KORED/StormNT : 공간 데이터베이스를 위한 저장관리자의 설계 및 구현

김종현, 김명근, 김성희, 배해영
인하대학교 전자계산공학과

KORED/StormNT : Design and Implementation of a Storage Manager for Spatial Database

Jong-Hyun Kim, Myung-Keun Kim, Sung-Hee Kim, Hae-Young Bae
Dept. of Computer Science & Engineering, Inha University

요 약

기존의 대용량 멀티미디어 데이터와는 달리 다양한 크기와 접근 패턴을 가지는 공간데이터를 효율적으로 저장 관리하기 위해서는 이러한 공간 데이터의 특성을 고려한 공간데이터베이스 시스템이 필요하며, 특히 고성능의 공간데이터베이스 시스템을 위해서는 공간데이터를 저장 관리하기 위한 저장관리자의 연구가 선행되어야 한다.

본 논문에서는 공간데이터의 특성을 고려하여 효율적으로 저장 관리하기 위한 공간 데이터베이스 관리 시스템의 저장관리자인 KORED/StormNT를 제안한다. KORED/StormNT에서는 다양한 크기를 가지는 공간데이터의 특성을 고려하여 디스크 입출력 비용을 최소로 줄일 수 있는 공간데이터 저장기법을 사용하였으며, 공간데이터의 크기에 따른 회복기법을 사용하여 회복비용을 최소화하였다. 또한, 공간조인(Spatial Join) 연산을 효율적으로 지원하기 위해 별도의 테이블을 이용한 기법을 사용한다.

1. 서론

최근 데이터베이스의 구축 규모의 증대와 인터넷 및 네트워크의 확산으로 공간 데이터베이스 시스템을 필요로 하는 분야가 점차 다양해지고 있다[5]. 공간 데이터는 기존의 멀티미디어 데이터와는 다르게 데이터 크기가 수십 바이트로부터 수 기가 바이트에 이르는 다양한 크기를 가지며, 공간 질의에 따라 일정하지 않은 접근 패턴을 가지고 있다[6]. 따라서, 공간 데이터를 저장, 관리하기 위해서는 공간 데이터의 특성을 고려하여 설계된 공간 데이터베이스시스템이 필요하다. 또한, 이러한 공간 데이터베이스시스템의 성능 향상을 위해서는 공간 데이터에 빠르게 접근하기 위한 색인 기법과 공간 데이터베이스시스템의 고장시 회복 비용을 최소화 할 수 있는 회복기법 및 효율적인 공간 질의 처리를 지원하기 위한 저장관리자가 요구된다[6].

본 논문에서는 공간 데이터의 특성을 고려하여 설계 및 구현된 공간 데이터 저장관리자인 KORED/StormNT를 제안한다. KORED/StormNT에서는 디스크 입출력 비용을 최소화하기 위해 작은 크기의 공간 데이터는 비공간 데이터와 함께 저장하고 특정 크기 이상의 공간 데이터는 대용량 데이터 저장기법을 사용하여 공간 낭비를 최소화하였다. 공간 데이터베이스시스템의 고장시 회복의 경우에는 작은 크기의 공간 데이터는 로깅(Logging) 방식[2]의 기법을 사용하고 대용량의 공간 데이터는 쉐도우(Shadow) 기법[3]을 사용함으로써 회복 비용을 최소화하였다. 또한, 고비용을 요구하는 공간 조인 연산에 대해서는 조인 테이블들의 레코드 ID만을 필드로 갖는 별도의 테이블을 구성함으로써 연산 비용을 최소화시키는 기법을 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 공간데이터의 특성에 대해 고찰하고, 3장에서는 KORED/StormNT의 저장관리자를 개발하면서 고려되었던 사항들에 대해 서술한다. 그리고 4장에서는 KORED/StormNT의 저장관리자 설계 및 구현에 대해 기술하고, 마지막으로 결론을 맺기로 한다.

* 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구결과임.

2. 공간데이터의 특성

공간데이터는 크기가 최소 수십 바이트에서 수 기가바이트까지 다양하기 때문에 최소 저장 단위가 한 페이지 이상인 기존의 대용량 데이터 저장 관리기법을 이용하면 공간데이터의 크기가 한 페이지보다 작을 경우, 페이지의 공간 낭비 및 데이터 크기에 비해 디스크 입출력이 많이 발생하여 시스템의 성능이 저하되는 문제가 발생한다[6].

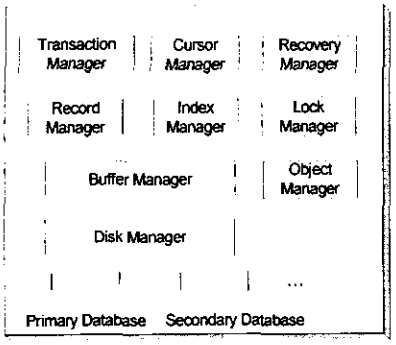
또한, 사용자의 공간질의는 비공간데이터와 공간데이터 동시에 이루어지기 때문에 공간테이블(Spatial Table)에 비공간데이터와 공간데이터가 따로 저장될 경우, 고비용의 조인연산이 필연적으로 발생한다. 그러므로 비공간데이터와 공간데이터를 함께 저장하는 것이 효율적이다[6].

일반적으로 공간데이타에 대한 공간질의 수행시 공간 연산 자체가 고비용이고, 공간데이터가 대용량 데이터일 경우 디스크 입출력이 많이 발생하여 시스템의 전체적인 성능을 저하시킨다[6].

3. 공간 저장 관리자 KORED/StormNT의 구조

KORED/StormNT 저장관리자는 디스크 관리기 레벨에서는 대용량 데이터를 지원하기 위해 64비트의 주소를 페이지 식별자로 사용하여 각 볼륨(Volume)당 최대 4테라바이트까지 지원하고 있다. 버퍼 관리기 레벨에서는 객체의 헤더정보의 신속한 접근을 위해 따로 객체 관리기에서 관리하며, 버퍼 관리기는 버퍼정책으로 Steal, No Force 정책[4]을 채택하고 있다. 레코드 관리기 레벨에서는 공간데이터의 지원을 위해 기본 데이터 타입 외에도 공간데이터 타입이 추가되었다. 또한 레코드 관리가 레벨에서 공간데이터에 대한 연산이 지원되며, 색인으로는 B^+-Tree외에 공간색인으

로 R-Tree를 지원하고 있다. 트랜잭션 레벨에서는 트랜잭션의 4단계 격리 레벨을 지원하고 있으며, 동시성 제어를 위해 잠금기법을 이용하고, 회복을 위해 ARIES[2]와 쉐도우 페이지 기법[3]을 이용한다. [그림 3-1]은 KORED/StormNT의 시스템 구조도이다.



[그림 3-1] KORED/StormNT 구조

4. KORED/StormNT의 주요 모듈

4.1 디스크 관리기

디스크 관리기 모듈은 KORED/StormNT의 가장 하부에 위치하여 물리적인 저장 구조인 디스크를 관리한다. 디스크 관리기에서 관리하는 물리적인 단위는 데이터베이스(database), 세그먼트(segment), 익스텐트(extent), 페이지(page)로 구성된다. 최종적인 데이터 관리는 페이지 단위로 관리를 하게 되며, 이러한 페이지를 관리하기 위해서 디스크 관리기는 각 페이지에 대해서 Page ID를 부여한다. Page ID는 데이터베이스 ID + Segment ID + Offset으로 구성되며 페이지의 물리적인 위치를 나타내므로 해당 페이지를 신속하게 찾을 수 있다.

4.2 버퍼 관리기

KORED/StormNT에서 버퍼 관리기는 페이지의 중요성에 가중치를 두는 힌트(참조 계수)를 적용한 최소 최근 사용 정책을 사용한다. 버퍼 관리기의 구조는 사용되고 있는 버퍼들의 리스트를 유지하는 해쉬 테이블(hash table)과 사용되고 있지 않는 자유 버퍼들의 리스트를 관리하는 프리 리스트(free_list), 그리고 버퍼상에서 내용이 변경된 페이지들을 관리하는 오손(dirty) 페이지 리스트로 구성된다.

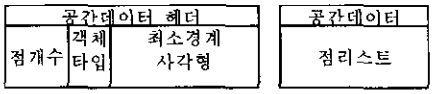
4.3 객체 관리기

KORED/StormNT에서는 테이블, 인덱스 등을 객체로 취급한다. 객체에 대한 정보를 갖고 있는 설명자는 여러 개의 트랜잭션에서 공유되므로 KORED/StormNT에서는 이러한 정보를 효율적으로 관리하기 위해 버퍼에 넣어서 관리하고 있다. 각각의 테이블을 위한 설명자 구조는 테이블에 대한 설명자와 해당 테이블에서의 인덱스들에 대한 설명자로 구성되는데 KORED/StormNT의 인덱스는 한 테이블에 대해 최대 10개까지 정의할 수 있으며 하나의 인덱스는 최대 10개의 필드들을 조합하여 키로 사용할 수 있도록 하여 버퍼 내에서 효율적으로 관리할 수 있도록 하였다.

특히 대용량의 공간데이터가 들어가 있는 테이블들 사이의 조인 연산을 지원하기 위해 페이지 ID(PID)와 페이지 내에서의 레코드 ID(RID)만을 가지는 Temp 테이블을 구성한다. 이러한 Temp 테이블을 사용함으로써 대용량의 공간데이터 테이블에 대한 빠른 조인 연산과 최소의 저장공간에 대한 효과를 얻을 수 있다.

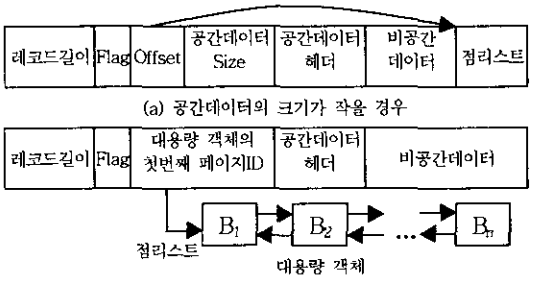
4.4 레코드 관리기

KORED/StormNT에서의 레코드는 공간데이터의 접근 패턴과 다양한 크기를 가지는 특성을 고려하여 효율적으로 관리하기 위해 기존의 공간데이터와 비공간데이터가 분리되어 저장되는 레코드 구조를 변경하였다. 즉, [그림 4-1]과 같이 공간데이터를 분리, 저장함으로써 공간연산을 위한 여과과정이나 R-Tree 구성시 발생되는 고비용의 디스크 입출력을 줄일 수 있다.



[그림 4-1] 공간데이터 구조

또한, 다양한 크기를 가지는 공간데이터에 대한 효율적인 저장관리를 위해 공간 레코드가 한 페이지의 1/3 이하이면 일반 레코드로 표현하여 저장하는 반면에 한 페이지의 1/3 이상이 되면 대용량 객체 타입으로 저장한다. 이때 대용량 객체 타입으로 저장될 때 일반 데이터 페이지에는 속성 레코드와 공간 데이터에 대한 헤더 정보만을 저장하고 실제 공간 데이터는 대용량 객체 페이지에 저장한다. 이렇게 함으로써 크기가 작은 공간데이터에 대해서 공간낭비의 문제를 해결할 수 있다. 이를 위해 KORED/StormNT에는 모든 레코드에 플래그(Flag)가 존재하는데 이 값을 가지고 현재 레코드에 공간데이터가 비공간데이터와 같이 저장이 되어있는지, 분리되어 대용량 객체로 저장되었는지를 표시한다. [그림 4-2]은 이를 반영하여 구현된 KORED/StormNT에서의 레코드 구조이다.



(a) 공간데이터의 크기가 작을 경우

(b) 공간데이터의 크기가 클 경우

[그림 4-2] KORED/StormNT 레코드 구조

4.5 대용량 객체 관리기

KORED/StormNT에서의 레코드의 구조는 공간데이터의 헤더는 항상 비공간레코드와 함께 저장이 되고 공간데이터의 점리스트는 크기에 따라서 작으면 비공간데이터 레코드와 함께 저장이 되고, 크면 대용량 객체 형식으로 저장을 한 후 대용량 객체의 첫 번째 페이지 ID만 비공간데이터 레코드에 기록이 된다.

대용량 객체 관리기는 이러한 대용량 객체로 저장된 공간데이터 페이지들의 관리 및 공간데이터 레코드들에 대한 삽입, 삭제, 갱

신 등의 기능을 수행한다.

4.6 색인 관리기

KORED/StormNT의 색인 관리기는 속성 데이터를 위한 B⁺-Tree와 공간 데이터를 위한 공간 색인 구조인 R-Tree를 지원한다. 또한 공간 조인 연산의 결과인 Temp 테이블에 대한 빠른 액세스를 위해 동시성 제어와 회복 기법을 사용하지 않는 변형된 B⁺-Tree를 색인으로 사용한다.

4.7 커서 관리기

KORED/StormNT는 커서(cursor)를 이용하여 특정 조건을 만족하는 레코드들에 대하여 순서대로 순회(navigation)하면서 레코드들을 읽거나 삽입, 삭제, 변경의 기능을 수행한다. 커서는 순차 커서(sequential cursor)와 인덱스를 이용하는 인덱스 커서(index cursor)로 구분된다.

4.8 잠금 관리기

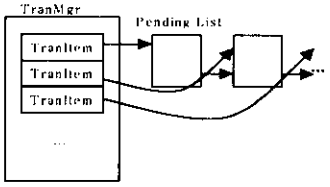
KORED/StormNT에서는 동시성 제어를 위해 ARIES의 잠금 기법을 이용한다. 비공간데이터와 공간데이터가 따로 저장되더라도 공간데이터에 대한 연산은 먼저 연산대상이 될 비공간레코드를 찾고 비공간레코드에 공간데이터가 들어있는 페이지 ID에 대한 정보를 통해 공간데이터에 대한 연산이 수행되기 때문에 비공간데이터 레코드에 대해서만 잠금을 요청한다.

4.9 트랜잭션 관리기

KORED/StormNT의 트랜잭션 관리기는 ARIES 기법으로 설계, 구현되었다.

트랜잭션 관리기는 개개의 트랜잭션을 관리하기 위한 트랜잭션 슬롯(트랜잭션 항목:TranItem) 구조와 이들을 총괄적으로 관리하는 트랜잭션 관리(TranMgr) 구조로 구성된다. [그림 4-3]은 트랜잭션 슬롯 구조와 이들의 집합과 총괄적인 정보를 갖고 있는 트랜잭션 관리 구조를 보인다.

KORED/StormNT에서는 효율적인 쓰레드 관리를 위해 트랜잭션 쓰레드를 최대 16개까지로 제한하고 있으며, 이로 인해 TransItem은 최대 16개까지 관리된다.



[그림 4-3] 트랜잭션 관리 구조

4.10 회복관리기

KORED/StormNT의 회복기법은 ARIES를 기반으로 한다. 그러나, 대용량의 데이터에 대해서 동일하게 ARIES의 회복의 근간인 로깅기법을 이용하면 디스크 공간 및 입출력이 많이 필요하여 시스템의 성능이 저하되므로 대용량 데이터의 회복방법인 쉐도우 방법을 적용한다. 따라서, 크기가 다양한 공간데이터의 특성을 고려하여 KORED/StormNT에서는 한 페이지보다 작은 데이터에 대해서는 ARIES에 기반한 로깅 회복기법을 적용하고 한 페이지보다 큰 대용량 데이터에 대해서는 쉐도우 페이지 기법을 적용한다.

KORED/StormNT에서는 대용량 공간데이터의 갱신연산인 쉐도우 페이지 기법을 약간 변형하였다. 예로, 갱신(Update)연산의 경우 기존의 데이터는 그대로 두고 새로운 데이터를 새로 할당된 대용량 객체에 기록하고 비공간데이터 레코드에 새로운 대용량 객체의 첫번째 페이지 식별자를 기록한다. 그리고 갱신 트랜잭션이 완료(Commit)될 때 기존의 대용량 객체 데이터를 삭제한다. 갱신연산을 수행할 때 대용량 객체의 삭제 연산을 트랜잭션의 완료시점까지 미루기 위하여 트랜잭션의 Pending 리스트라는 자료구조를 이용한다. 여기서 트랜잭션의 Pending 리스트는 각각의 트랜잭션들에 대해서 트랜잭션의 완료시점까지 지연시켜야 할 연산을 가지고 있다. 이렇게 트랜잭션 Pending 리스트에 추가하여 연산을 지연시키는 것은 대용량 객체에 속해 있는 페이지가 반환되어 다른 데이터가 쓰여지게 된다면, 이전 데이터가 유실되어 현재 트랜잭션을 취소시킬 수 없기 때문이다.

5. 결론 및 향후 연구방향

본 논문에서 제안한 KORED/StormNT는 공간데이터의 크기와 접근 패턴 등의 특성을 최대한 고려하여 효율적으로 공간데이터를 저장, 관리할 수 있도록 윈도우 NT/2000 서버 환경 하에서 구현된 저장관리자이다. 또한, 조인과 같은 고비용의 공간 연산을 효율적으로 수행하기 위해 Temp 테이블과 Temp 테이블에 대한 빠른 접근을 위해 변형된 B⁺-Tree 색인 기법을 사용하였으며, 공간데이터의 크기에 따라 서로 다른 회복기법을 사용함으로 회복 비용을 최소화 하였다.

향후, 고비용의 공간연산을 효율적으로 처리하기 위한 공간연산자에 관한 연구가 필요하다.

참고문헌

[1] Guting, R., "An Introduction to Spatial Database Systems", VLDB Journal, Vol.3, pp.357-399, 1994.

[2] C. Mohan, et al., "ARIES : A Transaction Recovery Method Supporting Fine-Granularity Locking and Patial Rollbacks Using Write-Ahead Logging," IBM Research Report RJ6649, IBM Almaden Research Center, 1989.

[3] EXODUS Project Group, "Using the EXODUS Storage System V3.1" EXODUS Project Document, University of Wisconsin-Madison, 1993.

[4] Transaction Support in an Extensible Operating System. Yasushi Saito and Brian N. Bershad. 1998 USENIX Annual Technical Conference. New Orleans, LA. June 1998.

[5] 이환재, 안준순, 강동재, 이경모, 정보홍, 배해영, "GEO/Millennium : 클라이언트-서버 공간 데이터베이스 시스템," 한국 정보과학회 춘계학술대회 발표 논문집, Vol. 27, No. 2, 2000.

[6] 김대일, "공간저장관리자의 공간데이타 특성을 고려한 저장관리 기법," 인하대학교 전자계산공학과 대학원 석사학위논문,