실시간 검색을 위한 다중 사용자용 주기억장치 자료저장 시스템 개발

권 오 수[†] · 홍 동 권[††]

요 약

주기억장치 자료저장 시스템은 실시간 트랜잭션에 충분한 여유 시간을 부여하여 실시간 트랜잭션의 성능을 높여준다. 이런 특성으로 인하여 주기억장치를 이용한 데이터관리 시스템들이 이동 통신 관리 시스템의 가입자 위치 관리와 같은 여유시간이 급박한 실시간 트랜잭션에 많이 활용되고 있다. 본 논문에서는 다중 사용자용 멀티쓰레드 방식의 실시간 검색 시스템 개발의 일환으로 대량의 실시간 검색 트랜잭션과 자료 변경 트랜잭션이 주기억장치 자료저장 시스템의 자료를 검색, 변경하는 환경에서 주기억장치 자료저장 시스템을 설계, 구현하였다. 구현된 시스템은 기존의 디스크 데이터베이스 시스템과 상호 보완적인 형태로 사용되는 내장형 방식으로 다중 쓰레드 방식으로 동작되며, 동시성 제어는 주기억장치의 특성을 살려 복잡한 잠금 방식이 아닌 래치를 사용한다. 주기억장치 자료저장 시스템에는 가장 최근의 데이터만을 저장하며, 동기화 기법으로는 디스크 데이터베이스 시스템에서 변경 트랜잭션이 발생하는 경우에 구현 시스템의 데이터를 갱신하는 방법을 사용한다. 시스템은 검색 및 변경 쓰레드의 비율을 제한하여 실시간 검색의 최소 성능을 보장할 수 있게 하였다.

Design and Implementation of a Main-memory Storage System for Real-time Retrievals

Oh-Su Kwon[†] · Dong-Kweon Hong[††]

ABSTRACT

Main memory storage system can increase the performance of the system by assigning enough slack time to real-time transactions. Due to its high response time of main memory devices, main memory resident data management systems have been used for location management of personal mobile clients to cope with urgent location related operations. In this paper we have developed a multi-threaded main memory storage system as a core component of real-time retrieval system to handle a huge amount of readers and writers of main memory resident data. The storage system is implemented as an embedded component which is working with the help of a disk resident database system. It uses multi-threaded executions and utilizes latches for its concurrency control rather than complex locking method. It only saves most recent data on main memory and data synchronization is done only when disk resident database asks for update transactions. The system controls the number of read threads and update threads to guarantee the minimum requirements of real-time retrievals.

키워드 : 실시간 검색(Real-time Retrieval), 주기억장치 상주형 자료 저장 시스템(Main-memory Resident Storage System), 동시성 제어(concurrency Control), 트랜잭션 스케줄링(Transaction Scheduling)

1. 서 론

실시간 시스템에서 다루는 데이터의 효율적인 저장 및 검색의 필요성과 데이터베이스 영역에서 보다 빠른 데이터베이스의 응답의 필요성이 인식되면서 데이터베이스 기술과 실시간 시스템의 접목에 대한 다양한 연구가 이루어지고 있다. 데이터베이스의 기본 연산 단위인 트랜잭션이 마감시간(deadline)과 같은 시간적 제약조건이 있다는 특징을 가지고 대용량 자료의 효율적인 처리를 동시에 수용하고, 실시간 시스템에 포함(embedded)되거나 독립적으로 수행되는 형태의 시스템을 실시간 데이터베이스 시스템(Real-Time Database System)[14, 16]이라 한다. 실시간 데이터베이스 시스템에 대한 연구 분야는 동시성 제어[11, 12, 18], 우선 순위 부여 방식[4, 7, 10], 회복 기능, 주기억장치 상주형 데이터베이스[8], 실시간 인덱스 기법 등이 있으며, 실시간 데이터베이스 시스템의 시험제품을 실제로 제작하는 구현 방법 및 응용분야로의 적용 방법 등도 활발히 진행되고 있다.

실시간 데이터베이스 시스템의 응용 예로는 주식 시장 응용, 레이더 추적 시스템, 공장 자동화 시스템, 교통 제어 시스템 등을 들 수 있다. 이들 시스템들은 모두 즉각적인 반응과 제한된 시간안에 필요한 동작이 이루어져야 시스템의 효율성이 높아진다. 기존의 데이터베이스 시스템은 보다 안전한 데이터의 저장으로 영속성을 제공하나 만족스럽지 못한

† 준 회 원 : (주)이니텍
†† 종신회원 : 계명대학교 컴퓨터공학 교수
논문접수 : 2002년 4월 3일, 심사완료 : 2002년 9월 9일

응답시간과 예측가능성(predictability)의 어려움으로 인한 부적합성 때문에 실시간 응용 분야로의 적용이 쉽지 않다. 기존의 데이터베이스 시스템에서는 대부분의 트랜잭션 처리가 보조기억장치에 저장되어 있는 데이터베이스에 대한 액세스를 필요로 한다. 이로 인하여 트랜잭션의 응답 시간은 디스크 액세스 지연시간에 의하여 제약을 받는다. 이러한 데이터베이스는 수 초 정도의 응답 시간이면 만족하는 일반 사용자를 대상으로 하는 일반적인 응용 분야에서는 충분하지만 실시간 응용 분야에서는 필요한 만큼의 빠른 응답 시간을 제공하기 어렵다. 이런 디스크 기반의 데이터베이스 시스템의 문제점을 감소시키기 위한 대안으로 고려되는 것이 주기억장치 상주형 데이터베이스 시스템이다. 주기억장치 상주형 데이터베이스 시스템은 주기억장치에 데이터베이스 시스템을 구축함으로서 디스크 입출력이 많은 디스크 상주형 데이터베이스 시스템에 비해 높은 성능과 빠른 응답 시간을 기대하는 것이다. 디스크 입출력이 적고 메모리가 고속으로 동작하기 때문에 트랜잭션의 예측가능성과 적시성이 높아지므로, 실시간 데이터베이스 시스템을 구축할 때 매우 유리하여 디스크 상주형 데이터베이스를 대체하기 위한 연구로 진행되고 있다. 현재 많은 연구 기관에서 주 메모리 상주형 데이터베이스에 대한 연구가 활발히 진행중이고, 이미 여러 분야에서 그 결과를 실제 업무에 적용중이다[2, 4].

본 논문에서는 이동 단말기의 위치 추적, 경계 시스템과 같이 시간적으로 급속히 변화하는 데이터의 실시간 검색을 위하여 다중 사용자용 주기억장치 자료저장 시스템을 설계, 구현하고 구현된 시스템의 성능을 평가하였다. 구현된 다중 사용자용 자료저장 시스템은 주기억장치의 특성을 이용한 동시성 제어 방법을 채택하였으며 검색의 성능을 높이기 위하여 검색 쓰레드와 변경 쓰레드의 수를 조절하는 기능을 첨가하였다. 개발한 다중 사용자용 주기억장치 자료저장 시스템은 기존의 디스크 상주형 데이터베이스의 성능을 보완하는 형태로 구성되어, 다양한 상용 DBMS와 함께 사용될 수 있다.

본 논문의 구성은 다음과 같다.

2장에서는 관련 연구에 대해서 설명하고, 3장에서는 실시간 검색 시스템의 구조에 대해서 기술한다. 4장에서는 주기억장치 자료저장 시스템에 대해 기술하고, 5장에서는 구현된 시스템의 실험 평가를 하고, 마지막으로 6장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

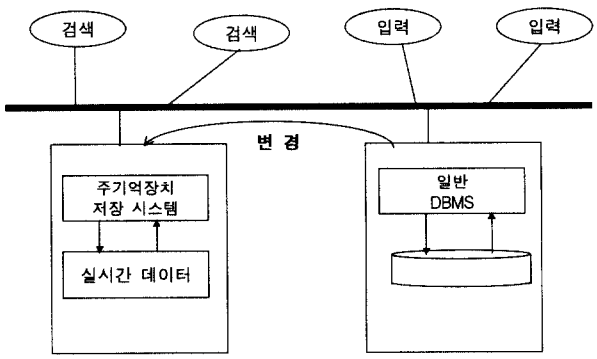
실시간 트랜잭션은 실시간 데이터베이스 시스템에서 가장 기본이 되는 일의 단위이다. 실시간 트랜잭션에는 비록 응용에 따라 차이는 있지만 시간제약, 중요도(criticalness), 가치함수(value function)와 같은 정보가 존재할 수 있고, 그 정보들은 실시간 트랜잭션의 스케줄링에 사용된다[14]. 실시간 트랜잭션의 시간 제약은 그 트랜잭션이 다루는 데이터의 유효성이 시간적으로 빨리 변하기 때문에 발생한다. 따라서 시간제약은 해당 트랜잭션이 수행되는 즉시 부여되며 수행 도중 마감 시간의 변경도 가능하다.

실시간 데이터베이스 시스템에서 스케줄링의 목적을 2가지 측면에서 보면, 하나는 시간적 제약을 만족하는 것이고, 다른 하나는 데이터의 일관성을 유지하는 것이다. 시간적 제약을 만족시키기 위해서 실시간 태스크 스케줄링 방법을 실시간 트랜잭션 스케줄링에 확장 적용하는 것이 가능하며, 데이터 일관성 유지를 위해서는 동시성제어 프로토콜이 필요하다. 일반적으로 실시간 데이터베이스에서 사용되는 동시성 제어 프로토콜은 기존에 있는 동시성 제어 프로토콜을 효율적으로 바꾸고, 트랜잭션 스케줄링으로는 좀 더 임박한 트랜잭션을 먼저 수행할 수 있도록 시간적 제약을 가지는 트랜잭션을 스케줄링하는 방법을 사용한다.

실시간 트랜잭션의 시간제약 조건을 만족시키기 위한 방법으로 기존 디스크 상주형 데이터베이스 시스템에 실시간 처리 특성을 추가하는 방법과 주기억장치에 전체 데이터베이스를 상주시키는 방법이 있지만, 전자의 방법은 디스크 입출력으로 인한 트랜잭션 처리 지연을 피할 수 없기 때문에 실시간 트랜잭션에는 후자의 방법이 더 가능성이 많다[8]. 주기억장치 상주형 데이터베이스 시스템은 디스크 상주형 데이터베이스 시스템에 비해 높은 성능과 빠른 응답 시간을 제공하고, 디스크 입출력이 적기 때문에 결과의 예측이 상당히 정확하다. 동시성 제어에 있어서도 오버헤드가 큰 레코드 단위의 잠금 방식보다 구현이 간단한 래치(latch)로도 높은 성능을 기대할 수 있다.

3. 실시간 검색 시스템의 구조

(그림 1)은 기존의 디스크 기반의 데이터베이스 시스템과 주기억장치 자료저장 시스템을 병렬로 결합한, 본 논문에서 제안한 실시간 검색 시스템의 실행 구조이다.



(그림 1) 실시간 검색 시스템의 실행 환경

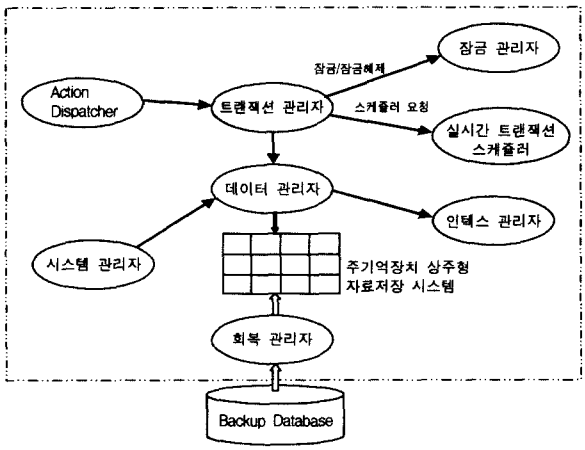
전체 실시간 검색 시스템에서 주기억장치 자료저장 시스템의 데이터 변경은 외부에서 직접적으로 실시간 데이터베이스로 접근되는 것이 아니다. 우선 디스크 상주형 데이터베

이스나 통신 모듈에 변경 요청이 있으면 디스크에 저장이 되고, 그 내용의 변화가 주기억장치 자료저장 시스템의 내용을 변화할 만큼의 시스템 관리자가 미리 정한 조건을 만족하면(예를 들어서 이동 통신 가입자의 현재 위치가 10M 이상 변화하면) 디스크 상주형 데이터베이스가 트리거와 저장 프로시저의 통신모듈을 이용하여 요청하거나 통신모듈이 판단하여 시스템에 요청하여 수행하게 된다. 본 논문에서는 디스크 상주형 데이터베이스로는 Oracle을 사용하였다.

4. 주기억장치 자료저장 시스템

4.1 주기억장치 자료저장 시스템의 구성요소

본 논문에서 사용하는 실시간 검색 시스템은 데드라인을 넘긴 트랜잭션도 끝까지 실행시키는 소프트 실시간 트랜잭션을 사용하며 (그림 2)와 같이 데이터관리자, 인덱스 관리자, 회복 관리자, 시스템 관리자, 트랜잭션 관리자, Action Dispatcher등으로 구성되어 있다. 트랜잭션 관리자는 스케줄러, 잠금 관리자를 포함하고, 다중 사용자를 고려하여 시스템의 성능을 향상시키기 위해 다중 쓰레드 구조를 가지며, 다중 쓰레드를 쉽게 구현할 수 있는 JAVA 언어로 구현하였다.



(그림 2) 주기억장치 자료 저장 시스템의 구성 요소

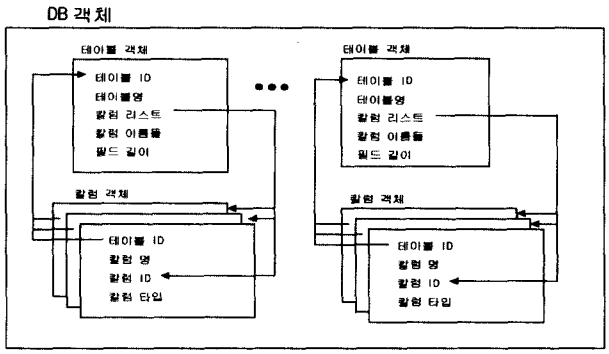
4.1.1 Action Dispatcher

Action Dispatcher는 클라이언트, 디스크 데이터베이스 시스템, 입력 통신용 모듈에서 들어온 요청들을 소켓을 사용하여 받아준다. 스트림을 사용하여 요청이 들어오면 Action Dispatcher에서 요청한 클라이언트의 주소, 스트림을 ID, 제한시간, SQL 문의 형태(검색, 입력)등의 형태로 분해하고, 이를 트랜잭션 관리자에 파라미터 형태로 넘겨준다.

4.1.2 트랜잭션 관리자(Transaction Manager)

트랜잭션 관리자는 Action Dispatcher로부터 넘어온 트랜잭션을 스케줄링한다. 넘어온 트랜잭션은 시스템 초기화 시점에 미리 생성해 놓은 쓰레드 풀(pool)에 적재된 후 제한시간(Deadline)에 의해 스케줄링된다. 쓰레드 풀에서는 시스템의 성능을 위해 검색과 입력의 요청 수를 제한하는 데 제한 개수 이상을 넘어가면 트랜잭션 대기 큐에 삽입해 재스케줄 되기를 기다린다. 검색과 변경의 쓰레드 수를 제한하는 이유는 실시간 데이터베이스를 제한시간 내에 유효한 정보를 검색할 수 있게 시스템을 구성하기 위함이다.

4.1.3 데이터 관리자(Data Manager)



(그림 3) 데이터 관리 구조

데이터 관리자는 스케줄링 되어 넘어온 트랜잭션의 SQL 문에 따라 할당되어진 메모리에 데이터를 갱신하거나 입력하거나 검색한다. 데이터의 관리 구조는 DB 객체, 테이블 객체, 칼럼객체를 가진다. DB 객체는 최상위 개념으로서 이름을 가지고 구별하며, 물리적 메모리를 할당받는다. 최하위구조는 칼럼이고, 칼럼들이 모여 테이블 객체를 구성하고, 데이터간의 관계를 나타내는 객체의 정보를 가진다. (그림 3)은 데이터 객체를 도식화한 것이다. 칼럼 객체는 테이블의 ID와 칼럼 이름, 칼럼 타입, 칼럼 ID 등의 정보를 가지고, 테이블 객체는 테이블 ID와 칼럼 객체들의 리스트와 칼럼 ID에 대한 정보를 유지하면서, 칼럼에 대한 정보를 참조한다. 테이블 객체는 또 테이블 타입의 정보도 가지고 있는데 이것은 인덱스 테이블과 일반 정보를 유지하는 테이블로 나누어진다. 이렇게 인덱스 테이블을 나누어 관리하는 것은 검색 시점에 속도를 빠르게 하기 위해서이다. 트랜잭션과 마찬가지로 칼럼과 테이블은 객체가 생성될 때 유일한 ID를 부여받아 중복된 객체가 생성되지 않도록 구별한다.

4.1.4 인덱스 관리자(Index Manager)

인덱스 관리자는 데이터 관리자의 요청에 의해 빠른 검색을 위한 인덱스를 생성, 관리한다. 본 논문에서는 해시와 Quadtree 인덱스를 동시에 지원하는데, 일반 데이터의 검색시에는 해시 인덱스를 사용하고, 범위 검색 또는 다차원 데이터 검색시 Quadtree를 사용하여 검색 속도를 높인다. 인덱스 관리자는 데이터 관리자의 요청에 따라 검색의 방법에 따라 사용하는 인덱스를 결정하게 된다.

4.1.5 시스템 관리자(System Manager)

시스템 관리자는 주기억장치 상주형 데이터베이스 시스템

의 전반적인 설정과 시스템의 상황을 모니터링 한다. 동시에 접속할 수 있는 클라이언트의 수, 최대 큐잉될 수 있는 검색과 입력의 수, DB 객체의 이름과 크기를 설정하여 데이터베이스 시스템을 시작하도록 하고, 메모리의 강제수거(garbage collection)를 한다. 사용되어질 최적의 메모리를 계산하여 할당하는 것도 중요하나 메모리는 데이터의 저장뿐만 아니라 운영체제와 데이터베이스 시스템에서도 사용하여야 하기 때문에 여유 분을 확보하는 것이 더 중요하기 때문에, 본 논문에서는 사용되고 난 메모리의 해제에 더 중점을 두고 시간 주기에 따라 실행되거나, 관리자가 메모리의 상태를 파악한 뒤 메모리의 강제 수거를 할 수 있도록 설계되었다.

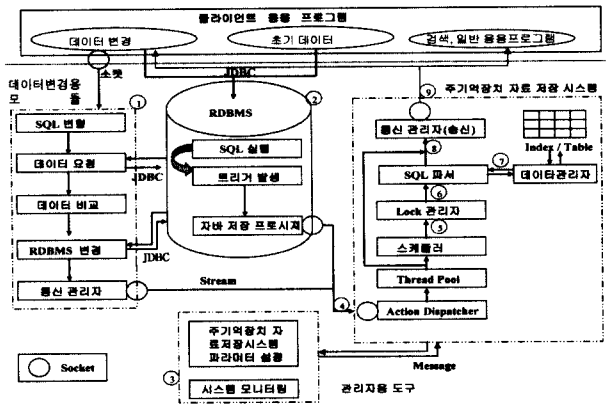
4.1.6 회복 관리자(Recovery Manager)

회복 관리자는 독립적으로 수행되며 시스템 회복과 초기화 작업을 담당한다. 일반적인 디스크 기반의 데이터베이스 시스템에서의 회복에 관련된 사항은 로그에 의해 이루어진다. 그러나 본 논문에서는 로그에 소요되는 디스크 입출력 시간을 없애기 위하여 회복을 위한 로깅을 사용하지 않고, 관리자의 요청에 의해 초기화 작업만으로 회복 관리를 한다. 초기화는 회복 관리자가 독립적으로 실행되어 히스토리가 저장되어 있는 디스크 상주형 데이터베이스 시스템으로부터 가장 최신의 정보를 가지고 오는 일괄작업으로 처리한다. 일괄작업으로 처리가 가능한 이유는 데이터와 테이블 구조 등 모든 스키마 정보가 디스크 상주형 데이터베이스에 저장되어 있고, 주기억장치 자료저장 시스템은 가장 최신의 자료들만을 필요로 하기 때문이다. 따라서 재실행(redo) 작업에 해당하는 것만을 수행함으로써 초기화 및 회복작업을 마치게 된다.

4.2 프로그램 흐름도

주기억장치 상주형 자료저장 시스템의 전체 처리 절차를 도식화하면 (그림 4)와 같다. ①과 ②는 앞에서 설명한 통신모듈을 표현한 것이다. 각각 소켓과 JDBC를 사용하여 접속하고, ①의 통신모듈에서 일반 데이터베이스 시스템으로 자료를 요청할 때도 JDBC를 사용하여 통신한다.

③은 시스템의 초기 시작시에 전체적인 설정을 하고, 시스템에서 발생하는 메시지를 감시한다. ④는 Action Dispatcher에서 스트림 형식으로 넘어온 요청 문을 적절한 형태로 분해하고, 적절한 요청문인지 검사한 후, 쓰레드 풀에 트랜잭션의 형태로 넘겨준다. 그리고, 요청이 들어온 클라이언트를 식별하여 결과를 반환할 주소도 함께 유지한다. ⑤는 쓰레드 풀에서 최대한 접속할 수 있는 수를 제한하고, 초과 시 클라이언트에게 재요청 하도록 한다. ⑥은 잠금 관리자에 의해 현재 스케줄링된 트랜잭션이 지금 실행될 수 있는지, 또는 다른 트랜잭션들을 정지시켜야 할지, 블록된 트랜잭션들의 잠금을 풀지 결정한다. 진행 가능하다면 SQL 파서를 호출하여 실행한다. ⑦은 SQL 파서에서 넘어온 트

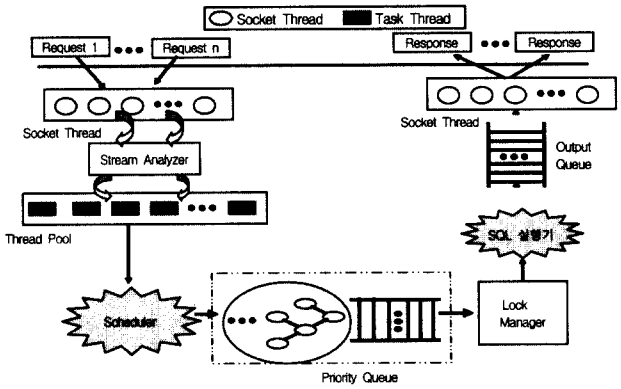


(그림 4) 프로그램 처리 절차

랜잭션이 정확한 SQL 문인지 검사하고, 어떠한 액션을 취하여야 하는지 결정한 후, 데이터 관리자에 요청하여 테이블과 인덱스의 변경이나 검색을 요청한 후 결과를 돌려 받는다. ⑧은 반환된 결과를 이용하여 결과 반환용 통신관리자에 보낸다. ⑨는 결과 반환용 통신관리자가 멀티 쓰레드 구조를 이용하여 트랜잭션에 보관된 클라이언트 주소로 해당 클라이언트에 결과를 반환한다.

4.3 서버 프로세스의 구현

4.3.1 서버 프로세스의 구조



(그림 5) 서버프로세스의 구조

서버 프로세스는 기본적으로 다중 쓰레드 구조를 갖는다. 스트림 분해에 있어서도 각각의 소켓이 이를 수행하며, 분해된 스트림은 트랜잭션이 된다.

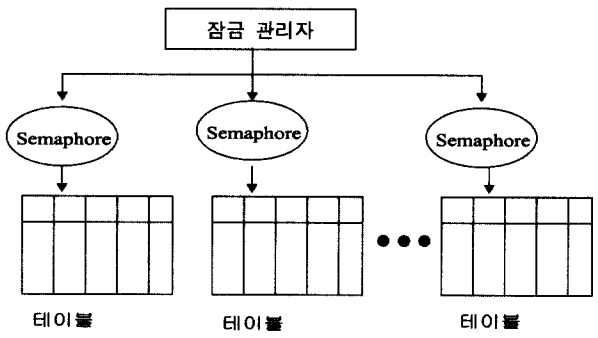
4.3.2 트랜잭션 스케줄러

본 논문에서 사용하는 실시간 트랜잭션은 발생 시간이 일정하지 않은 비주기성 트랜잭션이며 트랜잭션의 실행시간을 미리 예측하기 힘들며, 트랜잭션 발생시점에서는 어느 데이터를 요구할지 미리 알 수 없고, 잠금 충돌이나 트랜잭션 중단을 트랜잭션 발생시점에서는 알 수 없기 때문에, 동적 우선 순위 할당 방식인 ED(Earliest Deadline) 알고리즘[10]을 사용한다. 이 방식은 데드라인이 빠른 트랜잭션이 높은 우선

순위를 할당받는 방식이며 적당한 수의 트랜잭션이 적재된 경우에는 좋은 성능을 유지하지만 과도한 수의 트랜잭션이 적재된 경우에는 트랜잭션이 연속적으로 데드라인을 위반할 수 있는 결점이 있다. ED 알고리즘에서 발생하는 결점을 해결하기 위해서 과부하 분산(overload shedding)의 개념을 변형하여 사용한다.

4.3.3 동시성 제어

주기억장치 자료저장 시스템에서는 잠금의 대상이 되는 데이터가 모두 메모리 상에 존재하고, 실시간 시스템이므로 데드라인을 넘기는 트랜잭션의 수를 최소화하는 것이 동시성 제어의 목표가 되므로, 잠금의 방법도 디스크 기반 데이터베이스 시스템과는 다른 방식을 적용시킬 필요가 있다. 본 논문에서 제안한 주기억장치 자료저장 시스템의 경우에는 (그림 6)과 같이 테이블 단위의 세마포(또는 래치)를 사용하는 것만으로도 충분한 동시성 제어를 할 수 있다. 따라서, 별도의 잠금에 대한 정보를 유지하지 않아도 되므로 잠금 기능을 구현하기 위한 많은 부담을 줄일 수 있다.



(그림 6) 잠금의 설정

4.3.4 트리거(Trigger)

트리거란 데이터베이스가 미리 정해 놓은 조건을 만족하거나 또는 어떤 동작이 수행되면, 자동적으로 다른 동작을 수행하도록 하며, 관계형 데이터베이스내의 관계 유지와 예외 조건의 감지 및 데이터베이스 감시를 위한 흔적을 유지하는데 사용한다[17]. 본 논문에서는 데이터 값의 변경이 시스템 관리자가 미리 설정해 놓은 임계치(응용 도메인에 따라 다른 값을 설정) 이상이면 주기억장치로 변경 요청을 위하여 트리거를 구성하였다.

```
create or replace TRIGGER trig_ins
AFTER INSERT ON curr_pos FOR EACH ROW
      … (변수 선언)
BEGIN
  SELECT curr_x, curr_y, u_time INTO cur_x, cur_y,u_t
  FROM curr_pos
  WHERE curr_id = : new.curr_id AND u_check = 1 ;
  IF ( : new.curr_x - cur_x >= 10) OR
       ( : new.curr_y - cur_y >= 10) OR
       (cur_x - : new.curr_x >= 10) OR
       (cur_y - : new.curr_y >= 10) THEN
       UPDATE curr_pos
       SET u_check = 0
       WHERE curr_id = : new.curr_id AND u_time = u_t ;
       CALL proc_ins( : new.curr_id, : new.curr_x, : new.curr_y)
  END IF ;
END ;
```

본 논문에서 트리거는 데이터의 갱신시 새로운 데이터와 이전 데이터를 비교하여 주기억장치 자료저장 시스템으로 데이터를 전송할 만큼 미리 정한 임계값을 초과하는지의 여부를 판단하는 조건 검사를 하게 된다. 미리 정한 임계값 이상의 데이터의 변화가 있으면 자바 저장 프로시저를 호출하여 자바 저장 프로시저에서 소켓 통신을 하여 주기억장치 자료저장 시스템에 자료 갱신 요청을 하게 된다.

4.3.5 자바 저장 프로시저(Java Stored Procedure)

본 논문에서 사용한 Oracle 8i에서는 데이터베이스 커널 내부에 자바 가상 머신을 내장시킴으로서 자바언어로 작성한 저장 프로시저가 내부에서 실행 가능하다. 따라서 기존의 PL/SQL로 작성해 왔던 거의 모든 연산들을 자바로 작성하는 것이 가능하다. 자바 저장 프로시저는 publish라는 과정을 통해 PL/SQL 인터페이스로 wrap된 상태에서 사용되므로 PL/SQL 저장 프로시저와 동일하게 사용할 수 있다. 여기서 publish는 자바 메쏘드에 대한 SQL 인터페이스를 제공함을 의미한다. 실제로는 자바 메쏘드에 대한 일종의 프록시 역할을 하는 PL/SQL 프로그램을 작성하게 되며, SQL은 이 Wrapper를 호출하여 자바 메쏘드를 호출하게 된다. 일련의 과정으로 생성된 자바 저장 프로시저는 다음과 같이 트리거 혹은 자바 등의 데이터베이스 응용 프로그램에서 불러 사용할 수 있다.

CALL proc_ins(: new.curr_id, : new.curr_x, : new.curr_y)

```
public class CommDB
{
      public static void trigger_insert ( String curr_id, int curr_x,
                                          int curr_y)
      throws SQLException
      {
            sql = "insert into curr_pos values(" ;
            sql = sql + curr_id + "," + curr_x + "," + curr_y
                  + ") ; " ;
            …
            소켓으로 sql을 전송하는 부분
            …
      }
}
```

4.3.6 SQL의 재정의 및 스트림 형식 정의

주기억장치 자료저장 시스템은 일반적인 목적보다는 특별한 용도로 사용되어지도록 설계 되어있다. 따라서 목적에 맞게 최소화된 SQL과 통신용 스트림의 형식이 필요하다. 다음은 실험에 사용된 실시간 위치 관리 응용의 스트림의 형식으로 요청 시에는 클라이언트의 ID와 제한시간, 그리고 요청을 위한 SQL 문이 전송되고, 응답시에는 요청한 ID와

좌표 값이 하나의 문장으로 이루어져 반환된다.

① 요청 : ID | 시간제한 | SQL문 |
② 응답 : ID : X : Y | ID : X : Y | …

다음은 재정의된 SQL 문으로 주기억장치 상주형 자료저장 시스템은 제한된 용도이므로 고정된 형태로 일정 기능의 함수역할만 하고 있다.

① 테이블의 생성
create table {table name} ({column}, {column}, {column})
② ID에 의한 일반 검색
select×from {table name} where id = {ID}
③ 위치에 의한 범위 검색
select R {table name} values ({point_x}, {point_y}, {width}, {height})
④ 최근접 질의
select S {table name} values ({point_x}, {point_y})
⑤ 새로운 값 삽입
insert into {table name} values ({value}, {value}, {value})
⑥ ID에 의한 테이블의 값 변경
update {table name} set {column} = {value}, {column} = {value} where id = {ID}
⑦ ID에 의한 테이블의 값 삭제
delete from {table name} where id = {ID}

5. 실험 평가

5.1 실험 환경 및 방법

실험을 위한 서버의 환경은 아래와 같다.

- 데이터베이스 서버
  - 서버 : SUN Enterprise-250, RAM 512M
  - 운영체제 : Solaris 2.7
  - 데이터베이스 시스템 : Oracle 8.1.5
- 주기억장치 상주형 자료저장 시스템
  - 서버 : Pentium III 700MHz, RAM 1024M
  - 운영체제 : Windows 2000 professional

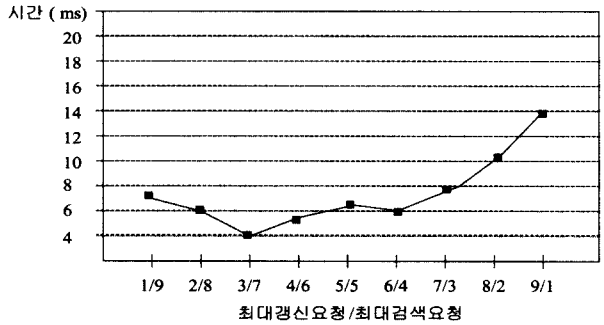
클라이언트는 Windows 98을 사용하였으며 데이터베이스 서버와 주기억장치 상주형 자료저장 시스템은 각기 다른 컴퓨터를 사용하였다. 실험모델은 네비게이터나 이동통신기기의 위치 추적을 고려하였기 때문에, 빈번한 위치의 갱신 요청이 있는 상태를 고려하여 검색과 갱신 요청의 비율을 4 : 1로 설정하고, 좌표의 크기는 256×256으로 설정하여, 검색이나 갱신시의 요청되어지는 좌표는 1에서 256사이의 임의의 값으로 하였다. 즉, 실험 데이터는 일정한 값을 가지는 것이 아니라 매 실험 시마다 임의로 발생되어지는 좌표를 사용한다. 실제 이동통신기기의 위치파악을 위한 시스템은 중복된 값이 거의 발생하지 않고 많은 개체(ID)를 가지나 실험에서는 적은 좌표의 크기를 가지고, 임의로 좌표와 개체(ID)가 생성되므로 중복된 값이 발생하게 되어, 예상보다 적은 개수의 튜플이 입력되게 되는 현상이 나타났다. 또한 범위 검색을 위한 Quadtree는 중복된 값은 입력되지 않으므로 이보다 적은 개수의 좌표가 입력된다. 검색은 모두 범위 검색 연산으로서 범위의 크기는 30×30으로 설정하고, 빈번한 요청이 발생하도록 하여 시스템에 비교적 높은 부하가 발생되도록 하였다.

〈표 1〉 실험 데이터

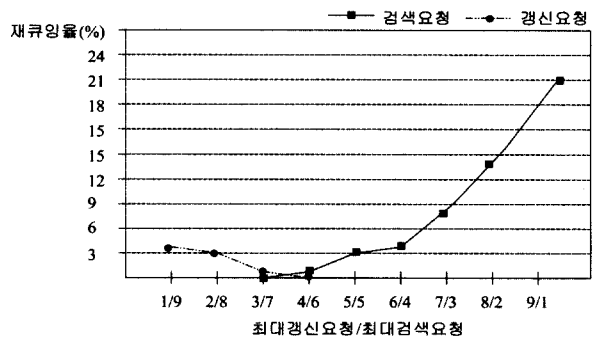
실험 데이터 종류	실험 데이터 값
튜플 개수	100~1000
좌표의 크기	256×256
검색 연산의 범위	30×30
테이블의 필드 수	2개

5.2 성능 분석

주기억장치 자료저장 시스템의 초기화 시점에 설정하는 갱신요청과 검색요청의 수를 조절하는데 따른 실시간 검색의 응답 시간 결과를 (그림 7)에 표시하였다. (그림 7)의 앞부분의 응답 시간이 더 나쁜 것은 검색 요청은 많지만 검색 쓰레드의 개수가 충분하지 않아 재큐잉이 많이 생겨서 생긴 현상임을 (그림 8)을 통해서 알 수 있다.

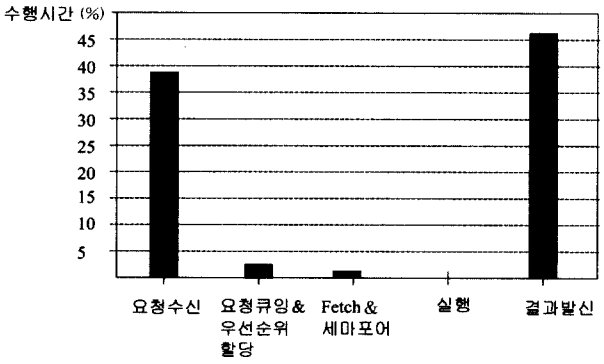


(그림 7) 갱신/검색 요청 비율에 따른 응답 시간



(그림 8) 갱신/검색 요청에 따른 재큐잉률

(그림 9) 클라이언트에서 요청을 하여 그 결과가 다시 클라이언트로 돌아오는 시간에 대한, 각 처리모듈별 수행시간의 비율을 나타낸 것이다. 하나의 트랜잭션의 결과가 돌아오는데 평균적으로 900ms정도가 걸리는데, 대부분의 수행시간이 통신모듈에서 걸리는 것을 알 수 있다.



(그림 9) 모듈별 수행시간의 비율

6. 결 론

본 논문에서는 실시간 검색을 위해서 기존의 디스크 기반의 데이터베이스 시스템을 그대로 사용하면서 주 메모리 데이터베이스 시스템에서 필요한 부분만을 사용하는 병렬 적인 구성을 가진 주기억장치 상주형 자료저장 시스템을 구현하였다. 본 논문에서 구현된 주기억장치 자료저장 시스템은 래치를 이용하여 데이터의 동시성 제어를 간단히 하고, 로그를 이용하는 롤백 방식의 회복 기능 대신에 롤포워드 기능을 사용하여 로그로 인한 비용을 줄여 빠른 응답 시간을 제공한다. 또한 안전한 데이터의 보관과 일반 트랜잭션을 위해서는 디스크 기반의 기존의 데이터베이스 시스템을 그대로 사용할 수 있게 하였다. 이로 인해서 기존의 디스크 기반의 시스템에 주기억장치 자료저장 시스템을 추가함으로써 실시간 검색을 위해 빠른 응답을 할 수 있는 실시간 자료 검색 시스템을 구축할 수 있다. 그러나 본 논문에서 구현된 자료저장 시스템이 더 나은 성능을 보장하기 위해서는 여전히 보완되어져야 할 부분이 있다. 첫째, 클라이언트와의 통신에 필요한 오버헤드의 해결이다. 실험의 결과에서 보았듯이 트랜잭션의 대부분의 시간은 서버와 클라이언트가 통신을 하는데 걸린다. 현재는 일반 TCP/IP 프로토콜을 사용하는데 경량화가 이루어져서 짧은 시간에 연결이 이루어질 수 있는 통신방법에 대한 연구가 필요하다.

둘째, 다양한 운영체제 환경과 시스템 구성 방법에 의해서 장시간의 시스템의 수행과 보완이 이루어져야 한다. 시스템에 독립적인 자바로 구현되어지기는 하였으나, 운영체제별 실행 특성이 조금씩은 틀리고, 시스템 구성 방법에 따른 결과도 달라질 수 있기 때문이다.

참 고 문 헌

[1] "데이터베이스 서비스 시스템 개발", 한국전자통신연구소, 1996.

[2] "주기억장치 상주형 실시간 DBMS 개발에 관한 연구", 한국전자통신연구소, 1996.

[3] R. Abbott, H. Garcia-Molina, "Scheduling Real-Time Transactions : A performance Evaluation," Proceedings of the 14th VLDB Conference, pp.1-12, 1998.

[4] B. Adelberg, H. Garcia, B. Kao, "Emulating Soft Real-Time Scheduling Using Traditional Operating System Schedulers," extended version of RTSS, 1994.

[5] D. Agrawal, A. El Abbadi, R. Jeffers, "Using delayed commitment in locking protocols for real-time databases," Proceedings of the ACM SIGMOD International Conference of Management of Data, pp.104-113, 1992.

[6] K. Arnold, J. Gosling, *The JAVA™ programming language*, Addison Wesley, 1996.

[7] M. J. Carey, R. Jauhari, M. Livny, "Priority in DBMS Resource scheduling," Proceedings of the 15th VLDB conference, pp.397-410, 1989.

[8] H. Garcia-Molina, K. Salem, "Main Memory Database Systems : An Overview," IEEE Transactions on Knowledge and Data Engineering, Vol.4, No.6, pp.509-516, Dec., 1992.

[9] J. Gray, A. Reuter, "Transaction Processing : Concepts and Techniques," Morgan Kaufmann, 1993.

[10] J. Haritsa, M. Livny, M. Carey, "Earliest Deadline Scheduling for real-time Database Systems," IEEE Real-Time Systems Symposium, pp.232-142, 1991.

[11] J. Haritsa, M. Carey, M. Livny, "On Being Optimistic About Real-Time Constraints," Principles of Database Systems, pp.331-343, Apr., 1990.

[12] J. Harista, M. Carey, M. Livny, "Dynamic Real-Time Optimistic concurrency Control," In Proceedings of the 11th Real-Time Systems Symposium, Orlando, FL, pp.94-103, Dec., 1990.

[13] Steven G. Harris, *Oracle 8i Java Developer's Guide, Release 8.1.5*, Oracle Corporation, 1999.

[14] B. Kao, H. Garcia-Molina, "An Overview of Real-Time Database Systems," in Advances in Real-Time Systems, Prentice Hall, pp.463-486, 1995.

[15] T. Kevin, *Building Intelligent Databases with Oracle PL/SQL, Trigger, and Stored Procedures*, Prentice Hall, 1996.

[16] G. Ozsoyoglu, R. Snodgrass, "Temporal and Real-Time Database : A Survey," IEEE Transactions on Knowledge and Data Engineering, 7(4), pp.513-532, Aug., 1995.

[17] Tom Portfolio, *Oracle 8i Java Stored Procedures Developer's Guide, Release 8.1.5*, Oracle Corporation, 1999.

[18] P. Yu, K. Wu, K. Lin and S. H. Son, "On Real-Time Databases : Concurrency Control and Scheduling," Proceedings of IEEE, Special Issue On Real-Time System, 82(1), pp.140-157, Jan., 1994.

권 오 수

e-mail : logos@kmvcc.ac.kr

1999년 계명대학교 컴퓨터공학과 졸업 학사

2001년 계명대학교 컴퓨터공학 석사 졸업

2001년 (주)ITM

2002년~현재 (주)이니텍

관심분야 : 데이터베이스, 실시간 데이터베이스

홍 동 권

e-mail : dkhong@kmucc.keimyung.ac.kr

1985년 경북대학교 전자공학과 졸업(학사)

1992년 University of Florida 전자계산학과 졸업(석사)

1995년 University of Florida 전자계산학과 졸업(박사)

1985년~1990년 한국전자통신연구원

1996년~1997년 한국전자통신연구원

1997년~현재 계명대학교 공학부 컴퓨터공학 전공 조교수

관심분야 : 능동 실시간 데이터베이스, 병렬 처리, 성능 평가, 시뮬레이션, 멀티미디어 처리