Microsoft SQL Server 2000

DBA라면 이 정도는 알아야 한다

Microsoft SQL Server

DBA 가이드

Microsoft

DBA라면 이 정도는 알아야 한다

Microsoft® SQL Server™ DBA 가이드

저자의 글

본 가이드가 DBA들이 데이터베이스 관리 업무에 대하여 좀 더 명확하게 이해하고 보다 체계적으로 관리 업무를 수행하는데 작은 밑거름이 되었으면 합니다. 또한, 본 가이드를 통해 좀 더 많은 정보를 알게 되는 계기가 되시기를 바랍니다.

저자 약력

전현경 / 에이디 컨설팅 SQL Server 책임 컨설턴트

- 싸이월드, 신세계닷컴, 인터파크지마켓 튜닝 컨설팅
- SQL Server 튜닝 소프트웨어 개발
- 옥션 DBA
- 인터넷 법률 사이트 전산팀장

감수자의 글

조직이나 시스템에 따라 DBA의 역할이 다를 수 있지만, 정보 시스템의 근간이 되는 데이터베이스를 관리하는 일은 매우 중요한 일입니다. DBA의 기술력과 업무에 임하는 자세가 데이터베이스의 품질과 서비스의 질에 큰 영향을 미칠 수 있으므로, 기술력을 확보하고 업무의 효율성을 향상시키며 데이터베이스의 품질을 개선하기 위한 노력을 지속적으로 기울여 주실 것을 당부드립니다. 진실된 노력의 끝에서 보람된 가치를 찾을 수 있을 것입니다.

감수자 약력

하성희 / 에이디컨설팅 대표

- SQL Server 컨설턴트로 활동 중 (싸이월드, 신세계닷컴, 에이스화재, 인터파크지마켓, 롯데카드, 삼성생명, 엔씨소프트, LG생활건강, 텔레메틱스, 부동산써브, SK 케미칼, 한빛소프트 등 다수의 SQL Server 튜닝 및 모델링 컨설팅)
- 옥션 DBA 팀장
- 마이크로소프트 SQL Server 기술지원 엔지니어/Data Access 팀장
- SYBASE DB 엔지니어 (다수의 기업체, 병원, 학교 DB 기술지원)

목차

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. 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 엔터프라이즈 관리자를 사용하는 대신, 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 가능성 여부 점검, 데이터 타입 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.

■ **일 단위로 수행해야 하는 작업**

- 시작되어야 할 서비스들이 제대로 시작되어 있는지 확인합니다.
- Windows NT 또는 Windows 2000의 이벤트 뷰어를 사용하여 오류 발생 여부를 점검합니다.
- SQL Server 오류 로그에 오류 메시지가 기록되어 있는지 점검합니다. 자세한 내용은 [SQL Server 오류 로그 보기]를 참조하십시오.
- 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 확

보합니다.

- SQL Server 작업(Job)의 성공/실패 여부를 점검합니다.
- 매일 데이터베이스 전체 백업 또는 차등 백업을 수행하기로 되어 있는 경우라면, 데이터베이스 전체 백업을 수행합니다. 자동화되어 있는 경우에는 백업이 성공적으로 수행되었는지 점검합니다. 데이터베이스 전체/차등 백업 주기는 시스템 여건과 복원 전략에 따라 달라집니다.
- SQL Server 트랜잭션 로그를 백업 받습니다. 자동화되어 있는 경우에는 백업이 성공적으로 수행되었는지 점검합니다. 백업 주기는 시스템 여건에 따라 분 단위, 시간 단위, 일 단위로 달라질 수 있으며, 트랜잭션 백업 주기에 따라 트랜잭션 로그 파일의 적정 크기가 달라집니다. 참고로 복원이 불필요한 테스트 DB에 대해서는 복구 모델을 단순으로 설정하면 트랜잭션 로그에 대한 주기적인 관리를 줄일 수 있습니다.
- Master, model, msdb, 배포(distribution) 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. CREATE INDEX WITH DROP_EXISTING 또는 DBCC DBREINDEX를 수행하여 인덱스를 재구성함으로써 물리적, 논리적 조각화를 제거할 수 있으며, DBCC INDEXDEFRAG를 사용하면 논리적인 조각화를 제거할 수 있습니다. 자세한 내용은 온라인 설명서를 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.
- DBCC CHECKDB를 수행하여 데이터베이스의 무결성을 점검합니다. DBCC CHECKDB를 수행하면 서비스나 다른 작업에 영향을 미칠 수 있으므로, 테스트 장비에 모든 시스템 데이터베이스와 운영 데이터베이스를 복원하고, 복원된 모든 시스템 데이터베이스와 운영 데이터베이스를 대상으로 DBCC CHECKDB를 수행하여 무결성을 점검하는 것이 바람직합니다.
- Sqldiag.exe를 수행하고 결과를 저장합니다.
- 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. 데이터베이스 유지 관리 계획 마법사와 SQL Server 작업(Job)에서는 자동으로 작업 결과를 저장하도록 설정 가능합니다.

데이터베이스 관리

데이터베이스 생성

번호	수칙	체크
01	트랜잭션 로그 파일은 로그 전용 드라이브에 배치합니다.	☐
02	트랜잭션 로그 파일을 저장할 디스크는 일반적으로 RAID10으로 구성합니다.	☐
03	데이터베이스를 만들 때 향후 예상되는 최대 데이터 크기를 고려하여 충분한 크기로 생성합니다.	☐
04	파일이 증가할 수 있는 최대 크기를 지정하는 것을 권고합니다.	☐
05	파일이 자동으로 증가하도록 설정하는 경우에는 자동 확장 증가 크기를 적절하게 설정합니다.	☐
06	파일 그룹을 사용하여 데이터를 배치합니다.	☐
07	데이터베이스를 생성한 경우에는 master 데이터베이스를 백업합니다.	☐
08	Tempdb는 I/O가 빠른 쪽에 배치할 것을 권고합니다.	☐

수칙1. 트랜잭션 로그 파일은 로그 전용 드라이브에 배치합니다.

데이터 파일들과 트랜잭션 로그 파일은 서로 다른 디스크에 배치합니다.
모든 데이터베이스는 최소 하나의 주 데이터 파일(Primary Data File)과 하나의 트랜잭션 로그 파일로 구성됩니다. 트랜잭션 로그 파일은 별도의 드라이브에 배치합니다.

[따라하기] 주 데이터 파일은 D 드라이브에 배치하고 트랜잭션 로그 파일은 E 드라이브에 배치하는 데이터베이스 생성하기

확장명은 파일의 용도를 정확하게 구분 할 수 있도록 주 데이터 파일은 .mdf, 보조 데이터 파일은 .ndf, 트랜잭션 로그 파일은 .ldf를 사용합니다.

```
USE master
GO
CREATE DATABASE sample                    /* 데이터베이스 이름 */
ON (
    NAME = sample_dat,                    /* 데이터 파일 이름 */
    FILENAME = 'd:\DBdata\sample_dat.mdf', /* 데이터 파일 위치 */
    SIZE = 100 MB,                        /* 데이터 파일 초기 크기 */
    MAXSIZE = 1 GB,                       /* 데이터 파일 최대 크기 */
    FILEGROWTH = 100 MB)                  /* 데이터 파일 증가량 */
LOG ON (
    NAME = sample_log,                    /* 로그 파일 이름 */
    FILENAME = 'e:\DBlog\sample_log.ldf', /* 로그 파일 위치 */
    SIZE = 20 MB,                         /* 로그 파일 초기 크기 */
    MAXSIZE = 500 MB,                     /* 로그 파일 최대 크기 */
    FILEGROWTH = 50 MB)                   /* 로그 파일 증가량 */
GO
```

수칙2. 트랜잭션 로그 파일을 저장할 디스크는 일반적으로 RAID10으로 구성합니다.

트랜잭션 로그 파일의 경우에는 복제가 구성되어 있거나 트리거가 빈번하게 수행되는 경우가 아니라면 대부분의 IO가 쓰기 작업이므로, 쓰기 작업의 성능을 위하여 트랜잭션 로그 파일은 RAID 10에 저장할 것을 권고합니다. 참고로, 로그에서 대기가 발생하는지는 다음 명령어로 확인할 수 있습니다.

```
DBCC SQLPERF (WAITSTATS)
GO
```

수칙3. 데이터베이스를 만들 때 향후 예상되는 최대 데이터 크기를 고려하여 충분한 크기로 생성합니다.

주 데이터 파일, 트랜잭션 로그 파일 모두 충분한 크기로 생성합니다. 파일이 증가하는 동안에는 쓰기 작업은 대기 상태가 되기 때문에 잦은 확장은 성능에 좋지 않은 영향을 미칠 수 있습니다. 트랜잭션 로그를 자동 증가하도록 옵션을 설정하되, 되도록이면 로그의 사이즈가 증가될 필요가 없도록 합니다. 트랜잭션 로그의 초기 크기는 트랜잭션 로그 백업을 수행한 후, 다음 로그 백업이 수행되기 전까지 발생하는 작업들을 저장하기에 충분한 크기로 생성합니다. 트랜잭션 로그 파일에 대하여 여러 번의 자동 증가가 발생하게 되면, 여러 개의 가상 로그 파일들로 조각화되어, 로그 관련 작업의 성능에 좋지 않은 영향을 미칩니다. 가상 로그 파일을 줄이는 방법은 [데이터 베이스 축소하기]를 참조하십시오.

만약 데이터베이스의 초기 크기가 작아서 확장이 발생하고 있다면, 파일의 크기를 충분한 크기로 확장하기 바랍니다.

[따라하기] 데이터베이스 파일 확장하기

```
ALTER DATABASE Sample
MODIFY FILE
    (NAME = sample_dat,
    SIZE = 2GB)
GO
```

수칙4. 파일이 증가할 수 있는 최대 크기를 지정하는 것을 권고합니다.

파일의 최대 크기를 지정하면, 파일의 크기가 증가하여 디스크 여유 공간이 전혀 없는 상태가 되는 것을 방지할 수 있습니다. 파일의 최대 크기를 지정하려면 CREATE DATABASE 문의 MAXSIZE 매개 변수를 사용하거나 엔터프라이즈 관리자의 등록 정보 대화 상자의 파일 증가 제한(MB) 옵션을 사용하면 됩니다.

만약 기존의 데이터베이스 파일의 최대 크기가 UNLIMITED로 설정되어 있다면, 최대 크기를 설정하기 바랍니다.

[따라하기] 데이터베이스 파일의 최대 크기 확인 및 설정하기

```
EXEC sp_helpdb Sample
-- 또는
EXEC Sample..sp_helpfile
GO
-- 결과 중 maxsize 정보를 확인 후 다음 명령어를 수행합니다.
ALTER DATABASE Sample
MODIFY FILE
    (NAME = sample_dat,
    MAXSIZE = 3GB)
GO
```

수칙5. 파일이 자동으로 증가하도록 설정하는 경우에는 자동 확장 증가 크기를 적절하게 설정합니다.

파일의 크기가 매우 작거나 매우 큰 경우에는 파일 자동 확장 증가분을 퍼센트 단위가 아닌 MB 단위로 지정하는 것을 권고합니다. 파일의 자동 확장 증가분을 지정하지 않으면 디폴트 값이 10% 확장으로 설정되는데, 데이터베이스의 크기가 큰 경우에는 새로운 데이터를 저장할 공간이 없어서 자동 확장이 이루어질 때 소요시간이 오래 걸림으로 인하여 트랜잭션 로그를 발생시키는 작업들이 대기 또는 실패하는 문제가 발생할 수 있습니다. 예를 들어, 데이터 파일의 크기가 100GB인 경우에 파일의 자동 확장 증가분이 10%로 설정되어 있다면, 자동 확장이 발생할 경우 10GB의 파일 확장을 수행합니다.

10GB의 확장작업은 상당한 시간이 걸리는 작업이므로 정상적인 서비스를 하지 못하는 문제를 유발할 수 있습니다. 반대로 파일의 크기가 매우 작은 경우에는 10%씩 증가하면 확장되는 크기가 작아서 빈번하게 재확장이 발생합니다.

로그 파일의 경우에 작은 크기의 확장이 여러 번 발생하면 여러 개의 작은 가상 로그 파일(VLF)들로 단편화가 발생하게 되어 성능을 저하시킬 수 있으므로 유의하기 바랍니다. 가상 로그 파일의 수는 일반적으로 25개 미만으로 유지하는 것을 권고하며, 가상 로그 파일의 수가 지나치게 많은 경우에는 가상 로그 파일의 수를 줄이는 작업을 수행하는 것이 좋습니다. 작업 방법은 [트랜잭션 로그 파일 축소하기]를 참조하십시오.

[따라하기] 데이터베이스의 데이터 파일 증가율 100MB로 변경하기

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

수칙6. 파일 그룹을 사용하여 데이터를 배치합니다.

주 데이터 파일에는 메타 데이터만 저장하고, 사용자 오브젝트들은 사용자 정의 파일 그룹에 저장하며, 디폴트 파일 그룹을 주 파일 그룹이 아닌 사용자 정의 파일 그룹으로 변경할 것을 권고합니다. 참고로 데이터베이스는 주 파일 그룹과 사용자 정의 파일 그룹으로 구성되며 주 파일이 있는 파일 그룹이 주 파일 그룹이 되고 주 파일 그룹에는 모든 시스템 테이블이 저장됩니다.

데이터베이스에 오브젝트를 만들 때 파일 그룹을 지정하지 않으면 오브젝트들은 디폴트 파일 그룹에 저장되며 디폴트로 주 파일 그룹이 디폴트 파일 그룹이 됩니다.

[따라하기] 파일 그룹이 있는 데이터베이스 생성하기

```
USE master
GO
CREATE DATABASE sample2
ON Primary (                                    /* PRIMARY 파일 그룹 */
    NAME = sample2_Pri_dat,
    FILENAME = 'D:\DBdata\sample2_pri_dat.mdf',
    SIZE = 200 MB,
    MAXSIZE = 1 GB,
    FILEGROWTH = 20 MB),
```

```
FILEGROUP Sample2FG1 (                          /* 두 번째 파일 그룹 */
    NAME = sample2_FG1_dat,
    FILENAME = 'E:\DBdata\sample2_FG1_dat.ndf',
    SIZE = 200 MB,
    MAXSIZE = 1 GB,
    FILEGROWTH = 20 MB),
FILEGROUP Sample2FG2 (                          /* 세 번째 파일 그룹 */
    NAME = sample2_FG2_dat,
    FILENAME = 'F:\DBdata\sample2_FG2_dat.ndf',
    SIZE = 200 MB,
    MAXSIZE = 1 GB,
    FILEGROWTH = 20 MB)
LOG ON (                                        /* 로그 파일 */
    NAME = sample2_log,
    FILENAME = 'G:\DBlog\sample2_log.ldf',
    SIZE = 10 MB,
    MAXSIZE = 50 MB,
    FILEGROWTH = 5 MB)
GO
```

[따라하기] 디폴트 파일 그룹 확인 및 변경하기

```
USE Sample2
SELECT * FROM sysfilegroups WHERE status = 16
GO
/* 'Primary' 파일 그룹이 디폴트 파일 그룹이면 1이 반환되고 그렇지 않으면 0이
반환됩니다. */
SELECT FILEGROUPPROPERTY('Primary', 'IsDefault')
GO
/* 디폴트 파일 그룹을 'Sample2FG1' 로 변경합니다. */
```

```
ALTER DATABASE Sample2
MODIFY FILEGROUP [Sample2FG1] DEFAULT
GO
```

수칙8. tempdb는 I/O가 빠른 쪽에 배치할 것을 권고합니다.

Tempdb는 I/O가 빠른 쪽에 배치하는 것이 성능을 위해 좋습니다. Tempdb를 여러 디스크에 스트라이핑하면 더욱 좋습니다. 또한 tempdb를 자주 쓰는 사용자 데이터베이스와 물리적으로 격리된 디스크에 배치할 것을 권고합니다. 특히 tempdb를 매우 많이 사용하는 대규모 시스템이라면 tempdb를 별도의 디스크 세트에 배치하면 더 나은 성능 향상을 기대할 수 있습니다. 유의할 사항은, 데이터베이스 데이터와 운영 시스템의 페이징 파일을 동일한 디스크에 배치하는 것은 어떤 경우라도 좋은 방법이라고 할 수 없습니다.

[참고] 그 외 파일 배치하기

운영 시스템은 RAID 1로 구성된 어레이에 있어야 합니다. 페이징 파일은 운영 시스템이 있는 드라이브에서 훨씬 잘 동작하며, 데이터베이스에 별도의 디스크를 할당할 수 있게 하기 위해서 같은 위치에 있어도 관계 없습니다. 페이징 파일을 옮겨야 할 필요가 있다면, 데이터베이스의 데이터, 로그, tempdb가 있는 곳에는 위치시키지 않아야 합니다. 이렇게 해야 디스크 오류에 빠르게 대응하여 복구할 수 있습니다.
이 때, 부트 디스크는 미러를 이용하여 부팅 가능해야 합니다.
시스템 백업을 동일 서버에 저장해야 한다면 반드시 데이터나 로그 파일이 없는 다른 디스크에 저장해야 합니다.
파일 그룹의 파일 수는 SQL Server의 성능과 관련이 없으므로, 파일은 관리하기 쉽게 배치합니다.

[참고] CREATE DATABASE가 실패하는 경우 문제 해결하기

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

A. model 데이터베이스가 사용 중일 때
model 데이터베이스를 사용하는 프로세스의 수행이 완료되기를 기다렸다가 재수행하거나, model 데이터베이스를 사용 중인 프로세스를 강제로 중지한 후에 재수행합니다.

B. 데이터베이스 파일의 물리적인 위치를 잘못 지정했을 때
지정한 폴더가 실제로 있는지 확인합니다.
지정한 드라이브에 충분한 여유 공간이 있는지 확인합니다.

C. CREATE DATABASE를 수행한 사용자에게 새로운 데이터베이스를 생성할 수 있는 권한이 없을 때
새로운 데이터베이스를 생성하기 위해서는 sysadmin 또는 dbcreator 역할의 구성원이어야 합니다. 이 역할의 구성원에게 작업을 요청하거나, 주기적으로 작업이 필요하다면 해당 사용자를 dbcreator 역할에 추가하면 됩니다.

D. 동일한 이름의 데이터베이스가 이미 존재할 때
sp_helpdb를 수행하거나 master..sysdatabases 테이블을 참조하여 확인합니다. 만약 기존의 데이터베이스가 불필요하다면 sp_renamedb 를 사용하여 기존의 데이터베이스를 다른 이름으로 변경하거나 삭제한 후에 다시 시도합니다.

E. 동일한 이름의 파일이 이미 존재할 때
존재하지 않는 파일 이름을 지정하고 다시 시도합니다.

데이터베이스 삭제하기

[구문]

```
USE master
GO
DROP DATABASE database_name [ ,...n ]
GO
```

[유의사항]

DROP DATABASE를 수행하면 모든 데이터베이스 파일들도 디스크에서 삭제됩니다. 만약 다시 복구하고자 하는 경우에는 백업본을 복원해야 합니다. 그러므로, 만약 다시 참조할 필요가 있는 데이터베이스를 삭제하고자 하는 경우에는 sp_detach_db를 사용하여 SQL

Server에서 데이터베이스 정보만 삭제하고 파일들은 디스크에 남겨 둘 것을 권고합니다. [데이터베이스 파일 위치 변경하기]를 참조하십시오.
데이터베이스를 삭제하면 master 데이터베이스의 시스템 테이블이 업데이트 되므로 데이터베이스가 삭제된 후에는 master 데이터베이스를 백업할 것을 권고합니다. master 데이터베이스를 복원할 필요가 있을 때, 마지막 master 백업 이후 삭제된 데이터베이스가 시스템 테이블에 남아 있으면 그로 인하여 오류가 발생할 수 있습니다.

[참고] DROP DATABASE가 실패하는 경우 문제 해결하기

삭제하고자 하는 데이터베이스를 다른 프로세스에서 연결 중이면 데이터베이스를 삭제할 수 없습니다. 데이터베이스 사용 중이어서 삭제할 수 없는 경우에는, 해당 데이터베이스를 사용하는 프로세스들이 완료되기를 기다렸다가 삭제하거나 아니면 다음과 같이 데이터베이스를 단일 사용자 모드로 변경한 다음에 삭제하거나 또는 spid를 확인하여 KILL 명령어로 프로세스들을 중지한 다음에 재시도합니다.

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

데이터베이스 이전하기

[따라하기] 사용자 데이터베이스 이전하기

주 데이터 파일과 트랜잭션 로그 파일을 다른 드라이브로 이전하는 예제입니다.

데이터베이스명	SAMPLE
데이터 파일명	sample_dat
변경 전 데이터 파일 위치	D:\DBdata\sample_dat.mdf
변경 후 데이터 파일 위치	F:\DBdata\sample_dat.mdf
로그 파일명	sample_log
변경 전 로그 파일 위치	E:\DBlog\sample_log.ldf
변경 후 로그 파일 위치	G:\DBlog\sample_log.ldf

1. 데이터베이스에 연결되어 있는 연결을 모두 비 연결 상태로 만들고, 단일 사용자 모드로 설정합니다. 다음은 5초 후에 모든 작업들이 ROLLBACK되고, 연결을 끊는 예제입니다.

```
USE master
GO
ALTER DATABASE Sample
SET SINGLE_USER
WITH ROLLBACK AFTER 5
GO
```

[참고] EXEC sp_dboption database_name, 'single user', true
GO

이 명령어를 수행하는 세션 외에 다른 세션에서 해당 데이터베이스에 연결을 맺고 있는 경우에는, 데이터베이스를 단일 사용자 모드로 변경할 수 없습니다. 이 방법은 DBA가 해당 데이터베이스에 연결되어 있는 모든 세션들을 강제로 중지하거나 또는 사용자들이 스스로 해당 데이터베이스에 대한 연결을 해제한 다음에 사용할 수 있습니다.

2. 해당 데이터베이스의 모든 데이터 파일과 트랜잭션 로그파일의 경로를 확인합니다.

```
EXEC sp_helpdb Sample
GO
```

3. 데이터베이스와 파일을 분리합니다.

```
EXEC sp_detach_db 'Sample', 'true'
GO
```

4. 데이터베이스 파일들을 원하는 위치에 복사합니다.
sample_dat.mdf, sample_log.ldf 파일을 각각 f:\DBdata, g:\DBlog 폴더로 복사합니다.

5. 변경된 위치의 파일 경로를 지정하여 데이터베이스를 서버에 추가합니다.

```
EXEC sp_attach_db 'sample'
, 'F:\DBData\sample_dat.mdf'
, 'G:\DBLog\sample_log.ldf'
GO
```

Tempdb 위치 변경하기

[따라하기] Tempdb를 디스크 상의 다른 위치로 이전하기

아래 예제는 tempdb에만 적용할 수 있으며, 사용자 데이터베이스를 이동하고자 하는 경우에는 sp_detach_db와 sp_attach_db를 사용하기 바랍니다. 이에 대한 자세한 내용은 [데이터베이스 이전하기]를 참조하십시오.

1. tempdb 데이터베이스의 논리 파일 이름을 확인합니다.

```
USE tempdb
GO
EXEC sp_helpfile
GO
/* 결과
tempdev    1
C:\Program Files\Microsoft SQL Server\MSSQL\data\tempdb.mdf    PRIMARY
        102400 KB    Unlimited        10%              data only
templog    2
C:\Program Files\Microsoft SQL Server\MSSQL\data\templog.ldf    NULL
        20480 KB     Unlimited        5120 KB          log only
*/
```

2. ALTER DATABASE 명령어를 사용하여 파일의 위치를 변경합니다.

```
USE master
GO
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
```

3. SQL Server 를 중지한 후 다시 시작합니다.

4. SQL Server 서비스가 시작된 다음에, 다음의 확인작업을 수행합니다.

```
USE tempdb
GO
EXEC sp_helpfile
GO
```

5. 기존의 tempdb 파일들을 삭제합니다.

데이터베이스 파일 변경하기

■ 파일 크기 확장하기

[따라하기] Sample 데이터베이스 sample_dat 논리 파일을 200MB로 확장하기

```
ALTER DATABASE sample
MODIFY FILE
    (NAME = sample_dat,
    SIZE = 200MB)
GO
```

■ 파일 증가 규칙 변경하기

[따라하기] Sample 데이터베이스 sample_dat 논리 파일의 증가율을 5MB로 변경하기

```
ALTER DATABASE sample
MODIFY FILE
    (NAME = sample_dat,
    FILEGROWTH = 5MB)
GO
```

■ **새로운 파일 그룹 추가하기**

[따라하기] 파일 그룹 추가하기

Sample 데이터베이스에 sample_Fg 파일 그룹을 추가한 후, 그 파일그룹에 Sample_New 파일을 추가합니다. 주 데이터 파일이 아닌 데이터 파일의 확장자는 .ndf를 사용합니다.

```
ALTER DATABASE Sample
ADD FILEGROUP Sample_FG
GO
ALTER DATABASE Sample
ADD FILE
    (NAME = Sample_New,
    FILENAME = 'f:\DBdata\Sample_New.ndf',
    SIZE = 10MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 3MB)
TO FILEGROUP Sample_FG
GO
```

데이터베이스 축소하기

IsAutoShrink 데이터베이스 옵션을 true로 설정하여 데이터베이스가 축소되도록 할 수는 있지만 성능적인 측면에서 IsAutoShrink 옵션은 비활성화하고, 데이터베이스 축소가 필요한 시점에 DBCC SHRINKDATABASE 명령어나 DBCC SHRINKFILE 명령어를 사용하여 수동으로 데이터베이스 파일의 크기를 축소할 것을 권고합니다. 데이터베이스의 특정 데이터 파일 또는 트랜잭션 로그 파일을 축소하는 경우에는 DBCC SHRINKFILE을 사용합니다.

■ 파일 지정 없이 축소하기

[따라하기] 파일 지정 없이 Sample 데이터베이스 전체 크기 중에서 10%의 여유공간이 남도록 파일 크기를 축소합니다.

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

■ 특정 파일 축소하기

[따라하기] Sample 데이터베이스의 sample_dat 파일을 10MB로 축소합니다.

```
USE Sample
GO
DBCC SHRINKFILE (sample_dat, 10)
GO
```

■ 가상 로그 파일 축소하기

트랜잭션 로그 파일이 여러 번 자동 증가가 발생한 경우, 여러 개의 가상 로그 파일들로 조각화되어, 로그 관련 작업의 성능에 좋지 않은 영향을 미칩니다. 로그 파일의 크기가 매우 큰 경우가 아니라면 가상 로그 파일이 25개 이상일 경우, 가상 로그 파일을 제거하고, 트랜잭션 로그 파일을 적절한 크기로 변경합니다.

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

1. 가상 로그 파일 정보를 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 삭제합니다.

```
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.

```
EXEC sp_helpfile
GO
DBCC SHRINKFILE (Sample_log, TRUNCATEONLY)
GO
```

4. 로그 파일의 크기를 적절하게 변경합니다.

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

데이터베이스 옵션 설정하기

[따라하기] 데이터베이스 옵션 확인하기

SQL Server 2000에서 제공하는 함수인 DATABASEPROPERTYEX를 사용하면 데이터베이스 옵션이나 속성의 현재 설정값을 확인할 수 있습니다.

```
SELECT DATABASEPROPERTYEX ('pubs', 'IsAutoUpdateStatistics')
GO
```

[참고]

새로운 데이터베이스를 만들 때 FOR ATTACH가 지정된 경우를 제외하면 model 데이터베이스의 데이터베이스 옵션 설정을 상속받습니다. 예를 들어, 디폴트로 model 데이터베이스의 IsAutoUpdateStatistics 옵션이 TRUE이기 때문에, 모든 데이터베이스들의 IsAutoUpdateStatistics 옵션이 TRUE로 설정됩니다. ALTER DATABASE를 사용하여 model 데이터베이스의 옵션을 변경하면, 그 이후에 새로 만들어지는 모든 데이터베이스에 변경된 옵션이 적용됩니다. 사용자 데이터베이스의 옵션에 대한 표준안이 정해지면 model 데이터베이스의 옵션을 변경해 두면 편리합니다.

```
/* model 데이터베이스의 옵션 설정 */
USE model
GO
ALTER DATABASE model
SET AUTO_UPDATE_STATISTICS OFF,
RECOVERY BULK_LOGGED
GO
/* model 데이터베이스의 변경된 옵션정보 확인 */
SELECT DATABASEPROPERTYEX ('model', 'IsAutoUpdateStatistics')
SELECT DATABASEPROPERTYEX ('model', 'Recovery')
GO
```

데이터베이스 소유자 변경하기

[따라하기] sample 데이터베이스의 소유자를 'dbadmin' 으로변경하기

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

데이터베이스 이름 변경하기

다른 사용자가 데이터베이스를 사용하지 않아야 성공적으로 이름이 변경됩니다.

[따라하기] sample 데이터베이스의 이름을 sample_rename으로 변경하기

```
EXEC sp_renamedb Sample, Sample_Rename
GO
-- 이후의 테스트를 위하여 다시 원래 이름으로 변경합니다.
EXEC sp_renamedb Sample_Rename, Sample
GO
```

백업과 복구

모든 데이터베이스 운영 환경은 반드시 장애 복구에 대한 백업과 복구 계획을 가지고 있어야 합니다. 백업과 복구 계획은 실제 운영 서버를 백업하여 실제와 동일한 상태로 철저히 테스트하고 문서화해야 합니다. 백업과 복구 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 얼마동안 다운되어도 무방한지, 어느 정도의 데이터가 유실되어도 되는지에 관련된 리소스 비용과 다운타임, 복구 비용을 고려합니다.

복구 모델

복구 모델은 트랜잭션 로그에 어떤 내용이 저장되었는지를 가리키는 것입니다. SQL Server Standard Edition, Enterprise Edition의 디폴트 복구 모델은 전체 백업 모드입니다. 디폴트 복구 모델은 데이터베이스가 만들어질 때 model 데이터베이스에 설정되어 있던 값에 의하여 결정됩니다.

■ 단순 복구 (Simple)

- 마지막으로 백업을 시행한 시점까지의 백업된 정보를 복구합니다.
- 전체 백업과 차등 백업은 가능하나, 트랜잭션 로그 백업은 수행할 수 없습니다.
- 이 모델은 응용 프로그램을 테스트하는 테스트 환경 또는 저장된 데이터를 복구할 필요가 전혀 없는 시스템에 적합합니다.

■ 전체 복구 (Full)

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

■ **대량 로그 복구 (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

```
ALTER DATABASE Sample
SET RECOVERY BULK_LOGGED
GO

ALTER DATABASE TestDB
SET RECOVERY SIMPLE
GO

ALTER DATABASE TestDB
SET RECOVERY FULL
GO
```

[참고] 복구 모델 전환 시 백업 전략

변경 전	변경 후	작업	설 명
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.

백업의 종류

■ 전체 백업

데이터베이스를 구성하는 모든 파일들을 백업합니다.
시스템 데이터베이스와 사용자 데이터베이스에 대해서 주기적으로 수행해 주어야 합니다.

■ 파일 또는 파일 그룹 백업

파일 그룹을 구성하는 파일들 중에서 하나의 파일이나 여러 개의 파일들을 백업합니다. 전체 백업보다 훨씬 빠르고, 업무 단위의 백업이 가능하여, 대량의 데이터베이스일 경우에 효율적이기는 하지만, 백업받은 데이터만 보호된다는 단점이 있습니다.

■ 차등 백업

마지막 전체 백업이 실행된 이후 변경된 정보를 백업합니다. 즉, 두 번째 차등 백업은 첫 번째 차등 백업과 중복되는 부분이 있으므로, 복원 시에는 전체 백업과 장애가 발생하기 전의 마지막 차등 백업을 복원하면 됩니다. 차등 백업 전략을 사용하면 복구 속도를 향상시킬 수 있습니다.

■ 트랜잭션 로그 백업

트랜잭션 로그 백업은 전체 복구 또는 대량 로그 복구 옵션으로 설정된 데이터베이스에서만 사용 가능하며, 이 경우 데이터베이스에 변경이 발생할 때마다 그 변경에 대한 모든 정보가 트랜잭션 로그에 기록됩니다. 트랜잭션 로그는 연속적으로 변경 내역을 저장합니다. 복원할 경우에는, 마지막 전체 백업을 실행한 시점부터 순차적으로 실행한 모든 트랜잭션 로그 백업이 필요합니다.
다시 말씀드리지만 복구 모델이 "단순 복구"일 경우에는, 트랜잭션 로그 백업은 사용할 수 없습니다.

백업 전략 세우기

전체 데이터베이스 백업은 항상 수행되어야 합니다. 일반적으로 트랜잭션 로그 백업은 대부분의 경우 수행합니다. 트랜잭션 로그 백업을 수행하지 않는 예외적인 경우는 데이터의 변경이 드물게 발생하거나 테스트 환경에서입니다. 차등 백업은 많은 트랜잭션이 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. 백업에 관한 정보는 엔터프라이즈 관리자를 사용하거나 쿼리 분석기에서 RESTORE 명령어를 수행하여 시스템 테이블을 쿼리하여 확인할 수 있습니다.

번호	수칙	체크
01	시스템 데이터베이스도 백업을 수행합니다.	☐
02	백업 전략은 복구 시간까지 감안하여 계획을 세웁니다.	☐
03	트랜잭션 로그를 정기적으로 백업하지 않는다면, 정기적으로 비워 줍니다.	☐
04	백업 파일은 데이터베이스 파일이 저장된 디스크와 물리적으로 다른 디스크에 저장합니다.	☐
05	주기적으로 백업 파일이 제대로 복원되는지 테스트합니다.	☐

수칙1. 시스템 데이터베이스는 변경이 발생할 때마다 백업해야 합니다.

사용자 데이터베이스뿐만 아니라, 시스템 데이터베이스에도 시스템에 관련된 중요한 정보들이 있으므로, 백업을 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

수칙2. 백업 전략은 복구 시간까지 감안하여 계획을 세웁니다.

백업전략은 데이터의 중요성, 데이터의 변경 주기, 복구 시간 등 여러 가지 요인들을 고려하여 수립합니다.

수칙3. 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 객체의 카운터 Percent Log Used의 성능 상태 경고를 통하여 모니터링 해야 합니다.

어떤 시스템의 경우에는 트랜잭션 로그 파일의 크기가 데이터 파일의 수십배에 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.

[예제] 트랜잭션이 완료된 로그 삭제하기

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

[참고]

데이터베이스가 단순 복구 모델이거나 "truncate log on checkpoint" 옵션이 선택되어 있을 때 트랜잭션 로그 백업을 하면, 엔터프라이즈 관리자에서는 트랜잭션 로그 옵션이 비활성화 상태가 되고, 쿼리 분석기에서는 4208 오류가 반환됩니다. 트랜잭션 로그 백업을 수행하기 위해서는, "truncate log on checkpoint" 옵션이 비활성화 상태라야 합니다.

수칙4. 백업 파일은 데이터베이스 파일이 저장된 디스크와 물리적으로 다른 디스크에 저장합니다.

디스크로 백업하고 별도의 위치로 백업 파일을 저장하는 것이 원칙입니다. 하드 디스크에 백업받는 경우에는 데이터베이스 파일이 저장된 디스크와 물리적으로 다른 디스크로 백업합니다.

수칙5. 주기적으로 백업 파일의 유효성과 백업이 실제로 정상적으로 복원되는지 테스트합니다.

[백업 검증하기]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 고객사를 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 성능 향상시키기

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, 백업성능이 좋지 않으며 네트워크 부하를 가중시킬 수 있으므로 유의하기 바랍니다.*

백업 검증하기

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 줄 수는 없습니다. 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 않았는지를 확인할 것을 권고합니다.

복원 전략 세우기

손상된 데이터베이스를 복구하는 첫번째 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.
복구 과정에서 복구되는 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 백업의 복구 시에는 RECOVERY 옵션을 사용합니다.

> ***[참고]***
> *트랜잭션 로그 백업이 RECOVERY 옵션으로 복구되면, 추가적인 로그는 복구될 수 없습니다. 만일 추가적인 로그가 존재하면, 복구 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 복원한 후, 새로운 서버의 로그인 정보를 복원한 데이터베이스의 사용자와 링크합니다. 사용자에 대한 로그인이 변경되는 경우에는 sp_change_users_login을 사용하면, 사용자의 권한을 상실하지 않고 새 로그인에 사용자를 링크할 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 복원 합니다.

```
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.

```
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 master..syslogins WHERE name='dbadmin'
SELECT sid FROM Sample..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 백업받습니다. (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 경우의 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' WITH INIT
월 15:00	BACKUP LOG Sample TO DISK='F:\DBBackup\sample_log.bak'

[따라하기] 문제가 발생하여 전체 데이터베이스 백업과 로그 백업으로 복구하기

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. (단계 1의 백업이 성공한 경우)

```
RESTORE LOG Sample
FROM DISK='F:\DBBackup\sample_log2.bak'
WITH RECOVERY
GO
```

■ 전체 백업과 파일 그룹 백업을 실행한 경우의 복원하기

백업 일시	백업
월 05:00	BACKUP DATABASE Sample2 TO DISK='F:\DBBackup\sample2.bak' WITH INIT
화 05:00	BACKUP DATABASE sample2 FILEGROUP = 'PRIMARY' TO DISK='F:\DBBackup\sample2_prm.bak'
화 17:00	BACKUP LOG Sample2 TO DISK='F:\DBBackup\sample2_log.bak'
수 05:00	BACKUP DATABASE Sample2 FILEGROUP = 'Sample2FG1' TO DISK='F:\DBBackup\sample2_FG1.bak'
수 17:00	BACKUP LOG Sample2 TO DISK='F:\DBBackup\sample2_log.bak'

[따라하기]

위의 백업을 실행 후에, Secondary 파일 그룹(Sample2FG1)이 깨졌다고 가정합니다. 전체 백업을 복구할 필요 없이, 파일 그룹 백업만으로 복구가 가능합니다. 대용량 데이터베이스일 경우, 파일 그룹 백업은 복원 시간 단축에 매우 효과적입니다. 참고로, 아래 스크립트는 파일 그룹에 관련되는 내용만 포함시킨 최소의 스크립트입니다.

1. Sample2FG1 파일 그룹의 백업을 복원합니다.

```
USE master
RESTORE DATABASE Sample2 FILEGROUP = 'Sample2FG1'
FROM DISK='F:\DBBackup\sample2_FG1.bak'
WITH FILE = 1, NORECOVERY
GO
```

2. 복원한 백업 이후의, 로그 백업을 순차적으로 복원합니다.

```
RESTORE LOG Sample2
FROM DISK='F:\DBBackup\sample2_log.bak'
WITH FILE = 2, RECOVERY
GO
```

■ 파일 위치 지정하여 복원하기

[따라하기]

sample_dat 데이터 파일은 "D:\DBData\" 에, sample_log 로그 파일은 "E:\DBLog\" 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 'D:\DBData\sample_dat.mdf'
, MOVE 'sample_log' TO 'E:\DBLog\sample_log.ldf'
, REPLACE
GO
```

[참고]

REPLACE 옵션은 지정한 위치에 같은 파일이 이미 존재할 때 사용합니다.

■ 지정 시간 복구하기

지정 시간 복구는 오직 트랜잭션 로그 백업 상태에서만 가능합니다. RESTORE 명령어에 STOPAT 옵션을 사용하면 날짜와 시간을 정하여 데이터베이스를 복구할 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.

[주의]

대량 로그 복구 모델 최소 로깅 작업을 수행한 경우에는 최소 로깅 작업 수행 이전까지만 지정 시간 복구가 가능합니다.

[따라하기]

2004년 12월 30일 오후 1시 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 13:00:00.000'
GO
```

■ 표시된 트랜잭션 복구하기

표시된 트랜잭션은 DBA가 잘못된 트랜잭션이 발생한 시점을 확인하는데 있어 유용하며, 보다 쉽게 복구를 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 복구하는 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 지점에

서 복구가 중지됩니다. AFTER datetime이 지정되면 지정한 일시 또는 지정한 시점 이후에 지정한 이름이 있는 첫 번째 표시 지점에서 복구가 중지됩니다.

[따라하기]

```
/* 테스트 테이블 생성 */
CREATE TABLE Tab_Sample (
    Col1    int identity(1,1)    NOT NULL PRIMARY KEY Nonclustered,
    Col3    int                  NULL
)
GO
INSERT Tab_Sample VALUES (1)
INSERT Tab_Sample VALUES (2)
GO
/* 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

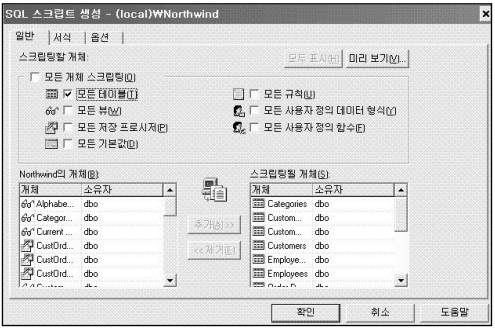
번호	수칙	체크
01	사용자 데이터베이스의 오브젝트 스크립트도 주기적으로 백업합니다.	☐
02	JOB 스크립트도 주기적으로 백업합니다.	☐
03	복제를 구성한 경우에는 복제 스크립트도 백업합니다.	☐

■ 오브젝트 스크립트 백업하기

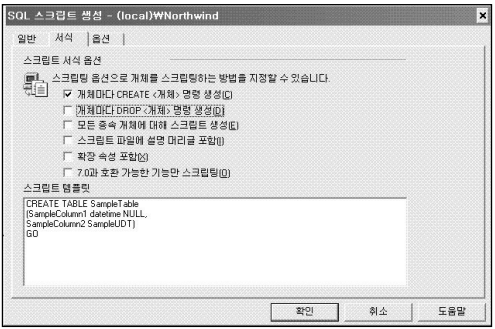
[따라하기]

1. 엔터프라이즈 관리자에서 스크립트를 생성하고자 하는 데이터베이스를 선택하고 마우스의 오른쪽 버튼을 클릭하여, [모든 작업] → [SQL 스크립트 생성]을 선택합니다.

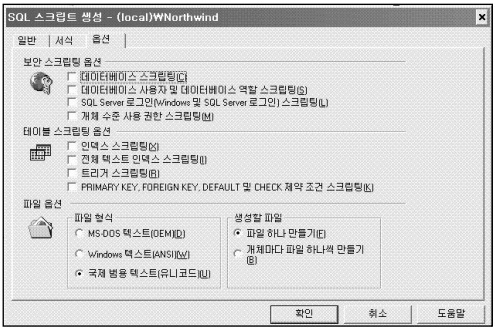
2. SQL 스크립트 생성 창의 [모두 표시]를 클릭하고, 필요한 경우 스크립트 백업 받기를 원하는 오브젝트 종류를 선택합니다. 테이블 스크립트를 저장하려고 한다면, [모든 테이블]을 선택합니다.



3. [서식] 탭을 선택합니다. [개체마다 DROP 〈개체〉명령 생성]의 선택을 제거할 것을 권고합니다. 생성된 SQL 스크립트를 실수로 운영 DB서버에서 수행하여 운영중인 오브젝트들이 모두 삭제되는 불상사가 간혹 발생하고 있으므로, 항상 DROP 옵션은 체크 해제한 상태에서 스크립트를 받을 것을 권고합니다.



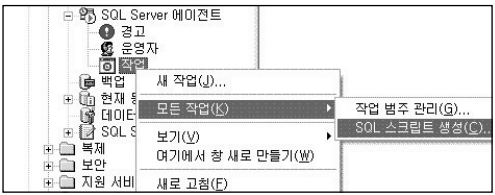
4. [옵션] 탭을 선택합니다. 필요한 옵션을 선택하고, [확인]을 클릭합니다, 테이블의 경우에는 일반적으로 제약 조건과 인덱스 스크립팅을 선택합니다. 예를 들어, 데이터베이스 내 모든 저장 프로시저들의 스크립트를 받고자 하는 경우에 [개체마다 파일 하나씩 만들기] 옵션을 선택하기도 하는데, 저장 프로시저의 수가 매우 많은 경우에는 이 옵션은 성능 문제를 유발할 수 있으므로 작업 부하가 가장 적은 시점에 사용하기 바랍니다.



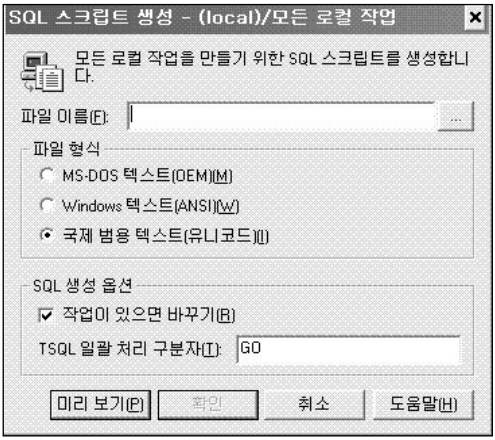
5. 저장할 파일명을 입력하고, [저장]을 클릭합니다.

■ JOB 스크립트 백업하기

1. EM의 [관리] → [SQL Server에이전트] → [작업] 위에서 마우스 오른쪽 버튼을 클릭하여 [모든 작업] →[SQL스크립트 생성]을 선택합니다.



2. SQL 스크립트 생성 창의 파일 이름을 입력하고, SQL 생성 옵션의 [작업이 있으면 바꾸기]의 선택을 제거합니다.



3. [확인]을 클릭합니다.

■ 복제 스크립트 백업하기

1. 게시자에서 SQL Server 엔터프라이즈 관리자를 열고, 서버 그룹을 확장하고, 복제 폴더를 마우스 오른쪽 단추로 클릭한 다음, SQL 스크립트 생성을 클릭합니다.
2. 스크립팅할 복제 구성 요소(배포자 속성, 게시 구독, 밀어넣기 구독 또는 끌어넣기 구독)를 선택하고 구성 요소를 생성하는 스크립트인지 구성 요소를 삭제하는 스크립트인지의 여부를 선택합니다.

테이블 관리

테이블 생성하기

번호	수칙	체크
01	동일한 속성의 데이터 타입은 동일하게 할당합니다.	☐
02	컬럼에 저장되는 데이터의 값과 특성을 고려하여 가장 적합한 데이터 타입을 선택합니다.	☐
03	데이터 무결성을 보장할 수 있도록 적절하게 제약 조건을 정의합니다.	☐
04	항상 값이 저장되는 컬럼에 대해서는 반드시 NOT NULL로 정의합니다.	☐

수칙1. 동일한 속성의 데이터 타입은 일관되게 동일하게 할당합니다.

동일한 속성을 가진 데이터를 서로 다른 테이블들에서 다른 데이터 타입으로 선언한 경우에는 데이터의 불일치 뿐 아니라 성능 저하를 유발할 수도 있으므로 유의하기 바랍니다.
동일한 속성임에도 불구하고 테이블에 따라 데이터 타입이 다른 경우도 있고 데이터 타입은 동일하지만 길이가 다른 경우도 있습니다. 또한 예를 들어, 주민등록번호나 계좌번호와 같은 성격의 데이터들에 대해서 어떤 컬럼은 char(13)으로 선언하고 '-' (하이픈) 없이 데이터를 저장하고 어떤 컬럼은 char(14)로 선언하여 '-' 을 추가하여 저장하는 경우가 있는데, 동일한 속성에 대해서는 동일한 데이터 타입, 동일한 데이터 포맷, 동일한 길이를 가지는 동일한 데이터 타입을 일관되게 사용해야 합니다.

수칙2. 컬럼에 저장되는 데이터의 값, 특성 등을 고려하여 적합한 데이터 타입을 선택합니다.

컬럼에 숫자만 저장되고 계산에 사용할 가능성이 있다면 숫자 데이터 타입을 할당합니다. 숫자 데이터 타입의 경우에는 tinyint, smallint, int, bigint의 네 가지 데이터 타입이 지원되므로 저장될 데이터 값의 범위를 확인하여 데이터 타입을 선택합니다. 예를 들어 0에서 255까지의 정수를 저장할 컬럼이라면 저장소 측면에서 int 대신 tinyint를 사용하는 것이 효율적이며, 21억이 넘는 큰 값이 저장될 컬럼이라면 bigint를 사용해야 오버플로우 오류가 발생하는 것을 방지할 수 있습니다.

소수점 이하 값이 없는 컬럼에 불필요하게 numeric, decimal 타입을 사용하는 경우를 볼 수 있는데, 소수점 이하 값이 없는 숫자형 데이터에 대해서는 numeric, decimal 대신 정수형 타입을 사용할 것을 권고합니다.
문자가 저장되는 컬럼은 문자 데이터 타입을 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 타입으로 설정하면, 하나의 테이블에 bit 타입이 여러 개 있는 경우에 레코드의 길이를 줄일 수 있습니다. 자세한 내용은 온라인 설명서를 참조하십시오.

[참고] 데이터 타입

분류	데이터 타입	범위	저장소 크기
정수	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	-2·63 ~ 2·63-1	8 바이트
부동소수점	Float[n]	-1.79E+308 ~ 1.79E+308 n = 1~24	4 바이트
	Float[n]	-1.79E+308 ~ 1.79E+308 n = 25~53	8 바이트
	Real	-3.40E + 38 ~ 3.40E + 38	4 바이트
문자데이터	char[n]	n = 1~8000	n 바이트
	Varchar[n]	n = 1~8000	입력한 데이터의 길이
	Text	최대 2,147,483,647자의 가변길이	
유니코드 문자데이터	Nchar	n = 1~4000	n*2 바이트
	nvarchar	n = 1~4000	입력한 데이터의 길이 *2 바이트

분류	데이터 타입	범위	저장소 크기
유니코드 문자데이터	Ntext	최대 1,073,741,823자의 가변길이	
이진데이터	binary	n = 1~8000	n+4 바이트
	varbinary	n = 1~8000	입력한 데이터의 길이+4 바이트
	Image	최대 2,147,483,647자의 가변길이	
날짜와시간	datetime	1753/1/1~9999/12/31	8 바이트
	smalldatetime	1900/1/1~2079/6/6	4 바이트
화폐	money	-922,337,203,685,477.5808~ +922,337,203,685,477.5807	8 바이트
	smallmoney	-214,748.3648~214,748.3647	4 바이트

[참고]
주요 키 컬럼에 대해서는 사용자 정의 데이터 타입을 활용하면 편리합니다.

수칙3. 데이터 무결성을 보장할 수 있도록 제약 조건을 적절하게 정의합니다.

데이터 무결성을 유지하여 데이터베이스의 품질을 보장할 것을 권고합니다. 테이블을 계획할 때 필요한 두 가지 주요 단계가 컬럼에 대하여 유효한 값이 무엇인지 확인하고 컬럼에 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. 데이터 무결성에 대한 자세한 내용은 온라인 설명서에서 "데이터 무결성"에 기술되어 있는 내용을 참조하십시오. SQL Server 온라인 설명서에서 [검색] 탭을 클릭한 다음에 "검색할 단어 입력" 란에 "데이터 형식"을 입력하여 검색하면 쉽게 찾을 수 있습니다.

보다 자세한 내용과 예제는 SQL Server 2000 온라인 설명서에서 제목 "CREATE TABLE"의 내용을 참조하십시오.

PRIMARY KEY 제약 조건

테이블 생성 시에는 PRIMARY KEY 제약 조건을 지정합니다. PRIMARY KEY 제약 조건은 테이블을 생성할 때에 생성하는 것이 바람직하지만, PK 제약 조건을 정의하지 않았더라도 테이블 생성 후에 추가로 생성할 수 있습니다. PRIMARY KEY 제약 조건을 설정할 컬럼은 NOT NULL 속성으로 정의되어야 하며, 고유한 데이터를 가지는 컬럼이어야 합니다. 두 개 이상의 컬럼에 PRIMARY KEY가 정의될 때에는, 각각의 컬럼에는 중복된 값이 있을 수 있지만, 키 컬럼들을 조합한 값은 고유해야 합니다.

CREATE TABLE 문에서 PRIMARY KEY를 정의하는 구문에 인덱스의 종류를 지정하지 않으면 디폴트로 PK 키 컬럼(들)에 Clustered Index가 생성됩니다. 어떤 시스템의 경우에는 모든 테이블들의 PK가 무조건 clustered index로 만들어져 있고 또 어떤 시스템의 경우에는 모든 테이블들의 PK가 무조건 nonclustered index로 만들어져 있는 경우를 볼 수가 있는데, PK 제약 조건은 Clustered Index와 Nonclustered Index 두 가지 중 성능적인 측면에서 보다 효율적인 인덱스를 사용해야 한다는 것에 유의하기 바랍니다. PK를 정의할 때에는 테이블을 만들기 전에 쿼리를 종합적으로 분석하여 어떤 인덱스를 사용하는 것이 가장 효율적일지 고려하여 인덱스 유형을 정의하기 바랍니다.

[따라하기]

테이블 생성 시 PRIMARY KEY 제약 조건을 Nonclustered Index로 설정하기

```
CREATE TABLE T1 (
    Col1    int     NOT NULL PRIMARY KEY Nonclustered,
    Col2    char(3) NULL,
    Col3    int     NULL
)
GO
```

테이블 생성 후, PRIMARY KEY 제약 조건 설정하기

```
CREATE TABLE T1 (
    Col1        int         NOT NULL,
    Col2        char(3)     NULL,
    Col3        int         NULL
)
GO
ALTER TABLE T1
    ADD CONSTRAINT PK_T1 PRIMARY KEY Nonclustered (Col1)
GO
```

UNIQUE 제약 조건

PRIMARY KEY가 아닌 컬럼(들)에 항상 고유한 값이 저장된다면 UNIQUE 제약 조건을 생성합니다. 예를 들어, 주민등록번호 컬럼이 PRIMARY KEY가 아닌 경우에 해당 테이블에서는 주민등록번호 컬럼이 항상 고유하다면 UNIQUE 제약 조건을 추가하여 중복값이 저장되지 않도록 합니다. UNIQUE 제약 조건도 Clustered Index와 Nonclustered Index 모두 사용 가능하므로, PRIMARY KEY에서 어떤 인덱스를 사용하는지 확인하고 해당 테이블을 참조하는 쿼리를 종합적으로 분석한 다음에 결정합니다. 지면의 제약으로 이 책에는 인덱스 튜닝에 대한 내용은 포함되어 있지 않습니다. 인덱스 튜닝에 대한 내용은 웹 사이트에 게재된 아티클과 SQL Server 2000 온라인 설명서를 참조하십시오.

[따라하기] UNIQUE 제약 조건 선언하기

```
CREATE TABLE Emp (
    EmpId       int          NOT NULL PRIMARY KEY Clustered,
    SSN         char(13)     NOT NULL UNIQUE Nonclustered,
    EmpName     varchar(50)  NOT NULL
)
GO
```

FOREIGN KEY 제약 조건

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

CHECK 제약 조건

CHECK 제약 조건을 사용하면, 컬럼에 저장되는 값이나 포맷을 제한할 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.

동일한 테이블 내의 여러 컬럼에 대해서도 CHECK 제약 조건 설정이 가능합니다. 예를 들어, 어떤 테이블에 MaxTemp(최고온도)와 MinTemp(최저온도)의 두 컬럼이 있을 때 항상 MaxTemp의 값이 MinTemp의 값보다 크도록 보장해야 한다면 컬럼 레벨이 아닌 테이블 레벨에 CHECK 제약 조건을 추가하면 됩니다. 만약 트리거로 무결성을 보장하고자 하는 경우가 발생하면 먼저 CHECK 제약 조건으로 구현 가능한지 점검한 다음에 CHECK 제약 조건으로 불가한 경우에 트리거를 사용하기 바랍니다.

[따라하기] CHECK 제약 조건 선언하기

```
CREATE TABLE Jobs
(
    Job_id      smallint IDENTITY(1,1) PRIMARY KEY CLUSTERED,
    Reg_date    smalldatetime    NOT NULL    DEFAULT (getdate()),
    min_lvl     tinyint          NOT NULL,
    max_lvl     tinyint          NOT NULL,
CONSTRAINT CK_Min_Max CHECK (min_lvl < max_lvl)
)
GO
```

DEFAULT 제약 조건

사용자가 컬럼에 저장되는 값을 명시적으로 지정하지 않은 경우에 디폴트로 어떤 값이 컬럼에 저장되도록 해 주는 기능입니다.

[권고사항]
제약 조건별로 명명 규칙을 정하고 규칙에 의거하여 이름을 부여할 것을 권고합니다. 제약 조건 외에도 모든 사용자 오브젝트들에 대해서는 표준화된 명명 규칙을 수립하고 그 규칙에 의거하여 오브젝트의 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 제약 조건	DF_	DF_CheckDate

수칙4. 항상 값이 저장되는 컬럼에 대해서는 반드시 NOT NULL로 정의합니다.
NULL이라는 값은 알 수 없는 값이라는 의미를 가지는 특수한 값입니다. NULL은 공백 문자나 0, 빈 문자열과는 전혀 다른 알 수 없는 값입니다. 항상 값이 저장되어야 하는 컬럼을 NULL 허용으로 정의하면 응용 프로그램의 오류로 NULL 값이 저장될 수 있으며, 그로 인하여 NULL 데이터로 인하여 논리적 비교가 더욱 복잡해지거나 오류 데이터로 인한 프로그램의 오동작을 유발할 수 있습니다. 그러므로 항상 명시적으로 값이 저장되는 컬럼에 대해서는 반드시 NOT NULL을 지정하기 바랍니다.

[참고]
Identity 컬럼은 tinyint, smallint, int, bigint, decimal(p,0) 또는 numeric(p,0) 컬럼에 할당될 수 있습니다. Identity 컬럼은 자동으로 값이 증가 또는 감소하는 속성을 가지고 있으므로 overflow 또는 underflow가 발생하지 않도록 주기적으로 데이터 타입을 점검합니다. 다음은 Identity 컬럼 목록을 추출하는 예제 스크립트입니다.

```
SELECT object_name(c.id) AS TableName, c.name AS Identity_ColumnName
, CASE WHEN t.name IN ('decimal', 'numeric') THEN t.name +
'('+ CAST(c.xprec AS varchar(5)) + ',' + CAST(c.xscale AS varchar(5)) +
')' ELSE t.name END  AS DataType
FROM sysobjects o JOIN syscolumns c ON o.id = c.id
JOIN master..systypes t ON c.xtype = t.xtype
WHERE o.type = 'U' AND c.colstat & 1 = 1
ORDER BY o.name
GO
```

테이블 삭제하기

■ 구문 : DROP TABLE *table_name*

테이블을 삭제하면, 테이블과 해당 데이터 및 인덱스가 삭제됩니다. Foreign key 제약 조건에 의해 참조되는 테이블은 삭제할 수 없으며, 이 경우에는 참조하는 Foreign key 제약 조건을 삭제한 후 테이블을 삭제해야 합니다. 삭제된 테이블을 참조하는 뷰나 저장 프로시저는 DROP VIEW나 DROP PROCEDURE를 사용하여 삭제합니다.

[따라하기] 다른 데이터베이스에 존재하는 테이블 삭제하기

Sample 데이터베이스에 있는 T1 테이블을 삭제합니다.

```
USE Northwind
GO
DROP TABLE Sample.dbo.T1
GO
```

테이블 변경하기

■ 컬럼 추가하기

테이블에 새로운 컬럼을 추가하는 경우에 NOT NULL 속성 컬럼을 추가할 수는 있지만, 이 경우에는 반드시 DEFAULT를 지정해야 합니다. NOT NULL 속성으로 컬럼을 추가하면 Sch-M Lock으로 인한 블로킹 문제가 발생할 수 있으므로 테이블의 크기가 큰 경우에는 사전에 테스트 서버에서 소요 시간을 확인하고 서비스 휴지 시간을 충분하게 확보한 다음에 작업할 것을 권고합니다. 당장 DEFAULT를 지정할 수 없는 경우에는 일단 NULL 속성으로 컬럼을 추가하고 NULL인 데이터들을 NULL이 아닌 값으로 업데이트한 다음에, 컬럼을 NOT NULL 속성으로 변경하면 됩니다.

[따라하기]

정해진 시간 내에 작업을 끝내야 하는 경우, 테이블 크기가 큰 테이블에 컬럼을 NOT NULL로 추가하였는데, 정해진 시간 내에 ALTER TABLE의 수행이 완료되지 않아서 장애로 이어지는 경우가 간혹 있습니다. 테스트 데이터베이스에서 소요시간을 미리 확인한 결과 소요시간이 제한된 휴지 시간을 초과한다면 다음의 팁을 활용해 보기 바랍니다. 미리 인덱스를 만들어 두면 서비스 중에 쪼개어 업데이트하면 블로킹 발생을 줄일 수 있습니다.

LargeTabAddNotNullCol 테이블에 데이터 타입이 tinyint NOT NULL DEFAULT (0) 속성을 가진 NotNullCol 컬럼을 추가한다는 가정 하에 작성된 예제 스크립트입니다. 다음의 스크립트를 자신의 시스템에 적합하도록 수정하여 활용하기 바랍니다.

```
USE Sample
GO
-- 테스트 테이블을 생성합니다.
SELECT IDENTITY(int, 1,1) AS SeqNo, o1.*
INTO LargeTabAddNotNullCol
FROM Northwind..Orders o1 CROSS JOIN Northwind..Orders o2
GO
```

```
-- 일단 NULL 허용 컬럼을 추가합니다.
ALTER TABLE LargeTabAddNotNullCol
ADD NotNullCol tinyint NULL DEFAULT (0)
GO
-- 업데이트 성능을 위하여 서비스 휴지 시간에 인덱스를 추가합니다.
CREATE INDEX IDX_NotNullCol ON LargeTabAddNotNullCol (NotNullCol)
GO
-- 전체 데이터를 한번에 업데이트하지 말고 분할하여 업데이트합니다.
SET ROWCOUNT 1000
DECLARE @UpdatedRows smallint SET @UpdatedRows = 1000
WHILE @UpdatedRows = 1000
BEGIN
    UPDATE LargeTabAddNotNullCol SET NotNullCol = 0 WHERE NotNullCol
    IS NULL
    SET @UpdatedRows = @@ROWCOUNT
END
SET ROWCOUNT 0
GO
-- NULL인 데이터가 없는지 확인합니다.
SELECT count(*) FROM LargeTabAddNotNullCol WHERE NotNullCol IS NULL
GO
-- NOT NULL로 변경합니다.
ALTER TABLE LargeTabAddNotNullCol
ALTER COLUMN NotNullCol tinyint NOT NULL
GO
-- 인덱스가 필요하지 않다면 다음 서비스 휴지 시간에 인덱스를 삭제합니다.
DROP INDEX LargeTabAddNotNullCol.IDX_NotNullCol
GO
```

■ 컬럼 삭제하기

[따라하기]

LargeTabAddNotNullCol 테이블에 DEFAULT를 설정한 Addcol 컬럼을 추가 한 다음에, 다시 그 컬럼을 삭제하는 예제입니다. 제약 조건이 설정된 컬럼은 제약 조건을 삭제한 후, 컬럼을 삭제합니다.

```
USE Sample
GO
ALTER TABLE LargeTabAddNotNullCol
ADD AddDate smalldatetime NULL
CONSTRAINT DF_AddDate DEFAULT getdate() WITH VALUES
GO
ALTER TABLE LargeTabAddNotNullCol
DROP CONSTRAINT DF_AddDate
GO
ALTER TABLE LargeTabAddNotNullCol
DROP COLUMN AddDate
GO
```

■ 컬럼 변경하기

[따라하기] 컬럼의 데이터 타입을 nchar(10)에서 char(10)으로 변경하기

```
USE Northwind
GO
SELECT * INTO OrdersTest FROM Orders
GO
EXEC sp_columns OrdersTest
GO
ALTER TABLE OrdersTest ALTER COLUMN CustomerID char(10)
GO
```

오브젝트 이름 변경하기

■ 테이블 이름 변경하기

[따라하기]

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

■ 테이블의 인덱스 이름 변경하기

[따라하기]

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

[용도]

일반적으로 인덱스의 이름에 컬럼의 이름을 포함시키므로, 컬럼의 이름을 변경한 다음에 인덱스의 이름까지 변경하고자 하는 경우에 사용할 수 있습니다. 튜닝 컨설팅 경험에 의하면, 쿼리에서 강제로 어떤 인덱스를 사용하도록 인덱스 힌트를 사용한 경우에는 인덱스의 이름에 HINT 접두어를 추가해서 DBA가 임의로 인덱스를 변경하지 않도록 경고하는 것이 필요하다고 생각합니다. 사전에 명명 규칙을 정해서 통일된 이름체계를 세우는 것이 좋습니다. 이것을 위해, 인덱스의 이름을 규칙에 맞추어 통일시키고자 하는 경우에 유용합니다.

■ 제약 조건 이름 변경하기

[따라하기]

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

[용도]

sp_rename을 사용하여 Customers 테이블을 Customers_Old로 테이블 이름을 변경한 후에, Customers 테이블을 새로 만든다고 가정합니다. 데이터베이스내에서 인덱스의 이름은 중복 가능하지만, Primary Key 제약 조건의 이름은 고유해야 합니다. 이런 경우에 Customers_Old 테이블의 Primary Key 제약 조건의 이름을 PK_Customers에서 PK_Customers_Old로 변경하면, 새로 만드는 Customers 테이블에 PK_Customers라는 이름의 제약조건을 만들 수 있습니다.

■ 저장 프로시저, 뷰, 트리거 이름 변경하기

[따라하기]

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

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

[주의 사항]
저장 프로시저, 뷰 또는 트리거의 이름을 변경하더라도 syscomments 테이블에 저장되어 있는 해당 개체의 이름은 변경되지 않습니다. 그러므로 개체의 스크립트를 생성하면 syscomments 테이블의 변경 전 이름이 CREATE 문으로 삽입되므로 문제가 발생할 수 있습니다. 이러한 개체 유형(저장 프로시저나 뷰)은 이름을 바꾸지 않는 것이 좋으며, 개체를 삭제한 다음에 새로운 이름으로 다시 만드는 것을 권고합니다. 그리고 이렇게 개체를 새로 만드는 경우에는 반드시 권한 설정을 이전과 동일하게 적용해야 애플리케이션에서 권한 문제로 실행이 실패하는 문제를 방지할 수 있습니다.

■ 사용자 정의 데이터 형식의 이름 변경하기

[따라하기]

```
EXEC sp_addtype 'UT_CustID', 'nchar(5)', 'NOT NULL'
GO
EXEC sp_rename 'UT_CustID', 'UT_CustomerID', 'USERDATATYPE'
GO
```

테이블 소유자 변경하기

[따라하기]
소유권 체인이 끊어지는 문제를 방지하기 위하여 오브젝트의 소유자를 소유자(owner)가 dbo가 아닌 오브젝트들의 소유자를 dbo로 변경하는 예제입니다.

1. 테스트를 위하여 소유자가 dbo가 아닌 테이블을 만듭니다.

```
EXEC sp_addlogin testuser, testuser, pubs
GO
USE pubs
EXEC sp_adduser testuser, testuser, db_ddladmin
GO
SETUSER 'testuser'
GO
CREATE TABLE IncorrectOwner (c1 int)
GO
/* IncorrectOwner 테이블의 소유자는 dbo가 아닌 testuser가 됩니다. */
SETUSER
GO
```

2. Owner가 dbo가 아닌 사용자 테이블 목록 확인하기

```
SELECT name FROM sysobjects WHERE type='U' AND uid 〈〉 1
ORDER BY name
GO
-- Owner 변경 스크립트 생성하기
SELECT 'EXEC sp_changeobjectowner ''' + USER_NAME(uid) + '.' + name +
''', ''dbo''' FROM sysobjects WHERE type='U' AND uid 〈〉 1
ORDER BY name
GO
```

3. 소유권 변경

```
EXEC sp_changeobjectowner 'testuser.IncorrectOwner', 'dbo'
GO
/* 결과 창의 메시지
주의: 개체 이름 부분을 변경하면 스크립트나 저장 프로시저를 손상시킬 수 있습니다.
*/
```

테이블 정보 확인하기

■ Foreign key 제약 조건 정보 확인하기

[따라하기]

Customers 테이블을 Foreign key로 참조하고 있는 테이블과 컬럼 등의 기본 정보를 반환합니다.

```
USE Northwind
GO
EXEC sp_fkeys N'Customers'
GO
```

■ 테이블의 컬럼 Privilege 정보 확인하기

[따라하기]

Employees 테이블의 각 컬럼의 INSERT, UPDATE, DELETE, REFERENCES등의 Permission 정보를 반환합니다.

```
USE Northwind
GO
EXEC sp_column_privileges Employees
GO
```

■ 테이블의 인덱스 정보 확인하기

[따라하기]

Employees 테이블의 인덱스 목록을 반환합니다.

```
USE Northwind
GO
EXEC sp_helpindex Employees
GO
```

■ 테이블의 제약 조건 정보 확인하기

[따라하기]

Employees 테이블에 관련된 모든 제약 조건의 정보를 반환합니다.

```
USE Northwind
GO
EXEC sp_helpconstraint Employees
GO
```

■ 테이블의 모든 정보 확인하기

[따라하기]

Employees 테이블에 관련된 컬럼, 인덱스, 제약 조건 등의 정보를 반환합니다.

```
USE Northwind
GO
EXEC sp_help Employees
GO
```

■ 테이블이 사용하는 공간 확인하기

[구문]

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

[따라하기]

데이터베이스 내의 모든 테이블의 사용 공간 확인하기

```
/* 방법1. 기존의 시스템 SP를 단순히 활용한 예제 */
EXEC sp_MSforeachtable 'EXEC sp_spaceused [?], ''TRUE'''
GO
/* 방법2. 기존의 시스템 SP를 활용하여 결과를 테이블에 저장한 예제 */
```

```
USE DBAdmin
GO
CREATE TABLE spaceused_pubs (
TableName       sysname,
Rows            int,
Reserved        varchar(20),
Data            varchar(20),
Index_size      varchar(20),
Unused          varchar(20))
GO
USE pubs
GO
INSERT INTO DBAdmin..spaceused_pubs
EXEC sp_MSforeachtable 'EXEC sp_spaceused [?], ''TRUE'''
GO
SELECT * FROM DBAdmin..spaceused_pubs
GO
/* 방법3. sp_spaceused의 소스 코드를 수정하여 사용하기 */
CREATE PROCEDURE sp_spaceused_all
AS
/*
내용 : 데이터베이스내의 모든 테이블의 크기
계산 근거 :
    reserved: sum(reserved) where indid in (0, 1, 255)
    data: sum(dpages) where indid < 2 + sum(used) where indid = 255 (text)
    indexsize: sum(used) where indid in (0, 1, 255) - data
    unused: sum(reserved) - sum(used) where indid in (0, 1, 255)
*/
SET NOCOUNT ON
DECLARE @Low bigint
```

```
SELECT @Low = low FROM mASter.dbo.spt_values
WHERE number = 1 and type = 'E'
SELECT OBJECT_NAME(tmp.id) AS Name
      ,convert(char(11),sum(rows)) AS Rows
      ,ltrim(str(sum(tmp.reserved) * @Low / 1024.,15,0) + ' ' + 'KB') AS Reserved
      ,ltrim(str(sum(tmp.data) * @Low / 1024.,15,0) + ' ' + 'KB') AS Data
      ,ltrim(str((sum(tmp.used) - sum(tmp.data)) * @Low / 1024.,15,0) + ' '
      + 'KB') AS
Index_Size
      ,ltrim(str((sum(tmp.reserved) - sum(tmp.used)) * @Low / 1024.,15,0) +
      ' ' + 'KB') AS Unused
FROM (
      SELECT obj.id AS id, sum(ind.rows) AS rows, sum(ind.reserved)
      AS reserved, sum(dpages) AS data, isnull(sum(used), 0) AS used
      FROM sysindexes ind JOIN sysobjects obj ON ind.id = obj.id
      WHERE obj.xtype='U' AND ind.indid < 2
      GROUP BY obj.id
      UNION
      SELECT obj.id AS id, 0 AS rows, sum(ind.reserved) AS reserved,
      isnull(sum(used), 0) AS data, isnull(sum(used), 0) AS used
      FROM sysindexes ind JOIN sysobjects obj ON ind.id = obj.id
      WHERE obj.xtype='U' AND ind.indid = 255
      GROUP BY obj.id ) tmp
GROUP BY tmp.id
ORDER BY sum(tmp.data) desc, sum(tmp.used) desc

SET NOCOUNT OFF
GO

EXEC pubs..sp_spaceused_all
GO
```

```
EXEC Northwind..sp_spaceused_all
GO
EXEC sp_MSforeachDB 'EXEC [?]..sp_spaceused_all'
GO
```

테이블 옵션 설정하기

사용자 정의 테이블의 옵션 값을 설정합니다.

[구문]

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

[따라하기]

A. Orders 테이블에 'text in row' 옵션 설정하기

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

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

B. Orders 테이블에 'pintable' 옵션 설정하기

'pintable' 옵션을 설정하면, 지정한 테이블의 데이터가 메모리에 상주합니다. 테이블 크기가 작고, 자주 사용하는 코드 테이블을 대상으로 사용할 수 있습니다.

```
USE Northwind
GO
EXEC sp_tableoption 'Orders', 'pintable', 'on'
GO
--메모리 고정 테이블 확인
SELECT OBJECTPROPERTY (OBJECT_ID('Orders'), 'TableIsPinned')
GO
```

[참고]

메모리에 테이블의 데이터를 상주시키기 위해 DBCC PINTABLE을 사용할 수도 있습니다.

```
-- 메모리 고정
DECLARE @objid int, @dbid int
SELECT @dbid = DB_ID('Northwind'), @objid = OBJECT_ID('Northwind..Orders')
DBCC PINTABLE (@dbid, @objid)
GO
-- 메모리 고정 해제
DECLARE @objid int, @dbid int
SELECT @dbid = DB_ID('Northwind'), @objid = OBJECT_ID('Northwind..Orders')
DBCC UNPINTABLE (@dbid, @objid)
GO
-- 데이터베이스내의 메모리 고정 테이블의 전체 크기 확인
SELECT sum(i.used * 8) AS [pin table space used (KB)]
FROM sysindexes i JOIN sysobjects o ON i.id = o.id
WHERE o.status & 1048576 <> 0 AND indid < 2
GO
```

[주의]

이 기능은 성능을 향상시킬 수 있지만 주의해서 사용해야 합니다. 커다란 테이블을 고정할 경우 많은 용량의 버퍼 캐시를 사용하기 때문에 다른 테이블에서 사용할 캐시가 부족하게 됩니다. 버퍼 캐시보다 용량이 큰 테이블을 고정하면 전체 버퍼 캐시를 채울 수도 있습니다. 이런 경우 sysadmin 고정 서버 역할의 구성원이 SQL Server를 중지한 후 다시 시작한 다음, 테이블을 고정 해제해야 합니다. 너무 많은 테이블을 메모리에 고정해도 이와 같은 문제가 발생할 수 있습니다.

시스템 오브젝트 생성

시스템 저장 프로시저 생성하기

시스템 저장 프로시저는 master 데이터베이스에 있으며 이름이 sp_ 라는 접두사로 시작하며, 모든 데이터베이스에서 master 데이터베이스라고 지정하지 않고 저장 프로시저의 이름 만으로 실행이 가능합니다. 그리고 master 데이터베이스가 아닌 데이터베이스에서 실행하면 그 데이터베이스의 컨텍스트 내에서 수행되는 특징을 가지고 있습니다. 예를 들어 저장 프로시저가 sysobjects 테이블을 참조한다고 가정하면 이 프로시저는 실제로는 master에 있음에도 불구하고 그 프로시저를 실행할 때 연결되어 있던 데이터베이스에 있는 sysobjects를 액세스합니다.

[따라하기] 시스템 저장 프로시저 생성하기

1. master 데이터베이스에 연결합니다.
2. sp_ 로 시작하는 이름으로 저장 프로시저를 생성합니다.

시스템 함수 조회 및 생성하기

SQL Server가 설치되는 동안 시스템 UDF(사용자 정의 함수)들이 생성되며, 이 시스템 함수는 모든 데이터베이스에서 함수의 이름만 사용하여 액세스가 가능합니다.
시스템 뷰를 만드는 것과 유사하게 master 데이터베이스에 존재하지만 모든 데이터베이스에서 데이터베이스를 지정하지 않고 쿼리를 수행할 수 있는 함수를 사용자가 직접 생성할 수 있습니다. 시스템 함수가 되기 위해서는 함수를 생성할 때 소유자를 system_function_schema로 생성합니다.

[따라하기] 시스템 함수 생성, 조회하기

```
/* 시스템 함수 생성하기 */
-- 시스템 테이블을 직접 수정할 수 있도록 설정합니다.
EXEC sp_configure 'allow updates',1
RECONFIGURE WITH OVERRIDE
GO
-- 함수는 master 데이터베이스에 생성하고,
-- 소유자는 system_function_schema로 지정합니다.
USE master
GO
CREATE FUNCTION system_function_schema.fn_greatest (@x bigint, @y bigint)
RETURNS bigint
AS
BEGIN
        RETURN(CASE WHEN @x> @y THEN @x ELSE @y END)
END
GO
CREATE FUNCTION system_function_schema.fn_least (@x bigint, @y bigint)
RETURNS bigint
AS
BEGIN
        RETURN(CASE WHEN @x<@y THEN @x ELSE @y END)
END
GO
-- 시스템 테이블을 직접 수정할 수 없도록 0으로 변경합니다. (반드시 수행 요망)
EXEC sp_configure 'allow updates',0
RECONFIGURE WITH OVERRIDE
GO
-- 생성한 시스템 함수는 모든 데이터베이스에서 호출 가능합니다.
```

```
USE Northwind
GO
SELECT fn_greatest(989, 998), fn_least(989, 998)
GO
/* 시스템 함수 조회하기 */
USE master
GO
SELECT name FROM sysobjects
WHERE uid=USER_ID('system_function_schema')
AND   (OBJECTPROPERTY(id, 'IsScalarFunction')=1
       OR OBJECTPROPERTY(id, 'IsTableFunction')=1
       OR OBJECTPROPERTY(id, 'IsInlineFunction')=1)
GO
```

INFORMATION 스키마 뷰 생성하기

SQL Server 2000에서 메타데이터 정보를 가져오는 방법은 시스템 저장 프로시저를 사용하는 방법과 INFORMATION 스키마 뷰를 사용하는 방법 등이 있습니다.

[참고]
메타데이터를 가져오기 위해서는 시스템 저장 프로시저, 시스템 함수 또는 시스템 제공 뷰를 사용할 것을 권고합니다. 시스템 테이블을 직접 쿼리하는 경우, 시스템 테이블이 이후 버전에서 변경될 때 정확한 정보를 제공하지 못할 수도 있습니다. 정보 스키마 뷰를 참조할 때는 다음과 같이 사용자 이름을 지정하는 위치에 INFORMATION_SCHEMA를 지정해야 합니다.

```
SELECT *
FROM Northwind.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = N'Customers'

[SQL Server 2000에서 제공하는 INFORMATION 스키마 뷰 목록]
CHECK_CONSTRAINTS
COLUMN_DOMAIN_USAGE
COLUMN_PRIVILEGES
COLUMNS
CONSTRAINT_COLUMN_USAGE
CONSTRAINT_TABLE_USAGE
DOMAIN_CONSTRAINTS
DOMAINS
KEY_COLUMN_USAGE
PARAMETERS
REFERENTIAL_CONSTRAINTS
ROUTINES
ROUTINE_COLUMNS
SCHEMATA
TABLE_CONSTRAINTS
TABLE_PRIVILEGES
TABLES
VIEW_COLUMN_USAGE
VIEW_TABLE_USAGE
VIEWS
```

[따라하기]

CHECK 제약 조건에 관한 정보를 제공해 주는 INFORMATION 스키마 뷰는 있지만 DEFAULT 제약 조건에 관한 정보를 제공해 주는 INFORMATION 스키마 뷰는 기본적으로 제공되지 않습니다. 이 예제에서는 기존의 CHECK_CONSTRAINTS라는 INFORMATION 스키마 뷰의 소스를 수정하여 DEFAULT 제약 조건에 관한 정보를 제공해 주는 INFORMATION 스키마 뷰를 만들어 보고자 합니다.

```
-- 기본으로 제공되는 INFORMATION 스키마 뷰 활용하기
USE Pubs
GO
SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS
GO
-- 사용자가 INFORMATION 스키마 뷰 생성하기
USE master
GO
EXEC sp_configure 'allow updates', 1
RECONFIGURE WITH OVERRIDE
GO
EXEC sp_MS_upd_sysobj_category 1
GO
CREATE VIEW INFORMATION_SCHEMA.DEFAULT_CONSTRAINTS
AS
SELECT db_name()    as CONSTRAINT_CATALOG
      ,user_name(c_obj.uid) as CONSTRAINT_SCHEMA
      ,c_obj.name    as CONSTRAINT_NAME
      ,com.text    as DEFAULT_CLAUSE
FROM sysobjects c_obj, syscomments com
WHERE c_obj.uid = user_id()
      AND c_obj.id = com.id
      AND c_obj.xtype = 'D'
GO
EXEC sp_MS_upd_sysobj_category 2
GO
EXEC sp_configure 'allow updates', 0
GO
RECONFIGURE WITH OVERRIDE
GO
```

```
-- 생성한 INFORMATION 스키마 뷰 활용하기
USE pubs
GO
SELECT *
FROM INFORMATION_SCHEMA.DEFAULT_CONSTRAINTS
GO
```

사용자 관리

로그인과 사용자 이해하기

SQL Server에 접근하기 위해서는 로그인 계정이 필요하고, 데이터베이스에 접근하기 위해서는 사용자(user)를 추가해 주어야 합니다.

권한 이해하기

사용자는 권한을 부여 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 이해하기

역할(Role)은 SQL Server 로그인 또는 데이터베이스 사용자들의 집합이며, 사용 권한을 적용할 수 있는 단일 단위입니다. 사용자별로 개별적으로 권한을 관리하지 말고, 권한의 수준이 다른 몇 개의 역할을 미리 만들고 역할에 사용자를 추가함으로써 권한을 관리할 것을 권고합니다. 시스템에서 미리 정의된 역할에는 서버 로그인 계정에게 사용할 수 있는 서버 역할과 데이터베이스 사용자에게 사용할 수 있는 데이터베이스 역할이 있습니다. 시스템에서 미리 정의된 역할을 활용하면 편리합니다.

로그인 계정 생성하고 권한 부여하기

[따라하기]

1. SQL Server 계정을 생성하여, Sample 데이터베이스에 사용자로 등록하고, 데이터베이스 내의 테이블들을 읽을 수 있고 데이터의 추가, 변경, 삭제가 가능하도록 권한을 부여하는 예제입니다. 고정 데이터베이스 역할 db_datareader는 데이터베이스 내의 모든 사용자 테이블들을 읽을 수 있으며 db_datawriter는 데이터베이스 내의 모든 사용자 테이블에 대하여 추가, 변경, 삭제 권한을 가지고 있습니다. SQL Server 연결 시 디폴트로 sample 데이터베이스에 연결됩니다.

```
EXEC sp_addlogin 'DevUser', 'ad1234', 'sample'
GO
USE Sample
GO
EXEC sp_adduser 'DevUser', 'DevUser', 'db_datareader'
GO
EXEC sp_addrolemember 'db_datawriter', 'DevUser'
GO
```

2. NT 로그인 계정을 sample 데이터베이스에 사용자로 등록하고, 데이터베이스내의 테이블들을 읽는 권한을 설정하는 예제입니다.
   NT 로그인 계정은 윈도우의 [내 컴퓨터 관리]나 [액티브 디렉터리 사용자 및 컴퓨터]를 이용하여 생성합니다. 생성된 계정은 Admin\winadmin이라고 가정합니다.

```
EXEC sp_grantlogin 'Admin\winadmin'
GO
EXEC sp_defaultdb 'Admin\winadmin', 'sample'
GO
USE Sample
GO
EXEC sp_grantdbaccess 'Admin\winadmin', 'winadmin'
GO
EXEC sp_addrolemember 'db_datareader', 'winadmin'
GO
```

3. SQL Server 로그인 이름은 'BackupAdmin', 암호는 '1234'로 설정하여 sample 데이터베이스의 사용자로 등록하고, 데이터베이스 백업을 수행할 수 있는 권한을 부여하는 예제입니다. 그런 다음에 BackupAdmin 로그인 계정을 삭제합니다. 고정 데이터베이스 역할 중에서 db_backupoperator 역할은 데이터베이스 백업을 수행할 수 있는 권한을 가지고 있습니다.

```
EXEC sp_addlogin 'BackupAdmin', '1234', 'sample'
GO
USE Sample
GO
EXEC sp_adduser 'BackupAdmin', 'BackupAdmin', 'db_backupoperator'
GO
-- 로그인 계정 삭제
EXEC sp_droprolemember 'db_backupoperator', 'BackupAdmin'
GO
EXEC sp_revokedbaccess 'BackupAdmin'
GO
EXEC sp_droplogin 'BackupAdmin'
GO
```

4. NETDOMAIN의 John, Sarah, Diane 계정을 courses 데이터베이스의 사용자로 등록하고, Student 역할에 등록하는 예제입니다. 그 후, Student 역할에서 Diane 계정을 삭제하고, 로그인 계정에서도 Diane을 삭제합니다.

```
USE master
GO
EXEC sp_grantlogin 'NETDOMAIN\John'
GO
EXEC sp_defaultdb 'NETDOMAIN\John', 'courses'
GO
EXEC sp_grantlogin 'NETDOMAIN\Sarah'
GO
EXEC sp_defaultdb 'NETDOMAIN\Sarah', 'courses'
```

```
GO
EXEC sp_grantlogin 'NETDOMAIN\Diane'
GO
EXEC sp_defaultdb 'NETDOMAIN\Diane', 'courses'
GO
USE courses
GO
EXEC sp_grantdbaccess 'NETDOMAIN\John'
GO
EXEC sp_grantdbaccess 'NETDOMAIN\Sarah'
GO
EXEC sp_grantdbaccess 'NETDOMAIN\Diane'
GO
EXEC sp_addrole 'Student'
GO
EXEC sp_addrolemember 'Student','NETDOMAIN\John'
GO
EXEC sp_addrolemember 'Student','NETDOMAIN\Sarah'
GO
EXEC sp_addrolemember 'Student','NETDOMAIN\Diane'
GO

EXEC sp_droprolemember 'Student','NETDOMAIN\Diane'
GO

EXEC sp_revokedbaccess 'NETDOMAIN\Diane'
GO
EXEC sp_revokelogin 'NETDOMAIN\Diane'
GO
```

[참고]
A. sp_revokelogin
sp_grantlogin 또는 sp_denylogin으로 만든 NT 사용자 또는 로그인 항목을 SQL Server에서 제거합니다. NT 로그인 계정이 SQL Server에 연결하는 것이 금지됩니다.

B. sp_denylogin
NT 로그인 계정이 SQL Server에 연결하는 것을 금지합니다.

기존 로그인과 사용자에 관한 정보 확인하기

[따라하기]

```
EXEC sp_helplogins
GO
USE sample
GO
EXEC sp_helpuser
GO
```

암호 변경하기

[따라하기] 암호 변경하기

로그인 계정 'DevUser'의 암호 'ad1234'를 'ad5678'로 변경합니다.

```
EXEC sp_password 'ad1234','ad5678', 'DevUser'
GO
```

서버 및 데이터베이스의 정보 확인

서버 이름, 버전, Edition 확인하기

[구문] SERVERPROPERTY (*propertyname*)

[따라하기]

```
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
```

서버의 옵션 확인하기

서버의 구성 옵션과 구성 옵션의 최소값과 최대값, 설정된 대상 값, 설정값을 확인합니다.

[구문]

```
sp_configure [ [ @configname = ] 'name' ]
[ , [ @configvalue = ] 'value' ]
```

[따라하기]

1. 서버 구성 옵션 확인하기

```
EXEC sp_configure
GO
```

2. 서버의 고급 구성 옵션 확인하기

서버의 일부 옵션은 고급으로 지정되어 있고, 이 값들도 변경할 수 있습니다. 고급 구성 옵션을 보기 위해서는 먼저 다음을 실행합니다.

```
USE master
GO
EXEC sp_configure 'show advanced option', '1'
RECONFIGURE WITH OVERRIDE
GO
```

데이터베이스 파일에 대한 I/O 통계 정보 확인하기

fn_virtualfilestats 함수는 시스템에서 기본으로 제공하는 함수로서, I/O에 대한 통계 정보를 제공합니다. 사용자들이 어떤 파일에 대하여 읽기나 쓰기를 수행하기 위하여 기다린 시간을 제공하므로 이 함수를 사용하면 어떤 파일들에 대하여 I/O가 많이 발생하는지 확인할 수 있습니다. IO로 인한 성능 저하가 의심되는 경우에는 fn_virtualfilestats 함수가 반환하는 IOStallMS를 점검할 것을 권고합니다.

[구문]

```
fn_virtualfilestats ( [@DatabaseID=] database_id
, [ @FileID = ] file_id )
```

[따라하기]

1. 모든 데이터베이스의 파일들의 I/O 정보 확인하기

```
SELECT * FROM ::fn_virtualfilestats(-1, -1)
GO
```

2. 모든 데이터베이스의 파일들의 I/O 정보 확인하기 (시스템 함수 활용)

```
SELECT DB_NAME(Dbid) AS DBName
, FileId
, FILE_NAME(FileId) AS FileName
, IoStallMS, NumberReads, NumberWrites, BytesRead
, BytesWritten
FROM ::fn_VirtualFileStats (-1, -1)
ORDER BY IoStallMS DESC
GO
```

3. 특정 데이터베이스의 모든 파일들에 대한 I/O 정보 확인하기 (예제: tempdb)

```
SELECT * FROM ::fn_virtualfilestats(2, -1)
GO
```

4. 특정 데이터베이스의 특정 파일의 I/O 정보 확인하기 (Tempdb의 Primary Data File)

```
SELECT * FROM ::fn_virtualfilestats(2, 1)
GO
```

5. 특정 데이터베이스의 특정 파일의 I/O 정보 확인하기 (Tempdb의 로그 파일)

```
SELECT * FROM ::fn_virtualfilestats(2, 2)
GO
```

기본적인 파일 정보 확인하기

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]

```
USE Sample
EXEC sp_helpfile
-- 또는
EXEC Sample..sp_helpfile
```

기본적인 파일 그룹 정보 확인하기

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 데이터베이스의 정보 확인하기

지정된 데이터베이스 또는 모든 데이터베이스 정보를 반환합니다.

[구문]

```
sp_helpdb [ [ @dbname= ] 'name' ]
```

■ 데이터베이스를 지정하지 않았을 경우, 결과 집합입니다.

컬럼명	내용
Name	데이터베이스 이름
Db_size	데이터베이스의 총 크기
Owner	데이터베이스의 소유자
Dbid	데이터베이스의 ID
Created	데이터베이스의 생성일자
Status	옵션의 값을 쉼표로 분리하여 나열한 것
Compatibility_level	호환성 수준(60,65,70,80)

■ 데이터베이스를 지정했을 경우, 위의 결과 집합에 다음의 결과 집합이 추가로 나타납니다.

Name	논리적 파일 이름
Fileid	파일 ID
Filename	파일의 물리적 경로와 이름
Filegroup	파일이 속한 그룹
Size	파일 크기
Maxsize	파일이 증가할 수 있는 최대 크기
Growth	파일의 증가량
Usage	파일의 사용법

데이터베이스 옵션 또는 현재 설정 확인하기

지정한 Property에 따른 결과값이 반환됩니다. property의 값 목록과 그에 따른 결과의 종류는 BOL을 참조해 주십시오.

[구문]

```
DATABASEPROPERTYEX(database_name, property)
```

[따라하기]

1. Northwind 데이터베이스의 통계 자동 업데이트 옵션 설정 확인하기

```
SELECT DATABASEPROPERTYEX ('Northwind','IsAutoUpdateStatistics')
AS IsAutoUpdateStatistics
GO
```

2. Northwind 데이터베이스의 복구 모델 설정 확인하기

```
SELECT DATABASEPROPERTYEX ('Northwind','Recovery') AS Recovery
GO
```

데이터베이스 옵션 설정 확인하기

[구문]

```
sp_dboption [ [ @dbname = ] 'database' ]
[ , [ @optname = ] 'option_name' ]
[ , [ @optvalue = ] 'value' ]
```

[따라하기]

A. 설정 가능한 옵션의 목록 확인하기

인수를 지정하지 않습니다.

```
EXEC sp_dboption
GO
```

B. 지정한 데이터베이스의 설정 옵션 목록 확인하기

데이터베이스 이름만 지정합니다. Northwind데이터베이스의 설정 옵션 목록을 확인합니다.

```
EXEC sp_dboption 'Northwind'
GO
```

C. 지정한 데이터베이스의 해당 옵션의 설정 상태 확인하기

데이터베이스와 해당 옵션을 설정합니다. Northwind데이터베이스의 자동 통계 갱신 옵션의 설정을 확인합니다.

```
EXEC sp_dboption 'Northwind', 'auto update statistics'
GO
```

■ 데이터베이스 옵션의 종류

- Auto create statistics
- Auto update statistics
- Autoclose
- Autoshrink
- ANSI null default
- ANSI nulls
- ANSI padding
- ANSI warnings
- Arithabort
- concat null yields null
- cursor close on commit

- Dbo use only
- default to local cursor
- merge publish
- numeric roundabort
- Offline
- Published
- quoted identifier
- read only
- Recursive triggers
- select into/bulkcopy
- single user
- Subscribed
- Torn page detection
- trunc. Log on chkpt.

Tempdb에 대한 공간 사용 정보 확인하기

[구문]

```
sp_tempdbspace
```

테이블 크기 확인하기

[구문]

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

[따라하기] Orders 테이블의 크기 확인하기

```
EXEC sp_spaceused 'Orders'
```

[참고]
테이블을 지정하지 않으면, 해당 데이터베이스의 크기가 반환됩니다.

로그 공간의 사용 정보 확인하기

서버 내에 존재하는 모든 데이터베이스의 로그 공간의 사용에 관한 통계를 확인합니다.

[구문]

```
DBCC SQLPERF ( LOGSPACE )
```

테이블의 조각화 정보 확인하기

테이블의 조각화는 INSERT, UPDATE, DELETE문 등을 수행하여 데이터를 수정할 때에 발생합니다. 이러한 수정들은 각 페이지의 채움 정도를 다르게 만들고, 테이블의 일부 또는 전부를 스캔하는 쿼리의 경우, 테이블 조각으로 인한 성능 저하가 발생할 가능성이 있습니다.

[구문]

```
DBCC SHOWCONTIG
[    ( { table_name | table_id | view_name | view_id }
       [ , index_name | index_id ]
       )
]
       [ WITH { ALL_INDEXES
       | FAST [ , ALL_INDEXES ]
       | TABLERESULTS [ , { ALL_INDEXES } ]
       [ , { FAST | ALL_LEVELS } ]
       }
]
```

[따라하기]

A. 지정한 테이블의 조각화 정보 확인하기

Orders 테이블의 조각화 정보를 확인합니다.

```
USE Northwind
GO
DBCC SHOWCONTIG ('Orders')
GO
```

B. 지정한 테이블의 지정한 인덱스의 조각화 정보 확인하기

Orders 테이블의 CustomerID 인덱스에 대한 조각화 정보를 확인합니다.

```
USE Northwind
GO
DBCC SHOWCONTIG ('Orders', 'CustomerID')
GO
```

C. 테이블 ID와 인덱스 ID로 조각화 정보 확인하기

테이블 이름과 인덱스 이름 대신, 테이블 ID와 인덱스 ID를 사용합니다.

```
USE Northwind
GO
DECLARE @ObjectId int, @IndexId int
SELECT @ObjectId = OBJECT_ID ('Orders')
SELECT @IndexId = indid FROM sysindexes
WHERE id = @ObjectId
DBCC SHOWCONTIG (@ObjectId, @IndexId)
GO
```

D. 데이터베이스내의 모든 테이블의 모든 인덱스에 대한 조각화 정보 확인하기

```
USE Northwind
GO
DBCC SHOWCONTIG WITH TABLERESULTS, ALL_INDEXES
GO
```

E. 데이터베이스에 수행 결과를 저장하기

```
-- 결과를 저장할 테이블 생성하기
USE DBAdmin
GO
CREATE TABLE ShowContig_Pubs
(
    ObjectName          sysname,
    ObjectId            int,
    IndexName           sysname,
    IndexId             tinyint,
    Level               tinyint,
    Pages               int,
    Rows                bigint,
```

```
        MinimumRecordSize       smallint,
        MaximumRecordSize       smallint,
        AverageRecordSize       smallint,
        ForwardedRecords        bigint,
        Extents                 int,
        ExtentSwitches          numeric(10,2),
        AverageFreeBytes        numeric(10,2),
        AveragePageDensity      numeric(10,2),
        ScanDensity             numeric(10,2),
        BestCount               int,
        ActualCount             int,
        LogicalFragmentation    numeric(10,2),
        ExtentFragmentation     numeric(10,2),
        CheckDate  smalldatetime DEFAULT (GETDATE())
)
GO
-- 시스템 저장 프로시저 생성하기
USE master
GO
CREATE PROCEDURE sp_DBCCSHOWCONTIG
AS
    DBCC SHOWCONTIG WITH TABLERESULTS
GO
-- 실행 예제
USE Pubs
GO
INSERT INTO DBAdmin..ShowContig_Pubs (
    ObjectName, ObjectId, IndexName, IndexId, Level, Pages,
    Rows, MinimumRecordSize, MaximumRecordSize,
    AverageRecordSize, ForwardedRecords, Extents,
```

```
    ExtentSwitches, AverageFreeBytes, AveragePageDensity,
    ScanDensity, BestCount, ActualCount, LogicalFragmentation,
    ExtentFragmentation)
EXEC sp_DBCCSHOWCONTIG
GO
-- 실행 결과 확인하기
SELECT * FROM DBAdmin..ShowContig_Pubs
WHERE ObjectName = 'Authors'
GO
SELECT * FROM DBAdmin..ShowContig_Pubs
WHERE LogicalFragmentation > 30
OR ExtentFragmentation > 30
GO
```

[참고]
WITH FAST 옵션을 사용하면, 인덱스의 잎 또는 데이터 수준 페이지를 읽지 않기 때문에, 생략된 정보를 빠르게 반환합니다.

[주의]
크기가 큰 테이블에 대해서 서비스 중에 DBCC SHOWCONTIG를 수행하면 성능 저하를 유발할 수 있으므로 유의합니다.

[권고사항]
수행 결과를 확인하고 단편화가 심한 테이블들에 대해서는 인덱스 재구성 작업을 수행합니다.

대기 정보 확인하기

각 waittype별 대기 시간을 확인합니다. 가장 대기가 많은 유형을 확인하고, 작업 부하에 따른 대기 유형의 변화도 확인합니다.

[구문]

```
DBCC SQLPERF (WAITSTATS)
```

인덱스 재구성하기

인덱스 조각화로 인한 성능 저하를 방지하기 위해서는 주기적으로 인덱스 조각화가 진행된 테이블들에 대한 조각화 제거 작업이 필요합니다. 가능한 한 인덱스 재구성 작업을 자동화하여 주기적으로 용이하게 수행할 수 있는 체계를 갖출 것을 권고합니다.

[주의]

DBCC DBREINDEX 작업은 서비스 휴지 시간에 수행해야 합니다.

[예제 스크립트]

다음은 인덱스 조각화 제거를 위해 활용할 수 있는 예제 스크립트입니다. 아래 예제 스크립트를 시스템의 환경에 적합하도록 수정 보완하여 활용하기 바랍니다.

```
-- Written by Kimberly L. Tripp - all rights reserved.
-- For more scripts, sample code and Kimberly' s schedule check out
-- www.SQLSkills.com
-- For "More than Just Training," see the Industry Experts at
-- www.SolidQualityLearning.com
-- Disclaimer - Thoroughly test this script, execute at your own risk.
```

```
/*
Execute this whole script to create the sp_RebuildIndexes stored procedure in Master.
Best Viewed with Courier New 12pt. and Tabs saved as 4 spaces not 8. (Tools, Options, Editor)
To use the sp_RebuildIndexes procedure once created use:
sp_RebuildIndexes
To Rebuild All Indexes on All Tables for all that have a Scan Density < 100%
sp_RebuildIndexes @ScanDensity = 80
To Rebuild All Indexes on All Tables with a Scan Density of < 80%
sp_RebuildIndexes 'Authors'
To Rebuild All Indexes on the authors table - for a Scan Density of < 100%
sp_RebuildIndexes 'Authors', 80
To Rebuild All Indexes on the authors table - for a Scan Density of < 80%
Object Name and ScanDensity are both optional parameters.
ScanDensity must be a whole number between 1 and 100.
*/
USE master
GO
IF OBJECTPROPERTY(object_id( 'sp_RebuildClusteredIndex' ), 'IsProcedure') =
1
DROP PROCEDURE sp_RebuildClusteredIndex
GO

IF OBJECTPROPERTY(object_id('sp_RebuildIndexes'), 'IsProcedure') = 1
DROP PROCEDURE sp_RebuildIndexes
GO

CREATE PROCEDURE sp_RebuildClusteredIndex
(
```

```
@TableName      sysname = NULL,
@IndexName      sysname = NULL
)
AS
-- Written by Kimberly L. Tripp of SYSolutions, Inc.
-- For more code samples go to http://www.sqlskills.com
-- NOTE: If your clustered index is NOT unique then rebuilding the clustered
--       index will cause the non-clustered indexes to be rebuilt.
--       If the nonclustered indexes were fragmented
--       then this series of scripts will build them again.
IF @TableName IS NOT NULL
BEGIN
IF (OBJECTPROPERTY(object_id(@TableName), 'IsUserTable') = 0
AND OBJECTPROPERTY(object_id(@TableName), 'IsView') = 0)
BEGIN
RAISERROR('Object: %s exists but is NOT a User-defined Table. This procedure
only accepts valid table names to process for index rebuilds.', 16, 1,
@TableName)
      RETURN
END
ELSE
BEGIN
      IF OBJECTPROPERTY(object_id(@TableName), 'IsTable') IS NULL
      BEGIN
RAISERROR('Object: %s does not exist within this database. Please check the
table name and location (which database?). This procedure only accepts existing
table names to process for index rebuilds.', 16, 1, @TableName)
       RETURN
       END
END
```

```
END

IF @IndexName IS NOT NULL
BEGIN
IF INDEXPROPERTY(object_id(@TableName), @IndexName, 'IsClustered') = 0
BEGIN
RAISERROR('Index: %s exists but is a Clustered Index. This procedure only
accepts valid table names and their clustered indexes for rebuilds.', 16, 1,
@IndexName)
      RETURN
END
ELSE
BEGIN
IF INDEXPROPERTY(object_id(@TableName),
@IndexName, 'IsClustered') IS NULL
  BEGIN
SELECT @TableName, @IndexName,
    INDEXPROPERTY(object_id(@TableName),
@IndexName, 'IsClustered')
RAISERROR('There is no index with name:%s on this table. Please check the
table name and index name as well as location (which database?). This
procedure only accepts existing table names and their clustered indexes for
rebuilds.', 16, 1, @IndexName)
       RETURN
END
END
END

-- So now we have a valid table, a valid CLUSTERED index and we're ready to
rebuild.
```

```
-- Here' s a quick overview of what this code will do:
-- Get the Column List and Index Defintion (Use the output from sp_helpindex)
-- Figure out if it' s UNIQUE - to specify in CREATE INDEX statement
-- Build and Execute the CREATE INDEX command through dynamic string execution

DECLARE @ExecStr                nvarchar(4000)
-- more than enough even if 16 cols of 128 chars,
-- Tablename of 128 and Indexname of 128...
-- but if this is the case you have other problems :).
      , @ColList      nvarchar(3000)
      , @Unique       nvarchar(7)
-- Will be either '' or 'Unique' and added to CR Index String
      , @FillFactor   nvarchar(100)

CREATE TABLE #IndexInfo
(
      IndexName    sysname,
      IndexDesc    varchar(210),
      IndexKeys    nvarchar(2126)
)

INSERT INTO #IndexInfo EXEC sp_helpindex @TableName

SELECT @ColList = IndexKeys
      , @Unique = CASE
        WHEN IndexDesc LIKE  ''clustered, unique%'
        THEN 'Unique'
        ELSE ''
      END --CASE Expression
```

```
        , @FillFactor = ', FILLFACTOR = '+ NULLIF(convert(nvarchar(3)
        , (SELECT OrigFillFactor
          FROM sysindexes
          WHERE id = object_id(@TableName) AND Name = @IndexName)), 0)
FROM #IndexInfo
WHERE IndexName = @IndexName

SELECT @ExecStr = 'CREATE'+ @Unique +'CLUSTERED INDEX'
+ QUOTENAME(@IndexName, ']') + 'ON'
+ QUOTENAME(@TableName, ']') + '('+ @collist
+ ') WITH DROP_EXISTING'+ ISNULL(@FillFactor,'')
-- For testing the String
-- SELECT @ExecStr

-- Create the Clustered Index
EXEC(@ExecStr)
GO

CREATE PROCEDURE sp_RebuildIndexes
(
      @TableName sysname = NULL,
      @ScanDensity                tinyint = 100
)
AS
-- Written by Kimberly L. Tripp of SYSolutions, Inc.
-- For more code samples go to http://www.sqlskills.com
-- This procedure will get the Fragmentation information for all tables and indexes
-- within the database.
-- Programmatically it will then walk the list rebuilding all indexes that have a scan
-- density less than the value passed in - by default any less than 100%
```

```
contiguous.
-- Use this script as a starting point. Modify it for your options, ideas, etc.
-- and then schedule it to run regularly.
-- NOTE - This gathers density information for all tables and all indexes.
-- This might be time consuming on large databases.
-- DISCLAIMER - Execute at your own risk. TEST THIS FIRST.
SET NOCOUNT ON

IF @ScanDensity IS NULL
SET @ScanDensity = 100

IF @ScanDensity NOT BETWEEN 1 AND 100
BEGIN
RAISERROR('Value supplied:%i is not valid. @ScanDensity is a percentage.
Please supply a value for Scan Density between 1 and 100.', 16, 1,
@ScanDensity)
RETURN
END
IF @TableName IS NOT NULL
BEGIN
IF OBJECTPROPERTY(object_id(@TableName),'IsUserTable') = 0
BEGIN
RAISERROR('Object: %s exists but is NOT a User-defined Table. This procedure
only accepts valid table names to process for index rebuilds.', 16, 1,
@TableName)
RETURN
END
ELSE
BEGIN
      IF OBJECTPROPERTY(object_id(@TableName), 'IsTable') IS NULL
```

```
BEGIN
RAISERROR('Object: %s does not exist within this database. Please check the
table name and location (which database?). This procedure only accepts existing
table names to process for index rebuilds.', 16, 1, @TableName)
RETURN
END
END
END

-- Otherwise the Object Exists and it is a table so we' ll continue from here.
-- First thing to do is create a temp location for the data returned from
-- DBCC SHOWCONTIG

CREATE TABLE #ShowContigOutput
(
ObjectName                  sysname,
ObjectId                    int,
IndexName                   sysname,
IndexId                     tinyint,
[Level]                     tinyint,
Pages                       int,
[Rows]                      bigint,
MinimumRecordSize           smallint,
MaximumRecordSize           smallint,
AverageRecordSize           smallint,
ForwardedRecords bigint,
Extents                     int,
ExtentSwitches              numeric(10,2),
AverageFreeBytes  numeric(10,2),
AveragePageDensity          numeric(10,2),
```

```
ScanDensity             numeric(10,2),
BestCount               int,
ActualCount             int,
LogicalFragmentation    numeric(10,2),
ExtentFragmentation     numeric(10,2)
)

IF @TableName IS NOT NULL
-- then we only need the showcontig output for that table
INSERT #ShowContigOutput
EXEC('DBCC SHOWCONTIG ('+ @TableName +') WITH FAST, ALL_INDEXES,
TABLERESULTS')
ELSE
-- All Tables, All Indexes Will be processed.
INSERT #ShowContigOutput
EXEC('DBCC SHOWCONTIG WITH FAST, ALL_INDEXES, TABLERESULTS')
PRINT N' '
-- Quick test to see if everything is getting here correctly
-- SELECT * FROM #ShowContigOutput
-- Walk the showcontig output table skipping all replication tables
-- as well as all tables necessary for
-- the UI. This is also where you can list large tables
-- that you don' t want to rebuild all at one time.
-- NOTE: If you take out a large table from rebuilding this script may have
--    already checked density
--    meaning that the expense in terms of time may have been
--    expensive.
--    Also, you should use a different procedure to rebuild a large table
--    specifically.
--    Even when you pass in the tablename it will be avoided here if
```

```
-- MANUALLY added to the list by you.
-- Test, Test, Test!

DECLARE @ObjectName          sysname,
        @IndexName           sysname,
        @QObjectName         nvarchar(258),
        @QIndexName          nvarchar(258),
        @IndexID             tinyint,
        @ActualScanDensity   numeric(10,2),
        @InformationalOutput nvarchar(4000),
        @StartTime           datetime,
        @EndTime             datetime

DECLARE TableIndexList CURSOR FAST_FORWARD FOR
SELECT ObjectName, IndexName, IndexID, ScanDensity
FROM #ShowContigOutput AS sc JOIN sysobjects AS so ON sc.ObjectID =
so.id
WHERE sc.ScanDensity < @ScanDensity
      AND (OBJECTPROPERTY(sc.ObjectID, 'IsUserTable') = 1
                  OR OBJECTPROPERTY(sc.ObjectID, 'IsView') = 1)
      AND so.STATUS > 0
      AND sc.IndexID BETWEEN 1 AND 250
      AND sc.ObjectName NOT IN ( 'dtproperties' )
-- Here you can list large tables you do not WANT rebuilt.
ORDER BY sc.ObjectName, sc.IndexID

OPEN TableIndexList
FETCH NEXT FROM TableIndexList INTO @ObjectName, @IndexName,
@IndexID, @ActualScanDensity
WHILE (@@fetch_status <> -1)
```

```
BEGIN
IF (@@fetch_status <> -2)
BEGIN
      SELECT @QObjectName = QUOTENAME(@ObjectName, ']')
      SELECT @QIndexName = QUOTENAME(@IndexName, ']')
      SELECT @InformationalOutput = N'Processing Table:'
+ RTRIM(UPPER(@QObjectName))
           + N'Rebuilding Index:'+ RTRIM(UPPER(@QIndexName))
       PRINT @InformationalOutput
       IF @IndexID = 1
       BEGIN
          SELECT @StartTime = getdate()
          EXEC sp_RebuildClusteredIndex @ObjectName, @IndexName
SELECT @EndTime = getdate()
          SELECT @InformationalOutput = N'Total Time to process ='
+ convert(nvarchar, datediff(ms, @StartTime, @EndTime))
+ N'ms'
           PRINT @InformationalOutput
       END
       ELSE
       BEGIN
          SELECT @StartTime = getdate()
          EXEC('DBCC DBREINDEX('+ @QObjectName +', '
+ @QIndexName +') WITH NO_INFOMSGS' )
          SELECT @EndTime = getdate()
          SELECT @InformationalOutput = N' Total Time to process ='
+ convert(nvarchar, datediff(ms, @StartTime, @EndTime))
+ N'ms'
          PRINT @InformationalOutput
       END
```

```
        PRINT N' '
        FETCH NEXT FROM TableIndexList
        INTO @ObjectName, @IndexName, @IndexID, @ActualScanDensity
    END
    END
    PRINT N' '
    SELECT @InformationalOutput = N'** All Indexes have been rebuilt.  ** '
    PRINT @InformationalOutput
    DEALLOCATE TableIndexList
    GO
```

오류 로그 보기

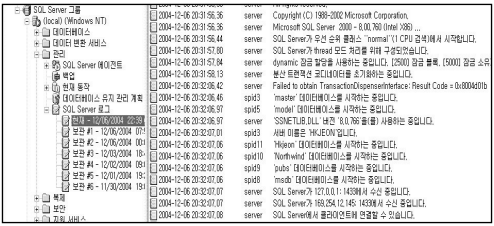
SQL Server에서는 이벤트를 SQL Server 오류 로그 및 응용 프로그램 로그에 기록합니다. 문제가 발생하였을 경우, 오류 로그에 기록되어 있는 정보를 사용하여, 문제의 원인을 찾을 수 있습니다.

■ SQL Server 오류 로그 보기

SQL Server 로그는 응용 프로그램의 상태 정보를 알기 위한 훌륭한 자료입니다. SQL Server 로그는 서비스가 시작할 때부터 서비스가 중지될 때까지 계속 메시지를 기록합니다. 모니터링 관리의 효율을 위하여 모니터링 담당자가 SQL Server 로그에서 찾아야 할 것을 정의합니다. 이 로그는 심각도 수준 19~25의 값을 가진 모든 오류를 기록합니다. 모니터링할 때, SQL Server 로그에서 심각도 수준 19~25 사이의 값을 가진 오류는 반드시 체크해야 합니다. 이 심각도 수준을 가진 오류는 트랜잭션을 실패하게 하고 응용 프로그램이 적절하게 운영되지 않도록 합니다. 심각도 수준 20에서 25사이의 오류는 치명적입니다. 만일 이 오류가 발생되면, 클라이언트 연결은 오류 메시지를 받은 후에 종료됩니다. 또한, RAISERROR...WITH LOG 문법을 사용하여 각각의 오류를 발견해 낼 수 있습니다. 이것은 오류 정보를 SQL Server 로그에 기록되게 합니다.

[따라하기]

1. EM에서 원하는 데이터베이스 서버를 선택합니다.
2. [관리]폴더를 클릭하고, [SQL Server 로그]를 클릭합니다.



3. 원하는 로그파일을 클릭하면, 오른쪽 창에 로그가 나타납니다.

> ***[참고] 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 overwrite되어 유실되는 것을 방지할 수 있습니다. 이 로그의 개수는 레지스트리 값을 수정하여 변경할 수 있습니다.*

```
-- 오류 로그 파일들의 파일 정보 확인
EXEC master..xp_enumerrorlogs
GO
-- 세번째 로그 파일 내용 보기
EXEC sp_readerrorlog 3
GO
```

```
-- 로그 파일의 개수를 20으로 변경하기
EXEC master..xp_regwrite 'HKEY_LOCAL_MACHINE'
, 'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer'
, 'NumErrorLogs'
, 'REG_DWORD'
, 20
GO
-- 확인
EXEC master..xp_regread 'HKEY_LOCAL_MACHINE'
, 'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer'
, 'NumErrorLogs'
GO
```

[참고] SQL Server 재시작 없이 새 오류 로그 생성하기

새 오류 로그는 SQL Server가 시작되는 경우에 생성되지만, SQL Server의 재시작 없이도 새 로그를 생성할 수 있습니다. 어떤 이유로 ERRORLOG 파일의 크기가 지나치게 커진 경우에는 sp_cycle_errorlog를 사용하여 새로운 ERRORLOG 파일을 생성할 것을 권고합니다.

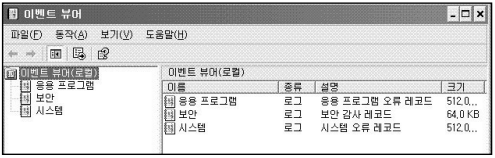
```
-- 새 로그 생성
EXEC sp_cycle_errorlog
GO
-- 오류 로그 파일별 정보 조회
EXEC master..xp_enumerrorlogs
GO
```

■ 응용 프로그램 로그 보기

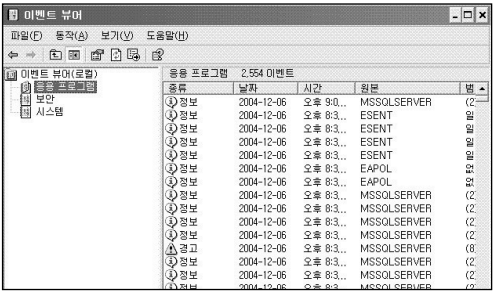
이벤트 뷰어는 사용자가 응용 프로그램, 보안, 시스템 로그에 기록되는 이벤트를 모니터링할 수 있도록 합니다. 이 로그는 SQL Server 로그와 SQL 에이전트 로그로 분리시켜 추가적인 정보를 제공합니다. SQL Server 메시지는 응용 프로그램 로그에서 발견됩니다. SQL Server 메시지는 "MSSQLSERVER" 또는 "SQLSERVERAGENT"라는 원본을 가진 메시지로 구별될 수 있습니다. RAISERROR 메시지도 여기에서 볼 수 있습니다.

[따라하기]

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



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



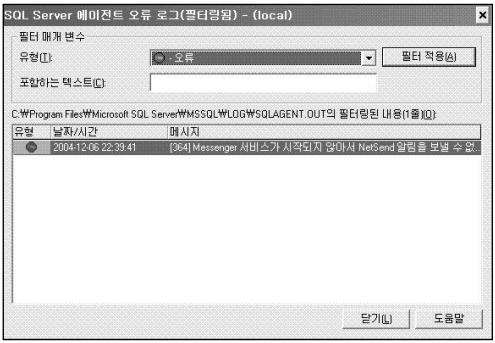
3. 원하는 사항을 더블 클릭하면, 자세한 정보를 얻을 수 있습니다.

■ SQL Server 에이전트 오류 로그 보기

SQL Server 에이전트 서비스는 작업 및 복제를 관리합니다.

1. EM에서 원하는 데이터베이스 서버를 선택합니다.
2. [관리] 폴더를 클릭합니다.

3. [SQL Server 에이전트]에 마우스를 대고 오른쪽 버튼을 클릭하고, [오류 로그 표시]를 선택합니다.

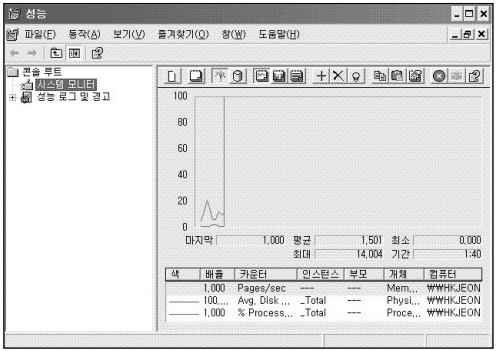


4. 오류를 선택하여 목록을 확인합니다.

성능 모니터링

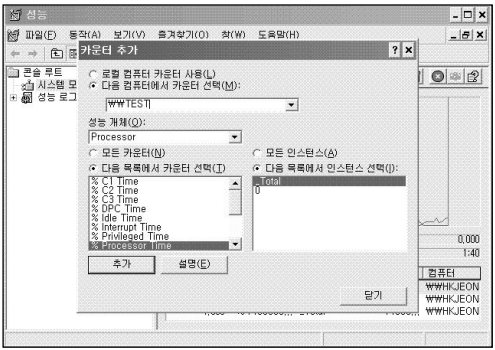
성능 모니터링 하기

1. [시작] → [설정] → [제어판] → [관리] → [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.



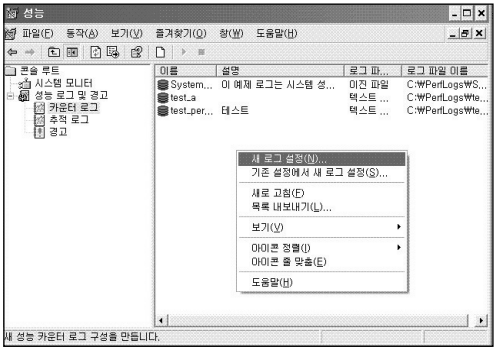
9. 선택한 카운터가 하단에 나타납니다.
10. 카운터를 제거할 경우에는, 제거하기를 원하는 카운터를 선택한 후, 상단 그래픽 메뉴에서, [x] (삭제키)를 클릭합니다.

성능 로그 생성하기

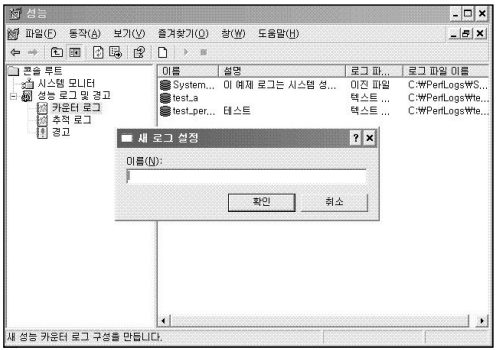
[따라하기]

1. [시작] → [설정] → [제어판] → [관리] → [성능]을 선택합니다.
2. [성능 로그 및 경고]의 더하기 기호(+)를 클릭합니다.
3. 카운터 로그를 선택합니다.

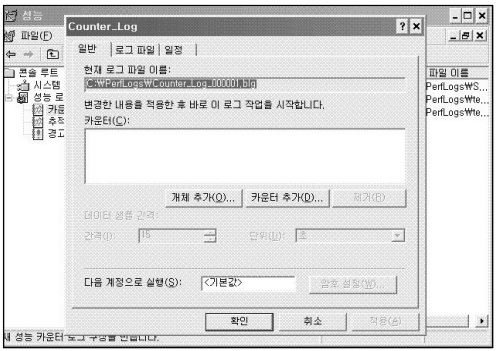
4. 오른쪽 창에 마우스를 대고 오른쪽 버튼을 클릭하여, [새 로그 설정]을 선택합니다.



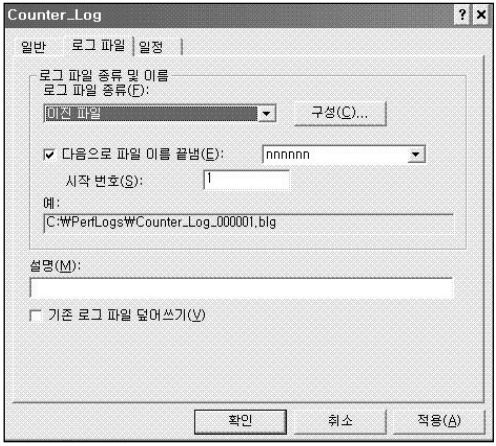
5. [새 로그 설정]에 원하는 로그 이름을 입력하고, [확인]을 클릭합니다.



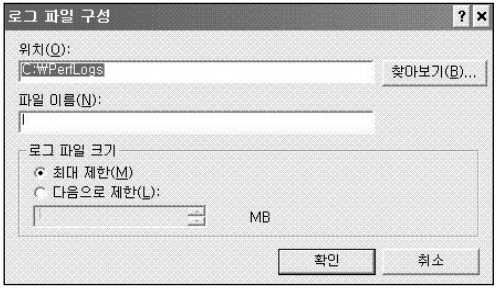
6. Counter_Log화면에 사용 정보와 파일 포맷을 설정합니다.



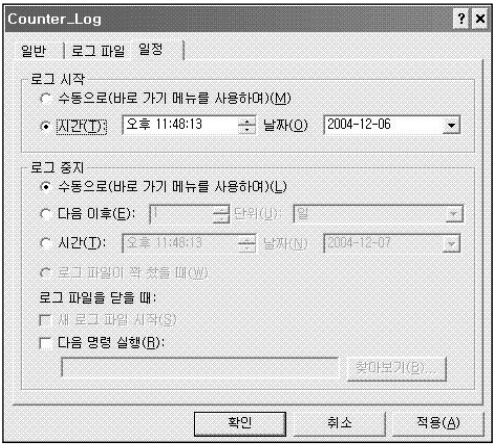
7. [개체추가]를 선택하여, 원하는 개체를 추가합니다.
8. [카운터추가]를 선택하여, 원하는 카운터를 추가합니다.
9. [데이터 샘플 간격]의 [간격]과 [단위]를 선택합니다.
10. [로그파일]탭을 선택합니다.
11. [로그 파일 종류]를 선택합니다.



12. 구성을 클릭하면, 로그 파일 구성 화면이 나타납니다.



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



21. 로그 시작 시간을 설정합니다.
22. 로그 중지 시간을 설정합니다.
23. 로그 파일을 닫을 때 실행할 명령이 있다면 선택합니다.
24. [확인]을 클릭합니다.
25. 오른쪽 창에 추가한 로그 파일의 목록이 나타납니다.

26. 새로 추가한 성능 로그의 이름위에서 마우스의 오른쪽 버튼을 클릭한 후 [다른 이름으로 설정 저장]으로 설정을 저장해 놓으면, 설정 파일을 재사용할 수 있습니다.

[참고]

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

B. 로그 파일 종류를 csv로 선택하여 수집하면, 분석 또는 집계하기에 편리합니다.

C. 수집 시간을 고려하여 데이터 샘플 간격을 설정합니다. 샘플 간격이 커질수록, 그래프의 정확도는 떨어지고, 샘플 간격이 작다면, 데이터의 크기가 커집니다.

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

D. 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.

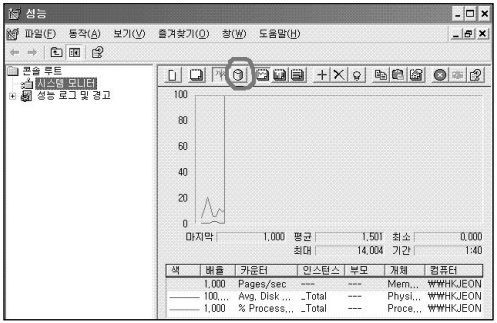
E. 수집 파일 크기는 적정한 값으로 제한합니다. 수집 파일이 너무 커지면, 파일이 열리지 않는 경우가 있습니다.

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

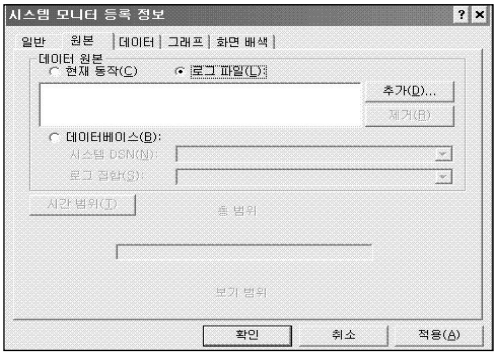
성능 로그의 재생

[따라하기]

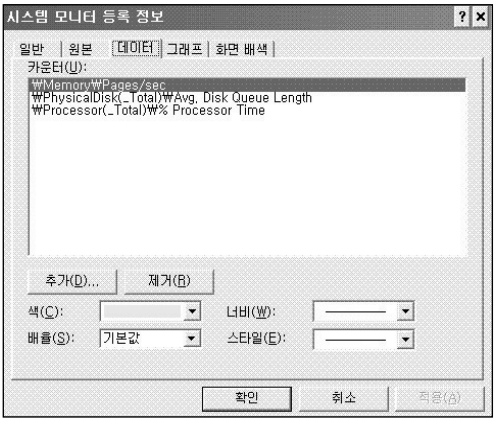
1. [시작] → [설정] → [제어판] → [관리] → [성능]을 선택합니다.



2. [로그 데이터 보기] 버튼을 클릭하면, [시스템모니터 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.

프로필러

성능 문제의 디버깅은 문제의 원인을 알아내는 것으로 시작합니다. 많은 경우, 성능 문제는 비효율적인 SQL 문에서 기인합니다. 비효율적인 SQL 문이 문제의 원인이라고 의심될 때, SQL 프로필러는 특정 SQL 문을 찾는 것에 도움을 줄 수 있습니다. 문제의 원인이 되는 SQL 문을 찾아, 튜닝하여 성능에 많은 도움을 줄 수 있습니다.

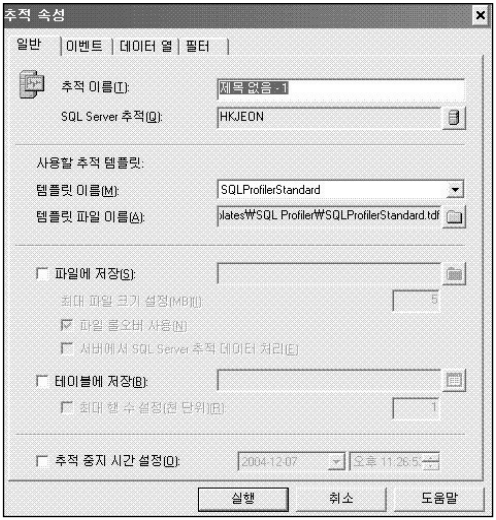
추적 수행하기 - GUI 사용

[따라하기]

1. 다음 방법 중 하나를 이용하여 프로필러를 실행합니다.
   [시작] → [프로그램] → [Microsoft SQL Server] → [프로필러] 또는
   엔터프라이즈 관리자의 상단 메뉴에서 [도구] → [SQL 프로필러]를 선택합니다.

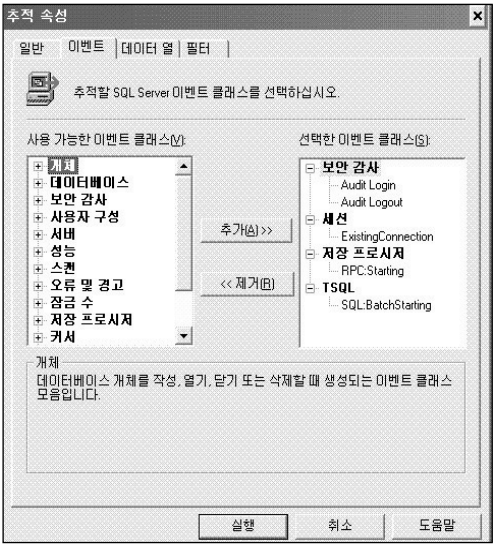
2. [파일] → [새로 만들기] → [추적]을 선택합니다.

3. 원하는 SQL 서버에 연결하면, [추적 속성]창이 나타납니다.

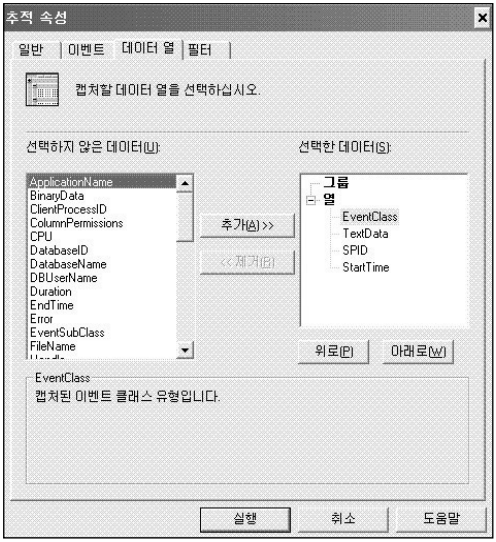


4. 추적이름을 입력합니다.
5. 추적할 SQL Server를 선택합니다.
6. 템플릿을 사용할 경우에 템플릿 이름을 선택합니다.
7. 파일에 저장하려면, [파일에 저장]을 선택하고, 저장할 위치와 파일명을 입력합니다.
8. 최대 파일 크기 설정을 합니다.

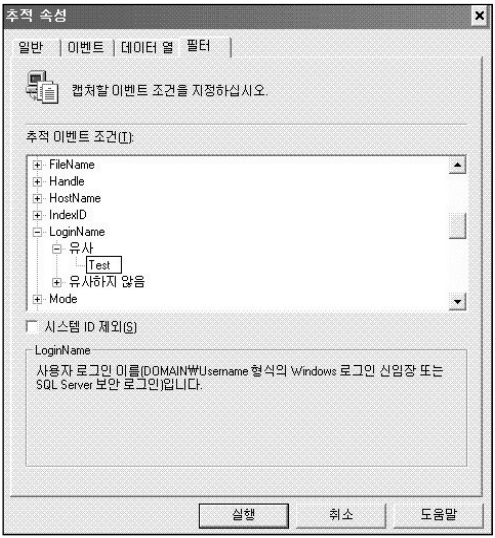
9. [이벤트] 탭을 선택한 후, 원하는 이벤트를 추가하거나, 제거합니다.



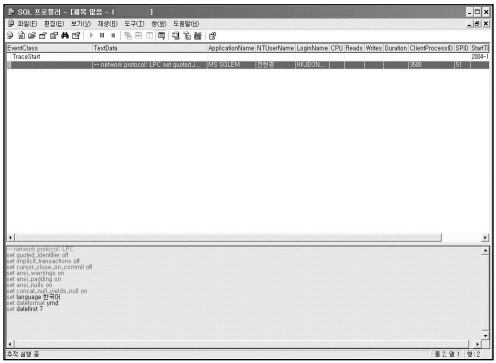
10. [데이터 열] 탭을 선택한 후, 수집할 데이터 열을 선택합니다.



11. 필터를 이용하고 싶다면, 필터 탭을 선택하여, 원하는 필터를 정의합니다. 예를 들어, LoginName이 Test인 것만 수집하고 싶다면 다음과 같이 설정합니다.



12. [실행]을 클릭하면, 수집이 시작됩니다.



13. 수집 중지를 위해서, 중지 버튼(붉은 네모)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.

[따라하기]

A. 오래 실행되는 SQL 문 찾기
오래 실행되는 쿼리는 잘못 튜닝된 시스템, 잘못 작성된 응용 프로그램, 또는 단순히 많은 동

작을 수행하는 작업등을 의미할 수 있습니다. 어떠한 경우건, 이러한 오래 실행되는 SQL 문을 찾아서 튜닝하는 것은 그 작업의 성능은 물론 전반적인 시스템 성능까지도 향상시킬 수 있습니다.

- 권장되는 추적 이벤트 : TSQL, SQL:BatchCompleted
- 정렬 기준 컬럼 : Duration

B. 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 컬럼 : CPU, Reads, Writes

C. 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 프로파일 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]

A. 불필요한 이벤트의 추가는 삼가합니다. 너무 많은 이벤트의 추가는 서버의 성능에 영향을 줄 수 있습니다.

B. 일반적인 경우라면, 모든 컬럼들을 포함시킵니다. 어떤 이벤트들은 보조적인 항목을 반환하는 어떤 항목들에 의존합니다. 적어도 다음 컬럼들은 포함하는 것이 좋습니다.

- *BinaryData*
- *ClientProcessID*

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

C. 테이블에 추적을 직접 저장하는 것은 좋지 않습니다. 추적 파일을 생성한 후, fn_trace_gettable 함수를 사용하면 추적 데이터를 테이블에 저장할 수 있습니다.

추적에 관련된 저장 프로시저의 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)=
'1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,2
9,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,',
-- 모든 컬럼
@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 컬럼을 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 컬럼들을 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

/* 실행 하기 */
EXEC sp_trace_Start
```

[참고]
A. output 파일을 지정하지 않으면, "c:\Trace" 밑에 추적 파일이 생성됩니다. 이 스크립트를 직접 실행하려면, "c:\Trace"를 생성합니다.
B. 추적 파일이 커질 수도 있으므로, output 파일이 생성되는 곳의 공간을 충분히 확보합니다.
C. 원하는 이벤트와 컬럼은 번호로 설정합니다. 이벤트와 컬럼 번호는 BOL을 참조하십시오.

■ 추적을 중지하는 저장 프로시저의 예제 스크립트 : sp_trace_stop

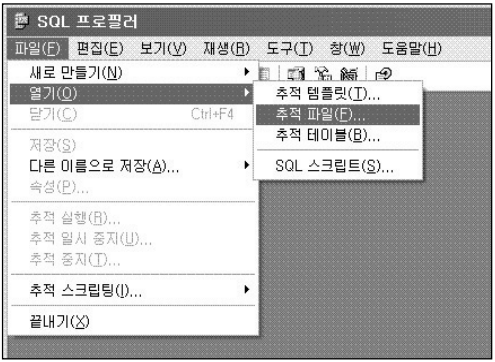
```
USE master
GO
CREATE PROCEDURE sp_trace_stop @TraceName sysname='trace'
AS
SET NOCOUNT ON

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

-- 추적 목록을 확인하여, 추적을 중지한다
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 프로시저 입니다.

추적 재생하기

1. 프로필러를 시작하고, [파일] → [열기] → [추적 파일]을 선택합니다.



2. 원하는 파일을 선택합니다.

자세한 내용은 SQL Server 온라인 설명서의 [추적 재생]을 참조하십시오.

유용한 유틸리티

■ PSSDIAG

PSSDIAG는 성능 분석에 필요한 여러가지 로그와 데이터를 수집할 수 있는 유틸리티입니다. 이 유틸리티를 사용하면 프로필러에 비해 부하를 덜 주면서 추적 데이터를 수집할 수 있으므로, 대용량 시스템의 경우에는 프로필러 대신 이 툴을 사용할 것을 권고합니다. 자세한 내용은 다음 url을 참조하십시오.

http://support.microsoft.com/?kbid=830232

■ Read80Trace

수집한 추적 정보를 분석하는데 매우 유용한 유틸리티입니다. 추적 데이터를 분석하여 로컬 데이터베이스에 분석한 정보를 저장해 주고, htm 파일 형태로 리소스를 많이 사용한 쿼리 또는 저장 프로시저, duration이 긴 쿼리 또는 저장 프로시저 등에 대한 분석 결과를 제공합니다. 자세한 내용은 다음을 참조하십시오.

http://support.microsoft.com/default.aspx?scid=kb;en-us;887057

문제 점검 및 해결

사용자 데이터베이스가 suspect로 표시된 경우에 문제 해결하기

Northwind 데이터베이스의 status 컬럼이 suspect로 설정된 경우를 예를 들어 설명합니다.

[주의]

sp_resetstatus SP는 아래와 같은 문제 해결을 위해서만 사용해야 하며, 사용 시 주의를 요합니다.

> ***[참고]***
> *SQL Server Errorlog 파일에 "Bypassing recovery for database 'Northwind' because it is marked SUSPECT." 와 같은 메시지가 기록됩니다.*

[따라하기]

1. sp_resetstatus가 없으면 생성합니다.

```
USE master
GO
EXEC sp_configure 'allow updates', 1
RECONFIGURE WITH OVERRIDE
GO
CREATE PROCEDURE sp_resetstatus @dbname varchar(30)
AS
DECLARE @msg varchar(80)
IF @@trancount > 0
BEGIN
   PRINT 'Can''t run sp_resetstatus from within a transaction.'
   RETURN (1)
END
```

```
IF suser_id() != 1
BEGIN
   SELECT @msg = 'You must be the System Administrator (SA)'
   SELECT @msg = @msg + 'to execute this procedure.'
   RETURN (1)
END
IF (SELECT COUNT(*) FROM master..sysdatabases WHERE name = @dbname)
!= 1
BEGIN
   SELECT @msg = 'Database '+ @dbname +' does not exist!'
   PRINT @msg
   RETURN (1)
END
IF (SELECT COUNT(*) FROM master..sysdatabases WHERE name = @dbname
AND status & 256 = 256) != 1
BEGIN
   PRINT 'sp_resetstatus can only be run on suspect databases.'
   RETURN (1)
END
BEGIN TRAN
   UPDATE master..sysdatabases SET status = status - 256
   WHERE name = @dbname
   IF @@error != 0 OR @@rowcount != 1
      ROLLBACK TRAN
   ELSE
   BEGIN
      COMMIT TRAN
      SELECT @msg = 'Database'+ @dbname +'status reset!'
      PRINT @msg
      PRINT''
```

```
        PRINT 'WARNING: You must reboot SQL Server prior to'
        PRINT 'accessing this database!'
        PRINT''
    END
GO
EXEC sp_configure 'allow updates', 0
RECONFIGURE WITH OVERRIDE
GO
```

2. Suspect 상태가 된 데이터베이스에 대하여 sp_resetstatus를 실행합니다.

```
EXEC sp_resetstatus Northwind
GO
```

3. ALTER DATABASE를 사용하여 Northwind 데이터베이스에 파일을 추가하여 여유 공간을 확보해 줍니다.

4. SQL Server를 중지하고 다시 시작합니다.

Tempdb가 suspect 상태가 된 경우의 문제 해결하기

Tempdb가 suspect 상태가 된 경우에 문제를 해결하는 방법을 설명합니다. 참고로 tempdb가 suspect 상태가 되면 SQL Server 서비스 시작이 실패할 수도 있습니다.

> ***[참고]***
> *SQL Server Errorlog 파일에 "Database 'tempdb' cannot be opened. It has been marked SUSPECT by recovery." 와 같은 메시지가 기록됩니다.*

[따라하기]

1. tempdb.mdf 파일과 tempdb.ldf 파일이 있는지 확인하고, 만약 있으면 파일들의 이름을 변경합니다.
2. 다음과 같은 명령어를 사용하여 명령 프롬프트 상에서 SQL Server를 시작합니다. 명명된 인스턴스인 경우에는 -s 매개변수를 지정합니다.

```
sqlservr -c -f -T3608 -T4022
```

[주의]

명령 프롬프트 창이 열린 채로 두어야 합니다. 명령 프롬프트 창을 닫으면 SQL Server 프로세스가 중지됩니다.

3. 쿼리 분석기에서 sp_resetstatus를 수행하여 tempdb의 suspect 상태를 해제합니다.

```
EXEC master..sp_resetstatus Tempdb
```

4. 명령 프롬프트 창에서 〈Ctrl+C〉 키를 눌러 SQL Server 서비스를 중지합니다.

5. SQL Server 서비스를 시작합니다. 이렇게 작업하면 Tempdb 데이터베이스 파일들이 새로 생성되며 tempdb가 정상적으로 복구됩니다.

손상된 데이터베이스 복구하기 (DBCC CHECKDB를 사용하여 오류 복구하기)

DBCC CHECKDB 명령어를 사용하면 특정 데이터베이스의 일관성(consistency)를 점검할 수 있습니다. DBCC CHECKDB 명령어는 데이터베이스 손상을 점검하는 주요 수단이며, 다음과 같은 사항들을 점검합니다.

- 인덱스 페이지와 데이터 페이지들이 제대로 연결되어 있는가
- 인덱스가 최신 상태이고, 제대로 정렬되어 있는가

- 포인트들이 일관성이 있는가 (Consistent)
- 각 페이지 상의 데이터가 최신 상태인가
- 페이지 오프셋이 최신 상태인가

[구문]

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

[사전지식]

DBCC CHECKDB나 DBCC CHECKTABLE에 REPAIR 옵션을 지정하여 수행하고자 하는 경우에는 SQL Server를 단일 사용자 모드로 시작해서는 안되고 해당 데이터베이스를 단일 사용자 모드(single user mode)로 설정해야 합니다.

[참고] 데이터베이스를 단일 사용자 모드로 설정하는 방법 세 가지

A. 엔터프라이즈 관리자에서 설정하기
1. 해당 데이터베이스에서 마우스의 오른쪽 버튼을 클릭한 다음에 [속성]을 선택합니다.
2. 속성 창에서 [옵션] 탭을 선택합니다.
3. "액세스 제한" checkbox를 선택한 다음에 "단일 사용자"를 선택하고 [확인] 버튼을 클릭합니다.

B. 쿼리 분석기에서 sp_dboption을 사용하여 설정하기

```
USE master
GO
EXEC sp_dboption db_name, single, true
GO
```

C. 쿼리 분석기에서 ALTER DATABASE를 사용하여 설정하기 (작업단계)

1. 지정한 시간이 경과한 후에 완료되지 않은 트랜잭션들을 롤백하고 단일 사용자 모드로 변경하고자 하는 경우
2. 완료되지 않은 트랜잭션들을 즉시 롤백하고 단일 사용자 모드로 변경하고자 하는 경우

[참고] 복구 옵션에 대하여 이해하기

A. REPAIR_FAST 옵션 : 사소한 손상을 복구하는 작업을 수행하며 수행 시간도 빠르고 데이터 유실도 유발하지 않습니다.

B. REPAIR_REBUILD 옵션 : REPAIR_FAST 에서 이루어지는 모든 복구 작업을 수행하고 인덱스 재작성과 같이 시간이 소요되는 복구 작업을 수행하며, 소요 시간이 길고 데이터 유실도 발생하지 않습니다.

C. REPAIR_ALLOW_DATA_LOSS 옵션 : REPAIR_REBUILD가 수행하는 모든 작업들을 동일하게 수행하며, 데이터 유실이 발생할 수 있는 작업을 추가로 수행합니다. 구조적인 문제와 페이지 오류를 정정하고 손상된 텍스트 오브젝트를 삭제하는 작업을 수행하기 때문에 데이터 유실이 발생할 수도 있습니다.

[참고] 복구 작업 단계

1. 해당 데이터베이스를 단일 사용자 모드로 변경합니다.
2. REPAIR 옵션을 지정하여 DBCC CHECKDB를 수행합니다.
   2-1. 먼저 REPAIR_FAST 나 REPAIR_REBUILD 옵션을 지정하여 문제 해결을 시도합니다.
   2-2. REPAIR_FAST 나 REPAIR_REBUILD 옵션으로 문제가 해결되지 않으면 REPAIR_ALLOW_DATA_LOSS 옵션을 사용합니다.

[주의]
REPAIR_ALLOW_DATA_LOSS 옵션을 사용하면 데이터의 유실이 발생할 수 있다는 점을 명심하기 바랍니다.

[권고사항]
REPAIR_ALLOW_DATA_LOSS 옵션을 사용하는 경우에는 명령어 수행 후에 다시 원래 상태로 복구할 수 있도록 하기 위해서 트랜잭션 내부에서 DBCC 명령어를 수행할 것을 권고합니다. 이와 같이 작업하면 복구 작업을 수행하고 결과를 확인한 다음에 필요한 경우에 롤백이 가능해집니다.

3. 복구가 완료되면 데이터베이스를 백업합니다.

[따라하기]

```
SELECT DATABASEPROPERTYEX ('Northwind', 'UserAccess')
GO
/* 결과:
MULTI_USER
*/

ALTER DATABASE Northwind
SET SINGLE_USER
WITH ROLLBACK AFTER 10 --10초 후에 완료되지 않은 트랜잭션들을 롤백
GO

SELECT DATABASEPROPERTYEX ('Northwind', 'UserAccess')
GO
/* 결과:
SINGLE_USER
*/

DBCC CHECKDB ('Northwind', REPAIR_FAST)
GO
```

```
ALTER DATABASE Northwind
SET MULTI_USER
GO
```

손상된 테이블 복구하기 (DBCC CHECKTABLE을 사용하여 오류 복구하기)

개별 테이블의 문제를 복구하고자 하는 경우에는 DBCC CHECKTABLE 명령어를 사용하면 됩니다.

[구문]

```
DBCC CHECKTABLE
    ( 'table_name' | 'view_name'
     [ , NOINDEX
       | index_id
       | { REPAIR_ALLOW_DATA_LOSS
       | REPAIR_FAST
       | REPAIR_REBUILD }
        ]
      )    [ WITH { [ ALL_ERRORMSGS | NO_INFOMSGS ]
                 [ , [ TABLOCK ] ]
                 [ , [ ESTIMATEONLY ] ]
                 [ , [ PHYSICAL_ONLY ] ] }
             ]
```

[따라하기]

```
SELECT DATABASEPROPERTYEX ('Northwind', 'UserAccess')
GO
```

```
/* 결과:
MULTI_USER
*/

ALTER DATABASE Northwind
SET SINGLE_USER
-- 10초 후에 완료되지 않은 트랜잭션들을 롤백
WITH ROLLBACK AFTER 10
GO

SELECT DATABASEPROPERTYEX ('Northwind', 'UserAccess')
GO
/* 결과:
SINGLE_USER
*/

USE Northwind
GO
DBCC CHECKTABLE (Orders, REPAIR_FAST)
GO

ALTER DATABASE Northwind
SET MULTI_USER
GO

-- EXEC sp_dboption 'Northwind', 'single user', 'FALSE'
-- GO
```

단일 로그 파일로 구성된 데이터베이스의 새로운 로그 파일 생성하기

[주의]
로그 파일이 오직 하나인 데이터베이스에 대해서만 사용할 수 있습니다.

[따라하기]
단일 로그 파일을 가지는 'TestDB'의 로그 파일이 유실/손상된 경우에 새로운 로그 파일을 생성하는 방법

```
EXEC sp_detach_db 'TestDB'
GO
EXEC sp_attach_single_file_db 'TestDB', 'E:\DBdata\TestDB_dat.mdf'
GO
```

DBCC REBUILD_LOG를 사용하여 새로운 로그 파일 생성하기

DBCC REBUILD_LOG는, 데이터베이스의 트랜잭션 로그 파일을 사용할 수 없는 경우에 새로운 로그를 재구축하는데 사용되는 명령어입니다.
예를 들어, 하드웨어의 장애로 인하여 로그 파일이 손상되거나 실수로 로그 파일을 삭제하여 기존의 로그 파일을 액세스할 수 없어서 데이터베이스를 사용할 수 없는 경우에 DBCC REBUILD_LOG를 사용하여 로그를 재구축할 수 있습니다. 로그 파일이 하나인 경우에는 먼저 위의 해결 방법을 적용해 본 다음에 실패하면 이 방법을 사용하기 바랍니다.

그러나, 이 명령어를 사용하면 로그에 반영되지 않은 유실된 트랜잭션들의 발생으로 데이터베이스의 일관성이 손상될 가능성이 매우 높다는 점을 유의해야 합니다. 문제가 발생하면 일단 다른 방법 (예를 들어, sp_attach_single_file_db)을 동원하여 문제 해결을 시도하고, 도저히 다른 방법으로는 데이터베이스를 복구할 수 없는 경우에 이 명령어를 사용하기 바랍니다. 이 명령어를 수행하면 로그에 반영되지 않는 트랜잭션의 발생으로 인하여 데이터 무결

성이 손상될 가능성이 높기 때문입니다. 이 명령어는 문서로 제공되지 않는 명령어이며 마이크로소프트 기술 지원 서비스의 지원 하에서 사용하는 것이 원칙이지만, 기술 지원 서비스를 받을 수 없는 경우의 응급 복구 작업에 참고하시라고 알려 드립니다. 또한, 하드웨어 장애와 같은 문제가 발생한 경우에는 트랜잭션 로그 파일 뿐만 아니라 데이터까지 손상시켰을 가능성이 높으므로, DBCC REBUILD_LOG가 성공적으로 완료된 이후에 단일 사용자 모드에서 DBCC CHECKDB를 수행하여 데이터의 일관성 (Consistency) 을 확인하는 작업이 반드시 필요합니다.

[구문]

DBCC REBUILD_LOG(*'db_name','log_filename'*)

db_name : 문제가 발생한 데이터베이스의 이름

log_filename : 새로운 로그 파일에 대한 완전한 물리적 경로

[주의]

이 방법을 사용하면 데이터 일관성이 손상될 가능성이 매우 높으므로 다른 방법으로는 도저히 데이터베이스를 복구할 수 없는 경우에 최후의 방법으로서 사용해야 하며, 매우 신중하게 작업해야 하며, 이 명령어는 문서로 제공되지 않는 명령어로서 마이크로소프트 제품 기술 지원 서비스의 지원 하에서 사용해야 합니다.

[오류 발생]

로그 파일이 유실 또는 손상된 경우에는 그 데이터베이스는 엔터프라이즈 관리자에 "주의 대상" (suspect)로 표시되며, 쿼리 분석기나 엔터프라이즈 관리자에서 주의
대상 상태가 된 데이터베이스를 액세스하려고 하면 다음과 같은 오류가 발생합니다

서버: 메시지 **945**, 수준 **14**, 상태 **2**, 줄 **1**
파일을 액세스할 수 없거나 메모리 또는 디스크 공간이 부족하여 **'RebuildLogTest'** 데이터베이스를 열 수 없습니다. 자세한 내용은 **SQL Server** 오류 로그를 참조하십시오.

그리고 SQL Server를 재시작하면 ERRORLOG 파일에 다음과 같은 오류 메시지가 기록됩니다

2005-01-12 14:51:56.72 spid11 'RebuildLogTest' 데이터베이스를 시작하는 중입니다.
2005-01-12 14:51:57.24 spid11 장치 활성화 오류입니다. 물리적 파일 이름
' C:\Program Files\Microsoft SQL
Server\MSSQL\data\RebuildLogTest_log.LDF' 이(가) 잘못된 것 같습니다.
이와 같은 오류가 발생하고 데이터베이스에 연결할 수 없는 경우에 다음과 같은 작업 단계로 복구 작업을 수행하면 로그를 재구축할 수 있습니다.

[예제]
다음은 RebuildLogTest라는 데이터베이스를 복구하는 예제입니다.
1. 설정된 옵션들을 확인합니다.

```
EXEC sp_dboption RebuildLogTest
GO
```

2. 시스템 테이블에 대한 직접적인 업데이트가 가능하도록 변경합니다.

```
EXEC sp_configure 'allow updates', 1
RECONFIGURE WITH OVERRIDE
GO
```

3. 문제가 발생한 데이터베이스를 응급 모드(bypass recovery)로 설정합니다

```
UPDATE master..sysdatabases SET status = 32768
WHERE name = 'RebuildLogTest'
GO
```

4. SQL Server 서비스를 중지하고 다시 시작합니다.

5. DBCC REBUILD_LOG를 수행합니다. 이 때 로그 파일의 이름에는 로그 파일이 저장될 경로까지 전체 이름을 기술해야 하며, 로그 파일은 기존에 존재하지 않는 이름을 지정해야 합니다.

```
USE master
GO
DBCC REBUILD_LOG('rebuildlogtest'
,'C:\Program Files\Microsoft SQL Server\MSSQL\data\RebuildLogTest_log.LDF')
GO
```

[참고]
이 명령어가 정상적으로 수행되면 결과창에 다음과 같은 메시지가 반환되며, 데이터베이스는 'dbo use only' 모드가 됩니다. 이전의 status 값과 무관하게 sysdatabases 테이블의 status 값이 2048로 설정됩니다. sp_dboption이나 엔터프라이즈 관리자를 사용하여 status 값을 원하는 값으로 변경하면 됩니다.

경고: 'RebuildLogTest' 데이터베이스에 대한 로그가 다시 작성되었습니다. 트랜잭션에 일관성이 없습니다. 물리적 일관성을 검사하려면 **DBCC CHECKDB**를 실행해야 합니다. 데이터베이스 옵션을 원래대로 설정하고 다른 로그 파일을 삭제해야 합니다.

DBCC 실행이 완료되었습니다. **DBCC**에서 오류 메시지를 출력하면 시스템 관리자에게 문의하십시오.

만약 이미 있는 파일 이름을 지정한 경우에는 다음과 같은 오류 메시지가 반환됩니다.

서버: 메시지 **5025**, 수준 **16**, 상태 **1**, 줄 **2**
'C:\Program Files\Microsoft SQL Server\MSSQL\data\RebuildLogTest_log.LDF' 파일이 이미 있습니다. 새 로그 파일을 만들려면 이 파일의 이름을 바꾸거나 삭제해야 합니다.

DBCC 실행이 완료되었습니다. **DBCC**에서 오류 메시지를 출력하면 시스템 관리자에게 문의하십시오.

6. 시스템 테이블을 직접 업데이트할 수 없도록 원래 값으로 변경합니다.

```
EXEC sp_configure 'allow updates', 0
RECONFIGURE WITH OVERRIDE
GO
```

7. 데이터베이스를 단일 사용자 모드로 변경하고 DBCC CHECKDB를 수행하여 일관성을 점검합니다. 데이터베이스를 단일 사용자 모드로 변경하는 보다 자세한 방법은 [손상된 데이터베이스 복구하기] 를 참조하기 바랍니다.

```
EXEC sp_dboption 'RebuildLogTest', 'single user', 'true'
DBCC CHECKDB('RebuildLogTest')
GO
```

8. DBCC CHECKDB를 수행한 결과 문제가 없으면, 그 데이터베이스를 정상적으로 사용할 수 있습니다. 그러나 롤백되어야 할 트랜잭션이 롤백되지 않거나, 데이터에 반영되어야 할 수정 작업이 반영되지 않는 문제가 발생할 수 있으므로, 논리적인 데이터의 무결성은 별도의 점검이 필요합니다.

9. 데이터베이스 옵션을 원래대로 설정하고, 로그 파일의 크기도 원래 크기로 확장합니다. 로그 파일을 재구축하면, 504KB의 작은 크기의 로그 파일이 생성됩니다.

교착상태(Deadlock) 발생 시 교착상태 추적하기

추적 플래그 1204를 사용하면 교착상태(Deadlock)에 대한 내용을 확인하는 것이 가능합니다. 명령 프롬프트에서 추적 플래그를 추가하여 SQL Server 서비스를 시작할 수도 있고, 엔터프라이즈 관리자에서 SQL Server 시작 매개 변수에서 추적 플래그를 추가할 수도 있습니다.

■ 명령 프롬프트에서 추적 플래그를 지정하는 방법

[따라하기]

1. SQL Server 서비스를 중지해도 되는 시점에 SQL Server 서비스를 중지합니다.

2. 다음과 같이 추적 플래그를 추가하고 SQL Server 서비스를 시작합니다.

[주의]

SQL Server 서비스를 시작한 명령 프롬프트의 창은 그대로 두어야 합니다. 명령프롬프트 창을 닫거나, 〈CTRL+C〉를 입력하면 SQL Server 서비스가 중지됩니다.

[참고] sqlservr.exe 파일은 SQL Server 설치 폴더의 하위 폴더 중 하나인 binn에 있습니다.

[예]

추적 플래그를 추가하여 디폴트 인스턴스 SQL Server 서비스를 시작하는 예제 (SQL Server 2000 서비스 팩3부터는 -T3605를 추가하지 않아도 ERRORLOG에 추적결과가 기록됩니다.)

```
sqlservr -c -T1204  -T3605
```

3. 위와 같이 작업하면 교착상태 추적 결과가 SQL Server 서비스가 시작된 콘솔 화면과 ERRORLOG 파일로 기록됩니다.

■ 엔터프라이즈 관리자에서 추적 플래그를 지정하는 방법

1. 엔터프라이즈 관리자에서 [속성] → [시작 매개 변수] → "매개 변수"에 -T1204와 -T3605를 입력하고 [추가] 버튼을 클릭한 다음에 [확인] 버튼을 클릭합니다.
2. SQL Server 서비스를 중지하고 재시작 합니다.
3. 교착상태가 발생하면 교착상태 추적 결과가 ERRORLOG 파일로 기록됩니다.

[교착상태에 대한 추적결과 예]

추적 플래그 1204를 추가하고 SQL Server 서비스를 시작하면 교착상태에 대한 추적결과가 다음과 같은 형태로 반환 또는 기록됩니다.

```
2003-02-25 05:12:55.13 spid4
Deadlock encountered .... Printing deadlock information
2003-02-25 05:12:55.13 spid4
2003-02-25 05:12:55.13 spid4     Wait-for graph
2003-02-25 05:12:55.13 spid4
2003-02-25 05:12:55.13 spid4     Node:1
2003-02-25 05:12:55.14 spid4     RID: 2:1:15:0                CleanCnt:1
Mode: X Flags: 0x2
2003-02-25 05:12:55.14 spid4      Grant List 0::
2003-02-25 05:12:55.14 spid4        Owner:0x192e32e0 Mode: X        Flg:0x0
Ref:0 Life:02000000 SPID:52 ECID:0
2003-02-25 05:12:55.15 spid4        SPID: 52 ECID: 0 Statement Type: DELETE
Line #: 1
2003-02-25 05:12:55.15 spid4        Input Buf: Language Event: delete deadt2

2003-02-25 05:12:55.15 spid4      Requested By:
2003-02-25 05:12:55.15 spid4         ResType:LockOwner Stype:'OR' Mode: U
SPID:51 ECID:0 Ec:(0x19CA3500) Value:0x192e3340 Cost:(0/98)
2003-02-25 05:12:55.16 spid4
2003-02-25 05:12:55.16 spid4     Node:2
2003-02-25 05:12:55.16 spid4     RID: 2:1:28:0                CleanCnt:1 Mode: X
Flags: 0x2
2003-02-25 05:12:55.17 spid4     Grant List 0::
2003-02-25 05:12:55.17 spid4        Owner:0x192e3400 Mode: X        Flg:0x0
Ref:0 Life:02000000 SPID:51 ECID:0
2003-02-25 05:12:55.17 spid4        SPID: 51 ECID: 0 Statement Type: DELETE
Line #: 1
2003-02-25 05:12:55.18 spid4        Input Buf: Language Event: delete deadt1

2003-02-25 05:12:55.18 spid4      Requested By:
```

```
2003-02-25 05:12:55.18 spid4        ResType:LockOwner Stype:'OR' Mode: U
SPID:52 ECID:0 Ec:(0x19AB9508) Value:0x192e3300 Cost:(0/98)
2003-02-25 05:12:55.19 spid4    Victim Resource Owner:
2003-02-25 05:12:55.19 spid4        ResType:LockOwner Stype:'OR' Mode: U
SPID:52 ECID:0 Ec:(0x19AB9508) Value:0x192e3300 Cost:(0/98)
```

블로킹 발생 시 원인 추적하기

다음에 소개하는 저장 프로시저들은 블로킹을 점검하는데 유용하게 사용할 수 있는 저장 프로시저들입니다. 다음의 저장 프로시저들은 master 데이터베이스에 생성해 두고 블로킹 발생 시에 활용하실 것을 권고합니다.

■ **sp_blocker_pss80**

블로킹에 관한 전반적인 정보를 수집할 수 있는 매우 유용한 저장 프로시저입니다. Microsoft 웹사이트의 다음 아티클에 sp_blocker_pss80 저장 프로시저 생성 스크립트가 있으므로 활용하시기 바랍니다.

http://support.microsoft.com/default.aspx?scid=kb;en-us;271509

(아티클 제목 : How to monitor SQL Server 2000 blocking)

■ **sp_leadblocker, sp_blockinglocks**

Inside SQL Server 책에 있는 스크립트로서, 블로킹 발생의 원인이 되는 프로세스에 대한 정보와, 블로킹에 관련되는 잠금에 대한 정보를 제공하는 저장 프로시저입니다.

[따라하기]

1. 블로킹 추적에 유용한 저장 프로시저들을 생성합니다.
(sp_blocker_pss80, sp_leadblocker, sp_blockinglocks)

```
USE master
GO
CREATE PROCEDURE sp_leadblocker  AS
IF EXISTS
  (SELECT * FROM master.dbo.sysprocesses
  WHERE spid IN (SELECT blocked FROM master.dbo.sysprocesses))
  SELECT
    spid, status, loginame=SUBSTRING(SUSER_SNAME(sid), 1, 12),
    hostname=substring(hostname, 1, 12),
      blk=CONVERT(char(3), blocked),
    dbname=substring(db_name(dbid),1,10),cmd, waittype
  FROM master.dbo.sysprocesses
  WHERE spid IN (SELECT blocked FROM master.dbo.sysprocesses)
  AND blocked=0
ELSE
SELECT 'No blocking processes found!'
GO
CREATE PROCEDURE sp_blockinglocks AS
SET NOCOUNT ON
SELECT  DISTINCT CONVERT (SMALLINT, L1.req_spid) AS SPID,
 L1.rsc_dbid AS DBID,
 L1.rsc_objid AS OBJID,
 L1.rsc_indid AS INDID,
 SUBSTRING (V.name, 1, 4) AS TYPE,
 SUBSTRING (L1.rsc_text, 1, 16) AS RESOURCE,
 SUBSTRING (U.name, 1, 8) AS MODE,
 SUBSTRING (X.name, 1, 5) AS STATUS
```

```
FROM  master.dbo.syslockinfo L1,
   master.dbo.syslockinfo L2,
  master.dbo.spt_values V,
  master.dbo.spt_values X,
  master.dbo.spt_values U
WHERE L1.rsc_type = V.number
  AND V.type = 'LR'
  AND L1.req_status = X.number
  AND X.type = 'LS'
  AND L1.req_mode + 1 = U.number
  AND U.type = 'L'
  AND L1.rsc_type <> 2 /* 2 : DB LOCK */
  AND L1.rsc_dbid = L2.rsc_dbid
  AND L1.rsc_bin = L2.rsc_bin
  AND L1.rsc_objid = L2.rsc_objid
  AND L1.rsc_indid = L2.rsc_indid
  AND L1.req_spid <> L2.req_spid
  AND L1.req_status <> L2.req_status
 --AND(L1.req_spid IN (SELECT BLOCKED FROM master..SYSPROCESSES)
  -- OR L2.req_spid IN (SELECT BLOCKED FROM master..SYSPROCESSES))
ORDER BY SUBSTRING (L1.rsc_text, 1, 16), SUBSTRING (X.name, 1, 5)
RETURN (0)
GO
```

2. 시스템 SP 및 DBCC 명령어와 작업 단계 1에서 추가한 SP를 수행하여 그 결과를 분석합니다.

2-1. sp_blocker_pss80 활용예

```
WHILE 1=1
BEGIN
EXEC master.dbo.sp_blocker_pss80
```

```
-- Or for fast mode
-- EXEC master.dbo.sp_blocker_pss80 @fast=1
-- Or for latch mode
-- EXEC master.dbo.sp_blocker_pss80 @latch=1
WAITFOR DELAY '00:00:15'
END
GO
2-2. sp_leadblocker, sp_blockinglocks 활용예
EXEC sp_leadblocker
EXEC sp_blockinglocks
GO
```

2-3. 블로킹을 유발하는 프로세스에 대하여 sp_lock, sp_who2, sp_who 등의 시스템 SP를 수행하면 잠금과 프로세스에 대한 보다 자세한 내용을 별도로 점검할 수 있습니다.

```
EXEC sp_who2 53
EXEC sp_lock 53
GO
```

2-4. 트랜잭션을 오픈한 채로 있는 프로세스가 블로킹을 유발하는 경우에는 DBCC OPENTRAN을 사용하여 특정 데이터베이스에서 가장 오래된 활성 트랜잭션에 대한 정보를 점검할 수 있습니다. 참고로, 트랜잭션에 트랜잭션 이름을 기술하면 문제가 있는 트랜잭션을 확인하는 작업이 용이해집니다.

```
USE pubs
DBCC OPENTRAN
GO
-- 또는
DBCC OPENTRAN ('pubs')
GO
```

대기(wait) 점검하기

[따라하기]

1. wait 정보를 저장할 데이터베이스를 생성합니다.

```
USE master
GO
CREATE DATABASE DBAdmin
ON (
   NAME = DBAdmin_dat,
   FILENAME = 'D:\DBdata\DBAdmin_dat.mdf'
   SIZE = 500 MB,
   MAXSIZE = 1 GB,
   FILEGROWTH = 100 MB)
LOG ON (
   NAME = DBAdmin_log,
   FILENAME = 'D:\ DBData\DBAdmin_log.ldf',
   SIZE = 100 MB,
   MAXSIZE = 500 MB,
   FILEGROWTH = 100 MB)
GO
```

2. wait 정보를 추적할 저장 프로시저를 생성합니다.

```
USE DBAdmin
GO
CREATE PROCEDURE get_waitstats
AS
-- This stored procedure is provided "AS IS" with no warranties,
-- and confers no rights.
-- Use of included script samples are subject to the terms specified at
-- http://www.microsoft.com/info/cpyright.htm
```

```
-- this proc will create waitstats report listing wait types by percentage
-- can be run when track_waitstats is executing
SET NOCOUNT ON

DECLARE @now datetime, @totalwait numeric(20,1)
      ,@endtime datetime,@begintime datetime
      ,@hr int, @min int, @sec int

SELECT @now=max(now),@begintime=min(now),@endtime=max(now)
FROM waitstats WHERE [wait type] =  'Total'

-- subtract waitfor, sleep, and resource_queue from Total
SELECT @totalwait = sum([wait time]) + 1
FROM waitstats
WHERE [wait type] not in
('WAITFOR','SLEEP','RESOURCE_QUEUE', 'Total', '***total***')
AND now = @now

-- insert adjusted totals, rank by percentage descending
DELETE waitstats WHERE [wait type] = '***total***' and now = @now
INSERT INTO waitstats select '***total***',0,@totalwait,@totalwait,@now

SELECT [wait type],[wait time]
,percentage=cast (100*[wait time]/@totalwait as numeric(20,1))
FROM waitstats
WHERE [wait type] not in
  ('WAITFOR','SLEEP','RESOURCE_QUEUE','Total')
AND now = @now
ORDER BY percentage DESC
GO
CREATE PROCEDURE track_waitstats (@num_samples int=10,@delaynum
int=1,@delaytype nvarchar(10)='minutes')
```

```
AS
-- T. Davidson
-- This stored procedure is provided "AS IS" with no warranties,
-- and confers no rights.
-- Use of included script samples are subject to the terms specified at
-- http://www.microsoft.com/info/cpyright.htm
-- @num_samples is the number of times to capture waitstats,
-- default is 10 times.  default delay interval is 1 minute
-- delaynum is the delay interval,
-- delaytype specifies whether the delay interval is minutes or seconds
-- create waitstats table if it doesn` t exist,
-- otherwise truncate
SET NOCOUNT ON
IF NOT EXISTS (SELECT 1 FROM sysobjects WHERE name = 'waitstats')
      CREATE TABLE waitstats ([wait type] varchar(80),
                  requests numeric(20,1),
                  [wait time] numeric (20,1),
                  [signal wait time] numeric(20,1),
                  now datetime default getdate())
ELSE TRUNCATE TABLE waitstats

DBCC SQLPERF (waitstats,clear)  -- clear out waitstats
DECLARE @i int,@delay varchar(8),@dt varchar(3)
,@now datetime, @totalwait numeric(20,1)
      ,@endtime datetime,@begintime datetime
      ,@hr int, @min int, @sec int
SELECT @i = 1
   SELECT @dt = CASE lower(@delaytype)
      WHEN 'minutes' THEN 'm'
      WHEN 'minute' THEN 'm'
      WHEN 'min' THEN 'm'
      WHEN 'mm' THEN 'm'
```

```
        WHEN 'mi' THEN 'm'
        WHEN 'm' THEN 'm'
        WHEN 'seconds' THEN 's'
        WHEN 'second' THEN 's'
        WHEN 'sec' THEN 's'
        WHEN 'ss' THEN 's'
        WHEN 's' THEN 's'
        ELSE @delaytype
END
IF @dt not in ('s','m')
BEGIN
PRINT 'please supply delay type e.g. seconds or minutes'
RETURN
END

IF @dt = 's'
BEGIN
SELECT @sec = @delaynum % 60
SELECT @min = cast((@delaynum / 60) as int)
SELECT @hr = cast((@min / 60) as int)
SELECT @min = @min % 60
END
IF @dt = 'm'
BEGIN
SELECT @sec = 0
SELECT @min = @delaynum % 60
SELECT @hr = cast((@delaynum / 60) as int)
END
SELECT @delay= right('0'+ convert(varchar(2),@hr),2) + ':' +
+ right('0'+convert(varchar(2),@min),2) + ':' +
+ right('0'+convert(varchar(2),@sec),2)
IF @hr > 23 or @min > 59 or @sec > 59
```

```
BEGIN
SELECT'hh:mm:ss delay time cannot 〉 23:59:59'
SELECT 'delay interval and type: ' + convert (varchar(10),@delaynum)
+ ',' + @delaytype + 'converts to' + @delay
RETURN
END
WHILE (@i 〈= @num_samples)
BEGIN
INSERT INTO waitstats ([wait type], requests, [wait time],[signal wait time])
EXEC ('DBCC SQLPERF(WAITSTATS)')
SELECT @i = @i + 1
WAITFOR DELAY @delay
END
--- create waitstats report
EXEC get_waitstats
GO
```

3. 수집 간격과 반복 실행 횟수를 지정하여 대기 정보를 수집합니다.

```
-- 2초 간격으로 10번 반복 수행 예제
USE DBAdmin
EXEC
Track_waitstats @num_samples=10,@delaynum=2,@delaytype='seconds'
GO
SELECT * FROM waitstats
GO
-- 실행 예제 : 디폴트 ( 실행 소요 시간 : 10분 )
USE DBAdmin
EXEC Track_waitstats
GO
```

마치면서

본 포켓 관리 가이드는 DBA가 기본적으로 알아야 할 내용을 담고 있습니다. 자세한 내용은 이를 바탕으로 더욱 정진하시기 바랍니다. 또한 지면 관계상 성능에 대한 내용은 포함시키지 못했습니다. 성능에 대해서는 포켓 가이드 시리즈인 "SQL Server 성능 향상을 위한 튜닝 가이드"를 참조하시기 바랍니다.

- SQL Server 컨설턴트 전현경 저

비매품

SQL Server DBA 가이드

- 저자 : 전현경
- 감수 : 하성희, 최인규
- 기획 : 하성희 (에이디 컨설팅)
- Contents 관련문의 : hkjeon129@empal.com

- 발행 : ㈜한국마이크로소프트
- 제작 : 에이디컨설팅
- 초판 발행일 : 2005년 1월
- 개정판 발행일 : 2005년 2월

Microsoft®
SQL Server® DBA 가이드

Microsoft®
(주)한국마이크로소프트
서울특별시 강남구 대치동 892번지 포스코센터 서관 5층
전화 : 02-531-4500(대)
팩스 : 02-555-1724
인터넷 : http://www.microsoft.com/korea/sql