# 낙관적 2단계 완료 규약에서 트랜잭션 상태전이를 기반으로 한 전역 트랜잭션 관리 규약

신 동 천†


## 요 약

분산 환경하에서 전역 트랜잭션의 원자성을 보장하기 위해 보편적으로 사용하는 2단계 완료규약(2PC)의 블록킹 현상을 완화하기 위해 낙관적 2PC를 사용할 수 있다. 낙관적 2PC는 보상 트랜잭션을 사용하여 트랜잭션의 원자성을 의미적으로 보장하게 되지만 보상 트랜잭션의 특성으로 인하여 전역 직렬성(global serializability)을 위배할 수 있다. 본 논문에서는, 전역 트랜잭션의 상태전이 분석을 통하여 전역 직렬성을 보장하는 올바른 전역 트랜잭션 관리 규약들을 제안한다. 제안된 규약들은 한 전역 트랜잭션의 실행을 결정하기 전에 동시에 실행되는 다른 전역 트랜잭션들의 상태를 검사하여 이 결과에 따라 전역 트랜잭션의 실행을 지연시킴으로써 전역 직렬성을 보장한다.


# Management Protocols of Global Transactions Based on Transaction State Transitions in an Optimistic 2-phase Commit Protocol

Dong Cheon Shin †


## ABSTRACT

To alleviate the blocking inherent in 2-phase commit protocol(2PC) that is widely used to ensure the atomicity of global transactions in distributed environments, an optimistic 2PC can be used. Under the optimistic 2PC, compensation transactions are introduced to ensure the atomicity of transactions, which may result in the violation of global serializability due to the properties of compensation transactions. In this paper, we propose several valid management protocols for global transactions to ensure the global serializability through the analysis of transaction state transitions. To determine the execution of one global transaction, the proposed protocols check the states of other concurrent global transactions. According to the result, execution of the global transaction may be delayed to ensure the global serializability.


## 1. 서 론

데이타베이스 트랜잭션은 데이타베이스를 액세스하는 기본 단위로 원자성, 일관성, 지속성, 고립성등 소위 ACID 특성을 갖는다[1, 8]. 이러한 트랜잭션의 특성을 만족시키기 위하여 트랜잭션 관리에 가장 필수적인 기능은 크게 동시성 제어와 회복 기능이다. 동시성 제어는 다중 사용자 환경에서 데이타베이스를 공유하는 경우에 데이타베이스의 일관성을 보장하기 위해 필요한 기능으로 직렬성(serializability)이 동시성 제어의 올바른 기준으로 인식되어 왔다[1, 11].

† 정 회 원: 중앙대학교 산업정보학과
논문접수: 1995년 11월 13일, 심사완료: 1996년 6월 20일

동시에 실행되는 트랜잭션들의 실행결과가 트랜잭션을 순차적으로 실행한 임의의 결과와 동일할 때 직렬성이 보장된다고 한다. 이는 직렬화 그래프(serialization graph:SG)에 사이클이 존재하지 않음을 의미한다.

분산 환경하에서 하나의 트랜잭션이 여러 사이트에 있는 데이타를 액세스 하여야 하는 경우 여러개의 부트랜잭션(subtransaction)으로 분해되어 해당 사이트에서 분산 처리된다. 이러한 경우에 가장 중요한 사항은 분산 처리되는 트랜잭션 즉, 전역 트랜잭션(global transaction)의 원자성(atomicity)과 전역 직렬성(global serializability)을 보장하는 일이다. 전역 트랜잭션의 원자성을 보장하기 위해 사용되는 가장 보편적인 규약은 2단계 완료규약(2PC)[7]이며 전역 직렬성을 보장하기 위해 보통 2단계 로킹규약(2PL)[1, 4]과 함께 사용된다.

2PC는 조정자(coordinator)가 참여자(participant)에게 투표를 요청하고 참여자가 '철회' 혹은 '완료'로 투표를 하는 투표단계(voting phase)와 조정자가 참여자의 투표결과에 따라 최종결정을 하여 참여자에게 통보하는 결정단계(decision phase)로 이루어진다. 참여자의 투표결과가 모두 '완료'이면 조정자는 '완료' 결정을 하고 그렇지 않으면 '철회' 결정을 하게 된다. 따라서, 참여자가 투표를 한후에 조정자의 최종결정을 기다리는 동안 참여자는 자신의 운명이 결정되지 않은 상태 즉, 불확실성(uncertainity) 상태가 된다.

2PC와 2PL이 결합되어 사용되는 경우에 2PC의 전형적인 문제점은 참여자의 불확실성으로 인한 블록킹(blocking) 현상[12]이다. 즉, 참여자 사이트에서 실행된 트랜잭션은 조정자 사이트로부터 최종적인 '결정' 메세지('철회' 혹은 '완료')를 수신할 때까지 자신이 소유한 데이타에 대한 로크를 소유하게 된다는 점이다. 이로 인하여, 해당 데이타를 필요로 하는 다른 트랜잭션의 실행이 지연되어 전체적으로 시스템의 가용성(availability)과 성능이 저하된다. 특히, 각 사이트의 자치성을 중요시 하는 다중 데이타베이스(multidatabase) 시스템[3]에서는 더욱 심각한 문제가 될 수 있다. 2PC의 블록킹 현상을 극복하기 위한 방법으로 3PC[12]와 낙관적 2PC[10]를 들 수 있다.

기존의 2PC에서 '결정' 메세지 수신때까지 로크를 소유하는 이유는 연속적 철회(cascading abort)를 피하여 트랜잭션의 고립성(isolation)을 보장하기 위함이다. 완료하지 않은 트랜잭션의 실행결과를 다른 트랜잭션이 액세스하는 경우에 해당 트랜잭션이 완료하지 않고 철회될 때 연속적 철회는 발생한다. 그러나 실제 환경에서 트랜잭션은 대부분 성공적으로 종료된다고 가정할 수 있다. [10]에서 제안한 낙관적 2PC는 이러한 가정에 기반을 두고 있다. 따라서, 낙관적 2PC는 트랜잭션이 성공적으로 종료되는 확률이 높을수록 좋은 성능을 보이게 된다.

낙관적 2PC는 사이트에서 실행된 트랜잭션이 '철회'로 투표하면 기존의 2PC와 마찬가지로 트랜잭션의 실행 결과를 'UNDO'한 후 소유한 로크를 반납하게 된다. 그러나, 트랜잭션이 '완료'로 투표하면 소유한 로크를 '결정' 메세지를 수신할 때까지 소유하지 않고 모두 반납하여 블록킹 현상을 피하자는 것이 낙관적 2PC의 기본 철학이다. 이러한 지역적 완료(locally-committed)는 해당 트랜잭션이 전역적으로 철회되는 경우 즉, 관련 사이트중에서 한 사이트라도 '철회'로 투표하는 경우에 트랜잭션의 원자성이 위배된다. 이 경우에 지역적 완료를 한 트랜잭션들의 실행 효과는 'UNDO'되어야 한다. 이를 해결하는 방안으로는 보상 트랜잭션(compensation transaction)[2, 6, 9, 10]을 사용하여 의미적 원자성(semantic atomicity)[5]을 보장하는 것이다.

보상 트랜잭션은 의미 독립성(independence of semantics)[2]을 갖으므로 각 사이트에서 실행되는 전역 보상 트랜잭션의 부트랜잭션들은 실행종료시에 소유한 로크들을 독립적으로 반납할 수 있게 되어 보상 트랜잭션을 도입하는 경우 전역 직렬성이 위배될 수 있다. 따라서, 임의의 트랜잭션은 보상 트랜잭션의 실행 후의 결과를 액세스하거나 실행전의 결과를 액세스해야한다는 보상 원자성(atomicity of compensation)[9]을 만족할 수 없게 되어 데이타베이스의 일관성이 위배되는 문제가 야기된다.

보상 트랜잭션의 도입으로 인한 직렬성 위배는 sagas[6]나 다단계 트랜잭션(multi-level transactions)[13, 14, 15] 개념을 적용하면 문제가 되지 않는다. sagas 나 다단계 트랜잭션은 복잡한 트랜잭션의 실행에서 동시성 정도의 향상등과 같은 시스템의 성능을 향상시키기 위하여 새로이 도입된 트랜잭션 모델 개념으로써 완료, 동시성 제어, 그리고 회복 등에 대한 제어 단위를 세분화한 것으로 볼 수 있다. 한편, [10]에서는 기

존의 단일 트랜잭션(flat transaction) 모델을 기반으로 분산 환경에서 트랜잭션의 원자성을 보장하기 위해 낙관적 2PC를 사용하는 경우에 전역 직렬성을 보장할 수 있는 2가지 규약을 제안하고 있다. 즉, [10]에서는 straitification 특성이라고 하는 직렬화 그래프(serialization graph:SG)의 특성을 유도하여 regular 사이클이 발생하지 않도록 전역 트랜잭션을 제한하므로써 직렬성을 보장할 수 있는 규약을 제안하고 있다.

본 논문에서는 낙관적 2PC하에서 전역 직렬성을 보장하기 위해 보상 트랜잭션을 도입하는 경우에 전역 트랜잭션 관리를 위해 가능한 모든 올바른 실행 규약들을 제안한다. [10]에서는 straitfication 특성을 이용하여 규약을 제안하고 있으나 본 논문에서는 전역 트랜잭션이 실행되는 동안 전이할 수 있는 트랜잭션의 상태를 분석하여 규약들을 제안하고 제안한 규약들의 정당성에 대해 논의한다. 상태전이 분석 결과, 한 전역 트랜잭션의 부트랜잭션들의 상태가 'UD' 와 'LC' (그림 2 참조) 가 공존하는 경우에 전역 직렬성이 위배될 가능성이 존재함을 알 수 있다. 제안된 규약들은 전역 트랜잭션들을 실행시키는 경우 전역 직렬성의 위배가능성이 존재하면 전역 트랜잭션의 실행을 지연하여 전역 직렬성을 보장한다. 동시성 정도에 따라 [10]에서 제안된 2개의 규약을 포함하여 11개의 가능한 규약들이 제안된다.

본 논문의 구성은 다음과 같다. 2장에서는 보상 트랜잭션의 특성과 본 논문에서 사용하는 트랜잭션 관리 모형에 대해 소개하고 3장에서는 전역 트랜잭션이 실행중에 전이할 수 있는 상태를 분석한다. 4장에서는 전역 트랜잭션 관리 규약을 제시하고 정당성에 대해 논의한다. 끝으로, 5장에서는 결론을 맺는다.

## 2. 트랜잭션 관리 모형

이 장에서는 낙관적 2PC에서 중요한 개념인 보상 트랜잭션에 대해 [10]에서 기술된 내용을 요약하여 소개하고 본 논문에서 사용하는 트랜잭션 관리 모형에 대해 기술한다.

### 2.1 보상 트랜잭션

트랜잭션 T의 보상 트랜잭션 CT는 T가 완료하기 전에 T의 실행 결과를 액세스한 트랜잭션들이 T가 철회되는 경우에 연속적으로 철회되지 않고 T의 실행 효과를 의미적으로 무효화 (semantically undo) 하기 위해 도입되는 트랜잭션이다. 따라서, CT의 실행이 T의 직간접 실행 효과의 물리적인 무효화(physical undo)를 보장하지는 않는다. 즉, CT 실행후의 데이타베이스 상태가 T가 실행되지 않은 상태와 동일함을 보상하지는 않고 의미(semantic)를 기초로 일관성 있는 데이타베이스 상태를 보장하게 된다.

보상 트랜잭션이 갖는 중요한 특성과 그 의미는 다음과 같다.

1. *보상 영속성* (persistence of compensation) [5, 6, 9]

보상 트랜잭션의 영속성이란 보상 트랜잭션이 일단 실행되면 성공적으로 종료되는 특성을 말한다. 분산 환경에서 이 특성이 갖는 의미는 보상 트랜잭션의 원자성을 보장하기 위해 완료 규약이 불필요하다는 것이다.

2. *의미 독립성* (independence of semantics) [2]

전역 트랜잭션 $T_i$의 전역 보상 트랜잭션 $CT_i$는 Ti의 부트랜잭션 $T_{ij}$ (j=1, 2,...,k)를 보상하기 위해 보상 부트랜잭션 $CT_{ij}$를 실행하게 된다. 이 경우, 부트랜잭션 $T_{ij}$가 서로 종속성을 갖는것과는 달리 보상 부트랜잭션 $CT_{ij}$는 서로 의미적으로 독립적인 부트랜잭션이라는 것이 의미 독립성이다. 이 특성은 실행을 종료한 보상 부트랜잭션은 다른 형제 보상 부트랜잭션과 독립적으로 로크를 반납할 수 있음을 의미하므로 보상 트랜잭션은 형제 부트랜잭션간에 기존의 분산 2PL 기법에 의해 동시성 제어가 되지 않게 된다.

3. *보상 원자성* (atomicity of compensation) [9]

보상 원자성이란 임의의 트랜잭션은 트랜잭션 T의 실행 결과를 액세스하거나 보상 트랜잭션 CT의 결과를 액세스하여야 한다는 것이다. 즉, 한 트랜잭션이 T의 결과와 CT의 보상결과를 동시에 액세스할 수 없다는 것이다.

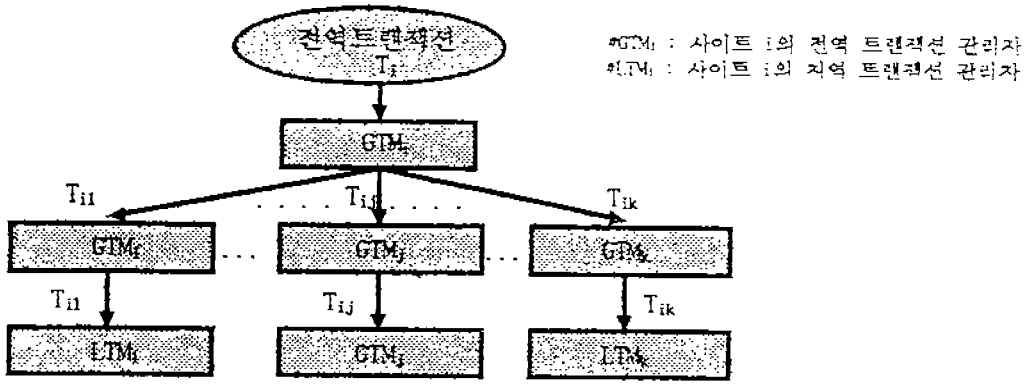
### 2.2 트랜잭션 관리 모형

본 논문에서 사용하고 있는 트랜잭션 관리 모형은 그림 1과 같다. 사이트 $S_1$, $S_2$,..., $S_k$의 데이타를 필요로 하는 전역 트랜잭션 $T_i$는 사이트 $S_i$의 전역 트랜잭션 관리자($GTM_i$)에 의해 부트랜잭션 $T_{i1}$, $T_{i2}$,..., $T_{ik}$로 분해되어 각 사이트의 전역 트랜잭션 관리자인

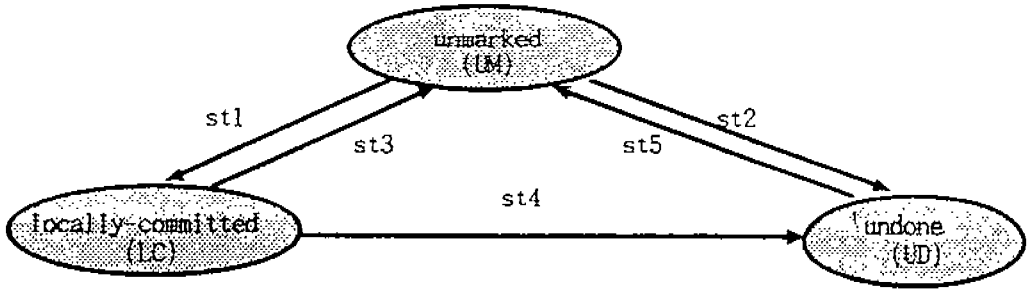
$GTM_1$, $GTM_2$,..., $GTM_k$로 제출된다. 부트랜잭션 $T_{ij}$ (j=1, 2...k)를 수신한 $GTM_j$는 $T_{ij}$를 사이트 $S_j$의 지역 트랜잭션 관리자($LTM_j$)에게 보내 실행을 시킨다. 뿐만 아니라, $GTM_j$는 $T_{ij}$의 상태정보를 유지하여 $T_{ij}$의 상태전이(그림 2 참조)가 일어나면 $GTM_i$ 에게 상태 전이를 알려 $GTM_i$가 전역 트랜잭션 $T_i$에 대한 상태를 유지할 수 있게 한다. 한편, 사이트 $S_j$에 있는 데이타만을 필요로 하는 지역 트랜잭션 (local transaction)은 $GTM_j$를 통하지 않고서 직접 $LTM_j$에게 제출된다.

전역 트랜잭션 $T_i$가 철회되는 경우 $T_i$가 실행되었던 각 사이트에는 전역 보상 트랜잭션(global compensating transaction) $CT_i$의 지역 보상 부트랜잭션 (local compensating subtransaction) $CT_{i1}$, $CT_{i2}$,..., $CT_{ik}$가 $T_i$의 실행 과정과 동일하게 제출되어 실행된다. 따라서, 지역 트랜잭션 관리자는 자신의 동시성 제어 기법을 사용하여 지역 트랜잭션, 부트랜잭션, 그리고 보상 부트랜잭션등을 스케줄링함으로써 지역 직렬성(local serializability)을 보장한다. 여기서, 전역 보상 트랜잭션 $CT_i$는 이미 정의가 되어 있으며 $CT_i$의 실행은 $T_i$의 실행 효과를 물리적 혹은 의미적으로 무효화 할 수 있다고 가정한다.

전역 트랜잭션 $T_i$가 철회된다는 것은 관련된 사이트 $S_j$(j=1, 2,..., k)중에 적어도 한 사이트는 '철회'로 투표했음을 의미한다. $CT_{ij}$(j=1, 2,..., k)를 수신한 $LTM_j$가 철회로 투표했었다면 해당 부트랜잭션 $T_{ij}$는 $LTM_j$에 의해 이미 회복(recovery) 기능에 의해 'UNDO'된 상황이므로 무시된다. 이와 달리, $LTM_j$가 완료로 투표했었다면 $T_{ij}$가 지역적 완료(locally-committed) 상태이므로 실제 보상 부트랜잭션이 실행되어 의미적 원자성(semantic atomicity)을 보장하게 된다.




(그림 1) 전역 트랜잭션 관리 모형
(Fig. 1) management model for global transactions




(그림 2) 트랜잭션 상태 전이도
(Fig. 2) transaction state transition diagram

## 3. 전역 트랜잭션의 상태전이

### 3.1 단일 사이트에서 상태전이 분석

전역 트랜잭션 $T_i$의 임의 사이트의 부트랜잭션 $T_{ij}$가 사이트 $S_j$에서 실행중에 갖을 수 있는 상태는 다음과 같은 3가지 상태중 하나이다. (그림 2 참조)

1. 'Unmarked(UM)'
부트랜잭션의 초기상태 혹은 종료상태(완료 혹은 철회)
2. 'Locally-Committed(LC)'
부트랜잭션이 지역적으로 완료된 상태
3. 'Undone(UD)'
부트랜잭션이 지역적으로 철회된 상태

그림 2에서 보는 바와 같이 사이트 $S_j$에서 임의의 부트랜잭션 $T_{ij}$가 임의의 한 상태에서 다른 상태로 전이할 수 있는 경우는 5가지이다. st1과 st2는 2PC의 투표단계에서 각각 '완료'와 '철회'로 투표한 경우에 일어나는 전이이며 st3와 st4는 지역적 완료 상태의 트랜잭션이 2PC의 결정단계에서 조정자로부터 최종 결정 메세지로 각각 '완료'와 '철회'를 받는 경우에 일어나는 전이이다. 한편, st5는 지역적으로 철회한 트랜잭션이 2PC의 투표단계에서 '철회'로 투표하여 st2 전이를 한 후에 2PC의 결정단계에서 조정자로부터 최종 메세지로 '철회'를 받는 경우 혹은, 지역적 완료 상태의 트랜잭션이 2PC의 결정단계에서 조정자로부터 최종 메세지로 '철회'를 받아 st4로 전이한 후에 일어나는 전이이다.

전역 트랜잭션 $T_i$의 부트랜잭션 $T_{ij}$가 실행되는 사이트 $S_j$에서 다른 전역 트랜잭션 $T_k$의 부트랜잭션 $T_{kj}$가 동시에 실행되는 경우 $T_i$와 $T_k$ 사이에 형성될 수 있는 사이트 $S_j$에서의 지역 직렬화 그래프(local serialization graph : LSG) 유형은 $T_{ij}$의 상태전이별로 다음과 같다.

1. St1(UM→LC) : $T_i \rightarrow T_k$
$T_{ij}$가 $T_{kj}$보다 먼저 실행을 시작했다면 2PL에 따라 $T_{ij}$가 소유한 로크를 반납한 후에 $T_{kj}$가 해당 로크를 얻어 실행이 가능하므로 사이트 Sj에서는 Ti가 Tk보다 먼저 실행된 결과가 성립된다.

2. St2(UM→UD) : $T_i \rightarrow CT_i \rightarrow T_k$
$T_{ij}$가 $T_{kj}$보다 먼저 실행을 시작했다면 $T_{ij}$의 실행효과를 undo한 후에(이것은 $CT_{ij}$를 실행시키는 것과 동일한 효과이므로 항상 $T_i \rightarrow CT_i$가 성립 즉, 보상 트랜잭션은 항상 해당트랜잭션의 실행뒤에 실행된 효과를 갖음) 소유한 로크를 반납하고 이후에 $T_{kj}$가 해당 로크를 얻어 실행될 수 있으므로 $T_i$가 $T_k$ 보다 먼저 실행된 결과가 성립된다.

3. St3(LC→UM) : $T_i \rightarrow T_k$
$T_{ij}$가 지역적으로 완료된 상태에서 소유한 로크를 반납한 후에 $T_{kj}$가 해당 로크를 얻어 실행될 수 있으므로 $T_i$가 $T_k$ 보다 먼저 실행된 결과가 성립된다.

4. St4(LC→UD) : $T_i \rightarrow T_k \rightarrow CT_i$
$T_{ij}$가 지역적으로 완료된 상태에서 소유한 로크를 반납한 후에 $T_{kj}$가 해당 로크를 얻어 실행될 수 있으므로 $T_i$가 $T_k$ 보다 먼저 실행된 결과가 성립된다. 한편, $T_{ij}$가 'UD'상태로 전이하므로 $CT_{ij}$의 실행이 수반되어야 한다. $CT_{ij}$의 실행은 2PL에 따라 $Tk_j$가 소유한 로크를 반납한 후에 가능하므로 $T_k$가 $CT_i$ 보다 먼저 실행된 결과가 성립된다.

5. St5(UD→UM) : $T_i \rightarrow CT_i \rightarrow T_k$ 혹은 $T_i \rightarrow T_k \rightarrow CT_i$
$T_{ij}$가 'UD' 상태로 전이된 경우는 st2나 st4 전이를 통해서이다.
경우1) st2를 통한 전이인 경우

〈표 1〉 지역 부트랜잭션 $T_{ij}$의 상태전이에 따른 지역 직렬화 그래프

〈Table 1〉 local serialization graph of local subtransaction $T_{ij}$ for each state transition

| 상태 전이 | 지역 직렬화 그래프 |
|---|---|
| St1 | $T_i \rightarrow T_k$ |
| St2 | $T_i \rightarrow CT_i \rightarrow T_k$ |
| St3 | $T_i \rightarrow T_k$ |
| St4 | $T_i \rightarrow T_k \rightarrow CT_i$ |
| St5 경우 1)<br>경우 2) | $T_i \rightarrow CT_i \rightarrow T_k$<br>$T_i \rightarrow T_k \rightarrow CT_i$ |

st2 전이를 내포하므로 $T_i \rightarrow CT_i \rightarrow T_k$가 성립된다.

경우2) st4를 통한 전이인 경우

st4 전이를 내포하므로 $T_i \rightarrow T_k \rightarrow CT_i$가 성립된다.

〈표 1〉은 $T_{ij}$와 $T_{kj}$가 $S_j$에서 동시에 실행되는 경우 $T_{ij}$의 상태전이별로 $T_i$와 $T_k$ 사이에 형성되는 지역 직렬화 그래프를 보여주고 있다.

### 3.2 다중 사이트에서 상태전이 조합의 분석

이 절에서는 한 전역 트랜잭션의 부트랜잭션들에 대해 각 사이트에서 고려할 수 있는 상태전이 조합을 분석하여 양립할 수 있는 상태전이 조합을 유도한다. 임의의 두 사이트 $S_j$와 $S_m$에서 실행되는 임의의 전역 트랜잭션 $T_i$를 고려하여도 일반성을 잃지 않으므로 여기서는 이 경우를 고려한다.

사이트 $S_j$와 $S_m$에서 실행되는 $T_i$의 부트랜잭션을 각각 $T_{ij}$와 $T_{im}$이라 할 때, 사이트 $S_j$와 $S_m$에서 고려할 수 있는 $T_i$의 상태전이 조합은 다음과 같다. 여기서 상태전이 조합 〈a, b〉는 〈사이트 $S_j$에서 $T_{ij}$의 상태전이, 사이트 $S_m$에서 $T_{im}$의 상태 전이〉를 의미한다.

C1 : 〈St1, St1〉

$T_{ij}$와 $T_{im}$이 2PC의 투표단계에서 모두 '완료'로 투표한 경우의 전이이므로 양립할 수 있는 상태전이 조합이다.

C2 : 〈St1, St2〉

$T_{ij}$와 $T_{im}$이 2PC의 투표단계에서 각각 '완료'와 '철회'로 투표한 경우의 전이이므로 양립할 수 있는 전이 조합이다.

C3 : 〈St1, St3〉

St1과 St3는 각각 2PC의 투표단계와 결정단계에서 일어나는 전이이므로 양립할 수 없는 상태전이 조합이다.

C4 : 〈St1, St4〉

St1과 St4는 각각 2PC의 투표단계와 결정단계에서 일어나는 전이이므로 양립할 수 없는 상태전이 조합이다.

C5 : 〈St2, St2〉

$T_{ij}$와 $T_{im}$이 2PC의 투표단계에서 모두 '철회'로 투표한 경우의 전이이므로 양립할 수 있는 상태전이 조합이다.

C6 : 〈St2, St3〉

St2와 St3는 각각 2PC의 투표단계와 결정단계에서 일어나는 전이이므로 양립할 수 없는 상태전이 조합이다.

C7 : 〈St2, St4〉

St2와 St4는 각각 2PC의 투표단계와 결정단계에서 일어나는 전이이므로 양립할 수 없는 상태전이 조합이다.

C8 : 〈St3, St3〉

$T_{ij}$와 $T_{im}$이 2PC의 결정단계에서 $T_i$가 완료하는 경우에 일어날 수 있는 전이이므로 양립할 수 있는 가능한 상태전이 조합이다.

C9 : 〈St3, St4〉

2PC의 결정단계에서 2PC 원칙에 따라 St4 전이가 일어나면 St3 전이는 발생할 수 없으므로 양립할 수 없는 상태전이 조합이다.

C10 : 〈St4, St4〉

$T_{ij}$와 $T_{im}$이 2PC의 결정단계에서 $T_i$가 철회되는 경우에 일어날 수 있는 전이이므로 양립할 수 있는 상태전이 조합이다.

C11 : 〈St3, St5〉

2PC의 결정단계에서 2PC 원칙에 따라 St5 전이가 일어나면 St3 전이는 발생할 수 없으므로 양립할 수 없는 상태전이 조합이다.

C12 : 〈St5, St5〉

$T_{ij}$와 $T_{im}$이 2PC의 결정단계에서 $T_i$가 철회되는 경우에 일어날 수 있는 전이이므로 양립할 수 있는 상태전이 조합이다.

C13 : 〈St5, St4〉

$T_{ij}$와 $T_{im}$이 2PC의 결정단계에서 $T_i$가 철회되는 경우에 일어날 수 있는 전이이므로 양립할 수 있는 상태전이 조합이다.

C14 : 〈St1, St5〉

St1과 St5는 각각 2PC의 투표단계와 결정단계에서 일어나는 전이이므로 양립할 수 없는 상태전이 조합이다.

C15 : 〈St5, St2〉

St5와 St2는 각각 2PC의 결정단계와 투표단계에서 일어나는 전이이므로 양립할 수 없는 상태전이 조합이다.

이상에서 양립할 수 있는 상태전이 조합은 C1, C2,

:5, C8, C10, C12, C13 임을 알 수 있다.

## 4. 전역 트랜잭션 관리 규약

### 4.1 필요성

낙관적 2PC하에서 사이트 $S_j$에서 전역 트랜잭션 $T_i$의 보상 부트랜잭션 $CT_{ij}$가 실행된다는 것은 $S_j$에서 부트랜잭션 $T_{ij}$가 지역적 완료상태에서 $T_i$가 철회됨을 의미한다. 한편, $T_{ij}$가 지역적 완료상태라는 것은 $T_{ij}$가 소유한 로크를 조기 반납하여 동시에 실행되는 다른 트랜잭션들이 $T_{ij}$의 실행결과를 액세스할 수 있음을 의미한다. 이 경우 $T_i$의 철회는 보상 부트랜잭션 $CT_{ij}$의 실행을 유발한다.

**Theorem 1**: 한 전역 트랜잭션의 보상 트랜잭션과 다른 전역 트랜잭션의 동시 실행은 보상 트랜잭션의 보상 원자성 위배를 유발할 수 있다.

증명) 전역 트랜잭션 $T_i$의 임의의 사이트 $S_j$에서 부트랜잭션 $T_{ij}$의 보상 부트랜잭션 $CT_{ij}$가 실행된다고 하자. $S_j$에서 $T_{ij}$의 실행 결과를 이미 액세스 할 수 있었던 트랜잭션들은 지역 트랜잭션 $LT_j$, 다른 전역 트랜잭션 $T_k$의 보상 부트랜잭션 $CTk_j$와 부트랜잭션 $Tk_j$이다.

경우1) $LT_j$가 액세스 한 경우

$LTM_j$의 동시성 제어에 의해 스케줄링되었으므로 지역 직렬성이 보장되며 지역 트랜잭션과의 동시 실행이므로 전역 직렬성과는 무관하다.

경우2) $CTk_j$가 액세스 한 경우

보상 부트랜잭션은 의미 독립성을 갖으므로 보상 트랜잭션들간의 전역 직렬성과는 무관하며 $CTk_j$와 $CT_{ij}$사이의 지역 직렬성은 $LTM_j$에 의해 보장된다.

경우3) $Tk_j$가 액세스 한 경우

$S_j$에서 지역 직렬화 그래프는 $T_{ij} \rightarrow T_{kj} \rightarrow CT_{ij}$가 된다. 한편, $T_k$의 임의의 사이트 $S_m$에서의 부트랜잭션을 $T_{km}$이라 하자. $S_m$에서는 $LTM_m$에 의해 $T_{km}$이 $T_i$의 보상 부트랜잭션 $CT_{im}$ 보다 늦게 스케줄링 되었다면 Sm에서의 지역 직렬화 그래프는 $T_{im} \rightarrow CT_{im} \rightarrow T_{km}$가 된다. 따라서, 전역 직렬화 그래프에서 $CT_i$와 $T_k$ 사이에 사이클이 형성되어 보상 트랜잭션 $CT_i$의 보상 원자성이 위배된다. □

Theorem 1 로부터 낙관적 2PC하에서 보상 트랜잭션을 도입하는 경우에 전역 트랜잭션의 실행에 대한 관리 규약이 필요하게 됨을 알 수 있다. 즉 트랜잭션의 실행이 전역 직렬화 그래프에 사이클을 형성하게 하는 경우 관련된 다른 트랜잭션의 실행이 종료될 때까지 트랜잭션의 실행을 지연시켜 사이클이 형성되는 것을 방지하는 규약이 필요하다.

### 4.2 관리 규약

전역 트랜잭션 $T_i$를 실행시키는 경우 가장 제한이 없는 규약은 $T_i$가 실행될 사이트에서 실행중인 임의의 다른 전역 트랜잭션 $T_j$의 각 사이트에서의 현재 상태를 고려하지 않고 무조건 실행을 시키는 것이다. 이와 반대로, 가장 제한적인 규약은 $T_j$의 현재 상태가 모두 동일한 상태인 경우에만 $T_i$를 실행시키는 규약이 가장 제한적인 규약이 된다. 따라서, 위의 두가지 극단적인 규약을 바탕으로 아래와 같은 규약들을 고려할 수 있다.

규약: 전역 트랜잭션 $T_i$가 실행될 사이트들에서 실행중인 임의의 전역 트랜잭션 $T_j$의 부트랜잭션들의 상태는

A 그룹:
P1. 모두 'UM'이어야 한다.
P2. 모두 'LC'이어야 한다.
P3. 모두 'UD'이어야 한다.

B 그룹:
P4. 모두 'UM'이거나, 혹은 모두 'LC'이어야 한다.
P5. 모두 'UM'이거나, 혹은 모두 'UD'이어야 한다.
P6. 모두 'LC'이거나, 혹은 모두 'UD'이어야 한다.

C 그룹:
P7. 모두 'UM'이거나, 혹은 모두 'UD'이거나, 혹은 모두 'LC'이어야 한다.

D 그룹:
P8. 모두 'LC'이거나 'UD'이어야 한다.
P9. 모두 'UM'이거나 'UD'이어야 한다.
P10. 모두 'UM'이거나 'LC'이어야 한다.

E 그룹:
P11. 모두 'UM'이거나, 혹은 모두 'LC'이거나 'UD'이어야 한다.

P12. 모두 'LC'이거나, 혹은 모두 'UM'이거나 'UD' 이어야 한다.
P13. 모두 'UD'이거나, 혹은 모두 'UM'이거나 'LC' 이어야 한다.
F 그룹:
P14. 모두 'UM'이거나 'LC'이거나 'UD'이어야 한다.

위 규약중에서 A 그룹에서 F 그룹으로 갈수록 전역 트랜잭션의 실행 제한이 완화된 규약임을 알 수 있다. 즉, 전역 트랜잭션의 동시실행 정도가 그 만큼 증가되는 규약임을 알 수 있다.

### 4.3 관리 규약의 정당성

이 절에서는 앞에서 고려한 14개의 전역 트랜잭션 관리 규약들이 전역 직렬성을 위배하지 않는 정당한 규약인지를 논의한다.

4.2절에서 고려한 규약들을 검토해 보면 전역 트랜잭션 $T_i$의 부트랜잭션들이 실행될 사이트들에서 실행중인 임의 전역 트랜잭션 $T_j$의 부트랜잭션들의 상태가 동일해야 하는 경우와 그렇지 않은 경우로 구분됨을 쉽게 알 수 있다.

**Lemma 1**: 전역 트랜잭션 $T_j$의 부트랜잭션들의 상태가 모두 동일하면 다른 전역 트랜잭션 $T_i$를 동시에 실행하여도 전역 직렬성은 보장된다.

증명) 임의의 두 사이트 $S_k$, $S_m$에서 실행중인 두 부트랜잭션 $T_{jk}$와 $T_{jm}$를 고려하여도 일반성을 잃지 않으므로 두 개의 부트랜잭션만을 고려하자. $T_{jk}$와 $T_{jm}$의 동일한 상태는 다음 3가지이다.

경우 1. 모두 'UM'인 경우

고려할 수 있는 $T_{jk}$와 $T_{jm}$의 상태전이는 각기 St1과 St2 두 개씩이다. 따라서, $T_{jk}$와 $T_{jm}$ 사이에 고려할 수 있는 상태전이 조합은 C1:⟨St1, St1⟩, C2⟨St1, St2⟩, C5⟨St2, St2⟩가 되며 이들 모두 양립할 수 있는 상태전이 조합이다(3.2절 참조). 한편, $S_k$와 $S_m$에서 $T_i$의 부트랜잭션이 동시에 실행되는 경우에 형성되는 지역 직렬화 그래프의 조합은 다음과 같으며(표 1 참조) 각 상태전이 조합에서 전역 직렬화 그래프에서 사이클이 형성되지 않는다.

C1: $\langle T_j \rightarrow T_i,\ T_j \rightarrow T_i \rangle$
C2: $\langle T_j \rightarrow T_i,\ T_j \rightarrow CT_j \rightarrow T_i \rangle$
C5: $\langle T_j \rightarrow CT_j \rightarrow T_i,\ T_j \rightarrow CT_j \rightarrow T_i \rangle$

경우 2. 모두 'LC'인 경우

고려할 수 있는 $T_{jk}$와 $T_{jm}$의 상태전이는 각기 St3과 St4 두 개씩이다. 따라서, $T_{jk}$와 $T_{jm}$ 사이에 고려할 수 있는 상태전이 조합은 C8:⟨St3, St3⟩, C9⟨St3, St4⟩, C10⟨St4, St4⟩가 되지만 실제로 양립할 수 있는 상태전이 조합은 C8과 C10 뿐이다(3.2절 참조). 한편, $S_k$와 $S_m$에서 $T_i$의 부트랜잭션이 동시에 실행되는 경우에 형성되는 지역 직렬화 그래프의 조합은 다음과 같으며(표 1 참조) 각 상태전이 조합에서 전역 직렬화 그래프에서 사이클이 형성되지 않는다.

C8: $\langle T_j \rightarrow T_i,\ T_j \rightarrow T_i \rangle$
C10: $\langle T_j \rightarrow T_i \rightarrow CT_j,\ T_j \rightarrow T_i \rightarrow CT_j \rangle$

경우 3: 모두 'UD'인 경우

고려할 수 있는 $T_{jk}$와 $T_{jm}$의 상태전이는 각기 St5 뿐이다. 따라서, $T_{jk}$와 $T_{jm}$ 사이에 고려할 수 있는 상태전이 조합은 C12:⟨St5, St5⟩가 되며 양립할 수 있는 상태전이 조합이다(3.2절 참조). 한편, 부트랜잭션이 'UD' 상태로 전이될 수 있는 경로는 st2나 st4 전이를 통해서이므로 $S_k$와 $S_m$에서 $T_i$의 부트랜잭션이 동시에 실행되는 경우에 다음과 같이 3가지 경우로 지역 직렬화 그래프의 조합을 고려할 수 있으며 (표 1 참조) 전역 직렬화 그래프에서 사이클이 형성되지 않는다.

1. $T_{jk}$와 $T_{jm}$ 모두 st2 전이를 통한 경우
C12:⟨St5 경우1, St5 경우1⟩가 되므로
C12: $\langle T_j \rightarrow CT_j \rightarrow T_i,\ T_j \rightarrow CT_j \rightarrow T_i \rangle$가 된다.

2. $T_{jk}$와 $T_{jm}$ 모두 st4 전이를 통한 경우

st4전이는 'LC' 상태에서 'UD' 상태로의 전이이다. 모든 부트랜잭션이 'LC' 상태에서는 2PC 결정단계에서 최종 메세지가 '완료'이므로 'UD' 상태로의 전이는 일어날 수 없다. 따라서, 모두 st4 전이를 통하는 경우는 일어나지 않는다.

3. $T_{jk}$와 $T_{jm}$이 각각 st2와 st4 전이를 통한 경우

이 경우는 $T_{jk}$와 $T_{jm}$이 각각 'UM'과 'LC' 상태에서 전이를 하는 경우이다.

따라서, st2와 st4는 각각 2PC의 투표단계와 결정단계에서 일어나는 전이 이므로 st2와 st4는 양립할 수 없는 전이이다.

경우 1) 2) 3)으로부터 부트랜잭션들의 상태가 동일한 경우 다른 전역 트랜잭션을 동시에 실행하여도 전역 직렬성은 보장된다. □

**Lemma 2**: 전역 트랜잭션 $T_j$의 부트랜잭션들의 상태가 모두 동일하지 않으면 다른 전역 트랜잭션 $T_i$의 동시 실행은 전역 직렬성을 보장할 수 없다.

증명) 임의의 두 사이트 $S_k$, $S_m$에서 실행중인 두 부트랜잭션 $T_{jk}$와 $T_{jm}$를 고려하여도 일반성을 잃지 않으므로 두 개의 부트랜잭션만을 고려하자. $T_{jk}$와 $T_{jm}$의 상태가 다른 경우는 다음 3가지이다.

경우 1: 'UM'과 'LC'인 경우

상태가 'UM'인 부트랜잭션의 경우 고려할 수 있는 상태전이는 St1과 St2이며 'LC'인 경우는 St3과 St4이다. 따라서, $T_{jk}$와 $T_{jm}$ 사이에 고려할 수 있는 상태전이 조합은 C3:⟨St1, St3⟩, C4:⟨St1, St4⟩, C6:⟨St2, St3⟩, C7:⟨St2 ,St4⟩가 되지만 이들 모두 양립할 수 없는 상태전이 조합이다 (3.2절 참조).

경우 2: 'UM'과 'UD'인 경우

상태가 'UM'인 부트랜잭션의 경우 고려할 수 있는 상태전이는 St1과 St2이며 'UD'인 경우는 St5이다. 따라서, $T_{jk}$와 $T_{jm}$ 사이에 고려할 수 있는 상태전이 조합은 C14:⟨St1, St5⟩, C15:⟨St2, St5⟩가 되지만 이들 모두 양립할 수 없는 상태전이 조합이다(3.2절 참조).

경우 3: 'LC'와 'UD'인 경우

상태가 'LC'인 부트랜잭션의 경우 고려할 수 있는 상태전이는 St3과 St4이며 'UD'인 경우는 St5이다. 따라서, $T_{jk}$와 $T_{jm}$ 사이에 고려할 수 있는 상태전이 조합은 C11:⟨St3, St5⟩, C13:⟨St4, St5⟩가 되지만 실제로 양립할 수 있는 상태전이 조합은 C13 뿐이다(3.2절 참조). 그리고, lemma1 의 경우 3)에서 기술한 바에 따르면 이 경우에 'UD' 상태로 올 수 있는 경로는 st2를 통해서이므로 상태전이 조합 C13은 C13:⟨St4, St5 경우1⟩가 된다. 따라서, $S_k$와 $S_m$에서 $T_i$의 부트랜잭션이 동시에 실행되는 경우에 형성되는 지역 직렬화 그래프의 조합은 다음과 같으며(표 1 참조) 전역 직렬화 그래프에서 사이클이 형성된다.

C13:⟨$T_j \rightarrow T_i \rightarrow CT_j$, $T_j \rightarrow CT_j \rightarrow T_i$⟩

경우 1) 2) 3)으로부터 부트랜잭션들의 상태가 모두 동일하지 않으면 다른 전역 트랜잭션의 동시 실행은 전역 직렬성을 보장할 수 없다. □

**Theorem 2**: 전역 트랜잭션 $T_j$의 부트랜잭션들이 'LC'와 'UD' 상태가 공존하는 경우 이외에는 다른 전역 트랜잭션 $T_i$를 동시에 실행하여도 전역 직렬성은 보장된다.

증명) **Lemma 1**과 **Lemma 2**로부터 당연하다. □

4.2절에서 고려한 규약들중에서 부트랜잭션들의 상태가 'LC'인 경우와 'UD'인 경우가 공존하는 조건을 갖는 관리 규약 은 P6, P11, P14이다. 따라서, 이들 3가지 규약을 제외한 나머지 11개의 규약은 Theorem 2로부터 정당한 규약임을 알 수 있다.

## 5. 결 론

본 논문에서는 전역 트랜잭션의 원자성을 보장하기 위해 비블록킹 규약인 낙관적 2PC를 사용하는 분산 환경에서 전역 직렬성 보장이 가능한 11가지의 전역 트랜잭션 관리 규약을 제안하였다. 제안된 규약들은 전역 트랜잭션을 실행시키기 전에 동시에 실행되는 다른 전역 트랜잭션의 상태를 검사하여 전역 직렬성의 위배 가능성이 존재하면 다른 전역 트랜잭션이 종료될 때까지 대기하게 하여 전역 트랜잭션의 실행을 지연한다. 상태분석으로부터 전역 직렬성의 위배 가능성이 존재할 조건은 동일한 전역 트랜잭션의 부트랜잭션들의 상태가 'UD' 상태와 'LC' 상태로 공존하는 경우임을 알 수 있다. 한편, 제안된 규약들은 전역 트랜잭션의 실행 조건의 차이로 특징지워지며 이

특징은 전역 트랜잭션의 동시 실행 정도의 차이를 의미한다.

## 참 고 문 헌


[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in a Database Systems, Addison-Wesley, Reading, MA, 1987.

[2] W. Du and A. K. Elmagarmid, "Quasi-Serializability: a Correctness Criterion for Global Concurrency Control in Interbase," Proc. of 15th Int. Conf. on Very Large Data Bases, pp. 347-355, 1989.

[3] A. K. Elmagarmid and C. Pu, Special Issue on Heterogeneous Databases, ACM Computing Surveys, Vol. 22, No. 3, Sep. 1990.

[4] K. Eswaran, J. Gray, R. Lorie, and I. Traiger, "The Notions of Consistency and Predicate Loks in a Database Systems," Comm. of ACM, Vol. 19, No. 11, pp. 624-633, Nov. 1976.

[5] H. Garcia-Molina, "Using Semantic Knowledge for Transaction Processing in a Distributed Database," ACM Tran. on Database Systems, Vol. 8, No. 2, pp. 186-213, June 1983.

[6] H. Garcia-Molina and K. Salem, "Sagas," Proc. of Int. Conf. on Management of Data, pp. 249-259, 1987.

[7] J. N. Gray, "Notes on Database Operating Systems, Lecture Notes in Computer Science, Operating Systems: An Advanced Course," Vol. 60, pp. 393-481, Springer-Verlag, 1978.

[8] T. Haerder and A. Reuter, "Principles of Transaction-Oriented Database Recovery," ACM Computing Surveys, Vol. 15, No. 4, pp. 288-316, Dec. 1983.

[9] H. F. Korth, E. Levy, and A. Silberschatz, "A Formal approach to Recovery by Compensating Transactions," Proc. of the 16th Int. Conf. on Very Large Data Bases, pp. 95-106, 1990.

[10] E. Levy, H. F. Korth, and A. Silberschatz, "An Optimistic Commit Protocol for Distributed Transaction Management," Proc. of Int. Conf. on Management of Data, pp. 88-97, 1991.

[11] N. Lynch, M. Merritt, W. Weihl, and A. Fekete, Atomic Transactions, Morgan Kaufmann Publishers, 1994.

[12] D. Skeen, "Non-Blocking Commit Protocols," Proc. of Int. Conf. on Management of Data, pp. 133-147, 1982.

[13] G. Weikum, "A Theoretical Foundations of Multi-level Concurrency Control," Proc. of ACM SIGACT-SIGMOD: Symposium on Principes of Database Systems, pp. 31-42, Mar. 1986.

[14] G. Weikum and H. J. Scheck, "Architectural Issues of Transaction Management in Layered Systems," Proc. of 10th Int. Conf. on Very Large Data Bases, pp. 454-465, Aug. 1984.

[15] G. Weikum and C. Hasse, "Multi-Level Transaction Management for Complex Objects: Implementation, Performance, Parallelism," VLDB Journal, Vol. 2, pp. 407-453, 1993.



**신 동 천**

1985년 2월 서울대학교 컴퓨터공학과 졸업(학사)
1987년 2월 한국과학기술원 전산학과 졸업(공학석사)
1991년 2월 한국과학기술원 전산학과 졸업(공학박사)
1991년 1월~1993년 2월 한국전산원 선임연구원
1993년 3월~현재 중앙대학교 산업정보학과 조교수
관심분야: 데이타베이스 시스템 및 설계, 정보 검색