객체지향 데이타베이스 관리 시스템에서의 부분철회

김 원 영**°, 이 영 구*, 황 규 영*
*한국과학기술원 전산학과/첨단정보기술 연구센터 **한국전자통신연구원 실시간DBMS팀
{yklee,kywhang}@mozart.kaist.ac.kr wykim@etri.re.kr

Partial Rollback in Object-Oriented Database Management Systems

Won-Young Kim**°, Young-Koo Lee*, and Kyu-Young Whang*
*Department of Computer Science and Advanced Information Technology Research Center(AITRC) KAIST
**Real-time DBMS team ETRI

요 약

DBMS에서의 부분철회는 트랜잭션 전체를 철회하지 않고 일부분만을 철회할 수 있는 유용한 기능으로 많은 RDBMS에서 지원되고 있다. 그러나, RDBMS와는 달리 OODBMS에서는 객체 버퍼와 페이지 버퍼로 구성되는 이중 버퍼를 유지하므로 페이지 버퍼만을 유지하는 RDBMS에서 사용하던 기존의 부분철회 방식을 그대로 이용할 수 없다. 따라서, 이러한 이중 버퍼 내의 데이타를 효과적으로 부분철회할 수 있는 새로운 회복 방법이 필요하다. 본 논문에서는 이중 버퍼구조를 사용하는 OODBMS를 위한 네가지 부분철회 방식들을 제안하고 비교한다.

1 서 론

세이브포인트 개념을 이용한 부분철회는[2][6] 트랜잭션 전체를 철회하지 않고 오류가 발생한 트랜잭션의 일부만을 철회하는 기능이다. 이 기능을 이용하면 트랜잭션의 진행 중에 권한이 없는 데이타의 읽기 시도, 교착 상태 또는 존재하지 않는 데이타의 요구와 같은 오류가 발생한 경우 오류가 발생한 이전의 세이브포인트로 상태를 되돌릴 수 있다

System R[1], DB2[6], Oracle, Sybase UniSQL 등의 많은 DBMS에서 부분철회를 직접 또는 내부적으로 지원하고 있다. 이와 같이 부분철회를 지원하는 DBMS들은 개발된 바 있으나, 부분철회를 지원하는 방법론에 관한 연구 발표는 그 수가 미미하고 주로 RDBMS에 국한되어 왔다.

OODBMS는 CAD나 멀티미디어 저작과 같은 사용자와의 상호작용이 많은 응용들에서 그 하부구조로 사용되고 있고, 이를 위하여 OODBMS에서는 부분철회의 지원이 매우 필수적이다. 대부분 OODBMS에서는 RDBMS와 달리 객체들에 대한 빠른 액세스를 위해 페이지 버퍼와 객체 버퍼로 구성된 이중 버퍼를 사용하고 있다[3][4]. 따라서, 이러한 구조에서는 페이지 버퍼뿐만 아니라 객체 버퍼에 걸쳐 존재하는 객체들을 어떻게 부분철회할 것인가에 대한 체계적인 연구가 필요하다.

본 논문에서는 이중 버퍼를 가진 OODBMS를 위한 네가지 부분철회 방식들— 페이지 버퍼 기반 부분철회 방식, 객체 버퍼 기반 부분철회 방식, 소프트로그를 이용한 이중 버퍼 기반 부분철회 방식과 새도우를 이용한 이중 버퍼 기반 부분철회 방식 —을 제안하고 각 방식들의 장단점을 비교한다.

논문의 구성은 다음과 같다. 제 2장에서는 RDBMS에서의 부분철회를 살펴 보고 제 3장에서는 OODBMS에서의 이중 버퍼와 이중 버퍼가 부분철회에 미치는 영향을 논의한다. 제 4장에서는 OODBMS에서의 네가지 부분철회 방식들을 제안하고 장단점을 비교한다. 제 5장에서는 결론을 내린다

2 RDBMS에서의 부분철회

RDBMS에서는 부분철회시 트랜잭션에 의해 갱신된 데이타베이스의 상태 뿐만 아니라 진행 중인 트랜잭션의 실행 상태를 복원한다. 먼저, 트랜잭션에 의해 갱신된 데이타베이스는 페이지 버퍼와 디스크에 걸쳐 존재하는 갱신된 데이타들로 이루어진다. RDBMS에서는 데이타베이스의 상태를 세이브포인트 시점으로 되돌리기 위해서 로그를 이용한다. 진행 중인 트랜잭션의 실행 상태는 트랜잭션이 수행되는 동안에 메모리 상에서 점유한 자원인 로크, 커서, 그리고 볼륨이나 화일에 대한 액세스 정보 등으로 이루어 진다[2]. 이러한 자료구조들은 트랜잭션의 진행과 함께 갱신이 자주 발생하고 데이타베이스에 비해 자료구조의 크기가 상당히 작다는 특징을 가진다. 이러한 특징으로 인해 RDBMS에서는 세이브포인트 설정시 해당 트랜잭션에 대한 실행 상태의 스냅샷(snapshot)을 저장해 두었다가 해당 세이브포인트로의 철회시 저장해 두었던 스냅샷을 복구한다[2].

3 OODBMS에서의 이중 버퍼

OODBMS에서는 페이지 버퍼와 객체 버퍼로 구성되는 이중 버퍼를 유지한다[5]. 페이지 버퍼는 디스크 입출력을 최소화하기 위한 것으로, 디스크에서와 동일한 형태로 객체들을 페이지 단위로 관리한다. 객체 버퍼는 포인터 스위즐링(pointer swizzling)[4]을 통하여 객체의 빠른 액세스를 지원하기 위한 것으로, 페이지 버퍼로부터 읽어들인 객체들을 메모리내에서 포인터를 사용하여 직접 액세스할 수 있도록 변환된 형태로 관리한다.

OODBMS에서는 데이타베이스가 디스크, 페이지 버퍼, 그리고 객체 버퍼에 산재되어 있으므로 부분철회시 이들 모두의 상태를 회복해야 한다. 특히, 객체 버퍼의 부분철회는 파손 회복이나 트랜잭션 철회와 달리 추가적인 조치가 필요하다. 파

† 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원과 웹기반 정보검색 OODBMS를 위한 Web-DBMS통합기술의 개발과제를 통하이 과학기술부의 지원을 받았음

손 회복이나 트랜잭션 철회 이후에는 해당 트랜잭션이 더 이상 진행하지 않으므로 그 트랜잭션이 독점적으로 사용하던 객체 버퍼내의 상태를 단순히 무시하면 된다. 그러나, 부분철회 이후에는 해당 트랜잭션이 계속 진행하면서 객체 버퍼를 사용하므로 부분철회 연산은 객체 버퍼의 상태도 세이브포인트 시점으로 되돌려야 한다.

OODBMS의 객체 버퍼에서 수행된 갱신은 아직 로그가 기록되지 않을'뿐만 아니라 디스크에도 반영되지 않은 상태이다. RDBMS에서의 프로그램의 실행 상태와 같이 세이브포인트 시점의 스냅샷을 메모리에 저장해두었다가 부분철회시 복구하는 방식을 객체 버퍼에 적용하는 것을 고려할 수 있다. 그러나, 이 방식은 세이브포인트마다 객체 버퍼의 스냅샷을 유지하기 위한 메모리 오버헤드가 너무 크기 때문에 바람직하지 않다. 이러한 요인을 고려하여 본 논문에서는 객체 버퍼를 포함한 데이타베이스의 상태를 회복할 수 있는 네가지 부분철회 방식을 제시한다.

4 OODBMS에서의 부분철회 방식들

본 장에서는 OODBMS에서 부분철회를 지원하기 위한 네가지 방식들을 제안한다. 각 부분철회 방식의 기본 개념, 동작 방식 및 장단점을 차례대로 기술한다.

4.1 단일 버퍼 기반 부분철회 방식

단일 버퍼 기반 부분철회 방식은 부분철회를 위한 회복 데이타의 관리 및 회복 과정이 페이지 버퍼나 객체 버퍼 중 한쪽에서 수행된다. 단일 버퍼 기반 부분철회 방식은 회복이 수행되는 버퍼에 따라서 페이지 버퍼 기반 부분철회 방식*(page buffer based partial rollback method. PB)*과 객체 버퍼 기반 부분철회 방식*(object buffer based partial rollback method: OB)*으로 구분한다.

4.1.1 PB

PB는 부분철회를 위하여 세이브포인트 시점에 객체 버퍼에서의 갱신된 객체들을 페이지 버퍼로 강제적으로 반영시키고 부분철회시 페이지 버퍼의 상태만을 회복하는 방식이다. 부분철회시 객체 버퍼에 있는 객체들은 지정된 세이브포인트 이후에 변경되었을 가능성이 있으므로 모두 삭제된다. 페이지 버퍼의 상태의 회복은 로그를 이용한다 차후 사용되는 소프트 로그와 구별하기 위해 이를 하드 로그라 부른다.

PB는 객체 버퍼를 갖지 않는 RDBMS의 부분철회 방식 위에 세이브포인트 설정 연산시 객체 버퍼의 플러시(flush)를 추가한 방식이다 이 방식은 별도의 회복 데이타 없이 하드 로그에 의존하여 OODBMS에서의 부분철회를 지원하므로 그 구현이 단순하다. 그러나, 세이브포인트 설정 연산마다 객체 버퍼의 갱신된 객체들을 모두 페이지 버퍼에 강제 반영시켜야 한다. 이 오버헤드는 객체 교체의 횟수를 증가시킬 뿐만 아니라 페이지 버퍼에서의 교체를 야기하게 되어.디스크 입출력 횟수를 증가시킨다. 앞으로 이를 강제 플러시 오버헤드*(forced flush overhead)*라고 부른다. 세이브포인트는 한 트랜잭션 안에서 사용자의 필요에 따라 빈번하게 설정될 수 있으므로 강제 플러시 오버헤드는 심각한 성능의 저하를 야기시킬 수 있다. 뿐만 아니라 부분철회시 객체 버퍼에 있었던 객체들은 모두 삭제되어 객체 버퍼의 버퍼링의 효과가 없어지게 되므로 부분철회 이후에는 객체의 스왑-인이 더 많이 요구된다.

4.1.2 OB

OB는 PB의 강제 플러시 오버헤드 및 객체 버퍼의 버퍼링 효과 감소 문제를 해결하기 위한 것으로서 객체 버퍼의 상태를 직접 회복하는 방식이다. OB에서는 이를 위해 객체 버퍼내에서 발생한 모든 갱신을 기록한 소프트 로그를 유지한다. 물론, 페이지 버퍼내에서 발생한 갱신을 기록한 하드 로그가 이와는 별도로 유지되지만 부분철회를 위해서는 사용되지 않는다.

부분철회시 객체 버퍼에 없는 객체들을 UNDO하기 위해서는 페이지 버퍼로부터 해당 객체를 스왑-인해야 한다. UNDO가 완료된 로그 레코드들은 소프트 로그에서 삭제된다. 부분철회시 페이지 버퍼에서는 UNDO가 수행되지 않으며 객체 버퍼에서의 UNDO로 인해 스왑-아웃된 객체로 인한 갱신은 하드 로그에 일반 갱신 연산으로 기록된다.

OB는 PB의 오버헤드를 해결하지만, 부분철회시 UNDO된 로그 레코드를 삭제함에도 불구하고 소프트 로그를 유지하기 위한 메모리 오버헤드가 매우 심각하다는 단점이 있다. 뿐만 아니라, 부분철회시 현재 객체 버퍼에 존재하지 않는 객체를 UNDO하기 위해서는 해당 객체를 스왑-인해야 하고 이로 인해 다른 객체가 스왑-아웃되어야 한다. 따라서, OB에서 객체 버퍼가 충분히 크지 않은 경우에는 부분철회시 객체 버퍼로부터 교체되는 객체의 수가 증가하게 되고, 이 결과 페이지 버퍼에서의 교체 및 디스크 입출력의 횟수가 증가하게 될 뿐만 아니라 하드 로그도 증가하게 되어 성능이 저하된다

4.2 이중 버퍼 기반 부분철회 방식

단일 버퍼 기반 부분철회 방식이 가지는 성능 저하의 근본적인 원인은 이중 버퍼에 걸쳐있는 갱신된 객체들을 세이브포인트 시점이나 부분철회 시점에 한쪽의 버퍼로 옮겨 주어야 하는 데서 비롯된다. 이중 버퍼 기반 부분철회 방식(dual buffer based partial rollback method:DB)은 이러한 오버헤드를 피하기 위해 이중 버퍼의 갱신된 객체들을 가능한 한 현재 있는 각각의 버퍼에서 관리하고 회복한다. 객체 버퍼에서의 회복을 위해서는 객체 버퍼내의 갱신을 기록한 회복 데이타를 유지하고, 페이지 버퍼에서의 회복을 위해서는 하드 로그를 이용한다.

DB에서 객체는 세이브포인트 시점과 부분철회 시점에 걸친 갱신 여부에 따라서 그림 1과 같이 세가지 유형으로 나누고 다른 회복 방식을 적용한다. 유형 1의 객체는 세이브포인트 시점 이전에만 갱신된 경우로서, 부분철회시 이를 UNDO하지 않는다. 유형 2의 객체는 세이브포인트 시점 이후에만 갱신된 경우로서, 부분철회 시점에 객체 버퍼에 그대로 존재하는 경우와 페이지 버퍼로 스왑아웃된 경우가 다르게 처리된다. 전자의 경우에는 갱신이 객체 버퍼에만 반영되어 있으므로 객체 버퍼에서 해당 객체를 단순히 삭제하는 반면, 후자의 경우에는 스왑-아웃되어 페이지 버퍼에 반영될 때 그 갱신을 기록한 하드 로그의 로그 레코드를 UNDO함으로써 이전의 값을 복원한다. 유형 3의 객체는 세이브포인트 전후에 각각 갱신이 있었던 경우로서, 이 경우에는 객체 버퍼의 회복 데이타를 이용하여 회복을 수행한다.

임의의 세이브포인트에 대한 객체의 유형을 구분하기 위해 세이브포인트 시점에 객체 버퍼에 갱신된 상태로 있었던 객체들의 리스트인 세이브포인트 변경 객체 리스트*(savepoint updated object list:SUOL)*를 유지한다. 어떤 세이브포인트가 설정된 이후 해당 SUOL에 속한 객체가 객체 버퍼에서 더 이상 갱신되지 않았다면 이 객체는 유형 1이 된다. 이 SUOL에

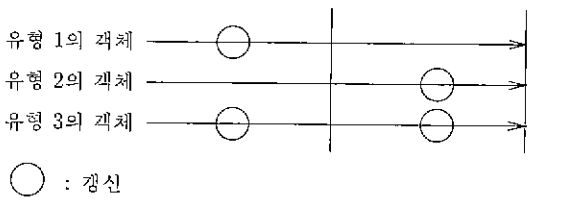

그림 1: 객체의 유형.

속한 객체가 스왑-아웃되지 않고 갱신되었다면 유형 3이 되고 이 SUOL에 속하지 않는 객체가 갱신된다면 유형 2의 객체가 된다.

객체 버퍼에서 관리하는 회복 데이타에 따라서 소프트 로그를 이용한 이중 버퍼 기반 부분철회 방식*(dual buffer based partial rollback method using soft log:DB-SL)*과 섀도우를 이용한 이중 버퍼 기반 부분철회 방식*(dual buffer based partial rollback method using shadows:DB-SO)*이라고 부른다. 앞으로 이 두가지 방식을 자세히 살펴 본다.

4.2.1 DB-SL

DB-SL은 주어진 세이브포인트에 대해 유형 3인 객체에 대해서 소프트 로그를 이용하여 부분철회를 수행하고 그 이외의 객체에 대해서는 객체 버퍼에 있는 객체를 삭제하거나 하드 로그를 이용하여 부분철회를 수행하는 방식이다.

DB-SL은 OB와 같이 소프트 로그로 인한 오버헤드를 가지지만 하드 로그를 이용한 회복을 병행하기 때문에 다음과 같은 큰 차이가 있다. 첫째, DB-SL에서는 SUOL에 속한 객체들에 대해서만 소프트 로그를 유지하므로 이 방식의 메모리 오버헤드가 OB에 비해서 매우 작다. 둘째, DB-SL에서는 소프트 로그에 의해 UNDO되는 객체의 수가 작으므로 이를 위해 객체 버퍼에 없는 객체들을 페이지 버퍼로부터 스왑-인하는 오버헤드도 OB에 비해서 상당히 작다. 그러나, DB-SL은 여전히 소프트 로그를 유지하기 위한 메모리 오버헤드를 가지며, 부분철회시 UNDO되는 객체를 위하여 페이지 버퍼로부터 객체 버퍼로의 객체 교체가 요구된다.

4.2.2 DB-SO

DB-SO에서는 DB-SL의 부분철회시 객체의 교체를 피하기 위해서 소프트 로그 대신에 섀도우들을 이용하여 부분철회를 수행한다. 섀도우는 어떤 시점에 한 객체의 사본(copy)이다. 임의의 SUOL에 속한 객체가 최초로 갱신될 때 이 객체에 대한 섀도우를 메모리에 생성한다.

DB-SO에서는 DB-SL과 비교하면 다음과 같은 세가지 장점을 가진다. 첫째, DB-SO에서는 부분철회시 SUOL에 속한 객체가 객체 버퍼에 존재하지 않더라도 메모리에 있는 섀도우를 그대로 객체 버퍼에 복사하기 때문에 페이지 버퍼로부터 해당 객체를 스왑-인하는 과정이 필요하지 않다. 둘째, DB-SL에서는 SUOL에 속한 객체에 대한 다수의 갱신을 모두 소프트 로그에 기록하는 반면, DB-SO에서는 첫 갱신에 대해서만 섀도우를 유지하므로 DB-SL에 비해 적은 메모리 오버헤드를 가진다. 세째, DB-SL의 메모리 오버헤드는 SUOL에 속한 객체들의 갱신의 수에 따라 매우 가변적이고 예측할 수 없는 반면, DB-SO의 메모리 오버헤드는 최악의 경우 각 SUOL에 속한 객체들의 총 수만큼의 섀도우들을 가짐으로 항상 예측이 가능하다.

5 결론

CAD나 멀티미디어 저작과 같은 OODBMS 응용은 사용자와의 상호 작용이 많이 있으나 이를 위해 필수적으로 지원해야 할 OODBMS에서의 부분철회 기능에 대한 연구는 현재 거의 전무한 상태이다. OODBMS에서는 RDBMS와 달리 이중 버퍼 구조를 사용하기 때문에 트랜잭션에 의해 갱신된 데이타베이스가 페이지 버퍼 뿐만 아니라 객체 버퍼에 산재되어 있으므로 OODBMS에서의 부분철회는 이중 버퍼에 산재되어 있는 데이타베이스의 상태를 사용자가 원하는 세이브포인트 시점으로 회복시켜주어야 한다.

본 논문에서는 OODBMS에서의 부분철회를 지원하는 네가지 방식을 제안하였다. PB는 RDBMS에서 사용되었던 기존의 부분철회 방식을 단순히 확장하여 구현할 수 있으므로 구현이 단순하다는 장점을 가지나 세이브포인트 연산이 빈번하게 수행되는 경우 강제 플러시 오버헤드로 인하여 성능이 떨어지는 단점이 있다. OB는 객체 버퍼가 충분히 큰 경우 좋은 성능을 보일 수 있지만 회복용 데이타인 소프트 로그를 유지하기 위한 메모리 오버헤드가 심각하다는 단점이 있다. DB-SL과 SB-SO는 그 구현 방법이 다소 복잡하지만 세이브포인트의 수와 객체 버퍼의 크기에 큰 영향을 받지 않는 좋은 성능을 보일 것이다. 특히, DB-SO는 DB-SL 과 비교하여 더 나은 성능을 보일 것이다. 또한, 이 두 방식의 메모리 오버헤드는 객체 버퍼 기반 부분철회 방식에 비해 상당히 작을 것이다.

부분철회 기능은 사용자와 상호 작용이 많은 시스템에 있어서는 매우 필수적인 기능이다. 따라서, 제안된 부분철회 방식들은 OODBMS 뿐만 아니라 이중 버퍼를 사용하는 시스템의 설계 및 구현에 많은 도움이 될 수 있을 것이라 생각된다.

참고문헌

[1] 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.

[2] Gray, J. and Reuter, A., *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993.

[3] Kemper, A. and Kossman, D., "Adaptable-Pointer Swizzing Strategies in Object Bases," In *Proc Tenth Int'l Conf. on Data Engineering*, IEEE, pp. 155–162, Apr. 1993.

[4] Kim, W., *Introduction to Object-Oriented Databases*, Computer System Series, The MIT Press, 1st ed., 1990.

[5] Loomis, M.E.S., *Object Databases: The Essentials*, Addison-Wesley, 1995.

[6] Mohan, C et al., "ARIES. A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollback Using Write Ahead Logging," *ACM Trans. on Database Systems*, Vol. 17, No. 1, pp. 94–162, Mar. 1992.