産業技術硏究(江原大學校 産業技術硏究所 論文集) 第 15輯 (1996)
Research Report. The Institute of Industrial Technology, Kangwon Nat'l Univ., Korea, Vol. 15, 1996

메인 메모리 상주 데이터베이스 회복 기법

김 상 욱 · 이 헌 길* · 김 용 석**

Recovery Techniques for Memory Resident Databases

Sang-Wook Kim · Heon-Gyil Lee* · and Yong-Seok Kim**

ABSTRACT

Databases can crash due to various failures in computer systems. Recovery is a mechanism for restoring consistent data from damages caused by the failures and is an essential feature in database systems. This paper surveys recovery techniques for memory resident database systems. We first describe the basic architecture for memory resident database systems, and point out the main factors affecting their performance enhancement. Next, we explain the write-ahead logging(WAL), a recovery technique widely-used in most disk resident database systems, for easy understanding of basic recovery mechanisms. And then, we discuss some new concepts employed in memory resident database systems recovery. Finally, we present a representative memory resident database recovery technique, which is based on a special purpose hardware called HALO, as a case study.

1. 서 론

본 논문에서는 메인 메모리 상주 데이타베이스 시스템(memory resident database system) 에서의 파손 회복 기법에 대하여 소개하고자 한다. 회복 관리자(recovery manager)란 데이타베이스 시스템내에 저장된 데이타를 일관성(consistency)있는 상태로 유지해 주는 모듈을 의미하며, 데이타베이스를 총체적으로 관리해주는 데이터베이스 관리 시스템(database management system)이 갖추어야 할 필수적인 기능중의 하나이다[Ver78][Hae83][Wha92]. 회복 기능이 필요한 이유는 컴퓨터 시스템에 여러가지 오류(failure)가 발생되기 때문이다. 오류란 컴퓨터 시스템이 사용자의 명령 이외의 비정상적인 동작을 하는 것으로서, 이러한 동작의 결과 데이타베이스내에 저장된 데이타는 사용자가 전혀 예기치 못하는 일관성이 깨어진 상태로 될 수 있

강원대학교 정보통신공학과 전임강사
* 강원대학교 컴퓨터공학과 조교수
** 강원대학교 컴퓨터공학과 전임강사

다. 데이타베이스가 일관성을 잃게되면 사용자는 예측하지 못한 잘못된 결과를 얻게 되며, 이미 일관성이 상실된 데이타를 기반으로 이 사실을 모르는 다른 사용자가 변경 연산을 시도하게 되므로, 문제는 점점 심화된다.

최근 메인 메모리 기술의 발달로 인하여 가격이 점차 저하됨에 따라 컴퓨터 시스템 내의 메인 메모리 용량은 점점 증가하는 추세에 있다. 이에 따라 데이타베이스 분야에서는 늘어나는 메인 메모리의 용량을 최대한 활용하여 디스크내에 저장된 데이타를 모두 메인 메모리로 상주시켜 데이타베이스 시스템의 성능을 개선시키기 위한 연구가 활발히 진행중에 있다.
[DeW84][Gar84][Leh86][Sha86].

메인 메모리 상주 데이타베이스 시스템이란 전체 데이타베이스를 메인 메모리내에 상주시킨 상태에서 각종 데이타베이스 연산을 수행하는 시스템을 의미한다[Sal89][Kum91]. 이 시스템의 특징은 모든 데이타베이스 연산이 디스크 액세스를 유발시키지 않고 메인 메모리내에서만 수행되므로 사용자에 대한 응답 시간을 단축시킬 수 있을 뿐만 아니라 전체 시스템의 성능을 높일 수 있다는 것이다[DeW84].

메인 메모리 상주 데이타베이스 시스템을 구축하는데 있어서 반드시 고려해야 하는 문제점의 하나는 파손시의 회복이다[Hag86][Sal86][Leh87][Eic87][Kum91]. 데이타베이스 전체가 메인 메모리내에 상주하기 때문에 시스템 파워나 메인 메모리 칩 등에 이상이 발생되는 경우, 데이타베이스 내용 전체가 손실되기 때문이다. 따라서 전체 데이타베이스가 파손되는 경우에도 이를 다시 일관된 상태로 회복시켜 줄 수 있는 기법이 필요하다.

메인 메모리 상주 데이타베이스 시스템에서도 디스크를 기반으로한 데이타베이스 시스템에서 발생되는 트랜잭션 오류, 시스템 오류, 미디어 오류가 발생될 수 있으며, 따라서 이에 대처할 수 있는 회복 기능이 요구된다. 메인 메모리 상주 데이타베이스 시스템에서는 데이타베이스를 저장하는 주요 매체가 메인 메모리이므로 디스크 기반 데이타베이스 시스템에서 사용되는 회복 알고리즘을 그대로 적용할 경우에는 그 효율성에 문제가 있다[Leh87]. 그러므로 메인 메모리 상주 데이타베이스 시스템에서는 메인 메모리의 특성을 충분히 활용할 수 있는 새로운 회복 알고리즘이 필요하다.

본 논문에서는 메인 메모리 상주 데이타베이스에서 사용되는 회복 기법에 대하여 논의한다. 메인 메모리 상주 데이타베이스 시스템에서 발생되는 세가지 오류중에서 트랜잭션 오류, 미디어 오류시의 회복 방법은 디스크 기반 데이타베이스 시스템과 큰 차이점이 없다. 따라서 본 논문에서는 주로 시스템 오류로부터의 회복 기법에 관하여 집중적으로 다루고자 한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 메인 메모리 상주 데이타베이스 시스템의 기본 아키텍춰에 대하여 설명하고, 메인 메모리 상주 데이타베이스 시스템이 디스크 기반 데이타베이스 시스템과 비교하여 뛰어난 성능을 가지는 원인에 대하여 언급한다. 제 3장에서는 회복에 대한 개념을 돕기 위하여 디스크 기반 데이타베이스 시스템에서 현재 가장 널리 사용되는 WAL(write-ahead logging) 기법에 대하여 소개한다. 제 4장에서는 메인 메모리 상주 데이타베이스 시스템에서 사용되는 회복 기법의 새로운 개념들에 대하여 논의한다. 제 5장에서는 사례 연구로서 Prinston 대학의 Salem과 Garcia-Molina가 제안한 기법에 대하여 소개한 후 제 6장에서 결론을 내린다.

2. 메인 메모리 상주 데이타베이스 시스템

본 장에서는 메인 메모리 상주 데이타베이스 시스템에 대하여 소개한다. 먼저 제 2.1절에서는 메인 메모리 상주 데이타베이스 시스템의 기본 아키텍춰와 각 구성 요소에 대하여 간략히 소개하고, 제 2.2절에서는 메인 메모리 상주 데이타베이스 시스템이 디스크 기반 데이타베이스 시스템에 비하여 뛰어난 성능을 나타낼 수 있는 주요 원인에 대하여 언급한다.

2.1 메인 메모리 상주 데이타베이스 시스템의 구성

(그림 1)은 메인 메모리 상주 데이타베이스 시스템의 기본 아키텍춰를 나타낸 것이다.

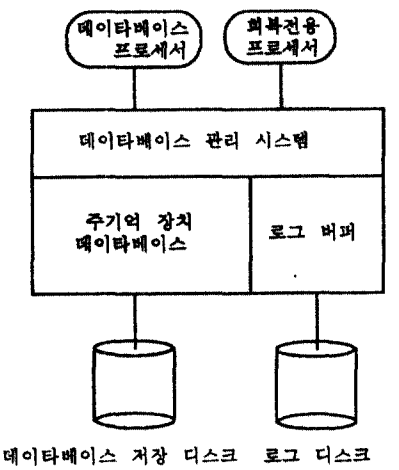

Fig. 1. Basic Architecture of Memory Resident Database Systems.

먼저 메인 메모리내에는 전체 데이타베이스가 상주하게 되고, 이를 관리하기 위한 데이타베이스 관리 시스템, 그리고 로그 레코드를 기록하기 위한 로그 버퍼가 존재한다. 한편, 디스크내의 로그 디스크는 메인 메모리내 로그 버퍼의 내용을 백업받음으로써 로그 버퍼를 재사용할 수 있도록 하기 위한 것이며, 디스크내에 존재하는 데이타베이스는 시스템을 초기화할 때 메인 메모리로 데이타베이스를 로드시키기 위한 것이다. 프로세서(processor)는 일반적으로 모든 데이타베이스 연산을 위하여 하나만으로도 동작이 가능하나 효율적인 회복을 위하여 회복 전용 프로세서를 별도로 둠으로써 회복을 위한 오버헤드를 분담시킬 수 있다.

2.2 메인 메모리 상주 데이타베이스 시스템의 성능 향상의 원인

메인 메모리 상주 데이타베이스 시스템에서 성능 향상의 가장 큰 원인은 디스크 액세스의 제거이다.[DeW84] [Sal86]. 디스크 기반 데이타베이스 시스템에서 응답 시간 지연의 주된 원인은 데이타 검색을 위한 디스크 액세스이다. 반면 메인 메모리 상주 데이타베이스 시스템에서는 데이타베이스 전체가 메인 메모리내에 상주하므로 데이타 검색을 위한 디스크 액세스가 발생되지 않으며, 따라서 상당한 성능의 향상을 가져올 수 있다. 이러한 성능 향상은 기존의 디스크 기반 데이타베이스 관리 시스템을 용량이 큰 메인 메모리를 갖는 컴퓨터 시스템에 옮기는 경우에도 그대로 나타나게 된다. 그 이유는 데이타베이스의 많은 부분이 메인 메모리내의 버퍼에 올라오게 되며, 그 결과 디스크 액세스의 횟수가 감소되기 때문이다.

두번째 성능 향상의 원인은 버퍼링 오버헤드의 제거이다. 디스크 기반 데이타베이스 시스템에서는 튜플이 저장된 페이지가 메인 메모리의 버퍼내에 존재하는 경우에도 해당 버퍼 페이지를 찾기 위하여 해싱과 선형 탐색(linear probing)을 반복해야 한다. 이러한 작업이 매 튜플 액세스마다 반복 수행되어야 하므로 상당한 오버헤드가 된다. 또한 해당 튜플에 대한 데이타베이스 연산을 수행하기 위하여 자신의 국지 영역(local area)으로 튜플을 복제해야 하

는 것도 또 다른 오버헤드가 된다. 그러나 메인 메모리 상주 데이타베이스 시스템에서는 메인 메모리내의 해당 튜플의 위치를 찾아 직접 액세스할 수 있으므로 위에서 언급한 두가지 오버헤드가 제거된다.

세번째 성능 향상의 원인은 메인 메모리에 적합한 색인 구조의 사용이다. 디스크 기반 데이타베이스 시스템에서는 디스크 액세스의 오버헤드가 성능에 큰 영향을 미치므로 평균적으로 디스크 액세스 횟수를 최소화할 수 있는 색인 구조를 사용해야 한다. 이러한 요구 조건에 부합되는 색인 구조로서 균형 트리인 B+ 트리를 주로 사용하고 있으나 B+ 트리에서는 균형을 지속적으로 유지시키기 위한 부가적인 오버헤드가 요구되며, 하나의 디스크 페이지와 대응되는 트리 노드내에는 많은 데이타가 존재하므로 검색시의 비교 연산의 오버헤드가 크다[DeW84].

메인 메모리 상주 데이타베이스 시스템에서는 데이타가 항상 메인 메모리내에 존재하기 때문에 액세스 횟수는 큰 문제가 되지 않는다. 그러므로 트리를 인위적으로 균형화시키는 불필요한 오버헤드를 감수하면서까지 B+ 트리를 사용할 필요가 없으며, 또한 트리 노드의 크기도 디스크 페이지와 동일하게 만들 필요가 없다. 따라서 메인 메모리 상주 데이타베이스 시스템에서는 일반적으로 AVL 트리[Knu73]등의 이진 트리를 사용함으로써 위에서 언급한 B+ 트리의 두가지 오버헤드를 피한다[DeW84].

메인 메모리 상주 데이타베이스 시스템의 마지막 성능향상의 원인은(Locking) 오버헤드의 제거이다. 디스크 기반 데이타베이스 시스템에서는 디스크 액세스로 인하여 트랜잭션 수행 시간이 길어지므로 각 트랜잭션의 응답 시간을 줄이기 위하여 여러 트랜잭션들의 수행이 병행(concurrent)되어야 한다. 그러나 메인 메모리 상주 데이타베이스 시스템에서는 각 트랜잭션의 수행 시간이 상대적으로 대단히 짧으므로 트랜잭션들의 순차적(serial) 수행이 가능하다[Sal86]. 물론 이러한 주장에 대하여 동의하지 않는 연구도 있으나[Eic87], 만일 순차적 수행으로 인하여 동시성 제어가 불필요해지면, 매 튜플 액세스할 때마다 액세스 가능성 여부를 검사하던 잠금의 오버헤드가 제거되므로 이것도 메인 메모리 상주 데이타베이스 시스템의 성능 향상의 한 요소가 된다.

지금까지 메인 메모리 상주 데이타베이스 시스템의 기본 아키텍춰와 성능 향상의 원인에 대하여 살펴보았다. 디스크 액세스의 제거를 비롯하여 버퍼링 오버헤드의 제거, 메인 메모리에 적합한 색인 구조의 사용, 잠금 오버헤드의 제거 등으로 인하여 메인 메모리 상주 데이타베이스 시스템은 디스크 기반 데이타베이스 시스템에 비하여 상당한 성능의 개선을 가져올 수 있음을 알 수 있다.

3. WAL 기법

본 장에서는 회복에 대한 이해를 돕기 위하여 디스크 기반 데이타베이스 시스템에서 시스템 오류로부터 회복하기 위한 기법으로 널리 사용되는 WAL(write-ahead logging) 기법[Moh92]에 대하여 설명한다. 먼저 제 3.1절에서는 WAL 기법의 기본 원리에 대하여 설명하고, 제 3.2절, 제 3.3절, 제 3.4절에서는 회복 기법의 주요 특성으로 나타나는 트랜잭션의 종료, 체크포인트, 그리고 회복을 위한 재시동에 관하여 각각 설명한다.

3.1 기본 원리

WAL 기법은 데이타 버퍼내의 변경된 내용이 디스크에 반영되기 전에, 그 변경

을 나타내는 로그 레코드가 있는 로그 버퍼의 내용이 먼저 디스크에 반영되어야 한다는 원칙을 이용하는 기법이다. 로그 레코드는 트랜잭션의 각 변경 연산이 적용되는 데이타 부분에 대하여 연산 수행전 값과 연산 수행후 값을 내용으로 갖는 자료구조이다. 변경전 값은 회복시에 디스크내에 반영된 내용을 철회시키는 경우에 사용되며, 변경후 값은 디스크에 반영될 내용이 시스템 오류로 인하여 디스크에 반영되지 않았을 경우에 이를 디스크에 반영시키는 재수행을 위하여 사용된다.

로그 버퍼의 내용을 디스크에 먼저 반영시키는 이유는, 오직 하나의 데이타 버전만이 존재하기 때문이다. 즉, 변경된 데이타가 먼저 디스크에 반영되고 로그 레코드는 디스크에 반영되기 전에 시스템 오류가 발생되면, 현재 데이타를 변경 이전의 상태로 철회시켜줄 수 있는 방법이 존재하지 않는다. 이것은 변경전 값을 가진 로그 레코드가 디스크에 존재하지 않기 때문이다. 그러나 로그 레코드가 먼저 디스크에 반영되면, 오류가 발생되더라도 이 로그 레코드내의 변경전과 변경후의 값을 이용하여 철회와 재시동이 가능하므로 일관성있는 상태로의 회복이 가능하다.

3.2 트랜잭션 종료시의 동작

정상적으로 종료된 트랜잭션의 변경 연산은 데이타베이스에 반드시 반영되어야 하므로 회복 관리자는 트랜잭션이 종료될 때 COMMIT 로그 레코드를 로그 화일에 반영시킨다. 이때 로그 버퍼내에 존재하는 내용들은 디스크내의 로그 화일에 강제적으로 반영된다. 그 이유는 로그 레코드의 내용이 버퍼에만 반영되고 디스크에 미처 반영되기 전에 시스템 오류가 발생되는 경우 회복 관리자는 그 트랜잭션이 정상적으로 종료된 사실을 감지할 수 없으며, 따라서 이 결과를 데이타베이스내에 반영시킬 수 없기 때문이다.

트랜잭션이 정상적으로 종료되는 시점은 해당 트랜잭션의 COMMIT 레코드가 디스크내의 로그 화일에 반영되는 시점이 된다. 트랜잭션의 모든 연산이 완료되더라도 COMMIT 레코드가 로그 화일에 반영되기 전에 시스템 오류가 발생된다면, 회복 관리자는 그 트랜잭션이 진행중인 것으로 간주한다. 회복을 위한 재시동시에 진행중인 것으로 간주되는 트랜잭션들은 철회되므로, 모든 연산이 완료되더라도 COMMIT 레코드가 로그 화일에 반영되지 않은 트랜잭션들은 아직 수행중인 트랜잭션과 마찬가지로 취급된다.

3.3 체크포인트시의 동작

시스템 오류가 발생되면 이 시점에서 진행중이던 트랜잭션들은 수행이 완료되지 않았으므로 디스크내에 반영되었던 일부 내용들을 철회시켜야 하며, 수행이 완료되었으나 결과가 디스크에 반영되기 전의 트랜잭션은 재수행 시켜야 한다.

철회작업을 하기 위해서는, 먼저 시스템 오류가 발생된 시점에서 종료되지 않은 트랜잭션들과 시스템 오류 이전에 수행이 완료된 트랜잭션들을 찾아내어야 한다. 로그 화일을 처음부터 시스템 오류 발생 시점까지 탐색하면서 이러한 트랜잭션들을 구분해 낼 수도 있으나, 이러한 방법은 로그 화일의 크기가 상당히 크다는 것을 고려할 때 매우 비효율적이다. 따라서 이러한 비효율성을 극복하기 위하여 체크포인트(checkpoint)의 개념이 도입되었다.

즉, 일정한 주기마다 로그 화일에 체크포인트 레코드를 취하게 되는데, 체크포인트 레코드에는 현재 진행중인 트랜잭션들의 리스트가 기록된다. 따라서 시스템 오류가 발생된 경우에, 로그 화일의 시작부터 탐색하는 것이 아니라, 가장 최근에 기록된 체크포인트 레코드에서부터 회복 작

업을 수행할 수 있으므로 효율적이다.

체크포인트시에는 다음과 같은 동작이 취해진다.

(1) 체크포인트의 시작을 알리는 체크포인트 시작 로그 레코드가 기록된다.

(2) 현재 진행중인 트랜잭션을 조사하여 로그 레코드에 기록한다.

(3) 현재 디스크에 반영되지 않은 변경 연산들 중에서 가장 먼저 수행된 변경 연산에 해당되는 로그 레코드의 위치를 체크포인트 레코드에 기록한다. 이 정보는 오류 발생시에 재수행의 시작 위치를 알려준다.

(4) 체크포인트가 끝남을 알리는 로그 레코드를 기록한다.

(6) 체크포인트가 시작되는 로그 레코드에 대한 포인터를 디스크내의 포인터 변수에 기록한다.

WAL 기법은 체크포인트 작업시 전체 시스템을 정지시키지 않는 퍼지 체크포인트를 사용한다. 이러한 특성으로 수행중인 트랜잭션의 동작이 체크포인트 동작과 병행될 수 있으며, 따라서 트랜잭션 응답 시간의 지연을 방지할 수 있다는 것이 큰 장점이다.

3.4 회복을 위한 재시동시의 동작

시스템 오류가 발생하였을 때, 회복을 위한 재시동은 로그 화일의 내용과 현재까지 수행된 변경 연산이 반영된 디스크내의 데이타베이스 내용을 기반으로 행해진다. 회복을 위한 재시동이 수행되는 단계는 아래와 같으며, (그림 2)는 회복 동작이 수행되는 것을 그림으로 나타낸 것이다.

(1) 체크포인트 동작의 단계 (6)에서 기록한 포인터 변수를 이용하여 가장 최근에 완료된 체크포인트의 시작 위치를 찾는다.

(2)체크포인트 레코드에서 시작하여 시스템 오류가 발생된 지점까지의 로그 화일을 전방향으로 탐색하며, 철회되어야 할 트랜잭션들과 재수행되어야 할 트랜잭션의 리스트를 찾아낸다.

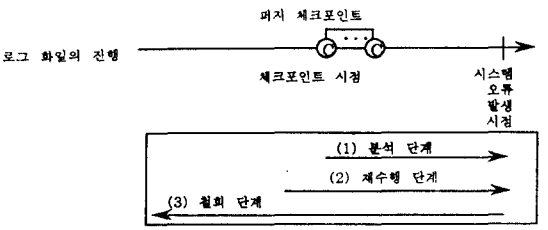

Fig. 2.. Recovery from System Crash Using the WAL Technique.

(3) 체크포인트 동작시 단계 (3)에서 기록한 정보를 이용하여 재수행의 시발점이 되는 로그 레코드의 위치를 찾는다. 이 위치는 변경이 디스크에 반영되지 않았을 가능성이 있는 변경 연산의 로그 레코드들 중에서 가장 먼저 기록된 로그 레코드의 위치이다. 여기서부터 시스템 오류가 발생한 시점까지 전방향으로 탐색하며 모든 변경에 대하여 재수행한다. 이때 재수행 되어야 할 트랜잭션뿐만 아니라 철회되어야 할 트랜잭션들 까지도 함께 재수행된다. 이것을 이력의 재수행(repeating history)이라 부른다. 오류가 발생한 시점까지 모든 로그 레코드의 내용을 재수행시키는 이유는 로그와 데이타베이스의 내용이 일치되는 시점을 만들기 위해서이다. 즉, "데이타베이스 = 로그"라는 관계를 정립하기 위한 것이다.

(4) 시스템 오류가 발생한 시점에서부터 시작하여 후방향으로 탐색하며 철회 리스트에 있는 트랜잭션의 변경을 철회시킨다.

(5) 체크포인트를 취한다.

시스템 오류로부터 회복 동작을 수행하는 과정에서 다시 시스템 오류가 발생되면, 최근의 체크포인트 레코드를 이용하여 다시 회복 동작을 수행한다. 디스크 기반

데이타베이스 시스템에서는 회복 동작 도중에 발생되는 시스템 오류의 횟수가 많아질수록 회복에 드는 시간은 점점 더 오래 걸리게 된다[Moh92].

4. 메인 메모리 상주 데이타베이스 시스템을 위한 회복 기법의 특징

메인 메모리 상주 데이타베이스 시스템에서의 회복 기법도 제 3장에서 기술한 디스크 기반 데이타베이스 시스템의 WAL 기법과 근본적으로는 큰 차이점이 없다. 그러나 성능을 위하여 메인 메모리내에 전체 데이타베이스가 상주한다는 특성이 회복 기법에서도 고려되어야 한다. 본 장에서는 메인 메모리 상주 데이타베이스 시스템 고유의 회복 기법에서 나타나는 특성에 대하여 논의한다. 제 4.1절에서는 트랜잭션 종료시의 처리에 관하여 설명하고, 제 4.2절과 제 4.3절에서는 각각 체크포인트시의 동작과 시스템 오류 발생시의 회복을 위한 재시동 동작에 대하여 설명한다.

4.1 트랜잭션 종료시의 동작

본 절에서는 트랜잭션 종료시 로그 레코드들을 안전한 장치내에 보존시키는 여러 기법에 대하여 논의한다.

(1) IMMEDIATE COMMIT

IMMEDIATE COMMIT은 디스크 기반 데이타베이스 시스템에서 사용되는 COMMIT과 같은 방식이다. 즉, 하나의 트랜잭션이 수행되는 동안 발생되는 모든 변경은 메인 메모리내의 데이타베이스에 그대로 반영되며, 대응되는 로그 레코드를 로그 버퍼내에 기록한다. 트랜잭션의 종료 시점에 이르면, 로그 버퍼내에 있는 그 트랜잭션의 모든 로그 레코드들을 디스크에 반영시킨다. 반영이 완료되는 시점이 그 트랜잭션의 COMMIT 시점이 되며, 이때 사용자에게 트랜잭션이 종료되었음을 알린다.

이 방식이 메인 메모리 상주 데이타베이스 시스템에 그대로 적용되는 경우 트랜잭션에 대한 응답 시간이 길어진다는 문제가 발생한다. 실제 데이타베이스를 액세스할 때에는 디스크 액세스가 발생되지 않는 반면 트랜잭션 종료시에는 해당 트랜잭션의 모든 로그 레코드들을 디스크에 반영시켜야 하므로 트랜잭션 종료시의 오버헤드는 상대적으로 더욱 크게 부각된다. 따라서 이 기법은 메인 메모리 상주 데이타베이스 시스템 고유의 장점을 살릴 수 없다.

(2) GROUP COMMIT

GROUP COMMIT[Sal86]은 로그 버퍼를 디스크에 반영시키는 횟수를 줄이기 위한 방법이다. 이 기법은 하나의 트랜잭션이 종료될 때, COMMIT 로그 레코드를 로그 버퍼에 기록한 직후 바로 그 로그 페이지를 디스크에 반영시키는 IMMEDIATE COMMIT 방법과는 달리 COMMIT 레코드가 존재하는 로그 버퍼의 페이지가 로그 레코드들에 의하여 가득 차는 경우에만 디스크에 반영시킨다. 트랜잭션이 COMMIT되는 시점은 해당 트랜잭션의 COMMIT 로그 레코드가 존재하는 로그 버퍼 페이지가 디스크에 반영되는 시점이 되며, 이때 사용자에게 트랜잭션이 종료되었음을 알린다.

GROUP COMMIT은 로그 버퍼 페이지가 디스크에 반영되는 횟수를 줄임으로써 전체 시스템의 성능은 개선시킬 수 있으나 트랜잭션 COMMIT 레코드가 존재하는 로그 버퍼 페이지가 가득 찰 때까지는 그 트랜잭션의 COMMIT을 유보해야 하므로 사용자에 대한 응답 시간은 IMMEDIATE COMMIT 기법보다도 오히려 길어지는 경

우가 발생한다. 또한 트랜잭션이 COMMIT될 때까지 액세스한 모든 데이타에 대하여 잠금을 유지해야 하므로 다른 트랜잭션들은 그 데이타를 액세스할 수 없다. 따라서 동시성이 떨어지게 되며, 이것이 전체 시스템의 성능을 저하시키는 원인이 된다.

(3) PRECOMMIT

GROUP COMMIT의 문제점의 하나는 트랜잭션의 COMMIT 로그 레코드가 존재하는 로그 버퍼의 페이지가 가득 차고, 이것이 디스크에 반영될 때까지 그 트랜잭션이 획득했던 잠금을 그대로 유지해야 한다는 것이다. 이러한 GROUP COMMIT의 문제점을 해결하기 위한 방법이 PRECOMMIT이다[Leh87]. PRECOMMIT의 기본 개념은 COMMIT 로그 레코드가 로그 버퍼에 기록됨과 동시에 그 트랜잭션이 액세스했던 모든 데이타에 대한 잠금을 반환한다는 것이다. 따라서 이 트랜잭션이 COMMIT되기 전에도 다른 트랜잭션이 같은 데이타를 액세스할 수 있으므로 동시성을 높일 수 있으며, 따라서 전체 시스템의 성능을 개선시킬 수 있다.

(4) 안정된 로그 버퍼의 사용

트랜잭션 종료시의 디스크 액세스 오버헤드를 피하기 위한 근본적인 해결책으로서 안정된 메인 메모리(stable main memory)[DeW84]를 사용하는 로그 버퍼(stable log buffer)가 도입될 수 있다. 안정된 로그 버퍼는 일반 메인 메모리에 비하여 가격이 비싸고 액세스 속도가 떨어지지만 시스템 오류가 발생한 경우에도 저장된 내용을 보장해 줄 수 있다. 따라서 트랜잭션 종료시에 해당 로그 레코드들을 즉시 디스크에 반영시킬 필요없이 단지 로그 버퍼에 COMMIT 로그 레코드를 기록하면 된다. 결과적으로 트랜잭션 종료시에도 디스크 액세스를 유발시키지 않으므로 사용자에 대한 응답 시간을 크게 단축시킬 수 있다[Sal86][Leh87][Eic87].

로그 버퍼로서 안정된 메인 메모리를 사용하는 경우, 버퍼의 크기는 제한되어 있으므로 트랜잭션들이 진행됨에 따라 버퍼가 로그 레코드들로 가득 차게된다. 따라서 로그 버퍼의 재사용을 위하여 로그 버퍼내의 내용을 주기적으로 로그 디스크에 반영시켜야 한다. 그러나 안정된 로그 버퍼의 내용을 디스크에 반영시키는 동작은 일반 트랜잭션과는 별도로 수행되므로 이에 소요되는 시간은 트랜잭션의 응답 시간에 영향을 미치지 않는다. 이러한 장점으로 인하여 많은 메인 메모리 상주 데이타베이스 시스템에서 안정된 로그 버퍼를 사용하고 있다.

4.2 체크포인트시의 동작

체크포인트 동작의 목적은 시스템 오류의 발생시 빠른 재시동을 가능하게 하는 것이다. 체크포인트시 로그 레코드만을 디스크에 반영시키는 디스크 기반 데이타베이스 시스템의 WAL 기법과는 달리 메인 메모리 상주 데이타베이스 시스템에서는 변경이 가해진 메인 메모리내의 데이타 부분을 모두 디스크에 반영시킨다[Leh87]. 그 이유는 재시동의 시작점을 가능한 최근의 변경이 반영된 데이터베이스로 지정함으로써 재시동시의 성능을 높이기 위한 것이다. 전체 시스템을 정지시킨 상태에서 체크포인트 동작을 수행하는 방법도 있을 수 있으나 이러한 경우 다른 트랜잭션들이 대기해야 하므로 일반적으로 디스크 기반 데이타베이스 시스템의 WAL 기법에서 사용된 퍼지 체크포인트를 사용한다[Sal89].

4.3 회복을 위한 재시동시의 동작

메인 메모리 상주 데이타베이스 시스템

에서 시스템 오류가 발생되면, 메인 메모리내의 데이타베이스 전체가 손상된다. 따라서 회복을 위한 재시동은 안전하게 보존된 로그 레코드들과 가장 최근에 체크포인트된 디스크내의 데이타베이스를 기반으로 수행된다[Hag86][Sal86][Eic87][Kum91].

회복의 시작이 되는 시발점은 가장 최근의 체크포인트 로그 레코드가 저장된 위치이다. 수행 순서는 먼저 디스크내의 가장 최근에 체크포인트되었던 데이타베이스를 메인 메모리에 로드한 후 체크포인트 로그 레코드 이후에 기록된 로그 레코드들을 적용함으로써 제 3장에서 기술되었던 WAL 기법의 회복 동작과 유사한 형태로 진행된다.

하나의 트랜잭션이 액세스하는 부분은 전체 데이타베이스와 비교해 볼 때 상당히 작은 부분에 해당된다. 따라서 전체 데이타베이스를 한꺼번에 회복시키지 않고, 필요한 시스템 데이타 부분만을 먼저 회복시킨 후, 트랜잭션의 수행을 허용하는 방법도 제안된 바 있다[Leh87]. 즉, 시스템 데이타가 회복된 후에는 일반 트랜잭션의 수행과 함께 회복용 프로세서가 백 그라운드로 회복을 병행하며, 만일 어떤 트랜잭션이 아직 회복되지 않은 부분을 액세스하고자 하면, 그 부분은 우선 순위를 높여 데이타베이스 프로세서가 즉각적으로 회복시켜 준다.

5. HALO를 이용한 회복 기법

본 장에서는 사례 연구로서 Prinston대학의 Salem과 Garcia-Molina가 제안한 메인 메모리 상주 데이타베이스 시스템에서의 회복 기법[Sal86]을 소개한다. 이 기법의 가장 큰 특징은 로깅을 위한 전용 하드웨어로서 HALO(HArdware LOgging)를 사용한다는 것이다. 제 5.1절에서는 이 기법에서 제안하는 전체 시스템 구성에 대하여 살펴보고, HALO의 기능에 대하여 언급한다. 제 5.2절에서는 트랜잭션 종료시의 동작에 대하여 알아본다. 제 5.3절에서는 체크포인트시의 동작에 대하여 살펴보고, 제 5.4절에서는 시스템 오류시의 회복 동작에 대하여 언급한다.

5.1 HALO를 이용하는 메인 메모리 상주 데이타베이스 시스템의 구성

(그림 3)은 HALO를 이용하는 메인 메모리 상주 데이타베이스 시스템의 전체 구성을 보여준다.

HALO는 로깅 전용 장치로서 로깅 기능이 하드웨어로써 구현되어 그 성능이 뛰어나며, 프로세서와 메인 메모리 사이에 존재한다. 데이타베이스 관리 시스템과 데이타베이스는 일반 메인 메모리내에 존재하며, 메인 메모리 상주 데이타베이스와 로그 버퍼의 내용을 백업하기 위하여 두 개의 디스크가 각각 존재한다.

(그림 4)는 HALO의 세부적인 구성과 그 기능을 보여준다.

프로세서가 메인 메모리내의 데이타베이스에 변경 연산을 요구하면, 이를 감지한 HALO는 프로세서로부터 얻어지는 변경후의 값과 변경되는 부분의 주소를 내부의 쓰기 레지스터 큐(write register queue)에 저장한다. 또한 HALO는 변경되는 부분의 주소를 이용하여 메인 메모리내에서 변경될 부분에 대한 변경전의 값을 읽어들인다. 따라서 프로세서의 변경 연산은 항상 읽기 연산을 수반한다.

변경 전후의 값과 주소가 얻어지면 HALO는 변경 연산에 대한 로그 레코드를 생성하고 이를 내부의 로그 버퍼 큐에 저장한다. 쓰기 레지스터 큐와 로그 버퍼 큐는 모두 안정된 메인 메모리이며, 포인터 IN과 OUT은 로그 버퍼 큐의 시작 부분과 끝 부분을 포인트한다. 변경 연산에 대한 로그 레코드가 로그 버퍼 큐에 올바르게

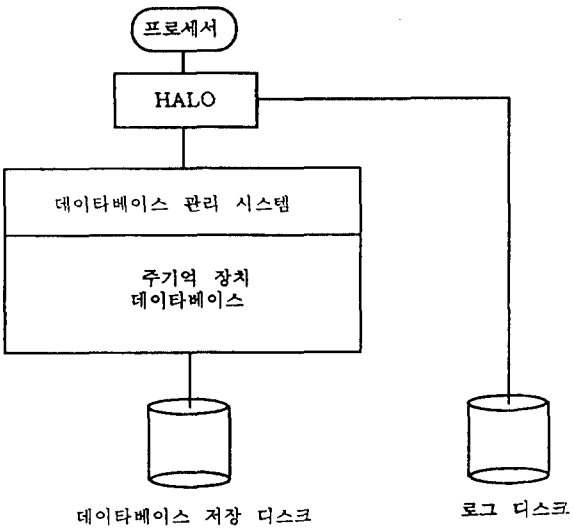

Fig. 3. Architecture of a Memory Resident Database System Based on HALO.

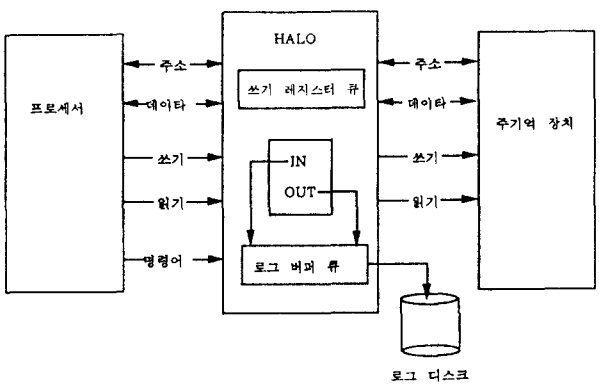

Fig. 4. Components and Their Functions of HALO.

기록되면, HALO는 메인 메모리내의 데이타베이스에 변경을 반영시킨다. HALO가 로깅에 관한 모든 것을 수행하므로 프로세서는 HALO에게 변경을 요청한 후 즉각 다른 데이타베이스 연산을 수행할 수 있으며, 따라서 전체 시스템은 뛰어난 성능을 가지게 된다.

5.2 트랜잭션 종료시의 동작

트랜잭션이 종료되면 프로세서는 HALO에게 이를 알린다. HALO는 트랜잭션의 종료를 의미하는 COMMIT 로그 레코드를 생성하고 이것을 로그 버퍼 큐에 쓴다. 로그 버퍼 큐는 안정된 메인 메모리를 사용하므로 디스크의 강제적 반영이 불필요하다. 트랜잭션이 COMMIT되면, HALO는 그 트랜잭션의 로그 레코드들 내에서 철회를 위한 변경전의 값을 모두 제거한다. 이것은 한번 COMMIT된 트랜잭션은 다시 철회되지 않는다는 원칙과 로그의 내용이 HALO의 로그 버퍼내에 존재하므로 제거 작업이 간단하다는 사실에 근거한 것이다. 이렇게 함으로써 로그 버퍼와 로그 디스크 내의 저장 공간을 보다 효율적으로 사용할 수 있다.

5.3 체크포인트시의 동작

HALO는 단지 로깅의 기능을 구현한 로깅 전용 하드웨어이므로 체크포인트시에는 일반 프로세서가 체크포인트 작업을 수행하게 된다. 체크포인트가 시작되면, 먼저 로그 버퍼에 체크포인트 시작을 나타내는 로그 레코드를 기록하고, 메인 메모리내의 데이타베이스에서 가장 최근의 체크포인트 작업 이후에 변경이 가해진 페이지들을 찾아내어 체크포인트 디스크에 반영시킨다. 모든 페이지가 디스크에 반영되면 체크포인트 완료를 나타내는 로그 레코드를 로그 버퍼내에 기록한다.

5.4 시스템 재시동시의 동작

시스템 오류가 발생하였을 때 회복을 위한 재시동 동작은 로그 디스크와 HALO내의 로그 버퍼의 내용, 그리고 가장 최근의 체크포인트 작업시에 백업하였던 디스크내의 데이타베이스 내용을 기반으로 행해진다. 재시동의 수행 단계는 다음과 같다. 먼

저 로그에서 가장 최근에 완료된 체크포인트 시작 위치를 찾는다. 그리고 디스크내에서 최근에 완료된 체크포인트 시점에 백업되었던 데이타베이스를 찾아 메인 메모리로 로드한다. 로드된 데이타베이스를 기반으로 최근의 체크포인트 시점 이후의 로그 레코드들을 이용하여 데이타베이스를 일관된 상태로 회복한다.

디스크 기반 데이타베이스 시스템에서는 회복 동작중에 다시 시스템 오류가 발생되는 경우 회복에 대한 오버헤드가 더 커지게 된다[Moh92]. 그러나 메인 메모리 상주 데이타베이스 시스템의 경우에서는 이러한 현상이 발생되지 않는다. 이것은 회복 동작중 재수행과 철회 연산을 직접 디스크내의 데이타베이스에 직접 반영시키는 디스크 기반 데이타베이스 시스템과는 달리 메인 메모리 상주 데이타베이스 시스템에서는 단지 메인 메모리내의 데이타베이스에만 재수행과 철회 연산이 반영될 뿐 디스크내의 백업용 데이타베이스내에는 전혀 영향을 미치지 않기 때문이다. 따라서 회복 동작중 시스템 오류가 발생되면, 첫 시스템 오류가 발생하였을 때와 같은 재시동 동작을 수행해 주면 된다.

6. 결 론

회복 기능은 사용자 인터페이스등의 형태로 드러나지 않으므로 데이타베이스를 사용하는 사용자의 입장에서는 그 중요성을 올바르게 인식하지 못하는 경우가 많다. 그러나 컴퓨터 시스템의 오류로 인하여 중요한 데이타가 손상되는 경우를 상상해 보면, 그 중요성을 짐작할 수 있다. 따라서 데이타베이스 관리 시스템이 반드시 갖추어야 할 필수적인 기능이 된다.

메인 메모리 상주 데이타베이스 시스템이란 전체 데이타베이스를 메인 메모리내에 상주시킨 상태에서 각종 데이타베이스 연산을 수행함으로써 뛰어난 성능을 나타낼 수 있다. 그러나 시스템 오류의 발생시 메인 메모리내에 상주하는 전체 데이타베이스 내용을 잃게 되므로 보다 안정된 회복 기능이 요구된다.

본 논문에서는 메인 메모리 상주 데이타베이스 시스템에서의 파손 회복 기법에 대하여 소개하였다. 먼저, 메인 메모리 상주 데이타베이스 시스템의 기본 아키텍춰와 성능 향상의 원인에 대하여 언급하였다. 또한 회복에 대한 기본 이해를 돕기 위하여 디스크 기반 데이타베이스 시스템에서 현재 널리 사용되는 WAL 기법에 대하여 소개하였다. 그리고 메인 메모리 상주 데이터베이스 시스템에서 사용되는 회복 기법의 새로운 개념들에 대하여 논의하였으며, 사례 연구로서 Salem과 Garcia-Molina가 제안한 회복 기법에 대하여 소개하였다.

참 고 문 헌

[DeW84] DeWitt, D. J., et al., "Implementation Techniques for Main Memory Database Systems," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 1-8, 1984.

[Eic87] Eich, M. H., "A Classification and Comparison of Main Memory Database Recovery Techniques," In Proc. Intl. Conf. on Data Engineering, IEEE, pp. 332-339, 1987.

[Gar84] Garcia-Molina, H., Lipton, R. J., and Valdes, J., "A Massive Memory Machine," IEEE Trans. on Computers, Vol. c-33, No. 5,

pp. 391-399, May 1984.

[Hae83] Haerder, T. and Reuter, A., "Principles of Transaction-Orien -ted Database Recovery," ACM Computing Surveys, Vol. 15, No. 4, pp. 287-317, Dec. 1983.

[Hag86] Hagmann, R. B., "A Crash Recovery Scheme for a Memory-Resident Database System,"IEEE Trans. on Computers, Vol. c-35, No. 9, pp. 839-843, Sept. 1986.

[Kum91] Kumar, V. and Burger, A., "Performance Measurement of Some Main Memory Database Recovery Algorithms," In Proc. Intl. Conf. on Data Engineering, IEEE, pp. 436-443, 1991.

[Leh86] Lehman, T. J. and Carey, M. J., "Query Processing in Main Memory Database Management Systems," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 239-250, 1986.

[Leh87] Lehman, T. J. and Carey, M. J., "A Recovery Algorithm for a High-Performance Memory-Resident Database System," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 104-117, 1987.

[Moh89] Mohan, C., et al., "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging, "ACM Trans. on Database Systems, Vol. 17, No. 1, pp. 94-162, Jan. 1989.

[Sal86] Salem, K. and Garcia-Molina, H., Crash Recovery Mechanisms for Main Storage Database Systems, CS-TR-034-86, Departmen-t of Computer Science, Prinston University, 1986.

[Sal89] Salem, K. and Garcia-Molina, H., "Checkpointing Memory-Resident Databases, "In Proc. Intl. Conf. on Data Engineering, IEEE, pp. 452-462, 1989.

[Sha86] Shapiro, L. D., "Join Processing in Database Systems with Large Main Memories,"ACM Trans. on Database Systems, Vol. 11, No. 3, pp. 239-264, Sept. 1986.

[Ver78] Verhofstad, J. S. M., "Recovery Techniques for Database Systems," ACM Computing Surveys, Vol. 10, No. 2, pp. 167-195, Dec. 1983.

[Wha92] Whang, K. Y., "Crash Recovery in Database Systems," Database Review, Korea Information Science Society, Vol. 8, No. 2, pp. 139-156, 1992.