論文 97-22-9-3

ATM 교환기용 분산 주기억장치 상주 데이터베이스 시스템에서의 T-tree 색인 구조의 회복 기법

正會員 이 승 선*, 조 완 섭**, 윤 용 익***

The T-Tree Index Recovery for Distributed Main-memory Database Systems in ATM Switching Systems

Seung-Sun Lee*, Wan-Sup Cho**, Yong-Ik Yoon*** *Regular Members*

요 약

DREAM-S는 ATM 네트워크용 교환 시스템에서 응용 프로그램들의 교환기 운용 데이터에 대한 실시간 처리 요구를 지원하기 위한 분산 주기억장치 상주 데이터베이스 시스템(Main Memory Database Systems)이다. DREAM-S는 클라이언트-서버 구조를 가지면서 서버 프로세서에만 디스크가 연결되어 있으며, 대량의 데이터로부터 원하는 데이터를 신속히 검색하기 위하여 T-Tree 색인 구조를 제공한다. 본 논문에서는 DREAM-S에서 T-Tree 색인 구조에 대한 회복 기법을 제안한다. 주기억장치 상주 데이터베이스는 디스크 상주 데이터베이스 보다 뛰어난 성능을 제공하지만 시스템 고장 시(정전 등과 같은 오류) 주기억장치에 저장된 모든 데이터(릴레이션과 색인 구조)가 파손될 수 있다. 따라서 고장 후 파손된 주기억장치 데이터베이스를 신속히 정상 데이터베이스 상태로 회복하는 회복 기법이 필수적이다. 제안된 회복 기법에서는 T-Tree 색인 구조를 각 프로세서의 주기억장치에만 유지하도록 함으로서 ATM 교환기 시스템의 성능에서 병목 현상을 일으킬 수 있는 서버 프로세서의 디스크 출입 오버헤드를 줄인다. 또한, 시스템 고장 후 서버와 모든 클라이언트 시스템들이 병렬 처리 방식으로 각자의 T-Tree(들)를 회복하도록 함으로서 클라이언트 개수가 많은 경우에도 신속한 회복이 가능하도록 하였다.

ABSTRACT

DREAM-S is a distributed main-memory database system for the real-time processing of shared operational data in ATM switching systems. DREAM-S has a client-server architecture in which only the server has the disk storage, and provides the T-Tree index structure for efficient accesses to the data. We propose a recovery technique

* 한국전자통신연구소 연구원
** 충북대학교 경영정보학과
*** 숙명여자대학교 전산학과
論文番號:97084-0303
接受日字:1997年 3月 3日

for the T-Tree index structure in DREAM-S. Although main-memory database systems offer efficient access performance, the database in the main-memory may be broken when system failure such as database transaction failure or power failure occurs. Therefore, a recovery technique that recovers the database (including index structures) is essential for fault tolerant ATM switching systems. Proposed recovery technique relieves the bottleneck of the server processors disk operations by maintaining the T-Tree index structure only in the main-memory. In addition, fast recovery is guaranteed even in large number of client systems in the ATM switching systems since the T-Tree index structure(s) in each system can be recovered concurrently.

Ⅰ. 서 론

한국전자통신연구원에서는 비동기 전송 방식(Asynchronous Transfer Mode:ATM)의 교환기 시스템을 개발하고 있다 (이하, ATM 교환기 시스템이라고 칭함). ATM 교환기 시스템에서 호 처리(call processing), 운용(administration), 유지 보수(maintenance and operation) 등의 기능을 수행하는 응용 프로그램들은 공유 데이터를 사용하여 빠른 시간 내에 처리되어야 하는 실시간 특성을 지닌다. 사용자 요구에 대하여 제한된 시간 안에 상대방과의 연결을 보장해야 하기 때문이다. ATM 교환기 시스템은 응용 프로그램들의 실시간 처리 특성을 보장하기 위하여 여러 개의 컴퓨터들로 분산 컴퓨터 시스템을 구성하고 있으며, 교환기 응용 프로그램에서 사용하는 공유 데이터를 주기억장치에 상주시켜 관리하는 주기억장치 데이터베이스 시스템(Main Memory Database System:MDBS)을 내장하고 있다.

DREAM-S (Distributed REAl-time database Management system for Sparc machine)는 ATM 교환기 시스템에 내장된 주기억장치 데이터베이스 시스템이다. DREAM-S는 ATM 교환기의 분산 컴퓨터 시스템에 적합하도록 설계된 분산 데이터베이스 시스템이며, 모든 데이터베이스를 각 프로세서의 주기억 장치에 저장함으로써 교환기 응용 프로그램들의 실시간 데이터 처리 요구를 만족시키고 있다. DREAM-S에서는 대용량의 데이터베이스로부터 원하는 데이터를 신속하게 찾기 위하여 해쉬(hash), 이진 접근(binary access) 방법 등 전통적인 접근 방법과 주기억장치 데이터베이스에서 뛰어난 성능을 제공하는 T-Tree 색인 구조[7, 8]를 지원한다. 그리고, 질의에 대하여 각 프로세서가 신속하게 해당 데이터로 접근할 수 있도록 지원하기 위하여 동일한 데이터를 여러 프로세서의 주기억장치에 중복 저장한 후 중복된 데이터의 일치성을 보장하는 중복 데이터 (replicated data) 관리 기능도 가진다.

DREAM-S는 모든 데이터베이스를 주기억장치에 상주시켜 응용 프로그램의 실시간 데이터 처리의 요구를 만족시키지만 정전 등과 같은 시스템 고장이 발생하면 주기억장치에 저장된 전체 데이터베이스가 파손될 수 있다는 문제를 안고 있다. 따라서, 주기억장치 데이터베이스를 고장 발생 이전의 정상 상태로 신속히 회복시키는 회복 기법(recovery technique)이 필수적이다. 이 때 유의할 점은 데이터의 회복뿐 아니라 신속한 데이터 검색을 위하여 유지되는 색인 구조의 회복이 병행되어야만 색인 구조를 사용하는 응용들의 정상적인 운용이 가능해진다는 점이다. 교환기 시스템에서 대부분의 응용들이 실시간 데이터 처리를 위하여 색인을 이용하므로 이들의 재개를 위하여 색인 구조 회복이 선제되어야 한다. 또한, 색인 구조에 대한 회복 기법 자체가 데이터 회복의 성능에도 영향을 미치므로 전체 회복 시간을 줄이도록 색인 구조(들)를 회복해야 한다.

본 논문에서는 DREAM-S에서 T-Tree 색인 구조의 회복 기법을 제안한다. DREAM-S에서 데이터의 회복에 관한 연구는 참고문헌 [12]에서 제시되었다. 제안된 회복 기법은 기존의 회복 기법과 달리 색인 구조에 대한 회복 방법을 데이터에 대한 회복 방법과 분리하여 별개로 취급하였으며, 다음과 같은 ATM 교환기 시스템의 고유 특성을 감안하여 설계 및 구현되었다. 첫째, 클라이언트-서버(client-server) 구조를 가지는 ATM 교환기 시스템은 서버에만 디스크가 연결되므로 모든 프로세서들이 서버 프로세서를 통하여 디스크 입출력 작업을 수행한다. 따라서 서버 프

로세서의 디스크 입출력 부담을 최소화하도록 회복 기법을 고안해야 한다. 둘째, ATM 교환기에서는 데이터베이스에 할당된 안정된 기억장치(stable memory)가 없으며 각 프로세서에 할당된 주기억 장치의 크기도 한정되어 있다. 따라서 안정된 기억장치와 주기억 장치를 많이 요구하는 회복 기법은 DREAM-S에 적용할 수 없으며, 회복을 위한 로그의 크기도 최소화되어야 한다. 세째, 교환기 시스템에서는 시스템이 초기 및 복구 시 로딩될 때 디스크로부터 제한된 시간 이내에 모든 로딩이 완료되어야 하므로 로딩될 데이터베이스의 크기가 최소화 되어야 한다. 넷째, ATM 교환기 시스템에서 DREAM-S는 주기억장치를 시스템으로부터 추가로 할당받을 수 없다(즉, C언어의 malloc 등의 기능이 제공되지 않는다). 따라서 데이터베이스의 운영 중 요구되는 주기억장치 영역 전체는 시스템의 초기화 시 시스템으로부터 할당되어져야만 한다. 다섯째, 데이터베이스는 off-line에서 모든 릴레이션을 등록 받아 미리 구축된 후 시스템에 실장 되어야 한다. 실시간 처리가 중요한 교환기 시스템의 응용 프로그램들은 미리 고정되어있고 시스템의 운영 중에 성능을 떨어뜨릴 수 있는 schema evolution 등을 요구하지 않으므로 off-line에서 데이터베이스를 미리 구축하는 것은 가능하다.

제안된 회복 기법의 핵심 아이디어는 릴레이션에 대한 회복 기법과 T-Tree 색인 구조에 대한 회복 기법을 구분하여 서로 다르게 처리함으로써 ATM 교환기 시스템의 고유 특성으로 인한 제약 사항들을 극복하는 것이다. 릴레이션에 포함된 튜플 데이터에 대한 회복 기법[12]에서는 튜플 변경에 대한 로그 정보를 각 프로세서의 로그 버퍼에 기록한 후 주기적으로 이를 디스크로 백업하고, 시스템 고장 시 디스크에 저장된 최신 데이터베이스 내용을 이용하여 데이터베이스를 회복시킨다. 반면에, 제안된 T-Tree 색인 구조에 대한 회복 기법은 T-Tree 색인 구조의 변경에 대하여 로그를 작성하는 것이 아니라 디스크로부터 튜플 데이터를 회복한 후에 이를 이용하여 각 클라이언트 주기억장치에 T-Tree 구조를 '재생성'함으로서 T-Tree 구조를 회복한다. 따라서 T-Tree 색인 구조는 디스크에 유지될 필요가 없으며, 각 프로세서의 주기억 장치에만 유지된다. 이 때, T-Tree 구조의 재생성 시간을 가속화하기 위하여 ATM 교환기 시스템의 모든 프로세서들이 릴레이션의 회복 작업과 T-Tree 재생성 작업을 병렬로 처리함으로써 T-Tree 재생성 시간의 오버헤드를 대폭 줄였다. DREAM-S에서는 이러한 T-Tree 회복 기법을 구현하기 위하여 운영체제의 임계 영역(critical section)에 T-Tree 롤백큐 (rollback queue) 라는 자료 구조를 운영하며, 회복된 릴레이션으로부터 각 프로세서의 주기억장치에 T-Tree를 재생성하는 T-Tree 롤백 프로세스(rollback process)를 구현하였다. 논문에서는 제안된 회복 기법의 성능을 수학적인 모델로 분석하였으며, 클라이언트 프로세서의 개수가 증가할수록 제안된 기법은 기존의 기법에 비하여 뛰어난 성능을 제공한다.

MDBS에서 제안된 기존의 회복 기법들은 다음과 같은 특성을 가지므로 ATM 교환기에서 운용되는 DREAM-S에서의 회복 기법으로는 적합하지 않다. 첫째, 기존의 MDBS에서는 색인 구조에 대한 변경 정보를 릴레이션 내의 튜플에 대한 변경 정보와 구분하지 않고 함께 로그를 작성하여 디스크에 저장하고 있다. 그러나, DREAM-S에서 이와 같은 방법을 사용하면 서버 프로세서에서 디스크 입출력 연산이 증가하여 백업과 회복의 성능이 크게 저하되므로 응용 프로그램들의 실시간 요구를 만족시킬 수 없게 된다. 교환기 데이터베이스에서는 일반 비즈니스 데이터베이스와 달리 전체 데이터베이스 영역에서 색인 구조의 크기가 차지하는 비중이 훨씬 높기 때문이다. 제 3.2절에서 이 문제를 상세히 다룬다. 둘째, 기존에는 각 MDBS가 자신의 디스크와 충분한 크기의 주기억 장치를 보유하고, 주기적으로 체크 포인팅 한다는 것을 전제로 한 회복 기법들이 제시되고 있다[3, 4, 5, 6, 8]. 이에 비하여 DREAM-S는 각 서브 시스템이 한정된 크기의 주기억 장치만을 사용할 수 있으며[1], 실시간 데이터 처리를 보장해야 하므로 대량의 디스크 출입 연산을 수반하는 체크 포인트 기법을 채택할 수 없다. 세째, 기존의 MDBS들은 로그 작업의 성능 개선을 위하여 로그 버퍼를 안정된 기억장치에 두는 방

[1] 각 프로세서의 주기억 장치는 OS와 DBMS 등 시스템 소프트웨어 영역, 메타 데이터베이스 영역, 릴레이션 영역, T-Tree 영역, 로그 버퍼 영역으로 구분되어 그 크기가 미리 정해져 있음.

식, 로그 및 체크 포인팅을 위한 특별한 하드웨어를 사용하는 방식, 안정된 기억장치나 특별한 하드웨어 없이 그림자(shadow) 페이지와 로그를 이용한 방식 등이 제시되고 있다[3, 8, 10]. 그러나, DREAM-S에서는 ATM 교환기의 구조적 제약으로 인하여 안정된 기억장치와 로그를 위한 특별한 하드웨어 및 CPU를 추가로 사용할 수가 없으며, 그림자 페이지 기법도 주기억장치의 오버헤드와 체크 포인팅으로 인한 디스크 입출력 오버헤드 때문에 채택할 수 없다[13].

본 논문의 구조는 다음과 같다. 제 2장에서는 관련 연구로서 ATM 교환기의 구조와 특징을 설명하고, DREAM-S에서 회복 기법을 설계하기 위하여 유의해야 할 점들을 분석한다. 제 3장에서는 DREAM-S에서 T-Tree 색인 구조의 설계 및 구현에 관하여 설명한 후에, T-Tree 색인 구조의 회복 기법을 제안하고, 그 성능과 장점을 논의한다. 제 4장에서는 결론을 내리고 향후 연구 방향을 기술한다.

Ⅱ. ATM 교환기 시스템의 구조와 회복 기법의 설계에서 유의할 점

본 장에서는 DREAM-S가 운용될 하부 시스템인 ATM 교환기 시스템의 구조를 설명하고, DREAM-S에서 회복 기법을 설계할 때 ATM 교환기의 구조적인 특성으로 인한 영향을 분석하여 반영하는 방안들을 살펴본다.

ATM 교환기는 클라이언트-서버(client-server) 구조를 갖는 분산 컴퓨터 시스템으로[9] 여러 응용에서 공유하는 데이터를 관리하기 위하여 DREAM-S를 내장하고 있다. 그림 1은 ATM 교환기에서 DREAM-S를 포함하는 서브 시스템들의 구조를 보여준다. 그림 1의 상단에 위치한 ACS (ATM Central Switching Subsystem)는 ATM 교환기의 서버 시스템으로서 ALS (ATM Local Switching Subsystem)라 불리는 다수의 클라이언트 시스템과 연결되어 있다. 서버 시스템인 ACS는 OMP(Operation and Maintenance Processor) 프로세서와 로컬 메모리로 구성되며, 디스크가 연결되어 있다. ACS에 연결된 디스크에는 각 프로세서의 정상적인 동작에 필요한 운영체제(Scalable Real-Time Operating System : SROS)[14], DREAM-S 등과 같은 시스템 소프트웨어와 다양한 교환기 응용 프로그램, 응용 프로그램에서 사용하는 공유 데이터 (시스템 카탈로그 포함) 등이 저장되어 있으며, 이들은 시스템을 초기화하거나 고장 후 회복할 때 디스크로부터 각 프로세서의 로컬 메모리로 로딩된다. 클라이언트 시스템인 각 ALS는 CCCP(Call and Connection Control Processor) 프로세서와 로컬 메모리를 가지며 디스크는 소유하지 않는다. DREAM-S는 그림 1의 교환기 시스템에서 ACS 뿐 아니라 각 ALS에 적재되며, 교환기 시스템의 운용에 필요한 분산 데이터베이스를 관리한다. 서버(ACS)와 각 클라이언트 (ALS)의 주기억 장치에 위치하는 로그 버퍼는 DREAM-S의 운용 중에 발생한 데이터베이스 변경 정보를 디스크에 반영하기 위하여 로그 레코드를 보관하는 임시 장소이다.

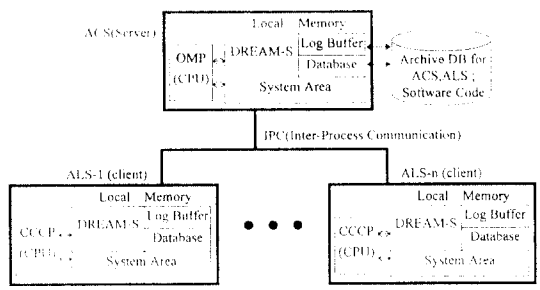


그림 1. ATM 교환기 시스템의 구조

그림 1의 ATM 교환기 시스템은 디스크가 서버 시스템인 ACS에만 장착되어 있는 클라이언트/서버 시스템으로서 다음과 같이 동작하게 된다.

- 시스템의 초기화 시에 서버 프로세서는 디스크의 내용을 각 프로세서의 주기억 장치로 다음과 같은 순서로 로딩한다. 먼저, 서버 프로세서인 OMP가 ACS 동작에 필요한 시스템 소프트웨어, DREAM-S, 데이터베이스, 응용 프로그램들을 디스크로부터 ACS의 주기억장치에 로딩한 후 이들을 동작시킨다. 다음으로, OMP는 클라이언트 시스템인 ALS가 동작하는데 필요한 시스템 소프트웨어, DREAM-S, 데이터베이스, 응용 프로그램들을 디스크로부터 ALS의 로컬 메모리로 로딩한다. OMP가 첫 ALS로 모든 내용을 로딩한 후, 다음 ALS로 디스크 내용을 로딩을 하는 동안에 이미 로딩이 완료된 ALS의

CCCP는 로딩된 자신의 소프트웨어들을 동작시킨다. 이러한 과정을 모든 ALS에 대하여 반복한다. 시스템 고장의 발생시에도 초기 로딩과 마찬가지로 OMP를 통하여 디스크로부터 전체 시스템을 로딩함으로써 각 서브 시스템을 재구동 시키며, 데이터베이스 내용도 회복하게 된다.

- 교환기 시스템의 운용 중 클라이언트 시스템의 주기억장치 데이터베이스에 대하여 발생한 변경과 이에 대한 로그(log) 정보는 OMP를 통하여 디스크 데이터베이스에 반영된다.

이러한 특징을 가지는 ATM 교환기 시스템에서 DREAM-S의 회복 기법을 설계할 때 유의해야 할 사항들은 다음과 같이 요약된다. 첫째, ATM 교환기에서 디스크로부터의 데이터 로딩이나 변경 정보를 디스크에 반영하는 작업이 모두 서버 프로세서(OMP)에 의하여 처리되므로 디스크 입출력을 최소화하여 OMP의 부하를 줄이는 것이 성능 향상을 위하여 중요한 과제이다. 즉, 주기억장치 연산에 비하여 오버헤드가 큰 디스크 입출력 연산을 줄이는 것 자체로 성능을 향상할 수 있으며, 서버 프로세서 (OMP)의 과부하로 인한 성능 저하도 방지할 수 있다. 특히, ATM 교환기에서 클라이언트 시스템인 ALS의 개수가 많아질수록 하나의 서버(OMP)에 요구되는 로딩 및 백업의 부하가 비례하여 증가하게 되므로 OMP의 부하가 심각해질 수 있다. OMP에 의한 디스크 입출력을 최소화하기 위한 방안으로는 다음 세가지를 들 수 있다.

(1) 회복 기법 중에서 체크 포인트와 같이 주기억장치의 현재 데이터베이스 상태를 디스크로 복사함으로써 대량의 디스크 입출력을 야기하는 방식은 실시간 데이터 처리를 보장해야 하는 ATM 교환기에서는 적합하지 않다.

(2) 각 클라이언트 데이터베이스가 변경될 때 발생하는 로그는 OMP에 의하여 디스크에 기록되므로 로그의 크기를 최소화하는 회복 기법을 고안해야 한다. 특히, 클라이언트 시스템(ALS)의 개수가 많아지면 적은 양의 로그이더라도 OMP에 의한 디스크 입출력이 시스템 성능에 있어 병목 현상을 초래할 수 있다.

(3) 시스템 회복 시(혹은 초기화 시)에 서버의 디스크로부터 각 사이트의 주기억장치로 로딩될 데이터의 양을 최소화해야 한다. 회복 시에 디스크로부터 각 프로세서의 주기억장치로 로딩되는 데이터 (프로그램과 데이터베이스)는 OMP를 통하여 모든 ALS에 순차적으로 전송이 이루어지므로 로딩될 데이터의 양을 줄이는 만큼 회복 시간은 빨라진다.

둘째, 그림 1에서와 같이 각 프로세서의 주기억 장치에는 시스템 프로그램의 영역, DREAM-S 영역, 데이터베이스(카탈로그와 인덱스 구조 포함) 영역, 로그를 위한 로그 버퍼 영역만 할당되므로 데이터베이스 크기만큼의 영역을 추가로 요구하는 그림자(shadow) 영역을 이용한 데이터베이스의 회복 기법을 구현하는 것은 불가능하다. 따라서, DREAM-S의 회복 기법에는 응용 프로그램으로부터의 데이터베이스 갱신 요구에 대하여 바로 데이터베이스를 갱신하는 in-place 갱신 방식[6, 11]이 적합하다.

Ⅲ. DREAM-S에서 T-Tree 색인 구조의 회복 기법

본 장에서는 DREAM-S에서 T-Tree 색인 구조의 설계 및 구현에 관하여 설명한 다음에 제안된 T-Tree 색인 구조의 회복 기법을 상세히 기술한다.

제안된 회복 기법은 제 2장에서 분석한 고려 사항들을 충분히 반영하기 위하여 T-Tree 색인 구조를 메모리 상에서만 유지하고, 주기억장치의 데이터베이스가 파손된 경우 릴레이션 데이터를 각 프로세서의 로컬 메모리로 로딩함으로써 릴레이션을 회복시킨 다음, 회복된 릴레이션 정보를 사용하여 각 프로세서에서 T-Tree(들)를 재생성하는 방식이다. 회복 시 T-Tree의 신속한 재생성만 보장된다면 제안된 방식에서는 T-Tree 색인 구조의 갱신에 대한 로그 정보를 디스크에 반영하는 작업과 T-Tree를 디스크로부터 로딩하는 작업이 불필요해지므로 서버 시스템의 디스크 입출력 부담을 줄여 신속한 회복을 가능하게 된다. 특히, 논문에서는 신속한 T-Tree의 재생성을 위하여 모든 프로세서가 릴레이션의 회복 작업과 T-Tree 재생성 작업을 병렬 수행하도록 한다. 이렇게 함으로서 클라이언트 프로세서가 많아질수록 제안된 회복 기법은 기존의 회복 기법보다 뛰어난 성능을 가지게 된다.

본 장의 구성은 다음과 같다. 제 3.1절에서는 DREAM-S에서 구현한 T-Tree 색인 구조와 그 특징을 설명한다. 제 3.2절에서는 디스크와 각 프로세서의 주기억장치에 저장된 데이터베이스 형태를 비교하여 설명한 다음에 DREAM-S에서 제안된 T-Tree 색인 구조의 회복 기법의 자료구조와 알고리즘을 상세히 설명한다. 제 3.3절에서는 제안된 회복 기법의 특성 및 장점을 제시한다.

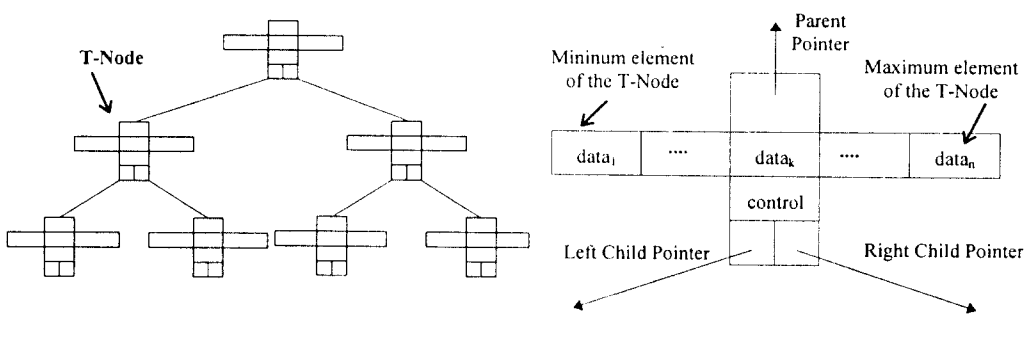
3.1 T-Tree 색인 구조의 설계 및 구현

T-tree 색인 구조는 AVL(Advanced Valanced Level) Tree[1]와 B-Tree[2]로부터 메모리 상주 데이터베이스에 적합하도록 개량한 것이다. DREAM-S에서 구현된 T-Tree 색인 구조는 Lehman[7, 8]에서 제안한 T-Tree 구조와 색인 관리 알고리즘을 기반으로ATM 교환기 시스템의 특성을 반영하여 구현하였다. 그림 2 (a)는 T-Tree의 구조를 보여준다. T-Tree는 T-Node라 불리는 노드들이 이진 트리를 형성하며, 각 T-Node에는 여러 개의 원소(element)가 저장된다. 그림 2 (b)는 T-Tree에 포함된 각 T-Node의 구조를 보여준다. T-Node에는 n개의 데이터 필드, 부모 노드와 왼쪽/오른쪽 자식 노드를 가리키는 세 포인터 필드, 제어 필드로 구성되는 세 필드가 존재한다. 제어 필드에는 T-Tree의 균형을 이루기 위한 깊이 정보와 노드에 저장된 키의 개수가 저장된다.

T-Tree는 이진 트리와 B-Tree의 장점을 모두 가지며, 주기억 장치 데이터베이스에서 효율적이다. 즉, T-Tree는 이진 트리이므로 AVL Tree의 이진 검색 특성을 가지며, B-Tree와 마찬가지로 한 노드에 여러 개의 원소를 포함하고 균형을 이루므로 주기억 장치의 저장 효율이 높으면서 B-tree의 장점인 수정(update) 연산을 용이하게 한다. T-Tree에서 원소의 삽입과 삭제 시 균형을 이루기 위한 rotation은 AVL Tree에서보다 적게 발생하며, 한 T-Node 내의 원소들은 오름차순으로 정렬되어 유지된다[7, 8]. 임의의 T-Node A에서 왼쪽 자식 포인터가 가리키는 T-Node의 원소들은 A노드의 최소 원소보다도 작으며, 오른쪽 자식 노드 포인터가 가리키는 T-Node의 원소들은 A노드의 최대 원소보다도 크다[7, 8].

DREAM-S에서 구현한 T-Tree 색인 구조는 Lehman [7, 8]에서 제안한 T-Tree구조와 마찬가지로 빠른 접근 속도를 제공하면서 ATM 교환기의 특성에 맞추어 각 프로세서의 주기억장치를 최소로 사용하도록 설계 구현되었다. 다음은 DREAM-S에서 구현된 T-Tree와 Lehman[7, 8]에서 제안된 T-Tree 색인 구조의 차이점을 요약한 것이다.

첫째, 각 프로세서에 할당된 한정된 주기억 장치를 효율적으로 사용하기 위하여 내부(internal) T-Node에서 키를 수용하는데 사용되는 데이터 필드의 개수와 실제 노드에 저장된 키의 개수의 비율이 1이 되도록 구현하였으며, 말단(leaf) T-노드에서는 그 비율이 1이하가 되도록 하였다. 이렇게 하는 이유는 내부 T-노드에서는 낭비되는 데이터 필드가 없으므로 주기억장치의 효율성을 최대로 높일 수 있기 때문이다. T-Tree 색인 구조를 이와 같이 구현함으로써 내부 T-노드에 키가 삽입 또는 삭제될 때 T-Tree의 균형을



(a) T-Tree 구조 (b) T-Node 구조

그림 2. T-Tree와 T-Node의 구조.

유지하기 위하여 T-Tree의 노드들을 rotate하는 작업이 빈번하게 발생할 수 있으나 이러한 작업은 디스크 출입 연산 없이 주기억장치에서만 발생하므로 신속히 처리할 수 있다.

둘째, 각 프로세서의 주기억장치에서 T-Tree 색인 구조가 차지하는 영역이 최소가 되도록 유도한다. ATM 교환기용 응용 프로그램에서 사용하는 릴레이션들은 데이터베이스 구축 이전에 각 릴레이션의 튜플수 (즉, 키의 수)와 각 튜플의 크기가 미리 결정되므로 각 T-Node에 수용되는 키의 개수를 적절하게 조정함으로써 T-Tree 색인 구조가 차지하는 주기억장치 영역을 최소화 할 수 있다. T-Tree 색인 구조의 크기는 릴레이션에 포함된 튜플의 개수와 각 T-Node에 수용될 키의 개수에 의하여 결정되기 때문이다. DREAM-S에서는 T-Node가 수용하는 키의 개수를 특정값 (5 또는 10 등과 같이 DREAM-S가 제한적으로 가하는 수치임) 이하로 정한 다음에 그 범위 내에서 T-Tree 색인 구조의 영역이 최소가 되도록 T-Node에 수용되는 키의 개수를 결정한다. 예를 들어, 각 노드에 포함될 키의 개수를 10 이내로 제한하고 DREAM-S에서 키의 개수를 결정하는 경우, 시뮬레이션에 의하여 키의 수가 1, 2,...,10 일 때 T-Tree 색인 구조 영역의 크기를 미리 계산한 후 가장 적은 영역을 필요로 하는 키의 개수를 찾아낸다.

3.2 DREAM-S에서의 회복 기법

본 절에서는 DREAM-S에서 각 프로세서의 주기억장치에 저장된 데이터베이스 구조를 디스크에 저장된 데이터베이스와 비교하여 설명한 다음에 DREAM-S의 회복 기법을 상세히 설명하고, 그 성능과 장단점을 분석한다.

3.2.1 데이터베이스 영역의 구성

DREAM-S에서 T-Tree 색인 구조를 디스크에 구축하면 제 2장에서 분석한 바와 같이 시스템 초기화나 회복 시에 디스크로부터 각 프로세서의 주기억 장치로 로딩할 데이터의 양이 많아지며, 데이터베이스 변경시 서버가 색인 구조에 대한 로그를 디스크에 기록하는 오버헤드가 증가하여 심각한 성능 저하가 예상된다. 이러한 주장은 ATM 교환기 시스템의 실제 데이터베이스에서 다음과 같은 간단한 실험으로 뒷받침된다.

교환기 데이터베이스에서 릴레이션에 포함된 튜플의 평균 크기는 24 바이트 정도로 매우 작다. 이는 교환기 데이터베이스가 주로 시스템 동작에 필요한 공유 데이터 만을 데이터베이스로 구축하기 때문이다. 그런데 튜플의 크기가 작은 릴레이션에 대하여 T-Tree 인덱스를 구축하면 데이터베이스가 차지하는 전체 영역에서 T-Tree 색인 구조가 차지하는 영역의 비율이 상당히 높게 된다. 실제로 ATM 교환기 시스템에서 사용하는 각 릴레이션들에 대하여 하나씩의 T-Tree 색인 구조를 구성한 후 릴레이션들이 차지하는 주기억장치의 크기와 비교한 실험에서 전체 데이터베이스 영역의 30%, 릴레이션 내 튜플 데이터의 50% 정도를 T-Tree 색인 구조가 차지한다는 결과를 얻었다. 따라서 교환기 시스템을 초기화하거나 고장 후 회복할 때 디스크로부터 각 프로세서의 주기억 장치로 로딩해야 할 데이터의 양이 T-Tree 색인 구조를 로딩하지 않는 경우보다 T-Tree 색인 구조로 인하여 42.8% 정도 증가하게 되어 디스크 입출력으로 인한 성능 저하가 예상된다.

다음으로 각 프로세서의 데이터베이스에서 튜플 데이터의 삽입 및 삭제가 발생할 경우 튜플 데이터에 대한 로그 정보와 함께 T-Tree 구조의 변경에 대한 로그 정보도 발생한다. 실제로 ATM 교환기 데이터베이스에서 실험한 결과 릴레이션에 포함된 튜플의 크기가 작으므로 튜플 데이터에 자체에 대한 로그 정보의 크기보다 색인 구조의 변경으로 인한 로그 정보의 크기가 더 많아지기까지 한다. 따라서 T-Tree 색인 구조를 디스크에 유지하게 되면 그와 관련된 로그 정보를 디스크에 반영하기 위한 디스크 입출력의 증가로 인하여 ATM 교환기 시스템의 성능이 저하되게 된다.

이와 같은 두 가지 문제를 해결하기 위하여 DREAM-S에서는 디스크에 저장되는 데이터베이스의 구조와 각 프로세서에 실장되는 주기억장치 내의 데이터베이스 구조를 서로 다르게 구축한다. 그림3 (a)는 주기억장치에 실장되는 데이터베이스의 구조이며, 그림 3 (b)는 디스크에 저장되는 데이터베이스의 구조이다. 각 프로세서의 주기억장치에는 시스템 카탈로그 영역, 릴레이션 영역, T-Tree 구조의 영역으로 구성되지만 디스크에 저장되는 데이터베이스에는 시스템 카탈로그 영역과 릴레이션을 위한 영역 만을 포

함한다. 그림 3과 같이 디스크와 주기억장치의 데이터베이스를 구성함으로써 시스템을 초기화 할 때와 회복 할 때 T-Tree 구조를 제외한 부분만 로딩되며, 데이터베이스 변경 시에도 T-Tree 구조에 대한 로그 정보를 디스크에 기록할 필요가 없게 된다.

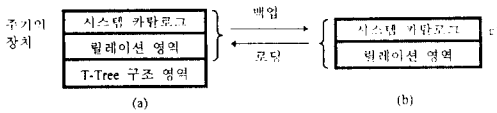


그림 3. 데이터베이스 영역의 구성 : (a) 각 프로세서의 주기억장치에 실장되는 데이터베이스 영역 (b) 디스크에 저장되는 데이터베이스의 영역

3.2.2 회복 기법

DREAM-S에서 T-Tree에 대한 회복 기법은 다음 네 가지로 요약될 수 있다. 첫째, DREAM-S에서는 사용자의 갱신 요구에 대하여 주기억장치상의 T-Tree 색인 구조를 바로 변경하는 in-place갱신 방법[6, 11]을 사용한다. 그 이유는 제 2절에서 설명하였듯이 ATM 교환기 시스템의 각 프로세서에 부착된 데이터베이스 적재 공간이 한정되어 있어 그림자 영역의 사용이 불가능하기 때문이다. 둘째, 시스템을 초기화하거나 회복할 때 디스크로부터 로딩 직후의 주기억장치 데이터베이스는 T-Tree 구조를 갖지 않으므로 로딩 직후에 시스템 카탈로그 및 릴레이션 영역을 검색하여 T-Tree 구조들을 재생성하는 작업이 필요하다. 특히, 이 작업은 전체 회복 시간의 상당한 부분을 차지할 수 있으므로 효율적인 재생성 작업이 이루어져야 한다. 제안된 회복 기법에서는 병렬 처리 방식으로 이 작업을 실행함으로써 전체 회복 시간에서 T-Tree 재생성 시간이 차지하는 비중을 줄였다. 세째, 디스크에 T-Tree를 유지하지 않음으로써 각 프로세서가 주기억장치의 데이터베이스를 갱신하는 경우에도 T-Tree에 대한 로그 정보를 디스크에 저장할 필요가 없다. 넷째, T-Tree 구조에 대한 로그를 작성하지 않고 in-place 방식의 변경을 실행하므로 트랜잭션의 abort시 각 서브 시스템에서 T-Tree 구조의 롤백을 위한 특별한 작업이 필요하다. 다음에는 이러한 특징을 가지는 DREAM-S의 회복 기법을 상세히 설명한다.

(1) 시스템의 고장 시 데이터베이스 회복 기법

본 절에서는 정전 등의 이유로 시스템의 고장이 발생한 경우에 파손된 주기억장치의 데이터베이스를 복구하는 방법을 구체적으로 제시한다. 제시된 방법은 시스템을 초기화하는 경우에도 그대로 적용된다.

시스템을 파손으로부터 회복하거나 초기화할 때 서버 시스템은 디스크로부터 시스템 소프트웨어와 로컬 데이터베이스(릴레이션과 시스템 카탈로그 포함)를 각 서브 시스템의 주기억장치에 로딩한다. 로딩 직후에는 주기억장치에 T-Tree 색인 구조가 생성될 영역이 할당만 이루어진 상태이며 실제 T-Tree 구조의 내용은 존재하지 않는다. 운영체제가 로딩된 DREAM-S를 수행시키면 DREAM-S는 시스템 카탈로그에 등록된 T-Tree(들)에 대하여 릴레이션의 튜플 데이터를 이용하여 T-Tree 색인 구조를 주기억장치에 생성한다. T-Tree 색인 구조의 생성은 릴레이션으로부터 T-Tree 인덱스의 키 값을 읽어 quick-sort를 이용하여 정렬한 후, 일괄적으로 T-Node에 기록함으로써 이루어진다.

제안된 회복 기법에서는 서버가 각 프로세서로 시스템 카탈로그와 릴레이션을 회복(로딩)하는 작업과 각 프로세서가 자신의 회복된 릴레이션을 바탕으로 T-Tree를 생성하는 작업을 병렬로 실행함으로서 T-Tree를 재생성하는데 소요되는 전체 시간을 줄이도록 한다. 그림 4는 DREAM-S에서 T-Tree 재생성 방법을 보여주고 있다. 그림에서 x 축은 회복 시간을, y 축은 프로세서를 나타낸다. 서버 프로세서 P0가 디스크로부터 자신의 시스템 소프트웨어와 로컬 데이터베이스(LD0이라고 부름)를 로딩한 후에 자신(P0)의 T-Tree(들)를 생성한다. 그림 4의 그래프에서 왼쪽 두 화살표가 이 작업에 소요되는 시간을 나타내고 있다. 다음으로 P0가 클라이언트 시스템 P1의 주기억장치에 P1의 운용에 필요한 시스템 소프트웨어와 로컬 데이터베이스 (LD1이라고 부름)를 로딩한 다음 클라이언트 프로세서 P2에 LD2를 로딩하는 동안에 클라이언트 프로세서 P1은 LD1을 이용하여 자신의 T-Tree 구조(들)를 재생성한다. 따라서 P1이 자신의 T-Tree를 생성하는 시간은 P0가 P2에 로컬 데이터 LD2를 로딩하는 시간과 동시에 처리되므로 별도의 시간이 소요되지 않는다. 주의할 점은 P1의 T-Tree 생성 시간은 메모리 작업이므로 디스크 작업인 P0의 LD2 로딩 시

간보다 일반적으로 적으며, 클라이언트 P1이 사용자 프로세스를 서비스하기 전이므로 T-Tree 생성으로 인한 P1의 부하도 무시할 수 있을 정도이다. 마찬가지로 서버 프로세서 P0가 프로세서 P3에 LD3를 로딩하는 동안에 클라이어언트 프로세서 P2는 LD2를 사용하여 자신의 T-Tree 구조(들)를 재생성한다. 모든 클라이언트 프로세서에 대하여 이러한 방식의 T-Tree 재생성 방법을 실행한다. 따라서 그림 4에서 점선 내부의 작업(P1, P2,..., Pn-1에서의 T-Tree 생성 작업)들은 P0가 각 프로세서의 주기억 장치에 LDi를 로딩하는 작업과 병렬처리되므로 추가적인 시간 소모 없이 실행됨을 알 수 있다.

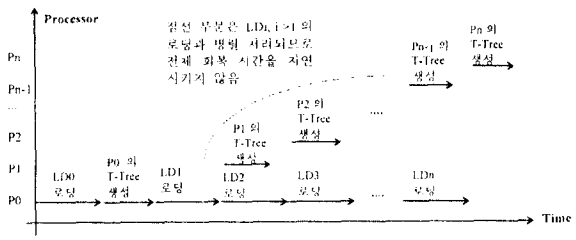


그림 4. 병렬 처리 방식의 T-Tree 재생성 과정.

다음에는 ATM 교환기 시스템에서 T-Tree 재생성에 소요되는 회복 시간을 수학적인 모델로 분석한다. 디스크로부터 각 프로세서의 주기억 장치에 로컬 데이터를 로딩하는데 필요한 요소 시간들을 다음과 같이 세분하여 정의한다.

- $T\_LD_i (i \geq 0)$: 서버가 프로세서 Pi의 주기억 장치에 시스템 소프트웨어 및 데이터베이스(릴레이션과 시스템 카탈로그)를 디스크로부터 로딩(회복)하는 시간
- $T\_TLD_i (i \geq 0)$: 프로세서 Pi의 주기억 장치에 디스크로부터 T-Tree를 로딩하는 시간 (이 경우는 디스크에 T-Tree 색인 구조를 유지하는 방식임)
- $T\_TConst_i (i \geq 0)$: 프로세서 Pi의 주기억 장치에 시스템 카탈로그와 릴레이션 정보를 사용하여 T-Tree를 재생성하는 시간

이와 같은 정의를 이용하여 T-Tree 구조를 디스크에 유지하는 기존의 방식에서 회복 시간은 각 프로세서의 주기억장치에 시스템 소프트웨어, 데이터베이스(릴레이션들과 시스템 카탈로그), T-Tree 색인 구조들을 로딩하는 시간의 합이므로 다음과 같이 표시된다.

$$\sum_{i=0}^{n} (T\_LD_i + T\_TLD_i) \quad (1)$$

반면에, 제안된 방식에서의 T-Tree 구조를 각 프로세서의 주기억 장치에만 유지하고 회복시에 병렬 처리 방식으로 각 프로세서에 회복(재생성)하므로 그림 4에서 표시된 바와 같이 전체 시스템의 회복 시간은 다음의 수식으로 표시된다.

$$\sum_{i=0}^{n} (T\_LD_i) + T\_TConst_0 + T\_TConst_n \quad (2)$$

주의할 점은 수식(1)에서 T_TLD_i는 디스크로부터 T-Tree 구조를 로딩하는 작업이고, 수식 (2)에서 T_TCont_i는 순수한 주기억장치 정보에 대한 연산이므로 수식 (2)의 비용은 수식 (1)의 비용보다 적다. 특히, 제안된 회복 기법에서는 클라이언트 프로세서의 개수가 증가함에 따라 병렬 처리의 효과로 인하여 기존의 기법보다 더욱 뛰어난 성능을 가지게 된다. 이는 수식 (1)과 (2)를 비교함으로써 입증될 수 있다. 수식 (2)에서는 클라이언트 프로세서가 하나 증가함에 따라서 T_LD_i 만큼의 시간이 증가되지만 수식 (1)에서는 $(T\_LD_i + T\_TLD_i)$ 만큼의 회복시간이 증가되기 때문이다.

한편, 정전 등의 이유로 주기억 장치 데이터베이스가 파손된 경우에도 디스크로부터 시스템 카탈로그와 릴레이션을 각 프로세서의 주기억장치로 로딩하여 복구한 다음 각 프로세서가 자신의 T-Tree(들)을 병렬 처리 방식으로 재생성한다.

(2) DREAM-S에서 트랜잭션 수행 절차

DREAM-S의 트랜잭션 수행 절차는 다음과 같다. 트랜잭션이 시작되면 데이터베이스 변경 (릴레이션 내 튜플에 대한 변경임)에 대하여 redo 로그와 undo 로그를 작성하면서 명령어들을 수행한다. 트랜잭션이 commit되면 작성된 로그를 서버 프로세서로 전송하고 회답을 받은 후 수행 트랜잭션을 종료한다. 트

랜잭션이 abort되면 작성된 undo 로그를 이용하여 시스템 카탈로그와 릴레이션 영역에 대하여 롤백을 수행한다[12]. 릴레이션이 복구되면 릴레이션의 식별자를 특정 프로세서(본 논문에서는 T-Tree 롤백 프로세스 라 명명함)에 전하여 그 프로세서로 하여금 복구된 릴레이션에 대하여 T-Tree 구조를 초기 로딩할 때와 마찬가지의 방법으로 재구성하도록 알린다. 즉, 제안된 회복 기법에서는 T-Tree 구조에 대하여는 로그를 작성하여 회복하는 대신에 일단 릴레이션 내 튜플 변경에 대한 로그 정보만을 이용하여 릴레이션 영역을 복구시킨 다음 복구된 릴레이션을 이용하여 T-Tree를 재생성하는 방식을 취한다. 마지막으로 트랜잭션을 수행하는 프로세스는 그 트랜잭션을 종료한다.

제안된 방식에서는 T-Tree 변경에 대한 로그가 생성되지 않으므로 서버 시스템에 의한 디스크 연산이 대폭 줄어든다. 제 3. 2. 1 절에서 기술한 실험 결과에 따르면 교환기 데이터베이스에서는 T-Tree 변경으로 인한 로그의 양이 튜플에 대한 로그의 양보다 더 큰 경우도 있기 때문이다.

(3)서버 프로세서의 로그 처리 방법

서버 프로세서는 로그를 전달받아 디스크 데이터베이스에 바로 적용하여 새로운 값으로 갱신한다. 로그의 내용을 바로 디스크에 반영하는 이유는 디스크 내에 데이터베이스 외에 로그를 추가로 저장할 경우 각 프로세서의 주기억 장치 데이터베이스를 회복 할 때 로그까지 추가로 전송해야 하기 때문이다. 또한, 체크 포인팅을 채택하지 않으므로 초기 제작된 디스크 데이터베이스로부터 데이터를 갱신한 내용이 모두 로그로 존재하게 되면 한정된 디스크의 용량을 초과하게 되기 때문이다.

(4)트랜잭션을 abort할 때 T-Tree 롤백 방법

DREAM-S에서 T-Tree 롤백 프로세스는 디스크로부터 시스템 카탈로그와 릴레이션을 로딩한 후에 생성되는 특별한 프로세스로 자신이 속한 서브 시스템 내에서 트랜잭션 abort가 발생하면 T-Tree 구조를 재생성(회복) 해주는 업무를 담당한다. T-Tree 롤백 프로세스에 의한 T-Tree 구조의 재생성 작업은 시스템을 초기화 시 로딩 직후 DREAM-S가 직접 T-Tree 구조를 생성한 것과 동일한 방법으로 수행된다. 트랜잭션을 수행하는 프로세스(이하 트랜잭션 프로세스라 칭한다)로부터 트랜잭션 abort로 인하여 T-Tree 롤백 프로세스로 T-Tree를 재생성 하도록 정보를 전달하는 것은 T-Tree 롤백 큐 라는 원형 큐 데이터 구조를 사용하여 이루어진다.

그림 5는 트랜잭션 abort를 수행하는 트랜잭션 프로세스와 T-Tree 롤백 프로세스가 T-Tree 롤백 큐를 통하여 롤백되어야 할 릴레이션의 식별자를 주고 받으며 트랜잭션을 롤백시키고 T-Tree를 회복시키는 과정을 보여주고 있다. 그림에서는 왼쪽의 트랜잭션 프로세스와 오른쪽의 T-Tree 롤백 프로세스가 중앙 아래쪽에 표시된 주기억장치 데이터베이스를 회복하는 과정을 점선 화살표로 표시하였다. 중앙 위쪽의 두 실선 화살표는 각각 트랜잭션 프로세스로부터 T-Tree 롤백큐로, T-Tree 롤백 큐로부터 T-Tree 롤백 프로세스로 회복되어야 할 릴레이션 식별자를 전달하는 과정을 보이고 있다. T-Tree 롤백큐는 운영체제가 제공하는 임계 영역에 정의되어 릴레이션 식별자를 저장하는 트랜잭션 프로세스들과 릴레이션 식별자를 가져가는 롤백 프로세스 간의 배타적 접근이 보장된다.

그림 5에서 트랜잭션이 abort되면 트랜잭션 프로세스는 작성된 로그에 의하여 시스템 카탈로그 및 릴레이션 영역에 대한 undo를 수행한 후 T-Tree 색인 구조의 롤백을 위하여 릴레이션 식별자를 T-Tree 롤백 큐에 저장한다. 일단 T-Tree 롤백 큐에 식별자가 저장된 릴레이션들에 대하여는 모든 트랜잭션 프로세스들로부터의 접근이 금지된다. 이는 회복된 릴레이션에 맞도록 색인 구조가 재구성되기 전에는 어떠한 트랜잭션 프로세스도 그 릴레이션에 접근해서는 안되기 때문이다. T-Tree 롤백 프로세스는 트랜잭션 프로세스들과는 독립적으로 수행되며, T-Tree 롤백 큐에 가장 먼저 저장된 릴레이션부터 순차적으로 T-Tree를 재구성한다. DREAM-S가 순차적으로 T-Tree를 재구성하는 이유는 T-Tree 롤백 큐에 저장되어 있는 릴레이션들 중 가장 오랜 시간 동안 접근이 금지되어 있던 릴레이션에 대한 접근을 먼저 허용함으로써 전체 응답 시간을 가능한 줄이도록 하기 위함이다. 특정한 릴레이션에 대하여 T-Tree 구조의 생성이 완료된 후 DREAM-S는 그 릴레이션에 대한 트랜잭션 프로세스들의 접근을 허용한다.

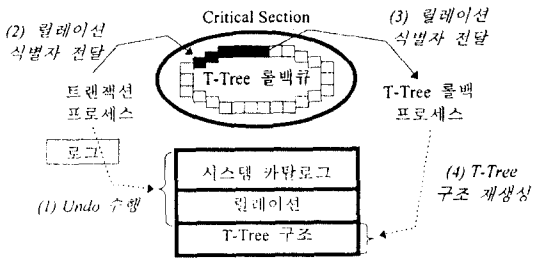


그림 5. 트랜잭션을 abort할 때 롤백 수행 과정.

제안된 T-Tree 회복 기법은 트랜잭션 abort시 T-Tree의 변경된 내용에 대하여 전혀 로그를 작성하지 않고 T-Tree 롤백 프로세스를 이용하여 해당 릴레이션의 T-Tree 구조를 재생성하는 방법이다.

트랜잭션의 수행 시 릴레이션 영역의 변경에 대한 로그(본 절의 이후부터는 '릴레이션 로그'라 한다)와 별도로 T-Tree 구조 영역에 대한 로그(본 절의 이후부터는 'T-Tree 로그'라 한다)를 주기억장치에 유지하다가 트랜잭션이 commit되면 T-Tree 로그를 삭제하고, 트랜잭션이 abort되면 T-Tree 로그에 의하여 T-Tree 구조를 롤백시키는 방법도 있을 수 있다. T-Tree 로그를 이용하여 롤백시키는 방법은 T-Tree 롤백 프로세스를 이용하는 방법보다 T-Tree 구조를 빨리 복구시킬 수 있다. 그러나 ATM 교환기 시스템에서는 다음의 시스템, T-Tree, 그리고 데이터베이스의 특징들로 인하여 T-Tree 로그를 주기억 장치에 유지하지 않는 것이 바람직하다.

- 시스템의 특징 : ATM 교환기에서 운용되는 DREAM-S를 포함한 모든 시스템은 교환기 주기억 장치를 최소한으로 사용해야 한다는 제약 조건을 안고있다.
- T-Tree 색인 구조의 특징 : 제 3.1절에서 주기억장치를 최소로 줄이기 위한 방법에 따라 구현된 T-Tree 색인 구조는 T-노드에 키가 삽입 또는 삭제될 때 T-Tree의 노드들을 rotate하는 작업이 빈번하게 발생함을 설명하였다. 즉, 릴레이션의 튜플이 삽입 또는 삭제될 때 발생하는 T-Tree 로그의 크기를 예측할 수 없고 릴레이션 로그보다 많은 양의 T-Tree 로그가 생길 수 있음을 의미한다.
- 교환기 데이터베이스의 특징 : 제 3.2.1절에서 설명한 바와 같이 ATM 교환기 시스템의 데이터베이스 내 튜플의 크기가 24바이트로 작으므로 튜플 데이터 자체에 대한 로그 정보의 크기보다 색인 구조의 변경으로 인한 로그 정보의 크기가 더 많아지기까지 한다. 따라서 DREAM-S는 많은 양의 T-Tree 로그를 주기억장치에 유지하는 것은 바람직하지 않다.

따라서 ATM 교환기 시스템에서는 주기억장치 내에서 T-Tree 로그를 유지하여 T-Tree 구조 영역을 롤백시키는 회복 기법은 적합하지 않으므로, T-Tree 롤백 프로세스를 이용하여 해당 릴레이션의 T-Tree 구조를 재생성하는 회복 기법이 가장 적합하다.

3.3 T-Tree 회복 기법의 구현

제안된 T-tree 색인 구조의 회복 기법은 다수의 SPARC 프로세서들로 구성되는 분산 컴퓨터 시스템인 ATM 교환기 시스템에서 운용되는 DREAM-S의 서브 모듈로 구현되었다. DREAM-S는 실시간 운영체제인 SROS (Scalable Real-time Operating System) [14] 상에서 운용되고 있으며, C 프로그래밍 언어 및 CHILL 프로그래밍 언어를 사용하여 구현되었다. 현재, DREAM-S는 상용화를 위한 기능 및 성능 시험이 완료되었으며, 실제 교환기 데이터베이스에서 성능은 80 microsecond 이내에 원하는 데이터를 검색할 수 있다. 시스템 고장으로 인하여 데이터베이스가 파손된 후 회복하는 시험에서도 수 분 이내에 데이터베이스 릴레이션과 T-Tree 색인 구조가 완전히 회복되어야 한다는 ATM 교환기의 제약 사항을 만족시키고 있다.

Ⅳ. 결 론

본 논문에서는 ATM 교환기 시스템에서 DREAM-S에 적합한 T-Tree 색인 구조의 회복 기법을 설계하고 구현하였다. ATM 교환기 시스템은 다수의 클라이언트 시스템이 서버 시스템에 연결된 분산 컴퓨터 시스템이며, 디스크가 서버에만 연결되어 있음으로써 서버의 디스크 출입 연산 오버헤드가 크다는 특징을 가진다. 제안된 기법의 핵심 아이디어는 T-Tree 색인 구조를 디스크 영역에 유지하지 않고 각 프로세서의 주기억장치에만 두며, 시스템 고장 시 디스크로부터 릴

레이션을 회복한 후 이를 바탕으로 각 프로세서가 병렬 처리 방식으로 재생성함으로써 서버의 부담을 크게 줄인다는 점이다. 이렇게 함으로서 T-Tree 구조의 변경에 대해서는 로그 정보를 디스크에 기록할 필요가 없어지며, 회복 시에 디스크로부터 각 프로세서의 주기억 장치로 T-Tree 색인 구조를 로딩할 필요가 없어지게 된다. 그 결과 전체 시스템의 디스크 출입 연산을 담당하는 서버 시스템의 부담이 줄어들게 된다. 실제 운용되는 ATM 교환기 데이터베이스에서 T-Tree 색인 구조의 크기는 릴레이션 크기의 50% 정도를 차지할 수 있으며, 튜플 변경에 대한 로그의 양보다 T-Tree 색인 구조의 변경에 대한 로그의 양이 더 커질 수 있으므로 제안된 회복 기법은 서버 시스템에 의한 디스크 입출력 오버헤드는 크게 줄일 수 있다. 또한, 제안된 방식은 T-Tree 재생성 작업을 각 프로세서의 로컬 데이터를 로딩하는 작업과 병렬로 처리함으로써 T-Tree 재생성 작업으로 인하여 추가되는 회복 시간이 거의 없도록 하였다.

본 논문에서 제안된 회복 기법은 MDBS에서 좋은 성능을 나타내는 T-Tree 색인 구조에 대하여 제안되었으나 B-Tree와 같은 다른 색인 구조에 대해서도 적용될 수 있다. 앞으로 ATM 교환기 시스템에서 서버 프로세서의 부담을 줄이기 위한 근본적인 해결책으로 각 프로세스가 대용량의 안정된 주기억장치와 디스크를 장착하는 연구와 이러한 환경에서 적합한 회복 기법을 연구하고자 한다. 이렇게 함으로써 보다 안정된 시스템 성능과 빠른 회복 기법들을 제공할 수 있을 것이다.

참 고 문 헌

1. A. Aho, J. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, 1974.
2. D.Comer, "The Ubiquitous B-Tree," *ACM Computing Surveys*, Vol. 3, No. 2, June, 1979.
3. M. H. Eich, "A Classification and Comparison of Main Memory Database Recovery Techniques," In *Proc. Intl. Conf. on Data Engineering*, pp. 332-339, 1987.
4. M. H. Eich, "Main Memory Database Recovery," *In IEEE Fall Joint Computer Conference*, Vol. 2, No. 6, pp. 1226-1232, 1986.
5. H. Garcia-Molina and K. Salem, "Main Memory Database Systems: An Overview," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 4, No. 6, pp. 509-516, Dec. 1992.
6. H. Jagadish, A. Silberschatz, and S. Sudarshan, "Recovering from Main-Memory Lapses," In *Proc. Intl. Conf. Very Large Data Bases*, pp. 391-404, 1993.
7. T. J. Lehman, "A Study of Index Structures for Main Memory Database Management Systems," In *Proc. 12th Intl. Conf. Very Large Data bases*, pp. 294-303, 1986.
8. T.J. Lehman, Design and Performance Evaluation of a Main Memory Relational Database System, Technical Report 656, University of Wisconsin Madison, Aug. 1986.
9. Y. B. Kim, et al., "An Architecture of Scalable ATM Switching System and Its Call Processing Capacity Estimation," *ETRI Journal*, Vol. 18, No. 3, pp. 107-124, Oct. 1996.
10. 우승균, 이윤준, 김명호, "주기억 장치 데이터베이스 시스템에서의 회복 기법에 대한 고찰," *정보과학회지*, 제 14권, 제 2호, pp. 38-46, 1996년 2월.
11. 차상균, 박상호, 박명대, 이성직, "M²RTSS: 주 메모리 실시간 저장 시스템," *정보과학회지*, 제 14권, 제 2호, pp. 14-23, 1996년 2월.
12. 한미경, 윤용익, "DREAM-S 데이터 백업 시스템에서의 Fault-Tolerant Logging Scheme," *한국정보과학회 추계학술발표회 자료집*, pp. 93-96, 1996년 10월.
13. 황규영, 김상욱, "주기억 장치 데이터베이스 시스템에서의 파손회복기법," *한국정보과학회지* 제 11권 1호, 1993 년 2월.
14. E. H Lee, S. I. Jun, and J. H. Cho, "Operating System Support for Cross Debugging on Switching Applications," *IEEE Intl. Conf. on Communication Systems (ICCS)*, Vol. 1 of 3, Singapore, 1996.

이 승 선(Seung-Sun Lee) 정회원

1992년 2월 : 한국과학기술대학 전산과(공학사)

1994년 2월 : 한국과학기술원 전산학과(공학석사)

1994년 3월~현재 : 한국전자통신연구원 연구원

※주관심분야 : 분산 데이터베이스, 실시간 데이터베이스

조 완 섭(Wan-Sup Cho) 정회원

1985년 2월 : 경북대학교 통계학과(이학사)

1987년 2월 : 한국과학기술원 전산학과(공학석사)

1996년 2월 : 한국과학기술원 전산학과(공학박사)

1987년 1월~1990년 11월 : 한국전자통신연구소 연구원

1996년 2월~1997년 2월 : 한국전자통신연구소 Post-Doc. 연구원

1997년 3월~1997년 8월 : 안동대학교 전자정보산업학부 전임강사

1997년 9월~현재 : 충북대학고 경영정보학과 전임강사

※주관심분야 : 정보통신, 정보검색, 데이타베이스 시스템, 객체지향 등 멀티미디어 시스템

윤 용 익(Yong-Ik Yoon) 정회원

1983년 2월 : 동국대학교 통계학과(공학사)

1985년 2월 : 한국과학기술원 전산학과(공학석사)

1994년 8월 : 한국과학기술원 전산학과(공학박사)

1985년 2월~1997년 9월 : 한국전자통신연구원 책임연구원

1997년 9월~현재 : 숙명여자대학교 전산학과 교수

※주관심분야 : 정보통신, 멀티미디어, 실시간 처리 시스템, 실시간 분산 데이터베이스 시스템, 분산 시스템