Microsoft SQL Server 2005

초보 DBA를 위한 SQL Server 2005 관리가이드

Microsoft

저자 약력

전현경 에이디 컨설팅 수석 컨설턴트

- SK 커뮤니케이션즈 싸이월드, 신세계닷컴, 인터파크지마켓, CGV, 삼성생명,
  연세 의료원, 아이템베이, 에이스 아메리칸 화재 해상 보험, 현대 자동차등 튜닝 컨설팅
- SQL Server 튜닝 소프트웨어 개발
- 옥션 DBA
- 인터넷 법률 사이트 전산팀장

감수자 약력

하성희 에이디 컨설팅 대표 컨설턴트 겸 SQL Server MVP

- SQL Server 컨설팅, 세미나 강의, 번역
- 옥션 DBA팀장
- 마이크로소프트 SQL Server 기술지원 엔지니어 겸 Data Access 팀장
- 포스데이타 SYBASE 기술 지원 엔지니어
- POSCO DBA

서문

SQL Server 2000에 이어 드디어 SQL Server 2005가 출시되었습니다.
제품이 업그레이드되면, 많은 좋은 기능이 생기고 성능 또한 좋아지기 때문에 새롭게 제품이 출시된다는 것은 좋은 일이라고 생각합니다.
그러나, 그에 따라 관리 방법이나 서버를 이해하는 포인트가 조금은 달라져야 하기 때문에, 이에 적응하기 위해 제품을 사용하는 사람이나 관리하는 사람들은 새로 출시된 제품에 대한 지식을 업그레이드 시키기 위해 할 일이 많아지고 분주해지는 것 또한 사실입니다.
SQL Server 2005도 마찬가지로 이전 버전에 비해 여러 좋은 기능들이 추가되었고, 기존의 기능들중에서 변경된 기능들도 있습니다.

이 책이 SQL Server를 이전부터 사용하셨던 분에게나 SQL Server를 새로 시작하시는 분에게 SQL Server 2005의 기초적인 정보를 제공할 수 있는 하나의 유효한 방편이 될 뿐만 아니라 좋은 DBA가 되기 위한 작은 도구가 되었으면 합니다.

(2006년 7월 AD 컨설팅)

컨텐츠 관련 문의처

hkjeon@adconsulting.co.kr

목차

SQL Server 구성 옵션 관리

SQL Server 2000에서는 구성 옵션의 수가 37개였는데, SQL Server 2005에서는 62개로 증가하였습니다. 새로운 기능의 추가, 보안 강화, 성능 구성 옵션 등의 추가로 많은 구성 옵션이 추가되었습니다.

모든 경우에 SQL Server 구성 옵션을 변경해야 하는 것은 아니지만, 시스템에 따라 SQL Server 구성 옵션의 조정이 필요한 경우들이 있습니다. 특히 대용량 시스템의 경우에는 SQL Server를 설치한 다음에 서버 구성 옵션을 적절하게 설정하는 것이 필요합니다.

SQL Server 구성 옵션은 SQL Server Management Studio나 sp_configure 시스템 저장 프로시저를 사용하여 확인 및 변경이 가능합니다. 일부 구성 옵션은 SQL Server 노출 영역 구성 도구를 사용하여 구성할 수도 있습니다. sp_configure를 사용하여 전체 구성 옵션을 한번에 확인할 수 있으며, sys.sysconfigurations라는 카탈로그 뷰를 통하여 구성 옵션을 확인할 수도 있습니다. sp_configure를 사용하여 구성 옵션을 조회하는 경우에는 'show advanced options' 구성 옵션이 1인 경우에만 고급 옵션을 조회할 수 있습니다.

다음은 구성 옵션 관리에 있어서의 권고사항입니다.

- DBA는 SQL Server 2005의 구성 옵션에 대하여 충분히 이해하고 있어야 합니다. 일부 구성 옵션의 경우에는 기본값을 그대로 사용하면 SQL Server 2000에서 실행되던 응용 프로그램이 정상적으로 동작하지 않는 문제가 발생할 수도 있으므로 온라인 설명서나 기술 문서를 참조하여 SQL Server 구성 옵션을 숙지하시기 바랍니다.

- DBA는 구성 옵션의 최적화를 통하여 SQL Server 환경을 최적화해야 합니다.

- SQL Server 구성 옵션 변경은 매우 신중한 계획 하에 이루어져야 합니다. 구성 옵션의 변경은 시스템에 영향을 주므로, 사전에 구성 옵션의 변경이 SQL Server나 사용자들에게 어떤 영향을 미치는지를 사전에 충분히 점검해야 합니다.

- SQL Server 구성 옵션에 대한 이력 관리가 필요합니다. 구성 옵션을 변경하기 전에 변경 전의 값을 기록하는 것을 습관화해야 하며, 누가 언제 어떤 이유로 어떤 구성 옵션을 변경했는지 그리고 구성 옵션을 변경한 결과가 어떠했는지 등에 대한 이력 관리가 이루어져야 합니다.

- 성능 향상을 목적으로 SQL Server 구성 옵션을 변경하고자 하는 경우에는, 구성 옵션 변경 전에 성능 데이터를 수집 및 분석하고 구성 옵션 변경 후에 동일한 기준으로 성능 데이터를 수집 및 분석하여 성능 개선 여부를 확인합니다. 성능 개선을 목적으로 구성 옵션을 변경하였는데 실제로는 구성 옵션의 변경이 오히려 성능을 저하시킬 수 있습니다.

- 구성 옵션의 부적절한 변경으로 인하여 장애가 발생하는 심각한 상황까지 가기도 합니다. 사전에 충분한 계획과 확인이 필요합니다.

[따라하기] 전체 구성 옵션 확인하기

```
SELECT name, minimum, maximum, value, value_in_use, description
        , is_dynamic, is_advanced
FROM sys.configurations ORDER BY name;
GO
-- 또는
sp_configure 'show advanced options','1';
RECONFIGURE;
EXEC sp_configure;
GO
```

[따라하기] 특정 고급 구성 옵션 확인하기

```
sp_configure 'show advanced options','1';
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell';
GO
sp_configure 'show advanced options', '0';
RECONFIGURE;
GO
```

SQL Server Management Studio를 사용하여 SQL Server 구성 옵션을 조회 및 변경하는 방법은 SQL Server 2005 온라인 설명서에 자세하게 기술되어 있으므로 다음을 참조하기 바랍니다.

ms-help://MS.SQLCC.v9/MS.SQLSVR.v9.ko/udb9/html/046fcb39-3825-491e-b9d1-92e750878efe.htm

이제 몇 가지 구성 옵션에 대하여 알아보겠습니다.

awe enabled 구성 옵션

AWE란 SQL Server 2005 Enterprise Edition에서 실제 메모리를 4GB 이상 사용할 수 있도록 하는 기능입니다. SQL Server 메모리 구성 옵션을 적절하게 구성하지 않아서 서버가 가지고 있는 실제 메모리를 제대로 활용하지 못하는 경우를 종종 볼 수 있습니다. 예를 들어, SQL Server가 보다 많은 실제 메모리를 사용하면 성능 개선에 도움이 되며 실제 메모리가 8GB이고 사용 가능 메모리는 5GB나 남아 있음에도 불구하고, SQL Server는 2GB미만의 메모리만 사용하는 경우가 있습니다. 이런 문제는 대부분 boot.ini 파일에 /pae 옵션을 지정하지 않거나 'awe enabled' 구성 옵션을 활성화하지 않아서 발생합니다.

하드웨어 리소스를 충분히 활용할 수 있도록 환경을 구성하는 것도 DBA의 중요한 임무이므로 메모리 관련 구성 옵션에 대한 충분한 이해가 필요합니다.

SQL Server가 사용할 수 있는 최대 실제 메모리 용량은 하드웨어 구성과 Windows 운영 체제, SQL Server 에디션에 의해 좌우됩니다. AWE는 SQL Server Enterprise Edition과 Developer Edition에서만 지원되며, 32bit인 경우에만 AWE 구성이 필요합니다. 64 bit 플랫폼의 경우에는 메모리 액세스가 4GB로 제한되지 않기 때문에 AWE 구성이 필요하지 않습니다.

Windows Server 2003 운영 체제부터 지원하는 실제 메모리 용량이 늘어났습니다.

운영 체제	최대 지원 가능한 실제 메모리
Windows Server 2003 Standard Edition	4 GB
Windows Server 2003 Enterprise Edition	32 GB
Windows Server 2003 Datacenter Edition	64 GB

Windows 2000의 경우에는 AWE를 활성화하고 max server memory 구성 옵션을 지정하지 않으면 실제 메모리를 128MB 이하만 남겨 두고 대부분의 사용 가능한 메모리를 SQL Server가 예약하므로 한 대의 서버에서 여러 개의 인스턴스를 실행하거나 다른 응용 프로그램이 실행되는 경우에는 max server memory 구성 옵션을 설정하는 것이 필요합니다. Windows Server 2003에서는 AWE로 매핑된 메모리를 동적으로 관리하므로 SQL Server 서비스 시작 시에는 사용 가능한 전체 실제 메모리에서 소량만 SQL Server에 할당됩니다.

SQL Server 2005가 4GB 이상의 실제 메모리를 사용하려면 다음과 같은 작업이 필요합니다.

[따라하기] AWE 구성하기

1. Windows의 [로컬 보안 정책] 관리 도구를 사용하여 "메모리의 페이지 잠그기(LOCK PAGE IN MEMORY)" 로컬 정책에 SQL Server 서비스 계정으로 지정된 윈도우 사용자 계정을 추가합니다.
2. boot.ini 파일에 /pae 매개 변수를 추가하고 컴퓨터를 다시 시작합니다. (실제 메모리가 16GB 이상인 경우에는 /3gb 매개 변수가 boot.ini 파일에 지정되어 있으면 안됩니다.)
3. SQL Server 구성 옵션 awe enabled를 활성화합니다. awe enabled 옵션은 고급 옵션으로서 show advanced options의 값이 1인 상태에서만 변경 가능합니다.

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
EXEC sp_configure 'awe enabled', 1;
RECONFIGURE;
GO
EXEC sp_configure 'show advanced options', 0;
RECONFIGURE;
GO
```

4. AWE가 실제로 적용되도록 SQL Server 인스턴스를 다시 시작합니다.
5. 현재의 SQL Server 오류 로그 파일에 AWE가 활성화되었다는 내용의 메시지가 기록되었는지 확인합니다.

max server memory와 min server memory 구성 옵션

max server memory 와 min server memory 구성 옵션은 Microsoft SQL Server 데이터베이스 엔진의 버퍼 풀이 사용하는 메모리 양의 상한 및 하한을 설정하는데 사용하는 옵션입니다. min server memory와 max server memory 구성 옵션을 동일한 값으로 지정한 경우에는 데이터베이스 엔진에 할당된 메모리가 해당 값에 도달하면 데이터베이스 엔진이 버퍼 풀에 대한 메모리의 동적 해제 및 확보를 중지합니다. 동일한 컴퓨터에서 여러 개의 SQL Server 인스턴스가 실행되거나, SQL Server 인스턴스 외에 다른 응용 프로그램이 실행되는 경우에는 min server memory와 max server memory 구성 옵션을 사용하여 SQL Server가 사용할 수 있는 메모리의 양을 제어하는 것이 필요합니다.

그런데 SQL Server는 실제로는 max server memory 옵션이 지정한 것보다 더 많은 메모리를 사용합니다. 일반적으로 버퍼 풀에 할당된 메모리가 SQL Server에서 사용하는 메모리의 가장 큰 부분을 차지하지만, SQL Server는 내부 및 외부 구성 요소를 위하여 버퍼 풀 외에도 메모리를 사용합니다. 성능 카운터 SQL Server Total Server Memory에는 버퍼 풀만 포함되며, Thread Memory나 CLR Memory 등은 Total Server Memory에 포함되지 않습니다.

AWE를 사용하는 대용량 시스템에서 max server memory 옵션을 너무 크게 설정하여 문제가 발생하는 경우가 있습니다. 버퍼 풀 외에 SQL Server가 추가로 메모리를 필요로 한다는 것을 염두에 두고 max server memory를 적절하게 구성할 필요가 있습니다.

다음에 max server memory를 계산하는 식이 있습니다. max server memory 값을 설정할 때에는 thread memory의 크기를 고려해야 합니다.

SQL Server max server memory 계산 식:

max server memory = 전체 시스템 메모리
- max worker threads X thread stack size
- 운영 체제/다른 응용 프로그램이 사용하는 메모리 (대개 2-4 GB)

SQL Server max server memory 계산 식 예:

IA64 / 64GB RAM 시스템 / max worker threds = 1024인 경우의
max server memory = 64GB - 1024 X 4MB - 4GB = 57,344 MB

Thread Stack Size:

	Thread Stack Size
32 bit	512 K
X64	2 MB
IA64	4 MB

max worker threads 구성 옵션

max worker threads 옵션은 Microsoft SQL Server 프로세스에 사용할 수 있는 작업자 스레드의 수를 구성하는 옵션입니다.

보통 각각의 클라이언트 연결에 대하여 별도의 운영 체제 스레드가 만들어집니다.
그러나 서버에 대하여 수백 개의 연결이 발생하는 경우에는 각각의 연결에 대하여 스레드를 하나씩 사용하면 시스템 리소스를 상당히 많이 소비하게 되므로 SQL Server에서 성능을 위하여 작업자 스레드 풀을 만들어 많은 클라이언트 연결을 처리합니다.

실제 사용자 연결 수가 max worker threads에 설정된 값보다 적으면 각 연결마다 스레드 하나가 사용됩니다. 그러나 실제 연결 수가 max worker threads에 설정된 값보다 많아지면 SQL Server 가 다음 사용할 수 있는 작업자 스레드가 요청을 처리할 수 있도록 작업자 스레드를 풀링합니다.

max worker threads의 기본값은 0입니다. 기본값인 0을 사용하면 SQL Server 시작 시 작업자 스레드 수가 자동으로 구성됩니다. 이 설정은 대부분의 시스템에서 가장 적합하지만 시스템 구성에 따라 max worker threads를 특정 값으로 설정하면 때때로 성능을 향상시킬 수 있습니다.

max worker threads는 고급 옵션입니다. sp_configure 시스템 저장 프로시저를 사용하여 설정을 변경하는 경우에는 show advanced options 값이 1로 설정된 경우에만 변경할 수 있으며, 새 설정값은 시스템을 다시 시작해야 적용됩니다.

Max Worker Threads 기본 값:

CPU 개수	32-bit	64-bit
<= 4	256	512
8	288	576
16	352	704
32	480	960

- 32비트 시스템의 경우에는 1024를 초과하지 않는 것을 권고합니다.
- 대용량 64비트의 경우에는 2048 또는 그 이상의 값을 설정할 수도 있습니다.

max degree of parallelism 구성 옵션

다중의 CPU를 가지는 컴퓨터에서 SQL Server 2005가 실행되는 경우에는 하나의 SQL 문을 여러 개의 프로세서들이 실행할 수 있습니다. max degree of parallelism 옵션을 사용하여 병렬 계획 실행에 사용할 프로세서 수를 제한할 수 있습니다. 기본값 0은 사용 가능한 모든 프로세서를 사용한다는 의미이고, max degree of parallelism을 1로 설정하면 병렬 계획이 생성되지 않고 하나의 SQL 문은 오직 하나의 프로세서가 실행하게 됩니다.

max degree of parallelism 옵션은 고급 옵션입니다. show advanced options값이 1인 경우에만 sp_configure를 사용하여 max degree of parallelism의 구성값을 변경할 수 있습니다. 이 설정은 SQL Server 서비스를 다시 시작하지 않아도 즉시 적용됩니다.

■ 일반적으로 동시 사용자가 많은 OLTP 시스템에서는 max degree of parallelism 구성 옵션 기본값을 사용하는 대신 1과 같은 작은 값으로 설정하는 것이 성능 향상에 도움이 될 수 있습니다.

■ max degree of parallelism 구성옵션을 1로 설정했더라도 쿼리문에 MAXDOP 쿼리 힌트를 지정하면 쿼리의 max degree of parallelism 값이 무시되고 MAXDOP에 지정한 값이 적용됩니다.

■ 인덱스를 만들거나 다시 작성하는 인덱스 작업 또는 클러스터형 인덱스를 삭제하는 인덱스 작업은 리소스를 많이 필요로 합니다. SQL Server 2005에서는 인덱스 작업 시에 MAXDOP 인덱스 옵션을 지정하여 인덱스 작업의 max degree of parallelism 값을 재정의할 수 있습니다.

default trace enabled 구성 옵션

default trace enabled 옵션을 1로 설정하면 기본 추적 데이터가 수집되어 문제 발생 시 활용할 수 있습니다. 이 옵션의 기본 설정은 1(설정)이며 값을 0으로 변경하면 추적 기능이 해제됩니다. 기본 추적 기능은 문제 발생 시 문제를 진단하는데 유용합니다.

기본 추적 로그는 기본적으로 롤오버 추적 파일을 사용하여 \MSSQL\LOG 디렉터리에 저장되며, 기본 추적 로그 파일의 기본 파일 이름은 log.trc입니다. 기본 추적 로그는 SQL Server 프로파일러에서 열어서 확인하거나 또는 fn_trace_gettable 시스템 함수를 사용하여 Transact-SQL로 쿼리할 수 있습니다.

[따라하기] 기본 추적 구성 옵션값 확인하기

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
-- 기본값이 1입니다.
EXEC sp_configure 'default trace enabled';
GO
EXEC sp_configure 'show advanced options', 0;
RECONFIGURE;
GO
```

[따라하기] 기본 추적 데이터 조회하기

```
SELECT *
FROM fn_trace_gettable
('C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\log_20.trc',
DEFAULT);
GO

SELECT t.StartTime, t.EventClass, e.name,
                    t.DatabaseName, t.ObjectName, t.ObjectID, t.IndexID
FROM fn_trace_gettable
('C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\log_21.trc',0) t
        JOIN sys.trace_events e
ON e.trace_event_id = t.eventclass
ORDER BY t.StartTime;
GO
```

xp_cmdshell 구성 옵션

SQL Server 2005에 새로 추가된 xp_cmdshell 구성 옵션은 시스템 관리자가 시스템에서 xp_cmdshell 확장 저장 프로시저를 실행할 수 있는지 여부를 제어할 수 있도록 해 줍니다. 이전 버전에서는 기본적으로 xp_cmdshell 확장 저장 프로시저를 사용할 수 있었지만 SQL Server 2005에서는 기본적으로 xp_cmdshell 옵션이 비활성화되며, xp_cmdshell을 실행하려면 다음과 같이 sp_configure 시스템 저장 프로시저를 실행하거나 또는 노출 영역 구성 도구를 사용하여 사용 가능하도록 변경해야 합니다.

[따라하기] xp_cmdshell 구성 옵션 활성화하기

```
sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
sp_configure 'xp_cmdshell', 1;
RECONFIGURE;
GO
-- 테스트
EXEC master..xp_cmdshell 'dir C:\"Program Files"\"Microsoft SQL Server"
\MSSQL.1\MSSQL\LOG\log*.trc';
GO
sp_configure 'show advanced options', 0;
RECONFIGURE;
GO
```

clr enabled 구성 옵션

SQL Server에서 사용자 어셈블리를 실행할 수 있는지의 여부를 결정하는 옵션입니다. CLR(common language runtime) integration 기능은 기본적으로 비활성화되어 있으며 CLR integration을 사용하여 구현된 사용자 개체를 사용하기 위해서는 clr enabled option 구성 옵션을 1로 변경해 주어야 합니다. 이 옵션은 고급 옵션이 아닙니다.

[따라하기] clr enabled 활성화하기

```
sp_configure 'clr enabled', 1;
RECONFIGURE ;
GO
```

blocked process threshold 구성 옵션

blocked process threshold 옵션을 지정하면 차단(blocking) 현상이 발생하는 경우에 차단된 프로세스 보고서가 생성됩니다. 기본적으로는 차단된 프로세스 보고서가 생성되지 않습니다. blocked process threshold를 설정한 다음에 SQL Trace나 프로파일러를 사용하여 사용자가 정의한 임계값을 초과하는 blocking 추적 이벤트를 수집할 수 있습니다.
자세한 수집방법은 http://www.microsoft.com/technet/prodtechnol/sql/2005/tsprfprb.mspx를 참조하기 바랍니다.
임계값은 초 단위이며 0부터 86,400까지의 값으로 설정할 수 있습니다. 이 옵션은 고급 옵션이며 즉시 적용됩니다.

[따라하기] blocked process threshold를 20초로 설정하기

```
sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
sp_configure 'blocked process threshold', 20;
RECONFIGURE ;
GO
sp_configure 'show advanced options', 0;
RECONFIGURE;
GO
```

시스템 데이터베이스

시스템 데이터베이스는 SQL Server가 제대로 동작하는데 있어서 매우 중요한 역할을 하는 데이터베이스이므로 시스템 데이터베이스에 대한 이해와 관리가 필요합니다.

시스템 데이터베이스 목록

다음은 SQL Server를 설치하면 기본적으로 만들어지는 시스템 데이터베이스와 그에 대한 설명입니다.

데이터베이스 이름	데이터베이스 ID	설명
master	1	SQL Server 인스턴스에 있는 모든 데이터베이스들에 대한 정보와 각 데이터베이스의 파일 정보, 서버 구성 옵션, 계정 등과 같은 중요한 정보들이 저장됩니다. SQL Server 초기화 정보도 master에 관리되기 때문에 master 데이터베이스를 사용할 수 없는 경우에는 SQL Server 서비스를 시작할 수 없습니다. master 데이터베이스는SQL Server 인스턴스의 중요한 정보를 관리하는 데이터베이스이므로 주기적인 백업이 필요합니다.
tempdb	2	쿼리를 실행하거나 작업을 처리하는데 있어서 임시 작업 공간으로 사용됩니다. tempdb 데이터베이스는 SQL Server가 시작될 때마다 model 데이터베이스를 템플릿으로 사용하여 다시 만들어집니다.

model	3	새로운 데이터베이스를 만들 때 템플릿으로 사용되는 데이터베이스입니다. model 데이터베이스 크기, 데이터베이스 옵션, 복구 모델, 사용자, 정렬 등을 변경하면, 이후에 새로 만들어지는 모든 데이터베이스에 변경사항이 적용됩니다. 테이블, 뷰, 저장 프로시저, 데이터 형식 등과 같은 어떤 개체라도 model 데이터베이스에 생성하면 새로 만들어지는 모든 데이터베이스에 자동으로 만들어집니다. SQL Server를 시작할 때마다 tempdb를 다시 만들어야 하는데 이 때 model 데이터베이스를 템플릿으로 사용하기 때문에 model 데이터베이스는 항상 SQL Server 시스템에 있어야 합니다.
msdb	4	SQL Server, SQL Server Management Studio, SQL Server 에이전트 서비스가 사용하는 작업 일정 정보, 백업 및 복원 기록 정보, 경고, SSIS 패키지, 복제 정보가 저장됩니다. msdb 데이터베이스에 변경이 발생할 때마다 백업해야 하며, 사용자 데이터베이스와 같은 방법으로 백업합니다.
리소스	32767	SQL Server 2005에 포함되어 있는 모든 시스템 개체가 저장되는 읽기 전용 데이터베이스입니다. 시스템 개체들은 실제로는 리소스 데이터베이스에 저장되지만 논리적으로는 모든 데이터베이스의 sys 스키마에서 보입니다. 리소스 데이터베이스에는 사용자 데이터나 사용자 메타데이터는 저장되지 않습니다. 리소스 데이터베이스의 도입으로 인하여 상위 SQL Server 버전으로의 업그레이드가 보다 쉽고 빨라지게 되었습니다.

시스템 데이터베이스 관리

다음은 시스템 데이터베이스 관리에 대한 권고 사항입니다.

■ 시스템 데이터베이스는 SQL Server 운영에 있어서 매우 중요한 데이터베이스이므로 내결함성이 있는 저장소에 보관합니다.

■ 시스템 데이터베이스에 대해서도 사용자 데이터베이스와 마찬가지로 백업 정책의 수립 및 백업 관리가 필요합니다.

master, msdb, model 데이터베이스에 대한 변경을 발생시키는 작업을 실행한 경우에는 해당 데이터베이스를 백업해야 하며, 이들 데이터베이스에 대한 백업 정책의 수립 및 주기적인 백업을 권고합니다. tempdb는 SQL Server가 시작될 때마다 다시 만들어지는 임시 작업 공간이므로 백업 대상에서 제외되며, 리소스 데이터베이스의 경우에는 사용자 데이터나 사용자 메타 데이터는 저장되지 않으며 코드만 저장되기 때문에 일반적인 데이터베이스 백업이나 복원 대상에서는 제외됩니다.

■ 시스템 데이터베이스의 시스템 테이블을 직접 쿼리하지 않습니다.

시스템 테이블을 직접 쿼리하는 Transact-SQL 문을 작성하는 것이 응용 프로그램에 필요한 정보를 얻는 유일한 방법이 아닌 한 직접 시스템 테이블을 조회하지 말고, 다음 항목을 사용하여 카탈로그나 시스템 정보를 조회합니다. 이 사항은 사용자 데이터베이스의 시스템 테이블에도 해당됩니다.

- 시스템 카탈로그 뷰 : 권장되는 액세스 방법
  SQL Server 2005 에서는 시스템 카탈로그 메타데이터를 표시하는 완전히 새로운 방식의 관계형 인터페이스로서 카탈로그 뷰를 제공합니다. 이 뷰를 사용하면 서버의 모든 데이터베이스에 저장된 메타데이터에 액세스할 수 있습니다.

메타데이터 액세스 방법으로 카탈로그 뷰가 권장되는 이유는 다음과 같습니다.

1. 카탈로그 뷰는 카탈로그 테이블 구현과는 다른 독립적인 형식으로 메타데이터를 제공하므로 기본 카탈로그 테이블이 변경되더라도 영향을 받지 않습니다.
2. 카탈로그 뷰는 핵심 서버 메타데이터에 액세스하는 가장 효율적인 방법입니다.
3. 카탈로그 뷰는 카탈로그 메타데이터를 표시하는 일반적인 인터페이스로서 이러한 메타데이터를 사용자 지정 형식으로 가져오고 변환하고 나타내는 가장 빠른 방법을 제공합니다.
4. 카탈로그 뷰 이름 및 열 이름에는 설명이 포함됩니다. 따라서 쿼리할 메타데이터에 해당하는 기능에 대해 일정 수준의 지식을 갖춘 사용자라면 쉽게 원하는 쿼리 결과를 얻을 수 있습니다.

- 정보 스키마 뷰

  정보 스키마 뷰는 SQL-92 표준의 카탈로그 뷰 정의를 기반으로 합니다.
  정보 스키마 뷰 역시 카탈로그 뷰와 마찬가지로 테이블 구현과는 다른 독립적인 형식으로 카탈로그 정보를 제공하므로 기본 카탈로그 테이블이 변경되더라도 영향을 받지 않습니다. 또한 이러한 뷰를 사용하는 응용 프로그램은 다른 유형의 SQL-92 규격 데이터베이스 시스템으로의 이식이 가능합니다. 그렇지만, 정보 스키마 뷰에는 SQL Server 2005 에서만 사용할 수 있는 메타데이터는 포함되지 않습니다.

- 시스템 저장 프로시저 및 기본 제공 함수

  Transact-SQL 은 카탈로그 정보를 반환하는 서버 시스템 저장 프로시저와 시스템 함수를 제공하며, 시스템 저장 프로시저를 통해 다양한 관리 및 정보 조회 작업을 수행할 수 있습니다. 시스템 저장 프로시저와 기본 제공 함수에 대한 자세한 내용은 SQL Server 2005 온라인 설명서를 참조하기 바랍니다. 이러한 저장 프로시저와 함수는 SQL Server 에서만 사용 가능합니다.

■ 시스템 데이터베이스의 테이블들을 직접 업데이트하지 않습니다.

시스템 데이터베이스에 있는 테이블들을 직접 업데이트해서는 안됩니다. 만일 시스템 데이터베이스의 변경이 필요한 경우에는 적당한 관리 도구나 저장 프로시저를 활용하여 변경해야 합니다. 데이터베이스 템플릿으로 사용되는 model 데이터베이스는 예외입니다.

■ 성능을 위하여 tempdb 데이터베이스에 대한 관리가 필요합니다.

tempdb는 사용자가 명시적으로 만든 임시 테이블, 임시 저장 프로시저, 테이블 변수, 커서 등의 임시 개체들과 SQL Server 데이터베이스 엔진에서 만드는 내부 작업 테이블, 스냅숏 격리가 사용되는 경우 업데이트된 레코드의 모든 버전, SORT_IN_TEMPDB를 지정하여 인덱스를 만들거나 다시 작성할 때의 임시 정렬 결과를 보관하기 위한 작업 영역을 제공합니다. tempdb 데이터베이스의 크기와 물리적인 배치가 시스템의 성능에 영향을 미칠 수 있습니다. SQL Server 2005에서는 이전 버전들보다 tempdb를 더 많이 사용하므로 적절한 tempdb 공간에 대한 용량 계획과 tempdb 사용 모니터링, 적절한 크기로의 확장이 필요합니다. SQL Server를 설치하면 기본적으로 tempdb의 주 데이터 파일 크기가 8 MB이고 필요할 때 10%씩 자동 확장되도록 설정됩니다. 처리량이 많은 운영 서버의 경우에는 8MB로는 부족하므로 tempdb가 빈번하게 자동 확장되는 문제가 발생합니다. 그러므로 tempdb 자동 확장으로 인한 오버헤드가 발생하지 않도록 사전에 tempdb를 충분한 크기로 확장해야 합니다. tempdb는 SQL Server가 재시작될 때마다 다시 만들어지기 때문에 만일 tempdb를 사전에 적절한 크기로 확장하지 않으면 SQL Server가 재시작한 이후에 매번 빈번한 자동 확장이 발생하므로, tempdb를 많이 사용하는 시스템의 경우에는 SQL Server를 설치한 이후에 가능한 한 빨리 tempdb 데이터 파일과 로그 파일의 크기를 적절한 크기로 확장시켜 주어야 합니다. 만일 SQL Server를 설치한 다음에 tempdb 확장 작업을 한번도 수행하지 않았다면 현재 tempdb 크기를 확인하고 tempdb 초기 크기보다 커졌다면 현재 크기에 여유공간을 추가하여 수동으로 확장해 줄 것을 권고합니다.

- tempdb 성능 최적화
  tempdb의 크기를 적절하게 미리 확장시켜 줌으로써 tempdb 자동 확장으로 인한 오버헤드를 제거해야 합니다. 시스템에 따라 차이가 있지만, tempdb 데이터베이스의 물리적인 배치와 데이터베이스 옵션에 대한 일반적인 권고 사항은 다음과 같습니다.

1. tempdb 데이터베이스는 필요할 때 자동으로 확장되도록 설정합니다. 그렇지만 자동 확장이 가능한 한 발생하지 않도록 관리하는 것을 권고합니다.
2. tempdb 데이터베이스 파일의 최초 크기를 그대로 두지 말고 자동 확장이 발생하지 않도록 예상 최대 크기만큼 사전에 확장해 줍니다.
3. 자동 확장이 발생하는 경우에 대비하여, tempdb 데이터베이스 파일이 너무 작은 크기로 확장됨으로써 계속 자동 확장이 발생하지 않도록, 파일 증가량을 적절한 크기로 설정합니다.
4. 성능을 위하여 tempdb 데이터베이스는 빠른 I/O 서브시스템에 배치하며, tempdb는 사용자 데이터베이스와 다른 디스크에 저장하며, 가능하다면 tempdb를 여러 디스크에 스트라이핑하는 것을 권고합니다.

- tempdb 용량 산정

  tempdb의 크기는 해당 시스템에서 발생하는 작업 부하와 실제로 어떤 SQL Server 기능을 사용하는가에 따라 적절한 tempdb의 크기가 달라지므로 테스트 서버를 구축하고 다음과 같은 일련의 작업을 통하여 tempdb의 크기를 예측하는 것을 권고합니다.

  1. tempdb에 대하여 자동 확장으로 설정합니다.
  2. 해당 시스템에서 주로 사용하는 쿼리를 실행하거나 작업 부하를 저장한 추적 파일들을 실행하면서 tempdb가 사용하는 공간을 모니터링합니다.
  3. 인덱스 재구성과 같은 인덱스 관리 작업을 실행하면서 tempdb가 사용하는 공간을 모니터링합니다.
  4. 앞 단계에서 얻은 결과값을, 예상되는 현재의 작업 부하에 맞도록 조정한 다음에 그에 따라 tempdb 크기를 결정합니다.

- tempdb 공간 모니터링

  sys.dm_db_file_space_usage 동적 관리 뷰를 사용하여 tempdb 파일의 디스크 공간을 모니터링할 수 있습니다.

  그리고 sys.dm_db_session_space_usage와 sys.dm_db_task_space_usage 동적 관리 뷰를 사용하면 session 또는 task 차원의 tempdb내 페이지 할당이나 페이지 해제를 모니터링 할 수 있습니다. 이런 뷰를 활용하면 tempdb 디스크 공간을 많이 사용하는 쿼리, 임시 테이블, table 변수를 확인할 수 있습니다.

- tempdb를 사용하는 작업
  1. DBCC CHECKDB
  2. 내부 개체
     - 커서 또는 스풀 작업(예:CTE)에 대한 작업 테이블
     - 임시 LOB(Large Object) 저장소
     - 해시 조인 또는 해시 집계 작업에 대한 작업 파일
     - 인덱스 생성 또는 다시 작성 시 SORT_IN_TEMPDB 옵션을 사용한 경우
     - GROUP BY, ORDER BY, UNION 쿼리의 중간 정렬 결과
  3. 서비스 브로커
  4. 사용자 개체
     - 전역 임시 테이블 및 인덱스, 로컬 임시 테이블 및 인덱스, 임시 저장 프로시저
     - table 변수
     - 커서
     - 테이블 값 함수에서 반환된 테이블
  5. 행 버전 관리
     - MARS (Multiple Active Result Sets)
     - 온라인 인덱스 재구성
       - 인덱스 크기의 2 ~ 3배가 필요함 (정렬, 임시 인덱스, 롤백)
     - Row versioning
       - [버전 저장소 크기]=2*[분당 생성되는 버전 저장소 데이터]*
         [가장 오래 실행되는 트랜잭션 실행 시간(분)] *
         동시에 수행되는 트랜잭션/사용자 수
       - 성능 모니터의 SQL Server 2005:Transactions 성능 개체에 버전 저장소와 관련하여 다음과 같은 성능 카운터들이 추가되었습니다.
         - Version Store Size (KB)
         - Version Store unit count
         - Version Store unit creation
         - Version Store unit truncation
     - 트리거

- tempdb의 동시성 강화

  큰 규모의 시스템에서 빈번하게 호출되는 저장 프로시저들에 임시 테이블과 인덱스를 만드는 DDL 문이 포함되어 있다면 SQL Server 2005에서도 추적 플래그 1118 이 필요하다고 합니다. 이는 tempdb 경합이 심하게 발생할 가능성이 있기 때문이며, tempdb 경합이 심하게 발생하면 시스템이 응답하지 않는 것과 같은 심각한 상황이 발생할 수도 있습니다.

  사용량이 많은 tempdb의 할당 리소스 경합을 줄이기 위하여 다음과 같은 방법을 사용합니다.

  1. SQL Server 시작 옵션에 추적 플래그 -T1118을 추가합니다. 추적 플래그 1118은 SGAM contention을 감소시키기 위하여 사용하는 추적 플래그입니다. (참고로 Shared Global Allocation Map 하나에 64000 개의 mixed extent들의 상태 정보가 기록됩니다.)
  2. 프로세서의 수와 동일한 숫자만큼 tempdb 데이터 파일 수를 늘립니다. 이 때 tempdb 데이터 파일의 크기는 모두 동일하게 만듭니다. 가령 8 CPU 서버라면 동일한 크기의 tempdb 데이터 파일을 8개가 되도록 데이터 파일을 추가합니다. 로그 파일은 하나인 상태 그대로 유지합니다.
  3. 계획휴지 시에 SQL Server를 재시작합니다.

■ model 데이터베이스를 활용하면 새로 만들어지는 데이터베이스에 원하는 옵션이나 개체가 상속되어 편리합니다.

model 데이터베이스는 새로운 데이터베이스가 만들어질 때 템플릿으로 사용되는 데이터베이스이므로, 그 SQL Server 인스턴스에서 새롭게 만들어지는 모든 데이터베이스에 데이터베이스 옵션이나 개체들이 동일하게 구성되는 것을 필요로 할 때 활용할 수 있습니다.

예를 들어, SQL Server 인스턴스에서 공통적으로 사용되는 별칭 데이터 형식을 model 데이터베이스에 생성해 두면 새로운 데이터베이스를 만들 때 별칭 데이터 형식 생성 작업을 수행하지 않아도 됩니다.

또한 새로운 데이터베이스를 만들 때 FOR ATTACH가 지정된 경우를 제외하면 model 데이터베이스의 데이터베이스 옵션 설정을 상속받습니다. 그러므로 ALTER DATABASE를 사용하여 model 데이터베이스의 옵션을 변경하면 변경된 옵션 설정이 이후에 새로 만들어지는 모든 데이터베이스에 적용됩니다.

[따라하기] model 데이터베이스의 옵션 변경하기

```
ALTER DATABASE model
SET AUTO_UPDATE_STATISTICS OFF,
    RECOVERY BULK_LOGGED;
GO
```

여러 테이블에 있는 열에 동일한 데이터 형식을 저장해야 하고 각 열의 데이터 형식, 길이가 동일해야 하는 경우에 별칭 형식을 사용할 수 있습니다. 기본 NULL 속성도 설정 가능하며, 디폴트 정의나 규칙도 바인딩이 가능합니다.

[따라하기] 별칭 데이터 형식 생성 및 활용하기

```
USE model;
GO
CREATE TYPE UDT_SSN FROM char(13) NOT NULL ;
GO
CREATE RULE Rule_SSN AS
@SSN LIKE '[0-9][0-9][0-1][0-9][0-3][0-9][1-4][0-9][0-9][0-9][0-9][0-9][0-9]';
GO
EXEC sp_bindrule 'Rule_SSN','UDT_SSN'
GO
CREATE DATABASE test;
GO
USE test;
GO
CREATE TABLE Members (
        MembID              varchar(50)    NOT NULL,
        MembName            varchar(50)    NOT NULL,
```

```
        Ssn                     UDT_SSN     NULL
CONSTRAINT PK_Members PRIMARY KEY CLUSTERED (MembID)
);
GO
CREATE TABLE Emps (
        EmpID                   int         NOT NULL,
        EmpName                 varchar(50) NOT NULL,
        Ssn                     UDT_SSN
CONSTRAINT PK_Emps PRIMARY KEY CLUSTERED (EmpID)
);
GO
```

■ **SQL Server 2005에서 새롭게 추가된 리소스 데이터베이스에 대하여 제대로 이해하고 다음 사항에 유의해야 합니다.**

- 리소스 데이터베이스의 물리 파일 이름은 mssqlsystemresource.mdf 인데, 이 파일의 이름을 변경하거나 잘못 이동하면 SQL Server가 시작되지 않으므로 유의합니다. 리소스 데이터베이스에 대하여 사용자가 수행해도 무방한 작업은, master 데이터베이스를 이동한 경우에 ALTER DATABASE MODIFY FILE 문을 사용하여 FILENAME의 경로를 master 데이터 파일의 경로와 동일하게 변경함으로써 mssqlsystemresource.mdf 파일과 mssqlsystemresource.ldf 파일의 위치를 변경하는 것입니다.

- 리소스 데이터베이스를 재구성(rebuild)하면 이전에 적용했던 서비스팩이나 QFE가 모두 유실되므로, 다시 적용해 주어야 합니다.

- 리소스 데이터베이스는 SQL Server 서비스가 단일 사용자 모드인 상태에서만 액세스가 가능하며, 그렇지 않은 경우에는 사용자들에게 보이지도 않고 연결할 수도 없습니다. 단일 사용자 모드에서 USE mssqlsystemresource를 실행하면 리소스 데이터베이스에 액세스됩니다. 그렇지만 이 데이터베이스는 문제 해결과 고객 기술 지원을 위하여 마이크로소프트 고객 기술 지원 서비스 전문가만 액세스하는 것을 원칙으로 합니다.

- 리소스 데이터베이스는 데이터베이스 백업이 되지 않습니다. 리소스 데이터베이스 파일인 mssqlsystemresource.mdf 파일을 이진(.EXE) 파일로 취급하여 실행 파일을 백업하듯이 파일 백업이나 디스크 백업으로 백업합니다.

- 리소스 데이터베이스 복원도 SQL Server에서는 불가하며, mssqlsystemresource.mdf 파일을 백업받은 복사본을 수동으로 복사함으로써 복원합니다. mssqlsystemresource.mdf 파일 백업을 복원한 다음에는, 이후에 이루어진 모든 변경사항을 다시 적용해 주어야 합니다.

- 리소스 데이터베이스와 관련하여 중요한 값은 버전 번호와 최종 업데이트 일시입니다.

[따라하기] 리소스 데이터베이스의 버전 번호 확인하기

```
SELECT SERVERPROPERTY('ResourceVersion');
GO
```

[따라하기] 리소스 데이터베이스의 최종 업데이트 일시 확인하기

```
SELECT SERVERPROPERTY('ResourceLastUpdateDateTime');
GO
```

[따라하기] 시스템 개체의 SQL 정의 확인하기

```
SELECT OBJECT_DEFINITION(OBJECT_ID('sys.objects'));
GO
```

[참고] SQL Server 서비스를 단일 사용자 모드로 시작하기

시작 옵션에 -m을 지정하면 SQL Server 인스턴스가 단일 사용자 모드로 시작됩니다. 예를 들어 어떤 문제가 발생하여 손상된 master 데이터베이스나 손상된 다른 시스템 데이터베이스를 복구하려고 하는 경우 또는 서버 구성 옵션을 변경하고자 하는 경우에는 SQL Server 인스턴스를 단일 사용자 모드로 시작해야 합니다.
단일 사용자모드로 SQL Server 인스턴스를 시작하기 전에 SQL Server 에이전트 서비스를 중단해야 합니다. 그렇지 않으면 SQL Server 에이전트 서비스에서 연결을 사용하여 연결을 차단 당합니다.
SQL Server 인스턴스를 단일 사용자 모드로 시작하면 다음과 같이 변경됩니다.

- *한 시점에는 오직 사용자 한 명만 서버에 연결할 수 있습니다.*
- *기본적으로 서버 시작 시에 자동 실행되는 CHECKPOINT 프로세스가 실행되지 않습니다.*
- *기본적으로 비활성화되어 있는 allow updates 구성 옵션이 활성화됩니다.*

tempdb 데이터베이스 이동

tempdb 데이터베이스의 크기와 물리적인 배치가 시스템의 성능에 영향을 미칠 수 있으므로 tempdb를 확장하거나 이동하는 작업을 수행하는 경우가 종종 발생합니다. 다음에 나오는 일련의 작업은 동일한 SQL Server 인스턴스에서 tempdb를 다른 위치로 이동하는 방법을 보여 주는 예제입니다.

[따라하기] tempdb를 디스크 상의 다른 위치로 이동하기

1. tempdb 데이터베이스의 논리 파일 이름을 확인합니다.

```
SELECT name, physical_name AS Current_Location, state_desc
FROM sys.master_files
WHERE database_id = DB_ID(N'tempdb');
GO
```

결과 예:

Name	current_Location	state_desc
Tempdev	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\tempdb.mdf	ONLINE
Templog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\templog.ldf	ONLINE

2. ALTER DATABASE 명령어를 사용하여 파일의 위치를 변경합니다.

```
ALTER DATABASE tempdb
MODIFY FILE
    (NAME = tempdev,
    FILENAME = 'E:\DBdata\tempdb.mdf');
GO
```

```
ALTER DATABASE tempdb
MODIFY FILE
    (NAME = templog,
    FILENAME = 'F:\DBdata\templog.ldf');
GO
```

성공적으로 작업이 수행되면 다음과 같은 메시지가 반환됩니다.

시스템 카탈로그에서 파일 "tempdev" 이(가) 수정되었습니다.
새 경로는 다음에 데이터베이스가 시작될 때 사용됩니다.
시스템 카탈로그에서 파일 "templog" 이(가) 수정되었습니다.
새 경로는 다음에 데이터베이스가 시작될 때 사용됩니다.

3. SQL Server를 중지한 후 다시 시작합니다.

4. SQL Server 서비스가 시작된 다음에, 제대로 이동되었는지 확인합니다.

```
SELECT name, physical_name AS Current_Location, state_desc
FROM sys.master_files
WHERE database_id = DB_ID(N'tempdb');
GO
```

5. 기존의 tempdb 파일들을 삭제합니다. tempdb는 SQL Server가 재시작할 때마다 다시 만들어지므로 기존의 tempdb 파일들을 새로운 위치로 이동할 필요가 없습니다.

데이터베이스 관리

데이터베이스 생성

데이터베이스를 만들기 위해서는 먼저 데이터베이스 이름, 크기, 데이터베이스 저장에 사용할 파일 및 파일 그룹, 소유자를 결정해야 합니다. 소유자는 데이터베이스를 만든 사용자입니다.

CREATE DATABASE 문을 실행하거나 SQL Server Management Studio를 사용하여 새로운 데이터베이스와 데이터베이스 저장에 필요한 파일들을 만들 수 있습니다. 새로운 데이터베이스를 만들면 model 데이터베이스에 있는 모든 사용자 정의 개체가 새로 만든 데이터베이스로 복사되며, 크기 매개 변수를 지정하지 않고 CREATE DATABASE 〈database_name〉 문을 실행하면 model 데이터베이스의 주 파일과 같은 크기의 주 데이터 파일이 생성됩니다.

모든 SQL Server 2005 데이터베이스는 기본적으로 데이터 파일과 로그 파일로 구성됩니다. 데이터 파일에는 테이블, 인덱스, 저장 프로시저, 뷰 등의 개체와 데이터가 저장될 수 있으며, 로그 파일에는 데이터베이스의 트랜잭션을 복구하는 데 필요한 정보가 저장됩니다.

데이터베이스를 저장하는 데 사용되는 파일 형식은 주 파일, 보조 파일, 트랜잭션 로그등 3가지이며, 주 파일과 보조 파일은 데이터 파일이고, 트랜잭션 로그는 로그 파일입니다.
데이터 파일은 할당 및 관리를 간편하게 수행하기 위해 파일 그룹으로 그룹화할 수 있습니다.

다음은 데이터베이스를 저장하는 데 사용되는 3가지 파일 형식의 설명입니다.

데이터베이스 파일	설명
주 데이터 파일	데이터베이스의 시작 정보를 포함하며, 모든 데이터베이스에는 반드시 하나의 주 데이터 파일이 존재합니다. 사용자 데이터를 이 파일에 저장할 수도 있고, 보조 데이터 파일에 저장할 수도 있습니다. 권장되는 주 데이터 파일 확장명은 .mdf 입니다.
보조 데이터 파일	선택적으로 사용하는 사용자 정의 데이터 파일이며 사용자 데이터를 저장합니다. 보조 파일은 서로 다른 디스크 드라이브에 파일을 분산 배치하여 데이터를 여러 디스크로 분산시키는 데 활용할 수 있습니다. 권장되는 보조 데이터 파일 확장명은 .ndf 입니다.
트랜잭션 로그 파일	데이터베이스 복구에 사용되는 로그 정보를 저장하며, 데이터베이스마다 최소한 하나의 로그 파일이 있어야 합니다. 권장되는 트랜잭션 로그 파일 확장명은 .ldf 입니다.

다음은 데이터베이스 생성 시의 권고 사항입니다.

■ 데이터 파일과 트랜잭션 로그 파일은 물리적으로 서로 다른 드라이브에 배치합니다.
모든 데이터베이스는 최소 하나의 주 데이터 파일(Primary Data File)과 하나의 트랜잭션 로그 파일로 구성되며, 하나의 데이터베이스에 로그 파일이 두 개 이상 있을 수 있습니다. 디스크 손상 시의 복구 가능성과 성능을 위하여 데이터 파일과 트랜잭션 로그 파일은 물리적으로 서로 다른 디스크에 배치합니다.

[따라하기] 데이터 파일과 로그 파일을 지정하여 데이터베이스 만들기

```
CREATE DATABASE Sales
ON
( NAME = Sales_dat,
  FILENAME = 'D:\DBData\saledat.mdf',
  SIZE = 10MB,
  MAXSIZE = 50MB,
  FILEGROWTH = 5MB )
LOG ON
( NAME = Sales_log,
  FILENAME = 'E:\DBData\salelog.ldf',
  SIZE = 5MB,
  MAXSIZE = 25MB,
  FILEGROWTH = 5MB );
GO
```

■ 파일 그룹을 사용하여 데이터베이스를 생성합니다.

사용자 정의 파일 그룹을 만들어 데이터 파일을 그룹화함으로써 관리, 데이터 할당 및 배치를 간편하게 수행할 수 있습니다. 주(PRIMARY) 파일 그룹의 주 데이터 파일에는 메타 데이터만 저장하고, 사용자 정의 파일 그룹에 사용자 개체들을 저장하는 것을 권고합니다. 사용자 정의 파일 그룹을 도입하게 되면, 기본 파일 그룹을 주 파일 그룹에서 사용자 정의 파일 그룹으로 변경합니다. 그리고 테이블이나 인덱스를 생성할 때 적절한 사용자 파일 그룹을 지정해야 하며, 파일 그룹을 지정하지 않은 상태에서 테이블 또는 인덱스를 생성하면 기본 파일 그룹으로 지정된 파일 그룹에 생성됩니다.

[따라하기] 파일 그룹이 있는 데이터베이스 만들기

1. 데이터베이스를 생성합니다.

```
USE master;
GO
CREATE DATABASE Sample                          /* 데이터베이스 이름 */
ON Primary (                                    /* 주 파일 그룹 */
    NAME = Sample_Pri_dat,                      /* 데이터 파일 이름 */
   FILENAME = 'D:\DBdata\Sample_Pri_dat.mdf',   /* 데이터 파일 위치 */
   SIZE = 200 MB,                               /* 데이터 파일 초기 크기 */
   MAXSIZE = 1 GB,                              /* 데이터 파일 최대 크기 */
   FILEGROWTH = 20 MB),                         /* 데이터 파일 증가량 */
FILEGROUP SamplesFG1 (                          /* 사용자 정의 파일 그룹 #1 */
   NAME = Sample_FG1_dat,
   FILENAME = 'E:\DBdata\Sample_FG1_dat.ndf',
   SIZE = 200 MB,
   MAXSIZE = 1 GB,
   FILEGROWTH = 20 MB),
FILEGROUP SamplesFG2 (                          /* 사용자 정의 파일 그룹 #2 */
   NAME = Sample_FG2_dat,
   FILENAME = 'F:\DBdata\Sample2_FG2_dat.ndf',
   SIZE = 200 MB,
   MAXSIZE = 1 GB,
   FILEGROWTH = 20 MB)
LOG ON (
/* 로그 파일 */
   NAME = Sample_log,
   FILENAME = 'G:\DBlog\Sample_log.ldf',
   SIZE = 10 MB,
```

```
    MAXSIZE = 50 MB,
      FILEGROWTH = 5 MB);
  GO
```

2. 주 데이터 파일에는 메타 데이터만 저장되도록, 기본 파일 그룹을 사용자 정의 파일 그룹 중 주로 사용하는 파일 그룹으로 변경합니다.

```
ALTER DATABASE Sample
MODIFY FILEGROUP [SamplesFG1] DEFAULT;
GO
```

3. 기본 파일 그룹을 확인합니다.

```
USE Sample;
GO
SELECT * FROM sys.filegroups WHERE is_default = 1;
GO
```

4. 테이블을 SamplesFG1 파일 그룹에 생성합니다. ON 절에 파일 그룹을 지정하면 됩니다.

```
CREATE TABLE TestTbl (col1 int, col2 int)
ON SamplesFG1;
GO
```

5. 클러스터형 인덱스를 생성합니다. 클러스터형 인덱스는 기본적으로 데이터와 동일한 파일 그룹에 생성되므로 파일 그룹을 지정할 필요가 없습니다. 만일 이 예제에서 클러스터형 인덱스를 만드는 CREATE INDEX 문의 ON 〈파일 그룹〉 절에 SamplesFG2를 지정하면 데이터도 SamplesFG2 파일 그룹으로 이동합니다.

```
  CREATE CLUSTERED INDEX CX_col1 ON TestTbl (col1);
  GO
```

6. 비클러스터형 인덱스는 SamplesFG2 파일 그룹에 생성합니다.

```
CREATE NONCLUSTERED INDEX NX_col2 ON TestTbl (col2) ON SamplesFG2;
GO
```

7. 인덱스 정보를 조회하면 다음과 같이 지정한 파일 그룹에 배치되어 있는 것을 확인할 수 있습니다.

인덱스 이름	설명	인덱스 키
CX_col1	clustered located on SamplesFG1	col1
NX_col2	nonclustered located on SamplesFG2	col2

■ 데이터 파일은 내결함성이 제공되며 해당 시스템에 적합한 RAID에 배치합니다.

데이터 파일은 RAID 1+0 또는 RAID 5로 구성된 드라이브에 배치하는 것이 일반적입니다. DBA는 RAID와 각 옵션의 장단점에 대하여 명확히 이해하고 디스크 드라이브가 해당 시스템에 최적으로 구성될 수 있도록 하드 디스크 구성에 관여하는 것이 바람직합니다. 하드웨어 RAID가 소프트웨어 RAID에 비해 비용이 많이 들지만 성능을 위하여 하드웨어 RAID를 사용하는 것을 권고하며, 비용이 문제가 되지 않는다면 높은 고가용성과 쓰기 작업의 성능을 위하여 데이터베이스의 데이터 파일은 디스크 스트라이핑과 미러링의 결합인 RAID 1+0로 구성된 드라이브에 배치하는 것을 권고합니다. 유의할 사항은, 데이터베이스 파일과 운영 시스템의 페이징 파일을 동일한 디스크에 배치하는 것은 어떤 경우라도 피해야 합니다.

■ 트랜잭션 로그 파일은 RAID 1로 구성된 드라이브에 배치합니다.

트랜잭션 로그를 별도의 미러된 드라이브에 배치하면 좋은 성능과 내결함성(Fault Tolerance)을 확보할 수 있습니다. 복제가 구성되어 있거나 트리거가 빈번하게 수행되는 경우가 아니라면, 트랜잭션 로그에는 주로 순차적인 쓰기 작업이 수행되며 롤백이 수행되는 경우에만 읽기 작업이 수행됩니다. 그러므로 쓰기 작업의 성능을 위하여 트랜잭션 로그 파일은 RAID 1에 저장할 것을 권고합니다. RAID 5는 RAID 1에 비해 디스크 구입에 드는 비용는 적지만, 한 번의 쓰기 작업에 대하여 네 번의 I/O 작업이 필요하기 때문에 쓰기 작업의 성능이 RAID 1에 비해 좋지 않으며 만일 두 개 이상의 디스크 드라이브가 동시에 손상되면 전체 어레이가 사용할 수 없게 되는 단점이 있습니다.

■ 데이터베이스를 만들 때 데이터 파일과 트랜잭션 로그 파일은 향후 예상되는 데이터 크기를 고려하여 충분한 크기로 생성합니다.

데이터 파일, 트랜잭션 로그 파일 모두 충분한 크기로 생성합니다. 파일이 증가하는 동안에는 새로운 익스텐트의 할당이 중단되므로 새로운 공간 할당을 필요로 하는 INSERT 작업이나 UPDATE 작업의 수행이 중단됩니다. 그러므로, 트랜잭션 로그를 자동 증가하도록 옵션을 설정하기는 하되, 파일이 자동 확장되지 않도록 충분한 크기로 생성합니다. 데이터 파일의 초기 크기는 용량 산정 작업을 통하여 적절한 크기를 결정하고, 트랜잭션 로그의 초기 크기는 트랜잭션 로그 백업을 수행한 후 다음 로그 백업이 수행되기 전까지 발생하는 로그를 저장하기에 충분한 크기로 생성합니다. 트랜잭션 로그 파일에 대하여 여러 번의 자동 증가가 발생하게 되면, 여러 개의 가상 로그 파일들로 조각화되어 로그 관련 작업의 성능에도 좋지 않은 영향을 미칩니다. 만일 데이터베이스가 이미 만들어져 운영 중인데, 데이터베이스의 초기 크기가 작아서 잦은 확장이 발생하고 있다면 파일의 크기를 충분한 크기로 확장해 주어야 합니다. 참고로, 빈번한 자동 확장으로 인하여 이미 가상 로그 파일의 수가 많아진 경우에는 [데이터베이스 축소]를 참조하여 가상 로그 파일의 수를 줄여 줄 것을 권고합니다.

[따라하기] 데이터베이스 파일 확장하기

```
ALTER DATABASE Sample
MODIFY FILE
  (NAME = Sample_dat,
  SIZE = 500 MB);
GO
```

■ 파일이 증가할 수 있는 최대 크기를 지정할 수 있습니다.

데이터 파일이 최대 크기에 도달하거나 드라이브에 파일 공간을 모두 사용한 경우에는 데이터 파일에 대한 쓰기 작업이 오류가 발생합니다. 지속적으로 데이터가 증가하는 데이터베이스의 경우에는 최대 크기에 도달해서 오류가 발생하지 않도록 최대 크기를 지정하지 않는 것이 좋고, 비정상적으로 특정 데이터베이스 파일의 크기가 증가하여 디스크 공간을 모두 사용함으로써 다른 데이터베이스들에까지 영향을 미치는 것을 방지하고자 한다면 최대 크기를 지정하는 것이 좋습니다.

파일의 최대 크기를 지정하려면 CREATE DATABASE 문의 MAXSIZE 매개 변수를 사용하거나 SQL Server Management Studio의 데이터베이스 속성 창의 파일 탭에서 자동 증가 열에 있는 (…) 버튼을 클릭하여 제한된 파일 증가 옵션을 선택한 후 최대 파일 크기를 설정할 수 있습니다.

[따라하기] 데이터베이스 파일의 최대 크기 설정하기

```
ALTER DATABASE Sample
MODIFY FILE
  (NAME = Sample_dat,
  MAXSIZE = 1 GB);
GO
```

[따라하기] 데이터베이스 파일의 최대 크기 확인하기

```
EXEC sys.sp_helpdb Sample;
GO
-- 또는
EXEC Sample.sys.sp_helpfile;
GO
-- 또는
SELECT * FROM Sample.sys.database_files;
GO
```

■ 파일이 자동으로 증가하도록 설정하는 경우에는 자동 증가 크기를 적절하게 설정합니다. 파일의 크기가 매우 작거나 매우 큰 경우에는 파일 증가 단위를 적절한 크기의 MB 단위로 설정하는 것을 권고합니다. SQL Server 2005에서는 파일 증가 단위를 지정하지 않으면 데이터 파일은 기본값이 1MB 단위로, 로그 파일은 10% 단위로 증가하도록 설정됩니다.

이러한 기본 설정을 그대로 두면, 저장하고자 하는 데이터의 크기가 매우 큰 경우에 1MB 단위로 빈번하게 자동 증가가 발생하게 되는 문제가 발생합니다. 10% 단위로 증가하도록 설정된 로그 파일의 경우, 로그 파일의 크기가 매우 크다면 10%에 해당하는 파일의 크기도 커지게 됩니다. 그러므로, 자동 증가가 완료되기까지 소요시간이 오래 걸리고, 이로 인하여 트랜잭션 로그를 발생시키는 작업들이 대기 또는 실패하는 문제가 발생할 수 있습니다.

반대로 로그 파일의 크기가 매우 작은 경우 10%씩 자동 확장하면, 증가되는 크기가 작아서 자동 증가가 빈번하게 발생하게 됩니다. 로그 파일의 경우에 작은 크기의 자동 확장이 빈번하게 발생하면 여러 개의 작은 가상 로그 파일(VLF)들로 단편화가 발생하게 되어 성능을 저하시킬 수 있으며, 이로 인하여 데이터베이스 시작뿐 아니라 로그 백업 및 복원 작업이 느려질 수 있습니다. 저자의 경험에 비추어 보면 데이터베이스 파일의 초기 크기를 적절하게 설정하는 것도 중요하지만, 만일의 경우에 발생할 자동 확장이 적절한 크기로 적절한 시간 안에 완료될 수 있도록 자동 증가 값을 %가 아닌 MB 단위로 지정하는 것을 권고합니다.

가상 로그 파일의 수가 지나치게 많은 경우에는 가상 로그 파일의 수를 줄이는 작업을 수행하는 것이 좋습니다. 작업 방법은 [트랜잭션 로그 파일 축소]를 참조하십시오.

[따라하기] 데이터베이스의 데이터 파일 증가율을 100MB로 변경하기

```
ALTER DATABASE Sample
MODIFY FILE
  (NAME = Sample_dat,
  FILEGROWTH = 100MB);
GO
```

■ tempdb는 I/O가 빠른 디스크 드라이브에 배치할 것을 권고합니다.

tempdb는 I/O가 빠른 쪽에 배치하는 것이 성능을 위해 좋습니다. 성능을 위하여 tempdb를 여러 디스크에 스트라이핑하거나 자주 사용되는 사용자 데이터베이스와 물리적으로 격리된 디스크에 배치하는 방안도 고려해 볼 수 있습니다. 특히 tempdb를 매우 많이 사용하는 대규모 시스템이라면 tempdb를 별도의 디스크 셋에 배치하면 성능 향상 효과를 얻을 수 있습니다.

[참고] 데이터베이스 파일 이외의 파일 배치

SQL Server 시스템의 운영 체제 드라이브는 RAID 1로 구성하며, 페이징 파일은 데이터베이스의 데이터, 로그, tempdb가 있는 곳에는 배치하지 않아야 합니다. 이렇게 구성함으로써 디스크 오류에 신속하게 대응하여 복구할 수 있습니다. 이 때, 부트 디스크는 미러를 이용하여 부팅 가능해야 합니다. 시스템 백업을 동일 서버에 저장해야 한다면 데이터 파일이나 로그 파일이 없는 별도의 디스크에 저장합니다.

[참고] 데이터베이스 생성이 실패하는 경우 문제 해결하기

데이터베이스 생성이 실패하는 원인에는 여러 가지가 있지만 주로 다음과 같은 문제로 인하여 새로운 데이터베이스의 생성이 실패합니다.

- *model 데이터베이스가 사용 중인 경우*
  model 데이터베이스에 대한 배타적 잠금을 확보하지 못하면 새로운 데이터베이스를 만들 수 없습니다. model 데이터베이스를 사용하는 프로세스의 수행이 완료되기를 기다렸다가 수행이 완료된 다음에 데이터베이스를 생성하거나, model 데이터베이스를 사용중인 프로세스를 강제로 중지한 후에 데이터베이스 생성 작업을 재시도합니다.

- *데이터베이스 파일의 물리적인 위치를 잘못 지정한 경우*
  CREATE DATABASE 문에서 지정한 폴더가 실제로 존재하는 폴더인지 확인합니다. 만일 지정한 폴더가 없다면 폴더를 만들어 주고 재시도합니다.

- *디스크 공간이 부족한 경우*
  지정한 드라이브에 지정한 크기로 물리적 파일을 생성하는데 충분한 여유 공간이 있는지 확인합니다.

- *CREATE DATABASE 문을 실행한 사용자에게 새로운 데이터베이스를 생성할 수 있는 권한이 없는 경우*
  새로운 데이터베이스를 생성하기 위해서는 sysadmin 역할 또는 dbcreator 역할의 구성원이어야 합니다. 이 역할의 구성원에게 작업을 요청하거나, 주기적으로 작업이 필요하다면 해당 사용자를 dbcreator 역할의 구성원으로 등록하면 됩니다.

- *동일한 이름의 데이터베이스가 이미 존재하는 경우*
  sp_helpdb를 수행하거나 sys.databases 카탈로그 뷰를 참조하여 확인합니다.
  만약 이미 존재하는 데이터베이스가 불필요하다면 ALTER DATABASE 문을 사용하여 기존 데이터베이스를 다른 이름으로 변경하거나 DROP DATABASE 문을 사용하여 동일한 이름의 데이터베이스를 삭제한 후에 재시도합니다.

- *동일한 이름의 파일이 이미 존재하는 경우*
  존재하지 않는 파일 이름을 지정하고 재시도합니다.

- *트랜잭션 내에서 CREATE DATABASE 문을 실행한 경우*
  CREATE DATABASE 문은 자동 커밋 모드(기본 트랜잭션 관리 모드)에서 실행해야 하며 명시적 또는 암시적 트랜잭션에서는 허용되지 않습니다.

파일 그룹

파일 그룹은 데이터베이스의 파일을 그룹화하는 관리 메커니즘으로, 데이터베이스 파일과 별개로 만들 수 없습니다. 파일 그룹은 데이터베이스를 처음 만들 때 만들거나 나중에 데이터베이스에 파일이 추가될 때 만들 수 있습니다. 그러나, 데이터베이스 파일을 다른 파일 그룹으로 이동할 수 없으며, 한 파일 그룹에만 속할 수 있습니다. 의미 없는 숫자이기는 하지만 각 데이터베이스에 최대 32,767개의 파일 그룹을 만들 수 있으며, 파일 그룹에는 데이터 파일만 포함할 수 있으며 트랜잭션 로그 파일은 포함할 수 없습니다.

[따라하기] 기본 파일 그룹 확인하기

```
USE Sample;
GO
-- 기본 파일 그룹이 어떻게 설정되어 있는지 확인합니다.
SELECT * FROM sys.filegroups WHERE is_default = 1;
GO
-- 'Primary' 파일 그룹이 기본 파일 그룹인지를 확인합니다.
-- 결과값이 1이 반환되면 기본 파일 그룹이고, 0이 반환되면 기본 파일 그룹이
아닙니다.
SELECT FILEGROUPPROPERTY('Primary', 'IsDefault');
GO
```

[참고] 파일 그룹 활용 시나리오 예제

1. 특정 테이블이나 인덱스를 특정 디스크 또는 특정 디스크 어레이에 배치하도록 개체를 저장할 파일 그룹을 지정할 수 있습니다.

2. 기본적으로 인덱스는 테이블이 저장된 파일 그룹에 만들어집니다. 성능 향상을 위하여 데이터와 비클러스터형 인덱스를 서로 다른 디스크에 배치하기 위하여 파일 그룹을 활용할 수 있습니다.

3. 매우 빈번하게 다른 테이블과 조인되어 참조되는 주요 테이블을 별도의 디스크에 배치함으로써 병렬 I/O를 유도하는 경우도 있습니다.

4. 특정 테이블에 일정 공간만 할당해야 하는 경우에 그 테이블을 단독으로 별도의 파일 그룹에 배치하고 그 파일 그룹에 속하는 파일의 크기와 최대 크기를 원하는 값으로 설정하면 됩니다.

5. 액세스 유형이나 작업 부하를 고려하여 다른 테이블들을 별개의 디스크 어레이로 분리하여 배치하기를 원하는 경우에 활용할 수 있습니다. 예를 들어, 네 개의 디스크로 구성된 어레이와 두 개의 디스크로 구성된 어레이가 있고 Table1과 Table2 두 개의 테이블이 있다고 가정합니다. Table1은 읽기 전용 데이터로서 매우 자주 액세스되며 주로 순차적으로 액세스되고, Table2는 업데이트가 주로 발생하고 Table1만큼 자주 액세스되지는 않습니다. Table1은 네 개의 디스크로 구성된 어레이에 배치하고 Table1는 두 개의 디스크로 구성된 어레이에 분리하여 배치하면, 더 많은 디스크가 Table1에 대한 높은 빈도의 읽기 작업을 수행할 수 있게 되며 Table2에 대한 쓰기 작업 때문에 방해를 받지 않기 때문에 순차적인 액세스를 유지하게 됩니다. 또한 두 개의 디스크는 Table2에 대한 업데이트 전용으로 사용할 수 있게 됩니다. 두 개의 파일 그룹을 만들고 원하는 파일 그룹에 테이블을 만들면 됩니다.

6. 수정이 발생하면 안되는 테이블을 별도의 파일 그룹에 저장하고 그 파일 그룹만 읽기 전용으로 설정하면 테이블이 실수로 업데이트되는 것을 방지할 수 있습니다.

[따라하기] 파일 그룹 읽기 전용 설정하기

```
ALTER DATABASE Sample
        MODIFY FILEGROUP SamplesFG2 READONLY;
GO
```

[따라하기] 파일 그룹 및 파일 추가하기

Sample 데이터베이스에 Samples_FG3 파일 그룹을 추가하고, 추가한 파일 그룹에 Sample_New 파일을 추가하는 예제입니다. 참고로, 보조 데이터 파일의 확장자는 .ndf를 사용하는 것을 권고합니다.

```
ALTER DATABASE Sample
ADD FILEGROUP Samples_FG3;
GO
ALTER DATABASE Sample
ADD FILE (
    NAME = Sample_New,
    FILENAME = 'e:\DBdata\Sample_New.ndf',
    SIZE = 10MB,
    MAXSIZE = 1GB,
    FILEGROWTH = 3MB)
TO FILEGROUP Samples_FG3;
GO
```

데이터베이스 삭제

DROP DATABASE 문을 사용하거나 SQL Server Management Studio를 사용하면 SQL Server에서 데이터베이스를 삭제할 수 있습니다.

다음은 데이터베이스 삭제 시에 유의해야 할 사항입니다.

■ 현재 다른 사용자가 연결되어 있는 데이터베이스는 삭제할 수 없습니다. 데이터베이스에서 사용자들의 연결을 강제로 해제하려면 ALTER DATABASE 문을 사용하여 데이터베이스를 SINGLE_USER로 설정할 수 있습니다.

■ 시스템 데이터베이스는 삭제할 수 없습니다.

■ DROP DATABASE를 수행하면 데이터베이스가 사용하는 물리적 파일도 삭제됩니다. 만약 삭제된 데이터베이스를 다시 복구하고자 하는 경우에는 백업본을 복원해야 합니다. 그러므로, 다시 참조할 필요가 있는 데이터베이스를 삭제하고자 하는 경우에는 sp_detach_db를 사용하여 SQL Server에서 데이터베이스 정보만 삭제하고 파일들은 디스크에 남겨 둘 것을 권고합니다.

■ 삭제 시 데이터베이스 또는 해당 데이터베이스의 파일 중 하나가 오프라인 상태이면 파일은 삭제되지 않습니다. 이 파일은 Windows 탐색기를 사용해 수동으로 삭제할 수 있습니다.

■ 데이터베이스를 삭제하려면 먼저 데이터베이스의 모든 데이터베이스 스냅숏을 삭제해야 합니다.

■ 데이터베이스에서 로그 전달 작업을 수행하고 있는 경우라면 데이터베이스를 삭제하기 전에 로그 전달을 제거합니다. 자세한 내용은 [로그 전달]을 참조하십시오.

■ 데이터베이스는 오프라인, 읽기 전용, 주의 대상과 같은 비정상적인 상태가 되어도 삭제할 수 있습니다. sys.databases 카탈로그 뷰를 사용하면 데이터베이스의 현재 상태를 확인할 수 있습니다.

■ 삭제된 데이터베이스는 백업 복원을 통해서만 다시 만들 수 있습니다. 데이터베이스 스냅숏은 백업할 수 없으므로 복원할 수 없습니다.

■ 트랜잭션 복제를 위해 게시된 데이터베이스, 병합 복제를 위해 게시 또는 구독된 데이터베이스를 삭제하려면 먼저 데이터베이스에서 복제를 제거해야 합니다. 데이터베이스가 손상되었거나 복제를 먼저 제거할 수 없거나 또는 두 가지 모두 해당되는 경우, 대부분 ALTER DATABASE를 사용하여 데이터베이스를 오프라인으로 설정한 뒤 삭제하는 방식으로 데이터베이스를 삭제할 수 있습니다.

■ 데이터베이스를 삭제하면 master 데이터베이스의 시스템 테이블이 업데이트되므로 데이터베이스가 삭제된 후에는 master 데이터베이스를 백업할 것을 권고합니다. master 데이터베이스를 복원할 필요가 있을 때, 마지막 master 백업 이후 삭제된 데이터베이스 정보가 시스템 테이블에 남아 있으면 그로 인하여 오류가 발생할 수 있습니다.

[참고] 데이터베이스 삭제가 실패하는 경우 문제 해결하기

데이터베이스 생성이 실패하는 원인에는 여러 가지가 있지만 주로 다음과 같은 문제로 인하여 새로운 데이터베이스의 생성이 실패합니다.

- ***자신이 삭제하고자 하는 데이터베이스에 연결하고 있는 경우***
  스크립트로 삭제하는 경우에는 DROP DATABASE 문을 실행하기 전에 다른 데이터베이스로 연결을 변경해야 합니다. 또한 자신이 SQL Server Management Studio 나 다른 창에서 그 데이터베이스에 연결을 맺은 상태는 아닌지 확인하고, 만일 있다면 닫거나 연결을 해제합니다.

- ***다른 사용자가 삭제하고자 하는 데이터베이스에 연결을 하고 있는 경우***
  다른 사용자가 연결을 하고 있는 경우에는 먼저 연결을 해제하도록 요청하고 연결 해제가 원활하게 이루어지지 않고 신속한 작업이 필요한 경우에는 강제로 데이터베이스를 단일 사용자 모드로 변경합니다.

- ***데이터베이스가 손상되었거나 복제를 먼저 제거할 수 없는 경우***
  ALTER DATABASE 문을 사용하여 데이터베이스를 오프라인으로 설정한 뒤 삭제합니다.

- ***삭제하고자 하는 데이터베이스를 다른 프로세스에서 연결 중인 경우***
  데이터베이스 사용 중이어서 삭제할 수 없는 경우에는, 데이터베이스를 사용하는 프로세스들이 완료되기를 기다렸다가 삭제합니다. 그런데 만일 프로세스들이 연결을 해제하거나 쿼리 실행이 완료되기를 기다릴 수 없는 상황이라면 완료되지 않은 트랜잭션을 롤백하고 데이터베이스를 단일 사용자 모드로 변경한 후에 재시도합니다. 단일 사용자 모드로 변경하기 위하여 ALTER DATABASE 문을 사용할 때 WITH ROLLBACK 절을 기술하지 않으면 대상 데이터베이스에 잠금이 있는 경우 ALTER DATABASE 문이 무기한 기다립니다.

[따라하기] 30초후, 완료되지 않은 트랜잭션을 롤백하고 단일 사용자 모드로 변경하기

```
USE Test;
GO
ALTER DATABASE Test
SET SINGLE_USER
WITH ROLLBACK AFTER 30;  -- 30초가 경과한 후에 롤백
GO
USE master;
GO
DROP DATABASE Test;
GO
```

데이터베이스 파일 이동

ALTER DATABASE 문을 사용하여 파일 정보를 변경하거나 데이터베이스 분리 및 연결 시스템 저장 프로시저를 사용하여 데이터베이스를 이동할 수 있습니다.

데이터베이스를 이동할 때는 분리 및 연결 작업보다 ALTER DATABASE를 이용하는 것이 좋습니다. ALTER DATABASE 문의 FILENAME 절에 새 파일 위치를 지정하면 리소스 데이터베이스 파일을 제외한 시스템 데이터베이스와 사용자 데이터베이스의 파일을 이동할 수 있습니다. 이 절차를 수행하려면 ALTER DATABASE 문을 실행할 데이터베이스 파일의 논리 이름을 알고 있어야 하며, sys.master_files 카탈로그 뷰에서 name 열을 조회하면 논리 파일 이름을 확인할 수 있습니다.

[따라하기] ALTER DATABASE 문을 사용하여 사용자 데이터베이스 파일 이동하기

C 드라이브에 주 데이터 파일과 트랜잭션 로그 파일이 있는 데이터베이스를 주 데이터 파일은 D 드라이브, 트랜잭션 로그 파일은 E 드라이브로 이동하는 예제입니다.

데이터베이스명	Sample
데이터 파일명	Sample_dat
변경 전 데이터 파일 위치	C:\DBdata\Sample_dat.mdf
변경 후 데이터 파일 위치	D:\DBdata\Sample_dat.mdf
로그 파일명	Sample_log
변경 전 로그 파일 위치	C:\DBlog\Sample_log.ldf
변경 후 로그 파일 위치	E:\DBlog\Sample_log.ldf

1. 데이터베이스 파일의 논리 이름을 확인합니다.

```
SELECT file_id, name, physical_name FROM sys.master_files
WHERE database_id = db_id('Sample');
GO
```

2. 사용자들에게 공지하여 작업하고자 하는 데이터베이스에 대한 연결을 해제하도록 합니다. 만일 해제되지 않는 연결이 남아 있어서 문제가 되는 경우에는 다음과 같은 방법을 사용하여 연결을 해제할 수 있습니다.

```
USE Sample;
GO
ALTER DATABASE Sample SET SINGLE_USER
WITH ROLLBACK AFTER 60;
GO
```

3. 데이터베이스를 오프라인 상태로 만듭니다.

```
ALTER DATABASE Sample SET OFFLINE;
GO
```

4. 데이터베이스 파일들을 새로운 위치로 이동합니다.

5. 데이터베이스 파일 위치 정보를 변경합니다.

```
ALTER DATABASE Sample MODIFY FILE (
    NAME = Sample_dat,
    FILENAME = 'D:\DBdata\Sample_dat.mdf');
GO
ALTER DATABASE Sample MODIFY FILE (
    NAME = Sample_log,
    FILENAME = 'E:\DBlog\Sample_log.ldf');
GO
```

6. 데이터베이스를 온라인 상태로 변경합니다.

```
ALTER DATABASE Sample SET ONLINE;
GO
```

만일 데이터베이스를 다른 서버로 이동하고자 하는 경우에는 분리 및 연결 작업을 사용합니다.

[따라하기] 분리 및 연결 작업을 사용하여 사용자 데이터베이스 이동하기

1. 해당 데이터베이스에 연결되어 있는 연결을 모두 비 연결 상태로 만들고, 단일 사용자 모드로 설정합니다. 다음은 600초 후에 모든 작업들을 롤백시키고, 연결을 끊는 예제입니다.

```
USE Sample;
```

```
GO
ALTER DATABASE Sample SET SINGLE_USER
WITH ROLLBACK AFTER 600;
GO
```

2. 해당 데이터베이스의 모든 데이터 파일과 트랜잭션 로그 파일의 경로를 확인합니다.

```
EXEC sp_helpdb Sample;
GO
```

3. 데이터베이스와 파일을 분리합니다.

```
USE master;
GO
EXEC sp_detach_db 'Sample', 'true';
GO
```

4. 데이터베이스 파일들을 원하는 위치에 복사합니다.

C 드라이브에 있는 Sample_dat.mdf, Sample_log.ldf를 각각 d:\DBdata, e:\DBlog 폴더로 복사합니다.

5. 새로운 위치의 파일을 지정하여 데이터베이스와 연결합니다.

```
CREATE DATABASE  Sample
ON PRIMARY
(FILENAME='d:\DBdata\Sample_dat.mdf')
LOG ON
(FILENAME='e:\DBlog\Sample_log.ldf')
FOR ATTACH ;
GO
```

[주의]

리소스 데이터베이스 파일을 이동하면 SQL Server 가 시작되지 않으므로 주의합니다. 또한 리소스 데이터베이스는 master 데이터베이스와 동일한 위치에 존재해야 하므로, master 데이터베이스를 이동한 경우에는 리소스 데이터베이스도 master 데이터베이스와 동일한 위치로 이동해야 합니다.

데이터베이스 축소

DBCC SHRINKDATABASE 또는 DBCC SHRINKFILE를 사용하여 수동으로 데이터베이스를 축소할 수 있습니다. 데이터베이스의 특정 데이터 파일이나, 트랜잭션 로그 파일의 크기를 축소하는 경우에는 DBCC SHRINKFILE을 사용합니다. 데이터베이스의 AUTO_SHRINK 옵션을 TRUE로 설정하면 파일에서 사용되지 않는 공간이 25% 이상이 되면 파일이 자동으로 축소됩니다. 그렇지만, 축소가 필요할 때에는 DBCC SHRINKDATABASE 문 또는 DBCC SHRINKFILE 문을 사용하여 수동으로 데이터베이스를 적절한 크기로 축소하는 것을 권고합니다.

■ 파일을 지정하지 않고 특정 데이터베이스의 모든 데이터와 로그 파일을 축소하고자 하는 경우에는 DBCC SHRINKDATABASE를 사용합니다.

[따라하기] 데이터베이스 전체 크기 중10% 여유공간이 남도록 데이터베이스 축소하기

```
DBCC SHRINKDATABASE (Sample, 10);
GO
```

■ 특정 데이터베이스의 특정 파일을 축소하고자 하는 경우에는 DBCC SHRINKFILE을 사용합니다. DBCC SHRINKFILE 명령어를 사용하면 특정 파일의 크기만 축소할 수 있으며, 데이터 파일이나 로그 파일을 초기 크기보다 더 작게 축소하고자 하는 경우에는 DBCC SHRINKFILE 명령어를 사용합니다.

[따라하기] Sample 데이터베이스의 Sample_dat 파일을 10MB로 축소하기

```
USE Sample;
DBCC SHRINKFILE (Sample_dat, 10);
GO
```

■ 트랜잭션 로그 파일 축소하기

DBCC SHRINKDATABASE와 DBCC SHRINKFILE에서 사용할 수 있는 TRUNCATE 옵션은 데이터 파일에만 적용되며 로그 파일에는 적용되지 않습니다. 로그 파일에 대하여 이 옵션을 사용한다고 해서 로그가 삭제되는 것은 아닙니다. 로그 파일은 즉시 크기가 줄어들지 않으며 트랜잭션 로그를 백업하거나 삭제할 때 크기가 줄어듭니다. 트랜잭션 로그 파일의 크기 축소는 가상 로그 파일의 크기 단위로 이루어집니다. 만일 로그 파일의 공간이 부족하여 빈번하게 자동 증가가 발생하면, 로그 파일이 많은 수의 가상 로그 파일들로 조각화되어 데이터베이스 시작뿐 아니라 로그 백업 및 복원 작업이 느려질 수 있다는 점을 DBA는 유의해야 합니다. 자동 확장을 예방하기 위해서는 사전에 로그 파일의 크기를 충분한 크기로 만들어 두어야 하며, 만일 이미 가상 로그 파일의 수가 지나치게 많아진 경우에는 가상 로그 파일 수를 줄여 주는 작업을 수행하고 트랜잭션 로그 파일을 적절한 크기로 변경할 것을 권고합니다.

[따라하기] 가상 로그 파일 개수 줄이기

Sample 데이터베이스의 Sample_log 로그 파일의 가상 로그 파일을 제거하여, 트랜잭션 로그 파일을 축소합니다.

1. 가상 로그 파일의 개수를 확인합니다. 결과 행의 수가 가상 로그 파일의 수입니다.

```
USE Sample;
GO
DBCC LOGINFO;
GO
```

2. 트랜잭션 로그 백업을 수행합니다. 로그 백업을 받을 수 없는 경우에는 로그를 잘라냅니다. 로그를 잘라낸다고 해서 실제 로그 파일의 크기가 줄어들지는 않습니다. 그러나 로그를 잘라내면 논리 로그의 크기가 줄어들고 논리 로그 부분을 포함하지 않는 가상 로그가 비활성으로 표시합니다. 로그 축소 작업이 수행되면 요청된 크기만큼 로그 파일이 충분히 축소될 수 있도록 비활성 가상 로그를 제거하게 되므로, 사전에 로그를 백업하거나 잘라내는 것이 좋습니다.

```
BACKUP LOG Sample TO DISK='D:\DBBackup\Sample_Log.bak';
GO
-- 또는
BACKUP LOG Sample WITH NO_LOG;
GO
```

3. 트랜잭션 로그 파일의 크기를 가능한 한 작은 크기로 축소합니다.

```
DBCC SHRINKFILE (Sample_log, TRUNCATEONLY);
GO
```

4. 로그 파일의 크기를 적절한 크기로 변경합니다.

```
ALTER DATABASE Sample
MODIFY FILE
    ( NAME = 'Sample_log'
    , SIZE = 30MB);
GO
```

데이터베이스 옵션 설정

SQL Server Management Studio에서 데이터베이스 옵션을 확인하고 설정하는 것이 가능하지만, SQL 명령어를 사용하여 데이터베이스 옵션을 확인하거나 변경할 경우가 종종 발생합니다.

ALTER DATABASE 문을 사용하면 데이터베이스 옵션을 설정할 수 있습니다. 데이터베이스 옵션을 설정하면 수정 사항이 즉시 반영됩니다. 새로 만들어지는 모든 데이터베이스에 적용되는 데이터베이스 옵션의 기본값을 변경하려면 model 데이터베이스에서 해당 데이터베이스 옵션을 변경합니다.

sp_dboption을 사용하여 데이터베이스의 옵션을 변경하는 것이 가능하지만 sp_dboption은 하위 버전과의 호환을 위하여 지원되는 기능이며, SQL Server 2005 온라인 설명서에 의하면 이 기능은 SQL Server 다음 버전에서 제거된다고 기술되어 있습니다. 그러므로 데이터베이스 옵션을 변경하고자 하는 경우에는 sp_dboption 대신 ALTER DATABASE 문을 사용할 것을 권고합니다. 만일 현재 sp_dboption을 사용하는 응용 프로그램이 있다면 응용 프로그램도 수정하기 바랍니다.

데이터베이스 옵션에 대한 현재 설정을 검색하려면 sys.databases 카탈로그 뷰 또는 DATABASEPROPERTYEX를 사용합니다.

[따라하기] 데이터베이스 복구 모델를 대량 로그 복구 모델로 변경하기

```
USE master;
GO
ALTER DATABASE AdventureWorks
SET RECOVERY BULK_LOGGED;
GO
```

[따라하기] 데이터베이스를 읽기 전용으로 변경하기

```
USE master;
GO
ALTER DATABASE AdventureWorks
SET SINGLE_USER
WITH ROLLBACK IMMEDIATE;
GO
ALTER DATABASE AdventureWorks
SET READ_ONLY;
GO
```

```
ALTER DATABASE AdventureWorks
SET MULTI_USER;
GO
```

[따라하기] 자동 통계 갱신 데이터베이스 옵션 비활성화하기

```
USE master;
GO
ALTER DATABASE Adventureworks
SET AUTO_UPDATE_STATISTICS OFF;
GO
```

[따라하기] 통계 자동 업데이터가 필요한 경우 컴파일 전에 통계가 업데이트되기를 기다리지 않도록 데이터베이스 옵션 설정하기

```
USE master;
GO
ALTER DATABASE Adventureworks
SET AUTO_UPDATE_STATISTICS ON;
GO
ALTER DATABASE Adventureworks
SET AUTO_UPDATE_STATISTICS_ASYNC ON;
GO
```

[따라하기] RCSI 데이터베이스 옵션 설정하기

```
USE master;
GO
ALTER DATABASE Adventureworks
SET READ_COMMITTED_SNAPSHOT ON;
GO
SELECT name, is_read_committed_snapshot_on
FROM sys.databases
WHERE name = N'Adventureworks';
GO
```

[따라하기] 스냅숏 트랜잭션 격리 수준 허용하기

```
USE master;
GO
ALTER DATABASE Adventureworks
SET ALLOW_SNAPSHOT_ISOLATION ON;
GO
SELECT name, snapshot_isolation_state, snapshot_isolation_state_desc AS
description
FROM sys.databases
WHERE name = N'Adventureworks';
GO
```

데이터베이스 소유자 변경

데이터베이스가 만들어질 때 설정된 소유자를 추후 변경할 수 있습니다. master, model 또는 tempdb 시스템 데이터베이스의 소유자는 변경할 수 없습니다.

[따라하기] Sample 데이터베이스의 소유자를 'dbadmin'으로 변경하기

```
USE Sample;
GO
EXEC sp_changedbowner 'dbadmin';
GO
```

데이터베이스 이름 변경

ALTER DATABASE MODIFY NAME 문을 사용하면 데이터베이스의 이름을 변경할 수 있습니다. 다른 사용자가 데이터베이스에 연결하지 않은 상태라면 ALTER DATABASE 문만 수행하면 이름이 변경되지만, 다른 사용자가 데이터베이스에 연결을 맺은 상태에서는 이름 변경이 실패합니다. 이 경우에는 데이터베이스의 이름을 변경하기 전에 데이터베이스를 단일 사용자 모드나 오프라인 모드로 변경한 상태에서 이름을 변경하면 됩니다.

sp_renamedb를 사용하여 데이터베이스의 이름을 변경할 수도 있지만, 온라인 설명서에 의하면 sp_renamedb는 다음 버전의 Microsoft SQL Server에서 제거된다고 기술되어 있습니다. 그러므로 이후로 데이터베이스 이름 변경 스크립트를 작성할 때에는 sp_renamedb 대신 ALTER DATABASE MODIFY NAME을 사용해야 하며, sp_renamedb 를 사용하는 응용 프로그램도 수정하기 바랍니다.

[따라하기] Sample 데이터베이스의 이름을 Sample_Rename으로 변경하기

1. 모든 사용자들에게 Sample 데이터베이스에 대한 연결을 해제하도록 통보합니다. SQL Server Management Studio에서 해당 데이터베이스에 연결하고 있는 상황이 발생하지 않도록 SQL Server Management Studio를 닫습니다. 만일 데이터베이스 이름을 신속하게 변경해야 하는데 사용자 연결이 계속 해제되지 않는 상태라면 사용자 프로세스들을 강제로 종료시킬 수 있습니다.

2. 데이터베이스를 단일 사용자 모드로 설정합니다.
```
USE master
ALTER DATABASE Sample SET SINGLE_USER;
GO
```

3. 데이터베이스 이름을 변경합니다.
```
ALTER DATABASE Sample MODIFY NAME = Sample_Rename;
GO
```

4. 데이터베이스를 다중 사용자 모드로 원복합니다. 데이터베이스를 단일 사용자 모드로 그대로 둠으로써 응용프로그램에서 오류가 발생하는 경우가 종종 있습니다.
```
ALTER DATABASE Sample_Rename SET MULTI_USER;
GO
```

5. 변경된 데이터베이스를 사용하는데 문제가 없는지 확인합니다.

데이터베이스 무결성 체크

지정한 데이터베이스에서 모든 개체의 할당과 구조적, 논리적 무결성을 검사합니다.

[구문]

```
DBCC CHECKDB
[
(
      'database_name' | database_id | 0
   [ , NOINDEX
   | { REPAIR_ALLOW_DATA_LOSS
   | REPAIR_FAST
   | REPAIR_REBUILD
   } ]
)
]
   [ WITH       {
         [ ALL_ERRORMSGS ]
         [ , [ NO_INFOMSGS ] ]
         [ , [ TABLOCK ] ]
         [ , [ ESTIMATEONLY ] ]
         [ , [ PHYSICAL_ONLY ] ] | [ , [ DATA_PURITY ] ]
      }
   ]
```

DBCC 기능을 사용하여 SQL Server의 상태를 확인할 수 있습니다.
DBCC CHECKDB는 디스크에서 메모리로 할당된 각 페이지를 읽어야 확인할 수 있습니다.
시스템에 작업이 많은 경우 DBCC CHECKDB를 실행하면 다음 두 가지 이유 때문에 DBCC 성능이 저하될 수 있습니다.

첫 번째 이유는 사용 가능한 메모리가 부족하여 SQL Server 데이터베이스 엔진이 DBCC CHECKDB의 내부 데이터 중 일부를 tempdb 데이터베이스로 스풀링하기 때문입니다. tempdb 데이터베이스는 디스크에 위치하므로 데이터가 디스크에서 기록될 때 I/O 작업의 병목 상태로 인해 성능이 저하됩니다. 두 번째 이유는 DBCC CHECKDB가 디스크에서 데이터를 읽는 방식을 최적화하려고 하기 때문입니다. 또한 동일한 디스크를 사용하는 작업이 집중되는 경우 최적화가 상당히 저하되어 실행 속도가 느려지게 됩니다. 그러므로, DBCC CHECKDB는 운영 서버에서 서비스 중에 실행하지 않을 것을 권고합니다. 데이터베이스를 테스트 서버에 복원하여 DBCC CHECKDB를 실행하면 서비스에 지장없이 DBCC CHECKDB를 실행할 수 있습니다.

SQL Server 2000에서는 DBCC CHECKDB를 실행하면 기본적으로 테이블 레벨의 스키마 잠금을 사용하여 그로 인한 블로킹이 발생하고 로그 정보를 읽기 때문에 트랜잭션 로그 잘라내기 작업이 블로킹될 수 있었습니다. SQL Server 2005는 데이터베이스 엔진에서 만든 내부 읽기 전용 데이터베이스 스냅숏에서 작동하기 때문에 이 명령이 실행될 때 블로킹(차단) 및 동시성 문제를 방지할 수 있습니다.

DBCC CHECKDB가 내부적으로 수행하는 작업을 간략히 정리하면 다음과 같습니다. 즉, DBCC CHECKDB를 실행하면 DBCC CHECKALLOC, DBCC CHECKCATALOG, DBCC CHECKTABLE이 자동으로 실행되므로 별도로 실행할 필요가 없습니다. 그렇지만 DBCC CHECKDB는 비활성화된 인덱스는 검사하지 않습니다.

1. 데이터베이스에 대해 DBCC CHECKALLOC을 실행합니다.
2. 데이터베이스의 모든 테이블 및 뷰에 대해 DBCC CHECKTABLE을 실행합니다.
3. 데이터베이스에 있는 Service Broker 데이터의 유효성을 검사합니다.
4. 데이터베이스에 대해 DBCC CHECKCATALOG를 실행합니다.
5. 데이터베이스에 있는 모든 인덱싱된 뷰의 내용에 대한 유효성을 검사합니다.

[참고] DBCC 명령에 대한 진행률 보고

SQL Server 2005의 sys.dm_exec_requests 카탈로그 뷰를 조회하면 DBCC CHECKDB, DBCC CHECKFILEGROUP, DBCC CHECKTABLE 명령의 현재 실행 단계와 진행률에 대한 정보를 확인할 수 있습니다. percent_complete 열은 명령의 완료 비율을 나타내고 command 열은 명령의 현재 실행 단계를 나타냅니다.

[따라하기] Sample 데이터베이스 내 모든 개체의 할당과 구조적 무결성 검사

```
DBCC CHECKDB ('Sample');
GO
```

[따라하기] 데이터베이스가 SUSPECT 모드인 경우 DBCC CHECKDB 실행하기

데이터 파일 또는 로그 파일이 손상되어 데이터베이스가 SUSPECT 상태인 경우,
데이터베이스를 EMERGENCY 모드로 설정하고 DBCC CHECKDB를 실행합니다.

1. 데이터베이스를 EMERGENCY 모드로 설정합니다.

```
ALTER DATABASE Sample SET EMERGENCY;
GO
```

2. 데이터베이스를 SINGLE_USER 모드로 설정합니다.

```
ALTER DATABASE Sample SET SINGLE_USER;
GO
```

3. 보고된 모든 오류를 복구합니다. 이러한 복구를 수행하면 일부 데이터가 손실될 수 있으므로 유의합니다

```
DBCC CHECKDB (Sample, REPAIR_ALLOW_DATA_LOSS)
GO
```

4. 데이터베이스를 MULTI_USER 모드로 설정합니다.

```
ALTER DATABASE Sample SET MULTI_USER;
GO
```

백업과 복원

SQL Server 2005에서는 고성능 백업 및 복원 기능을 제공합니다. SQL Server 백업 및 복원 구성 요소는 SQL Server 데이터베이스에 저장된 중요한 데이터를 보호하는데 필수적인 보호 기능을 제공합니다. 백업 및 복원 전략을 적절하게 계획하여 구현하면 다양한 오류로 인한 손상으로 인하여 데이터베이스의 데이터가 유실되는 것을 방지할 수 있습니다. 일련의 백업 복원과 데이터베이스 복구 전략을 사전에 충분히 테스트하여 재해에 효과적으로 대처할 수 있도록 대비하는 것이 필요합니다.

복구 모델

복구 모델은 데이터베이스의 백업 및 복원 작업의 기본 동작을 제어하는 데이터베이스 속성입니다. 데이터베이스의 복구 모델에 따라 트랜잭션의 로깅 방법, 트랜잭션 로그에 백업이 필요한지 여부 및 적용 가능한 복원 작업의 종류가 다릅니다. 새로 만들어지는 데이터베이스는 model 데이터베이스에서 복구 모델을 상속 받습니다.

복구 모델의 종류에는 단순 복구 모델(Simple), 전체 복구 모델(Full), 대량 로그 복구 모델(Bulk_Logged)이 있습니다. SQL Server Standard Edition 및 SQL Server Enterprise Edition의 복구 모델의 기본값은 전체 복구 모델입니다.

■ 단순 복구 모델(Simple)

- 마지막으로 백업을 시행한 시점까지의 백업된 정보를 복원합니다.
- 전체 백업과 차등 백업을 이용할 수 있으나, 트랜잭션 로그 백업은 할 수 없습니다.
- 이 모델은 응용 프로그램을 테스트하는 테스트 환경 또는 저장된 데이터를 복원할 필요가 전혀 없는 시스템에 적합합니다.

■ **전체 복구 모델(Full)**

- 모든 변경 사항이 트랜잭션 로그에 기록됩니다.
- 문제가 발생한 시점이나 과거의 백업을 받은 특정한 시점까지의 정보를 복원할 수 있습니다.
- 전체 백업, 차등 백업, 트랜잭션 로그 백업을 모두 이용할 수 있습니다.

■ **대량 로그 복구 모델(Bulk_Logged)**

- 대량 작업이나 대량 로딩에 대한 기록은 최소화하기 때문에, 백업 전에 발생한 대량 작업의 오류는 수작업으로 보정해야 합니다.
- 전체 백업, 차등 백업, 트랜잭션 로그 백업을 모두 이용할 수 있습니다.

[따라하기] 데이터베이스 복구 모델 변경하기

```
-- 대량 로그 복구 모델로 변경하기
ALTER DATABASE Sample
SET RECOVERY BULK_LOGGED;
GO
-- 단순 복구 모델로 변경하기
ALTER DATABASE Sample
SET RECOVERY SIMPLE;
GO
-- 전체 복구 모델로 변경하기
ALTER DATABASE Sample
SET RECOVERY FULL;
GO
```

[참고] 복구 모델 전환 시 백업 전략

변경 전	변경 후	작업	설명
전체 복구	대량 복구	없음	백업 전략의 변화는 없다.
전체 복구	단순 복구	변경하기 전에 선택적으로 트랜잭션 로그를 백업한다.	변경 시점까지 복원을 위해 변경 전에 로그 백업을 한다. 단순 복구 모델로 전환한 후에는, 로그 백업을 중지한다.
대량 복구	전체 복구	없음	백업 전략의 변화는 없다.
대량 복구	단순 복구	변경하기 전에 트랜잭션 로그를 선택적으로 백업한다.	변경 작업 전에 로그 백업을 하는 것으로 특정 시점까지 복원하는 것이 가능하다. 단순 복구 모델로 변경 후에는 로그 백업을 중지한다.
단순 복구	전체 복구	변경 후에 데이터베이스 백업을 수행한다.	전체 복구 모델로 전환된 후 전체 데이터베이스 백업 또는 차등 백업을 수행한다. 주기적으로 데이터베이스 백업, 로그 백업, (선택적으로) 차등 백업을 수행한다.
단순 복구	대량 복구	변경 후에 데이터베이스 백업을 한다.	대량 복구 모델로 전환된 후 전체 데이터베이스 백업 또는 차등 백업을 수행한다. 주기적으로 데이터베이스 백업, 로그 백업, (선택적으로) 차등 백업을 수행한다.

백업 개요

■ 단순 복구 모델에서의 백업

- 데이터 백업 유형
  - 전체 백업, 부분 백업, 파일 백업, 복사 전용 백업
- 차등 백업 유형
  - 전체 차등 백업, 부분 차등 백업, 파일 차등 백업, 복사 전용 차등 백업

■ 전체 복구 모델에서의 백업

- 데이터 백업 유형
  - 전체 백업, 부분 백업, 파일 백업, 복사 전용 백업
- 차등 백업 유형
  - 전체 차등 백업, 부분 차등 백업, 파일 차등 백업, 복사 전용 차등 백업
- 트랜잭션 로그 백업
  - 기본 로그 백업, 대량 로그 백업, 비상 로그 백업

■ 대량 로그 복구 모델에서의 백업

일반적으로 대량 로그 복구 모델은 전체 복구 모델과 비슷하므로 전체 복구 모델에 대한 정보는 대량 로그 복구 모델에도 적용됩니다.

백업 만들기

■ 데이터베이스 백업

데이터베이스 백업에는 전체 백업과 전체 차등 백업이 있습니다. 전체 백업(이전 이름은 데이터베이스 백업)과 전체 차등 백업은 복구시 사용할 수 있도록 트랜잭션 로그의 일부를 포함합니다. 전체 백업과 전체 차등 백업은 복구 모델에 관계없이 모든 데이터베이스에서 사용할 수 있습니다.

- 전체 백업은 데이터베이스에 있는 모든 데이터를 포함하며 전체 차등 백업의 기반이 되는 기준 백업으로 사용할 수 있습니다.

[구문]

BACKUP DATABASE *〈database_name〉*
TO *〈backup_device〉* …

- 전체 차등 백업은 이전 전체 백업 이후에 변경된 데이터만을 기록합니다. 따라서 전체 차등 백업은 전체 백업보다 작고 빠르기 때문에 그만큼 더 자주 백업할 수 있으며, 데이터 손실의 위험성을 줄일 수 있습니다.

[구문]

BACKUP DATABASE *〈database_name〉*
TO *〈backup_device〉*
WITH DIFFERENTIAL …

[참고] 전체 및 대량 로그 복구 모델에서 로그 백업하기
전체 및 대량 로그 복구를 사용하는 데이터베이스의 경우에는 어떤 종류의 데이터베이스 백업을 사용하더라도 정기적인 트랜잭션 로그 백업을 수행해야 합니다. 트랜잭션 로그 백업을 자주 수행하면 데이터 손실 위험을 줄일 수 있으며 로그를 잘라내므로 트랜잭션 로그가 가득차게 될 가능성도 낮아집니다.

■ 부분 백업 및 부분 차등 백업

부분 및 부분 차등 백업은 모든 복구 모델에서 지원되며, 단순 복구 모델에서 백업하는 데 많은 유연성을 제공하도록 디자인되었습니다.

- 부분 백업은 주 파일 그룹, 모든 읽기/쓰기 파일 그룹 및 선택적으로 지정된 파일의 모든 데이터를 포함합니다. 부분 백업은 마지막 백업 이후부터 읽기 전용이었던 하나 이상의 읽기 전용 파일 그룹이 데이터베이스 포함되어 있을 때 유용합니다. 읽기 전용 데이터베이스의 부분 백업은 주 파일 그룹만 포함합니다.

[구문]

BACKUP DATABASE *〈database_name〉* READ_WRITE_FILEGROUPS
TO *〈backup_device〉* ...

- 부분 차등 백업은 이전 부분 백업 이후에 파일 그룹에 변경된 데이터만 기록합니다. 부분 차등 백업은 부분 백업보다 작고 빠르기 때문에 자주 백업을 수행하여 데이터 손실의 위험을 줄일 수 있습니다.

[구문]

BACKUP DATABASE 〈database_name〉 READ_WRITE_FILEGROUPS
TO 〈backup_device〉
WITH DIFFERENTIAL ...

■ 파일 및 파일 그룹 백업

데이터베이스의 파일을 개별적으로 백업하고 복원할 수 있습니다. 파일 백업을 사용하면 데이터베이스의 나머지 부분을 복원하지 않고 손상된 파일만 복원할 수 있으므로 복구 속도를 증가시킬 수 있습니다. 예를 들어 데이터베이스가 서로 다른 디스크에 저장된 여러 개의 파일로 구성되어 있고 한 디스크에 오류가 있으면 오류가 있는 디스크의 파일만 복원하면 됩니다.

이 항목은 파일 그룹을 여러 개 포함하고 있는 데이터베이스에만 해당됩니다. 단순 복구 모델에서 파일 백업은 읽기 전용 파일 그룹에만 사용할 수 있고 읽기-쓰기 파일 그룹은 주 파일 그룹으로만 백업할 수 있습니다. 단순 복구 모델 데이터베이스에서 읽고 쓰기가 가능한 파일 그룹의 파일 백업을 만들 수는 있지만 파일 그룹을 읽기 전용으로 만든 다음 차등 파일 백업을 가져오지 않으면 복원 시 이 파일 백업을 사용할 수 없습니다.

- 파일 백업은 하나 이상의 파일 또는 파일 그룹에 있는 모든 데이터의 백업입니다.

[구문]

BACKUP DATABASE (*〈database_name〉*) *〈file_or_filegroup〉* [,...n] TO *〈backup_device〉* ...

- 각 파일에 대한 가장 최근의 전체 백업 이후 변경된 데이터를 포함하는 하나 이상의 파일에 대한 백업입니다.

[구문]

BACKUP DATABASE *〈database_name〉* *〈file_or_filegroup〉* [,...n]
TO *〈backup_device〉*
WITH DIFFERENTIAL ...

[참고]
IsReadOnly 속성은 개별 파일이 아닌 파일 그룹에 설정됩니다. 파일 그룹이 읽기 전용인 경우, 즉 파일 그룹의 IsReadOnly 속성이 true인 경우에는 해당 파일 그룹의 모든 파일이 읽기 전용입니다.

중요
같은 데이터베이스에서 데이터베이스 차등 백업과 파일 차등 백업을 함께 사용하지 마십시오.

■ 복사 전용 백업

일반적으로 데이터 백업은 백업 이후 수행되는 하나 이상의 차등 백업을 위한 기준 백업입니다. 그러나, 복사 전용 백업은 다른 백업과 달리 백업과 복원 시퀀스에 영향을 주지 않고 데이터베이스를 백업할 수 있습니다. 모든 백업 유형에 대해 복사 전용 백업을 만들 수 있습니다.

• 복사 전용 데이터 백업

복사 전용 데이터 백업이나 차등 백업을 만들려면 BACKUP DATABASE 문에 COPY_ONLY 옵션을 사용합니다. COPY_ONLY 옵션을 사용하여 수행된 데이터 백업은 기준 백업으로 사용될 수 없으며 기존의 모든 차등 백업에 영향을 주지 않습니다.

정기적인 데이터 백업 또는 차등 백업과 달리 복사 전용 백업은 트랜잭션 로그를 자르지 않습니다.

[구문]

```
BACKUP { DATABASE | LOG } 〈database_name〉 ...
WITH COPY_ONLY …
```

• 복사 전용 차등 백업

일반 차등 백업과 동일합니다.

■ 트랜잭션 로그 백업

트랜잭션 로그 백업(로그 백업이라고도 함)에는 이전 로그 백업에서 백업되지 않은 모든 로그 레코드가 포함됩니다. 이러한 로그 레코드는 전체 및 대량 로그 복구 모델에만 존재합니다. 정기적으로 트랜잭션 로그 백업을 하는 것이 백업 전략의 필수 구성 요소입니다. 로그 백업은 트랜잭션을 복원할 수 있도록 해주는 것과는 별도로 로그를 잘라 로그 파일에서 백업된 로그 레코드를 제거합니다. 로그를 자주 백업하지 않으면 로그 파일이 가득 찰 수 있습니다.

트랜잭션 로그 백업의 모든 시퀀스에 앞서 전체 백업 또는 전체 차등 백업이 선행되어야 합니다. 첫 번째 전체 백업 후에 로그를 백업할 수 있으며 이후에는 전체 백업 진행 중에도 백업할 수 있습니다.

로그 백업에는 기본, 대량 및 비상 로그 백업의 3가지 유형이 있습니다.

• 기본 로그 백업
대량 로그 복구 모델에서 대량의 내용이 변경되지 않은 간격에 대한 트랜잭션 로그 레코드만 포함하는 백업입니다.

• 대량 로그 백업
대량 작업에서 변경한 데이터 페이지와 로그 레코드가 포함된 백업입니다. 대량 로그 백업에서 지정 시간 복구는 사용할 수 없습니다.

• 비상 로그 백업
아직 백업되지 않은 로그 레코드를 캡처하기 위해 손상 가능성이 있는 데이터베이스에서 수행되는 로그 백업입니다. 비상 로그 백업은 오류 발생 후 작업 손실을 방지하기 위해 수행되고 기본 로그 또는 대량 로그 레코드를 포함할 수 있습니다.
대부분의 경우 SQL Server 2005에서는 전체 또는 대량 로그 복구 모델에서 현재 서버 인스턴스에 연결된 데이터베이스를 복원하기 전에 비상 로그 백업을 만들어야 합니다.
비상 로그 백업은 아직 백업되지 않은 로그(비상 로그)를 캡처하며 복구 계획에 포함된 마지막 백업입니다. RESTORE 문에 WITH REPLACE나 WITH STOPAT 절이 포함되어 있지 않은 한 비상 로그 백업을 먼저 수행하지 않고 데이터베이스를 복원하면 오류가 발생합니다

[참고]
Microsoft SQL Server 2005에서는 전체 백업이 실행되는 동안 로그 백업을 수행할 수 있습니다.

[참고]
SQL Server 7.0 및 SQL Server 2000에서는 파일 백업과 파일 차등 백업에 로그 레코드가 포함되지 않으므로 해당 데이터를 복구하려면 항상 로그 백업을 명시적으로 적용해야 합니다. 하지만 SQL Server 2005에서는 모든 데이터 및 차등 백업에 로그 레코드가 포함됩니다.

[따라하기] Sample 데이터베이스의 SamplesFG1 파일 그룹 백업하기

```
BACKUP DATABASE Sample
FILEGROUP = 'SamplesFG1'
TO Disk='F:\BackUp\ Sample_SamplesFG1.BAK'
GO
```

[따라하기] Sample 데이터베이스의 부분 백업하기

```
BACKUP DATABASE Samples
READ_WRITE_FILEGROUPS TO Disk='F:\BackUp\ Sample_SamplesFG1.BAK'
GO
```

미러된 백업

Microsoft SQL Server 2005 에서는 백업 미디어의 미러링 기능이 새롭게 추가되었습니다. 백업 미디어를 미러링하면 중복 백업을 확보함으로써 백업 안정성이 향상됩니다.
예를 들어, 관리자는 각 미디어 제품군에 대한 미러를 사용하여 두 개의 미디어 제품군을 백업하도록 네 개의 테이블 장치를 설정할 수 있습니다. 관리자는 미러 백업 집합을 최대 네 개까지 구현할 수 있습니다.

[따라하기] 미러된 백업하기

```
BACKUP DATABASE Sample
TO DISK='D:\Sample1a.bak', DISK='D:\Sample2a.bak'
MIRROR TO DISK='E:\Sample1b.bak', DISK='E:\Sample2b.bak'
WITH FORMAT;
GO
```

백업 전략 수립

전체 데이터베이스 백업은 항상 주기적으로 수행해 주어야 합니다. 트랜잭션 로그 백업은 모든 경우에 반드시 수행해야 하는 것은 아니지만 대부분의 경우에 로그 백업을 수행합니다. 트랜잭션 로그 백업을 수행하지 않는 예외적인 경우는 데이터의 변경이 드물게 발생하거나 테스트 환경에서입니다. 차등 백업은 트랜잭션이 빈번하게 발생하고 로그 백업의 크기가 큰 환경에서 주로 사용됩니다. 파일과 파일 그룹 백업 전략은 대용량 데이터베이스 환경에서 사용합니다. 다중 파일로 구성된 데이터베이스라도 한 번에 하나의 파일로 백업할 수 있습니다.

■ 시스템 데이터베이스는 변경이 발생할 때마다 백업해야 합니다.

사용자 데이터베이스와 마찬가지로 시스템 데이터베이스에 대해서도 백업이 필요합니다. 시스템 데이터베이스에는 시스템에 관련된 중요한 정보들이 관리되므로 백업을 수행해야 합니다. master 데이터베이스와 msdb 데이터베이스는 데이터베이스에 변경이 발생할 때마다 백업하는 것이 원칙입니다. 데이터베이스의 생성 및 변경, 로그인 정보의 변경, 연결된 서버의 변경, 구성 변경 등의 작업이 수행되면 master 데이터베이스 백업을 수행해야 합니다. 작업, 경고, 작업자, 스케줄 등이 생성되거나 변경될 때에는 msdb를 백업해야 합니다.

- master, msdb : 단순 복구 모델의 전체 백업
- model : 전체 복구 모델의 전체 백업

■ **백업 전략은 복원 시간까지 감안하여 계획을 세웁니다.**
백업전략은 데이터의 중요성, 데이터의 변경 주기, 복원 시간 등 여러 가지 요인들을 고려하여 수립합니다.

■ **트랜잭션 로그를 정기적으로 백업하지 않는다면, 정기적으로 비워 주어야 합니다.**
트랜잭션 로그가 가득 차면, 데이터베이스에서의 모든 변경 작업은 트랜잭션 로그가 삭제되거나 로그가 확장될 때까지 중단되므로, 로그 파일은 자동으로 증가되도록 설정할 것을 권고합니다. 그리고, 사용된 로그 공간의 양은 지속적으로 스크립트나 감사 테이블 또는 SQL Server:Databases 개체의 카운터 Percent Log Used의 성능 로그를 통하여 모니터링 해야 합니다.

어떤 시스템의 경우에는 트랜잭션 로그 파일의 크기가 데이터 파일의 수십 배에 달하는 경우를 간혹 볼 수 있습니다. 그 이유는 데이터베이스의 복구 모델이 전체(FULL) 또는 대량 로그(BULK_LOGGED)인데, 데이터베이스 전체 백업만 수행하고 로그 백업이나 삭제 작업은 수행하지 않았기 때문입니다. 전체 백업을 수행하더라도 트랜잭션 로그는 삭제되지 않으므로 주기적인 트랜잭션 로그 백업 또는 트랜잭션 로그 삭제가 필요합니다. 중요한 데이터가 저장된 데이터베이스라면 트랜잭션 로그를 정기적으로 백업하는 것을 권고하며, 테스트 DB와 같이 트랜잭션 로그 백업이 필요하지 않는 경우라면 트랜잭션 로그를 정기적으로 삭제해 주어야 합니다.

[따라하기] 트랜잭션이 완료된 로그 삭제하기

```
BACKUP LOG Sample WITH NO_LOG;
GO
-- 또는
BACKUP LOG Sample WITH TRUNCATE_ONLY;
GO
```

[참고]
데이터베이스가 단순 복구 모델이거나 "truncate log on checkpoint" 옵션이 선택되어 있을 때 트랜잭션 로그 백업을 하면, SQL Server Management Studio에서는 트랜잭션 로그 옵션이 비활성화 상태가 되고, 스크립트로 트랜잭션 로그 백업을 시도하면 4208 오류가 반환됩니다. 트랜잭션 로그 백업을 수행하기 위해서는, ALTER DATABASE를 사용하여 복구 모델을 변경하거나, "truncate log on checkpoint" 옵션이 비활성화되어야 합니다. "truncate log on checkpoint" 옵션을 False로 설정하면 복구 모델이 전체 복구 모드로 설정됩니다.

■ 백업 파일은 데이터베이스 파일이 저장된 디스크와 물리적으로 다른 디스크에 저장합니다.
디스크로 백업하고 별도의 위치로 백업 파일을 저장하는 것이 원칙입니다. 여건상 하드 디스크에만 백업받는 경우에는 최소한 데이터베이스 파일이 저장된 디스크와 물리적으로 다른 디스크로 백업합니다.

■ 주기적으로 백업 파일의 유효성과 백업이 실제로 정상적으로 복원되는지 테스트합니다.
[백업 검증]에 있는 내용을 참조하여 백업 세트의 유효성을 점검할 것을 권고합니다. 만일의 경우를 대비하여 백업을 열심히 받아 두었는데 막상 문제가 발생해서 복원하려고 하면 복원이 정상적으로 되지 않아서 낭패를 겪는 경우를 간혹 볼 수 있습니다. 백업 장비에 문제가 있는 경우도 있으므로, 특히 새로운 백업 장비 도입 시에는 백업 후 반드시 다른 DB 서버에서 복원을 테스트하기 바랍니다.

백업 작업의 제한 사항

SQL Server 2005에서는 데이터베이스가 온라인 상태이며 사용중일 때 백업할 수 있으며, 오프라인 데이터는 백업되지 않습니다. 다음과 같은 경우입니다.

- 전체 백업을 실행하는 데이터베이스의 파일 그룹 한 개가 오프라인일 경우, 전체 백업은 데이터베이스의 모든 파일 그룹을 포함하므로 이 작업은 실패합니다. 이 데이터베이스를 백업하려면 파일 또는 파일 그룹 백업을 사용하고 온라인 파일 그룹만 지정합니다.

- 부분 백업을 실행할 때 읽기/쓰기 파일 그룹이 오프라인인 경우, 부분 백업에는 모든 읽기/쓰기 파일 그룹이 필요하므로 작업은 실패합니다.

- 특정 파일의 파일 백업을 실행할 경우 파일 중 하나라도 온라인이 아닌 경우 작업은 실패합니다. 온라인 파일을 백업하려면 파일 목록에서 오프라인 파일을 생략하고 작업을 진행합니다.

대량 로그 복구 모델에서 이루어진 대량 로그 변경 내용이 없으면 하나 이상의 데이터 파일을 사용할 수 없더라도 로그 백업은 성공합니다. 파일에 대량 로그 변경 내용이 있으면 모든 파일이 온라인이어야 백업이 성공합니다.

전체 백업이 수행되는 동안에는 다음 작업이 허용되지 않습니다. 다음 작업 중 하나가 진행중일 때 백업이 시작되면 백업은 작업이 완료될 때까지 또는 세션 시간 제한에서 설정된 제한까지 기다립니다. 백업 작업이 진행 중일 때 다음 작업 중 하나를 시도하면 해당 작업이 실패하고 백업 작업은 계속됩니다.

- 데이터베이스 파일 생성 및 삭제
- 축소 작업 중 파일 잘라내기

백업 및 복원 성능 최적화

백업 및 복원 작업 속도를 향상시키고자 하는 경우에는 다음과 같은 방법을 고려합니다.

- 특정 여러 개의 백업 장치를 사용하면 백업을 모든 장치에 병렬로 기록할 수 있습니다. 여러 개의 장치를 사용하면 사용하는 장치 수에 비례하여 백업 속도가 증가하여 처리량이 증가할 수 있습니다. 마찬가지로 백업을 여러 장치에서 병렬로 복원할 수 있습니다.

- 특정 전체, 전체 차등 및 트랜잭션 로그 백업(전체 또는 대량 로그 복구 모델인 경우)을 조합해 사용하면 복구 시간을 최소화할 수 있습니다. 전체 차등 백업은 일반적으로 전체 백업보다 빠르게 만들 수 있으며 데이터베이스 복구에 필요한 트랜잭션 로그의 양도 줄일 수 있습니다.

데이터베이스 파일이 여러 개의 디스크에 분산되어 있으면 병렬로 디스크 I/O를 처리할 수 있으므로 백업 성능에 도움이 됩니다. 그리고 다중의 백업 디바이스로 백업하면 백업 수행 속도가 향상됩니다. 스트라이핑된 백업 세트를 생성할 때에는, 모든 백업 디바이스의 미디어 타입이 동일해야 합니다. 디스크 드라이브가 테이프보다 훨씬 빠르며 테이프 백업은 SQL Server에 물리적으로 장착이 되어야만 가능합니다. 속도를 향상시키고자 한다면, 먼저 직접 디스크에 백업을 받은 다음에 백업 파일을 오프사이트로 저장하기 위해 써드 파티 도구를 사용하여 테이프로 복사하거나 다른 드라이브로 복사합니다.

> ***[참고]***
> *네트워크 드라이브 백업이 가능하지만, 백업 성능이 좋지 않으며 네트워크 부하를 가중시킬 수 있으므로 유의합니다.*

[참고] 대규모 중요 환경에서의 백업 및 복원

중요 환경에서는 데이터베이스를 장기간에 걸쳐 계속해서 사용하거나 유지 관리 작업을 위한 작동 중지 시간을 최소화하면서 사용해야 하는 경우가 종종 있습니다. 따라서 데이터베이스 복원이 필요한 상황의 시간을 가능한 최소한으로 단축해야 합니다. 또한 중요 데이터베이스는 크기가 커서 백업하고 복원하는 데 더 많은 시간이 걸리는 경향이 있습니다. SQL Server에서는 백업과 복원 작업의 속도를 향상하여 두 작업이 실행되는 동안 사용자가 받는 영향을 최소화하는 몇 가지 방법을 제공합니다.

- *동시에 여러 개의 백업 장치를 사용하여 모든 장치에 동시에 백업을 기록할 수 있습니다. 마찬가지로 동시에 여러 개의 장치에서 백업을 복원할 수 있습니다.*
- *전체 복구 모델에서는 데이터베이스, 차등 데이터베이스, 트랜잭션 로그 백업의 조합을 사용하여 데이터베이스를 오류 시점으로 가져오기 위해 적용해야 할 백업의 수를 최소화합니다.*
- *전체 데이터베이스를 백업하거나 복원하는 대신 파일과 파일 그룹 백업, 트랜잭션 로그 백업 등을 사용하여 관련된 데이터가 포함된 파일만 백업하거나 복원합니다.*

백업 검증

RESTORE VERIFYONLY를 사용하면 백업을 복원하지 않고 백업 디바이스를 검사하여 백업 세트가 올바른지 그리고 모든 볼륨을 제대로 읽을 수 있는지 확인할 수 있습니다. 그러나, 이 명령어는 DB 데이터의 손상 여부까지 확인해 주지는 않습니다. 그러므로 대기 서버를 사용하여 DBCC 명령어를 수행하여 데이터의 손상 여부를 확인해야 완벽한 점검이 가능합니다. 주기적으로 운영 서버가 아닌 대기 서버에서 DB를 복원하고 DBCC CHECKDB 명령어를 사용하여 백업에 포함된 데이터가 손상되지 않았는지 확인할 것을 권고합니다.

복원과 복구

복원은 백업에서 데이터를 복사하고 기록된 트랜잭션을 데이터에 적용하여 대상 복구 지점으로 롤포워드하는 과정입니다.

복구는 데이터베이스를 일관성 있고 사용 가능한 온라인 상태로 만드는 일체의 작업 과정입니다. 일반적으로 복구 지점의 데이터베이스는 커밋되지 않은 트랜잭션을 가지며 일관성이 없고 사용할 수 없는 상태입니다. 이러한 경우 복구는 커밋되지 않은 트랜잭션의 롤백을 포함합니다.

■ 복원의 종류

• 전체 데이터베이스 복원

백업의 모든 데이터를 복사하고 백업에 로그가 포함되어 있으면 데이터베이스를 롤포워드한 다음 복구하여 커밋되지 않은 모든 트랜잭션을 롤백하고 데이터베이스를 온라인 상태로 만듭니다.

• 파일 복원

백업에서 지정한 파일이나 파일 그룹만 복사한 다음 백업에 로그가 포함되어 있으면 데이터베이스를 롤포워드합니다.

• 증분 복원

주 파일 그룹과 지정한 파일 그룹을 복사한 다음 백업에 로그가 포함되어 있으면 데이터베이스를 롤포워드합니다.
증분 복원을 사용하면 주 파일 그룹 및 일부 보조 파일 그룹을 초기에 부분적으로 복원한 다음 파일 그룹을 복원할 수 있습니다. 복원되지 않은 파일 그룹은 오프라인으로 표시되고 액세스할 수 없습니다. 그러나 오프라인 파일 그룹은 나중에 파일 복원으로 복원할 수 있습니다. 이렇게 서로 다른 시간에 단계별로 전체 데이터베이스를 복원할 수 있도록 증분 복원에서는 각 단계가 끝날 때마다 계속 데이터베이스의 일관성을 확인하고 그 정보를 유지합니다.
증분 복원 작업은 모든 복구 모델에서 가능하지만 단순 모델보다는 전체 및 대량 로그 모델에 대해 더 융통성이 있습니다.

• 트랜잭션 로그 복원

로그 백업을 복원하고 이 로그를 사용하여 데이터베이스를 롤포워드합니다.

• 온라인 복원

SQL Server 2005는 SQL Server를 실행하는 동안 복원 작업을 수행하는 기능을 제공합니다. 데이터베이스 파일 또는 페이지에 대한 부분적인 데이터베이스 복원이 수행 중인 동안에도 사용자가 데이터베이스에 액세스할 수 있습니다. 복원 중인 데이터만 사용할 수 없다는 점에서 온라인 복원은 SQL Server의 가용성을 향상시킵니다. 데이터베이스의 나머지 부분은 온라인 상태를 유지하며 계속 사용할 수 있습니다.

페이지 복원은 격리된 손상 페이지를 복구하기 위한 기능입니다. 페이지 복원은 소수의 페이지가 손상된 경우에 사용하는 방법입니다. 몇몇 페이지를 각각 복원하고 복구하면 복원 작업 도중 오프라인 상태인 데이터의 양이 줄어들어 파일 복원보다 빠를 수 있습니다. 그러나 파일에 있는 여러 페이지를 복원할 경우 일반적으로 전체 파일을 복원하는 것이 효율적입니다.

온라인 복원은 SQL Server Enterprise Edition에서만 사용할 수 있습니다.

모든 데이터베이스 운영 환경은 반드시 장애 복구에 대한 백업과 복원 계획을 가지고 있어야 합니다. 백업과 복원 계획은 실제 운영 서버를 백업하여 실제와 동일한 상태로 철저히 테스트하고 문서화해야 합니다. 백업과 복원 계획은 응용 프로그램과 운영 체제의 구성 요소를 포함하여, 전체 시스템에 대하여 문서화해야 하며, 발생 가능한 모든 장애 시나리오를 고려하여 문서화해야 합니다. 반드시 규칙적인 테스트를 수행해야 합니다. 계획을 수립할 때에는, 시스템이 얼마 동안 다운되어도 무방한지, 어느 정도의 데이터가 유실되어도 되는지에 관련된 리소스 비용과 다운타임, 복구 비용을 고려합니다.

복원 전략 수립

손상된 데이터베이스를 복원하는 첫 번째 단계는 현재의 트랜잭션 로그를 백업하는 것입니다. 이 작업은 트랜잭션 로그 파일이 액세스 가능하고 손상되지 않았을 때 가능합니다. 비록 데이터베이스가 suspect 상태일지라도, 마지막 트랜잭션 로그 백업의 시점부터 데이터베이스 파일이 손상되었을 시점까지의 전체 트랜잭션 로그를 백업합니다.

복원 과정에서 복원되는 마지막 백업은 문제 발생 후 백업한 트랜잭션 로그 백업이거나 마지막 로그 백업이며, 사용 가능한 트랜잭션 로그 백업이어야 합니다. 마지막 백업 이전의 복원 단계에서는 NORECOVERY 옵션을 사용해야 하며, 마지막 백업을 복원할 때에는 RECOVERY 옵션을 사용합니다.

> ***[참고]***
> *트랜잭션 로그 백업이 RECOVERY 옵션으로 복원되면, 추가적인 로그는 복원될 수 없습니다. 만일 추가적인 로그가 존재하면, 복원 프로세스는 반드시 마지막 전체 데이터베이스 백업을 가지고 처음부터 다시 시작해야 합니다.*

■ 전체 백업을 다른 서버에 복원하기

백업 일시	백업
월 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH NOINIT;
화 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH NOINIT;
수 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH NOINIT;

[따라하기] 전체 백업을 새로운 서버에 복원하기

전체 백업을 새로운 서버에 복원한 후, 새로운 서버의 로그인 정보를 복원한 데이터베이스의 사용자에 연결해 주는 예제입니다. sp_change_users_login을 사용하면, 사용자의 권한을 상실하지 않고 새 로그인에 사용자를 연결시킬 수 있습니다.

1. 백업 파일에 대한 정보를 확인합니다.

```
-- 모든 백업 세트들에 대한 백업 헤더 정보를 검색합니다.
RESTORE HEADERONLY
        FROM DISK='F:\DBBackup\Sample.bak';
GO
-- 복원할 백업 세트에 포함된 데이터베이스와 로그 파일 정보를 확인합니다.
RESTORE FILELISTONLY
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 3;
GO
```

2. 원하는 전체 백업 파일을 새로운 서버에 복원합니다.

```
USE master;
GO
RESTORE DATABASE Sample
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 3, RECOVERY;
GO
```

만약 복원에 문제가 발생하면, DBCC VERIFYONLY 명령어를 사용하여 백업 세트의 유효성을 확인합니다. 이 명령어는 실제 복원 작업보다는 수행 시간이 조금 짧기는 하지만, 수행 시간이 오래 걸립니다.

```
RESTORE VERIFYONLY
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 3;
GO
```

3. 복원이 완료되면, 사용자 정보를 연결합니다.

```
CREATE LOGIN dbadmin WITH PASSWORD='123abc';
GO
USE Sample;
GO
EXEC sp_change_users_login 'Update_One', 'dbadmin', 'dbadmin';
GO
```

[참고]

SQL Server는 GUID를 생성하여 syslogins.sid에 저장하며 이 sid를 로그인 이름의 security_identifier로 사용합니다. 서버가 다르면 Login 계정이 동일하더라도 이 sid값은 달라지며 로그인과 사용자에 대한 처리는 sid를 사용하므로, 원격 서버로 데이터베이스를 복원한 경우에는 새로운 서버의 로그인 계정과 복원한 데이터베이스의 사용자를 연결하는 작업이 필요합니다.

```
SELECT SUSER_SNAME (security_identifier);
SELECT sid FROM sys.syslogins WHERE name='dbadmin';
SELECT sid FROM Sample.sys.sysusers WHERE name='dbadmin';
```

■ 전체 백업과 차등 백업을 실행한 경우의 복원하기

백업 일시	백업
월 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH INIT;
화 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH DIFFERENTIAL, NOINIT;
수 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH DIFFERENTIAL, NOINIT;

[따라하기] 차등 백업을 사용하여 복원하기

차등 백업은 마지막 데이터베이스 백업 이후에 수정된 모든 페이지의 복사본을 저장하므로, 전체 백업 이후의 최종 차등 백업만 복원하면 됩니다. 문제가 발생하여 복원하는 경우에는 항상 복원 전에 현재의 트랜잭션 로그를 백업합니다. (로그 백업이 가능한 경우)

1. 백업 세트에 대한 정보를 확인합니다. (전체 백업을 다른 서버에 복원하기 참조)

2. 장애가 발생하기 전의 마지막 전체 백업을 복원합니다.

```
USE master;
GO
RESTORE DATABASE Sample
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 1, NORECOVERY;
GO
```

3. 복원한 전체 백업 후의, 마지막 차등 백업 파일을 복원합니다.

```
RESTORE DATABASE Sample
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 3, RECOVERY;
GO
```

■ 전체 백업과 트랜잭션 로그 백업을 실행한 경우 데이터베이스 복원하기

백업 일시	백업
월 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH INIT;
월 10:00	BACKUP LOG Sample TO DISK='F:\DBBackup\Sample_log.bak';
월 15:00	BACKUP LOG Sample TO DISK='F:\DBBackup\Sample_log.bak';

[따라하기] 전체 데이터베이스 백업과 로그 백업으로 데이터베이스 복원하기

트랜잭션 로그는 로그 백업 이후의 변경된 자료만을 가지고 있기 때문에, 복원할 경우에는 모든 로그 파일이 순차적으로 필요합니다.

1. NO_TRUNCATE 절을 사용하여 BACKUP LOG 문을 실행함으로써 현재 활성화된 트랜잭션 로그를 백업합니다.

```
BACKUP LOG Sample
TO DISK='F:\DBBackup\Sample_log2.bak'
WITH NO_TRUNCATE;
GO
```

2. 장애가 발생하기 전의 마지막 전체 백업을 복원합니다.

```
USE master;
GO
RESTORE DATABASE Sample
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 1, NORECOVERY;
GO
```

3. 복원한 전체 백업 이후, 첫 번째 로그 백업을 복원합니다.

```
RESTORE LOG Sample
        FROM DISK='F:\DBBackup\Sample_log.bak'
        WITH FILE = 1, NORECOVERY;
GO
```

4. 순차적으로 다음 로그 백업을 차례로 복원합니다.

```
RESTORE LOG Sample
        FROM DISK='F:\DBBackup\Sample_log.bak'
        WITH FILE = 2, NORECOVERY;
GO
```

5. 단계1에서 백업받은 로그 백업을 복원합니다. (백업이 성공한 경우)

```
RESTORE LOG Sample
        FROM DISK='F:\DBBackup\Sample_log2.bak'
        WITH RECOVERY;
GO
```

[참고]
복구 모델이 "대량 로그 복구"일 경우에는, SELECT INTO 등과 같은 최소 로깅 작업은 복원할 수 없습니다.

■ 전체 백업과 파일 그룹 백업을 실행한 경우 데이터베이스 복원하기

백업 일시	백업
월 05:00	BACKUP DATABASE Sample TO DISK='F:\DBBackup\Sample.bak' WITH INIT;
화 05:00	BACKUP DATABASE Sample FILEGROUP ='PRIMARY' TO DISK='F:\DBBackup\Sample_prm.BAK';
화 17:00	BACKUP LOG Sample TO DISK='F:\DBBackup\Sample_log.BAK';
수 05:00	BACKUP DATABASE Sample FILEGROUP ='SECONDARY' TO DISK='F:\DBBackup\Sample_scn.BAK';
수 17:00	BACKUP LOG Sample TO DISK='F:\DBBackup\Sample_log.BAK';

[따라하기] 파일 그룹 백업 복원하기

위의 백업을 실행 후에, Secondary 파일 그룹이 깨졌다고 가정합니다. 전체 백업을 복원할 필요 없이, 파일 그룹 백업만으로 복원이 가능합니다. 대용량 데이터베이스일 경우, 파일 그룹 백업은 복원 시간 단축에 매우 효과적입니다.

```
1. Secondary 백업을 복원합니다.
   RESTORE DATABASE Sample Secondary
            FROM DISK='F:\DBBackup\Sample_scn.bak'
            WITH FILE = 1, NORECOVERY;
   GO

2. 복원한 백업 이후의, 로그 백업을 순차적으로 복원합니다.
   RESTORE LOG Sample
            FROM DISK='F:\DBBackup\Sample_log.bak'
            WITH FILE = 2, RECOVERY;
   GO
```

[따라하기] 파일의 위치를 지정하여 데이터베이스 복원하기

Sample_dat 데이터 파일은 "c:\data\" 에, Sample_log 로그 파일은 "d:\log\" 로 위치를 변경하여 복원하고자 한다면, MOVE … TO 옵션을 사용하여 파일의 위치를 지정하면 됩니다.

```
   RESTORE DATABASE Sample
            FROM DISK='F:\DBBAckup\Sample.BAK'
            WITH MOVE 'Sample_dat' TO 'c:\data\Sample_dat.mdf'
            , MOVE 'Sample_log' TO 'd:\log\Sample_log.ldf'
            , REPLACE;
   GO
```

[참고]
REPLACE 옵션은 지정한 위치에 같은 파일이 이미 존재할 때 사용합니다.

■ 지정 시간 복구하기

지정 시간 복원은 오직 트랜잭션 로그 백업 상태에서만 가능합니다. RESTORE 명령어에 STOPAT 옵션을 사용하면 날짜와 시간을 정하여 데이터베이스를 복원할 수 있습니다.
이 경우 DBA는 사용자로부터 오류가 발생한 정확한 날짜와 시간을 알아내야 합니다. STOPAT 옵션은 정확하지 않은 데이터를 테스트하기 위하여 NORECOVERY 옵션과 함께 사용할 수가 없습니다. 정확한 시간이 필요합니다. RESTORE 문에 기술된 날짜와 시간 이전에 커밋되지 않은 트랜잭션은 롤백될 것이며 이는 데이터의 손실을 초래합니다. SQL Server 2005 Enterprise Edition에서만 지정 시간 복구를 사용할 수 있습니다.

[따라하기] 지정 시간으로 복원하기

2004년 12월 30일 오전 12시 상태로 데이터베이스를 복원하고 여러 로그와 여러 백업 장치와 관련된 복원 작업입니다.

```
USE master;
GO
RESTORE DATABASE Sample
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 1, NORECOVERY;
RESTORE LOG Sample
        FROM DISK='F:\DBBackup\Sample_log.bak'
        WITH FILE = 1, NORECOVERY;
RESTORE LOG Sample
        FROM DISK='F:\DBBackup\Sample_log.bak'
        WITH FILE = 2, RECOVERY, STOPAT = '2004-12-30 15:36:00.000';
GO
```

■ 표시된 트랜잭션 복원하기

표시된 트랜잭션은 DBA가 잘못된 트랜잭션이 발생한 시점을 확인하는데 있어 유용하며, 보다 쉽게 복원을 할 수 있도록 해 줍니다. WITH MARK 옵션을 사용하면 트랜잭션 이름이 트랜잭션 로그에 저장되며, 이 옵션을 사용하면 날짜와 시간 대신 표시된 트랜잭션을 사용하여 데이터베이스를 이전 상태로 복원할 수 있습니다.

로그에서 표시로 복원하는 방법은 다음 두 가지가 있습니다.

RESTORE LOG와 WITH STOPATMARK='mark_name' 절을 사용하여 표시된 부분까지 롤포워드하고 표시가 있는 트랜잭션을 포함시킵니다.

RESTORE LOG와 WITH STOPBEFOREMARK='mark_name' 절을 사용하여 표시된 부분까지 롤포워드하고 표시가 있는 트랜잭션은 제외시킵니다.

WITH STOPATMARK와 WITH STOPBEFOREMARK 절은 선택적인 AFTER datetime 절을 지원합니다. AFTER datetime이 생략되면 지정한 이름이 있는 첫 번째 표시 지점에서 복원이 중지됩니다. AFTER datetime이 지정되면 지정한 일시 또는 지정한 시점 이후에 지정한 이름이 있는 첫 번째 표시 지점에서 복원이 중지됩니다.

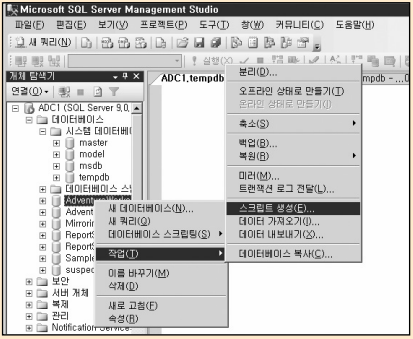
[따라하기] 표시된 트랜잭션 복원하기

```
/* 트랜잭션 표시 */
BEGIN TRANSACTION UpdateCol3 WITH MARK 'Update Col3 values';
GO
UPDATE Tab_Sample
SET Col3 = Col3 * 100;
GO
COMMIT TRANSACTION UpdateCol3;
GO
/* 표시된 트랜잭션 복원 */
USE master;
GO
RESTORE DATABASE Sample
        FROM DISK='F:\DBBackup\Sample.bak'
        WITH FILE = 1, NORECOVERY;
RESTORE LOG Sample
        FROM DISK='F:\DBBackup\Sample_log.bak'
        WITH FILE = 1, STOPATMARK ='UpdateCol3';
GO
```

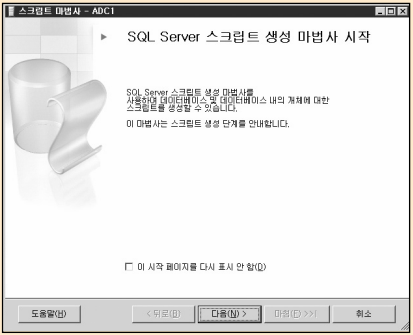
스크립트 백업

[따라하기] 오브젝트 스크립트 백업하기

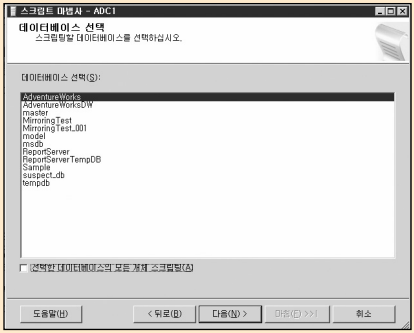
1. SQL Server Management Studio에서 스크립트를 생성하고자 하는 데이터베이스를 선택하고 마우스의 오른쪽 버튼을 클릭하여, [작업]→[스크립트 생성]을 선택합니다.



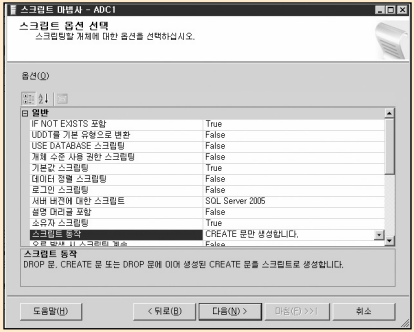
다음과 같은 스크립트 마법사가 실행됩니다.



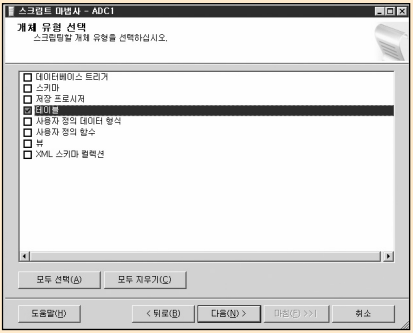
2. 스크립트를 생성하고자 하는 데이터베이스를 선택합니다.



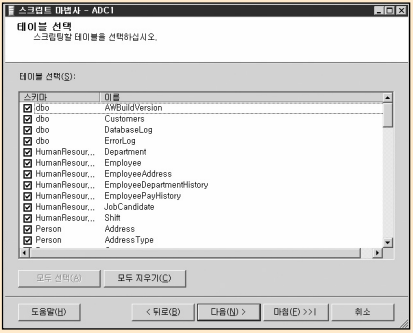
3. 스크립트 저장 옵션을 선택합니다. 스크립트 동작의 기본값인 [Create 문만 생성합니다]를 선택할 것을 권고합니다. 생성된 SQL 스크립트를 실수로 운영 DB서버에서 수행하여 운영중인 오브젝트들이 모두 삭제되는 불상사가 간혹 발생하고 있으므로, 항상 DROP 옵션은 해제한 상태에서 스크립트를 받을 것을 권고합니다.



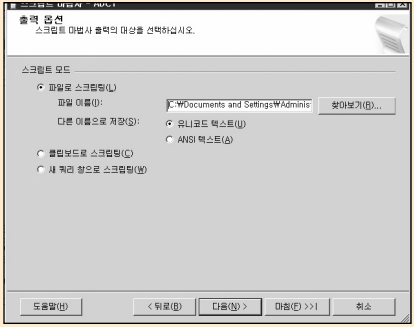
4. 스크립트로 저장할 개체 유형을 선택합니다. 테이블의 스크립트를 저장하려면 그림과 같이 [테이블]을 선택합니다.



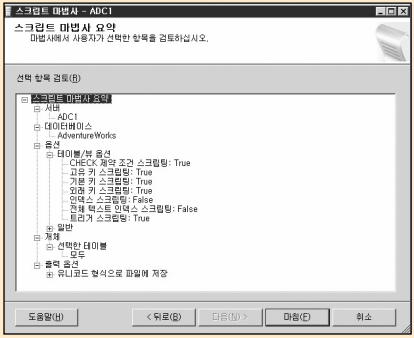
5. 저장할 개체를 선택합니다. 특정 개체를 선택하거나, [모두 선택]을 클릭하여 전체 개체를 선택합니다.



6. 출력 옵션을 선택합니다.



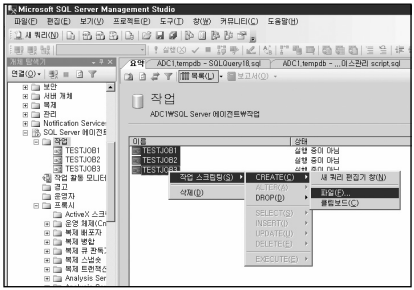
7. 스크립트 마법사 요약 정보를 확인합니다.



8. 그림과 같이 스크립트 생성 작업이 진행됩니다.

■ SQL Server 에이전트의 작업을 스크립트로 백업하기

SQL Server Management Studio에서 SQL Server 에이전트의 [작업]을 선택하고 [요약] 탭에서 원하는 작업을 선택하여 스크립트를 백업합니다.



[따라하기] 작업 정보 확인하기

SQL Server에서 자동화된 작업을 수행하기 위하여 SQL Server 에이전트가 사용하는 작업에 대한 정보를 반환합니다.

```
USE msdb;
EXEC dbo.sp_help_job;
GO
```

스키마 관리

스키마는 테이블, 뷰, 함수, 프로시저 등을 포함하는 보안 개체입니다. 사용자는 직접 테이블이나 뷰와 같은 데이터베이스 개체들을 직접 소유하는 대신 이러한 개체들이 포함되어 있는 스키마를 소유함으로써 각 개체들에 대한 권한을 얻게 됩니다. 사용자와 스키마가 분리 되어 있기 때문에, 사용자가 삭제 또는 변경이 되더라도 해당 스키마의 소유권을 다른 사용자에게로 변경하면 되기 때문에 데이터베이스 개체에는 영향을 미치지 않게 됩니다. 단일 사용자뿐만 아니라 데이터베이스 역할이나 Windows 그룹 계정도 스키마를 소유할 수 있기 때문에 이를 이용하여 하나의 스키마를 여러 사용자가 공유할 수 있습니다. SQL Server 2000과의 호환성을 위하여 dbo 스키마가 디폴트로 존재하며, 시스템 카탈로그의 스키마는 sys입니다.

■ 스키마 생성하기

현재 데이터베이스에 스키마를 생성합니다. CREATE SCHEMA는 한 문장으로 새로운 스키마를 생성하고, 그 스키마가 소유하는 테이블과 뷰를 생성할 수 있고, 이 오브젝트들에 대해 GRANT, DENY, REVOKE 권한 설정을 할 수 있습니다.

[따라하기] 스키마 생성하기

Person 스키마를 생성하면서 동시에 PersonInfo 테이블을 생성하는 예제입니다. Person 스키마는 UserA 사용자가 소유하며 UserB 에게는 Select 권한이 있고, UserC 에게는 Select 권한을 거부합니다.

```
CREATE LOGIN userA WITH PASSWORD='testA';
CREATE LOGIN userB WITH PASSWORD='testB';
CREATE LOGIN userC WITH PASSWORD='testC';
GO
USE Sample;
GO
CREATE USER userA;
```

```
CREATE USER userB;
CREATE USER userC;
GO
CREATE SCHEMA Person AUTHORIZATION userA
    CREATE TABLE PersonInfo (ID int, name varchar(50), Address varchar(100)
    GRANT SELECT TO userB
    DENY SELECT TO userC;
GO
```

■ 스키마 변경하기

ALTER SCHEMA는 동일한 데이터베이스 내에서 스키마 간에 개체를 이동할 때만 사용할 수 있습니다. 스키마 내에서 개체를 변경하거나 삭제하려면 해당 개체와 관련된 ALTER 또는 DROP 문을 사용합니다

[따라하기] 테이블을 동일 데이터베이스 내의 다른 스키마로 이동하기

```
-- PersonInfo 테이블을 Person 스키마에서 HR 스키마로 이동
USE Sample;
ALTER SCHEMA HR TRANSFER Person.PersonInfo;
GO
```

■ 스키마 삭제하기

스키마를 삭제하려면, 스키마가 소유하고 있는 테이블을 삭제한 다음에 스키마를 삭제할 수 있습니다.

[따라하기] 스키마 삭제하기

```
DROP TABLE HR.PersonInfo;
DROP SCHEMA HR;
GO
```

■ 스키마 정보 확인하기

sys.schemas 카탈로그 뷰를 통하여 스키마 목록을 확인할 수 있습니다.

■ 엔터티 소유권 변경하기

ACompany라는 회사에 userA라는 관리자가 있고 userA라는 관리자가 UserA.Table1이라는 테이블과 UserA.Procedure1이라는 저장 프로시저를 만들었으며 프로그램이나 SQL 쿼리 등에서 이 개체들을 사용해 왔습니다. 그런데 userA 가 퇴사를 하게 되고, userB라는 관리자가 이 자리를 대체하는 경우를 가정합니다. 이 경우, HR Schema의 소유권을 userB로 이전해 주기만 하면, HR 스키마는 userB가 소유하게 되고, 응용 프로그램이나 SQL 쿼리 등은 아무런 변경 사항 없이 HR.Table1 이나 HR.Procedure1 등을 사용할 수 있습니다. 그리고 userA 사용자 계정은 삭제할 수 있습니다.

[따라하기] 소유권 이전하기

```
ALTER AUTHORIZATION ON SCHEMA::HR TO userB;
GO
```

테이블 관리

테이블 생성

테이블을 만들 때에는 다음과 같은 사항들을 고려해야 합니다.

■ 데이터 형식

동일한 속성을 가진 데이터는 동일한 데이터 형식을 가져야 하며, 열에 저장되는 데이터의 값과 특성을 고려하여 적합한 데이터 형식을 선택합니다. 동일한 속성을 가진 데이터를 서로 다른 테이블들에서 다른 데이터 형식으로 선언한 경우에는 데이터의 불일치뿐 아니라 성능 저하를 유발할 수도 있고, 원하는 값의 결과와 다른 결과를 얻을 수도 있으므로 유의하기 바랍니다.

열에 저장할 데이터가 정수 데이터 형식의 경우에는 tinyint, smallint, int, bigint의 네 가지 데이터 형식이 지원되므로 저장될 데이터 값의 범위를 확인하여 데이터 형식을 선택합니다. 예를 들어 0에서 255까지의 정수를 저장할 열이라면 저장소 측면에서 int 대신 tinyint를 사용하는 것이 효율적이며, 21억이 넘는 큰 값이 저장될 열이라면 bigint를 사용해야 오버플로 오류가 발생하는 것을 방지할 수 있습니다.

소수점 이하 값이 없는 열에 불필요하게 numeric, decimal 타입을 사용하기 보다는 정수형 타입을 사용할 것을 권고합니다.

문자가 저장되는 열은 문자 데이터 형식을 할당하며, 저장되는 값의 길이가 일정하거나 길이의 차이가 적은 경우에는 고정 길이 문자형(char)을 사용하는 것이 성능적인 측면에서 유리합니다.

On/Off 또는 0/1, Yes/No와 같은 성격의 데이터는 bit 데이터 형식으로 설정하면, 하나의 테이블에 bit 타입이 여러 개 있는 경우에 레코드의 길이를 줄일 수 있습니다. 자세한 내용은 온라인 설명서를 참조하십시오.

[참고] 데이터 형식

분류	데이터 형식	범위	저장소 크기
정확한 수치	bit	0 또는 1	bit
	Int	-2,147,483,648 ~ 2,147,483,647	4 바이트
	smallint	-32,768 ~ 32,767	2 바이트
	tinyint	0 ~ 255	1 바이트
	bigint	-263 ~ 263-1	8 바이트
통화	money	-922,337,203,685,477.5808~ +922,337,203,685,477.5807	8 바이트
	smallmoney	-214,748,3648~214,748,3647	4 바이트
근사치	float[n]	-1.79E+308 ~ 1.79E+308 n = 25~53	8 바이트
	real	-3.40E + 38 ~ 3.40E + 38	4 바이트
문자열	char[n]	n = 1~8000	n 바이트
	varchar[n]	n = 1~8000	입력한 데이터의 길이
	text	최대 2,147,483,647자의 가변길이	
유니코드 문자열	nchar	n = 1~4000	n*2 바이트
	nvarchar	n = 1~4000	입력한 데이터의 길이*2 바이트
	ntext	최대 1,073,741,823자의 가변길이	
이진 문자열	binary	n = 1~8000	n+4 바이트
	varbinary	n = 1~8000	입력한 데이터의 길이+4 바이트

분류	데이터 형식	범위	저장소 크기
	image	최대 2,147,483,647자의 가변길이	
날짜 및 시간	datetime	1753/1/1~9999/12/31	8 바이트
	smalldatetime	1900/1/1~2079/6/6	4 바이트
기타 데이터 형식	cursor	CREATE TABLE 문에서 사용 불가	
	sql_variant	text, ntext, image, timestamp 및 sql_variant를 제외한, SQL Server 2005에서 지원하는 여러 가지 데이터 형식의 값을 저장하는 데이터 형식	
	table	결과 집합을 저장할 수 있는 특별한 데이터 형식	
	timestamp	데이터베이스 내에서 자동으로 생성된 고유 이진 숫자를 표시하는 데이터 형식	8 바이트
	uniqueidentifier	16바이트 GUID	
	xml	XML 데이터를 저장하는 데이터 형식. xml 유형의 변수 또는 열에 xml 항목을 저장	

- 큰 값 데이터 형식: varchar(max), nvarchar(max), varbinary(max)
  큰 개체 데이터 형식: text, ntext, image, varchar(max), nvarchar(max), varbinary(max), xml

■ 큰 값 데이터 형식

text, ntext, image 데이터 형식 대신 varchar(max), nvarchar(max), varbinary(max) 데이터 형식을 사용하는 것을 권고합니다.

SQL Server 2005에서 max 지정자가 새롭게 도입되어 varchar, nvarchar, varbinary 데이터 형식의 저장 기능이 확장되었습니다. varchar(max), nvarchar(max), varbinary(max)를 큰 값 데이터 형식이라고 합니다. 큰 값 데이터 형식을 사용하면 최대 2^31-1 바이트의 데이터를 저장할 수 있으면서, 동작은 기존의 varchar, nvarchar, varbinary의 동작과 유사하기 때문에 SQL Server에서 크기가 큰 문자열, 유니코드, 이진 데이터를 보다 효율적으로 저장하고 검색할 수 있게 되었으며 큰 값 데이터 형식을 사용하면 text, ntext, image 데이터 형식으로는 불가능한 일들이 가능하게 됩니다.

다음 표는 큰 값 데이터 형식과 이전 버전의 SQL Server에서 제공하는 데이터 형식의 관계를 보여 줍니다.

이전 버전의 LOB	큰 값 데이터 형식
text	varchar(max)
ntext	nvarchar(max)
image	varbinary(max)

[참고] identity 열 모니터링

identity 열은 tinyint, smallint, int, bigint, decimal(p,0) 또는 numeric(p,0) 열에 할당될 수 있습니다. Identity 열은 자동으로 값이 증가 또는 감소하는 속성을 가지고 있으므로 overflow, underflow가 발생하지 않도록 주기적으로 데이터 형식을 점검하는 것이 필요합니다.

[따라하기] xml 데이터 형식과 큰 값 데이터 형식 정의하기

```
CREATE TABLE test (
        KeyCol  int,
        Xcol    xml,
        Gif     varbinary(max),
        Descr   varchar(max));
GO
```

■ 제약 조건

데이터 무결성을 유지하여 데이터베이스의 품질을 보장하는 것이 필요하며, 데이터 무결성을 위하여 제약 조건을 적절히 사용해야 합니다. 테이블의 생성을 계획할 때 우선 열에 대하여 유효한 값이 무엇인지 확인하고, 그 다음은 열에 저장되는 데이터의 무결성을 유지하기 위한 방법을 결정하는 것입니다. 데이터 무결성은 엔터티 무결성, 도메인 무결성, 참조 무결성, 사용자 정의 무결성의 네 개의 범주로 구성되며, PRIMARY KEY 제약 조건, UNIQUE 제약 조건, FOREIGN KEY 제약 조건, CHECK 제약 조건, DEFAULT 정의, NOT NULL 정의, RULE 정의 등을 통하여 저장되는 값의 범위를 제한함으로써 무결성을 보장할 수 있습니다.

■ NULL 속성

테이블을 생성할 때, 항상 값이 저장되는 열은 반드시 NOT NULL 속성으로 정의합니다. NULL은 공백 문자나 0, 빈 문자열과는 전혀 다른 "알 수 없는 값" 입니다. 항상 값이 저장되어야 하는 열을 NULL 허용으로 정의하면 응용 프로그램의 오류로 NULL 값이 저장될 수 있으며, 그에 따른 NULL 데이터로 인하여 논리적 비교가 더욱 복잡해지거나 오류 데이터로 인하여 프로그램의 오동작을 유발할 수 있습니다. 그러므로 항상 명시적으로 값이 저장되는 열에 대해서는 반드시 NOT NULL 속성의 지정이 필요합니다.

데이터 무결성을 위한 제약 조건

제약 조건별로 명명 규칙을 정하고 규칙에 의거하여 이름을 부여할 것을 권고합니다. 제약 조건 외에도 모든 사용자 개체들에 대해서는 표준화된 명명 규칙을 수립하고 그 규칙에 의거하여 개체의 이름을 부여하기 바랍니다. 다음은 제약 조건별 접두어 규칙 예입니다.

제약 조건	접두어	이름 예제
PRIMARY KEY 제약 조건	PK_	PK_Orders
FOREIGN KEY 제약 조건	FK_	FK_Jobs_JobID
UNIQUE 제약 조건	UK_	UK_SSN
CHECK 제약 조건	CK_	CK_Quantity CK_MaxTemp_MinTemp
DEFAULT 정의	DF_	DF_CheckDate

■ PRIMARY KEY 제약 조건

테이블 생성 시에는 PRIMARY KEY 제약 조건을 지정합니다. PRIMARY KEY 제약 조건은 테이블을 생성할 때에 생성하는 것이 바람직하지만, PRIMARY KEY 제약 조건을 정의하지 않았더라도 테이블 생성 후에 추가로 생성할 수 있습니다. PRIMARY KEY 제약 조건을 설정할 열은 NOT NULL 속성을 가지고 있어야 하며, 고유한 데이터를 가지는 열이어야 합니다. 두 개 이상의 열에 PRIMARY KEY가 정의될 때에는, 한 열에 중복된 값이 있을 수 있지만, 열을 조합한 각 값은 고유해야 합니다.

CREATE TABLE 문에서 PRIMARY KEY를 정의하는 구문에 인덱스의 종류를 지정하지 않으면 PRIMARY KEY 열에 클러스터형 인덱스가 생성됩니다. 테이블들의 PRIMARY KEY는 반드시 클러스터형 인덱스일 필요는 없으며, 클러스터형 인덱스와 비클러스터형 인덱스 두 가지 중 성능적인 측면을 고려하여 효율적인 것을 사용해야 합니다.

[따라하기] 테이블 생성 시 PRIMARY KEY 제약 조건을 비클러스터형 인덱스로 설정하기

```
CREATE TABLE Tab_Sample (
    Col1        int identity(1,1)        NOT NULL PRIMARY KEY
Nonclustered,
    Col2        char(3)                  NULL,
    Col3        int                      NULL
);
GO
```

[따라하기] 테이블 생성 후, PRIMARY KEY 제약 조건 설정하기

```
CREATE TABLE Tab_Sample (
    Col1        int identity(1,1)        NOT NULL,
    Col2        char(3)                  NULL,
    Col3        int                      NULL
);
GO
ALTER TABLE Tab_Sample
ADD CONSTRAINT PK_Tab_Sample PRIMARY KEY Nonclustered (Col1);
GO
```

■ UNIQUE 제약 조건

PRIMARY KEY에 참여하지 않는 열 중에서 항상 고유한 값이 저장된다면 UNIQUE 제약 조건을 생성합니다. 데이터의 고유성이 보장되기 때문에, 응용 프로그램에서 데이터의 고유함을 확인할 필요가 없습니다.

■ FOREIGN KEY 제약 조건

참조 무결성이 보장되어야 하는 경우에는 FOREIGN KEY 제약 조건을 생성합니다. FOREIGN KEY 제약 조건이 없는 상태에서 응용 프로그램이 운영되는 상황에서 나중에 FOREIGN KEY 제약 조건을 추가하게 되면 응용 프로그램을 수정해야 하는 경우가 발생하므로, FOREIGN KEY 제약 조건은 최초에 테이블을 생성할 때 만드는 것이 좋습니다.

■ CHECK 제약 조건

논리 연산자에 따라 TRUE 또는 FALSE를 반환하는 논리식을 사용하여 CHECK 제약 조건을 생성할 수 있습니다. CHECK 제약 조건을 추가하면 데이터 무결성을 보장할 수 있을 뿐 아니라, CHECK 제약 조건에 위배되는 범위의 값을 조건절에서 검색하는 경우에는 실제로 테이블을 액세스하지 않고 결과를 바로 반환하므로 성능에도 도움이 됩니다. 이와 같이 성능에 도움이 되도록 하기 위해서는 CHECK 제약 조건을 WITH CHECK 옵션으로 생성해야 합니다.

열 하나에 여러 개의 CHECK 제약 조건을 적용할 수 있고, 한 테이블 내에서 테이블 단위로 CHECK 제약 조건을 만들어 여러 열에 적용할 수도 있습니다.

[따라하기] 열의 포맷 지정한 열(컬럼) 단위의 CHECK 제약 조건 생성하기

```
CREATE TABLE ColCheckTest (
        Seq int identity(1,1),
        State char(3),
        CONSTRAINT CK_State CHECK (State Like '[A-Z][0-9]S')
);
GO
```

[따라하기] 테이블 단위의 CHECK 제약 조건 생성하기

```
CREATE TABLE TblCheckTest (
        Seq int identity(1,1),
        MinValue int,
        MaxValue int,
);
GO
ALTER TABLE TblCheckTest
ADD CONSTRAINT CK_Min_Max CHECK (MinValue <= MaxValue);
GO
```

CHECK 제약 조건은 FALSE로 평가되는 값을 거부합니다. NULL은 알 수 없는 값으로 평가되므로 식에 NULL 값이 있으면 제약 조건이 무시됩니다. 예를 들어 MyColumn = 10과 같이 MyColumn에 값 10 만 포함할 수 있도록 int 열 MyColumn에 대한 제약 조건을 지정한다고 가정합니다. MyColumn에 NULL 값을 삽입할 경우 SQL Server 2005 데이터베이스 엔진에서는 NULL을 삽입하고 오류를 반환하지 않습니다.

CHECK 제약 조건 정보는 동적 관리 뷰 sys.check_constraints에서 확인할 수 있습니다.
제약 조건에 직접 xml 데이터 형식 메서드를 사용할 수 없습니다. CHECK 제약 조건에서 xml 데이터 형식 메서드를 사용하고자 하는 경우에는 UDF 래퍼를 생성하여 xml 데이터 형식 메서드를 래핑하면 됩니다.

xml 데이터 형식 메서드를 직접 CHECK 제약 조건으로 생성하려고 하면 다음과 같은 오류가 반환됩니다.

메시지 423, 수준 16, 상태 16, 줄 1
xml 데이터 형식 메서드는 CHECK 제약 조건에서 지원되지 않습니다. 메서드 호출을 래핑하려면 스칼라 사용자 정의 함수를 만드십시오. 이 오류는 테이블 "XmlTest"에서 발생했습니다.

[따라하기] XML 데이터 형식 메서드를 활용하여 xml 열에 CHECK 제약 조건 생성하기

```
USE Tempdb
GO
CREATE FUNCTION my_udf(@var xml) returns bit
AS BEGIN
RETURN @var.exist('/ProductDescription/@ProductID')
END
GO
CREATE TABLE XMLCheckTest (
   Col1 int primary key,
   Col2 xml check(dbo.my_udf(Col2)=1));
GO
```

UDF를 사용하여 보다 복잡한 CHECK 제약 조건의 생성이 가능합니다.

[따라하기] UDF를 사용하여 CHECK 제약 조건 생성하기

```
-- 동일한 값은 두 행까지만 삽입하도록 하는 예제입니다.
USE tempdb;
GO
CREATE TABLE DoubleKeys(
        key_col int NOT NULL,
        Other char(1)
);
GO
CREATE FUNCTION dbo.fn_DoubleKeysCnt (@key AS INT) RETURNS INT
AS
BEGIN
```

```
    RETURN (SELECT COUNT(*) FROM DoubleKeys WHERE key_col = @key);
END
  GO
-- UDF를 CHECK 제약 조건에 활용
ALTER TABLE DoubleKeys
    ADD CONSTRAINT CHK_DoubleKeys_NoMoreThanTwo
    CHECK (dbo.fn_DoubleKeysCnt(key_col) <= 2);
GO
-- 중복 키 데이터 INSERT
INSERT INTO DoubleKeys VALUES(1,'a');
INSERT INTO DoubleKeys VALUES(1,'a');
INSERT INTO DoubleKeys VALUES(1,'a');
INSERT INTO DoubleKeys VALUES(2,'a');
INSERT INTO DoubleKeys VALUES(2,'a');
INSERT INTO DoubleKeys VALUES(2,'a');
GO
-- 동일한 키 값은 두 건씩만 INSERT 되었음을 확인
SELECT * FROM DoubleKeys;
GO
```

■ DEFAULT 정의

열에 DEFAULT 정의가 있는 테이블로 행을 로드할 때 열에 대한 값이 지정되어 있지 않으면 SQL Server 2005 데이터베이스 엔진 에서 해당 열에 기본값을 삽입하도록 해 주는 기능입니다. NULL을 허용하는 열을 사용하는 것은 바람직하지 않으므로 가능하면 열에 DEFAULT 정의를 정의하는 것이 보다 좋은 해결책입니다.

열에서 NULL 값을 허용하지 않고 DEFAULT 정의도 없으면 열에 명시적으로 값을 지정해야 합니다. 그렇지 않으면 데이터베이스 엔진 이 열에서 NULL 값을 허용하지 않는다는 오류를 반환합니다.

테이블 삭제

[구문] DROP TABLE *table_name*;

테이블 정의 및 해당 테이블에 대해 지정된 모든 데이터, 인덱스, 제약 조건 및 권한을 제거합니다. FOREIGN KEY 제약 조건이 참조하는 테이블은 삭제할 수 없습니다. 참조하는 FOREIGN KEY 제약 조건 또는 참조하는 테이블을 먼저 삭제한 후, 테이블을 삭제합니다. 테이블을 삭제하면 테이블의 규칙 또는 기본값에 대한 바인딩이 해제되며 모든 관련 제약 조건도 자동으로 삭제됩니다. 테이블을 다시 작성할 경우 규칙 및 기본값을 다시 바인딩하고 필요한 모든 제약 조건을 추가해야 합니다. 삭제된 테이블을 참조하는 뷰나 저장 프로시저는 DROP VIEW나 DROP PROCEDURE를 사용하여 삭제합니다. 시스템 테이블에 대해서는 DROP TABLE 문을 사용할 수 없습니다.

[따라하기] 테이블 삭제하기

```
DROP TABLE Sample.dbo.Tab_sample;
GO
```

테이블 변경

■ 열 추가하기

테이블에 열을 추가할 경우, NOT NULL 속성 열을 추가할 수는 있지만, 이 경우에는 반드시 DEFAULT를 지정해야 합니다. DEFAULT를 지정할 수 없는 경우에는 일단 NULL 속성으로 열을 추가한 후, NULL 데이터를 조금씩 분할하여 NULL이 아닌 값으로 UPDATE하여 NULL인 값을 모두 NULL이 아닌 값으로 변경된 다음에, NOT NULL 속성으로 열을 변경하면 차단(Blocking)을 최소화하면서 NOT NULL 속성 열을 추가할 수 있습니다. 대용량 테이블에 NOT NULL 속성으로 열을 추가하면 차단(Blocking)이 발생하여 서비스에 지장을 초래할 수 있으므로 유의합니다.

정해진 시간 내에 작업을 끝내야 하는 시스템에서 크기가 큰 테이블에 열을 NOT NULL로 추가하였는데, 정해진 시간 내에 ALTER TABLE의 수행이 완료되지 않아서 서비스 장애로 이어지는 불상사가 간혹 발생합니다. 테스트 데이터베이스에서 소요시간을 미리 확인하고 소요시간이 제한된 시간을 초과한다면 다음의 팁을 활용하기 바랍니다.

대용량 LargeTabAddNotNullCol 테이블에, 데이터 형식이 char(50), NOT NULL 속성을 가진 NotNullCol 열을 추가하는 스크립트입니다.

[따라하기] NOT NULL 속성 열 추가하기

```
USE Sample;
GO
-- 테스트 테이블을 생성합니다.
SELECT identity(int,1,1) as Seq, p1.*
INTO LargeTabAddNotNullCol
FROM AdventureWorks.Production.Product p1
        CROSS JOIN AdventureWorks.Production.Product p2;
GO
-- NULL 속성으로 열을 추가합니다.
ALTER TABLE LargeTabAddNotNullCol
ADD NotNullCol char(50) NULL;
GO
-- 업데이터 성능을 위하여 인덱스를 생성합니다.
CREATE INDEX IDX_1 ON LargeTabAddNotNullCol (NotNullCol);
GO
-- 전체 데이터를 분할하여 업데이트합니다.
DECLARE @ROWS int
SET @ROWS=10000;
UPDATE TOP (@ROWS) LargeTabAddNotNullCol SET NotNullCol ='default value'
```

```
WHERE NotNullCol IS NULL;
WHILE @@ROWCOUNT = 10000
BEGIN
   UPDATE TOP (@ROWS) LargeTabAddNotNullCol
         SET NotNullCol ='default value'
   WHERE NotNullCol IS NULL;
END
GO
-- NotNullCol 열에 NULL 값이 있는지 확인합니다.
SELECT count(*) FROM LargeTabAddNotNullCol WHERE NotNullCol IS NULL;
-- NotNullCol 열에 존재하는 인덱스를 제거합니다.
DROP INDEX LargeTabAddNotNullCol.IDX_1;
GO
-- NotNullCol 열을 NOT NULL 속성으로 변경합니다.
ALTER TABLE LargeTabAddNotNullCol
ALTER COLUMN NotNullCol char(50) NOT NULL;
GO
```

■ 열 삭제하기

LargeTabAddNotNullCol 테이블에 DEFAULT를 설정한 Addcol 열을 추가한 다음에, 다시 그 열을 삭제하는 예제입니다. 제약 조건이 설정된 열은 제약 조건을 삭제한 후, 열을 삭제합니다.

[따라하기] 열 삭제하기

```
USE Sample;
GO
ALTER TABLE CheckTest
ADD Addcol CHAR(100) NOT NULL CONSTRAINT DF_Addcol DEFAULT 'default
value';
GO
ALTER TABLE CheckTest
DROP CONSTRAINT DF_Addcol;
GO
ALTER TABLE CheckTest
DROP COLUMN Addcol;
GO
```

■ 열 변경하기

ALTER COLUMN의 대상이 되는 열에 인덱스가 존재한다면 우선 인덱스를 삭제한 후 ALTER COLUMN을 실행해야 합니다.

[따라하기] nchar(3) 데이터 형식을 char(10) 데이터 형식으로 변경하기

```
USE Sample;
GO
SELECT * INTO ProductTest FROM AdventureWorks.Production.Product;
GO
EXEC sp_columns ProductTest;
GO
CREATE INDEX IDX_1 ON ProductTest (SizeUnitMeasureCode);
GO
```

```
/*
메시지 5074, 수준 16, 상태 1, 줄 1
인덱스 'IDX_1'은(는) 열 'SizeUnitMeasureCode'에 종속되어 있습니다.
메시지 4922, 수준 16, 상태 9, 줄 1
하나 이상의 개체가 이 열에 액세스하므로 ALTER TABLE ALTER COLUMN
SizeUnitMeasureCode이(가) 실패했습니다.
*/
DROP INDEX ProductTest.IDX_1;
GO
ALTER TABLE ProductTest ALTER COLUMN SizeUnitMeasureCode char(3);
GO
```

개체 이름 변경

사용자가 만든 개체의 이름을 변경할 수 있습니다. 개체 이름을 변경하면 스크립트나 저장 프로시저가 작동되지 않을 수 있으므로 사전에 충분한 확인이 필요합니다. 그리고 저장 프로시저, 트리거, 사용자 정의 함수, 뷰는 이름을 변경하지 않는 것이 좋습니다. 대신 개체를 삭제하고 새로운 이름으로 다시 만듭니다.

■ 테이블 이름 변경하기

[따라하기] 테이블 이름 변경하기

```
EXEC sp_rename 'Territories', 'Territs';
GO
```

■ **인덱스 이름 변경하기**

표준화된 명명 규칙에 따라 인덱스의 이름을 변경할 수 있습니다. 또한 예를 들어, 쿼리에서 강제로 어떤 인덱스를 사용하도록 인덱스 힌트를 사용한 경우에 인덱스의 이름에 HINT 접두어를 추가하여 표시함으로써 DBA가 임의로 인덱스를 변경하지 않도록 경고할 때 사용할 수도 있습니다.

[따라하기] 인덱스 이름 변경하기

```
EXEC sp_rename 'Customers.PostalCode', 'IX_ZipCode', 'INDEX'
GO
```

■ **제약 조건 이름 변경하기**

sp_rename을 사용하여 Customer 테이블을 Customers_Old으로 테이블 이름을 변경한 후에, Customers 테이블을 새로 만든다고 가정합니다. 데이터베이스 내에서 인덱스의 이름은 중복 가능하지만, PRIMARY KEY 제약 조건의 이름은 고유해야 합니다. 이런 경우에 Customers_Old 테이블의 PRIMARY KEY 제약 조건의 이름을 PK_Customers에서 PK_Customerd_Old로 변경하면, 새로 만드는 Customers 테이블에 PK_Customers라는 이름의 제약조건을 만들 수 있습니다.

[따라하기] 제약 조건 이름 변경하기

```
EXEC sp_rename 'Customers_Old.PK_Customers', 'PK_Customers_Old'
GO
```

■ **저장 프로시저, 뷰, 트리거 이름 변경하기**

저장 프로시저 및 뷰의 이름을 변경하면, 프로시저 캐시를 플러시하여 모든 종속 저장 프로시저 및 뷰가 재컴파일 됩니다.

저장 프로시저, 사용자 함수, 뷰 또는 트리거의 이름을 변경해도 sys.sql_modules 카탈로그 뷰의 definition 열에 있는 해당 개체의 이름은 변경되지 않으므로, 이 문장을 사용하여 변경하지 않을 것을 권고합니다. 대신에, 개체를 삭제한 다음에 새로운 이름으로 다시 만드는 것을 권고합니다.

[따라하기]

```
EXEC sp_rename 'Sales by Year', 'SalesByYear';
GO
```

■ 별칭 데이터 형식의 이름 변경하기

[따라하기]

```
CREATE TYPE Customer_SSN FROM varchar(13) NOT NULL;
GO
EXEC sp_rename 'Customer_SSN', 'SSN', 'USERDATATYPE';
GO
```

테이블 정보 확인

■ FOREIGN KEY 제약 조건 정보 확인하기

[따라하기] FOREIGN KEY 제약 조건 확인하기

Customers 테이블을 비활성화 되어 있는 FOREIGN KEY 제약 조건을 포함한 FOREIGN KEY 제약 조건의 FOREIGN KEY로 참조하고 있는 테이블과 열의 기본 정보를 반환합니다.

```
USE AdventureWorks;
EXEC sp_fkeys N'Sales.Customer';
GO
```

■ 테이블의 인덱스 정보 확인하기

[따라하기] 테이블 인덱스 목록 조회하기

```
USE AdventureWorks;
EXEC sp_helpindex N'Sales.Customer';
GO
```

■ 테이블의 제약 조건 정보 확인하기

[따라하기] 테이블 관련 모든 제약 조건 조회하기

```
USE AdventureWorks;
EXEC sp_helpconstraint 'Production.Product';
GO
```

■ 테이블의 모든 정보 확인하기

[따라하기] 테이블에 관련된 열, 인덱스, 제약 조건, 참조하는 뷰 확인하기

```
USE AdventureWorks;
EXEC sp_help 'Person.Contact';
GO
```

[참고]
SQL Server 2005에서의 sp_help 는 테이블 이외에도 sys.sysobjects 에 있는 데이터베이스 개체, 사용자 정의 데이터 형식, 데이터 형식에 관련된 정보를 반환합니다.

■ 테이블이 사용하는 공간 확인하기

[구문] sp_spaceused [[@objname =] 'objname']
[,[@updateusage =] 'updateusage']

[따라하기] 데이터베이스 내의 모든 테이블의 테이블별 사용 공간 확인하기

```
/* 방법1. 기존의 시스템 SP를 단순히 활용한 예제 */
USE AdventureWorks;
EXEC sp_MSforeachtable 'EXEC sp_spaceused [?], ''TRUE''';
GO

/* 방법2. 기존의 시스템 SP를 활용하여 결과를 테이블에 저장한 예제 */
USE Sample;
GO
CREATE TABLE spaceused_AdventureWorks (
```

```
        TableName   sysname,
        Rows        int,
        Reserved    varchar(20),
        Data        varchar(20),
        Index_size  varchar(20),
        Unused      varchar(20));
GO
USE AdventureWorks;
GO
INSERT INTO Sample.dbo.spaceused_AdventureWorks
EXEC sp_MSforeachtable 'EXEC sp_spaceused [?], ''TRUE''';
GO
SELECT * FROM Sample.dbo.spaceused_AdventureWorks
ORDER BY CAST(replace(Data, 'KB', '') AS int) DESC;
GO
SELECT * FROM Sample.dbo.spaceused_AdventureWorks
ORDER BY TableName ASC;
GO

/* 방법3. sp_spaceused의 소스 코드를 수정하여 사용하기 */
CREATE PROCEDURE sys.sp_spaceused_alltable
AS
SET NOCOUNT ON
SELECT t1.tablename,
    rows = convert(char(11),t1.rows),
    reserved = ltrim(str((t1.reservedpages+isnull(t2.reservedpages,0)) * 8192 /
1024.,15,0) + 'KB'),
    data = ltrim(str(t1.pages * 8192 / 1024.,15,0) + 'KB'),
    index_size = ltrim(str((t1.usedpages+isnull(t2.usedpages,0)) - t1.pages) * 8192
```

```
/ 1024.,15,0) + 'KB'),
      unused = ltrim(str(((t1.reservedpages+isnull(t2.reservedpages,0)) -
(t1.usedpages+isnull(t2.usedpages,0))) * 8192 / 1024.,15,0) + 'KB')
FROM (
      SELECT tablename = o.name,
            reservedpages = sum(a.total_pages),
            usedpages = sum(a.used_pages),
            pages = sum(
                  CASE
                        When a.type <> 1 Then a.used_pages
                              When p.index_id < 2 Then a.data_pages
                              Else 0
                              END
            ),
            rows = sum(
                  CASE
                              When (p.index_id < 2) and (a.type = 1) Then p.rows
                              Else 0
                              END
            )
      FROM sys.partitions p
            JOIN sys.allocation_units a ON p.partition_id = a.container_id
            JOIN sys.objects o ON p.object_id = o.object_id
      WHERE o.type_desc ='USER_TABLE'
            GROUP BY o.name ) t1
LEFT OUTER JOIN (
      SELECT tablename = o.name,
            reservedpages = sum(a.total_pages),
            usedpages = sum(a.used_pages)
```

```
    FROM sys.partitions p
        JOIN sys.allocation_units a ON p.partition_id = a.container_id
        JOIN sys.internal_tables it ON p.object_id = it.object_id and it.internal_type
IN (202,204)
        JOIN sys.objects o ON o.object_id = it.parent_id
    WHERE o.type_desc ='USER_TABLE'
    GROUP BY o.name) t2 on t1.tablename = t2.tablename;

SET NOCOUNT OFF
GO
```

행 오버플로

SQL Server 2005에서도 SQL Server 2000과 마찬가지로 행당 최대 바이트 수는 8,060바이트이지만 SQL Server 2000과 달리 SQL Server 2005에서는 varchar, nvarchar, varbinary 또는 sql_variant 또는 CLR 사용자 정의 유형 열을 사용하여 정의된 전체 테이블의 길이가 8,060바이트를 초과할 수 있습니다. 이것은 varchar, nvarchar, varbinary, sql_variant 또는 CLR 사용자 정의 유형 열을 만들거나 수정할 때 적용되며 데이터를 업데이트하거나 삽입할 때도 적용됩니다. 행 오버플로를 포함하는 테이블은 sys.dm_db_index_physical_stats 동적 관리 개체를 사용하여 정보를 얻을 수 있습니다.

■ 행당 8,060바이트를 초과하는 varchar, nvarchar, varbinary, sql_variant, CLR 사용자 정의 유형 열 사용시 고려할 사항

1. SQL Server 2005 데이터베이스 엔진에서 페이지당 8KB의 제한은 여전히 유효하기 때문에 8,060바이트의 행 크기 제한을 초과하면 성능이 저하될 수도 있습니다. 그러므로, 행 오버플로 데이터가 있는 행에서 쿼리가 자주 수행될 가능성이 높은 경우에는 일부 열이 다른 테이블로 이동하도록 테이블을 정규화하십시오.

2. varchar, nvarchar, varbinary, sql_variant 및 CLR 사용자 정의 유형 열의 개별 열 길이에는 여전히 8,000바이트의 제한이 적용 되며, 이 열이 결합되는 경우에만 테이블의 8,060바이트의 행 제한을 초과할 수 있습니다.

3. char 및 nchar 데이터를 비롯하여 다른 데이터 형식 열의 합계에는 8,060바이트의 행 제한이 적용되어야 합니다. (max)를 사용하는 큰 개체 데이터도 8,060바이트의 행 제한에서 제외됩니다.

4. 클러스터형 인덱스의 인덱스 키는 기존 데이터가 ROW_OVERFLOW_DATA 할당 단위에 있는 varchar 열을 포함할 수 없습니다.

5. 행 오버플로 데이터가 있는 열을 비클러스터형 인덱스의 키 열이나 키가 아닌 열로 포함할 수 있습니다.

[따라하기] 행 오버플로 확인하기

```
USE Sample;
GO
CREATE TABLE RowOverFlow (
        col1 int identity(1,1),
        col2 char(500),
        col3 varchar(8000),
        col4 varchar(8000)
);
GO
-- 명령이 완료되었습니다.

INSERT INTO RowOverFlow(col2, col3)
VALUES ('aaaaa','aaaaaa… 〈지면 관계로 생략〉….aaaa', 'aaaaaa…..aaaa');
GO
```

```
SELECT object_id,index_type_desc,alloc_unit_type_desc
FROM
sys.dm_db_index_physical_stats(db_id('sample'),Object_id('RowOverFlow3'),NULL
,NULL,NULL)
GO
/*
229575856          HEAP          IN_ROW_DATA
229575856          HEAP          ROW_OVERFLOW_DATA
*/
```

테이블 옵션 설정

사용자 정의 테이블의 옵션 값을 설정합니다.

[구문] sp_tableoption [@TableNamePattern =] 'table'
[@OptionName =] 'option_name'
[@OptionValue =] 'value'

■ 'text in row' 옵션

'text in row' 옵션을 설정하면, Text, ntext, image 열의 행에 저장할 최대 크기를 지정할 수 있습니다. 기본값은 256바이트이고, 값의 범위는 24에서 7000바이트입니다.
다음은 Orders 테이블의 text 열에 저장할 데이터를 1000바이트로 지정합니다.

중요
이 기능의 'text in row' 옵션은 이후 버전의 SQL Server 에서 제거될 예정입니다. 큰 값의 데이터를 저장하기 위해서는 varchar(max), nvarchar(max), varbinary(max) 데이터 형식을 사용할 것을 권고합니다.

[따라하기] 테이블에 'text in row' 옵션 설정하기

```
EXEC sp_tableoption 'orders', 'text in row', '1000'
GO
-- 설정값 확인
USE Northwind;
GO
SELECT OBJECTPROPERTY(OBJECT_ID('orders'),'TableTextInRowLimit');
GO
```

■ 'large value types out of row' 옵션

sp_tableoption을 사용하면 행 외부 옵션을 설정 및 해제할 수 있습니다.
varchar(max), nvarchar(max) 및 varbinary(max)를 큰 값 데이터 형식이라고 하며, 큰 값 데이터 형식을 사용하면 최대 231-1바이트의 데이터를 저장할 수 있습니다. sp_tableoption 저장 프로시저의 'large value types out of row' 옵션을 OFF로 설정하면 큰 값 형식의 행 내부 저장 용량 제한은 8,000바이트입니다. 이 옵션을 ON으로 설정하면 행 내부에 16바이트 루트가 저장됩니다.

대부분의 쿼리문이 큰 값 데이터 형식 열을 참조하지 않는 테이블에 대해서는 이 옵션을 ON으로 설정하는 것이 좋습니다. 이러한 열을 행 외부에 저장하면 각 페이지에 더 많은 행이 들어갈 수 있으므로 테이블을 검색하는 데 필요한 I/O 작업 수가 줄어듭니다. 이 옵션의 값을 OFF로 설정하면 각 페이지에 저장되는 데이터 행의 수가 줄어들어 쿼리를 처리하기 위해 읽어야 하는 페이지의 수가 늘어날 수 있습니다.

다음 문장 중에 하나를 사용하여 MyTable 테이블에 행 외부 옵션을 설정합니다.
sp_tableoption의 설정값은 true, on, 1 이고 해제값은 false, off, 0 입니다.
large value types out of row 옵션 값이 변경되는 경우, 기존 varchar(max), nvarchar(max), varbinary(max) 및 xml 값은 즉시 변환되지 않고, 문자열의 저장소는 다음에 업데이트될 때 변경됩니다.

[따라하기] 테이블에 행 외부 옵션 설정하기

```
EXEC sp_tableoption N'MyTable', 'large value types out of row', 'ON';
EXEC sp_tableoption N'MyTable', 'large value types out of row', 1
EXEC sp_tableoption N'MyTable', 'large value types out of row', 'true'
```

인덱스 관리

인덱스는 테이블이나 뷰의 조회를 빠르게 하기 위한 데이터 구조입니다. 인덱스는 테이블이나 뷰의 한 개 또는 한 개 이상의 열로 구성되어 있는 키로 구성되어 있으며, 그 키는 B-트리 구조로 구성되어 있습니다. 인덱스는 조회의 속도는 증가 시키지만, INSERT 속도는 저하시키므로 적절한 인덱스를 생성하는 것이 중요합니다. 인덱스는 테이블, 뷰, 또는 테이블의 XML 인덱스에 생성할 수 있습니다.

인덱스 디자인에 대해서 온라인 설명서에 다음과 같이 정보가 제공되므로 인덱스 디자인에 대한 내용을 다음을 참조하시기 바랍니다.

ms-help://MS.SQLCC.v9/MS.SQLSVR.v9.ko/udb9/html/cbb5bf99-6038-4a59-ada8-3886ef00454f.htm

인덱스 생성

■ 클러스터형 인덱스 생성

클러스터형 인덱스를 구성하는 키 값의 순서에 의해 테이블이나 뷰의 데이터가 물리적으로 정렬되는 인덱스입니다. 그러므로, 클러스터형 인덱스는 한 테이블에 한 개만이 존재할 수 있습니다. 클러스터형 인덱스가 존재하여 클러스터형 인덱스의 키 값에 따라 데이터가 정렬되어 있는 테이블을 "클러스터형 테이블" 이라고 하고, 클러스터형 인덱스가 없는 테이블을 "힙" 이라고 합니다.

[따라하기] 클러스터형 인덱스 생성하기

```
CREATE TABLE TestTable (
        Col1 int CONSTRAINT PK_TestTable PRIMARY KEY NONCLUSTERED ,
        Col2 int);
GO
CREATE CLUSTERED INDEX CX_Col2 ON TestTable (Col2)
WITH (FILLFACTOR = 90, MAXDOP = 0);
GO
```

■ 비클러스터형 인덱스 생성

비클러스터형 인덱스는 데이터와 분리된 데이터 구조를 가지는 인덱스입니다. 비클러스터형 인덱스는 비클러스터형 인덱스 키 값을 포함하며, 각 키 값에 해당하는 데이터 열이 위치한 포인터 "행 로케이터"를 가집니다. 클러스터형 인덱스가 있는 테이블의 비클러스터형 인덱스의 경우에는 행 로케이터가 클러스터형 인덱스 키이며, 클러스터형 인덱스가 없는 힙 테이블의 경우에는 열의 위치를 나타내는 RID가 행 로케이터가 됩니다.

[따라하기] 비클러스터형 고유 인덱스 생성하기

```
USE AdventureWorks;
GO
CREATE UNIQUE INDEX NX_UnitMeasure_Name
   ON Production.UnitMeasure(Name);
GO
DROP INDEX Production.UnitMeasure.NX_UnitMeasure_Name;
GO
```

■ xml 인덱스 생성

CREATE XML INDEX 명령어를 사용하면 xml 열에 XML 인덱스를 생성할 수 있습니다.
XML 인덱스를 만들 때는 다음 사항을 알아야 합니다.

- XML 인덱스를 생성하려면 해당 테이블에 클러스터형 인덱스가 있어야 합니다.
- XML 인덱스는 하나의 xml 열에만 만들 수 있습니다.
- 테이블의 각 xml 열에는 하나의 기본 XML 인덱스 및 여러 개의 보조 XML 인덱스가 있을 수 있습니다.
- 보조 XML 인덱스를 생성하려면 해당 xml 열에 기본 XML 인덱스가 있어야 합니다.
- XML 인덱스를 생성할 때에는 ONLINE 옵션을 사용할 수 없습니다.
- 계산 xml 열에 기본 XML 인덱스를 만들 수 없습니다.
- SET 옵션 설정은 인덱싱된 뷰 및 계산 열 인덱스에 필요한 설정과 동일해야 합니다. 특히 ARITHABORT 옵션은 ON으로 설정되어 있어야 합니다.

[따라하기] 기본 XML 인덱스 생성하기

```
USE AdventureWorks;
GO
IF EXISTS (SELECT * FROM sys.indexes
        WHERE name = N'PXML_ProductModel_CatalogDescription')
   DROP INDEX PXML_ProductModel_CatalogDescription
      ON Production.ProductModel;
GO
CREATE PRIMARY XML INDEX PXML_ProductModel_CatalogDescription
   ON Production.ProductModel (CatalogDescription);
GO
```

[따라하기] 보조 XML 인덱스 생성하기

```
USE AdventureWorks;
GO
IF EXISTS (SELECT name FROM sys.indexes
        WHERE name = N'IXML_ProductModel_CatalogDescription_Path')
   DROP INDEX IXML_ProductModel_CatalogDescription_Path
      ON Production.ProductModel;
GO
CREATE XML INDEX IXML_ProductModel_CatalogDescription_Path
   ON Production.ProductModel (CatalogDescription)
   USING XML INDEX PXML_ProductModel_CatalogDescription FOR PATH ;
GO
```

■ 포괄 열 인덱스 생성

SQL Server 2005에서는 키가 아닌 열을 포함한 비클러스터형 인덱스를 생성할 수가 있습니다. 포괄 열이 있는 인덱스는 인덱스 커버링 가능성을 높여 주기 때문에 성능 향상에 도움이 됩니다. 인덱스 커버링이 되면 쿼리 옵티마이저가 데이터를 참조하지 않고 비클러스터형 인덱스의 열 값만을 참조하여 원하는 데이터를 얻을 수 있게 되므로 적은 I/O를 발생시키기 때문입니다.

포괄 열이 있는 인덱스는 다음과 같은 장점이 있습니다.

- 인덱스 키 열로 허용되지 않는 데이터 형식을 포괄 열에는 허용됩니다. 즉, text, ntext, image 데이터 형식을 제외한 모든 데이터 형식이 허용됩니다.
- 비클러스터형 인덱스를 구성하는 인덱스 키 열의 개수 또는 인덱스 키의 크기 계산에서 제외됩니다.

[따라하기] 포괄 열을 추가한 인덱스 생성하기

```
USE AdventureWorks;
GO
-- 포괄 열을 추가한 인덱스 생성
CREATE NONCLUSTERED INDEX IX_Address_PostalCode
    ON Person.Address (PostalCode)
    INCLUDE (AddressLine1, AddressLine2, City, StateProvinceID);
GO
-- 포괄 열 인덱스로 인해 성능이 향상되는 쿼리 예
SELECT AddressLine1, AddressLine2, City, StateProvinceID, PostalCode
FROM Person.Address
WHERE PostalCode BETWEEN N'98000' and N'99999';
GO
```

■ 인덱스 옵션 지정

CREATE INDEX 문의 WITH 절에 옵션을 지정합니다. SQL Server 2005 에는 이전 버전에 없었던 새로운 인덱스 옵션이 몇 가지 추가되었으며, 옵션 지정 방법도 WITH (〈option_name〉 = ON) 또는 WITH (〈option_name〉 = OFF) 와 같은 형태로 변경되었습니다.

인덱스 옵션을 지정하면 다음과 같은 규칙이 적용됩니다.

- 새 인덱스 옵션은 WITH (option_name = ON | OFF)만 사용하여 지정할 수 있습니다.
- 옵션은 동일한 문에 이전 버전과 호환되는 구문과 새 구문을 동시에 사용하여 지정할 수 없습니다. 예를 들어 WITH (DROP_EXISTING, ONLINE = ON)과 같이 이전 버전에서의 옵션 구문과 SQL Server 2005 옵션 구문을 함께 지정하면 실패합니다.
- XML 인덱스를 만드는 경우 옵션은 WITH (option_name = ON | OFF)를 사용하여 지정해야 합니다.

인덱스 옵션	설명
MAXDOP = 〈maxdop값〉	인덱스 작업에 대한 최대 병렬 처리 수준 구성 옵션을 재정의합니다.
PAD_INDEX = ON | OFF	인덱스 패딩을 지정합니다.
FILLFACTOR = 〈fillfactor값〉	각 인덱스 페이지의 리프 수준을 어느 정도 채울지 나타내는 비율을 지정합니다.
SORT_IN_TEMPDB = ON | OFF	tempdb에 임시 정렬 결과를 저장할지 여부를 지정합니다.
STATISTICS_NORECOMPUTE = ON | OFF	배포 통계를 다시 계산할지의 여부를 지정합니다.
DROP_EXISTING = ON | OFF	인덱스 정의가 변경되지 않은 경우에 기존 제약 조건을 보존한 상태에서 인덱스를 다시 만듭니다.
ALLOW_ROW_LOCKS = ON | OFF	인덱스에 액세스할 때 행 잠금 허용 여부를 지정합니다.
ALLOW_PAGE_LOCKS = ON | OFF	인덱스에 액세스할 때 페이지 잠금 허용 여부를 지정합니다.
IGNORE_DUP_KEY = ON | OFF	중복 키 발생 시 처리 방식을 지정합니다.
ONLINE = ON | OFF	인덱스 작업의 온라인 수행여부를 지정합니다.

■ ONLINE 옵션

SQL Server 2005에서는 온라인으로 인덱스를 만들고 다시 작성하고 삭제할 수 있습니다. ONLINE 옵션을 사용하면 인덱스 작업이 실행되는 동안 테이블이나 클러스터형 인덱스 데이터 및 모든 관련 비클러스터형 인덱스에 동시에 액세스할 수 있습니다. 온라인 인덱스 작업 기능은 1년 365일, 하루 24시간 운영되는 비즈니스 환경에서 유용한 기능입니다. 온라인 인덱스 작업은 SQL Server 2005 Enterprise Edition 에서만 사용할 수 있습니다.

[따라하기] 온라인으로 인덱스 만들기

```
CREATE UNIQUE INDEX NX_UnitMeasure_Name
   ON AdventureWorks.Production.UnitMeasure(Name)
   WITH (ONLINE = ON);
GO
```

■ MAXDOP 옵션

인덱스 생성 시 MAXDOP 옵션을 지정하면, 데이터베이스 엔진의 max degree of parallelism 설정값이 무시되며 인덱스 작업에 대하여 최대 병렬 처리 수준 구성 옵션을 원하는 대로 재정의할 수 있습니다. MAXDOP 옵션을 지정하지 않으면 SQL Server 구성 옵션에 지정된 max degree of parallelism 값을 기준으로 인덱스 생성 작업에 사용될 최대 프로세서 개수가 결정됩니다. MAXDOP 옵션을 사용하여 병렬 계획 실행에 사용되는 프로세서 수를 제한할 수 있으며 프로세서의 최대값은 64개입니다. 병렬 인덱스 작업은 SQL Server 2005 Enterprise Edition 에서만 지원됩니다.

MAXDOP 옵션은 다음과 같은 경우에 유용합니다.

- 서버 차원의 MAXDOP 값이 1인 경우에 업무 부하가 없거나 적은 시점에 인덱스 관리 작업을 수행할 때 MAXDOP 값을 재정의함으로써 보다 빨리 인덱스 작업이 실행될 수 있도록 하기 위해서 사용할 수 있습니다.
- 매우 큰 인덱스를 생성, 다시 작성 및 삭제하는 작업에서 리소스가 많이 소모되어 인덱스 작업 중에 다른 응용 프로그램 또는 데이터베이스 작업이 사용할 리소스가 부족하지 않도록 하기 위하여 인덱스 작업에 사용할 프로세서 수를 서버 차원의 MAXDOP 값보다 적게 제한하기 위하여 사용될 수 있습니다.

다음은 MAXDOP 인덱스 옵션에 지정할 수 있는 유효한 값입니다.

MAXDOP 값	설명
MAXDOP = 0 (기본값)	현재 시스템의 작업부하에 따라 사용 가능한 CPU 수만큼 사용합니다.
MAXDOP = 1	병렬 계획이 생성되지 않습니다.
MAXDOP = 2 ~ 64	지정된 값으로 인덱스 작업에 사용될 프로세서 수를 제한합니다. 사용 가능한 CPU 수보다 더 큰 수를 지정하면 사용 가능한 실제 CPU 수가 사용됩니다.

[따라하기] 병렬 처리 수준 지정하기

```
CREATE UNIQUE INDEX NX_UnitMeasure_Name
   ON AdventureWorks.Production.UnitMeasure(Name)
   WITH (MAXDOP = 0);
GO
```

■ IGNORE_DUP_KEY 옵션

IGNORE_DUP_KEY 옵션을 활성화하면 다중 행 INSERT 작업 시 중복 키 값이 들어올 때 경고 메시지를 반환하고 고유 인덱스에 위배되는 행만 INSERT에서 제외시킵니다. IGNORE_DUP_KEY 옵션이 비활성화되어 있는 상태에서는 중복 키 값이 들어오면 오류 메시지가 반환되고 전체 INSERT 트랜잭션이 롤백됩니다.

이 옵션은 기본 모드가 중복 키 무시인 이기종 DBMS에서 SQL Server 2005로 마이그레이션한 시스템에 유용합니다.

[따라하기] IGNORE_DUP_KEY 옵션 사용하기

```
CREATE TABLE #Test (C1 nvarchar(10), C2 nvarchar(50), C3 datetime);
GO
CREATE UNIQUE INDEX AK_Index ON #Test (C2)
   WITH (IGNORE_DUP_KEY = ON);
GO
INSERT INTO #Test VALUES (N'OC', N'Ounces', GETDATE( ));
INSERT INTO #Test
SELECT * FROM Adventureworks.Production.UnitMeasure;
GO
/*
중복 키가 무시되었습니다.
*/
DROP TABLE #Test;
GO
```

인덱스 수정

ALTER INDEX 문을 사용하여 인덱스를 비활성화, 다시 작성 또는 다시 구성하거나 인덱스에 대한 옵션을 설정하여 기존 테이블 또는 뷰 인덱스(관계형 또는 XML)를 수정할 수 있습니다.

인덱스를 다시 분할하거나 다른 파일 그룹으로 이동하는 데는 ALTER INDEX를 사용할 수 없으며, ALTER INDEX 문을 사용하여 열 추가 또는 삭제, 열 순서 변경과 같은 인덱스 정의를 수정할 수도 없습니다. 이러한 작업을 수행하려면 DROP_EXISTING 절에 CREATE INDEX를 사용합니다.

옵션을 명시적으로 지정하지 않으면 현재 설정이 적용됩니다. 예를 들어 REBUILD 절에 FILLFACTOR 설정을 지정하지 않으면 다시 작성하는 동안 시스템 카탈로그에 저장된 채우기 비율 값이 사용됩니다. sys.indexes를 사용하면 현재의 인덱스 옵션 설정 정보를 확인할 수 있습니다.

ALTER INDEX 문을 사용하여 다음과 같은 작업을 수행할 수 있습니다. 인덱스 재구성, 인덱스 재작성, 인덱스 활성화 및 비활성화는 이 책의 뒷부분에 보다 자세하게 설명되어 있습니다.

- 인덱스 옵션 설정
- 인덱스 재구성
- 인덱스 재작성
- 인덱스 활성화 및 비활성화
- 제약 조건 활성화 및 비활성화

[따라하기] 인덱스 옵션 설정하기

```
USE AdventureWorks;
GO
ALTER INDEX AK_SalesOrderHeader_SalesOrderNumber ON
   Sales.SalesOrderHeader
SET (
   STATISTICS_NORECOMPUTE = ON,
   IGNORE_DUP_KEY = ON,
   ALLOW_PAGE_LOCKS = ON
   ) ;
GO
```

[따라하기] LOB 압축을 사용하여 인덱스 다시 구성하기

```
USE AdventureWorks;
GO
ALTER INDEX PK_ProductPhoto_ProductPhotoID ON
   Production.ProductPhoto REORGANIZE
   WITH (LOB_COMPACTION=ON);
GO
```

인덱스 조각화 정보 확인

테이블의 데이터가 INSERT, UPDATE, DELETE가 발생함에 따라서 조각화가 발생하게 되고, 따라서 테이블에 정의된 인덱스에도 조각화가 발생합니다. 조각화가 발생하면 원하는 데이터를 가져오기 위하여 읽어야 할 페이지 수가 늘어나, 성능에 좋지 않은 영향을 미칠 수 있으므로 주기적으로 조각화 제거 작업이 필요합니다.

조각화 정보를 알기 위하여 이전 버전에서 사용했던 DBCC SHOWCONTIG의 사용이 가능하지만, SQL Server 2005에서는 테이블의 조각화 정보를 sys.dm_db_index_physical_stats 동적 관리 뷰를 사용하여 확인할 수 있습니다.

[구문]

```
sys.dm_db_index_physical_stats (
    { database_id | NULL }          /* NULL: 서버내의 모든 데이터베이스 */
    , { object_id | NULL }          /* NULL: 해당 데이터베이스내의 모든 오브젝트 */
    , { index_id | NULL | 0 }       /* NULL: 해당 테이블의 모든 인덱스 */
    , { partition_number | NULL }   /* NULL: 해당 오브젝트의 모든 인덱스 */
    , { mode | NULL | DEFAULT }     /* NULL: LIMIT */
)
```

mode에는 DEFAULT, NULL, LIMITED, SAMPLED, DETAILED를 입력할 수 있으며, DEFAULT 또는 NULL은 LIMITED를 의미합니다.

LIMITED 모드는 가장 빠르고 가장 적은 페이지를 스캔합니다. 즉, 힙의 모든 페이지와 인덱스의 잎 레벨을 제외한 페이지를 스캔합니다.

SAMPLED 모드는 인덱스 또는 힙의 모든 페이지중에서 1% 샘플에 대한 통계를 반환합니다. 만일, 인덱스 또는 힙이 10,000 페이지보다 적다면 SAMPLED 대신에 DETAILED 모드가 사용됩니다.

DETAILED 모드는 모든 페이지를 스캔하고 모든 통계를 반환합니다.

[따라하기] 테이블의 조각화 정보 확인하기

```
DECLARE @db_id SMALLINT;
DECLARE @object_id INT;

SET @db_id = DB_ID(N'AdventureWorks');
SET @object_id = OBJECT_ID(N'AdventureWorks.Person.Address');
```

```
IF @db_id IS NULL
BEGIN;
   PRINT N'Invalid database';
END;
ELSE IF @object_id IS NULL
BEGIN;
   PRINT N'Invalid object';
END;
ELSE
BEGIN;
    SELECT * FROM sys.dm_db_index_physical_stats(@db_id, @object_id, NULL,
NULL , 'LIMITED');
END;
GO
```

인덱스 재구성

ALTER INDEX 문에 REORGANIZE 절을 사용하면 인덱스를 다시 구성할 수 있습니다. SQL Server 2000에서의 DBCC INDEXDEFRAG 문 대신 이 기능을 사용하는 것을 권고합니다. 인덱스를 다시 구성하면 리프 노드의 논리적 순서(왼쪽에서 오른쪽으로)와 일치하도록 리프 수준 페이지가 다시 정렬되어 테이블과 뷰의 클러스터형 및 비클러스터형 인덱스의 리프 수준이 조각 모음됩니다. 클러스터형 인덱스를 다시 구성하면 클러스터형 인덱스의 리프 수준에 포함된 모든 LOB 열이 압축됩니다. 비클러스터형 인덱스를 다시 구성하면 인덱스 내의 포괄 열인 모든 LOB 열이 압축됩니다.

sys.indexes 카탈로그 뷰의 채우기 비율 값을 기준으로 인덱스의 페이지를 압축하고, 압축으로 인해 생성된 빈 페이지는 제거됩니다. 재구성은 온라인으로 수행되며, 차단 잠금을 오래 보유하지 않으므로 쿼리나 업데이트의 실행을 차단하지 않습니다.

인덱스가 심하게 조각화되지 않은 경우에는 인덱스를 재구성하지만, 인덱스가 심하게 조각화된 경우에는 인덱스를 재작성하는 것이 좋습니다..

[따라하기] 인덱스 재구성하기

```
USE AdventureWorks;
GO
ALTER INDEX PK_ProductPhoto_ProductPhotoID ON Production.ProductPhoto
REORGANIZE ;
GO
```

인덱스 재작성

인덱스 조각화로 인한 성능 저하를 방지하기 위해서는 주기적으로 인덱스 조각화가 진행된 테이블들에 대한 조각화 제거 작업이 필요합니다. 조각화를 제거하지 위한 작업 중의 하나가 인덱스를 다시 작성하는 것입니다. 인덱스 재작성은 인덱스를 삭제한 후 다시 생성하는 작업으로, 인덱스를 논리적 정렬과 일치하도록 물리적으로 재정렬합니다. 인덱스가 만들어졌을 때 지정된 FILLFACTOR를 계산하여 인덱스의 페이지를 압축하여, 디스크 공간을 확보하고 필요한 만큼 새 페이지를 할당하여 인덱스 행을 연속되는 페이지에 다시 정렬합니다. 이 작업은 ALTER INDEX에 REBUILD 절을 사용하여 할 수 있으며, DBCC DBREINDEX 대신 이 기능을 사용하는 것을 권고합니다. 또한, CREATE INDEX에 DROP_EXISTING 절을 사용하여 인덱스 재작성 작업을 할 수 있습니다. 이 작업을 통하여, 요청한 데이터를 얻는 데 필요한 페이지 읽기 횟수를 줄일 수 있으므로 디스크 성능이 향상됩니다. ALL을 명시하면 하나의 트랜잭션으로 테이블의 모든 인덱스를 제거하고 다시 작성합니다. SQL Server 2005에서는 비클러스터형 인덱스를 온라인으로 재작성할 수 있습니다.

가능한 한 인덱스 재작성 작업을 자동화하여 주기적으로 용이하게 수행할 수 있는 체계를 갖출 것을 권고합니다.

ONLINE 옵션을 ON으로 설정하여 테이블의 인덱스를 재작성하면, 배타 테이블 잠금이 아주 짧은 시간 동안만 발생하기 때문에 데이터의 조회와 수정이 가능합니다. 온라인 인덱스 작업은 SQL Server 2005 Enterprise Edition에서만 가능합니다.

[따라하기] 비클러스터형 인덱스를 온라인상에서 재작성하기

```
ALTER INDEX IX_Employee_ManagerID ON HumanResources.Employee
REBUILD WITH (ONLINE=ON);
GO
```

인덱스 비활성화

인덱스 비활성화는 사용자가 인덱스를 사용할 수 없도록 하며, 제약 조건을 포함한 모든 인덱스에 사용 가능합니다. 즉, 비활성화된 인덱스는 유지 관리 되지 않고, 쿼리 옵티마이저에 의해 고려 되지도 않습니다. 비클러스터형 인덱스 또는 클러스터형 인덱스를 비활성화하면 인덱스의 메타 데이터는 시스템 카탈로그에 남기지만, 물리적으로 인덱스 데이터를 삭제합니다. 그러므로, 비활성화된 인덱스와 동일한 이름으로 인덱스를 생성할 수 없습니다.
클러스터형 인덱스의 경우는 비활성화하면 데이터는 삭제되지 않고 남아 있지만,
해당 테이블의 데이터에 접근할 수 없어, 데이터의 수정은 물론 조회도 불가능합니다.

인덱스의 비활성화 여부는 sys.indexes 카탈로그 뷰의 is_disabled 열을 확인합니다.
인덱스의 비활성화는 ALTER INDEX DISABLE을 사용하고, 비활성화된 인덱스를 다시 활성화 시키기 위해서는 ALTER INDEX REBUILD 또는 CREATE INDEX WITH DROP_EXISTING 구문을 사용합니다. 비활성화된 클러스터형 인덱스를 재작성할 때에는 ONLINE 옵션을 ON으로 설정할 수 없으며, 비활성화된 비클러스터형 인덱스를 재작성할 때에는 ONLINE 옵션을 ON으로 설정할 수 있습니다.

[따라하기] 비클러스터형 인덱스 비활성화한 후, 다시 활성화하기

```
USE AdventureWorks;
GO
-- 비클러스터형 인덱스 비활성화
ALTER INDEX IX_Employee_ManagerID ON HumanResources.Employee
DISABLE ;
GO
-- 비활성화된 인덱스 확인
SELECT is_disabled FROM sys.indexes
WHERE object_id=object_id('HumanResources.Employee')
AND name='IX_Employee_ManagerID';
/* 1 */
-- 온라인으로 인덱스 재작성
ALTER INDEX IX_Employee_ManagerID ON HumanResources.Employee
REBUILD WITH (ONLINE=ON);
GO
-- 재작성된 인덱스의 비활성화 여부 확인
SELECT is_disabled FROM sys.indexes
WHERE object_id=object_id('HumanResources.Employee')
AND name='IX_Employee_ManagerID';
GO
```

[따라하기] 클러스터형 인덱스 비활성화한 후, 다시 활성화하기

```
USE AdventureWorks;
GO
ALTER INDEX PK_Employee_EmployeeID ON HumanResources.Employee
DISABLE ;
```

```
GO
SELECT * FROM HumanResources.Employee;
GO
/*
메시지 8655, 수준 16, 상태 1, 줄 1
테이블 또는 뷰 Employee'의 인덱스 'PK_Employee_EmployeeID'이(가) 비활성화되
었으므로 쿼리 프로세서에서 계획을 생성할 수 없습니다.
*/
ALTER INDEX ALL ON HumanResources.Employee
REBUILD;
GO
```

[따라하기] FOREIGN KEY 제약 조건이 있는 클러스터형 인덱스 비활성화한 후, 활성화하기

```
USE Sample;
GO
-- 테스트 테이블 생성
CREATE TABLE STATES (STATE CHAR(2) NOT NULL);
GO
CREATE TABLE CITIES (
        CITY VARCHAR(30) NOT NULL,
        STATE CHAR(2) ,
        ZIP INT);
GO
-- PK 제약 조건과 FK 제약 조건 생성
ALTER TABLE STATES
ADD CONSTRAINT PK_STATES PRIMARY KEY CLUSTERED (STATE);
GO
```

```
ALTER TABLE CITIES
ADD CONSTRAINT PK_CITIES PRIMARY KEY CLUSTERED (CITY);
GO
ALTER TABLE CITIES
ADD CONSTRAINT FK_CITIES_STATES_STATE FOREIGN KEY (STATE)
REFERENCES STATES (STATE);
GO

-- CITIES 테이블에 데이터 INSERT
-- STATES 테이블에 없는 'PR' 값은 CITIES에 INSERT 할 수 없다
INSERT INTO STATES SELECT 'CA';
GO
INSERT INTO CITIES SELECT 'Los Angles', 'CA',111;
GO
INSERT INTO CITIES SELECT 'San Juan', 'PR' ,222;
GO
/*
메시지 547, 수준 16, 상태 0, 줄 1
INSERT 문이 FOREIGN KEY 제약 조건 "FK_CITIES_STATES_STATE"과(와) 충돌했
습니다. 데이터베이스 "sample", 테이블 "dbo.STATES", column 'STATE'에서 충돌
이 발생했습니다.
문이 종료되었습니다.
*/

-- STATES 테이블의 클러스터형 인덱스 비활성화
-- FOREIGN KEY 제약 조건도 동시에 비활성화된다
ALTER INDEX PK_STATES ON STATES
DISABLE;
GO
```

```
/*
경고: 인덱스 'PK_STATES'을(를) 비활성화한 결과 테이블 'STATES' 을(를) 참조하는
테이블 'CITIES'의 외래 키 'FK_CITIES_STATES_STATE'이(가) 비활성화되었습니다.
*/

-- CITIES 테이블에 데이터 INSERT
-- (FOREIGN KEY 제약 조건이 비활성화되었으므로 STATES 테이블에 없는 'PR' 값
을 INSERT 할 수 있다)
INSERT INTO CITIES SELECT 'San Juan', 'PR' ,222;
GO

-- STATES 테이블의 클러스터형 인덱스 재작성
ALTER INDEX PK_STATES ON STATES
REBUILD;
GO
-- CITIES 테이블에 데이터 INSERT
-- (FOREIGN KEY 제약 조건은 여전히 비활성화 상태이므로 STATES 테이블에 없는
'PR' 값을 INSERT 할 수 있다)
INSERT INTO CITIES SELECT 'Detroit', 'PR' ,333;
GO

-- FOREIGN KEY 제약 조건 재생성하기
ALTER TABLE  CITIES
CHECK CONSTRAINT FK_CITIES_STATES_STATE;
-- CITIES 테이블에 데이터 INSERT
-- (FOREIGN KEY 제약 조건이 재생성되었으므로 STATES 테이블에 없는 'PR' 값을
INSERT 할 수 없다)
INSERT INTO CITIES SELECT 'Kenmore', 'PR' ,444;
```

```
GO
/*
메시지 547, 수준 16, 상태 0, 줄 1
INSERT 문이 FOREIGN KEY 제약 조건 "FK_CITIES_STATES_STATE"과(와) 충돌했습니다. 데이터베이스 "sample", 테이블 "dbo.STATES", column 'STATE'에서 충돌이 발생했습니다.
문이 종료되었습니다.
*/
```

[참고]

PRIMARY KEY 제약 조건을 FOREIGN KEY 제약 조건이 참조하고 있는 경우, 인덱스가 비활성화되면 FOREIGN KEY 제약 조건도 동시에 비활성화됩니다. 그러나, PRIMARY KEY 제약 조건을 재구성했다고 하더라도 FOREIGN KEY 제약 조건은 재구성되지 않으므로, PRIMARY KEY 제약 조건을 재구성한 후 수동으로 FOREIGN KEY 제약 조건을 재생성해야 합니다.

인덱스 삭제

ALTER INDEX로는 인덱스를 구성하는 열을 변경하거나, 파일 그룹을 변경할 수 없습니다. 이와 같은 작업을 하고자 하는 경우에는 인덱스를 삭제한 후 다시 생성해야 합니다.

[구문] DROP INDEX 〈table_name〉.〈index_name〉

[따라하기] 인덱스 삭제하기

```
DROP INDEX HumanResources.Employee.IX_Employee_ManagerID;
GO
```

인덱스 사용 현황 확인

sys.dm_db_index_usage_stats 동적 관리 뷰를 사용하여 사용된 인덱스와 그 빈도를 알 수 있습니다. 이 뷰를 사용하여 응용 프로그램에서 전혀 사용하지 않는 인덱스를 확인할 수 있으며, 또한 유지 관리 오버헤드를 유발하는 인덱스를 확인할 수도 있습니다.

SQL Server 서비스를 시작할 때마다 카운터는 빈 상태로 초기화되며, 데이터베이스가 분리되거나 종료될 때마다(예: AUTO_CLOSE가 ON으로 설정된 경우) 데이터베이스와 관련된 모든 행이 제거됩니다.

[따라하기] Adventureworks 데이터베이스 인덱스 사용현황 확인하기

```
SELECT * FROM sys.dm_db_index_usage_stats
WHERE database_id = db_id('Adventureworks');
GO
```

[따라하기] 사용하지 않은 인덱스 확인하기

```
SELECT object_name(i.object_id) AS Table_Name, i.name AS Index_Name
FROM sys.indexes i, sys.objects o
WHERE  i.index_id NOT IN
        (SELECT s.index_id
         FROM sys.dm_db_index_usage_stats s
         WHERE s.object_id=i.object_id and
                        i.index_id=s.index_id and
                        database_id = db_id( ) )  -- dbid : db_id( ) 값을 입력
AND o.type = 'U'
AND o.object_id = i.object_id
ORDER BY object_name(i.object_id) ASC
GO
```

인덱스 I/O 및 잠금 정보

sys.dm_db_index_operational_stats 동적 관리 함수는 현재 데이터베이스 내의 테이블 또는 인덱스의 I/O, 잠금, 접근 방식을 반환합니다.

```
[구문] sys.dm_db_index_operational_stats (
        { database_id | NULL }
        , { object_id | NULL }
        , { index_id | NULL | 0 }
        , { partition_number | NULL }
    )
```

[따라하기] Adventureworks 데이터베이스내의 모든 인덱스에 대한 정보 확인하기

```
SELECT *
FROM sys.dm_db_index_operational_stats(db_id('AdventureWorks'),null,null,null);
GO
```

통계 갱신

SQL Server 데이터베이스 엔진은 각 인덱스의 키 값 배포에 관한 통계를 가지고 있으며, 이 통계를 사용하여 쿼리를 실행할 때에 사용할 인덱스를 결정합니다. 만약, 인덱싱 되어 있지 않은 열에 통계를 생성하려면, CREATE STATISTICS 문을 사용합니다.

많은 데이터의 추가, 변경 또는 삭제가 발생하여 인덱스의 키 값 분포가 많이 변경된 경우에는 해당 인덱스의 통계 갱신 즉, UPDATE STATISTICS를 실행하여 데이터베이스 엔진이 적절한 인덱스를 선택할 수 있도록 관리하는 작업이 필요합니다.

데이터베이스의 모든 사용자 정의 및 내부 테이블에 대해 UPDATE STATISTICS를 실행하기 위해서는 sp_updatestats를 실행합니다. sp_updatestats는 진행률을 나타내는 메시지를 표시하며, 업데이트가 완료되면 모든 테이블에 대해 통계가 업데이트되었다고 보고합니다. sp_updatestats는 비활성화된 비클러스터형 인덱스에 대한 통계는 업데이트하지만, 비활성화된 클러스터형 인덱스가 있는 테이블은 무시합니다.

명시적 또는 암시적 트랜잭션에서는 UPDATE STATISTICS가 허용되지 않습니다.
통계를 마지막으로 업데이트한 시기는 STATS_DATE 함수를 사용하여 확인 가능합니다.

[따라하기] 통계 업데이트하기

```
-- HumanResources.Employee 테이블의 모든 인덱스에 대한 통계 업데이트
USE AdventureWorks;
UPDATE STATISTICS HumanResources.Employee;
GO
-- HumanResources.Employee 테이블의 PK_Employee_EmployeeID 인덱스 통계 업데이트
USE AdventureWorks;
UPDATE STATISTICS HumanResources.Employee PK_Employee_EmployeeID;
GO
-- AdventureWorks 데이터베이스내의 모든 테이블의 통계 업데이트
USE AdventureWorks;
EXEC sp_updatestats;
GO
```

사용자 관리

로그인과 사용자

SQL Server에 접근하기 위해서는 로그인 계정이 필요하고, 데이터베이스에 접근하기 위해서는 사용자(user)가 필요합니다.

권한

사용자는 권한을 받아 작업을 수행할 수 있습니다. 권한에는 문장을 실행할 수 있는지에 따라 권한을 제한하는 명령문(Statement) 사용권한과 테이블, 색인, 뷰, 프로시저에 따라 권한을 제한하는 개체(Object) 사용권한이 있습니다.

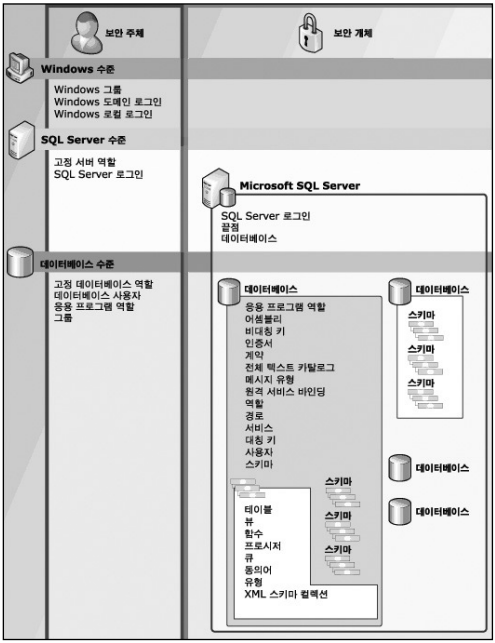
역할

역할은 로그인 또는 사용자들의 집합이며 각각의 역할에는 권한이 설정되어 있습니다.
각 사용자에게 직접 권한을 주는 것 보다는 필요한 권한이 있는 역할에 사용자를 추가시키는 것이 좋습니다. 시스템에서 미리 정의된 역할에는 로그인이 사용할 수 있는 서버 역할과 사용자가 사용할 수 있는 데이터베이스 역할이 있습니다

보안 주체와 보안 개체

보안 주체는 데이터베이스 사용자, 데이터베이스 역할, 응용 프로그램 역할, SQL Server 로그인 및 Windows 로그인 등을 말합니다. 모든 보안 주체는 고유한 SID(고유 보안 식별자)를 가지게 됩니다. 보안 개체는 테이블, 뷰, 프로시저와 같은 SQL 개체를 포함하는 스키마 개체 범위에 포함되는 구성 요소를 말합니다. 또한 스키마 및 사용자, 역할 등도 데이터베이스 범위에 포함이 되는 보안 개체이며, 데이터베이스, 로그인 등은 서버 범위에 포함이 되는 보안 개체입니다.

■ 데이터베이스 엔진 사용 권한 계층간의 관계



로그인 계정 생성 및 기본 스키마 지정

SQL Server 로그인 계정을 생성하고, 생성한 계정의 기본 스키마를 지정하는 예제입니다.

[따라하기] 로그인 및 사용자 생성하기

```
--dbadmin 로그인 계정을 생성합니다.
CREATE LOGIN dbadmin WITH PASSWORD = 'ad1234';
GO
-- dbadmin 로그인에 대한 사용자를 생성합니다.
-- 이 사용자의 기본 스키마는 MySchema로 설정합니다. 이때 해당 스키마는 미리
--  생성하지 않아도 됩니다.
CREATE USER UserA FOR LOGIN dbadmin
  WITH DEFAULT_SCHEMA = MySchema;
GO
-- MySchema 스키마를 생성하고 이 스키마의 소유권을 UserA 에 할당합니다.
CREATE SCHEMA MySchema AUTHORIZATION UserA;
GO
-- MySchema 내에서 Table1 테이블을 생성합니다.
CREATE TABLE MySchema.Table1 (seq int);
GO
```

기존 로그인과 사용자, 스키마 확인

[따라하기] 로그인 확인하기

```
EXEC sp_helplogins;
GO
```

[따라하기] 사용자 및 역할과 같은 보안 주체 확인하기

```
SELECT * FROM sys.database_principals;
GO
```

[따라하기] 현재 서버에 존재하는 스키마 정보 확인하기

```
-- 각 스키마는 자신을 소유하는 보안주체의 아이디 값(principal_id)을 가지고 있습니다.
SELECT * FROM sys.schemas;
GO
```

암호 변경

ALTER LOGIN 문을 사용하여 계정의 암호를 변경할 수 있습니다. 이전 암호를 알지 못하는 상태에서 로그인의 암호를 변경하기 위해서는 사용자가 CONTROL SERVER 권한을 가져야 합니다. 이 권한을 가지고 있는 경우에는 OLD_PASSWORD를 지정하지 않아도 암호를 변경할 수 있습니다.

[따라하기] 로그인 계정 'dbadmin'의 암호를 'ad1234' 에서 'ad5678'로 변경하기

```
ALTER LOGIN dbadmin WITH PASSWORD ='ad5678' OLD_PASSWORD='ad1234';
GO
```

[따라하기] 로그인 계정 'dbadmin'의 이전 암호를 모르는 상태에서 암호 변경하기

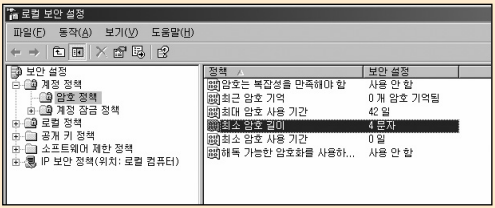
```
ALTER LOGIN dbadmin WITH PASSWORD ='ad5678';
GO
```

암호 정책

Windows Server 2003 이상의 운영체제에서 SQL Server 2005를 운영하는 경우, SQL Server는 Windows의 암호 정책을 사용할 수 있습니다.
로그인 생성 및 수정 시 CHECK_POLICY 옵션을 이용하면, Windows의 그룹 정책 사용 여부를 설정할 수 있습니다. 로그인 생성 시 MUST_CHANGE 옵션을 사용하면, 사용자가 최초 로그인 시 암호를 변경하도록 설정할 수 있습니다.

[따라하기] 암호 정책 변경하기

1. 관리 도구 → 로컬 보안 정책(secpol.msc)을 실행한 후, 계정 정책→ 암호 정책을 선택합니다.
2. 최소 암호 길이를 4문자로 설정합니다.



3. 쿼리 분석기에서 패스워드가 4자 이하인 경우, 다음과 같이 오류가 발생됩니다.

```
CREATE LOGIN dbadmin WITH PASSWORD ='ad4';
GO
/*
메시지15116, 수준16, 상태1, 줄1
암호의 유효성을 검사하지 못했습니다. 암호가 너무 짧아서 Windows 정책요구사항에
맞지 않습니다.
*/
```

4. CHECK_POLICY = OFF 옵션을 이용하여 수행하면, 오류 없이 생성됩니다.

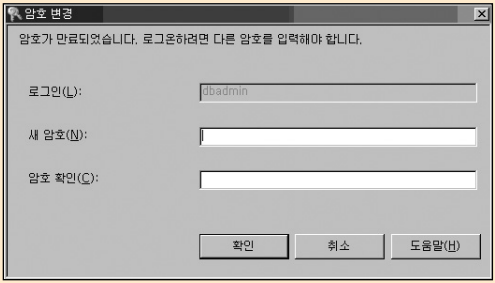
```
CREATE LOGIN dbadmin WITH PASSWORD ='ad4', CHECK_POLICY = OFF;
GO
```

[따라하기] 최초 로그인 시 암호를 변경하도록 설정하기

1. MUST_CHANGE 옵션을 이용하여 사용자가 최초 로그인 시 암호를 변경하도록 설정합니다.

```
CREATE LOGIN dbadmin
WITH PASSWORD ='adc1234' MUST_CHANGE, CHECK_EXPIRATION = ON;
GO
```

2. dbadmin으로 로그인 시 다음과 같은 변경 창이 나타납니다.



암호의 복잡성 및 암호 만료

Windows 2003 이상의 환경에서 암호 정책을 이용할 경우, 다음과 같은 기준의 암호 복잡성을 준수해야 합니다.

- 암호는 사용자 계정 이름의 일부 또는 전체를 포함할 수 없습니다.
- 암호의 길이는 최소 6자 이상이어야 합니다.
- 암호는 다음 4가지 범주 중 세 범주의 문자를 포함해야 합니다.

  - 알파벳 대문자 (A - Z)
  - 알파벳 소문자 (a - z)
  - 기본 숫자 10가지 (0 - 9)
  - 영숫자가 아닌 문자(예 : !, $, #, %)

암호 만료 정책을 사용하면 특정 기간 경과 시 사용자에게 기존 암호를 변경할 것과 암호가 만료되어 계정을 사용할 수 없게 됨을 알려 줍니다.

로그인 계정 비활성화

로그인을 활성화 또는 비활성화할 수 있습니다.

[따라하기] 로그인 계정 'dbadmin'을 비활성화하기

```
ALTER LOGIN dbadmin DISABLE;
GO
```

[따라하기] 비활성화된 로그인을 활성화하기

```
ALTER LOGIN dbadmin ENABLE;
GO
```

로그인 이름 변경

로그인의 이름을 변경할 수 있습니다.

[따라하기] 로그인 이름 변경하기

```
ALTER LOGIN dbadmin WITH NAME = dbadm;
GO
```

서버 및 데이터베이스 정보 확인

서버 이름, 버전, Edition 확인

서버 인스턴스에 대한 속성 정보를 반환하는 SERVERPROPERTY 함수를 사용합니다. SERVERPROPERTY 함수의 ServerName 속성과 @@SERVERNAME은 비슷한 정보를 반환하지만 결과가 다를 수도 있습니다. ServerName 속성은 Windows 서버 및 지정된 SQL Server 인스턴스에 대한 인스턴스 정보를 제공하고, @@SERVERNAME은 현재 구성된 로컬 서버 이름을 제공합니다. 만일 기본 서버 이름을 변경하지 않았다면 ServerName 속성과 @@SERVERNAME은 같은 정보를 반환합니다.

[구문] SERVERPROPERTY (propertyname)

[따라하기] SQL Server 인스턴스 속성 정보 확인하기

```
SELECT SERVERPROPERTY('ServerName') AS ServerName
, SERVERPROPERTY('MachineName') AS MachineName
, SERVERPROPERTY('InstanceName') AS InstanceName
, SERVERPROPERTY('Edition') AS Edition
, SERVERPROPERTY('ProductVersion') AS ProductVersion
, SERVERPROPERTY('ProductLevel') AS ProductLevel;
GO
```

기본 서버 이름을 바꾸지 않았다면 SERVERPROPERTY 함수의 ServerName 속성과 @@SERVERNAME은 같은 정보를 반환합니다. 보통 컴퓨터 이름이 변경된 경우에 로컬 서버의 이름을 동일하게 변경하고자 하여 sp_addserver를 사용하여 서버 이름을 변경합니다.

sp_addserver 는 원격 서버 또는 로컬 SQL Server 인스턴스 이름을 정의하는데 사용할 수 있습니다만, 원격 서버를 등록하는 경우에는 sp_addserver 대신 sp_addlinkedserver를 사용합니다.

sp_dropserver, sp_addserver 저장 프로시저를 사용하여 로컬 서버 이름을 변경한 경우에 SERVERPROPERTY('ServerName') 과 @@SERVERNAME의 결과가 달라지게 됩니다.

[따라하기] SQL Server를 실행하는 로컬 서버의 이름 확인하기

```
SELECT @@SERVERNAME;
GO
/*
TEST\SS2005
*/
EXEC sp_dropserver 'TEST\SS2005'
GO
/*
명령이 완료되었습니다.
*/

EXEC sp_addserver 'SAMPLE\SS2005', 'local'
GO
/*
명령이 완료되었습니다.
*/

SELECT @@SERVERNAME
GO
/*
TEST\SS2005
```

```
*/
SHUTDOWN
GO
/*
로그인 TEST\Administrator의 request에 의해 서버가 종료됩니다.
SQL Server가 이 프로세스를 종료합니다.
*/

-- SQL Server 서비스 재시작
SELECT @@SERVERNAME
GO
/*
SAMPLE \SS2005
*/
SELECT SERVERPROPERTY('ServerName')
GO
/*
TEST\SS2005
*/
```

카탈로그 뷰 및 호환성 뷰

카탈로그 뷰는 테이블, 인덱스 등과 같이 데이터베이스 엔진에서 사용하는 정보들을 조회할 수 있는 읽기 전용 뷰입니다. 카탈로그 뷰는 각 데이터베이스의 sys 스키마에 포함되어 있습니다. 또한 SQL Server 2005에서는 이전 버전에서 사용되었던 여러 시스템 테이블과의 호환성을 유지하기 위해 호환성 뷰 집합이 있습니다.

SQL Server 2000 시스템 테이블	SQL Server 2005 시스템 뷰
sysconfigures	sys.configurations
sysdevices	sys.backup_devices
syslockinfo	sys.dm_tran_locks
syslocks	sys.dm_tran_locks
syslogins	sys.server_principals
sysperfinfo	sys.dm_os_performance_counters
sysprocesses	sys.dm_exec_connections sys.dm_exec_sessions sys.dm_exec_requests
sysservers	sys.servers
syscolumns	sys.columns
syscomments	sys.sql_modules
sysdepends	sys.sql_dependencies
sysfilegroups	sys.filegroups
sysfiles	sys.database_files
sysforeignkeys	sys.foreign_keys
sysindexes	sys.indexes
sysindexkeys	sys.index_columns
sysmembers	sys.database_role_members
sysobjects	sys.objects
sysreferences	sys.foreign_keys
systypes	sys.types
sysusers	sys.database_principals

데이터베이스 파일에 대한 I/O 통계 정보 확인

sys.dm_io_virtual_file_stats 동적 관리 뷰는 데이터 및 로그 파일에 대한 I/O 통계 정보를 제공합니다. 사용자들이 어떤 파일에 대하여 읽기나 쓰기를 수행하기 위하여 기다린 시간을 제공하므로 이 뷰를 사용하면 어떤 파일들에 대하여 I/O가 많이 발생하는지 확인할 수 있습니다. I/O로 인한 성능 저하가 의심되는 경우에는 sys.dm_io_virtual_file_stats 뷰에서 io_stall 값을 점검할 것을 권고합니다.

[구문] sys.dm_io_virtual_file_stats(
{ database_id | NULL }
, { file_id | NULL }
)

[따라하기] 데이터베이스 파일의 I/O 정보 확인하기

```
-- 모든 데이터베이스의 모든 파일들의 I/O 정보 확인
SELECT DB_NAME(database_id) as DBName,
        File_ID,
        File_Name(File_ID) as FileName,
        IO_Stall,
        Num_Of_Reads, Num_Of_Writes,
        Num_Of_Bytes_Read, Num_Of_Bytes_Written
FROM sys.dm_io_virtual_file_stats (null, null)
ORDER BY IO_Stall DESC;
GO
-- 특정 데이터베이스의 모든 파일들에 대한 I/O 정보 확인 (예: tempdb)
SELECT * FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), null);
GO
-- 특정 데이터베이스의 특정 파일의 I/O 정보 확인 (예: tempdb의 Primary Data File)
```

```
SELECT * FROM sys.dm_io_virtual_file_stats (DB_ID('tempdb'), 1);
GO
-- 특정 데이터베이스의 특정 파일의 I/O 정보 확인 (예: tempdb의 로그 파일)
SELECT * FROM sys.dm_io_virtual_file_stats (DB_ID('tempdb'), 2);
GO
```

기본적인 파일 정보 확인

현재 데이터베이스와 연관된 파일의 물리적 이름과 특징을 반환합니다. 파일 이름을 지정하지 않으면, 데이터베이스내의 모든 파일에 대한 정보를 확인할 수 있습니다.

[구문] sp_helpfile [[@filename =] '*name*']

[따라하기] 데이터베이스 파일 정보 확인하기

```
USE Sample;
GO
EXEC sp_helpfile;
GO
-- 또는
SELECT * FROM Sample.sys.database_files;
GO
```

기본적인 파일 그룹 정보 확인

현재 데이터베이스와 연관된 파일그룹의 이름과 특징을 반환합니다. 파일그룹의 이름을 지정하지 않으면, 데이터베이스내의 모든 파일그룹에 대한 정보를 확인할 수 있습니다.

[구문] sp_helpfilegroup [[@filegroupname =] '*name*']

기본적인 데이터베이스 정보 확인

지정된 데이터베이스 또는 모든 데이터베이스 정보를 반환합니다.

[구문] sp_helpdb [[@dbname=] '*name*']

■ 데이터베이스를 지정하지 않으면 다음의 결과 집합이 반환됩니다.

열 이름	내용
name	데이터베이스 이름
db_size	데이터베이스의 총 크기
owner	데이터베이스의 소유자
dbid	데이터베이스의 ID
created	데이터베이스의 생성일자
status	옵션의 값을 쉼표로 분리하여 나열한 정보
compatibility_level	호환성 수준(60,65,70,80,90)

■ 데이터베이스를 지정하면 위의 결과 집합에 다음의 결과 집합이 추가로 나타납니다.

열 이름	내용
name	논리적 파일 이름
fileid	파일 ID
filename	파일의 물리적 경로와 이름
filegroup	파일이 속한 그룹, 로그 파일인 경우 NULL
size	파일 크기(MB)
maxsize	파일이 증가할 수 있는 최대 크기
growth	파일의 증가량
usage	파일의 용도

데이터베이스 옵션 또는 현재 설정 확인

DATABASEPROPERTYEX 함수를 사용하면 지정한 데이터베이스에 대하여 지정한 데이터베이스 옵션이나 속성의 현재 설정을 확인할 수 있습니다. DATABASEPROPERTYEX 함수는 지정한 Property에 대한 결과값을 반환하며 한번에 하나의 속성 설정만 조회 가능합니다. property의 값 목록과 그에 따른 결과의 종류는 온라인 설명서를 참조하기 바랍니다.

[구문] DATABASEPROPERTYEX (*database_name, property*)

sys.databases 카탈로그 뷰를 사용하면 한번에 여러 가지 속성 설정을 조회할 수 있으며, sp_dboption을 사용하여 데이터베이스에 현재 설정되어 있는 옵션을 확인할 수도 있습니다. 데이터베이스의 옵션 상태는 ALTER DATABASE를 이용하여 변경할 수 있습니다.

[따라하기] DATABASEPROPERTYEX를 사용하여 데이터베이스 옵션 설정 확인하기

```
-- AdventureWorks 데이터베이스의 통계 자동 업데이트 옵션 설정 확인
SELECT DATABASEPROPERTYEX ('AdventureWorks','IsAutoUpdateStatistics')
AS IsAutoUpdateStatistics;
GO
-- AdventureWorks 데이터베이스의 복구 모델 설정 확인
SELECT DATABASEPROPERTYEX ('AdventureWorks','Recovery') AS Recovery;
GO
```

[따라하기] sys.databases 뷰를 사용하여 데이터베이스 옵션 설정 확인하기

```
-- 모든 데이터베이스의 설정 옵션 목록 확인
SELECT * FROM sys.databases;
GO
-- AdventureWorks 데이터베이스의 설정 옵션을 확인
SELECT * FROM sys.databases WHERE Name ='AdventureWorks';
GO
-- AdventureWorks 데이터베이스의 통계 자동 생성 및 업데이트 옵션 설정 확인
SELECT name, is_auto_create_stats_on, is_auto_update_stats_on FROM
sys.databases
WHERE name='AdventureWorks';
GO
-- AdventureWorks 데이터베이스의 복구 모델 설정 확인
SELECT recovery_model, recovery_model_desc FROM sys.databases
WHERE name ='AdventureWorks';
GO
```

[따라하기] 데이터베이스의 현재 설정 옵션 확인하기

```
EXEC sp_dboption 'AdventureWorks';
GO
```

데이터베이스 공간 확인

시스템 카탈로그 sys. database_files를 이용하여 데이터베이스의 공간을 확인할 수 있습니다. 그러나, 대형 인덱스를 삭제하거나 다시 작성할 때 또는 대형 테이블을 삭제하거나 잘라낼 때 데이터베이스 엔진은 트랜잭션이 커밋될 때까지 실제 페이지 할당 취소 및 관련 잠금을 연기합니다. 삭제 작업이 지연되어도 할당된 공간이 즉시 해제되지는 않습니다. 따라서 대형 인덱스를 삭제하거나 잘라낸 직후 sys.database_files에서 반환한 값은 실제 사용할 수 있는 디스크 공간과 다를 수 있습니다. 동적 관리 뷰 sys.dm_db_file_space_usage를 이용하여 데이터베이스의 각 파일에 대한 공간 사용량 정보를 확인할 수도 있습니다.

[따라하기]

```
SELECT SUM(size)*1.0/128 AS [size in MB]
FROM tempdb.sys.database_files;
GO
```

[따라하기] 현재 데이터베이스의 여유 공간 정보 확인하기

```
SELECT SUM(unallocated_extent_page_count) AS [free pages],
(SUM(unallocated_extent_page_count)*1.0/128) AS [free space in MB]
FROM sys.dm_db_file_space_usage;
GO
```

테이블 크기 확인

자세한 내용은 [테이블 관리]의 [테이블 정보 확인]을 참조하기 바랍니다. sp_spaceused 저장 프로시저를 실행할 경우, 테이블을 지정하지 않으면 해당 데이터베이스의 크기가 반환됩니다.

[따라하기] Sales.Store 테이블의 크기 확인하기

```
EXEC sp_spaceused [Sales.Store];
GO
```

로그 공간의 사용 정보 확인

DBCC SQLPERF를 사용하면 서버 내에 존재하는 모든 데이터베이스의 트랜잭션 로그 공간의 사용에 관한 통계를 확인할 수 있습니다. 로그 공간 사용을 모니터링함으로써 적절한 로그 파일의 크기를 산정하거나 로그 백업 또는 잘라내기가 필요한 시점을 확인할 수 있습니다.

[따라하기] 데이터베이스별 로그 사용 공간 확인하기

```
DBCC SQLPERF ( LOGSPACE );
GO
```

테이블 조각화 정보 확인

테이블의 조각화는 INSERT, UPDATE, DELETE문 등의 데이터를 수정할 경우에 발생합니다. 이러한 수정들은 각 페이지의 채움 정도를 다르게 만들고, 테이블의 일부 또는 전부를 스캔하는 쿼리의 경우, 테이블 조각으로 인한 성능 저하가 발생할 가능성이 있습니다. 인덱스 관리의 인덱스 조각화를 참고하기 바랍니다.

```
[구문] sys.dm_db_index_physical_stats (
        { database_id | NULL }
        , { object_id | NULL }
        , { index_id | NULL | 0 }
        , { partition_number | NULL }
        , { mode | NULL | DEFAULT }
    )
```

오류 로그

SQL Server는 특정 시스템 이벤트와 사용자 정의 이벤트를 SQL Server 오류 로그 및 Microsoft Windows 응용 프로그램 로그에 기록합니다. SQL Server 오류 로그에 있는 정보를 사용하여 SQL Server와 관련된 문제의 원인을 찾을 수 있습니다.

문제의 원인을 확인하기 위해 SQL Server 오류 로그와 Windows 응용 프로그램 로그를 모두 사용할 수 있습니다. 예를 들어 SQL Server 오류 로그를 모니터링할 때 원인을 알 수 없는 오류 메시지가 표시될 수 있는 경우 두 로그 간의 이벤트에 대한 날짜와 시간을 비교하면 가능한 원인 목록을 좁혀갈 수 있습니다. SQL Server Management Studio 로그 파일 뷰어를 사용하면 SQL Server, SQL Server 에이전트 및 Windows 로그를 단일 목록으로 통합할 수 있어 관련된 서버 이벤트와 SQL Server 이벤트를 쉽게 이해할 수 있습니다.

■ **SQL Server 오류 로그 보기**

SQL Server Management Studio 또는 텍스트 편집기를 사용하여 SQL Server 오류 로그를 확인할 수 있습니다. SQL Server 로그는 응용 프로그램의 상태 정보를 알기 위한 유용한 자료이므로 주기적인 모니터링이 필요합니다.

SQL Server 로그는 서비스가 시작할 때부터 서비스가 중지될 때까지 계속 메시지를 기록하며, SQL Server가 시작될 때마다 새로운 오류 로그가 만들어집니다.

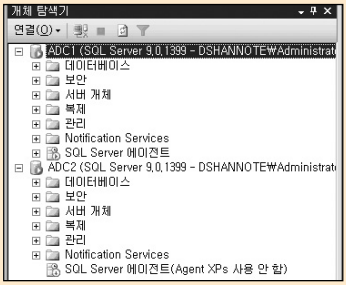
모니터링 관리의 효율을 위하여 모니터링 담당자가 SQL Server 로그에서 찾아야 할 것을 정의합니다. 이 로그는 심각도 수준 19~25의 값을 가진 모든 오류를 기록합니다. 모니터링 할 때, SQL Server 로그에서 심각도 수준 19~25 사이의 값을 가진 오류는 반드시 점검해야 합니다. 이 심각도 수준을 가진 오류가 발생하면 트랜잭션이 실패하게 하고 응용 프로그램이 제대로 동작하지 않기 때문입니다. 심각도 수준 20에서 25사이의 오류는 치명적이며, 만일 이 오류가 발생되면 클라이언트 연결은 오류 메시지를 받은 후에 종료됩니다.

TRY…CACHE 구문을 이용하여 심각도가 10 이상인 연결을 끊지 않는 모든 실행 오류를 Catch 할 수 있습니다.

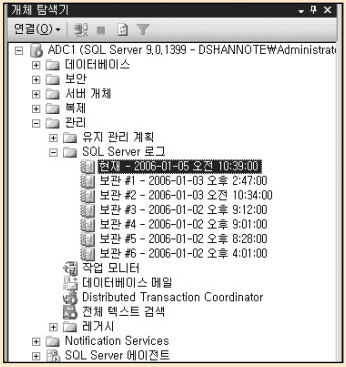
로그 파일 뷰어를 이용하면 날짜나 연결, 텍스트 문 등을 이용하여 필터를 설정해서 로그를 분석할 수 있으며, 텍스트 파일 및 CSV 형태로 로그 파일을 내보낼 수 있습니다.

[따라하기] 오류 로그 확인하기

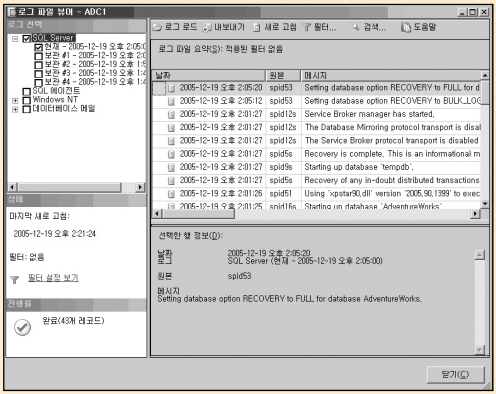
1. SQL Server Management Studio에서 원하는 데이터베이스 서버를 선택합니다.



2. [관리] 폴더를 클릭하고, [SQL Server 로그]를 클릭합니다.



3. 원하는 로그파일을 더블 클릭하면, 로그 파일 뷰어가 실행이 되면서 해당 로그 파일을 조회할 수 있습니다.



■ 로그 파일의 정보 확인 및 개수 변경하기

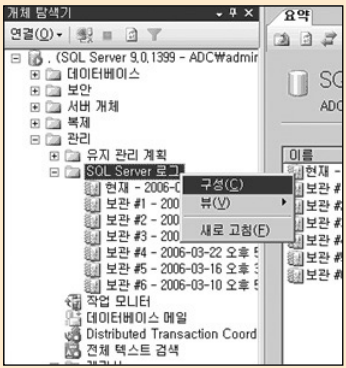
SQL Server 로그 파일의 개수는 현재 기록하고 있는 로그와 이전의 6개의 로그에 대한 백업을 가지고 있습니다. 이 로그 파일의 수를 크게 설정하면 SQL Server 재시작으로 인하여 문제를 진단하는데 단서가 될 수 있는 ERRORLOG 파일이 순환되어 유실되는 것을 방지할 수 있습니다. 이 로그의 개수는 SQL Server 로그 구성을 수정하여 변경할 수 있습니다.

[따라하기] 오류 로그 파일 개수 정보 확인 및 오류 로그 내용 보기

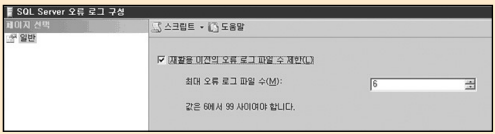
```
EXEC master..xp_enumerrorlogs ;
GO
EXEC sp_readerrorlog 3 ;
GO
```

[따라하기] SQL Server Management Studio에서 오류 로그 파일 개수 변경하기

1. [SQL Server 로그]에 커서를 위치시키고 마우스 오른쪽 버튼을 클릭합니다.
2. [구성] 메뉴를 선택합니다.



3. SQL Server 오류 로그 구성 창이 나타납니다. [재활용 이전의 오류 로그 파일 수 제한]을 체크합니다.



4. [최대 오류 로그 파일 수]에 원하는 오류 로그 파일 수를 입력한 후, [확인]을 클릭합니다.

■ **SQL Server 재시작 없이 새 오류 로그 생성하기**

SQL Server 가 시작될 때마다 현재 오류 로그의 이름은 errorlog.1로 변경되고, errorlog.1은 errorlog.2가 되고 errorlog.2는 errorlog.3이 되는 형식으로 순환됩니다. sp_cycle_errorlog를 사용하면 서버를 중지했다가 시작하지 않고 오류 로그 파일을 순환시킬 수 있습니다. 어떤 이유로 ERRORLOG 파일의 크기가 지나치게 커진 경우에는 sp_cycle_errorlog를 사용하여 새로운 ERRORLOG 파일을 생성할 것을 권고합니다.

[따라하기] 오류 로그 순환시키기

```
EXEC sp_cycle_errorlog;
GO
```

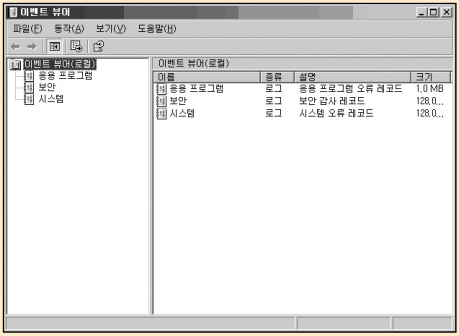
■ **응용 프로그램 로그 보기**

이벤트 뷰어는 사용자가 응용 프로그램, 보안, 시스템 로그에 기록되는 이벤트를 모니터링 할 수 있도록 합니다. 이 로그는 SQL Server 로그와 SQL 에이전트 로그로 분리시켜 추가적인 정보를 제공합니다. SQL Server 메시지는 응용 프로그램 로그에서 발견됩니다.

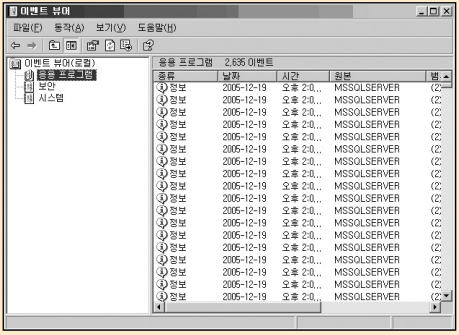
SQL Server 메시지는 "MSSQLSERVER" 또는 "SQLSERVERAGENT"라는 원본을 가진 메시지로 구별될 수 있습니다. RAISERROR 메시지도 여기에서 볼 수 있습니다.

[따라하기] 응용 프로그램 로그 확인하기

1. [시작] → [설정] → [관리도구] → [이벤트 뷰어]를 선택합니다. (운영체제에 따라 다름)



2. 디폴트로 로컬컴퓨터의 로그가 나타난다. 다른 컴퓨터의 [연결]을 클릭하여 원격 컴퓨터의 로그를 확인합니다.



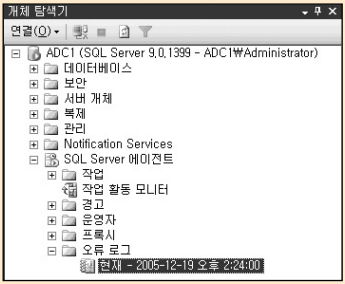
3. 원하는 사항을 더블 클릭하면, 자세한 정보를 얻을 수 있습니다.

■ **SQL Server 에이전트 오류 로그 보기**

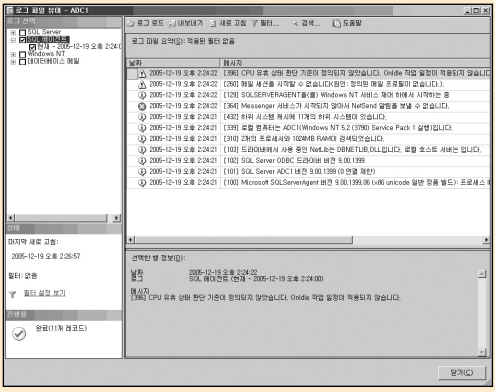
SQL Server 에이전트 서비스가 관리하는 작업이나 복제 상황을 모니터링하기 위하여 SQL Server 에이전트 오류 로그도 주기적인 모니터링이 필요합니다.

[따라하기] SQL Server 에이전트 오류 로그 보기

1. SQL Server Management Studio에서 원하는 데이터베이스 서버를 선택하고 해당 인스턴스를 확장합니다.
2. [SQL Server 에이전트]를 선택합니다.



3. [오류 로그]를 확장한 다음에 보고자 하는 오류 로그를 마우스 오른쪽 단추로 클릭하고 [에이전트 로그 보기]를 선택합니다.



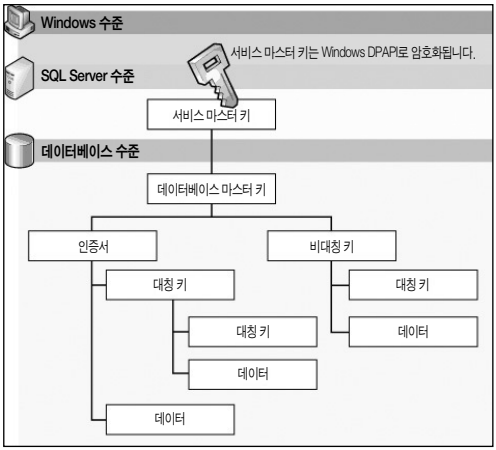
4. 필요에 따라 [필터] 버튼을 클릭한 다음 [필터 설정] 대화 상자에 매개 변수 값을 입력하여 로그 내용을 필터링합니다.
5. 필터 매개 변수를 선택한 경우 [필터 적용] 체크 박스를 선택한 다음 [필터 설정] 대화 상자에서 [확인]을 클릭합니다.
6. 로그 파일 요약에서 로그 내용을 봅니다.

암호화

암호화에는 대칭 암호화와 비대칭 암호화가 있고, SQL Server에서 사용되는 암호화 알고리즘은 다음과 같습니다.

- DES
- TRIPLE_DES
- RC2
- RC4
- DESX
- AES_128
- AES_192
- AES_256

다음은 암호화의 계층 구조입니다.



대칭 암호화

대칭 키를 사용하여 데이터를 암호화하는 구문입니다. 이 키는 열려 있어야 합니다.

[구문] EncryptByKey(key_GUID , { 'cleartext' | @cleartext }
[, { add_authenticator | @add_authenticator }
, { authenticator | @authenticator }])

대칭키를 사용하여 암호화한 데이터를 복호화하는 구문입니다.

[구문] DecryptByKey({ 'ciphertext' | @ciphertext }
[, add_authenticator
, { authenticator | @authenticator }])

암호화를 해제하지 않고도 암호화된 데이터 자체를 복사한다면 데이터를 수정할 수 있습니다. 이러한 값 자체의 대체 공격은 암호화를 무시합니다. 암호화하기 전에 컨텍스트 정보를 일반 텍스트에 추가하여 전체 값 대체 공격을 방지할 수 있습니다. 이러한 컨텍스트 정보는 일반 텍스트 데이터가 이동되지 않았는지 확인하는 데 사용됩니다.

데이터 암호화 시 인증자 매개 변수가 지정된 경우 DecryptByKey를 사용하여 데이터의 암호를 해독하려면 같은 인증자가 필요합니다. 암호화 시 인증자의 해시는 일반 텍스트와 함께 암호화됩니다. 암호 해독 시 같은 인증자가 DecryptByKey에 전달되어야 합니다. 두 값이 일치하지 않으면 해독에 실패합니다. 이것은 암호화 이후에 해당 값이 이동되었음을 의미합니다. 결과가 저장될 테이블의 기본 키를 이 매개 변수의 값으로 사용하는 것이 좋습니다.

대칭 암호화 및 암호 해독은 비교적 속도가 빠르며 대량의 데이터 작업 시 적합합니다.

[따라하기] 대칭 암호화 구현하기

다른 컨텍스트 id를 사용하여 ssn을 암호화하여 테이블을 생성하고, 뷰를 통해 복호화된 데이터를 보여준 후, ssn을 복사한 후 복호화된 데이터를 확인합니다. 그 후, 컨텍스트 id도 복사한 후 데이터를 확인하고, 키를 닫은 후에 복호화된 데이터를 확인하는 시나리오입니다.

1. 사원 정보를 저장하기 위한 임시 테이블을 생성합니다.

```
CREATE TABLE Employees (
            id int primary key,
             name varchar(100),
             ssn varbinary(128));
GO
```

2. ssn의 암호화를 위한 키를 생성합니다.

```
CREATE SYMMETRIC KEY SSN_Key_01
   WITH ALGORITHM = AES_256
   ENCRYPTION BY CERTIFICATE Employees001;
GO
```

3. 키를 열고 확인합니다.

```
OPEN SYMMETRIC KEY SSN_Key_01
  DECRYPTION BY CERTIFICATE Employees001;
GO
SELECT * FROM sys.openkeys;
GO
```

4. 데이터를 입력합니다.

```
INSERT INTO Employees VALUES
(101,'Mary',EncryptByKey(Key_GUID('SSN_Key_01'),'7204042030213',1,'101'));
```

```
INSERT INTO Employees VALUES
(102,'Jane',EncryptByKey(Key_GUID('SSN_Key_01'), '7007022043247',1,'102'));
GO
```

5. ssn 칼럼이 암호화되어 있는지 확인합니다.

```
SELECT * FROM Employees;
GO
/* 결과 예:
101     Mary        0x0063C891A5E8C143B1AAC765688FEE15010000008
102     Jane        0x0063C891A5E8C143B1AAC765688FEE1501000000A
*/
```

6. 복호화를 위한 뷰를 생성합니다.

```
CREATE VIEW View_Employees
AS
SELECT id, name, convert(varchar(13), DecryptByKey(ssn, 1, convert(varchar(30),id)))
as ssn
FROM Employees;
GO
```

7. 뷰를 통한 복호화를 확인합니다.

```
SELECT * FROM View_Employees;
GO
/*
101     Mary        7204042030213
102     Jane        7007022043247
*/
```

8. id가 102인 ssn을 id 101의 ssn으로 수정합니다.

```
UPDATE Employees
SET ssn = (SELECT ssn FROM Employees WHERE id = 101)
WHERE id = 102;
GO
```

9. id 102의 ssn을 수정한 후, 뷰를 통한 복호화를 확인합니다. id를 사용하여 암호화를 하였기 때문에 복호화가 실패합니다.

```
SELECT * FROM View_Employees;
GO
/* 결과 예:
101     Mary        7204042030213
102     Jane        NULL
*/
```

10. id 102인 행을 id도 101로 수정합니다. ssn을 수정한 후, id도 ssn을 암호화하기 위해 사용되었던 id로 수정하였기 때문에 뷰를 통한 복호화가 성공합니다.

```
UPDATE Employees SET id = 103 WHERE id = 101;
UPDATE Employees SET id = 101 WHERE id = 102;
GO
SELECT * FROM View_Employees;
GO
/*
101     Jane        7204042030213
103     Mary        NULL
*/
```

11. 키를 닫고 키가 닫혔는지 확인합니다.

```
CLOSE SYMMETRIC KEY SSN_Key_01;
GO
```

```
SELECT * FROM sys.openkeys;
GO
SELECT * FROM View_Employees;
GO
```

비대칭 암호화

인증서의 공개 키를 사용하여 데이터를 암호화합니다.

[구문] EncryptByCert (certificate_ID , {'cleartext' | @cleartext })

이 함수는 인증서의 공개 키를 사용하여 데이터를 암호화합니다. 이렇게 암호화된 텍스트는 이에 대응하는 개인 키로만 해독할 수 있습니다. 이 방법은 대칭 키를 사용한 보다 높은 보안 수준을 제공하지만 암호화 및 암호 해독에 비해 비용이 많이 들기 때문에, 크기가 큰 데이터를 작업할 경우에는 비대칭 암호화를 사용하지 않는 것이 좋습니다.

[따라하기]

```
INSERT INTO [AdventureWorks].[ProtectedData04]
   VALUES( N'data encrypted by certificate ''Shipping04''',
   EncryptByCert(Cert_ID('JanainaCert02'), @cleartext) );
GO
```

다음은 인증서의 개인 키로 데이터의 암호를 해독하는 구문입니다.

[구문] DecryptByCert (certificate_ID ,
{'ciphertext' | @ciphertext }
[, {'cert_password' | @cert_password }])

DDL 트리거

DDL 트리거는 UPDATE, DELETE, INSERT 등과 같은 명령문에 작동하는 DML 트리거와 달리 테이블이나 뷰에 대한 CREATE, ALTER 및 DROP 또는 사용자 계정이나 로그인 설정, 프로시저 생성 및 변경, 파티션 생성 및 변경 등과 같은 DDL문에 대하여 동작하는 트리거입니다.

다음과 같은 경우에 DDL 트리거를 활용할 수 있습니다.

- 데이터베이스 스키마에 대한 특정 변경 작업을 방지하려는 경우
- 데이터 스키마가 변경될 때 데이터베이스에서 특정 작업이 수행되도록 하려는 경우
- 데이터베이스 스키마의 변경 내용이나 이벤트를 기록하려는 경우

DDL 트리거는 SQL 문이 완료된 후에 실행이 되며, INSTEAD OF 트리거로 사용될 수는 없습니다. 또한 DML 트리거와 같이 inserted, deleted 테이블을 생성하지는 않습니다. DDL 트리거는 서버에 대해서 설정할 수도 있고 특정 데이터베이스에서만 수행되도록 설정할 수도 있습니다. 데이터베이스, 사용자, 끝점, 로그인 관련 이벤트는 서버 범위의 이벤트 그룹이며, 테이블, 뷰, 인덱스 등과 같은 데이터베이스 개체 관련 이벤트는 데이터베이스 범위의 이벤트 그룹입니다.

DDL 트리거를 디자인하기 전에 다음 사항이 필요합니다.

- DDL 트리거 영역에 대하여 이해해야 합니다.
- 어떤 Transact-SQL문(들)에 대하여 트리거를 발생시킬 것인지를 결정해야 합니다.

DDL 트리거 생성

[구문] CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[WITH 〈ddl_trigger_option〉 [,...n]]
{ FOR | AFTER } { event_type | event_group } [,...n]
AS { sql_statement [;] [,...n] | EXTERNAL NAME 〈 method specifier 〉 [;] }

[따라하기] 테이블의 DROP 및 ALTER 작업에 대하여 DDL 트리거 생성하기

```
USE AdventureWorks;
GO
IF EXISTS (SELECT * FROM sys.triggers WHERE parent_class = 0 AND name =
'safety')
    DROP TRIGGER safety ON DATABASE;
GO

CREATE TRIGGER safety
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
    PRINT '테이블을 변경/삭제하려면 "safety" 트리거를 비활성화 하세요.'
    ROLLBACK;
GO
-- safety라는 DDL 트리거를 비활성화합니다.
DISABLE TRIGGER safety ON DATABASE;
GO
-- safety라는 DDL 트리거를 활성화합니다.
ENABLE TRIGGER safety ON DATABASE;
GO
```

[따라하기] AdventureWorks 데이터베이스 내의 모든 DDL 문에 대하여, 사용 기록 남기기

```
USE AdventureWorks;
GO
CREATE TABLE ddl_log (PostTime datetime, DB_User nvarchar(100), Event
nvarchar(100), TSQL nvarchar(2000));
GO
CREATE TRIGGER log
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
DECLARE @data XML
SET @data = EVENTDATA( )
INSERT ddl_log (PostTime, DB_User, Event, TSQL) VALUES
    (GETDATE( ),
    CONVERT(nvarchar(100), CURRENT_USER),
    @data.value('(/EVENT_INSTANCE/EventType)[1]', 'nvarchar(100)'),
    @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'nvarchar(2000)') ) ;
GO
--생성한 트리거 테스트
CREATE TABLE TestTable (a int); --임시 테이블을 생성
DROP TABLE TestTable ;  --생성한 임시 테이블 삭제
GO

--DDL 로그 확인
SELECT * FROM ddl_log ;
GO
--트리거 삭제
DROP TRIGGER log ON DATABASE;
GO
```

```
--ddl_log 테이블 삭제
DROP TABLE ddl_log;
GO
```

DDL 트리거 정보 확인

동적 관리 뷰를 이용하여 트리거 정보를 확인합니다.

[따라하기] 데이터베이스 수준의 DDL 트리거 목록 확인하기

```
SELECT * FROM sys.triggers WHERE parent_class = 0;
GO
```

[따라하기] 서버 수준의 DDL 트리거 목록 확인하기

```
SELECT * FROM sys.server_triggers;
GO
```

[따라하기] 트리거 정의 확인하기

```
SELECT tr.name, sm.definition
FROM sys.triggers tr JOIN sys.sql_modules sm ON tr.object_id = sm.object_id
WHERE tr.parent_class = 0;
GO
```

관리자 전용 연결

관리자 전용 연결 목적

SQL Server 2005에서는 서버에 장애가 발생하여 일반적인 연결이 불가능할 때 관리자가 진단을 하기 위한 목적으로 접근할 수 있는 관리자 전용 연결(DAC: Dedicated Administrator Connection)을 제공합니다. DAC를 이용하여 관리자는 서버의 문제를 진단하며, 문제를 발생시키는 프로세스를 종료시키거나 데이터베이스 설정 변경과 같은 응급 복구 작업을 수행할 수 있습니다.

관리자 전용 연결 사용 제한

관리자 전용 연결은 서버에서 장애가 발생했을 때 문제 진단 및 응급 작업 만을 위한 연결이기 때문에 다음과 같은 제한이 있습니다.

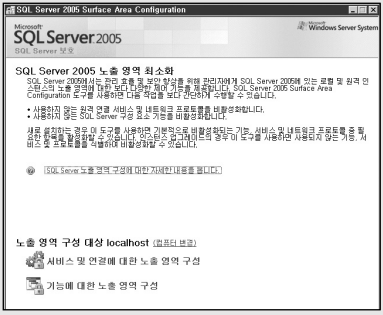
- 인스턴스당 하나의 DAC만 허용됩니다.
- DAC로 연결할 때 우선 로그인의 기본 데이터베이스에 연결을 시도한 후 해당 데이터베이스에 정상적으로 연결이 되면 master 데이터베이스를 이용할 수 있게 됩니다. 만약 접속하는 로그인의 기본 데이터베이스가 오프라인이거나 사용할 수 없는 경우 다음과 같은 명령어를 사용하여 기본 데이터베이스를 무시하고 master 데이터베이스에 직접 연결하도록 할 수 있습니다.

  sqlcmd -A -d master
- DAC를 사용하여 병렬 쿼리 또는 명령을 실행할 수 없습니다. 예를 들면, 백업(BACKUP), 복원(RESTORE)과 같은 명령은 사용할 수 없습니다.
- DAC는 제한된 리소스만 사용하기 때문에 많은 리소스가 필요한 쿼리 등은 수행해서는 안됩니다.
- DAC는 기본적으로 서버에서 실행되는 클라이언트에서만 연결이 허용됩니다. 만일 서버의 로컬이 아닌 원격에서 DAC를 사용하려면 SQL Server 구성 옵션인 'remote admin connections' 설정 값을 1로 변경해 주어야 합니다.

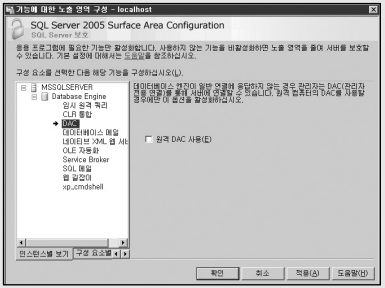
[참고] 원격 DAC 사용 허용하기

SQL Server 구성 옵션의 변경뿐만 아니라 SQL Server Configuration Manager에서도 원격 DAC 사용을 허용할 수 있습니다.

1. [시작] → [프로그램] → [Microsoft SQL Server 2005] → [구성 도구] → [SQL Server Configuration Manager]을 선택합니다.



2. 하단의 [기능에 대한 노출 영역 구성]을 선택합니다.



3. Database Engine의 DAC를 선택한 후, [원격 DAC 사용]의 체크 박스를 체크합니다.
4. [확인]을 클릭합니다.

관리자 전용 연결 사용

관리자 전용 연결은 sqlcmd나 SQL Server Management Studio를 통해 사용할 수 있습니다. sqlcmd를 이용하는 경우 DAC를 나타내는 -A를 사용하여 서버에 접속합니다. SQL Server Management Studio를 이용하는 경우 연결할 서버명 앞에 admin: 을 붙여서 DAC를 사용합니다.

[따라하기] sqlcmd를 이용하여 관리자 전용 연결 사용하기

명령 프롬프트 창에서 다음과 같이 접속합니다.

C:\> sqlcmd -S 〈*ServerName*〉 -U 〈*Login*〉 -P 〈*Password*〉 -A

[따라하기] SQL Server Management Studio를 이용하여 관리자 전용 연결 사용하기

서버 이름에 admin:〈ServerName〉으로 연결합니다.

로그 전달

로그 전달 고려 사항

일정 간격으로 하나의 데이터베이스(주 데이터베이스)에서 다른 데이터베이스(보조 데이터베이스)로 트랜잭션 로그를 전달하여 두 데이터베이스를 동기화하는 로그 전달(Log Shipping)을 구성할 수 있습니다.

로그 전달을 구성하기 위해서는 다음과 같은 전제 조건이 충족되어야 합니다.

- 로그 전달에 관련된 모든 서버 인스턴스에 SQL Server 2005 Standard Edition, SQL Server 2005 Workgroup Edition 또는 SQL Server 2005 Enterprise Edition 이 설치되어 있어야 합니다.
- 로그 전달에 관련된 모든 서버의 대/소문자 구분 설정이 동일해야 합니다.
- 원본 데이터베이스의 복구 모델은 전체 또는 대량 로그 복구 모델을 사용해야 합니다.
- 데이터베이스 미러링과 함께 로그 전달을 사용하는 경우 로그 전달 구성에 사용되는 현재 주 데이터베이스는 데이터베이스 미러링에 사용되는 현재 주 데이터베이스와 동일한 것이어야 합니다.
- 원본 데이터베이스의 Full Backup을 미리 받아 놓는 것이 로그 전달 구성 시 시간을 단축할 수 있습니다.
- 로그 전달을 구성하기 전에, 주 서버에 공유 폴더를 만들어 보조 서버에서 트랜잭션 로그 파일을 사용할 수 있도록 설정해야 합니다.
- 하나의 주 서버에서 여러 대의 보조 서버를 구성할 수 있습니다.

중요

Management Studio 로그 전달 도구로는 간단한 백업 및 복원만 처리할 수 있습니다. 파일 수가 많거나 기본 옵션 이외의 옵션이 사용된 데이터베이스와 같이 복잡한 BACKUP 또는 RESTORE 명령을 사용해야 하는 경우에는 수동 백업 및 복원을 사용합니다. 보조 데이터베이스가 복원되면 Management Studio 로그 전달 도구를 사용하여 로그 전달 설정을 완료합니다.

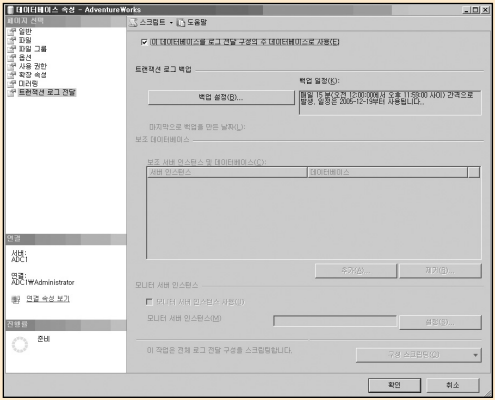
로그 전달을 구성하려면 각 서버 인스턴스에 대해 sysadmin 권한이 있어야 하며, 로그 전달 구성에서 백업 및 복원 디렉터리에 대한 요구 사항은 다음과 같습니다.

- 백업 작업을 수행하려면 주 서버 인스턴스의 SQL Server 서비스 계정과 백업 작업의 프록시 계정(기본적으로 주 서버 인스턴스의 SQL Server 에이전트 계정)에 백업 디렉터리에 대한 읽기/쓰기 권한이 있어야 합니다.
- 복사 작업을 수행하려면 복사 작업의 프록시 계정(기본적으로 보조 서버 인스턴스의 SQL Server 에이전트 계정)에 백업 디렉터리에 대한 읽기 권한과 복사 디렉터리에 대한 쓰기 권한이 있어야 합니다.
- 복원 작업을 수행하려면 보조 서버 인스턴스의 SQL Server 서비스 계정과 복원 작업의 프록시 계정(기본적으로 보조 서버 인스턴스의 SQL Server 에이전트 계정)에 복사 디렉터리에 대한 읽기/쓰기 권한이 있어야 합니다.

로그 전달 구성

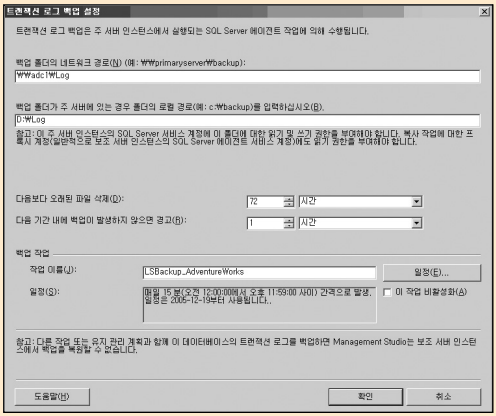
[따라하기] 로그 전달 구성하기

1. SQL Server Management Studio에서 로그 전달 구성의 주 데이터베이스로 사용할 데이터베이스를 마우스 오른쪽 단추로 클릭한 다음 속성을 클릭합니다.
2. [페이지 선택]에서 [트랜잭션 로그 전달]을 클릭합니다.



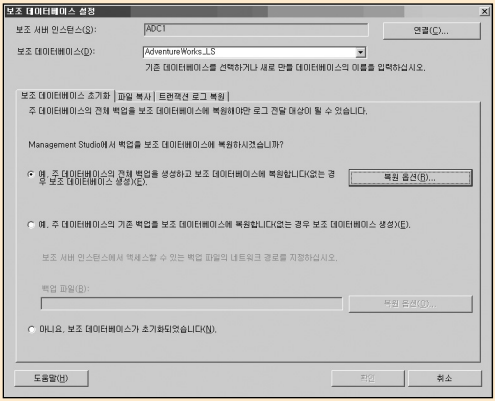
3. [이 데이터베이스를 로그 전달 구성의 주 데이터베이스로 사용] 확인란을 선택합니다.
4. [트랜잭션 로그 백업] 설정 부분에서 [백업 설정]을 클릭합니다.
5. [백업 폴더의 네트워크 경로]를 입력합니다. 백업 폴더가 주 서버에 있는 경우 [로컬 경로]도 입력합니다.
6. [다음보다 오래된 파일 삭제] 및 [다음 기간 내에 백업이 발생하지 않으면 경고] 부분에 적절한 값을 설정합니다.

7. 백업 [작업 이름] 및 [일정]을 설정 한 후 확인을 클릭합니다.



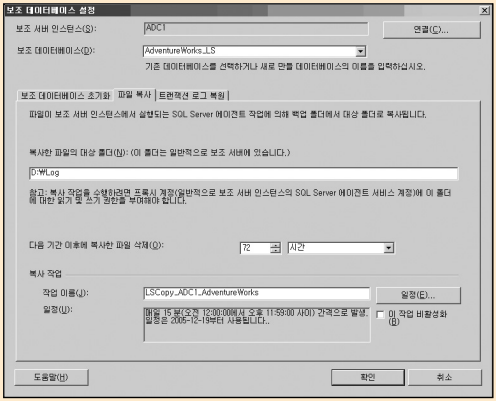
8. [보조 서버 인스턴스 및 데이터베이스]에서 추가를 클릭합니다.
9. [연결]을 클릭하여 보조 서버로 사용될 SQL Server 인스턴스를 설정합니다.
10. [보조 데이터베이스] 입력란의 목록에서 데이터베이스를 선택하거나 만들 데이터베이스의 이름을 입력합니다.

11. [보조 데이터베이스 초기화] 탭에서 보조 데이터베이스 초기화에 사용할 옵션을 선택합니다.



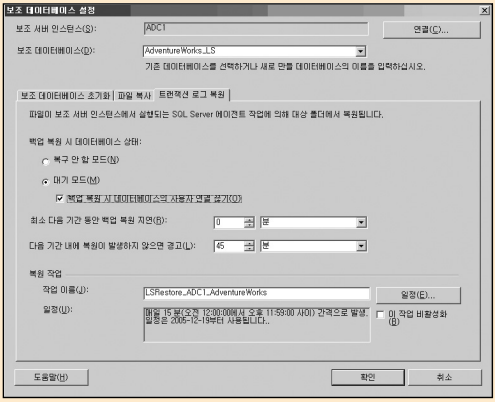
12. [파일 복사] 탭에서 보조 서버에서 로그 파일을 저장할 폴더의 위치인 [복사한 파일의 대상 폴더] 값을 지정합니다.

13. 복사 [작업 이름] 및 [일정]을 설정합니다.



14. [트랜잭션 로그 복원] 탭에서 백업 복원 시 [복구 안 함 모드] 혹은 [대기 모드]로 선택한 후, [백업 복원 지연] 시간, [경고] 시간을 설정합니다.

15. 복원 [작업 이름] 및 [일정]을 설정 한 후 [확인]을 클릭합니다.



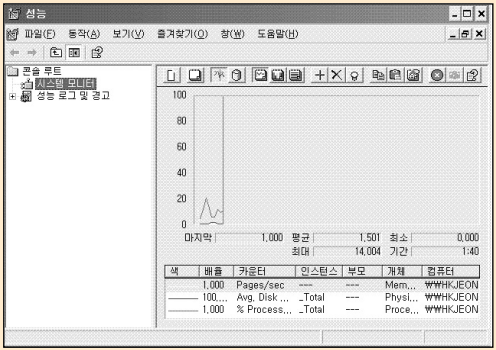
16. 모니터 서버 인스턴스를 사용할 경우 [모니터 서버 인스턴스 사용] 확인란을 선택하고 [설정]을 클릭합니다.
17. 모니터 연결에서 백업, 복사 및 복원 작업을 모니터 서버에 연결하는 데 사용할 [연결 방법]을 선택합니다.
18. 기록 보존에서 기록을 보존할 시간을 설정하고, 경고 [작업 이름] 및 [일정]을 설정한 후 확인을 클릭합니다.
19. 데이터베이스 속성 대화상자에서 [확인]을 클릭하여 구성 프로세스를 설정을 종료합니다.

성능 카운터 모니터링

성능 모니터 설정 방법

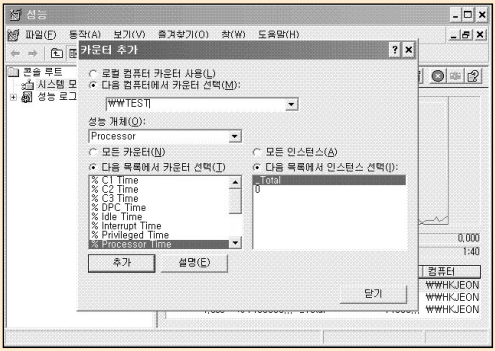
[따라하기] 성능 모니터 설정하기

1. [시작] --〉 [설정] --〉 [제어판] --〉 [관리도구] --〉 [성능]을 선택합니다.



2. 상단 그래픽 메뉴에서 [+]를 클릭하여 카운터 추가화면이 나타나면, 원하는 모니터 카운터를 추가합니다.
3. [로컬 컴퓨터 카운터 사용] 또는 [다른 컴퓨터에서 카운터 선택]을 선택합니다.
4. [성능개체]를 선택합니다.
5. [모든 카운터]를 선택하거나, [다음 목록에서 카운터 선택]를 선택한 후, 원하는 카운터를 선택합니다.
6. [모든 인스턴스]를 선택하거나, [다음 목록에서 인스턴스 선택]을 선택한 후, 원하는 인스턴스를 선택합니다.

7. [추가]를 클릭합니다.
8. 카운터 추가를 완료 한 후, [닫기]를 클릭합니다.

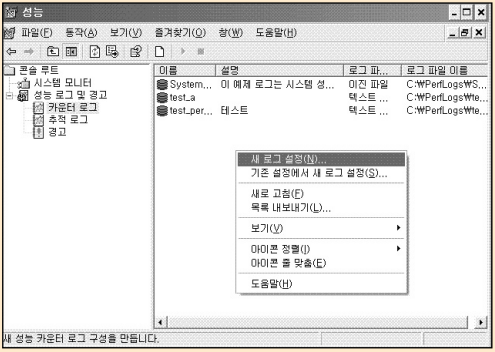


9. 선택한 카운터가 하단에 나타납니다.
10. 카운터를 제거할 경우에는, 제거하기를 원하는 카운터를 선택한 후, 상단 그래픽 메뉴에서, [X] (삭제키)를 클릭합니다.

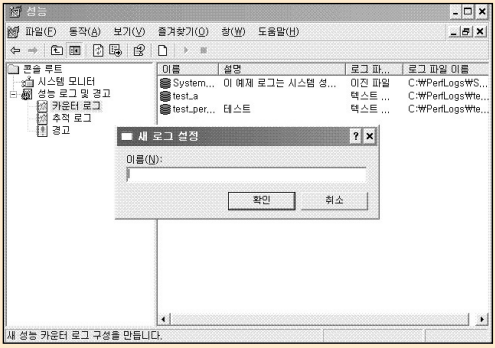
성능 로그 생성

[따라하기] 성능 로그 설정하기

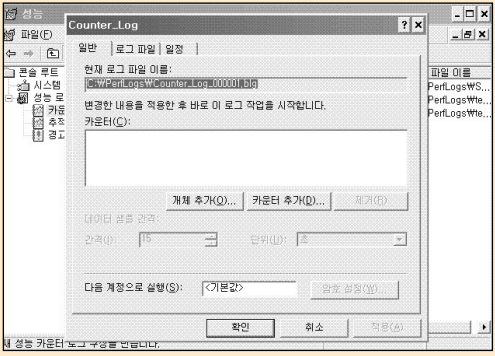
1. [시작] --〉 [설정] --〉 [제어판] --〉 [관리] --〉 [성능]을 선택합니다.
2. [성능 로그 및 경고]의 더하기 기호(+)를 클릭합니다.
3. 카운터 로그를 선택합니다.
4. 오른쪽 창에 마우스를 대고 오른쪽 버튼을 클릭하여, [새 로그 설정]을 선택합니다.



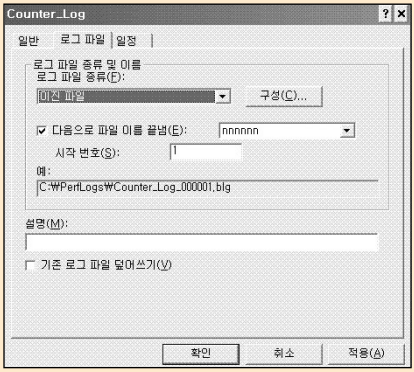
5. [새 로그 설정]에 원하는 로그 이름을 입력하고, [확인]을 클릭합니다.



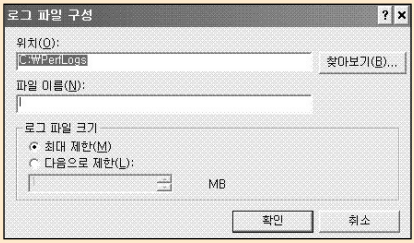
6. Counter_Log화면에 사용 정보와 파일 포맷을 설정합니다.



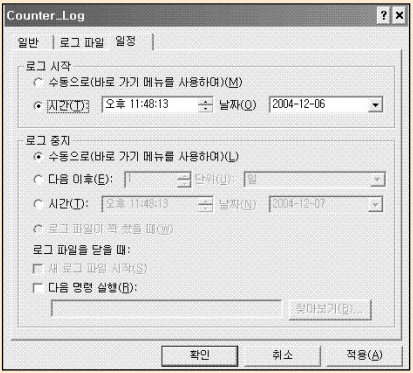
7. [개체 추가]를 선택하여, 원하는 개체를 추가합니다.
8. [카운터 추가]를 선택하여, 원하는 카운터를 추가합니다.
9. [데이터 샘플 간격]의 [간격]과 [단위]를 선택합니다.
10. [로그 파일]탭을 선택합니다.
11. [로그 파일 종류]를 선택합니다.
12. 구성을 클릭하면, 로그 파일 구성 화면이 나타납니다.



13. 구성을 클릭하면, 로그 파일 구성 화면이 나타납니다.



14. [찾아보기]를 클릭하여 로그 파일을 저장할 위치를 선택합니다.
15. 파일이름을 입력합니다.
16. 로그 파일 크기를 선택합니다. [다음으로 제한]할 경우, 제한 파일 크기를 입력합니다.
17. [확인]을 클릭합니다.
18. 파일명의 마지막 부분을 어떻게 설정할 것인지를 선택합니다.
19. 파일명의 마지막 부분을 일련 번호로 할 경우, 시작번호를 설정합니다.
20. 로그 파일의 설명을 입력합니다.
21. [일정]탭을 선택합니다.



22. 로그 시작 시간을 설정합니다.
23. 로그 중지 시간을 설정합니다.
24. 로그 파일을 닫을 때 실행할 명령이 있다면 선택합니다.
25. [확인]을 클릭합니다.
26. 오른쪽 창에 추가한 로그 파일의 목록이 나타납니다.
27. 새로 추가한 성능 로그의 이름 위에서 마우스의 오른쪽 버튼을 클릭한 후 [다른 이름으로 설정 저장]으로 설정을 저장해 놓으면, 설정 파일을 재사용할 수 있습니다.

[참고]

- *성능 카운터에서 서버의 많은 정보를 얻을 수 있습니다. 문제를 확인할 수 있도록, 충분한 시간 동안 필요한 카운터를 수집합니다.*

- *로그 파일 종류를 csv로 선택하여 수집하면, 분석 또는 집계하기 편리합니다.*

- *수집 시간을 고려하여 데이터 샘플 간격을 설정합니다. 샘플 간격이 커질수록 그래프의 정확도는 떨어지고, 샘플 간격이 작으면 데이터의 크기가 커집니다. 정해진 규칙은 없으며 저자의 경우에는 일반적으로 수집 시간에 따라 다음과 같이 샘플 간격을 설정합니다.*

수집 시간	샘플 간격
2시간	4 초
1일	30 초
5일	180 초
1일	15초

- *SQL Server를 모니터링하기 위하여 어떤 서버를 사용할지 결정합니다. 원격으로 모니터링할 수 있으나 장기간 동안 네트워크를 연결하여 카운터를 사용하는 것은 네트워크 트래픽을 가중시킵니다. 만일 SQL Server에 성능 모니터링 로그를 위한 공간이 있다면, 성능 로그 정보를 로컬로 기록합니다.*

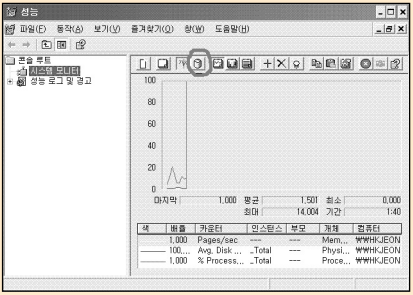
- *수집 파일 크기는 적정한 값으로 제한합니다. 수집 파일이 너무 커지면, 파일이 열리지 않는 경우가 있습니다.*

- *기존의 설정 파일(HTM)이 있는 경우에는 [기존 설정에서 새 로그 설정]을 사용하면 쉽게 구성이 가능합니다.*

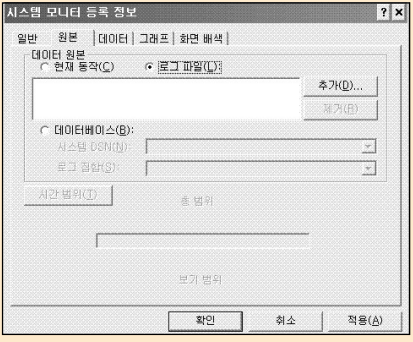
성능 로그 재생

[따라하기] 성능 로그 재생하기

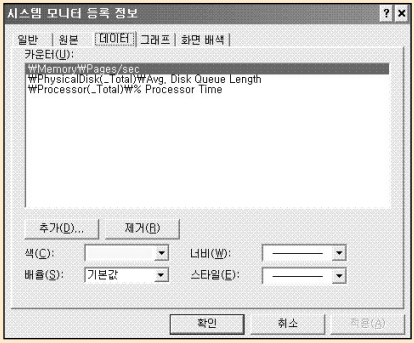
1. [시작] → [설정] → [제어판] → [관리] → [성능]을 선택합니다.



2. [로그 데이터 보기] 버튼을 클릭하면, [시스템 모니터 등록 정보] 창이 나타납니다.



3. 로그 파일을 선택하고, 추가 버튼을 클릭하여, 원하는 파일을 추가합니다.
4. [시간 범위]를 클릭하여, 원하는 시간대를 조절합니다.
5. [데이터] 탭을 클릭합니다.



6. [추가]를 클릭하여, 원하는 개체를 추가합니다.
7. [확인]을 클릭합니다.

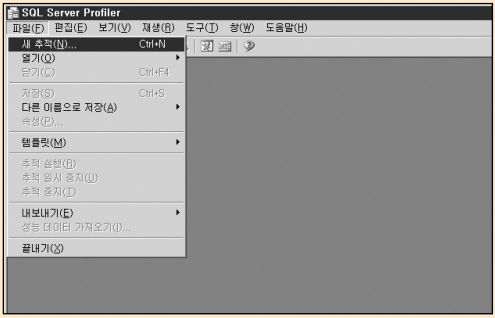
SQL Server 프로파일러

성능 문제의 디버깅은 문제의 원인을 알아내는 것으로 시작합니다. 많은 경우, 성능 문제는 비효율적인 SQL 문에서 기인합니다. 비효율적인 SQL 문이 문제의 원인이라고 의심될 때, SQL Server 프로파일러를 사용하면 문제의 원인이 되는 SQL 문을 쉽게 찾을 수 있기 때문에, 성능 튜닝에 유용합니다.

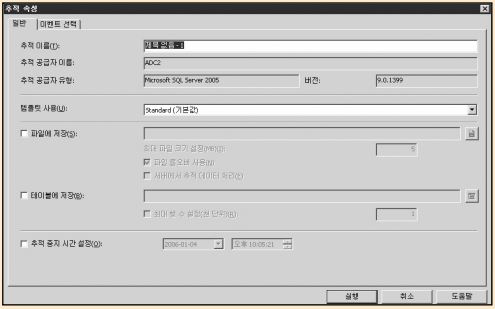
추적 수행하기 - GUI 사용

[따라하기] 추적 수집하기

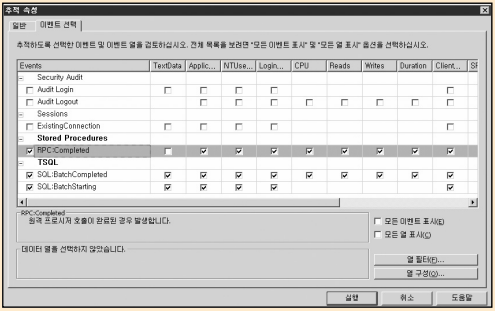
1. 다음 방법 중 하나를 이용하여 SQL Server 프로파일러를 실행합니다.
   [시작]→[프로그램]→[Microsoft SQL Server 2005]→[성능 도구]→[SQL Server Profiler]
   또는 SQL Server Management Studio의 상단 메뉴에서 [도구]→[SQL Server Profiler]
   를 선택합니다.
2. [파일]→[새 추적]을 선택합니다.



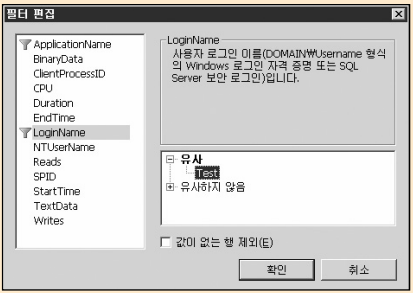
3. 원하는 SQL 서버에 연결하면, [추적 속성]창이 나타납니다.



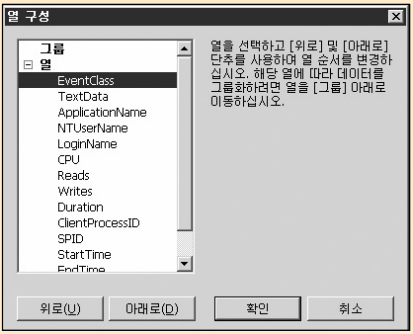
4. 추적이름을 입력합니다.
5. 템플릿을 사용할 경우에 템플릿을 선택합니다.
6. 파일에 저장하려면, [파일에 저장]을 선택하고, 저장할 위치와 파일명을 입력합니다.
7. 최대 파일 크기를 설정합니다.
8. [이벤트] 탭을 선택한 후, 추적을 원하는 이벤트와 이벤트 열을 추가하거나, 제거합니다. 많은 이벤트를 선택하는 것은 시스템에 상당한 부하를 일으킬 수 있으므로, 추적을 원하는 이벤트만 선택하시기를 권고합니다.



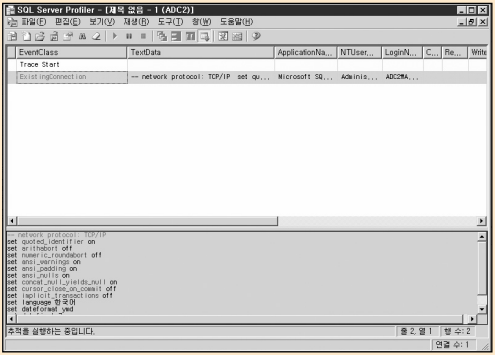
9. 필터를 이용하고 싶다면, [열 필터] 버튼을 클릭하여, 원하는 필터를 정의합니다. 예를 들어, LoginName이 Test인 것만 수집하고 싶다면 다음과 같이 설정합니다.



10. [열 구성] 버튼을 클릭한 후, 데이터 열의 순서를 조정합니다.



11. [실행]을 클릭하면, 수집이 시작됩니다.



12. 추적 정보 수집을 중지 하려면, 중지 버튼(적색 사각형 표시)을 클릭합니다.

추적 수행하기 - SP 사용

[파일] → [추적 스크립팅]을 이용하면, 원하는 확장 프로시저를 생성할 수 있습니다.
[추적 스크립팅]을 이용하여 작성한 저장 프로시저를 첨부합니다.

추적을 수집하는 목적에 따라 적절한 추적 이벤트와 이벤트 열의 설정이 필요합니다.

- 오래 실행되는 SQL 문 찾기
  오래 실행되는 쿼리는 잘못 튜닝된 시스템, 잘못 작성된 응용 프로그램, 또는 단순히 많은 동작을 수행하는 작업 등을 의미할 수 있습니다. 어떠한 경우건, 이러한 오래 실행되는 SQL 문을 찾아서 튜닝하는 것은 그 작업의 성능은 물론 전반적인 시스템 성능까지 향상 시킬 수 있습니다.
  권장되는 추적 이벤트 : **TSQL, SQL:BatchCompleted**
  정렬 기준 이벤트 열 : **Duration**

- 과도한 자원 사용자 찾기
  과도한 자원을 사용하는 응용 프로그램이나 사용자를 찾는 추적은 DBA에게 유용한 도구가 될 수 있습니다. 이러한 추적 유형은 CPU와 I/O 자원 모두를 많이 사용하는 SQL 문을 살펴야 합니다. 프로세스나 사용자를 식별하여, 응용 프로그램을 튜닝할 수 있습니다.
  권장되는 추적 이벤트 : **TSQL, SQL:BatchCompleted**
  정렬 기준 이벤트 열 : **CPU, Reads, 및 Writes**

- 교착 상태 알아내기
  사용자의 작업에 따라 교착상태는 시스템에서 문제가 될 수도 있고 또 그렇지 않을 수도 있습니다. 많은 경우에 있어 교착 상태는 심각한 문제일 수 있는데, 이 경우 원인을 알아내는 것은 성능을 향상시키는데 핵심이 됩니다. 그러나 이러한 이벤트를 수집 하는 것은 자원을 많이 사용하게 되므로 주의해야 합니다.
  - 권장되는 추적 이벤트
    TSQL, SQL:BatchStarting 동작하는 SQL 일괄 처리(batch)
    Locks, Lock:Deadlock 교착 상태 자체의 이벤트
    Locks, Lock:Deadlock Chain 교착 상태에 이르는 이벤트 순서

[참고]

- *불필요한 이벤트의 추가는 삼가합니다. 너무 많은 이벤트의 추가는 서버의 성능에 좋지 않은 영향을 줄 수 있습니다.*

- *일반적인 경우라면, 모든 이벤트 열을 포함시킵니다. 어떤 이벤트들은 보조적인 항목을 반환하는 어떤 항목들에 의존합니다. 적어도 다음 이벤트 열은 포함하는 것이 좋습니다.*

  - BinaryData
  - ClientProcessID
  - CPU
  - Duration
  - EndTime
  - EventClass
  - EventSubClass
  - HostName
  - IntegerData
  - LoginName
  - NTUserName
  - Reads
  - SPID
  - StartTime
  - TextData
  - Writes

- *테이블에 추적을 직접 저장하는 것은 성능상 좋지 않습니다. 추적 파일을 생성한 후, fn_trace_gettable 함수를 사용하면 테이블에 저장할 수 있습니다.*

추적 관련 저장 프로시저 예제 스크립트

다음은 추적 스크립팅을 저장 프로시저화한 예제 스크립트입니다. 시스템의 환경에 적합하도록 수정 보완하여 활용하기 바랍니다.

■ 추적을 시작하는 저장 프로시저의 예제 스크립트 : **sp_trace_start**

```
USE master
GO
CREATE PROCEDURE sp_trace_start @TraceFileName sysname=NULL,
@TraceName sysname='trace',
@Options int=2, -- TRACE_FILE_ROLLOVER
@MaxFileSize bigint=5,
@StopTime datetime=NULL,
@Events varchar(300)=
'10,12',
--  10 - RPC:Completed
--  12 - SQL:BatchCompleted
@Cols varchar(300)=
'1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
,31,32,33,34,35,36,37,38,39,40,41,42,43,44,'
-- 모든 이벤트 열
@IncludeFilter sysname=NULL,
@ExcludeFilter sysname=NULL
AS
SET NOCOUNT ON
-- 변수 선언
DECLARE @TraceId int
DECLARE @On bit
DECLARE @Rc int
```

```
SET @On=1

-- 이벤트와 이벤트 열을 확인한다.
IF @Events IS NULL or @Cols IS NULL BEGIN
   PRINT 'No Events or Coloumns.'
   RETURN -1
END

-- 파일경로와 파일명을 설정한다.
IF @TraceFileName IS NULL
SELECT @TraceFileName ='C:\Trace\Trace' + CONVERT(CHAR(8),getdate( ),112)

-- 추적 큐를 만든다
EXEC @Rc =sp_trace_create @TraceId OUT, @Options, @TraceFileName,
@MaxFileSize, @StopTime
IF @Rc <> 0 BEGIN
   PRINT 'Trace not started.'
   RETURN @Rc
END
PRINT 'Trace started.'
PRINT 'The trace file name is '+@TraceFileName+'.'

-- 추적할 이벤트 클래스들과 이벤트 열들을 지정한다
DECLARE @i int, @j int, @Event int, @Col int, @Colstring varchar(300)

IF RIGHT(@Events,1) <> ','SET @Events=@Events+','
SET @i=CHARINDEX(',',@Events)
WHILE @i <> 0 BEGIN
   SET @Event=CAST(LEFT(@Events,@i-1) AS int)
```

```
 SET @Colstring=@Cols
   IF RIGHT(@Colstring,1) <> ',' SET @Colstring=@Colstring+','
   SET @j=CHARINDEX(',',@Colstring)
   WHILE @j <> 0 BEGIN
      SET @Col=CAST(LEFT(@Colstring,@j-1) AS int)
      EXEC sp_trace_setevent @TraceId, @Event, @Col, @On
      SET @Colstring=SUBSTRING(@Colstring,@j+1 ,300)
      SET @j=CHARINDEX(',',@Colstring)
   END
   SET @Events=SUBSTRING(@Events,@i+1,300)
   SET @i=CHARINDEX(',',@Events)
END

-- 필터를 설정한다
EXEC sp_trace_setfilter @TraceId, 10, 0, 7, N'SQL Profiler'
EXEC sp_trace_setfilter @TraceId, 1, 0, 7,  N'EXEC% sp_%trace%'

IF @IncludeFilter IS NOT NULL
   EXEC sp_trace_setfilter @TraceId, 1, 0, 6, @IncludeFilter

IF @ExcludeFilter IS NOT NULL
   EXEC sp_trace_setfilter @TraceId, 1, 0, 7, @ExcludeFilter

-- 추적을 활성화한다
EXEC sp_trace_setstatus @TraceId, 1

-- 추적을 기록한다. (테이블 사용)
IF OBJECT_ID('tempdb..TraceQueueList') IS NULL BEGIN
   CREATE TABLE tempdb..TraceQueueList (TraceID int, TraceName varchar(20),
```

```
TraceFile sysname)
END

IF EXISTS(SELECT * FROM tempdb..TraceQueueList WHERE TraceName =
@TraceName) BEGIN
   UPDATE tempdb..TraceQueueList
   SET TraceID = @TraceId, TraceFile = @TraceFileName
   WHERE TraceName = @TraceName
END
ELSE BEGIN
   INSERT tempdb..TraceQueueList
      VALUES(@TraceId, @TraceName, @TraceFileName)
END

RETURN 0
GO

/* 실행하기 */
EXEC sp_trace_Start
GO
```

[참고]

- *output 파일을 지정하지 않으면, "C:\Trace" 폴더 밑에 추적 파일이 생성됩니다.*
- *이 스크립트를 직접 실행하려면, "C:\Trace" 폴더를 미리 생성해야 합니다.*
- *추적 파일이 커질 수 있으므로, output 파일이 생성되는 곳의 공간을 충분히 확보합니다.*
- *원하는 이벤트와 이벤트 열은 번호로 설정합니다. 이벤트와 이벤트 열 번호는 온라인 설명서를 참조하십시오.*

■ 추적을 중지하는 저장 프로시저의 예제 스크립트 : **sp_trace_stop**

```
USE master
GO
CREATE PROCEDURE sp_trace_stop @TraceName sysname='trace'
AS
SET NOCOUNT ON

-- 변수를 선언한다
DECLARE @TraceId int
DECLARE @TraceFileName sysname

-- 추적 목록을 확인하여, 추적을 중지합니다
IF OBJECT_ID('tempdb..TraceQueueList') IS NOT NULL BEGIN
   SELECT @TraceId = TraceID, @TraceFileName=TraceFile
         FROM tempdb..TraceQueueList
   WHERE TraceName = @TraceName

   IF @@ROWCOUNT <> 0 BEGIN
      EXEC sp_trace_setstatus @TraceId, 0
      EXEC sp_trace_setstatus @TraceId, 2
      DELETE tempdb..TraceQueueList
         WHERE TraceName = @TraceName
      PRINT 'Trace is stopped.'
         + 'The trace output file name is'+@TraceFileName
   END
   ELSE
      PRINT 'No active traces.'
END
ELSE
```

```
    PRINT 'No active traces.'

RETURN 0
GO

/* 실행하기 */
EXEC sp_trace_stop
```

[참고]
sp_trace_stop은 sp_trace_start로 실행한 추적(Trace)를 중지하는 저장 프로시저입니다.

추적 파일을 테이블로 복사하기

fn_trace_gettable 함수를 사용하면 추적 파일을 SQL Server 프로파일러에서 로드 가능한 테이블로 복사할 수 있습니다. 추적 파일을 테이블로 복사하면 T-SQL을 활용하여 다양한 분석이 가능하므로 편리합니다.

[따라하기] 추적 파일을 테이블로 복사하기

```
SELECT IDENTITY(int, 1, 1) AS SeqNo, * INTO temp_trc
FROM fn_trace_gettable('c:\temp\my_trace.trc', default);
GO
```

동적 관리 뷰 및 함수 활용

동적 관리 뷰 및 함수는 서버 인스턴스의 상태를 모니터링하거나 문제 진단 및 성능 튜닝에 사용할 수 있는 서버 상태 정보를 제공합니다. 동적 관리 뷰 및 함수가 성능 튜닝에 있어서 유용한 정보를 제공하지만, 과다한 사용은 성능에 좋지 않은 영향을 미칠 수 있으므로 유의합니다.

모든 동적 관리 뷰 및 함수는 sys 스키마에 존재하며, dm_* 형태로 존재합니다.
동적 관리 뷰 및 함수에는 다음과 같은 두 유형이 있습니다. 동적 관리 뷰 및 함수 참조는 성능에 영향을 미칠 수 있으므로 제한된 사용자에게만 허용하는 것이 좋습니다.

- 서버 범위 동적 관리 뷰 및 함수. 이 유형에는 서버에 대한 VIEW SERVER STATE 권한이 필요합니다.
- 데이터베이스 범위 동적 관리 뷰 및 함수. 이 유형에는 데이터베이스에 대한 VIEW DATABASE STATE 권한이 필요합니다.

대기 정보 확인

실행 중인 스레드로 인해 발생한 대기에 대한 정보를 확인하려면 sys.dm_os_wait_stats 동적 관리 뷰를 사용합니다. 이 뷰는 SQL Server 와 관련된 성능 문제뿐 아니라 특정 쿼리 및 일괄 처리와 관련된 성능 문제의 진단에도 도움이 됩니다. 각 대기 유형별 대기 시간을 확인하여, 가장 대기가 많은 유형을 확인하고, 작업 부하에 따른 대기 유형의 변화도 확인합니다.

[따라하기] 시스템의 대기 정보 확인하기

```
SELECT * FROM sys.dm_os_wait_stats
ORDER BY wait_time_ms DESC;
GO
```

[따라하기] 래치 대기 정보 확인하기

```
SELECT * FROM sys.dm_os_latch_stats;
GO
```

[따라하기] 대기 통계, 래치 대기 통계 재설정하기

```
DBCC SQLPERF('sys.dm_os_wait_stats', CLEAR);
GO
DBCC SQLPERF('sys.dm_os_latch_stats', CLEAR);
GO
```

[따라하기] I/O 대기 확인하기

```
SELECT database_id
        , file_id
        , io_stall,io_pending_ms_ticks
        , scheduler_address
FROM sys.dm_io_virtual_file_stats (NULL, NULL) AS t1,
    sys.dm_io_pending_io_requests AS t2
WHERE t1.file_handle = t2.io_handle;
GO

SELECT * FROM sys.dm_io_virtual_file_stats (NULL, NULL);
GO
SELECT * FROM sys.dm_io_pending_io_requests;
GO
```

[따라하기] I/O 할당 대기 확인하기

```
-- Tempdb (DBID=2)
SELECT session_id, wait_duration_ms, resource_description
FROM sys.dm_os_waiting_tasks
WHERE wait_type LIKE 'PAGE%LATCH_%'
AND resource_description LIKE '2:%';
GO

-- 사용자 DB (다음에서 dbid 부분을 db_id( ) 값을 입력하여 실행하면 됨 )
SELECT session_id, wait_duration_ms, resource_description
FROM sys.dm_os_waiting_tasks
WHERE wait_type LIKE 'PAGE%LATCH_%'
AND resource_description LIKE '5:%';
GO
```

실행중인 프로세스 정보 확인

sys.dm_exec_requests 동적 관리 뷰를 이용하여 서버에서 실행되고 있는 모든 세션들의 정보를 확인할 수 있습니다. 사용자에게 서버에 대한 VIEW SERVER STATE 권한이 있으면 SQL Server 인스턴스에서 실행 중인 모든 세션을 볼 수 있고, 그렇지 않으면 현재 세션만 볼 수 있습니다.

[따라하기] 현재 실행중인 프로세스의 세션 정보, 쿼리문, CPU 시간 정보 확인하기

```
SELECT r.session_id
        ,status
        ,wait_type
        ,substring(qt.text,r.statement_start_offset/2,
```

```
                    (case when r.statement_end_offset = -1
                    then len(convert(nvarchar(max), qt.text)) * 2
                    else r.statement_end_offset end -
                    r.statement_start_offset)/2)
        as query_text   --- 현재 실행 중인 일괄 처리 또는 프로시저
        ,qt.dbid        ,qt.objectid
        ,r.cpu_time
        ,r.total_elapsed_time
        ,r.reads
        ,r.writes
        ,r.logical_reads
        ,r.scheduler_id
FROM sys.dm_exec_requests r
CROSS APPLY sys.dm_exec_sql_text(sql_handle) as qt
WHERE r.session_id > 50
ORDER BY r.scheduler_id, r.status, r.session_id;
GO
```

쿼리 실행 분석

sys.dm_exec_query_stats 동적 관리 뷰를 이용하여 캐싱되어 있는 쿼리 실행 계획에 대한 집계 성능 통계를 확인할 수 있습니다. 이 뷰에는 각 쿼리 계획에 대한 행이 포함되어 있으며, 캐시에서 실행 계획이 제거되면 이 뷰에서도 해당 행이 제거됩니다.

[따라하기] 실행 소요 시간이 가장 긴 상위 50개의 쿼리 정보 조회하기

```
SELECT TOP 50
        sum(qs.total_worker_time) as total_cpu_time,
        sum(qs.execution_count) as total_execution_count,
        count(*) as  '#_statements',
        qt.dbid, qt.objectid, qs.sql_handle,
        qt.[text]
FROM sys.dm_exec_query_stats as qs
CROSS APPLY sys.dm_exec_sql_text (qs.sql_handle) as qt
GROUP BY qt.dbid,qt.objectid, qs.sql_handle,qt.[text]
ORDER BY sum(qs.total_worker_time) DESC,qs.sql_handle;
GO
```

[따라하기] CPU Time 순으로 상위 50개 쿼리 정보 조회하기

```
SELECT TOP 50
          qs.total_worker_time/qs.execution_count as [Avg CPU Time],
        substring (qt.text,qs.statement_start_offset/2,
        (case when qs.statement_end_offset = -1
        then len (convert (nvarchar(max), qt.text)) * 2
        else qs.statement_end_offset end -qs.statement_start_offset)/2)
        as query_text,
        qt.dbid,
        qt.objectid
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY [Avg CPU Time] DESC;
GO
```

[따라하기] 평균 I/O가 높은 상위 50개 쿼리 정보 조회하기

```
SELECT TOP 50
  (qs.total_logical_reads + qs.total_logical_writes) /qs.execution_count AS [Avg IO],
        SUBSTRING(qt.text,qs.statement_start_offset/2,
                  (case when qs.statement_end_offset = -1
                  then len(convert(nvarchar(max), qt.text)) * 2
                  else qs.statement_end_offset end -
qs.statement_start_offset)/2)
                  AS query_text,
                  qt.dbid, dbname=db_name(qt.dbid),
                  qt.objectid,
                  qs.sql_handle,
                  qs.plan_handle
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS qt
ORDER BY [Avg IO] DESC;
GO
```

[따라하기] CLR 내에서의 평균 사용 시간 확인하기

```
SELECT TOP 5 creation_time, last_execution_time, total_clr_time,
        total_clr_time/execution_count AS [Avg CLR Time], last_clr_time,
        execution_count, (SELECT SUBSTRING(text, statement_start_offset/2,
                (CASE WHEN statement_end_offset = -1 THEN
LEN(CONVERT(nvarchar(max),
text)) * 2
        ELSE statement_end_offset END - statement_start_offset)/2)
FROM sys.dm_exec_sql_text(sql_handle)) AS query_text
```

```
FROM sys.dm_exec_query_stats
ORDER BY [Avg CLR Time] DESC;
GO
```

[따라하기] 병렬로 실행 중인 프로세스 확인하기

```
SELECT r.session_id,
        r.request_id,
        max(isnull(exec_context_id, 0)) AS number_of_workers,
        r.sql_handle,
        r.statement_start_offset,
        r.statement_end_offset,
        r.plan_handle
FROM sys.dm_exec_requests r
        join sys.dm_os_tasks t ON r.session_id = t.session_id
        join sys.dm_exec_sessions s ON r.session_id = s.session_id
WHERE s.is_user_process = 0x1
GROUP BY r.session_id, r.request_id, r.sql_handle, r.plan_handle,
        r.statement_start_offset, r.statement_end_offset
HAVING max(isnull(exec_context_id, 0)) > 0;
GO
```

[따라하기] 재컴파일 확인하기

```
SELECT TOP 25
        sql_text.text,
        sql_handle,
        plan_generation_num,
        execution_count,
        dbid,
        objectid
FROM sys.dm_exec_query_stats a
        CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS sql_text
WHERE plan_generation_num > 1
ORDER BY plan_generation_num DESC;
GO
```

tempdb 점검

[따라하기] tempdb 공간 사용 현황 확인하기

```
SELECT
        SUM (user_object_reserved_page_count)*8 AS user_objects_kb,
        SUM (internal_object_reserved_page_count)*8 AS internal_objects_kb,
        SUM (version_store_reserved_page_count)*8 AS version_store_kb,
        SUM (unallocated_extent_page_count)*8 AS freespace_kb
FROM sys.dm_db_file_space_usage
WHERE database_id = 2;
GO
```

블로킹(차단) 점검

[따라하기] 블로킹(차단) 확인하기

master 데이터베이스에 생성해 놓고 사용하면 편리합니다.

```
create proc dbo.sp_block (@spid bigint=NULL)
as
SELECT t1.resource_type
        ,'database'=db_name(resource_database_id)
        ,'blk object'=isnull(object_name(t1.resource_associated_entity_id)
,t1.resource_associated_entity_id)
        , t1.request_mode
        , t1.request_session_id  -- spid
        , t2.blocking_session_id  -- spid
FROM sys.dm_tran_locks as t1,
        sys.dm_os_waiting_tasks as t2
WHERE t1.lock_owner_address = t2.resource_address
and t1.request_session_id = isnull(@spid,t1.request_session_id)
GO
```

SQLdiag 유틸리티

Microsoft SQL Server 2005에 포함된 SQLdiag 유틸리티는 콘솔 응용 프로그램 또는 서비스로 실행할 수 있는 범용 진단 정보 수집 유틸리티입니다. SQLdiag는 SQL Server를 설치하면 "Microsoft SQL Server\90\Tools\Binn" 폴더아래에 있습니다.

SQLdiag를 사용하여 SQL Server 및 기타 서버 유형에서 로그 및 데이터 파일을 수집할 수 있으며 이러한 파일을 사용하여 지속적으로 서버를 모니터링하거나 특정 서버 문제를 해결하는데 활용할 수 있습니다.

SQL Server 2005의 SQLdiag 유틸리티는 SQL Server 2000에 비해 많이 변경된 관계로, 커맨드라인 인수가 SQL Server 2000과 호환되지 않습니다.

SQLdiag 유틸리티를 사용하면 다음과 같은 유형의 진단 정보를 수집할 수 있습니다.

- Windows 성능 로그 (Windows performance logs)
- Windows 이벤트 로그 (Windows event logs)
- SQL Server 프로파일러 추적 (SQL Server Profiler traces)
- SQL Server 블로킹 정보 (SQL Server blocking information)
- SQL Server 구성 정보 (SQL Server configuration information)

기본적으로 제공되는 구성 파일인 SQLdiag.XML, SD_General.XML, SD_Detailed.XML 파일을 편집하면 위의 진단 정보들 중에서 원하는 정보만 수집할 수 있습니다.

다음은 SQLdiag 유틸리티 구문 및 실행 매개변수에 대한 설명입니다.

[구문] SQLdiag
{ [/? 도움말] }
|
{ [/I 구성 파일(configuration_file)]
[/O 출력 폴더 경로(output_folder_path)]
[/P 지원 폴더 경로(support_folder_path)]
[/N 출력 폴더 관리 옵션(output_folder_management_option)]
[/C 파일 압축 유형(file_compression_type)]
[/B [+]시작 시간(start_time)]
[/E [+]종료 시간(stop_time)]
[/A SQLdiag_application_name]
[/Q 자동(Quiet) 모드로 실행]
[/G 일반(Generic) 모드로 실행]
[/R 서비스 등록]
[/U 서비스 삭제]
[/L 연속 모드로 실행]
[/X 스냅숏 모드로 실행] }
|
{ [START | STOP | STOP_ABORT] }
|
{ [START | STOP | STOP_ABORT] /A SQLdiag_application_name }

옵션

옵션	내용
/?	사용 방법을 알려줍니다.
/I 구성 파일	SQLdiag에서 사용할 구성 파일을 설정합니다. /I는 기본적으로 SQLdiag.XML 파일로 정의되어 있습니다.
/O 출력 폴더 경로	SQLdiag 출력을 지정된 폴더에 기록합니다. /O 옵션을 지정하지 않으면 SQLdiag 출력이 SQLdiag 시작 폴더의 SQLDIAG라는 하위 폴더에 기록됩니다. SQLDIAG 폴더가 없으면 SQLdiag에서 이 폴더를 생성합니다. **[참고] 출력 폴더 위치는 /P로 지정할 수 있는 지원 폴더 위치에 대해 상대적입니다. 완전히 다른 출력 폴더 위치를 설정하려면 /O 옵션에 전체 디렉터리 경로를 지정합니다.**
/P 지원 폴더 경로	지원 폴더 경로를 설정합니다. 기본적으로 /P는 SQLdiag 실행 파일이 있는 폴더로 설정됩니다. 지원 폴더에는 XML 구성 파일, Transact-SQL 스크립트 및 진단 정보를 수집하는 동안 유틸리티에서 사용하는 기타 파일을 비롯한 SQLdiag 지원 파일이 있습니다. 이 옵션을 사용하여 대체 지원 파일 경로를 지정하면 SQLdiag는 지정한 폴더에 없는 경우 필요한 지원 파일을 자동으로 복사합니다. **[참고] 현재 폴더를 지원 경로로 설정하려면 다음과 같이 명령줄에서 %cd%를 지정합니다.** **SQLDIAG /P %cd%**

/N 출력 폴더 옵션	SQLdiag가 시작할 때, 출력 폴더를 덮어쓸 것인지 아니면 다른 이름으로 저장할 것인지를 지정하는 옵션입니다. 사용 가능한 옵션은 다음과 같습니다. 1 = 출력 폴더를 덮어씁니다. (기본값) 2 = SQLdiag가 시작할 때, 기존의 출력 폴더를 SQLdiag_00001, SQLdiag_00002와 같이 다른 이름으로 저장하게 됩니다. 현재의 출력 폴더를 다른 이름으로 저장한 후, SQLdiag는 기본 출력 폴더인 SQLdiag 폴더에 결과를 작성합니다. **[참고]** SQLdiag는 시작할 때 현재 출력 폴더에 출력을 추가하지 않습니다. 대신 기본 출력 폴더를 덮어쓰거나(옵션 1) 폴더의 이름을 바꾼 다음(옵션 2) SQLDIAG라는 새 기본 출력 폴더에 출력을 씁니다.
/C 파일 압축 유형	SQLdiag 출력 폴더 파일에 적용되는 파일 압축 유형을 지정합니다. 사용 가능한 옵션은 다음과 같습니다. 0 = 없음 (기본값) 1 = NTFS 압축 사용
/B [+]시작 시간	진단 데이터 수집 시작 날짜와 시간을 YYYYMMDD_HH:MM:SS 형식으로 지정합니다. 시간은 24시간제 표시법으로 지정합니다. 예를 들어 오후 2시는 14:00:00으로 지정해야 합니다. 날짜 없이 HH:MM:SS에 +를 사용하여 현재 날짜와 시간에 대한 상대 시간을 지정할 수 있습니다. 예를 들어 /B +02:00:00와 같이 지정하면 SQLdiag에서 2시간 후부터 정보 수집을 시작합니다. 이 때 +와 지정 시간 사이에 공백이 없어야 합니다. 지정한 시작 시간이 과거일 경우 SQLdiag에서 강제로 시작 일자를 변경하여 시작 일자와 시간을 미래 시간으로 설정합니다. 예를 들어 현재 시간이 08:00:00인데 /B 01:00:00이라고 지정하면 SQLdiag에서 임의로 시작 일자를 다음 날로 변경합니다. SQLdiag는 유틸리티를 실행하는 컴퓨터의 현지 시간을 사용합니다.

/E [+]종료 시간	진단 데이터 수집 종료 날짜와 시간을 YYYYMMDD_HH:MM:SS 형식으로 지정합니다. 시간은 24시간제 표시법으로 지정합니다. 예를 들어 오후 2시는 14:00:00으로 지정해야 합니다. 날짜 없이 HH:MM:SS에 +를 사용하여 현재 날짜와 시간에 대한 상대 시간을 지정할 수 있습니다. 예를 들어 시작 시간과 종료 시간을 /B +02:00:00 /E +03:00:00으로 지정하면 SQLdiag에서는 2시간 후부터 정보 수집을 시작하고 3시간 동안 정보를 수집한 다음 중지합니다. /B를 지정하지 않으면 SQLdiag에서 즉시 진단 정보 수집을 시작하고 /E로 지정된 날짜와 시간에 종료합니다. +와 지정한 시작 시간 또는 종료 시간 사이에 공백이 없어야 합니다. SQLdiag는 유틸리티를 실행하는 컴퓨터의 현지 시간을 사용합니다.
/A SQLdiag_application_name	동일한 SQL Server 인스턴스에서 여러 개의 SQLdiag 유틸리티 인스턴스가 실행될 수 있도록 하는 옵션입니다. 각 SQLdiag_application_name으로 서로 다른 **SQLdiag** 인스턴스를 식별합니다. SQLdiag_application_name 인스턴스와 SQL Server 인스턴스 이름은 전혀 관계가 없습니다. SQLdiag_application_name을 사용하여 특정 **SQLdiag** 서비스 인스턴스를 시작하거나 중지할 수 있습니다. SQLdiag START /A SQLdiag_application_name SQLdiag인스턴스를 서비스로 등록하려면 다음의 예와 같이 /R 옵션을 사용하여 실행하면 됩니다. SQLdiag /R /A SQLdiag_application_name **[참고]** SQLdiag는 SQLdiag_application_name에 대해 지정한 인스턴스 이름 앞에 DIAG$를 자동으로 붙입니다. SQLdiag를 서비스로 등록하는 경우 이를 통해 구분이 가능한 서비스 이름이 제공됩니다.
/Q	자동 모드에서 SQLdiag를 실행합니다. /Q는 암호 프롬프트와 같은 모든 프롬프트를 표시하지 않습니다.

/G	일반 모드에서 SQLdiag를 실행합니다. /G를 지정하면 시작 시 SQLdiag에서 SQL Server 연결을 확인하거나 사용자가 sysadmin 고정 서버 역할의 멤버인지 확인하지 않습니다. 대신 SQLdiag는 사용자에게 각각의 요청된 진단 정보를 수집할 수 있는 권한이 있는지 여부를 Windows를 통해 확인합니다. /G를 지정하지 않으면 SQLdiag에서 사용자가 Windows Administrators 그룹의 멤버인지 확인하고 Administrators 그룹 멤버가 아닐 경우 SQL Server 진단 정보를 수집하지 않습니다.
/R	SQLdiag를 서비스로 등록합니다. SQLdiag를 서비스로 등록할 때 지정하는 모든 명령줄 인수는 나중에 서비스를 실행할 때 사용할 수 있도록 보관됩니다. SQLdiag를 서비스로 등록할 경우 기본 서비스 이름은 SQLDIAG입니다. /A 인수를 사용하여 서비스 이름을 변경할 수 있습니다. 다음과 같이 START 명령줄 인수를 사용하여 서비스를 시작할 수 있습니다. SQLdiag START 다음과 같이 net start 명령을 사용하여 서비스를 시작할 수도 있습니다. net start SQLdiag **[참고]** /A 인수를 사용하여 SQLdiag_application_name을 지정하면 기본 구성 파일을 포함하여 유틸리티가 사용하는 많은 파일의 기본 이름이 변경됩니다. 예를 들어 SQLdiag_application_name이 MyApp인 경우 기본 구성 파일의 이름이 DIAG$MyApp.XML로 바뀝니다. SQLdiag를 서비스로 등록할 때 기본 구성 파일을 이 변경된 이름으로 복사하여 저장하거나 /I 인수를 사용하여 구성 파일로 지정해야 합니다.
/U	서비스로 등록된 SQLdiag의 등록을 취소합니다. 명명된 SQLdiag 인스턴스의 등록을 취소하는 경우에는 /A 인수도 사용합니다.

/L	각각 /B 또는 /E 인수를 사용하여 시작 시간이나 종료 시간도 지정한 경우 SQLdiag가 연속 모드로 실행됩니다. 예약된 종료로 인해 진단 정보 수집이 중지되면 SQLdiag가 자동으로 다시 시작됩니다. 예를 들어 /E 또는 /X 인수를 사용하면 종료됩니다. [참고] /B 및 /E 명령줄 인수를 사용하여 시작 시간이나 종료 시간을 지정하지 않는 경우 SQLdiag는 /L 인수를 무시합니다. /L은 서비스 모드를 나타내는 인수가 아닙니다. SQLdiag를 서비스로 실행할 때 /L을 사용하려면 해당 서비스를 등록할 때 커맨드라인에서 /L을 지정해야 합니다.
/X	스냅숏 모드에서 SQLdiag를 실행합니다. SQLdiag는 구성된 모든 진단 정보에 대해 스냅숏을 만들고 자동으로 종료됩니다.
START | STOP | STOP_ABORT	SQLdiag 서비스를 시작하거나 중지합니다. STOP_ABORT는 현재 수행하고 있는 진단 정보 수집을 완료하지 않고 가능한 빨리 서비스를 강제 종료합니다. 이러한 서비스 제어 인수를 사용할 때는 명령줄에서 첫 번째 인수로 사용해야 합니다. 예를 들면 다음과 같습니다. SQLdiag START 명명된 SQLdiag 인스턴스를 지정하는 /A 인수만 START, STOP 또는 STOP_ABORT와 함께 사용하여 특정 SQLdiag 서비스 인스턴스를 제어할 수 있습니다. 예를 들면 다음과 같습니다. SQLdiag START /ASQLdiag_application_name

SQLdiag.exe를 원하는 폴더에 복사한 후, 커맨드 창에서 SQLdiag.exe가 있는 곳으로 이동하여 SQLdiag.exe를 실행합니다.

[따라하기] SQLdiag를 자동으로 특정 시각에 시작하고 특정 시각에 종료하기

```
-- 2006년 1월 1일 오전 9시부터 오후 3시까지 진단 정보 수집을 실행합니다.
SQLdiag /B 20060101_09:00:00 /E 20060101_15:00:00
```

[따라하기] 구성 파일을 지정하고 출력 폴더를 지정하여 특정 시각에 SQLdiag실행하기

```
-- 구성 파일을 SD_General.XML로 지정하고 출력 폴더는 :E\Result_01로 지정하여
2006년 1월 1일 오후 2시부터 오후 3시까지 진단 정보 수집을 실행합니다.
SQLdiag /I SD_General.XML /O :E\Result_01 /B 20060101_14:00:00 /E
20060101_15:00:00
```

[따라하기] SQLdiag 실행시점부터 1시간 후에 정보 수집을 시작하고 3시간 후에 종료하기

```
SQLdiag /B +01:00:00 /E +03:00:00
```

[따라하기] SQLdiag가 자동으로 매일 03시부터 05시까지 진단 정보 수집 수행하기

```
SQLdiag /B 03:00:00 /E 05:00:00 /L
```

[따라하기] SQLdiag가 자동으로 매일 03시에 진단 정보의 스냅숏 수집하기

```
SQLdiag /B 03:00:00 /X /L
```

SQLdiag를 콘솔 응용 프로그램으로 실행하는 경우 SQLdiag가 실행 중인 콘솔 창에서 Ctrl+C를 눌러 중지할 수 있습니다. Ctrl+C를 누르면 SQLDiag 데이터 수집이 종료되며 프로세스가 종료될 때까지 몇 분 정도 기다려야 한다는 메시지가 콘솔 창에 표시됩니다. Ctrl+C를 두 번 눌러 모든 자식 진단 프로세스를 종료하고 즉시 응용 프로그램을 종료합니다.

주의

SQLdiag가 콘솔 응용 프로그램으로 실행 중인 경우 콘솔 창을 강제로 종료하거나 로그오프하지 않아야 합니다. 만약 SQL Server 프로파일러 추적 정보를 수집하고 있었다면 정상적으로 종료되지 않고 계속해서 추적 정보를 수집하게 되는 문제가 발생할 수 있습니다.

[따라하기] SQLdiag가 서비스로 실행 중일 때 종료하기

```
SQLDiag STOP
```

[따라하기] 인스턴스 이름(Instance1)을 지정하고 SQLdiag 인스턴스 종료하기

```
SQLdiag STOP /A Instance1
```

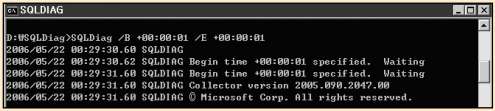
[따라하기] 실행 중인 진단 정보 수집 작업이 종료될 때까지 기다리지 않고 바로 종료 하기

```
SQLdiag STOP_ABORT
```

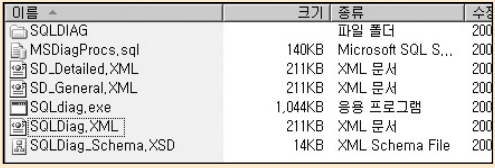
[따라하기] 구성 파일 수정하여 사용하기

구성 파일의 서버 이름이나 수집하는 이벤트들을 원하는 값으로 수정하여 필요할 때 재사용할 수 있습니다.

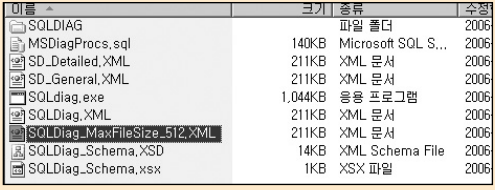
1. SQLdiag.exe를 원하는 폴더에 복사합니다.
2. 커맨드 창에서 SQLdiag.exe가 있는 곳으로 이동하여 SQLdiag를 실행합니다.



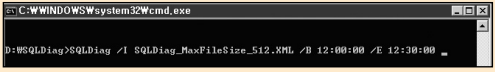
3. 실행이 종료된 후, SQLdiag.exe가 있는 폴더를 확인하면 구성 파일과 출력 폴더가 생성되었음을 확인합니다.



4. 변경을 원하는 구성 파일을 열어 원하는 값을 수정하여 다른 이름으로 저장합니다. 예를 들어 Machine name이나 수집되는 이벤트 종류 등을 수정합니다.



5. 원하는 옵션을 지정하여 SQLdiag를 실행합니다.



기타

주요 명령 프롬프트 유틸리티

다음은 SQL Server 2005에 포함되어 있는 주요 명령 프롬프트 유틸리티들입니다. 자세한 내용은 SQL Server 온라인 설명서를 참조 바랍니다.

도구이름	설명
Bcp	SQL Server 2005 인스턴스와 사용자가 지정한 형식의 데이터 파일 간에 데이터를 대량 복사하는데 사용됩니다. bcp 유틸리티를 사용하여 많은 수의 새로운 행을 SQL Server 테이블로 가져오거나 테이블에서 데이터 파일로 데이터를 내보낼 수 있습니다.
Dta	작업 부하를 분석하여 최적화 권고안을 제시하는데 사용됩니다. 명령 프롬프트에서 dta.exe를 사용하거나 GUI를 통해 데이터베이스 엔진 튜닝 관리자에 액세스할 수 있습니다. 커맨드라인 유틸리티를 사용하면 데이터베이스 엔진 튜닝 관리자 기능을 스크립트 및 소프트웨어 프로그램으로 통합할 수 있어서 편리합니다.
Dtexec	SSIS 패키지를 구성하고 실행하는데 사용됩니다.
Dtutil	SSIS 패키지를 관리하는데 사용됩니다.
Nscontrol	Notification Services를 관리하기 위한 명령 프롬프트 유틸리티입니다. Notification Services 인스턴스와 응용 프로그램을 배포, 구성, 모니터링 및 제어할 수 있는 명령을 제공합니다.
Osql	명령 프롬프트에서 Transact-SQL 문, 시스템 프로시저 및 스크립트 파일을 입력할 수 있습니다. 그러나, osql 유틸리티는 다음 버전에서 제거된다고 온라인 설명서에 기술되어 있으므로 osql 대신 sqlcmd 유틸리트를 사용하는 것을 권고합니다.

profiler90	명령 프롬프트에서 SQL Server 프로파일러 도구를 실행하는데 사용됩니다.
Rs	Reporting Services 스크립트를 실행하는데 사용됩니다.
Rsconfig	리포트 서버 연결을 구성하는데 사용됩니다.
rskeymgmt	리포트 서버에서 암호화 키를 구성하는데 사용됩니다.
Sac	SQL Server 2005 인스턴스들 간에 노출 영역 구성(Surface area configuration)을 가져오거나 내보내는데 사용됩니다. sac 유틸리티의 가장 쉬운 사용법은 SQL Server 노출 영역 구성 GUI(그래픽 사용자 인터페이스)를 사용하여 한 컴퓨터를 구성한 다음에 이 컴퓨터의 설정을 sac 유틸리티를 통해 파일로 내보내는 것입니다. 그러면 sac 유틸리티를 사용하여 SQL Server 2005 구성 요소에 대한 모든 설정을 로컬 컴퓨터나 원격 컴퓨터의 다른 SQL Server 2005 인스턴스에 손쉽게 적용할 수 있습니다.
Sqlagent90	명령 프롬프트에서 SQL Server 에이전트를 실행하는데 사용됩니다. 대개는 응용 프로그램에서 SQL-DMO 메서드를 사용하거나 SQL Server Management Studio를 통해 SQL Server 에이전트를 실행해야 하며, 문제가 발생하여 SQL Server 에이전트를 진단하거나 해결하고자 하는 경우에만 명령 프롬프트에서 sqlagent90을 실행합니다.
Sqlcmd	명령 프롬프트에서 T-SQL 문을 실행하는데 사용됩니다. sqlcmd 유틸리티를 사용하면 명령 프롬프트에서 Transact-SQL 문, 시스템 프로시저 및 스크립트 파일을 입력할 수 있습니다.
Sqldiag	콘솔 응용 프로그램 또는 서비스로 실행할 수 있는 범용 진단 정보 수집 유틸리티입니다. SQLdiag를 사용하여 SQL Server 및 기타 서버 유형에서 로그 및 데이터 파일을 수집할 수 있으며 이러한 파일을 사용하여 지속적으로 서버를 모니터링하거나 특정 서버 문제를 해결하는데 활용할 수 있습니다. SQL Server 2005에서는 SQLdiag 유틸리티가 획기적으로 변경된 관계로, 이 유틸리티의 명령줄 인수는 SQL Server 2000과 호환되지 않습니다.

Sqlmaint	이전 버전의 SQL Server에서 만들어진 데이터베이스 유지 관리 계획을 실행하는데 사용됩니다.
Sqlservr	SQL Server 인스턴스를 시작하고 중지하는데 사용됩니다. SQL Server 구성 관리자에서 SQL Server를 시작하고 중지하는 것이 원칙이며, 문제 해결을 목적으로 SQL Server를 시작하는 경우에만 명령 프롬프트에서 sqlservr.exe를 실행해야 합니다.
Sqlwb	명령 프롬프트에서 SQL Server Management Studio를 시작하는 데 사용합니다.
Tablediff	두 테이블에 존재하는 데이터를 비교하고 차이를 알려 줍니다. 일치하지 않는 두 테이블의 데이터를 비교하고자 할 때 사용하며, 특히 복제 토폴로지에서 데이터 불일치 문제를 해결하는 데 유용합니다.

주요 DBCC 명령어

다음은 DBA가 알고 있어야 하는 주요 DBCC 명령어입니다. 자세한 내용은 SQL Server 온라인 설명서를 참조 바랍니다.

DBCC 명령어	설명
DBCC CHECKALLOC	지정된 데이터베이스에 대한 디스크 공간 할당 구조의 일관성을 검사합니다.
DBCC CHECKCATALOG	지정한 데이터베이스 내의 카탈로그 일관성을 검사합니다.
DBCC CHECKCONSTRAINTS	현재 데이터베이스의 지정한 테이블에서 특정 제약 조건이나 모든 제약 조건의 무결성을 검사합니다.
DBCC CHECKDB	현재 데이터베이스에 있는 모든 테이블의 할당과 구조적 무결성을 검사합니다.

DBCC CHECKFILEGROUP	지정한 파일 그룹에서 현재 데이터베이스에 있는 모든 테이블의 할당과 구조적 무결성을 검사합니다.
DBCC CHECKIDENT	지정한 테이블의 현재 ID 값을 검사하고 필요에 따라 변경합니다.
DBCC CHECKTABLE	특정 테이블 또는 특정 인덱싱된 뷰를 구성하는 모든 페이지 및 구조의 무결성을 검사합니다.
DBCC DBREINDEX	지정한 데이터베이스의 테이블에 대해 하나 이상의 인덱스를 다시 작성합니다. 그렇지만 이 기능은 다음 버전에서 제거될 예정이므로 ALTER INDEX를 사용하는 것을 권고합니다.
DBCC DROPCLEANBUFFERS	버퍼 풀에서 버퍼를 모두 제거하는 명령어입니다. DBCC DROPCLEANBUFFERS를 사용하면 서버를 종료하고 다시 시작하지 않아도 완전히 빈 버퍼 캐시를 사용하여 쿼리를 테스트할 수 있습니다. 성능 튜닝 테스트에 활용할 수 있습니다.
DBCC FREEPROCCACHE	프로시저 캐시를 비우는 명령어입니다.
DBCC FREESESSIONCACHE	분산 쿼리에서 Microsoft SQL Server 인스턴스에 대해 사용한 분산 쿼리 연결 캐시를 플러시합니다.
DBCC FREESYSTEMCACHE	모든 캐시의 사용하지 않는 캐시 항목을 모두 해제합니다. SQL Server 2005 데이터베이스 엔진은 현재 항목에 필요한 메모리 확보를 위해 사용하지 않는 캐시 항목을 백그라운드에서 미리 정리합니다. 하지만 이 명령을 사용해 사용하지 않는 항목을 모든 캐시에서 직접 제거할 수 있습니다.
DBCC HELP	지정한 DBCC 명령의 구문 정보를 제공합니다.

DBCC INDEXDEFRAG	지정된 테이블 또는 뷰의 인덱스를 조각 모음합니다. 이 기능은 다음 버전에서 제거될 예정이므로, 이 명령어 대신 ALTER INDEX를 사용하는 것을 권고합니다.
DBCC INPUTBUFFER	클라이언트가 Microsoft SQL Server 2005 인스턴스로 마지막으로 전송한 SQL 문을 반환합니다.
DBCC OPENTRAN	지정한 데이터베이스에서 가장 오래된 활성 트랜잭션과 가장 오래된 분산 및 비분산 복제 트랜잭션에 대한 정보를 표시합니다. 활성 트랜잭션이 있거나 데이터베이스에 복제 정보가 있는 경우에만 결과가 반환됩니다.
DBCC OUTPUTBUFFER	지정한 session_id의 현재 출력 버퍼를 16진수와 ASCII 형식으로 반환합니다.
DBCC PINTABLE	제거된 기능입니다. 오류 메시지는 반환하지 않지만 실제로는 효력이 없습니다. 마찬가지로 DBCC UNPINTABLE도 제거되었습니다.
DBCC PROCCACHE	프로시저 캐시에 대한 정보를 테이블 형식으로 반환합니다.
DBCC SHOW_STATISTICS	지정한 테이블에서 특정 대상에 대한 현재 배포 통계를 제공합니다.
DBCC SHOWCONTIG	지정한 테이블의 데이터와 인덱스에 대한 조각화 정보를 제공합니다. 이 기능은 다음 버전에서 제거될 예정이므로 대신 sys.dm_db_index_physical_stats를 사용하는 것을 권고합니다.
DBCC SHRINKDATABASE	지정한 데이터베이스의 데이터 파일의 크기를 축소합니다.
DBCC SHRINKFILE	지정한 데이터베이스에서 지정한 파일(데이터 파일이나 로그 파일)의 크기를 축소합니다.

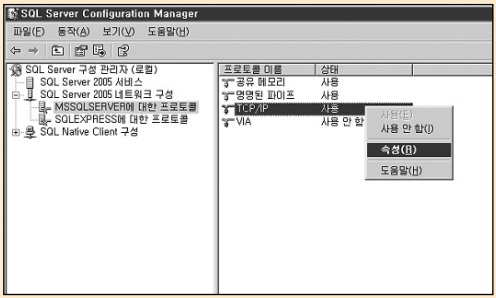
DBCC SQLPERF	모든 데이터베이스의 트랜잭션 로그 공간에 관한 통계를 제공합니다.
DBCC TRACEON	지정한 추적 플래그를 활성화합니다.
DBCC TRACEOFF	지정한 추적 플래그를 해제합니다.
DBCC TRACESTATUS	추적 플래그의 상태 정보를 제공합니다.
DBCC UPDATEUSAGE	카탈로그 뷰의 부정확한 페이지와 행 개수를 보고하고 수정합니다. 페이지와 행 개수가 부정확하면 sp_spaceused 시스템 저장 프로시저에서 반환하는 공간 사용 정보가 정확하지 않게 됩니다. SQL Server 2005에서는 항상 정확한 값이 유지 관리되기 때문에 SQL Server 2005에서 만든 데이터베이스에서는 개수가 부정확한 경우가 없지만 SQL Server 2005로 업그레이드한 데이터베이스에는 올바르지 않은 개수가 포함될 수 있으므로 SQL Server 2005로 업그레이드한 다음에는 DBCC UPDATEUSAGE를 실행하여 올바르지 않은 개수를 수정하는 것이 좋습니다.
DBCC USEROPTIONS	현재의 연결에 설정되어 있는 SET 옵션 정보를 제공합니다.

포트 변경

기본적으로 데이터베이스 엔진의 기본 인스턴스는 TCP 포트 1433에서 수신합니다. 그러나, 이 1433 포트는 보안을 위하여 변경하는 것이 좋습니다.

[따라하기] TCP 포트 확인한 후, 변경하기

1. [시작]→ [SQL Server 2005]→ [Configuration Tools]→ [SQL Server Configuration Manager]를 실행합니다.
2. 왼쪽 메뉴에서 MSSQLSERVER에 대한 프로토콜을 선택합니다.
3. 오른쪽 메뉴에서 [TCP/IP]에 커서를 대고 마우스 오른쪽 버튼을 클릭한 후, [속성]을 선택합니다.



4. TCP/IP 등록 정보 창이 나타납니다.

5. TCP 포트 열의 1433을 원하는 값으로 변경한 후, [확인]을 클릭합니다.

[참고] SQL Server Browser 서비스

SQL Server가 명명된 인스턴스로 실행하고 있는경우, 원격에서 해당 컴퓨터의 인스턴스 이름으로 접속하기 위해서는 SQL Server Browser 서비스가 실행되고 있어야 합니다. SQL Server Browser는 Microsoft SQL Server 리소스에 대해 들어오는 요청을 수신하고 컴퓨터에 설치된 SQL Server 인스턴스에 대한 정보를 제공합니다. SQL Server Browser 서비스가 실행되고 있지 않아도 특정한 포트 번호나 명명된 파이프를 제공하면 SQL Server 에 연결할 수 있습니다. 예를 들어 포트 SQL Server 의 기본 인스턴스가 포트 1433에서 실행 중이면 TCP/IP로 이 인스턴스에 연결할 수 있습니다.

DBA의 역할과 책임

DBA의 역할

시스템과 조직에 따라 DBA의 임무에 차이가 있을 수 있지만 일반적으로 대부분의 DBA는 다음과 같은 작업들을 책임지고 수행해야 하는 임무를 가집니다.

■ 설치와 환경설정
- 소프트웨어 설치
- 환경 설정

■ 보안 관리

■ 운영
- 백업과 복원
- 사용자 관리
- 기타 일상적인 운영 업무

■ 서비스 레벨 유지
- 성능 최적화 및 성능 모니터링
- 용량 계획 (Capacity Planning)

■ 시스템 가동 시간 관리
- 시스템 정지 시간의 계획과 일정 관리

■ 문서화 작업

■ 작업 절차 계획 및 규격화
- 운영 유지보수 계획 수립
- 재난 복구 계획 수립

■ 설계 및 개발 지원
- 데이터 모델링
- 데이터베이스 설계
- 저장 프로시저 개발
- 응용 프로그램 개발

■ 개발 환경 관리
- 개발 시스템 환경 별도 제공 및 개발 시스템 관리

■ 긴급 상황 해결/장애 복구
■ SQL Server 관리에 필요한 지식 숙지

DBA 작업의 기본적인 원칙

DBA가 시스템 유지를 위하여 일반적으로 수행하는 모든 작업들에 대하여 기본적으로 다음과 같은 원칙에 의거하여 작업할 것을 권고합니다.

■ 작업 표준화 체계 수립

표준화는 관리에 있어서 매우 중요한 요소입니다. 자신의 시스템에 가장 적합한 표준화 체계를 수립하고, 전체 시스템에 대하여 표준화된 관리 체계를 적용하여 관리해야 합니다. 예를 들어, 다중의 DB 서버를 관리하는 경우에는 표준화가 특히 중요합니다.

■ 문서화

DB 관리와 같이 중요한 작업은 사람의 기억에 의한 주먹구구식의 작업이 되어서는 안됩니다. 어떤 경우라도 항상 정확하고 일관된 작업이 가능하도록 문서화가 필요합니다. 기록 가능한 모든 작업들에 대해서 문서화하고, 변경이 발생하면 지속적으로 업데이트하는 관리가 필요합니다.

- **작업 매뉴얼** : 작업 수행 절차에 대한 정보 (설치, 장애 복구, 백업과 복원 전략, 주기적으로 수행하는 작업 등에 대한 작업 절차 및 참고 사항이 이에 포함될 수 있으며, 일반적이고 중요한 정보는 운영 매뉴얼에 기록하여 모든 DBA가 참조할 수 있도록 합니다.)

- **시스템 환경에 대한 정보** : 서버의 하드웨어, 소프트웨어, 네트워크 등에 대한 정보

- **담당자 및 관계자에 대한 정보** : 시스템과 관련된 내/외부 조직에 포함되는 모든 사람과 하드웨어/소프트웨어 제품 및 서비스 공급업체 및 담당자에 대한 정보

- 장애 기록 일지 : 발생된 문제와 문제 해결에 관한 모든 절차에 대한 기록 (장애 기록에 대한 내용은 활용 및 검색이 용이하도록 웹 기반으로 만들어, 유사한 문제의 재발 시에 신속하게 처리할 수 있도록 합니다.)

■ 스크립트화

반복적, 주기적으로 수행하는 모든 작업들은 스크립트를 작성하여 수행하는 것을 원칙으로 합니다. 스크립트를 사용하면 오류 발생 가능성을 최소화할 수 있으며 반복적인 작업을 효율적으로 수행할 수 있습니다. 스크립트는 보안을 위하여 안전한 디렉터리에 중앙 집중적으로 관리하는 것이 바람직하며, 스크립트 작성 시에는 응용 프로그램과 마찬가지로 주석을 기술하여 쉽게 이해하고 활용할 수 있도록 합니다. 만약 주석만으로 불충분한 경우에는 문서를 작성하여 관리합니다.

■ 자동화

주기적으로 수행해야 하는 작업들은 가능한 한 자동화하여 DBA의 업무 효율성을 제고할 것을 권고합니다. 예를 들어 DB 서버 성능 데이터의 수집, 디스크 공간의 확인, 백업, 블로킹 감지, 데이터 형식 오버플로우 감지 등의 작업들은 자동화가 가능합니다. 단순히 수행을 자동화하는 차원을 넘어서, SQL Server에서 제공하는 다양한 기능들을 활용하면 자동으로 경고 메일의 발송, 문자 메시지의 발신, 문제 해결을 위한 작업의 수행 등이 가능하기 때문에, DBA가 지속적으로 시스템을 모니터링하지 않더라도 시스템에 발생한 문제를 조기에 감지하는 것이 가능합니다. DBA가 주기적으로 수행되는 작업에 할애하는 시간은 가능한 한 최소화하고, 주기적인 관리 작업을 통하여 확보한 지식을 기반으로 응용 프로그램과 서버의 성능을 향상시키기 위한 전략을 모색하는데 많은 시간을 할애하는 것이 바람직합니다.

■ 신중한 변경 관리 및 롤백 전략 수립

운영중인 시스템에 어떤 변경작업을 수행하는 경우에는 가능한 한 충분한 사전 테스트를 거친 후에 작업해야 하며, 롤백 전략을 수립한 다음에 작업하는 것을 원칙으로 합니다. 또한 한번에 여러 가지 변경 작업을 수행하지 말고, 하나의 변경 작업을 수행하고 그 변경 작업이 미친 영향을 관찰하는 것이 바람직합니다.

모든 변경 작업에 대해서 롤백 전략을 수립하는 것이 원칙이며, 롤백에 필요한 사항들을 문서로 기록하고 롤백에 필요한 스크립트 등을 작성하고 테스트하여 검증합니다.
특히 대용량 데이터베이스의 경우에는 문제 발생 시 복구에 소요되는 시간이 길기 때문에 충분한 사전 테스트와 롤백 전략 수립이 매우 중요합니다.

DBA가 주기적으로 수행해야 하는 작업

시스템에 따라 차이가 있을 수 있지만, DBA는 시스템 유지를 위하여 일반적으로 수행해야 하는 작업들에 대하여 이해하고 있어야 하며, 다음과 같은 작업들을 주기적으로 수행해야 합니다. 다음 내용은 저자의 경험을 기준으로 분류한 것이므로, 시스템의 여건에 적합하게 최적화하는 것이 필요합니다.

■ 일 단위로 수행해야 하는 작업

- 시작되어야 할 서비스들이 제대로 시작되어 있는지 확인합니다.
- 오류 발생 여부를 점검합니다. Windows NT 또는 Windows 2003의 이벤트 뷰어를 통해서 확인하거나 SQL Server Management Studio의 개체 탐색기의 서버→관리→SQL Server 로그에 커서를 위치시키고 마우스 오른쪽 버튼을 클릭하여 뷰를 선택하면 SQL Server 로그 및 Windows 로그를 확인할 수 있습니다.
- SQL Server 오류 로그에 오류 메시지가 기록되어 있는지 점검합니다.
- 데이터베이스 파일과 로그 파일의 확장에 대비하여 디스크에 충분한 여유 공간이 있는지 확인합니다.
- 데이터베이스 파일과 로그 파일의 크기와 실제로 사용되는 공간을 모니터링하며, 공간 부족으로 자동 확장이 예상되는 경우에는 미리 파일을 확장하여 충분한 공간을 확보합니다.
- SQL Server 작업(Job)의 성공/실패 여부를 점검합니다.

- 매일 데이터베이스 전체 백업 또는 차등 백업을 수행하기로 되어 있는 경우라면, 데이터베이스 전체/차등 백업을 수행합니다. 자동화되어 있는 경우에는 백업이 성공적으로 수행되었는지 점검합니다. 데이터베이스 전체/차등 백업 주기는 시스템 여건과 복원 전략에 따라 달라집니다.

- SQL Server 트랜잭션 로그를 백업받습니다. 자동화되어 있는 경우에는 백업이 성공적으로 수행되었는지 점검합니다. 백업 주기는 시스템 여건에 따라 백업 주기는 분 단위, 시간 단위, 일 단위로 달라질 수 있으며, 트랜잭션 백업 주기에 따라 트랜잭션 로그 파일의 크기가 달라집니다. 참고로 복원이 불필요한 테스트 DB에 대해서는 복구 모델을 단순으로 설정하면 트랜잭션 로그에 대한 주기적인 관리를 줄일 수 있습니다.

- master, model, msdb, 데이터베이스도 변경 사항이 있으면 주기적으로 백업해야 합니다. 시스템 카탈로그의 변경이 이루어진 후에는 master 데이터베이스의 전체 백업을 수행합니다. 경고, 작업(Job), 운영자, 로그 전달(log-shipping), 복제, DTS 패키지 등에 변경이 발생한 다음에는 msdb를 백업해야 합니다. model 데이터베이스에 변경작업을 수행한 다음에는 model을 백업해야 합니다.

- 시스템 모니터를 사용하여 성능 카운터를 모니터링함으로써, 적절한 성능이 유지되고 있는지 점검합니다. 최소한 시스템 모니터에서 프로세서, 메모리, 디스크(I/O), 네트워크에 대한 카운터들은 필수로 점검해야 합니다. 문제 발생 시 또는 추가적인 분석이 필요한 경우에는 관련 성능 카운터들을 추가로 분석합니다.

- 복구 모델이 전체 복구가 아니라면, 최소 로깅 작업(Minimal-logged operation)을 수행한 다음에는 차등 백업을 수행합니다.

- 블로킹, 교착상태(Deadlock)의 발생 여부를 점검합니다.

- 오래 수행되는 쿼리 또는 리소스를 과다하게 사용하는 쿼리가 있는지 점검합니다.

- 문제가 발생하면 문제 해결을 위한 활동을 수행하며, 문제 분석 및 해결 과정에 대한 내용을 가능한 한 상세하게 문서화합니다.

- 통계 자동 갱신(Auto update statistics) 옵션이 비활성화되어 있는 데이터베이스의 테이블들에 대해서는 주기적으로 (예:매일, 매주) UPDATE STATISTICS 작업을 수행합니다.

■ 주간 단위로 수행해야 하는 작업

- 모든 시스템 데이터베이스와 운영중인 사용자 데이터베이스에 대한 전체/차등 데이터베이스 백업을 수행합니다.
- 통계 자동 갱신(Auto update statistics) 옵션이 비활성화되어 있는 데이터베이스의 테이블들에 대해서 UPDATE STATISTICS를 매일 또는 매주 수행합니다.
- 인덱스의 조각화를 제거합니다. ALTER INDEX를 사용하여 인덱스를 다시 작성 또는 다시 구성함으로써 물리적 조각화, 논리적 조각화를 제거할 수 있습니다. 자세한 내용은 온라인 설명서에서 ALTER INDEX를 참조하십시오.
- 대형 일괄 처리의 작업 등으로 인하여 로그 파일이 과다하게 확장된 경우에는 로그 파일의 사용되지 않는 여분의 공간을 제거합니다.

■ **월간 단위로 수행해야 하는 작업**

- 전체 운영 체제를 백업합니다.

- 최소 월 1회 모든 시스템 데이터베이스와 운영 데이터베이스에 대하여 전체 백업을 수행해야 합니다.

- SQL Server 2005에서는 DBCC CHECKDB를 실행할 필요가 없으며 SQL Server 2005에서는 동시성이 향상되었다고 기술되어 있습니다. 어떤 이유로 DBA가 데이터베이스 무결성을 주기적으로 점검하기를 원한다면, 동시성이 향상되었다고 하지만 가능하다면 운영 서버에서 직접 실행하지 말고 테스트 장비에 운영 데이터베이스를 복원한 다음에 복원된 데이터베이스를 대상으로 DBCC CHECKDB를 수행하여 무결성을 점검하는 것이 좋습니다.

- 성능 데이터를 수집하여 시스템이 충족시켜야 하는 기준과 비교하여, 성능 향상 및 향후의 용량 계획에 활용합니다.

[참고]
정확한 점검을 위해서는 모든 유지 관리 활동 작업에 대하여 로그를 저장하는 것이 필요합니다.

비매품

초보 DBA를 위한 SQL Server 2005 관리가이드

- 저 자: 전 헌 경
- 감 수: 하 성 희
- Contents 관련문의: hkjeon@adconsulting.co.kr

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

Microsoft SQL Server 2005

초보 DBA를 위한 SQl Server 2005 관리가이드

Microsoft
한국마이크로소프트(유)

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

SQL-200608-01