# 실시간 데이타 처리를 위한 확장 가능한 트랜잭션 모델에 관한 연구

# An Extensible Transaction Model for Real-Time Data Processing

문 승 진*
Seung-Jin Moon

## 요 약

본 논문은 실시간 트랜잭션 시스템(Real-Time Transaction System)에 하위 트랜잭션(subtransaction) 개념을 도입한 새로운 확장모델을 제시하였다. 제안된 모델은 J. Moss 모델을 실시간 단일 프로세스에 특정한 시간제약을 부과함으로 확장되었으며, 이를 기반으로 통합된 동시성 제어와 스케쥴링 알고리즘이 개발되었다. 이는 Sha의 우선 순위 제한 알고리즘에 기반하여 확장된 알고리즘으로, 실시간 트랙잭션의 시간제약을 보장함과 동시에 데이터베이스의 일관성도 함께 유지한다. 본 논문은 제안된 실시간 중첩 트랜잭션 모델이 무한정한 블록킹(blocking)과 데드락(dead lock)을 방지함과 동시에 실시간 트랜잭션의 직렬화도 유지함을 증명하였으며, 또한 트랜잭션의 상위 바운드를 설정하고, 고정 우선순위 기반 방법(Rate-Monotonic Priority Assignment)을 적용함으로 스케쥴링 가능성을 분석하였다. 본 연구는 다중및 분산 실시간 중첩 트랜잭션 모델로 확장하기 위한 첫단계이며, 또한 최근 관심을 모으는 웹기반 실시간 멀티미디아 데이터베이스 모델로 확장이 가능한 것으로 추정된다.

## Abstract

In this paper we present a new extensible model based upon the concept of subtransactions in real-time transaction systems. The nested transaction model originally proposed by J. Moss is extended for real-time uniprocessor transaction systems by adding explicit timing constraints. Based upon the model, an integrated concurrency control and scheduling algorithm is developed, that not only guarantees timing constraints of a set of real-time transactions but also maintains consistency of the database. The algorithm is based on the priority ceiling protocol of Sha *et al.* We prove that the Real-Time Nested Priority Ceiling Protocol prevents unbounded blocking and deadlock, and maintains the serializability of a set of real-time transactions. We use the upper bound on the duration that a transaction can be blocked to show that it is possible to analyze the schedulability of a transaction set using rate-monotonic priority assignment. This work is viewed as a step toward multiprocessor and distributed real-time nested transaction systems. Also, it is possible to be extended to include the real-time multimedia transactions in the emerging web-based database application areas.

## 1. 서 론

본 논문은 현재 실시간 데이타베이스에서 일반적으로 가정하는 "비계층적(flat)" 트랜잭션을 확장하여 단일 프로세서에 적용함과 동시에 다중및 분산시스템으로 발전될 수 있는 새로운 모델을 제시하였다. 비계층적 트랜잭션은 한 개의 제어 쓰레드(thread)를 가지고 있고, 스케쥴링, 동시성 제어(concurrency control), 고장회복(fault recovery) 때 하나의 단위로 관리된다. 이러한 간단한 구조의 비계층적 트랜잭션은 복잡하고 역동적인 시스템[1]에서 요구하는 유연성과 성능을 제공하지 않는다. 실시간 시스템은 점점 더 복잡해지고, 더욱 분산된 형태를 취하고 있으며, 특히 다중 프로세서및 분산 실시간 데이타베이스 시스템에서 요구되는 안정된 동작을 수행하기 위해 강력하고, 유연한 제어 구조를 필요로 한다. 이를 위해 동시성 제어와 복구를 세분화 시킬 수 있으며, 섬세한 제어를 할 수 있는 새로운 형태의 실시간 트랜잭션

* 종신회원 : 수원대학교 컴퓨터과학과 조교수
sjmoon@mail.suwon.ac.kr

모델이 필요하게 되었다. 전통적인 데이타베이스에서 이런 문제들에 대한 부분적인 해결책으로 J. Moss가 제안한 중첩 트랜잭션을 이용하는데, 이를 이용하면 단일 트랜잭션이 여러 개의 하위 트랜잭션 계층으로 분리되며, 하위 트랜잭션은 여러 개의 분산된 프로세서에서 병렬적으로 수행되거나, 단일 프로세서에서 독립된 하나의 단위로 동시에 스케줄링되고 실행될 수 있다[2].

본 논문에서는 중첩 트랜잭션의 이러한 특징(예로, 계층적인 분할과 통합, 그리고 섬세한 고장회복)을 이용하여 기존 실시간 트랜잭션에 통합 적용되어질 수 있는 단일 프로세서에 기반을 둔 새로운 실시간 중첩 트랜잭션 모델(Real-Time Nested Transaction Model, RT-NTM)을 제안하였다. 이 모델은 기존 중첩 트랜잭션의 개념을 확장하여 실시간 시스템 분야에서, 특히 데이터 처리에 관한 유용성을 향상시키며 비계층적 트랜잭션모델을 포함함과 동시에 중첩 트랜잭션의 장점을 가지고 있기 때문에, 실시간 데이타베이스의 연구 영역을 확장시키는 결과를 가져왔다. 본 논문에서는 단일 프로세서상의 중첩 트랜잭션의 실시간 스케줄링의 문제에 관해서 촛점을 맞추었으며, 제안된 모델은 추후 병렬 및 분산처리를 가능케 할 단계로 발전될 수 있으며 또한 웹기반 실시간 멀티미디어 데이타베이스로 확장이 가능하리라 추정된다.

## 2. 시스템 모델

### 2.1 실시간 중첩 트랜잭션 모델(Real-Time Nested Transaction Model, RT-NTM)

본 연구에서 사용된 모델은 J. Moss[2]에 의해 처음 제안된 중첩트랜잭션 모델을 확장한 것으로 다음의 개념 및 용어들로 설명된다. 트랜잭션 루틴(transaction routine)은 데이타베이스안에서 변환을 수행할 알고리듬과 그 알고리듬이 필요로 하는 자원을 포함하며, $P$와 $P_i$를 사용하여 트랜잭션 루틴을 나타낸다. 트랜잭션 인스턴스(transaction instance) $T$는 트랙잭션 루틴 $P$의 실행요청으로 트랜잭션 루틴 $P$의 실행은 도착시간 후의 특정 시점에 시작되며 $T$ 또는 $T_i$로 트랜잭션 인스턴스를 나타낸다. 실시간 트랜잭션은 자원(resources)을 이용하며, 자원에 대한 접근은 락(lock)에 의해서 제어되어지며 여러가지 종류의 락이 상호 충돌하는 관계를 갖게된다. 락을 획득하고 이를 해제하는 순서는 2단계 락킹 프로토콜(Two-Phase Locking Protocol, 2PL, [3])과 같이 미리 정의된 프로토콜을 따르는데, 이는 모든 트랜잭션은 어떤 락을 해제하기 전에 반드시 필요한 모든 락을 획득하고, 작업을 마치면 모든 락을 해제해야 하는 규약을 말한다. 초기에 모든 락은 해제(free)인 상태이며, 트랜잭션은 락을 획득한 후 해제시킬 때가지 계속해서 락을 유지(hold)하게 된다. 트랜잭션은 락을 유지하는 것보다 하위 개념인 락을 보유(retain)할 수 있으며 트랜잭션이 더 이상 락을 유지할 필요가 없을 때, 락을 해제시키는 대신 그러한 락을 부모 트랜잭션에 전달한다. 부모 트랜잭션은 이러한 락을 해제될 때까지 보유하게 된다. 본 연구에서는 오직 리프(leaf) 트랜잭션만이 직접적으로 데이타베이스 자원을 사용하고 동작(operations)을 수행한다고 가정하며, 만약 리프 트랜잭션이 아닌 트랜잭션이 자원을 사용하고자 한다면, 먼저 자식 트랜잭션을 생성하여, 생성된 자식 트랜잭션이 동작을 수행토록 하는 간접적인 방식을 택한다. 부모 트랜잭션이 자식 트랜잭션과 자원을 락하거나 언락(unlock) 하는 동작을 동시에 수행한다고 가정하므로, 부모 트랜잭션이 유지하고 있는 락을 자식 트랜잭션이 요청하는 일은 없으며, 그래서 부모와 자식 사이에 데드락은 존재하지 않는다. 자식 트랜잭션은 락을 유지하기 위해 그 부모 트랜잭션이 보유하고 있는 락을 요청할 수 있지만, 부모 트랜잭션은 자원을 '락/언락' 하는 직접적인 데이타베이스 동작을 수행하지 않는다고 가정한다. 부모 트랜잭션은 자식 트랜잭션에 의해 보유된 락을 요청할 수 없으며, 부모 트랜잭션은 자식 트랜잭션의 수행결과를 요구하고, 그 때문에

자식 트랜잭션이 모든 동작을 끝낼 때가지 기다려야 하므로, 대부분의 응용 데이타베이스에서 보편적 가정으로 이해되고 있다. 또한 형제 프로세스(Sibling Processes)사이에 선행관계(Precedence Relationship)는 존재하지 않는다고 가정하며, 트랜잭션 인스탄스 $T_i$는 어떤 자원에 대한 락을 유지하는 동안 수행동작을 자발적으로 지연하지 않고, $T_i$가 락을 유지하고 있는 동안에 같은 우선권을 지니고 있는 트랜잭션은 시분할형식(Time-Slicing)으로 스케쥴링되지 않는다고 가정하는데, 이러한 가정은 $T_i$가 다른 트랜잭션을 블록(block) 할 수 있는 시간을 제한하기 위해 필요하다. 스케줄링 분석(Scheduling Analysis)에서 우리는 실시간 트랜잭션 시스템이 단일 프로세서 상에서 간단한 주기적인 프로세스(Periodic Process)의 고정된 단위의 일을 실행한다고 가정한다. 개개의 주기적인 프로세스는 유일한 루트(root) 트랜잭션 루틴 $P_i$에 해당하며 $P_i$의 인스턴스인 $T_i$는 무한으로 발생하는 과정중 특정한 것을 지칭한다. 각각의 트랜잭션 $T_i$는 $T_i$ 대한 대체 트랜잭션(Substitute Transaction) $T'_i$가 있다고 가정하며, $T_i$가 리프 트랜잭션일 경우 $T'_i$ 가 대체 트랜잭션이 된다. $T_i$가 리프의 하위 트랜잭션이 아닌 경우 $T'_i$ 는 중단(abort) 되어야할 $T_i$의 하위 트랜잭션을 대체하는 작업을 한다. 모든 주기적인 프로세스 $P_i$는 $P_i$의 트랜잭션과 하위 트랜잭션에 기본 우선권(Base Priority), $p(P_i)$가 정적으로 할당되어지며, 각각의 트랜잭션 $T_i$는 기본 우선권 $p(T_i)=p(P_i)$를 가지며, 몇 개의 동적으로 할당되어진 계승 우선권(Inherited Priority)이 부여되어진다. 트랜잭션 인스탄스의 활동 우선권(Active Priority), $P^*_t(T_i)$는 이렇게 부여된 기본 우선권 중 최대치를 가지며, 이는 주어진 시간 $t$ 안에 계승된 (inherited) 모든 우선권을 지칭한다. 본 논문에서 하위 트랜잭션은 루트 트랜잭션의 기본 우선권을 물려받고, 트랜잭션 계층도(Transaction Hierarchy)에서 그들 상부의 모든 계승 우선권을 계승받는다고 가정하며, 루트 트랜잭션 $T_i$의 대체 트랜잭션 $T'_i$는 $T_i$의 수행이 포기되는 순간 $T_i$의 모든 활동 우선권을 물려받는다.

## 3. 프로토콜

### 3.1 실시간 중첩 우선순위 제한 프로토콜(Real-Time Nested Priority Ceiling Protocol, RTU-NPCP)

본 절에서는 Sha, Rajkumar, Lehoczky의 우선순위 제한 프로토콜[4,5]을 일반화 및 확장한 RTU-NPCP을 제안하고, 이 새로운 프로토콜이 무제한적인 블록킹과 데드락을 사전에 예방함과 동시에 루트 트랜잭션 집합의 직렬화(serializability)를 유지함을 증명한다. 프로토콜에서 사용된 개념 및 용어는 Lui Sha의 "Priority Ceiling Protocol"[4], "A Real-Time Locking Protocol"[5], Harder, T. and Rothermel, K.[1] 과 Moon, Seung-Jin[6] 에서 사용된 것과 동일하게 사용되었다. 프로토콜은 락에 관한 법칙과 우선 순위에 관한 법칙으로 두 부분으로 나누어 설명된다.

- 락/언락 운영 규칙(Lock/Unlock Operation rules):
  (가) 트랜잭션 $T_i$는 요청한 락이 객체 블록(object-block) 및 제한 블록(ceiling-block) 되지 않았을 경우에 한해 자원 $\sigma_i$에 대한 락을 획득할 수 있다.
  (나) 트랜잭션 $T_i$는 자신이 완료되거나 중단되었을때 부모 트랜잭션에게 자신이 소유한 모든 락을 돌려준다. 이때 부모는 완료되거나 중단된 하위 트랜잭션으로부터 돌아온 모든 락을 보유한다. 중단되었을 경우 대체 트랜잭션 $T'_i$는 중단으로부터 복구하기 위해서 초기화된다.

- 우선권 계승 규칙(Priority Propagatin rule):
  (가) $T_i$가 $T_j$에 의해 객체 블록이 되었을 경우,

$T_i$의 활동 우선권, $P^*(T_i)$는 $T_j$에 속하는 모든 트랜잭션 트리에 계승된다.

(나) $T_i$가 한계 블록이 되었을 경우 $T_i$의 활동 우선권, $P^*(T_i)$는 $\varnothing^* \geq P^*(T_i)$를 가진 모든 트랜잭션 트리에 의해 계승되어진다.

# 4. RT-NPCP 의 특성

본 절에서는 RTU-NPCP의 특성을 증명하며, 이를 위해 트랜잭션 인스탄스는 결코 중단 되거나 그 수행을 고의적으로 지연시키지 않는다고 가정한다.

## 4.1 데드락 비존재성(No Deadlock)

(1) 정리 1

*트랜잭션 $T_h$가 일단 락킹 존에 들어가면 그것은 블록 될 수 없다.*

(가) 증명:

$T_h$가 시간 $t_h$에 자신의 락킹에 들어갔다고 가정하자. 가정은 $P(T_h) > \varnothing^*$를 따르고, $P(T_h)$가 $T_h$의 기본 우선권 라고 하고, $\varnothing^*$가 시간 $t_h$에서의 시스템 한계(System ceiling)가 된다. $t_c$를 $T_h$가 블록된 $T_h$의 락킹 존내에서의 첫번째 시간(instant) 라고 하면 이것은 적어도 아래의 3가지 방법 하나에 의해 블록되어진다.

① 계층 블록(Inheritance-blocked) 경우

이 경우 더 낮은 기본 우선권를 가지고 $t_c$때의 $T_h$보다 높은 활동 우선권을 가진 몇몇 다른 트랜잭션 $T_l$이 존재한다. 이런 트랜잭션 $T_l$은 $T_h$보다 높은 활동 우선권을 가진 다른 트랜잭션 $T_x$로부터 우선권을 전달받아야 하는데 RTU-NPCP에 의하면, 이것은 $T_x$가 $T_l < T_c$인 어떤 순간에 $T_l$에 의해 락된 자원$\sigma_l$이 객체 블록 되었거나 제한 블록 되었을 경우에만 가능하다. 소유 한계(Property ceiling) 정의에 따라 $\varnothing t_l(\sigma_l) \geq P^*(\sigma_x) \geq P(T_h) > P(T_l)$ 이 성립한다.

② 개체 블록(Object-blocked) 경우

트랜잭션 $T_h$는 tc때에 어떤 자원 $\sigma_l$에 대한 락을 요청하여야 하고, 어떤 $T_l < T_c$인 때의 $T_c$에 의해 획득된 $\sigma_l$ 위에 모순락(conflicting lock)을 이미 유지하거나 보유하고 있는 어떤 다른 트랜잭션 $T_l$이어야 한다. 이는 트랜잭션이 락킹 블록 안에서 정지(suspend)할 수 없다고 가정하였기 때문에 $T_h$는 $T_l$을 선점해야하고, 그래서 $T_l$이 $T_h$보다 낮은 기본 우선권를 가져야만 한다. 우선권 한계(Priority ceiling) 정의에 따라 $\varnothing t_l(\sigma_l) \geq P(T_h) > P(T_l)$이다.

③ 한계 블록(Ceiling-blocked) 경우

트랜잭션 $T_h$은 어떤 자원 $\sigma_h$에 대해 lock 을 요구해야 하고, $T_l > T_c$인때에 얻어지는 $\varnothing t_l(\sigma_l) = \varnothing^* \geq P(T_h)$ 한계을 가진 어떤 자원 $\sigma_l$에 대해 락을 유지나 보유하는 어떤 다른 트랜잭션 $T_l$이어야 한다. 이는 트랜잭션이 락킹 존안에서 지연시킬 수 없다고 가정했기 때문에 $T_h$는 $T_l$을 선점해야하고, 그래서 $T_l$이 $T_h$보다 낮은 기본 우선권을 갖으며 이는 $\varnothing t_l(\sigma_l) \geq P(T_h) > P(T_l)$를 따른다.

모든 경우에, $\varnothing t_l(\sigma_l) \geq P(T_h) > P(T_l)$ 같이 자원 $\sigma_l$에 대해 락을 유지하고 있는 낮은 기본 우선권을 가진 트랜잭션 $T_l$이 있다. $T_h$가 락킹 존 안에서는 지체될 수 없으므로 $T_h$가 $\sigma_l$에서 락을 획득하기 위해 실행되어야 하며, $T_l$은 $t_l$때에 $T_h$를 블록을 상속해야 한다. $t_c$의 선택에 의해 $t_h < t_l < t_c$를 가질 수 없고, $t_l < t_h < t_c$에 따른다. 이 경우, $\varnothing^* t_{lh} \geq \varnothing t_l(\sigma_l) \geq P(T_x)$는 $T_h$가 $t_h$시간에서 블록 존에 들어갈 수 있다는 결론이 생겨 처음의 가정에 위반되며, 정리가 성립된다. □

(2) 정리 2

*$T_l$이 블록 되었을 경우 이것은 보다 낮은 기본권을 가진 트랜잭션에 의해 블록 된다.*

(가) 증명 :

이와 같은 추론은 위의 정의를 증명하는 사례분석에 따른다. □

(3) 정리 3

*트랜잭션 $T_i$은 자신이 블록킹 존안에 있는 경우에만 다른 트랜잭션을 블록킹 할 수 있다.*

(가) 증명 :

이는 이전의 세가지 블록킹의 정의로부터 직접적으로 따른다. 각각의 경우에서 $T_i$은 어떤 자원에 대해 락을 획득하고 있어야 하며, 객체 블록의 경우 $T_i$에 의해 유지되어진 락은 요청된 락과 충돌한다. 한계 블록의 경우 $T_i$에 의해 유지되어진 락은 기본 우선권보다 높아지기 위해서 활동 우선권이 필요하므로, 정리가 성립된다. □

만약 어떤 특정시간내에서 $T_3$가 $T_2$를 블록하고 $T_2$가 $T_1$을 블록한다고 가정하자. 그러면 $T_1$은 $T_3$에 의해서 이행적으로(transitively) 블록 되었다고 정의된다.

(4) 정리 4

*RTU-NPCP는 이행적인 블록킹을 막는다.*

(가) 증명 :

$T_3$가 $T_2$를 블록하고, $T_2$가 $T_1$을 블록한다고 가정하자. 정리 3에 의해서 $T_3$를 블록하려면 $T_2$는 블록킹 존(blocking zone)안에 존재해야만 한다. 하지만 정리 1에 의해 $T_2$는 블록킹 존 안에 존재하는 동안 $T_3$에 의해 블록될 수 없다. 이는 가정의 위반이 되므로, 정리 가 성립된다. □

'트랜잭션 $T_i$는 $T_i$가 $T_j$에 의해 자원 블록 되었을 경우 $T_j$를 기다린다*(wait-for)*' 라고 정의하자. 그러면, n 트랜잭션들의 집합에 대해 기다림 관계내의 사이클*(cycle in the wait-for relation)*은 모든 $j$에 대해($i \leq j \leq i+n-1$), $T_j$ *wait-for* $T_{j+1}$ 이며 $T_i = T_{i+n}$의 관계가 성립된다.

(5) 정리 5

*RTU-NPCP는 데드락을 방지한다.*

(가) 증명 :

데드락이 있다고 가정하자. 그러면 [7]에서 증명처럼 데드락은 기다림*(wait-for)* 관계내에 사이클이 존재해야 하며, n개의 루트 트랜잭션은 $T_1$, $T_2$, …, $T_n$의 사이클을 포함한다고 가정할수 있다. 그러나 정리 4에 의해 이 사이클내에 존재하는 트랜잭션의 수는 오직 2개이며 예로, $T_1$은 $T_2$를 기다리거나 그 반대의 경우이다. 또한 정리 3에 의해 두 트랜잭션은 모두 자신의 락킹 존에 속해 있어야 한다. 이를 일반화하여, $T_1$이 자신의 락킹 존에 저 들어간다고 가정하자. 그러면 정리 1로부터 $T_1$은 락킹 존 에 있는 동안 결코 블록 되지 않는데, 이는 모순이며 정리가 성립된다. □

## 4.2 복수 우선순위 반전 불허용(No Multiple Priority Inversions)

복수 우선 순위 반전은 루트 트랜잭션이 더 낮은 기본 우선권을 가진 트랜잭션에 의해 한 개 이상의 락킹 존 기간동안 블록되는 상황을 가정한다.

(1) 정리 6

*RTU-NPCP 는 복수 블록을 예방한다.*

(가) 증명 :

$T_h$가 복수 블록킹을 허락한다고 가정하자. 정리 2에서 트랜잭션은 보다 낮은 기본권을 가진 트랜잭션 일때만 블록을 할 수 있고, 정리 3에

의해 보다 낮은 기본 권한의 트랙잭션은 자신의 락킹 존 안에 있어야 한다. $T_h$는 어떤 단일한 보다 낮은 기본 권한의 락킹의 존속기간보다 길게 블록되어 있다고 가정했으므로 두 개 이상의 더 낮은 기본 권한의 트랙잭션, $T_{l1}$과 $T_{l2}$는 $T_h$를 블록하고 그들의 락킹 존에 있어야 한다. 트랜잭션은 자신의 락킹 존안에서는 지연되지 않는다고 가정하였기 때문에 $T_{l1}$과 $T_{l2}$ 중 하나가 자신이 락킹 존안에 존재하는 동안 다른 하나를 선점해야 한다. 이를 일반화하면, $T_{l2}$는 먼저 자신의 락킹 존에 들어가고, $T_{l1}$는 그것을 선점한다. 만약 $T_h$가 $T_{l2}$에 의해서 블록되었다면 그것은 자원 블록이거나 상속 블록이 되어야하고, 두 가지 경우를 가정해 볼 수 있다.

① 자원 블록인 경우

$\sigma$는 $T_h$가 블록되었을 당시에 $T_{l2}$에 의해 락된 자원이라고 하면, $P(T_h) \leq \varnothing(\sigma_2)$이다. $T_{l1}$이 $T_{h2}$를 선점하고 t1때에 락킹 존을 들어갔다고 가정한다. 락을 유지하고 있는 동안은 지연은 없고, $T_{h1}$은 어떤 상속 우선권을 갖고 있지 않다. $T_{l1}$이 성공적으로 그 락킹 존에 들어가면 $P(T_h) \leq \varnothing(\sigma_2) \leq \varnothing^* t_1 < P(T_{l1})$ ($\varnothing^* t_1$는 시간 $t_1$에서의 시스템 한계이다) 이며, 이것은 $P(T_{l1}) < P(T_h)$에 위반된다.

② 상속 블록인 경우

이는 $T_{l2}$가 상속 권한를 상속받았을 때 일어나고 어떤 보다 높은 권한 작업를 자원 블록할 때 일어난다. 그러나 이것은 이행적 블록킹의 경우에 해당하므로 정리 4를 위반하게 된다. 두 경우에서 $T_{l2}$가 락킹 존에 있는 동안 $T_{l1}$은 락킹 존에 들어가지 않으므로 정리가 성립된다. □

## 4.3 직렬화(Serializability)

본 절에서는 RTU-NPCP가 직렬성을 유지한다는 것을 증명한다. 이는 원래의 비계층적 PCP가 가지지 못하는 중첩 PCP의 가장 중요한 속성으로 중첩 트랜잭션 프로토콜에서 직렬화는 꼭 보존되어야 할 특성이다. 트랜잭션 인스탄스들의 집합에 대한 스케쥴은 리프 트랜잭션 동작(lock, unlock, read, write)으로, 기본적인 데이타베이스 동작이 수행되는 순서를 정해놓은 것이다. 각각의 트랜잭션 인스탄스의 기본적인 데이터베이스동작이 연속적으로 일어날 경우 스케쥴은 직렬(*serial*)이라고 하며 스케쥴의 동작결과가 어떤 직렬 스케쥴의 결과와 같을 때 직렬화 가능(*serializable*)이라고 한다. 본 논문에서는 전통적인 비계층적 트랜잭션 모델[8]에서 직렬화(serializability)를 증명하기 위해 사용되는 그래프인 직렬화 그래프 테스트를 이용하여 RTU-NPCP에 의해 만들어진 스케쥴이 직렬화가 가능함을 증명한다. 주어진 실행 스케쥴에 대한 직렬화 그래프는 노드가 트랜잭션 인스턴스인 직접적인 그래프이며 간선(edge) $T_i \rightarrow T_j$는 스케쥴상에서 $T_i$가 $T_j$ 이전에 충돌접근(*conflict-access*)하는 $\sigma_k$가 존재한다는 것을 나타낸다. 그러므로 트랜잭션 인스턴스의 혼합실행시(inter-leaved execution) 실행 스케쥴을 나타낸 직렬화 그래프가 사이클을 가지지 않을 때만 직렬화 가능하다[8]. 제안된 중첩 트랜잭션 모델은 트랜잭션의 계층적인 조합을 허용하기 때문에, 이는 직렬화 그래프의 모든 계층 계층의 트랜잭션을 포함하며, 직렬화 그래프상에 트랜잭션 $T_i$로부터 트랜잭션 $T_j$로의 간선이 있을 때, $T_i$의 락킹 존은 $T_i$의 모든 상위 트랜잭션으로 부터, $T_j$의 모든 상위 트랜잭션으로 간선이 있다는 것을 보증한다. 특히 $T_i$의 루트 트랜잭션에서 $T_j$의 루트 트랜잭션으로의 간선이 있다는 것과 같으며, 루트 트랜잭션의 직렬화가 가능하다면 이는 다른 계층의 트랜잭션도 직렬화가 가능하다고 결론지을 수 있다.

(1) 정리 7

*직렬화 그래프에서 트랜잭션 $T_i$에서 트랜잭션 $T_j$로 간선이 있을 경우 RTU-NPCP는 $T_j$가 락킹 존*

*을 끝내기 전에 $T_i$가 락킹 존을 끝낼 수 있도록 보증한다.*

(가) 증명 :

직렬화 그래프에서 $T_i \rightarrow T_j$로의 간선에 줄 수 있는 자원 중 처음으로 요청된 자원이 $\sigma_l$이라고 가정하자. RTU-NPCP에 따르면, $T_i$가 더 이상 락을 보유하지 않는 시점에서 모드 $m$ 내에 $T_i$가 $\sigma_l$을 접근할 때 인스턴스의 시스템 한계 $\varnothing^*$는 적어도 $\varnothing(\sigma_l,m)$보다 높아야 하며, 아래의 두 경우가 있다.

① 경우 1

$T_l$($T_l$의 하위 트랜잭션 포함)이 어떤 높은 활동 우선권을 가진 트랜잭션에 의해 선점되지 않는다면 전제가 유효하다.

② 경우 2

$T_i$가 높은 활동 우선권 트랜잭션(아마도 $T_j$)에 의해 선점되고, 그후에 $T_j$는 $T_i$가 선점되어 있는 동안 실행된다면, $T_j$의 $\sigma_l$에 대한 락 요청 $T_i$가 락킹 존에 존재할 때까지 거부된다(그 후 모든 유지된 락은 해제된다). 왜냐하면 $T_j$의 락 요청시 $P(T_j) \leq \varnothing(\sigma_l) \leq \varnothing^*$이기 때문이며 정리가 성립된다. □

(2) 정리 8

*RTU-NPCP에 의해 제작된 어떤 스케줄은 직렬화가 가능하다.*

(가) 증명 :

RTU-NPCP에 의해 제작된 스케줄의 직렬화 그래프는 $T_n \rightarrow T_l \rightarrow T_2 \rightarrow T_i \cdots T_{n-1} \rightarrow T_n$의 사이클을 가지고 있고, 각 $T_i$는 루트 트랜잭션 인스탄스이고 n>1 이라고 가정하자. 이것은 또 다른 $T_n$으로 들어오는 간선과 n으로부터 나가는 간선이 있다는 것을 나타낸다. 정리 7에 의해 $T_{n-1}$은 $T_n$이 수행되기 전에 락킹 존에 들어가고, $T_{n-2}$는 $T_{n-1}$이 수행되기 전에 그 락킹 존에 들어가야 한다. 그러므로 이행적으로, $T_{n-2}$는 $T_n$이 수행되기 전에 락킹 존에 들어가야 하며, 이런 연쇄(chain)에 따르면 $T_l$은 $T_n$이 수행되기 전에 자신의 락킹 존에 들어가야 한다는 것을 알 수 있으며, $T_n$은 $T_l$이 수행되기 전에 락킹 존에 들어가야 한다(예, $T_n \rightarrow T_l \rightarrow T_n$). 그러므로 $T_n$이 락킹 존을 나가는 즉시 $T_l \rightarrow T_n$처럼의 간선의 형태를 이루는 것은 불가능한데 이는 $T_n$이 더 이상 어떤 자원에도 접속하지 않는다는 가정을 위반하기 때문이며 정리가 성립된다. □

## 5. 결 론

본 연구에서는 중첩 트랜잭션에 시간제약을 결합한 확장된 트랜잭션 모델인 RT-NTM을 제시하였으며, 단일 프로세서상에서 중첩 트랜잭션을 가진 주기적인 프로세스를 스케쥴링하는 문제를 분석하였다. 본 모델은 통합 락 기반 동시성 제어와 우선 순위 스케쥴링 알고리듬을 기반으로 한 단일 프로세서 우선 순위 제한 프로토콜(Uniprocessor Priority Ceiling Protocol)을 사용하여, 경성 실시간 트랜잭션들의 시간제약을 보증해줄 뿐만 아니라 데이터베이스의 무결성도 유지시켜 주었다. 제안된 RTU-NPCP는 주기적인 루트 트랜잭션이 트랜잭션 처리 과정에서 발생되는 정상및 비정상적인 단계의 치명적 블럭시간의 상위 경계를 설정토록 하였으며, 고정 우선순위 기반 방법(Rate-Monotonic Priority Assignment)[9]과 RTU-NPCP를 이용하여 주기적인 루트 트랜잭션 집합에 관한 스케쥴링 분석을 가능케 하였다.

향후 본 연구는 제안된 모델이 최근 관심을 모으는 웹기반 실시간 멀티미디어 데이터베이스에 필수인 시간제약 만족성, 동시성 제어 및 우선 순위 기반 트랜잭션 스케쥴링이 가능하므로, 이를 웹기반 실시간 멀티미디아 데이터베이스 모델에

적용시킬 예정이며, 또한 병렬 및 분산시스템으로 확장할 예정이다.

## 참 고 문 헌


[1] Harder, T. and Rothermel, K., "Concurrency Control Issues in Nested Transactions", VLDB Journal, Vol. 2, No.1, pp. 39~74, 1993.

[2] Moss, J.E.B., "Nested Transactions: An Approach to Reliable Distributed Computing," MIT Press, 1985.

[3] Traiger, I.L., Eswaran, K.P., Gray, J.N., Lorie, R.A., "The Notion of Consistency and Predicate Locks in an Database System", Communication of the ACM, Vol. 19, No. 11, pp. 624~633, 1976.

[4] Sha, L., Rajkumar, R., and Lehoczky, J.P., "Priority Inheritance Protocols: An Approach to Real-Time Synchronization", IEEE Transaction on Computers, Vol. 39, No. 9, pp. 1175~1185, 1990.

[5] Sha, L., Rajkumar, R., Son, S.H., and Chang, C.H., "A Real-Time Locking Protocol", IEEE Transaction on Computers, Vol. 40, No. 7, pp. 793~800, 1991.

[6] Moon, Seung-Jin, "Synchronization and Scheduling Issues for Real-Time Nested Transaction", Ph.D Dissertation, The Florida State University, 1997.

[7] Holt, R.C., "Some Deadlock Properties of Computer Systems", ACM Computer Systems, Vol. 4, No. 3, pp. 179~196, 1972.

[8] Papadimitriou, C.H., "The Serializability of Concurrent Database Updates", The Journal of ACM, pp. 631~653, October 1979.

[9] Liu, C.L., and Layland, J.W., "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal of the Association for Computing Machinery, Vol. 20, No.1, pp. 46~61, 1973.


## ◐ 저 자 소 개 ◑


**문 승 진**

1986년 텍사스 주립대학교 (오스틴) 컴퓨터학과 졸업 (학사)
1991년 플로리다 주립대학교 컴퓨터학과 졸업 (석사)
1997년 플로리다 주립대학교 컴퓨터학과 졸업 (박사)
1997년 6월~1997년 8월 (주) 맥시스템 멀티미 디아 및 통신 연구소 소프트웨어 실장
1997년 9월~현재 수원대학교 전자계산학과 조교수
관심분야 : 실시간 데이타베이스, 웹기반 데이터베이스, 모빌 데이터베이스, 실시간 스케쥴링 등