Journal of Digital Contents Society Vol. 15 No. 5 Oct. 2014(pp. 631-642)
http://dx.doi.org/10.9728/dcs.2014.15.5.631

메인메모리 데이터베이스시스템에서의 어플리케이션 성능 평가

김희완*, 안연식**

요 약

메인 메모리 DBMS는 구동과 동시에 디스크에 존재하는 테이블의 내용이 메모리로 모두 올라가는 구조로 운영된다. 그러나, 메인메모리 DMBS도 디스크 파일시스템을 사용하여 데이터를 저장하고 트랜잭션 로그파일을 파일 시스템에 유지하여야 하기 때문에 CPU가 메모리에 접근하여 처리하는 속도에는 한계가 있게 마련이다. 본 논문에서는 메인메모리 DBMS인 Altibase 시스템에서 구현된 기술들이 디스크 기반 DBMS인 Sybase와 얼마나 차이가 있는지를 어플리케이션 측면에서 성능분석을 통하여 평가하였다. 메인메모리 DBMS의 어플리케이션 성능이 디스크기반 DBMS에서의 어플리케이션 성능과 비교하면, 메인 메모리 DBMS의 성능이 축구승무패 단일게임에서는 1.24배 ~ 3.36배, 축구승무패와 축구 스페셜 두 게임에 동시 발매시 1.29배 ~ 7.9배의 성능향상이 있었으며, 발매 트랜잭션의 응답시간 시험결과 1.78배 ~ 6.09배의 빠른 응답시간을 보였다.

키워드 : 메인 메모리 DBMS, 디스크기반 DBMS, 트랜잭션, 성능, 응답시간

Application Performance Evaluation in Main Memory Database System

Hee-Wan Kim*, Yeon S. Ahn**

Abstract

The main memory DBMS is operated which the contents of the table that resides on a disk at the same time as the drive is in the memory. However, because the main memory DBMS stores the data and transaction log file using the disk file system, there are a limit to the speed at which the CPU accesses the memory. In this paper, I evaluated the performance through analysis of the application side difference the technology that has been implemented in Altibase system of main memory DBMS and Sybase of disk-based DBMS. When the application performance of main memory DBMS is in comparison with the disk-based DBMS, the performance of main memory DBMS was outperformed 1.24~3.36 times in the single soccer game, and was outperformed 1.29~7.9 times in the soccer game / special soccer. The result of sale transaction response time showed a fast response time of 1.78 ~ 6.09 times.

Keywords : Main Memory DBMS, Disk-driven DBMS, Transaction, Performance, Response Time

1. 서론

급변하는 정보화 세계 속에서 기존의 정보시스템은 정보처리 속도의 제약으로 사용자의 신속한 정보처리 요구를 만족시키기에는 한계를 보이고 있다. 이러한 속도 제약의 배경에는 백-엔드에 위치한 데이터베이스 서버의 느린 트랜잭션 처리 성능이 주원인으로 지적되고 있다. 일반 디스크 기반 DBMS와 대비되는 MMDBMS의 가장 큰 차이점은 운영 데이터의 존재 위치

※ 교신저자(Corresponding Author): Yeon S. Ahn
접수일:2014년 10월 15일, 수정일:2014년 10월 30일
완료일:2014년 10월 31일
* 삼육대학교 컴퓨터학부(제1저자)
Tel: +82-31-750-5537, Fax: +82-31-750-5537
email: ahndreo@gachon.ac.kr
** 가천대학교 경영학부

가 다르다는 점이다. 디스크 DBMS는 보통 버퍼 매니저를 통하여 메모리상에 일정 크기만큼의 버퍼를 잡고서 알고리즘에 의하여 디스크에 존재하는 데이터를 메모리 버퍼에 올려서 접근(Access)한다. 버퍼의 크기가 유한하기 때문에 일정시간이 경과하거나 최근에 접근하지 않은 데이터는 다시 디스크로 내려서 데이터를 관리하게 되는데 DBMS 입장에서는 이로 인한 버퍼 매니저 관리 비용이 상당히 많은 단점을 가지고 있다. 한편 MMDBMS는 구동과 동시에 디스크에 존재하는 테이블의 내용이 메모리에 모두 올라가는 구조로 운영된다. 디스크 데이터와 메모리 데이터 관리를 위한 버퍼 매니저가 필요 없으며 버퍼 매니저 관리를 위한 비용을 절감하여 리소스를 다르게 활용할 수 있다[1]. 따라서 디스크 기반 DBMS에서 충분히 큰 메모리를 갖고 있을 경우에도 버퍼 매니저의 관리 비용으로 메모리 상주형 DBMS에서의 성능 보다는 우수하지 못함을 다른 연구에서도 보였다[2,3].

CPU가 메모리에 접근하는 속도는 디스크의 I/O 처리를 감안하면 수천 배의 빠름에도 불구하고 성능에서는 수십 배 정도의 차이가 나는 것은 MMDMBS도 디스크 파일시스템을 기본적으로 사용하여 데이터를 저장하고 만약의 경우를 대비하여 트랜잭션 로그파일을 파일 시스템에 유지하기 때문이다. 기존 연구에서는 MMDBMS와 디스크 DBMS에서의 성능을 SQL의 SELECT, INSERT, UPDATE, DELETE문에서 성능을 비교하였다. 데이터 변경 연산(Insert, Update, Delete문)은 약 50배 정도의 성능 차이를 보였으며, 데이터 조회 작업은 디스크 DBMS 역시 버퍼 캐쉬에 존재하는 데이터를 대상으로 하였기 때문에 약 3~4배 정도의 차이가 발생함을 보였다[4].

본 논문에서는 메인메모리 DBMS인 Altibase 시스템에서 구현된 기술들이 디스크 기반 DBMS인 Sybase와 어떻게 차이가 있는지를 어플리케이션 측면에서 성능분석을 통하여 평가하고자 한다.

2. 메인메모리 데이터베이스

2.1 메인메모리 데이터베이스 시스템

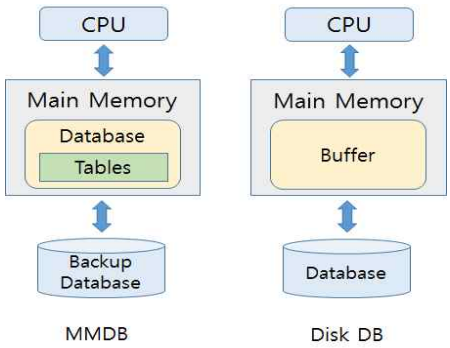
메인 메모리 데이터베이스 시스템은 메인 메모리에 영구적인 데이터베이스를 구축▪관리하는 데이터베이스 시스템이다. 즉 데이터베이스의 일부 또는 전부를 메인메모리에서 관리함으로서 디스크에 대한 접근 없이 메모리 접근만으로 직접 데이터를 처리하여 효율적인 트랜잭션 처리를 가능하게 하는 기술이다. 지금까지 실시간(real time) 분야에서는 빠른 응답 시간의 요구를 만족시키는 메인메모리를 활용한 실시간 데이터베이스 모듈을 개발하여 활용해 왔으나 이러한 접근법은 몇 가지 문제점에 안고 있다. 기존의 실시간 데이터베이스 모듈은 특정 응용에 한정되도록 개발됨으로써 범용성이 떨어질 뿐만 아니라 데이터베이스 관리의 개념이 희박하다는 점, 그리고 산업 표준을 준수하는 프로그래밍 인터페이스, 트랜잭션의 실시간 처리 능력, 확장 가능한 구조 등을 동시에 만족시키는 데이터베이스 시스템으로서의 역할을 하지 못한다는 점이다. 메인메모리 데이터베이스가 디스크 기반 데이터베이스에 비해 트래잭션 처리 용도에 적합한 이유는 낮은 접근 비용과 메모리 구조 최적화에 있다.

디스크 기반 데이터베이스 시스템의 기본적인 개념은 모든 데이터를 디스크에 저장, 관리하는 것이다. 저장 장치로서의 디스크는 메인 메모리에 비해 저장된 데이터의 지속성이 뛰어난데 비해 상대적으로 접근 속도가 느리다. 따라서 대부분의 기존 데이터베이스 시스템은 메모리에 임시 저장소(버퍼)를 마련하고 데이터를 이중적으로 관리한다. (그림 1)과 같이 접근 속도를 높이기 위해, 메모리에서 버퍼를 관리하는 버퍼 관리자를 통해 데이터에 접근한다. 버퍼 관리자는 한정된 버퍼를 효율적으로 활용하기 위한 다양한 교체(replacement) 알고리즘을 구현하고 있는데, 이러한 구조는 데이터 접근 및 질의 처리에 있어서 알고리즘의 복잡도를 증가시키는 한계를 갖는다.

반면 메인메모리 데이터베이스 시스템은 메모리에 데이터베이스를 구축하고 백업 데이터베이스를 디스크에서 관리하는 구조를 갖고 있다. 테이블과 인덱스를 비롯한 데이터베이스의 데이터 객체는 메인메모리에 직접 저장▪관리된다. 즉, 메인메모리에 상주한 데이터 객체들은 디스크에 저장된 객체에 대한 임시 저장소의 역할을 하는

것이 아닌 주저장소로서의 역할을 하게 된다.

(그림 1) 메인메모리 데이터베이스와 디스크기반 데이터베이스[5]



(Figure 1) MMDB and Disk DB[5]

데이터를 메인메모리에서 관리하는 최대 이점은 디스크 접근을 최소화 하는데 있다. 또한 데이터가 디스크에 존재한다는 가정이 사라지므로 메모리 구조에 효율적인 질의 처리를 통해 높은 트랜잭션 처리율을 보일 수 있다.

2.2 메모리-디스크간 주소 변환

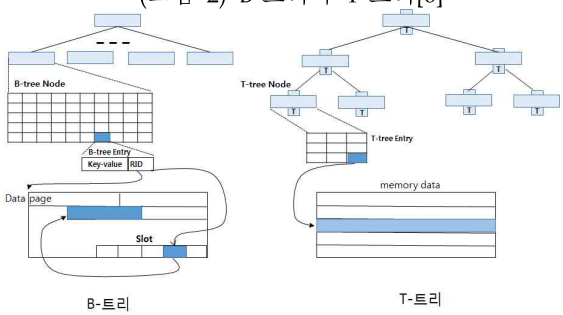
디스크 기반 데이터베이스는 기본적으로 디스크에 모든 데이터가 있다고 가정하기 때문에, 레코드를 접근하기 위해서는 RID(Record Identifier)를 사용한다. 따라서, 메모리 내에서 실제로 해당 레코드를 접근할 때, 메모리의 물리적 주소(physical address)로 매핑하는 주소변환이 필요하다. 이러한 메모리 주소와 데이터베이스 주소간 매핑의 단위 처리 시간은 아주 짧지만, 고속의 데이터 처리시에는 무시할 수 없는 성능 요소가 된다.

반면, 메인메모리 데이터베이스는 메모리내의 데이터베이스를 메모리 포인터로 직접 접근할 수 있다. 그러므로, 주소 변환 비용을 제거함으로써 성능상의 향상을 얻을 수 있다. 그러나, 디스크의 백업 데이터베이스에 대한 동기화와 회복을 위한 로그 생성시, 메모리 주소를 디스크 주소로 변환할 필요는 있다. 따라서, 이러한 주소 변환을 최소화 하는 기법과 효율적인 주소 변환 기법에 따라 성능상으로 큰 차이점이 있다.

2.2.1 메인메모리 데이터베이스의 인덱스 구조

디스크 기반 데이터베이스 시스템의 전형적인 인덱스 기법은 다양한 변형된 형태를 포함하여 B-트리 인덱스이다. 이 인덱스 기법은 (그림 2)와 같이 폭이 넓고 깊이는 얕은 트리 구조를 가지는데, 이것은 데이터를 접근할 때 필요한 트리의 노드 수를 최소화 하는 장점이 있다. 또한, 디스크 사용의 효율성을 위하여, 각 노드의 크기는 보통 디스크 페이지 크기에 대응되도록 구현된다. 즉, B-트리 인덱스는 디스크 I/O 회수를 줄이면서 디스크 공간을 적게 사용하는 것에 초점이 맞추어진 것이다. 그러나, B-트리는 균형(balance)을 맞추기 위하여 각 노드들간의 데이터 이동의 양과 회수가 많아지게 된다.

(그림 2) B-트리와 T-트리[6]



(Figure 2) B-Tree and T-Tree[6]

메인메모리 데이터베이스 시스템은 연산에 필요한 모든 데이터가 메인메모리에 올라와 있기 때문에, 인덱스 구조의 핵심은 전체적인 계산 시간을 줄이고 메모리 사용량을 줄이는데 있다. 이러한 요구에 의해, 메인메모리 데이터베이스에는 B-트리의 장점과 AVL-트리의 장점을 결합한 T-트리가 사용된다. T-트리는 B-트리에 비해 폭이 좁고 깊이는 깊은 구조를 가지는데, 이것은 데이터를 접근할 때 필요한 계산 시간을 최소화하는 장점이 있다. 이러한 구조는 균형을 유지하기 위해 필요한 회전(Rotation) 연산 비용을 메인메모리 구조에 최적화한 것으로, 데이터 검색뿐만 아니라 데이터 변경에 따른 인덱스 구조 변경에 필요한 시간을 최소화하는 장점이 있다[6].

2.2.2 메인메모리 기반의 질의 최적화

디스크 기반 데이터베이스 시스템에서 질의 최적화 알고리즘은 기본적으로 디스크 I/O를 최소화 하기위해 구현되어 있다. 따라서, 모든 데이터가 버퍼에 캐싱되어 있다고 하더라도 데이터가 디스크에 존재한다는 가정을 배제할 수 없기 때문에 최악(worst case)의 시나리오를 고려한 알고리즘을 사용할 수밖에 없다.

그러나, 메인메모리 데이터베이스 시스템은 메모리내의 데이터만 고려하면 되기 때문에, 디스크 기반 데이터베이스보다 더 최적화된 알고리즘을 구현할 수 있다. 또한, 데이터 접근 비용을 계산하는 평가비용이 상대적으로 훨씬 낮다.

2.2.3 메인메모리에 최적화된 로킹 및 회복 기법

메인메모리 데이터베이스 시스템은 (그림 1)에서처럼, 디스크에 저장되는 백업 데이터베이스가 존재하는데, 그 이유는 메인메모리의 휘발성(volatility) 특성을 보완하기 위한 안전장치이다.

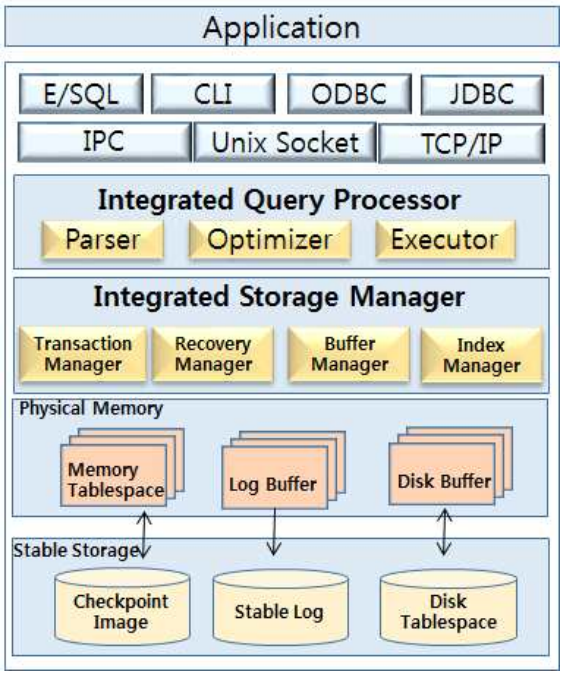
따라서, 데이터베이스의 기본 특성인 지속성을 보장하기 위해서는 먼저, 메모리내의 데이터베이스를 디스크에 동기화시키는 기술이 필요하다. 또한, 트랜잭션 처리시 ACID 특성을 만족시키기 위한 회복 및 복구기법[7]이 필요하다. 현재 상용화된 메인메모리 데이터베이스 시스템은 이러한 지속성을 보장하기 위하여 다양한 기술을 적용하고 있다. 로그파일 관리는 고성능 트랜잭션 처리를 위한 개선된 다중 병행수행 제어 방법을 제공한다. 데이터의 로킹 단위(granularity)를 세분화하여, 즉 테이블과 레코드 단위로 구분하여 다중버전 병행수행 제어 방법[8,9]를 사용한다. 버전 기법을 복구를 위해 사용하는 로그파일 측면에서는 메모리 테이블과 디스크 테이블에서 발생하는 트랜잭션을 순차적으로 일관성 있게 관리하기 위해서 동일한 로그파일을 이용 가능하도록 통합했으며 데이터 저장소는 메모리 테이블을 위한 메모리 테이블 스페이스 개념을 도입하여 디스크 테이블의 일반 테이블 스페이스와 구분하였다. 이와 같이 때로는 구조를 유연하게 합쳐서 일관성을 유지하였고 때로는 구조를 분리하여 구현함으로써 최대한의 성능을 향상시킬 수 있도록 설계되었다.

3. Altibase의 구조

3.1 전체적인 시스템 구조

ALTIBASE HDB의 전체 구성은 자료 저장 관리자와 질의 처리기로 구성된 서버 부분과 응용 프로그램 작성을 위한 클라이언트 라이브러리, 그리고 이들 간의 통신 모듈로 구성되어 있다. 또한, 여러 개의 유틸리티와 디스크에 저장되는 메모리 테이블스페이스의 Checkpoint Image파일, 디스크 테이블스페이스의 데이터 파일, 로그 파일이 있다.

(그림 3) Altibase 시스템 구조[10]



(Figure 3) Altibase System Structure[10]

메모리 테이블스페이스는 디스크에 데이터를 적재하고 일부 데이터를 버퍼에 cache 하는 것과는 달리 모든 데이터가 메모리에 적재되는 공간이다. DB가 구동될 때, 모든 메모리 데이터를 업로드하여 관리하기 때문에 디스크I/O가 거의 발생하지 않는다.

로그 버퍼는 DBMS에서 트랜잭션의 영속성을 보장하기 위해 리두로그(Redo log)를 물리적인 파일로 관리한다. 그런데 트랜잭션이 발생할 때

마다 물리적인 파일에 로그를 기록한다면 성능이 저하되기 때문에 메모리에 로그 버퍼를 만들어서 변경된 데이터베이스 내용에 대해서 메모리에 먼저 로깅을 수행한다.

디스크 버퍼는 디스크I/O를 최소화하기 위해서 데이터파일에 저장되어 있는 디스크 데이터를 메모리에 적재하는 공간이다. 이 때, 버퍼에는 데이터파일로부터 읽어온 데이터 페이지의 복사본이 저장된다.

체크포인트 이미지 파일은 메모리는 휘발성이라는 특징을 가지고 있기 때문에 시스템 전원이 나간다거나 장애가 발생할 경우에 DB에 저장된 메모리 데이터는 모두 유실되는데, 이런 경우를 대비하여 메모리 데이터를 디스크에 물리적인 파일로 주기적인 백업을 받는데 이 파일을 체크포인트 이미지 파일이라고 한다[10].

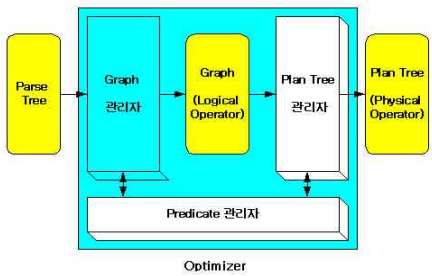
3.2 ALTIBASE의 질의 최적화 기술

SQL 질의 처리 단계에 있어서 가장 중요한 단계는 질의 최적화 단계이다. 질의 최적화는 궁극적으로 질의를 처리하는 과정에서 디스크 I/O, CPU 사용, 테이블 접근 회수, 중간 결과 크기 등을 최소화 하는 기술이다. 일반적으로 질의 최적화는 비용 기반(cost-based)과 규칙 기반(rule-based) 방법이 있다. 비용 기반 최적화란 주어진 SQL 문장을 처리하는데 있어서 각각의 연산에 대한 시간비용을 계산하여 가장 비용이 적게 드는 방법으로 plan tree를 생성하는 과정이고, 규칙 기반 최적화는 정해진 내부 최적화 규칙에 의해 plan tree를 생성하는 방법이다. 전자는 보다 최적화된 plan tree를 만들어 낼 수 있지만 비용을 계산하는데 시간이 많이 들고 알고리즘도 복잡하다. 후자는 정해진 규칙대로 plan tree를 생성하므로 그 과정이 단순하지만 비효율적인 plan tree가 생성되는 경우가 발생할 수 있다.

메인 메모리 DBMS에서 사용자의 질의 처리 시, 비용 기반으로 최적화된 데이터 접근 경로를 계산하기 위한 비용이 디스크 기반 DBMS 보다 상대적으로 적다. 또한 질의 변형(query transformation)을 비롯한 질의 최적화 알고리즘은 메모리 주소공간에 최적화된 기법이 적용된다. 이는 질의 최적화 시간을 최소화 시키며, 실제 데이터 접근 속도를 향상시킴으로써 월등한 성능 상의 차이를 가져온다. 따라서 메인 메모리 DBMS는 비용 기반 및 규칙 기반의 최적화 기법을 동시에 사용하는 것이 효과적이다.

질의 최적화의 개략적인 시스템 구조는 (그림 4)와 같이 그래프 관리자, 플랜 트리 관리자, 서술(predicate) 관리자로 구분한다. 각 관리자의 역할은 다음과 같다.

(그림 4) 질의최적화의 시스템 구조[11]



(Figure 4) System Structure of Query Optimizer[11]

- 그래프 관리자 : 주어진 파스 트리를 이용해 그래프를 생성하고, 각 그래프에 대한 최적화를 수행한다.
- 플랜 트리 관리자 : 그래프 관리자에 의해 생성된 그래프를 이용해 플랜 트리를 생성한다.
- 서술 관리자 : 그래프 관리자 및 플랜 트리 관리자가 공통적으로 사용하는 모듈로, 산술식(예, i1, i1 + 1)과 서술식(예, i1 = i2)을 관리한다. 즉, 그래프 관리자 및 플랜트 관리자는 서술 관리자를 통해 산술식 및 서술식에 대한 부가 정보를 이용 및 설정을 하게 된다.

질의를 처리하면서 내부적으로 가장 많이 접근하는 테이블은 메타 테이블이다. 메타 테이블은 데이터베이스의 정보 즉, 사용자 정의 테이블, 사용자, 그리고 기타 데이터베이스 관련 정보를 포함하는 DBMS 내부 테이블이라 할 수 있다. 이 메타 테이블을 DBMS의 질의 처리기와 자료저장 관리기가 수시로 참조하게 된다. 따라서 메타 테이블에 대한 접근 성능을 극대화할 필요성이 있는데, 이를 위해 디스크 기반 DBMS는 메타 테이블을 공유 메모리 영역에 상주시켜(캐싱하여) 성능 향상을 꾀하고 있다.

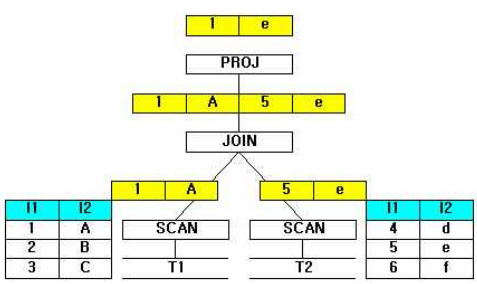
메인 메모리 DBMS는 메타 테이블을 포함하

여 모든 DB를 메인 메모리에 상주시켜 처리하므로 이미 메타 테이블을 메모리에 캐싱해 두고 있다고 볼 수 있다. 그러나 일부 메인 메모리 DBMS는 이에 만족하지 않고 보다 더 메타 테이블의 접근 성능을 높이기 위한 기술들을 적용하고 있다. DBMS 내부 모듈 중 질의처리기가 메타 테이블을 가장 많이 접근한다. 따라서 질의처리기는 메타 테이블을 일반 테이블과 동일한 방식으로 접근하는 대신에 질의처리에 효과를 발휘할 수 있도록 메타 테이블을 내부 구조에 맞게 캐싱하여 관리하는 기술을 사용한다.

Altibase의 질의 처리 방식은 기존의 질의 처리 방법이 파이프라인(Pipelining) 질의처리 방법 대신에 메모리 상주의 이점을 최대한 살린 투풀셋(Tuple-set) 질의 처리 방식을 사용한다.

파이프라인 질의 처리 방법은 하위의 플랜노드의 연산 결과를 임시 테이블로 만들어서 이 결과를 상위 플랜노드로 전달하는 방식을 취한다. "SELECT T1.I1, T2.I2 FROM T1, T2;" SQL을 처리한다고 가정할 때, (그림 5)처럼 최하위 두 개의 노드(SCAN)는 T1, T2를 각각 스캔 처리하여 스캔 결과를 임시 파일로 만들고 이 결과를 상위 플랜 노드(JOIN)에 전달한다. JOIN 플랜노드는 입력 받은 두 개의 테이블을 조건에 따라 조인 연산을 수행하여 결과로 한 개의 임시 테이블을 만들어 상위 노드(PROJ)에 전달한다. 최종적으로 PROJ 노드는 입력 받은 임시테이블로부터 해당 컬럼만 프로젝션하여 결과를 낸다. 파이프라인 질의 처리 방법은 구현하기는 쉽지만 임시테이블을 생성하고 등록하는 비용이 드는 단점이 있다. 최악의 경우 임시테이블 크기의 메모리 공간을 차지하므로 잦은 메모리 복사가 성능에 장애가 될 수 있다.

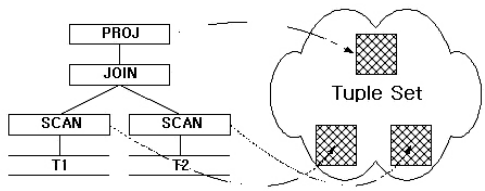
(그림 5) 파이프라인 질의 처리 예[11]



(Figure 5) Example of Pipeline Query Process[11]

Row Pointer-based 질의 처리 기법이라고도 하는 투풀 셋 질의처리 방법은 plan tree의 하위 노드부터 방문하며 처리하는 것은 파이프라인 질의 처리 방식과 동일하지만, 문제점은 하위의 플랜노드는 상위 노드가 어떤 일을 수행하는지 고려할 필요가 없이, 자신의 일만 수행하고, 각각의 플랜노드의 연산 결과를 임시 테이블로 만들지 않는다. 또한, 투풀 셋이라는 자료구조에 연산결과에 대한 조건과 이에 해당하는 Row에 대한 포인터만을 저장한다. 따라서 임시 테이블을 만들지 않으므로 메모리가 절약하며, 플랜노드 연산 결과가 Row에 대한 포인터만을 유지하므로 메모리 복사가 현저히 줄여서 질의를 빠르게 처리할 수 있도록 구현하였다.

(그림 6) 튜플셑 질의 처리 예[11]



(Figure 6) Example of Tuple Set Query Process[11]

3.3 ALTIBASE의 응용에 대한 이중화 기술

MMDBMS는 빠른 성능을 제공하는 것 이외에도 시스템 구성 측면에서도 탁월한 유연성을 제공하는데, 이는 '이중화(Replication)'라는 핵심 기술 때문이다.[1]

서버 구성에 따라 데이터베이스 이중화를 위한 통신 모델은 다음의 네 가지로 구분된다. 주-대기 서버 이중화 모델은 데이터베이스 이중화 뿐만 아니라 분산 시스템의 전형적인 이중화 모델로서, 정상적인 운영 모드에서 주 서버는 일반적인 데이터베이스 서비스를 제공하는 것처럼 보이지만, 실제로 자신의 변경 내용을 대기 서버로 전달하고 있으며, 대기 서버는 데이터베이스 서비스는 하지 않은 채, 전달 받은 내용을 자신의 데이터베이스에 반영하게 된다. 주 서버에 오류가 발생하면, 주 서버의 서비스를 받던 응용들을 대기 서버로 옮겨 대기 서버에서 새로운 서

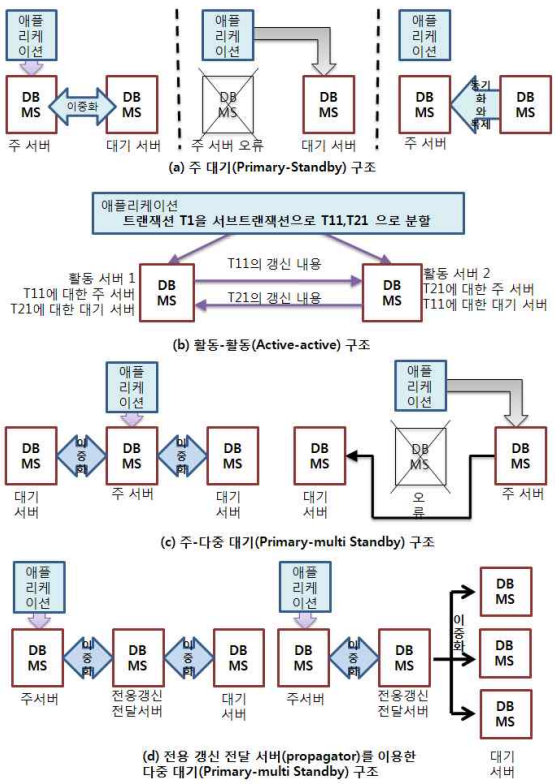
비스를 시작하게 된다. 주 서버의 오류가 복구되면 그 동안의 대기 서버의 변경 트랜잭션을 주 서버에 보내 이중화 작업을 계속 수행한다. 즉 애플리케이션 수준에서 트랜잭션 T1을 T11과 T21로 분할 한 후, 각 서브 트랜잭션을 (그림 7)의 (나) 모델을 적용하여, T11을 활동 서버1에게, T21을 활동 서버2에게 전송시켜 각각의 서버에서 서브 트랜잭션을 수행시킨 후, 결과를 조합하여 최종 결과를 얻을 수 있게 된다. 이때, 트랜잭션 T11에 대해서는 활동 서버1이 주 서버, 활동 서버2가 대기 서버 역할을 하게 되며, 반대로 트랜잭션 T21에 대해서는 활동 서버2가 주서버, 활동 서버1이 대기 서버 역할을 하게 된다. 데이터베이스 이중화 기능을 위해서는 활동 서버1에서 수행된 T11의 변경 내용은 활동 서버2로 전송되고, 마찬가지로 활동 서버2에서 수행된 T21의 변경 내용은 활동 서버1로 전송되어, 상호 이중화 기능을 갖도록 수행하는 한편, 두 서버를 이용한 부하 분산 기능을 갖도록 한다.

활동-활동 서버 이중화 모델은 응용 업무를 분리하여 처리하거나 부하 분산을 위해 주로 사용된다. 변경된 데이터 갱신 내용을 서로 상대편 서버에 이중화하는 작업을 수행하고, 응용에서 발생하는 트랜잭션을 두 그룹으로 분리하여 각각의 서버에서 수행하게 한다. 이후 변경 트랜잭션을 교환, 양 서버의 데이터 변경 내용을 일치시켜 상호 이중화하는 방식이다.

주-다중(Active-Multistandby) 대기 이중화 모델은 성능에 민감하고 높은 가용성을 요구하는 응용에 적합한 모델로서, 하나의 주 서버와 두 개 이상의 대기 서버로 구성된다. 주-대기 서버와 기본적으로 동일한 구성을 갖지만, 대기 서버 수가 많다는 점에서 시스템의 가용성을 더욱 높일 수 있다. 이 구조는 높은 고가용성을 얻을 수 있어, 크리티컬한 응용 시스템에 적합한 모델이지만 비용이 많이 든다는 단점이 있다.

다중 대기 서버에 변경 트랜잭션을 전달하기 위해 소요되는 주 서버의 오버헤드를 절감시키기 위해, (그림 7)의 (라)와 같은 전용 변경 전달 서버(propagator)를 운영하여 주-다중 대기 서버 모델을 운영할 수 있다[12].

(그림 7) Altibase 시스템 이중화 모델[12]



(Figure 7) Altibase System Duplication Model[12]

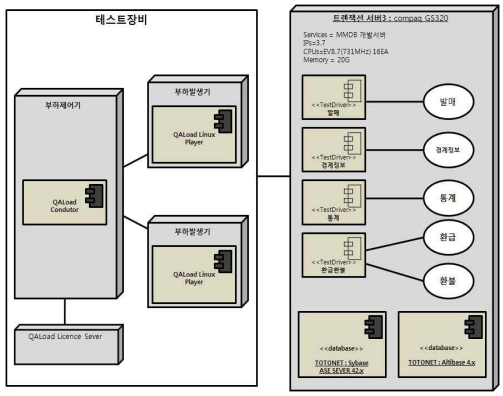
4. 어플리케이션 성능 평가

4.1 시험 시스템 구성

본 논문에서는 S사[13]에서 운영 중인 응용 트랜잭션을 근간으로 MMDBMS인 Altibase 시스템과 디스크 기반 DBMS인 Sybase에서 TPS(Transactions Per Second)를 기준으로 시험하였다. DBMS 자체의 정확한 성능 검증을 위하여 윈도우즈 기반의 성능테스트 도구가 갖게 되는 네트워크단의 부하를 최소화하기 위하여 S사에서 제공한 테스트 Driver형태의 테스트 Scripts를 사용하여 시험하였다. 본 절에서 수행한 시험은 (그림 8)과 같이 731MHz CPU 16개와 20GB의 메모리를 보유한 compaq GS320 서버에서 수행하였다. Sybase DBMS로부터 Altibase 의 Hybrid DBMS 를 Data store 로 사용하도록 수정된 기존 어플리케이션에 대하여

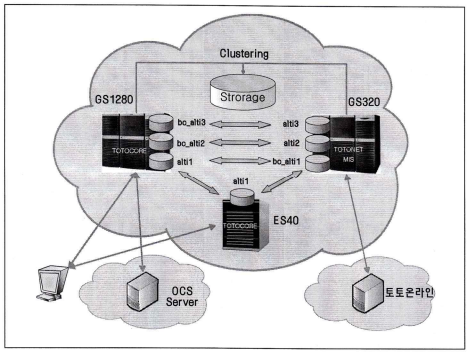
정량화된 성능 수치를 확보하고 검증하기 위해 수행한 발매 Transaction Load Test를 실시하였다. 본 어플리케이션 성능 시험에서는 S사가 현재 서비스하고 있는 많은 게임들 중에서 축구승무패와 축구스페셜 두 게임을 대상으로 발매에 대한 성능과 부하 처리능력을 Test 하였다.

(그림 8) 시험 시스템 구성



(Figure 8) Test System Configuration

(그림 9) 시험 시스템 구조



(Figure 9) Test System Structure

4.2 평가 시스템 환경

두 시스템의 성능 평가를 위한 데이터 크기는 동일한 데이터를 사용하였으며, Test 시간은 Altibase와 Sybase DBMS를 동시에 로드한 상태에서 테스트 하였으며, 그 내용은 다음 <표 1>과 같다. 또한 시험 트랜잭션은 ALTIBASE와 Sybase 시스템에서 제공하는 프로시저를 사용하였으며, 트랜잭션의 데이터베이스 요구는 모두 질의 처리기를 거쳐 최적 수행 계획에 따라 수행되었다. 전체 대상 프로그램은 1,996본이었으며, 7,189회의 이벤트를 발생시켜서 실험을 수행하였다. Sybase 어플리케이션의 성능은 이미 운영환경에서 수행한 결과를 Baseline으로 하였으며, 축구승무패 단일게임의 경우 180 TPS로 나타났으며, 축구승무패과 농구스페셜의 2 게임에서의 TPS는 80 으로 나타났다. 따라서, Altibase DBMS의 성능은 위의 TPS를 기준으로 비교하고자 한다.

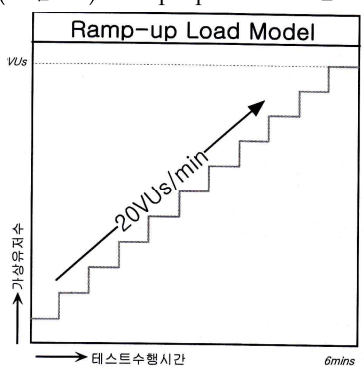
<표 1> 시험 데이터의 크기

table-name	rows
PROGRAM	1,996
EVENT	7,189
WINNING	9,265
Agencystate	9,265

<Table 1> Size of Test Data

사용자 부하량 증가에 따른 처리량(Throughput) 및 응답시간, 시스템 자원의 변화량에 따른 적정한 가상 사용자수 및 부하량에 따른 Trend 를 확인하기 위하여 (그림 10)과 같이 Ramp-up Load Model 테스트를 수행하였으며, 1분당 100 가상 사용자(100 VUs/min) 간격으로 6분 동안 부하량을 증가시키면서, Peak Load는 동시 가상 사용자수를 500으로 하였다.

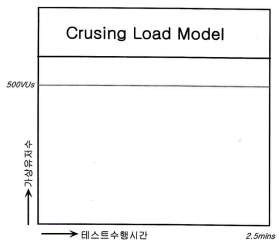
(그림 10) Ramp-up Load 모델



(Figure 10) Ramp-up Load Model

최대 가상 사용자수(500VUs)에 따른 처리량(Throughput) 및 응답시간, System Resource 변화량에 따른 Trend 를 확인하기 위하여 (그림 11)과 같이 Cruising Load Model 테스트를 시행하였다. Cruising Time 은 500 가상 사용자를 2.5분 동안 부하를 지속시키면서 Peak Load는 동시 가상 사용자를 200으로 하였다.

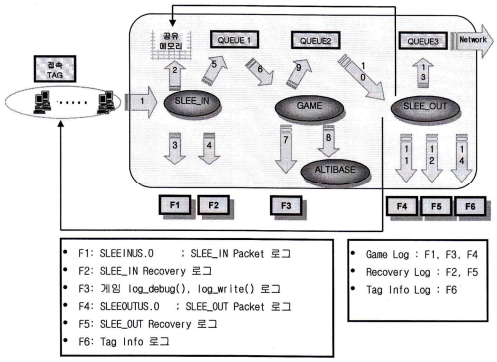
(그림 11) Cruising Load 모델



(Figure 11) Cruising Load Model

어플리케이션 최적화 차원에서 I/O Wait 으로 인한 근본적인 병목현상을 제거하고자 어플리케이션 단에서 시스템 장애 시 데이터 복구, 데이터 오류 시 대사 그리고 디버깅 등을 위해 남기고 있는 다양한 로그를 남기지 않도록 함으로써 전반적인 Transaction 처리 성능에 얼마만큼의 영향을 미치는지를 객관화 할 수 있도록 Test 모델에 다음 (그림 12)와 같은 어플리케이션 Logging 환경에서 Test를 수행하였다.

(그림 12) 어플리케이션 로깅 환경



(Figure 12) Application Logging Environment

위의 Logging 환경을 근간으로 테스트를 수행한 5가지 Logging Model은 다음 <표 2>와 같다.

<표 2> 시험 Logging 모델

Logging Model	Game Log	SLEE_IN/SLEE_OUT Archive Log	비고
Logging #0	O	Sync.	Sybase와 동일
Logging #1	O	X	
Logging #2	X	Sync.	
Logging #3	X	Async.	
Logging #4	X	X	

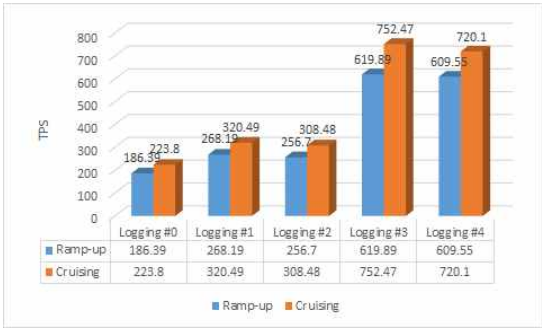
<Table 2> Test Logging Model

Logging #0 모델의 경우 Game Log를 사용하면서 동기식으로 발권하는 모델로 운영 중인 Sysbase와 동일한 환경의 모델이며, 나머지 #2, #3, #4 모델의 경우에는 Game Log인 F1, F3, F4를 사용하지 않고 동기/비동기로 발매를 하는 경우를 가정하여 테스트를 수행하였다.

4.3 축구승무패 단일게임 대상 테스트 결과

본 절에서는 축구승무패 단일게임을 대상으로 Test 를 수행한 결과는 (그림 13)과 같이 나타났다. Altibase 어플리케이션은 기존 Sybase 와 동일한 Logging Model 인 Logging #0 에서는 223.8 TPS 로 Test 장비가 운영 장비에 비해 Resource 사용 환경이 더 좋다는 것을 감안할 때 예상보다 미미한 기존 Sybase 성능인 180 TPS 대비 1.24 배 향상된 성능을 보였다.

(그림 13) 축구승무패 단일게임의 결과



(Figure 13) Results of Soccer Game Test

Altibase 어플리케이션에서 Game Log 를 제외하고 SLEE_IN, SLEE_OUT 의 Archive Logging 을 비동기(Async.)로 설정한 Logging Model Logging #3에서는 752.47 TPS 로 기존 Sybase 와 동일한 Logging Model 인 Logging #0 대비 3.36 배의 우수한 성능을 보였다.

4.4 축구승무패와 축구스페셜 두게임 동시발매 대상 테스트 결과

4.3절의 축구승무패 단일게임 대상 테스트 결과를 반영하여 축구승무패와 축구스페셜 두 게임에 대한 동시발매 테스트를 수행하였다. 좀 더 다양한 환경요소의 영향을 파악할 수 있도록 Logging Model 과 게임 Process 개수를 변경해 가며 <표 3>과 같이 시험을 수행하여 그 결과를 측정하였다.

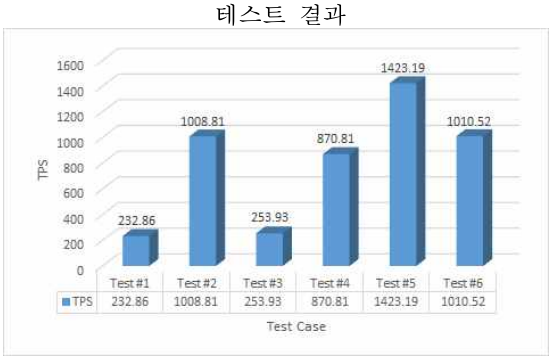
<표 3> 여러 프로세스 시험 Logging 모델

Load Model	Test Model	Test Case
Cruising Load 492 Vus	Test #1	Logging #0 / 1 game / 10 Process
	Test #2	Logging #2 / 1 game / 10 Process
	Test #3	Logging #0 / 2 game / 10 Process
	Test #4	Logging #2 / 2 game / 10 Process
	Test #5	Logging #2 / 1 game / 5 Process
	Test #6	Logging #2 / 2 game / 5 Process

<Table 3> Test logging model of multiple processes

(그림 14)와 같이 Sybase 와 동일한 조건의 Logging Model인 Logging #0 하에서 단일 게임의 경우 232 TPS로 Sybase 대비 1.29 배의 향상된 성능을 보였으며, 두 게임에서는 253 TPS 로 나타났다. 최소의 로그만을 남기도록 Log 를 조정한 Logging Model인 Logging #2를 적용하고 Game Process 개수를 5 개로 조정한 경우에는 단일 게임에서는 1,423 TPS, 두 게임에서는 1,010 TPS 를 보였다. 단일 게임을 대상으로 Logging #0 과 Logging #2 모델만을 변경하여 Test 를 수행한 결과 Logging #2 가 4.33 배 내외의 향상된 성능을 보였다. 두 게임을 대상으로 한 동시발매의 경우 Logging #0 과 Logging #2 모델만을 변경하여 Test를 수행한 결과 Logging #2 가 3.43 배 내외의 향상된 성능을 보였다. Logging #2 모델을 적용한 상태에서 단일 게임을 대상으로 Game Process 개수만을 10 개와 5 개로 변경하여 시험한 결과 Game Process 5 개가 1.41 배 내외의 향상된 성능을 보였다. 동일한 상황에서 두 게임을 대상으로 동시발매 Test 를 수행한 결과 Game Process 5 개가 1.16 배 내외의 향상된 성능을 보였다.

(그림 14) 축구승무패와 축구스페셜 두게임의 테스트 결과

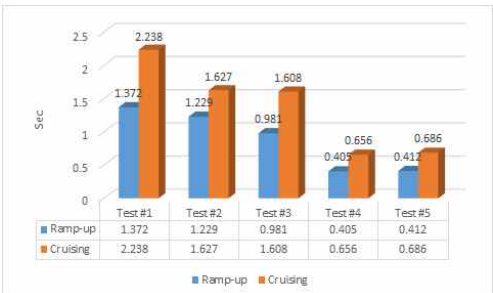


(Figure 14) Results of Soccer Game and Soccer Special Game Test

4.5 발매 트랜잭션에서의 응답시간 테스트 결과

Altibase 어플리케이션의 Logging Model을 변경하면서 Test 한 결과 기존 Sybase 와 동일한 Logging Model인 Logging #0에서는 2.238 Sec(Mean)로 기존 Sybase의 4 Sec 보다 빠른 응답시간을 보였다. Altibase 어플리케이션에서 Game Log 를 제외하고 Logging #3에서는 0.656 Sec(Mean)로 빠른 응답시간을 보였다.

(그림 15) 발매 트랜잭션 응답시간 결과



(Figure 15) Results of Ticketing Transaction Response Time

5. 결론

디스크기반 데이터베이스 관리시스템은 백-엔드에 위치한 데이터베이스 서버의 느린 트랜잭션 처리가 성능저하의 주원인으로 지적되고 있다. 일반 디스크 기반 DBMS와 대비되는 메인메모리 DBMS는 운영 데이터의 존재 위치가 다르다는 점이다. 메인 메모리 DBMS는 구동과 동시에 디스크에 존재하는 테이블의 내용이 메모리로 모두 올라가는 구조로 운영된다. 그러나, CPU가 메모리에 접근하는 속도는 디스크의 I/O 처리를 감안하면 수천 배의 빠름에도 불구하고 성능에서는 수십 배 정도의 차이가 나는 것은 메인메모리 DMBS도 디스크 파일시스템을 기본적으로 사용하여 데이터를 저장하고 만약의 경우를 대비하여 트랜잭션 로그파일을 파일 시스템에 유지하기 때문이다. 본 논문에서는 메인메모리 DBMS인 Altibase 시스템에서 구현된 기술들이 디스크 기반 DBMS인 Sybase와 어떻게 차이가 있는지를 어플리케이션 측면에서 성능분석을 통하여 평가하였다. 축구승무패 단일게임을 대상으로 시험한 결과는 메인메모리 DBMS에서의 어플리케이션의 TPS는 223.8 로 디스크기반 DBMS인 Sybase 성능인 180 TPS 대비 1.24 배 향상된 성능을 보였다. TPS 대비 1.24 배 향상된 성능을 보였다. 그러나, Altibase 어플리케이션에서 동기(Async.)로 설정한 경우에서는 752.47 TPS 로 기존 Sybase 와 비교할 때 3.36 배의 우수한 성능을 보였다. 축구승무패와 축구스페셜 두 게임에 대한 동시발매 테스트를 수행한 결과 Sybase 와 동일한 조건의 Logging Model인 Logging #0 하에서 단일 게임의 경우 232 TPS로 1.29 배의 행상된 성능을 보였으며, 두 게임에서는 253 TPS 로 나타났다. 최소의 로그만을 남기도록 로그를 조정한 모델에 적용한 결과 1,423 TPS로 7.9배, 두 게임에서는 1,010 TPS로 5.6배의 우수한 성능을 보였다.

발매 트랜잭션에서의 응답시간을 시험한 결과 Altibase 어플리케이션에서는 같은 조건의 환경에서는 2.238 Sec(Mean)로 기존 Sybase의 4 Sec 보다 1.78배 빠른 응답시간을 보였다. Altibase 어플리케이션에서 Game Log 를 제외한 모델에서는 0.656 Sec(Mean)로 6.09배 빠른 응답시간을 보였다.

따라서, 메인메모리 DBMS의 성능이 디스크기반 DBMS에서의 이플리케이션 성능평가 결과 축구 단일게임에서는 1.24배 ~ 3.36배, 축구승무패와 축구 스페셜 두게임에 동시 발매시 1.29배 ~ 7.9배의 성능향상이 있었으며, 발매 트랜잭션의 응답시간 시험결과 1.78배 ~ 6.09배의 빠른 응답시간을 보였다.

References

[1] S-H. Lim, “Utilization Plan of MMDBMS by Construction Example,” on the Net, pp.168-172, July 2006.

[2] P. Bohannon, J. Parker, R. Rastogi, S. Seshadri, A. Sil-berschatz, and S. Sudarshan, "Distributed Multi-Level Recovery in Main-Memory Databases," Proc. of the International Conference on Parallel and Distributed Information Systems, 1996.

[3] H. Garcia-Molina and K. Salem, "main Memory Database Systems : An Overview," IEEE Transactions on Knowledge and Data Engineering, Vol 4. No. 6, 1993.

[4] K. W. Lee, “Transaction performance evaluation of main memory DBMS,” Journal of Korea Computer Industrial Education Society, Vol.6, No.2, pp.559-565, June 2005.

[5] http://brainwave.tistory.com

[6] http://www.dbguide.net

[7] Per-Åke Larson, Spyros Blanas, Cristian Diaconu, Craig Freedman, Jignesh M. Patel, and Mike Zwilling, ““High-performance concurrency control mechanisms for main-memory databases,“ Journal of the VLDB Endowment, Vol.5. No.4. pp.298-309, 2011

[8] D. Agrawal and V. Krishnaswamy, "Using Multiversion Data for Non-Interfering Excution of Write-Only Transactions," Proc. of the ACM SIGMOD International Conference on Management of Data, 1991.

[9] P. M. Bober and M. J. Carey, "Multiversion Query

Locking," Proc. of the 18th Conference on Very Large Database, 1992

[10] http://www.gurubee.net/lecture/2154

[11] http://database.sarang.net/?inc=read&aid=21&criteria=mmdbms&subcrit=&id=&limit=&keyword=&page=

[12] http://database.sarang.net/?inc=read&aid=22&criteria=mmdbms&subcrit=&id=&limit=&keyword=&page

[13] Sports Toto Ltd., "STMMDB: SportsTOTO MMDB Construction Project - Application_Test_Report", Sports Toto Ltd, March 2007.

김 희 완

1995년 : 성균관대학교 컴퓨터공학부(공학석사)
2002년 : 성균관대학교 컴퓨터공학부(공학박사)

1987년~1991년: 한국전력공사 정보처리처(DBA)
1996년~2001년: 삼육의명대학 전산정보과 부교수
1996년 : 정보관리기술사 취득
2001년 : 정보시스템 수석감리원 취득
2001년~현재: 삼육대학교 컴퓨터공부 교수
관심분야 : 데이터베이스, 데이터베이스 보안, 정보시스템 감리, 서비스 기술 등

안 연 식

1989년 연세대학교 전산학과(공학석사)
2002년 국민대학교 정보관리학과(박사)

1981~1992 한국전력공사
1992~1996 한전 KDN(주)
전자계산조직응용기술사, 정보시스템감리사 자격보유

현, 가천대학교 경영학과 교수
관심분야 : 기술경영, 서비스경영, 데이터베이스, 정보시스템 평가 등