주기억 데이터베이스 시스템을 위한 병행수행 모델 에 관한 연구

조 성 제[U]
국립4년제 한 국 전 통 문 화 학 교
sjcho@nuch.ac.kr

Concurrency Control Model for Main Memory Database Systems

Sung-Je, Cho[U]
The Korean National University of Cultural Heritage

요 약

실시간 처리 시스템에서 발생하는 트랜잭션은 장시간이 요구되는(CAD/CAM) 긴 트랜잭션이 아닌 일반적인 트랜잭션을 처리하는 환경이 대부분이다.
주 메모리 실시간 DBMS는 디스크에 비하여 상대적으로 빠른 접근시간과 균일한 성능 특성 때문에 실시간 처리 환경에 적합하다. 그러나, 기존의 주기억 데이터베이스에서 병행수행 제어는 2단계 로킹 기법을 사용하고 있다. 이 기법은 특징은 첫째, 록 획득과 해제가 실제 트랜잭션 처리 보다 많은 오버헤드가 발생되고, 또한, 처리속도가 빠르기 때문에 데이터 충돌 발생률이 적다. 이러한 문제를 해결하기 위한 새로운 동시성제어 모델을 제안한다. 제안된 기법은 동시에 여러 트랜잭션을 처리 할 수 있고, 신속히 충돌여부를 결정할 수 있도록 디렉토리별로 정보를 기록하여 실시간에 적합하도록 모델을 제안한다.

1. 서론

최근 들어 컴퓨터 가격 하락과 장비의 발전은 데이터베이스베이스 시스템을 비롯한 많은 분야에 큰 영향을 주었다.특히, 이러한 환경으로 실시간 처리에 요구되는 응용분야는 최근 들어 중요한 연구분야로 자리잡고 있다. 이러한 실시간 응용시스템은 통신산업, 항공 우주산업, 방위산업 등 거의 모든 분야에서 중요성이 점점 증대되고 있다. 이러한 실시간 응용을 위한 데이터베이스 연구에서 데이터가 효용성을 지닐 수 있게 되는 규정시간을 어떻게 지정하는가 하는 문제는 상대적으로 연구가 미흡한 상태이고 처리하기 어려운 문제이다. 데이터 베이스 내의 데이터는 공용 데이터이고 데이터 요구가 중복 상충되는 환경이 수시로 변하기 때문에 규정시간 내의 보장하기는 어려운 일이다.

이러한 문제를 해결하기 위한 새로운 기법이 요구되고 있다.그러나, 기존의 데이터베이스관리 시스템에서는 기본적으로 전체 데이터베이스가 디스크에 저장되고 있고, 트랜잭션 처리시 필요로 하는 데이터만 주기억장치로 읽어 들여, 접근한 후 다시 디스크에 저장하는 방법을 사용하기 때문에 전반적으로 수행속도가 늦고, 실시간 처리 응용에 기존 기법을 이용하는 데는 많은 문제점을 안고 있다. 이러한 문제점을 극복하기 위하여 데이터를 주기억장치에 상주시키고 이를 처리는 주기억 데이터베이스 환경 구축이 바람직 하다할 수 있다. 주기억 데이터베이스 관리시스템에 관한 연구는 플래쉬(flash) 시스템,Dali 시스템, Starburst 시스템, System M 등이 있다. 이 기법은 대부분 2PL(2 Phase locking) 기법을 사용하고 있다. 주기억 데이터베이스관리 시스템의 트랜잭션 처리 속도가 디스크 기반 데이터베이스 관리 시스템 보다 훨씬 빠르다. 그래서, 주기억 데이터베이스 시스템은 디스크 기반 시스템에 비해 트랜잭션 간의 충돌이 훨씬 적다. 즉, 하나의 페이지를 로킹하는 시간은 처리 속도와 비슷하다. 이것은 디스크 기반시스템과 달리 록(lock)을 획득과 해지는 큰 오버헤드로 작용하게 된

다. 이런 문제점을 해결하기 위해서는 기존의 동시성 제어기법을 개선해야한다. 본 논문에서는 주기억장치에 적합한 트랜잭션 수행 모델을 설계하고, 그에 따른 병행수행제어 기법을 제한 하고자 한다. 논문의 구성은 다음과 같다. 2장은 기존의 기법을 살펴보고, 3장에서는 주기억장치 데이터베이스 시스템의 병행수행 제어 모델을 제시하여 비교 분석하고, 4장은 결론을 맺는다.

2. 관련연구

2.1 플래쉬(FLASH) 시스템

이 시스템[4]은 망정보 제어 및 관리시스템을 통한 광역 착신 과금 서비스와 신용 통화 서비스에 필수적인 실시간 검색 시스템으로 제한 기법이다. 주기억장치에 데이터가 상주하기 적합하도록 하는 기억장소 배치와 ECBH(Extendibles Chained Bucket Hash)이라는 접근 기법이다. ECBH 구조는 3단계의 계층 구조를 가지고 있다. 상위 레벨(Level)은 전체 디렉토리와 유사하다. 두 번째 레벨은 CHB테이블이다. 세 번째 RID의 리스트 배열에 대한 포인트이다. 이 모델은 관계형 데이터 모델만 지원하며, 아이템의 길이는 고정 길이이다. 그러나, 해쉬를 사용한 데이터 접근 방법은 로킹(LOCKING)을 사용하고 있다. 이 기법의 단점은 데이터 획득과 해지를 위한 시간이 처리 시간보다 훨씬 많이 소요된다.

2.2 Dali 시스템

이 기법[5]은 회복관리자, 동시성제어 관리자, 그리고 아이템 관리자로 되어있다. 이 시스템에서는 서로 관련된 데이터들을 묶어서 전체 데이터베이스를 여러 개의 데이터베이스 파일들로 구성하여 파일 단위로 디스크에 백업을 수행하며, 주기억장치에 로드시에는 페이지 별로 로드 시킨다. 이 시스템 데이터 파일에는 로그(log)데이터와 록(lock) 데이터가 저장된다. 사용자가 시스템 데이터 파일을 통해 직접 록 데이터를 얻어 데이터베이스에 대해 접근한다. 그리고, 이 기법은 관계형 데이터 모델만을 지원하며, 아이템은 고정길이 아이템을 지원한다.

3. 주기억 데이터베이스 시스템의 병행성 제어

주기억 시스템에 대한 접근은 기존 디스크 시스템에 비하여 매우 빠르기 때문에 트랜잭션의 수행이 빠르게 완료된다. 따라서, 잠금의 단위가 크게 잡는 것이 작은 단위 보다 훨씬 바람직하다.왜냐하면, 잠금의 단위가 크면 충돌의 발생 가능성은 크게 되지만, 이로 인한 로깅 지연이 짧기 때문에 큰 문제가 되지 않는다[1].

이와 같이 할 경우 잠금의 설정 및 해제, 데드락의 검출 및 처리 등과 같은 오버헤드가 제거되므로 오히려 기존의 수행하는 경우보다 실행 효율을 높일 수 있다. 그러나, 2PL 기법은 같은 페이지에 다른 아이템을 사용하지 못하므로 불필요 대기시간(waiting)이 발생한다. 이때 실시간 처리에 불합리함으로써 낙관적 기법을 사용하는 것이 유리하다. 디스크를 기반으로 하거나 주기억장치를 기반으로 한 실시간 데이터베이스 시스템의 병행수행 제어는 데이터 충돌이 작을 경우 2PL 기법보다 낙관적인 병행수행 기법이 우수함을 보여주고 있다. 본 논문에서는 이런 문제점 해결하고자 페이지별 디렉토리를 구성하여 낙관적 병행수행기법을 제한한다.

3.1 모델 및 가정

<가정1> 실시간 처리 환경이므로 주기억 데이터베이스 시스템 기반으로 한다.

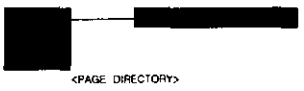
<가정2> 같은 페이지에 있는 동일한 데이터 아이템을 사용할 경우 충돌 상태이고, 그 외 경우를 비충돌 상태라고 정의한다.

본 논문의 시스템 환경은 기존 방법과 달리 로킹 단위를 페이지로 하여 빈번한 록을 제거하였다. 그 결과, 로킹의 오버헤드를 감소하였다. 또한, 충돌을 해결하기 위한 방안으로 PLT를 두어 신속한 페이지 획득이 가능하도록 하였다. 백업로그는 디스크에 유지하지 않고 배터리 백업로그를 이용하므로 써 디스크 입출력을 제거하였다. 전체 시스템 구성도는 <그림1>와 같다.

<그림1. 시스템 구조>

3.2 로킹 동작

트랜잭션 요구한 데이터 페이지를 주기억 데이터베이스 시스템으로부터 획득하여 <그림2>와 같이 Page 디렉토리에 페이지 번호와 트랜잭션 번호(Tr# 라고 한다)를 기록하고 무조건 해당 트랜잭션을 수행한다. 그리고, 트랜잭션 수행 완료 시 <그림3>와 같이 로그 레코드에 P#, offset, lock mode, Tr#를 기록하고, commit/abort를 결정한다. 충돌 여부는 commit check table(CCT 라고 한다)에 의해 결정한다.



<그림2. 페이지 디렉토리 구조>

<그림3. 로그 레코드 구조>

3.3 page 디렉토리

정보를 insert 시점은 트랜잭션이 데이터 페이지를 신청하면 병행수행 제어 시스템이 디렉토리에 페이지 번호와 트랜잭션 번호를 기록한다. 여기서 트랜잭션 번호는 같은 페이지를 사용하는 작업의 요구 순으로 일련번호를 부여한다. 정보를 delete 시점은 트랜잭션 수행 완료 직전 로그 레코드를 전송 한 후에 제거한다. 또는 해당 트랜잭션 abort시 제거한다. 충돌 발생시 Lock check 디렉토리에 로그 정보를 기록하고, 비 충돌시 로그레코드 정보와 해당 페이지에 있는 lock 정보를 모두 제거한다. 마지막으로 체크 포인트 발생시 모두 정보를 제거한다.

3.4 lock check 디렉토리

트랜잭션 수행 후 로그 레코트를 병행수행제어에게 전송된다. 병행수행제어가 Lock check 디렉토리에서 다른 트랜잭션이 같은 페이지를 사용시 데이터 아이템 충돌 여부 점거하여 기록 여부를 결정한다. 충돌하지 않을 때 해당 페이지 정보를 모두 제거한다.

예를 들면, 트랜잭션1에서 데이터페이지1에 있는 데이터 아이템 *100*을 *read* 한 후, 작업을 완료하고, *Lock* check 디렉토리 로그를 기록하였다. 그 다음 트랜잭션3에서 데이터 아이템을 이용하여 갱신 연산 후 commit 직전에 *Lock check* 디렉토리를 조사했을 때, 충돌 발생시 트랜잭션3을 정상 작업으로 처리한다. 그러나, 위의 예에서 트랜잭션3 작업이 먼저 완료 후 트랜잭션3 작업이 먼저 완료후 트랜잭션1이 완료되었을 때는 비정상 작업으로 간주하여 abort한다.

3.5 비교분석

본 논문이 제한한 병행수행제어 알고리즘의 특성을 살펴보면 다음과 같다.

.본 논문에서는 트랜잭션 전송단위가 페이지이고, 록 단위는 데이터 아이템으로 구성되어 있다.

.수행 완료된 트랜잭션들이 동시에 같은 페이지를 사용할 수 있다.

.수행 완료된 트랜잭션은 Lock check 디렉토리를 이용하여 충돌여부를 조사 한 후, 신속히 처리한다.

제한된 방법을 기존의 병행수행제어 기법과 비교할 때 다음과 같은 장점을 갖게된다.

1. 주기억 데이터베이스 환경에서의 페이지별로 록 정보를 따로 관리함으로써 신속히 충돌여부를 결정한다.

2.기존기법과 달리, 같은 페이지를 여러 트랜잭션들이 동시에 접근을 허용함으로써 시스템 병렬성이 향상된다.

3. 기존 기법의 병행수행제어 기법은 록을 획득 및 해제에 많은 시간이 소요되는 오버헤드 있다. 그러나, 낙관적 기법을 사용하여 문제점을 해결하였다.

4.기존 기법과 달리, 충돌 검사시 먼저 페이지별 검사 후 데이터 아이템 단위로 검사함으로써 실시간 처리 할 수 있다.

4. 결론

본 논문에서는 주기억 데이터베이스 시스템을 위해 낙관적인 기법에 기초로 한 새로운 병행수행 제어 기법을 제한하였다. 그 결과, 신속히 충돌여부를 결정 할 수 있고, 같은 페이지를 동시 접근함으로써 시스템 병렬성을 향상된다. 또한, 록을 획득 및 해제에 소요되는 오버헤드 문제점을 해결하였다. 앞으로 연구과제는 제시된 기법과 기존 기법을 시뮬레이션을 통한 비교분석이 필요하다.

5. 참고문헌

[1] Bernstein, P.,Hadzilacos, V., and Goodman, N., "*Concurrency control and Recovery in Database Systems*", Addition-Wesley, 1987

[2] Carey, M., Dewitt, D., Richardson J., Schekita, E., "Storage Management for Objects in EXODUS", in Object-Oriented Concepts, Databases, and Applications, W. Kim F. Lochovsky, eds., Addition-Wesley, 1989.

[3] Haerder, T., Reuter, A., "Principles of Transaction Oriented Database Recovery - A Taxonomy", Computing Surveys, Vol. 15, No. 4, Dec., 1983.

[4] Kaist "flash: A Main Memory Storage Sytem",94-257-1.

[5] Jagadish, H.v., Lieuwen, D.,"Dal: A High Performance Main Memory Storage Manager",VLDB Conference,1994.

[6] lehman, "An Evaluation of Starburst Memory Resident storage component,IEEE TR ..,PP555-565,1992

[7] *Salem,k and Garcia-Molina.,"system M:A tr processing* Testbed for Memory Resident Data,". IEEE TR ,PP161-172,1990