공간 데이터베이스 시스템의 공간 데이터 크기에 의한 선별적 회복 기법

김 명 근*, 김 성 희*, 조 숙 경*, 김 재 홍**, 배 해 영*
* 인하대학교 전자계산공학과
** 영동대학교 컴퓨터공학과

g2011494@inhavision.inha.ac.kr

A selective recovery scheme considering the size of spatial object in spatial database system

Myung-Keun Kim*, Sung-Hee Kim*, Sook-Kyoung Cho*, Jae-Hong Kim**, Hae-Young Bae*
* Dept. of Computer Science, Inha University
** Dept. of Computer Science, Youngdong University

요약

본 논문에서는 공간 데이터베이스 시스템의 공간 데이터 크기에 의한 선별적 회복 기법을 제안하고자 한다. 기존의 대용량 데이터베이스를 위한 회복 관리 기법에는 쉐도우 기법과 변형된 로그 기법이 쓰여지고 있다. 쉐도우 기법은 물리적 응집과 잠금 경쟁 문제가 있으며, 변형된 로그 기법은 공간 데이터의 가변 크기 특성에 대한 고려가 없기 때문에 디스크 입출력이 빈번히 발생하는 문제가 있다. 본 논문에서 제안하는 회복 기법은 공간데이터 크기에 따라 각각 다른 회복 기법을 적용하여, 로그 파일에 기록되는 로그의 양을 줄이며, 또한 트랜잭션 철회시 회복 연산으로 인한 시스템의 부하를 줄이는 장점을 갖는다.

1. 서론

기존의 DBMS 는 공간 데이터가 항상 크다는 가정 하에서 설계하였다[1,3]. 그러나 공간 데이터는 다양한 타입과 데이터 크기를 갖는다. 이러한 특성을 갖는 공간 데이터에 대한 회복 기법으로 기존의 대용량을 위한 회복 기법은 적합하지 않다.

기존의 대용량 데이터를 지원하는 DBMS 에서는 대용량 데이터에 대한 회복 기법으로 쉐도우 기법[1,3,5]과 변형된 로그 기법[1]이 많이 쓰여지고 있다. 쉐도우 기법은 갱신 연산이 계속되면 물리적 응집도가 떨어지며, 쉐도우 페이지들을 관리하기 위한 설명자에 잠금을 걸어야만 한다. 하지만 이러한 잠금은 전체 데이터에 잠금을 요구하는 것과 같은 효과를 갖기 때문에 동시성을 저하시킨다. 변형된 로그 기법 중에는 철회 로그만을 기록하는 회복 기법이 있다. 이러한 회복 기법에서 버퍼 관리자는 강제(force) 정책을 사용하여 재 수행 연산이 필요 없도록 한다. 하지만 이러한 회복 기법에서 크기 작은 공간 데이터를 페이지에 기록하기 위해 디스크 입출력이 빈번하게 발생하여 시스템의 성능을 크게 저하시킬 것이다[4].

본 논문에서는 공간 데이터의 크기에 따라서 서로 다른 로깅 방식을 사용하여 효과적인 공간 데이터의 회복을 지원하고자 한다. 공간 데이터의 크기에 따라 세가지(SMALL, MIDDLE, LARGE)로 분류하며 각각 다른 회복 기법을 적용한다. (1)한 페이지보다 작은 데이터는(SMALL) ARIES 회복 기법을 사용한다. 이는 데이터의 크기가 한 페이지를 넘지 않는 비공간 데이터와 같은 처리 과정이다. (2)한 페이지보다 크지만 버퍼 정책을 사용할 수 있을 정도의 수 페이지 크기를 가진 데이터에(MIDDLE) 대해서는

* 본 연구는 정보통신부의 대학 S/W 연구센터 지원 사업의 연구 결과임.

본 논문에서 새롭게 제안한 APR (Anchored physical log record)을 사용하여 로그 버퍼에 기록되는 로그의 양을 줄이며, 또한 (3) 버퍼에 유지가 힘든 데이터(LARGE)에 대해서는 트랜잭션 철회나 시스템 재시작을 위한 논리적인 로그만을 기록하고 직접 디스크에 기록하는 정책을 사용한다.

본 논문에서 제안한 MIDDLE 데이터를 위한 APR 은 데이터 자체를 로그 레코드에 포함하지 않고, 회복 세그먼트의 페이지를 가리키는 포인터만을 유지하기 때문에 로그 버퍼 풀로 인한 디스크 입출력이 줄어들며, 트랜잭션 철회 시 적은 양의 보상 로그가 기록되기 때문에 트랜잭션 철회로 인해 기록될 로그의 양이 줄여, 회복의 부하를 줄일 수 있다.

논문의 구성은 다음과 같다. 2 장에서는 기존 저장 시스템들의 대용량 데이터 회복 기법과 그 문제점을 설명한다. 3 장에서는 레코드 구조와 회복 세그먼트의 개념에 대해서 설명하고, 본 논문에서 제안된 회복 기법을 상세히 설명한다. 마지막으로 4 장에서는 결론에 대해서 기술한다.

2. 관련연구

대부분 시스템에서는 모든 공간 데이터가 대용량이라고 간주 하였기 때문에 대용량 데이터에 대한 회복 기법을 주로 사용하였다. 본 장에서는 기존의 대용량 데이터 관리 모듈과 회복 기법에 대해 간략하게 설명하고 각 기법의 문제점을 제시한다.

ORION[2]은 시스템 고장에 대한 회복만을 지원하고 디스크 고장에 대한 회복 기능은 지원하지 않는다. 다시 말해, ORION 은 시스템 고장이나 사용자에 의한 트랜잭션 고장에 대한 회복만을 지원하고 있다. 동시 수행성을 위해서는 그림자 페이지 기법보다는 로그 기법을 선택하였고 구현이 복잡하지 않도록 철회 로그만을 유지하도록 하고 있다. 철회 로그만을 사용하는 경우, 버퍼 관리기는 갱신된 객체가 있는 페이지를 트랜잭션이 끝날 때 디스크에 반영한다. 즉 버퍼 관리기는 강제 정책(FORCE)을 사용한다.

Starburst[3] 롱 필드 관리자는 롱 필드 데이터가 디스크 페이지에 어떻게 저장되어 있는지 설명하는 롱 필드 설명자(long field descriptor)와 실제 데이터가 저장되는 버디 세그먼트들로 구성되어 있다. 버디 세그먼트라고 불리는 가변 디스크 익스텐트는 롱 필드 데이터를 저장하기 위해 버디 공간(buddy space)으로부터 할당 받으며, 이진 버디 시스템을 이용하여 할당되기 때문에 2 의 급수 크기를 갖는다. Starburst 의 롱 필드 관리자는 대용량 데이터를 효율적으로 할당하고 반환하며 순차 접근에 용이하도록 설계되어 있다. 그러나 대용량 데이터에 대한 임의 접근 시 성능이 떨어지며 삽입, 삭제등의 갱신 연산 수행 시 많은 부가 비용을 발생시키는 단점이 있다.

EXODUS[1]는 대용량 데이터의 회복을 위하여 쉐도우와 로깅 방법을 함께 사용한다. 갱신된 내부 페이지들과 잎 블록들을 루트 페이지까지 쉐도우 시킨다. 갱신 연산들은 대용량 데이터에 대한 이전 헤더를 변경된 새로운 헤더로 덮어씀으로써 반영된다. 마지막 단계로 루트 페이지를 갱신하게 되는데 이전에 갱신된 페이지들을 디스크로 반영하며, 갱신 연산의 이름과 매개변수(parameter)들을 로그에 기록한다. EXODUS 는 B^+-tree 형태의 인덱스를 사용하기 때문에 회복 기법도 B^+-tree 가 갖는 문제점 그대로 갖고 있다.

3. 공간 객체 크기에 의한 선별적 회복 기법

본 장에서는 선별적 회복 기법을 설명하기 앞서 회복 기법을 위한 자료구조를 설명하고, 공간 데이터 크기에 따른 선별적 회복 기법을 설명한다.

3.1 로그 레코드 구조와 회복 세그먼트

[표 1]에서는 로그 레코드에 저장되는 필드와 각 로그 타입마다 저장되는 구조를 나타낸다. 저장되는 필드에는 TxID(트랜잭션 식별자), LSN(로그 순차 번호), PrevLSN(바로 이전에 기록된 로그 레코드 순차 번호), RID/PID(레코드 식별자/페이지 식별자), ARID(회복 세그먼트를 가리키는 페이지 식별자), Image(회복에 필요한 데이터)등이 있다. 또한 로그 레코드 종류에는 Physical(물리적인 로그 레코드), Logical(논리적인 로그 레코드), BTR/ETR(트랜잭션의 시작/종료를 나타내는 로그), BCR(검사점 시작 시 기록되는 로그), ECR(검사점 종료 시에 기록되는 로그),

CLR(취소 연산에 대한 보상 로그), APR(MIDDLE 타입의 공간 데이터에 대한 회복을 지원하기 위해서 사용하는 로그), APRCLR(APR 의 취소 연산에 대한 보상 로그)등이 있다. 각 로그 레코드가 저장하고 있는 필드들은 [표 1] 에서 보여주고 있다.

[표 1] 로그 레코드 타입에 따른 저장 내용

Log Type	TxID	LSN	PrevLSN	RID/PID	ARID	Image
Physical	O	O	O	O	×	O
Logical	O	O	O	O	×	×
BTR	O	O	NULL	×	×	×
ETR	O	O	O	×	×	×
BCR	×	O	×	×	×	×
ECR	×	O	×	×	×	O
CLR	O	O	O	O	×	O
APR	O	O	O	O	O	×
APRCLR	O	O	O	O	O	×

회복 세그먼트는 MIDDLE 데이터의 회복을 위한 저장 구조이다. SMALL 데이터에 대한 저장 구조는 페이지를 기본으로 하고, 클러스터링을 위해 익스텐트 개념을 사용하고 있다. 그러나, 이 같은 저장 구조로는 많은 페이지들로 구성되는 MIDDLE 데이터의 물리적인 연속성을 보장할 수 없으며 또한 BLOB 페이지의 크기가 익스텐트 하나의 크기이기 때문에 회복 세그먼트의 페이지는 기본 할당 단위를 익스텐트로 한다. 회복 세그먼트는 MIDDLE 데이터 로깅 시 갱신 전 이미지와 갱신 후 이미지를 저장하며, MIDDLE 데이터의 로그 레코드는 회복 세그먼트의 페이지 식별자만을 유지 함으로써 로그 버퍼에 기록될 로그의 양을 줄이며, 로그 버퍼 풀로 인해 생기는 디스크 입출력 빈도를 줄일 수 있다.

3.2 트랜잭션의 로그 기록 과정

본 절에서는 본 논문에서 새롭게 제안하는 APR 로그 기록과정에 대해 서술한다. [알고리즘 1]은 갱신 연산이 발생했을 경우 트랜잭션의 로그 기록 과정을 설명한 알고리즘이다.

회복 세그먼트중 사용 가능한 페이지를 할당 받고, 로그의 종류가 APR 이라면 회복 세그먼트에 갱신 전 데이터와 갱신 후 데이터를 기록하여 로그 버퍼가 풀 되지 않게 한다. 각 BLOB 페이지는 RecPID 를 유지하며, 이러한 RecPID 는 BLOB 페이지의 로그로써 기록된 회복 세그먼트의 페이지 식별자가 기록된다. APR 로그 레코드를 하나 만들고, APR.ARID 에 회복 세그먼트의 페이지 식별자를 기록한다. APR.ARID 는 트랜잭션 철회와 시스템 붕괴 시 회복을 위해서 필요하다. 트랜잭션 철회 시 BLOB 페이지는 디스크에 반영되었지만 트랜잭션이 철회를 하는 경우이며 이러한 경우에 APR.ARID 이 가리키는 회복 세그먼트의 페이지에 있는 갱신 전 데이터를 이용해 회복한다. 또한 트랜잭션이 완료한 상태에서 트랜잭션이 갱신한 페이지가 버퍼에 유지되고 있지만 디스크에 기록이 안 되는 경우가 있으며 이러한 경우는 트래잭션 완료 시에 로그 버퍼에 있는 내용이 로그 파일로 기록되었기 때문에 기록된 APR.ARID 의 회복 세그먼트의 갱신 후 데이터를 이용하여 데이터베이스 회복을 한다.

[알고리즘 1] 트랜잭션의 로그 기록 과정

```
LSN Write_Log_Record
(LogType, BeforeImage, AfterImage, Page)
{
    if(LogType == APR)   {
        RecPage = GetFreeRecPage();
        Write(RecPage, BeforeImage,
              AfterImage);
        Page.RecPID = RecPage.PID;
        APR = new Log(LogType)
        APR.ARID = RecPage.PID;
        AddRecPageList(RecPage.PID);
        WriteLog(APR);
    }
    else {
        LogRec = new(LogType, AfterImage,
                     BeforeImage);
        WriteLog(LogRec);
    }
}
```

3.3 회복 과정

철회 과정은 로그 레코드에 기록된 PrevLSN 을 따라가며 취소 연산을 한다. PrevLSN 은 트랜잭션이 기록한 로그 레코드들에 대한 역방향 링크드 리스트(backward linked list) 이다. Logical 과 Phsical 로그 레

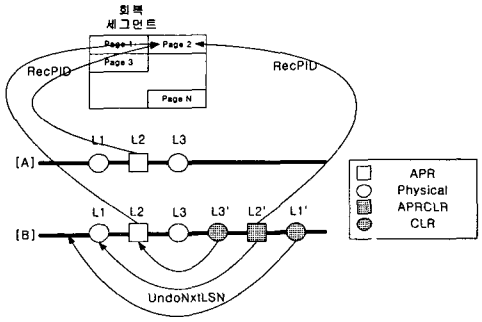
코드가 취소 가능하다면 취소연산을 행하며 physical 로그는 보상 로그로써 CLR 을 로그에 기록한다. CLR 과 APRCLR 은 physical 로그나 APR 에 대한 보상 로그 레코드이기 때문에 취소 연산을 할 필요가 없으며 취소 연산할 당시 가지고 있던 PrevLSN 인 UndoNxtLSN 을 사용하여 다음 취소할 로그 레코드를 선정한다.

[알고리즘 2] 트랜잭션 철회 알고리즘

```
TranX_Abort(TxID)
{
UndoNxt = TranX_Table[TxID].UndoNxtLSN;
    while(LogRec.LogType != BT) {
        LogRec = Log_Read(UndoNxt);
        if(LogRec.LogType == Physical ) {
           Undo_Update(LogRec.BeforeImage);
           WriteLog(CLR, LogRec);
           UndoNxt = LogRec.PrevLSN
        }
        else if(LogRec.LogType == Logical){
           Undo_Update(LogRec.Operate,
                       LogRec.RID);
        }
        else if( LogRec.LogType == APRCLR ||
                 LogRec.LogType == CLR )
            UndoNxt = LogRec.UndoNxtLSN;
        else if( LogRec.LogType == APR ) {
            RecPage = GetRecPage(LogRec.ARID);
            Undo_Update(RecPage.BeforeImage);
            WriteLog(APRCLR, LogRec);
            UndoNxt = LogRec.PrevLSN;
        }
        else
            UndoNxt = LogRec.PrevLSN;
    }
    Write_Log(ETR(abort)) // ETR 를 기록
}
```

[그림 1]은 트랜잭션 철회 시 APRCLR 이 로그에 기록되는 과정을 나타낸다.

L1, L2, L3 로그 3 개가 기록되어 있으며 L1 과 L3 는 physical 로그이고, L2 는 APR 이다. A 상태에서 B 상태로 트랜잭션 철회를 하는 과정을 보인 것이다. L2 의 Anchor 는 회복 세그먼트의 Page 2 를 가리킨다. L1 과 L3 의 트랜잭션 철회한 결과로 L1'와 L3'가 만들어진다. 이들 L1'와 L3'는 갱신 후 이미지를 유지하고 있으며 이는 로그의 양을 증가 시키는 요인이 된다. L2 가 트랜잭션 철회한 결과로 L2'가 만들어 지며 L2'의 anchor 는 Page 2 를 가리킨다. L2'는 갱신 후 이미지를 회복 세그먼트에 유지 함으로써 로그 버퍼에 기록되는 보상 로그의 크기를 줄일 수 있다.



[그림 1]은 트랜잭션 철회 시 로그 기록 과정

4. 결론

본 논문은 가변 길이의 공간 데이터에 대해 효과적인 회복을 지원하기 위하여 데이터 크기에 따른 선별 적인 회복 기법을 제안하였다. 제안된 회복 기법은 데이터의 크기를 3 가지 형태로 나누며, 그 중 MIDDLE 데이터를 위하여 새로운 로그 타입인 APR 을 제안하였다. APR 은 로그 버퍼 풀로 인해 생기는 디스크 입출력 시간을 줄이며 트랜잭션 철회 시 기록되는 보상 로그의 양을 줄임으로써 시스템의 성능을 크게 향상 시켰다.

참고문헌

[1] Carey M. J. et al., " Object and File Management in the EXODUS Extensible Database System," Proc. Of 12th VLDB, pp. 375-383, 1986.

[2] Garza, J. F. and Kim, W. " Transaction Management In An Object Oriented Database System" Proc. ACM-SIGMOD Int' l Conf. On Management of Data, pp. 37-45, June 1988

[3] Lehman, T. J., Lindsay, D. G., " The Starburst Long Field Manager," proc. Of 15th VLDB, pp. 276-285, 1989

[4] Shelter, T., " Brith Of the BLOB," BYTE, pp. 221-226, February 1990

[5] 방기식, 김재홍, 배해영, " 멀티미디어 DBMS 에서 대용량 데이터를 위한 효율적인 회복 기법" , 정보 과학회 논문지, 제 22 권, 제 2 호, pp 206-217, 1995