다단계 보안 데이터베이스 시스템에서 병행수행 제어의 직렬화 순서를 재조정하기 위한 요청 2단계 로킹기법

이승수°, 조진성, 정병수
경희대학교 컴퓨터 공학과
sakspin@dblab.kyunghee.ac.kr

Request Two-Phase Locking Method for Series Sequence Re-adjustment of Concurrency Control in Multi-Level Secure DBMS

Seungsoo Lee°, Jinsung Cho, Byungsoo Jeong
Dept. of computer Engineering, Kyung Hee Univ

요 약

다단계 보안데이터베이스 시스템에서 기본적인 병행수행 제어 기법들은 비밀채널과 교착상태등과 같은 문제들이 발생하였다. 이에 직렬화 순서를 동적으로 재조정함으로서 해결하려는 방안이 있었지만, 알고리즘의 복잡성으로 인해 오버헤드와 많은 수행시간이 필요하게 되었고, 이에 따라 많은 양의 시스템 자원과 높은 사양의 시스템을 요구하게 되었다. 또한 이러한 방법은 다중 버전을 사용함으로서 추가적인 관리비용이 높게 되었고, 각각의 트랜잭션이 지연 및 재수행이란 불필요한 과정을 반복하게 되었다. 따라서 본 논문에서는 제안한 알고리즘은 데이터베이스의 용도에 맞게 직렬화 순서를 보장하여 스케줄을 관리하는 요청 2단계 로킹기법(Request Two-phase Locking)으로서 이는 2단계 로킹기법의 기본원리에 요청로크를 사용함으로 보다 효율적으로 병행제어를 할 수 있다. 여기서 요청로크는 각각의 트랜잭션 스케줄에 로크획득 및 해제를 병행수행제어의 필요에 따라 유동적으로 할 수 있으며, 읽기로크, 쓰기로크, 요청로크라는 3가지 로킹모드를 통해 대처방안을 마련함으로서, 충돌을 방지하며, 충돌연산의 특성에 따라 직렬화 순서를 동적으로 조정함으로 블록킹을 막는 병행제어를 응용하여 병렬성을 유지한다.

1. 서 론

다단계 보안시스템이란 사용자를 위한 하나이상의 분류등급(security clearance level)과 시스템 내의 데이터들을 위한 하나이상의 분류등급(classification level)을 가진 시스템을 말하며, 이러한 등급을 이용하여 기밀정보의 노출을 제어하는 기법들을 제공한다. 다단계 보안 데이터베이스 관리 시스템(MLS/DBMS)[3]은 이러한 다단계 보안 시스템의 접근제어를 데이터베이스 관리 시스템에 적용한 것이다. 이러한 시스템의 목적은 기밀 정보를 인가 받지 않은 사용자로부터 보호하는 것이다.

다단계 데이터베이스 시스템(MLS/DBMS)에서 중요한 문제는 비밀채널(covert channel)의 발생이다. 병행수행 로크(lock)는 데이터베이스 시스템에서 전형적인 비밀채널의 예로 사용되어 왔다. 로킹(locking)은 많은 시스템에서 사용되는 기본적 병행수행 제어기법이지만, 다단계 보안 데이터베이스 시스템에서 사용하기에는 비밀채널의 형성이라는 문제점을 갖고 있다. 이러한 문제점을 해결하기 위한 비밀채널 제거방안이 다양하게 제시되었는데, 오렌지 로킹(orange locking)기법[3], 보안 2단계 로킹 기법[7] 등이 있다. 하지만 이와 같은 방법은 비밀채널 제거에 초점을 맞추었으므로 하위 등급 트랜잭션에 의해 상위 등급 트랜잭션의 수행이 반복적으로 지연되는 상위 등급 트랜잭션의 기아현상이 일어나거나, 직렬성을 만족시키지 못하는 문제가 발생된다. 이에 트랜잭션의 병행수행을 보장하며, 상위 등급 트랜잭션의 기아 현상을 완화하고, 보안 스케줄러가 데이터에 단일버전 상에서 수행하게 함으로서 트랜잭션의 병행성을 높일 수 있는 방법을 요구하게 되었다.

본 논문의 구성은 다음과 같다. 우선 2장에서 기존의 병행수행제어를 위한 여러 가지 기법들을 알아보고 문제점을 지적한다. 3장에서는 본 논문에서 제안한 읽기로크(read locked), 쓰기로크(write locked), 요청로크(요청 locked), 로크해제(unlocked)를 사용함으로써, 직렬화 순서를 조정하고, 병행제어에 따른 교착과 기아현상을 최대한 줄이고자 하는 스케쥴링 기법에 대해 설명하고 4장에서는 결론과 향후 연구를 논의한다.

2. 관련 연구

2.1 오렌지 로킹 알고리즘

오렌지 로킹 알고리즘은 지역 작업공간(local workspace)을 필요로 하는데, 지역 작업 공간은 각각의 트랜잭션마다 생성되는 임시 작업 공간으로서, 하향 판독(read-down)연산은 모두 지역 작업공간에서 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 검사점으로서, 트랜잭션 T가 home-free point에 도달했다는 것은 트랜잭션 T에 의해 하향 판독되는 모든 데이터 항목 x가 판독 로크를 얻어서 판독을 성공적으로 수행하거나, 오렌지 로크를 얻은 후 판독을 모두 성공적으로 수행했다는 것을 의미한다. 이 기법은 기존의 기법과는 달리 오렌지 로크라는 새로운 로크를 정의함으로서 상위 등급 트랜잭션의 불필요한 하향 판독 연산의 재수행 횟수를 줄인 점이 특징이다.

2.2 컬러링(Coloring)기법

병행 수행 알고리즘의 정확성 기준으로 직렬성이 아닌 MLS-직렬성(MLS-Serializability)을 사용하였다.
이 기법은 시스템 내에 보안등급이 완전 순서화(totally ordered)되어 있다는 가정 하에서만 수행 가능하다. 이러한 가정하에 수행되었을 때에만 MLS-직렬성을 만족함을 보장할 수 있다. 컬러링 기법의 알고리즘은 실제 컬러링 기법을 구현하기에는 너무 많은 자료구조들을 필요로 한다. 구현상의 오버헤드 또한 컬러링 기법의 제약점으로 작용한다. 즉 너무 많은 제약을 가진다.

2.3 타임스탬프 순서 기법

직렬성 순서를 결정하는 다른 방법의 하나로 진행될 트랜잭션들 중의 순서를 미리 선택하는 방법으로 가장 보편적인 방법이 타임스탬프 순서(time stamp ordering)기법을 이용하는 것이다.
시스템내의 각 트랜잭션에 고정된 타임스탬프를 트랜잭션이 수행되기 전에 데이터베이스 시스템에 의해 할당한다. 트랜잭션 Ti에 타임스탬프가 할당되고 새로운 트랜잭션 T_j가 시스템에 들어오면 $TS(T_i) < TS(T_j)$가 된다.
트랜잭션의 타임스탬프는 직렬성 순서를 결정한다. 이 기법을 구현하기 위해 각 데이터 항목 Q에 대하여 두 개의 타임스탬프 값을 사용한다.[7]

· W-타임스탬프(Q): write(Q)를 성공적으로 실행한 트랜잭션 중에서 가장 큰 타임스탬프
· R-타임스탬프(Q): read(Q)를 성공적으로 실행한 트랜잭션 중에서 가장 큰 타임스탬프

이들 타임스탬프는 새로운 읽기/쓰기 명령이 실행될 때마다 갱신된다. 타임스탬프 순서 규약은 어떤 충돌하는 read와 write연산이 타임스탬프 순서로 실행되도록 보장한다. 이 규약은 아래와 같이 운용된다.

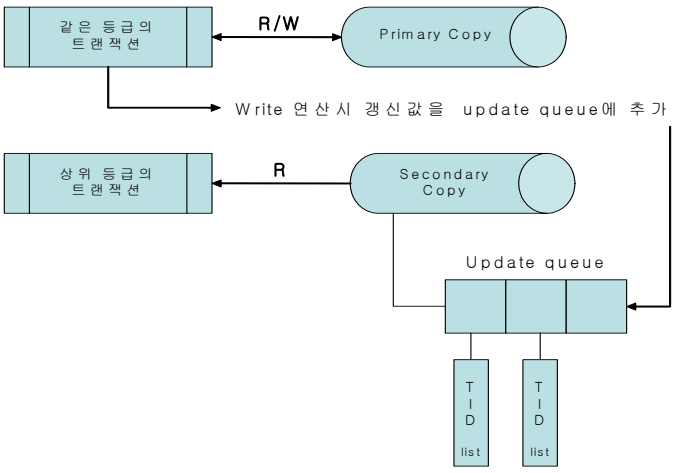
① 트랜잭션 T가 read(Q)를 낸다고 가정하자.
· TS(T) < W - 타임스탬프(Q)이면, T는 이미 덧씌어진 Q의 값을 읽을 필요가 있음을 의미한다. 따라서 read 연산은 거절되며, T는 복귀된다.
· TS(T) >= W - 타임스탬프(Q)이면, read 연산이 실행되며, R-타임스탬프(Q)는 R-타임스탬프(Q)와 TS(T)중에 최대값으로 된다.

②트랜잭션 T가 write(Q)를 낸다고 가정하자.
· TS(T) < R-타임스탬프(Q)이면, T 가 생성하려고 하는 Q의 값은 전에 필요로 하던 값이어서, 결코 생성되지 않으리라고 가정된 것임을 의미한다. 따라서 write연산은 거절되며 T는 복귀된다.
· TS(W) < W-타임스탬프(Q)이면, T가 출력하려고 하는 Q의 값은 쓸데없는 값임을 의미한다. 따라서 write연산은 거절되고 T는 복귀된다.
· 위의 두 경우가 아니며, write연산은 실행되며, W- 타임스탬프(Q)는 W-타임스탬프(Q)와 TS(T)중에 최대값으로 된다.

read와 write연산을 낸 결과로서, 병행 수행 제어 기법에 의해 복귀된 트랜잭션 T는 새로운 타임스탬프를 할당받고 다시 시작한다. 타임스탬프 순서 규약은 충돌 직렬성을 보장한다. 이것은 충돌하는 연산들이 타임스탬프 순서로 처리된다는 사실로부터 알 수 있다. 이 기법은 트랜잭션이 결코 기다리지 않기 때문에 교착상태(Dead Lock)가 발생하지 않는다.

2.4 보안 실시간 2단계 로크 규약

기본적인 2PL을 이용한 방법 중에 Primary Copy와 Secondary Copy 2개의 버전을 이용하여 첫 번째 버전에는 트랜잭션과 같은 등급의 데이터에 대한 기록, 판독 연산을 수행하고 두 번째 등급에서는 상위 등급의 트랜잭션이 하위 등급의 데이터에 대한 하향 판독 연산을 수행한다.[6]
이 방법의 시스템 구성도는 아래 그림과 같다.



[그림 1] Two-phase lock regulation diagram

기본적인 2단계 로크 규약은 데이터베이스 시스템의 동시성제어에 가장 많이 사용되는 방법이다. 현재 로크를 가지고 있는 트랜잭션은 다른 트랜잭션과의 데이터 충돌에 의해 취소되지 않지만, 새롭게 로크를 얻으려고 하는 트랜잭션은 현재 로크를 가지고 있는 트랜잭션이 로크를 해제 할 때까지 기다려야 한다. 어떤 경우에는 요청하는 트랜잭션은 취소되거나, 약간의 지연 뒤에 다시 로크를 얻는 경우가 발생한다.
이런 성질 때문에 기본적인 2단계 로크 규약은 낮은 등급의 트랜잭션이 높은 등급의 트랜잭션과의 데이터

충돌에 의해 지연되거나 취소되지 말아야 한다는 성질을 위배하므로 보안 데이터베이스에서는 적합하지 않다.
따라서 SRT 2PL은 다음의 구조를 가진다.

① 각 데이터 object는 두개의 복사 본을 유지한다.
② 첫째 버전은 보통의 데이터 object를 유지하고, object와 같은 등급의 트랜잭션에 의한 읽기/쓰기 접근을 허용한다.
③ 둘째 버전은 데이터 object보다 상위 등급의 트랜잭션에 의한 읽기 접근을 허용한다. 이 버전은 일반적인 데이터 object와 갱신을 위한 단일 큐의 두 데이터 구조를 유지한다. 큐는 첫 째 버전 상에서 갱신되었지만, 둘 때 버전 상에서는 아직 수행되지 않은 값을 가지고 있다. 각 갱신 큐 내의 데이터 엔트리에는 읽고자 하는 트랜잭션의 ID값을 가지고 있다.
이 기법의 로크를 관리하는 방법과 연산은 다음에 따른다.

· 각 트랜잭션은 수행 전에 필요로 하는 모든 로크를 획득하여야 한다.

· 각 버전 상에서 수행되는 로크 규약은 2단계 로크 규약을 사용한다. 다시 말하면 로크 충돌을 해결하기 위하여 각 object의 버전은 독립적으로 취급된다.

· 두 번째 버전은 오직 상위 등급의 트랜잭션에 의한 읽기 접근만 있으므로 상위 등급의 트랜잭션간의 데이터 충돌은 없다.

따라서 두 번째 버전의 읽기 로크에 대한 요구에 따른 지연은 발생하지 않는다.

같은 등급의 트랜잭션에 의해 갱신 큐의 내용이 간접적으로 변화한다 해도 이 트랜잭션은 두 번째 버전에 쓰기 로크를 얻을 필요는 없다.

3. 요청 2단계 로킹 기법

본 장에서는 요청 2단계 로크에 관한 핵심 내용을 설명한다.

3.1 요청 로크의 개요

요청 2단계 로킹기법(request two-phase locking method)에서 요청로크는 일종의 배타적 로크이다. 일반적인 2단계 로킹(two-phase loking)은 확장단계(growing phase)와 수축단계(shrinking phase)를 통해서 항목들에 대한 로크를 획득하거나 반대로 해제만을 할 수 있다. 이는 직렬성을 보장해 줄 수 있지만, 병행제어 면에서 많은 수행시간과 지연이 발생하게 된다. 그러나 요청로크를 통해 직렬화된 스케줄을 상황에 따라 로크해제 및 획득 가능하게 함으로서 효율적인 병행수행을 할 수가 있게 된다. 이는 엄격한 2단계 로킹기법과 달리 스케줄러의 필요에 따라 유동적인 스케줄 관리가 되므로 교착상태가 발생하지 않으면서 연쇄적인 롤백(roll back)을 해소하기 위함이다.

3.1 Request Lock를 사용한 다중버전 기법

항목에 대해 읽기(read), 쓰기(write), 요청(certify)의 3가지 로킹모드
· LOCK(X)의 상태:
① 읽기 로크(read locked)
② 쓰기 로크(write locked)
③ 요청 로크(요청 locked)
④ 로크 해제(unlocked)

아래의 표는 각각의 로크간에 호환성을 나타냈으며 하나의 트랜잭션 스케줄이 request일 경우 또 다른 트랜잭션 스케줄에서 request를 요구함으로 효율적인 병행제어를 가능하게 하고, 이 전의 트랙잭션 스케줄을 완료할 수가 있다.

	READ	WRITE	REQUEST
READ	yes	yes	no
WRITE	yes	no	no
REQUEST	no	no	yes

[표 1] 로크 호환성 테이블

3.2 요청 로크를 사용한 다중버전 2단계 로킹

① 각 항목은 x에 대하여 두개의 버전을 둘 수 있다.
② 하나의 버전은 항상 완료된 트랜잭션이 쓰기를 수행한 것이어야 한다.
③ 두 번째 버전 X′은 어떤 트랜잭션 T가 항목 X에 대해 쓰기 로크를 획득할 때 생성된다.
④ T가 쓰리 로크를 보유하고 있는 동안에도 다른 트랜잭션들은 완료된 버전 X를 계속 읽을 수 있다.
⑤ 트랜잭션 T는 완료된 버전의 X의 값에 영향을 미치지 않고 원하는 대로 X′의 값을 갱신할 수 있다.
⑥ 트랜잭션 T가 완료할 준비가 되면 완료하기 전에 현재 TM기 로크를 보유하고 있는 모든 항목들에 대하여 **요청로크**를 획득해야 한다. 요청로크는 읽기 로크와 호환성이 없다. 따라서, 트랜잭션 T가 요청로크를 얻기 위해서는 T가 쓰기로크를 보유하고 있는 항목들을 읽고 있는 다른 트랜잭션들의 로크가 해제될 때까지 트랜잭션 T의 완료가 지연되어야 한다.
⑦ 요청로크를 획득하면 그 데이터 항목의 완료된 버전 X는 버전 X′의 값으로 설정되고 버전 X′은 폐기되며 그리고 나서 요청로크가 해제된다.

3.3 스케줄의 충돌 직렬가능성 테스트 제안

직렬가능한 스케줄에서는 어떠한 정확성도 잃지 않으면서 동시 실행의 장점을 얻을 수 있다. 하지만 직렬가능성을 보장하기 위해 스케줄의 연산들이 어떻게 인터리빙 될 것인가를 미리 결정하기는 매우 어렵다. 다음은 세 개의 트랙션들을 두 가지 스케줄 방법으로

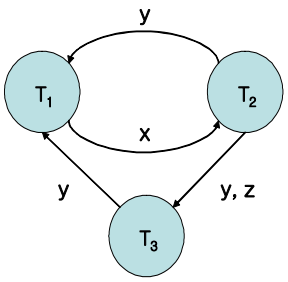
직렬가능한지를 보인 예이다.

· 트랜잭션 T1, T2, T3

T1: read(x), write(x), read(y), write(y)
T2: read(x), read(y), write(y), read(x), write(x)
T3: read(y), read(z), write(y), write(z)

① 연산의 충돌과 인터리빙이 없는 경우

T1	T2	T3
	read(z)	
	read(y)	
	write(y)	
		read(y)
		read(z)
read(x)		
write(x)		write(y)
		write(z)
	read(x)	
read(y)		
write(y)		
	write(x)	



스케줄 E에 대한 선행 그래프

동치인 직렬 스케줄 없음.

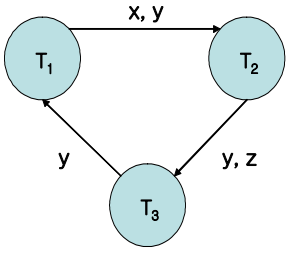
cycle 1: x(T1→T2), y(T2→T1)

cycle 2: x(T1→T2), yz(T2→T3), y(T3→T1)

[표 2] 직렬 가능성 테스트 예(스케줄 E)

② 연산의 충돌과 인터리빙이 있는 경우

T1	T2	T3
		read(y)
		read(z)
read(x)		
write(x)		
		write(y)
		write(z)
	read(z)	
read(y)		
write(y)		
	read(y)	
	write(y)	
	read(x)	
	write(x)	



스케줄 F에 대한 선행 그래프

동치인 직렬 스케줄 있음.

T3 → T1 → T2

[표 3] 직렬 가능성 테스트 예(스케줄 F)

4. 결 론

지금까지 다단계 보안 데이터베이스 시스템에서 보안을 유지하면서 성능을 향상시키기 위한 병행수행 기법이 여러 알고리즘과 방법을 통해 제시되어 왔다. 각기 제시되었던 여러 방법이 로킹이나 타임스탬프 기법을 변형하여 비밀채널을 제거함으로써 보안을 유지하는데 주안점을 두었기 때문에 다음과 같은 문제점이 대두되었다.

즉, 상위 등급 트랜잭션들이 하위 등급 트랜잭션에 의해 반복적으로 지연되는 기아현상을 유발시켰다. 상위 등급 트랜잭션의 지연 현상은 상위 등급 트랜잭션이 최근에 변경된 데이터 값을 읽을 수 있다는 장점을 가질 수 있지만, 상위 등급 트랜잭션들이 무한정 지연되어서는 안 되기 때문에 등급간의 충돌을 적절히 조절하여야 한다. 그렇게 함으로서 보안을 유지하는 동시에 상위 등급 트랜잭션에게도 보다 많은 수행기회가 주어져야 한다. 이러한 문제를 해결하기 위해 데이터베이스의 용도에 따른 직렬화 순서를 보장한 요청 2단계 로킹기법을 통해 스케줄을 관리하는 대처방안을 마련하고 이를 통해 충돌을 예방하며, 충돌연산의 특성에 따라 직렬화 순서를 동적으로 재조정하는 방식을 응용하여 상위 등급 트랜잭션의 기아현상을 완화 할 수 있는 병행제어 기법을 연구했다.

참 고 문 헌

[1] 윤인호, 박석, “실시간 데이터베이스 시스템에서 상대적 시간 일관성 만족을 위한 낙관적 병행수행 제어”, 한국 정보과학회 논문지 24권 5호, pp. 528-538, 1997

[2] 양지혜, 박석, “다단계 보안 데이터베이스 시스템에서 상위 등급의 기아현상을 제거한 병행수행 제어”, 동계 데이터베이스 학술대회 논문집 12권 1호 pp. 55-60, 1996

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

[4] T. F. Keefe and W. T. Tsai, "Multiversion Concurrency Control for Multilevel Secure Database Systems," Proceedings of the 10th IEEE Symposium on Research in Security and privacy, pp. 369-383, May 1990

[5] Kuma, V. "Performance of Concurrency Control Mechanism in Centralized Database Systems", Prentice-Hall, 1986

[6] Data Base System concept(4th Ed), Korth/Silberschatz, 2002

[7] Fundamentals of Database Systems(3th Ed) Elmasri/Navathe, 2000

[8] Database Systems, The Complet Book, Garcia-Molina/Ulman/Widom, 2002