공간 메인 메모리 DBMS를 위한 효율적인 회복 시스템†

An Efficient Recovery System for Spatial Main Memory DBMS

김정준* / Joung-Joon Kim **주성완**** / Sung-Wan Ju **강홍구***** / Hong-Koo Kang
홍동숙**** / Dong-Sook Hong **한기준******* / Ki-Joon Han

요 약

최근 실시간 서비스의 요구 사항을 갖는 위치 기반 서비스(Location Based Service : LBS)와 텔레매틱스(Telematics) 서비스를 효율적으로 제공하기 위해서 공간 메인 메모리 DBMS에 대한 관심이 급증하고 있다. 이러한 공간 메인 메모리 DBMS에서는 시스템 장애가 발생하였을 경우 메인 메모리상의 모든 공간 데이타를 잃어버릴 수 있으므로 데이타베이스의 안정성을 위한 회복 시스템은 매우 중요하다. 회복 시스템에서 로그와 체크포인트 수행 과정 중 발생하는 디스크 입출력은 전체 시스템 성능을 저하하는 중요한 요인이 되고 있다. 그러므로, 공간 메인 메모리 DBMS에서 디스크 입출력을 줄일 수 있는 효율적인 회복 시스템에 대한 연구가 절실히 필요하다. 본 논문에서는 공간 메인 메모리 DBMS를 위한 효율적인 회복 시스템에 대해서 연구하였다. 먼저 로그 기법으로는 디스크 입출력을 줄이고 트랜잭션의 동시성 향상을 위해 사전 완료 기법을 사용하였고, 전체 시스템 성능을 향상시키기 위해 기존의 퍼지-핑퐁 체크포인트 기법에서 발생하는 동일 페이지에 대한 중복 디스크 입출력 문제를 해결한 퍼지-쉐도우 체크포인트 기법을 제안하여 회복 시스템 구현 시 사용하였다. 그리고 마지막으로 본 논문에서 개발한 회복 시스템의 성능 평가를 수행하여 효율성을 입증하였다.

Abstract

Recently, to efficiently support the real-time requirements of LBS and Telematics services, interest in the spatial main memory DBMS is rising. In the spatial main memory DBMS, because all spatial data can be lost when the system failure happens, the recovery system is very important for the stability of the database. Especially, disk I/O in executing the log and the checkpoint becomes the bottleneck of letting down the total system performance. Therefore, it is urgently necessary to research about the recovery system to reduce disk I/O in the spatial main memory DBMS. In this paper, we study an efficient recovery system for the spatial main memory DBMS. First, the pre-commit log method is used for the decrement of disk I/O and the improvement of transaction concurrency. In addition, we propose the fuzzy-shadow checkpoint method for the recovery system of the spatial main memory DBMS.

† 본 연구는 서울시 산학연 협력사업의 지원으로 수행되었음.
■ 논문접수 : 2006.9.26 ■ 심사완료 : 2006.12.15
* 교신저자 건국대학교 대학원 컴퓨터공학과 박사과정 (jjkim9@db.konkuk.ac.kr)
** 한국게임산업개발원 산업진흥팀 연구원 (swju@db.konkuk.ac.kr)
*** 건국대학교 대학원 컴퓨터공학과 박사과정 (hkkang@db.konkuk.ac.kr)
**** 건국대학교 대학원 컴퓨터공학과 박사과정 (dshong@db.konkuk.ac.kr)
***** 건국대학교 컴퓨터공학부 교수 (kjhan@db.konkuk.ac.kr)

This method can solve the problem of duplicated disk I/O on the same page of the existing fuzzy-pingpong checkpoint method for the improvement of the whole system performance. Finally, we also report the experimental results confirming the benefit of the proposed recovery system.

주요어 : 공간 메인 메모리 DBMS, 회복 시스템, 사전 완료, 퍼지-쉐도우 체크포인트

Keyword : Spatial Main Memory DBMS, Recovery System, Pre-Commit, Fuzzy-Shadow Checkpoint

1. 서 론

최근 지리 정보 시스템(Geographical Information System : GIS) 기술이 발전함에 따라 위치 기반 서비스, 텔레매틱스, 지능형 교통 시스템(Intelligent Transport Systems : ITS) 등과 같은 GIS 응용 분야에서 복잡한 공간 데이타의 빠르고 효율적인 처리가 절실히 요구되고 있다. 이를 위해 전체 데이타베이스를 메인 메모리에 상주시켜 처리하는 공간 메인 메모리 DBMS가 연구 개발되고 있다[1,2]. 이러한 공간 메인 메모리 DBMS에서는 전체 데이타베이스를 휘발성 기억 공간인 메인 메모리에 저장하기 때문에 컴퓨터 고장, 트랜잭션 에러, 시스템 에러 등으로 인한 예기치 않은 고장으로 인해 전체 데이타베이스를 잃어버릴 수 있다. 그러므로 회복을 위해 데이타베이스의 백업 화일과 트랜잭션의 처리에 대한 로그 정보를 디스크에 유지하는 것이 필요하다[3,4,5].

일반적으로 DBMS의 회복 시스템은 회복을 위하여 일반적으로 트랜잭션 처리에 대한 로그 정보를 디스크상의 로그 화일에 기록하고, 주기적으로 체크포인트를 수행하여 메인 메모리상에 있는 데이타베이스의 상태를 디스크상의 백업 화일에 반영하게 된다. 그리고 이러한 로그 화일과 데이타베이스 화일을 이용하여 회복 기능을 수행하게 된다. 그러나 회복을 위한 로그 기록과 체크포인트 수행 과정 중 발생하는 디스크 입출력은 시스템 성능을 저하시키는 중요한 요인이 된다[6]. 따라서 시스템 성능을 최대화하기 위해서는 디스크 입출력을 줄일 수 있는 효율적인 회복 시스템이 필요하다.

현재 공간 메인 메모리 DBMS의 회복에서도 트랜잭션 처리에 대한 로그를 디스크에 기록하기 위해 디스크 입출력 횟수가 많이 발생하여 메모리 연산에 비하여 큰 오버헤드를 가져와 전체 시스템 성능을 크게 저하시키는 문제가 되고 있다. 또한, 현재 공간 메인 메모리 DBMS는 체크포인트 수행 시 트랜잭션에 의해 갱신된 공간 데이타의 복사본을 디스크에 이중으로 저장하는 방식이기 때문에 디스크 연산이 많이 발생하고 디스크 공간 효율성도 떨어진다. 따라서, 공간 메인 메모리 DBMS에서 디스크 입출력에 대한 오버헤드를 최소화하면서 트랜잭션 로그를 관리하고, 트랜잭션에 의해 변경된 공간 데이타에 대한 효율적인 체크포인트를 수행하여 공간 메인 메모리 DBMS의 일관성을 유지할 수 있는 회복 시스템이 매우 중요하다[7,8,9].

이에 본 논문에서는 공간 메인 메모리 DBMS에서 효율적인 회복 시스템을 위해 로그와 체크포인트 기법에 대한 연구를 수행하였다. 우선 기존의 로그 기법들을 분석하여 디스크 입출력 감소와 트랜잭션 동시성 향상을 위한 사전 완료 기법을 설계 및 구현하였다. 그리고 기존 대다수의 메인 메모리 DBMS에서 사용하고 있는 핑퐁 갱신(Ping-Pong Update) 기반 퍼지 체크포인트(Fuzzy Checkpoint) 기법이 가지는 중복된 데이타에 대한 체크포인트 수행과 공간 낭비의 문제점을 개선한 퍼지-쉐도우(Fuzzy-Shadow) 체크포인트 기법을 제안하고 공간 메인 메모리 DBMS에서 구현하였다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로 공간 메인 메모리 DBMS에서의 회복을 위한 로그 기법, 체크포인트 기법들에 대해 분석한다. 제 3 장에서는 공간 메인 메모리 DBMS를 위한 회복 시스템의 개발에 대해서 설명하고, 제 4 장에서는 실험을 통한 공간 메인 메모리 DBMS를 위한 회복 시스템의 성능 평가 결과를 알아본다. 마지막으로, 제 5 장에서는 결론 및 향후 연구에 대해 언급한다.

2. 관련 연구

본 장에서는 공간 메인 메모리 DBMS에서의 회복을 위한 기존의 로그 기법과 체크포인트 기법들에 대해 살펴본다.

2.1 로그 기법

모든 회복의 가장 기본적인 전략은 데이타를 중복 저장하여 관리함으로서 시스템 오류에 대비하는 것이다. 회복을 위한 데이타를 유지하기 위한 가장 대표적이고 보편적인 기법으로 로그를 기반으로 하는 기법이 있다[10]. 이 기법은 회복을 위해 트랜잭션 처리 과정에서 일어나는 모든 변경 사항에 대한 정보를 가지고 있는 로그를 안전한 저장 장치에 기록하였다가 시스템 오류 발생 시 기록된 로그를 이용하여 회복시키는 기법이다.

이렇게 기록된 로그는 트랜잭션의 각 변경 연산에 대하여 변경 전과 변경 후의 값을 저장하고 있으며 시스템 오류 시 회복을 수행하기 위한 중요한 자원이 된다. 변경 전 값은 연산을 철회시키기 위해 사용되며 UNDO 로그로 기록된다. 변경 후의 값은 시스템 오류로 인해 연산의 결과가 반영되지 않았을 경우 이를 다시 반영시키는 역할을 하며 REDO 로그로 기록된다.

기존의 메인 메모리 DBMS와 공간 메인 메모리 DBMS에서 대표적인 로그 기법으로는 즉시 완료(Immediate Commit), 그룹 완료(Group Commit), 사전 완료(Pre-Commit) 기법이 있다.

2.1.1 즉시 완료 기법

즉시 완료 기법은 트랜잭션 수행 중 발생하는 변경 사항을 메인 메모리 로그 버퍼에 저장하고, 트랜잭션이 부분 완료(Partially Committed) 상태에서 완료(Committed) 상태로 들어가기 전에 로그 버퍼에서 해당 트랜잭션의 모든 로그를 디스크상의 로그 화일에 기록한 후 트랜잭션을 완료시킨다[11]. 이 기법은 개별 트랜잭션의 응답시간이 짧은 장점이 있으나 트랜잭션의 완료 시 마다 해당 트랜잭션의 모든 로그를 로그 화일에 기록하기 때문에 빈번한 디스크 입출력이 발생하는 단점이 있다.

2.1.2 그룹 완료 기법

그룹 완료 기법은 각 트랜잭션별로 로그 정보를 로그 화일에 기록하지 않고, 로그 버퍼에 여러 트랜잭션들의 로그 정보를 모아두었다가 부분 완료 상태에 있는 모든 트랜잭션들의 로그를 일괄적으로 로그 화일에 기록한 후 해당 트랜잭션들을 완료시킴으로써 디스크 입출력 횟수를 줄일 수 있는 기법이다[12,13]. 이 기법은 로그 버퍼의 내용이 디스크에 반영되는 횟수가 감소하므로 즉시 완료 기법보다 성능이 향상되는 장점이 있으나 부분 완료 상태의 트랜잭션들이 완료될 때까지 잠금(Lock)을 유지하게 되어 동시성이 저하되는 단점이 있다.

2.1.3 사전 완료 기법

사전 완료 기법은 트랜잭션이 완료된 후가 아닌, 트랜잭션이 부분 완료 상태가 되었을 때 바로 해당 트랜잭션이 보유한 자원에 대한 잠금을 모두 해제한다. 즉, 로그 버퍼의 내용이 디스크에 출력되는 것을 기다리지 않고 미리

잠금을 해제하여 후행 트랜잭션에서 해당 자원에 대한 접근을 허용한다. 그리고 로그 버퍼에 있는 부분 완료 상태의 모든 트랜잭션들의 로그를 일괄적으로 트랜잭션간의 수행 순서대로 로그 화일에 기록한 후 해당 트랜잭션들을 완료시킨다[14]. 이 기법은 그룹 완료 기법과 비교하여 개별 트랜잭션들의 응답 시간과 로그 버퍼의 디스크 출력 횟수에는 변화가 없지만, 해당 트랜잭션들이 보유한 잠금을 기다리는 다른 트랜잭션들의 잠금 대기 시간을 줄여 트랜잭션 수행의 동시성을 향상시킴으로써 그룹 완료 기법보다 성능이 향상되는 장점이 있다[15].

2.2 체크포인트 기법

일반적으로 메인 메모리 DBMS의 수행 시간이 증가함에 따라 상대적으로 로그 레코드의 양이 증가하게 되어 시스템 재시작과 회복 시 많은 시간이 소요되게 된다. 이를 해결하기 위해 시스템이 정상적으로 동작하는 과정에서 일정 시간 간격으로 최근의 메인 메모리 데이타베이스에서 갱신된 데이타가 있는 페이지들을 안전한 저장 장치에 반영해 주는데 이러한 일련의 과정을 체크포인트라 한다.

기존의 메인 메모리 DBMS와 공간 메인 메모리 DBMS에서 대표적인 체크포인트 기법으로는 트랜잭션 일관(Transaction Consistent) 체크포인트, 행위 일관(Action Consistent) 체크포인트, 그리고 퍼지 체크포인트 기법이 있다.

2.2.1 트랜잭션 일관 체크포인트 기법

트랜잭션 일관 체크포인트 기법은 수행 중인 트랜잭션이 모두 완료된 시점에 체크포인트를 수행하는 기법이다[16]. 이 기법은 체크포인트가 발생하면 우선 모든 선행 트랜잭션을 완료시킨다. 선행 트랜잭션이 모두 완료되면 실제 체크포인트 트랜잭션을 수행하고, 체크포인트 수행이 모두 완료되면 후행 트랜잭션의 생성과 수행을 허용한다. 이 기법은 체크포인트가 수행된 데이타베이스에는 트랜잭션 일관성이 유지되지만 체크포인트 수행 중에는 후행 트랜잭션의 생성과 수행이 중단되므로 성능이 저하되는 단점이 있다.

2.2.2 행위 일관 체크포인트 기법

행위 일관 체크포인트 기법은 트랜잭션을 데이타베이스에 대한 행위의 연속으로 보고 행위가 완료된 상태에서 체크포인트를 수행한다. 이 기법에서는 체크포인트를 수행하는 동안 데이타베이스에 대한 모든 갱신 연산을 중지해야 하고, UNDO 로그와 REDO 로그를 모두 기록해야 한다. 이 기법은 트랜잭션 일관 체크포인트 기법과 같이 트랜잭션 수행과 동기를 맞추는 방식으로 체크포인트 수행 중에는 후행 트랜잭션의 생성과 수행이 중단되므로 성능이 저하되는 단점이 있다[17].

2.2.3 퍼지 체크포인트 기법

퍼지 체크포인트 기법은 다른 트랜잭션과 비동기식으로 체크포인트를 수행하는 방식으로 트랜잭션 수행과 무관하게 진행하므로 앞에서 살펴본 체크포인트 기법들 보다 우수한 성능을 가진다[18]. 그러나 퍼지 체크포인트 기법은 동일한 메인 메모리 데이타베이스의 페이지에 대해 갱신연산 트랜잭션과 체크포인트가 동시에 수행될 수 있으므로 체크포인트 수행 중에는 데이타베이스 일관성 유지가 어렵다. 따라서 체크포인트 수행 중에 시스템 오류가 발생하면 디스크상의 데이타베이스 화일을 회복에 사용할 수 없게 된다.

이러한 문제점을 해결하기 위해 많은 DBMS에서는 디스크에 두 개의 데이타베이스 화일을 가지고 교대로 체크포인트를 수행하는 핑퐁 갱신(Ping-Pong Update)을 사용한다[19,20,21]. 핑퐁 갱신을 기반으로 하는 퍼지 체크포인트

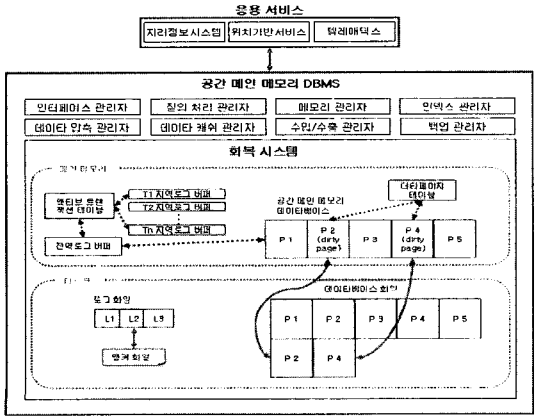
기법에서는 메인 메모리 데이타베이스내에서 갱신된 페이지는 두 개의 데이타베이스 화일에 모두 기록되어야 한다. 이는 DBMS에서 동일한 페이지에 대해 두 번의 디스크 입출력이 발생하므로 회복 시스템 성능이 저하되는 문제점이 있으며, 또한 두 개의 데이타베이스 화일을 유지하기 위한 추가 공간이 필요하고 이를 관리하기 위한 유지비용도 발생하게 된다.

3. 공간 메인 메모리 DBMS를 위한 회복 시스템의 개발

본 장에서는 공간 메인 메모리 DBMS를 위한 회복 시스템의 구조와 특징에 대해서 언급한 후 본 논문에서 제안하는 퍼지-쉐도우 체크포인트 기법을 설명한다. 그리고 회복 시스템에서의 로그 관리와 회복 관리에 대해 설명한다.

3.1 회복 시스템의 전체 구조

본 논문에서 설계한 공간 메인 메모리 DBMS를 위한 회복 시스템의 전체 구조는 그림 1에 나타나 있다. 현재 공간 메인 메모리 DBMS는 인터페이스 관리자, 질의 처리 관리자, 메모리 관리자, 인덱스 관리자, 데이타 압축 관리자, 데이타 캐쉬 관리자, 수출/수입 관리자, 백업 관리자, 회복 시스템으로 구성되어 있다.



<그림 1> 회복 시스템의 전체 구조

회복 시스템의 전체 구조에서 공간 메인 메모리 데이타베이스는 메인 메모리상의 공간 데이타 저장소이고, 액티브 트랜잭션 테이블(Active Transaction Table)은 현재 수행중인 트랜잭션 정보를 저장하는 테이블로서 구조는 표 1과 같다. 트랜잭션들은 하나 이상의 연산이 모여 이루어지는데 이러한 트랜잭션들의 원활한 수행을 위해서는 관련 정보들이 필요하다. 우선 트랜잭션을 유일하게 구분할 수 있는 식별자 정보가 필요하고, 현재 트랜잭션 상태 정보를 제공해야 하며, 동시성 제어를 위해 현재 트랜잭션이 점유 또는 요청하는 자원에 대한 리스트, 그리고 연관된 트랜잭션들에 대한 정보를 관리해야 한다. 그리고 이러한 정보는 액티브 트랜잭션 테이블을 이용하여 저장 관리되며, 다른 관리자들에게 이러한 정보들이 제공된다.

<표 1> 액티브 트랜잭션 테이블 구조

항 목	설 명
트랜잭션 아이디	트랜잭션의 식별자
트랜잭션 상태	트랜잭션의 상태(시작/수행중/부분완료)
연관된 트랜잭션 아이디	트랜잭션 부분 완료 시 잠금을 해제한 자원에 대해 연산을 수행한 트랜잭션에 대한 식별자
지역 로그 포인터	트랜잭션의 해당 지역 로그 버퍼에 대한 링크

더티 페이지 테이블(Dirty Page Table)은 최근 체크포인트 이후 변경된 페이지 정보를 저장하는 테이블이다. 트랜잭션이 완료될 때 해당 트랜잭션에 의해 변경된 페이지들을 더티 페이지라 하는데, 더티 페이지 테이블은 이러한 더티 페이지에 대한 식별자 값인 페이지 아이디(Page ID)를 저장한다.

지역 로그 버퍼(Local Log Buffer)는 각 트랜잭션의 로깅 경합에 따른 성능 저하를 방지하기 위하여 각 트랜잭션 별로 로그를 저장하는 버퍼로서 구조는 표 2와 같고 트랜잭션의 모든 연산이 수행 된 후(부분 완료 시) REDO

로그만 전역 로그 버퍼로 이동된다. 그리고 해당 트랜잭션에 의해 기록될 로그의 양을 예측할 수 없기 때문에 지역 로그 버퍼는 이중 연결 리스트의 구조로 되어있다.

<표 2> 지역 로그 버퍼 구조

항 목	설 명
로그 종류	트랜잭션 상태(시작/부분완료/완료) 연산 종류(입력/삭제/갱신)
페이지 아이디	연산이 수행된 페이지의 식별자
페이지 내 위치	연산이 수행된 페이지 내에서 위치
공간 데이타 종류	변경할 공간 데이타의 종류
변경 전 공간 데이타	연산이 수행되기 전의 공간 데이타
변경 후 공간 데이타	연산이 수행된 후의 공간 데이타

전역 로그 버퍼(Global Log Buffer)는 부분완료 상태의 트랜잭션들에 대한 REDO 로그를 저장하는 버퍼로서 트랜잭션 완료 시 버퍼에 있는 로그들을 로그 화일로 이동시킨다. 전역 로그 버퍼는 REDO 로그만을 가지고 있기 때문에 지역 로그 버퍼 구조에서 변경 후 공간 데이타 항목을 제외한 것과 같은 구조를 가진다.

로그 화일은 완료된 트랜잭션들의 REDO 로그 정보들을 저장하는 화일로서 구조는 표 3과 같고 로그의 구성요소는 전역 로그 버퍼와 동일하다. 로그 화일은 로그 화일에 저장된 로그 개수 정보를 위한 Head Block과 완료된 트랜잭션들의 로그 정보를 위한 Table Info Block으로 구성된다.

<표 3> 로그 화일 구조

Head Block	
로그 개수	로그 화일에 저장된 로그 개수
Table Info Block	
로그	완료된 트랜잭션들의 REDO 로그

트랜잭션의 완료 시 전역 로그 버퍼에서 해당 트랜잭션의 로그를 디스크상의 로그 화일에 기록한다. 로그 화일은 두 개 이상이 순차적으로 생성되어 일정 개수의 로그 화일을 유지하며, 하나의 로그 화일에 기록되는 로그의 양이 로그 화일의 최대 크기를 초과하면 새로운 로그 화일에 기록을 시작하는 로그 스위치가 이루어진다. 그리고 체크포인트 완료 시 액티브 트랜잭션 테이블의 모든 트랜잭션들에 대한 UNDO 로그를 앵커 화일에 기록한다. 앵커 화일(Anchor File)은 체크포인트 시 로그 화일과 데이타베이스 화일에 대한 메타정보 저장 화일로서 구조는 표 4와 같다. 앵커 화일은 체크포인트 상태, 마지막 로그, 로그 개수 정보를 위한 Head Block과 체크포인트 완료 시 수행중인 트랜잭션들의 로그 정보를 위한 Table Info Block으로 구성된다.

<표 4> 앵커 화일 구조

Head Block	
체크포인트 상태	체크포인트 시작/완료
마지막 로그	체크포인트 발생시 로그 화일의 마지막 로그
로그 개수	앵커 화일에 저장된 UNDO 로그 개수
Table Info Block	
로그	체크포인트 완료 시 수행중인 트랜잭션들의 UNDO 로그

데이타베이스 화일은 체크포인트 시 공간 메인 메모리 데이타베이스에 대한 스냅샷 이미지 저장 화일로서 구조는 표 5와 같다. 데이타베이스 화일은 헤더 크기, 시스템 종료 상태, 최대 테이블 개수, 테이블 개수 정보를 위한 Head Block과 테이블 아이디, 첫 페이지, 빈 페이지, 최대 튜플 개수 정보를 위한 Table Info Block 그리고 실제 테이블 페이지 정보를 위한 Body Block으로 구성된다.

<표 5> 데이타베이스 화일 구조

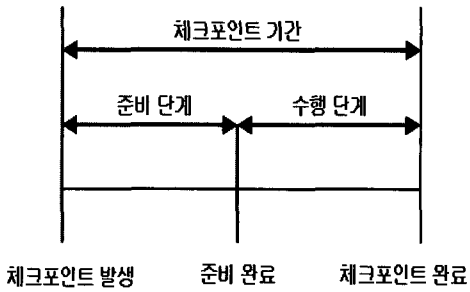
Head Block	
헤더 크기	데이타베이스 화일의 헤더 크기
시스템 종료 상태	시스템 정상 종료 여부
최대 테이블 개수	데이타베이스 화일의 최대 테이블 개수
테이블 개수	데이타베이스 화일에 저장된 테이블 개수
Table Info Block	
테이블 아이디	테이블 아이디
첫 페이지	테이블의 첫 번째 페이지 위치
빈 페이지	테이블의 비어있는 페이지 위치
최대 튜플 개수	페이지 당 최대 튜플 개수
Body Block	
페이지	테이블 페이지

3.2 퍼지-쉐도우 체크포인트 기법

공간 메인 메모리 DBMS에서는 평상 시 트랜잭션 수행속도가 매우 중요하기 때문에 체크포인트 수행은 시스템 전체의 성능에 매우 큰 영향을 미친다. 왜냐하면 체크포인트 수행은 각 트랜잭션들의 수행에 직접적인 영향을 미치고, 로그와 마찬가지로 메인 메모리 내에서의 연산이 아닌 비용이 큰 디스크 입출력을 필요로 하기 때문이다.

본 논문에서는 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 개선하여 체크포인트 수행 시 공간 메인 메모리 데이타베이스의 더티 페이지를 데이타베이스 화일의 새로운 빈 페이지에 기록하고, 체크포인트 수행 중 시스템 오류 시 기존 페이지를 이용하여 회복함으로써 데이타베이스 일관성을 유지시켜 주는 퍼지-쉐도우 체크포인트 기법을 제안한다.

이 기법은 그림 2와 같이 체크포인트 발생 시 실질적인 체크포인트 수행을 위해 준비를 하는 체크포인트 준비단계와 실제 디스크로 백업이 수행되는 체크포인트 수행단계로 구성된다. 체크포인트 준비단계는 현재 부분 완료 상태의 모든 트랜잭션들을 완료시키는 과정이다. 체크포인트 수행단계는 체크포인트 준비 단계 후 실제 체크포인트를 수행하여 공간 메인 메모리 데이타베이스의 더티 페이지들을 데이타베이스 화일에 반영하는 과정이다.



<그림 2> 퍼지-쉐도우 체크포인트 단계

퍼지-쉐도우 체크포인트 기법은 체크포인트 발생 시 체크포인트를 위한 프로세스를 생성한 후 체크포인트를 수행하며, 체크포인트는 다른 트랜잭션들과 비동기적으로 수행된다. 이때, 동일한 공간 메인 메모리 데이타베이스의 페이지에 대해 갱신 연산 트랜잭션과 체크포인트가 동시에 수행될 수 있다. 따라서 체크포인트 수행 중에 데이타베이스 일관성 유지를 위하여 공간 메인 메모리 데이타베이스의 더티 페이지를 데이타베이스 화일의 기존 페이지에 반영하지 않고 새로운 페이지를 할당하여 이 새로운 페이지에 반영하는 쉐도우 갱신을 사용한다. 한번의 체크포인트 작업이 성공적으로 완료되면 기존의 데이타베이스 화일의 페이지들에 대한 링크를 새로이 할당된 페이지로 변경한다. 그리고 기존의 페이지를 빈 페이지로 설정하여 다음 체크포인트 수행 시 새로운 페이지로 사용가능하게 한다.

이러한 퍼지-쉐도우 체크포인트 기법의 과정을 단계별로 기술하면 그림 3과 같다. 그림 4에서 1)~3)은 체크포인트 준비단계, 4)~8)은 체크포인트 수행단계를 나타낸다.

1) 시간 주기, 로그 화일의 개수, 사용자 명령에 의해 체크포인트발생
2) 액티브 트랜잭션 테이블에서 부분 완료 상태의 모든 트랜잭션 Ti에 대해 다음 과정 수행
 2.1) 전역 로그 버퍼에서 Ti에 대한 모든 로그를 로그 화일에 기록
 2.2) Ti에 의해 변경된 페이지들을 더티 페이지 테이블에 기록
 2.3) Ti를 완료상태로 변경
3) 로그 화일의 마지막 로그에 대한 정보를 앵커화일에 저장
4) 더티 페이지 테이블에 등록되어 있는 공간 메인 메모리 데이타베이스의 모든 더티 페이지 DPi에 대해 다음 과정 수행
 4.1) 데이타베이스 화일의 새로운 페이지 NPi 할당
 4.2) DPi를 NPi에 기록
5) 체크포인트 완료 시 액티브 트랜잭션 테이블의 모든 트랜잭션들에 대한 UNDO 로그를 앵커 화일에 기록
6) 데이타베이스 화일의 기존 페이지에 대한 링크를 새로운 페이지로 변경
7) 기존 페이지를 빈 페이지로 변경(다음 체크포인트 시 새로운 페이지로 사용)
8) 체크포인트 완료

<그림 3> 퍼지-쉐도우 체크포인트 수행 과정

3.3 로그 관리

공간 메인 메모리 DBMS를 위한 회복 시스템에서 사용하는 로그는 각 트랜잭션 수행 단계에서 기록되는 지역 로그와 트랜잭션의 전체 실행 결과가 기록되는 전역 로그로 나누어진다. 지역 로그는 트랜잭션 수행 시에 메인 메모리 내에 각 연산의 수행 내용이 기록된다. 본 회복 시스템은 트랜잭션 단위로 지역 로그를 관리함으로서 동시에 수행되는 트랜잭션들 간의 로깅 경합을 줄여 성능 저하를 방지한다. 지역 로그는 메모리상에서만 존재하며 이중 연결 리스트의 형태를 가진다.

로그 화일은 두 개 이상이 순차적으로 생성된다. 하나의 화일에 로그를 계속해서 기록한다면 로그의 크기는 한없이 커지게 되고 회복 시스템은 이를 효율적으로 관리하기 어려워진다. 그래서 로그 화일의 최대 크기를 지정하여 기록되는 로그의 양이 이 크기를 초과하게 되면 새로운 로그 화일을 생성하여 일정 개수의 로그 화일을 유지하도록 한다. 이때 하나의 로그 화일에 기록하는 작업을 끝내고 다음 새로운 로그 화일에 대한 기록을 시작하는 것을 로그 스위치(Log Switch)라 한다.

그리고 회복 시스템에서는 디스크 입출력 감소와 트랜잭션 동시성 향상을 위해 사전 완료 기법을 사용하였다. 사전 완료 기법은 트랜잭션 연산 수행 과정, 부분 완료 과정, 그리고 완료과정 순으로 수행된다. 트랜잭션의 모든 연산이 수행되는 동안의 일련의 과정을 트랜잭션 연산 수행 과정이라 한다. 트랜잭션의 모든 연산 작업이 끝난 후 해당 트랜잭션의 지역 로그 버퍼에서 REDO 로그만 전역 로그 버퍼로 이동시킨 후 트랜잭션을 부분 완료 상태로 변경하는 작업을 부분 완료 과정이라 한다. 그리고 특정시점(전역 로그 버퍼가 가득 차거나 일정한 시간 주기)이 되면 전역 로그 버퍼에서 부분 완료 상태에 있는 모든 트랜잭션들의 로그를 디스크상의 로그 화일로 기록한 후 해당 트랜잭션들을 완료시키는 일련의 과정을 완료 과정이라 한다.

로그 화일의 로그는 추후에 발생할 시스템 오류에 대비하기 위해 일정 기간 동안 보존되고, 최근 체크포인트 이후의 모든 완료된 트랜잭션들의 수행 결과를 저장하고 있어야 한다. 그림 4는 사전 완료 기법의 과정을 단계별로 보여준다. 그림 2에서 1)~2)는 트랜잭션 연산 수행 과정, 3)~5)는 부분 완료 과정, 그리고 6)은 완료 과정을 나타낸다.

1) 트랜잭션 Ti가 시작되면 액티브 트랜잭션 테이블에 Ti 등록
2) 변경연산에 대한 UNDO 로그를 지역 로그 버퍼에 기록 후 공간 메인 메모리 데이타베이스 갱신, 그리고 REDO 로그를 지역 로그 버퍼에 기록
3) 모든 연산 작업이 끝나면 해당 트랜잭션의 지역 로그 버퍼에서 REDO 로그만 전역 로그 버퍼로 이동
4) 액티브 트랜잭션 테이블에서 Ti를 부분 완료 상태로 변경
5) Ti가 가지고 있던 모든 잠금을 해제
6) 특정시점에 액티브 트랜잭션 테이블에서 부분 완료 상태의 모든 트랜잭션 Ti에 대해 다음 과정 수행
   6.1) 전역 로그 버퍼에서 Ti에 대한 모든 로그를 로그 화일에 기록
   6.2) Ti에 의해 변경된 페이지들을 더티 페이지 테이블에 기록
   6.3) Ti를 완료 상태로 변경, Ti가 완료되었음을 통보

<그림 4> 사전 완료 수행 과정

3.4 회복 관리

본 논문에서 설계하고 구현한 공간 메인 메모리 DBMS를 위한 회복 시스템은 메인 메모리상의 로그, 데이타베이스 화일, 로그 화일에서 최근 체크포인트 이후 반영되지 않은 완료된 트랜잭션들의 REDO 로그, 그리고 앵커 화일에서 최근 체크포인트 완료 시 수행 중이던 트랜잭션들의 UNDO 로그를 이용하여 회복을 수행한다.

본 논문에서는 오류의 종류를 크게 트랜잭션 오류와 시스템 오류로 구분한다. 트랜잭션 오류 시 회복은 해당 트랜잭션에 대해 다음과 같이 수행된다. 해당 트랜잭션의 UNDO 로그를 참조하여 각 연산을 철회시키는 작업을 수행하며 그 과정은 그림 5와 같다. 이때, 철회시키고자 하는 트랜잭션에 연관된 트랜잭션이 있을 시 연관된 모든 트랜잭션의 역순으로 그림 5의 과정을 반복 수행하게 된다.

1) 트랜잭션의 UNDO 로그를 이용하여 해당 트랜잭션에 의해 수행된 연산을 철회
2) 트랜잭션의 모든 로그를 폐기
3) 트랜잭션이 보유하고 있던 모든 잠금 해제
4) 트랜잭션 종료
5) 트랜잭션이 철회되었음을 통보

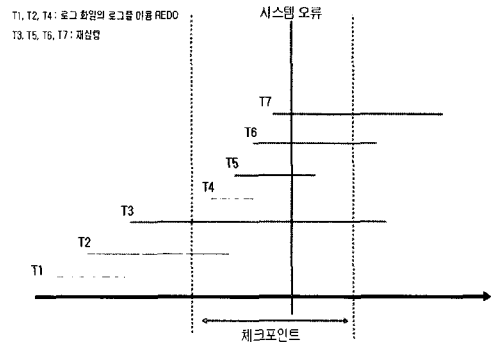
<그림 5> 트랜잭션 오류 시 회복 수행 과정

시스템 오류는 오류가 언제 발생하였느냐에 따라 첫째, 체크포인트 수행 중 시스템 오류, 둘째, 일반적인 상황에서 시스템 오류, 셋째, 회복 수행 중 시스템 오류와 같이 구분할 수 있다. 체크포인트 수행 중에 오류가 발생한 경우 다음과 같이 회복을 수행한다. 우선 디스크에 있던 데이타베이스 화일의 기존 페이지들을 이용하여 메인 메모리에 데이타베이스를 적재시킨다. 그리고 앵커 화일에 기록된 로그를 이용하여 UNDO를 수행하고 로그 화일에 기록된 로그를 이용하여 REDO를 수행한다. 이를 단계별로 자세히 기술하면 그림 6과 같다.

1) 데이타베이스 화일의 기존 페이지들을 이용하여 공간 메인 메모리 데이타베이스 적재
2) 이전 체크포인트 완료 시 앵커 화일에 기록된 로그들에 대한 UNDO 수행
3) 이전 체크포인트 발생 시 앵커 화일에 기록된 마지막 로그에 대한 정보를 참조하여 로그 화일에서 해당 로그 이후 기록된 로그들에 대한 REDO 수행

<그림 6> 체크포인트 수행 중 시스템 오류 시 회복 수행 과정

체크포인트 수행 중 시스템 오류 시 각 트랜잭션 유형별로 회복하는 방법을 정리하면 그림 7과 같다. 그림 7에서 트랜잭션 T1, T2, T4에 대해서는 로그 화일에 기록된 해당 트랜잭션의 로그들을 이용하여 REDO를 수행한다. 그리고 T3, T5, T6, T7은 재실행(Reexecute)을 해주어야 하는 트랜잭션들이다.



<그림 7> 체크포인트 수행 중 시스템 오류 시 트랜잭션별 회복

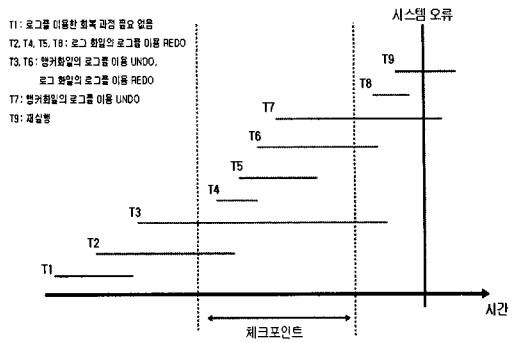
일반적인 상황에서 시스템 오류가 발생한 경우 다음과 같이 회복을 수행한다. 우선 디스크에 있던 데이타베이스 화일의 새로운 페이지들을 이용하여 메인 메모리에 데이타베이스를 적재시킨다. 그리고 앵커 화일에 기록된 로그를 이용하여 UNDO를 수행하고 로그 화일에 기록된 로그를 이용하여 REDO를 수행한다. 이를 단계별로 자세히 기술하면 그림 8과 같다.

1) 최근 완료된 체크포인트 수행 시 데이타베이스 화일에 새로이 반영된 페이지들을 이용하여 공간 메인 메모리 데이타베이스 적재
2) 해당 체크포인트 완료 시 앵커 화일에 기록된 로그들을 이용하여 UNDO 수행
3) 해당 체크포인트 발생 시 앵커 화일에 기록된 마지막 로그에 대한 정보를 참조하여 로그 화일에서 해당 로그 이후 기록된 로그들에 대한 REDO 수행

<그림 8> 일반적인 상황에서 시스템 오류 시 회복 수행 과정

일반적인 상황에서 시스템 오류 시 각 트랜잭션 유형별로 회복하는 방법을 정리하면 그림 9와 같다. 그림 9에서 트랜잭션 T1인 경우 갱신된 공간 데이타는 이미 체크포인트 되었으므로 로그를 이용한 회복 수행 과정은 필요 없다. T2, T4, T5, T8에 대해서는 로그 화일에 기록된 해당 트랜잭션의 로그들을 이용하여 REDO를 수행한다. T3과 T6인 경우 먼저 앵커 화일에 기록된 해당 트랜잭션의 로그들을 이용하여 UNDO를 수행한 후 로그 화일에 기록된 해당 트랜잭션의 로그들을 이용하여 REDO를 수행한다. T7에 대해서는 앵커 화일에서 해당 트랜잭션의 로그들을 이용하여 UNDO를 수행한다. 그리고 T9인 경우는 재실행을 해주어야 하는 트랜잭션이다.

회복 수행 중 시스템 오류가 발생하더라도 데이타베이스 화일과 로그 화일 그리고 앵커 화일의 내용에는 변화가 없게 된다. 그러므로



<그림 9> 일반적인 상황에서 시스템 오류 시 트랜잭션 유형별 회복

회복 중에 시스템 오류가 발생한 경우 수행중인 회복 과정을 처음부터 다시 수행하는 것만으로 해결할 수 있다. 즉, 초기의 시스템 오류가 어느 경우였느냐에 따라 그림 6 또는 그림 8의 과정을 다시 수행하여 회복한다.

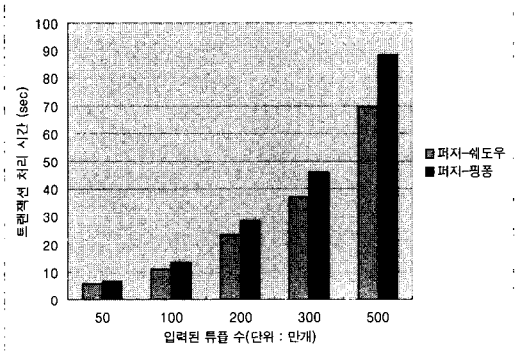
4. 성능 평가

본 장에서는 본 논문에서 개발한 공간 메인 메모리 DBMS를 위한 회복 시스템에 대한 성능 평가 결과를 살펴본다. 성능 평가를 위해 사용된 시스템의 하드웨어 사양은 듀얼 Intel Xeon CPU 2.40GHz, 2GB RAM이며, 운영체제는 RedHat Linux 9.0을 사용하였다. 그리고, 본 회복 시스템의 성능 평가를 위해서 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템과 비교 실험하였다.

4.1 트랜잭션 처리 성능 평가

그림 10은 본 논문에서 제시한 퍼지-쉐도우 체크포인트 기법을 사용하는 회복 시스템과 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템과의 공간 데이타 입력 시 트랜잭션 처리 성능을 비교한 그래프이다. 이때, 공간 객체는 실수형(Double) 포인트(X, Y)의 객체를 임의로 생성하였고, 입력된 튜플

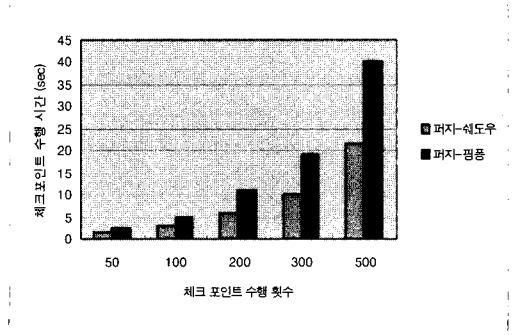
의 수는 500,000개, 1,000,000개, 2,000,000개, 3,000,000개, 5,000,000개이었다. 그림 10의 실험 결과를 통해 본 논문에서 구현한 시스템이 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템보다 평균 25%의 성능 향상을 보임을 알 수 있다.



<그림 10> 트랜잭션 처리 성능 비교

4.2 체크포인트 성능 평가

그림 11은 본 논문에서 제시한 퍼지-쉐도우 체크포인트 기법을 사용하는 회복 시스템과 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템과의 체크포인트 수행 성능을 비교한 그래프이다. 이때, 공간 객체는 실수형(Double) 포인트(X, Y)의 객체를 임의로 생성하였고, 입력된 튜플의 수는 5,000,000개이며, 전체 튜플 입력 중 매 10,000번째 공간 데이타의 입력 발생 시 마다 체크포인트를 발생시켰다.

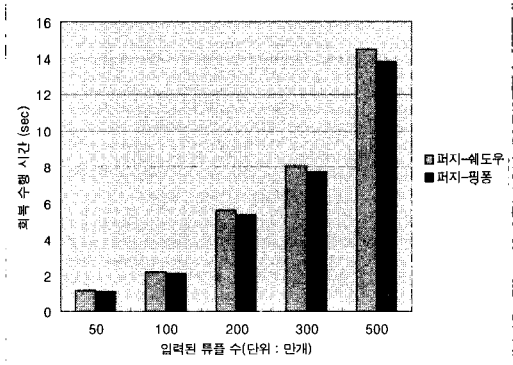


<그림 11> 체크포인트 수행 성능 비교

그림 11의 실험 결과를 통해 본 논문에서 구현한 시스템이 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템보다 평균 88%의 성능 향상을 보임을 알 수 있다.

4.3 회복 성능 평가

그림 12는 본 논문에서 제시한 퍼지-쉐도우 체크포인트 기법을 사용하는 회복 시스템과 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템과의 시스템 오류 발생 후 회복 성능을 비교한 그래프이다. 이때, 공간 객체는 실수형(Double) 포인트(X, Y)의 객체를 임의로 생성하였고, 입력된 튜플의 수는 500,000개, 1,000,000개, 2,000,000개, 3,000,000, 5,000,000개 이었다. 그림 12의 실험 결과를 통해 본 논문에서 구현한 시스템이 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템보다 평균 2%의 성능 감소를 보임을 알 수 있다.



<그림 12> 회복 수행 성능 비교

앞의 세 가지 실험 결과를 통해 본 논문에서 구현한 퍼지-쉐도우 체크포인트 기법의 회복 시스템이 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법의 회복 시스템보다 트랜잭션 처리와 체크포인트 수행에서는 우수한 성능을 가짐을 알 수 있으며, 회복 수행 시에는 약간의 성능 감소가 있음을 알 수 있다. 하지만 회복 수행은 시스템 오류가 발생하였을 경우에만 수행

되기 때문에 시스템이 정상적으로 동작중인 상황에서는 트랜잭션 처리 성능과 체크포인트 수행 성능이 회복 시스템에서 더욱 중요하다.

5. 결론 및 향후 연구

최근 지리 정보 시스템 기술이 발전함에 따라 위치 기반 서비스, 텔레매틱스, 지능형 교통 시스템 등과 같은 GIS 응용 분야에서 복잡한 공간 데이타의 빠르고 효율적인 처리가 절실히 요구되고 있다. 따라서, 디스크 기반 DBMS에 비해 구조가 간단하면서 높은 성능을 낼 수 있는 공간 메인 메모리 DBMS에 대한 높은 관심과 더불어 이에 대한 연구 개발이 활발히 진행되고 있다.

공간 메인 메모리 DBMS에서는 휘발성인 메인 메모리의 특성으로 인해 컴퓨터 고장, 시스템 붕괴, 시스템 오류 등과 같은 예기치 않은 상황 발생 시 메인 메모리에 저장된 모든 공간 데이타를 잃어버릴 수 있으므로 공간 메인 메모리 DBMS에서는 데이타베이스의 안정성을 위한 효율적인 회복 시스템이 중요하다. 또한 공간 메인 메모리 DBMS에서는 대부분의 작업들이 메인 메모리상에서 수행되는데 반하여 회복을 위한 로그 기록과 체크포인트 수행 과정 중 발생하는 디스크 입출력은 시스템 성능을 저하시키는 중요한 요인이다.

이에 본 논문에서는 공간 메인 메모리 DBMS를 위한 효율적인 회복 시스템을 설계 및 구현하였다. 회복 시스템은 사전 완료 기법의 로그 기법을 사용하여, 여러 트랜잭션들의 로그를 한번에 로그 화일에 기록함으로서 디스크 입출력 감소가 가능하다. 그리고 트랜잭션의 모든 연산이 수행된 후 트랜잭션이 보유한 잠금을 미리 해제하여 트랜잭션 동시 수행 성능을 향상할 수 있는 로그 처리가 가능하다.

또한, 회복 시스템은 데이타베이스 일관성을 유지하기 위해 대다수의 메인 메모리 DBMS에서 이용하고 있는 핑퐁 갱신을 기반으로 하는 퍼지 체크포인트 기법에서 발생하는 동일한 페이지에 대한 중복된 체크포인트 수행과 공간낭비의 문제점을 개선하였다. 즉, 체크포인트 수행 시 공간 메인 메모리 데이타베이스의 더티 페이지를 데이타베이스 화일의 새로운 빈 페이지에 기록하고, 체크포인트 수행 중 시스템 오류 시 기존 페이지를 이용하여 회복하는 퍼지-쉐도우 체크포인트 기법을 제안하고 적용하였다.

본 논문에서는 회복 시스템의 성능 평가를 위해서 기존의 핑퐁 갱신 기반 퍼지 체크포인트 기법을 사용한 회복 시스템과 비교 실험을 수행하였다. 실험 결과를 통해 기존의 시스템보다 본 논문에서 제시한 회복 시스템이 전체 시스템 트랜잭션 처리와 체크포인트 수행에서 높은 성능 향상을 보였다. 향후에는 시스템 오류 발생 후 회복 처리의 성능 향상을 위해 보다 효율적인 데이타베이스 화일 구조에 대한 연구가 필요하겠다.

참고문헌

1. Yun, J. K., Kim, J. J., Hong, D. S., and Han, K. J., "Development of an Embedded Spatial MMDBMS for Spatial Mobile Devices," Proc. of the 5th International Workshop on Web and Wireless Geographical Information Systems, 2005, pp.1-10.
2. Kairos Kairos Spatial. http://www.realtimetech.co.kr
3. 차상균 외 9인, "P*TIME: 제2세대 고성능 메인 메모리 DBMS," 한국정보과학회 2001년 춘계학술대회, 제28권, 제1호, 2001, pp.193-195.
4. 이인선, "분산 메인 메모리 데이타베이스 시스템을 위한 인과적 완료방식과 이중화," 서울대학교 대학원 공학박사학위논문, 2003.
5. The Times Ten Team, "In-Memory Data

Management for Consumer Transactions The TimesTen Approach," Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1999, pp.528-529.
6. 한국데이타베이스진흥센터, MM-DBMS-DDBMS의 파트너, 2004.
7. Jagadish, H. V., Sliberschatz, A., and Sudarshan, S., "Recovering from Main-Memory Lapses," Proc. of Int. Conf. on Very Large Databases, 1993, pp.391-404.
8. Le, G., Jing, H., Margaret, H. D., and Jun-Lin L., "Recovery in Main Memory Databases," Int. Journal of Engineering Intelligent Systems, Vol.4, No.3, 1996, pp.177-184.
9. Bohannon, P., Rastogi, R., Seshadri, S., Silberschatz, A., and Sudarshan, S., "Detection and Recovery Techniques for Database Corruption," IEEE Transactions on Knowledge and Data Engineering, 2003, pp.1120-1136.
10. Li, X., and Eich, M., "Post-crash Log Processing for Fuzzy Checkpointing Main Memory Databases," Proc. of the Ninth Int. Conf. on Data Engineering, 1993, pp.117-124.
11. Salem, K., and Garcia-Molina, H., "Checkpointing Memory-Resident Databases," Proc. of the Fifth Int. Conf. on Data Engineering, 1989, pp.452-462.
12. Garcia, H., and Salem, K., "Main Memory Database Systems: An Overview," IEEE Transactions on Knowledge and Data Engineering, Vol.4, No.6, 1992, pp.509-516.
13. 이인선, 염헌영, "주기억 장치 데이타베이스 시스템을 위한 디스크 그룹 완료 프로토콜," 한국정보과학회 논문지, 제31권, 제5호, 2004, pp.516-526.
14. Lehman, T. J., and Carey, M. J., "A Recovery Algorithm for A High Performance Memory-Resident Database System," Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1987, pp.104-117.
15. Margaret H. D., Jun-Lin L., and Xi L., "Fuzzy Checkpointing Alternatives for Main Memory Databases," Recovery Mechanisms in Database Systems, Prentice-Hall, Inc., 1997.
16. 장지웅, 문양세, 권영식, 황규영, "주기억 장치 데이타베이스 시스템에서 트랜잭션-일관성을 유지하는 비정지 체크포인팅 방법," SIGDB-KDBC, 한국정보과학회 데이타베이스연구회, 2003, pp.232-239.
17. Gray, J., and Reuter, A., "Transaction Processing: Concepts and Technology," Morgan Kaufmann Publishers, Inc., 1993.
18. Salem, K., and Garcia-Molina, H., "System M: A Transaction Processing Testbed for Memory Resident Data," IEEE Transactions on Knowledge and Data Engineering, Vol.2, No.1, 1990, pp.161-172.
19. Bohannon, P., Daniel, F., Rajeev, R., Abraham, S., Seshadri, S., and Sudarshan, S., "The Architecture of the Dali Main-Memory Storage Manager," The Journal of Multimedia Tools and Applications, Vol.4, No.2, 1997, pp.115-151.
20. Baulier J., et. al., "DataBlitz Storage Manager: Main Memory Database Performance for Critical Applications," Proc. of the ACM SIGMOD/PODS Int. Conf. on Management of Data, 1999, pp.519-520.
21. Altibase Spatio Temporal DBMS. http://www.altibase.com

김정준
2003년 건국대학교 컴퓨터공학과(공학사)
2004년 건국대학교 대학원 컴퓨터공학과(공학석사)
2005년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야: 공간 메인 메모리 데이타베이스, GIS, LBS, 텔레매틱스

주성완
2003년 건국대학교 컴퓨터공학과(공학사)
2006년 건국대학교 대학원 컴퓨터공학과(공학석사)
2006년~현재 한국게임산업개발원 산업진흥팀 연구원
관심분야: 공간 메인 메모리 데이타베이스, GIS, LBS

강홍구
2002년 건국대학교 컴퓨터공학과(공학사)
2004년 건국대학교 대학원 컴퓨터공학과(공학석사)
2004년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야: 공간 데이타베이스, GIS, LBS, USN, 센서 데이터베이스

홍동숙
1999년 건국대학교 컴퓨터공학과(공학사)
2001년 건국대학교 대학원 컴퓨터공학과(공학석사)
2000년~2003년 쌍용정보통신 모바일/GIS 기술팀
2003년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야: 데이터베이스, 이동체 데이터베이스, 유비쿼터스 컴퓨팅

한기준
1979년 서울대학교 수학교육학과(이학사)
1981년 한국과학기술원(KAIST) 전산학과(공학석사)
1985년 한국과학기술원(KAIST) 전산학과(공학박사)
1985년~현재 건국대학교 컴퓨터공학부 교수
1990년 Stanford 대학 전산학과 Visiting Scholar
2004년~2006년 한국공간정보시스템학회 회장
2004년~현재 한국정보시스템감리사협회 회장
관심분야: 공간 데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템 감리