소용량 데이터베이스 처리를 위한 DBMS의 성능 비교

장시웅*

*동의대학교

Comparison of DBMS Performance for processing Small Scale Database

Si-woong Jang*

*Dongeui University

E-mail : swjang@deu.ac.kr

요 약

대규모 용량의 데이터베이스를 처리하기 위한 상용 DBMS에 대한 성능의 비교는 벤치마크 테스트로 결과가 주어진 경우가 많은 반면, 소규모 용량의 데이터베이스를 처리하기 위한 DBMS의 성능에 대해서는 많이 알려져 있지 않다. 따라서 본 연구에서는 소규모 용량의 데이터베이스를 처리하기 위한 상용 DBMS 및 공개용 DBMS의 성능에 대해 비교하고 분석하였다.

분석결과, 오라클은 데이터 갱신 및 삽입에 관한 연산에서는 데이터의 안전성 보장을 위한 rollback 기능의 준비 작업이 많아 좋지 못한 성능을 보인 반면, MySQL이나 MS-SQL 등의 경우에는 별다른 오버헤드가 없어 오라클에 비해 좋은 성능을 보였다.

ABSTRACT

While a lot of comparisons of DBMS performance for processing large scale database are given as results of bench-mark tests, there are few comparisons of DBMS performance for processing small scale database. Therefore, in this study, we compared and analyzed on the performance of commercial DBMS and public DBMS for small scale database.

Analysis results show that while Oracle has low performance on the operations of update and insert due to the overhead of rollback for data safety, MySQL and MS-SQL have good performance without additional overhead.

키워드

DBMS, Database, 성능분석

Ⅰ. 서 론

DBMS의 성능 분석에 관한 발표는 일반적으로 논문연구를 통하여 제시되기 보다는 벤치마크 테스트 기관들에 의해 이루어진다. 많은 공신력 있는 기관이 DBMS 성능 평가를 위한 벤치마크 테스트를 수행하고 테스트 결과를 발표하면, DBMS 업체들은 공신력 있는 기관의 테스트 결과를 인용하여 자사 제품의 성능이 우수함을 주장한다[1]. 물론 대용량의 데이터베이스 처리를 위하여는 오랫동안 시장 점유율이 높은 오라클이 안정성과 성능면에서 우수한 것으로 평가되지만 중소용량의 데이터에 대해서는 견해가 다르다[2,3].

따라서 본 연구에서는 소규모 업체나 교육용의 워크로드에 적합한 소용량의 데이터 처리에 많이 사용되는 MySQL, MS-SQL 및 Oracle의 성능에 대해 비교 분석하였다.

데이터베이스 시스템의 튜닝을 통한 성능 향상에 대한 연구로 [4]가 있는데, 이 논문에서는 시스템의 파라미터를 최적으로 조정함으로써 전체 데이터베이스 시스템의 성능을 향상시키는 방법을 제시하고 있다. DBMS 성능에 관한 다른 연구[5]가 존재하지만 DBMS들 간의 성능비교를 수행한 연구는 아주 소수에 불과하다.

II. DBMS 성능 관련 요소

DBMS의 성능에 관련된 사항으로는 DBMS의 튜닝 파라미터, 동시성, 인덱싱, 파티셔닝 등이 있으며 구체적인 내용은 다음과 같다.

2.1 DBMS의 튜닝

데이터베이스 시스템의 일반적인 구성요소로서 튜닝의 대상이 되는 항목으로는 병행제어, 로그와 복구, 디스크, 메모리가 있으며, 튜닝에 관한 사항은 다음과 같다.

1) 잠금과 병행처리

모든 관계형 데이터베이스 시스템은 변동처리 단위의 잠금을 처리하게 되므로 가능하면 변동처리가 포함되는 잠금의 횟수를 적게 하는 것이 유리하다. 잠금의 튜닝시에 고려할 사항은 다음과 같다.

- 불필요한 잠금 제거
- 시스템이 제공하는 Long Read의 기능 이용
- 잠금 단위의 적절한 선택 및 통제
- 데이터의 분산화 고려
- 변경과 수정이 많은 데이터는 작게 만들고 별도로 관리
- 잠금의 간격을 조정하여 데드락 가능성 회피

2) 로그와 복구

복구 기능은 시스템 장애 발생시 무결성을 지켜주는 기능으로 변동처리의 상태에서 완료를 처리하든지 중간 복귀를 처리하는 것으로서 데이터가 항상 정확성을 유지하도록 하는 기능이다. 이러한 무결성을 지원하기 위해 데이터베이스 관리 시스템들은 이미지 파일과 로그 파일을 보유하고 있으며 장애 발생 이후 복구시에 이용된다.

2.2 동시성

동시성은 멀티-유저 환경에서 특정 사용자가 수정한 데이터가 다른 사용자에게 영향을 미치는지의 여부를 나타내는 중요한 지표이다. SQL 서버 2005는 기본적으로 읽기 작업에 대해 공유 읽기 잠금을 사용한다. 이것은 읽기/쓰기 작업이 동시에 발생하는 환경에서 동시 요청을 처리하는데 성능상 불리한 것으로 알려져 있다. SQL 서버 2005의 트랜잭션 읽기 일관성은 SQL 서버 2000에서 지원되지 않았기 때문에 이전 버전보다 개선되었다고 볼 수 있으나 오라클은 이와 유사한 멀티-버전 읽기 일반성을 기본으로 지원해 왔다.

2.3 인덱싱

인덱스를 이용하면 디스크의 I/O 작업을 크게 줄이고 데이터 인출 성능을 개선할 수 있다. 인덱스를 이용하면 이진 탐색이 가능하므로 테이블에 N개의 레코드가 존재하면 원하는 데이터를 검색하기 위해 데이터를 탐색하는 회수는 $\log_2 N+\alpha$ 이다. 이를테면 10,000,000($<2^{21}$)개의 레코드 중에서 원하는 데이터를 찾기 위해서는 약 21~25회 이내의 탐색으로 찾을 수 있으나 순차탐색의 경우에는 평균 5,000,000회의 탐색을 거쳐야 원하는 데이터를 찾을 수 있다.

오라클과 SQL 서버 2005는 모두 고전적인 B-트리 인덱스 구조를 지원한다. B-트리 인덱스는 순차적으로 정렬된 키 값과 원본 데이터에 대한 위치가 저장되어 있다.

2.4 파티셔닝

파티셔닝(Partitioning)은 테이블, 인덱스 등 대규모 데이터베이스 구조를 관리하기 쉽게 더 작은 단위로 분해하는 기능이다. 파티션은 레인지 파티셔닝, 해시 파티셔닝, 리스트 파티셔닝으로 분류할 수 있으며 각각의 내용은 다음과 같다.

레인지 파티셔닝은 일정 영역의 컬럼 값을 이용해 행을 파티션에 매핑한다. 이 옵션은 히스토리 데이터베이스에 유용하게 활용된다. 해시 파티셔닝은 파티션 컬럼에 해시 함수를 적용해서 데이터를 분산시키는 방법으로 균일하게 분포된 데이터에 효율적이다. 리스트 파티셔닝은 행을 파티션으로 맵핑하는 방법을 관리자가 명시적으로 설정하는 방법으로 관리자는 파티셔닝 컬럼을 위한 값의 리스트를 정의하는 방법으로 매핑 방법을 설정한다.

III. DBMS의 성능 평가

3장에서는 교육용으로 많이 사용되는 DBMS인 MySQL 3.51, MS-SQL 2000, 오라클 8i의 성능에 대해 평가하고 분석한다.

3.1 성능 평가 방법

고객 정보에 관한 테이블을 하나 생성한 후 데이터의 레코드 수를 삽입, 검색, 수정, 삭제하면서 3가지 DBMS인 MySQL, MS-SQL, 오라클의 성능을 평가하였다. 성능평가를 수행한 테이블의 구조는 표 1과 같다.

표 1. 고객 테이블의 구조

속성 \ 특성	데이터형	데이터크기
주민등록번호	char	30
ID	char	20
이름	char	20
마일리지	Integer	4

성능 평가를 위해 데이터를 삽입, 검색, 수정 및 삭제하기 위한 화면은 그림 1과 같다. DBMS에서 제공하는 SQL 명령어를 이용하여 테이블을 생성한 후 그림 1에서 설계한 화면에 서 데이터를 구성한다. 데이터의 워크로드는 표2와 같이 구성하였다.

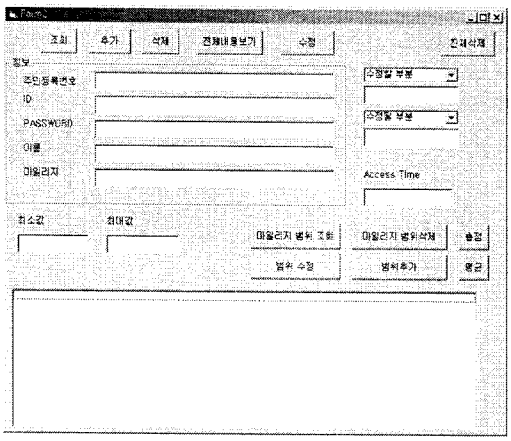

그림 1. 성능평가를 위한 데이터 조작화면

그림 1의 화면에서 범위 추가를 통하여 최소값과 최대값을 지정하면 주민등록번호가 일련번호로 최소값부터 최대값까지 생성된다. 따라서, 최소값을 1 최대값을 1000으로하면 1부터 1000까지의 주민등록번호가 생성되며, ID, PASSWORD, 이름 등에는 랜덤값(RAND)을 발생하여 임의의 문자열을 추가하고 마일리지에는 임의의 숫자를 생성하여 저장한다.

표 2. 성능평가를 위한 워크로드

명령어 \ 회수	1	2	3	4	5
insert	1000	2000	4000	8000	16000
select	1000	2000	4000	8000	16000
update	1000	2000	4000	8000	16000
delete	1000	2000	4000	8000	16000

각 회수에 대해 insert, select, update, delete의 명령어에 대한 수행시간을 3개 DBMS에 대해 평가한 후 성능을 분석하였다.

3.2 성능 분석

그림 2는 데이터의 개수를 1000개에서 16000개까지 2배씩 증가시켜가면서 데이터를 insert한 후 수행시간을 측정하였다. 실험결과 데이터의 용량이 소규모인 경우에는 MySQL, MS-SQL, Oracle 등의 순으로 좋은 성능을 나타냈다. MySQL의 경우에는 삽입할 레코드의 수가 증가되어도 수행시간이 완만하게 증가하는 반면, MS-SQL은 다소 빨리 증가하고, Oracle 은 급격히 수행시간이 증가하는 현상을 보였다. insert의 경우에는 데이터량이 2배로 증가하면 수행시간은 약 1.6~1.8배 정도씩 안정되게 증가하는 특성을 보인다. 따라서, insert 명령의 경우에는 한번에 처리하는 레코드의 수가 증가하여도 시스템의 효율성은 저하되지 않는 특성을 보여준다.

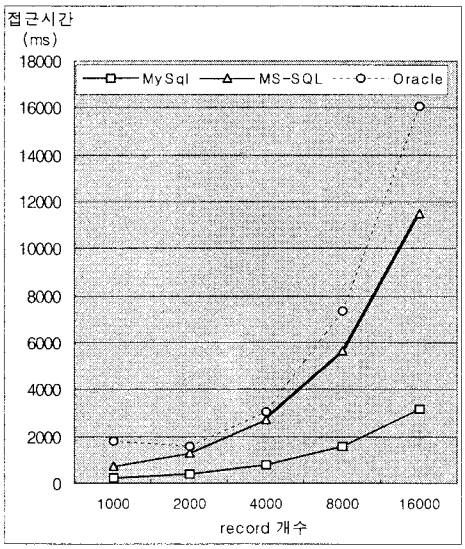

그림 2. DB별 데이터량 증가에 따른 insert 성능측정

그림 3의 그래프를 보면 select의 경우에도 수행시간에 있어서는 insert의 경우와 유사한 수행시간 패턴을 보였으나 수행시간은 select 명령어의 1/5 수준으로 나타났다.

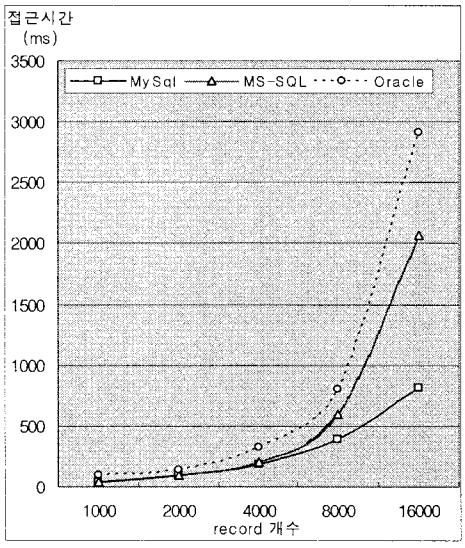

그림 3. DB별 데이터량 증가에 따른 select 성능측정

그림 4는 delete의 경우도 select의 경우와 유사한 패턴을 보이는 것을 보여주며, 수행시간도 select와 유사하게 나타났다. MySQL의 경우에는 삭제하고자하는 레코드의 수가 증가하여도 접근시간은 완만한 증가세를 보여 가장 좋은 성능을

보여 준다.

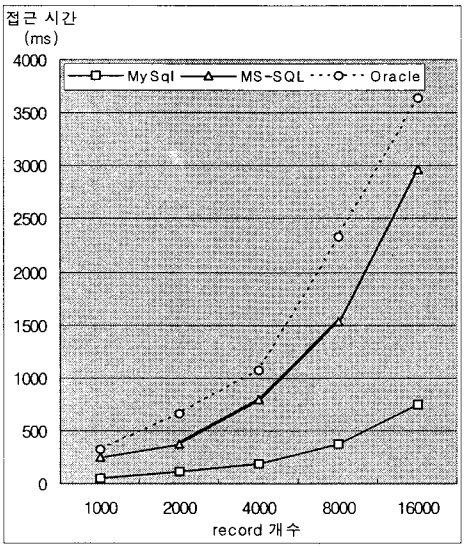

그림 4. DB별 데이터량 증가에 따른 delete 성능측정

그림 5는 update 성능을 비교한 것으로 다른 연산에 비해 수행시간이 월등히 많이 걸리는 것을 보여준다.

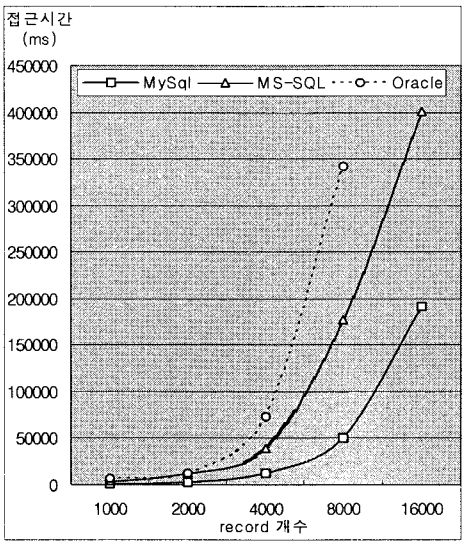

그림 5. DB별 데이터량 증가에 따른 Update 성능측정

특히, update 명령어의 절대적인 수행시간은 insert의 경우에 비해 20배이상 많이 걸리는 것으로 분석되었다. 이는 update의 경우에는 roll back을 고려한 오버헤드가 크기 때문인 것으로 분석된다. 특히 데이터의 레코드 개수가 4000에서 8000으로 증가할 때 record 개수에 따른 수행시간은 3~4배 이상 증가되어 한 번에 갱신하는 데이터의 양이 많은 경우에는 수행시간이 기하 급수적으로 증가하는 것을 보여준다.

IV. 결론

본 연구에서는 교육용이나 소규모 업체에 적합한 소용량 데이터를 처리할 경우 흔하게 사용되는 3종의 DBMS가 성능면에서 어떤 특성을 보이는지 살펴보았다.

예상했던 것처럼 소용량의 데이터에 대해서는 SQL의 명령어 종류에 관계 없이 DBMS의 성능은 MySQL, MS-SQL, Oracle의 순으로 나타났다. MS-SQL은 Oracle보다 성능면에서 앞서지만 record 개수에 따라 수행시간이 급격히 증가되어 오라클과 유사한 패턴을 보였다. MySQL은 모든 SQL 명령어에 대해 좋은 성능을 보였다. 이는상업용인 Oracle이나 MS-SQL은 MySQL보다 많은 기능과 보안성으로 인해 DBMS자체에 많은 코드들이 추가되어 오버헤드가 많아진 것 때문으로 해석된다.

한편, 대용량 데이터를 대상으로하는 벤치마크 테스트에서는 Oracle의 성능이 다른 DBMS에 비해 월등한 것으로 보고되어 있기 때문에 데이터 처리량의 규모에 따라 DBMS의 선택이 이루어지는 것이 바람직하다고 판단된다.

참고문헌

[1] 김도근, "성능 항목별 DBMS 3종 비교 분석", 마이크로소프트웨어, pp.154-165, 2006. 1
[2] 유경용, "DBMS 성능 벤치마크, 신뢰의 조건", 마이크로소프트웨어, pp.144-153,, 2006.1
[3] 원 훈, "틈새시장을 노린 DBMS들:MM DBMS 성능의 비밀", 마이크로소프트웨어, pp.198-204, 2006. 1
[4] 황호현, " ORACLE DBMS를 활용한 SQL 튜닝에 관한 연구", 경희대 교육대학원 석사학위논문, 2006
[5] 金基奉, "멀티쓰레드 서버 구조를 갖는 DBMS 성능분석", 忠南大 大學院 석사학위논문,1993