큰 공간 객체의 변경을 위한 동시성 제어
(Concurrency Control for Updating a Large Spatial Object)

서 영 덕 † 김 동 현 †† 홍 봉 희 †††
(Young Duk Seo) (DongHyun Kim) (Bong Hee Hong)

요 약 지리정보시스템(Geographic Information System)에서 지도 갱신 작업은 대화식으로 이루어지는 긴 트랜잭션이다. 지도 갱신 작업의 동시성을 향상시키기 위하여 특정 공간객체가 트랜잭션의 작업 영역보다 큰 공간 속성을 가질 때 다수의 트랜잭션이 해당 객체를 동시에 변경할 필요가 있다. 그러나, 큰 객체에 대한 쓰기 잠금이 충돌하기 때문에 둘 이상의 트랜잭션이 동일한 큰 객체를 동시에 변경할 수 없다. 이 논문에서는 큰 객체의 동시 변경을 지원하기 위하여 트랜잭션이 큰 객체의 부분에 대하여 잠금을 설정하는 부분 잠금 기법을 제시한다. 부분 잠금은 특정 공간객체의 부분객체에 대하여 사용자에 의해 설정되는 배타적 잠금으로 부분객체의 공간 속성을 이용하여 잠금 간의 충돌을 검사한다. 제안한 기법은 큰 객체에 대한 동시성 제어 단위를 낮추기 때문에 공간 객체를 변경하는 긴 트랜잭션의 동시성을 향상키는 장점을 가진다.

키워드 : 공간 DB, 공간 데이타 변경, 동시 제어, 트랜잭션 처리

Abstract The update transactions to be executed in spatial databases usually have been known as interactive and long duration works. To improve the parallelism of concurrent updates, it needs multiple transactions concurrently update a large spatial object which has a spatial extensions larger than workspace of a client. However, under the existing locking protocols, it is not possible to concurrently update a large spatial object because of conflict of a write lock. This paper proposes a partial locking scheme of enabling a transaction to set locks on parts of a big object. The partial locking scheme which is an exclusive locking scheme set by user, acquires locks for a part of the big object to restrict the unit of concurrency control to a partial object of a big object. The scheme gives benefits of improving the concurrency of un updating job for a large object because it makes the lock control granularity finer.

Key words : Spatial DB, Spatial data update, concurrency control, transaction processing

1. 서 론

지리정보 시스템(GIS)은 지도 데이타베이스를 수정/검색하기 위한 목적으로 많이 사용되고 있다. 지리정보 시스템은 사용자에게 가장 최신의 데이타를 제공하고 있어야 하므로, 실 세계에서의 지형상의 변경은 지리 정보 시스템의 데이타베이스 내에 항상 반영되어 있어야 한다. 지도 수정의 효율성을 위하여, 많은 작업자들의 공간 데이타베이스에 대한 동시 접근이 가능해야 한다. 또한, 그 수정은 작업자와 대화식의 작업을 수행해야 하므로, 지도 데이타베이스의 수정 트랜잭션은 몇 분 혹은 몇 시간의 수행시간을 가지는 긴 트랜잭션(long transaction)이 된다.

지도 데이타베이스의 수정 작업은 작업의 효율성과 편리성을 위하여 일반적으로 클라이언트-서버 환경에서 수행된다. 하나 이상의 사용자는 전역 트랜잭션(global transaction)의 제어 하에 수정 트랜잭션을 지역 트랜잭션(local transaction)으로 설정한다. 여기서 전역 트랜잭션은 특정한 영역에 위치한 공간 객체 집합을 수정하기 위한 논리적 트랜잭션(logical transaction)이다. 전역 트랜잭션은 개별적인 지역 트랜잭션인 부-트랜잭션(sub-transaction)의 집합으로 구성되어 있으며, 하나 이상의 지역 트랜잭션은 서로의 관심지역에 대한 수정작업을 위하여 동시에 실행될 수 있다. 또한, 하나 이상의 지역 트랜잭션이 겹치는 영역에 존재하는 큰 객체에 대한 동시 수정이 가능하다.

† 비 회 원 : 부산대학교 컴퓨터공학과
ydseo@pusan.ac.kr
†† 정 회 원 : 동서대학교 소프트웨어전문대학원 교수
pusrover@dongseo.ac.kr
††† 종신회원 : 부산대학교 전자전기정보컴퓨터공학부 교수
bhhong@pusan.ac.kr
논문접수 : 2003년 5월 19일
심사완료 : 2004년 9월 14일

지도 수정작업에 대한 연구는 크게 두 가지 접근방법으로 연구되어 왔다. 첫 번째 방법은 비관적인 접근방법으로서, 잠금을 활용하여 공유된 데이타에 대한 동시성 제어를 하게 된다[1]. 이 방법은 일반적으로 단순하고 충돌의 확률이 높은 환경에서 짧은 트랜잭션에 적절하다고 알려져 있다. 그러나, 비관적인 접근 방법은 긴 트랜잭션의 경우 사용자로 하여금 긴 대기시간을 유발시키는 문제점을 가지고 있다. 두 번째 방법으로 연구된 것은 낙관적 동시성 제어 기법이다[2]. 그러나, 이 방법은 긴 트랜잭션의 취소를 발생시키는 문제점을 있다. 긴 트랜잭션의 취소는 모든 수정 트랜잭션의 취소를 발생시키며, 이러한 현상은 공간 데이타와 같은 긴 시간 동안의 작업을 요구하는 트랜잭션에는 적절하지 않다.

잠금에 기반한 기법의 문제점은 큰 객체를 수정하는 트랜잭션에서 발생한다. 트랜잭션이 공유되고 있는 객체에 대하여 잠금 충돌이 발생할 경우, 해당 트랜잭션은 잠금이 해제될 때까지 대기해야만 한다. 그러나, 공간 객체가 도로, 강, 고속도로 등과 같이 큰 외형을 가지고 있는 경우 여러 트랜잭션에 의한 동일 객체의 서로 다른 부분에 대한 수정 요청이 된다. 이러한 경우 여러 트랜잭션이 다른 트랜잭션의 긴 트랜잭션이 완료될 때까지 대기해야 하는 문제점이 발생한다.

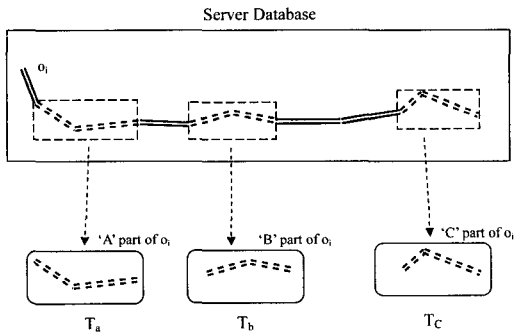


그림 1 큰 객체의 수정

예를 들어, 그림 1에서 세 개의 트랜잭션 T_a, T_b, T_c는 하나의 큰 객체 O_i의 서로 다른 부분을 수정하고 있다. 세 트랜잭션들은 공간 객체를 대화식 작업으로 수정하는 긴 트랜잭션이다. 기존의 연구에서 객체 O_i의 A, B, C부분은 서로 독립적인 영역이므로, 세 트랜잭션은 상호 간섭 없이 개별적인 동시 수정작업을 수행할 수 있다. 그러나, 트랜잭션 T_a, T_b, T_c는 객체 O_i에 대한 잠금 때문에 동시 수정이 불가능하다. 기존의 연구에서 T_a가 객체 O_i에 대한 배타적 잠금(exclusive lock)을 획득한 경우 T_a와 T_b는 O_i에 대한 T_a의 잠금이 해제될 때까지 대기해야만 한다. 객체수준의 잠금은 큰 객체를 수정할 때 서로 다른 영역에 대한 작업 시에도 대기해야 하는 문제점을 발생시킨다.

큰 공간 객체의 서로 다른 부분을 수정하는 트랜잭션에서 각각의 트랜잭션은 대기시간이 최소화되어야 한다. 이 논문에서는 객체의 일부분에 대하여 잠금을 설정하여 큰 객체에 대한 동시성 제어의 단위를 부분객체 단위로 제한함으로써 동시성을 높이고자 한다. 여러 트랜잭션이 하나의 큰 객체의 부분적인 지형에 대한 잠금을 설정하기 위하여 이 논문에서는 객체의 부분 잠금 기법을 제시한다. 이 잠금을 이용하여 각 클라이언트는 자신이 수정하고자 하는 부분에 대한 잠금을 획득할 수 있다. 그러므로, 객체의 다른 부분에 대한 수정을 하고자 하는 다른 클라이언트들은 그들의 트랜잭션을 독립적으로 대기 없이 수행할 수 있다.

이 논문의 구성은 아래와 같다. 먼저 2장에서는 지금까지 클라이언트-서버 환경에서 트랜잭션에 기반한 공간 데이타 변경에 관한 연구를 기술한다. 다음으로3장에서는 큰 공간 객체의 동시 수정시의 문제점을 제시한다. 4장에서는 부분 객체 개념을 제안하고 이에 기반 한 부분 잠금 기법을 설명하여 3장에서 제시된 문제에 대한 해결책을 제시한다. 5장에서는 실제 구현에 관한 알고리즘과 일관성 보장에 대한 증명을 제시한다. 6장에서는 시스템을 구성하고 실제 구현된 결과를 보여준다. 마지막으로 7장에서 결론을 맺는다.

2. 관련 연구

분산 데이타베이스에서 변경 충돌은 낙관적 접근 방법과 비관적 접근 방법으로 나눌 수 있다. 또한 비관적 접근 방법을 완화시킨 공간 관련성 기반 2PC프로토콜과 복합 객체의 잠금을 이용한 수정에 관한 연구 등으로 나눌 수 있다.

Lazy replication scheme[3]과 같은 낙관적 접근방법은 객체가 각 클라이언트에서 개별적으로 수정되는 것을 허용한다. 이 방법에서는 잠금 기법을 대체하기 위하여 다중 버전 개념이 도입되었다. 그러나, 공간 데이타베이스에서 수정 영역에서의 충돌이 발생하지 않더라도 공간 객체의 동시 수정은 분산 공간 관련성에 의한 불안정한(inconsistent) 상태를 이룰 수 있다. 그러므로 낙관적 접근 방법은 공간 데이타베이스에서 적용하기 힘들다.

비관적 접근 방법은 공유 데이타에 대한 동시성 제어를 위하여 잠금을 적용하였다. 이 접근 방법은 단순하고 충돌이 많이 발생하는 상황에서 효과적이기 때문에, 일반적으로 전통적인 짧은 트랜잭션에서 효과적이라고 알려져 있다. 비관적 접근 방법의 결정적 문제는 긴 수정시간을 가지는 대화식 트랜잭션에서의 긴 대기시간의

문제점이 있다.

긴 트랜잭션에서 동시성 향상을 위하여 [4,5]에서는 공간 관련성 기반 2PC프로토콜(SR-Based 2PC protocol)을 제안하였다. 이 연구는 공간 객체간의 분산 공간 관련성을 제시하였으며, 서로 다른 객체간의 동시 수정시의 의존성에 관한 연구를 수행하였다. 또한 [4,5]에서는 수정된 공간 객체의 유효성을 보장하기 위하여 잠금을 기반으로 하는 분산 공간 관련성을 제시하였다. 제안된 방법은 분산 저장된 공간 객체의 동시 작업을 지원하기 위하여 제시되었다. 그러나 이 방법은 하나의 큰객체에 대한 동시 수정을 지원하지 않는다.

객체 지향 데이타베이스관점에서 [6-8]에서는 복합객체의 잠금에 관한 연구를 수행하였다. 이 연구에서는 복합 객체가 여러 개의 컴포넌트 객체로 구성되어 있다고 가정한다. 복합객체가 개별적인 부분 객체로 형성되어 있기 때문에 각 클라이언트는 부분 객체에 대한 수정 연산을 동시에 수행할 수 있다. 만일 복합객체의 부분 객체가 배타적으로 참조되는 경우 복합객체는 동시 수정이 가능하다. 그러나, 이 방법은 복합 객체로 간주되지 않는 동일 객체의 동시수정을 목적으로 하지 않으며, 잠금 기법이 너무 복잡하여 공간 객체 수정과 같은 큰 객체에서의 수정에 적합하지 않다.

Multiple-granularity 잠금 기법은 [1,9]에서 기술되어 있으며, 데이타베이스를 잠금 크기의 측면에서 데이타베이스, 테이블, 레코드로서 정의한다. 이 잠금은 잠금 대상이 되는 데이타베이스, 테이블, 레코드들을 다양한 크기의 데이타로 간주하며 작은 크기의 잠금은 큰 크기의 잠금에 포함되도록 구성된다. 그 계층은 트리 형태로 표현될 수 있으며, 상위 계층의 잠금은 하위 계층의 잠금과 연결되도록 표현된다. 그러나, 이 기법은 공간 객체와 같이 객체 상호간의 관련성을 가진 잠금 대상에 대하여 적용하기 힘들며, 객체 단위 이하로 잠금이 설정되는 경우에 대한 정의가 없다.

3. 큰 공간 객체의 동시 갱신 문제점

공간 객체는 실 세계의 객체의 공간적 특성을 표현하기 위한 객체이다. 공간 객체의 속성정보는 공간 정보를 가진 속성인지 아닌지에 따라 공간 속성과 비 공간 속성으로 나눌 수 있다. 이 논문에서는 공간 객체의 속성을 공간 속성으로 한정한다. 공간 객체는 위치 정보를 표현하는 점 좌표의 집합으로 볼 수 있다. 위치 정보는 동일한 좌표계내에서 위치 정보의 좌표들을 포함하고 있다. 이 공간 정보는 2차원 좌표로서 저장된다. 그러므로 이 논문에서 공간 좌표는 2차원 좌표로 간주하며 공간 객체의 기하정보를 정의 1과 같이 정의한다.

정의 1. 공간 객체 O에 대하여, O.G를 O의 공간 속성으로 둔다. O.G는 기하 속성의 집합이다. 즉, O.G = $\{a_i | a_i = direct\ position,\ 0 \le i \le n\}$.

공간 객체의 수정작업은 전통적인 데이타베이스의 수정작업과 비교해서 긴 트랜잭션이라는 특징을 가진다. 즉, 공간 객체의 수정작업은 대화식 수정작업이기 때문에 공간 객체의 수정작업은 몇 분에서 몇 시간씩 지속되는 특징을 가진다.

기존 연구에서의 잠금에 기반한 수정작업은 하나의 단일객체를 대상으로 한다. 즉, 기존의 잠금 기법은 공간 객체에 대한 수정은 객체나 레코드 테이블 등과 같은 데이타베이스 단위 객체에 기반한다. 이때, 객체를 수정하고자 하는 트랜잭션은 해당 객체에 대한 잠금을 획득해야만 한다. 그러나, 두 서로 다른 트랜잭션은 수정된 객체의 일관성을 유지시키기 위하여 동기화되어야 한다. 이를 위하여 기존의 연구에서는 잠금 프로토콜을 적용하였다. 그러나, 이러한 잠금 기법은 잠금 단위 객체를 객체나 레코드에 제한하는 문제점을 가지고 있다.

예를 들어 그림 2의 지도 데이타베이스를 수정하고자 하는 4개의 지역 트랜잭션(T_1, T_2, T_3, T_4)으로 구성된 전역 트랜잭션을 가정하자. 전역 트랜잭션의 수정 영역은 그림 2의 외부 박스이며, 지역 트랜잭션 T_1, T_2, T_3,

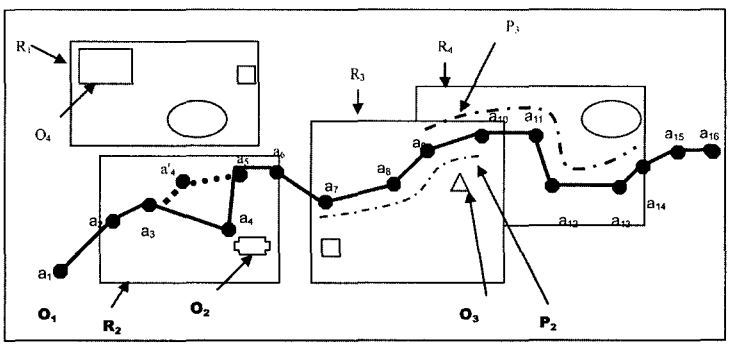


그림 2 공간 객체의 수정

T_4의 작업 영역은 각각 R_1, R_2, R_3, R_4이다. 지도를 수정하기 위하여 각 트랜잭션들은 그들의 작업 영역에 대한 잠금을 획득하여야 한다. 그림 2에서 T_1은 객체 O_4를 바로 수정할 수 있다. 즉, T_1은 R_1에 대한 영역 잠금을 요청하고, 만일 R_1이 다른 트랜잭션의 잠금 영역과 충돌되지 않으면 T_1은 O_1에 대한 잠금을 획득한 후 일관성 있는 상태에서 수정할 수 있다. 반면 객체 트랜잭션 T_3와 T_4에 의하여 공유되는 O_3을 수정하는 경우 만일 T_3가 객체 O_3에 대한 잠금을 가지고 있다면, T_4는 데이타베이스가 일관성 있는 상태가 될 때까지 대기하여야 한다.

이 논문에서의 문제는 공간 객체가 O_1과 같이 크기가 크고, 그 객체를 포함하기 위한 최소 근접 사각형(bounding box)의 크기가 커지는 경우에 발생한다. 문제점은 크게 두 가지로 볼 수 있다. 첫 번째 문제는 큰 크기를 가지는 공간 객체의 잠금 문제이다. 만일 T_2가 O_1의 점 a_4을 a_4'로 이동하고자 하는 경우 T_2는 전체 객체에 대한 수정이 필요 없지만, O_1객체 전체에 대한 잠금을 확보하여야 한다. 그리고, 두 번째 문제로서 T_3와 T_4는 T_3가 수정 중에 T_4와 어떤 공간 관련성도 없으나 비 일관성(inconsistent)있는 상태에 존재하게 된다. 그리고, T_3와 T_4는 O_1의 부분에 대한 수정작업이 불가능하다. 이것은 T_2가 O_1에 대한 전통적인 잠금 기법 하에서의 배타적 잠금을 획득하기 때문이며, T_3와 T_4는 T_2의 O_1에 대한 잠금이 해제될 때까지 대기해야만 한다. 그러므로, 3개의 트랜잭션은 긴 트랜잭션이기 때문에 T_3와 T_4는 긴 대기시간을 가지게 된다.

이 논문에서 동시에 수정되고자 하는 큰 크기를 가지는 공간 객체를 큰 객체(big object)라 하며 다음과 같이 정의한다.

정의 2. CW를 클라이언트의 작업 영역으로 간주하고, CW.G를 클라이언트 작업 영역의 공간 속성으로 둔다. 큰 객체는 객체의 공간 속성이 클라이언트 작업 영역의 공간 속성과 교차되는 공간 객체를 말하며 형식적으로 다음 조건을 만족한다.

$CW.G \cap O.G \neq \emptyset$

4. 객체 부분 잠금 기법

이 장에서는 큰 객체의 동시 수정을 지원하기 위한 부분 잠금 기법을 제시한다. 사용자가 정의한 큰 객체부분으로 잠금의 단위를 줄이는 것이 기본 아이디어이다.

4.1 부분 객체

부분 객체는 큰 객체로부터 선택된 점의 집합이다. 정의 1에서와 같이 공간 객체는 점의 집합으로 구성된다. 그 점은 실 세계의 위치 정보이다. 큰 객체는 수정을 위하여 여러 개의 부분 객체로 나누어 질 수 있다. 이 논문에서는 큰 객체로부터 선택된 점의 집합을 부분 객체라 하며 정의 3과 같이 정의한다.

정의 3. Client Workspace CW의 Geometry 정보인 CW.G와 객체의 Geometry정보인 O.G의 교차점을 CP라 정의한다. 수정하고자 하는 공간 객체 $O(a_i|a_i = (x_i, y_i))$ where $0<i<n$)에 대하여 부분 객체 PO는 수정하고자 하는 영역 내부의 점의 집합으로 구성되며 다음과 같다.

$PO_i = \{a_i|a_i \in CW.G \cap O.G\} \cup CP$, where $a_i \in O.G$

부분 객체는 원래의 큰 객체로부터 파생되어야 한다. 예를 들어 그림 3(a)에서 도로 객체 O_1은 큰 객체이다. 수정 트랜잭션 T_i는 수정 영역(updating area)에 대한 수정연산을 수행하기 위하여 O_1과 O_2에 대한 수정 영역을 정의한다. 이때 큰 객체 $O_1=\{a_0, a_1, a_2, \cdots a_9\}$의 부분 객체 PO_i는 $\{a_2, a_3, a_4, a_5, a_6\}$로서 정의된다.

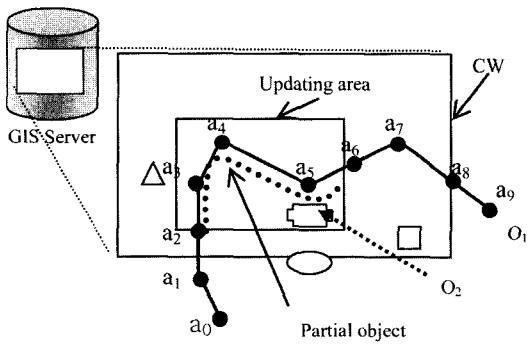


그림 3 부분 객체의 정의

4.2 부분 객체 잠금의 동시성 제어

두 클라이언트가 공간 객체를 동시에 수정할 때, 공간 객체간의 공간 관련성의 충돌이 고려되어야 한다[4]. 그러나, 만일 수정 객체가 큰 객체이면 그 객체에 대한 잠금을 수행할 수 없다. 이 절에서는 큰 객체의 잠금을 지원하기 위한 두 가지 잠금 모드인 PX(Partial eXclusive)잠금과 PR(Partial Region)잠금을 제시한다. 또한 부분 객체의 수정 트랜잭션을 위한 동시성 제어 방법도 제시된다.

부정확한 상태의 가능성이 있기 때문에 공간 관련성을 가지는 두 공간 객체는 동시에 수정되어서는 안 된다[3]. 공간 관련성은 두 공간 객체 사이의 Egenhofer[10]의 공간 관련성을 가진 경우에 정의되며, [10]에서는 8종류의 공간 관련성이 정의되어 있다(Disjoint, Meets, Equal, Inside1, Inside2, Covers1, Covers2, Overlap).

만일 트랜잭션들이 공간 관련성을 가지는 두 개의 부분 객체들을 수정하는 경우 그 수정은 불완전한 상태가 될 수 있다. 그러므로, 두 부분 객체를 일관성 있는 상

태에서 수정하기 위하여, 이 논문에서는 2PC의 확장인 부분 객체의 쓰기 잠금 기법을 도입한다. 이 잠금의 기본 아이디어는 대화식 긴 트랜잭션에서 동시성제어의 단위를 큰 객체의 부분 객체로 제한하는데 있다.

부분 객체의 쓰기 잠금인 PX잠금은 정의 4와 같이 정의된다.

정의 4. PX lock(Partial eXclusive lock)은 큰 객체의 부분 객체에 대하여 배타적 잠금을 설정하는 잠금이다. PX잠금은 공유 잠금에 대하여 호환된다.

두 개의 분리된 부분 객체는 PX잠금을 이용하여 동시에 수정이 가능하다. 그러나, 공간 관련성을 가지는 두 부분 객체는 순차적으로 수정될 수 있다. 예를 들어 그림 4에서 트랜잭션 T_1과 T_3는 큰 객체 O_1의 부분 객체 P_1과 P_3을 동시에 수정할 수 있다. 그러나, T_1과 T_2는 P_1과 P_2사이에 쓰기 잠금의 충돌이 발생하므로 동시 수행이 불가능하다. 그러므로 T_1과 T_2는 순차적으로 실행되어야 한다.

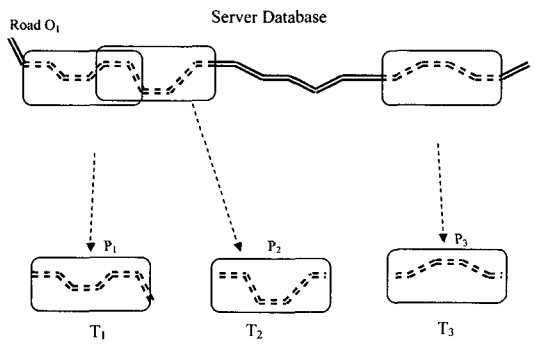


그림 4 부분 객체 잠금

의도 잠금(Intension lock)은 하위 객체에 대하여 쓰기 잠금(Write lock, PX lock등)이 설정되어 있음을 의미한다. 클라이언트들의 수정 트랜잭션이 진행되는 중에 모든 객체가 동시에 수정될 필요가 있는 것이 아니므로 클라이언트 작업 영역 내에 존재하는 전체 객체에 대한 쓰기 잠금은 필요 없다. 그러므로, 의도 잠금(intension lock)은 클라이언트 영역 내에 속하는 객체 집합에 대하여 공유 잠금을 설정하고 쓰기 잠금을 의도한다. 동일한 관점에서 이 논문에서는 큰 객체의 PR(Partial region) 잠금을 정의 5와 같이 정의한다. PR 잠금은 다중 크기 잠금 기법(Multiple granularity locking scheme)의 SIX lock 잠금과 유사하다. 그러나, PR잠금은 대화식 수정작업을 수행하고 있는 다른 트랜잭션의 성능을 증대시키기 위하여 PR잠금과 PX잠금을 허용한다.

정의 5. 트랜잭션 i에 의해서 수정하고자 하는 객체를 R_i라 둘 때, PR잠금은 R_i의 부분 객체 PO_i에 대한 쓰기 잠금을 의도하는 잠금이다. PR잠금은 공유 잠금과 쓰기 잠금에 대하여 호환한다.

PR잠금은 의도 잠금이다. 트랜잭션이 큰 객체의 부분 객체를 수정하고자 하는 경우 그 트랜잭션은 큰 객체에 대하여 PR잠금 여부를 확인해야 한다. 만일 그 객체에 PR잠금이 설정되어 있다면 그 트랜잭션은 부분 객체를 수정하기 위하여 PX잠금의 충돌여부에 대한 조사를 해야만 한다. 만일 충돌이 발생하지 않는다면 그 트랜잭션은 부분 객체에 대하여 PR잠금을 획득한 후 PX잠금을 요청할 수 있다.

예를 들어, 그림 4의 잠금 기법은 그림 5와 같이 기술할 수 있다. 객체 O_1에는 3개의 부분 객체에 대하여 각각 PR잠금이 설정되어 있으며 두 개의 PX잠금이 P_1과 P_3에 설정되어 있다. 그러므로, 부분 객체 P_2의 을 위하여 트랜잭션 T_2는 T_1에 의해 설정된 PX 해제를 대기해야 한다.

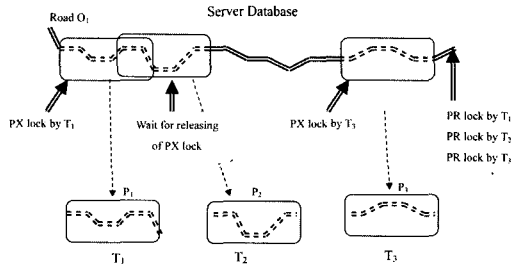


그림 5 그림 4에서의 PR, PX잠금 예

잠금 호환성 테이블은 표 1에서와 같이 PX와 PR자금을 포함하기 위하여 확장되었다. 표 1의 잠금 호환성에서 PR잠금은 읽기 잠금과 호환된다. 그러나, PR잠금은 잠금의 크기와 잠금의 의도성 측면에서 읽기 잠금과 다르다.

표 1 PX와 PR잠금을 포함한 2PC 잠금 호환성 테이블

	READ	PR	PX	WRITE
READ	Y	Y	Y	N
PR	Y	Y	Y	N
PX	Y	Y	N	N
WRITE	N	N	N	N

PR, PX잠금은 잠금 기법 자체만으로는 기존의 잠금 기법에 의한 dead lock으로부터 자유롭지 않다(dead-lock free). 이것은 공간 객체의 다양한 모양에 의하여 한 객체의 여러 부분에 걸쳐 동시에 PX와 PR잠금이 설정될 수 있기 때문이다. 그러나, 이 잠금은 전역 트랜잭션의 dead lock 제어 기법(wait-die/wound-wait)들

에 의하여 제어 가능하다. 즉, 여러 클라이언트의 잠금 설정은 사용자에 의하여 제어되며, 다른 사용자에 의한 dead lock은 막을 수 있다.

5. 부분 잠금에 의한 동시성 제어

이 장에서는 부분 잠금(PR lock)을 이용한 부분 객체의 동시성 제어를 위한 방법을 제시한다. 또한 부분 잠금의 알고리즘도 기술된다.

5.1 부분 잠금을 이용한 동시성 제어

부분 객체의 수정 시 충돌현상은 그림 6에서 보는 바와 같이 잠금 영역의 겹침 조건에 따라 분류될 수 있다. 만일 겹치는 영역이 없는 경우, 두 트랜잭션은 각각의 부분객체를 독립적으로 수정할 수 있다. 그림 6(h) 그 외의 경우 그림 6에서 보는 바와 같이 7가지 경우로 나눌 수 있다. 그러나 주목할 만한 점은 만일 트랜잭션이 PX잠금이 설정되어있는 경우에 PX잠금을 설정하는 경우가 아니라면 그 트랜잭션은 다른 트랜잭션의 잠금 영역에 대한 참조 없이 수정할 수 있다. 그러므로, 수정 충돌은 PX잠금의 충돌 시에만 발생한다.

두 잠금 영역간에 겹침이 있는 경우는 겹침 상태에 따라 완전 겹침, 부분 겹침, 독립 상태의 3가지 경우로 나누어 볼 수 있다.

그림 6의 (a)와 (b)에서 트랜잭션 T_i와 T_j는 완전 겹침 상태이다. 이 경우에는 CR_i와 CR_j는 완전 포함(inside)나 동일(equal)의 공간 관련성을 가진다. 이때, 두 트랜잭션은 PX잠금의 충돌로 인하여 동시에 실행될 수 없으며, 순차적으로 실행되어야 한다.

부분적인 겹침의 경우는 그림 6의 (c), (d), (e), (f), (g)이다. 이들 경우에는 겹침 영역의 존재 유무에 따라 두 경우로 나눌 수 있다. 만일 겹침이 그림 6의 (e), (f), (g)와 같은 관계를 가지는 경우, 즉 겹치는 영역이 없는 경우에는 동시 수정이 가능하다.

그러나, 그림 6의 (c), (d)의 경우에는 좀 더 복잡하다. 그림 6의 (d)의 경우 트랜잭션은 각각의 부분객체에 대한 잠금을 수행할 수 있으며, 자신의 영역에 대한 수정이 가능하다. 반면, (c)의 경우에는 동시수정은 협동작업이 없는 경우 수행이 불가능 하다. 경우 (c)와 (d)는 교차점 CP의 CR에 다한 포함 여부를 통하여 알 수 있다. 즉, 트랜잭션 T_i, T_j에 대하여 수정 객체 BO가 CP_{i1}과 CP_{i2}로 구성될 때, 만약 CP_{i1}이나 CP_{i2}가 CR_j에 속하면 단 하나의 트랜잭션만 수행 가능하다. 즉, 부분 객체의 교차점은 충돌 발생 여부를 판단하는 기준이 된다. 그림 7은 부분 객체 수정의 상황을 보여준다. 그림 7(a)에서 PR잠금 영역은 두 개의 트랜잭션으로 분리되어

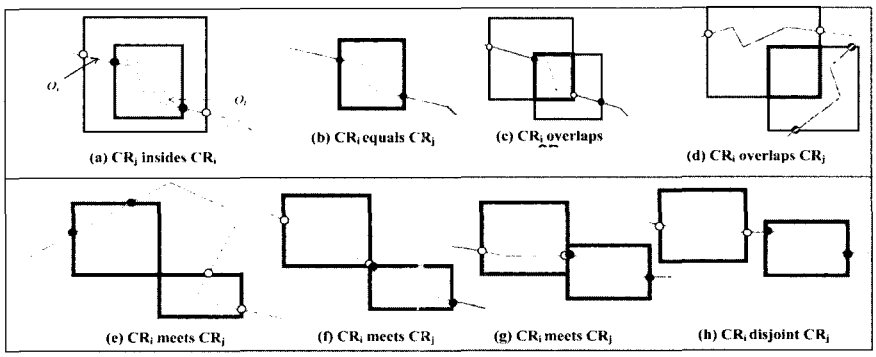
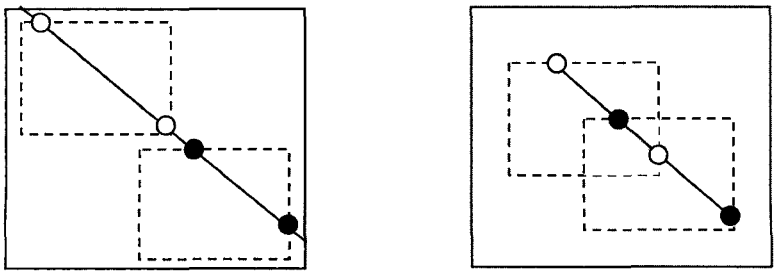


그림 6 수정 충돌 가능성과 큰 객체 수정시 겹침 영역의 존재에 대한 고려



(a) PR Region is separated (b) PR Region is overlapped

그림 7 PR잠금 영역과 PR잠금

있으며, 두 트랜잭션 T_i와 T_j는 동시 수행이 가능하다. 반면 그림 7의 (b)는 동시수행이 불가능하며, 협동작업이 필요하다.

표 2에서 부분 객체의 잠금 충돌의 경우를 요약하였다. 어떤 경우에든, 만약 큰 객체가 단 하나의 트랜잭션의 잠금 영역과 겹치는 경우에는 그 객체는 동시 수정이 가능하다.

5.2 부분 잠금 알고리즘

이 절에서는 잠금 알고리즘을 기술한다. 우선 잠금 정보 테이블에 대하여 기술한다. 잠금 정보 테이블은 표 3에서와 같이 ObjectID, LockType, TR_ID, PartialObject정보를 포함한다. PartialObject 속성은 LockType속성이 PX잠금인 경우에만 유효하며, 잠금 정보 테이블은 부분 객체의 잠금 정보를 저장하기 위한 메타 테이블이다.

PR잠금을 위한 잠금 알고리즘은 그림 8과 같다. 그림 8에서 보는 바와 같이 PR잠금은 읽기 잠금과 비슷하지만, WRITE잠금을 체크해야 한다.

```
function PR_LOCK(TR_ID, ObjectID)
  var found : bool
  begin
    found := Search WRITE lock on ObjectID in lock table
    if not found then
    begin
      Add TR_ID to PR_Lock_indicator of ObjectID
    end
    else
      return FALSE
return TRUE
```

그림 8 PR잠금 알고리즘

PX잠금은 큰 객체에 대한 PR잠금을 먼저 검사해야 한다. PX잠금 알고리즘의 그림 9와 같으며, PR잠금보다 복잡하다. 그림 9에서 PX잠금 함수는 쓰기 잠금에 대한 충돌을 먼저 검사한 후 해당 객체에 대한 의도 잠금(intention lock)의 존재 유무를 확인한다. 만약 존재한다면 이 함수는 요청된 부분 객체에 대한 잠금이 이전의 PX잠금과 충돌이 발생하는지를 검사한다. 이 검사에서 존재하지 않는다는 판정이 난 경우에 잠금을 요청한 트랜잭션에 PX잠금이 허용된다. 만약 그렇지 않다면 PX_LOCK함수는 잠금 요청을 거부한다.

```
function PX_LOCK(TR_ID, ObjID,rPartialObject)
var found: bool
var PartialObject
begin
if WRITE lock is granted on ObjID then
        reutn false;
if PR lock is granted on ObjID then
begin
        TransactionIDs = GetTransactionsOf(ObjID)
        for all TR_ID in
        begin
            PartialObject = GetPartialObject (TR_ID,ObjID)
            if IsOverlap(rPartialObject, PartialObject) then
                return false;
            end if
        end
end
AddLockInformationTable(TR_ID,ObjID, rPartialObject)
return true
end
```

그림 9 PX잠금 알고리즘

표 3 잠금 정보 테이블

이름	설명
ObjectID	잠금이 설정된 객체 ID
LockType	잠금의 종류(PX, PR, WRITE and READ)
TR_ID	잠금을 소유한 transaction id
PartialObject	PX잠금이 설정된 부분 객체

5.3 부분 객체 잠금 프로토콜의 일관성 단계

트랜잭션의 직렬화 상태를 나타내는 일관성 단계(consistency level)는 크게 4가지 단계로 구성된다. 단계 0는 비 완료 데이타읽기(uncommitted data read)단계로 최저 하위 단계이다. 단계 1은 완료 데이타 읽기(committed data read)단계로 트랜잭션이 비 완료 데이타읽기 오류를 방지하고, 완료된 데이타만 읽을수 있도록 하는 단계이다. 단계 2는 반복 읽기(repeatable read)단계로 비일관적 분석(inconsistent analysis)오류를 방지하고 트랜잭션 내에서 반복하여 읽기 연산을 수행하는 경

표 2 잠금 충돌 감지

영역잠금의 관련성	큰 객체와 수정영역간의 겹침 상태	잠금 충돌	수정 방법	경우 (그림 6)
Disjoint	Disjoing	Not Conflict	Concurrent update	(h)
Inside or equal	Overlapped with all	Conflict	Serial update with lock	(a),(b)
	Overlap with 1	Not conflict	Concurrent update	
Meet	Overlapped with all	Not conflict	Concurrent update	(e),(f),(g)
	Overlap with 1	Not conflict	Concurrent update	
Overlap	Overlap with shared region	Conflict	Concurrent update with clipping	(c)
	Overlap with independent region	Not conflict	Concurrent update with clipping	(d)

우에 동일한 값을 읽을 수 있도록 보장하는 단계이다. 그리고 단계 3은 직렬화(serializable)단계로 팬텀(phantom)오류를 방지하는 단계이다. 하위 단계에서 발생하는 오류들은 상위단계에서 방지되기 때문에 상위단계일수록 보다 더 엄격하다.

부분 객체 잠금 기법 하에서의 지역 트랜잭션들은 전역 트랜잭션에 의하여 제어되며, 수정 객체들은 PX잠금에 의하여 보호된다. 트랜잭션의 읽기 연산은 트랜잭션의 시작 시 전송되어, 캐싱된 지역 데이타에 대하여 수행되며, 쓰기 연산의 결과는 완료되기 전까지 PX잠금에 의하여 유지되며, 지역 저장장치 또는 임시 버퍼에서 유지된다. 즉, 정의 3에 따라 정의된 부분 객체 P_{BOI}는 공간 객체 UO의 부분 집합으로 구성되고, 트랜잭션 완료 후, 서버에 병합될 때까지 잠금에 의하여 다른 트랜잭션에게 노출되지 않는다. 따라서 트랜잭션의 비완료 데이타는 잠금에 의하여 다른 트랜잭션이 접근할 수 없고 읽기 집합의 모든 객체는 완료된 데이타로만 구성되기 때문에 부분 객체 잠금 프로토콜은 완료 데이타 읽기 단계를 보장한다.

비일관적 분석 오류는 참조 무결성(referential integrity)을 유지하지 못하는 경우에 발생하는 오류이다. 공간 데이타 측면에서 공간 객체간의 참조 무결성은 공간 관련성의 일관성 유지 여부에 있다. PX잠금에 의해 정의된 부분 객체 집합은 표 2의 잠금 충돌 감지 조건에 의하여 다른 객체와 충돌하지 않는다. 따라서, 부분 객체 잠금에 의한 트랜잭션은 비일관적 분석오류 발생이 없으며, 다음 단계의 트랜잭션에서 일관된 데이타로 저장되기 때문에 반복 읽기 단계를 보장한다.

부분 객체 잠금 트랜잭션 수행 시 팬텀 오류는 발생할 수 없다. 팬텀 오류는 트랜잭션의 수행 중에 다른 트랜잭션의 삽입연산에 의해 발생하는 오류이다. 그러나, 부분 객체 P_{BOi}의 수정 중에는 P_{BOi}는 PX잠금에 의하여 다른 트랜잭션에 의한 접근이 불가능하므로, 수행 중에는 변하지 않는다. 즉, 수행중인 트랜잭션은 다른 트랜잭션의 부분 객체에 대하여 접근할 수 없다. 따라서 트랜잭션은 직렬화 단계를 보장한다.[11]

6. 구현 및 실험

6.1 시스템 구조

그림 10은 본 논문의 구현 환경과 시스템 구조이다. 서버는 Linux서버를 사용했고, 클라이언트는 Windows XP®환경에서 구현했다. 공간 데이타베이스는 관계형 DBMS인 MySql®를 사용했다. 즉, 공간 객체는 레코드 형식으로 데이타베이스에 저장되어 있고, 공간 객체의 기하 속성은 BLOB형식으로 한 레코드의 한 필드에 저장된다.

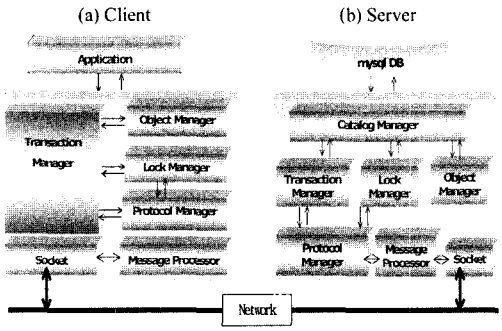


그림 10 구현 환경 및 시스템 구조

서버는 그림 10에서 보여주는 바와 같이 구성되어 있고 각각의 기능은 표 4와 같다.

표 4 서버 구성 모듈

모듈 이름	역할
카탈로그 관리 모듈 (Catalog Manger)	메타 정보 관리
잠금 관리 모듈 (Lock Manager)	잠금 정보를 설정하거나 가져오는 모듈
트랜잭션 관리모듈 (Transaction Manager)	트랜잭션의 상태 및 클라이언트 정보 등을 관리
객체 관리 모듈 (Object Manager)	부분 객체에 대한 전반 작업 관리
프로토콜 관리 모듈 (Protocol Manager)	전체적인 프로토콜을 관리
메시지 관리 모듈 (Message Processor)	메시지 처리 및 패킷을 구성 또는 분석

클라이언트는 5개의 모듈로 구성되며 각각의 기능은 표 5와 같다.

표 5 클라이언트 구성 모듈

모듈 이름	역할
트랜잭션 관리 모듈 (Transaction Manager)	클라이언트에서 트랜잭션 정보 등을 관리
객체 관리 모듈 (Object Manager)	클라이언트에서 갱신하는 부분 객체에 대한 관리
프로토콜 관리 모듈 (Protocol Manager)	서버에서 보내 온 메시지 처리
잠금 관리 모듈 (Lock Manager)	서버에 설정된 잠금 정보를 임시 관리
메시지 관리 모듈 (Message Processor)	서버 메시지 분석

6.2 구현 예시

6.2.1 동시 변경 시 충돌 없는 경우

그림 11은 구현된 시스템을 적용하여 큰 객체를 부분 객체로 재구성하여 공간 데이타 변경 작업을 진행 할

때, 부분 객체가 서로 분리되어 있고 변경 충돌이 없는 경우, 즉, 동시 수정이 가능한 경우를 보여주고 있다.

두 클라이언트 $Client_1$, $Client_2$가 BO의 서로 겹치는 두 부분 객체 BO_i, BO_j를 수정하고 있다. (a)는 $Client_1$과 $Client_2$가 작업을 시작할 때의 초기 화면이다. $Client_1$, $Client_2$는 동시에 작업을 시작하고 수정 영역을 선택하여 PR 잠금 설정을 서버에 요청한다(b). 서버는 $Client_1$, $Client_2$가 요청하는 PR 잠금 정보를 기록하고 잠금 수용 메시지를 보낸다(c). $Client_1$, $Client_2$는 PR 잠금 설정이 성공 했다는 것을 확인하고 각자 부분 객체 BO_i, BO_j에 대한 PX 잠금을 서버에 요청한다. 부분 객체 BO_i, BO_j는 서로 분리 되어 있고 동시 수정 시 변경 충돌 발생하는 가능성이 없어서 (d)에서는 $Client_1$, $Client_2$가 같은 객체 BO의 서로 다른 부분 즉, 부분 객체 BO_i, BO_j를 동시에 수정하는 모습을 보여 주고 있다.

6.2.2 동시 변경 시 충돌이 있는 경우

그림 12는 두 부분 객체를 동시 수정 시 변경 충돌이 발생 할 수 있는 경우이다. 두 클라이언트 Client1, Client2가 BO의 서로 겹치는 두 부분 객체 BO_i, BO_j를 수정하고 있다. Client1이 작업을 먼저 시작하고 수정 영역을 선택하여 PR 잠금 설정을 서버에 요청한다(b-1). 이와 동시에 Client2도 PX 잠금 영역을 설정하고 PR 잠금을 서버에 요청했다(b-2). PR 잠금은 본 논문에서 정의한 바와 같이 서로 겹쳐도 충돌이 없다. 즉, 서버는 Client1, Client2가 서로 겹치는 영역에 대해 PR 잠금 요청을 하는 것을 수용하고 PR 잠금 성공을 보내준다(1-1, 2-2). PX 잠금에 성공한 Client1, Client2는 각자 부분 객체 BO_i, BO_j에 대해 PX 잠금을 서버에 요청한다. 이 때, 먼저 요청한 Client1는 서버에서 PX 잠금 충돌이 없는 것을 확인하고 PX 잠금을 허락(1-3)했지만, Client2의 PX 잠금 요청은 Client1의 PX 잠금과 충돌이 있는 것을 확인하고 대기 메시지를 보낸다(2-3). Client1는 부분 객체 BO_i에서 클리핑 포인트와 인접한 Line Segment를 제외한(e-1) 기타 부분에 대한 수정 작업 후 PX잠금을 해제한다.(1-4) 서버에서는 메타 정보를 통해 Wake_Up해야 하는 트랜잭션이 존재하는지를 알아보고 Client2에게 Wake_Up 메시지와 변경된 부분 객체 정보를 함께 전송한다(2-4). 여기서 수정 작업을 끝낸 Client1는 PR 잠금도 해제하고, 수정 작업을 완료한다.(1-6). Client2는 Client1과 같은 순서로 수정 작업을 진행하고(2-5) PX 잠금을 해제한 후, 수정 트랜잭션을 종료한다.(2-6) 최종적으로 큰 객체 BO가 수정된 모습은 (1-6, 2-6)과 같다. PR 잠금 영역이 겹치지 않는 경우 또는 겹치지만 부분 객체 사이에 충돌이 없는 경우는 각 트랜잭션이 동시 수행이 가능하여 지도 데이타의 수정 동시성을 높이는 효과를 기대할 수 있다.

7. 결론 및 향후 연구

클라이언트-서버 지리정보시스템 환경에서 공간 객체의 기하적 속성에 대한 갱신 작업을 진행할 때, 공간 데이타 특성으로 인해 전통적인 속성 데이타 갱신 기법만으로는 효율적으로 갱신 작업을 진행하기 어렵다. 즉, 공간 데이타는 갱신 객체만 아니라, 갱신 객체와 공간 관련성이 있는 기타 객체들과 함께 고려하여 작업을 진행해야 한다. 특히 공간 데이타의 기하 속성은 한 클라이언트 갱신 화면에 객체 전체를 보여 줄 수 없는 경우가 많다. 기존에 연구해온 잠금 기법으로는 같은 객체에 대한 잠금 요청은 순차적으로 진행 할 수 밖에 없어서 긴 대기 시간을 초래한다.

본 논문에서 여러 클라이언트가 동일 공간 객체의 서로 다른 부분에 대한 갱신 작업을 요청할 때, 갱신 작업 동시성을 향상하기 위하여 부분 객체 개념을 제안하여 이에 기반 한 공간 객체 부분 잠금 기법을 제안하였다. 즉, 공간 객체를 잠금의 최소 단위로 가정하는 기존 연구와 달리, 갱신 영역에 따라 공간 객체를 동적으로 부분 객체로 분리하여 갱신 작업을 진행

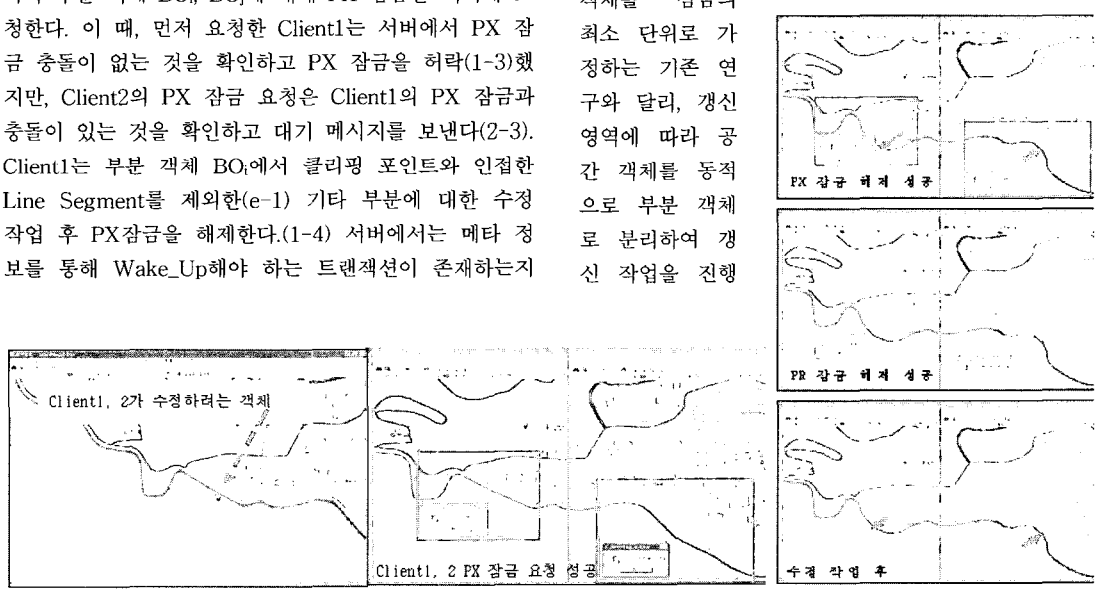


그림 11 부분 객체 동시 수정 시 변경 충돌이 없는 경우

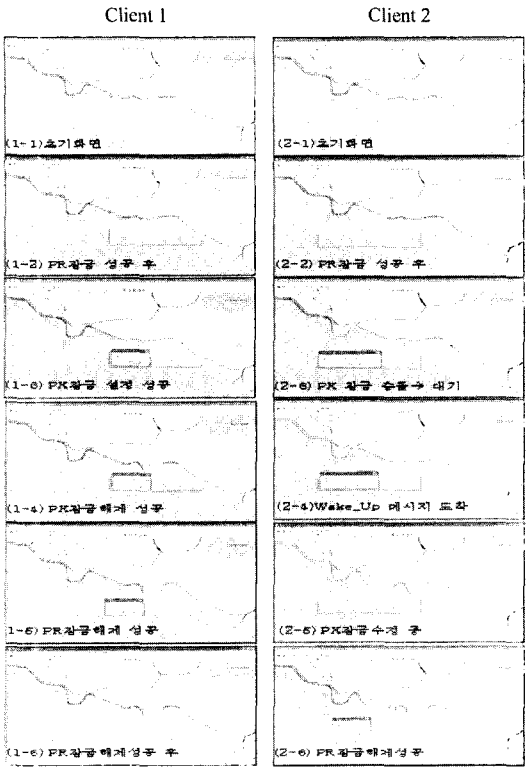


그림 12 변경 충돌이 있는 경우의 부분 객체 수정

한다. 갱신을 마친 부분 객체는 서버에서 다시 큰 객체로 병합하여 최종적으로 공간 데이타베이스에 반영하게 된다.

이 기법을 적용하여 아래와 같은 이점을 얻을 수 있다. 첫째, 한 공간 객체의 서로 다른 부분에 대하여 여러 클라이언트가 동시에 갱신 작업을 진행 할 수 있으며 갱신 작업의 동시성을 높인다. 둘째, 클라이언트에서 갱신 작업을 진행하기 위하여 공간 연산을 수행할 때, 전체 객체가 아닌 부분 객체만을 이용하여 공간 연산을 진행함으로써 작업 효율성을 높일 수 있다. 셋째, 큰 객체에 대한 갱신이 발생할 때마다 데이타를 데이타베이스에 반영하기 위하여 디스크에 접근을 하는 것이 아니라, 각 클라이언트가 동일 공간 객체에 대한 갱신 작업이 끝난 후, 한 번에 변경 결과를 공간 데이타베이스에 반영하여 작업 효율성을 높인다.

제안된 부분 객체 잠금 기법은 동일 객체의 서로 다른 부분을 동시에 갱신할 수 있다는 면에서는 갱신 작업 효율성을 높일 수 있는 장점이 있는 반면, 큰 객체를 구성하는 모든 점의 좌표를 일일이 조사하는 문제가 있다. 향후 연구에서는 부분 객체를 만들기 위한 클리핑 작업을 효율적으로 진행하기 위하여 큰 객체를 TR*-tree등 인덱싱 기법을 통하여 효율적으로 관리하는 방법을 연구 할 필요가 있다.

참 고 문 헌

[1] P.A. Bernstein, and N.Goodman, "An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases," ACM Tran. Database Systems, No.4(9), pp. 596-615, 1984.

[2] S.H. Phatak, and B.R.Badrinath, "Multiversion Reconciliation for Mobile Databases," Int. Conf. On Data Engineering. pp. 582-589.

[3] J.Gray, P.Helland, and D.Shasha, "The Dangers of Replication and a Solution," Proc. Of the 1996 ACM SIGMOD, pp. 173-182, 1996.

[4] J.O. Choi, Y.S. Shin, and B.H. Hong, "Update Propagation of Replicated Data in Distributed Spatial Databases," Proc. of 10th International Conference on DEXA '99, pp. 952-963, 1999.

[5] A.S. Oh, J.O. Choi, and B. H. Hong, "An Incremental Update Propagation Scheme for a Cooperative Transaction Model," Proc. of International Workshop on DEXA '96, pp. 353-362, 1996.

[6] W. Kim, J. Banerjee, H.T Chou, J.E Garza, and D.Woelk, "Composite Object support in an Object-Oriented Database Systems," in Proc. 2nd Intl. Conf. on Object-Oriented Programming Systems, Languages, and Applications(OOPSLA), Orlando, Florida, pp. 118-125, 1987.

[7] J. Banerjee, H. T. Chou, J. F. Garza, W. Kim, D. Woelk, N. Ballou and H. J. Kim, "Data Model Issues for Object-Oriented Applications," ACM Trans. on Office Information Systems, No.5(1), pp. 3-26, 1987.

[8] J.F. Garza, and W. Kim, "Transaction Management in an Object-Oriented Database System," ACM-SIGMOD Intl. Conf. on Management of Data, Chicago, pp. 37-45, 1988.

[9] A. Silberschatz, H.F. Korth, Sudarshan, "Database system concepts", McGraw-Hill, 1997, pp. 487-450.

[10] M.J. Egenhofer, "Reasoning about binary topological relations," 2th Int. Symposium, SSD'91, pp. 143-160, 1991.

[11] Y.D. Seo, D.H. Kim, and B.H. Hong," Concurrent Updating of Large Spatial Objects," 9th International Conference on Database Systems for Advanced Applications, DASFAA 2004, pp. 325-330, 2004.

서 영 덕

1997년 부산대학교 컴퓨터공학과(학사) 1999년 부산대학교 컴퓨터공학과(석사) 2004년 부산대학교 컴퓨터공학과(박사)관심분야는 지리정보시스템, 모바일 GIS, 병렬데이타베이스, 이동체 색인, 병렬색인

김 동 현

1995년 부산대학교 컴퓨터공학과(학사) 1997년 부산대학교 컴퓨터공학과(석사) 2003년 부산대학교 컴퓨터공학과(박사) 2004년~현재 동서대학교 소프트웨어전문대학원 전임강사관심분야는 지리정보시스템, 모바일 GIS, 모바일 트랜잭션, 이동체 색인

홍 봉 희

1982년 서울대학교 전자계산기공학과 졸업(학사). 1984년 서울대학교 전자계산기공학과 졸업(석사). 1988년 서울대학교 전자계산기공학과 졸업(박사). 1987년~현재 부산대학교 공과대학 컴퓨터공학과 교수/부산대학교 컴퓨터 및 정보통신연구소 책임연구원. 관심분야는 이동체 데이타베이스, 모바일 데이타베이스, 공간 데이타베이스