분산 다중 데이터베이스 시스템에서 완화된 제어 기준을 기반으로 한 전역 동시성 제어 알고리즘

신 동 천[†] · 김 진 배[††]

요 약

다중 데이터베이스 시스템에서 전통적인 전역 직렬가능성의 유지는 참여하는 지역 데이터베이데이스 시스템의 자치성 보장으로 인하여 어려운 문제로 인식되어 왔다. 본 논문에서는 데이터베이스 통합이 데이터베이스 무결성 보장에 영향을 주지 않는 응용에 적용할 수 있는 분산 다중 데이터베이스 시스템에서 완화된 동시성 제어 기준을 기반으로 하는 전역 동시성 제어 알고리즘을 제안한다. 제안된 알고리즘에서 전역 트랜잭션의 실행 여부를 결정하기 위한 과정은 내부 검증과 외부 검증으로 구성된다.

A Global Concurrency Control Algorithm based on Relaxed Control Criterion in Distributed Multidatabase Systems

Dong-Cheon Shin[†] · Jin-Bae kim[††]

ABSTRACT

To maintain the traditional notion of global serializability in multidatabase systems has been recognized as one of difficult problems due to the autonomy of participating local database systems. In this paper, we propose a global concurrency control algorithm based on the relaxed concurrency control criterion in distributed MDBSs, which can be applied to applications in which the database integration has no effect on the integrity constraints. In the proposed algorithm, the process to determine whether a global transaction can be executed or not consists of intra-validation phase and inter-validation phase.

1. 서 론

다중 데이터베이스 시스템(MultiDataBase System : MDBS)은 각 지역 사이트의 데이터 자원으로 구성되는 다중 데이터베이스를 관리하는 시스템으로 다중 데이터베이스의 논리적인 통합을 제공한다. 논리적인 통합에서의 데이터 처리 환경은 크게 분산(distribution), 이질성(heterogeneity), 자치성(autonomy)으로 특징지워진다[13]. 특히, MDBS는 이미 존재하는 지역 데이터베이스 시스템(LDBS)의 논리적인 통합이므로 각 LDBS의 자치성 보장은 MDBS에서 충족시켜야 할 가장 중요한 특성이 된다.

MDBS에서 여러 사이트의 데이터를 액세스하는 전역(global) 트랜잭션 실행에 가장 어려운 문제중의 하나는 트랜잭션의 특성을 보장하면서 지역 자치성을 보장하는 문제이다. 이러한 자치성의 보장은 MDBS에서 트랜잭션 관리 문제를 어렵게 하고 있으며 이러한 점이 기존의 분산 데이터베이스 환경과 근본적으

† 정 회 원 : 중앙대학교 정보시스템학과 교수
†† 준 회 원 : 한국과학기술원 대학원 정보 및 통신공학과
논문접수 : 1998년 8월 22일, 심사완료 : 1999년 10월 11일

로 다르다. 따라서, 기존의 분산 데이터베이스 관리 기법을 그대로 적용할 수 없으므로 새로운 원칙과 규약, 그리고 여러 기법 등의 개발이 필요하다[3, 4, 7, 9, 14, 15].

MDBS에서는 여러 지역 데이터베이스를 액세스하는 전역 트랜잭션(global transaction)과 지역 데이터베이스만을 액세스하는 지역 트랜잭션(local transaction)이 존재하게 된다. 각 LDBS는 전역 트랜잭션을 각 지역 트랜잭션과 동일하게 인식하여 실행함으로써 각 사이트에서의 직렬가능성(serializability)은 보장한다. 그러나, 각 사이트에서 실행된 전역 트랜잭션들의 실행 순서가 서로 다를 수 있으므로 지역(local) 직렬가능성의 보장만으로는 전역(global) 직렬가능성(GSR)을 보장할 수 없다[5, 7, 16].

MDBS에서 GSR의 보장은 각 지역 데이터베이스 시스템의 통합 후에도 무결성 제약 조건(integrity constraints)을 만족시키고 동시에 실행되는 전역 트랜잭션 사이에서 발생할 수 있는 바람직하지 않은 현상[2] 즉, 비일관적인 검색(inconsistent retrieval)이나 갱신 손실(lost update)등을 방지함으로써 올바른 동시 실행을 보장하기 위한 것이다. GSR을 보장하기 위해서는 각 사이트에서 실행된 전역 트랜잭션들의 실행 순서에 대한 정보가 필요하다. 그러나, 지역 자치성의 보장은 이러한 정보를 LDBS가 MDBS에게 제공하지 않음을 의미하므로 MDBS에서 GSR의 보장 문제가 어려워지게 된다.

충돌 기반의 직렬가능성(conflict-based serializability : CSR)[2]을 기준으로 한 많은 동시성 제어 알고리즘이 제안되었으나 이들 방법은 동시성 정도의 저하 가능성을 근본적으로 갖거나[1, 6, 13] 참여하는 LDBS에 제한[3, 4, 5]을 두고 있다. 이와 함께, GSR 보장 대신에 동시성 정도를 향상시키려는 목적에서 LSR[10], 2LSR[12], QSR[8] 등과 같은 완화된 새로운 동시성 제어 기준들이 제안되었다. 이러한 완화된 동시성 제어 기준을 기반으로 하는 알고리즘들이 데이터베이스 일관성을 보장하기 위해서는 제한 예를 들어, 무결성 제약(integrity constraint), 트랜잭션 구조, 그리고 응용의 데이터 액세스 등과 관련된 제한 등이 필요함을 기술하고 있다. 그러나, 이러한 제한은 MDBS 환경이 기존에 존재하는 LDBS의 통합 환경이라는 사실을 감안하면 MDBS 환경에 적합한 전역 응용들의 특성에 따라서는 현실적인 제한이라고 할 수 있다.

본 논문에서는, 동시성 정도를 향상시키기 위하여 완화된 새로운 동시성 제어 기준을 제시한다. 제안된 기준도 일관성을 보장하기 위해서는 기존의 완화된 기준[8, 10, 12]들과 마찬가지로 MDBS 구축 과정에서 데이터베이스 통합으로 무결성 보장에 영향을 받지 않는 응용으로 제한하고 있다. 다음으로, 기존의 대부분의 연구[1, 6, 11]가 하나의 전역 트랜잭션 관리자를 고려하는 것과는 달리 여러 개의 관리자가 존재하는 분산(distributed) MDBS에서 완화된 기준을 기반으로 하는 전역 동시성 제어 방법을 제안한다.

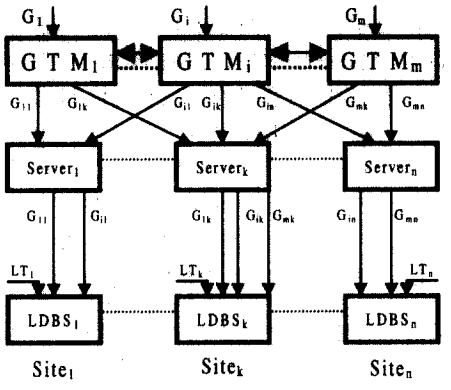
본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 가정하는 시스템 모델에 대해 소개한다. 3장에서는 완화된 동시성 제어 기준을 소개하고 4장에서는 제시된 기준을 기반으로 하는 전역 동시성 제어 알고리즘을 제시한다. 끝으로, 5장에서는 결론을 기술한다.

2. 분산 MDBS 모델

MDBS에서 지역 데이터에 대한 접근은 트랜잭션을 통하여 이루어진다. 트랜잭션은 C 나 PASCAL 등의 프로그래밍 언어를 호스트 언어로 하는 사용자 프로그램으로 원자성, 일관성, 고립성, 그리고 지속성(ACID)등의 특성을 갖는다. 본 논문에서는 트랜잭션 T_i는 시작(b_i) 연산과 완료(c_i) 혹은 철회(a_i) 연산 사이에 읽기(r_i)와 쓰기(w_i) 연산들의 집합으로 간주한다. MDBS에는 다음과 같은 두가지 유형의 트랜잭션이 있다.

- 지역 트랜잭션 : 지역 사이트에 있는 데이터만을 액세스하는 트랜잭션으로 LDBS에 직접 제출되어 MDBS의 제어를 받지 않고 LDBS에 의해 실행되고 관리된다.
- 전역 트랜잭션 : 여러 사이트에 존재하는 데이터를 액세스 하는 트랜잭션으로 여러개의 부트랜잭션들로 구성되며 MDBS에 의해 관리된다. 각 부트랜잭션은 LDBS에 의해 지역 트랜잭션과 동일하게 인식되어 실행된다.

본 논문은 (그림 1)과 같은 분산 MDBS 모델을 기반으로 하고 있다. 분산 MDBS 모델의 주요 구성 요소는 사이트 S_1, S_2, ..., S_m ($m \geq 2$)에 이미 존재하는 자치적인 LDBS들과 전역 트랜잭션의 제출과 관리를 담당하는 전역 트랜잭션 관리자(GTM)들, 그리고 각 LDBS와



(그림 1) 분산 MDBS 모델

연관된 서버(server)들이다. 전역 트랜잭션은 임의의 GTM_i에 제출되며 GTM_i에 제출된 전역 트랜잭션 G_i는 액세스하는 데이터가 존재하는 사이트에 따라 GTM_i에 의해 부트랜잭션 g_{i1}, g_{i2}, ..., g_{in}으로 분해된다. 분해된 부트랜잭션은 MDBS의 동시 실행 제어 알고리즘에 따라 해당 지역 사이트와 연관된 서버에 전송된다. 즉, g_{ii}는 사이트 S_i의 $LDBS_i$와 연관된 $server_i$에 전송된다. 각 서버는 GTM으로부터 연산들을 받아 LDBS에 제출하고 처리 결과를 해당 GTM에게 알려주어 GTM과 LDBS사이에서 인터페이스 역할을 한다. 한편, 전역 트랜잭션이 동시에 여러 GTM으로 제출될 수 있으므로 이들 GTM사이에는 동시실행에 관한 제어 정보의 교환이 이루어진다.

이러한 구성 요소와 관련하여 다음과 같은 기본적인 가정을 둔다.

- 각 LDBS에는 한 전역 트랜잭션에 대해 최대로 하나의 부트랜잭션이 존재한다. 즉, GTM은 전역 트랜잭션을 받아 액세스할 데이터의 위치에 따라 부트랜잭션들로 분해하여 해당 사이트에 전송한다.
- 전역 트랜잭션의 부트랜잭션들은 한번에 하나씩 LDBS에게 제출된다. 이는 한 전역 트랜잭션의 부트랜잭션들의 순차적인 실행을 의미하지는 않으며 단지 제출만 순차적으로 이루어짐을 의미한다.
- 각 LDBS는 자신에게 제출된 부트랜잭션의 실행 결과에 대해 연관된 서버에 통보해 주며 올바른 실행을 보장한다.
- 고장 발생은 고려하지 않는다.

3. 완화된 동시성 제어 기준

MDBS에서 일관성 보장은 직접 혹은 간접 충돌하는 전역 트랜잭션들의 실행을 올바르게 제어함으로서 가능하다. 완화된 동시성을 제시하기 위한 충돌의 정의는 다음과 같다.

[정의 1 : 직접 충돌] 동시에 실행되면서 데이터 x를 액세스 하는 트랜잭션 T_i와 T_j의 임의의 연산을 각각 $O_i(x)$와 $O_j(x)$라 하자. $O_i(x)$와 $O_j(x)$중 적어도 하나가 쓰기 연산일 때 T_i와 T_j는 *직접 충돌한다*고 말한다.

[정의 2 : 간접 충돌] 직접 충돌이 없는 트랜잭션 T_i와 T_j에 각각 직접 충돌하는 트랜잭션 T_k의 개입으로 T_i와 T_j사이에 $T_i(T_j)$-->$T_j(T_i)$ 의 직렬 순서(serialization order)가 형성될 때 T_i와 T_j는 *간접 충돌한다*고 말한다.

직렬가능성 이론[2]에 따르면 동시에 실행되는 트랜잭션들 사이의 직렬 순서는 아래와 같은 2가지 경우에 성립한다. 즉, 동시에 실행되는 트랜잭션사이의 직렬 순서는 한 트랜잭션의 연산 실행 효과가 동시에 실행되는 다른 트랜잭션의 실행 결과에 영향을 줄 때 형성된다.

1) 읽기/쓰기 충돌 경우 : 한 트랜잭션의 읽기 연산에 의해 읽은 값은 동시에 실행되는 다른 트랜잭션의 쓰기 연산과의 실행 순서에 종속된다.

2) 쓰기/쓰기 충돌의 경우 : 한 트랜잭션의 쓰기 연산에 의한 최종 값은 동시에 실행되는 다른 트랜잭션의 쓰기 연산과의 실행 순서에 종속한다.

한편, MDBS 환경에서는 지역 트랜잭션의 개입으로 직접 충돌이 없는 전역 트랜잭션사이에 간접 충돌이 가능하다. 따라서, 전역 트랜잭션사이의 직접 충돌뿐만 아니라 간접 충돌로 인하여 한 전역 트랜잭션들의 연산 실행 효과가 동시에 실행되는 다른 전역 트랜잭션의 실행 결과에 영향을 줄 수가 있다. 결국, 트랜잭션사이의 직렬 순서는 영향을 미치는 순서라고 할 수 있다. 따라서, MDBS 환경에서 데이터베이스

통합이 무결성 보장에 영향을 주지 않는다고 하면 전역 트랜잭션 입장에서는 지역 트랜잭션의 실행 결과를 또 다른 일관성 있는 데이터베이스 상태로의 전이라고 볼 수 있다. 그러므로, 지역 트랜잭션의 실행으로 전역 트랜잭션끼리 영향을 주는 경우에 동시 실행을 제어함으로써 트랜잭션의 동시 실행으로 발생할 수 있는 비일관적인 검색과 같은 바람직하지 않은 현상을 방지할 수 있다.

[정의 3 : 영향 순서] 데이터 x와 y를 액세스 하는 트랜잭션 T_i와 T_j의 임의의 연산을 각각 $O_i(x)$와 $O_j(y)$라 하자(x와 y는 다른 데이터일 필요는 없다). $O_i(x)$나 $O_j(y)$의 실행 결과가 이들 연산의 실행 순서에 종속될 때 T_i와 T_j 사이에 *영향 순서(influence order)*가 있다고 한다.

직접 충돌하는 전역 트랜잭션사이에는 영향 순서가 항상 존재하므로 기존의 직렬 순서와 동일하지만 간접 충돌하는 전역 트랜잭션사이에는 간접 충돌을 유발한 지역 트랜잭션의 역할에 따라 영향 순서가 존재하게 된다.

[정의 4 : 영향 그래프] 트랜잭션 집합 $T=\{T_1, T_2,...,T_n\}$에 대한 스케줄을 S라 하자. S에 대한 *영향 그래프(influence graph : IG)*는 T에 속한 트랜잭션들이 노드를 형성하며 S에 속한 T_i의 연산이 $T_j(i\neq j)$의 연산에 선행하고 T_i와 T_j사이에 영향 순서가 있을 때 에지 T_i-->T_j가 형성되는 방향성 그래프이다.

[정의 5 : 영향 직렬가능성] 전역 스케줄 S가 다음 조건을 모두 만족할 때 S는 *영향 직렬가능성(influence serializability : ISR)*을 만족한다.

조건 1) 각 지역 스케줄 $S_i(i=1,2,...,n)$는 직렬가능성을 보장한다.

조건 2) 각 지역 스케줄 S_i에서 전역 트랜잭션사이의 영향 순서는 모두 동일하다.

기존의 직렬 이론에서 SG에 사이클이 존재하면 직렬가능성을 만족하지 못하는 것과 마찬가지로 전역 스케줄 S에 대한 IG에 사이클이 존재하면 S는 ISR을 만족하지 못하게 된다. ISR의 조건 1)은 각 LDBS가 직렬가능성을 보장한다는 가정에 따라 당연하다. 조건 2)가 만족되는 경우는 두 가지로 나누어 생각할 수 있다. 첫째, 읽기/쓰기 나 쓰기/쓰기의 직접 충돌이 있는 경우로 이 경우에 모든 지역 스케줄의 영향 순서가 동일하다는 조건은 모든 지역 사이트에서의 전역 트랜잭션의 직렬 순서가 동일함을 의미한다. 따라서, 이 경우는 GSR의 조건과 동일한 조건이 됨을 알 수 있다. 둘째, 간접 충돌이 있는 경우는 지역 트랜잭션의 개입으로 인한 간접 충돌이 조건 2)의 보장에 미치는 영향을 분석할 필요가 있다. **정리 1**은 이러한 분석 결과를 보여 주고 있다.

[정리 1] 간접 충돌이 일어나는 임의의 사이트에서 직접 충돌이 없는 전역 트랜잭션들의 연산에 읽기 연산과 쓰기 연산 모두가 포함되지 않은 경우에는 간접 충돌이 발생하여도 전역 트랜잭션사이에 영향 순서는 존재하지 않는다.

<증명> 임의의 2개의 전역 트랜잭션을 고려해도 일반성을 잃지 않으므로 전역 트랜잭션을 G_i와 G_j라하고 간접 충돌이 일어나는 사이트를 S_n이라 하자. 직접 충돌이 없으므로 G_i와 G_j의 실행 효과에 영향을 줄 수 있는 경우는 지역 트랜잭션 LT의 개입으로 인하여 $G_i(G_j) \rightarrow LT \rightarrow G_j(G_i)$의 직렬 순서가 형성되는 경우이다. G_i와 G_j가 공통의 데이터를 액세스하는 경우에 둘 다 읽기 연산이면 직접 충돌이 아니므로 지역 트랜잭션의 개입으로 이러한 직렬 순서가 가능하다. 그러나, 이 경우 읽기 연산들이므로 이들 연산은 서로의 실행 효과에 영향을 주지 않는다. 액세스하는 데이터가 다른 경우를 생각하자. a와 b를 액세스 되는 데이터라고 할 때 G_i-->LT를 위해서는 이들 사이에 $r_i(a)w_t(a)$, 혹은 $w_i(a)w_t(a)$, 혹은 $w_i(a)r_t(a)$의 스케줄이 있어야 되고 LT-->G_j를 위해서는 이들 사이에 $r_t(b)w_j(b)$, 혹은 $w_t(b)w_j(b)$, 혹은 $w_t(b)r_j(b)$의 스케줄이 있어야 된다. 따라서, 이러한 스케줄을 포함하는 스케줄은 다음과 같이 18가지가 된다.

- $r_i(a)w_t(a)r_t(b)w_j(b)$
- $w_i(a)w_t(a)r_t(b)w_j(b)$
- $w_i(a)r_t(a)r_t(b)w_j(b)$
- $r_i(a)w_t(a)w_t(b)w_j(b)$
- $w_i(a)w_t(a)w_t(b)w_j(b)$
- $w_i(a)r_t(a)w_t(b)w_j(b)$
- $r_i(a)w_t(a)w_t(b)r_j(b)$
- $w_i(a)w_t(a)w_t(b)r_j(b)$
- $w_i(a)r_t(a)w_t(b)r_j(b)$
- $r_t(b)w_j(b)r_i(a)w_t(a)$
- $r_t(b)w_j(b)w_i(a)w_t(a)$
- $r_t(b)w_j(b)w_i(a)r_t(a)$
- $w_t(b)w_j(b)r_i(a)w_t(a)$
- $w_t(b)w_j(b)w_i(a)w_t(a)$
- $w_t(b)w_j(b)w_i(a)r_t(a)$
- $w_t(b)r_j(b)r_i(a)w_t(a)$
- $w_t(b)r_j(b)w_i(a)w_t(a)$
- $w_t(b)r_j(b)w_i(a)r_t(a)$

위의 스케줄 가운데 실제 서로의 실행 효과에 영향을 주는 스케줄은 $w_i(a)\ r_t(a)\ w_t(b)\ r_j(b)$ 뿐이다. 이 스케줄에서 G_i가 쓴 a의 값을 LT가 읽어 b의 값을 쓰는데 영향을 미친다면 b를 읽는 G_j는 G_i의 영향을 받은 것으로 생각할 수 있기 때문이다. 다른 스케줄들은 의미적으로 G_i와 G_j 사이에 서로 영향을 주지 않는다. 따라서, 영향을 주는 스케줄에는 G_i가 쓰기 연산을 갖고 있고 G_j가 읽기 연산을 갖고 있는 경우이다. 그러므로, 직접 충돌이 없는 전역 트랜잭션들이 쓰기 연산과 읽기 연산 모두를 포함하고 있지 않으면 간접 충돌로 인하여 전역 트랜잭션은 서로의 실행 효과에 영향을 받지 않으므로 영향 순서는 존재하지 않는다. ■

[보기 1] 데이터 a가 있는 사이트 S_1과 데이터 b와 c가 있는 사이트 S_2를 생각하자. 전역 트랜잭션 T_1과 T_2, 그리고 S_1의 지역 트랜잭션 T_3와 S_2의 지역 트랜잭션 T_4가 다음과 같다고 하자.

$T_1 : r_1(a)w_1(b)$ $\quad$ $T_2 : r_2(a)r_2(c)$

$T_3 : w_3(a)$ $\quad$ $T_4 : r_4(b)w_4(c)$

T_1과 T_2는 직접 충돌이 없고 T_3와 T_4는 모두 T_1과 T_2와 직접 충돌한다. 이 경우에 트랜잭션 T_1이 실행되고 완료된 후 트랜잭션 T_2가 실행되고 완료된다고 가정하면 각 사이트에서 아래와 같은 지역 스케줄이 가능하다. ■

$S_1 : r_1(a)c_1w_3(a)c_3r_2(a)c_2$ $\quad (T_1 \text{-->} T_3 \text{-->} T_2)$

$S_2 : r_4(b)w_1(b)c_1r_2(c)c_2w_4(c)c_4$ $\quad (T_2 \text{-->} T_4 \text{-->} T_1)$

따라서 위와 같은 스케줄은 지역 트랜잭션 T_3와 T_4의 개입으로 간접 충돌이 발생하여 SG에서 T_1과 T_2 사이에 사이클이 형성되므로 GSR을 만족하지 못하는 스케줄이지만 각 사이트에서 지역 트랜잭션 T_3와 T_4의 개입에도 불구하고 T_1이나 T_2의 실행 결과에 서로가 영향을 주지 않으므로 IG에 사이클이 존재하지 않게 되어 ISR을 만족하는 스케줄이다.

4. 동시성 제어 알고리즘

4.1 동시성 제어 알고리즘

분산 MDBS에서는 여러 개의 GTM이 존재하므로 각기 다른 GTM에게 제출되는 전역 트랜잭션사이의 영향 순서를 파악하기 위해서는 GTM 사이에 정보 교환이 필요하게 된다. **보기 2**는 직접충돌이 있는 경우에 이러한 필요성을 보여주고 있다.

[보기 2] : 데이터 a, b 와 데이타 d, e 가 존재하는 사이트 S_1과 S_2 를 고려하자. GTM_1과 GTM_2에 제출되는 전역 트랜잭션을 G_1과 G_2, S_1 과 S_2 의 지역 트랜잭션을 각각 LT_3 와 LT_4 라고 하고 다음과 같다고 하자.

$G_1 : w_1(a)r_1(e)$ $\quad$ $G_2 : r_2(b)w_2(d)$

$LT_3 : r_3(a)w_3(b)$ $\quad$ $LT_4 : r_4(d)w_4(e)$

GTM_1과 GTM_2가 각각 G_2와 G_1의 존재를 모르므로 이들 사이에 제어를 위한 정보 교환이 없으면 GTM_1과 GTM_2는 각각 G_1과 G_2의 실행을 허용하게 된다. 이런 경우 다음과 같은 스케줄이 가능하여 IG에 사이클이 형성되어 ISR은 위배된다.

$S_1 : w_1(a)r_3(a)w_3(b)r_2(b)$ $\quad (G_1 \text{-->} LT_3 \text{-->} G_2)$

$S_2 : w_2(d)r_4(d)w_4(e)r_1(e)$ $\quad (G_2 \text{-->} LT_4 \text{-->} G_1)$ ■

결국, 분산 MDBS에서 하나의 전역 트랜잭션을 실행하기 위해서는 제출된 GTM내의 다른 전역 트랜잭션과의 영향 순서뿐만 아니라 다른 GTM 이 관리하고 있으면서 동시 실행되는 전역 트랜잭션과의 영향 순서도 고려하여야 한다. 따라서, 하나의 전역 트랜잭션을 실행하기 위해서는 GTM은 자기에게 제출된 다른 전역 트랜잭션과의 동시 실행 및 다른 GTM에게 제출된 전역 트랜잭션과의 동시 실행이 ISR을 보장하는지를 검사하여야 한다.

동시성 제어 알고리즘을 기술하기 위해 k개의 GTM 집합 $GTM=\{GTM_1, GTM_2, \ldots\ldots, GTM_k\}$, n개의 전역 트랜잭션 집합 $G=\{G_1, G_2, \ldots\ldots, G_n\}$, 그리고 m개의 사이트 집합 $S=\{S_1, S_2, \ldots\ldots, S_m\}$을 고려하자. 그리고 GTM_c에 제출되어 사이트 S_k에서 실행될 G_i의 부트랜잭션을 ${}_cg_{ik}$ 라 하자. 분산 MDBS 환경에서 동시성 제어의 핵심은 GTM_c가 ${}_cg_{ik}$를 해당 사이트 S_k에 전송하여 실행시킬 것인지를 결정하는 것이다.

이를 위해 GTM_c는 1차(내부 검증)로 자신에게 제출되어 실행되는 전역 트랜잭션과의 동시실행이 ISR을 보장하는지를 검사한다. ISR을 보장하지 않으면 다른 GTM_p에게 제출된 전역 트랜잭션과의 영향 순서에 관계없이 ISR은 보장되지 않으므로 ${}_cg_{ik}$를 S_k에

전송하지 않고, 보장하면 다른 GTM_p에게 제출되어 실행되는 전역 트랜잭션과의 동시 실행도 ISR을 보장하는지를 2차(외부 검증)로 검사한다. 외부 검증에서도 보장이 되면 ${}_cg_{ik}$를 S_k에 전송하고 보장되지 않으면 전송하지 않는다.

GTM_c가 ${}_cg_{ik}$의 실행이 ISR을 보장하는지의 검사는 IG 상에 사이클이 존재할 가능성을 기반으로 이루어진다. 사이클에 포함된 임의의 전역 트랜잭션은 다른 전역 트랜잭션과의 영향 순서가 적어도 2개의 다른 사이트에서 있다. 바꾸어 말하면, 한 사이트에서의 영향 순서 참여만으로는 사이클의 형성이 불가능하므로 실행을 허용하여도 ISR은 보장된다. 즉, G_i가 다른 전역 트랜잭션과 영향 순서를 형성하면서 동시에 실행되는 서로 다른 사이트의 개수를 No_G_i라 할 때 No_G_i가 1이면 ${}_cg_{ik}$를 S_k에 전송한다. No_G_i는 G_i가 제출될 때 0으로 초기화 된다. 여기서, 동일한 사이트에서의 각기 다른 여러 트랜잭션과의 영향 순서는 $No\text{-}G_i$ 에 영향을 주지 않음을 주목하자.

외부 검증에서 GTM_c가 다른 GTM_p에게 제출되어 실행되는 전역 트랜잭션과의 동시 실행이 ISR을 보장하는지를 결정하기 위해서는 다른 모든 GTM_p와 통신하여 영향 순서에 관한 정보를 교환하고 유지하여야 한다. 이를 위해, 조정자 역할을 하는 GTM_c와 참여자 역할을 하는 다른 모든 GTM_p사이의 검증은 아래와 같이 2단계의 원자적(atomic)인 과정으로 이루어진다.

- 1단계 :
  - GTM_c가 S_k에서 실행되는 G_i의 연산 유형과 데이터를 나타내는 메세지 $<G_i,\{O_n_Type,Data\},S_k>$를 모든 다른 GTM_p에게 보낸다.
  - GTM_p는 G_i와 S_k에서 영향 순서 형성 여부를 검사하여 영향 순서가 형성되지 않으면 메시지 $<'yes',\{\}>$을, 그렇지 않으면 영향 순서가 형성되는 G_j에 대한 정보와 함께 메시지 $<'no',\{G_j,No_G_j\}>$을 GTM_c에게 보낸다.
- 2단계 :
  - GTM_c는 모든 GTM_p의 메시지가 'yes'이면 G_i의 부트랜잭션 ${}_cg_{ik}$를 S_k에 전송하는 것으로 결정하고 그렇지 않은 경우, No_G_i 와 No_G_j 값에 따라 ${}_cg_{ik}$를 S_k에 전송할 것인지를 결정한다. 전송으로 결정되면 'no'로 응답한 GTM_p의 $\{G_j\}$가 G_i와 S_k에서 영향 순서가 형성되어 실행중임을 나타내는 메시지 $<G_i,S_k,\{G_j\}>$를 모든 GTM_p에게 보내고 'yes'로 응답한 모든 GTM_p에게는 $<G_i,S_k,\{\}>$을 보낸다. 비전송으로 결정되면 영향 순서가 형성되지 않으므로 모든 GTM_p에게 메시지 $<G_i,S_k,\{\}>$을 보낸다.
  - GTM_p는 $\{Gj\}$와 G_i가 S_k에서 동시에 실행중이므로 No_Gj를 1 증가시킨다.

2단계에서 'no' 응답은 G_i와 영향 순서가 형성되는 GTM_p내의 전역 트랜잭션이 S_k에 존재함을 의미한다. 그러나, No_G_i 값이 1로 유지되고 있다면 그 이유는 G_i가 S_k에서 영향 순서가 있거나 다른 사이트 S_h에서 영향 순서가 있어서이다. 따라서, No_G_i 값이 S_k에서의 영향 순서로 유지되는 경우라면 GTM_p내의 전역 트랜잭션과 동일한 사이트 S_k에서의 영향 순서이므로 No_G_i값은 증가되지 않고 그렇지 않으면 증가된다.

(그림 2)는 GTM_c에 제출되어 사이트 S_k에서 실행되어야 하는 임의의 한 전역 트랜잭션 G_i의 부트랜잭션 ${}_cg_{ik}$를 GTM_c가 S_k에 전송할 것인지를 결정하는 과정을 정리한 알고리즘 **DGTM**이다. 알고리즘 **DGTM**은 각각 조정자와 참여자로서 GTM이 수행하는 부분에 대한 기술로 구성된다. (그림 2)에 나타나는 함수의 기능은 다음과 같다.

- *direct_conflict(t_1,t_2,s)* : 사이트 s에서 트랜잭션 t_1과 t_2가 직접충돌이면 "참값"을, 아니면 "거짓값"을 돌려주는 함수
- *mixed_op(t_1,t_2,s)* : 사이트 s에서 트랜잭션 t_1과 t_2의 연산의 합집합에 읽기 연산과 쓰기 연산이 모두 있으면 "참값"을, 아니면 "거짓값"을 돌려주는 함수
- *inter_validation(t,s,from)* : 사이트 s에서 트랜잭션 t의 실행을 위한 2단계 검증과정을 수행하는 함수로 from은 No_t의 값에 따라 검증 과정에서 태그(tag) 역할을 함

```
DGTM_c(RS_s(cg_ik), WS_s(cg_ik), OP_s(cg_ik))

/* 전역 트랜잭션 G_i의 부트랜잭션 cg_ik를 사이트 S_k에 전송하기 위한 조정자 역할*/
/* cGT_k: GTM_c의 관리하에 사이트 S_k에서 실행중인 전역 트랜잭션의 집합 */
/* RS_s(cg_ik), WS_s(cg_ik): 부트랜잭션 cg_ik 의 사이트 S 에서의 읽기 집합과 쓰기 집합 */
/* OP_s(cg_ik): 부트랜잭션 cg_ik의 사이트 S 에서의 연산자 유형('R','W') 집합 */
/* 입력: RS_s(cg_ik), WS_s(cg_ik), OP_s(cg_ik)  */
/* 출력: 부트랜잭션 cg_ik의 실행을 위한 전송 여부 */
{
switch (No_G_i)  {              /* 내부 검증 */
case 0: if (direct_conflict(cg_ik,cg_jk,S_k) OR distinct_op(cg_ik,cg_jk,S_k)) AND (!No_G_j))
                                                                  for all G_j ∈ cGT_k
            No_G_i++;                                             /* S_k에서 처음 영향순서 */
        inter_validation(cg_ik,S_k,'0');                          /* 외부 검증 */
        break;
case 1: if (direct_conflict(cg_ik,cg_jk,S_k) OR distinct_op(cg_ik,cg_jk,S_k)) for all G_j ∈ cGT_k
            cg_ik 를 S_k 에 전송하지 않는다;                       /* 전송하면 No_G_i=2  */
      else inter_validation(cg_ik,S_k,1);                         /* 외부 검증 */
      break;
default:  exit;                 /* error */
}
return;
}

inter_validation(cg_ik,S_k,from)                                  /* 외부 검증 */
{ 모든 GTM_p 에게 <cg_ik,{OP_Type,Data},S_k> 메시지를 보낸다;      /* 1단계(조정자) */
  모든 GTM_p 으로부터 응답을 기다린다;
  if (모든 응답이 'yes')                                          /* 2단계(조정자) */
    {  모든 GTM_p 에게 메세지 <G_i,S_k,{}>를 보낸다;
       cg_ik 를 S_k 에 전송한다;
       return;
    }
   else                                                           /* 'no' 응답 메세지 내의 모든 G_j 에 대해
*/
     switch (from)
      { case 0: if (!No_G_j)
                  { if (!No_G_j)
                        { No_G_i++;
                          'no' 와 'yes' 로 응답한 GTM_p 에 각각 메시지 <G_i,S_k,{G_j}> 와
                      <G_i,S_k,{}> 를 보낸다;
                          cg_ik 를 S_k 에게 전송한다;
                        }
                else {cg_ik 를 S_k 에게 전송하지 않는다;
                        모든 GTM_p 에게 메시지 <G_i,S_k,{}>를 보낸다; }
                  else
                    { if (!No_G_j)
                          { 'no'와 'yes' 로 응답한 GTM_p 에게 각각 메시지 <G_i,S_k,{G_j}> 와
                        <G_i,S_k,{}> 를 보낸다;
                          cg_ik 를 S_k 에게 전송한다; }
                else { No_G_i--;                                  /* 내부 검증 과정 보상 */
                        cg_ik 를 S_k 에게 전송하지 않는다;
                       GTM_p 에게 메세지 <G_i,S_k,{}>를 보낸다; }
                    break;
        case 1: 모든 GTM_p 에게 메세지 <Gi,Sk,{}>를 보낸다;
                cg_ik를 S_k 에게 전송하지 않는다;
                  break;
        default:                exit;                             /* error */
      }
    return;
}

DGTM_p( )
/* 외부 검증에서 참여자 역할 부분 */
{
GTM_c 로부터 메세지 <cg_ik,{OP_Type,Data},S_k>를 받는다;      /* 1단계(조정자) */
if (direct_conflict(cg_ik,pg_jk,S_k) OR distinct_op(cg_ik,pg_jk,S_k))  for all G_j ∈ pGT_k
            GTM_c 에게 메세지 <'no',{G_j,No_G_j}>를 보낸다;
else  GTM_c 에게 메세지 <'yes',{}>를 보낸다;
GTM_c 로부터 응답 메시지를 기다린다;
if (GTM_c 로부터 메세지 <G_i,S_k,{G_j}>를 받으면)              /* 2단계(조정자) */
    {G_j++};                    /* 매시지내의 모든 G_j 에 대해 */
}

direct_conflict(t_1,t_2,s)
{ if RS_s(t_1) ∩ WS_s(t_2) ≠ ∮ return(1);
  else return(0);
}

distinct_op(t_1,t_2,s)
{ if OP_s(t_1) ∪ OP_s(t_2) = {'R','W'} return(1);
  else return(0);
}
```

(그림 2) 알고리즘 DGTM

[정리 2] 알고리즘 DGTM은 ISR을 보장한다.

<증명> ISR은 IG에서 사이클이 존재할 때 위배되므로 알고리즘 **DGTM**이 ISR을 보장함을 증명하기 위해서는 사이클 형성과 관련된 다음 2가지가 정당함을 보이면 된다.

1) 영향 순서 형성의 결정

알고리즘 **DGTM**에서는 직접 충돌이 있거나(*direct-conflict()*) 간접 충돌하는 트랜잭션 연산의 합집합에 읽기와 쓰기 연산이 있는 경우(*distinct-op()*)에만 ISR 보장에 영향을 미치는 영향 순서로 간주하고 있다. (정리 1 참조)

2) 한 사이트에서의 영향 순서 형성은 허용

전역 트랜잭션 G가 한 사이트에서만 다른 전역 트랜잭션과 영향 순서를 형성하면 실행을 허용해도 즉, 전역 트랜잭션 G의 서로 다른 충돌 사이트 개수인 No_G의 값이 0 이거나 1인 경우 실행을 허용해도 ISR은 보장됨을 보이면 된다. 동시에 실행되는 임의의 전역 트랜잭션사이에 G_1-->G_2-->.....-->G_{n-1} 의 실행 순서가 있다고 가정하자. 이제 사이트 S_k에서 G_n이 G_{n-1}과 영향 순서가 있는데 실행을 허용하면 G_1-->G_2-->.....-->G_{n-1}-->G_n의 실행 순서가 가능하지만 사이클은 불가능하다. 만약, 다른 사이트 S_h에서 G_n이 G_1과 영향 순서가 있는데 실행을 허용하면 G_1-->G_2-->.....-->G_{n-1}-->G_n-->G_1의 실행 순서가 가능하여 사이클이 형성될 수 있다. 한편, 동일한 사이트 S_k에서 G_n이 영향 순서에 관여하면 G_1, G_{n-1}, 그리고 G_n은 S_k의 LDBS에 의해 사이클이 없는 실행 순서가 형성된다. 이상에서와 같이 G_n의 실행이 사이클이 형성되려면 G_n은 적어도 2개의 서로 다른 사이트에서 영향 순서에 관여해야 한다. 따라서, 한 사이트에서만 영향 순서를 형성하는 경우에는 실행을 허용하여도 ISR은 보장된다.

증명의 완전함을 위하여 이제 한 사이트에서만은 실행을 허용하는 결정의 기초가 되는 전역 트랜잭션 G의 No_G의 값이 올바르게 유지되는지를 보이기로 한다. 임의의 GTM_c에 제출되어 임의의 사이트 S_k에서의 실행되려는 임의의 전역 트랜잭션 G_i를 고려하자.

내부 검증에서 G_i가 S_k에서 GTM_c내의 다른 트랜잭션과 영향 순서가 처음으로 형성되면 No_G_i값은 증가된다. 그러나, 다음부터 형성되는 영향 순서는 동일한 사이트 S_k에서의 영향 순서이므로 No_G_i값은 변하지 않는다. 내부 검증에서 No_G_i 의 값이 0 이거나 1일 경우에는 GTM_c가 관리하는 전역 트랜잭션과의 동시 실행은 ISR을 보장하므로 임의의 다른 GTM_p가 관리하는 전역 트랜잭션과의 영향 순서 형성 여부를 검증하기 위하여 외부 검증을 수행하게 된다.

경우 1 : No_G_i의 값이 0에서 외부 검증하는 경우

GTM_p의 응답이 모두 'yes' 이면 G_i는 모든 사이트에서 영향 순서가 없으므로 실행이 허용되며 No_G_i는 변함이 없고, 그렇지 않으면 S_k에서만 GTM_p가 관리하는 전역 트랜잭션 G_j와 영향 순서가 있는 경우이다. 이 경우 G_i 의 실행을 허용하면 G_j 도 S_k에서 새로운 영향 순서에 관여하게 되므로 $No\text{-}G_j$ 의 값에 따라 G_i 의 실행이 결정된다. $No\text{-}G_j$ 의 값이 0 이면 G_i 의 실행을 허용해도 $No\text{-}G_j$ 의 값은 1이 되므로 G_i의 실행은 허용되고 No_G_i의 값은 1이 증가되어야 한다. $No\text{-}G_j$ 의 값이 1이면 G_i 의 실행으로 $No\text{-}G_j$ 의 값이 2가 되므로 G_i 의 실행은 허용되지 않는다.

경우 2 : No_G_i 의 값이 1에서 외부 검증하는 경우

No_G_i의 값이 1이 되는 경우는 1) G_i가 S_k에서 처음으로 영향 순서가 형성되었거나 혹은 2)다른 사이트 S_h에서 이미 영향 순서가 있는 경우이다. 전자의 경우, GTM_p의 응답이 모두 'yes' 이면 S_k에서만 영향 순서가 존재하므로 실행은 허용되며 No_G_i의 값은 내부 검증시에 1이 이미 증가되었으므로 변하지 않는다. 응답에 'no'가 있으면 GTM_p가 관리하는 전역 트랜잭션 G_j 도 S_k에서 새로운 영향 순서에 관여하게 되므로 $No\text{-}G_j$ 의 값에 따라 G_i 의 실행이 결정된다. $No\text{-}G_j$ 의 값이 0 이면 G_i 의 실행을 허용해도 $No\text{-}G_j$ 의 값은 1이 되므로 G_i 의 실행은 허용된다. 그러나 G_j와의 영향 순서 역시 S_k에서의 영향 순서이므로 No_G_i의 값은 증가되지 않아야 한다.

후자의 경우, GTM_p의 응답이 모두 'yes'이면 S_k에서의 영향 순서는 없으므로 실행은 허용되며 No_G_i의 값은 증가되지 않아야 한다. 응답에 'no' 가 있으면 S_k에서도 영향 순서를 형성하므로 실행은 허용되지 않고 따라서 No_G_i의 값도 증가되지 않아야 한다. 이와 마찬가지로, 외부 검증 후에 G_i와 S_k에서 영향 순서가 형성되는 GTM_c 내 혹은 GTM_p 내의 다른 전역

트랜잭션들의 값도 이에 따라 유지된다. 따라서, 알고리즘 **DGTM**은 이상에서 기술한 바에 따라 No_G_i의 값이 올바르게 유지된다.

1) 2) 로부터 알고리즘 **DGTM**은 ISR을 보장한다.■

알고리즘 **DGTM**에서 G_i가 완료되면 동시에 실행중이던 전역 트랜잭션들의 동시 실행 트랜잭션 목록에서 G_i를 제거하고 G_i와 동시에 실행 중이던 전역 트랜잭션의 실행이 종료될 때까지 G_i와 관련되어 유지되던 영향 순서 정보는 계속 유지하여야 한다. 왜냐하면, G_i가 완료하는 즉시 정보를 삭제하면 G_i 때문에 대기중인 다른 전역 트랜잭션 G_j의 실행이 가능하므로 동시성 정도의 향상을 기대할 수 있지만 G_i와 G_j 사이에 간접 충돌의 영향으로 영향 순서가 형성되어 ISR을 위배할 수 있기 때문이다.

4.2 논의

이 절에서는 중앙 집중식 MDBS에서 충돌 기반의 전역 직렬가능성을 동시성 제어 기준으로 하고 참여하는 LDBS에 제약을 두지 않는 환경에 적용할 수 있는 대표적인 동시성 제어 방법들[1,6,11]을 분산 MDBS로 확장하는 문제를 개괄적으로 언급하고 제안한 방법을 간략히 평가한다.

■ site locking 방법[1]을 확장하는 문제

이 방법은 동일한 사이트에서 동시에 전역 트랜잭션이 실행되지 않고 한 트랜잭션의 실행이 종료된 후에 다른 트랜잭션이 그 사이트에서 실행 가능하므로 동시실행 정도가 현저히 떨어지며 간접 충돌이 발생하는 경우에는 전역 직렬가능성이 보장되지 않는 방법이다. 이 방법을 분산 MDBS로 확장하는 방안의 개관은 다음과 같다. 사이트 S_k에서 실행되려는 전역 트랜잭션 G가 GTM_c내의 내부 검증에서 통과하여도 GTM_p내의 다른 전역 트랜잭션이 S_k에서 실행되는지 검증하여야 한다. GTM_p내의 전역 트랜잭션이 S_k에서 실행되지 않아야 G는 실행된다. 따라서, 이 방법은 중앙 집중식 MDBS에서 마찬가지로 검증 과정이 매우 단순하지만 동시실행 정도가 매우 낮게 된다.

■ site graph 방법[6]을 확장하는 문제

이 방법은 트랜잭션과 사이트를 정점으로 그래프를 유지하므로 사이클이 형성되지 않으면 한 사이트에서도 여러 트랜잭션을 동시에 실행시킬 수가 있다. 그러나, 간접 충돌로 인한 전역 직렬가능성 위배를 막기 위해 트랜잭션 완료 후에도 해당 트랜잭션의 에지 제거가 연기되어야 하므로 트랜잭션의 동시 실행이 제한을 받게 된다. 이 방법을 분산 MDBS로 확장하는 방안의 개관은 다음과 같다. GTM_c내의 트랜잭션 G가 내부 검증에서 성공하면 다른 GTM_p 내의 검증을 위한 그래프에서도 검증 과정을 거쳐야 한다. 이 과정에서 GTM_p내의 검증 그래프만 사용하면 전역 직렬가능성이 위배될 수 있으므로 GTM_c 내의 검증 그래프를 함께 통합하여 고려하여야 하는 부담이 있다. 이에 따라, 검증을 위해 GTM사이에 검증 그래프를 교환하거나 중복하여 유지하여야 하는 부담이 수반된다.

■ 낙관적 ticket 방법[11]을 확장하는 문제

이 방법은 각 사이트에서 제한 없이 트랜잭션을 실행시킨 후 완료하기 전에 정당한 실행이었는지를 검증하는 방법으로 트랜잭션의 재시작 가능성과 검증에 따른 정보 유지의 필요성이라는 문제점이외에 충돌이 없는 트랜잭션사이에도 강제로 충돌을 유발한다는 문제점도 갖는다. 이 방법을 분산 MDBS로 확장하는 방안의 개관은 다음과 같다. 검증 단계에서 모든 GTM과 통합하여 검증하여야 되므로 site graph 방법과 마찬가지로 GTM사이에 통합된 검증 비용과 이에 따라 검증을 위한 정보를 교환하거나 중복하여 유지하여야 하는 부담이 수반된다.

이상에서와 같이 분산 MDBS에서 동시성 제어는 GTM과의 정보 교환이 바탕이 되어야 한다. 따라서, 분산 MDBS에서의 동시성 제어 방법의 성능은 검증 방법의 효율성뿐만 아니라 GTM사이의 통신비용에 영향을 받게 된다. 한편, 본 논문에서는 GTM사이의 정보 교환이 원자적인 2단계 과정으로 이루어진다. 그리고, 당연히 제안된 방법은 정보를 중복하여 공유하거나 더 많은 정보를 유지함으로서 통신비용을 줄이거나 동시 실행 정도를 향상시킬 수 있다.

본 논문에서 제안한 방법은 한 사이트에서의 영향 순서만을 허용하고 그 이상의 사이트에서 허용할 경우 ISR 위배 가능성 때문에 동시 실행을 허용하지 않는다. 따라서, site lockig이나 site graph 방법과 마찬가지로 비관적인 방법으로 분류할 수 있다. 한편, 제안한 방법은 충돌 기반의 전역 직렬가능성을 동시성 제어 기준으로 하지 않고 동시에 실행되는 전역 트랜

잭션들이 서로에게 미치는 영향을 고려하였기 때문에 읽기(혹은 쓰기)전용의 전역 트랜잭션이 많이 존재하는 환경에서는 충돌 기반의 전역 직렬가능성을 기준으로 하는 다른 방법보다 좋은 동시 실행 효과를 기대할 수 있다. 다중 데이터베이스 시스템이 이미 자치적으로 운용되고 있는 여러 데이터베이스에 대한 액세스를 지원하는 시스템임을 감안하면 실제로 전역 트랜잭션이 읽기 전용일 경우가 많다고 판단할 수 있다. 이러한 환경이라면 읽기 전용의 트랜잭션을 배려하는 동시성 제어가 필요하며 제안한 방법은 이러한 환경에 적합한 방법이라 할 수 있다.

그러나, 이러한 완화된 동시성 제어 기준을 기반으로 하는 대부분의 방법은 일관성을 보장하기 위하여 데이터베이스 접근이나 응용에 제약 조건을 설정하고 있으며 제안된 방법도 예외는 아니다. 제안된 방법에서는 다중 데이터베이스 시스템 구축을 위한 데이터베이스의 통합이 무결성 보장에 영향을 주지 않는다는 전제가 필요하다. 따라서, 제안된 방법이 폭넓은 응용에 적용되기 위해서는 무결성 보장과 관련하여 일관성을 보장할 수 있는 무결성 제약조건 모델 도출에 대한 연구가 무엇보다도 필요하다.

5. 결 론

분산 MDBS에서 전역 직렬가능성을 보장하기 위해서는 여러 GTM사이의 통신이 필수적이기 때문에 중앙집중형 MDBS에서 전역 직렬가능성을 보장하기 위한 기법을 분산 MDBS 환경으로 확장하여 적용하는 경우에 중앙집중형 MDBS에서와 동일한 성능을 나타낸다고 단정하기는 어렵다. 본 논문에서는 이러한 인식에서 분산 MDBS에서의 동시성 제어 문제를 고찰하였다.

이를 위해 우선 본 논문에서는 전역 직렬가능성 기준을 완화하는 것이 동시성 향상에 기여한다고 판단하여 데이터베이스 통합이 무결성 보장에 영향을 주지 않는 응용에 적용할 수 있는 완화된 동시성 제어 기준으로 영향 직렬가능성(ISR)을 제안하였다. 제안된 기준은 충돌이 아닌 전역 트랜잭션사이의 영향 순서를 기반으로 하고 있어 영향 순서와 무관한 읽기 전용의 전역 트랜잭션이 많은 환경에서는 좋은 동시 실행 정도를 기대할 수 있다. 다음으로, 분산 MDBS에서 ISR을 기반으로 하는 동시성 제어 알고리즘을 제안하였다.

한편, 다른 완화된 직렬가능성 기준에서와 마찬가지로 데이터베이스 통합이 무결성 제약 조건을 만족시키는 데 영향을 주지 않는다는 전제에서 데이터베이스 일관성을 유지할 수 있다. 따라서, ISR을 기준으로 하는 동시성 제어 알고리즘이 적용될 수 있는 MDBS 응용 환경을 넓히기 위해서는 ISR이 적용될 수 있는 무결성 제약 조건의 유형(모델)등을 도출해 내는 연구가 무엇보다도 필요하다. 아울러, ISR을 기반으로 하는 여러 효율적인 알고리즘 개발 및 평가에 대한 연구도 필요하며 본 논문에서 논의하지 않은 교착상태, 회복, 그리고 완료 등에 대한 연구도 뒤따라야 할 것이다.

참 고 문 헌

[1] Alonso, R., Garcia-Molina, H., and Salem, K., Concurrency Control and Recovery for Global Procedures in Federated Database Systems, Data Eng., Vol.10, No.3, pp.5-11, Sept. 1987.

[2] Bernstein, P.A., hadzilacos, V., and Goodman, N., Concurrency Control and Recovery in Database Systems. Reading, MA : Addison-Wesley, 1987.

[3] Breitbart, Y., Garcia-Molina, H., and Silberschatz, A., Transaction Management in Multidatabase Systems, Modern Database Systems, Kim, W., Ed. Addison-Wesley, pp.573-591, 1995.

[4] Breitbart, Y., Garcia-Molinar, H., and Silberschartz, A., Overview of Multidatabase Transaction Management, VLDB Journal, Vol.2, pp.181-239, 1992.

[5] Breitbart, Y., Georgakopoulos, D., Rusinkiewiez, M., and Silberschartz, A., On Rigorous Transaction Scheduling, IEEE Trans. on Software Eng., Vol 17, No.9, pp.954-960, Sept. 1991.

[6] Breitbart, Y. and Silberschartz, A., Multidatabase Update Issues, Proc. of ACM SIGMOD, pp.135-141, June 1988.

[7] Breitbart, Y., Silberschartz, A., and Thompson, R., Transaction Management Issues in a Failure-Prone Multidatabase System Environment, VLDB Journal, Vol.1, pp.1-39, 1992.

[8] Du, W. and Elmagarmid, A. K., Quasi Serializability : a Correctness Criterion for Global Concurrency Control in InterBase, Proc. of VLDB, pp.347-355, 1989.

[9] Elmagarmid, A. K., Jing, J., and Kim, W., Global Committability in Multidatabase Systems, IEEE Trans. on Knowledge and Data Eng., Vol.8, No.5, pp.816-824, Oct. 1996.

[10] Garcia-Molina, H. and Kogan, B., Achieving High Availability in Distributed Databases, IEEE Trans. on Software Eng., Vol.14, No.7, pp.886-896, 1988.

[11] Georgakopoulos, D., Rusinkiewicz, M., and Sheth, A.P., Using Tickets to Enforce the Serializability of Multidatabase Transactions, IEEE Trans. Know. and Data Eng., Vol.6, No.1, pp.166-180, Feb. 1994.

[12] Mehrotra, S., Rastogi, R., Korth, H. F., and Silberschatz, A., Ensuring Consistency in Multidatabases by Preserving Two-Level Serializability, ACM Trans. on Database Syst., Vol.23, No.2, pp.199-230, June 1998.

[13] Sheth, A. and Larson, J., Federated databases : Architectures and Integration, ACM Computing Surveys, Vol.22, No.3, Sept. 1990.

[14] Soparkar, N., Korth, H. F., and Silberschatz, A., Failure-Resilient Transaction Management in Multidatabases, IEEE Computer, Vol.24, No.12, pp.28-37, Dec. 1991

[15] Triantafillou, P., An Approach to Deadlock Detection in Multidatabases, Information Systems, Vol.22, No.1, pp.39-55, 1997.

[16] Zhang, A. and Elmagarmid, A. K., A Theory of Global Concurrency Control in Multidatabase Systems, VLDB Journal, Vol.2, No., 3, pp.331-360, 1993.

신 동 천

e-mail : dcshin@cau.ac.kr

1985년 서울대학교 컴퓨터 공학과 졸업(학사)

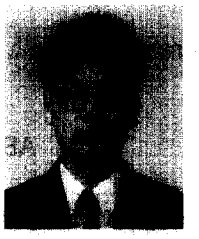
1987년 한국과학기술원 전산학과 졸업(공학석사)

1991년 한국과학기술원 전산학과 졸업(공학박사)

1991년~1993년 한국전산원 선임연구원

1993년~현재 중앙대학교 정보시스템학과 조교수, 부교수

관심분야 : 다중데이타베이스, 데이터웨어하우스, 이동컴퓨팅



김 진 배

e-mail : jbkim@dbsun3.kaist.ac.kr

1985년 경북대학교 전자공학과(학사)

1988년 한국과학기술원 전산학과 (석사)

1993년~현재 한국과학기술원 정보 및 통신공학과(박사과정)

1988년~1991년 삼성 SDS 기술연구소

1991년~1994년 코아시스템

1994년~현재 시.에스.텍

관심분야 : 데이터베이스 설계, 데이터웨어하우징, 시스템, 거래관리