이질형 통합 데이타베이스 시스템의 전역 트랜잭션을 위한 병행수행 제어 기법

이 규 웅†

요 약

이질형 통합 데이타베이스 시스템은 비정형 데이타을 비롯하여 관계형 데이타베이스 시스템의 정형 데이타에 이르기 까지, 지역적으로 여러 곳에 산재해 있는 유용한 정보를 통합하여 일관된 인터페이스를 통한 접근을 제공하는 통합 데이타베이스 시스템이다. 통합될 데이타 자원들은 서로 다른 질의 처리 능력을 제공할 뿐 아니라, 지역 자치성(local autonomy) 요구사항 때문에 전역적 트랜잭션 처리를 위한 지역 시스템의 상호 협조 운영이 불가능하여, 전역적 직렬성(global serializability)을 만족하는 전역 트랜잭션 관리기의 설계가 어렵다. 본 논문에서는 이질형 통합 데이타베이스 시스템의 전역 트랜잭션 관리 문제점 중에서 가장 잘 알려진 간접충돌의 문제를 해결하기 위해, 전역 무결성 제약사항의 특징을 이용한 전역 트랜잭션 모델을 정의한다. 전역 트랜잭션 모델을 기반으로 지역-로킹 연산과 이에 따른 프로토콜을 제안하고, 제안된 프로토콜이 전역 직렬성을 보장함을 증명한다. 또한 본 논문에서 제안한 지역-로킹 병행수행 제어 방법은 보다 현실적인 간접충돌의 범위를 정의함으로써, 기존 제안된 방법보다 높은 병행성 정도(concurrency degree)를 보장함을 지역 접근에 대한 경쟁률 분석을 통해 살펴본다.

Concurrency Control for Global Transaction Management in Integrated Heterogeneous Database Systems

Kyu Woong Lee†

ABSTRACT

Integrated heterogeneous database systems provide the unified interface for users and applications today in order to access the underlying diverse data sources located in different sites. The multiple heterogeneous data sources have the different and specialized data structures and transaction processing capabilities. Because of local autonomy, the local system does not have the capability of cooperation to control the global transaction. Hence designing the global transaction manager with supporting the global serializability is a difficult task. To resolve the well-known *indirect conflict*, we define the global transaction model by using the characteristics of global integrity constraints. And then we propose the site-locking operation and its protocol to manage the global transaction. The correctness and analysis of our site-locking protocol is proved and performance gain over the related other methods is also estimated in this paper.

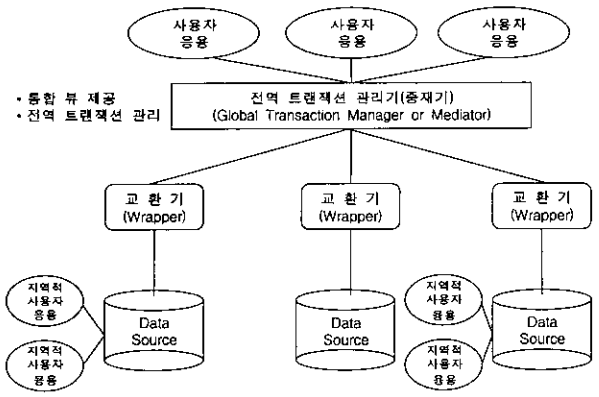
키워드 : 전역 직렬성(global serializability), 전역/지역 트랜잭션(global/local transaction), 병행수행 제어(concurrency control), 간접충돌(indirect conflict)

1. 서 론

최근 웹 문서 검색 등을 지원하기 위한 다양한 데이타 자원(data sources)이 발생하면서, 정보 통합에 대한 많은 기술 개발이 진행되고 있다. 이질형 통합 데이타베이스 시스템은 화일 시스템, 스프레드쉬트 데이타와 같은 비정형 데이타을 비롯하여 관계형 데이타베이스 시스템의 정형 데이타에 이르기 까지, 여러 곳에 산재해 있는 유용한 정보를 통합하여 일관된 인터페이스를 통한 접근을 제공하는 이질형 분산 데이타베이스 시스템의 한 유형이다[18-20]. 통합될 데이타 자원들은 서로 다른 질의 처리 능력을 제공할 뿐 아니라, 전역적 트랜잭션 처리를 위한 요소 기술이 부족하여 다양한 데이타 자원들을 통합 운영하기 위해 (그림 1)과 같은 일반적인 데이타 통합 시스템 구조를 사용한다[18, 19].

다양한 이질형 데이타를 통합한 시스템은 사용자에게 하나의 질의어를 통해 검색할 수 있게 하여야 하며, 동시에 일관성이 보장된 데이타를 검색할 수 있게 해야 한다. 이와 같은 기능은 전역 트랜잭션 관리기 또는 중재기(mediator)와 교환기(wrapper)를 통해서 이루어 진다[18, 19]. 전역 트랜잭션 관리기는 다양한 응용에서 발생하는 질의를 분해하여 교환기에서 지원하는 새로운 공통 질의어로 분할, 재구성한다. 교환기는 전역 트랜잭션 관리기로부터 요청된 부질

† 정 회 원 : 상지대학교 컴퓨터.정보공학부 교수
논문접수 : 2001년 1월 26일, 심사완료 : 2001년 7월 6일



(그림 1) 데이타 통합 시스템 공통 구조

의(sub-query)를 데이타 자원에 특정적인 질의와 명령으로 변화하여 해당 데이타 자원에서 수행할 수 있게 한다. 데이타 자원은 일반적으로 데이타베이스 시스템과 같은 데이타 관리를 위한 전용 시스템이거나 화일 시스템이다.

또한 각 데이타 자원은 기존에 사용되던 지역적 응용과 함께 통합되었으므로, 이질형 통합 데이타 관리 시스템에 운영되는 트랜잭션은 크게 전역 트랜잭션과 지역 트랜잭션으로 구분될 수 있다. (그림 1)에서 알 수 있듯이, 지역 트랜잭션은 하나의 데이타 자원에서 제공하는 지역 스키마 내의 데이타 만을 접근하는 트랜잭션이며, 전역 트랜잭션은 전역 트랜잭션 관리기를 통해 검색 또는 변경하고자 하는 데이타를 각 지역 데이타 소스에 요청하여 그 결과를 취합하여 제공하는 트랜잭션이다.

이질형 통합 데이타베이스 시스템의 전역 트랜잭션 관리기는 전체적인 데이타베이스 상태가 항상 일관성이 보장되도록 전역 트랜잭션을 제어해야 한다. 한편 지역 트랜잭션은 각 데이타 자원을 관리하는 지역 데이타 관리 시스템에서 일관된 수행이 되도록 제어 받는다. 그러나 각 지역 데이타베이스 시스템은 트랜잭션의 수행 제어 정보를 전역 트랜잭션 관리기에게 전달할 수 없다[13, 11, 21]. 그 이유는 이질형 데이타베이스 통합의 근본적인 목적이 데이타 자원, 데이타베이스 관리 시스템, 지역적 응용들에 대한 어떠한 수정도 없이 교환기, 전역 트랜잭션 관리기, 전역 응용을 그 위에 추가 탑재함으로써 전역적인 응용을 실행할 수 있게 하는 것이기 때문이다. 그러므로, 전역 트랜잭션 관리기는 지역 트랜잭션과 전역 트랜잭션의 수행이 공존하는 환경에서 지역 트랜잭션에 대한 정보 없이 오직 전역 트랜잭션들 만을 제어함으로써 전역 트랜잭션들 간의 직렬화 순서(serialization order)가 유지되는 일관된 수행임을 보장해야 하는 어려움이 있다.

간접충돌(indirect conflict)에 의한 전역 일관성의 위반은 전역 트랜잭션 관리에서 가장 잘 알려진 문제이다. 예를 들어, 지역 X에서 서로 데이타 충돌(data conflict)이 발생하지 않은 두 개의 전역 트랜잭션 G_{ix} 와 G_{jx}가 지역 트랜잭션 L_i를 사이에 두고 각각 직접적인 데이타 충돌(direct conflict)을 갖는 경우 두 전역 트랜잭션은 $G_{ix} \rightarrow L_i \rightarrow G_{jx}$ 또는 $G_{jx} \rightarrow L_i \rightarrow G_{ix}$ 의 직렬화 순서(serialization order)가 형성되게 된다. 그러나 전역 트랜잭션 관리기는 지역 트랜잭션 L_i의 수행을 알 수 없으므로 이와 같은 간접적인 충돌에 의한 직렬화 순서를 탐지 할 수 없는 문제점을 갖게 된다. 결국, 전역 트랜잭션 관리기는 일부 트랜잭션, 즉 지역 트랜잭션의 수행을 모른 채 직렬성을 보장해야 하는 어려움을 갖고 있다.

본 논문에서는 전역 무결성 제약사항의 특징을 이용하여 높은 병행성 정도(concurrency degree)를 제공하는 전역 트랜잭션 관리 기법을 제안한다. 이를 위해 전역 무결성 제약사항의 준수를 위한 전역 트랜잭션 모델을 제시하고, 이에 근거한 "지역 충돌" 프로토콜과 그 정확성을 증명한다. 본 논문의 구성은 다음과 같다. 2절에서는 기존 방법의 문제점과 연구 동기를 기술하고, 3절에서는 전역 트랜잭션 모델과 "지역 충돌" 프로토콜을 제안한다. 4절에서는 제안한 프로토콜의 정확성을 증명하고, 타 방법과의 지역 충돌 정도에 대한 비교 분석을 한다. 끝으로 5절에서 본 논문의 결론을 맺는다.

2. 관련연구 및 문제점

이질형 통합 데이타베이스 시스템에서 전역 직렬성(global serializability)을 보장하는 전역 트랜잭션 관리기법에 관한 연구가 많이 제안되었다[5, 8, 10, 11, 14]. 기존 연구들은 다음과 같은 세 분류로 나눌 수 있다.

- 지역 자치성(local autonomy)의 위반

이 분류의 방법들은 지역 트랜잭션을 제어하기 위해서 어느 정도의 지역 자치성을 위반한다[1, 9]. 즉, 전역 트랜잭션 관리기에 지역 트랜잭션의 수행 정보를 알려 주기 위해 지역 시스템의 수정이 필요한 경우이다. 또한 이 방법들은 전역 트랜잭션에 강한 제약을 두고 있으며, 참고 논문 [8]에서는 전역 직렬성이 위반됨을 증명하고 있다.

- 직렬성 기준의 완화

이 분류의 방법에서는 전역 병행수행 제어 기법의 지역적 접근과 전역적 접근이라는 계층적 특성을 이용하여 일반 데이타베이스 시스템의 기준으로 사용하고 있는 충돌 직렬성(conflict serializability : CSR)보다 완화된 "quasi" 직렬성[6, 7]과 이단계(two-level) 직렬성[14, 15]을 제안하고 있다. 이들 방법의 단점은 전역 트랜잭션들간의 종속성이 없음을 가정하거나 완화된 종속성을 가정하여, 현실적으로 응용되기 어렵다는 점이다.

- 지역 시스템의 제약

이 분류의 방법들은 지역 데이타 자원을 관리하는 시스템

에서 수행되는 트랜잭션에 약간의 제약을 두는 기법을 사용한다. 참고문헌 [17]은 각 지역 데이타베이스 시스템들이 모두 엄격한 이단계 잠금 방법(Strict 2 Phase Locking)을 사용하는 "2PC Agent" 방법을 제안하였다. 그러나 모든 지역 시스템이 생성한 스케쥴 즉, 지역 트랜잭션의 수행 순서가 "strictness"를 만족한다 해도 전역 직렬성을 만족할 수 는 없다. 또한 접근하는 모든 지역을 배타적으로 접근하도록 하는 사이트 로킹 방법 또한 전역 직렬성을 위반할 수 있음을 참고문헌[22]에서 보이고 있다. 각 지역 시스템에서 수행되는 트랜잭션의 순서가 "rigorousness"를 만족해야 하는 방법을 참고문헌 [3]에서 제안하였다. "rigorousness"는 "strictness"의 모든 성질을 만족하고, 추가적으로 이미 판독(read)된 데이타에 대해서 그 트랜잭션이 완료되거나 취소될 때 까지, 다른 트랜잭션이 그 데이타 항목에 대해 기록할 수 없다는 성질을 추가적으로 갖는 개념이다. 따라서 이 개념을 만족하는 스케쥴의 범위는 기존 일반 병행수행 제어 기법에서 제공하는 스케쥴의 범위보다 더 작기 때문에 이 개념을 만족하는 병행수행 제어기는 낮은 병행성 정도를 제공하게 된다. 또한, "rigorousness"를 만족하기 위해서 전역 트랜잭션이 접근해야 하는 데이타 집합에 제약을 두고 있으나 이는 데이타 통합이라는 근본적인 목적에 위반되는 가정이다. 참고문헌 [10, 11]에서는 티켓이라 불리우는 추가적인 데이타를 사용하여 전역 직렬성을 보장하는 낙관적 티켓 방법(optimistic ticket method : OTM)이라는 방법을 제안하고 있다. 이 방법은 각 지역 데이타 자원에 티켓이라는 추가적인 데이타를 두어 그 지역을 접근하는 모든 전역 트랜잭션이 강제적으로 접근하게 하여 전역 트랜잭션들간의 직렬화 순서를 탐지하는 방법이다. 극단적인 경우, 두 전역 트랜잭션이 전혀 데이타를 공유하지 않고, 또한 지역 트랜잭션에 의해서도 간접충돌이 발생하지 않는 경우에도 두 전역 트랜잭션은 강제적으로 티켓이라는 데이타를 접근해야 하므로 데이타 충돌(data conflict)이 발생하게 되고, 이로 인해 전역 트랜잭션들간의 직렬성을 보장할 수 있게 하는 방법이다. 이 방법은 전역 직렬성을 보장하기 위한 가장 간단한 방법이면서도, 한편 전역 트랜잭션들간의 높은 병행성 정도를 제공하지 못하는 단점을 가지고 있다. 본 논문에서는 병행수행 정도를 저하시키는 주요 요소인 강제적 데이타 충돌을 방지하는 새로운 전역 트랜잭션 병행수행 기법을 제시한다.

<표 1>은 위에서 나열한 전역 트랜잭션 관리를 위한 기존 방법들에 대한 정리 및 요약을 나타내고 있다. <표 1>에서 [3, 4, 5, 10, 11]는 전역 직렬성을 보장하고 있으며, [6, 7, 14, 15]에서는 완화된 직렬성을 보장한다. 제안된 대부분의 방법에서 전역 트랜잭션에 수행이나 데이타 접근에 대한 일정한 제한을 두고 있음을 알 수 있다.

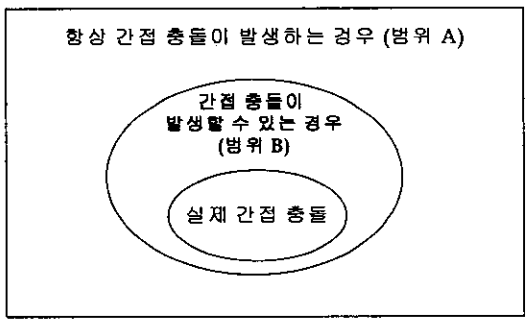
〈표 1〉 이질형 통합 데이타베이스 시스템에서의 병행수행 제어 기법들

관련 연구	지역 자치성	트랜잭션 수행 기준	전 역 직렬성	전역 트랜잭션의 제한	전 역 무결성 제약사항
global altruistic 2PL [1], [9]	위반	CSR	위반	제한됨	고려 안함
quasi serializability [6,7]		QSR	보통	제한됨	
2 Level serializability [14, 15]		2 Level SR	보통	제한됨	
rigorous scheduling, reliable management, 2PC Agent Method [3], [4, 5], [17]		CSR		제한됨	고려 안함
Ticket Method [10, 11]		CSR		없음	고려 안함

<표 1>에 기술된 전역 트랜잭션 관리 방법들의 문제점은 다음과 같이 요약될 수 있다.

- 지역 트랜잭션에 의한 간접충돌의 상황을 정확히 탐지할 수 없다. 일반적으로 두 전역 트랜잭션이 같은 지역을 접근하게 되는 경우 지역 트랜잭션에 의한 데이타 충돌을 파악하지 못하므로, 그 들 사이에 데이타 충돌을 강제적으로 유도하거나, 데이타 충돌이 있다는 가정 하에 병행수행 제어를 하게 된다.
- 기존 전역 트랜잭션 관리기에서 제공하는 병행성 정도는 순차적인 순서와 유사한 병행성 정도를 제공하므로 상당히 낮은 병행성 정도를 제공한다.
- 전역 트랜잭션의 간접충돌 상황을 정확히 판단하지 못하므로, 전역 트랜잭션들간의 간접충돌 발생 가능한 상황을 너무 넓고 포괄적으로 정의하여, 실제 충돌을 야기하지 않는 전역 트랜잭션들 까지도 병행수행을 할 수 없게 한다.

기존 연구들에서는 간접충돌이 발생하는 상황의 범위를 정확히 규명하지 못하여 (그림 2)의 범위 A와 같이 임의의 전역 트랜잭션 간에 항상 간접충돌이 있다는 가정하에 병행수행 제어방법을 제안하였다. 본 연구에서는 전역 무결성 제약사항의 특징을 이용하여, (그림 2)의 범위 B와 같이 실제



(그림 2) 간접충돌의 범위 정의

간접충돌 상황과 더욱 근사한 범위를 정의하여 A-B 만큼의 병행수행 성능 향상을 얻을 수 있는 전역 트랜잭션 관리 기법을 제안한다.

3. 전역 트랜잭션 관리를 위한 지역-로킹 병행수행 제어 기법

3.1 무결성 제약사항과 트랜잭션 모델

이질형 통합 데이타베이스 시스템 환경에서는 무결성 제약사항(integrity constraints : IC)을 지역적과 전역적 무결성 제약사항으로 나눌 수 있다. 지역적 무결성 제약사항(local IC)은 기존 지역적 응용에서 사용되던 데이타들을 위한 제약사항이며, 전역적 무결성 제약사항(global IC)은 데이타 자원들의 통합에 따른 추가적인 무결성 제약사항으로, 전역적 응용에 사용되는 데이타들을 위한 제약사항이다.

지역적 무결성 제약사항은 지역 데이타 자원을 관리하는 지역 시스템에서 유지될 수 있으며, 기존 상용 데이타베이스 시스템들은 이러한 제약사항의 준수를 보장한다. 그러나 서로 다른 데이타 자원상에 정의된 전역적 무결성 제약사항은 각 자원을 관리하는 데이타베이스 시스템에서 관리 및 검사될 수 없다. 예를 들어, 전역적 무결성 제약사항 "데이타 자원 X의 데이타 항목 a의 값은 데이타 자원 Y의 데이타 항목 b의 값보다 항상 커야 한다"에 정의된 데이타 항목 a의 값을 변경한다고 가정하자. 이 때, 데이타 자원 X를 관리하는 지역 데이타베이스 시스템에 의해 그 값이 변경된다면, 그 지역 데이타베이스 시스템 자체적으로는 데이타 자원 Y의 데이타 항목 b의 값을 알 수 없으므로, 데이타 항목 a의 무결성을 보장할 수 없게 된다.

데이타 자원간에 정의된 전역적 무결성 제약사항은 이질형 통합 데이타베이스 시스템의 데이타를 다음과 같이 분류할 수 있게 한다. 특정한 지역 i의 데이타 집합을 D_i라고 할 때 지역 데이타 집합 LD_i와 전역 데이타 집합 GD_i로 분류할 수 있으며, 이 때, $LD_i \cap GD_i = \emptyset$이고 $LD_i \cup GD_i = D_i$이다. 또한 두 데이타 항목 $d_i \in D_i$와 $d_j \in D_j$, $i \neq j$에 정의된 제약사항이 존재할 때, 데이타 항목 d_i와 d_j는 전역 데이타 항목 GD_i와 GD_j에 속하게 된다. 따라서 데이타 집합을 다음과 같이 분류할 수 있다[12].

- 전역 데이타 집합 : 전역적 무결성 제약사항에 정의된 데이타 집합
- 지역 데이타 집합 : 기존 사용되던 지역적 무결성 제약사항에 정의된 데이타 집합

전역적 무결성 제약사항을 모른 채, 지역 트랜잭션이 전역 데이타 집합의 값을 변경하면 전역 데이타베이스 상태(global database state)의 일관성이 위반될 수 있음은 명확한 일이다. 전역 트랜잭션 관리기를 통해 이루어진 전역 데이타 변경에 대한 무결성 제약사항 검사는 기존의 일반 데이타베이스 시스템의 방법으로 쉽게 검사될 수 있다. 그러나 전역 트랜잭션 관리기를 통해 발생되지 않은 지역 트랜잭션에 의한 전역 데이타의 변경은 비일관성을 야기하며, 이러한 전역 비일관성(global inconsistency)을 발생시키지 않기 위해서는 전역 트랜잭션 관리기가 지역 트랜잭션에 의한 전역 데이타의 변경에 대해서 계속 감시해야 한다. 이러한 작업은 매우 높은 오버헤드를 유발할 뿐만 아니라, 일시적인 비일관성을 야기할 수도 있게 된다. 따라서 지역 트랜잭션에 대한 데이타 접근 제한이 필수적이다. 즉, 지역 트랜잭션은 전역 무결성 제약사항을 모른 채 전역 데이타의 값을 갱신할 수 없다는 제약을 갖는다. 그러나 판독 데이타에 대해서는 지역 트랜잭션에 어떠한 제한도 두지 않는다. 또한 전역 트랜잭션은 데이타 접근에 아무런 제한도 받지 않는다.

기존의 몇몇 방법들에서는 지역 트랜잭션이 아닌 전역 트랜잭션의 데이타 접근에 제한을 두고 있을 뿐만 아니라, 병행수행 제어 기준으로 충돌 직렬성(conflict serializability)을 사용하지 않고 완화된 직렬성을 제안하여 사용하고 있다[6, 7, 13, 14]. 그러나 전역 트랜잭션이 데이타 접근에 제한을 갖는 다는 것은 데이타 자원을 통합한다는 기본적인 목적 자체에 위반된다. 즉, 여러 지역의 데이타를 접근하여 값을 판독하고, 갱신하고자 데이타를 통합하는 것인데, 그러한 응용이 수행에 제한을 받는 다는 것은 데이타 통합을 무의미하게 하는 가정이다. 따라서, 본 논문에서는 <표 2>와 같이 이질형 통합 데이타베이스 시스템을 위한 전역 트랜잭션 모델을 정의한다.

〈표 2〉 전역 트랜잭션 모델

트랜잭션 \ 데이타		데이타 항목	
		지역 데이타	전역 데이타
지역 트랜잭션	판독(read) 연산	O	O
	기록(wirte) 연산	O	X
전역 트랜잭션	기록(read) 연산	O	O
	기록(wirte) 연산	O	O

제안된 트랜잭션 모델에 따른 데이타의 분류는 전역 스키마(global schema)상에 정의되는 전역 제약사항을 이용하여 전역 및 지역 데이타로 분류하여 사용하게 된다. 제안된 트랜잭션 모델에 따라 지역 트랜잭션이 전역 데이타를 변경하는 것에 제약을 두게 되면, 기존 사용하던 지역 응용 프로그램의 사용이 제한을 받게 될 수 있다. 그러나 지역 응용 프로그램이 전역 트랜잭션 관리기 인터페이스를 통해서 전역 스키마상에서 수행된다면, 지역 응용 프로그램의 수정 또는 변경없이 그대로 이용할 수 있게 된다. 즉, 지역 응용 프로그램이 제약을 받게 되는 이유는 전역적 무결성 제약사항에 정의된 데이타를 전역 무결성 제약사항을 모른 채 변경되는 것을 방지하고자 하는 것이므로, 이러한 응용은 전역 트랜잭션 관리기 인터페이스를 통해 수행되어야 한다.

3.2 전역 직렬성의 보장

이질형 통합 데이타베이스 시스템 환경에서, 비록 지역 트랜잭션들의 스케쥴이 직렬화 가능한(serializable) 스케쥴이라 할지라도 전역 직렬성을 보장하기 어렵다는 것을 기존의 연구들에서 보여 왔다. 그 이유는 전역 트랜잭션 관리기가 지역 트랜잭션의 수행 속성, 즉 지역 트랜잭션이 접근한 데이타 항목, 지역 트랜잭션의 스케쥴에 대한 정보를 알 수가 없기 때문이다. 따라서, 지역 트랜잭션과 전역 트랜잭션간의 직접충돌 상황을 좀더 명확하게 판별할 필요가 있다. <표 3>은 지역 트랜잭션과 전역 트랜잭션의 직접충돌 발생 상황을 연산과 데이타 별로 분류하여 나타내고 있다.

〈표 3〉 지역 트랜잭션과 전역 트랜잭션의 직접충돌

O : 직접충돌 발생 안함
X : 직접충돌 발생

			전역 트랜잭션			
			판독(Read) 연산		기록(write) 연산	
			전역 데이타	지역 데이타	전역 데이타	지역 데이타
지역 트랜잭션	판독(Read) 연산	전역 데이타	O	O	X	O
		지역 데이타	O	O	O	X
	기록(Write) 연산	지역 데이타	O	X	O	X

<표 3>에서 "O"는 두 트랜잭션 사이에 서로 데이타 충돌이 없음을, "X"는 서로 데이타 충돌이 있을 수 있음을 나타내고 있다. 따라서 "X" 인 경우 전역 트랜잭션과 지역 트랜잭션이 직접충돌을 발생시키며, 이는 두개의 전역 트랜잭션 사이에 간접충돌을 유발할 수 있으므로, 두 전역 트랜잭션이 병행수행 할 수 없는 상황을 의미한다. 유사하게 "O" 인 경우, 전역 트랜잭션 사이에 간접충돌을 형성할 수 없음을 알 수 있다. 그러나 <표 3>에 나타난 분류는 지역 트랜잭션에 대한 정보를 알아야 하므로 전역 트랜잭션 관리에 직접적으로 이용할 수 없다. 따라서, <표 3>의 내용을 이용하여 지역 트랜잭션의 정보 없이 전역 트랜잭션들간의 간접 충돌 발생 가능성을 판별하여 전역 트랜잭션들간의 직렬성을 보장하는 "지역-로킹(site-locking)" 병행수행 제어기법을 제안한다.

3.3 지역 접근을 위한 로킹(locking) 연산

이 절에서는 전역 트랜잭션이 지역-로킹 연산을 통해 지역 접근의 허가를 획득하는 지역-로킹 병행수행 제어기법을 소개한다. 본 방법의 기본적인 아이디어는 간접충돌의 발생 가능성이 있는 전역 트랜잭션들은 그 지역에 대한 지역-로킹 연산에서 로킹을 획득할 수 없게 하여 그들 간의 배타적 수행을 보장하는 것이다. 프로토콜을 기술하기 전에 먼저 간접충돌 연산과 직접충돌 연산을 정의한다.

[정의 1] 간접충돌 연산(indirect conflicting operation)은 간접충돌을 유발하는 전역 트랜잭션의 연산이다. 즉, 전역 트랜잭션 G_i의 연산 $P_i(x)$와 전역 트랜잭션 G_j, $i \neq j$, 의 연산 $Q_j(y)$가 서로 간접충돌 관계에 있으면 두 연산 $P_i(x)$, $Q_j(y)$는 간접충돌 연산이다. 이 때 두 데이타 x와 y는 서로 다를 필요는 없다. 만약, 두 연산 $P_i(x)$, $Q_j(y)$에서 $x = y$ 이면, 두 연산 P, Q는 모두 판독 연산이어야 한다. 그렇지 않으면 두 연산은 직접충돌 연산(direct conflicting operation)이다.

간접충돌 연산들만 직렬화 가능한 스케쥴로 만들 수 있으면, 전역 직렬성은 보장된다. 전역 트랜잭션들 사이의 직접충돌은 기존의 병행수행 제어 방법으로도 충분히 직렬화 가능하게 할 수 있으므로, 간접충돌을 발생하는 연산에 대해서만 직렬화 가능하다면 전역 직렬성을 얻을 수 있게 된다. 제안하는 방법은 간접충돌 및 직접충돌 연산 모두에 대해서 기존의 이단계 로킹 방법 2PL처럼 지역-로킹 연산을 사용한다. 앞의 <표 3>에서 알 수 있듯이, 전역 트랜잭션의 전역 데이타 항목에 대한 판독 연산은 어떤 지역 트랜잭션과도 직접충돌을 야기하지 않으므로 간접충돌이 될 수 없다. 나머지 연산들은 간접충돌 연산이 될 수 있다. 따라서 <표 3>을 기준으로 "지역-로킹" 기법을 제안한다.

"지역-로킹"에서 로킹의 단위는 데이타가 아니라 지역이다. 전역 트랜잭션의 연산은 특정지역으로 보내지기 전에 판독 연산인지, 기록 연산인지가 구별되고, 전역 또는 지역 데이타 항목을 접근하는지 결정된다. 이와 관련하여 본 방법에서는 다음과 같은 세가지 로킹 연산을 사용한다.

- RGL : 전역 데이타 항목에 대한 판독 연산을 위한 로킹 연산
- RLL : 지역 데이타 항목에 대한 판독 연산을 위한 로킹 연산
- WL : 기록 연산을 위한 로킹 연산

판독 연산에 대해서는 접근하는 데이타 종류에 따라 간접충돌의 여부가 결정되므로 다른 로킹 연산을 사용하지만, 기록 연산에 대해서는 한 종류의 로킹 연산을 사용한다. $RGL_{Gi}(S_x)$와 $RLL_{Gi}(S_x)$는 전역 트랜잭션 G_i가 지역 S_x에 존재하는 전역 데이타 항목과 지역 데이타 항목에 대해 접근하기 위해 수행해야 하는 각각의 로킹연산을 의미한다. 유사하게 $WL_{Gi}(S_x)$는 지역 S_x를 기록 연산으로 접근하기 위해 수행해야 하는 로킹연산이다. 또한 $RGU_{Gi}(S_x)$, $RLU_{Gi}(S_x)$, $WU_{Gi}(S_x)$ 연산은 획득한 로킹을 해제하기 위해 수행하는 연산이다. 전역 트랜잭션은 특정지역에서 수행해야 할 연산들을 각각의 지역으로 보내기 전에 해당하는 로킹연산을 먼저 수행하여 해당 로킹을 획득한 후 보내져야 한다. 특별히 $RGL_{Gi}(S_x)$과 $WL_{Gi}(S_x)$ 로킹 연산인 경우에, 어떤 데이타 항목을 접근하는지에 대한 정보를 로킹 정보와 함께 유지해야 한다. 이는 로킹 연산 처리시 직접충돌의 여부를 결정하기 위해서 반드시 필요하다. 각각의 로킹에 대한 호환 성

질을 제안하기 위해 "충돌 유형"과 "지역 충돌"을 먼저 정의한다.

[정의 2] 로킹 연산 P, Q는 각각 지역 S_x를 접근하는 전역 트랜잭션 G_i와 G_j의 임의의 로킹연산이다. 이 때, 다음과 같은 조건을 만족하면 두 로킹연산은 서로 **충돌 유형**이라 한다.

i) P와 Q 두 연산중 한 연산이 WL 로킹 연산이고, 다른 나머지 하나가 RLL 또는 WL 로킹 연산인 경우

ii) P와 Q 두 연산중 한 연산이 WL 로킹 연산이고, 다른 나머지 하나가 RGL 로킹 연산이면서, 서로 같은 데이타 항목을 접근하는 경우

[정의 3] 두 로킹 연산 $P_{Gi}(S_x)$와 $Q_{Gj}(S_x)$에 대해 P와 Q가 서로 충돌 유형이고 $G_i \neq G_j$일 때 두 로킹 연산은 서로 **지역 충돌**이다.

정의 2와 정의 3에 의해 지역-로킹 호환표를 <표 4>와 같이 정의할 수 있다. 지역-로킹 호환표에 의한 로킹 연산을 통해, 지역 트랜잭션의 수행정보 없이 전역 트랜잭션간의 병행수행을 제어함으로써 전역 직렬성을 보장할 수 있게 된다. 전역 트랜잭션 사이에 간접충돌이 발생할 수 있거나 또는 직접충돌이 발생하는 경우 두 전역 트랜잭션의 로킹연산은 서로 지역충돌 모드이므로, 서로 배타적으로 수행해야 한다. 그렇지 않은 경우, 두 전역 트랜잭션은 전역 직렬성을 보장한 채 서로 병행수행 될 수 있다.

〈표 4〉 지역-로킹 호환표

Gj(요청자) \ Gi(소유자)		Read Lock		Write Lock
		RGL	RLL	WL
Read Lock	RGL	○	○	⊗
	RLL	○	X	X
Write Lock	WL	⊗	X	X

○ : 공유 모드 X : 지역 충돌 모드 ⊗ : 공유 또는 지역 충돌 모드

전역 트랜잭션은 <표 4>에 의해 정의된 호환표에 따라, 지역을 접근하기 전에 해당 로킹연산을 수행하여 해당 로크를 획득한 후 수행된다. 예를 들어, 지역 S_x의 지역 데이타 항목을 판독하기 위해서 그 지역의 RLL 로크를 획득해야 한다. 이 때, 이미 다른 전역 트랜잭션이 RLL 또는 WL 로크를 획득한 경우 다른 전역 트랜잭션의 로크가 해제 될 때 까지 기다려야 한다. 특별히, RGL 로크에 대해서는 다른 전역 트랜잭션이 같은 데이타 항목에 대해서 WL를 획득한 경우에만 지역 충돌 모드이다. 즉 두 전역 트랜잭션이 직접충돌인 경우이므로, 전역 트랜잭션간의 수행순서가 지역 시스템에 의해 바뀔 수 있는 경우를 배제하기 위해서이다.

3.4 지역-로킹 프로토콜

전역 직렬성을 보장하기 위해서는 전역 트랜잭션의 각 서브 트랜잭션들이 해당 지역 데이타베이스 시스템 내에서 상대적으로 같은 직렬화 순서를 보장해야 하며, 전역 트랜잭션 관리기 수준에서 결정된 직렬화 순서대로 지역 시스템 내에서의 수행이 보장되어야 한다. 본 논문에서 제안한 지역-로킹에 따른 전역 트랜잭션들은 지역에 대한 로킹을 획득한 순서로 각 지역에서 수행되므로, 로킹 획득 순서가 지역 수행순서를 결정한다.

지역-로킹 프로토콜에 따라 전역 트랜잭션 G_i의 전역 서브트랜잭션 G_{ix}가 지역 S_x로 보내지기 전에, 전역 트랜잭션 관리기에 해당 로크를 요청한다. 지역-로크가 획득되면, 전체 지역 내에서 트랜잭션이 완료될 때까지 해당 로크를 소유한다. 지역-로킹 프로토콜은 세가지 유형의 로킹 연산을 사용한다는 것과 로킹의 단위가 지역이라는 것을 제외하고는 기존의 2PL 방법과 유사하다. 전역 트랜잭션 관리기가 지역-로크를 관리하기 위한 규칙은 다음과 같다.

Rule 1 : 전역 트랜잭션 G_i의 서브트랜잭션 G_{ix}에 대하여, 해당 지역의 로크 유형을 결정하고, 해당 로크가 이미 로크를 소유한 전역 트랜잭션과 지역 충돌 모드인지 검사한다. 지역 충돌 모드이면, 기소유한 전역 트랜잭션이 로크를 해제할 때 까지 기다린다. 그렇지 않으면, 해당 로크를 획득하고 서브트랜잭션의 연산을 그 지역으로 보낸다.

Rule 2 : 전역 트랜잭션 G_i에 대해 지역-로크를 획득하면, G_i가 전역적으로 완료할 때 까지 그 지역-로크를 해제하지 않는다.

Rule 3 : 전역 트랜잭션 G_i에 대한 지역-로크를 한번 해제하고 나면, 추가적인 지역-로크를 획득할 수 없다.

Rule 1은 서로 지역 충돌 모드인 두 전역 트랜잭션이 서로 같은 지역에서 병행 수행되는 것을 방지하기 위한 규칙이다. 그러므로, 지역 충돌을 야기하는 두 전역 트랜잭션의 연산들은 로킹을 획득한 순서대로 그 지역에서 수행됨이 보장된다. 즉, 간접충돌을 유발하는 두 전역 트랜잭션간의 직렬화 순서를 전역 트랜잭션 관리기 수준에서 제어할 수 있게 되고, 직렬화 순서를 보장할 수 있다. Rule 2와 Rule 3은 기존 2PL의 획득 단계(growing phase)와 해제 단계(shrinking phase)에 관련한 규칙이다. 위 규칙에 따라 지역-로킹 프로토콜은 다음과 같이 요약될 수 있다.

전역 트랜잭션 분해 단계 : 전역 트랜잭션 G_i는 해당 지역으로 보내지기 위해 각각의 서브 트랜잭션 G_{i1} … G_{in}으로 분해된다.

지역-로크 요청 단계 : 해당 지역으로 보내지기 전에, 그 지역의 해당 로크를 요청한다.

서브 트랜잭션 수행 단계 : 서브 트랜잭션 G_{ix}가 해당 지역 로크를 획득하면 해당 지역으로 보내지게 되며, 그렇지 않으면 해당 지역-로크를 획득할 때 까지

기다린다.

지역 수행 단계 : 지역으로 보내어진 서브 트랜잭션 G_x은 지역 시스템의 제어하에 지역 트랜잭션과 같이 수행된다.

지역-로크 해제 단계 : 모든 서브 트랜잭션이 완료-준비 상태가 되면, 전역 트랜잭션 G_i는 전역적으로 완료하고 모든 해당 지역-로크를 해제한다.

[예제 1] 제안된 지역-로킹 프로토콜에 의해 전역 직렬성이 보장되는 전역 트랜잭션들의 수행 예를 보인다. 지역 데이타 항목 a를 갖고 있는 지역 S_x와 전역 데이타 항목 b, c를 갖고 있는 지역 S_y 가 있는 전역 데이타베이스를 가정하자. 두 개의 전역 트랜잭션 G_i, G_j와 전역 무결성 제약사항은 다음과 같다.

전역 트랜잭션 G_i : $R_{Gix}(a)$ $W_{Giy}(c)$
전역 트랜잭션 G_j : $W_{Gjx}(a)$ $R_{Gjy}(b)$
전역 무결성 제약사항 : (b > a)

또한 지역 S_y에서는 다음과 같은 지역 트랜잭션이 수행된다.

지역 트랜잭션 L_y : $R_{Ly}(b)$ $R_{Ly}(c)$

전역 트랜잭션 G_i는 지역 S_x와 지역 S_y에 대해 각각 RLL, WL 지역-로크를 획득해야 하며, 전역 트랜잭션 G_j는 각각 WL, RGL 지역 로크를 획득해야 한다. 따라서 전역 트랜잭션 관리기는 해당 로크를 관리하기 위해 (그림 3)과 같은 수행을 보인다. 지역 S_x에서, G_i의 연산 $R_{Gix}(a)$를 수행하기 위해 $RLL_{Gi}(S_x)$ 지역-로크를 획득하고, 연산을 수행한다. 또한 G_j의 연산 $W_{Gjx}(a)$을 수행하기 위해 $WL_{Gj}(S_x)$를 요청하나 이미 획득한 $RLL_{Gi}(S_x)$ 지역-로크와 지역 충돌을 야기하므로 기 획득된 지역-로크가 해제될 때 까지 기다린다. 한편 지역 S_y에서는 해당 지역-로크 $RGL_{Gj}(S_y)$, $WL_{Gi}(S_y)$가 서로 지역-충돌을 유발하지 않으므로 기다림 없이 모두 획득되어 순조롭게 수행된다. 그 후, G_i가 완료되고, 대기하던 G_j의 $WL_{Gj}(S_x)$ 지역-로크가 획득되어 G_j의 모든 연산도 각 지역에서 수행된다.

	1	2	3	4	5	6	7	8	9	10
GTM	RLL_{Gi} (Sx)	RGL_{Gj} (Sy)	WL_{Gj} (Sx) (denied)			WL_{Gi} (Sy)	Gi 완료 RLU_{Gi} (Sx)	RGU_{Gi} (Sy)	WL_{Gj} (Sx) (acquired)	
Sx	R_{Gix} (a)									W_{Gjx} (a)
Sy		R_{Giy} (b)		R_{Ly} (b)	R_{Ly} (c)	W_{Giy} (c)				

(그림 3) 예제 4의 전역 스케쥴

위 (그림 3)에서 보인 스케쥴은 제안된 두 개의 전역 트랜잭션이 직접 및 간접충돌에 대해 직렬화 순서를 유지하면서 병행수행하고 있음을 보이고 있다. 그러나 기존 제안된 방법들에서는 두 개의 지역 S_x와 S_y에서 각각 간접충돌의 가능성을 가지고 있기 때문에 위에서 보인 것과 같은 스케쥴을 생성할 수 없다. 즉 위와 같은 두 개의 전역 트랜잭션에 대해 병행수행 할 수 없게 된다.

4. 증명 및 분석

4.1 지역-로킹 프로토콜의 정확성 증명

이 절에서는 지역-로킹 프로토콜이 항상 전역적으로 직렬화된 순서를 생성함을 증명한다. 먼저, 지역 충돌이 없는 두 전역 트랜잭션 사이에는 간접충돌이 발생하지 않음을 증명하고, 지역 충돌이 있는 경우에는 전역 트랜잭션 관리기에 의해 그 직렬화 순서가 결정될 수 있음을 증명하여, 지역 충돌이 발생하는 경우와 그렇지 않은 경우 모두 전역 트랜잭션에 대해 직렬화 가능함을 증명한다.

[보조정리 1] 지역 S_x에서 수행되는 전역 서브트랜잭션 G_{ix}와 G_{jx}의 연산 $O_{Gi}(a)$와 $O_{Gj}(b)$에 대하여, 제안된 <표 4>의 지역-로킹 호환표에 의해 서로 지역 충돌이 발생하지 않는 경우 두 전역 트랜잭션사이에는 간접충돌이 발생하지 않는다.

(증 명) 두 연산 $O_{Gi}(a)$와 $O_{Gj}(b)$가 서로 지역 충돌이 없는 경우, 두 연산 중에 하나는 반드시 판독 연산이고, 다른 나머지 하나의 연산은 판독 연산이거나 기록 연산이다. 따라서, 오직 다음과 같은 스케쥴만 지역 충돌이 발생하지 않는다.

경우 1 : 두 연산이 모두 판독 연산인 경우 $R_{Gi}(a)$ $R_{Gj}(b)$
데이타 항목 a와 b중에 하나는 반드시 전역 데이타 항목이다. 만약, 둘 다 전역 데이타 항목인 경우, 두 데이타가 서로 다를 필요는 없다. 그렇지 않은 경우 $GD \cap LD = \emptyset$이므로 $a \neq b$이다.

경우 2 : 두 연산중 하나는 판독 연산이고 다른 하나는 기록 연산인 경우 $R_{Gi}(a)$ $W_{Gj}(b)$
데이타 항목 a는 반드시 전역 데이타 항목이어야 하며, 데이타 항목 b는 지역 또는 전역 데이타 항목일 수 있다. 이 때 $a \neq b$이다.

위의 스케쥴에서 간접충돌이 없음을 보이기 위해, G_{ix}가 G_{jx}와 간접충돌이 있음을 가정하자. G_{ix}와 G_{jx} 사이에 간접충돌을 만들게 하려면 이 들과 직접충돌을 갖는 지역 트랜잭션(들) L_i가 반드시 존재한다. 경우 1의 스케쥴에서 지역 트랜잭션 L_i가 전역 트랜잭션 G_{ix}, G_{jx}와 직접충돌을 가지려면 지역 트랜잭션은 전역 트랜잭션이 판독하는 데이타 항목에 대한 기록 연산을 포함해야 한다. 따라서 지역 트랜잭션이 반드시 전역 데이타 항목에 대해서 기록 연산을 수행해야만 두

전역 트랜잭션 G_{ix}, G_{jx} 사이에 간접충돌이 발생하게 된다. 그러나 앞에서 제안한 전역 트랜잭션 모델에 따라 지역 트랜잭션이 전역 데이타 항목에 기록할 수 없다는 제약사항에 위반된다. 따라서 경우 1의 스케쥴이 간접충돌을 갖는다는 가정은 모순이다. 또한 경우 2의 스케쥴에서도 유사하게 두 개의 전역 트랜잭션 G_{ix}, G_{jx}와 지역 트랜잭션(들) L_i가 반드시 직접충돌을 가져야만 G_{ix}와 G_{jx} 사이에 간접충돌이 발생한다. 경우 2의 스케쥴에서 최소한 하나의 연산은 전역 데이타 항목에 대한 판독 연산이다. 따라서 지역 트랜잭션 L_i가 전역 데이타 항목에 대한 최소한 하나의 기록 연산을 가져야만 두 전역 트랜잭션 사이에 간접충돌이 발생한다. 경우 1의 경우와 마찬가지로, 이는 본 논문에서 제안한 전역 트랜잭션 모델에 위반된다. 결국, 경우 1과 경우 2의 모든 스케쥴에서 두 전역 트랜잭션 사이에 간접충돌을 발생시키려면 지역 트랜잭션이 전역 데이타 항목에 기록 연산을 가져야만 하지만, 이는 본 논문에서 사용하는 전역 트랜잭션 모델에 따라 발생할 수 없는 상황이다. 따라서, 두 전역 트랜잭션이 간접충돌을 갖는다는 가정에 모순이다. 그러므로, 전역 서브트랜잭션 G_{ix}와 G_{jx} 의 두 연산 $O_{Gi}(a)$와 $O_{Gj}(b)$이 서로 지역 충돌 모드가 아닌 경우에는 간접충돌이 발생하지 않는다.

[보조정리 2] 지역 S_x에서 전역 서브 트랜잭션 G_{ix}, G_{jx}의 두 연산 $O_{Gi}(a)$와 $O_{Gj}(b)$가 지역 충돌 모드인 경우, 전역 트랜잭션 관리기는 두 전역 트랜잭션 G_{ix}, G_{jx}간의 직렬화 순서를 결정할 수 있다.

(증명) 지역 충돌인 경우, 두 연산 $O_{Gi}(a)$와 $O_{Gj}(b)$ 사이에는 간접충돌이 발생한다. 즉, 하나 이상의 지역 트랜잭션에 의해 두 연산과 직접충돌을 유발한다. 전역 트랜잭션 관리기는 간접충돌이 발생할 두 연산을 직렬화 하기 위해서 해당 지역-로크 연산을 수행시킨다. <표 4> 지역-로킹 호환표에 의해 지역 충돌인 두 로킹 연산은 전역적 수준에서 배타적으로 수행된다. 즉, 먼저 로킹을 선점한 트랜잭션이 전역적으로 완료된 후, 해당 로킹을 획득하고 그 지역으로 보내져 수행되므로, 전역적 수준에서 직렬화 순서가 결정된다. 따라서, 지역-로킹 호환표에 의해 지역 충돌인 두 연산은 로킹 획득순서에 의해 그 직렬화 순서가 결정된다.

[정리 1] 제안한 지역-로킹 프로토콜은 전역적 직렬성을 보장한다.

(증명) 전역적 직렬성을 보장함을 증명하기 위해 전역 트랜잭션들의 데이타 충돌에 대해 직렬화 가능함을 보인다. 데이타 충돌이 없는 경우에는 직렬화 순서를 결정할 필요가 없다. 데이타 충돌이 발생하는 경우에는 간접충돌이 발생하는 경우와 직접충돌이 발생하는 경우로 나눌 수 있다. 전역 트랜잭션 사이에 직접충돌이 발생하는 경우 기존 2PL 방법과 같은 방법으로 로킹 연산을 취하므로 직렬성이 보장됨을 쉽게 알 수 있다. 또한 간접충돌이 있는 경우, 전역 트랜잭션들은 지역-로킹 연산을 지역-로킹 호환표에 따라 획득해야 하므로, 전역 트랜잭션 관리기 수준에서 결정된 직렬화 순서가 각 지역 시스템의 직렬화 순서와 동일함을 보조정리 2에서 증명하였다. 따라서 보조정리 2에 따라 직접 또는 간접충돌, 즉 지역 충돌이 발생하는 경우 전역적 직렬성을 보장함을 알 수 있다. 또한 보조정리 1에 따라, 전역 트랜잭션 사이에 지역 충돌이 발생하지 않는 경우, 직접충돌 뿐만 아니라 간접충돌도 발생하지 않음을 알 수 있다. 따라서, 전역 트랜잭션 사이에 발생하는 모든 데이타 충돌은 제안한 지역-로킹 프로토콜에 따라 직렬화 가능함을 알 수 있다.

4.2 지역 경쟁률(site contention)의 분석

2절에서도 언급한 바와 같이, 낙관적 티켓 방법은 이질형 통합 데이타베이스 시스템의 전역 트랜잭션을 관리하는 기법 중에서 간단하면서도 전역적 직렬성을 강력히 보장하는 한 방법이다. 본 절에서는 제안된 지역-로킹 방법과 기존의 낙관적 티켓 방법에 대한 지역 경쟁(*site contention*)률을 비교 분석한다. 기존 데이타베이스 시스템에서 트랜잭션들이 데이타 접근 경쟁(data contention)을 하는 것과 유사하게, 지역 경쟁률은 전역 트랜잭션의 지역 접근 경쟁을 나타낸 것이다.

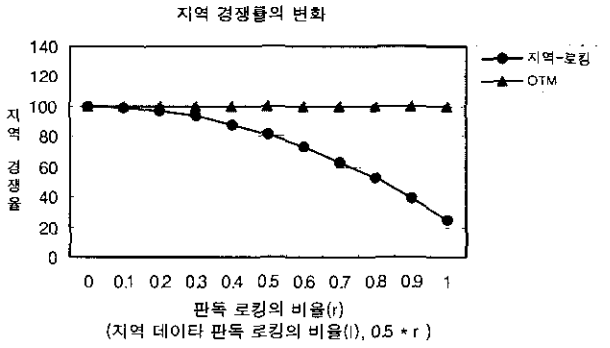
낙관적 티켓 방법의 기본적인 요지는 전역 트랜잭션이 지역을 접근하기 위해 사용하는 추가적인 데이타 항목(티켓)을 각 지역 시스템에 위치시키는 것이다. 모든 전역 트랜잭션들이 해당 지역을 접근하기 위해 티켓이라는 데이타를 하나씩 증가시키게 하므로써, 전역 트랜잭션 사이에 강제적인 데이타 충돌을 유발하게 하고, 이를 통해 전역 트랜잭션 관리기 수준에서 해당지역에서 실행된 지역 수행 순서를 파악하게끔 하는 것이다. 낙관적인 티켓 방법에 대한 자세한 알고리즘은 [10, 11]에 기술되어 있다. 공정하게 지역 경쟁률을 측정하기 위해 제안하는 방법과 낙관적 티켓 방법 모두 지역적으로 2PL을 사용한다고 가정한다.

지역 경쟁률은 전역 트랜잭션들이 병행수행될 때, 각 지역에 대한 접근 경쟁이 얼마나 되는가를 나타내고 있다. 지역 경쟁률(SC)는 다음과 같이 표현할 수 있다.

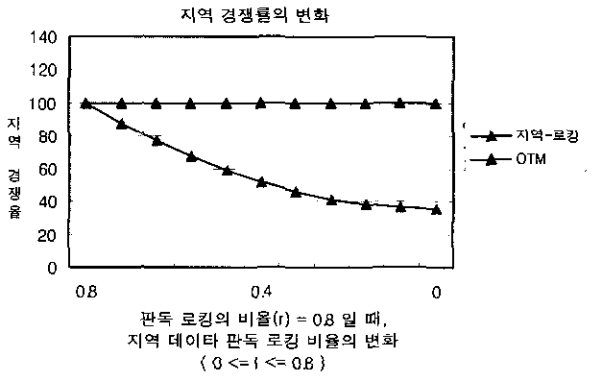
$$SC = SC = \frac{(1-(r^2-l^2))\times k^2\times N}{S},$$

N : 현재 활성화된 전역 트랜잭션의 총 개수
k : 전역 트랜잭션이 요구하는 지역 로킹의 수
S : 전체 지역의 개수
r : 요청한 전체 로킹 수 중에서 판독 로킹에 관한 비율 $(0 \le r \le 1)$
l : 요청한 전체 로킹 수 중에서 지역 데이타에 대한 판독 로킹(RLL)의 비율$(0 \le l \le r)$

SC의 값은 [2, 16]에 기술된 데이타 경쟁률 표현식을 이용하여 지역에 대한 접근 경쟁률을 표현한 것이다. 원래의 데이타 경쟁률 표현식에서는 기존의 2PL 로킹 호환표에 따른 경쟁률을 표현하기 위해 판독연산의 비율만으로 경쟁률을 표현할 수 있다. 본 식에서는 제안된 <표 4>의 호환표에 따른 지역 경쟁률을 표현하기 위해, 상호 RLL 로킹 충돌시의 경쟁을 으로 표현하여 데이타 경쟁률의 표현식[2, 16]을 수정하였다. <표 4>의 호환표에서 로 표기된 RGL과 WL로킹 충돌시의 조건적 충돌 부분은 수식의 단순화를 위하여 무조건 비호환, 즉 충돌이 발생하는 경우로 가정하여 제안된 방법에서 제공할 수 있는 경쟁률보다 다소 높은 경쟁률을 보인다. 그러나 비교 대상인 OTM방법은 본 수식에 의해 정확한 경쟁률을 보인다. 본 수식의 SC 값이 클수록 지역에 대한 접근 경쟁이 심하여 트랜잭션이 블록되거나 대기하는 시간이 길어지게 된다. 요청하는 판독 로킹의 수의 변화에 따라 SC 값의 변화를 집중적으로 측정하기 위하여, r 값이 0일 때 $k^2 \times N/S$ 부분을 100으로 고정하여 측정하였다. (그림 4)는 RGL 로킹 대 RLL 로킹의 비율을 0.5(l = 0.5 × r)로 하여 판독 로킹의 비율에 따라 지역 접근에 대한 경쟁률의 변화를 각각의 방법에 대하여 측정한 값을 나타내고 있다. (그림 5)는 판독 연산의 비율을 0.8로 고정하고, RLL 로킹의 비율을 변화하였을 때, 즉 $0 \le l \le 0.8$의 값을 가질 때의 지역 경쟁률을 보인다.



(그림 4) 판독 로킹 비율 변화에 따른 지역 경쟁률 비교
(RLL의 비율 = 0.5, 즉, l = 0.5 × r)



(그림 5) 판독 로킹 비율이 0.8일 때, RLL의 비율 변화에 따른 지역 경쟁률 비교

앞에서 소개한 낙관적 티켓 방법은 판독 연산의 비율이 높아지더라도, 티켓이라는 데이타 항목에 대해 기록 연산을 수행해야 하므로, 모든 다른 전역 트랜잭션과의 지역 경쟁이 발생한다. 따라서 판독 연산의 비율의 변화에 반응 없이 SC의 값이 100으로 고정됨을 알 수 있다. 그러나 본 방법에서는 전역 트랜잭션들의 판독 연산 비율이 높아지거나 판독연산중 전역 데이타에 대한 판독 연산 비율이 높아질수록, 지역 충돌이 발생하지 않는 경우가 증가하여, SC의 값이 현저하게 낮아짐을 알 수 있다. 즉, 지역-로킹 호환표상에 공유모드에 해당하는 경우가 증가하게 되어 전역 트랜잭션의 병행수행 가능성이 증가하게 되고, 동시성 정도도 높아지게 된다.

5. 결론 및 추후연구

본 논문의 목적은 이질형 통합 데이타베이스 시스템에서 기존 방법들보다 높은 병행성을 제공하는 전역 트랜잭션 관리 방법의 설계이다. 기존 연구의 주요 문제점은 지역 트랜잭션에 의한 갱신을 알 수 없는 상황에서 전역 일관성을 보장하는 전역 트랜잭션의 병행수행 제어 방법을 설계하기 어렵다는 것과 전역 트랜잭션 간의 간접충돌 상황을 정확히 파악할 수 없어서 매우 포괄적인 범위를 간접충돌 상황으로 간주함으로써 전역 트랜잭션의 병행성을 떨어지게 하는 것이다. 이러한 문제들은 전역 트랜잭션 사이에 불필요한 데이타 충돌을 야기할 뿐만 아니라 전역 일관성마저 위반하는 상황을 유도하게 된다. 본 논문에서는 전역 무결성 제약사항의 특징을 이용하여 전역 트랜잭션 모델을 정의하고, 이를 기반으로 지역-로킹 연산과 이에 따른 프로토콜을 제안하여 전역 일관성을 유지하는 병행수행 제어 기법을 설계하였다. 본 방법은 전역 트랜잭션이 판독 연산을 많이 포함할수록 그 기대 효과가 크며, 그렇지 않은 상황에서도 기존 방법보다 더 적은 데이타 경쟁을 보여 성능상에 이익을 얻을 수 있다. 또한 통합 데이타베이스를 기반으로 한 웹 응용 시스템에서 동적 데이타를 게시하고, 이를 기반으로 판독 전용인 분석적 요구를 많이 발생 시키는 응용, 즉 증권 분석 시스템이나 의사 결정 시스템과 같은 응용에서 더 큰 성능 향상을 볼 수 있다. 본 방법은 로킹 연산에 기반을 둔 방법이므로, 전역적 교착상태(deadlock)을 유발할 수 있으나 기존 제안된 교착상태 방지 또는 탐지 알고리즘을 활용하여 쉽게 해결할 수 있다. 또한 여러 특정응용에 적합하도록 확장이 용이하며, 현재 본 방법을 기반으로 한 낙관적 병행수행 제어 기법을 연구 중이다.

참 고 문 헌

[1] R. Alonso, H. Garcia-Molina, and K. Salem. "concurrency control and recovery for global procedures in federated database systems," A quarterly bulletin of the IEEE

Technical Committee on Data Engineering, 10(3) : pp.5-11, 1987.

[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman. "Concurrency Control and Recovery in Database Systems," Addison-Wesley Publishing Company, 1987.

[3] Y. Breitbart, D. Georgakopolous, M. Rusinkiewicz, and A. Silberschatz. "on rigorous transaction scheduling," IEEE Transactions on Software Engineering, 17(9) : pp.954-960, 1991.

[4] Y. Breitbart and A. Silberschatz. "strong recoverability in multidatabase systems," In Proceedings of the Research Issues in Data Engineering, pp.170-175, 1992.

[5] Y. Breitbart, A. Silberschatz, and G. R. Thompson, "reliable transaction management in a multidatabase system," In Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, pp.215-224, 1990.

[6] W. Du and A. K. Elmagarmid. "quasi serializability : a correctness criterion for global concurrency control in interbase," In Proceedings of the 15th International Conference on Very Large Data Bases, pp.347-355, 1989.

[7] W. Du, A. K. Elmagarmid, and W. Kim. "maintaining quasi serializability in multidatabase systems," In Proceedings of the Research Issues in Data Engineering, pp.360-367, 1991.

[8] W. Du, A. K. Elmagarmid, Y. Leu, and S. Osterman. "effects of autonomy on maintaining global serializability in heterogeneous distributed database systems," In Proceedings of the 2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering, pp.113-120, 1989.

[9] A. K. Elmargarmid and A. Heral. "supporting updates in heterogeneous distributed database systems," In IEEE Proceedings of the 4th International Conference on Data Engineering, pp.564-569, 1988.

[10] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. "on serializability of multidatabase transactions through forced local conflicts," In Proceedings of the 7th International Conference on Data Engineering, pp.314-323, 1991.

[11] D. Georgakopoulos, M. Rusinkiwicz, and A. P. Sheth. "using tickets to enforce the serializability of multidatabase transactions," IEEE Transactions on Knowledge and Data Engineering, 6(1) : 166-180, 1993.

[12] Kyuwoong Lee and S. Park. "Chapter 7 : Optimistic Concurrency Control for Maintaining the Global Integrity Constraints in MDBSs, IFIP TC11 WG11.5 Integrity and Internal Control in Information Systems, Volume 1," Chapman & Hall, 1997.

[13] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz. "the concurrency control problem in multidatabases : Characteristics and solutions," In Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, pp.288-297, 1992.

[14] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. "non-serializable execution in heterogeneous distributed database systems," In Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems, pp.245-252, 1991.

[15] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. "relaxing serializability in multidatabase systems," In Proceedings of the Research Issues in Data Engineering, pp. 205-212, 1992.

[16] Y. C. Tay, N. Goodman, and R. Suri. "locking performance in centralized databases," ACM Transactions on Database Systems, 10(4) : 415-462, Dec. 1985.

[17] A. Wolski and J. Veijalainen. "2pc agent method : Achieving serializability in presence of failures in a heterogeneous multidatabase," In Proceedings of PARBASE-90 Conference, pp.268-287, 1990.

[18] Vasilis Vassalos and Yannis Papakonstantinou, "describing and using query capabilities of heterogeneous sources," In Proceedings of the 23rd International Conference on Very Large Data Bases, pp.256-275, 1997.

[19] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang, "optimizing queries across diverse data sources," In Proceedings of the 23rd International Conference on Very Large Data Bases, pp.276-285, 1997.

[20] Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-Molina, "object fusion in mediator systems," In Proceedings of the 23rd International Conference on Very Large Data Bases, pp.413-424, 1996.

[21] Kyuwoong Lee, Seog Park, and Gil-Rok Oh, "concurrency control for global transaction management in MDBS," Lecture Notes in Computer Science, Vol 1677, pp.812-821, 1999.

[22] Y. Breitbart, H Garcia-Molina,A. Silberschatz, "Overview of Multidatabase Transaction Management," VLDB Journal, Vol.1, No.2, pp.181-23. October, 1992.

이 규 웅

email : leekw@mail.sangji.ac.kr

1990년 한국외국어대학교 전자계산학과 (이학사)

1992년 서강대학교 대학원 전자계산학과 (공학석사)

1998년 서강대학교 대학원 전자계산학과 (공학박사)

1998년~2000년 한국전자통신연구원 인터넷서비스 연구부 선임연구원

2000년~현재 상지대학교 컴퓨터.정보공학부 전임강사

관심분야 : 트랜잭션 처리, SAN 기반 자료저장 시스템, 분산 및 실시간 DB