지 않은 경우
- 인덱스 키가 고유(UNIQUE)하고 테이블의 분할 열을 포함하지 않는 경우
- 테이블이 다른 조인 열을 사용해서 또 다른 테이블과 콜러케이션된 조인에 참여하도록 유도하는 경우

[따라하기] 비 정렬 인덱스 만들기

```
-- 테이블 분할 구성표와 별도로 특정 파일 그룹에 저장
CREATE UNIQUE NONCLUSTERED INDEX NC_SOH_SalesOrderID
ON SalesOrderHeaderHistory (SalesOrderID)
ON [PRIMARY]
```

[참고]
분할 전환을 사용하려면 테이블의 모든 인덱스가 정렬되어야 합니다.

주의할 것은 인덱스 키로 사용되는 열과 분할 열에 대한 관계입니다. 인덱스 키 열이 기본적으로 분할 열을 포함하고 있다고 전제할 수 없으므로, 인덱스 키 열 지정 시 분할 열의 포함 여부가 중요한 설계 요소가 됩니다.

보다 상세한 설명은 온라인 설명서의 "분할 인덱스에 대한 특수 지침(Special Guidelines for Partitioned Indexes)"을 참고하시기 바랍니다.

카탈로그 뷰를 사용한 메타데이터 검색

분할된 테이블 및 인덱스와 그 관련된 분할 정보(분할 구성표, 함수, 경계 값 등)에 대한 메타 데이터 검색을 위해서 여러 가지 카탈로그 뷰들이 제공됩니다.

아래 각 카탈로그 뷰에 대해서 검색 작업을 수행해 보시고, 각 칼럼 등에 대한 상세 정보들은 온라인 설명서 등을 통해서 확인해 보시기 바랍니다.

■ 분할된 테이블 또는 인덱스에 대한 분할 정보
- sys.tables
- sys.indexes
- sys.index_columns

■ 개별 분할 구성표에 대한 정보
- sys.destination_data_spaces

■ 데이터베이스 내의 모든 분할에 대한 정보
- sys.partitions

■ 데이터베이스 내의 모든 분할 구성표에 대한 정보
- sys.partition_schemes
- sys.data_spaces

■ 개별 분할 함수에 대한 정보
- sys.partition_functions

■ 분할 함수의 개별 매개 변수에 대한 정보
- sys.partition_parameters

■ 분할 함수의 경계 값에 대한 정보
- sys.partition_range_values

분할된 테이블에 대한 쿼리

분할된 테이블의 설계 목표에 따라서 최적의 실행 계획 산출을 위한 인덱스와 검색 조건 식 등이 중요한 성능 문제가 됩니다. 대용량을 처리하는 쿼리와 작지만 많이 수행되는 단순 쿼리의 성능을 모두 만족하기 위해서 검색 조건 식에 분할 키를 포함해야 합니다.

[따라하기] 분할 키를 포함한 단순 쿼리와 실행 계획 출력

1. 쿼리 실행

```
-- 텍스트 형식의 실행 계획 출력을 위한 옵션 설정
SET STATISTICS PROFILE ON

-- 두 개의 분할을 사용하는 쿼리
SELECT *
FROM dbo.SalesOrderHeaderHistory
WHERE OrderDate IN ( '2003-11-01' , '2004-02-01' )

SET STATISTICS PROFILE OFF
```

분할 처리를 포함하는 쿼리의 실행 계획을 보면, 요구하는 분할만을 처리하는 것을 확인할 수 있습니다. 두 개 이상의 분할된 테이블 혹은 인덱스를 요구하는 경우 해당 실행 계획에서 "상수 스캔"(CONSTANT SCAN) 연산자를 볼 수 있습니다. 이 연산자는 WHERE 절에 기반해서 요구되는 파티션 번호(ID)를 결정하며, 해당 파티션 번호를 외부 입력으로 참조해서 분할 테이블을 검색하게 됩니다.

2. 실행 계획 출력 결과 (... 는 생략된 부분)

```
-- 실행 계획의 결과를 보면 분할 ID: 3, 4 에서 검색이 이루어지는 것을 확인할 수 있다.
SELECT *  FROM dbo.SalesOrderHeaderHistory  ...
  |--Nested Loops(..., OUTER REFERENCES:([PtnIds1003]) PARTITION
ID:([PtnIds1003]))
      |--Constant Scan(VALUES:(((3)),((4))))
      |--Clustered Index ...
```

대용량 데이터를 조인하는 쿼리에서도 분할 기능이 적용될 수 있습니다. 일반적으로 부모/자식 관계를 가진 관련 테이블 (예: SalesOrderHeader 와 SalesOrderDetails)을 동일한 분할 키와 분할 함수를 사용해서 분할되는 경우 두 테이블이 콜러케이션되었다(Collocated) 고 합니다. 최적화 프로그램은 콜러케이션된 테이블이 조인되는 경우 같은 분할에 있는 데이터를 먼저 조인한 다음 결과들을 결합할 수 있습니다. 이는 다중 CPU를 보다 효율적으로 사용할 수 있도록 합니다.

분할 유지관리

일반적으로 분할에 대한 유지관리에는 분할에 대량 데이터를 입력하거나 혹은 분할을 제거하는 등의 작업이 포함됩니다. 이러한 작업들은 분할 함수를 가지고 분할 경계 값을 기준으로 분할 나누기(Split)를 하거나 병합(Merge)하는 방법을 통해서 수행될 수 있습니다. 또한 분할된 테이블과 일반 테이블 간의 분할 전환(Switch)을 통해서 대량의 데이터 집합을 보다 신속하고 효율적으로 전송할 수 있습니다.

[참고]
분할 유지관리에 나오는 예제는 참고만하시고, 실제 따라 하기는 이후에 나오는 전환 (Switch) 부분의 따라 하기나 슬라이딩 윈도우 부분의 따라 하기를 이용하십시오.

■ 분할 나누기(Splitting)

- 분할 함수를 수정함으로써 분할 경계 값을 새로 추가하고 그 결과 새로운 경계를 포함한 분할을 만들 수 있습니다. 이 때 해당 분할 함수를 사용하는 관련 테이블 및 인덱스에 모두 영향을 줍니다.
- 파일 그룹은 온라인 상태이어야 하며 새 분할을 보유하기 위해 분할 함수를 사용하는 해당 분할 구성표(Scheme)를 수정해서 해당 파일 그룹이 NEXT USED로 표시되어야 합니다. (혹은 분할 구성표 생성 시, 추가 파일 그룹 지정으로 NEXT USED로 표시될 수도 있습니다)

[예제] 분할 나누기

```
-- 새로운 파일 그룹을 새 분할 포함 영역으로 지정
ALTER PARTITION SCHEME annual_scheme_1
NEXT USED annual_2005
Go

-- 새로운 분할 경계 지정
ALTER PARTITION FUNCTION annual_range( )
SPLIT RANGE ( '2005-01-01' )
```

[참고]
분할 나누기로 인해 발생할 수 있는 데이터의 이동 및 분할 데이터 구조의 변경 등은 경우의 따라 시스템 성능의 많은 영향을 미칠 수 있습니다.

■ 분할 병합하기(Merging)

- 분할 함수를 수정함으로써 기존 분할 경계를 제거하고 그 결과 해당 경계를 기준으로 두 분할을 병합할 수 있습니다. 병합의 기준은 분할 함수의 [LEFT]RIGHT]에 따라서 결정됩니다. 해당 분할 함수를 사용하는 관련 테이블 및 인덱스에 모두 영향을 줍니다.
- 병합의 결과 사용되지 않은 파일 그룹은 분할 스키마에서 제거되거나 NEXT USED 속성으로 표시됩니다.

[예제] 분할 병합하기

```
-- 2002년과 2003년 분할을 병합
ALTER PARTITION FUNCTION annual_range( )
MERGE RANGE ( '2002-01-01' )
go

-- 병합된 범위 확인
SELECT * FROM sys.partition_range_values
```

[참고]
분할 나누기에서 언급한 것처럼 동일한 성능 문제가 유발될 수 있습니다.

■ 분할 전환하기(Switching)

- ALTER TABLE 구문의 SWITCH를 사용해서 한 테이블 혹은 특정 분할을 다른 테이블의 빈 분할 혹은 빈 테이블로 빠르게 전환할 수 있습니다. 이는 메타 데이터의 기반한 작업이므로 실제 데이터 이동이 일어나지 않으며, 아주 빠른 데이터 이동 작업이 가능합니다.
- 원본과 대상은 동일한 파일 그룹 내에 존재해야 하며, 원본은 대상에서 요구하는 모든 인덱스를 포함해야 하며, 필요한 경우 정렬된 인덱스이며 파일 그룹이 일치해야 합니다.
- 대상이 분할 영역인 경우, 원본은 CHECK 제약 조건을 가지거나 대상 범위와 동일한 분한 영역을 가져야 합니다.
- 관련된 모든 인덱스들도 함께 자동 전환됩니다.

아래 따라 하기에서 몇 가지 예제를 소개합니다. 분할 대 분할 간의 전환은 슬라이딩 윈도우에서 소개합니다.

[따라하기] 분할에서 빈 테이블로 전환하기

```
-- 전환될 테이블 (일명, Staging 테이블)
CREATE TABLE dbo.SalesOrderHeaderHistory_2001(
  SalesOrderID int NOT NULL
, RevisionNumber tinyint NOT NULL
, OrderDate datetime NOT NULL
, DueDate datetime NOT NULL
, CustomerID int NOT NULL
) ON annual_2001
GO
-- 동일한 인덱스 생성
CREATE UNIQUE CLUSTERED INDEX SOH_2001_SalesOrderID
ON dbo.SalesOrderHeaderHistory_2001(OrderDate, SalesOrderID)
ON annual_2001
GO

-- ALTER TABLE ... SWITCH 를 이용, 특정 분할을 테이블로 전환
ALTER TABLE dbo.SalesOrderHeaderHistory
SWITCH PARTITION $partition.annual_range( '2001-01-01' )
TO SalesOrderHeaderHistory_2001
```

[따라하기] 테이블에서 빈 분할로 전환하기

```
-- 분할로 전환하기 위해서는 원본 테이블의 CHECK 제약 조건이 필요
ALTER TABLE dbo.SalesOrderHeaderHistory_2001
WITH CHECK
  ADD CONSTRAINT CK_SalesOrderHeaderHistory_2001
    CHECK (OrderDate >= '2001-01-01' AND OrderDate < '2002-01-01' )
go
```

```
-- 전환
ALTER TABLE dbo.SalesOrderHeaderHistory_2001
SWITCH TO
dbo.SalesOrderHeaderHistory PARTITION $partition.annual_range( '2001-01-01' )
```

슬라이딩 윈도우(Sliding Window)

슬라이딩 윈도우는 대용량 데이터베이스 관리를 위해 필요한 시나리오입니다. 매 시간, 매일, 매주 그리고 매월 단위로 새로운 데이터를 추가하기 위해 새 분할을 추가하고 오래된 이전 분할은 제거를 하는 것입니다.

■ 새로운 분할은
- 테이블에 통합되기 전에 배치 로드가 되거나, 정리 혹은 변환하는 등의 작업이 필요할 수 있습니다.
- 또는 빈 분할로 시작한 뒤 트랜잭션을 사용해서 점진적으로 채워지거나 대량 로드로 증가될 수 있습니다.

■ 오래된 분할은
- 데이터 보관을 위해 백업하거나
- 압축하거나 혹은 이후에 복원 등에 사용될 수 있습니다.

추가할 새 데이터에 대한 중간 처리 과정(로드, 정리 혹은 변환)이 필요한 경우 이를 별도의 테이블(Staging 테이블)로 처리할 수도 있습니다. 마찬가지로 오래된 이전 분할의 데이터 또한 별도의 테이블로 옮기거나 또 다른 분할된 테이블의 빈 분할로 전환할 수 있으며, 데이터 웨어하우스에 저장할 수도 있습니다. INSERT INTO ... SELECT 문과 달리 물리적인 데이터 이동이 필요치 않으므로, 데이터 집합의 크기에 관계없이 빠르고 효율적으로 전송이 가능합니다.

다음은, 분할된 테이블과 또 다른 분할된 테이블 간의 데이터 전환을 통한 슬라이딩 윈도우 처리 과정을 간략히 소개합니다.

■ 따라 하기의 슬라이딩 윈도우 처리 과정 (테이블은 모두 분할된 테이블입니다)
- 과거 분할을 저장하기 위한 새로운 분할된 테이블 구조(함수, 구성표, 테이블 등)를 생성 (따라 하기는 2001년 데이터를 전환할 것입니다)
- ALTER PARTITION SCHEMA ... NEXT USED 구문을 사용해서 원본 테이블에 새로운 분할을 받을 파일 그룹을 지정합니다. (2005년 데이터를 위한 파일 그룹을 지정합니다)
- ALTER PARTITION FUNCTION ... SPLIT 구문을 사용해서 새로운 분할 영역을 추가합니다. (2005년 데이터를 위한 함수 영역을 정의합니다)
- 목적지 테이블에도 새로운 분할을 받을 파일 그룹을 지정합니다. (2001년 데이터)
- 목적지 분할 함수에도 새로운 분할 영역을 정의합니다. (2002년 함수 영역)
- 원본 테이블에서 전환될 분할 영역인 2001년에 대한 CHECK 제약 조건을 추가합니다.
- ALTER TABLE ... SWITCH 구문을 사용해서 첫 번째 분할을 목적지 두 번째 분할로 이동합니다.
- 원본 테이블에서 2001, 2002년 분할을 병합합니다.
- 대상 테이블에서 2001, 2002년 분할을 병합합니다.
- 원본 테이블에 정의한 CHECK 제약 조건을 제거합니다.
- 데이터를 확인합니다.

[따라하기] 슬라이딩 윈도우

```
-- 1. 과거 분할을 위한 새로운 분할 테이블 구성 작업
CREATE PARTITION FUNCTION annual_range_ar (DATETIME)
AS RANGE RIGHT
FOR VALUES
(
  ' 2001-01-01'
)
GO
CREATE PARTITION SCHEME annual_scheme_1_ar
```

```
AS PARTITION annual_range_ar TO
(
   annual_2001
, annual_2001
)
GO
CREATE TABLE dbo.SalesOrderHeaderHistory_ar(
   SalesOrderID int NOT NULL
,  RevisionNumber tinyint NOT NULL
,  OrderDate datetime NOT NULL
,  DueDate datetime NOT NULL
,  CustomerID int NOT NULL
) ON annual_scheme_1_ar(OrderDate)
GO
CREATE UNIQUE CLUSTERED INDEX CL_SOH_ar_SalesOrderID
ON dbo.SalesOrderHeaderHistory_ar(OrderDate, SalesOrderID)
GO
-- 2. 새로운 분할용 파일 그룹
ALTER PARTITION SCHEME annual_scheme_1
NEXT USED annual_2005;
GO
-- 3. 새로운 분할 영역 추가
ALTER PARTITION FUNCTION annual_range( )
SPLIT RANGE ( '20050101' );
GO
-- 4. 목적지 추가 분할 영역을 위한 파일 그룹
ALTER PARTITION SCHEME annual_scheme_1_ar
NEXT USED annual_2002;
GO
```

```
-- 5. 목적지 추가 분할 영역 정의
ALTER PARTITION FUNCTION annual_range_ar( )
SPLIT RANGE ( '20020101' );
GO
-- 6. CHECK 제약 조건 추가
ALTER TABLE SalesOrderHeaderHistory
  ADD CONSTRAINT CK_SalesOrderHeaderHistory_DateRange
      CHECK (OrderDate >= '2001-01-01 00:00:00.000' );
GO
-- 7. 분할 간의 전환
ALTER TABLE SalesOrderHeaderHistory
SWITCH PARTITION 1
TO SalesOrderHeaderHistory_ar PARTITION 2;
GO
-- 8. 원본에서 2001년 분할과 병합
ALTER PARTITION FUNCTION annual_range( )
MERGE RANGE ( '20020101' );
GO
-- 9. 대상에서 2001년 분할과 병합
ALTER PARTITION FUNCTION annual_range_ar( )
MERGE RANGE ( '20010101' );
GO
-- 10. CHECK 제약 조건 제거
ALTER TABLE SalesOrderHeaderHistory
    DROP CONSTRAINT CK_SalesOrderHeaderHistory_DateRange
GO
-- 11. 데이터확인, 대상 테이블에서도 동일한 쿼리로 확인 가능
SELECT $partition.annual_range_ar(OrderDate) AS 분할번호
  , MIN(OrderDate) AS 시작일, MAX(OrderDate) AS 종료일
```

```
  , COUNT(*) AS 행수
FROM dbo.SalesOrderHeaderHistory_ar
GROUP BY $partition.annual_range_ar(OrderDate)
ORDER BY 분할번호
```

백업과 복원 및 유지 관리

분할에 대한 백업 및 복원 등의 유지 관리에 있어서 SQL Server 2005에 향상된 기능을 활용한 이득을 얻을 수 있습니다. 특히, 읽기-전용 파일 그룹과 증분 백업 및 복원 기능은 아주 유용하게 적용될 수 있습니다.

■ SQL Server 2005 향상된 처리를 활용하면

- 읽기-전용의 파일 그룹은 트랜잭션 로그의 적용 없이 복원이 가능합니다.
- 이력 데이터가 변경될 필요가 없다면 정규 백업 작업의 데이터 볼륨을 줄일 수 있습니다.
- 읽기-전용 파일 그룹은 한 번만 수행할 수 있습니다.
- 정규 백업은 기본 파일 그룹과 다른 활성 파일 그룹만 수행합니다.
- 복원 시, 기본 파일 그룹과 활성 파일 그룹을 복원하고 로그를 복구합니다. 그리고 나서 읽기-전용 파일 그룹 들을 개별적으로 복원합니다. (증분 복원)

■ 분할 인덱스 유지 관리 작업

- 단일 분할 영역에 대한 인덱스 다시 작성(Rebuild)를 위해서 ALTER INDEX 구문을 사용합니다.
  ALTER INDEX ... REBUILD PARTITION = partition number
- 단일 분할 영역에 대한 인덱스 다시 구성(Reorganize)를 위해서 ALTER INDEX 구문을 사용합니다.
  ALTER INDEX ··· REORGANIZE PARTITION = partition number

■ 참고 사항

- 이력 데이터에 대해서는 인덱스 다시 작성이나 구성 작업이 필요하지 않을 것입니다.
- 대용량의 분할된 테이블 및 인덱스에서 서로 다른 분할에 대해 서로 다른 일정으로 인덱스 작업이 수행될 수 있을 것입니다.

[참고]
SQL Server 2005의 온라인 인덱스 다시 구성은 개별 분할에 대해 수행되지 않습니다.

■ 권장 사례

- 쿼리 성능을 위한 적절한 분할 키와 고유(UNIQUE) 인덱스를 설계해야 합니다.
- 분할 키와 중요한 쿼리의 WHERE 절에서 검색을 제한할 수 있어야 합니다.
- 정렬된 인덱스를 사용합니다.
- 분할 나누기 및 병합 시에는 최적 성능 위해서 반드시 빈 분할에서 수행합니다.
- 가능한 한 많은 디스크에 데이터를 분포시킵니다.
- 다음 작업을 포함한 분할 관련 모든 작업을 테스트 합니다.
  〉 증분(Piecemeal) 백업과 복원
  〉 유지 관리 계획들
  〉 나누기 / 병합 / 전환 작업들

복제(Replication) 향상 기능

SQL Server의 복제 기술은 OLTP 시스템과 작업부하(Workload)를 분리시켜서 데이터의 분석 및 통계 작업 그리고 보고서 생성 작업을 수행하기 위한 시스템, 데이터 웨어하우스(DW)의 스테이징(Staging) 데이터베이스 용으로 사용될 수 있습니다. 또한 웹사이트의 최대 동시 사용자를 지원하기 위한 데이터베이스 수평 확장을 고려하거나 모바일 사용자 데이터 간의 동기화에 사용될 수 있으며, 심지어 이기종의 데이터 통합을 위해서도 좋은 시나리오를 제공할 수 있습니다.

SQL Server 2005의 복제(Replication) 기술은 또한 놀라울 정도로 많은 신규 기능들과 향상 기능을 포함하고 있습니다. 그 중에서도 특히 엄청나게 간편해진 복제 구성 마법사, 새로운 피어 투 피어(Peer to Peer) 복제 기술, HTTPS 통한 웹 동기화, 진단 분석이 가능한 복제 모니터, Oracle 에서의 트랜잭션 복제 기능 등은 기존 복제 사용자들을 충분히 만족시킬 수 있는 멋진 기능 들입니다.

이제 SQL Server 2005에서 어떤 새로운 기술들이 추가되고 향상되었는지 개략적으로 살펴보고, 그 중에서도 의미 있는 중요 기능들을 몇 가지 소개하도록 하겠습니다.

새로운 기능

SQL Server 2005에서 많은 추가 향상 기능들이 도입되고 적용되었습니다. 우선, 표 형태로 각각의 영역별 향상 기능들을 정리해 보겠습니다.

다음은 병합 복제(Merge Replication) 관련된 추가 기능 및 향상 기능 목록입니다.

[표] 병합 복제 신규 기능

미리 계산된 파티션	병렬 업로드 및 다운로드
다양한 아티클 형식-다운로드 전용 등	BLOB 스트리밍
논리적 레코드	에이전트 신뢰성
웹 동기화	재 시작 가능한 스냅숏 전송
DDL 지원	새로운 데이터 형식 지원(.NET, XML, varchar(MAX), 등)
구독자 초기화를 통한 동적 스냅숏	향상된 ID 범위 관리
선언적 아티클 정렬	

다음 트랜잭션 복제에 있어서 추가 기능 목록입니다.

[표] 트랜잭션 복제 신규 기능

피어 투 피어 복제	Oracle 게시
DDL 지원	추적 프로그램 토큰(Trace Token)
배포 에이전트에 대한 병렬 스트림	업데이트 가능 구독에 대한 재 게시
업데이트 구독에 대한 Blob 지원	백업으로부터 구독하기
재 시작 가능한 스냅숏 전송	새로운 데이터 형식 지원(.NET, XML, varchar(MAX), 등)
전체 텍스트를 포함한 새로운 인덱스 형식 지원	

그 밖에 프로그래밍과 사용자 인터페이스에 관련된 새로운 기능들은 다음과 같이 정리할 수 있습니다.

■ 프로그래밍 능력
■ RMO (Replication Management Objects)
.NET 관리 코드(Managed Code)의 기반의 프로그래밍 가능한 복제 관리 개체
■ 사용자 인터페이스
■ 단순해진 복제 구성 마법사
■ 복제 모니터
- 상세한 아티클 통계 정보 제공
- 복제 시스템의 전반적인 대한 모니터링 제공

목록을 통해서 알 수 있는 것처럼, 복제에 관련된 추가 향상 기능들은 아주 많이 있습니다만. 그 중에서도 중요한 몇 가지 핵심 기술에 대해서 좀 더 상세하게 살펴보도록 하겠습니다.

단순하고 편리해진 복제 구성 마법사

SQL Server 2000 버전에서 복제 마법사를 통한 복제 구성은 마법사 그 자체의 뉘앙스처럼 분명 쉽고 편리하다고 말할 수는 없었습니다. 게시 구성만으로도 상당한 페이지와 설정 작업을 요구했으며, 구독 작업 또한 만만치가 않았습니다. 물론 초기 페이지에서 고급 옵션을 설정하지 않으면 그나마 조금 더 단순해지지만 끌어오기 구독을 허용하기 위한 무인 구독 허용을 위해서라도 필요하게 되었습니다. 또한 각각의 페이지가 많은 설정 값과 옵션들로 구성되어 있어, 단순한 복제 구성 만으로도 쉽지가 않은 작업이었습니다. (그래서 전문가들은 역시 마법사보다 스크립트를 선호하게 됩니다)

SQL Server 2005에서는 복제 구성 마법사 놀라운 정도로 쉽고 단순해 집니다. 전체 마법사 페이지가 이전보다 무려 40%나 줄어 들었으며, 각각의 화면들이 아주 단순하게 바뀌었습니다. 또한 설정 값의 자동화나 기본 값 처리가 보다 향상되었으며 분기 작업들 또한 이전보다 훨씬 줄어든 것을 알 수 있습니다.

구독 마법사 또한 이전의 밀어넣기 구독과 끌어오기 구독을 별도로 구성한 반면, 새로운 구독 마법사는 하나의 구독 마법사로 통합이 되어 보다 편리하게 작업할 수 있습니다. (단, 두 구독이 모두 필요할 때는 두 번의 실행을 해야 할 것입니다). 또한 하나의 게시에 대해 서로

다른 속성을 가진 다중 구독을 생성하는 것도 가능하며, 이제 게시와 구독의 결과를 스크립트로 바로 생성하고 재 사용할 수 있도록 보다 편리한 메뉴가 제공됩니다.

자, 그럼 실제로 복제 구성 마법사를 사용해서 복제 게시와 구성 작업이 얼마나 단순하고 편리해졌는지 경험해 보도록 하시죠.

복제 구성 따라하기

[참고]
복제를 구성하기 위해서는 두 가지 옵션을 준비해야 합니다. 첫 번째는 SQL Server Agent 서비스가 반드시 시작되어 있어야 합니다. 그리고 두 번째는 Agent 관련 확장 저장 프로시저의 사용을 활성화 하는 것입니다. 다음은 관련된 소개입니다.

SQL Server 2005는 서버 구성 속성의 개수가 이전 보다 훨씬 더 많아진 것을 알 수 있습니다. 추가 항목 중에서는 확장 저장 프로시저에 대한 것이 많이 있습니다. 이는 보안 문제를 예방하기 위한 사전 조치라고 보여집니다. 복제를 위해서도 Agent 서비스에 관련된 확정 저장 프로시저의 활성화가 필요합니다. 관련 속성이 비활성화된 상태일 경우, sp_configure 등을 이용해서 아래와 같이 해당 구성 속성을 활성화 시켜 주어야 합니다.

1) sp_configure 를 사용해서 설정하는 방법

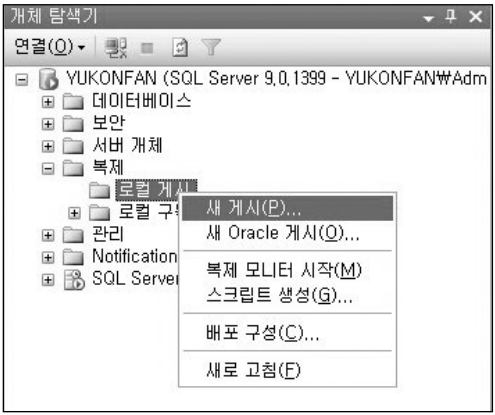
```
-- 속성이 보이지 않을 경우, 고급 옵션을 1로 설정
EXEC sp_configure 'show advanced options', 1
RECONFIGURE

-- Agent XPs 속성을 1로 설정
EXEC sp_configure 'Agent XPs', 1
RECONFIGURE
```

이 속성의 설정 여부는 SQL Server Management Studio의 SQL Server Agent 아이콘을 통해서도 보여집니다.

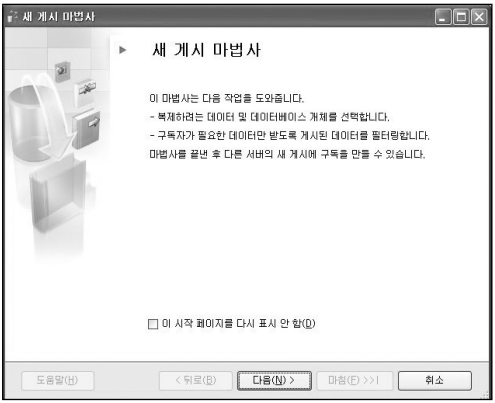
SQL Server Management Studio의 개체 탐색기(Object Explorer) 내에 복제 항목에서 복제에 관련된 전반적인 작업을 수행할 수 있습니다. 아래 따라 하기는 SQL Server 2000에서 예제 데이터베이스용으로 제공되던 Northwind 데이터베이스를 사용해서 트랜잭션 게시와 구독을 구성하고, 그 결과를 확인할 것입니다.

[따라하기] 트랜잭션 게시 만들기

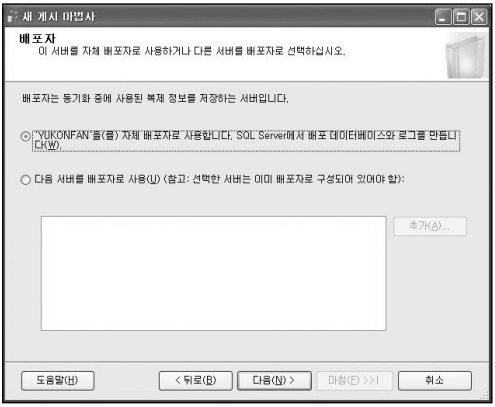
1 개체 탐색기 - 해당 서버 - 복제 - 로컬 게시 - 새 게시 메뉴를 선택합니다.



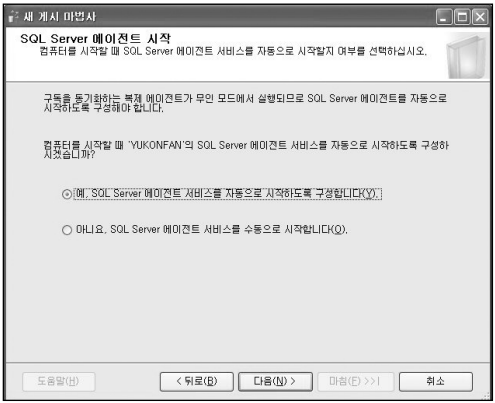
2 새 게시 마법사가 시작됩니다. 첫 페이지는 복제 구성 마법사를 통해서 수행할 중요 작업에 대한 소개입니다.



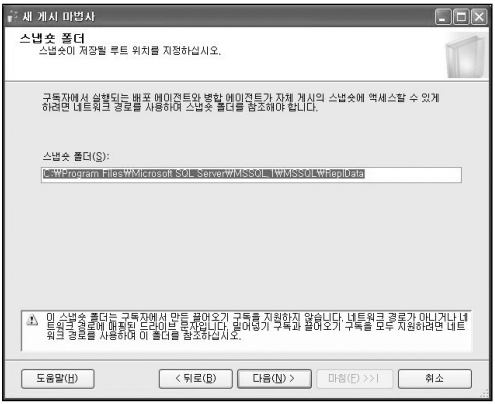
3 배포자 서버를 선택합니다. 현재 서버로 선택하거나 이미 배포자 서버로 구성된 다른 서버 인스턴스를 추가할 수 있습니다. 해당 배포자 서버에는 "배포"라는 데이터베이스가 생성되고, 복제 정보 저장용 데이터베이스로 사용됩니다.



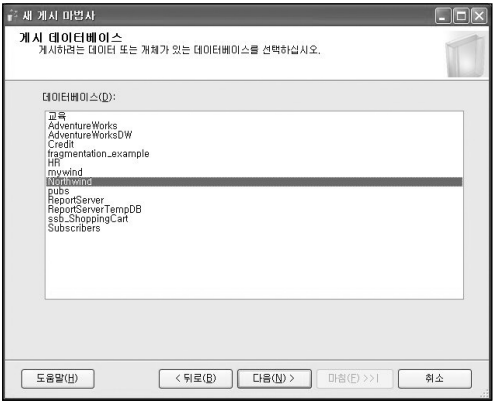
4 SQL Server 에이전트 서비스의 자동 시작 여부를 선택합니다.



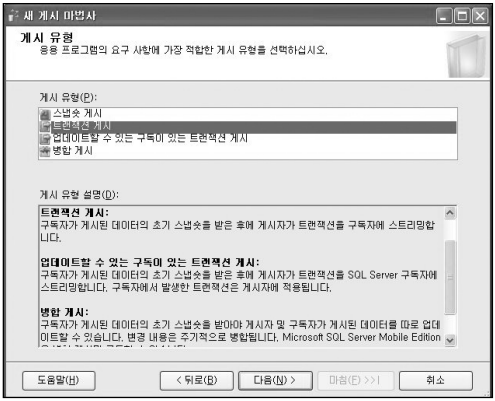
5 복제 스냅숏의 저장 위치를 지정합니다. 기본적으로 SQL Server의 ReplData 폴더가 지정되지만, 구독자에서의 접근을 허용하기 위해서는 네트워크 공유 폴더로서의 경로나 매핑된 드라이브를 지정합니다.



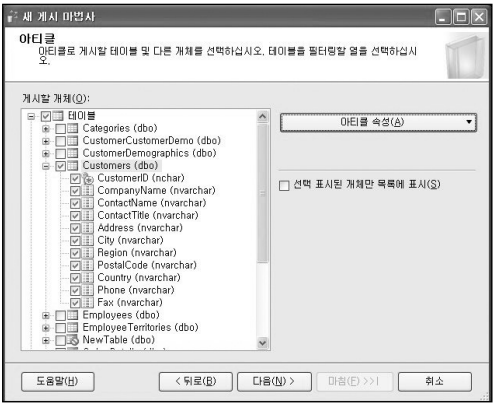
6 게시할 데이터나 개체를 포함하고 있는 원본 데이터베이스를 선택합니다.



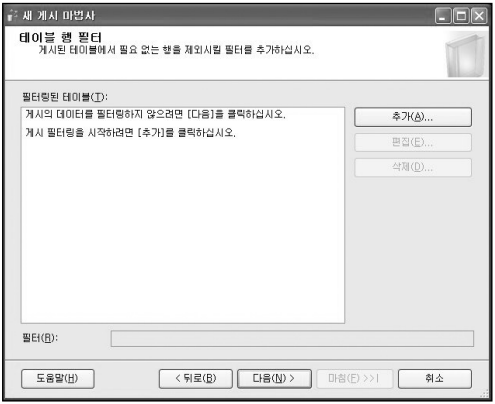
7 게시 유형을 선택합니다. 하단에 게시 유형 설명을 통해서 선택된 게시 형식에 대한 자세한 도움말을 얻을 수 있습니다.



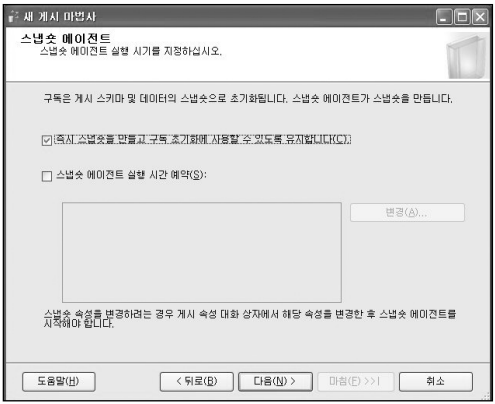
8 게시를 구성하기 위한 하나 이상의 아티클(Article)들을 선택합니다. 테이블의 경우 칼럼 단위 필터링을 위해서 일부 칼럼 만으로 아티클을 지정할 수 있습니다. 오른쪽 아티클 속성 버튼을 통해서 아티클에 대한 게시 관련 다양한 속성 값을 조정할 수 있습니다.



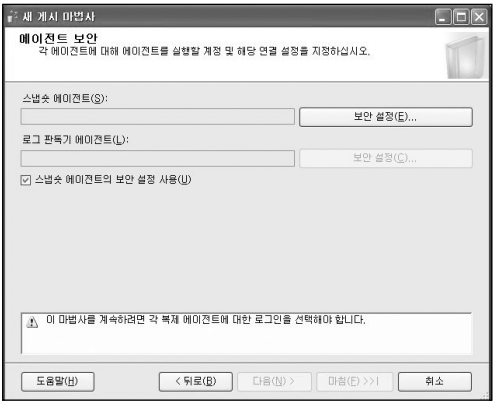
9 게시되는 테이블에서 원하지 않는 행의 복제를 제한하기 위해 행 필터링을 추가하거나 편집 및 삭제할 수 있습니다.



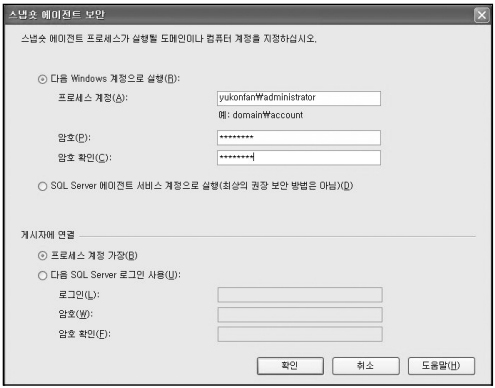
10 구독자에서 현재 게시되는 복제를 구독할 시에, 게시되는 아티클에 대한 데이터 구조(Schema)와 해당 데이터를 스냅숏을 통해 초기화할 수 있습니다. 이를 위해 스냅숏을 생성하고 유지하도록 지정합니다. 또한 해당 스냅숏 에이전트 실행을 위해 일정을 조정하실 수도 있습니다.



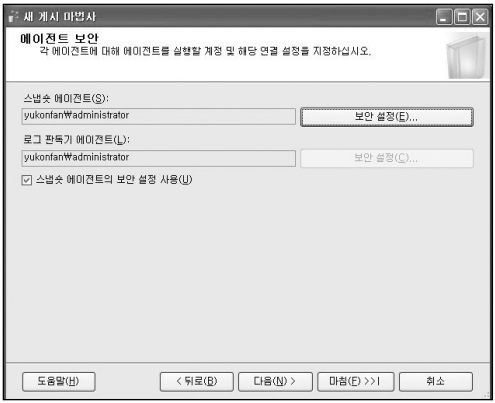
11 복제에 실질적인 수행을 담당하는 각 에이전트들에 대해서(복제 유형에 따른) 실행 권한을 가진 계정을 지정합니다. 하나의 에이전트에 대해 계정을 지정하고 다른 에이전트는 동일한 계정 하에서 실행되도록 지정할 수도 있습니다. 오른쪽 보안 설정... 버튼을 클릭합니다.



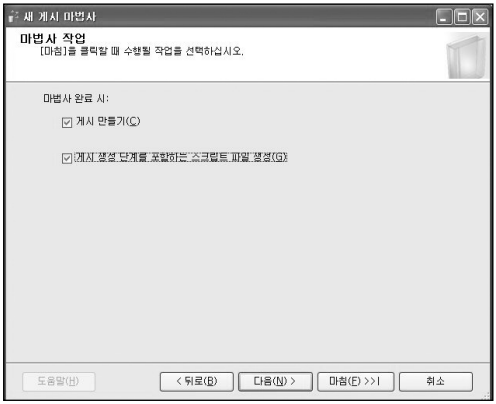
12 에이전트가 어떤 사용자 계정으로 실행될 것인지 지정합니다. 기본적으로 도메인 사용자 계정이나 로컬 컴퓨터 사용자 계정을 지정하며, 해당 사용자는 관리자 수준의 필요 권한을 가져야 합니다. 그리고 게시자에 연결할 때 사용할 계정으로 프로세스 계정을 대리(Impersonate)하거나 SQL Server 표준 사용자 계정을 지정할 수 있습니다.



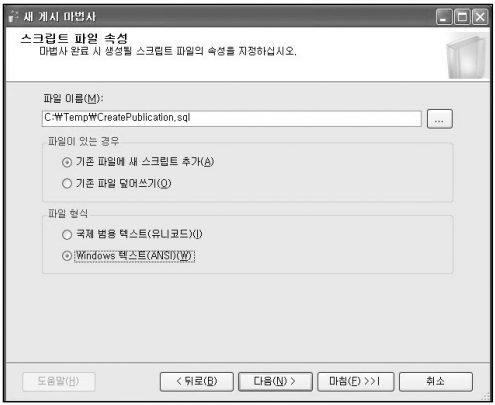
13 스냅숏 에이전트의 실행을 위한 사용자 계정 만을 지정하고 로그 판독기 에이전트는 동일한 계정으로 처리했습니다.



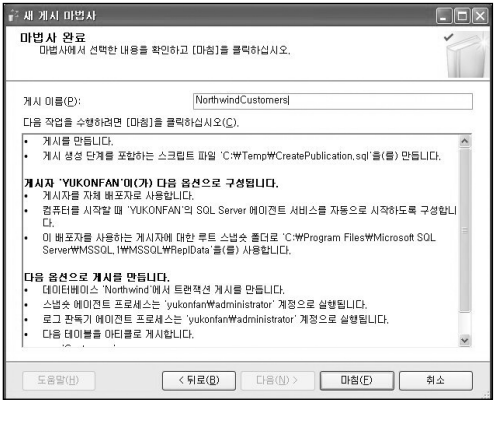
14 마법사의 최종 작업 결과로 게시를 생성하게 되며, 필요할 경우 게시 만들기의 내용을 스크립트로 생성하고 파일로 저장하도록 지정할 수 있습니다. 예제에서는 스크립트를 생성하도록 지정을 했습니다.



15 스크립트 생성 옵션을 지정하는 경우, 다음 화면에서 스크립트를 저장할 파일과 저장 관련 옵션들(덮어쓰기 여부, 파일 형식)을 지정합니다.



16 지금까지 설정한 게시 관련 속성 값들에 대해 정리를 합니다. 원하는 게시인지를 확인한 후 생성될 게시의 이름을 입력합니다.



17 마지막으로 게시 생성 작업을 수행하고 문제가 발생할 경우 관련 메시지를 링크 형태로 보여줍니다. 하단의 보고서 버튼을 통해서 수행 결과에 대한 추가 내용을 확인할 수 있습니다.

이제 복제 - 로컬 게시 항목에서 새로운 게시 [Northwind]:NorthwindCustomers 가 보여집니다.

그리고 마법사에서, 스크립트 생성을 통해 만들어진 스크립트 파일의 내용은 다음과 같습니다.

[예제 게시 만들기 스크립트, CreatePublication.sql]

```
/****** 서버 YUKONFAN의 복제 구성을 스크립팅하고 있습니다. (생략)... ******/
/****** 참고: 보안을 위해 모든 암호 매개 변수는 NULL 또는 (생략)... ******/

/****** 서버 YUKONFAN을(를) 배포자로 설치하고 있습니다. (생략)... ******/
use master
exec sp_adddistributor @distributor = N' YUKONFAN' , @password = N' '
GO
```

```
exec sp_adddistributiondb @database = N' 배포' , @data_folder = N' C:₩Program
Files₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩Data' , @data_file_size = 4,
@log_folder = N' C:₩Program Files₩Microsoft SQL Server₩MSSQL.1₩MSSQL
₩Data' , @log_file_size = 2, @min_distretention = 0, @max_distretention = 72,
@history_retention = 48, @security_mode = 1
GO

use [배포]
if (not exists (select * from sysobjects where name =  'UIProperties'
                        and type =  'U  '))
        create table UIProperties(id int)
if (exists (select * from ::fn_listextendedproperty( 'SnapshotFolder' ,  'user' ,  'dbo' ,
 'table' ,  'UIProperties' , null, null)))
   EXEC sp_updateextendedproperty N' SnapshotFolder' , N' C:₩Program Files
₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩ReplData' ,  'user' , dbo,  'table' ,
 'UIProperties'
else
   EXEC sp_addextendedproperty N' SnapshotFolder' ,  'C:₩Program Files
₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩ReplData' ,  'user' , dbo,  'table' ,
 'UIProperties'
GO

exec sp_adddistpublisher @publisher = N' YUKONFAN' , @distribution_db = N' 배
포' , @security_mode = 1, @working_directory = N' C:₩Program Files₩Microsoft
SQL Server₩MSSQL.1₩MSSQL₩ReplData' , @trusted = N' false' ,
@thirdparty_flag = 0, @publisher_type = N' MSSQLSERVER'
GO

use [Northwind]
exec sp_replicationdboption @dbname = N' Northwind' , @optname = N' publish' ,
```

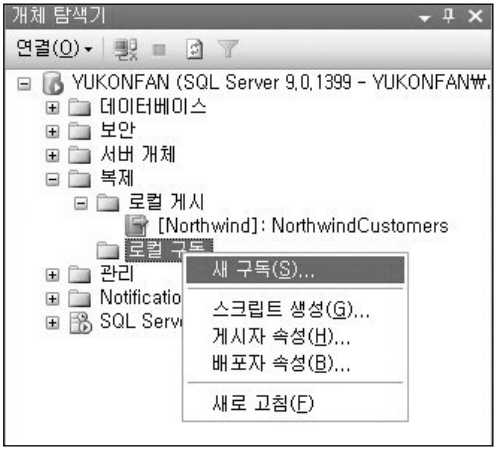
```
@value = N' true'
GO

use [Northwind]
exec [Northwind].sys.sp_addlogreader_agent @job_login = N' yukonfan
₩administrator' , @job_password = null, @publisher_security_mode = 1,
@job_name = null
GO

-- 트랜잭션 게시를 추가하는 중
use [Northwind]
exec sp_addpublication @publication = N' NorthwindCustomers' , @description =
N' 게시자 " YUKONFAN" 의 데이터베이스 " Northwind" 에 대한 트랜잭션 게시입
니다.' , @sync_method = N' concurrent' , @retention = 0, @allow_push = N' true' ,
@allow_pull = N' true' , @allow_anonymous = N' true' , @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' false' , @enabled_for_p2p = N' false' ,
@enabled_for_het_sub = N' false'
GO

exec sp_addpublication_snapshot @publication = N' NorthwindCustomers' ,
@frequency_type = 1, @frequency_interval = 0, @frequency_relative_interval = 0,
@frequency_recurrence_factor = 0, @frequency_subday = 0,
@frequency_subday_interval = 0, @active_start_time_of_day = 0,
@active_end_time_of_day = 235959, @active_start_date = 0, @active_end_date =
```

```
0, @job_login = N' yukonfan₩administrator' , @job_password = null,
@publisher_security_mode = 1

use [Northwind]
exec sp_addarticle @publication = N' NorthwindCustomers' , @article = N'
Customers' , @source_owner = N' dbo' , @source_object = N' Customers' , @type
= N' logbased' , @description = null, @creation_script = null, @pre_creation_cmd =
N' drop' , @schema_option = 0x000000000803509F,
@identityrangemanagementoption = N' manual' , @destination_table = N'
Customers' , @destination_owner = N' dbo' , @vertical_partition = N' false' ,
@ins_cmd = N' CALL sp_MSins_dboCustomers' , @del_cmd = N' CALL
sp_MSdel_dboCustomers' , @upd_cmd = N' SCALL sp_MSupd_dboCustomers'
GO
```

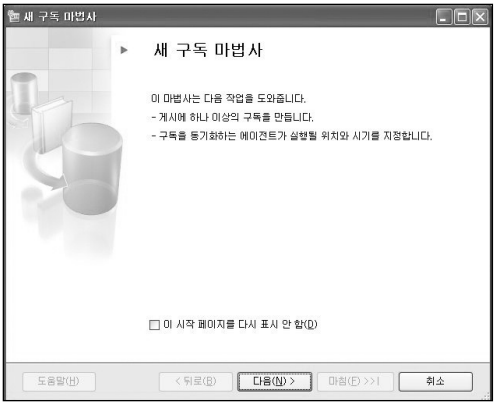
게시된 Northwind:NorthwindCustomers 에 대해 구독자로의 밀어넣기 구독을 만드는 내용으로 따라하기를 살펴보도록 하겠습니다.

[따라하기] 밀어넣기 구독 만들기

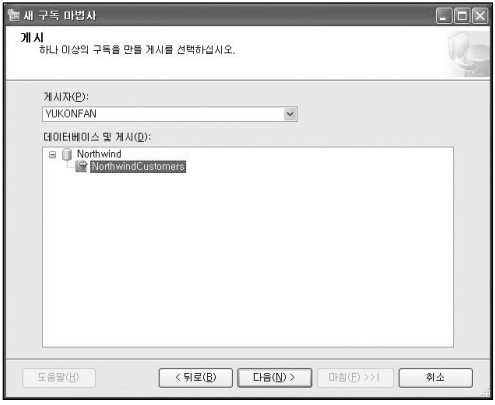
1 구독 - 로컬 구독 - 새 구독 메뉴를 선택합니다.



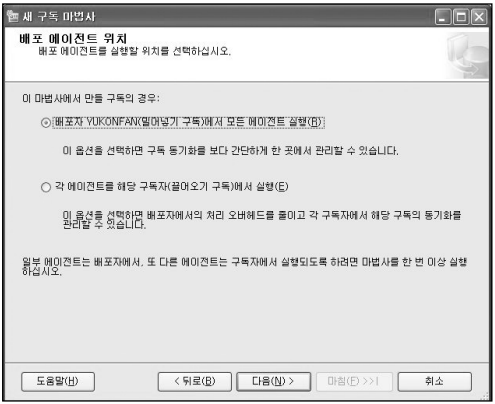
2 새 구독 마법사를 시작합니다. 역시 마법사에서 수행할 핵심 작업에 대해 설명하고 있습니다.



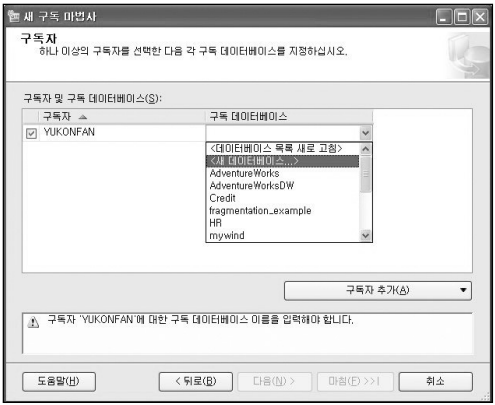
3 구독 원본을 가진 게시자 서버와 게시 항목을 선택합니다.



4 트랜잭션 복제에서 배포 에이전트는 트랜잭션 로그와 관련 메타 데이터 등의 저장과 구독자로의 전달을 담당합니다. 이러한 배포 에이전트의 실행 위치는 지정합니다. 모든 에이전트가 배포자(별도의 배포자이거나 혹은 게시자와 동일한 위치의 배포자) 에서 수행(밀어넣기 구독)되도록 설정하거나 혹은 각 에이전트를 구독자에서 수행(끌어오기)하도록 설정할 수 있습니다. 또한 필요할 경우 마법사를 한 번 이상 수행해서 병행할 수도 있습니다.

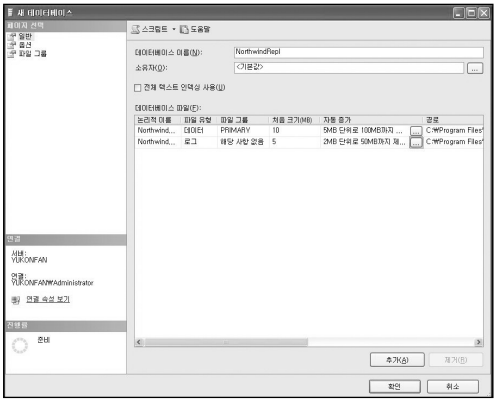


5 구독자를 선택(혹은 추가)하고 구독 데이터베이스를 선택합니다. 목록 상자에서 〈새 데이터베이스...〉 메뉴를 이용하면 새로운 데이터베이스를 생성하고 구독 데이터베이스를 지정할 수도 있습니다.

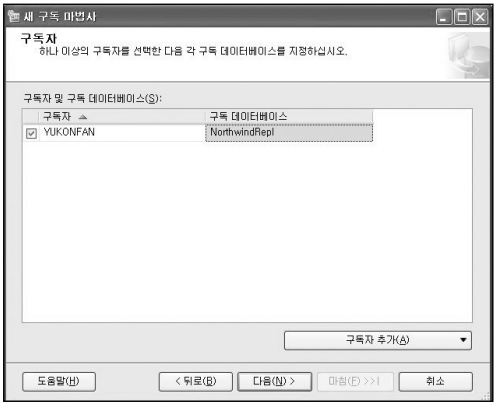


6 〈새 데이터베이스...〉 메뉴를 선택하면 새 데이터베이스 대화상자가 보입니다. 새로 만들 데이터베이스에 대한 정보를 입력합니다.

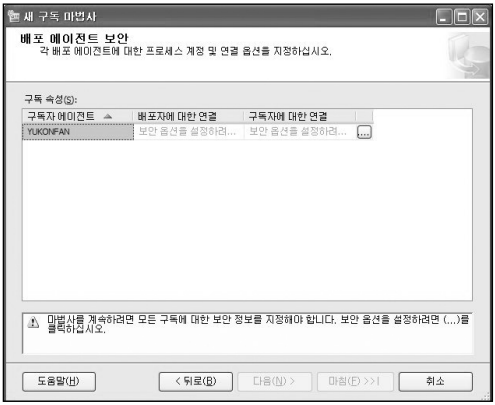
[참고]
실제로 구현 시에는 구독할 게시 데이터의 전체 크기와 향후 발생 트랜잭션의 크기를 충분하게 지정하셔야 합니다



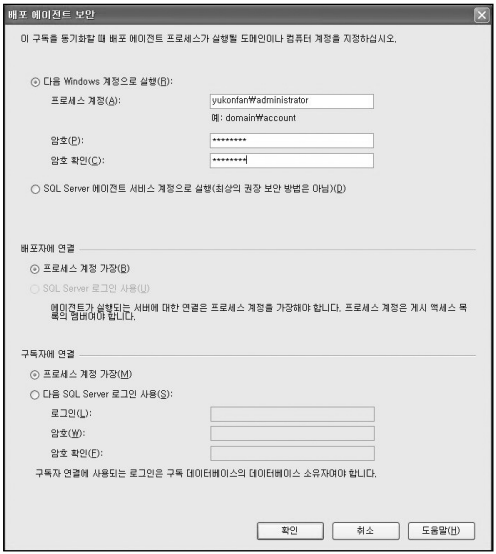
7 최종 구독 데이터베이스를 지정합니다.



8 배포 에이전트 수행 권한을 가질 사용자 계정과 배포자 및 구독자와의 연결 정보를 지정합니다.

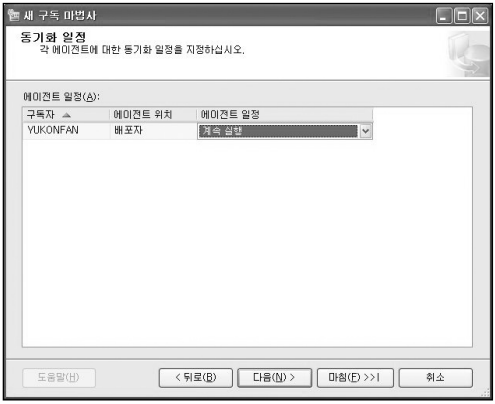


9 배포 에이전트가 어떤 사용자 계정으로 실행될 것인지 지정합니다. 기본적으로 도메인 사용자 계정이나 로컬 컴퓨터 사용자 계정을 지정하며, 해당 사용자는 관리자 수준의 필요 권한을 가져야 합니다. 그리고 배포자와 구독자에 연결할 때 사용할 계정으로 프로세스 계정을 대리하거나 SQL Server 표준 사용자 계정을 지정할 수 있습니다.



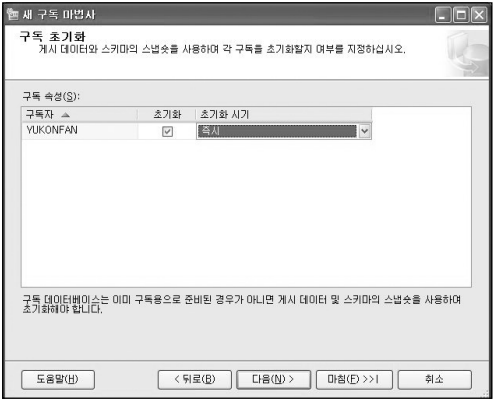
10 지정된 정보를 최종 확인합니다.

11 각 에이전트의 동기화 일정을 지정합니다. 에이전트 일정 항목의 목록 상자에서 데이터 동기화 작업을 수행할 시점에 대한 방법을 선택할 수 있습니다.

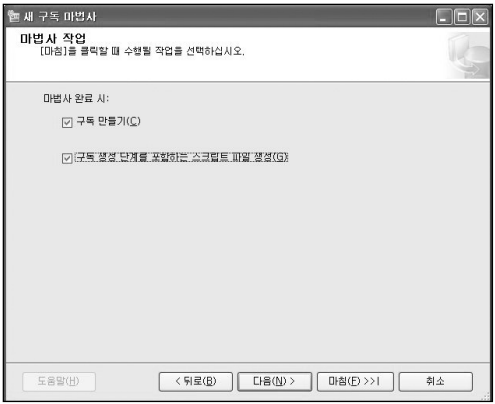


12 게시에서 만들어진 아티클의 스키마와 데이터에 대한 스냅숏으로 구독자 데이터베이스에 초기화를 수행할 것인지 여부와 방법을 지정합니다.

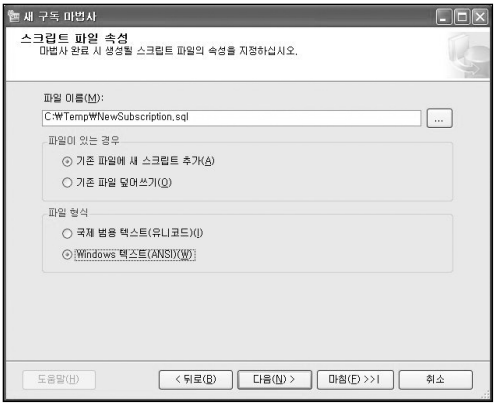
> ***[참고]***
> *SQL Server 2005는 대용량 데이터베이스의 스냅숏 초기화에 들어가는 시간과 비용 측면의 부하를 줄이기 위해 백업을 통한 초기화 지원 등의 새로운 기술을 지원합니다*



13 마법사의 최종 작업 결과로 구독을 생성하게 되며, 필요할 경우 구독 만들기의 내용을 스크립트로 생성하고 파일로 저장하도록 지정할 수 있습니다. 예제에서는 스크립트를 생성하도록 지정을 했습니다.

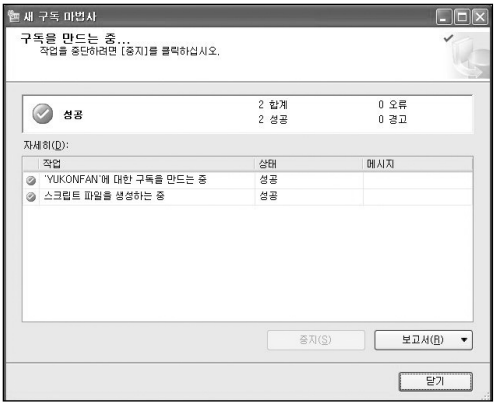


14 스크립트 생성 옵션을 지정하는 경우, 다음 화면에서 스크립트를 저장할 파일과 저장 관련 옵션들(덮어쓰기 여부, 파일 형식)을 지정합니다.

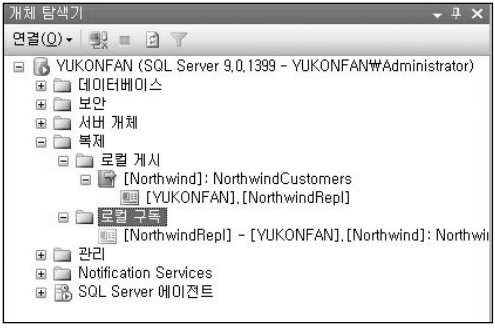


15 지금까지 설정한 구독 관련 속성 값들에 대해 정리를 합니다. 원하는 구독인지를 확인한 후 마침 버튼을 클릭합니다.

16 마지막으로 구독 생성 작업을 수행하고 문제가 발생할 경우 관련 메시지를 링크 형태로 보여줍니다. 하단의 보고서 버튼을 통해서 수행 결과에 대한 추가 내용을 확인할 수 있습니다.



17 이제 복제 - 로컬 구독 항목에서 새로운 구독 [NorthwindRepl - [YUKONFAN]. [Northwind] :NorthwindCustomers 가 보여집니다.



그리고 마법사에서, 스크립트 생성을 통해 만들어진 스크립트 파일의 내용은 다음과 같습니다.

[예제 구독 만들기 스크립트, NewSubscription.sql]

```
----------시작: 게시자 'YUKONFAN' 에서 실행할 스크립트----------
use [Northwind]
exec sp_addsubscription @publication = N' NorthwindCustomers' , @subscriber = N' YUKONFAN' , @destination_db = N' NorthwindRepl' , @subscription_type = N' Push' , @sync_type = N' automatic' , @article = N' all' , @update_mode = N' read only' , @subscriber_type = 0

exec sp_addpushsubscription_agent @publication = N' NorthwindCustomers' , @subscriber = N' YUKONFAN' , @subscriber_db = N' NorthwindRepl' , @job_login = N' yukonfan\administrator' , @job_password = null, @subscriber_security_mode = 1, @frequency_type = 64, @frequency_interval = 0, @frequency_relative_interval = 0, @frequency_recurrence_factor = 0, @frequency_subday = 0, @frequency_subday_interval = 0, @active_start_time_of_day = 0, @active_end_time_of_day = 235959, @active_start_date = 20060218, @active_end_date = 99991231, @enabled_for_syncmgr = N' False' , @dts_package_location = N' Distributor'
GO
----------끝: 게시자 'YUKONFAN' 에서 실행할 스크립트----------
```

지금까지 새로운 복제 구성 마법사를 통해서 트랜잭션 복제를 구성해보고, SQL Server 2005에서 복제 구성 마법사가 얼마나 쉽고 편리해 졌는지 경험해 보았습니다.

향상된 복제 모니터

SQL Server 2005의 복제 모니터 또한 이전과 비교할 수 없을 정도로 보다 실질적이고 강력한 기능들을 제공하기 위해 재 설계되었습니다. 이전까지 복제 관리자들의 모니터링 관련 중요한 문제 중의 하나는 복제 시스템의 성능 모니터링과 병목 상황에 대한 판단을 위한 지원이 부족하다는 것이었습니다. 몇 가지 질문을 예로 들어 보면,

복제 성능에 대한 공통적인 질문

- 복제 시스템이 정상적으로 동작하고 있는가?
- 에이전트가 실행되지 않는 이유는 무엇인가?
- 시스템이 왜 이렇게 느리지?
- 복제하는데 시간이 얼마나 걸리는가?
- 어디에 잠재적인 문제가 발생하고 있는가?
- 문제 상황을 사전에 인식하고 조치를 취할 수는 없는가?

SQL Server 2005에 복제 모니터는 이러한 질문의 답을 제공합니다. 여러 가지 혁신적인 기술들을 포함하고 있는 새로운 복제 모니터는 이제 복제 작업 전반에 걸쳐 모니터링이 가능합니다. 게시 및 배포와 구독 상태에 대해 진단 분석이 가능하면 성능 문제에 대한 자세한 모니터링이 가능합니다. 또한 임계 값을 기준으로 경고 설정 기능이 제공됩니다.

새로운 복제 모니터의 특징을 정리하면 다음과 같습니다.

복제 모니터의 특징	설명
성능 임계값 설정	잠재적인 성능 문제를 사전에 식별하기 위해 기준이 되는 조건과 임계 값을 정의하면 해당 상황에서 경고를 발생시키고 모니터링을 할 수 있습니다. 복제 구성 시 기본적인 몇 가지 조건이 설정됩니다.
모니터링 부하 최소화	복제 모니터는 많은 컴퓨터를 효율적으로 모니터링 하도록 설계되었습니다. 복제 모니터가 사용하는 쿼리는 정기적으로 캐시되며 새로 고쳐집니다. 캐싱을 사용하므로 여러 페이지를 볼 때 불 필요한 쿼리 및 계산 작업을 줄일 수 있습니다. 기본적으로 주 복제 모니터 창은 자동으로 5초마다 새로 고침을 수행하며, 게시자 설정(Publisher Settings) 메뉴를 통해서 자동 새로 고침 여부와 빈도를 조정할 수 있습니다.
병합 복제 모니터링	병합 복제의 각 처리 단계(변경 내용 업로드, 다운로드 등)에 소요된 시간을 포함해서 동기화 중에 처리된 각 아티클에 대한 자세한 통계를 표시하므로 성능 문제의 원인을 식별하는데 도움을 줍니다.
트랜잭션 복제 모니터링	트랜잭션 복제는 이제 트랜잭션의 전 단계의 걸쳐 실 시간의 추적이 가능합니다. 이는 게시자에서 배포자로 그리고 하나 이상의 구독자에 대해 트랜잭션이 커밋되는 시간을 저장하는 추적 프로그램 토큰(tracer token)이라는 새로운 기술을 통해 제공됩니다.

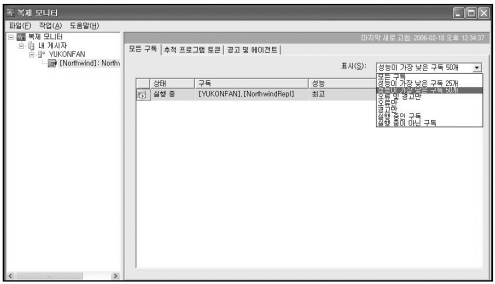
이외에도 많은 향상 기능들이 제공이 됩니다.

다음 그림들은 복제 모니터를 수행 화면에 대한 중요 기능 들을 보여줍니다.

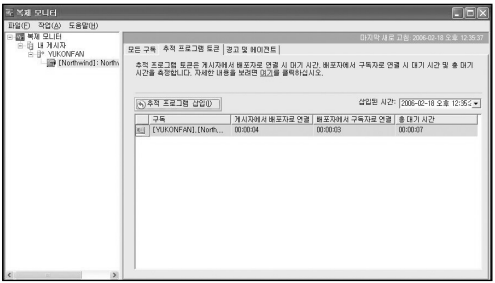
[따라하기] 복제 모니터

1 개체 탐색기 - 해당 서버 - 복제 에서 팝업 메뉴를 띄우고, 복제 모니터 시작 메뉴를 선택합니다.

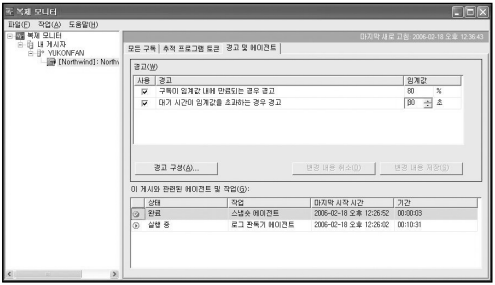
2 복제 모니터에서 해당 게시 정보를 선택하면 관련된 정보를 모니터링 할 수 있습니다. 우선 모든 구독 탭에서 모든 구독자의 실행 정보와 관련된 작동 상태 및 성능 문제 여부 등을 바로 확인할 수 있으며 오른쪽의 표시: 목록 상자를 통해서 성능 문제가 있거나 오류 및 경고가 있는 구독에 대한 필터링을 수행할 수 있습니다.



3 추적 프로그램 토큰 탭에서는 추적 프로그램 삽입 버튼을 이용해서 트랜잭션 복제 로그에 추적 프로그램 토큰을 추가하도록 지정하고 해당 토큰의 복제 경로를 따라 복제 수행에 걸리는 각 영역별(게시->배포, 배포->구독) 수행 시간 및 지연 시간 정보 등을 모니터링할 수 있습니다.



4 경고 및 에이전트 탭에서는 성능 문제 임계 값을 설정하거나 게시 관련된 에이전트와 작업에 대한 상세 정보를 모니터링 할 수 있습니다.

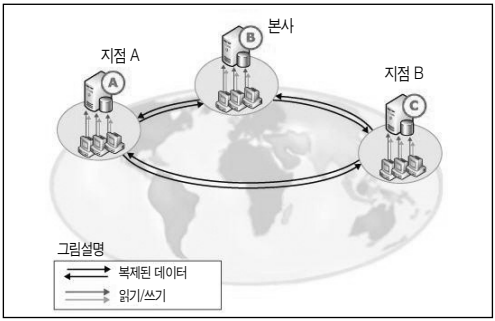


복제 모니터뿐만 아니라, SQL Server Management Studio, Transact-SQL 및 RMO, 복제 에이전트에 대한 경고 설정, 윈도우즈 시스템 모니터 등을 통해서 복제에 대한 모니터링을 수행할 수 있습니다.

피어 투 피어(Peer-to-Peer) 복제

SQL Server 2005는 피어 투 피어 복제라는 새로운 개념의 복제를 도입했습니다. 이는 피어 투 피어 네트워크 토폴로지와도 개념이 유사합니다. 즉, 일반적인 트랜잭션 복제가 게시자 밑에 여러 구독자가 존재하는 계층 구조 형식이며, 구독자가 일반적으로 읽기 전용이라는 것을 기본으로 합니다(물론 업데이트 가능 구독도 지원을 하지만 이 또한 계층적 형식으로 반영됩니다).

반면에, 피어 투 피어 복제는 각각의 노드(node)가 게시자이자 구독자 역할을 하며 서로 다른 노드끼리 1대1의 복제를 구성하는 형식입니다.



웹 서비스를 하는 데이터베이스를 가정해 보죠, 피어 투 피어 복제가 항상 동기화된 데이터를 가지고 있으므로 각 노드에 대한 분산된 데이터 읽기 혹은 쓰기 작업이 가능하며, 사용자의 데이터 처리 또한 각 노드에 적용하거나 특정 노드에만 적용할 수 있습니다. 한 서버에 장애가 발생하는 경우 동일한 데이터를 가진 다른 서버를 사용할 수 있습니다.

피어 투 피어 복제는 시나리오에 따라서 다양한 토폴로지를 설계할 수 있습니다.

피어 투 피어 복제의 구성은 기본적으로 트랜잭션 복제 형식으로 구성한 뒤에, 피어 투 피어 구독을 허용하고 나서 피어 투 피어 토폴로지 구성(Configure Peer-To-Peer Topology) 메뉴를 통해 초기 구성을 할 수가 있습니다.

또한 초기 토폴로지 구성 후에도 필요한 경우 노드를 추가하는 작업이 가능하므로 이를 통해 확장 가능한 피어 투 피어 복제 구성을 할 수 있습니다.

피어 투 피어 복제는 특히 다음과 같은 이득이 필요한 요구 사항이 있을 경우 적용 시나리오를 고려해 볼 수 있습니다.

피어 투 피어 복제를 통해 얻을 수 있는 이득

- 데이터베이스 어플리케이션에 대한 고 가용성과 수평 확장 능력 제공
- 단일 실패 지점의 문제 해결
- 온라인 상태에서의 유지 관리 작업
- 쿼리에 대한 데이터베이스 간 확장 기능 제공

피어 투 피어 복제를 구성 과정은 다음과 같습니다.

- 복제에 참여하는 모든 노드(서버)에 개별 배포자를 구성합니다.
- 복제에 참여하는 모든 노드에는 동일한 데이터베이스를 구성합니다.
- 첫 번째 노드에서 트랜잭션 복제를 구성합니다. 이 때 필터링과 같은 제한 작업들을 구성해서는 안 됩니다.
- 만들어진 게시의 속성(Publication Properties) 메뉴에서 구독 옵션 부분에 피어 푸 피어 구독 허용(Allow peer-to-peer subscriptions) 항목을 True로 설정합니다.
- 게시의 피어 투 피어 토폴로지 구성(Configure Peer-to-peer Topology) 메뉴를 통해서 마법사를 구동하고, 다른 노드를 추가하고 필요한 속성을 설정하는 작업을 수행합니다.

추적 프로그램 토큰

많은 서버 및 데이터베이스 간의 트랜잭션 복제는 모니터 및 유지 관리 작업에 부담이 됩니다. SQL Server 2005는 추적 프로그램 토큰(Tracer Token)이라는 기술을 통해서 복제 시스템의 추적 및 진단분석 그리고 성능 모니터링에 대한 보다 향상된 기능을 제공합니다.

앞서 복제 모니터를 통해서 소개된 대로 트랜잭션 게시의 경우 추적 프로그램 토큰(Tracer Token) 을 삽입할 수가 있습니다. 추적 프로그램 토큰은 적은 양의 데이터로 게시자의 트랜잭션 로그에 삽입된 후 배포자와 구독자로 함께 복제가 됩니다. 따라서 게시에서부터 구독에 이르기까지 명령이 전달되는 과정을 추적할 수가 있게 됩니다. 추적 프로그램 토큰이 각 시스템 간의 이동이 일어날 때 관련 통계가 수집되고 시스템 테이블에서 이러한 통계를 쿼리를 할 수가 있는 것입니다.

추적 프로그램 토큰을 통해서 다음과 같은 상세 정보를 얻을 수 있습니다
- 로그 판독기의 지연 시간
- 배포 지연 시간
- 게시부터 구독까지의 지연 시간
- 위 상세 정보를 통해서 다음과 같은 진단 분석을 할 수 있습니다
- 가장 오래 걸리 구독은 무엇인가?
- 아직 연결되지 않은 구독자는 누구인가?

DDL 구문 지원

이전 버전에서는 복제 아티클로 지정된 테이블에 칼럼을 추가하거나 삭제하는 작업이 복제용의 특정 프로시저를 사용하도록 제한적으로만 지원되었습니다.

SQL Server 2005에서는 다음과 같은 DDL 구문에 의한 스키마 변경은 자동 복제됩니다
- ALTER TABLE
- ALTER VIEW
- ALTER PROCEDURE
- ALTER FUNCTION
- ALTER TRIGGER

이 속성은 해당 게시의 구독 옵션을 통해서 조정할 수도 있습니다.

Oracle도 이제 복제 게시자 역할 수행

드디어 Oracle에서 SQL Server로의 복제가 지원됩니다. 현업의 많은 시스템들이 서로 다른 업무나 서비스, 혹은 솔루션 및 소프트웨어의 도입 과정에서 Oracle과 SQL Server를 함께 운영하고 있습니다. 그러나 이기종 시스템 간의 데이터를 협업하는 과정에서 어려움을 겪고 있는 것 또한 사실입니다.

SQL Server 2005에서는 트랜잭션(혹은 스냅숏) 복제에 대해 Oracle이 게시자 역할을 할 수 있도록 지원합니다(Oracle 버전 8 이상). SQL Server 복제와 동일한 방식으로 Oracle 데이터베이스를 SQL Server에 직접 복제를 할 수 있는 것입니다.

Oracle 측에서 또한 추가 소프트웨어 설치가 필요 없으며, SQL Server처럼 관리를 할 수가 있습니다. 따라서 기존의 복제 구성 및 관리에 대한 SQL Server 지식을 그대로 활용할 수가 있는 것입니다.

- Oracle 게시 및 복제 방법은 Oracle 데이터베이스 내의 게시된 각 테이블에 대해 트리거와 추적 테이블 생성되고 게시 테이블의 데이터가 변경되면 데이터베이스 트리거가 실행되어서 변경된 각 행에 대한 정보를 관련된 복제 테이블에 삽입합니다. SQL Server 배포자의 로그 판독기 에이전트는 데이터 변경 정보를 배포 데이터베이스로 이동시키며 이후의 동작은 SQL Server 표준 복제로서 수행됩니다.
- Oracle 게시를 통해 수행할 수 있는 작업은 다음과 같습니다.
- 데이터 공존 및 협업을 위한 전략
- .NET 개발, 리포팅, 데이터 웨어하우스 스테이징 데이터베이스를 위한 시나리오
- 데이터 마이그레이션 작업에 대한 지원

기타 추가 향상 기능

■ 구독 초기화 지원

SQL Server 2000에서는 구독 데이터베이스 초기화를 위해서 스냅숏 복제를 사용함으로 인해서 대용량 데이터 집합의 처리 시 많은 시간과 비용을 요구했습니다. SQL Server 2005에서는 다음과 같은 새로운 방법을 통해서 구독 데이터베이스 초기화를 수행할 수 있도록 지원합니다.

■ 백업과 복원
■ 데이터베이스 복사
■ 동적 스냅숏

■ 논리적 레코드 - 병합 복제의 동시성 지원

SQL Server 2005는 논리적 레코드(Logical Record)라는 개념의 도입으로 병합 테이블의 아티클 간의 관계를 정의함으로써, 관련된 행 집합이 모두 함께 전송되어서 데이터 일관성을 보장할 수 있도록 지원합니다.

예를 들어, 실제 주문이 고객, 주문, 주문상세 테이블로 표현된다고 가정해 보죠. 만일 특정 주문이 주문, 주문상세 테이블을 통해서 등록되거나 변경된 경우 두 테이블의 변경 사항이 모두 다 복제에 적용되거나 아무것도 적용되지 않아야 하는 비즈니스 시나리오를 가진다면, 동기화를 수행하는 도중에 복제가 중단되거나 문제가 발생하는 경우 데이터의 일관성을 보장해 줄 수 있어야 합니다. 병합 복제(게시) 구성 시의 논리적 레코드의 관계를 설정합니다. 논리적 레코드는 테이블 간의 설정된 관계를 통해서 이러한 문제를 해결하고 논리적 레코드 단위의 충돌을 처리할 수 있게 지원합니다.

■ 프로그래밍 능력

RMO (Replication Management Objects)

이전의 프로그래밍 가능한 SQL Server 관리 개체였던 SQL-DMO와 달리, SQL Server 2005에서는 .NET 관리 코드(Managed Code)의 기반의 프로그래밍 가능한 복제 관리 개체인 RMO를 제공합니다. Visual Studio 2005 등의 개발 환경에서 Microsoft.SqlServer.Replication .NET Programming Interface (microsoft.sqlserver.rmo.dll) 참조를 통해서 SQL Server 2005의 새로운 복제 기능을 포함한 복제 시스템의 프로그래밍 요구 사항들을 모두 구현할 수 있는 강력한 프로그래밍 개체로 제공됩니다.

[참고]
RMO에 대한 예제는 다른 모듈에서 별도로 소개될 것입니다.

웹을 통한 동기화

무선 사용자의 증가로 인해서 웹을 통한 데이터의 복제 및 동기화 방법의 요구가 커지고 있습니다. 보안 상의 이유로 VPN(Virtual Private Network, 가상사설망)을 고려할 수 있지만 적지 않은 비용이 요구되거나 작은 조직에서 사용하기엔 부담스러운 구성입니다.

SQL Server 2005는 병합 복제에 대해 웹 동기화를 도입했습니다. 이는 마이크로소프트 IIS 웹 서버에서 HTTPS 와 XML 메시지를 통해 모바일 장비 간 혹은 방화벽을 사이에 둔 서버 간의 복제를 지원합니다.

SQL Server 2005에서 사라지는 기능들

이전 버전에서 지원되던 다음 기능들은 SQL Server 2005에서 지원되지 않을 수도 있습니다.

■ 데이터 복제 시 DTS 패키지를 연동
■ 업데이트 가능 구독에서 마이크로소프트 메시지 큐 지원
■ ODBC 구독자

SQL Server 2005 성능 및 확장성

SQL Server 2005는 대용량 데이터베이스의 복제 지원을 위한 성능 향상 및 확장성의 증대를 위해 많은 추가 기능들을 제공합니다. 다음 표에서 이러한 기능들을 간단히 정리합니다.

[표] SQL Server 2005 성능 및 확장성

성능 및 확장성	설명
미리 계산된 파티션	병합 복제에서 매개 변수가 있는 필터링(이전의 동적 필터링)을 사용하는 경우 구독자가 게시자와 동기화되면 게시자는 구독자의 필터를 평가해서 그 구독자의 파티션이나 데이터 집합에 속하는 지를 평가해야 합니다. 이 과정을 파티션 평가라고 합니다. 미리 계산된 파티션은 게시자의 변경 내용에 대해 파티션 멤버 자격을 미리 계산하고 변경 후에 지속하므로 구독자와 게시자가 동기화될 때 파티션 평가를 거치지 않고 관련된 변경 내용을 즉시 다운로드 할 수 있으므로 게시와 변경이 많은 경우에 성능이 향상됩니다.
공유 키 업데이트	SQL Server 2000에서는 고유 키에 대한 수정이 실제로는 DELETE와 INSERT로 복제되었습니다. SQL Server 2005에서는 이제 UPDATE로 복제되므로 정기적으로 데이터가 수정되는 경우의 성능을 향상시킵니다.
병렬 트랜잭션 복제	트랜잭션 복제에서, 게시자에서 배포자로 로그 판독기 에이전트에 의한 트랜잭션 비율은 적합한 반면, 배포자에서 구독자로의 트랜잭션 전송 비율이 상대적으로 떨어져서 성능 문제가 발생할 수 있었습니다. 이는 다중 클라이언트에서 게시자에 트랜잭션이 병렬로 발생하는 반면 구독자로는 직렬로 전송되기 때문입니다. SQL Server 2000에서는, 독립 에이전트 옵션으로 게시자와 구독자별로 독립적인 에이전트를 사용하도록 구성해서 성능 문제를 해결하는데 도움이 되었습니다. SQL Server 2005에서는 이 옵션이 디폴트입니다.

다운로드 전용 아티클	병합 게시에서 게시자에서만 데이터가 변경되고 구독자로 다운로드만 필요한 경우, 다운로드 전용으로 아티클을 지정해서 성능을 향상시킬 수 있습니다.
충돌 감소	보고서 및 읽기 작업 위주로 만들어진 구독에서 충돌을 줄이기 위해 SQL Server 2005의 새로운 기술들을 사용할 수 있습니다. 예를 들어, 최신 데이터를 필요로 하는 대량의 구독자를 가진 경우 스냅숏 격리 수준을 사용할 수 있습니다. 혹은 특정 시점의 데이터만 필요로 하는 경우, 데이터베이스 스냅숏을 만들어서 제공할 수 있습니다.

데이터베이스 스냅숏

2005는 CREATE DATABASE 문에서 새로운 AS SNAPSHOT OF 라는 절을 추가했습니다. (엔터프라이즈 버전)이 SNAPSHOT(스냅숏)을 이용하면 읽기전용이며 트랜잭션의 일관성을 보장해는 데이터베이스의 스냅숏을 생성합니다. 이 스냅숏의 일반적인 용도는 클라이언트에서 리포팅과 같은 읽기 전용의 응용프로그램을 위한 미러링 된 데이터베이스로 사용할 수 있습니다. 클라이언트에게 최근의 자료를 유지 하기 위해서는 주기적으로 새로운 스냅숏을 생성해야 합니다. 각각의 데이터베이스 스냅숏은 다른 데이터베이스에 대해서 독립적으로 분리되어 존재 합니다.

또 다른 스냅숏의 활용방안은 데이터베이스 사용자가 실수를 했을 경우입니다. 이런 경우 최근의 스냅숏으로부터 손실된 데이터를 복구 할 수 있습니다.

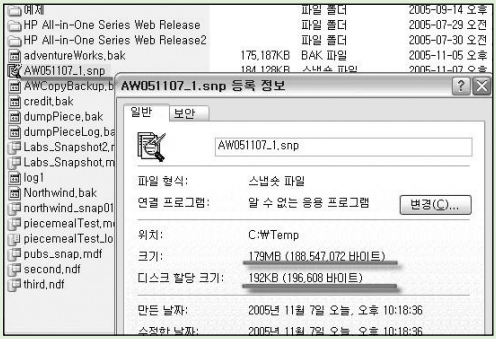
스냅숏을 생성하기 위해서는 Management Studio에서 GUI를 통해서 생성할 수 없습니다. CREATE DATABASE 명령문에 AS snapshot of 문을 사용하여 생성해야 합니다.

[따라하기]스냅숏데이터베이스 생성하기

```
--1) 스냅숏생성하기
USE master
GO

CREATE DATABASE AW_snap051107_1
ON
(
  NAME = AdventureWorks_Data
, FILENAME=' c:\temp\AW051107_1.snp'
)
AS SNAPSHOT OF AdventureWorks
```

GO

스냅숏데이터베이스의 특징은 크기는 원본 데이터베이스과 같으나 디스크할당 크기는 얼마되지 않습니다. 현재 원본소스의 껍데기만 가지고 있는 것입니다.

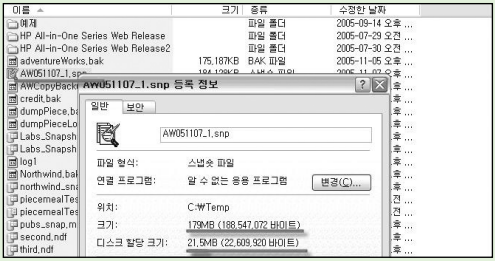


--원본 데이터베이스에 테이블을 수정합니다.

```
USE AdventureWorks
GO
UPDATE Sales,SalesOrderDetail
SET OrderQty = OrderQty + 1
WHERE SalesOrderID 〈 =43669

SELECT * FROM Sales,SalesOrderDetail
WHERE SalesOrderID 〈 =43669

USE AW_snap051107_1
go
SELECT * FROM Sales,SalesOrderDetail
WHERE SalesOrderID 〈 =43669
```

위의 두개의 쿼리를 비교하면 서로 다른 OrderQty 값을 가지고 있습니다.

아래의 그림은 원본데이터베이스의 테이블을 수정하면 크기는 동일하나 디스크 할당 크기가 늘어난 것을 볼 수 있습니다.



-- snapshot 데이터베이스에 없는 정보는 원 소스 데이터베이스에서 읽어오게 됩니다.

SELECT * FROM Sales,SalesOrderDetail
WHERE SalesOrderID = 43700

-- 스냅숏데이터베이스에 대한 질의
스냅숏데이터베이스를 생성하게 되면 원본데이터베이스의 할당크기와 동일한 데이터베이스를 생성합니다. 이 스냅숏데이터베이스에는 실제적인 값이 있지 않습니다. 원본데이터베이스에 변경이 생기면 변경이전의 값을 스냅숏데이터베이스에 복사하여 기록합니다. 이때 비로소 스냅숏데이터베이스에 값을 기록하게 됩니다.
스냅숏데이터베이스를 이용하여 쿼리를 수행할 때 스냅숏데이터베이스에 없는 값(즉, 스냅숏 생성 후 변경이 없는 행)을 읽어오는 경우에는 원본데이터베이스에서 그 값을 불러옵니다.

-- snapshot 데이터베이스에 대한 업데이트

```
-- 스냅숏 데이터베이스는 읽기데이터베이스이기 때문에 수정이나 제거는 불가능합
니다.
UPDATE Sales.SalesOrderDetail
SET OrderQty = OrderQty + 1
WHERE SalesOrderID 〈 =43669

---------------------------------------------------------------------------
--메시지3906, 수준16, 상태1, 줄2
--데이터베이스" northwind_snapshot01" 은(는) 읽기전용이므로업데이트할수없습니다.
```

[따라하기]스냅숏데이터베이스의 소스 데이터베이스를 알아내기.

```
USE  master
GO

SELECT  db_name( source_database_id) FROM sys.databases
WHERE  database_id =  db_id( 'AW_snap051107_1' )
```

스냅숏데이터베이스를 이용한 사용자 실수 복구

스냅숏데이터베이스를 이용하시면 사용자의 실수로 인한 데이터 손실을 복구 할 수 있습니다.

[따라하기] 사용자 실수로부터 테이블 복구하기

```
SET NOCOUNT ON
USE master
go
-- 스냅숏데이터베이스를 생성합니다.
CREATE DATABASE aw_051110_01 ON
```

```
(
        NAME = AdventureWorks_data
,       FILENAME =' c:\temp\AW_051110_01.ss'
)
AS SNAPSHOP OF AdventureWokrs
GO

-- AdventureWorks 데이터베이스에 있는 테이블을 삭제합니다.
USE AdventureWorks
GO
DROP TABLE dbo.databaseLog;
go

USE master
GO

-- 스냅숏데이터베이스를 이용하여 복원을 합니다.
RESTORE DATABASE AdventureWorks
FROM DATABASE_SNAPSHOP =  'aw_051110_01'
GO

USE AdventureWorks
GO

SELECT * FROM dbo.DatabaseLog
GO
```

주의점

데이터베이스의 스냅숏은 소스데이터베이스와 동일한 인스턴스에 존재해야 합니다.
하나의 소스데이터베이스에 대해서 여러 개의 스냅숏데이터베이스가 가능합니다.
이들 역시 동일한 인스턴스에 있어야 합니다.

1. 페이지가 업데이트 될 때 마다 스냅숏에 기록을 해야 하기 때문에 성능은 감소됩니다.

2. 스냅숏데이터베이스가 있으면 원본데이터베이스는 제거될 수 없습니다. 스냅숏데이터베이스가 우선적으로 삭제되어야 원본데이터베이스를 제거 할 수 있습니다.

3. 시스템데이터베이스인 master, msdb, model 데이터베이스는 대상이 되지 않습니다.

4. 스냅숏데이터베이스는 백업, 복원, 연결, 분리되지 않습니다.

5. 스냅숏데이터베이스의 허가권은 원본 데이터베이스의 것을 그대로 상속받습니다. 하지만, 읽기 전용속성을 가지고 있기 때문에 원본 데이터베이스의 보안 특성이 변경된다 해서 스냅숏데이터베이스에 적용되지 않습니다.

6. 스냅숏데이터베이스는 NTFS로 구성된 디스크파티션에서만 구현이 가능합니다.

백업의 새로운 점

복사전용(COPY ONLY)백업

2005에서 새로운 백업 옵션을 도입했습니다. 복사백업은 테스트 등을 목적으로 만드는 백업복사본을 만듭니다. 복사 백성을 생성하는 과정에서 기존의 전체, 차등, 로그백업의 순서에는 영향을 전혀 주지 않습니다.

복사 전용 백업은 백업 전체에 영향을 주지 않도록 지정합니다. 복사 전용 백업은 백업 전체에 영향을 주지 않고 데이터베이스에 대한 프로시저를 복원합니다.

[따라하기] 생성 방법은 다음과 같습니다.

```
BACKUP DATABASE AdventureWorks
TO disk=' c:₩temp₩AWCopyBackup.bak'
WITH COPY_ONLY
```

복사백업을 이용하여 수행된 데이터 백업은 차등백업이나 다른 백업의 기준으로 사용될 수 없습니다. 그리고, 전체 백업 후 수행된 복사 백업이 차등백업에 영향을 주지 않습니다.

[따라하기] 다음 과 같은 경우를 생각해봅시다.

```
--12시정각에 전체백업을 받습니다.
BACKUP DATABASE AdventureWorks
TO disk=' c:₩temp₩AWCopyBackup.bak'
WITH INIT

--12시20분데이터변경이발생됩니다. (Update, Insert, Delete)
```

```
UPDATE sales.SpecialOffer
SET DisCountPct = DisCountPct + 0.01

--12시30분복사백업을전체백업으로받습니다.
BACKUP DATABASE AdventureWorks
TO disk=' c:₩temp₩AWCopyBackup.bak'
WITH COPY_ONLY

--12시40분차등백업을수행합니다.
BACKUP DATABASE AdventureWorks
TO disk=' c:₩temp₩AWCopyBackup.bak'
WITH DIFFERENTIAL
```

위의 경우에서 12시 40분에 수행하는 차등백업은 기존의 백업 방식에서 12시 30분에 수행된 전체 백업 이후에 변경된 내용을 백업하게 됩니다. 하지만 12시 30분에 수행된 복사 백업은 전체 백업 프로세스에 영향을 주지 않고 12시 정각 이후 발생된 내용에 대해서 백업을 수행하게 됩니다.

로그백업에서 복사백업 옵션을 수행 시 로그는 잘려지지 않고 이 복원과정과는 상관이 없습니다.

부분백업 (Partial Backup)

■ 부분백업
부분 백업은 전체 데이터베이스 백업과 유사하지만 주 파일 그룹과 모든 읽기/쓰기 파일 그룹만 포함해야 합니다. 경우에 따라 읽기 전용 파일을 BACKUP 명령에 나열하여 부분 백업에 포함할 수 있습니다.
부분 백업을 지정하려면 READ_WRITE_FILEGROUPS 옵션을 이용합니다.

예를 들면 다음과 같습니다.

```
BACKUP DATABASE AdventureWorks
READ_WRITE_FILEGROUPS
TO 〈backup_device〉
```

■ 부분 백업 다음에 수행된 부분 차등 백업
부분 차등 백업에는 주 파일 그룹과 모든 읽기/쓰기 파일 그룹에서 변경된 데이터만 포함됩니다.

부분 백업은 전체 백업과 비슷하지만 부분 백업에는 모든 파일 그룹이 포함되지는 않습니다. 부분 백업에는 주 파일 그룹, 모든 읽기/쓰기 파일 그룹 및 지정한 모든 읽기 전용 파일의 모든 데이터가 포함됩니다. 읽기 전용 데이터베이스의 부분 백업에는 주 파일 그룹만 포함됩니다.

부분 백업 및 부분 차등 백업은 사용이 쉬우며 주로 단순 복구 모델에서 사용하기 위해 제공됩니다. 그러나 부분 백업은 복구 모델에 관계없이 모든 데이터베이스에 대해 수행할 수 있습니다.

백업미디어에 대한 신뢰성 향상

백업 및 복원 전략을 잘 세워서 수행을 한다 해도 백업미디어에서 오류가 있다면 이것은 별개의 문제입니다. 또한 RAID 장비를 이용하여 이중 삼중 데이터보호에 신경을 쓴다 해도 백업 미디어에 문제가 있다면 백업된 자료는 별 도움이 되지 않습니다.

2005에서는 체크섬을 이용하여 백업 및 복원시 데이터베이스의 정합성을 검사합니다. 또한, 미러백업을 통해서 같은 내용을 서로 다른 위치에 백업이 가능하도록 되었습니다. 그리고, 백업과 복원과정에서 발생되는 에러에 대해서 경고를 발생합니다. 이런 기능을 통해서 만에 하나 일어날 수 있는 에러상황에서도 데이터를 복원할 수 있으며 백업미디어의 에러를 줄이게 됩니다.

체크섬(Checksum) 이용한 백업과 복원의 데이터 신뢰성 향상

BACKUP 과 RESTORE 명령어에서 CHECKSUM 절을 지원합니다. 이 절은 백업과 복원의 작업에 있어서 데이터베이스 자체의 신뢰성을 향상시켰습니다. CHECKSUM 절을 이용하면 백업과 복원과정에서 데이터베이스의 검증과 에러탐색을 가능하게 합니다.

백업과정에서 이 절을 사용하면 SQL Server는 백업과정에서 Checksum값을 계산하게 됩니다. 아울러 SQL Server는 페이지수준의 정보까지 검증하게 됩니다. 만약 페이지의 checksum 값이나 Torn 페이지가 있는지 확인합니다.
msdb..backupset 테이블에 has_backup_checksums 컬럼에 해당 백업세트에 checksum에 대한 정보가 있습니다.

[따라하기]백업명령어에 Checksum절을 사용하기

```
BACKUP DATABASE Adventureworks
TO disk =' c:₩temp₩Adventureworks.bak'
WITH CHECKSUM , init
```

```
--
--파일1에서데이터베이스' Adventureworks' , 파일' AdventureWorks_Data' 에대해
21808개의페이지를처리했습니다.
--파일1에서데이터베이스' Adventureworks' , 파일' AdventureWorks_Log' 에대해2
개의페이지를처리했습니다.
--BACKUP DATABASE이(가) 21810개의페이지를20.226초동안처리했습니다
(8.833MB/초).
```

문제가 발생되면 백업은 중지됩니다.

복원 시에 이 옵션을 사용하면 복원하는 과정에서 데이터의 정합성을 검사하게 됩니다.

```
RESTORE DATABASE adventrureworks
FROM 'c:₩temp₩Adventureworks.bak'
WITH CHECKSUM
```

만약 CONTINUE_AFTER_ERROR옵션을 사용하면 checksum에러가 발생해도 백업과 복원은 그대로 진행이 됩니다.

한편, 체크섬을 이용하지 않고 백업받은 자료에 대해서는 WITH Checksum 옵션을 이용하여 복원할 수 없습니다.

```
-- CheckSum없이 백업하기
BACKUP DATABASE Adventureworks
TO disk =' c:\temp\Adventureworks.bak'  with init

-- CheckSum 을 사용하여 복원하기
RESTORE DATABASE adventrureworks
FROM Disk  =' c:₩temp₩Adventureworks.bak'
```

```
WITH CHECKSUM

--메시지3187, 수준16, 상태1, 줄1
--백업세트에체크섬정보가없으므로RESTORE WITH CHECKSUM을지정할수없습
니다.
--메시지3013, 수준16, 상태1, 줄1
--RESTORE DATABASE이(가) 비정상적으로종료됩니다.
```

RESTORE VERIFYONLY 명령문은 복원하기 전에 단지 checksum 옵션만을 수행하게 됩니다.

```
RESTORE VERIFYONLY
FROM disk = 'c:₩temp₩adventureworks.bak'
WITH CHECKSUM
--파일1에서백업세트를사용할수있습니다.
```

주의

일반적으로 백업을 생성하거나 검증할 때 Checksum은 성능에 영향을 미칠 수 있습니다. 백업과 복원의 명령어를 사용할 때 기본값은 NO_CHECKSUM입니다.

미러백업

2005에서는 백업미디어를 미러링 함으로써 데이터의 잠재적인 미디어문제로부터의 복구 능력을 향상시킬 수 있습니다. 이 방법은 동일한 미디어종류에 백업을 수행하게 됩니다. 이런 백업을 통해서 데이터의 손실에 대한 위험성을 줄일 수 있습니다.

[따라하기] 미러백업 만들기

```
BACKUP DATABASE AdventureWorks
TO disk =' c:₩temp₩adventurworks.bak'
MIRROR TO disk =' d:₩temp₩adventureworks.bak'
WITH FORMAT
```

백업할 때는 지정한 모든 백업미디어에 접근이 가능해야 하지만, 복원 시에는 하나의 미디어에만 접근해도 복원을 성공할 수 있습니다. 즉 위의 경우 C:₩에 있는 백업원본이나 D:₩에 있는 미러링된 백업사본 모두 복원이 가능합니다. 주의할 점은 미러링된 백업사본에서 복원을 할 경우는 MOVE 옵션을 이용하여야 합니다.

[따라하기] 미러백업사본을 이용하여 복원하기

```
-- 미러링된 백업사본에서 복원하는 경우
RESTORE DATABASE adventureworks
FROM DISK = 'd:₩temp₩adventureworks.bak'  --미러백업된 사본
WITH MOVE 'AdventureWorks_Data' TO 'C:₩Program Files₩Microsoft SQL Server\MSSQL.1\MSSQL₩Data\AdventureWorks_Data.mdf'
, MOVE 'AdventureWorks_Log' TO 'C:₩Program Files₩Microsoft SQL Server\MSSQL.1₩MSSQL₩Data₩AdventureWorks_Log.ldf'
, Checksum
```

백업기록과 백업에 관련된 자세한 내용을 알고 싶다면 msdb 데이터베이스에 있는 다음의 테이블을 살펴보면 됩니다.

```
msdb.dbo.backupset
msdb.dbo.backupmediaset
msdb.dbo.backupmediafamily
```

```
msdb.dbo.backupfilegroup
msdb.dbo.backupfile
```

미러백업을 할 때는 미러백업에 사용되는 모든 미디어를 필요로 합니다. 반면 복원작업에 있어서는 복원작업 당시 하나의 미디어만 있으면 됩니다. 하지만 복원할 때 에러가 발생되었을 경우 다른 미러 미디어가 있으면 문제를 신속히 해결할 수 있습니다.

RESTORE 와 RESTORE VERIFYONLY 명령어에는 손상된 미디어에 해당하는 다른 미디어를 이용하여 복원시키는 옵션을 지원합니다.

주의
미러백업의 기능을 사용하기 위해서는 미디어내의 모든 장치는 동일해야 합니다. (예를 들면, 테이프 드라이브는 같은 회사의 같은 모델 번호여야 합니다.)

에러와 독립적으로 복원하기

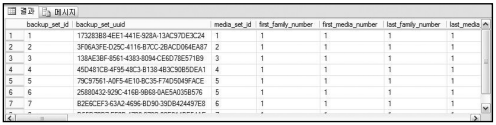
2005에서는 복원 중에 오류가 발생해도 끝까지 복원을 마칠 수 있도록 새로운 옵션을 추가하였습니다. 이전 버전에서는 백업자료에 문제가 있다면 복원을 할 수 없었습니다.
2005에서는 새로운 옵션 CONTINUE_AFTER_ERROR이 지원되며 아래의 예제처럼 사용합니다. 이 옵션을 사용하면 에러를 로그에 남기며 계속해서 복원을 진행합니다.

[따라하기] 에러상황에서도 복원가능하도록 하기

```
RESTORE DATABASE AdventureWorks
FROM  disk =' c:₩temp₩adventureworks.bak'
WITH CONTINUE_AFTER_ERROR
```

백업미디어 안에 오류가 있다는 표시를 지정하고 msdb 데이터베이스의 suspect_pages 테이블에 들어 있는 페이지를 추적합니다. 그리고, SQL Server 오류에 로그를 남깁니다. 또한, 백업시 오류가 발생되면 msdb.backupset 테이블의 is_damaged 열에 1값을 남깁니다. 그리고, 백업이 종료되면 백업이 되었지만 오류가 있다고 메시지를 남깁니다.

msdb..backupset의 결과



	backup_set_id	backup_set_uuid	media_set_id	first_family_number	first_media_number	last_family_number	last_media
1	1	1732E3B8-4EE1-441E-928A-13AC97DE3C24	1	1	1	1	1
2	2	3F06A3FE-025C-4116-B7CC-2BACD064EA87	2	1	1	1	1
3	3	138AE38F-8561-4383-8094-CE6D78E571B9	3	1	1	1	1
4	4	45D481CB-4F95-48C3-B138-4B3C90B5DEA1	4	1	1	1	1
5	5	79C57561-A0F5-4E10-8C35-F74D5049FACE	5	1	1	1	1
6	6	25880432-929C-416B-9B68-0AE5A035B576	5	1	1	1	1
7	7	B2E6CEF3-63A2-4696-BD90-39D8424497E8	6	1	1	1	1

주의

이 옵션을 사용한다 하여 모든 백업데이터가 복원되는 것은 아닙니다. 에러의 종류에 따라서 이러한 복원은 성공 또는 실패할 수 있습니다. 만약 체크섬의 검증단계에서 오류가 있다면 오류가 없는 다른 데이터에 대한 복원은 성공할 수 있습니다. 하지만 저장된 미디어 즉, 테이프드라이버나 디스크의 오류가 있다면 복원은 실패하게 됩니다. 에러가 있는 상태에서 복원된 데이터베이스는 의심스러운 데이터베이스(Suspect)라는 표시를 하고 수작업을 통해서 어떤 에러가 있었고 데이터베이스에 무슨 영향을 줄지에 대해서 체크해볼 수 있도록 합니다.

다중인스턴스의 변화

2000에서는 에디션과 상관없이 서버당 16개의 인스턴스를 생성할 수 있었습니다. 그러나, 2005 엔터프라이즈 에디션에서는 서버당 50개 인스턴스를 지원합니다. 그 외 다른 에디션에서는 16개까지 지원합니다. 또한 달라진 것은 한 서버에 여러 개의 에디션의 인스턴스를 구성할 수 있습니다. 예를 들면 한 컴퓨터에 3개의 엔터프라이즈 에디션과 다른 여러 개의 스탠다드 에디션의 인스턴스를 구성할 수 있습니다.

복원의 새로운 점

온라인 복원

SQL Server 2005 엔터프라이즈 에디션에는 백업으로부터 온라인상에서 복원할 수 있는 새로운 기술을 장착했습니다. 이러한 기능은 2000에서는 모든 파일이 복원이 되어야 서비스가 되었던 것에 비해 서비스 중지시간을 줄일 수 있습니다. (단 전체복구 모델과 대량복구모델에 대해서만 가능합니다.)

■ 가능환경

1. 엔터프라이즈 에디션과 디벨로퍼 에디션에서만 지원합니다.
2. 다수의 파일그룹을 가지고 있는 데이터베이스에 해당됩니다.
3. Secondary 파일그룹이 복원되는 동안만 접근이 불가능합니다.
4. 파일그룹이 복원될 때, 모든 파일그룹이 복원이 되어야만 데이터베이스의 일관성이 유지됩니다.

2000와 마찬가지로 2005에서 오프라인 복원작업을 그대로 하실 수 있습니다. 또한 SQL Server Management Studio를 이용해서 데이터베이스와 로그를 복원하실 수 있습니다.

2005에서는 2000 및 7.0에서 백업사본에 대해서는 오프라인 복원만 지원합니다. 온라인 복원을 수행하기 위해서는 2005에서 백업된 소스를 이용해야 합니다.

2005의 RESTORE 명령어는 새로운 옵션 RESTRICTED_USER을 포함하고 있습니다. 이 옵션은 복원된 데이터베이스에 db_owner, dbcreator, sysadmin 역할을 가진 구성원에게 제한적으로 접근을 하게 합니다.

온라인 복원은 두 가지 레벨로 지원됩니다.

■ Page Online Restore(페이지단위 온라인 복원)
온라인 복원의 특징은 문제가 있는 페이지를 고립시켜서 이 문제된 페이지가 다른 정상적인 데이터베이스를 복원하는데 있어 그 영향을 거의 주지 않는 다는 것입니다. 손상된 페이지는 데이터베이스로그에 남겨지게 됩니다.

손상된 페이지에 접근을 하면 에러가 발생 할 것이며 에러로그에 그 결과가 남게 될 것입니다. 이렇게 얻어진 정보를 통해서 페이지의 위치를 알아내고 유용한 백업자료에서 복원을 하게 됩니다.

[페이지 복원 절차]

```
--1) 손상된 페이지의 ID를 찾아냅니다.
SELECT * FROM msdb..suspect_page_table

--2) 트랜잭션로그의 진행중인 부분을 백업합니다.
BACKUP LOG AdventureWorks
TO DISK = 'Active_Log.bak' WITH NO_TRUNCATE

--3) 가장 최근의 전체 백업이나 차등백업으로부터 손상된 페이지를 복원합니다.
RESTORE DATABASE AdventureWorks PAGE = '3:3241'
FROM DISK = '...' WITH NORECOVERY

--4) 로그백업을 수행하고 데이터베이스를 복원합니다.
RESTORE LOG AdventureWorks
FROM DISK = '...' WITH NORECOVERY
.
.
.
RESTORE LOG AdventureWorks
FROM DISK = '...' WITH RECOVERY
```

```
--5) 수행중인 로그백업을 사용해서 복원합니다.
RESTORE LOG AdventureWorks
FROM DISK = 'Active_Log.bak'
WITH RECOVERY
```

■ 파일 단위 온라인 복원

데이터베이스의 파일그룹에서 어느 한 파일이라도 복원 중에 있다면 전체 파일 그룹은 오프라인 상태에 있게 됩니다. 그래서 주 파일그룹에 있는 하나의 파일이라도 복원 중이면 전체 데이터베이스는 오프라인상태가 됩니다. 하지만 주파일 그룹의 복구가 끝나면 데이터베이스는 온라인상태가 되며 각각의 이차파일그룹(Secondary filegroup)이 복원이 되면 해당 파일 그룹은 자동으로 온라인이 됩니다.

[파일단위 복원 절차]

```
--1) 활성중인 트랜잭션로그 백업하기
BACKUP LOG AdventureWorks
TO DISK = 'Active_Log.bak' WITH NO_TRUNCATE

--2) 손상된 파일 복원하기
RESTORE DATABASE AdventureWorks FILE = 'Logical_Damaged_Filename'
FROM DISK = '...' WITH NORECOVERY

--3) 트랜잭션 로그 복원하기
RESTORE LOG AdventureWorks
FROM DISK = '...' WITH NORECOVERY

--4) 활성중인 트랜잭션 로그를 복원하고 해당 데이터베이스 복구하기.
RESTORE LOG AdventureWorks
FROM DISK = 'Active_Log.bak'
WITH RECOVERY
```

Piecemeal 복원

2005 엔터프라이즈 에디션에서는 Piecemeal 복원을 지원합니다. 이 복원은 파일그룹단위로 복원을 진행합니다. 이때 복원되는 그룹단위로 온라인 상태로 됩니다. 이전 버전에서는 모든 파일 그룹이 복원이 되기 때문에 우선적으로 온라인 상태가 되어야 하는 경우 복원시간이 많이 소요되었습니다. 예를 들어 파일그룹하나에 문제가 있는 경우 다른 파일그룹에까지 영향을 미치게 되어 복원에 문제가 발생할 수 있습니다.

2005에서 이 Piecemeal 복원을 이용하면 중요도 높은 파일 그룹부터 복원하고 서비스를 할 수 있게 됩니다.

[따라하기] Piecemeal 복원

```
USE master
GO
DROP  DATABASE piecemealTest
GO
-- 테스트용데이터베이스생성
CREATE DATABASE [piecemealTest] ON  PRIMARY
( NAME = N' piecemealTest' , FILENAME = N' c:₩temp₩piecemealTest.mdf'  ,
SIZE = 3MB , FILEGROWTH = 10%),
 FILEGROUP [second]
( NAME = N' second' , FILENAME = N' c:₩temp₩second.ndf'  , SIZE = 3MB ,
FILEGROWTH = 10%),
 FILEGROUP [third]
( NAME = N' third' , FILENAME = N' c:₩temp₩third.ndf'  , SIZE = 3MB ,
FILEGROWTH = 10%)
 LOG ON
( NAME = N' piecemealTest_log' , FILENAME = N' c:₩ temp
₩piecemealTest_log.ldf'  , SIZE = 1024KB , FILEGROWTH = 1024KB )
GO
```

```
USE piecemealTest
GO

-- 데이터베이스의기본파일그룹을변경합니다.
ALTER DATABASE piecemealTest
modify FILEGROUP second default
GO

-- od라는테이블을northwind 데이터베이스에있는order details 테이블을이용하여만
듭니다.
SELECT TOP  100
productid, quantity
INTO od
FROM northwind..biGOd
GO

-- 주파일그룹에테이블odonPrimary를생성합니다.
CREATE table odonPrimary
(productID          int
,quantity           int
) on [primary]
GO

-- 세번째파일그룹에테이블od3rd를생성합니다.
CREATE table od3rd
(productID          int
,quantity           int
) on THIRD
GO
```

```
-- 위에서생성한테이블에값을넣습니다.
INSERT INTO od3rd
        SELECT productID, quantity FROM od
INSERT INTO odOnPrimary
        SELECT productID, quantity FROM od
GO

-- 백업장치dumpPiece와dumpPieceLog를생성합니다.
exec sp_addumpdevice' disk' ,' dumpPiece' ,' c:₩temp₩dumpPiece.bak'
exec sp_addumpdevice' disk' ,' dumpPieceLog' ,' c:₩temp₩dumpPieceLog.bak'
GO
-- 생성확인을합니다.
sp_helpdevice

SELECT TOP  100 * FROM od

-- od 테이블(second file group)에값을변경합니다.
update od  -- second group
set quantity=2000
where productID=11
GO

-- od3rd 테이블(third file group)에값을변경합니다.
update od3rd  --third group
set quantity=3000
where productID=11
GO

-- odOnPrimary 테이블(Primary file group)에값을변경합니다.
```

```
update odOnPrimary   -- Primary group
set quantity=1000
where productID=11
GO

-- 전체백업을수행합니다.
BACKUP DATABASE piecemealTest to dumpPiece WITH init
GO
-- 로그백업을수행합니다.
BACKUP log piecemealTest to dumpPiecelog WITH init
GO

USE master
GO

RESTORE DATABASE piecemealTest
FILEGROUP=' Primary'   --FILEGROUP=' Primary'
FROM dumpPiece WITH partial,norecovery,replace

RESTORE DATABASE piecemealTest
FROM dumpPiece WITH partial,norecovery,replace

RESTORE log piecemealTest FROM dumpPieceLog
GO

USE piecemealTest
GO
SELECT * FROM odOnPrimary
GO
```

```
SELECT * FROM od
GO
SELECT * FROM od3rd
GO
--결과메시지
--복원은Primary 에서이루어졌기때문에
--Second에있는od 테이블과third에있는od3rd는온라인상태가 되지못했다.
(100개행적용됨)
메시지8653, 수준16, 상태1, 줄1
테이블' od' 이(가) 온라인상태가아닌파일그룹에있어서쿼리프로세서에서이테이블또
는뷰에대한계획을생성할수없습니다.

메시지8653, 수준16, 상태1, 줄1
테이블' od3rd' 이(가) 온라인상태가아닌파일그룹에있어서쿼리프로세서에서이테이
블또는뷰에대한계획을생성할수없습니다.
```

SQL 2000에서는 위까지의 과정을 수행하면 모든 파일이 복구 됩니다.

```
USE master
GO
BACKUP log piecemealTest to disk = 'c:₩temp₩log1' WITH no_truncate, init


-- 이번에는Second 그룹에있는파일을복원합니다.
RESTORE DATABASE piecemealTest
FILEGROUP=' SECOND'
FROM dumpPiece
WITH partial,norecovery
GO
```

```
RESTORE log piecemealTest FROM dumpPieceLog

USE piecemealTest
GO
SELECT * FROM odOnPrimary
GO
SELECT * FROM od
GO
SELECT * FROM od3rd
GO
-- 결과에서Primary 그룹과Second 그룹은복원이되었고
-- third 그룹은 아직접근할수없다.
(100개행적용됨)

(100개행적용됨)
메시지8653, 수준16, 상태1, 줄1
테이블' od3rd' 이(가) 온라인상태가아닌파일그룹에있어서쿼리프로세서에서이테이
블또는뷰에대한계획을생성할수없습니다.
```

응급데이터베이스 상태

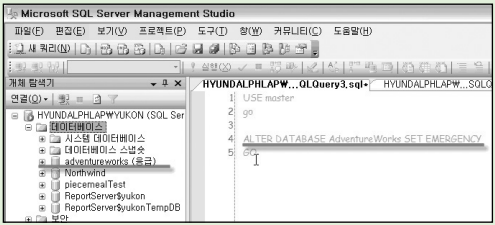
백업매체의 문제로 인한 복원의 실패나 문제가 있는 데이터베이스를 복구하려 할 때 ALTER DATABASE 명령문에 EMERGENCY 문을 추가해서 데이터베이스를 응급상태로 전환 시킬 수 있습니다.

데이터베이스가 주의상태로 표시된다면 데이터베이스의 상태를 응급으로 변경할 수 있습니다. 이렇게 변경을 하면 데이터베이스는 읽기전용으로 바꾸거나 데이터베이스를 분리할 수 있습니다.

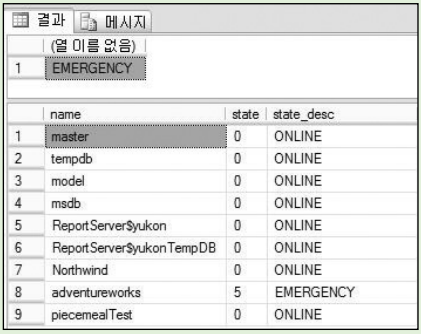
응급상태로 전환을 하게되면 데이터베이스는 단독접근(Single user)이 되고 sysadmin 서버 역할을 가진 로그인만 접속 가능합니다. 그리고, 읽기전용데이터베이스가 됩니다. 이런 상태에서 데이터베이스의 복구를 가능하게 합니다.

[따라하기] AdventureWorks 데이터베이스 응급상태로 전환하기

```
USE master
go

ALTER DATABASE AdventureWorks
SET EMERGENCY
GO
```




```
-- 데이터베이스의 특성을 살펴봅니다.
SELECT DATABASEPROPERTYEX( 'AdventureWorks' ,  'Status' )
SELECT name, state, state_desc FROM sys.databases
```



결과 메시지

	(열 이름 없음)
1	EMERGENCY

	name	state	state_desc
1	master	0	ONLINE
2	tempdb	0	ONLINE
3	model	0	ONLINE
4	msdb	0	ONLINE
5	ReportServer$yukon	0	ONLINE
6	ReportServer$yukonTempDB	0	ONLINE
7	Northwind	0	ONLINE
8	adventureworks	5	EMERGENCY
9	piecemealTest	0	ONLINE

```
USE AdventureWorks
GO

SELECT * FROM sales.Store

BEGIN TRAN
Go

--아래의문장은실패합니다.
UPDATE sales.Store SET name = 'X' +name
go

ROLLBACK
```

메시지3908, 수준16, 상태1, 줄3
데이터베이스' adventureworks' 은(는) 복구무시모드이므로BEGIN TRANSACTION을 실행할 수 없습니다.
문이 종료되었습니다.

```
메시지3903, 수준16, 상태1, 줄2

--아래의 쿼리도 실패합니다.
BACKUP DATABASE AdventureWorks TO DISK=' c:₩temp₩AWBACK.bak'
GO

메시지3033, 수준16, 상태0, 줄3
응급모드에서 열린 데이터베이스에서는BACKUP DATABASE를 사용할 수 없습니다.
메시지3013, 수준16, 상태1, 줄3
BACKUP DATABASE이(가) 비정상적으로 종료됩니다.


데이터베이스의 상태를 정상적으로 복구합니다.
use master
go
ALTER DATABASE AdventureWorks SET ONLINE
GO
```

ATTACH_REBUILD_LOG 절

OLTP환경에서 대용량 데이터베이스를 가지고 있다고 생각해봅시다. 이 데이터베이스를 다른 서버에 읽기위주의 용도로 옮긴다고 생각을 하면 단지 데이터파일만 옮기면 됩니다. 그러기 위해서는 서비스를 중단시키고 .mdf 파일과 .ndf 파일을 복사해서 옮깁니다. 그 후 CREATE DATABASE 명령어에서 ATTACH_REBUILD_LOG 절을 추가하여 수행합니다. 이렇게 하면 SQL Server는 데이터베이스를 연결하면서 새롭게 로그를 생성합니다.

완벽하게 수행하기 위해서는 이 작업이 끝난 뒤 바로 전체 백업을 수행해야 합니다. 이전 버전의 sp_attach_single_file_db 저장 프로시저는 호환성을 위해서만 사용됩니다.

2005에서는CREATE DATABASE 문장에 ATTACH_REBUILD_LOG을 사용할 것을 권장합니다.

문법은 그 예제는 다음과 같습니다.

```
CREATE DATABASE database_name
    ON 〈filespec〉 [ ,...n ]
    FOR { ATTACH [ WITH 〈service_broker_option〉 ]
        | ATTACH_REBUILD_LOG }

예제)
CREATE DATABASE AdventureWorks
FOR ATTACH_REBUILD_LOG
```

데이터베이스 스냅숏에는 ATTACH_REBUILD_LOG옵션을 사용할 수 없습니다.

데이터베이스 미러링

데이터베이스 미러링은 주 서버(Principal Server)와 미러서버(Mirror Server)를 이용한 고가용성 기술입니다. 주 서버에서 일어나는 트랜잭션에 대해서 미러서버에 다시 한번 수행을 하게 됩니다. 감시서버(Witness Server)는 미러 세션을 주목하고 있으면서 주 서버에 문제가 발생하면 미러서버를 주 서버로 그 역할을 변경하고 서비스를 이어갑니다.

(이 서비스는 서비스팩 1 이전에는 시작시 - t1400옵션을 추가하여 sqlserver서비스를 시작하여야 합니다.)

데이터베이스 미러링이 가능하도록 서비스 시작하기

■ 명령창에서 수행하기

net start mssqlserver - T1400

■ SQL Server 구성관리자에서 추가하기

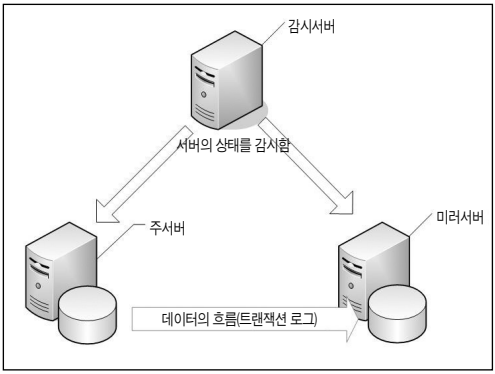
데이터베이스 미러링의 장점

대기서버에 완벽한 데이터베이스 복사본을 이용하여 데이터베이스의 이중화를 유지합니다. 또한 클러스터링에 비해 약 3초 미만의 시간 내에 서비스가 이어집니다. 그리고, 특별한 장비의 요구사항이 없습니다. 그렇기 때문에 네트워크에 연결되어있는 일반적인 저 사양의 장비로 훌륭하게 데이터가용성을 구현 할 수 있습니다. 또한 구현도 쉽습니다. 그리고, 문제가 발생하면 자동으로 서버의 역할을 교대하여 서비스를 유지합니다. 이럴 때 데이터베이스서버에 연결된 클라이언트 측에서는 어떤 서버를 사용하는지 모르는 상태에서 서버의 변경이 가능합니다.

■ 서버역할

자동적인 데이터베이스미러링을 구현하기 위해서는 다음과 같은 역할을 가진 서버가 있어야 합니다.

역할	설명
Principal Server (주 서버)	데이터베이스를 가지고 서비스를 운영하는 서버
Mirror Server(미러서버)	주 서버의 데이터베이스의 복사본을 유지하는 서버
Witness Server(감시서버)	서버간의 연결과 주 서버의 문제 발생시 자동적으로 문제점을 해결하는 서버



■ 구성방법

데이터베이스미러링은 상황에 맞게 3가지 구성방법이 있습니다.

1. 감시서버를 포함한 동기화 미러링(Synchronous mirroring)

   특징:

   데이터의 손실이 없고 자동으로 장애조치를 지원합니다.

   주 서버에서 일어난 트랜잭션이 미러서버에 기록이 정확하게 이루어 져야만 주 서버에 커밋이 됩니다. 데이터의 손실이 없는 장점에 비해 성능에 영향을 줄 수 있습니다.

2. 감시서버를 포함한 비동기화 미러링방식(Asynchronous mirroring)

   특징:

   동기화 미러링방식과는 달리 주 서버에서 트랜잭션이 바로 커밋됩니다. 이런 이유로 동기화 방식에 비해 성능이 좋습니다. 이 방식은 주 서버와 미러서버간의 거리가 멀리 떨어지거나 서버간의 지연이 있는 경우 고려할 수 있는 방식입니다.

3. 감시서버를 포함하지 않은 동기 미러링

특징:

이 구성은 데이터의 정확성을 우선으로 하며 서버의 서비스중지 시간을 어느 정도 인정하며 잠재적인 성능 이슈를 견딜 수 있는 상황에서 유용합니다. 이 구성은 자동 장애조치기능이 구현되지 않습니다.

■ 구성방법개요

미러링을 구성하기 위해서는 2~3개의 서버 또는 인스턴스가 필요합니다. 구성의 순서는 다음과 같습니다.

1. 미러서버 구성
2. 미러데이터베이스 구성
3. 끝점(EndPoints) 구성
4. 미러세션 구성
5. 감시서버 구성

■ 구성방법 살펴보기

1. 미러서버 구성하기.

주 서버가 미러서버에 접속가능 해야하며 트러스터된 상태이어야 합니다. 권장하는 것은 같은 도메인상에 속해있는 것입니다.

한 서버에 여러 인스턴스를 이용하여 구성이 가능합니다. 하지만 이렇게 하면 성능상의 문제가 있기에 권장하지 않습니다.

2. 미러데이터베이스 구성하기

미러데이터베이스는 자동이 아닌 수동으로 구성됩니다. 두 데이터베이스(주 서버, 미러서버)는 같은 파일 구조를 가져야 합니다. 또한 전체복구모델로 선택되어 있어야 합니다.

미러데이터베이스 생성 즉시 주 데이터베이스의 전체 백업본을 미러서버에 With NORECOVERY 절을 이용하여 복원해야 합니다.

3. 미러링 구성을 위한 끝점(Endpoints) 구성하기

1) 미러서버에 파트너 끝점(Partner Endpoint) 구성하기

주 서버와의 통신을 위해 주 서버와 사용할 파트너 끝점이 필요합니다.

아래의 예제 코드를 이용하면 Mirror_EndP라는 파트너 끝점을 만듭니다. STATE 파라미터를 STARTED로 지정하게 되면 바로 활성화 됩니다. TCP 포트 5022는 미러링에서 기본적으로 사용하는 포트번호입니다.

```
CREATE ENDPOINT Mirror_EndP
STATE=STARTED
AS TCP(LISTENER_PORT=5022)
FOR DATABASE_MIRRORING (ROLE=PARTNER)
```

2) 주 서버에 파트너 끝점 구성하기

미러서버에서 보여준 것처럼, 주서버는 미러링 서비스를 제공하는 SQL서버 인스턴스와 통신을 할 파트너 끝점이 필요합니다.

```
CREATE ENDPOINT Mirror_EndP
STATE=STARTED
AS TCP (LISTENER_PORT=5022)
FOR DATABASE_MIRRORING (ROLE=WITNESS)
```

3) 목격자 서버에 목격자 끝점 구성하기

시나리오에 미러링 끝점이 포함되어 있으면 목격자서비스를 제공하는 SQL서버 인스턴스상에 끝점이 필요합니다.

```
CREATE ENDPOINT Mirror_EndP
STATE=STARTED
AS TCP (LISTENER_PORT=5022)
FOR DATABASE_MIRRORING (ROLE=WITNESS)
```

4. 미러세션 설정하기
위에서 미러서버와 데이터베이스 준비되고 각 서버에 Endpoint가 생성이 되면 다음과 같이 미러세션을 설정합니다.

1) 미러서버에서 파트너쉽 생성하기
우선 미러서버에서 ALTER DATBASE 명령어를 수행합니다.

```
ALTER DATABASE AdventureWorks
SET PARTNER =  'TCP://Seoul:5022'
```

2) 주 서버에서 파트너쉽 생성하기

```
ALTER DATABASE AdventureWorks
SET PARTNER =  'TCP://Busan:5022'
```

5. 감시서버 구성하기
자동 장애조치를 구성하려면 목적서버를 구성해야 합니다. 목적서버는 주 서버 및 미러서버와는 다른 서버에 구성이 되어야 합니다. 그렇지만 하나의 목적서버는 여러 개의 미러세션을 담당할 수 있습니다.

```
ALTER DATABASE AdventureWorks
SET WITNESS =  'TCP://DAEGU:5022'
```

6. 감시서버 제거하기
미러서버가 제거되어도 미러링 세션은 유지 됩니다. 다만, 자동 장애조치는 불가능합니다. 감시서버를 제거하는 방법은 다음과 같습니다.

```
ALTER DATABASE AdventureWorks
SET WITNESS OFF
```

■ 데이터베이스미러링 모니터하기

2005에서는 데이터베이스미러링 세션을 모니터하기 위해서 성능모니터의 카운터, 프로필러의 이벤트, 시스템 카탈로그뷰(System Catalog View), 동적관리뷰(Dynamic Management View) 등을 이용합니다.

1) 성능모니터
데이터베이스 성능모니터링을 위해서 SQL Server:Database Mirroring 개체에 11개 카운터를 지원합니다.

2) 프로필러
데이터베이스 미러링의 상태가 변화할 때 데이터베이스 이벤트 클래스상의 데이터베이스 미러링 상태 변경 이벤트(Database Mirroring State Change Event)를 이용할 수 있습니다.

3) 시스템카탈로그 및 동적관리뷰

Sys,database_mirroring :
현재 인스턴스에 존재하고 있는 각각의 데이터베이스 행수를 기록합니다.
Sys,database_mirroring_endpoint :
현재의 끝점에 대한 정보를 보여줍니다.
Sys,database_mirroring_witnesses :
목격자 역할을 하는 서버에 대한 정보를 돌려줍니다.
Sys,db_database_mirroring_connection:
데이터베이스 미러링 구성에 포함된 네트워크정보를 보여주는 동적관리 뷰입니다.

■ 데이터베이스미러링 응용

OLAP환경의 읽기 위주의 쿼리를 미러서버에 있는 데이터베이스를 사용하는 경우 주 서버의 부하를 분산시킬 수 있습니다.

비매품

SQL Server 2005 관리자 가이드

- 저 자: 김 정 선, 성 대 중, 현 중 근
- 감 수: 정 원 혁
- Contents 관련문의: mjkim@feelanet.com

- 발 행: 한국마이크로소프트(유)
- 제 작: ㈜필라넷
- 초판 발행일: 2006년 4월 1일

Microsoft
SQL Server 2005
관리자 가이드

Microsoft
한국마이크로소프트(유)

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

SQL-200603-01