Microsoft®
SQL Server™ 2005

Microsoft SQL Server 2005에서 쿼리 최적화를 사용한 통계

Microsoft®

요약

Microsoft® SQL Server™ 2005는 데이터베이스에 저장된 인덱스와 열 데이터에 대한 통계 정보를 수집합니다. 이들 통계는 데이터 검색이나 업데이트를 위한 가장 효율적인 계획을 선택하기 위해 SQL Server 쿼리 최적화 프로그램에서 사용되고 있습니다. 이 백서에서는 데이터 수집, 저장되는 위치 및 통계를 생성하고 업데이트하고 삭제하는 명령을 설명합니다. 기본적으로 SQL Server 2005는 이러한 작업이 유용한 것으로 간주될 때 자동으로 통계를 생성하고 업데이트합니다. 이 백서는 이러한 기본값을 다른 수준(열, 테이블 및 데이터베이스)에서 변경하는 방법도 설명합니다.

Contents

SQL Server 2005의 통계 데이터

Microsoft® SQL Server™ 2005는 개별 열(단일 열 통계) 또는 여러 열(여러 열 통계)에 대한 통계를 수집합니다. 통계는 식의 선택성과 중간 및 최종 쿼리 결과의 크기를 예측하는 쿼리 최적화 프로그램에서 사용됩니다. 훌륭한 통계를 사용하면 최적화 프로그램이 여러 쿼리 계획의 비용을 정확하게 평가하고 고품질 계획을 선택할 수 있습니다. sysindexes 테이블에 있는 단일 행의 여러 열과 통계 바이너리 대형 개체(statblob)에 저장된 단일 통계 개체에 대한 모든 정보는 내부 전용 테이블에 유지됩니다. 또한 통계에 대한 정보는 새로운 메타데이터 보기 sys.stats 및 sys.indexes에서 볼 수 있습니다.

통계 기능의 요약

SQL Server 2005에는 통계를 유지 관리하는 많은 기능이 있습니다. 가장 중요한 기능은 통계를 자동으로 생성하고 업데이트하는 기능입니다. 이 기능은 기본적으로 SQL Server 2005 및 SQL Server 2000에 있습니다. SQL Server 2000 설치의 약 98%는 이 기능을 사용 상태로 유지하며 이것이 가장 좋은 방법입니다. 대부분의 데이터베이스와 응용 프로그램의 경우 개발자와 관리자들은 데이터에 대한 포괄적이고 정확한 통계를 제공하는 통계의 자동 생성과 업데이트 기능을 활용할 수 있습니다. 이를 통해 SQL Server 2005 쿼리 최적화 프로그램은 훌륭한 쿼리 계획을 일관되게 생성하는 한편, 개발과 관리 비용을 낮은 수준으로 유지합니다. 훌륭한 쿼리 계획을 확보하고 통계 수집의 오버헤드를 관리하는 통계 생성과 업데이트를 보다 철저하게 제어해야 한다면 수동 통계 생성과 업데이트 기능을 사용할 수 있습니다.

높은 처리 성능의 온라인 트랜잭션 처리 응용 프로그램 환경을 위한 중요한 새 기능은 자동으로 통계를 비동기적으로 업데이트하는 기능입니다. 이는 이러한 환경에서 쿼리 응답 시간의 예측성을 개선할 수 있습니다.

SQL Server 2005 통계 기능을 사용하여 다음을 수행할 수 있습니다.

- 기본 샘플링 속도로 통계를 암시적으로 생성 및 업데이트합니다.(SELECT, INSERT, DELETE 및 UPDATE 명령에서 WHERE 또는 JOIN 절 같은 쿼리 조건의 열을 사용하면 필요할 경우 자동 통계 생성과 업데이트를 사용할 때 통계가 생성되거나 업데이트됩니다)
- 원하는 샘플링 속도로 통계를 수동으로 생성 및 업데이트하고 통계를 삭제합니다.(CREATE STATISTICS, UPDATE STATISTICS, DROP STATISTICS, CREATE INDEX, DROP INDEX)
- 데이터베이스에서 모든 테이블의 모든 열에 대해 통계를 대량으로 수동 생성합니다.(sp_createstats)
- 데이터베이스에서 기존의 모든 통계를 수동으로 업데이트합니다.(sp_updatestats)
- 테이블이나 데이터베이스에 존재하는 통계 개체를 나열합니다.(sp_helpstats, 카탈로그 보기 sys.stats, sys.stats_columns)
- 통계 개체에 대한 설명 정보를 표시합니다.(DBCC SHOW_STATISTICS)
- 데이터베이스 범위 또는 특정 테이블이나 통계 개체에 대해 통계의 자동 생성과 업데이트를 사용 및 해제합니다.(ALTER DATABASE 옵션 AUTO_CREATE_STATISTICS 및 AUTO_UPDATE_STATISTICS, sp_autostats 및 CREATE STATISTICS 및 UPDATE STATISTICS에서 NORECOMPUTE 옵션)
- 통계의 비동기 자동 업데이트를 사용 및 해제합니다.(ALTER DATABASE 옵션 AUTO_UPDATE_STATISTICS_ASYNC)

또한 SQL Server Management Studio를 사용하면 Object Explorer 보기 내에서 통계 개체를 그래픽으로 탐색하고 제어할 수 있습니다. 통계는 Object Explorer에서 각 테이블 개체 아래의 폴더에 나열됩니다.

SQL Server 2005의 새 통계 기능

SQL Server 2005에는 쿼리 최적화 프로그램이 광범위한 쿼리에 대해 더 나은 쿼리 계획을 선택하거나 통계 관리를 개선할 수 있는 새로운 통계 기능들이 포함되어 있습니다. 여기에는 다음과 같은 기능 향상이 포함되어 있습니다.

- **문자열 요약 통계:** 하위 문자열의 빈번한 배포에 대한 정보는 문자열에 대해 유지됩니다. 이를 통해 최적화 프로그램이 LIKE 연산자를 사용하는 조건의 선택성을 보다 효과적을 예측할 수 있습니다.
- **통계 비동기 자동 업데이트**: ALTER DATABASE 옵션 AUTO_UPDATE_STATISTICS_ASYNC는 SQL Server 2005에서 새로운 기능으로 기본적으로 해제되어 있습니다. 이 옵션을 사용하면 SQL Server 2005가 백그라운드에서 통계의 자동 업데이트를 수행합니다. 통계 업데이트를 실행시키는 쿼리는 차단되지 않으며 기존 통계를 처리할 수 있습니다. 이는 일부 작업 로드에 대해 보다 예측 가능한 쿼리 응답 시간을 제공합니다.
- **계산된 열 통계**: 통계는 계산된 열에서 수동 또는 자동으로 생성하고 업데이트할 수 있습니다(이는 SQL Server 2000에서 부분적으로 지원되었지만 명문화되는 않았습니다).
- **대규모 개체 지원**: 열 유형 ntext, text, image, 뿐만 아니라 새 유형 nvarchar(max), varchar(max) 및 varbinary(max)를 통계 열로 지정할 수 있습니다.
- **개선된 통계 로딩 프레임워크**: 최적화 프로그램은 SQL Server 2000과 비교하여 통계의 로딩을 내부적으로 개선합니다. 이제 모든 통계와 필요한 통계만 로드하여 최적화 결과 품질과 성능을 개선합니다.
- **계산된 열에서 통계를 자동으로 생성하는 기능 개선**: 쿼리에 계산된 열의 식과 동일한 식이 포함되어 있다면 SQL Server 2005는 필요할 경우 계산된 열에서 통계를 자동으로 생성할 수 있습니다.
- **최소 샘플 크기**: 최소 8MB 크기의 데이터 또는 더 적은 경우 테이블 크기가 이제 통계 수집 동안 샘플링됩니다.
- **통계 숫자에 대한 제한 증가**: 테이블당 허용되는 열 통계 개체 숫자는 2,000으로 증가되었습니다. 테이블 당 최대 2,249 통계에 대해 추가 249 인덱스 통계가 제공될 수도 있습니다.
- **향상된 DBCC SHOW_STATISTICS 출력**: DBCC SHOW_STATISTICS는 이제 모호함을 방지하기 위해 표시되는 통계 개체의 이름을 표시합니다.
- **통계 자동 업데이트는 이제 열 수정 카운터를 기초로 이루어집니다**: SQL Server 2000에서 통계 업데이트는 행 변경 갯수에 따라 결정됩니다. 이제 변경은 열 수준에서 추적되며 통계의 자동 업데이트는 이제 통계 업데이트를 보장하는 데 충분한 만큼 변경되지 않은 열을 피할 수 있습니다.
- **내부 테이블에 대한 통계**: 통계는 XML과 전체 텍스트 인덱스, Service Broker 큐 및 쿼리 알림 테이블을 포함하여 sys.internal_tables에 나열된 테이블에서 완벽하게 지원됩니다.
- **DBCC SHOW_STATISTICS에 대한 단일 행 집합 출력**: DBCC SHOW_STATISTICS는 헤더, 밀도 벡터 및 히스토그램을 개별적으로 단일 행 집합으로 출력하는 옵션을 지원합니다. 이렇게 하면 DBCC SHOW_STATISTICS 출력을 자동으로 처리할 때 더 쉽게 프로그래밍할 수 있습니다.
- **최대 32열의 통계**: 통계 개체에서 통계 숫자에 대한 제한은 16에서 32까지 증가되었습니다.

- **파티션된 테이블의 통계**: 통계는 SQL Server 2005의 새로운 기능인 파티션된 테이블에서 완벽하게 지원됩니다. 히스토그램은 파티션 단위가 아닌 테이블 단위로 유지됩니다.
- **전체 스캔을 위해 수집되는 병렬 통계**: fullscan을 사용하여 수집된 통계의 경우 단일 통계 개체의 생성은 파티션되지 않았거나 파티션된 테이블 모두에 대해 병렬로 수행될 수 있습니다.
- **통계가 없는 경우 개선된 재컴파일 및 통계 생성**: 통계의 자동 생성이 실패한 경우 통계 없이 생성된 계획의 후속 실행에서 자동 생성이 수행되고 계획이 재컴파일되며 없는 통계 조건은 지속되지 않습니다. 자세한 내용은 Marathe의 백서 Batch Compilation, Recompilation, and Plan Caching Issues in SQL Server 2005를 참조하십시오.
- **빈 테이블에 대한 개선된 재컴파일 논리 및 통계 업데이트**: 테이블에서 0부터 0 이상의 행으로 변경되면 쿼리가 재컴파일되고 통계가 업데이트됩니다. 자세한 내용은 Marathe의 백서 Batch Compilation, Recompilation, and Plan Caching Issues in SQL Server 2005를 참조하십시오.
- **히스토그램의 보다 선명하고 일관성 있는 디스플레이**: DBCC SHOW_STATISTICS는 히스토그램이 카탈로그에 저장되기 전에 항상 확장되기 때문에 개선되었습니다.
- **유추 날짜 상관 관계 제약 조건**: DATE_CORRELATION_OPTIMIZATION 데이터베이스 설정을 사용하여 SQL Server가 외래 키로 링크된 테이블 쌍에서 datetime 필드의 상관 관계에 대한 정보를 유지할 수 있습니다. 이 정보는 일부 쿼리에 대한 암시적 단언을 결정할 수 있도록 사용됩니다. 정보는 최적화 프로그램에 의한 선택성 예측이나 비용을 위해 직접 사용되지 않기 때문에 엄격한 의미에서 통계는 아니지만 더 나은 쿼리 계획을 입수하는 데 사용되는 보조 정보이기 때문에 통계와 밀접하게 관련이 있습니다.
- **sp_updatestats**: Microsoft SQL Server 2005에서 sp_updatestats는 sys.sysindexes 호환성 보기의 rowmodctr 정보를 기반으로 업데이트하여 변경되지 않은 항목의 불필요한 업데이트를 제거하는 통계만 업데이트합니다. 호환성 수준이 90 이상으로 실행되는 데이터베이스의 경우 sp_updatestats는 특정 인덱스나 통계에 대한 자동 UPDATE STATISTICS 설정을 유지합니다.

또한 통계 작업에 몇 가지 사소한 변경도 있습니다. 특히, sys.sysindexes에서 statblob 열은 이제 항상 NULL로 제공됩니다. statblob 자체는 내부 전용 카탈로그 테이블에 유지됩니다.

정의

SQL Server 2005 통계와 관련된 다음 용어를 정의합니다.

- **statblob:** 통계 바이너리 대규모 개체. 이 개체는 내부 카탈로그 보기, sys.sysobjvalues에 있는 시스템 카탈로그에 저장됩니다.
- 문자열 요약: 문자열 열에 있는 하위 문자열의 빈번한 배포를 요약하는 통계 형태. 이것은 LIKE 조건부의 선택성을 예상하는 데 사용됩니다. 문자열에 대한 statblob에 저장됩니다.
- **sysindexes**: 테이블과 인덱스에 대한 정보가 들어 있는 sys.sysindexes 카탈로그 보기.
- **조건부**: true 또는 false를 평가하는 조건. 조건부는 데이터베이스 쿼리에서 WHERE 또는 JOIN 절에 나타납니다.
- **선택성**: 조건부를 만족하는 조건부의 입력 집합에서 행의 일부. 보다 정교한 선택성 평가가 joins, DISTINCT 및 다른 연산자에 의해 생성된 행 갯수를 예측하는 데도 사용됩니다. 예를 들어, SQL Server 2005는 AdventureWorks 데이터베이스에서 조건부 "Sales.SalesOrderHeader.OrderID = 43659" 의 선택성을 1/31465 = 0.00003178로 예측합니다.

- **카디널리티 예측**: 결과 집합의 크기 예측. 예를 들어, 테이블 T에 100,000개의 행이 있고 쿼리에 형태 T.a=10의 선택성 조건부가 포함되고 히스토그램은 T.a=10의 선택성이 10%임을 보여주고 쿼리에서 고려해야 하는 T의 행 일부에 대한 카디널리티 예측이 10% * 100,000 = 10,000인 것을 보여줍니다.
- **LOB**: 대규모 개체(유형 값 text, ntext, image, varchar(max), nvarchar(max), varbinary(max)).

SQL Server 2005에서 수집된 통계

SQL Server 2005는 테이블 수준에서 다음 정보를 유지합니다. 이것은 통계 개체의 일부가 아니지만 SQL Server 2005는 쿼리 비용 예측 동안 일부 경우에서 사용합니다.

- 테이블 또는 인덱스에서 행의 갯수(sys.sysindexes에서 rows 열).
- 테이블 또는 인덱스가 점유하는 페이지 수(sys.sysindexes에서 dpages 열).

SQL Server 2005는 테이블 열에 대한 다음 통계를 수집하고 통계 개체(statblob)에 저장합니다.

- 통계를 수집한 시간.
- 히스토그램과 밀도 정보를 생성하는 데 사용된 행 수(아래에서 설명).
- 평균 키 길이.
- 단계 수를 포함하여 단일 열 히스토그램.
- 열에 문자 데이터가 포함된 경우 문자열 요약. DBCC SHOW_STATISTICS 출력에는 통계 개체에 문자열 요약이 포함된 경우 값이 YES인 "문자열 인덱스"가 포함되어 있습니다.

히스토그램은 주어진 열의 최대 200개 값이 있는 집합입니다. 주어진 열에 있는 값 전부 또는 샘플이 저장됩니다. 대부분 통계적으로 중요한 정보가 캡처되도록 순서는 최대 199개 간격으로 나누어집니다. 일반적으로 이러한 간격은 크기가 동일하지 않습니다. 다음 값 또는 이러한 값을 추출하는 데 충분한 정보가 히스토그램의 각 단계에 저장되어 있습니다.

RANGE_HI_KEY	히스토그램 단계의 상한을 보여주는 키 값
RANGE_ROWS	범위 내에 포함되는 행의 수를 지정합니다(이 RANGE_HI_KEY보다 작지만 이전의 작은 RANGE_HI_KEY보다는 큽니다).
EQ_ROWS	RANGE_HI_KEY와 정확하게 동일한 행의 수를 지정합니다.
AVG_RANGE_ROWS	범위 내 고유 값당 행의 평균 개수.
DISTINCT_RANGE_ROWS	범위 내 고유 키 값의 수를 지정합니다(RANGE_HI_KEY 및 RANGE_HI_KEY 이전의 키는 제외)

SQL Server 2005의 히스토그램은 통계 개체의 키 열 집합에서 첫 번째 열인 단일 열에 대해서만 생성됩니다.

SQL Server 2005는 세 단계로 열 값의 저장된 집합으로부터 히스토그램을 생성합니다.

- **히스토그램 초기화**: 첫 번째 단계에서 저장된 집합의 처음에서 시작하는 값의 순서가 처리되고 RANGE_HI_KEY, EQ_ROWS, RANGE_ROWS 및 DISTINCT_RANGE_ROWS의 최대 200개 값이 수집됩니다(RANGE_ROWS 및 DISTINCT_RANGE_ROWS는 이 단계 동안 항상 0입니다). 첫 번째 단계는 모든 입력이 소모되었거나 200개의 값이 발견되었을 때 종료됩니다.
- **버킷 병합을 사용한 스캔**: 통계 키에서 주요 열의 각 추가 값이 두 번째 단계에서 정렬된 순서대로 처리됩니다. 끝에서 마지막 범위나 새 범위에 추가된 각 연속 값이 생성됩니다(이는 입력 값이 정렬되었기 때문에 가능합니다). 새 값이 생성된 경우 한 쌍의 기존 인접 범위는 단일 범위로 축소됩니다. 이 범위 쌍은 정보 손실을 최소화하도록 선택됩니다. 범위를 축소한 후 단계 수는 이 단계 전체에서 200으로 유지됩니다. 이 방법은 maxdiff 히스토그램의 변형을 기반으로 합니다.
- **히스토그램 통합**: 세 번째 단계에서 많은 양의 정보가 손실된 경우 추가 범위가 축소될 수 있습니다. 따라서 열에 200개 이상의 고유한 값이 있는 경우에도 히스토그램은 200개 미만의 단계를 가질 수 있습니다.

히스토그램이 샘플을 사용하여 생성된 경우 RANGE_ROWS, EQ_ROWS, DISTINCT_RANGE_ROWS 및 AVG_RANGE_ROWS의 값은 예측되기 때문에 정수가 될 필요는 없습니다.

밀도는 주어진 열 또는 열 조합에서 중복 수에 대한 정보이며 1/(개별 값 수)로 계산됩니다. 열이 동등 조건부에서 사용되면 정규화 열의 개수는 히스토그램에서 추출되는 밀도를 사용하여 예측됩니다. 히스토그램은 동등하지 않은 선택성 조건부, 조인 및 기타 연산자의 선택성을 예측하는 데도 사용됩니다.

통계를 수집한 시간을 보여주는 타임 스탬프, 테이블에 있는 행 개수, 히스토그램을 생성하기 위해 샘플링된 행 개수, 밀도 정보와 평균 키 길이 및 히스토그램 자체 외에 단일 열 통계 정보에는 통계 열 집합의 접미사를 구성하는 각 열 집합에 대한 All density 값이 포함되어 있습니다. 이는 DBCC SHOW_STATISTICS의 두 번째 행 집합 출력에서 나타납니다. All density는 1/(접미사 열 집합에서 고유한 값 개수)의 예상치입니다. 다음 섹션에서 이와 같은 예제를 제공합니다.

주: dbcc show_statistics에서 반환된 첫 번째 행에 포함된 Density 값은 RANGE_HI_KEY 값 이외의 샘플링된 모든 값의 밀도입니다. RANGE_HI_KEY 값은 일반적으로 배포에서 더 많은 빈도 값입니다. 그러기 때문에 표시된 밀도는 빈번하지 않은 값의 밀도에 대한 유용한 정보를 제공할 수 있습니다.

하나의 열 집합에 대한 여러 열의 통계는 통계 정의에서 첫 번째 열에 대한 하나의 히스토그램, 첫 번째 열에 대한 하나의 밀도 값 및 첫 번째 열을 포함하여 열의 각 접미사 조합에 대한 All Density 값으로 구성되어 있습니다. 여러 열 통계의 각 집합(히스토그램과 둘 이상의 밀도 값)은 마지막 통계 업데이트, 통계 정보를 생성하는 데 사용되는 샘플에 있는 행 개수, 히스토그램에서 단계 수 및 키의 평균 길이와 함께 한 statblob에 저장됩니다. 문자열 요약은 문자 데이터가 포함된 경우에만 첫 번째 열에 대해서만 포함됩니다.

sp_helpindex 및 sp_helpstats를 사용하여 주어진 테이블에 대한 모든 통계 목록을 표시할 수 있습니다. sp_helpindex는 테이블의 모든 인덱스를 나열하고 sp_helpstats는 테이블의 모든 통계를 나열합니다. 또한 각 인덱스는 이 열에 대한 통계 정보를 제공합니다. CREATE STATISTICS 명령을 사용하여 생성한 통계 정보는 같은 열에서 CREATE INDEX 명령으로 생성한 통계와 같습니다. 유일한 차이는 CREATE STATISTICS 명령이 기본적으로 샘플링을 사용하는 반면 CREATE INDEX 명령은 어쨌든 인덱스에 대한 모든 행을 처리해야 하기 때문에 fullscan을 사용하여 통계를 수집합니다.

통계 생성 및 표시: 예제

다음 예제는 자동이나 수동으로 통계가 생성되는 방법을 나타내고 통계에 대한 정보를 나열하고 표시하는 방법을 보여 줍니다. 모든 명령이 아닌 일부에 대해 결과가 제공됩니다. SQL Server 2005에서 생성되는 출력은 동작을 설명하는 데 유용할 때 표시됩니다. 이 예제를 직접 실행하여 완전한 출력을 볼 수 있습니다.

```
USE tempdb
GO
-- Clean up objects from any previous runs.
IF object_id(N' Person.Contact' ,' U' ) IS NOT NULL
        DROP TABLE Person.Contact
GO
IF EXISTS (SELECT * FROM sys.schemas WHERE name = N' Person' )
        DROP SCHEMA Person
GO
-- Create a sample schema and table.
CREATE SCHEMA Person
GO
CREATE TABLE Person.Contact(
        FirstName nvarchar(60),
        LastName nvarchar(60),
        Phone nvarchar(15),
        Title nvarchar(15)
)
GO
-- Populate the table with a few rows.
INSERT INTO Person.Contact
  VALUES(N' James' ,N' Smith' ,N' 425-555-1234' ,N' Mr' )
INSERT INTO Person.Contact
  VALUES(N' James' ,N' Andersen' ,N' 425-555-1111' ,N' Mr' )
INSERT INTO Person.Contact
  VALUES(N' James' ,N' Andersen' ,N' 425-555-3333' ,N' Mr' )
INSERT INTO Person.Contact
  VALUES(N' Christine' ,N' Williams' ,N' 425-555-0000' ,N' Dr' )
INSERT INTO Person.Contact
  VALUES(N' Susan' ,N' Zhang' ,N' 425-555-2222' ,N' Ms' )
GO
-- Show that there are no statistics yet on the Person.Contact table.
sp_helpstats N' Person.Contact' ,  'ALL'
GO
-- Implicitly create statistics on LastName.
SELECT * FROM Person.Contact WHERE LastName = N' Andersen'
GO
```

```
-- Show that statistics were automatically created on LastName.
sp_helpstats N' Person.Contact' , 'ALL'
GO
```

결과:

statistics_name	statistics_keys
_WA_Sys_00000002_1B29035F	LastName

```
-- Create an index, which also creates statistics.
CREATE NONCLUSTERED INDEX Phone on Person.Contact(Phone)
GO
-- Show that creating the index created an associated statistics object.
sp_helpstats N' Person.Contact' , 'ALL'
GO
```

결과:

statistics_name	statistics_keys
_WA_Sys_00000002_1B29035F	LastName
Phone	Phone

```
-- Create a multi-column statistics object on first and last name.
CREATE STATISTICS FirstLast ON Person.Contact(FirstName,LastName)
GO
-- Show that there are now three statistics objects on the table.
sp_helpstats N' Person.Contact' , 'ALL'
GO
```

결과:

statistics_name	statistics_keys
_WA_Sys_00000002_1B29035F	LastName
FirstLast	FirstName, LastName
Phone	Phone

```
-- Display the statistics for LastName.
DBCC SHOW_STATISTICS (N' Person.Contact' , LastName)
GO
```

결과:

통계 헤더 정보:

이름	업데이트 내용	행	샘플링된 행	단계	밀도	평균 키 길이	문자열 인덱스
_WA_Sys_ 00000002_ 1B29035F	Mar 25 2005 11:21AM	5	5	4	0	13.6	예.

열 집합 접미사 및 관련 밀도와 길이:

All Density	평균 길이	열
0.25	13.6	LastName

히스토그램 단계:

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
Andersen	0	2	0	1
Smith	0	1	0	1
Williams	0	1	0	1
Zhang	0	1	0	1

```
-- If you take the name of the statistics object displayed by
-- the command above and subsitute it in as the second argument of
-- DBCC SHOW_STATISTICS you can form a command like the following one
--(the exact name of the automatically created statistics object
-- will typically be different for you).

DBCC SHOW_STATISTICS (N' Person.Contact' , _WA_Sys_00000002_2D7CBDC4)

-- Executing the above command illustrates that you can show statistics by
-- column name or statistics object name.
GO
-- The following displays multi-column statistics. Notice the two
-- different density groups for the second rowset in the output.
DBCC SHOW_STATISTICS (N' Person.Contact' , FirstLast)
```

결과(두 번째 행 집합만 해당):

열 집합 접미사 및 관련 밀도와 길이:

All Density	평균 길이	열
0.3333333	11.6	FirstName
0.25	25.2	FirstName, LastName

대규모 테이블에 대해 완벽하게 채워진 히스토그램을 보려면 다음 명령을 실행하십시오.

```
USE AdventureWorks
-- Clean up objects from previous runs.
IF EXISTS (SELECT * FROM sys.stats
        WHERE object_id = object_id( 'Sales.SalesOrderHeader' )
        AND name =  'TotalDue' )
       DROP STATISTICS Sales.SalesOrderHeader.TotalDue
GO
CREATE STATISTICS TotalDue ON Sales.SalesOrderHeader(TotalDue)
GO
DBCC SHOW_STATISTICS(N' Sales.SalesOrderHeader' , TotalDue)
```

SQL Server 2005를 이용한 통계 생성

아래의 설명과 같이 여러 가지 방법으로 SQL Server 2005에서 통계를 생성할 수 있습니다.

- 최적화 프로그램은 AUTO_CREATE_STATISTICS가 기본적으로 설정되어 있을 때 SELECT, INSERT, UPDATE 및 DELETE 문을 최적화할 때의 부수적인 결과로 필요할 때 단일 열 통계를 자동으로 생성합니다.
- SQL Server 2005에서는 위에서 설명하는 통계 정보를 명시적으로 생성하는 기본 문이 두 가지가 있습니다. CREATE INDEX는 첫 번째 위치에 선언된 인덱스를 생성하고 인덱스 키를 구성하는 열 조합(다른 포함된 열은 제외)에 대해 하나의 통계 집합도 생성합니다. CREATE STATISTICS는 주어진 열이나 열 조합에 대한 통계만 생성합니다.
- 또한 통계나 인덱스를 생성하는 여러 가지 방법이 있습니다. 궁극적으로 각각 위의 두 명령 중 하나를 실행합니다. sp_createstats를 사용하여 현재 데이터베이스에 있는 모든 사용자 테이블에 대해(XML 열을 제외하고) 모든 자격 있는 열에 대한 통계를 생성합니다. 이미 통계 개체가 있는 열에 대한 새로운 통계 개체는 생성되지 않습니다.
- dbcc dbreindex를 사용하여 지정한 데이터베이스에 있는 테이블에 대해 하나 이상의 인덱스를 다시 생성합니다.
- Management Studio에서 Table 개체 아래의 폴더를 확장하고 Statistics 폴더를 마우스 오른쪽 단추로 클릭하고 New Statistics를 선택합니다.
- Database Tuning Advisor(DTA)를 사용하여 인덱스를 생성합니다.

AdventureWorks.Person.Contact 테이블에 대한 CREATE STATISTICS 명령의 예제는 아래와 같습니다.

```
CREATE STATISTICS FirstLast2 ON Person.Contact(FirstName,LastName)
WITH SAMPLE 50 PERCENT
```

대개 기본 샘플링을 가진 통계는 좋은 쿼리 계획을 생성하는 데 충분합니다. 그러나 주어진 열 샘플의 값이 임의적이 아닐 때처럼 더 많은 샘플 크기가 있는 통계가 쿼리 최적화를 이용할 수 있을 때가 있습니다. 데이터가 정렬되거나 클러스터링된 경우 임의적이 아닌 샘플이 발생할 수 있습니다. 정렬 또는 클러스터링은 인덱스 생성 또는 이미 정렬되었거나 클러스터링된 힙 구조로 데이터를 로드하는 것으로 인해 발생할 수 있습니다. 가장 일반적으로 사용되는 더 큰 샘플 크기는 가장 정확한 통계를 제공하기 때문에 fullscan입니다. 더 큰 샘플 크기를 가진 통계를 사용하는 비용은 통계를 생성하는 데 필요한 시간입니다.

위 명령은 단일의 두 열 통계 개체를 생성합니다. 이 경우 SAMPLE 50 PERCENT는 무시되며 테이블이 너무 작기 때문에 전체 스캔이 수행됩니다. 샘플링은 주로 데이터를 과도하게 스캔하는 것을 방지하고 1,024 이상의 페이지(8MB)가 있는 테이블과 인덱스에만 영향을 미치도록 하는 데 사용됩니다.

SQL Server 2005에서 통계는 인덱스 생성 시간에 모든 인덱스에 대해 생성됩니다. SQL Server는 쿼리를 컴파일할 때 단일 열 통계를 자동으로 생성합니다. 이러한 통계는 최적화 프로그램이 대략적인 밀도나 배포를 예측해야 하는 열에 대해 생성됩니다. 이 규칙에는 다음과 같은 예외가 있습니다. 통계는 (1) 데이터베이스가 읽기 전용이고, (2) 진행 중인 컴파일 작업이 너무 많고, (3) 열의 데이터 유형이 자동 시작에 대해 지원되지 않을 때는 자동으로 생성되지 않을 수 있습니다.

자동 통계 생성 기능은 다음 명령을 실행하여 데이터베이스 수준에서 사용되지 않을 수 있습니다.

```
ALTER DATABASE dbname SET AUTO_CREATE_STATISTICS OFF
```

마찬가지로 자동 통계 생성은 다음 명령을 실행하여 데이터베이스 수준에서 사용될 수 있습니다.

```
ALTER DATABASE dbname SET AUTO_CREATE_STATISTICS ON
```

이 설정을 ON으로 두는 것이 좋습니다. 일부 테이블의 기본 샘플링 속도 이외의 다른 값을 지정해야 하는 경우처럼 알려진 성능 문제를 해결하기 위해 수행해야 하는 경우에만 사용하지 마십시오.

기본적으로 통계는 CREATE STATISTICS 명령을 실행하거나 통계를 자동으로 생성해야 할 때 데이터 집합을 샘플링하여 생성됩니다. CREATE INDEX는 전체 데이터 집합을 스캔하기 때문에 인덱스 통계는 초기에는 샘플링 없이(fullscan과 동일) 생성됩니다. CREATE STATISTICS 명령을 사용하면 fullscan 또는 데이터의 비율이나 스캔할 행의 개수를 지정하여 WITH 절에서 샘플 크기를 설정할 수 있습니다. 후자는 근사치로 해석됩니다. 또한 UPDATE STATISTICS 명령에서 WITH RESAMPLE을 지정할 때 이전 샘플 크기에도 상속될 수 있습니다. 이는 일부 열 또는 열 집합(원래 fullscan 통계를 사용하여 생성)에 인덱스가 있고 일부 다른 열 또는 열 집합(원래 SAMPLE 통계를 사용하여 생성)의 통계만 있을 때 특히 유용합니다. UPDATE STATISTICS에서 resample 옵션을 사용하면 나머지 열에 대해 인덱스와 샘플 통계를 위한 fullscan 통계를 유지합니다.

작은 테이블의 경우 최소 8MB의 데이터가 샘플링됩니다. 테이블을 작게 시작하고 기본 샘플링 속도로 샘플링한 후에 통계를 업데이트할 때 resample 옵션을 사용하는 경우 테이블이 8MB를 초과하여 성장할 때도 fullscan의 동일성을 확보하게 됩니다. 테이블 크기가 변화할 때 기본 샘플링 속도를 원하는 경우 resample을 사용하지 마십시오.

resample 샘플링 속도는 이전 통계 계산 동안 샘플링된 행의 개수와 테이블에 있는 행의 총 개수의 함수로 계산됩니다. 실제 샘플링 속도는 샘플링의 임의 특성으로 인해 다를 수 있기 때문에 resample 속도는 전체 스캔이 아닌 샘플에 대한 이전 샘플링 속도의 근사치일 뿐입니다. 일관성 있게 반복 가능한 샘플링의 경우 resample을 사용하는 대신 UPDATE STATISTICS를 사용하기 전에 사용하는 것과 같은 샘플링 속도를 명시적으로 지정하십시오.

dbcc show_statistics 명령은 Rows Sampled 머리글 아래에 샘플 크기를 표시합니다. 다음 절에서 설명하는 것처럼 자동으로 생성되었거나 업데이트된 통계는 항상 기본 샘플링을 사용하여 생성됩니다. 기본 샘플링 속도는 테이블 크기의 느리게 성장하는 함수입니다. 따라서 매우 큰 테이블의 경우에도 상대적으로 빨리 통계를 수집할 수 있습니다.

통계를 생성하고 업데이트할 때 최적화 프로그램은 통계를 수집하는 액세스 경로를 선택해야 합니다. 액세스 경로는 힙, 클러스터링된 인덱스 또는 클러스터링되지 않은 인덱스를 포함할 수 있습니다. 샘플링된 통계의 경우 최적화 프로그램은 주요 통계 키 열에 물리적으로 정렬된 액세스 경로를 피하려고 합니다. 이는 더 많은 임의 샘플을 제공하여 보다 정확한 통계로 이끕니다. 통계 키에 정렬되지 않은 액세스 경로의 경우(액세스 경로가 존재하는 경우) 가장 낮은 비용의 액세스 경로가 선택됩니다. 이는 가장 좁은 인덱스 또는 힙입니다. fullscan 통계인 경우 액세스 경로의 정렬 순서는 통계 정확도에 중요하지 않기 때문에 가장 낮은 비용의 액세스 경로가 선택됩니다.

SQL Server Profiler는 자동 통계 생성을 모니터링할 수 있습니다. Auto Stats 이벤트는 Performance 추적 이벤트의 그룹에 포함되어 있습니다. 추적을 정의할 때 Auto Stats 이벤트를 위한 IntegerData, Success 및 ObjectID 열도 선택합니다. AutoStats 이벤트를 캡처한 후에 Integer Data 열에는 주어진 테이블에 대해 생성되었거나 업데이트된 통계 개수가 포함되며 개체 ID는 테이블의 ID이고 TextData 열(기본적으로 추적 정의에 포함)에는 Created: 또는 Updated:접미사와 함께 통계 생성이나 업데이트를 수행할 열의 이름이 포함되어 있습니다. Success 열에는 Auto Stats 작업의 성공이나 실패를 표시하는 내용이 포함되어 있습니다. 특히 성공에는 세 가지 가능한 값이 있습니다.

이름	값	정의
FAILED	0	THROTTLED 이외의 이유로 Auto Stats 생성 또는 업데이트가 실패했습니다(아래 참조). 예를 들어, 데이터베이스가 읽기 전용이었습니다.
SUCCESS	1	Auto Stats 생성 또는 업데이트가 성공했습니다.
THROTTLED	2	너무 많은 최적화가 진행 중이어서 Auto Stats 생성 또는 업데이트가 실패했습니다.

경우에 따라 통계가 생성되었거나 업데이트되지 않은 경우 AutoStats 이벤트를 관찰할 수도 있습니다. 이런 이벤트는 auto update statistics가 해제되었거나 쿼리에서 참조하는 테이블이 많이 변경되었고 쿼리 구조와 외래 키 제약 조건의 존재로 인해 최적화 프로그램이 쿼리 계획에서 해당 테이블에 대한 모든 참조를 제거할 수 있을 때 생성됩니다.

DROP STATISTICS 명령은 통계를 삭제하는 데 사용되지만 인덱스의 부산물인 통계를 삭제할 수는 없습니다. 이런 통계는 인덱스가 삭제되었을 때만 제거됩니다.

SQL Server 2005의 통계 유지 관리

테이블에서 일련의 INSERT, DELETE 및/또는 UPDATE를 수행한 후에 통계는 주어진 열이나 인덱스에 진정한 데이터 배포를 반영하지 못할 수 있습니다. SQL Server 쿼리 최적화 프로그램이 통계를 마지막으로 생성했거나 업데이트한 이후로 업데이트 작업이 진행 중인 테이블에서 특정 열에 대한 통계를 요구하는 경우 SQL Server는 열 값을 샘플링하여(auto update statistics 사용) 통계를 자동으로 업데이트합니다. 통계 자동 업데이트는 쿼리 최적화나 컴파일된 계획을 실행하여 트리거되며 쿼리에서 참조하는 열의 하위 집합만 포함됩니다. 통계는 AUTO_UPDATE_STATISTCS_ASYNC가 OFF인 경우 쿼리를 컴파일하기 전에 업데이트되며 ON인 경우 비동기적으로 업데이트됩니다.

쿼리를 처음 컴파일할 때 최적화 프로그램이 특정 통계 개체를 요구하고 해당 통계 개체가 존재하는 경우 통계 개체가 오래되었으면 업데이트됩니다. 쿼리가 실행되고 계획이 캐시에 있을 때 계획이 사용하는 통계가 오래되지 않았는지 검사합니다. 오래되었으면 캐시에서 계획이 제거되고 쿼리를 재컴파일하는 동안 통계가 업데이트됩니다. 또한 사용하는 통계가 변경된 경우에도 계획은 캐시에서 제거됩니다.

SQL Server 2005는 column modification counters(colmodctrs)의 변경을 기반으로 통계를 업데이트할지 여부를 결정합니다.

통계 개체는 다음과 같은 경우에 오래된 것으로 간주됩니다.

- 통계가 일반 테이블에 정의된 경우 다음과 같을 때 오래된 것입니다.
  1. 테이블 크기가 0에서 0개 이상의 행을 갖게 된 경우
  2. 통계를 수집했을 때 테이블에 있는 행 개수가 500 미만이었고 통계 개체의 주요 열의 colmodctr이 그 이후로 500 이상 변경된 경우
  3. 통계를 수집했을 때 테이블에 500개 이상의 행이 있었고 통계 개체의 주요 열의 colmodctr이 통계를 수집했을 때 테이블에 있는 행 수의 500 + 20% 이상 변경된 경우

- 통계 개체가 임시 테이블에 정의된 경우 앞의 목록에서 테스트 2와 동일한 테이블을 가지고 6개의 행에서 재컴파일을 위한 추가 임계값이 있는 경우를 제외하고는 위에서 설명한 것처럼 오래된 것입니다.

테이블 변수에는 전혀 통계가 없습니다.

위에서 설명한 auto update statistics 기능은 다른 수준에서 꺼져 있을 수 있습니다.

- 데이터베이스 수준에서 다음 명령을 사용하여 자동 업데이트 통계를 해제하십시오.

```
ALTER DATABASE dbname SET AUTO_UPDATE_STATISTICS OFF
```

- 테이블 수준에서 UPDATE STATISTICS 명령 또는 CREATE STATISTICS 명령의 NORECOMPUTE 옵션을 사용하여 자동 업데이트 통계를 해제합니다.
- sp_autostats를 사용하여 테이블, 인덱스 또는 통계 개체에 대한 자동 업데이트 통계를 표시하고 변경합니다.

통계 자동 업데이트를 다시 사용하려면 ALTER DATABASE, UPDATE STATISTICS 또는 sp_autostats를 사용하여 비슷하게 실행할 수 있습니다.

SQL Server 2005는 데이터베이스, 테이블 및 인덱스나 통계 개체 수준에서 자동 통계 업데이트 설정을 유지합니다. 단일 sp_autostats 명령을 사용하여 한 테이블에서 모든 테이블에 대해 이 설정을 변경할 수 있지만 이는 주어진 테이블에서 각 통계 개체와 인덱스에 대한 설정을 개별적으로 변경하여 구현됩니다. 전체 테이블에 대해 자동 업데이트 통계가 ON이거나 OFF라는 것을 메타데이터는 명시적으로 기록하지 않습니다.

다음 표는 여러 데이터베이스, 테이블 및 인덱스 설정의 복합적인 영향을 보여 주고 있습니다.

데이터베이스 설정	인덱스 또는 통계 개체 설정	통계 자동 업데이트를 개체에 적용
ON	ON	ON
ON	OFF	OFF
OFF	ON	OFF
OFF	OFF	OFF

통계 개체 수준에서 ON으로 설정하여 자동 업데이트 통계에 대한 데이터베이스 설정 OFF를 무시할 수 없습니다.

자동 통계 업데이트는 기본 샘플링 속도를 사용하여 인덱스나 테이블을 샘플링함으로써 항상 수행됩니다. 샘플링 속도를 명시적으로 설정하려면 CREATE 또는 UPDATE STATISTICS를 실행하십시오.

통계 업데이트는 통계 생성과 같은 SQL Profiler 이벤트에서 다룹니다.

문자열 요약 통계

SQL Server 2005에는 LIKE 조건의 선택성을 예측하는 특허 기술이 포함되어 있습니다. 이 기술은 문자 열(문자열 요약)의 하위 문자열 주기 배포를 위한 통계 요약을 생성합니다. 여기에는 열 유형 text, ntext, char, varchar 및 nvarchar가 포함됩니다. 문자열 요약을 사용하여 SQL Server는 패턴이 모든 조합에서 많은 와일드카드를 가질 수 있는 LIKE 조건의 선택성을 정확하게 예측할 수 있습니다. 예를 들어, SQL Server는 다음 형태의 조건부 선택성을 예측할 수 있습니다.

```
Column LIKE 'string%'
Column LIKE '%string'
Column LIKE '%string%'
Column LIKE 'string'
Column LIKE 'str_ing'
Column LIKE 'str[abc]ing'
Column LIKE '%abc%xy'
```

사용자 정의 이스케이프 문자가 LIKE 패턴에 있을 경우(즉, 패턴이 LIKE *pattern ESCAPE escape_character* 형태) SQL Server 2005는 선택성을 추측합니다.

이는 후행 와일드카드 % 이외의 와일드카드가 LIKE 패턴에 사용될 때 선택성을 추측해야 하며 이와 같은 경우에 예측의 정확도를 제한하는 SQL Server 2000보다 향상된 기능입니다.

DBCC SHOW_STATISTICS에 의해 반환된 첫 번째 행 집합의 String Index 필드는 통계 개체가 문자열 요약도 포함하는 경우 값 YES를 포함합니다. 문자열 요약의 내용은 표시되지 않습니다. 문자열 요약은 히스토그램에 표시된 것 이외의 추가 정보를 포함하고 있습니다.

80자 이상의 긴 문자열의 경우 첫 번째와 마지막 40문자가 문자열에서 추출되고 문자열 요약 작성 시 문자열로 간주되기 전에 연결됩니다. 따라서 사용할 수 없는 문자열의 무시된 부분에만 나타내는 하위 문자열에 대한 정확한 빈도를 예측합니다.

계산된 열의 통계

SQL Server 2005는 쿼리에 이름 순의 계산된 열이 포함되어 있지 않고 계산된 열 식을 포함하고 있을 때도 계산된 열에 대한 통계의 생성, 업데이트 및 사용을 지원합니다. SQL Server 2000은 열이 쿼리에서 명시적으로 지정된 경우에만 계산된 열에 대한 통계를 자동으로 생성, 업데이트 및 사용할 수 있습니다.

SQL Server 2005에서 다음 Transact-SQL 스크립트를 실행하는 경우 AdventureWorks 데이터베이스 테이블 열 Sales.SalesOrderHeader.TotalDue에 대한 계산된 열 통계의 자동 생성을 관찰할 수 있습니다.

```
USE AdventureWorks
GO
-- Remove all statistics for Sales.SalesOrderHeader
DECLARE c CURSOR FOR
SELECT name FROM sys.stats
WHERE object_id = object_id( 'Sales.SalesOrderHeader' )
AND auto_created <> 0 AND user_created <> 0

DECLARE @name NVARCHAR(255)

OPEN c

FETCH next FROM c INTO @name

WHILE @@FETCH_STATUS = 0
BEGIN
        EXEC ( 'drop statistics Sales.SalesOrderHeader.' + @name)
```

```
        FETCH NEXT FROM c INTO @name
END

CLOSE c
DEALLOCATE c

-- Query Sales.SalesOrderHeader with an expression equivalent
-- to the TotalDue computed column,
-- ((isnull(([SubTotal]+[TaxAmt])+[Freight],(0))).
SELECT *
FROM Sales.SalesOrderHeader h, Sales.SalesOrderDetail d
WHERE h.SalesOrderID = d.SalesOrderID
AND (isnull(([SubTotal]+[TaxAmt])+[Freight],(0))) > 200000.00
ORDER BY TotalDue DESC

-- List the created statistics. Observe that statistics
-- are created for TotalDue even though it is not explicitly
-- referenced in the query.
sp_helpstats  'Sales.SalesOrderHeader'
```

SQL Server 2005는 CLR 사용자 정의 함수의 실행 같이 CLR(공용 언어 런타임) 식을 포함하는 지속되지 않는 계산된 열에 대한 통계는 지원하지 않습니다. CLR 계산된 열에 대한 통계를 생성하려면 열은 PERSISTED로 표시되어야 합니다.

사용자 정의 유형 열에 대한 통계

SQL Server 2005는 유형이 바이너리 순서를 지원하는 경우 CLR 사용자 정의 유형 열에 대한 통계의 생성, 업데이트 및 사용을 지원합니다. 통계는 바이너리 순서를 지원하지 않는 사용자 정의 유형은 지원하지 않습니다. 유형은 IsByteOrdered 플래그가 유형 정의의 일부로 지정된 SqlUserDefinedType 특성에서 true로 설정된 경우 바이너리 순서입니다. 유형이 바이너리 순서를 지원한다면 표준 바이너리 정렬 순서가 유형에 대해 문법적으로 올바른 순서라는 것을 의미합니다.

통계 및 인덱스 보기

일반적으로 통계는 인덱스된 보기에서는 요구되지 않습니다. 이는 인덱스된 보기를 쿼리 계획으로 대체하는 것이 기본 테이블과 인덱스의 모든 통계가 쿼리 계획에 연결된 후에만 고려되기 때문입니다. 한 가지 예외는 있습니다. 보기가 NOEXPAND 힌트를 사용하여 FROM 절에서 직접 참조되는 경우 통계는 사용됩니다. NOEXPAND 힌트가 인덱스가 없는 보기에 사용되는 경우에는 오류가 생성되고 계획은 작성되지 않습니다.

제한된 사용으로 인해 인덱스된 보기의 통계는 sp_createstats를 사용하여 생성되거나 sp_updatestats를 사용하여 업데이트되지 않습니다. auto update 및 auto create statistics 기능은 인덱스된 보기에 작동하지 않습니다. 그러나 이전에 언급한 것처럼 이런 통계는 최적화 프로그램에 필요하기 때문에 인덱스된 보기가 쿼리의 NOEXPAND 힌트와 함께 사용되고 자동 업데이트/생성 통계 옵션이 ON 상태인 경우에만 생성됩니다. 또한 인덱스된 보기 열에서 CREATE STATISTICS를 수동으로 수행하거나 UPDATE STATISTICS를 사용하여 인덱스된 보기에서 열이나 인덱스 통계를 업데이트할 수도 있습니다.

통계 관리를 위한 최상의 방법

SQL Server에서 통계를 사용하는 목적은 최적화 프로그램이 훌륭한 쿼리 계획을 찾을 수 있도록 우수한 카디널리티를 입수하는 동시에, 통계의 타당성 수집과 관련된 오버헤드와 지연을 유지하는 것입니다. 아래에 가장 중요한 순서대로 SQL Server에서 통계를 관리하는 최상의 방법이 제시되어 있습니다.

자동 생성 및 자동 업데이트 통계 사용

대부분의 SQL Server 설치의 경우 가장 중요한 방법은 데이터베이스 차원의 자동 생성과 자동 업데이트 통계를 사용하는 것입니다. 자동 생성과 자동 업데이트 통계는 기본적으로 설정되어 있습니다. 잘못된 계획을 관찰하여 없거나 오래된 통계가 문제라고 의심된다면, 자동 생성과 자동 업데이트 통계가 설정되어 있는지 확인하십시오.

필요할 경우 FULLSCAN 통계를 선택적으로 사용

자동 생성과 자동 업데이트 통계를 사용 중이고 정확하지 않거나 최신이 아닌 통계로 인해 잘못된 쿼리 계획을 얻는 경우 다음을 수행하십시오.

- 자동 생성과 자동 업데이트 통계를 ON 상태로 둡니다.
- 정확하지 않거나 최신이 아닌 통계에만 다음을 사용합니다.

```
CREATE STATISTICS ... WITH FULLSCAN, NORECOMPUTE
```

배치 작업에는 정기적으로 다음을 사용합니다.

```
UPDATE STATISTICS ... WITH FULLSCAN, NORECOMPUTE
```

통계를 업데이트하는 주기는 응용 프로그램에 따라 다르며 몇 가지 실험이 필요할 수 있습니다. fullscan 업데이트 주기의 좋은 출발점은 고려하고 있는 테이블이 높은 업데이트 속도를 가진 경우 fullscan 통계 업데이트를 야간에 실행하는 것입니다. 테이블이 느린 업데이트 속도를 가진 경우 fullscan 통계 업데이트를 매주 실행하십시오.

쿼리에 로컬 변수 사용 자제

매개 변수나 문자 대신 쿼리 조건부에 로컬 변수를 사용하는 경우 최적화 프로그램은 낮은 품질의 예측치를 갖게 되거나 조건부 선택성을 예측에 의존하게 됩니다. 로컬 변수 대신 쿼리에 매개 변수나 문자를 사용해야 하며 최적화 프로그램은 일반적으로 보다 우수한 쿼리 계획을 선택할 수 있습니다. 예를 들어, 로컬 변수를 사용하는 다음 쿼리를 고려하십시오.

```
declare @StartOrderDate datetime
set @StartOrderDate =  '20040731'
select * from Sales.SalesOrderHeader h, Sales.SalesOrderDetail d
WHERE h.SalesOrderID = d.SalesOrderId
AND h.OrderDate >= @StartOrderDate
```

최적화 프로그램이 예측하는 Sales.SalesOrderHeader의 행 개수는 조건 h.OrderDate >= @StartOrderDate가 9439.5이며 정확히 테이블 크기의 30%인지 확인합니다. 쿼리의 그래픽 실행 계획을 사용하고 Sales.SalesOrderHeader의 계획 노드를 마우스 오른쪽 단추로 클릭하여 이 카디널리티 예측을 표시할 수 있습니다. 이 백서를 준비하는 동안 사용된 SQL Server 2005의 시험판 버전에서 선택한 계획은 병합 조인을 사용합니다(관측은 동일한 SQL Server 2005 버전을 기반으로 이루어졌으며 결과는 SQL Server 버전, 가용 메모리 등에 따라 달라질 수 있습니다). 이제 로컬 변수를 사용하지 않는 동일한 쿼리를 검토해 보겠습니다.

```
SELECT * FROM Sales.SalesOrderHeader h, Sales.SalesOrderDetail d
WHERE h.SalesOrderID = d.SalesOrderId
AND h.OrderDate >=  '20040731'
```

조건부 "h.OrderDate >= '20040731' " 에 대한 결과 집합의 카디널리티는 선택성이 0.13%인 경우 쿼리의 그래픽 실행 계획(filter 연산자를 마우스 오른쪽 클릭)에서 40으로 예측됩니다. 이 쿼리에 선택된 계획은 향상된 예측 기능 덕분에 병합 조인 대신 중첩된 루프 조인을 사용합니다.

쿼리에 로컬 변수가 사용되었을 때 조차 동일성 조건부의 경우에 추측보다 나은 예측이 사용됩니다. 양식 "@local_variable = column_name"의 조건에 대한 선택성은 column_name에 대한 히스토그램의 평균 값 주기를 사용하여 예측됩니다. 따라서 예를 들어, 열 column_name에 모든 고유한 값이 포함된 경우 정확한 1/(열에서 고유한 값의 개수)의 선택성 예측이 사용될 것입니다.

로컬 변수의 사용을 예측하려면 (1) 변수 대신 문자를 사용하도록 쿼리를 재생성하거나, (2) 로컬 변수의 사용을 바꾸는 매개 변수가 있는 sp_executesql을 사용하거나, (3) 로컬 변수의 사용을 바꾸는 매개 변수가 있는 저장 프로시저를 사용하는 방법을 고려하십시오. EXEC를 통한 동적 SQL은 로컬 변수를 제거하는 데 유용할 수도 있지만 일반적으로 더 높은 컴파일 오버헤드가 발생할 것입니다.

다중 명령문 TVF 및 테이블 변수의 제한적인 사용 고려

다중 명령문 테이블 값 함수(TVFs)에는 통계가 없습니다. 최적화 프로그램은 결과의 크기를 추측해야 합니다. 마찬가지로 테이블 변수에는 통계가 없으며 최적화 프로그램은 카디널리티의 추측에 의지해야 합니다. 이러한 추측으로 인해 잘못된 결과가 발생하는 경우 표준 테이블이나 임시 테이블을 TVF 결과의 임시 저장 위치로 사용하거나 테이블 변수의 교체를 고려해야 합니다. 이에 따라 최적화 프로그램이 보다 우수한 카디널리티 예측을 사용하도록 할 수 있게 됩니다.

접을 수 없는 식과 내장된 스칼라 함수가 추측을 발생시킬 수 있음

SQL Server는 컴파일 시간에만 제한 조건을 포함하는 일부 식을 평가할 수 있습니다. 이를 constant folding이라고 합니다. 접을 수 있는 식은 선택성 예측 동안 문자처럼 취급됩니다. 접을 수 없는 식은 선택성 예측 동안 추측을 발생시킵니다. 예를 들어, 다음과 같은 Transact-SQL 스크립트를 생각해 보겠습니다. 이 스크립트는 200개의 행이 있는 UserLog 테이블을 채웁니다. 행의 절반은 고유한 UserName 값을 가지며 다른 절반은 동일한 UserName 값을 가지기 때문에 비대칭 배포가 이루어집니다.

```
IF object_id( 'UserLog' ) IS NOT NULL
        DROP TABLE UserLog
GO
CREATE TABLE UserLog (UserName NVARCHAR(255), Action NVARCHAR(1000))
DECLARE @i INT
SET @i = 1
SET nocount ON
WHILE @i <= 100
BEGIN
        INSERT UserLog VALUES(suser_sname(),  'login' )
        INSERT UserLog VALUES(newid(),  'login' )
        SET @i = @i + 1
END
```

suser_sname() 내장 함수는 현재 Windows 사용자에 대한 domain_name\user_name을 반환하고 newid()는 고유한 값을 제공합니다. 이제 동등한 2개의 쿼리를 실행할 것입니다. 아래의 첫 번째 쿼리에는 조건부 UserName = suser_sname()이 포함되어 있습니다. 최적화 프로그램은 결과의 카디널리티를 추측해야 하며 1.98을 추측합니다(SET STATISTICS XML ON 지시어에 의해 생성된 XML 실행 계획의 EstimateRows 특성을 통해 이를 확인할 수 있습니다). 이는 실제 카디널리티가 100이기 때문에 50 이상의 인수에 인해 OFF 되었습니다.

```
GO
SET STATISTICS XML ON
GO
SELECT * FROM UserLog WHERE UserName = suser_sname()
GO
SET STATISTICS XML OFF
GO
```

The second query is issued as a parameterized query through sp_executesql. The suser_sname() value is passed in as a parameter instead of appearing in the query as an expression.

```
SET STATISTICS XML ON
GO
DECLARE @UserName NVARCHAR(255)
SET @UserName = suser_sname()
EXEC sp_executesql N' SELECT * FROM UserLog WHERE UserName = @n' ,
      N' @n nvarchar(255)' , @UserName
GO
SET STATISTICS XML OFF
GO
```

이번에는 최적화 프로그램이 UserName = @n의 선택성을 위해 50%의 정확성 예측을 사용합니다. 생성된 XML 실행 계획을 보면 EstimateRows가 100으로 완벽한 것을 볼 수 있습니다. 대규모 데이터 집합의 보다 크게 복잡한 쿼리에서 이와 같은 유형의 오류는 잘못된 계획을 선택할 수 있습니다. 이것이 응용 프로그램의 문제인 경우 위에서 설명한 것과 같은 기술을 사용해 보십시오. 문제 쿼리를 포함하는 sp_executesql 또는 저장 프로시저를 사용하고 접을 수 없는 식의 미리 계산된 결과를 매개 변수로 전달하십시오. 이러한 방식을 통해 문제를 해결하고 우수한 카디널리티 예측을 입수할 수 있습니다.

쿼리가 다중 열 조건을 가질 때 다중 열 통계 사용

쿼리가 다중 열 조건을 가지고 있을 때, 최적화 프로그램이 쿼리를 위한 최상의 계획을 생성하지 않는다고 의심된다면 다중 열 통계 사용을 고려하십시오. 다중 열 조건을 지원하는 다중 열 인덱스가 이미 있다면 통계를 명시적으로 생성할 필요가 없기 때문에 다중 열 통계를 다중 열 인덱스의 생성에 따른 부산물로서 확보할 수 있습니다. 자동 생성 통계는 다중 열 통계가 아닌 단일 열 통계만 생성합니다. 따라서 다중 열 통계가 필요한 경우 수동으로 생성하거나 다중 열 인덱스를 생성하십시오.

AdventureWorks.Person.Contact 테이블에 액세스하는 쿼리를 고려하고 다음 조건을 포함하십시오.

```
FirstName =  'Catherine'  AND LastName =  'Abel'
```

이 쿼리에 대한 선택성 예측을 보다 정확하게 생성하려면 다음과 같은 통계 개체를 생성하십시오.

```
CREATE STATISTICS LastFirst ON Person.Contact(LastName,FirstName)
```

이 통계 개체는 LastName 및 FirstName은 물론 LastName에 한정된 조건부만을 포함한 쿼리에 유용합니다. 일반적으로 다중 열 통계 개체에서 열 집합의 접미사에 대한 조건부의 선택성은 해당 통계 개체를 사용하여 예측할 수 있습니다.

통계 개체가 다중 열 조건을 완벽하게 지원하려면 다중 열 통계 개체의 열 접미사는 조건에 열을 포함해야 합니다. 예를 들어, 열 (a,b,c)의 다중 열 통계 개체는 조건 a=1 AND c=1를 일부분만 지원하며, a=1에 대한 선택성에 히스토그램이 사용되지만 c의 밀도 정보는 b가 조건에 없기 때문에 사용되지 않습니다. (a,c) 또는 (a,c,b)에 대한 다중 열 통계는 조건 a=1 AND c=1을 지원하며 선택성 예측을 개선하기 위해 밀도 정보를 사용할 수 있습니다.

SQL Server가 선택성 추측에 사용되는 상황에 대한 경고

SQL Server는 여러 상황에서 선택성을 추측합니다. 많은 경우, 추측은 타당하고 데이터 크기가 작기 때문에 문제가 되지 않거나 추측으로 인해 잘못된 쿼리 계획이 생성되지 않습니다. 그럼에도 불구하고 SQL Server는 쿼리 조건부의 선택성을 추측해야 하며 최적이 아닌 쿼리 계획 결과를 얻게 되는 경우가 있습니다. 쿼리가 원하는 대로 수행되지 않고 차선책의 쿼리 계획을 선택하는 것으로 의심된다면 선택성이 통계를 기반으로 예측되는 것이 아니라 추측되는 것을 나타내는 징후가 있는지 쿼리와 결과 계획에서 확인해 보십시오. 많은 경우에 쿼리 또는 응용 프로그램 일부를 수정하여 추측을 방지할 수 있습니다. 아래에는 추측을 야기하는 몇 가지 상황과 가능한 해결 방법을 나열하고 있습니다.

- **손실된 통계**: 자동 생성 통계가 사용되거나 CREATE STATISTICS 또는 sp_createstats를 사용하여 통계를 수동으로 생성하는지 확인합니다. 자동 생성 통계가 작동하지 않도록 데이터베이스가 읽기 전용인지 확인합니다.
- 쿼리 조건에서 로컬 변수를 사용합니다(앞에서 설명했음).
- 쿼리 조건에서 상수가 아닌 접을 수 있는 식(예: T.x+1 = 0, suser_sname() = T.UserName). 식을 제거하도록 쿼리를 다시 작성하거나, 쿼리를 실행하기 전에 식을 평가하고 결과를 쿼리에 로컬 변수가 아닌 매개 변수로 전달합니다. T.x+1 = 0의 경우 식을 T.x = -1로 다시 작성합니다. 이는 등가의 방정식이며 추측이 아닌 정확한 예측을 가능하게 합니다.
- **복합 식** 예: "Price + Tax 〉 100" 또는 "Price * (1+TaxRate) 〉 100". 원하는 쿼리 성능보다 느린 경우 동일한 식을 포함하고 있는 계산된 열을 만들고 계산된 열에 통계나 인덱스를 생성하는 것을 고려하십시오. 자동 생성 통계는 계산된 열에 대한 통계를 생성하기 때문에 자동 생성 통계가 설정되어 있다면 계산된 열 통계를 수동으로 생성할 필요가 없습니다.

쿼리에 사용하기 전, SP에서 저장 프로시저 매개 변수 수정 방지

최고의 쿼리 성능을 위해서는 경우에 따라 새 값을 저장 프로시저의 프로시저 내에 매개 변수로 할당한 다음, 쿼리에서 매개 변수 값을 사용하는 것을 방지해야 합니다. 저장 프로시저와 그 안의 모든 쿼리는 매개 변수로서 쿼리에 처음 전달되는 매개 변수 값을 사용하여 처음으로 컴파일됩니다. 이를 parameter sniffing이라고 합니다. 특정 날짜 또는 그 이후 판매 정보를 입수하거나 NULL이 전달된 경우 판매의 최근 3개월 간의 판매 정보를 입수하는 다음 저장 프로시저를 생각해 보십시오.

```
CREATE PROCEDURE GetRecentSales (@date datetime) WITH RECOMPILE AS
BEGIN
    IF @date IS NULL
        SET @date = dateadd("mm" ,-3,(SELECT MAX(OrderDATE)
                                      FROM Sales.SalesOrderHeader))
    SELECT * FROM Sales.SalesOrderHeader h, Sales.SalesOrderDetail d
    WHERE h.SalesOrderID = d.SalesOrderID
    AND h.OrderDate > @date
END
```

이 SP를 NULL로 호출하는 경우 최종 SELECT 문이 @date = NULL에 대해 최적화됩니다. 행에는 NULL OrderDate가 없기 때문에 이 필터를 SalesOrderHeader에 적용하는 결과의 카디널리티 예측은 매우 낮습니다(1 row). 그러나 런타임에 날짜는 NULL이 아니라 최종 OrderDate보다 3개월 이전입니다. 자격을 지정하는 SalesOrderHeader 행의 실제 값은 5,736입니다. 최적화 프로그램은 NULL이 GetRecentSales에 전달될 때 쿼리에 대해 중첩된 루프 조인을 선택하는 반면 최적화 계획에는 병합 조인이 포함됩니다. 다음 스크립트를 사용하여 선택한 계획과 예상 및 실제 카디널리티를 볼 수 있습니다.

```
SET STATISTICS PROFILE ON
GO
EXEC GetRecentSales NULL
GO
SET STATISTICS PROFILE OFF
GO
```

위의 GetRecentSales 저장 프로시저에 WITH RECOMPILE 옵션이 사용되면 카디널리티 예측 오류가 제거되지 않습니다. 이 예제의 쿼리가 훌륭한 예측을 획득하도록 적절한 매개 변수 값으로 최적화되도록 보장하는 한 가지 방법은 저장 프로시저를 여러 부분으로 나누도록 다음과 같이 수정하는 것입니다.

```
CREATE PROCEDURE GetRecentSales (@date datetime)  AS
BEGIN
      IF @date IS NULL
            SET @date = dateadd( "mm" ,-3,(SELECT MAX(OrderDATE)
                                                FROM Sales.SalesOrderHeader))
      EXEC GetRecentSalesHelper @date
END

CREATE PROCEDURE GetRecentSalesHelper (@date datetime) WITH RECOMPILE AS
BEGIN
      SELECT * FROM Sales.SalesOrderHeader h, Sales.SalesOrderDetail d
      WHERE h.SalesOrderID = d.SalesOrderID
      AND h.OrderDate 〉 @date -- @date is unchanged from compile time,
                                          -- so a good plan is obtained.
END
```

오름차순 키에 대해 보다 빈번한 주기의 통계 수집 고려

실제 타임 스탬프를 나타내는 IDENTITY 열 또는 datetime 열 같은 오름차순 키 열은 새로운 값이 모두 히스토그램 외부에 있기 때문에 빈번한 INSERTS로 테이블에 대한 부정확한 통계를 발생시킬 수 있습니다. 응용 프로그램이 오름차순 키 열에 조건을 가진 쿼리에 대해 부정확한 쿼리 계획을 획득한다고 생각된다면 배치 작업으로 이러한 열에 대한 통계를 자주 업데이트해 보십시오. 배치 작업의 실행 빈도는 응용 프로그램에 따라 다릅니다. 매일 또는 매주 간격으로 수행할 수도 있고 응용 프로그램에 따라 보다 자주 실행하는 방안을 고려할 수 있습니다.

동기 업데이트에 원하지 않는 지연이 발생하는 경우 비동기 통계 업데이트 사용

대형 데이터베이스를 보유하고 있으며 OLTP 워크로드와 AUTO_UPDATE_STATISTICS가 설정된 경우, 대개 거의 실시간으로 실행되는 몇몇 트랜잭션은 통계를 업데이트시키기 때문에 수 초 이상 소요될 수 있습니다. 이와 같이 지연이 분명하게 나타나는 것을 피하려면 AUTO_UPDATE_STATISTICS_ASYNC를 사용하십시오. 장시간 실행되는 쿼리 워크로드의 경우, 최선의 계획을 확보하는 것이 컴파일에서 간헐적으로 지연이 발생하는 것보다 중요합니다. 이 경우 비동기 자동 업데이트 통계 대신 동기 업데이트를 사용하십시오.

요약

SQL Server 2005에는 통계 관리 기능에 대한 많은 향상이 포함되어 있습니다. 가장 중요한 것은 통계의 자동 생성과 업데이트를 통해 대부분 좋은 쿼리 계획을 유지할 수 있다는 것입니다. 기본 샘플링 속도를 지닌 자동 통계를 사용하면 통계 샘플링 속도와 업데이트 시간을 명시적으로 제어할 수 있습니다. 통계 또는 비용 예측과 관련하여 차선책의 쿼리 계획을 관찰하고 있다면, 이 백서에서 설명하는 최상의 방법을 고려해 보십시오.

참조 문헌

[Mar04] *Arun Marathe, Batch Compilation, Recompilation, and Plan Caching Issues in SQL Server 2005,* http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx, 2004년 7월.

Microsoft®

한국마이크로소프트(유)

서울특별시 강남구 대치동 892번지 포스코센터 ▌전화 : 080-985-2000 ▌인터넷 : www.microsoft.com/korea