□특 집□

주기억 장치 데이타베이스 시스템에서의 파손 회복 기법†

한국과학기술원 전산학과 황규영* · 김상욱**

목 차

Ⅰ. 서 론

회복 기능(recovery facility)이란 데이타베이스 시스템내에 저장된 데이타를 일관성(consistency)있는 상태로 유지해 주는 기능을 의미하며, 데이타베이스를 총체적으로 관리해주는 데이타베이스 관리 시스템(database management system)이 갖추어야 할 필수적인 기능중의 하나이다[Ver78][Hae83][Ber87][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 있다.

예를 들어, 금전 관리 시스템에서 오류로 인하여 데이타베이스의 일관성이 깨어지는 과정을 살펴보자. 갑이란 사람의 계정에서 을이란 사람의 계정으로 5,000원의 금액을 이월할 때, 두 계정에 들어있는 금액을 각각 20,000원과 10,000원으로 가정하면, 시스템이 정상적으로 동작하는 경우, 다음과 같은 두개의 연산을 수행하게 된다.

(1) 갑의 계정에서 5,000원을 뺀다.
(2) 을의 계정에 5,000원을 더한다.

즉, 이월이 끝난 후의 두 계정의 총액은 갑과 을이 각각 15,000원이 되어 이월전의 총액과 같은 30,000이 되며, 따라서 이월 전후의 데이타는 일관성을 유지하게 된다.

위의 예에서 (1)의 연산이 끝난 직후, 컴퓨터 시스템에 오류가 발생하여 동작이 중단된 경우를 가정해 보자. 갑의 계정에서는 5,000원이 빠져 나갔음에도 을의 계정에서는 오류로 인하여 5,000원이 더해지지 못하였으므로 이월 후의 총액은 이월전의 총액과 다른 25,000원이 된다. 이러한 경우 데이타베이스내에 저장된 데이타는 일관성을 잃게되므로, 사용자는 전혀 예측하지 못한 결과를 얻게 된다. 또한 이미 일관성이 상실된 데이타를 기반으로 이 사실을 모르는 다른 사용자가 연산을 시도하게 되므로, 문제는 점점 심화된다.

† 이 연구는 192 과기처 첨단요소과제 연구비 지원에 의한 결과임 (계정번호 : NN09010).
* 종신회원
** 준회원

회복 관리자(recovery manager)는 컴퓨터 시스템에 발생된 오류로 인하여 데이타의 일관성이 깨어지는 경우, 이를 다시 일관성있는 상태로 회복시켜 주는 기능을 갖는다. 위의 예에서 연산 (1)과 (2)로 구성되는 이월 동작은 하나의 단위 연산과 같이 처리되어야 하며, 따라서 연산 (1)과 (2)가 모두 수행되거나 혹은 모두 수행되지 않아야 데이타의 일관성을 유지할 수 있다. 이와같이 일관성의 단위가 되는 연산들의 집합을 트랜잭션(transaction)이라 하며, 회복 관리자는 일관성의 유지를 위하여 트랜잭션 단위로 회복 기능을 수행하게 된다. 따라서 회복의 단위는 트랜잭션이 된다[Hae83][Ber87].

회복 관리자는 데이타베이스 관리 시스템의 구성 요소인 저장 관리자(storage manager), 동시성 제어자(concurrency controller)등과 상호 협조하여 데이타의 일관성을 유지시켜 준다. 회복 관리자는 사용자 인터페이스의 형태로 드러나는 것이 아니므로 사용자가 직접적으로 회복 관리자의 필요성을 느낄 수 있는 기회는 많지 않으나, 시스템에 오류가 발생되어 필요한 데이타들이 손상되었을 때, 그 진가가 발휘된다. 특히 다루는 데이타의 중요성이 클수록 회복 관리자의 기능은 필수적이다. 예를 들어, 은행에서 저장되는 데이타의 일관성이 깨어진다면, 큰 혼란이 야기될 것은 너무나 자명한 일이다. 이러한 경우 데이타베이스 관리 시스템에서는 사용자에게 보이지 않는 회복 관리자를 가동시켜 일관성이 깨어진 데이타베이스내의 상태를 다시 일관성있는 상태로 만들어 준다.

최근 주기억 장치 기술의 발달로 인하여 주기억 장치의 가격이 점차 지하됨에 따라 컴퓨터 시스템내의 주기억 장치 용량은 점점 증가하는 추세에 있다. 이에 따라 데이타베이스 분야에서는 늘어나는 주기억 장치의 용량을 최대한 활용하여 디스크내에 저장된 데이다를 모두 주기억 장치로 상주시켜 데이타베이스 시스템의 성능을 개선시키기 위한 연구가 활발히 진행중에 있다[DeW84][Gar84][Leh86][Sha86][Wha90].

주기억 장치 데이타베이스 시스템이란 전체 데이타베이스를 주기억 장치내에 상주시킨 상태에서 각종 데이타베이스 연산을 수행하는 시스템을 의미한다[Sal89][Kum91]. 이 시스템의 특징은 모든 데이타베이스 연산이 디스크 액세스를 유발시키지 않고 주기억 장치내에서만 수행되므로 사용자에 대한 응답 시간을 단축시킬 수 있을 뿐만 아니라 전체 시스템의 성능을 높일 수 있다는 것이다.

주기억 장치 데이타베이스 시스템을 구축하는데 있어서 반드시 고려해야 하는 문제점의 하나는 파손시의 회복이다[Hag86][Sal86][Leh87][Eic87][Kum91][Wha87]. 데이타베이스 전체가 주기억 장치내에 상주하기 때문에 시스템 파워나 주기억 장치 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.

본 논문에서는 데이타베이스 시스템의 회복 기능에 대하여 논의하고자 한다. 먼저 디스크 기반 데이타베이스 시스템에서 발생할 수 있는 오류를 트랜잭션 오류, 시스템 오류, 미디어 오류의 세가지로 분류하고, 각각의 오류로부터 복구할 수 있는 기존의 회복 기법들에 대하여 소개한다. 특히 시스템 오류로부터 회복할 수 있는 구체적인 방법으로서 System R에서 사용된 섀도우(shadow) 기법[Lor77][Gra81]과 write-ahead-logging(WAL) 기법[Moh92]에 대하여 자세히 논의한다. 또한 회복 기법의 주요 특성을 트랜잭션의 종료, 체크포인트, 회복을 위한 재시동, 그리고 회복 동작중 발생한 오류로 인한 재시동의 네가지로 분류하고, 각 기법에서의 이 네가지 특성을 논의한다.

본 논문에서는 또한 주기억 장치 데이타베이스 시스템의 회복 기법에 대해서도 논의한다. 주기억 장치 데이타베이스 시스템에서도 디스크 기반 데이타베이스 시스템에서 발생되는 트랜잭션 오류, 시스템 오류, 미디어 오류[1]가 발생될 수 있으며, 따라서 이에 대처할 수 있는 회복 기능이 요구된다. 주기억 장치 데이타베이스 시스템에서는 데이타베이스를 저장하는 주요 매체가 주기억 장치의 특성을 따르게 되므로 디스크 기반 데이타베이스 시스템에서 사용되는 회복 알고리즘을 그대로 적용할 경우에는 그 효율성에 문제가 있다[Leh87]. 그러므로 주기억 장치 데이타베이스 시스템에서는 주기억 장치의 특성을 충분히 활용할 수 있는 새로운 회복 기법이 필요하다.

본 논문의 구성은 다음과 같다. 먼저 제 2장에서는

[1] 주기억 장치 데이타베이스 시스템에서도 시스템 초기화시에 주 기억 장치에 데이타베이스를 로드하기 위하여 디스크내에 데이타베이스를 보존하므로 디스크에 이상이 발생하는 미디어 오류가 존재한다

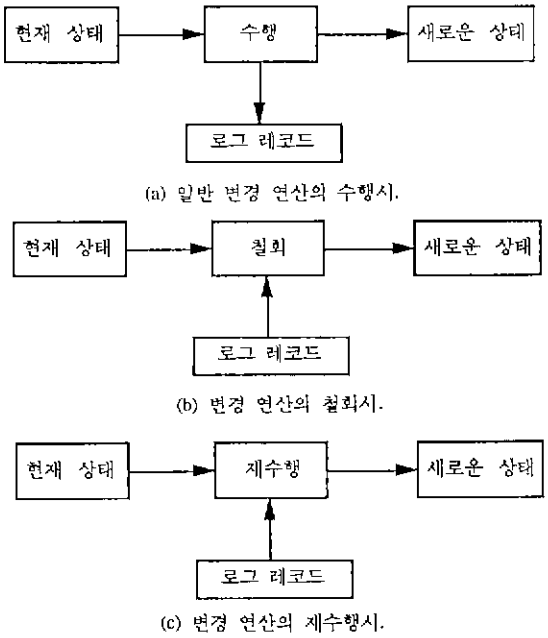
기본 개념이 되는 로깅(logging)에 대하여 소개하고, 제 3장에서는 트랜잭션 오류와 회복 방법에 대하여 설명한다. 제 4장에서는 시스템 오류와 회복 방법에 대하여 설명하고, 제 5장과 제 6장에서는 시스템 오류로부터 회복하는 구체적인 방법으로서 세도우(shadow) 기법[Lor77][Gra81]과 WAL(write-ahead logging) 기법[Moh92]에 대하여 소개한다. 제 7장에서는 미디어 오류와 회복 기법에 대하여 설명한다. 제 8장에서는 디스크 기반 데이타베이스 시스템 회복 기법과 비교될 수 있는 주기억 장치 데이타베이스 시스템의 회복 기법의 특성에 대하여 설명하고, 제 9장에서 결론을 내린다.

II. 로 깅

회복 기능의 가장 기본적인 전략은 데이타를 중복되도록 저장, 관리한다는 것이다. 데이타베이스내에 변경이 발생되면 데이타베이스에 직접 변경을 반영함과 동시에 또 다른 곳에 회복용 데이타를 중복되게 저장한다. 따라서 오류로 인하여 데이타베이스가 일관성을 잃은 경우에도 회복용 데이타를 이용하여 데이타베이스를 일관성 있는 상태로 회복할 수 있다. 본 장에서는 회복용 데이타를 유지하는 방법인 로깅(logging)[Dat84][Ber87][Gra81][Moh92]에 대하여 설명하고자 한다.

로깅은 회복용 데이타를 유지하기 위한 가장 보편적인 방법의 하나로서, 트랜잭션에 대한 로그 레코드(log record)를 기록하는 작업이다. 여기서 트랜잭션이란 사용자가 요구하는 작업의 단위로서, 여러개의 연산들의 연속된 형태로 구성된다. 로그 레코드는 트랜잭션의 각 변경 연산에 대하여 연산 수행전의 데이타인 사전 이미지와 연산 수행 후의 데이타인 사후 이미지를 내용으로 갖는 자료 구조이다. 사전 이미지는 회복 관리자가 디스크내에 반영된 내용을 철회시키는 경우에 사용되며, 사후 이미지는 디스크에 반영될 내용이 시스템 오류로 인하여 디스크에 반영되지 않았을 경우, 이를 디스크에 반영시키는 재수행을 위하여 사용된다.

그림 2.1은 로그 레코드의 생성과 그 기능을 나타낸 것이다. 먼저 데이타베이스에 변경 연산이 가해지면, 해당 변경이 데이타베이스에 반영됨과 동시에 해당되는 로그 레코드가 생성된다(그림 2.1(a) 참조). 만일 오류의 발생으로 변경의 철회가 요구되면, 회복 관리자는 해당 로그 레코드의 사전 이미지를 이용하여 데이타베이스를 원래의 상태로 복귀시켜 준다(그림 2.1(b) 참조). 또한 오류로 인하여 변경시킨 내용이 데이타베이스에 반영되지 않았을 경우에는, 해당 로그 레코드의 사후 이미지를 이용하여 변경을 재수행 시킨다(그림 2.1(c) 참조).

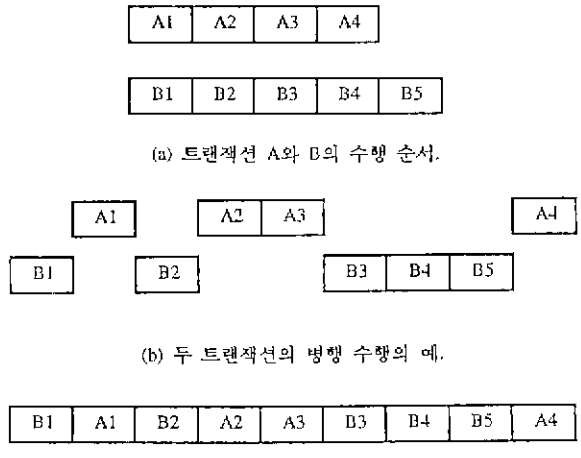


(a) 일반 변경 연산의 수행시.

(b) 변경 연산의 철회시.

(c) 변경 연산의 재수행시.

(그림 2.1) 로그 레코드의 생성과 기능



(a) 트랜잭션 A와 B의 수행 순서.

(b) 두 트랜잭션의 병행 수행의 예.

(c) 로그 레코드의 기록 순서.

(그림 2.2) 두 트랜잭션이 병행적으로 수행되는 경우의 예

로그 레코드는 데이타베이스에 가해지는 변경 연산의 시간적 발생 순서대로 로그 화일에 기록되며, 이 로그 화일은 테이프나 디스크와 같은 저장 장치에 기록된다. 단일 사용자만을 허용하는 데이타베이스 관리 시스템의 경우에서는 한 트랜잭션에 속하는 모든 로그 레코드들이 로그 화일의 인접한 위치에 기록된다. 그러나 여러 사용자들이 동시에 데이타베이스를 엑세스하는 다사용자용 데이타베이스 관리 시스템의 경우에서는 여러 트랜잭션의 수행이 병행되므로 서로 다른 트랜잭션의 로그 레코드들이 혼합된 상태로 기록된다.

그림 2.2는 두개의 트랜잭션이 수행되는 경우에 로그 화일에 저장되는 로그 레코드들의 저장 순서를 나타낸 것이다. 그림 2.2에서 나타난 Ai($1 \le i \le 4$)와 Bj($1 \le j \le 5$)는 각각 트랜잭션 A와 B의 변경 연산을 의미한다. 그림 2.2(a)는 트랜잭션 A와 B가 독립적으로 수행되는 경우에서의 각각의 수행 순서를 나타낸 것이며, 그림 2.2(b)는 두 트랜잭션이 다사용자 환경에서 수행될 때 발생될 수 있는 한 수행 순서의 예를 나타낸 것이다. 또한 그림 2.2(c)는 그림 2.2(b)의 수행 순서대로 진행되는 경우 로그 화일에 기록되는 로그 레코드들의 순서를 나타낸 것이다. 두 트랜잭션의 수행이 병행되더라도 각 트랜잭션의 고유의 수행순서는 그대로 유지됨을 볼 수 있다.

III. 트랜잭션 오류

본 장에서는 트랜잭션 오류(transaction failure) [Gra81][Dat84][Ber87]가 무엇이며, 그 발생 원인과 회복 방법에 대하여 설명한다.

트랜잭션은 BEGIN 연산으로 시작되어 COMMT 혹은 ROLLBACK 연산으로 끝나게 된다. COMMIT으로 끝나는 경우는 사용자가 요구하는 작업을 완료한 경우이며, ROLLBACK으로 끝나는 경우는 어떤 예외 조건이 발생되어 사용자의 요구작업이 성공적으로 완료되지 못한 경우를 나타낸다.

트랜잭션이 ROLLBACK된다는 것은 BEGIN 이후에 데이타베이스에 반영된 모든 연산들을 철회시키는 것을 의미한다. 비행기 좌석 예약 트랜잭션의 경우, 빈 좌석이 없어 예약에 실패하면 그 트랜잭션이 그동안 데이타베이스에 가한 변경을 모두 철회시켜야 하므로 이것이 ROLLBACK에 해당된다. 따라서 사용자는 미리 예외 조건을 예상하여 트랜잭션 내에 예외 조건이 발생되면, 트랜잭션이 ROLLBACK 되도록 조치시켜 놓아야 한다.

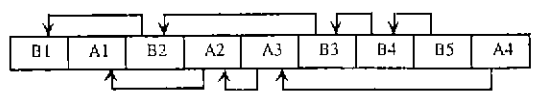
트랜잭션 오류란, 사용자의 예기치 못한 실수로 인하여 트랜잭션이 비정상적으로 끝나는 것을 의미한다. 이러한 트랜잭션 오류의 발생 원인으로서는 오버플로우의 발생, 나눗셈에서 젯수로 0을 사용하는 경우, 메모리의 보호구역의 침범 등이 있다. 이러한 경우 트랜잭션은 COMMIT도 ROLLBACK도 되지 않은 상태에서 종료되므로 데이타베이스내에 있는 데이타는 일관성이 깨어지게 된다. 따라서 회복 관리자는 비정상적으로 종료된 트랜잭션들을 찾아 이들을 강제적으로 ROLLBACK시켜 데이타베이스에 반영된 내용을 철회시켜야 한다.

트랜잭션 오류로부터의 회복은 시스템 자체에 문제가 발생된 것이 아니므로 문제가 발생된 트랜잭션에 해당되는 로그 레코드들만을 찾아내어야 한다. 해당 트랜잭션의 로그 레코드들을 찾는 작업은 로그 화일을 후방으로 연속 탐색을 해야 하므로 작업 시간의 오버헤드가 크다. 따라서 회복 관리자는 로깅시에 같은 트랜잭션에 해당되는 로그 레코드들을 포인터로 연결시키는 후방 포인터 구조를 사용함으로써 트랜잭션 오류로부터의 회복 동작을 효과적으로 처리할 수 있다.

트랜잭션 오류시에 회복 관리자는 최종적으로 로그 화일에 기록된 해당 트랜잭션의 로그 레코드를 찾아 로그 레코드에 기록된 사전 이미지를 이용하여 이미 디스크에 반영된 변경을 철회시킨다. 이러한 철회 작업은 후방 포인터를 이용하여 계속적으로 진행되며, 트랜잭션의 시작을 알리는 로그 레코드를 만나면 트랜잭션의 모든 변경이 철회되었으므로 회복 작업은 종료된다.

그림 3.1은 트랜잭션 오류로부터의 회복을 효과적으로 처리하기 위한 로그 화일의 구조를 나타낸 것이다.

Ai($1 \le i \le 4$)와 Bj($1 \le j \le 5$)는 각각 트랜잭션 A와 B의 변경 연산을 의미한다. 만일 트랜잭션 A에 오류가 발생하면, 먼저 A4를 철회하고 그 후방 포인터를 이용하여 A3를 찾는다. 그리고 A3의 로그 레코드를 이용하여 A3의 변경을 철회시킨 후 다시 후방 포인



(그림 3.1) 트랜잭션 오류로부터의 회복을 효율적으로 처리하기 위한 후방 포인터 구조

터를 이용하여 A2를 찾는다. 이러한 방식으로 A1까지의 모든 변경을 철회하면, 데이타베이스는 트랜잭션 A의 오류로부터 회복이 완료된 것이다.

IV. 시스템 오류

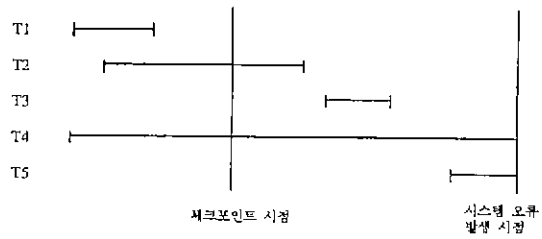
본 장에서는 시스템 오류(system failure)[Gra81][Dat84][Ber87][Moh92]의 발생과 시스템 오류로부터 회복하는 방법에 대하여 설명한다.

시스템 오류란 시스템의 작동을 정지시키는 예기치 못한 사건으로서 이러한 경우 시스템을 재시동시켜야 한다. 시스템 오류의 발생 후에는 디스크에 저장된 데이타베이스의 내용은 손상되지 않지만, 주기억 장치내의 내용은 아직 디스크에 반영되지 않은 상태에서 손실된다. 이러한 경우 시스템 오류가 발생된 시점에서 진행중이던 트랜잭션들은 수행이 완료되지 않았으므로 디스크 내에 반영되었던 일부 내용들을 철회시켜야 하며, 수행이 완료되었으나 결과가 디스크에 반영되기 전의 트랜잭션은 재수행시켜야 한다.

철회 작업을 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, 주어진 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, 최근에 기록된 체크포인트 레코드에서부터 회복 작업을 수행할 수 있으므로 효율적이다.

체크포인트의 개념을 보다 자세히 이해하기 위하여 그림 4.1의 예를 살펴보자. 그림에서 나타난 T1에서 T5까지는 각각 트랜잭션을 의미한다. T1은 최근에 기록된 체크포인트 시점 이전에 이미 종료된 트랜잭션을 의미하고, T2는 체크포인트 시점 이전에 시작되어 시스템 오류 시점 이전에 종료된 트랜잭션을 의미한다. T3는 체크포인트 시점 이후에 시작되어 시스템 오류 시점 이전에 이미 종료된 트랜잭션을 의미하고, T4는 체크포인트 시점 이전에 시작되어



(그림 4.1) 시스템 오류로부터 회복시의 체크포인트의 역할

시스템 오류 시점에서 계속 진행중인 트랜잭션을 의미한다. 끝으로 T5는 체크포인트 시점 이후에 시작되어 시스템 오류 시점에서 계속 진행중이던 트랜잭션을 의미한다.

위의 예에서 시스템 오류가 발생하였을 경우, 트랜잭션 T4와 T5는 철회되어야 한다. 그 이유는 시스템 오류가 발생한 시점에서 두 트랜잭션은 아직 종료되기 전이므로 진행중에 데이타베이스에 가한 변경이 데이타베이스를 일관성 없는 상태로 만들었기 때문이다. 트랜잭션 T2와 T3는 재수행 되어야 한다. 그 이유는 트랜잭션은 시스템 오류 발생 이전에 종료되었으나, 이것이 실제적으로 디스크에 아직 반영되지 않고 주기억 장치내의 버퍼에만 반영되어 이 데이타가 시스템 오류로 인하여 손상되었을 가능성이 있기 때문이다. 따라서 회복 관리자는 이러한 트랜잭션에 대하여 로그를 이용하여 재수행함으로써 변경된 내용을 디스크내의 데이타베이스에 반영시켜야 한다.

시스템 오류가 발생되었을 때, 회복 관리자가 취해야 할 행동은 다음과 같이 요약될 수 있다.

(1) 로그 화일에 기록된 가장 최근의 체크포인트 레코드를 찾는다. 이를 위해서 디스크내의 특정 위치에 포인터 변수를 두어 최근의 체크포인트 레코드의 위치를 이곳에 기록해 둔다.

(2) 단계 (1)에서 얻은 체크포인트 레코드에서 체크포인트 레코드를 기록할 당시에 진행중인 트랜잭션의 리스트를 얻어, 이를 철회되어야 할 트랜잭션 리스트에 넣는다.

(3) 체크포인트 레코드가 기록된 부분에서 시작하여 시스템 오류가 발생된 시점까지 로그 화일을 순차적으로 탐색한다.

(3.1) 새로운 트랜잭션이 시작되면, 이 트랜잭션을 철회되어야 할 트랜잭션 리스트에 첨가한다.

(3.2) 트랜잭션의 종료를 알리는 로그 레코드를

만나면, 이 트랜잭션을 철회되어야 할 트랜잭션 리스트에서 삭제하고, 재수행되어야 할 트랜잭션 리스트에 첨가한다.

(4) 철회될 트랜잭션 리스트와 재수행되어야 할 트랜잭션 리스트를 이용하여 실제적으로 철회와 재수행을 한다.

본 장에서는 시스템 오류시에 발생되는 문제점과 이 때 회복 관리자가 취해야 할 행동에 대하여 알아보았다. 그러나 본 장에서 기술된 내용은 시스템 오류로부터의 회복을 이해하기 위한 개념적인 내용이다. 제 5장과 제 6장에서는 보다 구체적인 내용으로서, 널리 쓰이고 있는 두 가지 기법인 세도우(shadow) 기법[Lor77][Gra81]과 WAL(write-ahead logging) 기법[Moh92]에 대하여 살펴보도록 한다.

V. 세도우 기법

본 장에서는 시스템 오류로부터 회복하기 위한 기법의 하나인 세도우(shadow) 기법[Lor77][Gra81]에 대하여 설명한다. 먼저 세도우 버전과 현재 버전의 차이점을 설명하고, 다음은 회복 기법의 주요 특성으로 나타나는 네가지 기능 즉, 트랜잭션의 종료, 체크포인트, 회복을 위한 재시동, 그리고 회복 동작중 발생한 오류로 인한 재시동에 대하여 설명한다.

5.1 세도우 버전, 현재 버전

세도우 기법은 데이타베이스 관리 시스템의 하나인 System R에서 사용된 회복 기법으로서 데이타가 저장된 각 화일은 세도우 버전(shadow version)과 현재 버전(current version)의 두 가지 버전을 갖는다. 세도우 버전은 시스템 오류로부터 회복을 할 때 기반이 되는 버전이며, 최근의 체크포인트 시점까지의 변경 연산과 시스템 오류 발생전에 이미 정상적으로 종료된 변경 연산을 반영한다. 현재 버전은 세도우 버전에 현재 진행중인 트랜잭션의 변경 연산을 반영시키는 버전이 된다.

트랜잭션이 정상적으로 종료되면, 현재 버전은 새로운 일관된 상태를 반영하며, 새로운 트랜잭션이 다시 이 화일을 변경시키고자 할 때, 세도우 버전으로 복제된다. 하나의 화일에 대하여 두 가지의 버전이 존재하므로 시스템 오류가 발생하면, 회복 관리자는 회복의 기반이 되는 세도우 버전과 시스템 오류 이전에 종료된 트랜잭션을 재수행 시킬 수 있는 로그 화일을 가지고서 데이타베이스의 일관성을 회복할 수 있다.

5.2 트랜잭션 종료시의 동작

정상적으로 종료된 트랜잭션의 변경 연산은 데이타베이스에 반드시 반영되어야 하므로, 회복 관리자는 트랜잭션이 종료될 때, 해당 트랜잭션에 해당되는 로그 레코드와 종료를 나타내는 COMMIT 레코드를 로그 화일에 반영시킨다. 이때 로그 화일에 반영되는 내용들은 디스크내의 로그 화일에 강제적으로 반영된다. 그 이유는 로그 레코드의 내용이 버퍼에만 반영되는 경우 디스크에 미처 반영되기 전에 시스템 오류가 발생된다면, 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. 회복시에 진행중인 것으로 간주되는 트랜잭션들은 철회되므로, 모든 연산이 완료되더라도 COMMIT 레코드가 로그 화일에 반영되지 않은 트랜잭션들은 아직 수행중인 트랜잭션과 마찬가지로 취급된다.

5.3 체크포인트시의 동작

전장에서 기술된 바와 같이 체크포인트는 회복 동작시의 작업량을 줄여줌으로써 회복의 효율을 높여주는 역할을 한다. 세도우 기법에서 체크포인트를 취하기 위한 작업은 다음과 같다.

(1) 시스템에서 수행중인 모든 트랜잭션의 동작들을 중단시킨다.

(2) 체크포인트 레코드를 기록하고, 현재까지 로그 버퍼에 기록된 내용을 디스크내의 로그 화일에 강제적으로 반영시킨다.

(3) 현재 버전의 내용을 가진 모든 데이타 버퍼의 내용들을 디스크에 강제적으로 반영시킨다.

(4) 체크포인트 시점에서의 모든 현재 버전들이 가진 내용들을 세도우 버전에 반영시킨다.

(5) 체크포인트 레코드의 위치를 디스크내의 포인

터 변수에 기록한다.

단계 (4)에 의하여 세도우 버전들이 현재 버전들의 내용을 반영하므로, 현재 진행중인 트랜잭션의 변경 연산의 결과도 세도우 버전에 반영될 수 있다. 따라서 체크포인트가 끝난 시점에서의 세도우 버전들의 상태는 수행중인 트랜잭션의 변경 연산의 결과도 반영된 상태가 된다. 그러나 단계 (2)에서 로그 화일도 디스크에 기록되므로 세도우 버전의 모든 변경 상태를 로그 화일이 그대로 반영하고 있게 된다. 즉, 체크포인트 시점에서는 디스크 내의 세도우 버전에 기록된 변경내용과 디스크 내의 로그 화일에 기록된 내용이 일치된다. 즉, "데이타베이스=로그"라는 관계가 성립된다.

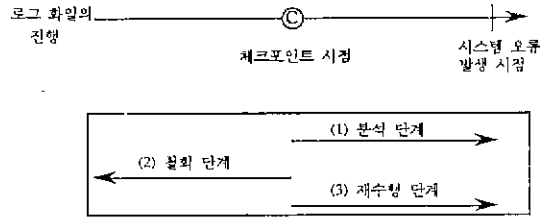
회복 관리자가 체크포인트를 위하는 동작이 완료되는 시점은 단계 (5)가 끝나는 시점이다. 만일 단계 (5)가 끝나기 전에 시스템 오류가 발생되면, 회복 관리자는 현재 로그 화일에 기록된 최근의 체크포인트 레코드가 아닌 디스크에 기록되어 있는 포인터 변수가 가리키는 다른 체크포인트 레코드를 참조하여 회복 작업을 시작하게 된다. 따라서 회복 관리자는 포인터 변수가 가리키는 체크포인트 레코드를 기준 시점으로 사용하여 데이타베이스를 일관성있는 상태로 회복할 수 있다.

5.4 회복을 위한 재시동시의 동작

제 5.3 절에서 기술된 바와 같이 체크포인트 레코드가 로그 화일에 반영된 직후에는 세도우 페이지와 로그 화일은 변경의 진행 상황이 같다. 따라서 시스템 오류가 발생되었을 때 회복 작업은 체크포인트 레코드를 기점으로 수행되게 되며, 모든 현재 버전들은 무시되고 로그 화일에 기록된 로그 레코드들과 세도우 버전들이 회복을 위한 정보로서 사용된다. 회복의 작업 단계는 아래와 같으며, 그림 5.1은 회복 작업이 진행되는 것을 그림으로 나타낸 것이다.

(1) 디스크내의 포인터 변수가 가리키는 최근의 체크포인트 레코드를 찾는다. 여기에는 체크포인트 시점에서 진행중이던 트랜잭션의 리스트가 기록되어 있다.

(2) 체크포인트 레코드에서 시작하여 시스템 오류가 발생된 시점까지 로그 화일을 탐색하며, 철회되어야 할 트랜잭션과 재수행되어야 할 트랜잭션을 구분하게 되는데 이것이 분석 단계이다.



(그림 5.1) 세도우 기법에서의 시스템 오류로부터의 회복 과정

(3) 체크포인트 레코드에서 시작하여 철회될 트랜잭션의 로그들을 로그 화일의 후방향으로 탐색하며, 철회 작업을 한다. 이러한 철회 작업의 결과는 세도우 버전에는 영향을 미치지 않은 상태에서 세도우 버전의 내용을 복제한 현재 버전에만 반영된다.

(4) 체크포인트 레코드에서 시작하여 재수행될 트랜잭션의 로그들을 로그 화일의 전방향으로 탐색하며, 재수행 작업을 한다. 이러한 재수행 작업의 결과는 단계 (3)에서와 마찬가지로 세도우 버전에는 영향을 미치지 않은 상태에서 현재 버전에만 반영된다.

(5) 단계 (4)까지의 작업이 완료되면, 이 시점을 기준으로 새로운 체크포인트를 취한다.

5.5 회복중 발생되는 시스템 오류로부터의 회복시의 동작

오류가 발생된 직후는 시스템의 상태가 불안정한 상태이므로 다시 오류가 발생될 가능성이 매우 높다. 따라서 회복 관리자가 회복 작업을 수행하는 과정에서도 시스템 오류는 발생할 수 있다. 그러므로 회복 관리자는 이러한 상황에서도 다시 데이타베이스를 일관된 상태로 회복시키는 기능을 가져야 한다.

다음 장에서 소개될 WAL(write-ahead logging) 기법 등의 다른 회복 기법에서는 이러한 회복 중 발생되는 오류로부터의 회복이 상당히 복잡한 과정을 거치게 되지만, 세도우 기법에서는 매우 간단하게 처리될 수 있다. 즉, 회복 작업의 단계 (3)과 (4)에서 나타난 바와 같이 회복 작업이 끝나기 전에는 모든 철회와 재수행 작업이 현재 버전에만 반영되므로 회복 작업중에 시스템 오류가 발생되더라도, 세도우 버전에는 첫번째 시스템 오류시와 같은 상태의 데이타가 그대로 보존된다. 따라서 원래의 세도우 버전과 최

근의 최근의 체크포인트 레코드를 이용하여, 첫번째 시스템 오류로부터 회복할 때와 마찬가지 동작을 취함으로써 회복이 가능하다.

세도우 기법은 시스템 오류로부터의 회복시의 성능이 좋고, 회복 도중 또 다시 시스템 오류가 발생되었을 때, 이를 간단히 다시 회복할 수 있다는 것이 장점이다. 그러나 세도우 기법은 변경되는 각 데이타에 대하여 현재 버전과 세도우 버전의 두 가지 형태를 유지해야 하므로 이를 위한 저장 공간의 오버헤드가 크며, 체크포인트를 취할 때 현재 비젼을 세도우 버전에 반영시켜야 하므로 체크포인트시의 성능이 떨어진다는 단점이 있다. 또한 체크포인트시에 전체 시스템을 정지시켜 모든 트랜잭션의 수행이 일시적으로 중단된다는 것이 단점이다. 다음 장에서는 이러한 단점을 극복할 수 있는 회복 기법으로서 WAL (write-ahead logging) 기법을 소개한다.

VI. WAL 기법

본 장에서는 시스템 오류로부터 회복하기 위한 기법의 하나인 WAL(write-ahead logging) 기법[Moh 92]에 대하여 설명한다. 먼저 WAL 기법의 장점 및 기본 원리에 대하여 설명하고, 다음은 회복 기법의 주요 특성으로 나타는 네가지 기능, 즉, 트랜잭션의 종료, 체크포인트, 회복을 위한 재시동, 그리고 회복 동작 중 발생한 오류로 인한 재시동에 대하여 설명한다.

6.1 기본 원리

WAL 기법은 데이타 버퍼에 있는 내용이 디스크에 반영되기 전에, 그 변경을 나타내는 로그 레코드가 있는 로그 버퍼의 내용이 먼저 디스크에 반영되어야 한다는 원칙을 이용하는 기법이다. 이 기법은 세도우 버전을 사용하지 않고 데이타베이스에 직접 변경을 반영함으로써 저장 공간의 오버헤드를 줄일 수 있으며, 하나의 버전만을 유지하면 되므로 체크포인트시의 세도우 기법이 가지는 성능 저하의 문제점도 해결된다. 또한 퍼지 체크포인팅(fuzzy checkpointing)을 쓸 수 있어 체크포인트시에도 전체 시스템을 정지시키지 않으므로 트랜잭션이 중단되는 현상을 방지할 수 있다.

로그의 내용을 디스크에 먼저 반영시키는 이유는, 오직 하나의 데이타 버전만이 존재하기 때문이다. 즉, 세도우 기법에서는 세도우 버전에서 데이타의 변경전 상태를 그대로 반영하고 있지만, WAL 기법에서는 하나의 데이타 버전에 변경이 그대로 반영된다. 따라서 데이타 버전이 먼저 디스크에 반영되고 로그는 디스크에 반영되기 전에 시스템 오류가 발생되면, 현재 데이타 비젼을 변경 이전의 상태로 철회시켜 줄 수 있는 방법이 존재하지 않는다. 이것은 사전 이미지를 가진 로그 레코드가 디스크에 존재하지 않기 때문이다. 그러나 로그 레코드가 먼저 디스크에 반영되면, 데이타 버전이 디스크에 반영되기 전에 시스템에 오류가 발생되더라도 로그 레코드내의 사전 이미지를 이용하여 철회가 가능하므로 일관성있는 상태로 회복될 수 있다.

현재 상용화된 데이타베이스 관리 시스템에서는 WAL 기법의 기본 개념을 이용한 여러가지 변형된 기법들을 사용하고 있다. 그 전형적인 예가 IBM에서 개발한 ARIES(Algorithm for Recovery and Isolation Exploiting Semantics) 기법[Moh92]이다. 본 장에서는 ARIES 기법의 특성을 소개한다.

6.2 트랜잭션 종료 및 체크포인트시의 동작

트랜잭션의 정상적인 종료시에 회복 관리자가 취해야 할 동작은 세도우 기법에서와 같다. 반면, 체크포인트에서는 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.

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

단계 (1)에서 세도우 기법에서와는 달리 전체 시스템을 정지시키지 않는 퍼지 체크포인트를 사용한다. 따라서 수행중인 트랜잭션의 동작이 체크포인트 동작과 병행될 수 있으므로 트랜잭션 응답 시간의 지연을 방지할 수 있다는 것이 큰 장점이다. 또한 세도우 기법에서와는 달리 단계 (3)에서 로그 화일과 데이타

화일을 디스크에 강제적으로 반영시키지 않는다. 데이타 버퍼를 디스크에 반영시키는 작업은 상당히 시간이 걸리는 큰 오버헤드를 가지므로 체크포인트시에 WAL기법의 성능은 세도우 기법보다 나은 성능을 보인다.

6.3 회복을 위한 재시동시의 동작

시스템 오류가 발생하였을 때, 회복 기법은 단지 로그 화일에 내용과 현재까지 수행된 변경 연산이 반영된 데이타베이스의 내용을 기반으로 행해진다. 회복 기법이 수행되는 단계는 아래와 같으며, 그림 6.1은 회복 동작이 수행되는 것을 그림으로 나타낸 것이다.

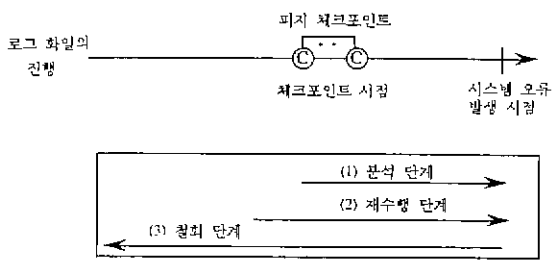
(1) 체크포인트 동작의 단계 (6)에서 기록한 포인터 변수를 이용하여 최근의 완료된 체크포인트의 시작 위치를 찾는다.

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

(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 트랜잭션들까지도 함께 재수행된다. 이것을 이력의 재수행(repeating history)이라 부른다.

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

(5) 체크포인트를 취한다.



(그림 6.1) WAL 기법에서의 시스템 오류로부터의 회복 과정

단계 (4)에서 오류가 발생한 시점까지 모든 로그 레코드의 내용을 재수행시키는 이유는 로그와 데이타베이스의 내용이 일치되는 시점을 만들기 위해서이다. 즉, "데이타베이스=로그"라는 관계를 정립하기 위한 것이다. 세도우 기법에서는 체크포인트시에 모든 트랜잭션의 동작을 정지시키고, 로그 화일과 변경이 발생된 데이타베이스를 디스크에 직접 반영시키므로 체크포인트 시점에서는 로그 화일과 데이타베이스의 내용이 일치된다. 그러나 WAL 기법에서는 다른 트랜잭션의 동작을 정지시키지 않고, 또한 세도우 버전을 이용하지 않으므로 오류가 발생한 시점까지 모든 로그 레코드를 재수행함으로써 데이타베이스의 내용과 로그 화일의 내용이 일치되는 시점을 얻을 수 있다.

6.4 회복중에 발생되는 시스템 오류로부터의 회복시의 동작

WAL 기법에서는 세도우 버전을 유지하는 세도우 기법에서와는 달리 회복 동작 중에 데이타를 직접 변경시킨다. 철회 동작이 완료된 후 철회 동작의 완료를 알리는 로그 레코드만이 디스크에 반영되고 데이타 버퍼는 디스크에 반영되지 않은 상태에서 시스템 오류가 발생되면, 데이타 베이스가 일관성을 잃은 상태로 되는 경우가 발생된다. 이는 철회 동작이 어디까지 수행되었는가를 알 수 없기 때문에 발생되는 현상이다. 따라서 이를 해결하기 위하여 철회 동작에 대하여 로그 레코드를 기록하게 되는데 이를 CLR(compensation log record)이라 한다.

시스템 오류로부터 회복 동작을 수행하는 과정에서 다시 시스템 오류가 발생되면, 최근의 체크포인트 레코드를 이용하여 다시 회복 동작을 수행한다. 첫번째 시스템 오류로부터의 회복 동작 도중 단계 (4)에서 철회 동작에 대한 CLR이 추가되었으므로 두번째 시스템 오류로부터의 회복 동작 단계 (3)에서 재수행되어야 하는 로그 레코드의 수가 더 많아지고, 마찬가지 원리에 의하여 단계 (4)에서 철회되어야 하는 로그 레코드의 수도 많아진다. 따라서 회복 동작 도중에 발생되는 시스템 오류의 횟수가 많아질수록 회복에 드는 시간은 점점 더 오래걸리게 된다. ARIES에서는 UndoNextLSN이라는 포인터를 이용하여 이러한 효율성의 문제점을 해결하고 있으나, 자세한 설

명은 지면 관계상 생략한다.

VII. 미디어 오류

본 장에서는 미디어 오류(media failure)[Gra81][Dat84]와 이로부터 회복하는 방법에 대하여 언급하고자 한다.

미디어 오류란 디스크의 일부가 파손되어 저장된 데이타베이스의 내용 자체가 손상된 것을 의미한다. 디스크 파손의 원인으로는 디스크 헤드가 디스크를 파손시키는 경우, 디스크가 충격 등으로 인하여 물리적으로 손상되는 경우 등이 있다. 따라서 미디어 오류는 시스템 오류와는 달리 디스크에 저장된 데이타베이스 자체가 손상되는 것이므로 이를 회복하기 위한 또 다른 회복 전략이 필요하다.

미디어 오류로부터의 회복을 위한 기본적인 전략은 시스템 오류로부터의 회복 기법과 유사하다. 즉, 디스크에 저장된 데이타베이스의 내용을 또 다른 저장장치(디스크 혹은 테이프)에 덤프시키고, 로그 화일을 유지한다. 미디어 오류가 발생되면, 덤프된 내용을 기반으로 덤프 이후에 종료된 트랜잭션의 변경을 로그를 이용하여 회복시킬 수 있다.

데이타베이스를 덤프시키는 동작을 살펴보자. 덤프는 데이타베이스에 대하여 일정 간격으로나 혹은 어떤 작업이 종료된 후, 또는 시스템이 바쁘지 않을때나 혹은 사용자의 요구에 의하여 행해질 수 있다. 덤프는 디스크내의 내용을 읽어 다시 디스크(혹은 테이프)에 기록하여야 하므로, 작업시간이 크다. 따라서 덤프가 행해지는 동안 트랜잭션의 처리가 중단되며 시스템의 응답시간에 큰 영향을 미치게 되므로 덤프중에도 다른 트랜잭션을 정상적으로 처리할 수 있는 덤프방식이 바람직하다.

덤프는 데이타베이스 전체에 대하여 취해지기도 하지만, 이러한 경우 전체 데이타베이스를 읽고 쓰는 작업 시간이 지나치게 커진다. 따라서 최종적으로 덤프된 이후에 변경된 부분에 대하여 덤프시키는 방식이 많이 사용된다. 이러한 덤프 방식을 추가식 덤프 방식(incremental dump)이라 한다. 이러한 추가식 덤프 방식은 방대한 양의 데이타베이스를 다루는 시스템에서 사용된다.

미디어 오류가 발생한 경우 회복 관리자가 취해야 하는 행동은 다음과 같다.

(1) 최근에 덤프된 데이타베이스의 내용을 저장장치에서 로드시킨다. 이것이 미디어 오류로부터 회복하기 위한 기반 데이타베이스가 된다.

(2) 단계 (1)에서 로드한 데이타베이스를 기반으로 하고, 최근에 덤프된 시점 이후에 완료된 트랜잭션을 로그 화일에서 찾아 재수행시킨다.

지금까지 언급된 모든 회복 방식은 로그 화일에 손상되지 않는다는 가정을 전제로 한 것이다. 그러나 실제로는 로그 화일도 디스크에 저장되는 것이므로 손상될 가능성이 있다. 따라서 만일 로그 화일 자체가 손상된다면, 회복이 불가능해지므로 상용되는 데이타베이스 관리 시스템에서는 동일한 로그 화일을 서로 다른 저장 장치에 따로 유지하는 방식을 사용하고 있다. 따라서 로그 화일이 저장된 곳에 미디어 오류가 발생하더라도 또 다른 로그 화일을 사용하여 회복할 수 있게 된다.

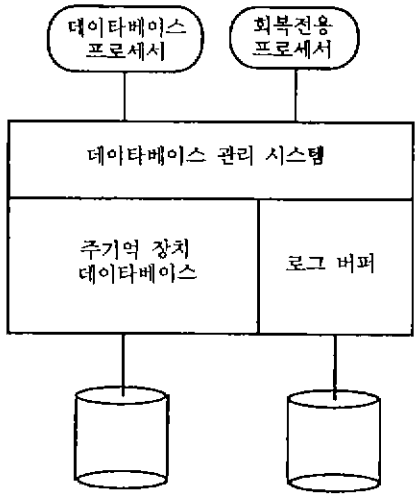
VIII. 주기억 장치 데이타베이스 시스템을 위한 회복 기법의 특징

주기억 장치 데이타베이스 시스템에서의 회복 기법도 전술한 디스크 기반 데이타베이스 시스템의 회복 기법과 근본적으로는 큰 차이점이 없다. 그러나 성능을 위하여 주기억 장치내에 전체 데이타베이스가 상주한다는 특성이 회복 기법에서도 고려되어야 한다. 본 장에서는 주기억 장치 데이타베이스 시스템 고유의 회복 기법에서 나타는 특성에 대하여 논의한다. 제 8.1절에서는 주기억 장치 데이타베이스 시스템의 기본 아키텍춰와 각 구성 요소에 대하여 간략히 소개하고, 제 8.2절에서는 트랜잭션 종료시의 처리에 관하여 설명한다. 제 8.3절과 제 8.4절에서는 각각 체크포인트시의 동작과 시스템 오류 발생시의 회복을 위한 재시동 동작에 대하여 설명한다.

8.1 주기억 장치 데이타베이스 시스템 구성

그림 8.1은 주기억 장치 데이타베이스 시스템의 기본 아키텍춰를 나타낸 것이다[Eic87].

먼저 주기억 장치내에는 전체 데이타베이스가 상주하게 되고, 이를 관리하기 위한 데이타베이스 관리 시스템, 그리고 로그 레코드를 기록하기 위한 로그 버퍼가 존재한다. 한편, 디스크내의 로그 디스크는 주기억 장치내 로그 버퍼의 내용을 백업 받음으로써 로그 버퍼를 재사용할 수 있도록 하기 위한 것이며, 디스크내에 존재하는 데이타베이스는 시스템을 초기화할 때 주기억 장치로 데이타베이스를 로드시키기



(그림 8.1) 주기억 장치 데이타베이스 시스템의 기본 아키텍춰

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.

8.2 트랜잭션 종료시의 동작

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

8.2.1 IMMEDIATE COMMIT

IMMEDIATE COMMIT[Sal86]은 디스크 기반 데이타베이스 시스템에서 사용되는 WAL 기법의 트랜잭션 종료와 같은 방식으로 수행된다. 즉, 하나의 트랜잭션이 수행되는 동안 모든 변경은 주기억 장치내의 데이타베이스에 그대로 반영되며, 대응되는 로그 레코드를 로그 버퍼내에 기록한다. 트랜잭션의 종료 시점에 이르면, COMMIT 레코드를 기록하고 이 시점까지의 로그 레코드들을 모두 디스크에 반영시킨다. 반영이 완료되는 시점이 그 트랜잭션의 COMMIT 시점이 되며, 이때 사용자에게 트랜잭션이 종료되었음을 알린다.

이 방식이 주기억 장치 데이타베이스 시스템에 그대로 적용되는 경우의 문제점은 트랜잭션에 대한 응답 시간이 길어진다는 것이다. 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 주기억 장치 데이타베이스 시스템 고유의 장점을 살릴 수 없다.

8.2.2 GROUP COMMIT

GROUP COMMIT[Sal86]은 로그 버퍼를 디스크에 반영시키는 횟수를 줄이기 위한 방법이다. 이 기법은 하나의 트랜잭션이 종료될 때, COMMIT 레코드를 로그 버퍼에 기록한 직후 바로 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 레코드의 로그 버퍼 페이지가 디스크에 반영되는 시점이 되며, 이때 사용자에게 그 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 가득찰 때까지는 그 트랜잭션의 COMMIT를 유보해야 하므로 사용자에 대한 응답 시간은 IMMEDIATE COMMIT 기법보다도 오히려 길어진다. 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.

8.2.3 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. PRECOMMIT을 사용하더라도 실질적으로 엄격한 이단계 잠금(strict two-phase locking) 프로

토콜을 준수하는 것이므로 데이타베이스의 일관성을 유지하는 데에는 문제가 되지 않는다. 예를 들어, 트랜잭션 A가 잠금을 반환한 데이타를 트랜잭션 B가 액세스한다고 가정해 보자. 로그 버퍼내에서 트랜잭션 B의 COMMIT 레코드는 반드시 트랜잭션 A의 COMMIT 레코드 이후에 기록되게 된다. 따라서 트랜잭션 A의 COMMIT 레코드가 디스크에 반영되기 전에 시스템 오류가 발생되면, 트랜잭션 A 뿐만 아니라 트랜잭션 B도 모두 철회되므로 데이타베이스를 일관성 있는 상태로 만들 수 있다. 트랜잭션이 COMMIT되기 전에도 다른 트랜잭션이 같은 데이타를 액세스할 수 있으므로 동시성을 높일 수 있으며, 따라서 전체 시스템의 성능을 개선시킬 수 있다.

8.2.4 안정된 로그 버퍼의 사용

트랜잭션 종료시의 디스크 액세스 오버헤드를 피하기 위한 근본적인 해결책으로서 안정된 주기억 장치(stable main memory)[DeW84]를 사용하는 로그 버퍼(stable log buffer)가 도입될 수 있다. 안정된 로그 버퍼는 일반 주기억 장치에 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 주기억 장치를 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 수행되는 시간은 트랜잭션의 응답 시간에 영향을 미치지 않는다. 이러한 장점으로 인하여 많은 주기억 장치 데이타베이스 시스템에서 안정된 로그 버퍼를 사용하고 있다.

8.3 체크포인트시의 동작

체크포인트시 로그 레코드만을 디스크에 반영시키는 디스크 기반 데이타베이스 시스템의 WAL 기법과는 달리 주기억 장치 데이타베이스 시스템에서는 변경이 가해진 주기억 장치내의 데이타 부분을 모두 디스크에 반영시킨다[Leh87]. 그 이유는 가능한 디스크내에 최신의 데이타베이스를 저장해 놓음으로써 시스템 오류의 발생시 회복을 위한 재시동의 오버헤드를 줄이기 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].

8.4 회복을 위한 재시동시의 동작

주기억 장치 데이타베이스 시스템에서 시스템 오류가 발생되면, 주기억 장치내의 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 체크포인트 되었던 데이타베이스를 주기억 장치에 로드한 후 체크포인트 레코드 이후에 기록된 로그 레코드들을 적용함으로써 디스크 기반 데이타베이스 시스템 회복 기법의 재시동 동작과 유사한 형태로 진행된다.

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

IX. 결 론

회복 기능은 사용자 인터페이스 등의 형태로 드러

나지 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.

본 논문에서는 컴퓨터 시스템의 오류로 인하여 데이타베이스의 데이타가 손상된 경우의 문제점은 언급하였다. 또한 발생되는 오류를 트랜잭션 오류, 시스템 오류, 미디어 오류의 세 가지로 분류하고, 회복할 수 있는 여러가지 회복 기법을 소개하였다. 또한 주기억 장치 데이타베이스 시스템에서의 회복 기법의 특징에 대하여 언급하였다.

본 논문에서 언급된 내용들은 대부분 이해 위주의 쉬운 개념적인 내용들만을 다루었다. 그러나 실제로 구현함에 있어서는 여러가지 문제점들이 많이 대두되며, 여러가지 오류에 대하여 완벽한 회복 기능을 갖춘 데이타베이스 관리 시스템을 만든다는 것은 상당히 힘든 작업이 된다. 회복 기능은 효율에 관련된 문제라기 보다는 기능에 관련된 문제이기 때문이다. 예를 들어, 한 경우에 대해서라도 오류로부터 올바르게 회복되지 못한다면, 그 회복 관리자의 구현은 실패로 평가된다.

현재까지 국내에서도 데이타베이스 관리 시스템에 대한 많은 연구가 계속되고 있으나 상용화된 대형 데이타베이스 관리 시스템은 아직 발표되지 않고 있다. 그러나 점차로 쌓여가는 기술 축적으로 인하여 향후에는 많은 데이타베이스 관리 시스템이 출현될 것이다. 특히 모든 오류에 대해서도 회복할 수 있는 기능을 갖춘 훌륭한 데이타베이스 관리 시스템의 출현을 바라며, 많은 분들의 관심과 노력을 기대한다.

참 고 문 헌

[Ber87] Bernstein, P. A., Hadzilacos, V., and Goodman, N., *Concurrency Control and Recovery in Database Systems,* Addison-Wesley Publishing Company, 1987.

[Dat84] Date, C. J., *An Introduction to Database Systems,* Vol. II, Addison-Wesley Publishing Company, 1984.

[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.

[Gra81] Gray, J. *et al.*, "The Recovery Manager of the System R Database Manager," *ACM Computing Surveys,* Vol. 13, No. 2, pp. 223~242, June 1981.

[Hae83] Haeder, T. and Reuter, A., "Principles of Transaction-Oriented 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.

[Lor77] Lorie, R. A., "Physical Integrity in a Large Segmented Database," *ACM Trans. on Database Systems,* Vol. 2, No. 1, pp. 91~104, Mar. 1977.

[Moh89] Mohan, C. *et al.*, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rolbacks 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, Department 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.

[Wha87] Whang, K. Y. *et al.*, "Office-by-Example: An Integrated Office System and Database Manager," *ACM Trans. on Office Information Systems*, Vol. 15, No. 4, pp. 393~427, Oct. 1987.

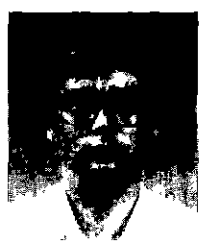
[Wha90] Whang, K. Y. and Krishnamurthy, R., "Query Optimization in a Memory-Resident Domain Relational Calculus Database Systems," *ACM Trans. on Database Systems*, Vol. 15, No. 1, pp. 67~95, Mar. 1990.

[Wha92] 황규영, "데이타베이스 시스템에서의 파손 회복 기법," 데이타베이스 연구회지, 한국정보과학회, Vol. 8, No. 2, pp. 139~156, 1992.

황 규 영

1973 서울대학교 전기과 졸업(B.S.)
1975 한국과학기술원 전기 및 전자공학과 1회 졸업(M.S.)
1982 Stanford University (전산학, M.S.)
1983 Stanford University (전산학, Ph D)
1975 ~1978 국방과학연구소 선임연구원
1983 ~1990 IBM T.J. Watson Research Center, Research Staff Member
1990 ~현재 한국과학기술원 전산학과 부교수, 인공지능연구센터 데이타베이스 및 멀티미디어 연구실장
1991 Visiting Professor, Computer Science Department, Stanford University
1992 Visiting Associate Professor College of Computing, Georgia Institute of Technolgoy
Editor, The VLDB Journal
Editor: Distributed and Parallel Databases: An International Journal
Associate Editor: The IEEE Data Engineering Bulletin 1990~1992.
관심 분야 : 멀티미디어 데이타베이스, 객체지향 데이타베이스, 인익 데이타베이스 공학 데이터베이스, 사무자동화, CASE, 전문가 시스템.

김 상 욱



1989 서울대학교 컴퓨터공학과 졸업(B.S.)
1991 한국과학기술원 전산학과 졸업(M.S.)
1991 ~현재 한국과학기술원 전산학과 박사과정, 인공지능연구센터 데이타베이스 및 멀티미디어 연구실
관심 분야 : 다중키 액세스, 질의 최적화, 성능 분석, 공간 데이타베이스, 파손 회복 기법