Microsoft SQL Server 2000

T-SQL 작성 시 체크리스트

※ 본 체크리스트를 충실히 실행하시면, 더 좋은 성능을 기대하실 수 있습니다.

체크	항목
	반복 기반의 솔루션을 작성하지 말고, 최소의 SQL 문으로 구성된 집합 기반의 솔루션을 작성합니다. 대개의 경우 하나의 SQL 문으로도 원하는 작업을 수행할 수 있으며, 일반적으로 반복 솔루션보다 집합 기반 솔루션이 성능도 우수하고 코드 또한 단순합니다.
	커서는 많은 리소스를 필요로 하며 상당한 부하를 발생시키므로 사용하지 않는 것이 최선입니다. 만약 커서를 사용하고자 한다면 다양한 커서의 유형에 대하여 이해하고 적절한 커서를 사용해야 합니다.
	한 가지 솔루션을 개발하였다고 만족해서는 안 되고 다양한 솔루션을 개발하고 각 솔루션의 성능, 코드의 명료성, 확장성 등을 비교하여 가장 최적의 솔루션을 찾아내는 노력을 해야 합니다.
	쿼리 작성 후에는 결과만 확인하지 말고 항상 쿼리 옵티마이저가 어떤 실행 계획을 작성했는지 확인합니다. 쿼리 분석기의 실행 계획 표시 기능을 사용하거나 SET STATISTICS PROFILE ON 을 사용하면 실행 결과와 함께 실행 계획 정보의 확인이 가능합니다. 또한 SET SHOWPLAN_ALL ON, SET SHOWPLAN_TEXT ON과 예상 실행 계획 표시 기능을 사용하면 실제로 쿼리를 실행하지는 않고 실행 계획 정보만 확인할 수 있습니다.
	SELECT 문에 * 대신 필요한 컬럼들의 이름을 기술합니다. * 를 사용하면 네트워크 트래픽을 증가시키고 더 많은 버퍼와 처리를 발생시킬 뿐 아니라, 테이블이나 뷰의 구조가 변경될 때 오류 발생 가능성이 높아집니다.
	INSERT 문에 실제로 값이 입력되는 컬럼들의 이름을 기술합니다. 모든 컬럼에 대하여 INSERT를 수행하는 것은 비효율적일 뿐 아니라, 테이블이나 뷰의 구조가 변경될 때마다 INSERT 문을 수정해야 하는 불편이 따릅니다.
	오브젝트를 참조할 때에는 항상 오브젝트 소유자를 지정합니다.
	필터를 지정합니다. 클라이언트 측에서 데이터를 필터링하기 보다는 쿼리에 필터를 적용하여 SQL 서버가 접근하는 물리적인 데이터의 범위를 제한하고 인덱스를 사용할 수 있도록 하는 것이 성능 측면에서 도움이 됩니다.
	WHERE 절은 SARG가 되도록 작성합니다. SARG의 컬럼에는 함수, 산술연산, 또는 기타 컬럼을 가공하는 식을 사용해서는 안되며, 선택성이 좋은 SARG를 추가합니다.
	〈〉, !=, NOT IN, NOT EXISTS와 같은 비동등 연산자의 사용을 최소화합니다.
	옵티마이저가 반환되는 행의 퍼센트를 가정하여 실행 계획을 작성함으로써 비효율적인 실행 계획이 작성되지 않도록, WHERE 조건절에 로컬 변수 대신 상수값 또는 입력 매개 변수를 사용합니다.
	SQL 서버가 묵시적으로 컬럼의 데이터 타입을 변환함으로써 성능 저하가 발생하지 않도록, 조건절에서 비교되는 식의 데이터 타입을 일치시킵니다.
	테이블의 전체 행 수를 확인하고자 하는 경우에는 count(컬럼명) 대신 count(*)를 사용합니다.
	빈 테이블 또는 Table 변수나 temp table에 INSERT를 한 다음에 INSERT된 행의 수를 확인하고자 하는 경우에는 count(*) 대신 @@ROWCOUNT 전역 변수를 사용합니다.
	테이블에 어떤 조건에 해당되는 행이 존재하는지 확인하는 경우에는 count(*) 대신 IF EXISTS 절을 사용합니다.
	INSERT, UPDATE, DELETE 문에 대하여 적절한 오류 처리 루틴을 작성합니다. @@ERROR, @@ROWCOUNT, 저장 프로시저의 Return Status 값을 활용합니다.
	ORDER BY, GROUP BY, DISTINCT, HAVING 절은 반드시 필요한 경우에만 사용합니다. 데이터가 반드시 특정 순서로 정렬될 필요가 있을 때에만 ORDER BY 절을 지정하며. DISTINCT가 반드시 필요한 경우에는 DISTINCT 대신 EXISTS나 IN의 사용을 고려합니다.
	SELECT 문이 반환하는 결과 집합이 중복 행을 포함하지 않는 경우에는 불필요한 정렬이 발생하지 않도록 UNION 대신 UNION ALL을 사용합니다.
	UPDATE 문 또는 DELETE 문의 FROM 절과 조인을 잘 활용합니다. 여러 번 나누어서 처리하지 말고, FROM 절과 조인을 적절하게 사용하여 하나의 UPDATE 문이나 DELETE 문으로 처리합니다.

T-SQL 작성 시 체크리스트

※ 본 체크리스트를 충실히 실행하시면, 더 좋은 성능을 기대하실 수 있습니다.

체크	항목
	Business Requirements에 위배되지 않는 범위 내에서 가능한 한 낮은 Transaction Isolation level을 사용합니다. Dirty Read가 문제가 되지 않는 경우에는 SELECT 문에 WITH (READUNCOMMITTED) 힌트를 사용합니다.
	트랜잭션은 가능한 한 짧게 작성하며, 트랜잭션 내에서 오류가 발생하면 롤백을 수행하도록 작성해야 하며, 가능한 한 빨리 롤백을 처리합니다.
	조인 연산은 ANSI-표준 조인 구문을 사용합니다. 외부 조인의 경우에는 ANSI 스타일의 조인과 T-SQL 스타일의 조인 결과가 다르므로 이에 대한 명확한 이해가 필요하며 코딩 시 유의해야 합니다.
	NULL을 처리할 때 ANSI 표준 구문을 사용합니다. @var = NULL 또는 @var 〈〉 NULL 대신, @var IS NULL 또는 @var IS NOT NULL 형태로 작성합니다.
	동일한 컬럼에 대하여 여러 개의 상수값을 비교하는 경우에는 OR 대신 IN 을 사용합니다. 또한 OR 또는 IN 을 범위로 제한이 가능한 경우에는 BETWEEN 으로 변경합니다.
	우선 순위가 서로 다른 연산자들이 혼재되어 있는 식의 경우에는 코드가 명확하도록 괄호를 사용합니다. 괄호를 사용하지 않으면 원하는 값이 아닌 잘못된 결과를 얻는 문제가 발생할 수 있습니다.
	가능한 한 임시 테이블을 사용하지 않습니다. 대개의 경우 임시 테이블을 사용하지 않는 솔루션의 성능이 더 좋습니다. 임시 테이블이 필요한 경우에는 table 변수의 사용을 고려해 볼 수 있습니다.
	테이블에 있는 모든 행들을 삭제하는 경우에는 DELETE 문 대신 TRUNCATE TABLE 명령어를 사용합니다. TRUNCATE TABLE은 삭제된 행 단위로 트랜잭션 로그를 기록하지 않고 페이지 할당을 해제하는 방식이므로 속도가 빠릅니다. 그렇지만, 실행 권한과 외래 키 제약 조건 존재 여부, IDENTITY 속성에 대한 내용을 확인한 후에 사용해야 합니다.
	반환되는 행의 수가 지나치게 많은 경우에는 결과 집합의 크기를 제한합니다.
	보편적인 경우에 성능이 좋다는 것을 검증한 다음에 인덱스 힌트를 사용합니다. 잘못된 인덱스 힌트의 사용은 오히려 성능을 떨어뜨릴 수 있습니다.
	T-SQL 문장의 접미사로 세미콜론(;)을 사용합니다. SQL 서버 2005부터는 일부 문장에 세미콜론의 사용이 필수입니다.
	/* 주석 */, -- 을 사용하여 적절하게 주석을 기술합니다.
	Ad-hoc Query 대신 저장 프로시저를 사용합니다. 저장 프로시저는 데이터의 보안 및 일관성 유지, 클라이언트와 서버 간의 불필요한 라운드 트립 감소로 인한 네트워크 부하 감소, 쿼리 실행 계획의 재사용이 가능합니다.
	사용자 데이터베이스에 생성하는 저장 프로시저의 이름은 'sp_' 로 시작하는 이름을 사용하지 않아야 합니다.
	저장 프로시저 호출 시 반드시 소유자를 지정합니다. (예: EXEC dbo.up_MyProc)
	저장 프로시저의 입력 매개 변수는 매개 변수의 위치에 맞추어 값만 기술하지 말고 매개 변수의 이름과 값을 함께 기술합니다.
	저장 프로시저의 시작 부분에 SET NOCOUNT ON 을 기술함으로써, 클라이언트와 서버 간의 네트워크 라운드 트립을 줄입니다. 이런 원칙은 저장 프로시저 표준 템플릿을 작성하여 활용하면 편리합니다.
	저장 프로시저의 입력 매개 변수에 대하여 디폴트 값을 부여합니다. 그리고 저장 프로시저의 시작 부분에서 입력 매개 변수의 유효성을 점검함으로써 불필요한 코드 실행을 방지합니다.
	저장 프로시저 재컴파일을 줄이기 위하여, 저장 프로시저의 시작 부분에 모든 DDL들을 기술한 다음에 DML을 기술하며 User option, language, dateformat 에 대한 표준화 원칙을 수립하고 표준화 원칙을 따라 작성합니다.
	동적 SQL 문의 사용은 가능한 한 피하며, 동적 SQL 문을 사용해야 하는 경우에는 EXEC 대신 sp_executesql 을 사용합니다.

※ 본 체크리스트는 SQL Server 2000 기반으로 작성되었으며, SQL Server 2005 기반 체크리스트는 2006년 상반기중 제공할 예정입니다.