Microsoft

# Microsoft SQL Server 2005

# Oracle Migration Guide Book

## 저자의 글

Oracle에서 SQL Server로의 마이그레이션은 간단한 작업이라고 할 수 없습니다.
기존 환경에 대한 사전 분석을 통해서 평가하고 설계해야 하며 또한, 마이그레이션 시 발생할 수 있는 위험을 면밀하게 분석해야만 합니다. 뿐만 아니라, SQL Server와 Oracle에 대한 충분한 사전 지식을 가지고 있어야 합니다. 이에 대해서 마이크로소프트는 Oracle 마이그레이션을 좀더 효율적으로 진행할 수 있도록 마이그레이션 자동화 툴(SSMA)를 제공하고 있습니다. 본 가이드에서는 마이그레이션 방법론과 SSMA의 기능과 활용 방법 그리고, SQL Server와 Oracle의 차이점 및 Oracle과의 통합 운영에 대해서 기본적인 지식을 제공하고 있습니다. 지면 관계상 많은 내용을 담지는 못하였지만 Oracle에서 SQL Server로 마이그레이션을 준비하는 IT 선구자 여러분들에게 유익한 자료가 되기를 바랍니다.

## 저자 약력

이종인 온디멘드 수석 컨설턴트 / 다우 교육원 SQL Server 전임강사

- FMG 수석 컨설턴트
- 이룬인포텍 DB팀장
- 산업은행 여신지원시스템 DBA
- 동서증권 대리
- 산업은행 / LG,Philips LCD / 한국문화진흥 등 다수의 SQL Server 튜닝 컨설팅 및 Oracle 마이그레이션 컨설팅
- MCSE+Internet, MCDBA, MCT, MCITP, OCP, HPCP-Master ASE, HPCP-ASE+StorageWorks

# 목차

# 1. 마이그레이션 개요

급격하게 변화하는 사회에 맞추어 IT 산업도 빠르게 변화하고 있습니다. e-비즈니스는 24시간 가동을 요구하고 다양하게 발생하는 기업의 인수 및 합병은 IT 인프라의 통합을 요구합니다. 따라서, 비즈니스 환경은 급격하게 변화하는 업무 요건에 따라서 유연하고 신속한 IT의 지원을 요구하고 있습니다. 또한, 과거에 그때그때 요구 사항에 맞추어 업무와 관련된 응용 프로그램을 난 개발하다 보니 현재의 전산 시스템이 마치 잡화상처럼 메인프레임, UNIX, LINUX, Windows와 SQL Server, Oracle, DB2 등이 혼재되어 있게 됩니다. 당연히 유지 보수는 힘들어 지고 현업의 요구 사항을 IT가 신속하게 부합하지 못하는 상황이 발생하게 됩니다. 이와 같은 요소를 극복하기 위한 최선의 수단 가운데 하나가 마이그레이션입니다. 복잡하게 구성되어있는 IT 인프라를 통합함으로써 신속하고 용이한 개발환경을 제공하고 효율적인 유지 보수가 가능해 집니다.

이와같은 마이그레이션이 갖는 비즈니스 요구 사항에 대한 효과를 요약하면 다음과 같습니다.

**1) IT 인프라의 총 소유비용(TCO) 감소**

① 하드웨어 비용 감소 : IT 환경의 서버 장비나 스토리지, 라우터, 스위치 장비의 비용 감소
② 소프트웨어 비용 감소 : 운영체제, DBMS, 개발 도구 및 사용자 및 관리 응용 프로그램 등의 비용 감소
③ 시스템 유지 보수 비용 감소 : 하드웨어 및 소프트웨어 유지 보수 비용 및 시스템 보안 유지를 위한 비용 감소
④ 전산 담당자 유지 비용 감소 : 전산 담당자의 기술 수준 유지를 위한 교육 비용 감소

**2) 투자 대비 효과를 극대화한 생산성 향상**

① 기업 혁신 및 업무 개선에 유연한 개발을 통한 생산성 향상 : 시스템 통합 및 유지 보수를 위한 신속하고 효율적인 개발 환경 제공
② 시스템 안전성 향상을 통한 생산성 향상 : 시스템의 다운타임을 줄임으로써 업무 생산성 향상

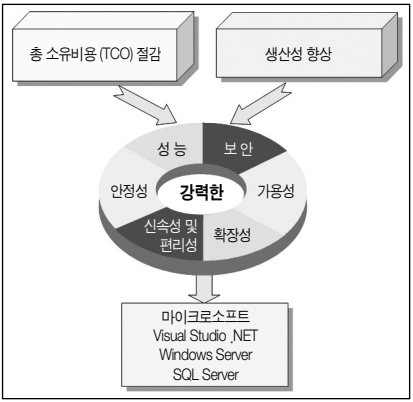
③ 성능 향상을 통한 생산성 향상
④ 자산 투자 회수 기간 단축 : 저렴한 소프트웨어 구입 및 유지 보수 비용, 생산성 향상을 통한 투자 비용 회수 기간 단축

이제 실제로 자신의 환경에서 마이그레이션에 소요되는 하드웨어와 소프트웨어 비용, 그리고 마이그레이션 프로젝트에 소요되는 비용을 따져봐야 합니다. 그리고, 마이그레이션이 완료된 이후에 절감되는 비용을 종합해보고, 이 비용과 마이그레이션을 통해서 IT 인프라를 통합하지 않았을 때 발생하는 기회 비용을 비교해야 합니다. 관리회계와 예산기획이 효율적인 회사라면 그리 어렵지 않게 확인할 수 있을 것입니다.

이제 마이그레이션의 필요성이 인지되었다면 "과연 어떤 제품으로 통합할 것인가?" 하는 문제가 남게 됩니다.

마이크로소프트는 신속하고 효율적인 개발이 가능하며 통합된 환경을 제공하는 Visual Studio .NET을 개발 도구로, 보안과 성능 그리고 확장성 면에서도 뛰어난 Windows Server와 SQL Server를 플랫폼과 데이터베이스 관리시스템으로 제공하고 있습니다. 특히, SQL Server 2005는 강력한 보안, 확장성 및 가용성, 안정성 뿐만 아니라 응용 프로그램을 더욱 쉽게 구축, 배치 및 관리할 수 있도록 돕는 차세대 데이터 관리 및 분석 솔루션으로 강화되었습니다.

뿐만 아니라 우리와 친숙한 마이크로소프트 오피스와 같은 클라이언트 응용 프로그램에서부터 Share Point Portal Server, BizTalk Server 등의 Server 제품군을 IT 인프라의 총 소유 비용(TCO) 절감과 생산성 향상을 위해서 제공하고 있습니다.


총 소유비용 (TCO) 절감
생산성 향상
성 능
보 안
안정성
강력한
가용성
신속성 및
편리성
확장성
마이크로소프트
Visual Studio ,NET
Windows Server
SQL Server

## 2. SQL Server 2005 소개

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

SQL Server 2005를 통해서 각 기업에서는 데이터를 기반으로 신속하게 의사결정을 내릴 수 있고, 개발 인력의 생산성과 유연성을 향상시키며, IT 부분의 총 소유비용을 절감하고, 지속적으로 증가하는 비즈니스 요구 사항을 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 응용 프로그램의 구축, 배치 및 관리
- 데이터베이스 응용 프로그램 구축, 배치 및 관리의 복잡성을 해소함으로써 IT 생산성 극대화
- 다수의 플랫폼, 응용 프로그램 및 장치 간 데이터 공유를 통해서 내부 및 외부 시스템에 더욱 쉽게 연결
- 성능, 가용성, 확장성, 보안의 침해 없이 총 소유비용 통제

## SQL Server 2005 도입 효과

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

| 도입 효과 | 설명 |
|---|---|
| 데이터 자산 활용 | SQL Server 2005는 업무용 및 분석 응용 프로그램을 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 System과 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 응용 프로그램의 개발, 구축 및 관리 작업을 보다 용이하게 수행할 수 있습니다. |
| 저렴한 총 소유비용 (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 구성 요소

SQL Server 2005는 엔터프라이즈 데이터관리 및 BI(Business Intelligence) 응용 프로그램을 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 응용 프로그램을 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<br>(Relational Database Engine)<br>  | 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 데이터베이스 엔진입니다. |
| 분석 서비스<br>(SQL Server Analysis Services)<br>  | 온라인 분석 처리 응용 프로그램(OLAP) 과 데이터 마이닝을 지원하는 강력한 비즈니스 인텔리전스 솔루션입니다. |
| 통합서비스<br>(SQL Server Analysis Services)<br>  | 데이터를 가져오고 내보내는 솔루션으로서 데이터의 이동이 이루어질 때 변환과정을 수행합니다 |

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

| | |
|---|---|
| 닷넷 CLR 기반 서비스<br>(.NET Common Language Runtime)<br>  | 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<br>(Replication)<br>  | 한쪽 데이터베이스에서 다른 데이베이스로 데이터 및 데이터베이스 객체들을 복사 이동시키고 난 후 데이터베이스간의 일관성이 동기화되도록 하여줍니다. |
| 전체 텍스트 검색<br>(Full Text Search)<br>  | SQL Server 데이터베이스에 있는 텍스트로 저장된 키워드 기반 쿼리에 대한 빠르고 유연한 인덱스 검사를 가능하게 하여 줍니다 |

## SQL Server 2005의 새로운 기능

1) 데이터베이스 개발 측면

| 주요 특징 | 설명 |
|---|---|
| .NET Framework 호스팅. | SQL Server 2005를 통해 개발자들은 Microsoft Visual C# .NET 및 Microsoft Visual Basic .NET과 같은 친숙한 언어를 이용해 데이터베이스 개체를 만들 수 있습니다. 또 한 개발자는 사용자 정의 형식과 집계 등 두 가지 새로운 개체를 만들 수 있습니다. |
| XML 기술 | XML은 로컬 네트워크와 인터넷을 통해 여러 다양한 응용 프로그램에 데이터를 배포할 수 있도록 지원하는 중요한 표준입니다. SQL Server 2005는 XML 문서 저장소 및 쿼리를 기본적으로 지원합니다 |
| ADO.NET Version 2.0 | SQL Server 2005의 ADO.NET은 새로운 SQL 형식 지원에서 MARS(Multiple Active Result Sets)에 이르기까지 데이터 집합 액세스와 처리 기능을 개선하여 보다 뛰어난 확장성과 융통성을 발휘합니다. |
| 향상된 보안 기능 | SQL Server 2005의 새로운 보안 모델은 사용자들을 개체에서 분리하고 세분화된 액세스 권한을 부여하며 데이터 액세스에 대한 한층 강화된 제어 기능을 제공합니다. 또한, 모든 시스템 테이블이 뷰로 구현되기 때문에 데이터베이스 시스템 개체를 더욱 강력하게 제어할 수 있습니다. |
| Transact-SQL 기능 향상 | 확장 가능한 데이터베이스 응용 프로그램을 개발하기 위한 새로운 언어 기능들이 제공됩니다. 이들 향상된 기능에는 오류 처리, 새로운 재귀 쿼리 기능, 관계형 연산자 PIVOT, APPLY, ROW_NUMBER 및 기타 행 순위 지정 함수 등이 있습니다. |

| | |
|---|---|
| SQL Service Broker | SQL Service Broker는 대규모 업무용 응용 프로그램을 위한 비동기식 분산 응용 프로그램 프레임워크를 제공합니다. |
| Notification Services | 알림 서비스를 사용하여 주식 정보, 뉴스 구독, 소포 배달 알림, 항공권 가격 등과 같은 개인화된 최신 정보를 모든 장치에 전달하는 풍부한 알림 응용 프로그램을 작성할 수 있습니다. |
| 웹 서비스 | 개발자들은 SQL Server 2005를 통해 데이터베이스 계층에서 웹 서비스를 개발함으로써 SQL Server를 웹 서비스 중심 응용 프로그램을 위해 새로운 유형의 데이터 액세스 기능을 제공하는 HTTP 수신기로 만들 수 있습니다. |
| 리포팅서비스 (Reporting Services) | SQL Server 2005를 이용한 Reporting Services에서는 Visual Studio 2005에 포함된 보고서 컨트롤을 제공합니다. 통합된 보고 컨트롤은 엔터프라이즈 응용 프로그램에 대한 향상된 보고 기능을 제공합니다. |
| 전체 텍스트 검색 향상 | SQL Server 2005는 풍부한 기능의 전체 텍스트 검색 응용 프로그램을 지원합니다. 카탈로그 작성 기능은 카탈로그로 작성할 대상을 지정하는 데 있어 한층 높은 융통성을 발휘할 수 있도록 향상되었습니다.<br>또한 쿼리 성능과 확장성이 대폭 향상되었으며 새로운 관리 도구를 통해 전체 텍스트 구현을 보다 심층적으로 파악할 수 있습니다. |

2) 데이터베이스 관리 측면

| 주요 특징 | 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 액세스 할 수 있기 때문에 SQL Server의 가용성을 높일 수 있습니다. |
| 온라인 인덱싱 작업 | 온라인 인덱스 옵션을 사용하여 인덱스 DDL(data Definition Language) 실행중에 기본 테이블이나 클러스터형 인덱스 데이터 및 모든 관련 인덱스를 동시에 수정(업데이트, 삭제 및 삽입)할 수 있습니다. 예를 들어 클러스터형 인덱스가 다시 작성되는 동안 관리자들은 지속적으로 기본 데이터를 업데이트하고 이 데이터에 대해서 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 강력한 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-SQL문을 실행하여 서버의 문제를 해결할 수 있습니다. |
| Oracle 게시자 복제 | 표준 스냅숏 복제와 트랜잭션 복제에 대해서 Oracle(8 버전 이상)이 게시자 역할을 수행할 수 있도록 지원합니다. 따라서, Oracle에서의 변경 사항을 SQL Server에서 손쉽게 받아 볼 수 있습니다. |
| Peer-to-Peer 복제 | Peer-to-Peer 네트워크 토폴로지와 유사한 구조로 계층 구조가 아닌 수평 구조의 복제를 구성하여 고 가용성과 수평 확장 능력을 제공합니다. |

## SQL Server 2005 에디션별 특징

| 에디션 | 이점 | 크기 | 주요 기능 |
|---|---|---|---|
| Express (무료) | 간단한 데이터 중심 응용 프로그램을 배우고 구축 및 배포하는 가장 빠른 방법 | 1개의 CPU<br>1GB RAM<br>4GB DB 크기 | **간단한 관리도구 간단한 보고**<br>복제 및 SSB 클라이언트 |
| Workgroup | 소규모 부서와 성장하는 기업을 위한 가장 합리적인 가격대의 사용하기 쉬운 데이터베이스 솔루션 | 2개의 CPU<br>3GB RAM | Management Studio 가져오기/내보내기 제한된 복제 게시 백업 로그 전달 |
| Standard | 중견 기업 및 대규모 부서를 위한 완벽한 데이터관리 및 분석 플랫폼 | 4개의 CPU<br>무제한 RAM | **클러스터링**<br>**데이터베이스 미러링**<br>기본적인 ETL Analysis Service가 지원되는 표준 OLAP 서버 Data Mining Reporting Service가 지원되는 표준 보고 완전 복제 및 SSB 게시 원시 32 및 64비트 에디션에서 사용가능 Itanium2 및 x64지원 |

| | | | |
|---|---|---|---|
| Enterpise | 비즈니스 크리티컬 엔터프라이즈 응용 프로그램을 위한 완전히 통합된 데이터 관리 및 분석 플랫폼 | **무제한 확장 및 파티셔닝** | **향상된 데이터베이스 미러링**<br>**완벽한 온라인 및 병렬 조작**<br>**데이터베이스 스냅샷**<br>전체 OLAP 및 Mining을 비롯한 향상된 분석도구<br>**사용자 정의 및 확장성이 뛰어난 임의(ad hoc) 보고 기능을 통한 향상된 보고 기능**<br>복잡한 데이터 라우팅 및 변환 기능을 통한 향상된 ETL |

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

## SQL Server 2005 관련 자료

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

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

- SQL Server 2005 관련 백서
  http://www.microsoft.com/korea/sql/prodinfo/White_paper.mspx

- Microsoft SQL Server TechCenter
  http://www.microsoft.com/korea/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/korea/sql/2005/productinfo/sql2005features.mspx

- 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
  http://lab.msdn.microsoft.com/express/sql/

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

본 모듈의 내용은 SQL Server 2005 관리자 가이드 및 개발자 가이드의 내용을 저자와 협의하여 수록하였습니다.

# 3. 마이그레이션 수행 방법론

앞의 모듈에서 마이그레이션의 목적을 살펴보았습니다. 이제 마이그레이션을 수행하기 위한 방법론을 살펴봅니다. 마이그레이션을 수행하기 위해서는 미리 분석해야 하는 사항들이 많이 있습니다. 마이그레이션을 어느 부분까지 할 것인지 범위를 지정해야 하며 해당 마이그레이션에 어떠한 위험요소가 있는 지도 분석해야 합니다. 또한 Oracle이 제공하는 기능과 동일한 기능이 SQL Server에 있는 지 여부도 점검해야 합니다. 이와 같은 분석을 통해서 적절한 전략을 수립해야 하며 수립된 전략의 절차에 따라서 마이그레이션을 수행해야 합니다. 따라서, 마이그레이션 프로젝트를 수행하기 위해서는 다음과 같은 단계에 따라서 효율적이고 안정된 마이그레이션을 수행할 수 있도록 합니다.

1) 마이그레이션 요구 사항 분석 및 마이그레이션 범위 결정
2) 기존 환경 진단 및 분석에 따른 마이그레이션 평가 및 평가 리포트 생성
3) 마이그레이션 위험 요소 및 기술 분석
4) 마이그레이션 방안 상세 설계를 통한 전략 수립
5) 스키마 개체 마이그레이션
6) 데이터 마이그레이션
7) PL/SQL 구문 마이그레이션
8) 응용 프로그램 마이그레이션
9) 마이그레이션 점검
10) 최적화 작업

## 마이그레이션 요구 사항 분석 및 마이그레이션 범위 결정

먼저, 마이그레이션 프로젝트의 요구 사항을 분석해야 합니다. 마이그레이션의 목적과 예산,프로젝트 기간 등의 요구 사항에 따라서 마이그레이션의 범위가 결정됩니다. 마이그레이션 프로젝트의 범위는 다음과 같을 구분할 수 있습니다.

① DBMS만 마이그레이션 하는 경우
② Web 서버 등 응용 프로그램만 마이그레이션 하는 경우
③ Web 서버 등 응용 프로그램과 DBMS를 마이그레이션 하는 경우
④ Web 서버 등 응용 프로그램 및 middleware (트랜잭션 모니터), DBMS 모두 마이그레이션 하는 경우
⑤ Web 서버 등 응용 프로그램을 .NET으로 새로 개발하며 DBMS를 마이그레이션하는 경우
⑥ 마이그레이션과 함께 추가 기능의 모듈을 개발하는 경우 등

## 기존 환경 진단 및 분석에 따른 마이그레이션 평가 및 평가 리포트 생성

마이그레이션의 요구 사항을 분석하고 그에 따른 범위가 결정되었다면 기존 시스템의 환경을 철저히 분석해야 합니다. 응용 프로그램은 어떤 개발 언어로 작성되었는지, 해당 응용 프로그램을 마이그레이션 하기 위해서 필요한 기술은 무엇인지, 기본 Oracle에서 보안 옵션이나 파티셔닝, 복제 기능 등을 사용하고 있는 지 여부와 마이그레이션 해야 하는 데이터베이스 스키마 개체의 개수나 복잡성, 패키지나 프로시저 등의 구문 행 수나 복잡성 등을 평가해야 합니다. 또한, 각 운영체제나 DBMS의 환경 설정 사항도 면밀히 분석해야 합니다. 그리고 나서, 이렇게 평가한 내용을 리포트로 작성해서 위험 요소를 분석하고 프로젝트에 소요 되는 인원과 기간을 산정하기 위한 자료로써 활용해야 합니다.

## 마이그레이션 위험 요소 및 기술 분석

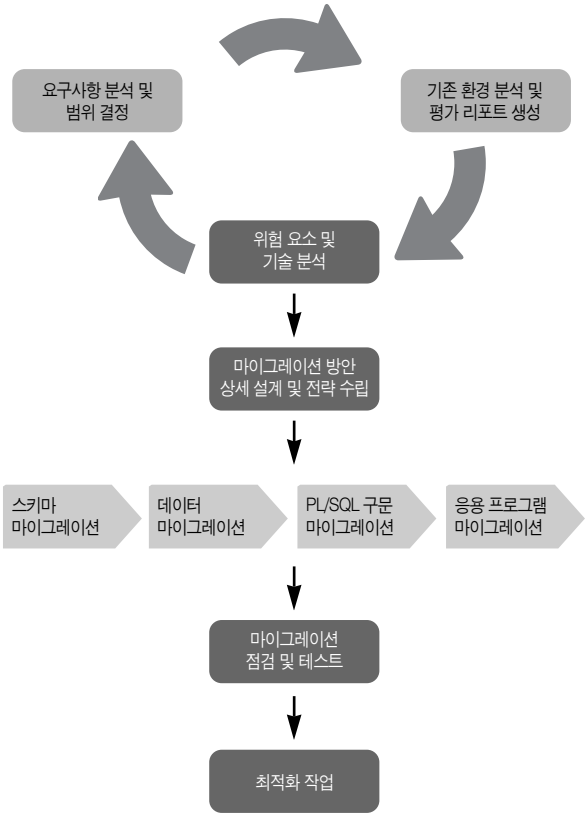
마이그레이션 평가 리포트가 작성되었으면 해당 리포트를 근거로 해당 마이그레이션 프로젝트의 위험 요소를 분석해야 합니다. 위험 요소는 다음과 같은 사항을 점검하고 대응 방안을 수립해야 합니다.

① 데이터베이스 디자인 변경
② 마이그레이션 대상 시스템의 동일 기능 지원 여부
③ 데이터 변환 시 한글 지원 문제
④ 기존 데이터의 정합성 검토
⑤ 업무 프로세스 변경

⑥ 새로운 기술의 도입
⑦ 대용량 데이터 마이그레이션 방법 및 오류 발생
⑧ 고가용성 기능 선택
⑨ 작업 소요 기간 초과
⑩ 프로젝트 수행 인원 교체 등

## 마이그레이션 방안 상세 설계에 따른 전략 수립

자, 이제 마이그레이션의 범위도 결정되었고, 마이그레이션 대상 시스템의 분석도 완료되었습니다. 또한, 마이그레이션 작업 시의 위험요소도 분석하고 그에 대한 대응 방안도 수립하였습니다. 그렇다면 이제 이제 마이그레이션 프로젝트를 세부 단계로 분류하고, 담당자를 지정하며, 각 단계 업무 항목마다 작성되어야 할 표준화된 산출물을 지정해야 합니다. 또한 각 작업 항목의 순서도 결정해야 합니다. 이와 같은 상세 설계와 전략의 수립은 정형화되고 표준화되어야 합니다. 앞에서 살펴본 1) 마이그레이션 요구 사항 분석 및 마이그레이션 범위 결정, 2) 기존 환경 진단 및 분석에 따른 마이그레이션 평가 및 평가 리포트 생성, 3) 마이그레이션 위험 요소 및 기술 분석 등 세 단계를 유기적으로 수행하여 빈틈없는 전략을 수립할 수 있도록 면밀히 검토합니다.


요구사항 분석 및
범위 결정
기존 환경 분석 및
평가 리포트 생성
위험 요소 및
기술 분석
마이그레이션 방안
상세 설계 및 전략 수립
스키마
마이그레이션
데이터
마이그레이션
PL/SQL 구문
마이그레이션
응용 프로그램
마이그레이션
마이그레이션
점검 및 테스트
최적화 작업

## 스키마 개체 마이그레이션

Oracle의 테이블, 뷰, 인덱스와 같은 개체를 SQL Server로 쉽게 마이그레이션 할 수 있습니다. 또한, SQL Server 2005는 테이블과 인덱스의 범위 기반 분할을 지원하므로 Oracle에서 파티션된 대용량의 테이블을 마이그레이션 할 수 있습니다. 데이터베이스의 스키마 개체에 관한 Oracle과 SQL Server의 차이점은 [모듈 6 SQL Server와 Oracle 차이 이해] 부분을 참조합니다.

## 데이터 마이그레이션

데이터 마이그레이션은 테이블의 개수와 데이터 량을 고려해서 마이그레이션 방법을 결정해야 합니다. 오류의 확률을 줄이기 위해서 가능한 동일한 방법으로 모든 테이블의 데이터를 마이그레이션하는 것이 효율적입니다. ETL 도구를 사용한 다면 ETL 도구를 사용하고 복제를 사용한다면 복제라는 방법으로 모든 테이블의 데이터를 마이그레이션해야 작업 수행 절차를 간소화하고 관리하기 용이하기 때문입니다. 다만, 일부 몇 개의 테이블이 대용량이거나 데이터의 정제, 또는 변환이 필요할 경우에는 해당 성격 별로 구분하여 처리하는 것을 고려해야 합니다. SQL Server는 SQL Server Integration Service라는 훌륭한 ETL 도구를 기본으로 제공하고 있습니다. 또한, Oracle 게시자 복제를 통해서 일정 기간 동안 변경되는 사항을 지속적으로 마이그레이션 할 수도 있습니다. 뿐만 아니라 대용량 데이터인 경우에는 BCP 유틸리티나 BULK INSERT 구문 등을 통해서 처리할 수도 있습니다.

## PL/SQL 구문 마이그레이션

PL/SQL 구문을 마이그레이션하기 위해서는 Oracle의 PL/SQL과 SQL Server의 T-SQL에 모두 익숙한 전문가가 필요합니다. 또한 PL/SQL로 작성된 패키지나 프로시저, 함수 등이 많은 경우에는 마이그레이션 전문 도구를 사용할 수 있습니다. 독립 소프트웨어 공급 업체가 제공하는 도구에 따라서 구문의 변환률 등이 다르지만 기본 적인 변환을 마이그레이션 도구를 통해서 자동화하고 마이그레이션 도구가 변환하지 못하는 부분을 전문 개발자가 수동으로 변환하게 되면 작업 시간을 상당히 단축할 수 있습니다. PL/SQL과 T-SQL의 차이점은 [모듈 6 SQL Server와 Oracle 차이 이해] 부분을 참조하시고, 마이그레이션 전문 도구는 [모듈 4 SSMAv2.0을 이용한 마이그레이션 자동화] 부분을 참조하시기 바랍니다.

## 응용 프로그램 마이그레이션

응용 프로그램을 마이그레이션하는 경우를 다음과 같이 구분할 수 있습니다.

① 기존 응용 프로그램 환경을 그대로 사용하고 연결 설정 부분만 변경하는 경우
② UNIX용 응용 프로그램을 Microsoft Windows Services For UNIX를 통해서 그대로 사용하는 경우
③ 응용 프로그램을 Win32API로 재 작성하는 경우
④ 응용 프로그램을 .NET으로 새로 작성하는 경우

Web 서버나 응용 프로그램을 마이그레이션 하는 경우에는 응용 프로그램 환경을 면밀하게 검토해야 합니다. Web 서버나 클라이언트 응용 프로그램이 UNIX 환경인 경우에는 연결 계층의 구성 부분도 면밀히 검토해야 하며 응용 프로그램이 작성된 개발 언어에 따라서 마이그레이션 시의 위험 요소를 면밀히 검토해야 합니다.

## 마이그레이션 점검 및 테스트

마이그레이션은 마이그레이션 범위에 따라서 다음과 같은 사항에 대해서 수행해야 합니다.

① 데이터베이스 스키마 개체 개수 점검
② 데이터 행수 및 정합성 점검
③ 프로시저 및 함수, 트리거 등의 동일한 결과값 반환 여부 점검
④ 응용 프로그램의 오류 여부 점검
⑤ 백업 및 보안 정책 정합성 점검
⑥ 신규 시스템 환경 설정 점검

## 최적화 작업

전문가의 손길이 각별히 필요한 단계입니다. Oracle은 Oracle이고 SQL Server는 SQL Server입니다. Oracle의 PL/SQL로 작성된 구문을 동일한 결과가 반환되거나 동일하게 처리하도록 T-SQL로 작성했다고 해서 마이그레이션이 완료된 것이 아닙니다.
더욱더 SQL Server에 맞게 T-SQL 구문을 효율적으로 작성해야 하고 적절한 인덱스를 구성하는 등의 최적화 작업을 수행해야 합니다.

# 4. SSMAv2.0를 이용한 마이그레이션 자동화

## SSMA v2.0 개요

SSMA(SQL Server Migration Assistant)의 주요 기능은 마이그레이션의 복잡성과 작업 시간을 산출하여 보고서를 작성하는 기능뿐만 아니라 테이블 및 뷰와 같은 스키마 개체와 데이터, 그리고 PL/SQL 구문으로 작성된 저장 프로시저와 트리거, 함수 및 동적 SQL 구문도 마이그레이션 할 수 있습니다. 따라서, SSMA를 사용하면 마이그레이션 프로젝트가 갖는 위험성과 작업 시간 및 비용을 크게 줄일 수 있게 됩니다.

SQL Server Migration Assistant For Oracle v2.0은 오라클의 대부분의 버전(7,8,8i,9i,10g)에서 SQL Server 2000 및 SQL Server 2005로의 마이그레이션을 지원합니다.

다음은 SSMA의 주요 기능입니다.

1) 평가 리포트 생성
2) 스키마 마이그레이션
3) 데이터 마이그레이션
4) PL/SQL 코드 (비즈니스 로직) 마이그레이션
5) 유효성 검사 및 테스트

## SSMA 설치 준비

SSMA를 설치하기 전에 준비해야 하는 사항은 다음과 같습니다.

1) 소프트웨어 다운로드
2) J2RE 설치
3) JAVA 환경 설정
4) SQL Server 클라이언트 및 Oracle 클라이언트 설치

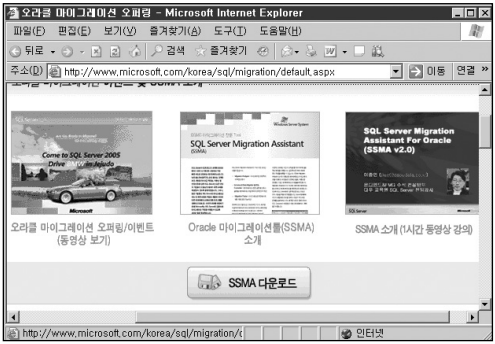
**■ 소프트웨어 다운로드**

SSMA를 설치하고 운영하기 위해서 필요한 소프트웨어는 다음과 같습니다.

① SSMA v2.0 및 확장팩
② SSMA v2.0 라이센스 키
③ JAVA Runtime Environment 1.4 이상
④ Oracle Client
⑤ SQL Server Client 도구

Oracle 및 SQL Server의 Client 설치는 본 가이드에서 설명하지 않습니다.

■ 먼저, SSMA v2.0 다운로드로드 사이트로 이동합니다. SSMA v2.0을 다운로드 할 수 있는 사이트는 다음과 같습니다.




www.microsoft.com/korea/sql/migration/default.aspx
http://www.microsoft.com/sql/solutions/ssm/ssmav2.mspx

■ SSMA v2.0 라이센스 키 다운로드 사이트는 다음과 같습니다.
http://www.microsoft.com/sql/solutions/ssm/ssmalicense.mspx

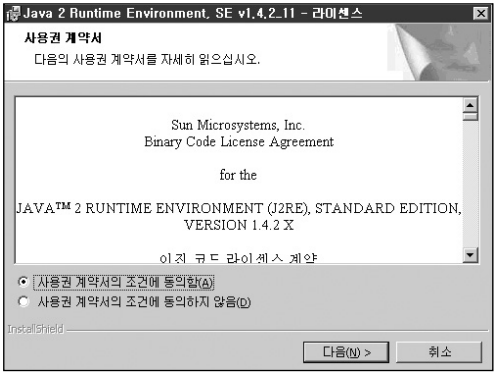
■ Java Runtime Evironment 다운로드 사이트는 다음과 같습니다.
http://java.sun.com/j2se/1.4.2/download.html

## ■ JRE 1.4 설치

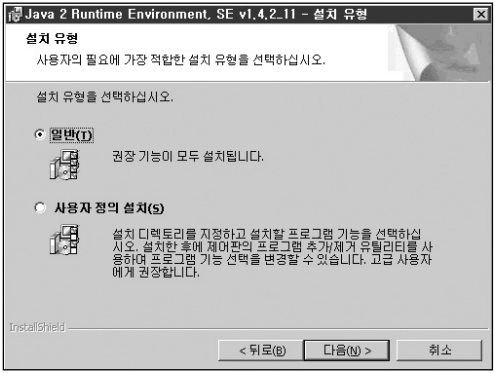
미리 다운로드 받은 JRE 1.4 이상 버전을 설치합니다. 설치가 완료되면 다음과 같이 올바른 환경설정을 구성하여야 합니다.

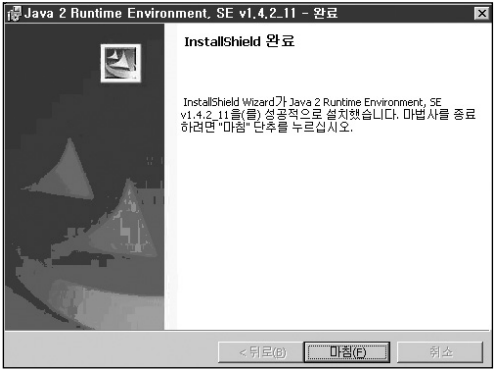
① JRE 1.4.2 이상 버전의 설치
② JAVA 환경 변수 설정

JRE v1.4 이상을 설치 프로그램을 시작합니다. 사용권 계약서에 동의하고 [다음 (N)] 버튼을 누릅니다.

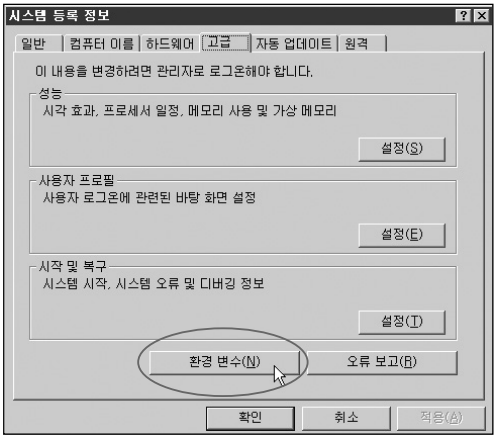


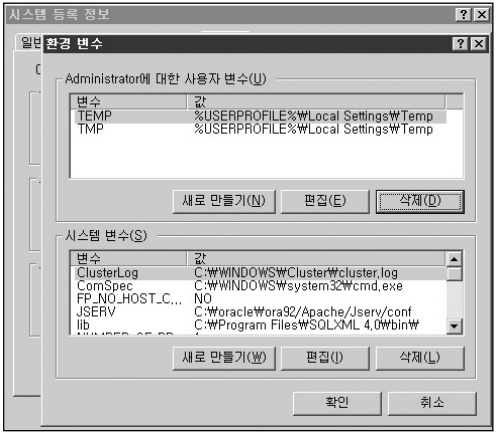
[설치 유형] 페이지에서 [일반 (T)]을 선택하고 [다음 (N)] 버튼을 누릅니다.



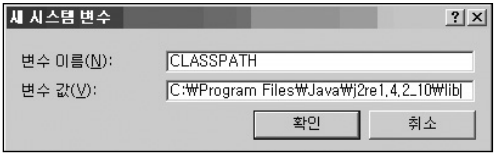




설치가 완료되면 [제어판]에서 [시스템]을 더블 클릭합니다. [시스템 등록 정보] 창에서 [고급] 탭을 선택하고 [환경 변수(N)] 버튼을 누릅니다.

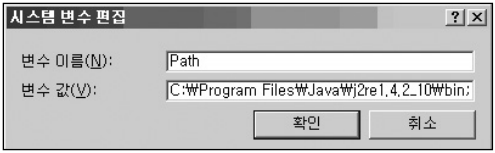

시스템 등록 정보
일반
컴퓨터 이름
하드웨어
고급
자동 업데이트
원격
이 내용을 변경하려면 관리자로 로그온해야 합니다.
성능
시각 효과, 프로세서 일정, 메모리 사용 및 가상 메모리
설정(S)
사용자 프로필
사용자 로그온에 관련된 바탕 화면 설정
설정(E)
시작 및 복구
시스템 시작, 시스템 오류 및 디버깅 정보
설정(T)
환경 변수(N)
오류 보고(R)
확인
취소
적용(A)


시스템 등록 정보
환경 변수
Administrator에 대한 사용자 변수(U)
변수
값
TEMP
%USERPROFILE%₩Local Settings₩Temp
TMP
%USERPROFILE%₩Local Settings₩Temp
새로 만들기(N)
편집(E)
삭제(D)
시스템 변수(S)
변수
값
ClusterLog
C:₩WINDOWS₩Cluster₩cluster.log
ComSpec
C:₩WINDOWS₩system32₩cmd.exe
FP_NO_HOST_C...
NO
JSERV
C:₩oracle₩ora92/Apache/Jserv/conf
lib
C:₩Program Files₩SQLXML 4.0₩bin₩
새로 만들기(W)
편집(I)
삭제(L)
확인
취소

[시스템 변수 (S)]에서 [새로 만들기(W)] 버튼을 누르고 다음과 같이 [변수 이름(N)]에 [CLASSSPATH]를 입력하고 [변수 값 (V)] 부분에 JRE 1.4 이상의 라이브러리 파일 경로를 입력하고 [확인] 버튼을 누릅니다.




이어서 [시스템 변수 (S)] 가운데 [Path]를 선택하고 [편집 (I)] 버튼을 눌러서 [변수 값 (V)] 부분에 JRE 1.4 이상의 바이너리 파일 경로를 지정하고 [확인] 버튼을 누릅니다.




## SSMA 설치 및 시작하기

### ■ SSMA 설치 단계

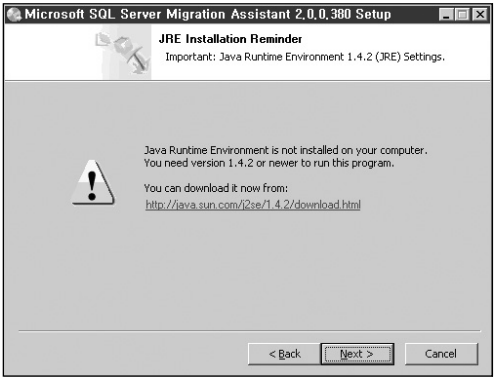
① SSMA 설치
② SSMA 확장 팩 설치
③ 라이센스 키 등록

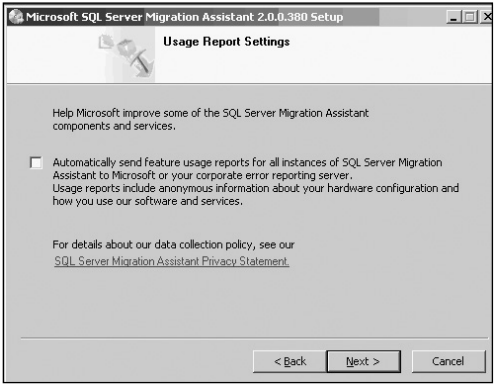
## ■ SSMA 설치

① 미리 다운로드 받아 둔 SSMA 설치 파일(SSMA-Setup-2.0.0.380.exe)을 더블 클릭하여 설치 마법사를 시작합니다.

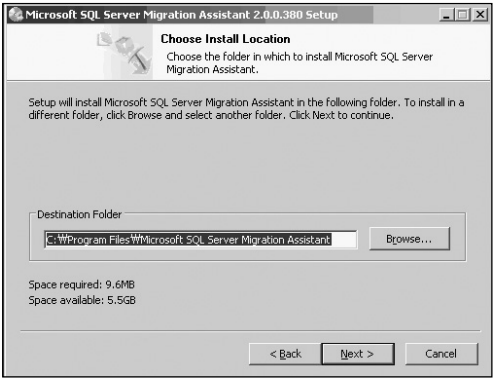

② 다음 버튼을 누르고 사용자 라이센스 동의 화면에서 동의 버튼을 누릅니다.

③ Java Runtime Environment 1,4,2가 설치되지 않은 경우 다음과 같이 경고 페이지가 나타납니다. 이미 JRE를 설치한 경우 리포트 설정 화면에서 다음 버튼을 누릅니다.

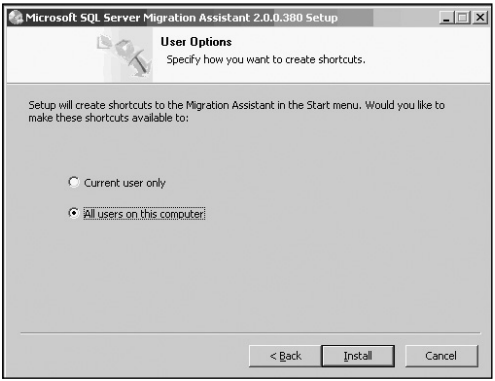


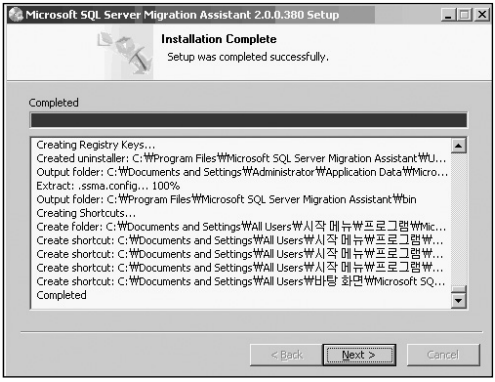
④ 설치 경로를 확인하고 다음 버튼을 누르거나, [Browse…] 버튼을 눌러서 설치경로를 변경하고 다음 버튼을 누릅니다.

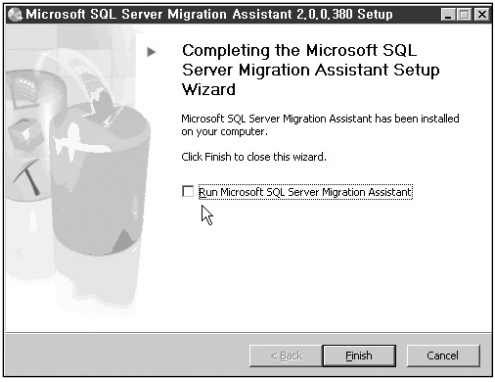



⑤ 사용자 옵션 화면에서 모든 사용자를 선택하고 설치 버튼을 누릅니다.



⑥ 설치 진행 사항을 확인하고 다음 버튼을 누르고 Microsoft SQL Server Migration Assistant 체크 박스를 해제한 뒤 마법사를 종료합니다.

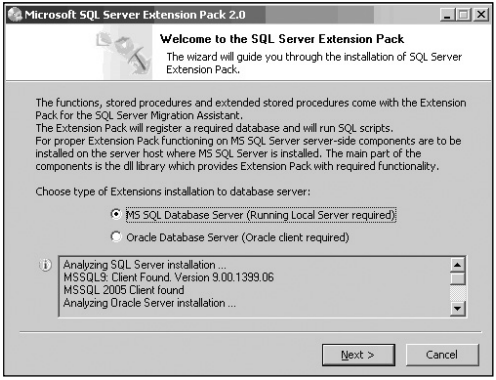


## ■ SSMA 확장 팩 설치

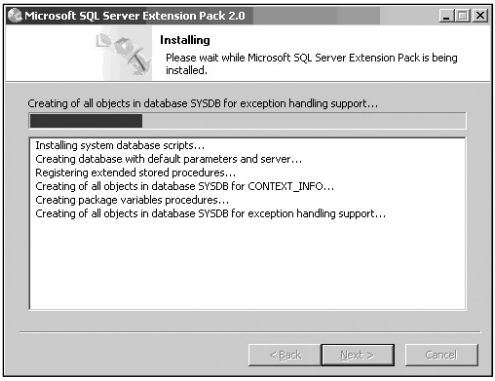
① 미리 다운로드 받아 둔 확장 팩 설치 파일(SSMAExtPack-2.0.0.380.exe)을 더블 클릭하여 설치 마법사를 실행합니다.

② SQL Server를 선택하여 SQL Server용 확장 팩을 먼저 설치합니다.
SQL Server용 확장 팩과 Oracle용 확장 팩의 설치 순서는 상관없습니다.

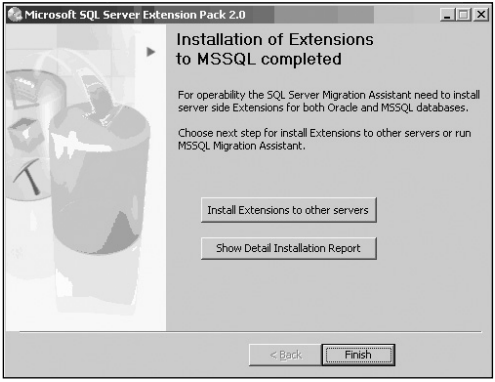



③ SQL Server 구성 페이지에서 확장 팩을 설치하고자 하는 SQL Server 인스턴스 이름을 선택하고 SQL 로그인 계정과 패스워드를 입력합니다. Windows 인증은 지원하지 않습니다.

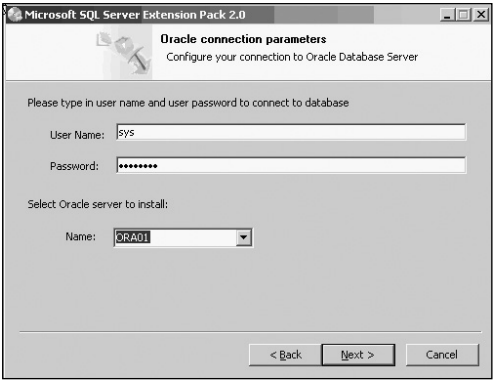
④ 설치 진행 과정을 확인합니다.
SYSDB 데이터베이스를 생성하고 오라클 시퀀스나 패키지, 함수 등을 에뮬레이션하는 기능을 하는 확장 저장 프로시저(xp_ora2ms_exec, xp_ora2ms_exec_ex, xp_ora2ms_versioninfo) 와 테이블 등을 생성합니다.

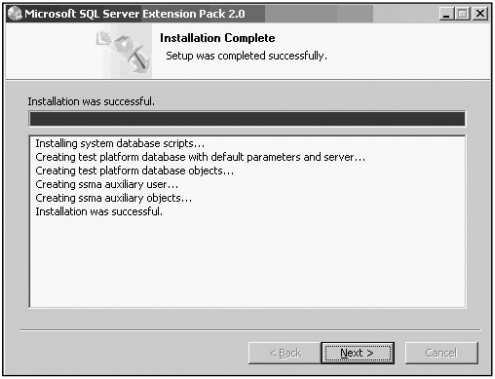



설치가 완료되면 [Next]를 누르고 [Install Extensions to other servers] 버튼을 눌러서 Oracle용 확장 팩을 설치합니다.



⑤ 마이그레이션 하고자 하는 Oracle 의 네트워크 서비스 이름을 입력하고 유효한 권한을 소유한 사용자 계정과 패스워드를 입력하고 다음 버튼을 클릭합니다.



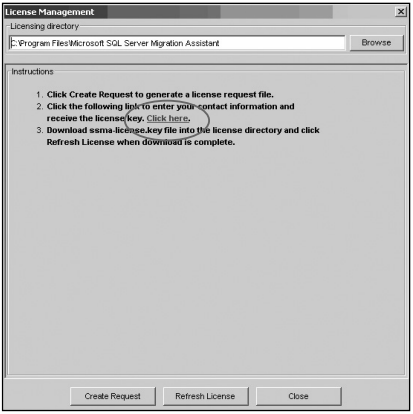




⑥ Oracle 용 확장 팩 설치도 완료되면 SSMAv2.0 프로그램을 실행합니다.

## ■ SSMA 시작 및 라이센스 키 등록하기

① [시작]→ [모든 프로그램(P)]→ [Microsoft SQL Server Migration Assistant]에서 [Microsoft SQL Server Migration Assistant]를 선택해서 SSMA를 시작합니다.
설치 후 처음 시작하는 경우 다음과 같은 라이센스 키를 등록하라는 메시지 박스가 보여집니다.


② [Yes] 버튼을 누르고 라이센스 키 관리 페이지에서 미리 라이센스 키를 다운로드 한 경우에는 다운로드 되어있는 폴더를 지정하고 처음 설치하는 경우라서 라이센스 키를 다운로드 하지 않은 경우는 [Click here] 링크를 눌러서 마이크로소프트 사이트로 이동하여 라이센스 키를 다운로드하고 해당 경로를 지정 한 뒤 [Refresh License] 버튼을 누릅니다.



## ■ SSMA 옵션 구성 하기

① 메뉴 바의 [Tools] 메뉴에서 [Default Option]을 선택합니다. [Default Option]은 전반적으로 적용되는 SSMA의 기본 설정 옵션을 지정하는 데 사용되고 [Project Option]은 특정 프로젝트에만 적용되는 옵션을 지정하는 데 사용됩니다.

② [Default Option]은 다음과 같이 총 8개의 탭으로 구성되어 있습니다.

가) General 탭 : 변환하고자 하는 SQL Server 버전 선택, 변환하고자 하는 Oracle 시스템 스키마 선정 등

나) Assessment Reports 탭 : 변환 작업에 대상 개체의 총 숫자 및 변환에 소요되는 시간과 복잡성에 대한 보고서 작성과 관련된 옵션 등

다) SQL Converter 탭 : PL/SQL을 T-SQL로 변환 시 옵션 설정 등

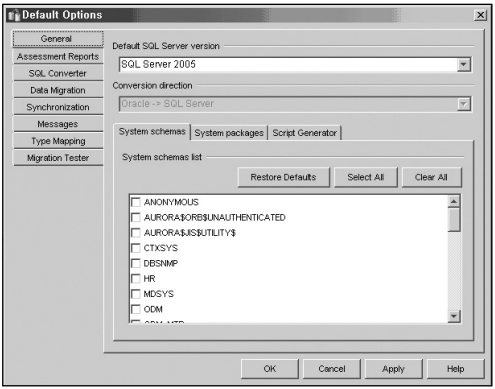
라) Data Migration 탭 : 데이터 마이그레이션을 위한 연결된 서버 설정 등

마) Synchronization 탭 : 작업자의 워크 스페이스와 데이터베이스간의 동기화 옵션 설정 등

바) Messages 탭 : 변환 작업 중 발생한 오류와 로그 정보관련 옵션 설정 등

사) Type Mapping 탭 : 데이터 형식의 변환 옵션 설정 등

아) Migration Tester 탭 : 마이그레이션 테스트 작업을 위한 옵션 설정 등



이제 각 옵션 탭의 자세한 기능에 대해서 살펴봅니다.

가) General 탭

- Default SQL Server Version : 변환하고자 하는 SQL Server 버전을 SQL Server 2000 또는 SQL Server 2005 중 선택할 수 있습니다.
- System Schemas : system schema list에서 마이그레이션 하고자 하는 시스템 스키마를 선택합니다. 일반적으로 시스템 스키마는 마이그레이션할 필요가 없습니다.
- System Packages : system package list에서 마이그레이션 하고자 하는 시스템 패키지를 지정할 수 있습니다. 일반적으로 시스템 패키지는 마이그레이션할 필요가 없습니다.
- Script Generator : 기존에 동일한 스키마 개체가 있는 경우에 먼저 삭제를 할 수 있도록 [Generate Drop statement] 체크 박스를 선택할 수 있습니다.

나) Assessment Reports 탭 :

- Report Directory : 평가 리포트 기본 저장 경로를 지정합니다.
- Queries lists : 평가 리포트에 포함할 컬럼을 지정합니다. 컬럼의 내용은 [SSMA v2.0 마이그레이션 프로세스]의 평가 리포트 부분에서 자세히 설명합니다.

다) SQL Converter 탭 :

- ROWID column generation : 각 테이블 생성시 ROWID 컬럼을 추가합니다.
- Sequence to identity conversion : Oracle 시퀀스 개체를 SQL Server 테이블의 identity 속성으로 변환합니다.
- Transactions Parameters : 트랜잭션 처리구문을 변환합니다.
- Exceptions Parameters : 예외 처리를 변환합니다.
- Conversion Mode : Default, Optimistic, Full, Custom 모드를 지원하며 [Default] 모드는 기본 변환을 수행하며 [Optimistic] 모드는 최적의 변환을 수행합니다. [Full] 모드는 최대한 변환하도록 변환을 수행하며 [Custom] 모드는 예외 처리등과 관련해서 사용자가 변환 옵션을 지정합니다. [Custom] 모드에서 [Optimistic] 모드와 [Full] 모드의 차이점을 구별할 수 있습니다.
- Code Generator Messages : 구문 변환 시 오류 메시지 및 경고 메시지 등을 표시할 지 여부를 지정합니다.

라) Data Migration 탭 :

- 대상 SQL Server에서 데이터 마이그레이션 시에 사용할 연결된 서버 설정을 합니다.

마) Synchronization 탭 :

- Object change status : 다음과 같은 네 가지의 개체 변경 상태를 지정합니다.
  ① 워크 스페이스 개체는 변경되었지만 데이터베이스에 반영되지 않은 경우
  ② 데이터베이스의 개체는 변경되었지만 워크 스페이스에 반영되지 않은 경우
  ③ 데이터베이스의 개체와 워크 스페이스 개체가 별도로 변경된 경우
  ④ 데이터베이스에 새로운 개체가 추가되었지만 워크 스페이스에 반영되지 않은 경우

- Synchronization action : Object change status에서 지정한 상태에 따라 처리하는 동작을 다음과 같은 네 가지로 지정합니다.
  ① Do Nothing : 아무런 작업을 수행하지 않습니다.
  ② Always overwrite workspace copy with database object : 항상 데이터베이스의 변경 사항으로 워크 스페이스에 반영합니다.
  ③ Always overwrite database object with workspace copy : 항상 워크 스페이스의 변경사항으로 데이터베이스에 반영합니다.
  ④ Ask User : 사용자에게 변경사항의 반영 여부를 묻습니다.

바) Messages 탭 :

- Output Directory : 로그를 저장할 폴더를 지정합니다.
- Output Filename : 로그 파일이름을 지정합니다.
- Maximum log file size : 로그 파일의 최대 크기를 지정합니다.
- Categories : 워크 스페이스 동기화나 코드 변환 등 작업 시 경고, 오류, 디버그 등에 대해서 정의 합니다.

사) Type Mapping 탭 :

- Data Type Usage Category : 다음 네 가지 경우의 데이터 형식 매핑 유형을 지정합니다.
  ① Type mapping for arguments : 매개 변수 데이터 형식 매핑
  ② Type mapping for Local Variables : 지역 변수의 데이터 형식 매핑
  ③ Type mapping for Return values : 반환 값의 데이터 형식 매핑
  ④ Type mapping for Table Columns : 테이블의 컬럼 데이터 형식

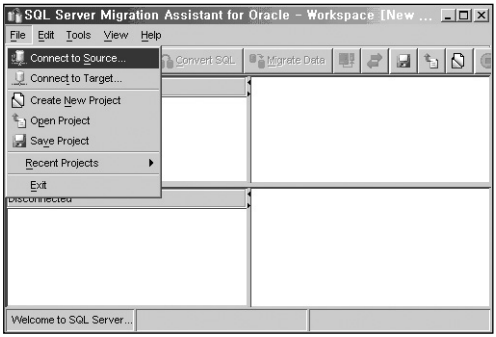
- Type Mapping Setting : Source Type 부분에 Oracle 원본의 데이터 형식을 지정하고, Target Type 부분에 대상 SQL Server 데이터 형식을 지정합니다.

아) Migration Tester 탭 :

- Use Existing Table' s Data : 현재 마이그레이션된 데이터로 테스트를 진행합니다.
- Compare function return values : 함수의 반환 결과 값의 동일 여부를 비교합니다.
- Compare OUTPUT parameters : 출력 매개 변수의 결과 값의 동일 여부를 비교합니다.
- Compare Result sets : 반환 결과 값의 동일 여부를 비교합니다.
- Strip trailing spaces : 반환 값의 후행 공백을 무시하여 비교합니다.

## 원본 Oracle 및 대상 SQL Server 연결하기

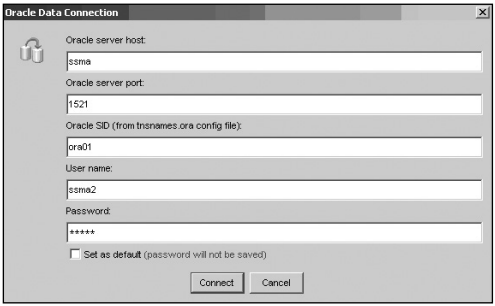
다음 그림과 같이 SSMA가 실행되면 [File] 메뉴의 [Connect to Source]를 선택해서 원본 Oracle 서버에 연결하고 [Connect to Target]을 선택해서 대상 SQL Server에 연결합니다.



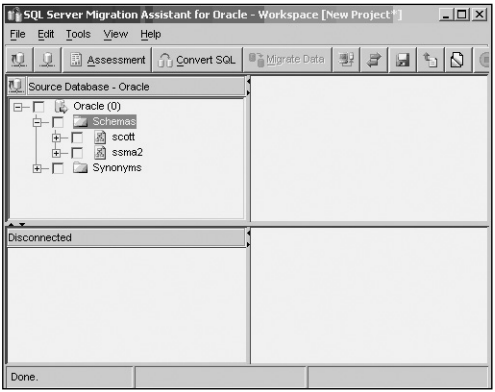

다음과 같이 메뉴바 밑의  아이콘을 통해서도 연결을 설정할 수 있습니다.

① 원본 Oracle에 연결하기

[File] 메뉴에서 [Connect to Source]를 선택하면 다음과 같이 [Oracle Data Connection] 페이지가 실행되며 [Oracle server host] 부분에는 Oracle이 설치된 호스트 이름을 입력하고, [Oracle server port] 부분에는 연결하고자 하는 Oracle 인스턴스가 사용하는 TCP 포트 번호를 지정합니다. Oracle의 기본 포트는 1521입니다. [Oracle SID(from tnsnames.ora config file)] 부분에는 tnsnames.ora 파일에 지정된 연결하고자 하는 Oracle 서비스 명에 대한 접속 문자열을 입력합니다. [User namer]과 [Password] 부분에는 마이그레이션 하고자 하는 스키마에 대해서 적절한 권한을 가진 계정과 패스워드를 입력합니다.



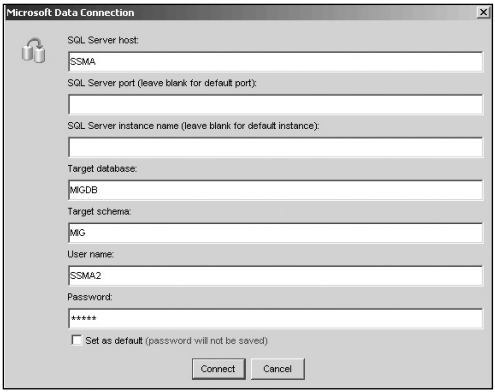

[Connect] 버튼을 눌러서 연결을 시도하면 다음과 같이 대상 Oracle에 접속해서 필요한 정보를 SSMA로 로딩하여 원본 Oracle의 스키마 개체에 대한 정보를 확인 할 수 있습니다.



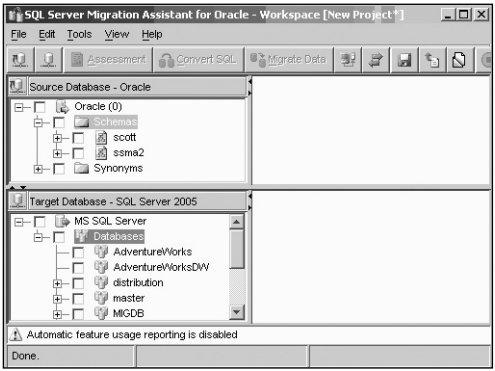

② 대상 SQL Server에 연결하기

대상 SQL Server에 연결하기 위해서 [File] 메뉴의 [Connect to Target]를 선택하고 다음과 같이 입력합니다. [SQL Server host]부분에는 SQL Server의 호스트 이름을 입력하고 [SQL Server port]부분에는 대상 SQL Server 인스턴스가 사용중인 TCP 포트 번호를 입력합니다. SQL Server의 기본 포트인 1433을 사용하는 경우에는 빈칸을 그대로 두어야 합니다.

[SQL Server instance name]부분에는 연결하고자 하는 SQL Server 인스턴스 이름을 입력합니다. 기본 인스턴스에 연결하는 경우에는 인스턴스 이름을 입력하지 말고 빈칸으로 그대로 두어야 하며, 명명된 인스턴스에 연결하는 경우에는 [기본 인스턴스 이름\명명된 인스턴스 이름]을 모두 입력하지 말고 [명명된 인스턴스 이름]부분만 입력합니다. 명명된 인스턴스 이름이 [ANGIE\SECOND]의 경우라면 [SECOND]만 입력합니다. [Target database]부분에는 대상 데이터베이스 이름을 지정합니다. [Target schema]부분에는 대상 스키마 이름을 지정합니다. [User name]과 [Password] 부분에는 해당 데이터베이스에 스키마 개체를 생성할 수 있는 권한이 부여된 계정과 패스워드를 입력합니다. 다음에서는 대상 데이베이스는 MIGDB와 대상 스키마는 MIG 스키마를 사용하도록 한 경우입니다.




[Connect] 버튼을 눌러서 다음과 같이 대상 SQL Server의 정보를 SSMA로 로딩합니다.

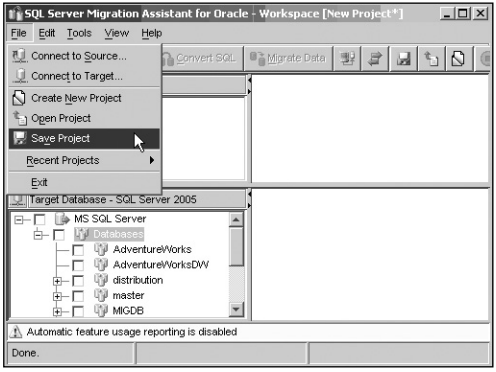


## 워크 스페이스 생성

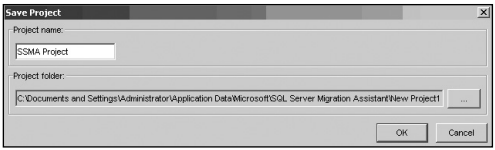
워크 스페이스란 마이그레이션 작업 시 계속해서 원본 Oracle과 대상 SQL Server에 연결을 유지할 필요 없이 현재의 작업 상태를 파일에 저장하여 계속해서 작업을 할 수 있도록 하는 저장소입니다.

① 프로젝트를 워크 스페이스에 저장

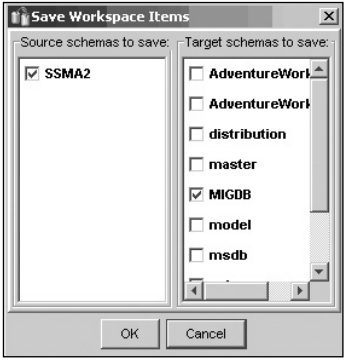
다음 그림과 같이 [File] 메뉴의 [Save Project] 메뉴를 통해서 현재 작업중인 프로젝트를 저장할 수 있고 [Open Project]를 통해서 원본 Oracle과 대상 SQL Server에 다시 연결할 필요 없이 계속해서 마이그레이션 작업을 진행할 수 있습니다.



다음 그림과 같이 [Project name] 부분에 프로젝트 이름을 지정하고 [Project folder]부분에 파일 저장 경로를 지정하고 [OK] 버튼을 누릅니다.

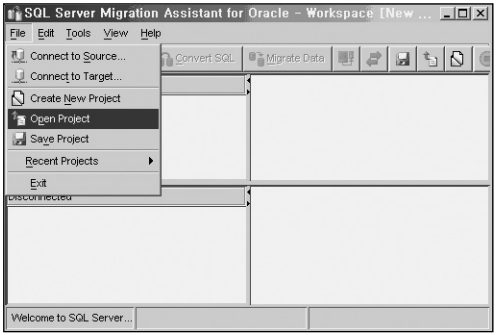



다음과 같이 워크 스페이스에 저장할 아이템을 선택하는 대화 상자에서 정보를 저장할 필요가 있는 원본 Oracle의 스키마와 대상 SQL Server 데이터베이스를 선택하고 [OK] 버튼을 누릅니다. 프로젝트를 워크 스페이스에 저장하는 동안 다소의 시간이 소요될 수 있습니다.

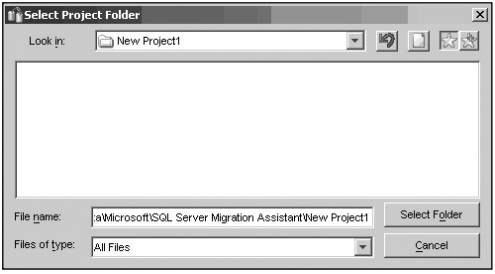


② 저장된 프로젝트를 워크 스페이스에서 열기

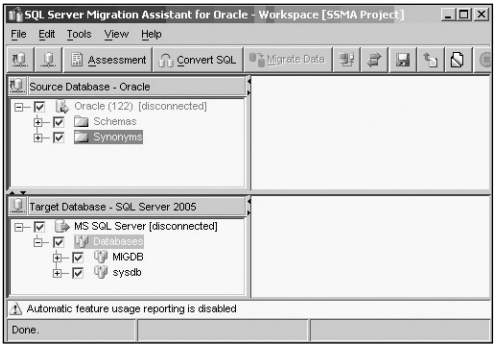
원본 Oracle과 대상 SQL Server에 연결을 유지할 필요 없이 기존의 작업 상태가 워크 스페이스에 저장된 경우에는 다음과 같이 [File] 메뉴의 [Open Project]를 선택해서 기존에 저장된 프로젝트를 열어서 최근에 저장된 상태에서 다시 작업을 진행할 수 있습니다.




다음과 같이 프로젝트가 저장된 경로를 지정합니다. 파일을 선택하는 것이 아니라 다음과 같이 프로젝트 저장 시 지정한 [Project Folder]를 지정합니다.



이어서 [OK] 버튼을 누르면 다음과 같이 원본 Oracle과 대상 SQL Server에 연결 없이 기존에 저장된 상태의 환경을 로딩합니다.




## SSMA v2.0 마이그레이션 프로세스

SQL Server Migration Assistant For Oracle v2.0은 다음과 같은 단계로 마이그레이션 프로세스를 진행합니다.

1) 평가 리포트 생성
2) 스키마 마이그레이션
3) 데이터 마이그레이션
4) PL/SQL 코드 (비즈니스 로직) 마이그레이션
5) 유효성 검사 및 테스트

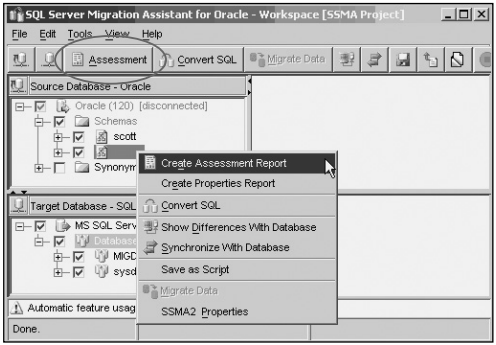
### ■ 평가 리포트 생성

SSMA 평가 리포트는 마이그레이션의 원본 데이터베이스를 분석해서 다음과 같은 사항에 대한 리포트를 생성합니다.

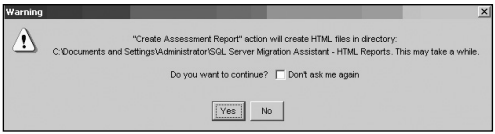
① 대상 개체의 총 개수
② 스키마 개체 별 개수 및 SQL 구문의 행수
③ 변환 작업의 복잡성 산출
④ 수동 변환 시 예상 작업 시간 산출
⑤ SSMA를 통한 자동 변환 비율
⑥ Oracle RECORD, TABLE 형식 등의 총 개수 및 자동 변환 비율
⑦ Rownum 키워드 사용 및 예외 선언 절 등의 총 개수 등

이와 같은 평가 리포트는 해당 마이그레이션 프로젝트의 복잡성, SSMA를 통한 자동 변환 비율, 소요 시간 및 비용 등을 평가 할 수 있어서 전체 프로젝트를 가늠할 수 있는 중요한 자료가 됩니다.

다음은 평가 리포트를 생성하는 방법입니다. 평가 리포트는 다음과 같이 평가 리포트를 생성하고자 하는 스키마를 선택하고 메뉴 바의 Assessment 버튼을 누르거나 평가 리포트를 생성하고자 하는 스키마를 선택하고 오른쪽 버튼을 눌러서 [Create Assessment Report]를 선택합니다.

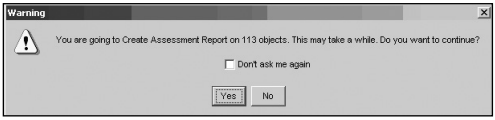


다음과 같이 평가 리포트 저장 폴더를 확인하는 경고 메시지 창이 나타납니다. [Yes] 버튼을 누릅니다.

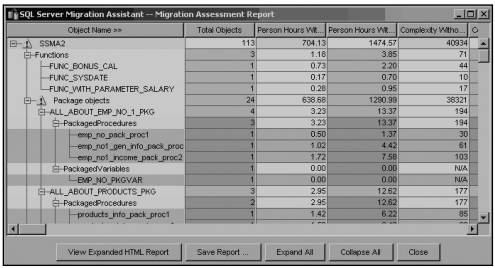



〈확장 HTML 리포트 저장 경로 지정〉

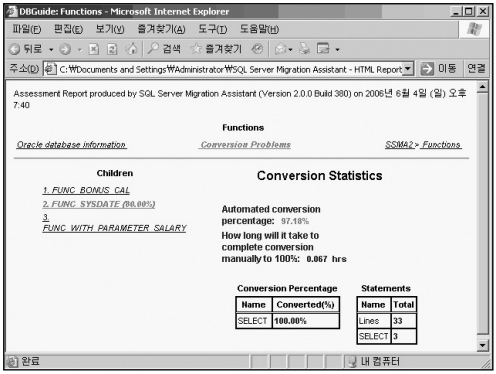
이어서 다음과 같이 평가 리포트 생성 대상 개체 정보를 확인하고 [Yes] 버튼을 눌러서 평가 리포트를 생성합니다. 평가 리포트를 생성하는 작업은 스키마 개체의 개수나 행의 수에 따라서 다소 시간이 소요됩니다




평가 리포트는 다음과 같습니다.



[View Expanded HTML Report] 버튼은 **〈확장 HTML 리포트 저장 경로 지정〉** 그림에서 확인한 폴더에 저장된 다음과 같은 확장 HTML 리포트를 보여줍니다.




[Save Report] 버튼은 평가 리포트 내용을 별도의 쉼표로 구별된 파일(.csv)로 저장할 때 사용하며 마이크로소프트 Excel과 같은 응용 프로그램에서 사용할 수 있습니다.

다음은 주로 사용하는 평가 리포트의 컬럼 설명입니다.

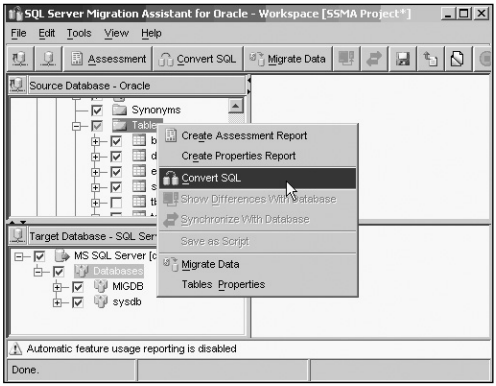
| 컬럼 이름 | 설명 |
|---|---|
| Total Objects | 전체 개체 수 |
| Person Hours(Tot) | SSMA가 예상한 수동 변환 소요 시간 |
| Complexity | 해당 작업의 복잡도 추정 |
| All(Tot) | SQL 구문의 총 개수 |
| All(Conv%) | SSMA로 자동 변환되는 SQL 구문의 비율 |
| Record Type Decl(Tot) | Oracle 레코드 타입 선언 총 개수 |
| Record Type Decl (Conv%) | Oracle 레코드 타입의 자동 변환 비율 |
| Rownum | Rownum 키워드 사용 개수 |
| Exception Handler | 예외절 선언 개수 |
| Unparsed(Tot) | SSMA가 구문 분석하지 못한 SQL 구문 총 개수 |

**■ 스키마 마이그레이션**

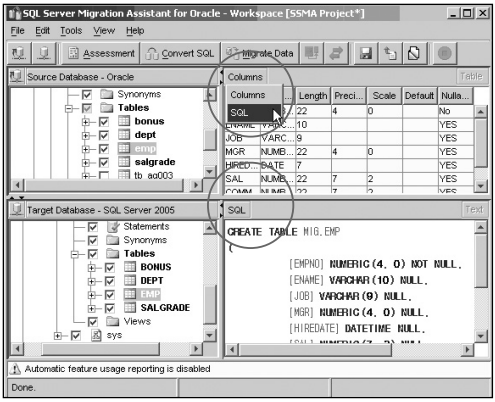
스키마 개체는 SQL Server와 Oracle간에 유사합니다. 그렇지만 두 DBMS 간의 기능 차이 때문에 일대일 매핑이 이루어지지 않을 수도 있습니다. Oracle의 시퀀스나 개체 형식은 동일하게 변환하여 사용할 수 없습니다. SSMA는 시퀀스는 SQL Server 테이블의 identity 속성으로 매핑하거나 확장 팩과 함께 설치되는 SYSDB를 통해서 유사한 기능을 지원합니다. 개체 형식은 테이블 디자인을 변경할 수 있으므로 면밀한 점검을 필요로 합니다. 일반적인 테이블, 제약 조건, 인덱스는 대부분 자동으로 변환할 수 있습니다.

■ 테이블 마이그레이션

다음은 SSMA를 사용해서 테이블, 제약 조건, 인덱스를 마이그레이션하는 방법을 설명합니다. 다음과 같이 마이그레이션 하고자 하는 스키마 개체를 선택하고 오른쪽 버튼을 클릭하고 [Convert SQL]을 누릅니다.

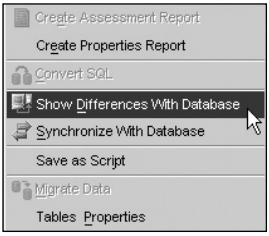



다음과 같이 수행한 스키마 개체의 마이그레이션 결과를 확인할 수 있습니다.
[Target Database-SQL Server 2005]부분의 [Tables] 부분의 테이블 가운데 하나를 선택하고 ◯ 표시된 부분에서 [column] 또는 [SQL]을 선택하면 다음 그림과 같이 해당 테이블 생성 T-SQL 스크립트나 컬럼의 정보를 확인할 수 있습니다.

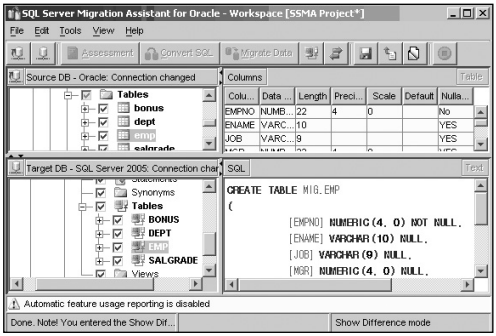



■ 프로젝트 워크 스페이스와 대상 SQL Server 변경 사항 확인

현재의 변환은 프로젝트의 메모리 공간에서만 발생하였고 대상 SQL Server에는 반영되지 않은 상태입니다. 대상 SQL Server의 반영 여부를 확인하기 위해서는 다음과 확인하고자 하는 대상 개체를 선택하고 오른쪽 버튼을 클릭해서 [Show Differences With Database]를 선택합니다.



다음과 같이 프로젝트의 워크 스페이스와 대상 SQL Server와의 차이를 아이콘으로 보여줍니다.

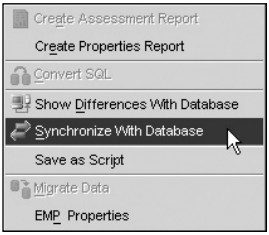



다음은 프로젝트 워크 스페이스(또는 메모리)와 대상 SQL Server와의 차이를 설명하는 아이콘 입니다. 모니터 모양의 아이콘은 워크 스페이스를 표시하며 그 옆의 PC모양의 아이콘은 대상 SQL Server를 표시하며 컬러로 표시되면 변경 사항이 발생한 것으로 이해하면 좋습니다.

| 아이콘 | 설명 |
|---|---|
|  | 데이터베이스와 프로젝트 워크 스페이스 모두 변경되지 않음 |
|  | 워크 스페이스에서는 변경되었으나 데이터베이스에는 반영되지 않음 |
|  | 데이터베이스에서는 변경되었으나 워크 스페이스에서는 변경되지 않음 |
|  | 데이터베이스와 워크 스페이스 모두 변경 사항이 발생함 |
|  … | 데이터베이스에는 존재하지만 워크 스페이스에는 존재하지 않음 |

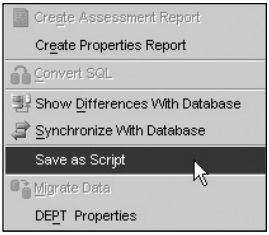
■ 프로젝트 워크 스페이스 변경 사항 반영

이제 대상 SQL Server에 생성하고자 하는 테이블을 선택하고 오른쪽 버튼을 클릭해서 다음과 같이 [Synchronize With Database]를 선택합니다.

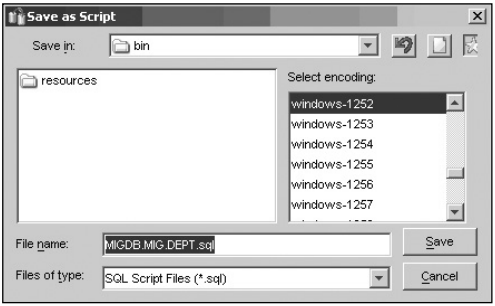



■ 테이블 생성 스크립트 저장

프로젝트 워크 스페이스에서 변환된 테이블을 데이터베이스에 즉시 반영하지 않고 향후에 반영하기 위해서 테이블 생성 스크립트를 생성해서 저장할 수 있습니다. 저장소를 지정하거나 데이터 형식 매핑의 변경, 또는 향후 운영시스템에 스크립트로 반영하기 위한 것 등이 이유가 될 수 있습니다. 스크립트를 생성해서 저장하려면 다음과 같이 해당 개체를 선택하고 오른쪽 버튼을 클릭해서 [Save as Script]를 선택합니다.



저장하고자 하는 대상 폴더와 스크립트 파일의 이름을 지정하고 [Save] 버튼을 누릅니다.

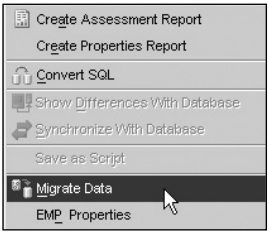



## ■ 데이터 마이그레이션

테이블이 마이그레이션 되면 데이터를 마이그레이션 할 수 있습니다. 데이터는 지정한 테이블별로 마이그레이션 하거나 선택한 테이블 전체를 한번에 마이그레이션할 수 있습니다.
대용량의 테이블을 마이그레이션할 때는 BCP, BULK INSERT, SSIS (SQL Server Integration Services) 등을 사용할 수 있습니다. 또한, 복잡한 변환이 필요한 경우에는 SSIS (SQL Server Integration Services)를 사용하면 효율적이고 다양한 변환이 가능합니다.
대용량 테이블을 마이그레이션하고자 하는 경우에는 원본 Oracle의 롤백 세그먼트(UNDO 테이블스페이스)의 할당이나 설정 옵션을 면밀히 검토해야 합니다. Oracle로부터 데이터를 추출하는 동안 ORA-01555 "snapshot too old" 오류가 발생할 수 있으므로 특시 주의해야 합니다.

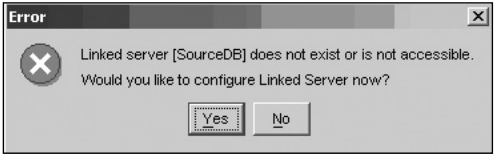
> ***[참고]***
> *데이터 마이그레이션과 관련된 BCP, BULK INSERT,SSIS 등은*
> *[5장 Oracle 기반 데이터 인터페이스 마이그레이션] 모듈을 참조하시기 바랍니다.*

데이터를 마이그레이션 하려면 테이블을 선택하고 메뉴 바의 Migrate Data 버튼을 누르거나 다음과 같이 오른쪽 버튼을 클릭하여 [Migrate Data]를 선택합니다.

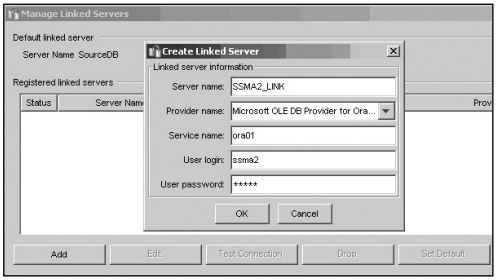



■ SSMA 데이터 마이그레이션용 연결된 서버 설정

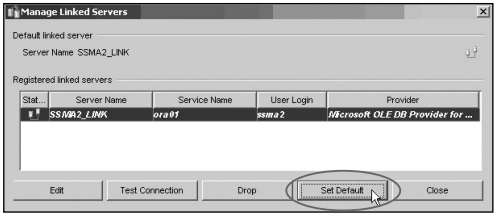
SSMA를 통해서 데이터를 마이그레이션 하기 위해서는 SSMA에서 연결된 서버가 설정되어야 합니다. 연결된 서버가 설정되지 않은 경우에는 다음과 같은 오류 메시지 창이 나타납니다.



SSMA 데이터 마이그레이션용 연결된 서버의 설정은 [Tools]메뉴의 [Default Options] 또는 [Project Options] 을 선택하고 [Data Migration] 탭에서 [Manage Linked Servers] 버튼을 누릅니다. 다음과 같이 [Add] 버튼을 눌러서 연결된 서버를 구성합니다.

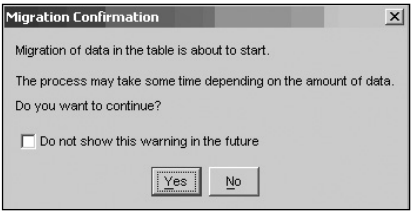



SSMA 데이터 마이그레이션용 연결된 서버가 설정되면 다음과 같이 확인할 수 있습니다.



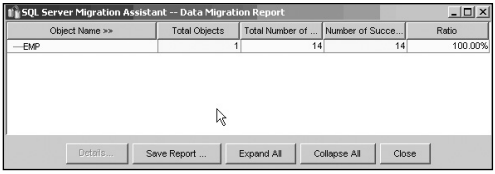

이때 반드시 [Set Default] 버튼을 눌러서 SSMA가 해당 연결된 서버를 사용할 수 있도록 해야 합니다.

이제 다시 데이터 마이그레이션을 시도하면 다음과 같이 데이터를 마이그레이션하는 동안 다소 시간이 소요된다는 확인 메시지 창이 나타나며 [Yes] 버튼을 눌러서 데이터 마이그레이션을 시작합니다.




데이터 마이그레이션이 수행되는 동안 다음과 같이 진행중임을 확인할 수 있고 [Stop]버튼을 통해서 작업을 중지할 수 있습니다.


데이터 마이그레이션은 OPENQUERY 함수를 사용해서 INSERT ~ SELECT 구문으로 수행되며 데이터 마이그레이션이 완료되면 다음과 같은 [데이터 마이그레이션 리포트]를 제공합니다. 데이터 마이그레이션 리포트는 마이그레이션한 개체의 개수 및 데이터 행 수, 성공한 행 수 및 성공 비율을 보고합니다.

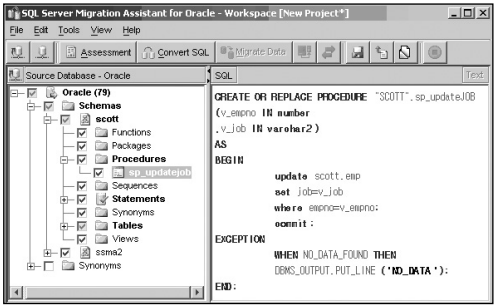



***[참고]***
*OPENQUERY의 자세한 사항은 88페이지 [3) 연결된 서버를 통한 Query 실행하기] 부분을 참조하기 바랍니다.*

## ■ PL/SQL 구문 (비즈니스 로직) 마이그레이션

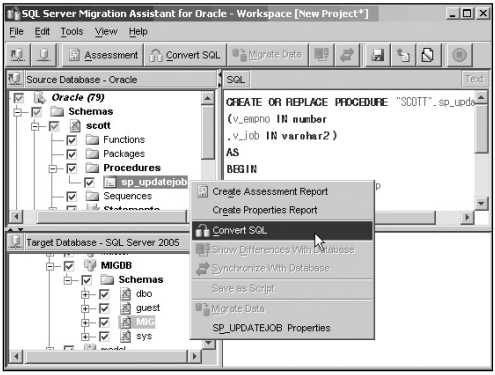
SSMA는 PL/SQL로 작성된 쿼리나 저장 프로시저, 함수, 패키지 등 구문을 T-SQL로 자동변환합니다. 일부 일대일 매핑이 되지 않는 기능은 확장 팩으로 설치된 SYSDB와 확장 저장프로시저를 사용해서 동일한 기능을 에뮬레이션하도록 변환합니다. 패키지의 경우는 패키지내의 함수와 프로시저를 각각 별도의 함수와 프로시저로 변환합니다. 개체의 이름은 패키지명$함수명 또는 패키지명$프로시저명 형태의 명명 규칙을 사용합니다.

이제 SSMA를 사용하여 저장 프로시저를 마이그레이션하는 방법을 살펴봅니다.
왼쪽의 원본 데이터베이스 창에서 변환하고자 하는 프로시저를 선택하면 오른쪽 코드 창에서 해당 프로시저의 구문을 확인할 수 있습니다.

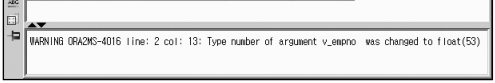



〈원본 PL/SQL 코드〉

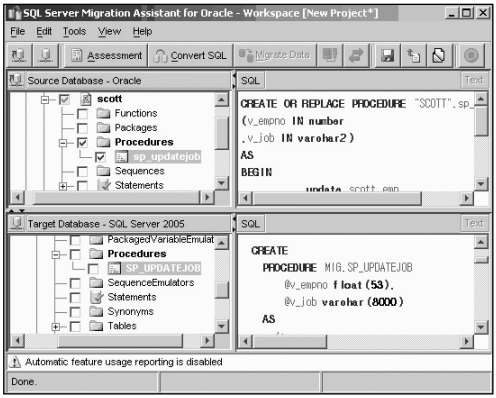
변환하고자 하는 프로시저를 선택하고 다음과 같이 오른쪽 버튼을 클릭해서 [Convert SQL]을 선택합니다.



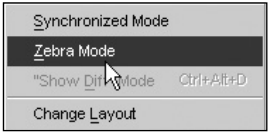

앞에서 살펴본 [5) SSMA 실행 및 옵션 구성 하기]에서 SSMA 옵션와 같이 경고 지정 옵션에 따라서 코드 변환 중 다음과 같은 경고가 발생할 수 있습니다. 다음은 데이터 형식의 변환에 대한 경고입니다. 데이터 형식의 변환은 SSMA 옵션에서 구성을 하든지 직접 코드 창에서 수정할 수 있습니다.



코드 변환이 완료되면 다음과 같이 원본 데이터베이스 코드 창과 대상 데이터베이스 코드 창에 원본 코드와 변환된 코드를 확인할 수 있습니다.



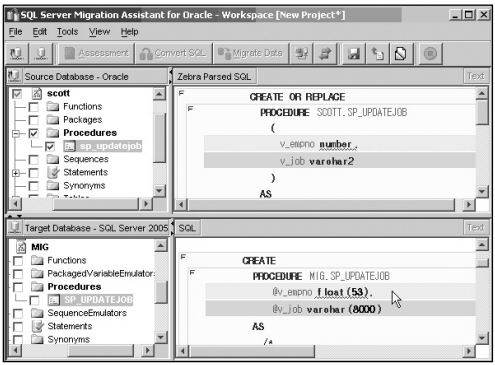

다음은 메뉴 바의 [View] 옵션입니다. SSMA의 보기 모드는 세 가지를 지원합니다.



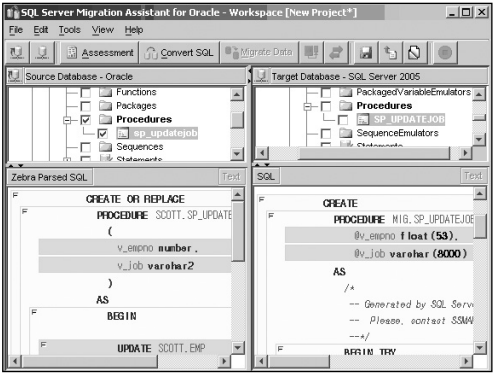

① "Show Diff" Mode : 앞에서 살펴본 바와 같이 ... 등의 아이콘을 통해서 데이터베이스와 워크 스페이스의 변경 여부를 확인할 수 있는 모드입니다.

② Synchronized Mode : 원본 코드와 변환된 대상 코드를 한번에 볼 수 있는 모드입니다. 원본 개체를 선택하면 변환된 대상 개체가 자동으로 대상 코드 창에 나타납니다.

③ Zebra Mode : 원본 코드와 대상 코드가 각 행별 또는 블록 별로 컬러로 구분되어 행간에 비교하여 검토할 때 유용합니다. [Default Option] 또는 [Project Options] 창의 [Messages] 탭에서 [Zebra Parameters]를 통해서 컬러를 변경할 수 있습니다. 단, Zebra Mode를 선택한 경우에는 코드 창에서 코드를 직접 수정할 수 없습니다.




또한, SSMA의 레이아웃을 변경하고자 하면 옵션 아랫부분의 [Change Layout]을 선택하면 다음과 같이 SSMA의 레이아웃을 변경할 수 있습니다. 다음의 경우는 변환된 코드를 행 별로 비교할 때 유용한 설정입니다.




다음은 변환된 T-SQL 코드입니다. 시작 부분에 경고나 오류 부분을 주석으로 보여줍니다.

```
/*****
 *  WARNING ORA2MS-4016 line: 2 col: 13: Type number of argument v_empno
was changed to float(53)
 *  WARNING ORA2MS-4016 line: 3 col: 11: Type varchar2 of argument v_job  was
changed to varchar(8000)
 *****/

CREATE  PROCEDURE MIG,SP_UPDATEJOB
      @v_empno float(53),
      @v_job varchar(8000)
AS
/*
-- Generated by SQL Server Migration Assistant for Oracle
-- Please, contact SSMAHELP@microsoft.com or visit
```

```
  http://www.microsoft.com/sql/migration for more information.
--*/
    BEGIN TRY
     UPDATE MIG.EMP
       SET MIG.EMP.JOB = @v_job
       WHERE (MIG.EMP.EMPNO = @v_empno)
     IF (@@TRANCOUNT > 0)
     COMMIT WORK
    END TRY
    BEGIN CATCH
     DECLARE
       @ErrorMessage nvarchar(4000),
       @ErrorNumber int,
       @ErrorSeverity int,
       @ErrorState int,
       @ExceptionIdentifier nvarchar(4000)
     SELECT
        @ErrorMessage = ERROR_MESSAGE( ),
        @ErrorNumber = ERROR_NUMBER( ),
        @ErrorSeverity = ERROR_SEVERITY( ),
        @ErrorState = ERROR_STATE( )
SELECT @ExceptionIdentifier =
       SYSDB.SSMA.db_error_get_oracle_exception_id(@ErrorMessage,
@ErrorNumber)

     IF (@ExceptionIdentifier = N' oracle:{SYS|STANDARD|NO_DATA_FOUND}' )
       BEGIN
        PRINT N' 오류발생'
       END
```

```
ELSE
    BEGIN
     IF (@ExceptionIdentifier IS NOT NULL)
      BEGIN
       RAISERROR (59999, 16, 1, @ExceptionIdentifier)
      END
     ELSE
      BEGIN
       RAISERROR (@ErrorNumber, @ErrorSeverity, @ErrorState)
      END
    END
 END CATCH
```

SSMA는 원본 PL/SQL 프로시저 (그림 〈원본 PL/SQL 코드〉)에 예외절이 선언되어 있으므로 해당 예외절을 SQL Server 2005에서 지원하는 TRY/CATCH 구문으로 변환하였습니다. TRY/CATCH 구문을 통한 오류 처리는 [모듈 6 SQL Server와 Oracle 차이 이해]의 [오류 처리] 부분을 참조하기 바랍니다.

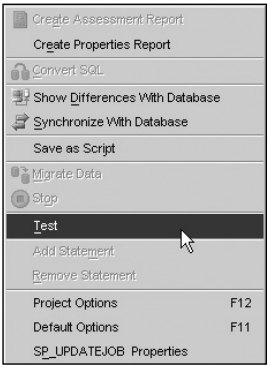
또한, SSMA는 오류 처리를 원본과 동일하게 처리하기 위해서 SYSDB에 생성된 db_error_get_oracle_exception_id 함수를 사용하였습니다. SSMA가 깔끔하게 변환하고자 하였으나 굳이 동일하게 처리할 필요가 없는 경우에는 오류 처리부분을 수동으로 수정합니다.

변환된 T-SQL 구문 개체를 대상 SQL Server에 반영하는 방법은 스키마 개체 마이그레이션 단계와 동일합니다. SSMA에서 해당 개체를 선택하고 [Synchronize With Database]를 선택하거나 [Save as Script]를 선택하여 저장한 뒤 직접 SQL Server에서 실행할 수 있습니다.

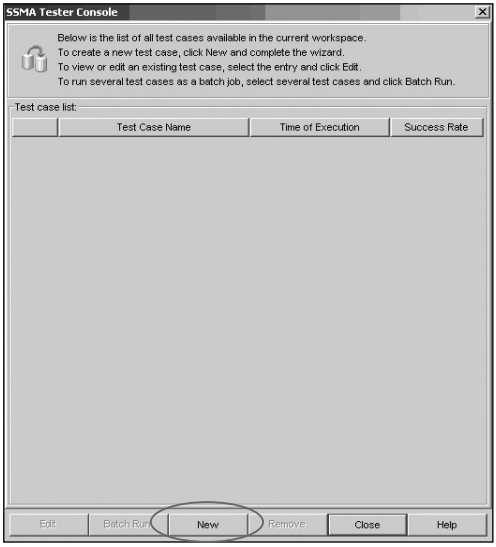
### ■ 유효성 검사 및 테스트

자, 이제 스키마 개체와 데이터, 저장 프로시저 등 PL/SQL 코드 등을 모두 마이그레이션 하였다면 검증 단계가 기다리고 있습니다. 검증 체크리스트를 가지고 하나 하나 꼼꼼하게 검증을 수행해야 합니다. SSMA는 효율적인 검증을 위해서 자동화된 마이그레이션 검증 기능을 제공합니다.

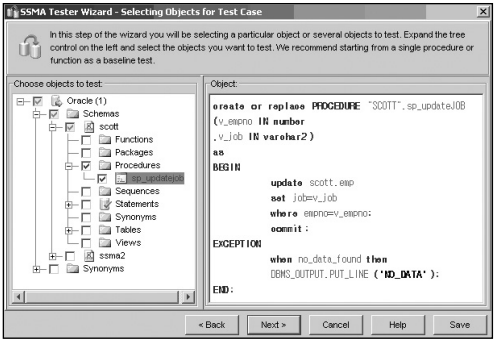
메뉴 바에서 [Tools]를 선택하고 다음과 같이 [Test]를 누릅니다.



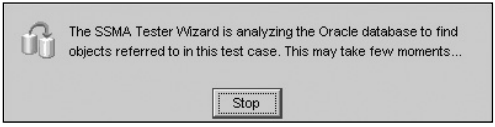
다음과 같은 [SSMA Test Console] 페이지에서 [New] 버튼을 눌러서 새로운 테스트 케이스를 생성하는 SSMA Tester Wizard를 실행합니다.

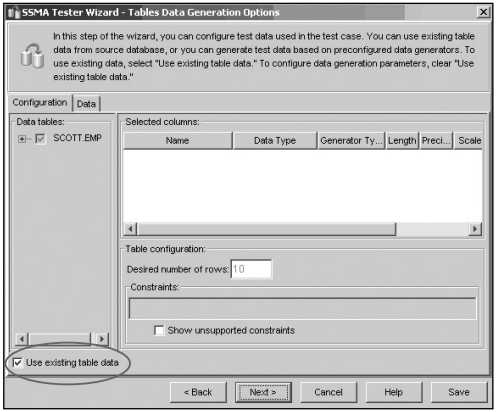



[Next] 버튼을 눌러서 테스트하고자 하는 개체를 선택합니다. 다음은 sp_updaeJOB 저장 프로시저를 테스트합니다.

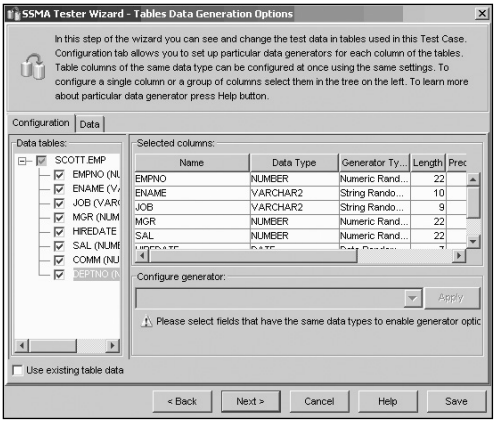


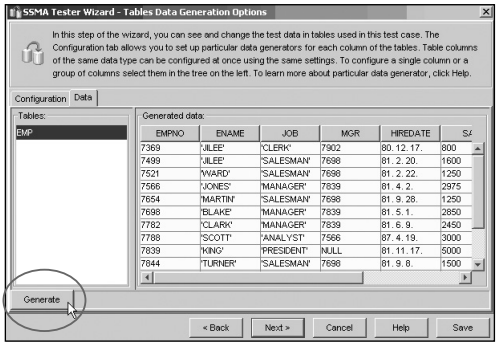

[Next] 버튼을 누르면 테스트하고자 하는 sp_updateJOB 저장 프로시저와 관련된 개체를 분석한다는 메시지 창이 진행되고 이어서, 대상 테이블에 관한 설정 페이지가 실해됩니다.



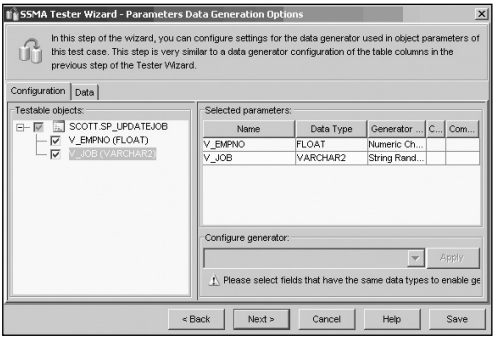



[Tables Data Generations Options]의 [Configuration] 페이지의 아랫 부분의 [Use existing table data] 부분이 선택되면 현재 마이그레이션된 데이터를 사용합니다. 선택을 해제하면 [Configuration] 탭에서 테스트에 필요한 컬럼을 선택할 수 있습니다. 또한 [Data] 탭에서 다음에 표시된 [Generate] 버튼을 눌러서 테스트용 데이터를 새롭게 로딩할 수 있습니다.

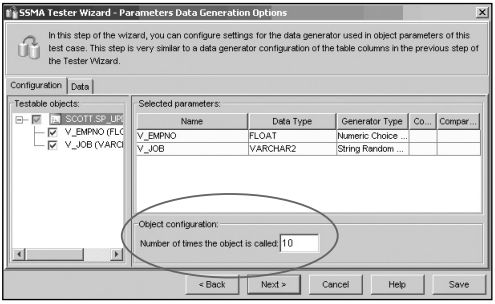

SSMA Tester Wizard - Tables Data Generation Options
In this step of the wizard you can see and change the test data in tables used in this Test Case. Configuration tab allows you to set up particular data generators for each column of the tables. Table columns of the same data type can be configured at once using the same settings. To configure a single column or a group of columns select them in the tree on the left. To learn more about particular data generator press Help button.
Configuration
Data
Data tables:
SCOTT.EMP
EMPNO (NU
ENAME (VA
JOB (VARC
MGR (NUM
HIREDATE
SAL (NUME
COMM (NU
DEPTNO (N
Selected columns:
Name
Data Type
Generator Ty...
Length
Prec
EMPNO
NUMBER
Numeric Rand...
22
ENAME
VARCHAR2
String Rando...
10
JOB
VARCHAR2
String Rando...
9
MGR
NUMBER
Numeric Rand...
22
SAL
NUMBER
Numeric Rand...
22
Configure generator:
Apply
Please select fields that have the same data types to enable generator optic
Use existing table data
< Back
Next >
Cancel
Help
Save


SSMA Tester Wizard - Tables Data Generation Options
In this step of the wizard, you can see and change the test data in tables used in this test case. The Configuration tab allows you to set up particular data generators for each column of the tables. Table columns of the same data type can be configured at once using the same settings. To configure a single column or a group of columns select them in the tree on the left. To learn more about particular data generator, click Help.
Configuration
Data
Tables:
EMP
Generated data:
EMPNO
ENAME
JOB
MGR
HIREDATE
SA
7369
'JILEE'
'CLERK'
7902
80.12.17.
800
7499
'JILEE'
'SALESMAN'
7698
81.2.20.
1600
7521
'WARD'
'SALESMAN'
7698
81.2.22.
1250
7566
'JONES'
'MANAGER'
7839
81.4.2.
2975
7654
'MARTIN'
'SALESMAN'
7698
81.9.28.
1250
7698
'BLAKE'
'MANAGER'
7839
81.5.1.
2850
7782
'CLARK'
'MANAGER'
7839
81.6.9.
2450
7788
'SCOTT'
'ANALYST'
7566
87.4.19.
3000
7839
'KING'
'PRESIDENT'
NULL
81.11.17.
5000
7844
'TURNER'
'SALESMAN'
7698
81.9.8.
1500
Generate
< Back
Next >
Cancel
Help
Save

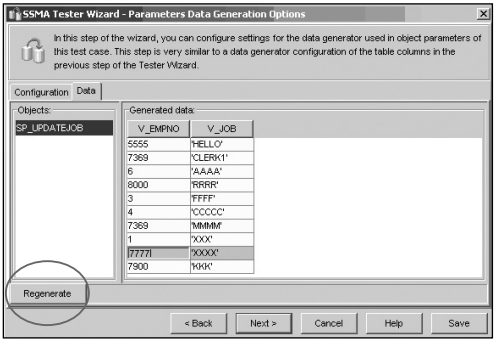
[Next] 버튼을 누르면 [Parameters Data Generation Options] 페이지로 이동해서 테스트하고자 하는 sp_updateJOB 저장 프로시저의 매개 변수 부분을 지정합니다.



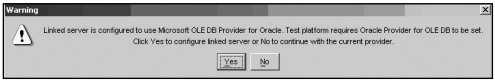

또한, 다음과 같이 테스트하고자 하는 저장 프로시저의 수행 횟수도 지정할 수 있습니다.



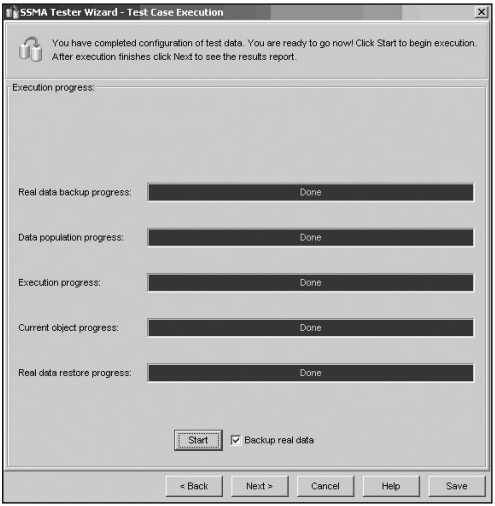
[Data] 탭에서는 다음과 같이 테스트에 사용할 매개 변수를 새롭게 생성하거나 수정할 수 있습니다.




테스트하고자 하는 매개 변수를 편집한 뒤 [Next] 버튼을 누르면 다음과 같이 [Oracle Provider for OLE DB]가 필요하다는 경고 메시지가 나타나지만 무시해도 상관없습니다.



다음은 테스트를 진행하는 페이지 입니다. 총 다섯 개 부분으로 테스트 진행 상황을 확인할 수 있으며 페이지 하단부의 [Start] 버튼을 누르면 테스트를 진행합니다.

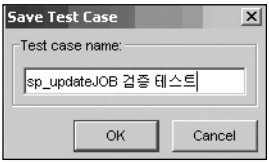



① Real Data Backup Process : 현재의 실제 데이터를 백업하는 과정을 진행합니다.
② Data Population Process : 기존 데이터를 사용하지 않는 경우 테스트용 데이터를 대상 테이블에 입력합니다.
③ Execution Process : 테스트하고자 하는 저장 프로시저를 지정된 매개 변수로 실행합니다. 매 테스트마다 테스트용 데이터를 삭제하고 다시 입력하는 과정도 병행합니다.
④ Current Object Process : 각각 수행된 결과를 비교하는 과정을 수행합니다.
⑤ Real Data Restore Process : 테스트용 데이터를 삭제하고 백업된 실제 데이터를 복구합니다.

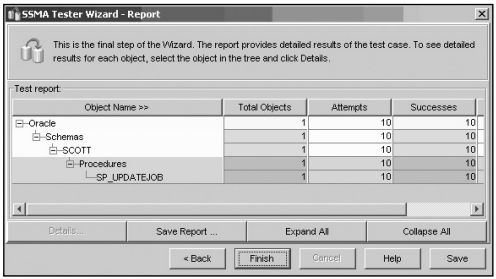
[Test Case Execution] 페이지의 하단부의 [Backup Real Data]부분을 선택하면 ① Real Data Backup Process 단계와 ⑤ Real Data Restore Process 단계를 진행해서 테스트 종료 후에도 테스트를 수행하지 않은 데이터 상태를 유지할 수 있습니다.

**[주의사항]**
SSMA Tester Wizard를 통한 저장 프로시저, 함수 등 데이터를 수정하는 테스트는 지정한 테스트 회수 마다 테스트용 데이터를 삭제하고 다시 입력하는 과정을 반복합니다. 따라서, 용량이 큰 테이블의 경우에는 심각한 성능상의 문제를 유발할 수 있습니다.
따라서 SSMA Tester Wizard를 사용하여 저장 프로시저, 함수 등의 실행을 검증하는 경우에는 각별한 주의를 요구합니다.

[Tester Wizard] 페이지 하단부의 [Save] 버튼은 해당 작업까지의 테스트 케이스를 저장할 때 사용합니다. 다음과 같이 테스트 케이스의 이름을 입력하고 [OK] 버튼을 눌러서 저장합니다.



이제 테스트가 완료되었으면 [Next] 버튼을 눌러서 다음과 같이 테스트 결과에 대한 보고서를 확인합니다.




테스트를 수행한 개체 별로 수행 횟수와 성공 횟수 및 성공 비율 등을 확인할 수 있습니다. 테스트의 수행 횟수는 Tester Wizard의 [Parameters Data Generation Options] 페이지에서 지정하였습니다. 리포트를 저장은 다른 리포트와 마찬가지로 쉼표로 구별된 파일(.csv)로 저장하여 마이크로소프트 Excel과 같은 응용 프로그램에서 사용할 수 있습니다.

# 5. Oracle 기반 데이터 인터페이스 마이그레이션

이번 모듈에서는 기존의 Oracle의 일부 등을 SQL Server 2005로 마이그레이션 한 뒤에 필요한 기존 Oracle 기반 데이터 인터페이스 통합에 대해서 살펴봅니다.
SQL Server는 Oracle 데이터베이스 링크와 유사한 기능을 하는 연결된 서버라는 기능을 통해서 인스턴스간에 연결을 지원하며, 기존에 여러 Oracle간에 복제가 구성되어 있는 경우 Oracle 간의 복제를 SQL Server 2005에서 Oracle 게시자 복제 또는 Oracle 구독자 복제로 Oracle과 SQL Server 2005간에 변경된 사항을 상호간에 반영할 수 있도록 지원합니다. 또한, 플랫 파일 형태의 데이터를 로드하는 Oracle SQL Loader와 같은 기능을 SQL Server에서는 BCP, BULK INSERT, 데이터 가져오기 및 내보내기 마법사와 SQL Server Integration Services 등을 더욱 확장된 기능으로 제공합니다.

## SQL Server에서 Oracle로 연결하기

### ■ 연결된 서버

Oracle에서 여러 개의 인스턴스 환경을 운영하기 위해서 데이터베이스 링크를 구성하여 쿼리하는 것과 같이 SQL Server도 분산된 환경에서 여러 개의 SQL Server 인스턴스를 운영하거나 이기종과의 연결을 위해서 연결된 서버를 사용합니다. 부하를 분산하거나 Oracle, DB2 , 플랫 파일 데이터나 Excel 등 다른 데이터 원본과의 통합 운영을 위해서도 사용하게 됩니다. 연결된 서버로 구성할 수 있는 데이터 원본은 Microsoft가 제공하는 OLE DB 공급자나 독립 소프트웨어 공급 업체가 개발한 드라이버가 제공되어야 합니다. Microsoft는 Oracle용 32Bit OLE DB 공급자를 기본적으로 제공합니다.

연결된 서버는 sp_addlinkedserver 등 시스템 저장 프로시저를 사용하거나 SQL Server Management Studio를 통한 GUI 환경에서 구성할 수 있습니다. 연결된 서버를 구성하기 위해서는 SQL Server가 운영되는 서버에 적절한 Oracle 클라이언트 네트워킹 소프트웨어를 설치하여야 하고 tnsnames.ora 파일 등과 같이 Oracle로 접속할 수 있는 환경이 구성되어야 합니다. 또한 연결을 위해 적절한 권한을 갖는 계정을 준비합니다.

**1) 시스템 저장 프로시저를 사용하여 연결된 서버 구성하기**

시스템 저장 프로시저를 사용하는 경우에는 다음 세가지 단계를 수행합니다.

① 연결된 서버 구성 : sp_addlinkedserver

```
EXEC sp_addlinkedserver [ @server= ] 'server' [ , [ @srvproduct= ] 'product_name' ]
    [ , [ @provider= ] 'provider_name' ]
    [ , [ @datasrc= ] 'data_source' ]
    [ , [ @location= ] 'location' ]
    [ , [ @provstr= ] 'provider_string' ]
    [ , [ @catalog= ] 'catalog' ]
```

② 연결된 서버 보안 구성 : sp_addlinkedsrvlogin

```
EXEC sp_addlinkedsrvlogin [ @rmtsrvname = ] 'rmtsrvname'
    [ , [ @useself = ] 'useself' ]
    [ , [ @locallogin = ] 'locallogin' ]
    [ , [ @rmtuser = ] 'rmtuser' ]
    [ , [ @rmtpassword = ] 'rmtpassword' ]
```

③ 연결된 서버 옵션 구성 : sp_serveroption

```
EXEC sp_serveroption [@server = ] 'server'
    ,[@optname = ] 'option_name'        ,[@optvalue = ] 'option_value' ;
```

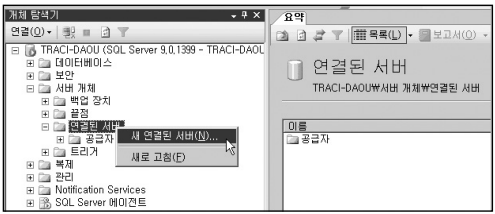
연결된 서버 옵션은 서버간에 데이터 정렬 옵션이 서로 달라서의 데이터 정렬을 지정하는 경우등에 사용하며며 기본 설정 사항을 사용하는 경우에는 일반적으로 수행하지 않아도 됩니다.

이제 실제로 앞에서 살펴본 세 단계를 수행해서 다음과 같이 Oracle로 연결된 서버를 구성합니다.

```
EXEC sp_addlinkedserver
     'ORA01'          ---① 연결하고자 하는 Oracle 서비스 명에 대한 접속 문자열
   , 'Oracle'         ---② 제품명이며 지정하기 나름이고 SQL Server2005부터는
                           반드시 기입해야 합니다.
   , 'MSDAORA'        ---③ OLE DB 공급자 이름
GO
EXEC sp_addlinkedsrvlogin
     'ORA01'          ---① 연결된 서버 이름
   , 'SES'            ---② 연결에 사용하는 SQL Server 계정
   , 'SCOTT'          ---③ 연결에 사용하는 Oracle 계정
   , 'tiger'          ---④ 연결에 사용하는 Oracle 계정의 패스워드
GO
```

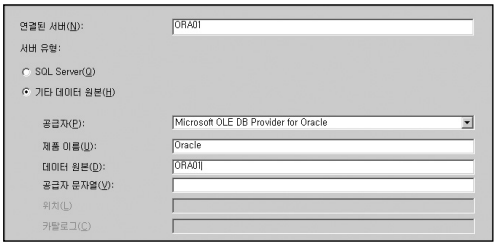
2) SQL Server Management Studio를 사용하여 연결된 서버 구성하기

개체 탐색기→ 서버→ 서버 개체→ 연결된 서버→ 새 연결된 서버



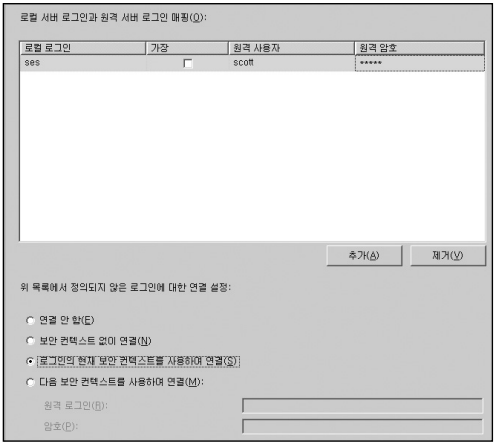
먼저, 일반 탭에서는 연결된 서버의 이름을 지정합니다. 서버 유형에서는 SQL Server가 아니므로 [기타 데이터 원본(H)]을 선택하고 [공급자(P)]에서 Microsoft OLE DB Provider For Oracle을 선택합니다. 제품 이름을 입력하고 [데이터 원본(D)]에 다음과 같은 연결하고자 하는 Oracle 인스턴스의 SQL*Net 별칭을 입력합니다.


〈Oracle Net Manager에서 연결하고자 하는 Oracle 인스턴스 별칭 확인〉



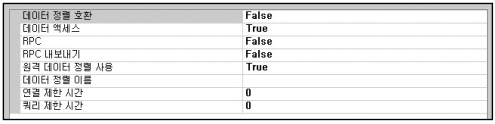

〈연결된 서버 일반 페이지〉

이어서 보안 페이지를 선택하고 다음과 같이 연결하고자 하는 Oracle 데이터베이스에 적절한 권한을 갖는 계정과 로그인 매핑을 지정합니다. 다음은 로컬 ses로그인 계정을 Oracle의 SCOTT 계정과 매핑합니다. 그리고, 매핑되지 않은 계정은 현재의 보안 컨텍스트를 사용해서 Oracle에 접속 시도를 하도록 구성한 경우 입니다.




〈연결된 서버 보안 페이지〉

다음은 옵션 페이지의 서버 옵션의 기본 설정 값으로 데이터 정렬은 원격 서버(Oracle)의 정렬을 사용하며 데이터에 대한 쿼리를 지원하도록 구성됩니다.



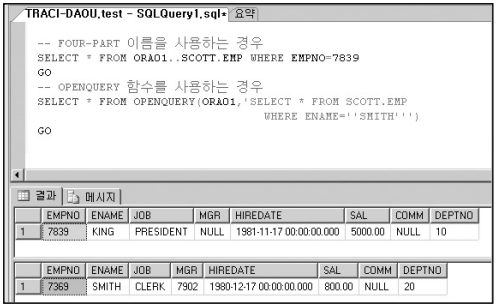
| | |
|---|---|
| 데이터 정렬 호환 | **False** |
| 데이터 액세스 | **True** |
| RPC | **False** |
| RPC 내보내기 | **False** |
| 원격 데이터 정렬 사용 | **True** |
| 데이터 정렬 이름 | |
| 연결 제한 시간 | **0** |
| 쿼리 제한 시간 | **0** |

〈서버 옵션 페이지〉

### 3) 연결된 서버를 통한 Query 실행하기

이제 연결된 서버 구성이 완료되었으면 다음과 같이 연결된 서버에 쿼리를 실행하여 올바른 구성 여부를 확인합니다.

연결된 서버를 통해서는 Four-Part 이름을 사용하는 쿼리나 OPENQUERY 함수를 사용하는 쿼리를 수행하며, Four-Part 이름을 사용하는 쿼리에서 Four-Part 이름은 [인스턴스 이름].[데이터베이스 이름].[스키마 이름].[개체 이름]으로 구성되며 Oracle의 경우에는 ORA01..SCOTT.EMP와 같이 [인스턴스 이름] 부분에 연결된 서버 이름을 지정하고 [데이터베이스 이름] 부분은 비워두며 모두 대문자를 사용해야 합니다.




OPENQUERY함수를 사용하는 쿼리는 테이블 이름처럼 FROM 절 뒤에 OPENQUERY함수를 연결된 서버이름과 수행할 쿼리를 매개변수로 지정하여 사용합니다.

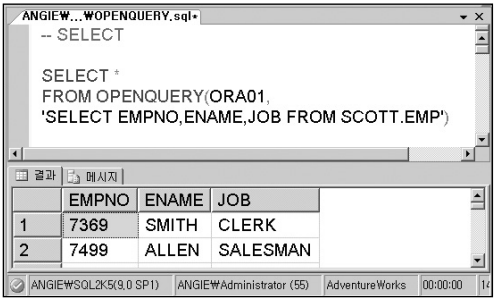
```
FROM OPENQUERY(연결된 서버이름, '쿼리')
```

연결된 서버의 쿼리는 성능적인 관점에서 OPENQUERY가 Four-Part 이름을 사용하는 경우보다 효율적인 경우가 많습니다. 따라서, 대상 인스턴스에서 효율적인 실행 계획을 작성되어 쿼리가 실행되는 지 면밀히 검토하고 쿼리를 작성해야 합니다.

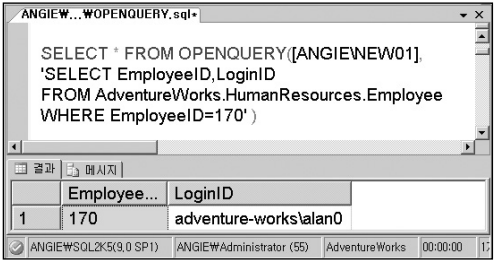
다음은 OPENQUERY를 사용하는 쿼리의 예입니다.

> ***[참고]***
> *OPENQUERY의 보다 자세한 사항은 온라인 설명서를 참조하시기 바랍니다.*

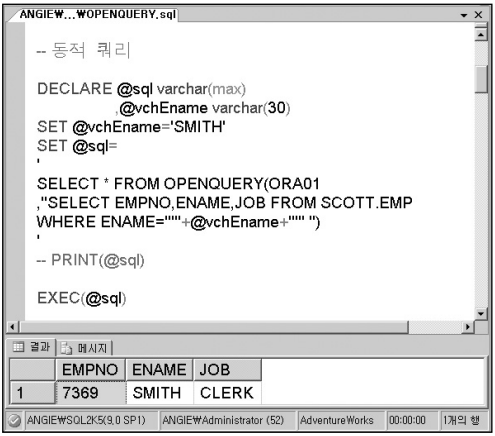
① Oracle과 연결된 서버를 통한 SELECT 구문



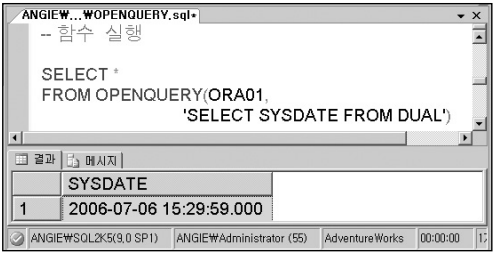

② SQL Server와 연결된 서버를 통한 단순 SELECT 구문



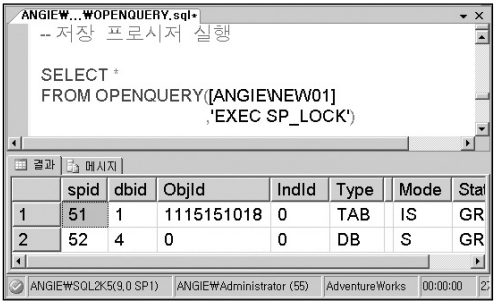
③ 동적 SQL 구문




④ 함수 실행 구문

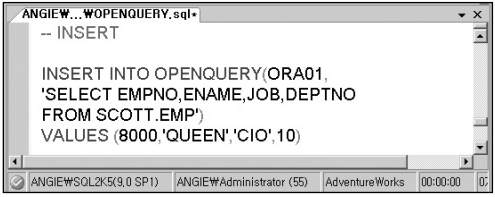


⑤ 저장 프로시저 실행 구문

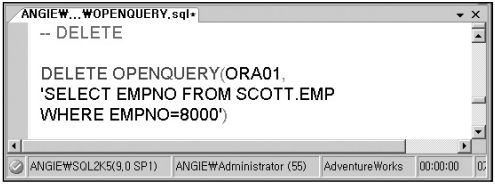



* Oracle과의 연결된 서버에 대한 저장 프로시저의 실행은 지원하지 않습니다.

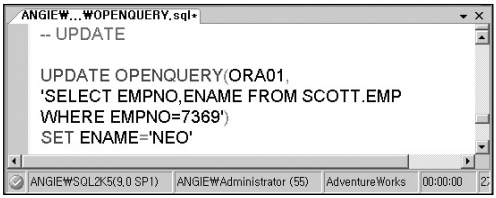
⑥ INSERT 구문



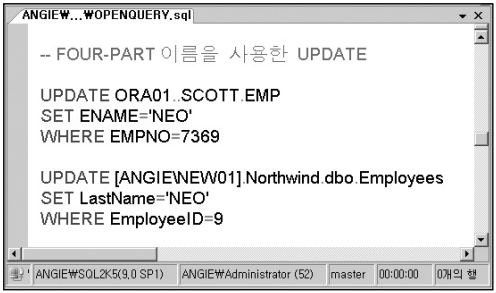

⑦ DELETE 구문



⑧ UPDATE 구문




DELETE와 UPDATE 구문을 실행하는 경우에는 다음과 같이 Four-Part 이름을 사용하는 쿼리가 더욱 효율적인 경우가 많습니다.

## ■ Oracle 게시자 트랜잭션 복제

SQL Server의 복제 토폴로지는 원본 서버와 대상 서버 그리고, 복제 메타데이터를 저장하고 복제의 핵심 기능을 처리하는 서버로 구성됩니다. 이와 같은 역할을 하는 서버를 각각 게시자, 구독자, 배포자라고 합니다. SQL Server 2005 이전 버전은 Oracle을 복제의 구독자로 구성하여 스냅숏 복제와 표준 트랜잭션 복제를 통해서 SQL Server 게시자에서 변경된 데이터를 Oracle이 구독하는 기능을 제공합니다. SQL Server 2005는 Oracle(8.0.5 버전 이상)을 복제의 게시자로 지정하여 Oracle이 게시한 개체의 행에 대한 변경 내용을 SQL Server 구독자로 적용할 수 있습니다. 지원하는 복제 유형은 스냅숏 복제와 표준 트랜잭션 복제입니다.
따라서, 이와 같은 SQL Server 2005의 Oracle 게시자 복제를 이용하여 SQL Server와 Oracle의 통합 운영이나 데이터 마이그레이션에 적용할 수 있습니다.

SQL Server 복제에 관한 자세한 사항은 온라인 설명서나 다음의 웹사이트를 참조합니다.

http://msdn2.microsoft.com/en-us/library/ms151229.aspx

### 1) Oracle 게시자 복제의 제한 사항

- 테이블, 인덱스, IOT, 구체화된 뷰에 대한 복제를 지원합니다.

- 기본적으로 Oracle 개체 이름은 대문자로 생성됩니다. Oracle 데이터베이스에서 Oracle 개체 이름이 대문자인 경우 이 개체를 SQL Server 배포자를 통해 게시할 때 해당 이름을 대문자로 입력해야 합니다. 대/소문자를 올바르게 입력하지 않으면 개체를 찾을 수 없다는 오류 메시지가 나타납니다.

- 트랜잭션 복제인 경우에 LOB(큰 개체) 데이터는 LOB을 포함하는 행이 삽입 또는 삭제되거나 LOB 열이 아닌 컬럼이 업데이트되는 경우에 복제됩니다. 따라서, LOB 데이터만 업데이트 되지 않도록 해야 합니다.

- 트랜잭션 복제는 Oracle의 트리거를 기반으로 변경 사항을 추적합니다. 따라서, TRUNCATE 구문이나 Direct Path Load와 같이 트리거가 동작하지 않는 작업이 수행되면 변경 사항이 반영되지 않습니다.

- SQL Server와 Oracle의 최대 용량 사양의 차이 및 개체의 차이를 이해하고 처리해야 합니다. 본 가이드의 [모듈6 SQL Server와 Oracle 차이 이해] 부분을 참조합니다.

- 복제의 초기화는 백업에서 초기화할 수 없습니다.

- 게시된 Oracle 테이블의 스키마 변경은 지원되지 않습니다. 스키마를 변경하려면 먼저 게시를 삭제하고 스키마를 변경한 다음 게시 및 구독을 다시 만듭니다.

**2) Oracle 게시자 복제의 구성**

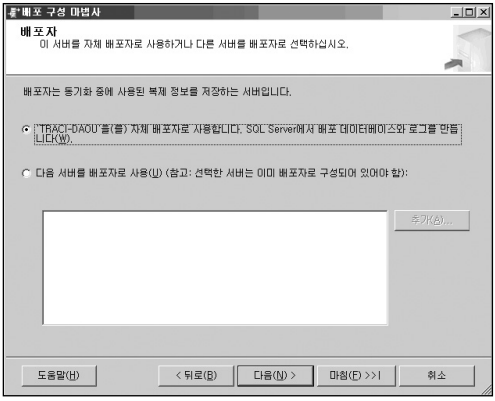
SQL Server 복제는 다음의 순서로 구성한다.

1. 배포자
2. 게시자 구성
3. 게시 생성
4. 구독자 및 구독 구성

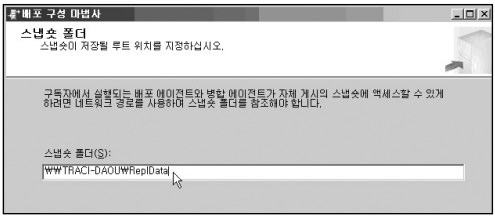
**3) 배포자 구성**

다음과 같이 개체 탐색기에서 서버를 확장한 다음 복제를 선택하고 오른쪽 버튼을 눌러서 [배포 구성(C)]을 선택합니다.

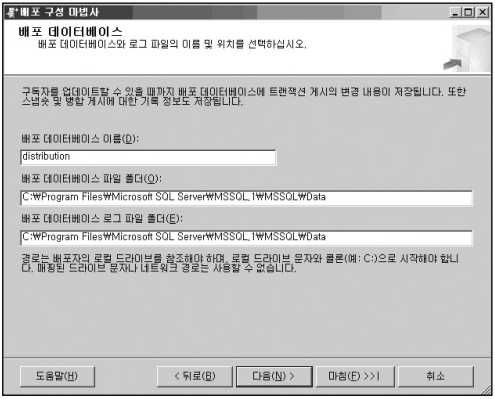
다음은 [TRACI-DAOU] 서버를 배포자로 사용하는 경우입니다. 배포자로 기본 선택된 현재 서버를 확인하고 [다음(N)] 버튼을 눌러서 [스냅숏] 페이지로 이동합니다.



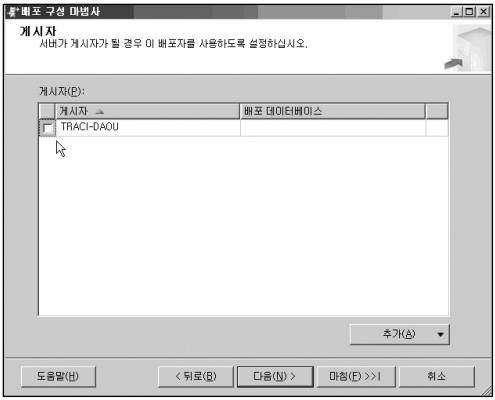

[스냅숏 폴더(S)]는 스냅숏을 구성하기 위한 플랫 파일을 저장하는 폴더로 기본 경로는 [₩Program Files₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩repldata]로 관리자 공유($ 공유)를 사용해서 공유하지 말고 별도의 공유를 지정합니다. 다음은 [₩₩TRACI-DAOU₩ReplData]라는 공유 이름을 사용한 경우 입니다. 스냅숏 폴더의 공유를 지정하고 [다음(N)] 버튼을 눌러서 [배포 데이터베이스 구성] 페이지로 이동합니다.



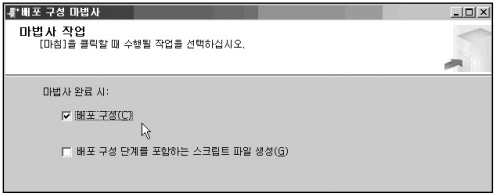
배포 데이터베이스 이름 지정을 확인합니다. 기본 이름은 [배포]입니다. 이어서 배포 데이터베이스의 설치 경로를 확인합니다. 게시가 많은 경우 충분한 공간이 확보된 경로를 지정합니다.

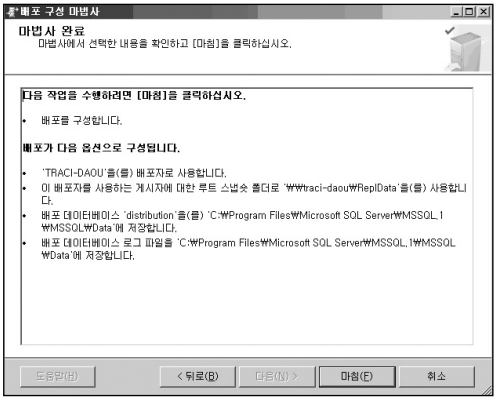



이어서 다음과 같은 게시자 페이지에서 기본으로 선택되어 있는 서버의 선택을 해제합니다. 페이지 하단의 [추가(A)] 메뉴에서 게시자를 추가할 수 있습니다. 게시자는 뒤에서 별도로 추가할 것이므로 [다음(N)] 버튼을 눌러서 [마법사 작업] 페이지로 이동합니다.




[마법사 작업] 페이지에서는 현재 배포를 구성할 것인지 아니면 스크립트만 생성할 것인지 선택할 수 있습니다. 두 가지 모두 선택하여 동시에 작업할 수도 있습니다. [배포 구성(C)]만 선택하고 [다음(N)] 버튼을 눌러서 [마법사 완료] 페이지로 이동하여 요약 사항을 점검하고 [마침(F)] 버튼을 눌러서 배포 구성을 완료합니다.






#### 4) 게시자 생성

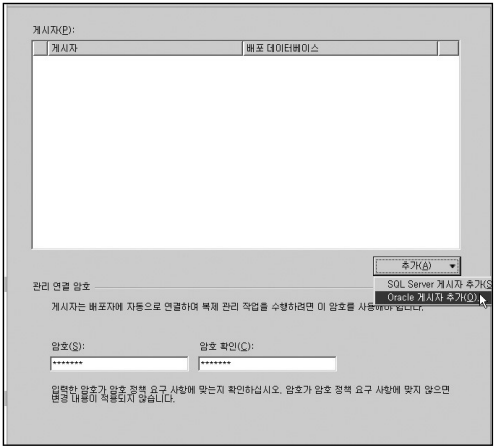
Oracle 게시자에서 게시는 SQL Server 게시의 스냅숏 및 트랜잭션 게시가 만들어지는 것과 같은 방식으로 만들어지지만 Oracle 게시자에서 먼저 SQL Server 설치 경로의 Install 폴더에 있는 oracleadmin.sql 스크립트를 수행해야 합니다. oracleadmin.sql 스크립트는 사용자 생성과 오라클 게시를 위한 적절한 권한을 부여합니다. 단, 임시 테이블스페이스 지정 구문이 없으므로 해당 Oracle에서 기본 임시 테이블스페이스 지정이 없는 경우 다음과 같이 별도의 스크립트를 수행하는 것이 바람직합니다.

다음의 SQLRepl이라는 계정을 임시 테이블스페이스를 지정하여 생성하고 게시자 구성을 위한 적절한 부여하는 스크립트 입니다. 권한 부여시 예제와 같이 SELECT ANY TABLE 권한을 부여하지 말고 게시되는 테이블에 대해서만 SELECT 권한을 부여합니다.

```
create user SQLRepl
identified by lperlqs
default tablespace users
```

```
temporary tablespace temp
quota unlimited on users;
GRANT create public synonym TO SQLRepl;
GRANT drop public synonym TO SQLRepl;
GRANT create procedure TO SQLRepl;
GRANT create sequence TO SQLRepl;
GRANT create session TO SQLRepl;
GRANT create any trigger TO SQLRepl;
GRANT create table TO SQLRepl;
GRANT create view TO SQLRepl;
GRANT select any table TO SQLRepl;
```

게시자의 생성은 배포자 생성 과정의 게시자 페이지에서 추가할 수도 있으며, 배포자가 생성된 뒤에 배포자의 속성에서 다음과 같이 게시자 페이지에서 추가할 수 있습니다.



[Oracle 게시자 추가(A)] 버튼을 누르게 되면 다음과 같이 Oracle에 접속하기 위한 창에서 복제 용으로 Oracle에 생성한 계정의 아이디와 패스워드를 입력합니다.


인증이 완료되면 다음과 같이 Oracle 게시자가 추가 됩니다.
[확인] 버튼을 눌러서 Oracle 게시자 추가를 완료합니다.


Oracle 게시자를 구성하면 다음 작업이 이루어 집니다.

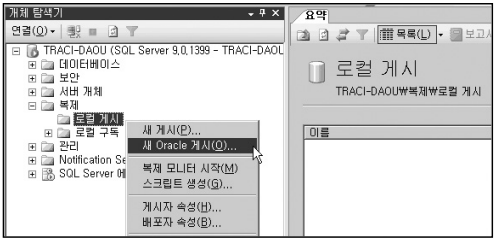
1. Oracle 복제 전용의 연결된 서버가 생성됩니다. 이 때 연결 계정은 Oracle 복제 용 사용자 스키마(계정)가 됩니다. 앞의 예에서는 SQLRepl 계정입니다.
2. Oracle 복제용 사용자 스키마(계정)로 다음의 테이블, 뷰, 시퀀스, 패키지 등을 생성합니다.

| 개체 이름 | 개체 유형 | 설 명 |
|---|---|---|
| HREPL_Changes | 테이블 | 트랜잭션 집합에 할당되기 위해 대기 중인 변경 내용 수를 확인하기 위해 Xactset 작업에서 내부적으로 사용되는 테이블입니다. |
| HREPL_Distributor | 테이블 | Oracle 게시자와 연결된 SQL Server 배포자에 대한 정보를 유지 관리하는 데 사용되는 배포자 상태 테이블입니다. |
| HREPL_Event | 테이블 | 스냅숏과 행 개수 요청을 동기화하는 데 사용되는 이벤트 테이블입니다. |
| HREPL_Mutex | 테이블 | Oracle 패키지 프로시저인 PopulatePollTable이 로그 판독기 에이전트와 데이터베이스 작업 모두에 의해 동시에 실행되지 않도록 하는 데 사용되는 테이블입니다. |
| HREPL_Poll | 테이블 | 게시된 테이블에 대한 변경 내용 집합과 연결된 로그 테이블 항목을 식별하는 데 사용되는 테이블입니다. |
| HREPL_PublishedTables | 테이블 | 트랜잭션 게시의 각 아티클에 대한 항목을 포함하는 테이블입니다. |
| HREPL_Publisher | 테이블 | 게시자별 정보를 유지 관리하는 데 사용되는 게시자 상태 테이블입니다. |
| HREPL_SchemaFilter | 테이블 | 새 게시 마법사를 통해 게시할 때 표시되지 않는 스키마를 포함하는 테이블입니다. |
| HREPL_ XactsetCreateTimes | 테이블 | 각 트랜잭션 집합과 연결된 작성 시간을 식별하는 테이블입니다. |
| HREPL_XactsetJob | 테이블 | Xactset 작업에 대한 현재 매개 변수 설정을 포함하는 테이블입니다. |

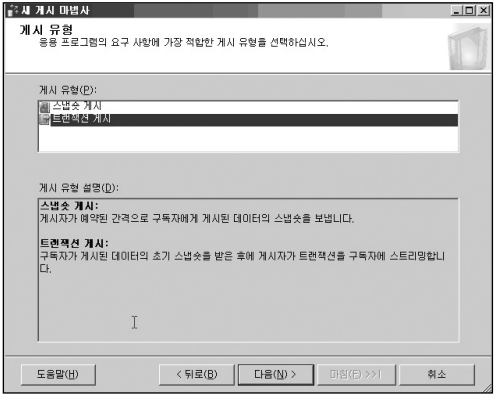
| | | |
|---|---|---|
| HREPL_Pollid | 시퀀스 | 폴링 ID를 생성하는 데 사용되는 시퀀스입니다. |
| HREPL_Seq | 시퀀스 | 명령 변경의 순서를 지정하는 데 사용되는 시퀀스입니다. |
| HREPL_Seq | 시퀀스 | 명령 변경의 순서를 지정하는 데 사용되는 시퀀스입니다. |
| HREPL_Stmt | 시퀀스 | 문 ID를 생성하는 데 사용되는 시퀀스입니다. |
| HREPL | 패키지 및 패키지 본문 | 게시자에서 생성된 게시자 지원 코드의 패키지입니다. |
| MSSQLSERVERDISTRIBUTOR | 공용 동의어 | HREPL_Distributor 테이블에 대한 공용 동의어입니다. Oracle 게시자와 함께 사용하도록 배포자를 구성하면 이 동의어가 이미 데이터베이스에 있는 경우 동의어가 삭제되고 다시 생성됩니다.CASCADE 옵션으로 공용 동의어와 구성된 Oracle 복제 사용자를 삭제하면 Oracle 게시자에서 모든 복제 개체가 제거됩니다. |
| HREPL_DropPublisher | 프로시저 | Oracle 게시 패키지 코드 외부에 정의되어 Oracle 게시자를 삭제하는 데사용되는 프로시저입니다. |
| HREPL_ ExecuteCommand | 프로시저 | Oracle 게시 패키지 코드 외부에 정의되어 게시자에서 명령을 실행하는 데 사용되는 프로시저입니다. |

### 5) 게시 생성

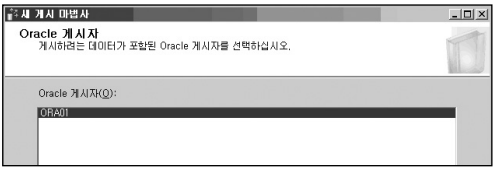
다음과 같이 개체 탐색기→ 서버→ 복제→ 로컬 게시의 오른쪽 버튼을 눌러서
[새 Oracle 게시(O)]를 선택합니다.




게시 마법사가 실행되면 [게시 유형] 페이지에서 적합한 게시 유형을 선택합니다.
게시 유형은 페이지 하단의 [게시 유형 설명(D)] 섹션에서 간단한 설명을 확인할 수 있습니다.
[트랜잭션 게시]를 선택하고 [다음(N)] 버튼을 누릅니다.

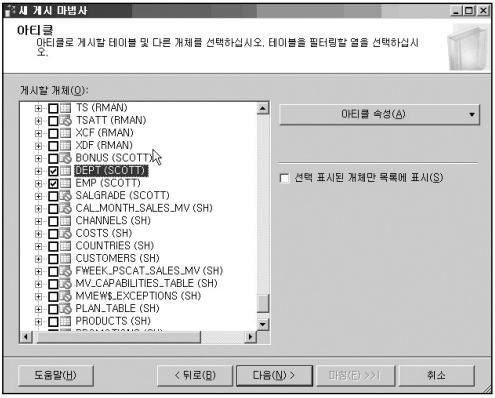


[Oracle 게시자] 페이지에서 Oracle게시자를 선택합니다.

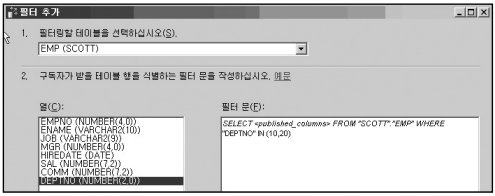



① [아티클] 페이지에서 게시하고자 하는 스키마 개체를 선택합니다. SQL Server 복제는 게시하는 스키마 개체를 각각 아티클(Article)이라고 표현합니다. 다음과 같이 EMP 테이블과 DEPT 테이블을 선택합니다. 트랜잭션 복제를 하기 위해서는 해당 아티클에 기본 키 제약 조건을 가지고 있어야 합니다. 열 필터를 정의 하기 위해서는 해당 아티클을 확장하고 필터링할 각 열 옆에 있는 확인란의 선택을 취소합니다.

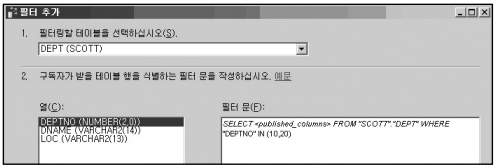
[다음(N)] 버튼을 누르고 [행 필터] 페이지로 이동합니다.



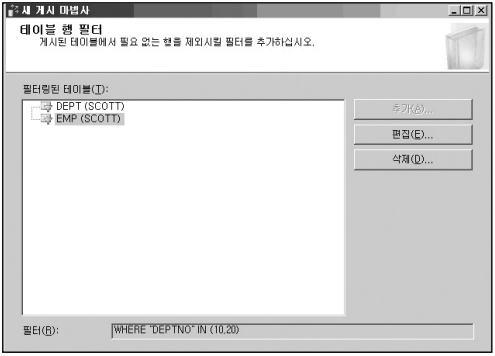
[행 필터] 페이지에서 추가 버튼을 누르고 다음과 같이 EMP(SCOTT) 테이블을 선택하고 페이지 오른쪽 부분의 [필터 문(F)]에 DEPTNO 컬럼에 대한 필터 값을 입력합니다. 다음은 DEPTNO가 10과 20인 값만 게시하는 내용입니다.



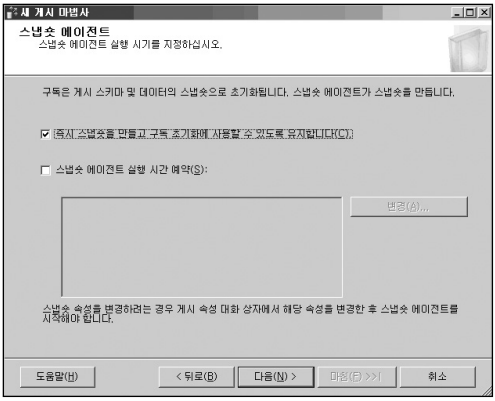

마찬 가지로 DEPT(SCOTT) 테이블에도 데이터의 무결성이 손상되지 않도록 동일한 필터를 적용합니다.



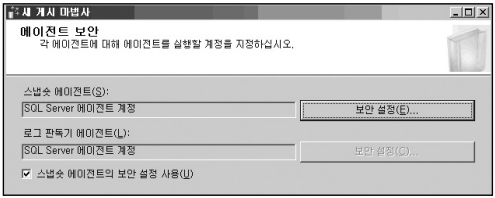
다음과 같이 [테이블 행 필터] 페이지에서 각 아티클에 필터 정보를 확인할 수 있습니다.



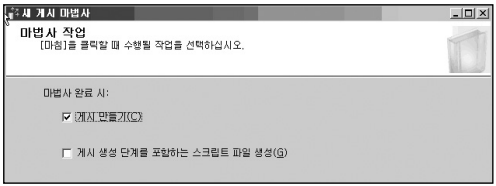

[다음(N)] 버튼을 눌러서 [스냅숏 페이지]로 이동합니다. 트랜잭션 복제를 위해서는 [즉시 스냅숏을 만들고 구독 초기화에 사용할 수 있도록 유지합니다(C).]를 선택합니다. 스냅숏 복제를 하는 경우에는 [스냅숏 에이전트 실행 시간 예약(S)]를 선택하고 정기적으로 수행할 일정을 지정할 수 있습니다. 트랜잭션 복제는 구독 초기화를 필요로 합니다. SQL Server간의 복제는 구독 초기화를 위해서 백업이나 SQL Server Integration Service를 이용할 수 있지만 Oracle 게시자 복제는 반드시 스냅숏을 통해서만 초기화할 수 있습니다.

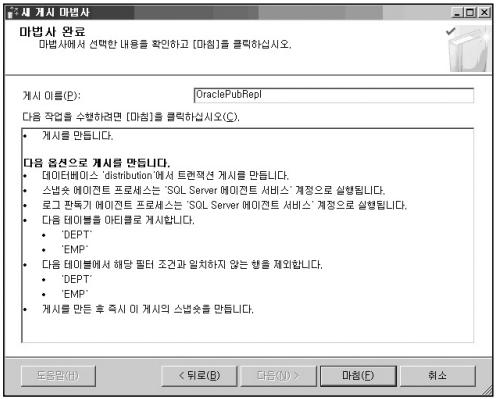



[에이전트 보안] 페이지에서 [보안 설정(E)] 버튼을 눌러서 스냅숏 에이전트 계정을 지정합니다. SQL Server 2005는 복제 에이전트 계정 지정 시 본 가이드의 [자동화 작업] 부분에서 설명한 복제용 프록시 사용을 권장합니다. SQL Server 에이전트 계정은 복제를 위한 권한 이상을 보유하고 있으므로 최소 권한 부여의 보안 기본 원칙을 위반할 수 있습니다.



[마법사 작업] 페이지에서 [게시 만들기(C)]가 선택된 것을 확인하고 [다음(N)] 버튼을 눌러서 [마법사 완료] 페이지의 요약 사항을 확인하고 [마침(F)] 버튼을 눌러서 게시를 생성합니다.



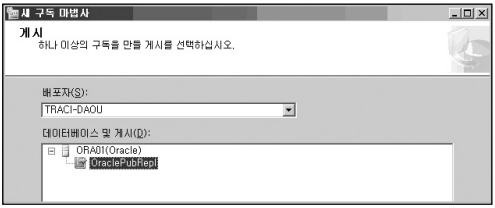



게시 구성이 완료되면 Oracle 게시자에는 다음과 같은 테이블과 함수, 트리거, 뷰가 생성됩니다.

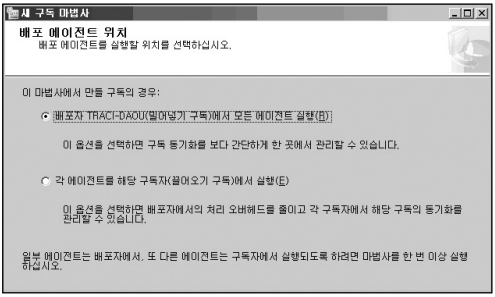
| 개체 이름 | 개체 유형 | 설 명 |
|---|---|---|
| HREPL_ArticleNlog_V | 테이블 | 게시된 테이블에 변경 내용이 적용될 때 정보를 저장하는 데 사용되는 변경 내용 추적 테이블입니다. 변경 내용 추적 테이블은 게시된 각 테이블에 대해 생성됩니다. |
| HREPL_Len_I_J_K | 함 수 | Oracle 게시 패키지 코드 외부에 정의되어 LONG 열의 길이에 대한 쿼리에 사용되는 함수입니다. 게시된 LONG 열이 있는 테이블에 대한 매개 변수가 있는 명령을 생성할 때 사용됩니다. LONG 열이 있는 게시된 각 테이블에 대해 함수가 생성됩니다. |
| HREPL_ArticleN_ Trigger_Row | 트리거 | 게시된 각 테이블에 대해 생성되어 행 변경 내용을 추적하는 데 사용되는 트리거입니다. |
| HREPL_ArticleN_ Trigger_Stmt | 트리거 | 게시된 각 테이블에 대해 생성되어 문 수준 변경 내용을 추적하는 데 사용되는 트리거입니다. |
| HREPL_Article_I_J | 뷰 | 게시된 각 테이블에 대해 생성되어 게시된 테이블을 쿼리하는 데 사용되는 뷰입니다. |
| HREPL_Log_I_J_K | 뷰 | 게시된 각 테이블에 대해 생성되어 변경 내용 추적 테이블을 쿼리하는 데 사용되는 뷰입니다. |

### 6) 구독자 및 구독 생성

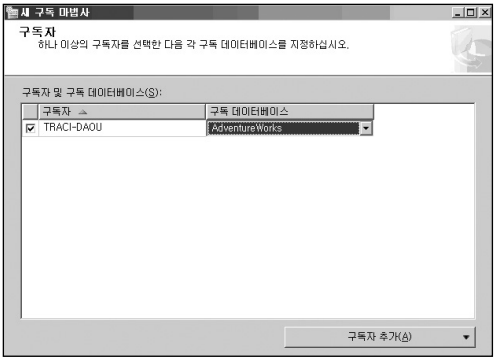
이제 Oracle에서 게시된 아티클을 구독할 구독자와 구독을 생성합니다.
개체 탐색기→ 서버→ 복제→ 로컬 구독의 오른쪽 버튼을 눌러서 [새 구독(S)]을 선택합니다.
다음과 같이 새 구독 마법사가 실행되면 [게시] 페이지에서 구독할 Oracle 게시를 선택합니다.




이어서 배포 작업을 수행하는 배포 에이전트의 위치를 지정합니다. [배포자에서 모든 에이전트 실행(R)] 기본 선택을 확인하고 [다음(N)] 버튼을 눌러서 [구독자] 페이지로 이동합니다.

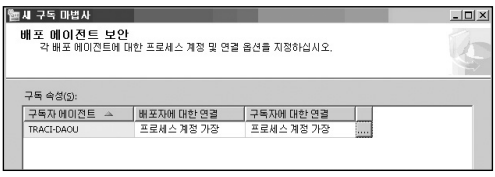


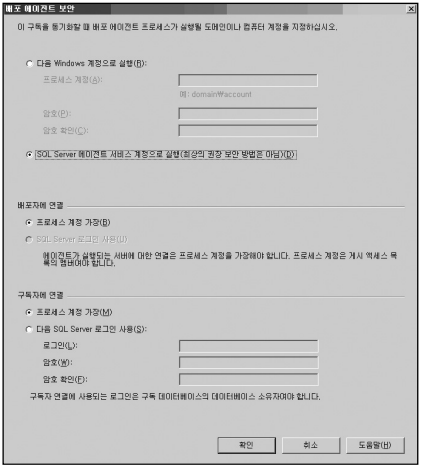

[구독자] 페이지에서 구독 서버를 선택하고 구독하고자 하는 데이터베이스를 선택합니다.
예제는 [AdventureWorks] 샘플 데이터베이스를 구독 데이터베이스로 사용합니다.
여러 서버를 구독자로 지정하고자 하는 경우에는 페이지 하단의 [구독자 추가(A)] 버튼을 눌러서 구독자를 추가할 수 있습니다. [다음(N)] 버튼을 눌러서 [배포 에이전트 보안]페이지로 이동합니다.



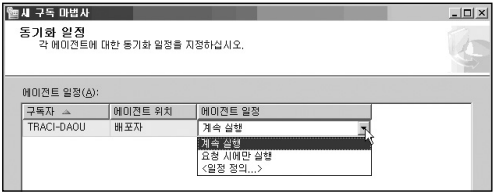

[배포 에이전트 보안] 페이지 에서는 배포 업무를 담당하는 배포 에이전트를 실행하는 계정과  배포자와 구독자에 연결하기 위한 보안 사항을 설정합니다. [구독자에 대한 연결] 옆의 […] 버튼을 눌러서 [배포 에이전트 설정] 팝업 화면을 실행합니다.

배포 에이전트를 실행하는 계정은 앞에서 게시 생성 부분에서 스냅숏 에이전트 보안 부분에서 살펴본 바와 마찬 가지로 복제용 프록시 사용을 권장합니다. SQL Server 에이전트 계정은 복제를 위한 권한 이상을 보유하고 있으므로 최소 권한 부여의 보안 기본 원칙을 위반할 수 있습니다. 본 예제에서는 쉽게 구성할 수 있는 SQL Server 에이전트 서비스 계정을 사용합니다. 프록시에 대한 자세한 사항은 본 가이드의 [자동화 작업] 부분이나 온라인 설명서를 참조하기 바랍니다.

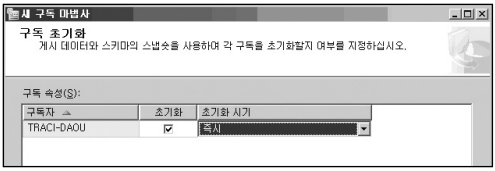




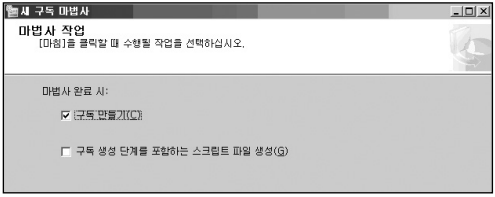

이어서 에이전트 일정을 지정합니다. 게시자에서 트랜잭션이 수행되면 즉시 구독자로 전송하고자 한다면 [에이전트 일정]에서 [계속 수행]을 지정하고 지정된 시간마다 수행하도록 일정을 지정하고자 한다면 [〈일정 정의…〉] 버튼을 눌러서 세부 일정을 지정할 수 있습니다.

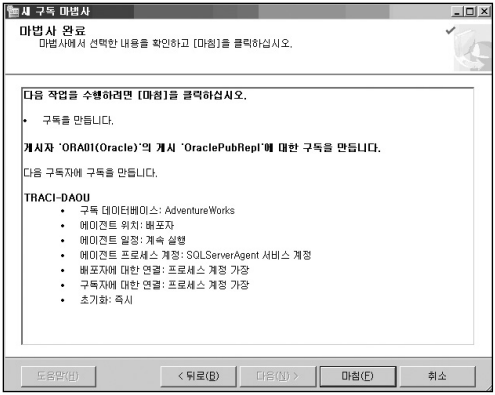



이어서 [구독 초기화] 페이지에서 구독을 초기화 하기 위한 스냅숏 에이전트의 실행 여부와 초기화 시기를 결정합니다. 대용량의 테이블인 경우에는 수동으로 초기화를 하기 위해서 [초기화]의 선택을 해제할 수 있습니다.

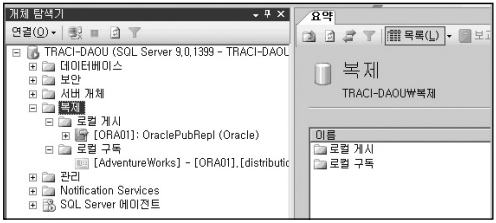



[마법사 작업] 페이지에서 [구독 만들기] 또는 [구독 생성 단계를 포함하는 스크립트 파일 생성(G)]을 선택할 수 있습니다. 구독 생성 스크립트를 편집하여 여러 구독에 적용하고 자 할 때 스크립트를 생성해서 편집하여 적용하면 편리합니다.



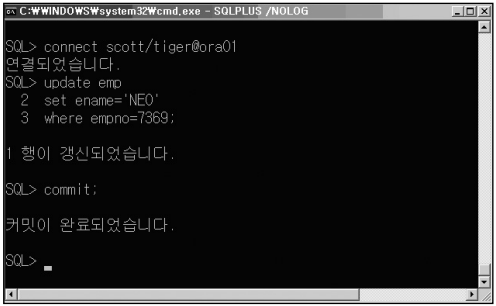



[마법사 완료] 페이지에서 요약 사항을 확인하고 [마침(F)] 버튼을 눌러서 구독을 생성하고 다음과 같이 구독 생성 여부를 확인합니다.

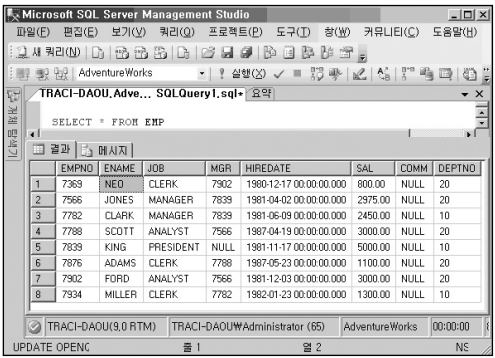



### 7) 복제 테스트

다음과 같이 Oracle 게시자에서 게시 아티클에 대한 수정 작업을 수행합니다.

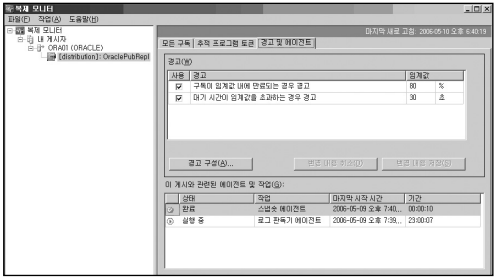


이어서 SQL Server 구독자에서 게시자의 수정 사항이 반영 여부를 확인합니다.




복제가 이상 없이 수행되는 것을 확인하였다면 다음과 같이 복제 모니터를 실행해서 복제의 수행 상태와 성능을 모니터합니다.


> ***[참고]***
> *복제 모니터링은 온라인 설명서의 [복제 모니터로 복제 모니터링]의 설명을 참조하거나 [Microsoft SQL Server 2005 관리자 가이드]를 참조하기 바랍니다.*




## Oracle에서 SQL Server로 연결하기

### ■ Transparent Gateway

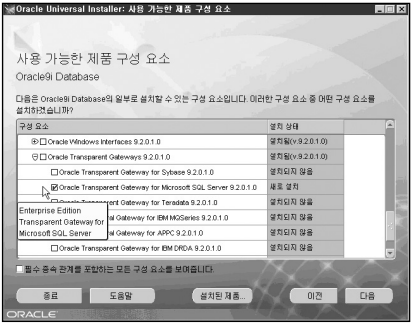
Oracle은 이기종과의 연결 작업을 위해서 Gateway Service 기능을 제공합니다. Oracle9i 이전 버전에는 TG4MSQL이라는 이름으로 별도로 제공되던 제품이 Oracle9i 버전부터 제품 CD에 포함되어 있습니다. 단, 설치 및 사용을 위해서는 반드시 별도의 라이센스를 구매해야만 합니다. 다음에서 Oracle Transparent Gateway For Microsoft SQL Server (이하 TG4MSQL)를 구성하고 TG4MSQL을 통해서 SQL Server와 통합된 쿼리를 사용하는 방법에 대해서 살펴봅니다.

TG4MSQL 구성은 원본 Oracle 인스턴스 및 TG4MSQL 인스턴스, 그리고 연결 대상 SQL Server로 구성되며 다음 다섯 단계로 진행하여 설치 및 구성을 완료합니다.

① Oracle Universal Installer를 통한 Transparent Gateway For Microsoft SQL Server 설치 및 설정
② SQL Server 및 원본 Oracle 인스턴스에서 각각 TG4MSQL용 계정 생성 및 권한 부여
③ TG4MSQL 인스턴스 리스너 구성
④ 원본 Oracle 인스턴스에서 TG4MSQL 인스턴스에 대한 tnsname.ora 파일 구성
⑤ 원본 Oracle 인스턴스에서 TG4MSQL 인스턴스에 대한 데이터베이스 링크 구성
⑥ 원본 Oracle 인스턴스에서 데이터베이스 링크를 통한 연결 대상 SQL Server로 쿼리 수행

1) TG4MSQL 설치

다음과 같이 Oracle Universal Installer를 실행하고 사용 가능한 제품 구성 요소 선택 페이지에서 Oracle Transparent Gateway For Microsoft SQL Server를 선택하고 [다음] 버튼을 누릅니다.




설치 과정을 계속 진행하다가 Microsoft SQL Server 및 데이터베이스 이름 지정 페이지에서 다음과 같이 연결하고자 하는 SQL Server 이름과 데이터베이스 이름을 입력합니다.

이 사항은 TG4MSQL을 설치 완료한 뒤 %ORACLE_HOME%의 tg4msql\admin 폴더의 initTG4MSQL.ora 초기 매개변수 파일에서 수정할 수 있습니다. 주의 사항은 명명된 인스턴스의 경우에는 TG4MSQL이 인식하지 못하므로 SQL Server Configuration Manager 에서 서버의 별칭을 생성하여 접속해야 합니다.

**2) TG4MSQL 인스턴스 리스너를 구성**

설치가 완료되면 다음과 같은 내용을 추가하여 TG4MSQL 인스턴스 리스너를 구성합니다. TG4MSQL 인스턴스 리스너에 다음과 같은 내용을 추가합니다.

```
SID_LIST_LISTENER =
    (SID_LIST =
        (SID_DESC =
            (SID_NAME = tg4msql) ←지정한 SID
            (ORACLE_HOME = G:₩oracle₩ora92) ←설치 시 지정한 홈 폴더

            (PROGRAM = tg4msql)
        )
    )
```

**3) 원본 Oracle 인스턴스에서 TG4MSQL 인스턴스에 대한 클라이언트 구성 설정**

원본 Oracle 인스턴스에서 TG4MSQL 인스턴스에 대한 tnsname.ora 파일을 다음과 같이 구성합니다.

```
TG4MSQL =
    (DESCRIPTION =
        (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = home1)(PORT = 1521))
         )
        (CONNECT_DATA =
        (SERVICE_NAME = tg4msql)
        )

        (HS = OK) ←이 부분 추가 확인
    )
```

Net Manager를 사용하는 경우에는 서비스 이름으로 [TG4MSQL]을 지정하고 서비스 ID의 고급 버튼을 누른고 이기종 서비스 체크 박스를 선택합니다.


**4) 원본 Oracle 인스턴스에서 TG4MSQL 인스턴스에 대한 데이터베이스 링크 구성**

원본 Oracle 인스턴스에서 TG4MSQL 인스턴스에 대한 다음과 같이 데이터베이스 링크를 생성합니다.

```
CREATE PUBLIC DATABASE LINK [LINK명]
CONNECT TO [계정명] IDENTIFIED BY [패스워드]
USING  '[GATEWAY 서비스명]'
```

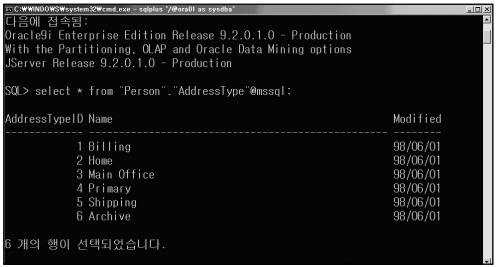
이때 해당 계정은 Oracle 원본 인스턴스와 대상 SQL Server에 모두 존재해야 합니다.

5) TG4MSQL 구성 확인

다음과 같이 TG4MSQL을 통해서 Oracle 원본 인스턴스에서 SQL Server로 쿼리를 수행하여 TG4MSQL이 올바르게 구성되었음을 확인합니다.

```
SELECT * FROM table_name@db_link_name ;
```

다음은 원본 Oracle 인스턴스에서 [mssql]이라는 이름의 데이터베이스 링크를 통해서 대상 SQL Server 2005의 Adventureworks.Person.AddressType 테이블에 테스트 쿼리를 수행한 결과입니다.




## 플랫 파일 데이터 로드하기

Oracle은 플랫 파일에서 데이터를 로드하기 위해서 다음과 SQL*Loader라는 도구를 사용합니다. SQL*Loader 도구는 다음과 같이 컨트롤 파일에 세부 사항을 지정하고 실행 시에 해당 컨트롤 파일의 위치를 지정합니다.

SQL*Loader 실행 구문 :

```
sqlldr userid=SCOTT/TIGER@ora01 control=d:₩flatdata₩sqlldrTEST.ctl
```

```
컨트롤 파일 :

LOAD DATA
infile 'D:₩flatdata₩emp.txt'     → 데이터 플랫 파일 지정
INTO TABLE SCOTT.EMP             → 대상 스키마 및 테이블 지정
Append                           → 데이터 추가 여부 지정
FIELDS TERMINATED BY '#'         → 컬럼 종결자 지정
(
EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO
)
```

이와 같은 기능은 SQL Server에서는 BCP, BULK INSERT, 데이터 가져오기 및 내보내기 마법사, SQL Server integration Services 등 다양한 방법을 통해서 가능하며 요구 사항에 따라서 사용자가 적절한 방법을 선택할 수 있습니다. BCP, 데이터 가져오기 및 내보내기 마법사, SSIS는 데이터의 로드뿐만 아니라 다양한 데이터의 추출 및 변환 기능도 제공합니다.

### ■ BCP

BCP 유틸리티는 SQL Server 인스턴스와 사용자가 지정한 형식의 데이터 파일 간에 데이터를 대량으로 복사하는 도구로 플랫 파일로 데이터를 추출하거나 플랫 파일의 데이터를 SQL Server 인스턴스로 로드할 수 있습니다. 또한, Queryout 옵션을 지정하면 데이터를 쿼리를 통해서 추출하거나 변환하는 기능도 제공합니다.

#### 1) BCP 유틸리티 구문

```
BCP {[[database_name.][owner].]{table_name | view_name} | "query"}
    {in | out | queryout | format} data_file
    [-mmax_errors] [-fformat_file] [-x] [-eerr_file]
    [-Ffirst_row] [-Llast_row] [-bbatch_size]
```

```
[-n] [-c] [-w] [-N] [-V (60 | 65 | 70 | 80)] [-6]
 [-q] [-C { ACP | OEM | RAW | code_page } ] [-tfield_term]
 [-rrow_term] [-iinput_file] [-ooutput_file] [-apacket_size]
 [-Sserver_name[₩instance_name]] [-Ulogin_id] [-Ppassword]
 [-T] [-v] [-R] [-k] [-E] [-h"hint [,...n]"]
```

2 ) BCP 유틸리티 옵션

| 옵션 | 설 명 |
|---|---|
| database_name | 지정한 테이블 또는 뷰가 있는 데이터베이스 이름입니다.<br>지정하지 않으면 사용자의 기본 데이터베이스를 사용합니다. |
| owner | 테이블 또는 뷰의 스키마 이름입니다. |
| table_name I view_name | 데이터를 가져올 때 (in)의 대상 테이블이나 뷰, 내보낼 때의 (out) 원본 테이블이나 뷰 이름입니다. |
| query | 쿼리에서 데이터를 대량 복사할 때 지정하는 T-SQL 쿼리입니다. |
| in I out I queryout I format | 대량 복사 방향을 지정합니다.<br>in : 데이터 로드 / out : 데이터 추출<br>queryout : 쿼리로부터 데이터 추출<br>format : 서식 파일 지정 |
| datafile | 데이터 파일의 전체 경로입니다. |
| -b | 로드하는 데이터의 일괄 처리 당 행수를 지정하며 커밋되는 행수를 나타냅니다 |
| -n | 원시 데이터 형식을 사용하여 작업을 수행합니다. |
| -c | 문자 데이터 형식을 사용하여 작업을 수행합니다. |
| -t | 컬럼 종결자를 지정합니다.<br>기본값은 ₩t (탭문자)입니다. |

| | |
|---|---|
| -S *server_name* [₩*instance_name*] | 연결할 SQL Server 인스턴스를 지정합니다. |
| -U *login_id* [ -P *password*] | 사용자 로그인 ID와 패스워드를 지정합니다. 로그인 ID와 패스워드는 대소문자를 구분합니다. -U와 -P옵션을 지정하지 않으면 -T 옵션을 지정하지 않더라도 BCP 유틸리티를 실행하는 현재 로그온한 Windows 계정으로 Windows 인증 모드를 사용하여 SQL Server에 로그인합니다. |
| -T | 사용자 이름과 암호를 사용하는 대신 트러스트된 연결을 사용하여 SQL Server에 로그인합니다. 기본적으로 BCP 유틸리티는 트러스트된 연결 옵션을 사용합니다. |
| -? | BCP 유틸리티 옵션의 구문 요약 정보를 표시합니다. |

3) 다음은 Adventureworks 데이터베이스의 HumanResources 스키마의 Employee 테이블에서 D:₩Employee.txt 파일로 추출하는 구문입니다. 컬럼 종결자는 쉼표를 사용했으며 Windows 인증을 사용해서 SQL Server에 연결합니다.

```
BCP Adventureworks.HumanResources.Employee  OUT D:\Employee.txt -c -t, -T
```

## ■ BULK INSERT

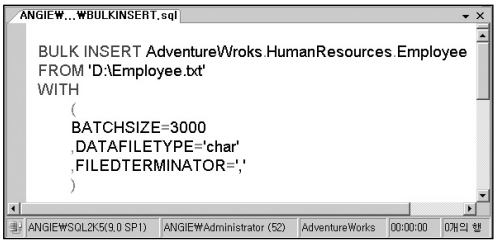
BULK INSERT T-SQL 구문은 데이터 파일로부터 사용자가 지정한 형식으로 테이블이나 뷰로 데이터를 로드합니다.

### 1) BULK INSERT 구문

```
BULK INSERT  [database_name].[schema_name].{table_name | view_name}
FROM  'data_file'
WITH ( [[,] BATCHSIZE = batch_size ]
      [[,] DATAFILETYPE  = { 'char' |' native' |' widechar' |' widenative' }]

     [[,] FILEDTERMINATOR=' filed_terminator' ]
      [[,] FORMATFILE=' format_file_path' ]
     )
```

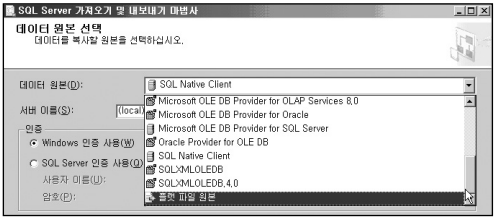
2) 다음은 AdventureWorks 데이터베이스 HumanResources 스키마의 Employee 테이블을 대상으로 컬럼 종결자는 쉼표로 지정되고 문자 데이터 형식으로 구성된 D:₩Employee.txt 데이터 플랫 파일을 3,000 행 마다 커밋하며 데이터를 로드하는 BULK INSERT 구문입니다.



## ■ 데이터 가져오기 및 내보내기 마법사

데이터 가져오기 및 내보내기 마법사는 지원되는 데이터 공급자를 통해서 데이터 원본과 대상간에 데이터를 복사하고 변환하는 기능을 제공합니다.

SQL Server 2005에서 기본으로 제공하는 데이터 공급자는 다음과 같이 SQL Server Native Client, Microsoft OLE DB Provider For Oracle, Microsoft Excel 등 총 17개를 기본으로 제공합니다.

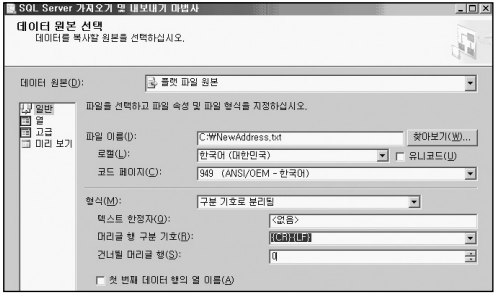



데이터 가져오기 및 내보내기 마법사는 ① 데이터 원본 선택, ② 데이터 대상 선택 ③ 원본 및 대상 테이블 선택과 매핑 편집 등 세 단계를 거쳐서 작업을 수행합니다.

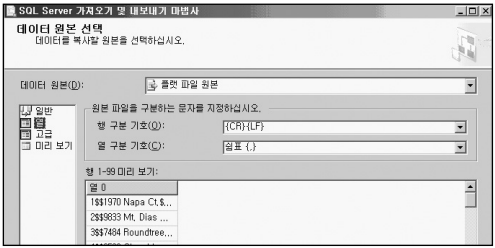
### 1) 데이터 원본 선택

데이터 가져오기 및 내보내기 마법사를 실행하려면 개체 탐색기에서 해당 서버를 확장하고 데이터베이스 선택한 뒤 오른 쪽 버튼을 눌러서 [작업(T)] 메뉴의 [데이터 가져오기(I)] 또는 [데이터 내보내기(X)] 메뉴 중 하나를 선택합니다. 두 메뉴 모두 동일하게 DTSWizard.exe를 실행합니다. [시작] 버튼을 누른 뒤 [실행(R)] 창에서 DTSWizard.exe를 실행해도 데이터 가져오기 및 내보내기 마법사를 실행할 수 있습니다.

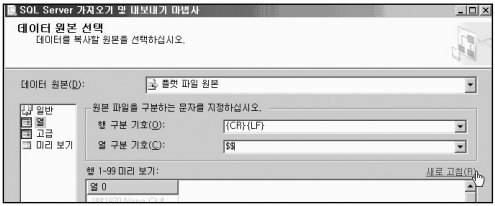

[데이터 가져오기 및 내보내기 마법사 시작] 페이지에서 [다음(N)] 버튼을 눌러서 [데이터 원본 선택] 페이지로 이동합니다.



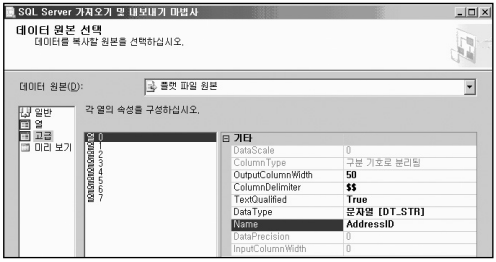
[데이터 원본 선택]의 [일반] 페이지의 [데이터 원본(D):] 부분에서 원본의 공급자를 지정합니다. [파일 이름(I)] 부분에서 원본 플랫 파일을 지정합니다. 이어서 다음과 같이 [열] 페이지를 선택합니다.




[열 구분 기호(C):] 부분에 원본 플랫 파일의 컬럼 종결자 기호를 입력합니다. 예제는 $$로 열이 구분되어 있으므로 다음과 같이 $$를 직접 입력하고 [새로 고침(R)] 링크를 클릭합니다.

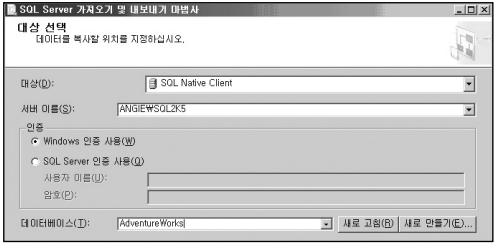


이어서 [고급] 페이지로 이동하여 다음과 같이 각 컬럼 별로 이름을 포함한 속성을 변경할 수 있습니다.

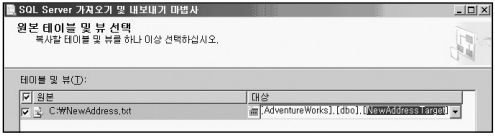



지정이 완료되면 [미리 보기] 탭에서 데이터 원본에 대해서 올바른 설정을 했는지 여부를 확인합니다. [다음(N)] 버튼을 눌러서 [대상 선택] 페이지로 이동합니다.

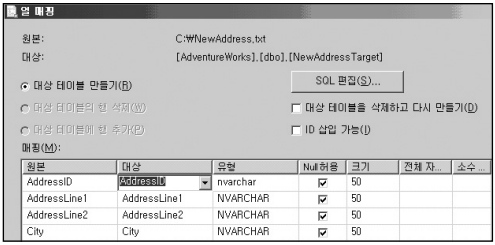
[대상(D):] 부분에는 SQL Server 2005와 연결하기 위한 SQL Native Client를 지정하고,
[서버 이름(S):] 부분에 대상 SQL Server 인스턴스를 지정합니다.
[데이터베이스(T):] 부분에는 플랫 파일 데이터를 로드할 대상 데이터베이스 이름을 지정합니다.



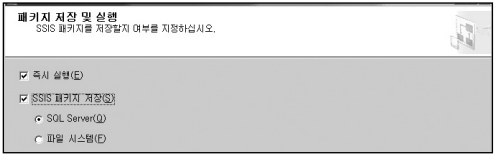
[다음(N)] 버튼을 눌러서 [원본 테이블 및 뷰 선택] 페이지로 이동합니다.
대상 테이블의 이름은 원본 플랫 파일의 이름을 기본으로 설정합니다. 수정하고자 하는 경우에 다음과 같이 테이블 이름을 직접 수정할 수 있습니다.




컬럼의 매핑을 변경하고자 하는 경우는 [원본 테이블 및 뷰 선택] 페이지 하단의 [매핑 편집] 버튼을 눌러서 다음과 같이 [열 매핑] 페이지를 실행하여 편집할 수 있습니다.

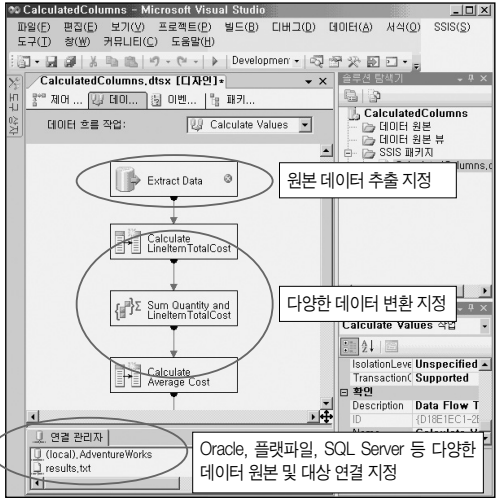



[다음(N)] 버튼을 눌러서 [패키지 저장 및 실행] 페이지로 이동하여 패키지를 즉시 실행하거나 SSIS 패키지를 저장할 수 있습니다.



## ■ SQL Server integration Services(SSIS) 소개

SSIS는 데이터 웨어하우징을 위한 ETL(데이터 추출, 변환 및 로드) 패키지를 비롯하여 고성능 데이터 통합 솔루션을 작성하기 위한 플랫폼입니다. SSIS에는 데이터를 추출, 변환 및 로드하는 패키지를 작성하고 디버깅을 할 수 있는 그래픽 도구와 마법사, FTP 작업, SQL 문 실행 및 전자 메일 보내기와 같은 워크플로 기능 수행을 위한 작업, 데이터 추출 및 로드 시 데이터 정리, 집계, 병합 및 복사를 위한 변환 기능 및 관리 기능, 그리고 개체 모델 프로그래밍을 위한 API(응용 프로그래밍 인터페이스) 등이 포함되어 있습니다.




〈Business Intelligence Studio를 사용한 SSIS 패키지 예〉

***[참고]***
*SSIS 에 관한 자세한 사항은 온라인 설명서나 SQL Server 2005의 설치 옵션에서 제공하는 자습서와 예제를 통해 확인하기 바랍니다.*

# 6. SQL Server와 Oracle 차이점 이해

SQL Server와 Oracle은 구조나 관리, 개발 방법에서 많은 부분이 유사하고 또한, 많은 부분이 서로 상이하여 통합관리나 마이그레이션 작업을 위해서는 각각의 DBMS를 잘 이해하고 있어야 할 뿐만 아니라 상이점도 잘 알고 있어야 합니다.

다음과 같이 SQL Server와 Oracle의 차이점을 살펴봅니다.

① 아키텍처 이해 및 최대 용량 사양
② 스키마 개체
③ T-SQL vs PL/SQL
④ 트랜잭션 처리
⑤ 시스템 함수
⑥ SQLCMD 유틸리티
⑦ 보안관리
⑧ 관리작업 자동화
⑨ 자주사용하는 시스템 저장 프로시저 및 함수

## 아키텍처 이해 및 최대 용량 사양

1) 인스턴스
Oracle의 인스턴스는 메모리에 로드된 데이터베이스와 메모리 공간 그리고, 다양한 백 그라운드 프로세스로 구성됩니다. 따라서, 하나의 데이터베이스가 인스턴스의 기본 단위가 되지만 SQL Server의 인스턴스는 하나의 데이터베이스가 아니라 하나의 SQL Server가 기본 단위입니다. 따라서, 하나의 물리적인 서버 컴퓨터에 여러 개의 SQL Server의 인스턴스를 구성하기 위해서는 구성하고자 하는 인스턴스의 수 만큼 SQL Server를 설치해야 합니다.
그리고 나서 SQL Server 인스턴스는 여러 개의 데이터베이스를 서비스하게 됩니다.
SQL Server 2005는 하나의 컴퓨터에 최대 50개의 인스턴스를 설치할 수 있습니다.

2) 인스턴스 구성 요소

| 구 성 요 소 | SQL Server | Oracle |
|---|---|---|
| 데이터베이스 | 필수 시스템 데이터베이스 및 사용자 데이터베이스 | 사용자 데이터베이스 |
| 버퍼 | 버퍼 풀 및 MemToLeave 등 | SGA, PGA, UGA 등 |
| 백그라운드 프로세스 | Worker Thread, SQLOS, DB Cleanup, DB Shrink, Lazy Writer, CHECKPOINT, LOG Writer 등 | Server Process, PMON, SMON, CHECKPOINT, DBWR, LGWR, ARCHIVE, RECO 등 |
| 인스턴스 구성 옵션 등 | | 제어 파일, 초기 파라미터 파일 |
| 트랜잭션 로그 | 각 데이터베이스 별 트랜잭션 로그 | 온라인 리두 로그 |

3) 시스템 데이터베이스

Oracle은 하나의 데이터베이스가 인스턴스의 단위가 되므로 하나의 데이터베이스에 인스턴스에 필요한 모든 구성 요소를 포함하지만 SQL Server는 하나의 인스턴스가 여러 개의 데이터베이스로 구성됩니다. 필수 구성 요소로서 시스템 데이터베이스를 필요로 하며 필수 시스템 데이터베이스는 master, model, msdb, tempdb, resource 데이터베이스 입니다. 각 시스템 데이터베이스의 기능은 다음과 같습니다.

| 데이터베이스 | 설 명 |
|---|---|
| master | SQL Server 인스턴스에 대한 모든 시스템 수준의 정보를 저장합니다. Oracle의 SYSTEM 테이블스페이스와 유사합니다. |
| tempdb | 임시 작업의 저장소로 임시 개체나 중간 결과 집합을 저장하기 위한 작업 영역으로 사용됩니다. Oracle의 TEMP 테이블스페이스와 유사합니다.<br>뿐만 아니라 SQL Server 2005에서는 SNAPSHOT 트랜잭션 고립화 수준을 사용하는 경우에 최근 커밋된 데이터의 이미지를 저장하는 Oracle의 롤백 세그먼트(UNDO 테이블스페이스) 기능을 수행하기도 합니다. |
| model | SQL Server에서 생성되는 모든 데이터베이스에 대한 템플릿 기능을 수행합니다. model 데이터베이스에 설정한 사용자 정보, 파일의 크기, 정렬, 복구 모델, 데이터베이스 옵션의 내용이 이후에 생성되는 모든 데이터베이스에 기본적으로 적용됩니다. |
| msdb | 계획된 작업, 경고, 일정 및 복제 정보와 같은 SQL Server 에이전트 서비스의 정보를 저장합니다. Oracle의 SYSTEM 테이블스페이스 또는 SYSAUX 테이블스페이스와 유사합니다. |
| 리소스 데이터베이스 | SQL Server 2005에 포함된 시스템 개체가 들어 있는 읽기 전용 데이터베이스 입니다. 시스템 개체는 리소스 데이터베이스에 저장되고 논리적으로는 각 데이터베이스의 sys 스키마 개체로 나타납니다. |

#### 4) SQL Server 데이터베이스

SQL Server는 시스템 데이터베이스를 포함한 모든 데이터베이스가 각각 독립적으로 데이터 파일과 트랜잭션 로그 파일을 가지게 됩니다. 다음은 SQL Server와 Oracle의 데이터베이스의 저장소 구조입니다.

| 구분 | SQL Server | Oracle |
|---|---|---|
| 데이터베이스 | 데이터베이스 | 데이터베이스 |
| 개체의 저장 대상 | 파일 그룹 | 테이블스페이스 |
| 파일 그룹내의 물리적인 파일 | 데이터 파일 | 데이터 파일 |
| 저장소를 갖는 스키마 개체 | 테이블/인덱스 | 세그먼트 |
| 개체의 저장소 할당 단위 | 익스텐트 | 익스텐트 |
| IO 단위 | 페이지 | 블록 |
| 트랜잭션 로그를 기록하는 물리적인 파일 | 트랜잭션 로그 파일 | 온라인 리두 로그 파일 |

#### 5) 최대 용량 사양

SQL Server와 Oracle의 최대 용량 사양은 다음과 같습니다.

| 특 징 | SQL Server 2005 | Oracle |
|---|---|---|
| 서버 당 인스턴스 개수 | 50 | 리소스 제한까지 |
| 최대 데이터베이스 크기 | 512PB | 4PB<br>(65,536 * 64 GB) |
| 최대 파일 개수 | 32,767 | 65,536 |
| 최대 데이터 파일 크기 | 16 TB | 64 GB (128TB) |
| 최대 로그 파일 크기 | 2 TB | 64 GB |

| | | |
|---|---|---|
| 최대 테이블스페이스 개수 | 32,767 (파일 그룹) | 65,536 |
| 최대 컨트롤 파일 크기 | 관련 없음 | 20,000 블록 |
| 최대 익스텐트 크기 | 64 KB (고정) | 4 GB |
| 블록(페이지) 크기 | 8 KB (고정) | 2 KB~32 KB |
| 파일 당 최대 블록 개수 | 20억 개 | 4 백만 개/40억개<br>(BigFile 테이블스페이스) |

## 스키마 개체

SQL Server와 Oracle의 스키마 개체는 다음과 같습니다.

| SQL Server | Oracle |
|---|---|
| 테이블 | 테이블 |
| 인덱스 | 인덱스 |
| 뷰 | 뷰 |
| 동의어 (synonym) | 동의어 |
| 없음 (유사 기능 : Identity 속성) | 시퀀스 |
| 저장 프로시저 | 프로시저 |
| 함수 | 함수 |
| 없음 | 패키지 |
| 서비스 브로커(Service Broker) | Advanced Queuing |
| CLR 타입 | 개체 타입 |
| XML 스키마 컬렉션 | XML 데이터베이스 |

### ■ 개체 식별자 및 이름 지정

Oracle은 개체와 컬럼의 이름은 기본적으로 대문자로 저장됩니다. 그렇지만, "Case Sense"와 같이 이중 인용 부호(" ")를 사용하면 대소문자를 구별해서 저장할 수 있습니다.
또한 개체나 컬럼의 이름에 빈 공간 등을 표시하고자 할 때도 이중 인용 부호를 사용합니다.

SQL Server는 개체와 컬럼의 이름을 저장할 때 생성 시에 지정한 대로 저장합니다. 단, 대소문자를 구별할 지 여부는 데이터베이스의 정렬 지정에 따라서 달라집니다. 데이터베이스 정렬이 대소문자를 구별하는 경우에는 생성 시에 지정한 이름 그대로를 사용하고 정렬이 대소문자를 구별하지 않는 경우에는 생성 시 지정과 상관없이 구별하지 않습니다. 또한 빈 공간 같은 것을 이름에 사용하고자 할 때는 이중 인용 부호(" ")나 대괄호([ ])를 사용합니다.

Oracle과 SQL Server 모두 개체의 이름은 동일한 스키마 내에서는 유일해야 합니다.
SQL Server의 개체의 이름은 [server_name].[database_name].[schema_name].[object_name] 과 같이 4부분으로 구별됩니다.

Oracle은 데이터베이스 이름 8바이트, 개체이름은 30바이트, 데이터베이스 링크 이름은 128바이트이며 SQL Server는 임시 테이블만 유니코드로 116 자이고, 나머지 개체는 모두 유니코드 128자까지 지정 가능하며 첫 번째 글자는 Oracle 및 SQL Server 모두 숫자를 사용할 수 없습니다.

### ■ 테이블

① 테이블 생성 구문

| 테이블 유형 | SQL Server | Oracle |
|---|---|---|
| 테이블 | CREATE TABLE [schema].table_name<br>( column_name data type<br>column constraints,<br>…<br>table_constraints )<br>)<br>[ON filegroup / partition_scheme] | CREATE TABLE [schema]. table_name<br>( column_name data type<br>column_constraints,<br>…<br>table_constraints )<br>)<br>[TABLESPACE tablespace] |

| | | |
|---|---|---|
| 임시<br>테이블 | 로컬 임시 테이블:<br>CREATE TABLE #table_name<br>전역 임시 테이블:<br>CREATE TABLE ##table_name | CREATE GLOBAL TEMPORARY<br>TABLE table_definition<br>ON COMMIT DELETE I PRESERVE<br>ROWS |

② 테이블 수정 구문

| 작업<br>내용 | SQL Server | Oracle |
|---|---|---|
| 컬럼<br>추가 | ALTER TABLE table_name<br>ADD column_name column_property | ALTER TABLE table_name<br>ADD (column_name column_property) |
| 컬럼<br>수정 | ALTER TABLE table_name<br>ALTER COLUMN column_name<br>new_column_property | ALTER TABLE table_name<br>MODIFY (column_name<br>new_column_property ) |
| 컬럼<br>이름<br>변경 | EXEC sp_rename<br>Table_name,Old_column_name<br>New_column_name | ALTER TABLE table_name<br>RENAME COLUMN<br>Old_column_name New_column_name |
| 컬럼<br>삭제 | ALTER TABLE table_name<br>DROP column_name | ALTER TABLE table_name<br>DROP (COLUMN column_name) |
| 컬럼<br>주석 | EXEC sp_addextendedproperty<br>@name=N' property_name' ,<br>@value=' description' … | COMMENT ON COLUMN<br>Table_name,column_name IS<br>comment |

■ **제약 조건**

Oracle과 SQL Server는 다음과 같은 제약 조건을 제공합니다.

■ DEFAULT – 기본값을 지정합니다. (Oracle은 컬럼 속성으로 취급합니다.)
■ NOT NULL – 반드시 값을 가져야 합니다. (NULL을 허용하지 않습니다.)
■ CHECK – 주어진 조건에 만족하는 값만 허용합니다.
■ UNIQUE – 중복 값을 허용하지 않습니다.
■ PRIMARY KEY – 테이블의 행을 구별하는 기준 역할을 합니다.
■ FOREIGN KEY – 서로 다른 테이블간에 부모-자식 관계를 설정합니다.

**1) 제약 조건 생성 구문**

■ DEFAULT

| | SQL Server | Oracle |
|---|---|---|
| 컬럼 선언 시 | create table table_name<br>(id int<br>,phone varchar(30)<br>DEFAULT '02)3468-0600' ) | create table table_name<br>(id number(4)<br>,phone varchar2(30)<br>DEFAULT '02)3468-0600' ) |
| 제약 조건 추가 | alter table table_name<br>add constraint constraint_name<br>DEFAULT '02)3468-0600'<br>for column_name<br>[WITH VALUES] | alter table table_name<br>modify phone varchar2(30)<br>DEFAULT '02)3468-0600' |

기본 값은 테이블 수준 제약 조건으로 선언할 수 없습니다.

■ NOT NULL

| | SQL Server | Oracle |
|---|---|---|
| 컬럼 선언 시 | create table table_name<br>(id int NOT NULL<br>,phone varchar(30)<br>) | create table table_name<br>(id number(4)<br>constraint n NOT NULL<br>,phone varchar2(30) |
| 컬럼 수정 시 | alter table table_name<br>alter column id int NOT NULL | alter table table_name<br>modify id number(4) NOT NULL |

NOT NULL은 테이블 수준 제약 조건으로 선언할 수 없습니다.

■ CHECK

| | SQL Server | Oracle |
|---|---|---|
| 컬럼 선언 시 | create table table_name<br>(id int<br>,gender char(1) constraint ck<br>CHECK IN ( 'M' , 'F' )<br>) | create table table_name<br>(id number(4)<br>,gender char(1) constraint ck<br>CHECK IN ( 'M' , 'F' )<br>) |
| 제약 조건 추가 | alter table table_name<br>add constraint ck CHECK<br>(gender in ( 'M' , 'F' )) | alter table table_name<br>add constraint ck CHECK<br>(gender in ( 'M' , 'F' )) |

■ UNIQUE

| | SQL Server | Oracle |
|---|---|---|
| 컬럼 선언 시 | create table table_name<br>(id int<br>,student_id varchar(30)<br>UNIQUE NONCLUSTERED<br>ON filegroup_name<br>) | create table table_name<br>(id number(4)<br>,student_id varchar2(30)<br>UNIQUE<br>USING INDEX index_name<br>TABLESPACE tablespace_name) |
| 컬럼 수정 시 | alter table table_name<br>add constraint constraint_name<br>UNIQUE NOCLUSTERED<br>(student_id)<br>ON filegroup_name | alter table table_name<br>add constraint constraint_name<br>UNIQUE (student_id)<br>USING INDEX index_name<br>TABLESPACE tablespace_name |

Oracle은 USING 구문을 사용해서 인덱스의 이름과 저장소를 지정할 수 있으며 여러 개의 NULL 값을 허용합니다. SQL Server는 인덱스 생성 옵션인 CLUSTERED 또는 NOCLUSTERED를 지정하지 않는 경우에 비클러스터형(NONCLUSTERED) 인덱스를 생성하며 저장소를 지정하지 않을 경우 기본 파일 그룹에 저장돠며 단 하나의 NULL 값만을 허용합니다.

■ PRIMARY KEY

| | SQL Server | Oracle |
|---|---|---|
| 컬럼 선언 시 | create table table_name<br>(id int PRIMARY KEY<br>NOCLUSTERED<br>,student_id varchar(30)<br>) | create table table_name<br>(id number(4)<br>PRIMARY KEY<br>USING INDEX<br>TABLESPACE tablespace_name<br>,student_id varchar2(30)<br><br>) |
| 컬럼 수정 시 | alter table table_name<br>add constraint constraint_name<br>PRIMARY KEY (id)<br>ON filegroup_name | alter table table_name<br>add ( PRIMARY KEY (id)) |

■ FOREIGN KEY

| | SQL Server | Oracle |
|---|---|---|
| 컬럼 선언 시 | create table table_name<br>(id int<br>,student_id varchar(30)<br>, dept_no varchar(6) FOREIGN KEY<br>REFERENCES dept(dept_no)<br>) | create table table_name<br>(id int<br>,student_id varchar(30)<br>, dept_no constraint<br>constraint_name<br>REFERENCES dept(dept_no)<br>) |
| 컬럼 수정 시 | alter table table_name<br>add constraint constraint_name<br>FOREIGN KEY (dept_no)<br>REFERENCES dept(dept_no) | alter table table_name<br>add constraint constraint_name<br>FOREIGN KEY (dept_no)<br>REFERENCES dept(dept_no) |

Oracle의 기본 키, 유일 제약 조건, 참조 키는 최대 32개 컬럼을 지정할 수 있으며 SQL Server는 16개 컬럼을 지정할 수 있습니다. 참조 키 정의 시 부모 테이블의 변경에 따라서 다음과 같이 자식 테이블의 데이터를 변경하는 수행될 작업을 지정할 수 있으며 Oracle은 부모 테이블의 수정에 대한 옵션을 지원하지 않습니다.

| 변경 사항 | 수행 작업 | SQL Server | Oracle | 구문 예제 |
|---|---|---|---|---|
| ON DELETE | NO ACTION | ○ | ○ | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON DELETE NO ACTION |
| ON DELETE | CASCADE | ○ | ○ | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON DELETE CASCADE |
| ON DELETE | SET NULL | ○ | ○ | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON DELETE SET NULL<br>* 이 경우에 참조 키 컬럼은 반드시 NULL을 허용 해야 함 |

| | | | | |
|---|---|---|---|---|
| ON DELETE | SET DEFAULT | ○ | X | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON DELETE SET DEFAULT<br>* 이 경우 참조 키 컬럼은 기본값을 가지고 있거나 NULL을 허용 해야 함 |
| ON UPDATE | NO ACTION | ○ | X | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON UPDATE NO ACTION |
| ON UPDATE | CASCADE | ○ | X | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON UPDATE CASCADE |

| | | | | |
|---|---|---|---|---|
| ON UPDATE | SET NULL | ○ | X | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>ON UPDATE SET NULL<br>* 이 경우에 참조 키 컬럼은 반드시 NULL을 허용 해야 함 |
| ON UPDATE | SET DEFAULT | ○ | X | CREATE TABLE emp<br>(…,<br>dept_no VARCHAR2(6),<br>…,<br>CONSTRAINT emp_dept_fk FOREIGN KEY<br>REFERENCES dept(dept_no)<br>* 이 경우 참조 키 컬럼은 기본값을 가지고 있거나 NULL을 허용 해야 함 |

② 제약 조건 관리

Oracle과 SQL Server는 다음과 같이 제약 조건을 관리할 수 있습니다.

| 작업 | SQL Server | Oracle |
|---|---|---|
| 활성화/<br>비활성화 | ALTER TABLE table_name<br>WITH { CHECK \| NOCHECK } ]<br>ADD CONSTRAINT constraint_name<br>ALTER TABLE table_name<br>{ CHECK \| NOCHECK } CONSTRAINT<br>{ ALL \| constraint_name [ ,...n ] }<br><br>ALTER INDEX IX_Employee_ManagerID ON<br>HumanResources.Employee<br>DISABLE<br><br>ALTER INDEX ALL ON Production.Product<br>REBUILD WITH (FILLFACTOR = 80,<br>SORT_IN_TEMPDB=ON,<br>STATISTICS_NORECOMPUTE = ON);<br>ALTER TABLE table_name<br>{ENABLE \| DISABLE} | [VALIDATE \| NOVALIDATE]<br>{PRIMARY KEY \| UNIQUE \|<br>CONSTRAINT constraint_name}<br><br>또는<br><br>ALTER TABLE table_name<br>MODIFY<br>CONSTRAINT constraint_name<br>{ENABLE \| DISBALE} |
| 제약<br>조건<br>추가 | ALTER TABLE table_name<br>WITH {CHECK \| NOCHEK}<br>ADD CONSTRAINT constraint_name<br>{ PRIMARY KEY \| UNIQUE \|<br>FOREIGN KEY \| CHECK \| DEFAULT} | ALTER TABLE table_name<br>ADD CONSTRAINT<br>constraint_name<br>{ PRIMARY KEY \| UNIQUE \|<br>FOREIGN KEY \| CHECK \|<br>CONSTRAINT}<br>{VALIDATE \| NOVALIDATE} |

| | | |
|---|---|---|
| 이름<br>변경 | sp_rename<br>[ @objname = ] 'object_name'<br>, [@newname = ] 'new_name'<br>[,[@objtype =] 'object_type' ]<br>--' object_type' 은 'constraint' 로 지정 | ALTER TABLE … RENAME<br>old_constraint_name TO<br>new_constriant_name |
| 제약<br>조건<br>삭제 | ALTER TABLE table_name<br>DROP CONSTRAINT<br>constraint_name WITH (MAXDOP =<br>max_degree_of_parallelism<br>\| ONLINE = {ON \| OFF } \| MOVE TO {<br>partition_scheme_name ( column_name ) \|<br>filegroup \| "default" } [ ,...n ] ) | ALTER TABLE table_name<br>DROP { PRIMARY KEY \| UNIQUE \|<br>FOREIGN KEY \| CHECK \|<br>CONSTRAINT} constraint_name} |

SQL Server 2005에서는 인덱스를 비활성화 해서 기본 키와 유일 제약 조건을 비활성화 할 수 있습니다.
클러스터형 인덱스를 비활성화 하는 경우에는 실제 데이터에 대한 액세스가 불가능합니다.

**■ 인덱스**

Oracle과 SQL Server는 다음과 같은 인덱스를 제공합니다.

| 인덱스 종류 | SQL Server | Oracle |
|---|---|---|
| B-Tree 인덱스 | ○ | ○ |
| UNIQUE 인덱스 | ○ | ○ |
| 복합 인덱스 | ○ | ○ |
| 오름 차순 인덱스 | ○ | ○ |
| 내림 차순 인덱스 | ○ | ○ |
| 클러스터형 인덱스 | ○ | Index-Organized Table |

| 함수 기반 인덱스 | 계산된 컬럼 인덱스 | ○ |
|---|---|---|
| 분할된 (Partition) 인덱스 | ○ | ○ |
| Reverse Key / 비트맵 인덱스 | X | ○ |

SQL Server의 클러스터형 인덱스는 Oracle의 Index-Orgarnized Table(IOT)과 유사하지만 IOT가 Primary Key컬럼에만 생성할 수 있는 것에 반해 SQL Server의 클러스터형 인덱스는 모든 컬럼에 지정 가능합니다. 따라서, 범위 검색, 집계, 값이 증가하는 컬럼, 데이터를 정렬해서 저장해야 하는 경우 등에 지정하면 성능 향상의 효과를 가져옵니다. Oracle은 인덱스의 컬럼을 32개 까지 지정할 수 있고 SQL Server는 16개 컬럼 을 지정할 수 있지만 INCLUDE 절을 사용해서 최대 1,024개 컬럼까지 지정 가능합니다. INCLUDE 절에 지정된 컬럼은 검색을 제한하는 용도가 아니라 인덱스 검색 후 데이터를 검색하는 과정을 생략하기 위한 목적으로 사용합니다.

① 인덱스 생성 구문

| | SQL Server | Oracle |
|---|---|---|
| 인덱스 생성 | CREATE [UNIQUE]<br>{NONCLUSTERED \| CLUSTERED<br>INDEX index_name<br>ON 〈table_name or view_name〉<br>(column_name [ASC\|DESC],*n*…,) | CREATE [UNIQUE]<br>INDEX index_name<br>ON table_name<br>(column_name [ASC\|DESC],,…*n*)<br>추가 컬럼 지정<br>[INCLUDE (column_name,…*n*) |
| 추가 컬럼 지정 | [INCLUDE (column_name,…*n*) | |

| | | |
|---|---|---|
| 인덱스 옵션 | [WITH (<br>PAD_INDEX = { ON \| OFF }<br>\| FILLFACTOR = *fillfactor*<br>\| SORT_IN_TEMPDB = { ON \| OFF }<br>\| IGNORE_DUP_KEY = { ON \| OFF }<br>\| STATISTICS_NORECOMPUTE = { ON \| OFF }<br>\| DROP_EXISTING = { ON \| OFF }<br>\| ONLINE = { ON \| OFF }<br>\| ALLOW_ROW_LOCKS = { ON \| OFF }<br>\| ALLOW_PAGE_LOCKS = { ON \| OFF }<br>\| MAXDOP = *n* | COMPUTE STATISTICS<br>COMPRESS \| NOCOMPRESS<br>MONITORING<br>ONLINE …*n*<br>LOGGING \|NOLOGGING |
| 저장소 지정 | [ON partition_scheme<br>\|filegroup_name] | PARTITION \| TABLESPACE STORAGE 절 |

SQL Server 인덱스의 생성 또는 재 구성 시 지정하는 FILLFACTOR 및 PAD_INDEX 옵션은 Oracle의 PCTFREE 변수와 유사한 동작을 합니다. 인덱스에 행이 지속적으로 입력되면 새 데이터를 수용하기 위해서 인덱스 페이지는 분할하게 되며 이와 같은 분할이 발생하는 것을 억제하기 위해서 채우기 비율을 지정하는 데 FILLFACTOR와 PAD_INDEX 옵션이 사용됩니다. FILLFACTOR 옵션은 인덱스의 리프 수준의 페이지에 대한 채우기 비율을 지정하며 PAD_INDEX 옵션은 FILLFACTOR 옵션에 지정한 채우기 비율을 중간 수준 페이지에도 적용합니다. 단, PAD_INDEX 옵션은 FILLFACTOR 옵션을 지정한 경우에만 사용할 수 있습니다. 인덱스 생성 시 또는 재 작성 시 옵션을 다음과 같이WITH (PAD_INDEX=ON , FILLFACTOR =90 ) 으로 지정한 경우에 Oracle에서 PCTFREE를 10으로 지정한 것과 같습니다.

인덱스 페이지의 분할이 많이 발생하게 되면 범위 검색 시 성능상의 문제를 유발하게 됩니다. 따라서, 다음에 설명하는 인덱스 유지 관리 구문으로 인덱스를 다시 구성하거나 다시 작성해야 합니다.

- 인덱스 재 구성: 리프 노드의 논리적 순서와 물리적인 순서를 일치하도록 리프 수준의 페이지를 기존에 할당된 페이지를 사용해서 다시 정렬합니다. 항상 온라인으로 작업을 수행하며 기존의 채우기 비율을 사용합니다.
- 인덱스 재 작성: 기존 인덱스를 삭제하고 페이지를 압축하여 다시 정렬합니다. 기존의 채우기 비율이나 새로운 채우기 비율을 지정할 수 있으며 새로운 페이지를 할당하여 최적의 압축을 수행합니다.

② 인덱스 유지 관리 구문

| | SQL Server | Oracle |
|---|---|---|
| 상태 변경 | ALTER INDEX { index_name \| ALL }<br>ON 〈object〉 REBUILD \| DISABLE | ALTER INDEX index_name<br>ENABLE\|DISABLE |
| 이름 변경 | EXEC sp_rename<br>'table_name,old_index_name'<br>, 'new_index_name' | ALTER INDEX old_index_name<br>RENAME TO new_index_name |
| 이동 | CREATE INDEX index_name<br>WITH (DROP_EXISTING=ON)<br>ON new_filegroup_name | ALTER INDEX index_name<br>REBUILD new_tablespace_name<br>Storage_attribute<br>[ONLIINE]<br>[LOGGING\|NOLOGGING] |
| 재 구성 | ALTER INDEX { index_name \| ALL }<br>ON 〈object〉 REORGANIZE<br>[ PARTITION = partition_number ]<br>[ WITH ( LOB_COMPACTION = { ON \| OFF})]<br>\| SET ( 〈set_index_option〉 [ ,...n ] ) }<br>또는<br>DBCC INDEXDEFRAG (database_name,<br>table_name,index_name<br>,partition_number) | ALTER INDEX index_name<br>COALESCE |
| 재 작성 | ALTER INDEX { index_name \| ALL }<br>ON 〈object〉 REBUILD<br>[ PARTITION = partition_number ]<br>[ WITH ( LOB_COMPACTION = { ON \| OFF})]<br>\| SET ( 〈set_index_option〉 [ ,...n ] )<br>또는<br>DBCC DBREINDEX<br>(table_name, index_name, fill_factor) | ALTER INDEX index_name<br>REBUILD<br>[ONLIINE]<br>[LOGGING\|NOLOGGING] |

| | | |
|---|---|---|
| 삭제 | DROP INDEX index_name<br>ON 〈object〉<br>[ WITH ( 〈drop_clustered_index_option〉<br>[ ,...n ] ) ] | DROP INDEX index_name |
| 통계<br>갱신 | UPDATE STATISTICS table \| view<br>[{index \| statistics_name}]<br>[ WITH [FULLSCAN]<br>\| SAMPLE number [PERCENT\| ROWS]<br>\| RESAMPLE]<br>[[,][ALL \| COLUMNS \| INDEX]<br>[[,] NORECOMPUTE]] | ALTER INDEX … UPDATE<br>STATISTICS table \| view<br>ESTIMATE\|COMPUTE<br>또는<br>ANALYZE INDEX<br>또는<br>DBMS_STATS 패키지 |

### ■ 트리거

Oracle과 SQL Server는 다음과 같은 트리거를 제공합니다.

| 트리거 종류 | SQL Server | Oracle |
|---|---|---|
| DML 트리거 | ○ | ○ |
| DDL 트리거 | ○ | ○ |
| BEFORE | INSTEAD OF | ○ |
| AFTER | ○ | ○ |
| INSTEAD OF | 테이블 / 뷰 | 뷰 |
| 행 수준 트리거 | X | ○ |
| 문장 수준 트리거 | ○ | ○ |
| UPDATE, DELETE<br>트리거의이전 이미지 참조 | DELETED 테이블 | :OLD |

| | | |
|---|---|---|
| UPDATE, INSERT<br>트리거의 새 이미지 참조 | INSERTED 테이블 | :NEW |
| 트리거 비 활성화 | ALTER TABLE /<br>ENABLE/ DISABLE TRIGGER | ALTER TABLE /<br>ALTER TRIGGER |

SQL Server는 BEFORE 트리거를 별도로 제공하지 않지만 INSTEAD OF 트리거를 일반 테이블에도 생성할 수 있어서 Oracle에서 BEFORE 트리거와 동일한 기능을 수행할 수 있습니다.

① DML 트리거 생성 구문

| 부 분 | SQL Server | Oracle |
|---|---|---|
| 이름 지정 | CREATE TRIGGER trigger_name | CREATE TRIGGER trigger_name |
| 대상 개체 | ON table_name \|view_name | |
| 수행 시점 | FOR \| INSTEAD OF \| AFTER | BEFORE \| AFTER \| INSTEAD OF |
| 동 작 | [INSERT] [,] [UPDATE] [,] [DELETE]<br>* 특정 컬럼 수정 시<br>* UPDATE ( ) \| COLUMNS_UPDATED( ) | INSERT \| UPDATE {OF<br>column_name} \| DELETE |
| 대상 개체 | | ON table_name \| view_name \|<br>Database \| Schema |
| 참조 | [DELETED,column_name]<br>[INSERTED,column_name] | REFERENCING<br>[OLD AS :old] [NEW AS :new] |
| 본문 | AS { SQL_STATEMENT [ ; ] [ ...n ] \|<br>* CLR 통합:<br>EXTERNAL NAME 〈method specifier 〉; } | AS { SQL_STATEMENT } |

SQL Server는 DML 트리거는 트리거 동작 시 트리거가 정의된 테이블과 동일한 구조의 DELETED 및 INSERTED 테이블을 생성합니다. INSERT 트리거에서는 INSERTED 테이블이 생성되고, DELETE 트리거에서는 DELETED 테이블이 생성되며 UPDATE 트리거에서는 INSERTED 및 DELETED 테이블이 모두 생성됩니다. 따라서, INSERT, UPDATE, DELETE 문장이 수행될 때 해당 문에 영향을 받은 행이 해당 테이블에 추가됩니다. SQL Server가 행 수준 트리거가 없다고 하더라도 이와 같은 DELETED 및 INSERTED 테이블을 조작하여 동일한 처리를 수행할 수 있습니다.

② DDL 트리거 생성 구문

| 부 분 | SQL Server | Oracle |
|---|---|---|
| 이름 지정 | CREATE TRIGGER trigger_name | CREATE TRIGGER trigger_name |
| 대상 개체 | ON DATABASE\|ALL SERVER | |
| 수행 시점 | [ EXECUTE AS Clause ]<br>{ FOR \| AFTER } | BEFORE \| AFTER \| INSTEAD OF |
| 동 작 | { event_type \| event_group } [,...n ] | {ALTER\|CREATE<br>\|DROP\|DDL\| LOGON<br>\|LOGOFF\|STARTUP\|<br>SHUTDOWN\| SERVERERROR} |
| 대상 개체 | | ON<br>{DATABASE\|Schema_name.SCHEMA} |
| 본문 | AS { SQL_STATEMENT [ ; ] [ ...n ] \|<br>* CLR 통합:<br>EXTERNAL NAME <method specifier > ; } | AS { SQL_STATEMENT } |

SQL Server 2005 DDL 트리거는 다양한 DDL 이벤트를 지원합니다. 다음은 SQL Server 2005 DDL 트리거의 이벤트입니다.

데이터베이스 수준의 DDL 이벤트

| 이벤트 대상 | 이벤트 |
|---|---|
| APPLICATION_ROLE | CREATE_ / ALTER_ / DROP_ |
| ASSEMBLY | CREATE_ / ALTER_ / DROP_ |
| AUTHORIZATION_DATABASE | ALTER_ |
| CERTIFICATE | CREATE_ / ALTER_ / DROP_ |
| CONTRACT | CREATE_ / ALTER_ |
| DATABASE | GRANT_ / DENY_ / REVOKE_ |
| EVENT_NOTIFICATION | CREATE_ / DROP_ |
| FUNCTION | CREATE_ / ALTER_ / DROP_ |
| INDEX | CREATE_ / ALTER_ / DROP_ |
| MESSAGE_TYPE | CREATE_ / ALTER_ / DROP_ |
| PARTITION_FUNCTION | CREATE_ / ALTER_ / DROP_ |
| PARTITION_SCHEME | CREATE_ / ALTER_ / DROP_ |
| PROCEDURE | CREATE_ / ALTER_ / DROP_ |
| QUEUE | CREATE_ / ALTER_ / DROP_ |
| REMOTE_SERVICE_BINDING | CREATE_ / ALTER_ / DROP_ |
| ROLE | CREATE_ / ALTER_ / DROP_ |
| ROUTE | CREATE_ / ALTER_ / DROP_ |

| | |
|---|---|
| SCHEMA | CREATE_ / ALTER_ / DROP_ |
| SERVICE | CREATE_ / ALTER_ / DROP_ |
| STATISTICS | CREATE_ / ALTER_ / DROP_ |
| SYNONYM | CREATE_ / DROP_ |
| TABLE | CREATE_ / ALTER_ / DROP_ |
| TRIGGER | CREATE_ / ALTER_ / DROP_ |
| TYPE | CREATE_ / DROP_ |
| USER | CREATE_ / ALTER_ / DROP_ |
| VIEW | CREATE_ / ALTER_ / DROP_ |
| XML_SCHEMA_COLLECTION | CREATE_ / ALTER_ / DROP_ |

서버 수준의 DDL 이벤트

| 이벤트 대상 | 이벤트 |
|---|---|
| AUTHORIZATION_SERVER | ALTER_ |
| DATABASE | CREATE_ / ALTER_ / DROP_ |
| ENDPOINT | CREATE_ / DROP_ |
| LOGIN | CREATE_ / ALTER_ / DROP_ |
| SERVER | GRANT_ / DENY_ / REVOKE_ |

SQL Server 2005 DDL 트리거는 트리거를 실행하는 이벤트에 대한 정보를 EVENTDATA( ) 함수를 사용해서 캡처합니다. 이 함수는 XML 값을 반환하며 XML 스키마에는 각 이벤트에 따라서 다양한 정보를 포함합니다. 각 이벤트 별로 반환 정보는 ALTER_TABLE과 같은 해당 이벤트로 온라인 설명서를 통해서 확인할 수 있습니다.
다음은 ALTER_TABLE 이벤트 발생 시 반환하는 XML 스키마입니다.

```
〈EVENT_INSTANCE〉
  〈EventType〉이벤트 유형〈/EventType〉
  〈PostTime〉이벤트 발생 시간〈/PostTime〉
  〈SPID〉서버 프로세스 식별자〈/SPID〉
  〈ServerName〉서버 이름〈/ServerName〉
  〈LoginName〉로그인 이름〈/LoginName〉
  〈UserName〉데이터베이스 사용자 이름〈/UserName〉
  〈DatabaseName〉데이터베이스 이름〈/DatabaseName〉
  〈SchemaName〉스키마 이름〈/SchemaName〉
  〈ObjectName〉개체 이름〈/ObjectName〉
  〈ObjectType〉개체 유형〈/ObjectType〉
  〈TSQLCommand〉수행된 T-SQL 구문〈/TSQLCommand〉
〈/EVENT_INSTANCE〉
```

## ■ 데이터 형식

SQL Server와 Oracle의 데이터 형식은 다음과 같습니다.

| 형 식 | SQL Server | | Oracle | |
|---|---|---|---|---|
| | 데이터 형식 | 크기/범위 | 데이터 형식 | 크기/범위 |
| 글자 | Char | 1~8000 바이트 | Char | 1~2000 바이트 |
| | Nchar | 1~4000 자 | Nchar | 1~2000 바이트 |
| | varchar | 1~8000 바이트 | varchar | 1~4000 바이트 |
| | Nvarchar | 1~4000 자 | Nvarchar | 1~4000 바이트 |
| 숫자 | BigInt | $-2^{63}$~$2^{63}$-1 | Number(19,0) | $10^{19}$-1 ~$10^{19}$-1 |
| | Int | $-2^{31}$~$2^{31}$-1 | Int / Number(10,0) | $-10^{10}$-1 ~$10^{10}$-1 |
| | SmallInt | $-2^{15}$~$2^{15}$-1 | SmallInt / Number(6,0) | $-10^{6}$-1 ~$10^{6}$-1 |
| | TinyInt | 0 ~ 255 | Number(3,0) | $-10^{3}$-1 ~$10^{3}$-1 |
| | Decimal | $-10^{38}$+1~ $10^{38}$-1 | Number(p,s) | $-1\times10^{-130}$ ~ $9.9..9\times10^{125}$ |
| | Numeric(p,s) | $-10^{38}$+1~ $10^{38}$-1 | Number(p,s) | $-1\times10^{-130}$ ~ $9.9..9\times10^{125}$ |
| | Float(53) | - 1.79E+308~ -2.23E-308, 0, 2.23E-308~ 1.79E+308 | Float(53) | - 1.79E+308~ -2.23E-308, 0, 2.23E-308~ 1.79E+308 |

| | | | | |
|---|---|---|---|---|
| 숫자 | Real / | Float(24)<br>- 3.40E+38~<br>-1.18E - 38, 0,<br>1.18E-38~<br>3.40E + 38 | Real /<br>Float(24) | - 3.40E+38~<br>-1.18E - 38, 0,<br>1.18E-38~<br>3.40E + 38 |
| 비트 | Bit | 0 또는 1 | Number(1) | |
| 화폐 | Money | 약 -922조~ +922조<br>(소수 4자리) | Number(19,4) | |
| | SmallMoney | -214,748.3648 ~<br>+214,748.3647 | Number(10,4) | |
| 날짜 | Datatime | 1/1000 초 단위<br>까지 저장 | Timestamp | 날짜와 시간 및<br>단편 형 초 저장 |
| | Smalldatatime | 분 단위 까지 저장 | date | 분 단위 까지 저장 |
| CLOB | image | 2GB | CLOB | 4GB |
| | Varbinary(max) | 2GB | | |
| BLOB | Text / Ntext | 2GB | BLOB | 4GB |
| | Varchar(max)/<br>Nvarchar(max) | 2GB | Long RAW | 2GB |
| Binary | Binary/Varbinary | 8000 바이트 | RAW | 2000 바이트 |
| GUID | uniqueidentifier | 16 바이트 | | |
| XML | xml | 2GB | XMLType | 4GB |
| 행버전 | Rowversion/<br>timestamp | 8 바이트 | | |
| 공통 | Sql_variant | | | |

다음은 SQL Server 에서 데이터 형식 사용 시 주의 사항입니다.

① Oracle에서는 char 데이터 형식과 varchar 데이터 형식에 값을 저장할 때 특히 주의를 기울어야 합니다. 문자형의 비교 연산 시 일단 데이터의 길이를 비교하므로 길이가 다른 경우에는 동일한 값으로 처리하지 않습니다. 반면에 SQL Server에서는 저장된 값의 데이터 형식의 길이나 후행 공백과 상관없이 비교 연산을 합니다.

② SQL Server 2005에서는 TEXT, IMAGE 데이터 형식 보다는 varchar(max), varbinary(max) 데이터 형식 사용을 권장합니다.

③ 화폐 데이터 형식인 Money, Smallmoney 데이터 형식은 통화 기호와 같이 저장할 수 있습니다.

④ 날짜 데이터 형식과 getdate( ) 함수를 함께 사용할 수 있습니다. getdate() 함수는 천분의 일초 단위까지 반환합니다.

⑤ uniqueidentifier 데이터 형식은 newid( ) 함수를 DEFAULT 제약 조건과 함께 사용하면 16 바이트 크기의 GUID 값을 생성합니다.

⑥ rowversion / timestamp 데이터 형식 가운데에서 rowversion 데이터 형식의 사용을 권장합니다. timestamp 데이터 형식은 ANSI가 지정한 날짜를 저장하는 timestamp 데이터 형식과 혼동할 수 있습니다. 또한 Oracle의 timestamp 데이터 형식과도 유사하지 않습니다. SQL Server의 rowversion(timestamp) 데이터 형식은 각 데이터베이스 별로 처리되며 행의 변경이 발생하는 경우 계속 증가합니다.

⑦ sql_variant 데이터형식은 8000 바이트 이내의 문자, 숫자, 바이너리 유형의 데이터를 저장할 수 있습니다. 단, text, ntext, imate, rowversion(timestamp) 데이터 형식과는 호환되지 않습니다.

⑧ 각 데이터 형식의 자세한 사항은 SQL Server 2005 온라인 설명서를 참조하기 바랍니다.

## T-SQL vs. PL/SQL

### ■ SELECT

| 설 명 | SQL Server | Oracle |
|---|---|---|
| 스칼라 함수 사용 | SELECT getdate( ) | SELECT sysdate FROM DUAL |
| 컬럼 별칭 | SELECT name '별칭'<br>FROM emp<br>SELECT name AS '별칭'<br>FROM emp<br>SELECT '별칭' =name<br>FROM emp | SELECT name "별칭"<br>FROM emp |
| 문자열 연결 | SELECT firstname + ',' +lastname | SELECT firstname \|\| ',' \|\| firstanme |
| 문 종료 문자 | 세미 콜론 (;) | 세미 콜론 (;) |
| 문 주석 | /* ~ */ , -- | /* ~ */ , -- |
| 변수 선언 | DECLARE | DECLARE |
| 변수 할당 | SET<br>SELECT<br>FETCH INTO | :=<br>SELECT ~ INTO<br>FETCH INTO |
| 문 블록 | BEGIN ~ END | BEGIN ~ END |
| 무조건 종료 | RETURN | RETURN |
| 스트링 출력 | PRINT | DBMS_OUTPUT.PUT_LINE |
| 오류 발생 | RAISERROR | RAISE_APPLICATION_ERROR |

| | | |
|---|---|---|
| TOP-n 쿼리 | SELECT TOP (5) * FROM emp<br>ORDER BY name<br>(SET ROWCOUNT 5<br>SELECT * FROM emp<br>ORDER BY name) | SELECT * FROM<br>(SELECT * FROM emp<br>ORDER BY name) as E<br>WHERE ROWNUM <=5 |
| 값이 NULL인 경우 변형 | ISNULL(column, val) | NVL(column, val) |
| 합집합 | UNION | UNION |
| 교집합 | INTERSECT | INTERSECT |
| 차집합 | EXCEPT | MINUS |
| INNER JOIN | SELECT *<br>FROM a JOIN b<br>ON a.col=b.col | SELECT *<br>FROM a JOIN b<br>ON {(a.col=b.col) \| USING (col)} |
| | SELECT *<br>FROM a , b<br>WHERE a.col=b.col | SELECT *<br>FROM a , b<br>WHERE a.col=b.col |
| LEFT OUTER JOIN (Oracle9i ANSI JOIN 지원) | SELECT *<br>FROM a LEFT OUTER JOIN b<br>ON a.col=b.col | SELECT *<br>FROM a LEFT OUTER JOIN b<br>ON (a.col=b.col) |
| | T-SQL 조인(*= / =*)은 SQL Server 2005 에서 지원하지 않음 | SELECT * FROM a,b<br>WHERE a.col=b.col(+) |
| FULL OUTER JOIN | SELECT *<br>FROM a FULL OUTER JOIN b<br>ON a.col=b.col | SELECT *<br>FROM a FULL OUTER JOIN b<br>ON (a.col=b.col) |

| | | |
|---|---|---|
| CROSS JOIN | SELECT *<br>FROM a CROSS OUTER JOIN b | SELECT *<br>FROM a CROSS OUTER JOIN b |
| | SELECT *<br>FROM a , b | SELECT *<br>FROM a , b |
| SUB-QUERY | SELECT 절, FROM 절, WHERE 절,<br>ORDER BY 절 등 | SELECT 절, FROM 절, WHERE 절,<br>ORDER BY 절 등 |
| 계층 구조<br>재귀 쿼리 | Common Table Expressions<br>(공통 테이블식) | CONNECT BY ~<br>START WITH ~ |
| IF 구문 | IF ~ BEGIN ~ END<br>ELSE IF ~ BEGIN ~END<br>ELSE BEGIN ~ END | IF ~ THEN<br>ELSIF ~ THEN<br>END IF |
| CASE 구문 | CASE WHEN ~THEN ~<br>ELSE THEN ~ END | CASE WHEN ~THEN ~ELSE ~END /<br>DECODE( ) |
| 반복(루프) 구문 | WHILE ~<br>BEGIN<br>END | WHILE~END LOOP<br>LOOP ~END LOOP<br>FOR ~ END LOOP |
| 현재 루프<br>블록 탈출 | BREAK | EXIT |

***[참고]***
*계층 구조 재귀쿼리는 온라인 설명서의 [공통 테이블식을 사용하는 재귀쿼리]의 설명과 예제를 참조하거나 [Microsoft SQL Server 2005 개발자 가이드]를 참조하기 바랍니다.*

## ■ DML

대부분의 INSERT, UPDATE, DELETE 구문은 수정할 필요 없이 사용이 가능합니다.
단, Oracle 9i 이후 버전에서 지원하는 INSERT ALL 구문 또는 MERGE 구문 등은 별도로 수정해야 하며 SQL Server는 DML 구문에서도 JOIN 절의 사용을 지원합니다.

| 구 문 | SQL Server | Oracle |
|---|---|---|
| 다중 INSERT | INSERT [INTO] table_name1<br>OUTPUT Deleted.column_name<br>INTO {table_name2 \| table_variable}<br>VALUES | INSERT ALL ~ |
| JOIN INSERT | INSERT [INTO] table_name<br>SELECT ~<br>FROM ~ JOIN ~<br>ON ~<br>WHERE ~ | Oracle의 DML에서 JOIN 구문을 지원하지 않습니다. 따라서, SUB-QUERY를 사용합니다. |
| JOIN UPDATE | UPDATE table_name<br>SET ~<br>FROM table_name Alias<br>JOIN ~<br>ON ~<br>WHERE ~ | 상 동 |
| JOIN DELETE | DELETE table_name<br>FROM table_name Alias<br>JOIN ~<br>ON ~<br>WHERE ~ | 상 동 |

SQL Server 2005에서 제공하는 OUTPUT 절은 Oracle의 RETURNING 절과도 유사합니다.
단, SQL Server 2005의 OUTPUT 절은 변수뿐만 아니라 일반 테이블에도 값을 입력할 수 있습니다.

### ■ 동적 SQL

Oracle과 SQL Server는 각각 2가지 방법으로 동적 SQL 구문을 실행할 수 있습니다. Oracle은 DBMS_SQL 시스템 패키지를 사용하거나 EXECUTE IMMEDIATE 구문을 통해서 동적 SQL을 수행하고 SQL Server는 sp_executesql 시스템 저장 프로시저를 사용하거나 EXECUTE() 구문을 통해서 동적 SQL 구문을 사용할 수 있습니다. sp_executesql 시스템 저장 프로시저의 사용은 유니코드 데이터 형식을 매개 변수로 받으므로 쿼리의 크기는 4000바이트로 제한 됩니다. 그렇지만 EXECUTE( ) 구문의 사용보다 쿼리의 재 컴파일을 줄일 수 있어서 권장하는 방법입니다만 EXECUTE( ) 구문은 작성하기가 용이해서 많이 사용하는 방법입니다. 각각의 사용법은 다음과 같습니다.

① sp_executesql

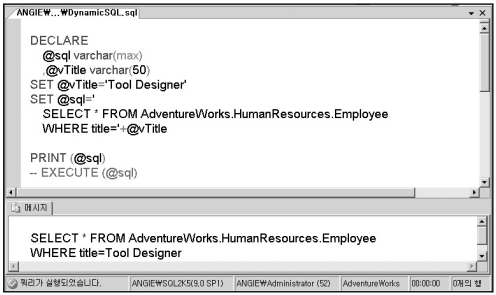
```
EXECUTE sp_executesql
N' SELECT * FROM  AdventureWorks.HumanResources.Employee WHERE
title=@vTitle'
, N' @vTitle varchar(50)' , @vTitle=N' Tool Designer'
```

sp_executesql 시스템 저장 프로시저는 세 부분으로 구성됩니다.
먼저 ① 매개 변수를 받는 쿼리 전체 부분과 ②매개 변수의 선언 부분, 그리고 ③매개 변수에 값을할당하는 부분으로 구성됩니다.

② EXECUTE( )

```
DECLARE
        @sql varchar(8000)
        ,@vTitle varchar(50)
SET @vTitle=' Tool Designer'
SET @sql='
        SELECT * FROM AdventureWorks.HumanResources.Employee
        WHERE title=' +@vTitle
EXECUTE (@sql)
```

EXECUTE( ) 구문은 쿼리 전체를 스트링 형태의 매개 변수로 받아서 해당 스트링을 수행하는 방법을 사용합니다. 따라서, 다음과 같이 PRINT 명령으로 구문의 유효성을 먼저 확인하면 편리합니다.




## 트랜잭션 처리

### ■ 트랜잭션 유형

- 시스템 트랜잭션 : 트랜잭션의 시장과 종료를 시스템에서 자동으로 처리합니다.
- 암시적 트랜잭션 : 트랜잭션의 시작은 시스템에서 매 문장마다 실행하며 종료는 사용자에 의해서 명시적으로 처리하며 SET IMPLICIT_TRANSACTIONS ON 구문을 통해서 활성화 됩니다.
- 명시적 트랜잭션 : 트랜잭션의 시작과 종료를 모두 사용자가 처리합니다. 오류 처리 로직이 필요하며 SQL Server는 DDL도 명시적 트랜잭션에 포함시킬 수 있습니다.

- 중첩 트랜잭션 : 트랜잭션 내에서 추가 적인 트랜잭션을 시작할 수 있으며 커밋은 열려 있는 트랜잭션의 수만큼 수행해야 하며 롤백은 현재 열려있는 모든 트랜잭션을 롤백합니다.
- 분산 트랜잭션 : 하나 이상의 인스턴스가 참여하는 트랜잭션으로 Windows Server의 분산 트랜잭션 코디네이더(MSDTC)에 의해서 2 Phase Commit 프로토콜로 처리됩니다. XACT_ABORT 옵션을 반드시 활성화 해야 합니다.

| 트랜잭션 유형 | SQL Server | Oracle |
|---|---|---|
| 시스템 트랜잭션 | 지원 / 기본 설정 | DDL 수행 시 |
| 암시적 트랜잭션 | SET IMPLICIT_TRANSACTIONS {ON \|OFF} | 기본 설정 |
| 명시적 트랜잭션 | 지원 | 지원하지 않음 |
| 중첩 트랜잭션 | 지원 | 지원하지 않음 |
| 분산 트랜잭션 | 지원 (MSDTC) | 지원 |

① SQL Server와 Oracle은 트랜잭션을 처리하는 기본 설정이 다르므로 개발자가 혼란스러울 수 있습니다. Oracle과 같이 시스템에서 트랜잭션을 시작하고, 커밋과 롤백은 사용자가 처리하기 위해서는 SET IMPLICIT_TRANSACTIONS 세션 옵션을 활성화 하여 처리할 수 있습니다. 단, SQL Server와 Oracle은 잠금을 처리하는 방식이 서로 다르므로 본 모듈의 [트랜잭션의 고립화 수준]과 [트랜잭션 고립화 수준 관리]섹션 부분을 이해한 뒤에 설정하여 사용하기 바랍니다.

② 명시적인 트랜잭션의 사용은 사용자의 적절한 오류 처리를 필요로 합니다. 본 모듈의 [오류 처리]섹션 부분을 이해하기 바랍니다.

③ 중첩 트랜잭션은  트랜잭션 내에서 추가 적인 트랜잭션을 시작할 수 있으며 커밋은 열려 있는 트랜잭션의 개수만큼 수행해야 하며 롤백은 한번의 지정으로 해당 세션에서 현재 열려있는 모든 트랜잭션을 롤백합니다. 현재 열려있는 트랜잭션의 개수 확인은 @@trancount 시스템 함수를 통해서 확인할 수 있습니다.

④ 분산 트랜잭션은 하나이상의 SQL Server 인스턴스가 참여하는 트랜잭션입니다.
SQL Server 뿐만 아니라 Oracle 등 다른 DBMS등이 참여한 경우에도 분산 트랜잭션으로 처리됩니다. 이와 같은 경우에 분산 트랜잭션의 관리는 SQL Server가 담당하는 것이 아니라 Windows Server의 분산 트랜잭션 코디네이터(MSDTC) 서비스에 의해서 2 Phase Commit 프로토콜을 사용해서 처리하게 됩니다. 따라서, 분산 트랜잭션을 사용하고자 하는 경우에는 다음과 같이 Windows Server의 MSDTC 서비스의 네트워크 DTC 액세스 설정과 보안 구성을 적절하게 구성해야 합니다.

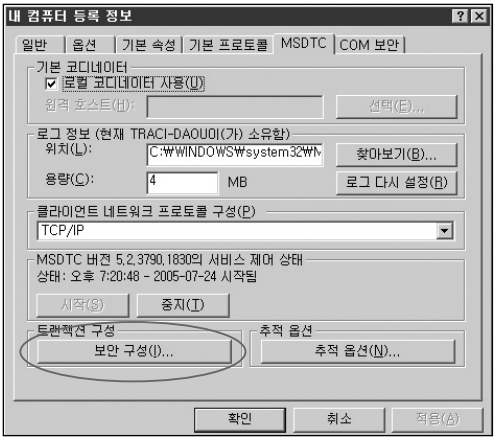
**■ Windows 2003 Server DTC 구성**

Windows 2003 Server는 기본적으로는 로컬 호스트 안에서의 분산 트랜잭션만을 허용하고 다중의 호스트가 참여하는 분산 트랜잭션에 대해서는 구성되어 있지 않습니다.
따라서, 다중의 호스트와의 분산 트랜잭션을 가능하게 하기 위해서는 구성 요소 서비스의 내 컴퓨터의 속성을 선택하고 MSDTC 탭을 선택하면 [로컬 코디네이터 사용(U)]에 체크가 되어있음을 확인합니다. 이어서 하단에 있는 [트랜잭션 구성의 보안구성(I)] 버튼을 눌러 [네트워크 DTC 액세스(D)]의 체크박스 상태를 확인합니다. 만약 이 체크박스가 선택되어 있지 않다면 다중 호스트의 분산 트랜잭션이 구성되어 있지 않다는 것을 의미합니다.

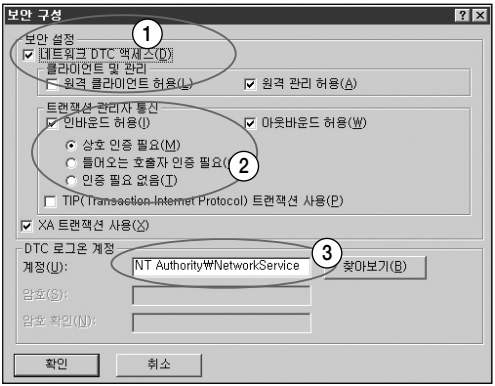
다른 호스트와의 분산 트랜잭션을 처리하기 위해서는 다음의 단계를 진행해야 합니다.

① 네트워크 DTC 액세스 설정
② 네트워크 DTC의 보안 구성

먼저, 네트워크 DTC 액세스 설정입니다.
관리 도구에서 구성 요소 서비스를 실행하거나 시작에서 실행을 선택한 후 Dcomcnfg를 입력한 후 [확인] 버튼을 누릅니다. 내 컴퓨터의 속성을 선택합니다. 아래의 그림과 같은 창이 실행되면 밑부분의 트랜잭션 구성의 [보안구성(I)] 버튼을 누릅니다.


내 컴퓨터 등록 정보
일반
옵션
기본 속성
기본 프로토콜
MSDTC
COM 보안
기본 코디네이터
로컬 코디네이터 사용(U)
원격 호스트(H):
선택(E)...
로그 정보 (현재 TRACI-DAOUOI(가) 소유함)
위치(L):
C:₩WINDOWS₩system32₩M
찾아보기(B)...
용량(C):
4
MB
로그 다시 설정(R)
클라이언트 네트워크 프로토콜 구성(P)
TCP/IP
MSDTC 버전 5.2.3790.1830의 서비스 제어 상태
상태: 오후 7:20:48 - 2005-07-24 시작됨
시작(S)
중지(T)
트랜잭션 구성
보안 구성(I)...
추적 옵션
추적 옵션(N)...
확인
취소
적용(A)

다음 그림과 같이 보안 구성 창에서 ①번으로 표시된 [네트워크 DTC 액세스(D)]의 체크박스를 선택합니다. 창을 닫기 위해 확인 버튼을 누르면 MSDTC 서비스를 재 시작합니다.
이제 1단계가 완료되었습니다.




두 번째 단계는 네트워크 DTC의 보안 구성 단계입니다. 위 그림의 보안 구성 창에서 ②번 부분과같이 설정합니다. DTC 로그온 계정은 기본적으로 NT Authority\NetworkService 계정으로 설정되어 있습니다.

### ■ J2EE를 사용하는 경우 DTC 구성

JEUS와 같은 J2EE 기반 WAS에서 로컬 트랜잭션 또는 분산 트랜잭션을 처리하기 위해서는 SQL Server 2005에서 다음과 같은 추가 작업을 해야 합니다.

1) SQL Server 2005용 JDBC 드라이버를 다운로드 받지 않은 경우에는 다음의 사이트로 이동하여 다운로드합니다. UNIX용 버전과 Windows용 버전을 모두 다운받을 수 있습니다.

http://www.microsoft.com/downloads/details.aspx?displaylang=ko&FamilyID=e22bc83b-32ff-4474-a44a-22b6ae2c4e17

2) JDBC가 설치된 폴더로 이동합니다.

[C:\Microsoft SQL Server 2005 JDBC Driver\sqljdbc_1.0\kor\xa]

3) SQL Server 2005 버전이 32bit인 경우에는 X32폴더의, 64bit 인 경우에는 X64 폴더의 sqljdbc_xa.dll 파일을 SQL Server 2005 인스턴스의 binn 폴더로 복사합니다. 다중의 SQL Server 2005 인스턴스가 분산 트랜잭션에 참여하는 경우에는 모든 인스턴스의 binn 폴더로 복사해야 합니다.

SQL Server 2005 기본 인스턴스의 binn 폴더 경로〉
C:₩Program Files₩Microsoft SQL Server₩MSSQL.1₩MSSQL₩binn

4) JDBC가 설치된 경로의 xa 폴더의 xa_install.sql 파일을 SQL Server Management Studio에서 실행합니다. xa_install.sql 파일은 다음과 같은 작업을 수행합니다.

① sqljdbc_xa.dll 파일을 이용하여 다음의 확장 저장 프로시저 생성
- xp_sqljdbc_xa_init
- xp_sqljdbc_xa_start

- xp_sqljdbc_xa_end
- xp_sqljdbc_xa_prepare
- xp_sqljdbc_xa_commit
- xp_sqljdbc_xa_rollback
- xp_sqljdbc_xa_forget
- xp_sqljdbc_xa_recover

② master 데이터베이스에 [SqlJDBCXAUser] 데이터베이스 역할 생성
③ [SqlJDBCXAUser] 역할에 ①번 작업에서 생성한 확장 저장 프로시저 실행 권한 부여

5) 다음 구문을 실행해서 SQL Server 2005에 연결 사용자 계정을 master 데이터베이스에도 생성하고 [SqlJDBCXAUser] 역할에 포함합니다.

```
USE master
GO
CREATE USER '연결 계정 이름' FOR LOGIN '연결 계정 이름'
GO
EXEC sp_addrolemember [SqlDBCXAUser], '연결 계정 이름'
GO
```

**■ 잠금의 종류**

SQL Server와 Oracle은 모두 행 수준의 잠금과 테이블 수준의 잠금을 지원합니다.
SQL Server는 잠금의 수준을 보다 세분화해서 페이지 수준의 잠금과 익스텐트 수준의 잠금을 지원합니다. 또한 SQL Server는 잠금과 관련된 리소스를 절약하기 위해서 트랜잭션이 지정된 임계값을 초과하여 잠금을 얻게 되면 행 수준의 잠금에서 테이블 수준의 잠금으로 에스컬레이션하는 기능도 제공합니다.

SQL Server는 잠금을 잠금 관리자를 통해서 자동으로 관리하지만 본 모듈의 [트랜잭션 고립화 수준 관리]섹션의 설명과 같이 사용자가 세션 수준이나 쿼리 수준에서 잠금을 제어할 수 있습니다. SQL Server와 Oracle의 잠금의 종류는 다음과 같습니다.

| SQL Server | Oracle | 설 명 |
|---|---|---|
| Shared (S) | Share (S) | SELECT 문처럼 데이터를 변경하거나 업데이트하지 않는 읽기 작업에 사용합니다. |
| Update (U) | Row Share(RS) | 업데이트할 수 있는 리소스에 사용합니다. 여러 개의 세션이 리소스를 읽고, 잠그고, 나중에 업데이트할 때 발생하는 일반적인 교착 상태를 방지합니다. |
| Exclusive (X) | Row Exclusive (RX) | INSERT, UPDATE, DELETE와 같은 데이터 수정 작업에 사용합니다. 여러 개의 업데이트 작업이 같은 리소스에 대해 동시에 이루어지지 못하게 합니다. |
| Exclusive (X) | Shared Row Exclusive (SRX) | 수정하지 않는 S 잠금 및 RS 잠금만을 허용합니다. |
| Exclusive (X) | Exclusive (X) | |
| Intent Shared(IS) | | 계층 구조의 아래쪽에 있는 일부 리소스에 대해 요청되거나 확보된 공유 잠금을 보호합니다. |
| Intent Exclusive(IX) | | 계층 구조의 아래쪽에 있는 일부 리소스에 대해 요청되거나 확보된 배타 잠금을 보호합니다. IX는 IS의 상위 집합으로, 하위 수준 리소스에 대한 공유 잠금 요청도 보호합니다. |

| | | |
|---|---|---|
| Shared with Intent Exclusive(SIX) | | 계층 구조의 아래쪽에 있는 모든 리소스에 대해 요청되거나 확보된 공유 잠금 및 하위 수준 리소스 일부에 대해 요청되거나 확보된 의도 배타 잠금을 보호합니다. 최상위 수준 리소스에서는 동시 IS 잠금이 허용됩니다. 예를 들어 테이블에 대한 SIX 잠금을 확보하면 수정되는 페이지에 대한 의도 배타 잠금 및 수정되는 행에 대한 배타 잠금도 동시에 확보됩니다. 리소스당 한 번에 하나의 SIX 잠금을 설정할 수 있으므로 다른 트랜잭션이 테이블 수준에서 IS 잠금을 얻어 계층 구조 아래쪽에 있는 리소스를 읽을 수는 있어도 다른 트랜잭션이 리소스를 업데이트할 수는 없습니다. |
| Schema Modification (Sch-M) | Exclusive DDL | 열을 추가하거나 테이블을 삭제하는 등 테이블 DDL(데이터 정의 언어) 작업이 수행 중일 때 사용합니다. |
| Schema Stability (Sch-S) | Breakable Parse | 쿼리를 컴파일 할 때 사용 |
| Bulk-Update (BU) | | 데이터를 테이블로 대량 복사하는 경우와 TABLOCK 힌트가 지정된 경우 사용합니다. |

### ■ 트랜잭션 고립화 수준 및 관리

다음은 SQL Server 2005가 지원하는 다섯 개 수준의 트랜잭션 격리 수준이며 Oracle은 이 가운데에서 READ COMMITTED와 SERIALIZABLE 두 가지 트랜잭션 격리 수준을 지원합니다.

| 고립화 수준 | 설 명 |
|---|---|
| READ UNCOMMITTED | 트랜잭션이 읽기 작업동안 잠금을 획득하지 않음 |
| READ COMMITTED | 트랜잭션이 읽기 작업 동안 공유 잠금을 획득하고 읽기 기간 동안 보유 |
| REPEATABLE READ | 트랜잭션이 읽기 작업 동안 공유 잠금을 획득하고 트랜잭션 기간 동안 보유 |
| SNAPSHOT | 트랜잭션이 읽기 작업 시작 시 트랜잭션 별로 데이터의 버전의 읽기 일관성을 유지 |
| SERIALIZABLE | 트랜잭션이 읽기 작업 동안 공유 잠금 및 범위 잠금을 획득하고 트랜잭션 기간 동안 보유 |

SQL Server에서 트랜잭션의 고립화 수준은 세션 수준 또는 쿼리 수준에서 설정할 수 있습니다. 트랜잭션 고립화 수준의 관리는 다음과 같이 세션 수준에서는 세션 옵션을 활성화 하여 관리하며 쿼리 수준에서는 테이블 힌트를 통해서 관리합니다.

① 세션 수준

```
SET TRANSACTION ISOLATION LEVEL
  { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |
SNAPSHOT | SERIALIZABLE }
```

② 쿼리 수준

```
~ FROM table_name
WITH ( { READUNCOMMITTED | READCOMMITTED | REPEATABLEREAD |
SERIALIZABLE
      | NOLOCK | HOLDLOCK | PAGLOCK | ROWLOCK | TABLOCK | TABLOCKX
| UPDLOCK | XLOCK | READPAST } )
```

**■ 행의 다중 버전을 통한 일관성 관리**

Oracle은 명시적으로 또는 암시적으로 데이터를 읽는 모든 SQL 구문에서 다중 버전의 읽기 일관성을 지원합니다. 읽기 일관성이란 데이터를 읽기 위해서 잠금을 얻거나 다른 잠금이 해제될 때까지 기다리지 않고 데이터의 행을 읽을 수 있도록 롤백 세그먼트(UNDO 테이블 스페이스)를 사용해서 데이터 행의 스냅샷을 작성해서 최근 커밋된 데이터를 제공합니다. SQL Server 2005는 다음 두 가지 방법으로 이와 같은 기능을 구현할 수 있습니다.

① 모든 데이터베이스 사용자에게 행 버전을 통한 읽기 일관성 제공
② 해당 세션 옵션을 활성화한 사용자에게만 행 버전을 통한 읽기 일관성 제공

모든 데이터베이스 사용자에게 행 버전을 통한 읽기 일관성을 제공하기 위해서는 다음과 같이 READ_COMMITTED_SNAPSHOT 데이터베이스 옵션을 활성화 합니다.

```
ALTER DATABASE database_name
SET READ_COMMITTED_SNAPSHOT { ON | OFF }
[ WITH {NO_WAIT | ROLLBACK {  IMMEDIATE | AFTER n [SECONDS] }} ]
```

해당 세션 옵션을 활성화한 사용자에게만 행 버전을 통한 읽기 일관성을 제공하기 위해서는 다음과 같이 ALLOW_SNAPSHOT_ISOLATION 데이터베이스 옵션을 활성화 하고, 해당 세션에서 SNAPSHOT 트랜잭션 고립화 수준을 활성화합니다.

```
-- ALLOW_SNAPSHOT_ISOLATION 데이터베이스 옵션을 활성화
ALTER DATABASE database_name
SET ALLOW_SNAPSHOT_ISOLATION { ON | OFF }
GO
-- 세션 수준의 SNAPSHOT 트랜잭션 고립화 수준을 활성화
SET TRANSACTION ISOLATION LEVEL SNAPSHOT
GO
```

***[참고]***
*이와 같이 행 버전을 통한 읽기 일관성을 지원하는 경우에 SQL Server 2005는 Oracle 이 롤백 세그먼트(UNDO 테이블스페이스)를 사용하는 것과 같이 tempdb를 사용합니다. 따라서, tempdb의 성능 향상을 위해서 tempdb의 데이터 파일을 동일한 크기로 지정해서 CPU의 개수 만큼 별도의 물리적인 디스크 공간에 생성합니다.*

### ■ 오류 처리

SQL Server 2005는 C# 및 C++, JAVA 언어의 예외 처리와 유사한 TRY/CATCH 구문을 지원합니다. TRY/CATCH 구문은 발생한 오류의 심각도가 10을 넘으며 데이터베이스 연결을 끊는 심각한 오류가 아닌 모든 오류를 포착할 수 있으며 T-SQL 구문을 블록으로 묶어서 해당 TRY 블록 내에서 오류가 발생하는 경우 바로 다음에 정의된 CATCH 블록으로 제어가 전달되도록 합니다.

TRY/CATCH 구문을 사용하기 위해서는 XACT_ABORT 세션 옵션을 활성화해야 합니다. 또한,TRY/CATCH 구문은 여러 일괄 처리나 T-SQL 블록에 걸쳐서 정의 될 수 없으며 TRY 블록 다음에는 곧바로 연관된 CATCH 블록이 이어져야 합니다. END TRY와 BEGIN CATCH 사이에 다른 구문을 두면 구문 오류가 발생하게 됩니다.

다음은 CATCH 블록에서 사용할 수 있는 함수입니다.

| SQL Server 함수 | 설 명 |
|---|---|
| ERROR_NUMBER( ) | 오류 번호 반환 |
| ERROR_SEVERITY( ) | 심각도 반환 |
| ERROR_STATE( ) | 상태 정보 반환 |
| ERROR_PROCEDURE( ) | 오류가 발생한 저장 프로시저 이름 반환 |
| ERROR_LINE( ) | 오류가 발생한 루틴의 줄 번호 반환 |
| ERROR_MESSAGE( ) | 오류 메시지 반환 |
| XACT_STATE( ) | 트랜잭션이 커밋하지 못하는 경우에 -1 반환 |

다음은 TRY/CATCH 구문입니다.

```
BEGIN TRY
    BEGIN TRAN
                T-SQL 트랜잭션 구문
    COMMIT
END TRY
BEGIN CATCH
    ROLLBACK TRAN
    T-SQL 오류 처리 구문
END CATCH
GO
```

## 시스템 함수

SQL Server와 Oracle에서 제공하는 시스템 함수는 다음과 같습니다. 각 함수의 매개 변수와 자세한 사항은 SQL Server 2005의 온라인 설명서를 참조하기 바랍니다.

### 1) 문자열 함수

| SQL Server | Oracle | 설 명 |
|---|---|---|
| ASCII | ASCII | 문자 식에서 가장 왼쪽 문자의 ASCII 코드 값을 반환 |
| CHAR \| NCHAR | CHR | ASCII 코드를 문자(유니코드)로 변환 |
| CHARINDEX \| PATINDEX | INSTR | 문자열에서 지정한 식의 시작 위치를 반환 |
| DIFFERENCE | | 두 문자 식에서 SOUNDEX 값의 차이를 나타내는 정수 값을 반환 |
| LEFT | (SUBSTR) | 문자열의 왼쪽부터 지정된 수만큼의 문자를 반환 |
| LEN | LENGTH | 지정된 문자열 식의 바이트 수 대신 후행 공백을 제외한 문자 수를 반환 |
| DATALENGTH | LENGTHB | 식을 표시하는 데 사용된 바이트 수를 반환 |
| LOWER | LOWER | 대문자 데이터를 소문자 데이터로 변환한 문자 식을 반환 |
| LTRIM | LTRIM | 선행 공백을 제거한 문자 식을 반환 |
| REPLACE | REPLACE /TRANSLATE | 첫 번째 문자열 식에서 두 번째 문자열 식에 해당되는 모든 항목을 찾아 세 번째 식으로 대체 |

| | | |
|---|---|---|
| REPLICATE | REPLICATE | 지정한 횟수만큼 문자 식을 반복 |
| REVERSE | | 문자 식을 반대로 반환 |
| RIGHT | (SUBSTR) | 지정된 문자 수만큼 문자열의 오른쪽 부분을 반환 |
| RTRIM | RTRIM | 후행 공백을 제거한 문자 식을 반환 |
| SOUNDEX | SOUNDEX | 두 문자열의 유사성을 평가하기 위한 4자의 SOUNDEX 코드를 반환 |
| SPACE | (RPAD) | 반복된 공백 문자열을 반환 |
| STUFF | (SUBSTR) | 지정한 시작 지점에서 지정한 문자 길이를 삭제한 다음 다른 문자 집합을 삽입 |
| SUBSTRING | SUBSTR | 문자, 이진, 텍스트 또는 이미지 식의 일부를 반환 |
| UPPER | UPPER | 소문자 데이터를 대문자로 변환한 문자 식을 반환 |

2) 날짜 및 시간 함수

| SQL Server | Oracle | 설 명 |
|---|---|---|
| DATEADD | 날짜 열의 +/- 또는 ADD_MONTHS | 지정한 날짜에 시간 간격을 더하여 새 datetime 값을 반환 |
| DATEDIFF | 날짜 열의 +/- 또는 MONTHS_BETWEEN | 지정한 두 날짜 사이에 있는 날짜와 시간 경계의 수를 반환 |
| DATENAME | TO_CHAR | 지정한 날짜의 특정 부분을 나타내는 문자 열을 반환 |
| DATEPART | EXTRACT | 지정한 날짜의 특정 부분을 나타내는 정수를 반환 |
| DAY | TO_NUMBER | 지정한 날짜의 일 datepart 부분을 나타내는 정수를 반환<br>(DATEPART(dd, date)와 동일함) |
| GETDATE | SYSDATE | 현재 시스템 날짜와 시간을 datetime 값을 반환 |
| GETUTCDATE | SYS_EXTRACT_UTC | 현재 UTC 시간(국제 표준시 또는 그리니치 표준시)을 나타내는 datetime 값을 반환 |
| MONTH | TO_NUMBER | 지정된 날짜의 월 부분을 나타내는 정수를 반환<br>(DATEPART(mm, date)와 동일) |
| YEAR | TO_NUMBER | 지정한 날짜의 연도 부분에 해당하는 정수를 반환<br>(DATEPART(yy, date)와 동일) |

3) 수치 연산 함수

| SQL Server | Oracle | 설 명 |
|---|---|---|
| ABS | ABS | 지정한 수식의 절대(양수) 값을 반환 |
| CEILING | CEIL | 지정한 숫자 식보다 크거나 같은 최소 정수를 반환 |
| COS | COS | 지정한 식에서 지정한 각도의 삼각법 코사인을 라디안 단위로 반환 |
| COT | COT | 지정한 float 식에서 지정한 각도의 삼각법 코탄젠트를 라디안 단위로 반환 |
| DEGREE | | 라디안으로 지정된 각도를 도 단위로 반환 |
| EXP | EXP | 지정한 float 식의 지수 값을 반환 |
| FLOOR | FLOOR | 지정된 숫자 식보다 작거나 같은 최대 정수를 반환 |
| LOG | LOG | 지정된 float 식의 자연 로그를 반환 |
| LOG10 | LOG | 지정된 float 식의 상용 로그를 반환 |
| % | MOD/REMAINDER | 나머지 값 반환 |
| PI | | PI의 상수 값을 반환 |
| POWER | POWER | 지정된 식을 거듭제곱한 값을 반환 |
| RADIANS | | 숫자 식을 도 단위로 입력하면 라디안을 반환 |
| RAND | | 0부터 1까지의 임의 float 값을 반환 |
| ROUND | ROUND | 특정 길이나 전체 자릿수로 반올림한 숫자 식을 반환 |

| | | |
|---|---|---|
| SIGN | SIGN | 지정된 식의 양수(+1), 영(0) 또는 음수(-1) 기호를 반환 |
| SIN | SIN | 지정된 각도의 삼각 사인을 근사치 float 식에서 라디안으로 반환 |
| SQRT | SQRT | 지정한 식의 제곱근을 반환 |
| SQUARE | (POWER) | 지정된 식의 제곱을 반환 |
| TAN | TAN | 입력 식의 탄젠트를 반환 |

***[참고]***
*ABC, CEILIING, DEGREES,FLOOR, POWER, RADINAS, SIGN 등의 산술 함수는 입력 값과 동일한 데이터 형식의 값을 반환합니다. EXP, LOG, LOG10, SQUARE, SQRT를 포함하여 삼각 함수 등의 기타 함수는 입력 값을 FLOAT 데이터 형식으로 변환하고 FLOAT값을 반환합니다.*

4) 집계 함수

| SQL Server | Oracle | 설 명 |
|---|---|---|
| AVG | AVG | NULL 값을 무시하고 그룹에 속한 값의 평균을 반환 |
| COUNT/ COUNT_BIG | COUNT | 그룹에 포함된 항목 개수를 반환 |
| GROUPING | GROUPING | 행이 CUBE 또는 ROLLUP 연산자를 통해 추가될 때는 추가 열을 1로, 행이 CUBE 또는 ROLLUP의 결과가 아닐 때는 추가 열을 0으로 출력 |
| MAX | MAX | 식의 최대값을 반환 |
| MIN | MIN | 식의 최소값을 반환 |
| SUM | SUM | 식의 모든 값 또는 DISTINCT 값의 합계를 반환 |
| STDEV | STDEV | 지정한 식의 모든 값에 대한 통계적 표준 편차를 반환 |
| STDEVP | STDEV_POP | 지정한 식에 있는 모든 값의 모집단에 대한 통계적 표준 편차를 반환 |
| VAR | VAR | 지정한 식에 있는 모든 값의 통계적 분산을 반환 |
| VARP | VAR_POP | 지정한 식에 있는 모든 값의 모집단에 대한 통계적 분산을 반환 |

5) 순위 함수

| SQL Server | Oracle | 설 명 |
|---|---|---|
| NTILE | NTILE | 정렬된 파티션의 행을 지정된 수의 그룹으로 분산 |
| DENSE_RANK | DENSE_RANK | 결과 집합 파티션 내 행의 순위를 순위 간격 없이 반환 |
| RANK | RANK | 결과 집합의 파티션 내에 있는 각 행의 순위를 반환 |
| ROW_NUMBER | ROW_NUMBER | 결과 집합 파티션 내의 행 일련 번호를 반환 |

* 순위 함수는 SQL Server 2005, Oracle 9i 이상의 버전에서 지원합니다.

6) 시스템 함수

| SQL Server | Oracle | 설 명 |
|---|---|---|
| CAST / CONVERT | CAST / CONVERT | 식을 다른 데이터 형식으로 명시적으로 변환 |
| COALECSE | COALECSE | 해당 인수 중에서 Null이 아닌 첫 번째 식을 반환 |
| CURRENT_USER | SYS_CONTEXT ( 'USER_ENV' , ' SESSION_USER' ) | 현재 사용자의 이름을 반환. USER_NAME() 과 동일 |
| DB_NAME / DB_ID | ora_database_name/ SYS_CONTEXT ( 'USER_ENV' , ' DB_NAME' ) | 데이터베이스 이름/ID를 반환 |

| | | |
|---|---|---|
| ERROR_LINE | | TRY...CATCH 구문의 CATCH 블록 실행을 유발한 오류가 발생한 줄 번호를 반환 |
| ERROR_ MESSAGE | SQLERRM | TRY···CATCH 구문의 CATCH 블록을 실행시키는 오류의 메시지 텍스트를 반환 |
| ERROR_NUMBER | SQLCODE | TRY...CATCH 구문의 CATCH 블록을 실행시킨 오류의 오류 번호를 반환 |
| ERROR_ PROCEDURE | | TRY···CATCH 구문의 CATCH 블록이 실행되는 오류가 발생한 저장 프로시저나 트리거의 이름을 반환 |
| ERROR_SEVERITY | | TRY...CATCH 구문의 CATCH 블록이 실행되도록 한 오류의 심각도를 반환 |
| ERROR_STATE | | TRY...CATCH 구문의 CATCH 블록을 실행하도록 만든 오류의 상태 번호를 반환 |
| HOST_NAME / HOST_ID | SYS_CONTEXT ( 'USER_ENV' ,' HOST' ) | 워크스테이션 이름/ID를 반환 |
| ISNULL | NVL | NULL을 지정된 대체 값으로 교체 |
| NEWID | | uniqueidentifier 형식의 고유 값 생성 |
| NULLIF | NULLIF | 지정된 두 식이 같으면 Null 값을 반환 |
| OBJECT_NAME/ OBJECT_ID | ora_dict_obj_ name | 스키마 범위 개체에 대한 데이터베이스 개체 이름/ID를 반환 |
| ROWCOUNT_BIG/ @@rowcount | | 최근 실행한 문의 영향을 받은 행 수를 반환 |

| | | |
|---|---|---|
| USER_NAME / USER_ID | USER/ ora_login_user | 지정된 ID 번호/사용자 이름에서 데이터베이스 사용자 이름/ID를 반환 |
| XACT_STATE | | 세션에 활성 트랜잭션이 있는지 여부와 트랜잭션이 커밋될 수 있는지 여부를 나타내는 등 세션의 트랜잭션 상태를 보고 |
| SUSER_SNAME | SYS_CONTEXT ( 'USER_ENV' ,' OS_USER) | SID(보안 ID)와 연결된 로그인 이름을 반환 |

## SQLCMD 유틸리티

SQL Server는 Oracle의 sqlplus와 같이 명령줄 기반으로 T-SQL을 실행할 수 있는 sqlcmd 유틸리티를 제공합니다. 이전 버전의 명령줄 도구인 osql 유틸리티 보다 다양한 추가 기능을 제공하여 sqlplus로 수행하도록 한 스크립트 등을 마이그레이션 하거나 효율적인 관리작업을 수행할 수 있습니다. 향상된 기능은 다음과 같습니다.

① 변수 사용 지원
② 호출자 환경에 오류 정보 전달
③ 운영 체제 명령 실행
④ 관리자 전용 연결(Dedicated Administrator Connection) 지원

관리자 전용 연결(DAC) : 관리자 연결 전용을 위해서 별도의 스케줄러를 사용하여 기존 스케줄러가 "max worker theads" 임계 값에 도달하는 등 스케줄러가 응답할 수 없는 상황에도 관리자는 SQL Server 2005에 연결해서 관리 작업을 수행할 수 있습니다. 이전 버전에서는 스케줄러가 응답하지 않는 상황에서는 관리자가 추가 적인 접속을 할 수 없어서 문제의 진단이나 해결의 작업을 진행하지 못하고 서비스를 재 시작해야 하는 경우가 있었으나 SQL Server 2005에서는 관리자 전용 연결을 통해서 서비스가 중지되거나 일시 중지되지 않은 경우를 제외하고 SQL Server 2005에 언제든지 연결해서 문제의 진단이나 해결작업을 수행할 수 있습니다. 단, 관리자 전용 연결은sqlcmd 유틸리티로 -A 스위치를 사용하거나 SSMS 연결창의 [서버 이름(s)]부분에서 서버 이름 앞에 [admin:]부분을 추가하여 연결하며, 단 하나의 연결만 지원합니다.

1) 다음은 sqlcmd 유틸리티의 구문입니다.

```
sqlcmd [{ { -U login_id [ -P password ] } | -E } ]
[-S server_name [ \ instance_name ] ] [ -H wksta_name ] [ -d db_name ]
[ -q "query" ] [ -Q "query" ]
[ -i input_file ] [ -o output_file ]
[ -v ]
[ -A ]
[ -l time_out ] [ -t time_out ] [ -h headers ]
[ -s col_separator ] [ -w column_width ] [ -a packet_size ]
[ -e ] [ -I ][ -c cmd_end ] [ -L [ c ] ]
[ -m error_level ] [ -V ] [ -W ] [ -u ] [ -r [ 0 | 1 ] ]
[ -f 〈 codepage 〉 | i: 〈 codepage 〉 [ 〈 , o: 〈 codepage 〉 ]
[ -k [ 1 | 2 ] ] [ -y display_width ] [-Y display_width ]
[ -p [ 1 ] ] [ -R ] [ -b ] [ -X [ 1 ] ] [ -x ]
[ -? ]
```

2) 다음은 주요 명령줄 옵션(스위치)입니다.

| 옵션 | 설 명 |
|---|---|
| -U *login_id* [ -P *password* ] | 사용자 로그인 ID와 패스워드를 지정합니다. 로그인 ID와 패스워드는 대소문자를 구분합니다. -U와 -P옵션을 지정하지 않으면 -E 옵션을 지정하지 않더라도 sqlcmd 유틸리티를 실행하는 현재 로그온한 Windows 계정으로 Windows 인증모드를 사용하여 SQL Server에 로그인합니다. SQLCMDUSER/SQLCMDPASSWORD 환경변수로도 지정할 수 있습니다. |
| -E | 사용자 이름과 암호를 사용하는 대신 트러스트된 연결을 사용하여 SQL Server에 로그인합니다. 기본적으로 sqlcmd는 트러스트된 연결 옵션을 사용합니다. |
| -S *server_name* [ ₩ *instance_name* ] | 연결할 SQL Server 인스턴스를 지정합니다. sqlcmd 스크립팅 변수 SQLCMDSERVER를 설정합니다. |
| -H *wksta_name* | 워크스테이션 이름을 지정합니다. 이 옵션은 sqlcmd 스크립팅 변수 SQLCMDWORKSTATION을 설정합니다. 워크스테이션 이름은 **sys.processes** 카탈로그 뷰의 **hostname** 열에서 확인할 수 있습니다. |
| -d *db_name* | 이 변수는 초기 데이터베이스를 지정합니다. sqlcmd를 시작할 때 USE db_name 문을 실행합니다. 이 옵션은 sqlcmd 스크립팅 변수 SQLCMDDBNAME을 설정합니다. |
| -q "*query*" | sqlcmd가 시작될 때 쿼리를 실행하지만 쿼리가 완료되더라도 sqlcmd를 끝내지 않습니다 |
| -Q "*query*" | 쿼리를 실행하고 바로 sqlcmd를 냅니다. |

| | |
|---|---|
| -i *input_file* | SQL 문 또는 저장 프로시저의 일괄 처리가 포함된 파일을 나타냅니다. |
| -o *output_file* | sqlcmd 에서 출력을 받는 파일을 나타냅니다. |
| -v var=" *value*" [ var=" *value*" ...] | sqlcmd 스크립트에서 사용할 수 있는 sqlcmd 스크립팅 변수를 만듭니다. 문자가 포함된 값은 따옴표로 묶습니다. 여러 *var*=" *values*" 값을 지정할 수 있습니다. |
| -A | DAC(관리자 전용 연결)를 사용하여 SQL Server에 로그인합니다. 이 연결 유형은 서버 문제를 해결하는 데 사용됩니다. 이 연결은 DAC를 지원하는 서버 컴퓨터에만 사용할 수 있습니다. |
| -h *headers* | 컬럼 이름 사이에 출력할 행의 수를 지정합니다. 10을 지정하면 10행 마다 컬럼 이름을 표시합니다. 이 옵션은 sqlcmd 스크립팅 변수 SQLCMDHEADERS를 설정합니다. 머리글을 출력하지 않으려면 -1을 사용합니다. |
| -h *headers* | 컬럼 이름 사이에 출력할 행의 수를 지정합니다. 10을 지정하면 10행 마다 컬럼 이름을 표시합니다. 이 옵션은 sqlcmd 스크립팅 변수 SQLCMDHEADERS를 설정합니다. 머리글을 출력하지 않으려면 -1을 사용합니다. |
| -s *col_separator* | 열 구분 기호 문자를 지정합니다. 기본값은 공백입니다. 이 옵션은 sqlcmd 스크립팅 변수 SQLCMDCOLSEP을 설정합니다. 앰퍼샌드($)나 세미콜론(;)과 같이 운영 체제에서 특별한 의미를 갖는 문자를 사용하려면 해당 문자를 따옴표(")로 묶습니다. |
| -w *column_width* | 출력할 화면 너비를 지정합니다. 이 옵션은 sqlcmd 스크립팅 변수 SQLCMDCOLWIDTH를 설정합니다. 기본 너비는 80자입니다. |

| | |
|---|---|
| -m *error_level* | 오류 메시지 표시를 사용자 지정합니다. 지정한 심각도 이상의 오류에 대한 메시지 번호, 상태 및 오류 수준이 표시됩니다. |
| -V | sqlcmd 가 보고하는 가장 낮은 심각도를 지정합니다. Transact-SQL 스크립트에서 오류가 발생할 경우 심각도가 -V 스위치로 지정한 값보다 크거나 같은 경우에만 심각도가 보고됩니다. |
| -? | sqlcmd 옵션의 구문 요약 정보를 표시합니다. |

〈sqlcmd 유틸리티 주요 명령줄 옵션(스위치)〉

3) 다음은 sqlcmd에서 Transact-SQL 문 외에도 사용할 수 있는 명령입니다. 다음 명령어는 GO를 제외하고는 명령어 접두사로 콜론(:)을 붙여야 합니다. 단, osql 유틸리티와의 호환성으로 [:]으로 표시된 명령어는 생략할 수 있습니다.

| 명 령 | 설 명 |
|---|---|
| GO [count] | GO는 일괄 처리의 끝을 알려 주고 캐시된 Transact-SQL 문을 실행하도록 신호를 보냅니다. *count* 값을 지정하면 캐시된 문이 *count*에 지정한 횟수만큼 단일 일괄 처리로 실행됩니다. |
| :Connect *server_name*[₩*instance_name*] [-l *timeout*] [-U *user_name* [-P *password*]] | SQL Server 인스턴스에 연결합니다. |
| [:]ED | 텍스트 편집기를 시작합니다. Oracle sqlplus의 ED 명령과 유사합니다. 텍스트 편집기는 SQLCMDEDITOR 환경 변수에 의해 정의됩니다. |

| | |
|---|---|
| :Error<br>〈*filename*〉|STDERR|STDOUT | *filename*에 지정한 파일, stderr 또는 stdout으로 모든 오류 출력을 리디렉션합니다. 스크립트에서 오류 옵션이 여러 번 나타날 수 있습니다. 기본적으로 오류 출력은 stderr로 보내집니다. |
| [:]Exit [(query)] | 쿼리가 지정된 경우에 쿼리를 포함한 일괄 처리를 실행하며 쿼리 결과를 반환한 다음 종료합니다. 쿼리가 생략된 경우는 sqlcmd를 종료합니다. |
| :Help | sqlcmd 명령과 각 명령에 대한 간단한 설명을 표시합니다. |
| :List | 문 캐시 내용을 출력합니다. |
| :ListVar | 현재 설정되어 있는 스크립팅 변수의 목록을 표시합니다. |
| :On Error<br>[ exit\| ignore] | 스크립트나 일괄 처리를 실행하는 동안 오류가 발생할 때 수행할 작업을 설정합니다. |
| :Out<br>〈*filename*〉|STDERR|STDOUT | *filename*에 지정한 파일, stderr 또는 stdout으로 모든 쿼리 결과를 리디렉션합니다. 기본적으로 출력은 stdout으로 보내집니다. |
| :Perftrace<br>〈*filename*〉|STDERR|STDOUT | *filename*에 지정한 파일, stderr 또는 stdout으로 모든 성능 추적 정보를 리디렉션합니다. 기본적으로 성능 추적 출력은 stdout으로 보내집니다. |
| [:]Quit | sqlcmd를 종료합니다. |
| :r 〈 *filename*〉 | 〈 *filename*〉에 지정한 파일에서 추가 Transact-SQL 문과 sqlcmd명령을 문 캐시로 구문 분석합니다. |
| [:]RESET | 문 캐시를 지웁니다. |

| | |
|---|---|
| :ServerList | 로컬로 구성된 서버와 네트워크상에서 브로드캐스팅 하는 서버의 이름을 표시합니다. |
| :SetVar 〈var〉 [ "*value*" ] | sqlcmd 스크립팅 변수를 정의합니다. 스크립팅 변수의 형식은 다음과 같습니다. $(VARNAME) |
| :XML {ON\|OFF} | XML 출력이 예상되는 경우 사용합니다. |
| [:]!! | 운영 체제 명령을 실행합니다. |

〈sqlcmd 유틸리티 명령〉

4) 다음은 sqlcmd에서 사용하는 주요 스크립팅 변수입니다. sqlcmd 스크립트에서 사용할 수 있는 변수를 스크립팅 변수라고 합니다. 스크립팅 변수를 사용하면 하나의 스크립트를 다양한 시나리오에서 융통성 있게 사용할 수 있습니다. 예를 들어 하나의 스크립트를 여러 서버에서 실행해야 하는 경우 각 서버에 맞게 스크립트를 수정하는 대신 서버 이름에 스크립팅 변수를 사용할 수 있습니다. 스크립팅 변수는 :setvar 명령을 사용하여 명시적으로 정의하거나 〈sqlcmd 유틸리티 주요 명령줄 옵션(스위치)〉의 표에 나와있는 sqlcmd 스위치를 통해서 암시적으로 정의할 수 있습니다.

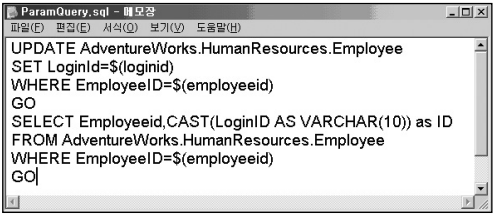
| 변 수 | 관련 스위치 | 설 명 |
|---|---|---|
| SQLCMDUSER | -U | 사용자 로그인 ID를 지정합니다. |
| SQLCMDPASSWORD | -P | 사용자 패스워드를 지정합니다. |
| SQLCMDSERVER | -S | 연결할 SQL Server 인스턴스를 지정합니다. |
| SQLCMDWORKSTATION | -H | 워크스테이션 이름을 지정합니다. |
| SQLCMDDBNAME | -d | 초기 데이터베이스를 지정합니다. |
| SQLCMDLOGINTIMEOUT | -l | sqlcmd 로그인 제한 시간(초)을 지정합니다. |
| SQLCMDHEADERS | -h | 컬럼 이름 사이에 출력할 행의 수를 지정합니다. |
| SQLCMDCOLSEP | -s | 열 구분 기호 문자를 지정합니다. |

| | | |
|---|---|---|
| SQLCMDCOLWIDTH | -w | 출력할 화면 너비를 지정합니다. |
| SQLCMDPACKETSIZE | -a | 패킷 크기를 지정합니다. |
| SQLCMDERRORLEVEL | -m | 오류 메시지 표시를 사용자 지정합니다. |

〈sqlcmd 스크립팅 변수〉

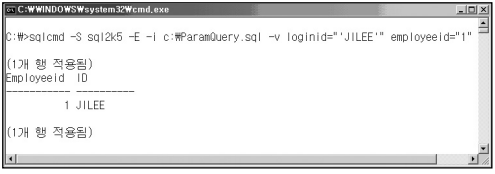
5) sqlcmd 유틸리티를 사용하여 변수를 사용하는 쿼리 수행하기

먼저 매개 변수를 사용하는 쿼리를 다음과 같이 파일에 저장합니다.




sqlcmd 유틸리티에서 매개 변수를 사용하는 방법은 앞에서 살펴본 바와 같이 환경 변수나 :setvar 명령이나 -v 스위치를 사용한 스크립팅 변수를 사용합니다.

다음은 -v 스위치를 사용하는 경우입니다.



## 보안 관리

### ■ 권한 관리(GRANT/DENY/REVOKE) 구문

권한 관리는 효율성을 위해서 개별 사용자에게 직접 권한을 부여하기 보다는 그룹이나 역할 등 사용자의 집합을 대상으로 하는 것이 효율적입니다. 이 때 사용자가 수행할 수 있는 권한은 사용자가 개별적으로 부여 받은 권한과 사용자가 소속된 역할에게 부여된 권한을 모두 합친 권합니다. SQL Server는 사용자나 역할에 권한을 부여하는 DCL(Data Control Language)에서 DENY 구문을 지원합니다. DENY는 기존에 GRANT 명령으로 부여 받은 권한에 우선하여 해당 권한의 사용을 명시적으로 금지합니다. 따라서, DENY 구문을 지원하지 않는 Oracle 보다 훨씬 효율적으로 권한을 관리할 수 있습니다.

| 권한 구문 | SQL Server | Oracle |
|---|---|---|
| GRANT | 권한 부여 | 권한 부여 |
| DENY | 권한의 명시적 금지 | 없 음 |
| REVOKE | 권한 부여 또는 금지로부터 해제 | 권한 해제 |

SQL Server는 본 모듈의 [아키텍처 이해 및 최대 용량 사양] 부분에서 살펴본 바와 같이 인스턴스(서버)내에 여러 개의 데이터베이스를 서비스하게 됩니다. 따라서, 계정이나 역할도 인스턴스(서버) 수준과 각 데이터베이스 수준으로 별도로 존재하게 됩니다. 서버 수준의 계정을 로그인 계정이라고 하고 데이터베이스 수준의 계정을 데이터베이스 사용자 계정이라고 합니다. 사용자는 로그인 계정을 사용해서 SQL Server에 로그인 한 다음에 사용자 계정이 생성되어 액세스가 허용되는 데이터베이스에만 접속할 수 있습니다. 역할도 서버 역할과 데이터베이스 역할로 구분됩니다.

■ SQL Server 로그인 계정 및 사용자 계정

1) 로그인 계정

SQL Server 2005 로그인 계정은 기본적으로 Windows 로그인 계정과 SQL Server 로그인 계정으로 구분됩니다. Windows 계정은 운영체제에 로그온하는 계정이나 Windows 그룹을 하나의 로그인 계정으로 처리하며 SQL Server 로그인 계정은 SQL Server가 자체적으로 아이디와 패스워드를 부여하여 처리하는 계정입니다. SQL Server 로그인 계정은 Service Broker에서 사용하기 위해서 인증서나 비대칭 키로 생성할 수 도 있습니다. SQL Server 2005의 SQL Server 로그인 계정은 Windows Server의 암호 정책과 암호 만료 정책을 적용할 수 있습니다.

다음은 Windows 로그인 계정과 SQL Server 로그인 계정을 생성하는 구문입니다.

**Windows 로그인 계정 생성 구문**

```
CREATE LOGIN login_name FROM WINDOWS
WITH DEFAULT_DATABASE=database_name
```

**SQL Server 로그인 계정 생성 구문**

```
CREATE LOGIN login_name WITH PASSWORD=' password'  [MUST_CHANGE]
[, DEFAULT_DATABASE=database_name ]
[, CHECK_EXPIRATION = {ON | OFF} ]
[, CHECK_POLICY = {ON | OFF} ]
[, CREDENTIAL = credential_name ]
```

2) 사용자 계정 (데이터베이스)
다음은 데이터베이스 사용자 계정을 생성하는 구문입니다.

**데이터베이스 사용자 계정 생성 구문**

```
CREATE USER user_name  {FOR | FROM} { LOGIN login_name
                                    | CERTIFICATE cert_name
                                    | ASYMMERIC KEY asym_key_name }
                               | WITHOUT LOGIN
WITH DEFAULT_SCHEMA = schema_name
```

로그인 계정 생성 시 인증서나 비대칭 키를 사용하여 생성할 수 있는 것과 같이
데이터베이스 사용자 계정을 생성할 때에도 인증서나 비대칭 키를 사용할 수 있습니다.

### ■ SQL Server 역할 및 역할 관리

SQL Server는 관리의 효율성을 위해서 이미 지정된 권한을 부여받은 시스템 역할을 제공합니다.
시스템 역할은 삭제하거나 권한을 변경할 수 없습니다.

1) SQL Server 고정된 서버 역할

| 역 할 | 설 명 |
|---|---|
| sysdamin | GRANT 옵션을 사용하여 허가된 사용 권한: CONTROL SERVER (서버/데이터베이스 수준 모든 권한) |
| serveradmin | 허가된 사용 권한: ALTER ANY ENDPOINT, ALTER RESOURCES, ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN, VIEW SERVER STATE |
| setupadmin | 허가된 사용 권한: ALTER ANY LINKED SERVER |

| | |
|---|---|
| securityadmin | 허가된 사용 권한: ALTER ANY LOGIN |
| processadmin | 허가된 사용 권한: ALTER ANY CONNECTION, ALTER SERVER STATE |
| dbcreator | 허가된 사용 권한: CREATE DATABASE |
| diskadmin | 허가된 사용 권한: ALTER RESOURCES |
| bulkadmin | 허가된 사용 권한: ADMINISTER BULK OPERATIONS |

2) SQL Server 고정된 데이터베이스 역할

| 역 할 | 설 명 |
|---|---|
| db_owner | GRANT 옵션을 사용하여 허가된 사용 권한: CONTROL (DROP DATABASE / RESTORE DATABASE 제외한 모든 권한) |
| PUBLIC | 해당 데이터베이스의 모든 사용자가 포함되는 역할 |
| db_accessadmin | 허가된 사용 권한: ALTER ANY USER, CREATE SCHEMA GRANT 옵션을 사용하여 허가된 사용 권한: CONNECT |
| db_datareader | 허가된 사용 권한: SELECT |
| db_datawriter | 허가된 사용 권한: DELETE, INSERT, UPDATE |
| db_ddlamin | 허가된 사용 권한: ALTER ANY ASSEMBLY, ALTER ANY ASYMMETRIC KEY, ALTER ANY CERTIFICATE, ALTER ANY CONTRACT, ALTER ANY DATABASE DDL TRIGGER, ALTER ANY DATABASE EVENT, NOTIFICATION, ALTER ANY DATASPACE, ALTER ANY FULLTEXT CATALOG, ALTER ANY MESSAGE TYPE, ALTER ANY REMOTE SERVICE BINDING, ALTER ANY ROUTE, ALTER ANY SCHEMA, ALTER ANY SERVICE, ALTER ANY SYMMETRIC KEY, |

| | |
|---|---|
| | CHECKPOINT, CREATE AGGREGATE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE QUEUE, CREATE RULE, CREATE SYNONYM, CREATE TABLE, CREATE TYPE, CREATE VIEW, CREATE XML SCHEMA COLLECTION, REFERENCES |
| db_securityadmin | 허가된 사용 권한: ALTER ANY APPLICATION ROLE, ALTER ANY ROLE, CREATE SCHEMA, VIEW DEFINITION |
| db_backupoperator | 허가된 사용 권한: BACKUP DATABASE, BACKUP LOG, CHECKPOINT |
| db_denydatareader | 거부된 사용 권한: SELECT |
| db_denydatawriter | 거부된 사용 권한: DELETE, INSERT, UPDATE |

SQL Server는 고정된 데이터베이스 역할 외에도 사용자가 직접 데이터베이스 역할을 생성하고 해당 역할에 적절한 권한을 주어서 권한을 효율적으로 관리할 수 있습니다.
사용자 정의 역할은 사용자 정의 데이터베이스 역할과 응용 프로그램 역할로 구분됩니다.

**3) 사용자 정의 데이터베이스 역할**

사용자 정의 데이터베이스 역할은 고정된 데이터베이스 역할과 유사한 역할로 동일한 권한을 갖는 사용자들을 그룹화한 개념입니다. 관리자는 사용자 정의 데이터베이스 역할을 생성하고 적절한 권한을 부여한 뒤 각 데이터베이스 사용자를 해당 데이터베이스 역할에 포함하거나 제거할 수 있습니다. 다음은 사용자 정의 데이터베이스 역할을 관리하는 구문입니다.

```
CREATE ROLE role_name [AUTHORIZATION owner_name]
ALTER ROLE role_name WITH NAME= new_name
DROP ROLE role_name
```

사용자 정의 데이터베이스 역할을 삭제하는 경우 먼저 해당 역할의 멤버를 제거해야 합니다.

#### 4) 응용 프로그램 역할

응용 프로그램 역할은 응용 프로그램이 사용자와 같이 자체 사용 권한으로 실행할 수 있도록 설정하기 위해서 사용합니다. 응용 프로그램 역할을 사용하면 특정 데이터에 대한 액세스를 특정 응용 프로그램을 통해 연결하는 사용자로만 허용할 수 있습니다. 데이터베이스 역할과는 달리 응용 프로그램 역할은 멤버를 추가하거나 삭제할 수 없고 sp_setapprole 시스템 저장 프로시저를 지정된 암호로 실행해서 활성화합니다.

다음은 응용 프로그램 역할을 관리하는 구문입니다.

```
CREATE APPLICATION ROLE application_role_name
WITH PASSWORD =' password'  [, DEFAULT_SCHEMA=schema_name]

ALTER APPLICATION ROLE application_role_name
WITH NAME= new_application_name | PASSWORD ='  new_password'
 | DEFAULT_SCHEMA= schema_name

DROP APPLICATION ROLE application_role_name

EXEC sp_setapprole  'role_name' ,  'password'
```

#### 5) 역할의 권한 및 멤버 관리

SQL Server에서 권한의 부여, 거부 및 해제 등은 GRANT/DENY/REVOKE 구문을 사용하며 역할에 사용자 계정을 추가하거나 제거하는 작업은 다음과 같이 sp_addrolemember / sp_droprolemember 시스템 저장 프로시저를 사용해서 처리합니다.

```
EXEC sp_addrolemember  'role_name' ,  'user_account'
GO
EXEC sp_droprolemember  'role_name' ,  'user_account'
```

### ■ 모듈 실행 보안 컨텍스트

SQL Server 2005에서는 실행 보안 컨텍스트는 EXECUTE AS 구문을 실행하거나 모듈에 EXECUTE AS 절을 지정하여 다른 사용자 또는 로그인으로 전환할 수 있습니다.
보안 컨텍스트가 전환된 후 SQL Server에서 EXECUTE AS 구문 또는 모듈을 호출하는 사용자대신 해당 계정에 대한 로그인 및 사용자의 권한을 확인합니다

① 암시적 모듈 실행 보안 컨텍스트 전환
저장 프로시저, 트리거, 큐 또는 사용자 정의 함수와 같은 모듈의 실행 보안 컨텍스트는 모듈 정의에 EXECUTE AS 절에 사용자 또는 로그인 이름을 지정하여 암시적으로 변경할 수 있습니다. Oracle의 AUTHID 절과 유사합니다.

| SQL Server | Oracle | 설 명 |
|---|---|---|
| EXECUTE AS CALLER | AUTHID CURRENT_USER | 모듈 내부의 구문이 모듈 호출자의 보안 컨텍스트에서 실행되도록 지정 |
| EXECUTE AS SELF | | 모듈을 만들거나 수정하는 사용자의 보안 컨텍스트에서 실행되도록 지정 |
| EXECUTE AS OWNER | AUTHID DEFINER | 모듈 내부의 구문이 모듈 소유자의 보안 컨텍스트에서 실행되도록 지정 |
| EXECUTE AS 'user_name' | | 모듈 내부의 문이 user_name에 지정된 사용자의 보안 컨텍스트에서 실행 되도록 지정 |

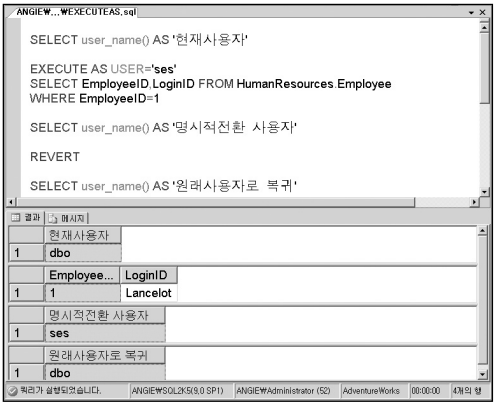
다음은 호출자의 실행 보안 컨텍스트에서 실행되는 저장 프로시저의 예입니다.

```
CREATE PROC usp_UpEmpLogin
@LoginID varchar(10)
,@Employeeid int
WITH EXECUTE AS CALLER
AS
UPDATE AdventureWorks,HumanResources,Employee
SET LoginId=@LoginID
WHERE EmployeeID=@Employeeid
GO
```

② 명시적 모듈 실행 보안 컨텍스트 전환

세션 또는 모듈의 실행 컨텍스트는 EXECUTE AS USER 구문에 사용자 또는 로그인 이름을 지정해서 명시적으로 변경할 수 있습니다. REVERT 구문을 사용해서 이전 보안 컨텍스트로 돌아갑니다.

다음은 dbo 사용자로 데이터베이스에 연결해서 ses 로 실행 보안 컨텍스트를 변경하여 쿼리를 실행하고 다시 이전 사용자인 dbo로 실행 보안 컨텍스트를 변경하는 예입니다.



■ **데이터 암호화**

SQL Server 2005는 별도의 추가 구매 없이 다양한 방법으로 데이터를 암호화하고 복호화하는 기능을 제공합니다.

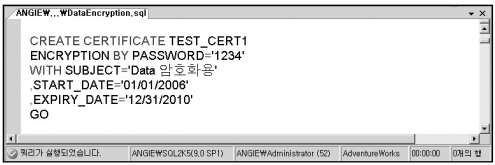
SQL Server는 설치하는 동안에 Service Master Key를 자동으로 생성하며, Windows 운영체제에서 기본적으로 제공하는 Data Protection API로 암호화 되어 SQL Server서비스 계정의 자격 증명으로 사용됩니다. Service Master Key는 연결된 서버 또는 연결 문자열 등과 같은 인스턴스 수준 설정에 사용되거나 데이터베이스 마스터 키를 암호화 하는 데 사용할 수 있습니다. 각 데이터베이스 마스터 키는 인증서와 비대칭 키를 암호화하는 데 사용됩니다.
인증서와 비대칭 키는 로그인 계정이나 데이터베이스 계정 생성을 위해서나 데이터를 암호화하는 데 사용되거나 대칭 키를 암호화할 수 있으며 인증서는 비대칭 키를 암호화할 수 있습니다.

다음은 SQL Server 2005가 제공하는 데이터 암호화 방법입니다.

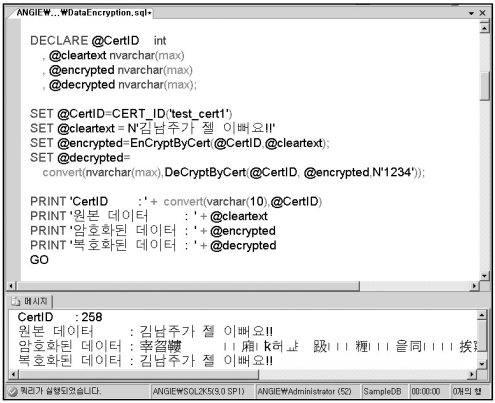
| 데이터 암호화 방법 | 해당 암호화 함수 |
|---|---|
| 인증서 | EncryptByCert, DecryptByCert |
| 비대칭 키 | EncryptByAsmKey, DecryptByAsmKey |
| 대칭 키 | EncryptByKey, DecryptByKey |
| 패스워드 구문 | EncryptByPassPhrase, DecryptByPassPhrase |

### 1) 인증서를 사용한 데이터 암호화 / 복호화

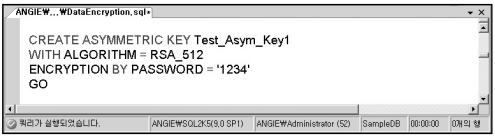
SQL Server 2005는 X.509 표준을 따르고 X.509 V1 필드를 지원하는 인증서를 만들수 있는 인증서 서버 기능을 제공합니다. 다음과 같이 먼저 인증서를 생성합니다.



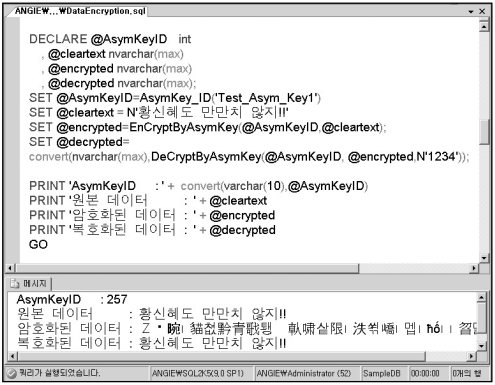

생성된 인증서와 EncryptByCert, DecryptByCert 함수를 사용해서 데이터를 암호화 / 복호화합니다.



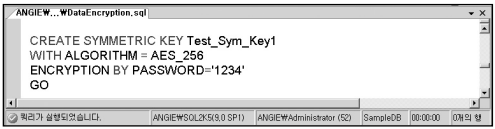
### 2) 비대칭 키를 사용한 데이터 암호화 / 복호화




먼저 RSA_512 알고리즘 등을 사용하여 비대칭 키를 생성하고 생성한 비대칭 키로 다음과 같이 EncryptByAsmKey, DecryptByAsmKey 함수를 사용해서 암호화 / 복호화합니다.

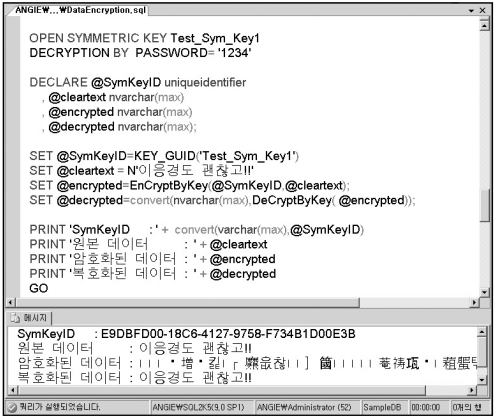


### 3) 대칭키를 사용한 데이터 암호화 / 복호화



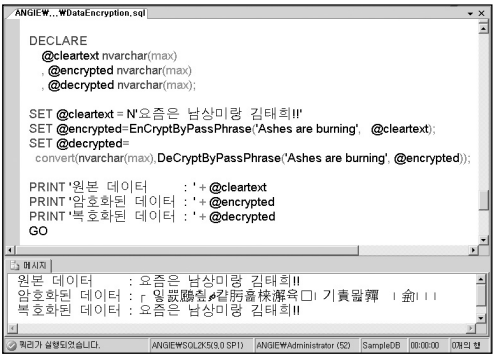

AES_256 등의 알고리즘을 사용해서 대칭 키를 생성한 뒤 생성한 대칭 키로 다음과 같이 암호화 / 복호화합니다.

대칭 키를 사용하기 위해서는 대칭키를 암호화 하는데 사용된 인증서나 비대칭 키 또는 다른 대칭 키, 패스워드 등을 지정해서 다음과 같이 OPEN SYSMMETRIC KEY 구문으로 해독한 뒤 EncryptByKey, DecryptByKey 함수를 사용해서 암호화 / 복호화합니다.



4) 패스워드 구문을 사용한 데이터 암호화 / 복호화

다음과 같이 인증서나 비대칭 키, 대칭 키 등이 없이 임시적으로 암호화하거나 복호화할 때 EncryptByPassPhrase, DecryptByPassPhrase 함수를 통해서 데이터를 암호화/ 복호화할 수 있습니다.




## 관리 작업 자동화

Oracle의 경우 관리 작업의 자동화를 위해서 DBMS_JOB 패키지 또는 DBMS_SCHEDULER 패키지를 사용하거나, OEM(Oracle Enterprise Manager)의 작업 생성 기능을 통해서, 또는 운영 체제의 쉘 스크립트 등을 통하는 경우를 볼 수 있습니다. 필자는 가능한 운영 체제의 스크립트를 통한 자동화를 권장하지 않습니다. 그 이유는 쉘 스크립트와 같이 운영 체제에 스크립트를 통한 자동화는 관리의 이중화로 비 효율적일 뿐만 아니라 책임 소재도 불명확하게 됩니다. 반면 DBMS에 작성된 자동화 작업은 DBA에 의해서 효율적이며 중앙 집중적인 관리가 가능합니다. 따라서, Oracle의 자동화 작업은 SQL Server의 에이전트 서비스에 작업으로 등록하여 실행하고 관리하는 것을 권장합니다.

굳이 UNIX 쉘 스크립트로 작성된 자동화 작업을 동일한 형태로 옮겨야 합니다면 Windows Script Host를 사용할 수 있습니다만 SQL Server 에이전트 서비스에서 운영체제 명령이나 ActiveX 스크립트 작업으로 동일하게 수행할 수 있습니다.

### ■ 자동화 작업 생성 및 전자메일을 이용한 작업결과 통보하기

#### 1) 자동화 작업을 위한 프록시 설정

SQL Server 2005는 이전 버전에 비해서 엄청나게 보안을 강화하였습니다. 특히 자동화 작업을 위해서 각 작업 마다 프록시를 설정할 수 있습니다. 프록시란 SQL Server 에이전트 작업 단계의 보안 컨텍스트에 대한 정의로 Windows Server와 상호 작용을 하는 경우에 지정하게 됩니다. 이전 버전에서는 sysadmin 고정 서버 역할의 멤버는 SQL Server 에이전트 서비스의 로그온 계정을 통해서 sysadmin 고정 서버 역할의 멤버가 아닌 계정은 별도로 프록시 계정을 하나만 지정하여 운영 체제와 상호 작용하는 작업을 수행하였습니다. SQL Server 2005는 수행할 작업을 위해서 필요한 운영 체제 권한에 따라서 각각의 작업 별로 적절한 Windows 계정과 연결된 자격 증명(credential)을 생성하고, 이 자격 증명과 연결된 프록시를 생성합니다. 프록시는 작업의 성격에 따라서 ActiveX 스크립트, 운영 체제(CmdExec), 복제 배포자, 복제 병합, 복제 큐 판독기, 복제 스냅숏, 복제 트랜잭션 로그 판독기, SQL Server Analysis Services 명령, SQL Server Analysis Services 쿼리, SQL Server Integration Services 패키지로 구분됩니다. SQL Server 에이전트 작업에서 해당 프록시를 사용하면 SQL Server 에이전트는 작업 수행 시 해당 프록시에 정의되어 있는 자격 증명을 가장한 다음 해당 보안 컨텍스트를 사용하여 각 작업 단계를 수행합니다.
프록시를 설정하기 위해서는 다음 단계를 수행합니다.

① 수행하고자 하는 작업에 적절한 권한을 갖는 Windows 계정을 선택하고 이 계정과 연결되는 자격 증명을 다음과 같이 생성합니다.

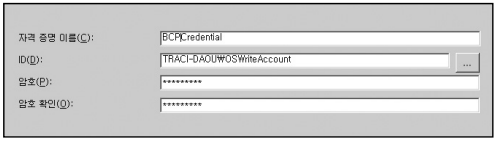
다음 예는 BCPCredential이라는 자격 증명을 만듭니다.

이 자격 증명에는 TRACI-DAOU\OSWriteAccount라는 Windows 사용자 계정을 사용하며 해당 계정의 암호인 !@#$qwer를 지정합니다.

T-SQL :

```
CREATE CREDENTIAL BCPCredential
WITH IDENTITY = 'TRACI-DAOU\OSWriteAccount' ,
   SECRET = 'I@#$qwer' ;
GO
```

SQL Server Management Studio :
개체 탐색기→ 서버→ 보안→ 자격 증명→ 새 자격 증명

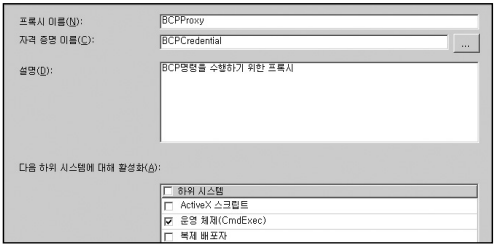



② 해당 자격 증명을 사용하는 프록시를 다음과 같이 생성합니다.

T-SQL :

```
USE msdb ;
GO
EXEC dbo.sp_add_proxy @proxy_name = 'BCPproxy' ,
   @enabled = 1,
   @description = 'BCP 명령을 수행하기 위한 프록시' ,
   @credential_name = 'BCPCredential' ;
GO
```

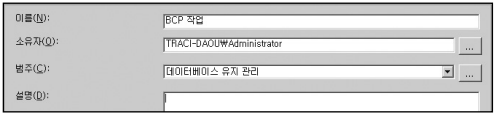
SQL Server Management Studio :
개체 탐색기→ 서버→ SQL Server 에이전트→ 프록시→ 새 프록시



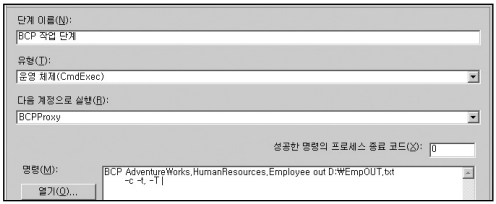

### 3) 작업 생성 하기

다음은 BCP 명령줄 도구를 사용해서 HumanResources.Employee 테이블의 테이터를 플랫 파일 형태로 추출하는 작업을 생성합니다.

개체 탐색기→ 서버→ SQL Server 에이전트→ 작업→ 새 작업
일반 탭에서 다음과 같이 입력합니다.



단계 탭을 선택하고 [새로 만들기] 버튼을 클릭해서 [새 작업 단계] 창을 실행하고 [유형(T):] 부분에는 [운영체제(CmdExec)]를 선택합니다. [다음계정을 실행(R):]부분에는 [BCPProxy]를 입력하고 [명령(M):]부분에는 다음과 같이 BCP 수행구문을 입력합니다.




**BCP 수행 구문:**

BCP AdventureWorks.HumanKesources.Employee OUT D:\EmpOUT.txt-c-t,-T

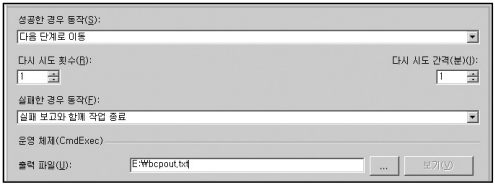
**[주의]**

예제의 BCP 구문은 Windows 인증을 사용하여 BCP 작업을 수행하므로 BCPProxy에 연결된 [TRACI-DAOU\OSWriteAccount] 운영 체제 계정에 대해서 SQL Server로그인 계정이 미리 생성된 상태라야 하며 이 로그인 계정은 AdventureWorks, Human Resouces.Emploee 테이블에 SELECT 권한을 가지고 있어야 합니다. 또한, [TRACI-DAOU\OSWriteAccount] 운영 체제 계정은 D-드라이브에 쓰기 권한을 가지고 있어야 합니다.

***[참고]***

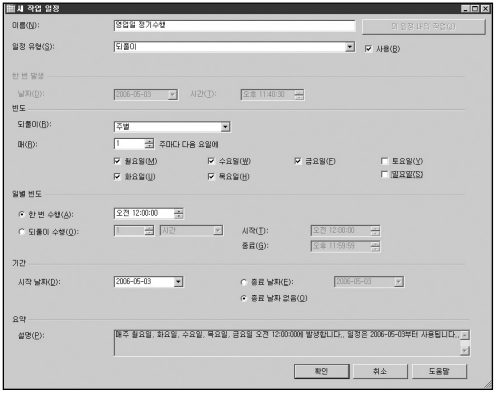
*BCP 유틸리티의 자세한 사항은 [모듈5 Oracle 기반 데이터 인터페이스 마이그레이션] 부분을 참조하시기 바랍니다.*

이어서 고급 탭에서 다음과 같이 작업 실패 시 다시 시도 횟수 및 다시 시도 간격 그리고 출력 파일 설정을 합니다.




#### 4) 작업 일정 설정 하기

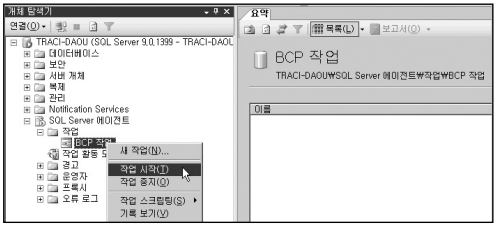
일정 탭에서 다음과 같이 월요일부터 금요일까지 오전 12시에 수행하는 일정을 생성합니다.




확인 버튼을 누르고 다시 한번 확인 버튼을 눌러서 새 작업 생성 작업을 완료합니다.

### 5) 작업 수행 결과 확인 하기

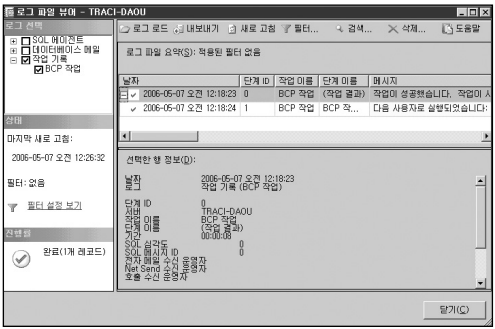
이제 작업을 수행해보고 작업이 성공했는지 여부와 출력 파일 등을 확인할 차례입니다. 다음과 같이 서버→ SQL Server 에이전트→ 작업→ BCP 작업으로 이동해서 오른 쪽 버튼을 누르고 작업 시작을 눌러서 작업을 실행합니다.




작업의 실행 결과는 다음과 같이 해당 작업의 [기록 보기(V)]를 눌러서 확인할 수 있습니다.


물론, 작업의 실행은 sp_start_job 시스템 저장 프로시저로 수행할 수 있고, 실행 결과 확인은 sp_help_jobhistory 시스템 저장 프로시저로 확인할 수 있습니다.

다음은 작업의 결과를 확인하는 로그 파일 뷰어로 작업의 결과 뿐만 아니라 SQL 에이전트 로그와 데이터베이스 메일 로그도 확인할 수 있는 통합 인터페이스를 제공합니다.



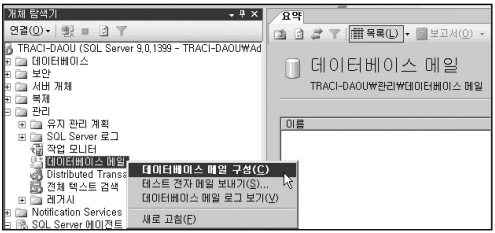

### 6) 작업 결과 통보 설정 하기

이제 생성한 작업의 결과를 메일 등을 통해서 관리자가 통보 받는 기능에 대해서 살펴보겠습니다. 작업의 결과를 전자 메일로 통보 받기 위해서는 먼저 데이터베이스 메일을 구성해야 합니다. 또한 작업 결과를 통보 받을 대상인 운영자를 SQL Server 에이전트에서 생성해야 합니다.

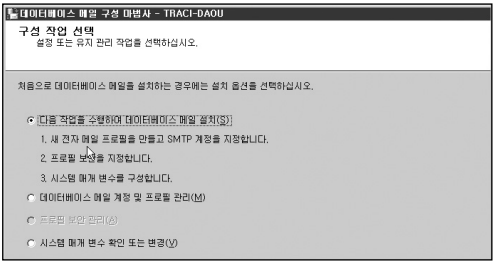
① 데이터베이스 메일 설정하기

SQL Server 2005 데이터베이스 메일은 이전 버전의 SQL Mail과 SQL 에이전트 메일을 통합하고 강화한 기능입니다. 이전 버전은 전자 메일 클라이언트인 OUTLOOK의 프로필을 단 하나만 사용하여 MAPI프로토콜을 사용해서 메일을 전송하였으나 데이터베이스 메일은 SMTP를 사용해서 메일을 전송하며 작업 내용에 따라서 서로 다른 메일 프로필을 설정할 수 있으며, 메일 프로필에 대한 장애조치도 지원할 뿐만 아니라 SMTP를 사용하므로 상용 SMTP 서버 및 무료 버전인 Windows SMTP 등을 지원합니다. 다음은 데이터베이스 메일을 설정하는 방법입니다.

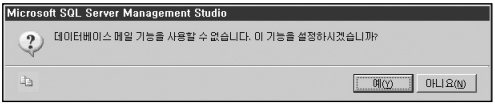
개체 탐색기→ 서버→ 관리→ 데이터베이스 메일→ [데이터베이스 메일 구성(C)]를 선택합니다.




데이터베이스 메일 구성 마법사가 실행되면 [다음(N)] 버튼을 눌러서 [구성 작업 선택]페이지로 이동합니다. [다음 작업을 수행하여 데이터베이스 메일 설치]가 기본으로 선택되었음을 확인하고 [다음(N)] 버튼을 누릅니다.

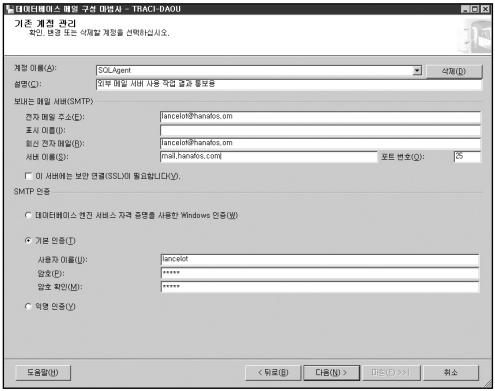


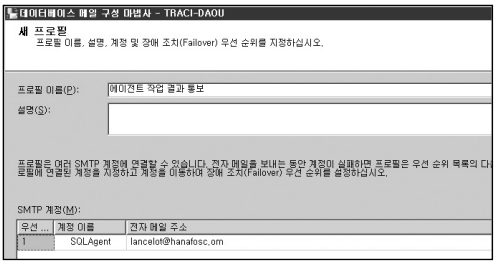

데이터베이스 메일 기능 설정 여부를 묻는 대화창이 나타나면 [예(Y)] 버튼을 누릅니다.

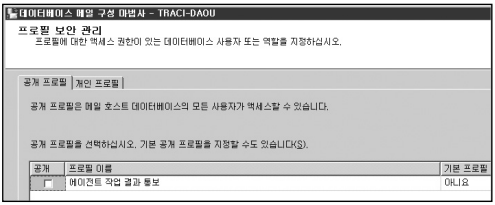


이와 같은 데이터베이스 메일 기능 활성화는 SQL Server 노출 영역 구성 도구나 sp_configure 시스템 저장 프로시저로 Database Mail XPs 서버 옵션 활성화를 통해서도 설정할 수 있습니다.

새 프로필 페이지에서 [추가(A)] 버튼을 누르고 SMTP 메일 계정 설정을 합니다.



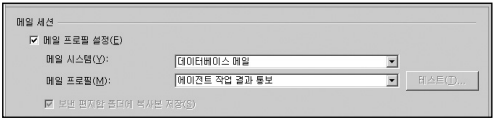






이어서, 시스템 매개 변수 구성 페이지에서 내용을 확인하고 [다음(N)] 버튼을 누르고 마법사 완료 페이지에서 [마침(F)] 버튼을 눌러서 다음과 같이 데이터베이스 메일 구성 작업을 완료합니다.

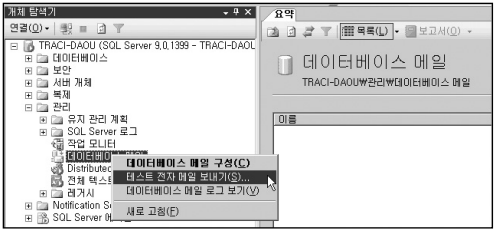

② SQL Server 에이전트 메일 프로필 설정

개체 탐색기→ 서버→ SQL Server 에이전트 속성→ 경고 시스템→ 메일 프로필 설정에서 메일 시스템 목록에서 데이터베이스 메일을 선택합니다. 다음과 같이 메일 프로필 목록에서 데이터베이스 메일에 대한 메일 프로필을 선택합니다. SQL Server 에이전트 서비스를 재 시작합니다.



***[참고]***
*[메일 프로필(M)] 리스트 박스 옆의 테스트 버튼은 이전 버전과의 호환성을 위한 SQLMail을 사용하는 경우에만 활성화 됩니다.*

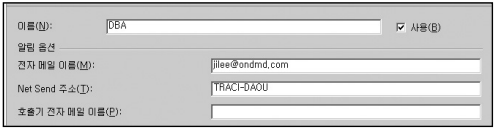
데이터베이스 메일의 테스트는 다음과 같이 데이터베이스 메일을 선택하고 오른 쪽 버튼의 메뉴 가운데 [테스트 전자 메일 보내기(S)] 메뉴를 선택하여 수행합니다.




③ 운영자 생성 하기

운영자란 SQL Server 에이전트의 작업이 완료되거나 경고가 발생할 때 전자 메일 알림을 받을 수 있는 사람이나 그룹의 별칭입니다. 따라서 자동화 작업의 수행 결과와 경고의 발생을 신속히 전자 메일을 통해서 관리자에게 알리기 위해서 운영자를 생성합니다.

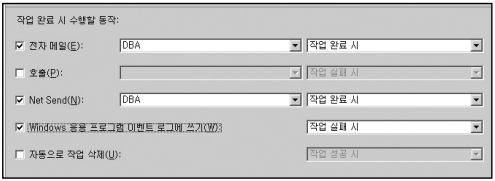
개체 탐색기→ 서버→ SQL Server 에이전트→ 운영자 오른쪽 버튼→ [새 운영자(N)]를 눌러서 다음과 같은 새 운영자 일반 페이지에서 이름 및 통보를 받을 전자 메일 주소, Net send 메시지를 받을 컴퓨터 명 또는 IP 주소를 입력하고 [확인] 버튼을 눌러서 운영자 생성을 완료합니다.



④ 작업 결과 통보 설정 하기

이전의 단계에서 데이터베이스 메일 설정, SQL Server 에이전트 메일 프로필 설정,운영자 생성 등을 완료하였습니다. 이와 같은 처리가 완료되었다면 작업 결과를 통보하도록 설정하는 작업은 작업 생성 단계에서 처리해도 무방합니다. 이미 생성된 작업의 작업 결과 통보는 다음의 단계를 수행합니다.

[3] 작업 생성 하기]섹션에서 생성한 [BCP 작업]의 [속성(R)]창을 열어서 [알림] 페이지를 선택합니다. 다음과 같이 [전자 메일(E):]과 [Net Send(N):] 부분에 이전 단계에서 생성한 운영자를 지정하고 통보 받을 조건을 [작업 완료 시], [작업 실패 시], [작업 성공 시] 가운데에서 지정합니다. 권장하는 사항은 [작업 완료 시]입니다. 지정된 시간에 운영자가 작업 결과를 통보 받지 못했다면 SQL Server, 네트워크, 메일 서버 등 회사 네트워크 내에 문제가 있음을 인지할 수 있기 때문입니다.

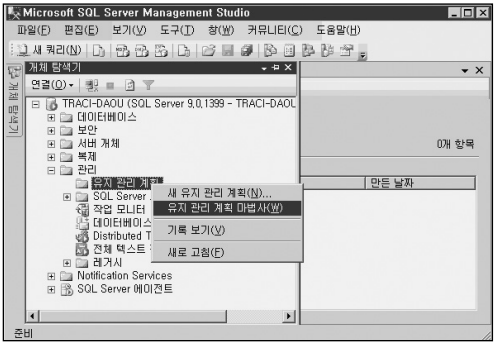



## ■ SQL Server 유지 관리 계획 마법사를 사용하여 자동화 작업 생성 하기

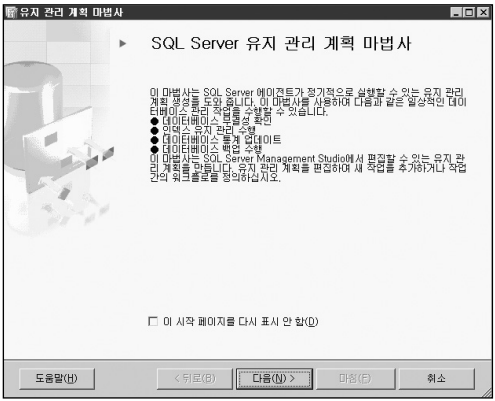
SQL Server는 데이터베이스 관리자가 일반적으로 수행하는 관리 작업을 쉽게 생성할 수 있도록 유지 관리 계획 마법사를 제공합니다. 따라서, 백업이나 통계 정보 갱신 등의 일반적인 관리 작업은 SQL Server 유지 관리 계획 마법사를 통해서 쉽고 빠르게 작업을 생성할 수 있습니다.

이제 SQL Server 관리자가 수행해야 하는 가장 중요한 관리업무 가운데 하나인 시스템 데이터베이스 백업 작업을 유지 관리 계획 마법사를 통해서 생성는 방법에 대해서 살펴봅니다. SQL Server의 시스템 데이터베이스는 SQL Server 인스턴스가 서비스하는 데 있어서 중요한 핵심 정보를 저장하고 있는 데이터베이스 입니다. 따라서, 정기적인 백업 작업을 수행해야 하며 서비스 팩이나 보안 패치 등을 설치하게 되면 즉시 백업을 수행해야 합니다. 각 시스템 데이터베이스의 기능은 이 모듈의 앞 부분에서 설명하였습니다.

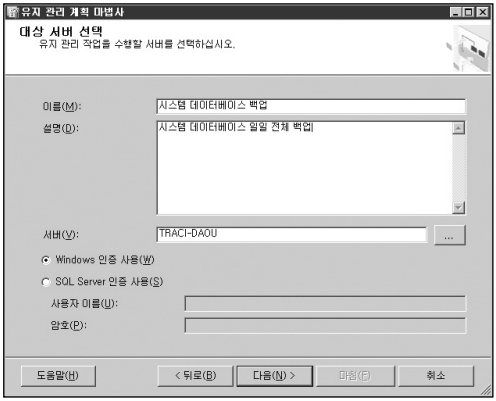
먼저 다음과 같이 개체 탐색기의 인스턴스를 확장하고 관리의 유지 관리 계획에서 오른쪽 버튼을 클릭하고 [유지 관리 계획 마법사(W)]를 선택합니다.



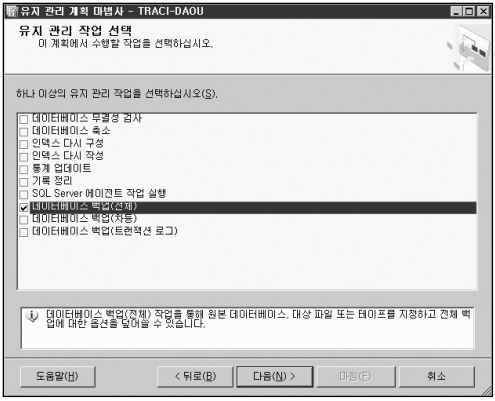

다음과 같이 SQL Server 유지 관리 계획 마법사가 실행되면 [다음(N)] 버튼을 누릅니다.



[대상 서버 선택] 페이지에서 다음과 같이 대상 서버를 선택하고 [이름(M)] 부분에 작업의 이름을 입력합니다. [설명(D)] 부분에는 작업의 생성 목적 등 설명을 입력합니다. [Windows 인증사용(W)] 부분이 선택되어져 있는 것을 확인하고 [다음(N)] 버튼을 누릅니다.

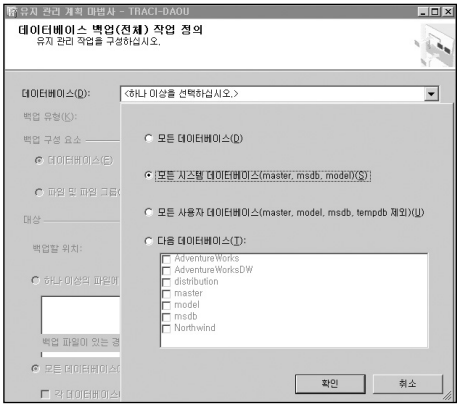



이제 [유지 관리 작업 선택] 페이지에서 [데이터베이스 백업(전체)]를 선택하고 [다음(N)] 버튼을 누릅니다. 유지 관리 계획 마법사를 통해서는 다음 그림에서 확인할 수 있는 것과 같이 데이터베이스 무결성 검사, 데이터베이스 축소, 인덱스 다시 구성, 인덱스 다시 작성, 통계 업데이트, 기록 정리, SQL Server 에이전트 작업 실행, 데이터베이스 백업(전체), 데이터베이스 백업(차등), 데이터베이스 백업(트랜잭션 로그) 작업을 생성하고 일정을 등록할 수 있습니다.

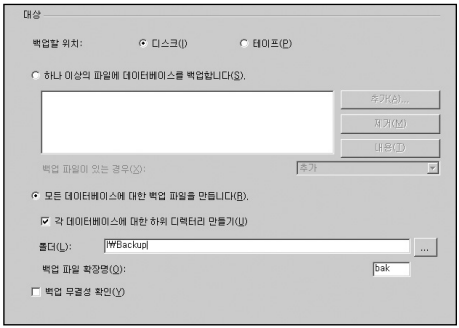



[유지 관리 작업 순서 선택] 페이지에서는 앞의 [유지 관리 작업 선택] 페이지에서 여러 개의 작업을 선택한 경우에 순서를 지정할 수 있습니다. [다음 (N)] 버튼을 눌러서 다음 페이지로 이동합니다.


[데이터베이스 백업(전체) 작업 정의] 페이지에서는 앞에서 선택한 작업에 대한 세부 항목을 지정합니다. 다음과 같이 [데이터베이스(D)]의 드롭다운 리스트 박스를 열어서 [모든 시스템 데이터베이스(master, msdb, model)(S)]를 선택하고 [확인] 버튼을 누릅니다.

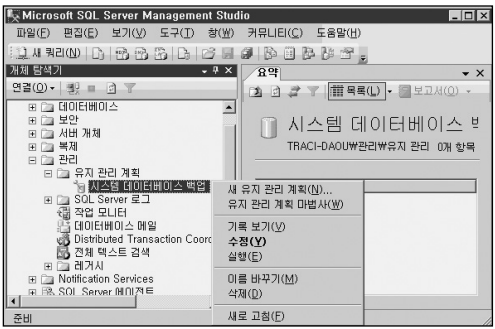



[대상] 섹션에서는 백업 대상 디바이스 및 파일의 확장 명 등을 지정합니다. 다음의 선택한 사항은 시스템 데이터베이스를 디스크에 백업하며 [폴더 (L)] 부분에 지정한 경로에 각 시스템 데이터베이스 별로 데이터베이스 이름과 동일한 폴더를 생성하고 해당 폴더에 .bak 확장자 파일로 백업을 수행하도록 지정하였습니다.



[다음 (N)] 버튼을 눌러서 다음과 같이 [계획 속성 선택] 페이지에서 시스템 데이터베이스 백업 작업의 일정을 등록합니다. [변경 (C)] 버튼을 눌러서 새로운 일정을 등록합니다. 일정의 등록 및 보고서 옵션 지정은 앞에서 살펴본 [자동화 작업 생성]의 [일정 등록 하기] 부분과 [작업 결과 통보 설정 하기] 부분을 참조하여 유지 관리 계획 마법사 작업을 완료합니다.


SQL Server 유지 관리 계획 마법사로 생성한 작업은 다음과 같이 해당 유지 관리 계획을 선택하고 오른쪽 버튼을 클릭해서 실행하거나 수정할 수 있습니다. 뿐만 아니라 [개체 탐색기]의 [SQL Server 에이전트]의 [작업]에서도 실행하거나 수정할 수 있습니다.



## 자주 사용하는 시스템 저장 프로시저 및 함수

| 시스템 저장 프로시저 / 함수 | 설 명 |
|---|---|
| sp_configure | 서버의 구성 옵션을 설정합니다. |
| sp_who2 | 서버에 연결된 현재 사용자와 프로세스 정보를 반환합니다. |
| sp_lock | 잠금에 대한 정보를 반환합니다. |
| sp_helpserver | 현재 인스턴스에 등록된 모든 서버에 관한 정보를 반환합니다. |
| sp_dropserver | 로컬 SQL Server 인스턴스, 원격 서버 및 연결된 서버를 제거합니다. |
| sp_addserver | 원격 서버나 로컬 SQL Server 인스턴스 이름을 정의합니다. |
| sp_serveroption | 원격 서버 및 연결된 서버용 서버 옵션을 설정합니다. |
| sp_helpsort | 서버의 정렬 순서와 문자 집합을 반환합니다. |
| sp_helpdb | 전체 또는 지정한 데이터베이스의 정보를 반환합니다. |
| sp_helpfilegroup | 현재 데이터베이스의 파일 그룹 정보를 반환합니다. |
| sp_helpfile | 현재 데이터베이스의 파일 정보를 반환합니다. |
| sp_help | 지정한 데이터베이스 개체에 대한 정보를 반환합니다. |
| sp_helpindex | 지정한 개체의 인덱스 정보를 반환합니다. |
| sp_helpstats | 지정한 개체의 통계 정보를 반환합니다. |
| sp_helpconstraint | 지정한 테이블의 제약 조건 정보를 반환합니다. |
| sp_helptrigger | 지정한 테이블의 DML 트리거 유형을 반환합니다. |
| sp_helplogins | 각 데이터베이스의 로그인 및 연관된 사용자에 관한 정보를 반환합니다. |
| sp_helpuser | 현재 데이터베이스에서 데이터베이스 수준의 보안 주체 정보를 반환합니다. |
| sp_spaceused | 데이터베이스 또는 지정한 개체의 행 수 및 디스크 공간 사용 정보를 반환합니다. |
| sp_tables | 현재 데이터베이스의 테이블과 뷰의 정보를 반환합니다. |
| sp_rename | 지정한 개체의 이름을 변경합니다. |
| suser_sname( ) | 지정된 로그인 이름을 반환합니다. |
| user_name( ) | 지정된 데이터베이스 사용자 이름을 반환합니다. |
| @@error | 최근 실행된 T-SQL 구문의 오류 여부를 반환합니다. |
| @@rowcount | 최근 실행된 문의 영향을 받은 행 수를 반환합니다. |
| @@trancount | 현재 연결에서 활성화된 트랜잭션 수를 반환합니다. |

## 비매품

# Oracle Migration Guide Book


- 저 자: 이 종 인
- 기 획: ㈜온디맨드
- Contents 관련문의: jilee@ondmd.com



- 발 행: 한국마이크로소프트(유)
- 제 작: ㈜온디맨드
- 초판 발행일: 2006년 8월



본 책에 실린 글과 그림, 사진 및 프로그래밍 코드등의 저작권 및 배포권은
한국마이크로소프트(유)와 ㈜온디맨드에 있으며, 저작권자의 동의 없이는 사용할 수 없습니다.

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