클라이언트-서버 환경에서 공간 데이터의 변경 트랜잭션을 위한 회복 기법

(Recovery Schemes for Spatial Data Update Transactions in Client-Server Computing Environments)

박 재 관 † 최 진 오 †† 홍 봉 희 †††

(Jae-Kwan Park) (Jin-Oh Choi) (Bong-Hee Hong)

요 약 클라이언트-서버 환경에서 공간 데이터를 변경하는 트랜잭션은 사용자와 대화식으로 진행되는 트랜잭션으로 수정 시간이 길며, 미완료된 데이터 읽기(dirty read)를 허용하기 때문에 연쇄 철회(cascading rollback)가 발생할 수 있고, 공간 객체들은 서로 공간 관련성을 가지는 특징이 있다. 기존의 회복 기법은 이러한 공간 데이터의 변경 트랜잭션 철회에서 긴 트랜잭션을 단순히 철회(rollback)함으로써 고비용 문제가 발생하며, 철회된 트랜잭션에 의해 불필요한 다른 트랜잭션들이 순차적으로 철회되는 문제가 발생한다. 또한 공간 데이터가 가지는 새로운 일관성 제약 조건인 공간 관련성이 고려되지 않기 때문에 공간 데이터의 무결성(integrity) 보장에 문제가 있다.

이 논문은 이러한 문제점들을 보완하기 위하여 공간 데이터의 변경 트랜잭션을 위한 새로운 회복 기법을 제시한다. 먼저 회복에서의 공간 관련성을 위해 회복 종속성을 정의하고 이것을 연쇄 철회의 조건으로 처리함으로써 공간 데이터의 무결성을 보장한다. 둘째, 부분 철회(partial-rollback) 기법을 제시하여 긴 트랜잭션의 고비용 철회 문제를 해소한다. 셋째, 회복의 상태를 유형별로 분류하고 각 상태에 따라 undo-delta와 partial-redo 그리고 partial-undo의 연산을 실행하는 회복 제어 기법을 제시하여 불필요한 연쇄 철회를 줄인다. 마지막으로, 이 논문에서 제안한 기법들을 구현 실험하여 정확성을 보인다.

키워드 : 트랜잭션 회복, 공간 관련성, 부분 철회

Abstract In client-server computing environments, update transactions of spatial data have the following characteristics. First, a transaction to update maps needs interactive work, and therefore it may take a long time to finish. Second, a long transaction should be allowed to read the dirty data to enhance parallelism of executing concurrent transactions. When the transaction is rolled back, it should guarantee the cascading rollback of all of the dependent transactions. Finally, two spatial objects may have a weak dependency constraint, called the spatial relationship, based on geometric topology.

The existing recovery approaches cannot be directly applied to this environment, due to the high rollback cost and the overhead of cascading rollbacks. Furthermore, the previous approaches cannot guarantee the data integrity because the spatial relationship, which is a new consistency constraint of spatial data, is not considered.

This paper presents new recovery schemes for update transactions of spatial data. To guarantee the data integrity, this paper defines recovery dependency as a condition of cascading rollbacks. The partial-rollback is also suggested to solve the problem of high rollback cost. The recovery schemes proposed in this paper can remove the unnecessary cascading rollbacks by using undo-delta, partial-redo and partial-undo. Finally, the schemes are performed to ensure the correctness

Keywords : Transaction Recovery, Spatial Relationship, Partial Rollback

† 비 회 원 : 부산대학교 컴퓨터공학과
jkpack@pusan.ac.kr
†† 정 회 원 : 부산외국어대학교 컴퓨터공학과 교수
jochoi@taejo.pufs.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@pusan.ac.kr
논문접수 : 2001년 10월 18일
심사완료 : 2002년 10월 14일

1. 서 론

일반적으로 트랜잭션은 시스템 오류, 교착상태(deadlock), 그리고 사용자 오류 등의 이유로 실패할 수 있다. 이 때 원자성(atomicity) 원칙을 보장하기 위해 시스템은 트랜잭션 시작 전의 상태로 되돌릴 수 있어야 한다. 이를 위해 회복 시스템(recovery system)은 다음의 두 가지 처리를 해야 한다. 첫째, 실패한 트랜잭션에 의해 변경된 데이터를 시작 전 상태로 되돌려야 한다. 둘째, 실패한 트랜잭션의 변경에 영향을 받은 트랜잭션들을 연쇄 철회해야 한다. 이것은 회복 시스템의 정확성(correctness) 조건이다[1].

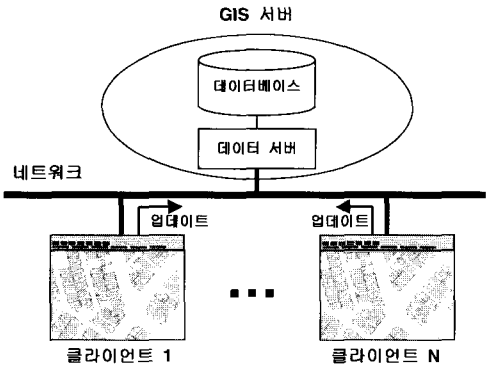

그림 1 클라이언트-서버 GIS 환경에서의 동시 변경

그림 1은 클라이언트-서버 GIS환경에서 동시 변경 작업의 예이다. 서버와 클라이언트는 네트워을 통해 연결되어 있고 클라이언트들은 자신의 수정 작업을 위해 각각 트랜잭션을 실행한다. 이러한 환경에서 지도 수정 트랜잭션은 다음과 같은 특징을 가진다.

대화식(interactive) 트랜잭션이므로 사용자가 트랜잭션 연산을 결정한다[2].

긴 트랜잭션이므로 동시성 지원을 위해 미완료된 데이터 읽기를 허용한다. 그러나, 이로 인해 연쇄 철회가 발생할 수 있다.

긴 트랜잭션(long-duration transaction)이므로 고비용의 전체 철회가 발생할 수 있다[3].

공간 객체는 위상 관계에 의한 공간 관련성을 지니고 있다. 회복 시에 이를 고려하지 않으면 데이터의 일관성이 깨어질 수 있다.

이러한 특징들 때문에 기존의 기법으로는 지도 수정 트랜잭션의 회복 문제를 처리하기 어렵다. 전통적인 트랜잭션 회복 방법은 회복 가능성(recoverability)을 보장하기 위해 쓰기-잠금이 설정된 데이터에 접근한 트랜잭션의 완료를 쓰기-잠금을 설정한 트랜잭션이 완료될 때까지 지연시킨다. 이 기법은 트랜잭션의 완료를 위해 장시간 대기해야 하므로 동시성이 크게 저하되는 문제가 있다. 따라서, 트랜잭션의 동시성을 지원하기 위한 변형된 기법들이 제시되었다. 먼저 보상 트랜잭션(compensating transaction)에 의한 회복 기법[4]은 미완료된 데이터를 읽은 트랜잭션을 지연없이 완료한다. 그리고 오류 발생시에는 완료된 트랜잭션을 무효화하는 트랜잭션을 실행함으로써 무결성을 유지한다. 그러나 이 기법은 긴 트랜잭션의 철회에서 전체 트랜잭션을 취소하는 고비용 때문에 받아들이기 어려운 문제가 있다. 또 다른 변형인 중첩 트랜잭션(nested transaction)에서의 회복 기법[5]은 긴 트랜잭션을 서브-트랜잭션으로 나누어 처리한다. 그리고 서브-트랜잭션의 취소는 보상 트랜잭션을 사용함으로써 동시성을 보장한다. 그러나 미완료된 데이터를 읽은 트랜잭션을 연쇄 철회의 대상으로 처리하기 때문에 공간 객체의 수정 트랜잭션에 적용하면 다수의 불필요한 연쇄 철회가 발생하는 문제가 있다.

이와 같은 기존의 회복 기법이 공간 데이터 변경 트랜잭션의 회복 기법으로 사용될 수 없는 이유는 공간 데이터가 다음의 요구 조건을 추가로 필요로 하기 때문이다. 첫째, 트랜잭션 철회 시에 공간 관련성을 고려하여야 한다. 즉 회복 기법은 수정 트랜잭션에 의해 변경된 공간 객체간의 공간 관련성에 따라 처리 방법이 달라져야 한다. 이는 공간 데이터가 객체간의 위상 관계(topology relationship)에 의한 종속성을 가지기 때문이다. 둘째, 전체 철회(total rollback)가 허용되어서는 안 된다. 긴 트랜잭션의 취소는 정신적, 경제적 손실이 크다[3]. 따라서 긴 트랜잭션을 서브-트랜잭션으로 분리하여 회복을 제어할 필요성이 있다. 셋째, 연쇄 철회의 발생을 줄이는 기법이 필요하다. 동시성의 지원을 위해서는 한 트랜잭션에 의해 잠금이 걸린 객체들에 대하여 다른 트랜잭션들의 새로운 접근을 허용하여야 한다. 그러나 이러한 경우 다수의 연쇄 철회가 발생할 수 있는데, 공간 객체를 수정하는 트랜잭션은 철회 비용이 기존 트랜잭션에 비하여 크기 때문에 미완료된 데이터를 읽은 모든 트랜잭션들을 철회시키는 기존의 획일적 기법은 적용될 수 없다. 철회 시점에서 트랜잭션들의 현재 문맥(context)에 따라 선별적으로 연쇄 철회를 처리하는 기법이 필요하다.

이와 같은 요구 사항을 만족시키기 위하여, 이 논문에서는 먼저 철회 발생시에 공간 관련성을 유지하기 위해 회복 종속성(recovery dependency) 개념을 제시한다. 이 개념을 트랜잭션 회복의 유형별 분류에서 연쇄 철회될 트랜잭션의 선택 조건으로 사용하여 회복에서의 공간 관련

성 문제를 해결한다.

그리고 부분 철회(partial rollback) 기법으로 전체 철회(total rollback)의 고비용 문제를 해소한다. 이를 위해 클라이언트는 긴 트랜잭션을 여러 개의 서브-트랜잭션(sub-transaction)으로 나누어 실행한다. 공간 객체를 수정하는 서브-트랜잭션들의 직렬성(serializability) 문제는 공간 관련성에 의한 종속성의 보장으로 해결될 수 있다[2][6]. 따라서 회복 종속성의 보장으로 철회의 단위를 하나의 서브-트랜잭션 단위로 제한할 수 있다.

마지막으로 철회 발생시 미완료된 데이터를 읽은 서브-트랜잭션들의 상태를 유형별로 분류하여 철회 발생시에 무결성을 위한 최소한의 연쇄 철회만 발생시키는 새로운 회복 제어 기법과 프로토콜을 제시한다.

본론으로 들어가기 전에 이 논문에서 사용하는 용어들에 대한 의미를 살펴보면 다음과 같다.

공간 관련성(spatial relationship) - 두 공간 객체간의 기하에 의한 관련성으로써 Disjoint, Meets, Equals, Inside1, Inside2, Covers1, Covers2, Overlaps 등이 있다.

서브-트랜잭션(sub-transaction) - 긴 트랜잭션을 독립적인 실행 단위로 나눈 것으로 완료 후에도 철회 가능한 트랜잭션이다.

협동 작업(cooperative work) - 지도 수정 과정에서 공간 관련성은 동적으로 변화하게 되는데, 이 때 발생하는 문제를 해결하기 위해서 도입되는 사용자 간의 의사결정 작업이다. 협동 작업에 참여한 긴 트랜잭션들은 함께 종료된다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구를 기술하고 3장에서는 공간 데이터의 종속성을 설명한다. 4장에서는 회복을 위한 연산에 대해 설명하며 5장에서는 트랜잭션의 회복 기법을 기술한다. 6장에서는 회복 프로토콜을 설명하고 7장에서는 구현 내용을 기술한다. 마지막으로 8장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

이 논문과 관련된 연구에는 첫째, 회복 가능성(recoverability)을 보장하는 전통적인 회복 기법이 있다. 트랜잭션의 특징 중 지속성(durability)은 완료 후에 그 결과가 유지됨을 의미한다. 따라서 트랜잭션은 완료 후에 취소가 불가능하므로 미완료된 데이터를 읽은 트랜잭션의 완료를 지연시켜 회복 가능성을 유지하는 기법이다. 둘째, 보상 트랜잭션을 이용하여 기존의 회복 가능성에 의한 트랜잭션의 완료 지연 문제를 해소하는 기법이 있다. 이 기법은 트랜잭션의 완료 후에 오류가 발생하면 보상 트랜잭션을 실행하여 이전 상태로 되돌린다. 셋째, 중첩 트랜잭션 모델에서 사용하는 회복 기법이 있다. 이 기법은 서브-트랜잭션의 회복 문제를 설명한다. 넷째, 협동 작업에서의 트랜잭션 회복에 관한 연구가 있다. 이것은 회복 처리를 협동 작업 중인 트랜잭션으로 제한함으로써 서버의 과부하를 해소하며 오류를 방지한다.

기존의 데이터베이스 시스템에서 사용되는 일반적인 회복 기법은 무결성을 보장하기 위해 회복 가능성을 유지한다[7]. 회복 가능성은 미-완료된(uncommitted) 데이터에 접근한 트랜잭션의 완료를 그 데이터에 쓰기-잠금을 설정한 트랜잭션이 완료될 때까지 연기한다. 이 기법을 지도 수정 트랜잭션의 회복 기법에 적용하면 회복 가능성을 보장하기 위해 트랜잭션의 실행을 매우 엄격하게 제한하여 장시간 대기 문제를 발생시키므로 대상 환경에 적용하기 어렵다.

그림 2에서 이러한 문제를 설명한다. 클라이언트 1과 클라이언트 2가 각각 공간 객체 SO_A와 SO_B를 수정하려고 한다. T_A가 SO_A에 쓰기-잠금을 설정한 후, T_B가 SO_B에는 쓰기-잠금을, SO_A에는 읽기-잠금을 설정한다. 이때 T_B가 먼저 수정 작업을 완료하더라도 T_A의 미-완료된 데이터(SO_A)에 읽기-잠금을 설정하였으므로, T_A가 완료될 때 까지 T_B는 대기해야 한다.

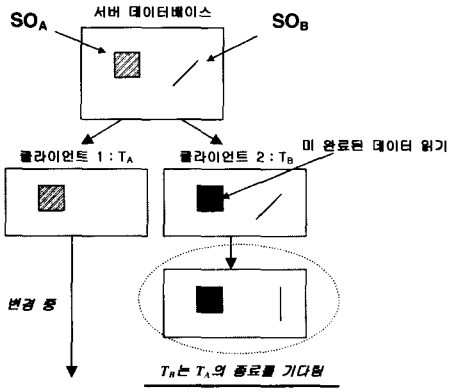

그림 2 전통적인 트랜잭션 회복 기법의 문제점

전통적인 회복 기법을 탈피하여 무결성과 동시성을 위해 보상 트랜잭션(compensating transaction)을 사용하는 기법[4]이 제시되었다. 기존의 회복 기법에서 문제가 되었던 긴 시간 대기 문제를 해결하기 위해 미완료된 데이터를 읽은 트랜잭션을 지연시키지 않고 완료한다. 만약 쓰기-잠금을 설정한 트랜잭션이 철회되면 영향을 받아 완료한 타 트랜잭션들은 보상 트랜잭션에 의해 취소된다.

그러나 이 기법을 지도 수정 환경에 적용하면 실행 시간이 긴 트랜잭션이 철회되는 문제가 발생한다. 다시 말해서, 일부 객체의 오류로 인하여 오랜 시간 동안(수 초 ~ 수 분) 실행된 트랜잭션이 철회되어 재작업을 요구하게 된다.

중첩 트랜잭션 모델(nested transaction model)에서의 회복 기법[5]은 고비용의 철회 문제를 해결하기 위해 긴 트랜잭션을 서브-트랜잭션으로 나누어 처리한다. 그리고 서브-트랜잭션의 취소는 보상 트랜잭션을 이용하여 해결한다. 서브-트랜잭션의 결과는 해당 부모 트랜잭션의 연산 범위에서만 사용되어질 수 있고 부모 트랜잭션이 완료된 후에는 다른 부모 트랜잭션에서 사용될 수 있다. 부모 트랜잭션은 완료되기 전에는 철회될 수 있으며 다른 부모 트랜잭션에 영향을 주지 않으므로 철회 작업이 한 클라이언트에서만 발생한다

이 기법을 대상 환경에 적용하면 다음과 같은 문제점이 생긴다. 첫째, 지도 수정 작업은 수정할 객체의 주변 객체들을 함께 출력하여 잠금을 걸기 때문에 잠금의 단위가 크다. 이 때 수정 작업한 트랜잭션이 철회되면 다수의 트랜잭션들이 불필요하게 연쇄 철회되므로 시스템의 처리량(throughput)이 감소하는 결과를 초래한다. 둘째, 공간 관련성이 고려되지 않기 때문에 공간 데이터의 무결성을 보장할 수 없다. 예를 들어, 반드시 접해 있어야 하는 건물과 도로 등과 같은 공간 객체들은 트랜잭션의 실행 전후에 이를 유지해야 한다. 그러나 이 기법은 서로 다른 공간 객체간의 관련성을 고려하지 못한다. 따라서 여러 클라이언트가 완료 후에 독립적으로 철회된다면 실버 폴리곤[8]이 발생하거나 에지 매칭이 되지 않는 지도의 무결성 문제가 발생할 수 있다.

GIS에서의 협동 작업을 위한 트랜잭션 회복 기법[2]이 있다. 이 기법은 연쇄 철회의 대상을 협동 작업 중인 트랜잭션으로 제한한다. 즉, 협동 작업 영역에서 철회가 발생하면 모든 협동 트랜잭션을 연쇄 철회되도록 처리하여 오류를 방지하는 비관적인 기법이다. 그러나 모든 협동 트랜잭션들이 철회될 필요는 없다. 왜냐하면 (수정 영역의 교차가 수정 객체의 관련성을 의미하지는 않기 때문이다. 그림 3에서 이러한 문제를 자세히 설명한다.

서브-트랜잭션 ST_A가 객체 SO_A를 수정하고, ST_B-ST_N이 SO_B-SO_N을 수정한다고 가정하자(그림 3 - (가)). 이 때 SO_A를 수정했던 ST_A가 철회되면 ST_A와 협동 작업 중이던 ST_B-ST_N 모두 연쇄 철회된다(그림 3 - (나)). 그리고 마찬가지로 ST_B, …, ST_N 등과 각각 협동 작업 중이던 다른 트랜잭션도 모두 연쇄 철회된다.

그런데 그림 3의 두 객체 SO_A와 SO_N는 공간 관련성이 존재하지 않는 경우이므로 ST_A에 의해 ST_N이 연쇄 철회될 필요가 없다. 즉, 이 기법을 지도 수정 환경에 그대로 적용하면 불필요한 연쇄 철회가 발생하고 영역 교차로 인해 이것이 점차 전파되는 문제가 있다.

그러므로 이 논문은 지도 수정 트랜잭션의 특징을 고려한 공간 데이터의 변경 트랜잭션을 위한 회복 기법을 제시한다. 즉, 공간 데이터 변경 트랜잭션의 회복 유형을 상세히 분류하고 각 분류에 대해 적용될 회복 연산을 제시한다.

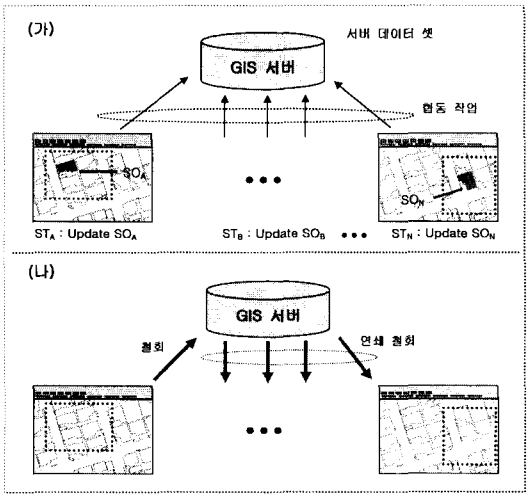

그림 3 협동 작업을 위한 회복 기법의 문제점

3. 공간 데이터의 종속성

이 장에서는 공간 데이터의 특징인 공간 관련성에 의한 변경 종속성과 회복 종속성에 대하여 기술한다.

3.1 공간 관련성에 의한 변경 종속성과 회복 종속성

공간 관련성이란 두 공간 객체간의 위상(topology) 관계에 의한 관련성을 분류한 것이다. Egenhofer는 이 공간 관련성을 Disjoint, Equals, Meets, Inside1, Inside2, Covers1, Covers2, Overlaps의 8가지로 분류하였다[9]. 기하를 가지는 공간 객체는 인접한 객체와 공간 관련성을 지니므로 잠금의 대상이 아니더라도 병렬 수정될 수 없다. 이러한 공간 관련성에 의한 병렬 수정 제약 조건을 공간 관련성에 의한 변경 종속성이라 한다[2].

공간 관련성에 의해 지도 수정의 정확성은 기존의 의미와는 다르게 적용된다. 일반적으로 트랜잭션의 실행 후에 데이터베이스의 정확성은 알려진 모든 제약 조건을 위반하지 않는 것이다. 그러나 지도 수정을 목적으로 하는 GIS환경에서는 기하 데이터를 정확성의 대상으로 해야

한다. 지도 수정 결과의 정확성은 협동 작업 중인 모든 사용자에 의해 기하 데이터의 위상 관련성에 있어 오류가 없음을 인정받는 것을 의미한다[10].

두 트랜잭션 사이에 변경 종속성이 있을 경우에는 수정이 순차적으로 진행된다. 그러나 순차적 실행이 수정 결과의 정확성을 보장할 수 없는 경우도 있다. 예를 들어 두 사람(A, B)이 공간 관련성이 존재하는 공간 데이터를 변경한다고 가정하자. 이 때, 한 작업자(A)가 공사 도면에 근거하여 지도를 변경할 때 오류를 발생 시킬 수 있으며, 도면 자체에 오류가 있을 수도 있다. 또한, 작업자들이 서로 다른 좌표의 정확도(precision)를 사용할 수도 있다. 이로 인해 수정 결과가 부정확할 수 있으며 다른 작업자(B)의 수정 결과가 정확하다 하더라도 데이터베이스의 일관성은 위배된다. 따라서 오류 발생 시에 이전 상태로 되돌리는 회복 기법이 필요하며 회복 시에 완료된 두 수정 결과는 독립적으로 복구되지 않고 서로의 관련성을 고려하여 처리되어야 한다.

서로 다른 두 트랜잭션이 공간 관련성이 있는 두 공간 데이터를 변경할 때 한 트랜잭션에서 철회가 발생하면 그 두 트랜잭션들은 하나의 철회 그룹으로 처리되어야 하는데, 이것을 회복 종속성이라고 한다. 회복 종속성은 다음과 같이 정의된다.

> 정의 1. 서로 다른 두 클라이언트 C_A, C_B에서 수행되는 서브-트랜잭션을 각각 T_A^i, T_B^j라 하고, SO_i, SO_j를 각각 TAi와 TBj가 수정하는 공간 객체라고 할 때, 두 공간 객체 SOi와 SOj간에 공간 관련성이 존재할 경우 두 서브-트랜잭션 중 하나가 철회되면 다른 서브-트랜잭션도 철회되어야 무결성이 유지된다. 이러한 두 서브-트랜잭션의 종속성을 회복 종속성(recovery dependency)이라고 정의한다.

그림 4의 예제는 트랜잭션의 회복 시에 회복 종속성이 고려되어야만 무결성이 유지됨을 설명한다. 서로 다른 서브-트랜잭션 T_A와 T_B는 공간 객체 SO_A와 SO_B를 각각 수정한다. (a)는 수정 완료 후 T_A만 철회된 결과를 보여준다. 이 경우 SO_A와 SO_B는 접하는(meet) 공간 관련성을 유지하지 못하는 상태가 된다. 따라서 데이터베이스는 무결성을 유지하지 못한다.

이 문제를 해결하기 위해 회복 종속성 개념을 적용한다. (b)는 T_A와 T_B 모두 수정을 완료한 후 T_A가 철회되면 회복 종속성에 의해 T_B도 철회된다. 따라서 SO_A와 SO_B가 변경 이전의 접하는 공간 관련성을 유지하게 된다. 즉, 공간 데이터를 변경하는 트랜잭션의 모든 철회에 대해 회복 종속성을 고려하여 처리하여야 한다.

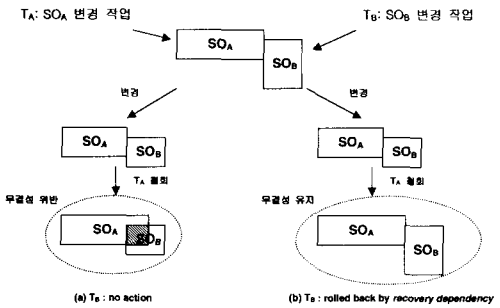

그림 4 회복 종속성을 이용한 무결성 유지

4. 회복을 위한 연산

이 논문에서 제시하는 회복 기법에서 사용되는 연산에는 *부분 완료(partial-commit)*, *부분 철회(partial-rollback)*, *연쇄 부분 철회(cascading partial-rollback)*가 있다. 연쇄 부분 철회는 *부분 취소(partial-undo)*와 *부분 재시작(partial-redo)*의 두 가지 하부 연산 중 하나를 실행한다. 이 장에서는 이러한 연산들에 대해서 자세히 설명한다.

4.1 부분 완료

이 논문은 서브-트랜잭션의 완료 및 전파를 위해서 점진적 변경 전파 기법[2]을 사용한다. 점진적 변경 전파는 *부분 완료(partial-commit)*로 설명되는데 그 정의는 다음과 같다.

> 정의 2. 클라이언트 측에서 수행되는 하나의 서브-트랜잭션에서 수정을 완료한 객체들에 대하여 정확성(correctness)을 보장하기 위해 협동 작업 중인 클라이언트들에게 변경 결과를 전파하는 연산을 부분 완료(partial-commit)라고 정의한다.

부분완료는 확장된 2단계 완료 프로토콜(SR-based 2PC)[2]에서 지도 수정 트랜잭션의 실행 중에 공간 관련성으로 인해 발생하는 오류를 없애기 위해서 사용된다. 그림 5는 부분 완료를 이용하여 건물과 도로 객체들을 점진적으로 변경하는 방법을 설명한다.

클라이언트1이 도로 객체를 부분 완료하였을 때 수정 전과의 차이를 delta라고 한다. 서버는 회복을 위해 delta를 log에 기록한 후 클라이언트2에 전파한다. 클라이언트

2는 delta를 자신의 데이터 셋에 병합(merge)한다. 모든 클라이언트의 delta 병합이 끝나면 부분 완료의 수행은 종료된다.

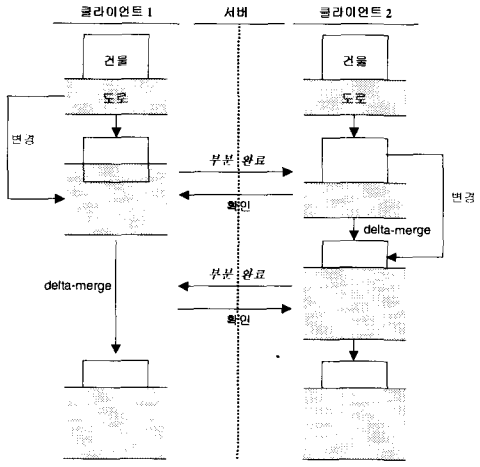

그림 5 부분 완료를 이용한 점진적 변경의 예

4.2 부분 철회와 연쇄 부분 철회

*부분 철회(partial-rollback)*는 철회의 고비용 문제를 해결하기 위해 공간 데이터 변경 트랜잭션에서 사용될 수 있는 회복의 최소 단위이며 서브-트랜잭션의 철회를 의미한다.

> 정의 3. 클라이언트 C_A에서 수행되는 트랜잭션 T_A가 있을 때, T_A의 서브-트랜잭션 T_A^i가 객체 셋 SO_{set-i}를 변경하였으나, 작업의 오류로 인해 T_A^i를 철회할 수 있는데 이것을 부분 철회(partial-rollback)라고 정의한다.

부분 철회는 공간 데이터를 수정하는 서브-트랜잭션 간의 회복 종속성 때문에 *연쇄 부분 철회(cascading partial rollback)*를 발생시키는데 다음과 같이 정의한다.

> 정의 4. 서로 다른 두 클라이언트 C_A, C_B에서 수행되는 서브-트랜잭션을 각각 T_A^i, T_B^j라고 할 때 두 트랜잭션 사이에 회복 종속성이 존재하는 경우 T_A^i의 부분 철회는 T_B^j의 부분 철회를, T_B^j의 부분 철회는 T_A^i의 부분 철회를 요구하게 된다. 이러한 종속적인 부분 철회를 연쇄 부분 철회(cascading partial-rollback)라고 정의한다.

그림 6에서 부분 철회와 연쇄 부분 철회에 대하여 자세히 설명한다. 두 클라이언트 C_A, C_B에서 서브-트랜잭션 T_A^1 - T_A^4와 T_B^1 - T_B^3가 각각 실행된다. 로그(log)에는 각 서브-트랜잭션이 종료될 때 마다 데이터 로그 레코드가 기록된다. 이 때 C_A에서 T_A^3의 철회가 발생할 경우, C_A에서는 T_A^3의 실행 이전 상태로 되돌려야 하는데 이것이 부분 철회이다(①). 그리고 T_A^3와 T_B^2는 서로 인접한 객체를 수정하여 회복 종속성이 있다. 따라서 클라이언트 C_B에서 실행된 서브-트랜잭션 T_B^2도 철회되는데, 이것이 연쇄 부분 철회이다(②). 즉, 서브-트랜잭션 중에서 회복 종속성이 존재하는 서브-트랜잭션만을 연쇄 철회되도록 제어함으로써 불필요한 철회의 발생을 방지할 수 있다.

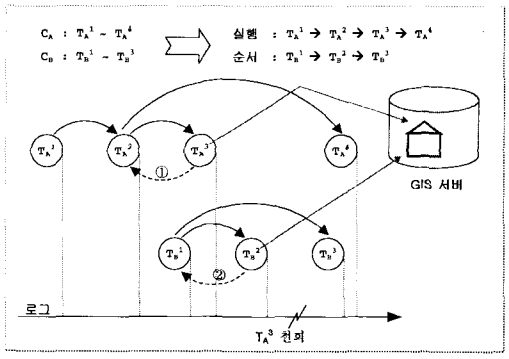

그림 6 부분 철회와 연쇄 부분 철회의 예

4.3 부분 취소와 부분 재시작

회복 종속성에 의해 연쇄 부분 철회되는 트랜잭션은 두 연산 *부분 취소(partial-undo)*와 *부분 재시작(partial-redo)* 중 하나를 실행한다. 이것은 서브-트랜잭션의 완료 여부에 따라 결정된다. 부분 취소는 중첩 트랜잭션 모델의 undo와 유사한 의미로써 완료된 서브-트랜잭션을 취소하기 위한 연산이다. 부분 재시작은 현재 수정 중인 서브-트랜잭션이 완료될 경우 잘못된 수정 결과를 초래할 수 있기 때문에 재 수정하는 연산이다. 재수정 작업은 수동적으로 사용자에 의해 작업되어야 한다. 왜냐하면 대화식 트랜잭션에서 완료된 트랜잭션의 자동적 재시작(automatic redo)은 무의미하기 때문이다[3].

부분 재시작은 철회되는 서브-트랜잭션과 회복 종속성이 있는 서브-트랜잭션이 실행 중일 경우에 발생한다. 부분 재시작을 다음과 같이 정의한다.

> 정의 5. 긴 트랜잭션 T_A의 서브-트랜잭션 T_A^i가 부분 철회될 때 긴 트랜잭션 T_B의 실행 중인 서브-

트랜잭션 T_B^j가 회복 종속성에 의해 연쇄 부분 철회될 경우 T_B^j를 재 시작해야 하는데 이것을 부분 재시작(partial-redo)라고 정의한다

부분 재시작은 서브-트랜잭션이 완료되기 전에 재시작하는 것이므로 다른 서브-트랜잭션에게 아무런 영향을 주지 않는다. 따라서 이 연산에는 서버와 부분 재시작되는 클라이언트만 관여한다.

부분 취소(partial-undo)는 완료된 서브-트랜잭션의 취소를 의미하며 다음과 같이 정의한다.

정의 6. 긴 트랜잭션 T_A의 서브-트랜잭션 T_A^i가 부분 철회될 때 긴 트랜잭션 TB의 완료된 서브-트랜잭션 T_B^j가 회복 종속성에 의해 연쇄 부분 철회될 경우 보상-트랜잭션을 이용하여 T_B^j의 영향(effects)을 제거하게 되는데 이 연산을 부분 취소(partial-undo)라고 정의한다

부분 취소는 부분 재시작 연산과 달리, 부분 완료에 의해 변경 전파된 결과를 무효화하는 작업이 실행된다.

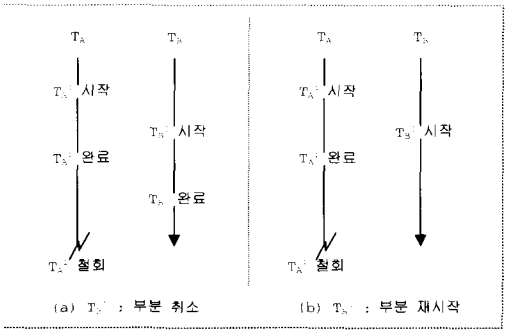

그림 7 부분 취소와 부분 재시작의 예

그림 7은 부분 취소와 부분 재시작의 예를 보여 준다. 긴 트랜잭션 T_A, T_B의 서브-트랜잭션 T_A^i, T_B^j는 공간 관련성이 있는 객체를 수정한다고 가정한다. 따라서 T_A^i와 T_B^j는 회복 종속성 관계에 있으므로 T_A^i가 부분 철회될 경우, T_B^j는 연쇄 부분 철회되어야 한다. (a)의 경우 T_A^i가 철회될 때 T_B^j는 이미 완료한 상태이므로 부분 취소 연산을 실행하여 클라이언트들에 대한 T_B^j의 영향을 제거해야 한다. (b)의 경우 T_A^i가 철회될 때 T_B^j는 미완료 상태이므로 부분 재시작 연산을 실행한다.

5. 트랜잭션 회복 기법

이 장에서는 트랜잭션 간의 관계를 유형별로 분류하고 이에 따른 연산 제어 기법을 기술한다.

5.1 회복 제어

지도 수정 작업은 사용자와의 상호 작용이 필요하므로 많은 시간이 걸린다. 따라서 작업의 철회는 고비용의 재수정을 요구한다. 지도 수정 작업의 이러한 특징 때문에 회복 기법은 철회의 발생을 줄일 수 있어야 한다. 이 논문에서 제시하는 회복 기법은 트랜잭션간의 관계를 유형별로 나누어 불필요한 연쇄 철회를 제거한다. 먼저 표 1에서는 설명을 위해 사용될 심볼들을 정리하였다.

표 1 심볼과 의미

심 볼	의미
T_A^i	트랜잭션 T_A의 서브-트랜잭션
$SR_{before}[T_A^i, T_B^j]$	수정 시작 전 T_A^i, T_B^j의 수정 객체간 공간 관련성 여부
$Partial\text{-}Commit[T_A^i]$	서브-트랜잭션 T_A^i의 완료 여부
$SR_{after1}[T_A^i, T_B^j]$	T_A^i의 완료, T_B^j의 수정 상태에서 공간 관련성 여부
$SR_{after2}[T_A^i, T_B^j]$	T_A^i의 완료, T_B^j의 완료 상태에서 공간 관련성 여부

5.1.1 트랜잭션 철회의 유형별 분류

서브-트랜잭션 T_A^i가 수정 중에 철회될 경우는 서버의 데이터를 변경하거나 다른 클라이언트에 전파되지 않은 상태이므로 쉽게 처리된다. 그러나 서브-트랜잭션이 완료 후에 철회가 발생할 경우는 그 수정 결과가 다른 트랜잭션에 영향을 주기 때문에 처리하기 어렵다.

다음에서 서브-트랜잭션 T_A^i의 완료 후 철회 발생 시에 서브-트랜잭션 T_B^j와의 관계를 8 가지 경우로 나누어 각각에 대해서 자세히 설명한다.

- 공간 관련성을 유지하는 유형 - 1 ($SR_{before}[T_A^i, T_B^j]$ = TRUE, $Partial\text{-}Commit[T_B^j]$ = FALSE, $SR_{after1}[T_A^i, T_B^j]$ = TRUE)

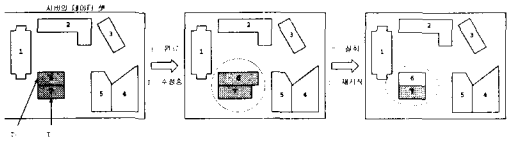

그림 8 회복 유형 - 1

그림 8은 T_A^i의 철회 발생에 대해 T_B^j가 재시작 해야함

을 보여 준다. 수정 시작부터 철회 발생 현재까지, 공간 관련성을 유지하는 수정(6과 7)이므로 계속해서 T_B^j가 실행될 경우 공간 관련성이 깨어지는 잘못된 수정을 할 수 있다.

- 공간 관련성을 유지하는 유형 - 2 ($SR_{before}[T_A^i, T_B^j]$ = TRUE, Partial-Commit$[T_B^j]$ = FALSE, $SR_{after1}[T_A^i, T_B^j]$ = FALSE)

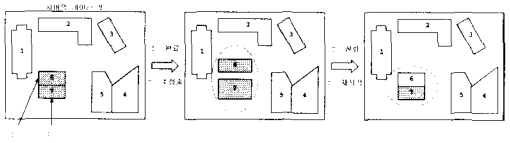

그림 9 회복 유형 - 2

그림 9는 T_A^i의 철회 발생에 대해 T_B^j가 재시작(redo)되어야함을 보여 준다. 수정 시작 전에 T_A^i와 T_B^j가 수정하는 객체(6과 7)간에 공간 관련성이 있다. 그리고 수정 중에는 공간 관련성 정보가 변경되는 것을 허락하므로 공간 관련성을 유지하는 유형의 수정이다. 따라서 정확한 수정을 위해 서브-트랜잭션 T_B^j가 재시작되어야 한다.

- 공간 관련성을 유지하는 유형 - 3 ($SR_{before}[T_A^i, T_B^j]$ = TRUE, $SR_{after2}[T_A^i, T_B^j]$ = TRUE)

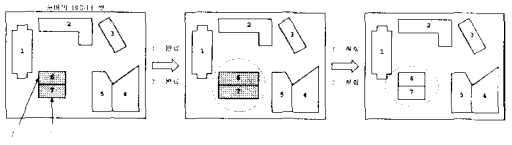

그림 10 회복 유형 - 3

그림 10은 T_A^i의 철회 발생에 대해 T_B^j도 철회되어 공간 관련성을 유지하는 예를 보여준다. T_A^i와 T_B^j의 수정 객체는 변경 전과 후에 공간 관련성이 유지되었으므로 공간 관련성을 유지하는 유형의 수정이다. 따라서 T_A^i의 철회에 의해 완료된 T_B^j도 철회되어야 한다.

- 공간 관련성을 제거하는 유형 - 1 ($SR_{before}[T_A^i, T_B^j]$ = TRUE, $SR_{after2}[T_A^i, T_B^j]$ = FALSE)

그림 11 회복 유형 - 4

그림 11은 두 수정 객체가 수정 시작 전에 공간 관련성이 있었으나 수정 완료 후에는 공간 관련성이 제거된 경우를 보여 준다. 수정 작업의 결과는 협동 작업에 의해 처리되므로 정확하다. 따라서 두 트랜잭션은 공간 관련성을 제거하는 수정을 한 것이다. 그러므로 T_A^i의 철회 발생에 대해 T_B^j는 완료된 상태를 그대로 유지해야 한다.

- 공간 관련성을 생성하는 유형 - 1 ($SR_{before}[T_A^i, T_B^j]$ = FALSE, Partial-Commit$[T_B^j]$ = FALSE, $SR_{after1}[T_A^i, T_B^j]$ = TRUE)

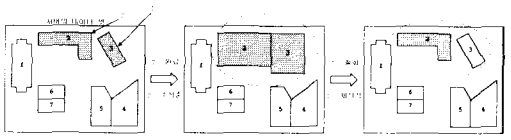

그림 12 회복 유형 - 5

그림 12는 두 수정 객체가 수정 시작 전에 공간 관련성이 없었으나 철회가 발생한 현재, 공간 관련성이 존재하므로 공간 관련성을 생성하는 수정으로 간주된다. 따라서 T_B^j의 수정 작업이 계속되면 잘못된 수정을 초래할 수 있다. 그러므로 T_A^i의 철회 발생에 대해 T_B^j는 수정 작업을 재시작 해야 한다.

- 공간 관련성이 없는 유형 - 1 ($SR_{before}[T_A^i, T_B^j]$ = FALSE, Partial-Commit$[T_B^j]$ = FALSE, $SR_{after1}[T_A^i, T_B^j]$ = FALSE)

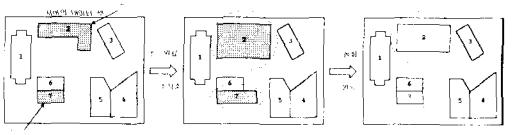

그림 13 회복 유형 - 6

그림 13은 두 수정 객체가 수정 시작 전과 철회가 발생했을 때 공간 관련성이 없는 경우를 보여준다. 즉, T_A^i와 T_B^j는 독립적으로 실행되는 트랜잭션이다. 따라서 T_B^j가 수정 작업을 계속해도 오류는 없다.

- 공간 관련성을 생성하는 유형 - 2 ($SR_{before}[T_A^i, T_B^j]$ = FALSE, $SR_{after2}[T_A^i, T_B^j]$ = TRUE)

그림 14는 수정 시작 전에 두 수정 객체간에 공간 관련성이 없었으나 협동 작업을 완료하였을 때 공간 관련성이 생성된 경우를 보여준다. 이 때 T_A^i의 철회만 처리한다면 T_B^j의 잘못된 수정이 제거되지 않아서 지도에 오류가 발

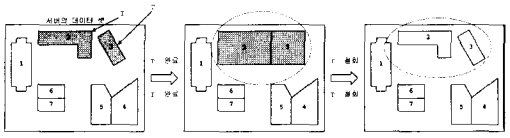

그림 14 회복 유형 - 7

생하게 된다. 그러므로 T_B^j도 철회되어야 한다.

- 공간 관련성이 없는 유형 - 2 ($SR_{before}[T_A^i, T_B^j]$ = FALSE, $SR_{after2}[T_A^i, T_B^j]$ = FALSE)

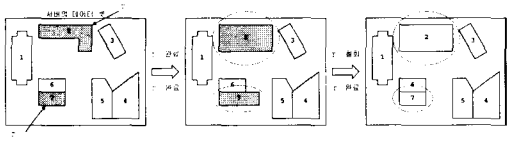

그림 15 회복 유형 - 8

그림 15는 수정 시작 전과 후에 공간 관련성 없이 두 서브-트랜잭션이 독립적으로 수정하는 경우를 보여 준다. 이 경우 각각의 철회에 대해 서로 아무런 영향을 받지 않는다. 따라서 T_A^i의 철회 발생에 대해 T_B^j는 완료된 상태를 그대로 유지해야 한다.

5.1.2 회복 연산 제어

한 서브-트랜잭션이 부분 철회되면 수행 중이거나 완료된 타 협동 트랜잭션의 서브-트랜잭션은 '*partial-commit 여부*와 'SR_{before}*의 유무*' 그리고 'SR_{after1} *또는* SR_{after2}*의 유무*'에 따라 partial-redo, partial-undo 그리고 undo-delta 중에 하나의 작업을 수행한다. partial-redo와 partial-undo는 연쇄 부분 철회에 속하는 연산이며 undo-delta는 변경 이전 값(delta)을 전송받아 undo를 실행하는 연산이다.

표 2 철회 시 서브-트랜잭션의 상태와 실행 연산

T_B^j의 상태		실행 연산	
partial-commit	SR_{before}	SR_{after2}	*partial-undo*
		¬ SR_{after2}	*undo-delta*
	¬ SR_{before}	SR_{after2}	*partial-undo*
		¬ SR_{after2}	*undo-delta*
¬ partial-commit	SR_{before}	SR_{after1}	*partial-redo*
		¬ SR_{after1}	*partial-redo*
	¬ SR_{before}	SR_{after1}	*partial-redo*
		¬ SR_{after1}	*undo-delta*

표 2는 서브-트랜잭션 T_A^i의 부분 철회에 대한 타 서브-트랜잭션 T_B^j가 실행해야 하는 연산을 5.1.1 절에서 제시된 철회 유형을 토대로 정리한 것이다. 즉, T_A^i가 철회되는 경우, 다른 서브-트랜잭션 T_B^j의 가능한 모든 상태를 분석하고 이 때 실행해야 하는 연산을 나타낸 것이다. T_B^j의 상태는 부분 완료(partial-commit) 여부, 실행 전 T_A^i의 수정 객체와의 공간 관련성 여부(SR_{before}), 실행 중 T_A^i의 수정 객체와의 공간 관련성 여부(SR_{after1}) 그리고 완료 후 T_A^i의 수정 객체와의 공간 관련성 여부(SR_{after2})의 4가지 요소에 의해 분류된다. partial-commit은 T_B^j가 완료된 상태를 나타내고 ¬partial-commit은 수정 중인 상태를 나타낸다.

실행 연산에는 undo-delta와 partial-redo 그리고 partial-undo가 있는데, undo-delta는 T_B^j가 철회된 T_A^i에 의해 영향을 받지 않은 상태일 경우에 실행되며 서버로부터 전송된 변경 이전 값(delta)으로 undo를 실행한다. partial-redo는 T_A^i에 의해 영향을 받아서 T_B^j가 수행 중일 경우에 실행되는 연산으로써 T_B^j를 재시작한다. 그리고 partial-undo는 완료된 T_B^j가 T_A^i에 의해 영향을 받은 경우에 실행되는 연산으로서 T_B^j의 실행을 취소한다.

예를 들어 클라이언트 C_A, C_B에서 두 서브-트랜잭션 T_A^i, T_B^j가 실행된다고 가정하면, T_A^i와 T_B^j에 의해 수정되는 객체 SO_i와 SO_j가 수정 시작 전에 공간 관련성이 있었고(SR_{before}) 수정을 완료한 후에도 공간 관련성이 있을 때(SR_{after2}), C_A에서 T_A^i가 부분 철회되면, C_B에서는 T_B^j의 partial-undo연산을 실행하게 된다.

표 2를 연산별로 형식화시켜 표현하면 다음과 같다.

- partial-undo = (partial-commit ∩ SR_{before} ∩ SR_{after2}) ∪ (partial-commit ∩ ¬ SR_{before} ∩ SR_{after2})
- partial-redo = (¬ partial-commit ∩ SR_{before} ∩ SR_{after1}) ∪ (¬ partial-commit ∩ SR_{before} ∩ ¬ SR_{after1}) ∪ (¬ partial-commit ∩ ¬ SR_{before} ∩ SR_{after1})
- undo-delta = (partial-commit ∩ SR_{before} ∩ ¬ SR_{after2}) ∪ (partial-commit ∩ ¬ SR_{before} ∩ ¬ SR_{after2}) ∪ (¬ partial-commit ∩ ¬ SR_{before} ∩ ¬ SR_{after1})

따라서 다음의 결과를 얻을 수 있다.

- partial-undo = partial-commit$[T_B^j]$ ∩ $SR_{after2}[T_A^i, T_B^j]$
- partial-redo = ¬partial-commit$[T_B^j]$ ∩ {$SR_{before}[T_A^i, T_B^j]$ ∪ $SR_{after1}[T_A^i, T_B^j]$}
- undo-delta = {partial-commit$[T_B^j]$ ∩ ¬SR_{after2}[

T_A^i, T_B^j]} ∪ {¬partial-commit[T_B^j] ∩ ¬ SR_{before}[T_A^i, T_B^j] ∩ ¬SR_{after1}[T_A^i, T_B^j]}

3가지 연산에 대한 비용은 undo-delta, partial-redo, partial-undo의 순서로 높아진다. 먼저, undo-delta는 변경 이전 값(delta)을 전송받아 undo하는 연산이므로 가장 비용이 적다. partial-redo는 변경 이전 값을 전송받아 undo하고 처음부터 다시 작업해야 하므로 undo-delta 보다는 고비용 연산이다. partial-undo는 자신의 클라이언트와 변경 전파했던 타 클라이언트들에 대한 undo 작업이 필요하며 이로 인해 발생하는 새로운 연쇄 부분 철회를 처리해야 하므로 가장 고비용 연산이다. 즉, 3가지 실행 연산인 undo-delta와 partial-redo 그리고 partial-undo를 제시하고 표 2의 각 상태에 대해 이 연산들 중에 저비용 연산을 우선적으로 고려하여 트랜잭션의 회복을 처리한다.

6. 회복 프로토콜

이 장에서는 5장에서 설명된 회복 기법을 위한 프로토콜을 제시한다. 트랜잭션 모델과 트랜잭션 연산 및 알고리즘을 기술하고 프로토콜 적용 예제로 자세히 설명한다.

6.1 트랜잭션 모델

그림 16은 이 논문에서 사용되는 트랜잭션 모델이다.

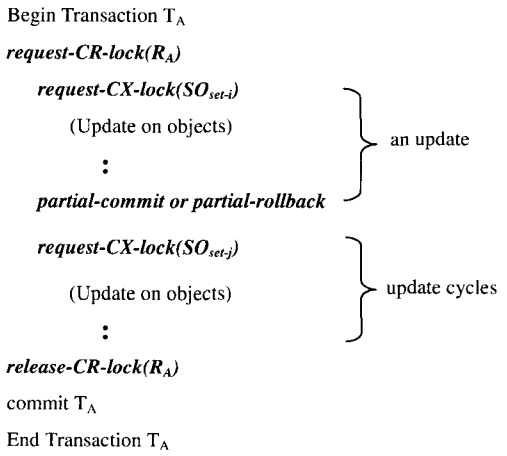

그림 16 트랜잭션 모델

이 논문에서는 트랜잭션의 회복을 위해 [10]의 트랜잭션 모델을 확장하였다. 먼저 영역 잠금(CR-lock)[10]을 서버에 요청하면서 긴 트랜잭션은 시작된다. 영역 잠금은 사용자가 객체를 수정하고자 하는 영역을 의미한다. 그리고 서브-트랜잭션은 수정할 객체들에 대하여 쓰기 잠금(CX-lock)[10]을 서버에 요청하면서 시작한다. 이 때, 교착 상태를 방지하기 위해 이 논문에서는 하나의 서브-트랜잭션에서는 한 번의 쓰기 잠금만을 설정하도록 제한한다. 클라이언트가 잠금 권한을 획득하면 그 객체들에 대하여 수정 작업을 수행한다. 그리고 작업이 완료되면 서버에 변경 결과를 전달(partial-commit요청)하여 SR 기반 2PC를 실행하는데 이것이 완료되면 서브-트랜잭션은 종료된다. 그러나 수정 작업 중에 부분 철회하여 서브-트랜잭션을 취소할 수 있다. 이 때 부분 철회되는 서브-트랜잭션은 현재 이 클라이언트에서 실행 중인 서브-트랜잭션이거나 이미 완료된 서브-트랜잭션이다. 나머지 서브-트랜잭션들은 이와 동일한 과정을 수행하며 마지막 서브-트랜잭션이 끝나면 영역 잠금 해제를 서버에 요청하여 긴 트랜잭션을 종료한다.

그림 16의 트랜잭션 모델을 제안하는 이유는 다음과 같다.

서브-트랜잭션 단위의 작업 완료에 의한 점진적 변경 기법은 다수의 클라이언트가 동시 수정하는 환경에서 동시성을 향상시킬 수 있다.

오류 발생시 철회의 단위를 서브-트랜잭션으로 처리하면 회복의 비용을 줄일 수 있다.

6.2 트랜잭션 연산

이 절에서는 클라이언트-서버 환경의 회복 프로토콜을 위한 트랜잭션 연산을 기술한다. 트랜잭션 연산은 사용자에 의해 실행되는 연산과 시스템에 의해 자동으로 실행되는 연산으로 나뉜다. 먼저 사용자에 의해 실행되는 트랜잭션 연산은 다음과 같다.

- *partial-commit* : 객체들의 수정 결과(delta)를 전달하고 공간 관련성 기반 2PC를 시작한다.
  - reply-partial-accept : 부분 완료에 대한 허락을 전파한다.
  - reply-partial-reject : 부분 완료에 대한 거부를 전파한다.
  - send-global-commit : 부분 완료를 전체 협동 작업 클라이언트에 전파한다.
  - send-global-abort : 부분 철회를 협동 작업 클라이언트에 전파한다.
- *partial-rollback* : 부분 완료에 의한 변경 중 원하는 객체의 변경을 취소하고 이전의 상태로 되돌리도록 서버에 요청한다.
  - reply-partial-rollback-ok : 부분 철회의 성공을 위한 응답 연산
  - reply-partial-rollback-fail : 부분 철회의 실패를

위한 응답 연산

- *partial-undo(cascade-partial-rollback)* : 클라이언트의 부분 철회에 의해 회복 종속성이 존재하는 완료된 서브-트랜잭션을 철회한다.
  - reply-partial-undo-ok : 클라이언트가 연쇄 부분 철회를 허락하는 응답 연산
  - reply-partial-undo-fail : 클라이언트가 연쇄 부분 철회를 거부하는 응답 연산
- *partial-redo(cascade-partial-rollback)* : 클라이언트의 부분 철회에 의해 회복 종속성이 존재하는 서브-트랜잭션이 실행 중이면 이를 재시작한다.
  - reply-partial-redo-ok : 클라이언트가 부분 재시작을 결정하는 응답 연산
  - reply-partial-redo-fail : 클라이언트가 부분 재시작을 거부하는 응답 연산
- *request-UNDO-data* : 클라이언트가 서버의 철회된 데이터를 확인하고 가져오기를 요청한다.
  - reply-UNDO-data : 철회된 데이터를 전파한다.

시스템에 의해 자동 실행되는 트랜잭션 연산은 다음과 같다.

- *undo-delta* : 부분 철회에 의해 서버가 클라이언트에 변경 이전 데이터(delta)로 undo할 것을 요청하여 클라이언트 측 회복 관리기가 이를 실행한다.

6.3 트랜잭션 알고리즘

이 절에서는 주요 트랜잭션 연산의 알고리즘과 이를 이용한 변경 전파 프로토콜을 설명한다.

6.3.1 부분 완료 알고리즘

이 논문에서는 전진적 변경 전파를 위하여 부분 완료(partial-commit)[2] 개념을 도입한다. 부분 완료는 클라이언트 트랜잭션들의 동시성을 높이고 지도 수정 시 공간 관련성에 의한 지도 수정 오류의 발생을 제거하기 위해서 협동 작업으로 수행된다. 클라이언트 트랜잭션들의 협동 작업은 기존의 2단계 완료 프로토콜(2PC)을 확장한 공간 관련성 기반 2단계 완료 프로토콜(SR 기반 2PC)[2]로 지원한다. SR 기반 2PC는 조정자와 참여자, 그리고 비 참여자간의 프로토콜로써, 부분 완료를 서버에 요청한 클라이언트가 조정자가 되며 수정 영역이 겹치는(Non-DisJoint, 이하 NDJ) 클라이언트가 참여자가 된다. 그리고 나머지 클라이언트는 비 참여자가 된다.

```
svr rcv partial commit (DELTAA)
begin
    if (no Participants)
        logging (DELTAA)
        send global-partial-commit
        return

    for (NDJ-CLIENT in Participants)
        send partial-commit (DELTAA) to NDJ CLIENT

    for (timeout)
        wait for response from CLIENT
    if (any partial-reject) then
    begin
        for (NDJ-CLIENT in Participants)
            send global-partial-abort to NDJ-CLIENT
    end
    else if (all partial-accept) then
    begin
        logging (DELTAA)
        update object with DELTAA
        for (NDJ-CLIENT in Participants)
            send global-partial-accept to NDJ CLIENT
    end
end
```

```
clnt-rcv-partial-commit (DELTAA)
begin
    Display DELTAA
    wait for user decision
    if (decision is partial-accept) then
        send partial-accept to SERVER
    else
        send partial-reject to SERVER
end
```

그림 17 부분 완료 알고리즘

그림 17은 부분 완료 알고리즘을 기술하고 있다. 부분 완료는 참여자의 유무에 따라 다르게 실행 되는데, 참여자가 없을 경우 2PC가 필요 없으므로 즉시 서버의 데이터를 변경한다. 그러나 참여자가 있을 경우에는 SR 기반 2PC를 통해 공동 작업자의 변경할 내용을 확인한 후 변경 여부를 결정한다.

6.3.2 부분 철회 알고리즘

SR 기반 2PC를 이용하여 객체 변경을 완료 및 전파하였지만, 수정 오류를 발견하였을 경우 변경 결과를 취소할 수 있다. 부분 철회(partial-rollback)는 서브-트랜잭션의 실행을 취소하기 위해 사용되는 트랜잭션 연산이다.

표 3 부분 철회에 의한 참여자의 Operations

T_A^1 \ T_B^1	Operation of Participant		
partial-rollback	undo-delta	partial-redo	partial undo

서브-트랜잭션이 철회될 때는 철회 전 데이터에 기반하여 실행된 트랜잭션들을 고려해야 한다. 표 3은 서브-트랜잭션 T_A^i에 대한 부분 철회 요청 시 참여자들의 일관성 유지를 위해 실행하는 연산의 종류를 보인다.

- *undo-delta* : 서버로부터 메시지가 전송될 때 함께 보내어지는 delta를 이용하여 undo한다. 이 연산은 부분 철회될 서브-트랜잭션에 의해 변경된 객체를 가지고 있는 클라이언트에서 실행된다. 서브-트랜잭션을 수행 중이더라도 재시작할 필요가 없으며 이 연산은 시스템에 의해 자동으로 실행된다.
- *partial-redo* : 실행 중인 서브-트랜잭션이 재시작해야 함을 의미한다. 이 연산은 부분 철회되는 서브-트랜잭션과 회복 종속성이 있는 서브-트랜잭션이 존재하는 클라이언트에서 실행된다. 이 연산은 사용자에 의해 실행 여부가 결정된다.
- *partial-undo* : 완료된 서브-트랜잭션을 취소하는 연산이다. 이 연산은 부분 철회되는 서브-트랜잭션과 회복 종속성이 존재하는 서브-트랜잭션을 실행한 클라이언트에서 수행되며 새로운 부분 철회가 발생할 수 있다. 그리고 각 클라이언트의 사용자에 의해 실행 여부가 결정된다.

Participant는 수정 진행 상태에 따라 다양한 유형으로 나뉘어지는데 자세한 유형별 분류는 표 2에서 제시하였다.

그림 18은 부분 철회 알고리즘을 설명한다. partial-rollback 메시지가 서버에 도착하면 로그로부터 delta를 가져와서 undo를 실행한다. 그리고 협동 작업에 참여했던 클라이언트들의 일관성 유지를 위하여 해당 클라이언트에 partial-undo, partial-redo, undo-delta 메시지를 각각 보낸다. 클라이언트가 cascade-partial-rollback 메시지(partial-undo 또는 partial-redo)를 받으면 undo를 실행하고 사용자의 결정을 기다린다. 사용자의 결정이 연쇄 부분 철회일 경우는 서버에 cascade-partial-rollback- ok 메시지(reply-partial-undo-ok 또는 reply-partial-redo-ok)를 보내고 아닐 경우에는 cascade-partial-rollback-fail 메시지(reply-partial-undo-fail 또는 reply-partial-redo-fail)를 서버에 보낸다. 서버에서는 클라이언트로부터 받은 응답이 reply-partial-undo-ok 일 경우에만 연쇄 부분 철회를 실행한다.

```
svr-rcv-partial-rollback (TR_IDA)
begin
    get DELTAA from LOG
    UNDO OBJA with DELTAA
    check DEP_LIST
    for (CLIENT in DEP_LIST)
    begin
      send partial-undo (DELTAA) to CLIENT
    end
    check COOP_LIST
    for (CLIENT in COOP_LIST)
    begin
      if (recovery dependency) then
        if (CLIENT is in-transaction) then
          send partial-redo (DELTAA) to CLIENT
        else
          send partial-undo (DELTAA) to CLIENT
      else
        send undo-delta (DELTAA) to CLIENT
    end
    check INDEP_LIST
    for (CLIENT in INDEP_LIST)
    begin
      store DELTAA for CLIENT
    end
end
```

```
clnt-rcv-cascade-partial-rollback (DELTAA)
begin
    UNDO OBJA with DELTAA
    wait for user decision
    if (decision is cascade-partial-rollback) then
      send cascade-partial-rollback-ok to SEVER
    else
      send cascade-partial-rollback-fail to SEVER
end
```

그림 18 부분 철회 알고리즘

6.3.3 Undo 알고리즘

이 논문에서는 협동 작업하지 않은 클라이언트의 일관성 유지를 위해 탐지기반 기법(detection-based)을 적용한다. 탐지 시기는 다음과 같이 두 가지로 나눈다.

- 새로운 긴 트랜잭션을 시작하기 위해 영역 잠금을 설정할 때
- 철회된 데이터에 대한 사용자의 요구가 있을 때

위의 두 시기가 되기 전까지 서버는 undo에 대한 정보만 유지할 뿐 전파하지 않는다. 이 기법을 도입하는 이유는 부분 철회에 의해 발생하는 서버의 과부하 문제를 해소할 수 있기 때문이다. 그림 19는 클라이언트가 undo 정보를 서버에 요청하는 알고리즘을 보여준다.

```
svr-rcv-request-UNDO-data (TR_IDA)
begin
    check UNDO_INFORM
    if (exist) then
    begin
      get DELTAA from UNDO_INFORM
      send reply-UNDO-data(DELTAA)
            to TR_IDA
```

```
        delete DELTAA from UNDO_INFORM
      end
end
```

```
clnt request-UNDO-data (TR_IDA)
begin
    send request-UNDO-data msg
          to SEVER
    for (timeout)
      wait for response
    if (data in reply UNDO-data) then
    begin
      UNDO OBJA with DELTAA
      Display OBJA
    end
end
```

그림 19 Undo 알고리즘

request-UNDO-data는 협동 작업에 참여하지 않았던 클라이언트들의 일관성 유지를 위한 메시지이다. 클라이언트가 request-UNDO-data 메시지를 보내면 서버는 해당 클라이언트에 대한 undo 정보가 있는지를 검사한다. 그리고 만약 존재한다면 클라이언트에 무효화된 데이터(delta)를 전송하게 되고 클라이언트는 undo를 실행하여 일관성을 유지한다.

6.4 회복 프로토콜 예제

이 절에서는 이 논문에서 설계한 회복 프로토콜을 이용하는 예제를 보인다. 그림 20은 각 클라이언트들이 T_A, T_B 그리고 T_C의 트랜잭션을 실행하여 서버의 데이터를 동시 변경할 때, 오류 발생에 의한 T_A의 철회 시 회복 절차를 보여준다.

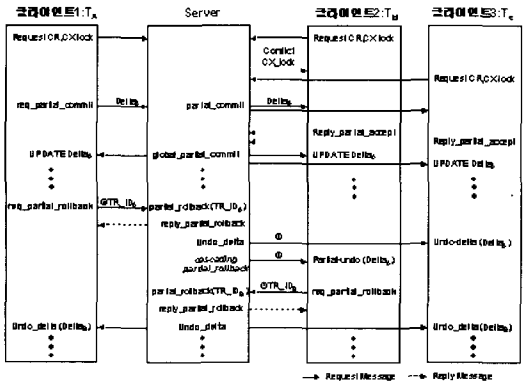

그림 20 회복 프로토콜의 예

T_A에서는 서브-트랜잭션 T_A^i가 실행 중이고 T_B에서는 T_B^j가 실행 중이며 T_C에서는 T_C^k가 실행 중이다. 그리고 각 서브-트랜잭션은 2단계 완료 프로토콜[2]에 의해 변경을 완료한다. 이 때 클라이언트1은 사용자의 오류 때문에 T_A^i의 부분 철회를 요청하게 되고(() 서버는 표 2에 따라 타 클라이언트의 트랜잭션을 제어한다. 따라서 서버는 T_A^i와 회복 종속성 없이 협동 작업 중이었던 클라이언트3에는 undo-delta 메시지를 보내고((), T_A^i와 회복 종속성이 존재하는 T_B^j에 대한 partial-undo(연쇄 부분 철회 연산) 메시지를 클라이언트2에 보낸다((). 그리고 T_B^j는 연쇄 부분 철회를 결정하고 서버에 T_B^j의 부분 철회를 요청한다((). 이에 서버는 클라이언트1과 클라이언트3에 undo-delta 메시지를 보내어 회복 작업을 종료한다.

7. 구현

그림 21은 설계 및 구현한 시스템의 구조를 보이고 있다. 서버는 DEC 400/500에 설치된 오라클 데이터베이스(Oracle 8^{TM})를 대상으로 구현하였고 클라이언트는 Windows 98 환경의 Microsoft Visual C++ 6.0 도구를 이용하여 구현하였다. 그리고 서버와 클라이언트는 소켓 통신으로 데이터를 전송한다.

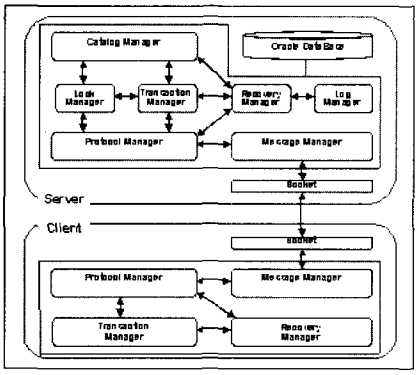

그림 21 시스템의 구조

서버는 카탈로그 관리 모듈(Catalog Manager), 잠금 관리 모듈(Lock Manager), 트랜잭션 관리 모듈(Transaction Manager), 회복 관리 모듈(Recovery Manager), 프로토콜 관리 모듈(Protocol Manager), 로그 관리 모듈(Log Manager), 그리고 메시지 관리 모듈(Message Manager)의 7개의 모듈로 구성되며 각각의 기능은 다음과 같다.

- 카탈로그 관리 모듈 : Oracle Database에 접근해서 메타 정보를 관리하는 모듈

- 잠금 관리 모듈 : 잠금 정보를 설정하거나 가져오는 모듈
- 트랜잭션 관리 모듈 : 트랜잭션 현재 상태나 트랜잭션이 실행 중인 클라이언트 등의 정보를 관리하는 모듈
- 회복 관리 모듈 : 회복에 관련된 제어를 하는 모듈
- 프로토콜 관리 모듈 : 클라이언트 요청 메시지에 대해 전체적인 프로토콜을 관리하는 모듈
- 로그 관리 모듈 : 로그에 대한 접근을 관리하는 모듈
- 메시지 관리 모듈 : 소켓을 통해 전파될 패킷을 구성하거나 분해하는 기능을 담당하는 모듈
- 클라이언트는 4개의 모듈로 구성되며 각각의 기능은 다음과 같다.
- 트랜잭션 관리 모듈 : 클라이언트에서의 트랜잭션에 대한 정보를 관리하는 모듈
- 회복 관리 모듈 : 클라이언트 내에서 회복을 위한 정보를 관리하는 모듈
- 프로토콜 관리 모듈 : 서버에 요청하는 메시지나 서버로부터 받은 메시지에 대한 처리를 관리하는 모듈
- 메시지 관리 모듈 : 소켓을 통해 전파될 패킷을 구성, 분해하는 기능을 담당하는 모듈

그림 22는 구현된 시스템을 적용하여 지도의 동시 변경에서 발생하는 트랜잭션 회복의 예를 보여 준다. 데이터셋은 도시 지역의 '주택와 건물' 레이어로써 현재 서로 다른 세 클라이언트가 협동 수정 작업을 하고 있다. 그 중 클라이언트 C_1과 C_2는 공간 관련성이 존재하는 두 객체를 수정한다. (가)는 초기 상태를 보이고 있고 (나)는 각 클라이언트의 수정 작업이 완료된 상태를 보인다. 여기서, 굵은 사각형은 사용자에 의해 설정된 영역 잠금[10]이다. (다)는 클라이언트 C_1이 부분 철회를 요청하려는 상태로써, 화면의 대화상자는 C_1에서 수행된 서브-트랜잭션의 리스트를 보여준다. (라)는 서버가 C_1으로부터 서브-트랜잭션(164.125.69.2447009-0)의 부분 철회에 대해 회복 작업을 실행한 결과를 보여준다. C_1은 reply partial-rollback-ok 메시지를 받아 undo를 실행한 상태이고, C_2는 부분 철회되는 서브-트랜잭션과 회복 종속성이 있는 서브-트랜잭션(164.125.69.2447010-0)에 대한 partial-undo 메시지를 받은 상태이다. 그리고 C_3는 서버로부터 undo-delta 메시지를 받아 변경 이전 데이터로 업데이트를 실행하려는 상태를 보인다. (마)는 C_2에서 요청한 서브-트랜잭션(164.125.69.2447010-0)의 연쇄 부분 철회에 대한 회복 작업을 제어한 결과를 보여준다. C_1과 C_3에는 더 이상 회복 종속성이 있는 서브-트랜잭션이 없으므로 undo-delta 메시지가 도착하였고 C_2는 서버로부터 응답을 받아 undo 작업을 완료하였다. (바)는 작업이 완료된 상태를 보인다.

8. 결론 및 향후 연구

이 논문은 서버의 공간 데이터가 클라이언트에서 실행되는 트랜잭션에 의해 변경되는 클라이언트-서버 환경에서 트랜잭션이 철회(rollback)될 때 발생하는 문제를 해결하기 위한 회복 기법을 다루었다.

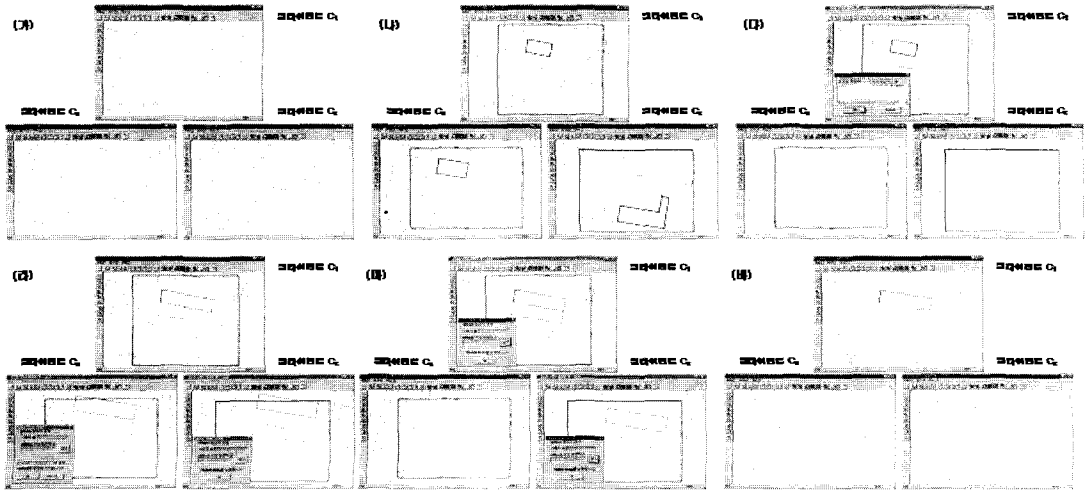

그림 22 구현된 시스템에 의한 트랜잭션 회복의 예

이 논문에서 제시하는 트랜잭션 회복 기법의 주요 내용은 다음과 같다. 첫째, 트랜잭션의 철회에 의해 공간 관련성이 유지되지 못하는 문제를 트랜잭션간의 회복 종속성 개념을 정의하고 이것을 회복 기법에 적용하여 해결하였다. 둘째, 실행 시간이 긴 지도 수정 트랜잭션이 전체 철회(total rollback)될 경우 고비용의 재작업이 필요하다. 이를 해소하기 위해 회복의 단위를 서브-트랜잭션으로 처리하는 부분 철회(partial-rollback)를 제시하였다. 셋째, 미완료된 데이터를 읽은 트랜잭션을 연쇄 철회(cascading rollback)의 대상으로 하는 기존의 트랜잭션 회복 기법을 수정 객체의 주위 객체에도 잠금을 설정하는 지도 수정 트랜잭션에 적용하면 불필요한 연쇄 철회 트랜잭션이 증가하는 문제가 있다. 이 논문은 회복의 상태를 분류하고 회복 연산인 undo-delta와 부분 재시작(partial-redo) 그리고 부분 취소(partial-undo)를 각 상태에 적절하게 적용하는 회복 제어 기법을 제시하여 불필요한 연쇄 철회 트랜잭션을 줄였다. 마지막으로 이 논문에서 제시한 회복 기법을 증명하기 위해 회복 제어 알고리즘 및 프로토콜을 설계하고 구현하여 결과를 보였다.

이 논문에서 다루지 않고 있는 시스템 오류(system failures)와 미디어 오류(media failures)에 대해서는 향후 연구가 필요하다. 또한 공간 데이터 변경 트랜잭션의 회복 기법에 대한 상세한 성능 평가 및 분석이 향후 연구되어야 한다.

참 고 문 헌

[1] J. Eliot, B. Moss, "Log-Based Recovery for Nested Transactions", Proceedings of the 13th VLDB Conference, pp.427-423, 1987

[2] 최진오, 홍봉희, "분산된 지리정보시스템에서 새로운 잠금 기법을 이용한 중복된 공간 데이터의 변경 전파",한국정보과학회 논문지, vol.26, no.9, pp.1061-1072, 1999

[3] Gail E. Kaiser, "Cooperative Transactions for Multiuser Environments", Modern Database Systems, pp.409-433, 1995

[4] Henry F. Korth, Eliezer Levy, Abraham Silberschatz, "A Formal Approach to Recovery by Compensating Transactions", Proceedings of the 16th VLDB Conference, PP.95-106, 1990

[5] David B. Lomet, "MLR : A Recovery Method for Multi-level Systems", ACM SIGMOD, pp.185-194, 1992

[6] Jin-oh Choi, Young-sang Shin, and Bong-hee Hong, "Update Propagation of Replicated Data in Distributed Spatial Databases', Proc. of 10th International Conference on DEXA '99, pp.952-963, 1999

[7] G. Alonso, D. Agrawal, A. El Abbadi, "Reducing Recovery Constraints on Locking based Protocols", ACM SIGMOD/PODS 94, pp.129-138, 1994

[8] R. Laurini, "Fundamentals of Spatial Information Systems", Academic Press, 1992

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

[10] 신영상, 최진오, 조대수, 홍봉희, "클라이언트 변경 트랜잭션에서 동시성 및 일관성 제어", '99 한국정보과학회 가을 학술발표논문집, vol.26, no.2, pp.323-325, 1999

[11] Abraham Silberschatz, Henry F. Korth, S.Sudarshan, "Chapter 15. Recovery System", Database System Concepts, pp.511-542, 1997

[12] C. Mohan, "ARIES/CSA: A Method for Database Recovery in Client-Server Architectures", ACM SIGMOD, vol.23, no.2, pp.55-66, 1994

[13] C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, Peter Schwarz, "ARIES : A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", ACM Transactions on Database Systems, Vol.17, No.1, pp.94-162, 1992

[14] Theo Haerder, Kurt Rothermel, "Concepts for Transaction Recovery in Nested Transactions", ACM, pp.239-248, 1987

[15] William E. Weihl, "Transaction Processing Techniques", Distributed Systems, pp.329-352, 1993

[16] Won Kim, Henry F. Korth, "A Model of CAD Transactions", Proceedings of VLDB 85, pp.25-33, 1985

[17] 박재관, 김동현, 최진오, 홍봉희, "클라이언트-서버 환경에서 공간 데이터 변경 트랜잭션을 위한 회복 기법의 설계 및 구현", '2000 한국정보과학회 가을 학술발표논문집, vol.27, no.2, pp.101-103, 2000

박 재 관

1999년 2월 부산대학교 컴퓨터공학과 졸업(공학사). 2001년 2월 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2001년 3월 ~ 현재 부산대학교 대학원 컴퓨터공학과 박사과정 재학중.관심분야는 지리정보 시스템, 모바일 GIS, 웹 GIS, 공간색인

최 진 오

1991년 2월 부산대학교 컴퓨터공학과 졸업(공학사). 1995년 2월 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 1998년 ~ 2002년 경동대학교 정보통신공학부 전임강사. 2000년 2월 부산대학교 대학원 컴퓨터공학가 졸업(공학박사). 2000년 ~ 현재 부산외국어대학교 컴퓨터전자공학부 조교수. 관심분야는 지리정보 시스템, 분산시스템, 모바일 데이타베이스

홍 봉 희

1982년 2월 서울대학교 컴퓨터공학과 졸업(공학사). 1984년 2월 서울대학교 대학원 컴퓨터공학과 졸업(공학석사). 1988년 8월 서울대학교 대학원 컴퓨터공학과 졸업(공학박사). 1987년 4월 ~ 현재 부산대학교 공과대학 컴퓨터공학과 교수. 현재 부산대학교 컴퓨터 및 정보통신 연구소 연구원, 부산대학교 대학원 GIS학과 주임 교수