通信情報保護學會論文誌
第9卷 第1號, 1999. 3

다단계 보안 데이타베이스에서 직렬화 순서의 동적 재조정을 사용한 병행수행 제어 기법

김 명 은*, 박 석**

Concurrency Control with Dynamic Adjustment of Serialization Order in Multilevel Secure DBMS

Myung-Eun Kim*, Park Seok**

요 약

다단계 보안 데이타베이스 관리 시스템(multilevel secure database management system : MLS/DBMS)은 시스템이 사용자를 위한 여러 개의 보안 인가 등급(security clearance level)과 시스템 내의 데이타를 위한 여러 개의 분류 등급(classification level)을 가진 데이타베이스 관리 시스템으로써 이러한 시스템의 목적은 기밀 정보를 인가 받지 않은 사용자로부터 보호하는 것이다.

이미 연구되어온 알고리즘들은 로킹이나 타임스탬프 기법을 변형하여 비밀채널을 제거하는데 주안점을 두었기 때문에 상위 등급 트랜잭션들이 하위 등급 트랜잭션에 의해 반복적으로 지연되는 기아현상을 유발시켰다.

이러한 문제를 해결하기 위해 본 논문에서는 오렌지 로크를 기반으로 하는 다단계 보안 DBMS에서 직렬화 순서의 동적인 재조정 방식을 이용하여 상위 등급 트랜잭션의 기아현상을 완화하는 알고리즘을 제안한다. 기존의 보안 알고리즘이 다중버전상에서 수행되었던 것과는 달리 제안된 알고리즘은 단일버전상에서 수행함으로써 트랜잭션의 병행성 정도를 높일 수 있는 알고리즘을 제안한다. 제안한 알고리즘이 병행수행 상에서 직렬성을 만족함을 증명하고, 보안성을 만족함을 보인다.

Abstract

In Multilevel Secure Database Management System(MLS/DBMS), we assume that system has a security clearance level for each user and a classification level for each data item in system and the

* 동양시스템하우스
** 서강대학교
본 연구는 정보통신연구진흥원 98년 대학기초연구 지원비에 의한 것임 (C1-98-0774)

objective of these systems is to protect secure information from unauthorized user.

Many algorithms which have been researched have focus on removing covert channel by modifying conventional lock-based algorithm or timestamp-based algorithm. but there is high-level starvation problem that high level transaction is aborted by low level transaction repeatedly.

In order to solve this problem, we propose an algorithm to reduce high-level starvation using dynamic adjustment of serialization order, which is basically using orange lock. Because our algorithm is based on a single version unlike conventional secure algorithms which are performed on multiversion, it can get high degree of concurrency control. we also show that it guarantees the serializability of concurrent execution, and satisfies secure properties of MLS/DBMS.

Key word : Multilevel Secure Datatbse Management System, Orange Lock, Dynamic Adjustment of Serialization Order, Local Workspace

1. 서 론

데이타베이스 시스템(database system)은 미리 정의된 연산들의 집합과 데이타베이스의 효율적인 관리 및 이러한 데이타에 대한 연산들로 구성된 트랜잭션들을 위한 트랜잭션 관리를 제공하는 시스템을 의미한다[1]. 여러 분야의 다양한 요구에 따른 다양한 데이타베이스의 발달과 더불어 개인정보의 보호 및 기밀문서 관리의 문제가 중요시되자, 데이타 보안에 대한 새로운 인식과 필요성이 대두되었다. 이러한 보안문제를 해결하기 위해 다단계 보안 시스템(multilevel secure system : MLS)에 대한 연구가 시작되었다. 다단계 보안 시스템은 시스템이 사용자를 위한 하나 이상의 보안인가 등급(security clearance level)과 시스템 내의 데이타들을 위한 하나 이상의 분류 등급(classification level)을 가진 시스템을 말하며 이러한 등급을 이용하여 기밀정보의 노출을 제어하는 기법을 제공한다[1][5].

다단계 보안 데이타베이스 관리 시스템(MLS/DBMS)은 다단계 보안 시스템의 접근제어를 데이타베이스 관리 시스템에 적용한 것이다. 이러한 시스템의 목적은 기밀 정보를 인가 받지 않은 사용자로부터 보호하는 것이다. 이를 위해 MLS/DBMS에서는 제어하는 모든 데이타 항목에 대해 고유한 등급을 가져야 하며, 사용자가 데이타에 접근하는 것은 사용자의 보안 인가와 데이타의 보안 등급에 의해 제어되도록 하고 있다. 이러한 다단계 보안 데이타베이스에서 가장 중요한 문제는 비밀 채널(covert channel)의 발생이다. 즉, 비밀채널이란 낮은 등급의 데이타베이스 시스템 사용자가 트로이안 목마(trojan horse)나 바이러스(virus)와 같은 경로를 통하여 자신보다 높은 보안등급으로부터 자신의 보안등급으로 정보를 유출하여 얻는 것을 의미한다.[1][5] 병행 수행 로크(lock)는 데이타베이스 시스템에서 전형적인 비밀채널의 예로 사용되어 왔다. 로킹(locking)은 많은 시스템에서 사용되는 기본적인 병행수행 제어 기법이지만 다단계 보안 데이타베이스 시스템에서 사용하기에는 비밀채널의 형성이라는 문제점을 안고 있다. 이러한 문제점을 해결하기 위해 비밀채널을 제거한 여러 가지 알고리즘이 연구되었는데, 오렌지 로킹(orange locking)기법, 보안 2단계 로킹 기법 등이 이러한 알고리즘의 예이다. 기존의 보안 알고리즘에서는 이러한 비밀채널을 제거하는 부분에 초점을 맞추었기 때문에 하위 등급 트랜잭션에 의해 상위 등급 트랜잭션의 수행

이 반복적으로 지연되는 상위 등급 트랜잭션의 기아현상이 일어나거나, 직렬성을 만족시키지 못하는 문제가 발생된다.

본 논문에서는 위에서 언급한 직렬성 위배 문제를 방지하고 상위 등급 트랜잭션의 기아현상을 보다 완화시킴으로써 재수행 및 지연에 의한 자원 낭비를 막고 병행 수행상의 효율을 높임과 동시에 보안성을 만족하는 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 1장에서는 다단계 보안 데이타베이스 시스템의 모델을 설명하고, 2장에서는 다단계 보안 데이타베이스 환경의 병행수행 기법으로써 연구되어온 오렌지 로킹 기법과 이러한 병행수행 기법의 문제점을 살펴본다. 3장에서는 이러한 문제점을 해결하기 위해 직렬화 순서의 동적 재조정 방식(dynamic adjustment of serialization order : DASO)을 사용한 병행수행 제어기법을 제시하고 이 알고리즘이 정확성을 만족함을 증명한다. 4장에서는 실험을 통하여 기존의 알고리즘과 본 논문에서 제안한 알고리즘과의 성능을 비교 분석한 후 5장에서 결론을 맺는다.

2. 다단계 보안 데이터베이스

2.1 다단계 보안 데이타베이스

다단계 보안 시스템에서는 강제적 접근제어(mandatory access control)을 가정한다. 강제적 접근제어는 부분순서를 가지는 보안등급 집합을 기반으로 한다. 보안 시스템이 보안 정책을 어떻게 시행하는지에 대한 모델을 보안 모델이라고 하며 강제적 보안 정책을 수행하는 가장 일반적인 모델은 벨-라파둘라 모델(Bell-LaPadula model)이다[6]. 보안 데이타베이스 시스템에서는 벨-라파둘라 모델을 기본으로 하고 있다.

시스템에 있는 데이타는 비밀 등급이라는 보안 등급이 부여되고 사용자들에게는 인가 등급이라는 보안 등급이 부여되어 시스템이 다음과 같은 특성을 만족하도록 한다. 보안 등급을 이루는 구성요소는 계층적 등급(hierarchical level)과 범주 집합(category set)으로 이루어진다. 계층적 등급은 전체적으로 순서화 되어 있고, 범주 집합은 부분적으로 순서화 되어 있다. 보안 등급은 L = (A, B)과 같이 정의된다. 여기서 A는 계층적 등급으로 Top secret(TS) 〉 Secret(S) 〉 Confidential (C) 〉 Unclassified(U) 등으로 구성되며 B는 범주 집합으로 인사, 군수, 작전, 정보 등으로 구성된다. 보안모델은 아래와 같은 성질을 만족해야만 한다. 여기서 L(T)과 L(X)는 각각 트랜잭션 T의 인가등급과 데이타 항목 X의 분류등급을 나타낸다.

단순 보안 성질 (Simple Security Condition)

트랜잭션 T가 L(T)≥L(X)를 만족할 때에만 데이타 항목 X에 대한 판독 연산이 가능하다.

제한된 *-성질 (Restricted Star-Property)

트랜잭션 T가 L(T)=L(X)를 만족할 때에만 데이타 항목 X에 대한 기록 연산이 가능하다.

위의 두 보안성질 외에도 보안이 유지되는 시스템에서는 비밀채널을 통해 비합법적인 정보가 흐르지 않도록 스케줄링해야 한다. 이러한 조건을 만족하는 수행기록을 생성하는 스케줄러를 보안 스케줄러라고 한다. 본 논문에서는 병행성을 높이면서, 보안성을 만족하는 새로운 보안 스케줄러를 설계하는데 중점을 둘 것이다.

2.2 다단계 보안 병행수행 제어와 문제점

지금까지 발표된 보안을 유지하기 위한 알

고리즘들은 기존의 대표적인 기법인 로킹과 타임스탬프를 변형한 방법들이었다. 따라서 보안을 유지하기 위해서는 많은 추가적인 작업을 필요로 한다. 그러나, 기존의 방법들은 보안이라는 개념을 고려하지 않았기 때문에 보안 유지를 위한 추가적인 기능을 제시했음에도 불구하고, 상위 등급의 기아현상과 직렬성을 위반하는 또다른 문제점이 발생한다.

기존의 병행수행 방법 중 하나인 로킹은 직렬성을 보장하기 위하여 판독중인 데이타를 기록할 수 없도록 한 장치이다. 만약 상위 등급 트랜잭션이 하위 등급 데이타에 대하여 판독 로크를 갖고 있을 경우, 하위 등급 트랜잭션이 기록을 요청했다고 가정해 보자. 보안 데이타베이스에서는 상위 등급의 판독을 하위 등급 사용자에게 알려서는 안되기 때문에 상위 등급 트랜잭션의 판독 중에 하위 등급의 기록을 허용해야 한다. 따라서 판독 중에 기록을 허락한 상위 등급 트랜잭션의 직렬성이 보장되고, 상위 등급 트랜잭션이 반복적으로 지연되지 않도록 알고리즘을 수정해야 한다.

(1) 오렌지 로킹 알고리즘

오렌지 로킹 알고리즘의 기본적인 내용은 다음과 같다. 오렌지 로킹 알고리즘은 지역 작업 공간(local workspace)을 필요로 하는데, 지역 작업 공간은 각각의 트랜잭션마다 생성되는 임시 작업 공간으로써, 하향 판독(read-down) 연산은 모두 지역 작업 공간에서 이루어진다. 또한 작업 공간으로부터 하향 판독 연산이 모두 성공적으로 끝났는지 검사하기 위해 home-free point라는 검사점을 두게 된다. 여기서 home-free point란 각각의 트랜잭션이 모든 처리를 완료하기 전에 도달하는 검사점으로써, 트랜잭션 T가 home-free point에 도달했다는 것은 트랜잭션 T에 의해 하향 판독되는 모든 데이타 항목 x가 판독 로크를 얻어서 판독을 성공적으로 수행하거나, 오렌지 로크를 얻은 후 판독을 모두 성공적으로 수행했다는 것을 의미한다[1].

오렌지 로킹 알고리즘은 기존의 로킹 기법과는 달리 오렌지 로크라는 새로운 로크를 정의함으로써 상위 등급 트랜잭션의 불필요한 하향 판독 연산의 재수행 횟수를 줄인 점이 특징이다[1]. 즉, 오렌지 로킹 알고리즘에서는 트랜잭션이 수행될 때 먼저 하향 판독 연산을 우선으로 하여 수행한 후, 동급 데이타에 대한 연산을 처리한다. 만일 상위 등급 트랜잭션이 데이타 항목 x에 대하여 하향 판독하고 있는 중에 하위 등급 트랜잭션이 동일한 데이타 항목에 대하여 기록 연산을 수행하려 한다고 가정하자. 이 때 하위 등급 트랜잭션을 지연시킨다면 비밀채널이 생성되므로 우선 상위 등급 트랜잭션이 가지고 있는 판독 로크를 오렌지 로크로 전환한 다음 하위 등급 트랜잭션에게 즉시 기록 로크를 인가한다. 하위 등급 트랜잭션이 모든 수행을 마치고 완료한 뒤, 상위 트랜잭션은 재시작하게 되는데 이 때 재시작 시점을 오렌지 로크가 걸린 시점으로 정함으로써 불필요한 하향 판독 연산의 재수행 횟수를 줄인 것이 오렌지 로킹 알고리즘이다. 상위 트랜잭션이 하향 판독을 모두 성공적으로 수행했을 경우 home-free point에 도달하게 되고, 나머지 연산은 strict 2PL 방식에 의해 수행된다. 또한 다중버전이 아닌 단일버전상에서 수행하므로 시스템 효율을 높인다. 일반적으로 상위 등급 사용자와 하위 등급 사용자가 볼 수 있는 데이타를 여러 개 만들어 놓음으로써 데이타베이스 상에서의 비밀채널 생성을 방지하는 알고리즘이 많이 연구되어 왔다. 그러나 오렌지 로킹 기법은 오렌지 로크를 정의함으로써 단일 버전 상에서 비밀채널을 제거함으로 시스템 작업공간의 효율성을 높였다.

(2) 오렌지 로킹 알고리즘의 문제점

오렌지 로킹 알고리즘과 같이 보안 요구사

항을 만족하기 위해 기존의 로크 기반 알고리즘을 변경할 경우, 상위 트랜잭션을 취소시킴으로써 병행수행 제어의 효율성이 떨어지고, 재수행 비용이 많이 든다. 다음 예의 스케줄을 통하여 오렌지 로킹의 문제점인 불필요한 상위 트랜잭션의 재수행에 대하여 설명하겠다.

[예 1] $T=\{T_1, T_2\}$

	1	2	3	4	5
T_1(TS)	$r_1(x)$			$w_1(y)$	C_1
T_2(TS)		$w_2(x)$	C_2		

트랜잭션 T_1, T_2로 이루어진 수행기록 H가 있다고 가정하자. T_1의 보안등급은 TS(Top Secret), T_2의 보안등급은 S(Secret)라 하고, 데이타 항목 x, y가 각각 TS, S등급이라고 할 때, 오렌지 로킹 알고리즘으로 스케줄링한다면, 위의 예제 스케줄의 경우 하위 등급의 트랜잭션 T_2가 이미 상위 등급의 트랜잭션이 판독로크를 가지고 있는 데이터 x에 대하여 기록로크를 요구할 때, 비밀채널을 통한 정보의 누출을 막기 위해 데이터 x에 대하여 먼저 판독로크를 가지고 있는 T_1이 취소된다.

[예 2] T = {T1, T2, T3)

	1	2	3	4	5	6	7	8	9	10	11	12
T_1(TS)	$r_1(y)$	$r_1(p)$	$r_1(x)$	$w_1(z)$	$w_1(q)$						$r_1(t)$	C_1
T_2(TS)								$r_2(p)$	$w_2(1)$	c_1		
T_3(S)						$w_3(p)$	c_3					

트랜잭션 T_1, T_2, T_3로 이루어진 수행기록 H가 있다고 가정하자. 트랜잭션 T_1, T_2, T_3 각각의 보안등급이 TS, TS, S라고 하고, 데이타 항목 l, t, y, x, z, q는 TS등급을 가지고, p는 S등급을 가진다고 할 때 다음과 같이 수행된다. 상위 등급 트랜잭션 T_1이 데이타 항목 p에 대하여 판독 로크를 가지고 판독 연산을 수행하고 있는데, 하위 등급 트랜잭션인 T_2가 동일한 데이타 항목인 p에 대하여 기록 연산을 요청했다면 트랜잭션 T_1의 판독 로크는 오렌지 로크로 전환된다. 데이타 p에 대하여 오렌지 로크를 가지고 있는 상태에서 상위 등급 트랜잭션 T_1이 판독 연산을 수행시 T_2가 기록한 데이타 값을 T_1이 읽는다면 이는 직렬성에 위배되므로, 스케줄러는 T_1의 작업 전체를 취소시킨다.

그러나 이러한 상위 등급 트랜잭션의 취소가 하위 등급 트랜잭션들의 빈번한 기록이나 고의적인 방해로 인해 반복되는 것을 상위 등급 트랜잭션의 기아현상이라고 하며, 이렇게 취소된 트랜잭션은 재수행시 많은 비용이 든다. 실제적으로 오렌지 로킹 알고리즘을 구현하여 적용한다면 불필요한 재수행으로 인해 시스템 성능이 현저하게 떨어지게 되는 문제점이 발생할 것이다. 그러므로 이러한 불필요한 재수행의 횟수를 줄임으로써 시스템의 성능을 높이고 직렬성을 만족하는 알고리즘을 제시하는 것이 본 논문의 목적이다.

3. 다단계 보안 데이타베이스 상에서 직렬화 순서의 동적인 재조정을 사용한 병행수행 제어 기법

3.1 직렬화 순서의 동적인 재조정 기법을 이용한 낙관적 병행수행 제어 기법

DASO(Dynamic Adjustment of Serialization Order)는 실시간 데이터베이스에서 사용되는 기법으로써 충돌 연산의 특성에 따라 직렬화

순서를 동적으로 조정하여 불필요한 재시작이나 블록킹을 막는 기법이다[2][3][4]. 예를 들면, 두 개의 트랜잭션 T_1, T_2를 가정하자. 이 때 T1의 우선 순위가 높다고 하고, T_1이 데이터 x를 판독하기 전에 T_2가 기록 연산을 수행한다. 2PL-HP를 사용하면, T_1 은 T_2를 취소시키든지, 아니면, T_2가 기록 로크를 해제할 때까지 기다려야 한다. 이렇게 되는 이유는 이미 과거 수행기록에서 직렬화 순서가 $T_2 \rightarrow T_1$로 결정되어 있기 때문이다. 따라서 T_1은 직렬화 순서에서 T_2를 앞설 수 없다. 그러나 DASO기법을 이용하여 두 트랜잭션의 직렬화 순서를 $T_1 \rightarrow T_2$로 조정하면 T_1이 블록 되거나 T_2가 취소될 필요가 없다. 그러므로 직렬화 순서를 동적으로 재조정함으로써, 보다 많은 트랜잭션이 취소되지 않고 완료할 수 있는 기회를 줄 수 있다. 따라서 트랜잭션의 불필요한 재수행 횟수를 줄일 수 있다.

이러한 장점을 가진 DASO기법을 OCC (optimistic concurrency control) 환경하에 적용하면 다음과 같다. OCC상에서 DASO가 적용된 프로토콜의 기본적인 수행과정은 OCC 프로토콜의 수행과정과 동일하다. 즉, 읽기단계(read phase), 검증단계(validation phase), 쓰기단계(write phase)로 나뉘어지며, DASO기법은 검증단계에서 적용된다. 트랜잭션 T_i가 트랜잭션 T_j전에 직렬화 되었다면, 검증단계에서 OCC 프로토콜상의 직렬화 순서를 보장하기 위해서 다음의 두 조건을 만족해야 한다[2][3][4].

[**조건** 1] T_i의 기록 연산은 T_j의 읽기 단계에 영향을 미치면 안된다.

[**조건** 2] T_i의 기록연산은 T_j의 기록연산을 덮어쓰면 안된다.

검증(validation) 트랜잭션을 T_i라고 하고, 현재 병행적으로 수행중인 활성(active) 트랜잭션을 T_j 라고 가정할 때, 다음의 세 가지 데이터 충돌 유형에 따라 T_i와 T_j사이의 직렬화 순서를 결정할 수 있다.

첫째, $RS(i) \cap WS(j) \neq \phi$(기록-판독 충돌)의 경우로써 활성 트랜잭션 T_j와 검증 트랜잭션 T_i간의 기록-판독 충돌은 T_i의 판독 연산이 T_j의 기록 연산에 의해 영향을 받지 않기 위해서는 T_i와 T_j사이의 직렬화 순서를 $T_i \rightarrow T_j$으로 조정함으로써 해결할 수 있다. 이러한 종류의 직렬화 조정을 순방향 조정 (forward adjustment)이라 한다.

둘째, $WS(i) \cap RS(j) \neq \phi$(판독-기록 충돌)의 경우로써 트랜잭션 T_j와 T_i사이의 판독-기록 충돌은 T_i와 T_j사이의 직렬화 순서를 $T_j \rightarrow T_i$로 조정함으로써 해결될 수 있다. 이것은 동일한 데이터에 대하여 T_j의 판독 연산이 T_j의 기록 연산에 앞선다는 것을 뜻한다. 즉, T_i가 T_j이전에 완료되었을지라도 T_j는 T_i보다 우선하는, 역방향으로 조정(backward adjustment)된 타임스탬프를 부여받는다.

셋째, $WS(i) \cap WS(j) \neq \phi$(기록-기록 충돌)의 경우로써 T_i와 T_j간의 기록-기록 충돌은 T_i의 기록 연산을 T_j가 덮어쓰지 않도록 하기 위해 T_i와 T_j간의 직렬화 순서를 $T_i \rightarrow T_j$로 조정함으로써 해결할 수 있다. 이러한 직렬화 조정을 순방향 조정(forward adjustment)이라고 한다.

위와 같이 트랜잭션의 충돌을 분류하여 직렬화 순서를 조정함으로써 상위 트랜잭션의 불필요한 재수행을 막을 수 있다. 다음의 여러 가지 예제를 통하여 DASO 알고리즘이 실제로 어떻게 적용되는지 보이겠다.

[예 3] $T = \{T_1, T_2\}$

	1	2	3	4	5
T_1(TS)	$r_1(x)$			$w_1(y)$	C_1
T_2(TS)		$w_2(x)$	C_2		

트랜잭션 T1, T2로 이루어진 수행기록 H가

있다고 가정하자. T1의 보안등급은 TS(Top Secret), T_2의 보안등급은 S(Secret)라 하고, 데이타 항목 x, y가 각각 TS, S등급이라고 할 때, DASO에 의해 수행된다면 다음과 같다.

step 1. T_2가 검증하는 시점에서 볼 때, T_2는 T_v가 되고, T_1은 T_a가 된다.

step 2. $WS(T_v) \cap RS(T_a) \neq \phi$이므로, 역방향 조정되어 직렬화 순서는 $T_1 \rightarrow T_2$가 된다.

step 3. T_2가 수행을 종료하고 완료한다.

step 4. T_1이 검증하는 시점에서 볼때, 활성 트랜잭션이 없으므로 완료한다.

[예 4] T = {T1, T2, T3)

	1	2	3	4	5	6	7	8	9	10	11	12
T_1(TS)	$r_1(y)$	$r_1(p)$	$r_1(x)$	$w_1(z)$	$w_1(q)$						$r_1(t)$	C_1
T_2(TS)								$r_2(p)$	$w_2(1)$	c_1		
T_3(S)						$w_3(p)$	c_3					

[예 4]에서는 트랜잭션 T_1, T_2, T_3로 이루어진 수행기록 H가 있다고 가정하자. 트랜잭션 T_1, T_2, T_3 각각의 보안등급이 TS, TS, S라고 하고, 데이타 항목 l, t, y, x, z, q는 TS등급을 가지고, p는 S등급을 가진다고 할 때 다음과 같이 수행된다.

step 1. T_3가 검증하는 시점에서 볼 때, 활성 트랜잭션은 T_1이다. 그러므로 T_3는 T_v가 되고, T_1는 Ta가 된다.

step 2. $WS(T_v) \cap RS(Ta) \neq \phi$이므로 , 역방향 조정되어 직렬화 순서는 $T_1 \rightarrow T_3$가 된다.

step 3. $TS(T_3)$ = t6, LCT(T1)≤ t6, WTS(y) = WTS(z) = t6이 되고 T_3는 완료한다.

step 4. T_2가 검증하는 시점에서 볼 때, 활성 트랜잭션은 T_1이다. 그러므로 T_2는 T_v가 되고, T_1는 T_a가 된다.

step 5. $WS(Tv) \cap RS(Ta) = \phi$이므로 , 순방향 조정되어 직렬화 순서는 $T_2 \rightarrow T_1$이 된다.

step 6. $TS(T_2)$ = t9, WTS(t) = RTS(z) = t9이 되고, T_2는 완료한다.

step 7. T_1이 검증하는 시점에서 활성 트랜잭션이 없으므로, RTS(t) = t11로 기록하고, 완료한다.

step 8. 주어진 스케줄의 직렬화 순서는 $T_2 \rightarrow T_1 \rightarrow T_3$이 된다.

3.2 다단계 보안 데이타베이스상에서 직렬화 순서의 동적인 재조정 기법을 사용한 병행수행 제어 기법

2장에서 살펴 보았듯이, 기존에 연구되어온 보안 알고리즘들은 보안 개념이 없는 일반 병행수행 기법들에 보안을 위한 개념들을 삽입한 형태를 취함으로써 보안 조건은 만족하나, 다른 측면에 많은 문제점을 가지고 있다. 상위등급 기아현상이나, 직렬성 위배 문제 등이 바로 그러한 문제점들이다.

두가지 문제를 해결하기 위해, 본 논문에서는 오렌지 로크에 기반한 다단계 보안 데이타베이스 상에서 직렬화 순서의 동적 재조정 기법을 이용하여 상위 등급 기아현상을 완화시킨 알고리즘을 제시하고자 한다. 제안된 알고리즘의 특징은 첫째, 오렌지 로크를 사용함으로써 이미 수행된 상위 등급 트랜잭션의 하향

판독 연산을 다시 수행하지 않아도 되는 점과 둘째, 직렬화 순서의 동적 재조정 기법을 사용함으로써 상위 등급 트랜잭션의 불필요한 취소를 줄이는 것이다. 셋째 다중버전이 아닌 단일버전상에서 수행하므로 시스템 효율을 높인다. 일반적으로 상위 등급 사용자와 하위 등급 사용자가 볼 수 있는 데이타를 여러 개 만들어 놓음으로써 데이타베이스 상에서의 비밀채널 생성을 방지하는 알고리즘이 많이 연구되어 왔다. 그러나 이러한 다중 버전 방식은 각 데이타 항목마다 복수개의 버전을 가지게 되므로 시스템 효율이 현저히 떨어지게 되고, 시스템 자원이 풍부하다는 가정하에만 효율적으로 사용할 수 있다. 그러나 오렌지 로킹 기법에서와 같이 오렌지 로크를 정의함으로써 단일 버전 상에서 비밀채널을 제거할 수 있게 되었다.

제안된 알고리즘은 다음과 같은 작업을 공통적으로 수행한다고 가정한다. 첫째, 모든 트랜잭션은 지역 작업 공간을 가진다. 둘째, 판독연산은 데이타베이스에서, 기록연산은 지역 작업공간에서 이루어진다. 셋째, 하향 판독 연산을 모두 성공적으로 수행한 경우에는 home-free point에 도달했다고 하고, 이후의 동급 데이타에 관한 연산은 strict 2PL 규칙에 의해 수행한다. 넷째, 모든 연산은 연산을 수행하기 전에 로크를 얻어야 한다. 즉, 판독일 경우 판독연산 시작 전에, 기록연산일 경우 지역작업 공간에 기록하기 전에 얻어야 한다.

다음은 이 알고리즘에서 사용되는 공통적인 기호들의 정의이다.

[표1] 사용되는 주요기호

기 호	내 용
RS(i)	트랜잭션 T_i의 판독집합
WS(i)	트랜잭션 T_i의 기록집합
WTS(X)	데이타 X에 기록한 완료된 트랜잭션들 중 가장 큰 타임스탬프 값
RTS(X)	데이타 X를 판독한 완료된 트랜잭션들 중 가장 큰 타임스탬프 값
TR(i,p)	트랜잭션 T_i가 데이타 항목 Dp를 판독할 때 Dp의 WTS 값
LCT(i)	트랜잭션 T_i가 완료할 수 있는 가장 큰 타임스탬프값
Bset(i)	T_i 이전으로 역방향 조정된 트랜잭션들의 집합

본 논문에서는 제한된 *-성질을 사용하므로 다양한 연산들을 다음과 같이 판독 연산 (동일등급), 하향 판독 연산, 기록 연산 (동일등급) 등의 세 가지 종류로 구분할 수 있다. 또한 이러한 연산을 수행시 발생되는 충돌을 보안등급에 따라 나누어서 생각해 보면 다음과 같다.

첫째, 동일 등급간의 판독연산과 기록연산(L(TH) = L(TR))일 경우는 다음과 같다. 즉, 충돌하는 트랜잭션들이 동일 등급일 경우, 기존에 사용되는 strict 2PL방식을 사용한다. strict 2PL의 로크호환성 표(lock compatability table)는 다음과 같다.

[표 2] $L(T_H) = L(T_R)$일 때의 로크 호환성 표

T_R \ T_H	Read	Write
Read	Y	N
Write	N	N

T_H : Lock Holder
T_R:Lock Requester

로크 호환성 표에서 알 수 있듯이 판독-판독 충돌일 경우 허용하나, 그 이외의 충돌은 허용하지 않으므로 해당 로크를 얻기까지 연산은 지연되어야 한다. 다음은 이 단계에서의 상세한 알고리즘이다.

① 스케줄러가 TM(transaction manager)으로부터 $p_i[x]$를 받으면, 스케줄러는 $pl_i[x]$가 이미 걸려있는 $ql_j[x]$와 충돌을 일으키는

지 검사한다. 만일 충돌을 일으킨 두 트랜잭션간의 보안등급이 $L(T_i) = L(T_j)$ 한 관계라면, $p_i[x]$는 트랜잭션 T_j가 로크를 얻을 때까지 지연된다. 만일 충돌을 일으키지 않는다면, 스케줄러는 $pl_i[x]$를 인가하고, $p_i[x]$를 DM(data manager)에 보낸다.

② $p_i[x]$가 판독 연산일 경우, TR(i,p)를 기록한다.

③ 스케줄러가 T_i의 로크를 인가했을 때, 적어도 DM으로부터 트랜잭션이 완료(commit)되었거나 취소(abort)되었다는 응답을 받기 전까지 로크를 해제할 수 없다.

④ 스케줄러가 트랜잭션 T_i에 대한 로크를 해제할 때 다음 사항을 검사한다.

a. T_i가 필요한 로크를 모두 얻었는지 검사한다.

b. T_i가 x를 참조하는 연산을 더 이상 수행하지 않을 것인지를 검사한다.

⑤ 트랜잭션 수행 완료 후에, LCT가 null이 아닌 경우에는 LCT값을 TS로 받고, null인 경우에는 완료시점을 TS값으로 부여받는다. 변경된 데이타 항목에는 RTS와 WTS 값을 완료시점으로 기록한다. Bset에 있는 트랜잭션들의 LCT를 TS-ε으로 부여한다.

둘째, 하향 판독일 경우에는 로크를 요구하는 트랜잭션과 이미 로크를 소요하고 있는 트랜잭션의 보안등급에 따라 다음 두 가지 경우로 나눌 수 있다.

가)로크를 소유하고 있는 트랜잭션의 보안등급이 로크를 요구하는 트랜잭션의 보안등급보다 낮을 경우($L(T_H) < L(T_R)$), 로크 호환성 표는 다음과 같다.

[표 3] $L(T_H) > L(T_R)$일 때의 로크 호환성 표

T_R \ T_H	Read	Write
Read	Y	Block
Write		

T_H : Lock Holder
T_R : Lock Requester

로크 호환성 표에서 색칠된 부분은 발생할 수 없는 상황을 뜻하며, 판독-판독 충돌의 경우 허용한다. 그러나 문제가 되는 것은 판독-기록 충돌이다. 즉 하위 등급 트랜잭션이 이미 기록 로크를 소유하고 있을 때, 상위 등급 트랜잭션이 동일한 데이타 항목에 대하여 판독 로크를 요구할 경우에는 하위 등급 트랜잭션이 모든 수행을 완료하고, 소유한 로크를 해제할 때까지 기다려야 한다. 즉 상위 등급 트랜잭션은 블록되어야 한다. 다음은 이 단계에서의 상세한 알고리즘이다.

① 스케줄러가 TM(transaction manager)으로부터 $p_i[x]$를 받으면, 스케줄러는 $pl_i[x]$가 이미 걸려있는 $ql_j[x]$와 충돌을 일으키는지 검사한다. 만일 충돌을 일으킨 두 트랜잭션간의 보안등급이 $L(T_i) > L(T_j)$한 관계라면, $p_i[x]$는 트랜잭션 T_j가 로크를 얻을 때까지 지연된다. 만일 충돌을 일으키지 않는다면, 스케줄러는 $pl_i[x]$를 인가하고, $p_i[x]$를 DM(data manager)에 보낸다.

② $p_i[x]$가 판독 연산일 경우, TR(i, x)를 기록한다.

③ 스케줄러가 T_i의 로크를 인가했을 때, 적어도 DM으로부터 트랜잭션이 완료(commit)되었거나 취소(abort)되었다는 응답을 받기 전까지 로크를 해제할 수 없다.

④ 스케줄러가 트랜잭션 T_i에 대한 로크를 해제할 때 다음 사항을 검사한다.

a. T_i가 필요한 로크를 모두 얻었는지 검사한다.

b. T_i가 x를 참조하는 연산을 더 이상 수행하지 않을 것인지를 검사한다.

⑤ 트랜잭션 수행과, LCT가 null이 아닌 경우에는 LCT값을 TS로 받고, null인 경우에는 완료시점을 TS값으로 부여받는다. 변경된 데이타 항목에는 RTS와 WTS 값을 완료시점으로 기록한다. Bset에 있는 트랜잭션들의 LCT를 TS-ε으로 부여한다.

나)로크를 소유하고 있는 트랜잭션의 보안등급이 로크를 요구하는 트랜잭션의 보안등급보다 높을 경우($L(T_H)>L(T_R)$), 로크 호환성 표는 다음과 같다.

[표 4] $L(T_H)>L(T_R)$일 때의 로크 호환성 표

T_R \ T_H	Read	Write
Read	Y	
Write	Y*	

T_H : Lock Holder
T_R : Lock Requester

Y* : 판독로크를 오렌지로크로 전환하고, 기록연산을허용.

로크 호환성 표에서 색칠된 부분은 발생할 수 없는 상황을 뜻하며, 판독-판독 충돌은 허용한다. 그러나 판독-기록 충돌일 경우 비밀채널을 발생시킬 수 있다. 즉, 상위 등급 트랜잭션이 판독 로크를 가지고 있는 데이타 항목에 대하여 하위 등급 트랜잭션이 기록 로크를 요구할 경우, 상위 등급 트랜잭션이 하위 등급 트랜잭션을 지연시키면 비밀채널이 생성된다. 그러므로 상위 등급 트랜잭션은 소유하고 있는 로크를 즉시 하위 트랜잭션에게 인가해주어야 한다. 다음은 이 단계에서의 상세한 알고리즘이다.

① 스케줄러가 TM(transaction manager)으로부터 $p_i[x]$를 받으면, 스케줄러는 pli[x]가 이미 걸려있는 $ql_i[x]$와 충돌을 일으키는지 검사한다.

② 충돌이 있을 경우, 두 트랜잭션간의 보안등급이 $L(T_i)<L(T_j)$한 관계라면, 상위 등급 트랜잭션 T_j의 판독로크 ql_j는 오렌지 로크(orange lock)로 전환한다.

③ $p_i[x]$가 판독 연산일 경우, TR(i,x)를 기록한다.

④ 충돌이 일어난 상위 등급 트랜잭션 T_j를 하위 등급 트랜잭션 T_i의 Bset에 넣고 하위 등급 트랜잭션 T_i에게 $pl_i[x]$를 인가한다.

⑤ 스케줄러가 T_i의 로크를 인가했을 때, 적어도 DM으로부터 트랜잭션이 완료(commit)되었거나 취소(abort)되었다는 응답을 받기 전까지 로크를 해제할 수 없다.

⑥ 하위 등급 트랜잭션이 모든 수행을 마치면, 다음 작업을 수행하고 트랜잭션을 완료한다.

⑦ 직렬화 순서를 재조정한다. ($T_H \rightarrow T_L$)

⑧ 스케줄러가 트랜잭션 T_i에 대한 로크를 해제할 때 다음 사항을 검사한다.

a. T_i가 필요한 로크를 모두 얻었는지 검사한다.

b. T_i가 x를 참조하는 연산을 더 이상 수행하지 않을 것인지를 검사한다.

⑨ 트랜잭션 완료시에 LCT가 null이 아닌 경우에는 LCT값을 TS로 받고, null인 경우에는 완료시점을 TS값으로 부여받는다. 변경된 데이타 항목에는 RTS와 WTS 값을 완료시점으로 기록한다. Bset에 있는 트랜잭션들의 LCT를 TS-ε으로 부여한다.

⑩ 블록 되었던 상위 트랜잭션이 수행이 다시 시작될 때 다음 사항을 검사한다.

```
if (TR(h,p) > LCT(TH))
then
        abort high level transaction;
endif
```

⑪ 스케줄러가 트랜잭션 T_i에 대한 로크를 해제할 때 다음 사항을 검사한다.

a. T_i가 필요한 로크를 모두 얻었는지 검사한다.

b. T_i가 x를 참조하는 연산을 더 이상 수행하지 않을 것인지를 검사한다.

⑫ 트랜잭션 수행과, LCT가 null이 아닌 경우에는 LCT값을 TS로 받고, null인 경우에는 완료시점을 TS값으로 부여받는다. 변경된 데이타 항목에는 RTS와 WTS 값을 완료시점으로 기록한다. Bset에 있는 트랜잭션들의 LCT를 TS-ε으로 부여한다.

다음 [예5]를 이용하여 제시된 알고리즘이 수행되는 과정을 보이겠다.

[예 5] T={T1, T2, T3, T4}

	1	2	3	4	5	6	7	8	9	10	11	12	13
TS(T_1)	$r_1(x)$	$r_1(y)$	$r_1(z)$									$r_1(t)$	c_1
S(T_2)							$r_1(x)$			$w_2(q)$	c_2		
C(T_3)				$w_3(y)$	$w_3(z)$	c_3							
C(T_4)								$w_4(t)$	c_4				

step 1. t4의 시점에서 T_3에 의해 T_1은 데이타 항목 y에 대해서 판독 로크를 오렌지 로크로 전환한다.

step 2. T_1을 T_3의 Bset에 넣는다.

step 3. t5의 시점에서 T_3에 의해 T_1은 데이타 항목 z에 대해서 판독 로크를 오렌지 로크로 전환한다.

step 4. t6의 시점에서 T_3는 완료할 때 다음 작업을 수행한다.

1) $T_1 \rightarrow T_3$로 직렬화 순서를 조정한다.
2) T_3의 LCT값을 t6-ε값으로 부여한다.
3) TS(T_3) = t6으로 부여받는다.

step 5. t8의 시점에서 T_4에 의해서 T_2는 데이타 항목 t에 대해서 판독 로크를 오렌지 로크로 전환한다.

step 6. T_2를 T_4의 Bset에 넣는다.

step 7. t9의 시점에서 T_4는 완료할 때 다음 작업을 수행한다.

1) 직렬화 순서를 $T_2 \rightarrow T_4$로 재조정한다.
2) T_2의 LCT값을 t9-ε값으로 부여한다.
3) TS(T_4) = t9로 부여받는다.

step 8. t11의 시점에서 T_2는 완료하고 LCT값이 null이 아니므로, LCT(T_2)값을 TS(T_2)값으로 부여받는다.

step 9. t12의 시점에서 LCT값이 null이 아니므로 다음 사항을 체크한다.
TR(1, t) 〉 TS(T_1) 이므로 T_1은 취소된다.

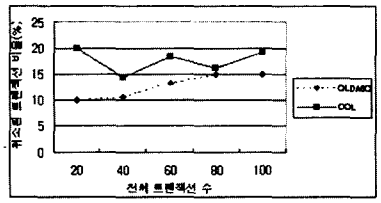
그러므로 주어진 수행기록은 T_3, $T_2 \rightarrow T_4$ 의 직렬화 순서로 수행되고 트랜잭션 T_1은 제안된 알고리즘에 의해 취소됨을 알 수 있다. 위에서 수행된 결과로 알 수 있듯이, 트랜잭션 T_2는 트랜잭션 T_4에 의해 취소되어야 하나 직렬화 순서를 재조정해 줌으로써 불필요한 재수행을 제거하였다.

4. 비교분석 및 성능평가

본 논문에서 제안한 알고리즘의 성능 평가를 위해 다음과 같은 두가지 측정 기준을 가지고 성능 평가를 수행한다. 서론에서 언급한 것과 같이, 제안된 알고리즘은 기존의 보안 알고리즘이 해결하지 못한 두 가지 문제 즉, 상위 등급 트랜잭션이 하위 등급 트랜잭션에 의해 반복적으로 수행이 지연되는 상위 등급 기아현상과 상위 등급 트랜잭션의 재수행 비용

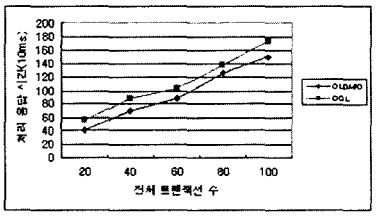
의 증가로 인한 병행성 감소문제에 중점을 두었다.

[그림 1]은 트랜잭션의 수가 증가할 때, 그에 따라 변화하는 취소되는 트랜잭션의 비율을 보인 것이다. 그림에서 알 수 있듯이, 제안된 알고리즘은 직렬화 순서의 동적인 재조정 기법을 사용하여 상위 트랜잭션의 불필요한 재시작을 줄였기 때문에, 상위 트랜잭션과 하위 트랜잭션이 충돌을 일으켰을 경우, 일방적으로 상위 트랜잭션을 취소하는 낙관적 오렌지 로크 알고리즘에 비해 더 좋은 성능을 보이는 것을 알 수 있다.



[그림 1] 트랜잭션 증가에 따른 취소된 트랜잭션 비율의 비교

[그림 2]는 트랜잭션 증가에 따른 처리 응답 시간을 비교한 것이다. 제안된 알고리즘은 불필요한 재시작 횟수를 줄였기 때문에 재시작에 따른 오버헤드 또한 감소하였다. 따라서 전체 처리 응답 시간이 불필요한 재시작에 따른 많은 오버헤드를 안고 있는 낙관적 오렌지 로크 알고리즘에 비해 좋은 성능을 보이는 것을 알 수 있다.



[그림 2] 트랜잭션 증가에 따른 처리 응답시간의 비교

5. 결 론

다단계 보안 시스템(multilevel Secure System : MLS)은 시스템이 사용자를 위한 하나 이상의 보안 인가 등급(security clearance level)과 시스템 내의 데이타들을 위한 하나 이상의 분류 등급(classification level)을 가진 시스템이다. 다단계 보안 데이타베이스 관리 시스템(MLS/DBMS)은 다단계 보안 시스템의 접근제어를 데이타베이스 관리 시스템에 적용한 것이다. 이러한 시스템의 목적은 기밀 정보를 인가 받지 않은 사용자로부터 보호하는 것이다.

지금까지 다단계 보안 데이타베이스 시스템에서 보안을 유지하면서 성능을 향상시키기 위한 병행수행 기법들에 대해서 살펴보았다. 이미 연구되어온 알고리즘들은 로킹이나 타임스탬프 기법을 변형하여 비밀채널을 제거함으로써 보안을 유지하는데 주안점을 두었기 때문에 다음과 같은 문제점이 대두되었다.

즉, 상위 등급 트랜잭션이 하위 등급 트랜잭션에 의해 반복적으로 지연되는 기아현상을 유발시켰다. 상위 등급 트랜잭션의 지연 현상은 상위 등급 트랜잭션이 최근에 변경된 데이타 값을 읽을 수 있다는 장점을 가질 수 있지만, 상위 등급 트랜잭션들이 무한정 지연되어서는 안되기 때문에 등급간의 충돌을 적절히 조절하여야 한다. 그렇게 함으로써 보안을 유지하는 동시에 상위 등급 트랜잭션에게도 보다 많은 수행기회가 주어져야 한다.

이러한 문제를 해결하기 위해 본 논문에서는 직렬화 순서의 동적인 재조정 방식을 이용하여 상위 등급 트랜잭션의 기아현상을 완화하고, 기존의 보안 스케줄러가 데이타에 다중 버전을 둠으로써 보안을 유지한 것과는 달리 단일버전상에서 수행하게 함으로써 트랜잭션의 병행성 정도를 높일 수 있는 알고리즘을 제안하였다. 제안한 알고리즘이 병행수행 상에

서 직렬성을 만족함을 증명하였고, 보안성을 만족함을 보였다.

참고문헌

[1] J. McDermott and S. Jajodia, "Orange Locking: Channel-Free Database Cuncurrency Control via Locking," IFIP WG 11.3 Worlshop in Database Security, August, pp. 267-284, 1992.

[2] K. Lam, K. Lam and S. Hung, "Real-Time Optimistic Concurrency Control Protocol with Dynamic Adjustment of Serialization Order," IEEE Symposium on Real-Time Technology and Applications, pp. 174-179, May 1995.

[3] J. Lee and S.H. Son, "Using Dynamic Adjustment of Serialization Order for Real-Time Database Systems," Proceedings of 14th IEEE Real-Time Systems Symposium, pp. 66-75, December 1993.

[4] S. H. Son, S. Park, and Y. Lin, "An Integrated Real-Time Locking Protocol," Proceedings of 8th International Conference on Data Engineering, pp. 527-534, Feburary 1992.

[5] J. McDermott, S. Jajodia, and R. Sandhu, "A Single-level Scheduler for Replicated Architecture for Multilevel Secure Databases," Proceedings of 7th Annual Computer Security Application Conference, pp. 2-11, 1991.

[6] S. Castane, M. Fugini, G. Martella, and P. Samarati, Database Security, Addison-Wesley, 1995.

[7] S. Jajodia, L. V. Mancini, and I. Ray, "Secure Locking Protocols for Multilevel Database Management Systems," IFIP TC11/WG11.3 Tenth International Conference on Database Security, pp. 177-194, July, 1996.

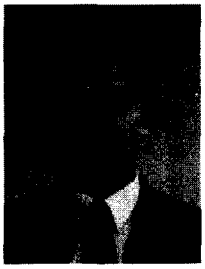
[8] C. Park and S. Park, "Priority-Driven Secure Multiversion Locking Protocol for Real-Time Secure Database Systems," Proceedings of the 11th Annual IFIP WG 11.3 Working Conference on Daatabase Security, pp.200-211, August, 1997.

[9] 김홍숙, 박석, "실시간 데이타베이스 시스템에서 상대적 시간 일관성 만족을 위한 낙관적 병행수행 제어" 한국 정보과학회 논문지 24권 5호, pp. 528-538, 1997.

[10] 윤인호, 박석, "실시간 데이타베이스 시스템의 혼합 트랜잭션 환경에서 직렬화 순서의 동적인 조정 방식을 이용한 병행수행 제어" 동계 데이타베이스 학술대회 논문집 13권 1호 pp. 151-156, 1997.

[11] 양지혜, 박석, "다단계 보안 데이타베이스 시스템에서 상위 등급의 기아현상을 제거한 병행수행 제어" 동계 데이타베이스 학술대회 논문집 12권 1호 pp. 55-60, 1996.

□ 著者紹介



박 석

1978년 서울대학교 계산통계학과(학사)
1980년 한국과학기술원 전산학과(석사)
1983년 한국과학기술원 전산학과(박사)
1983년 ~ 현재 서강대학교 컴퓨터학과 정교수
1989년 ~ 1990년 미국 버지니아 대학 교환교수

※ 주관심분야 : 데이타베이스 보안, 보안평가, 컴퓨터보안, 실시간 시스템, 데이타베이스 통합

김 명 은

1996년 숭실대학교 소프트웨어 공학과(학사)
1998년 서강대학교 전자계산학과(석사)
1998년 ~ 현재 동양시스템하우스에 재직

※ 주관심분야 : 다단계 보안 데이타베이스, 실시간 데이타베이스, 데이타 웨어하우징