데이터베이스에서 셀프조인 쿼리를 위한 성능평가

이원조° 이단영 권순덕 고재진
울산대학교 대학원 컴퓨터정보통신공학과
wjlee@mail.ulsan-c.ac.kr

Performance Evaluation for Self-Join Queries in Database

Won-Jo Lee° Dan-Young Lee Sun-Deok Kwon Jae-Jin Koh
School of Computer Engineering and Information Technology,
University of Ulsan

데이터베이스의 성능개선을 위해서 여러 가지 튜닝기법을 사용하고 있다. 그러나 DBMS의 성능문제는 약 60%가 응용프로그램의 SQL문에서 발생한다. 따라서 본 연구에서는 여러 가지 상황 중 MS SQL Server 2000의 [SQL Query Analyzer]를 사용하여 동일 쿼리의 셀프조인과 외부조인의 실험을 통하여 수행성능을 평가하였다.

1. 서 론

최근 데이터베이스의 사용이 일반화되어 데이베이스의 성능을 개선하기 위한 튜닝에 대한 연구가 활발하게 진행되고 있으며, 데이터베이스 제품들은 자체 튜닝기능을 내장하여 이를 지원하고 있다. 본 연구에서는 데이터베이스 설계시 셀프조인으로 쿼리해야 하는 데이터베이스 설계의 경우 셀프조인과 분할된 두 테이블을 외부조인 한 경우 쿼리의 성능을 평가하여 쿼리 성능개선에 유용한 평가자료로 활용하고자 한다. 두 가지 실험의 결과를 통하여 효율적인 데이터베이스 설계와 테이블 분할과 셀프조인에 대한 의문을 해결하고자 한다. 따라서 테이블의 통합과 분할에는 여러 가지 기본원칙이 있으나, 이 실험에서는 셀프조인과 단일 테이블을 분할한 두개 테이블의 외부조인을 MS SQL Server 2000의 [SQL Query Analyzer]를 사용하여 성능을 비교평가 한다.

2. 관련연구

2.1 서버 부하의 특성

서버시스템에서 부하의 발생요소는 크게 O/S의 자원할당 제어기능과 CPU 그리고 메모리, 디스크, 네트워크 그리고 응용프로그램 등으로 설명될 수 있다. 여기서 부하를 줄이는 방법과 부하에 영향을 주는 자원의 확장으로 성능을 개선할 수 있다. 또한 부하방지를 위한 시스템의 분산은 크게 두 가지 측면으로 보는데 하나는 네트워크의 부하를 분산하는 일과 수행하는 업무를 분산하는 일이다. 정보시스템에서 부하 분산 시 작업의 특성을 잘 고려해야 한다. 작은 규모로 분산할 경우에는 관리적인 측면의 비효율성을 가져온다. 서버시스템 내에서의 부하요소는 O/S, CPU, RAM, Disk, BUS, I/O Controller, Disk Controller, Network등을 이해하고 점검해야 한다. 그리고 CPU, RAM, I/O Controller는 시스템 버스에 종속되어 있으므로 시스템 버스의 구조는 시스템의 성능에 중요한 영향을 미친다. 따라서 연산이 많은 시스템의 경우에는 이 시스템 버스에서 부하가 집중적으로 발생하게 된다.[6][7][9]

2.2 SQL Server의 성능조정

-대부분의 조정작업은 SQL Server에게 위임
-RAM은 제한된 리소스

-유용한 인덱스를 만들어 유지
-디스크 I/O 하위 시스템 성능을 모니터
-응용 프로그램 및 쿼리 조정
-프로파일러와 인덱스 튜닝 마법사 활용
-성능 모니터로 병목지점 개선
-쿼리 분석기와 그래픽 표시의 이점 활용

2.3 질의 분석기(SQL Query Analyzer)

SQL Server 2000에는 DBA나 개발자가 쿼리를 작성하고, 동시에 다중 쿼리를 실행하고, 결과를 보고, 쿼리 계획을 분석하고, 쿼리의 성능을 향상시킬 수 있는 도움을 주는 대화식 그래픽 도구인 쿼리 분석기[SQL Query Analyzer]가 있다. [Show Execution Plan] 옵션은 SQL Server 쿼리 옵티마이저에서 선택한 데이터 검색 방법을 그래픽으로 표시해 준다. 이것은 쿼리의 성능특징을 이해하는 데 아주 유용하다. 또한 쿼리 분석기는 쿼리를 효율적으로 처리할 수 있는 쿼리 옵티마이저의 능력을 향상시키는 인덱스화되지 않은 컬럼의 추가 인덱스나 통계를 제시해 주기도 한다. 특히 쿼리 분석기는 통계를 읽은 것이 무엇인지 보여주므로 쿼리 옵티마이저가 선택성을 예측하게 해줄 뿐 아니라 아주 간단히 이러한 통계를 작성할 수 있도록 지원해준다.[1][2]

2.4 셀프조인(Self Join)

SQL문으로 데이터를 처리할 때 동일한 테이블내의 필드를 조인하여 쿼리 해야하는 경우가 종종 있다. 예를 들면 사원테이블 내에서 사원의 팀장을 쿼리하는 경우이다. 사원과 팀장의 정보가 동일한 테이블 내에 존재하기 때문에 별도의 테이블 같이 사용하기 위해서 하나의 테이블로 두 개의 Alias 작성하여 FROM절에 두개의 테이블을 사용하는 것과 같이한다.[1][2]

2.5 외부조인(Outer Join)

조건을 만족하지 않을 때는 정확한 질의결과에 나타나지 않으며, 정상적으로 조인조건을 만족하지 못하는 행들을 보기 위해 Outer join을 사용한다. 따라서 Outer join 연산자를 조인조건에 사용하면 조인조건을 만족하지 않는 행들도 결과에 나타날 수 있다. Outer join 연산자는 "(+)"이고, 조인시킬 값이 없는 조인측에 "(+)"를 위치 시킨다. 그 연산자는 괄호로 묶인 플러스 기호(+)이며, 정보가 부족한 조인측에 위치한다. (+)연산자는 한 개 이상의 NULL 행을 생성하고 정보가 충분한 테이블의 한 개 이상의 행들이 이런 NULL 행에 조인된다.[2]

3. 제안모델

사원 테이블에서 그 사원의 팀장을 쿼리 해야하는 경우에 사원 테이블을 두 개의 테이블로 Alias하여 Self Join 하게된다. 이러한 경우는 자주 발생하는 문제들이다. 만약 사원 테이블에서 [사번+성명]만을 포함한 팀장 테이블을 별도로 분리하여 Outer Join을 한다면 수행속도에 어떤 영향을 주는지, 이 두 실험의 수행결과를 분석하여 평가한다.[3][8]

4. 성능평가

4.1. 실험에 사용된 시스템 사양

-Hardware System : CPU Pentium 466, RAM 320MB
-Operating System : Microsoft Windows 2000 Server
-Database : Microsoft SQL Server 2000

4.2 실험방법

실험의 결과는 시스템의 사양과 운영체제와 데이터베이스의 종류, 테이블의 구조, 데이터의 량에 따라 다를 수 있다. 먼저 Join 실험을 위한 2개의 테이블을 생성하고 55건의 실험 데이터를 Up Load한다.[2][3]

```
-DEMO 테이블 생성
CREATE TABLE  DEMO
   (EMPNO          VARCHAR(05) NOT NULL,
   JUMIN_NO        VARCHAR(13),
   EMP_NAME        VARCHAR(10),
   JIKGB_CODE      VARCHAR(02),
   DEPT_CODE       VARCHAR(04),
   MGR_NO          VARCHAR(05),
      중략
   AREA_CODE       VARCHAR(02))
CREATE UNIQUE INDEX DEMO_1 ON DEMO
  (EMPNO)

-SELF 테이블 생성
CREATE TABLE  SELF
  (EMPNO          VARCHAR(05) NOT NULL,
   EMP_NAME       VARCHAR(10))
CREATE UNIQUE INDEX SELF_1 ON SELF
  (EMPNO)
```

Self Join과 Outer Join Script 문을 [SQL Query Analyzer]에서 실행한다.

```
-Self Join Script
Set Statistics Time On
SELECT a.EMPNO, a.JUMIN_NO, a.EMP_NAME,
        a.JIKGB_CODE, a.JIKCH_CODE,
        a.DEPT_CODE, a.MGR_NO,
        중략
        a.AREA_CODE
        FROM demo a  JOIN demo b
```

```
        ON a.MGR_NO = b.EMPNO

-Outer Join Script
Set Statistics Time On
SELECT a.EMPNO, a.JUMIN_NO, a.EMP_NAME,
        a.JIKGB_CODE, a.JIKCH_CODE,
        a.DEPT_CODE, a.MGR_NO,
        중략
        a.AREA_CODE
        FROM demo a  JOIN self b
        ON a.MGR_NO = b.EMPNO
```

4.3 실험결과

-Query Process Execution Plan

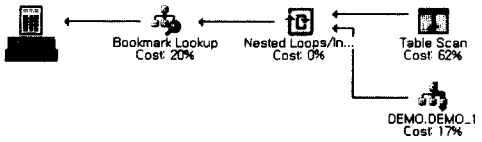

그림 1. Self Join Execution Plan

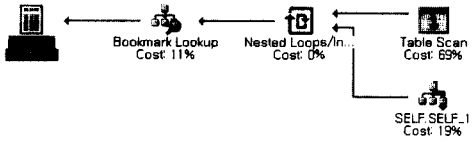

그림 2. Outer Join Execution Plan

-[Show Execution Plan] Option 지정

구 분		CPU time	Elapsed time
Self Join	Parse&Compile Times	0ms	0ms
	Execution Times	0ms	28ms
Outer Join	Parse&Compile Times	19ms	19ms
	Execution Times	20ms	33ms

표 1. [Show Execution Plan] Option 결과

-Statistics Times 결과

구 분		CPU time	Elapsed time
Self Join	Parse&Compile Times	0ms	0ms
	Execution Times	10ms	16ms
Outer Join	Parse&Compile Times	19ms	19ms
	Execution Times	0ms	16ms

표 2. Statistics Times 결과

5. 결 론

본 논문은 서론에서 언급된 바와 같이 셀프 쿼리와 테이블 분할에 대한 의문을 실험적으로 보여 주는데 목적이 있다. 따라서 실험의 결과와 같이 테이블 스캔(Table Scan) 효율은 Self Join이 우수하고, Execution Times은 Outer Join이 우수하나, Parse&Compile Times은 Self Join의 성능이 우수하게 평가되었다. 따라서 향후 과제는 대량 데이터, 다중 테이블 환경에서의 실험적 연구가 필요하다.

6. 참고문헌

[1] Dejan Sunderic, SQL Server2000 Stored Procedure Programming, McGraw-Hil, 2001
[2] Microsoft, Microsoft SQL Server7.0 Databse Implementation Microsoft Press, 2000
[3] 이공명, SQL 프로젝트 실무, 구민사, 2000
[4] Patrick Killelea, Web Performance Tunning, O'REILLY,2000
[5] A.J.H. Peddemors, A high performance distributed database system for enhanced Internet services, 1998
[6] 조행래, 워크스테이션 클러스터 환경 기반의 병렬 DBMS 개발 2001
[7] 정훈진, 클러스터 시스템에서의 동적 여분 부하 균등화 학국정보과학회, 2000
[8] 최용락, 데이터모델링을 이용한 데이터베이스시스템 성능 향상 데이터베이스연구회지 13권4호 147~160p
[9] 이원조, 케싱서버를 이용한 네트워크 최적화에 관한 연구 울산과학대학, 제27권, 2000 .