# 비휘발성 이중면 FeRAM을 이용한 데이타베이스 시스템의 회복 알고리즘

김 용 걸[†] · 박 진 원[††] · 진 성 일[†††] · 조 성 현[††††]


## 요 약

휘발성 메모리를 이용하는 데이타베이스 관리 시스템은 시스템 고장에 대비한 데이타 보호를 위한 회복 기능을 가진다. 이러한 회복 기능은 트랜잭션 처리를 위한 시스템의 부담을 가중시키고 있으며 시스템 성능 저하의 주요 요인이 되고 있다. 최근 반도체 기술의 발달로 인하여 비휘발성 메모리가 등장하게 되었고, 비휘발성 메모리인 FeRAM(Ferroelectronic Random Access Memory)을 이용하여 데이타베이스 관리 시스템이 안고 있는 트랜잭션 처리 및 회복을 위한 부담을 감소시키는 연구가 계속되고 있다. 그러나 기존의 이중면 FeRAM은 데이타베이스 특성인 작은 단위 로킹을 제공하지 못하는 문제를 가진다.

본 논문에서는 이러한 문제를 해결하기 위한 이중면 FeRAM(Dual plane FeRAM : DFeRAM)의 구조를 제안한다. 또한 이중면 FeRAM을 적용한 시스템에 대해 그림자 페이지 기법을 기반으로 하는 회복 알고리즘을 제안하고 제안된 기법과 기존 기법과의 성능을 분석하였다.


# A Recovery Algorithm for Database Systems using Nonvolatile DFeRAM

Yong Keol Kim[†] · Jin Won Park[††] · Seong Il Jin[†††] · Sung Hyun Cho[††††]


## ABSTRACT

Database management systems(DBMS) using volatile memory should have a recovery function to protect data against system failures. Recovery requires much overhead in transaction processing and is one of the great factors that deteriorate the system performance. Recently, there have been a lot of studies on database systems with nonvolatile memory to enhance the performance. A nonvolatile memory called DFeRAM is one of the promising memory devices of the future technology, but this device does not support fine-granularity locking.

In this paper, we present a dual plane FeRAM(DFeRAM) architecture to support the fine-granularity locking. We also propose a recovery algorithm for the database system with the DFeRAM based on a shadow paging method. In order to analyze the performance of the proposed algorithm, we present an analytical model and analyze the performance using the model.


## 1. 서 론

데이타베이스에서의 회복은 트랜잭션의 원자성(atomicity)을 보장하기 위한 데이타베이스 관리 시스템의 필수 기능이다. 이러한 기능을 제공하기 위해 데이타베이스 관리 시스템에서는 데이타베이스의 회복을 위한 많은 부담을 감수하고 있다. 따라서 회복을 위한 부담을 감소시키려는 연구가 매우 활발히 진행되

† 정 회 원 : 대전보건전문대학 사무자동화과
†† 정 회 원 : 한국전자통신연구소 시스템 공학 연구실
††† 종신회원 : 충남대학교 컴퓨터과학과
†††† 정 회 원 : 홍익대학교 과학기술대학 전기전자컴퓨터공학부
논문접수 : 1996년 10월 24일, 심사완료 : 1997년 1월 31일

어 왔다. 가장 대표적인 회복 기법으로는 로그를 사용하는 기법과 그림자 페이지를 사용하는 기법이 있다[5][8]. 그림자 페이지 기법은 로그를 사용하는 기법에 비해 페이지 테이블의 입출력과 클러스터링에 문제가 있어, 일반적으로 상용화된 데이타베이스 관리 시스템은 로그를 기반으로 하는 회복 기법을 사용하고 있다[14].

이러한 회복 기법은 휘발성 메모리가 안고 있는 문제에서 비롯되었다. 그러나 최근 반도체 기술의 발달로 인하여 비휘발성 메모리가 등장하게 되었고, 이러한 비휘발성 메모리의 등장은 데이타베이스 관리 시스템이 가지는 트랜잭션 처리 및 회복을 위한 문제점을 많이 감소시킬 수 있을 것이다.

지금까지 발표된 논문에 따르면 비휘발성 메모리는 주기억 장치 데이타베이스의 효율적인 데이타베이스 회복을 위해 주로 활용되어 이에 따른 시스템의 성능을 향상시켰다[4][8][13][15]. 그러나 주기억장치 데이타베이스는 수백GB 이상의 대용량 데이타베이스를 주기억장치에 상주시킬 수 없는 단점을 가진다.

Copeland는 디스크 기반 데이타베이스 시스템을 위해 주기억 장치에 안정 메모리(Safe RAM)를 추가하여 데이타베이스 시스템의 성능을 향상시켰다. 이 기법은 스풀링 영역을 안정 메모리로 유지하여 디스크에 대한 입출력 요구가 유휴할 때 백그라운드 프로세스인 입출력 프로세스가 스풀링 영역에 대한 디스크 입출력을 수행하게 하였다. 이러한 스풀링 영역을 이용하여 트랜잭션의 디스크 입출력에 대한 부담을 경감시켜 시스템의 성능을 향상시켰다[3].

그러나 회복을 위하여 로그를 이용하는 기법은 다음과 같은 부담을 가진다. 각 트랜잭션이 갱신을 수행할 때마다 로그 레코드를 생성하는 부담, WAL(Write-Ahead Log) 프로토콜에 따른 로그 레코드의 저장을 위한 동기식 입출력에 대한 부담, 체크포인트 시간 동안 모든 트랜잭션이 기다리는 부담, 로그 관리에 대한 시스템 부담, 그리고 로그 화일의 저장을 위한 저장 용량 부담이 있다.

본 논문에서는 회복을 위한 로그 기법과 그림자 페이지 기법의 단점을 최소화하고 이에 따른 시스템의 성능을 극대화하기 위하여 비휘발성 메모리인 DFe-RAM 구조를 제안하고, 이러한 메모리를 활용한 데이타베이스 시스템에서의 회복 알고리즘을 제안한다. 또한, 기존 기법과의 성능 비교를 위한 성능 모델을 제시하고 이에 따른 결과를 분석한다.

2장에서는 본 논문의 제안 알고리즘에 기반이 되는 그림자 페이지 기법에 대한 배경을 설명하고, 3장에서는 본 논문에서 제안한 비휘발성 메모리의 구조를 설명하고, 4장에서는 본 논문에서 제안한 비휘발성 메모리를 적용한 데이타베이스 시스템의 회복 알고리즘을 설명한다. 5장에서는 로그 기법, 그림자 페이지 기법, 그리고 제안된 기법간의 성능 분석을 수행한다. 마지막으로 6장에서는 결론에 대해 다룬다.

## 2. 관련 연구

그림자 페이지 기법은 로그 기법에 비해 상대적으로 성능이 떨어지는 것으로 발표되었고[2][5], 다음과 같은 문제점으로 인하여 1985년 이후 그 연구가 거의 지속되지 않고 있다. 그림자 페이지 기법의 장점은 회복 기법이 간단하고 회복 시 철회가 필요 없다는 것이다. 반면에 그림자 페이지 기법의 단점은 작은 단위 로킹의 제공, 긴 읽기 전용 트랜잭션, 미디어 회복, 2 단계 완료(commit), 또는 부분적 복귀(partial roll-back)가 어려운 단점을 가지며, 추가적으로 물리적 페이지의 클러스터링이 약하고 페이지 테이블의 입출력 부담이 크다는 단점을 가졌다. 그림자 페이지 기법에 관한 기존 연구는 다음과 같다.

Lorie는 그림자 페이지 알고리즘을 처음으로 제안하였고[9], 많은 교재에서 이 알고리즘을 소개하고 있다. 이 개념은 디스크상의 고정된 위치에 두개의 페이지 테이블을 유지하고, 하나의 페이지 테이블은 현재의 데이타베이스 상태를 포함하고 다른 하나는 이전의 트랜잭션에 일치되는 데이타베이스 상태를 나타내도록 구성되었다. 추가적으로 하나의 마스터 레코드가 있어 현재의 데이타베이스를 나타내는 페이지 테이블이 어느 것인지를 알 수 있다. 읽기 연산은 요구하는 페이지 번호를 현재의 페이지 테이블을 이용하여 물리적 페이지 번호와 매핑하고 이 물리적 페이지를 읽어 수행한다. 쓰기 연산은 새로운 페이지를 할당하고 새로운 페이지에 새로운 내용을 복사하고 새로운 페이지를 포인트하기 위해 현재의 페이지 테이블을 수정한다.

System R은 다중 트랜잭션의 동시성을 지원하기

위하여 로그와 그림자 페이지 기법을 함께 사용하여 해결하였다[5].

Lampson과 Sturgis는 트랜잭션의 원자성을 구현하기 위한 방법을 기술하였다[14]. 그들의 개념은 한 트랜잭션에 의한 모든 변화를 하나의 내포 리스트(intention list)에 저장하는 것이다. 완료의 1단계에서 내포 리스트를 안정 기억 장소에 저장하고 2단계에서 완료를 수행한다. 만일 트랜잭션이 취소되면 내포 리스트는 수행되지 않는다. 만일 내포 리스트가 수행되기 전에 고장이 발생되면 회복 동안에 재실행된다. 내포 리스트는 재시도(redo-only)만을 수행하는 로그 기법의 변형으로 볼 수 있다. 내포 리스트는 그림자 페이지 기법을 사용하는 경우는 물론 그림자 페이지 기법을 사용하지 않는 경우에도 사용이 가능하다. 내포 리스트는 로킹에 의해 서로 다른 트랜잭션들이 서로 다른 페이지에 대한 접근을 보장하기 때문에 다중 트랜잭션의 동시성을 구현할 수 있다.

Agrawal과 DeWit도 내포 리스트를 이용한 증분 페이지 테이블(incremental page table)을 사용하였다[1][2]. 이 기법은 트랜잭션이 수행 중인 경우 전역 페이지 테이블을 이용하는 차이를 가지고 있다.

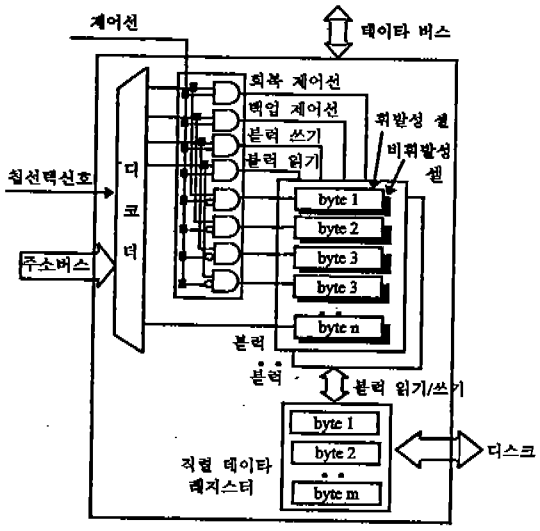
Kent는 페이지 테이블을 트리로 구성하였고, 이미 존재하는 페이지들을 수정할 수 없는 대신에 새로운 물리적인 페이지가 수정된 페이지 테이블의 페이지에 할당된다[7]. 더 높은 레벨의 페이지 테이블의 페이지에는 같은 방법으로 낮은 레벨의 페이지 테이블의 페이지를 포함한다. 가장 높은 레벨의 페이지 테이블의 페이지의 현재 위치를 기록하는데 사용하기 위해 안정 저장 장소의 고정된 위치에 페이지 테이블의 포인터가 있다. 한 트랜잭션에 대한 페이지 테이블의 변화를 가지기 위해 각 트랜잭션마다 중분 페이지 테이블을 사용한다. 이 알고리즘은 다중 트랜잭션의 동시성을 지원하고 모든 트랜잭션의 지속성과 고립화를 지원한다. 또한 쓰레기 수집(Garbage Collection)이 필요 없다.

Ylvnen은 Kent의 그림자 페이지 기법에 근거하여 작은 단위 로킹, 쓰기 최적화, 클러스터링, 미디어 장애, 그리고 2 단계 완료에 대한 방법을 제시하였다[14]. Lorie는 디스크 실린더를 하나의 클러스터로 사용하여 인접한 논리적 페이지는 가능한 같은 실린더에 위치시킨다[9]. Reutor는 각 논리적 페이지에 대한 두개의 물리적 페이지를 할당하기 위하여 TWIST알고리즘을 사용하였다[12]. 타임 스탬프는 어떤 물리적 페이지가 그 페이지의 현재 값을 포함하고 있는지를 결정한다.

## 3. 이중면 FeRAM의 구조

이중면 FeRAM의 각 비트는 휘발성 셀과 비휘발성 셀의 쌍으로 구성된다. 평상시의 읽기와 쓰기는 쓰기 횟수의 제한이 없는 휘발성의 셀에서 수행하고, 특수한 경우에만 휘발성 셀의 내용을 비휘발성 영역으로 백업하여 비휘발성 영역의 쓰기 횟수를 상당 부분 줄임으로써 쓰기 횟수 증가에 따른 유전체 열화 문제를 해결한다[15].

그림자 페이지 기법은 현재 버전과 그림자 페이지 버전을 사용하는 기법이다. 그러므로 이중면 FeRAM의 두개의 면 중 하나는 현재 버전으로 사용하고, 다른 한 면은 그림자 버전으로 사용함으로써 그림자 페이지 기법을 구현할 수 있다. 그러나 현재 사용 가능한 이중면 FeRAM은 정전 시 백업을 위하여 전체 면을 하나의 제어선으로 제어하기 때문에 하나의 현재 버전과 하나의 그림자 버전 만이 존재한다. 그러므로




(그림 1) 페이지 기반 이중면 FeRAM
(Fig. 1) Dual plane FeRAM based on page

여러 트랜잭션이 동시에 수행될 경우 페이지 단위로 제어할 수 없게 되며, 동시성 제어가 어려워진다. 이러한 문제점은 이중면 FeRAM의 백업 및 복구를 분할된 블럭 단위로 수행함으로써 해결된다.

본 논문에서 제안한 이중면 FeRAM의 구조는 (그림 1)과 같다. (그림 1)에서는 페이지 단위의 백업 및 복구를 가능하게 하는 이중면 FeRAM구조를 보여준다. 새로운 메모리 칩은 외부와 연결되는 물리적인 핀의 수가 기존의 메모리와 호환성을 가지는 것이 요구된다. 이를 위해 본 논문에서의 페이지 단위로 백업 및 복구 제어가 가능한 이중면 FeRAM은 기존의 DFeRAM과 같은 수의 핀을 사용하지만, 내부의 디코더는 기존의 DFeRAM과 차이가 난다.

본 논문에서의 페이지 기반 이중면 FeRAM은 주소 버스, 데이타 버스 및 제어선을 사용하지만, 디코더의 기능은 기존의 DFeRAM과 상이하다. 기존의 DFeRAM에서는 제어선에 가해지는 신호의 형태에 의해 백업 또는 복구가 결정되었다. 즉, 하나의 제어선으로 두 가지 기능을 제어한 방법이다. 그러나 페이지 기반 이중면 FeRAM은 제어선이 0일 경우에는 일반적인 메모리 액세스를 하게 하고, 제어선이 1인 경우에는 메모리 내부 각각의 페이지에 연결된 제어선을 구동 시키게 된다. 제어선이 1인 경우 주소 버스가 블럭의 주소 0을 가지면 회복이 수행되고, 1이면 복구를 수행하고, 2이면 디스크 읽기를 수행하고, 3이면 디스크 쓰기를 수행한다.

## 4. 이중면 FeRAM을 적용한 회복 구조

### 4.1 비휘발성 메모리의 적용 사례

회복을 가장 큰 문제점으로 하는 주기억장치 데이타베이스에서는 비휘발성 메모리를 적용한 회복 기법에 대한 연구가 계속 진행되고 있다. Takakura는 검사점 수행시 수행중인 트랜잭션에 영향을 최소화하기 위한 기법을 제안하였으며[13], Lehman은 주 처리기와 회복 처리기를 사용하고 안정 로그 버퍼를 이용한 회복 기법을 제안하였으며[8], Eich는 그림자 메모리, 아카이브 데이타베이스 디렉토리, 검사점 비트맵과 로그 레코드 버퍼를 이용한 회복 기법을 제안하였고[4], [15]는 이중면 FeRAM을 이용한 회복 기법을 제안하였다.

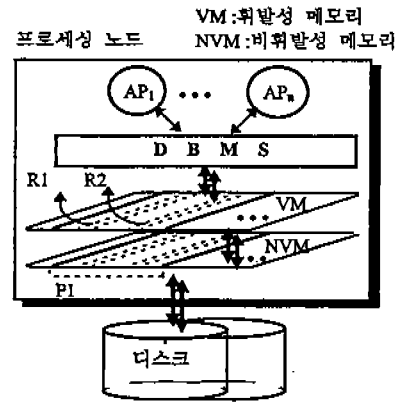
디스크를 기본으로 하는 데이타베이스 시스템에서 Copeland는 스풀링 영역으로 안정 메모리를 유지하여 디스크에 대한 입출력 요구가 유휴할 때 백그라운드 프로세스인 입출력 프로세스가 스풀링 영역에 대한 디스크 입출력을 수행하게 하였다. 이러한 스풀링 영역을 이용하여 트랜잭션의 디스크 입출력에 대한 부담을 경감시켜 시스템의 성능을 향상시켰다[3].

병렬 데이타베이스 시스템에서 Rahm은 동시성/응집성 제어를 위한 메세지의 전송을 감소시키고, 전역 데이타 구조를 저장하기 위한 공유 메모리 영역을 비휘발성 메모리를 사용하여 동시성/응집성 제어 및 입출력 성능을 향상시켰다[11].

### 4.2 제안 회복 알고리즘

일반적으로 로그를 기반으로 하는 회복 구조에서는 각 트랜잭션이 로그를 기록하고, 시스템이 로그를 관리하기 위한 부담이 트랜잭션의 응답 시간과 처리율을 떨어뜨리는 주요 원인이 되고 있다. 그러나 이러한 부담을 줄이기 위한 그림자 페이지 기법은 운영상 페이지 테이블의 디스크 입출력에 많은 부담을 가지고 있고, 디스크상의 클러스터링에도 많은 문제를 가지고 있다.

본 장에서는 이중면 FeRAM을 적용함으로써 로그를 운영하지 않는 그림자 페이지 기법의 문제점을 해결하는 알고리즘을 제안한다. 이중면 FeRAM을 적용한 데이타베이스 시스템 구조는 (그림 2)와 같다.




(그림 2) 데이타베이스 시스템 구조
(Fig. 2) Database system architecture

이중면 메모리의 상위면인 VM(Volatile Memory)을 현재 버전으로 사용하고, 하위면인 NVM(Non-Volatile Memory)은 그림자 버전으로 사용된다. 이전 알고리즘과 같이 디스크상에서 두개의 버전을 관리하기 위한 부담을 줄이고 있으며, 디스크에서는 하나의 페이지만이 유지되는 in-place 갱신을 수행한다.

(그림 3)에서의 내포 리스트는 다음과 같은 기능을 가진다. (1) 다중 트랜잭션의 지원 및 작은 단위 로크 기능을 제공한다. 트랜잭션이 완료되었을 때 VM에서 NVM으로 이동시킬 레코드에 대한 정보를 제공한다. (2) 부분 복귀(partial rollback) 기능을 제공한다. 트랜잭션의 수행 도중 부분 복귀의 요구가 있으면 내포 리스트를 검사하여 복귀한다. (3) 트랜잭션이 철회되었을 때 내포 리스트를 검사하여 NVM의 레코드를 VM으로 이동한다.

이러한 기능을 제공하기 위한 내포 리스트의 각 엔트리는 $\langle P, R_{id}, R_{new}, R_{old} \rangle$이다. P는 페이지 번호를 나타내고, $R_{id}$는 레코드 번호를 나타내고, $R_{new}$는 레코드에 갱신된 새로운 값이고, $R_{old}$는 레코드의 이전 값이다. 내포 리스트는 각 트랜잭션에 대해 유지되어야 하며 트랜잭션이 완료되면 메모리에서 삭제된다. 기존의 내포 리스트는 안정 저장 장소에 저장하여 사용하였으나 본 논문에서의 내포 리스트는 트랜잭션이 수행 중일 때에만 유지된다.

트랜잭션과 입출력 프로세스의 수행 절차는 (그림 3)과 같다. (그림 2)를 참조하여 설명하면 우선 트랜잭션(Tr.1)이 요구한 페이지 P1은 VM과 NVM으로 적재된다. Tr.1이 페이지 P1의 레코드 단위로 접근하

```
1. 트랜잭션 처리를 완료할 때까지 1.1-1.4 수행.
   1.1 페이지 접근을 위해 페이지와 레코드의 로크 획득.
   1.2 요구된 페이지를 VM 과 NVM 으로 적재.
   1.3 페이지 훼손 비트(dirty bit)를 재고정(reset).
   1.4 갱신 수행.
   1.5 접근한 페이지와 레코드 정보를 트랜잭션의 내포 리스트에 첨가.
2. 트랜잭션이 완료된 경우, 2.1-2.7 수행,
   2.1 트랜잭션의 내포 리스트를 NVM 으로 저장.
   2.2 트랜잭션 식별자를 완료 리스트에 첨가.
   2.3 트랜잭션의 내포 리스트에 있는 해당 레코드를 VM 에서 NVM 으로 이동.
   2.4 트랜잭션의 로크 해제.
   2.5 NVM 의 해당 페이지의 훼손 비트를 고정(set).
   2.6 해당 페이지가 입출력 프로세스에 의해 디스크 쓰기를 수행하는지 검사,
       디스크 쓰기 중이면 입출력 프로세스에 인터럽트 요구.
   2.7 완료 리스트의 트랜잭션 식별자 삭제.
3. 트랜잭션이 취소된 경우, 3.1-3.2 수행
   3.1 트랜잭션 내포 리스트의 레코드를 NVM(그림자)에서 VM(현재)으로 이동.
   3.2 트랜잭션의 로크 해제.
```

(그림 3) 트랜잭션의 수행 절차

(Fig. 3) Transaction processing procedure in the database system

```
1. 트랜잭션의 입출력 요구.
   1.1 요구한 페이지를 위해 디스크 접근.
2. 트랜잭션의 입출력 요구가 없으면,
   NVM 에 있는 각 페이지가 훼손 비트가 고정되어 있는 지 순환적으로 검사.
   2.1 고정되어 있으면 디스크로 저장.
   2.2 쓰기 도중 인터럽트를 받으면, 해당 페이지에 대한 디스크 쓰기 재시도.
   2.3 디스크 쓰기가 완료되면, 페이지의 훼손 비트 재고정.
```

(그림 4) 입출력 프로세스의 수행 절차

(Fig. 4) I/O process procedure in the database system

```
1. NVM 의 모든 페이지를 NVM 에서 VM 으로 이동.
2. 완료 리스트에 트랜잭션 식별자가 있는 지 검사.
3. 완료 리스트에 있는 트랜잭션에 대해,
   3.1 내포 리스트를 이용한 재실행.
   3.2 갱신된 레코드를 NVM 으로 이동.
```

(그림 5) 재시동 수행 절차

(Fig. 5) Restart processing procedure in the database system

기 위하여 페이지(P1)와 레코드(R1)를 로크한 후 갱신을 수행한다. 동시에 다른 트랜잭션(Tr.2)이 같은 페이지(P1)의 다른 레코드(R2)를 접근하기 위하여 P1과 R1을 로크한다. 이때 각 트랜잭션은 내포 리스트를 유지하고 있다. 트랜잭션은 로크된 데이타에 대해 갱신을 수행한다. 트랜잭션(Tr.1)의 수행이 완료된 경우에는 Tr.1에 관련된 R1만을 NVM에 저장한다. 이때 해당 페이지가 입출력 프로세스에 의해 디스크 쓰기를 수행하고 있는 경우에는 입출력 프로세스에 인터럽트를 요청하고 NVM으로 저장한다. 이때 인터럽트를 받은 입출력 프로세스는 NVM으로 쓰기를 기다리고(거의 0 시간임) 다시 해당 페이지에 대해 디스크에 쓰기를 다시 수행한다. NVM으로 쓰기가 완료되면 NVM의 해당 페이지의 훼손 비트를 고정하고 트랜잭션은 자신이 가지고 있는 모든 로크를 해제한다.

만일 트랜잭션이 수행 도중 철회하는 경우에는 내포 리스트의 정보를 이용하여 트랜잭션(Tr.1)에 관련된 R1만을 NVM(그림자 버전)에서 VM(현재 버전)으로 저장한 후 트랜잭션의 모든 로크를 해제한다.

(그림 4)의 입출력 프로세스는 디스크 입출력을 담당하는 프로세스이다. 트랜잭션의 디스크 입출력 요구가 없는 경우에는 NVM의 각 페이지의 훼손 비트를 순환적으로 검사하여 고정되어 있는 페이지를 디스크로 이동한다.

정전으로 인한 재시동 시에는VM의 레코드를 NVM

으로 데이타 쓰기를 하는 도중에 정전이 된 트랜잭션에 대해서 내포 리스트를 참조하여 철회(undo)를 수행하는 비용만이 요구된다(그림 5). 또한 미디어 장애를 위해서는 이중면 FeRAM을 이중화하여 구현한다.

이 기법의 장점은 다음과 같다. (1) 트랜잭션의 로그 레코드 유지 및 시스템의 로그 관리 기능이 필요없어 회복을 위한 트랜잭션의 부담이 없다. (2) 기존의 그림자 페이지 기법은 현재 버전과 그림자 버전의 페이지가 필요하여 디스크 용량을 많이 사용하는 단점을 가졌으나, 이 기법에서는 추가적인 디스크의 용량을 요구하지 않는다. (3) 디스크 상에 그림자 페이지 테이블을 사용하지 않으므로 기존의 그림자 페이지 기법의 클러스터링 문제를 해결한다. (4) 시스템 장애 시 재시동 시간이 짧다. (5) 체크 포인트를 위한 부담이 없다.

이 기법의 단점은 다음과 같다. (1) 특수하게 설계된 메모리를 사용한다. (2) 트랜잭션이 페이지를 처리할 때 갱신을 위한 페이지는 모두 메모리에 상주하게 된다(NO-STEAL).

## 5. 성능 분석

### 5.1 비용 모델

동시성 제어와 회복 알고리즘의 성능을 측정하기 위하여 [2]의 방법과 같이 트랜잭션 처리에서의 부담을 모델링하였다. 하나의 트랜잭션이 수행을 시작하면 다음과 같은 3가지 결과가 나올 수 있다. 1) 트랜잭션이 수행을 마치고 완료한다. (2) 트랜잭션이 사용자나 잘못된 데이타 입력으로 철회한다. (3) 트랜잭션이 시스템에 의해 철회되면 재 실행하여 완료한다.

(3)의 경우에는 트랜잭션이 시작한 시간에서 재실행하는 시간까지의 부담과 철회하는 데의 부담과 트랜잭션의 시작부터 완료까지의 부담으로 나눌 수 있다. 복과 동시성 제어 알고리즘에 의해 한 트랜잭션에 부과되는 부담(BX)은 다음과 같다.

$$BX = O_{setup} + P_{fail} * O_{fail} + P_{succ} * O_{succ} + P_{rerun} * O_{rerun} \quad (1)$$

$$P_{succ} + P_{fail} = 1 \quad (2)$$

$O_{setup}$은 트랜잭션의 초기화를 위한 비용이고, $P_{fail}$은 트랜잭션이 실패할 확률이고, $O_{fail}$은 트랜잭션이 실패했을 때의 부담이고, $P_{succ}$는 트랜잭션이 성공할 확률, $O_{succ}$는 트랜잭션이 완료되었을 때의 부담, $P_{rerun}$은 트랜잭션이 재실행할 확률이고, $O_{rerun}$은 트랜잭션이 시스템에 의해 재실행되었을 때 발생하는 부담이다.

- 트랜잭션 모델

하나의 트랜잭션이 접근하는 전체 데이타베이스 페이지의 수는 $NP_t$이다. $NP_t$이외에 $NP_u$는 갱신되는 페이지 수이다. 메모리 히트율에 따라 페이지는 디스크로부터 읽혀지고 트랜잭션의 끝에서 갱신된 페이지가 디스크에 쓰여진다. 각 페이지는 한 트랜잭션에 의해 정확히 한번 읽어진다.

- 시스템 매개변수

$T_{io}$는 하나의 디스크 페이지를 읽거나 쓰기 위한 시간이고, $T_{page}$는 메모리에서 한 페이지를 처리하기 위한 CPU 시간이고, $T_{rec}$는 메모리에서 한 레코드를 처리하기 위한 CPU 시간이고 DBSize는 데이타베이스의 크기이다.

- 동시성 제어에서의 가정

(1) 로크의 획득 요구를 처리하기 위한 시간은 $T_{al}$이고, 로크 해제 요구를 처리하기 위한 시간은 $T_{rl}$이다. (2) 로킹의 단위는 페이지이다. (3) 한 트랜잭션이 $NP_t/2$개의 페이지를 읽고 $NP_u/2$ 페이지를 갱신하였을 때 트랜잭션은 취소된다. (4) 교착상태는 Wait-for Graph에서 사이클을 검사함으로써 해결된다. 하나의 로크 요구가 충돌할 확률은 $P_{conflict}$이고, 이 검사를 위해 소요되는 CPU 시간은 $T_{ddlk}$이며, 사이클이 발견될 확률은 $P_{ddlk}$이다.

### 5.2 이중면 FeRAM을 이용한 그림자 페이지 기법의 비용 함수

#### 5.2.1 가 정

(1) VM에서 NVM으로 쓰기 하는데 걸리는 시간은 0이다.

(2) 각 트랜잭션은 갱신이 수행될 때마다 내포 리스트의 엔트리를 생성해야 한다. 이 엔트리는 로그 레코드와 같은 비용을 가진다.

$$\{ceil(LogFrac * NP_u/2) * Tpage\} \quad (3)$$

(3) 트랜잭션이 철회될 때에는 NVM의 페이지의 해당

레코드만을 VM으로 이동하는 비용만이 요구된다.

철회되는 트랜잭션의 수행 비용 $= T_{page} * NP_t/2$ (4)

(4) 페이지 테이블의 크기는 PtSize이고, S-Map 페이지의 수는 다음과 같이 결정된다.

$$PtPages(X) = PtSize(1-(1-1/PtSize)^X) \quad (5)$$

(5) 데이타나 로그 버퍼의 내용이 디스크로 쓰기 위한 시간이 요구된다.

$$DFlush(X) = \max\{0, X - DBuff\} \quad (6)$$

5.2.2 비용 함수

$O_{setup}$ = 완료/취소 레코드를 쓰는 비용$\{T_{io}\}$. (7)

$Q_{succ}$ = 로크 획득 비용$\{NP_t * (T_{al} + P_{conflict} * T_{ddlk})\}$
+내포 리스트의 엔트리를 갱신하기 위한 CPU 비용$\{NP_u * T_{rec}\}$
+갱신된 S-Map 페이지를 쓰기 위한 I/O 비용$\{PtPages(NP_u) * Tio\}$
+로크 해제 비용$\{NP_t * T_{rl}\}$ (8)

$Q_{fail}$ = 트랜잭션 철회하기 전의 부담
+undo 수행을 위한 부담
= 로크를 획득하고 해제하기 위한 비용
$\{(T_{al} + P_{conflict} * T_{ddlk} + T_{rl}) * NP_t/2\}$
+데이타 페이지의 undo를 위한 CPU 비용
$\{DFlush(NP_u/2) * T_{page}\}$ (9)

$O_{rerun} = O_{fail}$ +철회 전의 트랜잭션 수행 비용
$= O_{fail} + (T_{io} + T_{page}) * NP_t/2 * (1 - Hit_{Memory})$
$+ DFlush(NP_u/2) * T_{io}$. (10)

5.3 매개 변수

〈표 1〉 성능 분석을 위하여 사용한 매개변수들이다. 디스크 입출력을 위한 $T_{io}$는 다음과 같이 계산되었다.

$T_{io}$ = 평균 탐색 시간 +지연 시간 +전송 시간 = 37.525ms (11)

또한 1 MIPS 프로세서를 가정하였으며 하나의 레코드를 처리하기 위한 명령어 수는 500명령어이고, 약 10개의 레코드를 가지는 한 페이지를 처리하는 명령어는 5,000명령어가 필요하다. 데이타베이스 크기(DBSize)는 100MB이다. 그림자 페이지 기법을 위한 매개변수는 데이타베이스 버퍼(DBuff)는 1로, 그림자 페이지 테이블을 저장하기 위하여 사용되는 버퍼(SBuff)는 10으로 하였고, 페이지 테이블의 각 엔트리는 4바이트로 가정하였으며, 그림자 페이지 테이블 맵(S-Map)의 크기는 25페이지이다. 로깅을 위해서는 DBuff 크기는 10으로, 로그 레코드를 저장하기 위한 버퍼(LBuff)의 크기는 1로 하였고, LogFrac는 0.1로 가정하였다. LogFrac는 페이지 처리에 따른 로그 페이지의 생성 비율을 의미한다. 로크 요구에 대한 CPU 처리 시간과 로크의 해제 시간 그리고 교착상태 감시 시간은 각각 0.5ms로 가정하였다.

〈표 1〉 성능 매개 변수
〈Table 1〉 Performance parameters

| 매개변수 | 값 | 매개변수 | 값 |
|---|---|---|---|
| 디스크 | 30 | 로깅 | 10 |
| 표면 수 | 555 | Dbuff | 1 |
| 실린더 수 | 4 | Lbuff | 0.1 |
| 트랙 당 블럭 수 | 4,096바이트 | LogFrac | 0.5ms |
| 블럭 크기 | 16.7ms | 로크 요구의 처리 시간 | 0.5ms |
| 회전 시간 | 25ms | 로크의 해제 시간 | 1 MIPS |
| 평균 탐색 시간 | | 프로세서 처리 속도 | 500명령어 |
| 그림자 페이지 | 1 | 한 개의 레코드 처리 명령어 수 | 5,000명령어 |
| DBuff | 10 | 한 개의 페이지 처리 명령어 수 | 0.1Gbytes |
| SBuff | 4바이트 | DBSize | 37.525ms |
| 페이지 테이블 엔트리 | 25 페이지 | $T_{io}$ | 0.5ms |
| S-Map 크기 | | $T_{ddlk}$ | 0.4 |
| | | $Hit_{Memory}$ | |

〈표 2〉 트랜잭션 크기 및 데이타베이스 특성
〈Table 2〉 Transaction sizes and database characteristics

| 매 개 변 수 | TX1 | TX2 | TX3 | TX4 | TX5 |
|---|---|---|---|---|---|
| 접근한 페이지 수($NP_t$) | 2 | 25 | 50 | 100 | 250 |
| 갱신된 페이지 수($NP_u$) | 1 | 10 | 15 | 25 | 50 |
| 트랜잭션 실패 확률($P_{fail}$) | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| 로크 충돌 확률($P_{conflict}$) | 0.00088 | 0.00436 | 0.00406 | 0.00513 | 0.00525 |
| 재실행할 확률($P_{rerun}$) | 0 | 0 | 0.00246 | 0.03773 | 0.08791 |

〈표 2〉는 다양한 유형의 트랜잭션들에 대한 실험을 위하여 사용된 각 트랜잭션의 크기 및 데이타베이스 특성에 대한 매개변수 값이다.

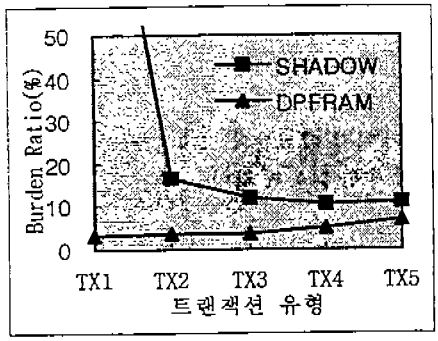
### 5.4 실험 결과

본 논문에서 제안한 기법과의 비교를 위한 로그 기법의 비용 함수와 [9][14]에서 제안된 그림자 페이지 기법의 비용 함수는 [2]를 참고하여 비교하였다. 각 기법에 대한 트랜잭션 수행을 위한 부담율은 다음과 같이 계산된다.

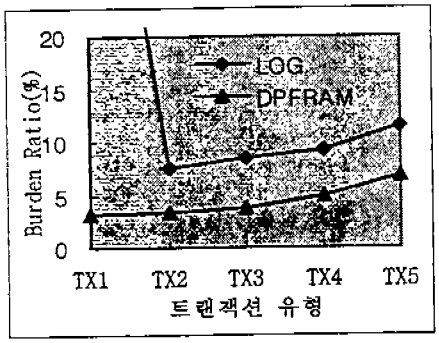
$$\begin{aligned}\text{부담율} &= BX/\text{트랜잭션의 수행 시간} * 100 \\ &= (O_{setup} + P_{fail} * O_{fail} + P_{succ} * O_{succ} + P_{rerun} * O_{rerun})/ \\ &\quad (NP_t * T_{io} * (1\text{-}Hit_{Memory}) + NP_t * T_{page} \\ &\quad + NP_u * T_{io}) * 100 \end{aligned} \quad (12)$$

트랜잭션 유형에 따른 각 기법의 실험 결과는 (그림 6)과 같다. (그림 6(a))는 기존의 그림자 페이지 기법과 제안된 기법의 부담율을 비교한 그래프이다. 짧은 처리를 요구하는 트랜잭션인 경우 그림자 페이지 기법은 페이지 테이블의 디스크 입출력에 대한 부담으로 부담율이 매우 높은(약 93%) 반면에 긴 트랜잭션일수록 부담율이 작아지는(약 10%) 특성을 가진다. 그러나 제안된 기법은 트랜잭션의 유형에 관계없이 6% 이하의 부담율을 가지고 있다. 따라서 제안된 기법을 적용한 시스템은 트랜잭션의 순수한 처리 시간 이외의 부담을 갖지 않게 되어 트랜잭션을 매우 효율적으로 관리할 수 있다. 제안된 기법의 부담율이 작은 이유는 기존의 그림자 페이지 기법과는 달리 디스크상에서는 하나의 페이지만을 유지하고 메모리 부분에서 그림자 페이지 기법을 적용함으로써 그림자 페이지 기법의 가장 큰 단점인 페이지 테이블 입출력 시간을 없앴기 때문인 것으로 분석된다.

(그림 6(b))는 로그 기법과 제안된 기법과의 부담율을 나타낸다. 기존의 로그 기법도 짧은 트랜잭션에 대해 36%정도의 부담을 가지고 있으나 긴 트랜잭션에 대한 부담율은 6.2%에서 10.5%정도의 부담율을 가지고 있다. 그러나 제안된 기법은 2%에서 6%정도의 낮은 부담율을 가져, 전체적으로 부담율이 낮으며 특히 짧은 트랜잭션인 경우에는 로그 기법이 36%, 그림자 페이지 기법이 93%의 부담을 가지는 것에 비해

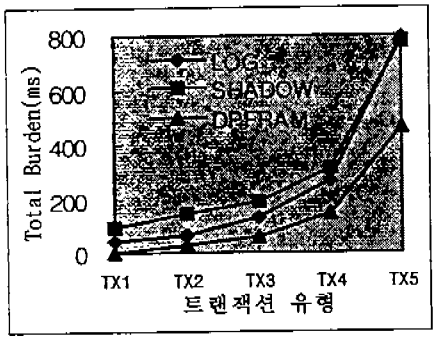



(a) 그림자 페이지 기법과의 비교




(b) 로그 기법과의 비교

(그림 6) 회복 기법들의 부담율
(Fig. 6) Burden ratio of recovery methods




(그림 7) 회복 기법들의 전체 부담 시간
(Fig. 7) Total burden time of recovery methods

약 2%정도의 부담밖에 가지지 않아 짧은 트랜잭션의 처리를 위한 시스템에는 매우 높은 성능을 제공할 것으로 예측된다. 이러한 결과는 제안된 기법이 로그 기법에 비해서는 로그 레코드의 생성, 저장, 그리고 관리의 부담을 없애고, 그림자 페이지 기법에 대해서는 페이지 테이블 입출력 부담을 없앴기 때문인 것으로 분석된다. (그림 7)은 각 기법의 전체 부담 시간을 나타낸 그래프이다. 제안된 기법은 로그 기법이나 그림자 페이지 기법에 비해 부담 시간이 매우 작음을 볼 수 있다. 짧은 트랜잭션인 경우에는 거의 부담 시간이 없으며 긴 트랜잭션인 경우에도 로그 기법이나 그림자 페이지 기법은 0.8초인데 반해 제안된 기법은 0.4초로 부담 시간이 매우 낮다.

## 6. 결 론

그림자 페이지 기법은 페이지 테이블의 입출력 및 클러스터링에 많은 부담을 가지고 있어 데이타베이스의 회복을 지원하기 위한 시스템에 대한 큰 부담을 가지고 있다. 따라서 상용화된 많은 데이타베이스 관리 시스템에서는 로그 기법을 적용하여 데이타베이스 회복을 지원하고 있다. 그러나 로그를 기본으로 하는 회복 기법은 로그 레코드를 생성, 저장 그리고 관리를 위한 시스템의 부담이 큰 단점을 가지고 있다.

본 논문에서는 데이타베이스 회복을 위한 이러한 부담을 최소화하기 위하여 이중면 FeRAM 구조를 제안하고, 이러한 메모리를 적용한 시스템에 대한 회복 알고리즘을 제안하였다. 이중면 FeRAM은 휘발성 기억소자를 가지는 상위면(VM)과 비휘발성 기억소자를 가지는 하위면(NVM)으로 구성된다. 이러한 이중면 구조의 DFeRAM을 데이타베이스 시스템에 적용할 경우 일반적인 단일면 구조의 휘발성 메모리를 사용한 구조보다 우수한 것을 보여 주었다. 특히 디스크 기반에서는 로그 방식에 비해 성능이 떨어지는 그림자 페이지 기법이 이중면 FeRAM 을 사용한 구조에 가장 적합하다는 것을 보였다.

다양한 트랜잭션의 유형에 따라 성능 분석을 수행한 실험 결과는 다음과 같다. (1) 짧은 트랜잭션을 수행하기 위한 시스템의 부담이 로그 기법에 비해서는 위해 본 논문에서 제안된 기법은 기존의 로그나 그림자 페이지를 적용한 시스템의 회복부담 9배 이상의 부담을 줄일 수 있었으며, 그림자 페이지 기법보다는 16배 이상의 부담을 줄일 수 있었다. (2) 긴 트랜잭션에 대해서는 2배 이상의 부담을 줄일 수 있었다.

## 참 고 문 헌


[1] R. Agrawal and D. Dewitt, "Recovery Architecture for Multiprocessor Database Machine," *ACM SIGMOD*, pp.131-145, 1985.

[2] R. Agrawal and D. Dewitt, "Integrated Concurrency Control and Recovery Mechanisms: Design and Performance Evaluation," *ACM Transactions on Database Systems, Vol. 10, No. 4*, pp.529-564, Dec. 1985.

[3] G. Copeland, et. al., "The Case For Safe RAM," *Proceedings of the 15th International Conference on VLDB*, pp.327-335, 1989.

[4] M. H. Eich, "Main Memory Database Research Directions," 88-CS-35, Department of Computer Science and Engineering, Southern Methodist University, 1988.

[5] J. Gray, P. McJones, M. Blasgen, B. Lindsay, R. Lorie, T. Price, F. Putzolu, and I. Triger, "The Recovery Manager of the System R Database Manager," *ACM Computing Survey, Vol. 13, No. 2*, pp.223-242. Jun. 1981

[6] T. Haerder and A. Reuter, "Principles of Transaction-Oriented Database Recovery," *ACM Computing Surveys, Vol. 15, No. 4*, pp.287-317, Dec. 1983.

[7] J. Kent and H. Garcia-Molina, "Optimizing Shadow Recovery Algorithms," *IEEE Transactions on Software Engineering, Vol. 14, No. 2*, pp. 155-168, Feb. 1988.

[8] T.J. Lehman and M. J. Carey, "A Recovery Algorithm for A High-Performance Memory Resident Database System," *Proc. Of Intl. Conf. On Management of Data, ACM SIGMOD*, pp. 104-117, 1987.

[9] R. A. Lorie, "Physical integrity in a large segmented database," *ACM Transactions on Data-*

*base Systems*, 2(1), pp.91-101, 1977.

[10] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," *ACM Transactions on Database Systems, Vol. 17, No. 1*, pp.94-162, Mar. 1992.

[11] E. Rahm, "Performance Evaluation of Extended Storage Architectures for Transaction Processing," *Proc. ACM SIGMOD Conf.*, San Diego, CA, 1992.

[12] A. Reuter, "A fast transaction-oriented logging scheme for UNDO recovery," *IEEE Transactions on Software Engineering*, SE-6(4), pp.348-356, 1980.

[13] H. Takakura and Y. Kambayashi, "A Design of a Transparent Backup System Using a Main Memory Database," *Proc. of Int'l Conf. on Database Systems for Advanced Applications*, pp.178-184, 1993.

[14] T. Ylonen, Shadow Paging is Feasible, Licentiate's Thesis, Department of Computer Science, Helsinki University of Technology, 1994.

[15] 백영식, 주기억장치 데이타베이스 시스템에서 비휘발성 메모리를 이용한 회복 기법, 박사학위 논문, 충남대학교, 1996. 2.

**김 용 걸**

1985년 충남대학교 자연과학대학 계산통계학과(학사)
1987년 충남대학교 자연과학대학 계산통계학과(석사)
1997년 충남대학교 전산학과(박사)
1992년~현재 대전보건전문대학 사무자동화과 교수
관심분야: 병렬데이타베이스, 성능분석, 비디오데이타베이스

**박 진 원**

1975년 서울대학교 공과대학 산업공학과(학사)
1977년~1980년 한국개발연구원(KDI)
1982년 미국 오하이오 주립대학교 산업공학과(석사)
1987년 미국 오하이오 주립대학교 산업공학과(박사)
1987년~1988년 미국 남콜로라도대학교 조교수
1988년~현재 한국전자통신연구원 시스템공학연구실장
관심분야: 컴퓨터 아키텍춰, 시스템 이론, 시뮬레이션

**진 성 일**

1978년 서울대학교 자연과학대학 계산통계학과(학사)
1980년 한국과학기술원 전산학과(석사)
1994년 한국과학기술원 전산학과(박사)
1987년~1989년 Northwestern대학 전산학과 객원교수
1990년~1992년 충남대학교 전자계산소장
1989년~현재 정보과학회 데이타베이스연구회 운영, 편집, 협력위원
1994년~현재 충남대학교 컴퓨터과학과 교수
1995년~현재 정보처리학회 멀티미디어연구회 운영위원
1996년~현재 데이타베이스학회 편집위원
1996년~현재 충남대학교 소프트웨어연구센터 소장
1996년~현재 CALS 학회 충청지부 이사
1996년~현재 정보과학회 평의원
관심분야: 데이타베이스, 성능분석, 정보모델링, 멀티미디어

**조 성 현**

1978년 서울대학교 자연과학대학 계산통계학과(학사)
1980년 서울대학교 자연과학대학 계산통계학과(석사)
1980년~1983년 육군사관학교 교관
1995년 미국 UCLA 전산학과(박사)
1995년~1996년 미국 UCLA 전산학과(Postdoc)
1996년~현재 홍익대학교 과학기술대학 전기전자컴퓨터공학부 교수
관심분야: 분산시스템, 데이터베이스