Journal of Korea Multimedia Society Vol. 8, No. 4, April 2005(pp. 439-449)

분산 주기억장치 데이터베이스에서 카탈로그 관리 기법의 성능평가

정한라[†], 홍의경[††], 김 명[†††]

요 약

분산 주기억장치 데이터베이스 관리 시스템 (DMM-DBMSs)은 데이터베이스를 참여 사이트의 주기억장치에 저장한다. 지역 데이터베이스에 신속하게 접근할 수 있고 사이트들 사이에 고속 통신이 가능하므로 DMM-DBMS는 높은 성능을 보장한다. 최근 들어 DMM-DBMS에 대해 많은 연구 결과가 발표되었으나 아직까지 DMM-DBMS의 카탈로그 관리 기법에 대한 성능 분석 결과는 발표된 것이 없다. 본 연구에서는 높은 사이트 자치성을 보장하는 DMM-DBMS의 분할식 카탈로그 관리 기법의 성능을 실험적으로 평가하였다. 분할식 카탈로그 관리 기법을 캐시없는 분할식 카탈로그 방식(PCWC), 점진적 캐시를 이용한 분할식 카탈로그 방식(PCWIC), 완전 캐시를 이용한 분할식 카탈로그 방식(PCWFC)으로 분류하였고, 성능평가는 사이트 수, 사이트 당 터미널 수, 버퍼 크기, 쓰기 질의 비율, 지역 질의 비율 등을 중심으로 분석하였다. 분석 결과 PCWFC가 모든 경우에 가장 높은 성능을 보였다. 이는 또한 PCWIC가 시간이 흐름에 따라 더욱 높은 성능을 보인다는 것을 뜻한다. PCWFC 방식은 디스크 기반 분산 DMBS에서는 사이트 부하가 크거나 카탈로그 쓰기 비율이 높거나 원격 데이터 객체가 빈번히 액세스되는 상황에서 고성능을 보장하지 못하지만, DMM-DBMS에서는 원격 데이터 객체의 카탈로그가 자주 갱신된다고 해도 질의 컴파일과 원격 카탈로그 액세스가 고속으로 이루어질 수 있기 때문에 높은 성능을 보장하는 것이다.

Performance Evaluation of Catalog Management Schemes for Distributed Main Memory Databases

Han-Ra Jeong[†], Eui Kyeong Hong[††], Myung Kim[†††]

ABSTRACT

Distributed main memory database management systems (DMM-DBMSs) store the database in main memories of the participating sites. They provide high performance through fast access to the local databases and high speed communication among the sites. Recently, a lot of research results on DMM-DBMSs has been reported. However, to the best of our knowledge, there is no known research result on the performance of the catalog management schemes for DMM-DBMSs. In this work, we evaluated the performance of the partitioned catalog management schemes through experimental analysis. First, we classified the partitioned catalog management schemes into three categories : Partitioned Catalogs Without Caching (PCWC), Partitioned Catalogs With Incremental Caching (PCWIC), and Partitioned Catalogs With Full Caching (PCWFC). Experiments were conducted by varying the number of sites, the number of terminals per site, buffer size, write query ratio, and local query ratio. Experiments show that PCWFC outperforms the other two schemes in all cases. It also means that the performance of PCWIC gradually increases as time goes by. It should be noted that PCWFC does not guarantee high performance for disk-based distributed DBMSs in cases when the workload of individual site is high, catalog write ratio is high, or remote data objects are accessed very frequently. Main reason that PCWFC outperforms for DMM-DBMSs is that query compilation and remote catalog access can be done in a very high speed, even when the catalogs of the remote data objects are frequently updated.

Key words: Main Memory Database(주기억장치 데이터베이스), Catalog(카탈로그), Distributed Database (분산데이터베이스), Performance Evaluation(성능평가)

※ 교신저자(Corresponding Author) : 홍의경, 주소 : 서울시 동대문구 전농동 90(130-743), 전화 : 02)2210-2497, FAX : 02)2210-5274, E-mail : ekhong@venus.uos.ac.kr

접수일 : 2004년 8월 5일, 완료일 : 2004년 11월 22일

[†] 서울시립대학교 컴퓨터과학부
(E-mail : hrjeong@venus.uos.ac.kr)

[††] 서울시립대학교 컴퓨터과학부 교수

[†††] 정회원, 이화여자대학교 컴퓨터학과 부교수
(E-mail : mkim@ewha.ac.kr)

※ 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

1. 서 론

주기억장치 데이터베이스 관리 시스템(MMDBMS)은 데이터베이스 전체를 주기억장치에 저장해 놓고 있으므로 디스크 기반의 DBMS에 비해 우수한 성능을 보장한다. 기존의 DBMS 기술들 중에 상당 부분은 그대로 MMDBMS에 적용하는 것이 가능하지만 디스크의 효율적인 관리에 최적화되어 있는 카탈로그 관리, 질의 최적화, 버퍼 풀 관리, 인덱스 기법들은 MMDBMS에 그대로 적용하는 것이 힘들다.

근거리 통신망(LAN)을 통해 데이터를 고속으로 전송하는 것이 가능해짐에 따라 MMDBMS는 주로 분산 환경에서 구현된다. 즉, 데이터베이스를 관련 있는 데이터 객체들의 그룹으로 분할하고, 각 그룹을 참여 사이트에 분산시켜 저장하는 것이다. 이러한 시스템을 분산 주기억장치 데이터베이스 관리 시스템(DMM-DBMS)이라고 한다.

최근 들어 DMM-DBMS에 대해 많은 연구 결과가 발표되었다[2,4,6,7,9-11,16]. 이들은 주로 동시성 제어(concurrency control), 완료 처리(commit processing), 접근 방법(access method), 질의 처리(query processing), 회복 기법(recovery), 성능 및 응용 프로그램 인터페이스 등에 관한 것들이다. 그러나 아직까지 DMM-DBMS의 카탈로그 관리 기법에 대한 성능 분석에 대한 결과는 발표된 것이 없다. 카탈로그 관리 기법은 매우 중요한 데이터베이스 관리 기법 중의 하나로 사이트 자치성(site autonomy)[12], 질의 최적화[13], 뷰 관리[1], 권한 관리기법(authorization mechanism)[17], 데이터의 투명성[12] 등에 큰 영향을 미친다.

카탈로그를 시스템 데이터베이스라고도 하며, 이는 데이터베이스 시스템이 관리하는 여러 객체들에 대한 정보를 포함한다[3]. 릴레이션, 뷰, 인덱스, 사용자, 액세스 모듈(실행 코드), 액세스 권한 등이 이러한 객체들로 분류된다. 카탈로그 자체도 사용자의 데이터 릴레이션과 마찬가지로 릴레이션들도 구성되어 있다. 따라서 릴레이션으로 저장되어 있는 카탈로그들도 사용자가 정의한 릴레이션들과 마찬가지로 질의어로 접근될 수 있고 동일한 동시성 제어와 회복 메카니즘의 적용을 받는다.

카탈로그는 사용자가 질의에서 정의한 객체를 낮은 수준의 객체 식별자와 객체의 위치에 사상하는 역할을 한다. 카탈로그는 데이터베이스 객체들의 스키마를 정의하고 이들에 접근할 수 있는 액세스 경로를 명시한다. 카탈로그는 또한 질의 최적화, 사용자의 데이터베이스 객체에 대한 인증, 데이터베이스 객체들 간의 의존성 등에 대한 통계 자료를 제공한다.

DMM-DBMS의 카탈로그는 중앙 집중식, 완전 중복식, 분할식으로 구분할 수 있다. 이들 중에서 분할식 카탈로그 관리 기법은 디스크 기반 분산 DBMS에서 많이 채택되어 사용되고 있으며 다른 기법들에 비해 높은 성능을 보이고 있다[8].

분할식 카탈로그 관리 기법은 원격 데이터 객체의 카탈로그를 언제, 얼마나 지역 사이트에 캐시하고 관리하는가에 따라 여러 개의 변형으로 분류될 수 있다[8]. 본 연구에서는 이러한 변형들을 합리적인 환경 설정을 통해 시뮬레이션함으로써 이러한 변형 기법들의 성능을 평가하고자 한다. 우선 DMM-DBMS 모델을 설계하였고, 이를 시뮬레이터로 구현하여 각 카탈로그 관리 기법들의 성능을 평가하였다.

본 연구에서 다루는 분할식 카탈로그 관리 기법들은 (1) 캐시가 없는 방식, (2) 점진적으로 캐시를 하는 방식, (3) 완전 캐시를 하는 방식 등 세 가지 방식으로 나뉜다. 방식 (1)과 (2)는 사이트가 과거에 참조되었던 원격 데이터 객체의 카탈로그의 사본을 지역 사이트에서 관리하는 것이 얼마나 효율적인가를 분석하기 위해 비교하였다. 방식 (2)와 (3)은 지역 사이트에 캐시해서 관리해는 카탈로그의 비율이 성능에 미치는 영향을 분석하기 위해 비교하였다. 참고로, 완전 캐시를 하는 방식은 각 사이트에 모든 원격 데이터 객체의 카탈로그 사본을 유지 관리한다는 것이 아니라, 점진적인 캐시 방식이 시간이 지남에 따라 관련 원격 카탈로그의 대부분을 가지고 관리하는 상태로 볼 수 있다.

시뮬레이션의 결과, 캐시 비율이 높으면 성능이 높아진다는 결과를 보였다. 이는 점진적 캐시 방식이 시간이 감에 따라 더욱 높은 성능을 보인다는 것을 뜻한다. 주요 이유는 근거리 통신망에서는 원격 데이터 객체의 카탈로그가 빈번히 갱신되는 경우라도 질의 컴파일과 원격 카탈로그 액세스가 고성능으로 처리될 수 있기 때문이다. 그러나 이러한 기법은 각 사이트의 부하가 크거나, 카탈로그 읽기 비율이 낮거나 원격 데이터 객체가 자주 액세스되는 경우에 기존의 디스크 기반 분산 DBMS에서는 좋은 성능을 보이지 못한다고 보고되었다[8].

본 논문은 다음과 같이 구성된다. 2절에서 질의 처리 단계와 질의 처리에 카탈로그가 어떻게 사용되는가를 간략하게 설명한다. 3절에서는 DMM-DBMS를 위한 분할식 카탈로그 관리 기법 세 가지를 소개한다. 4절에서는 DMM-DBMS 모델을 소개하고 시뮬레이션 환경과 시뮬레이션 매개변수들을 설명한다. 5절에서는 실험 종류와 실험 결과에 대해 설명하고, 마지막으로 6절에서 결론을 맺는다.

2. 질의 처리 단계

카탈로그 관리 기법들을 비교 분석하기 전에, 질의 처리 단계와 카탈로그가 질의 처리에 어떻게 참조되고 갱신되는가를 간략하게 설명하기로 한다. 디스크 기반의 DBMS들과는 달리 DMM-DBMS들의 카탈로그는 참여 사이트의 주기억 장치에 저장되어 있다. 즉, 카탈로그들은 디스크를 액세스하지 않고 해당 사이트의 주기억 장치에서 직접 읽히고 갱신된다.

질의 처리의 일곱 단계를 설명하기로 한다. 본 논문에서는 카탈로그 이름에 SQL/DS의 이름 부여 방식(naming convention)을 그대로 사용하기로 한다.

단계 1: (파싱) 카탈로그를 참조하지 않은 채로 질의의 구문을 점검한다.

단계 2: (이름 변환) 사용자가 정의해서 쓰는 이름들이 시스템 차원의 이름으로 변환된다. 이 때 릴레이션이나 뷰마다 SYSSYNONYMS 카탈로그가 한 번씩 읽혀진다.

단계 3: (카탈로그 검색) 질의가 참조하는 릴레이션마다 카탈로그 엔트리가 검색된다. 카탈로그 검색 기법에 따라 원격 사이트를 액세스할 수도 있다. 이 때 릴레이션마다 SYSCATALOG 카탈로그가 읽혀지고, 애트리뷰트마다 SYSCOLUMNS 카탈로그가 읽혀지며, 각 릴레이션의 인덱스마다 SYSINDEXES 카탈로그가 읽혀진다.

단계 4: (권한 검사) 질의가 참조하는 객체마다 사용자의 액세스 권한을 점검한다. 릴레이션마다 SYSTABAUTH 카탈로그가 적어도 한 번 이상 검색된다.

단계 5: (최적화) DBMS가 사용하는 질의 최적화 알고리즘을 통해 총 비용을 최소화하는 질의 액세스 계획이 생성된다.

단계 6: (액세스 모듈 생성) 최적화된 액세스 계획이 액세스 모듈로 컴파일된다. 이 계획은 SYSACCESS 카탈로그에 저장된다.

단계 7: (의존성 기록) 액세스 모듈은 객체들이나 액세스 경로, 인증의 존재성과 유효성에 의존적이다. 이러한 의존성은 SYSUSAGE 카탈로그에 기록된다.

3. 카탈로그 관리 기법

데이터베이스 질의는 일반적으로 카탈로그 읽기 질의와 쓰기 질의로 구분할 수 있다. 카탈로그 읽기 질의는 질의 처리를 위해 카탈로그로부터 데이터를 읽는 질의이고, 쓰기 질의는 카탈로그의 데이터를 갱신하는 질의를 말한다. 각 그룹에 속하는 SQL 명령문들은 표 1과 표 2와 같이 정리할 수 있다. 카탈로그 읽기 질의에는 데이터베이스 읽기 질의뿐만 아니라 데이터베이스 갱신 질의들도 포함된다.

분산 데이터베이스에서의 카탈로그 관리 기법은 크게 중앙 집중식 카탈로그(centralized catalogs), 완전 중복식 카탈로그(fully replicated catalogs), 분할식 카탈로그(partitioned catalogs)로 구분할 수 있다. 중앙 집중식 카탈로그 기법과 완전 중복식 카탈로그 기법은 분산 환경에 적합하지 않으나, 분할식 카탈로그 기법은 사이트의 자치성과 높은 성능을 보장한다[8,12]. 분할식 카탈로그 기법은 다시 크게 아래와 같이 3가지로 분류될 수 있으며[8], 본 논문에서는 분할식 카탈로그 기법들의 성능을 구체적으로 분석하고자 한다.

(1) 캐시가 없는 분할식 카탈로그(PCWC: Partitioned Catalogs Without Caching)

각 사이트는 그 사이트에 저장된 데이터 객체에 대한 카탈로그 정보만 관리한다. 원격 사이트에 저장된 데이터 객체들에 대한 카탈로그에 대한 캐시를 다른 사이트에서 유지하지 않으므로 원격 사이트의 데이터 객체를 접근하는 질의가 입력될 때마다 매번 원격 사이트의 카탈로그 정보를 참조하며, 원격 카탈로그 정보는 질의를 수행한 후 삭제된다[8].

표 1. 카탈로그에 대한 읽기 질의

연 산	기 능
SELECT	릴레이션으로부터 투플을 검색
INSERT	릴레이션에 투플을 삽입
DELETE	릴레이션으로부터 투플을 삭제
UPDATE	릴레이션의 투플을 수정

표 2. 카탈로그에 대한 쓰기 질의

연 산	기 능
CREATE TABLE	테이블을 생성
DROP TABLE	테이블을 삭제
ALTER TABLE	애트리뷰트를 추가
CREATE INDEX	인덱스를 추가
DROP INDEX	인덱스를 삭제
CREATE VIEW	뷰를 생성
DROP VIEW	뷰를 삭제
GRANT	사용자에게 권한을 부여
REVOKE	사용자에게 권한을 취소
CREATE SYNONYM	릴레이션이나 뷰의 별명을 정의
DROP SYNONYM	SYNONYM을 삭제
UPDATE STATISTICS	최적의 접근경로를 선택하기 위해 통계 정보를 갱신

(2) 점진적 캐시를 이용한 분할식 카탈로그 (PCWIC: Partitioned Catalogs With Incremental Caching)

각 사이트는 그 사이트에 저장된 데이터 객체에 대한 카탈로그 정보를 관리한다. 원격 데이터 객체의 카탈로그 정보는 해당 데이터 객체를 처음 액세스할 때 그 사이트로부터 읽어와서 캐시된다. 다음에 이 원격 데이터 객체를 참조하는 질의가 다시 입력되면 원격 사이트의 카탈로그를 참조하지 않고, 이 사이트에 캐시된 카탈로그를 이용하여 질의를 컴파일한다.

어떤 사이트의 카탈로그 정보가 변경될 때마다 다른 사이트들에 캐시되어 있는 카탈로그 정보도 갱신되지는 않는다. 캐시된 카탈로그를 최신 상태로 유지하는 대신에, 질의가 입력된 사이트에서 사용한 캐시의 최신 여부를 식별하기 위하여 버전 번호를 사용한다[10]. 질의가 입력된 사이트에서 최신 카탈로그를 사용하지 않은 것을 질의에서 참조된 데이터 객체를 저장하고 있는 원격 사이트에서 탐지하게 되면 질의가 입력된 사이트로 에러 메시지를 보낸다. 이런 경우에 질의가 입력된 사이트는 원격 사이트를 다시 접근하여 해당 데이터 객체의 최신 카탈로그를 읽어와서 질의를 다시 컴파일하며, 새로운 질의 액세스 계획을 수립한 후에 원격 사이트에 재분배한다. 시간이 지나감에 따라 각 사이트에 캐시된 카탈로그의 분량이 점점 증가할 수 있다.

(3) 완전 캐시를 이용한 분할식 카탈로그 (PCWFC: Partitioned Catalogs With Full Caching)

각 사이트는 지역 데이터 객체뿐만 아니라 모든 원격 데이터 객체에 대한 카탈로그 사본을 관리하며 그 이외에는 PCWIC와 동일하다. 각 사이트가 처음부터 모든 원격 카탈로그의 사본을 가지고 시작한다고 보는 것은 비현실적이며, PCWFC는 PCWIC로 시작하여 필요한 원격 카탈로그의 거의 대부분이 캐시된 상태라고 보는 것이 타당하다. 따라서 원격 데이터 객체를 참조하는 질의도 원격 사이트의 카탈로그를 접근할 필요 없이 지역 사이트에 캐시된 카탈로그만 참조하여 컴파일할 수 있다.

본 논문에서는 위에서 언급한 세 가지 기법의 성능을 평가한다. PCWC와 PCWIC와의 비교 실험은 캐시 효과를 분석하기 위한 것이다. 즉, 과거에 참조했던 데이터의 카탈로그 정보를 지역 사이트에 저장하고 관리하는 것이 얼마나 효율적인가를 분석한다. 이에 반해서 PCWIC와 PCWFC와의 비교 실험은 캐시 비율의 효과를 분석하기 위한 것이다.

4. 카탈로그 관리 기법들의 성능평가

카탈로그 관리 기법들의 성능을 평가하기 위해 우선 데이터베이스 사이트 모델을 정의하였다. 이 모델상에서 적절한 환경 설정을 한 후, 각 카탈로그 관리 기법의 성능을 시뮬레이션을 통해 평가하였다. 이 절에서는 시뮬레이션 환경 설정, 시뮬레이션 모델, 성능평가 기준, 시뮬레이터의 구조를 설명하기로 한다. 우선 환경은 다음과 같이 설정하였다.

- 응용의 질의는 카탈로그 읽기 및 쓰기 질의를 포함한다.
- 동시성 제어 기법은 2 단계 로킹 알고리즘(two-phase locking algorithm(2PL))[5]을 사용한다.
- 교착상태 방지를 위해서는 wound-wait 알고리즘[15]을 사용한다.
- 각 트랜잭션의 원자적 완료를 위해 2 단계 완료 규약(two-phase commit protocol: 2PC)[5]을 사용한다.
- 데이터 로킹과 접근은 페이지 단위로 한다.
- 네트워크는 완전히 연결되어 있고 방송 기능을 가진다.

• 사이트 고장이나 네트워크 연결 정지는 발생하지 않는다.

(1) 시뮬레이션 모델

시뮬레이션에 사용할 DMM-DBMS 모델을 소개하기로 한다. DMM-DBMS는 근거리 통신망에 연결되어 있는 여러 개의 DB 사이트들로 구성되며, 각 DB 사이트는 그림 1과 같은 구조를 갖는다. 한 사이트는 여러 명의 사용자(또는 터미널), CPU, 디스크와 5개의 큐로 구성된다. CPU는 '일반연산 큐(operation queue)'나 '메시지 큐(message queue)'에 있는 요청들을 처리한다. 일반연산 큐에서 대기하고 있는 요청들은 질의 컴파일, 동시성 제어, 2-단계 완료 처리, 카탈로그 읽기/쓰기 등이다. 메시지 큐에서 대기하는 트랜잭션은 다른 사이트로부터 받은 요청들이거나 다른 사이트가 처리해야 할 요청들이다. 메시지 큐에서 대기하는 요청들은 일반연산 큐에서 대기하는 요청들보다 우선순위가 높다. 따라서 CPU는 메시지 큐에서 대기하는 요청이 없는 경우에 일반연산 큐에서 대기하는 요청을 처리한다.

또한 DB 사이트에는 '지연 큐(blocked queue)', '채널 큐(channel queue)', '디스크 큐(disk queue)'가 있다. '지연 큐(blocked queue)'에는 동시성 제어로 인해 지연되는 요청들이 저장된다. 이러한 요청들은 지연 조건이 해결되는 순간에 일반연산 큐로 옮겨진다. '채널 큐'에는 네트워크를 통해 다른 사이트로 보내질 요청들이 저장된다. 네트워크를 통해 전송이 가능하게 되면 이러한 요청들은 해당 사이트로 순서대로 보내진다. '디스크 큐'에는 2-단계 로그 레코드를 디스크에 쓰려는 요청들이 대기한다. 로그 레코드는 메모리의 로그버퍼가 가득 차게 될 경우에만 디스크에 기록된다.

DB 사이트 모델에서의 질의 처리 흐름을 설명하기 위해 카탈로그 읽기 질의 처리 과정을 간략하게 소개하기로 한다. 카탈로그 관리 기법으로는 PCWIC 기법을 사용한다고 가정해 보자. 터미널에서 카탈로그 읽기 질의가 생성되면, 이 질의는 일반연산 큐에 삽입된다. 질의는 파싱 단계와 이름 변환 단계를 거친다. 질의가 지역 릴레이션들만 참조하는 경우는 이름 변환 단계가 지역 사이트에서 처리된다. 그러나 질의가 카탈로그가 캐시되어 있지 않은 원격 릴레이션을 참조하는 경우에는 해당 원격 카탈로그를 검색하는 요청이 생성되어 메시지 큐에 삽입된다. 메시지 큐에 있는 이러한 요청들은 나중에 채널 큐로 옮겨진다. 카탈로그 읽기 질의는 카탈로그 엔트리의 동시성

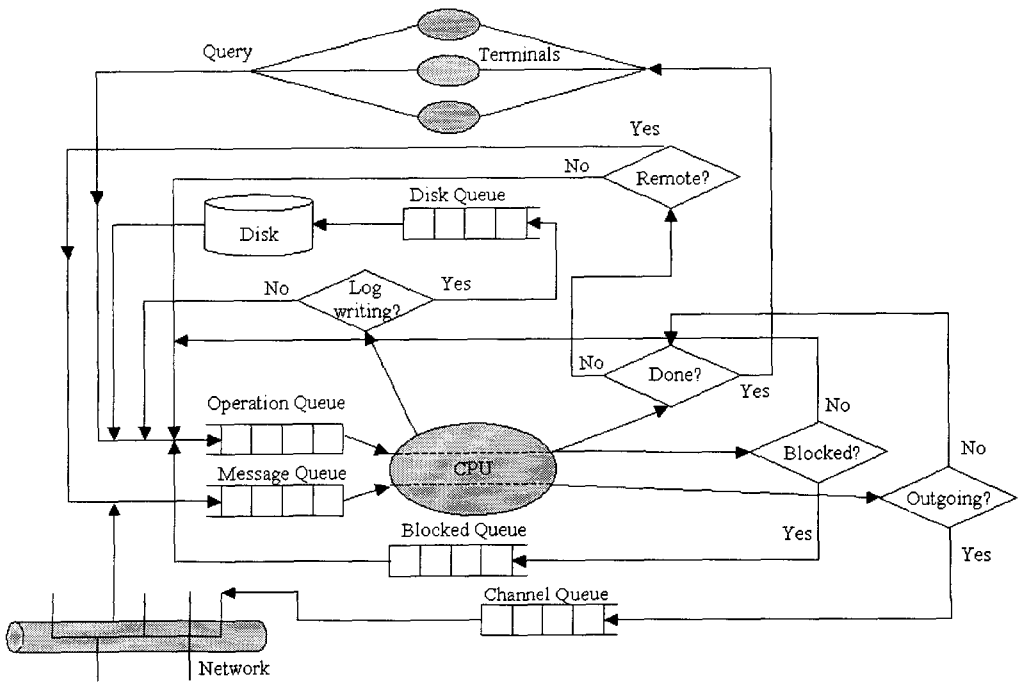


그림 1. 데이터베이스 사이트 모델.

제어를 해결하기 위해 일반 연산 큐에 다시 삽입된다. 카탈로그 읽기 질의는 카탈로그 엔트리를 공유 모드(shared mode)로 로크하고 카탈로그 갱신 질의는 카탈로그 엔트리를 독점 모드(exclusive mode)로 로크한다. 동시성 제어 요청이 해결되면, 질의는 일반연산 큐에서 대기하면서 카탈로그 검색, 권한 검사, 질의 최적화 단계를 차례로 거치면서 처리된다. 동시성 제어가 해결되지 못한 질의들은 지연 큐에서 대기하게 된다.

질의에 대한 전역적 액세스 계획이 완성되면, 해당 액세스 계획이 메시지 큐에 삽입된다. 이러한 요청은 다시 채널 큐로 옮겨지고 나중에는 다른 사이트로 보내진다. 분산 계획은 질의 명령문 원본, 카탈로그 엔트리, 카탈로그 엔트리의 버전 번호와 전역적 질의 액세스 계획으로 구성된다. 캐시된 카탈로그 엔트리가 최신 버전인 경우에는 질의 처리가 여기에서 완성되고 종료된다. 그렇지 않은 경우의 설명은 생략하기로 한다.

(2) 성능 지수와 시뮬레이터

성능 평가를 위한 일반적인 지수(performance index)로 응답 시간(response time)과 처리율(throughput)을 들 수 있다. 대부분의 경우에 두 지수가 비슷한 결과를 나타내므로 본 실험에서는 응답 시간을 성능 지수로 사용하기로 한다. 질의 응답 시간은 질의 발생시간으로부터 질의 수행이 완료되어 결과가 반환될 때가지 경과된 시간이다. 질의가 철회(abort)된 경우에는 새로운 질의가 자동으로 시작되고 철회된 시점까지의 수행 시간도 응답 시간 측정에 포함된다. 응답 시간 이외에 보조 성능 지수로서 큐 대기 시간을 사용하였는데, 큐 대기 시간을 통해 어떤 자원에서 병목 현상이 발생하였는가를 파악할 수 있다.

시뮬레이터의 입력 매개변수로는 시스템을 위한 매개변수들, 질의 생성과 자원 관리를 위한 매개변수들로 구성되며 이들은 표 3에 정리되어 있다. 매개변수들의 수치 값은 주로[7,16] 문헌을 근거로 하였다.

각 카탈로그 관리 기법마다 시뮬레이터를 구축하였으며, 시뮬레이터 구축에는 SimJava [14]를 사용하였다. 이는 에딘버러 대학교에서 개발된 자바 기반의 시뮬레이션 패키지이다. SimJava는 프로세스 중심의 패키지이므로 모든 프로세스를 동시에 실행시킬 수 있도록 설계되었다. 프로세스를 특정 기간 동안 정지시킬 수도 있고 특정 이벤트가 발생할 때까지 대기하도록 할 수도 있다. 시뮬레이터는 Windows NT 5.0이 실행되는 인텔 펜티엄 IV PC에서 구현되었다. 분산 주기억장치 데이터베이스 시스템은 Sun Enterprise 450 시스템에서 실행되며, 네트워크 대역폭은 100Mbps로 가정하였다.

표 3. 시뮬레이션에 사용되는 매개 변수들

1) 시스템을 위한 매개변수

NumOfSites	3~15	사이트 수
NumOfTerminals	3~15	사이트당 터미널 수
NomOfPages	5000	전체 페이지 수

2) 트랜잭션 생성을 위한 매개변수

MeanThinkTime	5000msec	트랜잭션 생성 빈도
ReadXactRatio	60~90%	읽기 연산의 비율
AccessPages	1~20	트랜잭션당 접근 페이지 수
LocalQueryRatio	60~90%	현재 사이트에서 질의 접근 비율

3) 자원 관리를 위한 매개변수

XactCountPerTerminal	100~500	한 터미널이 처리해야 할 트랜잭션 개수
PageSize	4KB	페이지의 크기
LogBufferSize	128~512KB	로그 버퍼의 크기
LogFileSize	1MB	로그 파일의 크기
LogWriteTime	20msec	로그 기록 시간
MessaegProcessingTime	0.4msec	메시지 처리 시간
LockAcquireTime	0.05msec	로크 획득 시간
LockReleaseTime	0.05msec	로크 반환 시간
DataReadingTime	0.2msec	데이타 읽는 시간
ParsingTime	1msec	질의 파싱 시간
NameTime	0.2msec	이름 변환 시간
CatalogLookupTime	0.5msec	카탈로그 찾는 시간
AuthorityCheckingTime	0.2msec	권한 검사 시간
OptimizationTime	2msec	질의 최적화 시간
CodingTime	1msec	코드 생성 시간
CPU Hz	480MHz	중앙 처리 장치의 클럭 수
NetTransTime	0.08msec	네트워크 전송 시간

5. 성능평가를 위한 실험 및 결과

3절에서 설명한 세 가지 카탈로그 관리 기법들의 성능을 사이트 수, 사이트 당 터미널의 수, 로그 버퍼의 크기, 카탈로그 읽기 연산의 비율, 지역 질의 비율을 변화시켜 가면서 다양하게 분석하였다. 특별한 언급이 없는 한, 실험에서는 시스템에 7개의 사이트가 있고, 각 사이트마다 7개의 터미널이 있다고 가정한다. 로그 버퍼의 크기는 512K 바이트로 가정한다. 카탈로그 읽기 질의 비율과 지역 질의 비율은 모두 80%로 가정하고, 각 실험에는 5,000개의 질의를 사용하였다.

(1) 실험 1: 사이트 수의 변화에 따른 성능평가

이 실험에서는 사이트 수를 3~15개로 변화시켜 가면서 카탈로그 읽기/쓰기 질의의 평균 응답 시간을 측정하였다. PCWFC가 가장 우수한 성능을 나타냈고, 실험 결과는 그림 2와 그림 3에 나타나 있다. PCWFC 기법의 경우, 각 사이트는 질의의 80%에 해당하는 카탈로그 읽기 질의를 처리하기 위해 캐시된 카탈로그를 사용하기 때문이다. 반면에 PCWC의 경우 각 사이트는 원격 카탈로그가 필요한 경우 항상 원격 사이트의 카탈로그를 가져오므로, 네트워크를 통한 데이터 전송과 메시지 처리를 위한 오버헤드가 발생한다.

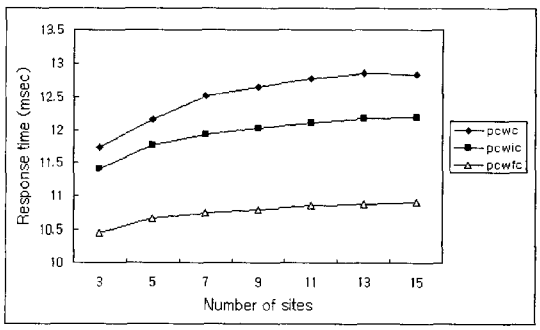


그림 2. 사이트 수 변화에 따른 읽기 트랜잭션의 평균 응답 시간

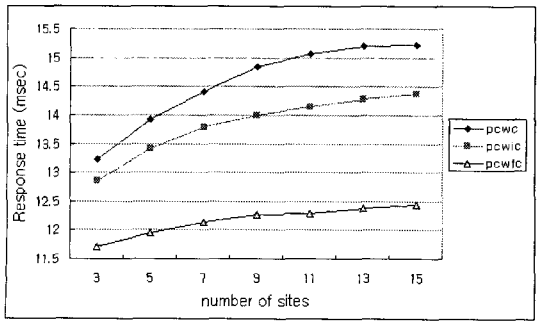


그림 3. 사이트 수 변화에 따른 쓰기 트랜잭션의 평균 응답 시간

사이트 수가 적은 경우에는 응답 시간이 사이트 개수에 비례한다. 그러나 사이트의 수가 10개를 초과하게 되면 응답 시간은 거의 일정한 수준을 유지한다. 주요 이유는 질의 처리를 위해 원격 사이트에서 갱신된 카탈로그를 검색하는 회수는 시스템의 사이트 수와 큰 관계가 없기 때문이다. 그림 2와 그림 3을 비교해 보면 쓰기 질의가 읽기 질의에 비해 처리 시간이 긴데, 그 이유는 동시성 제어를 위해 로킹 오버헤드가 더 들기 때문이다.

표 4에 자원 사용률과 큐 대기 시간에 대한 분석 결과가 있다. 시스템에는 CPU, DISK, QUEUE의 3

표 4. 사이트 수 변화에 따른 자원 사용 비율 및 큐 대기 시간

자원 사용 비율

사이트 개수	자원	PCWC	PCWIC	PCWFC
3	CPU	55.80%	59.10%	67.90%
	Disk	0.00%	0.00%	0.00%
	Queue	44.20%	40.90%	32.10%
9	CPU	45.50%	50.30%	58.80%
	Disk	0.00%	0.00%	0.00%
	Queue	54.50%	49.70%	41.20%
15	CPU	43.90%	48.70%	57.20%
	Disk	0.00%	0.00%	0.00%
	Queue	56.10%	51.30%	42.80%

큐 대기 시간

사이트 개수	큐 종류	PCWC	PCWIC	PCWFC
3	일반연산큐	15.20%	15.60%	35.30%
	메시지 큐	53.40%	54.90%	43.00%
	채널 큐	31.40%	29.50%	21.70%
9	일반연산큐	8.50%	8.50%	21.40%
	메시지 큐	55.60%	57.30%	50.10%
	채널 큐	35.90%	34.20%	28.50%
15	일반연산큐	7.70%	7.70%	19.60%
	메시지 큐	55.80%	57.40%	50.90%
	채널 큐	36.50%	34.90%	29.50%

종류의 자원이 있다. 그림 1에서와 같이 시스템에는 5개의 큐가 있다. 이들은 일반연산 큐, 메시지 큐, 지연 큐, 디스크 큐, 채널 큐이다. 표 4를 보면 디스크가 전혀 사용되고 있지 않다는 것을 알 수 있는데, 그 이유는 디스크 액세스가 필요 없을 정도로 512K 크기의 로그 버퍼가 충분하다는 것을 나타낸다. 표 4의 분석 자료에 지연 큐는 포함되지 않았다. 총 페이지 수는 5,000이고, 질의가 액세스하는 평균 페이지 수는 10이며, 카탈로그 읽기 질의 비율이 80%이므로 이러한 경우에는 질의 처리가 지연되지 않기 때문이다.

표 4와 같이 PCWFC는 다른 두 기법보다 CPU를 더 많이 사용한다. PCWC에서 큐 대기 시간이 더 길며, 이는 PCWFC와 PCWIC가 지역 사이트에 캐시된 카탈로그 관리를 위해 CPU를 더 많이 쓰기 때문이다. 반면 PCWC는 다른 두 기법에 비해 원격 자원을 더 많이 쓴다. 세 기법 모두 사이트 수가 증가함에 따라 큐 대기 시간이 길어지며 CPU 사용은 줄어든다. 이는 사이트 수가 증가함에 따라 카탈로그 관리를 위한 오버헤드가 늘어난다는 것을 뜻한다.

(2) 실험 2: 사이트 당 터미널 수의 변화에 따른 성능평가

이 실험에서는 사이트 당 터미널 수를 3~15까지 변화시켰으며, 평가 결과는 그림 4에 있다. 터미널 수가 증가함에 따라 질의 응답 시간이 증가하기는 하지만 큰 차이가 없다. 또한 PCWFC는 안정적이며 다른 두 기법에 비해 성능이 크게 우수하다는 것을 알 수 있다. 표 5는 터미널 수 변화에 따른 자원 사용율과 큐 대기 시간 변화를 보인다. PCWIC와 PCWFC는 메모리상에 캐시를 관리하므로 CPU 사용 시간은 길지만 큐 대기 시간은 짧다는 것을 알 수 있다. 터미널 수가 증가하면 세 기법 모두 CPU

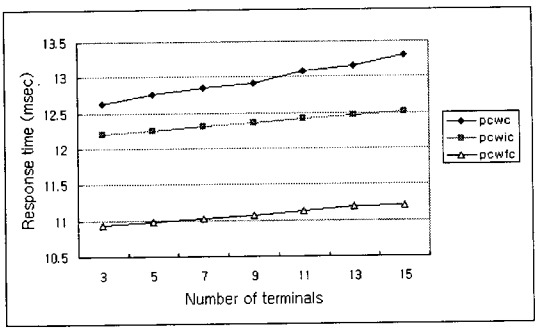


그림 4. 터미널 수 변화에 따른 카탈로그 읽기/쓰기 질의 평균 응답 시간

표 5. 터미널 수 변화에 따른 자원 사용 비율과 큐 대기 시간

자원 사용 비율

터미널 개수	자원	PCWC	PCWIC	PCWFC
3	CPU	47.50%	51.70%	60.50%
	디스크	0.00%	0.00%	0.00%
	큐	52.50%	48.30%	39.50%
9	CPU	46.90%	51.20%	60.10%
	디스크	0.00%	0.00%	0.00%
	큐	53.10%	18.80%	39.90%
15	CPU	46.10%	50.70%	59.80%
	디스크	0.00%	0.00%	0.00%
	큐	53.90%	49.30%	40.20%

큐 대기 시간

터미널 개수	큐 종류	PCWC	PCWIC	PCWFC
3	일반연산 큐	9.00%	9.20%	23.10%
	메시지 큐	56.10%	57.40%	49.50%
	채널 큐	34.90%	33.40%	27.40%
9	일반연산 큐	9.00%	9.20%	23.00%
	메시지 큐	55.70%	57.10%	49.30%
	채널 큐	35.30%	33.70%	27.70%
15	일반연산 큐	8.90%	9.20%	22.90%
	메시지 큐	55.30%	56.80%	49.20%
	채널 큐	35.80%	34.00%	27.90%

사용이 줄어들고 큐 대기 시간이 길어지기는 하지만 그 차이는 그리 크지 않다.

(3) 실험 3: 로그 버퍼의 크기 변화에 따른 성능평가

이 실험에서는 로그 버퍼의 크기를 128K~512K 바이트로 변화시키면서 버퍼 크기가 카탈로그 관리 기법의 성능에 미치는 영향을 평가하였으며, 평가 결과는 그림 5~7에 나타나 있다. 그림 5에서와 같이 버퍼의 크기가 128K 바이트 정도로 작은 경우에는 디스크 액세스가 빈번하게 발생한다. 세 기법 모두 사이트 수가 증가함에 따라 응답 시간이 길어지다가 사이트 수가 9에 이르면 성능이 안정적이라는 것을

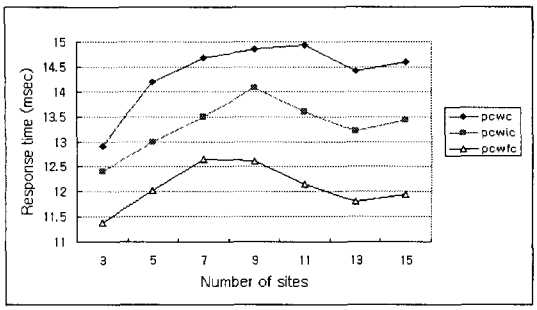


그림 5. 카탈로그 읽기/쓰기 질의 평균 응답 시간 (로그 버퍼 크기: 128K 바이트)

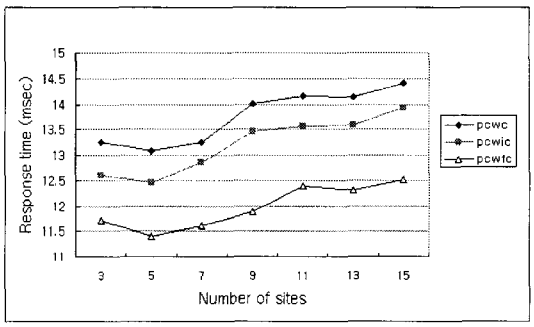


그림 6. 카탈로그 읽기/쓰기 질의 평균 응답 시간 (로그 버퍼 크기: 256K 바이트)

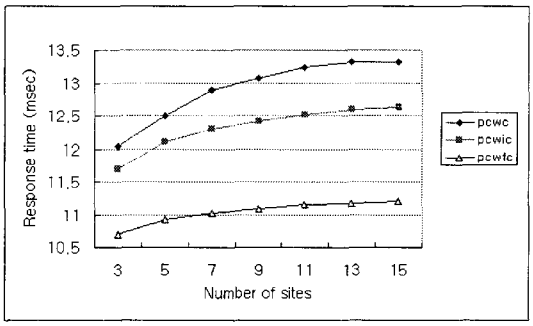


그림 7. 카탈로그 읽기/쓰기 질의 평균 응답 시간 (로그 버퍼 크기: 512K 바이트)

알 수 있다. 특히 모든 경우에 PCWFC가 PCWIC와 PCWC에 비해 우수한 성능을 나타낸다.

(4) 실험 4: 카탈로그 읽기 질의 비율에 따른 성능평가

이 실험에서는 사이트의 수를 7로 고정하고, 카탈로그 읽기 질의 비율을 60%~90%로 변화시켜 보았다. 실험 결과는 그림 8에 있으며, 모든 경우에 카탈로그 읽기 질의 비율이 높아짐에 따라서 응답 시간이 약간이나마 줄어든다는 것을 알 수 있다. 또한 모든

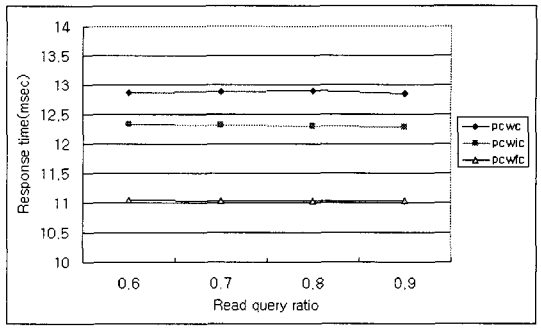


그림 8. 카탈로그 읽기 질의 평균 응답 시간 (사이트 수 : 7개)

경우에 PCWFC가 다른 두 기법에 비해 성능이 우수하였다. 카탈로그 읽기 질의 비율이 카탈로그 쓰기 질의 비율에 비해 낮은 경우에는 로그 버퍼가 512K 바이트만큼 큰 경우에도 로그를 디스크에 쓰기 위해 디스크를 빈번하게 액세스하며, 현실적이지 못하므로 실험에서 고려하지 않았다. 사이트의 수가 5인 경우와 9인 경우에도 이 실험을 하였는데, 비슷한 결과를 얻었다.

(5) 실험 5: 지역질의 비율에 따른 성능평가

이 실험에서는 사이트의 수를 7로 고정하고, 지역 질의 비율을 60%~90%로 변화시키면서 카탈로그 관리 기법들의 성능을 평가하였다. 지역 질의는 원격 사이트의 데이터 객체를 접근하지 않는 질의를 뜻한다.

지역 질의의 비율이 커질수록 PCWC와 PCWIC의 성능이 그림 9에서와 같이 크게 증가한다. 이는 원격 데이터 객체의 카탈로그 액세스 비율이 감소함에 따라 네트워크를 통한 데이터 전송 오버헤드가 줄어들기 때문이다. 그림에서와 같이 PCWFC의 경우는 지역 질의 비율의 변화에 성능이 민감하지 않다는 것을 알 수 있다. 즉 대부분의 질의가 캐시되어 있는 카탈

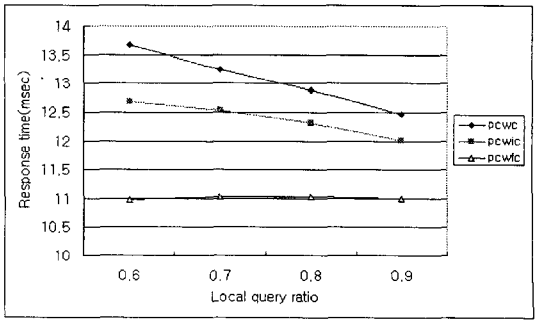


그림 9. 지역 질의 비율에 따른 카탈로그 읽기/쓰기 질의 평균 응답 시간 (사이트 수 7개).

로그로 처리되고 있기 때문이다. 이 실험을 사이트 수가 5일 때와 9일 때로 가정하고 시행하여 보았으며, 유사한 결과를 얻을 수 있었다.

6. 결 론

본 연구에서는 분산 주기억장치 DBMS(DMM-DBMS)에 사용가능한 여러 가지 카탈로그 관리 기법들을 소개하고 성능 평가를 수행하였다. DMM-DBMS는 카탈로그를 주기억장치에 상주시키기 때문에, 기존의 디스크 기반의 분산 DBMS의 카탈로그 관리 기법들이 DMM-DBMS에서도 유사한 성능을 나타낸다는 보장이 없기 때문에 이러한 분석은 필수적이다. 실험을 통해 세 종류의 분할식 카탈로그 관리 기법의 성능을 평가하였다.

우선 DMM-DBMS 모델을 제안하였고, 각 기법을 시뮬레이션하였다. 원격 데이터 객체의 카탈로그를 지역 사이트에서 관리하는 것의 효율성과 지역 사이트에 캐시해야 하는 카탈로그의 비율이 성능에 미치는 영향을 중심으로 실험을 하였다. 실험 결과, 캐시 비율이 높은 기법의 성능이 높다는 것을 보였다. 실험 결과는 점진적으로 캐시 비율을 높이는 기법은 시간이 감에 따라 성능이 높아진다는 것을 뜻한다. 다시 말하면, 각 사이트는 궁극적으로 원격 데이터 객체의 카탈로그를 모두 가지게 되어 거의 대부분의 연산이 캐시된 카탈로그로 해결된다는 것을 의미한다. 원격 사이트의 카탈로그가 자주 갱신되는 경우에도 캐시를 많이 해 놓는 것이 더 유리하다는 결과도 보였다. 주요 이유는 질의 컴파일과 원격 카탈로그 액세스가 고속으로 이루어질 수 있기 때문이다. 그러나 분산 주기억장치 DBMS에서 좋은 성능을 보이는 카탈로그 관리 기법은 각 사이트의 부하가 크거나 카탈로그 읽기 비율이 낮거나 원격 데이터 객체의 액세스가 매우 빈번한 경우에 디스크 기반의 분산 DBMS한테는 좋은 성능을 보장하지 못한다.

참 고 문 헌

[1] E. Bertino, L. M. Haas, and B. G. Lindsay, "View Management in Distributed Database Systems," Proceedings of the 9th International Conference on Very Large Data Bases, pp. 376-378, 1983.

[2] D. J. DeWitt, R. H. Katz, F. Olken, L. D. Shapiro, M. Stonebraker, and D. A. Wood, "Implementation Techniques for Main Memory Database Systems," Proceedings of the ACM SIGMOD Int. Conf. on Management of Data, Vol. 14, No. 2, pp. 1-8, 1984.

[3] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 3th Edition, Addison-Wesley, Boston, MA, 2000.

[4] H. Garcia-Molina and K. Salem, "Main Memory Database System," IEEE Trans. on Knowledge and Data Engineering, Vol. 4, No. 6, pp. 509-516, 1992.

[5] J. Gray, "Notes on Data Base Operating Systems," Lecture Notes in Computer Science, Vol. 60, pp. 393-481, 1978.

[6] J. Griffioen, T. Anderson, Y. Breitbart, and R. Vingralek, "DERBY : A Memory Management System for Distributed Main Memory Databases," Proc. 6th International Workshop on Research Issues in Data Engineering, pp. 150-159, 1996.

[7] M. Han and Y. Yoon, "An Implementation and Performance Analysis of Backup System using Concurrent Processing in Real-time DBMS," Proceedings of the 4th International Workshop on Real-Time Computing Systems and Applications (RTCSA '97), pp. 118-126, 1997.

[8] E. K. Hong and J. W. Cho, "Performance Evaluation of Catalog Management Schemes in Distributed Database Systems," Information Systems, Vol. 16, No. 2, pp. 125-144, 1991.

[9] H. V. Jagadish, D. Lieuwen, R. Rastogi, and A. Silberschatz, "Dali: A High Performance Main Memory Storage Manager," Proceedings of the 20th International Conference on Very Large Data Bases, pp. 48-59, 1994.

[10] T. J. Lehman, and M. J. Carey, "A Study of Index Structures for Main Memory Database Management Systems," Proceedings of the

12th International Conference on Very Large Data Bases, pp. 294-303, 1986.

[11] T. J. Lehman, E. J. Shekita and L. F. Cabrera, "An Evaluation of Starburst's Memory Resident Storage Component," IEEE Trans. on Knowledge and Data Engineering, Vol. 4, No. 6, pp. 555-566, 1992.

[12] B. G. Lindsay and P. G. Selinger, "Site Autonomy Issues in R*: A Distributed Database Management System," IBM Research Report RJ2927, San Jose, Calif., Sept. 1980.

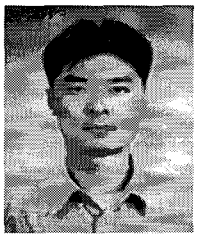
[13] G. M. Lohman, C. Mohan, L. Haas, D. Daniels, B. Lindsay, P. Selinger, and P. Wilms, "Query Processing in R*: A Distributed Database Manager," IBM Research Report RJ3740, San Jose, Calif., Jan. 1983.

[14] R. McNab and F. W. Howell, "Using Java for Discrete Event Simulation," Proceedings of the 12th UK Computer and Telecom Performance Engineering Workshop, pp. 219-228, 1991.

[15] D. J. Rosenkrantz, et al., "System Level Concurrency Control for Distributed Database Systems," ACM Trans. on Database Systems, Vol. 3, No. 2, pp. 178-198, 1978.

[16] The TimesTen Team, "In-Memory Data Management for Consumer Transactions," Proceedings of the ACM SIGMOD Conference., pp. 528-529, 1999.

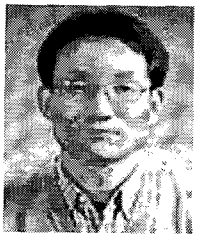
[17] P. F. Wilms and B. G. Lindsay, "A Database Authorization Mechanism Supporting Individual and Group Authorization," IBM Research Report RJ3137, San Jose, Calif., May 1981.



정 한 라

2000년 서울시립대학교 컴퓨터통계학과 졸업(B.S.)
2002년 서울시립대학교 컴퓨터통계학과 졸업(M.S.)
2001년~현재 한국인터넷정보센터(NIDA)

관심분야 : 분산주기억장치 데이터베이스, 웹 서버 보안, DNS(Domain Name Server)



홍 의 경

1981년 서울대학교 사범대학 수학교육과 졸업(B.S.)
1983년 한국과학기술원 전산학과 졸업(M.S.)
1991년 한국과학기술원 전산학과 졸업(Ph.D.)
1984년~현재 서울시립대학교 컴퓨터과학부 교수

관심분야 : 데이터베이스, XML, 데이터 마이닝

김 명

1981년 이화여자대학교 수학과 졸업(B.S.)
1983년 서울대학교 계산통계학과 졸업(M.S.)
1993년 캘리포니아 주립대학교 (산타바바라) 컴퓨터학과 졸업 (Ph.D)
1993년~1994년 캘리포니아 주립대학교 (산타바바라) 컴퓨터학과 강사, Postdoc.
1995년~현재 이화여자대학교 컴퓨터학과 부교수

관심분야 : 고성능 컴퓨팅, 지식공학, 웹기반 컴퓨팅, 시스템 모니터링