멀티 데이타베이스 시스템에서 고장을 허용하는 트랜잭션 관리

신 성 철† 황 부 현††

요 약

멀티 데이타베이스 시스템에서 가장 중요한 사항은 지역 자치성의 보장과 전역적 일관성을 유지하는 것이다. 전역적 일관성은 전역 동시성 제어 알고리즘과 회복 알고리즘에 의해 유지될 수 있다. 본 논문은 지역 자치성을 보장하면서 전역적 직렬성을 만족하는 전역 동시성 제어 알고리즘과 지역 자치성을 보장하면서 각종 고장으로부터 멀티 데이타베이스를 회복할 수 있는 전역 회복 알고리즘을 제안한다. 제안된 전역 동시성 제어 알고리즘은 상향식 방법으로 3단계 트랜잭션 처리 모델에 기반을 두고, 지역 트랜잭션들로 인하여 부트랜잭션들 사이에 간접 충돌이 발생할 때 서버에서 가상연산을 사용하여 지역 데이타베이스 시스템에서 수행되는 부트랜잭션들의 수행 순서와 직렬화 순서가 동일하게 유지할 수 있도록 함으로써 간접 충돌 문제를 해결할 수 있다. 전역 모듈에서는 충돌 관계가 있는 전역 트랜잭션들에 대해서만 전역적 직렬성을 검사함으로써 전역적 직렬성 검사를 더욱 효율적으로 할 수 있다.

A Fault Tolerant Transaction Management in Multidatabase Systems

Seoung Chul Shin† and Bu Hyun Hwang††

ABSTRACT

In the multidatabase systems(MDBS), local autonomy and global consistency are important issues. Global consistency could be maintained by a global concurrency control algorithm and a global recovery algorithm. In this thesis, we propose a global concurrency control algorithm to ensure local autonomy and to guarantee global serializability, and a global recovery algorithm which is possible to recover the multidatabase from any failures. The proposed global concurrency control algorithm uses bottom-up approach, based on three-level transaction processing model. It can produce a local history that the execution order of subtransactions is identical to their serialization order by using dummy-operations in the server when an indirect conflict is caused between subtransactions due to local transactions. At the global module, it can efficiently validate global serializability of global transactions by checking global serializability only for the global transactions which conflict with each other.

1. 서 론

멀티 데이타베이스 시스템(multidatabase systems)은 데이타 모델, 데이타 정의 및 조작어, 트랜잭션 관리 전략 그리고 내부 구조가 서로 상이한 데이타베이스 시스템들이 통신망으로 상호 연결되어 있는 분산 데이타베이스 시스템이다[9]. 이질적인 지역 데이타베이스 시스템들을 논리적으로 통합할 때 고려해야 할 중요한 점은 설계 자치성, 수행 자치성, 통신 자치성으로 특징 지워질 수 있는 지역 자치성(local autonomy)[3]과 전역 데이타베이스의 일관성이다.

- 이 논문은 1993년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.
† 정 회 원 : 전남대학교 전산통계학과
†† 정 회 원 : 전남대학교 전산학과 교수
논문접수 : 1994년 3월 21일, 심사완료 : 1994년 5월 16일

본 논문은 멀티 데이타베이스 시스템의 트랜잭션 관리에 촛점을 둔다. 멀티 데이타베이스 시스템에서 트랜잭션 관리의 목적은 지역 자치성을 보장하면서 트랜잭션 수행의 동시성을 높이고, 전역 데이타베이스의 일관성을 보장하는 것이다. 전역 데이타베이스의 일관성 유지는 전역 동시성 제어 알고리즘과 그에 관련된 전역 회복 알고리즘에 의하여 이루어진다. 전역적 일관성(global consistency)은 전역 트랜잭션들의 원자성(atomicity), 일관성(consistency), 고립성(isolation), 지속성(durability)을 보장하므로써 유지될 수 있다. 전역 트랜잭션들의 일관성과 고립성은 동시성 제어 알고리즘에 의해 달성될 수 있으며, 원자성과 지속성은 전역 회복 알고리즘에 의해 보장될 수 있다.

본 논문에서는 상향식 방법을 사용하여 지역 자치성과, 전역적 직렬성(global serializability)을 보장하는 전역 동시성 제어 알고리즘과, 각종 고장으로부터 멀티 데이타베이스를 회복시킬 수 있는 전역 회복 알고리즘을 제안한다. 제안된 전역 동시성 제어 알고리즘은 각 지역 사이트의 상위 계층에 존재하는 서버에서 부트랜잭션들의 수행 순서(execution order)와 직렬화 순서(serialization order)의 불일치 문제를 해결하기 위하여 가상연산을 사용하는 낙관적 가상연산 방법(Optimistic Dummy-operations Method)이다. 이 가상연산은 각 지역 사이트에 존재하는 자료 항목에 대해 그 값을 변화시키지 않고 단순히 판독하여 다시 기록하는 연산이다.

본 알고리즘은 지역 트랜잭션으로 인하여 부트랜잭션들 사이에 간접 충돌이 발생되는 경우에만 가상연산을 사용하여 그 부트랜잭션들 사이에 직접 충돌을 발생시킴으로써, 부트랜잭션들의 수행 순서와 직렬화 순서의 불일치 때문에 전역적 직렬성이 파괴되는 문제를 해결한다. 전역 모듈에서는 각 지역 사이트에서 부트랜잭션들 사이에 충돌이 발생된 전역 트랜잭션들에 대해서만 전역적 직렬성 검사를 한다. 제안된 전역 회복 알고리즘도 지역 자치성을 보장하면서 멀티 데이타베이스 시스템에서 발생될 수 있는 각종 고장으로부터 멀티 데이타베이스를 회복시켜 전역 데이타베이스의 일관성을 유지하도록 한다.

본 논문의 구성은 다음과 같다. 2장에는 관련 연구, 3장에는 트랜잭션 처리 모델과 제안된 낙관적 가상연산 방법, 4장에는 제안된 전역 회복 알고리즘에 대해 기술하였으며, 5장에는 결론을 기술하였다.

2. 관련 연구

멀티 데이타베이스 시스템에서 서로 다른 지역 동시성 제어 알고리즘들을 논리적으로 통합하는 방법은 하향식 방법(top-down approach)과 상향식 방법(bottom-up approach)으로 분류된다. 일반적으로 하향식 방법[1, 4, 8, 10]은 전역 단계에서 전역 트랜잭션들의 전역적 직렬화 순서를 결정하고 그 순서대로 각 지역 사이트에서 전역 트랜잭션들이 수행되도록 하는 방법으로 비관적(pessimistic)이다. 이 방법은 지역 자치성을 보장할 수 있고 완료준비 상태(prepare-to-commit state)에 있는 전역 트랜잭션들을 철회하지 않지만, 전역 트랜잭션들의 많은 지연이 발생[4, 8, 10]되어 트랜잭션 수행에 있어 낮은 동시성을 제공한다. 또한 [1]에서 제안된 트랜잭션 그래프에서는 전역 트랜잭션이 완료되자마자 트랜잭션 그래프에서 제거되고, 예제 2.1과 같이 각 지역 사이트에서 전역 트랜잭션들의 수행 순서와 직렬화 순서의 불일치 문제가 발생한다면 이 방법은 전역적 직렬성을 위배할 수 있다[6].

예제 2.1 : 직렬화 순서와 수행 순서의 불일치

사이트 S_1과 S_2에 자료 항목 {a}와 {b, c}가 각각 존재한다고 하자. 또한 G_1, G_2는 전역 트랜잭션, L_1은 지역 트랜잭션 그리고 G_1과 G_2의 전역 스케줄 GS_1이 다음과 같다고 가정하자.

$G_1 : W_{g1}(a)\ R_{g1}(b)$

$G_2 : R_{g2}(a)\ W_{g2}(c)$

$GS_1 : W_{g1}(a)\ R_{g1}(b)\ R_{g2}(a)\ W_{g2}(c)$

사이트 S_1과 S_2의 이력(history)이 H_1과 H_2라고

하자.

H_1 : $W_{g1}(a)$ $R_{g2}(a)$

H_2 : $W_{l1}(b)$ $R_{g1}(b)$ $W_{g2}(c)$ $R_{l1}(c)$

H_1에서 G_1과 G_2는 직접 충돌이 발생하였고, H_2에서는 G_1과 G_2가 L_1으로 인해 간접 충돌이 발생하였다. H_1과 H_2에서 전역 트랜잭션들의 수행 순서는 모두 G_1G_2이다. 그러나 H_2에서는 L_1으로 인한 간접 충돌로 인하여 직렬화 순서가 G_2G_1이 되기 때문에 수행 순서와 직렬화 순서가 다르게 된다. 그래서 S_1에서 G_1과 G_2의 직렬화 순서는 G_1G_2이고, S_2에서의 직렬화 순서는 G_2G_1이 되어 전역 이력은 직렬 가능하지 않다.

그러나 상향식 방법[5, 6]은 낙관적(optimistic)으로, 즉 일단 모든 전역 트랜잭션들을 수행하고 그 전역 트랜잭션들이 완료준비 상태에 도달되었을 때 전역 단계에서 각 지역 사이트의 동시성 제어 정보를 이용하여 전역적 직렬성을 검사하는 방법이며, 트랜잭션들의 길이가 짧다든지 또는 트랜잭션들 사이에 충돌이 적게 발생되는 경우에 트랜잭션 수행의 동시성 정도를 높이는 방법이다. 이 방법은 완료준비 상태에 있는 전역 트랜잭션들의 철회가 많이 발생될 수 있다.

[6]에서 제안된 낙관적 티켓 방법(Optimistic Ticket Method)은 상향식 방법이면서도 지역 자치성을 보장한다. 이 방법은 모든 전역 트랜잭션들에게 지역 사이트에서 전역 트랜잭션들만이 접근할 수 있는 티켓 값을 갱신하도록 하고 그 값으로 지역 사이트에서 부트랜잭션들의 상대적 직렬화 순서를 전역 단계에서 알 수 있도록 하였으며, 예제 2.1과 같은 간접 충돌 문제를 해결하였다. 그러나 낙관적 티켓 방법은 모든 전역 트랜잭션들에게 티켓 값을 갱신하도록 하므로써 모든 전역 트랜잭션들 사이에 티켓 충돌(ticket conflict)로 인한 직접 충돌이 발생한다. 이 티켓 충돌로 인하여 각 지역 사이트에서 부트랜잭션들의 철회가 많아진다.

전역 회복 기법은 전역 트랜잭션들의 원자성과 지속성을 보장하는 것으로 부트랜잭션들의 영향을 지역 데이타베이스로 반영시키는 시점에 따라 두 가지 방법으로 분류된다. 그 하나는 전역 트랜잭션이 전역적으로 완료되었을 때 지역 데이타베이스에 부트랜잭션들의 영향을 반영시키는 경우에 사용될 수 있는 재시작(restart) 방법으로, 전역적으로 완료되었으나 지역적으로 철회된 부트랜잭션을 재시작시킴으로써 그 부트랜잭션의 영향이 그 지역 데이타베이스에 반영되도록 하는 방법이다. 다른 하나는 부트랜잭션의 마지막 연산이 수행되자마자 지역 데이타베이스에 부트랜잭션의 영향을 반영시키는 경우에 사용될 수 있는 보상(compensation) 방법으로, 전역적으로 철회되었으나 지역적으로 완료된 부트랜잭션에 대한 보상 트랜잭션을 수행시킴으로써 그 부트랜잭션의 영향을 그 지역 데이타베이스에서 제거하는 방법이다[8]. 그러나 보상 방법은 미사일 발사 시스템과 같이 트랜잭션의 수행 결과가 바로 행동으로 나타나는 시스템에서는 그 결과가 잘못되었을 때 회복할 수 없기 때문에 일반적인 방법이라 할 수 없다. 재시작 방법에는 고장난 트랜잭션을 다시 수행하는 재시도(retry) 방법과 갱신 연산만을 다시 수행하는 재수행(redo) 방법이 있다.

완료준비 상태는 트랜잭션의 연산들이 모두 수행된 후 지역 사이트 고장을 제외한 어떠한 이유로도 지역 동시성 제어 알고리즘에 의해 철회(지역 교착상태 등으로)되지 않는 상태를 말한다. 멀티 데이타베이스 시스템에서 각 지역 사이트는 완료준비 상태를 지원하는 경우와 지원하지 않는 경우로 구분할 수 있다. 지역 사이트가 완료준비 상태를 지원한다면 서버에서는 단순히 지역 사이트로부터 수신한 부트랜잭션들의 완료준비 상태를 이용하여 원자적 완료 프로토콜(atomic commitment protocol)의 참여자로서의 역할을 수행할 수 있다. 만약 지역 사이트가 완료준비 상태를 지원하지 않는다면 각 사이트 상위에 존재하는 서버에서 완료준비 상태를 지원하여야 한다. 그러나 지역 사이트에서 수행을 끝낸 부트랜잭션이 지역 교착상태의 발생으로 인하여 그 희생자로 선정된다면 서버에서는 완전한 의

미의 완료준비 상태를 지원할 수가 없다. 이는 서버에서 부트랜잭션에 대한 원자적 완료 프로토콜의 참여자로서의 역할을 수행할 수 없기 때문에 완료준비 상태에 준한 어떤 다른 상태를 지원하여야만 한다. 이를 위하여 한 지역 사이트에서 수행되는 부트랜잭션의 연산들이 전부 수행된 상태를 가상의 완료준비 상태(simulated prepare-to-commit state)라고 한다. 지역 사이트가 가상의 완료준비 상태를 지원한다는 것은 이 상태에 있는 부트랜잭션들이 지역 동시성 제어 알고리즘에 의해 철회되지 않는다는 보장을 할 수 없다는 것을 의미한다[7]. 지역 사이트에서 수행 중인 부트랜잭션이 이 상태에 도달하면 서버는 원자적 완료 프로토콜의 참여자로서의 역할을 수행할 수 있다.

멀티 데이타베이스 회복시 다음과 같은 문제점들이 발생될 수 있다. (1) 전역 회복 기법으로 재시도 방법을 사용하는 경우, 사이트 고장이 발생하였을 때 어떤 지역 사이트가 예제 2.2와 같은 이력을 생성한다면 갱신 분실 문제가 발생되어 전역 데이타베이스의 일관성을 파괴할 수 있다. (2) 지역 사이트에 고장이 발생하여 전역적 회복이 완료되기 전에 재수행되는 부트랜잭션들과 충돌 관계가 있는 새로운 지역 트랜잭션들을 수행하도록 한다면 예제 2.3과 같은 갱신 분실 문제가 발생하여 전역적 일관성을 파괴할 수 있다. (3) 만약 지역 사이트가 완료준비 상태를 지원하지 않을 때, 가상의 완료준비 상태에 있는 부트랜잭션들이 지역 동시성 제어 알고리즘에 의해 철회된다면 예제 2.4와 같이 전역적 직렬성 위배 문제가 발생될 수 있다.

예제 2.2 : 갱신 분실 문제 1

사이트 S_1과 S_2에 자료 항목 {a}와 {b}가 각각 존재한다고 하자. G_1을 전역 트랜잭션, L_2를 지역 트랜잭션, PC를 완료준비 상태, C를 완료 연산, A를 철회 연산이라고 하자.

G_1 : $R_{g1}(a)$ $R_{g1}(b)$ $W_{g1}(b)$

L_2 : $W_{12}(b)$

G_1이 전역적으로 완료되어 S_1에서는 완료되었지만 S_2에서는 지역적으로 완료되지 않았을 때 사이트 고장이 발생했다고 가정하자. 그러면 S_2에서는 G_1에 대한 재시도 트랜잭션 G_3를 수행한 지역 이력이 LH_2라고 가정하자.

LH_2 : $R_{g1}(b)$ $W_{g1}(b)$ PC_{g1} $W_{12}(b)$ C_{12} Failure A_{g1} $R_{g3}(b)$ $W_{g3}(b)$

이때 재시도되는 G_3로 인해 $W_{12}(b)$의 값을 잃게 되는 갱신 분실 문제가 발생한다.

예제 2.3 : 갱신 분실 문제 2

사이트 S_1과 S_2에 자료 항목 {a}와 {b}가 각각 존재하고, G_1을 전역 트랜잭션, L_2를 사이트 S_1에 제출된 지역 트랜잭션이라고 가정하자.

G_1 : $R_{g1}(a)$ $W_{g1}(a)$ $W_{g1}(b)$

L_2 : $R_{12}(a)$ $W_{12}(a)$

G_1이 전역적으로 완료되었다고 가정하자. 사이트 S_2에서는 G_1에 대한 완료 연산을 수행하였으나, 사이트 S_1에서는 G_1에 대한 완료 연산을 수행하기 전에 사이트 고장이 발생하여 G_1이 철회되었다고 하자. 그런 다음 사이트 S_1에서 G_1을 재수행하기 전에 지역 트랜잭션 L_2를 먼저 수행하고 완료할 수 있다. G_1에 대한 재수행 트랜잭션 G_3: $W_{g3}(a)$를 수행한 후의 S_1의 지역 이력이 LH라고 하자.

LH : $R_{g1}(a)$ $W_{g1}(a)$ PC_{g1} Failure A_{g1} $R_{12}(a)$ $W_{12}(a)$ C_{12} $W_{g3}(a)$ C_{g3}

이렇게 되면 재수행 되는 $W_{g3}(a)$로 인하여 $W_{12}(a)$의 값을 잃게 되는 갱신 분실 문제가 발생한다.

예제 2.4 : 전역적 직렬성 위배 문제

사이트 S_1과 S_2에 자료 항목 {a, b}와 {x, y}가 각각 존재한다고 하자. 또한 G_1, G_2를 전역 트랜잭션, L_3, L_4를 지역 트랜잭션들이라고 하자.

G_1 : $W_{g1}(a)$ $W_{g1}(x)$

G_2 : $W_{g2}(b)$ $W_{g2}(y)$

L_3 : $R_{13}(a)$ $R_{13}(b)$

L_4 : $R_{14}(x)$ $R_{14}(y)$

이때 G_1과 G_2는 전역적으로 완료가 되어 G_2는 두 사이트에서 모두 완료하고, G_1은 사이트 S_2에서

만 완료하고 사이트 S_1에서는 G_1의 완료 연산을 수행하기 전에 지역 교착상태 해결을 위한 희생자로 선정되어 G_1이 철회되었다고 가정하자. 그러던 사이트 S_1에서 재수행 트랜잭션 G_3:$W_{g3}(a)$이 수행하게 되는데 그때 사이트 S_1과 S_2의 지역 이력이 H_1과 H_2라고 하자.

$H_1 : W_{g1}(a)\ A_{g1}\ W_{g2}(b)\ C_{g2}\ R_{t3}(a)\ R_{t3}(b)\ C_{t3}$
$W_{g3}(a)\ C_{g3}$

$H_2 : W_{g1}(x)\ C_{g1}\ R_{t4}(x)\ R_{t4}(y)\ C_{t4}\ W_{g2}(y)\ C_{g2}$

전역 트랜잭션 G_1과 G_2의 직렬화 순서는 사이트 S_2에서는 G_1G_2이고, 사이트 S_1에서는 L_3로 인해 G_2G_1이 되어 전역적 직렬성을 위배하게 된다.

이와 같은 문제들은 (1) 지역 동시성 제어 알고리즘에 의해 생성되는 지역 이력에 제약이 가해져야 하며, (2) 지역 사이트의 고장 회복 후 재수행되는 트랜잭션들과 충돌 관계가 있는 새로운 트랜잭션들을 전역적 회복이 완료되기 전에 수행할 수 없도록 하고, (3) 지역 사이트가 완료 준비 상태를 지원하지 않을 때에는 전역 트랜잭션에 대한 전역적 결정이 지역적으로 수행되기 전에 가상의 완료준비 상태에 있는 부트랜잭션들과 충돌 관계가 있는 새로운 트랜잭션들을 수행할 수 없도록 하므로써 해결할 수 있다.

3. 전역 동시성 제어 알고리즘

본 논문에서 전역 동시성 제어 알고리즘으로 제안한 낙관적 가상연산 방법은 지역 트랜잭션들로 인해 전역 트랜잭션들 사이에 간접 충돌이 발생되어 전역적 직렬성이 파괴되는 문제를 해결하면서, 전역 트랜잭션들의 불필요한 지연이나 철회가 발생되지 않도록 하는 데에 중점을 둔 상향식 방법이다.

본 알고리즘은 지역 자치성을 보장하면서 전역적 직렬성을 만족하는 전역 동시성 제어 알고리즘으로 지역 데이타베이스 시스템에서 부트랜잭션들의 수행 순서와 직렬화 순서를 동일하게 유지시키고, 부트랜잭션들 사이에 충돌이 발생된 전역 트랜잭션들에 대해서만 전역적 직렬성을 검사하도록 하였다.

본 알고리즘에서는 다음과 같은 가정을 한다.

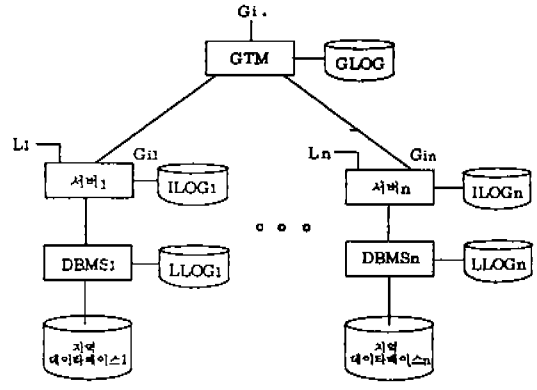
(1) 모든 각 지역 동시성 제어 알고리즘은 직렬 가능한 지역 이력을 생성한다.

(2) 트랜잭션들이 접근하는 자료 항목들을 트랜잭션 수행 전에 알 수 있다.

가정 (1)은 대부분의 지역 동시성 제어 알고리즘들이 직렬 가능한 이력을 생성하기 때문에 타당하다. 가정 (2)는 트랜잭션들이 수행되기 전에 트랜잭션들이 사용하는 물리적 수준의 자료 항목(예를 들면, 레코드 수준의 자료 항목)들은 알 수 없으나, 논리적 수준의 자료 항목(예를 들면, 릴레이션 수준의 자료 항목)들은 알 수 있기 때문에 타당하다.

3.1 트랜잭션 처리 모델

본 논문에서 사용되는 트랜잭션 처리 모델은 (그림 1)과 같다. 본 모델은 3단계 모델로서 전역 모듈, 서버, 지역 모듈의 세 요소로 구성되어 있다. 전역 모듈에서 전역 트랜잭션들에 대한 지연은 없다. 서버는 각 지역 사이트의 상위 계층에 존재하며, 지역 트랜잭션 관리자로 제출되는 지역 트랜잭션들이 서버로 제출된다. 서버에서도 전역



(그림 1) 트랜잭션 처리 모델
(Fig. 1) Transaction processing model

트랜잭션들의 지연은 없다. 다만 전역 회복 단계에서 재수행되는 전역 트랜잭션들과 충돌이 발생되는 지역 트랜잭션들을 제외하면 지역 트랜잭션들도 전혀 지연되지 않는다. 지역 모듈은 일반적인 지역 데이타베이스 시스템과 동일하며 지역 트랜잭션과 전역 트랜잭션을 구분하지 않는다.

3.2 가상연산

멀티 데이타베이스 시스템에서 일반적으로 많이 사용되는 전역 동시성 제어 알고리즘의 정확성에 대한 평가 기준은 전역적 직렬성이다. 그러나 전역적 직렬성은 부트랜잭션들의 수행 순서가 모든 지역 사이트에서 동일하다 해도 보장되지 않는 경우가 발생될 수 있다. 이는 지역 트랜잭션들로 인한 부트랜잭션들 사이에 간접 충돌이 발생되어 지역 사이트에서 부트랜잭션들의 직렬화 순서가 그들의 수행 순서와 달라지기 때문이다. 본 알고리즘은 각 지역 사이트에서 부트랜잭션들의 직렬화 순서를 결정하기 위하여 지역 트랜잭션들로 인하여 부트랜잭션들 사이에 간접 충돌이 발생되는 경우에만 그 부트랜잭션들에게 가상연산을 사용한다.

서버에서는 (1) 지역 트랜잭션들과 부트랜잭션들 사이의 충돌 관계가 표현되는 지역/전역 트랜잭션 충돌 그래프의 유지, (2) 부트랜잭션들을 지역 트랜잭션 관리자에게 제출, (3) 가상연산의 사용, (4) 충돌이 발생된 부트랜잭션들의 직렬화 순서가 표현되는 직렬화 순서 그래프 유지, (5) 부트랜잭션들의 수행 결과를 전역 트랜잭션 관리자에게 송신, (6) 원자적 완료 프로토콜의 참여자로서의 기능들을 수행한다.

정의 3.1 : 지역/전역 트랜잭션 충돌 그래프 (Local & Global transaction Conflict Graph : LGCG)

LGCG = (V, E)는 무방향 그래프이다.
V : 지역 또는 부트랜잭션들의 집합
E : $rwset(T_i) \cap rwset(T_j) \neq \phi$ 일 때
T_i — T_j로 생성되는 선분들의 집합
rwset : 판독 또는 갱신 연산들의 집합
T_i, T_j : 부 또는 지역 트랜잭션

전역 부트랜잭션 스케줄러는 트랜잭션들의 충돌 관계를 표현하기 위하여 지역/전역 트랜잭션 충돌 그래프를 유지한다. 지역 트랜잭션들이 서버로 제출되면 지역/전역 트랜잭션 충돌 그래프에 노드를 생성하고, 다른 트랜잭션들과 충돌 관계를 조사하여 직접 충돌 관계가 존재하면 선분을 생성한 다음 즉시 지역 트랜잭션 관리자에게 제출한다. 부트랜잭션에 대해서는 그 부트랜잭션이 수행을 시작할 때 노드를 생성하고, 다른 트랜잭션들과 충돌 관계를 조사하여 직접 충돌 관계가 존재하면 선분을 생성한다. 지역/전역 트랜잭션 충돌 그래프에서 노드를 제거하는 시점은 지역 트랜잭션은 완료되었을 때이며, 전역 트랜잭션들은 철회되거나 또는 완료되고 그 그래프에서 지역 트랜잭션들과 충돌 관계가 존재하지 않을 때이다.

정의 3.2 : 가상연산(Dummy-operations)

가상연산은 R(x) W(x)로 구성된다.
여기서, x : 부트랜잭션들이 접근하는 지역 데이타베이스의 자료 항목
R : 판독(Read) 연산
W : 갱신(Write) 연산

지역 트랜잭션으로 인하여 부트랜잭션들 사이에 간접 충돌 관계가 존재할 때 지역 사이트에서 부트랜잭션들의 직렬화 순서와 수행 순서의 불일치 문제가 발생할 수 있기 때문에, 이를 해결하기 위하여 서버에서 가상연산을 사용하여 그 부트랜잭션들 사이에 직접 충돌을 발생시킨다. 서버에서는 지역 트랜잭션으로 인해 부트랜잭션들 사이에 간접 충돌 관계가 존재하고, 그 부트랜잭션들 사이에 직접 충돌 관계가 존재하지 않을 때에만 가상연산을 사용한다. 예제 3.1에 가상연산

사용의 예를 나타내었다. 만약 예제 3.1에서 서버가 G_2에 대해서 가상연산을 사용하려고 할 때, 수행을 마친 G_1의 연산이 하나도 존재하지 않는다면 서버는 가상연산의 사용을 수행을 마친 G_1의 연산이 존재할 때까지 연기한다. G_2가 완료준비 상태가 되려고 할 때까지도 수행을 마친 G_1의 연산이 존재하지 않는다면, 서버는 지역 데이타베이스 시스템에서 수행 중인 G_1의 연산이 사용하고 있는 자료 항목을 사용하여 연기했던 가상연산을 완료준비 연산 앞에 사용한다.

예제 3.1 : 가상연산의 사용

어느 한 지역 사이트에 자료 항목 {a, b, c}가 존재한다고 하자. 또한 G_1, G_2를 그 사이트에 제출된 부트랜잭션, L_3를 지역 트랜잭션이라고 하자.

$G_1 : R_{g1}(a)\ W_{g1}(b)$

$G_2 : W_{g2}(c)$

$L_3 : W_{l3}(a)\ R_{l3}(c)$

L_3와 G_1:$W_{g1}(b)$가 지역 데이타베이스 시스템에서 수행중이라고 가정하면 L_3:$W_{l3}(a)$와 G_1:$R_{g1}(a)$사이에 직접 충돌 관계가 있으므로 (그림 2)의 (1)과 같이 지역/전역 트랜잭션 충돌 그래프에 표현되어 있다. 서버가 G_2:$W_{g2}(c)$를 지역 트랜잭션 관리자에게 제출하려고 할 때, G_2:$W_{g2}(c)$와 L_3:$R_{l3}(c)$ 사이에 직접 충돌이 발생되어 그 관계가 (그림 2)의 (2)와 같이 지역/전역 트랜잭션 충돌 그래프에 표현되기 때문에, G_1과 G_2사이에 지역 트랜잭션 L_3로 인한 간접 충돌 관계가 존재한다는 것을 서버가 알 수 있다. 서버는 G_2와 간접 충돌 관계에 있는 G_1의 연산들 중에서 가장 최근에 수행을 끝마친 연산이 사용한 자료 항목(여기서는 a)을 선정하여 가상연산 $D_{g2}(a)$:$R_{g2}(a)\ W_{g2}(a)$를 G_2:$W_{g2}(c)$연산 앞에 삽입한다. 그래서 전역 트랜잭션 G_2의 연산들이 G_{g2}:$R_{g2}(a)\ W_{g2}(a)\ W_{g2}(c)$로 변경되어 G_1과 G_2사이에 자료 항목 a로 인하여 직접 충돌 관계가 (그림 2)의 (3)과 같이 생성된다. 그렇기 때문에, 지역 사이트에서 $W_{l3}(a)\ R_{g1}(a)\ R_{g2}(a)\ W_{g2}(a)\ W_{g2}(c)\ R_{l3}(c)$와 같은 지역 이력은 그 수행 순서가 $L_3 \rightarrow G_1 \rightarrow G_2 \rightarrow L_3$가 되기 때문에 제안된 알고리즘의 가정 (1)에 의해 지역 동시성 제어 알고리즘이 생성하지 않는다. 그래서 지역 트랜잭션으로 인해 발생되는 부트랜잭션들의 직렬화 순서와 수행 순서의 불일치 문제를 해결할 수 있다.

(1) $G_1\{a, b\} - L_3\{a, c\}$

(2) $G_1\{a, b\} - L_3\{a, c\} - G_2\{c\}$

(3) $G_1\{a, b\} - L_3\{a, c\} - G_2\{a, c\}$

(그림 2) 지역/전역 트랜잭션 충돌 그래프의 표현
(Fig. 2) The representations of LGCG

서버에서 지역 트랜잭션으로 인하여 간접 충돌이 발생한 부트랜잭션들에게 가상연산을 사용하면 정리 3.1에 의해 지역 데이타베이스 시스템에서 부트랜잭션들의 수행 순서와 직렬화 순서는 동일하다.

정리 3.1 : 가상연산을 사용하면 지역 데이타베이스 시스템에서 부트랜잭션들의 수행 순서와 직렬화 순서는 동일하다.

증 명 : 본 논문에서 지역 스케줄러는 직렬 가능한 이력을 생성한다고 가정한 바 있다.

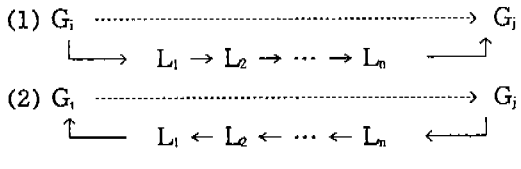
(1) 부트랜잭션들 사이에 아무런 충돌 관계가 존재하지 않는 경우에는 전역적 직렬성에 아무런 영향을 주지 않는다는 것은 당연하다.

(2) 부트랜잭션들 사이에 직접 충돌 관계가 존재하는 경우에는 부트랜잭션들의 수행 순서와 직렬화 순서는 동일하다. 왜냐하면 가정에 의해 지역 스케줄러가 직렬 가능하지 않는 이력은 생성하지 않기 때문이다.

(3) 부트랜잭션들 사이에 지역 트랜잭션들로 인하여 간접 충돌 관계가 존재하는 경우에는, 충돌이 발생된 부트랜잭션의 연산들 중에서 이미 수행을 완료한 자료 항목으로 서버에서 가상연산을 사용하기 때문에 부트랜잭션들 사이에 직접 충돌이 발생되어 (2)의 경우가 된다. 그래서 가정에 의해 지역 스케줄러는 (그림 3)의 (1)과 같

은 이력은 생성하지만 (2)와 같은 이력은 생성하지 않는다.

그러므로 가상연산을 사용하면 지역 데이타베이스 시스템에서 부트랜잭션들의 수행 순서와 직렬화 순서는 동일하다.



트랜잭션 충돌
가상연산에 의한 충돌

(그림 3) 가상연산의 효과
(Fig. 3) The effects of the dummy-operations

본 알고리즘에서는 충돌이 발생한 전역 트랜잭션들에 대해서만 전역적 직렬성을 검사하기 위하여, 서버에서 이미 수행을 마친 부트랜잭션들의 연산들에 대해서 충돌 여부를 조사하여 충돌이 발생된 부트랜잭션들을 직렬화 순서 그래프에 표현한다. 또한 부트랜잭션들의 처리 결과를 전역 트랜잭션 관리자에게 송신할 때 직렬화 순서 그래프에 표현되어 있는 부트랜잭션들의 직렬화 순서(선행, 후행 부트랜잭션들의 순서)를 함께 송신한다. 이 직렬화 순서는 전역 모듈에서 전역 트랜잭션들의 전역적 직렬성을 검사할 때 사용된다. 직렬화 순서 그래프에서 노드의 생성은 부트랜잭션들이 수행을 시작할 때이며, 노드의 제거는 부트랜잭션들이 전역적으로 철회되거나 부트랜잭션들이 완료되고 지역/전역 트랜잭션 충돌 그래프에서 지역 트랜잭션들과 충돌 관계가 존재하지 않으면서 그 부트랜잭션보다 직렬화 순서가 빠른 부토랜잭션들이 존재하지 않을 때이다.

정의 3.3 : 직렬화 순서 그래프(Serialization Order Graph : SOG)

SOG = (V, E)은 방향 그래프이다.
V : 부트랜잭션들의 집합
E : 충돌이 발생된 부트랜잭션들 사이의 직렬화 순서를 나타내는 선분들의 집합으로 T_i가 T_j보다 직렬화 순서가 빠르다면 $T_i \dashrightarrow T_j$로 표현
$T_i, T_j \in V$

3.3 전역적 직렬성 검사

전역 트랜잭션들 사이에 어떠한 충돌 관계도 존재하지 않는다면 전역적 직렬성은 항상 만족한다. 그래서 본 알고리즘에서는 전역적 직렬성 검사를 완료준비 상태(또는 가상의 완료준비 상태)에 있는 전역 트랜잭션들 중에서 그들의 부트랜잭션들 사이에 충돌 관계가 존재하는 전역 트랜잭션들에 대해서만 수행한다. 부트랜잭션들 사이에 충돌 발생 여부는 서버에서 유지되는 직렬화 순서 그래프에 표현되어 있다. 그래서 전역 모듈에서 전역적 직렬성을 검사할 때 서버로부터 수신된 부트랜잭션들의 직렬화 순서를 사용한다.

정의 3.4 : 정당성 검사 그래프(Validation Test Graph : VTG)

VTG = (V, E)은 방향 그래프이다.
V : 충돌이 발생된 전역 트랜잭션들의 집합
E : 전역 트랜잭션들의 직렬화 순서를 표현하는 선분들의 집합

전역 모듈에서는 전역 트랜잭션들의 전역적 직렬성을 검사하기 위하여 정당성 검사 그래프를 유지한다. 정당성 검사 그래프에서 노드의 생성은 각 서버로부터 처리 결과와 함께 수신되는 직렬화 순서가 존재할 때이고, 존재하지 않으면 생성하지 않는다. 전역 모듈에서는 각 서버로부터 수신된 직렬화 순서와 동일하게 정당성 검사 그래프에 선분을 생성한다. 선분이 생성된 다음 그래프가 비순환적이라면 그 전역 트랜잭션의 수행은 정당하다. 만약 그래프가 순환적이거나 전역 트랜잭션들이 지역 교착상태 등으로 인해 철회되었을 때에는 그 노드와 연결된 모든 선분들은 정

당성 검사 그래프에서 제거되고 그 노드도 제거된다. 어떤 전역 트랜잭션이 완료됐을 때 정당성 검사 그래프에서 그 노드의 제거는 그 노드보다 선행하는 노드(즉, 직렬화 순서가 빠른 전역 트랜잭션)들이 존재하지 않을 때이다.

낙관적 가상연산 방법은 정리 3.2에 의해 전역적 직렬성을 보장한다. 정리 3.1에 의해 각 지역 사이트에서 전역 부트랜잭션들의 수행 순서와 직렬화 순서가 동일하기 때문에 부트랜잭션들의 수행 순서로서 전역적 직렬성 검사를 정확하게 할 수 있다.

정리 3.2 : 낙관적 가상연산 방법은 전역적 직렬성을 보장한다.

증 명 : 정리 3.1에 의하여 전역 트랜잭션들의 수행 순서와 직렬화 순서는 동일하다. 증명을 위하여 직렬화 순서로 표현되는 정당성 검사 그래프가 어떠한 경우에도 순환이 발생되지 않음을 보이면 된다. 그래서 낙관적 가상연산 방법은 정당성 검사 그래프에 순환이 발생된다고 가정하자. 정당성 검사 그래프에서의 정점들은 수행중인 트랜잭션들과 완료된 트랜잭션들로 구성될 수 있다. 정리 3.1과 정의 3.3에 의하여 정당성 검사 그래프에서 이미 완료된 트랜잭션의 진입차수(indegree)는 완료 당시의 진입차수보다 증가되지 않는다. 또한 방향 그래프에서 진입차수가 0인 정점을 포함하는 순환은 존재하지 않는다. 그래서 진입차수가 0인 완료된 트랜잭션을 포함하는 순환은 정당성 검사 그래프에서 존재할 수 없기 때문에 그 정점을 제거할 수 있다. 또한 정당성 검사 그래프에서 순환을 발생시킨 트랜잭션은 전역적 직렬성 검사시 제거되기 때문에 정당성 검사 그래프는 항상 비순환적이다. 따라서, 가정에 모순이 되므로 낙관적 가상연산 방법은 전역적 직렬성을 보장한다.

멀티 데이타베이스 시스템에서 전역적 교착상태에 대한 연구는 앞으로 계속 연구되어져야 하는 분야이다[2, 6]. 본 논문에서는 전역적 교착상태를 해결하기 위해 타임아웃 방법이나 각 서버에서 대기 그래프(WFG)를 유지하므로써 해결할 수 있다. 그러나 효율적인 전역적 교착상태의 해결을 위해서는 계속 연구가 진행되어야 되는 부분이다.

4. 전역 회복 알고리즘

멀티 데이타베이스 시스템에서의 전역 회복 알고리즘은 전역 트랜잭션들의 원자성과 지속성의 성질을 보장하므로써 전역적 일관성을 유지시킬 수 있다. 멀티 데이타베이스 시스템에서 발생할 수 있는 고장들은 멀티 데이타베이스 시스템의 내부 구조등의 과괴로 발생될 수 있는 시스템 고장, 지역 사이트의 운영체제에 의한 서버 프로세스의 철회 등으로 발생될 수 있는 서버 고장, 통신 고장, 지역 사이트 고장, 트랜잭션 고장들이 존재한다[3].

제안하는 전역 회복 알고리즘은 다음과 같은 가정을 한다. (1) 예제 2.2와 같은 갱신 분실 문제가 발생되지 않도록 하기 위해서는 완료준비 상태에 있는 부트랜잭션이 완료(또는 철회)될 때까지 그 부트랜잭션과 충돌 관계가 있는 트랜잭션들은 완료될 수 없도록 해야 된다. 이를 위해서는 지역 동시성 제어 알고리즘들이 최소한 엄정한(strict) 이력을 생성해야만 하기 때문에[8] 본 알고리즘에서는 지역 동시성 제어 알고리즘들이 엄정한 이력을 생성한다. (2) 보상 방법은 2장 관련 연구에서 설명한 바와 같이 일반적인 전역 회복 기법이라 할 수 없으며, 재시도 방법은 재수행 방법보다 많은 비용이 들기 때문에 본 알고리즘에서는 재수행 방법을 전역 회복 기법으로 사용한다. (3) 지역/전역 트랜잭션 충돌 그래프에 재수행 트랜잭션들의 정보를 다른 트랜잭션들보다 먼저 표현하기 위하여 고장 회복 후 제출되는 트랜잭션들보다 재수행 트랜잭션들이 서버로 먼저 제출된다. (4) 전역 모듈과 지역 사이트 사이의 통신 회선 고장이나 통신 시스템 고장으로부터의 회복은 통신 시스템 자체의 문제로서, 이

로 인한 정보의 분실은 발생되지 않는다.

본 논문에서는 각 모듈마다 고장에 대비해 회복에 필요한 정보를 각각의 로그(log)에 기록한다. 전역 모듈의 GLOG에는 전역 트랜잭션들의 상태(제출, 수행, 완료, 철회)를 기록하여 전역적 회복 단계에서 전역 트랜잭션들에 대한 완료 목록(list)과 철회 목록을 생성한다. 완료 목록에는 GLOG에 완료라고 기록된 전역 트랜잭션들이 포함되며, 철회 목록에는 GLOG에 완료가 기록되지 않은 전역 트랜잭션들로 구성된다. 서버의 LLOG에는 부트랜잭션들에 관한 정보만을 유지한다. 이 LLOG에서 유지되는 정보는 부트랜잭션들에 대한 시작, 갱신 연산과 자료 항목, 완료준비, 전역적 완료, 완료 또는 철회이다. 이 모듈에서는 회복 단계에서 부트랜잭션들에 대한 철회 목록을 생성하여 전역 모듈로 통보하고, 전역 모듈로부터 수신한 완료 목록과 LLOG를 이용하여 부트랜잭션들에 대해서 재수행 목록을 생성하여 다른 트랜잭션들과 공유하는 자료 항목없이 배타적으로 수행한다. 철회 목록에는 LLOG에 전역적 완료가 기록되지 않은 부트랜잭션들이 포함되며, 재수행 목록에는 전역적 완료는 기록되어 있으나 완료는 기록되어 있지 않는 부트랜잭션과, 전역 모듈로부터 통보받은 완료 목록에는 포함되어 있으나 완료가 기록되어 있지 않는 부트랜잭션들이 포함된다. 각 지역 데이타베이스 시스템은 LLOG를 기록, 관리하며 지역 데이타베이스의 일관성 유지를 책임진다.

전역 회복 관리자는 고장이 발생했을 때 다음 사항을 보장하여야 한다. (1) 만약 전역 트랜잭션이 전역적 결정(완료나 철회) 상태에 도달되지 않고 고장이 발생한 사이트에서 수행 중인 부트랜잭션들이 최소한 하나 이상 존재한다면, 그 전역 트랜잭션을 철회하도록 한다. (2) 만약 전역적 결정이 되었다면, 어떠한 고장이 발생하더라도 전역적 일관성을 파괴하지 않으면서 모든 부트랜잭션들이 동일한 결정 상태가 되도록 한다. (1)의 사항은 원자적 완료 프로토콜에 의해 어렵지 않게 달성될 수 있기 때문에 제안된 전역 회복 알고리즘에서는 (2)의 사항에 중점을 둔다.

각종 고장으로부터 멀티 데이타베이스를 회복할 때에는 2장 관련 연구에서 살펴본 바와 같이 3가지 문제가 존재한다. 제안된 전역 회복 알고리즘은 다음과 같이 그 문제들을 해결한다. (1) 갱신 분실 문제 1(예제 2.2 참조)는 전역 트랜잭션 G_1이 완료되기 전에 지역 트랜잭션 L_{i2}를 수행하여 완료하였기 때문에 발생된다. 그래서 멀티 데이타베이스를 회복할 수 있는 전역 동시성 제어 알고리즘이 되기 위해서는 지역 동시성 제어 알고리즘들이 최소한 엄정한 지역 이력을 생성하여야만 한다. 지역 동시성 제어 알고리즘이 엄정한 이력을 생성한다면, 사이트 S_2에서 전역 트랜잭션 G_1:$R_{g1}(b)$ $W_{g1}(b)$가 완료되지 않았기 때문에 지역 트랜잭션 L_{i2}:$W_{i2}(b)$가 완료될 수 없으므로 LH_2:$R_{g1}(b)$ $W_{g1}(b)$ PC_{g1} $W_{i2}(b)$ C_{i2} Failure A_{g1} C_{g3} $R_{g3}(b)$ $W_{g3}(b)$과 같은 이력은 생성되지 않는다. 제안된 알고리즘에서는 전역 회복 알고리즘의 가정 (1)에 의해 이 문제를 해결한다. (2) 갱신 분실 문제 2(예제 2.3 참조)는 지역 사이트의 고장 회복 후 재수행되는 전역 트랜잭션 G_3:$W_{g3}(a)$의 수행보다 재수행 트랜잭션과 충돌 관계가 있는 새로운 트랜잭션 L_{i2}:$R_{i2}(a)$ $W_{i2}(a)$를 먼저 수행하므로써 발생되는 문제이다. 제안된 알고리즘은 서버에서 트랜잭션들 사이의 충돌 관계가 표현되는 지역/전역 트랜잭션 충돌 그래프를 유지하고 있다. 또한 전역 회복 알고리즘의 가정 (3)에 의해서 지역 사이트가 고장을 회복한 후 재수행되는 부트랜잭션들과 충돌이 발생된 트랜잭션들을 서버에서 알 수 있다((그림 2) 참조). 그래서 서버에서 재수행 트랜잭션 G_3:$W_{g3}(a)$와 충돌 관계가 있는 새로운 트랜잭션 L_{i2}:$R_{i2}(a)$ $W_{i2}(a)$의 수행을 충돌 관계가 없을 때까지(즉, 재수행 트랜잭션 G_3:$W_{g3}(a)$가 완료될 때까지) 지연시킴으로써 예제 2.3과 같은 갱신 분실 문제 2를 해결할 수 있다. (3) 전역 직렬성 위배 문제(예제 2.4 참조)는 지역 사이트가 완료준비 상태를 지원하지 않기 때문에 전역적으로 완료된 트랜잭션들이 그 전역적 결정을 지역 사이트에서 성공

적으로 수행하기 전에 지역 동시성 제어 알고리즘에 의해 철회되고, 전역적으로 완료된 부트랜잭션과 충돌 관계가 있는 트랜잭션들이 수행되기 때문에 발생되는 문제이다. 그래서 전역적 결정을 기다리는(즉, 가상의 완료준비 상태에 있는) 부트랜잭션과 충돌 관계가 있는 트랜잭션들의 수행을 충돌 관계가 없을 때까지 지연시킴으로써 이 문제를 해결할 수 있다. 제안된 알고리즘에서는, 사이트 S_1의 지역 동시성 제어 알고리즘에 의해 철회된 부트랜잭션 G_1:W_{g1}(a)은 전역적으로 철회되지 않았기 때문에 서버의 지역/전역 트랜잭션 충돌 그래프에 그 트랜잭션에 대한 정보가 표현되어 있다. 그래서 서버에서 가상의 완료준비 상태에 있는 부트랜잭션 G_1:W_{g1}(a)에 대한 전역적 결정을 사이트 S_1에서 성공적으로 수행될 때까지 그 트랜잭션과 충돌 관계가 있는 트랜잭션 L_3:R_{l3}(a) R_{l3}(b)의 수행을 지연시킴으로써 예제 2.4와 같은 전역적 직렬성 위배 문제를 해결할 수 있다. 제안된 전역 회복 알고리즘은 정리 4.1에 의하여 전역적 일관성을 유지한다.

정리 4.1 : 제안된 전역 회복 알고리즘은 지역 트랜잭션들과 재수행 트랜잭션들의 동시 수행으로 인하여 전역적 일관성을 파괴하지 않고 각종 고장으로부터 회복할 수 있다.

증 명 : 지역 트랜잭션들과 부트랜잭션들은 그들이 접근하는 자료 항목을 알 수 있는 서버를 경유하여 지역 트랜잭션 관리자로 제출된다. 재수행 트랜잭션들은 다른 트랜잭션들과 공유하는 자료 항목없이 배타적으로 수행된다. 지역 트랜잭션들과 재수행 트랜잭션들의 동시 수행으로 전역적 일관성이 파괴된다고 가정하자. 전역적 일관성은 고장이 발생된 사이트에서 지역 트랜잭션들이 재수행 트랜잭션들에게 영향을 주었을 때 파괴될 수 있다. 이는 지역 트랜잭션들과 재수행 트랜잭션들 사이에 충돌 관계가 존재하면서 동시에 수행된다는 것을 의미한다. 그러나 이것은 재수행 트랜잭션들이 다른 트랜잭션들과 공유하는 자료 항목없이 배타적으로 수행된다는 것에 모순된다. 그러므로 제안된 전역 회복 알고리즘은 전역적 일관성을 보장한다.

5. 결 론

본 논문에서는 지역 자치성을 보장하면서 전역적 직렬성을 만족하는 전역 동시성 제어 알고리즘으로 낙관적 가상연산 방법과, 지역 자치성을 보장하고 각종 고장으로부터 전역적 일관성을 유지하는 전역 회복 알고리즘을 제안하였다. 낙관적 가상연산 방법은 전역 트랜잭션들의 불필요한 지연이나 철회를 야기시키지 않고, 지역 트랜잭션으로 인해 부트랜잭션들 사이에 간접 충돌이 발생되는 경우에만 가상연산을 사용하여 부트랜잭션들 사이에 직접 충돌을 발생시킴으로써 부트랜잭션들의 수행 순서와 직렬화 순서가 동일하게 되도록 한다. 또한 전역 모듈에서는 부트랜잭션들 사이에 충돌 관계가 있는 전역 트랜잭션들만 전역적 직렬성을 검사하므로써 완료준비 단계에 있는 전역 트랜잭션들의 철회를 감소시킬 수 있다.

멀티 데이타베이스 시스템이 각종 고장으로부터 전역적 일관성을 유지하기 위해서는 각 지역 동시성 제어 알고리즘들이 최소한 엄정한(strict) 지역 이력(history)을 생성하여야 한다. 제안된 전역 회복 알고리즘은 지역 사이트가 완료준비 상태의 지원 유무에 관계없이 각종 고장으로부터 전역적 일관성을 유지할 수 있다.

앞으로는 효율적인 전역 교착상태의 해결 방안과 모의실험을 통한 알고리즘의 성능 분석 및 평가에 대한 연구가 진행되어야 한다. 그리고 전역 모듈이 분산되어 있는 환경에서 전역 데이타베이스의 일관성을 효율적으로 유지할 수 있는 방법에 대해서도 계속 연구가 진행되어야 한다.

참 고 문 헌

[1] Yuri Breitbart, Avi Silberschatz and Glenn Thompson, "Transaction Management in a Multidatabase Environment", Integration

of Information Systems : Bridging Heterogeneous Databases, ed. A. Gupta, IEEE Press, 1989.

[2] Yuri Breitbart, Avi Silberschatz and Glenn Thompson, "Reliable Transaction Management in a Multidatabase System", Proceedings of the ACM SIGMOD, pp. 215-224, May 1990.

[3] Y. Breitbart, A. Silberschatz and G.R. Thompson, "An Approach to Recovery Management in a Multidatabase Systems", VLDB Journal, Vol. 1, No. 1, pp. 1-39, 1992.

[4] Weimin Du, Ahmed K. Elmagarmid and Won Kim, "Maintaining Quasi Serializability in Multidatabase Systems", Proceeding Seventh International Conference on Data Engineering, pp. 360-367, April 1991.

[5] Ahmed K. Elmagarmid and A. A. Hela, "Supporting Updates in Heterogeneous Distributed Database Systems", Fourth International Conference on Data Engineering, pp. 564-569, Feb 1988.

[6] Dimitrios Geogakopoulos, Marek Rusinkiewicz and Amit Sheth, "On Serializability of Multidatabase Transactions Through Force Local Conflicts", Proceedings Seventh International Conference on Data Engineering, pp. 314-324, April 1991.

[7] Dimitrios Geogakopoulos, "Multidatabase Recoverability and Recovery", IEEE, pp. 348-355, 1991.

[8] Buhyun Hwang, "Three-level Transaction Scheduling in Multidatabase Systems", Ph.D. Thesis, KAIST, 1994.

[9] Songchun Moon and Jong T. Lim, "A Checkpointing Scheme for Heterogeneous Database Systems", 11th International Conference on Distributed Computing Systems, Arlington, Texas, U.S.A., pp. 608-615, May 1991.

[10] Hyunyoon Yun and Buhyun Hwang, "A Pessimistic Concurrency Control Algorithm In Multidatabase Systems", DASFAA 93, pp. 379-386, April 1993.

신 성 철

1983년 전남대학교 계산통계학과 졸업(학사).

1988년 전남대학교 대학원 전산통계학과(이학석사)

1989년~92년 송원실업전문대학 전산과 전임강사

1990년~현재 전남대학교 대학원 전산통계학과 박사과정 재학

관심분야 : 분산 시스템, 컴퓨터 네트워크

황 부 현

1978년 숭실대학교 전산학과 졸업(학사)

1980년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1980년~현재 전남대학교 전산학과 교수

관심분야 : 분산 시스템, 분산 데이타베이스 보안, 객체지향 시스템