모바일 GIS를 위한 대리 트랜잭션 관리자의 설계 및 구현

(Design and Implementation of the Surrogate Transaction Manager for Mobile GIS)

반 재 훈 † 문 선 희 †† 김 동 현 ††† 홍 봉 희 ††††
(ChaeHoon Ban) (SunHee Moon) (DongHyun Kim) (BongHee Hong)

요 약 이동 클라이언트를 이용한 공간 데이타의 변경 트랜잭션은 서버와 통신을 단절한 상태에서 사용자와의 상호 연산을 통하여 수정하는 긴 트랜잭션이다. 따라서 트랜잭션의 동시성 제어를 위해 잠금을 이용한 비관적 기법을 사용하면 이동 클라이언트가 잠금을 획득하기 위해 오랜 시간동안 대기해야 하므로 일반적으로 검증 작업을 이용한 낙관적 기법이 적합하다.

본 논문에서는 이동 클라이언트를 이용해 공간 데이타를 변경하는데 적합한 S-S-M(서버-대리 PC-이동 클라이언트) 구조를 위한 대리 트랜잭션 모델을 제안하고 모델에 따른 관리자를 구현한다. 서버와 이동 클라이언트가 대리 PC를 통해 통신하는 이 구조에서 대리 트랜잭션의 동시성 제어를 위해 공간 객체간의 위상 관계인 공간 관련성을 고려하여 기존의 검증 조건을 확장한다. 또한, 충돌이 발생한 트랜잭션의 완료 비용을 최소화하기 위해 대리 PC에서 충돌이 발생한 객체에 대하여 조정 작업을 수행하며 이를 지원하기 위한 확장된 완료 프로토콜을 제시한다.

키워드 : 모바일GIS, 공간객체변경, 대리트랜잭션, 일관성제어, 완료규약

Abstract Transactions of updating spatial data with mobile clients are long transactions because a user disconnected from a server surveys real features and updates them. In this environment, it is appropriate to exploit the optimistic approach based on the validation test in order to control the concurrency of transactions. On the contrary, the pessimistic concurrency control scheme makes transactions wait for a long time due to the lock.

In this paper, we propose the surrogate transaction model and implement its manager for the S-S-M(Server-Surrogate PC-Mobile Client) structure which is appropriate for updating spatial data in mobile environments. In the S-S-M structure, the mobile client communicates with the server by the surrogate PC. We extend the validation condition in consideration of spatial relationships between spatial objects in this model. We also present the commit protocol where the user of a surrogate PC adjusts objects of the conflicted surrogate transaction to minimize costs for the abortion of the transaction.

Key words : mobile GIS, spatial object update, surrogate transaction, consistency control, commit protocol

† 정 회 원 : 경남정보대학 인터넷응용제열 교수
chban@kit.ac.kr
†† 비 회 원 : 삼성전자
suhemoon@pusan.ac.kr
††† 학생회원 : 부산대학교 컴퓨터공학과
pusrover@pusan.ac.kr
†††† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@pusan.ac.kr
논문접수 : 2002년 3월 29일
심사완료 : 2003년 1월 28일

1. 서 론

이동 컴퓨팅 환경이 급속하게 발전하면서 필요한 컴퓨팅 환경을 언제, 어디서나 사용자에게 제공하기 위해 휴대폰 또는 PDA와 같은 이동 클라이언트에서 제공되는 서비스에 대한 연구가 활발히 진행중이다. 특히, GIS 분야에서 이동 클라이언트는 정확한 데이타를 실시간으로 입력 및 검증하기 위해 다양하게 사용될 수 있다. 예

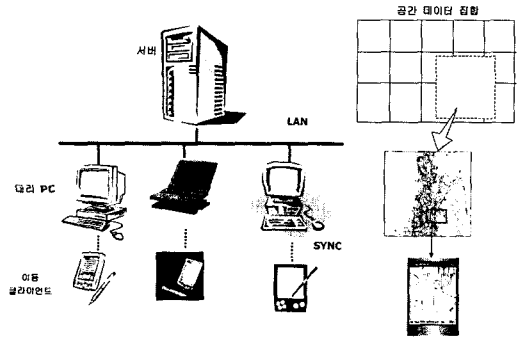


그림 1 S-S-M(Server-Surrogate pc-Mobile client) 구조

를 들어, 시설물 관리 시스템에서 이동 클라이언트를 이용하여 현장에서 서버의 데이타와 실제 시설물을 비교하면서 새로운 시설물 데이타를 입력하거나 또는 잘못된 데이타를 수정할 수 있다.

그러나 현재의 이동 컴퓨팅 환경은 통신 단절, 배터리 부족 그리고 비싼 통신 요금과 같은 제약을 가지고 있기 때문에 현실적으로 무선 네트워크를 기반으로 한 S-M(Server-Mobile client)구조보다는 그림 1의 3-계층 구조인 S-S-M(Server-Surrogate pc-Mobile client) 환경에서 공간 데이타 변경 작업이 수행된다. S-S-M구조는 공간 데이타베이스를 기반으로 한 GIS서버, 공간 데이타 수정 작업을 수행하는 이동 클라이언트 그리고 이동 클라이언트를 대신하여 변경 내용을 서버에 반영하기 위한 대리 PC로 구성된다. 서버와 대리 PC는 랜(LAN)과 같은 유선 네트워크를 통하여 연결되어 있으며 대리 PC와 이동 클라이언트는 USB 케이블로 연결되어 싱크(synchronization)기능을 이용하여 통신한다.

S-S-M 구조에서 서버와 이동 클라이언트는 대리 PC를 통해 데이타를 교환하기 때문에 무선 네트워크의 제약 조건이 적용되지 않는다. 그러나 기존의 2 계층 구조와는 달리 대리 PC가 중간 계층으로써 구성되기 때문에 대리 PC에서 수행되면서 이동 클라이언트에서 수행된 변경 트랜잭션들을 하나의 단위로 대행할 수 있는 트랜잭션이 필요하다. 이 논문에서는 3 계층으로 구성된 S-S-M 환경을 고려하여 대리 PC에서 이동 클라이언트의 변경 트랜잭션들을 낙관적 기법을 이용하여 대행하는 대리 트랜잭션 모델을 제시한다.

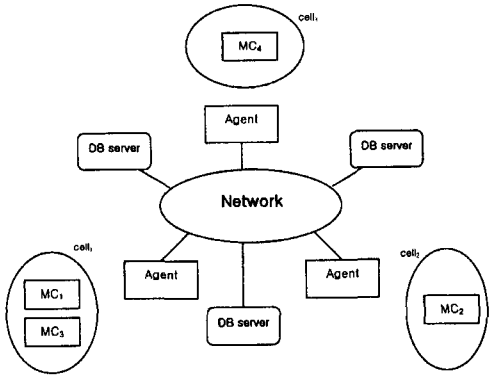
대리 트랜잭션은 이동 클라이언트의 이동 트랜잭션의 집합으로 구성된 상위 트랜잭션으로 대리 PC에서 이동 트랜잭션들을 대행한다. 이동 트랜잭션들은 이동 클라이언트에서 수행되며 서버와 연결이 단절된 상태에서 지역 저장장치의 공간 데이타를 수정한다. 모든 이동 트랜잭션의 작업이 완료되면 이동 클라이언트는 대리 PC에 연결되며 대리 트랜잭션이 이동 트랜잭션을 대행하여 서버에 변경 내용의 완료를 요청한다. 긴 대기 문제 및 단절 상태를 고려하기 위하여 이동 트랜잭션의 대행인 대리 트랜잭션은 낙관적 기법인 검증 기법(validation scheme)을 이용한다. 그러나 기존의 검증 조건은 공간 관련성에 의한 변경 충돌을 검사할 수 없기 때문에 이 논문에서는 기존의 검증 조건을 확장하여 공간 관련성에 의한 변경 충돌을 포함한 검증 조건을 제시한다. 그리고 대리 트랜잭션의 변경 충돌이 발생한 경우에 해당 트랜잭션을 완료하기 위하여 조정 단계를 포함한 확장된 완료 프로토콜을 제시한다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구를 설명하고 3장에서는 대리 트랜잭션의 개념을 정의한다. 그리고 4장에서는 대리 트랜잭션들간의 동시성을 제어하고 공간 관련성을 고려한 확장된 검증 조건을 제시하며 5장에서 충돌된 완료하기 위한 조정 단계를 포함한 완료 프로토콜을 기술한다. 그리고 6장에서는 제시된 대리 트랜잭션 모델에 따른 관리자의 설계 및 구현 내용을 기술한다. 마지막으로 7장에서는 결론을 기술한다.

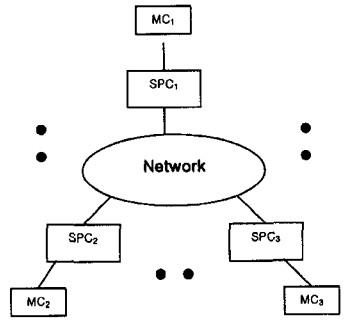
2. 관련연구

유선 네트워크 환경과는 달리 이동 컴퓨팅 환경에서는 예측할 수 없는 끊김(unforeseeable disconnection), 배터리 부족, 한정된 메모리 그리고 통신 과금등의 제약 조건이 있다. 이러한 이동 환경의 제약을 보완하기 위해 계층 구조 형태인 그림 2-(a)의 에이전트-기반(agent-based) 이동 컴퓨팅 시스템이 제시되었다[1]. 에이전트-기반의 계층 구조는 이동 클라이언트, 에이전트 그리고 서버로 이루어진 구조로서 서버와 에이전트는 유선 네트워크로 연결되며 이동 클라이언트는 에이전트에서 전적으로 관리한다. 따라서 이동 클라이언트가 서버와 단절되어 있는 상황에서도 이동 클라이언트는 에이전트를 통해 저장된 데이타가 유효한지 확인하고, 접근하는 것이 가능하다.

에이전트-기반의 시스템에서 이동 클라이언트에 저장된 데이타 집합은 에이전트 데이타 집합의 부분 집합이고 또한 에이전트의 데이타 집합은 서버 데이타 집합의 부분 집합이다. 에이전트는 이동 클라이언트에서 수행된 변경 트랜잭션들 간의 충돌 검사를 처리하며 서버는 에이전트들 간의 전체 트랜잭션을 관리한다. 즉 이동 클라



(a) agent-based mobile computing system



(b) server-surrogate PC-mobile client system

그림 2 에이전트-기반의 이동 컴퓨팅 시스템과 S-S-M 구조의 비교

이언트의 변경 트랜잭션에 대한 처리를 서버에서 모두 처리한다면 서버의 과부화를 초래하므로 에이전트는 서버로 전달되는 정보들을 여과한다[2]. 그러나 이 기법은 기존의 클라이언트-서버 구조를 계속 유지하기 위해 에이전트와 이동 클라이언트간에 무선 네트워크를 사용하기 때문에 무선 네트워크의 제약 조건에 따른 문제점을 그대로 유지하고 있는 문제가 있다.

이 논문에서는 이동 컴퓨팅 환경에서의 공간 데이타 변경을 위하여 에이전트 기반의 계층 구조를 확장한 서버-대리 PC-이동 클라이언트의 S-S-M 구조를 제안한다. 이 구조에서는 접속 끊김 등의 무선 통신의 제약을 고려하여 대리 PC와 이동 클라이언트 사이의 통신은 싱크로 대신한다.

클라이언트-서버 환경이나 분산 환경에서 비관적 동시성 제어 기법인 잠금을 사용하여 협동 작업을 통해서 공간 데이타를 변경하는 기법들이 제시되었다[3,4,5]. 이 기법에서는 특히 공간 객체들간의 관련성을 고려한 잠금 기법을 제시하여 클라이언트들간의 공간 데이타 수정을 위한 일관성 및 동시성을 제어하였다. 그러나 이 방법은 이동 클라이언트를 사용하여 공간 데이타를 수정하는 환경에서는 적합하지 않다. 이동 클라이언트의 경우 현장에서 실제 지형 지물을 보면서 수행하는 현장 작업에 사용되므로 잠금 기법을 사용하면 동시성이 현저히 떨어지기 때문이다.

3. 대리 트랜잭션

S-S-M 구조는 GIS 서버와 다수의 대리 PC 및 이동 클라이언트들로 구성되며 서버와 이동 클라이언트는 대리 PC를 통해서만 데이타 교환이 가능하다. 따라서 이동 클라이언트가 현장에서 변경 작업을 수행할 때 서버와 물리적으로 단절되어 있기 때문에 이동 클라이언트의 공간 데이타 변경 작업 기간은 서버와의 데이타 교환이 불가능한 단절 상태가 된다.

이 논문에서 이동 클라이언트에서 수행된 트랜잭션을 이동 트랜잭션이라 하고 다음과 같이 정의한다.

> 정의 1 : 이동 트랜잭션(Mobile Transaction, MT)은 단절 상태인 이동 클라이언트에서 수행되는 공간 데이타 변경 트랜잭션으로 다음의 작업으로 구성된다.
> ◇ 대리 PC와 싱크하여 공간 데이타를 이동 클라이언트로 다운로드
> ◇ 이동 클라이언트의 공간 데이타 변경(삽입, 삭제, 수정)
> ◇ 공간 데이타 변경 작업을 대리 PC로 업로드

이동 트랜잭션은 변경 작업이 완료되어 대리 PC에 접속할 때까지 서버와 물리적으로 단절되어 있으며 작업 기간동안 지역 저장 장치에 저장되어 있는 공간 데이타를 변경한다. 하나의 이동 클라이언트에서 다수의 이동 트랜잭션이 수행되지만 동시 수행되지 않고 각각의 이동 트랜잭션은 사용자에 의해 순서화된다. 하나의 이동 트랜잭션이 완료되면 트랜잭션의 변경 결과는 지역 저장 장치에 저장되며 이동 클라이언트가 대리 PC에 연결될 때 단절된 수행 기간동안 완료된 이동 트랜잭션들의 결과들이 서버에 반영된다. 이를 위해 단절 상태에서 이동 클라이언트의 수행된 이동 트랜잭션을 완료하는 것을 가상 완료(virtual commit)라 한다.

이동 클라이언트가 대리 PC에 접속할 때 단절 기간동안 수행된 이동 트랜잭션들의 결과를 서버에 반영하기 위하여 대리 PC가 이동 트랜잭션들을 대행하여 서버에 완료를 요청한다. 대리 PC에서 수행되는 트랜잭션을 대리 트랜잭션이라 하고 다음과 같이 정의한다.

> 정의 2 : 대리 트랜잭션(Surrogate Transaction, ST)은 대리 PC에서 수행하는 트랜잭션으로 다수의 이동

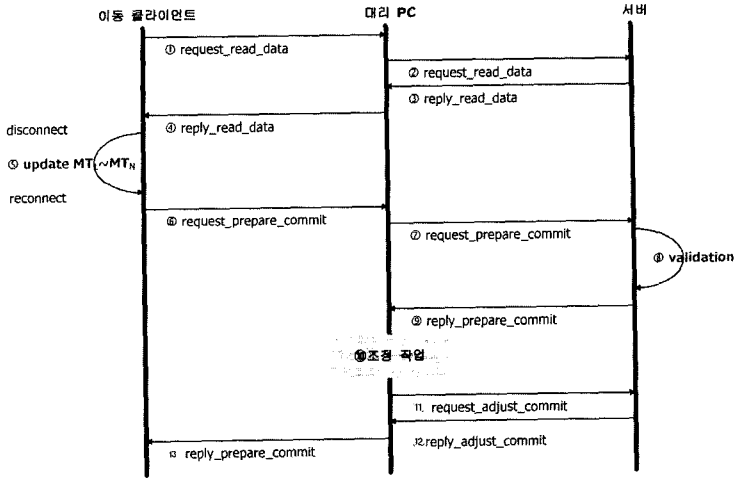


그림 3 대리 트랜잭션 프로토콜

트랜잭션들로 구성된다. 대리 트랜잭션은 다음과 같은 작업을 수행한다.

- ◇ 서버의 공간 데이타를 이동 클라이언트에 전송
- ◇ 이동 클라이언트가 변경한 공간 데이타를 서버에 전송
- ◇ 서버에서 검증한 공간 데이타 변경 완료 및 복귀를 이동 클라이언트에 전달

대리 트랜잭션은 이동 트랜잭션들의 상위 트랜잭션으로 이동 클라이언트에서 수행된 이동 트랜잭션들을 대신하여 서버에 완료를 요청한다. 대리 트랜잭션은 이동 클라이언트가 대리 PC에 공간 데이타를 요청하고 이에 의해 서버에 필요한 공간 데이타 집합을 요청할 때 시작된다. 이동 클라이언트가 접속하여 이동 트랜잭션의 결과를 완료 요청할 때 대리 트랜잭션은 이동 트랜잭션을 대신하여 서버에 연결하여 검증 작업을 수행한다. 만약 충돌이 발생하지 않으면 서버에 변경 결과를 반영한 후 완료된다. 이동 트랜잭션의 가상 완료는 대리 트랜잭션이 완료되어야 최종적으로 완료된 것이다.

그림 3은 S-S-M 구조에서 하나의 이동 클라이언트와 대리 PC에서 이동 트랜잭션과 대리 트랜잭션을 수행하기 위한 대리 트랜잭션의 프로토콜을 보여준다. 이동 클라이언트가 대리 PC에게 공간 데이타를 요청하고 요청을 받은 대리 PC가 서버에 공간 데이타를 요청하면 대리 트랜잭션이 시작되고 서버는 대리 트랜잭션의 시작을 로그에 저장한다(그림 3-①~④). 대리 PC로부터 필요한 데이타를 받은 이동 클라이언트는 단절 상태에서 이동 트랜잭션을 시작하고 공간 데이타의 변경 작업을 수행한다(그림 3-⑤). 모든 공간 데이타의 변경 작업이 완료되면 이동 클라이언트는 대리 PC에 연결을 설정하고 이동 트랜잭션들의 완료를 요청한다. 완료 요청을 받은 대리 트랜잭션은 이동 트랜잭션들을 대신하여 서버에 완료를 요청한다(그림 3-⑦). 이 때 대리 트랜잭션은 모든 이동 트랜잭션의 쓰기 집합을 서버로 전송한다.

대리 트랜잭션의 완료 작업시 다른 대리 트랜잭션과의 충돌 발생 여부를 검사하기 위한 검증 단계를 수행한다(그림 3-⑧). 검증 단계에서 대리 트랜잭션간에 충돌이 발생하는 경우 해당 대리 트랜잭션은 조정(adjust) 작업을 수행한다. 조정 작업을 위해 대리 PC는 서버로부터 충돌하는 다른 대리 트랜잭션의 결과를 전달받는다. 조정 작업은 대리 PC에서 사용자에 의해 수행되며 충돌한 대리 트랜잭션의 결과를 수정하거나 또는 취소한다(그림 3-⑩). 대리 PC가 서버에 조정 완료를 요청하면 서버는 이를 데이타베이스에 반영하고 대리 트랜잭션과 서브 트랜잭션인 이동 트랜잭션들은 완료된다.

4. 확장 검증 조건

공간 데이타를 변경하는 기존의 트랜잭션 처리 기법은 동시성 제어를 위하여 비관적 제어 기법인 잠금 기법을 사용하였으며 트랜잭션의 중간 결과를 협동 작업 사용자에게 변경 전파(update propagation)하는 협동 작업을 통하여 동시 수정된 공간 데이타의 일관성을 보장하였다[3,4,5]. 그러나 이동 트랜잭션은 단절 상태에서 공간 데이타를 변경하기 때문에 잠금 기법과 협동 작업을 적용하기 어렵다. 따라서 단절 상태에서 수행된 이동

가정 : Validation(ST_i) < Validation(ST_j)

조건 1 : Finish(ST_i) < Start(ST_j)
조건 2 : Start(ST_j) < Finish(ST_i) < Validation(ST_j) AND
Write_Set(ST_i) ∩ Write_Set(ST_j) = ø

그림 4 기존의 검증 조건

트랜잭션의 동시성 제어를 위하여 낙관적 제어 기법인 검증 기법을 확장하여 사용한다.

4.1 검증 기법

검증 기법하에서 이동 트랜잭션은 판독 단계(read phase)시 이동 클라이언트의 지역 데이타를 이용하여 변경 작업을 수행한 후에 가상 완료된다. 대리 PC에서 연결된 후 검증 단계(validation phase)에서 이동 트랜잭션의 결과는 대리 PC로 이동되고 대리 트랜잭션이 검증 조건을 이용하여 대리 트랜잭션이 대행하는 이동 트랜잭션들의 결과가 직렬성에 위배되는지 서버에서 검사한다. 만약 위배되지 않는다면 기록단계(write phase)에서 변경 결과는 서버의 안정 저장소에 저장되고 대리 트랜잭션과 이동 트랜잭션들은 최종 완료된다.

검증 단계에서 이동 트랜잭션의 검증 검사(validation test)를 위하여 Start, Commit 그리고 Finish 타임스탬프를 사용한다. Validation(ST_i)는 이동 클라이언트가 대리 PC에 연결된 후 대리 트랜잭션 ST_i가 서버에서 완료 작업을 위한 검증 검사를 시작할 때의 타임 스탬프이다. 그림 4는 트랜잭션간의 충돌이 발생하지 않는 기존의 검증 조건들을 보여준다. 두 개의 대리 트랜잭션 ST_i과 ST_j가 있을 때 조건 1 또는 조건 2를 만족하면 ST_i와 ST_j는 충돌하지 않는다.

그러나 공간 객체는 일반적인 객체와는 달리 기하 데이타에 의한 객체간의 공간 관련성이 존재한다. 따라서 서로 다른 공간 객체에 대한 동시 변경시 공간 객체간의 관련성에 대한 일관성도 유지되어져야 한다. 그러나 그림 4의 조건은 트랜잭션간의 공유 객체에 대한 일관성만을 검증하기 때문에 두 대리 트랜잭션 ST_i와 ST_j가 서로 다른 공간 객체인 o_i과 o_j에 대하여 동시 수정할 때 o_i와 o_j간의 공간 관련성의 비일관성은 검증할 수 없다.

4.2 확장된 검증 조건

동시 수정된 두 공간 객체 사이의 위상 관계인 공간 관련성의 비일관성을 검증하기 위하여 이 논문에서 기존의 검증 조건을 확장한다. 확장된 검증 조건에서 두 대리 트랜잭션 사이에 충돌이 발생할 수 있는 조건을 시간(Time), 데이타 집합(Data Set) 그리고 공간 관련성(Spatial Relation)으로 구분한다. 그림 5는 간섭 배제 조건의 첫번째 요소인 시간을 나타내는데, 두 트랜잭션 ST_i와 ST_j의 Start, Validation 그리고 Finish 타임스탬프로써 충돌 여부를 결정한다. 그림 5-(a)와 같이 ST_i가 완료한 후 ST_j가 시작하면 ST_i와 ST_j는 시간상으로 순서화되어 있으므로 충돌하지 않는다. 그러나 그림 5-(b)와 같이 ST_i와 ST_j가 시간적으로 동시 수행된 경우 두 번째 조건인 데이타 집합을 이용하여 충돌 여부를 검사해야 한다.

그림 6은 두 번째 검증 조건인 데이타 집합을 나타낸다. 두 트랜잭션 ST_i와 ST_j가 $Start(ST_j)<Finish(ST_i)<Validation(ST_j)$의 조건을 만족하면서 각 수정 결과인 데이타 집합간에 공유 객체가 존재하면 ST_j는 ST_i와 충돌한다. 그러나 동시 수정한 공유 객체가 없는 경우에는 세 번째 검증 조건인 공간 관련성을 검사한다.

조건 2. Data Set

If $Start(ST_j) < Finish(ST_i) < Validation(ST_j)$

(a) $Write\_Set(ST_i) \cap Write\_Set(ST_j) = \phi$ → 공간 관련성을 검사

(b) $Write\_Set(ST_i) \cap Write\_Set(ST_j) \neq \phi$ → 트랜잭션 T_j를 조정

그림 6 두 번째 검증 조건 - 데이타 집합

마지막으로 그림 7은 세 번째 검증 조건인 공간 관련성을 나타낸다. 두 트랜잭션 ST_i와 ST_j가 시간상으로 동시 수행되고 변경한 데이타 집합 사이에 공유 객체가

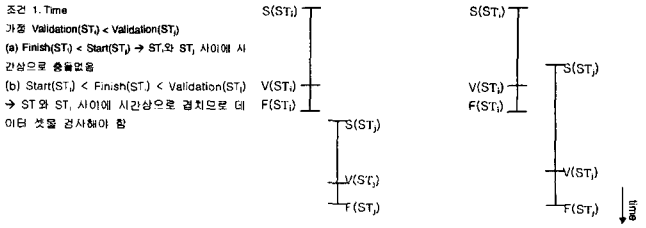


그림 5 첫 번째 검증 조건 - 시간

없더라도 두 트랜잭션이 변경한 공간 객체간에 공간 관련성이 존재할 수 있다. 만약 ST_i가 수정한 객체 o_i와 ST_j가 수정한 공간 객체 o_j간에 disjoint를 제외한 overlaps, meets, insides, contains 그리고 equals의 공간 관련성이 존재하면 동시 수정으로 인해 생성된 o_i와 o_j간의 공간 관련성이 비일관적일 수 있기 때문에 ST_j는 ST_i와 충돌한다. 그러나 o_i와 o_j간의 공간 관련성이 disjoint이면 o_i와 o_j는 동시 수정될 수 있기 때문에 ST_j는 완료된다. $SR(o_i, o_j)$는 공간 객체 o_i와 o_j간의 공간 관련성을 나타낸다.

조건 3. Spatial Relationship

If $Start(ST_i) < Finish(ST_i) < Validation(ST_i)$ AND $Write\_Set(ST_i) \cap Write\_Set(ST_j) \neq \phi$

(a) $SR(\exists o_i, \exists o_j) = \{meet, inside, cover, overlap, equal\}$ where $o_i \in$ Write_Set(ST_i) and $o_j \in$ Write_Set(ST_j) → ST_j는 ST_i와 충돌
(b) $SR(\forall o_i, \forall o_j) = \{disjoint\}$ where $o_i \in$ Write_Set(ST_i) and $o_j \in$ Write_Set(ST_j) → STj는 STi와 충돌하지 않음

그림 7 세 번째 검증 조건 - 공간 관련성

이제까지 기술한 바와 같이 시간, 데이타 집합 그리고 공간 관련성을 이용하여 기존의 검증 조건을 확장하였다. 표 1은 확장된 검증 조건을 보여준다.

5. 조정 단계를 포함한 확장된 완료 프로토콜

5.1 충돌 객체를 위한 조정 단계

기존의 검증 기법은 트랜잭션간에 충돌이 발생하면 해당 트랜잭션을 취소하고 새로운 트랜잭션을 다시 시작한다. 그러나, 이동 트랜잭션은 사용자 상호 작업에 의해 공간 데이타를 수정하기 때문에 이동 트랜잭션과 상위 트랜잭션인 대리 트랜잭션은 긴 트랜잭션이다. 따라서 기존의 기법처럼 충돌된 트랜잭션을 취소하고 새로운 트랜잭션을 다시 시작하면 공간 데이타를 변경하기 위한 비용이 너무 크다.

이 논문에서는 충돌이 발생한 경우 트랜잭션을 완료하기 위한 비용을 최소화하기 위하여 대리 트랜잭션의 완료 작업시 조정 단계를 추가하였다. 조정 단계는 대리 PC에서 수행되는 단계로 충돌이 발생한 경우 사용자 판단에 의해 해당 트랜잭션을 취소하거나 또는 충돌이 발생한 객체를 사용자가 재수정하는 단계이다. 이를 위해 대리 PC는 서버로부터 다른 대리 트랜잭션의 변경 결과를 전파받는다. 공간 데이타 수정 작업은 트랜잭션 시작시 미리 정의된 연산에 의해 수행되지 않고 사용자에 의해 수행되기 때문에 사용자 조정 작업을 통하여 충

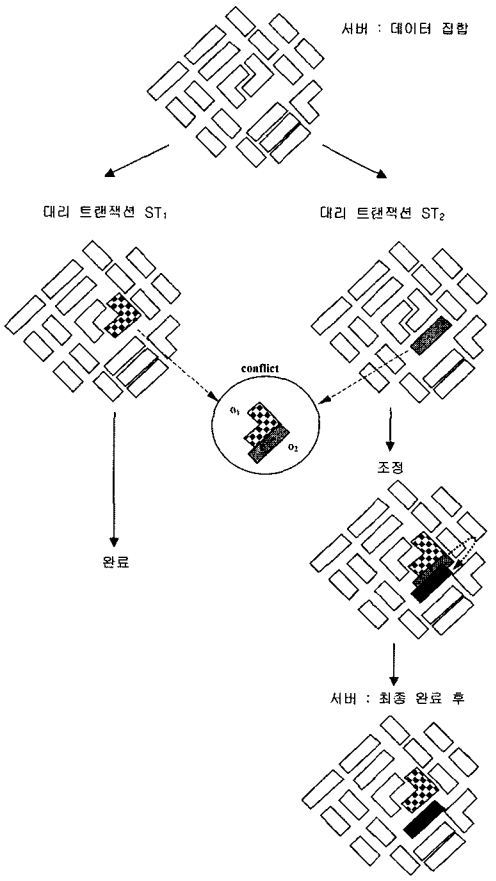


그림 8 조정 작업의 예

표 1 확장된 검증 조건

검증 조건		결과
(1) Time	$Finish(ST_i) < Start(ST_j)$	비충돌
	$Start(ST_j) < Finish(ST_i) < Validation(ST_j)$	(2) 검사
(2) Data Set	$Write\_Set(ST_i) \cap Write\_Set(ST_j) = \emptyset$	(3) 검사
	$Write\_Set(ST_i) \cap Write\_Set(ST_j) \neq \emptyset$	충돌
(3) Spatial Relationship	$SR(o_i, o_j)$ = {meet, inside, cover, overlap, equal} where $o_i \in$ Write_Set (ST_i) and $o_j \in$ Write_Set(ST_j)	충돌
	$SR(o_i, o_j)$ = {disjoint} where $o_i \in$ Write_Set(ST_i) and $o_j \in$ Write_Set(ST_j)	비충돌

돌 객체의 비일관성을 해결할 수 있다. 대리 PC에서 조정 단계가 수행되는 동안 서버는 완료 작업을 중지하고 조정 단계가 완료되어 충돌이 해소될 때까지 대기한다.

그림 8은 대리 트랜잭션 ST_2가 ST_1과 충돌할 때 조정 작업을 수행하여 충돌 객체의 충돌을 해소하는 예이다. ST_1과 ST_2는 각각 o_1과 o_2를 수정하고 ST_1이 ST_2보다 먼저 완료되었다. ST_2의 검증 작업시 o_1과 o_2가 meet관계를 유지하고 있으므로 ST_2는 ST_1과 충돌한다. 따라서 동시 수정된 o_1과 o_2간의 올바른 공간 관련성을 유지하기 위하여 ST_2의 대리 PC인 SPC_2에서는 ST_1의 변경된 결과를 이용하여 조정 작업을 수행한다. 조정 작업이 끝나고 ST_2가 조정 완료를 서버에 알리고 조정된 결과 값을 서버에 전송하면 서버는 전송된 조정 결과를 반영하고 ST_2가 완료되었음을 SPC_2에게 알린다. SPC_2는 ST2가 완료되면 ST2의 서브 트랜잭션인 이동 트랜잭션의 결과가 조정되었음을 MC_2에게 전달하고 ST_2는 최종적으로 완료된다.

5.2 확장 완료 프로토콜

기존의 클라이언트 서버 구조에서 클라이언트의 트랜잭션이 완료를 요청하면 서버에서 검증 단계를 수행한 후에 충돌이 발생하지 않으면 쓰기 단계에서 트랜잭션의 결과를 서버에 저장하였다. 그러나 충돌이 발생하면 트랜잭션의 결과와 충돌된 트랜잭션을 취소한다. 그러나 충돌한 경우에 트랜잭션을 취소하지 않고 조정 작업을 수행하기 위하여 기존의 완료 프로토콜을 확장하여 조정 단계를 추가한다.

이동 트랜잭션 MT_i~MT_k가 이동 클라이언트 MC_i에서 수행된 후에 SPC_i에 접속하여 완료를 요청하고 이에 따라 대리 트랜잭션 ST_i가 서버 S에 완료를 요청하여 완료 작업을 수행할 때 다음의 순서로 완료 프로토콜을 수행한다.

- Phase 1 : 이동 클라이언트 MC_i가 대리 PC SPC_i에 이동 트랜잭션 MT_i~MT_k의 완료를 요청하면 SPC_i는 <prepare ST_i>를 로그에 기록하고 ST_i의 완료를 서버에 요청한다.
- Phase 2 : 서버 S는 ST_i와 충돌할 수 있는 변경 정보를 확인하고 변경 정보가 존재하지 않으면 서버에 ST_i의 결과를 반영하고 <commit ST_i>를 로그에 기록한다. 만약 변경 정보가 존재하면 ST_i가 다른 클라이언트의 트랜잭션과 충돌하는지 검사하기 위해 검증 작업을 수행한다. 만약 ST_i가 ST_j와 충돌하면 서버는 <adjust ST_i>를 로그에 기록하고 조정 메세지를 SPC_i에 전달한다. 그러나 ST_i가 충돌하지 않으면 <commit ST_i>를 로그에 기록하고 ST_i의 결과를 서버의 저장 장치에 반영한다. 그리고 ST_i가 완료되었음을 SPC_i에 전송한다.
- Phase 3 : SPC_i는 서버에서 완료 메세지가 도착하면 <commit ST_i>를 로그에 저장하고 ST_i의 결과를 SPC_i의 저장 장치에 반영한다. 그리고 ST_i의 서브 트랜잭션이 MT_i~MT_k의 결과가 서버에 반영되었음을 MC_i에게 전달한다. 그러나 만약 조정 메세지가 SPC_i에 전달되면 <adjust ST_i>를 로그에 저장하고 SPC_i의 사용자는 ST_i의 결과들을 조정한다. 사용자의 조정 작업이 끝나면 SPC_i는 서버에 조정 완료 요청 메시지를 전달한다.
- Phase 4. 서버는 SPC_i로부터 조정 완료 요청 메세지가 도착하면 <commit T_i>를 로그에 저장한 후에 ST_i의 결과를 저장한다. 그리고 완료 메시지를 SPC_i에 전송한다. 만약 취소 요청이 도착하면 로그에 <abort ST_i>를 저장한 후에 ST_i를 취소한다. SPC_i는 서버로부터 완료 메시지가 도착하면 <commit ST_i>를 로그에 저장하고 ST_i의 조정된 결과를 SPC_i에 저장한다. 그리고 조정된 결과와 MT_i~MT_k가 완료되었음을 MC_i에 전달한다.

그림 9는 확장된 완료 프로토콜의 흐름도를 보여준다. 완료 규약을 실행하는 도중에 발생한 장애(failure)는 다음과 같이 처리한다.

첫째, 이동 클라이언트와 대리 PC 사이의 완료 요청 및 완료 응답은 사용자의 조작으로 이루어지므로 이동 클라이언트와 대리 PC 사이에 장애는 사용자가 직접 회복을 수행한다.

둘째, 대리 PC에서 장애가 발생하면 사용자는 대리 PC의 로그를 기반으로 조치를 취한다. 로그에 <commit ST_i>가 존재하면 ST_i는 완료되었으므로 대리 PC는 트랜잭션 ST_i의 결과를 대리PC에 저장하고 ST_i를 완료한다. 로그에 <adjust ST_i>가 존재하고 <commit ST_i>가 존재하지 않으면 서버로부터 전송되었던 다른 트랜잭션의 결과를 이용하여 ST_i에 대한 조정 작업을 수행한다. 로그에 <prepare ST_i>가 존재하고 <commit ST_i>나 <adjust ST_i>가 존재하지 않으면 서버에게 ST_i의 완료 작업의 상태를 요구한다. 만약 ST_i가 성공적으로 완료되었으면 ST_i의 결과를 대리 PC에 저장하고 ST_i가 완료되었음을 이동 클라이언트에게 전달한다. 그리고 ST_i를 완료한다. 만약 ST_i가 충돌되었으면 서버로부터 다른 트랜잭션의 결과를 전송받은 후에 대리 PC에서 ST_i에 대한 조정 작업을 수행한다.

셋째, 서버에서 장애가 발생하면 서버가 회복될 때까지 대기한다. 서버가 복구되면 서버는 로그 레코드를 검

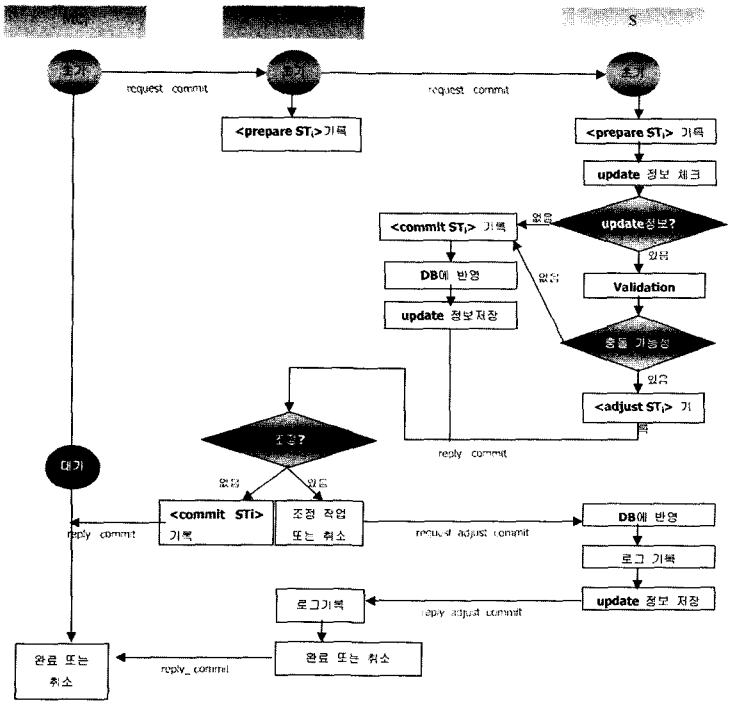


그림 9 변경 완료 규약의 흐름도

색하여 회복 작업을 수행한다. 로그에 <commit ST_i>가 존재하면 ST_i의 결과를 서버에 저장하고 ST_i를 완료한다. 만약 로그에 <adjust ST_i>가 존재하고 <commit ST_i>가 존재하지 않으면 ST_i는 현재 대리 PC에서 조정작업을 수행하고 있기 때문에 조장 작업이 완료될 때까지 STi의 완료 작업을 대기한다. 로그에 <prepare ST_i>가 존재하고 <commit ST_i>나 <adjust ST_i>가 존재하지 않으면 ST_i의 검증 작업을 수행하고 이 후의 완료 작업을 계속한다.

6. 구현

6.1 시스템 구조

그림 10은 본 논문의 구현 환경 및 시스템 구조이다.

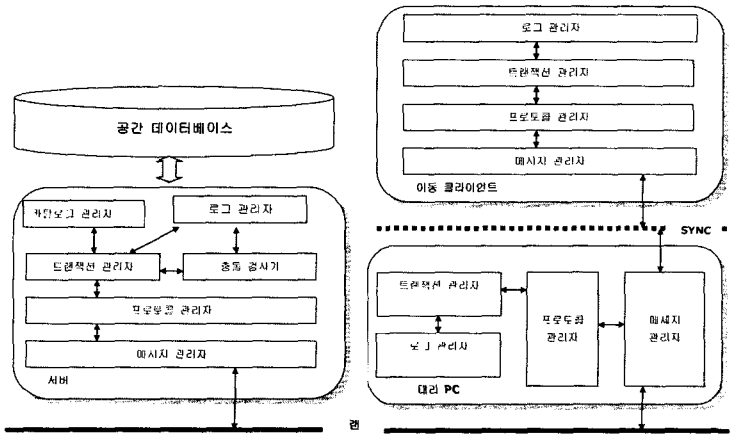


그림 10 S-S-M 구조를 갖는 전체 시스템의 구성 요소

서버는 리눅스 환경의 공간 데이타베이스를 대상으로 구현하였으며, 대리 PC와 이동 클라이언트는 각각 Windows 2000과 Windows CE 환경에서 Microsoft Visual C++ 6.0과 Embedded Visual C++ 3.0으로 구현하였다. 서버와 대리 PC는 소켓 통신으로 데이타를 전송하며, 대리 PC와 이동 클라이언트는 USB를 통해 싱크하여 데이타를 교환한다.

서버 시스템의 모듈은 다음과 같다.

- 트랜잭션 관리자 : 대리 PC가 요청한 대리 트랜잭션을 처리하고 정보를 관리하는 모듈
- 카탈로그 관리자 : 대리 트랜잭션의 상태 정보와 변경된 정보를 저장하고 관리하는 모듈
- 로그 관리자 : 대리 트랜잭션의 로그를 저장하고 관리하는 모듈
- 충돌 검사기 : 완료 요청된 대리 트랜잭션의 충돌 여부를 검사하는 모듈
- 프로토콜 관리자 : 서버와 대리 PC간의 요청 메시지와 응답 메시지에 대한 프로토콜을 관리하는 모듈

대리 PC와 이동 클라이언트를 구성하는 각 모듈과 그 역할은 다음과 같다.

- 트랜잭션 관리자 : 이동 클라이언트의 이동 트랜잭션들을 관리하고 이들을 대행하는 대리 트랜잭션을 처리하는 모듈
- 로그 관리자 : 대리 트랜잭션과 이동 트랜잭션들의 로그를 저장하고 관리하는 모듈
- 프로토콜 관리자 : 서버와 대리 PC, 대리 PC와 서버 사이의 요청 및 응답 메시지에 대한 프로토콜을 관리하는 모듈

6.2 구현 예

이 절에서는 대리 PC를 통해 전달 받은 공간 데이타를 이동 클라이언트의 이동 트랜잭션이 변경한 후에 이동 트랜잭션들의 완료 요청시 대리 트랜잭션을 통하여

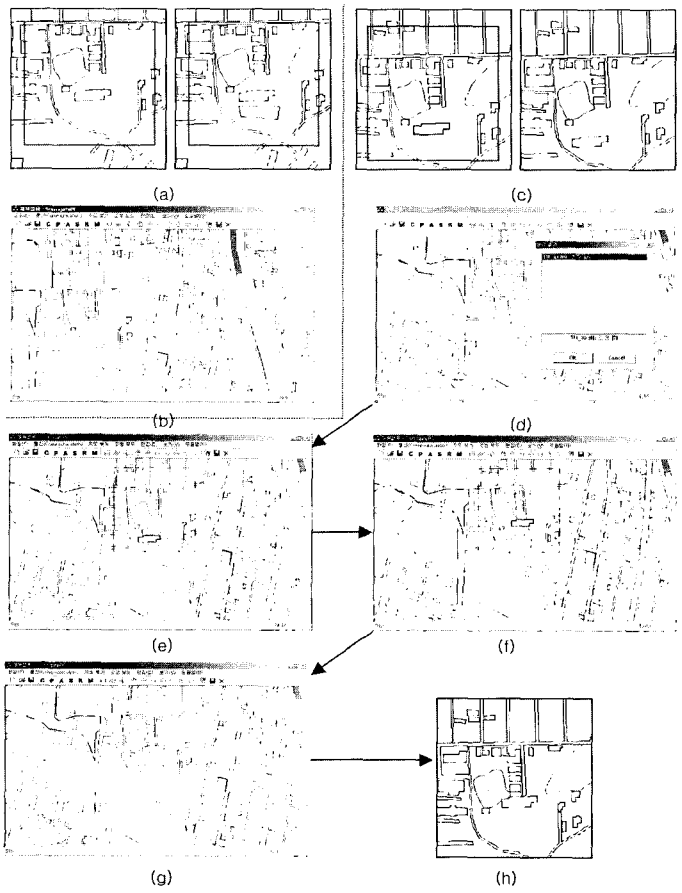


그림 11 충돌된 ST_2에 대하여 조정 작업을 수행하는 예

완료 작업이 수행되는 과정을 보인다. 그림 11은 두 대리 트랜잭션 사이에 충돌이 발생하여 대리 PC에서 조정 작업을 수행하는 예를 보여준다. 그림 11-(a)와 그림 11-(b)는 이동 클라이언트 MC1의 변경 트랜잭션인 MT_1의 작업 상황이며 그림 11-(c)~(h)는 MT_2의 변경 트랜잭션이다. 그림 11-(a)에서 MT_1은 공간 객체를 삽입하는 작업을 수행하고 그림 11-(c)에서 MT_2는 공간 객체를 이동하는 작업을 수행하였다. 먼저 완료를 요청한 MT_1과 MT_1의 대리 트랜잭션 ST_1이 완료된 후에 (그림 11-(b)) MT_2가 완료를 요청한다. 따라서 ST_2가 완료 작업을 수행하기 위하여 서버에서 검증 작업을 수행한다.

MT_2가 수정한 객체는 MT_1이 삽입한 객체와 overlaps 관련성을 가지고 있기 때문에 ST_2는 ST_1과 공간 관련성으로 인한 변경 충돌이 발생한다(그림 11-(d)). 서버는 ST_2의 대리 PC에게 충돌이 발생하였음을 알리고 ST_2의 사용자는 조정 작업을 수행한다. ST_2의 사용자는 그림 11-(f)와 그림 11-(g)와 같이 ST_1의 결과를 이용하여 충돌이 발생한 객체에 대한 조정 작업을 수행한다. 조정 작업이 완료되면 ST_2는 서버에 조정 완료를 요청한다. 서버는 ST_2의 조정된 결과를 저장한 후에 ST_2가 완료되었음을 대리 PC에게 알린다. 대리 PC는 ST_2의 결과를 저장한 후에 조정된 결과를 이동 클라이언트에 전달하며 MT_2와 ST_2는 최종적으로 완료된다.

7. 결 론

이 논문에서는 이동 컴퓨팅 환경에서 3-계층 구조의 S-S-M(서버-대리 PC-이동 클라이언트) 구조에서 공간 데이타 변경 트랜잭션을 처리하기 위한 대리 트랜잭션 모델을 제안하고 그에 따른 관리자를 구현하였다. S-S-M 구조는 단일 GIS 서버와 여러 대리 PC 및 이동 클라이언트로 이루어지며 이동 클라이언트의 요청은 대리 PC를 통해 서버로 전달되고 서버의 응답 역시 대리 PC를 통해 이동 클라이언트로 전달된다. 또, 이동 클라이언트는 대리 PC와 접속을 해제한 후 지도를 수정한다.

S-S-M 구조에서 대리 트랜잭션의 동시성을 제어하기 위해 잠금 기법을 사용하면 긴 대기 문제가 발생하므로 낙관적 기법의 하나인 검증 기법을 사용하였다. 그러나 기존의 검증 조건은 트랜잭션 간의 공유 객체에 대해서만 정의되었기 때문에 공간 객체간의 공간 관련성에 의한 비일관성을 검증하지 못한다. 따라서 이 논문에서는 기존의 검증 조건을 확장하여 시간, 데이타 집합 그리고 공간 관련성의 세가지 요건을 사용하는 검증 조건을 제시하였다. 또한 충돌이 발생한 트랜잭션의 완료 비용을 최소화하기 위하여 충돌이 발생한 트랜잭션을 취소하지 않고 대리 PC에서 조정 작업을 수행하여 완료되도록 하였다. 이를 위해 완료 작업시 조정 단계가 추가된 확장된 완료 프로토콜을 제시하였다.

참 고 문 헌

[1] Keith K. S. Lee, Y. H. Chin, "A New Replication Strategy for Unforeseeable Disconnection under Agent Based Mobile Computing System," International Conference on Parallel and Distributed Systems, 1998.

[2] Jim Gray, Pat Helland, Patrick O'Neil, Dennis Shasha, "The Danger of Replication and a Solution," ACM SIGMOD International conference on Management of data, 1996.

[3] 신영상, 최진오, 조대수, 홍봉희, "클라이언트 변경 트랜잭션에서 동시성 및 일관성 제어", 한국정보과학회 추계 학술발표 논문집, 1999.

[4] 최진오, 홍봉희, "분산된 지리정보시스템에서 새로운 잠금 기법을 이용한 중복된 공간 데이타의 변경 전파", 한국정보과학회 논문지, Vol 26, No 9, 1999.

[5] 박재관, 김동현, 최진오, 홍봉희, "클라이언트 서버 환경에서 공간 데이타 변경 트랜잭션을 위한 회복 기법의 설계 및 구현", 한국정보과학회 추계 학술발표논문집, 2000.

[6] Chrysanthis, P.K., "Transaction Processing in a Mobile Computing Environment," IEEE workshop on Advances in Parallel and Distributed Systems, 1993.

[7] Pu C., Kaiser G., and Hutchinson, "Split transactions for Open-ended Activities," Very Large Data Bases, 1988.

[8] Sanjay Kumar Madria, Bharat Bhargava, "A Transaction Model for Mobile Computing," Database Engineering and Applications Symposium, International Database Engineering and Application Symposium, 1998.

[9] Shirish Hemant Phatak, B.R. Badrinath, "Multiversion Reconciliation for Mobile Databases," International Conference on Data Engineering, 1999.

[10] H. T. Kung, John T. Robinson, "On Optimistic Methods for Concurrency Control," ACM Transaction on Database Systems. Vol 6, Issue 2, 1981.

[11] Bharat Bhargava, "Concurrency Control in Database Systems," IEEE Transactions on Knowledge and Data Engineering , Vol 11, Issue 1 , 1999.

[12] Eliezer Levy, Henry F. Korth and Abraham Silberschatz, "An Optimistic Commit Protocol for

Distributed Transaction Management", ACM SIGMOD international conference on Management of data, 1991.
[13] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts," Mcgraw Hill.
[14] Michael F. Worboys, "GIS, A Computing Perspective," Taylor & Francis.
[15] 문선희, 반재훈, 홍봉희, "이동 클라이언트의 공간 데이타 변경을 위한 대리 트랜잭션 모델", 한국정보과학회 춘계 학술발표논문집, 2001.

반 재 훈
1997년 2월 부산대학교 컴퓨터공학과 공학사. 1999년 2월 부산대학교 컴퓨터공학과 공학석사. 2001년 2월 부산대학교 컴퓨터공학과 박사수료. 2002년 3월~현재 경남정보대학 인터넷응용계열 전임강사. 관심분야는 이동객체, 객체지향데이타베이스

문 선 희
2000년 2월 부산대학교 컴퓨터공학과 공학사. 2002년 2월 부산대학교 컴퓨터공학과 공학석사. 2002년 3월~현재 삼성전자. 관심분야는 트랜잭션, 무선통신

김 동 현
정보과학회논문지 : 데이타베이스
제 30 권 제 2 호 참조

홍 봉 희
정보과학회논문지 : 데이타베이스
제 30 권 제 1 호 참조